Apunte 12 - Introducción al Diseño de APIs

Protocolo HTTP

El Protocolo de Transferencia de Hipertexto (HTTP) es ampliamente utilizado en internet, debido a que es el protocolo fundamentalmente empleado por las páginas web. Cuando se origina el protocolo HTTP, su concepción original era la de proporcionar un medio para la transferencia de contenido estático. En otras palabras, estaba diseñado para manejar información que se encontrara almacenada en archivos expuestos a través de servidores web. De esta forma, el protocolo ofrece un mecanismo eficaz para la obtención de tales archivos. Inicialmente, dichos archivos consistían mayormente en documentos de texto, aunque con la introducción del lenguaje HTML (Hypertext Markup Language), estas páginas web comenzaron a adoptar elementos de diseño que los navegadores podían interpretar y presentar de manera coherente y estructurada.

Además de su papel inicial en la entrega de contenido estático almacenado en archivos, el HTTP se adaptó para satisfacer la demanda de consumir datos que no necesariamente estaban almacenados en archivos convencionales. En lugar de limitarse únicamente a archivos físicos, el protocolo se transformó para permitir la obtención de datos generados dinámicamente en tiempo de ejecución. Estos datos podían ser generados al instante por el servidor en respuesta a una solicitud, sin requerir su previo almacenamiento en la infraestructura del servidor.

URI

Los recursos dentro del contexto de la arquitectura web están dotados de identidad, denotada por un identificador único. Este identificador de origen se forma a partir del nombre del servidor y un nombre distintivo del recurso en cuestión. En los sitios web los recursos adoptan diversas formas como archivos HTML, scripts JavaScript, hojas de estilo CSS, imágenes y más. Cada uno de estos elementos dispone de su propia denominación exclusiva. La concatenación del nombre del servidor y el identificador del recurso en forman el Uniform Resource Identifier (URI), que identifica de manera única y precisa el recurso de forma única en internet.

La característica principal es que el protocolo HTTP no establece una imposición de que las URI deban necesariamente hacer referencia o identificar entidades físicamente almacenadas en los servidores. En realidad, las URI pueden representar y señalar una amplia gama de recursos, independientemente de si están o no almacenados. Al recibir una solicitud hacia una URI, el servidor tiene la potestad de determinar la acción a seguir.

En una situación donde el servidor identifica una URI que apunta a un recurso físicamente almacenado, puede responder al cliente con el contenido de ese recurso. Esto es especialmente cierto en el caso de archivos estáticos, como páginas HTML, imágenes o archivos de estilo. Pero también existen escenarios en los cuales el servidor, al interpretar la petición recibida, opta por generar datos en tiempo de ejecución como respuesta.

Por ejemplo, en el caso de necesitar exponer datos provenientes de una base de datos. Suponiendo una base de datos de clientes, donde cada cliente está identificado por una clave primaria única, el servidor puede

establecer que las URI con el formato "servidor/clientes/5" no hacen referencia a archivos físicos, sino a los datos almacenados en la base de datos del cliente cuya clave primaria es 5.

De una manera similar, si para referenciar a facturas de ventas, puede el servidor establecer que las uri con formato "servidor/facturas/1234" corresponden con los datos de la factura número 1234, mientras que la uri "servidor/facturas/1234/detalles" identifican al conjunto de ítems de detalle de tal factura y que "servidor/facturas/1234/detalles/2" identifica al segundo ítem de la misma.

Verbos de HTTP

Los verbos que conforman el conjunto de comandos proporcionados por el protocolo son esenciales en el proceso de interacción entre un software cliente y un servidor. El protocolo HTTP se establece como una serie de comandos que definen las acciones que pueden ser ejecutadas en este contexto. En total, se identifican entre el HTTP original y extensiones aproximadamente 40 comandos, de los cuales sólo se emplearán aquellos necesarios para cada situación específica.

Estos comandos, representados como verbos, consisten en las operaciones que un cliente puede solicitar al servidor. Cada uno de ellos denota una acción precisa y define cómo debe ser tratada por el servidor en el momento de recibir la petición. Cada petición generada por un software cliente estará compuesta por dos elementos: el verbo o nombre de la acción que se desea realizar, y la URI, que designa el recurso sobre el cual se pretende ejecutar dicha acción. El servidor, al recibir una petición, interpreta el verbo HTTP como la operación que debe llevar a cabo sobre el recurso señalado por la URI.

Los comandos más utilizados al desarrollar APIs son los siguientes:

GET

Simboliza la acción de obtener un recurso o su representación. Este comando le comunica al servidor el deseo del cliente de obtener una representación actual del estado del recurso. Para una URI como o "clientes/5", el verbo "GET" transmitirá al servidor la intención de acceder al estado más reciente del cliente número 5.

Como respuesta, el servidor determinará cómo representar ese estado y lo presentará al cliente. Esta representación podría implicar una consulta a la base de datos que almacena información sobre los clientes y posteriormente generar una respuesta que refleje la situación actual del cliente, en algún formato acordado con el cliente, tal como JSON.

POST

El verbo "POST" indica la acción de que un cliente envíe un recurso nuevo al servidor. En el contexto de una aplicación REST, el verbo "POST" se utiliza para la inserción de nuevos datos en una base de datos. Al efectuar una petición "POST", el cliente envía la URI que identifica el recurso que se busca crear. A continuación, se proporcionan los datos correspondientes al nuevo recurso. El servidor, tras recibir esta información, evalúa su validez y en tal caso agrega un nuevo elemento con los datos suministrados. En una base de datos relacional, esto se reflejaría como la inserción de una nueva fila en una tabla. El verbo "POST" actúa como una herramienta para la creación de recursos nuevos y la ampliación de la base de datos.

En el caso de las bases de datos relacionales en las que una tabla posea como clave primaria un dato auto-numérico que se genera a causa de la inserción de una fila, el cliente no puede conocer el valor de dicha clave en el momento de realizar el POST. En ese caso, la URI no indica el identificador definitivo que poseerá el

recurso luego su creación. En esos casos es esperable que el servidor informe de la URI definitiva como resultado del comando.

PUT

El verbo "PUT", por su parte, adquiere sentido al modificar un recurso ya existente. Al efectuar una petición con el verbo "PUT", el cliente incluye en la URI el recurso que se busca alterar. Por ejemplo, "clientes/5" apuntaría al cliente número 5. La petición incorpora además todos los datos que constituirían el nuevo estado del recurso. El servidor toma esta información y, tras evaluar su validez, aplica los cambios al recurso que ya está almacenado. Es decir que "PUT" opera como una acción de reemplazo, actualizando un recurso con nuevos datos proporcionados por el cliente.

PATCH

El verbo "PATCH" se asemeja a "PUT", pero difiere en su enfoque. Al realizar una petición "PATCH", el cliente se concentra en modificar únicamente partes específicas de un recurso, en lugar de reemplazarlo en su totalidad. Esta operación permite actualizaciones parciales, lo cual es útil cuando sólo se requiere cambiar ciertos atributos del recurso, por ejemplo, deshabilitar un cliente o modificar el stock de un artículo.

DELETE

Cuando un cliente desea eliminar un recurso empleará una petición con el verbo "DELETE" y la correspondiente URI que identifica el recurso a borrar.

Seguridad e idempotencia

La seguridad y la idempotencia son conceptos importantes en el contexto de los comandos HTTP, que definen aspectos cruciales en la interacción entre clientes y servidores en la web.

Seguridad

Los comandos de HTTP se consideran seguros cuando no modifican el estado de los recursos. Particularmente el verbo GET se considera seguro, por lo tanto la programación del servidor debe evitar que un GET realice modificaciones del estado del recurso solicitado, porque las aplicaciones cliente lo van a ejecutar suponiendo dicho comportamiento.

Idempotencia

La idempotencia se refiere a la característica de ciertos verbos de ser capaces de ser ejecutados múltiples veces sin cambiar el estado del recurso en el servidor luego del primer intento. En otras palabras, realizar la misma operación varias veces no debería tener un efecto diferente al realizarla solo una vez. Esto es fundamental para garantizar la coherencia y la predictibilidad en las interacciones entre clientes y servidores.

Los verbos "GET", "PUT" y "DELETE" son considerados idempotentes. Esto significa que realizar múltiples solicitudes con estos verbos no debería cambiar el estado del recurso en el servidor después de la primera solicitud exitosa. Por ejemplo, un "DELETE" debe borrar un recurso la primera vez que se lo ejecuta, pero otro DELETE con la misma URI no debería tener ningún efecto porque dicho recurso ya fue eliminado la primera vez.

Sin embargo, los verbos "POST" y "PATCH" no son indempotentes, ya que realizar la misma solicitud varias veces podría generar múltiples cambios en el estado del recurso en el servidor. Esto es importante tenerlo en cuenta al diseñar aplicaciones y sistemas que utilizan estos verbos, ya que múltiples solicitudes accidentales o repetidas podrían tener consecuencias no deseadas.

Códigos de respuesta

El protocolo HTTP está diseñado de manera que cuando un servidor recibe una petición, debe interpretarla y responder con un código de respuesta que informe al cliente sobre el estado de la solicitud. Esta respuesta consta de dos componentes: las cabeceras y el cuerpo de la respuesta.

Las cabeceras contienen metadatos que describen el contenido y el comportamiento de la respuesta. Estos metadatos pueden incluir información sobre el tipo de contenido, las cookies, la fecha de la respuesta, y otros detalles relevantes. Las cabeceras proporcionan contexto adicional para que el cliente comprenda cómo debe procesar el cuerpo de la respuesta.

El cuerpo de la respuesta contiene los datos reales que el servidor envía al cliente como resultado de la solicitud. Esto podría ser HTML, JSON, imágenes u otros tipos de contenido, dependiendo de la naturaleza de la solicitud y el recurso al que se esté accediendo.

Los códigos de respuesta son números enteros que están estandarizados en el protocolo HTTP. Cada código tiene un significado específico que indica el estado de la respuesta del servidor. Por ejemplo, el código "200 OK" indica que la solicitud se ha procesado correctamente y el recurso solicitado se encuentra disponible. Otros códigos, como "404 Not Found", señalan que el recurso solicitado no se ha encontrado en el servidor.

Estos códigos están diseñados para ser comprensibles tanto por máquinas como por desarrolladores, lo que facilita la identificación y el manejo de diferentes situaciones. Además, algunos códigos pueden ser redefinidos o adaptados por los servidores para ajustarse a situaciones específicas. Por ejemplo, una aplicación web puede definir un código de respuesta personalizado para indicar una situación particular que no está cubierta por los códigos estándar.

Cuerpo de las respuestas

Después del código de respuesta en una interacción HTTP, es común encontrar el contenido de la respuesta. Este contenido es opcional, pero suele ser incluido en la mayoría de los casos. El contenido de la respuesta es una parte vital de la comunicación entre el servidor y el cliente, ya que es la información que el servidor envía como resultado de la solicitud realizada.

En el contexto de una solicitud "GET" para obtener el estado de un recurso, por ejemplo, el servidor responderá con un código de respuesta que indica el resultado de la solicitud, como "200 OK" si todo está bien, y luego incluirá el contenido que describe el estado del recurso solicitado. El contenido de la respuesta es un componente de texto, ya que los protocolos de comunicación en la web son mayoritariamente basados en texto.

El tipo de contenido que se envía en la respuesta se define en la cabecera "Content-Type". Esta cabecera indica al cliente el formato en el que se encuentra el contenido de la respuesta, permitiéndole interpretarlo correctamente. Los tipos de contenido comunes incluyen:

- `text/plain`: Este tipo indica que el contenido es texto sin formato.
- `text/html`: Usado cuando se envía contenido en formato HTML.

- `application/json`: Utilizado cuando el contenido está en formato JSON. JSON es ampliamente utilizado para estructurar datos y se utiliza para representar objetos y estructuras en la respuesta.
- `application/xml`: Empleado para contenido en formato XML.

La elección del formato de contenido depende de la naturaleza de los datos y de la forma en que el servidor y el cliente interactúan. JSON se ha convertido en uno de los formatos más populares debido a su facilidad de uso y legibilidad, así como a la facilidad con la que puede ser interpretado tanto por máquinas como por desarrolladores.

Es importante destacar que si el contenido es binario (como una imagen) debe ser codificado en un formato de texto. Un ejemplo común de esto es el uso de codificación Base64 para enviar contenido binario en una respuesta HTTP.

Categorías de las respuestas

Los códigos de respuesta en el protocolo HTTP están estandarizados y siguen una estructura específica para informar sobre el resultado de una acción o solicitud. Estos códigos están compuestos por números enteros de tres dígitos, y cada uno proporciona una categorización y contexto sobre la respuesta del servidor.

Los códigos de respuesta están categorizados según el primer dígito del número:

- Informativos (1xx): Estos códigos indican que la solicitud ha sido recibida y el servidor continúa procesando. Por ejemplo, "100 Continue" significa que el servidor está esperando que el cliente continúe con la solicitud.

- Satisfactorios (2xx): Los códigos que comienzan con "2" indican que la solicitud fue recibida, comprendida y aceptada correctamente. "200 OK" es el más habitual e indica que la solicitud se procesó sin problemas.

- Redirección (3xx): Estos códigos señalan que el cliente necesita realizar una acción adicional para completar la solicitud. Por ejemplo, "301 Moved Permanently" indica que el recurso solicitado ha sido trasladado de manera permanente a otra ubicación.

- Error del cliente (4xx): Los códigos que empiezan con "4" indican que ha habido un error por parte del cliente. "404 Not Found" es un ejemplo común que indica que el recurso solicitado no pudo ser encontrado en el servidor.

- Error del servidor (5xx): Los códigos que comienzan con "5" apuntan a errores que ocurren en el lado del servidor. "500 Internal Server Error" es un ejemplo conocido, indicando que ha ocurrido un error interno en el servidor.

Uso para CRUD

La reutilización de códigos de respuesta en una API REST que implementa operaciones CRUD (Crear, Leer, Actualizar, Eliminar) es una práctica importante para mantener la consistencia en la comunicación entre el servidor y el cliente. Estos códigos de respuesta tienen significados que ayudan a los clientes a entender el estado y el resultado de sus solicitudes. Si bien pueden ser reinterpretados a conveniencia de cada caso es importante aplicarlos de manera coherente. Así, si entre frontend y backend se establece que cierto código de respuesta va a ser interpretado con un significado diferente al estándar, ese nuevo significado debe ser aplicado consistentemente en todo el sistema.

A continuación se presentan algunos ejemplos de cómo se pueden aplicar los códigos de respuesta en una API REST que implementa operaciones CRUD:

- GET (Leer):
    - Código 200: Se utiliza cuando la solicitud de consulta es exitosa y se devuelve el recurso solicitado.
    - Código 404: Indica que el recurso no se encontró en el servidor.
- POST (Crear):
    - Código 201: Se emplea cuando la solicitud de inserción de un nuevo recurso es exitosa. Junto con este código, la respuesta podría incluir la URI del recurso creado.
    - Código 400: Se usa si la solicitud está mal formada o no se pueden procesar los datos enviados por el cliente.
- PUT / PATCH (Actualizar):
    - Código 200: Puede usarse para indicar que la actualización ha sido exitosa.
    - Código 204: Indica que la actualización fue exitosa y no hay contenido adicional que enviar en la respuesta.
    - Código 404: Si el recurso no se encuentra.
- DELETE (Eliminar):
    - Código 204: Puede utilizarse para indicar que la eliminación ha sido exitosa y no hay contenido adicional que enviar en la respuesta.
    - Código 404: Si el recurso no se encuentra.
- Errores:
    - Código 400: Indica una solicitud mal formada o con datos incorrectos.
    - Código 401: Indica que el cliente no está autorizado para acceder al recurso solicitado.
    - Código 403: Indica que el cliente no tiene permiso para acceder al recurso.
    - Código 404: Indica que el recurso no se encontró.
    - Código 422: Se utiliza cuando los datos enviados por el cliente no cumplen con las reglas de negocio o son inválidos para la acción solicitada.
    - Código 500: Indica un error interno del servidor.

En todos los casos, es importante que el servidor proporcione contenido en la respuesta que explique la situación. En las respuestas exitosas, este contenido puede ser el propio recurso solicitado o una confirmación del éxito de la operación. En las respuestas de error, el contenido puede incluir detalles adicionales sobre el error para ayudar al cliente a comprender qué salió mal.