LOS 6 HTTP MÉTODOS MÁS USADOS

DD

Doctrina del desarrollador

Indice

GET method

GET method

¿Para que se usa?

El método **get** se utiliza para “traer” la información de un recurso, como puede ser una website, api o servidor

¿Es seguro?

Si, este método no cambia la información consultada, solo debe ser de lectura

¿Es idempotente?

Si, este método es idempotente, va a devolver siempre la misma información.
*Tengamos en cuenta que la información puede cambiar, por qué se actualizó por determinado motivo o proceso, pero el método en sí es idempotente

DD

Doctrina del desarrollador

GET method

Tipos de respuesta y status code

- 200: Si el recurso es accesible se devuelve un 200 con el body
- 400: Si enviamos de manera incorrecta la solicitud se devuelve un 400 con su respectivo objeto de error
- 404: Si no se encontró la información solicitada se devuelve un 404 con su respectivo objeto de error

Convención en el nombrado del recurso

- **GET** /api/**users**
- **GET** /api/**users/{user-id}**

Sustantivos y verbos
Es importante usar sustantivos sobre verbos. La razón es que los recursos representan datos e información, mientras que los verbos representan acciones.

Nombrado en plural
La Api está recuperando una colección de recursos y el endpoint puede devolver varias instancias del recurso, por ende es recomendable colocar la entidad en plural

DD
Doctrina del desarrollador

POST method

POST method

¿Para que se usa?

El método **post** se utiliza para crear un nuevo recurso. Por ejemplo en una base de datos

¿Es seguro?

No, este método se encarga de crear nuevos recursos

¿Es idempotente?

No, si enviamos la misma petición varias veces, se crearán múltiples recursos con identificadores (id) diferentes, pero conteniendo la misma información

*Es posible implementar el control de idempotencia en el backend, de modo que una solicitud repetida no genere un nuevo recurso

DD

Doctrina del desarrollador

POST method

Tipos de respuesta y status code

- 201: Si el recurso se pudo crear se puede devolver un 201 y el nuevo recurso creado
- 400: Si enviamos de manera incorrecta la solicitud se devuelve un 400 con su respectivo objeto de error
- 404: Si no se encontró la información solicitada se devuelve un 404 con su respectivo objeto de error

Convención en el nombrado del recurso

- **POST** /api/**users**
- **POST** /api/**users/{user-id}/addresses**

Sustantivos y verbos
Es importante usar sustantivos sobre verbos. La razón es que los recursos representan datos e información, mientras que los verbos representan acciones.

Nombrado en plural
La Api está recuperando una colección de recursos y el endpoint puede devolver varias instancias del recurso, por ende es recomendable colocar la entidad en plural

DD
Doctrina del desarrollador

PUT method

PUT method

¿Para que se usa?

El método **put** se utiliza para actualizar completamente un recurso existente

¿Es seguro?

No, este método se encarga de actualizar datos en un recurso

¿Es idempotente?

Si, porque al enviar la misma solicitud de manera consistente, obtendremos la misma respuesta al actualizar el mismo recurso

DD

Doctrina del desarrollador

PUT method

Tipos de respuesta y status code

- 200 o 204: Si el recurso se pudo actualizar se puede devolver un 200 y el nuevo recurso creado o un 204 sin el recurso nuevo creado
- 400: Si enviamos de manera incorrecta la solicitud se devuelve un 400 con su respectivo objeto de error
- 404: Si no se encontró la información solicitada se devuelve un 404 con su respectivo objeto de error

Convención en el nombrado del recurso

- **PUT** /api/**users/{user-id}**
- **PUT** /api/**users/{user-id}/addresses/{address-id}**

Sustantivos y verbos
Es importante usar sustantivos sobre verbos. La razón es que los recursos representan datos e información, mientras que los verbos representan acciones.

Nombrado en plural
La Api está recuperando una colección de recursos y el endpoint puede devolver varias instancias del recurso, por ende es recomendable colocar la entidad en plural

DD
Doctrina del desarrollador

DELETE method

DELETE method

¿Para que se usa?

El método **delete** se utiliza para eliminar un recurso existente

¿Es seguro?

No, este método se encarga de eliminar un recurso existente

¿Es idempotente?

Si, porque al enviar la misma solicitud de manera consistente, obtendremos la misma respuesta al eliminar el recurso

*Se debe tener en cuenta que si el recurso ya se eliminó se puede obtener un error (404)

DD

Doctrina del desarrollador

DELETE method

Tipos de respuesta y status code

- 200 o 204: Si el recurso se pudo actualizar se puede devolver un 200 y el nuevo recurso creado o un 204 sin el recurso nuevo creado
- 400: Si enviamos de manera incorrecta la solicitud se devuelve un 400 con su respectivo objeto de error
- 404: Si no se encontró la información solicitada se devuelve un 404 con su respectivo objeto de error

Convención en el nombrado del recurso

- **DELETE**/api/**users/{user-id}**
- **DELETE** /api/**users/{user-id}/addresses/{address-id}**

Sustantivos y verbos
Es importante usar sustantivos sobre verbos. La razón es que los recursos representan datos e información, mientras que los verbos representan acciones.

Nombrado en plural
La Api está recuperando una colección de recursos y el endpoint puede devolver varias instancias del recurso, por ende es recomendable colocar la entidad en plural

DD
Doctrina del desarrollador

PATCH method

PATCH method

¿Para que se usa?

El método **patch** se utiliza para actualizar un recurso existente, pero solo actualiza parcialmente al mismo

¿Es seguro?

No, este método se encarga de actualizar datos en un recurso

¿Es idempotente?

No, porque el resultado de solicitudes idénticas sucesivas puede diferir. A diferencia del put, que debería reemplazar el recurso por completo, patch solo actualiza partes o propiedades específicas del recurso. Si un recurso se actualiza parcialmente, enviar la misma solicitud nuevamente puede llevar a un estado diferente, según el estado actual del recurso. Este comportamiento es la razón por la que no se garantiza que las solicitudes produzcan el mismo resultado cuando se aplican varias veces.

DD

Doctrina del desarrollador

PATCH method

Tipos de respuesta y status code

- 200 o 204: Si el recurso se pudo actualizar se puede devolver un 200 y el nuevo recurso creado o un 204 sin el recurso nuevo creado
- 400: Si enviamos de manera incorrecta la solicitud se devuelve un 400 con su respectivo objeto de error
- 404: Si no se encontró la información solicitada se devuelve un 404 con su respectivo objeto de error

Convención en el nombrado del recurso

- **PATCH** /api/**users**
- **PATCH**/api/**users/{user-id}/addresses/{address-id}**

Sustantivos y verbos
Es importante usar sustantivos sobre verbos. La razón es que los recursos representan datos e información, mientras que los verbos representan acciones.

Nombrado en plural
La Api está recuperando una colección de recursos y el endpoint puede devolver varias instancias del recurso, por ende es recomendable colocar la entidad en plural

DD
Doctrina del desarrollador

HEAD method

HEAD method

¿Para que se usa?

El método **head** se utiliza para obtener la información del recurso (version/length/type). Cabe destacar que no se devuelve el body del request. Además es muy similar a método get pero mas rápido

¿Es seguro?

Si, este método no cambia la información consultada, solo debe ser de lectura

¿Es idempotente?

Si, este método es idempotente, va a devolver siempre la misma información.

DD

Doctrina del desarrollador

HEAD method

Tipos de respuesta y status code

- 200: Si el recurso es accesible se devuelve un 200 sin el body
- 400: Si enviamos de manera incorrecta la solicitud se devuelve un 400 con su respectivo objeto de error
- 404: Si no se encontró la información solicitada se devuelve un 404 con su respectivo objeto de error

Convención en el nombrado del recurso

- **GET** /api/**users**

Sustantivos y verbos
Es importante usar sustantivos sobre verbos. La razón es que los recursos representan datos e información, mientras que los verbos representan acciones.

Nombrado en plural
La Api está recuperando una colección de recursos y el endpoint puede devolver varias instancias del recurso, por ende es recomendable colocar la entidad en plural

DD
Doctrina del desarrollador

Referencias

- **RFC 9110:**
  https://www.rfc-editor.org/rfc/rfc9110#name-get
- **testfully:**
  https://testfully.io/blog/http-methods/