




Endpoints & Datos

 2025-01   20 min.

Que es un endpoint?

Un endpoint está conformado por 3 partes:

1. Metodo HTTP
2. Ruta
3. Implementación

Parte 1: Metodo HTTP

1. POST
2. GET
3. PATCH
4. PUT
5. DELETE

Parte 2: Ruta

Por ejemplo, « localhost:3030/usuarios »

Usualmente en el código solo es « /usuarios »

Endpoint - Hasta ahora

```
GET « localhost:3030/usuarios »
```

```
POST « localhost:3030/usuarios »
```

Parte 3: Implementacion

```
1 app.get("/", function (request, response) {  
2   response.json({ name: "Andres" });  
3 });
```

Endpoint - Implementacion

```
1 // Esta es la implementacion de un metodo GET en la ruta "/"
2 // request y response son los dos parametros de la funcion implementacion del endpoint
3
4 app.get("/", function (request, response) {
5     response.json({ name: "Andres" });
6 });
```

Endpoint - Implementacion

```
1  app.METODO(RUTA, function (request, response) {  
2    // IMPLEMENTACIÓN  
3  });
```


Enviar y recibir datos

El proposito de una API es responder a peticiones del cliente con informacion del servidor.

Enviar datos

Un backend puede enviar datos con una funcion muy sencilla llamada "send", del objeto "response".

```
1 app.get("/", (request, response) {  
2   response.send("Hello World!");  
3 });
```

Enviar datos - JSON

Usualmente, se utiliza la funcion JSON para enviar datos en formato JSON.

```
1 app.get("/", (request, response) {  
2   response.json({"name": "Andres"});  
3 });
```

Enviar datos - Status

Y se añade también el status code. El valor predeterminado es 200.

```
1 app.get("/", (request, response) {  
2   response.status(200).json({"name": "Andres"});  
3 });
```

Status code & Codigos HTTP

1XX	Informacion
2XX	Exito
3XX	Redireccion
4XX	Errores de cliente
5XX	Errores de servidor

Informacion

102

Procesando

Exito

200	OK
201	Creado
202	Aceptado
204	No hay contenido
205	Contenido Resetteado

Redireccion

301	Movido permanentemente
302	Movido temporalmente
304	No modificado

Error de cliente

400	Mala Peticion
401	No autorizado
402	Pago requerido
403	Prohibido
405	Metodo no permitido
406	No aceptable
408	Timeout

Error de cliente

409	Conflicto
410	No disponible
414	URI muy larga
423	Bloqueado
429	Muchas peticiones
451	No disponible por razones legales

Error de servidor

500	Error interno de servidor
501	No implementado
502	Gateway/proxy malo/incorrecto
503	Servicio no disponible
504	Gateway timeout
507	Espacio insuficiente

Recibir datos

Cuando el cliente hace una petición, le acompañan datos relevante a esta petición.

Donde están alojados esos datos?

Recibir datos - Opciones

Hay 3 lugares:

1. Body
2. Params (Tambien llamado URL Params)
3. Query (Tambien llamado Query Params)

Body - Que es?

1. No está en la URL
2. Protegido por HTTPS

Usualmente, se utiliza para enviar *datos* del cliente al servidor.

Body - Patrón de Diseño

- Se utilizan en peticiones POST/PUT/PATCH.
- Se utiliza formato JSON.

Params - Que es?

1. Está en la URL
2. No está protegido por HTTPS

```
« localhost:3030/usuarios/[id_usuario]/informacion »
```

Donde `[id_usuario]` es el param, por ejemplo:

```
« localhost:3030/usuarios/14/informacion »
```


Params - Patrón de Diseño

- Se puede usar para cualquier metodo HTTP.
- Idealmente hay un solo param en una URL.
- Usualmente es el identificador unico de un recurso.

Query - Que es?

1. Está en la URL
2. No está protegido por HTTPS

```
« localhost:3030/usuarios/?mes_nac=06 »
```

Donde `[?mes_nac=06]` es un valor del query.

Query - Patrón de Diseño

- Se usa en metodos GET con parametros para busquedas, ordenamientos, filtros, paginacion, y similares.

Hay alguna prohibicion sobre estos patrones?

No.



Saben de endpoints!