Rutas

Como vamos?

```
let express = require("express");
let cors = require("cors");
let app = express();

app.use(cors());

app.get("/products/:id", function (req, res, next) {
  res.json({ msg: "This is CORS-enabled for all origins!" });
});

app.listen(3000);
```

Endpoint - Metodo HTTP

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

Endpoint - Ruta

Por ejemplo,
« localhost:3030/usuarios »

Usualmente en el codigo solo es
« /usuarios »

Endpoint - Hasta ahora

GET « localhost:3030/usuarios »

POST « localhost:3030/usuarios »

Endpoint - Implementacion

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

Endpoint - Implementacion

```
// Esta es la implementacion de un metodo GET en la ruta "/"
// request y response son los dos parametros de la funcion implementacion del endpoint

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

Endpoint - Implementacion

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

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".

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

Enviar datos - JSON

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

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

Enviar datos - Status

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

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

Status code & Codigos HTTP

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

1XX => Informacion

102 ⇒ Procesando

2XX => Exito

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

3XX => Redireccion

301 ⇒ Movido permanentemente
302 ⇒ Movido temporalmente
304 ⇒ No modificado

4XX => Error de cliente

400 ⇒ Mala Peticion
401 ⇒ No autorizado
402 ⇒ Pago requerido
403 ⇒ Prohibido

405 ⇒ Metodo no permitido
406 ⇒ No aceptable
408 ⇒ Timeout
409 ⇒ Conflicto
410 ⇒ No disponible

4XX => Error de cliente

414 ⇒ URI muy larga
423 ⇒ Bloqueado
429 ⇒ Muchas peticiones
451 ⇒ No disponible por razones legales

5XX => 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

Codigos HTTP

https://http.cat/

Recibir datos

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

Donde estan alojada 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?

No están en la URL, y está protegido por HTTPS.

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

Body - Patron de Diseño

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

Params - Que es?

Es un valor ubicado en la ruta del endpoint, de esta forma:

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

Donde [id_usuario] es el param.

Params - Patron 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?

Es un valor ubicado en la ruta del endpoint, de esta forma:

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

Donde [?mes_nac=06] es el query.

Query - Patron de Diseño

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

Hay alguna prohibicion sobre estos patrones?

Hay alguna prohibicion sobre estos patrones?

No.