Desarrollo Backend con NodeJS

Métodos HTTP y Postman

course[It]

Métodos HTTP

- GET
- POST
- PUT
- DELETE

GET

Sirve para pedir **información**, se pueden enviar parámetros que viajarán expuestos en la url por eso nunca se deben utilizar para **comunicar información sensible**

POST

Sirve para enviar **información**, se pueden enviar parámetros que viajarán ocultos en el **body** de la request, por eso se lo considera la forma segura de enviar información sensible

PUT

Sirve para modificar **información**, al igual que post se pueden enviar parámetros que viajarán ocultos en el **body** de la request, por eso se lo considera la forma segura de modificar información sensible

DELETE

Sirve para borrar **información**, los parámetros al igual que en el get se envían expuestos en la url

Códigos de estado

O status codes, son una convención que se utiliza para notificar sobre el estado de la comunicación en las requests HTTP

- 10x - informativos
- 20x - éxito
- 30x - redirecciones
- 40x - errores en el cliente
- 50x - errores en el servidor

Importante

Es importante destacar que todo esto es una convención, y por lo tal requiere ser implementada por los desarrolladores de forma correcta

TL;DR: ¿Puedo mandar un mensaje de éxito acompañado de un estado 400? SI. ¿Debería? **CLARO QUE NO**

¿Cómo se ve un post a nivel código?

En nuestras rutas (sin controller):

```
router.post("/user", function (req, res, next) {
  const { body } = req; // esto es lo mismo que hacer req.body
  console.log(body);

  res.send("request realizada con éxito");
});
```

Bonus track: Javascript object destructuring

Es una forma simple de acceder de forma individual a las propiedades de un objeto como variables

(y más -> https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Operadores/Destructuring_assignment)

```
const user = {
  id: 42,
  is_verified: true
};

const {id, is_verified} = user;

console.log(id); // 42
console.log(is_verified); // true
```

¿Cómo se ve un post a nivel código?

En nuestras rutas (con controller):

```
router.post("/user", function (req, res, next) {
  UserInstance.postUser(req, res);
});
```

En el controller:

```
postUser(req, res) {
    const { body } = req;
    console.log(body);

    res.status(200).send("usuario agregado con éxito");
  }
}
```

Pero... ¿Qué es el body?

El body es un objeto de Javascript que nos permite enviar la información deseada en formato clave: valor. Durante la comunicación el protocolo lo interpretará como JSON

¿Cómo se ve una request POST desde el front?

Con Axios:

```
const response = await axios.post('/user', {
  firstName: 'Fred',
  lastName: 'Flintstone'
})

console.log(response) // 200, 400 o lo que nos devuelva la API
```

Con Fetch (funcionalidad nativa de Javascript):

```
// Example POST method implementation:
async function postData(url = '', data = {}) {
  // Default options are marked with *
  const response = await fetch(url, {
    method: 'POST', // *GET, POST, PUT, DELETE, etc.
    mode: 'cors', // no-cors, *cors, same-origin
    cache: 'no-cache', // *default, no-cache, reload, force-cache, only-if-cached
    credentials: 'same-origin', // include, *same-origin, omit
    headers: {
      'Content-Type': 'application/json'
      // 'Content-Type': 'application/x-www-form-urlencoded',
    },
    redirect: 'follow', // manual, *follow, error
    referrerPolicy: 'no-referrer', // no-referrer, *no-referrer-when-downgrade, origin, origin-when-cross-origin,
same-origin, strict-origin, strict-origin-when-cross-origin, unsafe-url
    body: JSON.stringify(data) // body data type must match "Content-Type" header
  });
  return response.json(); // parses JSON response into native JavaScript objects
}
```

Headers

Son indicaciones informativas que puede traer nuestra request. Desde el backend las recibimos y podemos utilizarlas para generar condiciones como por ejemplo, solo permitir que pueda acceder a un endpoint **una persona que tiene un header determinado** con un valor aceptado por nuestra API.

PUT

Desde el front:

```
const response = await axios.put('/user', {
  firstName: 'Fred',
  lastName: 'Flintstone'
})
console.log(response) // 200, 400 o lo que nos devuelva la API
```

En la ruta:

```
router.put("/user", function (req, res, next) {
  // nuestra funcionalidad
});
```

DELETE

Desde el front:

```
const response = await axios.delete(`/delete/${id}`)
console.log(response) // 200, 400 o lo que nos devuelva la API
```

En la ruta:

```
router.delete("/delete/:id", function (req, res, next) {
  // nuestra funcionalidad
});
```

¿Podemos enviar una request con body (y headers) sin un front?

¡SI!

- cURL
  - https://curl.se/docs/httpscripting.html
  - ```
    curl -d '{"name":"Bel"}' -H "Content-Type: application/json" -X POST http://localhost:3000/user
    ```
- Postman
  - https://www.postman.com/downloads/
- Y muchas más
    ```