<codoa codo/>

Agencia de Aprendizaje a lo largo de la vida

FULL STACK FRONTEND
Clase 28

Node 4
(Parte II)

BA

Les damos la bienvenida

Vamos a comenzar a grabar la clase

Clase 27

Clase 28

Clase 29

Node

- Express
- Servidor estático
- Nodemon
- Rutas

Node

- GET
  POST
  PUT
  DELETE

Node

Ahora probemos con POST.

Parseando datos recibidos

Hasta el momento venimos trabajando con **GET**, pero para que nuestro servidor pueda recibir peticiones por POST necesitamos convertir los datos recibidos en el **BODY** a un formato que **entienda el servidor**.

Usando los **middlewares** nativos .urlencoded() y .json() podemos convertir la data de estos formatos a uno que el **servidor pueda manejar.**

Para poder armar una request en POSTMAN que nos permita enviar una nueva pelicula para cargar a nuestro array de peliculas, deberemos proceder de la siguiente manera:

***nota:** en versiones previas a **express 4.16.0** se utilizaba una librería llamada body-parser para este propósito.*

Enviando la petición al servidor

Indicamos el método

Invocamos la ruta

Agregamos información al body de la “request” y especificamos que lo haremos agregando un JSON

Programando en el servidor el POST

Al igual que en los casos anteriores, creamos la ruta en la cual recibiremos la request con la información para agregar a nuestro array.

```
24  // Crear una nueva película
25  router.post('/', (req, res) => {
26
27  });
```

En este caso, tenemos dentro de req, la propiedad body, que en su interior recibe los parámetros enviados a través del POST en el frontend (o en este caso desde postman).

Podemos acceder a los valores que necesitamos a través de:

req.body.title
req.body.director
req.body.year

Programando en el servidor el POST

Ahora, una vez más agregando un poco de Javascript creamos un objeto para agregar a nuestro array de películas:

```
24  // Crear una nueva película
25  router.post('/', (req, res) => {
26      const newMovie = {
27          id: movies.length + 1,
28          title: req.body.title,
29          director: req.body.director,
30          year: req.body.year
31      };
32      movies.push(newMovie);
33
34  });
```

Al igual que en el GET, la manera de resolver cómo agregar la película al array puede diferir de acuerdo a la solución que estemos pensando.

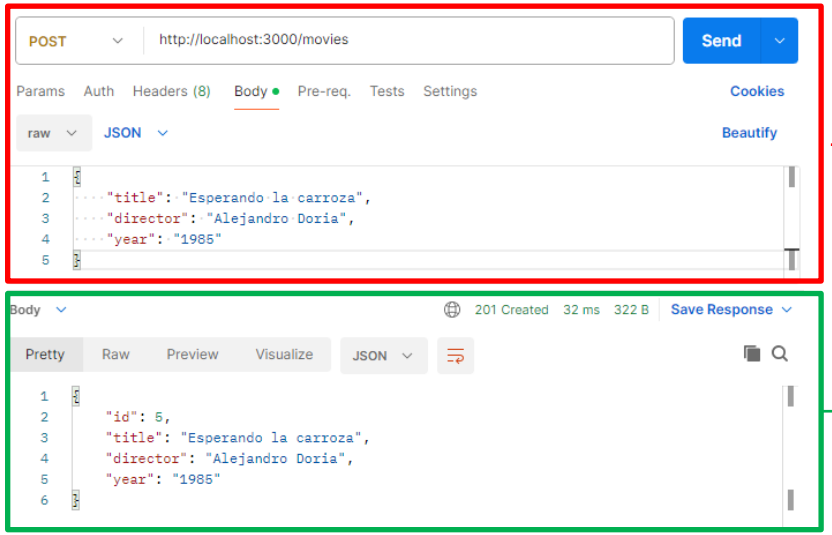
Programando en el servidor el POST

Ahora, lo único que falta es determinar que vamos a responder como resultado de esta petición. Por convención, en general en un pedido “post” se suele devolver el id generado o bien el objeto completo. En este caso, vamos a devolver el objeto completo bajo el status 201.

```
24  // Crear una nueva película
25  router.post('/', (req, res) => {
26      const newMovie = {
27          id: movies.length + 1,
28          title: req.body.title,
29          director: req.body.director,
30          year: req.body.year
31      };
32      movies.push(newMovie);
33      res.status(201).json(newMovie);
34  });
```

Probando la petición en POSTMAN

Ahora, lo único que falta es determinar que vamos a responder como resultado de esta petición. Por convención, en general en un pedido “post” se suele devolver el id generado o bien el objeto completo. En este caso, vamos a devolver el objeto completo bajo el status 201.



REQUEST

RESPONSE

Agencia de Aprendizaje
a lo largo de la vida

No te olvides de dar el presente

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.
- **Realizá los ejercicios obligatorios.**

Todo en el Aula Virtual.