<codoa codo/>


# FULL STACK FRONTEND
# Clase 28

## Node 4
## (Parte III)

BA

# Les damos la bienvenida

Vamos a comenzar a grabar la clase

Clase 27

Clase 28

Clase 29

**Node**

- Express
- Servidor estático
- Nodemon
- Rutas

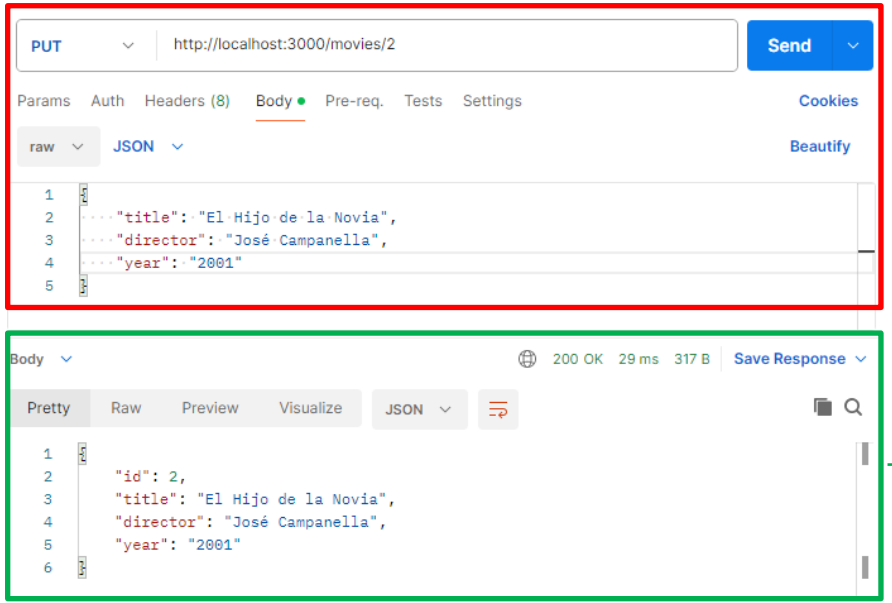
**Node**

- GET
  POST
  PUT
  DELETE

**Node**

# Metodo PUT

El método PUT se utiliza para reemplazar completamente un recurso existente, a diferencia de PATCH que realiza modificaciones parciales.
En este caso, en el método PUT evíamos por parámetro en la URL el id de la película que queremos modificar y en el body el nuevo objeto que reemplazara al anterior:




REQUEST



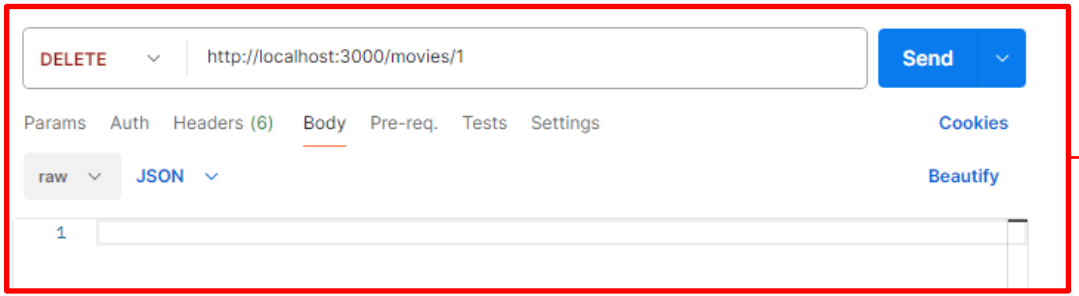

RESPONSE

Y del lado del servidor, a través de Javascript, gestionamos la modificación de la película correspondiente. La respuesta en el status será un 200, lo que indica que se procesó exitosamente. Como es la respuesta por defecto, no es necesario hacerla explícita en el código.

```
36  // Actualizar una película por ID
37  router.put('/:id', (req, res) => {
38      const movie = movies.find(m => m.id === parseInt(req.params.id));
39      if (!movie) return res.status(404).send('Movie not found');
40
41      movie.title = req.body.title || movie.title;
42      movie.director = req.body.director || movie.director;
43      movie.year = req.body.year || movie.year;
44
45      res.json(movie);
46  });
```

# Método DELETE

Se utiliza para eliminar un recurso específico en el servidor. Después de realizar la solicitud, el recurso se eliminará. En el método DELETE, se puede optar por una respuesta standard o devolver el objeto que se eliminó.




REQUEST


RESPONSE

Se utiliza para eliminar un recurso específico en el servidor. Después de realizar la solicitud, el recurso se eliminará. En el método DELETE, se puede optar por una respuesta standard o devolver el objeto que se eliminó.

```
48  // Eliminar una película por ID
49  router.delete('/:id', (req, res) => {
50      const movieIndex = movies.findIndex(m => m.id === parseInt(req.params.id));
51      if (movieIndex === -1) return res.status(404).send('Movie not found');
52
53      const deletedMovie = movies.splice(movieIndex, 1);
54      res.json(deletedMovie);
55  });
56
```

# Middlewares

Casi de forma instintiva, ya utilizamos varios middlewares para configurar nuestro programa.

**¿Qué es un middleware entonces?**

Son simplemente funciones que se ejecutan antes o después de otras y los hay de distintos tipos:

- **Middlewares de nivel de aplicación:** Son middlewares que se aplican a toda la aplicación y se configuran utilizando **app.use()**
- **Middlewares de nivel de ruta:** Son middlewares que se aplican a una ruta específica.
- **Middlewares de manejo de errores:** Son middlewares especiales que se utilizan para manejar errores en la aplicación.

A lo largo de las clases iremos viendo distintos middlewares y también aprenderemos a crear los propios.

# 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.**