DigitalHouse >
Coding School

PROCESAMIENTO
PUT Y DELETE

“

La **información** que viaje como petición **PUT** y **DELETE** debe hacerlo a través de un **formulario**.

1.

PETICIONES PUT

MANEJAR PETICIONES PUT

Usamos el método put para:

- **Enviar información** sensible al servidor de manera segura
- **Modificar** un recurso existente

Cuando definimos una ruta podemos hacerlo directamente sobre la ejecución de express, implementar un sistema de ruteo o también incorporar controladores que se encarguen de manejar las rutas.

*Sin importar el camino que elijamos para implementar en nuestra aplicación, es en el **callback** de la **ruta que estamos definiendo** en donde escribiremos la lógica para manejar la petición que esté llegando.*

MANEJAR PETICIONES PUT

En un contexto en donde quisiéramos **modificar** datos de una película almacenada en nuestro sistema, tendríamos que crear dos **rutas**: una que **muestre** el formulario de **edición** y otra que se encargue de **procesar** la información.

En ambas rutas, deberemos definir un parámetro que nos ayude a identificar al *recurso único* que estamos queriendo modificar. Generalmente, se usa el **id**.

{}

```
// ruta que envía un formulario de edición a la vista → GET
router.get('/pelicula/:id/editar', (req,res) => {res.render('editar')});

// ruta que procesa la información del formulario → PUT
router.put('/pelicula/:id/editar/', (req,res) => {...});
```

2.

PETICIONES
DELETE

MANEJAR PETICIONES DELETE

Usamos el método delete para:

- **Eliminar** un recurso existente

Cuando definimos una ruta podemos hacerlo directamente sobre la ejecución de express, implementar un sistema de ruteo o también incorporar controladores que se encarguen de manejar las rutas.

*Sin importar el camino que elijamos para implementar en nuestra aplicación, es en el **callback** de la **ruta que estamos definiendo** en donde escribiremos la lógica para manejar la petición que esté llegando.*

MANEJAR PETICIONES DELETE

En un contexto en donde quisiéramos **eliminar** una película almacenada en nuestro sistema, tendríamos que crear una ruta que se encargue de buscar ese recurso y eliminarlo. En la ruta deberemos definir un parámetro que nos ayude a identificar al recurso que estamos queriendo eliminar. *Generalmente, se usa el **id**.*

{}

```
// ruta que procesa la información del formulario → DELETE
router.delete('/pelicula/:id/eliminar/', (req,res) => {...});
```

En el **callback** de las peticiones **DELETE** y **PUT** tendremos que definir la **lógica** de lo que queremos implementar con esa información que recibimos.

Después, haremos un **redireccionamiento** para cortar el ciclo *request/response*.

```
res.redirect('/peliculas');
```

HABILITAR MÉTODOS HTTP

Los métodos **PUT** y **DELETE** no son soportados por todos los navegadores. Para poder implementarlos en nuestro formulario es necesario instalar el paquete `method-override`:

```
>_ npm install method-override --save
```

Una vez instalada, hay que configurar la aplicación en `app.js` para poder **sobreescribir** el método original y poder implementar los métodos *PUT* o *DELETE*:

```
{}
const methodOverride = require('method-override');
app.use(methodOverride('_method'));
```

CONFIGURAR EL FORMULARIO

Para terminar de habilitar el envío de información a través de alguno de estos dos métodos, tenemos que agregarle un **query string** al `action` del formulario:

- `?_method=PUT`
- `?_method=DELETE`

De esta forma estamos aclarando que, sin importar el método original que tenga seteado el formulario, queremos enviar la información usando PUT o DELETE.

html

```
<form method="POST" action="/pelicula/:id/editar?_method=PUT">
    ...
</form>
```

Aunque los formularios sigan teniendo configurado el método POST, la información estará viajando a través del **método** que **definamos** en cada **ruta**.

De esta forma, es fácil identificar **qué hace** cada ruta de nuestro sistema tan solo viendo el **método** que **implementa**.