Procesamiento
PUT y DELETE

DigitalHouse >
Coding School

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

Los métodos PUT y DELETE no son soportados nativamente por los formularios y deberemos utilizar métodos alternativos para enviar la información.”

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** y también **incorporar controladores** que se encarguen de manejarlas.

En todos los casos las rutas recibirán dos parámetros:

- Un string con la ruta que estaremos procesando.
- Un callback donde definiremos qué lógica ejecutaremos cuando el cliente pida esa ruta.

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, vamos a 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) => {...});
```

Manejar peticiones DELETE

Usamos el método delete para:

- **Eliminar** un recurso existente.

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 el 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) => {...});
```

Peticiones PUT y DELETE

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

Normalmente al terminar de procesar el pedido haremos un **redireccionamiento** hacia otra página que muestre el resultado de la acción.

{}

```
router.post('/pelicula/:id/eliminar', (req,res) => {
    // Eliminamos la película solicitada
    // Si todo sale bien, redirigimos al listado de películas.
    res.redirect('/peliculas');
});
```

Habilitar métodos HTTP

Los métodos **PUT** y **DELETE** no son soportados por los formularios de HTML. Para poder recibir información usando estos métodos es necesario instalar el paquete `method-override`:

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

Una vez instalado, 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 según el método que queramos implementar:

- ?_method=PUT
- ?_method=DELETE

De esta forma estamos aclarando que, sin importar el método original que tenga seteado el formulario, queremos recibir 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 llegará a la ruta que tenga el método que nosotros especificamos, como PUT o DELETE.

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

”


DigitalHouse>
Coding School