Middlewares a nivel aplicación

DigitalHouse >
Coding School

Son aquellos middlewares que queremos que se ejecuten **siempre** que se haga un pedido a la aplicación, sin importar a qué ruta ingrese el usuario.

¿Cómo lo implementamos?

Invocando el método `app.use()` podemos configurar un middleware que se va a ejecutar cada vez que se haga un pedido a nuestro servidor.

Este método recibe un callback con tres parámetros:

- el objeto **request**.
- el objeto **response**.
- la función **next**.

```
{}
app.use(function(req, res, next) {
    ...
})
```

¿Qué es next()?

`next()` es un callback que se va a encargar de apilar todos los middlewares que apliquen a una misma petición y luego **ejecutarlos** uno tras otro.

Cuando llegue al último y, si se ejecutaron correctamente, pasará al siguiente paso que es ejecutar el método del controlador que maneja esa ruta.

*Por eso siempre al terminar cada middleware, ejecutamos **next**.*

```
{}
app.use(function(req, res, next) {
    ...
    next();
})
```

Algunos de los middlewares a **nivel aplicación** que ya utilizamos son:

```
// configuración de carpeta de archivos estáticos
app.use(express.static(__dirname + '/public'));

// configuración de ruteo
const rutasProductos = require('./routes/products');
app.use('/', rutasProductos);
```

Implementando el error 404

404 es el código de error que representa que la ruta solicitada no fue encontrada. Por lo general, es el tipo de error que recibimos cuando hemos escrito una dirección equivocada.

Haciendo uso de los middlewares a nivel aplicación, podemos decirle a nuestra aplicación que, en caso de que una ruta no exista, se responda con ese error.

Como este middleware se debe ejecutar solo si todas las rutas fallan, es importante que siempre esté a lo último.

¡Vamos a ver cómo escribirlo y cómo funciona!

{ código }

```
app.use((req, res, next) => {
    res.status(404).render('404-page');
    next();
});
```

En el **entry point** de la aplicación y al final de toda las rutas que tengamos, llamamos al método **use()** para implementar un middleware que aplique a todas las peticiones del sitio.

{ código }

```
app.use((req, res, next) => {
    res.status(404).render('404-page');
    next();
});
```

Le pasamos el callback con los tres parámetros necesarios: el request, el response y el **next**.

{ código }

```
app.use((req, res, next) => {
    res.status(404).render('404-page');
    next();
});
```

Si la página **no existe**, devolverá un error 404 y, adicionalmente, renderizará la vista que hayamos preparado para ese escenario.

{ código }

```
app.use((req, res, next) => {
    res.status(404).render('404-page');
    next();
});
```

La función **next** se encargará de ejecutar el próximo paso.

Si la página **existe**, llamará al controlador y este devolverá la vista solicitada.


DigitalHouse>
Coding School