Coding Bootcamp

Sprint 2

Temario

Temario

- Middlewares.
- Uso de middlewares en Express.
- Middlewares integrados.

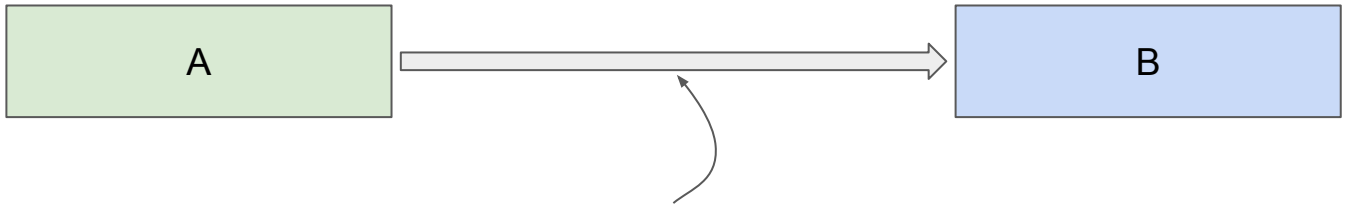
Middlewares

Middlewares (1/2)

En el ambiente de Express se habla mucho de *middlewares*, aunque no es un concepto exclusivo de Express y probablemente hayan oído hablar de *middlewares* en otros ámbitos.

Como dice el nombre, el *middleware* es un "pedazo de código" que se ejecuta en medio de una comunicación, con el fin de agregar cierta funcionalidad en el proceso.

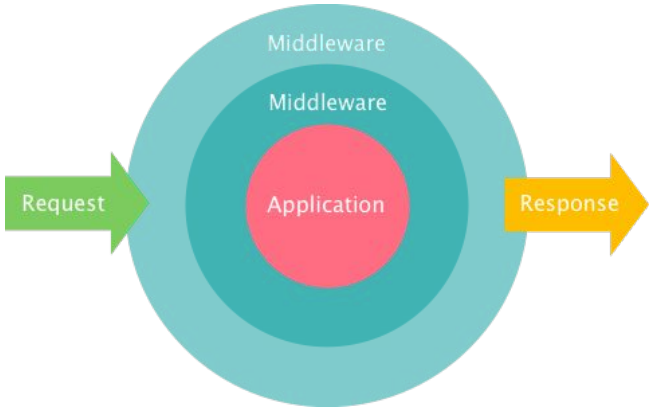
Ejemplo, un sistema A le quiere enviar datos a un sistema B:



Podríamos crear un *middleware* que se "meta" en medio de esa comunicación con el fin de, por ejemplo, manipular los datos que están siendo enviados. Por ejemplo, podríamos hacer un corrección ortográfica de los datos. Y podríamos "encadenar" otros *middlewares*.

Middlewares (2/2)

En el ámbito de las aplicaciones web, típicamente se utilizan *middlewares* para filtrar los *requests* HTTP, los cuales se hacen pasar por varias capas de *middlewares* antes de llegar al “core” de la aplicación.



Uso de middlewares en Express

Uso de middlewares en Express (1/4)

En Express, un *middleware* es una función que recibe tres parámetros:

1. Objeto Request: `req`.
2. Objeto Response: `res`.
3. Función `next`.

La función *middleware* puede hacer una o varias de las siguientes acciones:

- Ejecutar cierto código (en principio, cualquier código).
- Hacer modificaciones sobre los objetos *Request* y *Response*.
- Finalizar el ciclo de *request-response*.
- Llamar al próximo *middleware* que hay para ejecutar.

Uso de middlewares en Express (2/4)

Ejemplo de un *middleware* que se encarga de hacer un *log* cada vez que llega un *request* y de agregarle el atributo `requestTime` al *request* que llegó.

```
const miLogger = (req, res, next) => {

  console.log("Se recibió un request");

  req.requestTime = new Date();

  next(); // Para dar paso al siguiente middleware.

};
```

Notar que por ahora sólo se creó una función. Aún falta decirle a Express cuándo se debería ejecutarla.

Uso de middlewares en Express (3/4)

Siguiendo con el ejemplo anterior, ahora que tenemos creada la función *middleware* tenemos que indicarle a Express cuándo usarla.

Hay más de una forma de hacerlo.

Si queremos que el *middleware* se utilice para toda la aplicación, podemos hacerlo así:

```
const express = require("express");
const app = express();
app.use(miLogger);
```

Notar que el orden en que se hacen los *bindings* será el orden en que se ejecuten los *middlewares*.

Uso de middlewares en Express (4/4)

En caso de sólo se quiera ejecutar el *middleware* para todos los métodos de determinada ruta, se puede especificar de la siguiente manera:

```
app.use("/productos", miLogger);
```

También se puede agregar un *middleware* para determinada ruta y método:

```
app.post("/productos", miLogger, function (req, res) {
   // Código normal del handler...
});
```

Para ver otras formas de usar *middlewares*, consultar la documentación oficial.

Middlewares integrados

Middlewares integrados (1/2)

Express ya viene con una serie de *middlewares* integrados que podemos usar.

Por ejemplo:

- `express.static` para servir archivos estáticos como imágenes, CSS, etc.
- `express.json` para *parsear requests* que vienen con contenido de tipo JSON.
- `express.urlencoded` para *parsear requests* que vienen con contenido de tipo URL-Encoded (como cuando nos están enviando datos que vienen desde un formulario).

👉 Definición: “Parsear” significa analizar un texto sintácticamente y convertirlo en otra estructura como, por ejemplo, un objeto. Un servidor, cuando recibe un *request* con contenido, recibe simplemente un texto, una cadena de caracteres. Por lo tanto hay que indicarle a Express cómo queremos *parsear* dicho texto.

Middlewares integrados (2/2)

Ejemplo de uso:

```
const express = require("express");
const app = express();


app.use(express.urlencoded({ extended: true }));
```

Con este código le estamos diciendo a Express que cuando lleguen datos de tipo “URL-Encoded” se debe crear un atributo `body` dentro del objeto *request*:

```
req.body
```