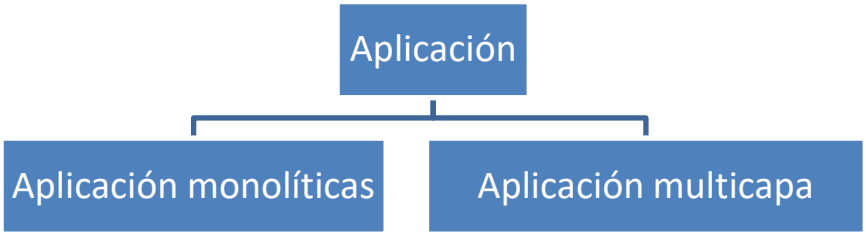
Pila de middleware

Introducción

Básicamente, el desarrollo de aplicaciones webs se puede hacer desde dos puntos de vista, de manera monolítica o multicapa. Una aplicación monolítica (*monolithic application*) es aquella en la que se desarrolla *todo* en un único componente, capa o controlador. En cambio, una aplicación multicapa (*multitier application*) es aquella que utiliza varios componentes, capas o controladores cada uno de los cuales posee una funcionalidad y procesamiento definido.

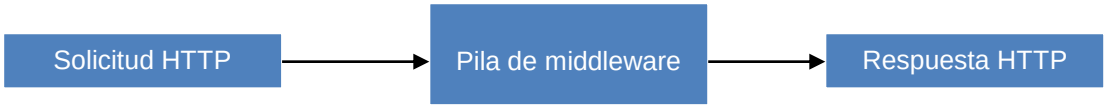


Actualmente, preferimos los entornos multicapa porque son más sencillos de desarrollar y, por encima de todo, de mantener y probar. Como no podía ser de otra manera, Express permite el desarrollo de aplicaciones multicapa y lo hace mediante el uso de la pila de *middleware*.

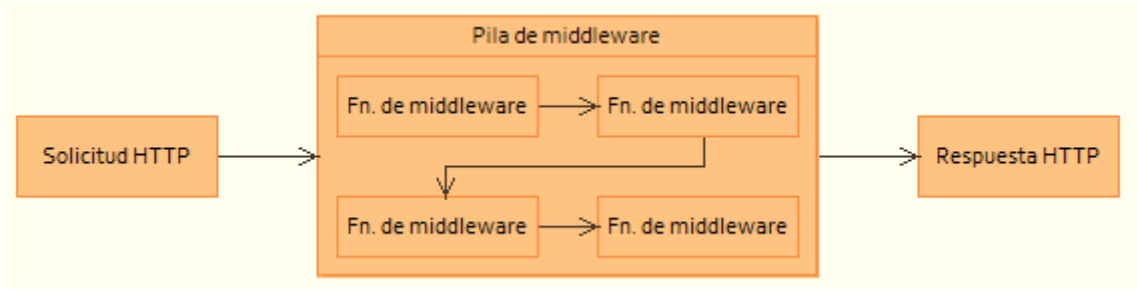
Antes de presentar la pila, hay que tener claro que es el *middleware*. Un componente de middleware (*middleware component*) no es más que el término formal con que se conoce a una pieza de software reutilizable. La cual realiza una determinada funcionalidad de procesamiento de las peticiones HTTP. Así, por ejemplo, tenemos componentes de *middleware* para llevar a cabo el proceso de autenticación, la aplicación de restricciones de seguridad, la publicación de contenido estático, etc.

Por su parte, la pila de middleware (*middleware stack*), también conocida como flujo de procesamiento (*processing flow*) o conducto (*pipeline*), contiene la secuencia de funciones de *middleware* que procesan, una detrás de otra, las peticiones HTTP recibidas de los clientes para construir, entre todas ellas, las respuestas HTTP a remitir como contestación. Todo hay que decirlo, algunas funciones no participan en la redacción de la respuesta como, por ejemplo, el *middleware* de registro de eventos que escribe en un archivo o en la salida estándar información sobre la solicitada en procesamiento.

La idea que se esconde bajo este sistema de *middleware* es que toda petición que reciba la aplicación pase por el flujo de funciones de *middleware* registradas en la pila y que, entre todas ellas, lleven a cabo su tratamiento, generándose la respuesta a remitir al cliente.



En Express, la pila está formada por funciones, conocidas formalmente como *funciones de middleware* (*middleware functions*) o controladores de petición (*request handlers*), disponibles a través de componentes de *middleware*. A modo de ejemplo, consideremos el componente serve-static. Se utiliza cuando deseamos que la aplicación sirva contenido estático. Este componente dispone de una función, *no middleware*, que recibe la ruta del directorio que contiene los archivos que puede servir estáticamente. Y devuelve la función de *middleware* que hay que registrar en el flujo de procesamiento para que así pueda servir contenidos cuando sea necesario.



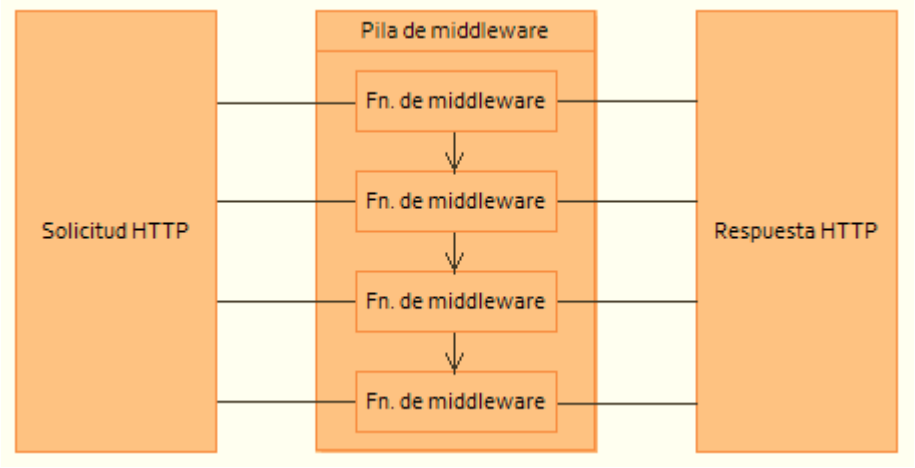
Un componente de *middleware* puede ser básicamente dos cosas:

- Una función de *middleware* por sí misma.
- Una función que devuelve funciones de *middleware*.

En cualquier caso, lo importante a recordar es que en el flujo de procesamiento de la aplicación sólo debemos registrar funciones de *middleware* o controladores de petición.

El componente de la aplicación Express que se encarga de ejecutar ordenadamente las distintas funciones de *middleware* registradas en la pila de procesamiento, se conoce formalmente como motor de middleware (*middleware engine*).

En líneas generales, cuando se recibe una petición HTTP, la aplicación se la pasa al motor de *middleware*, el cual va invocando, una a una en orden de registro, las distintas funciones registradas. Tras finalizar la ejecución de la pila, el motor le pasa la respuesta HTTP generada por el *middleware* a la aplicación para su envío al cliente.



Funciones del middleware

Como ya sabemos, una función de *middleware* es una función JavaScript que realiza una determinada funcionalidad de la aplicación. Es un controlador de petición. Puede trabajar sobre el objeto que representa la petición HTTP recibida del cliente y/o el objeto que representa la respuesta HTTP que la aplicación acabará remitiendo al cliente como contestación.

La función, al ser un controlador de petición, debe presentar la siguiente estructura:

```
function(req,res)
function(req,res,next)
```

req	Request	Solicitud HTTP en procesamiento.
res	Response	Respuesta HTTP que se está generando.
next	Function	Función que debe invocar el middleware para indicarle al motor que ejecute el siguiente componente de la pila de procesamiento.

Registro de funciones

Mediante el registro de middleware (*middleware register*) se añade, al final de la pila de procesamiento, una función de *middleware*. Se realiza mediante el método use() de la aplicación:

```
use(fn)
use(route,fn)
```

route	string	Ruta a la que se aplicará el componente middleware. Si no se especifica, se asumirá que se debe ejecutar para todas las peticiones.
fn	function	Función que implementa lógica del componente de middleware.

Con el registro de funciones de *middleware* añadimos funcionalidades a la aplicación.

Orden de registro

El orden en que se registra las funciones de *middleware* es importante. Si un determinado componente utiliza algo generado por otro, es necesario registrar primero la función de la que depende para que de esta manera el motor de *middleware* la invoque primero y, así, la segunda pueda acceder a cualquier objeto generado por éste.