* entender el proceso requerido para el procesamiento de archivos del lado del servidor.
* Manejar la librería más utilizada para este procedimiento llamada Multer.
* Implementar el lugar dentro del servidor en donde queramos que se almacenen los archivos.
* Generar una correcta nomenclatura de archivos para que los mismos sean persistentes a lo largo de toda la aplicación.

1. Introducción/configuración
2. Subiendo archivos
3. Validación de los datos

1.

Librería **Multer:** mecanismo para subida de archivos en el servidor (imgane de avatart). usado generalmente como un middleware y no solo nos va a permitir **subir archivos al servidor,** sino que también nos permite elegir en qué **ubicación** del servidor queremos que se almacenen estos archivos y el **nombre** con el que se guardará cada uno de ellos.

Existen otras librerías además de multer.

Instalación**. Npm install multer.** Requerirla donde la necesitemos (por ejemplo en los **routers)**.

En el enrutador porque vamos a querer utilizar multer para procesar la info que viene del formulario de creación.

Los fomr se usarán así: con el atributo **enctype** le indicamos que va a recibir archivos.

asi el fomr se prepara para enviar más info (una imagen, no solo texto).

**Código de ejemplo de los videos:** <https://github.com/codelando/dh-node-multer-instalacion>

2.

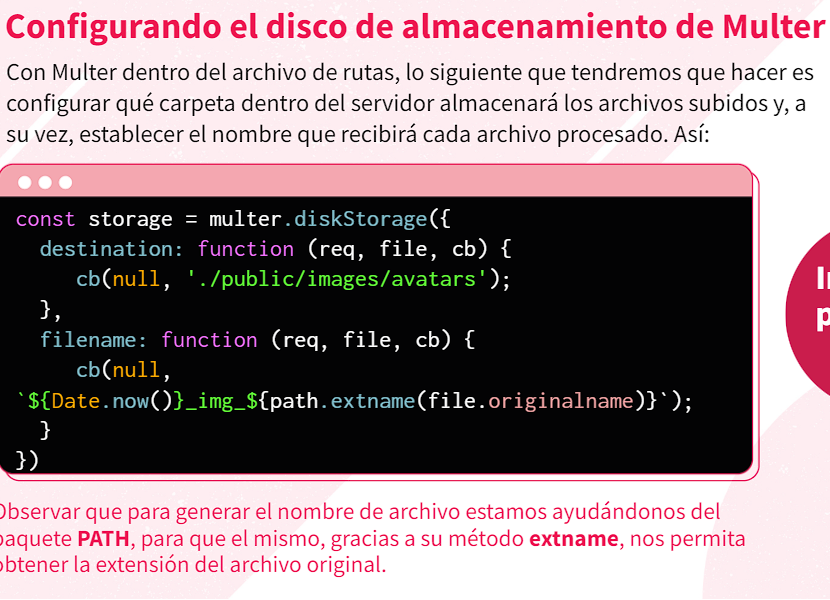
**Subiendo archivos:**

**resaltart el enctypo y el tipo=”file” y el name.**

usando el const multer, traeremos a **storage** que contrendra **muliter.diskStorage** que nos ayudara a decir donde almanenar la info dentro del servidor y que nombre asignarle. **diskStorage es un objeto con dos métodos: destination y filename.**

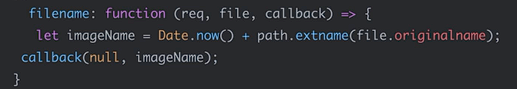
**DESTINATION: tres parámetros (req, file y callback)** req reprecente el request enviado por el cliente, file representa el archivo enviado, y el tercero es una callback que se encargara de almacenar el archivo en el destino indicado





En destinación usamos path para indicarle la ruta a donde deberá guardar los archivos:

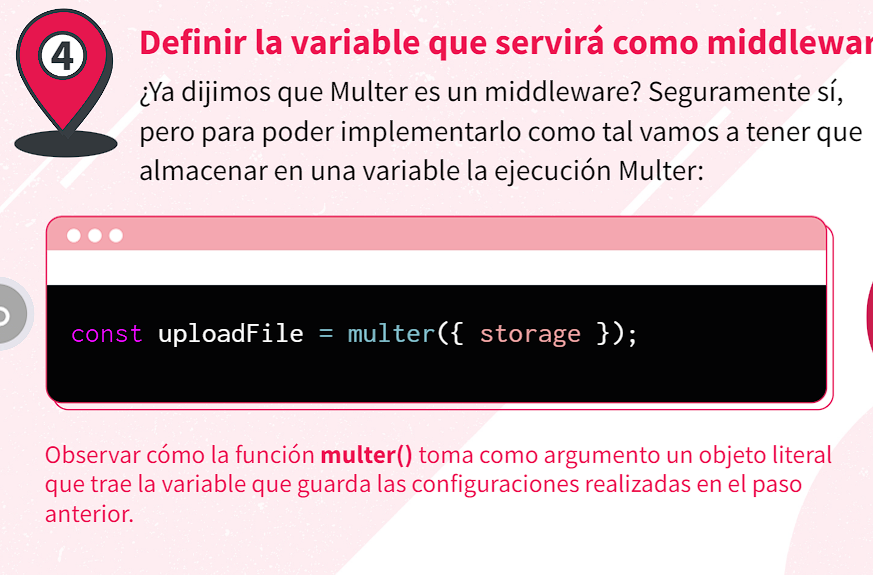


**FILENAME: tres parámetros (req, file y callback)** representan lo mismo que en destination. Con filename le indicaremos el nombre del archivo que vamos a guardar: 

El armado del nombre puede ser como uno quiera, otro ejemplo:





Especificar a multer la configuración anterior como disco de almacenamiento de archivos



Asi tendremos todo listo para procesar y almacenar los archivos.

¿Cómo hacemos para indicarles cual campo de archivo queremos procesar?



‘Avatart’ es lo que tiene en atributo name=”” en el from.

Código de ejemplo: <https://github.com/codelando/dh-node-multer-subiendo-archivos>

3.

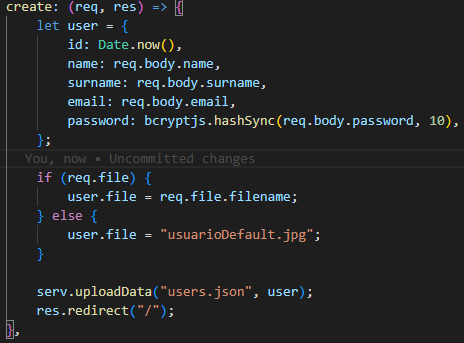
Validación de los datos:

En el momento de almacenar los datos del formularios sabemos que vienen por el **req.body**, pero la imagen del usuario viene por el **req.file**.

Aquí haremos la lógica para validar si el formato del archivo que nos envían es aceptado o si no viene nada por parte del usuario

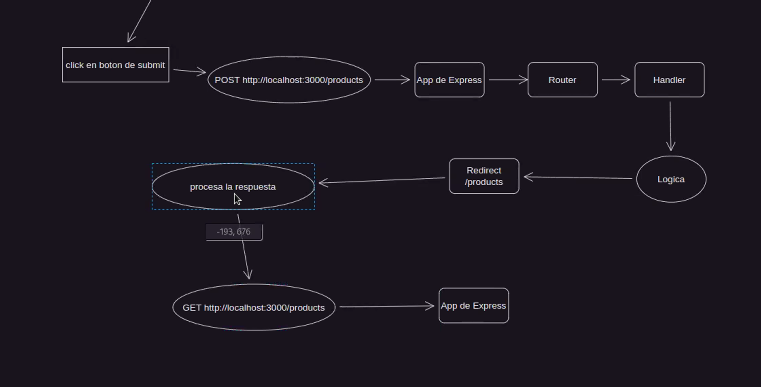


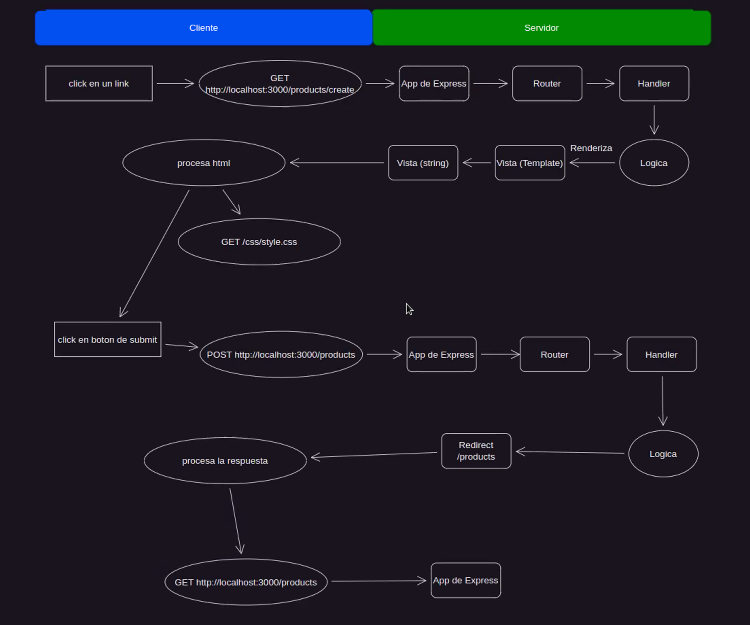
Otro ejemplo



**Nota de la clase:**

Un. Rendirect te manda una **sugerencia**, si le digo que redirija a otra ruta el navegador va a ser un get a esa ruta





Qué métodos de **fs** se usan para escribir un archivo?

Tenemos dos métodos que sirven para escritura en fs (writeFile y writeFileSync), pero normalmente usamos writeFileSync.

¿Qué métodos usamos para subir archivos con Multer?

Si queremos subir una sola imagen, el método single() es el correcto. Pero si tenemos muchas, usamos any() o array().