El siguiente documento es a modo de resumen para consultar la documentación ingresar en:
https://www.npmjs.com/package/multer

1 - Primero instalamos el paquete

```
npm install multer --save
```

2 - Configuración en el formulario

```
<form method="POST" action="" enctype="multipart/form-data">
    <input type="file" name="avatar"></input>
</form>
```

La imagen muestra los requisitos mínimos que debe tener un formulario para poder subir un archivo:
=> en la etiqueta <form> se debe ingresar el atributo *enctype="multipart/form-data"*
=> debe haber como mínimo una etiqueta *<input type="file"></input>*
=> En caso de querer permitir la subida de mas de un archivo, se debe escribir el html de la siguiente manera:
<input type="file" multiple></input>
=> En caso de querer limitar los formatos a subir se debe usar el atributo accept:
<input type="file" accept="image/png, image/jpeg, image/gif"></input> En este caso acepta solo png, jpeg y gif
*<input type="file" accept="image/**"></input>* En este caso acepta solo imagenes, pero cualquier formato de ellas

3 - Congifuración en el router

```
let storage = multer.diskStorage({
    destination: function (req, file, cb) {
      cb(null, 'public/images/products')
    },
    filename: function (req, file, cb) {
      cb(null, file.fieldname + '-' + Date.now() + path.extname(file.originalname))
    }
  })

let upload = multer({ storage: storage })
```

El método diskStorage de multer permite definir la ubicación (destination) donde se guardaran los archivos, y el nombre (filename) con el cual se guardaran.

El primer parámetro de diskStorage es *destination* en el cual solo se debe modificar la ruta, en el segundo parámetro de cb.

El segundo parámetro de diskStorage es *filename* en el cual el segundo parámetro de cb define el nombre a asignar a cada archivo. Si se toma el ejemplo de la imagen, los archivos se guardan con la concatenación del nombre del campo del formulario, luego un guión medio y el número asociado a la fecha actual (esto permite que ningún archivo tenga el mismo nombre), y luego la extensión del archivo subido.

4 - Congifuración en la ruta

Multer funciona como los middlewares. Por lo tanto al definir una ruta de un formulario que tiene que guardar un archivo en el servidor, se debe hacer como si fuera un middleware, como se muestra en la siguiente imagen:

```
router.post('/', upload.any(),productController.createConfirm);
```

Podemos ver que el *upload.any( )* es la sintaxis para invocar a multer y guardar el archivo. Podemos ver que *any( )* tiene la sintaxis de un método de *upload*. Esto es porque es un método que acepta cualquier archivo que se suba. Pero puede usarse otro método.

5 - Acceso desde el controlador

Hasta el momento configuramos multer para subir archivos al servidor, y aunque haya errores en el controlador, los archivos se suben.
Si nosotros queremos acceder a la información subida en el controlador, debemos saber que los datos de multer no viajan por *req.body*, sino que viajan por *req.files*.
En la imagen se muestra el *req.files* al subir un único archivo.
Si subimos mas de un archivo al mismo tiempo, será un array y cada archivo será una posición distinta, pero internamente tienen la misma forma.
Si subimos un único archivo, *req.files* es igualmente un array, por lo que para acceder a cualquier atributo es necesario hacer *req.files[0].atributo*


```
[
  - {
        fieldname: "image",
        originalname: "cv.jpg",
        encoding: "7bit",
        mimetype: "image/jpeg",
        destination: "public/images/products",
        filename: "image-1613603932722.jpg",
        path: "public\images\products\image-1613603932722.jpg",
        size: 25166,
    }
]
```