# SUBIDA DE ARCHIVOS

“

Cuando queremos subir **archivos** a través de un formulario para **almacenar** en un servidor, se requiere de un **proceso** específico para poder lograrlo.

# CONFIGURAR EL FORMULARIO

El primer paso para **subir** un archivo es **configurar** el formulario en la estructura html para que tenga todo lo necesario para que el mismo viaje hasta el servidor. Esos requerimientos son:

- Agregar el **input** de tipo *file* que le permite al usuario subir un archivo
- Agregar en la etiqueta **form** el **atributo** enctype="multipart/form-data"

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

El paquete `multer` trae consigo funcionalidades que nos van a permitir procesar y almacenar esta subida de archivos.

Para usarlo hay que instalarlo a través de npm.

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

# .diskStorage()

Es un **método** que trae el paquete `multer` que nos va a permitir operar con los archivos que estemos queriendo procesar. Recibe como parámetro un **objeto literal** con dos propiedades:

- `destination`, donde definiremos la carpeta donde se va a almacenar el archivo
- `filename`, donde indicaremos con qué nombre se guardará ese archivo en el servidor

Ambas propiedades son funciones. Cada una de ellas recibe tres parámetros: el **request**, el **archivo** y un **callback**.

# { código }

```
var storage = multer diskStorage({
  destination: (req, file, cb) => {
      . . .
  },
  filename: (req, file, cb) => {
      . . .
  }
})
```

# destination

En **destination**, usaremos el *callback* para definir la carpeta en donde queremos almacenar los archivos. El *primer parámetro* será `null`, el *segundo*, la **ruta** hacia la carpeta de destino.

{ }

```
var storage = multer.diskStorage({
      destination: (req, file, cb) => {
            cb(null, 'public/img/avatars')
      },
      filename: (req, file, cb) => {
            ...
      }
})
```

# filename

En **filename**, usaremos el *callback* para definir el nombre con el que guardaremos el archivo. El *primer parámetro* será `null`, el *segundo*, la **nombre**. Aquí usaremos la variable *file* junto con el paquete `path`.

Para crear el nombre del archivo, usaremos el método `extname()` del paquete, pasándole como parámetro el nombre original del archivo para que nos devuelva únicamente su extensión.

{ }

```
file.fieldname + '-' + Date.now() + path.extname(file.originalname)
```

En la documentación oficial de la librería, podemos encontrar toda esta **configuración** ya definida para *copiar e incorporar* a nuestro proyecto.

# { código }

```
var storage = multer.diskStorage({
  destination: (req, file, cb) => {
      cb(null, 'public/img/avatars')
  },
  filename: (req, file, cb) => {
      cb(null, file.fieldname + '-' + Date.now() + path.extname(file.originalname))
  };
})
var upload = multer({ storage: storage });
```

Deberemos incorporar este código en **cada archivo** en donde exista una **ruta** que esté **implementando** la **subida de archivos**.

# CONFIGURAR LAS RUTAS

A la ruta que se encargue de manejar la *petición de la subida de archivos*, habrá que pasarle un **nuevo parámetro** antes del callback: el método `upload.any()`.

```
app.post('/', upload.any() , (req, res) => {
    ...
})
```

Adicionalmente, habrá que pasarle un **tercer parámetro** al callback -*comúnmente nombrado* ***next***- para que todo funcione.

```
app.post('/', upload.any() , (req, res, next) => {
    ...
})
```

# PASO A PASO

- Agregar un `<input type="file">` en el formulario
- Agregar el atributo `enctype="multipart/form-data"` en la etiqueta `<form>`
- Instalar **multer**
- Requerir el módulo e Implementar la configuración del método `diskStorage()` *-extraída de la documentación oficial de multer-* en los archivos de rutas que lo requieran
- En la propiedad `destination`, configurar la **ruta** de la carpeta en donde se almacenarán los archivos

# PASO A PASO

- En la propiedad `filename`, configurar el **nombre** con el que se guardará el archivo
  - Usar `path.extname(file.originalname)` para obtener la extensión del archivo
- Agregarle `upload.any()` como segundo parámetro -*antes del callback*- a todas las rutas que implementen subida de archivos
- Agregar la variable `next` como tercer parámetro al callback

A cada **ruta** dentro de la aplicación que esté **procesando archivos**, deberemos configurarle estos parámetros para que implemente toda la **funcionalidad** de multer.