Subir Imagenes (Multer & Cloudinary)

Índice

Introducción

Con estas dos herramientas, podremos subir imágenes a Cloudinary con el objetivo de obtener el enlace público para acceder a ellas desde cualquier lugar. Utilizaremos `Multer` para recibir los archivos de fotos y, con la conexión a `Cloudinary`, podremos subirlas.

Crear Cuenta (Cloudinary)

Primero, debemos crear una cuenta en `Cloudinary` para poder subir nuestras fotos. Es recomendable hacerlo utilizando GitHub.

URL: https://console.cloudinary.com/users/register_free

Instalación

Para instalar `Multer` y `Cloudinary`, deberemos ejecutar los siguientes comandos en la consola (dentro de un proyecto de Node.js):

```
npm i cloudinary
npm i multer
```

Configuracion (Multer)

Para configurar `Multer`, primero debemos crear una carpeta donde se almacenarán las imágenes localmente (ya sea de forma temporal o permanente). Esta carpeta debe estar ubicada dentro del proyecto, ya sea dentro de la carpeta src o en una carpeta separada.

La configuración de `Multer` debe estar en el router de la ruta donde deseamos trabajar. Por ejemplo, si queremos gestionar la subida de fotos de perfil, configuraremos `Multer` en el router de usuarios.

Primero, debemos importar el módulo de `Multer`:

```
import multer from 'multer';
```

A continuación, debemos realizar la configuración de `Multer`. El código debería verse así:

```
const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);

// Poner la ubicacion de la carpeta de Uploads correspondiente, en este caso se ubica dentro del SRC
```

```
const uploadDir = join(__dirname, "../uploads");

// Se define donde se va a ubicar el archivo que vamos a subir y el nombre, este se puede modificar, en este cas
const storage = multer.diskStorage({
    destination: function (req, file, cb) {
        cb(null, uploadDir);
    },
    filename: function (req, file, cb) {
        cb(null, `${Date.now()}-${file.originalname}`)
    }
});

// El siguiente filtro es para que se suban unicamente archivos con extensiones especificas. En este caso serian
const fileFilter = (req, file, cb) => {
    const allowedTypes = ['image/png', 'image/jpeg', 'image/jpg'];
    if (allowedTypes.includes(file.mimetype)) {
        cb(null, true);
    } else {
        cb(new Error('Invalid file type. Only PDF, PNG, JPEG, and JPG files are allowed.'), false);
    }
};

const upload = multer({
    storage: storage,
    fileFilter: fileFilter
});
```

Conexion (Cloudinary)

Para realizar la conexión con `Cloudinary`, debemos acceder a nuestro dashboard personal de `Cloudinary` en el siguiente enlace para obtener nuestras credenciales.

Desde ahí, podemos acceder a nuestro Cloudname, que se encuentra en el centro de la página, y debemos copiarlo. A continuación, hacemos clic en el botón 'Go to API Keys', ubicado a la derecha del Cloudname, donde encontraremos la API Key y el API Secret, los cuales también debemos copiar.

En nuestro proyecto, debemos crear un archivo llamado `upload.js`, donde configuraremos toda la conexión.

```
import { v2 as cloudinary } from 'cloudinary';

cloudinary.config({
  cloud_name: "<cloudname>",
  api_key: "<api_key>",
  api_secret: "<api_secret>"
});

export default cloudinary;
```

Almacenamiento de Fotos (Local)

Para que los archivos se suban localmente, primero debemos definir desde qué ruta lo haremos, configurándolo en el router correspondiente. El siguiente ejemplo es para la creación de usuarios, pero puede modificarse para otras rutas según sea necesario.

```
// Crear Usuarios
router.post("/registerUsers", upload.single('file'), registerController.registerUsers);
```

De esta manera, cuando se suban las imágenes, se guardarán en la carpeta `uploads` que hemos creado.

Además, desde el controlador también podemos controlar las extensiones de los archivos subidos utilizando el siguiente código:

```
// Crear Usuarios
const imageFile = req.file.path;

const extension = imageFile.split('.').pop();
const extensionesPermitidas = ['pdf', 'png', 'jpeg', 'jpg'];

if (!extensionesPermitidas.includes(extension)) {
    console.error('Extensión de archivo no permitida');
    return res.status(400).send('Error: Extensión de archivo no permitida. Extensiones admitidas: PDF, PNG, JPEG
}
```

Subida a la nube con Cloudinary

Para subir las imágenes a `Cloudinary`, debemos importar la configuración desde el archivo que habíamos creado (`upload.js`) utilizando el siguiente código en el controlador donde vamos a trabajar:

```
import cloudinary from '../upload.js';
```

Para subir las imágenes a `Cloudinary`, dentro del controlador donde estemos trabajando, debemos usar el `imageFile` que estábamos gestionando con `Multer`. Luego, debemos ejecutar la siguiente función para subir la imagen, la cual nos devolverá un JSON con la información. Con esta información, podemos almacenar la URL de donde se ubicará la imagen:

```
const imageFile = req.file.path;

const result = await cloudinary.uploader.upload(imageFile, {
    folder: 'analisis',
});

const imageUrl = result.secure_url;
```

A continuación, podemos guardar el enlace en la base de datos de la siguiente forma:

```
const query = 'INSERT INTO public.users (imagen) VALUES ($1)';

await client.query(query, [imageUrl]);
```

Archivos temporales con Multer (para subir a la nube)

Para que los archivos sean temporales, simplemente importa el módulo `fs` en el controlador y ejecuta la función `unlinkSync` para eliminar los archivos después de que ya hayan sido procesados. Aquí tienes un ejemplo de cómo hacerlo:

```
import fs from "fs";

const imageFile = req.file.path;

const result = await cloudinary.uploader.upload(imageFile, {
    folder: 'analisis',
});

const imageUrl = result.secure_url;

const query = 'INSERT INTO public.users (imagen) VALUES ($1)';

await client.query(query, [imageUrl]);

fs.unlinkSync(imageFile); // Eliminar el archivo local
```