```
//UPLOAD FRONTEND: klik na upload button – zabiera files z input[type=file].files I
apenduje je do nowotworzonego form data. My files to name dla tej paczki.
    if(e.target.classList.contains('img-filedrop-add-btn')){
        const formData = new FormData()

        for(const file of imgInput.files){
            formData.append('myFiles',file)
        }
        for(const [key, value] of formData){
            console.log(`key: ${key}`);
            console.log(`value: ${value}`);
        }
        console.log(formData);
        fetch('http://localhost:5000/upload-images', {
            method: 'POST',
            body: formData,
            "content-type": "multipart/form-data"
        })
        .then(()=>{
            //no errors, show message to the user
```

BACKEND: multer konfig:

```
//multer file upload
const multer = require('multer')
const storage = multer.diskStorage({
  destination: (req, file, cb)=> {   definiuje folder do zapisywania
    cb(null, "public/img/img-large")
  },
  filename: (req, file, cb)=> { definiuje nazwe pliku
    const {originalname} = file
    cb(null, originalname)
  }
})
const upload = multer({ storage }); ustawia parameter upload.storage = storage z
powyzszego consta
```

I banalny route: ktory uzywa jako middleWare funkcji UPLOAD.ANY(), a sam tylko wysyła res.message

```
router.post('/upload-images', upload.any(), (req, res, next)=>{
    console.log(req.files);
    res.status(201).send('files received')


})
```

Na frontendzie pliki mozna przesylac albo przez fetch i w body przeslac plik poprzez konstruktor FormData:

```
//UPLOAD
    if(e.target.classList.contains('img-filedrop-add-btn')){
       const formData = new FormData()

       for(const file of imgInput.files){
           formData.append('myFiles',file)
       }
       for(const [key, value] of formData){
           console.log(`key: ${key}`);
           console.log(`value: ${value}`);
       }
       console.log(formData);
       fetch('http://localhost:5000/upload-images', {
           method: 'POST',
           body: formData,
           "content-type": "multipart/form-data"
       })
       .then(()=>{
```

Albo można to zrobic przy uzyciu formularza z metoda post, i atrybutem enctype=”multipart/form-data”

Wtedy plik przesle się nie przez req.body tylko req.file.

UNIKALNY FILENAME:

```
 cb(null,  file.filename)
```

filename to hasz ktory multer stworzyl przy parsowaniu, unikalny. Original name to nazwa pliku jaki wyslal uzytkownik. Jak jest name clash to nadpisze stary plik. Dlatego można dodac hasz do nazwy i wtedy kazy plik będzie zawsze unikalny (jeśli tego chcemy). Jeśli uzyjemy original name to multer nie generuje swojego haszu wiec trzeba dodac np. z daty.

```
   cb(null, originalname + '-' + new Date().toISOString())
```

kolejna opcja multera to filtr plikow. Tutaj zrobione jest to na frontendzie już, ale gdyby nie było to można przez multer ustawic jakie typy mimetype przyjmie server:

```
  fileFilter: (req, file, cb) => {
    if(file.mimetype === 'image.png' || file.mimetype === 'image.jpeg'){
      cb(null, true) >>gitara, zapisujemy
    }else{
      cb(null, false) >> error zly format
    }
  }
```