Stage 2022

-

Gelles Julien

Compte-Rendu n°5

-

Semaine du 16/05 au 20/05

LUXONDES

See the electromagnetic waves

Travaux réalisés :

1. Dézippage d’archive

Avec l’aide de Julien Wylleman, j’ai donc réussi à dézipper une archive qui était uploadée sur la page, envoyé vers le serveur, dézippée à cette endroit puis renvoyée à la page.
Pour se faire il a fallut utiliser une fonction fetch coté front pour envoyer les données au serveur et une fonction application.post du coté *backend* pour récupérer, traiter et ensuite renvoyer les données à la page avec le bon url.

```
93      fetch("./upload_files", {
94        method:'POST',
95        body: formData
96      })
97        .then((res) =>
98        {
99          return res.json(); // réponse format json
100       })
101       .then(json =>
102       {
103         loadFiles(json.files); // chargement fichiers
104       })
105       .catch((err) => ("Submit Error", err)); // retour d'erreur
106   }
107
108   function loadFiles(urls)
109   {
```

Fonction fetch en frontend

```
29  // On traite les requetes POST vers l'URL upload_files,
30  app.post("/upload_files",
31    upload.array("files"), // copie des fichiers par multer dans son répertoire de travail
32    uploadFiles);
33
34    function uploadFiles(req, res)
35    {
36      // fonction de décompression
37      unzipRec(req.files, res);
38    }
39
40    function unzipRec(files, res, rep=[])
41    {
42      if(0 == files.length)
43        // envoie de la liste d'url en front
44        res.json({success:true, files:rep});
45      else
46      {
47        let file = files.shift();
48        // dézippage
49        decompress(file.path, "public/files").then(unzippedFiles =>
50        {
51          unzippedFiles.forEach(unzippedFile =>
52          {
53            rep.push('files/'+unzippedFile.path); // ajout des url à la liste d'url
54          });
55          unzipRec(files, res, rep); // appel récursif
56        });
57      }
58    }
```

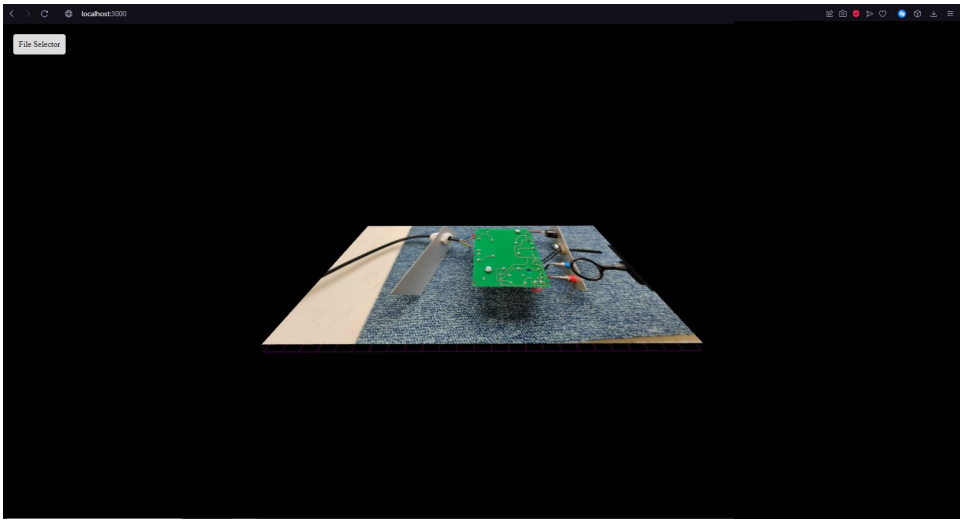
Fonction post en backend

2. Utilisation des données

Suite à cela, il a donc été possible d'utiliser les données et notamment de les afficher sur la page, comme montré ci dessous, dans le menu :

Menu modifié avec données img et xml affichées

Il a d'autant plus été possible de faire une première modification de l'image présente sur le plan à l'aide de l'upload d'une archive :



Modification du plan avec nouvelle image

3. Supression des fichiers du dézippage

Enfin pour éviter une accumulation des fichiers dézipper dans le dossier du projet, j'ai crée une fonction, de la même façon que pour le dézippage, avec une méthode fetch puis post afin d'envoyer un signal au serveur pour supprimer les fichiers à chaque rafraîchissement de la page ou bien à sa fermeture.

```
197     async function deleteSignal(){
198       console.log('del');
199       fetch("./delete_signal", { // envoi du signal coté serveur pour supprimer fichiers uploadés
200         method:'POST',
201         body: "delete_signal"
202       })
203         .catch((err) => ("Submit Error", err)); // retour d'erreur
204     }
205
206     window.onbeforeunload = deleteSignal; // delete avant chaque fermeture page (ou F5)
```

Fonction coté frontend pour l'envoi du signal de suppression

```
60  //supprime fichiers lorsque l'on quitte ou refresh la page
61  app.post("/delete_signal", () => {
62    // supression ./public/files
63    if (fs.existsSync('./public/files')) fs.rm('./public/files', { recursive: true, force: true }, (err) => {
64      if (err) throw err });
65
66    // supression ./uploads
67    if (fs.existsSync('./uploads')) {
68      fs.rmSync('./uploads', { recursive: true, force: true }); // supprime dossier
69      fs.mkdir('./uploads', (error) => { // puis on le recrée vide pour le bon fonctionnement de multer
70        if (error) console.log(error);
71      });
72    }
73  });
```

Fonction coté serveur pour la supression des fichiers