Gestions d'images

L'application que vous devez créer doit permettre la gestion d'images.

La base de données

Les données doivent être stockées dans une base de données postgreSQL comprenant 2 tables :

Une table « pictures » avec

- Un id
- Un nom (un texte quelconque)
- Un lien (le nom du fichier)
- Une liste de noms-clés séparés par des points-virgules.
- Un nombre de likes

Une table « comments » avec les commentaires sur les images avec

- Un id
- Un texte
- Une date d'ajout
- L'id de l'image sur laquelle porte le commentaire

Le backend

Le backend, écrit avec Express.js doit disposer des points d'entrée suivant :

- « GET /pictures » : retourne l'ensemble des données de la table « pictures »
- « GET /pictures ?keyword=toto » : retourne les images dont l'un des mots-clés est « toto »
- « GET /comments/n » : retourne les commentaires associés à l'image « n » dans l'ordre chronologique
- « POST /pictures » : ajoute une image, c'est-à-dire une ligne dans la table « pictures » et le fichier dans un répertoire dont le nom est « public »
- « REMOVE /pictures/n » : supprime l'image dont l'id est « n », c'est-à-dire la ligne dans la table « pictures » et le fichier dans le répertoire « public »
- « PATCH /pictures/n/comments » : ajoute un commentaire pour l'image « n »

D'autres routes peuvent être ajoutées si besoin

Le Frontend

Le frontend, écrit avec React.js, doit permettre de :

- Visualiser les images avec éventuellement un filtre par mot-clé
- Visualiser une image unique avec ses commentaires
- Ajouter une image
- Supprimer une image
- Commenter une image

Options

S'il vous reste du temps, vous pouvez :

- Afficher les images par 3, avec un bouton « précédent » et « suivant » pour limiter le nombre de données extraites depuis le serveur
- Permettre d'ajouter un « like » à une image et comptabiliser les likes

Aide

Afficher une image stockée côté *backend*

Sur le *backend*, on peut créer un répertoire particulier, appelons-le « img » dans lequel on stocke les fichiers avec nos images.
Pour accéder à cette image depuis un navigateur, il faut écrire le code suivant dans le fichier « server.js » :

```
const fileUpload = require('express-fileupload');
…
app.use('/pix, express.static(img))
    .use(fileUpload({
        useTempFiles: true,
        tempFileDir: '/tmp/'
    }))
```

Ainsi, dès qu'une route commence par « /pix » (par exemple (« /pix/image01.png ») le serveur recherche le fichier qui apparait dans la route (ici « image01.png ») dans le répertoire « img » et le retourne.

Côté *frontend*, l'affichage d'une image stockée côté *backend*, se fait à l'aide d'une balise <img> classique. Par exemple :

```
  <img src="http://localhost:8000/pix/image01.png" alt="no img"/>
```

Enregistrer une image côté *backend*

Pour enregistrer une image, il faut d'abord créer un formulaire côté *frontend*, puis l'envoyer au *backend* à l'aide d'une requête *axios*.

Le formulaire

```
<form onSubmit={e => addPicture(e)}>
      <p><input id="picturename" required="required"/></p>
      <p><input id="fileField" type="file" accept=".png,.jpg,.jpeg"/></p>
      <div>
            <button type="submit">Create</button>
      </div>
</form>
```

Remarquez le type du champ input qui permet de prendre un fichier.
Il faut définir la fonction *addPicture* :

La fonction associée

```
async function addPicture(e) {
        e.preventDefault();
        const selectedFile = e.target.fileField.files[0];
        const data = new FormData();
        data.append('file', selectedFile, selectedFile.name);
        data.append('name', e.target.picturename.value);
        await axios.post('http://localhost:8000/pix, data);
        await loadPictures();
    }
```

Remarquez l'utilisation d'un objet FormData pour transmettre les données au *backend*.

La route côté backend

```
.post('/pictures',
    async (req, res) => {
        console.log("files", req.files);
        console.log("body", req.body);

        await db.query('insert into pictures(name,link) values($1,$2)',
            [req.body.name, req.files.file.name]
        );
        req.files.file.mv(__dirname + '/img/' + req.files.file.name,
            (err) => {
                if (err) {
                    console.error(err);
                    return res.status(500).send(err);
                }
                res.status(200).end();
            }
        );
    });
```

Remarques :

- L'objet *req* contient les données du formulaire
- `req.files.file.mv` Permet de placer le fichier dans le répertoire voulu
- `__dirname` est une variable spéciale qui correspond au nom du répertoire courant sur le *frontend*

Supprimer une image côté *frontend*

Pour cela, on utilise le module *filesystem* défini dans node.js.
Voir https://nodejs.org/api/fs.html

```
const fs = require('fs');
```

Permet d'utiliser le module (pas de npm install)

```
fs.unlink(__dirname + '/img/image01.png, err => err && console.error(err))
```

Permet de supprimer le fichier