

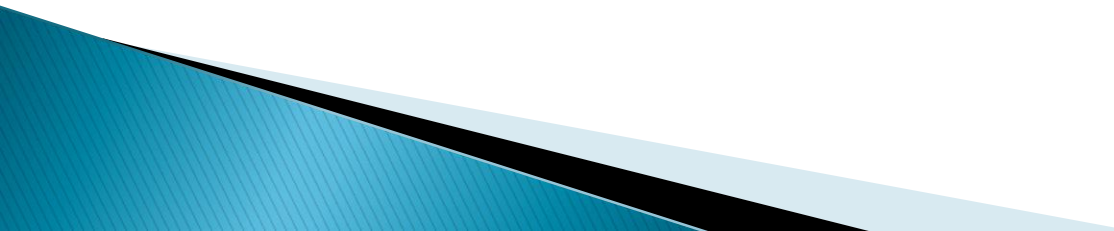
Créez un réseau social d'entreprise

Gwenaël GRESSIER

Introduction

- ▶ Pour ce projet j'incarnerais un développeur en poste depuis plus d'un an chez Connect-E une agence web
- ▶ Ou j'ai été missionné pour créer un réseau social interne pour les employés de Groupomania. Le but de cet outil est de faciliter les interactions entre collègues.

Mon Projet MERN

- ▶ Pour ce projet j'ai choisie une approche dite MERN stack ce qui veut dire que je vais faire appelle à:
 - ▶ MongoDB
 - ▶ Express
 - ▶ React
 - ▶ Node.js
- 

Les spécifications de l'API auth

	Point d'accès	Request body (le cas échéant)	Type de réponse attendue	Fonction
post	/api/user/register	{ pseudo, email, password }	{ user: user._id }	Crée notre user dans la base de donne et lui attache un id
post	/api/user/login	{ email, password }	Cookie{token} { user: user._id }	Cree un token en cookie et me log avec mon id
get	/api/user/logout		Cookie{« »}	Suprime le token des cookie

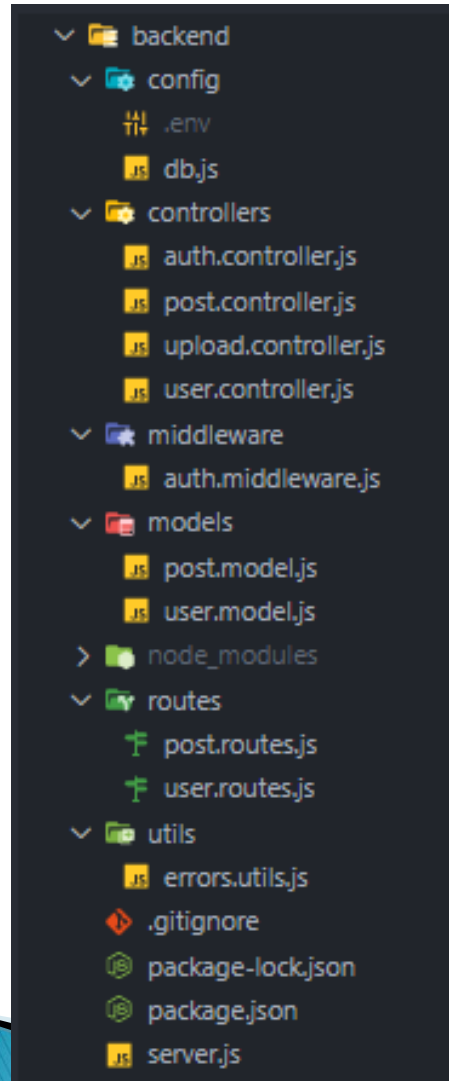
Les spécifications de l'API users

	Point d'accès	Request body (le cas échéant)	Type de réponse attendue	Fonction
get	/api/user/		Array des user	Affiche tout les user
get	/api/user/:id		un user	Affiche un user
put	/api/user/:id		un user modifié	Modifie la fiche user
delete	/api/user/:id		{ message }	Supprime un user

Les spécifications de l'API post

	Point d'accès	Request body (le cas échéant)	Type de réponse attendue	Fonction
post	/api/post			
get	/api/post			
put	/api/post/:id			
delete	/api/post/:id			
patch	/api/post/like-post/:id			
patch	/api/post/unlike-post/:id			
patch	/api/post/comment-post/:id			
patch	/api/post/edit-comment-post/:id			
patch	/api/post/delete-comment-post/:id			

Le code



Config

Ici sera stock tout les donner sensible et la connexion à ma base de donne

Controlers

C'est mon code en gros qui dit qui fait quoi

Middleware

Se sont mes autorisation

Models

Sert à définir mes schémas (interfaces) post et user

Routes

Définit les règles du CRUD

Utils

Va me servir à définir les messages d'erreur

server.js

Définit mes routes et mes headers(CORS)

Back-end/server.js

```
1  const express = require("express");
2  const cookieParser = require("cookie-parser");
3  const cors = require("cors");
```

Importation des
package de node.js

```
14  const corsOptions = {
15    origin: process.env.CLIENT_URL,
16    credentials: true,
17    allowedHeaders: ["sessionId", "Content-Type"],
18    exposedHeaders: ["sessionId"],
19    methods: "GET,HEAD,PUT,PATCH,POST,DELETE",
20    preflightContinue: false,
21  };
```

Définition des CORS
de la ligne 14 à 21

```
28  // jwt token
29  app.get("*", checkUser);
30  app.get("/jwtid", requireAuth, (req, res) => {
31    res.status(200).send(res.locals.user._id);
32  });
33
34  // routes
35  app.use("/api/user", userRoutes);
36  app.use("/api/post", postRoutes);
```

Gestion de la
vérification et de la
gestion des tokens

principaux chemin
de l'API: user et post

Back-end/routes.js

user

post

Importation des Package et des routes. Et c'est ici que je fait la définition des chemins avec ou sans autorisation.

```
1 const router = require("express").Router();
2 const postController = require("../controllers/post.controller");
3 const multer = require("multer");
4 const upload = multer();
5
6 router.post("/", upload.single("file"), postController.createPost);
7 router.get("/", postController.readPost);
8 router.put("/:id", postController.updatePost);
9 router.delete("/:id", postController.deletePost);
10 router.patch("/like-post/:id", postController.likePost);
11 router.patch("/unlike-post/:id", postController.unlikePost);
12
13 router.patch("/comment-post/:id", postController.commentPost);
14 router.patch("/edit-comment-post/:id", postController.editCommentPost);
15 router.patch("/delete-comment-post/:id", postController.deleteCommentPost);
16
17 module.exports = router;
```

```
1 const router = require("express").Router();
2 const authController = require("../controllers/auth.controller");
3 const userController = require("../controllers/user.controller");
4 const uploadController = require("../controllers/upload.controller");
5 const multer = require("multer");
6 const upload = multer();
7
8 // auth
9 router.post("/register", authController.signUp);
10 router.post("/login", authController.signIn);
11 router.get("/logout", authController.logout);
12
13 // users
14 router.get("/", userController.getAllUsers);
15 router.get("/:id", userController.userInfo);
16 router.put("/:id", userController.updateUser);
17 router.delete("/:id", userController.deleteUser);
18 router.patch("/follow/:id", userController.follow);
19 router.patch("/unfollow/:id", userController.unfollow);
20
21 // upload
22 router.post("/upload", upload.single("file"), uploadController.uploadProfile);
23
24 module.exports = router;
```

Back-end / utils / errors-utils.js

```
1 //messages d'erreur à la création de profil
2 module.exports.signUpErrors = (err) => {
3   let errors = { pseudo: "", email: "", password: "" };
4   //Le pseudo n'est pas supporté
5   if (err.message.includes("pseudo")) errors.pseudo = "Pseudo invalide";
6   //L'email n'est pas valide en temps qu'email
7   if (err.message.includes("email")) errors.email = "Email invalide";
8   //mot de passe trop court ou trop long
9   if (err.message.includes("password"))
10     errors.password = "Le mot de passe doit faire 6 caractères minimum";
11   /*Le code d'erreur 11000 s'affiche quand un élément unique existe déjà,
12   on distingue simplement les pseudo et mails dans ce cas de figure*/
13   if (err.code === 11000 && Object.keys(err.keyValue)[0].includes("pseudo"))
14     errors.pseudo = "Ce pseudo est déjà enregistré";
15
16   if (err.code === 11000 && Object.keys(err.keyValue)[0].includes("email"))
17     errors.email = "Cet email est déjà enregistré";
18
19   return errors;
20 };
21 //erreurs à la connexion
22 module.exports.signInErrors = (err) => {
23   let errors = { email: "", password: "" };
24
25   if (err.message.includes("email")) errors.email = "Email Inconnu";
26
27   if (err.message.includes("password"))
28     errors.password = "Le mot de passe ne correspond pas";
29
30   return errors;
31 };
32
33 //erreurs d'upload
34 module.exports.uploadErrors = (err) => {
35   let errors = { format: "", maxSize: "" };
36
37   if (err.message.includes("invalid file"))
38     errors.format = "Format incompatible";
39
40   if (err.message.includes("max size"))
41     errors.maxSize = "Le fichier dépasse 5000ko";
42
43   return errors;
44 };
```

Gestion des erreurs dans la partie inscription

Gestion des erreurs dans la partie connexion

Gestion des erreurs d'upload d'image

Back-end/models par exemple pour le user

```
5 //Schema de création d'une fiche utilisateur.
6 const userSchema = new mongoose.Schema(
7   {
8     pseudo: {
9       type: String,
10      required: true,
11      minLength: 3,
12      maxLength: 30,
13      unique: true,
14      trim: true,
15    },
16    email: {
17      type: String,
18      required: true,
19      validate: [isEmail],
20      lowercase: true,
21      unique: true,
22      trim: true,
23    },
24    password: {
25      type: String,
26      required: true,
27      max: 1024,
28      minlength: 6,
29    },
30    picture: {
31      type: String,
32      default: "../profil/random-user.png",
33    },
34    bio: {
35      type: String,
36      max: 1024,
37    },
38    followers: {
39      type: [String],
40    },
41    following: {
42      type: [String],
43    },
44    likes: {
45      type: [String],
46    },
47  },
48  {
49    timestamps: true,
50  }
51 );
```

Grace a l'importation du package mongoose pour la DB je peut faire la définition de mon schéma sous forme de JSON

Plusieurs type possible: String, number, Array...

Back-end/models/user.model.js

```
53 // fonction pour "saler"(crypter) les mots de passe.
54 userSchema.pre("save", async function (next) {
55     const salt = await bcrypt.genSalt();
56     this.password = await bcrypt.hash(this.password, salt);
57     next();
58 });
59
60 //fonction pour décrypter un mot de passe
61 userSchema.statics.login = async function (email, password) {
62     const user = await this.findOne({ email });
63     if (user) {
64         const auth = await bcrypt.compare(password, user.password);
65         if (auth) {
66             return user;
67         }
68         throw Error("incorrect password");
69     }
70     throw Error("incorrect email");
71 };
```

Utilisation du hook de mongoose .pre qui permet d'exécuter ma fonction avant mon user schema

Fonction qui permet de comparer mon mot de passe en fonction du mail renseigné

Back-end/middleware/auth.middleware.js

```
1 const jwt = require("jsonwebtoken");
2 const UserModel = require("../models/user.model");
3
4 module.exports.checkUser = (req, res, next) => {
5   const token = req.cookies.jwt; //recupere le token dans les cookies
6   if (token) {
7     jwt.verify(
8       token,
9       process.env.TOKEN_SECRET,
10      async (err, decodedToken) => {
11        if (err) {
12          res.locals.user = null;
13          res.cookie("jwt", "", { maxAge: 1 }); //si une err sur le token, on supprime le cookie
14          next();
15        } else {
16          let user = await UserModel.findById(decodedToken.id);
17          res.locals.user = user;
18          console.log(user);
19          next();
20        }
21      }
22    );
23  } else {
24    res.locals.user = null;
25    next();
26  }
27 };
28
29 //Verification de l'existence et de la validité du token de l'utilisateur
30 module.exports.requireAuth = (req, res, next) => {
31   const token = req.cookies.jwt;
32   if (token) {
33     jwt.verify(
34       token,
35       process.env.TOKEN_SECRET,
36      async (err, decodedToken) => {
37        if (err) {
38          console.log(err);
39          res.send(200).json("no token");
40        } else {
41          console.log(decodedToken.id);
42          next();
43        }
44      }
45    );
46  } else {
47    console.log("no token");
48  }
49 };
```

- ▶ Ces dans cette partie que je fait la gestion de mes tokens
- recupération
- suppression et
- validation

Back-end/controllers/auth.controllers.js

```
1 const UserModel = require("../models/user.model");
2 const jwt = require("jsonwebtoken");
3 const { signUpErrors } = require("../utils/errors.utils");
4 const { signInErrors } = require("../utils/errors.utils");
5
6 //temps en millisecondes: 1000ms(=1seconde) * 60s(=1minute) * 60min(=1heure) * 24heures. Le token sera donc valable durant 24h
7 const maxAge = 1000 * 60 * 60 * 24;
8 const createToken = (id) => {
9   return jwt.sign({ id }, process.env.TOKEN_SECRET, {
10     expiresIn: maxAge,
11   });
12 };
13
14 module.exports.signUp = async (req, res) => {
15   const { pseudo, email, password } = req.body;
16
17   try {
18     /*Lorsqu'on veut créer un utilisateur, ces trois éléments
19     obligatoires doivent être présents, sinon ça passe au catch en dessous.*/
20     const user = await UserModel.create({ pseudo, email, password });
21     res.status(201).json({ user: user._id });
22   } catch (err) {
23     const errors = signUpErrors(err);
24     res.status(200).send({ errors });
25   }
26 };
27
28 module.exports.signIn = async (req, res) => {
29   const { email, password } = req.body;
30
31   try {
32     const user = await UserModel.login(email, password);
33     const token = createToken(user._id);
34     res.cookie("jwt", token, { httpOnly: true, maxAge });
35     res.status(200).json({ user: user._id });
36   } catch (err) {
37     const errors = signInErrors(err);
38     res.status(200).json({ errors });
39   }
40 };
41
42 module.exports.logout = async (req, res) => {
43   //on attribue un cookie vide qui va vivre 1ms, puis rediriger l'utilisateur
44   res.cookie("jwt", "", { maxAge: 1 });
45   res.redirect("/");
46 };
47
```

temps de valider du token egal a 24h

Lorsque l'on veut crée un utilisateur vérifie que pseudo,email et password sont présent

Fonction de connexion

Fonction qui supprime le cookies et qui renvoi a la page principal

différent renvoie vers mon fichier utils/error

Back-end/controllers/upload.controllers.js

```
1 const UserModel = require("../models/user.model");
2 const fs = require("fs");
3 const { uploadErrors } = require("../utils/errors.utils"); //recup detail erreur dans utils/errors.utils.js
4
5 module.exports.uploadProfil = async (req, res) => {
6   try {
7     //verification du format du fichier (s'assurer que c'est une image)
8     if (
9       req.file.mimetype !== "image/jpg" &&
10       req.file.mimetype !== "image/png" &&
11       req.file.mimetype !== "image/jpeg"
12     ) {
13       throw Error("invalid file");
14     }
15     if (req.file.size > 5000000) throw Error("max size"); //verif du poids du fichier
16   } catch (err) {
17     const errors = uploadErrors(err);
18     return res.status(201).json({ errors });
19   }
20   const images = req.file.mimetype.split("/");
21   const extension = images.slice(-1).pop();
22   const fileName = req.body.name + "." + extension; //nouveau nom du fichier
23
24   //stockage de la nouvelle image.
25   fs.writeFile(
26     `${__dirname}/../../frontend/public/uploads/profil/${fileName}`,
27     req.file.buffer,
28     (err) => {
29       if (err) throw err;
30     }
31   );
32
33   try {
34     await UserModel.findByIdAndUpdate(
35       req.body.userId,
36       { $set: { picture: `./uploads/profil/${fileName}` },
37         { upsert: true, setDefaultsOnInsert: true },
38       },
39       { err, docs } => {
40         if (!err) return res.send(docs);
41         else return res.status(500).send({ message: err });
42       }
43     );
44   } catch (err) {
45     return;
46   }
47 }
```

Définition de mes
mimetype

Modification du nom
des images elle aurons
en nom 'pseudo'. 'type
de l'image'

Définition du lieu de
stockage de mon
image

Si une image est déjà
présente la modifie
grâce à \$set

Back-end/controllers/user.controller.js

```
1 const UserModel = require("../models/user.model");
2 const ObjectID = require("mongoose").Types.ObjectId;
3 const fs = require("fs");
4
5 //Obtenir les données de tous les utilisateurs
6 module.exports.getAllUsers = async (req, res) => {
7   const users = await UserModel.find().select("-password");
8   res.status(200).json(users);
9 };
10
11 //Obtenir les données d'un seul utilisateur
12 module.exports.userInfo = (req, res) => {
13   if (!ObjectID.isValid(req.params.id))
14     //Si l'id de la requête n'est pas valide, je m'arrête là, et je réponds avec une erreur
15     return res.status(400).send("ID unknown : " + req.params.id);
16
17   UserModel.findById(req.params.id, (err, docs) => {
18     /*Si l'id de la requête est valide je récupère les données de l'utilisateur concerné,
19     sauf le mot de passe, que je dois veiller à ne jamais envoyer dans le front.*/
20     if (!err) res.send(docs);
21     else console.log("ID unknown : " + err);
22   }).select("-password");
23 };
```

Récupère les données de tous les users sauf le mot de passe

Vérifie si l'id demandé est valide

Récupère les données de mon user sauf le mot de passe

Back-end/controllers/user.controller.js

```
25 //Updater le profil d'un utilisateur
26 module.exports.updateUser = (req, res) => {
27   if (!ObjectID.isValid(req.params.id))
28     return res.status(400).send("ID unknown : " + req.params.id);
29
30   try {
31     UserModel.findOneAndUpdate(
32       { _id: req.params.id },
33       {
34         $set: {
35           bio: req.body.bio,
36         },
37       },
38       { new: true, upsert: true, setDefaultsOnInsert: true },
39       (err, docs) => {
40         if (!err) return res.send(docs);
41         if (err) return res.status(500).send({ message: err });
42       }
43     );
44   } catch (err) {
45     console.log("docs");
46
47     return res.status(500).json({ message: err });
48   }
49 };
50
51 //Supprimer un utilisateur
52 module.exports.deleteUser = async (req, res) => {
53   if (!ObjectID.isValid(req.params.id))
54     return res.status(400).send("ID unknown : " + req.params.id);
55
56   try {
57     const docs = await UserModel.findById(req.params.id);
58     if (!docs) console.log("Image non supprimée");
59     else fs.unlink(docs.picture, () => {});
60   } catch (err) {
61     return console.error(err);
62   }
63
64   try {
65     await UserModel.deleteOne({ _id: req.params.id }).exec();
66     res.status(200).json({ message: "Utilisateur Supprimé." });
67   } catch (err) {
68     return res.status(500).json({ message: err });
69   }
70 };
```

Va chercher les info de mon user dans la Base de donnée et crée la bio ou la modifie

Pour la partie suppression des utilisateur

Supprime l'image de profil si il une image est présente avec fs.unlink

Supprime l'utilisateur grâce à .deleteOne

Post controller

```
8 //CRUD : Create
9 module.exports.createPost = async (req, res) => {
10   let fileName;
11   if (req.file) {
12     try {
13       /*verification du format du fichier
14       (s'assurer que c'est une image, et que son format est supporté)*/
15       if (
16         req.file.mimetype !== "image/jpg" &&
17         req.file.mimetype !== "image/png" &&
18         req.file.mimetype !== "image/jpeg" &&
19         req.file.mimetype !== "image/gif"
20       )
21         throw Error("invalid file");
22
23       //verif du poids du fichier
24       if (req.file.size > 5000000) throw Error("max size");
25     } catch (err) {
26       const errors = uploadErrors(err);
27       return res.status(201).json({ errors });
28     }
29     //nouveau nom du fichier
30     const images = req.file.mimetype.split("/");
31     const extension = images.slice(-1).pop();
32     fileName = req.body.posterId + Date.now() + "." + extension;
33
34     //stockage de la nouvelle image.
35     fs.writeFile(
36       `../frontend/public/uploads/posts/${fileName}`,
37       req.file.buffer,
38       (err) => {
39         if (err) throw err;
40       }
41     );
42   }
```

définition de mes
mimetype

Définit un poids
max de 5MO

Modifie le nom du
fichier

Stock l'image dans
le dossier
correspondant

Post controller

```
44 const newPost = new postModel({
45   posterId: req.body.posterId,
46   message: req.body.message,
47   picture: req.file ? `./uploads/posts/` + fileName : "",
48   likers: [],
49   comments: [],
50 });
51
52 try {
53   const post = await newPost.save();
54   return res.status(201).json(post);
55 } catch (err) {
56   return res.status(400).send(err);
57 }
58
59 //CRUD : Read
60 module.exports.readPost = (req, res) => {
61   PostModel.find((err, docs) => {
62     if (!err) res.send(docs);
63     else console.log("Error to get data : " + err);
64   }).sort({ createdAt: -1 });
65 }
66
67 //CRUD : Update
68 module.exports.updatePost = (req, res) => {
69   if (!ObjectID.isValid(req.params.id))
70     return res.status(400).send("ID unknown : " + req.params.id);
71
72   const updatedRecord = {
73     message: req.body.message,
74   };
75
76   PostModel.findByIdAndUpdate(
77     req.params.id,
78     { $set: updatedRecord },
79     { new: true },
80     (err, docs) => {
81       if (!err) res.send(docs);
82       else console.log("update error : " + err);
83     }
84   );
85 }
86
```

Creation de mon post avec son model .

Save le nouveau post dans la bdd

Fonction qui va recuperer mes post dans ma bdd

grâce au `.sort({creatAT:-1})` je vais pouvoir afficher les post dans l'ordre anté-chronologique

Recupere le message dans mon body

Cherche le message et re-inject le nouveau contenu

Post controller

```
88 //CRUD : Delete
89 module.exports.deletePost = (req, res) => {
90   if (!ObjectID.isValid(req.params.id))
91     //
92     return res.status(400).send("ID unknown : " + req.params.id); //
93
94   PostModel.findByIdAndRemove(req.params.id, (err, docs) => {
95     if (!err) {
96       fs.unlink(docs.picture, () => {});
97       res.send(docs);
98     } else console.log("Deleting error : " + err);
99   });
100 };
```

Cherche dans la base de donner le message par sont id et le supprime grace a findByIdAndRemove

Ensuite fait un fs.unlink de mon image afin de la supprimer

Post controler les like et unlike

```
102 //Like-post
103 module.exports.likePost = async (req, res) => {
104   if (!ObjectID.isValid(req.params.id))
105     return res.status(400).send("ID unknown : " + req.params.id);
106
107   try {
108     let updatedLikers = await PostModel.findByIdAndUpdate(
109       req.params.id,
110       /*j'ajoute avec $addToSet l'id dans le tableau likers du post*/
111       { $addToSet: { likers: req.body.id } },
112       //true pour renvoyer le document modifié
113       { new: true }
114     );
115     res.json({ updatedLikers });
116     let updatedLikes = await UserModel.findByIdAndUpdate(
117       req.body.id,
118       { $addToSet: { likes: req.params.id } },
119       { new: true }
120     );
121     res.json({ updatedLikes });
122   } catch (err) {
123     return;
124   }
125 };
126
127 //unlike-post
128 module.exports.unlikePost = async (req, res) => {
129   if (!ObjectID.isValid(req.params.id))
130     //
131     return res.status(400).send("ID unknown : " + req.params.id);
132
133   try {
134     let updatedLikers = await PostModel.findByIdAndUpdate(
135       req.params.id,
136       { $pull: { likers: req.body.id } },
137       { new: true }
138     );
139     res.json({ updatedLikers });
140     let updatedLikes = await UserModel.findByIdAndUpdate(
141       req.body.id,
142       { $pull: { likes: req.params.id } },
143       { new: true }
144     );
145     res.json({ updatedLikes });
146   } catch (err) {
147     return;
148   }
149 };
```

Grace à la commande `addToSet` j'ajoute la nouvelle id post dans le tableaux de mon user

Et je fait de même pour mon post à l'inverse en ajoutant l'id de l'user a mon post

Ses la même fonction que la précédente sauf que au lieu de `addToSet` je pull mes ids

Post controler les commentaires

```
151 module.exports.commentPost = (req, res) => {  
152   if (!ObjectID.isValid(req.params.id))  
153     //  
154     return res.status(400).send("ID unknown : " + req.params.id);  
155   try {  
156     return PostModel.findByIdAndUpdate(  
157       req.params.id,  
158       {  
159         $push: {  
160           comments: {  
161             commenterId: req.body.commenterId,  
162             commenterPseudo: req.body.commenterPseudo,  
163             text: req.body.text,  
164             timestamp: new Date().getTime(),  
165           },  
166         },  
167       },  
168       (err, docs) => {  
169         if (!err) return res.send(docs);  
170         else return res.status(400).send(err);  
171       }  
172     );  
173   } catch (err) {  
174     return;  
175   }  
176 };  
177  
178 module.exports.editCommentPost = (req, res) => {  
179   if (!ObjectID.isValid(req.params.id))  
180     return res.status(400).send("ID unknown : " + req.params.id);  
181   try {  
182     return PostModel.findById(req.params.id, (err, docs) => {  
183       const theComment = docs.comments.find((comment) =>  
184         comment._id.equals(req.body.commentId)  
185       );  
186  
187       if (!theComment) return res.status(404).send("Comment not found");  
188       theComment.text = req.body.text;  
189  
190       return docs.save((err) => {  
191         if (!err) return res.status(200).send(docs);  
192         return res.status(500).send(err);  
193       });  
194     });  
195   } catch (err) {  
196     return res.status(400).send(err);  
197   }  
198 }  
199 };
```

Création de mon
commentaire avec
son model

Va chercher le post
dans la bdd

Va chercher le
commentaire par son
id dans la bdd

Si il ne trouve pas le
commentaire

Post controler

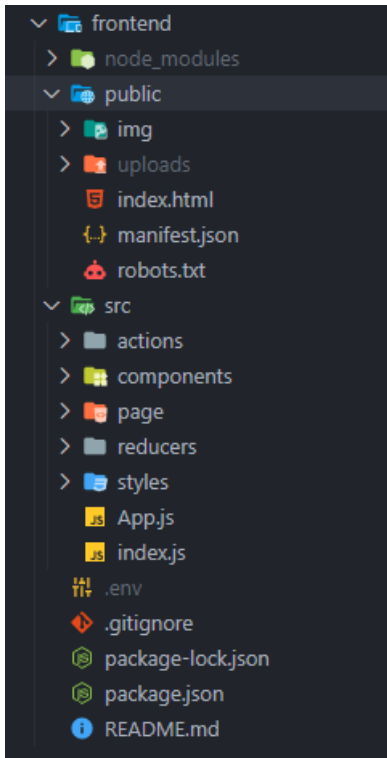
Suppression du
commentaire avec
la fonction pull

```
201 module.exports.deleteCommentPost = (req, res) => {
202   if (!ObjectID.isValid(req.params.id))
203     return res.status(400).send("ID unknown : " + req.params.id);
204
205   try {
206     return PostModel.findByIdAndUpdate(
207       req.params.id,
208       {
209         $pull: {
210           comments: {
211             _id: req.body.commentId,
212           },
213         },
214       },
215       { new: true },
216       (err, docs) => {
217         if (!err) return res.send(docs);
218         else return res.status(400).send(err);
219       }
220     );
221   } catch (err) {
222     return res.status(400).send(err);
223   }
224 }
```

front

- ▶ Donc pour ma partie front je vais utiliser react et redux ainsi que du sass pour la mise en page, je ne m'attarderais pas sur se dernier sauf si vous le souhaitais
- ▶ Pour la mise en place de react j'ai utiliser la commande `npx creat-react-app`

Architecture du code



public

Ici sera stocké tout les images de mon site et les uploads des utilisateurs
Il y a aussi la page index qui est ma page de référence pour tout mon site

src

C'est le dossier ou il va y avoir tout mon code

actions

Se sont mes action qui vont déclencher mes reducers

components

Ses ici que son coder les composant qui vont formée ma page

page

Definit l'ensemble des page de mon site ses ici que nous faisons l'assemblage de nos composant

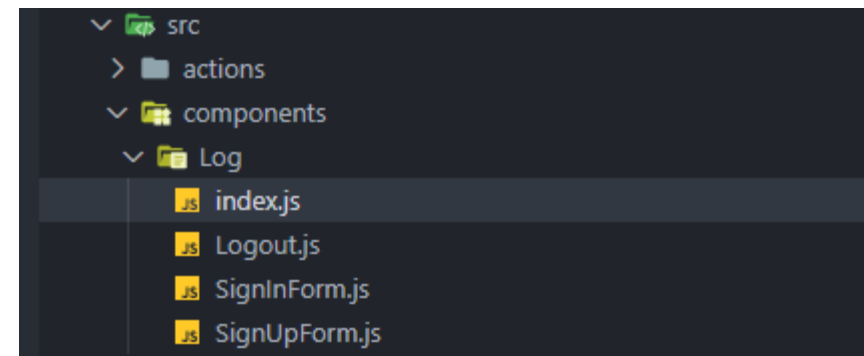
reducers

Gestion de mes reducers

Styles

Ses ici que sera définit toute la mise en page du site

Authentification



```
5 // eslint-disable-next-line
6 /*suivant la page de par la quelle on appelle le component on aura signup ou signin afficher*/
7 const Log = (props) => {
8   const [signUpModal, setSignUpModal] = useState(props.signup);
9   const [signInModal, setSignInModal] = useState(props.signin);
10
11   // eslint-disable-next-line
12   /*si on appuie sur s'inscrire change les etat si on appuie sur se connecter change les etat a l'inverse*/
13   const handleModals = (e) => {
14     if (e.target.id === "register") {
15       setSignInModal(false);
16       setSignUpModal(true);
17     } else if (e.target.id === "login") {
18       setSignUpModal(false);
19       setSignInModal(true);
20     }
21   };
22 }
```

Déclaration de mes variables

Fonction qui gère l'état de mes variables suivant de par qui elle a été appelée

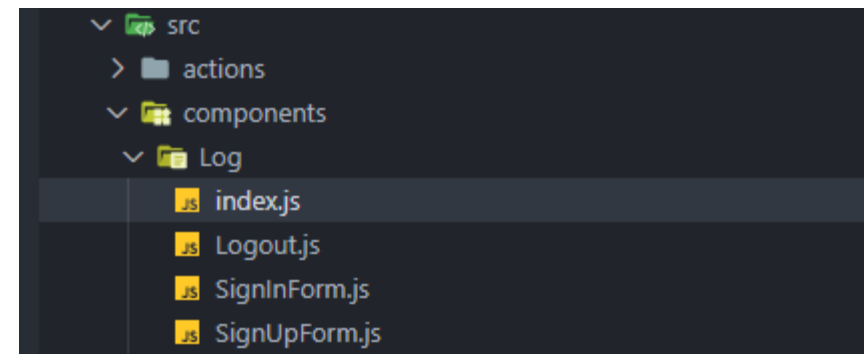
```
const [signUpModal, setSignUpModal] = useState(props.signup);
```

ma variable

Commande qui sert à modifier ma variable

Props useState sert à déclarer l'état initial de ma variable

Authentication



```
23 return (  
24   <div className="connection-form">  
25     <div className="form-container">  
26       <ul>  
27         <li  
28           onClick={handleModals}  
29           id="register"  
30           className={signInModal ? "active-btn" : null}  
31         >  
32           S'inscrire  
33         </li>  
34         <li  
35           onClick={handleModals}  
36           id="login"  
37           className={signInModal ? "active-btn" : null}  
38         >  
39           Se connecter  
40         </li>  
41       </ul>  
42       {signInModal && <SignUpForm />}  
43       /*si on click sur inscription alor SignUpForm*/  
44  
45       {signInModal && <SignInForm />}  
46       /*si on click sur se connecter alor SignInForm*/  
47     </div>  
48   </div>  
49 );  
50  
51 export default Log;
```

Dans le return nous allons retourner toute notre mise en page

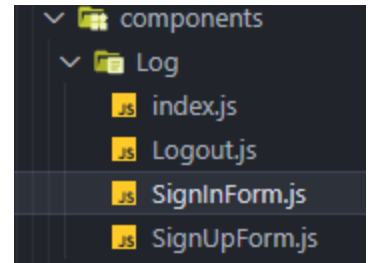
Au click nous allons déclencher la fonction handleModal qui sera traite l'infomation en fonction de l'id

La class de mon bouton changera en fonction de mes variables

Condition pour l'appelle de mes conposant

Par soucie de temps je vous passerais toute les parties return sauf celle que je juge pertinente nous pourrons cependant revenir ensemble sur ces dernière à la fin si vous avez la moindre question

Components/SignInForm



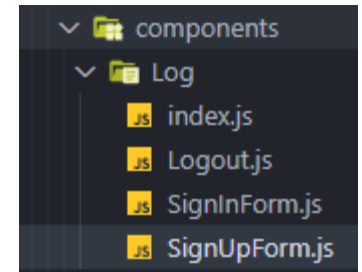
```
8  const handleLogin = (e) => {
9    e.preventDefault();
10   const emailError = document.querySelector(".email.error");
11   const passwordError = document.querySelector(".password.error");
12
13   axios({
14     method: "post",
15     url: `${process.env.REACT_APP_API_URL}api/user/login`,
16     withCredentials: true,
17     data: { email, password },
18   })
19   .then((res) => {
20     if (res.data.errors) {
21       emailError.innerHTML = res.data.errors.email;
22       passwordError.innerHTML = res.data.errors.password;
23     } else {
24       window.location = `/`;
25     }
26   })
27   .catch((err) => {
28     console.log(err);
29   });
30 }
```

Envoie une requêtes à la base de donnée avec axios (cette partie va être récurrente donc je vais l'expliquer une fois ici et je ne l'expliquerais pas pour les fois prochaine)

Traite les messages d'erreur

Si aucune erreur sur l'adresse mail et le mot de passe alors me redirige vers la page principal

Components / SignUpForm



```
23 passwordConfirmError.innerHTML = "";
24 termsError.innerHTML = "";
25
26 if (password !== controlPassword || !terms.checked) {
27   if (password !== controlPassword)
28     passwordConfirmError.innerHTML =
29       "Les mots de passe ne correspondent pas";
30
31   if (!terms.checked)
32     termsError.innerHTML =
33       "Veuillez valider les conditions générales";
34 } else {
35   await axios({
36     method: "post",
37     url: `${process.env.REACT_APP_API_URL}api/user/register`,
38     data: {
39       pseudo,
40       email,
41       password,
42     },
43   })
44     .then((res) => {
45       console.log(res);
46       if (res.data.errors) {
47         pseudoError.innerHTML = res.data.errors.pseudo;
48         emailError.innerHTML = res.data.errors.email;
49         passwordError.innerHTML = res.data.errors.password;
50       } else {
51         setFormSubmit(true);
52       }
53     })
54     .catch((err) => console.log(err));
55 }
56
```

Mes des strings vide dans mes const

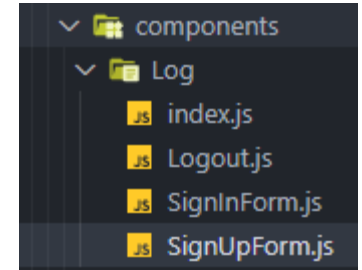
Vérifie que les deux mots de passe soit similaire

Validation des conditions général avec acceptation de cookies

Gestion des erreurs

Passage a true de la constante FormSubmit

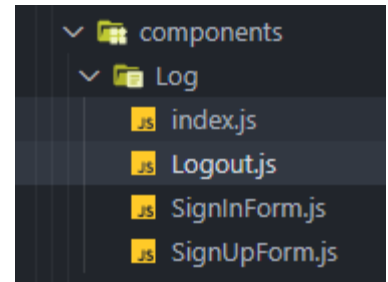
Components / SignUpForm



```
63 | <|
64 |   <SignInForm />
65 |   <h4 className="success">...
67 |   </h4>
68 | </>
```

Mise en place de fragment car en jsx pour un bon fonctionnement si deux parties sont au même niveau elle doivent être encapsuler

Components/log/logout

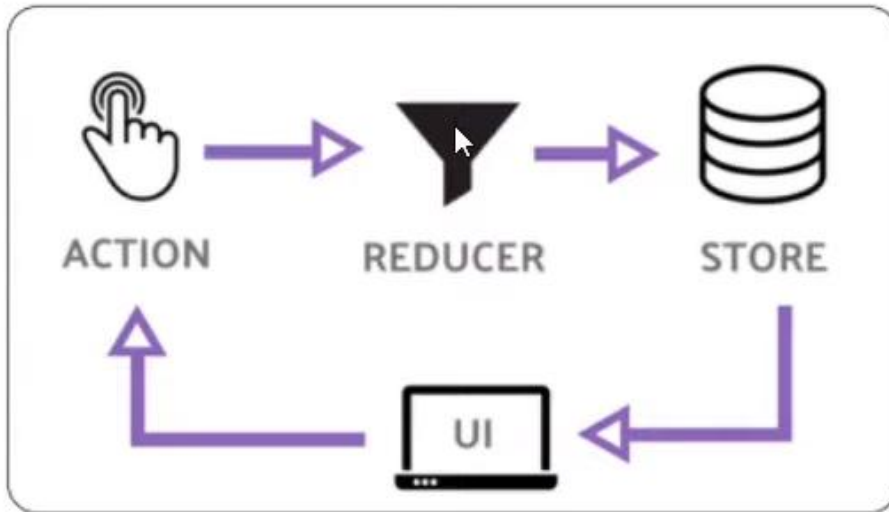


```
5  const Logout = () => {  
6    const removeCookie = (key) => {  
7      if (window !== "undefined") {  
8        cookie.remove(key, { expires: 1 });  
9      }  
10   };  
11  
12   const logout = async () => {  
13     await axios({  
14       method: "get",  
15       url: `${process.env.REACT_APP_API_URL}api/user/logout`,  
16       withCredentials: true,  
17     })  
18       .then(() => removeCookie("jwt"))  
19       .catch((err) => console.log(err));  
20  
21     window.location = "/";  
22   };  
23 }
```

Fonction de suppression du cookie

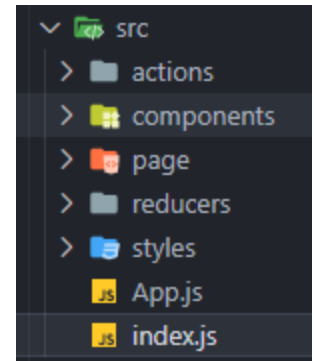
Redirection vers la page d'accueil

Redux



- ▶ Alors redux comment sa marche
- ▶ comme vous le voyais il y a 3 element 1 action 1 reducer 1 store
- ▶ il faut comprendre ses 3 element donc le store ses la ou on va stocker toute nos donner toute les donner stocker dedans sont accesible a n'importe quelle component dans le userinterface
- ▶ donc que peut on faire par exemple on se dit tien je ves follow un utilisateur donc ceci est une action sur la userinterface
- ▶ a partir de la l'action va dire a la base de donne telle user follow tel user
- ▶ et sa passe au reducer qui va dire la meme chose au store et force de redux ses que toute info change dans le store l'est immédiatement a l'ecrand
- ▶ donc par exemple le bouton ou il y avais ecrit suivre deviendra abonner a la personne
- ▶ le user aura un component de suivi des user il va changer instantanement alor que se sont des component qui sont diferent et qui n'interagisse pas entre eux
- ▶ mes leur point commun ses que il prenne tous leur data du store

Mise en place de redux



```
1  import React from "react";
2  import App from "../App";
3  import "../styles/style.css";
4  import { createRoot } from "react-dom/client";
5  import { Provider } from "react-redux";
6  import { applyMiddleware, createStore } from "redux";
7  import thunk from "redux-thunk";
8  import rootReducer from "../reducers";
9  import { getUsers } from "../actions/users.actions";
10
11  // dev tools
12  import { composeWithDevTools } from "redux-devtools-extension";
13
14  const store = createStore(
15    rootReducer,
16    composeWithDevTools(applyMiddleware(thunk))
17  );
18
19  store.dispatch(getUsers());
20
21  const container = document.getElementById("root");
22  const root = createRoot(container);
23
24  root.render(
25    <Provider store={store}>
26      <App tab="home" />
27    </Provider>
28  );
```

Mise en place d'un devtools pour redux a enlever lors de la mise en prod

Création de mon store

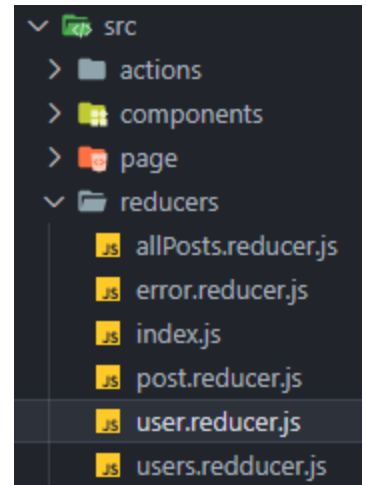
Thunk est une sorte de middleware qui permet de faire des requête asynchrone avec redux

Mise en place de notre provider qui sert a connecter notre store à react

```

1 import { GET_USER, UPDATE_BIO, UPLOAD_PICTURE } from "../actions/user.actions";
2
3 const initialState = {};
4
5 export default function userReducer(state = initialState, action) {
6   switch (action.type) {
7     case GET_USER:
8       return action.payload;
9     case UPLOAD_PICTURE:
10      return {
11        ...state, //retourne toute les donnees
12        picture: action.payload, //et modifie que la picture
13      };
14     case UPDATE_BIO:
15      return {
16        ...state,
17        bio: action.payload,
18      };
19     default:
20      return state;
21   }
22 }
23

```

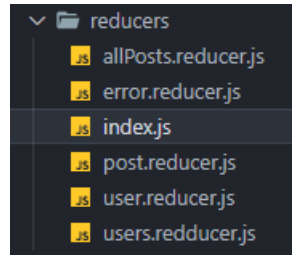


Tout nos reducers doivent avoir un initialState

Creation de ma fonction avec en parametre mon state et une action

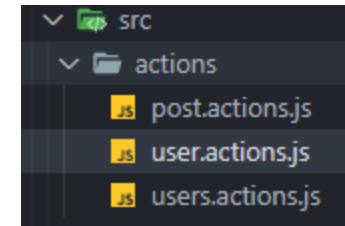
me retourne la data en fonction de l'action qui a été appelé

Quand nous modifions que une chose du store si nous ne voulons pas tout écraser il est important de rappeler toute les données



```
1 import { combineReducers } from "redux";
2 import userReducer from "../user.reducer";
3 import usersReducer from "../users.reducer";
4 import postReducer from "../post.reducer";
5 import errorReducer from "../error.reducer";
6 import allPostsReducer from "../allPosts.reducer";
7
8 export default combineReducers({
9   userReducer,
10  usersReducer,
11  postReducer,
12  errorReducer,
13  allPostsReducer,
14 });
```

- ▶ Le fichier index.js va être celui qui me permet d'organiser tout les reducers
- ▶ Import de tout mes reducer
- ▶ combine ensemble tout mes reducer



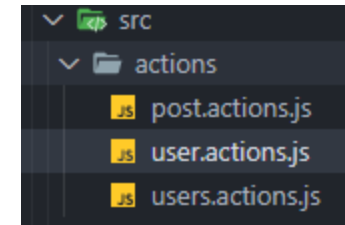
```
1 import axios from "axios";
2
3 export const GET_USER = "GET_USER";
4 export const UPLOAD_PICTURE = "UPLOAD_PICTURE";
5 export const UPDATE_BIO = "UPDATE_BIO";
6 export const GET_USER_ERRORS = "GET_USER_ERRORS";
7
8 export const getUser = (uid) => {
9   //dispatch ses se qui est envoyer au reducer
10  return (dispatch) => {
11    return axios
12      .get(`${process.env.REACT_APP_API_URL}api/user/${uid}`)
13      .then((res) => {
14        dispatch({ type: GET_USER, payload: res.data });
15      })
16      .catch((err) => console.log(err));
17  };
18 };
19
20 export const uploadPicture = (data, id) => {
21  return (dispatch) => {
22    return axios
23      .post(`${process.env.REACT_APP_API_URL}api/user/upload`, data)
24      .then((res) => {
25        return axios
26          .get(`${process.env.REACT_APP_API_URL}api/user/${id}`)
27          .then((res) => {
28            dispatch({
29              type: UPLOAD_PICTURE,
30              payload: res.data.picture,
31            });
32          });
33      })
34      .catch((err) => console.log(err));
35  };
36 };
```

Ici je défini 'la table des matières' de toute nos actions

Dispatch ses se qui part au reducer

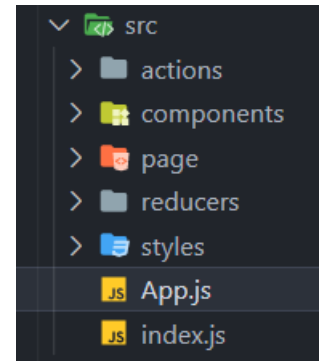
Envoie une requêtes à la base de donnée avec axios

Et récupère les info pour les faire passer à mon reducer (donc les GET_USER contiendra le res.data en info)



```
38 export const updateBio = (userId, bio) => {
39   return (dispatch) => {
40     return axios({
41       method: "put",
42       url: `${process.env.REACT_APP_API_URL}api/user/` + userId,
43       data: { bio },
44     })
45     .then((res) => {
46       dispatch({ type: UPDATE_BIO, payload: bio });
47     })
48     .catch((err) => console.log(err));
49   };
50 }
```

- Pour la partie updateBio aucun changement que se qui a déjà été fait au dessus
- ses a dire return du dispatch requette avec axios et envoie a mon reducer les données



```
12 const App = () => {  
13   const [uid, setUid] = useState(null);  
14   const dispatch = useDispatch();  
15   //a chaque fois que app est appelé controle du token  
16   useEffect(() => {  
17     const fetchToken = async () => {  
18       await axios({  
19         method: "get",  
20         url: `${process.env.REACT_APP_API_URL}jwtid`,  
21         withCredentials: true,  
22       })  
23       .then((res) => setUid(res.data)) //fait evoluer le uid  
24       .catch((err) => console.log("No token"));  
25     };  
26     fetchToken();  
27     if (uid) dispatch(getUser(uid));  
28   }, [uid, dispatch]); // [uid] à chaque fois que uid evolue tu rejoue App  
29  
30   return (  
31     <UidContext.Provider value={uid}>  
32       <BrowserRouter>  
33         <Navbar />  
34         <LeftNav />  
35         <Routes>  
36           <Route path="/" element={<Home />} />  
37           <Route path="/profil" element={<Profil />} />  
38           /*path="*" fonctionne si jamais l'url ne correspond à rien déclaré au dessus */  
39           <Route path="*" element={<Home />} />  
40         </Routes>  
41       </BrowserRouter>  
42     </UidContext.Provider>  
43   );  
44 };  
45  
46 export default App;
```

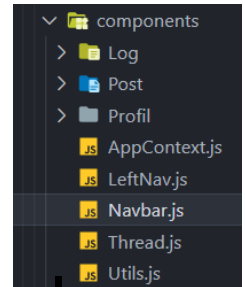
A chaque fois que App est appelé nous vérifions le token

appelle de mon getUser avec en param mon uid

Suivant le chemin taper en adresse sa va me renvoyer sur les component suivant

Apelle de mon component appcontext qui va me permettre de stocker mon user id au plus aux de mon application

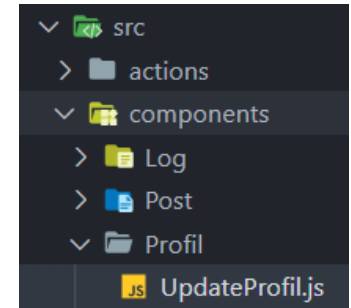
Components / Navbar.js



```
7  const Navbar = () => {  
8    const uid = useContext(UidContext);  
9    const userData = useSelector((state) => state.userReducer);  
10  
11    return (  
12      <nav>  
13        <div className="nav-container">  
14          <div className="logo">  
15            <NavLink to="/">  
16              <div className="logo">  
17                  
18                <h3>Groupomania</h3>  
19              </div>  
20            </NavLink>  
21          </div>  
22          {uid ? (  
23            <ul>  
24              <li className="welcome">  
25                <NavLink to="/profil">  
26                  <h5>Bienvenue {userData.pseudo}</h5>  
27                </NavLink>  
28              </li>  
29              <Logout />  
30            </ul>  
31          ) : (  
32            <ul>  
33              <li>  
34                <NavLink to="/profil">  
35                    
36                </NavLink>  
37              </li>  
38            </ul>  
39          )}  
40        </div>  
41      </nav>  
42    );  
43  };
```

La mise en place
du reducer va me
permettre de
pouvoir récupérer
ma valeur
dynamique comme
ici mon pseudo

UpdateProfil



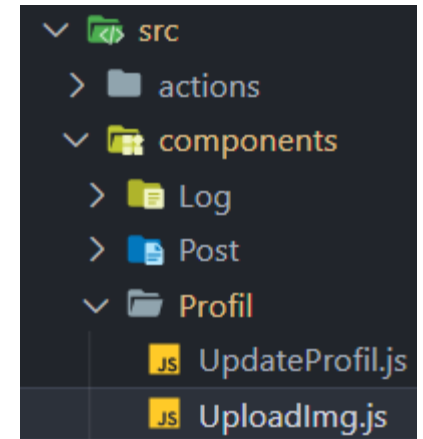
```
9  const UpdateProfil = () => {  
10    const [bio, setBio] = useState("");  
11    const [updateForm, setUpdateForm] = useState(false);  
12    const userData = useSelector((state) => state.userReducer);  
13    const dispatch = useDispatch();  
14  
15    const handleUpdate = () => {  
16      dispatch(updateBio(userData._id, bio));  
17      setUpdateForm(false);  
18    };  
19  
20    const deleteUser = async () => {  
21      await axios({  
22        method: "DELETE",  
23        url: `${process.env.REACT_APP_API_URL}api/user/${userData._id}`,  
24        withCredentials: true,  
25      })  
26      .then(async () => {  
27        const removeCookie = (key) => {  
28          if (window !== "undefined") {  
29            cookie.remove(key, { expires: 1 });  
30          }  
31        };  
32  
33        await axios({  
34          method: "get",  
35          url: `${process.env.REACT_APP_API_URL}api/user/logout`,  
36          withCredentials: true,  
37        })  
38        .then(() => removeCookie("jwt"))  
39        .catch((err) => console.log(err));  
40      })  
41      .then(() => (window.location = "/"))  
42      .catch((err) => {  
43        console.log(err);  
44      });  
45    };  
46  }  
47  };
```

Si l'on appui sur le bouton pour modifier la bio sa appelle la fonction handleUpdate qui fait appelle a l'action updateBio et l'envoi dans le mon state

lors de la suppression d'un user le déconnecte aussi en lui retirant le cookie et je le redirige vers la page d'accueil

UploadImg

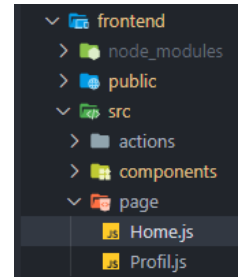
```
1 import React, { useState } from "react";
2 import { useDispatch, useSelector } from "react-redux";
3 import { uploadPicture } from "../../actions/user.actions";
4
5 const UploadImg = () => {
6   const [file, setFile] = useState();
7   const dispatch = useDispatch();
8   const userData = useSelector((state) => state.userReducer);
9
10  const handlePicture = (e) => {
11    const data = new FormData();
12    data.append("name", userData.pseudo);
13    data.append("userId", userData._id);
14    data.append("file", file);
15
16    dispatch(uploadPicture(data, userData._id));
17  };
18
19  return (
20    <form action="" onSubmit={handlePicture} className="upload-pic">
21      <label htmlFor="file">Changer d'image</label>
22      <input
23        type="file"
24        id="file"
25        name="file"
26        accept=".jpg, .jpeg, .png"
27        onChange={(e) => setFile(e.target.files[0])}
28      />
29      <br />
30      <input type="submit" value="Envoyer" />
31    </form>
32  );
33 };
34
35 export default UploadImg;
```



Lors de l'appuie sur mon form lance la fonction handlePicture

Qui va récupérer les données et les faire passer dans mon dispatch et appelle l'action uploadPicture

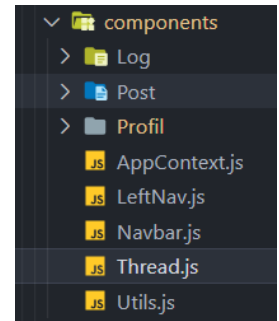
Fil d'actualité



```
1 import React, { useContext } from "react";
2 import { UidContext } from "../components/AppContext";
3 import NewPostForm from "../components/Post/NewPostForm";
4 import Thread from "../components/Thread";
5 import Log from "../components/Log";
6
7 const Home = () => {
8   const uid = useContext(UidContext);
9
10  return (
11    <div className="home">
12      <div className="main">
13        <div className="home-header">
14          {uid ? (
15            <NewPostForm />
16          ) : (
17            <Log signin={true} signup={false} />
18          )}
19        </div>
20        <Thread />
21      </div>
22    </div>
23  );
24 };
25
26 export default Home;
```

Si je suis
connecter il
m'affiche le
component
newpost form
sinon la page de
log

Fil d'actualité infinite scroll



```
7  const Thread = () => {
8    const [loadPost, setLoadPost] = useState(true);
9    const [count, setCount] = useState(5);
10   const dispatch = useDispatch();
11   const posts = useSelector((state) => state.postReducer);
12
13   const loadMore = () => {
14     //si on arivent en bas de la scroll barre alor on r'affiche les 5 prochains posts
15     if (
16       window.innerHeight + document.documentElement.scrollTop + 1 >
17       document.scrollingElement.scrollHeight
18     ) {
19       setLoadPost(true); //on affiche les 5 prochains posts
20     }
21   };
22
23   useEffect(() => {
24     if (loadPost) {
25       dispatch(getPosts(count)); //recupere les 5 prochains posts
26       setLoadPost(false); //passe a false pour ne pas charger de nouveaux posts
27       setCount(count + 5); //incrimente le nombre de posts a recuperer
28     }
29
30     window.addEventListener("scroll", loadMore);
31     return () => window.removeEventListener("scroll", loadMore);
32   }, [loadPost, dispatch, count]);
33
34   return (
35     <div className="thread-container">
36       <ul>
37         {!isEmpty(posts[0]) &&
38           posts.map((post) => {
39             return <Card post={post} key={post._id} />;
40           })}
41       </ul>
42     </div>
43   );
44 }
```

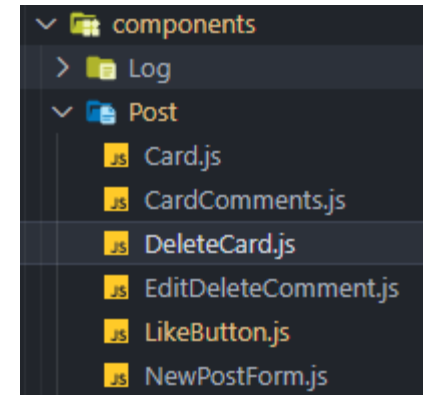
useSelector sert à récupérer les info du store

Gestion d'un infinite scroll

Si on arrive a 1px du bas la scroll bar alors on affiche les 5 message suivant

Gestion des droits d'admin

Vérification si on est l'auteur ou si on est un admin alors les bouton de suppression seront disponibles

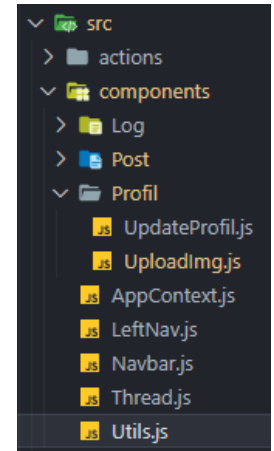


```
35     return (  
36       <div className="edit-comment">  
37         {(isAuthor || userData.isAdmin) && edit === false && (  
38           <span onClick={() => setEdit(!edit)}>  
39               
40           </span>  
41         )}  
42         {(isAuthor || userData.isAdmin) && edit && (  
43           <span onClick={() => setEdit(!edit)}>  
44               
45           </span>  
46         )}  
47       </div>  
48     )  
49   )  
50 }
```

Chose importante en jsx lors des condition les && on la priorité sur les || donc il faut mettre des parenthèse sur les condition ou si on ne veut pas avoir des erreurs

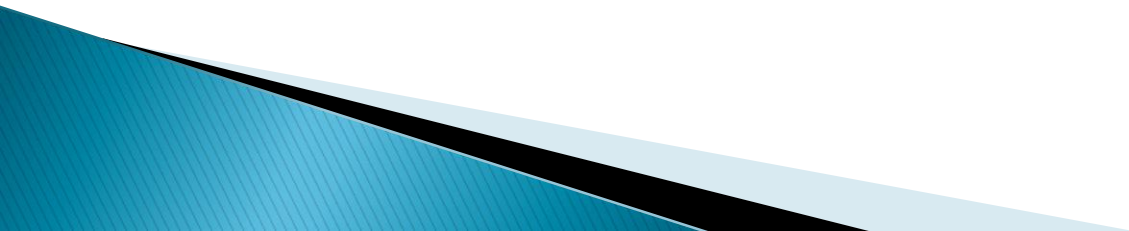
utils

```
1 > /**...
2
3 export const dateParser = (num) => {
4   //creation de mes option d'affichage de la date
5   let options = {
6     hour: "2-digit",
7     minute: "2-digit",
8     second: "2-digit",
9     weekday: "long",
10    year: "numeric",
11    month: "short",
12    day: "numeric",
13  };
14  //recupere la date en millisecondes depuis le 1er janvier 1970
15  let timestamp = Date.parse(num);
16  //convertie ma date en millisecondes au format de date fr
17
18  let date = new Date(timestamp).toLocaleDateString("fr-FR", options);
19
20  return date.toString();
21 };
22
23 > /**...
24
25 export const timestampParser = (num) => {
26   let options = {
27     hour: "2-digit",
28     minute: "2-digit",
29     second: "2-digit",
30     weekday: "long",
31     year: "numeric",
32     month: "short",
33     day: "numeric",
34   };
35
36   let date = new Date(num).toLocaleDateString("fr-FR", options);
37
38   return date.toString();
39 };
```



```
_id: ObjectId('62cd12f4a5aed685c2b7256b')
posterId: "62cd12daa5aed685c2b72562"
message: "qzzfzb"
picture: "./uploads/posts/62cd12daa5aed685c2b725621657606900193.png"
likers: Array
comments: Array
  0: Object
    commenterId: "62cd12daa5aed685c2b72562"
    commenterPseudo: "test"
    text: "wxfcg"
    timestamp: 1657612866201
    _id: ObjectId('62cd2a423abeb2c26b73ae1f')
    createdAt: 2022-07-12T06:21:40.195+00:00
    updatedAt: 2022-07-12T08:01:06.202+00:00
    __v: 0
```

annexe



MongoDB

- ▶ MongoDB permet de gérer les bases de données. Il se différencie du **SQL**. MongoDB est utilisé dans la MERN stack car les données se manipulent sous format JSON et il est vraiment très simple de transformer des données **JavaScript vers MongoDB** et inversement grâce à des librairies tel que mongoose par exemple.

Express

- ▶ Express est un langage de **modélisation des données** qui spécifie formellement les données. Il donne la capacité de développer une **application web complexe**. Express fait appel au système de middleware qui est dédié à la gestion **d'application web** de grande complexité. Express est dédié à la création de la REST API. La REST API correspond à un site web qui va récupérer des données via des requêtes HTTP. Ces requêtes sont exécutées avec le langage React.js.

React.js

- ▶ React.js est une **bibliothèque JavaScript** développée par Facebook depuis 2013. React.js développe la partie front-end du site web. Ce langage est réputé pour **accélérer les vitesses de chargement de sites internet**. Les animations, chargements et tout autre transition sont beaucoup plus rapides. Netflix utilise du React.js par exemple, c'est le meilleur choix technologique si vous souhaitez développer **une application web**, une application mobile ou encore un logiciel rapidement.



Node.js

- ▶ Node.js fonctionne **avec le JavaScript**. Ces deux langages sont fortement liés et sont très simples à faire interagir entre eux. Par conséquent, un **développeur maîtrisant le JavaScript** n'aura aucune difficulté à apprendre du Node.JS.
- ▶ En outre, Node.JS a la particularité d'être un langage extrêmement rapide, l'un des langages les plus rapides de tous les autres langages de la programmation informatique, car c'est également un langage asynchrone. Des entreprises de renom telles que Facebook utilisent Node.js.