

Groupomania

Groupomania

Rechercher...

John Doe
Développeur
Inscrit le 13/10/2022

Déconnexion Modifier

Ecrire un message...

Utilisateurs (3)

- Admin
- Jane Doe
- John Doe

10. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse a leo at leo cursus consectetur eu sed magna. Aenean feugiat tempor leo. Maecenas tristique sodales gravida. Aenean eget sapien dictum, porta dolor id, ullamcorper purus. Interdum et malesuada fames ac ante ipsum primis in faucibus.



Ecrire un commentaire... ▶

Groupomania

Rechercher...

John Doe
Développeur
Inscrit le 13/10/2022

Déconnexion Modifier

Ecrire un message...

Admin

10. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse a leo at leo cursus consectetur eu sed magna. Aenean feugiat tempor leo. Maecenas tristique sodales gravida. Aenean eget sapien dictum, porta dolor id, ullamcorper purus. Interdum et malesuada fames ac ante ipsum primis in faucibus.



1 like ▶

Ecrire un commentaire... ▶

Groupomania

Rechercher...

John Doe
Développeur
Inscrit le 13/10/2022

Déconnexion Modifier

Ecrire un message...

Admin

10. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse a leo at leo cursus consectetur eu sed magna. Aenean feugiat tempor leo. Maecenas tristique sodales gravida. Aenean eget sapien dictum, porta dolor id, ullamcorper purus. Interdum et malesuada fames ac ante ipsum primis in faucibus.



1 like ▶

Ecrire un commentaire... ▶

Contexte

- Construire un réseau social interne pour une entreprise
- Réaliser : Backend, Frontend, Base de données
 - => Framework frontend : React
 - => Database : mongoDB

Attentes

- Design : responsive, logo, police “Lato”, reste libre ;
- Inscription : données sécurisées dans la database ;
- Connexion : persistante jusqu’à la déconnexion ;
- Déconnexion : retour à l’écran de connexion ;
- Postes : créer/modifier/supprimer un post (texte/image) ;
- Likes : possibilité de liker les postes ;
- Admin : peut modifier/supprimer tous les postes.

Présentation

- Démonstration des fonctionnalités
- Présentation du code
- Bilan & Axes d’amélioration

The screenshot shows a dark-themed user interface for a social media platform named "Groupomania". At the top right, there is a user icon and a search bar labeled "Rechercher...". The main area features a profile card for "John Doe", a developer, who joined on 13/10/2022. Below the profile is a message input field with placeholder text "Ecrire un message...". To the right, a sidebar titled "Utilisateurs (3)" lists three users: "Admin", "Jane Doe", and "John Doe". The main content area displays two posts. The first post is by "Admin" with a timestamp of 14/10/2022 à 23:59. It contains a block of placeholder Latin text and a vibrant, colorful photograph of a city street at night with neon signs. The second post is by "John Doe" with a timestamp of 14/10/2022 à 23:59. It also contains placeholder Latin text and a photograph of a city street at night, similar to the first one but with different lighting and signage.

1. Démonstration

2. Code : intro

Database - MongoDB

MongoDB Compass

Collections

Create collection View

posts

Storage size: 20.48 kB
Documents: 10
Avg. document size: 496.00 B
Indexes: 1
Total index size: 36.86 kB

users

Storage size: 20.48 kB
Documents: 3
Avg. document size: 336.00 B
Indexes: 2
Total index size: 73.73 kB

Exemple Post

```
_id: ObjectId('63487d4812a70e0c41e53d94')
userId: "63487bfe12a70e0c41e53d57"
desc: "1. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse..."
image: "16657808117717.jpg"
likes: Array
  0: "63487dd012a70e0c41e53da2"
  1: "63487bfe12a70e0c41e53d57"
  2: "63487f0312a70e0c41e53db4"
createdAt: 2022-10-13T21:04:08.214+00:00
updatedAt: 2022-10-15T00:02:21.910+00:00
__v: 0
```

Exemple User

```
_id: ObjectId('63487f0312a70e0c41e53db4')
admin: false
email: "e3f121a822aaada57ae5306e6e7ddc1e42f12f474ec5f079770664e8dbf8c3057"
password: "$2a$10$nnmW9T4C3/6iQ18m7jOuwu9kW3Kkxe9.h0eQjzC3KD70fc9b0n1rG"
firstname: "John"
lastname: "Doe"
profilePicture: "1665929427224user(2).jpg"
about: "Developpeur"
createdAt: 2022-10-13T21:11:31.528+00:00
updatedAt: 2022-10-16T14:10:27.877+00:00
__v: 0
```

Backend - Rest Api

Structure

api	
controllers	
auth_controller.js	Controllers
post_controller.js	
user_controller.js	
middleware	
auth_mw.js	Middleware
mail-check_mw.js	
pass-check_mw.js	
models	
post_model.js	Models
user_model.js	
node_modules	
public	
routes	
auth_router.js	Routes
post_router.js	
upload_router.js	
user_router.js	
.env	.env
index.js	
package.json	
yarn.lock	

package.json

```
"dependencies": {  
    "bcryptjs": "^2.4.3",      Hashage password  
    "cors": "^2.8.5",         HTTP Headers  
    "crypto-js": "^4.1.1",     Cryptage Email  
    "dotenv": "^16.0.3",      Données sensibles  
    "express": "^4.18.2",     Framework  
    "helmet": "^6.0.0",       Pack Sécurité (XSS...)  
    "jsonwebtoken": "^8.5.1",   Authentification  
    "mongoose": "^6.6.5",     Librairie MongoDB  
    "mongoose-unique-validator": "^3.1.0",  
    "multer": "^1.4.5-lts.1", Upload images  
    "nodemon": "^2.0.20",     auto restart server  
    "password-validator": "^5.3.0",  
    "path": "^0.12.7",        Fichiers locaux  
    "validator": "^13.7.0"     Contrôle email  
}
```

index.js

```
import dotenv from "dotenv";  
import express from "express";  
import mongoose from "mongoose";  
import helmet from "helmet";  
import cors from "cors";  
// Routes  
import auth_router from "./routes/auth_router.js";  
import user_router from "./routes/user_router.js";  
import post_router from "./routes/post_router.js";  
import upload_router from "./routes/upload_router.js";  
  
//? App  
dotenv.config();  
const app = express();  
const PORT = process.env.PORT || 5000;  
  
//? Middleware  
app.use(helmet({ crossOriginResourcePolicy: { policy: "same-site" } }));  
app.use(cors());  
app.use(express.json());  
// Local images  
app.use(express.static("public"));  
app.use("/images", express.static("images"));  
// Routes  
app.use("/api/auth", auth_router);  
app.use("/api/users", user_router);  
app.use("/api/posts", post_router);  
app.use("/api/upload", upload_router);  
  
//? Start  
const connectDB = (uri) => {  
  return mongoose.connect(uri);  
};  
const start = async () => {  
  try {  
    await connectDB(process.env.MONGO_URI);  
    app.listen(PORT, () =>  
      console.log(`Server is listening on port ${PORT}`)  
    );  
  } catch (error) {  
    console.log({ message: error.message });  
  }  
};  
start();
```

import Packages

import Routers

Middlewares

Appel Routes

Database

Lancement

Backend - Thunder Client

THUNDER CLIENT ⚡ ...

New Request

Activity Collections Env

Collection Requests

- DEL** Delete User
1 min ago
- Post Requests**
 - GET** Test route posts
1 min ago
 - POST** Create Post
1 min ago
 - PUT** Update Post
1 min ago
 - DEL** Delete Post
1 min ago
 - PUT** Like / Dislike Post
1 min ago
 - GET** Get Post
1 min ago
 - GET** Get Posts
1 min ago
 - GET** Get Timeline Posts

TC Register User X

Méthode

POST localhost:5000/api/auth/register Route Send

Query Headers 2 Auth Body 1 Tests Pre Run New

Json Content Format

```
1 {
  2   "email": "test@mail.com",
  3   "password": "testtest1",
  4   "firstname": "test",
  5   "lastname": "test"
  6 }
```

Requête

Status: 200 OK Size: 642 Bytes Time: 241 ms

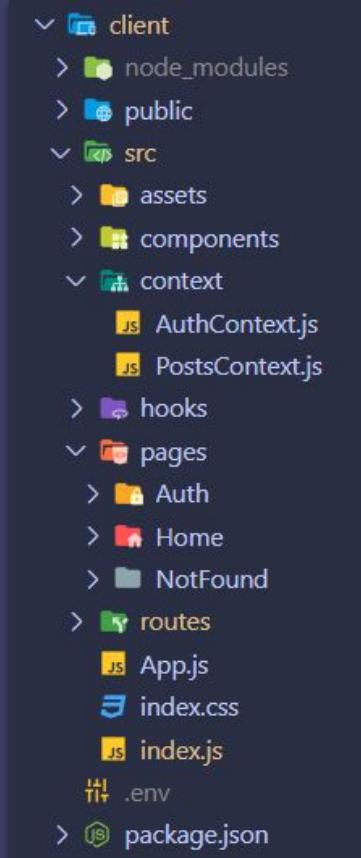
Response Headers 19 Cookies Results Docs { } =

```
1 {
  2   "user": {
  3     "admin": false,
  4     "email": "8da5e71c07ab23b3f0ac9a08f26668f6903235b70793c630847d2eb9bb32a2e1",
  5     "password": "$2a$10$aWioyhWJ294vWTCWyjBvs.n.n200f5VHqSB.DVuRrhNItfZFJgCis",
  6     "firstname": "test",
  7     "lastname": "test",
  8     "profilePicture": "",
  9     "about": "",
  10    "_id": "634c8fcfb1d9041715d6e02a",
  11    "createdAt": "2022-10-16T23:12:15.788Z",
  12    "updatedAt": "2022-10-16T23:12:15.788Z",
  13    "__v": 0
  14  },
  15  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6IjhkYTlNzFjMDdhYjIzYjNmMGFjOWEwOGYyNjY2OGY2OTAzMjM1YjcwNzkzYzYzMdg0N2QyZWl5YmIzMmEyZTEiLCJpZCI6IjYzNGM4ZmNmYjFkOTA0MTcxNWQ2ZTAyYSIiSmIhdCI6MTY2NTk2MTkzNSwiZXhwIjoxNjY2MDQ4MzM1fQ.MQxJ2NAhChIa25pJBuUKxdd98_0AsShNEQpN9fcGxA"
  16 }
```

Réponse

Frontend - React : Structure

```
{  
  "proxy": "http://localhost:5000",  
  "name": "starter",  
  "version": "0.1.0",  
  "private": true,  
  "dependencies": {  
    latest  
    "@iconscout/react-unicons": "^1.1.6",  
    latest  
    "@mantine/core": "^5.5.5",      Librairie modals  
    latest  
    "@mantine/hooks": "^5.5.5",  
    latest  
    "@testing-library/jest-dom": "^5.16.5",  
    latest  
    "@testing-library/react": "^13.4.0",  
    latest  
    "@testing-library/user-event": "^14.4.3",  
    latest  
    "axios": "^1.1.3",            Requêtes http  
    latest  
    "date-fns": "^2.29.3",        Affichage dates  
    latest  
    "react": "^18.2.0",          Framework React +  
    latest  
    "react-dom": "^18.2.0",       Router  
    latest  
    "react-router-dom": "6.4.2",  
    latest  
    "react-router-hash-link": "^2.4.3",  
    latest  
    "react-scripts": "5.0.1",  
    latest  
    "web-vitals": "^3.0.3"       Analyser performances  
  },  
}
```



Frontend - Accessibilité : WCAG

WAVE
web accessibility evaluation tool

powered by
[WebAIM](#)

Styles: OFF ON

Summary

Errors: 0 **Contrast Errors:** 0 **Alerts:** 0 **Features:** 61 **Structural Elements:** 35 **ARIA:** 33

[View details](#)

Congratulations! No errors were detected! Manual testing is still necessary to ensure compliance and optimal accessibility.

Groupomania

The screenshot shows the Groupomania website with an overlaid accessibility audit interface from WAVE. The audit highlights several issues across the page, including:

- Errors:** 0
- Contrast Errors:** 0
- Alerts:** 0
- Features:** 61
- Structural Elements:** 35
- ARIA:** 33

Key highlighted elements include:

- A logo icon with the text "*logo groupomania*".
- An input field labeled "Rechercher..." with an ARIA label "Recherche".
- A search icon with an ARIA label "lancer recherche".
- A button labeled "ajouter une image" with an ARIA label "ajouter une image".
- A user profile icon with an ARIA label "aller à votre page de profil".
- A "Maître des lieux" section containing an "Inscrit le" date "13/10/2022" and a "join date" button.
- A "Déconnexion" button.
- A "Modifier" button.
- A "photo utilisateur" image with an ARIA label "photo utilisateur".
- A "options" icon with an ARIA label "options".
- Section headers: h2 "utilisateur", h3 "Admin", h3 "Jane Doe".
- Text blocks: "10. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse a leo at leo cursus consectetur eu sed magna. Aenean feugiat tempor leo. Maecenas dictum sodales gravida. Aenean eget sapien dictum, porta de corpore purus. Interdum et" and a code block "</> Code".

Frontend - React : Router (1)

src/public/index.html

```
<!DOCTYPE html>
<html lang="fr">

<head>
  <meta charset="utf-8" />
  <link rel="icon" href="./favicon.png" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <meta name="theme-color" content="#000000" />
  <meta name="description" content="Web site created using create-react-app" />
  <title>Groupomania</title>
</head>

<body>
  <div id="root"></div>
</body>
```

Root ->

src/index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import { BrowserRouter } from "react-router-dom";
// Context
import { PostsContextProvider } from "./context/PostsContext"; Context
import { AuthContextProvider } from "./context/AuthContext"; Context
// Utilities
import { MantineProvider } from "@mantine/core";
import App from "./App";
import "./index.css";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <MantineProvider theme={{ colorScheme: "dark" }} withGlobalStyles>
      <AuthContextProvider> Context
        <PostsContextProvider> Router
          <BrowserRouter>
            <App /> Application
          </BrowserRouter>
        </PostsContextProvider> Router
      </AuthContextProvider> Context
    </MantineProvider>
  </React.StrictMode>,
);
```

Frontend - React : Router (2)

src/app.js

```
import React from "react";
import { useAuthContext } from "./hooks/useAuthContext"; import { user context }
import Router from "./routes";
import Auth from "./pages/Auth/Auth";

function App() {
  const { user } = useAuthContext();

  return (
    <div className="app">
      <div className="container">{user == null ? <Auth /> : <Router />}</div>
    </div>
  );
}

export default App;
```

Condition d'affichage

src/routes/index.js

```
import { Routes, Route } from "react-router-dom";
// Components
import AppBar from "../components/AppBar/AppBar";
// Pages
import Home from "../pages/Home/Home";
import NotFound from "../pages/NotFound/NotFound";
```

```
const Router = () => {
```

```
  return (
```

```
<>
```

```
<AppBar />
```

```
<Routes>
```

```
  <Route path="/" element={<Home />} />
  <Route path="*" element={<NotFound />} />
</Routes>
```

```
</>
```

```
)>;
```

```
export default Router;
```

Barre de navigation

Accueil

404

Frontend - React : Composants

✓	components	
✓	AppBar	1
appbar.css		
AppBar.jsx		
>	Auth	
>	Comments	2
>	Online	3
>	Post	4
>	PostCreate	6
>	PostOptionsModal	7
>	PostUpdateModal	8
>	ProfileCard	9
>	ProfileDetailsModal	10
>	ProfilePictureModal	11

Groupomania

1 : AppBar

2 : PostCreate

3 : Online

4 : Post(s)

5 : PostCreate

Utilisateurs (3)

- Admin
- Jane Doe
- John Doe

Ecrire un message...

5 : PostCreate

9

Admin

Maitre des lieux

Inscrit le 13/10/2022

Déconnexion

Modifier 10

4 : Post

Admin

14/10/2022 à 23:59

10. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse

consectetur eu sed magna. Aenean feugiat tempor leo. Maecenas tristique sodales gravida.

Aenean eget sapien dictum, porta dolor id, ullamcorper purus. Interdum et malesuada fames

ac ante ipsum primis in faucibus.

Supprimer

7 → 8

Modifie

Ecrire un commentaire...

1

2

The screenshot displays the Groupomania application's user interface. At the top, there is a navigation bar with a logo, a search bar, and user profile icons. Below the navigation bar, the main content area is divided into several sections. One section shows a user profile card for 'Admin' with a photo, a camera icon, a back arrow, and a '11' notification. Another section shows a post creation form with a message input field and two buttons. To the right, there is a sidebar titled 'Utilisateurs (3)' listing three users: Admin, Jane Doe, and John Doe. The bottom half of the screen features a large image of a city street at night with neon signs, overlaid with a post card. The post card includes a user profile for 'Admin', a timestamp, a text excerpt, and interaction buttons for 'Supprimer' and 'Modifier'. Below the post card, there is a comment section with a reply button and a comment input field. The entire interface is styled with a dark theme and uses blue outlines to highlight specific components.

2. Code : Authentification

Inscription - Backend (1)

Model : User

```
import mongoose from "mongoose";
const UserSchema = new mongoose.Schema(Mongoose
{
    admin: {
        type: Boolean,
        default: false,
    },
    email: { mail
        type: String,
        required: [true, "Please provide email"],
        unique: true,
        lowercase: true,
    },
    password: { password
        type: String,
        required: [true, "Please provide password"],
        minlength: 8,
    },
    firstname: { prenom
        type: String,
        required: [true, "Please provide firstname"],
        minlength: 2,
        maxlength: 20,
    },
    lastname: { nom
        type: String,
        required: [true, "Please provide lastname"],
        minlength: 2,
        maxlength: 20,
    },
    profilePicture: { image profil
        type: String,
        default: "",
    },
    about: { description
        type: String,
        maxlength: 20,
        default: "",
    },
},
{ timestamps: true
});
const UserModel = mongoose.model("User", UserSchema);
export default UserModel;
```

```
import User from "../models/user_model.js";
import bcrypt from "bcryptjs";
import jwt from "jsonwebtoken";
import cjs from "crypto-js";
Imports : Model, hash mot de passe, crypt email, token authentification

//? Register
export const register = async (req, res) => {
    // email
    const cryptedMail = cjs.HmacSHA256(req.body.email, process.env.CJS_KEY);
    req.body.email = cryptedMail;
    // password
    const salt = await bcrypt.genSalt(10);
    const hashedPass = await bcrypt.hash(req.body.password, salt);
    req.body.password = hashedPass;
    Cypter email
    Hasher password

    try {
        let user = await User.findOne({ email: cryptedMail.toString() });
        if (user) {
            return res.status(400).json({ message: "Email already in use" });
        }
        user = await User.create({ ...req.body });
        Créer utilisateur

        const token = jwt.sign(
            {
                email: user.email,
                id: user._id,
            },
            process.env.JWT_KEY,
            { expiresIn: process.env.JWT_TIME }
        );
        Crée token authentification (connexion)
        res.status(200).json({ user, token });
    } catch (error) {
        res.status(500).json({ message: error.message });
    }
}
```

Controller
User :
register

Inscription - Backend (2)

Middleware : valider password

```
import passwordValidator from "password-validator"

const passwordSchema = new passwordValidator();

passwordSchema
    .is().min(8)                                // Minimum length 8
    .is().max(32)                                // Maximum length 100
    .has().uppercase()                           // Must have uppercase letters
    .has().lowercase()                           // Must have lowercase letters
    .has().digits(1)                             // Must have at least 1 digits
    .has().not().spaces()                        // Should not have spaces
    .is().not().oneOf(['Passw0rd', 'Password123', '123456']); // Blacklist these values

const validPassword = (req, res, next) => {
    if (passwordSchema.validate(req.body.password)) { Méthode
        next();
    } else {
        return res
            .status(403)
            .json({
                error: `Le mot de passe n'est pas assez fort`,
            });
    }
};
```

Paramètres

// Minimum length 8
// Maximum length 100
// Must have uppercase letters
// Must have lowercase letters
// Must have at least 1 digits
// Should not have spaces

Middleware : valider email

```
import validator from "validator"

const validateEmail = (req, res, next) => {
    const { email } = req.body;
    if (validator.isEmail(email)) { Méthode
        next();
    } else {
        return res
            .status(403)
            .json({ error: `${email} n'est pas un email valide` });
    }
};
```

Router : Authentification

```
import express from "express";                                         Controller
import { login, register } from "../controllers/auth_controller.js";
import mail from "../middleware/mail-check_mw.js";   Middleware
import pass from "../middleware/pass-check_mw.js";  Middleware

const router = express.Router();
```

/// Routes

```
router.post("/register", mail, pass, register);
router.post("/login", login);

export default router;
```

Middleware

Database : User

```
_id: ObjectId('63487f0312a70e0c41e53db4')
admin: false
email: "e3f121a822aada57ae5306e6e7ddc1e42f12f474ec5f079770664e8dbf8c3057"
password: "$2a$10$nnmW9T4C3/6iQ18m7j0uwu9kW3Kkxe9.h0eQjzC3KD70fc9b0n1rG"
firstname: "John"
lastname: "Doe"
profilePicture: "1665929427224user(2).jpg"
about: "Developpeur"
createdAt: 2022-10-13T21:11:31.528+00:00
updatedAt: 2022-10-16T14:10:27.877+00:00
__v: 0
```

Inscription - Frontend

Composant : Register

The screenshot shows a registration form on a dark-themed application. The fields are:

- Prénom: James state
- Nom: Bond state
- Email: jamesbond state
Validation message: Please include an '@' in the email address. 'jamesbond' is missing an '@'.
- Mix de 8 chiffres & lettres minimum state
- Confirmer mot de passe state
- S'inscrire handleSubmit

At the bottom: Se connecter, Changement de composant

Formulaire d'inscription

```
const Register = ({ setIsRegistering }) => {
  const [firstname, setFirstname] = useState("");
  const [lastname, setLastname] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [passwordConfirm, setPasswordConfirm] = useState("");

  const { register, error, isLoading } = useRegister();

  const handleSubmit = async (e) => {
    e.preventDefault();
    await register(firstname, lastname, email, password);
  };  
appel du Hooks "register"

  return (
    <form onSubmit={handleSubmit} className="register">
      <img src={logo} alt="logo" className="auth-logo" />

      <div className="firstname.lastname-container">
        <div className="name-wrapper">
          <label htmlFor="firstname">Prénom</label>
          <input
            required
            type="text"
            placeholder="James"
            value={firstname} useState
            onChange={(e) => setFirstname(e.target.value)}
            pattern="^(?![\s.]+$)[A-zÀ-Ù\s\-\-]{1,25}$"
            contrôle
          />
        </div>
      </div>
    </form>
  );
}
```

Inscription - Frontend (2)

Hook : useRegister / inscription

```
import { useState } from "react";
import { useAuthContext } from "./useAuthContext";      useState + Context

export const useRegister = () => {
  const [error, setError] = useState(null);
  const [isLoading, setIsLoading] = useState(null);
  const { dispatch } = useAuthContext();                  Données récupérés

  const register = async (firstname, lastname, email, password) => {
    setIsLoading(true);
    setError(null);

    const response = await fetch('/api/auth/register', { req POST api
      method: "POST",
      headers: { "Content-type": "application/json" },
      body: JSON.stringify({firstname, lastname, email, password}),
    });
    const json = await response.json();
    if (!response.ok) {
      setIsLoading(false);
      setError(json.error);
    }
    if (response.ok) {
      // save the user to local storage   Sauvegarde user dans localStorage
      localStorage.setItem("user", JSON.stringify(json));          localStorage
      // update auth context
      dispatch({type: 'LOGIN', payload: json})      User context : dispatch
      setIsLoading(false)
    }
  };
  return {register, isLoading, error}
};
```

Connexion

Context : userContext / utilisateur

```
import { createContext, useReducer, useEffect } from "react";
import Auth from "../pages/Auth/Auth";

export const AuthContext = createContext();

export const authReducer = (state, action) => {
  switch (action.type) {
    case "LOGIN":
      return { user: action.payload, };           Connexion
    case "LOGOUT":
      return { user: "null" } Déconnexion
    case "UPDATE":
      return { user: action.payload }; Modifier
    default:
      return state;
  }
};

// Persistence de la connexion
export const AuthContextProvider = ({ children }) => {
  const [state, dispatch] = useReducer(authReducer, { user: null });

  useEffect(() => {
    const user = JSON.parse(localStorage.getItem("user"));           localStorage

    if (user) {
      dispatch({ type: "LOGIN", payload: user });                   Connexion
    }
  }, []);
  return (
    <AuthContext.Provider value={{ ...state, dispatch }}>
      {children}
    </AuthContext.Provider>  Appliquer dans le root de src/index.js
  );
};
```

Connexion - Backend

User controller : Login

```
//? Login
export const login = async (req, res) => {
  const { email, password } = req.body;

  if (!email || !password) {
    return res.status(400).json({ message: "All fields must be filled" });
  } else {
    const cryptedMail = cjs.HmacSHA256(req.body.email, process.env.CJS_KEY);
    try {
      const user = await User.findOne({ email: cryptedMail.toString() });
      Correspondance email database ?
    });

    if (user) {
      Correspondance password ?
      const validpass = await bcrypt.compare(password, user.password);

      if (!validpass) {
        res.status(400).json("Incorrect password");
      } else {
        const token = jwt.sign(
          {
            mail: user.email,
            id: user._id,
          },
          process.env.JWT_KEY,
          { expiresIn: process.env.JWT_TIME }
        );
        res.status(200).json({ user, token });
        Récupérer user + token
      }
    } else {
      res.status(404).json("Incorrect email");
    }
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};
```

Requête frontend

User Router

```
import express from "express";
import { login, register } from "../controllers/auth_controller.js";
import mail from "../middleware/mail-check_mw.js";
import pass from "../middleware/pass-check_mw.js";

const router = express.Router();

//? Routes
router.post("/register", mail, pass, register);
router.post("/login", login); Connexion

export default router;
```

Connexion - Frontend

Composant : Login



Formulaire de connexion

```

const Login = ({ setIsRegistering }) => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const { login, error, isLoading } = useLogin();

  const handleSubmit = async (e) => {
    e.preventDefault();
    await login(email, password);    STATES
  };

  return (
    <form onSubmit={handleSubmit} className="login">
      <img src={logo} alt="logo" className="auth-logo" />

      <label htmlFor="login-email">Email</label>
      <input
        type="email"
        placeholder=""
        value={email}
        onChange={(e) => setEmail(e.target.value)}
        required
        requisi
      />    Appel Hook Login

      <label htmlFor="login-password">Mot de passe</label>
      <input
        required
        type="password"
        placeholder=""
        value={password}
        onChange={(e) => setPassword(e.target.value)}    Récupérer email
      />    Récupérer pass

      <button disabled={isLoading} type="submit">
        Se connecter
      </button>
      {error && <div className="error">{error}</div>}
      <p>
        <span onClick={() => setIsRegistering(true)}>S'inscrire</span>
      </p>
    </form>
);

```

Hook : useLogin / connexion

```

import { useState } from "react";
import { useAuthContext } from "./useAuthContext";

export const useLogin = () => {
  const [error, setError] = useState(null);
  const [isLoading, setIsLoading] = useState(null);
  const { dispatch } = useAuthContext();

  const login = async (email, password) => {
    setIsLoading(true);
    setError(null);
    fetch POST : envoi de la requête
    const response = await fetch('/api/auth/login', {
      method: "POST",
      headers: { "Content-type": "application/json" },
      body: JSON.stringify({ email, password })
    });
    const json = await response.json();
    if (!response.ok) {
      setIsLoading(false);
      setError(json.error);
    }
    if (response.ok) {
      // save the user to local storage
      localStorage.setItem("user", JSON.stringify(json));
      // update auth context
      dispatch({type: 'LOGIN', payload: json})
      setIsLoading(false)
    }
  };
  return {login, isLoading, error}
};

Sauvegarde LocalStorage
Dispatch -> UserContext

```

Local Storage

Key	Value
user	{"user": {"_id": "63487f0312a70e0c41e53db4", "admin": false, "em..."} ▼ {user: {_id: "63487f0312a70e0c41e53db4", admin: false,...}, token: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtYWlsIjo...
► user	{user: {_id: "63487f0312a70e0c41e53db4", admin: false,...}}

Déconnexion

Composant : Profile Card



Actions ...

```
<button onClick={handleLogout}>Déconnexion</button>
```

```
//? Logout
const { logout } = useLogout();
const handleLogout = () => {
  logout();
};
```

Hook : Logout / déconnexion

```
import { useAuthContext } from "./useAuthContext";

export const useLogout = () => {
  const { dispatch } = useAuthContext();

  const logout = () => {
    // remove user from storage
    localStorage.removeItem("user");
    // dispatch logout action
    dispatch({ type: "LOGOUT" });
  };

  return { logout };
};
```

Local storage

Context user

Console : localStorage vidé

```
User : ▼ {user: null} ⓘ
      user: null
▶ [[Prototype]]: Object
```

Retour au login

 Groupomania

Email

Mot de passe

Se connecter

S'inscrire

2. Code : Posts

Posts - Backend : Routeur + Sécurité

Router : Posts + Upload (images)

```
app.use("/api/posts", post_router);
app.use("/api/upload", upload_router);

import express from "express";
import {
  createPost,
  updatePost,
  deletePost,
  likePost,
  getPost,
  getPosts,
  getUserPosts,
} from "../controllers/post_controller.js";
import auth from "../middleware/auth_mw.js";

const router = express.Router();

//? Routes
router.use(auth); middleware authentication
router.post("/", createPost);
router.put("/:id", updatePost);
router.delete("/:id", deletePost);
router.put("/:id/like", likePost);
router.get("/:id", getPost);
router.get("/", getPosts);
router.get("/from/:id", getUserPosts);

export default router;
```

Middleware : Authentification - Json Web Token

```
import jwt from "jsonwebtoken";

const requireAuth = async (req, res, next) => {
  const { authorization } = req.headers; Authorization header ?

  if (!authorization) { Echec requête : Token absent
    return res.status(401).json({ error: "Authorization token required" });
  }

  try {
    const token = authorization.split(" ")[1]; Secret
    const decodedToken = jwt.verify(token, process.env.JWT_KEY);
    const userId = decodedToken.userId;
    req.auth = {
      userId: userId, Récupérer id de l'utilisateur à partir du
      Token
    };
    next();
  } catch (error) {
    res.status(401).json({ error: "Request is not authorized" });
  }
};

export default requireAuth;
```

Posts - Backend : Model, Créer

Model : Post

```
import mongoose from "mongoose";                                Mongoose

const PostSchema = new mongoose.Schema(
{
    userId: {                                                 ID de l'utilisateur
        type: String,
        required: true,
    },
    desc: {                                                 Message
        type: String,
    },
    image: {                                                 Image
        type: String,
    },
    likes: {                                                 Tableau pour Likes
        type: [],
    },
},
{ timestamps: true }
);

const PostModel = mongoose.model("Post", PostSchema);
export default PostModel;
```

Controller : Créer un message

```
//? Create
export const createPost = async (req, res) => {
    const reqPost = req.body;                      Récupérer la requête
    const newPost = new Post({                     Appliquer le modèle Mongoose
        ...reqPost,
    });
    try {
        await newPost.save(); Sauvegarder en database
        res.status(200).json(newPost);
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
};
```

Database : Post

```
_id: ObjectId('63487d4812a70e0c41e53d94')
userId: "63487bfe12a70e0c41e53d57"
desc: "1. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse"
image: "16657808117717.jpg"
likes: Array
  0: "63487dd012a70e0c41e53da2"
  1: "63487bfe12a70e0c41e53d57"
  2: "63487f0312a70e0c41e53db4"
createdAt: 2022-10-13T21:04:08.214+00:00
updatedAt: 2022-10-15T00:02:21.910+00:00
__v: 0
```

Posts - Backend : Route upload + Multer

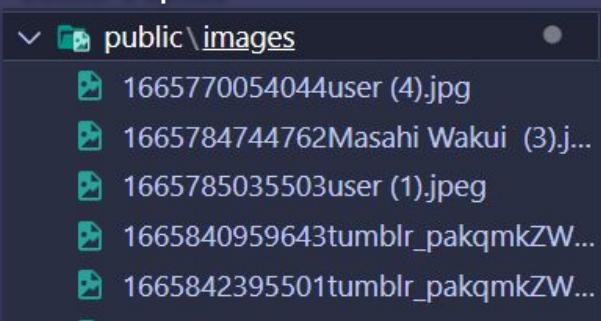
Appel routes dans index.js

```
app.use("/api/posts", post_router);
app.use("/api/upload", upload_router);
```

Accès dossier local

```
app.use(express.static("public"));
app.use("/images", express.static("images"));
```

Dossier d'upload



Router / Controller : Multer

```
import express from "express";
import multer from "multer";
import auth from "../middleware/auth_mw.js";

const router = express.Router();

const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, "public/images");
  },
  filename: (req, file, cb) => {
    cb(null, req.body.name);
  },
});

const upload = multer({ storage });

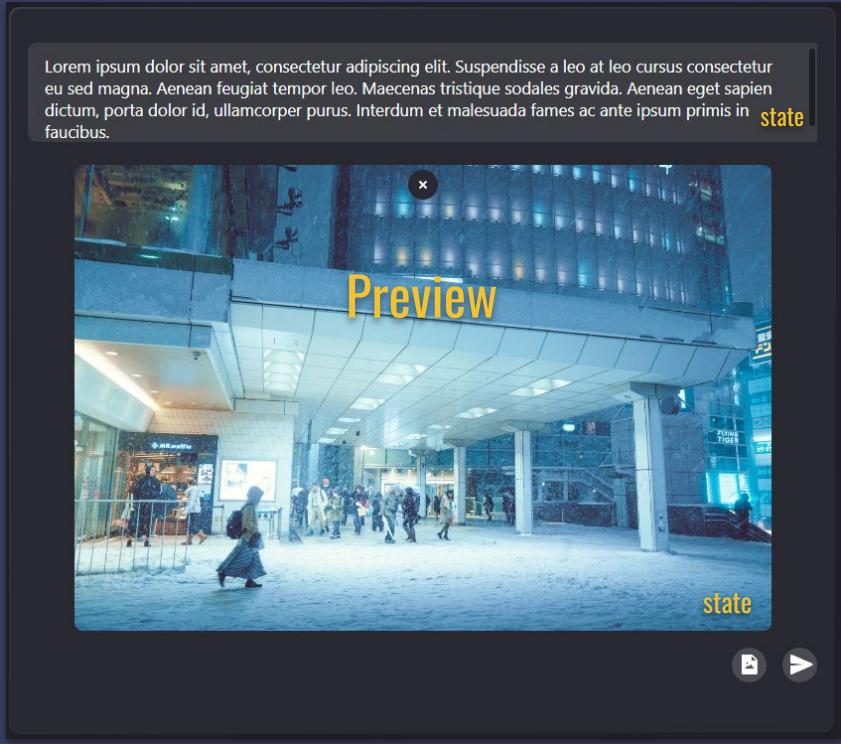
router.post("/", auth, upload.single("file"), (req, res) => {
  try {
    return res.status(200).json("File uploaded successfully");
  } catch (error) {
    console.log({ message: error.message });
  }
});

export default router
```

Configuration de Multer =
Destination du fichier +
Création du nom du fichier
Route d'upload

Posts - Frontend : Créer (1)

Composant : Create Post



```
return (
  <article className="createpost gradient-border">
    <form onSubmit={handleSubmit}>
      <textarea
        type="text"
        placeholder="Ecrire un message ... "
        value={desc}
        onChange={(e) => setDesc(e.target.value)}>
      />
      {file && (
        <div className="uploaded-image">
          <img src={URL.createObjectURL(file)} />
          <div className="close-icon" onClick={() => setFile(null)}>
            <img src={closeIcon} alt="remove" />
          </div>
        </div>
      )}
    </form>
  </article>
);
```

state message

Preview image

```
<div className="btns">
  <label htmlFor="image" aria-label="select file">
    <div>
      <img
        src={fileIcon}
        alt="select file"
        onClick={() => imageRef.current.click()}
      />
    </div>
  </label>
<input
  type="file"
  accept="image/png, image/jpeg, image/jpg, image/webp"
  ref={imageRef}
  onChange={onImageChange} Envoyer image dans state
/>
<button type="submit" aria-label="submit">
  <img src={sendIcon} alt="send" />
</button>
</div>
</form>
</article>
);
```

Choisir un fichier

Envoyer image dans state

Posts - Frontend : Créer (2)

Envoyer du Post

```
//? Submit post
const handleSubmit = async (e) => {
  e.preventDefault();
  if (desc === "" && file === null) { Annuler si : Post vide
    return;
  }
  const newPost = {
    userId: auth.user._id,          Récupérer id utilisateur
    desc: desc,
  };

  if (file) {                      Si image ...
    const data = new FormData();
    const fileName = Date.now() + file.name;
    data.append("name", fileName);
    data.append("file", file);
    newPost.image = fileName;      Requête route upload
    try {
      await axios.post("/api/upload", data, {
        headers: {
          Authorization: `Bearer ${auth.token}`,
        },
      });
    } catch (error) {
      console.log({ message: error.message });
    }
  }                                  Requête api
  try {
    const res = await axios.post("/api/posts", newPost, {
      headers: {
        Authorization: `Bearer ${auth.token}`,
      },
    });
    setDesc(""); Vider les states   Context : dispatch
    setFile(null);
    dispatch({ type: "CREATE_POST", payload: res.data });
  } catch (error) {
    console.log({ message: error.message });
  }
};
```

Context Post

```
import { createContext, useReducer } from "react";
export const PostsContext = createContext();

export const postsReducer = (state, action) => {
  switch (action.type) {
    case "SET_POSTS":
      return {
        posts: action.payload,
      };
    case "CREATE_POST":           Afficher
      return {
        posts: [action.payload, ...state.posts],
      };
    case "DELETE_POST":          Créer
      return {
        posts: state.posts.filter(
          (post) => post._id !== action.payload._id
        ),
      };
    case "UPDATE_POST":          Supprimer
      return {
        posts: state.posts.map((post) => {
          if (post._id !== action.payload._id) {
            return post;
          }
          return action.payload;
        }),
      };
    default:
      return state;
  };
};

export const PostsContextProvider = ({ children }) => {           Modifier
  const [state, dispatch] = useReducer(postsReducer, { posts: null });

  return (
    <PostsContext.Provider value={{ ...state, dispatch }}>
      {children}
    </PostsContext.Provider>   Reducer
  );
};
```

Appliquer dans le root de src/index.js

Posts - Backend : Afficher

Controller : Afficher tous les messages antéchronologiquement

```
//? Get all
export const getPosts = async (req, res) => {
  try {
    const posts = await Post.find(); Méthode .find
    res.status(200).json(
      posts
        .filter((post) => post != null)
        .sort((a, b) => {
          return new Date(b.createdAt) - new Date(a.createdAt);
        })
    );
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};
```

Database : Posts

```
_id: ObjectId('63487d4812a70e0c41e53d94')
userId: "63487bfe12a70e0c41e53d57"
desc: "1. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse..."
image: "16657808117717.jpg"
> likes: Array
createdAt: 2022-10-13T21:04:08.214+00:00
updatedAt: 2022-10-15T00:02:21.910+00:00
__v: 0
```

Date de création

```
_id: ObjectId('63487e4c12a70e0c41e53db0')
userId: "63487dd012a70e0c1e53da2"
desc: "2. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse..."
image: "16657807817712.jpg"
> likes: Array
createdAt: 2022-10-13T21:08:28.335+00:00
updatedAt: 2022-10-15T00:02:20.490+00:00
__v: 0
```

```
_id: ObjectId('63487ff312a70e0c41e53de0')
userId: "63487f0312a70e0c1e53db4"
desc: "3. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse..."
image: "16657808219851.jpg"
> likes: Array
createdAt: 2022-10-13T21:15:31.445+00:00
updatedAt: 2022-10-14T21:18:30.602+00:00
__v: 0
```

```
_id: ObjectId('6349d88a207401d6193dc533')
userId: "63487bfe12a70e0c41e53d57"
desc: "4. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse..."
image: "16657839464273.jpg"
> likes: Array
createdAt: 2022-10-14T21:45:46.453+00:00
updatedAt: 2022-10-14T21:46:05.046+00:00
__v: 0
```

Posts - Frontend : Afficher



Jane Doe
14/10/2022 à 23:45

Composant POST

8. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse a leo at leo cursus
consectetur eu sed magna. Aenean feugiat tempor leo. Maecenas tristique sodales gravida.
Aenean eget sapien dictum, porta dolor id, ullamcorper purus. Interdum et malesuada fames
ac ante ipsum primis in faucibus.



1 like Ecrire un commentaire... ▶



Admin
14/10/2022 à 23:45

Composant POST

7. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse a leo at leo cursus
consectetur eu sed magna. Aenean feugiat tempor leo. Maecenas tristique sodales gravida.
Aenean eget sapien dictum, porta dolor id, ullamcorper purus. Interdum et malesuada fames
ac ante ipsum primis in faucibus.



0 like Ecrire un commentaire... ▶

Composant
POST(S)

Composants : POSTS - Affichage de tous les composants POST

```
import { useEffect } from "react";
import axios from "axios";
import { usePostsContext } from "../../hooks/usePostsContext";
import { useAuthContext } from "../../hooks/useAuthContext";
import Post from "./Post";

const Posts = () => {
  const { posts, dispatch } = usePostsContext(); Context Post : affichage dynamique
  const { user: auth } = useAuthContext(); Context User : autorisation

  useEffect(() => {
    const getPosts = async () => {
      const res = await axios.get("/api/posts", {
        headers: {
          Authorization: `Bearer ${auth.token}`, Token autorisation
        },
      });
      dispatch({ type: "SET_POSTS", payload: res.data }); Context : dispatch
    };
    getPosts();
  }, [auth.token, dispatch]);

  return (
    <div className="posts">
      {posts && posts.map((post) => <Post post={post} key={post._id} />)}
    </div> Méthode map -> applique le composant "POST" à chaque objet post
  );
}

export default Posts;
```

Requête GET api

Posts - Backend : Modifier + Supprimer

Controller : Modifier un message

```
//? Update
export const updatePost = async (req, res) => {
  const postId = req.params.id;          Récupérer ID du Post +
  const { userId, admin } = req.body;    Données de l'utilisateur

  try {
    const post = await Post.findById(postId);  Correspondance ID database
    if (admin || post.userId === userId) { Autorisation
      if ((req.body.image && post.image) || post.image) {
        const filename = post.image.split("public/images/")[0];
        fs.unlink(`public/images/${filename}`, () => {});           Supprimer ancienne image hébergée
      }

      await post.updateOne({ $set: req.body });
      res.status(200).json(req.body);           Modifier avec nouvelle données
    } else {
      res.status(403).json("You can only update your own posts !");
    }
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};
```

Controller : Supprimer un message

```
//? Delete
export const deletePost = async (req, res) => {
  const postId = req.params.id;          Récupérer ID du Post +
  const { userId, admin } = req.body;    Données de l'utilisateur
  try {
    const post = await Post.findById(postId);  Correspondance ID database
    if (admin || post.userId === userId) {
      if (post.image) {                     Supprimer ancienne image hébergée
        const filename = post.image.split("public/images/")[0];
        fs.unlink(`public/images/${filename}`, () => {
          post.deleteOne();
          res.status(200).json(post);
        });
      } else {
        post.deleteOne();                  Supprimer de la database
        res.status(200).json(post);
      }
    } else {
      res.status(403).json("You can only delete your own posts !");
    }
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};
```

Posts - Frontend : Modifier + Supprimer

Condition : Affichage modal Modifier / Supprimer

```
{auth.user.admin || (auth.user._id === post.userId) ? (
  <PostOptionsModal options={options} />
) : null}
```



Composant : Post - Requêtes

```
//? Delete Post
const handleDelete = async () => {
  const req = {
    userId: auth.user._id,
    admin: auth.user.admin, Autorisation
  };
  if (auth.user.admin || auth.user._id === post.userId) {
    try {
      const res = await axios.delete(`./api/posts/${post._id}`); Requête DELETE api
      {
        data: req,
        headers: {
          Authorization: `Bearer ${auth.token}`,
        },
      }
    };
    dispatch({ type: "DELETE_POST", payload: res.data });
  } catch (error) {
    console.log({ message: error.message });
  }
};

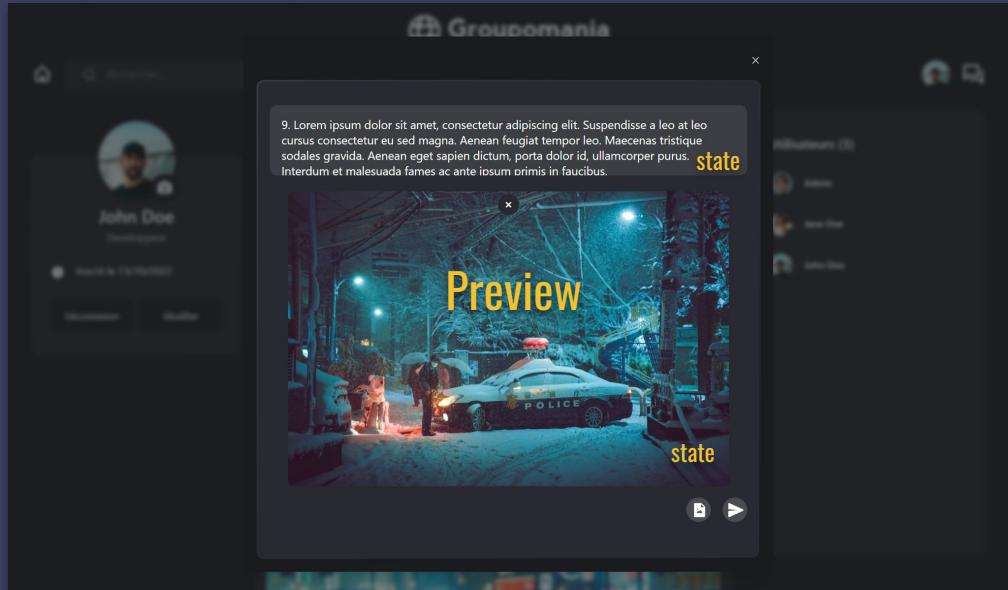
//? Update Post
const handleUpdate = () => { Autorisation
  if (auth.user.admin || auth.user._id === post.userId) {
    setUpdatePostModal(true); Afficher Modal de modification
  }
};
```

SUPPRIMER

MODIFIER

Posts - Frontend : Modifier (modal)

Composant : Modal Update Post



```
//? Submit updated post
const handleSubmit = async (e) => {
  e.preventDefault();
  if (updatePost.desc === "" && file === null) { Si message vide ...
    return;
  }
  if (auth.user.admin || auth.user._id === data.userId) {
    try {
      if (file) { Si image ...
        const data = new FormData();
        const fileName = Date.now() + file.name;
        data.append("name", fileName);
        data.append("file", file);
        updatePost.image = fileName;
        try {
          await axios.post("/api/upload", data, {
            headers: {
              Authorization: `Bearer ${auth.token}`,
            },
          });
        } catch (error) {
          console.log({ message: error.message });
        }
      } else {
        updatePost.image = null;
      }
      const res = await axios.put(`api/posts/${data._id}`, updatePost, {
        headers: {
          Authorization: `Bearer ${auth.token}`,
        },
      });
      Context : dispatch
      dispatch({ type: "UPDATE_POST", payload: res.data });
      setUpdatePostModal(false);
      setFile(null);
    } catch (error) {
      console.log({ message: error.message });
    }
  }
};
```

1 - Backend : Post controller - Likes

```
//? Like / Dislike
export const likePost = async (req, res) => {
  const postId = req.params.id; Quel post ? Quel utilisateur ?
  const { userId } = req.body;
  try {
    const post = await Post.findById(postId);
    if (!post.likes.includes(userId)) { Likes n'inclue pas user
      await post.updateOne({ $push: { likes: userId } });
      res.status(200).json("Post liked !");
    } else { Likes inclue déjà user
      await post.updateOne({ $pull: { likes: userId } });
      res.status(200).json("Post disliked !");
    }
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};
```

3 - Frontend : Bouton / Affichage de l'icône

```
<div className="post__reactions">
  <img state
    src={liked ? likeIcon : likeOutlined}
    alt="like"
    onClick={handleLikes}
    title="Aimer ce message"
  />
  <p>{like}</p>
</div>
```



The component displays a button with a thumbs-up icon and the number '1' above it, indicating one like. Below the button, the number '0' is shown, likely representing the total count or a different metric.

Post - Likes

2 - Frontend : Requête

```
//? Like system          Post est-il liké ?           STATES
const [liked, setLiked] = useState(false);
const [like, setLike] = useState(post.likes.length);
useEffect(() => { POST déjà liké ?
  setLiked(post.likes.includes(auth.user._id));
}, [auth.user._id, post.likes]);
```

Nombre de likes

```
const handleLikes = () => { Requête PUT api
  axios.put(
    "/api/posts/" + post._id + "/like",
    { userId: auth.user._id }, Ajouter userId au tableau LIKES
    {
      headers: {
        Authorization: `Bearer ${auth.token}`,
      },
    }
  );
  Si déjà liké ? = -1 / Sinon +1
  setLike(liked ? like - 1 : like + 1);
  setLiked(!liked); Modifier le state de false à true, etc.
};
```

Bilan

Objectifs atteints

- Utilisation d'un framework JavaScript
- Logo + Police respectés, couleurs libres
- Design totalement responsive
- Affichage postes ordre antéchronologique
- Connexion persistante, Déconnexion
- Données du site stockées en database (users/posts)
- Données utilisateurs sécurisées en database (mail/pass)
- Créer / Modifier / Supprimer les postes respectifs
- Un poste peut contenir texte/image
- Système de Likes pour les postes
- Administrateur peut supprimer/modifier tous les postes
- Possibilité de cloner et lancer l'application (test réalisé)
- Respect des standards d'accessibilité WCAG (test Wave)

Axes d'amélioration

- Page de profil pour chaque utilisateur
- Page pour chaque poste (meilleure visibilité)
- Système de commentaires
- Barre de recherche fonctionnelle
- Afficher les utilisateurs connectées (volet droite)
- Implémenter un messenger avec socket.IO
- Améliorer l'édition de messages
- Renforcer la sécurité niveau backend (OWASP)
- Optimiser le code pour meilleures performances
- Choix de thème light/dark
- Guider l'utilisateur dans le formulaire d'inscription
- Contrôler le poids/dimensions max d'une image ajoutée
-
-