

Groupomania



Groupomania

Faites rager vos collègues avec vos photos de vacances

Inscription

Prénom	<input type="text"/>	Nom	<input type="text"/>
Adresse e-mail		<input type="text"/>	
Mot de passe		<input type="password"/>	

Inscription

[Déjà un compte? Connectez-vous.](#)



Groupomania

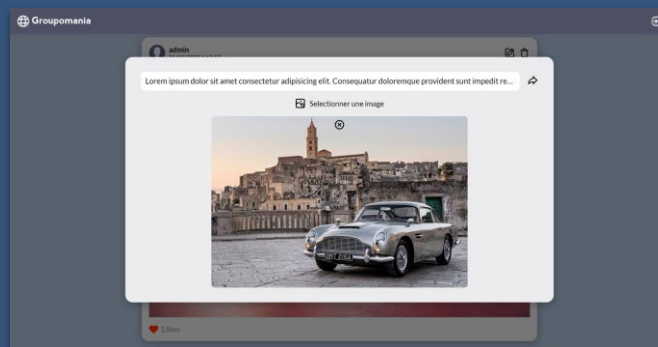
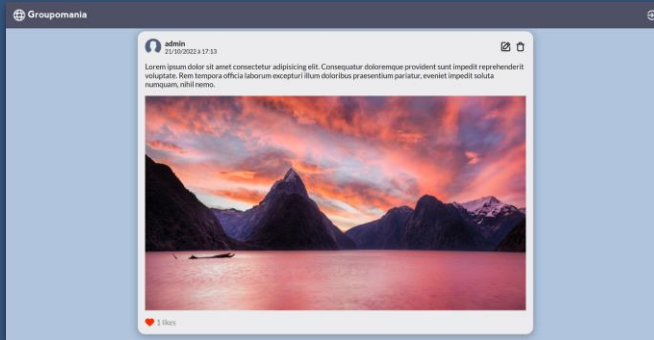
Faites rager vos collègues avec vos photos de vacances

Connexion

Adresse e-mail	<input type="text"/>
Mot de passe	<input type="password"/>

Connexion

[Vous n'avez pas de compte? Inscrivez-vous!](#)



Contexte

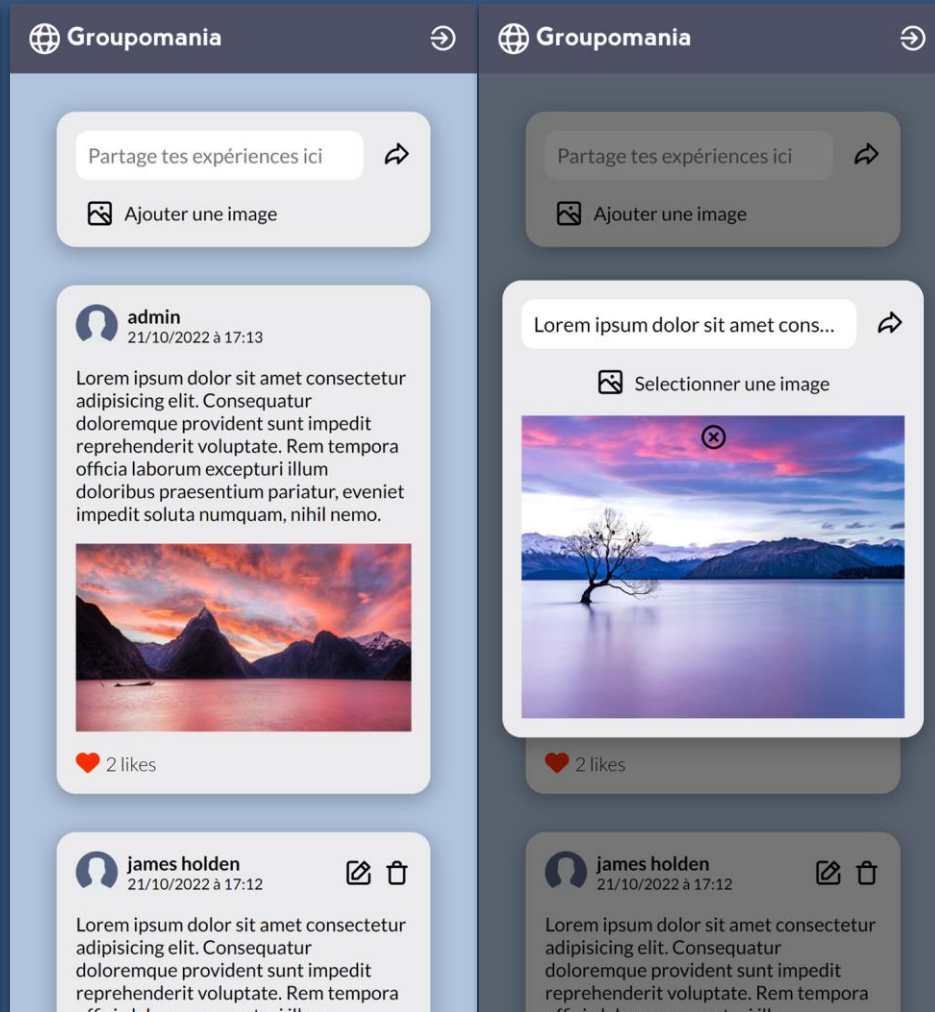
- Le client souhaite un réseau social interne à l'entreprise pour faciliter l'interaction entre collègues.
- Réalisation du backend, du frontend avec React et de la base de données avec mongoDB en partant de zéro

Mission et critères à respecter

- Le site doit être responsive avec le logo et la police Lato
- Les données d'inscription doivent être sécurisées en database
- La connexion doit être persistante jusqu'à la déconnexion
- La déconnexion doit nous retourner sur la page de connexion
- La création, modification et suppression des postes doit comporter du texte et/ou une image
- L'utilisateur à la possibilité d'aimer les postes
- Un compte admin doit pouvoir modifier et supprimer tous les postes dans le but de pouvoir faire de la modération

Présentation





- 1 - Démonstration des fonctionnalités
- 2 - Présentation détaillé du code
- 3 - Bilan & Axes d'amélioration



1. Démonstration des fonctionnalités

2. Database et Organisation

Database - MongoDB

Collections					MongoDB Compass				
Create collection					View  				
					Sort by Collection Name  				
posts									
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:					
20.48 kB	6	556.00 B	1	36.86 kB					
users									
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:					
20.48 kB	5	211.00 B	2	73.73 kB					

```
_id: ObjectId('6352b728955d9722ae4673d1')
userId: "6346dfdd81fa1333d480cd80"
desc: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequatur d..."
image: "1666365224043New Zealand (2).jpg"
likes: Array
  0: "6346dfdd81fa1333d480cd80"
createdAt: 2022-10-21T15:13:44.061+00:00
updatedAt: 2022-10-21T19:33:09.763+00:00
__v: 0
```

Post

```
_id: ObjectId('6346dfdd81fa1333d480cd80')
email: "admin.test@test.fr"
password: "$2b$10$9Fxe5ehc5wHYC7ZxwPEDOL8mdXFN.J2d1CHNedYvm8q3LevYL1HO"
firstname: "admin"
lastname: "admin"
admin: true
__v: 0
```

User

Backend - Organisation

Structure

- api
 - controllers
 - JS auth.controllers.js
 - JS post.controllers.js
 - JS user.controllers.js
 - middleware
 - JS auth.js
 - JS validator.mail.js
 - JS validator.pass.js
 - models
 - JS Post.models.js
 - JS User.models.js
 - node_modules
 - public\images
 - routes
 - JS auth.routes.js
 - JS post.routes.js
 - JS upload.routes.js
 - JS user.routes.js
 - services
 - JS database.js
 - .env
 - .env.sample
 - JS index.js
 - package.json
 - package-lock.json

package.json

```
"dependencies": {  
  latest  
  "bcrypt": "^5.1.0",  
  latest  
  "cors": "^2.8.5",  
  latest  
  "dotenv": "^16.0.3",  
  latest  
  "express": "^4.18.2",  
  latest  
  "helmet": "^6.0.0",  
  latest  
  "jsonwebtoken": "^8.5.1",  
  latest  
  "mongoose": "^6.6.7",  
  latest  
  "mongoose-unique-validator": "^3.1.0",  
  latest prerelease ^1.4.5-lts.1  
  "multer": "^1.4.5-lts.1",  
  latest  
  "password-validator": "^5.3.0",  
  latest  
  "path": "^0.12.7",  
  latest  
  "validator": "^13.7.0"  
},  
"devDependencies": {  
  latest  
  "nodemon": "^2.0.20"  
}
```

index.js

```
const express = require("express");  
const helmet = require("helmet");  
const cors = require("cors");  
require("dotenv").config();  
require("./services/database");  
  
/** IMPORT ROUTE  
const authRoutes = require("./routes/auth.routes");  
const userRoutes = require("./routes/user.routes");  
const postRoutes = require("./routes/post.routes");  
const uploadRoute = require("./routes/upload.routes")  
  
/** APP  
const app = express();  
const port = process.env.PORT || 4200;  
  
/** MIDDLEWARE  
app.use(helmet({ crossOriginResourcePolicy: { policy: "same-site" } }));  
app.use(cors());  
app.use(express.json());  
  
/** IMAGES LOCAL  
app.use(express.static("public"));  
app.use("/images", express.static("images"));  
  
/** ROUTES  
app.use("/api/auth", authRoutes);  
app.use("/api/users", userRoutes);  
app.use("/api/posts", postRoutes);  
app.use("/api/upload", uploadRoute);  
  
/** LANCEMENT SUR LE PORT  
app.listen(port, () => console.log("Listening on port : " + port));
```

database.js

```
const mongoose = require("mongoose");  
  
mongoose  
  .connect(process.env.MONGODB_URI)  
  .then(() => {  
    console.log("Connexion à MongoDB réussie !");  
  })  
  .catch((error) => {  
    console.error("Connexion à MongoDB échouée ! " + error);  
  });  
  
module.exports = { mongoose };
```

Frontend - Organisation

Structure

```

└─ client
  └─ node_modules
  └─ public
  └─ src
    └─ assets
    └─ components
    └─ context
      └─ AuthContext.js
      └─ PostContext.js
    └─ hooks
      └─ useAuthContext.js
      └─ useLogin.js
      └─ useLogout.js
      └─ usePostsContext.js
      └─ useSignup.js
    └─ pages
      └─ app.css
      └─ App.js
      └─ index.js
    └─ .env
    └─ .env.sample
  └─ package.json
  └─ package-lock.json
```

package.json

```

{
  "proxy": "http://localhost:4200",
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "latest":
      "@emotion/cache": "^11.10.3",
    "latest":
      "@emotion/react": "^11.10.4",
    "latest":
      "@emotion/serialize": "^1.1.0",
    "latest":
      "@emotion/styled": "^11.10.4",
    "latest":
      "@emotion/units": "^1.2.0",
    "latest":
      "@mui/material": "^5.10.10",
    "latest":
      "@tabler/icons": "^1.107.0",
    "latest":
      "@testing-library/jest-dom": "^5.16.5",
    "latest":
      "@testing-library/react": "^13.4.0",
    "latest":
      "@testing-library/user-event": "^14.4.3",
    "latest":
      "date-fns": "^2.29.3",
    "latest":
      "dotenv": "^16.0.3",
    "latest":
      "react": "^18.2.0",
    "latest":
      "react-dom": "^18.2.0",
    "latest":
      "react-router-dom": "^6.4.2",
    "latest":
      "react-scripts": "5.0.1",
    "latest":
      "web-vitals": "^3.0.4"
  },
}
```

Frontend - Accessibilité

The image shows a web application interface for 'Groupomania' with a Wave accessibility evaluation tool overlay on the left. The tool reports 0 errors, 1 alert, and 2 structural elements. The application header includes the Groupomania logo and a 'Déconnexion' button. The main content area features a 'Partage tes expériences ici' section with a 'Partager' button and a 'Selectionner' button for adding images. Below this is a user profile for 'admin' with a default profile photo and a 'delete' button. The profile bio text is: 'Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequatur doloremque provident sunt impedit reprehenderit voluptate. Rem tempora officia laborum excepturi illum doloribus praesentium pariatur, eveniet impedit soluta numquam, nihil nemo.' The profile picture is a landscape image with a red sky and mountains. A 'Code' button is visible at the bottom of the profile section.

Wave Accessibility Evaluation Tool Summary:

- Errors: 0
- Contrast Errors: 0
- Alerts: 1
- Features: 35
- Structural Elements: 2
- ARIA: 3

[View details](#)

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

Groupomania Header:

- Groupomania logo
- Logo Groupomania
- Déconnexion

Main Content:

Partage tes expériences ici

Partager

Selectionner Ajouter une image

aria-label="Partage tes expériences ici"

aria-label="Soumettre"

aria-label="Selectionner une image"

User Profile:

admin

21/10/2022 à 17:13

edit delete

photo de profil par défaut

Code

bio text: Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequatur doloremque provident sunt impedit reprehenderit voluptate. Rem tempora officia laborum excepturi illum doloribus praesentium pariatur, eveniet impedit soluta numquam, nihil nemo.

Frontend - React Router

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>
</html>
```

```
import React from "react";
import ReactDOM from "react-dom/client";
import { AuthContextProvider } from "../context/AuthContext";
import { PostsContextProvider } from "../context/PostContext";
import App from "../App";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <AuthContextProvider>
      <PostsContextProvider>
        <App />
      </PostsContextProvider>
    </AuthContextProvider>
  </React.StrictMode>
);
```

index.js

app.js

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Signup from "../pages/Auth/Signup";
import Login from "../pages/Auth/Login";
import Home from "../pages/Home/Home";
import NotFound from "../pages/NotFound/NotFound";
import "../app.css";

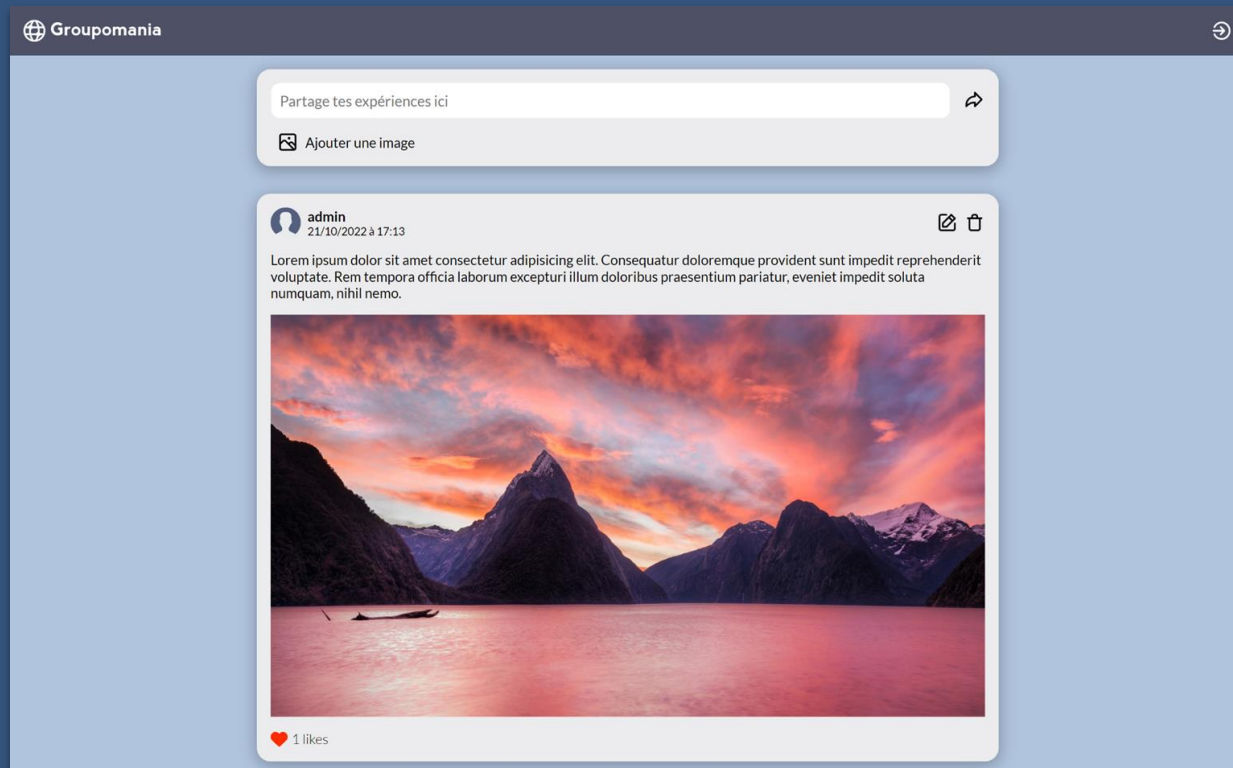
function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/signup" element={<Signup />} />
        <Route path="/" element={<Login />} />
        <Route path="/home" element={<Home />} />
        <Route path="*" element={<NotFound />} />
      </Routes>
    </BrowserRouter>
  );
};

export default App;
```

Frontend - Composants

components

- > Auth
- > Post
- > Posts
- > PostShare
- > PostUpdate
- > TopBar



2. Authentication

Backend - Inscription (1)

Model
User

```
const mongoose = require("mongoose");
const uniqueValidator = require("mongoose-unique-validator");

const userSchema = new mongoose.Schema({
  email: {
    type: String,
    required: true,
    lowercase: true,
    unique: true,
    trim: true,
  },
  password: {
    type: String,
    required: true,
    minlength: 8,
  },
  firstname: {
    type: String,
    required: true,
    lowercase: true,
    trim: true,
  },
  lastname: {
    type: String,
    required: true,
    lowercase: true,
    trim: true,
  },
  admin: {
    type: Boolean,
    default: false,
  },
}, {
  timestamps: true
});

userSchema.plugin(uniqueValidator);
module.exports = mongoose.model("User", userSchema);
```

Controller
User

```
const bcrypt = require("bcrypt");
const jwt = require("jsonwebtoken");
const User = require("../models/User.models");

/** SIGNUP */
const signup = async (req, res) => {
  const salt = await bcrypt.genSalt(10);
  const hashedPass = await bcrypt.hash(req.body.password, salt);
  req.body.password = hashedPass;

  try {
    let user = await User.findOne({ email: req.body.email });
    if (user) {
      return res.status(400).json({ error: "Email déjà utilisé" });
    }
    user = await User.create({ ...req.body });
    res.status(200).json({ user });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};
```

Backend - Inscription (2)

Middleware validation email

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

Router Authentification

```
const router = require("express").Router();
const { signup, login } = require("../controllers/auth.controllers")
const mail = require ('../middleware/validator.mail')
const pass = require ('../middleware/validator.pass')

/** ROUTES
router.post("/signup", mail, pass, signup);
router.post("/login", login);

module.exports = router;
```

Middleware validation password

```
const passwordValidator = require("password-validator");
const passwordSchema = new passwordValidator();
passwordSchema
  .is().min(8)
  .is().max(32)
  .has().digits(1)
  .has().uppercase(1)
  .has().not().spaces();
const validPassword = (req, res, next) => {
  if (passwordSchema.validate(req.body.password)) {
    next();
  } else {
    return res.status(403).json({
      error: `Le mot de passe doit contenir entre 8 et 32 caractères avec au moins un chiffre et une majuscule`
    });
  }
};
module.exports = validPassword;
```

Frontend - Inscription

Composant Signup



Faites rager vos collègues avec vos photos de vacances

Inscription

Inscription

[Déjà un compte? Connectez-vous!](#)

Formulaire d'inscription

```
import { useState } from "react";
import { Link } from "react-router-dom";
import { useSignup } from "../../hooks/useSignup";

const Signup = () => {
  const [firstname, setFirstname] = useState("");
  const [lastname, setLastname] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const { signup, error, isLoading } = useSignup();
  const handleSignup = async (e) => {
    e.preventDefault();
    await signup(email, password, firstname, lastname);
  };

  return (
    <div>
      <form className="authForm signup" onSubmit={handleSignup}>
        <h3>Inscription</h3>
        <div className="grid">
          <input
            required
            aria-label="Prénom"
            placeholder="Prénom"
            className="authFormInput"
            type="text"
            onChange={(e) => setFirstname(e.target.value)}
            value={firstname}
            pattern="^(?![\s.]+)$[A-zÀ-ú\s\~]{1,25}$"
          />

```

Hook useSignup

```
import { useState } from "react";
import { useNavigate } from "react-router-dom";

export const useSignup = () => {
  const [error, setError] = useState(null);
  const [isLoading, setIsLoading] = useState(null);
  let navigate = useNavigate();

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

    const response = await fetch("/api/auth/signup", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ email, password, firstname, lastname })
    });
    const json = await response.json();

    if (!response.ok) {
      setError(json.error);
    }
    if (response.ok) {
      navigate("/");
    }
    setIsLoading(false);
  };

  return { signup, isLoading, error };
};
```

Backend - Connexion

Controller Login

```
/* LOGIN
const login = async (req, res) => {
  const { email, password } = req.body;

  if (!email || !password) {
    return res.status(400).json({ error: "Tous les champs doivent être remplis" });
  }
  try {
    const user = await User.findOne({
      email: email,
    });
    if (user) {
      const validpass = await bcrypt.compare(password, user.password);
      if (!validpass) {
        res.status(400).json({ error: "Mot de passe incorrect" });
      } else {
        const token = jwt.sign({
          mail: user.email,
          id: user._id,
        },
          process.env.JWT_TOKEN,
          { expiresIn: process.env.JWT_TIME }
        );
        res.status(200).json({ user, token });
      }
    } else {
      res.status(404).json({ error: "Adresse email incorrecte" });
    }
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};
```

Router User

```
const router = require("express").Router();
const { signup, login } = require("../controllers/auth.controllers");
const mail = require("../middleware/validator.mail");
const pass = require("../middleware/validator.pass");

/* ROUTES
router.post("/signup", mail, pass, signup);
router.post("/login", login);

module.exports = router;
```

Frontend - Connexion (1)

Formulaire de connexion

```
import { useState } from "react";
import { Link } from "react-router-dom";
import { useLogin } from "../../hooks/useLogin";

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

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

  return (
    <form className="authForm login" onSubmit={handleLogin}>
      <h3>Connexion</h3>
      <input
        aria-label="Adresse e-mail"
        placeholder="Adresse e-mail"
        className="authFormInput"
        type="email"
        onChange={(e) => setEmail(e.target.value)}
        value={email}
      />
      <input
        aria-label="Mot de passe"
        className="authFormInput"
        placeholder="Mot de passe"
        type="password"
        onChange={(e) => setPassword(e.target.value)}
        value={password}
      />
      {error && <div className="errorAuth">{error}</div>}
      <button disabled={!isLoading} className="button authFormButton" type="submit">
        Connexion
      </button>
      <div>
        <button className="authFormText">
          <Link to="/signup">
            Vous n'avez pas de compte? Inscrivez-vous!
          </Link>
        </button>
      </div>
    </form>
  );
};

export default Login;
```

Hook useLogin

```
import { useState } from "react";
import { useNavigate } from "react-router-dom";
import { useAuthContext } from "../useAuthContext";

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

  const login = async (email, password) => {
    setIsLoading(true);
    setError(null);

    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));
      navigate("/home");
      // update the auth context
      dispatch({ type: "LOGIN", payload: json });
      setIsLoading(false);
    }
  };

  return { login, isLoading, error };
};
```

Composant Login



Groupeomania

Faites rager vos collègues avec vos photos de vacances

Connexion

Adresse e-mail



Mot de passe



Connexion

[Vous n'avez pas de compte? Inscrivez-vous!](#)

Frontend - Connexion (2)

Context AuthContext

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

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

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

  useEffect(() => {
    const user = JSON.parse(localStorage.getItem("user"));
    if (user) {
      dispatch({ type: "LOGIN", payload: user });
    }
  }, []);
  console.log("AuthContext state: ", state);

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

Local Storage

```
{user: {_id: "6352b4e3955d9722ae467333", email: "james.holden@gmail.com",...},
  token: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtYWlsIjoiamFtZXMuG9sZGVuQ",
  ▶ user: {_id: "6352b4e3955d9722ae467333", email: "james.holden@gmail.com",...}}
```

Console

```
AuthContext state: AuthContext.js:2
▼ Object 1
  ▶ user: {user: {...}, token: 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
```

Frontend - Déconnexion

Bouton déconnexion



Composant Topbar

```
<Link to="/">
  <img
    className="topBarLogout"
    src={signOut}
    alt="Déconnexion"
    onClick={handleLogout} />
</Link>
```

Fonction HandleLogout

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

Hook Logout

```
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 };
};
```

Retour au login

 **Groupomania**

Faites rager vos collègues avec vos photos de vacances

Connexion

Connexion

[Vous n'avez pas de compte? Inscrivez-vous!](#)

Console : localStorage vidé

```
AuthContext state:
▼ {user: null} ⓘ
  user: null
  ► [[Prototype]]: Object
```

2. Posts

Backend - Routeur + Authentification

Router Posts

```
const express = require("express");
const requireAuth = require("../middleware/auth");
const {
  createPost,
  getAllPosts,
  getOnePost,
  modifyPost,
  deletePost,
  likePost,
} = require("../controllers/post.controllers");
const router = express.Router();

/** ROUTES
router.use(requireAuth);
router.post("/", createPost);
router.get("/", getAllPosts);
router.get("/:id", getOnePost);
router.put("/:id", modifyPost);
router.delete("/:id", deletePost);
router.post("/:id/like", likePost);

module.exports = router;
```

Middleware d'authentification avec JWT

```
const jwt = require("jsonwebtoken");

/** AUTHENTICATION USER
const requireAuth = async (req, res, next) => {
  const { authorization } = req.headers;
  if (!authorization) {
    return res.status(401).json({ error: "Authorization token required" });
  }
  try {
    const token = authorization.split(" ")[1];
    const decodedToken = jwt.verify(token, process.env.JWT_TOKEN);
    const userId = decodedToken.id;
    req.auth = {
      userId: userId,
    };
    next();
  } catch (error) {
    console.log(error);
    res.status(401).json({ error: "Request is not authorized" });
  }
};

module.exports = requireAuth;
```

Backend - Model + Créer un post

Model Post

```
const mongoose = require("mongoose");

const postSchema = mongoose.Schema(
  {
    userId: {
      type: String,
      required: true
    },
    desc: {
      type: String,
      required: false
    },
    image: {
      type: String
    },
    likes: {
      type: []
    },
  },
  { timestamps: true }
);

module.exports = mongoose.model("Post", postSchema)
```

Controller Créer un post

```
/* CREATE */
const createPost = async (req, res) => {
  const reqPost = req.body;
  const newPost = new Post({
    ...reqPost,
  });
  try {
    await newPost.save();
    res.status(200).json(newPost);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};
```

Backend - Upload d'image + Multer

Routes dans index.js

```
app.use("/api/posts", postRoutes);  
app.use("/api/upload", uploadRoute);
```

Dossier local

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

Dossier d'upload

📁 public\images

- 🖼️ 1666365026692jason-clarke-mars-overflight-final-00
- 🖼️ 1666365101296quatre-aston-dans-le-nouveau-bon
- 🖼️ 1666365125440New Zealand (1).jpg
- 🖼️ 1666365171674New Zealand (6).jpg
- 🖼️ 1666365224043New Zealand (2).jpg
- 🖼️ 1666450133183New Zealand (3).jpg

Router + Controller Multer

```
const express = require("express");  
const multer = require("multer");  
  
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("/", upload.single("file"), (req, res) => {  
  try {  
    return res.status(200).json("File uploaded successfully")  
  } catch (error) {  
    console.log({ error: error.message });  
  }  
});  
  
module.exports = router;
```

```

return (
  <section className="postShare">
    <form onSubmit={handleShare}>
      <div>
        <input
          type="text"
          placeholder="Partage tes expériences ici"
          onChange={(e) => setDesc(e.target.value)}
          value={desc}
          aria-label="Partage tes expériences ici"
        />
        <button type="submit" aria-label="Soumettre">
          <img className="postShareButton" src={share} alt="Partager"/>
        </button>
      </div>
      {error && <div className="error">{error}</div>}
      <div className="postShareOptions">
        <div onClick={() => imageRef.current.click()}>
          <img src={image} alt="Selectionner" />Ajouter une image
        </div>
        <input
          className="postShareOption"
          type="file"
          accept="image/png, image/jpeg, image/jpg, image/webp"
          ref={imageRef}
          onChange={onImageChange}
          aria-label="Selectionner une image"
        />
      </div>
      {file && (
        <div className="uploaded-image">
          <img src={URL.createObjectURL(file)} alt="Image du post"/>
          <div className="close-icon" onClick={() => setFile(null)}>
            <img src={crossRemove} alt="Supprimer l'image"/>
          </div>
        </div>
      )}
    </form>
  </section>
);

```

Création du Post

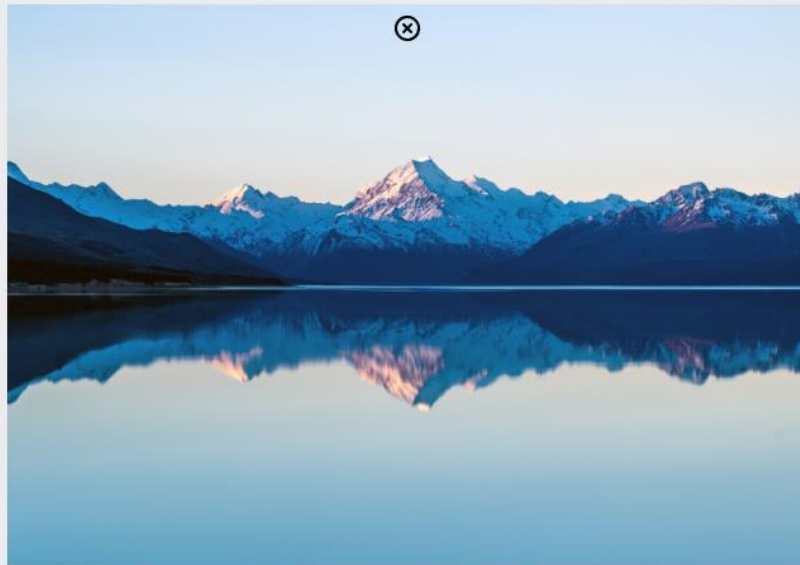
Frontend - Créer Post (1)

Composant PostShare

Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequatur dolorem



Ajouter une image



Envoi du Post

```
const handleShare = async (e) => {
  e.preventDefault();
  if (!currentUser) {setError("Vous devez être connecté");
    return;
  }
  if (desc === "" && file === null) {
    setError("Vous devez ajouter une description et/ou une photo");
    return;
  }
  const post = {
    userId: currentUser.user._id,
    desc: desc,
  };
  if (file) {
    const data = new FormData();
    const fileName = Date.now() + file.name;
    data.append("name", fileName);
    data.append("file", file);
    post.image = fileName;
    try {
      await fetch("/api/upload", {
        method: "POST",
        headers: { Authorization: `Bearer ${currentUser.token}` },
        body: data,
      });
    } catch (error) { console.log({ message: error.message }) }
  }
  try {
    const response = await fetch("/api/posts", {
      method: "POST",
      body: JSON.stringify(post),
      headers: {
        "Content-Type": "application/json",
        Authorization: `Bearer ${currentUser.token}`,
      },
    });
  };
  const json = await response.json();
  setDesc("");
  setFile(null);
  setError(null);
  dispatch({ type: "CREATE_POST", payload: json });
} catch (error) {
  console.log({ message: error.message });
}
};
```

Frontend - Créer Post (2)

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":
      return { posts: [action.payload, ...state.posts] };
    case "DELETE_POST":
      return { posts: state.posts.filter(
        (post) => post._id !== action.payload._id
      ) };
    case "UPDATE_POST":
      return {
        posts: state.posts.map((post) => {
          if (post._id !== action.payload._id) {
            return post;
          }
          return action.payload;
        }),
      };
    default:
      return state;
  }
};

export const PostsContextProvider = ({ children }) => {
  const [state, dispatch] = useReducer(postsReducer, { posts: null });
  return (
    <PostsContext.Provider value={{ ...state, dispatch }}>
      {children}
    </PostsContext.Provider>
  );
};
```


Backend - Afficher les posts

Controller Afficher tous les posts

```
/** GET ALL
const getAllPosts = async (req, res) => {
  try {
    const posts = await Post.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({ error: error.message });
  }
};
```

Posts dans la database

```
{
  _id: ObjectId('6352b6c5955d9722ae467399')
  userId: "6352b4e3955d9722ae467333"
  desc: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequatur d..."
  image: "1666365125440New Zealand (1).jpg"
  > likes: Array
  createdAt: 2022-10-21T15:12:05.780+00:00
  updatedAt: 2022-10-21T15:13:20.729+00:00
  __v: 0
}

{
  _id: ObjectId('6352b6e3955d9722ae46739d')
  userId: "6352b4e3955d9722ae467333"
  desc: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequatur d..."
  image: "1666450133183New Zealand (3).jpg"
  > likes: Array
  createdAt: 2022-10-21T15:12:41.481+00:00
  updatedAt: 2022-10-22T14:48:53.535+00:00
  __v: 0
}

{
  _id: ObjectId('6352b6f3955d9722ae4673a1')
  userId: "6352b4e3955d9722ae467333"
  desc: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequatur d..."
  image: "1666365171674New Zealand (6).jpg"
  > likes: Array
  createdAt: 2022-10-21T15:12:51.998+00:00
  updatedAt: 2022-10-23T22:56:09.270+00:00
  __v: 0
}

{
  _id: ObjectId('6352b728955d9722ae4673d1')
  userId: "6346dfdd81fa1333d480cd80"
  desc: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequatur d..."
  image: "1666365224043New Zealand (2).jpg"
  > likes: Array
  createdAt: 2022-10-21T15:13:44.061+00:00
  updatedAt: 2022-10-24T00:19:50.066+00:00
  __v: 0
}
```

Frontend - Afficher les posts

Composants Posts

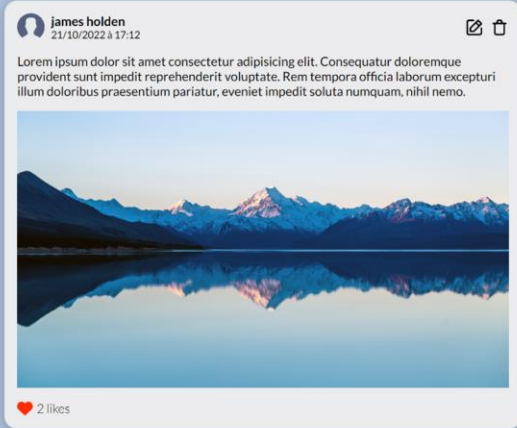
```
import { useEffect } from "react";
import { useAuthContext } from "../../hooks/useAuthContext";
import { usePostsContext } from "../../hooks/usePostsContext";
import Post from "../Post/Post";

const Posts = () => {
  const { user } = useAuthContext();
  const { posts, dispatch } = usePostsContext();

  useEffect(() => {
    const fetchPosts = async () => {
      const response = await fetch("/api/posts", {
        headers: {
          Authorization: `Bearer ${user.token}`,
        },
      });
      const json = await response.json();
      if (response.ok) {
        dispatch({ type: "SET_POSTS", payload: json });
      }
    };
    if (user) {
      fetchPosts();
    }
  }, [user, dispatch]);

  return (
    <section>
      {posts && posts.map((post) => <Post post={post} key={post._id} />)}
    </section>
  );
};

export default Posts;
```



Backend - Modifier + Supprimer

Controller Modifier un post

```
/* MODIFY
const modifyPost = async (req, res) => {
  const postId = req.params.id;
  const { userId, admin } = req.body;
  try {
    const post = await Post.findById(postId);
    if (admin || post.userId === userId) {
      if ((req.body.image && post.image) || post.image) {
        const filename = post.image.split("public/images/")[0];
        unlink(`public/images/${filename}`, () => {});
      }
      await post.updateOne({ $set: req.body });
      res.status(200).json(req.body);
    } else {
      res.status(403).json({
        error : "Vous ne pouvez mettre à jour que vos propres messages !"
      });
    }
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};
```

Controller Supprimer un post

```
/* DELETE
const deletePost = async (req, res) => {
  const postId = req.params.id;
  const { userId, admin } = req.body;
  try {
    const post = await Post.findById(postId);
    if (admin || post.userId === userId) {
      if (post.image) {
        const filename = post.image.split("public/images/")[0];
        unlink(`public/images/${filename}`, () => {
          post.deleteOne();
        });
      }
      post.deleteOne();
      res.status(200).json(post);
    } else {
      res.status(403).json({
        error : "Vous ne pouvez mettre à jour que vos propres messages !"
      });
    }
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};
```

Frontend - Modifier + Supprimer

Composant Post

```
/** UPDATE POST
const handleUpdate = () => {
  if (auth.user.admin || auth.user._id === post.userId) {
    setUpdatePostModal(true);
  }
};

/** DELETE POST
const handleDelete = async () => {
  const deleteReq = {
    userId: auth.user._id,
    admin: auth.user.admin,
  };
  try {
    if (auth.user.admin || auth.user._id === post.userId) {
      const deleteRes = await fetch(`/api/posts/${post._id}`, {
        method: "DELETE",
        body: JSON.stringify(deleteReq),
        headers: {
          "Content-Type": "application/json",
          "Access-Control-Allow-Origin": "*",
          Authorization: `Bearer ${auth.token}`,
        },
      });
      const json = await deleteRes.json();
      dispatch({ type: "DELETE_POST", payload: json });
    }
  } catch (error) {
    console.log({ message: error.message });
  }
};
```

Condition de l'affichage des boutons Modifier / Supprimer

```
{auth.user.admin || auth.user._id === post.userId ? (
  <div>
    <img src={edit} alt="edit" onClick={handleUpdate} className="ico" />
    <PostUpdateModal updatePostModal={updatePostModal}
      setUpdatePostModal={setUpdatePostModal} data={post} />
    <img src={trash} alt="delete" onClick={handleDelete} className="ico" />
  </div>
) : null}
```



Frontend - Modifier

Composant - PostUpdate

```
const handleUpdate = async (e) => {  
  e.preventDefault();  
  if (updatePost.desc === "" && file === null) {  
    return  
  }  
  if (auth.user.admin || auth.user._id === data.userId) {  
    try {  
      if (file) {  
        const data = new FormData();  
        const fileName = Date.now() + file.name;  
        data.append("name", fileName);  
        data.append("file", file);  
        updatePost.image = fileName;  
  
        try {  
          await fetch("/api/upload", {  
            method: "POST",  
            headers: { Authorization: `Bearer ${auth.token}` },  
            body: data,  
          });  
        } catch (error) {  
          console.log({ message: error.message });  
        }  
      } else {  
        updatePost.image = null;  
      }  
      const response = await fetch(`/api/posts/${data._id}`, {  
        method: "PUT",  
        body: JSON.stringify(updatePost),  
        headers: {  
          "Content-Type": "application/json",  
          Authorization: `Bearer ${auth.token}`,  
        },  
      });  
      const json = await response.json();  
      setUpdatePostModal(false);  
      setFile(null);  
      dispatch({ type: "UPDATE_POST", payload: json });  
    } catch (error) {  
      console.log({ message: error.message });  
    }  
  }  
}
```

Modifier Post

Lorem ipsum dolor sit amet consectetur adipisicing elit. Consequatur doloremque p...



Selectionner une image



Backend + Frontend - Likes

Backend - Controller Like

```
/** LIKE
const likePost = async (req, res) => {
  const postId = req.params.id;
  const { userId } = req.body;
  try {
    const post = await Post.findById(postId);
    if (!post.likes.includes(userId)) {
      await post.updateOne({ $push: { likes: userId } });
      res.status(200).json("Post liked !");
    } else {
      await post.updateOne({ $pull: { likes: userId } });
      res.status(200).json("Post disliked !");
    }
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};
```

Frontend - Bouton (changement de l'icône)

```
<div onClick={handleLike} className="postLike">
  <img
    src={liked ? likeIcon : likeEmpty}
    alt="liked" /> {like} likes
</div>
```

Frontend - Composant Post

```
/** LIKE POST
const [like, setLike] = useState(post.likes.length);
const [liked, setLiked] = useState(false);

useEffect(() => {
  setLiked(post.likes.includes(auth.user._id));
}, [auth.user._id, post.likes]);

const handleLike = async () => {
  const likeReq = {
    userId: auth.user._id,
  };
  const likeRes = await fetch("/api/posts/" + post._id + "/like", {
    method: "POST",
    body: JSON.stringify(likeReq),
    headers: {
      "Content-Type": "application/json",
      Authorization: `Bearer ${auth.token}`,
    },
  });
  const json = await likeRes.json();
  dispatch({ type: "LIKE_POST", payload: json });

  setLike(liked ? like - 1 : like + 1);
  setLiked(!liked);
};
```

3 - Bilan & Axes d'amélioration

Cahier des charges

- Identité graphique respectés (Logo & Police)
- Produit totalement responsive
- Mis en place du backend, frontend et base de données
- Users & Posts stockés en database (mongoDB)
- Password utilisateurs sécurisés en database
- Utilisation d'un framework JavaScript (React)
- Connexion persistante + Déconnexion
- Postes affichés de façon antéchronologique
- Créer + Modifier + Supprimer un post
- Un poste peut contenir du texte et/ou une image
- L'utilisateur peut aimer les postes
- Un compte peut supprimer et modifier chaque postes
- Lancement du projet en suivant le readme
- Respecte les standards d'accessibilité (WCAG)

Axes d'amélioration

- Une page de profil pour chaque utilisateur avec plus d'informations (photo de profil, date de naissance, etc...)
- Une page ou un modal pour chaque poste pour avoir une meilleure visibilité
- Un système de commentaires pour plus d'interaction
- Un système de chat pour que différents utilisateurs puissent échanger
- Avoir le choix entre un light & dark mode
- Contrôler le poids max d'une image ajoutée
- Minifier et optimiser le code (meilleures performances)
- Renforcer la sécurité dans le backend (OWASP)