```markdown
# GROUPOMANIA

## Créer un réseau social d'entreprise

*Projet 7 de la formation Développeur web d'OpenClassRooms.*

Le projet consiste à construire un réseau social interne pour les employés de Groupomania.
Le but de cet outil est de faciliter les interactions entre les collègues.

## Compétences évaluées

#### La création de l'application fait appel aux compétences suivantes:

- Implémenter un stockage de données sécurisé en utilisant SQL
- Personnaliser le contenu envoyé à un client web
- Authentifier un utilisateur et maintenir sa session
- Gérer un stockage de données à l'aide de SQL

## Comment utiliser l'application?

#### Les installations et commandes suivantes sont nécessaires:

- Installer Node, Sass, Npm, mySql sur votre poste de travail
- Télécharger ou cloner le projet
- Vous devez créer 3 base de données Sql sur votre poste de travail (groupomania, groupomania_test et groupomania_production) en les laissant dans le dossier par défaut de vos bases de données MySql.
- Pour vous connecter à cette base de données, vous devez renseigner son nom dans le fichier backend/config/config.json
- Vous devez aussi mettre vos informations de connection mysql à cet endroit, dans mon cas sqluser/password.
- Une fois cela fait, vous pouvez créer les tables avec les modèles Sequelize que j'ai concus. La commande pour ca est:  npx sequelize db:migrate (depuis le terminal backend).
- Certaines colonnes des tables ne seront pas utilisées car par manque de temps je n'ai pas pu développer toutes les fonctionnalités que j'aurais voulues.
- Pour profiter pleinement de actions possibles dans l'application, il faut créer un administrateur (qui fait office de modérateur). Pour cela, inscrivez vous de manière classique sur l'application. Puis dans votre terminal mySql, et taper: UPDATE 'users' SET 'isAdmin' = '1' WHERE 'id'=1 (si c'est le premier utilisateur que vous créez)
- Aller dans le frontend (via un terminal) et faire `npm install`
- Faire `npm run serve` , le serveur se lance sur http://localhost:8080
- Aller dans le backend (via un terminal) et  faire `npm install`
- Toujours dans le dossier backend,  créer un fichier .env et insérer dedans (TOKEN='Random_secret_token')
- Faire `nodemon`. L'écoute se fait sur le port 3000 et vous pouvez vous rendre à présent sur l'adresse du localhost pour tester l'application.


#### Sécurisation de l'application et des données?

- bcrypt (hash et salage du mot de passe utilisateur)
- jsonwebtoken (sécurisation de l'authentification pour les requêtes)
- Base de données mySQL sur laquelle les informations sensibles ne sont pas stockées sans cryptage
- Création d'une API REST avec le framework Express
- Vérification du jeton et du propriétaire pour la suppression de contenu
- Jeton avec une durée d'expiration de 12h

## Environnement de développement

####

- Framework VUE JS (VUE CLI, VUE ROUTER, VUE X)
- Node / Sass / Npm
- Sequelize / Axios
- Nodemon
- MySql

#### Pour en savoir plus sur la sécurité informatique (développement/production/utilisation):

- [Framework Express: meilleures pratiques en production](https://expressjs.com/fr/advanced/best-practice-security.html)
- [GitHub Advisory Database](https://github.com/advisories)
- [OWASP Top 10 Web Application Security Risks](https://owasp.org/www-project-top-ten/)
```

```
.DS_Store
node_modules
/dist


/backend/images

# local env files
.env.local
.env.*.local

# Log files
npm-debug.log*
yarn-debug.log*
yarn-error.log*
pnpm-debug.log*

# Editor directories and files
.idea
.vscode
*.suo
*.ntvs*
*.njsproj
*.sln
*.sw?
```

~~~markdown
# groupomania

VUE 2
VUE CLI 4.5.15
DART SASS
VUE ROUTEUR
VUE X

## Project setup
```
npm install
```

### Compiles and hot-reloads for development
```
npm run serve
```

### Compiles and minifies for production
```
npm run build
```

### Lints and fixes files
```
npm run lint
```

### Customize configuration
See [Configuration Reference](https://cli.vuejs.org/config/).
~~~

```
{
  "name": "groupomania",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "lint": "vue-cli-service lint"
  },
  "dependencies": {
    "axios": "^0.24.0",
    "core-js": "^3.6.5",
    "vue": "^2.6.11",
    "vue-router": "^3.2.0",
    "vuex": "^3.4.0"
  },
  "devDependencies": {
    "@vue/cli-plugin-babel": "~4.5.0",
    "@vue/cli-plugin-eslint": "~4.5.0",
    "@vue/cli-plugin-router": "~4.5.0",
    "@vue/cli-plugin-vuex": "~4.5.0",
    "@vue/cli-service": "~4.5.0",
    "babel-eslint": "^10.1.0",
    "eslint": "^6.7.2",
    "eslint-plugin-vue": "^6.2.2",
    "sass": "^1.26.5",
    "sass-loader": "^8.0.2",
    "vue-template-compiler": "^2.6.11"
  }
}
```

```html
<!DOCTYPE html>
<html lang="">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
    <link rel="icon" href="<%= BASE_URL %>favicon.ico">
    <title>Groupomania, réseau social</title>
  </head>
  <body>
    <noscript>
      <strong>We're sorry but <%= htmlWebpackPlugin.options.title %> doesn't work properly without JavaScript enabled. Please enable it to continue.</strong>
    </noscript>
    <div id="app"></div>
    <!-- built files will be auto injected -->
  </body>
</html>
```

```scss
@import "variables";
@import "mixins";

h1 {
  font-style: italic;
  font-weight: bold;
  text-align: left;
  padding: 0 1rem;
}

h2 {
  text-align: left;
  padding-left: 1rem;
  text-decoration: underline;
}

body {
  margin: 0;
}

.footer {
  background-color: #f2f2f2;
  padding: 1rem;
  margin-top: 5rem;
  display: flex;
  justify-content: space-around;
  flex-wrap: wrap;
  &__content {
    padding: 1rem;
  }
}

.data {
  display: flex;
  flex-direction: column;
  align-items: center;

  &__button {
    background-color: $soft-grey;
    max-width: 150px;
    padding: 1rem;
    margin: 1rem;
    color: black;
    @include shadow;
    align-self: center;
  }

  &__container {
    border: 2px solid;
    border-radius: 10px;
    margin: 1rem 0;
    padding: 1rem;
    width: 80%;

    @include tablet {
      width: 50%;
    }
  }
}
```

```scss

@mixin flex-column {
  display: flex;
  flex-wrap: wrap;
  flex-direction: column;
}

@mixin desktop {
  @media screen and (min-width: 1020px) {
    @content;
  }
}

@mixin tablet {
  @media screen and (min-width: 420px) {
    @content;
  }
}

@mixin soft-typo {
  font-size: 1rem;
  background-color: $soft-grey;
  padding: 1rem;
  font-size: 1rem;
  text-shadow: 1px 0 1px rgba(0, 0, 0, 0.3);
}

@mixin bold-typo {
  font-size: 1rem;
  font-weight: 800;
  text-shadow: 1px 0 1px rgba(0, 0, 0, 0.3);
  background-color: $mid-grey;
  padding: 1rem;
}

@mixin shadow {
  border-radius: 1rem;
  box-shadow: 0.7rem 0.7rem 1rem rgb(200, 200, 200);
  &:hover {
    box-shadow: 0.7rem 0.7rem 1rem rgb(130, 130, 130);
    transition: all 500ms;
  }
}
```

Fichier : /home/thomas/groupomania/frontend/styles/_variables.scss Page 1 sur 1

```scss
$rouge: #ec4e48;
$bleu: #838bcc;
$vert: #3cd6b0;

$soft-grey: #f7f7f7;
$mid-grey: #ededed;
$hard-grey: #353535;

$fa-font-path: "./webfonts";
```

```javascript
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'

Vue.config.productionTip = false

new Vue({
  router,
  store,
  render: h => h(App)
}).$mount('#app')
```

```
<template>
  <div id="app">
    <router-view/>
  </div>
</template>

<style lang="scss">
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
}
</style>
```

```
<template>
  <div>
    <Header />
    <Navbar />
    <h2>Informations du compte</h2>
    <div class="data">
      <div class="data__container">
        <section>
          <div class="account__Message">
            <p>
              Prénom: <span class="precision-typo">{{ firstName }}</span>
            </p>
          </div>
          <div class="account__Message">
            <p>
              Nom: <span class="precision-typo"> {{ lastName }} </span>
            </p>
          </div>
          <div class="account__Message">
            <p>
              Email: <span class="precision-typo"> {{ email }} </span>
            </p>
          </div>
          <div class="account__Message">
            <p>
              Identifiant utilisateur:
              <span class="precision-typo"> {{ id }} </span>
            </p>
          </div>
          <div v-if="isAdmin === 'false'" class="account__Message">
            <p>Vous n'ẽ́tes pas administrateur du site.</p>
            <p>
              Vous pouvez le contacter par mail à l'adresse
              <span class="precision-typo">admin@admin.fr</span>
            </p>
          </div>
          <button @click="deleteMyAccount()" class="data__button">
            Supprimer mon compte
          </button>
        </section>
      </div>
    </div>
    <Footer />
  </div>
</template>

<script>
import Header from "../components/Header.vue";
import Navbar from "../components/Navbar.vue";
import Footer from "@/components/Footer.vue";

import axios from "axios";

export default {
  name: "Account",
  components: {
    Header,
    Navbar,
    Footer,
  },
  data() {
    return {
      isAdmin: localStorage.getItem("isAdmin"),
      firstName: "",
      lastName: "",
      email: "",
      id: localStorage.getItem("userId"),
    };
  },

  created() {
    this.getAccount();
  },
  methods: {
    getAccount() {
      const id = localStorage.getItem("userId");

      axios
        .get("http://localhost:3000/api/user/" + id, {
          headers: {
            Authorization: "Bearer " + localStorage.getItem("token"),
          },
        })
        .then((response) => {
          this.email = response.data.email;
          this.firstName = response.data.firstName;
          this.lastName = response.data.lastName;
        })

        .catch((error) => {
          console.log(error);
        });
    },

    deleteMyAccount() {
      const userId = localStorage.getItem("userId");

      axios
        .delete("http://localhost:3000/api/user/" + userId, {
          headers: {
            "Content-Type": "application/json",
            Authorization: "Bearer " + localStorage.getItem("token"),
          },
        })
        .then((res) => {
          console.log("compte supprimé");
          localStorage.clear();
          alert("Compte supprimé");
          this.$router.push("/");

          return res.status(200).json({ message: "Compte supprimé" });
        })
        .catch((error) => {
          console.log(error);
        });
    },
  },
};
</script>

<style scoped lang="scss">
@import "./styles/main.scss";

.account {
  background-color: #f2f2f2;
  border: 3px solid;
  border-radius: 10px;
  margin: 1rem 0;
  padding: 1rem;
  width: 80%;

  @include tablet {
    width: 50%;
  }

  &__Message {
    background-color: #f2f2f2;
    border: 1px solid;
    margin: 1rem 0;
  }


}
</style>
```

```
<template>
  <div>
    <Header />
    <Navbar />
    <h2>Les derniers utilisateurs inscrits</h2>
    <div class="data">
      <div class="data__container" v-for="user in accounts" :key="user.id">
        <div class="accounts--info--precision">
          Prénom:
          <span class="precision-typo">{{ user.firstName }}</span>
        </div>
        <div class="accounts--info--precision">
          Nom:
          <span class="precision-typo">{{ user.lastName }}</span>
        </div>
        <div class="accounts--info--precision">
          Identifiant:
          <span class="precision-typo">{{ user.id }} </span>
        </div>
        <div class="accounts--title">
          Inscrit depuis le:
          <span class="precision-typo">{{ user.createdAt }} </span>
        </div>
        <div class="accounts--content">
          Email:

          <span class="precision-typo">{{ user.email }} </span>
        </div>
        <button
          v-if="isAdmin === 'true'"
          @click="deleteAccount(user)"
          class="data__button"
        >
          Supprimer le compte
        </button>
        <!-- <button
              @click="getOneAccount(user)"
              class="accounts--info--deletePost"
            >
              Voir le compte
            </button> -->
      </div>
    </div>
  </div>
</template>

<script>
import axios from "axios";
import Header from "@/components/Header.vue";
import Navbar from "@/components/Navbar.vue";

export default {
  name: "AllAccounts",
  components: {
    Header,
    Navbar,
  },
  data() {
    return {
      accounts: [],
      user: "",
      isAdmin: localStorage.getItem("isAdmin"),
    };
  },

  created() {
    this.getAllAccounts();
  },

  methods: {
    getAllAccounts() {
      axios
        .get("http://localhost:3000/api/user/", {
          headers: {
            Authorization: "Bearer " + localStorage.getItem("token"),
          },
        })
        .then((res) => {
          console.log("ALLACCOUNTS consultés");

          this.accounts = res.data;
        })
        .catch((error) => console.log(error));
    },

    // getOneAccount(user) {
    //   axios
    //     .get("http://localhost:3000/api/user/" + user.id, {
    //       headers: {
    //         Authorization: "Bearer " + localStorage.getItem("token"),
    //       },
    //     })
    //     .then((response) => {
    //       this.email = response.data.email;
    //       this.firstName = response.data.firstName;
    //       this.lastName = response.data.lastName;
    //       this.userId = response.data.id;
    //       this.$router.push('/user/' + user.id);
    //     })

    //     .catch((error) => {
    //       console.log(error);
    //     });
    // },

    deleteAccount(user) {
      axios
        .delete("http://localhost:3000/api/user/" + user.id, {
          headers: {
            "Content-Type": "application/json",
            Authorization: "Bearer " + localStorage.getItem("token"),
          },
        })
        .then((res) => {
          console.log("Compte supprimé");
          if (this.$route.path === "/allAccounts") {
            window.location.reload();
          } else this.$router.push("/allAccounts");
          res.status(200).json({ message: "Post supprimé" });
        })
        .catch((error) => {
          console.log(error);
        });
    },
  },
};
</script>

<style scoped lang="scss">
@import "./styles/main.scss";

.accounts {

  &--info {
    @include flex-column;
    font-weight: bold;
    font-size: 1rem;
    padding: 1rem;
    color: $rouge;

  }
  &--title {
    @include bold-typo;
  }
  &--content {
    @include soft-typo;
  }
}
</style>
```

```
<template>
  <div>
    <Header />
    <Navbar />
    <div class="notOnMobile">
      <div class="notOnMobile__title"><h2>Poster votre article</h2></div>
      <div class="notOnMobile__createPost"><CreatePost /></div>
    </div>
    <h2>Les derniers articles publiés</h2>

    <section class="data">
      <div class="data__container" v-for="post in posts" :key="post.id">
        <div class="posts--info">
          <div class="posts--info--precision">
            Article numéro:
            <span class="precision-typo">{{ post.id }} </span>
          </div>
          <div class="posts--info--precision">
            Posté par:
            <span class="precision-typo">{{ post.userId }}</span>

            Le:
            <span class="precision-typo">{{ post.createdAt }} </span>
          </div>
        </div>
        <div class="posts--title">{{ post.title }}</div>
        <div class="posts--content">{{ post.content }}</div>
        <button
          v-if="isAdmin === 'true' || post.userId == userId"
          @click="deletePost(post)"
          class="data__button"
        >
          Supprimer
        </button>
        <!-- @click="showComment = !showComment"
                <button

                  @click="getAllComments(post)"
                  class="show-comment"
                >
                  Voir les commentaires
                </button>
                <div v-show="showComment" class="comment">
                  Affichage?
                </div> -->
      </div>
    </section>
  </div>
</template>

<script>
import Header from "@/components/Header.vue";
import Navbar from "@/components/Navbar.vue";
import CreatePost from "../components/CreatePost.vue";

import axios from "axios";

export default {
  name: "AllPosts",
  components: {
    Header,
    CreatePost,
    Navbar,
  },
  data() {
    return {
      posts: [],
      post: "",
      isAdmin: localStorage.getItem("isAdmin"),
      id: "",
      userId: localStorage.getItem("userId"),
      //showComment: false,
      //comments: [],
    };
  },

  created() {
    this.getAllPosts();
    //this.getAllComments();
  },

  methods: {
    // getAllComments(post) {
    //   const userId = localStorage.getItem("userId");
    //   console.log("userId", userId);
    //   console.log("post", post.id);

    //   axios
    //     .get("http://localhost:3000/api/comment/", +post.id, {
    //       headers: {
    //         Authorization: "Bearer " + localStorage.getItem("token"),
    //       },
    //       body: {
    //         userId: userId,
    //       },
    //     })
    //     .then((res) => {
    //       console.log("COMMENTS consultés");
    //       this.comments = res.data;
    //     })

    //     .catch((error) => {
    //       console.log(error);
    //     });
    // },

    getAllPosts() {
      axios
        .get("http://localhost:3000/api/post/", {
          headers: {
            Authorization: "Bearer " + localStorage.getItem("token"),
          },
        })
        .then((res) => {
          console.log("ALLPOSTS consultés");
          this.posts = res.data;
        })

        .catch((error) => {
          console.log(error);
        });
    },

    deletePost(post) {
      axios
        .delete("http://localhost:3000/api/post/" + post.id, {
          headers: {
            "Content-Type": "application/json",
            Authorization: "Bearer " + localStorage.getItem("token"),
          },
        })
        .then((res) => {
          console.log("Post supprimé");
          if (this.$route.path === "/allPosts") {
            window.location.reload();
          } else this.$router.push("/allPosts");
          res.status(200).json({ message: "Post supprimé" });
        })
        .catch((error) => {
          console.log(error);
        });
    },
  },
};
</script>

<style lang="scss">
@import "./styles/main.scss";

.notOnMobile {
  display: none;

  @include desktop {
    display: flex;
    flex-direction: column;

    &__createPost {
      display: flex;
      justify-content: center;
    }
  }
}

.posts {
  &--info {
    @include flex-column;
    font-weight: bold;
    font-size: 1rem;
    padding: 1rem;
    color: $rouge;
  }
  &--title {
    @include bold-typo;
  }
  &--content {
    @include soft-typo;
  }
}

.precision-typo {
  @include bold-typo;
  padding: 0;
  color: black;
}
</style>
```

```
<template>
  <div>
    <Header />
    <div class="home">
      <div class="welcomeAnim">
        <img
          class="home__bigLogo"
          alt="logo de Groupomania"
          src="../assets/icon.svg"
        />
        <img
          class="home__smallLogo"
          alt="logo de Groupomania"
          src="../assets/icon-left-font.svg"
        />
      </div>
    </div>
    <section>
      <div class="welcomeMessage">
        Groupomania vous souhaite une Bonne Année 2022! Nous avons pour cette
        occasion le plaisir de vous proposer un réseau social afin que chaque
        collaborateur puisse interagir avec les autres, partager, proposer...
        N'attendez plus pour vous inscrire et commencer à en profiter!
      </div>
    </section>
    <Footer />
  </div>
</template>

<script>
// @ is an alias to /src
import Header from "@/components/Header.vue";
import Footer from "@/components/Footer.vue";

export default {
  name: "Home",
  components: {
    Header,
    Footer,
  },
};
</script>

<style scoped lang="scss">
@import "./styles/main.scss";

.welcomeAnim {
  @include desktop {
    display: flex;
    justify-content: space-around;
  }
}

.welcomeMessage {
  font-size: 1rem;
  font-weight: bold;
  margin: 0 2rem 2rem 2rem;

  @include tablet {
    font-size: 1.5rem;
    margin: 0 5rem 5rem 2rem;
  }
}

.home {
  &__bigLogo {
    display: none;
    @include desktop {
      max-width: 400px;
      width: 50%;
      margin: 5rem;
      display: flex;
      animation: spinner-rotate-logo 3000ms linear infinite;
    }
  }
  &__smallLogo {
    width: 100%;
    max-height: 200px;
    animation: spinner-opacity2-logo 4000ms linear infinite;

    @include tablet {
      animation: spinner-opacity-logo 3000ms linear infinite;
      max-width: 400px;
      width: 25%;
      align-self: center;
    }
  }
}

@keyframes spinner-rotate-logo {
  0% {
    transform: rotate(0deg);
  }

  100% {
    transform: rotate(360deg);
  }
}

@keyframes spinner-opacity-logo {
  0% {
    opacity: 0;
  }

  50% {
    opacity: 1;
  }

  100% {
  }
}
@keyframes spinner-opacity2-logo {
  0% {
    opacity: 0;
  }

  100% {
    opacity: 1;
  }
}
</style>
```

```
<template>
  <div>
    <Header />
    <div class="login">
      <form @submit.prevent="login" class="login__form">
        <div class="login__form__input">
          <label for="email" class="login__form__input__label">Email</label>
          <input type="email" v-model="email" id="email" name="email" />
        </div>

        <div class="login__form__input">
          <label for="password" class="login__form__input__label"
            >Mot de passe</label
          >
          <input
            type="password"
            v-model="password"
            id="password"
            name="password"
          />
        </div>

        <button class="login__form__button">Valider</button>
      </form>
    </div>
    <Footer />
  </div>
</template>

<script>
// @ is an alias to /src
import Header from "@/components/Header.vue";
import Footer from "@/components/Footer.vue";
import axios from "axios";

export default {
  name: "Login",
  components: {
    Header,
    Footer,
  },
  data() {
    return {
      email: "",
      password: "",
    };
  },
  methods: {
    login() {
      axios
        .post("http://localhost:3000/api/user/login", {
          email: this.email,
          password: this.password,
        })
        .then((res) => {
          localStorage.setItem("userId", res.data.userId);
          localStorage.setItem("token", res.data.token);
          localStorage.setItem("isAdmin", res.data.isAdmin); // faire une méthode isAdminByToken ?
          this.$router.push("/allPosts");
        })
        .catch((error) => {
          console.log(error);
        });
    },
  },
};
</script>

<style scoped lang="scss">
@import "./styles/main.scss";

.login {
  display: flex;
  justify-content: center;
  align-items: center;
  margin: 3rem 0 15rem 0; // voir pour une position absolu du footer? également sur toutes les pages.

  @include desktop {
    margin-top: 10rem;
  }

  &__form {
    @include flex-column;
    background-color: #f2f2f2;
    align-items: center;
    width: 85%;
    border: 2px solid #3f3d56;
    border-radius: 5px;
    padding: 1.5rem 0;

    @include desktop {
      width: 50%;
      padding: 3rem 0;
      border: 3px solid #3f3d56;
    }

    &__input {
      display: flex;
      flex-direction: column;
      margin-bottom: 2rem;
      width: 70%;
      &__label {
        text-align: start;
        font-weight: bold;
        margin-bottom: 0.5rem;
      }
    }
    &__button {
      border: 1px solid $hard-grey;
      border-radius: 5px;
      font-weight: bold;
      padding: 0.9rem;
      &:hover,
      &:focus {
        border: 1px solid $hard-grey;
        background-color: $rouge;
        cursor: pointer;
      }
    }
  }
}
</style>
```

```
<template>
  <div>
    <Header />
    <Navbar />
    <h2>Poster votre article</h2>
    <div class="data">
      <CreatePost />
    </div>
    <Footer />
  </div>
</template>

<script>
import Header from "../components/Header.vue";
import Navbar from "../components/Navbar.vue";
import CreatePost from "../components/CreatePost.vue";
import Footer from "@/components/Footer.vue";


export default {
  name: "Post",
  components: {
    Header,
    Navbar,
    CreatePost,
    Footer
  },
};
</script>

<style lang="scss">
@import "./styles/main.scss";


</style>
```

```
<template>
  <div>
    <Header />
    <div class="signup">
      <form @submit.prevent="signup" class="signup__form">
        <div class="signup__form__input">
          <label for="firstName" class="signup__form__input__label"
            >Prénom</label
          >
          <input
            type="text"
            v-model="firstName"
            id="firstName"
            name="firstName"
          />
        </div>

        <div class="signup__form__input">
          <label for="lastName" class="signup__form__input__label">Nom</label>
          <input type="text" v-model="lastName" id="lastName" name="lastName" />
        </div>

        <div class="signup__form__input">
          <label for="email" class="signup__form__input__label">Email</label>
          <input type="email" v-model="email" id="email" name="email" />
        </div>

        <div class="signup__form__input">
          <label for="password" class="signup__form__input__label"
            >Mot de passe</label
          >
          <input
            type="password"
            v-model="password"
            id="password"
            name="password"
          />
        </div>

        <button class="signup__form__button">Valider</button>
      </form>
    </div>
    <Footer />
  </div>
</template>

<script>
// @ is an alias to /src
import Header from "@/components/Header.vue";
import Footer from "@/components/Footer.vue";
import axios from "axios";

export default {
  name: "Signup",
  components: {
    Header,
    Footer,
  },
  data() {
    return {
      firstName: "",
      lastName: "",
      email: "",
      password: "",
    };
  },
  methods: {
    signup() {
      axios
        .post("http://localhost:3000/api/user/signup", {
          firstName: this.firstName,
          lastName: this.lastName,
          email: this.email,
          password: this.password,
        })
        .then(() => {
          this.$router.push("/login");
        })
        .catch((error) => {
          console.log(error);
        });
    },
  },
};
</script>

<style scoped lang="scss">
@import "./styles/main.scss";

.signup {
  display: flex;
  justify-content: center;
  align-items: center;
  margin: 3rem 0 5rem 0;

  @include desktop {
    margin-top: 10rem;
  }

  &__form {
    @include flex-column;
    background-color: #f2f2f2;
    align-items: center;
    width: 85%;
    border: 2px solid #3f3d56;
    border-radius: 5px;
    padding: 1.5rem 0;

    @include desktop {
      width: 50%;
      padding: 3rem 0;
      border: 3px solid #3f3d56;
    }

    &__input {
      display: flex;
      flex-direction: column;
      margin-bottom: 2rem;
      width: 70%;
      &__label {
        text-align: start;
        font-weight: bold;
        margin-bottom: 0.5rem;
      }
    }
    &__button {
      border: 1px solid $hard-grey;
      border-radius: 5px;
      font-weight: bold;
      padding: 0.9rem;
      &:hover,
      &:focus {
        border: 1px solid $hard-grey;
        background-color: $rouge;
        cursor: pointer;
      }
    }
  }
}
</style>
```

```
import Vue from 'vue'
import Vuex from 'vuex'

Vue.use(Vuex)

export default new Vuex.Store({
  state: {
    status: "Non connecté",
    organismeDeFormation: "OpenClassRooms",

  },
  mutations: {
  },
  actions: {
  },
  getters: {
                copyright: (state) => {
                        const currentYear = new Date().getFullYear()

                        return `Copyright ${state.organismeDeFormation} ${currentYear}`
                },


        },
  modules: {
  }
})
```

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Home from '../views/Home.vue'

Vue.use(VueRouter)

const routes = [
  {
    path: '/',
    name: 'Home',
    component: Home
  },
  {
    path: '/signup',
    name: 'Signup',
    component: () => import('../views/Signup.vue')
  },
  {
    path: '/login',
    name: 'Login',
    component: () => import('../views/Login.vue')
  },
  {
    path: '/user/:id',
    name: 'Account',
    component: () => import('../views/Account.vue')

  },
  {
    path: '/allAccounts',
    name: 'AllAccounts',
    component: () => import('../views/AllAccounts.vue')
  },
  {
    path: '/post',
    name: 'Post',
    component: () => import('../views/Post.vue')
  },
  {
    path: '/allPosts',
    name: 'AllPosts',
    component: () => import('../views/AllPosts.vue')
  }
]

const router = new VueRouter({
  routes
})

export default router
```

```
<template>
  <div class="data__container">
    <form @submit.prevent="createPost" class="createpost__items">
      <label> Titre: <input v-model="title" /> </label>
      <br /><br />
      <label> Contenu: </label>
      <br /><br />
      <textarea
        v-model="content"
        placeholder="Que souhaitez vous partager?"
        rows="5"
        cols="5"
      ></textarea>

      <input type="file" ref="file"  @change="uploadPhoto" />
      <div v-if="imageURL.length > 0">
        <img :src="imageURL" height="300px" width="300px" />
      </div>
      <button class="createpost data__button" type="submit">Publier</button>
    </form>
  </div>
</template>

<script>
import axios from "axios";

export default {
  name: "CreatePost",
  components: {},
  data() {
    return {
      title: this.title,
      content: this.content,
      userId: "",
      image: "",
      imageURL: "",
    };
  },

  methods: {
    uploadPhoto(event) {
      this.image = event.target.files[0];
      console.log("image", this.image);
      this.imageURL = URL.createObjectURL(this.image);
      console.log("imageURL", this.imageURL);
    },

    createPost() {
      const formData = new FormData();

      formData.append("title", this.title);
      formData.append("content", this.content);
      formData.append("attachment", this.image);


      axios
        .post("http://localhost:3000/api/post", formData, {
          headers: {
            "Content-Type": "multipart/form-data",
            Authorization: "Bearer " + localStorage.getItem("token"),
          },
        })
        .then((res) => {
          console.log("POST effectué");
          if (this.$route.path === "/allPosts") {
            window.location.reload();
          } else this.$router.push("/allPosts");
          res.status(200).json({ message: "Post effectué" });
        })
        .catch((error) => {
          console.log(error);
        });
    },
  },
};
</script>

<style lang="scss" scoped>
@import "./styles/main.scss";

.createpost {
  &__items {
    @include flex-column;
  }
}
</style>
```

```
<template>
  <footer class="footer">
    <div class="footer__content">A propos</div>
    <div class="footer__content">{{ copyright }}</div>
    <div class="footer__content">Assistance</div>
  </footer>
</template>

<script>
import { mapGetters, mapState } from "vuex";

export default {
  name: "Footer",

  computed: {
    ...mapGetters({
      copyright: "copyright",
    }),
    ...mapState({
      organismeDeFormation: "OpenClassRooms",
    }),
  },
}
</script>

<style lang="scss">
@import "./styles/main.scss";
</style>
```

```
<template>
  <div class="navbar">
    <router-link to="/">
      <img
        class="navbar__smallLogo"
        alt="logo de Groupomania"
        src="../assets/icon-left-font.png"
      />
    </router-link>

    <img
      class="navbar__smallLogo"
      alt="logo de Groupomania"
      src="../assets/icon.png"
    />
    <div class="navbar__container">
      <router-link v-if="!id" class="navbar__container--link" to="/signup"
        >S'inscrire</router-link
      >

      <router-link v-if="!id" class="navbar__container--link" to="/login"
        >Se connecter</router-link
      >
      <a
        v-if="id"
        class="navbar__container--link"
        :href="'http://localhost:8080/#/user/' + id"
        >Mon compte</a
      >
      <button
        v-if="id"
        @click="showModal = !showModal"
        class="navbar__container--link"
      >
        Déconnexion
      </button>
      <LogoutModale v-show="showModal" class="modal" />
    </div>
  </div>
</template>

<script>
import LogoutModale from "@/components/LogoutModale.vue";

export default {
  name: "Navbar",
  components: {
    LogoutModale,
  },

  data: () => ({
    id: localStorage.getItem("userId"),
    showModal: false,
  }),
};
</script>

<style lang="scss">
@import "./styles/main.scss";

.navbar {
  display: flex;
  align-items: center;
  justify-content: space-between;
  flex-wrap: wrap;

  &__container {
    display: flex;
    flex-direction: column;

    @include tablet {
      flex-direction: row;
    }

    &--link {
      color: black;
      text-decoration: none;
      font-weight: bold;
      padding: 0.5rem;

      @include tablet {
        margin: 1rem 0;
      }

      @include desktop {
        margin: 1rem;
        padding: 0.9rem;
      }

      &:hover,
      &:focus {
        cursor: pointer;
        border: 1px;
        background-color: $rouge;
        border-radius: 5px;
      }
    }
  }
  &__smallLogo {
    max-height: 100px;
  }
}
</style>
```

```
<template>
  <div>
    <div>Il est {{ heureCourante() }}</div>
  </div>
</template>

<script>
export default {
  name: "HeureCourante",

  created() {
    setInterval(() => this.$forceUpdate(), 10000); //tester avec le statut connecté
  },
  methods: {
    heureCourante() {
      let now = new Date();
      let heure = ("0" + now.getHours()).slice(-2);
      let minute = ("0" + now.getMinutes()).slice(-2);
      return `${heure}h${minute}`;
    },
  },
};
</script>

<style lang="scss">
@import "./styles/main.scss";


</style>
```

```
<template>
  <div id="myModal" class="modal">
    <div class="modal-content">
      <div class="modalLineOne">
        <div>Etes-vous certain de vouloir vous déconnecter?</div>
        <span @click="closeModal" class="close">&times;</span>
      </div>
      <div class="modalLineTwo">
        <button @click="disconnect">OUI</button>
        <button @click="closeModal">NON</button>
      </div>
    </div>
  </div>
</template>

<script>
export default {
  name: "LogoutModale",

  methods: {
    disconnect() {
      if (this.$route.path === "/") {
        localStorage.clear();
        window.location.reload();
      } else localStorage.clear();
      this.$router.push("/");
    },

    closeModal() {
      var modal = document.getElementById("myModal");
      modal.style.display = "none";
    },
  },
};
</script>

<style scoped lang="scss">
@import "./styles/main.scss";


.modal {

  position: fixed;
  z-index: 1;
  left: 0;
  top: 0;
  width: 100%;
  height: 100%;
  overflow: auto;
  background-color: rgb(0, 0, 0);
  background-color: rgba(0, 0, 0, 0.4);
}


.modal-content {
  background-color: #fefefe;
  margin: 15% auto;
  padding: 20px;
  border: 1px solid #888;
  width: 80%;
  color: black;
  font-weight: bold;
}


.close {
  color: #aaa;
  float: right;
  font-size: 28px;
  font-weight: bold;
}

.close:hover,
.close:focus {
  color: black;
  text-decoration: none;
  cursor: pointer;
}

.modalLineOne {
  display: flex;
  justify-content: space-between;
}

.modalLineOne div {
  align-self: center;
}

.modalLineTwo {
  display: flex;
  justify-content: space-around;
}
</style>
```

```
<template>
  <div class="options">
    <div class="options__container--clock">
      <HeureCourante />
    </div>
    <a
      class="options__container--link"
      :href="'http://localhost:8080/#/user/' + id"
      >Gérer mon compte</a
    >
    <router-link class="options__container--link" to="/post"
      >Poster un article</router-link
    >
    <router-link class="options__container--link" to="/allPosts"
      >Tous les articles</router-link
    >
    <router-link class="options__container--link" to="/allAccounts"
      >Tous les utilisateurs</router-link
    >
  </div>
</template>

<script>
import HeureCourante from "@/components/HeureCourante.vue";

export default {
  name: "Navbar",
  components: {
    HeureCourante,
  },

  data: () => ({
    id: localStorage.getItem("userId"),
  }),
};
</script>

<style scoped lang="scss">
@import "./styles/main.scss";

.options {
  background-color: #f2f2f2;
  display: flex;
  justify-content: center;
  flex-wrap: wrap;
  width: 100%;

  &__container {

    &--link {
      color: black;
      text-decoration: none;
      margin: 0.3rem;
      font-weight: bold;
      padding: 0.5rem;

       @include tablet {
        margin: 1rem;
         padding: 0.9rem;
       }

    }
    &--clock {
      color: black;
      margin: 0.3rem;
      padding: 0.5rem;
      @include tablet {
      margin: 1rem;
      padding: 0.9rem;
      }
    }
  }
}

p {
  margin: 0.5rem;
}
</style>
```

```
const express = require('express');
const path = require('path');

require('dotenv').config();

const app = express();

const userRoutes = require('./routes/user');
const postRoutes = require('./routes/post');
const commentRoutes = require('./routes/comment');

app.use((req, res, next) => {
    res.setHeader('Access-Control-Allow-Origin', '*');
    res.setHeader('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content, Accept, Content-Type, Authorization');
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, PATCH, OPTIONS');
    next();
});

app.use(express.json()); // Permet de ne plus utiliser body-parser qui est inclut à présent dans Express
app.use(express.urlencoded({ extended: true }));

app.use('/images', express.static(path.join(__dirname, 'images')));

app.use('/api/user', userRoutes);
app.use('/api/post', postRoutes);
app.use('/api/comment', commentRoutes);


module.exports = app;
```

```
const http = require('http');
const app = require('./app');

const normalizePort = val => {
    const port = parseInt(val, 10);

    if (isNaN(port)) {
        return val;
    }
    if (port >= 0) {
        return port;
    }
    return false;
};
const port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

const errorHandler = error => {
    if (error.syscall !== 'listen') {
        throw error;
    }
    const address = server.address();
    const bind = typeof address === 'string' ? 'pipe ' + address : 'port: ' + port;
    switch (error.code) {
        case 'EACCES':
            console.error(bind + ' requires elevated privileges.');
            process.exit(1);
            break;
        case 'EADDRINUSE':
            console.error(bind + ' is already in use.');
            process.exit(1);
            break;
        default:
            throw error;
    }
};

const server = http.createServer(app);

server.on('error', errorHandler);
server.on('listening', () => {
    const address = server.address();
    const bind = typeof address === 'string' ? 'pipe ' + address : 'port ' + port;
    console.log('Listening on ' + bind);
});

server.listen(port);
```

```json
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "dotenv": "^10.0.0",
    "express": "^4.17.2",
    "jsonwebtoken": "^8.5.1",
    "multer": "^1.4.4",
    "mysql2": "^2.3.3",
    "sequelize": "^7.0.0-alpha.3"
  },
  "devDependencies": {
    "nodemon": "^2.0.15",
    "sequelize-cli": "^6.3.0"
  }
}
```

```
const express = require('express');
const router = express.Router();
const userCtrl = require('../controllers/user');

const getAuth = require('../middleware/getAuth');
const putDeleteAuth = require('../middleware/putDeleteAuth');

const multer = require('../middleware/multer-config');

router.post('/signup', userCtrl.signup);
router.post('/login', userCtrl.login);

router.get('/', getAuth, userCtrl.getAllAccounts);
router.get('/:id', getAuth, userCtrl.getAccount);

router.delete('/:id', getAuth, putDeleteAuth, userCtrl.deleteAccount);

//router.get('/:id', getAuth, userCtrl.getOneAccount); //facultatif
//router.put('/:id', userCtrl.modifyAccount); //facultatif


module.exports = router;
```

```
const express = require('express');
const router = express.Router();
const postCtrl = require('../controllers/post');

const getAuth = require('../middleware/getAuth');
const postAuth = require('../middleware/postAuth');
const putDeleteAuth = require('../middleware/putDeleteAuth');
const multer = require('../middleware/multer-config');

router.post('/', postAuth, multer, postCtrl.createPost);

router.get('/', getAuth, postCtrl.getAllPosts);
router.get('/:id', getAuth, postCtrl.getOnePost);

router.delete('/:id', getAuth, putDeleteAuth, postCtrl.deletePost);

//router.put('/:id', postCtrl.modifyPost); //facultatif



module.exports = router;
```

```
const express = require('express');
const router = express.Router();
const commentCtrl = require('../controllers/comment');

const getAuth = require('../middleware/getAuth');
const putDeleteAuth = require('../middleware/putDeleteAuth');

//router.post('/:postId', getAuth, commentCtrl.createComment); // en cours

//router.get('/:postId', getAuth, commentCtrl.getAllComments); // en cours

//router.delete('/:commentId', putDeleteAuth, commentCtrl.deleteComment); // en cours


module.exports = router;
```

```
'use strict';
const {
  Model
} = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class user extends Model {
    /**
     * Helper method for defining associations.
     * This method is not a part of Sequelize lifecycle.
     * The `models/index` file will call this method automatically.
     */
    static associate(models) {
      // define association here

      models.user.hasMany(models.post);
      models.user.hasMany(models.comment);
    }
  };
  user.init({
    firstName: DataTypes.STRING,
    lastName: DataTypes.STRING,
    email: DataTypes.STRING,
    password: DataTypes.STRING,
    isAdmin: DataTypes.BOOLEAN,
    isActive: DataTypes.BOOLEAN,
    keepConnected: DataTypes.BOOLEAN
  }, {
    sequelize,
    modelName: 'user',
  });
  return user;
};
```

```
'use strict';
const {
  Model
} = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class post extends Model {
    /**
     * Helper method for defining associations.
     * This method is not a part of Sequelize lifecycle.
     * The `models/index` file will call this method automatically.
     */
    static associate(models) {
      // define association here
      models.post.belongsTo(models.user, {
        foreignKey: 'userId'
      })
      models.post.hasMany(models.comment);

    }
  };
  post.init({
    userId: DataTypes.INTEGER,
    title: DataTypes.STRING,
    content: DataTypes.STRING,
    attachment: DataTypes.STRING
  }, {
    sequelize,
    modelName: 'post',
  });
  return post;
};
```

```javascript
'use strict';

const fs = require('fs');
const path = require('path');
const Sequelize = require('sequelize');
const basename = path.basename(__filename);
const env = process.env.NODE_ENV || 'development';
const config = require(__dirname + '/../config/config.json')[env];
const db = {};

let sequelize;
if (config.use_env_variable) {
  sequelize = new Sequelize(process.env[config.use_env_variable], config);
} else {
  sequelize = new Sequelize(config.database, config.username, config.password, config);
}

fs
  .readdirSync(__dirname)
  .filter(file => {
    return (file.indexOf('.') !== 0) && (file !== basename) && (file.slice(-3) === '.js');
  })
  .forEach(file => {
    const model = require(path.join(__dirname, file))(sequelize, Sequelize.DataTypes);
    db[model.name] = model;
  });

Object.keys(db).forEach(modelName => {
  if (db[modelName].associate) {
    db[modelName].associate(db);
  }
});

db.sequelize = sequelize;
db.Sequelize = Sequelize;

module.exports = db;
```

```
'use strict';
const {
  Model
} = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class comment extends Model {
    /**
     * Helper method for defining associations.
     * This method is not a part of Sequelize lifecycle.
     * The `models/index` file will call this method automatically.
     */
    static associate(models) {
      // define association here
    }
  };
  comment.init({
    userId: DataTypes.INTEGER,
    postId: DataTypes.INTEGER,
    content: DataTypes.STRING
  }, {
    sequelize,
    modelName: 'comment',
  });
  return comment;
};
```

```
'use strict';
module.exports = {
  up: async (queryInterface, Sequelize) => {
    await queryInterface.createTable('comments', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      userId: {
        type: Sequelize.INTEGER,
        allowNull: false,
        references: {
          model: "users",
          key: "id",
        },
      },
      postId: {
        type: Sequelize.INTEGER,
        allowNull: false,
        references: {
          model: "posts",
          key: "id",
        },
      },
      content: {
        type: Sequelize.STRING,
        allowNull: false,
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
  },
  down: async (queryInterface, Sequelize) => {
    await queryInterface.dropTable('comments');
  }
};
```

```
'use strict';
module.exports = {
  up: async (queryInterface, Sequelize) => {
    await queryInterface.createTable('posts', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      userId: {
        type: Sequelize.INTEGER,
        allowNull: false,
        references: {
          model: "users",
          key: "id",
        },
      },
      title: {
        type: Sequelize.STRING,
        allowNull: false,
      },
      content: {
        type: Sequelize.STRING,
        allowNull: false,
      },
      attachment: {
        type: Sequelize.STRING,
        allowNull: true,

      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
  },
  down: async (queryInterface, Sequelize) => {
    await queryInterface.dropTable('posts');
  }
};
```

```
'use strict';
module.exports = {
  up: async (queryInterface, Sequelize) => {
    await queryInterface.createTable('users', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      firstName: {
        allowNull: false,
        type: Sequelize.STRING
      },
      lastName: {
        allowNull: false,
        type: Sequelize.STRING
      },
      email: {
        allowNull: false,
        type: Sequelize.STRING
      },
      password: {
        allowNull: false,
        type: Sequelize.STRING
      },
      isAdmin: {
        allowNull: false,
        type: Sequelize.BOOLEAN,
        defaultValue: false
      },
      isActive: {
        allowNull: false,
        type: Sequelize.BOOLEAN,
        defaultValue: false
      },
      keepConnected: {
        allowNull: false,
        type: Sequelize.BOOLEAN,
        defaultValue: false
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
  },
  down: async (queryInterface, Sequelize) => {
    await queryInterface.dropTable('users');
  }
};
```

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

module.exports = (req, res, next) => {
  console.log(
    "______________________________________________________________________________________________"
  );

  console.log(
    "__________________PARTIE COMMUNE, CONCERNE LA REQUETE____________________________________________________________________"
  );

  if (req.headers.authorization === "" || !req.headers.authorization) {
    console.log("pas le cas req.headers.authorization");
    console.log(req.body.headers.Authorization);
  } else {
    if (req.params.id === "" || !req.params.id) {
      console.log("pas le cas req.params.id");

      try {
        console.log(
          "______________________________________________________________________________________________"
        );

        console.log(
          "________________________AUTH (en cours de vérification du NIVEAU)_________________________"
        );
        const token = req.headers.authorization.split(" ")[1];
        const decodedToken = jwt.verify(token, process.env.TOKEN);
        console.log("decodedToken", decodedToken);
        const isAdmin = decodedToken.isAdmin;
        console.log(
          "_________________________________________________________________________________"
        );
        if (isAdmin) {
          console.log(
            "_______________________________________________________________________________"
          );

          console.log(
            "________________________GET ACCESS  => TOKEN OK_(ADMIN ACCESS)_____________________________________________________________"
          );
          next();
        } else {
          console.log(
            "_______________________________________________________________________________"
          );

          console.log(
            "________________________GET ACCESS => TOKEN OK_(USER ACCESS)________________________________________________________________"
          );
          next();
        }
      } catch (error) {
        res.status(403).json({ error: error | "Unauthorized request" });
      }
    }

    if (req.params.id) {
      console.log("cas req.params.id , req.params.id =", req.params.id);
      next();
    }
  }
};
```

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

const MIME_TYPES = { // Dictionnaire mime types pour ceux disponibles depuis le frontend
    'image/jpg': 'jpg',
    'image/jpeg': 'jpg',
    'image/png': 'png'
};

const storage = multer.diskStorage({ // Objet de configuration pour multer. Nécessite 2 élements: destination et filename
    destination: (req, file, callback) => {
        callback(null, 'images')
    },
    filename: (req, file, callback) => { // Générer le nouveau nom
        const name = file.originalname.split(' ').join('_'); // Partie avant l'extension
        const extension = MIME_TYPES[file.mimetype]; // Extension
        callback(null, name + Date.now() + '.' + extension); // Nom global: partie avant l'extension + Timestamp créant un fichier unqiue + extension
    }
});

module.exports = multer({ storage: storage }).single('image'); // Export du middleware configuré avec méthode multer précisant l'unicité du fichier et le type image
```

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

module.exports = (req, res, next) => {
  console.log(
"_____________________________________________________________________________________________"
  );
if (req.headers.authorization === "" || !req.headers.authorization) {
    console.log("req.headers.authorization ABSENT");
  }
  next();
};
```

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

module.exports = (req, res, next) => {


if (req.params.id) {
    console.log("cas req.params.id , req.params.id =", req.params.id);

    try {
      console.log(
        "________________________AUTH (en cours de vérification du NIVEAU)_________________________"
      );
      const token = req.headers.authorization.split(" ")[1];
      const decodedToken = jwt.verify(token, process.env.TOKEN);
      console.log("decodedToken", decodedToken);
      const isAdmin = decodedToken.isAdmin;
      const userId = decodedToken.userId;
      const reqId = req.params.id;
      console.log(userId);
      console.log(
        "_____________________________________________________________________________________________"
      );
      if (isAdmin) {
        console.log(
          "________________________AUTH NIVEAU 3  => TOKEN OK_(ADMIN ACCESS)__________________________________________"
        );
        next();
      } else {
        if (userId === reqId) {
          console.log(
            "________________________AUTH NIVEAU 2  => TOKEN OK_(DELETE AND MODIF ACCESS)__________________________________________"
          );
          next();
        } else userId !== reqId;
        {
          console.log(
            "________________________AUTH NIVEAU 1  => TOKEN OK_(READ ONLY ACCESS)__________________________________________"
          );
          next();
        }
      }
    } catch (error) {
      res.status(403).json({ error: error | "Unauthorized request" });
    }
  }
}
```

```
const db = require("../models/index");
const jwt = require("jsonwebtoken");

exports.createComment = (req, res, next) => {
  // // const token = req.body.headers.Authorization.split(" ")[1];
  // // const decodedToken = jwt.verify(token, process.env.TOKEN);
  // // const userId = decodedToken.userId;

  //       db.comment
  //         .create({
  //           content: req.body.content,
  //           postId: req.body.postId,
  //           userId: req.body.userId,
  //         })
  //         .then(() => res.status(201).json({ message: "Commentaire créé" }))
  //         .catch((error) => res.status(400).json({ error: "erreur catch" }));

};

exports.getAllComments = (req, res, next) => {


  // db.comment
  //   .findAll({
  //     order: [["createdAt", "DESC"]],
  //     where: { postId: req.params.postId}
  //   })
  //   .then((comments) => {
  //     res.status(202).json(comments);
  //   })
  //   .catch((error) => res.status(400).json({ error }));
};

exports.deleteComment = (req, res, next) => {
    // const postId = JSON.parse(req.params.id);

    // db.comment
    //   .findOne({
    //     attributes: ["id"],
    //     where: { id: postId },
    //   })
    //   .then((post) => {
    //     if (!post) {
    //       return res.status(401).json({ error: "Post inexistant" });
    //     }
    //     db.post
    //       .destroy({
    //         attributes: ["id"],
    //         where: { id: postId },
    //       })
    //       .then(() => res.status(200).json({ message: "Post supprimé" }))
    //       .catch(() => res.status(500).json({ error: "erreur catch" }));
    //   })
    //   .catch((error) => res.status(500).json({ error: "erreur catch" }));

};
```

```
const db = require("../models/index");
const jwt = require("jsonwebtoken");
const fs = require("fs");

exports.createPost = (req, res, next) => {

  console.log("req.body", req.body)

  const token = req.headers.authorization.split(" ")[1];
  const decodedToken = jwt.verify(token, process.env.TOKEN);
  const userId = decodedToken.userId;

  db.user
    .findOne({
      where: { id: userId },
    })

    .then((user) => {
      if (user) {

        db.post
          .create({
            content: req.body.content,
            title: req.body.title,
            userId: user.id,
            attachment: req.file ? `${req.protocol}://${req.get('host')}/images/${req.file.filename}`: req.body.image,
          })
          .then(() => res.status(201).json({ message: "Post créé" }))
          .catch((error) => res.status(400).json({ error: "erreur catch" }));
      } else {
        return res.status(404).json({ error: "Utilisateur non trouvé" });
      }
    })
    .catch((error) => res.status(501).json({ error: "erreur catch" }));
};

exports.getAllPosts = (req, res, next) => {
  db.post
    .findAll({
      order: [["createdAt", "DESC"]],
    })
    .then((posts) => {
      res.status(202).json(posts);
    })
    .catch((error) => res.status(400).json({ error }));
};

exports.getOnePost = (req, res, next) => {
  const postId = JSON.parse(req.params.id);

  db.post
    .findOne({
      attributes: ["id", "userId", "title", "content", "attachment"],
      where: { id: postId },
    })

    .then((post) => {
      if (!post) {
        return res.status(401).json({ error: "Post inexistant" });
      }
      res.status(200).json(post);
    })
    .catch((error) => res.status(500).json({ error: "erreur" }));
};

```

```
exports.modifyPost = (req, res, next) => {};

exports.deletePost = (req, res, next) => {
  const postId = JSON.parse(req.params.id);
  console.log(req.params.id);

  db.post
    .findOne({
      attributes: ["id"],
      where: { id: postId },
    })
    .then((post) => {
      if (!post) {
        return res.status(401).json({ error: "Post inexistant" });
      }
      db.post
        .destroy({
          attributes: ["id"],
          where: { id: postId },
        })
        .then(() => res.status(200).json({ message: "Post supprimé" }))
        .catch(() => res.status(500).json({ error: "erreur catch" }));
    })
    .catch((error) => res.status(500).json({ error: "erreur catch" }));
};
```

```
const db = require("../models/index");
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");

exports.signup = (req, res, next) => {
  const email = req.body.email;
  const firstName = req.body.firstName;
  const lastName = req.body.lastName;
  const password = req.body.password;

  if (
    email == null ||
    email == "" ||
    firstName == null ||
    firstName == "" ||
    lastName == null ||
    lastName == "" ||
    password == null ||
    password == ""
  ) {
    return res.status(400).json({ error: "Formulaire incomplet" });
  }

  db.user
    .findOne({
      where: { email: req.body.email },
    })

    .then((user) => {
      if (user) {
        return res.status(401).json({ error: "Email déjà utilisé!" });
      }
      bcrypt
        .hash(req.body.password, 5)
        .then((hash) => {
          db.user.create({
            firstName: req.body.firstName,
            lastName: req.body.lastName,
            email: req.body.email,
            password: hash,
            isAdmin: false,
          });
        })
        .then(() => res.status(201).json({ message: "Utilisateur créé" }))
        .catch((error) => res.status(400).json({ error }));
    })

    .catch((error) => res.status(500).json({ error: "error" }));
};

exports.login = (req, res, next) => {
  const email = req.body.email;
  const password = req.body.password;

  if (email == null || email == "" || password == null || password == "") {
    return res.status(400).json({ error: "Champs manquants" });
  }

  db.user
    .findOne({ where: { email: req.body.email } })

    .then((user) => {
      if (!user) {
        return res.status(401).json({ error: "Utilisateur non trouvé" });
      }

      bcrypt
        .compare(req.body.password, user.password)
        .then((valid) => {
          if (!valid) {
            return res.status(401).json({ error: "Mot de passe incorrect" });
          }
          res.status(200).json({
            email: user.email,
            lastName: user.lastName,
            firstName: user.firstName,
            userId: user.id,
            isAdmin: user.isAdmin,
            token: jwt.sign(
              { userId: user.id, isAdmin: user.isAdmin },
              process.env.TOKEN,
              {
                expiresIn: "12h",
              }
            ),
          });
        })
        .catch((error) => res.status(500).json({ error }));
    })

    .catch((error) => res.status(500).json({ error: "erreur" }));
};

exports.getAccount = (req, res, next) => {

  db.user
    .findOne({
      attributes: ["id", "firstName", "lastName", "email", "isAdmin"],
      where: { id: req.params.id },
    })
    .then((user) => {
      if (!user) {
        return res.status(401).json({ error: "Utilisateur inexistant" });
      }
      res.status(200).json(user);
    })
    .catch((error) => res.status(500).json({ error: "erreur" }));
};

exports.getOneAccount = (req, res, next) => {
  const userId = JSON.parse(req.params.id);
  console.log("req.params.id", req.params.id);

  db.user
    .findOne({
      attributes: ["id", "firstName", "lastName", "email"],
      where: { id: userId },
    })
    .then((user) => {
      if (!user) {
        return res.status(401).json({ error: "Utilisateur inexistant" });
      }
      res.status(200).json(user);
    })
    .catch((error) => res.status(500).json({ error: "erreur" }));
};

exports.modifyAccount = (req, res, next) => {};

exports.deleteAccount = (req, res, next) => {
  const userId = JSON.parse(req.params.id);
  console.log("req.params.id", req.params.id);

  db.user
    .findOne({
      attributes: ["id"],
      where: { id: userId },
    })

    .then((user) => {
      if (!user) {
        return res.status(401).json({ error: "Utilisateur inexistant" });
      }
      db.user
        .destroy({
          attributes: ["id"],
          where: { id: userId },
        })
        .then(() => res.status(200).json({ message: "Compte supprimé" }))
        .catch(() => res.status(500).json({ error: "erreur" }));
    })

    .catch((error) => res.status(500).json({ error: "erreur" }));
};

exports.getAllAccounts = (req, res, next) => {
  db.user
    .findAll({
      order: [["createdAt", "DESC"]],
      attributes: [
        "id",
        "firstName",
        "lastName",
        "email",
        "createdAt",
        "isAdmin",
      ],
    })
    .then((users) => res.status(202).json(users))
    .catch((error) => res.status(400).json({ error: "erreur" }));
};
```

```json
{
  "development": {
    "username": "sqluser",
    "password": "password",
    "database": "groupomania",
    "host": "127.0.0.1",
    "dialect": "mysql"
  },
  "test": {
    "username": "sqluser",
    "password": "password",
    "database": "groupomania_test",
    "host": "127.0.0.1",
    "dialect": "mysql"
  },
  "production": {
    "username": "sqluser",
    "password": "password",
    "database": "groupomania_production",
    "host": "127.0.0.1",
    "dialect": "mysql"
  }
}
```