Groupomania

Développement d'un réseau social d'entreprise

Groupomania

Groupomania

Plan de la présentation

-La mission
- Les outils
- Le développement
- La sécurité
- Les tests
- Conclusion

La mission

La mission

- Développer un réseau social d'entreprise
  - Création d'un compte utilisateur pour accéder au forum
    - Modifiable
    - qui peut être supprimé
  - Accéder à un forum avec du texte ou des contenus multimédia
    - Ajout de commentaires
  - Un modérateur peut modifier ou supprimer les messages existants

La mission en détails

- Mise en place de la base de données
- Développement du backend
    - Développement du serveur
    - Création, modification des utilisateurs
    - Création modification et suppression des messages du forum.
    - Authentification des utilisateurs
- Développement du frontend
    - Mise en forme des pages de connexion/création de compte ou posts
    - Utilisation d'un framework

Contraintes techniques imposées

- Base de données
  - Utilisation d'une Base de données qui se manipule avec le langage SQL
  - Choix de mySQL

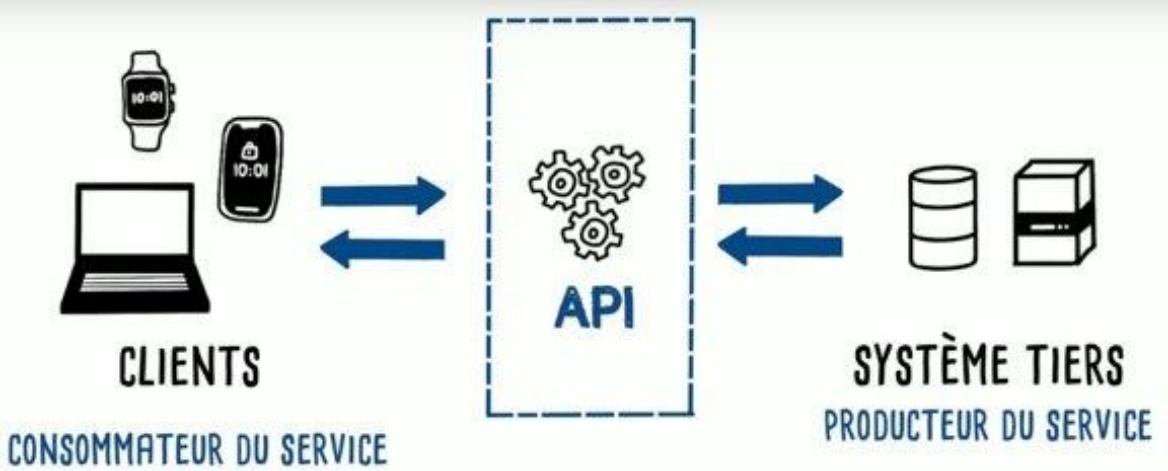
- Utilisation d'un framework de notre choix:
  - Choix: vueJS3 CDN

Les outils

Backend: développement de l'API

- **API: (**Application Programming Interface) signifie en français : « Interface de Programmation d'Application».

- Une interface de programmation permet d'accéder à des services fournis par un système tiers.



Backend : express et nodeJs

- Le développement de l'API se fait avec NodeJS et Express

- Node permet d'écrire toutes les tâches côté serveur, en JavaScript, telles que la logique métier, la persistance des données et la sécurité.

- Express est un environnement de développement reposant sur Node, qui facilite la création et la gestion des serveurs Node.

Backend: basic-auth et jsonWebtoken

- Authentification des utilisateurs à l'aide d'un jeton.
- Permet de sécuriser les connexions.

- Groupomania:

  A la connexion d'un utilisateur:

  - Un jeton est calculé dans le backend et renvoyé au fontend.
  - Le frontend renvoie le jeton à chaque requête pour s'authentifier.
  - Le backend vérifie que l'utilisateur est autorisé à exécuter la requête

Backend : multer

- Multer est un middleware pour la gestion des images et le chargement des fichiers.

- Groupomania:
  - Multer crée dans un dossier appelé "images" l'image envoyée par le front sous forme de datas.

SGBD: Les bases de données MySQL

- SGBD: Système de Gestion de Base de données.

  - Une **base de données** permet de stocker et de retrouver des données structurées.

  - Une **base de données relationnelle:** l'information est organisée dans des tableaux à deux dimensions appelés des *relations* ou *tables.*

SGBD: Les bases de données MySQL

- Les données de groupomania sont stockées dans une base de données relationnelle MySQL.
  - SQL = Structured Query Language

SGBD: Les bases de données MySQL

- Mysql pour Groupomania:
  - Creation de 3 tables:
    - Users: la table des utilisateurs avec les infos de login
      - Moderateur oui/non
    - Posts: la table des posts avec:
      - userId = auteur du post
      - Le post
      - L'image
    - Comments: la table des commentaires avec :
      - Le commentaire
      - postId: le post du commentaire
      - userId: l'auteur du commentaire

SGBD: Les bases de données MySQL

- Les relations permettent d'intéragir entre les tables.
  - Ex Groupomania:
    - "userId" dans la table posts donne les informations de l'utilisateur
    - "postId" et "userId" dans la table comments donnent les informations sur le post et l'utilisateur

- L'instruction ON DELETE CASCADE permet:
  - Lors de la suppression d'un utilisateur

    -> la suppression des posts associés

    -> la Suppression des commentaires associés

Frontend vueJs

- VueJs est un framework .

- Un **framework** (ou **infrastructure logicielle** en français) désigne un ensemble d'outils et de composants logiciels utilisés pour la création de l'application.

Les outils: Frontend vueJs3 CDN

- Utilisation de vueJs3 CDN (content delivery network)
  - Permet d'intégrer du java Script dans le DOM
  - Plus facile que l'utilisation du java script classique.

Frontend fetch

- Envoie des requêtes à l'API
  - Header
  - La méthode:
    - POST
    - PUT
    - DELETE
  - Les données

```
const data = {
  email: this.email,
  password: this.passwd,
  pseudo: this.pseudo,
};

//Url  du user
const userUrl = "http://localhost:3000/api/auth/signup";

//header
const bearerToken = "bearer null";

fetch(userUrl, {      You, 2 weeks ago • Frontend index.html: ft login: a
  method: "POST",
  headers: {
    "Content-Type": "application/json",
    Authorization: bearerToken,
  },
  body: JSON.stringify(data),
})
```

Le développement

Plan de développement

- Backend:
  - Le serveur
  - Les routes
  - Les middlewares
  - Les requetes de MySQL
  - Les controllers
- Frontend:
  - Prise en main de VueJs3
  - La création du compte utilisateur
  - La connexion
  - Les posts

Le CRUD

- Le CRUD est un acronyme pour les actions sur des données stockées
  - CREATE : Créer
  - Read : Lire
  - Update : Mettre à jour
  - Delete : Supprimer

- L'API de groupomania est développée pour réaliser chacune de ces actions pour les utilisateurs et les posts/commentaires.

Backend: Les routes

- Users:
  - POST: création d'un utilisateur / connexion
  - PUT: modification
  - DELETE: suppression
- Posts / commentaires
  - POST: création
  - PUT: modification
  - DELETE: suppression

```
//routes POST pour signup et login envoyé par le frontend
router.post("/signup", userCtrl.signup);
router.post("/login", userCtrl.login);

//Route PUT pour modification d'un utilisateur
router.put("/:userId", auth, userCtrl.modifyUser);

//Suppression d'un utilisateur  DELETE
router.delete("/:userId", auth, userCtrl.deleteUser);
```

Backend: Les routes

- Pour chaque route:
  - L'URL
  - Les middlewares
  - Le controller chargé du traitement

Routes pour les posts:

```
router.post("/", authPost, multer, postCtrl.createPost);

router.get("/", authPost, postCtrl.getAllPost);
router.get("/:postId", authPost, postCtrl.getOnePost);

router.put("/:postId", authPost, authPostOwner, multer, postCtrl.modifyPost);

router.delete("/:postId", authPost, authPostOwner, postCtrl.deletePost);
```

Backend : les middlewares

- Les middlewares: assurent la communication entre les applications
- Auth:
  - Authentification de l'utilisateur
    - Seul l'utilisateur connecté peut :
      - Modifier son profil
      - Supprimer son compte
  - Posts et commentaires:
    - Vérification user = auteur ou modérateur

```
Ex: sql: SELECT moderator FROM users WHERE userId ='5';
```

Backend : les middlewares

- Les middlewares:
- Multer:
  - Traite les images reçues du frontend:
    - Crée une image dans le dossier images.

Backend : les middlewares

- Exemple

```
router.put("/:postId", authPost, authPostOwner, multer, postCtrl.modifyPost);

router.delete("/:postId", authPost, authPostOwner, postCtrl.deletePost);
```

- AuthPost: Vérifie que le user est authentifié
- AuthpostOwner: vérifie que le user est le propriétaire ou est modérateur
- Multer: traite l'image si elle existe.

Backend : les controllers – requetes SQL

users:

Mise à jour des requêtes SQL pour:
- Créer un utilisateur
- Modifier un utilisateur
- Supprimer un utilisateur

Exemple :

Création d'un nouvel utilisateur:

```
INSERT INTO users ((email, passwd, pseudo) VALUES ('email','password","pseudo")
```

Suppression d'un utilisateur:

```
DELETE from "users"  WHERE userId=<userId>
```

Le "ON DELETE CASCADE" détruira les posts et les commentaires de ce user.

Backend : les controllers – requetes SQL

posts et commentaires:

Mise à jour des requêtes SQL pour:
- Créer un post/un commentaire
- modifier un post/un commentaire
- supprimer post/un commentaire

Exemple :
Création d'un nouvel post

```
 INSERT INTO posts (post, imageUrl, userId, PostDate) VALUES
('Une jonquille','http://localhost:3000/images/Jonquille.png1654610409287.png','5' ,NOW());

NOW: donne la date et l'heure de la création
```

Backend : les controllers – requetes SQL

Lire les informations:

Exemple: lire tous les posts

```
SELECT * , DATE_FORMAT(postDate,'%d/%m/%Y   %H:%i:%S') AS date
FROM  posts
LEFT OUTER JOIN users
ON  posts.userId=users.userId
ORDER by postDate DESC;
```

DATE_FORMAT(postDate,'%d/%m/%Y %H:%i:%S')AS date
permet d'avoir la date formatée format français date et heure

Backend : les controllers – requetes SQL

Lire les informations:

Un tableau est envoyé au frontend avec les informations suivantes pour chaque post:

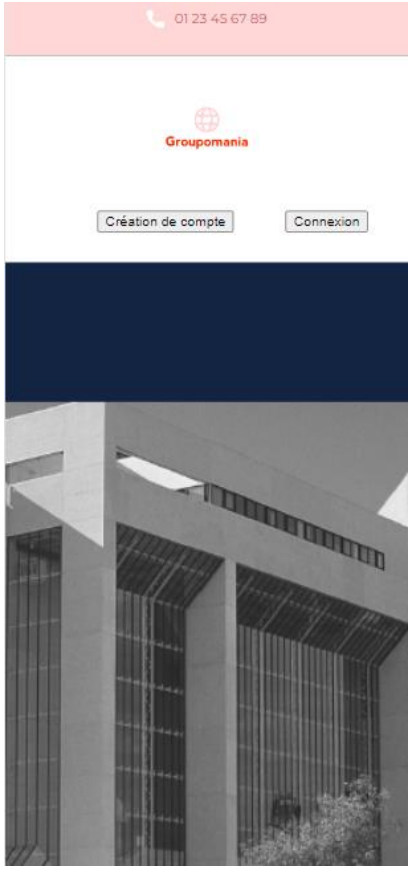
```
postId: 137,
userId: 5,
post: 'La jonquille',
imageUrl: 'http://localhost:3000/images/Jonquille.png1654266287196.png',
postDate: 2022-06-03T14:24:47.000Z,
email: 'titi@test.fr',
passwd: '$2b$10$EU/NUaiJbB.y4pzz6e3vO.2JwqPF36RtaByb.Kw4JgTpHwpJDEHue',
pseudo: 'titi20',
moderator: 1,
date: '03/06/2022   16:24:47'
```

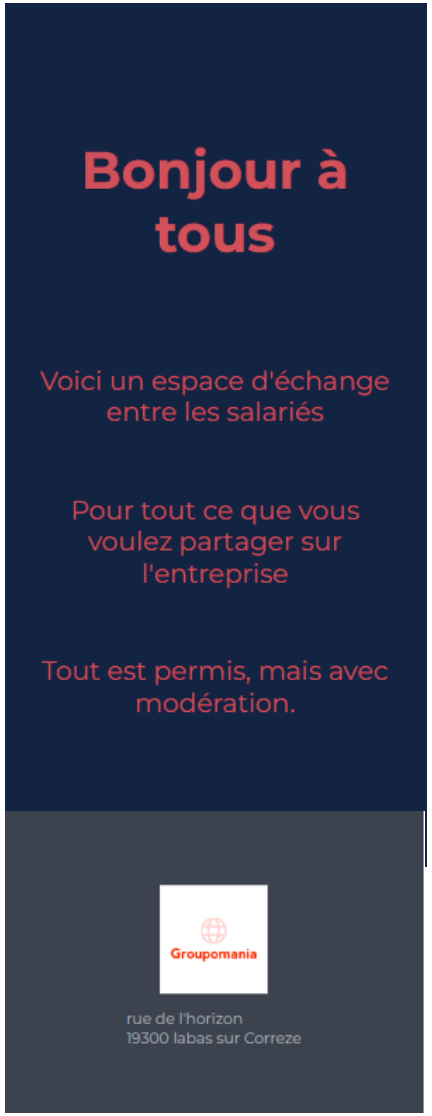
Backend : les controllers

Développement des fonctions pour:

Create/modify/delete users
Create/modify/delete posts
Create/modify/delete commentaires.
Affichage des posts
Affichage des commentaires.

Frontend : index.html





Frontend :posts.html

Frontend : fetch

- Fetch permet de communiquer avec l'API
  - Pour chaque requête
    - Headers avec token d'identification de l'utilisateur
    - L'URL de la route à joindre
    - Les données si nécessaire.

  - Traitement du code et des datas renvoyées par le serveur.

Frontend : fetch

- Fetch permet de communiquer avec l'API
  - Cas particulier des images: utilisation de formData

```
const formData = new FormData();
formData.append("post", this.newPost);

//Ajout image si sélectionnée
if (this.images) {
  formData.append("imageUrl", this.images.name);
  formData.append("image", this.images, this.images.name);
}
```

Frontend : fetch

```
//Url du post
const postUrl = "http://localhost:3000/api/post";

fetch(postUrl, {
  method: "POST",
  headers: {
    //  "Content-Type": "application/json",
    Authorization: bearerToken,
  },
  // body: JSON.stringify(data),
  body: formData,
})
  .then((response) => response.json()) //reponse du fetch
  .catch((error) => {
    console.log("Fetch error: ", error);
  })
```

Frontend : Vuejs

- Les datas:
  - Définition de toutes les variables affichées dans le HTML
- Le cycle de vie de vue :
  - BeforeCreate():
    - Affichage de la page au chargement:
      - Vérifie la bonne connexion au serveur
  - Created():
    - Etape d'initialisation
      - Contacte l'API pour afficher les posts, l'utilisateur connecté.
  - Mounted():
    - Page complètement chargée.

Frontend: VueJS 3

- Les composants:
  - Pour users et posts:
    - Envoi des requêtes au backend en ft des données des formulaires du HTML
    - Analyse des réponses du backend.
  - VueJS permet de faire apparaitre des parties du HTML en fonction de tests:
    - Utilisation balise moustache {{}} pour afficher les variables
    - V-if: test d'une variable
    - V-for: boucle (par exemple pour afficher les posts ou les commentaires
    - V-model: pour récupérer une valeur rentrée dans un formulaire

Frontend: VueJS 3 – exemple v-if

```
<h1>le forum des salariés de Groupomania</h1
<template v-if="!isUserConnected">
  <!-- Cas du user non connecté qui aurait c
  <h2>Vous n'êtes pas connecté</h2>
  <h2>Retournez sur la page accueil</h2>
  <h2>pour vous connecter</h2>
</template>

<template v-else>
  <!-- Affichage entete pour les users connectés -->
  <div class="postHeader">
    <h3>Bonjour {{connectedName}}</h3>
    <template v-if="connectedModerator">
      <h3>Vous êtes moderateur</h3>
    </template>
```

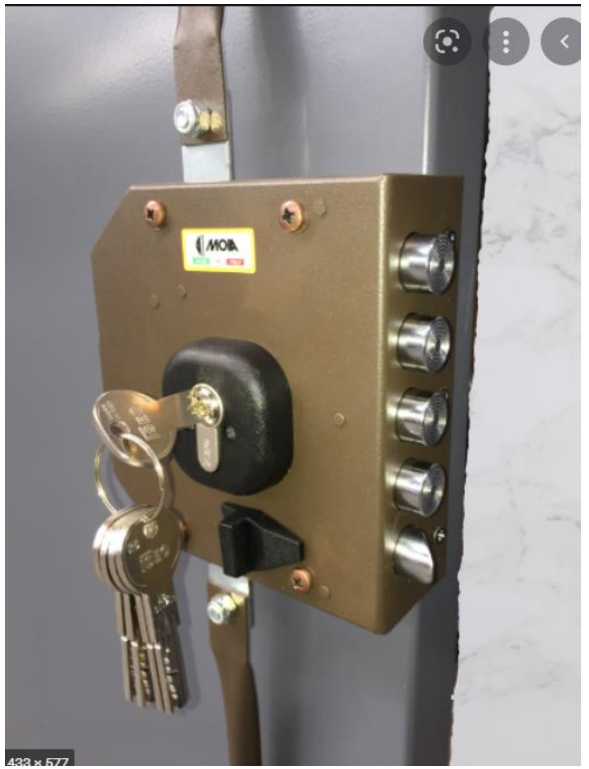
Cas 1: user riri (ordinaire)

```
<h1>le forum des salariés de Groupomania</h1>
<!-- Affichage entete pour les users connectés -->
▼<div class="postHeader"> flex
  <h3>Bonjour riri</h3>
  <!--v-if-->
```

Cas 2: user titi (modérateur

```
<h1>le forum des salariés de Groupomania</h1>
<!-- Affichage entete pour les users connectés -->
▼<div class="postHeader"> flex
  <h3>Bonjour titi</h3>
  <h3>Vous êtes moderateur</h3>
```

La sécurité



Le cryptage des mots de passe

- Le mot de passe est crypté avec bcrypt et sauvegardé crypté.
  - On ne peut pas retrouver le mot de passe à partir de la valeur cryptée.
  - On peut vérifier qu'un nouveau mot de passe est le bon ou pas.

Le système de jetons

- A chaque connexion:
  - Le backend crée un jeton qui est envoyé au frontend.
  - A chaque requête:
    - Le frontend renvoie le jeton au backend
    - Le backend vérifie le jeton
    - Le userId est déduit du jeton
  - Pour les modifs et suppression de user:
    - Seul le propriétaire pour exécuter la requête
  - Pour les modifs et suppression de mots et commentaires:
    - Seul le propriétaire ou le modérateur peuvent exécuter la requête.

Les tests

Plan de tests

- Le plan de test est divisé en:
  - Tests nominaux
  - Tests d'erreur
  - Tests aux limites

Outils de tests

- Test du backend pour vérifier toutes les fonctions:
  - Postman pour tester l'API

Détail des tests – les utilisateurs

- Création du compte
- Connexion
- Modification du profil
  - Email
  - Pseudo
  - Mot de passe
- Déconnexion

Détail des tests – utilisateurs

- Supprimer un utilisateur
  - Vérifier qu'il ne peut plus accéder au forum
  - Vérifier que ses posts et commentaires sont détruits.

Détail des tests – les posts / commentaires

- Pour les posts ou commentaires:
  - Création
    - Post: Avec et sans image
- Modification
- Suppression

- Affichage
  - Vérifier l'affichage des boutons Modify/delete pour le propriétaire ou modérateur

- Vérifier dans le dossier image la création ou suppression de l'image du post.

Détail des tests – tests d'erreur

- Utilisateurs:
  - Création de compte utilisateur avec mail ou pseudo existant
  - Connexion avec mauvais mot de passe

- Tests avec outil type postman
  - Posts:
    - Tentative de modification ou destruction d'un post alors que le user n'est ni propriétaire ni moderateur

Conclusion

Conclusion sur le projet

- Les fonctionnalités demandées ont été implémentées.

- Affichage à améliorer:
  - Faire un sondage auprès des salariés sur le graphisme

  - Faire tester le produit par les salariés de Groupomania.

Liens utiles

- API REST avec express et mySQL:
  - https://dev.to/nurofsun/building-simple-rest-api-with-express-js-and-mysql-140p

- ROW DATA packet;
- https://stackoverflow.com/questions/31221980/how-to-access-a-rowdatapacket-object

- Mots clé exports et require
- https://h-deb.clg.qc.ca/Sujets/Web/nodejs-exports-require.html
- https://www.sitepoint.com/understanding-module-exports-exports-node-js/

Liens utiles

- Codes de retour:
  - https://www.ibm.com/docs/fr/odm/8.5.1?topic=api-rest-response-codes-error-messages
- Mysql: on delete cascade
  - https://www.mysqltutorial.org/mysql-on-delete-cascade/
- Doc de vueJS
  - https://vuejs.org/
  - Cours VueJS3 par Antoine Creuzet
  - https://www.training-dev.fr/Cours/Les-bases-de-Vue.JS-3/

Liens utiles

fetch

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

Chaines de caractères:

Changer toutes les occurrences d'une chaine de caracteres. (cas plusieurs ')

multipart

https://refine.dev/blog/how-to-multipart-upload/

MVC:

https://rosedienglab.defarsci.org/a-quoi-sert-une-architecture-mvc-son-fonctionnement/

Installations

- basic-auth: vérification authentification
- bcrypt: cryptage du mot de passe
- dotenv: gestion du fichier pour les variables d'environnement sensibles comme user/passwd ou clés secrétes.
- express: express est un environnement de développement reposant sur Node, qui facilite la création et la gestion des serveurs Node.
- jsonwebtoken: création d'un token pour l'utilisateur
- multer: gestion des images
- mysql: base de données relationnelle
- nodemon: permet de redemarrer le server à chaque mise à jour de fichier.

Glossaire

- API: Application Programming Interface
- CDN (content delivery network)
- CRUD: Create Read Update Delete
- Framework: infrastructure de développement
- Hash: Un hash est le résultat de l'application d'une fonction de hachage cryptographique. Pour vérifier la validité du mot de passe d'un utilisateur, il suffit de calculer le hash du mot de passe envoyé et de comparer à la valeur stockée en base de données.
- JSON: JavaScript Object Notation.
- HTTP: HyperText Transfert Protocol
- Middleware is software that lies between an operating system and the applications running on it. Essentially functioning as hidden translation layer, middleware enables communication and data management for distributed applications.

Glossaire

- MVC: Modèle Vue Controleur
- NodeJS: est une plateforme logicielle libre en JavaScript, orientée vers les applications réseau évènementielles
- REST: Representational State Transfer
- SGBD: Système de Gestion de Base de Données
- SQL = Structured Query Language

Utile

- = est utilisé pour attribuer des valeurs à une variable en JavaScript.
- == est utilisé pour la comparaison entre deux variables quel que soit le type de la variable.
- === est utilisé pour une comparaison stricte entre deux variables c'est à dire que cela vérifiera le type et la valeur.