Spécifications de l’API : Projet 7 Groupomania

L'application Groupomania est un réseau social d'entreprise ou les utilisateurs connectés peuvent accéder aux post des autres utilisateurs les commenter et les liker.
L'utilisateur connecté peut lui même rédiger son propre post qui sera visible par tous les autres utilisateurs ayant un compte et étant authentifié.

Les routes de l'API :

Routes « AUTH »

	Point d'accès	Request body	Réponse attendue	Fonction
POST	https://backend.thiery-samuel.com/api/auth/signup	{lastName: string, firstName: string, password: string, email: string}	{message: 'compte créé avec succès'}	Ajout d'un nouvel utilisateur à la base de donnée hache le mot de passe et crypte l'email.
POST	https://backend.thiery-samuel.com/api/auth/login	{email : string, password: string}	{userId: string, token: string}	Vérifie les données utilisateur pour son authentification, génère un token contenant le userId

Routes « USERS »

	Point d'accès	Request Body	Réponse attendue	Fonction
GET	https://backend.thiery-samuel.com/api/users/:userId	id de l'utilisateur en paramètre	{admin: number, first_name: string, last_name: string, user_id: number}	Renvoie le nom, prénom si l'utilisateur connecté est administrateur ainsi que son id.
PUT	https://backend.thiery-samuel.com/api/users/:userId/password	{oldPassword: string, newPassword: string}	{message: 'mot de passe changé avec succès'}	Authentifie la requête en comparant les mots de passe, puis modifie le mot de passe si celui ci respecte le schéma.
PUT	https://backend.thiery-samuel.com/api/users/:userId	{newLastname: string, newFirstName: string, newEmail : string}	{message: 'profil modifié avec succès'}	Modifie les valeurs dans la bdd avec celles de l'utilisateur, crypte le nouvel email.
DELETE	https://backend.thiery-samuel.com/api/users/:userId	{password: string}	{message: 'profil supprimé'}	Supprime l'utilisateur de la base de donnée ainsi que ses commentaires likes et articles associés.

Routes « COMMENTS »

		Request Body	Réponse attendue	Fonction
GET	https://backend.thiery-samuel.com/api/user/:userId/article/:articleId/comments	id de l'article en paramètre & id de l'utilisateur en paramètre	{last_name: string, first_name: string, comment_text: string, id: number, comment_author}	extrait de la bdd tous les commentaires de l'article dont l'id est passé en paramètre
POST	https://backend.thiery-samuel.com/api/user/:userId/article/:articleId/comment	{writterOfComment: number, comment: string, articleId: number}	{message: 'commentaire posté'}	Enregistre les commentaire posté par un utilisateur.
DELETE	https://backend.thiery-samuel.com/api/user/:userId/article/:articleId/comments/:commentId	id de l'article en paramètre & id de l'utilisateur en paramètre & id de l'article en paramètre & id du commentaire en paramètre	{message: 'commentaire effacé par son rédacteur/ administrateur'}	retire de la bdd un commentaire, seul le rédacteur ou l'administrateur le peuvent.

Routes « LIKES »

		Request Body	Réponse attendue	Fonction
POST	https://backend.thiery-samuel.com/api/user/:userId/article/:articleId/like	{userId: number, article: number, like: number}	{message: string}	Vérifie si l'utilisateur a déjà liké l'article, si oui, la valeur est remplacée par la nouvelle, si non, une nouvelle valeur est crée dans la base de donnée.
GET	https://backend.thiery-samuel.com/api/user/:userId/article/:articleId/like	id de l'article en paramètre & id de l'utilisateur en paramètre	{likes: number, article: number}	renvoie les likes de l'utilisateur connecté par articles dont l'id est passé en paramètre.
GET	https://backend.thiery-samuel.com/api/user/:userId/article/:articleId/total-likes	id de l'article en paramètre & id de l'utilisateur en paramètre	{likes: number}	Renvoie la somme des likes par articles dont l'id est passé en paramètre de la requête

Routes « Articles »

		Request Body	Réponse attendue	Fonction
GET	https://backend.thiery-samuel.com/api/user/:userId/articles	id de l'utilisateur en paramètre	{article: string, author: number, first_name: string, id: number, last_name: string, post_date: date, title: string, url: string}	Envoie au front les articles demandés avec un offset et limit passé en paramètre dans l'url.
GET	https://backend.thiery-samuel.com/api/user/:userId/article/:articleId	id de l'article en paramètre & id de l'utilisateur en paramètre	{article: string, author: number, first_name: string, id: number, last_name: string, post_date: date, title: string, url: string}	Envoie au front un article dont l'id est passé en paramètre de l'url.
GET	https://backend.thiery-samuel.com/api/user/:userId/articles/search/:keyword	id de l'article en paramètre & mots saisies dans le champ de recherche en paramètre	{article: string, author: number, first_name: string, id: number, last_name: string, post_date: date, title: string, url: string}	Envoie au front un ou des articles dont le titre, le texte ou le nom et prénom matchent avec u mot clé passé en paramètre de l'url.
GET	https://backend.thiery-samuel.com/api/user/:userId/toparticles	id de l'utilisateur en paramètre	{article: string, author: number, first_name: string, id: number, last_name: string, post_date: date, title: string, url: string}	Envoie au front les articles les plus likés avec la commande SQL ORDER BY SUM et limitée à 6 résultats.
POST	https://backend.thiery-samuel.com/api/user/:userId/article	{title: string, text: string, author: number} {file}	{message: 'votre article est publié'}	Les données sont filtrées par une regex avant d'être insérées dans la base de données. Le fichier image est stockés dans le dossier 'images'.
DELETE	https://backend.thiery-samuel.com/api/user/:userId/article/:articleId	id de l'article en paramètre & id de l'utilisateur en paramètre	{message: 'article effacé par son auteur/ administrateur'}	L'id de l'utilisateur est passée en paramètre puis comparée à celle de l'auteur de l'article dont l'id est passée en paramètre, si l'utilisateur connecté est le même que l'auteur ou est administrateur alors l'article est effacé, avec les commentaires et likes liés.

Middleware d'authentification :

Toutes les routes articles, comments likes et users doivent disposer d'une autorisation qui est envoyé par le back lors de la connexion (login) et qui est stocké par le front qui le renvoie dans l'entête de chaque requête (Bearer <token>)
Chaque routes doit avoir en paramètre d'URL le userId de l'utilisateur connecté qui est controlé par le middleware 'auth.js'. le middleware compare le token au userId passé en paramètre sur toutes les routes (sauf celles d'authentifications).
Seule exception tolérée si l'utilisateur connecté est administrateur.

Base de donnée MY SQL :

La base de donnée My SQL est composée de 4 tables :
- users,
- likes,
- comments,
- articles.

La table USERS a 6 colonnes :
user_id de type NUMBER Auto Increment,
last_name de type VARCHAR
first_name de type VARCHAR
password de type VARCHAR contient le mot de passe haché par Bcrypt
email de type VARCHAR contient l'email crypté par cryptoJs
admin de type BOOLEAN

La table ARTICLES a 6 colonnes:
id de type NUMBER Auto Increment,
title de type VARCHAR, contient le titre de l'article,
article de type VARCHAR contient le corps de l'article,
author de type NUMBER contient l'id de l'auteur de l'article,
url de type VARCHAR contient le chemin url pour les images uploadées par le rédacteur de l'article,
post_date de type DATETIME contient la date et l'heure de rédaction de l'article.

La table LIKES a 4 colonnes :
id de type NUMBER Auto Increment,
user de type NUMBER, contient l'id de l'auteur du like
likes de type BOOLEAN (0 ou 1), contient le choix de l'utilisateur de liker ou non l'article.
article de type NUMBER, contient l'id de l'article liké.

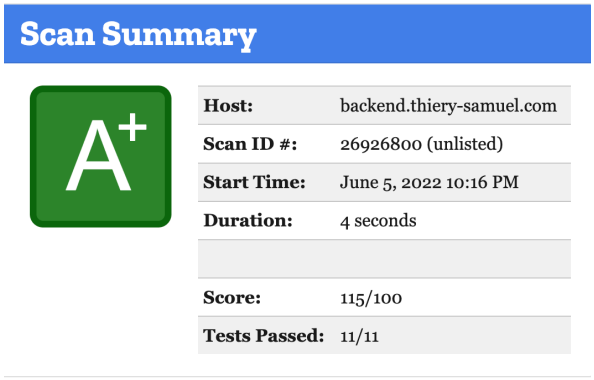
La table COMMENTS a 4 colonnes :
id de type NUMBER Auto Increment,
comment_author de type NUMBER, contient l'id de l'auteur du commentaire,
comment_text de type VARCHAR, contient le texte du commentaire,
commented_article de type number, contient l'id de l'article commenté.

Les relations sont :
users.id -> articles.author ON DELETE CASCADE
articles.id -> comments.commented_article ON DELETE CASCADE
articles.id -> likes.article ON DELETE CASCADE

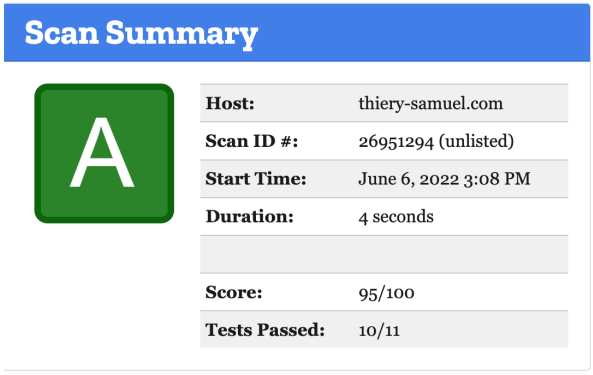
Sécurité :

La sécurité est assurée côté backend par :
- Helmet, appliqué sur l'api.
- Rate-limiter appliqué sur l'api.
- Bcrypt crypte le mot de passe sur les routes d'authentification.
- Crypto-js crypte l'email de l'utilisateur (HmacSHA256).
- Chaque requête est filtrée par des regex (les mêmes que celles utilisées par le front).
- Une redirection en HTTPS est mise en place dans le fichier htaccess (uniquement sur la version en ligne).
- Des messages d'erreurs personnalisés.
- Les mots de passes de base de donnée et clés de cryptage dans un fichier dotenv (inclus dans le .gitignore) et dans environment variable sur la version en ligne.

La sécurité côté front est assurée par :
- L'ajout de balise meta-tag pour les CSP (Content Security Policy) n'autorisant que le service google fonts, fontawesome ou l'API.
- Les certificats SSL pour que le site soit en HTTPS.
- Une redirection dans le fichier HTACCESS de HTTP vers HTTPS.
- l'en-tête X-Frame : 'deny'.



Score obtenu du Backend sur Mozilla Observatory



Score obtenu du Frontend sur Mozilla Observatory

Le code de l'application est disponible sur github à cette adresse :

https://github.com/FedoraSam/p7-OpenClassRooms.git

L'application est en ligne à cette adresse :

https://thiery-samuel.com
L'identifiant de l'administrateur est : admin.admin@gmail.com
Le mot de passe est : Supermotdepasse56