GROUPE-7 API REST BACKEND DOCUMENTATION

Bienvenue sur le WIKI du projet API du groupe-7, dans ce document vous trouverez toutes les informations nécessaires pour lancer le serveur, tester les routes et comprendre son architecture.

Arborescence du projet

Voici l'arborescence du backend :

```
|   app.js
|   index.js
|   package-lock.json
|   package.json
|
+---controllers
|       astuce.js
|       commentaires.js
|       login.js
|       o_auth_login.js
|       profil.js
|       register.js
|       users.js
+---files
+---keys
|       keys.js
|
+---middlewares
|       add_file.js
|       token_auth.js
|
+---models
|       astuces.js
|       commentaires.js
|       tags.js
|       users.js
|
\---routes
        accueil.js
        astuce.js
        commentaires.js
        login.js
        o_auth_login.js
        profil.js
        register.js
        users.js
```

Quelques Explications

- Le fichier **index.js** est le point d'entrée du serveur, c'est le premier fichier qui sera exécutée quand on essaie d'accéder à une route.
- Le fichier **app.js** est celui qui coordonne tous les fichiers nécessaires pour l'exécution des routes de tous les routes .
- Dans le dossier **routes**, il existe un fichier pour chaque route. La logique d'appel, le verbe HTTP à utiliser y est définie.

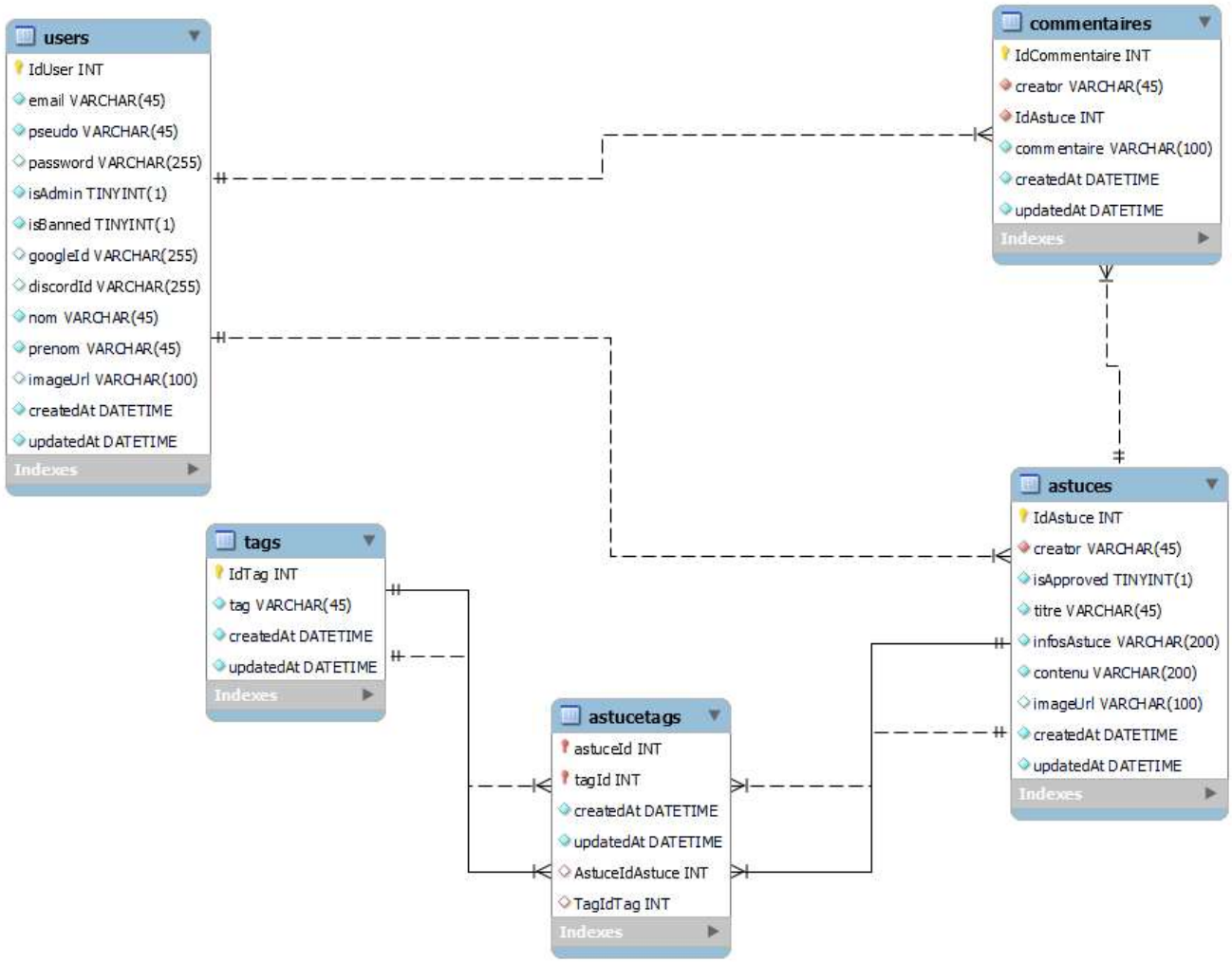
- Dans le dossier **controllers**, il existe un fichier pour chaque route: chaque fichier contenant les fonctions d'accès ou modifications des données, Les fichiers se trouvant dans le dossier routes ont besoin de ces fichiers pour bien s'executer.
- Dans le dossier **models**, se trouve toute la logique de notre base de données : la connexion, la création des tables.
- Un fichier correspondant à chaque table dans la base de données selon la modélisation faite avec Sequelize
- Le dossier **files** contiendra l'ensemble des fichiers uploadés par les utilisateurs, il est vide pour le moment
-

Modélisation de la BASE DE DONNES

Voici l'architecture de la base de données :
Les relations entre les tables sont comme ceci :

1. Un utilisateur est relié à 0 ou plusieurs astuces et une astuce ne peut appartenir qu'à un seul utilisateur
2. Une astuce peut avoir 0 ou plusieurs commentaires et un commentaire n'est lié qu'à une seule astuce.
3. Un utilisateur peut ajouter 0 ou plusieurs commentaires sur 0 ou plusieurs astuces , un commentaire n'est lié qu'à un seul utilisateur.
4. Une astuce peut avoir 0 ou plusieurs tags et un tag peut être associé à une ou plusieurs astuces.

Schéma



Stockage des Images

Nous avons décidé de stocker les images de notre projet dans un dossier et à transmettre le lien vers ces images à votre base de données au lieu de stocker directement les images dans la base de données car ca nous permet de gagner en performance, en place et en simplicité de gestion, ce qui peut être particulièrement utile dans le cadre de ce projet.

Technologies Utilisées

Les technologies utilisées sont celles qui ont été dictées dans l'énoncé :

1. Node et Express JS pour les frameworks backend
2. Passport JS (local, google, discord) pour l'authentification
3. MySQL et Sequelize pour l'interfacage avec une base de données
4. Multer pour la gestion des fichiers

MISE EN ROUTE

Une fois, le contenu du projet téléchargé, rendez-vous sur le répertoire du projet et initialisé un projet node à l'aide de la commande :

```
npm init
```

Laissez les options par défaut et au niveau de l'entry point, mettez:

```
entry point: index.js
```

Pour une bonne mise en route, notre API a besoin des dépendances suivantes:

```
express,jsonwebtoken,multer,mysql2,nodemon,passport,passport-jwt,passport-local,sequelize, passp
```

Elles sont aussi listées dans le fichier :

> package.json

Toujours dans le répertoire racine du projet, lancez la commande:

```
npm install
```

Elle installera toutes les dépendances nécessaires pour lancer le projet.

Ou vous pouvez installer un à un les dépendances avec cette commande:

```
npm install dependance_name --save
```

LES ROUTES DISPONIBLES DE L'API

Endpoint	Paramètres	Méthode d'envoi	Données Retournées	Prérequis	Description
/accueil	aucun paramètre	GET	Point d'entrée du serveur, renvoie un BIENVENUE SUR NOTRE PAGE D'accueil	Aucun	Envoie juste une salutation à l'utilisateur

Endpoint	Paramètres	Méthode d'envoi	Données Retournées	Prérequis	Description
/register	email, username, password, firstName, lastname, file	POST	Renvoie OK si l'utilisateur a bien été crée ou KO sinon	Aucun	Enregistrer l'utilisateur dans la base de données à travers les informations fournies, il pourra ensuite se connecter
/login	email, password	POST	renvoie un token pour l'utilisateur s'il existe et s'il a fourni les bonnes données	Vérification dans la base de données si l'utilisateur correspond	Permet d'authentifier l'utilisateur: à travers son email/password; et ensuite lui renvoie un token qui va être generé à travers son pseudo et d'autres paramètres
/profil/show	username : email de l'user	GET	Renvoie les informations personnelles correspondantes au username transmis	La requête doit être authentifiée à l'aide d'un TOKEN	Permet à un utilisateur disposant d'un token valide(connecté) d'avoir accès à ses informations de profil, un user ne peut voir que son profil
/profil/modify	les nouvelles informations de l'utilisateur (comme dans /register) +	POST	Renvoie un message de confirmation ou d'echec selon le résultat de la requête	La requête doit être authentifiée à l'aide d'un TOKEN	Permet de modifier les informations de profil d'un utilisateur sous réserve d'une

Endpoint	Paramètres	Méthode d'envoi	Données Retournées	Prérequis	Description
	l'attribut IdUser de l'utilisateur : les cas ou l'utilisateur envoie ou non une nouvelle photo de profil ont été gerés				bonne authentification de celui ci
/astuces/add	Informations de l'astuce(titre, infosAstuce, contenu, file)	POST	Permet de créer une astuce et de l'ajouter à la base de données	La requête doit être authentifiée à l'aide d'un TOKEN, un utilisateur connecté ayant logiquement un token	Permet à un utilisateur de créer une astuce et de l'ajouter à la base de données
/astuces/all	Aucun	GET	Renvoie toutes les astuces présentes dans la base de données, qu'elles soient approuvées ou pas	La requête doit être authentifiée à l'aide d'un TOKEN, et l'utilisateur possédant le token doit être un administrateur	Permet à l'administrateur de voir toutes les astuces présentes dans la base de données, ce qui lui permettra d'approuver ou de supprimer certaines
/astuces/all/approved	Aucun	GET	Renvoie toutes les astuces approuvées présentes dans la base de données, qu'elles soient approuvées ou pas	Aucun, accès tout public	Permet aux utilisateurs de voir toutes les astuces approuvées présentes dans

Endpoint	Paramètres	Méthode d'envoi	Données Retournées	Prérequis	Description
					la base de données
/astuces/show	l'Identifiant de l'astuce 'IdAstuce'	GET	Renvoie sous format JSON, les données d'une astuce ainsi que les commentaires associées à cette astuce, l'astuce devant être approuvée au préalable	La requête doit être authentifiée à l'aide d'un TOKEN, un utilisateur connecté ayant logiquement un token	Permet à un utilisateur connecté de consulter une astuce ainsi que les commentaires associés
/astuces/:creator (email)	Aucun	GET	Renvoie sous format JSON, toutes les astuces crées par l'utilisateur qui a effectué la requête	La requête doit être authentifiée à l'aide d'un TOKEN, un utilisateur connecté ayant logiquement un token	Permet à un utilisateur connecté de consulter toutes les astuces lui appartenant ainsi que les commentaires associés
/astuces/delete	Identifiant de l'astuce à supprimer 'IdAstuce'	DELETE	Suppression d'une astuce, renvoie un message de confirmation	La requête doit être authentifiée à l'aide d'un TOKEN, un utilisateur connecté ayant logiquement un token	Permet à un utilisateur de supprimer une astuce lui appartenant, ou à un admin de supprimer n'importe qu'elle astuce
/astuces/approve	Identifiant de l'astuce 'IdAstuce'	POST	Approbation d'une astuce par un administrateur, renvoie un message de confirmation	La requête doit être authentifiée à l'aide d'un TOKEN, l'utilisateur	Permet à un admin d'approuver une astuce

Endpoint	Paramètres	Méthode d'envoi	Données Retournées	Prérequis	Description
				associé étant un admin	
/commentaires/add	Données sur le commentaire (creator, idAstuce, commentaire)	POST	Ajout de Commentaires : renvoie un message de confirmation	La requête doit être authentifiée à l'aide d'un TOKEN, un utilisateur connecté ayant logiquement un token	Permet à un utilisateur d'ajouter un commentaire sous une astuce
/commentaires/delete	Identifiant du commentaire à supprimer	DELETE	Suppression d'un commentaire, renvoie un message de confirmation	La requête doit être authentifiée à l'aide d'un TOKEN, l'utilisateur associé étant un admin	Permet à un admin de supprimer un commentaire
/commentaires/read	Identifiant du commentaire à lire	GET	Renvoie le commentaire avec tous ses attributs correspondant à l'Id en paramètre	La requête doit être authentifiée à l'aide d'un TOKEN, l'utilisateur associé étant logiquement connecté	Permet de lire un commentaire
/users/bann	pseudo de l'utilisateur à bannir	POST	Bannir un utilisateur : Renvoie un message de confirmation	La requête doit être authentifiée à l'aide d'un TOKEN, l'utilisateur associé étant un admin	Permet à un admin de bannir un utilisateur, celui ci ne pourra plus se connecter

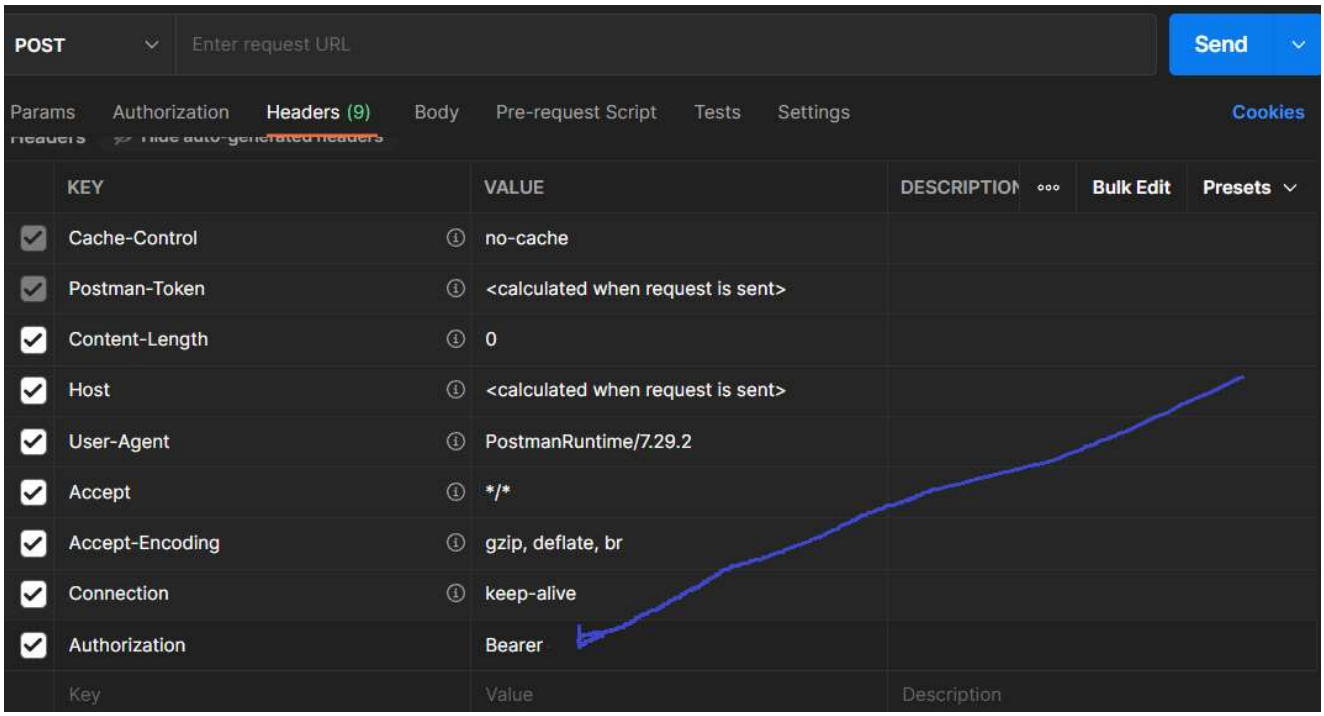
Endpoint	Paramètres	Méthode d'envoi	Données Retournées	Prérequis	Description
/users/debann	pseudo de l'utilisateur à débannir : 'username'	POST	Débannir un utilisateur : Renvoie un message de confirmation	La requête doit être authentifiée à l'aide d'un TOKEN, l'utilisateur associé étant un admin	Permet à un admin de débannir un utilisateur, celui ci pourra désormais se connecter
/users/makeadmin	pseudo de l'utilisateur à promouvoir 'username'	POST	Prmouvoir un utilisateur : Renvoie un message de confirmation	La requête doit être authentifiée à l'aide d'un TOKEN, l'utilisateur associé étant un admin	Permet à un admin de rendre un autre utilisateur admin , celui ci aura désormais les mêmes droits
/auth/discord	AUCUN	GET	Renvoie un token sur la route /auth/discord/callback pour valider l'authentification	AUCUN	Permet à un User de s'authentifier via Discord
/auth/google	AUCUN	GET	Renvoie un token sur la route /auth/google/callback pour valider l'authentification	AUCUN	Permet à un User de s'authentifier via Google

Qui est ADMIN ?

Pour créer un utilisateur admin, veuillez à mettre dans le champ **isAdmin** à true lors de la création de l'user
Par défaut, si vous ne fournissez pas le champ **isAdmin** lors de la création, il sera mis automatiquement à true, donc il sera admin

Envoi des TOKENS

Pour tester les routes nécessitant une authentification via TOKEN à travers POSTMAN, veuillez renseigner votre **token** en procédant comme suit:
Rendez-vous sur **headers**, creez un attribut de name : **Authorization**, et dans value, mettez : **Bearer votre token** sans oublier l'espace entre Bearer et le token.
A noter également que nous avons rendu possible d'envoyer le token via POST ou GET en créant un attribut de name **token** et pour value **votre token**

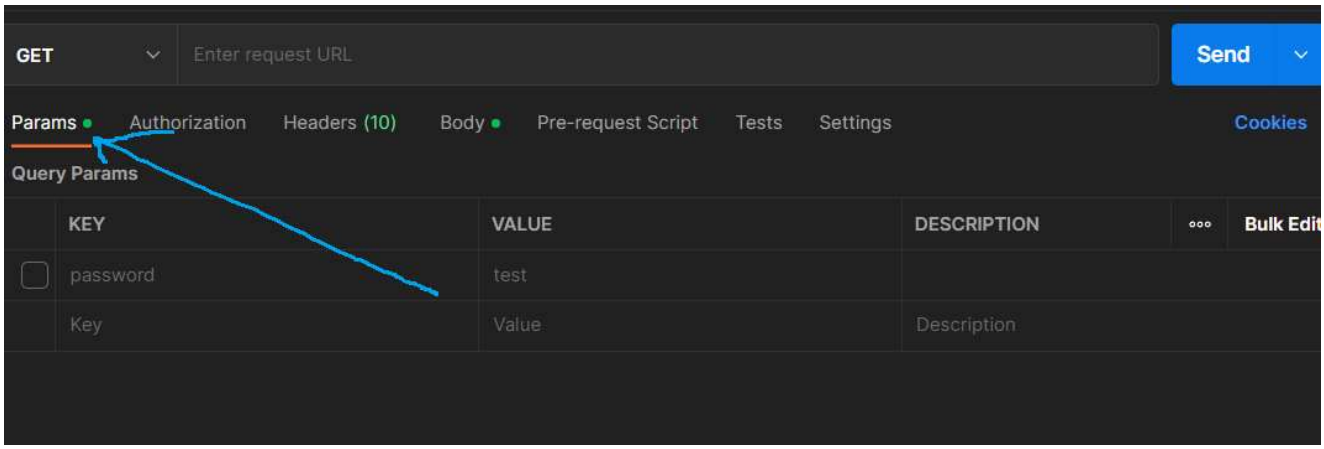


Utilisation des Méthodes GET,POST,DELETE avec POSTMAN

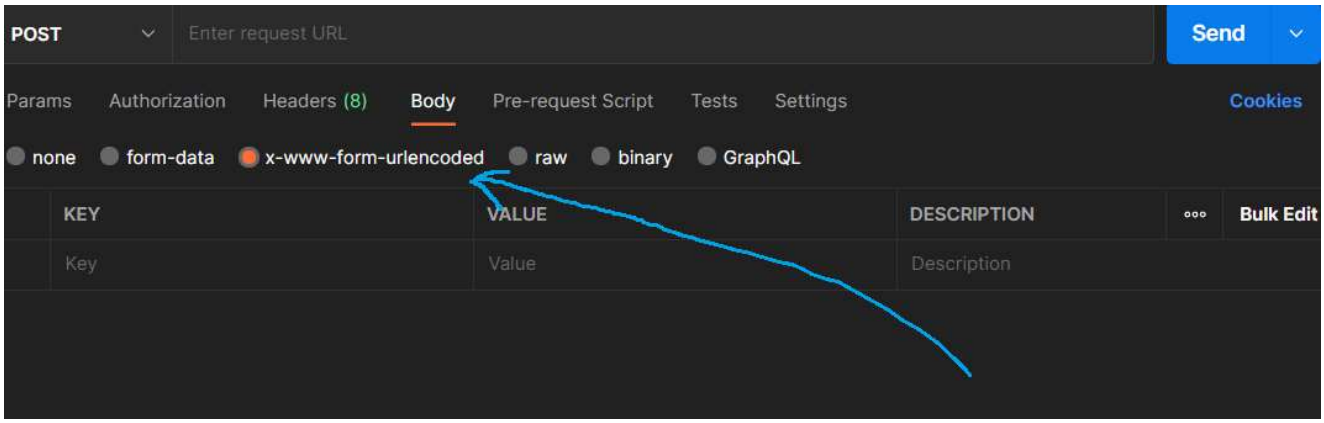
En utilisant POSTMAN pour l'envoi des données à l'API selon les différentes routes de l'API, veuillez respecter :

- **La nomenclature des paramètres**
- **Le nombre de paramètres à envoyer**

Pour les routes utilisant la méthode **GET,** les paramètres doivent être renseignés comme sur l'image suivante:



Pour les routes utilisant soit la **POST** ou **DELETE** les paramètres doivent être renseignés comme sur l'image suivante:



Toutes les fonctionnalités du backend ont été implémentées, une collection POSTMAN se trouve dans le dossier et voici un autre lien :
https://api.postman.com/collections/16866522-22e9529e-398b-4d57-a479-31de97678779?access_key=PMAT-01GNJ6TB9PAY1BDACXWHTNB1FW