Partie 7 Images

Tâches de l’activité

La partie que j'ai développée n'est pas entièrement fonctionnelle, car la vérification de l'extension de fichier ne fonctionne pas correctement. Bien que seule l'extension "jpg" soit prise en compte, il n'y a pas de vérification appropriée de cette extension.

Les fonctionnalités du service de gestion d'images

Le module exporte plusieurs fonctions pour effectuer des opérations CRUD sur un modèle d'image à l'aide de Mongoose.

Les fonctionnalités implémentées de ce code comprennent :

- getAllImages : cette fonction de contrôleur récupère toutes les images de la base de données et les renvoie au client sous forme de réponse JSON.
- getImage : Récupère une image par son ID et l'envoie au client sous forme de réponse JSON.
- createImage : Crée une nouvelle image en enregistrant les données de l'image envoyées dans le corps de la requête dans la base de données.
- updateImage : Met à jour une image existante par son ID en utilisant les données envoyées dans le corps de la requête.
- deleteImage : Supprime une image existante par son ID.
- getCompressedImage : Télécharge une image à partir de l'URL fournie dans le champ imageUrl d'un objet image, la compresse et la sert au client en tant que réponse.

L'implémentation utilise plusieurs modules Node.js :

- express : Framework web Node.js populaire qui simplifie le routage et la gestion des requêtes.
- mongoose : Outil de modélisation d'objets MongoDB qui fournit une solution basée sur le schéma pour modéliser les données d'application.
- sharp : Bibliothèque de traitement d'image Node.js qui fournit des capacités de manipulation d'image haute performance.

- request : Client HTTP simplifié pour Node.js qui fournit une interface facile à utiliser pour effectuer des requêtes HTTP.
- request-promise-native : Bibliothèque Node.js qui étend les fonctionnalités de request en retournant une Promesse au lieu d'un rappel.
- fs : Module Node.js qui fournit une API pour interagir avec le système de fichiers.
- axios : Client HTTP basé sur des Promesses pour Node.js qui fournit une interface facile à utiliser pour effectuer des requêtes HTTP.

La structure du code suit le modèle d'architecture MVC (Modèle-Vue-Contrôleur), où les contrôleurs définissent la logique d'application pour la gestion des images, le modèle définit le schéma pour la collection Image dans la base de données, et les routes sont définies dans un module séparé qui importe et utilise les contrôleurs.

En résumé, mon module fournit une solution simple et évolutive pour la gestion et le traitement des images dans une application Node.js en utilisant MongoDB et la bibliothèque de traitement d'images Sharp.

Gitlab

L'activité 7 que j'ai réalisée se trouve sur la branche GitLab "Mathéo". Cependant, j'ai choisi délibérément de ne pas inclure entièrement le code de notre projet de SAE sur cette branche, car je pense que cela faciliterait la compréhension et l'utilisation du code, aussi bien pour vous que pour les membres de mon groupe qui souhaiteraient l'analyser. Par conséquent, j'ai opté pour une séparation entre les deux, afin de rendre le code plus accessible et compréhensible pour tous.

Comment l’utiliser

Nous abordons maintenant la partie que je considère la plus intéressante, à savoir l'explication de l'utilisation.

Tout d'abord, je vous conseille de lancer la commande 'npm i'. Ensuite, assurez-vous que vous disposez d'une base de données vide ou contenant des images créées. Si vous n'avez pas déjà créé une base de données avec le nom 'Images', je vous

recommande de la créer. Si vous préférez utiliser une base de données, vous pouvez modifier la connexion dans le fichier 'serveur.js'.

```
mongoose.connect(process.env.MONGODB_URI || 'mongodb://127.0.0.1:27017/Images', {
  useNewUrlParser: true,
  useUnifiedTopology: true
});
```

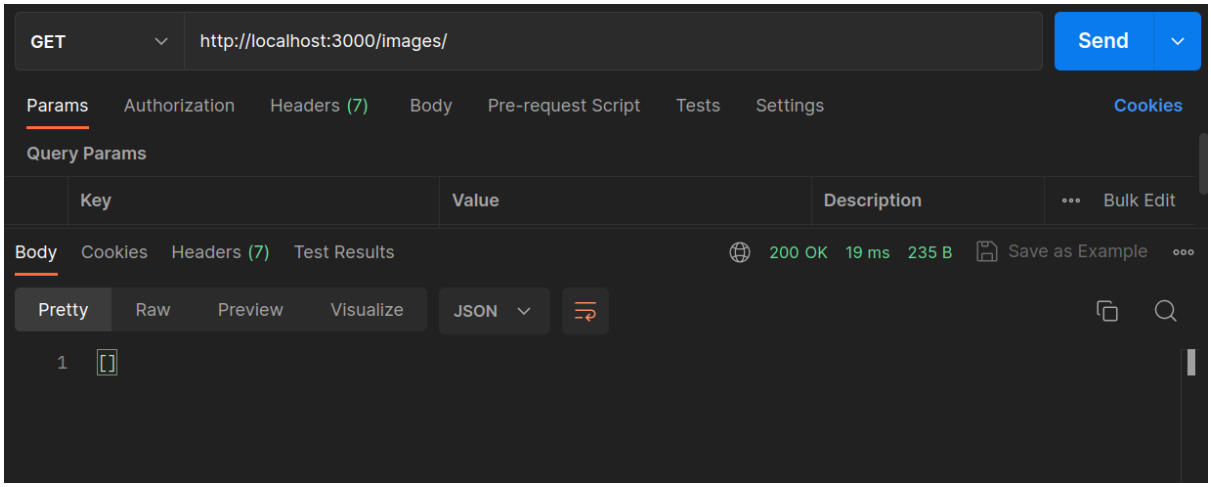
Une fois que vous avez créé la base de données, vous pouvez lancer le serveur en exécutant la commande 'nodemon serveur.js'. À ce stade, vous devriez voir apparaître ceci :

```
matheo@matheo-Latitude-3510:~/Bureau/NoSQLbb$ nodemon serveur.js
[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node serveur.js`
Server running on port 3000
Connexion réussie à la base de données.
```

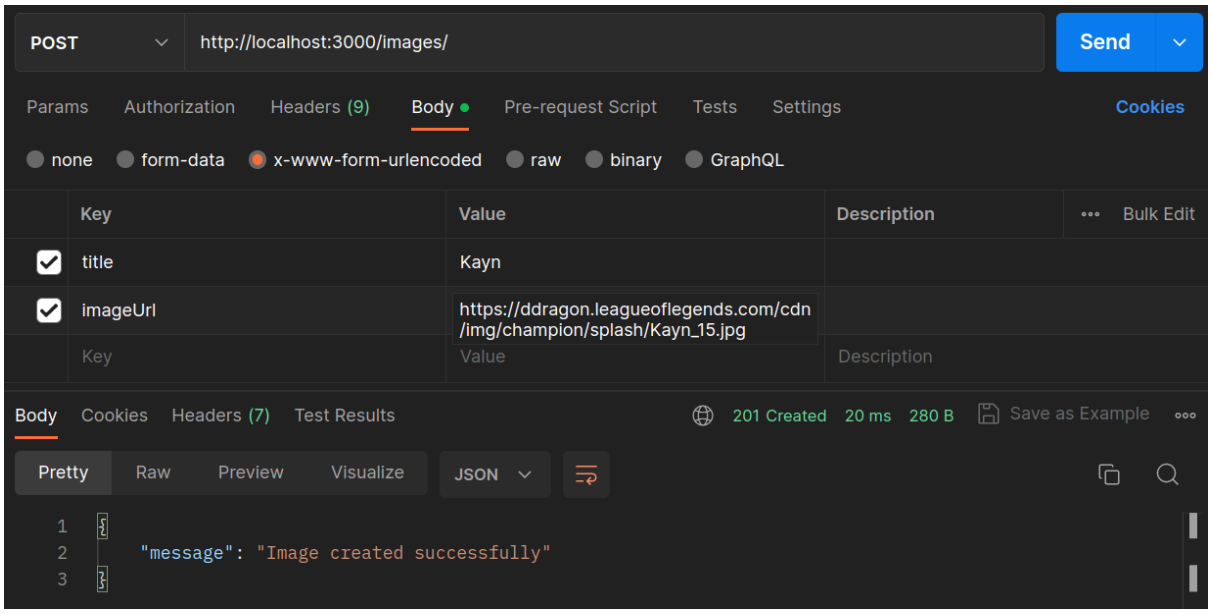
Maintenant, rendez-vous sur Postman.

Je vais vous fournir une liste de liens ainsi qu'un bref résumé de leur utilisation.

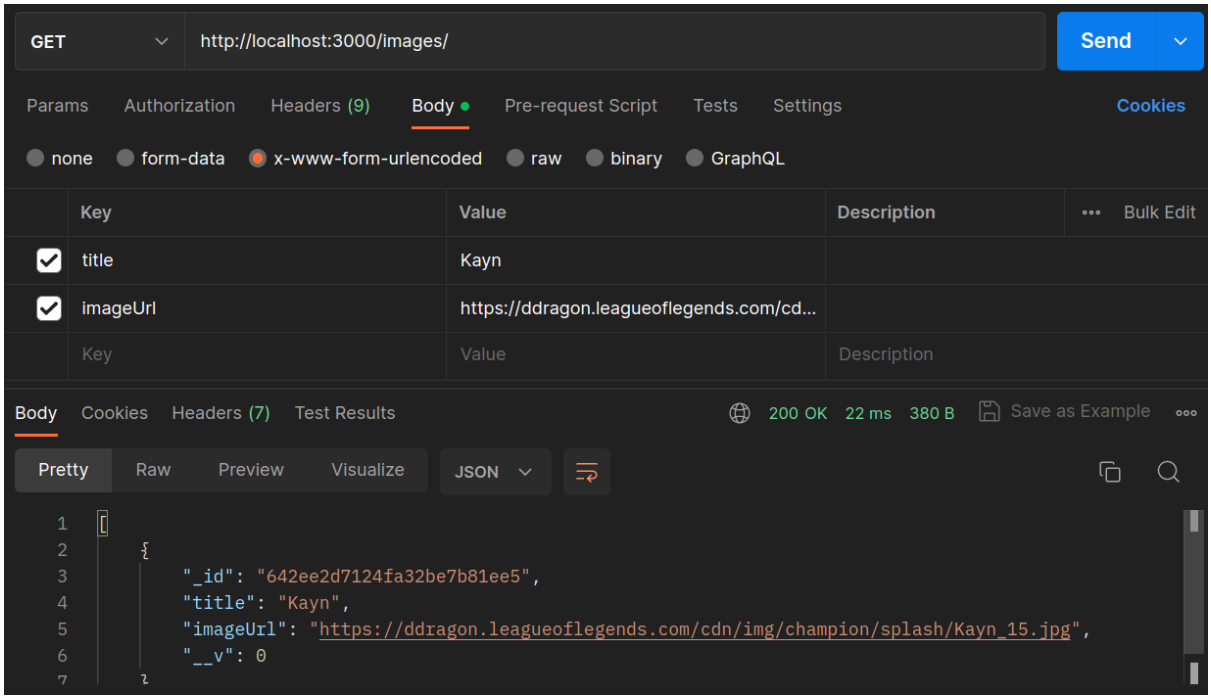
- `http://localhost:3000/images/` Ce lien renvoie toutes les images stockées dans la base de données. Pour cet exemple, j'ai utilisé une base de données vide, comme vous pouvez le constater.



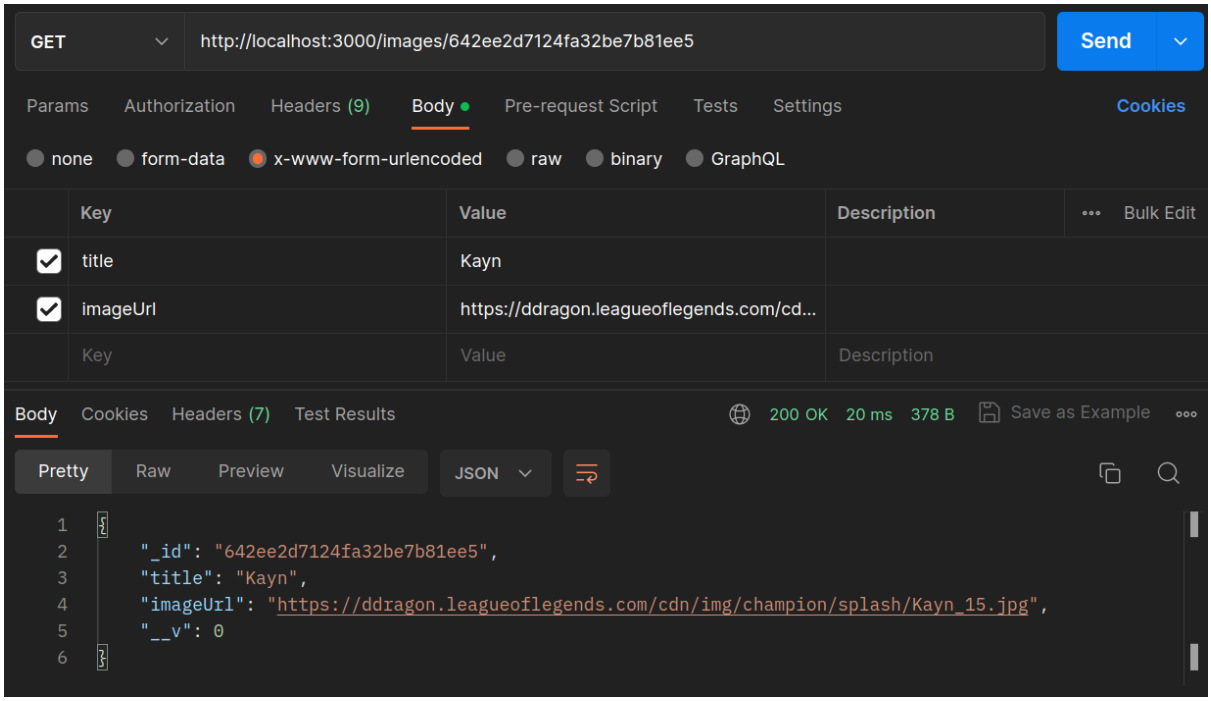
- Sous le même lien, cliquez sur 'Body', puis sur 'x-www-form-urlencoded'. Dans 'Key', saisissez 'title' et 'imageUrl'. Les valeurs que vous entrez sont de votre choix. N'oubliez pas de changer de 'GET' à 'POST'.



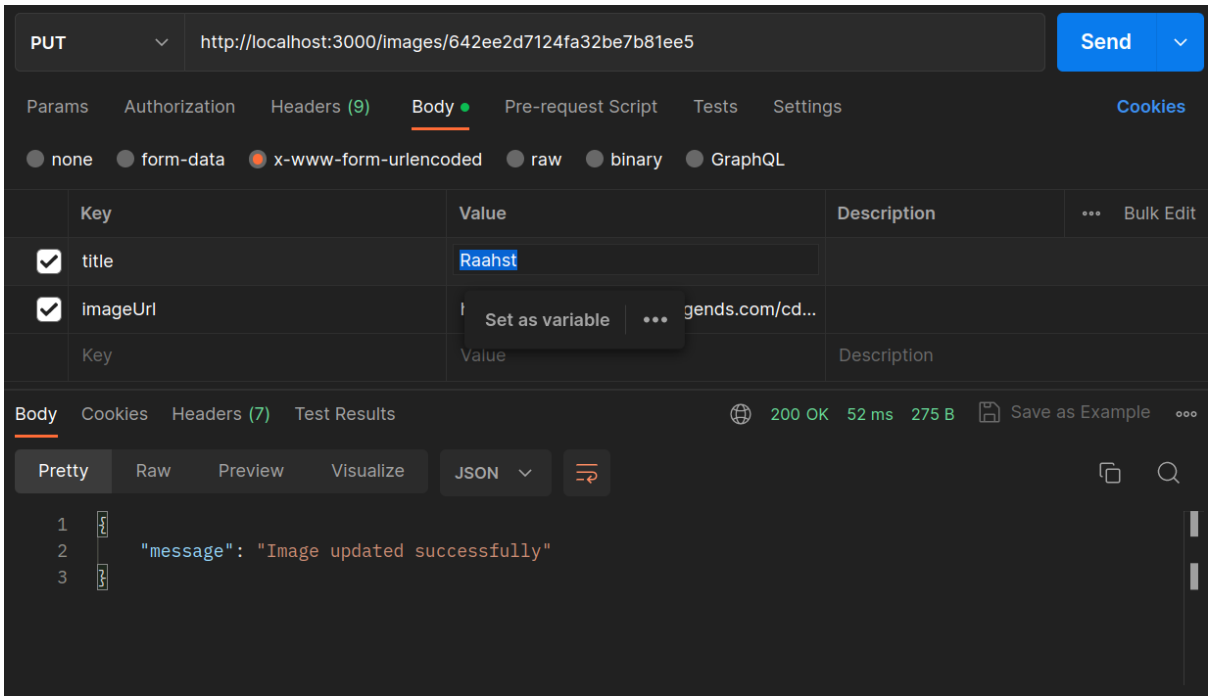
Maintenant, lorsque nous allons sur 'GET', nous pouvons constater que l'image a été créée avec un identifiant personnel et son titre respectif.



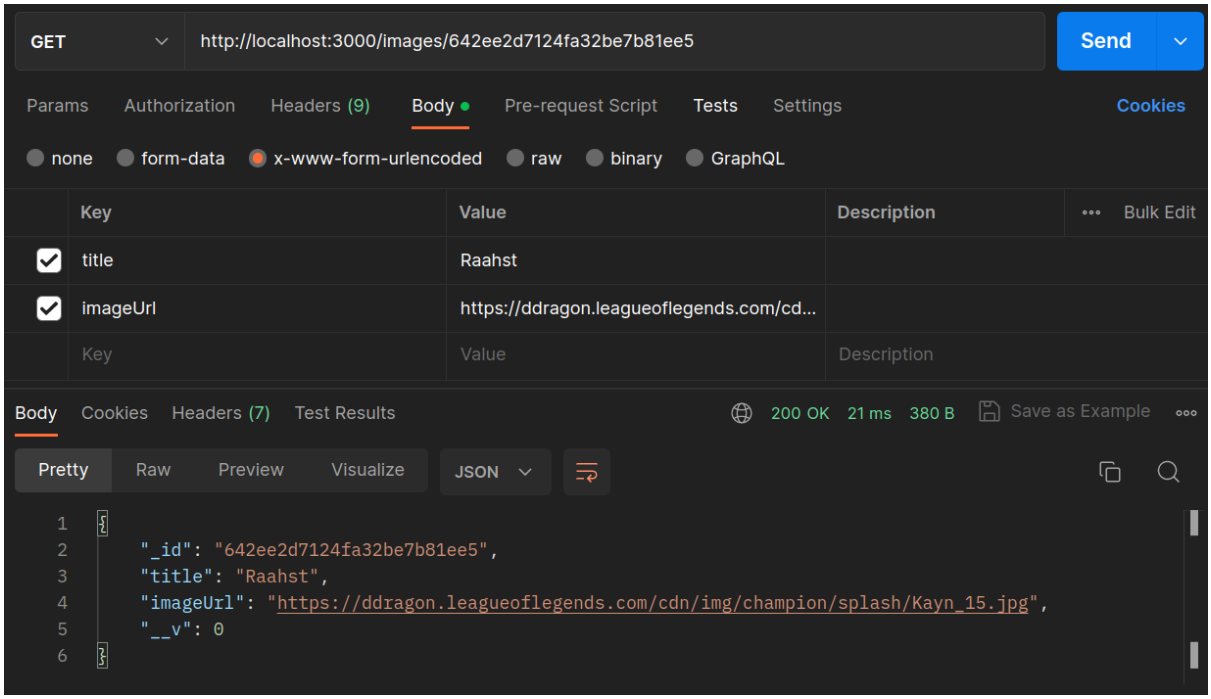
- `http://localhost:3000/images/id` Je tiens à préciser qu'il est possible de récupérer les informations d'une seule image en récupérant son identifiant et en le mettant dans l'URL, comme indiqué sur la capture d'écran suivante.



- Lorsque nous allons sur 'PUT', nous pouvons mettre à jour une image en modifiant son titre ou son URL à partir de son identifiant.

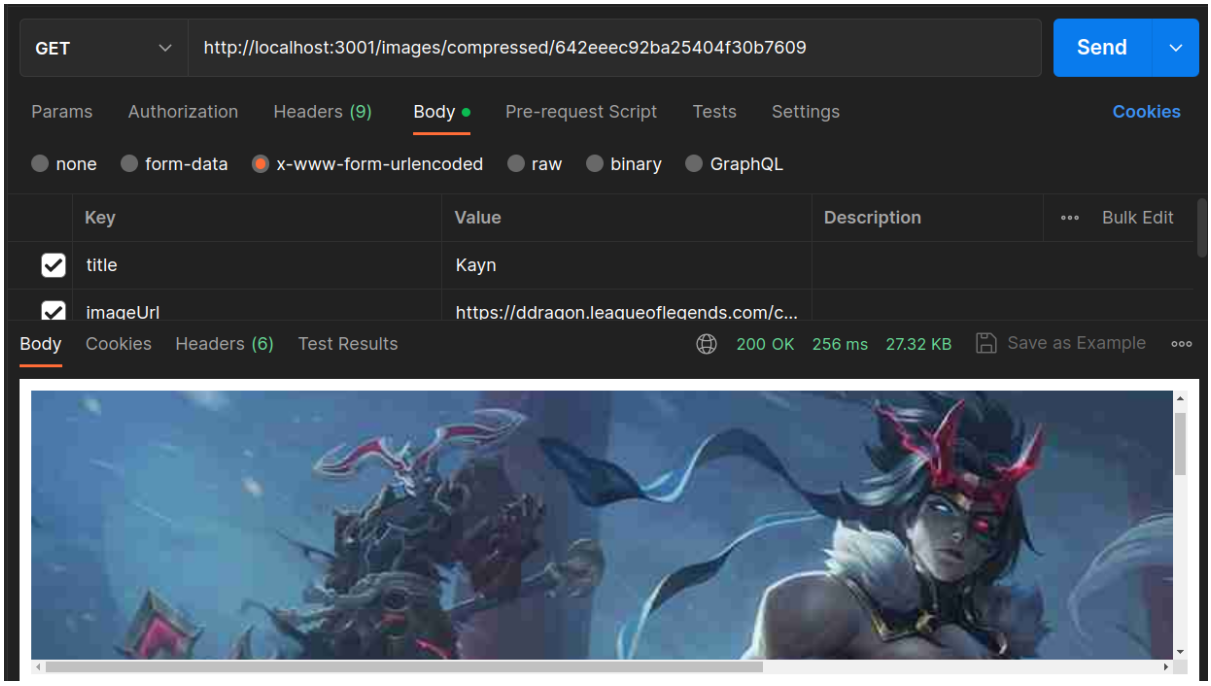


Ici, j'ai modifié le titre en "Raahst". Lorsque je vais sur 'GET', je peux constater que le titre a bien été modifié.

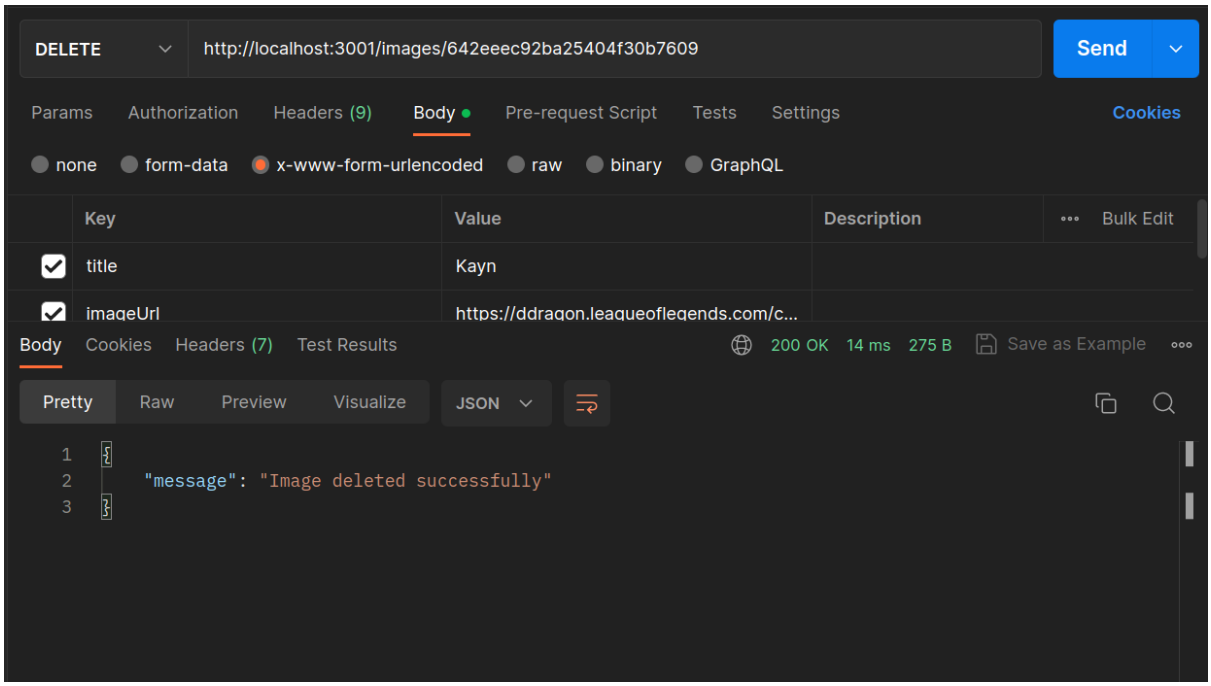


- `http://localhost:3000/images/compressed/id` Grâce à ce lien, l'image sélectionnée sera compressée. Les valeurs par défaut incluses sont, selon moi, suffisantes. Cependant, si vous souhaitez obtenir une image plus compressée, il faut se rendre dans le fichier 'imageController.js' et modifier la valeur de qualité.

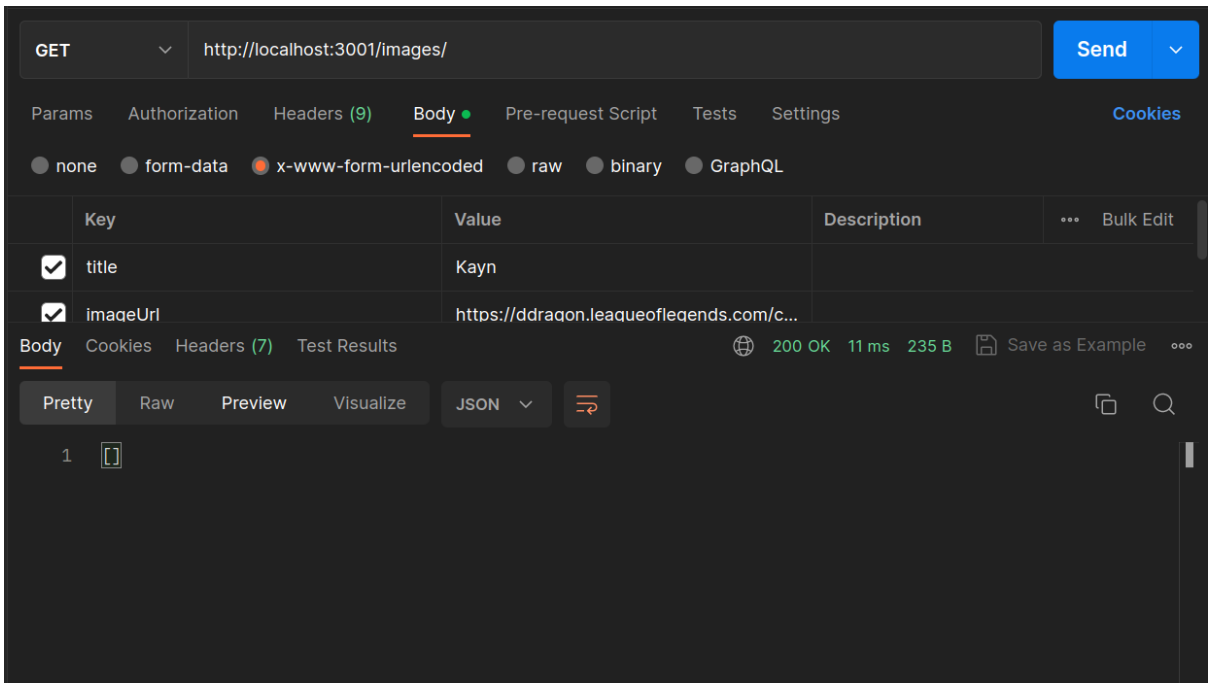
```
// Réduire la taille de l'image et enregistrer la version compressée avec sharp
await sharp(originalPath)
.jpeg({quality: 20})
.toFile(compressedPath);
```



- `http://localhost:3000/images/id` Finalement, nous pouvons supprimer une image en allant sur 'DELETE'.



Nous pouvons constater qu'elle n'existe plus.



Girard

Mathéo

S4-A3