

Rapport de Projet No SQL :

API de Gestion d'image pour le fimu

Contexte :

Nous développons un application pour le FIMU (site web et appli android), ils nous ont demandé de faire quelque chose pour la gestion de leurs images, ils nous donnent des liens et a partir de ces liens on peu obtenir une image, c'est donc à partir de là que j'ai commencé l'API de gestion d'image

Pour lancer l'API, il vous faut :

- Le fichier API image dans le github
- MongoDB
- NodeJS avec npm
- Postman pour pouvoir tester les requêtes

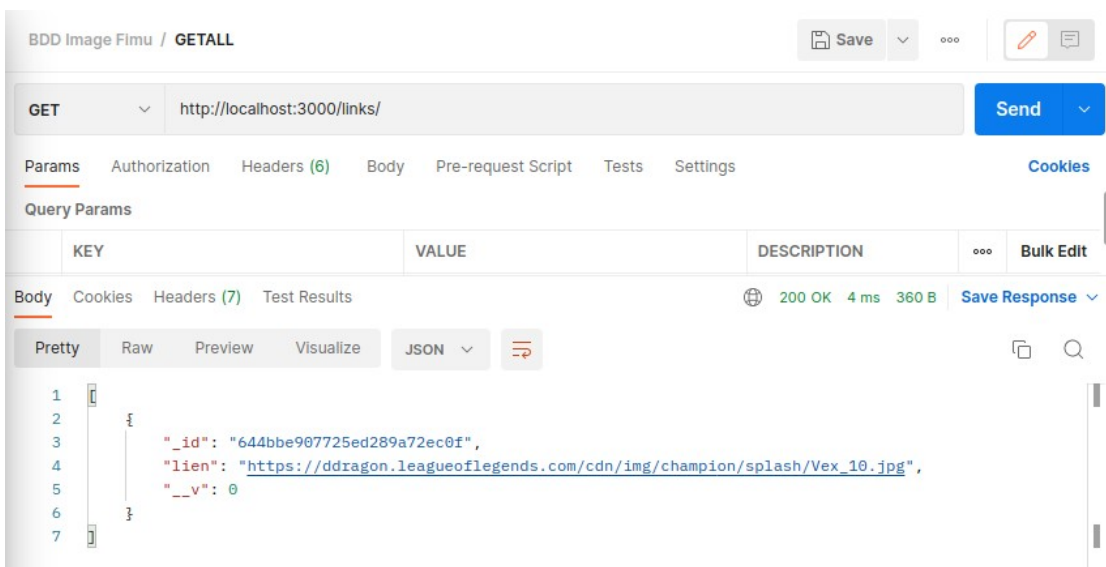
Dans le .env il faut que vous mettiez le lien de votre Base de Donnée, ensuite faites un npm i pour tout mettre à jour, et pour lancez l'API vous avez juste à faire un npm start.

Comment tester l'API ?

Je vous ai transmis un document un .json à mettre sur postman avec toute les routes dessus pour que vous puissiez tout tester directement

La Liste des Routes :

GETALL : Retourne tout les liens qui ont été transmis à la BDD.



CREATE : Cette route permet d'enregistrer le lien dans la BDD, il faut mettre votre liens dans le Body, le lien doit se terminer par le nom de l'image, car cela va enregistrer l'image directement dans l'API avec comme nom la fin du lien par exemple ce lien

https://ddragon.leagueoflegends.com/cdn/img/champion/splash/Vex_10.jpg, le nom de l'image sera donc **Vex_10.jpg**, cela enregistre donc l'image original qui par la suite est retouché grâce à sharp ([Doc sharp](#)), pour être compressé et miniaturisé, par la suite l'image original est supprimé.

Création d'une nouvelle image :

The screenshot shows the Postman interface for a POST request to `http://localhost:3000/links/`. The 'Body' tab is selected, showing a single key-value pair: 'lien' with the value 'https://ddragon.leagueoflegends.com/cdn...'. The response status is '200 OK' with a message 'Succès !'.

KEY	VALUE	DESCRIPTION
lien	https://ddragon.leagueoflegends.com/cdn...	

Body | Cookies | Headers (7) | Test Results | 200 OK | 358 ms | 235 B | Save Response

1 Succès !

Et si l'image existe déjà :

The screenshot shows the Postman interface for a POST request to `http://localhost:3000/links/`. The 'Body' tab is selected, showing a single key-value pair: 'lien' with the value 'https://ddragon.leagueoflegends.com/cdn...'. The response status is '500 Internal Server Error' with a message 'error: "existe déjà !"'. The response is displayed in JSON format.

KEY	VALUE	DESCRIPTION
lien	https://ddragon.leagueoflegends.com/cdn...	

Body | Cookies | Headers (7) | Test Results | 500 Internal Server Error | 6 ms | 281 B | Save Response

```
1 {
2   "error": "existe déjà !"
3 }
```

GETCOMPRESSED : Retourne l'image compressé de l'API, pour obtenir l'image il faut mettre directement le nom de l'image à la fin de lien par exemple http://localhost:3000/compressed/Vex_10.jpg

GET

http://localhost:3000/compressed/Vex_10.jpg

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

KEY	VALUE	DESCRIPTION
Key	Value	Description

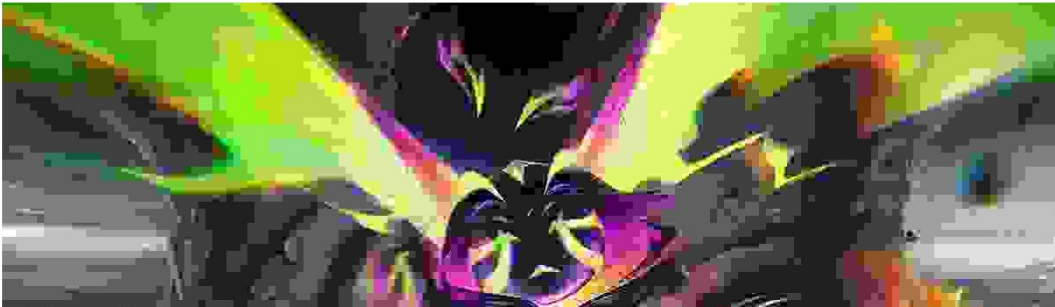
Body

Cookies

Headers (6)

Test Results

Status: 200 OK



GETTHUMBNAILS : Retourne l'image miniaturisé de L'API, pour obtenir l'image il faut mettre directement le nom de l'image à la fin de lien par exemple http://localhost:3000/thumbnails/Vex_10.jpg

GET

http://localhost:3000/thumbnails/Vex_10.jpg

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params


KEY	VALUE
Key	Value

Body

Cookies

Headers (6)

Test Results



DELETEALL : Supprime tout les liens dans la BDD et les images dans l'API.

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** http://localhost:3000/links/
- Params:** Authorization, Headers (6), Body, Pre-request Script, Tests, Settings
- Query Params:** A table with columns KEY and VALUE.
- Body:** Pretty, Raw, Preview, Visualize, JSON (selected), and a refresh icon.
- Response:** A JSON object: {"message": "Tous les liens ont été supprimés avec succès."}

DELETEONE : Supprime le lien et l'image allant avec le nom de l'image donné par exemple http://localhost:3000/links/Vex_10.jpg

Si l'image existe :

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** http://localhost:3000/links/Vex_10.jpg
- Params:** Authorization, Headers (6), Body, Pre-request Script, Tests, Settings
- Query Params:** A table with columns KEY and VALUE.
- Body:** Pretty, Raw, Preview, Visualize, JSON (selected), and a refresh icon.
- Response:** A JSON object: {"message": "L'image Vex_10.jpg a été supprimé avec succès."}

Si elle n'existe pas :

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** http://localhost:3000/links/Vex_10.jpg
- Params:** Authorization, Headers (6), Body, Pre-request Script, Tests, Settings
- Query Params:** A table with columns KEY and VALUE.
- Body:** Pretty, Raw, Preview, Visualize, JSON (selected), and a refresh icon.
- Response:** A JSON object: {"error": "L'image n'existe pas"}

Conclusion :

L'API d'image est parfaitement fonctionnelle. J'aurais pu créer une route pour obtenir une image plus floue, par exemple, car Sharp facilite grandement la retouche d'images et offre de nombreuses fonctionnalités. Cependant, pour le projet Fimu, ce n'est pas nécessaire, car je n'ai besoin que d'une image miniature et d'une image compressée. Dans l'API que je vous présente, j'ai beaucoup compressé l'image pour vous montrer que cela fonctionne, mais grâce à Sharp, je peux choisir le pourcentage de compression, etc. Sharp est un outil très utile pour créer une API qui permet de retoucher les images. Cependant, pour l'API d'image finale, je devrais encore ajuster le pourcentage de compression. Les tâches sont partiellement réalisées pour moi car mon API ne fonctionne qu'avec des images PNG ou des images JPG. Elle ne prend pas en charge tous les formats d'images.