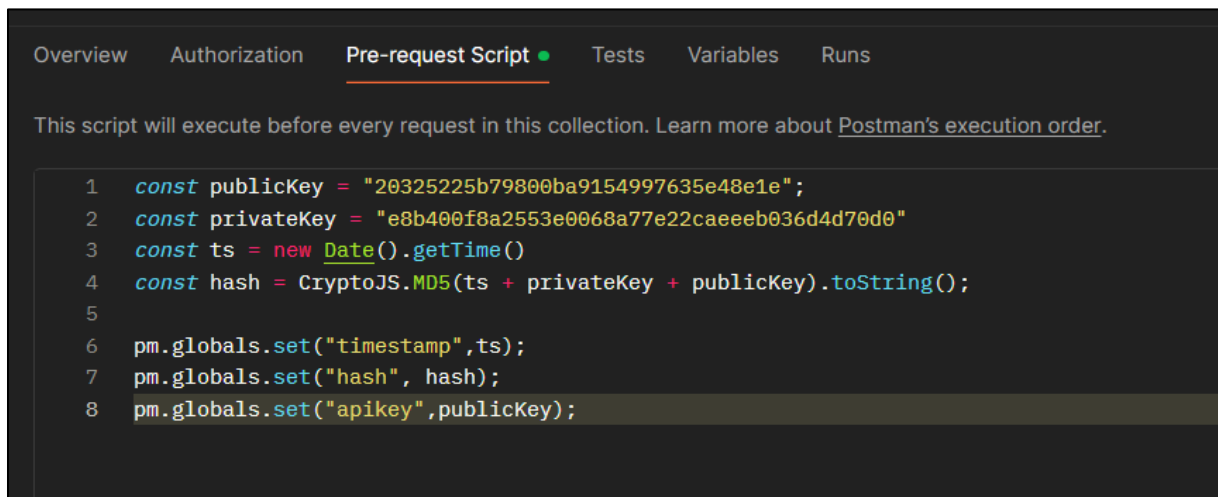


Etape 1 :

J'ai pu créer mon compte et tester l'API sur le site, tout marche correctement.

Etape 2 :

Il fallait utiliser l'API depuis Postman et utiliser le module « pre-request Script » pour générer le hash dans la requête de l'API.



```
1  const publicKey = "20325225b79800ba9154997635e48e1e";
2  const privateKey = "e8b400f8a2553e0068a77e22caeeeb036d4d70d0"
3  const ts = new Date().getTime()
4  const hash = CryptoJS.MD5(ts + privateKey + publicKey).toString();
5
6  pm.globals.set("timestamp",ts);
7  pm.globals.set("hash", hash);
8  pm.globals.set("apikey",publicKey);
```

Figure 1 : Pre-Request Script utilisé pour la requête.

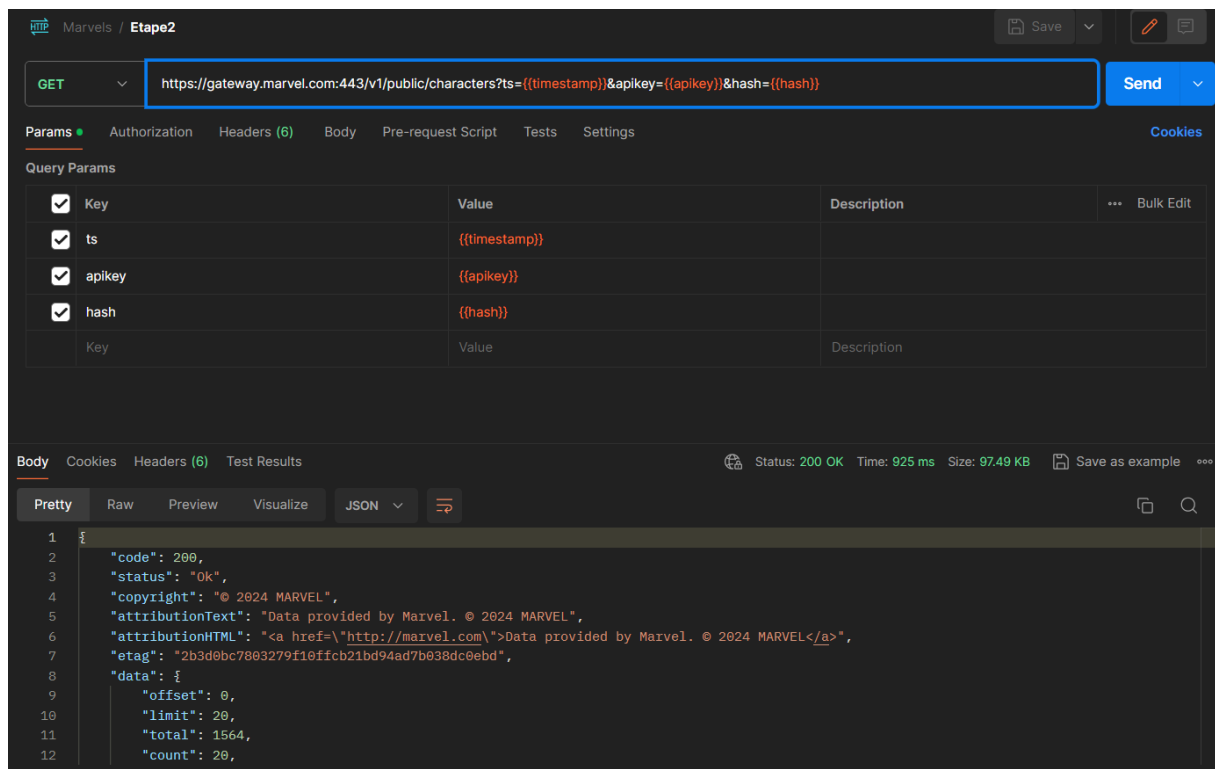


Figure 2 : Test de l'API Marvel sur Postman.

Grâce à ces deux captures d'écran, nous pouvons voir que la requête a bien généré une réponse, renvoyant les personnages Marvel.

Cette étape n'était pas difficile à faire, et j'ai pu apprendre à utiliser les « Pre-Request Script » que je ne connaissais pas.

Etape 3 :

Cette étape consistait à utiliser l'API depuis un projet NodeJS, elle m'a pris plus de temps que prévu à cause de la façon dont je calculais les timestamp, mais j'ai finalement réussi à la mettre en place avant la fin du cours.

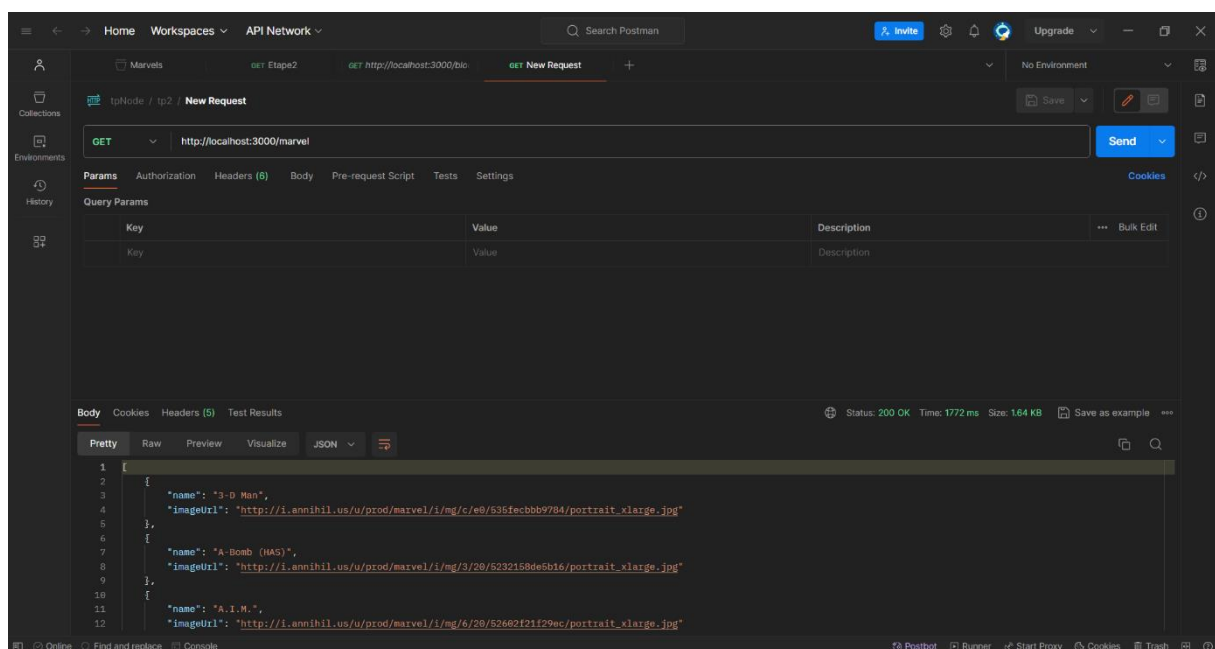


Figure 3 : Test de mon application depuis Postman.

Lorsque l'on fait une requête GET Marvel à mon serveur, on obtient bien la liste des personnages possédant une thumbnail valide en format portrait XL.

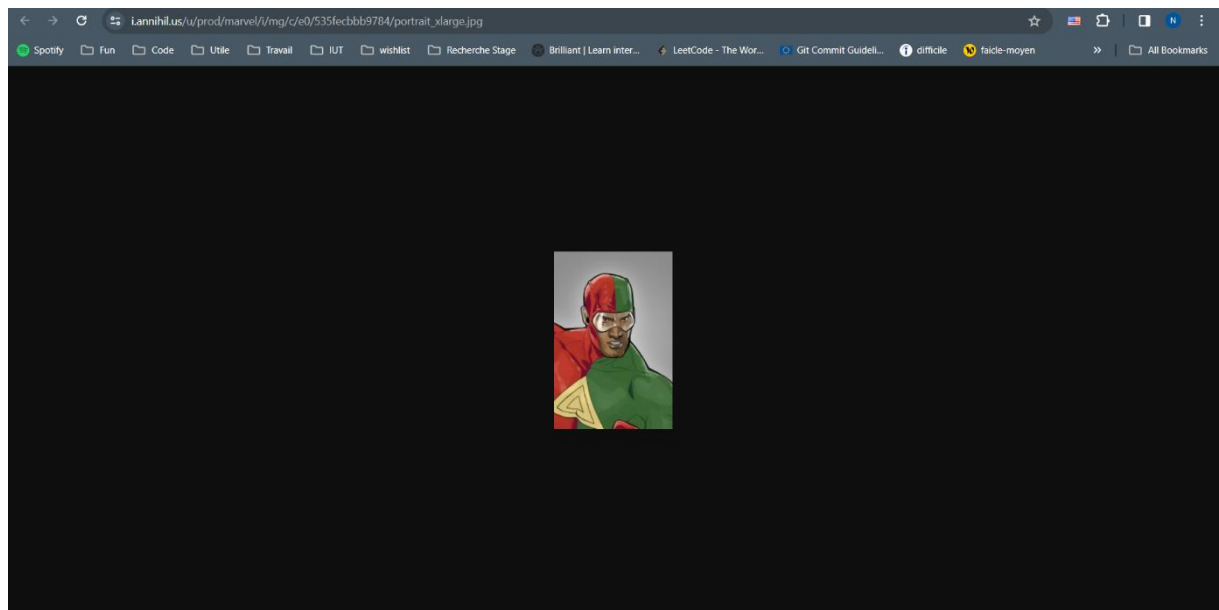


Figure 4 : Test de validité du champ imageUrl.

Voici le résultat lorsque l'on clique sur le champ imageUrl du premier personnage de la liste.

Etape 4 :

J'ai pris un peu de temps à comprendre comment marchait Fastify, mais j'ai réussi à mettre en place le système d'affichage avec les handlebars en me référant au cours. J'ai également changé le fonctionnement de l'URL de la fonction `appData(url)` car je n'avais pas bien compris comment était censé marcher la fonction.

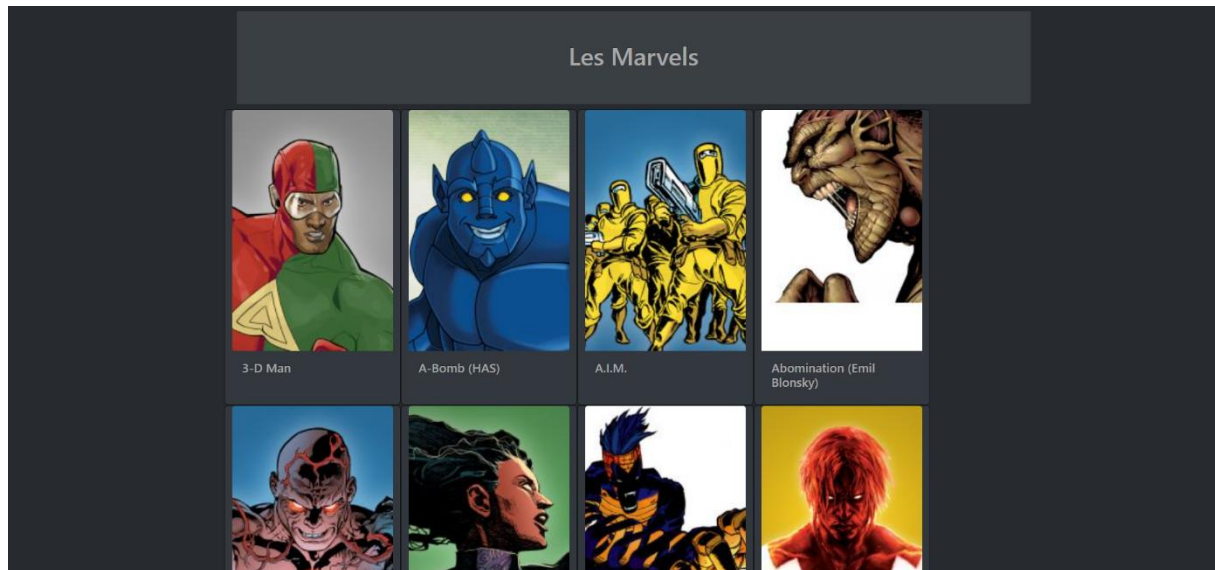


Figure 5 : Aperçu du serveur Fastify.

On peut voir sur cette capture d'écran, les personnages Marvel affichés sous forme de carte.

Etape 5 :

J'ai pris un peu plus de temps à mettre en place Docker, c'était la première fois que j'utilisais cet outil, mais j'ai pu comprendre rapidement comment il marchait

```
PS D:\Noe\Projets\Scolaires\NodeJS\tp2DevAvance> docker run -it -p 3000:3000 tp2v3

> lesmarvels@1.0.0 start
> node src/server.js

{"level":30,"time":1707039730828,"pid":24,"hostname":"7a6b596d3ddf","msg":"Server listening at http://0.0.0.0:3000"}
```

Figure 6 : lancement du serveur Fastify depuis docker.

Ce serveur était bien accessible, cependant le lien fournit par Fastify dans la console n'est pas valide, il faut utiliser <http://localhost:3000/>

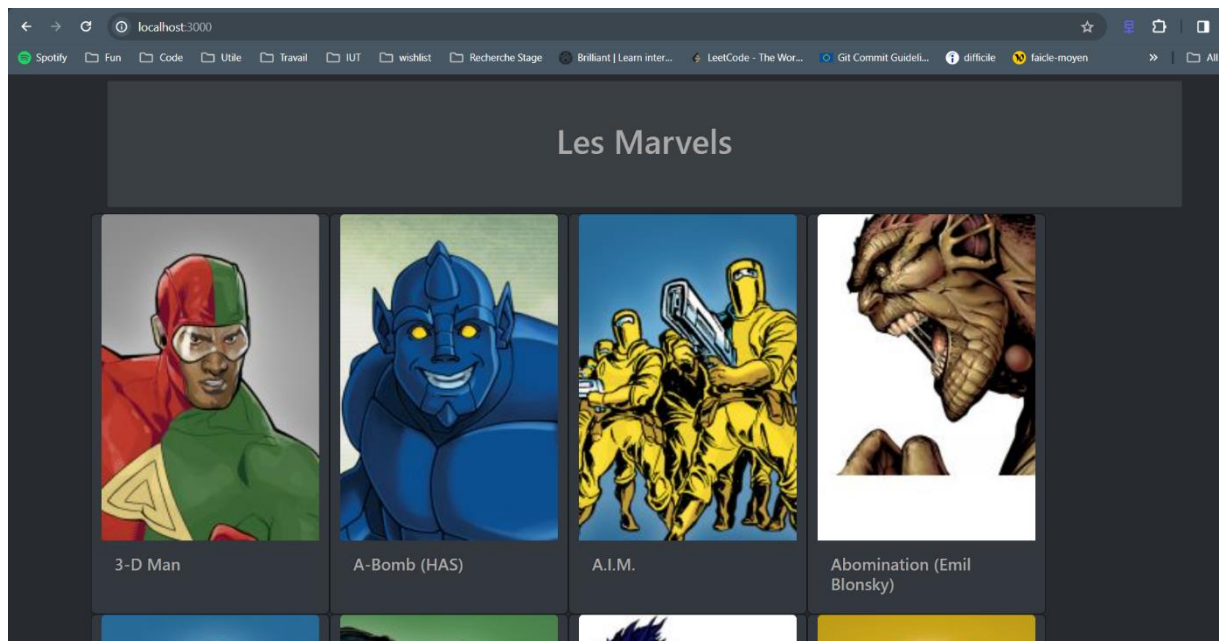


Figure 7 : Aperçu de <http://localhost:3000/>

Pour ce qui est des variables dans le fichier .env, je n'ai pas eu de mal à les mettre en place.

Je n'ai pas eu le temps de faire les étapes bonus.