**Rapport sur le TP2 Intégration de l'API Marvels**

L'utilisation d'une API REST offre la possibilité d'accéder à des données via une connexion HTTP. Dans le cadre de ce projet, nous avons exploité l'API de Marvels, qui permet d'accéder aux informations de sa base de données sur les Comics. Notre objectif était de créer une application web permettant de récupérer ces données, de les afficher et éventuellement de les manipuler.

La première étape consistait à s'inscrire sur le site de Marvels pour obtenir une clé d'API et explorer la première requête /v1/public/characters pour comprendre le format des données renvoyées par l'API.

Il n’y a pas eu de difficultés particulières il fallait juste comprendre comment fonctionne l'organisation des données renvoyées par l'API. Il fallait analyser le modèle de réponse et identifier les différents éléments.

La seconde étape visait à accéder aux résultats de manière distante en utilisant l'authentification fournie sur le compte. Nous avons utilisé Postman pour tester l’API et réaliser des requêtes.

Postman nous a permis d’avoir un affichage de ce que rendait l’api une fois les requêtes faites. Les difficultés rencontré ont était dans la syntaxe car une plusieurs petites erreurs ont été faite ce qui empêchait le pré-script de fonctionnait.

Une image contenant logiciel, texte, Logiciel multimédia, capture d’écran

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran

Description générée automatiquement

Ensuite **l’étape 3** le but était de refaire les requêtes qui ont été réaliser sur Postman donc en récupérant le squelette du code j’ai codé les 2 fonctions getData() et getHash

La fonction getData est une fonction asynchrone qui permet de récupèrer les données à partir de URL en utilisant l'API Marvels. Je commence par créer le hash grâce à la fonction crypto.createHash() du module crypto. Elle est crée grâce au timestamp, la clé privée et la clé public fournie par le site marvel puis ajouter à l’url de l’API . Ensuite grâce à fetch j’envoie un get à l’url. Ensuite Si la réponse contient des données valides (responseData.data.results), la fonction extrait ces données et les stocke dans responseResults. Apres la fonction filtre les résultats pour ne garder que ceux qui ont une image thumbnail valide. Les résultats sont stockés dans responseWithThumbnail. Et pour chaque personnage retenu, la fonction crée un objet newCharacter contenant les mêmes attributs, mais avec une URL d'image formatée pour la taille portrait\_xlarge. Les objets caractères ainsi créés sont renvoyés sous forme de tableau.

La fonction getHash est une fonction asynchrone qui génère un hash MD5 à partir des clés publiques et privées ainsi que du timestamp fourni. La fonction utilise la méthode crypto.createHash() du module crypto pour créer un hash MD5. Elle prend en compte le timestamp, la clé privée et la clé publique et les combine pour former une chaîne de caractères. Ensuite, j’utilise la méthode update pour alimenter l'objet de hachage avec la chaîne de caractères créée. Puis j’utilise la méthode digest("hex") pour obtenir la représentation hexadécimale du hash.  
  
J’ai pris beaucoup de temps sur cette étape car au lieu de mettre apikey j’avais mis publickey et le code ne générer pas d’erreur donc je ne comprenais pas pourquoi sa ne fonctionnait pas. J’ai pris aussi du temps à comprendre le résultat obtenue car en clickant sur le lien je ne voyais pas l’image mais sa : Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement  
Alors que je n’avais pas besoin de cliquait dessus car l’objet était déjà la

Une image contenant texte, capture d’écran, Police

Description générée automatiquement



**Etape 4**

Dans cette étape, j’utilise Fastify et son plugin @Fastify/view pour gérer le rendu visuel avec Handlebars. Donc après avoir installé tous les modules nécessaire, j’ai ensuite suivi les instructions pour réaliser cette étape. J’ai rencontré des difficultés à comprendre comment on utilise fastify et handlebars.

**Etape 5**

Pour cette étape j’ai vraiment juste suivi les instruction de l’énoncé et fais des recherches sur l’utilisation de docker desktop. Je n’est pas rencontrés de difficultés particulières.

**Récupération des données de l'endpoint avec authentification**: La fonction getData utilise fetch pour accéder à l'endpoint en utilisant les identifiants publicKey et privateKey, et en calculant le hachage MD5 à l'aide de la fonction getHash.

1. Utilisation de node-fetch pour effectuer des requêtes HTTP asynchrones.
2. Utilisation de crypto.createHash() pour générer un hachage MD5 à partir de la clé publique, de la clé privée et du timestamp.
3. Gestion des erreurs lors de la récupération des données depuis l'API Marvel.
4. Traitement des données reçues pour filtrer les personnages avec des images valides.
5. Construction d'un tableau de personnages avec les URL des images en format portrait\_xlarge.