Rapport TP2

Babylone ISSHAK - 303

Etape 2 :

J'ai eu beaucoup de mal encore une fois avec Postman, je pensais qu'il fallait déclarer les variables d'environnnement dans les pré-requis de la requête (get/http://gateway.marvel.com/v1/public/characters?ts={{timestamp}}&apikey={{clePublic}}&hash={{hachage}}), or il fallait les positionner dans les pré-requis à la racine de notre collection Marvels Postman afin que ce soit un pré-requis pour toutes les requêtes de cette collection !
De plus, je ne savais pas ce que c'était que Timestamp, j'ai donc demandé à mon professeur ce que c'était et surtout à quoi cela servait, pourquoi l'api Marvel en a besoin ? J'ai donc compris que c'était la date et heure précise (date format ISO) qui est envoyée avec la requête afin de bien vérifier que la requête a bien été déjà envoyée. Car en effet, une personne malintentionnée pourrait copier la requête et prétendre que cela vient de nous, mais grâce à la date, ceux qui gèrent l'api marvels pourront vérifier si c'est bien nous ou non.
Pour le type de hachage en MD5, je me suis simplement aidée de la documentation sur devdocs et autres sites internet.
Enfin, pour avoir la syntaxe pour créer les variables d'environnement, Postman propose cela directement à l'aide d'un bouton "set an environment variable"

Etape 3 :

Pour la fonction getHash(...), je me suis souvenue de ce que l'on avait fait la semaine précédente, à savoir : l'utilisation de createHash() avec le module `node:crypto`
J'ai donc changer sha256 en md5 :

```
export const getHash = async (publicKey, privateKey, timestamp) : Promise<ArrayBuffer>  => {
    const hashInput = timestamp + privateKey + publicKey;
    return await createHash('md5').update(hashInput).digest( algorithm: 'hex');
}
```

Voici ce que cela me renvoyait : Promise { <pending> }, il n'y a donc pas d'erreurs, la fonction retourne bien une promesse.

Pour la fonction getData(url), c'est ici que j'ai appelé la fonction getHash sans mention await car il y est déjà dans le return de la fonction getHash. Ci-dessous la fonction getData :

```
export const getData = async (url) : Promise<json>  => {
    // A Compléter
    const hash : Promise<ArrayBuffer>  = getHash(publicKey, privateKey, Date.now());
    const response : Response  = await fetch( url: url + "?ts=" + Date.now() +"&apikey="+ publicKey +"&hash=" + hash);
    const data : Promise<unknown>  = response.json(); //sous forme de json
    console.log(data) // test
    return data;
}
```

Enfin, je n'avait pas compris cette partie de l'énoncé : ¨ Effectuer la sélection des résultats qui possèdent une image thumbnail valide. ¨ Constituer un tableau de Personnage contenant pour le champ imageUrl la destination de la version portrait_xlarge de l'image.", alors j'ai demandé à chatGPT de m'expliquer, j'ai donc compris à quoi servait thumbnail. En JavaScript, quand une case/ objet d'une liste est absent, son accès peut entraîner des erreurs, en particulier si elle est utilisée dans une condition ou une opération. La vérification character.thumbnail && character.thumbnail.path permet d'éviter ces erreurs quand on les utilise à l'aide d'une condition if. Voici donc ci-dessous la fonction qui permet ce filtrage pour exclure les personnages qui n'ont pas d'image ou dont le chemin de l'image est "image_not_available" :

```
/**
 * Filtrer les personnages sans visuel
 * @param {Array} characters - Tableau de personnages
 * @return {Array} - Tableau filtré de personnages avec visuel
 */
1+ usages
const filterCharacters = (characters : (any)[] ) : (any)[]  => {
    return characters.filter(character => character.thumbnail && character.thumbnail.path !== "image_not_available");
};
```

Il faut ensuite appeler cette méthode lors du retour de getData, mais attention, pas n'importe comment ! `return filterCharacters(data.data.results);`
"data.result" est souvent utilisé par convention pour les API Rest afin d'obtenir le résultat de la requête envoyée (ici, à Marvel).

Etape 4 :

La première étape est simple, il faut juste installer fastify en faisant un npm : npm install fastify @fastify/view handlebars
Je n'avais pas compris ce que c'était que les partials, alors j'ai demandé à chatGPT et je me suis rendu compte que cela ressemblait aux configurations du tp précédent avec le port que l'on doit indiquer etc…

J'ai bloqué pendant quelques heures car l'api me renvoyait toujours "InvalidCredentials : APIKey is invalid", tout cela pour trouver l'erreur "toute bête" : dans la méthode getData, lors du fetch des données, je lui passe une URL et les paramètres qui vont avec (timestamp, clé publique, et hachage). Cependant, j'avais utilisé des doubles ou simples guillemets, or, il fallait utiliser des backticks → `` : `fetch(url + `?ts=${date}&apikey=${publicKey}&hash=${hash}`)`
En javascript, lorsque je mets des guillemets (simples ou doubles) et que j'appelle les variables à l'intérieur de celles-ci, cela crée une chaîne littérale, et les variables à l'intérieur ne sont pas interpolées correctement. Donc pour bien s'assurer que les valeurs sont des chaînes de caractères, il faut utiliser des backticks.
De plus, une autre erreur qui m'a pris du temps est que sur la page localhost:3000, rien ne s'affichait, il fallait juste que je redéclare le tableau de characters (personnages marvels) en appelant une nouvelle fois la fonction getData().

Une fois le résultat affiché, j’observe que tous les personnages ayant des images sont bien affichés, cependant, je ne sais pas pourquoi elles ne chargent pas. Voici ce que j'obtiens :

Etape 5 :

Je ne comprenait pas comment utiliser Docker avec l’énoncé, alors afin de mieux comprendre, voici une page qui m’a beaucoup aidé et dont je me suis surtout inspirée pour créer le fichier Dockerfile : https://openclassrooms.com/fr/courses/2035766-optimisez-votre-deploiement-en-creant-des-conteneurs-avec-docker/6211517-creez-votre-premier-dockerfile.
Pour l’instant j’ai juste pu remplir le fichier Dockerfile et .dockerignore. Cependant, je fais partie de celles et ceux qui ne peuvent pas déplacer leur ordinateurs à l’IUT, donc je dois utiliser la technique de la VM stockée dans une clé USB, or je n’ai absolument rien compris aux instructions d’installation et d’utilisation.
Donc pour la prochaine séance, je vais demander comment utiliser la VM sur clé, et aussi comment utiliser Dockerfile puisque j’ai “compris” sur le principe, mais je n’ai pas eu l’occasion d’appliquer la théorie étant donné que je n’ai pas compris la méthode de la VM sur clé.