TP2 Dev. Avancé

Etienne Kita 301

Étape 2 :

Ce fût légèrement compliqué de découvrir à la fois le fonctionnement des collections Postman avec leur script de pré-requête et leurs variables de collections ainsi que de s'adapter au système d'authentification de l'api de marvel.com. Une fois tout ça comprit, j'ai enregistré les clés en variables de collections, puis dans le script de pré requête, je calcule le hash et je l'ajoute en paramètre.

```
const publicKey = pm.collectionVariables.get('publicKey');
const privateKey = pm.collectionVariables.get('privateKey');
const ts = new Date().getTime().toString();

const hash = CryptoJS.MD5(""+ ts + privateKey + publicKey).toString();

pm.request.url.addQueryParams([
    { key: 'apikey', value: publicKey },
    { key: 'ts', value: ts },
    { key: 'hash', value: hash }
]);
```

Étape 3 :

Pour la fonction *getHash,*, j'ai pu utiliser le même principe que celui utilisé dans le TP1, sans difficulté.

Pour la fonction *getData*, j'ai pensé à vérifier si l'URL donnée contient déjà des paramètres, pour savoir comment rajouter les paramètres d'authentification. Une fois la requête effectuée, pour enlever les héros sans image, je filtre ceux qui n'ont pas "image_not_available" dans le chemin de leur image, observé depuis l'interface de test de l'api.

Pour ceux qui ont une image valide, concaténer le chemin et l'extension n'est pas compliqué, en revanche, je n'ai pas compris la partie concernant la taille de l'image.

Étape 4 :

Mettre en place FastifyView et HandleBars m'a pris du temps et de nombreuses erreurs, mais finalement, l'utilisation de HandleBars est simple d'utilisation. J'ai pu réunir les *partials* et créer la liste de héros via les données de l'API.

Étape 5 :

Je me suis arrêté à l'étape 4 et je n'ai pas réussi l'étape 5.