Développement Avancé

TP2 – Les Marvels – Wania Jean-Pierre (302)

Thèmes

- Accès API REST distante
- Le module Fastify
- Le moteur de template Handlebars
- Dockerisation d'une application

Étape 1 – « Avengers, rassemblement ! »

Pour cette étape j'ai créé un compte sur https://developer.marvels.com/, ce qui m'a permis d'obtenir une clé publique et une privée afin de faire des requêtes sur leur API. J'ai ensuite utilisé leur Interactive Documentation, pour connaître la structure de la réponse, et aussi pour savoir comment utiliser leur API, voilà deux exemples de ce que j'ai pu créer :

```
https://gateway.marvel.com:443/v1/public/characters?apikey=e4310f5363438e771574a562327bf668
```

```
https://gateway.marvel.com:443/v1/public/characters?name=mm&apikey=e4310f5363438e771574a562327bf668
```

Avec ces appels tests, j'ai constaté qu'il est nécessaire d'avoir une apikey, peu importe la demande que je fais. Je peux aussi ajouter d'autres paramètres tel que le nom *(name)* ou la limite *(limit)*.

Étape 2 – Toujours plus loin

Dans cette étape, j'ai créé une collection dans Postman afin de pouvoir utiliser le système de pré-requête. J'ai donc créé une pré-requête *(pre-request)*, qui me permet de définir les valeurs des variables d'environnement de ma collection , maintenant et hachage, qui correspondent respectivement au timestamp et au hachage du timestamp, de la clé privée et de la clé publique.

```
1  const publicKey = "e4310f5363438e771574a562327bf668"
2  const privateKey = "25a23b65a25ed01647bf6f5000f26296ccb6e1ec"
3
4  const ts = Date.now();
5  const hashp = CryptoJS.MD5(ts+privateKey+publicKey).toString();
6
7  pm.environment.set("maintenant", ts);
8  pm.environment.set("hachage", hashp);
```

Les variables sont ensuite utilisées comme paramètre dans la requête — il aurait été intéressant de créer une variable pour le paramètre apikey :

Query Params

	Key	Value	Description	••• Bulk Edit
☑	apikey	e4310f5363438e771574a562327bf668		
☑	ts	{{maintenant}}		
☑	hash	{{hachage}}		
	Key	Value	Description	

Finalement, en envoyant la requête complétée, on obtient au retour, le même retour qu'avec l'Interactive Documentation :


```
{
    "code": 200,
    "status": "Ok",
    "copyright": "© 2024 MARVEL",
    "attributionText": "Data provided by Marvel. © 2024 MARVEL",
    "attributionHTML": "<a href=\"http://marvel.com\">Data provided by Marvel. © 2024 MARVEL</a>",
    "etag": "2b3d0bc7803279f10ffcb21bd94ad7b038dc0ebd",
    "data": {
        "offset": 0,
        "limit": 20,
        "total": 1564,
        "count": 20,
        "results": [
            {
                "id": 1011334,
                "name": "3-D Man",
```


Difficultés

Sur cette partie, j'ai eu quelques difficultés. Elles sont principalement liées à des manques d'attention. Par exemple, l'oublie du toString pour le hachage :

```
1  const publicKey = "e4310f5363438e771574a562327bf668"
2  const privateKey = "25a23b65a25ed01647bf6f5000f26296ccb6e1ec"
3
4  const ts = Date.now();
5  const hashp = CryptoJS.MD5(ts+privateKey+publicKey);
6
7  pm.environment.set("maintenant", ts);
8  pm.environment.set("hachage", hashp);
```

Ce qui entraîne une erreur au niveau de la requête, et donc une réponse indiquant que les paramètres sont invalides.

```
1  {
2      "code": "InvalidCredentials",
3      "message": "That hash, timestamp and key combination is invalid."
4  }
```

J'ai perdu énormément de temps à cause de ce type d'erreur très simple à régler, mais plus difficile à trouver lorsque je ne fais pas attention.

Étape 3 – Tu veux ma photo ?

L'étape 3 consiste à reproduire ce que j'ai fait sur Postman, mais cette fois-ci en code. Pour ce faire j'ai complété la fonction getHash(publicKey, privateKey, timestamp) qui permet d'obtenir le hash. Puis je me suis occupée de getData(url), en essayant d'utiliser une structure pour les paramètres, ce qui n'a marché puisque l'API renvoie une erreur :

```
const param = {apikey: publicKey, ts: ts, hash : hash};
const response = await fetch(url, {
   method: 'get',
   param: JSON.stringify(param),
   headers: {'Content-Type': 'application/json'}
});
```

J'ai donc fait quelque chose de plus simple, qui cette fois-ci a marché :

```
const response = await
fetch(url+"apikey="+publicKey+"&ts="+ts+"&hash="+hash);
```

Ensuite, j'ai procédé au filtrage des résultats, en ne récupérant que les personnages qui ont un thumbnail, pour ce faire j'ai utilisé la méthode filter() pour ne pas avoir à créer une boucle — j'aurais dû nommer la variable data différemment :

```
const charactersWithImages = data.data.results.filter(character =>
   character.thumbnail &&
!character.thumbnail.path.includes("image_not_available")
);
```

Finalement, je crée le tableau personnage, en utilisant cette fois-ci la méthode map, toujours ne pas créer de boucle. Je récupère le nom et la description du personnage, puis construit l'URL de l'image en me basant sur celui pré-existant:

```
const characters = charactersWithImages.map(character => ({
   name: character.name,
   description: character.description,
   imageUrl:
`${character.thumbnail.path}/portrait_xlarge.${character.thumbnail
.extension}`
}));
```

Étape 4 – On s'accroche au guidon

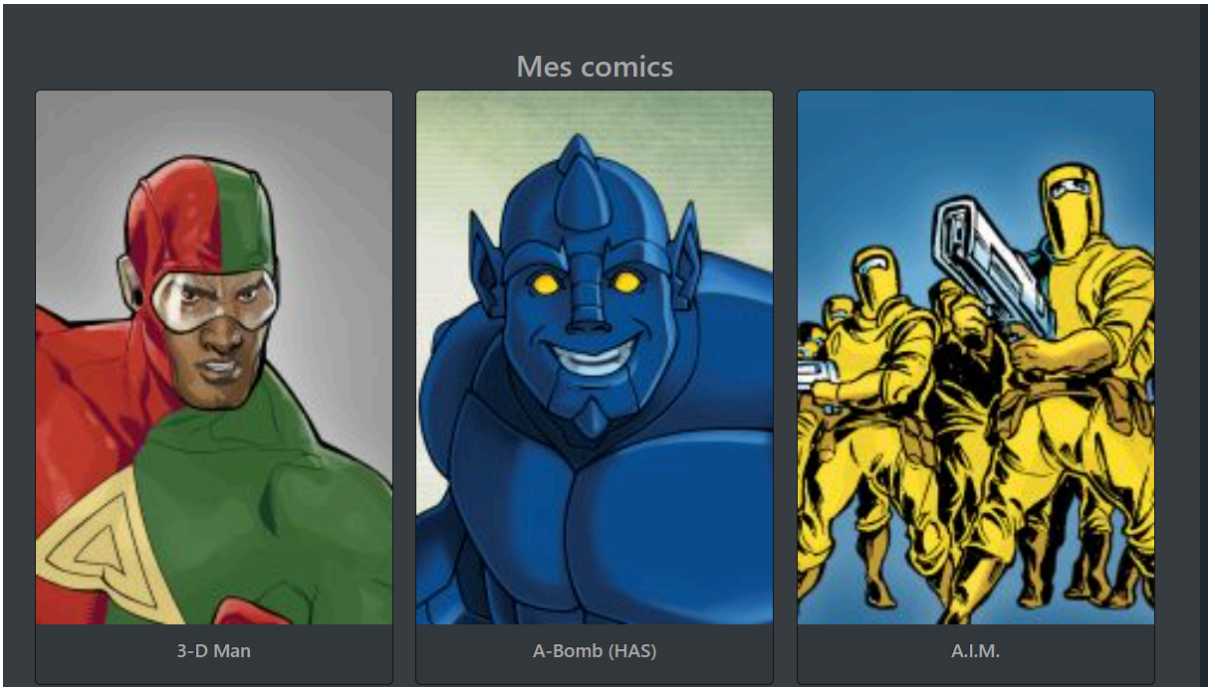
Durant cette étape, j'ai créé la présentation pour la page web. Pour ce faire, j'ai utilisé Fastify et Handlebars.

Après avoir récupéré l'instance Fastify, j'ai enregistré les templates handlebars, pour pouvoir les utiliser sur la page d'affichage :

```
app.register(fastifyView, {
   engine: { handlebars },
```

```
    templates: 'templates',
    includeViewExtension: true,
    options: {
        partials: {
            header: 'header.hbs',
            footer: 'footer.hbs'
        }
    }
});
```

J'ai ensuite complété index.hbs pour ajouter l'utilisation des partials header et footer. J'ai ensuite enregistré une route, pour accéder à index.hbs qui s'occupe de l'affichage des données. Ce qui m'a permis d'obtenir ce résultat.



Étape 5 – C'est dans la boite !

Après avoir créé, le Dockerfile et le .dockerignore, et complétée ces derniers en suivant les consignes, j'ai essayé de lancer la construction de l'image. Toutefois, ça n'a pas fonctionné, après 3 jours à essayer, jei me suis résignée et j'ai abandonné.