Florent Lelion - 302 01/02/2024

Compte rendu - Sujet 2
Les Marvels

Étape 1 - « Avengers, rassemblement ! »

Après avoir suivi toutes les instructions de l'énoncé, j'ai pu observer l'organisation de la réponse afin de repérer les champs les plus intéressants pour moi. J'ai exécuté la requête directement et j'ai obtenu l'url de requête suivante https://gateway.marvel.com:443/v1/public/characters?apikey=14133da1d6de221028f7dc592c734233 ainsi que le corps de la réponse des plusieurs Marvel à la suite (et le code de réponse et le header de la réponse)..
Par la suite, j'ai ajouté dans le champs name le Marvel **Iron Man** et quand j'ai exécuté la requête j'ai obtenu toutes les informations du Marvel en question, mais si j'écris un Marvel qui n'existe pas je reçois une réponse incomplète avec presque aucune information car le Marvel n'existe pas.

Étape 2 - Toujours plus loin

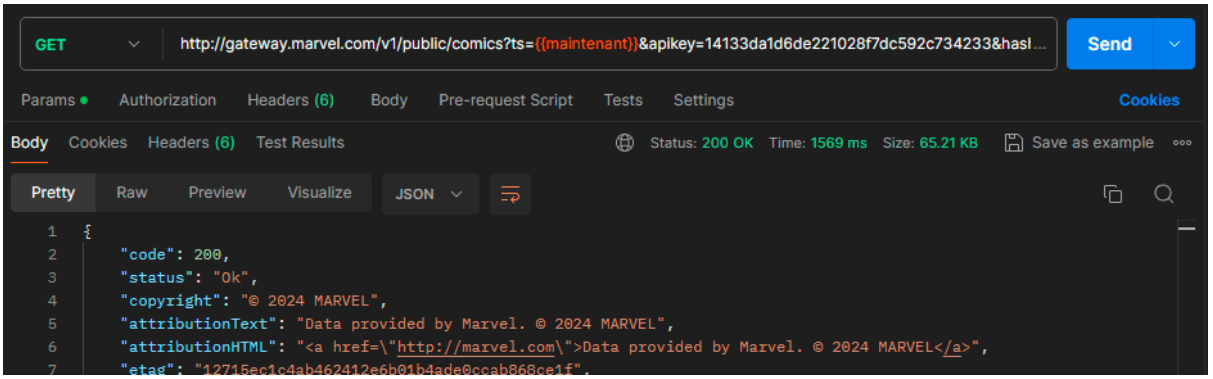
J'ai créé la Collection Marvel. Dedans j'ai ajouté le prescript suivant :

```
const publicKey = "14133da1d6de221028f7dc592c734233"
const privateKey = "1110547705a0d58a5a4a3f5129900d9fc171e1ae"
const ts = new Date().getTime();
const hash = CryptoJS.MD5(ts + privateKey + publicKey).toString();

console.log(ts + hash);

pm.environment.set("maintenant", ts);
pm.environment.set("hachage", hash);
```

Pour la valeur **ts** je récupère le nombre de milliseconde depuis le 01/01/1970 et pour la valeur **hash** je calcule le hachage MD5 de la chaîne de caractère composée de ts, privateKey et publicKey. Par la suite, je rajoute deux paramètres à ma requête : **maintenant** et **hash** et je rajoute à la main la variable **apiKey**. Voici mes paramètres :

	Key	Value
☑	ts	{{maintenant}}
☑	apikey	14133da1d6de221028f7dc592c734233
☑	hash	{{hachage}}

Voici ma requête au final qui me renvoie une réponse correcte et complète :



Étape 3 – Tu veux ma photo ?

Voici la méthode *getData* qui refait le comportement de postman en prenant tous les paramètres et en renvoyant la réponse :

```
export const getData = async (url) : Promise<json>  => {
    const publicKey : string  = "14133da1d6de221028f7dc592c734233";
    const privateKey : string  = "1110547705a0d58a5a4a3f5129900d9fc171e1ae";
    const ts : number  = new Date().getTime();
    const hash : ArrayBuffer  = await getHash(publicKey, privateKey, ts);

    const url1 : string  = url + "/v1/public/characters?ts=" + ts + "&apikey=" + publicKey + "&hash=" + hash;
```

Par la suite j'ai ajouté ce code (dans la méthode *getData*) qui filtre tous les tous les résultats qui ont une image thumbail valide. Ensuite je créer l'objet personnage qui contient le nom et l'url de la destination de l'image du personnage :

```
const characters = data.data.results.filter(result => {
    return result.thumbnail && result.thumbnail.path && !result.thumbnail.path.includes("image_not_availabl
}).map(result => {
    const thumbnailUrl : string  = `${result.thumbnail.path}/portrait_xlarge.${result.thumbnail.extension}`;
    return { name: result.name, imageUrl: thumbnailUrl };
});

console.log(characters);

return characters;
```

J'ai également codé la méthode *getHash* qui génère un hash à partir de la publicKey, privateKey et du timestamp :

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

Quand je test j'obtiens le bon résultat dans la console (tous les personnages validés avec leur nom et leur image) :


```
[
  {
    name: '3-D Man',
    imageUrl: 'http://i.annihil.us/u/prod/marvel/i/mg/c/e0/535fecbbb9784/portrait_xlarge.jpg'
  },
  {
    name: 'A-Bomb (HAS)',
    imageUrl: 'http://i.annihil.us/u/prod/marvel/i/mg/3/20/5232158de5b16/portrait_xlarge.jpg'
  },
```


Étape 4 – On s’accroche au guidon

Tout d’abord, je commence par enregistrer le moteur de rendu Handlebars avec Fastify, en spécifiant les partials :

```
const app : FastifyInstance<...> & PromiseLike<...>  = fastify();

app.register(import('@fastify/view'), {
    engine: {
        handlebars
    },
    templates: './templates',
    options: {
        partials: {
            header: './header.hbs',
            footer: './footer.hbs'
        }
    }
});
```

Ensuite j’ai codé le app.get qui execute la requête et qui envoie la liste de personnages à la page index :

```
app.get('/', async (request : FastifyRequest<RouteGeneric, http.Server, http.IncomingM
    const characters : json  = await getData( url: "http://gateway.marvel.com");
    return reply.view('index', { characters: characters });
});
```

Enfin, dans le index.hbs, je boucle pour ce tableau de personnages et affiche chaque personnage sous forme de carte, avec son nom et son image.

```
{{#each characters}}
    <div class="col-md-3">
        <div class="card mb-4 box-shadow">
            <img class="card-img-top" src="{{ imageUrl }}" alt="{{ name }}">
            <div class="card-body">
                <h5 class="card-title">{{ name }}</h5>
            </div>
        </div>
    </div>
{{/each}}
```

Voici le rendu final sur le http://localhost:3000/ :