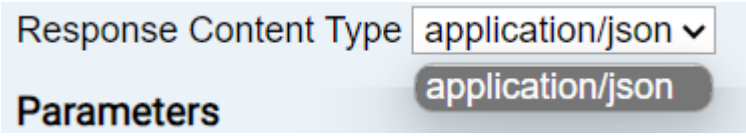
ÉNONCE TRAVAUX PRATIQUE - SUJET 2

Les Marvels

Étape 1 - « Avengers, rassemblement ! »

Dans un premier temps je me suis créé une API avec le lien présent dans le sujet du TP. Après cela je suis allé voir la documentation pour pouvoir déplier les premières requêtes GET /v1/public/characters.



Request URL

```
https://gateway.marvel.com:443/v1/public/characters?apikey=bd3546c18ac868d22f02995dd1c67f7f
```

Response Body


```
{
  "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": "3cdad81eb071614ff249b73ecef3d79cde180b0c",
  "data": {
    "offset": 0,
    "limit": 20,
    "total": 1564,
    "count": 20,
    "results": [
      {
        "id": 1011334,
        "name": "3-D Man",
        "description": "",
        "modified": "2014-04-29T14:18:17-0400",
```


Response Code

```
200
```

Response Headers

```
{
  "Content-Type": "application/json; charset=utf-8",
  "Date": "Thu, 01 Feb 2024 13:59:06 GMT"
}
```

Voici l'URL quand j'appuie sur Try it out sans parametre :

`https://gateway.marvel.com:443/v1/public/characters?apikey=bd3546c18ac868d22f02995dd1c67f7f`

Et quand je rajoute un nom en paramètre :

Request URL

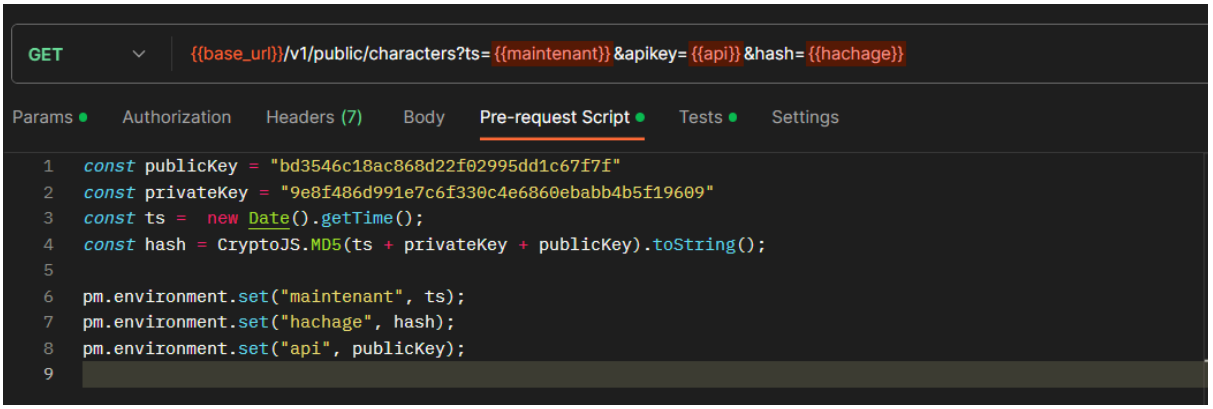
```
https://gateway.marvel.com:443/v1/public/characters?name=Abyss&apikey=bd3546c18ac868d22f02995dd1c67f7f
```

Response Body

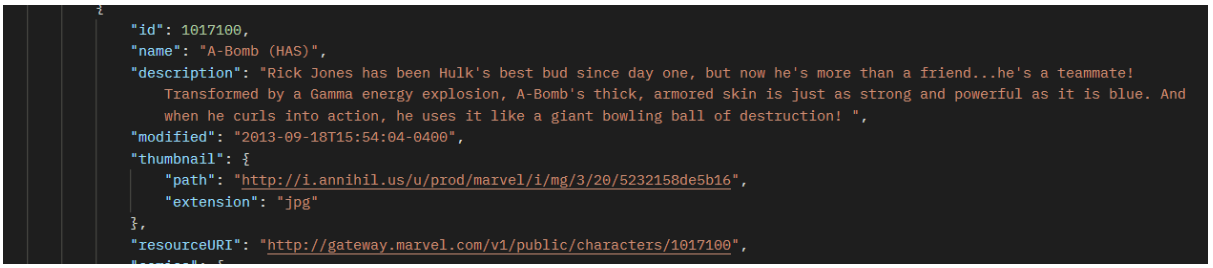

```
{
  "code": 200,
  "status": "Ok",
  "copyright": "© 2024 MARVEL",
  "attributionText": "Data provided by Marvel. © 2024 MARVEL",
  "attributionHTML": "<a href=\"http://marvel.com\">Data provided by Marvel. © 2024 MARVEL</a>",
  "etag": "298021c021e1f8b8e765aa565d87152eedd685ba",
  "data": {
    "offset": 0,
    "limit": 20,
    "total": 1,
    "count": 1,
    "results": [
      {
        "id": 1009149,
        "name": "Abyss",
        "description": "",
        "modified": "2014-04-29T14:10:43-0400",
```

Étape 2 - Toujours plus loin

J'ai mis en place le pré-script



Voici le résultats



Étape 3 - Tu veux ma photo ?

Dans un premier temps j'ai repris le code postman que j'ai adapté sur phpstorm pour pouvoir récupérer toutes les données via l'API en codant les fonctions getData et getHash.

```
export const getData = async (url) : Promise<json>  => {
    const publicKey : string  = "bd3546c18ac868d22f02995dd1c67f7f";
    const privateKey : string  = "9e8f486d991e7c6f330c4e6860ebabb4b5f19609";
    const ts : number  =  new Date().getTime();
    const hash : ArrayBuffer  = await getHash(publicKey, privateKey, ts);
    const fullurl : string  = url+"/v1/public/characters?ts="+ts+"&apikey="+publicKey+"&hash="+hash;
    try {
        const response : Response  = await fetch(fullurl);
        const data = await response.json();
        return data;
    } catch (error) {
        console.error("Error fetching data:", error);
        throw error;
    }
}


/**
 * Calcul la valeur md5 dans l'ordre : timestamp+privateKey+publicKey
 * cf documentation developer.marvels.com
 * @param publicKey
 * @param privateKey
 * @param timestamp
 * @return {Promise<ArrayBuffer>} en hexadecimal
 */
1+ usages  calli77 *
export const getHash = async (publicKey, privateKey, timestamp) : Promise<ArrayBuffer>  => {
    const hash : string  = crypto.createHash( algorithm: 'md5').update( data: timestamp + privateKey + publicKey).digest(
    return hash;
```

Ensuite pour filtrer j'ai créé un tableau qui retient seulement les personnages ayant une image valide. Pour cela, j'ai créé un tableau qui stocke uniquement les personnages avec une image disponible dans leur propriété 'thumbnail'.

```
try {
    const response : Response  = await fetch(fullurl);
    const data = await response.json();
    const charactersAvecImage = data.data.results.filter(character => {
        return character.thumbnail && character.thumbnail.path !== "image_not_available";
    });
    const characters = charactersAvecImage.map(character => {
        return {
            name: character.name,
            description: character.description,
            imageUrl: `${character.thumbnail.path}/portrait_xlarge.${character.thumbnail.extension}`
        };
    });
    return characters;
```

A la fin je ne garde que le nom la description et l'url de l'image le nom du personnage, sa description s'il en a une, et l'URL de l'image en utilisant la version portrait_xlarge de l'image.

Pour tester les résultat j'ai fais un console.log dans le server

```
import {getData} from "./api.js";

console.log(await getData( url: "https://gateway.marvel.com:443"));
```

```
{
  name: '3-D Man',
  description: '',
  imageUrl: 'http://i.annihil.us/u/prod/marvel/i/mg/c/e0/535fecbbb9784/portrait_xlarge.jpg'
},
{
  name: 'A-Bomb (HAS)',
  description: "Rick Jones has been Hulk's best bud since day one, but now he's more than a friend...he's a teammate! Transformed by a Gamma energy explosion, A-Bomb's thick, armored skin is just as strong and powerful as it is blue. And when he curls into action, he uses it like a giant bowling ball of destruction! ",
  imageUrl: 'http://i.annihil.us/u/prod/marvel/i/mg/3/20/5232158de5b16/portrait_xlarge.jpg'
},
{
  name: 'A.I.M.',
  description: 'AIM is a terrorist organization bent on destroying the world.',
  imageUrl: 'http://i.annihil.us/u/prod/marvel/i/mg/6/20/52602f21f29ec/portrait_xlarge.jpg'
},
{
  name: 'Aaron Stack',
  description: '',
  imageUrl: 'http://i.annihil.us/u/prod/marvel/i/mg/b/40/image_not_available/portrait_xlarge.jpg'
},
{
  name: 'Abomination (Emil Blonsky)',
  description: 'Formerly known as Emil Blonsky, a spy of Soviet Yugoslavian origin working for the KGB, the Abomination gained his powers after receiving a dose of gamma radiation similar to that which transformed Bruce Banner into the incredible Hulk.',
  imageUrl: 'http://i.annihil.us/u/prod/marvel/i/mg/9/50/4ce18691cbf04/portrait_xlarge.jpg'
},
```

Étape 4 - On s'accroche au guidon

Dans un premier temps j'ai configuré fastiview en mettant les paramètre menant au fichier contenant les templates ensuite j'ai configurer le get en mettant /caractère pour mener a la page contenant les personnage

```
const app : FastifyInstance<...> & PromiseLike<...>  = fastify();
app.register(fastifyView, {
    engine: {
        handlebars,
    },
    templates: "./templates",
    options: {
        partials: {
            header: 'header.hbs',
            footer: 'footer.hbs'
        }
    },
});

new *
app.get('/characters', async (req : FastifyRequest<RouteGeneric, Server<...>, IncomingMessage, SchemaCompiler, FastifyTypeProviderD
    try {
        const characters : json  = await getData( url: 'https://gateway.marvel.com');
        await res.view('index.hbs', { characters });
    } catch (error) {
        console.error("Error fetching data:", error);
        res.status( statusCode: 500).send( payload: 'Internal Server Error');
    }
});
```

Dans index.hbs j'ai rajouté les partials header et footer et j'ai fais une boucle pour afficher chaque image de chaque personnage.

```
{{> header}}

<div class="container-fluid m-2 p-5 bg-primary text-center">
    <h1>Les Marvels</h1>
</div>
<div class="container m-2 p-5 bg-primary text-center">
    <h2>Mes comics</h2>

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

{{> footer}}
```

Voici le résultat :