Rapport TP2

LesMarvel

Étape 1 et 2.

La première étape consistait à obtenir une clé API auprès de Marvel Developer et à comprendre le mécanisme d'authentification via **timestamp, clé publique et clé privée**. L'authentification repose sur la génération d'un **hash MD5** de la forme suivante :

hash = md5(timestamp + private_key + public_key)

Étape 3 et 4.

Implémentation

Nous avons créé une fonction `getData()` dans `api.js` permettant de :

1. **Générer le hash d'authentification** avec `crypto-js`.
2. **Construire l'URL de requête** avec les paramètres nécessaires.
3. **Effectuer une requête GET** vers l'API Marvel.
4. **Filtrer les résultats** pour ne conserver que les personnages avec une image disponible.
5. **Retourner un tableau de personnages** avec `nom`, `description` et `imageUrl`.

Nous avons mis en place un **serveur Fastify** pour servir les données et afficher les personnages sous forme de cartes en utilisant **Handlebars**.

Configuration de Fastify

Dans `server.js`, nous avons :

1. **Enregistré le moteur de template Handlebars** avec `@fastify/view`.
2. **Défini le dossier contenant les fichiers de template** (`templates/`).

3. **Créé une route principale `/`** qui récupère les personnages via `getData()` et les envoie à la vue `index.hbs`.

Structure des Templates

Nous avons organisé les fichiers Handlebars comme suit :

- `layout.hbs` : Contient l'en-tête et la structure HTML de la page.
- `index.hbs` : Affiche la liste des personnages sous forme de **cartes Bootstrap**.
- `header.hbs` et `footer.hbs` : Partials réutilisables pour structurer l'interface.

Étape 5.

Afin de rendre notre application **portable et facile à déployer**, nous avons créé un **Dockerfile** et un fichier `.dockerignore`.

Fichier `.dockerignore`

Nous avons exclu les fichiers inutiles pour éviter d'alourdir l'image Docker :

node_modules
npm-debug.log

Fichier `Dockerfile`

Nous avons défini un **Dockerfile optimisé** pour exécuter notre application Fastify de manière sécurisée et performante.

1. **Utilisation de l'image légère de Node.js** :
   FROM node:lts-bullseye-slim
2. **Création de l'arborescence de l'application** :
   RUN mkdir -p /home/node/app/node_modules && chown -R node:node /home/node/app
3. **Définition d'un utilisateur non root** :
   USER node

Installation des dépendances et copie du code :
COPY package.json package-lock.json ./

RUN npm install --omit=dev

4. COPY . .

Exposition du port 3000 et lancement de l’application :
EXPOSE 3000

5. CMD ["node", "src/server.js"]

Construction et exécution du conteneur

Nous avons exécuté les commandes suivantes pour tester le conteneur :

docker build -t lesmarvels .
docker run -p 3000:3000 lesmarvels
L’application était alors accessible à l’adresse `http://localhost:3000/`.