Programmation Web

Nicolas Raby

Janvier 2019

1 Introduction

Pour ce projet nous avons du réaliser une application web consommant les données d'un serveur d'application. Il nous a été demandé de même de consommer une API tierce. Pour ce projet j'ai décidé de réaliser une application sur le thème du très célèbre jeu Pokemon. Cette application aura pour but final de permettre à ses utilisateurs d'afficher une liste complète des pokémons. Ils pourront aussi faire une recherche d'un pokémon en particulier et ainsi avoir accés à plus d'informations sur ce dernier. L'utilisateur, si il le souhaite, pourra se connecter grâce à un identifiant et un mot de passe. Ainsi il aura accès à un système de création d'équipe. L'utiliateur pourra ainsi créer plusieurs équipes et y incorporer les pokémons qu'il veut. L'API utilisée se nomme PokéAPI et est disponible ici PokéAPI

2 Outils utilisés

- Serveur d'application : NodeJS
- Application web : Angular/NodeJS
- Serveur public : L'application n'est pas déployée sur un serveur public actuellement. Après de nombreux essais en vain via Héroku, j'ai décidé d'abandonné afin de me concentrer sur l'application.

3 Fonctionnement

L'application se divise en deux parties. Le dossier "backend" représente notre serveur d'application. Le reste étant notre application web.

3.1 Serveur d'application

C'est lui qui consomme l'API tierce. On y retrouve ainsi les différentes routes. La plus importante est pokemon.js. Ce fichier permet de définir les urls qui nous serviront plus tard à acceder aux données comme le montre la Figure **??**

Chaques routes a son controller afin d'exercer les requêtes vers PokéAPI. "PokeController.js", ainsi, peut requêter l'API tierce afin de récupérer, soit tous les pokémons, soit les pokémons en fonction de leurs identifiants etc.

3.2 Application Web

L'application web comme dit plus tôt a été réalisée avec le framework Angular. Notre application dispose ainsi de plusieurs composants (catégories). On peut retrouver une barre de navigation comportant plusieurs onglets. En intéragissant avec ce composant nous pourrons ainsi naviguer à travers l'application. Nous allons expliquer ici deux onglets : Kanto Pokedex et Research.

- Kanto Pokedex : Cette page nous renvoie une liste complète des 151 premiers pokémons. Ainsi après intéraction de l'utilisateur via cet onglet, une requête est envoyée vers notre backend qui, lui, adresse une requête à l'API Tierce afin de récuperer les identifiants ainsi que les noms des pokémons et de les afficher avec leurs images correspondantes.
- Research : Cette catégorie permet à l'utilisateur de faire une recherche d'un pokémon en particulier, que ce soit via sont identifiant (numéro) ou bien via son nom en anglais. Une fois les requêtes faites de la même manière que pour le Pokedex, les informations du pokémon apparaissent (nom, numéro, sprites et ses caractéristiques) cf Figure 2

4 Difficultés rencontrées

- La mise en place sur un serveur public de notre api à été un réel frein à l'avancement du projet, en effet, une fois déployé il était impossible d'accéder à l'url proposée par héroku.
- L'optimisation des requêtes. PokéAPI est particulièrement bien organisé mis apart pour la récupération complète de la liste des pokémons. J'ai trouvé une solution qui marche mais qui est lente et envoie beaucoup trop de requêtes que ce soit entre le frontend et le backend, ou bien entre le backend et l'API tierce. Voulant optimiser ce nombre de requête j'ai eu du mal à trouver une solution et, préssé par le temps j'ai décidé de laisser cela de côté en gardant une version un peu "lente" mais fonctionnelle.
- Lors de la récupération de la liste complète des pokémons, j'ai été confronté à un problème qui m'a pris du temps à résoudre. En effet, la liste des pokémons qui m'était retournée été complète mais pas dans le bon ordre. J'ai réussi après plusieurs tentatives à résoudre le problème grâce à un forkjoin().

```
router.get('/', async function(req, res) {
  await res.header('Content-Type', 'application/json');
  await res.write(JSON.stringify(await (new pokeController()).getAllPokemons()));
  await res.end();
});

router.get('/:id', async function(req, res) {
    await res.header('Content-Type', 'application/json');
    await res.write(JSON.stringify(await (new pokeController()).getPokemonById(req.params.id)));
    await res.end();
  });
```

Figure 1: Exemples routes

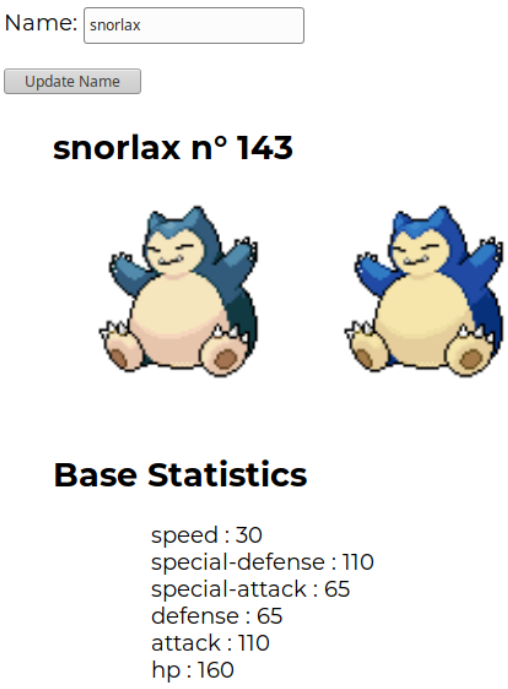

Figure 2: Recherche de Pokemon

5 Améliorations

Beaucoup d'améliorations pour cette application sont envisageables.

- Rendre la connection possible pour les utilisateurs. Une connection à la base de donnée est établie et fonctionne, mais cause du manque de temps, la gestion des utilisateurs n'a pas été faites.
- Optimiser les requêtes. Lors du requêtage pour avoir une liste des 151 pokémons, on fait 151 requêtes au backend et le backend en fait lui aussi 151 vers PokéAPI. Si cela est fait ainsi, c'est par ce que je n'ai pas réussi l'optimisation à temps. Au vu du fonctionnement de PokéAPI il faudra bel et bien 151 requêtes pour récupérer notre liste. Cependant il est possible de ne faire qu'une requête de notre front vers notre back afin de récupérer cette liste.
- Implémenter le système d'équipe et permettre à un utilisateur de pouvoir enregistrer les pokémons, que ce soit via la liste totale ou bien via l'outil de recherche.

6 Conclusion

Réaliser ce projet relevait d'un véritable défis. En effet, la programmation web est une nouveauté pour moi et je m'y intéresse de plus en plus. Grâce à la mise en place de plusieurs outils, ce projet m'a permis de mieu comprendre et appréhender les notions importantes de programmation web. Il à été d'autant plus important pour moi de réaliser ce projet puisque mon stage de fin d'année sera dans ce domaine, qui plus est avec le framework Angular. J'ai pour regret de ne pas avoir pu finir et aller aussi loin que je l'aurai voulu mais j'ai acquis énormément de connaissances au cours de ce projet.