Express API - TP 01

01 . Mise en place

Le but de l'application sera de produire une API REST sur le thème du Pokédex. Ce dernier permettra de manipuler des Pokémons et des dresseurs (utilisateurs).

La stack technique utilisée sera :

- NodeJS v22.x
- Framework Express.JS
- Base de données MongoDB

Dans un soucis de simplicité, les images des Pokémons pourront être prise du site Poképedia. Soit en gérant l'upload, soit en indiquant les URLs.

02 . Création du projet

Créez sur votre ordinateur un dossier.

Créez dans ce dossier un nouveau projet d’API via la commande suivante :

```
npm init
```

03 . Architecture du projet

Dans un dossier `src` vous aller créer les dossier qui serviront à définir l'architecture de notre projet :

- models
- middlewares
- controllers
- services
- le fichier `server.js`

Cela nous permettra de séparer les différentes responsabilités.

Installez les dépendances minimum dont vous aurez besoin :

```
npm install bcrypt express jsonwebtoken mongoose nodemon
```

Récupérez le serveur portable MongoDB sur https://iutdijon.u-bourgogne.fr/intra/info/softs/ (si le lien ne fonctionne pas, voir avec votre enseignant) et placez le dans un dossier accessible en écriture. Créez un dossier data qui contiendra les données de votre base et lancez (par ligne de commande, § A2) le serveur MongoDB.

Testez que tout s'exécute correctement. Et modifiez si besoin votre `package.json` pour incorporer `nodemon`

04 . Une première route

Nous allons créer notre première route. Celle-ci aura pour but de retourner la liste des types de Pokémons.

L'url de notre route sera GET `/api/pkmn/types`.

```
Vous pouvez notez dans l'url un /api. Celui-ci sera présent sur toute les routes. Pour éviter de devoir le répéter à chaque route, vous pouvez le configurer dans votre server.js avec un router utilisant tous les autres.
```

Pour commencer, nous attaquerons la partie Data. Dans le dossier `models`, nous allons créer un array nommé `PkmnType`. Celle-ci contiendra la liste des types en majuscule. Et ... c'est tout !

Ensuite, dans notre service, une fonction pour récupérer les différentes valeurs de notre array puis les retourner en tant que liste de String. Je vous laisse chercher comment faire.

Et puis pour finir, nous allons appeler notre service dans notre fonction du controller. Puis, ce dernier construira la réponse pour qu'elle soit dans un format JSON valide.

L'objectif, c'est que l'on puisse visiter notre URL en ayant un retour similaire (l'ordre des clés n'importe pas, et les noms des types peut être dans une autre langue !)

```
{
    "data": [
        "NORMAL",
        "FIRE",
        "WATER",
                ...
        "FAIRY"

    ],
    "count": 18
}
```

Et voila, si tout fonctionne ! Le TP 01 est terminé ! La suite au TP 02 !