Le projet - WeatherApp
Day 4 - Stocker l'information en base de données

Objectif de la journée

A ce stade du projet, il y a un problème de persistance des informations. A chaque redémarrage du serveur les informations des villes sont perdues.
L'objectif est donc d'utiliser une base de données pour que celle-ci puisse stocker les informations des villes afin de les retrouver même après un redémarrage du serveur.

1- Préparer l'environnement de travail

- Création du répertoire **part4** dans le répertoire **./lacapsule/project/weatherapp**
- Reprendre le livecode **part3** et le déposer dans le répertoire **part4**
- Se positionner sur le répertoire **part4** via le terminal

```
cd ./lacapsule/project/weatherapp/part4
```

2- Initialisation de mongoDB

Création de la base de données

- Créez sur mlab une base de données nommée **openweatherapp**.
- Créez ensuite un utilisateur pour cette base de données

Connexion à la base de données

- Installez les module mongoose
- Import du module dans le projet, dans le fichier **route/index.js**
- Connectez-vous à la base de données avec vos paramètres de connexions

3- Enregistrer une ville

A ce stade, l'application enregistre les informations des villes dans une variable globale nommée **cityList.** L'objectif est de ne plus passer par cette variable globale et de la remplacer par la base de données.

- Définissez un schéma qui vous servira à stocker les informations des villes
  Note:
  Pour ne rien oublier, regardez les informations contenues dans **cityList**.

- Définissez un modèle, choisissez comme nom de collection **cities**.

- Dans le fichier **route/index.js** repérez la route qui est responsable de l'enregistrement des villes.

- Une fois que vous avez localisé la bonne route, refaite un point sur les étapes nécessaires pour définir les emplacements exacts où devront être écrit les requêtes à la base de données.

  - ★ Interrogez le webservice
  - ★ Attendre la réponse du webservice
  - ★ **Enregistrez en base de données**
  - ★ Attendre que la base de données ait fini l'enregistrement
  - ★ **Récupérez l'intégralité des informations des villes en base de données**
  - ★ Attendre le retour de la base de données
  - ★ Renvoyez la réponse au navigateur

  Note:
  Attention au côté asynchrone des opérations.
  L'asynchrone est localisé au niveau des étapes qui nécessite une attente d'un résultat.
  Autre point important, l'utilisation de deux requêtes successives:
  - ★ La première pour enregistrer la nouvelle ville,
  - ★ La deuxième pour récupérer l'intégralité des villes déjà enregistrées (mais également la nouvelle).

  Cette dernière est indispensable pour afficher les résultats au navigateur.

- Une fois que vous avez une vision claire de l'emplacement et des étapes à réaliser, écrivez la première requête pour enregistrer les informations d'une ville.
  Note:
  Rappelez-vous, l'enregistrement se fait en 2 étapes:
  - ★ La préparation des données
  - ★ L'écriture en base de données

- Passez à la deuxième requête pour récupérer l'intégralité des villes en base de données.
  Note:

Attention à l'asynchrone, pensez à regarder la décomposition des étapes que vous avez fait précédemment.

- Une fois que vous avez récupéré l'intégralité des villes, envoyez via `res.render()` cette liste au fichier **index.ejs**

  Note:
  Encore une fois faites attention à l'asynchrone, vous ne pouvez renvoyer un résultat au navigateur que lorsque vous avez la garantie d'avoir reçu le résultat de la base de données.

4- Supprimer une ville

Tout comme l'étape de suppression, l'objectif est de ne plus passer par la variable globale **cityList** et de la remplacer par la base de données.

- Dans le fichier `route/index.js`, repérez la route où la requête qui permettra de supprimer une ville devra être écrite.
  Note:
  Rappelez-vous de l'objectif: Lorsque l'on clique sur l'icône de la croix d'une ville, on supprime les informations de celle-ci en base de données.

- Une fois que vous avez localisé la bonne route, refaite un point sur les étapes nécessaires pour définir les emplacements exacts où devront être écrit les requêtes à la base de données
  - ★ Supprimez la ville sélectionnée en base de données
  - ★ Attendre que la base de données ait fini la suppression
  - ★ Récupérez l'intégralité des informations des villes en base de données
  - ★ Attendre le retour de la base de données
  - ★ Renvoyez la réponse au navigateur

  Note:
  Le côté asynchrone est localisé au niveau des étapes qui nécessitent une attente d'un résultat.
  Autre point important, l'utilisation de deux requêtes successives:
  - ★ La première pour supprimer une ville,
  - ★ La deuxième pour récupérer l'intégralité des villes présentes en base de données. Cette dernière est indispensable pour afficher les résultat au navigateur.

- Une fois que vous avez une vision claire de l'emplacement et des étapes à réaliser, écrivez la première requête pour supprimer une ville.
  Note:
  Rappelez-vous la suppression d'un élément en base de données nécessite un filtre.
  La question qui en découle est, comment faire pour cibler précisément un élément d'une base de données ?

On ne peut plus utiliser la position de la ville comme on l'avait fait avec la variable globale **cityList**. En effet, la notion d'ordre en base de données ne fonctionne pas du tout pareil, il est donc préférable de trouver une autre approche.

La solution la plus optimale pour cibler précisément un élément en base de données, et d'utiliser son ID.

- Dans le fichier **index.ejs**, modifiez la requête `/delete-city` qui se trouve dans la balise `<a>` pour que celle-ci renvoi l'ID de la ville et non sa position.

- Revenez dans le fichier **/route/index.js** et exploitez l'ID envoyé par le Front-End dans la requête de suppression.

- Passez à la deuxième requête pour récupérer l'intégralité des villes en base de données.
  Note:
  Attention à l'asynchrone, pensez à regarder la décomposition des étapes que vous avez fait précédemment.

- Une fois que vous avez récupéré l'intégralité des villes, envoyez via `res.render()` cette liste au fichier **index.ejs**
  Note:
  Encore une fois, faites attention à l'asynchrone, vous pouvez renvoyer un résultat au navigateur que lorsque vous avez la garantie d'avoir reçu le résultat de la base de données.

5- Chargement de l'application

Etant donné que la variable globale **cityList** n'existe plus la route / qui l'utilise encore ne fonctionnera plus.

- Ajoutez une requête à cette route pour récupérer l'intégralité des villes en base de données pour les envoyer en réponse au navigateur.