Compte rendu : test technique Sinapps

Pour ce test, j'ai choisi de faire une api rest à partir de spring en Java. Spring permet la mise en place rapidement et facilement d'une api rest complète d'où ce choix.

Je m'excuse tout d'abord pour le délai que cela m'a pris, je suis tombé en pleine semaine de rendu d'examen. Si je devais comptabiliser en nombre d'heures travailler ce test m'aura pris 7 heures au total.

Vous pouvez trouver :

Un lien pour utiliser le swagger : http://localhost:8080/swagger-ui.html

Un autre lien pour le github : https://github.com/DSGAndre/car-park

Choix

J'ai tout d'abord étudié les informations mises à ma disposition et établit ce qui était le plus important à traité. Sur l'image que l'on m'a fournie et qui devait me servir d'exemple on y voit :

Les parkings les plus proches, les types de véhicules pouvant s'y stationner, un nom, une capacité totale et le nombre de places libres.

J'ai fait le choix de ne pas intégrer les notions de prix car il n'y avait aucunes règles communes à tous les parkings (Chaque parking a son propre prix en fonction de son propre ratio prix/temps).

Il était simple de prendre en compte la notion de covoiturage et de deux roues/vélos vu que le fonctionnement reste à peu près le même que celui des voitures, c'est pourquoi j'ai ajouté des requêtes concernant ses deux catégories.

Je n'ai pas plus personnalisé le design ou encore le swagger car j'ai préféré me focaliser sur la partie technique de l'exercice, d'autant plus que j'avais une semaine de retard.

Requêtes

Tout d'abord il y a les CRUD basiques :

- Recevoir un parking en fonction d'un id donné
- Recevoir tous les parkings
- Supprimer un parking en fonction d'un id donné
- Modifier un parking en fonction d'un id donné

J'ai pensé que l'utilisateur pourrait ne pas forcément connaître l'id mais aurait voulu faire une recherche par nom donc j'ai mis en place une requête qui fait une recherche par nom.

Une recherche permettant de trouver les cinq parkings les plus proches en fonction d'une position au format latitude, longitude.

Les utilisateurs préfèrent surement établir une liste de parkings proches de leur position mais également des parkings ouverts d'où la requête appeler « fiveOpenAndClosestParks » toujours en fonction d'une latitude et longitude donnée.

Et enfin le même type de requêtes pour les deux roues/vélos et le covoiturage :

- Tous les parkings ayant des places pour les deux roues
- Tous les parkings ayant des places pour les vélos
- Les cinq parkings les plus proches ayant des places pour les deux roues
- Les cinq parkings les plus proches ayant des places pour les vélos

A améliorer

Je sais que la suppression et la modification sont des requêtes que l'on devrait limiter en fonction d'un rôle. N'importe qui ne devrait pas pouvoir supprimer ou modifier une information. Il aurait fallu établir soit un système de connexion, soit un système de token afin de contrôler l'accès à ce genre de requête.

J'aurai également aimé passer plus de temps sur le retour des requêtes, actuellement elle retourne aucune valeur si rien n'est trouvé. Ce qui n'est pas quelque chose d'optimal pour une api Rest.

Et enfin il est important de contrôler les différentes valeurs ajouter à l'aide d'annotations spring tel que min, max pour des nombres ou vérifier si toutes les valeurs d'entrées ne sont pas vide/null.