FLEURIMONT Clarisse, LAGIER Elodie, ZARETTI Guillaume

BACHELOR HUNTERZ

Le bot par les étudiants de la HEIG-VD, pour les étudiants de la HEIG-VD.

FONCTIONNALITÉS ET COMMANDES

DESCRIPTION

Bachelor Hunterz est un bot destiné aux étudiants de la HEIG-VD. Sa principale fonctionnalité est de fournir des exercices récoltés auprès des professeurs de la HEIG-VD, afin de se préparer aux mieux pour les travaux écrits et examens.

CAHIER DES CHARGES

Les fonctionnalités ci-dessous ont été annoncées lors de la rédaction du cahier des charges. Elles ont été implémentées pour une grande majorité.

Commande	Description	Implémentation
/saymyname	Affiche notre prénom et nom	Fonctionnelle
/help	Affiche les commandes	Fonctionnelle
/topusers	Affiche le top 10 des utilisateurs ayant rentré le plus d'exercices.	Fonctionnelle
/topusersexerciseliked	Affiche le top des utilisateurs ayant inséré le plus d'exercice et dont on a aimé un exercice	Fonctionnelle
/randomexercise	Affiche un exercice aléatoire	Fonctionnelle
/exercisesliked	Affiche les exercices qu'on a liké	Fonctionnelle
/recommandations	Affiche les recommandations d'exercices	Fonctionnelle
/newexercise <SIGLE_COURS>	Création d'un exercice	Fonctionnelle
/getuserbyusername <username>	Affiche l'utilisateur selon l'username fourni	Fonctionnelle
/exercisesbyuser <userID>	Affiche les exercices créés par l'utilisateur spécifié	Fonctionnelle
/exercisesbyteacher <SIGLE_PROF>	Affiche les exercices liés au prof spécifié	Fonctionnelle
/exercisesbycourse <SIGLE_COURS>	Affiche les exercices liés au cours spécifié	Fonctionnelle
/exercisesbyteacherandcourse <SIGLE_PROF> <SIGLE_COURS>	Affiche les exercices liés au prof et au cours spécifiés	Fonctionnelle
-	Donner une difficulté à un exercice	Non-fonctionnelle

TECHNOLOGIES ET IMPLÉMENTATIONS

LANGAGE JAVA

Le robot a été implémenté en JAVA, qui, bien qu'impliquant de recoder et adapter toutes les fonctionnalités déjà faites, nous paraissait plus intuitif.

MONGODB & NEO4J

Comme spécifié dans la donnée, nous avons utilisé MONGODB pour stocker nos ressources et Neo4J pour les lier et générer des graphes.

La base de données type documents contient les **utilisateurs** ainsi que les **exercices**.

La base de données type graphe contient les interactions entre ceux-ci :

- **DISPENSE** : lie un professeur à un cours. Le prof dispense le cours.
- **INSERE** : lie un utilisateur à un exercice. Un utilisateur insère un exercice.
- **LIKE** : lie un utilisateur à un exercice. Un utilisateur like un exercice.

REQUÊTES

Nous avons implémenté deux requêtes plus compliquées que les autres :

1. getExercisesRecommandation(String userID)

```
public List<String> getExercisesRecommandation(String userID) {
    try (Session session = driver.session()) {
        return session.readTransaction(tx -> {
            List<String> exercises = new ArrayList<>();
            Result result = tx.run(
                    s: "MATCH (u1:User{userID: '_" + userID + "'})-[l1:LIKE]->(e:Exercice)<-[:INSERE]-(u2:User)-[:INSERE]->(e2:Exercice)<-[l3:LIKE]-(u3:User)\n" +
                            "RETURN e2, COUNT(*)\n" +
                            "ORDER BY COUNT(*) DESC\n" +
                            "LIMIT 5");
            while (result.hasNext()) {
                exercises.add(result.next().get(0).asString());
            }
            return exercises;
        });
    }
}
```

Qui renvoie 5 exercices de l'utilisateur dont nous avons liké un exercice et dont une tierce personne a liké un autre exercice.

2. getTopUserWithALikedExercise(String userID)

```
public List<String> getTopUsersWithALikedExercise(String userID) {
    try (Session session = driver.session()) {
        return session.readTransaction(tx -> {
            List<String> users = new ArrayList<>();
            Result result = tx.run( s: "MATCH (u1:User{userID: '_" + userID + "'})-[l:LIKE]->(e:Exercise)<-[:INSERE]-(u2:User)\n"
                    + "RETURN u2.userID, count(*) AS cnt\n"
                    + "ORDER BY cnt DESC LIMIT 10");
            while (result.hasNext()) {
                users.add(result.next().get(0).asString());
            }
            return users;
        });
    }
}
```

Qui renvoie les 10 utilisateurs qui créent le plus d'exercices par ordre décroissant et dont nous avons liké un exercice.