MAC - Projet

“Bot telegram sur les jeux-vidéos”

18.01.2021

Étudiants
Robin Demarta
Loïc Dessaules
Simon Mattei

Professeur
Nastaran Fatemi

Assistants
Christopher Meier
Guillaume Hochet

1. Introduction **3**

2. Technologies **3**

3. Jeu de données **3**

4. Fonctionnalités implémentées **4**

4.1. Recherche de jeux (inline query) 4

4.2. Notation d'un jeu 4

4.3. "Liker" un tag 5

4.4. Recommandation d'autres jeux 5

5. Requête complexe **5**

6. Conclusion **6**

1. Introduction

Le but de ce projet est de créer un bot Telegram utilisant un jeu de données plus ou moins grand qui sera stocké dans une base de données document. De plus, une base de données graphe sera mise en place afin de gérer des relations et requêtes plus complexes.

Nous avons décidé de partir sur le thème des jeux-vidéos. Il sera possible, via notre bot Telegram, de rechercher un jeu, voir ses informations, le noter (de 1 à 5), de "liker" des tags de jeux puis de demander des recommandations d'autres jeux-vidéos par rapport à nos préférences (tag "likés" et jeux notés).

2. Technologies

Nous avons décidé d'utiliser TypeScript avec l'exemple de code fourni pour le laboratoire afin de réaliser le bot telegram. En ce qui concerne les bases de données, MongoDB pour la base de données document et Neo4j pour celle en graphe.

3. Jeu de données

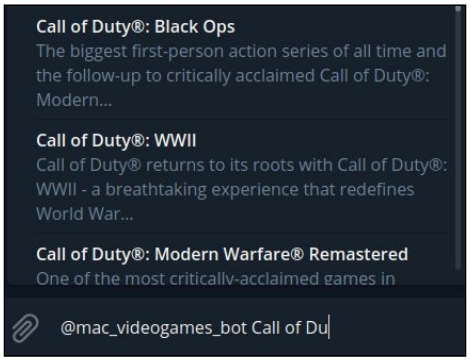
Nous avons trouvé un jeu de données public portant sur le thème des jeux-vidéos (de la bibliothèque de jeux Steam) comportant environ 40'000 entrées. Nous utilisons principalement des informations telles que : le nom, la description, la date de sortie et les tags populaires concernant le jeu vidéo.

Voici le lien du jeu de données :
https://www.kaggle.com/trolukovich/steam-games-complete-dataset

4. Fonctionnalités implémentées

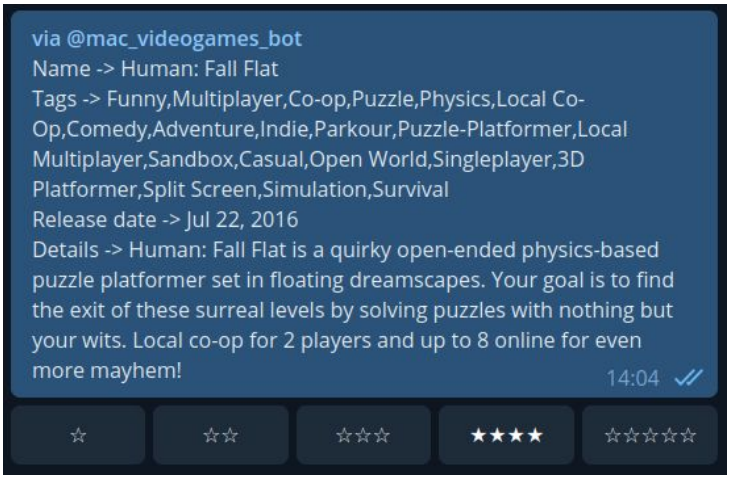
4.1. Recherche de jeux (inline query)

En appelant notre bot depuis n'importe quelle conversation avec la commande **@<botName> <term>** Il est possible de rechercher un jeu par son nom. Par exemple, **@MACbot Call of** fera apparaître tous les jeux commençant par "Call of".



4.2. Notation d'un jeu

Quand nous avons trouvé le jeu que nous voulions grâce à la recherche de jeux, il suffit de cliquer dessus pour voir apparaître plus de détails sur celui-ci. En dessous du texte, nous avons intégré un système de notation afin de noter le jeu de 1 à 5, respectivement la moins bonne à la meilleure note.



Cette note influencera la recommandation des jeux afin de rendre celle-ci plus pertinente. De plus, il est possible de changer notre vote par après, mais pas d'enlever un vote car cela n'a pas été implémenté.

4.3. “Liker” un tag

La commande **/liketag <tag>** permet de “liker” un tag spécifique. Il est possible de like autant de tags que nous le voulons.

Ces likes influenceront la recommandation des jeux afin de rendre celle-ci plus pertinente. L’implémentation de “dislike” n’a pas été mise en place.

4.4. Recommandation d’autres jeux

La commande **/recommendgames** permet de recommander jusqu’à 10 jeux qui seront susceptibles de nous plaire en se basant sur les jeux que nous avons préalablement notés et aux tags likés.

5. Requête complexe

/recommendgames

La commande *recommendgames* utilise deux requêtes Neo4J. La première a pour rôle de trouver les jeux qui possèdent les mêmes tags que les jeux notés par l’utilisateur (dans l’ordre des meilleures notes) et dont ces tags ont aussi été likés. La deuxième va quant à elle rechercher uniquement les jeux ayant les mêmes tags que les jeux notés par l’utilisateur. Ceci a pour but de pouvoir recommander des jeux même si l’utilisateur n’a pas encore spécifié ces tags préférés.

Les entités de notre base de données Neo4J sont les *Users*, les *Games* et les *Tags*. La première requête décrite ci-dessus doit notamment démarrer à partir d’un utilisateur étant déjà en relation (relation *RATED*) avec un (ou plusieurs) jeu. Ce dernier est lui-même lié à un ou plusieurs tags qui, quant à eux, ont potentiellement une relation avec l’utilisateur lui-même (relation *LIKED*). Tout ce cheminement représente les différents niveaux que doit traverser la requête afin de proposer les suggestions de jeux.

6. Conclusion

Malgré le peu de temps que nous avons eu dû aux différents cours en parallèle, nous sommes très satisfait du résultat obtenu.

Le bot Telegram est fonctionnel et nous arrivons à avoir des recommandations de jeux comme nous le voulions en utilisant des requêtes Neo4j traversant plusieurs nœuds et relations, comme demandé.

De plus, afin de complexifier notre bot, nous avons réussi à ajouter la possibilité d'aimer des tags spécifiques, ce qui permet un filtrage plus précis lors de la recommandation de jeux.