I404C NoSQL:Mongo DB

Domaine d'application

Pour le travail NoSQL j'ai choisi une application de gestion de sauvegardes. On assume qu'on a un jeux vidéo qui va régulièrement manier la base de données (principalement pour faire des sauvegardes régulières). Un compte peut avoir plusieurs sauvegardes mais la configuration du jeux est lié au compte et donc il n'y aura qu'un seul fichier config. Le jeux pris comme référence est un RPG et dans les sauvegardes ont stockes surtout des informations lié au personnage, on pourrait imaginer qu'il il y a d'autres informations qu'on devrait stocker dans des fichiers de type différents (des info sur le World State) mais cela devrait être assez facile à faire. Stocker tout info lié à un personnage dans un seul gros fichier serait aussi possible. Les données stockées sont basés sur de vrai sauvegardes et ces données peuvent être trouvés dans le dossier références.

Dans un fichier sauvegarde on stocke le nom du personnage, son niveau, sa lucidité, le dernier endroit ou le joueur se trouvait avant de se déconnecter, le temps de jeu du personnage, les stats du personnage contenu dans un dictionnaire et les items qu'il avait équipé avant de se déconnecter, aussi contenu dans un dictionnaire.

Requêtes et mode d'emploi

Les trois requêtes que l'application est capable de faire sont:

Facile: Changer la luminosité

Medium: Donner le temps de jeux total

Difficile: Retourner une liste de la différence de stats entre deux personnages.

Pour utiliser le programme il faut d'abord installer les librairies nécessaires (voir le readme). Une fois fait il faut un serveur MongoDB, une fois le serveur lancé il suffit d'exécuter FillDB.py pour remplir la base de données. Une fois fait on peut tester les requêtes dans Requests.py, les 3 requêtes sont des fonctions appartenant à un objet SaveManager. Il n'y a pas d'interpréteur donc il faut rentrer dans le code pour tester les requêtes.

La requête la plus difficile à fini par être la requête medium et non la difficile. Au début l'idée était de sommer les attributs playtime pour avoir le temps total dans une seule requête mais: 1. On ne sait pas sommer deux objets datetime mais on peut sommer des timedelta 2. On peut sérialiser un objet datetime mais pas un timedelta.

Stocker un timedelta en tant que string et faire une conversion par après et possible mais je n'ai pas trouver de module python qui faisait exactement ce que je voulais. En fin de compte j'ai utiliser le module isodate pour encoder et décoder des de la librairie datetime.

Lors ce qu'on travaille avec des objets qu'on a définis nous-mêmes pour pouvoir le stocker dans un DB il suffit de rajouter une méthode toJSON ou de définir le __str__ de l'objet. Lorsqu'on travaille avec des objets venant de sources externes la conversion JSON peut poser problème (surtout si l'objet en question et rarement utilisé et qu'il y à très peu de librairies existantes pour le sérialiser, il avait bien plus de manières d'encoder un datetime.datetime qu'un datetime.timedelta)

Il y a aussi une requête supplémentaire (Facile-Medium) qui somme le nombre de lucidité total. Ceci étais supposé être utilisé pour sommer les temps de jeux mais la requête mongDB ne fonctionne bien qu'avec des int et je n'ai pas trouver de moyen de modifier le type de modifier le type de donner avant de fair un $sum. J'ai laisser la méthode comme référence mais il les requêtes mongoDB était assez restrictives. Cependant il est très facile de retrouver des info et les modifier par après.

Conclusion

Un des avantages de la base de données type documents et le fait qu'on peut stocker des fichier ayant une structure différente les unes de autres. Dans cette application on a des objets SaveData et des objets ConfigFile. Les deux types sont stockées comme fichiers dans la db SaveFiles sans poser de problèmes et d'autres objets pourrais facilements être rajouté. Le problème vient du fait que tout ce qui est stocké devra passer par le format JSON et ci un veut récupèrer un objet de la base de donnée (et non juste les info que contient l'objet) il faudra créer une sorte de décodeur. Un des autre avantages est la lisibilité des documents JSON et la facilité avec laquelle ont peut mettre a jour le document, très important pour des fichier sauvegarde. Il n'y a pas vraiment de raison pour laquelle j'ai choisi MongoDB au lieu d'autre DB type documents apart le fait que c'était assez bien documenté et que j'avais rapidement trouvé un driver python pour manipuler MongoDB et comme le temps dedié au projet allait être limité, l'accessibilité était un facteur important.