ESILV LÉONARD DE VINCI | ÉCOLE D'INGÉNIEURS PARIS-LA DÉFENSE

CLOUD APPLICATION DEVELOPMENT

RAPPORT DE PROJET

MATTHIEU LANVERT – VICTOR QUERETTE

IBO 5A – JANVIER 2019

SOMMAIRE

1) ARCHITECTURE DE L'APPLICATION

A) FONCTIONNEMENT GÉNÉRAL

Notre projet consiste en une application web, basée sur une API écrite en python et utilisant le framework Flask.

Cette API fait l'intermédiaire entre le front-end web réalisé en HTML et js, et le cluster de serveurs mis en disposition par le service Atlas de mongoDB.

B) SCHÉMA DES DONNÉES

L'import des données dans notre base se fait en deux étapes. La première est une copie des tables SQL du dataset choisi (disponible ici) dans mongoDB. Cette copie construit une base de données *imports*, contenant le même schéma « relationel » que la DB originale. Ce schéma est fournie dans l'archive du dépôt au format svg.

Cet import se fait en utilisant conjointement les programmes *mysqldump* et *mongoimport*. Ce dernier supporte nativement le format csv que sort *mysqldump* et effectue la conversion JSON directement.

Dans un second temps, ces données sont remaniées en un schéma noSQL. Cette étape est effectuée par la webapp directement : au démarrage, si la base de données *data* est absente du serveur, la dénormalisation est effectuée.

La dénormalisation crée deux collections dans la base de données *data* : *teams* et *players*. Les inforamtions relatives aux compostitions d'équipes et coaches sont embarquées dans les documents de ces collections.

L'organisation de ces collections est détaillée au fur et à mesure de leur construction dans le fichier *mongo_ops.py*.

C) INTERROGATIONS

Simples :

- Combien de fois au cours de son histoire une équipe a-t-elle atteint les playoffs ?

- Pour un coach donné, quelles équipes a-t-il entraîné, en quelles années ?

- Pour une joueuse donnée, qui ont été ses entraîneurs et quand ?

- Pour une joueuse donnée, quand est-elle arrivée en playoffs et avec quelles équipes ?

Lourdes :

- Pour un coach donné, la liste de toutes les joueuses qu'il a entraîné (groupée par joueuse, triée par année).

- (map-reduce) Pour une année donnée, renvoyer pour chaque équipe la proportion de panier 3 points marqués (sur tous les 3 points tentés).

2) LA WEBAPP

A) INSTALLATION ET UTILISATION

Le projet peut être trouvé sur le dépôt git hébergé sur cloud_app_dev_tp, par clonage git ou en téléchargant une archive dans l'onglet *releases*.

Le fichier README.md contient toutes les instructions nécéssaires au build et au run de l'application. Il est à noter que les dépendances de cette dernière y sont listées et sont requises.

La projet a été testé au cours du développement, au build et à l'éxécution, sur les plateformes Linux suivantes : Debian 9, Majaro 18, Alpine 3.8, Ubuntu 18 et Gentoo.

B) CHOIX D'IMPLÉMENTATION

Les scripts python constituant la webapp sont éxécutés dans un environnent virtuel afin de n'installer les dépendances requises (le driver *pymongo* notamment) que dans cet environnement. Cet environnement est généré via *make*. Le lancement de la webapp et des fonctionnalités de gestion de la base de données s'effectue via un script bash *run* situé à la raçine du dépôt.

Le script *app.py* est le point d'entrée de la webapp, et est celui éxécuté par Flask. Il contient la définition de toutes les routes de la webapp et de l'API.

Le script *mongo_ops.py* définit une classe *MongoOps* contenant le code nécessaire à l'interaction avec la base de données. C'est lors de l'instanciation de cette classe que l'application vérifie la présence de la DB dénormalisée, et, le cas échéant, se charge des opérations de passage à un schéma noSQL.

Ce script contient également les fonctions passant les requêtes de l'application au serveur mongo.

Le front-end (la webapp à proprement parler) effectue des requêtes à l'API via *jQuery.*