Dergal Nacer

Leroux Gwenael

Informatique et Réseaux

2ème année

Rapport de Développement

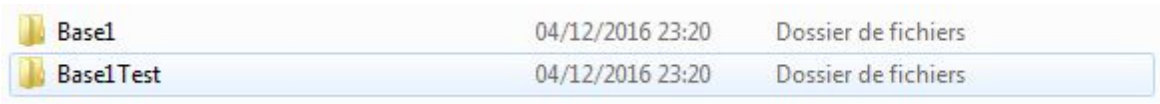
Année 2016-2017

Promotion 2015-2018

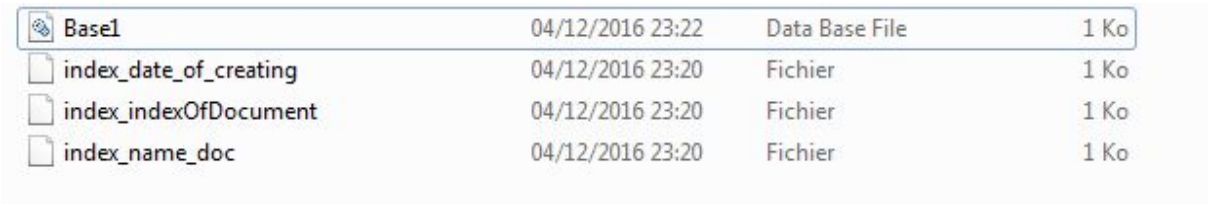
1. La structure de Papaya DB sur le disque :

Dans le workspace du projet, il y a un répertoire « Database ». Ce dossier contient l'ensemble du système de gestion de base de données.

Dans ce dossier se trouve un dossier par base de données que Papaya DB a créé nommé par le nom de la base de données.



Dans chaque dossier d'une base de données se trouve un fichier «nom_de_la_base_de_données.db» et l'ensemble des fichiers concernant l'indexation des documents dans la base de données.



exemple de fichier nom_de_la_base_de_données.db :

```
{name_doc:sdoc1;indexOfDocument:i1;price:i50;}
{name_doc:sdoc2;indexOfDocument:i2;price:i90;author:sauthor1;}
{name_doc:sdoc3;indexOfDocument:i3;price:i40;}
```

Dans ce fichier, un document est spécifié de la façon suivante :

{nom_du_champ:'C'value;nom_du_champ2:'C'value2;...}

Le 'C' correspond au char désignant le type du champ.

Liste des lettres de type actuellement :

- 's' = String
- 'i' = Integer
- 'd' = Date

exemple de fichier index_nom_du_champ_indexé :

```
1,2,8,5,6
```

Dans ce fichier, chaque nombre correspond à l'index d'un document de la base de données. Ils sont rangés dans l'ordre de tri naturel du type du champ indexé.

2. La structure de Papaya DB dans le programme :

Tout d'abord, la classe DataBaseManagementSystem correspond à notre système de gestion de base de données. C'est uniquement avec cette classe que le serveur de base de données va communiquer pour la gestion de la base de données.

Cette classe contient une HashMap<String, DataBaseInMemory>, ce qui correspond à un cache de base de données en mémoire, le String étant le nom de la base de données et DataBaseInMemory est une DataBase associé à une Thread qui va la décharger après une minute si aucune action dessus n'a été faite.

L'ajout ou la suppression d'une base de données est implémenté dans cette classe. Pour les autres, cette classe transmet le traitement à la classe DataBase et renvoie au serveur la valeur retournée par DataBase.

La classe DataBase contient la liste des documents de la base de données ainsi que l'ensemble des indexs. Pour l'ajout d'un document, elle l'ajoute dans sa liste de document et pour chaque champ du document l'indexe, ses actions sont aussi effectivement directement sur le disque. Pour la suppression, elle supprime le document de tous les indexs le référent (mets à jour les indexs sur le disque aussi) par contre elle ne fait que marquer le document comme à supprimer et une Thread viendra faire le travail quand le nombre de documents à supprimer atteindra plus de 80% de la base de données.

La classe Document contient les informations relatives à un document. Il contient une HashMap<String, GenericValue> où le String est le nom d'un champ et GenericValue un objet contenant la valeur du champ.

GenericValue permet au document d'avoir soit un int soit un String soit une Date.

3. API cliente

L'API cliente s'instancie en lui donnant l'adresse d'un serveur REST.

Elle est composés de 2 types de requêtes, les HTTP et les HTTPS. Les URI sont différentes car le protocole est différent. Je me sers des différents status Code du protocol pour renvoyer à l'utilisateur le résultat de sa requête sous forme de string. Les méthode createDB et dropDB nécessitent un couple login mot de passe que nous cryptons en plus de l'envoyer en https pour plus de sécurité.

J'utilise ici les 4 méthodes du protocol HTTP et je me sers principalement de la classe java.net.http.HttpClient mais les méthodes utilisant la méthode DELETE du protocol HTTP ne peuvent pas utiliser HttpClient, je suis donc contraint d'utiliser HttpURLconnection et HttpsURLConnection qui sont beaucoups plus longue à mettre en place en terme de lignes de codes.

J'utilise dans la classe main d'exemple des annotations pour pouvoir appeler les méthodes sans passer par un switch, j'utilise un classValue pour ne avoir à calculer les annotations si cela a déjà été fait auparavant.

4. Serveur REST

Concernant la partie REST de notre projet nous utilisons vertx qui nous est imposés par le sujet. Vertx est capable de créer des serveurs HTTP et HTTPS. Dans notre programme nous lançons deux serveurs avec chacun un routeur différents. Nous attentions les requêtes en fonction de la méthode HTTP ou HTTPS utilisé puis en fonction de l'URI. La suite est délaissée à un handler que prends en charge la requête. En fonction des requêtes on récupère parfois des paramètres dans l'URI et on se charge de transmettre au serveur de base de donnée. Cette partie est en faites un proxy.

5. Serveur de Base de Donnée

Cette partie ressemble beaucoup au serveur REST, mais au lieu de transmettre la requête nous une instance de base de donnée dans laquelle nous appelons nos méthodes. Ces méthodes renvoyant un stream je les place dans le body de la réponse et je la renvoie au serveur REST qui lui se chargera de la renvoyer à l'utilisateur. Selon les méthodes nous décryptons un couple login:motdepasss et nous testons l'authentification pour savoir si l'utilisateur a le droit de faire cette requête. En fonction de cela ou des différentes exceptions levées par les méthodes de la base de donnée nous renvoyons des codes statut différents et la réponse sera donc différentes chez l'utilisateur.