Moteur de Recherche

Agustín Cartaya

Mach 2021

Vidéo explicative
https://youtu.be/Ps4B8EhtHXU

Objectifs cherches

On cherche à créer un « moteur de recherche » en charge de traverser le Web pour récupérer sur chaque page l'url de cette (Source), les urls contenues dans le balises <a> de la page (Destiny), et le texte contenu dans ces balises (Word), pour assembler des triades de la manière [Source, Destiny, Word] et les enregistrer.

Ces triades doivent être accessibles par une ou plusieurs keys, c'est-à-dire qu'un trio peut être consulté soit par leur Source, soit par leur Destiny ou par les deux

Apres avoir obtenu les triades on va utiliser l'algorithme HITS pour classer les pages, cet algorithme va donner comme résultat une liste de pages classe par son hub (quantité de pages qui sont envisagée par cette page) et son Autorithy (quantité de pages qui envisagent cette page)

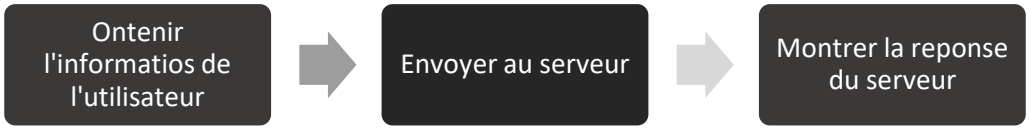
Analyse du problème

Un moteur de recherche est principalement séparé en 2 parties/programmes (Client-Serveur)

Client

Cette partie interagit avec l'utilisateur et s'occupe principalement de :

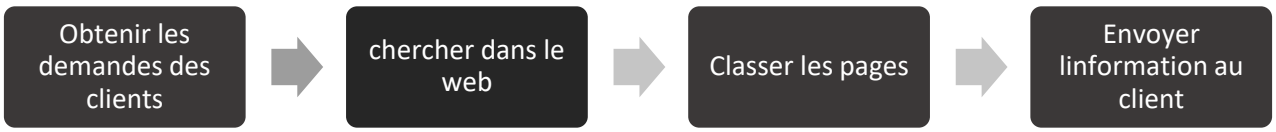
- Obtenir les informations que l'utilisateur veut rechercher
- Envoyer les informations au serveur
- Afficher à l'utilisateur la réponse émise par le serveur



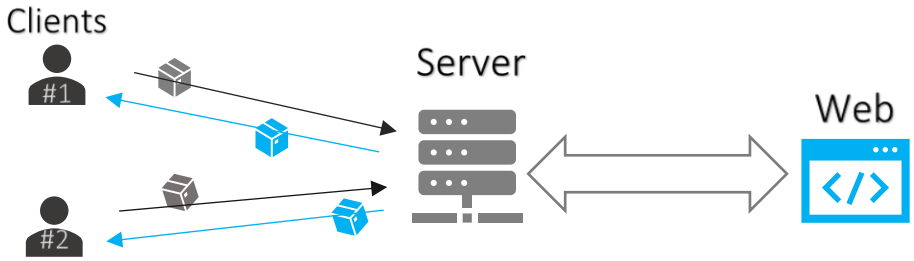
serveur

Cette partie est responsable de :

- Obtenir des informations émises par le client
- Effectuez une ou plusieurs recherches sur le Web
- Classer les pages obtenues
- Envoyer les informations au client



Un serveur peut avoir plusieurs clients simultanément, l'interaction de base entre les clients et le serveur est affichée dans le diagramme suivant



Architecture du programme

Le SearchEngine est composé de 3 programmes :

- SearchEngineCrawler et SearchEngineHits situés dans le côté serveur
- SearchEngineClient) situe dans le côté client.

SearchEngineClient

Fait tout le travail du côté du client comme montrée avant. Il se compose d'un seul thread d'exécution et communique avec SearchEngineCrawler

RechercheEngineCrawler

En charge de :

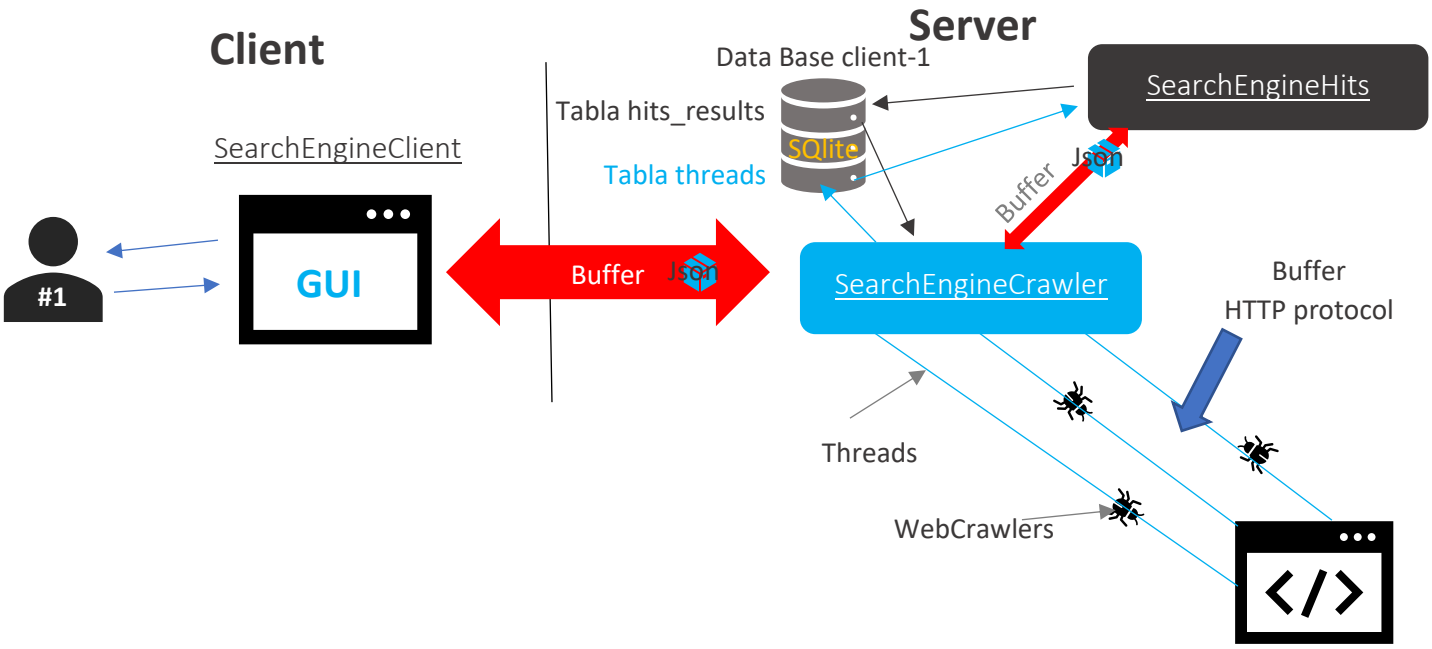
- Obbtenir les informations des clients
- Recueillir des données sur le Web avec une ou plusierus « webCrals » (programmes qui passent par le web en recollectant information, dans notre cas, les url et ses liens avec les autres)
- Enregistrer les informations dans le tableau de la base de données du client
- Envoie l'adresse de base de données client au programme SearchEngineHits pour travailler sur les données obtenues
- Récupérer le signal de fin du programme SearchEngineHits pour obtenir les résultats
- Envoyer les ces résultats au client.

SearchEngineHits

En charge de :

- Se communiquer avec le programme SearchEngineCrawler
- Charger en mémoire les données obtenues par le programme SearchEngineCrawler dans des structures qui ont diffèrent manières(keys) pour accéder aux donnes
- Exécuter l'algorithme HITS
- Enregistrer le résultat dans la base de données du client

Modèle de communication du programme



Problèmes à résoudre - (Manager)

1. Obtention de donnes e interaction con l'utilisateur– (SearchEngineClient)
2. Communication avec le serveur (SearchEngineClient - SearchEngineCrawler)
3. Envoi d'informations au serveur - (SearchEngineClient)
4. Obtenir des informations auprès d'un ou plusieurs clients – (SearchEngineCrawler)
5. Naviguer sur le Web pour obtenir les url - (SearchEngineCrawler)
6. Enregistrer les relations entre les url sous la forme de trios - (SearchEngineCrawler)
7. Communication interne (SearchEngineCrawler - SearchEngineHits)
8. Classer les url avec l'algorithme HITS – (SearchEngineHits)
9. Envoyer des url classifiées au client (s) - (SearchEngineCrawler)
10. Afficher les résultats à l'utilisateur - (SearchEngineClient)

Résolution de problèmes

1. Obtention de donnes e interaction avec l'utilisateur - (SearchEngineClient)

Pour résoudre ce problème, une interface graphique a été créée, qui est composée principalement de 3 zones

Zone 1 - Obtention des données

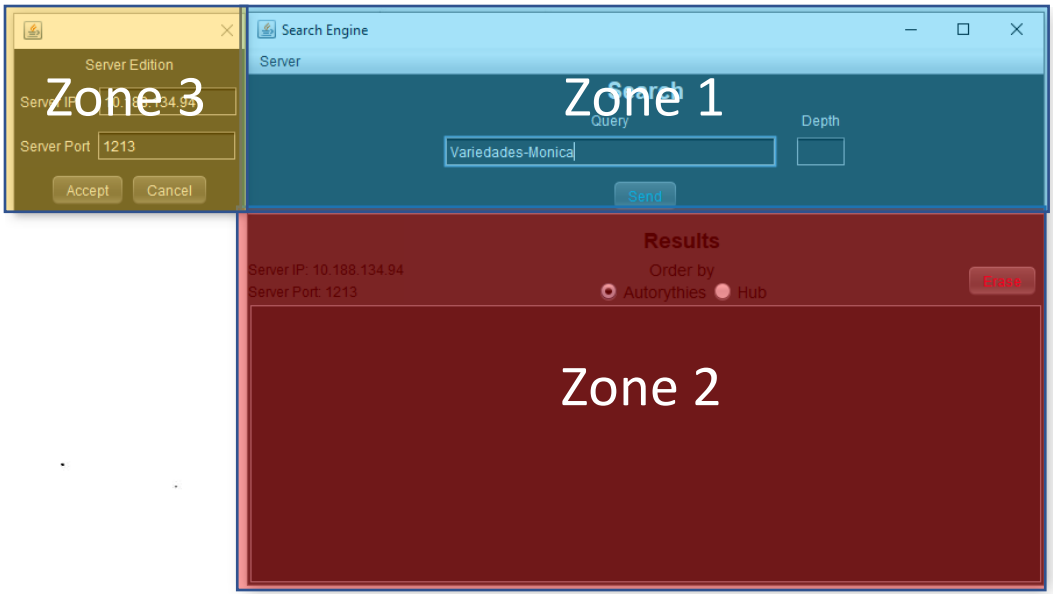
Dans cette zone on obtient, la requête (les mots clés a chercher sur internet) et la profondeur de la recherche (cela sera expliqué plus tard) en utilisant 2 champs d'entrée de texte, et a également un bouton « send » pour envoyer ces informations au serveur

Zone 2 - Panel d'information

Cette zone se compose d'un écran où les informations doivent être affichées, et d'un ensemble de sélecteurs uniques qui sont responsables du tri de l'information

Zone 3 - modification des données de communication avec le serveur

Dans cette zone, l'utilisateur peut modifier l'hôte/ip du serveur et le port que le serveur utilise pour communiquer pour accéder à cette zone, vous devez sélectionner l'option de modification sur la barre d'outils. À partir du menu serveur.



2. Communication avec le serveur (SearchEngineClient - SearchEngineCrawler)

Pour communiquer entre le client et le serveur, un protocole TCP/IP a été mis en œuvre à l'aide de Sockets dont le SocketServer est dans le programme SearchEngineCrawler et le socket (client) est dans le programme SearchEngineClient. Le port de communication par défaut est 1213.

SearchEngineClient.core.Cient

```
//Création du Socket cote client
socket = new Socket(this.serverIP, this.serverPort );
```

SearchEngineCrawler.externalComunication.CrawlerSever

```
//Création du ServerSocket cote serveur
this.server = new ServerSocket(1213);
Socket client;
while (true) {
   client = this.server.accept();
   Thread crawlerClient = new CrawlerClient(client, this.clientIndex);
   crawlerClient.start();
   this.clientIndex++;
}
```

3. Envoi d'informations au serveur - (SearchEngineClient)

Les informations envoyées du client au serveur se composent d'une requête (chaine de caractères), et d'une profondeur (variable numérique). Ces informations sont encodées en format json et envoyées via un Bufferer au serveur.

SearchEngineClient.core.Cient

```
//Ouverture de buffers d'ecriture et lecture pour comuniquer qvec le serveur
writer=new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
reader=new BufferedReader(new InputStreamReader(socket.getInputStream()));
writer.write(prepareMessage(query, depth) + "\n%END%\n");

//Codage de l'information en format Json
private String prepareMessage(String query, int depth){
   JsonObject jObject = new JsonObject();
   jObject.addProperty("query", query );
   jObject.addProperty("depth", depth );
   return jObject.toString();
}
```

4. Obtenir des informations auprès d'un ou plusieurs clients - (SearchEngineCrawler)

Pour gérer les demandes de plusieurs clients, une sous-routine (thread) est créé par chaque client qui se connecte au serveur, permettant l'envoi et la réception d'informations à l'aide de buffers. De tel manière qu'on peut avoir plusieurs clients de manière concurrente

SearchEngineCrawler.externalComunication.CrawlerSever

```
//Creation de soub-routine pour chaque client
Thread crawlerClient = new CrawlerClient(client, this.clientIndex);
crawlerClient.start();
```

5. Naviguer sur le Web pour obtenir les urls - (SearchEngineCrawler)

Pour chaque client qui envoie une demande (requête) au serveur, une ou plusieurs sous-routines sont exécutées, qui iront dans le web et recueilleront les informations. Ces sous-routine sont appelées webcrawlers et chacune commence par un chemin (url) différent. Il y a autant de webcrawlers comment chemins disponibles.

Pour obtenir les chemins initiaux une demande GET est faite à Google en utilisant le protocole HTTP et obtient ainsi le HTML de la page résultante (page de résultats Google) à partir de laquelle on obtient toutes les url importantes (celles contenues dans le <a> tags qui englobent le <h3> titres).

Les webcrawlers travaillent de façon récursive pour parcourir le web en profondeur, et ils le font de la manière suivante :
Supposons qu'on a un web crawlers avec les valeurs initiales suivantes

- Chemin initial http://www.page1.com
- Profondeur : 1

1. Ce webcrawler s'adente dans la page contenue dans l'url du chemin et obtient son code HTML (cela est fait fait grâce à une demande get a l'hôte de la page en utilisant des sockets)

2. Obtient les urls valides contenues dans les balises « a » de cette page (Ce processus se fait grâce à l'utilisation de la bibliothèque Jsoup, qui permet d'obtenir les informations de la page HTML en forme objet)
3. Ce processus est répété pour chacune de ces url obtenues dans la page
4. Chaque fois que le webcrawler entre dans une url, le compteur de profondeur diminue et continue comme ceci jusqu'à ce que ce compteur arrive à zéro
5. Si le compteur est zéro on enregistre les url de la page

SearchEngineCrawler.core.WebCrawler

```
//methode recoursif pour obtenir les url dans le web
public void indexSetOfURLS(String url, int depth) throws IOException{
   //si on peut aller plus profond
   if (depth > 0) {
      try {
      //on obtient toutes les url de la page et on appelle la méthode
      //de Nouveau avec un profondeur en moins
         for (Element element: getHyperlinks(url)) {
            String href = element.attr("href");
              if (href.startsWith("https")) {
                 indexSetOfURLS(href, depth - 1);
              }
            }
       }catch (Exception e) {
            System.out.println(e.getMessage());
       }
   //si on ne peut pas aller plus profond on stocke les url contenues
   //dans la page web
   }else{
       try {
           for (Element element: getHyperlinks(url)) {
                String href = element.attr("href");
                if (href.startsWith("https")) {
                   this.dbTriadWriter.insertTriad( new Triad(url, href,
   element.text() ) );
                }
              }
       } catch (Exception e) {
                       System.out.println(e.getMessage());
       }
    }
 }
```

6. Enregistrer les relations entre les url sous la forme de trios - (SearchEngineCrawler)

Ce processus se fait chaque fois que les webcrawlers obtiennent une url valide.

Les informations sont enregistrées dans la table triades d'une base de données appartenant au client. Ce tableau contient 3 colonnes qui sont (source url, url Destiny, mot clé) (Le gestionnaire de base de données utilisé est SQLite)

SearchEngineCrawler.core. TriadManager

```
//methode qui stocke dans le base de donnes du client la triade obtenue par un webcrawler
public synchronized void insertTriad(Triad triad) {
  String sql = "INSERT INTO triads (source, destiny, content) VALUES
(?,?,?)";
  try {
      PreparedStatement preparedStatement = this.dbConnectionManager.ge
      tConnection().prepareStatement(sql);
      if (this.dbConnectionManager.getConnection() != null)
            preparedStatement.setString(1, triad.getSource());
            preparedStatement.setString(2, triad.getCible());
            preparedStatement.setString(3, triad.getMot());
            preparedStatement.executeUpdate();
      }
   } catch (SQLException e) {
   // System.out.println(e.getMessage());
   }
}
```

Communication interne (SearchEngineCrawler - SearchEngineHits)

Comme le serveur a été divisé en 2 programmes (l'un en charge de la collecte d'informations et l'autre de l'exécution de l'algorithme d'indexation HITSon comunique ces deux programmes ave un protocole TCP / IP en utilisant des Sockets (étant le programme SearchEngineHits le serveur et le programme SearchEngineCrawler le client). Et aussi à travers des ressources partagées qui sont les bases de données des clients.

Moments de communication

- ✓ À la fin de la recherche sur le Web, le programme SearchEngineCrawler envoie un message au programme searchEngineHits indiquant le chemin de la base de données du client
  // SearchEngineCrawler.internalComunication. internalComunicationClient
  // SearchEngineHits.internalComunication. internalComunicationServer

- ✓ Le programme searchEngineHits lit le tableau « triads « dans la basse de donne du client (resource partage)
  // SearchEngineHits.core. DBCrawlerResultsReader

- ✓ À la fin de l'indexation des pages. Le programme SearchEngineHits envoie un message au programme SearchEngineCrawler indiquant l'en-tête de la table où les données ont été enregistrées au moment de l'indexation
  // SearchEngineCrawler.internalComunication. internalComunicationClient
  // SearchEngineHits.internalComunication. internalComunicationServer

- ✓ Le programme SearchEngineCrawler lit le tableau « hits_results » dans la basse de donne du client (resource partage)
  // SearchEngineCrawler.internalComunication. DBHitsResultsReader

7. Classer les url avec l'algorithme hits - (SearchEngineHits)

Pour la classification des url, telle que proposée dans les consignes, l'algorithme HITS a été utilisé.

Dans un premier temps, le programme SearchEngineHits est responsable de trouver la table de «triades» dans la base de données client pour après charger les urls obtenues dans la mémoire principale (on ne charge que les url différentes)

Pour le chargement en mémoire chaque url a été encapsulé dans un objet HitsResults, qui contiennent, l'url et deux variables numériques « hub » et « autorité » nécessaires pour l'implémentation de l'algorithme hubs. Et chacun de ces objets ont été mis dans différentes tables de hachage pour être en mesure d'y accéder avec différentes keys cela est fait par l'objet HitsResultsManager.

SearchEngineHits.core.HitsResult

```
public class HitsResult {
   //url de la page
   private String url;
   //valeur autority de la page
   private double authority;
   // valeur hub de la page
   private double hub;
   …
```

SearchEngineHits.core.HitsResultManager

```
public class HitsResultManager {
    // Table de hachage contenant toute les pages(sans répétition) trouvée par
    //le programme du crawler
    private Hashtable<String, HitsResult> pages;
    // Table de hachage contenant toutes les pages sources des pages cibles
    private Hashtable<String, HashSet<HitsResult>> sourcePages;
    // Table de hachage contenant toutes les pages cibles des pages sources
    private Hashtable<String, HashSet<HitsResult>>  destinyPages;
    …
```

HitsResultsManager Cet objet a 3 tables de hachage, qui permettent

- Accéder à un objet HitsResults à l'aide de son url
- Accéder à un ensemble d'éléments HitsResults par une source d'url (appelée IncomingNeighbors dans l'algorithme hits)
- Accéder à un ensemble d'éléments HitsResults par une dentiny url (appelé dans OutgoingNeighbors dans l'algorithme hits)

La première table est chargée directement au début de la classe, et les 2 autres sont chargées a fur et mesure que l'algorithme hits est exécuté

Lors de l'exécution de l'algorithme Hits, les valeurs de hub et d'autorité de chaque HitsResults est modifié.

À la fin, tous ces HitsResults sont enregistrés dans la base de données client dans une table appelée «hits_results »

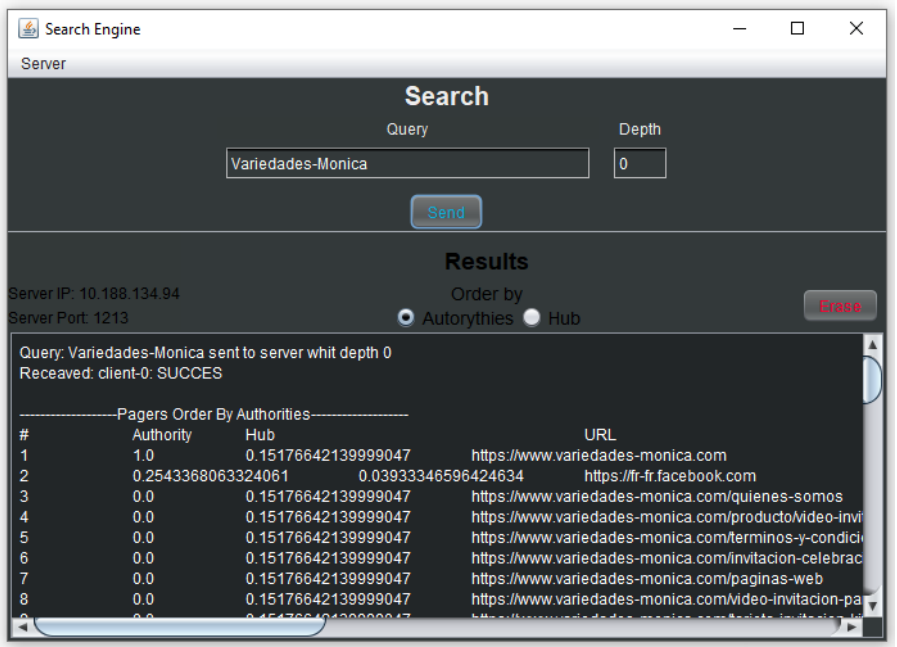
Envoyer les url classes au client(s) - (SearchEngineCrawler)

Lorsque le programme SearchEngineHits termine la classification des url, il envoie un signal au programme SearchEngineCrawler (en utilisant le socket ouvert) de sorte qu'il obtient les informations de la base de données, les convertit en format Json et les envois au client

```
//Envoi des pages classées au client sous forme de json
DBHitsResultsReader dbHitsResultsReader = new DBHitsResultsReader
(this.dbConnectionManager);
this.dbConnectionManager.connect();
String packageToSend = dbHitsResultsReader.getJsonHitsTable();
this.dbConnectionManager.close();
if(packageToSend!= null) {
    this.clientWriter.write( "client-"+this.clientIndex+": SUCCES\n");
    this.clientWriter.write(packageToSend);
    this.clientWriter.write( "\n%END%");
    this.clientWriter.flush();
}else {
  sendBadResult(0);
}
```

8. Afficher les résultats à l'utilisateur - (SearchEngineClient)

Une fois que le client reçoit les résultats du serveur, il est responsable de les décoder et de trier la liste d'urls par rapport à leur variable de hub ou leur variable d'autorithy (pour ce faire un ArrayList a était implemmente et on a utilisé l'interface comparable pour le trier)



Structure du projet

Comme expliqué ci-dessus le projet est divisé en 3 programmes, chaque programme a une structure Maven qui est en gros organisée comme suit.

SearchEngineClient

Ce programme a 2 paquets et une classe principale (Main) situé dans le paquet par défaut

core

Contient les classes nécessaires pour modéliser le client, se connecter pour servir et gérer les résultats du serveur

Classes:

- Client
- HitsResult HitsResult HitsResult HitsRes
- HitsResults
- ServerAnswer

view

Contient les classes nécessaires pour créer l'interface graphique

Classes :

- ClientWindow
- ServerEditionWindow

SearchEngineCrawler

Ce programme a 3 paquets et une classe principale (Main) situé dans le paquet par défaut

core

Contient les classes nécessaires pour gérer la base de données du client et récupérer des informations sur le Web

Classes:

- DBConnectionManager
- Triad
- TriadManager
- WebCrawler
- WebCrawlingManager

externalComunicaction

Contient les classes nécessaires pour communiquer avec le client et créer un client Google

Classes

- CrawlerClient
- CrawlerServer
- GoogleSearch (en)

internalComunicaction
Contient les classes nécessaires pour communiquer avec le programme SearchEngineHits

Classes:

- DBHitsResultsReader
- InternalComunicationClient (en)

SearchEngineHits:
Ce programme a 2 paquets et une classe principale (Main) situé dans le paquet par défaut
core
Contient les classes nécessaires pour gérer la base de données des clients, et exécuter l'algorithme de visites

Class

- DBConnectionManager
- DBCrawlerResultsReader
- DBHitsWriter (en)
- HitsImplementation
- HitsResult
- HitsResultManager

internalComunicaction
Contient les classes nécessaires pour communiquer avec le programme SearchEngineCrawler

Cours

- HitsClient
- InternalComunicationServer (en)

Tests et utilisation

1. Sur la machine serveur Exécuter le searchEngineCrawler.jar et SearchEngineHits programme.jar (de préférence en utilisant un terminal pour être en mesure de visualiser ce qui se passé)
2. Sur une autre machine (peut fonctionner aussi dans la même machine serveur), exécuter le programme SearchEngineClient.jar qui affichera la fenêtre graphique du client client
3. Dans cette fenêtre, vous devez modifier l'adresse IP du serveur en sélectionnant sur la barre d'outils Server>Edit et en ecrivant l'ip serveur correspondante et cliquez sur OK pour fermer la fenêtre. Remarquer que l'adresse IP a été bien modifiée dans la fenêtre principale (ne faites cette étape que si le client est en cours d'exécution sur une machine autre que le serveur)
4. Écrire la recherche et placer une profondeur dans les lieu correspondants (si la profondeur n'est pas placée ceci, n'est pas numérique ou est inférieure à 0, il sera envoyé comme 0 au serveur)
5. Un message indiquant que les données ont été envoyées doit être reflété sur l'écran
6. Alors vous serez en mesure de voir la réponse émise par le serveur
7. S'il s'agit d'une bonne réponse, cela peut être trie à l'aide des radios buttons disponibles
8. L'écran peut être complètement effacé à l'aide du bouton « clean »

Gestion de la concurrence

SearchEngineCrawler.core.TriadManager

//pour gérer la concurrence pendant l'insertion des donnes

```
public synchronized void insertTriad(Triad triad)
```

Améliorations implémentes

- Le projet compte avec une interface graphique pour le client
- Le code envoyé entre le client et le serveur est code en format Json ce qui permet là d'utiliser un autre langage de la programmation pour le client
- Il est possible choisir quand profond on peut s'adenter dans le web pour obtenir des informations en variant la variable depth
- L'utilisation de sqlite comme administrateur de basse de donne permet que le traitement des données soit plus efficace

Améliorations possibles

- Le projet SearchEngine est fonctionnel mais il n'est pas encore fini.
- Il est possible de créer des bibliothèques pour réutiliser le code. Par Example, les code de connexion aux basses de donnes. Ces améliorations se feront ultérieurement.
- Il est aussi possible d'améliorer le langage de communication entre le serveur et le client pour obtenir plus d'information de la partie du serveur au moment de la réponse de celui.
- La gestion des erreurs peut être plus efficace

Matériel Utilisée

Sistem d'exploitation pour la création du projet:

Debian GNU/Linux 10

IDE et structuration du projet

- NetBeans: utilise pour dessiner la partie graphique du client
- Atom : utilise pour faire le code en général
- Maven: utilisé pour la structuration du projet

Librerias externas

- Sqlite: Data base manager
- Gson: Pour coder et décoder en format Json
- Jsoup: Pour traite le code HTML comme un objet et avoir accès plus facilement a ses différents parties
- Socket – ServerSocket: Créer la communication entre le client et le serveur
- BufferedReader – BufferedWriter: Pour l'envoi et la réception des donnes
- Hashtable: Creer des structures pour organiser l'information en mémoire

Bibliographie

Implémentation et pseudocode de l'algorithme hits
https://en.wikipedia.org/wiki/HITS_algorithm

Utilisation de Jsoup
https://jsoup.org

Utilisation de Sqlite avec java
https://www.sqlite.org/index.html
https://www.sqlitetutorial.net/sqlite-java/

Java
https://docs.oracle.com/javase/7/docs/api/

Sockets
https://en.wikipedia.org/wiki/Socket