UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și tehnologia informației
SPECIALIZAREA:Tehnologia informației

Indexare și căutare

Proiect nr. 1 la disciplina
Regăsirea Informațiilor pe WEB

Profesor îndrumător:
Ș. l. dr. Ing. Alexandru Archip

Student
Emanuel Ursache
Grupa - 1410A

Iași, 2020

Cuprins

Capitolul 1. Prezentarea temei de proiect

Prima componenta de proiect a disciplinei „Regăsirea Informațiilor pe WEB” va propune să realizați un set de modificări asupra aplicațiilor dezvoltate pe parcursul laboratoarelor 1 – 4 astfel încât:

1. sa obțineți o procesare adecvată a cuvintelor determinate în cadrul unui text;

2. să studiați și alte tipuri de baze de date și mecanisme de stocare de date (precum MongoDB);

3. să studiați alte metode de realizare a operațiilor de căutare.

Astfel, pentru prima cerință de mai sus, metodele de construire a indecșilor direcți și inverși trebuie să implementeze mecanisme prin intermediul cărora un cuvânt de dicționar să fie adus la așa numita formă canonică. Indecșii astfel determinați, vor fi apoi stocați prin intermediul unor baze de date ne-relaționale, precum MongoDB. După cum a fost amintit în cadrul laboratorului nr. 4, căutarea booleană nu include mecanisme de determinare a unui eventual scor de relevanță pentru rezultatele identificate. A treia cerintă ne propune să studiem impactul distanței de tip cosinus asupra rezultatelor obținute de un motor de căutare.

Capitolul 2. Prezentarea generală a aplicației

Această aplicație este scrisă în limbajul de programare Java, utilizând mediul de dezvoltare Eclipse.

Proiectul este unul de tip Spring Maven.

Mave este un sistem de build și administrare a proiectelor. Face parte din proiectele găzduite de Apache Software Fundation.

Funcționalitățile sale principale sunt descrierea procesului de build al software-ului și descrierea dependențelor acestuia. Proiectele sunt descrise printr-unul sau mai multe fișier XML, denumite POM-uri (Project Object Model).

Aceste POM-uri au o structură implicită, ceea ce încurajează structurarea similară a proiectelor. POM-ul principal conține informații despre module, precum și despre dependențele proiectului (alte proiecte). Ordinea operațiunilor de build este definită prin declararea unor pluginuri folosite, din cadrul cărora unele goaluri sunt plasate și configurate în diferitele faze predefinite din ciclul de viață al unui build. Maven descarcă dinamic bibliotecile Java și pluginurile, din unul sau mai multe repository-uri.

Spring Framework oferă un model cuprinzător de programare și configurare pentru aplicații enterprise moderne bazate pe Java - pe orice tip de platformă de implementare. Un element cheie al Spring este suportul infrastructural la nivel de aplicație: Spring se concentrează pe „instalarea” aplicațiilor pentru întreprinderi, astfel încât echipele să se poată concentra pe logica de afaceri la nivel de aplicație, fără legături inutile cu medii de implementare specifice.

2.1. Interfața cu utilizatorul

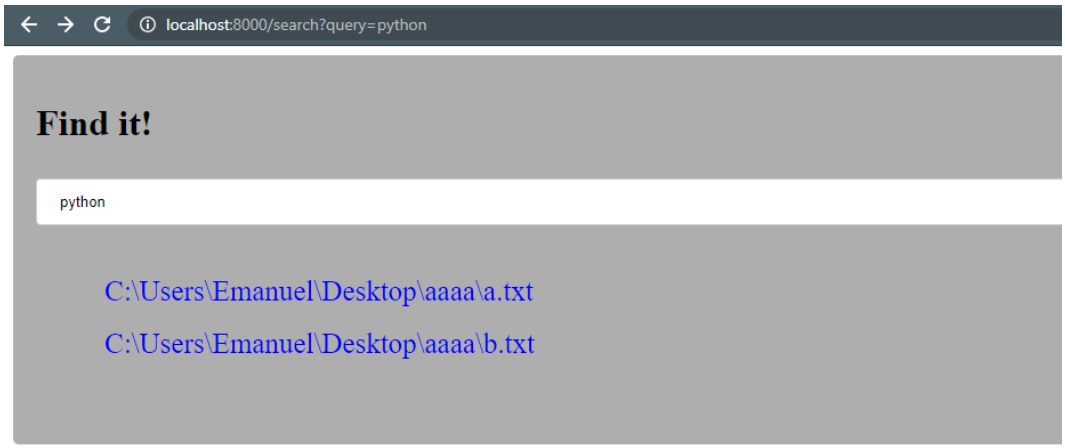
2.2. Exemplu de execuție a aplicației

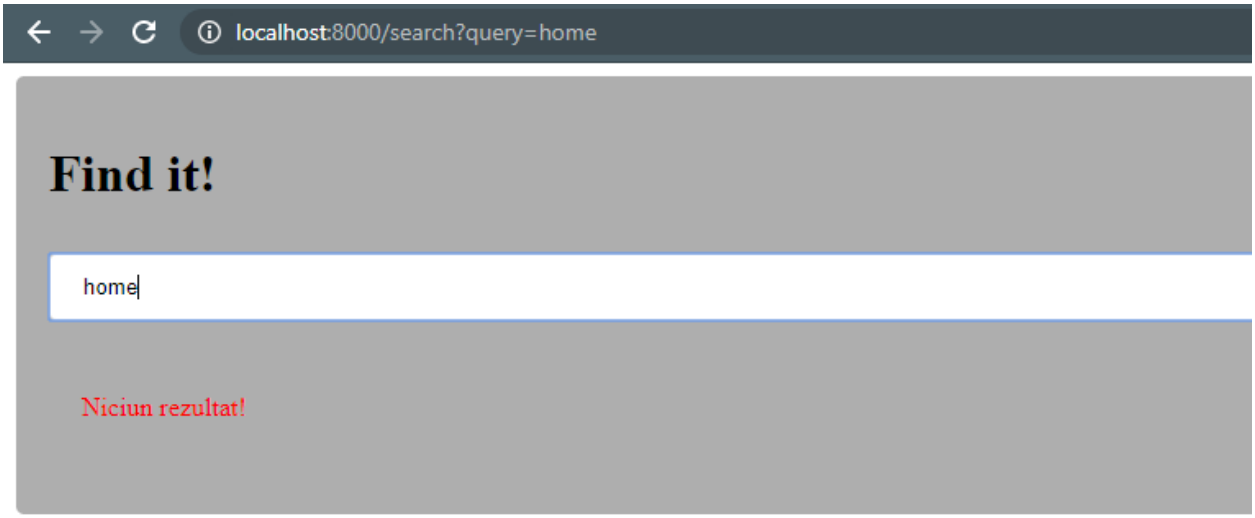
```
2020-04-23 22:58:55.723  INFO 15492 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8000 (http) with context path '
2020-04-23 22:58:55.728  INFO 15492 --- [           main] com.riw.proiect.riwApplication           : Started riwApplication in 6.357 seconds (JVM running for 6
2020-04-23 22:58:55.733  INFO 15492 --- [           main] org.mongodb.driver.cluster               : Cluster created with settings {hosts=[localhost:27017], mo
Server connection successfully to locahost on port 27017!
2020-04-23 22:58:55.739  INFO 15492 --- [localhost:27017] org.mongodb.driver.connection           : Opened connection [connectionId{localValue:2, serverValue:
2020-04-23 22:58:55.740  INFO 15492 --- [localhost:27017] org.mongodb.driver.cluster               : Monitor thread successfully connected to server with descr
Collection directIndex selected successfully!
Start indexing...
2020-04-23 22:58:55.767  INFO 15492 --- [           main] org.mongodb.driver.connection           : Opened connection [connectionId{localValue:3, serverValue:
Time elapsed for creating direct index: 0.034 seconds!
Time elapsed for creating inverse index: 0.003 seconds!
```

2.3. Exemplu de căutare

```
Start indexing...
2020-04-23 22:58:55.767  INFO 15492 --- [           main] org.mongodb.driver.connection
Time elapsed for creating direct index: 0.034 seconds!
Time elapsed for creating inverse index: 0.003 seconds!
2020-04-23 23:01:16.977  INFO 15492 --- [nio-8000-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]
2020-04-23 23:01:16.977  INFO 15492 --- [nio-8000-exec-1] o.s.web.servlet.DispatcherServlet
2020-04-23 23:01:17.009  INFO 15492 --- [nio-8000-exec-1] o.s.web.servlet.DispatcherServlet
Searching query : python
2 de rezultate afisate in browser!
Searching query : programming
2 de rezultate afisate in browser!
Searching query : home
Niciun rezultat!
```

Am căutat urmatoarele cuvinte: python, programming (care au fost găsite în fișierele text) și home (care nu a fost găsit)




localhost:8000/search?query=home
Find it!
home
Niciun rezultat!

Capitolul 3. Explicarea modulelor aplicației

3.1. Creare index direct + indice „tf”

Acest modul creează index-ul direct al tuturor documentelor HTML găsite în directoarele și subdirectoarele website-ului sursă. Pentru parcurgerea directoarelor, este folosită o coadă, deoarece se dorește o parcurgere secvențială și nu recursivă.

Textul rezultat este procesat caracter cu caracter, dinaceleași considerente în ceea ce privește viteza de lucru. Se extrag astfel toate cuvintele din textul obținut, iar acestea sunt trecute prin 3 filtre:

1. sunt testate contra unei liste de excepțiile = cuvinte ce prezintă interes, dar nu se găsesc în dicț ionar, așa încât trebuie tratate ca atare, în forma lor curentă.
2. se verifică dacă nu sunt de tip stopwords = cuvinte ce nu prezintă interes pentru relevanța documentelor rezultate din căutare; sunt ignorate, pur și simplu.
3. cuvintele de dicț ionar sunt trecute printr-un proces de stemming, utilizând algoritmul lui Porter; se aduc astfel la o formă de bază, eliminându-se terminațiile ce determină forme diverse ale aceleiași noțiuni.

Indicele „tf” este creat tot în această etapă, datorită ușurinței cu care se poate accesa fiecare document în parte, local.

3.2. Creare index indirect + indice „idf”

În această etapă, este creat index-ul indirect, ce se bazează, în mare, pe logica de funcționare a etapei precedente. Se preia fiecare fișier de index direct creat anterior pentru a obține lista de cuvinte din acel document. Fiecare cuvânt în parte este luat în considerare în crearea structurii finale de date.

Indicele „idf” este creat în această etapă, deoarece avem acces global la toate documentele, iar formula acestui indice utilizează numărul total de documente și numărul de documente în care apare un cuvânt.

Este cel mai eficient în a accesa aceste informații chiar atunci când este creat index-ul indirect. La fel ca la etapa anterioară, se obține un fișier de mapare ce va conține locația index-ului indirect pentru fiecare fișier în parte, deși se creează și unul global, cu toate cuvintele din sursele extrase.

3.3. Căutare vectorială

Principiul căutării, în aplicația propusă, este similar cu cel de la căutarea booleană. Se parsează interogarea utilizatorului și se împarte în cuvinte, utilizând aceleași 3 filtre pentru cuvintele rezultate.

Lista de cuvinte cheie este transformată, apoi, într-un vector de aceeași formă ca și cei creați în etapa anterioară, cu observația că indicele tf este local interogării! Cu alte cuvinte, interogarea este tratată ca un document în sine, așa încât tf-ul este calculat folosind interogarea ca și parametru pentru document

Algoritm 1 cautareVectoriala(D, q)

1: transforma fiecare document $d_j \in D$ in $\overrightarrow{d_j} = \{key : tf(key,d) \cdot idf(key)\}$
2: transforma interogarea $\overrightarrow{q} = \{key : tf(key,d) \cdot idf(key)\}$
3: **for all** $\overrightarrow{d_j} \in D$ **do**
4: calculeaza $s_j = similaritateCosinus(\overrightarrow{d_j}, \overrightarrow{q})$
5: **end for**
6: sorteaza documentele descrescator din punct de vedere al scorului anterior s_j
7: **return** setul relevant de documente