Ανάκτηση και Εξόρυξη Πληροφορίας

Πάσσος Δημήτριος : 2024201000083

Μπραούνι Γιοάνι : 2022201800131

Περιεχόμενα

Βιβλιοθήκες για την σωστή λειτουργεία του προγράμματος.

Το πρόγραμμα χρησιμοποιεί την βιβλιοθήκη lucene 8.0.0, την οποία μπορείτε να κατεβάσετε πηγαίνοντας στον ακόλουθο σύνδεσμο και επιλέγοντας το lucene-8.0.0.tgz (https://archive.apache.org/dist/lucene/java/8.0.0/). Ο λόγος που επιλέξαμε την συγκεκριμένη έκδοση είναι επειδή βρήκαμε ένα πρόγραμμα που μπορεί να οπτικοποιήσει το περιεχόμενο του ανεστραμένου ευρετηρίου. Το πρόγραμμα ονομάζεται luke και ο σύνδεσμος του είναι https://github.com/DmitryKey/luke/releases/tag/luke-swing-8.0.0.

Η υλοποίηση του προγράμματος έχει γίνει σε java έκδοση 17.0.1 και επίπεδο γλώσσας 17 (sealed types) και για IDE έχει χρησιμοποιηθεί το IntelliJ Community Edition (https://www.jetbrains.com/idea/download/#section=linux).

Οι βιβλιοθήκες που χρησιμοποιούνται στην εργασία και πρέπει να εισαχθούν είναι:

1. Lucene-8.0.0/common/lucene-analyzers-common-8.0.0.jar
2. Lucene-8.0.0/core/lucene-core-8.0.0.jar
3. Lucene-8.0.0/queryparser/lucene-queryparsers-8.0.0.jar
4. Lucene-8.0.0/queries/lucene-queries-8.0.0.jar
5. Lucene-8.0.0/memory/lucene-memory-8.0.0.jar
6. Lucene-8.0.0/highlighter/lucene-highlighter-8.0.0.jar

Αντίστοιχα το οπτικό κομμάτι της εργασίας το έχουμε υλοποιήσει σε javafx.

Εισαγωγή

Το πρόγραμμα χωρίζεται σε φακέλους, ένας για την λειτουργία του ευρετηριάσου, αναζήτησης, με το όνομα Server_Parsing και ένας με γραφικό περιβάλλον εν ονόματι U.I. Η επικοινωνία μεταξύ των δύο φακέλων γίνεται με μία κλάση server δηλαδή ακολουθούμε Client-Server μοντέλο. Απαραίτητο για την σωστή λειτουργία του προγράμματος είναι να ανοιχτούν αυτοί οι δύο φάκελοι χωριστά και όχι να γίνει άνοιγμα του φακέλου Project1_Passos_Braouni. Ο κώδικας είναι ανεβασμένος και στο github σαν προσωπική εργασία, άμα χρειαστείτε πρόσβαση παρακαλούμε επικοινωνήστε μαζί μας για να σας προσθέσουμε.

Οι κλάσεις του φακέλου Server_Parsing

Όπως αναφέραμε και προηγουμένως ο φάκελος αυτός περιέχει όλες τις λειτουργίες ευρετηριασμού και αναζήτησης που αξιοποιούνται από την βιβλιοθήκη lucene. Μέσα περιέχεται και ο αρχικός φάκελος που μας δόθηκε κατά την εκφώνηση της εργασίας (Reuters) καθώς και ο φάκελος όπου σώζονται ανεστραμμένα ευρετήρια του lucene (Index).

Κλάση TReSaFields

Η κλάση αυτή περιέχει στατικές και τελικές συμβολοσειρές που χρησιμοποιούνται κατά τον ευρετηριασμό και την αναζήτηση. Περιέχει τα πεδία places, people, title, body, filename. Η χρήση της κλάσης γίνεται για να αποφευχθούν τυχόν τυπογραφικά λάθη.

Κλάση TReSaIndex

Η κλάση TReSaIndex είναι υπεύθυνη για την εισαγωγή νέων δεδομένων στο ευρετήριο, για έλεγχο αν τα δεδομένα υπάρχουν ήδη στο ευρετήριο καθώς και για την διαγραφή κειμένων από το ευρετήριο.

Συνάρτηση close

Η συνάρτηση είναι υπεύθυνη για το κλείσιμο του IndexWritter. Χρησιμοποιείται πάντα μετά το τέλος χρήσης του εγγραφέα. Το μη κλείσιμο του εγγραφέα επιφέρει κλείδωμα δηλαδή ανικανότητα στην εγγραφή και διαγραφή δεδομένων από το ευρετήριο.

Συνάρτηση commit

Η συνάρτηση είναι υπεύθυνη για να κάνει flush τον inderWritter. Μπορεί να χρησιμοποιηθεί είτε η συνάρτηση commit κατά την ολοκλήρωση της χρήσης του indexWritter είτε η συνάρτηση close.

Συνάρτηση deleteFolderFromIndex

Η συγκεκριμένη συνάρτηση είναι υπεύθυνη για την μαζική διαγραφή αρχείων από το ευρετήριο, που είναι αποθηκευμένα σε έναν φάκελο. Αρχικά ελέγχει όλα τα αρχεία των φακέλων για το αν δεν είναι φάκελοι, κρυμμένα, ανύπαρκτα, με δικαιώματα ανάγνωσης και αν είναι txt. Αφού περάσουν οι έλεγχοι καλείτε η συνάρτηση fromUI (βλ παρακάτω).

Συνάρτηση createIndex

Η συνάρτηση κάνει τους ίδιους ελέγχους με την deleteFolderFromIndex για την εγκυρότητα των αρχείων που υπάρχουν στον φάκελο και έπειτα καλεί την συνάρτηση indexFile η οποία ξεκινάει την λειτουργία ευρετηριασμού (βλ παρακάτω).

Συνάρτηση createSingleIndex

Η συνάρτηση έχει την ίδια λειτουργία με την createIndex απλά χρησιμοποιείται για τον ευρετηριασμό ενός αρχείου και όχι φακέλου.

Συνάρτηση indexFile

Η συνάρτηση δημιουργεί ένα document αρχείο, το οποίο είναι ειδικό αρχείο της lucene που χρησιμοποιείται για να ευρετηριαστεί το εκάστοτε αρχείο ή αρχεία. Το document εισάγεται με την getDocument (βλ παρακάτω). Έπειτα περνάμε στον πρώτο έλεγχο της if η οποία ελέγχει αν υπάρχει ενεργό ευρετήριο που σε αυτήν την περίπτωση θα προσθέσει εκεί τα νέα δεδομένα ή αν δεν υπάρχει ευρετήριο να δημιουργήσει. Στην συνέχεια το αρχείο που πρόκειται να εισαχθεί ελέγχεται για το αν υπάρχει ήδη στο ευρετήριο χρησιμοποιώντας την συνάρτηση ένα hashset που βρίσκεται στην κλάση server. Σε περίπτωση που δεν βρει το όνομα του αρχείου στο hashet η συνάρτηση το εισάγει. Αν όμως βρει το αρχείο μέσα στο ευρετήριο τότε το αρχείο που είναι αποθηκευμένο διαγράφεται και εκχωρείται το νέο αρχείο.

Συνάρτηση getDocument

Η συνάρτηση μετατρέπει ένα αρχείο σε document και το επιστρέφει. Αρχικά παίρνει το αρχείο γραμμή-γραμμή, αν εντοπίσει το πεδίο </title> τότε το εισάγει στο document στο πεδίο title. Αντίστοιχη διαδικασία ακολουθείται και για τα πεδία places και people. Στην περίπτωση που δεν εντοπίσει κανένα από τα παραπάνω τότε οι νέες σειρές του εγγράφου εκχωρούνται σε έναν StringBuilder μέχρι να τελειώσει το αρχείο. Έπειτα εισάγονται στο αρχείο ο StringBuilder στο πεδίο body και το όνομα του αρχείου στο πεδίο filename.

Συνάρτηση deletingFiles

Δέχεται μία συμβολοσειρά, βρίσκει το αρχείο με αυτό το όνομα και καλεί την deletoDoc (βλ παρακάτω) για να διαγράψει το εκάστοτε αρχείο από το ευρετήριο.

Συνάρτηση fromUI

Η συνάρτηση κατασκευάστηκε για να χρησιμοποιείται από την main. Καλεί την deleteDoc και αν η διαδικασία είναι επιτυχής επιστρέφει true. Σε κάθε άλλη περίπτωση επιστρέφει false.

Συνάρτηση deleteDoc

Η συνάρτηση παίρνει ένα αρχείο και σκοπό έχει να το διαγράψει από το ευρετήριο. Δημιουργούμε ένα document με την getDocument εισάγοντας το αρχείο που επιλέγει ο χρήστης. Αν το όνομα του αρχείου υπάρχει στο hashset τότε δημιουργούμε Terms και εξάγουμε από document το εκάστοτε πεδίο. Καλούμε έπειτα την deleteDocuments (συνάρτηση της lucene) για την διαγραφή των Terms από το ευρετήριο. Αν το hashset δεν περιέχει το όνομα του αρχείου σημαίνει ότι το αρχείο που θέλουμε να διαγράψουμε δεν υπάρχει στο ευρετήριο. Η συνάρτηση επιστρέφει false και σταματάει. Στην συνάρτηση χρησιμοποιείται η χρήση της Boolean μεταβλητής foundError η οποία σε περίπτωση που ένα αρχείο δεν περιέχεται στο index του εμφανίζει μήνυμα λάθους.

Συνάρτηση isAlreadyIndexed

Η συνάρτηση ελέγχει αν ένα αρχείο βρίσκεται μέσα στο ευρετήριο. Αφού δημιουργήσουμε έναν DirectoryReader για να μπορούμε να διαβάσουμε το ευρετήριο, περνάμε στην δημιουργία ενός TermQuery το οποίο θα ελέγξει αν το όνομα του αρχείου που προσπαθούμε να βάλουμε υπάρχει ήδη στο ευρετήριο σωσμένο στο πεδίο filename. Αυτό το TermQuery το περνάμε σε ένα booleanquery που του λέμε ότι πρέπει να βρει αν αληθεύει αυτό που εισάγαμε στο termQuery, δηλαδή αν το όνομα του αρχείου που επρόκειτο να μπει στο ευρετήριο υπάρχει ήδη. Με τον indexSearcher πραγματοποιείται η αναζήτηση του BooleanQuery και σε περίπτωση που τα αποτελέσματα που μας επιστρέψει είναι μηδέν τότε σημαίνει ότι το αρχείο δεν έχει εκχωρηθεί. Αν όμως βρει αποτελέσματα θα επιστρέψει true δηλαδή ότι εντόπισε στο ευρετήριο αρχείο με το ίδιο όνομα. Η συνάρτηση αυτή δεν χρησιμοποιήθηκε καθώς η απόδοση της δεν ήταν επιθυμητή. Αντίθετα δημιουργήσαμε ένα hashset το οποίο θα εξηγήσουμε κατά την ανάλυση της κλάσης server.

Κλάση QuerySearch

Η κλάση είναι υπεύθυνη για την αναζήτηση query καθώς και την αναζήτηση των πιο συναφή άρθρων.

Constructor της κλάσης QuerySearch

Στον constructor αρχικοποιούμε τον indexReader για να διαβάζουμε από το ευρετήριο μας, τον indexSearcher για να ψάξουμε στο ευρετήριο μας. Επίσης επιλέγουμε οι βαθμολογίες να στηρίζονται σε tf-idf. Αυτό γίνεται επειδή ζητείται από την άσκηση, στα αρχεία του lucene προτείνεται το BM25Similarity. Να σημειωθεί ότι το ClasicSimilarity δεν βρίσκει τα πιο κοινά άρθρα χρησιμοποιώντας την διαφορά συνημίτονων αλλά χρησιμοποιεί ένα δικό του σύστημα βαθμολόγησης.

Συνάρτηση search

Η συνάρτηση ψάχνει και βρίσκει την φράση που έχει εισάγει ο χρήστης και επιστρέφει τα αποτελέσματα. Αρχικά αρχικοποιούμε ένα TopScoreDocCollerctor εδώ που θα μας επιστραφούν όλα τα αρχεία που περιέχουν τους όρους που αναζητήσαμε. Έπειτα χρησιμοποιούμε τα wrapper των αναλυτών που χρησιμοποιήσαμε και στον ευρετηριασμό για το εκάστοτε πεδίο και τα περνάμε σε ένα MultiFieldQueryParser. Το MultiFieldQueryParser κάνει αναζήτηση στα πεδία που επιθυμούμε, στην συγκεκριμένη περίπτωση σε όλα. Έτσι την συμβολοσειρά που έδωσε ο χρήστης την κάνουμε parse με το MultiFieldQueryParser και μας το επιστρέφει σαν query, διότι το lucene κάνει search με query και όχι με συμβολοσειρά. Έπειτα ψάχνουμε με τον indexSearcher το query και εισάγουμε τα αποτελέσματα στο TopScoreDocCollector. Τα αποτελέσματα αυτά τα εισάγουμε σε ένα πίνακα ScoreDoc που κρατάει και την βαθμολογία ομοιότητας και τον επιστρέφουμε.

Συνάρτηση searchFile

Η συνάρτηση ψάχνει και βρίσκει όμοια αρχεία με αυτό που εισήγαγε ο χρήστης και γυρίζει τα αποτελέσματα. Αρχικά εισάγουμε πάλι τα wrapper των αναλυτών και περνάμε και τις λέξεις τις οποίες θέλουμε να αγνοεί. Παίρνουμε την είσοδο του χρήστη και τον μετατρέπουμε σε document χρησιμοποιώντας την συνάρτηση getDocument που έχουμε στην κλάση ευρετηριασμού. Δημιουργούμε ένα αντικείμενο της κλάσης MoreLikeThis. Αυτή η κλάση είναι υπεύθυνη για την αναζήτηση ομοιοτήτων document και όχι term. Εισάγουμε στο αντικείμενο της κλάσης MoreLikeThis τον indexReader δηλαδή τον αναγνώστη ευρετηρίου, και δηλώνουμε και τα πεδία που θέλουμε να ψάξει. Ανεβάζουμε το maxClauseCount γιατί παρατηρήθηκε ότι επέστρεφε σφάλμα σε περιπτώσεις όπου το κείμενο ήταν αρκετά μεγάλο. Παίρνουμε όλα τα πεδία από το document του αρχείου που έδωσε ο χρήστης και τα εξάγουμε σε ένα StringBuilder. Χρησιμοποιώντας την κλάση preprocessor κάνουμε επεξεργασία της συμβολοσειρές εξάγοντας σημεία της στίξης και λέξεις όπου ο MultiFieldQueryParser χρησιμοποιεί για Boolean αναζητήσεις (π.χ το θαυμαστικό είναι ειδικός χαρακτήρας ο οποίος σημαίνει not και πρέπει να αφαιρεθεί). Χρησιμοποιείται και PorterStemmer κατά την επεξεργασία της συμβολοσειράς αλλά δεν εξάγονται οι πιο κοινές λέξεις. Έπειτα ακολουθείται η ίδια διαδικασία με την συνάρτηση search και επιστρέφουμε τα αποτελέσματα σε ένα ScoreDoc

Συνάρτηση getIndexSearcher

Επιστρέφει τον indexSearcher.

Συνάρτηση closeReader

Κλείνει τον αναγνώστη ευρετηρίου.

Κλάση Preprocessor

Η κλάση χρησιμοποιείται για επεξεργασία του κειμένου που έδωσε ο χρήστης κατά την αναζήτηση ομοιότητας άρθρων.

Constructor Preprocessor

Εισάγουμε την συμβολοσειρά που θέλουμε να επεξεργαστούμε

Συνάρτηση CurentLinePrep

Αφαιρούμε όλα τα σημεία τις στίξης από το κείμενο καθώς και τα tags των πεδίων. Επεξεργαζόμαστε κυρίως τους αριθμούς που υπάρχουν στο κείμενο. Κυρίως τους μεγάλους αριθμούς που είναι γραμμένοι χωρισμένοι με κόμμα. Τέλος αν η λέξη που περνάει είναι όντως λέξη και όχι αριθμός κάνουμε stemming χρησιμοποιώντας τον Porter Stemmer που διατίθεται από την βιβλιοθήκη lucene.

Συνάρτηση toString

Επιστρέφει τα αποτελέσματα της συνάρτησης CurrentLinePrep σε μία συμβολοσειρά

Συνάρτηση checkIfEmpty

Αφού αφαιρέσει όλα τα πεδία από το κείμενο του χρήστη ελέγχει αν το κείμενο είναι άδειο η όχι. Αν είναι άδειο επιστρέφει true

Κλάση TextFileFilter

Ελέγχει αν το αρχείο που μας δόθηκε είναι της μορφής .txt.

Κλάση Server

Η κλάση είναι υπεύθυνη για την επικοινωνία μεταξύ του server με το U.I

Συνάρτηση run

Η συνάρτηση εκκίνησης του server. Χρησιμοποιεί την διεύθυνση localhost και θύρα 5555.Η συνάρτηση ελέγχει αρχικά αν αυτό που μας προωθείται από το UI είναι κενό που σε αυτήν την περίπτωση δεν κάνει τίποτα. Σε περίπτωση που δεχθεί συμβολοσειρά η οποία περιέχει την κωδικοποίηση “@@@” τότε επιλέγουμε την συμβολοσειρά ξεκινώντας από το τρίτο (τέταρτο πρακτικά) χαρακτήρα της συμβολοσειρά δηλαδή προσπερνώντας το “@@@”. Καταλαβαίνουμε από την κωδικοποίηση ότι ο χρήστης θέλει να εισάγει ένα αρχείο στο ευρετήριό μας, και καλούμε από την main την singleFile και αφού η συνάρτηση ολοκληρωθεί προωθείται στον χρήστη το μήνυμα ότι το αρχείο του καταχωρήθηκε. Η ίδια λογική χρησιμοποιείται και για την προσθήκη φακέλου καλώντας την αντίστοιχη συνάρτηση από την κλάση main όπως και για την περίπτωση διαγραφή αρχείου και φακέλου από το ευρετήριο. Άμα έρθει συμβολοσειρά με την κωδικοποίηση “*&&” τότε παίρνουμε την συμβολοσειρά από τον τέταρτο χαρακτήρα και κάνουμε διαχωρισμό με το κενό διότι γνωρίζουμε ότι αφού σπάσουμε την συμβολοσειρά θα μείνουμε με το όνομα του αρχείου και τον αριθμό των πιο κοινών άρθρων. Επιστρέφεται από την συνάρτηση searchFileInIndex ένα HashMap με το όνομα του αρχείου καθώς και την βαθμολογία του. Τέλος σε κάθε άλλη περίπτωση καταλαβαίνουμε ότι πρόκειται για αναζήτηση query και καλούμε την συνάρτηση printSearchResults η οποία θα επιστρέψει ένα HashMap που περιέχει το όνομα του αρχείου και άλλο ένα HashMap (nested) όπου περιέχει το κομμάτι του άρθρου όπου βρέθηκε από το query και την βαθμολογία.

Συνάρτηση readSet

Η συνάρτηση χρησιμοποιείται κατά την εκκίνηση του προγράμματος και σκοπό έχει να ενημερώνει το HashSet με τα ονόματα των αρχείων που έχουν ήδη ευρετηριαστεί. Το HashSet χρησιμοποιείται ώστε να είμαστε σίγουροι ότι δεν θα ευρετηριασούμε το ίδιο αρχείο δύο φορές.

Κλάση TReSaMain

Όπως αναφέρει και το όνομα της η κλάση αυτή είναι η κύρια κλάση του προγράμματος, είναι υπεύθυνη για το κάλεσμα άλλων κλάσεων για ευρετηριασμό και αναζήτηση.

Συνάρτηση main

Η συνάρτηση main είναι η κύρια συνάρτηση του προγράμματος και δίνει στον χρήστη να επιλέξει την λειτουργία που θέλει να εκτελέσει μέσα από γραμμή εντολών. Η χρήση της συστήνεται μόνο για debugging καθώς το πρόγραμμα έχει κατασκευαστεί για να λειτουργεί με javafx

Συνάρτηση searchFileInIndex

Η συνάρτηση αξιοποιεί την συνάρτηση searchFile της κλάσης QuerySearch. Τα αποτελέσματα εισάγονται σε ένα HashMap ώστε να σταλθούν στον UI μέσω της κλάσης server.

Συνάρτηση initializeIndexWriter

Ορίζουμε αρχικά τον φάκελο ο οποίος θα χρησιμοποιηθεί για να σώσει τα αρχεία της η βιβλιοθήκη lucene. Στην αρχή δημιουργούμε μία λίστα από λέξεις τις οποίες θα προσθέσουμε έπειτα στους αναλυτές ώστε να μην σωθούν στο ανεστραμμένο ευρετήριο. Οι λέξεις αυτές είναι το places, people, title, body, reuter οι οποίες εμφανίζονται σε κάθε κείμενο. Έπειτα προσθέτουμε αυτή την λίστα στην ήδη υπάρχουσα λίστα της lucene (ENGLISH_STOP_WORDS_SET) η οποία περιέχει τις πιο συχνά χρησιμοποιούμενες αγγλικές λέξεις καθώς μας δίνουν μικρό σημασιολογικό περιεχόμενο.

Έπειτα περνάμε στην δημιουργία των αναλυτών, οι αναλυτές είναι υπεύθυνοι για την λημματοποίηση και την εξαγωγή συμβόλων. Για τα περιεχόμενα των places, people, title χρησιμοποιήσαμε τον StandardAnalyzer ο οποίος δεν κάνει λημματοποίηση τα συγκεκριμένα πεδία αλλά κάνει εξαγωγή συμβόλων. Ο λόγος που το θεωρήσαμε σωστό είναι ότι θέλουμε τα συγκεκριμένα πεδία που διαθέτουν λιγότερη πληροφορία να τα κρατήσουμε αυτούσια. Αντίθετα στο πεδίο body χρησιμοποιούμε τον EnglishAnalyzer ο οποίος κάνει λημματοποίηση και εξαγωγή συμβόλων.

Όλα τα παραπάνω τα εισάγουμε σε ένα PerFieldAnalyzerWrapper ώστε να χρησιμοποιηθεί ο κατάλληλος αναλυτής για το εκάστοτε πεδίο και τα περνάμε σε ένα IndexWriterConfig το οποίο είναι υπεύθυνο για τις ρυθμίσεις που έχουμε ορίσει. Τις ρυθμίσεις αυτές τις περνάμε έπειτα στο IndexWriter το οποίο είναι υπεύθυνο για την δημιουργία και εγγραφή κειμένων στο ευρετήριο.

Συνάρτηση createOneIndex

Χρησιμοποιείται για να ευρετηριάσει έναν φάκελο από αρχεία καλώντας συναρτήσεις της κλάσης TReSaIndex.

Συνάρτηση singleFile

Χρησιμοποιείται για να ευρετηριάσει έναν αρχείο καλώντας συναρτήσεις της κλάσης TReSaIndex.

Συνάρτηση folderDeletion

Χρησιμοποιείται για να διαγράψει έναν φάκελο αρχείων από το ευρετήριο, καλώντας συναρτήσεις της κλάσης TReSaIndex.

Συνάρτηση deleteSingleFileFromUI

Χρησιμοποιείται για να διαγράψει ένα αρχείο από το ευρετήριο καλώντας συναρτήσεις της κλάσης TReSaIndex.

Συνάρτηση simplePrint

Χρησιμοποιείται για να επιστρέψει από έναν πίνακα τις συμβολοσειρές. Χρησιμοποιείται στην συνάρτηση printSearchResults που θα αναλύσουμε παρακάτω

Συνάρτηση printSearchResults

Η συνάρτηση είναι υπεύθυνη για την επιστροφή ενός HashMap όπου περιέχει τα αρχεία που βρέθηκαν κατά απαίτηση του χρήστη, μαζί με την γραμμή που περιέχεται αυτήν η γραμμή καθώς και την τελική βαθμολογία. Αρχικά δημιουργούμε έναν HighLighter για να μας επιστρέψει τα σημεία όπου βρέθηκε το query και μέσω το Fragmenter περνάμε το σύνολο των χαρακτήρων που θέλουμε να επιστρέψει. Εμείς επιλέξαμε το τριάντα πέντε (35) θεωρώντας το ένα φυσιολογικό μέγεθος. Έπειτα έχοντας ήδη πάρει το ScoreDoc από την search την QuerySearch περνάμε σε ένα βρόγχο επανάληψης ώστε να εισάγουμε όλα τα αρχεία στο HashMap το οποίο θα επιστρέψουμε στον χρήστη. Έπειτα για κάθε αποτέλεσμα ψάχνουμε μέσω του HighLighter να βρούμε ποιο πεδίο του αρχείου περιέχει του query όπου ο χρήστης αναζήτησε το οποίο μαρκάρεται με <B><B>. Τέλος περνάμε σε ένα ακόμα HashMap το οποίο θα καταχωρήσουμε στο πρώτο HashMap το όνομα του αρχείου και την βαθμολογία ομοιότητας του εκάστοτε αρχείο σε σχέση με το query.

Οι κλάσεις του φακέλου U.I

Ο φάκελος αυτός περιέχει τις κλάσεις οι οποίες είναι υπεύθυνες για την απεικόνιση καθώς και για την επικοινωνία χρήστη και server.

Κλάση TReSaMain

Η κλάση αυτή αποτελεί το αρχικό παράθυρο που βλέπει ο χρήστης και είναι υπεύθυνη για την επικοινωνία με τον backend. Το γραφικό περιβάλλον θυμίζει αυτό της εταιρείας Google (πολύ πρωτότυπο) και ο χρήστης μπορεί από τα options να επιλέξει τι θέλει να κάνει. Άμα απλά δώσει ένα query και πατήσει search τότε το ευρετήριο, αν δεν είναι άδειο, θα του επιστρέψει τα αποτελέσματα ταξινομημένα σε φθίνουσα σειρά. Στις επιλογές του είναι και η εισαγωγή ή διαγραφή αρχείου/φακέλου στο ευρετήριο του συστήματος. Κατά την ολοκλήρωση εισαγωγής ή διαγραφής ο χρήστης ενημερώνεται με αντίστοιχο μήνυμα. Ο χρήστης έπειτα μπορεί να ανοίξει το εκάστοτε αρχείο πατώντας πάνω στο όνομά του. Στις επιλογές του πέρα από την εισαγωγή/διαγραφή αρχείων ο χρήστης μπορεί να συγκρίνει ένα άρθρο και να βρει παρόμοια. Αφού εισάγει του άρθρο θα του ζητηθεί να πληκτρολογήσει των αριθμό των κοινών άρθρων που θέλει να του επιστραφούν. Σε αυτήν την διαδικασία προσθέτουμε ένα άρθρο παραπάνω εμείς ώστε ο χρήστης να μπορεί να δει την βαθμολογία του ίδιου του άρθρου που εισάγει και να έχει μία γενικότερη εικόνα για την βαθμολογία όμοιων άρθρων. Πατώντας το κουμπί History μπορεί να δει το ιστορικό αναζητήσεων, τον αριθμό αποτελεσμάτων που έλαβε κατά την ολοκλήρωση της αναζήτησης καθώς και την ημερομηνία και ώρα που έκανε το αίτημα. Τέλος υπάρχει το πλήκτρο Boolean. Αν ο χρήστης δεν το επιλέξει μπορεί να αναζητήσει απλά query. Επεξεργαζόμαστε την επιλογή του προσθέτοντας πριν τα σημεία της στίξης ή ειδικούς χαρακτήρες την κάθετο (/). Άμα δεν γίνει αυτή η διαδικασία ο queryparser θα πετάξει λάθος καθώς αυτά τα σημεία αναζήτησης χρησιμοποιούνται για Boolean αναζήτηση. Αν επιλέξει όμως το πλήκτρο Boolean ο χρήστης δεν έχει αυτούς τους περιορισμούς και μπορεί να υποστηρίξει Boolean εκφράσεις. (βλ. Οδηγίες Χρήσης Boolean Query)

Κλάση TReSaResults

Είναι η κλάση που είναι υπεύθυνη για την παρουσίαση των αποτελεσμάτων του χρήστη όταν γίνεται αναζήτηση με query. Περιέχει ένα ObservableList το οποίο περνάμε σε ένα TableView.

Κλάση Articles

Είναι μία απλή κλάση που τα αντικείμενά της τα περνάμε στο TableView της TReSaResults.

Κλάση History

Περιέχει το ιστορικό αναζήτησης του χρήστη. Ο χρήστης μπορεί να διαγράψει σειρές από το ιστορικό πατώντας το πλήκτρο delete. Η History παίρνει τα δεδομένα διαβάζοντας ένα txt το οποίο βρίσκεται στην αρχή των φακέλων με όνομα searches.txt. Κατά την εκκίνηση της η κλάση ξαναδιαβάζει το αρχείο και εισάγει τα δεδομένα σε ένα ObservableList το οποίο περνάει σε ένα TableView. Η History αξιοποιεί αντικείμενα της κλάσης HistoryDummy.

Κλάση HistoryDummy

Κλάση που αξιοποιείται για την εισαγωγή αντικειμένων στην History. Περιέχει setters και getters.

Κλάση TReSaArticleCompare

Παρουσιάζει τα αποτελέσματα της σύγκρισης άρθρων. Η λειτουργία της είναι ίδια με τις προηγούμενες κλάσεις. Χρησιμοποιεί αντικείμενα της κλάσης Articles.