# Μηχανή αναζήτησης άρθρων της Wikipedia


**Γιαννικόπουλος Χαράλαμπος | Α.Μ. : 2417**

**Νικόλαος-Ορέστης Ντάλλας | Α.Μ. : 2507**

**Ο πηγαίος κώδικας της εργασίας βρίσκεται στον ακόλουθο σύνδεσμο:**

https://drive.google.com/drive/folders/1o5vZM6kAd8D1hAQZtiQO89gBFheZgVGR?usp=sharing

# Περιεχόμενα

# 1. Πρόλογος – Γενικές πληροφορίες

## 1.1 Παρουσίαση ζητουμένου

Στα πλαίσια του μαθήματος ΜΥΕ003-Ανάκτηση Πληροφορίας κληθήκαμε να υλοποιήσουμε μία μηχανή αναζήτησης πάνω σε επιλεγμένα άρθρα από την Wikipedia. Στην υλοποίηση έπρεπε να παρέχεται η δυνατότητα στον χρήστη να θέτει διαφορετικού τύπου ερωτήματα, καθώς και επιλογές για τον τρόπο προβολής των αποτελεσμάτων. Επιπλέον, να διατηρείται το ιστορικό αναζήτησης, με το οποίο θα μπορεί να αλληλεπιδράει ο χρήστης.

## 1.2 Σκοπός της εργασίας

Μετά την υλοποίηση αυτής της εργασίας θα βρισκόμαστε σε θέση να κατανοήσουμε καλύτερα τις παρακάτω διαδικασίες:

- Μαζική εξαγωγή χρήσιμης πληροφορίας από ανεπεξέργαστα δεδομένα
- Ευρετηριοποίηση δεδομένων
- Μορφοποίηση των δεδομένων για την παρουσίαση τους.

# 2. Αναλυτική παρουσίαση της εκπόνησης

## 2.1 Δημιουργία συλλογής

Για τη δημιουργία της συλλογής μας χρησιμοποιήσαμε έναν web scrapper, δικής μας υλοποίησης. Εκεί επαναληπτικά παίρνουμε τον πηγαίο κώδικα τυχαίων άρθρων από τη Wikipedia και αποθηκεύουμε μόνο το ωφέλιμο κομμάτι τους με σκοπό τη δημιουργία των εγγράφων, τα οποία θα ευρετηριοποιηθούν. Κατά την επιλογή των άρθρων εξασφαλίστηκε, ότι το καθένα θα βρίσκεται μόνο μία φορά στην συλλογή. Επιπλέον, στη συλλογή περιέχονται άρθρα από 3 κατηγορίες (*computer, history, politics)*, η οποία συμπληρώνεται από εντελώς τυχαία επιλεγμένα άρθρα. Το συνολικό πλήθος των άρθρων είναι 6000.

## 2.2 Διαδικασία ευρετηριοποίησης

Αφού ολοκληρώθηκε η δημιουργία της συλλογής ξεκίνησε η διαδικασία της ευρετηριοποίησης. Αρχικά, έγινε η μοντελοποίηση των δεδομένων σε αντικείμενα της κλάσης Document (η οποία παρέχεται από το API της Lucene) με πεδία τα οποία αναφέρονται στον τίτλο, στον σύνδεσμο και στο κυρίως περιεχόμενο του κάθε άρθρου.

Έπειτα, έγινε η ανάλυση στα πεδία των παραπάνω εγγράφων. Αυτή περιλαμβάνει διαδικασίες stemming, απαλοιφή stop words, επέκταση συνωνύμων, όπως αυτές παρέχονται από τον Standard Analyzer του API.
Η παραπάνω διαδικασία ολοκληρώνεται με την δημιουργία του ευρετηρίου για τα πεδία των άρθρων, καθώς και με την αποθήκευσή τους σε τοπικό επίπεδο (hard drive).

## 2.3 Αναζήτηση άρθρων

Για την ανάλυση του ερωτήματος κατά την αναζήτηση χρησιμοποιήθηκε ο ίδιος αναλυτής, από τον οποίο πέρασαν τα έγγραφα (documents).
Σε πρώτη φάση, με την έναρξη του προγράμματος ανακτάται από τον δίσκο το ευρετήριο. Αφού αναλυθεί το ερώτημα του χρήστη γίνεται η αναζήτηση στο ευρετήριο.
Στην υλοποίηση δίνεται η δυνατότητα για διαφορετικού τύπου ερωτήματα. Σε αυτά περιλαμβάνονται: η χρήση λέξεων κλειδιών, boolean, wildcard, φράσεων, ενώ υποστηρίζεται και η αναζήτηση ανά πεδίο (έχει αποκλειστεί το πεδίο link).
Τέλος, κάθε ερώτημα που θέτει ο χρήστης αποθηκεύεται με σκοπό την προβολή ενός ιστορικού αναζητήσεων στο οποίο περιλαμβάνονται και ερωτήματα, που τέθηκαν σε προηγούμενες εκτελέσεις του προγράμματος.

## 2.4 Τρόποι παρουσίασης αποτελεσμάτων

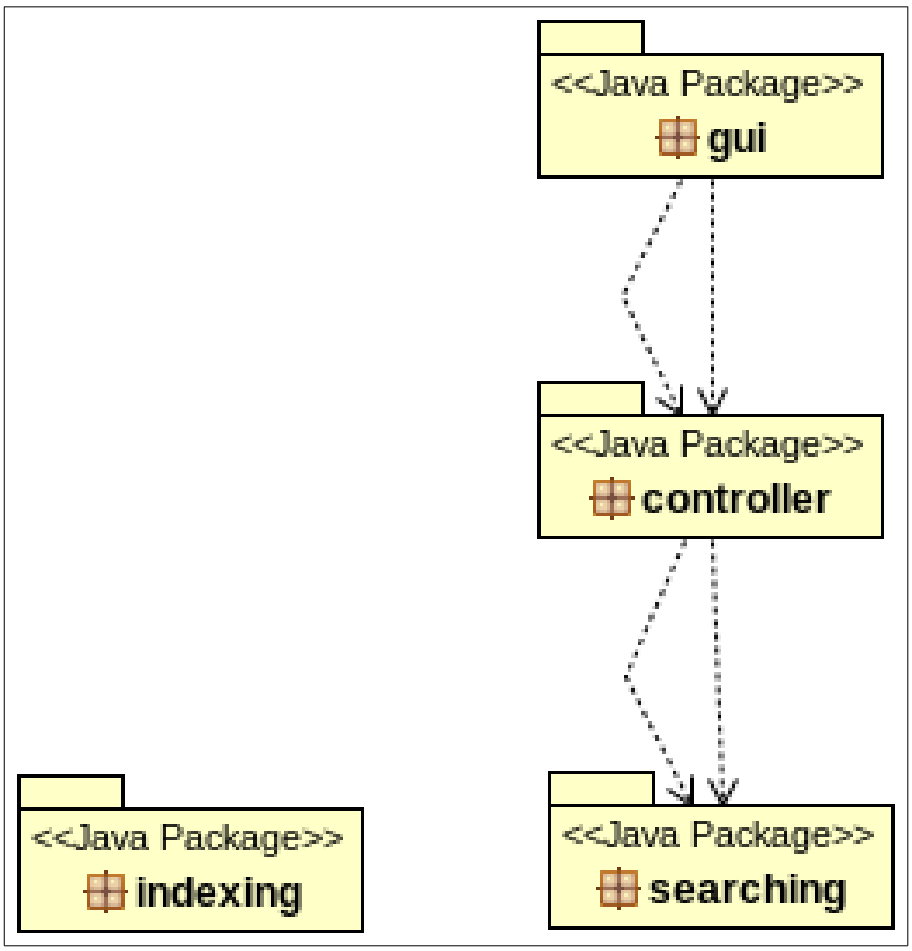
Στο κομμάτι της προβολής των αποτελεσμάτων χρησιμοποιήθηκε ο εξής τρόπος: Για κάθε ανακτημένο έγγραφο εμφανίζεται πρώτα ο τίτλος του άρθρου, μετά ένας λειτουργικός υπερσύνδεσμος που οδηγεί στο άρθρο στη σελίδα της Wikipedia και τέλος ένα απόσπασμα που περιέχει όρο του ερωτήματος του χρήστη. Τα παραπάνω, γίνονται σε περίπτωση που η αναζήτηση αφορούσε το κυρίως σώμα του άρθρου, διαφορετικά (αναζήτηση στον τίτλο) εμφανίζεται μόνο ο τίτλος και ο υπερσύνδεσμος.
Υποστηρίζονται 3 διαφορετικά είδη διάταξης των αποτελεσμάτων. Ο προεπιλεγμένος τρόπος είναι με βάση τη συνάφεια του εγγράφου σε σχέση με το ερώτημα του χρήστη. Παρέχεται επίσης η δυνατότητα για διάταξη βάσει μεγέθους των εγγράφων (μικρότερο → μεγαλύτερο) για το πεδίο που περιέχει το κύριο σώμα ή με αλφαβητική σειρά ως προς τον τίτλο.
Επιπλέον, ανάλογα με το πεδίο της αναζήτησης τονίζονται λέξεις οι οποίες αντιστοιχούν σε όρους του ερωτήματος.
Τέλος, δίνεται η επιλογή στον χρήστη να διατρέξει τα αποτελέσματα του ερωτήματός του με τη χρήση μίας μπάρας (Scroll bar).

## 2.5 Σχεδίαση Λογισμικού

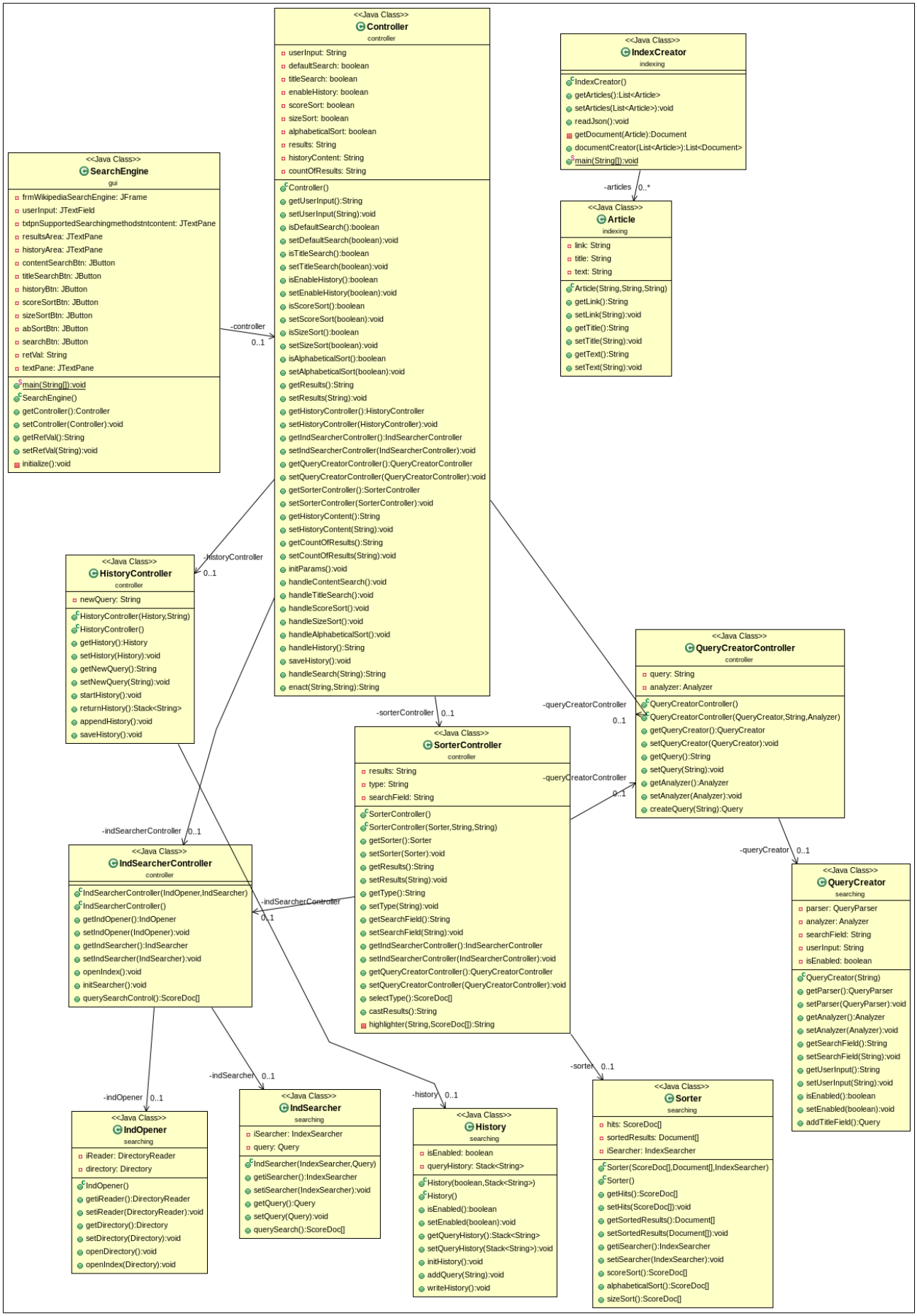
Όσον αφορά τη σχεδίαση του λογισμικού, το οποίο υλοποιεί την εφαρμογή, επιλέχθηκε η μέθοδος MVC ( Model-View-Controller). Ο κώδικας αναπτύχθηκε στις 3 παραπάνω κατηγορίες πακέτων.
Στο κομμάτι του Model το οποίο αφορά την επεξεργασία των δεδομένων από το σύστημα, περιέχονται τα πακέτα indexing και searching. Στο πρώτο γίνονται οι διαδικασίες της δημιουργίας του ευρετηρίου, ενώ στο δεύτερο υλοποιούνται αυτές που αφορούν την αναζήτηση αλλά και την παρουσίαση των αποτελεσμάτων.
Για τον Controller, που είναι υπεύθυνος για την επικοινωνία, αλλά και την ανεξαρτησία μεταξύ των Model και View, περιέχεται το ομώνυμο πακέτο controller.
Η αλληλεπίδραση του χρήστη με το σύστημα γίνεται μέσα από το πακέτο gui, που αποτελεί την συνιστώσα του View.

Η παραπάνω μέθοδος επιλέχθηκε λαμβάνοντας υπόψη τις δυνατότητες που παρέχει τόσο στο κομμάτι της συντήρησης του λογισμικού, όσο και σε αυτό της επεκτασιμότητάς του.

Στην Εικόνα 1 φαίνεται το διάγραμμα των πακέτων, ενώ στην Εικόνα 2 το διάγραμμα των κλάσεων.



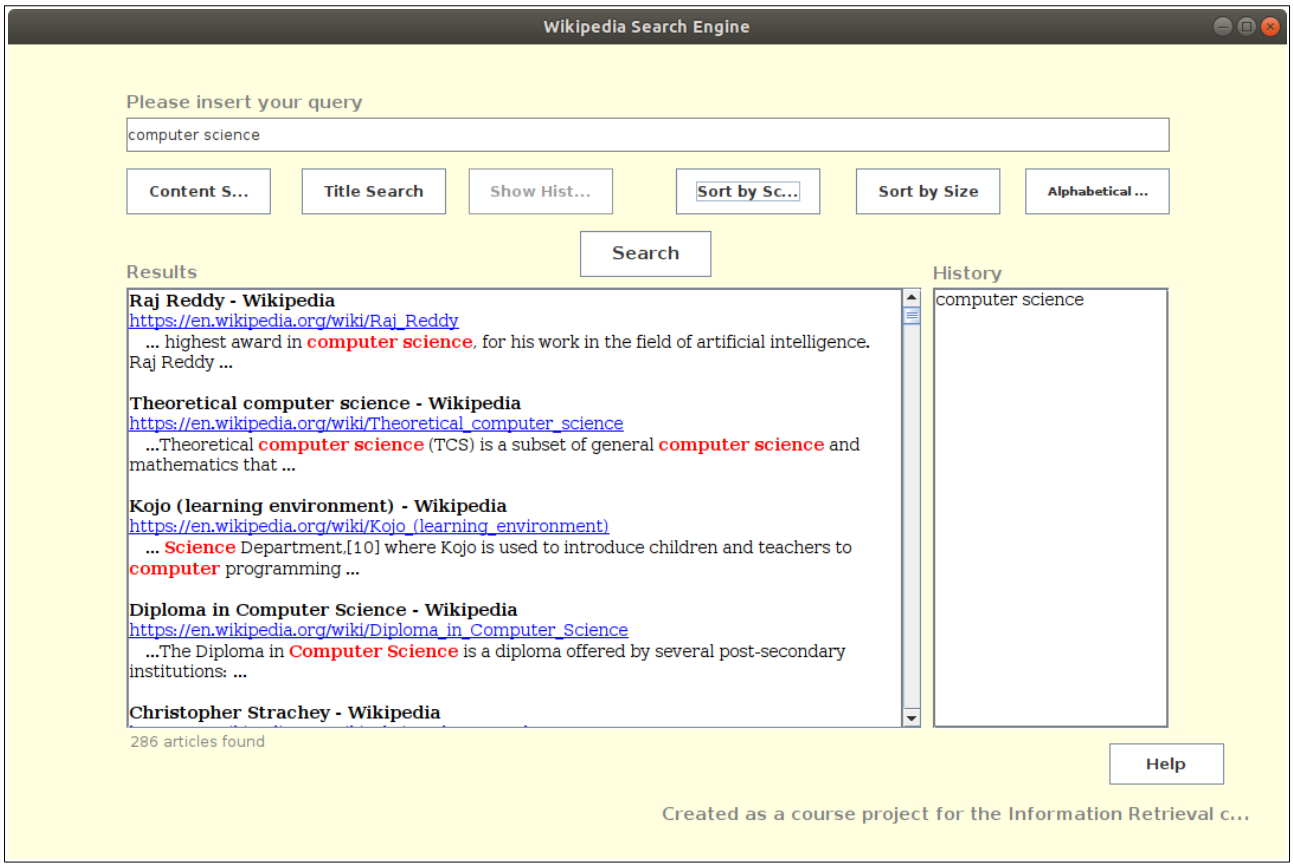

*Εικόνα 1*


<<Java Class>>
Controller
controller
userInput: String
defaultSearch: boolean
titleSearch: boolean
enableHistory: boolean
scoreSort: boolean
sizeSort: boolean
alphabeticalSort: boolean
results: String
historyContent: String
countOfResults: String
Controller()
getUserInput():String
setUserInput(String):void
isDefaultSearch():boolean
setDefaultSearch(boolean):void
isTitleSearch():boolean
setTitleSearch(boolean):void
isEnableHistory():boolean
setEnableHistory(boolean):void
isScoreSort():boolean
setScoreSort(boolean):void
isSizeSort():boolean
setSizeSort(boolean):void
isAlphabeticalSort():boolean
setAlphabeticalSort(boolean):void
getResults():String
setResults(String):void
getHistoryController():HistoryController
setHistoryController(HistoryController):void
getIndSearcherController():IndSearcherController
setIndSearcherController(IndSearcherController):void
getQueryCreatorController():QueryCreatorController
setQueryCreatorController(QueryCreatorController):void
getSorterController():SorterController
setSorterController(SorterController):void
getHistoryContent():String
setHistoryContent(String):void
getCountOfResults():String
setCountOfResults(String):void
initParams():void
handleContentSearch():void
handleTitleSearch():void
handleScoreSort():void
handleSizeSort():void
handleAlphabeticalSort():void
handleHistory():String
saveHistory():void
handleSearch(String):String
enact(String,String):String
<<Java Class>>
IndexCreator
indexing
IndexCreator()
getArticles():List<Article>
setArticles(List<Article>):void
readJson():void
getDocument(Article):Document
documentCreator(List<Article>):List<Document>
main(String[]):void
-articles
0..*
<<Java Class>>
Article
indexing
link: String
title: String
text: String
Article(String,String,String)
getLink():String
setLink(String):void
getTitle():String
setTitle(String):void
getText():String
setText(String):void
<<Java Class>>
SearchEngine
gui
frmWikipediaSearchEngine: JFrame
userInput: JTextField
txtpnSupportedSearchingmethodstntcontent: JTextPane
resultsArea: JTextPane
historyArea: JTextPane
contentSearchBtn: JButton
titleSearchBtn: JButton
historyBtn: JButton
scoreSortBtn: JButton
sizeSortBtn: JButton
abSortBtn: JButton
searchBtn: JButton
retVal: String
textPane: JTextPane
main(String[]):void
SearchEngine()
getController():Controller
setController(Controller):void
getRetVal():String
setRetVal(String):void
initialize():void
-controller
0..1
-historyController
0..1
<<Java Class>>
HistoryController
controller
newQuery: String
HistoryController(History,String)
HistoryController()
getHistory():History
setHistory(History):void
getNewQuery():String
setNewQuery(String):void
startHistory():void
returnHistory():Stack<String>
appendHistory():void
saveHistory():void
-sorterController
0..1
<<Java Class>>
SorterController
controller
results: String
type: String
searchField: String
SorterController()
SorterController(Sorter,String,String)
getSorter():Sorter
setSorter(Sorter):void
getResults():String
setResults(String):void
getType():String
setType(String):void
getSearchField():String
setSearchField(String):void
getIndSearcherController():IndSearcherController
setIndSearcherController(IndSearcherController):void
getQueryCreatorController():QueryCreatorController
setQueryCreatorController(QueryCreatorController):void
selectType():ScoreDoc[]
castResults():String
highlighter(String,ScoreDoc[]):String
-queryCreatorController
0..1
-queryCreatorController
0..1
<<Java Class>>
QueryCreatorController
controller
query: String
analyzer: Analyzer
QueryCreatorController()
QueryCreatorController(QueryCreator,String,Analyzer)
getQueryCreator():QueryCreator
setQueryCreator(QueryCreator):void
getQuery():String
setQuery(String):void
getAnalyzer():Analyzer
setAnalyzer(Analyzer):void
createQuery(String):Query
-indSearcherController
0..1
-indSearcherController
0..1
<<Java Class>>
IndSearcherController
controller
IndSearcherController(IndOpener,IndSearcher)
IndSearcherController()
getIndOpener():IndOpener
setIndOpener(IndOpener):void
getIndSearcher():IndSearcher
setIndSearcher(IndSearcher):void
openIndex():void
initSearcher():void
querySearchControl():ScoreDoc[]
-queryCreator
0..1
<<Java Class>>
QueryCreator
searching
parser: QueryParser
analyzer: Analyzer
searchField: String
userInput: String
isEnabled: boolean
QueryCreator(String)
getParser():QueryParser
setParser(QueryParser):void
getAnalyzer():Analyzer
setAnalyzer(Analyzer):void
getSearchField():String
setSearchField(String):void
getUserInput():String
setUserInput(String):void
isEnabled():boolean
setEnabled(boolean):void
addTitleField():Query
-indOpener
0..1
-indSearcher
0..1
-history
0..1
-sorter
0..1
<<Java Class>>
IndOpener
searching
iReader: DirectoryReader
directory: Directory
IndOpener()
getiReader():DirectoryReader
setiReader(DirectoryReader):void
getDirectory():Directory
setDirectory(Directory):void
openDirectory():void
openIndex(Directory):void
<<Java Class>>
IndSearcher
searching
iSearcher: IndexSearcher
query: Query
IndSearcher(IndexSearcher,Query)
getiSearcher():IndexSearcher
setiSearcher(IndexSearcher):void
getQuery():Query
setQuery(Query):void
querySearch():ScoreDoc[]
<<Java Class>>
History
searching
isEnabled: boolean
queryHistory: Stack<String>
History(boolean,Stack<String>)
History()
isEnabled():boolean
setEnabled(boolean):void
getQueryHistory():Stack<String>
setQueryHistory(Stack<String>):void
initHistory():void
addQuery(String):void
writeHistory():void
<<Java Class>>
Sorter
searching
hits: ScoreDoc[]
sortedResults: Document[]
iSearcher: IndexSearcher
Sorter(ScoreDoc[],Document[],IndexSearcher)
Sorter()
getHits():ScoreDoc[]
setHits(ScoreDoc[]):void
getSortedResults():Document[]
setSortedResults(Document[]):void
getiSearcher():IndexSearcher
setiSearcher(IndexSearcher):void
scoreSort():ScoreDoc[]
alphabeticalSort():ScoreDoc[]
sizeSort():ScoreDoc[]

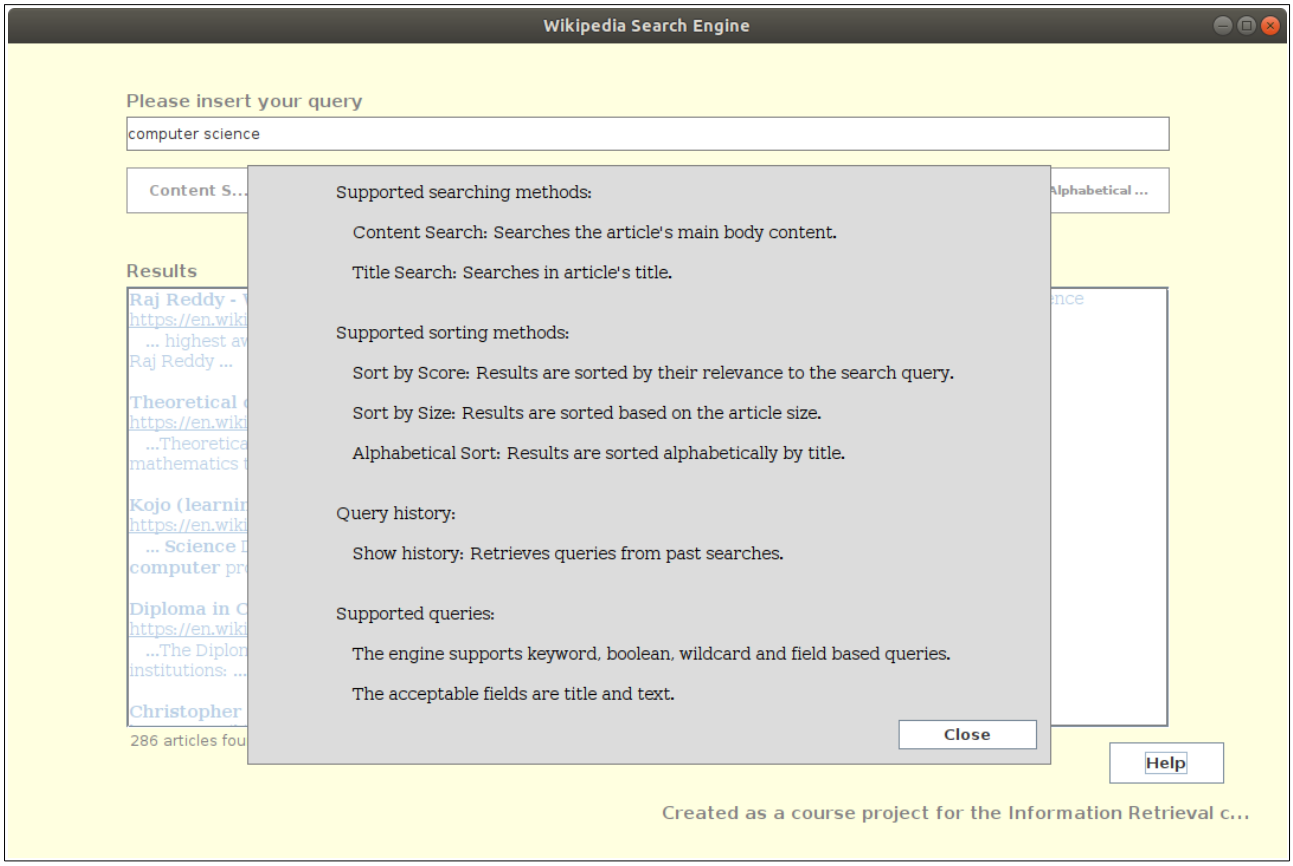
*Εικόνα 2*

## 3. Περιβάλλον αλληλεπίδρασης

Στο γραφικό περιβάλλον της εφαρμογής δίνονται στον χρήστη οι παρακάτω επιλογές. Στο πεδίο με ετικέτα που αναφέρεται στην εισαγωγή ερωτημάτων ο χρήστης γράφει το ερώτημα που θέλει να αναζητήσει. Ακολουθούν κουμπιά για την επιλογή του πεδίου αναζήτησης, ένα για το περιεχόμενο και ένα για τον τίτλο του άρθρου. Επίσης υπάρχουν διαθέσιμα 3 κουμπιά, υπεύθυνα για την διάταξη των αποτελεσμάτων. Η διαδικασία της αναζήτησης ενός ερωτήματος ολοκληρώνεται πατώντας το κουμπί αναζήτησης (*Search)*. Τέλος, υπάρχουν ακόμα 2 κουμπιά, το ένα είναι υπεύθυνο για την εμφάνιση του ιστορικού των προηγούμενων αναζητήσεων, ενώ το άλλο για την εμφάνιση ενός παραθύρου με μερικές οδηγίες για τη χρήση του συστήματος.
Η αρχική οθόνη της εφαρμογής συμπληρώνεται με 2 πεδία στα οποία φιλοξενούνται τα αποτελέσματα των αναζητήσεων και το ιστορικό.
Σε περίπτωση που βρέθηκαν συναφή έγγραφα, ο χρήστης ενημερώνεται και για το πλήθος τους. Σε αντίθετη περίπτωση εμφανίζεται σχετικό μήνυμα.
Ακολουθούν 2 στιγμιότυπα χρήσης στις εικόνες 3 και 4.




*Εικόνα 3*




*Εικόνα 4*

## 4. Δυνατότητες επέκτασης

Λαμβάνοντας υπόψη μελλοντικές χρήσεις αλλά και βελτιώσεις της εφαρμογής οδηγηθήκαμε, όπως αναφέρθηκε και νωρίτερα, σε αυτή την υλοποίηση.
Στο κομμάτι των βελτιώσεων, θα μπορούσε να γίνει χρήση των embeddings για την καλύτερη απόδοση στην διαδικασία της αναζήτησης. Επίσης θα μπορούσαμε να χρησιμοποιήσουμε διαφορετικού τύπου ευρετήρια, όπως για παράδειγμα ευρετήρια με πληροφορία και για τη θέση των όρων μέσα στο έγγραφο.
Από την άλλη, ως προς την επεκτασιμότητα, θα μπορούσαμε να αντικαταστήσουμε το γραφικό περιβάλλον της εφαρμογής, με τη χρήση μιας διαδικτυακής εφαρμογής κάνοντας διαθέσιμο το κομμάτι των Model και Controller σε έναν web server.

## 5. Αναφορές

Apache Lucene (v8.5.1): https://lucene.apache.org/
bs4: https://pypi.org/project/beautifulsoup4/
Swing Java: https://docs.oracle.com/javase/tutorial/index.html
Wikipedia: https://www.wikipedia.org
GSON: https://github.com/google/gson