ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΣΕΡΡΩΝ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ

ΕΞΟΡΥΞΗ ΣΤΟΙΧΕΙΩΝ ΣΧΕΤΙΚΩΝ ΜΕ ΤΗΝ ΑΓΟΡΑ ΕΡΓΑΣΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή εργασία του

Μπαρζόκα Βασίλειου (2304)

Επιβλέπων: Δρ. Ν. Πεταλίδης, Επιστημονικός Συνεργάτης

ΣΕΡΡΕΣ, ΝΟΕΜΒΡΙΟΣ 2012

Υπεύθυνη δήλωση

Υπεύθυνη Δήλωση: Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Πληροφορικής & Επικοινωνιών του Τ.Ε.Ι. Σερρών.

Σύνοψη

Η ανάγκη για τους επαγγελματίες μηχανικούς λογισμικού να έχουν ένα portfolio γνώσεων που να αντιστοιχεί όσο το δυνατόν περισσότερο στις ανάγκες της αγοράς εργασίας είναι τη σημερινή εποχή μεγαλύτερη από ποτέ. Για το λόγο αυτό είναι σημαντικό ο επαγγελματίας να γνωρίζει τόσο τις απαιτήσεις των εργοδοτών όσο και τις τάσεις που δημιουργούνται σε ό,τι αφορά σε απαιτούμενες ικανότητες και γνώσεις σε νέες τεχνολογίες.

Η συγκεκριμένη πτυχιακή αφορά στην ανάπτυξη μιας εφαρμογής σε μορφή Web Service η οποία:

- αναζητά και εξάγει από δημοφιλείς ιστοτόπους αγγελιών τα προσόντα και τις τεχνολογίες που αναζητούν οι εργοδότες
- κατηγοριοποιεί τα προσόντα που αναζητούνται (για παράδειγμα, μια απαίτηση για γνώσεις SQL Server εντάσσεται στην γενική κατηγορία Databases)
- παράγει χρήσιμα στατιστικά στοιχεία για τα αποτελέσματα, όπως οι τάσεις που επικρατούν ανα κατηγορία ή ανα χρονική περίοδο
- προσφέρει τα στατιστικά στοιχεία προς το κοινό μέσω ενός RESTful API

Περιεχόμενα

Κατάλογος πινάκων

Κατάλογος διαγραμμάτων

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κύριο Νικόλαο Πεταλίδη για την καθοριστική και πολύτιμη βοήθεια του σε όλη τη διάρκεια της εκπόνησης της πτυχιακής εργασίας και για τις γνώσεις που μου μετέδωσε κατά τη διάρκεια των σπουδών μου. Επίσης θα ήθελα να ευχαριστήσω τους γονείς μου Πούλιο και Αναστασία για την υποστήριξη που μου παρείχαν στις σπουδές μου.

Ορισμοί

Web Application Εφαρμογή που είναι προσβάσιμη μέσω του διαδικτύου και έχει διεπαφή χρήστη

Web Scraper Εφαρμογή που εξάγει δεδομένα απο ιστοσελίδες

Web Service Υποκατηγορία web application που συνήθως δεν έχει διεπαφή χρήστη και προορίζεται για επικοινωνία συστημάτων μέσω αυστηρά καθορισμένων προτοκόλλων

RESTful Web service που ακολουθεί το πρότυπο REST

JSON JavaScript Object Notation, μορφοποίηση δεδομένων για ανταλλαγή μέσω εφαρμογών

API Application Programming Interface, διεπαφή ανάμεσα σε 2 τμήματα λογισμικού

PHP Γλώσσα προγραμματισμού μέσω σεναρίων εντολών, χρησιμοποιείται κυρίως για εφαρμογές web και εκτελείται στον server

Javascript Γλώσσα προγραμματισμού μέσω σεναρίων εντολών, χρησιμοποιείται κυρίως για εφαρμογές web και εκτελείται στον client

Κεφάλαιο 1

Εισαγωγή

Η εξαγωγή της πληροφορίας από ιστοσελίδες είναι ένας τομέας με ολοένα και περισσότερο αυξανόμενο ενδιαφέρον λόγω φυσικά της χρησιμότητας των πληροφοριών, αλλά και του τρόπου που είναι δομημένες οι ιστοσελίδες έχοντας ως αποδέκτη της πληροφορίας που περιέχουν τον ανθρώπινο παράγοντα και όχι τα πληροφοριακά συστήματα.

Δημιουργείται έτσι η ανάγκη εξεύρεσης και δημιουργίας μεθόδων για την βέλτιστη αναγνώριση και εξαγωγή της πληροφορίας κάτι το οποίο αποτελεί το κύριο θέμα της παρούσης εργασίας. Δημιουργεί επίσης την ανάγκη ώστε κατά την αποθήκευση της η πληροφορία να είναι διαθέσιμη για επεξεργασία και ανάλυση από οποιοδήποτε άλλο ανεξάρτητο σύστημα.

Η πληροφορία που αναζητούμε είναι τα προσόντα που απαιτούνται από τους εργοδότες σε θέσεις του χώρου της πληροφορικής μέσω αγγελιών δημοσιευμένων σε εξειδικευμένες ιστοσελίδες προσφοράς και ζήτησης εργασίας. Έχοντας συγκεντρώσει έναν κατάλογο με περισσότερα από 2.000 προσόντα, σε μορφή λέξεων κλειδιών, προσπαθούμε να σχηματίσουμε μια όσο το δυνατόν πληρέστερη αντίληψη για την ζήτηση που επικρατεί στην Ελλάδα αλλά και στο εξωτερικό.

Για περισσότερη ανάλυση και ευκολότερη αναζήτηση τα προσόντα έχουν χωρισθεί σε κατηγορίες σύμφωνα με τον ευρύτερο τομέα εξειδίκευσης που ανήκουν και για κάθε ένα από αυτά αποθηκεύεται και η ημερομηνία δημοσίευσης του καθώς και η ιστοσελίδα που δημοσιεύθηκε, δίνοντας τη δυνατότητα δημιουργίας χρήσιμων στατιστικών δεδομένων για τις τάσεις που επικρατούν ανα κατηγορία, ανα χρονική περίοδο και ανα χώρα δημοσίευσης.

Φυσικά αυτά τα δεδομένα δεν έχουν αξία αν δε μπορούν να τα δουν όσοι από τον χώρο της πληροφορικής αναζητούν εργασία ή θέλουν να είναι συνεχώς ενημερωμένοι γύρω από τις τάσεις που επικρατούν ανα πάσα στιγμή. Για τον λόγο αυτό η εργασία έχει ως παράλληλο στόχο τη δημιουργία μιας υπηρεσίας ιστού η οποία δέχεται ερωτήματα προκαθορισμένης σύνταξης μέσω της διεπαφής της και προσφέρει τα στατιστικά δεδομένα σε οποιοδήποτε σύστημα συνδεθεί με αυτήν, καταστώντας την έτσι το μέσο ενημέρωσης των ενδιαφερομένων χρηστών μέσω της παροχής των εξαγόμενων δεδομένων.

1.1 Σχετικές εργασίες

Το γεγονός ότι οι γνώσεις που απαιτούνται από την αγορά εργασίας αποτελεί έναν τομέα με συνεχές ενδιαφέρον, έχει συμβάλλει στην δημιουργία παρόμοιων προσπαθειών στο παρελθόν, κάποιες από τις οποίες είναι ιδιαίτερα αξιόλογες και συνεχίζονται μέχρι σήμερα. Για παράδειγμα τα αποτελέσματα της εργασίας με όνομα *Degree-Oriented Guide to Skills in Information Technology* (Aken 2012) , (Litecky κ.ά. 2010) που είναι δημοσιευμένη στην ιστοσελίδα *www.dogs-it.org/* έχει διαχωρισμένα τα προσόντα ανα κατηγορία εκπαίδευσης και εξειδίκευσης, για τα οποία προσφέρει στατιστικά στοιχεία εμφάνισης τους και τις τάσεις που ακολουθούν έχοντας μεγάλες ημερολογιακές περιόδους προς επιλογή από τον χρήστη.

Ένα ακόμη ιδιαίτερα δημοφιλές site είναι το *www.indeed.com/jobtrends* το οποίο έχει στατιστικά στοιχεία για πολλά δημοφιλή προσόντα, χωρίς όμως να είναι κατηγοριοποιημένα ανα τομέα εξειδίκευσης, κατηγορία ή χώρα. Το κυριότερο χαρακτηριστικό του είναι ότι εκτός από τα στατιστικά εμφάνισης των προσόντων προσφέρει και την δυνατότητα προβολής των αγγελιών που ζητούν το κάθε ένα από αυτά.

1.1.1 Διαφοροποίηση

Η εργασία μας διαφοροποιείται από τις υπάρχουσες σχετικές εργασίες σε 2 κύρια θέματα:

1. Η κατηγοριοποίηση των προσόντων σε επίπεδο τομέα εφαρμογής, όπου η κάθε κατηγορία περιέχει όσο το δυνατόν περισσότερα προσόντα υπάρχουν. Παραδείγματα κατηγοριών είναι: *"γλώσσες προγραμματισμού", "βάσεις δεδομένων"*.

2. Η δυνατότητα προσφοράς των στατιστικών δεδομένων δυναμικά, μέσω του API ενός RESTful Web Service, ανα πάσα στιγμή μέσω διαφορετικών προκαθορισμένων ερωτημάτων, προσαρμόσιμα στις ανάγκες των χρηστών.

Έτσι προσφέρουμε ένα καινοτόμο προϊόν λογισμικού στην κατηγορία της αναζήτησης προσόντων γύρω από τον τομέα της Πληροφορικής, ευελπιστώντας να καταστεί ικανό να καλύψει τις ανάγκες κατάρτισης των σύγχρονων επαγγελματιών αλλά και γενικότερα των ανθρώπων που ασχολούνται και ενδιαφέρονται για αυτόν τον χώρο και τις εξελίξεις του. Προσφέροντας όσο το δυνατόν εγκυρότερα αποτελέσματα και έχοντας κατά νου την εύχρηστη και απρόσκοπτη αλληλεπίδραση με τους χρήστες.

1.2 Δομή της εργασίας

Στις επόμενες ενότητες θα γίνει ανάλυση του τρόπου ανάπτυξης και υλοποίησης της εργασίας, αφού αρχικά γίνει μια βασική επεξήγηση θεωρητικών όρων και εννοιών που χρησιμοποιήθηκαν κατά τη διάρκεια της υλοποίησης της.

Στην ενότητα 2 γίνεται αρχικά μια θεωρητική ανάλυση γύρω από τις μεθόδους εξαγωγής πληροφορίας από ιστοσελίδες που χρησιμοποιήθηκαν και διάφορες άλλες παραλλαγές υλοποίησης. Επίσης στην υποενότητα 2.2 γίνεται αναφορά σε κάποια συστήματα που χρησιμοποιήθηκαν και ενσωματώθηκαν στην εφαρμογή, όπως το ORM Framework. Τέλος στην υποενότητα 2.3 αναφέρονται έννοιες γύρω από τις υπηρεσίες ιστού και την αρχιτεκτονική REST.

Στην ενότητα 3 γίνεται μια περιγραφή του συστήματος παρουσιάζοντας τις απαιτήσεις μαζί με το τι κάνει η εφαρμογή μέσα από σενάρια χρήσης και διαγράμματα σεναρίων χρήσης UML, χωρισμένη σε 3 υποενότητες όπου στην κάθε μια αναλύεται

το κάθε ένα ανεξάρτητο υποσύστημα ξεχωριστά. Στην 3.1 ο web scraper, στην 3.2 το web service και στην 3.3 το περιβάλλον διαχείρισης των 2 υποσυστημάτων.

Στην ενότητα 4 αναλύεται ο τρόπος, η σχεδίαση και οι μέθοδοι υλοποίησης των υποσυστημάτων με διαγράμματα ακολουθίας και κλάσεων μαζί με μερικά παραδείγματα σημαντικών σημείων κώδικα.

Στην ενότητα 5 παρουσιάζονται τα αποτελέσματα της εργασίας μέσα από στατιστικά δεδομένα, χωρισμένα ανά κατηγορία και ανά μήνα.

Τέλος στην ενότητα 6 παρουσιάζονται προσωπικά συμπεράσματα από την εφαρμογή και προτάσεις περαιτέρω μελλοντικής εξέλιξης της.

Κεφάλαιο 2

Θεωρητική ανάλυση

2.1 Web Page Scraping

Οι ιστοσελίδες που υπάρχουν στο διαδίκτυο περιέχουν μεγάλη ποσότητα πληροφορίας η οποία είναι δημιουργημένη σε δομές έτσι ώστε να προορίζεται για ανάγνωση και προσκόμιση από τον άνθρωπο. Για να μπορέσουμε να καταφέρουμε να επεξεργαστούμε αυτές τις δομές πληροφορίας μέσω ενός λογισμικού θα πρέπει πρώτα να τις μετατρέψουμε σε κατάλληλες δομές ώστε να είναι επεξεργάσιμες από τον υπολογιστή και έπειτα να μπορεί να εξαχθεί η πληροφορία που αναζητούμε μέσα από αυτές. Η τεχνικές και οι διαδικασίες εξαγωγής πληροφορίας από ιστοσελίδες ονομάζεται Web Page Scraping και εν συντομία το πρόγραμμα που υλοποιεί αυτές τις διαδικασίες Scrapper .

Η διαδικασία του web scraping περιέχει αρκετές δυσκολίες όπως:

- Την εύρεση ενός επαναλαμβανόμενου προτύπου (pattern) στον κώδικα HTML μιας ιστοσελίδας που να επιτρέπει τον εντοπισμό και τον διαχωρισμό των ορίων της πληροφορίας που θέλουμε να εξάγουμε.
- Το σχέδιο αυτό είναι μοναδικό για κάθε ιστοσελίδα, που σημαίνει ότι για την κάθε μια ιστοσελίδα από την οποία θέλουμε να εξάγουμε πληροφορίες πρέπει να βρεθεί διαφορετικό πρότυπο.
- Αν η σχεδίαση της ιστοσελίδας αλλάξει, πρέπει να αλλάξει και το πρότυπο.
- Πολλές φορές η φυσική γλώσσα περιέχει χαρακτήρες και σημεία στίξης που καθιστούν δυσκολότερο τον εντοπισμό και τον διαχωρισμό των λέξεων

Για τους λόγους αυτούς δεν υπάρχει κάποιο πρότυπο με το οποίο πρέπει απαραίτητα να υλοποιηθεί ένας scrapper, αφού το πρόβλημα που καλείται να επιλύσει είναι εξαιρετικά διαφορετικό σε κάθε περίπτωση. Ωστόσο υπάρχουν κάποιες μέθοδοι που εφαρμόζονται συνήθως στα περισσότερα προβλήματα και αποδίδουν πολύ καλά αποτελέσματα. Στις επόμενες ενότητες παρουσιάζονται οι μέθοδοι που χρησιμοποιήθηκαν στην εργασία.

2.1.1 Η γλώσσα XPath

Η γλώσσα XPath (XML Path) χρησιμοποιείται στην πλοήγηση και αναζήτηση δεδομένων μέσα από έγγραφα που βρίσκονται σε μορφή XML. Πρόκειται για ένα σύνολο συντακτικών και σημασιολογικών κανόνων που παραπέμπουν σε τμήματα των εγγράφων XML (W3C 2010).Η χρήση της αποτελεί ένα σημαντικό εργαλείο στην απομόνωση της πληροφορίας από το περιεχόμενο των ιστοσελίδων, προκειμένου έπειτα να εφαρμοστούν περισσότερο στοχευμένες τεχνικές εξόρυξης δεδομένων από αυτές.

Στην XPath ένα έγγραφο XML διαχειρίζεται ως ένα δέντρο (tree) από κόμβους (nodes), οι οποίοι είναι 7 (element, attribute, text, namespace, processing-instruction, comment, document nodes). Το ανώτερο στοιχείο στο δέντρο ονομάζεται ρίζα (root) του εγγράφου.Ένα σετ από κόμβους μπορεί να εμπεριέχει κανέναν ή και περισσότερους από έναν κόμβους. Για παράδειγμα το παρακάτω έγγραφο XML

```
1 <?xml version="1.0" encoding="UTF-8"?>
2  <list>
3     <item new='true'>
4       <title land="en">DVD Music</title>
5       <author>K. A. Bred</author>
6       <year>2012</year>
7       <price currency="USD" >29.99</price>
8     </item>
9  </list>
```

Listing 2.1: Παράδειγμα εγγράφου XML

περιέχει τους εξής κόμβους:

```
1 <list> (root element node)
2 <author>K. Bred</author> (element node)
3 lang”=”en,  new’=’true (attribute nodes)
```

Listing 2.2: Ανάλυση κόμβων εγγράφου XML

Οι εκφράσεις της XPath επιτρέπουν την αναφορά σε στοιχεία και χαρακτηριστικά ενός εγγράφου σε μορφή XML καθώς όμως και HTML, που είναι και αυτό που μας ενδιαφέρει περισσότερο στην περίπτωση που κάνουμε αναζήτηση σε ιστοσελίδες. Για την αναφορά αυτή χρησιμοποιείται το σύμβολο ’@’ , για παράδειγμα η ακόλουθη έκφραση XPath προσδιορίζει τα στοιχεία των οποίων η τιμή συναλλάγματος περιλαμβάνει την τιμή USD:

```
1 /list/item/price[@currency=”USD”]
```

Listing 2.3: Παράδειγμα έκφρασης XPath

Ωστόσο οι ιστοσελίδες δεν ακολουθούν την δομή XML, αλλά την μορφοποίηση με HTML. Αυτό δεν μας εμποδίζει να κάνουμε αναζήτηση μέσα στο περιεχόμενο τους, καθώς η XPath υποστηρίζει και άλλα χαρακτηριστικά, όπως αναζήτηση με βάση το CSS ID ή Class ενος στοιχείου HTML ή ακόμη και τον τύπο του. Για παράδειγμα η παρακάτω έκφραση υλοποιημένη σε γλώσσα PHP αναζητά μέσα σε ένα HTML έγγραφο το στοιχείο του οποίου το ID είναι ”advertisment”

```
1 $xpath->query(”//*[@id=’advertisment’]”)->item(0);
```

Listing 2.4: Παράδειγμα έκφρασης XPath σε PHP

Με ανάλογο τρόπο μπορούμε να αναζητήσουμε οποιοδήποτε στοιχείο μας ενδιαφέρει, απομονώνοντας έτσι την πληροφορία από το υπόλοιπο έγγραφο.

2.1.2 Regular Expressions

Μια γλώσσα, όπως η φυσική γλώσσα, αποτελείται από λέξεις όπου η κάθε λέξη αποτελείται από διάφορα σύμβολα, τα οποία αποτελούν το αλφάβητο αυτής της γλώσσας.Για παράδειγμα η Ελληνική γλώσσα αποτελείται από τα σύμβολα α, β, γ, δ, ε ... χ, ψ, ω τα οποία αποτελούν το αλφάβητο της.Αν έχουμε ένα σύνολο από λέξεις τότε μπορούμε να πούμε ότι αυτές σχηματίζουν μια πρόταση ή ένα μήνυμα το οποίο έχει συγκεκριμένο σημασιολογικό περιεχόμενο (Raymond Greenlaw 1998). Ένας τρόπος για

να αναζητήσουμε δεδομένα μέσα σε ένα σύνολο προτάσεων είναι να κατανοήσουμε το σημασιολογικό τους περιεχόμενο και να βγάλουμε κάποιο συμπέρασμα από αυτό, ξεχωρίζοντας έτσι τις τα δεδομένα(λέξεις) που μας ενδιαφέρουν. Ένας άλλος τρόπος όμως είναι να αναζητήσουμε δεδομένα με χρήση συγκεκριμένων συντακτικών κανόνων που ακολουθούνται από το σύνολο των δεδομένων στο οποίο αναζητούμε πληροφορίες. Έστω για παράδειγμα το παρακάτω σύνολο των λέξεων που περιγράφει μια λίστα από φοιτητές μιας τάξης

- George Smith - Male - 21/12/1989
- John Smith - Male- 11/01/1990
- Mary Brown- Female - 12/03/1986
- John Doe - Male - 21/13/1986

Παρατηρούμε ότι υπάρχει ένα επαναλαμβανόμενο μοτίβο για τον διαχωρισμό των χαρακτηριστικών των φοιτητών που είναι {Όνομα}{κενό}{Επώνυμο}{κενό}{-}{κενό}{Φύλλο}{κενό}{-}{μμ/ΜΜ/ΕΕΕ}.Αν τώρα θα θέλαμε να αναζητήσουμε μέσα σε αυτό το σύνολο μόνο όσους φοιτητές είναι αγόρια θα έπρεπε να διαβάσουμε όλα τα στοιχεία του συνόλου και έπειτα να διαχωρίσουμε μόνο εκείνα που στο στοιχείο Φύλλο έχουν την ιδιότητα ”Male”. Προφανώς αυτό αποτελεί κάτι ιδιαίτερα χρονοβόρο που σε καμία περίπτωση δεν είναι παραγωγικό και επαναχρησιμοποιήσιμο. Αντί αυτού η διαδικασία θα ήταν πιο εύκολη και περισσότερο ευέλικτη και δυναμική αν υπήρχε κάποιου είδους γλώσσα που να μας επιτρέπει να δημιουργήσουμε εκφράσεις αναζήτησης με βάση κάποιο μοτίβο, όπως το παραπάνω. Αυτό γίνεται με την χρήση ενός συνόλου εργαλείων που αποτελούνται από συγκεκριμένους συμβολισμούς που ονομάζονται κανονικές εκφράσεις ή Requlars Expressions (ή εν συντομία RegEx). Οι περισσότερες γλώσσες προγραμματισμού υποστηρίζουν την δημιουργία regular expressions και ανεξάρτητα από την εντολή που χρησιμοποιείται στην κάθε μια ακολουθούν συνήθως ένα κοινό σύνολο από λέξεις από το ίδιο αλφάβητο, με μικρές ίσως διαφοροποιήσεις.

2.1.3 Regular Expressions σε PHP

Οι κανονικές εκφράσεις στην γλώσσα PHP υλοποιούνται με την ενσωματωμένη βιβλιοθήκη PCRE (Perl Compatible Regular Expression) που είναι μια βιβλιοθήκη της

C και περιέχει ένα σύνολο μεθόδων που εφαρμόζουν αναζήτηση μοτίβων. Ακολουθεί το συντακτικό ύφος και την σημασιολογία της γλώσσας Perl για επεξεργασία RegEx, έχοντας κάποιες ελάχιστες διαφορές από αυτήν.

Οι βασικές εντολές για αναζήτηση με RegEx είναι (Group 2012)

- array preg_grep
- int preg_match
- int preg_match_all
- mixed preg_replace

Διαχωριστικά

Όλα τα μοτίβα πρέπει να εμπεριέχονται μέσα σε διαχωριστικά. Η σύνταξη των διαχωριστικών(delimiters) είναι ίδια με αυτήν στην γλώσσα Perl. Ένα διαχωριστικό μπορεί να είναι ένας οποιοδήποτε χαρακτήρας εκτός από αλφαριθμητικά, backslash(\) και το χαρακτήρα κενό. Αν το διαχωριστικό πρέπει να είναι μέρος της έκφρασης θα πρέπει να γίνει escape με χρήση backslash πχ:

```
1  /<\/\w+>/ Το  # δεύτερο / δενείναιμέροςδιαχωριστικού     ,
      αφούέχειγίνει   escape \/
```

Listing 2.5: Παράδειγμα RegEx σε PHP

Τροποποιήσεις Μοτίβων

Παρακάτω μπορούμε να δούμε τις διαφορετικές εναλλακτικές τροποποιήσεις που υπάρχουν για αναζήτηση μοτίβων.

i *(PCRE_CASELESS*) με πεζά ή κεφαλαία

m *(PCRE_MULTILINE)* η ”αρχή της γραμμής” (ʃ και ”τέλος της γραμμής” ($) ταιριάζουν όσες νέες γραμμές βρίσκονται αμέσως μετά ή αμέσως πριν από το αναζητούμενο κομμάτι

s *(PCRE_DOTALL)* ο μεταχαρακτήρας τελεία ”.” στο μοτίβο ταιριάζει όλους τους χαρακτήρες, συμπεριλαμβανομένων των νέων γραμμών

x *(PCRE_EXTENDED)* ο χαρακτήρας του κενού (whitespace) αγνοούνται πλήρως εκτός όταν γίνονται escape ή βρισκονται μέσα σε μια κλάση χαρακτήρων

[...] καθώς και οι χαρακτήρες που βρίσκονται μεταξύ ενός unescaped # και το επόμενο n θεωρούμενα ως σχόλια

```
1   /\d{2}\w+/ pattern is equal  /\d{2} \w+ #
       εδώείναιμερικάσχόλια    /x
```

Listing 2.6: Παράδειγμα RegEx σε PHP

```
1  //Match dates formatted like MM/DD/YYYY or MM-DD-YY or MM.DD.YY,
       ...
2  $date = "14/01/1979";
3
4  $p    = "!(\\d\\d)[-/.](\\d\\d)[-/.](\\d\\d(?:\\d\\d)?)!";
5
6  if (preg_match($p,$date,$matches) {
7
8  $month = $matches[1];
9
10 $day   = $matches[2];
11
12 $year  = $matches[3];
13
14 }
```

Listing 2.7: Παράδειγμα RegEx σε PHP

2.1.4 Προσκόμιση διαδικτυακών δεδομένων - cURL

Η ενσωματωμένη στην PHP βιβλιοθήκη cURL (libcurl) δίνει την δυνατότητα να μπορέσουμε να λάβουμε το περιεχόμενο που έχει μια απομακρυσμένη ιστοσελίδα του διαδικτύου μέσω του συντακτικού της διεύθυνσης της (url). Προσφέρει αρκετές δυνατότητες όπως την επιλογή του user-agent που θα εμφανίζεται στον server από τον οποίο θα ζητηθούν τα δεδομένα ώστε να είναι σε θέση να αναγνωρίσει και να καταγράψει τις επισκέψεις που έχουν γίνει. Επίσης δίνει την δυνατότητα επιλογής αποθήκευσης σε αρχείο των δεδομένων που προσκομίσθηκαν, για την επεξεργασία τους αργότερα, είτε απλά σε μεταβλητή που αποθηκεύεται μόνο προσωρινά.

Το πιο απλό παράδειγμα της σύνταξης αυτής της εντολής είναι το παρακάτω:

```
curl www.example.com
```

Listing 2.8: Παράδειγμα cURL

Το παραπάνω παράδειγμα θα μπορούσε εναλλακτικά να γίνει με την εντολή της PHP file_get_contents, ωστόσο δεν παρέχει τις δυνατότητες που αναφέραμε παραπάνω και γιαυτό δεν προτιμάται για σύνθετα προβλήματα.

2.1.5 Ενημέρωση μέσω RSS

Είναι πολύ συνηθισμένο φαινόμενο οι ιστοσελίδες που περιέχουν συνεχώς καινούρια και δυναμικά ανανεωμένα δεδομένα, να δίνουν την δυνατότητα στους χρήστες-αναγνώστες τους να ενημερώνονται άμεσα μέσω της παραγωγής αρχείων XML που περιέχουν συγκεκριμένη δομή και σύνταξη και συνήθως ακολουθούν το πρότυπο σύνταξης γνωστό ως Rich Site Summary ή ροές RSS. Μια ροή RSS περιέχει ολόκληρο ή ενδεικτικό μέρος του κειμένου του περιεχομένου που περιγράφει καθώς και την ημερομηνία δημοσίευσης του και τον σύνδεσμο προς αυτό το περιεχόμενο. Μπορεί επίσης να περιέχει και περισσότερα και πιο σύνθετα δεδομένα που ανταποκρίνονται καλύτερα στο ύφος της ιστοσελίδας. Αυτά τα αρχεία συνήθως παράγονται αυτόματα κάθε φορά που δημοσιεύεται νέο περιεχόμενο στην ιστοσελίδα ή σε τακτά χρονικά διαστήματα που ορίζει ο διαχειριστής της. Το πώς θα παραχθούν αφορά το εκάστοτε σύστημα στο οποίο στηρίζεται η κάθε ιστοσελίδα και δεν έχει καμία διαφορά όσον αφορά το τελικό αποτέλεσμα.

Έτσι μπορούμε μέσω αυτών των ροών να είμαστε πάντα ενημερωμένοι σχετικά με το πιο πρόσφατο περιεχόμενο που προστέθηκε σε μια ιστοσελίδα, χωρίς να χρειαστεί να περιηγηθούμε χειροκίνητα σε αυτήν και να μπούμε στη διαδικασία να ψάξουμε για το νέο περιεχόμενο. Αυτό μας βοήθησε ιδιαίτερα στην εργασία μας, καθώς όλες οι ιστοσελίδες δημοσίευσης αγγελιών εργασίας παρέχουν αυτήν την δυνατότητα στους χρήστες τους και έτσι μπορούμε πάντα να ξέρουμε ποιες αγγελίες έχουμε επεξεργαστεί και ποιες είναι οι καινούριες.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
<channel>
        <title>RSS Title</title>
        <description>This is an example of an RSS feed</description>
```

```
6         <link>http://www.someexamplerssdomain.com/main.html</link>
7         <lastBuildDate>Mon, 06 Sep 2010 00:01:00 +0000 </
              lastBuildDate>
8         <pubDate>Mon, 06 Sep 2009 16:45:00 +0000 </pubDate>
9         <ttl>1800</ttl>
10
11        <item>
12                <title>Example entry</title>
13                <description>Here is some text containing an
                      interesting description.</description>
14                <link>http://www.wikipedia.org/</link>
15                <guid>unique string per item</guid>
16                <pubDate>Mon, 06 Sep 2009 16:45:00 +0000 </pubDate>
17        </item>
18
19 </channel>
20 </rss>
```

Listing 2.9: Παράδειγμα ροής RSS

2.2 ORM - Object Relational Mapping

Η δομή των σχεσιακών βάσεων δεδομένων και η προσκόμιση και επεξεργασία των δεδομένων τους διαφέρει αρκετά από την προσέγγιση των αντικειμενοστραφών γλωσσών προγραμματισμού και αποτελούν ουσιαστικά δυο διαφορετικά συστήματα μεταξύ τους. Για αυτόν τον σκοπό έχουν δημιουργηθεί αρκετά εργαλεία λογισμικού τα οποία έχουν σκοπό να συνδέσουν τα αντικείμενα που δημιουργούνται σε μια αντικειμενοστραφή γλώσσα προγραμματισμού (πχ Java ή PHP) με τους πίνακες μιας σχεσιακής βάσης δεδομένων. Η σύνδεση αυτή επιτυγχάνεται με την χρήση επιπρόσθετων κλάσεων που τοποθετούνται σε ένα ενδιάμεσο επίπεδο ανάμεσα στην κυρίως εφαρμογή και την βάση δεδομένων και περιγράφουν την αντιστοιχία μεταξύ των αντικειμένων και της βάσης δεδομένων.

2.2.1 NetBeans - db2php ORM

Για την σύνδεση των δεδομένων της σχεσιακής βάσης δεδομένων MySQL με την αντικειμενοστραφή σχεδίαση του συστήματος μας σε κλάσεις και αντικείμενα της PHP και την ευκολότερη διαχείριση και τροποποίηση τους, χρησιμοποιήθηκε ένα πρόσθετο εργαλείο(plugin) για το περιβάλλον του NetBeans το οποίο δημιουργεί αυτόματα τις κλάσεις για την μοντελοποίηση της βάσης δεδομένων και ονομάζεται db2php. Μέσω αυτού μπορούμε να επιλέξουμε μέσα από γραφικό περιβάλλον αρχικά την βάση δεδομένων και έπειτα τους πίνακες από τους οποίους θα δημιουργηθούν οι κλάσεις σε php. Για τον κάθε έναν πίνακα δημιουργείται και μια ξεχωριστή κλάση η οποία περιέχει τις ίδιες μεθόδους όπως set, get, getBy και άλλες που διευκολύνουν και τυποποιούν την σύνδεση με την βάση δεδομένων.

2.3 Web Services

Με τον όρο *web service* εννοούμε μια αυτόνομη τεχνολογία που επιτρέπει την αμφίδρομη επικοινωνία μέσω ενός δικτύου(όπως το διαδίκτυο) ανεξαρτήτως πλατφόρμας και γλώσσας προγραμματισμού με σκοπό την εξυπηρέτηση των αναγκών είτε απευθείας των χρηστών της είτε άλλων τμημάτων λογισμικού (Papazoglou 2006). Έχουν μεγάλο εύρος εφαρμογής που μπορεί να είναι από απλές αιτήσεις (έλεγχος διαθέσιμου υπολοίπου τραπεζικού λογαριασμού, έλεγχος αποθεμάτων αποθήκης, ενημέρωση για τον καιρό) μέχρι πολύπλοκες εμπορικές εφαρμογές που συνδυάζουν τα δεδομένα τους από διαφορετικές πηγές για να προσφέρουν αξιόπιστα δεδομένα στον χρήστη τους. Με την ραγδαία εξέλιξη του διαδικτύου ολοένα και περισσότερες επιχειρήσεις στρέφονται στη λύση της χρήσης μιας web service αντί για τα συμβατικά λογισμικά κυρίως λόγω της ευελιξίας που προσφέρουν αλλά συνήθως και του χαμηλότερου κόστους τους. Έτσι εισάγεται μια νέα έννοια στον χώρο της πληροφορικής που ονομάζεται *Service Oriented Computing(SOC)*, κατά την οποία ακόμη και ολόκληρα συστήματα είναι χτισμένα στην φιλοσοφία των web service.

Σε γενικές γραμμές μπορούμε να πούμε ότι υπάρχουν 2 ειδών web services:

1. αυτές που είναι σχετικά απλές και περιμένουν πάντα μια αίτηση για να την επεξεργαστούν, να ετοιμάσουν τα δεδομένα και να στείλουν την κατάλληλη απά-

ντηση.

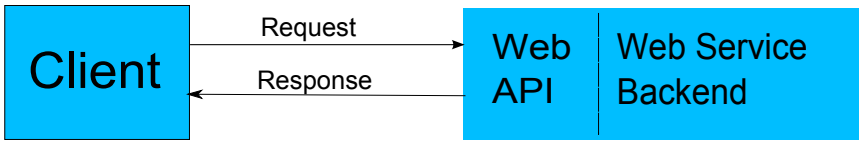
2. αυτές που είναι περισσότερο σύνθετες και εφαρμόζουν ένα είδος συντονισμού ανάμεσα σε άλλες υπηρεσίες.

Έτσι ένα σύνθετο web service συνήθως καταγράφει την κατάσταση(state) του πελάτη ανάμεσα στις κλήσεις των μεθόδων της και ονομάζεται *Stateful*, ενώ μια απλή που δεν το εφαρμόζει ονομάζεται *Stateless*.

2.3.1 Αρχιτεκτονική REpresentational State Transfer (REST)

Ο όρος REpresentational State Transfer (εν συντομία REST) αναφέρεται σε μια αρχιτεκτονική υλοποίησης υπηρεσιών ανάμεσα σε κατανεμημένα συστήματα υπερμέσων (Hypermedia). Το κύριο αντιπροσωπευτικό τέτοιο σύστημα είναι ο παγκόσμιος ιστός (World Wide Web) και είναι ο λόγος που αναπτύχθηκε η αρχιτεκτονική REST, ώστε να περιγράψει ένα σύνολο απο κανόνες και λειτουργίες που πρέπει να διέπουν ένα σύστημα υπερμέσων όπως αυτό. Πήρε την ονομασία του απο τον Roy Fielding, οταν κατα τη διαρκεια εκπόνησης της διδακτορικής του διατριβής το 2000 προσπάθησε να περιγράψει την αρχιτεκτονική δομή του διαδικτύου (Masse 2011)

Ενα σημαντικό κομμάτι της εργασία μας θα στηριχθεί πάνω στην αρχιτεκτονική REST δημιουργώντας ένα RESTful API για την επικοινωνία του εξυπηρετητή (web service) με τον πελάτη ανεξάρτητα απο την πλατφόρμα υλοποίησης του (Web, Desktop, Mobile) . Το σχήμα 2.1 περιγράφει αυτή την διαδικασία



Διάγραμμα 2.1: Η αρχιτεκτονική REST

Η αρχιτεκτονική REST βασίζεται στις εξής 6 βασικές αρχές:

Πελάτη-Εξυπηρετητή

Το βασικό στοιχείο πάνω στο οποίο βασίζονται όλα τα υπόλοιπα είναι ότι το σύστημα ακολουθεί την μορφή πελάτη-εξυπηρετητή, με τρόπο έτσι ώστε η τεχνολογική κατάσταση του κάθε ενός να είναι εντελώς ανεξάρτητη από τον άλλον, διατηρώντας διακριτούς ρόλους στην επικοινωνία τους.

Ομοιομορφία διεπαφής

Οι αλληλεπιδράσεις μεταξύ πελάτη-εξυπηρετητή θα πρέπει να ακολουθούν τα καθιερωμένα πρότυπα,αλλιώς η επικοινωνία μεταξύ τους θα καταρρεύσει.

Λανθάνουσα Μνήμη (Cache)

Με τη δυνατότητα αποθήκευσης σε μνήμη cache σε οποιοδήποτε από τα επίπεδα επικοινωνίας ελαττώνεται σημαντικά ο όγκος των δεδομένων προς μεταφορά, προσδίδοντας έτσι μεγαλύτερη ταχύτητα και αξιοπιστία στην εφαρμογή.

Συστημα πολλαπλών επιπέδων

Η χρήση της ομοιόμορφης διεπαφής επιτρέπει την εισαγωγή ενδιάμεσων επιπέδων ανάμεσα στον εξυπηρετητή και τον πελάτη με σκοπό την καλύτερη εξυπηρέτηση του δικτύου. Ενα ενδιάμεσο επίπεδο πρέπει να μην παίζει ρόλο στην τελική διεπαφή και να είναι αόρατο κατά τη διάρκεια της επικοινωνίας.

Ανεξαρτησία κατάστασης

Βασική προϋπόθεση είναι ότι ο εξυπηρετητής δεν είναι υποχρεωμένος να γνωρίζει συνεχώς τις προηγούμενες καταστάσεις που βρισκόταν ο πελάτης, απαιτώντας έτσι σε κάθε επικοινωνία τους ο πελάτης να περιέχει όλες τις απαραίτητες παραμέτρους που χρειάζονται για να επιτευχθεί η σωστή μεταξύ τους επικοινωνία

Code On Demand

Μερικές φορές είναι απαραίτητο ο πελάτης να χρησιμοποιήσει μια τεχνολογία μόνο εφόσον το ζητάει απαραίτητα ο εξυπηρετητής, γιαυτό το λόγο πρέπει να είναι σε θέση να αναγνωρίσει και να εκτελέσει το περιεχόμενο αυτής της τεχνολογίας που θα μπορούσε να είναι κώδικας JavaScript, Flash video, Java applet κλπ.

Υπάρχουν κάποιοι βασικοί κανόνες πάνω στους οποίους πρέπει να είναι χτισμένο ένα REST API. Ο σημαντικότερος από αυτούς είναι ο αναγνωριστικός σχεδιασμός της επικοινωνίας έχοντας ως σημείο αναφοράς ένα μοναδικό ομοιόμορφο Αναγνωριστικό πόρων - Uniform Resource Identifier (URI). Ενα URI χρησιμοποιείται για να προσδιορίσει την έννοια λειτουργίας και το αποτέλεσμα των δεδομένων που παράγει μια Web Service προς τον πελάτη. Η πιο συνηθισμένη και διαδεδομένη μορφή URI στο

διαδίκτυο είναι ο Ενιαίος Εντοπιστής Πόρων (URL). Η βασική σύνταξη ενός URI έχει οριστεί ως εξής : (IETF 2005)

URI = scheme ”://” [”?” query] [”#” fragment]

Για παράδειγμα εφαρμόζοντας την παραπάνω σύνταξη το κάθε ένα από τα παρακάτω URI θα πρέπει να παράγει σε συγκεκριμένη μορφή τα αντίστοιχα αποτελέσματα ώστε να τα επεξεργαστεί ο πελάτης:

- http://students.api.com/classes?class=icd
- http://students.api.com/classes/icd/semester?s=3
- http://students.api.com/classes/icd/semester/3/students?name=George

Η αναπαράσταση των δεδομένων των πόρων γίνεται συνήθως σε κάποια μορφή κειμένου, με τις πιο διαδεδομένες να είναι οι XML και JSON, χωρίς να είναι απαραίτητο να επιλέγεται μόνο μια αλλά μπορεί να δίνεται η δυνατότητα στον πελάτη να διαλέξει την μορφή που τον εξυπηρετεί περισσότερο. Στην εργασία μας θα χρησιμοποιήσουμε την μορφή JSON . Για να ακολουθεί όμως την αρχιτεκτονική REST μια αναπαράσταση JSON απαιτείται να ακολουθεί τουλάχιστον τις στοιχειώδεις αρχές που περιγράφηκαν προηγουμένως. Μια τυπική μορφή μιας αίτησης από τον πελάτη για την εμφάνιση των στοιχείων ενός φοιτητή και μιας απάντησης αναπαριστάμενης σε μορφή JSON θα μπορούσε να είναι η παρακάτω:

```
{
  # αίτηση
  GET /students/icd?aem=2304 HTTP/1.1
  Host: api.students.teiser.gr


  # απάντηση
  HTTP/1.1 200 OK
  Content-Type: application/json;
    "class" : "icd" ,
    "semester" : "3" ,
     "name" : "George",
     "age" : 20,
     "graduated" : false,
     "courses" : ['Math','Programming','Physics'],
      "familyMembers" : [
          {name: "John", age: 63, relation: "Father"},
          {name: "Nick", age: 25, relation: "Brother"}
    ]
    "links" :
    {
      "self" :
      {
        "href" : "http://api.students.teiser.gr/students/2304"
           ,
        "rel" : "http://api.students.teiser.gr/common/self"
      }
    }
}
```

Listing 2.10: Παράδειγμα JSON

Κεφάλαιο 3

Περιγραφή Συστήματος

3.1 Web Scraper

3.1.1 Ανάλυση απαιτήσεων

Το αντικείμενο αυτής της εργασίας απαιτεί την δημιουργία ενός web scraper ο οποίος θα μπορεί να εντοπίζει το κυρίως κείμενο αγγελιών που είναι δημοσιευμένο σε ιστοσελίδες του διαδικτύου και έπειτα να το επεξεργάζεται κατάλληλα ώστε εξάγει μέσα από αυτό τα δεδομένα που έχουμε εξαρχής ορίσει ως χρήσιμα σε μορφή λέξεων-κλειδιών (keywords).Τα δεδομένα αυτά αποθηκεύονται σε μια βάση δεδομένων και έπειτα είναι διαθέσιμα για να τα προσκομίσει το web service και να τα στείλει στον πελάτη που τα ζήτησε. Για την ορθή υλοποίηση του web scraper, απαιτείται αρχικά ο διαχειριστής του συστήματος να συλλέξει και να καταγράψει τα δεδομένα στα οποία εφαρμόζεται η αναζήτηση, όπως τις διευθύνσεις (url) των ιστοσελίδων που περιέχουν τις αγγελίες, καθώς και τις ροές ενημέρωσης RSS αλλά και το επαναλαμβανόμενο κομμάτι κώδικα HTML μέσα στο οποίο περιέχεται το κυρίως κείμενο των αγγελιών. Σε προκαθορισμένη ώρα της ημέρας εκτελείται για την κάθε μια ιστοσελίδα αγγελιών το σενάριο εντολών(script) του web scraper και καταγράφει τα δεδομένα που εξήγαγε στη βάση δεδομένων, διαχωρίζοντας όσα βρίσκονται σε κοντινές θέσεις μεταξύ τους, σχετικά με το υπόλοιπο κείμενο, με εκείνα που απέχουν αρκετά και πιθανόν να μην αποτελούν μέρος των προσόντων που αναζητούνται.

3.1.2 Περιπτώσεις χρήσης

Σε αυτό το κομμάτι του συστήματος ως actors θεωρούνται ο μηχανισμός του scraper, καθώς και ο διαχειριστής του συστήματος.

Για τις ανάγκες υλοποίησης του Web Scraper προέκυψαν οι παρακάτω περιπτώσεις χρήσης:

Σενάριο χρήσης 1	**Ιδανικό Σενάριο**
Scope:	Scrap Data
Primary Actor:	Σύστημα - Scraper
Ενδιαφερόμενα μέρη και ενδιαφέροντα:	• Διαχειριστής συστήματος: κατάσταση εκτέλεσης web scraper
Προϋποθέσεις:	Έχουμε εισάγει τις ρυθμίσεις για τα website που θα κάνει αναζήτηση ο web scraper και έχουμε εισάγει τα keywords στην βάση δεδομένων

Ιδανικό σενάριο:

1. scraper: Συνδέεται επιτυχώς με τις σελίδες αγγελιών.
2. scraper: Λαμβάνει τις διευθύνσεις (URL) από τις τελευταίες αγγελίες σε μορφή RSS.
3. scraper: Επεξεργάζεται τα δεδομένα και βρίσκει keywords που είναι αποθηκευμένα στη βάση δεδομένων
4. scraper: Καταγράφει τα δεδομένα που εξήγαγε(ημερομηνία, αριθμό εμφανίσεων κλπ) στη βάση δεδομένων

Εναλλακτικά Σενάρια:

1.a scraper: Δεν καταφέρνει να συνδεθεί επιτυχώς με τις σελίδες αγγελιών:

2. scraper: Καταγράφει την ημερομηνία και το response error της ιστοσελίδας ή του συστήματος σε αρχείο καταγραφής σφαλμάτων (log file)
3. scraper: Προσπαθεί σε καθορισμένο διάστημα να επαναδημιουργήσει επιτυχή σύνδεση έως ότου τα καταφέρει
4. Βήματα 2-4 Ιδανικού Σεναρίου

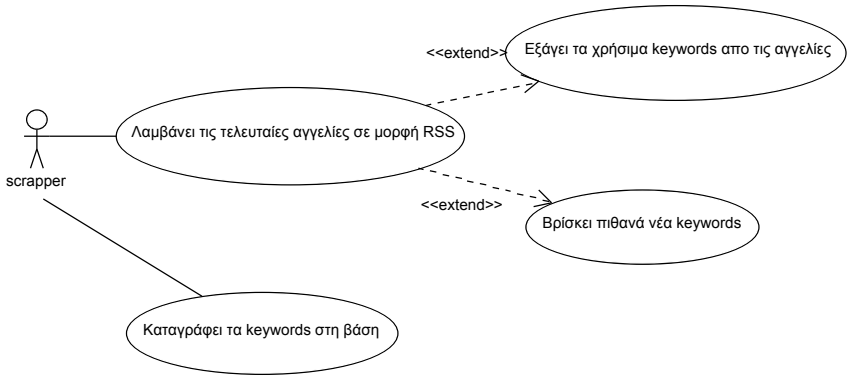
3.a Βρίσκει keywords που δεν είναι αποθηκευμένα στη βάση και μέσω του αλγορίθμου αναγνώρισης τα επισημαίνει ως πιθανά σημαντικά:

1. scraper: Καταγράφει αυτά τα keywords στη βάση σε ειδικό πίνακα "αγνώστων keywords"
2. scraper: Ενημερώνει σε τακτά χρονικά διαστήματα τον διαχειριστή για τα πιθανά νέα άγνωστα keywords
3. διαχειριστής: Αξιολογεί τα keywords και είτε τα εισάγει στα επιλέξιμα από τον scraper είτε τα απορρίπτει και διαγράφονται από τη βάση

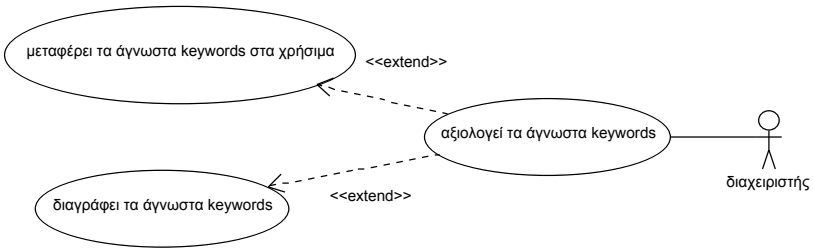
(α΄) Βήμα 4 Ιδανικού Σεναρίου

3.1.3 Διαγράμματα περιπτώσεων χρήσης

Οι περιπτώσεις χρήσης του Web Scraper συνοψίζονται στα διαγράμματα 3.1 και 3.2



Διάγραμμα 3.1: Περιπτώσεις χρήσης Scraper



Διάγραμμα 3.2: Περιπτώσεις χρήσης Scraper

3.2 Web Service

3.2.1 Ανάλυση απαιτήσεων

Αφού έχουν αποθηκευτεί και επεξεργαστεί κατάλληλα τα δεδομένα αναζήτησης, είναι διαθέσιμα προς τους χρήστες μέσω μιας υπηρεσίας ιστού(web service) η οποία είναι φτιαγμένη σύμφωνα με το πρότυπο REST και είναι σε θέση να παρέχει απλά ή και περισσότερο σύνθετα στατιστικά δεδομένα που ζητάει ο χρήστης μέσω συγκεκριμένης σύνταξης. Η υπηρεσία είναι συνεχώς online ώστε να μπορεί να εξυπηρετήσει άμεσα σε οποιαδήποτε στιγμή.Τα δεδομένα που ζητάει ο χρήστης επιστρέφονται σε κατάλληλη μορφοποίηση JSON και σε περίπτωση λανθασμένου ερωτήματος επιστρέφεται κατάλληλο μήνυμα σφάλματος.

3.2.2 Περιπτώσεις χρήσης

Οι βασικές περιπτώσεις χρήσης του web service συνοψίζονται παρακάτω:

Σενάριο χρήσης 1	**Πες μου την κατάσταση του Web Service**
Πεδίο εφαρμογής:	Web Service
Κύριοι παράγοντες:	Web service, Χρήστης
Ενδιαφερόμενα μέρη και τι ζητούν:	• Χρήστης του web service: να μάθει την κατάσταση λειτουργίας της, ώστε να μπορέσει να συνεχίσει να εκτελεί ερωτήματα
Προϋποθέσεις:	-

Ιδανικό σενάριο:

1. client: Ζητάει την κατάσταση του web service
2. web service: Δοκιμάζει τη σύνδεση με τη βάση δεδομένων και επιβεβαιώνει ότι όλο το σύστημα λειτουργεί
3. web service: δίνει την αντίστοιχη απάντηση της σωστής κατάστασης της στον client.
4. client: μπορεί να συνεχίσει να ζητάει στατιστικά δεδομένα

Σενάριο αποτυχίας:

2.α web service: Δοκιμάζει τη σύνδεση με τη βάση δεδομένων και είναι αποτυχημένη ή το υπόλοιπο σύστημα δεν λειτουργεί σωστά

1. web service: Επιστρέφει κατάλληλο μήνυμα σφάλματος
2. client: Δεν συνεχίζει να πραγματοποιεί ερωτήματα

Σενάριο χρήσης 2	**Φέρε μου όλες τις κατηγορίες των προσόντων που αναγνωρίζεις**
Πεδίο εφαρμογής:	Web Service
Κύριοι παράγοντες:	Χρήστης, Web service
Ενδιαφερόμενα μέρη και τι ζητούν:	• Σύστημα(client) του χρήστη του web service: να προσκομίσει όλες τις κατηγορίες, ώστε να τις εμφανίσει κατάλληλα στον χρήστη και αυτός να επιλέξει εκείνη για την οποία επιθυμεί να μάθει περισσότερα στατιστικά στοιχεία
Προϋποθέσεις:	-

Ιδανικό σενάριο:

1. client: Ζητάει τα όλες τις κατηγορίες των προσόντων ακολουθώντας τη σύνταξη που ορίζεται από το web service
2. web service: Επεξεργάζεται το ερώτημα του client και εφόσον είναι έγκυρο συλλέγει τα δεδομένα που ζητήθηκαν
3. web service: Δίνει σε συγκεκριμένη μορφή και μορφοποίηση τα δεδομένα που ζητήθηκαν
4. client: Λαμβάνει επιτυχώς τα δεδομένα και τα εμφανίζει όπως επιθυμεί

Σενάριο αποτυχίας:

1. 1.α client: Τα δεδομένα του ερωτήματος δεν είναι έγκυρα ή δεν ακολουθούν ορθή σύνταξη

    1. web service: Επιστρέφει κατάλληλο μήνυμα σφάλματος

Σενάριο χρήσης 3	**Φέρε μου όλα τα προσόντα που αναγνωρίζεις**
Πεδίο εφαρμογής:	Web Service
Κύριοι παράγοντες:	Χρήστης, Web service
Ενδιαφερόμενα μέρη και τι ζητούν:	• Σύστημα(client) του χρήστη του web service: να προσκομίσει όλα τα προσόντα, ώστε να τα εμφανίσει κατάλληλα στον χρήστη και αυτός να επιλέξει εκείνο για το οποίο επιθυμεί να μάθει περισσότερα στατιστικά στοιχεία
Προϋποθέσεις:	-

Ιδανικό σενάριο:

1. client: Ζητάει τα όλα τα προσόντα ακολουθώντας τη σύνταξη που ορίζεται από το web service
2. web service: Επεξεργάζεται το ερώτημα του client και εφόσον είναι έγκυρο συλλέγει τα δεδομένα που ζητήθηκαν
3. web service: Δίνει σε συγκεκριμένη μορφή και μορφοποίηση τα δεδομένα που ζητήθηκαν
4. client: Λαμβάνει επιτυχώς τα δεδομένα και τα εμφανίζει όπως επιθυμεί

Σενάριο αποτυχίας:

1.α client: Τα δεδομένα του ερωτήματος δεν είναι έγκυρα ή δεν ακολουθούν ορθή σύνταξη

    1. web service: Επιστρέφει κατάλληλο μήνυμα σφάλματος

Σενάριο χρήσης 4	**Φέρε μου για μια κατηγορία τα στατιστικά της**
Πεδίο εφαρμογής:	Web Service
Κύριοι παράγοντες:	Χρήστης, Web service
Ενδιαφερόμενα μέρη και τι ζητούν:	• Χρήστης του web service: να ενημερωθεί για τα στατιστικά εμφάνισης προσόντων σε μια συγκεκριμένη κατηγορία από τις υπάρχουσες, χωρίς περιορισμό στο εύρος ημερομηνιών
Προϋποθέσεις:	-

Ιδανικό σενάριο:

1. client: Ζητάει τα δεδομένα για μια συγκεκριμένη κατηγορία ακολουθώντας τη σύνταξη που ορίζεται από το web service
2. web service: Επεξεργάζεται το ερώτημα του client και εφόσον είναι έγκυρο συλλέγει τα δεδομένα που ζητήθηκαν
3. web service: Δίνει σε συγκεκριμένη μορφή και μορφοποίηση τα δεδομένα που ζητήθηκαν
4. client: Λαμβάνει επιτυχώς τα δεδομένα και τα εμφανίζει όπως επιθυμεί

Σενάριο αποτυχίας:

1.α client: Τα δεδομένα του ερωτήματος δεν είναι έγκυρα ή δεν ακολουθούν ορθή σύνταξη

    1. web service: Επιστρέφει κατάλληλο μήνυμα σφάλματος

Σενάριο χρήσης 5	**Φέρε μου για ένα προσόν τα στατιστικά του**
Πεδίο εφαρμογής:	Web Service
Κύριοι παράγοντες:	Χρήστης, Web service
Ενδιαφερόμενα μέρη και τι ζητούν:	• Χρήστης του web service: να ενημερωθεί για τα στατιστικά εμφάνισης συγκεκριμένου προσόντος από τα υπάρχοντα, χωρίς περιορισμό στο εύρος ημερομηνιών
Προϋποθέσεις:	-

Ιδανικό σενάριο:

1. client: Ζητάει τα δεδομένα για ένα συγκεκριμένο προσόν ακολουθώντας τη σύνταξη που ορίζεται από το web service
2. web service: Επεξεργάζεται το ερώτημα του client και εφόσον είναι έγκυρο συλλέγει τα δεδομένα που ζητήθηκαν
3. web service: Δίνει σε συγκεκριμένη μορφή και μορφοποίηση τα δεδομένα που ζητήθηκαν
4. client: Λαμβάνει επιτυχώς τα δεδομένα και τα εμφανίζει όπως επιθυμεί

Σενάριο αποτυχίας:

1.α client: Τα δεδομένα του ερωτήματος δεν είναι έγκυρα ή δεν ακολουθούν ορθή σύνταξη

    1. web service: Επιστρέφει κατάλληλο μήνυμα σφάλματος

Σενάριο χρήσης 6	**Φέρε μου για μια κατηγορία τα στατιστικά της για συγκεκριμένη περίοδο**
Πεδίο εφαρμογής:	Web Service
Κύριοι παράγοντες:	Χρήστης, Web service
Ενδιαφερόμενα μέρη και τι ζητούν:	• Χρήστης του web service: να ενημερωθεί για τα στατιστικά εμφάνισης προσόντων σε μια συγκεκριμένη κατηγορία από τις υπάρχουσες με συγκεκριμένο περιορισμό στο εύρος ημερομηνιών
Προϋποθέσεις:	-

Ιδανικό σενάριο:

1. client: Ζητάει τα δεδομένα για μια συγκεκριμένη κατηγορία και για συγκεκριμένο εύρος ημερομηνιών ακολουθώντας τη σύνταξη που ορίζεται από το web service
2. web service: Επεξεργάζεται το ερώτημα του client και εφόσον είναι έγκυρο ως προς τη σύνταξη και οι ημερομηνίες είναι σωστά καθορισμένες και προγενέστερες συλλέγει τα δεδομένα που ζητήθηκαν
3. web service: Δίνει σε συγκεκριμένη μορφή και μορφοποίηση τα δεδομένα που ζητήθηκαν
4. client: Λαμβάνει επιτυχώς τα δεδομένα και τα εμφανίζει όπως επιθυμεί

Σενάριο αποτυχίας 1:

1.α client: Τα δεδομένα του ερωτήματος δεν είναι έγκυρα ή δεν ακολουθούν ορθή σύνταξη

1. web service: Επιστρέφει κατάλληλο μήνυμα σφάλματος

Σενάριο αποτυχίας 2:

1.α client: Τα δεδομένα του ερωτήματος είναι έγκυρα και ακολουθούν ορθή σύνταξη αλλά οι ημερομηνίες είναι μεταγενέστερες της τωρινής

1. web service: Επιστρέφει κατάλληλο μήνυμα σφάλματος

Σενάριο χρήσης 7	**Φέρε μου για ένα προσόν τα στατιστικά του για συγκεκριμένη περίοδο**
Πεδίο εφαρμογής:	Web Service
Κύριοι παράγοντες:	Χρήστης, Web service
Ενδιαφερόμενα μέρη και τι ζητούν:	• Χρήστης του web service: να ενημερωθεί για τα στατιστικά εμφάνισης συγκεκριμένου προσόντος από τα υπάρχοντα με συγκεκριμένο περιορισμό στο εύρος ημερομηνιών
Προϋποθέσεις:	-

Ιδανικό σενάριο:

1. client: Ζητάει τα δεδομένα για ένα συγκεκριμένο προσόν και για συγκεκριμένο εύρος ημερομηνιών ακολουθώντας τη σύνταξη που ορίζεται από το web service
2. web service: Επεξεργάζεται το ερώτημα του client και εφόσον είναι έγκυρο ως προς τη σύνταξη και οι ημερομηνίες είναι σωστά καθορισμένες και προγενέστερες συλλέγει τα δεδομένα που ζητήθηκαν
3. web service: Δίνει σε συγκεκριμένη μορφή και μορφοποίηση τα δεδομένα που ζητήθηκαν
4. client: Λαμβάνει επιτυχώς τα δεδομένα και τα εμφανίζει όπως επιθυμεί

Σενάριο αποτυχίας 1:

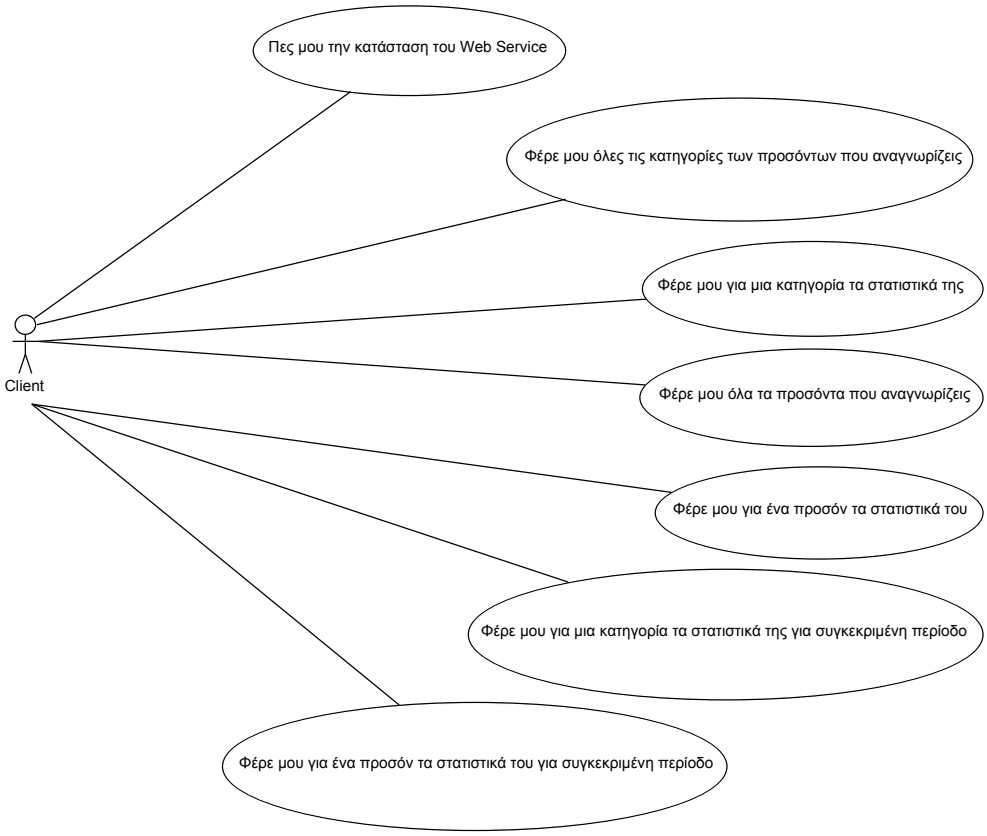
1.α client: Τα δεδομένα του ερωτήματος δεν είναι έγκυρα ή δεν ακολουθούν ορθή σύνταξη

1. web service: Επιστρέφει κατάλληλο μήνυμα σφάλματος

Σενάριο αποτυχίας 2:

1.α client: Τα δεδομένα του ερωτήματος είναι έγκυρα και ακολουθούν ορθή σύνταξη αλλά οι ημερομηνίες είναι μεταγενέστερες της τωρινής

1. web service: Επιστρέφει κατάλληλο μήνυμα σφάλματος



Διάγραμμα 3.3: Περιπτώσεις χρήσης Web Service

3.3 Περιβάλλον διαχείρισης εφαρμογής

Πολλές φορές είναι πιο εύχρηστο και περισσότερο λειτουργικό αν μια εφαρμογή έχει ένα γραφικό περιβάλλον διαχείρισης των δυνατοτήτων της και επίβλεψης της κατάστασης της.Για τον λόγο αυτό υλοποιήθηκε ένα περιβάλλον στο οποίο ο διαχειριστής του συστήματος είναι σε θέση να ελέγχει τον web scraper σε διάφορα θέματα όπως:

- κάθε πότε θα κάνει αναζήτηση δεδομένων
- αν θα τα αποθηκεύσει στη βάση δεδομένων ή αν θα είναι δοκιμαστικά
- να προσθέτει και να τροποποιεί site από τα οποία γίνεται η αναζήτηση
- δυνατότητα προσθήκης και επεξεργασίας keywords προς αναζήτηση καθώς και κατηγοριοποίηση τους
- αν κάνει σωστά την αναζήτηση του, συγκρίνοντας το κείμενο και τα δεδομένα που σύλλεξε

καθώς και διάφορα ακόμη ζητήματα που κάνουν το χειρισμό του πιο απλό και τη λειτουργία του περισσότερο αποδοτική.

Επίσης μέσα από το περιβάλλον διαχείρισης δίνεται η δυνατότητα ελέγχου του web service, ώστε ο διαχειριστής να μπορεί να ελέγχει και να επιβλέπει αν λειτουργεί η σύνδεση με τη βάση δεδομένων και γενικότερα η υπηρεσία.

3.3.1 Περιπτώσεις χρήσης

Σε αυτό το κομμάτι ως actor θεωρείται ο διαχειριστής του συστήματος.

Για τις ανάγκες υλοποίησης της σελίδας διαχείρισης προέκυψαν οι παρακάτω περιπτώσεις χρήσης:

Σενάριο χρήσης 1	**Εισαγωγή Ιστοτόπου**
Scope:	Control Panel
Primary Actor:	Διαχειριστής
Ενδιαφερόμενα μέρη και ενδιαφέροντα:	• Διαχειριστής συστήματος: εισαγωγή ενός νέου ιστοτόπου προς αναζήτηση
Προϋποθέσεις:	-

Ιδανικό σενάριο:

1. διαχειριστής: Επιλέγει από το μενού το στοιχείο ”Manage Websites”
2. διαχειριστής: Επιλέγει το κουμπί ”Insert Website”
3. σύστημα: Εμφανίζει την φόρμα με τα απαραίτητα πεδία προς εισαγωγή
4. διαχειριστής: Εισάγει τα δεδομένα
5. σύστημα: Πραγματοποιεί έλεγχο εγκυρότητας των δεδομένων
6. σύστημα: Αποθηκεύει τα δεδομένα στη βάση δεδομένων

Εναλλακτικό σενάριο επεξεργασίας:

2.α διαχειριστής: Επιλέγει κάποιον από τους υπάρχοντες ιστοτόπους και πατάει στο κουμπί ”Edit”

1. σύστημα: Εμφανίζει τα αποθηκευμένα στοιχεία του ιστοτόπου που επιλέχθηκε
2. διαχειριστής: Επεξεργάζεται τα δεδομένα
3. : Συνέχεια από το βήμα 5 του ιδανικού σεναρίου

Σενάριο χρήσης 2	**Εισαγωγή Κατηγορίας**
Scope:	Control Panel
Primary Actor:	Διαχειριστής
Ενδιαφερόμενα μέρη και ενδιαφέροντα:	• Διαχειριστής συστήματος: εισαγωγή μιας νέας κατηγορίας προσόντων ή ενός νέου προσόντος
Προϋποθέσεις:	-

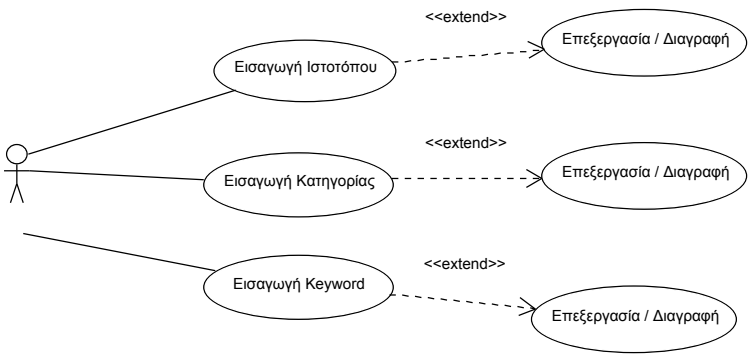
Ιδανικό σενάριο:

1. διαχειριστής: Επιλέγει από το μενού το στοιχείο ”Manage Keywords/Categories”
2. διαχειριστής: Επιλέγει το κουμπί ”Click to insert”
3. σύστημα: Εμφανίζει την φόρμα με τα απαραίτητα πεδία προς εισαγωγή
4. διαχειριστής: Εισάγει τα δεδομένα
5. σύστημα: Πραγματοποιεί έλεγχο εγκυρότητας των δεδομένων
6. σύστημα: Αποθηκεύει τα δεδομένα στη βάση δεδομένων

Εναλλακτικό σενάριο επεξεργασίας:

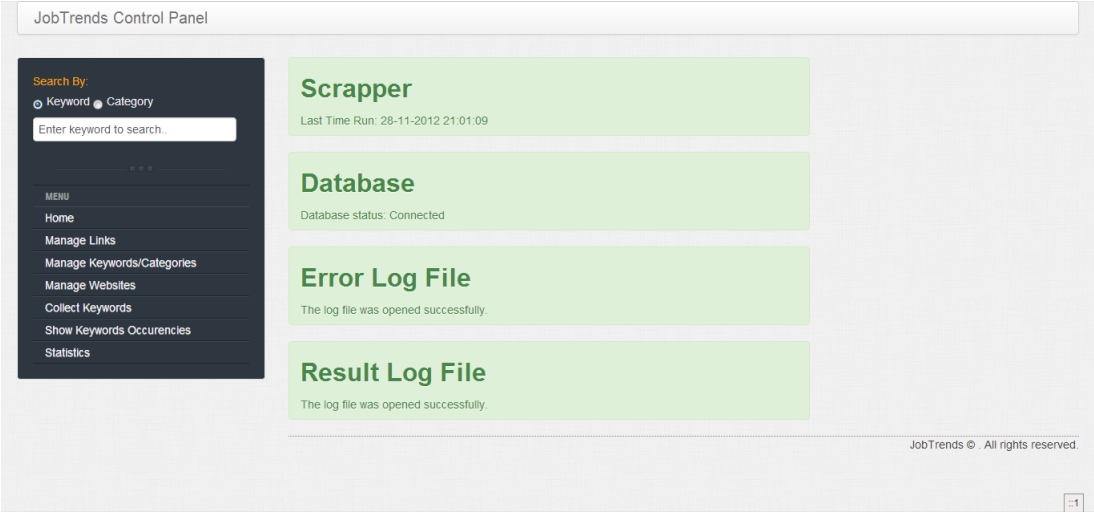
2.α διαχειριστής: Επιλέγει κάποια από τις υπάρχουσες κατηγορίες ή προσόντα και πατάει στο κουμπί ”Edit”

1. σύστημα: Εμφανίζει τα αποθηκευμένα στοιχεία που επιλέχθηκαν
2. διαχειριστής: Επεξεργάζεται τα δεδομένα
3. : Συνέχεια από το βήμα 5 του ιδανικού σεναρίου

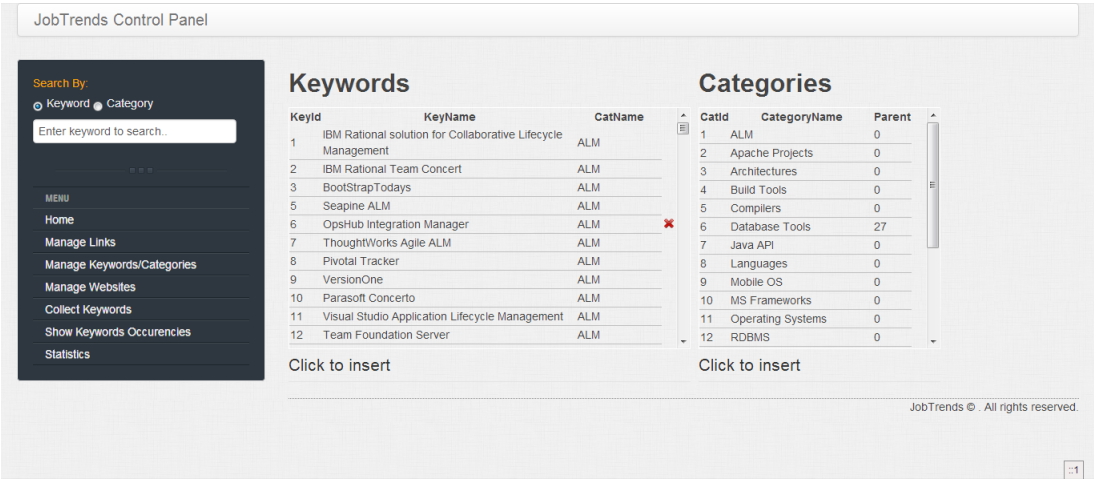


Διάγραμμα 3.4: Περιπτώσεις χρήσης Control Panel

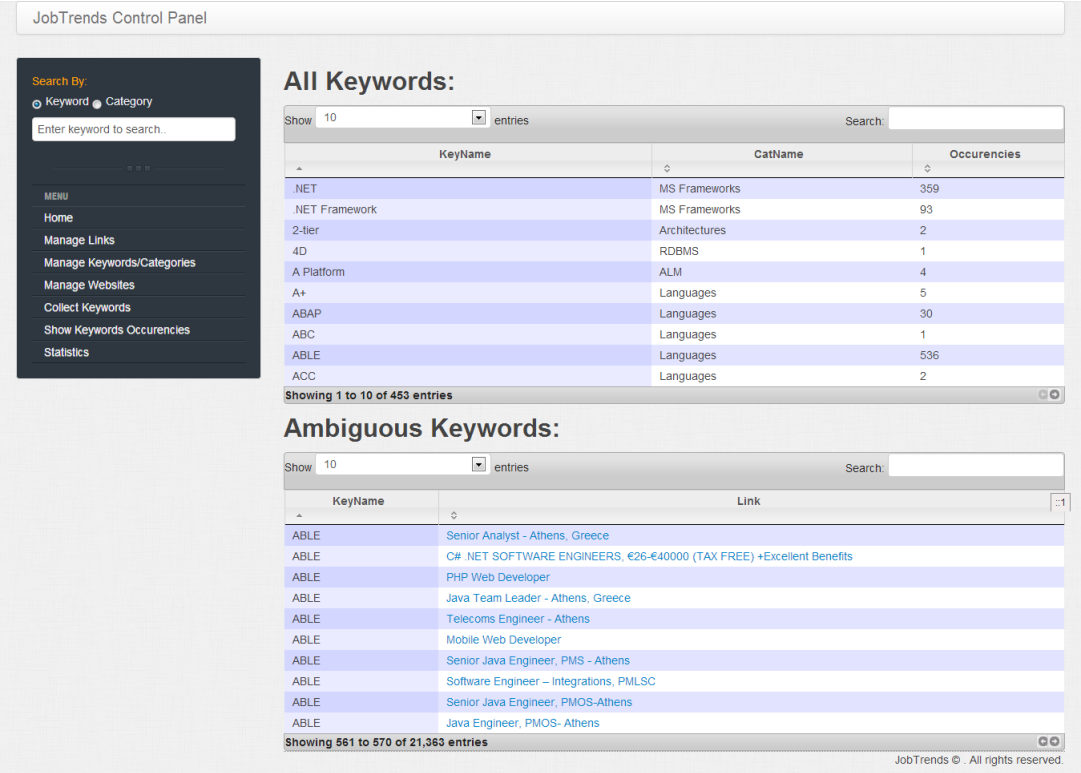
3.3.2 Παραδείγματα με Screenshot



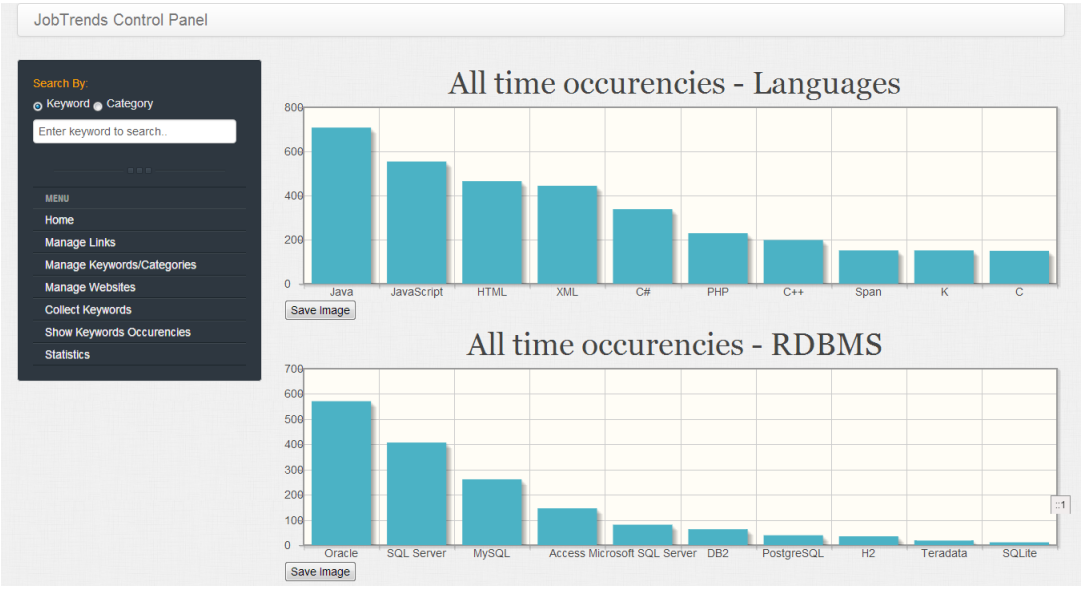
Διάγραμμα 3.5: Αρχική σελίδα διαχείρισης



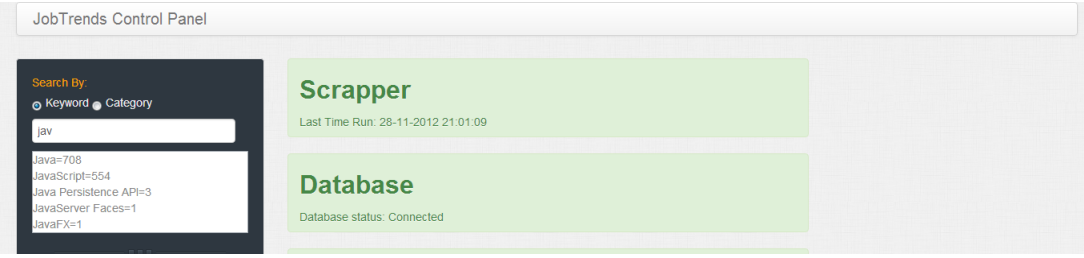
Διάγραμμα 3.6: Διαχείρισης λέξεων κλειδιών



Διάγραμμα 3.7: Προβολή εμφανίσεων



Διάγραμμα 3.8: Σελίδα προβολής στατιστικών



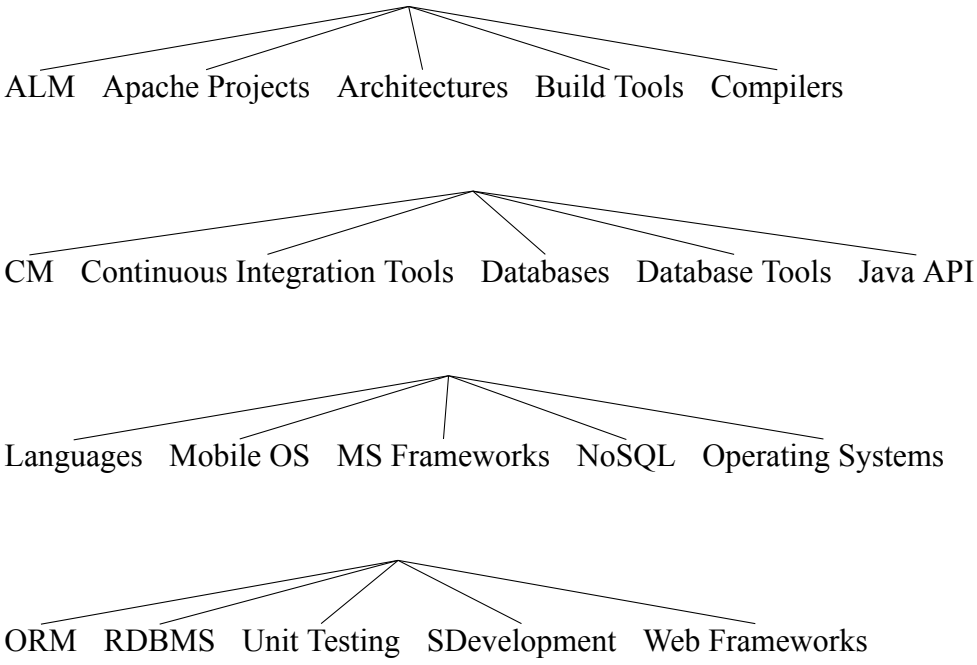
Διάγραμμα 3.9: Γρήγορη αναζήτηση

Κεφάλαιο 4

Ανάλυση και σχεδίαση συστήματος

4.1 Κατηγοριοποίηση προσόντων προς αναζήτηση

Για την αποτελεσματικότερη υλοποίηση της αναζήτησης στους ιστοτόπους αγγελιών, συλλέξαμε όσο το δυνατόν περισσότερα προσόντα σε μορφή λέξεων κλειδιών και έγινε κατηγοριοποίηση τους σε κατηγορίες και υποκατηγορίες. Παρακάτω μπορούμε να δούμε τις κατηγορίες που έχουμε χωρίσει τα προσόντα.

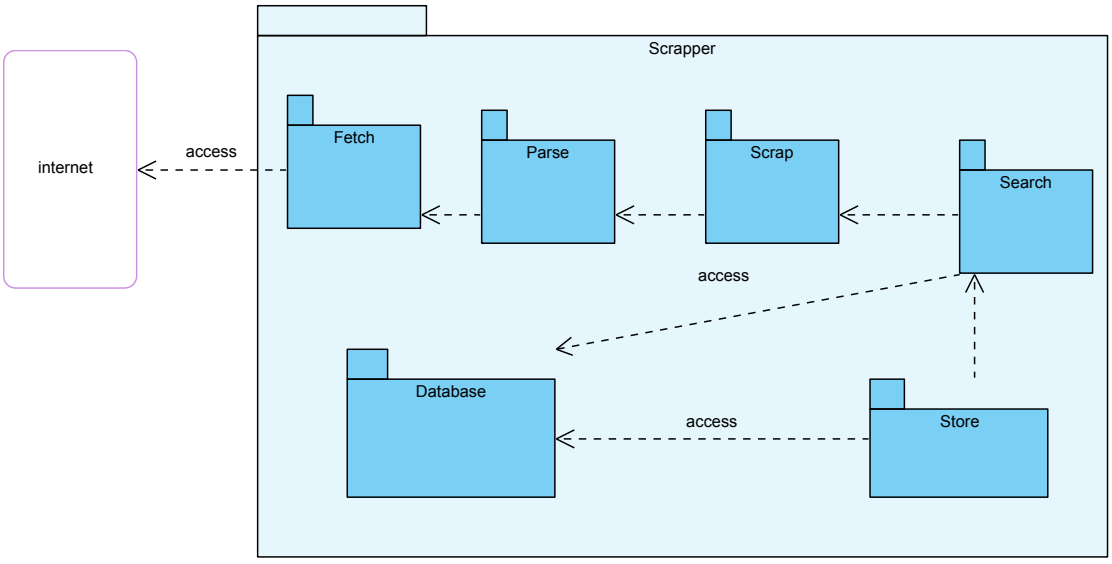


4.2 Web Scraper

4.2.1 Βασικά μέρη

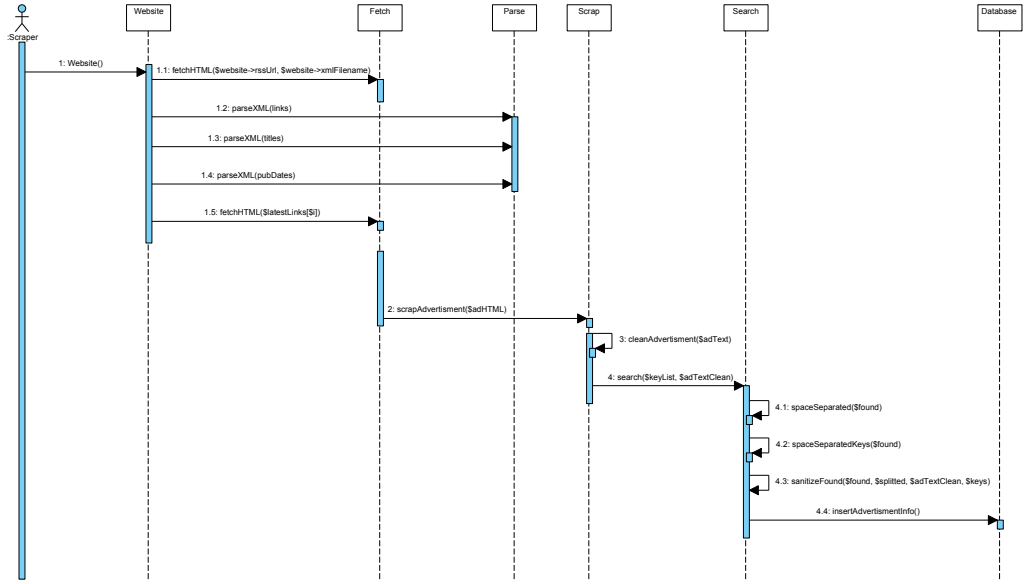
Το κομμάτι της υλοποίησης του web scraper αποτελείται από τα εξής βασικά μέρη τα οποία περιγράφονται και από το διάγραμμα ακολουθίας 4.2 :

- Fetch: Προσκόμιση δεδομένων από το διαδίκτυο για επεξεργασία, όπως ιστοσελίδες σε μορφή HTML, ροές RSS σε μορφή XML.
- Parse: Ανάλυση των στοιχείων του κάθε δεδομένου από μια ροή RSS.
- Scrap: Εντοπισμός και εξαγωγή του μέρους της πληροφορίας που μας ενδιαφέρει μέσα από ένα σύνολο δεδομένων HTML. Αφαίρεση των πιο κοινών σημείων στίξης και μετατροπή του σε καθαρό κείμενο λέξεων με μοναδικό διαχωριστικό το κενό.
- Search: Αναζήτηση μέσα στο εξαγμένο κομμάτι πληροφορίας για προκαθορισμένα αλλά και νέα άγνωστα keywords.
- Store: Αποθήκευση στην βάση δεδομένων των στοιχείων που εντοπίστηκαν, καθώς και διάφορα ακόμη στοιχεία για στατιστικούς κυρίως λόγους όπως ημερομηνία που δημοσιεύθηκε η αγγελία, τον τίτλο και τη διεύθυνση (url) της.



Διάγραμμα 4.1: Διάγραμμα πακέτων Scraper

Το παρακάτω διάγραμμα υλοποιεί το ”Σενάριο χρήσης 1” του Web Scraper



Διάγραμμα 4.2: Διάγραμμα ακολουθίας Scraper

4.2.2 Βάση Δεδομένων

Αφού γίνει η αναζήτηση σε κάθε μια από τις αγγελίες, αποθηκεύονται στη βάση δεδομένων ο αριθμός εμφάνισης των keywords και η ημερομηνία που βρέθηκαν συσχετίζοντας τα με την αγγελία στην οποία βρέθηκαν, για την οποία αποθηκεύεται το url της και ο τίτλος της. Επίσης ανάλογα με το αποτέλεσμα που επιστρέφει η εκτέλεση του αλγορίθμου αναγνώρισης προτύπων Maximin, τοποθετείται ένα flag, το ambiguous, που μας αναφέρει αν το keyword έχει σχέση με την αγγελία ή αν είναι πιθανότατα κάτι που δεν ανήκει στα προσόντα που αναζητούνται. Επομένως η δομή αυτή μας δίνει την δυνατότητα να μπορούμε να εξάγουμε απλά ή και σύνθετα στατιστικά στοιχεία, χωρίς να είναι απαραίτητη η αποθήκευση πολλών δεδομένων. Τα δεδομένα που αναζητάει ο web scraper καθώς και αυτά που βρίσκει, βρίσκονται και αποθηκεύονται σε μια σχεσιακή βάση δεδομένων MySQL. Από μια ιστοσελίδα αγγελιών καθώς και ακολούθως από τη ροή RSS της χρειαζόμαστε να αποθηκεύσουμε τα εξής στοιχεία:

url η διεύθυνση της ιστοσελίδας

xpath η έκφραση xpath για το στοιχείο html που περιέχει τη χρήσιμη πληροφορία της αγγελίας

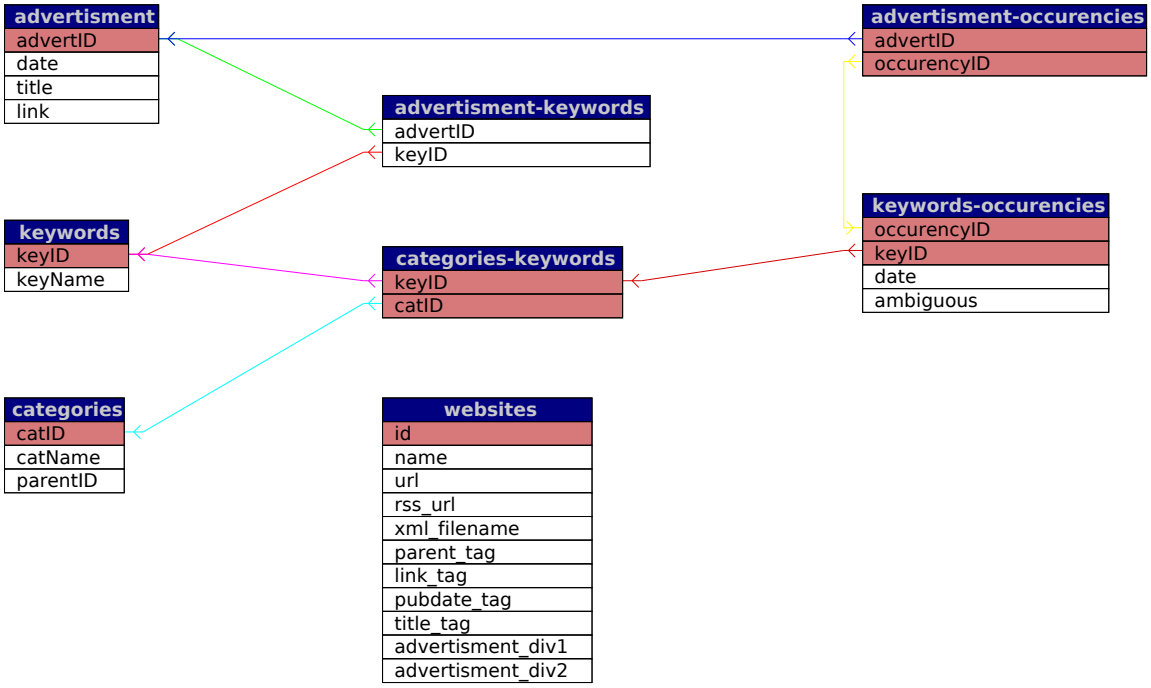
url η διεύθυνση στην οποία δημοσιεύεται η ροή

parent_tag το γονικό στοιχείο της δομής XML

link_tag το στοιχείο που περιέχει τον σύνδεσμο(url) της αγγελίας

pubdate_tag το στοιχείο που περιέχει την ακριβή ημέρα και ώρα δημοσίευσης της αγγελίας

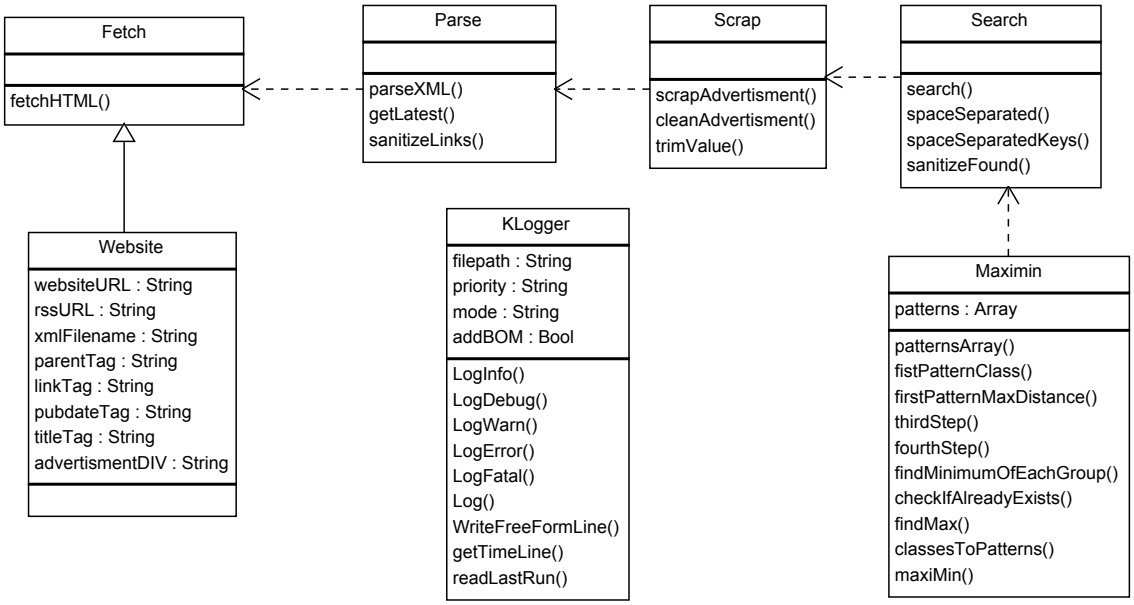
title_tag(προαιρετικά) το στοιχείο που περιέχει τον τίτλο της αγγελίας



Διάγραμμα 4.3: Διάγραμμα συσχετίσεων βάσης δεδομένων

4.2.3 Υλοποίηση

Για την υλοποίηση των βασικών μερών του Web Scraper όπως αναφέρονται στην προηγούμενη ενότητα, δημιουργήθηκαν οι κλάσεις που περιγράφονται παρακάτω και αναπαρίστανται από το διάγραμμα κλάσεων της εικόνας 3.2



Διάγραμμα 4.4: Διάγραμμα κλάσεων Scraper

4.2.4 Προσκόμιση δεδομένων

Η προσκόμιση των δεδομένων από το διαδίκτυο γίνεται μέσω της κλάσης ”Fetch” η οποία είναι υπεύθυνη για την δημιουργία σύνδεσης με μια ζητούμενη ιστοσελίδα, τον έλεγχο για τη διαθεσιμότητα της και την λήψη του περιεχομένου της. Η διαδικασία γίνεται με την βοήθεια της βιβλιοθήκης curl η οποία είναι ενσωματωμένη στην PHP και έχει την δυνατότητα να μεταφέρει δεδομένα μέσω διαφόρων πρωτοκόλλων συμπεριλαμβανομένου και του HTTP.

4.2.5 RSS Parse

Για να διαβάσουμε όλες τις τελευταίες αγγελίες που έχουν δημοσιευθεί στην κατηγορία που μας ενδιαφέρει, που συνήθως αναφέρεται ως IT and Communications ή Computer Science, χρησιμοποιούμε τις ροές RSS που προσφέρουν σχεδόν όλες οι

ιστοσελίδες. Η επεξεργασία τους γίνεται μέσω της κλάσης Parse η οποία περιέχει μεθόδους για το επαναληπτικό διάβασμα ανάμεσα στα αντικείμενα του εγγράφου XML και τον διαχωρισμό και την κατηγοριοποίηση τους ώστε να χρησιμοποιηθούν τα δεδομένα τους, όπως το url της αγγελίας, από τα υπόλοιπα μέρη του συστήματος. Επίσης ελέγχει εάν η ημέρα και ώρα δημοσίευσης της αγγελίας είναι μεταγενέστερη από την τελευταία φορά που εκτελέστηκε το script του scraper ώστε να λαμβάνουμε προς επεξεργασία μόνο τις πρόσφατες αγγελίες.

Παρακάτω βλέπουμε μια από τις βασικές μεθόδους η οποία διαβάζει ένα αρχείο XML και διαχωρίζει τα μέρη του, ανάλογα με τις παραμέτρους που δίνουμε.σ

```
1   /**
2     * parse RSS XML feed and extract advertisment data
3     *
4     * @param string $xmlFilename the file to parse
5     * @param string $parentTag the parent tag of the XML structure
6     * @param string $childTag the child of parent tag
7     * @return array $data
8     *
9     **/
10    function parseXML($xmlFilename, $parentTag, $childTag)
11    {
12        $xmlParse = new DOMDocument('1.0', 'utf-8');
13        $xmlParse->load($xmlFilename);
14        $data = array();
15        foreach( $xmlParse->getElementsByTagName($parentTag) as
              $node)
16        {
17            $value = $node->getElementsByTagName($childTag)->item(0)
                  ->nodeValue;
18            array_push($data, $value);
19        }
20
21        return $data;
22    }
```

Listing 4.1: Class Parse - parseXML function

4.2.6 Data scraping

Αφού έχει ληφθεί το περιεχόμενο της ιστοσελίδας, πρέπει να διαχωριστεί το σημαντικό κομμάτι της ώστε να εφαρμοστεί αργότερα μόνο μέσα σε αυτό η αναζήτηση και όχι σε όλη τη σελίδα.Η απομόνωση γίνεται με χρήση της xPath.

```
1   /**
2    * parse HTML of advertisment and scrap the selected part
3    *
4    * @param string $html has the fetched html from web
5    * @param string $xpathElement is the desired part from HTML
         that contains data we want
6    * @return array $adText
7    *
8    **/
9   function scrapAdvertisment($html, $xpathElement = null)
10  {
11      $doc = new DOMDocument('1.0', 'UTF-8');
12      $doc->loadHTML('<?xml encoding="UTF-8">' . $html);
13      $xpath = new DOMXPath($doc);
14      if( !is_null($xpathElement) && $xpathElement != '' )
15      {
16          $element = $xpath->query($xpathElement)->item(0);
17      }
18      else
19      {
20          $element = $doc->getElementsByTagName('body')->item(0);
21      }
22
23      $this->adText = $element->nodeValue;
24      return $this->adText;
25  }
```

Listing 4.2: Class Scrap - scrapAdvertisment function

Επειδή το κείμενο μιας πρότασης περιέχει και σημεία στίξης και διάφορα άλλα σύμβολα για παράδειγμα ẅe ask candidates with knowledge of: 'javascript/php/Microsoft-Windows',¨ εφαρμόζεται στο απομονωμένο κείμενο ένας καθαρισμός από όλα τα σημεία στίξης και τα περιττά σύμβολα έτσι ώστε το τελικό κείμενο να περιέχει μόνο

λέξεις με κενά.Έτσι διασφαλίζεται το γεγονός ότι δεν θα χάσουμε κάποιο keyword που υπάρχει μέσα στο κείμενο άλλα περικλείεται σε κάποια σύμβολα.

```
1   /**
2     * Get some text and and clean it by removing unesseccary
          punctuation characters
3     *
4     * @param $adText is the text to be cleaned
5     * @return array $words
6     */
7   function cleanAdvertisment($adText)
8   {
9             $this->adTextClean = null;
10
11            /*split the phrase by any number of ',' or '/' or ')' or
                  ':' or '-' or space characters,which include " ", \r
                 , \t, \n and \f
12             */
13            $words = preg_split(ILLEGAL_CHARS,$adText,-1,
                 PREG_SPLIT_NO_EMPTY);
14
15            //apply trim_value function to each member of $words
                 array
16            array_walk($words,array($this,'trimValue'));
17
18            foreach ($words as $word)
19            {
20                $this->adTextClean .= " " . $word;
21            }
22
23            return $this->adTextClean;
24  }
```

Listing 4.3: Class Scrap - cleanAdvertisment function

4.2.7 Search

Αφού έχουν ληφθεί τα δεδομένα και έχουν καθαριστεί από περιττά σημεία στίξης και σύμβολα, γίνεται αναζήτηση μέσα στο κείμενο για εμφανίσεις των keywords που

έχουμε στη βάση δεδομένων με χρήση των συναρτήσεων της PHP preg_match() και preg_grep() οι οποίες είναι ιδιαίτερα γρήγορες και αποτελεσματικές. Η υλοποίηση των μεθόδων γίνεται στην κλάση Search και γίνεται σε πολύ μικρό χρόνο. Ενδεικτικά σε ένα καθαρό κείμενο αγγελίας 1000 λέξεων ψάχνοντας για 2000 keywords που βρίσκονται στην βάση δεδομένων βρίσκει το αποτέλεσμα των εμφανίσεων σε λιγότερο από 2 milliseconds.

Μια απο τις βασικές μεθόδους περιγράφεται παρακάτω και

```
1  /*
2     * Search for each keyword in advertisment text using string
          comparison
3     *
4     * @param array $keyList has all keywords
5     * @param array $found has all the keywords found in $adText
6     * @return array $offsetFound
7     */
8    function search($keyList,$adText)
9    {
10       $found = array();
11       $foundKeys = array();
12       $keyOffset = array();
13
14       foreach($keyList as $keyword)
15       {
16           $keyword=trim($keyword);
17           $charlist="/+-*#!.";
18           $keyword = addcslashes($keyword,$charlist);
19           preg_match('/ ' . $keyword . ' /i', $adText, $search,
                PREG_OFFSET_CAPTURE);
20           //$search=preg_match('/ ' . $keyword . ' /i', $adText);
21            if($search==true){
22                $string = stripcslashes($keyword);
23                //array_push($found, $string);
24                array_push($foundKeys, $string);
25                array_push($keyOffset, $search[0][1]);
26            }
27       }
28       array_push($found, $foundKeys);
```

```
29          array_push($found, $keyOffset);
30          //var_dump($foundKeys);
31          //var_dump($keyOffset);
32          //$offsetFound = $this->filterOffset($foundKeys, $keyOffset)
                ;
33          return $found;
34      }
```

Listing 4.4: Class Search - search function

4.2.8 Αναγνώριση προτύπων - Αλγόριθμος MaxiMin

Ένα από τα προβλήματα που προέκυψαν κατά την υλοποίηση της εργασίας ήταν το γεγονός ότι αρκετά από τα χιλιάδες keywords που είναι καταχωρημένα στη βάση και με τα οποία γίνεται η αναζήτηση, αποτελούν και μέρος της φυσικής γλώσσας όπως για παράδειγμα η γλώσσα προγραμματισμού *"Click"* ή οι *"C"*, *"D"*, *"F"*, *"Code"* και αρκετές ακόμη που αν και δεν είναι ιδιαίτερα δημοφιλείς ή γνωστές δεν παύουν να αποτελούν μέρος της αναζήτησης. Έτσι ο αλγόριθμος αναζήτησης όταν εντόπιζε μια από αυτές τις λέξεις τις καταχωρούσε στη βάση δεδομένων με τα αποτελέσματα αν και τις περισσότερες φορές αυτά τα keywords ήταν μέρος μιας έκφρασης φυσικής γλώσσας και όχι ένα από τα προσόντα που ζητούσαν οι εργοδότες στις αγγελίες τους. Παρατηρήθηκε ότι όταν αυτές οι λέξεις δεν ήταν μέρος των απαιτήσεων της αγγελίας είχαν πολύ μεγάλη απόσταση από το σημείο εκείνο όπου συνήθως οι απαιτήσεις είναι συγκεντρωμένες όπως για παράδειγμα το κείμενο της παρακάτω αγγελίας, στην οποία έχουμε αφαιρέσει κάποια σημεία για να μην είναι ιδιαίτερα μακροσκελής:

The ideal candidate has strong background in development of IT applications S/he will join the company's Development team responsible for the analysis development maintenance and integration of applications [...] supplier of IT services to European Union Institutions international organizations European Agencies and national government Administrations all over Europe We currently have a vacancy for a Senior JAVA J2EE Analyst Developer with a good command of the English language to join our teams in Brussels as well as Helsinki The work will be carried out either in the company's premises or on site at customer premises In the context of the first assignment the successful candidate will be integrated in the Development team of the company that

will closely cooperate with a major client's IT team on site Your tasks Analysis of […] project artefacts Create documentation technical specifications and short reports Support the software during transitions between different environments Your skills Minimum 8 years of combined relevant University Studies and Experience in IT Minimum 5 years experience in software development with JAVA/J2EE Minimum 2 years of analysis and programming experience Extended experience with Oracle WebLogic SQL SOA RUP Scrum and Agile methodologies Experience with UML or case tools Excellent command of English both written and spoken with the ability to participate in English-only meetings Knowledge of French will be considered as an asset Our offer If you are seeking a career in an exciting and dynamic company as part of a team of a major international Institution operating in an international multilingual and multicultural environment where you can expect real chances to make a difference please send us your detailed CV in English to the following e-mail address […] To send your CV please click here

Εδώ βλέπουμε ότι αρχικά ο αλγόριθμος αναζήτησης θα εντοπίσει ορθά ως keywords τα SOA, J2EE, Oracle, Java αλλά εσφαλμένα και τα make , Click τα οποία δεν είναι μέρος των προσόντων που απαιτούνται αλλά απλές λέξεις της φυσικής γλώσσας. Διακρίνουμε ωστόσο ότι τα SOA, J2EE, Oracle, Java είναι σχετικά κοντά μεταξύ τους στο κείμενο σε αντίθεση με τα Jakarta , Click τα οποία δεν έχουν καμία σχέση μεταξύ τους.

Για την αντιμετώπιση αυτού το προβλήματος εφαρμόστηκε ο αλγόριθμος αναγνώρισης προτύπων χωρίς επόπτη MaxiMin ο οποίος διαχωρίζει σε κλάσεις ή αλλιώς ομάδες (clusters) τα πρότυπα που δέχεται ως είσοδο. Πρόκειται για μία μέθοδο προσδιορισμού του πλήθους και του περιεχομένου των συγκεντρώσεων των προτύπων που βασίζεται στη χρήση αποστάσεων μεταξύ των προτύπων (Στρουθόπουλος 2003) Για την υλοποίηση του χρειάστηκε να κρατείται η θέση του κάθε ευρισκόμενου keyword μέσα στο κείμενο, ώστε τελικά όλες οι θέσεις όλων των keyword που βρέθηκαν να αποτελέσουν τα πρότυπα με βάση τα οποία θα γίνει ο διαχωρισμός σε κλάσεις.

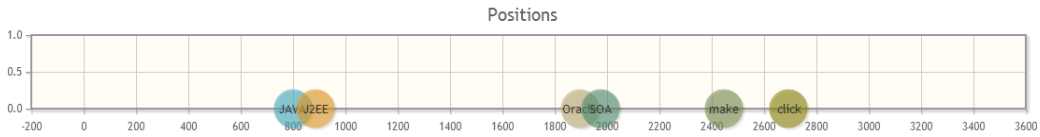
Στο παραπάνω παράδειγμα οι θέσεις των keywords είναι:

- Java = 882
- J2EE = 887
- Oracle = 1956

- SOA = 1976
- make = 2447
- Click = 2595

Μετά την εφαρμογή του τα παραπάνω Keywords έχουν διαχωριστεί σε 4 κλάσεις οι οποίες είναι:

- Κλάση 1: [J2EE(887), Java(882)]
- Κλάση 2: [Click(2595)]
- Κλάση 3: [SOA(1976), Oracle(1956)]
- Κλάση 4: [make(2447)]



Διάγραμμα 4.5: Θέσεις των keywords με βάση την απόσταση μεταξύ τους

Βλέπουμε εδώ ότι οι κλάσεις 1 και 3 έχουν τα Keywords τα οποία ήταν πολύ κοντά μεταξύ τους μέσα στο κείμενο ενώ οι 2 και 4 έχουν αυτά τα οποία απείχαν από όλα τα άλλα σημαντική απόσταση. Επομένως αυτό που εφαρμόστηκε στην εργασία μας ήταν να απορρίπτονται τα πρότυπα τα οποία ήταν τα μοναδικά που ανήκαν σε μια κλάση, όπως στο παράδειγμα τα click και make. Αυτό έχει μια μικρή πιθανότητα να απορριφθούν μερικές φορές και κάποια keywords που όντως ανήκουν στα ζητούμενα προσόντα, όμως μετά από εφαρμογή του αλγορίθμου σε σημαντικό αριθμό αγγελιών παρατηρήθηκε ότι το ποσοστό αυτό είναι ελάχιστο σε σύγκριση με το φιλτράρισμα που γίνεται από τα εσφαλμένα εντοπισμένα keywords.

Ο αλγόριθμος αποτελείται από 5 βήματα κατά τα οποία το 3 έως 5 επαναλαμβάνονται με βάση το κριτήριο τερματισμού που θέτει ο χρήστης.

Βήματα αλγορίθμου Maximin

Βήμα 1ο: Επιλέγουμε ένα τυχαίο πρότυπο $\Pi_{\tau_t} = \Pi_{\tau_1}$ $\tau_t = (1,\ldots,K)$ και με αυτό ορίζουμε την πρώτη κλάση $\omega_t = \omega_1$

Βήμα 2ο: Δημιουργούμε το σύνολο D_1 των αποστάσεων των προτύπων του S από το Π_{τ_1} Βρίσκουμε το πρότυπο Π_{τ_2} (τ_2=1,…,K) που απέχει την μέγιστη απόσταση M_1 από το Π_{τ_1}

Βήμα 3ο: Αυξάνουμε το t κατά ένα και ορίζουμε την κλάση ω_t με στοιχείο το Π_{τ_t}, ω_t = Π_{τ_t}

Βήμα 4ο: Ταξινομούμε κάθε $\Pi_k \in S$ στις τάξεις ω_i, (i=1,…,t) με το κριτήριο της ελάχιστης απόστασης. Δημιουργούμε τα σύνολα D_i των αποστάσεων των προτύπων κάθε κλάσης ω_i από το πρότυπο που όρισε την κλάση.

Βρίσκουμε την μέγιστη απόσταση M_k μεταξύ όλων των αποστάσεων των D_i και το αντίστοιχο πρότυπο Π_k το οποίο ονομάζουμε Π_{τ_t+1}

Βήμα 5ο: Αν $M_t M_{t+1} \leq \rho \ll 1$, όπου ρ θετικός προκαθορισμένος αριθμός σημαντικά μικρότερος της μονάδας, η διαδικασία σταματάει και το πλήθος των ομάδων είναι ο αριθμός t. Αλλιώς συνεχίζεται επαναληπτικά από το βήμα 3.

4.3 Web Service

4.3.1 Υλοποίηση

Η υλοποίηση του web service βασίστηκε στο πρότυπο REST, απλοποιώντας και δημιουργώντας ένα πρότυπο για την διαδικασία επικοινωνίας με τον client. Όλα τα ερωτήματα που δέχεται πρέπει να είναι σε μεθόδους GET και να ακολουθούν συγκεκριμένη αυστηρή σύνταξη. Για κάθε κλήση με τη μέθοδο GET, γίνεται έλεγχος και πιστοποίηση αν τα δεδομένα που ζητούνται είναι επαρκή και έχουν την απαραίτητη δομή και σε διαφορετική περίπτωση επιστρέφεται σχετικό μήνυμα που περιγράφει το σφάλμα.Τα δεδομένα που επιστρέφει είναι πάντα μορφοποιημένα σε JSON και περιέχουν τα εξής στοιχεία

success

(bool) Ενημερώνει αν το ερώτημα για τα αποτελέσματα που ζητήθηκαν ήταν επιτυχές ή όχι.

category

(string) Το όνομα της κατηγορίας στην οποία έγινε αναζήτηση.

numValues

(int) Ο αριθμός των επιστρεφόμενων τιμών.

keyId/catId

(int) Το αναγνωριστικό της κατηγορίας ή της λέξης προς αναζήτηση.

occurencies

(int) Ο αριθμός των εμφανίσεων

links

(string) Σύνδεσμοι που μας δείχνουν το τρέχων url όπου βρισκόμαστε και πού μπορεί να βρει κάποιος τεκμηρίωση για την επικοινωνία με το web service.

Για παράδειγμα ένα σωστό ερώτημα θα ήταν το παρακάτω:

```
1 ?q=keywordDate&keyID=1108&dateFrom=2012/01/06&dateTo=2012/10/22
```

Listing 4.5: Παράδειγμα σωστού ερωτήματος

και θα επέστρεφε αυτό:

```
1 {"success":true,"category":"keywordDate","numValues":2,"keyID":"1108
    ","occurencies":"172","links":{"self":[{"href":"http:\/\/api.
    jobtrends\/serve"},{"rel":"http:\/\/api.jobtrends\/common\/self"
    }]}}
```

Listing 4.6: Απάντηση σωστού ερωτήματος

ενώ το παρακάτω λανθασμένο ερώτημα:

```
1 ?q=keywordDate&keyID=1108&dateFrom=today&dateTo=2012/10/22
```

Listing 4.7: Παράδειγμα λανθασμένου ερωτήματος

επιστρέφει:

```
1 {"success":false,"error_code":null,"error_message":["The dateFrom
    must be valid date in YYYY\/DD\/MM format"],"links":{"self":[{"
    href":"http:\/\/api.jobtrends\/serve"},{"rel":"http:\/\/api.
    jobtrends\/common\/self"}]}}
```

Listing 4.8: Απάντηση λανθασμένου ερωτήματος

4.3.2 Factory Pattern

Για τις ανάγκες της δημιουργιας διαφορετικών επιστρεφόμενων τιμών ανα ερώτημα εφαρμόστηκε το πρότυπο σχεδίασης Factory, το οποίο αντιμετωπίζει το πρόβλημα της δημιουργίας νέων αντικειμένων κλάσεων(object instantiation) όταν ο κώδικας επικεντρώνεται σε μεθόδους abstract (Zandstra 2010). Συγκεκριμένα για την επίλυση του προβλήματος μας δημιουργήσαμε για κάθε ένα ερώτημα και μια ξεχωριστή κλάση η οποία ξέρει πώς να το επεξεργαστεί και να καλέσει τις απαραίτητες μεθόδους από την κλάση Database ώστε να επιστρέψει τα ζητούμενα αποτελέσματα στην κατάλληλη μορφοποίηση. Όλες οι αιτήσεις διαχειρίζονται από μία κεντρική κλάση, την requestHandlerFactory, και μια abstract την requestHandler όπως φαίνεται στον κώδικα παρακάτω.

```
1  abstract class requestHandler extends validation
2  {
3
4      abstract public function parseRequest($request);
5
6  }
```

Listing 4.9: Factory Method

```
1  class requestHandlerFactory
2  {
3
4      public static function getHandler($request)
5      {
6          if( isset($request['q']) )
7          {
8              switch ($request['q'])
9              {
10                 case 'status':
11                     $handler = new statusHandler();
12                     break;
13                 case 'categories':
14                     $handler = new categoriesHandler();
15                     break;
16                 case 'keywords':
17                     $handler = new keywordsHandler();
```

```
18                      break;
19                  case 'keywordNoDate':
20                      $handler = new keywordNoDateHandler();
21                      break;
22                  case 'keywordDate':
23                      $handler = new keywordDateHandler();
24                      break;
25                  case 'categoryNoDate':
26                      $handler = new categoryNoDateHandler();
27                      break;
28                  case 'categoryDate':
29                      $handler = new categoryDateHandler();
30                      break;
31                  case 'allOccurencies':
32                      $handler = new allOccurenciesHandler();
33                      break;
34                  default:
35                      $handler = new errorHandler();
36              }
37          }else
38          {
39              $handler = new defaultHandler();
40          }
41          return $handler;
42      }
```

Listing 4.10: Factory Method

Ο τρόπος που χρησιμοποιούμε τις 2 παραπάνω κλάσεις είναι να πάρουμε μέσω της μεθόδου GET την αίτηση που έχει γίνει και να την περάσουμε ως όρισμα στην static μέθοδο getHandler ως εξής:

```
1       $request = $_GET;
2
3       $handler = requestHandlerFactory::getHandler($request);
```

Listing 4.11: Factory Method Instantiate

4.3.3 Request Validation

Επειδή ο τρόπος που γίνονται οι κλήσεις των μεθόδων που επιστρέφουν τα δεδομένα που ζητάει ο πελάτης γίνεται μέσω της μεθόδου GET η οποία μπορεί εύκολα να παραποιηθεί και να ζητήσει εσφαλμένα δεδομένα ή να δημιουργήσει πρόβλημα στο σύστημα μέσω injections, δημιουργήθηκε η κλάση Validation η οποία κάνει πιστοποίηση των δεδομένων που έστειλε ο πελάτης πριν τα στείλει ως παραμέτρους στις μεθόδους των κλάσεων, ώστε να εξακριβωθεί ότι αυτά που ζητάει είναι στην αναμενόμενη μορφή και έχουν όλες τις απαραίτητες παραμέτρους.Για παράδειγμα στα αιτήματα που απαιτείται ημερολογιακό εύρος είναι απαραίτητο, όπως θα δούμε αναλυτικά στην επόμενη ενότητα, να στείλει ο πελάτης και την ημερομηνία ”από” και ”έως” την περίοδο που επιθυμεί σε συγκεκριμένη μορφή ώστε να επιστραφεί σωστό αποτέλεσμα. Επομένως με την παρακάτω μέθοδο της κλάσης γίνεται ο απαραίτητος έλεγχος:

```
1     /*
2      * @param string $sDate
3      * @return bool
4      */
5     public function isValidDate($sDate)
6     {
7         $sDate = str_replace(' ', '-', $sDate);
8         $sDate = str_replace('/', '-', $sDate);
9         $sDate = str_replace('--', '-', $sDate);
10        if( preg_match('/^(\d{4})-(\d{2})-(\d{2})$/', $sDate,
              $matches) == 1)
11        {
12            return checkdate($matches[2], $matches[3], $matches[1]);
13        }
14        return false;
15    }
```

Listing 4.12: Validate Date Request

Για να ορίσουμε πόσες και τί παραμέτρους περιμένει στις μεθόδους της η κάθε κλάση που επεξεργάζεται το κάθε ένα request, ορίζουμε στον constructor της τα μέρη που αναμένεται να πάρει καθώς και τον τύπο τους ως εξής:

```
1 class keywordDateHandler extends requestHandler
2 {
```

```
3     public function __construct()
4     {
5         $this->expected['q'] = 'string';
6         $this->expected['keyID'] = 'numeric';
7         $this->expected['dateFrom'] = 'date';
8         $this->expected['dateTo'] = 'date';
9     }
```

Listing 4.13: Validate Date Request

Και έπειτα στην κλάση validation έχουμε ορίσει, όπως παραπάνω με την ημερομηνία και την πιστοποίηση των υπολοίπων αναμενόμενων τύπων αιτημάτων.

4.3.4 Επικοινωνία με το API

Η επικοινωνία με το web service γίνεται μέσω συγκεκριμένων αιτημάτων http. Αφού επεξεργαστεί το ερώτημα, δημιουργείται η απάντηση και επιστρέφεται σε μορφή JSON στον πελάτη. Η βασική σύνταξη για την επικοινωνία με το API του web service είναι η εξής:

```
1  http://DOMAIN/webservice/?q=REQUEST
```

Όπου ”DOMAIN” μπορεί να είναι οποιοδήποτε domain name ή ip και το ”REQUEST” παίρνει μια από τις παρακάτω τιμες

status

Επιστρέφει την κατάσταση του web service.

Παράμετροι

καμία

categories

Επιστρέφει όλες τις κατηγορίες των keywords.

Παράμετροι

καμία

keywords

Επιστρέφει όλα τα keywords στα οποία μπορεί να γίνει αναζήτηση.

Παράμετροι

καμία

keywordNoDate

Επιστρέφει τον αριθμό των εμφανίσεων του keyword που ζητήθηκε, σε όλο το ημερολογιακό διάστημα των αναζητήσεων.

Παράμετροι

keyId το μοναδικό id της κατηγορίας

keywordDate

Επιστρέφει τον αριθμό των εμφανίσεων του keyword που ζητήθηκε, σε συγκεκριμένο εύρος ημερομηνιών.

Παράμετροι

- keyID το μοναδικό id του keyword
- dateFrom η ημέρα από την οποία θα ξεκινήσει στην μορφή YYY/MM/DD
- dateTo η ημέρα από την οποία θα ξεκινήσει στην μορφή YYY/MM/DD

categoryNoDate

Επιστρέφει τον αριθμό των εμφανίσεων των keywords στην κατηγορία που ζητήθηκε, σε όλο το διάστημα των αναζητήσεων. Παράμετροι: catID το μοναδικό id της κατηγορίας

categoryDate

Επιστρέφει τον αριθμό των εμφανίσεων των keywords στην κατηγορία που ζητήθηκε, σε συγκεκριμένο εύρος ημερομηνιών.

Παράμετροι

- catID το μοναδικό id της κατηγορίας
- dateFrom η ημέρα από την οποία θα ξεκινήσει στην μορφή YYY/MM/DD
- dateTo η ημέρα από την οποία θα ξεκινήσει στην μορφή YYY/MM/DD

4.4 Περιβάλλον διαχείρισης

4.4.1 Model-View-Controller Pattern

Η δομή του βασίζεται στην αρχιτεκτονική MVC, η οποία είναι ευρέως διαδεδομένη στις web εφαρμογές λόγω των δυνατοτήτων και της ευχρηστίας που προσφέρει.Σε αυτήν, η εφαρμογή χωρίζεται σε 3 διαφορετικά αλληλένδετα επίπεδα και ειδικότερα:

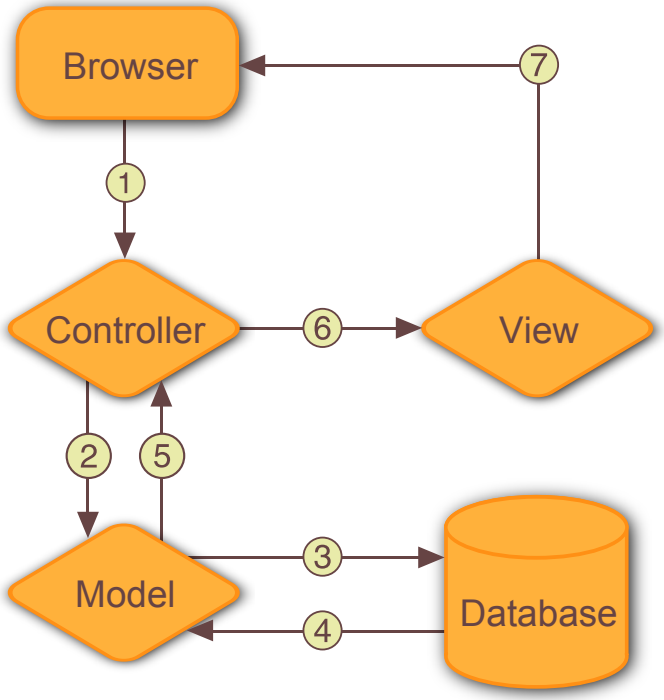
το Model όπου αποτελεί το κυρίως μέρος της επικοινωνίας με την βάση δεδομένων με σκοπό την επεξεργασία των στοιχείων που ζητούνται, την καταχώρηση τους αλλά και ακόμη πιο σύνθετες λειτουργίες πάνω σε αυτά τα δεδομένα, όπως εξειδικευμένη κατηγοριοποίηση και έλεγχος εγκυρότητας. Αυτό το επίπεδο δεν γνωρίζει ούτε πώς αλλά ούτε και σε ποιόν να δείξει τα δεδομένα που παράγει.

το View αποτελεί την υλοποίηση της εμφάνισης των δεδομένων που παράγονται ύστερα από επεξεργασία στο Model, έτσι ώστε να τα δει ο χρήστης που τα ζήτησε. Είναι υπεύθυνο για την κατάλληλη μορφοποίηση των δεδομένων με τέτοιον τρόπο ώστε να κάνει την εφαρμογή να φαίνεται αντίστοιχη των δυνατοτήτων της, μιας και είναι αποκλειστικά το κομμάτι εκείνο της εφαρμογής με το οποίο έχει επαφή ο χρήστης. Αυτό το επίπεδο δεν γνωρίζει πώς να παράγει δεδομένα αλλά ούτε και από ποιόν να τα ζητήσει.

ο Controller αποτελεί το συνδετικό κομμάτι ανάμεσα στο model και στο view. Είναι εκείνος που ευθύνεται να δεχθεί και να επεξεργαστεί το ερώτημα που ζητάει ο χρήστης και έπειτα να γνωρίζει πώς να καλέσει το κατάλληλο μοντέλο για την παραγωγή των δεδομένων και αφού αυτή ολοκληρωθεί να τα περάσει στο κατάλληλο view για να τα εμφανιστούν στον χρήστη.Αυτό το επίπεδο δεν γνωρίζει ούτε πώς να παράγει δεδομένα αλλά ούτε και πώς να τα εμφανίσει.

4.4.2 Επεξεργασία δεδομένων με AJAX και JQuery

Η εμφάνιση και η επεξεργασία των δεδομένων στο Control Panel της εφαρμογής γίνεται μέσω της τεχνολογίας AJAX που στηρίζεται στην ασύγχρονη μεταφορά δεδομένων κάτι που σημαίνει ότι δεν χρειάζεται να γίνεται φόρτωση της σελίδας για να εμφανιστούν νέα δεδομένα ή να ανανεωθούν τα υπάρχοντα. Η τεχνολογία αυτή χρησιμοποιεί μεθόδους της Javascript που περιγράφονται από το XMLHttpRequest API το



Διάγραμμα 4.6: Το πρότυπο MVC

οποίο είναι αυτό που ξέρει πώς να στέλνει και να λαμβάνει δεδομένα από τον server στο παρασκήνιο, χωρίς να διακόπτει την εμπειρία του χρήστη.

```
var xmlhttp;
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","http://www.DOMAIN.com/data_serve.php",true);
```

```
18 xmlhttp.send();
```

Listing 4.14: Παράδειγμα εκτέλεσης AJAX

Οι μέθοδοι και ο τρόπος επικοινωνίας με AJAX ακολουθούν πάντα το ίδιο πρότυπο σύνταξης σε Javascript και για αυτό η ευρέως διαδεδομένη βιβλιοθήκη JQuery έχει ενσωματώσει τις μεθόδους που απαιτούνται και έχει κάνει ιδιαίτερα απλή την εφαρμογή μιας ασύγχρονης επικοινωνίας του client με κάποιον server. Επίσης δίνει την δυνατότητα να εμφανιστούν τα επιστρεφόμενα δεδομένα σε οποιοδήποτε στοιχείο του DOM της σελίδας HTML.

```
1 $.ajax({
2   url: 'http://www.DOMAIN.com/data_serve.php',
3   success: function(data) {
4     $('.result').html(data);
5     alert('Load was performed.');
6   }
7 });
```

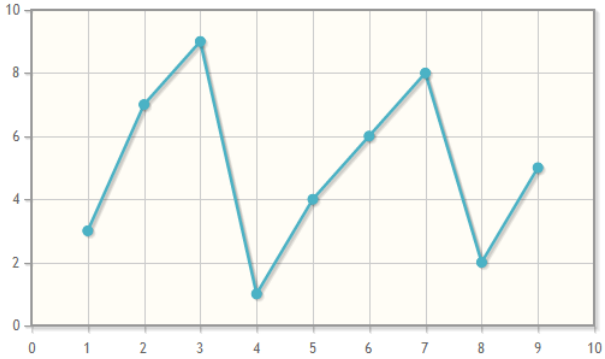
Listing 4.15: Παράδειγμα εκτέλεσης AJAX με JQuery

4.4.3 Σχεδίαση διαγραμμάτων - Jqplot

Για την οπτικοποίηση των στατιστικών δεδομένων που εξάγονται από τα στοιχεία που είναι καταχωρημένα στην βάση δεδομένων από τον web scraper χρησιμοποιήθηκε η Javascript βιβλιοθήκη jQplot που αποτελεί επέκταση της βιβλιοθήκης jQuery. Παρέχει δυνατότητες εμφάνισης μεγάλου όγκου δεδομένων σε διαφορετικές μορφές απεικόνισης, όπως γραφήματα με μπάρες, πίτες, γραμμικά αλλά και διαγράμματα ανα εύρος ημερομηνιών. Ο τρόπος χρήσης της για τον κάθε τύπο γραφήματος είναι διαφορετικός, αλλά ακολουθεί μια βασική σύνταξη που περιέχει το όνομα του στοιχείου(DIV) της σελίδας που θα εφαρμοστεί και τα αριθμητικά δεδομένα σε μορφή πίνακα:

```
1 $(document).ready(function(){
2   var plot1 = $.jqplot ('chart1', [[3,7,9,1,4,6,8,2,5]]);
3 });
```

Listing 4.16: Παράδειγμα γραφήματος με jQplot



Διάγραμμα 4.7: Παράδειγμα γραφήματος

4.5 Διασφάλιση ποιότητας

Η αποτελεσματικότητα της εφαρμογής μας διασφαλίζεται μέσα από τα σενάρια δοκιμών που έχουν δημιουργηθεί για το μεγαλύτερο μέρος του συνόλου των μεθόδων των κλάσεων του scraper και του web service, γνωστά ως Unit Testing, το οποίο υλοποιήθηκε στο ίδιο περιβάλλον που αναπτύχθηκε η εφαρμογή,το NetBeans, με χρήση της πρόσθετης βιβλιοθήκης phpUnit η οποία επιταχύνει τη δημιουργία τέτοιων μεθόδων. Μέσα από το unit testing η εύρεση κρίσιμων σφαλμάτων στον κώδικα γίνεται πιο γρήγορα και έτσι απλοποιείται η διαδικασία επίλυσης προβλημάτων κάνοντας την εφαρμογή αξιόπιστη και αποτελεσματική. Η ενσωμάτωση της βιβλιοθήκης phpUnit στο Netbeans IDE είναι μια απλή διαδικασία που περιγράφεται στην επίσημη τεκμηρίωση της εφαρμογής.

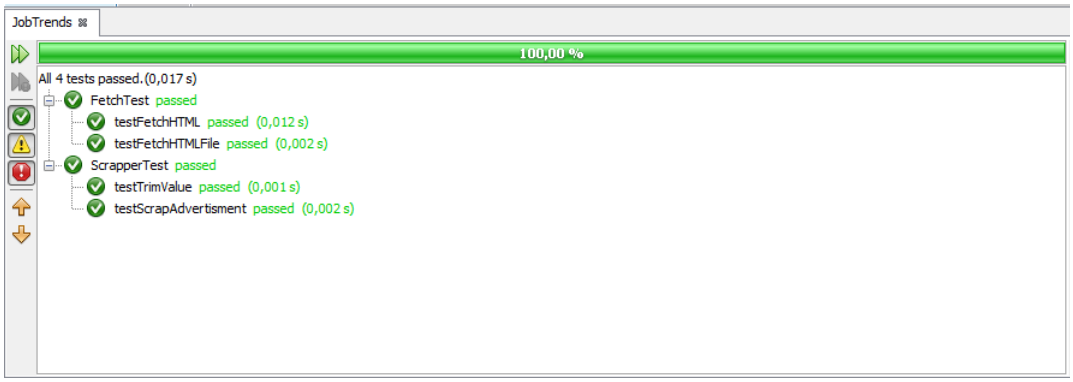
Παρακάτω παρατίθενται ενδεικτικά κάποια από τα unit test που υλοποιήθηκαν καθώς και το αποτέλεσμα της εκτέλεσης τους μέσω του IDE του NetBeans.

```
1  /**
2  * @covers Fetch::FetchHTML()
3  * Test if webpage exists
4  */
5  public function testFetchHTML()
6  {
7    $url='http://www.google.gr';
8    $filename='fetchHTML.html';
9    $this->assertEquals('200',$this->object->fetchHTML($url, $filename
         ));
10 }
```

Listing 4.17: Παράδειγμα unit testing - Κλάση Fetch

```
1  /**
2  * @covers scraper::scrapAdvertisment()
3  * Test if a sentence is converted to an array of words
4  */
5  public function testScrapAdvertisment()
6  {
7    $html = '<html><div id='someDiv'>This is a sample text</div></html
         >';
8    $xpath = '//*[@id="content"]';
9    $words=$this->object->scrapAdvertisment($html, $xpath);
10   $this->assertNotEmpty($words);
11 }
```

Listing 4.18: Παράδειγμα unit testing - Κλάση Scrap



Διάγραμμα 4.8: Αποτελέσματα των παραπάνω δειγμάτων unit test

Κεφάλαιο 5

Αποτελέσματα - Στατιστικά δεδομένα

Αφού έχουν περάσει σχεδόν *12 μήνες* συλλογής δεδομένων από τις ιστοσελίδες αγγελιών στην περίοδο *15-01-2012* με *10-12-2012*, έχουμε ένα ικανοποιητικό αποτέλεσμα στατιστικών στοιχείων από το οποίο μπορούμε να εξάγουμε χρήσιμα συμπεράσματα για τις τάσεις που επικρατούν στα προσόντα γύρω από τον χώρο της πληροφορικής.Παρακάτω μπορούμε να δούμε μερικά στοιχεία σχετικά με αυτά τα δεδομένα μέσα από την αναζήτηση που έγινε από τον web scraper στις αγγελίες.

Συνολικές αγγελίες στις οποίες έγινε αναζήτηση:	3521
Εμφανίσεις προσόντων στις αγγελίες:	21691
Μοναδικά προσόντα ανάμεσα στις εμφανίσεις:	461
Ποσοστό μοναδικών προσόντων επί του συνόλου των εμφανίσεων:	2,1%
Εμφανίσεις που θεωρήθηκαν ως άκυρες (MaxiMin):	8003

Πίνακας 5.1: Γενικές στατιστικές πληροφορίες - έως 10/12/2012

Ιστοσελίδες στις οποίες έγινε αναζήτηση:	3
Ιστοσελίδες από Ελλάδα:	2
Ιστοσελίδες από Αμερική:	1
Αγγελίες από Αμερική:	1365
Αγγελίες από Ελλάδα:	2156

Πίνακας 5.2: Πληροφορίες ιστοσελίδων - έως 10/12/2012

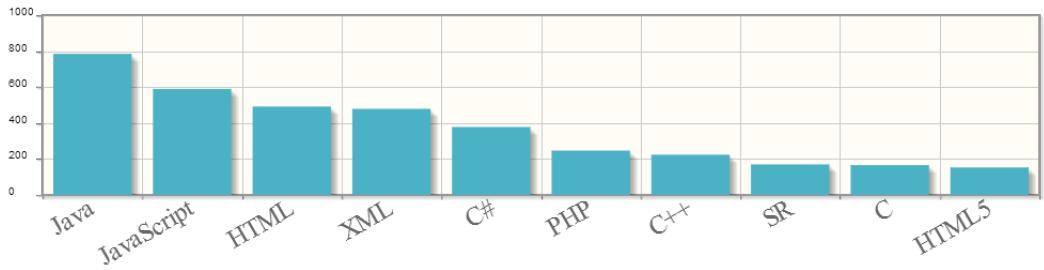
Στις επόμενες ενότητες βλέπουμε μέσω διαγραμμάτων την κατανομή των εμφανί-

σεων των αγγελιών. Αν και έχει γίνει αρκετά καλός διαχωρισμός των προσόντων που πιθανών να είναι άκυρα, μέσω του αλγορίθμου MaxiMin, υπάρχουν σίγουρα κάποια δεδομένα που είναι λανθασμένα αλλά δεν επηρεάζουν σε σημαντικό βαθμό τα αποτελέσματα.

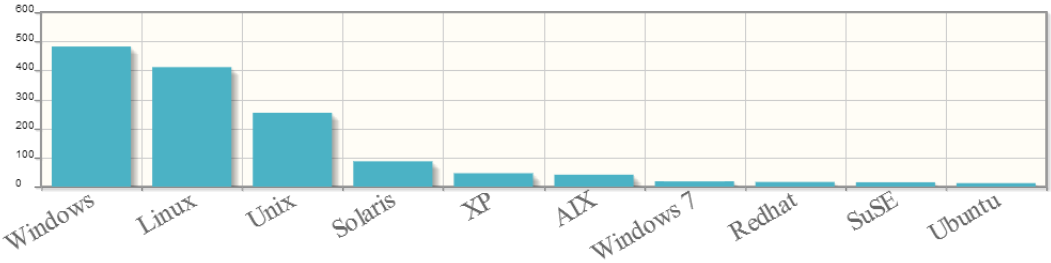
Στο κάθε διάγραμμα σε μορφή μπάρας ο κάθετος άξονας Ψ δείχνει τον αριθμό των εμφανίσεων του κάθε προσόντος που φαίνεται στον οριζόντιο άξονα Χ.

5.1 Τάσεις ανα κατηγορία

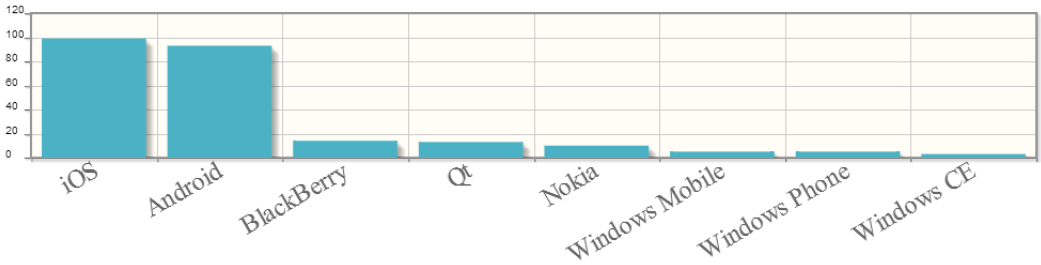
Στους παρακάτω πίνακες μπορούμε να δούμε τα 10 δημοφιλέστερα προσόντα που αναζητούνται σε 5 δημοφιλείς κατηγορίες: γλώσσες προγραμματισμού, λειτουργικά συστήματα, λειτουργικά κινητών συσκευών, RDBMS και compilers



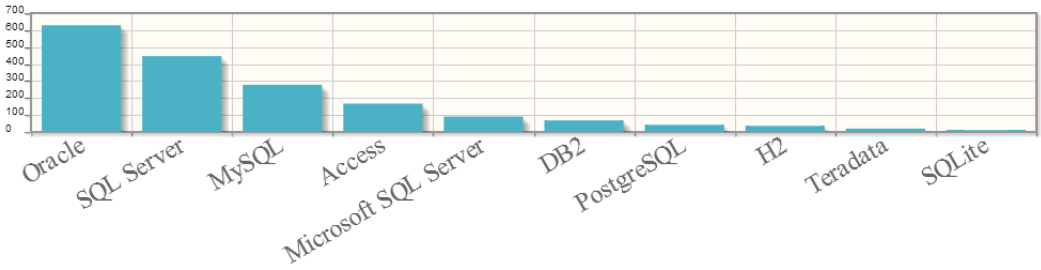
Διάγραμμα 5.1: Top 10 Programming Languages - έως 10/12/2012



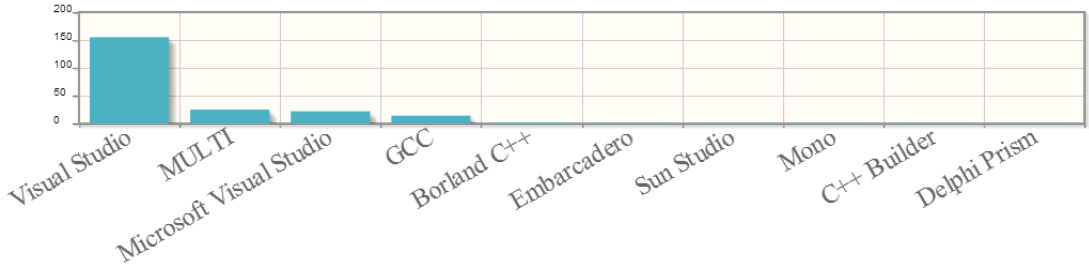
Διάγραμμα 5.2: Top 10 Operating Systems - έως 10/12/2012



Διάγραμμα 5.3: Top 10 Mobile Operating Systems - έως 10/12/2012



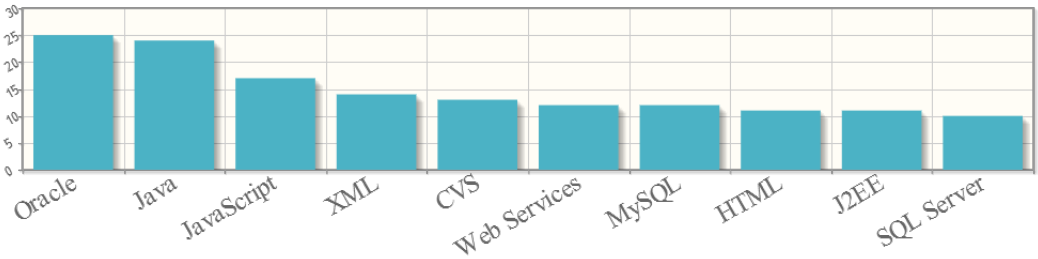
Διάγραμμα 5.4: Top 10 RDBMS - έως 10/12/2012



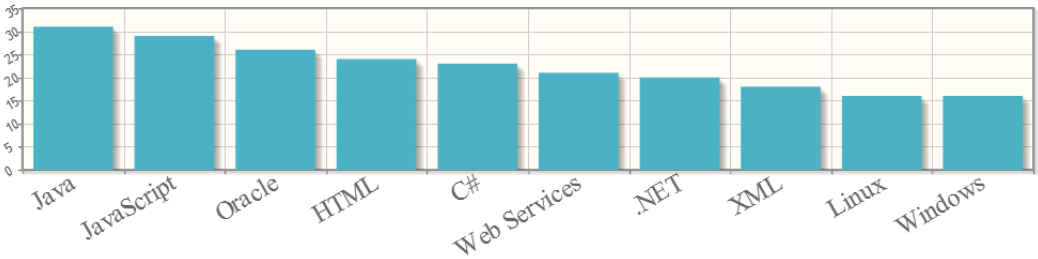
Διάγραμμα 5.5: Top 10 Compilers - έως 10/12/2012

5.2 Τάσεις ανα μήνα

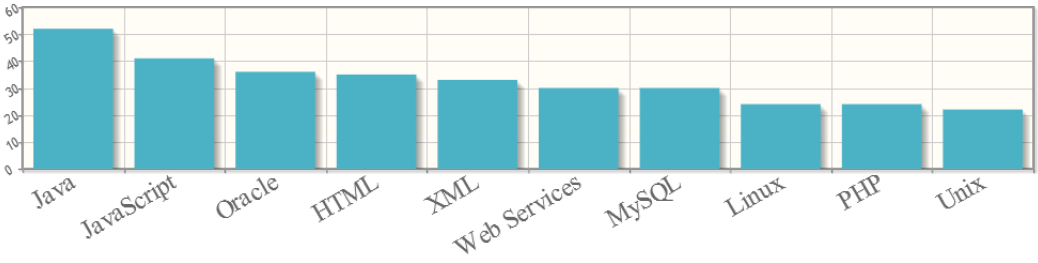
Στους παρακάτω πίνακες βλέπουμε συγκεντρωτικά τα 10 δημοφιλέστερα προσόντα που αναζητήθηκαν για κάθε μήνα για το διάστημα που πραγματοποιήθηκε η αναζήτηση.



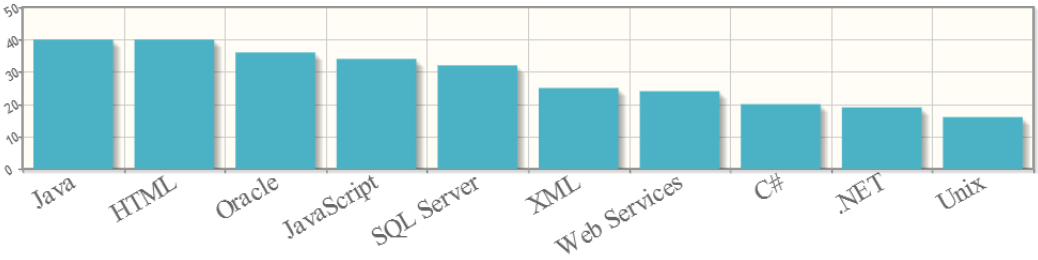
Διάγραμμα 5.6: Ιανουάριος top 10



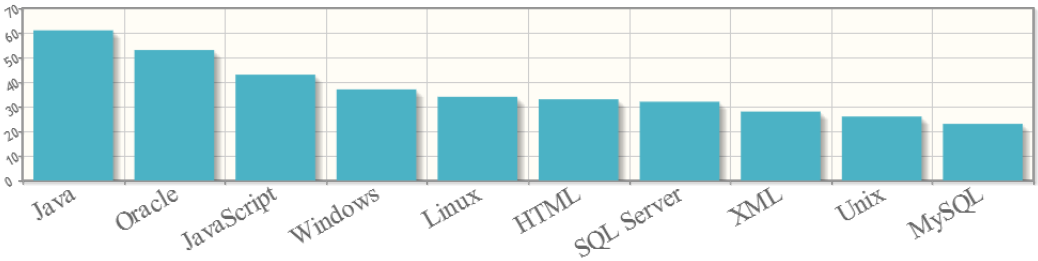
Διάγραμμα 5.7: Φεβρουάριος top 10



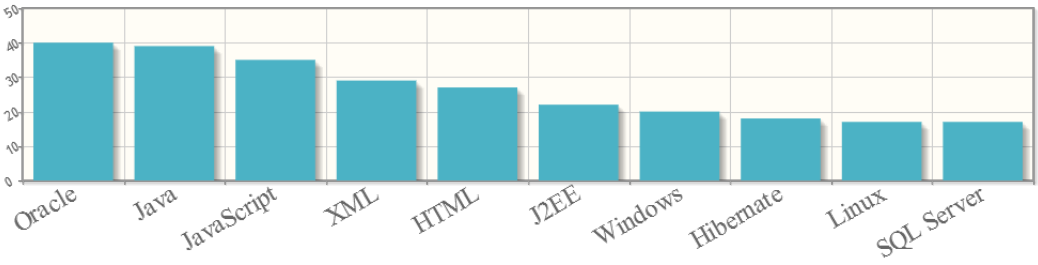
Διάγραμμα 5.8: Μάρτιος top 10



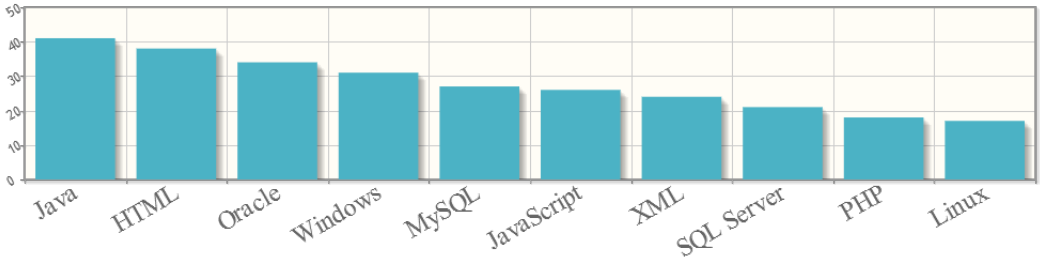
Διάγραμμα 5.9: Απρίλιος top 10



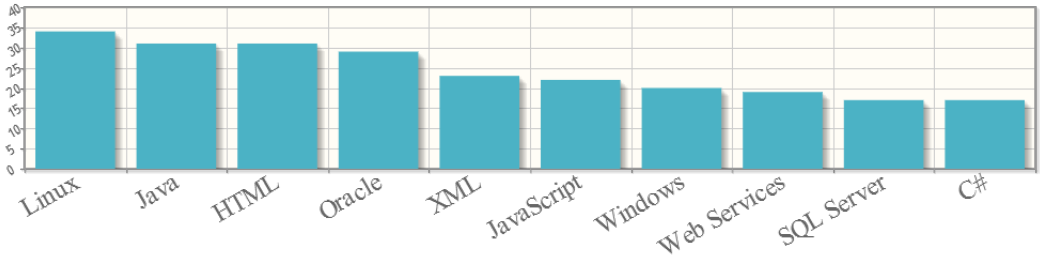
Διάγραμμα 5.10: Μάιος top 10



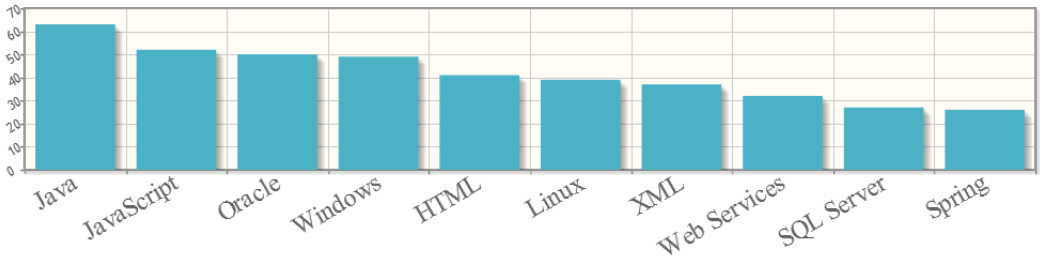
Διάγραμμα 5.11: Ιούνιος top 10



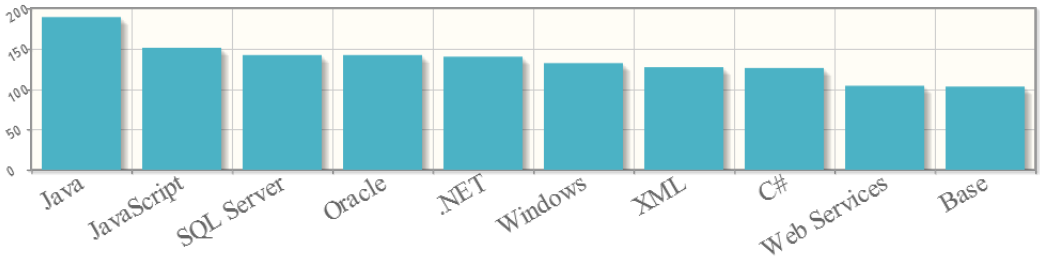
Διάγραμμα 5.12: Ιούλιος top 10



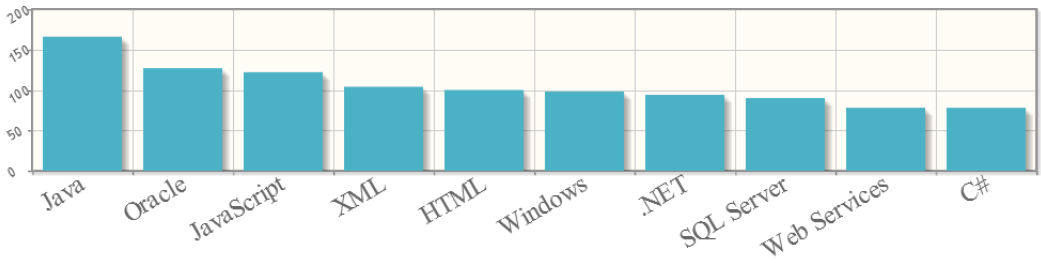
Διάγραμμα 5.13: Αύγουστος top 10



Διάγραμμα 5.14: Σεπτέμβριος top 10



Διάγραμμα 5.15: Οκτώβριος top 10



Διάγραμμα 5.16: Νοέμβριος top 10

Κεφάλαιο 6

Συμπεράσματα από την εφαρμογή

Έχοντας στη διάθεση μας τα δεδομένα που έχουν συλλεχθεί όλο αυτό το διάστημα, άλλα και συνεχίζουν να συλλέγονται καθημερινά, είμαστε πλέον σε θέση να έχουμε μια πιο εμπεριστατωμένη άποψη σχετικά με τα προσόντα που ζητούνται από την ελληνική αλλά και τη διεθνή αγορά εργασίας του χώρου της πληροφορικής. Πολλά από αυτά τα στοιχεία που προέκυψαν ήταν σε μεγάλο βαθμό αναμενόμενα, για κάποια άλλα υπάρχει μια έκπληξη, αλλά αυτό που έχει σημασία είναι ότι έχουμε μια πραγματική εικόνα με έγκυρα στοιχεία που δεν βασίζεται πια στην εμπειρία του κάθε ενός γύρω από τον χώρο της πληροφορικής. Έτσι οι νέοι άνθρωποι που θέλουν να ασχοληθούν πάνω σε αυτόν τον τομέα, αλλά και όσοι ασχολούνται ήδη μπορούν να στρέψουν τις δυνατότητες τους προς τα εκεί που υποδεικνύουν τα στοιχεία της εργασίας ώστε να έχουν τη δυνατότητα να ακολουθούν από πιο κοντά τις εξελίξεις στον ταχύτατα αναπτυσσόμενο αυτόν τομέα.

6.1 Διαχείριση έργου

Το συνολικό διάστημα που δαπανήθηκε από την ημερομηνία ανάληψης της εργασίας μέχρι την ολοκλήρωση της είναι ένα ημερολογιακό έτος, από τον *Νοέμβριο 2011* έως τον *Νοέμβριο του 2012*. Το διάστημα αυτό θα ήταν σημαντικά μικρότερο, εάν δεν μεσολαβούσε ενδιάμεσα η υποχρεωτική εκπόνηση της εξάμηνης πρακτικής άσκησης(*Απρίλιος - Σεπτέμβριος*), κάτι που συνετέλεσε αρνητικά στην πορεία ανάπτυξης της εργασίας και των αρχικών ορίων που είχα θέσει. Αν αφαιρέσουμε αυτό το διάστημα, ο πραγματικός χρόνος υλοποίησης δεν ξεπερνά τους 5 μήνες συνολικά,

συμπεριλαμβανομένης και της συγγραφής του παρόντος συγγράμματος.

Αρχικά έγινε μια ανάλυση των απαιτήσεων της εφαρμογής και μια πρώτη συγκέντρωση πληροφοριών γύρω από το αντικείμενο της εργασίας, μέσα από υπάρχουσες παρόμοιες εργασίες και από διάφορες πηγές βιβλιογραφίας. Η υλοποίηση ξεκίνησε με την εύρεση όσο το δυνατόν περισσότερων προσόντων για τον εμπλουτισμό των λέξεων κλειδιών προς αναζήτηση. Έπειτα υλοποιήθηκε ο web scraper, όπου και δαπανήθηκε το μεγαλύτερο μέρος του χρόνου υλοποίησης και στη συνέχεια η web service. Τέλος, αν και δεν ήταν απαιτούμενο από τις προϋποθέσεις της εργασίας, υλοποίησα το περιβάλλον διαχείρισης της και δαπάνησα αρκετό πολύτιμο χρόνο (κυρίως στην εμφάνιση του), που ίσως να μην ήταν απαραίτητο στην παρούσα φάση και να μπορούσε να γίνει σε μελλοντική εξέλιξη της εργασίας.

Για την διαχείριση του κώδικα της εφαρμογής χρησιμοποιήθηκε ένα αποθετήριο SubVersioN, στο οποίο αποτυπώθηκε όλη η εξέλιξη του έργου, δίνοντας δυνατότητες διαχείρισης αλλά και καλύτερης επίβλεψης της πορείας του από τον επιβλέποντα καθηγητή.

Τα διάφορα προβλήματα που προέκυψαν κατά τα στάδια υλοποίησης της εφαρμογής ήταν κυρίως τεχνικής φύσεως και αφορούσαν το πώς να υλοποιηθούν τα τμήματα των απαιτήσεων με τον βέλτιστο δυνατό τρόπο. Η επίλυση τους έγινε με προσωπική αναζήτηση κυρίως από πηγές του διαδικτύου και από υπάρχουσα βιβλιογραφία πάντα υπό την καθοδήγηση του επιβλέποντα καθηγητή.

6.2 Προτάσεις εξέλιξης

Η εφαρμογή υλοποιεί σε μεγάλο βαθμό τους αρχικούς στόχους της, ωστόσο σίγουρα επιδέχεται βελτίωσης και περαιτέρω εξέλιξης. Ένα σημείο που θα μπορούσε να βελτιωθεί ίσως να είναι στο σημείο που γίνεται ο διαχωρισμός των λέξεων κλειδιών που εντοπίζονται εσφαλμένα και τώρα εντοπίζονται και διαχωρίζονται με τη χρήση του αλγορίθμου MaxiMin, να πραγματοποιηθεί με κάποιον καλύτερο τρόπο ώστε να υπάρχουν όσο το δυνατόν μικρότερες απώλειες. Οι δυνατότητες της Web Service είναι πάρα πολλές και μέσω του API της θα μπορούσε να προσφέρει ακόμη περισσότερα δεδομένα, σε περισσότερες μορφές, όπως XML και με περισσότερες λειτουργίες προς τον χρήστη. Θα μπορούσε επίσης να υλοποιηθεί με χρήση κάποιο API key ώστε να

παρακολουθείται η ζήτηση των δεδομένων της από τους ιστοτόπους ή τους χρήστες που τα ζητούν και να μπορούν να προκύψουν στατιστικά χρήσης της. Ένας άλλος τομέας που θα μπορούσε να επεκταθεί αυτή η εργασία είναι αυτός του Natural Language Proccessing όπου θα μπορούσε να εξάγεται και σημασιολογικό περιεχόμενο από τις αγγελίες ώστε να καταχωρούνται και πιο αφηρημένες έννοιες εκτός των προσόντων ως keyword, όπως η προϋπηρεσία, ο μισθός και άλλα.

Γλωσσάρι

RDBMS Relational Database Management System

SQL Structured Query Language

PHP PHP: Hypertext Preprocessor

REGEX Regular Expression

XPATH XML Path Language

XML Extensible Markup Language

RSS Really Simple Syndication

ORM Object-Relational Mapping

REST REpresentational State Transfer

API Application Programming Interface

JSON JavaScript Object Notation

AJAX Asynchronous JavaScript and XML

DOM Document Object Model

MVC Model–View–Controller

Βιβλιογραφία

Aken, Andrew (2012). *Degree-Oriented Guide to Skills in Information Technology*. (Επίσκεψη 2012).

Group, The PHP (2012). *PHP Manual*. (Επίσκεψη 2012).

IETF (2005). *Uniform Resource Identifier (URI): Generic Syntax*. URL: `http://www.ietf.org/rfc/rfc3986.txt` (επίσκεψη 2011).

Litecky, C. κ.ά. (2010). «Mining for Computing Jobs». Στο: *Software, IEEE* 27.1, σσ. 78—85.

Masse, Mark (2011). *REST API Design Rulebook*. O'Reilly.

Papazoglou, Michael P. (2006). *Web Services: Principles and Technology*. Pearson.

Raymond Greenlaw, H. James Hoover (1998). *Fundamentals of the Theory of Computation: Principles and Practice*. Morgan Kaufmann.

W3C (2010). *XML Path Language (XPath) 2.0 (Second Edition)*. URL: `http://www.w3.org/TR/xpath20/` (επίσκεψη 2012).

Zandstra, Matt (2010). *PHP Objects, Patterns and Practice*. APress.

Στρουθόπουλος, Δρ Χαράλαμπος Π. (2003). *Αναγνώριση Προτύπων - Νευρωνικά Δίκτυα*. ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΣΕΡΡΩΝ.

Η εργασία αυτή στοιχειοθετήθηκε με το πρόγραμμα XeLaTeX. Για τη στοιχειοθέτηση της βιβλιογραφίας χρησιμοποιήθηκε το πρόγραμμα `biber` και `biblatex`. Οι γραμματοσειρές που χρησιμοποιήθηκαν είναι οι Times New Roman και `Courier New`.