ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΤΕΧΝΟΛΟΓΙΕΣ ΕΞΑΓΩΓΗΣ ΠΕΡΙΕΧΟΜΕΝΟΥ ΑΠΟ ΙΣΤΟΣΕΛΙΔΕΣ ΓΙΑ ΑΝΑΛΥΣΗ ΑΓΓΕΛΙΩΝ

Διπλωματική Εργασία

του

Πέτσκου Δημητρίου

Θεσσαλονίκη, Ιούνιος 2019

ΤΕΧΝΟΛΟΓΙΕΣ ΕΞΑΓΩΓΗΣ ΠΕΡΙΕΧΟΜΕΝΟΥ ΑΠΟ ΙΣΤΟΣΕΛΙΔΕΣ
ΓΙΑ ΑΝΑΛΥΣΗ ΑΓΓΕΛΙΩΝ

Πέτσκος Δημήτριος

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ
ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Εμμανουήλ Στειακάκης

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 24/06/2019

Στειακάκης Εμμανουήλ	Χατζηγεωργίου Αλέξανδρος	Γεωργιάδης Χρήστος
.....	.....	.....

Πέτσκος Δημήτριος

.....

Περίληψη

Η χρήση του διαδικτύου σε όλες τις πτυχές της καθημερινότητας μας έχει επηρεάσει και τον τρόπο αναζήτησης και δημοσίευσης θέσεων εργασίας. Λόγω του μεγάλου όγκου αγγελιών που κατακλύζει πλέον τον παγκόσμιο ιστό, μια αυτοματοποιημένη συλλογή τους μπορεί να βοηθήσει στην εξαγωγή χρήσιμων συμπερασμάτων. Στην παρούσα εργασία παρουσιάζονται οι τεχνολογίες που επιτρέπουν μια τέτοιου είδους αυτοματοποιημένη ανάκτηση. Μέσω μιας υλοποίησης σε Java παρουσιάζεται ο τρόπος που μπορεί να χρησιμοποιηθεί η τεχνολογία του Web Crawling και πως δεδομένα αγγελιών μπορούν να συγκεντρωθούν σε μια βάση για περαιτέρω αξιοποίηση. Τέλος παρατίθενται παραδείγματα γραφημάτων που μπορούν να προκύψουν από τα ανακτημένα δεδομένα.

Λέξεις Κλειδιά: Ανάκτηση αγγελιών από ιστοσελίδες, Web Crawler, Web Scrapper, Java, Selenium, XPath, MySQL

Abstract

The use of the Internet in all the aspects of everyday life has affected the way job advertisements are searched and published. Due to the high volume of job advertisements that floods the Worldwide Web, their automatic collection can help in the extraction of useful conclusions. In this thesis the technologies that allow such an automatic retrieval are presented. Through an implementation in Java, the way the technology of Web Crawling can be used is presented and how the data of job advertisements can be collected into a database for further use. Finally, examples of graphs are given, which can be created using the retrieved data.

Keywords: Web retrieval of job advertisements, Web Crawler, Web Scrapper, Java, Selenium, XPath, MySQL

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

1 Εισαγωγή

Η ραγδαία εξέλιξη των τεχνολογιών του διαδικτύου έχει αλλάξει εντελώς την διαδικασία αναζήτησης εργασίας. Για τους υπεύθυνους ανθρώπινου δυναμικού και στελέχωσης μια εταιρείας είναι ευκολότερο από ποτέ να βρουν υποψήφιους που να ταιριάζουν στις εξειδικευμένες δεξιότητες μιας θέσης ή ακόμη και να έρθουν σε επικοινωνία με υποψηφίους οι οποίοι ενδέχεται να μην αναζητούν ενεργά εργασία. Από την άλλη, όσοι αναζητούν εργασία μπορούν να προσεγγίσουν απευθείας τους εργοδότες, να δημιουργήσουν ηλεκτρονικά προφίλ για να τους προσελκύσουν και να υποβάλουν αίτηση για εργασία στέλνοντας το βιογραφικό τους σημείωμα απλά με το πάτημα ενός κουμπιού.

Στο παρελθόν, ο μόνος τρόπος αναζήτησης ήταν μέσω τυπωμένων αγγελιών σε κάποια εφημερίδα ή με την επίσκεψη σε κάποια έκθεση ενημέρωσης για επαγγελματικά θέματα και θέσεις εργασίας, ενώ στη συνέχεια ήταν απαραίτητο να εκτυπώσει κάποιος το βιογραφικό του σημείωμα και ίσως κάποια συνοδευτική επιστολή και είτε να τα ταχυδρομήσει στον εργοδότη, ή να τα στείλει με φαξ, ή να τα παραδώσει ο ίδιος αυτοπροσώπως.

Με την εισχώρηση του Διαδικτύου σε κάθε πτυχή της ζωής δεν θα μπορούσε να μείνει ανεπηρέαστος ο τρόπος με τον οποίο γίνεται η αναζήτηση εργασίας. Κάτι τέτοιο δεν αποτελεί έκπληξη, καθώς σήμερα απλά με την δημιουργία ενός προφίλ σε μια ιστοσελίδα όπως το Monster.com ή το Linkedin.com μπορεί κάποιος να έχει πρόσβαση σε χιλιάδες αγγελίες που ανανεώνονται συνεχώς.

Αυτός ο τεράστιος όγκος πληροφορίας δεν σημαίνει ωστόσο ότι κάποιος νυν ή μελλοντικός υποψήφιος ή κάποιος εργοδότης έχει πλήρη εικόνα της προσφοράς και της ζήτησης που υπάρχει στην αγορά εργασίας, όπως το ποιες είναι οι δεξιότητες με την μεγαλύτερη ζήτηση σε μια περιοχή ή χώρα, σε ποιες περιοχές του κόσμου ή μιας χώρας παρουσιάζει κάποιος κλάδος απασχόλησης άνθηση, ποιες δεξιότητες μπορεί να συνδέονται με υψηλότερες απολαβές κ.α. Αυτό είναι και το πρόβλημα το οποίο πραγματεύεται η παρούσα εργασία, μέσω της ανάπτυξης μιας υλοποίησης για την συλλογή δεδομένων από αγγελίες εργασίας και την εξαγωγή πληροφοριών από αυτά.

Σκοπός της εργασίας είναι η μελέτη των διαθέσιμων τεχνολογιών για την εξαγωγή περιεχομένου από ιστοσελίδες (web data extraction), με βασική εστίαση στην τεχνολογία των Web Crawlers, όπως και των προκλήσεων που παρουσιάζει η τεχνολογία

αυτή. Θα επικεντρωθούμε στην ανάκτηση δεδομένων από πηγές HTML (π.χ. ένα Web Document), παρουσιάζοντας μηχανισμούς και εργαλεία που μπορούν να μας βοηθήσουν να περιορίσουμε την ανιαρή δουλειά της ανάκτησης δεδομένων από το Διαδίκτυο.

Όσων αφορά την διάρθρωση της μελέτης, αρχικά αναλύεται το θεωρητικό υπόβαθρο της εργασίας, δηλαδή τι είναι το Web Crawling ενώ παρουσιάζεται και ένας απλός και βασικός αλγόριθμος για crawling. Στη συνέχεια, γίνεται η παρουσίαση ενός βασικού τρόπου εξαγωγής της πληροφορίας (parsing) και των παγίδων και κινδύνων που εμπεριέχει το Web Crawling. Η θεωρία συμπληρώνεται με την ανάλυση της μεθοδολογίας και των εργαλείων που χρησιμοποιήθηκαν για την ανάπτυξη της υλοποίησης. Τέλος, γίνεται η παρουσίαση της εφαρμογής που υλοποιήθηκε κυρίως από την οπτική γωνία του προγραμματιστή με παράθεση και αποσπασμάτων κώδικα, όπου αυτό κρίνεται απαραίτητο, αλλά και από την πλευρά του χρήστη για επίδειξη της λειτουργικότητας.

2 Θεωρητικό Υπόβαθρο

2.1 Τι είναι το Web Crawling και το Web Scraping

Παρότι οι δύο αυτές έννοιες χρησιμοποιούνται συχνά στην βιβλιογραφία για να περιγράψουν το ίδιο πράγμα και πιθανόν στην εξέλιξη της εργασίας να γίνεται το ίδιο, δεν είναι και οι δύο απόλυτα ταυτόσημες. Ως Web Crawlers (μπορεί να τους δούμε και ως bots, robots, spiders) εννοούμε ένα σύστημα για τη μαζική λήψη ιστοσελίδων. Οι Web Crawlers χρησιμοποιούνται για ποικίλους σκοπούς. Πιο συγκεκριμένα, αποτελούν ένα από τα βασικά συστατικά των μηχανών αναζήτησης ιστού, δηλαδή των συστημάτων που συγκεντρώνουν ένα μεγάλο αριθμό ιστοσελίδων, τις καταχωρούν σε ευρετήρια και επιτρέπουν στους χρήστες να εκδίδουν ερωτήματα αναζήτησης σε αυτά και να βρίσκουν τις ιστοσελίδες που ταιριάζουν με τα ερωτήματα τους.

Ο σημαντικότερος λόγος ύπαρξης για τους Web Crawlers έγκειται στο γεγονός ότι ο ιστός δεν είναι ένας κεντρικά διαχειρίσιμος χώρος αποθήκευσης πληροφοριών, αλλά αποτελείται από εκατοντάδες εκατομμύρια ανεξάρτητους παρόχους περιεχομένου ιστού, καθένας από τους οποίους προσφέρει τις δικές του υπηρεσίες και πολλοί ανταγωνίζονται μεταξύ τους. Με άλλα λόγια, ο ιστός μπορεί να θεωρηθεί ως αποκεντρωμένος χώρος αποθήκευσης πληροφοριών, στον οποίο τηρείται ένα σύνολο συμφωνηθέντων πρωτοκόλλων και μορφών δεδομένων, όπως το πρωτόκολλο ελέγχου μετάδοσης (TCP), η υπηρεσία DNS (Domain Name Service), η μεταφορά υπερκειμένου (HTTP), η γλώσσα HTML (Hypertext Markup Language) και το πρωτόκολλο αποκλεισμού ρομπότ (robots.txt).

Ο βασικός αλγόριθμος Web Crawling είναι απλός: Δίνεται στο πρόγραμμα ένα σύνολο διευθύνσεων URL, ένας crawler κατεβάζει όλες τις ιστοσελίδες που βρίσκονται σε αυτές τις διευθύνσεις URL, εξάγει τους υπερσυνδέσμους που περιέχονται στις σελίδες και επαναλαμβάνει την διαδικασία για τις ιστοσελίδες στις οποίες κατευθύνεται μέσω αυτών των υπερσυνδέσμων (Olston C. and Najork M. (2010)).

Σύμφωνα με τον Mitchell R. (2013), ως Web Scraping ορίζεται μια αυτοματοποιημένη διαδικασία που περιλαμβάνει κάποια ανάλυση δεδομένων για να ανακτηθούν μόνο οι πληροφορίες που χρειάζονται. Ένας ακόμα από τους πολλούς ορισμούς αυτής της έννοιας σύμφωνα με τον Castrillo-Fernández (2015), είναι πως ένα εργαλείο Web Scraping είναι μια τεχνολογική λύση για την εξαγωγή δεδομένων από

ιστοτόπους, με γρήγορο, αποτελεσματικό και αυτοματοποιημένο τρόπο, προσφέροντας δεδομένα σε πιο δομημένη και ευκολότερη στη χρήση μορφή.

Μια επιχείρηση μπορεί, για παράδειγμα, να θέλει να ελέγξει αυτόνομα τις τιμές των προϊόντων των ανταγωνιστών της ή ένας φοιτητής μπορεί να θέλει να ενοποιήσει πληροφορίες από ανακοινώσεις διαφόρων πανεπιστημίων από τις ιστοσελίδες τους και να τις παρουσιάσει σε ένα ημερολόγιο στο δικτυακό του τόπο. Εάν ο κάτοχος των πληροφοριών δεν παρέχει ένα ανοιχτό API, η λύση είναι να γραφτεί ένα πρόγραμμα που στοχεύει την ίδια την ιστοσελίδα (Cording P. H. (2011); Martins D. et al. (2015)).

Παρά το γεγονός ότι χρησιμοποιώντας ένα API (συνήθως αποστέλλοντας ένα αίτημα GET σε έναν κεντρικό διακομιστή για να ανακτήσει τα δεδομένα σε μορφή JSON, XML, CSV κ.α.) μπορεί να θεωρηθεί τεχνικά ως Web Scraping, καθώς εν τέλει γίνεται ανάκτηση και ανάλυση δεδομένων, η έννοια αυτή θεωρείται γενικά ότι ισχύει μόνο για ιστοτόπους που έχουν σχεδιαστεί να προβάλλονται από τους ανθρώπους μέσω ενός προγράμματος περιήγησης ιστού και όχι από ένα αυτοματοποιημένο σύστημα. Με την ανάκτηση ολόκληρου του HTML κώδικα από έναν ιστότοπο, την ανάλυση του σε ένα αντικείμενο χρησιμοποιώντας οποιοδήποτε αριθμό διαθέσιμων βιβλιοθηκών και την απομόνωση και επεξεργασία των επιθυμητών δεδομένων, μπορεί κάποιος να είναι απελευθερωμένος από τους περιορισμούς και τις διαθέσιμες δυνατότητες των API που έχουν ήδη αναπτυχθεί για την ανάκτηση των δεδομένων (Mitchell R. (2013)).

Θεωρητικά, το Web Scraping είναι η πρακτική της συλλογής δεδομένων μέσω οποιουδήποτε άλλου μέσου εκτός από ένα πρόγραμμα που αλληλεπιδρά με ένα API ή προφανώς μέσω ενός ανθρώπου που χρησιμοποιεί ένα πρόγραμμα περιήγησης ιστού. Αυτό επιτυγχάνεται συνήθως όταν γίνεται η συγγραφή ενός αυτοματοποιημένου προγράμματος που στέλνει αιτήματα σε έναν διακομιστή ιστού (web server), ζητά δεδομένα (συνήθως με τη μορφή HTML και άλλων αρχείων που συνθέτουν ιστοσελίδες) και στη συνέχεια αναλύει τα δεδομένα για να εξαγάγει τις απαραίτητες πληροφορίες (Mitchell R. (2018); Schmidt M. (2015)).

Η συλλογή δεδομένων με χειροκίνητο τρόπο είναι πραγματικά αναποτελεσματική. Θα πρέπει να γίνει αναζήτηση, αντιγραφή και επικόλληση δεδομένων σε ένα υπολογιστικό φύλλο (π.χ. Excel) για μελλοντική επεξεργασία. Αυτή είναι μια ανιαρή, επαναλαμβανόμενη και κουραστική διαδικασία που παράλληλα είναι και επιρρεπής σε λάθη. Επομένως, είναι πολύ πιο λογικό να αυτοματοποιείται. Η πληροφορία που γίνεται διαθέσιμη στο διαδίκτυο μπορεί να έχει διάφορες μορφές, από

αρχεία PDF μέχρι απλό κείμενο εντός μιας ιστοσελίδας. Ωστόσο, μια ιστοσελίδα έχει μια δομημένη μορφή (κώδικας HTML) και αυτή η δομημένη φύση της με τους κόμβους και τις ιδιότητες τους πολλαπλασιάζει τις δυνατότητες για την εφαρμογή τεχνικών crawling. Έτσι διευκολύνεται το έργο των διάφορων αυτοματοποιημένων εργαλείων (Castrillo-Fernández (2015); Garg A. et al. (2017)).

Η HTML, η γλώσσα σήμανσης που χρησιμοποιείται για τη δομή των δεδομένων σε ιστοσελίδες, προορίζεται για τη δημιουργία μιας οπτικά ελκυστικής διεπαφής για τον άνθρωπο. Το μειονέκτημα των υφιστάμενων τεχνικών που χρησιμοποιούνται για την ανάκτηση δεδομένων μέσω crawling είναι ότι η σήμανση μπορεί να αλλάξει είτε επειδή ο ιστότοπος είναι ιδιαίτερα δυναμικός είτε απλά επειδή ενημερώνεται η εμφάνιση και η αίσθηση τακτικά (Cording P. H. (2011); Martins D. et al. (2015)).

Η αυτοματοποιημένη συλλογή δεδομένων από το διαδίκτυο είναι σχεδόν τόσο παλιά όσο και το ίδιο το Διαδίκτυο. Αν και το Web Scraping δεν είναι ένας νέος όρος, τα τελευταία χρόνια η πρακτική είναι γνωστή και ως screen scraping, data mining, web harvesting ή παρόμοιες παραλλαγές. Η επικρατούσα άποψη σήμερα φαίνεται να ευνοεί τον όρο Web Scraping, οπότε αυτός είναι ο όρος που χρησιμοποιείται συνήθως σε αυτήν την εργασία, αν και μπορεί να αναφερόμαστε σε προγράμματα που διαπερνούν πολλαπλές σελίδες ως Web Crawlers ή στα ίδια τα προγράμματα συλλογής δεδομένων ως bots (Mitchell R. (2018); Schmidt M. (2015)).

Το να πούμε ότι το Web Scraping είναι απλά μια χρήσιμη ικανότητα είναι μια υποτίμηση. Είτε κάποιος ικανοποιεί την περιέργεια του, γράφοντας ένα γρήγορο κομμάτι κώδικα σε ένα απόγευμα είτε κατασκευάζοντας το επόμενο Google, η δυνατότητα να ανακτήσει κάποιος τυχόν online δεδομένα, σε οποιοδήποτε αριθμό, σε οποιαδήποτε μορφή, επιλέγοντας τον τρόπο με τον οποίο θέλει να τα αποθηκεύσει και να τα ανακτήσει, είναι ένα ζωτικό κομμάτι σε κάθε εργαλειοθήκη ενός προγραμματιστή (Mitchell R. (2013)).

Στην πράξη, το Web Scraping περιλαμβάνει μια μεγάλη ποικιλία τεχνικών και τεχνολογιών προγραμματισμού, όπως η ανάλυση δεδομένων, η ανάλυση φυσικής γλώσσας (NLP – Natural Language Processing) και η ασφάλεια των πληροφοριών. Επειδή το εύρος του πεδίου είναι τόσο μεγάλο, η εργασία αυτή καλύπτει τα θεμελιώδη και βασικά στοιχεία του Web Scraping και του Web Crawling.

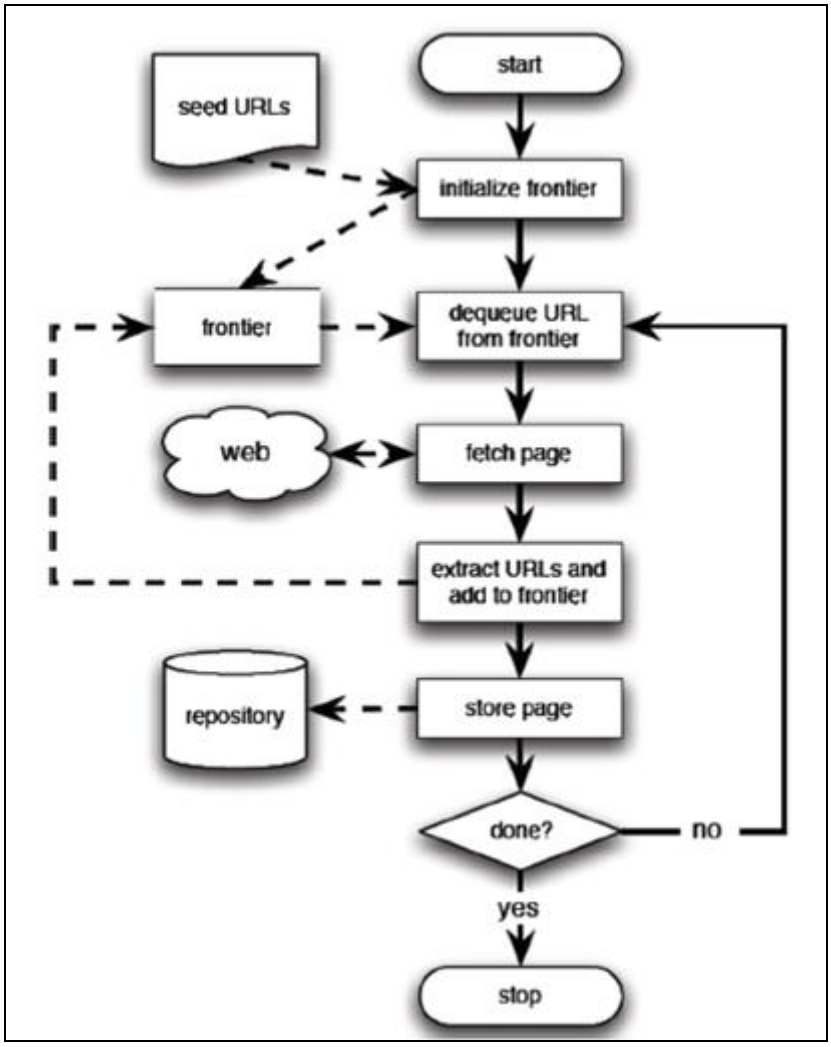
2.2 Ένας βασικός αλγόριθμος για crawling

Στην απλούστερη μορφή του, ένας Web Crawler ξεκινά από ένα σύνολο σελίδων σπόρου (seed URLs) και στη συνέχεια χρησιμοποιεί τους συνδέσμους μέσα σε αυτές για να ανακτήσει άλλες σελίδες. Οι σύνδεσμοι σε αυτές τις σελίδες εξάγονται με τη σειρά τους και επισκέπτονται οι αντίστοιχες σελίδες και ούτω καθεξής. Η διαδικασία επαναλαμβάνεται μέχρι να επισκεφθεί κάποιος επαρκής αριθμός σελίδων ή να επιτευχθεί κάποιος άλλος στόχος. Αυτή η απλή περιγραφή αποκρύπτει πολλά λεπτά θέματα που σχετίζονται με τις συνδέσεις δικτύου, τις παγίδες στις οποίες μπορεί να πέσει ο crawler, την κανονικοποίηση των διευθύνσεων URL, την γραμματική και συντακτική ανάλυση κάθε σελίδας και τη δεοντολογία περί ανίχνευσης. Στην πραγματικότητα, οι ιδρυτές της Google, Sergey Brin και Lawrence Page, προσδιόρισαν τον ανιχνευτή ιστού ως το πιο εξελιγμένο αλλά εύθραυστο στοιχείο μιας μηχανής αναζήτησης (Liu B. (2011); Garg A. et al. (2017)).

Η Εικόνα 2-1 δείχνει τη ροή ενός βασικού ακολουθιακού crawler. Ένα τέτοιο πρόγραμμα ανίχνευσης ανακτά μια σελίδα τη φορά και διατηρεί μια λίστα μη διευθετημένων διευθύνσεων URL που ονομάζεται frontier. Η λίστα αρχικοποιείται με διευθύνσεις URL σπόρων που μπορούν να παρέχονται από το χρήστη ή άλλο πρόγραμμα. Σε κάθε επανάληψη του κύριου βρόχου του, ο crawler επιλέγει την επόμενη διεύθυνση URL από το frontier, ανακτά τη σελίδα που αντιστοιχεί στη διεύθυνση URL μέσω HTTP, αναλύει την ανακτημένη σελίδα για να εξαγάγει τις διευθύνσεις URL, προσθέτει τα νέα URL που βρέθηκαν στο frontier και αποθηκεύει τη σελίδα ή άλλες εξαγόμενες πληροφορίες σε ένα τοπικό χώρο αποθήκευσης (Shang J. et al. (2016); Agre G.H. and Mahajan N.V. (2015)).

Η διαδικασία ανίχνευσης μπορεί να τερματιστεί όταν κάποιος αριθμός σελίδων έχει ανιχνευθεί. Η ανίχνευση μπορεί επίσης να αναγκαστεί να σταματήσει εάν το frontier αδειάσει, παρόλο που αυτό συμβαίνει σπάνια στην πράξη λόγω του υψηλού μέσου αριθμού συνδέσμων (υπολογίζεται ότι υπάρχουν κατά μέσο όρο τουλάχιστον δέκα υπερσυνδέσεις ανά σελίδα στον ιστό). Ένας crawler είναι ουσιαστικά ένας αλγόριθμος αναζήτησης γραφήματος (graph search algorithm). Ο ιστός μπορεί να θεωρηθεί ως ένα μεγάλο γράφημα με σελίδες ως κόμβους και υπερσυνδέσμους ως τις άκρες του. Ένα πρόγραμμα ανίχνευσης ξεκινά από μερικούς από τους κόμβους (σπόρους) και ακολουθεί τις άκρες για να φτάσει σε άλλους κόμβους. Η διαδικασία της ανάκτησης μιας σελίδας

και της εξαγωγής των συνδέσμων μέσα σε αυτήν είναι ανάλογη με την επέκταση ενός κόμβου (node expanding) στην αναζήτηση γραφημάτων (Liu B. (2011); Desai K. et al. (2017); Peng D. et al. (2018)).



Εικόνα 2-1: Διάγραμμα ροής ενός βασικού ακολουθιακού crawler (Liu B. (2011))

2.3 Θέματα υλοποίησης

2.3.1 Ανάκτηση (fetching)

Για την ανάκτηση σελίδων, ο crawler λειτουργεί ως Web client. Στέλνει ένα αίτημα HTTP στο διακομιστή που φιλοξενεί τη σελίδα και διαβάζει την απάντηση. Ο

πελάτης χρειάζεται να έχει κάποιο είδος timeout ώστε να αποφευχθεί η αχρείαστη δαπάνη χρόνου αναμένοντας για απαντήσεις από αργούς διακομιστές ή για την ανάγνωση σελίδων με μεγάλο όγκο. Ο πελάτης αναλύει τις κεφαλίδες απόκρισης για τους κωδικούς κατάστασης και πιθανές ανακατευθύνσεις. Επιθυμητό είναι να ανιχνεύονται πιθανοί βρόχοι επαναπροσανατολισμού έτσι ώστε να αποφεύγεται η ανάκτηση της ίδιας σελίδας δύο και πλέον φορές στην ίδια εκτέλεση.

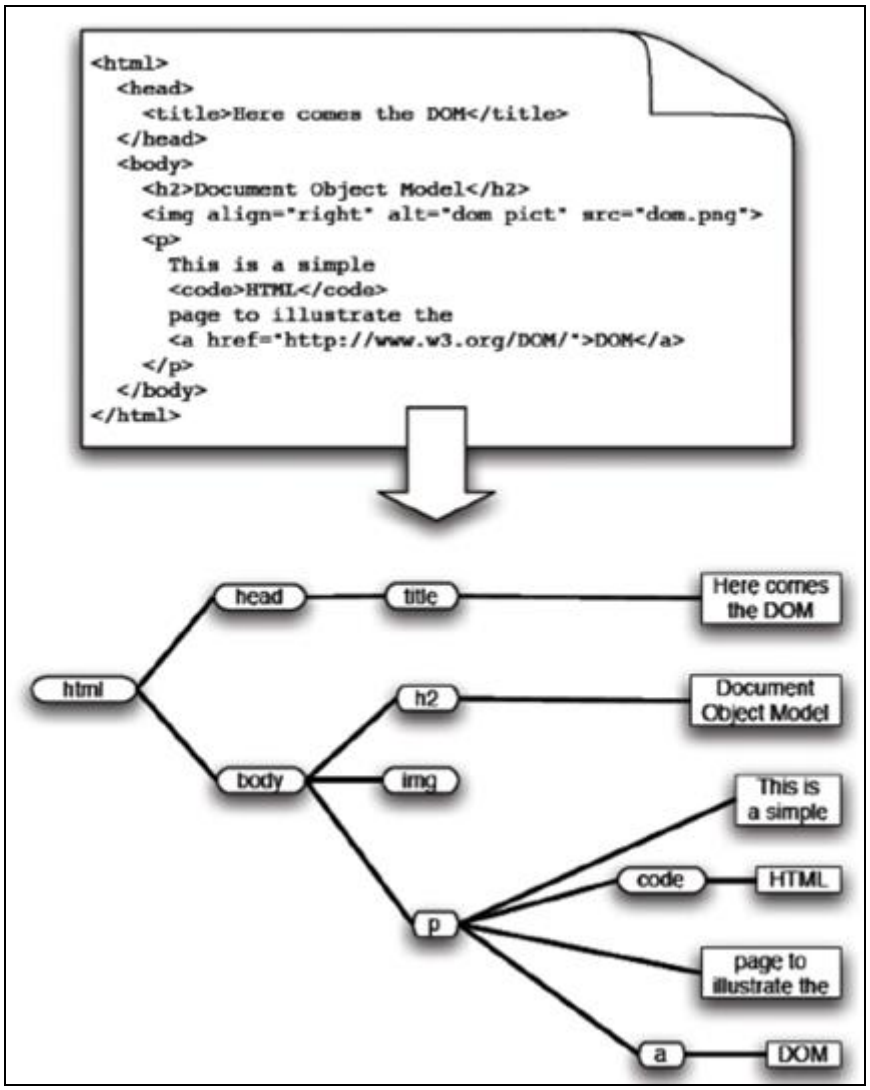
Μπορεί κανείς να αναλύσει και να αποθηκεύσει την τελευταία τροποποιημένη κεφαλίδα για να καθορίσει την ηλικία του εγγράφου, αν και αυτές οι πληροφορίες είναι γνωστό ότι είναι αναξιόπιστες. Ο έλεγχος σφαλμάτων και ο χειρισμός εξαιρέσεων είναι σημαντικοί κατά τη διαδικασία λήψης της σελίδας, καθώς ο ίδιος κώδικας πρέπει να ασχολείται με δυνητικά εκατομμύρια απομακρυσμένους διακομιστές. Επιπλέον, μπορεί να είναι ωφέλιμη η συλλογή στατιστικών στοιχείων σχετικά με τα χρονικά όρια και τους κωδικούς κατάστασης για τον εντοπισμό προβλημάτων ή την αυτόματη προσαρμογή των τιμών timeout. Γλώσσες προγραμματισμού όπως Java, Python και Perl παρέχουν απλές προγραμματικές διεπαφές για την ανάκτηση σελίδων από το Web. Ωστόσο, πρέπει κάποιος να είναι προσεκτικός στη χρήση διασυνδέσεων υψηλού επιπέδου όπου μπορεί να είναι πιο δύσκολο να ανιχνευθούν προβλήματα χαμηλότερου επιπέδου, όπως ο σωστός έλεγχος των χρονικών ορίων σύνδεσης (Liu B. (2011)).

2.3.2 Ανάλυση (parsing)

Μόλις ληφθεί μια σελίδα, ο crawler αναλύει το περιεχόμενό της, δηλαδή το ωφέλιμο φορτίο HTTP και εξάγει πληροφορίες τόσο για να υποστηρίξει την κύρια εφαρμογή του crawler (π.χ., ευρετηρίαση της σελίδας αν ο crawler υποστηρίζει μια μηχανή αναζήτησης) όσο και για να μπορέσει να συνεχίσει να τρέχει (εξάγοντας συνδέσμους που θα προστεθούν στο frontier). Η ανάλυση μπορεί να συνεπάγεται απλή εξαγωγή διεύθυνσης URL από υπερσυνδέσμους ή βαθύτερη ανάλυση του κώδικα HTML. Το Μοντέλο Αντικειμένου Εγγράφου (Document Object Model - DOM) καθιερώνει τη δομή μιας σελίδας HTML ως δέντρο κόμβων, όπως απεικονίζεται στην Εικόνα 2-2 (Liu B. (2011)).

Με βάση αυτήν την δομή ένα έγγραφο HTML μπορεί να αναλυθεί σε ένα δένδρο από αντικείμενα. Τα αντικείμενα αυτά διακρίνονται σε Elements και TextNodes. Αυτή η

δενδροειδής μορφή δίνει την δυνατότητα να γίνει η προσπέλαση των δεδομένων με τη χρήση κάποιου διαθέσιμου parser, όπως είναι το Jsoup, ή με την υποβολή ερωτημάτων ως προς την δομή της HTML, όπως είναι η χρήση της Xpath (Martins D. et al. (2015)).



Εικόνα 2-2: Αναπαράσταση της δενδροειδούς δομής DOM μιας απλής σελίδας HTML. Οι εσωτερικοί κόμβοι (σε οβάλ) αναπαριστούν τα HTML tags, με το tag <html> ως ρίζα. Οι κόμβοι φύλλα (σε παραλληλόγραμμα) αντιστοιχούν σε κομμάτια κειμένου. (Liu B. (2011))

Σε αντίθεση με τον κώδικα του προγράμματος, ο οποίος πρέπει να μεταγλωττίσει σωστά ή αλλιώς θα αποτύχει με ένα σφάλμα σύνταξης, η ορθότητα του κώδικα HTML τείνει να παραβλέπεται σε ένα βαθμό από τα προγράμματα περιήγησης. Ακόμη και όταν τα πρότυπα HTML απαιτούν αυστηρή ερμηνεία, τα de facto πρότυπα που επιβάλλονται

από τις εφαρμογές του προγράμματος περιήγησης είναι πολύ επιεική. Αυτό, μαζί με τον τεράστιο πληθυσμό των μη εξειδικευμένων δημιουργών που κατασκευάζουν ιστοσελίδες, προσθέτει μια σημαντική πολυπλοκότητα σε έναν HTML parser ενός crawler.

Πολλές σελίδες δημοσιεύονται χωρίς τις απαιτούμενες ετικέτες, ετικέτες που είναι λανθασμένα εμφωλευμένες, ετικέτες που δεν είναι κλειστές, ονόματα και τιμές ιδιοτήτων με ορθογραφικά λάθη ή ελλείψεις, ελλιπή εισαγωγικά γύρω από τιμές χαρακτηριστικών (attributes), unescaped ειδικοί χαρακτήρες και ούτω καθεξής. Για παράδειγμα, ο χαρακτήρας διπλών εισαγωγικών στην HTML είναι δεσμευμένος για τη σύνταξη ετικέτας και επομένως απαγορεύεται να χρησιμοποιηθεί μέσα στο κείμενο. Η ειδική οντότητα HTML '"' πρέπει να χρησιμοποιηθεί στη θέση του. Ωστόσο, μόνο ένας μικρός αριθμός συγγραφέων το γνωρίζουν και ένα μεγάλο μέρος των ιστοσελίδων περιέχει αυτόν τον παράνομο χαρακτήρα. Ακριβώς όπως τα προγράμματα περιήγησης, οι crawlers πρέπει να είναι ανεκτικοί σε αυτές τις περιπτώσεις. Δεν έχουν την πολυτέλεια να απορρίψουν πολλές σημαντικές σελίδες, όπως θα έκανε ένας αυστηρός parser.

Επίσης, ένα αυξανόμενο τμήμα ιστοσελίδων είναι γραμμένο σε μορφές διαφορετικές από HTML. Οι ανιχνευτές που υποστηρίζουν μηχανές αναζήτησης μεγάλης κλίμακας αναζητούν συστηματικά και ευρετηριακά έγγραφα σε πολλές ανοικτές και ιδιόκτητες μορφές όπως το απλό κείμενο, το PDF, το Microsoft Word και το Microsoft PowerPoint. Ανάλογα με την εφαρμογή της μηχανής ανίχνευσης, αυτό μπορεί να απαιτηθεί ή όχι. Ορισμένες μορφές παρουσιάζουν ιδιαίτερες δυσκολίες καθώς είναι γραμμένες αποκλειστικά για αλληλεπίδραση ανθρώπων και επομένως είναι ιδιαίτερα εχθρικές στους crawlers. Για παράδειγμα, μερικοί εμπορικοί ιστότοποι χρησιμοποιούν γραφικά animations σε Flash. Αυτά είναι πολύ δύσκολο για έναν crawler να τα αναλύσει για να αποσπάσει συνδέσμους και το κειμενικό περιεχόμενο τους. Άλλα παραδείγματα περιλαμβάνουν τις σελίδες που κάνουν βαριά χρήση της Javascript για αλληλεπίδραση . Νέες προκλήσεις πρόκειται να προστεθούν καθώς τα νέα πρότυπα όπως το Scalable Vector Graphics (SVG), το Asynchronous Javascript, το XML (AJAX) και άλλες γλώσσες που βασίζονται σε XML κερδίζουν δημοτικότητα. Στην παρούσα εργασία ωστόσο θα ασχοληθούμε μόνο με ανάκτηση από σελίδες HTML (Liu B. (2011); Martins D. et al. (2015); Van Deursen A. et al. (2015)).

2.3.3 Προ-επεξεργασία κειμένου

Κατά το parsing μιας ιστοσελίδας για την εξαγωγή του περιεχομένου ή την ανάκτηση νέων διευθύνσεων URL από αυτήν, είναι συχνά χρήσιμο να καταργηθούν οι αποκαλούμενες stopwords, δηλαδή όροι όπως άρθρα και συζεύξεις, που είναι τόσο συνηθισμένοι ώστε να παρεμποδίζουν τη διάκριση σελίδων με βάση το περιεχόμενο τους. Μια άλλη χρήσιμη τεχνική είναι αυτή του Stemming, με την οποία οι μορφολογικές παραλλαγές των όρων συγχωνεύονται σε κοινές ρίζες (stems). Και οι δύο αυτές τεχνικές είναι τυπικές και ευρέως χρησιμοποιούμενες για την ανάκτηση πληροφοριών. Για ιστοσελίδες, επιπρόσθετες εργασίες, όπως η αφαίρεση ετικετών HTML και η αναγνώριση των κύριων μπλοκ περιεχομένου θα πρέπει επίσης να ληφθούν προσεκτικά υπόψιν (Liu B. (2011); Flores F.N. and Moreira V. P. (2016)).

Ως stopwords αναφέρονται οι συχνά εμφανιζόμενες και ασήμαντες λέξεις σε μια γλώσσα που συμβάλλουν στην κατασκευή ορθών συντακτικά προτάσεων αλλά δεν προσθέτουν κανένα περιεχόμενο σε ένα έγγραφο. Τα άρθρα, οι προθέσεις και οι σύνδεσμοι, όπως επίσης και ορισμένες αντωνυμίες ανήκουν σε αυτή την κατηγορία. Κάποιες κοινές τέτοιες λέξεις στα αγγλικά είναι: a, about, an, are, as, at, be, by, for, from, how, in, is, of, on, or, that, the, these, this, to, was, what, when, where, who, will, with. Τέτοιες λέξεις θα πρέπει να αφαιρεθούν πριν τα έγγραφα ευρετηριαστούν και αποθηκευτούν.

Οι stopwords αφαιρούνται εξ ολοκλήρου από το κείμενο για να επικεντρωθούμε στις πιο σημαντικές λέξεις. Για παράδειγμα, υπάρχει μια φράση “how to create database”. Περιέχει δύο stopwords: “how” και “to.” Εάν η μηχανή αναζήτησης προσπαθεί να χρησιμοποιήσει “how”, “to”, “create” και “database.” για να βρει τις επιθυμητές σελίδες HTML, μπορεί να βρει περισσότερες σελίδες HTML που περιέχουν “how” και “to” από αυτές που περιέχουν “create” και “database”. Για να καταργήσουμε τις stopwords που βασίζονται στην αγγλική γραμματική, χρησιμοποιούμε μία από τις υπάρχουσες λίστες διαθέσιμες στο διαδίκτυο (Zhang S. (2017)).

Σε πολλές γλώσσες, μια λέξη έχει διάφορες συντακτικές μορφές ανάλογα με το πλαίσιο μέσα στο οποίο χρησιμοποιείται. Για παράδειγμα, στα αγγλικά, τα ουσιαστικά έχουν πολυμορφικές μορφές, τα ρήματα έχουν μορφές γερουνδίου (με την προσθήκη “ing”), και τα ρήματα που χρησιμοποιήθηκαν στον παρελθόντα χρόνο είναι διαφορετικά από τον παρόντα χρόνο. Αυτές θεωρούνται συντακτικές παραλλαγές της ίδιας ριζικής

μορφής. Τέτοιες παραλλαγές προκαλούν χαμηλή ανάκληση για ένα σύστημα ανάκτησης, επειδή ένα έγγραφο μπορεί να περιέχει μια παραλλαγή μιας λέξης, αλλά όχι την ίδια ακριβώς λέξη. Αυτό το πρόβλημα μπορεί να αντιμετωπιστεί εν μέρει από το Stemming (Flores F.N. and Moreira V. P. (2016)).

Το Stemming αναφέρεται στη διαδικασία μείωσης των λέξεων στις ρίζες τους. Ως stem ορίζεται το τμήμα μιας λέξης που μένει μετά την αφαίρεση των προθεμάτων και των επιθεμάτων. Στα αγγλικά, οι περισσότερες παραλλαγές μιας λέξης δημιουργούνται με την εισαγωγή επιθεμάτων (και όχι προθεμάτων). Έτσι, Stemming στα αγγλικά συνήθως σημαίνει αφαίρεση του επιθέματος. Για παράδειγμα οι λέξεις "computer", "computing" και "compute" μπορούν να ελαχιστοποιηθούν σε "comput". Αντίστοιχα οι λέξεις "walks", "walking" και "walker" σε "walk". Το Stemming επιτρέπει να λαμβάνονται υπόψη διάφορες παραλλαγές της λέξης κατά την ανάκτηση, γεγονός που βελτιώνει την ποιότητα της προς ανάκτηση πληροφορίας (Rajput B.S. and Khare N. (2015)).

Με την πάροδο των ετών, πολλοί ερευνητές αξιολόγησαν τα πλεονεκτήματα και τα μειονεκτήματα της χρήσης του Stemming. Σαφώς, η τεχνική αυτή αυξάνει τον αριθμό της ουσιαστικής πληροφορίας εντός του κειμένου και μειώνει το μέγεθος του σε πιο διαχειρίσιμα μεγέθη. Εντούτοις, μπορεί να βλάψει την ακρίβεια της πληροφορίας, διότι πολλά άσχετα έγγραφα μπορεί να θεωρηθούν σχετικά. Για παράδειγμα, τόσο η λέξη "cop" όσο και η λέξη "cope" μειώνονται στην λέξη στέλεχος "cop". Ωστόσο, αν κάποιος ψάχνει για έγγραφα σχετικά με την αστυνομία, ένα έγγραφο που περιέχει μόνο την λέξη "cope" είναι απίθανο να είναι σχετικό. Παρόλο που πολλά πειράματα διεξήχθησαν από ερευνητές, δεν υπάρχουν ακόμη τρανταχτά αποδεικτικά στοιχεία υπέρ ή κατά της χρήσης αυτής της τεχνικής. Στην πράξη, θα πρέπει να πειραματιστεί ένας προγραμματιστής κατά την επεξεργασία των δεδομένων που έχει στη διάθεσή του για να αποφασίσει αν μια τέτοια υλοποίηση τον βοηθάει ή όχι. Στην παρούσα εργασία αποφασίστηκε να μην γίνει χρήση του Stemming (Liu B. (2011)).

Υπάρχουν όμως και επιπλέον προπαρασκευαστικές εργασίες που πρέπει να πραγματοποιηθούν. Οι αριθμοί και οι όροι που περιέχουν ψηφία καταργούνται στα παραδοσιακά συστήματα ανάκτησης δεδομένων εκτός από ορισμένους συγκεκριμένους τύπους, π.χ. ημερομηνίες, ώρες και άλλους προκαθορισμένους τύπους. Επίσης, συνήθως όλα τα γράμματα μετατρέπονται είτε σε κεφαλαία είτε σε πεζά (Zhang S. (2017)).

Επιπροσθέτως, η αφαίρεση του χαρακτήρα hyphen (δηλαδή ο χαρακτήρας ένωσης '-') πραγματοποιείται συνήθως για την αντιμετώπιση της ασυνέπειας της χρήσης. Για παράδειγμα, μερικοί άνθρωποι θα χρησιμοποιήσουν την έκφραση "state-of-the-art", ενώ άλλοι την έκφραση "state of the art". Αν οι παύλες στην πρώτη περίπτωση αφαιρεθούν, καταργείται το πρόβλημα ασυνέπειας. Ωστόσο, ορισμένες λέξεις μπορεί να έχουν μια παύλα ως αναπόσπαστο τμήμα της λέξης, π.χ. "Y-21". Έτσι, γενικά, το σύστημα μπορεί να ακολουθήσει έναν γενικό κανόνα (π.χ. να αφαιρέσει όλες τις παύλες) και επίσης να έχει κάποιες εξαιρέσεις. Να σημειωθεί ότι υπάρχουν δύο τύποι αφαίρεσης, δηλαδή πρώτον κάθε παύλα αντικαθίσταται με ένα κενό και δεύτερον κάθε παύλα απλά αφαιρείται χωρίς να αφήνεται κενό έτσι ώστε η έκφραση "state-of-the-art" να μπορεί να αντικατασταθεί είτε από την "state of the art" ή την "stateoftheart". Σε ορισμένα συστήματα, και οι δύο μορφές μπορεί να αποθηκεύονται κατά περίπτωση καθώς είναι δύσκολο να προσδιοριστεί το σωστό, π.χ. εάν η λέξη "pre-processing" μετατραπεί σε "pre processing", τότε σε ορισμένες αναζητήσεις η σχετική πληροφορία δεν θα βρεθεί εάν ο όρος του ερωτήματος αναζήτησης είναι "preprocessing". Κάτι παρόμοιο ισχύει και για τα σημεία στίξης που μπορούν να αντιμετωπιστούν με την ίδια λογική που αντιμετωπίζονται και οι παύλες.

Όπως είπαμε στην αρχή της ενότητας οι ιστοσελίδες διαφέρουν από τα παραδοσιακά έγγραφα κειμένου. Επομένως, χρειάζεται πρόσθετη προεπεξεργασία. Η αφαίρεση των ετικετών HTML μπορεί να αντιμετωπιστεί με τρόπο παρόμοιο με αυτόν των σημείων στίξης. Παρόλα αυτά χρειάζεται προσεκτική εξέταση, για το πώς μπορεί αυτό να επηρεάζει τα δεδομένα προς ανάκτηση. Η HTML είναι εγγενώς μια γλώσσα οπτικής παρουσίασης. Σε μια τυπική εμπορική σελίδα, οι πληροφορίες παρουσιάζονται σε πολλά ορθογώνια μπλοκ. Η απλή κατάργηση των ετικετών HTML μπορεί να προκαλέσει προβλήματα συνδέοντας κείμενο που δεν πρέπει να ενωθεί. Αυτό το πρόβλημα δεν αντιμετωπίστηκε ικανοποιητικά από τις μηχανές αναζήτησης τη στιγμή που γράφτηκε αυτή η εργασία.

Επόμενο βήμα είναι ο εντοπισμός των κύριων μπλοκ περιεχομένου. Μια τυπική ιστοσελίδα, ειδικά μια εμπορική σελίδα, περιέχει μια μεγάλη ποσότητα πληροφοριών που δεν αποτελούν μέρος του κύριου περιεχομένου της σελίδας. Για παράδειγμα, ενδέχεται να περιέχει διαφημίσεις banner, γραμμές πλοήγησης, ειδοποιήσεις πνευματικών δικαιωμάτων κ.λπ., οι οποίες μπορεί να οδηγήσουν σε ανεπαρκή αποτελέσματα εξόρυξης δεδομένων. Για το λόγο αυτό χρησιμοποιούνται μόνο τα κύρια

μπλοκ περιεχομένου για να βελτιωθούν τα αποτελέσματα εξόρυξης δεδομένων (Liu B. (2011)).

2.4 Ηθικοί φραγμοί και νομιμότητα

Οι εφαρμογές Web Scraping έχουν την φήμη πως βρίσκονται σε μια γκρίζα περιοχή. Όπως πολλοί τομείς της τεχνολογίας, το νομικό προηγούμενο για το Web Scraping είναι περιορισμένο. Ένας καλός κανόνας για να παραμείνουμε εκτός προβλήματος είναι να ακολουθούμε πάντα τους όρους χρήσης και τα έγγραφα πνευματικής ιδιοκτησίας σε ιστοτόπους που χρησιμοποιούμε (αν υπάρχουν) (Mitchell R. (2013)).

Υπάρχουν ορισμένες περιπτώσεις στις οποίες η πράξη του Web Crawling είναι η ίδια σε σκοτεινό νομικό έδαφος, ανεξάρτητα από τον τρόπο με τον οποίο χρησιμοποιούνται τα δεδομένα. Το crawling συχνά απαγορεύεται στους όρους υπηρεσίας (Terms of Service) των ιστοτόπων όπου τα συγκεντρωτικά δεδομένα είναι πολύτιμα (για παράδειγμα, ένας ιστότοπος που περιέχει έναν κατάλογο προσωπικών διευθύνσεων στις Ηνωμένες Πολιτείες) ή όπου είναι διαθέσιμο ένα εμπορικό API. Το Twitter, για παράδειγμα, απαγορεύει ρητά το crawling (τουλάχιστον από οποιαδήποτε δεδομένα από τα tweet) στους όρους υπηρεσίας του: *"crawling the Services is permissible if done in accordance with the provisions of the robots.txt file, however, scraping the Services without the prior consent of Twitter is expressly prohibited"*.

Το ίδιο ισχύει και για τις περισσότερες σελίδες αναζήτησης εργασίας. Για παράδειγμα εντός των όρων χρήσης της ιστοσελίδας monster.com "*All Monster Users agree to not: (d) use any data mining, robots or similar data gathering or extraction methods*".

Κάτι αντίστοιχο ισχύει και για το indeed.com "*Unless you have been specifically permitted to do so in a separate, written agreement with Indeed, you agree that you will not crawl, scrape, reproduce, duplicate, copy, sell, trade or resell the Site for any purpose*".

Ο λόγος για αυτό είναι προφανής. Οι crawlers, ειδικά όταν είναι αποτελεσματικοί, μπορούν να ασκήσουν σημαντική πίεση στους πόρους των διακομιστών Web, κυρίως στο bandwidth του δικτύου τους. Ένας crawler που στέλνει πολλά αιτήματα σελίδων σε έναν διακομιστή με γρήγορη διαδοχή, π.χ. δέκα ή

περισσότερα ανά δευτερόλεπτο, θεωρείται μη ευγενικός. Ο λόγος είναι ότι ο διακομιστής θα ήταν τόσο απασχολημένος να απαντήσει στο πρόγραμμα ανίχνευσης που η υπηρεσία του σε άλλες αιτήσεις, συμπεριλαμβανομένων εκείνων που προέρχονται από την ανθρώπινη περιήγηση, θα επιδεινωθεί. Στην ακραία περίπτωση, ένας διακομιστής που πλημμυρίστηκε με αιτήματα από έναν επιθετικό crawler δεν θα μπορούσε να ανταποκριθεί σε άλλα αιτήματα, που έχει ως συνέπεια μια αποτελεσματική επίθεση άρνησης εξυπηρέτησης (denial of service) από τον crawler.

Για να αποφευχθούν τέτοιου είδους συμβάντα, είναι απαραίτητο ένας crawler να θέσει σε εφαρμογή μέτρα για τη διανομή των αιτημάτων του σε περισσότερο από κάποιο εύλογα καθορισμένο μέγιστο ρυθμό (ένα αίτημα κάθε λίγα δευτερόλεπτα). Αυτή η πρακτική όχι μόνο απαιτείται από ευγένεια προς τους εξυπηρετητές, αλλά έχει και τα πρόσθετα οφέλη από τον περιορισμό της επίδρασης των παγίδων των crawlers (περισσότερα για αυτό στο επόμενο κεφάλαιο) και από την μη υπερφόρτωση του διακομιστή, ο οποίος θα αρχίσει να ανταποκρίνεται αργά. Η αποτροπή της υπερφόρτωσης του διακομιστή είναι μόνο μία από τις πολιτικές που απαιτούνται από τους ηθικούς Web Crawlers (Liu B. (2011)).

Αυτές οι πολιτικές συχνά αναφέρονται συλλογικά ως crawler etiquette. Στα πλαίσια αυτών των πολιτικών ανήκει και η συμμόρφωση με το πρωτόκολλο αποκλεισμού ρομπότ. Πρόκειται για ένα de facto πρότυπο που παρέχει έναν τρόπο για τους διαχειριστές του διακομιστή Web να επικοινωνούν, ποια αρχεία ενδέχεται να μην είναι προσβάσιμα από έναν crawler. Αυτό επιτυγχάνεται μέσω ενός προαιρετικού αρχείου με το όνομα robots.txt στον root κατάλογο του διακομιστή Web, π.χ. http://www.somehost.com/robots.txt (Desai K. et al (2017)). Το αρχείο παρέχει πολιτικές πρόσβασης για διαφορετικούς crawlers, που προσδιορίζονται από το πεδίο κεφαλίδας User-agent. Για κάθε τιμή User-agent (ή το προεπιλεγμένο "*"), ένας αριθμός καταχωρίσεων Disallow προσδιορίζει τις δευτερεύουσες γραμμές καταλόγου που πρέπει να αποφεύγονται. Οι crawlers που θέλουν να ακολουθούνε αυτή την καθοδήγηση πρέπει να παραλάβουν και να αναλύσουν το αρχείο robots.txt ενός διακομιστή πριν στείλουν αιτήματα σε αυτόν. Ένα παράδειγμα μιας πολιτικής που δηλώνεται στο robots.txt δίνεται παρακάτω:

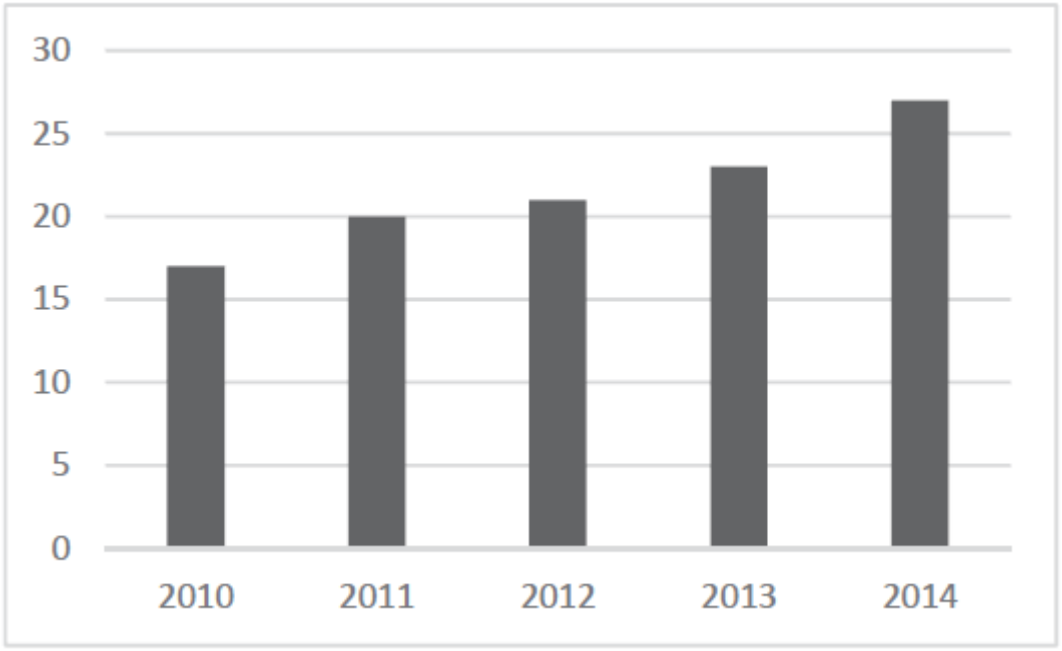
```
User-agent: *
Disallow: /
```

Η παραπάνω δήλωση καθοδηγεί κάθε crawler να παραμείνει μακριά από ολόκληρο το διακομιστή. Περισσότερες λεπτομέρειες σχετικά με τα πρωτόκολλα αποκλεισμού ρομπότ μπορούν να βρεθούν στο http://www.robotstxt.org/wc/robots.html.

Όταν συζητάμε για τις αλληλεπιδράσεις μεταξύ παρόχων πληροφοριών και μηχανών αναζήτησης ή άλλων εφαρμογών που βασίζονται σε crawlers ιστού, δημιουργείται σύγχυση μεταξύ των δεοντολογικών, τεχνικών και νομικών συνεπειών του πρωτοκόλλου αποκλεισμού ρομπότ. Η συμμόρφωση με το πρωτόκολλο είναι μεν ένα ζήτημα δεοντολογίας, είναι όμως προαιρετική και ένα αρχείο robots.txt δεν μπορεί να την επιβάλει (Liu B. (2011)). Εκτός εάν απαγορεύεται ρητά από τους όρους παροχής υπηρεσιών, δεν υπάρχει ουσιώδης διαφορά μεταξύ της πρόσβασης σε έναν ιστότοπο (και των πληροφοριών του) μέσω ενός προγράμματος περιήγησης και την πρόσβαση σε αυτόν μέσω αυτοματοποιημένου προγράμματος. Το αρχείο robots.txt μόνο του δεν έχει αποδειχθεί ότι είναι νομικά δεσμευτικό, αν και σε πολλές περιπτώσεις οι όροι υπηρεσίας μπορούν να είναι (Mitchell R. (2013)).

2.5 Παγίδες και πώς αποφεύγονται

Όπως αναφέρθηκε ήδη, οι Web Crawlers απαιτούν σημαντικό εύρος ζώνης και μπορεί να ασκήσουν υψηλές πιέσεις σε έναν διακομιστή. Επιπλέον, παρά την ύπαρξη του πρωτόκολλου αποκλεισμού ρομπότ (robots.txt), για να δίνονται οδηγίες σχετικά με την ανίχνευση ιστοτόπου σε Web Crawlers από τους ιδιοκτήτες ιστοτόπων, περίπου μόνο το ένα τρίτο των πηγών ιστού χρησιμοποιεί αυτό το πρότυπο για να ρυθμίσει τις δραστηριότητες ανίχνευσης, ενώ δεν συμβιβάζονται όλα οι Web Crawlers με το πρότυπο. Κατά τον Murphy I. (2016) πλέον πάνω από το 46% της κίνησης στο διαδίκτυο είναι bots που κατεβάζουν δεδομένα από ιστοτόπους. Για το λόγο αυτό, οι περισσότεροι ιστότοποι προσπαθούν να εφαρμόσουν τεχνολογίες ανίχνευσης robots και διάκρισης ανάμεσα σε browsers και crawlers. Σε αυτούς του τρόπους αντιμετώπισης του crawling και στους τρόπους αποφυγής τους βάσει της βιβλιογραφίας θα επικεντρωθούμε σε αυτό το κεφάλαιο.



Εικόνα 2-3: Μέσος όρος επισκεψιμότητας από crawlers ανά ιστότοπο, %. (Menshchikov A. et al (2017))

Σίγουρα είναι πολύ απογοητευτικό να μελετάμε το output ενός Web Scraper που με κόπο δημιουργήσαμε και τρέξαμε και να διαπιστώνουμε την απουσία των δεδομένων που είναι τόσο εμφανή στο πρόγραμμα περιήγησης, ή να αποκλειστεί η διεύθυνση IP μας από έναν ιστότοπο για άγνωστους λόγους. Αυτά είναι μερικά από τα πιο δύσκολα προβλήματα για την επίλυση, όχι μόνο επειδή μπορούν να είναι τόσο απροσδόκητα, αλλά επειδή δεν έχουν οποιαδήποτε μηνύματα σφάλματος ή stack traces που θα μπορούσαν να χρησιμοποιηθούν. Το πρόγραμμα έχει αναγνωριστεί ως bot, απορρίφθηκε και δεν ξέρουμε γιατί.

Το πρώτο που κοιτάει ένας διακομιστής όταν δέχεται ένα αίτημα είναι οι κεφαλίδες HTTP. Οι κεφαλίδες HTTP είναι λίστες ιδιοτήτων ή προτιμήσεων που αποστέλλονται κάθε φορά που υποβάλλεται ένα αίτημα σε έναν διακομιστή ιστού. Το HTTP ορίζει δεκάδες τύπους κεκαλυμμένων κεφαλίδων, οι περισσότεροι από τους οποίους δεν χρησιμοποιούνται συνήθως. Τα παρακάτω επτά πεδία, ωστόσο, χρησιμοποιούνται από τα περισσότερα δημοφιλή προγράμματα περιήγησης κατά την εκκίνηση οποιασδήποτε σύνδεσης (Mitchell R. (2018)).

Πίνακας 2-1: Πεδία Http header (Mitchell R. (2018))

Host	**https://www.google.com/**
Connection	keep-alive
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.95 Safari/537.36
Referrer	https://www.google.com/
Accept-Encoding	gzip, deflate, sdch
Accept-Language	en-US,en;q=0.8

Το βασικότερο από τα παραπάνω πεδία είναι αυτό του User-Agent. Όταν ένας διακομιστής λαμβάνει ένα αίτημα χωρίς αυτό το πεδίο τότε μπορεί εύκολα να συμπεράνει πως δεν πρόκειται για κάποιον χρήστη αλλά για κάποιο robot. Άρα θα πρέπει να γίνει προγραμματιστικά η τοποθέτηση του πεδίου στο αίτημα που θα στείλει ο crawler. Ένα παράδειγμα τέτοιας τοποθέτησης σε Java δίνεται παρακάτω:

```
DefaultHttpClient client = new DefaultHttpClient();
HttpProtocolParams.setUserAgent(client.getParams(), “Mozilla/5.0
Firefox/26.0”);
```

Αν παρά την τροποποίηση της κεφαλίδας User-Agent, εξακολουθούμε να διαπιστώνουμε μια περίεργη συμπεριφορά, όπως το να είναι η σελίδα που λαμβάνεται από τον διακομιστή ιστού κενή, να λείπουν πληροφορίες ή διαφορετικά να μην είναι αυτό που αναμένεται (με βάση του τι βλέπουμε στο πρόγραμμα περιήγησης ιστού), τότε θα πρέπει να βεβαιωθούμε ότι τα cookies διατηρούνται σωστά μεταξύ της κάθε φόρτωσης σελίδας και ότι τα cookies αποστέλλονται στον ιστότοπο για κάθε αίτημα. Ακόμα, πιθανότατα κάτι τέτοιο να προκαλείται από την εκτέλεσή JavaScript στον ιστότοπο για τη δημιουργία της σελίδα (Menshchikov A. et al. (2017)).

Ορισμένες ιστοσελίδες με καλή προστασία θα μπορούσαν να αποτρέψουν την αλληλεπίδραση με τον ιστότοπο, εάν αυτή γίνεται πολύ γρήγορα. Έτσι, η λήψη πολλών πληροφοριών από έναν ιστότοπο σημαντικά ταχύτερα από έναν κανονικό άνθρωπο είναι

ένας καλός τρόπος να ανιχνευτεί ένας crawler και να μπλοκαριστεί. Επομένως, παρόλο που ο πολυνηματικός προγραμματισμός μπορεί να είναι ένας πολύ καλός τρόπος για να φορτώσουμε σελίδες γρηγορότερα - επιτρέποντας την επεξεργασία των δεδομένων σε ένα νήμα, ενώ συνεχώς γίνεται το κατέβασμα σελίδων σε άλλο - είναι ωστόσο μια πολύ κακή πολιτική για τη σύνταξη καλών crawlers. Η γρήγορη απόσπαση είναι μια κακή πρακτική που θέτει ένα βαρύ φορτίο στους διακομιστές του διαχειριστή του ιστοτόπου. Μπορεί να μας φέρει μπροστά σε νομικό πρόβλημα, όπως είδαμε προηγουμένως και είναι η πρώτη αιτία για την τοποθέτηση των crawlers στη μαύρη λίστα.

Για το λόγο αυτό θα πρέπει πάντα να γίνεται προσπάθεια να διατηρείται ο ρυθμός αποστολής αιτημάτων σε μεμονωμένες σελίδες όσο το δυνατόν χαμηλότερος. Θα πρέπει δηλαδή να προσθέσουμε κάποια δευτερόλεπτα αναμονών ανάμεσα στα αιτήματα (π.χ. Thread sleep in java) και να αφήσουμε τον crawler να τρέξει όλη τη νύχτα.

Το επόμενο που θα πρέπει να προσέξει κάποιος κατά την συγγραφή ενός crawler είναι η αποφυγή των honeypots. Παρόλο που το CSS σε ένα μεγάλο βαθμό κάνει τη ζωή εξαιρετικά εύκολη όταν πρόκειται για τη διαφοροποίηση χρήσιμων πληροφοριών από μη χρήσιμες πληροφορίες (π.χ. διαβάζοντας τις ετικέτες id και class), μπορεί μερικές φορές η χρήση του να είναι προβληματική για τους scrapers. Αν ένα πεδίο περιέχει έναν σύνδεσμο ο οποίος κρύβεται από έναν χρήστη μέσω του CSS, είναι λογικό να υποθέσουμε ότι ο μέσος χρήστης που επισκέπτεται τον ιστότοπο δεν θα μπορεί να τον επιλέξει, μιας και δεν εμφανίζεται στο πρόγραμμα περιήγησης. Αν ο σύνδεσμος επιλεχθεί, είναι πιθανό αυτό να έχει γίνει από ένα bot.

Αυτό ισχύει όχι μόνο για συνδέσμους, αλλά και για φόρμες, εικόνες, αρχεία και οποιοδήποτε άλλο στοιχείο στον ιστότοπο που μπορεί να διαβαστεί από ένα bot αλλά είναι κρυφό από τον μέσο χρήστη που επισκέπτεται τον ιστότοπο μέσω ενός προγράμματος περιήγησης. Μια επίσκεψη σελίδας σε έναν “κρυφό” σύνδεσμο σε έναν ιστότοπο μπορεί εύκολα να ενεργοποιήσει μια δέσμη ενεργειών που θα αποκλείσει τη διεύθυνση IP του χρήστη, θα αποσυνδέσει τον χρήστη από τον ιστότοπο ή θα λάβει κάποια άλλη ενέργεια για να αποτρέψει την περαιτέρω πρόσβαση.

Όπως γίνεται αντιληπτό με την εφαρμογή κατάλληλων μέτρων, όλα τα εμπόδια μπορούν να παρακαμφθούν. Ένα πράγμα μόνο δεν μπορεί να παραποιηθεί και αυτό είναι η διεύθυνση IP. Εάν λαμβάνουμε σφάλματα HTTP και ειδικά το “403 Forbidden”, μπορεί αυτό να υποδηλώνει ότι ο ιστότοπος έχει αναγνωρίσει τη διεύθυνση IP που χρησιμοποιείται ως bot και δεν επιθυμεί να δεχτεί άλλες αιτήσεις. Θα πρέπει είτε να

περιμένουμε μέχρι να αφαιρεθεί η διεύθυνσή IP από τη «μαύρη λίστα» ή να αποκτήσουμε μια νέα διεύθυνση IP. Το να φτάσει κάποιος στο σημείο να μπλοκάρει τις διευθύνσεις IP είναι λίγο σαν ένας αγρότης που παραιτείται από το ψεκασμό φυτοφαρμάκων στο χωράφι και αντ' αυτού προτιμά την απλή πυρκαγιά του πεδίου. Είναι μια τελευταία αλλά αποτελεσματική μέθοδος απόρριψης πακέτων που αποστέλλονται από ενοχλητικές διευθύνσεις IP. Ωστόσο, υπάρχουν προβλήματα με αυτήν τη λύση.

Οι λίστες πρόσβασης στις διευθύνσεις IP είναι επώδυνο να διατηρηθούν. Παρόλο που οι μεγάλες ιστοσελίδες έχουν συνήθως τα δικά τους προγράμματα που αυτοματοποιούν κάποια από τη συνηθισμένη διαχείριση αυτών των λιστών (bots blocking bots!), κάποιος πρέπει να τα ελέγχει περιστασιακά ή τουλάχιστον να παρακολουθεί την ανάπτυξή τους για προβλήματα.

Κάθε διεύθυνση εντός της λίστας προσθέτει ένα μικρό χρόνο επεξεργασίας για τη λήψη πακέτων, καθώς ο διακομιστής πρέπει να ελέγξει τα ληφθέντα πακέτα έναντι των περιεχομένων της λίστας για να αποφασίσει αν θα τα εγκρίνει. Πολλές διευθύνσεις πολλαπλασιασμένες με πολλά πακέτα είναι κάτι που μπορεί να συμβεί σε σύντομο χρονικό διάστημα. Για να εξοικονομήσουν χρόνο και πολυπλοκότητα επεξεργασίας, οι διαχειριστές συχνά ομαδοποιούν αυτές τις διευθύνσεις IP σε μπλοκ και δημιουργούν κανόνες όπως «όλες οι 256 διευθύνσεις σε αυτό το εύρος είναι αποκλεισμένες» εάν υπάρχουν μερικοί στενά συγκεντρωμένοι παραβάτες. Αυτό μας οδηγεί στο τρίτο σημείο.

Το κλείδωμα διεύθυνσης IP μπορεί να οδηγήσει επίσης στην αποτροπή των «καλών παιδιών». Για παράδειγμα, αν κάποιος φοιτητής ενός πανεπιστημίου τρέχει κάποιο λογισμικό αυτοματοποιημένης πρόσβασης σε έναν ιστότοπο και ο διακομιστής αναγνωρίσει αυτό το πρόγραμμα και προχωρήσει στο μπλοκάρισμα των διευθύνσεων IP ολόκληρου του πανεπιστημίου, αυτό θα έχει ως αποτέλεσμα πολλοί πραγματικοί και ίσως τακτικοί χρήστες να μην μπορούν να έχουν πρόσβαση στον ιστότοπο αυτό. Παράλληλα ο φοιτητής αυτός θα μπορεί να μεταφέρει το λογισμικό του σε άλλη διεύθυνση IP και να συνεχίσει την εκτέλεση του αφήνοντας όμως το πρόβλημα στους υπόλοιπους χρήστες.

Παρά τα μειονεκτήματά της, η απαγόρευση πρόσβασης σε μια διεύθυνση IP παραμένει μια εξαιρετικά κοινή μέθοδος για τους διαχειριστές διακομιστών ώστε να σταματήσουν την πρόσβαση ύποπτων Web Scrapers σε ιστοσελίδες. Αν μια διεύθυνση IP είναι αποκλεισμένη, η μόνη πραγματική λύση είναι το crawling από διαφορετική

διεύθυνση IP. Αυτό μπορεί να επιτευχθεί μετακινώντας τον crawler σε ένα νέο διακομιστή ή δρομολογώντας τα αιτήματα μέσω ενός διαφορετικού διακομιστή χρησιμοποιώντας μια υπηρεσία όπως το Tor (Mitchell R. (2018)).

Ένας τελικός και ίσως ο πιο αποτελεσματικός τρόπος για την αποτροπή του crawling είναι το CAPTCHA. Ένα CAPTCHA είναι ένα πρόγραμμα που μπορεί να δημιουργήσει τεστ που οι περισσότεροι άνθρωποι μπορούν να περάσουν, αλλά τα αυτοματοποιημένα εργαλεία συλλογής πληροφοριών δεν μπορούν να περάσουν. Είναι ο πιο δημοφιλής τρόπος για την παροχή δοκιμής Turing σε μια ιστοσελίδα. Αυτό βέβαια θα οδηγήσει σε μια πιο δύσχρηστη ιστοσελίδα. Ίσως όμως είναι ένα τίμημα που θα χρειαστεί να πληρώσει ο ιδιοκτήτης της για να μπορέσει να μπλοκάρει ανεπιθύμητους crawlers (Menshchikov A. et al. (2017); Liu B. (2011)).

Οι ιδιοκτήτες ιστοτόπων μπορούν να κάνουν πραγματικά δύσκολο για τα bots να ανακτούν τα δεδομένα τους. Υπάρχουν όπως είδαμε πολλοί τρόποι για να γίνει ένας ιστότοπος "scraping-proof". Παρόλο που στην πραγματικότητα δεν υπάρχει τεχνική ασπίδα που θα μπορούσε να εμποδίσει ένα πλήρως αναπτυγμένο crawler από την ανάκτηση δεδομένων.

Με όλα τα παραπάνω υπόψιν και με βάση τον Toth A (2017), αν η ιστοσελίδα έχει πολλές παγίδες για crawlers, captchas και άλλα επίπεδα άμυνας εναντίον bots, τότε σίγουρα δεν είναι ευπρόσδεκτη η εφαρμογή αυτοματοποιημένης ανάκτησης δεόμενων με crawlers. Σε αυτή την περίπτωση, θα πρέπει να το σκεφτεί κάποιος δύο φορές πριν από την εφαρμογή κάποιου crawler στον ιστότοπο. Από τεχνική άποψη είναι δυνατό να καταπολεμηθούν όλοι οι τύποι άμυνας απέναντι σε bots, αλλά πρέπει ο προγραμματιστής να σκεφτεί αν αξίζει πραγματικά ο προγραμματιστικός κόπος που θα απαιτηθεί για αυτό.

3 Εργαλεία και τεχνολογίες ανάπτυξης εφαρμογής

3.1 Java

Παρά το γεγονός ότι όλο και περισσότερες γλώσσες προγραμματισμού γίνονται πιο δημοφιλείς, η Java παραμένει από τις πιο διαδεδομένες. Ο λόγος που συμβαίνει αυτό είναι πως όλα αυτά τα χρόνια η Java έχει υποστεί μια συνεχή ανάπτυξη στην αποτελεσματικότητα της και είναι σχεδιασμένη για να τρέχει συνεχώς και με συνέπεια. Το JDK (Java Development Kit) είναι μια συλλογή από χρήσιμα εργαλεία για την ανάπτυξη λογισμικού σε Java. Το πιο σημαντικό από αυτά είναι το javac (Java compiler), το οποίο μεταφράζει τα αρχεία Java σε Java bytecode. Το Java bytecode μπορούμε να το δούμε ως αρχεία .class που δημιουργούνται μετά την μεταγλώττιση των αρχείων .java. Αυτά τα Java bytecode αρχεία διαβάζονται από την Εικονική Μηχανή της Java (JVM – Java Virtual Machine) και μεταφράζονται σε γλώσσα assembly χαμηλότερου επιπέδου με βάση το κάθε λειτουργικό σύστημα.

Ένας από τους πιο σημαντικούς λόγους για τους οποίους η Java είναι τόσο δημοφιλής είναι ότι είναι ανεξάρτητη από την υποκείμενη πλατφόρμα. Τα προγράμματα σε Java μπορούν να λειτουργούν σε διάφορους τύπους υπολογιστών. Το μότο της Sun Microsystems, αναπτύσσοντας την Java, ήταν: “Write once, run anywhere”. Αυτό έγινε εφικτό με την δημιουργία διάφορων JVMs, ώστε να προστεθεί ένας buffer μεταξύ των λειτουργικών συστημάτων και επεξεργαστών και του Java bytecode που τρέχει σε αυτούς. Έτσι αντί να χρειάζεται κάποιος να γράψει λογισμικό για κάθε διαφορετικό περιβάλλον στο οποίο το λογισμικό αυτό θα έτρεχε, με την Java οι προγραμματιστές το γράφουν μια φορά, το μεταγλωττίζουν και αφήνουν στην JVM τα υπόλοιπα (Mitchell R. (2013)).

Εφόσον ο υπολογιστής έχει εγκατεστημένο το Java Runtime Environment (JRE), μπορεί να εκτελεστεί σε αυτόν ένα πρόγραμμα Java. Οι περισσότεροι τύποι υπολογιστών θα είναι συμβατοί με ένα JRE, συμπεριλαμβανομένων αυτών που λειτουργούν με Windows, Macintosh, Unix ή Linux, και μεγάλοι υπολογιστές mainframe, καθώς και κινητά τηλέφωνα.

Η Java είναι θεμελιωδώς αντικειμενοστρεφής. Ο κώδικας είναι τόσο ισχυρός επειδή τα αντικείμενα Java δεν περιέχουν αναφορές σε εξωτερικά δεδομένα. Η γλώσσα θεωρείται αρκετά απλή. Ωστόσο, έρχεται με μια βιβλιοθήκη κλάσεων που προσφέρουν

χρήσιμη λειτουργικότητα, απαραίτητη για τα περισσότερα προγράμματα Java. Το Java API, η βιβλιοθήκη κλάσεων, είναι τόσο σημαντικό μέρος της Java όσο και η ίδια η γλώσσα. Στην πραγματικότητα, η πραγματική πρόκληση της εκμάθησης του πώς να χρησιμοποιήσει κάποιος την Java δεν είναι η εκμάθηση της γλώσσας και της σύνταξης της, αλλά η εκμάθηση του API. Η γλώσσα αποτελείται από 50 λέξεις-κλειδιά, αλλά το Java API έχει χιλιάδες κλάσεις με δεκάδες χιλιάδες μεθόδους που μπορούν να χρησιμοποιηθούν σε ένα πρόγραμμα (Yuk C. (2017)).

Όπως επισημαίνει και ο Mitchell R. (2013) υπάρχουν πολλοί λόγοι για τους οποίους η Java είναι μια γλώσσα που μπορεί να χρησιμοποιηθεί για Web Scraping. Αυτοί είναι:

- Ο εξαιρετικός χειρισμός exception της Java επιτρέπει την σύνταξη κώδικα που χειρίζεται κομψά τα συχνά ακατάστατα δεδομένα ιστού
- Οι επαναχρησιμοποιήσιμες δομές δεδομένων δίνουν την δυνατότητα σε ένα προγραμματιστή να γράφει κάτι μία φορά και να το χρησιμοποιεί παντού με ευκολία και ασφάλεια
- Οι βιβλιοθήκες συντρέχοντος προγραμματισμού της Java επιτρέπουν τη συγγραφή κώδικα που μπορεί να επεξεργάζεται τα δεδομένα ενώ περιμένει τους διακομιστές να επιστρέφουν πληροφορίες (το βραδύτερο τμήμα οποιασδήποτε εφαρμογής Web Scraping)
- Υπάρχει μια ποικιλία τυπικών βιβλιοθηκών για τη λήψη δεδομένων από διακομιστές, καθώς και βιβλιοθήκες τρίτων για την ανάλυση αυτών των δεδομένων ακόμα και για την εκτέλεση JavaScript (που απαιτείται για την ανάκτηση των δεδομένων ορισμένων ιστοτόπων)
- Μπορεί να γίνει πολύ εύκολα η ενσωμάτωση διαφόρων framework (όπως το Selenium και το crawler4j) για αυτοματοποιημένο άνοιγμα ιστοσελίδων

3.2 Eclipse

Σύμφωνα με τον Mitchell R. (2013), η γραφή σε Java μπορεί συχνά να γίνει ανιαρή λόγω των αρκετά σχολαστικών απαιτήσεων της γλώσσας. Επειδή η Java είναι μια "strongly typed" γλώσσα, είναι απαραίτητο να θυμόμαστε ποιο τύπο δεδομένων έχουμε σε μια συγκεκριμένη μεταβλητή που βρίσκεται σε μια συγκεκριμένη ενότητα κώδικα και ποιες λειτουργίες μπορούν να γίνουν σε αυτήν. Μια κλάση που έχει οριστεί να δέχεται μια τιμή float δεν μπορεί να καλείται με ένα όρισμα double σε άλλο σημείο και μια στατική κλάση δεν μπορεί να καλείται αργότερα σε μη στατικό περιβάλλον. Για τον λόγο αυτό κατά την συγγραφή κώδικα σε Java, βοηθάει να υπάρχει ένα καλό εργαλείο - εκεί έρχεται ο IDE.

Ένα καλό ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) είναι σχεδόν απαραίτητο για την εργασία με την Java, ειδικά σε πολύπλοκα έργα. Ένα IDE επιτρέπει να συλλαμβάνονται τα σφάλματα κατά το compile και τα συντακτικά σφάλματα σε σύντομο χρονικό διάστημα, να χρησιμοποιούνται προηγμένες λειτουργίες debugging, να γίνεται η επιθεώρηση των αντικειμένων και ακόμη και να γίνεται εφαρμογή αυτόματης συμπλήρωσης κώδικα (code auto-complete).

Το Eclipse είναι ένα τέτοιο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για την ανάπτυξη εφαρμογών χρησιμοποιώντας τη γλώσσα προγραμματισμού Java, όπως και άλλες γλώσσες προγραμματισμού όπως C / C ++, Python, PERL, Ruby κλπ. Η πλατφόρμα Eclipse που παρέχει τις βάσεις για το Eclipse IDE αποτελείται από plug-ins και έχει σχεδιαστεί για να είναι επεκτάσιμη χρησιμοποιώντας πρόσθετα plug-ins. Αναπτύχθηκε με χρήση της Java και μπορεί να χρησιμοποιηθεί για την ανάπτυξη ενισχυμένων εφαρμογών, ολοκληρωμένων περιβαλλόντων ανάπτυξης και άλλων εργαλείων. Το Eclipse μπορεί να χρησιμοποιηθεί ως IDE για οποιαδήποτε γλώσσα προγραμματισμού για την οποία υπάρχει διαθέσιμο plug-in. Το Eclipse είναι διαθέσιμο κάτω από το Eclipse Public License (EPL), το οποίο διαβεβαιώνει πως το IDE αυτό είναι ελεύθερο για εγκατάσταση και χρήση (Tutorial Point (2019)).

Τα μεγαλύτερα πλεονεκτήματα στην χρήση του Eclipse είναι:

- Αυτοματοποιημένη ολοκλήρωση κώδικα (code auto-complete), που γλιτώνει τον χρήστη από πολλές ώρες αναζήτησης εντός της τεκμηρίωσης των διαφόρων βιβλιοθηκών και APIs που χρησιμοποιούνται.

- Υποστηρίζει το refactoring, δίνοντας μεγάλη ευελιξία κατά τη μετονομασία μεθόδων, μεταβλητών, κλάσεων κ.α.
- Προσφέρει άμεσο έλεγχο σύνταξης κατά την πληκτρολόγηση, βοηθώντας στην σωστή συγγραφή κώδικα.
- Τα imports συμπληρώνονται και οργανώνονται αυτόματα.
- Υποστηρίζει το Maven για την διαχείριση των dependencies.
- Προσφέρει έναν εύκολο στη χρήση debugger βοηθώντας στην αναζήτηση σφαλμάτων.
- Η ενσωμάτωση plug-ins μπορεί να γίνει πολύ εύκολα.
- Είναι δωρεάν και open source

3.3 MySQL

Ένα DBMS (σύστημα διαχείρισης βάσεων δεδομένων) είναι ένα σύστημα που διαχειρίζεται βάσεις δεδομένων και τις συνδέει με το λογισμικό. Για παράδειγμα, μια βάση δεδομένων μπορεί να χρησιμοποιηθεί για την εκτέλεση μιας ιστοσελίδας, για τη λειτουργία της βάσης δεδομένων ενός ERP ή οποιουδήποτε άλλου λογισμικού. Η MySQL είναι ένα ισχυρό, δωρεάν σύστημα διαχείρισης βάσεων δεδομένων ανοιχτού κώδικα που λειτουργεί εδώ και χρόνια. Είναι πολύ σταθερό και έχει μια μεγάλη κοινότητα που βοηθά στη διατήρηση, τον εντοπισμό σφαλμάτων και την αναβάθμιση του. Η MySQL μπορεί να μην είναι τόσο δημοφιλής για μεγαλύτερα συστήματα τα οποία θα εκτελούνται ως επί το πλείστον σε Microsoft SQL Server ή Oracle. Αυτά τα εμπορικά DBMS είναι πιο κλιμακούμενα, έχουν περισσότερους πόρους διαθέσιμους στην αγορά και έχουν πιο προηγμένες δυνατότητες από ότι η MySQL.

Η τελευταία έκδοση της MySQL, κατά τον Branson T. (2017), είναι μία από τις πιο δημοφιλής στον κόσμο. Είναι ανοιχτού κώδικα, αξιόπιστη, συμβατή με όλους τους σημαντικούς hosts, οικονομικά αποδοτική και εύκολη στη διαχείριση. Η MySQL έρχεται με το πλεονέκτημα της μεγάλης αυτής ευελιξίας. Η μεγάλη και σταθερή διαθεσιμότητα είναι ένα ακόμα χαρακτηριστικό της MySQL - οι επιχειρήσεις που την χρησιμοποιούν απολαμβάνουν διαθεσιμότητα 24/7. Ακόμα η MySQL έρχεται με μια μεγάλη ποικιλία διακομιστών σε cluster, όπως και master-slave ρυθμίσεων που επιτρέπουν άμεσο failover για συνεχή πρόσβαση. Είτε επιλέξει κάποιος να δημιουργήσει ένα ηλεκτρονικό

κατάστημα είτε ένα σύστημα επεξεργασίας υψηλής ταχύτητας, η MySQL έχει σχεδιαστεί για να επεξεργάζεται εκατομμύρια ερωτήματα και χιλιάδες συναλλαγές, εξασφαλίζοντας ταυτόχρονα ευρετήρια πλήρους κειμένου και βέλτιστη ταχύτητα.

Η προστασία ευαίσθητων επιχειρηματικών πληροφοριών είναι το κύριο μέλημα κάθε επιχείρησης. Η MySQL μπορεί και διασφαλίζει τα δεδομένα με εξαιρετικά χαρακτηριστικά προστασίας δεδομένων. Η ισχυρή κρυπτογράφηση δεδομένων αποτρέπει την μη εξουσιοδοτημένη προβολή δεδομένων και υποστηρίζει SSH και SSL εξασφαλίζοντας ασφαλέστερες συνδέσεις. Διαθέτει επίσης έναν ισχυρό μηχανισμό που περιορίζει μόνο σε εξουσιοδοτημένους χρήστες την πρόσβαση στον server.

Η MySQL, όπως προσθέτει ο Branson T. (2017), είναι πολύ απλή στην χρήση και κάποιος μπορεί να μεταβεί από τη λήψη του λογισμικού στην πλήρη εγκατάσταση του σε μόλις 15 λεπτά. Η MySQL είναι εξαιρετικά γρήγορη, ανεξάρτητα από την υποκείμενη πλατφόρμα. Διαθέτει δυνατότητες αυτοδιαχείρισης όπως αυτόματη επανεκκίνηση, επέκταση χώρου και αυτόματες αλλαγές διαμόρφωσης προσδίδοντας ευκολία στην διαχείριση της βάσης. Έρχεται επίσης με ένα ολοκληρωμένο σύνολο εργαλείων για υποστήριξη migration και μια γραφική σουίτα διαχείρισης με πολλές δυνατότητες. Η MySQL επιτρέπει την παρακολούθηση επιδόσεων σε πραγματικό χρόνο για την έγκαιρη αντιμετώπιση προβλημάτων επιχειρησιακής λειτουργίας από ένα μόνο σταθμό εργασίας. Για όλους αυτούς τους λόγους, οι οργανισμοί χρησιμοποιούν τη MySQL για την άμεση ανάπτυξη και εκκίνηση εφαρμογών και για αυτό επιλέχθηκε και στην παρούσα υλοποίηση.

Για την δημιουργία και διαχείριση της βάσης χρησιμοποιείται το phpMyAdmin. Το εργαλείο αυτό είναι δωρεάν και ανοικτού κώδικα. Είναι ένα από τα πιο δημοφιλή εργαλεία διαχείρισης βάσεων MySQL κυρίως για τις υπηρεσίες web hosting. Μερικά από τα κύρια χαρακτηριστικά του εργαλείου είναι η ύπαρξη web interface, η δυνατότητα εισαγωγής δεδομένων από CSV και SQL, η εξαγωγή σε διάφορους τύπους δεδομένων (CSV, SQL, XML, PDF, Word, Excel κ.α.), η δυνατότητα δημιουργίας γραφικής αναπαράστασης του layout της βάσης, η ικανότητα δημιουργίας πολύπλοκων ερωτημάτων (queries) με την λειτουργία Query-By-Example (QBE), συμβατότητα με πολλά λειτουργικά συστήματα κ.α.

Για το hosting της βάσης χρησιμοποιείται στην υλοποίηση το remotemysql.com. Αποτελεί μια hosting υπηρεσία που είναι δωρεάν, παρέχει δυνατότητα άμεσης ενεργοποίησης, διαχείρισης και χρήσης μέσω του phpMyAdmin, όπως επίσης παρέχει

την δυνατότητα για απομακρυσμένη σύνδεση στη βάση απλά με χρήση username και password. Τέλος παρά το ότι είναι δωρεάν δεν έχει κανένα όριο στα ερωτήματα που μπορούν να τεθούν όπως και κανένα όριο στο bandwidth.

3.4 Selenium

Το Selenium είναι ένα σύνολο διάφορων εργαλείων λογισμικού το καθένα με διαφορετική προσέγγιση για την υποστήριξη της αυτοματοποίησης. Τα εργαλεία αυτά είναι ευρέως διαδεδομένα στην διασφάλιση ποιότητας λογισμικού, ωστόσο οι λειτουργίες τους που είναι εξαιρετικά ευέλικτες, επιτρέπουν πολλές επιλογές για τον εντοπισμό στοιχείων εντός μιας ιστοσελίδας όπως και για την ανάκτηση ιστοσελίδων. Αυτό τα κάνει μια πολύ καλή επιλογή και για τη χρήση από crawlers. Ένα από τα βασικά χαρακτηριστικά του Selenium είναι η υποστήριξη για την εκτέλεση δοκιμών σε πολλές πλατφόρμες προγραμμάτων περιήγησης (Selenium Project (2019)).

Το Selenium ξεκίνησε για πρώτη φορά το 2004 από τον Jason Huggins. Αναπτύχθηκε μια βιβλιοθήκη Javascript που θα μπορούσε να οδηγήσει σε αλληλεπιδράσεις με μια ιστοσελίδα, επιτρέποντάς της να επαναλάβει αυτόματα τις δοκιμές σε πολλαπλά προγράμματα περιήγησης. Αυτή η βιβλιοθήκη τελικά έγινε το Selenium Core. Το Selenium ήταν πρωτοποριακό επειδή κανένα άλλο προϊόν ως τότε δεν επέτρεπε τον έλεγχο ενός προγράμματος περιήγησης από μια γλώσσα της επιλογής μας.

Το 2006, ένας μηχανικός της Google με το όνομα Simon Stewart άρχισε να εργάζεται σε ένα έργο που ονομάζεται WebDriver. Η Google έκανε από καιρό βαριά χρήση του Selenium, αλλά οι testers έπρεπε να εργαστούν ώστε να βρουν λύσεις στους περιορισμούς του προϊόντος. Ο Simon ήθελε ένα εργαλείο δοκιμών που θα μπορούσε να μιλήσει απευθείας στο πρόγραμμα περιήγησης χρησιμοποιώντας “native” μεθόδους για το πρόγραμμα περιήγησης και το λειτουργικό σύστημα, αποφεύγοντας έτσι τους περιορισμούς ενός περιβάλλοντος Javascript. Το έργο WebDriver ξεκίνησε με στόχο την επίλυση των σημείων στα οποία υστερούσε το Selenium.

Το πρωτεύον νέο χαρακτηριστικό στο Selenium 2.0 είναι η ενσωμάτωση του API WebDriver. Το WebDriver έχει σχεδιαστεί για να προσφέρει μια απλούστερη, πιο συνοπτική προγραμματιστική διασύνδεση. Ο στόχος του WebDriver είναι να παρέχει ένα

καλά σχεδιασμένο αντικειμενοστρεφές API που παρέχει βελτιωμένη υποστήριξη για σύγχρονα και εξελιγμένα προβλήματα δοκιμής εφαρμογών ιστού (Selenium Project (2019)).

Ο τρόπος που μπορεί να εγκατασταθεί το Selenium, ώστε να μπορεί να γίνει συγγραφή ενός προγράμματος που θα το χρησιμοποιεί, εξαρτάται από τη γλώσσα προγραμματισμού και το περιβάλλον ανάπτυξης. Ο ευκολότερος τρόπος για τη δημιουργία ενός έργου στην Java στο οποίο θα γίνεται χρήση του Selenium είναι με τη χρήση του Maven. Το Maven θα κατεβάσει τα java bindings (η βιβλιοθήκη πελατών Selenium 2.0 java) και όλες τις εξαρτήσεις του και θα δημιουργήσει το έργο, χρησιμοποιώντας ένα αρχείο maven pom.xml (project configuration). Μόλις γίνει αυτό θα μπορεί να ανοιχτεί το έργο maven στο προτιμώμενο IDE, όπως είναι το IntelliJ IDEA ή το Eclipse.

Δεν θα παρουσιαστούν οι λεπτομέρειες των αρχείων pom.xml ή πως γίνεται η χρήση του Maven, αφού υπάρχουν ήδη εξαιρετικές αναφορές σε αυτό. Η προσθήκη που πρέπει να γίνει στο αρχείο pom.xml θα μοιάζει με την παρακάτω:

```
<dependency>
      <groupId>org.seleniumhq.selenium</groupId>
      <artifactId>selenium-java</artifactId>
      <version>3.7.1</version>
</dependency>
```

Μόλις εγκατασταθεί το project, μπορεί να παρατηρήσει κανείς ότι το WebDriver λειτουργεί όπως κάθε κανονική βιβλιοθήκη. Είναι απολύτως αυτοτελής και συνήθως δεν χρειάζεται ο προγραμματιστής να θυμάται να ξεκινήσει οποιαδήποτε πρόσθετη διαδικασία ή να εκτελέσει οποιονδήποτε εγκαταστάτη πριν τη χρησιμοποιήσει. Ένα απλό παράδειγμα χρήσης του Web Driver δίνεται παρακάτω, με το οποίο ανοίγει αυτόματα ο φυλλομετρητής Mozilla, όπου κατευθύνεται σε μια ιστοσελίδα της οποίας ο πηγαίος κώδικας προβάλλεται στην κονσόλα.

```
package org.openqa.selenium.example;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.firefox.FirefoxProfile;

public class SeleniumExample  {
```

```
    public static void main(String[] args) {
        // Create a new instance of the Firefox driver
        WebDriver driver = new FirefoxDriver();

        // And now use this to visit Google
        driver.get("http://www.google.com");

        // Print page source in the console
        System.out.println(driver.getPageSource());

        //Close the browser
        driver.quit();
    }
}
```

Το WebDriver είναι το όνομα του βασικού interface το οποίο όμως έχει αρκετές υλοποιήσεις. Σε αυτές περιλαμβάνονται οι FirefoxDriver, InternetExplorerDriver, Chrome, Opera. Στην παρούσα εργασία χρησιμοποιείται ο FirefoxDriver. Ο λόγος είναι ότι τρέχει σαν ένας κανονικός browser υποστηρίζοντας και Javascript. Επίσης είναι αρκετά γρήγορος και δεν προσθέτει overhead. Επιπλέον, δίνει έναν εύκολο τρόπο διαχείρισης των cookies ενώ μέσω της χρήσης του Firefox Profile έχουμε τη δυνατότητα να τροποποιήσουμε και τον User Agent ενός session. Παρακάτω φαίνεται πως μπορεί να γίνει αυτό με ένα απλό παράδειγμα:

```
// Go to the correct domain
driver.get("http://www.example.com");

// Now set the cookie. This one's valid for the entire domain
Cookie cookie = new Cookie("key", "value");
driver.manage().addCookie(cookie);

// You can delete cookies in 3 ways
// By name
driver.manage().deleteCookieNamed("CookieName");
// By Cookie
driver.manage().deleteCookie(loadedCookie);
// Or all of them
driver.manage().deleteAllCookies();
```

```
FirefoxProfile profile = new FirefoxProfile();
profile.addAdditionalPreference("general.useragent.override", "some UA
string");
WebDriver driver = new FirefoxDriver(profile);
```

3.5 Jsoup

Με μια απλή αναζήτηση για τους τρόπους ανάκτησης ιστοσελίδων, μπορεί κανείς να ανακαλύψει ότι μια από τις πιο διαδεδομένες είναι η βιβλιοθήκη Jsoup. Η βιβλιοθήκη αυτή είναι σχεδιασμένη για την λειτουργία με τα δεδομένα HTML που είναι διαθέσιμα στις διάφορες ιστοσελίδες. Προσφέρει ένα πολύ βολικό και ευέλικτο API για την εξαγωγή και διαχείριση των δεδομένων της HTML, χρησιμοποιώντας και αξιοποιώντας το DOM, το CSS και μεθόδους εφαρμογής ερωτημάτων (Hedley J. (2019)).

Το Jsoup υλοποιεί τις προδιαγραφές της HTML5 και αναλύει το HTML έγγραφο με τον ίδιο τρόπο, δηλαδή στο ίδιο DOM, όπως τα σύγχρονα προγράμματα περιήγησης. Οι δυνατότητες του συνοψίζονται στα εξής:

- Ανάκτηση και ανάλυση HTML από ένα URL, ένα αρχείο ή ένα αλφαριθμητικό
- Αναζήτηση και εξαγωγή δεδομένων, με βάση το DOM ή με χρήση CSS selectors
- Διαχείριση των HTML στοιχείων, όπως οι κόμβοι, τα attributes και το απλό κείμενο
- Δυνατότητα εκκαθάρισης περιεχομένου που εισάγεται από τον χρήστη
- Παραγωγή τακτοποιημένου HTML κειμένου.

Για την εγκατάσταση του Jsoup, αρκεί να εισάγουμε την αντίστοιχη εξάρτηση εντός του pom.xml και το Maven θα εισάγει όλες τις απαραίτητες βιβλιοθήκες εντός του project μας.

```
<dependency>
	<groupId>org.jsoup</groupId>
	<artifactId>jsoup</artifactId>
	<version>1.11.3</version>
</dependency>
```

Για την ανάκτηση μιας ιστοσελίδας με την χρήση αυτού του API χρησιμοποιείται η μέθοδος Jsoup.connect(String url). Αυτή η μέθοδος δημιουργεί ένα αντικείμενο της διασύνδεσης Connection της βιβλιοθήκης αυτής, απ’ όπου μπορεί να γίνει η ανάκτηση του αρχείου HTML όπως φαίνεται παρακάτω.

```
Document doc = Jsoup.connect("http://example.com/").get();
String title = doc.title();
String html = Jsoup.connect("http://example.com/").get().html();
```

Οι συνδέσεις περιέχουν αντικείμενα Connection.Request και Connection.Response. Η διαμόρφωση ενός αιτήματος μπορεί να γίνει χρησιμοποιώντας είτε τις μεθόδους συντόμευσης στη σύνδεση (π.χ. userAgent (String)), είτε με μεθόδους απευθείας στο αντικείμενο Connection.Request. Όλες οι ρυθμίσεις αιτήματος πρέπει να γίνουν πριν από την εκτέλεσή του.

```
Document doc = Jsoup.connect("http://example.com")
  .data("query", "Java")
  .userAgent("Mozilla")
  .cookie("auth", "token")
  .timeout(3000)
  .post();
```

3.6 XPath

Η XPath (XML Path Language) είναι μια γλώσσα ερωτήματος (query language) και αποτελεί βασικό στοιχείο του προτύπου XSLT (eXtensible Stylesheet Language Transformations). Η XPath καθορίστηκε από την Κοινοπραξία World Wide Web (W3C) και βασίζεται σε μια αναπαράσταση δέντρου του εγγράφου XML, παρέχοντας τη δυνατότητα πλοήγησης στο δέντρο, επιλέγοντας κόμβους με βάση διάφορα κριτήρια. Επιπλέον, η XPath μπορεί να χρησιμοποιηθεί για να υπολογίσει τις τιμές (π.χ. αλφαριθμητικά, αριθμούς ή τιμές Boolean) από το περιεχόμενο ενός εγγράφου XML. Στη δημοφιλή χρήση (αν και όχι στην επίσημη τεκμηρίωση), μια έκφραση XPath αναφέρεται συχνά απλά ως “XPath” (W3Schools (2019)).

Η XPath έχει υιοθετηθεί από μια σειρά βιβλιοθηκών και εργαλείων επεξεργασίας XML μαζί με μια βιβλιοθήκη τυπικών λειτουργιών που περιλαμβάνει. Χρησιμοποιεί εκφράσεις διαδρομής τοποθεσίας (paths) για πλοήγηση σε έγγραφα XML που αποτελούν και το πιο σημαντικό κομμάτι της. Ένας κόμβος ή ένα σύνολο κόμβων επιλέγεται

ακολουθώντας τα βήματα μιας έκφρασης διαδρομής. Παρακάτω δίνεται ένας πίνακας των πιο χρήσιμων εκφράσεων για την επιλογή κόμβων μέσα σε μια έκφραση διαδρομής.

Πίνακας 3-1: XPath expressions (W3Schools (2019))

Έκφραση	Περιγραφή
nodename	Επιλέγει όλους τους κόμβους με όνομα "*nodename*"
/	Ξεκινάει την επιλογή από τον κόμβο ρίζα (root node)
//	Ξεκινάει την επιλογή εντός του εγγράφου από τον κόμβο που ταιριάζει στην περιγραφή που ακολουθεί ανεξάρτητα από τη θέση εντός του εγγράφου
.	Επιλέγει τον τρέχων κόμβο
..	Επιλέγει τον κόμβο πατέρα του τρέχοντος κόμβου
@	Επιλέγει attributes

Η πιο συνηθισμένη σύνταξη της XPath είναι "//nodename[@attribute='value']" (Guojun Z. et al. (2017)). Εντός των αγκυλών, που είναι προαιρετικές, μπορεί να εισαχθεί ένα ή παραπάνω predicates τα οποία μπορούν να χρησιμοποιηθούν για να περιοριστεί η αναζήτηση. Αν για παράδειγμα υπήρχε το παρακάτω έγγραφο XML θα μπορούσαν να επιλεχθούν όλοι οι κόμβοι με όνομα "book" με την XPath "//book". Για την επιλογή των κόμβων title που έχουν στο attribute lang ως γλώσσα τα αγγλικά η αντίστοιχη έκφραση θα ήταν " //title[@lang='en'] "

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>

<book>
  <title lang="en">Harry Potter</title>
  <price>29.99</price>
</book>
<book>
  <title lang="en">Learning XML</title>
  <price>39.95</price>
</book>
</bookstore>
```

Υπάρχουν δύο τρόποι γραφής μιας έκφρασης διαδρομής τοποθεσίας, ο απόλυτος (Absolute XPath) και ο σχετικός (Relative XPath). Μια απόλυτη διαδρομή τοποθεσίας αρχίζει με μια κάθετο (/) και μια σχετική διαδρομή τοποθεσίας είτε χωρίς κάθετο ή με διπλή κάθετο. Και στις δύο περιπτώσεις, η διαδρομή τοποθεσίας αποτελείται από ένα ή περισσότερα βήματα, κάθε ένα από τα οποία χωρίζεται με μια κάθετο. Κάθε βήμα αξιολογείται σε σχέση με τους κόμβους του τρέχοντος συνόλου κόμβων και έχει τρία στοιχεία τα οποία είναι (i) Axis, (ii) Node-test, (iii) κανένα ή περισσότερα predicates. Μια έκφραση XPath αξιολογείται σε σχέση με έναν *context node*. Ένας προσδιοριστής άξονα (Axis Specifier) όπως "child" ή "descendant" καθορίζει την κατεύθυνση πλοήγησης από τον *context node*. Το πεδίο Node-test μπορεί να αποτελείται από συγκεκριμένα ονόματα κόμβων ή γενικότερες εκφράσεις και τα predicates είναι γραμμένα ως εκφράσεις σε αγκύλες. Και τα δύο παραπάνω χρησιμοποιούνται για να φιλτράρουν τους κόμβους που καθορίζονται από τον προσδιοριστή άξονα. Η σύνταξη είναι της μορφής " axisname::nodetest[predicate] ". Για περισσότερες πληροφορίες και παραδείγματα, μπορεί κανείς να ανατρέξει στην εκτενή τεκμηρίωση που υπάρχει δημοσιευμένη (W3Schools (2019)).

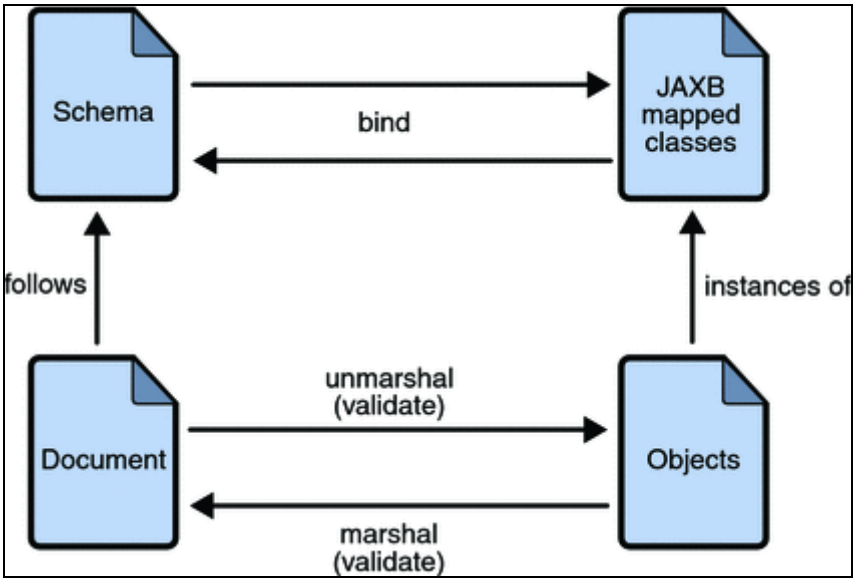
3.7 JAXB API

Το Java Architecture for XML Binding, εν συντομία JAXB, παρέχει ένα API και εργαλεία που αυτοματοποιούν τη χαρτογράφηση μεταξύ εγγράφων XML και αντικειμένων Java. Το JAXB παρέχει στους προγραμματιστές Java έναν αποδοτικό και καθορισμένο τρόπο χαρτογράφησης μεταξύ του κώδικα XML και της Java. Οι προγραμματιστές που χρησιμοποιούν το JAXB είναι πιο παραγωγικοί επειδή μπορούν να γράψουν λιγότερες γραμμές κώδικα και δεν χρειάζεται να είναι ειδικοί σε XML. Το JAXB διευκολύνει τους προγραμματιστές να επεκτείνουν τις εφαρμογές τους με τεχνολογίες XML και Web Services (Oracle (2013)).

Η XML αποτελεί ένα πρότυπο στο κλάδο της τεχνολογίας για την ανταλλαγή δεδομένων ανάμεσα σε διάφορες εφαρμογές γραμμένες ακόμα και σε διαφορετική γλώσσα. Όταν εργαζόμαστε με την XML, χρειάζεται ένας τρόπος να ληφθούν τα δεδομένα από ένα αρχείο XML και στη συνέχεια αυτά τα δεδομένα να μετατραπούν σε κάποια μορφή, την οποία το πρόγραμμά μας μπορεί να χειριστεί. Πρέπει επίσης να

υπάρχει η δυνατότητα τα δεδομένα των αντικειμένων της Java να μπορούν να σειριοποιηθούν ώστε να περαστούν σε κάποιο έγγραφο XML. Κατά τον Thompson J. (2018) αυτές τις λειτουργίες που στο framework αυτό αναφέρονται ως unmarshaling και marshaling μπορούν να πραγματοποιηθούν εύκολα με το JAXB.

- Αποσύνθεση περιεχομένου XML σε αναπαράσταση Java (Unmarshal)
- Σύνθεση της αναπαράστασης σε Java του περιεχομένου XML και πάλι σε περιεχόμενο XML (Marshal)



Εικόνα 3-1: Βήματα της διαδικασίας σύνδεσης στο JAXB (Oracle (2010))

Για να μπορέσουμε να χρησιμοποιήσουμε το API απλά προσθέτουμε το παρακάτω dependency στο pom.xml του project μας.

```
<dependency>
	<groupId>javax.xml.bind</groupId>
	<artifactId>jaxb-api</artifactId>
	<version>2.3.0</version>
</dependency>
```

Σύμφωνα με τον Thompson J. (2018), για να μπορεί ένα Java αντικείμενο να μετατρέπεται από και προς την μορφή XML θα πρέπει να οριστούν σε αυτό τα παρακάτω JAXB annotations:

- @XmlRootElement: χρησιμοποιείται στην κλάση του ανώτερου επιπέδου για να υποδείξει το στοιχείο ρίζας στο έγγραφο XML. Το χαρακτηριστικό

όνομα στο σχολιασμό είναι προαιρετικό. Εάν δεν έχει καθοριστεί, το όνομα της κλάσης χρησιμοποιείται ως το στοιχείο ρίζας XML στο έγγραφο.

- @XmlAttribute: χρησιμοποιείται για να δηλωθεί μια ιδιότητα κάποιου στοιχείου.
- @XmlElement: Αυτός ο σχολιασμός χρησιμοποιείται στις ιδιότητες της κλάσης που θα είναι τα επιμέρους στοιχεία του στοιχείου ρίζας.

Ένα παράδειγμα μιας κλάσης που γίνεται χρήση των annotations δίνεται παρακάτω (Mkyong (2012)).

```
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class Customer {

	String name;
	int age;
	int id;

	public String getName() {
		return name;
	}
	@XmlElement
	public void setName(String name) {
		this.name = name;
	}
	public int getAge() {
		return age;
	}

	@XmlElement
	public void setAge(int age) {
		this.age = age;
	}
	public int getId() {
		return id;
	}

	@XmlAttribute
	public void setId(int id) {
		this.id = id;
	}

}
```

Παρακάτω δίνεται ένα παράδειγμα χρήσης του JAXB Marshaler, για την μετατροπή ενός αντικειμένου Customer του παραδείγματος σε αρχείο XML (Mkyong (2012)).

```
import java.io.File;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;

public class JAXBExample {
        public static void main(String[] args) {

          Customer customer = new Customer();
          customer.setId(100);
          customer.setName("dimitris");
          customer.setAge(29);

          try {

                File file = new File("C:\\file.xml");
                JAXBContext jaxbContext =
JAXBContext.newInstance(Customer.class);
                Marshaller jaxbMarshaller = jaxbContext.createMarshaller();

                // output pretty printed
                jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
true);

                jaxbMarshaller.marshal(customer, file);
                jaxbMarshaller.marshal(customer, System.out);

              } catch (JAXBException e) {
                e.printStackTrace();
              }

        }
}
```

Το αρχείο που θα παραχθεί από τα παραπάνω είναι:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<customer id="100">
    <age>29</age>
    <name>dimitris</name>
</customer>
```

Τέλος ένα παράδειγμα της αντίστροφης διαδικασίας, την μετατροπή δηλαδή ενός XML αρχείου σε ένα αντικείμενο Customer, δίνεται παρακάτω (Mkyong (2012)).

```
import java.io.File;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Unmarshaller;

public class JAXBExample {
        public static void main(String[] args) {

         try {

                File file = new File("C:\\file.xml");
                JAXBContext jaxbContext =
JAXBContext.newInstance(Customer.class);

                Unmarshaller jaxbUnmarshaller =
jaxbContext.createUnmarshaller();
                Customer customer = (Customer)
jaxbUnmarshaller.unmarshal(file);
                System.out.println(customer);

          } catch (JAXBException e) {
                e.printStackTrace();
          }

        }
}
```

Αυτό που θα τυπωθεί στην κονσόλα είναι

```
Customer [name=dimitris, age=29, id=100]
```

3.8 HTML Cleaner

Βάσει της τεκμηρίωσης (Sourceforge (2019)) ο HtmlCleaner είναι ένας HTML parser ανοιχτού κώδικα γραμμένος σε Java. Τα HTML έγγραφα που υπάρχουν στο Web είναι συνήθως βρώμικα, κακά δομημένα και ακατάλληλα για περαιτέρω επεξεργασία. Για οποιαδήποτε σοβαρή «κατανάλωση» τέτοιων εγγράφων, είναι απαραίτητο να καθαριστεί πρώτα το χάος και να μπουν σε κάποια τάξη οι ετικέτες, τα χαρακτηριστικά

και το απλό κείμενο. Για κάθε έγγραφο δεδομένων σε HTML, το HtmlCleaner αναδιατάσσει μεμονωμένα στοιχεία και παράγει καλά μορφοποιημένη XML. Από προεπιλογή, ακολουθεί παρόμοιους κανόνες που χρησιμοποιούν τα περισσότερα προγράμματα περιήγησης ιστού για τη δημιουργία του DOM (Document Object Model). Ωστόσο, είναι δυνατή η παροχή προσαρμοσμένων συνόλων ετικετών και κανόνων για φιλτράρισμα και αντιστοίχηση ετικετών.

Το HtmlCleaner μπορεί να χρησιμοποιηθεί είτε στον κώδικα Java, είτε ως εργαλείο γραμμής εντολών ή ως εργασία Ant. Είναι σχεδιασμένο να είναι μικρό, ανεξάρτητο (χωρίς εξαρτήσεις κατά τον χρόνο εκτέλεσης εκτός από το JRE 1.5+), γρήγορο και ευέλικτο (η συμπεριφορά του είναι ρυθμιζόμενη μέσω του αριθμού των παραμέτρων). Παρόλο που το κύριο κίνητρο πίσω από την ανάπτυξη αυτής της βιβλιοθήκης ήταν να δώσει τη δυνατότητα στους προγραμματιστές να προετοιμάσουν τον κοινό κώδικα HTML, ώστε να μπορεί να γίνει επεξεργασία XML με χρήση XPath, XQuery και XSLT, τα δομημένα δεδομένα που παράγονται από το HtmlCleaner μπορούν να καταναλωθούν και να χειριστούν και με άλλους τρόπους.

Συνοπτικά τα βασικά στοιχεία της λειτουργικότητας του HtmlCleaner είναι:

- Η ανάλυση (parsing) της HTML που εισάγεται και η παραγωγή δομημένης δενδροειδούς μορφής κατάλληλης για προγραμματιστική επεξεργασία.
- Περιλαμβάνει Serializers υπεύθυνους για την εξαγωγή της DOM δομής σε XML, HTML, JDom.
- Η φάση ανάλυσης βασίζεται σε περιγραφές ετικετών και μπορεί να προσαρμοστεί από το χρήστη μέσω ενός μεγάλου αριθμού παραμέτρων.
- Είναι thread safe, και ένα μόνο στιγμιότυπο μπορεί να καθαρίσει πολλές πηγές html ταυτόχρονα.
- Το HtmlCleaner μπορεί να χρησιμοποιηθεί από τον κώδικα Java, από τη γραμμή εντολών ή ως εργασία Ant.
- Απαιτεί έκδοση Java 1.5 ή μεγαλύτερη.

Για να είναι δυνατή η χρήση των λειτουργιών που προσφέρει η βιβλιοθήκη του HtmlCleaner απλά εισάγεται το παρακάτω dependency στο pom.xml ενός project.

```
<dependency>
      <groupId>net.sourceforge.htmlcleaner</groupId>
      <artifactId>htmlcleaner</artifactId>
      <version>2.21</version>
</dependency>
```

Παρακάτω δίνεται ένα παράδειγμα για το πώς ένα λανθασμένα μορφοποιημένο έγγραφο σε HTML που περιέχει ανοιχτά tags και quotes που λείπουν, μετατρέπεται σε ένα καθαρό και σωστά δομημένο έγγραφο XML.

Τα δεδομένα πριν την επεξεργασία:

```
<table id=table1 cellspacing=2px
    <h1>CONTENT</h1>
    <td><a href=index.html>1 -> Home Page</a>
    <td><a href=intro.html>2 -> Introduction</a>
```

Τα δεδομένα μετά την επεξεργασία και τον καθαρισμό από το HtmlCleaner:

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
   <head />
   <body>
      <h1>CONTENT</h1>
      <table id="table1" cellspacing="2px">
         <tbody>
            <tr>
               <td>
                  <a href="index.html">1 -&gt; Home Page</a>
               </td>
               <td>
                  <a href="intro.html">2 -&gt; Introduction</a>
               </td>
            </tr>
         </tbody>
      </table>
   </body>
</html>
```

Ένα παράδειγμα χρήσης σε Java της μεθόδου clean που προσφέρεται από το HtmlCleaner δίνεται παρακάτω.

```
CleanerProperties props = new CleanerProperties();

// set some properties to non-default values
props.setTranslateSpecialEntities(true);
```

```
props.setTransResCharsToNCR(true);
props.setOmitComments(true);

// do parsing
TagNode tagNode = new HtmlCleaner(props).clean(
    new URL("http://www.chinadaily.com.cn/")
);

// serialize to xml file
new PrettyXmlSerializer(props).writeToFile(
    tagNode, "chinadaily.xml", "utf-8"
);
```

3.9 Πρότυπα σχεδίασης

Κατά τον Χατζηγεωργίου Α.(2005), «τα προβλήματα που αντιμετωπίζει ένας προγραμματιστής κατά τη διάρκεια σχεδίασης και υλοποίησης ενός συστήματος λογισμικού, πολύ σπάνια εμφανίζονται για πρώτη φορά μόνο στο συγκεκριμένο έργο. Συνήθως πρόκειται για προβλήματα που έχουν παρουσιαστεί και αντιμετωπισθεί επιτυχώς σε προηγούμενα έργα λογισμικού από άλλους προγραμματιστές. Ο στόχος των προτύπων σχεδίασης (design patterns) είναι να συστηματοποιήσουν συνηθισμένες λύσεις σε συνηθισμένα προβλήματα λογισμικού.» Στην παρούσα εργασία έγινε η χρήση τριών προτύπων σχεδίασης, που είναι το "Μοναδιαίο", το πρότυπο "Στρατηγική" και το πρότυπο "Μέθοδος Υπόδειγμα". Παρακάτω γίνεται μια σύντομη περιγραφή των προτύπων, ενώ στην παρουσίαση του έργου που δημιουργήθηκε, θα εξηγηθεί πως αυτά υλοποιήθηκαν.

3.9.1 Μοναδιαίο – Singleton

Το πρότυπο σχεδίασης "Μοναδιαίο" (Singleton) εξασφαλίζει ότι μια κλάση θα έχει μόνο ένα στιγμιότυπο και παρέχει ένα καθολικό σημείο πρόσβασης σε αυτό. Το αντικείμενο αυτό συνήθως δημιουργείται κατά την έναρξη της εφαρμογής και διαγράφεται με το πέρας της. Ο ρόλος του είναι η διαχείριση των υπολοίπων αντικειμένων της εφαρμογής και για το λόγο αυτό, αποτελεί λογικό σφάλμα να δημιουργηθούν περισσότερα του ενός τέτοια αντικείμενα – διαχειριστές.

Αυτό το πρότυπο σχεδίασης, εξασφαλίζει τη δημιουργία ενός και μόνο αντικειμένου, περιλαμβάνοντας μια ειδική μέθοδο κατασκευής στιγμιοτύπων. Όταν καλείται αυτή η μέθοδος, ελέγχει αν κάποιο αντικείμενο έχει ήδη δημιουργηθεί. Αν ναι, η μέθοδος επιστρέφει απλώς έναν δείκτη προς το υπάρχον αντικείμενο. Αν όχι, η μέθοδος δημιουργεί ένα νέο αντικείμενο και επιστρέφει δείκτη προς αυτό. Ο κατασκευαστής της κλάσης δηλώνεται ως προστατευόμενος (protected) ή ιδιωτικός (private). Με τον τρόπο αυτό, δεν είναι δυνατόν να δημιουργηθεί ένα αντικείμενο παρακάμπτοντας την παραπάνω ειδική μέθοδο (Χατζηγεωργίου Α. (2005)).

3.9.2 Στρατηγική – Strategy

Το πρότυπο σχεδίασης "Στρατηγική" (Strategy) ορίζει μια οικογένεια αλγορίθμων, τους ενσωματώνει και επιτρέπει την εναλλαγή μεταξύ αυτών. Το πρότυπο δίνει τη δυνατότητα μεταβολής των αλγορίθμων, ανεξάρτητα από τους πελάτες που τους χρησιμοποιούν. Αναγνωρίζει ότι σε πολλές εφαρμογές, υπάρχει μια γενική φιλοσοφία – στρατηγική που είναι κοινή σε πολλές περιπτώσεις, η συγκεκριμένη υλοποίηση όμως κάθε περίπτωσης είναι διαφορετική. Ουσιαστικά, υπάρχει ένας κοινός αλγόριθμος (π.χ. η ανάκτηση μιας ιστοσελίδας), η λεπτομερής όμως υλοποίηση του είναι διαφορετική σε κάθε περίπτωση (π.χ. ανάκτηση με JSoup και ανάκτηση με WebDriver) (Χατζηγεωργίου Α. (2005)).

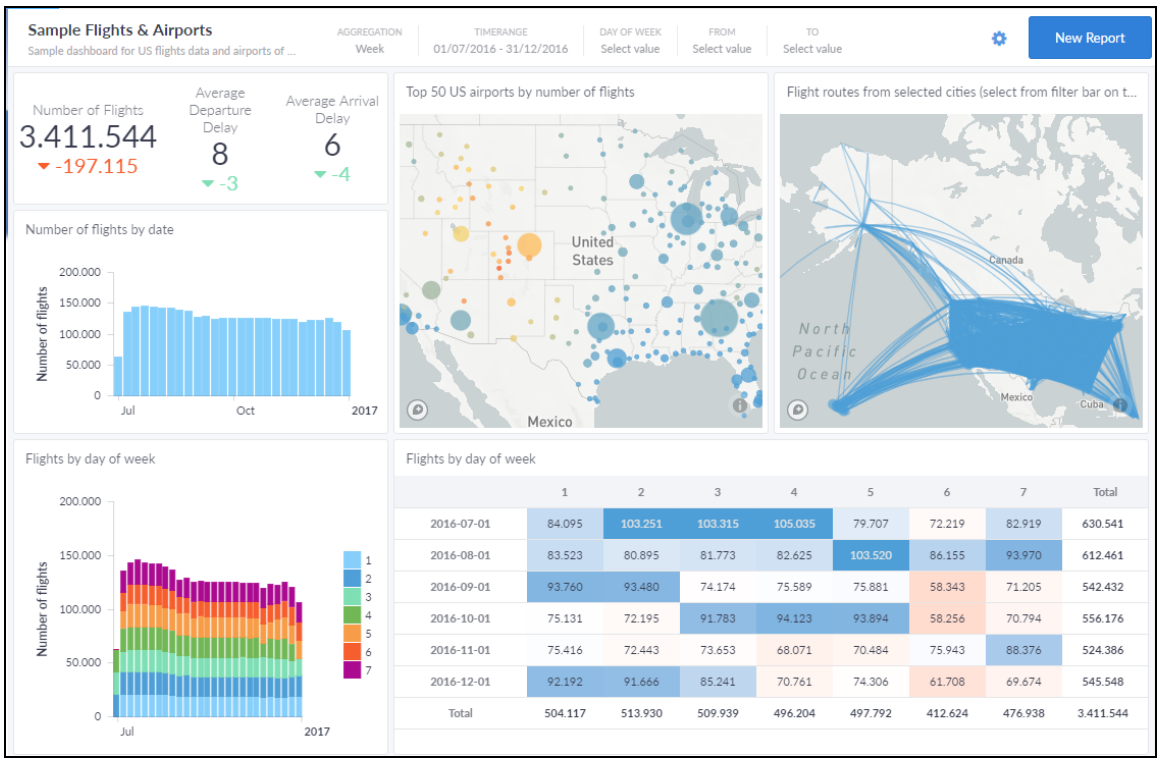
3.9.3 Μέθοδος Υπόδειγμα – Template Method

Το πρότυπο σχεδίασης "Μέθοδος – Υπόδειγμα" (Template Method) ορίζει το περίγραμμα ενός αλγορίθμου σε μια λειτουργία, αφήνοντας ορισμένα βήματα στις παράγωγες κλάσεις. Το πρότυπο επιτρέπει στις παράγωγες κλάσεις να επαναορίσουν ορισμένα βήματα του αλγορίθμου χωρίς να αλλάξουν τη δομή του. Στόχος είναι ο διαχωρισμός ενός γενικού αλγορίθμου από συγκεκριμένες υλοποιήσεις, εκμεταλλευόμενο το μηχανισμό της κληρονομικότητας (Χατζηγεωργίου Α. (2005)).

3.10 Cluvio

Το Cluvio είναι μια πολύ διαδεδομένη πλατφόρμα ανάλυσης μετρικών που χρησιμοποιείται από πολλές εταιρίες και από ομάδες στις οποίες οι αποφάσεις στηρίζονται στα δεδομένα. Με την χρήση SQL και R, η πλατφόρμα αυτή συνδυάζει τη δύναμη της ανάλυσης με όμορφα γραφικά και φιλική προς το χρήστη διεπαφή.

Ο τρόπος λειτουργίας της είναι απλός. Το μόνο που χρειάζεται είναι να συνδέσει κάποιος μια βάση δεδομένων. Η πλατφόρμα υποστηρίζει όλες τις δημοφιλείς SQL βάσεις, συμπεριλαμβανομένης της MySQL. Επόμενο βήμα είναι η δημιουργία μιας αναφοράς με τη χρήση ενός ερωτήματος SQL ή ενός σεναρίου (script) σε R και η προβολή της με χρήση ενός οπτικού διαγράμματος από την πληθώρα επιλογών της πλατφόρμας. Ο επεξεργαστής ερωτημάτων της πλατφόρμας υποστηρίζει αυτόματη συμπλήρωση, έντονη γραφή της σύνταξης, παραμετροποίηση και προτάσεις διαγραμμάτων. Τέλος, οι διάφορες αναφορές μπορούν να συγκεντρωθούν σε ένα κεντρικό dashboard (Cluvio (2019); Capterra (2019)).



Εικόνα 3-2: Παράδειγμα ενός dashboard στην πλατφόρμα του Cluvio (Cluvio (2019))

4 Ανάπτυξη εφαρμογής

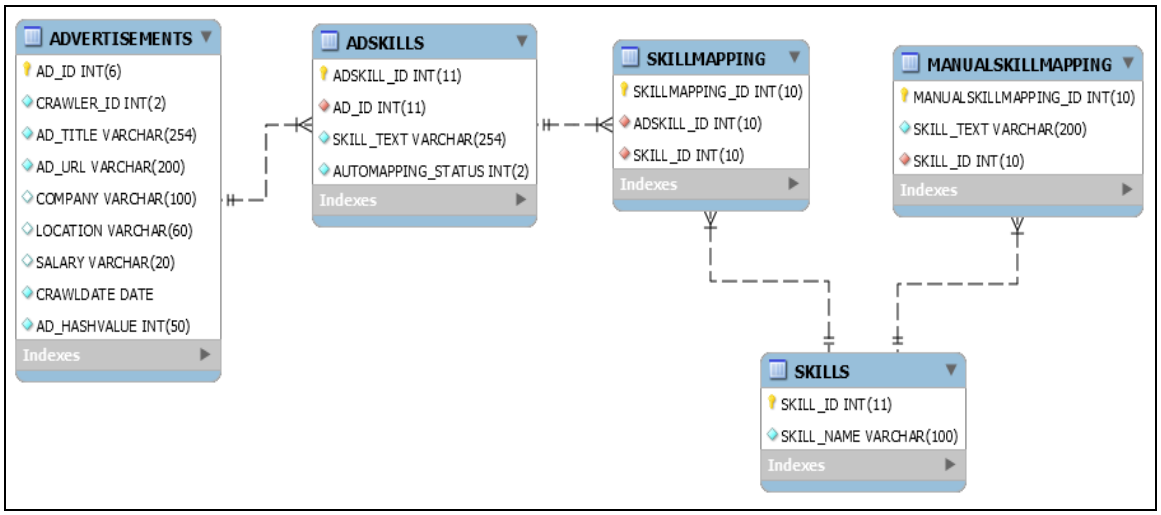
Με βάση την βιβλιογραφική μελέτη έγινε η προσπάθεια ανάπτυξης μιας εφαρμογής, η οποία θα δίνει την δυνατότητα στον χρήστη να πραγματοποιεί την αυτόματη ανάκτηση δεδομένων από ιστοσελίδες αγγελιών, να χαρτογραφεί τις δεξιότητες των αγγελιών, είτε αυτόματα είτε χειροκίνητα, σε κάποιες βασικές κατηγορίες δεξιοτήτων και τελικά να εφαρμόζει ερωτήματα για εξαγωγή συμπερασμάτων από την συλλογή των δεδομένων αυτών.

Η υλοποίηση αποτελείται από τρία μέρη:

- Μια μονάδα λογισμικού για την υλοποίηση της ανάκτησης των αγγελιών και την εισαγωγή τους στην βάση (Crawling module)
- Μια μονάδα λογισμικού για την υλοποίηση της αυτόματης ή μη χαρτογράφησης των δεξιοτήτων (Mapping module)
- Μια δωρεάν πλατφόρμα ανάλυσης μετρικών (Cluvio)

Εικόνα 4-1: Η γενική ιδέα πίσω από την υλοποίηση

Οι δύο μονάδες λογισμικού αναπτύχθηκαν σε Java με την χρήση του Eclipse και σε αυτές χρησιμοποιήθηκαν όλες οι τεχνολογίες που παρουσιάστηκαν στο προηγούμενο κεφάλαιο (Selenium, Jsoup, XPath, JAXB, HTMLCleaner). Για την υλοποίηση της βάσης δεδομένων χρησιμοποιήθηκε η MySQL. Παρακάτω φαίνεται το διάγραμμα οντοτήτων συσχετίσεων που δημιουργήθηκε στην παρούσα υλοποίηση.



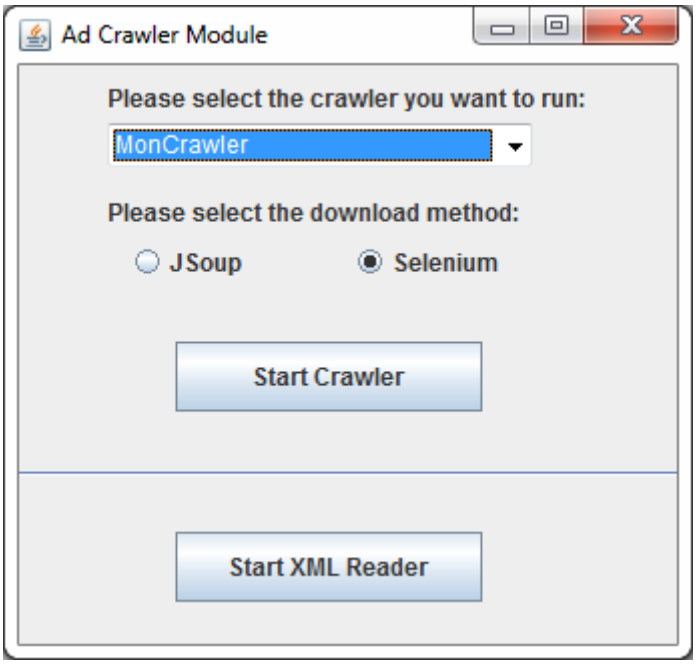
Εικόνα 4-2: Διάγραμμα οντοτήτων συσχετίσεων

Κατά την υλοποίηση της βάσης δεδομένων λήφθηκαν υπόψιν τα εξής:

- Οι αγγελίες με τις κύριες πληροφορίες τους αποθηκεύονται σε ένα πίνακα (Advertisements)
- Κάθε αγγελία είναι μοναδική και θα πρέπει να αποθηκεύεται μία φορά. Για το λόγο αυτό δημιουργείται ένα Hash Value για κάθε αγγελία στο οποίο μπαίνει περιορισμός μοναδικότητας στη βάση (Unique Key Constraint)
- Κάθε αγγελία μπορεί να περιέχει πολλές απαιτούμενες δεξιότητες. Αυτές αποθηκεύονται στον πίνακα Adskills.
- Οι δεξιότητες κάθε αγγελίας χαρτογραφούνται κάτω από κάποιες γενικές κατηγορίες δεξιοτήτων (πίνακας Skills) μέσω του πίνακα SkillMapping. Ο λόγος είναι ότι κάθε δεξιότητα σε μια αγγελία μπορεί να πρέπει να χαρτογραφηθεί με περισσότερες από μια βασικές δεξιότητες.
- Για κάθε κατηγορία δεξιοτήτων του πίνακα Skills, μπορούν να δημιουργηθούν χειροκίνητα πολλαπλές συνδέσεις. Για παράδειγμα, η δεξιότητα “Team Spirit”, “Team Player” και “Working well in a team” μπορεί να χαρτογραφηθεί στην βασική δεξιότητα “Teamwork”. Αυτό επιτυγχάνεται μέσω του πίνακα ManualSkillMapping.

4.1 Crawling module

Η μονάδα λογισμικού που είναι υπεύθυνη για το crawling αποτελείται από μια γραφική διεπαφή όπως φαίνεται στην Εικόνα 4-3. Ο χρήστης μπορεί να επιλέξει ανάμεσα στους διαθέσιμους crawlers των διαφόρων ιστοσελίδων, όπως και με ποια μέθοδο θα γίνει η ανάκτηση των ιστοσελίδων. Πατώντας το κουμπί "Start Crawler" ξεκινάει η εκτέλεση του επιλεγμένου Crawler από την οποία θα δημιουργηθεί το XML αρχείο με τα δεδομένα των αγγελιών που συλλέχθηκαν. Αφότου έχει ολοκληρωθεί η διαδικασία του crawling, μπορεί με το πάτημα του κουμπιού "Start XML Reader" να γίνει η εκτέλεση του XML Reader που είναι υπεύθυνος για την εισαγωγή των δεδομένων των XML αρχείων στην βάση δεδομένων.



Εικόνα 4-3: Γραφική διεπαφή crawling module

Για την παραμετροποίηση της εκτέλεσης αυτής της μονάδας λογισμικού όπως και κάθε crawler, χρησιμοποιείται ένα αρχείο app.config. Εκεί μπορούν να δηλωθούν παράμετροι όπως ο χρόνος αναμονής ανάμεσα σε κάθε ανάκτηση ιστοσελίδας, το directory στο οποίο βρίσκονται αρχεία απαραίτητα για την εκτέλεση, όπως και η ιστοσελίδα σπόρος κάθε crawler. Το αρχείο αυτό έχει την παρακάτω μορφή.

```
#General crawling configuration
crawl.politenessDelay=2000
crawl.maxSkillLength=60
crawl.logDir=C:\\temp\\Thesis\\
crawl.outputDir=C:\\temp\\Thesis\\
crawl.firefoxdir=./geckodriver.exe
crawl.useLocalFile=false
crawl.saveToFile=false

#Connect with DB
db.url=remotemysql.com:3306/9GnGyC9rwX
db.username=…
db.password=…

#XML directory
xml.directory=C:\\temp\\Thesis\\

#MonCrawler configuration
MonCrawler.baseUrl=https://www.monexample.co.uk/

#JSCrawler configuration
JSCrawler.baseUrl= https://www.jsexample.co.uk/
```

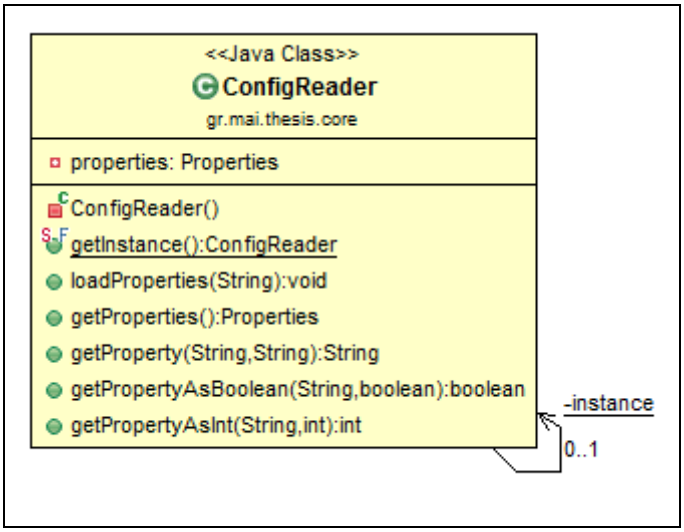
Για την ανάκτηση των παραμέτρων αυτών και την χρήση τους στα διάφορα σημεία του κώδικα δημιουργήθηκε η κλάση ConfigReader. Για την υλοποίηση της κλάσης χρησιμοποιήθηκε το πρότυπο σχεδίασης “Μοναδιαίο” (Singleton). Ο λόγος είναι πως οι παράμετροι αυτοί είναι επιθυμητό να εισάγονται μία φορά κατά την εκκίνηση της εκτέλεσης του module και να βρίσκονται σε ένα μοναδικό αντικείμενο της κλάσης ConfigReader το οποίο θα βοηθάει στην διαχείριση των υπόλοιπων αντικειμένων της μονάδας λογισμικού. Παρακάτω παρουσιάζεται η προγραμματιστική υλοποίηση.

```
public class ConfigReader {

	private static ConfigReader instance;
	private Properties properties;

	private ConfigReader() {
		properties = new Properties();
	}

	public final static ConfigReader getInstance() {
		if (instance == null) {
			instance = new ConfigReader();
		}
		return instance;
	}

	public void loadProperties(String propsFileName) {
		InputStream is;
		try {
			is = new FileInputStream(propsFileName);
```

```
                properties.load(is);
            }catch (FileNotFoundException e) {
                e.printStackTrace();
            }catch (IOException e) {
                e.printStackTrace();
            }
        }

        public String getProperty(String name, String defaultValue) {
            String prop = properties.getProperty(name);

            if (prop != null && !prop.isEmpty()) {
                return prop;
            }
            return defaultValue;
        }
}
```

Με την εκκίνηση του module γίνεται η κλήση της getInstance() που αρχικοποιεί το αντικείμενο και φορτώνονται οι παράμετροι του αρχείου app.config. Κάθε επόμενη κλήση της getInstance θα επιστρέψει το αντικείμενο-διαχειριστή από το οποίο μέσω της μεθόδου getProperty θα μπορεί να γίνει η ανάκτηση κάθε επιθυμητής παραμέτρου.

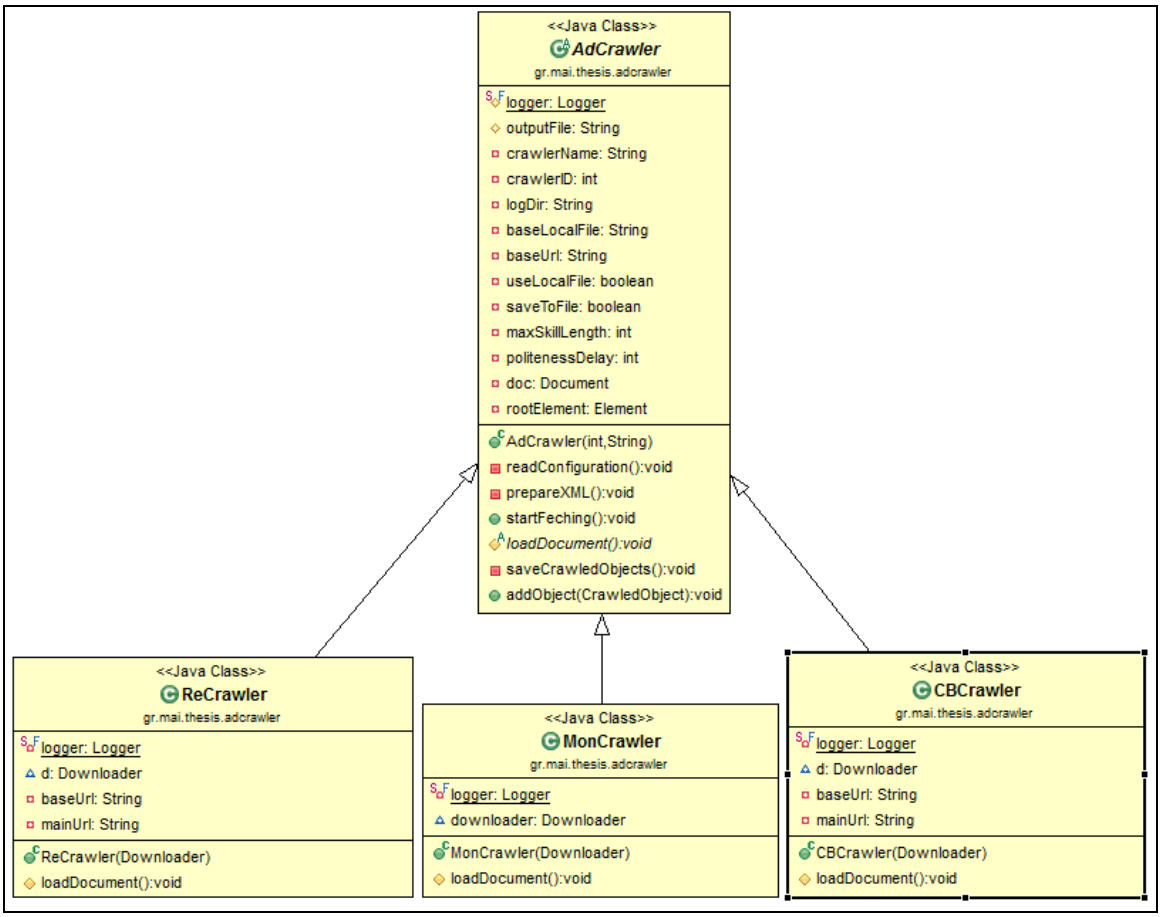
```
ConfigReader.getInstance().loadProperties("app.config");
```



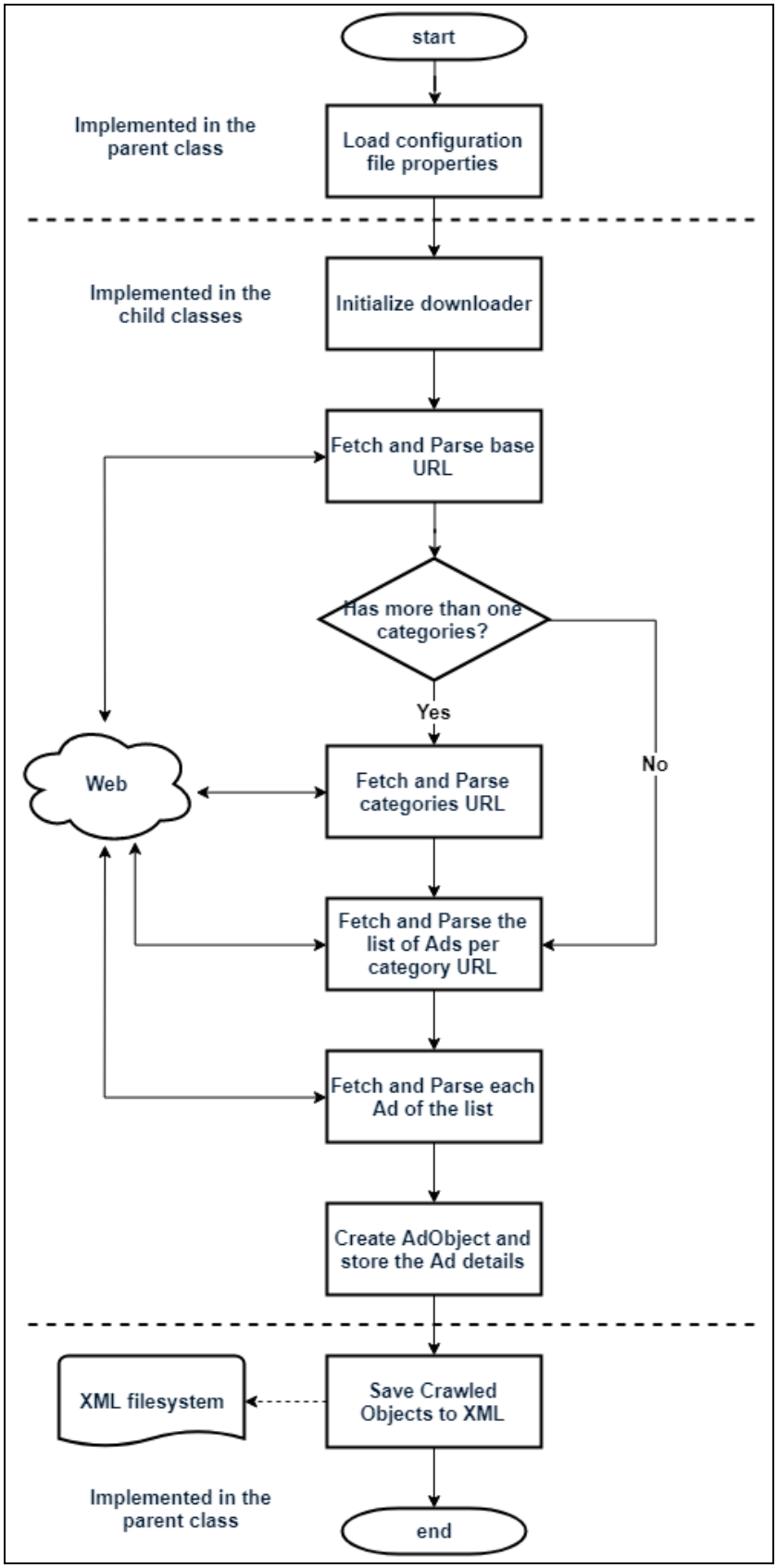
Εικόνα 4-4: UML υλοποίησης του προτύπου Μοναδιαίο

4.1.1 Υλοποίηση του crawler

Ο αλγόριθμος που χρησιμοποιείται για τον crawler έχει την ροή της Εικόνας 4-6. Για την υλοποίηση, χρησιμοποιήθηκε το πρότυπο σχεδίασης "Μέθοδος Υπόδειγμα" (Template Method). Δημιουργήθηκε μια υπερκλάση η AdCrawler, που περιλαμβάνει το περίγραμμα του αλγορίθμου. Οι κλάσεις που την επεκτείνουν είναι προσαρμοσμένες στην εκάστοτε ιστοσελίδα. Ο λόγος είναι πως η εκκίνηση του crawler και η αποθήκευση των δεδομένων είναι ίδια για όλους. Επειδή όμως κάθε ιστοσελίδα έχει την δική της δομή, το κομμάτι της ανάκτησης και της επεξεργασίας πραγματοποιείται στις υποκλάσεις της AdCrawler.



Εικόνα 4-5: UML υλοποίησης του προτύπου Μέθοδος Υπόδειγμα



Εικόνα 4-6: Διάγραμμα ροής του αλγορίθμου ανάκτησης δεδομένων από ιστοσελίδες.

Αρχικά, με την δημιουργία ενός αντικειμένου AdCrawler γίνεται η αρχικοποίηση των ιδιοτήτων που βρίσκονται στο αρχείο ρυθμίσεων app.config. Επίσης γίνεται η προετοιμασία για την τελική αποθήκευση σε XML.

```
public AdCrawler(int crawlerID, String crawlerName) {

	this.crawlerID = crawlerID;
	this.crawlerName = crawlerName;

	readConfiguration();
	prepareXML();
}

private void readConfiguration() {
	ConfigReader configReader = ConfigReader.getInstance();

	baseUrl = configReader.getProperty(crawlerName + ".baseUrl", baseUrl);

	outputFile = configReader.getProperty("crawl.outputDir", outputFile);
	outputFile = outputFile + crawlerName + ".xml";

	logDir = configReader.getProperty("crawl.logDir", logDir);
	baseLocalFile = logDir + "base.html";

	maxSkillLength = configReader.getPropertyAsInt("crawl.maxSkillLength",
maxSkillLength);
	politenessDelay =configReader.getPropertyAsInt("crawl.politenessDelay",
politenessDelay);

}

private void prepareXML() {
	try {
		DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
		DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
		doc = dBuilder.newDocument();

		// root element
		rootElement = doc.createElement("ads");
		rootElement.setAttribute("crawlerID", crawlerID + "");
		rootElement.setAttribute("crawlingDate", new
SimpleDateFormat("dd.MM.yyyy.HH:mm:ss").format(new java.util.Date()));
		doc.appendChild(rootElement);

	}
	catch (ParserConfigurationException e) {
		logger.warn("Unable to create Document Builder: {}",
e.getMessage());
	}
	catch (Exception e) {
		logger.warn("Unknown Exception", e);
	}
}
```

Η εκκίνηση της ανάκτησης γίνεται με την μέθοδο startFetching(), που αποτελεί και την μέθοδο υπόδειγμα και περιλαμβάνει την κλήση στις μεθόδους loadDocument() και saveCrawledObjects(). Η πρώτη εξ' αυτών είναι η μέθοδος με την οποία ξεκινάει το μεταβλητό κομμάτι του αλγορίθμου και υλοποιείται στις υποκλάσεις. Η δεύτερη είναι υπεύθυνη να αποθηκεύσει τα δεδομένα που ανακτήθηκαν και θα τρέξει με την ολοκλήρωση της λειτουργίας της υποκλάσης.

```
public void startFeching() {

	loadDocument();

	saveCrawledObjects();
}

protected abstract void loadDocument();

private void saveCrawledObjects() {
	try {
	// for output to file, console
	TransformerFactory transformerFactory=TransformerFactory.newInstance();
	Transformer transformer = transformerFactory.newTransformer();

	// for pretty print
	transformer.setOutputProperty(OutputKeys.INDENT, "yes");
	DOMSource source = new DOMSource(doc);

	StreamResult file = new StreamResult(new File(outputFile));
	transformer.transform(source, file);
	logger.warn("DONE!!!");

	}
	catch (TransformerConfigurationException e) {
	logger.warn("Unable to create Transform: {}", e.getMessage());
	}
	catch (TransformerException e) {
	logger.warn("Error while transforming XML doc - DOMSource - to file:
{}", e.getMessage());
	}
	catch (Exception e) {
	logger.warn("Unknown Exception: {}", e);
	}
}
```

Μέσω της μεθόδου loadDocument() δίνεται η δυνατότητα στις υποκλάσεις να ξεκινήσουν την δική τους υλοποίηση. Παρακάτω δίνεται ένα παράδειγμα το πώς γίνεται αυτό σε έναν από τους crawlers που υλοποιήθηκαν. Γίνεται η εκκίνηση του αντικειμένου ανάκτησης και γίνεται η πρόσβαση στην ιστοσελίδα σπόρο (baseUrl) που για κάθε crawler δηλώνεται εντός του αρχείου ρυθμίσεων.

```
public class MonCrawler extends AdCrawler {

	Downloader downloader;

	public MonCrawler(Downloader d) {
		super(1, "MonCrawler");

		this.downloader = d;
		d.setPolitenessDelay(this.getPolitenessDelay());
	}

	@Override
	protected void loadDocument() {
		String baseHtml = null;
		try {

			downloader.openBrowser();

			baseHtml = downloader.download(this.getBaseUrl());

			if (baseHtml != null)
				parseCategories(baseHtml);

		}
		catch (Exception e) {
			logger.warn("Unknown Exception: {}", e);
		}
		finally {
			downloader.closeBrowser();
		}

	}
	// ... parsing continues

}
```

4.1.2 Υλοποίηση της ανάκτησης (Fetching)

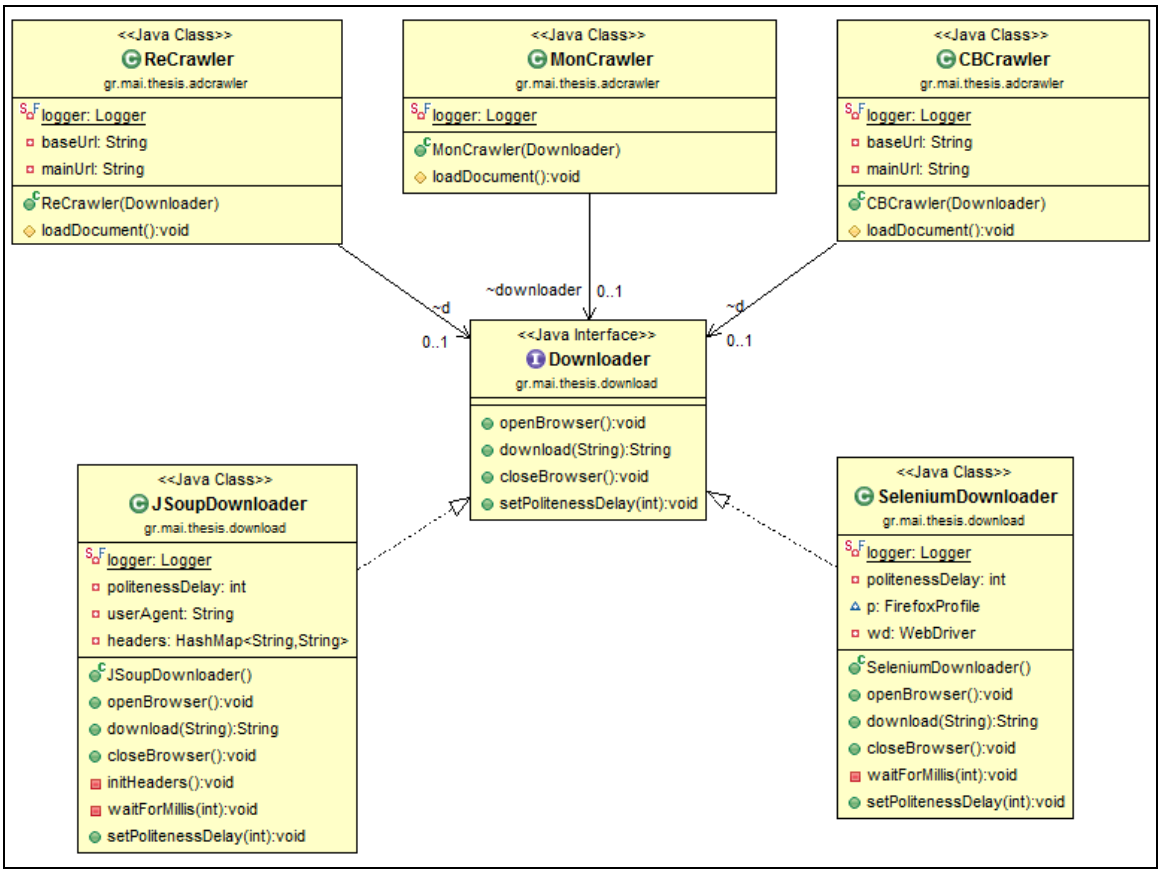
Με βάση την βιβλιογραφική έρευνα χρησιμοποιήθηκαν δύο εργαλεία για την ανάκτηση των ιστοσελίδων. Αυτά είναι η βιβλιοθήκη JSOUP και το WebDriver API που αποτελεί μέρος του Selenium. Για την υλοποίηση αυτής της λειτουργικότητας χρησιμοποιήθηκε το πρότυπο σχεδίασης «Στρατηγική». Δημιουργήθηκε γι' αυτό το λόγο μια κλάση διασύνδεσης (Interface) με το όνομα Downloader η οποία περιλαμβάνει τη δήλωση των μεθόδων που χρησιμοποιεί ο κάθε crawler για την ανάκτηση των ιστοσελίδων. Οι κλάσεις JSoupDownloader και SeleniumDownloader υλοποιούν τις μεθόδους αυτής της διασύνδεσης. Παρακάτω φαίνεται πως γίνεται αυτό στον κώδικα της εφαρμογής. Από την πλευρά του, ένας crawler έχει ένα αντικείμενο Downloader χωρίς

να τον ενδιαφέρει ποια από τις δυο υλοποιήσεις χρησιμοποιείται. Απλά καλεί την μέθοδο download() για να ανακτήσει το περιεχόμενο της ιστοσελίδας.

```
public interface Downloader {

	public abstract void openBrowser();

	public abstract String download(String url);

	public void closeBrowser();

	public void setPolitenessDelay(int politenessDelay);

}
```

```
public class JSoupDownloader implements Downloader {
	// define variables
	// ...
	public JSoupDownloader() {
		headers = new HashMap<String, String>();
		initHeaders();
	}
	public String download(String url) {
		String html = null;

		try {
		logger.warn("Processing: {}", url);

		html = Jsoup.connect(url).userAgent(userAgent).header("Accept",
headers.get("Accept")).header("Connection",headers.get("Connection")).header("
Accept-Language", headers.get("Accept-Language")).header("Accept-Encoding",
headers.get("Accept-Encoding")).header("Accept-Charset", headers.get("Accept-
Charset")).get().html();

		if (politenessDelay > 0)
			waitForMillis(politenessDelay);

		}
		catch (Exception e) {
			logger.warn("Unknown Exception", e);
		}
		return html;
	}

	private void initHeaders() {
		userAgent = "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:66.0)
Gecko/20100101 Firefox/66.0";
```

```
            headers.put("Accept",
"text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0
.8,image/png,*/*;q=0.8");
            headers.put("Connection", "keep-alive");
            headers.put("Accept-Language", "en-us,en;q=0.5");
            headers.put("Accept-Encoding", "none");
            headers.put("Accept-Charset", "UTF-8,*");

        }
        // ...
}
```



Εικόνα 4-7: UML υλοποίησης του προτύπου Στρατηγική

Όπως αναφέρεται στην βιβλιογραφική μελέτη, για να αποφευχθεί το μπλοκάρισμα από τους παρόχους, η αυτοματοποιημένη περιήγηση θα πρέπει να μοιάζει όσο το δυνατόν σε αυτήν ενός ανθρώπου. Όπως αναφέρεται το βασικό είναι να θέσουμε στο πρόγραμμα τον User Agent ενός browser και τις κατάλληλες κεφαλίδες. Το Jsoup προσφέρει αυτήν την λειτουργικότητας με τις μεθόδους userAgent() και header(). Όπως θα δούμε παρακάτω, κάτι τέτοιο δεν είναι αναγκαίο στην περίπτωση του Selenium μιας και εκεί ανοίγει ένας browser. Για το λόγο αυτό στην κλάση SeleniumDownloader

δηλώνεται και το path όπου η εφαρμογή θα βρει το εκτελέσιμο του browser με τη κλήση της System.setProperty(), ενώ υλοποιούνται και οι μέθοδοι openBrowser() και closeBrowser(). Και στις δύο περιπτώσεις προστίθεται μια αναμονή ανάμεσα στα downloads ώστε να αποφευχθεί η ανάκτηση με πολύ γρήγορο ρυθμό, όπως προτείνεται και στην βιβλιογραφία. Αυτό γίνεται μέσω της μεταβλητής politenessDelay. Η ρύθμιση του ύψους της αναμονής ορίζεται εντός του αρχείου app.config και περνιέται στον Downloader κατά την εκκίνηση του crawler.

```
public class SeleniumDownloader implements Downloader {
	// define variables
	// ...
	public void openBrowser() {

				System.setProperty("webdriver.gecko.driver",
ConfigReader.getInstance().getProperty("crawl.firefoxdir",
"./geckodriver.exe"));
		wd = new FirefoxDriver();

	}

	public String download(String url) {
		String html = null;

		try {
			logger.warn("Processing: {}", url);
			wd.get(url);
			html = wd.getPageSource();

			if (politenessDelay > 0)
				waitForMillis(politenessDelay);

		}
		catch (Exception e) {
			logger.warn("Exception", e);
		}
		return html;
	}

	public void closeBrowser() {
		if (wd != null)
			wd.quit();

	}

	// ...
}
```

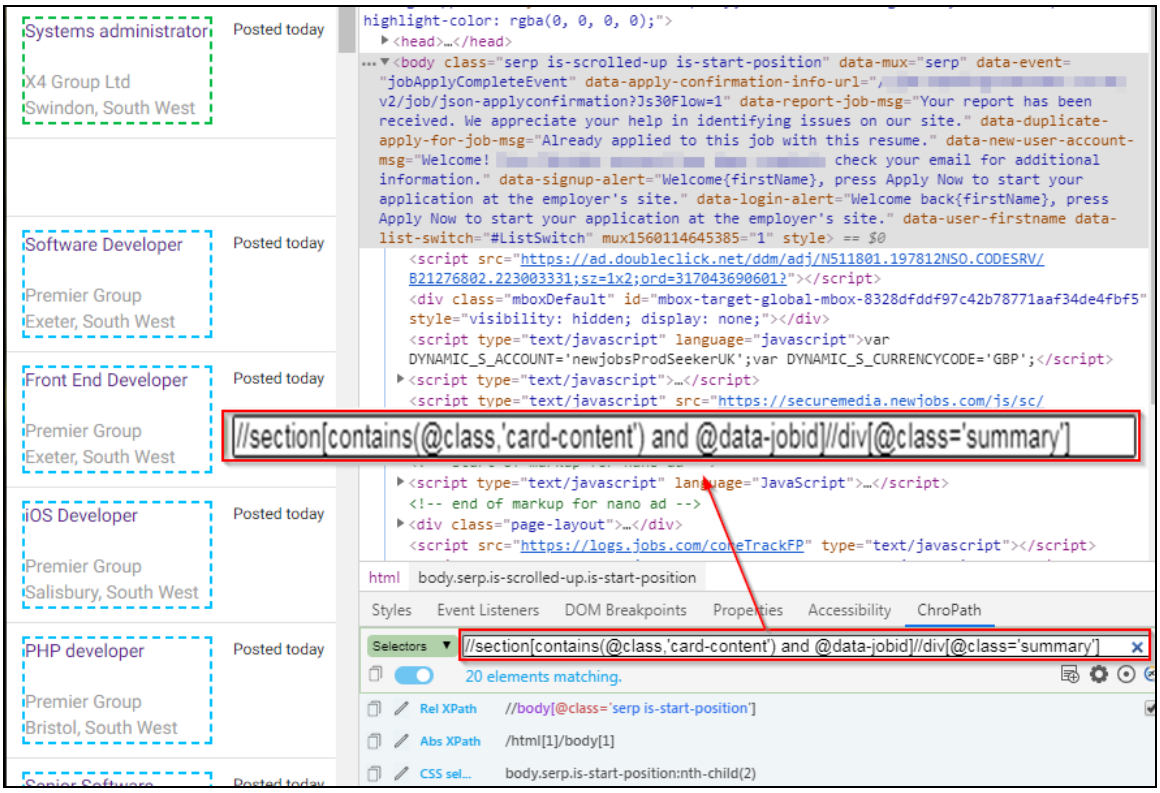
4.1.3 Υλοποίηση της ανάλυσης (Parsing)

Με την ολοκλήρωση της ανάκτησης μιας ιστοσελίδας μπορεί να ξεκινήσει η ανάλυση της για την εξαγωγή των επιθυμητών πληροφοριών. Στην παρούσα υλοποίηση αυτό γίνεται με τη χρήση ερωτημάτων XPath. Αρχικά το έγγραφο Html περνάει μια προ-επεξεργασία (περισσότερα για αυτό στο επόμενο κεφάλαιο) για τον καθαρισμό του και για την μετατροπή του σε ένα δένδρο κόμβων σε μορφή XML. Στη συνέχεια γίνεται σε αυτό η υποβολή ενός ή περισσοτέρων ερωτημάτων XPath και λαμβάνονται οι κόμβοι που περιέχουν την επιθυμητά δεδομένα. Διατρέχοντας τους κόμβους αυτούς εξάγονται τα δεδομένα από τους κόμβους και αποθηκεύονται σε ένα αντικείμενο της κλάσης AdObject η οποία και θα αναλυθεί σε επόμενο κεφάλαιο.

Τα δεδομένα των αγγελιών τα οποία επιχειρούνται να ανακτηθούν σε κάθε crawler είναι:

- Ο τίτλος της θέσης της αγγελίας
- Η εταιρεία που δημοσιεύει την αγγελία (μπορεί να είναι η ίδια εταιρεία που θα κάνει την πρόσληψη ή κάποιος recruiter)
- Η τοποθεσία της εταιρείας όπου βρίσκεται η θέση
- Η ιστοσελίδα της αγγελίας
- Ο μισθός, εάν υπάρχει
- Το σύνολο των δεξιοτήτων που απαιτεί η θέση, με την προϋπόθεση ότι αυτά βρίσκονται καλά δομημένα εντός της αγγελίας.

Μέσω των εργαλείων που παρέχονται από τους φυλλομετρητές ιστού, μπορεί να γίνει η ανάλυση της HTML μιας ιστοσελίδας και η εφαρμογή ερωτημάτων XPath σε αυτήν. Ένα παράδειγμα δίνεται στην Εικόνα 4-8.



Εικόνα 4-8: Εύρεση XPath για την ανάκτηση κόμβων αγγελιών

Συνεχίζοντας το παράδειγμα του crawler του προηγούμενου κεφαλαίου και με βάση το διάγραμμα που παρουσιάστηκε στην Εικόνα 4-6, το parsing ξεκινάει με την σελίδα των κατηγοριών των αγγελιών και συνεχίζει στην ανάκτηση και ανάλυση της λίστας των αγγελιών.

```
private void parseCategories(String baseHtml) {
	String categoryHtml = null;
	String category;
	String categoryUrl;

	// Get list of categories
	NodeList nList=(NodeList)
Util.getXpath(Util.HTMLCleanerToDoc(baseHtml), "//ul[@class='card-
columns']/li/h3/a", XPathConstants.NODESET);

	for (int i = 0; i < nList.getLength(); i++) {
		Element elem = (Element) nList.item(i);
		category = elem.getTextContent();
		categoryUrl = elem.getAttribute("href");

		categoryHtml = downloader.download(categoryUrl);

		if (categoryHtml != null)
			parseAdList(categoryHtml);
	}
```

```
}

private void parseAdList(String categoryHtml) {

	String adHtml = null;
	String adTitle = "";
	String adUrl = "";
	String company = "";
	String location = "";

	// Get list of ads
	NodeList nList = (NodeList)
Util.getXpath(Util.HTMLCleanerToDoc(categoryHtml),"//section[contains(@class,'
card-content') and @data-jobid]//div[@class='summary']",
XPathConstants.NODESET);
	for (int i = 0; i < nList.getLength(); i++) {
		Element elem = (Element) nList.item(i);
		adTitle = Util.getSubNodeValue(elem, "h2", "class",
"title").trim();
		adUrl = Util.getSubNodeAttributeValue(elem, "a", "href").trim();
		company = Util.getSubNodeValue(elem, "div", "class",
"company").trim();
		location = Util.getSubNodeValue(elem, "div", "class",
"location").trim();

		adHtml = downloader.download(adUrl);

		if (adHtml != null)
			parseAd(adHtml, adTitle, adUrl, company, location);
	}
}
```

Με αυτόν τον τρόπο ο crawler ανιχνεύει τις ιστοσελίδες των αγγελιών και μπορεί να προχωρήσει στην εξαγωγή των δεξιοτήτων και την αποθήκευση τους.

4.1.4 Υλοποίηση της προ-επεξεργασίας

Προτού γίνει η παρουσίαση του τρόπου που αποθηκεύεται μια αγγελία, αξίζει να γίνει αναφορά στην προ-επεξεργασία που υπόκεινται τα δεδομένα που ανακτήθηκαν προτού αποθηκευτούν. Μια αρχική επεξεργασία γίνεται κατά την μετατροπή από HTML σε XML που παρουσιάστηκε παραπάνω. Σε αυτό το σημείο (μέθοδος HTMLCleanerToString()) πραγματοποιείται ένας αρχικός καθαρισμός του HTML κώδικα με την χρήση της μεθόδου clean() του HtmlCleaner, όπως παρουσιάστηκε στην βιβλιογραφία.

Επιπλέον, χρησιμοποιώντας μια από τις λίστες των πιο δημοφιλών Stopwords της αγγλικής γραμματικής δημιουργήθηκε μια μέθοδος removeStopWords για την αφαίρεση

τους όπου είναι αναγκαίο. Συγκεκριμένα για το πεδίο των αγγελιών που περιέχει τον μισθό, δημιουργήθηκε η μέθοδος fixSalary, για να αφαιρέσει τυχόν γράμματα που υπάρχουν στον μισθό ή τον χαρακτήρα της παύλας κ.α. (γενικά τους μη αριθμητικούς χαρακτήρες). Τέλος δημιουργήθηκε η μέθοδος makeNice ώστε να κάνει το κείμενο πιο ευανάγνωστο, αφαιρώντας επιπλέον και αχρείαστα κενά που υπάρχουν εντός ενός πεδίου κειμένου, αφαιρώντας HTML tags.

4.1.5 Η κλάση AdObject και η μέθοδος addObject

Για την αποθήκευση των δεδομένων κάθε αγγελίας δημιουργήθηκε η κλάση AdObject που περιλαμβάνει ένα πεδίο για κάθε κομμάτι της αγγελίας που ανακτάται και παρουσιάστηκε στο προηγούμενο κεφάλαιο. Επίσης, καθώς τα δεδομένα θα πρέπει να αποθηκευτούν σε μορφή XML, γίνεται η δήλωση μέσω των JAXB annotations στα αντίστοιχα πεδία, για το ποια από αυτά θα πρέπει ο marshaler να μετατρέψει σε XML.

```
@XmlRootElement
public class AdObject extends CrawledObject {

	private String adTitle, adUrl, company, location, salary, adTextBody;
	private ArrayList<String> skills;

	public AdObject() {
		super();
		skills = new ArrayList<String>();
	}

	@XmlElement
	public String getAdTitle() {
		return adTitle;
	}

	public void setAdTitle(String adTitle) {
		this.adTitle = adTitle;
	}

	@XmlElement
	public String getAdUrl() {
		return adUrl;
	}

	public void setAdUrl(String adUrl) {
		this.adUrl = adUrl;
	}

	@XmlElement
```

```
	public String getCompany() {
		return company;
	}

	public void setCompany(String company) {
		this.company = company;
	}

	@XmlElement
	public String getLocation() {
		return location;
	}

	public void setLocation(String location) {
		this.location = location;
	}

	@XmlElement
	public String getSalary() {
		return salary;
	}

	public void setSalary(String salary) {
		this.salary = salary;
	}

	@XmlElement
	public String getAdTextBody() {
		return adTextBody;
	}

	public void setAdTextBody(String adTextBody) {
		this.adTextBody = adTextBody;
	}

	public ArrayList<String> getSkills() {
		return skills;
	}

	public void addToSkills(String skill) {
		skills.add(skill);
	}

}
```

Έτσι, επιστρέφοντας στον crawler του παραδείγματος, γίνεται η αποθήκευση των δεδομένων που ανακτήθηκαν στο αντικείμενο της κλάσης AdObject. Για την προσθήκη του αντικειμένου αυτού σε μορφή XML στο προς αποθήκευση Document της μητρικής κλάσης (δείτε μέθοδο prepareXML σε προηγούμενο κεφάλαιο) χρησιμοποιείται σαν τελικό βήμα η μέθοδο addObject.

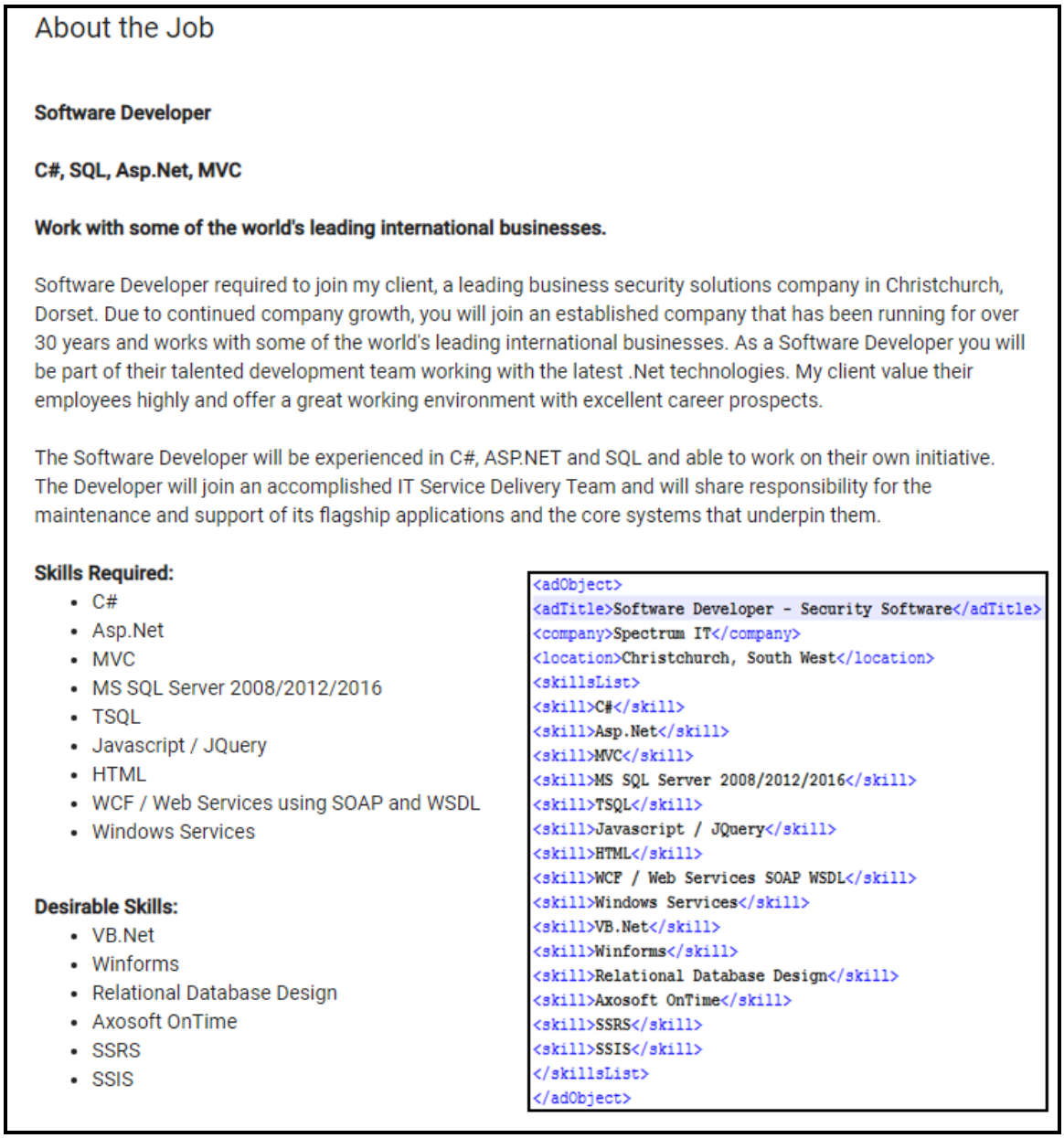
```
private void parseAd(String adHtml, String adTitle, String adUrl, String
company, String location) {

	String salary, jobBody;
	AdObject ao = new AdObject();

	ao.setAdTitle(adTitle);
	ao.setAdUrl(adUrl);
	ao.setCompany(company);
	ao.setLocation(location);

	// Parse Salary
	Node salaryNode = (Node) Util.getXpath(Util.HTMLCleanerToDoc(adHtml),
"//div[@id='JobSalary']/section/div/div/div", XPathConstants.NODE);

	Element salaryElem = (Element) salaryNode;
	if (salaryElem != null) {
		salary = salaryElem.getTextContent().trim();
		salary = Util.removeNonDigitsFromSalary(salary);
		salary = Util.makeNice(salary);
		ao.setSalary(salary);
	}

	// Parse list of skills
	NodeList skillsList = (NodeList)
Util.getXpath(Util.HTMLCleanerToDoc(adHtml), "//div[@id='JobBody']//li",
XPathConstants.NODESET);

	if (skillsList.getLength() > 0) {
		for (int i = 0; i < skillsList.getLength(); i++) {
			Element elem = (Element) skillsList.item(i);

			String skill = Util.makeNice(elem.getTextContent());
			skill = Util.removeStopWords(skill);

			if (skill.length() < this.getMaxSkillLength())
				ao.addToSkills(skill);
		}
	}
	addObject(ao);
}
```

Η μέθοδος addObject που υλοποιείται στην υπερκλάση AdCrawler είναι και αυτή που θα μετατρέψει το αντικείμενο AdObject σε μορφή XML. Για να πραγματοποιηθεί, γίνεται η χρήση ενός αντικειμένου της κλάσης Marshaler το οποίο δημιουργείται έχοντας ως περιεχόμενο την δομή που ορίστηκε από τα annotatios της κλάσης AdObject. Ο κώδικας της μεθόδου παρουσιάζεται παρακάτω.

```
public void addObject(CrawledObject co) {
	try {
	if (co instanceof AdObject) {
		AdObject ao = (AdObject) co;
```

```
            JAXBContext adObjectContext =
JAXBContext.newInstance(AdObject.class);
            Marshaller adObjectMarshaller =
adObjectContext.createMarshaller();
            // for pretty-print XML in JAXB
            adObjectMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
Boolean.TRUE);

            adObjectMarshaller.marshal(ao, rootElement);

            Element lastChildElem=(Element) rootElement.getLastChild();

            if (ao.getSkills().size() > 0) {
                  Element skillsListElement =
doc.createElement("skillsList");
                  for (int i = 0; i < ao.getSkills().size(); i++) {

                        Element skillElem = doc.createElement("skill");
                        skillElem.setTextContent(ao.getSkills().get(i));

                        skillsListElement.appendChild(skillElem);

                  }
                  lastChildElem.appendChild(skillsListElement);
            }

      }
      }
      catch (JAXBException e) {
            logger.warn("JAXBException: {}", e);
      }
      catch (Exception e) {
            logger.warn("Unknown Exception: {}", e);
      }
}
```

Με την ολοκλήρωση της ανάκτησης όλων των αγγελιών από τις ιστοσελίδες ή σε περίπτωση που προκύψει σφάλμα σε κάποιο στάδιο, ο crawler επιστρέφει στην μέθοδο saveCrawledObjects() που βρίσκεται στην μητρική κλάση και στην οποία έγινε αναφορά στην πρώτη ενότητα αυτού του κεφαλαίου. Έτσι ολοκληρώνεται η λειτουργία του crawler και όσες αγγελίες έχουν ανακτηθεί από έναν crawler αποθηκεύονται σε ένα XML σε ένα directory που καθορίζεται από το αρχείο ρυθμίσεων. Παρακάτω δίνεται ένα παράδειγμα μιας αγγελίας και πως αυτή αποθηκεύεται στο XML (Εικόνα 4-9).



About the Job

Software Developer

C#, SQL, Asp.Net, MVC

Work with some of the world's leading international businesses.

Software Developer required to join my client, a leading business security solutions company in Christchurch, Dorset. Due to continued company growth, you will join an established company that has been running for over 30 years and works with some of the world's leading international businesses. As a Software Developer you will be part of their talented development team working with the latest .Net technologies. My client value their employees highly and offer a great working environment with excellent career prospects.

The Software Developer will be experienced in C#, ASP.NET and SQL and able to work on their own initiative. The Developer will join an accomplished IT Service Delivery Team and will share responsibility for the maintenance and support of its flagship applications and the core systems that underpin them.

Skills Required:

- C#
- Asp.Net
- MVC
- MS SQL Server 2008/2012/2016
- TSQL
- Javascript / JQuery
- HTML
- WCF / Web Services using SOAP and WSDL
- Windows Services

Desirable Skills:

- VB.Net
- Winforms
- Relational Database Design
- Axosoft OnTime
- SSRS
- SSIS

```
<adObject>
<adTitle>Software Developer - Security Software</adTitle>
<company>Spectrum IT</company>
<location>Christchurch, South West</location>
<skillsList>
<skill>C#</skill>
<skill>Asp.Net</skill>
<skill>MVC</skill>
<skill>MS SQL Server 2008/2012/2016</skill>
<skill>TSQL</skill>
<skill>Javascript / JQuery</skill>
<skill>HTML</skill>
<skill>WCF / Web Services SOAP WSDL</skill>
<skill>Windows Services</skill>
<skill>VB.Net</skill>
<skill>Winforms</skill>
<skill>Relational Database Design</skill>
<skill>Axosoft OnTime</skill>
<skill>SSRS</skill>
<skill>SSIS</skill>
</skillsList>
</adObject>
```

Εικόνα 4-9: Ανάκτηση δεξιοτήτων από αγγελία

4.1.6 Η κλάση XMLReader

Για την εισαγωγή των δεδομένων που ανακτήθηκαν στην βάση δημιουργήθηκε η κλάση του XMLReader. Αυτή προσφέρει την δυνατότητα της προσθήκης των δεδομένων ανεξάρτητα από το πότε και από ποιόν αυτά ανακτήθηκαν. Με το πάτημα του κουμπιού της εκτέλεσης του XMLReader, δημιουργείται ένα αντικείμενο της κλάσης αυτής και ξεκινάει η ανάγνωση όλων των XML που βρίσκονται στο directory που ορίζεται εντός του app.config.

```
  public void startXMLReader() {
    openConnectionWithDB();
    readDirectory(new File(xmlDir));
  }

  private void openConnectionWithDB() {
    try {
      conn = DriverManager.getConnection(dbUrl);
      logger.warn("Connection with DB established");
    }
    catch (SQLException e) {
      logger.warn("SQL Exception: {}", e);
    }
  }

  private void readDirectory(File directory) {
    try {
      final FileNameExtensionFilter extensionFilter = new
FileNameExtensionFilter(null, "xml");
      for (final File file : directory.listFiles()) {
        if (extensionFilter.accept(file)) {

          DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
          DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
          Document doc = dBuilder.parse(file);

          parseXMLFile(doc);
        }
      }
    }
    catch (IOException e) {
      e.printStackTrace();
    }
    catch (ParserConfigurationException e) {
      e.printStackTrace();
    }
    catch (SAXException e) {
      e.printStackTrace();
    }

  }
```

Κατά την ανάκτηση των δεδομένων από τα αποθηκευμένα XML, γίνεται πάλι η χρήση του JAXB API και με την βοήθεια ενός αντικειμένου Unmarshaller οι κόμβοι AdObject εντός του XML μετατρέπονται σε αντικείμενα της κλάσης AdObject.

```
private void parseXMLFile(Document doc) {
    try {

      Node pNode = doc.getFirstChild();
      String crawlerID = Util.getAttrValue(pNode, "crawlerID");
      String cDate = Util.getAttrValue(pNode, "crawlingDate");
      DateTimeFormatter dtf =
DateTimeFormatter.ofPattern("dd.MM.yyyy.HH:mm:ss");
      LocalDate crawlingDate = LocalDate.parse(cDate, dtf);

      NodeList nList = (NodeList) Util.getXpath(doc, "//adObject",
XPathConstants.NODESET);

      JAXBContext adObjectContext = JAXBContext.newInstance(AdObject.class);
      Unmarshaller adObjectUnmarshaller= adObjectContext.createUnmarshaller();

      for (int i = 0; i < nList.getLength(); i++) {
        Node element = nList.item(i);
        NodeList nSkillsList = (NodeList) Util.getXpath(element,
".//skillsList/skill", XPathConstants.NODESET);

        if (nSkillsList.getLength() > 0) {
          AdObject ao = (AdObject) adObjectUnmarshaller.unmarshal(element);

          for(int j=0;j<nSkillsList.getLength();j++) {
            Node skillElement = nSkillsList.item(j);
            ao.addToSkills(skillElement.getTextContent());

          }

          addObjectToDB(crawlerID, crawlingDate, ao);
        }
      }
    }
    catch (JAXBException e) {
      e.printStackTrace();
    }
  }
```

Ακολούθως τα αντικείμενα που ανακτήθηκαν αποθηκεύονται με την χρήση του JDBC στην MySQL βάση. Αυτό πραγματοποιείται με τις μεθόδους addObjectToDB() και addAdSkillsToDB(). Για την αποφυγή της εισαγωγής της ίδιας αγγελίας δύο φορές η εισαγωγή γίνεται με την χρήση της SQL “INSERT IGNORE”. Με αυτήν όταν θα γίνει η προσπάθεια να εισαχθεί μια γραμμή που παραβιάζει κάποιο περιορισμό (constraint) του πίνακα, δεν θα ολοκληρωθεί και θα αγνοηθεί χωρίς να οδηγήσει σε σφάλμα Unique Constraint Violation.

```
private void addObjectToDB(String crawlerID, LocalDate crawlingDate, AdObject
ao) {

    ResultSet rs = null;
    int adId = 0;

    try {

      String insertAdSQL = "INSERT IGNORE INTO ADVERTISEMENTS(CRAWLER_ID,
AD_TITLE, AD_URL, COMPANY, LOCATION, SALARY, CRAWLDATE, AD_HASHVALUE)
VALUES(?, ?, ?, ?, ?, ?, ?, ?)";
      PreparedStatement stmt = conn.prepareStatement(insertAdSQL,
Statement.RETURN_GENERATED_KEYS);

      stmt.setInt(1, Integer.parseInt(crawlerID));
      stmt.setString(2, ao.getAdTitle());
      stmt.setString(3, ao.getAdUrl());
      stmt.setString(4, ao.getCompany());
      stmt.setString(5, ao.getLocation());
      stmt.setString(6, ao.getSalary());
      stmt.setDate(7, java.sql.Date.valueOf(crawlingDate));
      stmt.setInt(8, ao.getAdUrl().hashCode());

      int rowAffected = stmt.executeUpdate();

      if (rowAffected == 1) {
        rs = stmt.getGeneratedKeys();
        if (rs.next()) {
          adId = rs.getInt(1);

          addAdSkillsToDB(adId, ao);
        }
      }

    }
    catch (SQLException e) {
      e.printStackTrace();
    }
  }

  private void addAdSkillsToDB(int adId, AdObject ao) {
    ArrayList<String> skills = ao.getSkills();
    String insertAdSkillSQL = "INSERT INTO ADSKILLS(AD_ID, SKILL_TEXT)
VALUES(" + adId + ", ?)";

    for (int i = 0; i < skills.size(); i++) {
      try {

        PreparedStatement stmt = conn.prepareStatement(insertAdSkillSQL);
        stmt.setString(1, skills.get(i));
        stmt.execute();
      }
      catch (SQLException e) {
        e.printStackTrace();
      }

    }
  }
```

Έτσι με την ολοκλήρωση της εκτέλεσης και του XMLReader τα δεδομένα των αγγελιών εισάγονται στην βάση. Συνεχίζοντας στο παράδειγμα του προηγούμενου κεφαλαίου (Εικόνα 4-9), μπορούμε να δούμε ότι τα ανακτημένα δεδομένα έχουν εισαχθεί στην βάση (Εικόνα 4-10).

```
SELECT * FROM ADVERTISEMENTS a, ADSKILLS b WHERE a.AD_ID = b.AD_ID AND a.AD_ID = 310
```

Show all | **Restore column order** | Number of rows: 25 | Filter rows: Search this table

Options

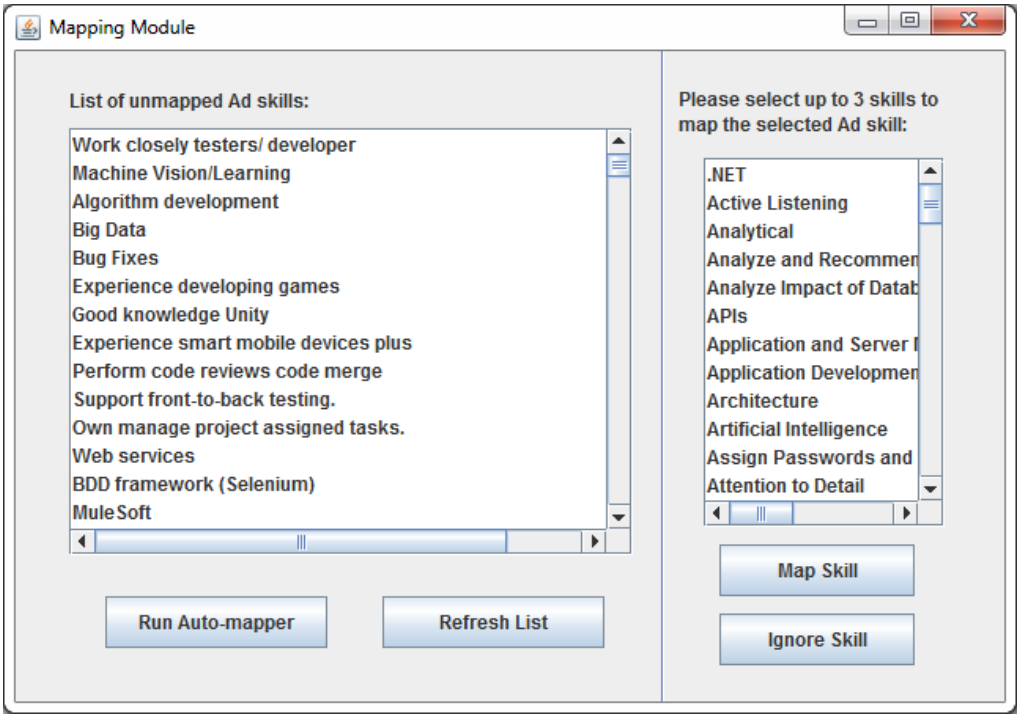
AD_ID	CRAWLER_ID	AD_TITLE	COMPANY	LOCATION	ADSKILL_ID	SKILL_TEXT
310	1	Software Developer - Security Software	Spectrum IT	Christchurch, South West	1316	C#
310	1	Software Developer - Security Software	Spectrum IT	Christchurch, South West	1317	Asp.Net
310	1	Software Developer - Security Software	Spectrum IT	Christchurch, South West	1318	MVC
310	1	Software Developer - Security Software	Spectrum IT	Christchurch, South West	1319	MS SQL Server 2008/2012/2016
310	1	Software Developer - Security Software	Spectrum IT	Christchurch, South West	1320	TSQL
310	1	Software Developer - Security Software	Spectrum IT	Christchurch, South West	1321	Javascript / JQuery
310	1	Software Developer - Security Software	Spectrum IT	Christchurch, South West	1322	HTML
310	1	Software Developer - Security Software	Spectrum IT	Christchurch, South West	1323	WCF / Web Services SOAP WSDL
310	1	Software Developer - Security Software	Spectrum IT	Christchurch, South West	1324	Windows Services
310	1	Software Developer - Security Software	Spectrum IT	Christchurch, South West	1325	VB.Net
310	1	Software Developer - Security Software	Spectrum IT	Christchurch, South West	1326	Winforms
310	1	Software Developer - Security Software	Spectrum IT	Christchurch, South West	1327	Relational Database Design
310	1	Software Developer - Security Software	Spectrum IT	Christchurch, South West	1328	Axosoft OnTime
310	1	Software Developer - Security Software	Spectrum IT	Christchurch, South West	1329	SSRS
310	1	Software Developer - Security Software	Spectrum IT	Christchurch, South West	1330	SSIS

Εικόνα 4-10: Αποθήκευση ανακτημένων δεδομένων στην βάση

4.2 Mapping module

Η μονάδα λογισμικού που είναι υπεύθυνη για το mapping αποτελείται από μια γραφική διεπαφή όπως φαίνεται στην Εικόνα 4-11. Ο χρήστης μπορεί να δει στα αριστερά μια λίστα με το ποιες δεξιότητες έχουν ανακτηθεί και οι οποίες ανήκουν σε διάφορες αγγελίες. Στην ουσία το τμήμα αυτό αποτελεί μια αναπαράσταση του πίνακα Adskills της βάσης. Δεξιά υπάρχει μια λίστα με τις βασικές κατηγορίες δεξιοτήτων όπως αυτές έχουν οριστεί στον πίνακα Skills της βάσης δεδομένων. Ο χρήστης έχει τις εξής επιλογές:

- Να τρέξει την αυτόματη χαρτογράφηση πατώντας το κουμπί “Run Auto-mapper”
- Να επιλέξει μια δεξιότητα και να την χαρτογραφήσει χειροκίνητα με έως τρεις βασικές κατηγορίες πατώντας το κουμπί “Map Skill”
- Να επιλέξει να αγνοηθεί κάποια δεξιότητα που ανακτήθηκε πατώντας το κουμπί “Ignore Skill”
- Να ανανεώσει την λίστα των δεξιοτήτων που έχουν ανακτηθεί σε περίπτωση που παράλληλα τρέχει η εισαγωγή στην βάση από το Crawling module. Αυτό γίνεται πατώντας το κουμπί “Refresh List”



Εικόνα 4-11: Γραφική διεπαφή mapping module

Για την υποστήριξη της επικοινωνίας αυτής της μονάδας λογισμικού με την βάση δεδομένων, δημιουργήθηκε η κλάση DatabaseManager. Μέσω της κλάσης αυτής δημιουργείται η σύνδεση με την βάση και εκτελούνται ερωτήματα SQL. Οι μέθοδοι που δημιουργήθηκαν είναι καθαρά για να εξυπηρετήσουν την λειτουργικότητα της γραφικής διεπαφής. Παρακάτω δίνεται ο κώδικας για την υλοποίηση της σύνδεσης και των μεθόδων της εκτέλεσης των ερωτημάτων SQL.

```
public DatabaseManager() {

	// Read configuration
	ConfigReader configReader = ConfigReader.getInstance();

	dbUrl = "jdbc:mysql://" + configReader.getProperty("db.url", null);
	dbUrl += "?user=" + configReader.getProperty("db.username", null);
	dbUrl += "&password=" + configReader.getProperty("db.password", null);

	conn = null;
}

public void openConnectionWithDB() {
	try {
		conn = DriverManager.getConnection(dbUrl);
		logger.warn("Connection with DB established");
	}
	catch (SQLException e) {
		logger.warn("SQL Exception: {}", e);
	}
}

public void closeConnection() {
	try {
		conn.close();
	}
	catch (SQLException e) {
		e.printStackTrace();
	}
}
```

Ένα παράδειγμα μεθόδου για την ανάκτηση μιας λίστας αλφαριθμητικών από την βάση δίνεται παρακάτω. Επίσης δίνεται και η μέθοδος για την απλή εκτέλεση μιας εντολής DML.

```
public ArrayList doGetStringList(String query) {

	ArrayList<String> stringList = new ArrayList<String>();

	try {
		Statement st = conn.createStatement();
		ResultSet rs = st.executeQuery(query);

		while (rs.next()) {
			String columnText = rs.getString(1);
			stringList.add(columnText);
		}
	}
	catch (SQLException e) {
		e.printStackTrace();
		closeConnection();
	}
	return stringList;
```

```
}

public void doDML(String query) {
	PreparedStatement stmt;
	try {
		stmt = conn.prepareStatement(query);
		stmt.execute();
	}
	catch (SQLException e) {
		e.printStackTrace();
		closeConnection();
	}
}
```

4.2.1 Υλοποίηση της αυτόματης χαρτογράφησης

Κατά την διάρκεια της ανάκτησης των δεδομένων των αγγελιών πραγματοποιείται, όπως είδαμε στο προηγούμενο κεφάλαιο, μια προ-επεξεργασία έτσι ώστε οι δεξιότητες που ανακτώνται να πλησιάζουν όσο γίνεται στις βασικές δεξιότητες που είναι αποθηκευμένες στην βάση. Το επόμενο βήμα είναι η προσπάθεια για αυτόματη χαρτογράφηση των ανακτημένων δεξιοτήτων πατώντας το κουμπί “Run Auto-mapper”. Με την λειτουργία αυτή ανακτώνται από την βάση όλες οι δεξιότητες αγγελιών που δεν είναι χαρτογραφημένες και συγκρίνονται μια προς μια με κάθε βασική κατηγορία αγγελιών. Για την σύγκριση χρησιμοποιείται επίσης επικουρικά και ένα αντικείμενο της LevenshteinDistance του πακέτου Apache Commons Lang 3.3. Με αυτόν το τρόπο μπορεί να ελεγχθεί η ομοιότητα ανάμεσα στα δύο αλφαριθμητικά που είναι προς σύγκριση. Η υλοποίηση της λογικής αυτής φαίνεται παρακάτω.

```
private void fillSkillsMap() {

  String skillsQuery = "SELECT SKILL_NAME, SKILL_ID  FROM SKILLS";
  skillsMap = dbManager.doGetStringIntMap(skillsQuery);
}

private void mapSkills() {

  String adSkillsQuery = "SELECT ADSKILL_ID, SKILL_TEXT FROM ADSKILLS WHERE
ADSKILL_ID NOT IN (SELECT ADSKILL_ID FROM SKILLMAPPING)";

  HashMap<Integer, String> adSkillsMap = dbManager
.doGetIntStringMap(adSkillsQuery);
```

```
  LevenshteinDistance ld = new LevenshteinDistance();

  Iterator adSkillsIter = adSkillsMap.entrySet().iterator();

  while (adSkillsIter.hasNext()) {
    HashMap.Entry<Integer, String> unmappedEntry = (Entry<Integer, String>)
adSkillsIter.next();
    int id = unmappedEntry.getKey();
    String skillText = unmappedEntry.getValue();
    skillText = skillText.toLowerCase();

    Iterator skillsIter = skillsMap.entrySet().iterator();

    while (skillsIter.hasNext()) {
      HashMap.Entry<String, Integer> skillsEntry = (Entry<String, Integer>)
skillsIter.next();

      if (skillsEntry.getKey().contains(skillText) ||
skillText.contains(skillsEntry.getKey()) || ld.apply(skillText,
skillsEntry.getKey()) < 2) {

        mapAdSkillToSkill(id, skillsMap.get(skillsEntry.getKey()));
      }

    }
  }
}

private void mapAdSkillToSkill(int id, int skillId) {
  String insertSkillMappingSQL = "INSERT IGNORE INTO SKILLMAPPING(ADSKILL_ID,
SKILL_ID) VALUES(" + id + ", " + skillId + ")";

  dbManager.doDML(insertSkillMappingSQL);
}
```

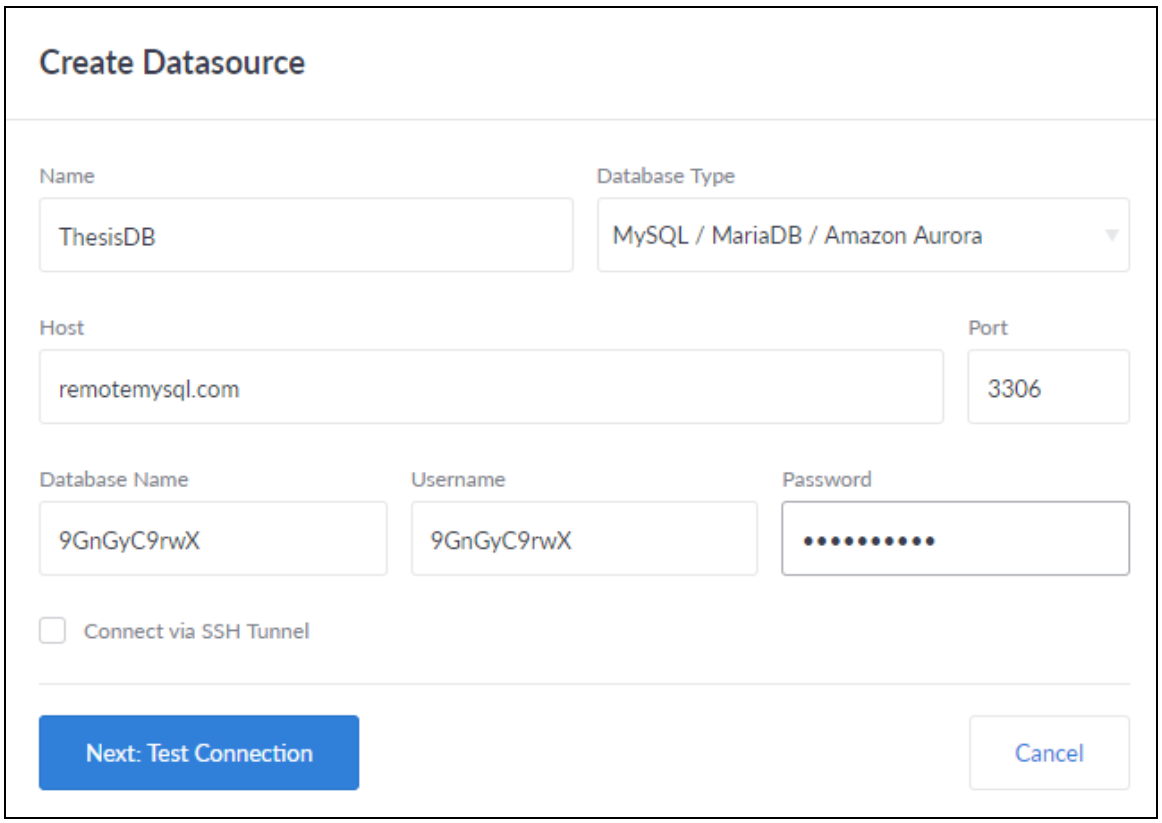
4.2.2 Υλοποίηση της χειροκίνητης χαρτογράφησης

Για όσες δεξιότητες δεν γίνει επιτυχώς η αυτόματη χαρτογράφηση, υπάρχει η δυνατότητα πραγματοποίησής της χειροκίνητα από τον χρήστη. Αυτό γίνεται επιλέγοντας την δεξιότητα από την λίστα που βρίσκεται στα αριστερά και τις δεξιότητες (μέχρι τρεις) των βασικών κατηγοριών που βρίσκονται στα δεξιά. Η ενέργεια του χρήστη ολοκληρώνεται με το πάτημα του πλήκτρου “Map Skill”. Σε περίπτωση που ο χρήστης θέλει να αγνοήσει την συγκεκριμένη δεξιότητα, είτε γιατί η ανάκτηση έγινε λανθασμένα, είτε γιατί δεν μπορεί να αντιστοιχηθεί σε κάποια από τις βασικές κατηγορίες, μπορεί να το πραγματοποιήσει με το πάτημα του πλήκτρου “Ignore Skill”. Παρακάτω δίνεται η υλοποίηση των ενεργειών αυτών.

```
private void mapAdSkillToSkills(String selectedAdSkill, List
selectedSkillsList) {
	String adSkillsQuery = "SELECT ADSKILL_ID FROM ADSKILLS WHERE
SKILL_TEXT = '" + selectedAdSkill + "'";
	String skillsQuery = "SELECT SKILL_ID FROM SKILLS WHERE SKILL_NAME IN
(";

	for (int i = 0; i < selectedSkillsList.size(); i++) {
		if (i == 0)
			skillsQuery += "'" + selectedSkillsList.get(i) + "'";
		else
			skillsQuery += ",'" + selectedSkillsList.get(i) + "'";
	}
	skillsQuery += ")";

	ArrayList<Integer> fetchedAdSkillsIdList =
dbManager.doGetIdList(adSkillsQuery);
	ArrayList<Integer> fetchedSkillsIdList =
dbManager.doGetIdList(skillsQuery);

	for (int i = 0; i < fetchedAdSkillsIdList.size(); i++) {
		int id = fetchedAdSkillsIdList.get(i);
		if (id != 0) {
			for (int j = 0; j < fetchedSkillsIdList.size(); j++) {
				mapAdSkillToSkill(id, fetchedSkillsIdList.get(j));
			}
		}
	}
}

private void mapAdSkillToSkill(int id, int skillId) {
	String insertSkillMappingSQL = "INSERT IGNORE INTO
SKILLMAPPING(ADSKILL_ID, SKILL_ID) VALUES(" + id + ", " + skillId + ")";

	dbManager.doDML(insertSkillMappingSQL);
}

private void deleteAdSkillFromDB(String selectedValue) {
	String deleteAdSkillSQL = "DELETE FROM ADSKILLS WHERE SKILL_TEXT = '" +
selectedValue + "'";

	dbManager.doDML(deleteAdSkillSQL);

}
```

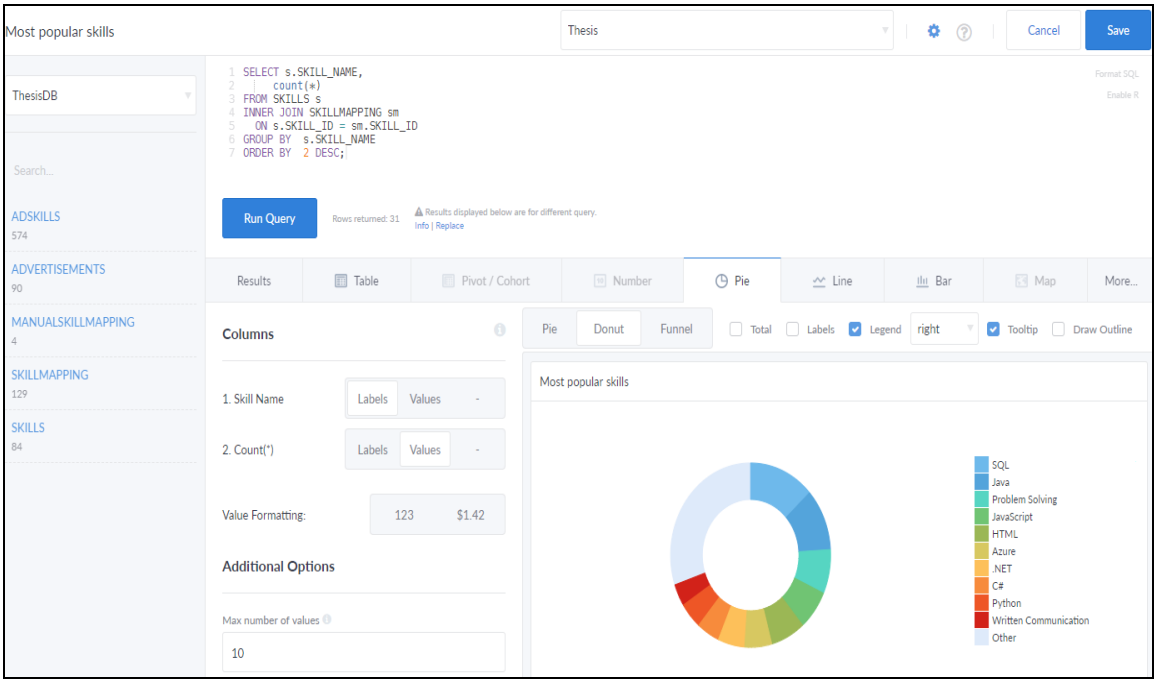
4.3 Ανάλυση με τη χρήση του Cluvio

Με την ολοκλήρωση της ανάκτησης και της χαρτογράφησης των δεδομένων των αγγελιών, μπορεί να γίνει η διασύνδεση της βάσης με κάποια πλατφόρμα ανάλυσης. Στην παρούσα υλοποίηση χρησιμοποιείται το Cluvio. Η σύνδεση με την βάση γίνεται μέσω ενός βοηθού της πλατφόρμας, όπου θέτουμε τα στοιχεία της σύνδεσης.



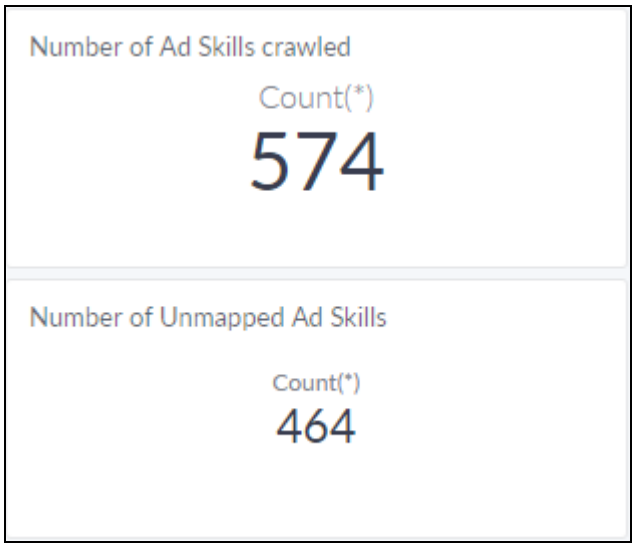
Εικόνα 4-12: Δημιουργία σύνδεσης της πλατφόρμας Cluvio με την βάση

Αφού δημιουργηθεί η σύνδεση, μπορεί να γίνει η δημιουργία αναφορών και γραφημάτων, εφαρμόζοντας ερωτήματα (queries) SQL, μέσω ενός εύχρηστου query editor. Με την εκτέλεση του ερωτήματος, αμέσως η πλατφόρμα μας προτείνει γραφήματα που είναι κατάλληλα για την προβολή των ανακτημένων δεδομένων. Ένα παράδειγμα δημιουργίας ενός τέτοιου ερωτήματος φαίνεται στην Εικόνα 4-13.



Εικόνα 4-13: Φόρμα δημιουργίας αναφορών μέσω της εκτέλεσης ερωτημάτων

Παρακάτω δίνονται κάποια παραδείγματα αναφορών και γραφημάτων που μπορούν να δημιουργηθούν μέσω της πλατφόρμας. Για παράδειγμα, δίνονται μετρήσεις στατιστικών των αποτελεσμάτων της ανάκτησης, όπως το πόσες ξεχωριστές δεξιότητες έχουν ανακτηθεί και πόσες χρειάζεται να χαρτογραφηθούν (Εικόνα 4-14). Τέτοιου είδους μετρήσεις είναι χρήσιμες για τους χρήστες της υλοποίησης και μπορούν να δώσουν μια γενική εικόνα της εξέλιξης της ανάκτησης των αγγελιών, όπως και το πόσες δεξιότητες πρέπει να χαρτογραφηθούν χειροκίνητα, μιας και η αυτόματη χαρτογράφηση δεν μπόρεσε να τις συνδέσει με κάποια από τις βασικές κατηγορίες δεξιοτήτων.

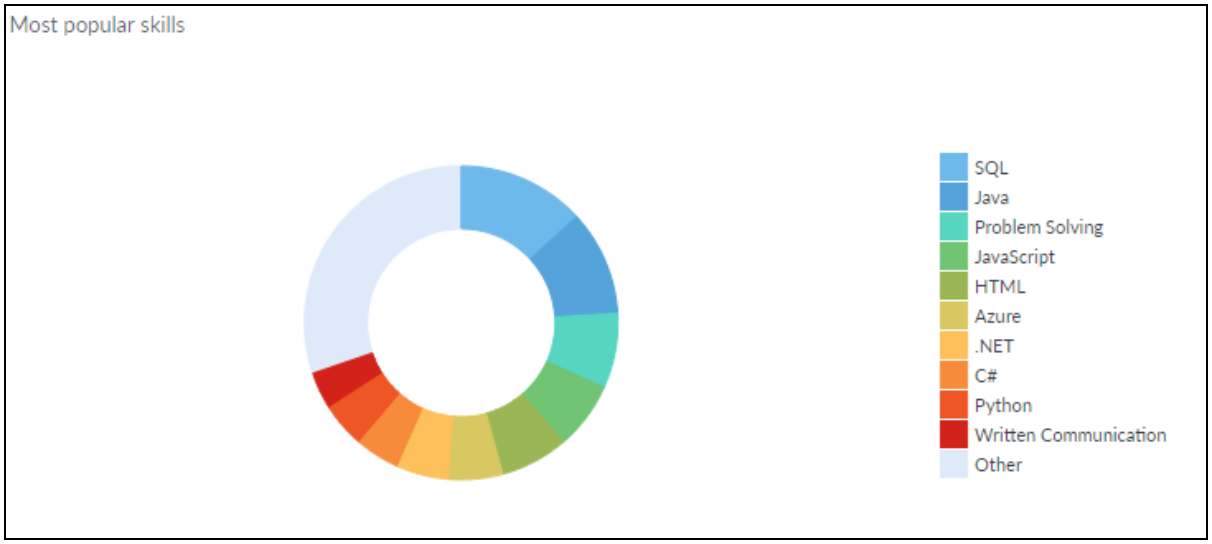


Εικόνα 4-14: Εμφάνιση στατιστικών σχετικά με τις δεξιότητες που έχουν ανακτηθεί

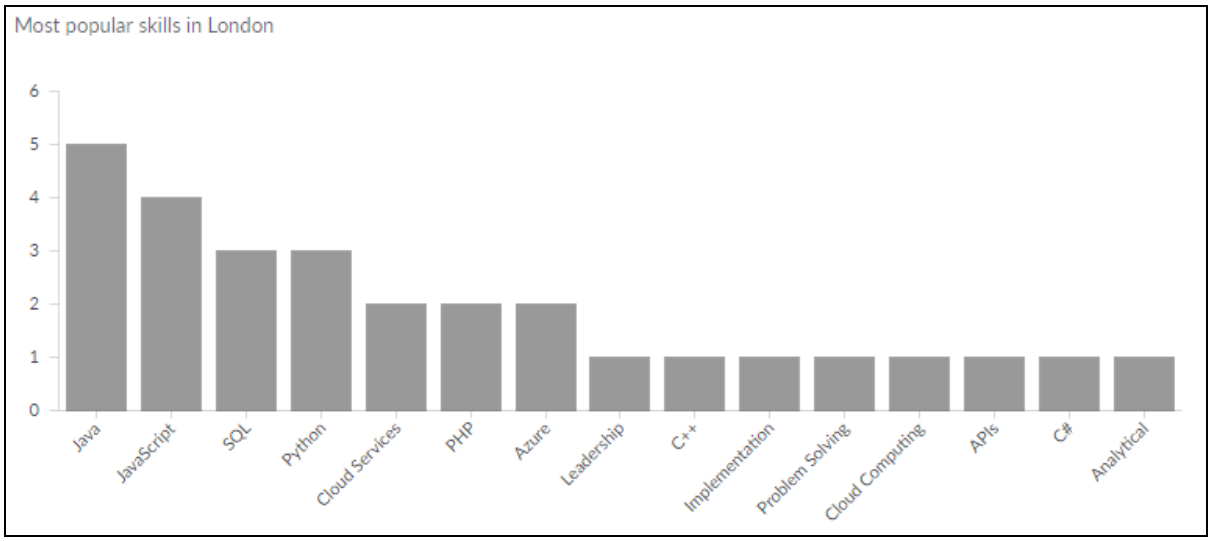
Ο χώρος της τεχνολογίας είναι ιδιαίτερα ευρύς και υπάρχουν πολλές ενδιαφέρουσες κατευθύνσεις τις οποίες μπορεί να ακολουθήσει κάποιος. Το ερώτημα που προκύπτει λοιπόν είναι ποιες δεξιότητες είναι σε μεγαλύτερη ζήτηση στην αγορά εργασίας σε αυτόν τον τομέα; Ίσως το πιο αναμενόμενο και χρήσιμο διάγραμμα που μπορεί να προκύψει από την διαδικασία της συλλογής των δεδομένων, έχει να κάνει ακριβώς με αυτό, δηλαδή με το ποιες είναι οι πιο δημοφιλείς δεξιότητες που ζητούνται.

Καθώς οι αγγελίες και οι δεξιότητες που ανακτήθηκαν ως παραδείγματα για αυτήν την εργασία αφορούν τον κλάδο της πληροφορικής, είναι αναμενόμενο πως οι τεχνικές δεξιότητες είναι αυτές που έχουν το μεγαλύτερο μερίδιο στην πίτα. Με αυτόν τον τρόπο, μπορεί κάποιος να αποκτήσει μια εικόνα των δεξιοτήτων που θα του επιτρέψουν να αυξήσει τις προοπτικές του στην αγορά εργασίας σε αυτόν τον κλάδο.

Στις εικόνες 4-15 και 4-16 παρατίθενται δύο παραδείγματα γραφημάτων που αντικατοπτρίζουν μια τέτοια εικόνα. Αρχικά παρουσιάζεται το ποιες δεξιότητες είναι οι πιο δημοφιλείς στο σύνολο των δεδομένων που ανακτήθηκαν. Στη συνέχεια δίνεται ένα παράδειγμα διαχωρισμού των δεξιοτήτων αυτών με βάση τον γεωγραφικό προσδιορισμό, πχ. ποιες δεξιότητες αναφέρονται συχνότερα στις αγγελίες θέσεων εργασίας που έχουν ως τοποθεσία το Λονδίνο.



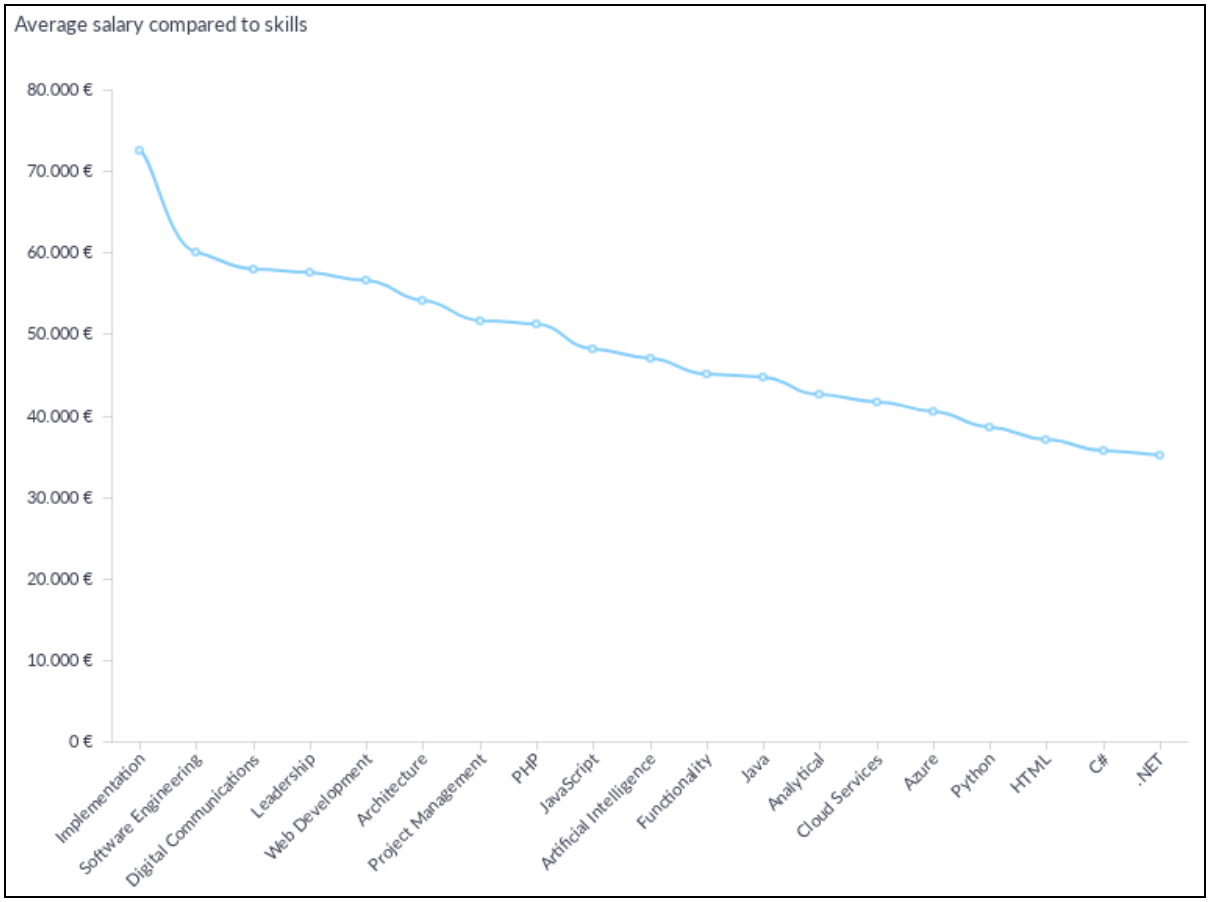
Εικόνα 4-15: Οι πιο δημοφιλείς δεξιότητες, όπως προέκυψε από την ανάκτηση και την χαρτογράφηση



Εικόνα 4-16: Οι πιο δημοφιλείς δεξιότητες με γεωγραφικό προσδιορισμό

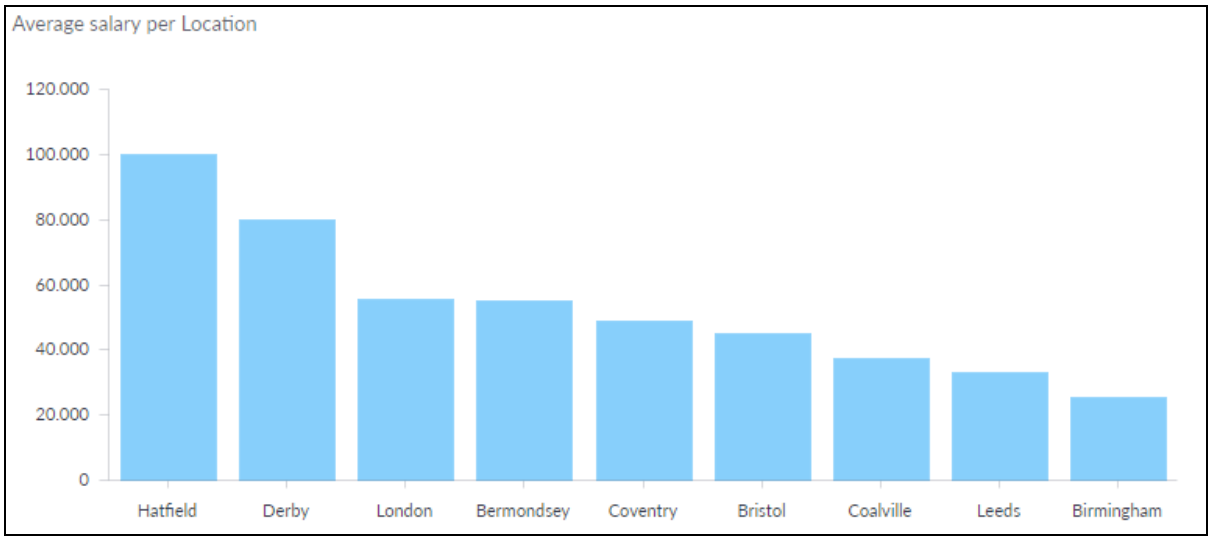
Ένα επίσης σημαντικό κριτήριο για την εξαγωγή συμπερασμάτων σχετικά με την αγορά εργασίας σε ένα κλάδο είναι πώς οι μισθοί συνδέονται με τις δεξιότητες στις αγγελίες που δημοσιεύονται. Δηλαδή ποιες είναι αυτές οι δεξιότητες τις οποίες θα πρέπει να κατέχει κάποιος υποψήφιος εργαζόμενος για να μπορέσει να διεκδικήσει μια θέση σε κάποια εταιρεία με υψηλότερες απολαβές. Συνδυάζοντας λοιπόν τους μισθούς από τις αγγελίες που ανακτήθηκαν με τις δεξιότητες που ζητούνται σε αυτές μπορεί να εξαχθεί ένας μέσος όρος στο ύψος των μισθών ανά δεξιότητα. Ένα παράδειγμα ενός τέτοιου

γραφήματος δίνεται παρακάτω στην Εικόνα 4-17. Όπως μπορεί να δει κάποιος σε μια πρώτη ανάγνωση, δεξιότητες όπως η Τεχνολογία Λογισμικού (Software Engineering), οι δεξιότητες Ηγεσίας (Leadership) ή η Διαχείριση Έργων (Project Management) είναι συνδεδεμένες με αγγελίες που προσφέρουν υψηλότερες απολαβές συγκριτικά με την απλή γνώση κάποιας γλώσσας προγραμματισμού, όπως η Java, η Python και η .NET.



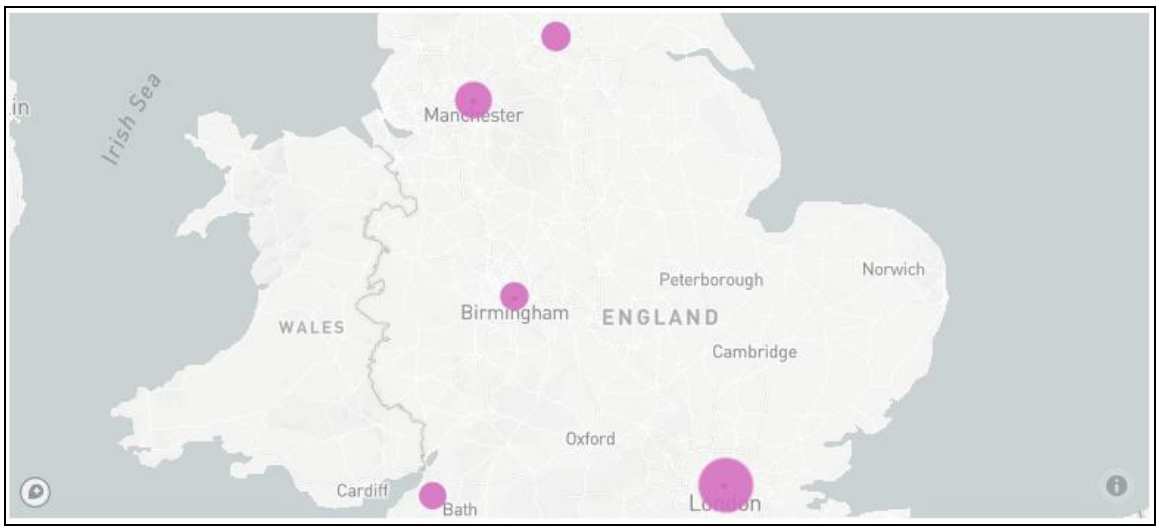
Εικόνα 4-17: Μέσος όρος μισθών ανά δεξιότητα

Σε συνέχεια του προηγούμενου γραφήματος μπορεί να γίνει ακόμα μια συσχέτιση όσων αφορά τις προσφερόμενες απολαβές των αγγελιών που ανακτήθηκαν. Αυτή έχει να κάνει με τον μέσο όρο των απολαβών ανά τοποθεσία της κάθε θέσης. Ένα τέτοιας μορφής γράφημα δίνεται στην Εικόνα 4-18. Παρότι ένα τέτοιο γράφημα μπορεί να μην δίνει μια συγκεκριμένη κατεύθυνση σε έναν υποψήφιο μιας και μπορεί να επηρεάζεται από έναν μικρό αριθμό αγγελιών σε κάποιες τοποθεσίες, ωστόσο μπορεί να δώσει την γενική εικόνα των απολαβών που προσφέρονται σε διάφορες τοποθεσίες και συγκεκριμένα σε πιο μεγάλες πόλεις, όπως είναι στο παράδειγμα το Λονδίνο και το Μπρίστολ.



Εικόνα 4-18: Μέσος όρος μισθών ανά περιοχή

Τέλος, ένας ακόμα σημαντικός δείκτης έχει να κάνει με το πώς καταμερίζονται οι αγγελίες στον χάρτη. Ξεκινώντας την αναζήτηση εργασίας, είναι σίγουρα σημαντικό να μπορεί να διαπιστώσει κάποιος πως οι αγγελίες κατανέμονται στις διάφορες τοποθεσίες, δηλαδή ποιες περιοχές είναι αυτές που παρουσιάζουν έναν υψηλό αριθμό δημοσιευμένων αγγελιών και πώς αυτές συγκρίνονται μεταξύ τους. Τοποθετώντας κάποιες από τις αγγελίες που ανακτήθηκαν σε έναν χάρτη μέσω των δυνατοτήτων που προσφέρει η πλατφόρμα του Cluvio μπορεί να εξαχθεί ένα γράφημα όπως αυτό της εικόνας 4-19.



Εικόνα 4-19: Συγκέντρωση αγγελιών ανά περιοχή

5 Επίλογος

Στην παρούσα εργασία παρουσιάστηκαν οι τεχνολογίες που μπορούν να χρησιμοποιηθούν για την ανάκτηση δεδομένων από σελίδες ιστού. Η μεθοδολογία επικεντρώθηκε στην χρήση βιβλιοθηκών και εργαλείων, τα οποία μπορούν να ενσωματωθούν άμεσα σε ένα έργο γραμμένο σε Java και με τα οποία μπορεί να πραγματοποιηθεί το άνοιγμα ιστοσελίδων για ανάκτηση του περιεχομένου τους. Αυτά είναι το JSoup και το Selenium. Επιπλέον, για την ανάλυση του περιεχομένου χρησιμοποιήθηκαν επιπλέον εργαλεία όπως το HTML Cleaner και η γλώσσα ερωτημάτων XPath.

Όλα τα παραπάνω ενσωματώθηκαν σε μια υλοποίηση που πέραν της ανάκτησης με την δημιουργία crawler, περιλάμβανε την αποθήκευση των δεδομένων σε μια βάση MySQL. Για να μπορέσει να γίνει η χρήση των αποθηκευμένων δεδομένων, ήταν απαραίτητη η εφαρμογή μιας χαρτογράφησης ανάμεσα στον ακατάστατο τρόπο ανάκτησης των δεξιοτήτων και στην δομημένη οργάνωση αυτών σε ευρύτερες κατηγορίες. Ολοκληρώνοντας, παρουσιάστηκε η χρήση μιας πλατφόρμας ανάλυσης, για την δημιουργία γραφημάτων εξάγοντας δεδομένα από τη χαρτογραφημένη πληροφορία εντός της βάσης.

5.1 Συμπεράσματα

Λόγω της αυξανόμενης και μεταβαλλόμενης ανάρτησης αγγελιών εργασίας στο διαδίκτυο, παρουσιάζεται η ανάγκη ανάλυσης αυτού του όγκου της πληροφορίας με έναν πιο συγκεντρωτικό τρόπο. Η μελέτη του θεωρητικού υποβάθρου επικεντρώθηκε στον τρόπο με τον οποίο μπορεί να πραγματοποιηθεί η αυτοματοποιημένη συλλογή δεδομένων από πολλαπλές πηγές με την χρήση Web Crawling και πως αυτή μπορεί να εφαρμοστεί σε ιστοσελίδες αγγελιών. Μέσω της υλοποίησης γίνεται φανερό πως είναι εφικτό να πραγματοποιηθεί μια αυτόματη περιήγηση σε ιστοσελίδες αγγελιών. Με την επανάληψη αυτής της διαδικασίας σε καθημερινή βάση, μπορεί να συγκεντρωθεί ένας ικανοποιητικός όγκος δεδομένων, τα οποία μετά την κατάλληλη χαρτογράφηση, μπορούν να χρησιμοποιηθούν για την εξαγωγή χρήσιμων συμπερασμάτων.

Ένας πρώτος περιορισμός που συναντήθηκε κατά την εφαρμογή της διαδικασίας ανάκτησης μέσω του crawling είναι πως οι περισσότερες ιστοσελίδες κάνουν εκτενή χρήση Javascript. Επίσης σε ορισμένες ιστοσελίδες η εξουσιοδότηση πρόσβασης ελέγχεται με την χρήση cookies. Αυτοί οι περιορισμοί ήταν δυνατό να παρακαμφθούν, όπου χρειαζόταν, με την χρήση του webdriver του Selenium API. Βέβαια, καθώς η υλοποίηση αυτή απαιτεί το άνοιγμα ενός browser, σε αντίθεση με την χρήση του JSoup, ο χρόνος εκτέλεσης είναι σημαντικά μεγαλύτερος.

Ένας ακόμα περιορισμός που συναντήθηκε κατά την διάρκεια της ανάκτησης, έχει να κάνει με την προσπάθεια να μειωθεί στο ελάχιστο η επιβάρυνση προς τον διακομιστή κάθε ιστοσελίδας. Για το λόγο αυτό προστέθηκε στην διάρκεια της εκτέλεσης μια καθυστέρηση ευγένειας (politeness delay), η οποία όμως πρόσθεσε επιπλέον χρόνο στην συνολική διάρκεια εκτέλεσης κάθε crawler. Έτσι, εξαιτίας των περιορισμών αυτών, παρά το ότι η περιήγηση και η ανάκτηση γίνεται αυτοματοποιημένα, δεν σημαίνει ότι γίνεται και γρήγορα, έχοντας ως αποτέλεσμα ο όγκος των δεδομένων που συλλέγεται με κάθε εκτέλεση να είναι δυσανάλογος προς τον χρόνο εκτέλεσης.

Πέραν τούτου, μέσω της υλοποίησης παρουσιάζεται ένας τρόπος ώστε η πληροφορία σχετικά με τις δεξιότητες εντός των αγγελιών να αποθηκευτεί σε μια βάση δεδομένων. Παρότι γίνεται χρήση μιας σχεσιακής βάσης για να υποστηρίξει την υλοποίηση, αξίζει να διερευνηθεί αν η χρήση μιας μη σχεσιακής βάσης (NoSQL) μπορεί να ταιριάξει καλύτερα στην ακατάστατη φύση των δεδομένων που ανακτώνται από το διαδίκτυο, όπως προτείνεται από αρκετές πηγές της βιβλιογραφίας.

Αξίζει να σημειωθεί, πως ακόμα και αν η αυτοματοποιημένη περιήγηση και ανάκτηση είναι εφικτή με την χρήση των crawlers, αυτό δεν σημαίνει πως αυτή είναι μια διαδικασία που δεν χρειάζεται καμία επίβλεψη. Η ανθρώπινη παρέμβαση χρειάζεται σε περίπτωση που η διάταξη μιας ιστοσελίδας αλλάξει. Σε αυτή την περίπτωση είναι απαραίτητη και η τροποποίηση του crawler και η προσαρμογή σε αυτήν την νέα διάταξη. Σαφώς, πιο εξελιγμένες υλοποιήσεις, οι οποίες εφαρμόζονται από εταιρείες κολοσσούς, είναι ικανές να αναγνωρίζουν και να «καταλαβαίνουν» αυτές τις αλλαγές και να προχωράνε σε αυτόματες τροποποιήσεις χωρίς να είναι απαραίτητη η ανθρώπινη παρέμβαση.

Παρά την ύπαρξη των περιορισμών που αναφέρθηκαν, κάνοντας χρήση μιας τέτοιας υλοποίησης δίνεται η δυνατότητα να παραχθούν αναφορές που μπορούν να

καθοδηγήσουν ανθρώπους που βρίσκονται σε αναζήτηση εργασίας, για το πώς μπορούν να αυξήσουν τις πιθανότητες επιτυχής ευρέσεως εργασίας. Μέσω της δημιουργίας γραφημάτων τα οποία περιλαμβάνουν συγκεντρωτικές πληροφορίες για το πλήθος των αγγελιών, τις δεξιότητες που δημοσιεύονται σε αυτές, το ύψος των μισθών ή τον καταμερισμό των αγγελιών σε διάφορες τοποθεσίες, προστίθεται ένα επιπλέον μέσο στο οπλοστάσιο των αναζητούντων εργασία, αλλά και των νέων επιστημόνων που έχουν έλλειψη γνώσης της ζήτησης που υπάρχει στον κλάδο της πληροφορικής, καθώς δεν έχουν βγει ακόμα στην αναζήτηση εργασίας.

Ωστόσο παρά το γεγονός ότι υπάρχουν τα τεχνολογικά μέσα καθιστώντας δυνατή την εφαρμογή μια τέτοιας υλοποίησης σε έναν μεγάλο όγκο δεδομένων, οι περιορισμοί τίθενται από τους όρους χρήσης της εκάστοτε υπηρεσίας. Όπως παρουσιάστηκε, όλες οι ιστοσελίδες οι οποίες έχουν δημοσιευμένες αγγελίες απαγορεύουν ρητά την συλλογή των δεδομένων που υπάρχουν σε αυτές. Αυτός είναι και ο σημαντικότερος ηθικός περιορισμός που τίθεται. Υλοποιώντας μια τέτοια λύση και εφαρμόζοντας την ευρέως, ο προγραμματιστής θα πρέπει να πάρει μια τολμηρή απόφαση που μπορεί να οδηγήσει ακόμα και σε νομικές συνέπειες.

5.2 Μελλοντικές Επεκτάσεις

Παρά το γεγονός ότι η ανάκτηση, η χαρτογράφηση και η ανάλυση στην παρούσα εργασία πραγματοποιείται επιτυχώς, υπάρχουν σημαντικά περιθώρια βελτίωσής της. Λόγω της φύσης των δεδομένων κάθε αγγελίας, η οποία δημοσιεύεται μια φορά και δεν ανανεώνεται σχεδόν καθόλου μέχρις ότου λήξει η προθεσμία της, δεν υπάρχει η ανάγκη για ανάκτηση των δεδομένων με υψηλή συχνότητα, αλλά ανά τακτά χρονικά διαστήματα. Σε περίπτωση που υπήρχε απαίτηση για μια πιο ευέλικτη λύση, η οποία θα θέτει ερωτήματα ανάκτησης σε σύντομα χρονικά διαστήματα, χρειάζεται μια διαφορετική προσέγγιση.

Τέλος, παρά το ότι η μεθοδολογία ανάκτησης κρίνεται αρκετά αξιόπιστη, η μεθοδολογία για την ανάλυση του περιεχομένου των ιστοσελίδων και η αυτόματη χαρτογράφηση των δεξιοτήτων, μπορεί να επεκταθεί με την χρήση μηχανιστικής μάθησης ή ειδικών αλγορίθμων για την συσχέτιση αλφαριθμητικών.

Βιβλιογραφία

Χατζηγεωργίου Α., 2005. *Αντικειμενοστρεφής σχεδίαση.* Αθήνα: Κλειδάριθμος

Castrillo-Fernández O., 2015. *Web Scraping: Applications and Tools*. ePSIplatform Topic Report No. 2015 / 10

Olston C. and Najork M., 2010. *Web Crawling*, Foundations and Trends in Information Retrieval: Vol. 4: No. 3, pp 175-246

Mitchell R., 2013. *Instant Web Scraping with Java.* Birmingham: Packt Publishing

Cording P. H., 2011. *Algorithms for Web Scraping*, Master Thesis, DTU Informatics, Technical University of Denmark

Zhang S., 2017. *Intelligent Web Crawler for Semantic Search Engine*. Master's Projects, Faculty of the Department of Computer Science, San Jose State University

Mitchell R., 2018. *Web Scraping with Python (2nd Edition).* Sebastopol, CA: O'Reilly Media, Inc.

Liu B., 2011. *Web Data Mining, Exploring Hyperlinks, Contents and Usage Data, Second Edition.* Berlin: Springer-Verlag

Menshchikov A., Komarova A., Gatchin Y., Korobeynikov A., Tishukova N., 2017. *A Study of Different Web-Crawler Behaviour*. Proceeding Of The 20th Conference Of Fruct Association, Saint-Petersburg National Research University of Information Technologies, Mechanics and Optics, Saint-Petersburg, Russia

Guojun Z., Wenchao J., Jihui S., Fan S., Hao Z., Jiang L., 2017. *Design and application of intelligent dynamic crawler for web data mining.* 2017 32nd Youth Academic Annual Conference of Chinese Association of Automation (YAC), Hefei, China

Schmidt M., 2015. *Web Crawler*. All Capstone Projects, 148. Governors State University

Martins D., Lam R., Rodrigues J.M.F, Cardoso P.J.S., Serra F., 2015. *A Web Crawler Framework for Revenue Management.* Advances in Electrical and Computer Engineering. Tenerife, Canary Islands, Spain

Shang J., Lin J., Qin V., Li B., Wu M., 2016. *Design of Analysis System for Documents Based on Web Crawler*. 2nd IEEE International Conference on Computer and Communications, Chengdu, China

Agre G.H., Mahajan N.V., 2015. *Keyword Focused Web Crawler*. IEEE Sponsored 2nd International Conference On Electronics And Communication System (ICECS 2015), Coimbatore, India

Van Deursen A., Mesbah A., Nederlof A., 2015. *Crawl-based analysis of web applications: Prospects and challenges.* Science of Computer Programming: Vol. 97: Part 1, pp 173-180

Desai K., Devulapalli V., Agrawal S., Kathiria P., Patel A., 2017. *Web Crawler : Review of Different Types of Web Crawler, Its Issues, Applications and Research Opportunities.* International Journal of Advanced Research in Computer Science: Vol. 8: No. 3, pp 1199-1202

Garg A., Gupta K., Singh A., 2017. *Survey of Web Crawler Algorithms.* International Journal of Advanced Research in Computer Science: Vol. 8: No. 5, pp 426-428

Peng D., Li T., Wang Y., Chen C.L.P., 2018. *Research on Information Collection Method of Shipping Job Hunting Based on Web Crawler.* 8th International Conference on Information Science and Technology, pp 57-62. Granada, Cordoba, and Seville, Spain

Flores F.N. and Moreira V. P., 2016. *Assessing the impact of Stemming Accuracy on Information Retrieval – A multilingual perspective.* Information Processing & Management: Vol. 52: No. 5, pp 840-854

Rajput B.S. and Khare N., 2015. *A survey of Stemming Algorithms for Information Retrieval.* IOSR Journal of Computer Engineering: Vol. 17: No. 3, pp 76-80

Murphy I. *Web scraping bots are 46% of web traffic.* Enterprise Times, 31 Αυγούστου 2016 [online] Available at: <https://www.enterprisetimes.co.uk/2016/08/31/web-scraping-bots-46-web-traffic/> (3 Απριλίου 2019)

Toth A. *6 Misunderstandings About Web Scraping.* Scraping Authority, 24 Νοεμβρίου 2017. [online] Available at: <https://www.import.io/post/6-misunderstandings-about-web-scraping/> (6 Ιανουαρίου 2019)

Yuk C. *Why is Java so popular for developers and programmers?* Pearson Frank, 25 Οκτωβρίου 2017. [online] Available at: <https://www.pearsonfrank.com/blog/why-is-java-so-popular-developers/> (5 Ιανουαρίου 2019)

Tutorials Point. *Eclipse – Overview.* [online] Available at: <https://www.tutorialspoint.com/eclipse/eclipse_overview.htm> (10 Φεβρουαρίου 2019)

Branson T. *The 5 best reasons to choose MySQL – and its 5 biggest challenges.* 14 Απριλίου 2017. [online] Available at: <https://dataconomy.com/2017/04/5-reasons-challenges-mysql/> (12 Φεβρουαρίου 2019)

Selenium Project. *Selenium Documentation.* [online] Available at: < https://www.seleniumhq.org/docs/ > (8 Μαρτίου 2019)

W3Schools. *XML and Xpath.* [online] Available at: < https://www.w3schools.com/xml/xml_xpath.asp > (9 Μαρτίου 2019)

Thompson J., *Using JAXB for XML With Java.* 11 Φεβρουαρίου 2018 [online] Available at: < https://dzone.com/articles/using-jaxb-for-xml-with-java > (15 Μαρτίου 2019)

Oracle, 2010. *The Java EE 5 Tutorial - JAXB Architecture.* [online] Available at: < https://docs.oracle.com/javaee/5/tutorial/doc/bnazg.html > (15 Μαρτίου 2019)

Oracle, 2013. JAXB Release Documentation. [online] Available at: < https://javaee.github.io/jaxb-v2/doc/user-guide/release-documentation.html > (15 Μαρτίου 2019)

Mkyong. *JAXB hello world example.* 29 Αυγούστου 2012 [online] Available at: < https://www.mkyong.com/java/jaxb-hello-world-example/ > (15 Μαρτίου 2019)

Sourceforge. Htmlcleaner transforms HTML to well-formed XML. [online] Available at: < http://htmlcleaner.sourceforge.net/contact.php > (9 Μαρτίου 2019)

Hedley J., jsoup: Java HTML Parser. [online] Available at: < https://jsoup.org/ > (10 Απριλίου 2019)

Cluvio, How it works. [online] Available at: < https://www.cluvio.com/how-it-works.html > (24 Μαρτίου 2019)

Capterra, Cluvio. [online] Available at: < https://www.capterra.com/p/162332/Cluvio/ > (24 Μαρτίου 2019)