ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη και Λειτουργία Συστήματος Κατανεμημένης Ανάκτησης Πληροφοριών»

Του φοιτητή
Βασίλειος Παρλάκογλου
Αρ. Μητρώου: 154520

Επιβλέπων
Ονοματεπώνυμο
Μιχαήλ Σαλαμπάσης

Ημερομηνία ……

Τίτλος Δ.Ε. Ανάπτυξη και Λειτουργία Συστήματος Κατανεμημένης Ανάκτησης Πληροφοριών
Κωδικός Δ.Ε. 20114
Ονοματεπώνυμο φοιτητή/τών Βασίλειος Παρλάκογλου
Ονοματεπώνυμο εισηγητή Μιχαήλ Σαλαμπάσης
Ημερομηνία ανάληψης Δ.Ε. 29-03-2020
Ημερομηνία περάτωσης Δ.Ε.15-06-2021

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Αφιέρωση»

Σε όλους τους καθηγητές και συμφοιτητές μου

Πρόλογος

Η πτυχιακή αυτή, επιλέχθηκε ως ευκαιρία να επεκταθούν οι γνώσεις μου στο περιβάλλον της Java, στις υπηρεσίες διαδικτύου, να καταλάβω την λειτουργία ενος μεγάλου Project, και ως μια πρόκληση για τον εαυτό μου για την επίλυση και αντιμετώπιση προβλημάτων κατα την επέκταση και επιπρόσθεση λειτουργιών σε ένα υπάρχον λογισμικό. Απο την πτυχιακή αυτή, κατάφερα να αποκτήσω μεγαλύτερη γνώση της γλώσσας Java, και πως χρησιμοποιείται σε ένα μεγαλύτερο πεδίο εφαρμογής, παρατήρησα και έμαθα πως λειτουργούν οι υπηρεσίες διαδικτύου, ειδικότερα υπηρεσίες αναζήτησης και επέκτεινα τις επιδεξιότητες μου στην επίλυση κρίσιμων προβλημάτων. Ειδικότερα, επέκτεινα τις γνώσεις μου στην Java, HTML, XML, JavaScript, επικοινωνία πελάτη-διακομιστή για την λειτουργία μιας web εφαρμογής και τον τρόπο με τον οποίο λειτουργεί μια μηχανή αναζήτησης.

Περίληψη

Η πτυχιακή εργασία με τίτλο “Ανάπτυξη και Λειτουργία Συστήματος Κατανεμημένης Ανάκτησης Πληροφοριών” περιγράφει την ανάπτυξη λογισμικού αναζήτησης federated search με όνομα “PerFedPat”, το οποίο κάνει χρήση του ezDL framework.

Στη σημερινή κοινωνία, η δημιουργία πατεντών είναι ένα καθημερινό φαινόμενο, με πολλά γραφεία πατεντών στον κόσμο, να δέχονται αιτήσεις για νέες καθημερινά. Αυτό έχει ως αποτέλεσμα να υπάρχει η ανάγκη της εύρεσης όλων αυτών των πατεντών απο κάθε άτομο που επιθυμεί, χωρίς να χρειάζεται να αναζητεί σε κάθε γραφείο πατεντών ξεχωριστά. Το PerFedPat, είναι ένα εργαλείο για αυτόν ακριβώς το σκοπό, δηλαδή την αναζήτηση πατεντών σε διάφορες ιστοσελίδες, με μόνο μια αναζήτηση. Το PerFedPat, ένα πρόγραμμα σε Java, είναι βασισμένο στο Framework της ezDL, ένα εργαλείο Framework το οποίο δίνει στους προγραμματιστές την εύκολη υλοποίηση κώδικα για αναζήτηση σε web υπηρεσίες.

Για την ανάπτυξη του προγράμματος θα πρέπει να γίνει κατανοητό πως γίνεται η επικοινωνία μεταξύ υπηρεσιών μηχανών αναζήτησης και γενικότερα του web. Στη σημερινή εποχή, για την επικοινωνία χρήστη διακομιστή σε μια ιστοσελίδα χρησιμοποιούνται συνήθως τα REST API endpoints για την παραλαβή και αποστολή δεδομένων με σκοπό την δημιουργία HTML κειμένου και την εμφάνιση του στον χρήστη. Τα δεδομένα μπορεί να είναι σε μορφή επίσης HTML κειμένου, XML δεδομένων ή πιο ευρέως τα τελευταία χρόνια, σε μορφή JSON. Το PerFedPat, χρησιμοποιεί αυτά τα endpoint, είτε είναι προερχόμενα απο επίσημες υπηρεσίες API ακολουθώντας βοηθητικά έγγραφα για την υλοποίηση, είτε κάνοντας αυτόνομα ανάλυση το πως λειτουργεί μια ιστοσελίδα χρησιμοποιώντας έτοιμα εργαλεία που υπάρχουν σε κάθε πρόγραμμα περιήγησης. Επίσης, λόγω της συνεχής ανάπτυξης τεχνολογιών που χρησιμοποιούνται στις ιστοσελίδες, είναι πολλές φορές δύσκολο να γίνει κατανοητή η επικοινωνία που γίνεται. Για αυτό γίνεται επίσης χρήση Headless Browsers για την προσομοίωση μιας αναζήτησης που θα έκανε ένας κανονικός χρήστης, με σκοπό την αυτοματοποίηση της για τους σκοπούς μας.

«Development and Operation of a Distributed Information Retrieval System»

«Vasilios Parlakoglou»

Abstract

The thesis titled "Development and Operation of a Distributed Information Retrieval System" describes the development of federated search software called PerFedPat, which uses the ezDL framework.

In today's society, patent creation is a daily occurrence, with many patent offices around the world accepting applications for new ones every day. As a result, there is a need for all of these patents to be found by anyone, without having to search each patent office separately. PerFedPat is a tool for this very purpose, i.e. the search for patents on various websites, with just one search. PerFedPat, a Java program, is based on the ezDL Framework, a Framework tool that gives developers easy implementation of code for searching any web services for our purpose.

For the development of the program we need to understand how the communication between search engine services and the web in general takes place. Nowadays, REST API endpoints are commonly used for server-user communication on a web page to receive and send data in order to create HTML text and display it to the user. The data can be in HTML text format themselves, XML data or more broadly in recent years, JSON format. PerFedPat uses these endpoints, whether they come from official API services following the given public documentation or do a self analysis of how a web site works thanks to the tools available on web browsers. Also, due to the constant development of technologies used in websites, it is often difficult to understand the communication that takes place. This is why Headless Browsers are also used to simulate a search that a normal user would do, in order to automate it for our purposes.

Περιεχόμενα

Συντομογραφίες

HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
XML	Extensible Markup Language
JSON	JavaScript Object Notation
CSS	Cascading Style Sheets
AJAX	Asynchronous JavaScript and XML

Κεφάλαιο 1ο: Εισαγωγή στις πατέντες

1.1 Έννοια της πατέντας

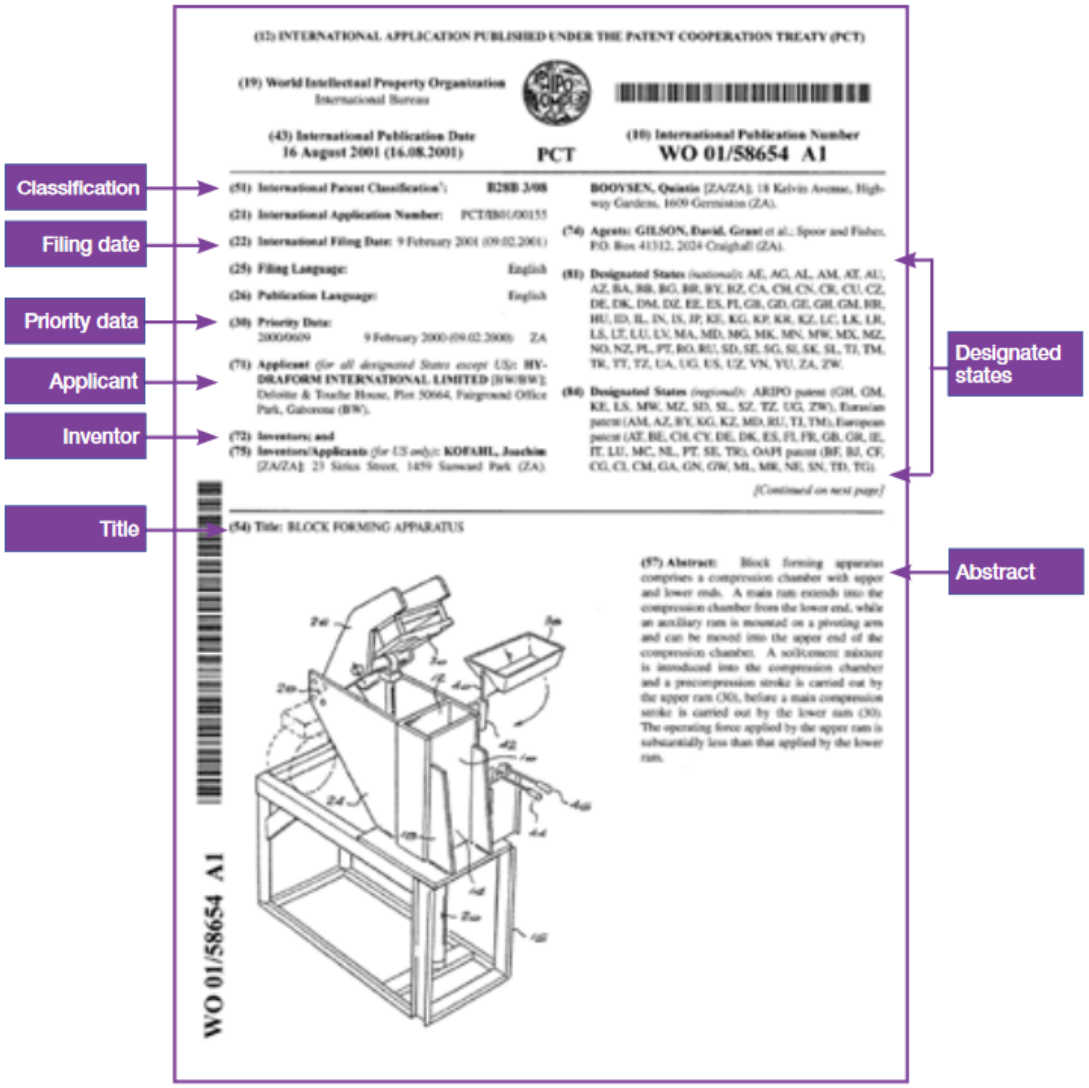
Η Πατέντα (ή Δίπλωμα Ευρεσιτεχνίας) είναι ένα έγγραφο, το οποίο εκδίδεται, κατόπιν αιτήσεως, συνήθως από ένα κυβερνητικό γραφείο, το οποίο περιγράφει μια εφεύρεση και δημιουργεί μια νομική κατάσταση στην οποία η κατοχυρωμένη πατέντα μπορεί να χρησιμοποιηθεί (κατασκευη, πώληση, χρήση, εισαγωγή) μόνο με την άδεια του κατόχου της συγκεκριμένης πατέντας. «Εφεύρεση» σημαίνει λύση σε ένα συγκεκριμένο πρόβλημα στον τομέα της τεχνολογίας. Μια εφεύρεση μπορεί να σχετίζεται με ένα προϊόν ή μια διαδικασία. Η προστασία που παρέχει μία πατέντα είναι περιορισμένη χρονικά (συνήθως στα 20 έτη, αναλόγως την νομοθεσία της εκάστοτε χώρας).

1.2 Χαρακτηριστικά μιας πατέντας

Μια πατέντα που έχει υποβληθεί σε ένα σύστημα πατεντών, θα πρέπει να περιέχει μια εκτεταμένη περιγραφή με συγκεκριμένα χαρακτηριστικά. Η περιγραφή αυτή θα πρέπει να περιέχει προσδιορισμό του τεχνικού πεδίου στο οποίο αναφέρεται η εφεύρεση, να υποδεικνύει το υπόβαθρο της τεχνολογίας το οποίο, σε βαθμό που είναι γνωστό στον αιτούντα, μπορεί να θεωρηθεί χρήσιμο για την κατανόηση της εφεύρεσης, τη σύνταξη της έκθεσης αναζήτησης και την εξέταση της αίτησης πατέντας, και, κατά προτίμηση, να αναφέρει τα έγγραφα που αποτελούν αυτήν την πατέντα.

Αναλύοντας τα χαρακτηριστικά αυτά, μια ολοκληρωμένη περιγραφή της πατέντας θα πρέπει να περιλαμβάνει όλες τις πληροφορίες που έχουν δημοσιευτεί σε ένα έγγραφο πατέντας ή να περιέχει πληροφορίες που μπορούν να προέλθουν από ανάλυση στατιστικών στοιχείων. Στο σχήμα 1.1 δίνεται ένα παράδειγμα. Αυτα περιέχουν:

- Τεχνικές πληροφορίες. Η ανάλυσή τους γίνεται από την ίδια την περιγραφή και με εικόνες-σχέδια τα οποία συνοδεύονται με την εφεύρεση.
- Νομικές πληροφορίες. Η ανάλυσή τους γίνεται από την περιγραφή των αξιώσεων της πατέντας που ορίζουν το πεδίο εφαρμογής της και απο τη νομική του υπόσταση
- Επιχειρησιακές πληροφορίες. Η ανάλυση τους γίνεται από τα δεδομένα αναφοράς που προσδιορίζουν τον εφευρέτη, την ημερομηνία κατάθεσης της πατέντας, την χώρα προέλευσης και λοιπα.
- Πληροφορίες δημόσιας πολιτικής. Δεδομένα λαμβανόμενα από ανάλυση τάσεων αρχειοθέτησης που χρησιμοποιούνται από υπεύθυνους χάραξης πολιτικής, για παράδειγμα μια εθνική βιομηχανική πολιτική στρατηγική.



Σχήμα 1.1: Τα μέρη μιας τυπικής πατέντας

1.3 Εννοιολογικές έννοιες πατεντών

Σε τεχνικά χαρακτηριστικά, μια πατέντα θα πρέπει να περιέχει τα παρακάτω πεδία/στοιχεία:

- Αιτών (Applicant). Το όνομα του ατόμου ή της εταιρείας που υπέβαλε την αίτηση για την προστασία της εφεύρεσης.
- Εφευρέτης (Inventor). Το όνομα του ατόμου ή ατόμων που επινόησαν την νέα τεχνολογία και ανέπτυξαν την εφεύρεση.
- Περιγραφή (Description). Μια σαφής και συνοπτική εξήγηση γνωστών υπαρχουσών τεχνολογιών που σχετίζονται με την νέα εφεύρεση και εξήγηση του τρόπου με τον οποίο αυτή η εφεύρεση θα μπορούσε να εφαρμοστεί για την επίλυση προβλημάτων που δεν αντιμετωπίζονται από υπάρχουσες τεχνολογίες. Επίσης, αναφέρονται συγκεκριμένες ενσωματώσεις της νέας τεχνολογίας.

- Αξιώσεις (Claims). Νομικός ορισμός του αντικειμένου για το οποίο ζητείται ή χορηγείται προστασία (της πατέντας). Κάθε αξίωση είναι μια πρόταση σε νομική μορφή που ορίζει μια εφεύρεση και τα μοναδικά τεχνικά χαρακτηριστικά της. Οι αξιώσεις πρέπει να είναι σαφείς και συνοπτικές και να υποστηρίζονται πλήρως από την περιγραφή τους.
- Κατάθεση προτεραιότητας (Priority filing). Η πρώτη πρωτότυπη κατάθεση βάσει της οποίας μπορούν να γίνουν περαιτέρω διαδοχικές εθνικές, τοπικές ή διεθνής καταθέσεις εντός της περιόδου προτεραιότητας του ενός έτους.
- Ημερομηνία προτεραιότητας (Priority date). Η ημερομηνία της πρώτης αυτής κατάθεσης απο την οποία ξεκινά η περίοδος προτεραιότητας αυτού του ενός έτους για περαιτέρω αιτήσιες.
- Ημερομηνία κατάθεσης. (Filing date). Η ημερομηνία της υποβολής μεμονωμένης αίτησης πατέντας σε ένα συγκεκριμένο γραφείο πατεντών (patent office).
- Καθορισμένα κράτη (Designated states). Εάν η αίτηση είναι εθνική ή διεθνής, προσδιορίζει τις χώρες (ή πολιτείες/νομοί κλπ.) στις οποίες μπορούν να επεκταθούν τα δικαιώματα της πατέντας.
- Σχέδια/Εικόνες (Drawings). Μια εικονική περιγραφή της πατέντας χρησιμοποιώντας περιγραφικά σχέδια.
- Αριθμός δημοσίευσης (Publication Number). Μια πατέντα πάντα έχει ένα μοναδικό αριθμό δημοσίευσης που την προσδιορίζει.

Μια πατέντα επίσης κατατάσσεται κάτω από μια ιεραρχία ταξινόμησης πατεντών. Η πιο διαδεδομένη είναι η IPC (International Patent Classification). Η IPC έχει ρόλο την διεθνής ομοιόμορφη ταξινόμηση των πατεντών με σκοπό την δημιουργία ενός αποτελεσματικού εργαλείου αναζήτησης, για την ανάκτηση πατεντών απο γραφεία πατεντών και άλλα άτομα/χρήστες. Κάθε πατέντα κατατάσσεται στη κορυφή από το πρώτο σύμβολο ταξινόμησης. Για παράδειγμα η IPC ορίζει το γράμμα C ως “Chemistry, Metallurgy” που σημαίνει πως μία πατέντα οπου το IPC της έχει πρώτο γράμμα C θα ανήκει κάτω από την κατηγορία χημείας και μεταλλουργίας.

Κεφάλαιο 2ο: Αναζήτηση στο διαδίκτυο

2.1 Μηχανές Αναζήτησης

Μια μηχανή αναζήτησης είναι ένα σύστημα ανάκτησης πληροφοριών σχεδιασμένο με σκοπό την εύρεση μιας ή περισσότερων πληροφοριών σε ένα υπολογιστικό σύστημα. Η πιο διαδεδομένη χρήση τους είναι η διαδικτυακές μηχανές αναζήτησεις. Τα αποτελέσματα μιας αναζήτησης επιστρέφονται στο χρήστη σε μια μορφή λίστας αποτελεσμάτων. Οι πληροφορίες αυτές μπορεί να είναι σύνδεσμοι ιστοσελίδων, εικόνες, βίντεο, έγγραφα κλπ. Παραδείγματα μηχανών αναζήτησης στο διαδίκτυο μπορεί να είναι μια γενικού σκοπού όπως η Google ή μια με πιο συγκεκριμένο πεδίο εφαρμογής όπως το Skroutz που κάνει αναζήτηση προϊόντων για διαδικτυακή αγορά. Μια μηχανή αναζήτησης αποτελείται συνήθως από τρία κύρια μέρη. Τον “Crawler”, “Indexer” (που πολλές φορές περιγράφονται μαζί ως ένα σύστημα) και τον “Query Processor”.

2.1.1 Query Processor (Επεξεργαστής ερωτημάτων)

Ο Query Processor είναι ένα λογισμικό μιας μηχανής αναζήτησης που είναι υπεύθυνο για την επεξεργασία ερωτημάτων ενος χρήστη, και την επιστροφή των αποτελεσμάτων του ερωτήματος αυτού. Ο Query Processor περιλαμβάνει μια διεπαφή γραφικού περιβάλλοντος για την υποβολή των ερωτημάτων ενός χρήστη. Σε πιο βαθύ επίπεδο, περιλαμβάνει επίσης τον μηχανισμό αξιολόγησης της σχετικότητας των όρων του ερωτήματος και των πληροφοριών που υπάρχουν διαθέσιμες απο την μηχανή αναζήτησης. Τέλος περιέχει και έναν μηχανισμό μορφοποίησης των αποτελεσμάτων, συνήθως σε μια μορφή λίστας, που επιστρέφονται στον χρήστη. Τα αποτελέσματα που επιστρέφονται εξαρτώνται κυρίως από το τι παραμέτρους έχει υποβάλει ο χρήστης (π.χ. κάποιο κείμενο, τίτλο, ημερομηνία κ.α.). Σε πολλές μηχανές αναζήτησης, τα αποτελέσματα που επιστρέφονται μπορεί να επηρεάζονται και από εξωτερικούς παράγοντες, όπως επικαιρότητα, χρονικό διάστημα αναζήτησης, ακόμα και από τα χαρακτηριστικά ενός χρήστη (π.χ. ιστορικό αναζήτησης, προτιμήσεις κ.α.). Οι τεχνικές που χρησιμοποιούνται από τις μηχανές αναζήτησης δεν είναι γνωστές στο ευρύ κοινό και μόνο εικασίες μπορούν να γίνουν για το πώς δουλεύουν πραγματικά οι μηχανισμοί που περιγράφηκαν παραπάνω. Με την εξέλιξη της μηχανικής μάθησης και του Deep Learning, οι μηχανές αναζήτησης μπορούν να γίνουν ακόμα πιο πολύπλοκες. Σε γενικότερες γραμμές όμως, χρησιμοποιούν αλγορίθμους τύπου Page Rank οι οποίες δίνουν διαφορετική βαρύτητα σε κάθε ιστοσελίδα ανάλογα με τους υπάρχοντες όρους σε κάποιο ερώτημα ενώ από την άλλη, μια διαφορετική προσέγγιση είναι η στατιστική ανάλυση των περιεχομένων των ιστοσελίδων και η επιστροφή αποτελεσμάτων βάσει κάποιων μετρικών.

2.1.2 Indexer (Ευρετήριο)

Ο ρόλος ενός Indexer σε μια μηχανή αναζήτησης είναι η καταχώρηση των όρων που περιέχονται σε μια ιστοσελίδα, δημιουργώντας ένα ευρετήριο για πιο άμεση πρόσβαση στα αποτελέσματα ενός ερωτήματος αναζήτησης. Το ευρετήριο αυτό αποθηκεύεται σε μια βάση δεδομένων που έχει πρόσβαση η μηχανή. Περιέχει πληροφορίες που είναι αποτέλεσμα μιας διαδικασίας επεξεργασίας του περιεχομένου των ιστοσελίδων. Αυτές οι πληροφορίες συνήθως είναι τα κύρια στοιχεία του αποτελέσματος, όπως ο τίτλος, επικεφαλίδες, μετα-δεδομένα, λέξεις κλειδιά, κ.α. Με αυτόν τον τρόπο η αναζήτηση γίνεται με μεγαλύτερη απόδοση και καλύτερη εξοικονόμηση χώρου δεδομένων.

2.1.3 Web Crawler

Ο ρόλος του Web Crawler είναι η εύρεση ιστοσελίδων και η παράδοσή τους στον indexer. Ο Crawler προσπελαύνει συστηματικά το παγκόσμιο διαδίκτυο με σκοπό να λάβει απαραίτητες πληροφορίες που χρειάζονται για τον Indexer. Σκοπός τους είναι να λάβουν πληροφορίες από όσο το δυνατό περισσότερες ιστοσελίδες. Οι τεχνικές που χρησιμοποιεί ένας crawler είναι η αναζήτηση όλων των εξωτερικών υπερσυνδέσμων (hyperlinks) από ένα πλήθος αρχικής λίστας ιστοσελίδων η οποία συνεχίζεται αναδρομικά. Ο Crawler επίσης θα πρέπει να είναι υπεύθυνος οι πληροφορίες που λαμβάνει και στέλνει στον indexer να είναι και πρόσφατες, αφου πολλές ιστοσελίδες μπορεί να έχουν δυναμικό περιεχόμενο που αλλάζει από στιγμή σε στιγμή. Ιστοσελίδες που δεν καταλήγουν ποτέ να είναι προσβάσιμες από έναν Web Crawler θεωρούνται στη τελική ως Deep Web, για παράδειγμα περιεχόμενο ιστοσελίδας που απαιτεί σύνδεση λογαριασμού, ή πληρωμή.

2.2 Μετα-αναζήτηση (Meta-search)

Οι μεταμηχανές αναζήτησης είναι ένα είδος μηχανών αναζήτησης διαδικτύου, που επιτρέπουν την αναζήτηση σε άλλες πολλαπλές μηχανές αναζήτησης. Με την ενέργεια της αναζήτησης αποστέλλονται αιτήματα σε πολλαπλές πηγές και, τα αποτελέσματα συναθροίζονται. Στο τέλος τα αποτελέσματα της μετα-αναζήτησης αποτελούνται από μια συλλογή αποτελεσμάτων πολλαπλών μηχανων αναζητησης . Συνήθως, τα αποτελέσματα παρουσιάζονται σε μια λίστα με τα σχετικά έγγραφα, συνδέσμους και τις πηγές από τις οποίες αυτές ελήφθησαν. Σε πολλές μετα-μηχανές δίνεται η δυνατότητα προσαρμογής των πηγών από τις οποίες θα λαμβάνονται αποτελέσματα. Η χρήση των μετα-μηχανών στην εποχή μας μπορεί να μην είναι διαδεδομένη και αναγκαία, κυρίως λόγω της εκσυγχρονισμού των κάθε ξεχωριστών μηχανών αναζήτησης που τις καθιστούν χρήσιμες από μόνες τους. Παρ'όλα αυτά, λογω του γεγονότος ότι εν τέλει κάθε μηχανή αναζήτησης είναι μοναδική στον τρόπο αναζήτησης των αποτελεσμάτων του μαζι με το γεγονός ότι το διαδίκτυο συνεχώς επεκτείνεται, η ζήτηση μηχανών μετα-αναζήτησης παραμένει.

2.2.1 Χαρακτηριστικά μηχανής μετα-αναζήτησης

Όλες οι μηχανές μετααναζήτησης, έχουν κάποια κοινά χαρακτηριστικά. Η υλοποίηση τους μπορεί να διαφέρει από μηχανή σε μηχανή. Τα χαρακτηριστικά αυτά είναι:

- Δυνατότητα κλήσης των ερωτημάτων σε άλλες μηχανές αναζήτησης. Καθιστά το κύριο χαρακτηριστικό μιας μετα-μηχανής. Μια τέτοια μηχανή, θα πρέπει να μπορεί να κάνει τέτοιες κλήσεις σε πολλαπλές πηγές μηχανών. Η επιλογή αυτών των μηχανών αναζήτησης καθιστά σημαντικό μέρος κατα την ανάπτυξη μιας μετα-μηχανής. Θα πρέπει
- Η ταυτόχρονη αναζήτηση στις μηχανές αναζήτησης. Μια μηχανή μετααναζήτησης θα πρέπει να είναι ικανή να μπορεί να διαχειρίζεται παραλληλοποίηση των ερωτημάτων σε κάθε μηχανή. Με αυτόν τον τρόπο, ο χρόνος για την ολοκλήρωση ενος ερωτήματος στην μετα-μηχανή, θα είναι πάντα τουλάχιστον ο χρόνος επιστροφής αποτελεσμάτων της αργότερης πηγής αναζήτησης.
- Η αναγνώριση, επεξεργασία και εμφάνιση των αποτελεσμάτων. Τα αποτελέσματα από πηγή σε πηγή, διαφέρουν, όχι μόνο ως προς τα ίδια τα αποτελέσματα που επιστρέφουν, αλλα και προς την δομή και τον τρόπο με την οποία επιστρέφονται αυτές. Η μετα-μηχανή αναζήτησης θα πρέπει να μπορεί να διαχειριστεί για κάθε πηγη τα αποτελέσματα αυτά ξεχωριστά, και να τα συλλέξει για επεξεργασία. Η επεξεργασία περιλαμβάνει την συγχώνευση κοινών αποτελεσμάτων, διαγραφή διπλότυπων αποτελεσμάτων, την τελική επιλογή "βαθμολογίας"

των αποτελεσμάτων (το πρώτο αποτέλεσμα μιας πηγής, μπορεί να είναι 5ο σε μια άλλη, οπότε θα πρέπει να γίνει η κατάλληλη τελική επιλογή στην σειρά εμφάνισης και στα αποτελέσματα της μετα-μηχανής). Τέλος θα πρέπει να γίνεται η εμφάνιση στον χρήστη των αποτελεσμάτων ως μια ενιαία λίστα με επιπρόσθετες δυνατότητες, ανάλογα με τις ανάγκες της κάθε μετα-μηχανής. Για παράδειγμα η δυνατότητα κατηγοριοποίησης των αποτελεσμάτων ανα θεματολογία, αλφαβητικά, ανα πηγή κλπ.

2.2.2 Τεχνικά χαρακτηριστικά μηχανών μετααναζήτησης

Παρακάτω παρουσιάζονται μερικά τεχνικά χαρακτηριστικά που χρησιμοποιούν μηχανές μετααναζήτησης.

- Τρόπος συγχώνευσης αποτελεσμάτων. Όπως προαναφέρθηκε, η συγχώνευση των αποτελεσμάτων είναι απαραίτητο χαρακτηριστικό των μετα-μηχανών αναζήτησης. Χρησιμοποιούνται διάφορες τεχνικές για την συγχώνευση των αποτελεσμάτων αυτών και την σειρά εμφάνισης τους. Για αρχή, η σειρά εμφάνισης τους μπορεί να γίνει με ανάμιξη των "βαθμολογιών προτεραιότητας" από κάθε πηγή. Αυτό μπορεί να γίνει με χρήση συντελεστών "σημαντικότητας" για κάθε πηγή, υπολογισμός μέσου όρου των βαθμολογιών σε κάθε πηγή (αν υπάρχει). Άλλη τεχνική είναι η εμφάνιση της σειράς με τεχνικές voting-based, για παράδειγμα η Borda Count, στην οποία κάθε αποτέλεσμα έχει ένα συντελεστή "ψήφου" αναλόγως την θέση προτεραιότητας από κάθε πηγή. Για παράδειγμα το πρώτο αποτέλεσμα μιας πηγής Α, μπορεί να έχει 10 ψήφους, το δεύτερο 9 κ.ο.κ. Το αποτέλεσμα με τις περισσότερες "ψήφους" θα είναι και το πρώτο αποτέλεσμα στον χρήστη της μετα-μηχανής
- Γραφική διασύνδεση Χρήστη. Οι περισσότερες μετα-μηχανές είναι web-based που σημαίνει πως η γραφική διασύνδεση γίνεται συνήθως μέσω ενός browser. Υπάρχουν και εφαρμογές μετα-μηχανής ως ανεξάρτητα προγράμματα βασισμένα όμως παντα σε μια Server-Client αρχιτεκτονική και σε γλώσσα που υποστηρίζει εύκολα διεπαφές χρήστη (π.χ. Java, C++ κ.α.). Στις web-page εφαρμογές, γίνεται συνήθως η χρήση φορμών HTML, στις οποίες περιέχεται ένα ή περισσότερα πεδία καταχώρησης (user input). Κάποιες εφαρμογές μπορούν να έχουν μόνο ένα, απλοποιόντας την εμπειρία του χρήστη, ενω άλλες μπορούν να έχουν μια πιο προχωρημένη διασύνδεση εαν επιθυμειται συγκεκριμενοποίηση της αναζήτησης. Για παράδειγμα μπορεί να περιέχει πεδία εισαγωγής ημερομηνίας, radio buttons για συγκεκριμένες επιλογές κ.α..
- Επεξεργασία αποτελεσμάτων. Τα αποτελέσματα που επιστρέφονται απο μια πηγή αναζήτησης θα έχουν μια συγκεκριμένη μορφή. Αυτή η μορφή μπορεί να είναι μια HTML σελίδα, που θα χρειαστεί περαιτέρω επεξεργασία για την απόκτηση της πληροφορίας (HTML scrapping). Πολλές μηχανές επίσης επιστρέφουν τα αποτελέσματα μεσω χρήση REST API σε μορφή JSON (ή μερικές παλαιότερες σε XML). Σε κάθε περίπτωση, η πληροφορία με τα αποτελέσματα που επιστρέφεται πρέπει να επεξεργαστεί και να γίνει η κατάλληλη εξαγωγή της ζητούμενης πληροφορίας (και αγνόηση περιττών) και την εμφάνιση της με κοινό τρόπο στον χρήστη.

2.3 Ενοποιημένη αναζήτηση (Federated Search)

2.3.1 Ορισμός Ενοποιημένης αναζήτησης (Federated Search)

Η τεχνική της ενοποιημένης αναζήτησης (Federated Search) ονομάζεται η τεχνολογία ανάκτησης πληροφορίας που αποσκοπεί την ανάπτυξη μεθόδων αναζήτησης σε κατανεμημένα και πιθανότατα ετερογενή σύνολα δεδομένων, καθώς και στην επιστροφή των επιμέρους αποτελεσμάτων ως μια ενιαία συλλογή.

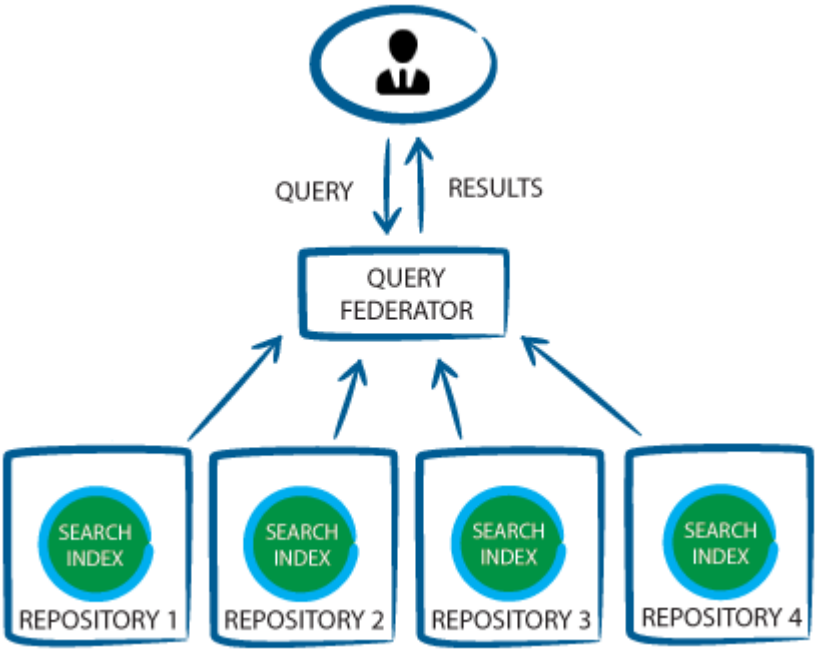
2.3.2 Βασικά χαρακτηριστικά της ενοποιημένης αναζήτησης

Στα συστήματα αναζήτησης τύπου Federated Search, ο στόχος είναι η αναζήτηση σε ανεξάρτητες ετερογενείς συλλογές δεδομένων και η επιστροφή των αποτελεσμάτων ως μια μεμονωμένη λίστα με αποδοτικό τρόπο. Το κεντρικό μέρος του συστήματος αναζήτησης, ο Query-federator ή Broker, έχει ως κύρια λειτουργία του την λήψη των ερωτημάτων του χρήστη και την υποβολή τους στις κατάλληλες υπηρεσίες αναζήτησης που υποστηρίζονται. Οι υπηρεσίες που θεωρούνται κατάλληλες είναι αυτές που θα έχουν τις πιο σχετικές συλλογές δεδομένων με τα υποβληθέντα ερωτήματα. Για να γίνει δυνατή η επιλογή, ο Query-federator έχει αποθηκεύσει κάποιες πληροφορίες για τα δεδομένα της κάθε υπηρεσίας που έχει στη διάθεσή του. Βέβαια, κάτι τέτοιο είναι δυνατό με τις υπηρεσίες που έχουν συνεργάσιμο περιβάλλον (cooperative environment) και παρέχουν στατιστικά στοιχεία για τις συλλογές τους. Για τα μη συνεργάσιμα περιβάλλοντα (uncooperative environment), χρησιμοποιείται η τεχνική της συλλογής πληροφορίας μέσω υποβολής δειγμάτων-ερωτημάτων (probe queries). Οι συλλογές αυτές ονομάζονται representation set και βοηθούν στον υπολογισμό συνάφειας ενός ερωτήματος και των υπηρεσιών που είναι διαθέσιμες. Τα τελικά αποτελέσματα συχνά αποτελούνται από απαντήσεις που επιστρέφονται από πολλαπλές συλλογές.

2.3.3 Αρχιτεκτονική Federated Search

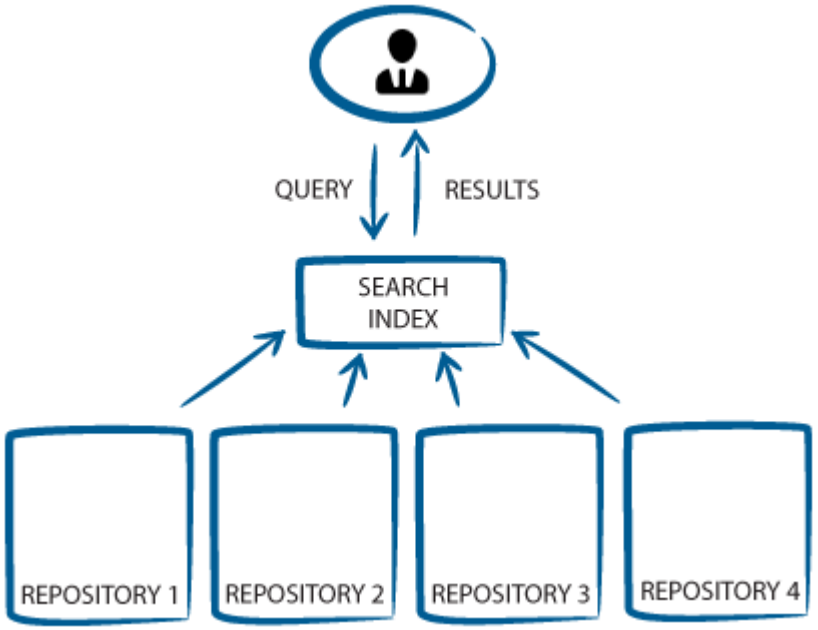
Για την αναζήτηση και ανάκτηση αποτελεσμάτων μέσω της Federated Search χρησιμοποιούνται διάφορες αρχιτεκτονικές, αναλόγως την προσέγγιση και τις απαιτήσεις που χρειάζονται για το κάθε σύστημα.

- Search-Time Merging. Στις περισσότερες περιπτώσεις, είναι η πιο γρήγορη αρχιτεκτονική και ευκολότερη στην υλοποίηση της. Στην αρχιτεκτονική αυτή, ο “query federator” παραλαμβάνει ένα ερώτημα και το αποδίδει παράλληλα σε άλλες μηχανές αναζήτησης που υποστηρίζονται. Έπειτα, γίνεται η αναμονή των απαντήσεων από την κάθε μηχανή αναζήτησης, οι οποίες μόλις ληφθούν, επεξεργάζονται και συγχωνεύονται σε μια λίστα αποτελεσμάτων. Το μεγαλύτερο πλεονέκτημα της αρχιτεκτονικής αυτής είναι ότι μπορεί να έχει μεγάλη κάλυψη στο διαδίκτυο (αν χρησιμοποιεί σημαντικά μεγάλες και ανεπτυγμένες πηγές για παράδειγμα η Google) ενώ επίσης με αυτόν τον τρόπο, η Federated Search μηχανή δεν χρειάζεται να αποθηκεύει τα δεδομένα των αποτελεσμάτων σε κάποιο ευρετήριο (index) που σημαίνει χρήση λιγότερων πόρων. Το σχήμα 2.1 περιγράφει εικονικά την αρχιτεκτονική αυτή



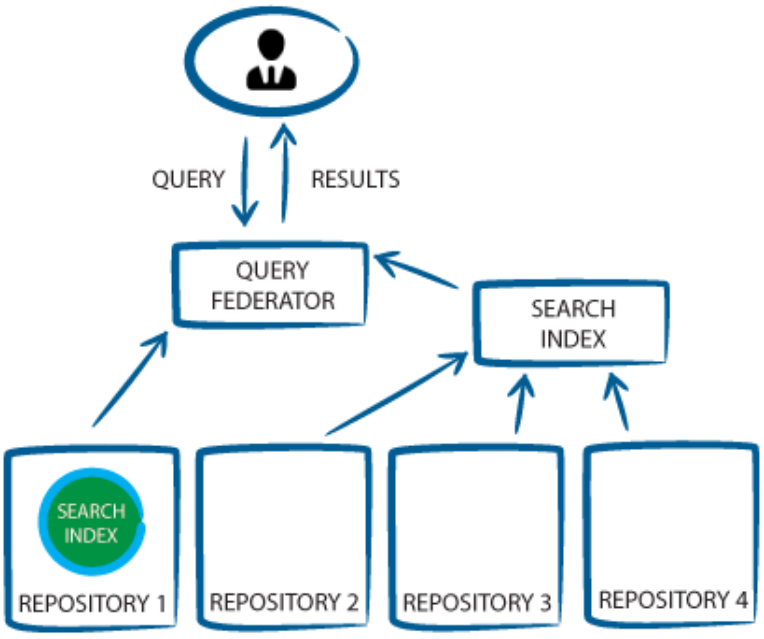
Σχήμα 2.1: Search-Time Merging

- Index-Time Merging. Η συγκεκριμένη αρχιτεκτονική απαιτεί το περιεχόμενο των αποτελεσμάτων να αποθηκευτεί σε ενα κεντρικό σύστημα ευρετηρίου (index). Είναι μια τυπική αρχιτεκτονική κοινών μηχανών αναζήτησης, στην οποία αποθηκεύονται κύρια μέρη αποτελεσμάτων με σκοπό την επαύξηση των τελικών αποτελεσμάτων όπως καλύτερη εμφάνιση σειράς συνάφειας. Το σχήμα 2.2 περιγράφει εικονικά αυτή την αρχιτεκτονική



Σχήμα 2.2: Index-Time Merging

- Hybrid Federated Search. Η τεχνική αυτή είναι μια υβριδοποίηση των δύο προηγουμένων αρχιτεκτονικών. Υποστηρίζεται η χρήση indexing στο σύστημα (Index-Time Merging) ενω παράλληλα γίνεται η υποστήριξη της πιο άμεσης αναζήτησης χωρίς την παρέμβαση του indexer (Search-time Merging). Το σχήμα 2.3 περιγράφει εικονικά αυτήν την αρχιτεκτονική.



Σχήμα 2.3: Hybrid Federated Search

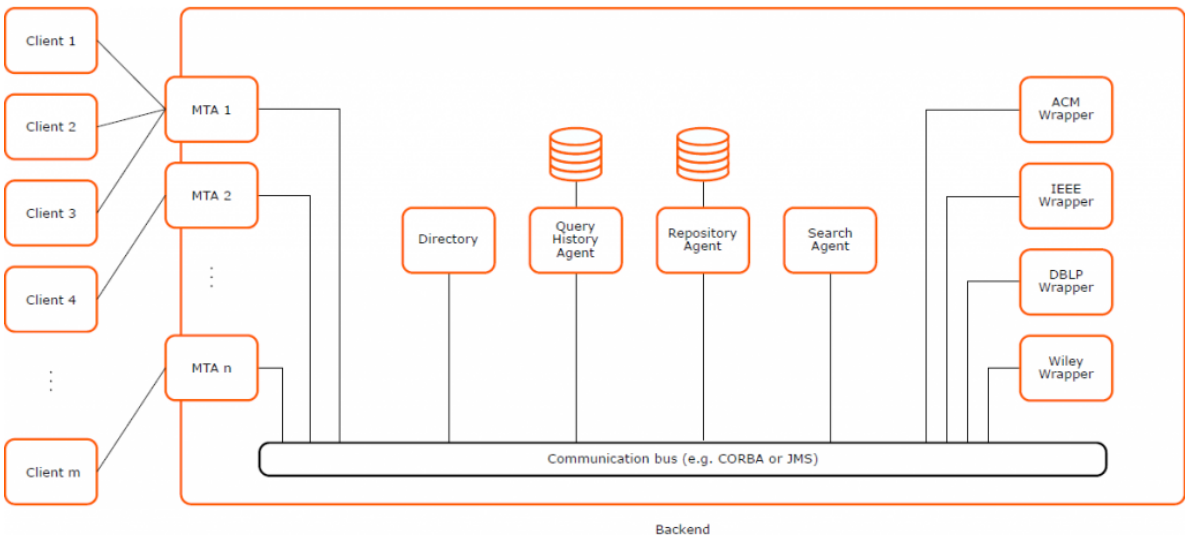
Κεφάλαιο 3ο: Ενοποιημένη αναζήτηση και ezDL

3.1 Εισαγωγή στην ezDL

Το ezDL είναι ένα σύστημα και framework (πλαίσιο λογισμικού) βασισμένο στην γλώσσα Java, φτιαγμένο με σκοπό την ανάπτυξη λογισμικών για εύκολη πρόσβαση σε αναζήτηση ψηφιακών βιβλιοθηκών, βάσεις δεδομένων και άλλων πηγών για την ενοποίηση των αποτελεσμάτων τους μαζι με επιπρόσθετες δυνατότητες για καλύτερη οργάνωση και εμφάνιση. Το ezDL ενσωματώνει την φιλοσοφία της μετα-μηχανής αναζήτησης και της ενοποιημένης αναζήτησης (Federated Search) συναθροίζοντας αποτελέσματα ενός πλήθους πηγών παρέχοντας εργαλεία χρήσιμα για την τελική προβολή τους στον χρήστη. Δίδεται επίσης επεκτασιμότητα και επαναχρησιμοποίηση βασικών λειτουργιών ενω παράλληλα αποτελεί και σύστημα αξιολόγησης αποτελεσμάτων αναζήτησης. Αυτό που κάνει ξεχωριστό το ezDL είναι η ενοποίηση όλων αυτών των χαρακτηριστικών σε ένα κοινό εργαλείο.

3.2 Αρχιτεκτονική της ezDL

Η αρχιτεκτονική του συστήματος ezDL κάνει εκτενή χρήση διαχωρισμού των εννοιών μιας ενοποιημένης μηχανής αναζήτησης. Περιορίζει τις αλληλεξαρτήσεις στο ελάχιστο με σκοπό να κάνει το σύστημα πιο σταθερό. Στο επίπεδο του συστήματος, υπάρχει σαφής διαχωρισμός μεταξύ της διεπαφής χρήστη, με την λογική του back-end. Παράλληλα στο ίδιο το back-end βρίσκουμε διαχωρισμούς, όπως την κάθε μεμονωμένη υπηρεσία συμπεριέχοντας και τις υλοποιήσεις αναζήτησης της κάθε μηχανής αναζήτησης που θέλουμε να υποστηρίξουμε. Αυτά ονομάζονται “agents” τα οποία από μόνα τους μπορούν έπειτα να διαχωριστούν αναλόγως τη λειτουργία τους. Η διεπαφή χρήστη επίσης διαχωρίζεται σε μέρη που ονομάζονται “tools” οπου το κάθε tool αποτελεί μια λειτουργία του front-end συστήματος. Στο σχήμα 3.1 περιγράφεται σχηματικά η αρχιτεκτονική που αποτελεί το ezDL



Σχήμα 3.1: Αρχιτεκτονική της ezDL

3.2.1 Back-end αρχιτεκτονική της ezDL

Το back-end αποτελεί ένα από τα βασικότερα στοιχεία αφου από αυτό γίνεται εν τέλει η αναζήτηση και επιστροφή αιτημάτων και απαντήσεων. Το back-end ουσιαστικα αποτελεί το κυρίως πρόγραμμα με το οποίο θα επικοινωνούν πολλοί clients μεσω της front-end εφαρμογής. Εκεί γίνεται η επικοινωνία με την βάση δεδομένων και των λειτουργιών που σχετίζονται με αυτή, όπως η σύνδεση του χρήστη απο το front-end στο backend, αποθήκευση ιστορικού χρήστη, caching αποτελεσμάτων κλπ. και έπειτα η επικοινωνία με τους agents για την αναζήτηση αποτελεσμάτων και την επιστροφή τους.

Η βασική ροή εργασίας κατα την εκτέλεση ενός ερωτήματος απο το front-end περιγράφεται παρακάτω

- Αποστολή ερωτήματος από τον client στον MTA (Message Transfer Agent) με τον οποίο είναι συνδεδεμένος. Ο MTA Agent είναι ουσιαστικά ο σύνδεσμος μεταξύ του client και του back-end. Κάθε MTA είναι υπεύθυνος για την πιστοποίηση χρηστών και τη μεταφορά μηνυμάτων από και πρός το back-end. Η σύνδεση γίνεται με μεσω binary πρωτοκόλλου με TCP σύνδεση.
- Αφού λάβει το ερώτημα ο MTA, το προωθεί στον Search Agent. Ο Search agent είναι υπεύθυνος για την διαχείριση του φόρτου εργασίας του συστήματος.
- Ο Search Agent ζητάει από το Directory Agent τα ονόματα των Wrapper Agents και αποστέλλει τα αποτελέσματα σε αυτούς. Ο Directory Agent είναι μοναδικός στο σύστημα και έχει σκοπό τη καταγραφή οποιουδήποτε άλλου agent και των υπηρεσιών του ωστε να γίνει σωστά η δρομολόγηση του κάθε αιτήματος στην στον κατάλληλο Wrapper Agent. Ένας Wrapper Agent είναι η υπηρεσία υπεύθυνη για την διαχείριση του ερωτήματος, της επικοινωνίας με μια πηγή μηχανής αναζήτησης, και την επιστροφή των αποτελεσμάτων στην μορφή που χρειάζεται.
- Ο κάθε Wrapper Agent λαμβάνει εν τελει το ερώτημα και το αποστέλλει. Έπειτα τα αποτελέσματα της μετατρέπονται σε μια συλλογή που την στέλνει πίσω στον Search Agent.
- Ο Search Agent συλλέγει στο τέλος όλες τις συλλογές απαντήσεων από κάθε Wrapper Agent, επεξεργάζεται τις διπλοεγγραφές και έπειτα αξιολογεί τα αποτελέσματά τους για προτεραιότητα εμφάνισης. Η αξιολόγηση αυτή γίνεται με το σύστημα RSV ή με λειτουργία που προσφέρει η Lucene. Επίσης για να αποθηκευτούν τα αποτελέσματα αυτά για γρηγορότερη μελλοντική αναζήτηση αποστέλλονται τα αποτελέσματα στον Repository Agent και στον Query History Agent
- Τέλος, η τελική συλλογή αποστέλλεται στον MTA agent που αιτήθηκε την αναζήτηση ο οποίος στην τελική επικοινωνεί με το front-end για την αποστολή του.

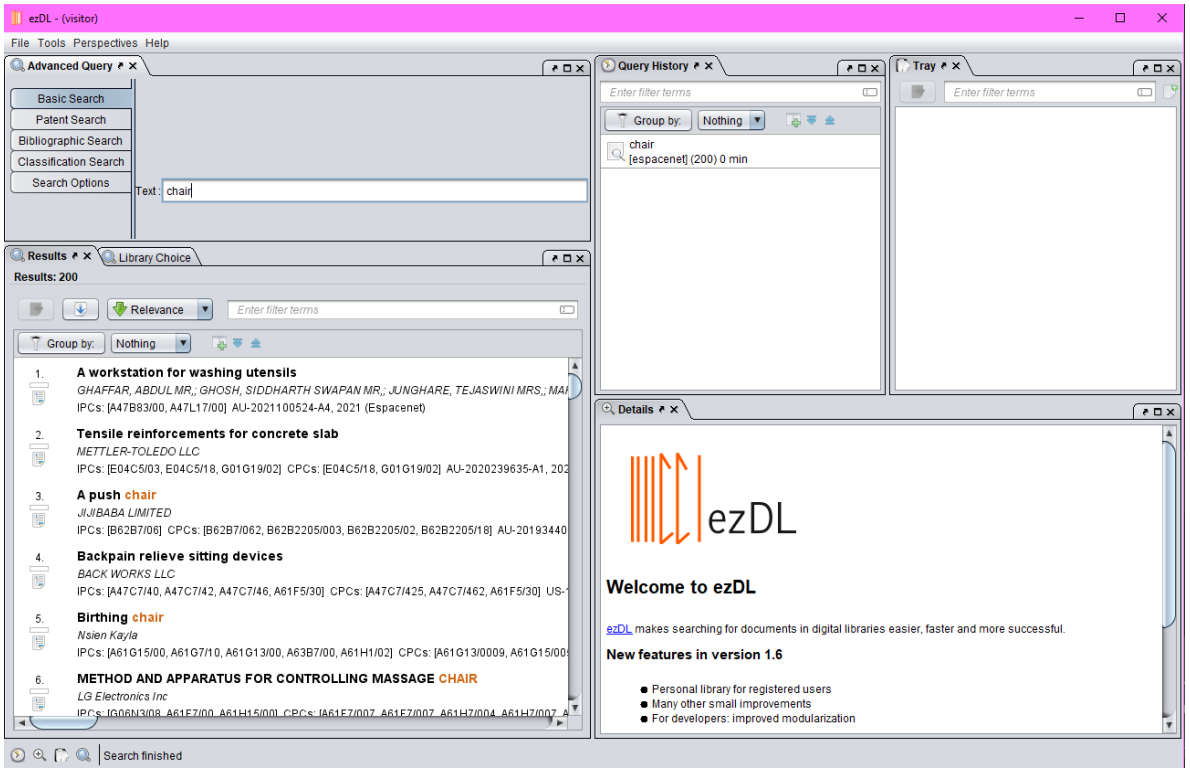
3.2.2 Front-end αρχιτεκτονική της ezDL

Το ezDL προσφέρει front-end εφαρμογή σε desktop και mobile εφαρμογή σε Android. Θα γίνει μια περεταίρω περιγραφή στην desktop εφαρμογή διότι αποτελεί τον πιο συχνό client της ezDL. Το front-end είναι γραμμένο σε Java με χρήση της βιβλιοθήκης Java FX.

Η κύρια διεπαφή χρήστη που προσφέρεται, αποτελείται από διάφορα κομμάτια ή tabs όπως φαίνεται στο σχήμα. Τα βασικότερα απο αυτά είναι:

- Search ή Query Tab. Σε αυτό το tab ο χρήστης εισάγει ερωτήματα τα οποία θέλει να αναζητήσει. Αναλόγως την υποστήριξη της τελικής εφαρμογής που θα βασιστεί στο μοντέλο της ezDL, υπάρχουν και επίσης οι επιπρόσθετες επιλογές.

- Results Tab. Σε αυτό το tab ο χρήστης θα δει την λίστα με τα αποτελέσματα της αναζήτησης του. Ο χρήστης έχει την δυνατότητα να αλλάξει τον τρόπο εμφάνισης της σειράς η να τα εμφανίσει ανα κατηγορίες, αναλόγως την υλοποίηση.
- Details Tab. Σε αυτό το tab, ο χρήστης μπορεί να δει περισσότερες πληροφορίες για ένα από τα επιλεγμένα αποτελέσματα που επιστράφηκαν στον Result Tab.



Σχήμα 3.2: Παράδειγμα διεπαφής ενός ezDL προγράμματος

Η επικοινωνία του front-end με του backend γίνεται με χρήση μιας εσωτερικής υποδομής επικοινωνίας που ονομάζεται Internal Messaging Dispatcher. Ο Dispatcher είναι ο κεντρικό κρίκος για κάθε λειτουργία του front-end. Οποιαδήποτε επικοινωνία του front-end με τον backend γίνεται μέσω της λειτουργίας “Communication”. Σε αυτό το κομμάτι το front-end έχει οποιοδήποτε ερώτημα δοθεί συμπεριλαμβανομένου και των ερωτημάτων για αναζήτηση, αλλα και ερωτήματα σχετικα με μεταδεδομένα του ezDL όπως για παράδειγμα οι διαθέσιμες πηγές αναζήτησης.

Τα ερωτήματα που μπορεί να δημιουργήσει ο χρήστης, εκτός από τη χρήση μονομερών λέξεων και φράσεων, μπορεί να γίνει συνδυασμός των λέξεων κλειδιών “AND”, “OR”, “NOT” και τελεστές “NEAR” για εύρεση απόστασης λέξεων μεταξύ τους. Για παράδειγμα:

```
Type=car AND Year<=2010 AND (Model=Audi OR Model=Ford)
```

Το ezDL εσωτερικά, αναπαριστά αυτά τα ερωτήματα σε μορφή δένδρων με κομβους (trees and tree-nodes) .

Κεφάλαιο 4ο: PerFedPat

4.1 Τι είναι το PerFedPat

Το σύστημα του PerFedPat είναι ένα σύστημα αναζήτησης για τον τομέα των πατεντών που δημιουργήθηκε χρησιμοποιώντας το framework της ezDL. Με τη βοήθεια λοιπόν του ezDL, το PerFedPat καταφέρνει να δώσει σε έναν χρήστη πρόσβαση σε αναζήτηση πολλαπλών πηγών δεδομένων για πατέντες. Αυτές οι πηγές είναι ουσιαστικά ήδη υπάρχουσες ιστοσελίδες, συγκεκριμένα άλλες μηχανές αναζήτησης. Χρησιμοποιώντας αυτές τις μηχανές αναζήτησης, το PerFedPat καθιστά μια Federated Search μηχανή η οποία στέλνει τα ερωτήματα της σε αυτές τις μηχανές, και λαμβάνει τα αποτελέσματα για να τα εμφανίσει στον χρήστη. Ο χρήστης χρησιμοποιεί το front-end πρόγραμμα για να μπορέσει να κάνει ερωτήματα σχετικα για πατέντες και να δει τα αποτελέσματα.

4.2 Τεχνολογίες που χρησιμοποιούνται

4.2.1 Java

Η Java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού γενικού σκοπού με πλήρη χαρακτηριστικά, η οποία μπορεί να χρησιμοποιηθεί για να αναπτυχθούν διάφορες εφαρμογές. Στην σημερινή εποχή χρησιμοποιείται για διάφορους σκοπούς όπως δημιουργία διαδικτυακών εφαρμογών back-end αλλα και άλλων αυτόνομων εφαρμογών με διεπιφάνειες χρήστη. Στο PerFedPat γίνεται η χρήση της έκδοσης Java 1.7. Η Java 1.7 κυκλοφόρησε το 2011 με τελευταία έκδοση της το 2014. Λόγω της συνεχής ανάπτυξης του Web και των ασφαλειών σχετικά με τα πρωτόκολλα που χρησιμοποιούνται σήμερα, η χρήση της Java 1.7 για διαδικτυακές εφαρμογές μπορεί να αποτελέσει εμπόδιο στην εποχή της δεκαετίας του 2020. Ένα τέτοιο εμπόδιο είναι η μη υποστήριξη καινούριων κρυπτογραφικών αλγορίθμων, που απαιτείται η χρήση τους απο κάποιες νεότερες ιστοσελίδες για την σύνδεσή τους.

4.2.2 Apache Maven

Το Apache Maven είναι ένα ολοκληρωμένο εργαλείο διαχείρισης και αυτοματοποίησης διαδικασιών μεταγλώττισης, παραγωγής και ανάπτυξης λογισμικών προγραμμάτων. Το Maven αποτελεί ένα σύνολο λειτουργιών χρήσιμες για τον κύκλου παραγωγής ενός λογισμικού. Χάρη μεγάλο σύνολο επιπλέον συστατικών λογισμικού που μπορούν να προστεθούν στο κυρίως πρόγραμμα, τα λεγόμενα plug-ins, το Maven παρέχει ένα επαλήθευσης, μεταγλώττισης, ελέγχου, πακεταρίσματος,αναφοράς και ανάπτυξης προγραμμάτων.

Βασικό χαρακτηριστικό στη χρήση του Maven στο PerFedPat, είναι η δυνατότητα δημιουργίας ενός multi-module project, που σημαίνει πως το κυρίως project μπορεί να αποτελείται από άλλα μικρότερα υπο-project που συνδέονται μεταξύ τους. Όπως είχε περιγραφεί στην αρχιτεκτονική του ezDL (και έπειτα του PerFedPat), το back-end είναι δομημένο απο διάφορες υπηρεσίες όπως οι Wrappers. Στο PerFedPat για παράδειγμα, κάθε Wrapper αποτελεί και ουσιαστικά την υλοποίηση του συστήματος για εκπόνηση ερωτοαπαντήσεων σε μια συγκεκριμένη μηχανή αναζήτησης πατεντών. Οι ρυθμίσεις αυτές γίνονται σε ένα xml αρχείο το οποίο το Maven μπορεί να αναγνωρίσει.

Ένα άλλο βασικό χαρακτηριστικό είναι τα Repositories. Για την ανάπτυξη ενός πλήρες λογισμικού, απαιτείται πολλές φορές η χρήση τρίτων βιβλιοθηκών διαθέσιμες για το κοινό από διάφορα Repositories. Αυτά τα Repositories είναι ουσιαστικά cloud υπηρεσίες διαθέσιμες για όλους τους προγραμματιστές με σκοπό την άμεση πρόσβαση σε ποικιλία βιβλιοθηκών. Με τη χρήση των

ρυθμίσεων του Maven, μπορούμε να δηλώσουμε την βιβλιοθήκη στην οποία ενδιαφερόμαστε να έχουμε πρόσβαση, και την πηγή της. Μερικές από τις πιο γνωστές πηγες Repositories είναι το mvnRepository και το maven Central. Στο PerFedPat για παράδειγμα, σε κάποιο υπο-project θα θέλουμε να χρησιμοποιήσουμε την βιβλιοθήκη της Apache "httpclient" που δίνει διάφορες δυνατότητες για ευκολότερη σύνδεση http. Αυτό μπορεί να γίνει βάζοντας για παράδειγμα το παρακάτω "dependency"

```
<dependency>
        <groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpclient</artifactId>
        <version>4.5.1</version>
 </dependency>
```

4.2.3 Lucene

Η Lucene είναι μια βιβλιοθήκη open-source για Java εφαρμογές, που χρησιμοποιείται για την ανάπτυξη μηχανών αναζήτησης. Η Lucene μπορεί να χρησιμοποιηθεί σε μια εφαρμογή που χρειάζεται ευρετήριο πλήρους κειμένου (full text indexing) και δυνατότητες αναζήτησης του. Είναι μια απο τις πιο γνωστές και διαδεδομένες βιβλιοθήκες που χρησιμοποιείται σε διαδικτυακές μηχανές αναζήτησης και μηχανές αναζήτησης τοπικών δεδομένων. Δίνεται επίσης η δυνατότητα εφαρμογής ασαφών αναζητήσεων (fuzzy search) βασισμένο στην απόσταση Levenshtein, η οποία βοηθά στην εμφάνιση αποτελεσμάτων ακόμα και αν το κείμενο που δόθηκε κατα την αναζήτηση περιέχει ορθογραφικά ή άλλα λάθη. Η Lucene δίνει την δυνατότητα επίσης να προσφέρει συστάσεις κατα την αναζήτηση.

Στο PerFedPat, η Lucene χρησιμοποιείται για την αξιολόγηση των αποτελεσμάτων μιας αναζήτησης χρήστη. Παραδείγματα κλάσεων στο project που χρησιμοποιείται έιναι η LuceneRanker.java LuceneQueryConverter.java DLObjectQueryHandler.java κ.α.. Αυτές οι κλάσεις έχουν σκοπό, εφόσον το υποστηρίξει ο Wrapper, την ανάδειξη των αποτελεσμάτων με βάση την σχετικότητα τους απο το κείμενο που δόθηκε για αναζήτηση.

4.2.4 XML/HTML Parsing

Η πιο συχνή μορφή που παρουσιάζεται στον χρήστη μια πληροφορία στο διαδίκτυο, είναι με την χρήση HTML ή XML. Το ίδιο συνήθως συμβαίνει και σε πολλές ιστοσελίδες μηχανών αναζήτησης. Ο χρήστης καλεί μια ιστοσελίδα με σκοπό να φέρει αποτελέσματα τα οποία πιθανότατα επιστρέφονται σε HTML μορφή. Αυτό σημαίνει πως υπάρχει μια συγκεκριμένη μορφοποίηση του κειμένου HTML από την οποία μπορούμε να λάβουμε για παράδειγμα τα αποτελέσματα μιας αναζήτησης.

Στο PerFedPat, χρησιμοποιούνται διάφορες τεχνικές προσπέλασης ενός HTML κειμένου, με σκοπό να ληφθεί και να επεξεργαστεί η πληροφορία των αποτελεσμάτων που μας ενδιαφέρουν. Μια απο αυτή είναι η τεχνική ανάκτησης με xPath ερωτήματα. Με αυτό, μπορούμε να ανακτήσουμε συγκεκριμένες πληροφορίες που μας αφορούν, δίνοντας ένα "path", δηλαδή μια κωδικοποιημένη οδηγία για το που θα βρεθεί η πληροφορία. Όπως φαίνεται στο παρακάτω παράδειγμα XML και το σχήμα 4.1, αν θελήσουμε να λάβουμε όλους τους τίτλους των βιβλίων θα μπορούσαμε να βρούμε την πληροφορία με το παρακάτω xPath:

```
/bookstore/book/title[text()]
```

```xml
<book category="cooking">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>

<book category="children">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

<book category="web">
  <title lang="en">XQuery Kick Start</title>
  <author>James McGovern</author>
  <author>Per Bothner</author>
  <author>Kurt Cagle</author>
  <author>James Linn</author>
  <author>Vaidyanathan Nagarajan</author>
  <year>2003</year>
  <price>49.99</price>
</book>

<book category="web">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>

</bookstore>
```

Σχήμα 4.1: Απόσπασμα XML αρχείου

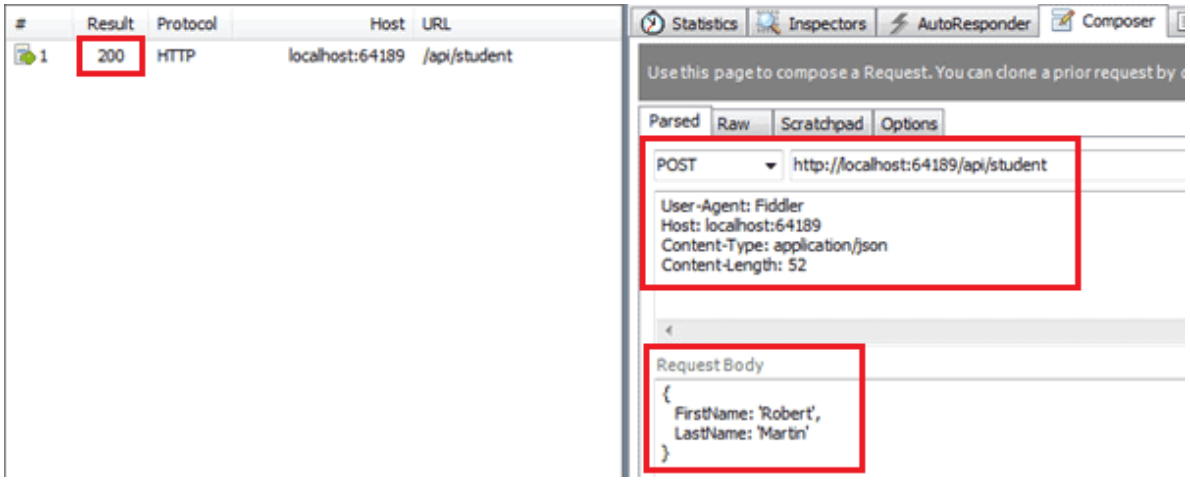
Με αυτόν τον τρόπο και με χρήση βιβλιοθηκών που υποστηρίζουν xPath αναζήτηση, μπορούμε να λάβουμε την πληροφορία. Όπως φαίνεται, ψάχνουμε πρώτα τα tags με όνομα “bookstore” τα οποία

έχουν μέσα “book” και έπειτα “title”. Τέλος λαμβάνουμε όλους τους τίτλους χρησιμοποιώντας το “texr()” που σημαίνει θα μας δοθεί μια λίστα από τίτλους.

Μερικές βιβλιοθήκες που μπορούν να χρησιμοποιηθούν για τον λόγο αυτό είναι η jsoup και η Jericho, ενω η Java υποστηρίζει επίσης από την έκδοση 1.5 και μετά, την προκαθορισμένη κλάση “XPath” για αναζητήσεις xPath.

4.2.5 APIs και JSON

Στις σύγχρονη επικοινωνία διακομιστή με χρήστη με χρήση HTTP πρωτοκόλλων, θα δούμε πως πολλές ιστοσελίδες , διανέμουν το περιεχόμενό τους δυναμικά με χρήση APIs. Το API είναι ακρωνύμιο του “Application Programming Interface”. Τα APIs συνεπώς είναι μια σειρά από λειτουργίες και διαδικασίες που επιτρέπουν στην δημιουργία εφαρμογών στις οποίες δίνεται πρόσβαση σε δεδομένα και λειτουργίες από άλλες εφαρμογές, υπηρεσίες ή λειτουργικά συστήματα. Η πιο γνωστή χρήση των APIs στο web έιναι η χρήση REST APIs. Με αυτά τα APIs δίνεται η δυνατότητα να γίνει “κλήση” μιας λειτουργίας χρησιμοποιώντας “endpoints” τα οποία μια υπηρεσία διαθέτει. Με αυτόν τον τρόπο, η πληροφορία μπορεί να επιστρέφεται με δυναμικό τρόπο στις ιστοσελίδες, αφου δεν χρειάζεται πια η πληροφορία να βρίσκεται πάνω στο αρχικό HTML αρχείο διότι μπορεί να εμπλουτιστεί αργότερα μεσω κλήσης API δυναμικά.



Σχήμα 4.2: Παράδειγμα API call

Όπως φαίνεται στο σχήμα 4.2, η χρήση του API γίνεται με αίτηση χρησιμοποιώντας το HTTP πρωτοκολλο σε έναν διακομιστή με το URL που δίνεται διαθέσιμο. Στο παράδειγμα δίνεται η δυνατότητα εύρεσης ενός μαθητή. Σε αυτό δίνουμε τις HTTP (όπως Content-Type) και στο Request Body δίνονται τα πεδία που απαιτούνται η υποστηρίζονται σε μορφή που υποστηρίζεται, όπως JSON, XML ή άλλο Content-Type που υποστηρίζει το Endpoint. Σε πολλές υπηρεσίες API, απαιτείται προηγουμένως η επαλήθευση λογαριασμού χρήστη.

Το Endpoint στη τελική, θα απαντήσει με κάποιο περιεχόμενο (ή και με απλώς ένα “OK” οπου αυτό στο HTTP είναι ο κωδικος 200) το οποίο μπορεί να είναι και αυτό σε αντίστοιχη μορφή JSON, XML κ.α. Αυτή η επικοινωνία έπειτα χρησιμοποιείται για εμφάνιση δυναμικού περιεχομένου σε ιστοσελίδες, στις οποίες με χρήση Javascript ή άλλης script γλώσσας, λαμβάνονται δυναμικά οι πληροφορίες, και η εμφάνισή τους γίνεται με όποιον τρόπο επιθυμεί το front-end.

Αυτός ο τρόπος όμως είναι ιδανικός και για άλλες λειτουργίες πέρα από εμφάνιση περιεχομένου σε ιστοσελίδες. Μπορεί να χρησιμοποιηθεί το ίδιο API και απο τρίτες εφαρμογές, είτε υπάρχει ένα Public API, με οδηγίες για τη χρήση του (Documentation), είτε βλέποντας την υπάρχουσα επικοινωνία που συμβαίνει σε μια ιστοσελίδα, στην οποία μπορούμε να χρησιμοποιησουμε το "ανεπίσημο" API (εάν έχουν). Το αρνητικό σε APIs τα οποία δεν είναι επίσημα υποστηριζόμενα για το κοινό (διότι μπορεί να υπάρχουν μόνο για χρήση τους σε μια ιστοσελίδα μόνο), είναι ότι μπορούν να αλλάξουν απρόσμενα χωρίς κάποια ενημέρωση, με αποτέλεσμα η εφαρμογή που το χρησιμοποιεί να μη λειτουργήσει σωστά και να χρειάζεται επαναπρογραμματισμός της.

4.2.6 Headless Browsers

Οι Headless Browsers (Περιηγητές χωρίς "κεφαλή") είναι προγράμματα περιήγησης ιστού χωρίς το γραφικό περιβάλλον χρήστη που συνήθως υπάρχει. Είναι δηλαδή ενα πρόγραμμα περιήγησης ή κομμάτι λογισμικού το οποίο έχει πρόσβαση σε ιστοσελίδες χωρις να τις εμφανίζει (τρέχουν στο παρασκήνιο).

Η χρήση ενός Headless Browser γίνεται συνήθως για αυτοματοποίηση λειτουργιών πάνω σε έναν περιηγητή. Η πιο συχνή χρήση του είναι για testing ιστοσελίδων όπως για παράδειγμα με την χρήση του Selenium για αυτοματοποιημένα τεστ μια ιστοσελίδας. Αυτό μπορεί να ελέγχει κάποιες λειτουργίες μιας ιστοσελίδας, για παράδειγμα μια διαδικασία Log in, και έπειτα να ελέγχει αν η σελίδα έχει εμφανίσει με σωστό τρόπο το αναμενόμενο αποτέλεσμα. Με αυτον τον τρόπο, δεν χρειάζεται συνεχές testing λειτουργιών ιστοσελίδας κάθε φορά που γίνεται μια αλλαγή, αφου την δουλεια αυτή θα την αναλάβει το αυτοματοποιημένο test που θα τρέξει εναν Headless Browser.

Τα περισσότερα γνωστά προγράμματα περιήγησης προσφέρουν λειτουργία headless, όπως το Google Chrome και το Mozilla Firefox χωρίς επιπρόσθετη εγκατάσταση.

Ένας Headless Browser, πέρα από τις λειτουργίες testing που μπορεί να χρησιμοποιηθεί, μπορεί να χρησιμοποιηθεί και για αυτοματοποίηση λειτουργιών με τελικό σκοπό το HTML Scrapping δεδομένων που μας ενδιαφέρουν. Ουσιαστικα με χρήση βιβλιοθηκών, ένα πρόγραμμα μπορεί να ελέγξει τον Headless Browser (όπως η βιβλιοθήκη Selenium). Για παράδειγμα μπορούμε να στήσουμε ένα πρόγραμμα να ανοίγει εναν Headless Browser, να πηγαίνει σε μια ιστοσελίδα που θέλουμε, να "γράψει" πάνω στη μπάρα text-box, και να κάνει μια αναζήτηση. Έπειτα, προγραμματιστικά έχουμε πρόσβαση στα δεδομένα που εμφανίζονται, και μπορούμε να τα λάβουμε για τον σκοπό μας (με κάποιο τρόπο HTML scrapping).

Η χρήση Headless Browser αποτελεί πολλές φορές ο μόνος τρόπος να μπορέσουμε να λάβουμε πληροφορίες από μια ιστοσελίδα αυτοματοποιημένα. Πολλές φορες οι ιστοσελίδες χρησιμοποιούν AJAX (Asynchronous Javascript and XML) που σημαίνει πως η πληροφορία μπορεί να μην είναι άμεσα διαθέσιμη μόνο με κλήση μιας ιστοσελίδας με παραμέτρους ή μόνο με χρήση API. Είναι πιθανό να δημιουργούνται δυναμικά κάποιες λειτουργίες, μη προβλεπόμενες, για να μπορέσει να γίνει η επικοινωνία σωστά με τον server. Το μειονέκτημα της χρήσης του είναι πως χρειάζεται πάντα ένας Headless Browser να τρέχει στο παρασκήνιο, και επίσης η αδυναμία παράκαμψης εμποδίων ελέγχου "ρομπότ" όπως για παράδειγμα οι έλεγχοι Captcha.

Κεφάλαιο 5ο: EspaceNet

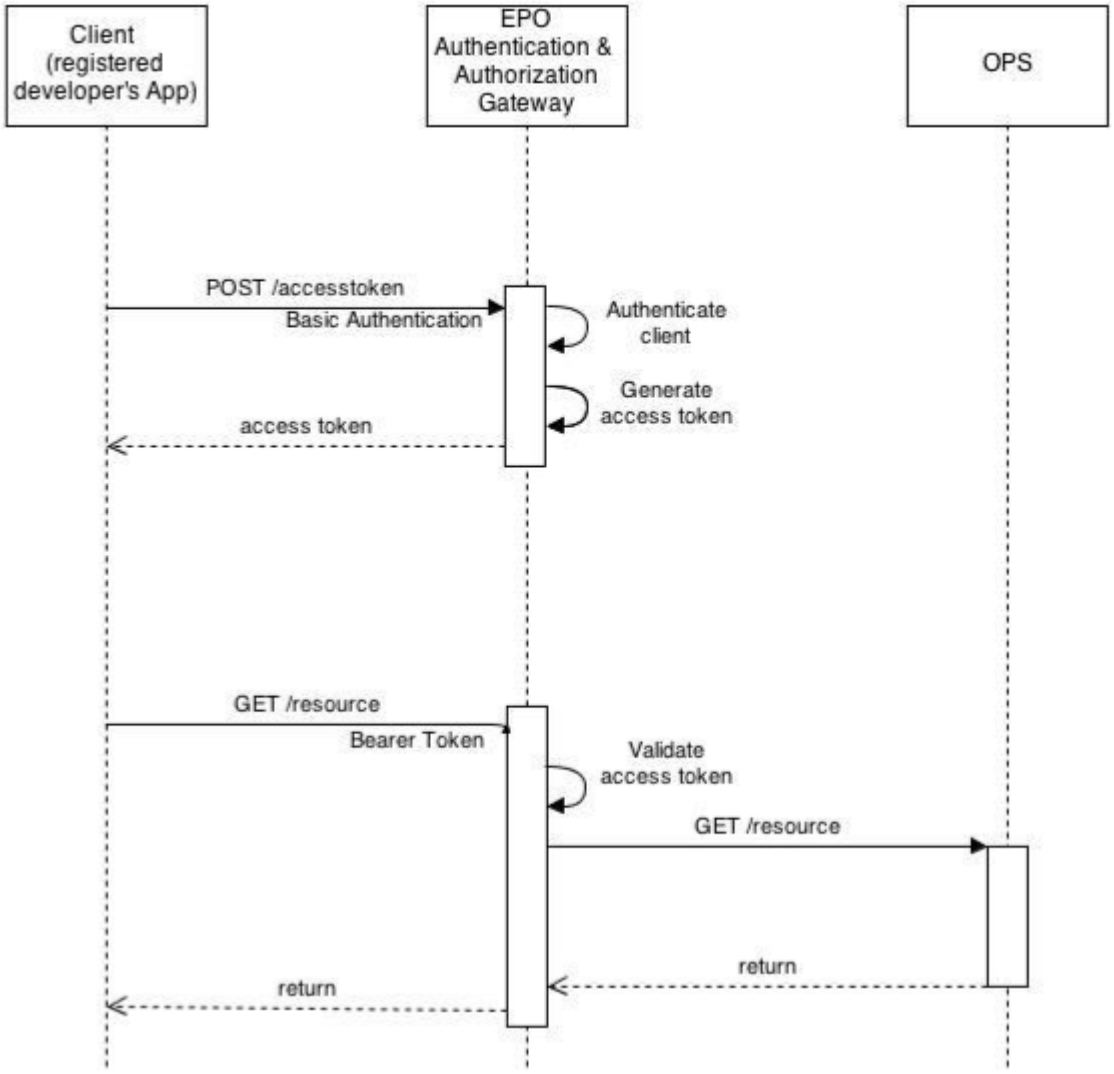
5.1 Εισαγωγή στην EspaceNet

Η EspaceNet είναι μια δωρεάν πλατφόρμα αναζήτησης πατεντών απο την EPO, το Ευρωπαϊκό Γραφείο Πατεντών που αναπτύχθηκε μαζί με μέλη της Ευρωπαϊκής Οργάνωσης Πατεντών. Η EspaceNet δίνει την δυνατότητα σε προγραμματιστές να έχουν πρόσβαση στις υπηρεσίες αναζήτησης των πατεντών αυτών μέσω του API που προσφέρει ονομαζόμενο Open Patent Service (OPS). Η χρήση του API είναι ανοιχτή για χρήστες μετα απο αίτηση για λογαριασμό. Στις υπηρεσίες που προσφέρει το OPS API περιλαμβάνει τις κύριες που χρησιμοποιεί το PerFedPat, δηλαδή την αναζήτηση πατεντών μεσω κειμένου ή συγκεκριμένων στηλών όπως χρονολογία, εφευρέτης κλπ.

5.2 Το API του EspaceNet

Όπως προαναφέρθηκε, το API του EspaceNet είναι το Open Patent Services RESTful Web Services και είναι ανοιχτό προς το κοινό μετα απο αίτηση. Ο τρόπος με τον οποίο γίνεται η χρήση του API είναι με REST calls. Το API προσφέρει οδηγίες με τις οποίες μπορούμε να δούμε τι απαιτείται σε κάθε κλήση του API.

Συγκεκριμένα για την χρήση του API στο PerFedPat χρειαζόμαστε για αρχή να επιβεβαιώσουμε τον λογαριασμό χρήσης του OPS API. Απο τη στιγμή που έχουμε πρόσβαση σε έναν λογαριασμό EspaceNet, έχουμε τη δυνατότητα να κάνουμε και αυθεντικοποίηση με την υπηρεσία OPS API. Αυτό γίνεται με τη χρήση ενός Access Token που θα λάβουμε αφού γίνει αυθεντικοποίηση με OAuth. Στο σχήμα 5.1 παρουσιάζεται η διαδικασία με την οποία συμβαίνει η σύνδεση με OAuth στο EspaceNet.



Σχήμα 5.1: Σύνδεση OAuth στο EspaceNet

Κάθε λογαριασμός EspaceNet έχει την δυνατότητα δημιουργίας ενός ζεύγου Public Key και Secret Key. Με αυτά, τα αποστέλλουμε στο POST /accesstoken endpoint για να μπορέσουμε να λάβουμε το Access Token το οποίο απαιτείται σε οποιαδήποτε άλλο endpoint του OPS. Η αποστολή των keys γίνεται αφου μετατρεψουμε το ζεύγος αυτών των κλειδιών σε ένα String με Base64 μορφή. Δηλαδή για παράδειγμα αν εχουμε το ζεύγος:

```
y3AOG86qwqjU0QU69VYGTJ4JGxUN8EVG:rrXdr5WA7x9tudmP
```

Αυτό θα μετατραπεί σε Base64 στη παρακάτω μορφή:

```
eTNBT0c4NnF3cWpVMFFVNjlWWUdUSjRKR3hVTjhFVkc6cnJYZHI1V0E3eDl0dWRtUA==
```

το οποίο θα αποστείλουμε για αυθεντικοποίηση.

Το API Endpoint που χρησιμοποιούμε για αυτην την αυθεντικοποίηση περιγράφεται στο σχήμα 5.2

HTTPS	**POST** https://ops.epo.org/3.2/auth/accesstoken
HTTP Headers	
Authorization:	Basic **eTNBT0c4NnF3cWpVMFFVNjlWWUdUSjRKR3hVTjhFVkc6cnJYZHI1V0E3eDl0dWRtUA==**
Content-Type:	application/x-www-form-urlencoded
POST request payload	
grant_type=client_credentials	

Σχήμα 5.2: Κλήση για αυθεντικοποίηση Oauth στο EspaceNet

Η αναμενόμενη απάντηση από αυτή τη κλήση HTTP, θα περιέχει σε JSON μορφή τα δεδομένα που χρειαζόμαστε σχετικά με το access token όπως το ίδιο το access token, το χρονικό διάστημα το οποίο λήγει, κ.α.

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
  "issued_at": "1364247843353",
  "application_name": "511d82a3-aa0e-4775-ba48-05ccd9275c56",
  "scope": "",
  "status": "approved",
  "expires_in": "1199",
  "api_product_list": "[ops31-prod]",
  "organization_id": "0",
  "access_token": "4AWoepfVNgf09DRmimEnGdXcgoFU",
  "organization_name": "epo",
  "refresh_count": "0"
}
```

Σχήμα 5.3: Απάντηση αυθεντικοποίησης OAuth στο EspaceNet

Χρησιμοποιώντας το access token που μας δόθηκε, έπειτα μπορούμε να συνεχίσουμε στην αναζήτηση πατεντών. Αυτό γίνεται με τη χρήση των Endpoint που περιγράφονται στο σχήμα 5.4

```
GET http://ops.epo.org/rest-services/published-
data/search?q=applicant%3DIBM

Accept: application/exchange+xml
```

```
POST http://ops.epo.org/rest-services/published-data/search

Accept: application/exchange+xml
Request Body: q=applicant%3DIBM
```

Σχήμα 5.4: API Endpoints για αναζήτηση στο EspaceNet

Είτε γίνει η χρήση του GET Endpoint ή του POST endpoint τα αποτελέσματα είναι ίδια. Στο σχήμα 5.3 γίνεται η αναζήτηση πατεντών με 'Applicant' την 'IBM'. Ένα μέρος των αποτελεσμάτων φαίνεται στο σχήμα 5.4 σε μορφή XML

Αφού λάβουμε τα αποτελέσματα, κάνουμε Parse το XML περιεχόμενο λαμβάνοντας τα δεδομένα που μας ενδιαφέρουν, στη περίπτωση μας θα λάβουμε τη λίστα με τα αποτελέσματα που βρέθηκαν στην αναζήτηση και δεδομένα όπως όνομα πατέντας, applicant, date κ.α.

```
<?xml version="1.0" encoding="UTF-8"?>
<ops:world-patent-data xmlns:ops="http://ops.epo.org"
xmlns="http://www.epo.org/exchange">
  <ops:biblio-search total-result-count="10000">
    <ops:query syntax="CQL">applicant=IBM</ops:query>
    <ops:range begin="1" end="25"/>
    <ops:search-result>
      <exchange-documents>
        <exchange-document system="ops.epo.org" family-id="37717388"
country="KR" doc-number="20100130646" kind="A">
          <bibliographic-data>
            <publication-reference>
              <document-id document-id-type="docdb">
                <country>KR</country>
                <doc-number>20100130646</doc-number>
                <kind>A</kind>
                <date>20101213</date>
              </document-id>
              <document-id document-id-type="epodoc">
                <doc-number>KR20100130646</doc-number>
                <date>20101213</date>
              </document-id>
            </publication-reference>
            <!-- classifications removed -->
            <application-reference doc-id="331447793">
              <document-id document-id-type="epodoc">
                <doc-number>KR20107026618</doc-number>
                <date>20060720</date>
              </document-id>
              <document-id document-id-type="original">
                <doc-number>1020107026618</doc-number>
              </document-id>
            </application-reference>
            <!-- claims removed -->
            <parties>
              <applicants>
                <applicant sequence="1" data-format="epodoc">
                  <applicant-name>
                    <name>IBM [US]</name>
                  </applicant-name>
                </applicant>
                <applicant sequence="1" data-format="original">
                  <applicant-name>
                    <name>INTERNATIONAL BUSINESS MACHINES CORPORATION</name>
                  </applicant-name>
                </applicant>
              </applicants>
              <!-- inventors removed -->
            </parties>
            <invention-title lang="en">INJECTION MOLDED MICROLENSES FOR
OPTICAL INTERCONNECTS</invention-title>
          </bibliographic-data>
        </exchange-document>
```

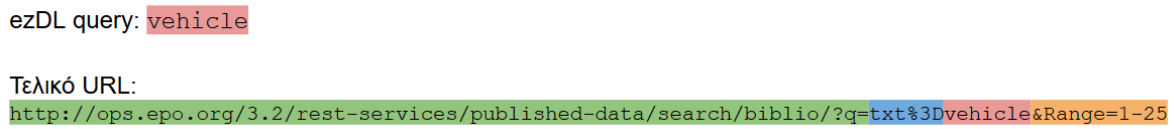
Σχήμα 5.5: Απόσπασμα XML αποτελέσματος αναζήτησης στο EspaceNet

5.3 Δημιουργία ezDL queries για την EspaceNet

Για τη δημιουργία του τελικού Endpoint URL το οποίο θα χρησιμοποιήσουμε για να λάβουμε τη λίστα αποτελεσμάτων θα χρειαστεί να "χτίσουμε" το τελικό URL το οποίο θα καλέσουμε. Σύμφωνα με τις

οδηγίες που δόθηκαν για να γίνει αυτό, θα χρειαστεί το "base" URL να είναι το `http://ops.epo.org/3.2/rest-services/published-data/search/biblio/`

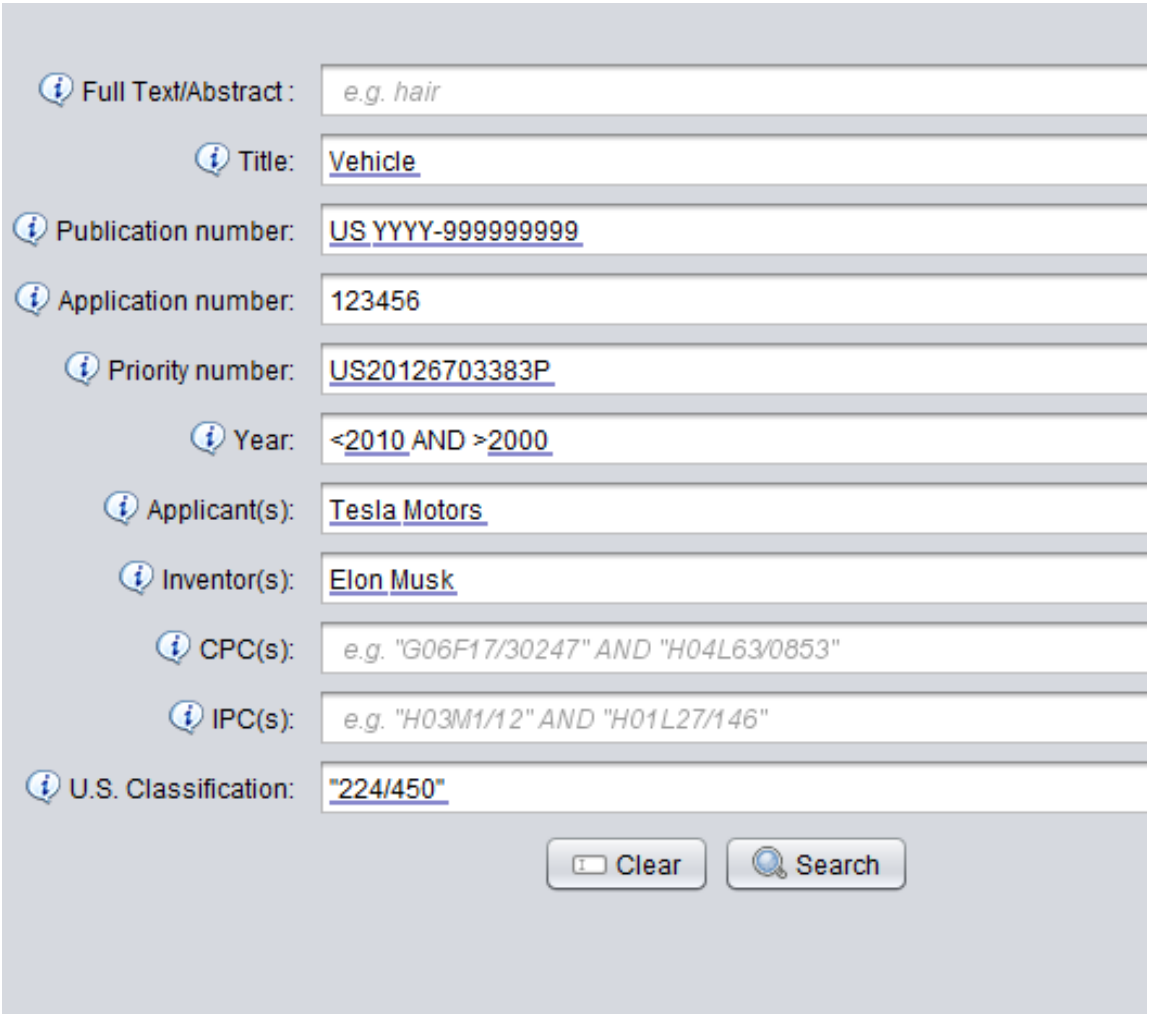
Έπειτα, θα πρέπει να μεταφράσουμε το τι αναζητούμε απο την μορφή που παράγει το ezDL στη μορφή που μπορεί να αναγνωρίσει το EspaceNet. Στη περίπτωση του EspaceNet, αρκεί να προσθέσουμε ως url variables το τι θέλουμε να αναζητήσουμε. Σε μια απλή text αναζήτηση για παράδειγμα "`vehicle`" αυτό που θα γίνει είναι να προστεθεί ως URL παράμετρος το "`?q=`" που σημαίνει πως ακολουθεί ερώτημα, και μετα θα προστεθεί το query μας με τον μεταφρασμένο τρόπο. Για αρχή στο σχήμα 5.6 φαίνεται πως το μόνο που χρειάζεται ακόμα είναι η προσθήκη του '`txt%3D`' (διότι υποθέτουμε πως κάνουμε αναζήτηση κειμένου) και δίπλα την αναζήτησή μας τον όρο αναζήτησης μας '`vehicle`'. Έπειτα προσθέτουμε και τη παράμετρο για τα πόσα αποτελέσματα να εμφανιστούν. Στο παράδειγμα μας θέλουμε να εμφανιστούν τα αποτελέσματα απο 1 μεχρι 25 οπότε προσθέτουμε το '`Range=1-25`'.



Σχήμα 5.6: Μετατροπή ezDL query σε URL EspaceNet

Σε μία πιο πολύπλοκη όμως αναζήτηση, στην οποία υποστηρίζουμε διάφορα πεδία όπως Applicant, Year κλπ. θα χρειαστεί να μεταφράσουμε και τα πεδία και λέξεις κλειδιά του ezDL query στα αντίστοιχα πεδία και λέξεις πεδία του EspaceNet. Για παράδειγμα στο ezDL για την αναζήτηση Applicant ορίζεται ως '`pa=`' και μετά το όνομα του Applicant.

Στο σχήμα 5.7 κάνουμε ένα παράδειγμα αναζήτησης με διάφορα πεδία που θέλουμε



Σχήμα 5.7: Παράδειγμα αναζήτησης στο PerFedPat

Εδώ όταν γίνει η αναζήτηση, τα πεδία αυτά θα μεταφραστούν σε ezDL query, και έπειτα μεσω του wrapper θα μεταφραστεί στο τελικο URL που θα καλέσουμε.

ezDL query:
```
IPC=H01L27/146 AND Priority=US20126703383P AND CPC=G06F17/30247 AND
Title=vehicle AND USPC=224/450 AND Application_Num=123456 AND Applicant=Tesla AND
Applicant=Motors AND Inventor=Elon AND Inventor=Musk AND Year<2010 AND Year>2000 AND
Publication_Num=US AND Publication_Num=YYYY-99999999
```

Τελικό URL:
```
http://ops.epo.org/3.2/rest-services/published-data/search/biblio/?q=ic%3DH01L27%2F146%20and%20s
pr%3DUS20126703383P%20and%20cpc%3DG06F17%2F30247%20and%20ti%3Dvehicle%20and%20txt%3D224%2F450%20
and%20ap%3D123456%20and%20pa%3DTesla%20and%20pa%3DMotors%20and%20in%3DElon%20and%20in%3DMusk%20a
nd%20pd%20within%202011,1999%20and%20pn%3DUS%20and%20pn%3DYYYY-99999999&Range=1-25
```

Σχήμα 5.8: Μετατροπή "advanced query" σε EspaceNet URL

Όπως φαίνεται με στο σχήμα 5.8, γίνεται η αντιστοιχία του ezDL query με το query που υποστηρίζεται απο το EspaceNet. Επίσης γίνεται επιπρόσθετη URL κωδικοποίηση έτσι ώστε το query που περιέχεται στη URL παράμετρο 'q' να μην δημιουργεί πρόβλημα με υπόλοιπες παράμετροι (όπως το Range που είναι στο τέλος)

Μερικά απο τα πεδία που αντιστοιχούνται φαίνονται στο παρακάτω σχήμα 5.9:

ezDL	EspaceNet
Text	txt
IPC	ic
Priority	spr
CPC	cpc
Title	ti
USPC	txt (not-supported)
Application_Num	ap
Applicant	pa
Inventor	in
Year	pd
Publication_Num	pn

Σχήμα 5.9: Αντιστοίχιση ezDL πεδίων με πεδία του EspaceNet

5.4 Διαχείριση και εμφάνιση αποτελεσμάτων

Στο σημείο αυτό, έχουμε καταφέρει να επικοινωνήσουμε και να αποστέλλουμε ένα ερώτημα στην μηχανή αναζήτησης της EspaceNet. Αφου γίνει αυτό, πρέπει να διαχειριστούμε την πληροφορία που επιστρέφεται. Η πληροφορία των αποτελεσμάτων που βρέθηκαν όπως περιγράφηκε προηγουμένως είναι σε μορφή XML. Το XML αυτό έχει συγκεκριμένη μορφή που βοηθάει να βρούμε πολύ εύκολα για κάθε αποτέλεσμα και τις πληροφορίες που χρειαζόμαστε.

Το XML περιέχει το tag `<ops:search-result>` το οποίο περιέχει μέσα του τη λίστα με τα αποτελέσματα. Κάθε αποτέλεσμα αποτελείται απο το tag `<exchange-documents>` . Οπότε με χρήση μιας βιβλιοθήκης XML parsing και xPath ερωτημάτων, μπορούμε να λάβουμε την λίστα. Κάθε αποτέλεσμα περιέχει περαιτέρω XML tags με πληροφορίες που ενδιαφερόμαστε. Για παράδειγμα τον τίτλο θα τον βρούμε κάτω από το tag `<invention-title lang="en">` . Παρατήρηση χρειάζεται πως κάθε tag μπορεί να περιέχει και κάποιες επιπρόσθετες επιλογές όπως το lang στη περίπτωσή μας που δείχνει πως ο τίτλος είναι σε Αγγλική γλώσσα.

Έπειτα αφου λάβουμε τα δεδομένα, τα εισάγουμε σε ένα Object τύπου 'Document' της ezDL. Αυτό αποστέλλεται στο front-end και γίνεται η εμφάνιση των αποτελεσμάτων σε μια λίστα με τα στοιχεία που βρέθηκαν.

Κεφάλαιο 6ο: Google Patents

6.1 Εισαγωγή στη Google Patents

Η ιστοσελίδα και υπηρεσία Google Patents (γνωστή ως και gPatents) είναι μια μηχανή αναζήτησης πατεντών απο την Google. Η Google Patents κρατούν ευρετήριο για περισσότερα από 120 εκατομμύρια πατέντες απο περισσοτερα απο 100 γραφεία πατεντών μερικά εκ των οποίων είναι τα United States Patent and Trademark Office (USPTO), European Patent Office (EPO), World Intellectual Property Organization (WIPO) και άλλα διάφορα εθνικά γραφεία πατεντών. Οι πατέντες απο τις Ηνωμένες Πολιτείες έχουν χρονολογία απο 1790 και μετά, χαρης τεχνολογιών που παρέχει η Google για αναζήτηση με χρήση Οπτικής Αναγνώρισης Χαρακτήρων σε κείμενα σαρωμένων εικόνων απο χειρόγραφες ή έντυπες πατέντες. Επίσης με τη χρήση αδελφικών Project της Google, όπως το Google Scholar και το Google Books, η Google Patents καταφέρνει να αξιοποιήσει και να ταξινομήσει πατέντες με τον κωδικό CPC (Cooperative Patent Classification).

6.2 Το ανεπίσημο API της GPatents και η χρήση του

Η ιστοσελίδα της Google Patents προσφέρει αναζήτηση με χρήση απλού κειμένου και σύνθετης αναζήτησης. Δυστυχώς, η Google Patents δεν προσφέρει επίσημη υπηρεσία API για χρήση της αναζήτησης των πατεντών για προγραμματιστές. Παρ'ολα αυτά, η επικοινωνία που γίνεται από front-end με το back-end, γίνεται με την χρήση ενος εσωτερικής χρήσης API (Internal API) το οποίο δεν υπάρχει δημόσιο Documentation. Μπορούμε όμως βέβαια να υποθέσουμε πως λειτουργεί το API αυτό ανακτώντας τις κλήσεις του API που συμβαίνουν κατα τη διάρκεια μιας "χειροκίνητης" αναζήτησης, δηλαδή την αναζήτηση που κάνουμε αν μπούμε ως ένας κοινός χρήστης στην ιστοσελίδα. Χάρης των εργαλείων που προσφέρουν τα προγράμματα περιήγησης, μπορούμε να δούμε την ώρα που γίνεται η αναζήτηση, με ποιά API endpoints επικοινωνεί η ιστοσελίδα. Έπειτα ψάχνουμε να βρούμε το API endpoint που κλήθηκε το οποίο περιέχει το query που δώσαμε για αναζήτηση και ως απάντηση περιέχει τα δεδομένα των αποτελεσμάτων που βρέθηκαν.

Σχήμα 6.1: Η διεπαφή της Google Patents

Στη περίπτωση της Google Patents αυτό συμβαίνει με την κλήση στο παρακάτω endpoint

```
GET   https://patents.google.com/xhr/query
```

Το οποίο λαμβάνει την παράμετρο ‘url’ και ‘oq’ τα οποίο συμπεριλαμβάνουν το query που θέλαμε να αναζητήσουμε. Για διάφορους λόγους, τους οποίους χρειάζεται η ίδια η google (των οποίων δεν μπορούμε να γνωρίζουμε σαφως επειδή το API είναι για χρήση εσωτερική) η παράμετρος ‘url’ περιέχει υπο παραμέτρους που ορίζουν το ερώτημα. Για απλά text ερωτήματα η υπο παράμετρος θα είναι το ‘q’ ακολουθούμενο απο τον text όρο που αναζητούμε. Για άλλους λόγους το API επίσης αποστέλλει τη παράμετρο ‘oq’ που περιέχει το πλήρες όρο που δόθηκε στη μπάρα αναζήτησης. Τέλος δίδεται η παράμετρος exp η οποία είναι πάντα κενή.

Για παράδειγμα αν κάνουμε μια απλή αναζήτηση πατεντών για “car”, το τελικό URL του API endpoint που θα αποσταλεί στη Google Patents θα είναι το παρακάτω

```
GET
https://patents.google.com/xhr/query?url=q%3Dcars%26oq%3Dcars&exp=
```

Σύμφωνα με τις οδηγίες της Google Patents. Υποστηρίζεται και η χρήση “μέτα-αναζήτησης” δηλαδή αναζήτησης συγκεκριμένων πεδίων όπως “assignee”, “before” και “inventor” κατι που στο PerFedPat υποστηρίζεται και είναι ιδανικό για στην υλοποίηση του Wrapper μας.

6.3 Δημιουργία ezDL queries για την Google Patents

Για να μπορέσει να γίνει η υλοποίηση του Wrapper για τη χρήση του Google Patents, θα πρέπει να μπορέσουμε να μεταφράσουμε τα queries που παράγονται από ένα ezDL ερώτημα, σε ερώτημα το οποίο αναγνωρίζει η Google Patents με τη χρήση του API που αναλύσαμε προηγουμένως.

Μετά από ανάλυση των “μετα-δεδομένων” που υποστηρίζονται από τη Google Patents σε σχέση με αυτών που παράγονται από το ezDL φαίνεται πως δυστυχώς δεν γίνεται συσχέτιση με όλα τα πεδία του ezDL. Τα πεδία του PerFedPat που σχετίζονται με αυτά της Google Patents είναι ο Assignee, Inventor, Date. Τα υπόλοιπα πεδία θα θεωρούνται ως απλό κείμενο αναζήτησης (Text). Αν και αυτή η λύση δεν είναι ιδανική, η Google Patents μπορεί να αναγνωρίσει εύκολα αν κάποιο μέρος του Text που αποστέλλουμε είναι όντως κάποιο Priority, IPC, CPC ή Application Number με αποτέλεσμα να εμφανίζονται και πάλι αποτελέσματα σχετικά με αυτά που θέλουμε να δούμε.

Στον σχήμα 6.2 θα δούμε πως γίνεται η συσχέτιση των πεδίων PerFedPat με αυτά στο API της Google Patents

PerFedPat	Google Patents
IPC	q
Priority	q
CPC	q
USPC	q
Application Number	q
Text	q
Applicant	assignee
Inventor	inventor
Date	publication

Σχήμα 6.2: Αντιστοίχιση PerFedPat πεδίων με πεδία της Google Patents

Με αυτόν τον πίνακα μπορούμε να δημιουργήσουμε το τελικό URL που θα χρησιμοποιηθεί για τη χρήση του API της Google Patents με την παρακάτω διασύνδεση.

ezDL query: IPC=A63F13/98 AND AND Priority=US7753795B2 AND CPC=G07F17/3241 AND USPC=US9245150B2 AND Application Num=123456 AND Game AND Console AND Applicant=Sony AND Applicant=Entertainment AND Inventor=Adam AND Inventor=Harris AND Year<2015 AND Year>2006 AND Title=Gaming AND Title=Console AND Publication_Num=US AND Publication_Num=YYYY-99999999

Τελικό URL:

https://patents.google.com/xhr/query?url=q%3DA63F13/98%2BUS7753795B2%2BUS9245150B2%2B123456%2BGame%2BConsole%2BGaming%2BConsole%2BUS%2BYYYY-99999999%26assignee%3DSony%2BEntertainment%26inventor%3DAdam%2BHarris%26before%3Dpublication%3A20150101%26after%3Dpublication%3A20060101%26num%3D100

Σχήμα 6.3: Μετατροπή ezDL query σε Google Patents URL

Όπως φαίνεται, όποια πεδία δεν υποστηρίζονται απλώς κατηγοριοποιούνται μαζί στο "q" (έμφαση στο γεγονός ότι υπάρχει URL encoding οπότε το '=' συμβολίζεται ως %3D και το '+' ως %2B). Στο τέλος του URL επίσης προστίθεται το πεδίο το οποίο μπορούμε να ορίσουμε το πλήθος των αποτελεσμάτων που θα μας επιστρέψει το API. Αυτό το ορίζουμε με το 'num' το οποίο θα μας επιστρέφει 100 αποτελέσματα.

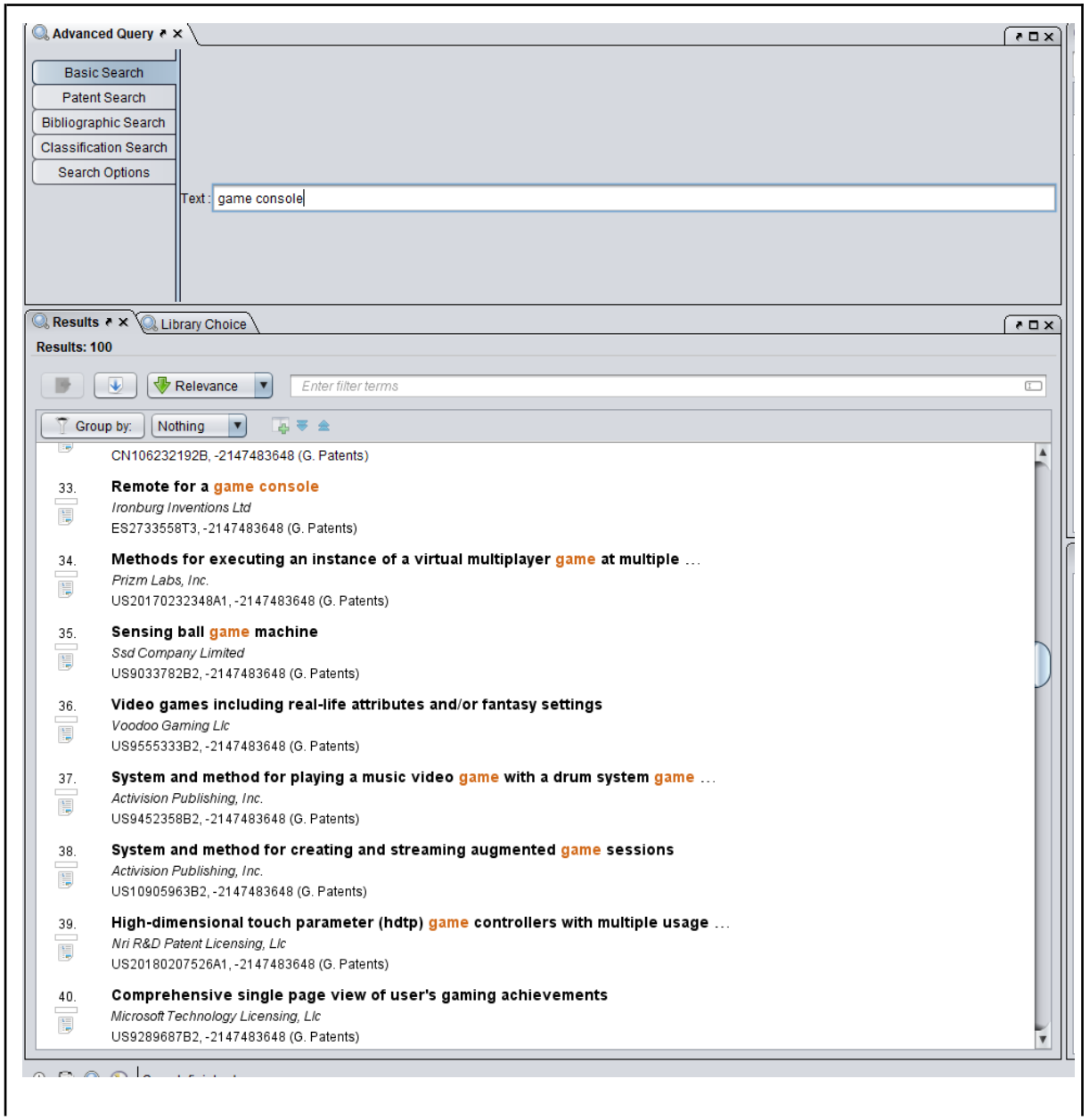
6.4 Διαχείριση αποτελεσμάτων

Τα αποτελέσματα που θα επιστραφούν, έρχονται σε μορφή JSON. Παρακάτω ακολουθεί ένα μερικό παράδειγμα ενός αποτελέσματος JSON που θα μας επιστρέψει η Google Patents.


```
{
    "results": {
        "total_num_results": 63661,
        "total_num_pages": 100,
        "many_results": false,
        "num_page": 0,
        "cluster": [
            {
                "result": [
                    {
                        "rank": 0,
                        "patent": {
                            "title": " Entertainment system, entertainment apparatus, recording medium, and program",
                            "snippet": " Some entertainment systems having entertainment apparatus which include video <b>game machines</b> ex
                            control of a manual controller while game images are being displayed on a display &hellip;",
                            "priority_date": "1999-09-02",
                            "filing_date": "2002-01-23",
                            "grant_date": "2005-01-25",
                            "publication_date": "2005-01-25",
                            "inventor": "Masaru Ohnuma",
                            "assignee": "<b>Sony</b> Computer Entertainment Inc.",
                            "publication_number": "US6846237B2",
                            "language": "en",
                            "thumbnail": "08/d2/5f/172910d0ce0c62/US06846237-20050125-D00000.png",
                            "pdf": "e9/6e/61/80c328c9e818a9/US6846237.pdf",
                            "figures": [
                            "family_metadata": {
                        },
                        "is_similar_document": true,
                        "entity_matches": {
                    },
                    {
                        "rank": 1,
                        "patent": {
                        "is_similar_document": false,
                        "entity_matches": {
                    },
                    {
                        "rank": 2,
                        "patent": {
                        "is_similar_document": true,
                        "entity_matches": {
                            "additional": 0
                        }
                    },
                    {
                        "rank": 3,
                        "patent": {
                            "title": " Invalidating network devices with illicit peripherals",
```


Σχήμα 6.4: Παράδειγμα JSON αποτελέσματος από αναζήτηση στην Google Patents

Όπως φαίνεται υπάρχουν διάφορες πληροφορίες τις οποίες θα αγνοήσουμε για τον Wrapper μας καθώς δεν μας καθιστούν χρήσιμες. Οι πληροφορίες που χρειαζόμαστε, είναι τα αποτελέσματα που βρίσκονται κάτω από το πεδίο "cluster" το οποίο έχει το πεδίο "result" που αποτελεί έναν πίνακα απο άλλα JSON Objects. Κάθε Object του πίνακα αποτελεί και ένα αποτέλεσμα. Σε κάθε αποτέλεσμα υπάρχει το πεδίο "patent" το οποίο περιέχει όλες τις πληροφορίες που τελικά χρειαζόμαστε για να εμφανίσουμε αποτελέσματα στο PerFedPat. Τα πεδία αυτά είναι το "title" ως τίτλο, "snippet" το οποίο περιέχει μια μικρή περιγραφή της πατέντας, "publication_date" ως την ημερομηνία (date), "inventor", "assignee", και "publication_number" το οποίο είναι και ο μοναδικός κωδικός τον οποίο έχει κάθε πατέντα. Συγκεκριμένα για το "publication_number" είναι πληροφορία που θα χρειαστούμε να κρατήσουμε αργότερα με σκοπό την ανάδειξη περισσότερων πληροφοριών για μια πατέντα Τέλος, χαρης αυτής της διαδικασίας τα αποτελέσματα εμφανίζονται όπως στο σχήμα 6.5



Σχήμα 6.5: Αποτελέσματα Google Patents στο PerFedPat

Στον κώδικα του Wrapper, τη μετάφραση των δεδομένων που μας δίνει η Google Patents γίνεται με χρήση της βιβλιοθήκης org.json τα οποία περιέχει χρήσιμα αντικείμενα όπως JSONObject και JSONArray. Με αυτά μπορούμε πολύ εύκολα να μετατρέψουμε ενα JSON κείμενο σε αντικείμενο τύπου JSONObject και έπειτα να βρούμε εύκολα την πληροφορία που θέλουμε και να γεμίσουμε το αντίστοιχο αντικείμενο που χρειάζεται να επιστρέψουμε του ezDL (Document). Παρακάτω στο σχήμα 6.6 παρουσιάζεται απόσπασμα υλοποίησης της ανάγνωσης αποτελεσμάτων του JSON και μετατροπή της σε Document αντικείμενα, τα οποία γεμίζουμε σε ένα WrapperDLObject.

```
JSONObject jsonObject = new JSONObject(response.toString());
JSONArray jsonResults = jsonObject.getJSONObject("results") JSONObject
        .getJSONArray( key: "cluster") JSONArray
        .getJSONObject( index: 0) JSONObject
        .getJSONArray( key: "result");

WrapperDLObjectList wrapperResults = new WrapperDLObjectList();

for (int i = 0; i < jsonResults.length(); i++) {
    JSONObject jsonPatent = jsonResults.getJSONObject(i).getJSONObject("patent");

    TextDocument document = new TextDocument();
    document.setFieldValue(Field.TITLE, getOnlyText(jsonPatent.getString( key: "title")));

    PersonList assigneeList = new PersonList();
    assigneeList.add(new Person(getOnlyText(jsonPatent.getString( key: "assignee"))));
    document.setApplList(assigneeList);

    PersonList inventorList = new PersonList();
    inventorList.add(new Person(getOnlyText(jsonPatent.getString( key: "inventor"))));
    document.setInvList(inventorList);

    document.setPubNum(getOnlyText(jsonPatent.getString( key: "publication_number")));

    document.setFieldValue(Field.PUBLICATION_YEAR, Integer.getInteger(getOnlyText(jsonPatent.getString( key: "publication_date")).substring( i: 0,  i1: 3)));

    document.setFieldValue(Field.Description, jsonPatent.getString( key: "snippet"));

    document.addDetailURL(new URL( s: "https://patents.google.com/patent/" + document.getPubNum()));

    WrapperDLObject wrapperResult = new WrapperDLObject(document, new SourceInfo(getSourceID(),  detailsInfo: "dummyLink"));
    wrapperResults.add(wrapperResult);

}
```

Σχήμα 6.6: Απόσπασμα κώδικα μετατροπής των αποτελεσμάτων

6.5 Εμφάνιση περισσότερων πληροφοριών πατέντας (Google Patents)

Το PerFedPat δίνει την δυνατότητα να εμφανιστούν επιπρόσθετες πληροφορίες για μια πατέντα εφόσων το επιθυμεί ο χρήστης επιλέγοντας ένα από τα αποτελέσματα που του εμφανίστηκαν. Για την Google Patents, οι πληροφορίες που δίνονται μονο με τη χρήση του query API δεν είναι αρκετές (όπως στη περίπτωση του EspaceNet οπου όλες οι πληροφορίες δίνονται στα ίδια τα αποτελέσματα αναζήτησης χωρίς να χρειάζεται περαιτέρω ενέργεια).

Για να μπορέσουμε να αντλήσουμε για κάθε πατέντα ξεχωριστά, περισσότερες πληροφορίες, θα πρέπει να γίνει μια νέα ξεχωριστή κλήση σε κάποιο Endpoint το οποίο να μας επιστρέφει αυτές τις πληροφορίες. Δυστυχώς όμως για την περίπτωση της Google, δεν υπάρχει ένα API που να επιστρέφει αυτές τις πληροφορίες σε μορφή JSON όπως κάναμε προηγουμένως. Μπορούμε να παρατηρήσουμε, κάνοντας μια απλή περιήγηση και έχοντας ανοιχτό το “capturing” των “calls” που γίνονται στον περιηγητή μας, να δούμε πως η Google Patents προφανώς αφού κάνουμε κλικ σε κάποιο αποτέλεσμα, μας μεταφέρει στην σελίδα που περιέχει όλες αυτές τις επιπρόσθετες πληροφορίες που ψάχνουμε. Παρατηρώντας τις κλήσεις που έγιναν και τα αποτελέσματα που επιστρέφει η κάθε μία, φαίνεται πως τα αποτελέσματα επιστρέφονται κατευθείαν με το που μπούμε στη σελίδα της πατέντας. Αυτό σημαίνει πως θα χρειαστεί να λάβουμε την πληροφορία με HTML Scrapping. Το URL που θα

χρησιμοποιήσουμε είναι:

```
https://patents.google.com/patent/{publication_number}
```

όπου το "{publication_number}" θα βάλουμε το publication number της πατέντας που ο χρήστης έχει επιλέξει στο PerFedPat. Η πληροφορία για το URL που περιέχει τις περισσότερες πληροφορίες την δημιουργούμε εκ τον προτέρω στην φάση της εμφάνισης των αποτελεσμάτων αφού το publication_number το λαμβάνουμε τότε.

Με το που γίνει η κλήση αυτής της σελίδας, λαμβάνουμε HTML περιεχόμενο το οποίο θα περιέχει τις λεπτομερείς πληροφορίες της πατέντας. Αυτό θα γίνει με την χρήση της βιβλιοθήκης Jsoup. Χάρης αυτή τη βιβλιοθήκη μπορούμε να καλέσουμε το URL και να λάβουμε απευθείας το περιεχόμενό της. Έπειτα αναζητούμε την πληροφορία που βρίσκεται μέσα στα κατάλληλα "tags" είτε χρησιμοποιώντας εύρεση μέσω μεθόδων όπως getElementsByAttributeValue ή μπορούμε να λάβουμε με χρήση xPath. Στην υλοποίηση του Wrapper μας, γίνεται η χρήση και των δύο μεθόδων. Για παράδειγμα για να λάβουμε την περιγραφή της πατέντας (abstract) θα την βρούμε ουσιαστικα κάτω απο html tag με την τιμή του attribute class να είναι "abstract". Για να γίνει η εύρεση των CPCs, χρησιμοποιούμε το xPath "ul[itemprop='cpcs']" διότι υπάρχουν περισσότερα από ένα και με αυτόν τον τρόπο λαμβάνουμε ολη τη λίστα τον ul tags που έχουν το attribute itemprop ίσο με "cpcs". Έπειτα ακολουθεί και περαιτέρω αναζήτηση-επεξεργασία μέχρις ότου φτάσουμε στο σημείο που βρίσκεται η πληροφορία μας. Η πολυπλοκότητα αυτή εξαρτάται από το πώς είναι χτισμένη η ιστοσελίδα, πόσα HTML tags χρησιμοποιεί και τον τρόπο που χρησιμοποιούνται. Για παράδειγμα η πληροφορία που θέλουμε μπορεί να μην είναι πάντα κάτω απο ενα ιδανικο tag με class attribute όπως στη περίπτωση των CPCs.

Στο σχήμα 6.7 ακολουθεί απόσπασμα του κώδικα που θα εκτελεστεί τη στιγμή που ο χρήστης θα θέλει να δει περισσότερες πληροφορίες για μια πατέντα.

```java
@Override
public void retrieveDetails(StoredDLObjectList incompleteObjects) {
    DLObject document = incompleteObjects.get(0).getDLObject();
    Document document1 = null;
    try {
        document1 = Jsoup.connect( url: "https://patents.google.com/patent/" + document.getFieldValue(Field.PubNum).toString() + "/en").get();
    } catch (IOException e) {
        e.printStackTrace();
        return;
    }
    if (document1 != null) {
        Elements abstractElements = document1.getElementsByAttributeValue("class", "abstract");
        if (abstractElements != null && !abstractElements.isEmpty()) {
            String abstractText = abstractElements.first().text();
            document.setFieldValue(Field.ABSTRACT, abstractText);
        }

        Elements descriptionElements = document1.getElementsByAttributeValue("class", "description");
        if (descriptionElements != null && !descriptionElements.isEmpty()) {
            String descriptionText = descriptionElements.first().text();
            document.setFieldValue(Field.Description, descriptionText);
        }

        StringList cpcList = new StringList();
        for (Element ul : document1.select( cssQuery: "ul[itemprop='cpcs']")) {
            cpcList.add(ul.select( cssQuery: "li").last().getElementsByAttributeValue("itemprop", "Code").first().text());
        }
        document.setFieldValue(Field.CPC, cpcList);
    }
}
```

Σχήμα 6.7: Απόσπασμα μεθόδου retrieveDetails για απόκτηση περισσότερων πληροφοριών πατέντας

Κεφάλαιο 7ο: Χρήση Headless Browsers

7.1 Εισαγωγή στη διαδικασία των Headless Browser

Σε αυτό το κεφάλαιο θα περιγράψουμε πως μπορούμε να υλοποιήσουμε έναν Wrapper για οποιαδήποτε ιστοσελίδα διαθέσιμη στο διαδίκτυο. Ουσιαστικά, με χρήση κώδικα, θα αυτοματοποιήσουμε την διαδικασία αναζήτησης που θα έκανε ένας κοινός χρήστης στην ιστοσελίδα που μας ενδιαφέρει. Αυτό γίνεται με τη χρήση Headless Browser, που όπως περιγράφηκε σε προηγούμενο κεφάλαιο, είναι ένα πρόγραμμα περιήγησης χωρίς την διεπιφάνεια για προγραμματιστική χρήση. Για την περίπτωσή μας, θα χρησιμοποιήσουμε την βιβλιοθήκη HtmlUnit που μας δίνει την δυνατότητα να τρέξουμε έναν Headless Browser χωρίς κάποια προηγούμενη εγκατάσταση. Επίσης η HtmlUnit μας δίνει δυνατότητες εκτέλεσης JavaScript και AJAX εντολών προσομοιώνοντας περιηγητές όπως Google Chrome, Mozilla Firefox, Internet Explorer (Edge).

Δυστυχώς όμως, η επικοινωνία πια με πολλές ιστοσελίδες, χρειάζονται προηγμένα SSL πρωτόκολλα τα οποία η Java 7 μπορεί να μην είναι ικανή να τα υποστηρίξει. Στη περίπτωσή μας, αυτό μπορούμε να το αποφύγουμε, μεταποιωντας κάποια βασικά αρχεία της Java έκδοσης που έχουμε, συγκεκριμένα με τη χρήση του Bouncy Castle JCE με το οποίο αντικαθιστούμε το java.security αρχείο με αυτό που μπορεί να παρέχει υποστήριξη στις σημερινές ιστοσελίδες. Χρειάζεται υπενθύμιση πως η έκδοση της Java 7 δεν υποστηρίζεται πια επίσημα από την Oracle και επιπλέον αναβαθμίσεις (οι οποίες περιέχουν και αναβαθμίσεις ασφαλείας, συμπεριλαμβανομένου και αυτών που χρειαζόμαστε) με πληρωμή.

Όσο αναφορά τον κώδικα, αυτό που χρειάζεται αρχικά να γίνει, είναι η εισαγωγή της βιβλιοθήκης της HtmlUnit στο pom.xml αρχείο μας. Αυτό θα γίνει απλα προσθέτοντας το παρακάτω κομμάτι XML

```
<dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-lang3</artifactId>
        <version>3.5</version>
 </dependency>
```

Έπειτα, θα πρέπει να βρούμε το σημείο στον κώδικα που θέλουμε να κάνουμε εκκίνηση τον web client μας. Αυτό μπορεί να είναι κατα την πρώτη αναζήτηση η κατα την εκκίνηση του προγράμματος. Διαλέγουμε την έκδοση Browser που θέλουμε (οπου στις περιπτώσεις μας, δεν έχει κάποια μεγάλη σημασία) και έπειτα ορίζουμε κάποιες ρυθμίσεις. Αυτές οι ρυθμίσεις έχουν να κάνουν με υποστηριζόμενες λειτουργίες, ρυθμίσεις ασφαλείας κλπ. Για τη χρήση μας, θέλουμε τις απλούστερες ρυθμίσεις αλλα ταυτόχρονα και λειτουργικές. Για παράδειγμα η ενεργοποίηση CSS δεν είναι απαραίτητη καθόλου, αφού το πρόγραμμα θα τρέχει σε πλαίσια Headless, δηλαδή χωρίς διεπαφή. Οπότε είναι λογικό να μην δίνουμε πόρους για CSS απο τη στιγμή που κανείς δε θα δει την ιστοσελίδα που θα "επισκεφθούμε". Όμως χρειάζεται να έχουμε ενεργή την JavaScript, redirects και να αγνοήσουμε οποιοδήποτε script error έτσι ώστε οποιοδήποτε πρόβλημα (που μπορεί να προκληθεί και από την ίδια την ιστοσελίδα και να μην επηρεάζει το σκοπό μας) να μην αποτελέσει μπλόκο στη διαδικασία. Παρακάτω παρουσιάζεται απόσπασμα κώδικα για την εκκίνηση ενος Headless Browser.

```
WebClient webClient;
webClient = new WebClient(BrowserVersion.CHROME);
webClient.getOptions().setThrowExceptionOnScriptError(false);
webClient.getOptions().setCssEnabled(false);
webClient.getOptions().setRedirectEnabled(true);
webClient.getOptions().setJavaScriptEnabled(true);
webClient.getOptions().setUseInsecureSSL(true);
webClient.getOptions().setSSLClientProtocols(new String[]{ "TLSv1.2", "TLSv1.1", "TLSv1"});
```

Σχήμα 7.1: Απόσπασμα κώδικα για προετοιμασία του Headless Browser

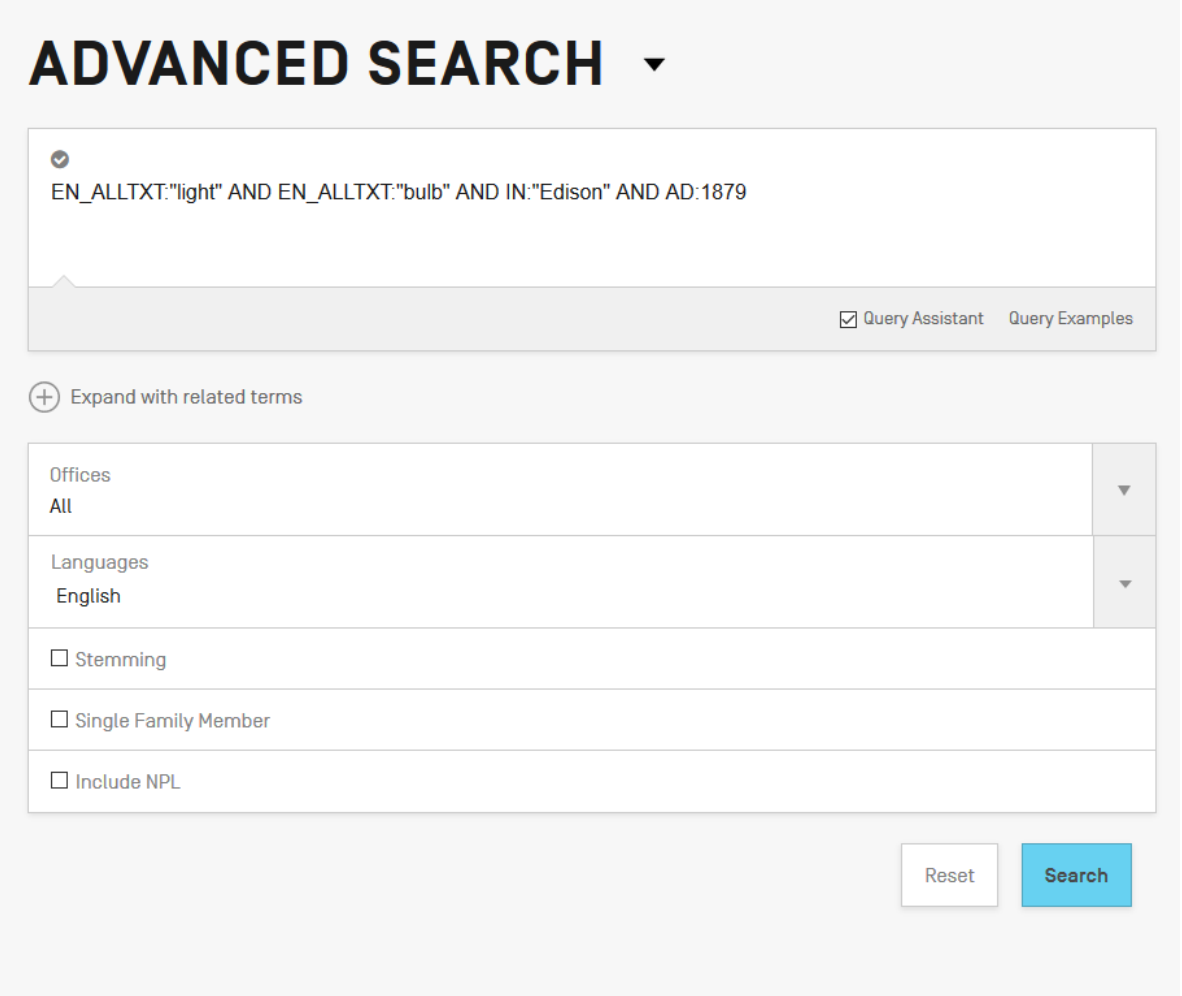
Ένα μειονέκτημα της χρήσης Headless Browser και HtmlUnit είναι πως αν η ιστοσελίδα περιέχει έλεγχο bot με κάποιο είδους Captcha, η ιστοσελίδα αυτή μπορεί πολύ εύκολα να μας μπλοκάρει τη πρόσβαση της. Σαφώς είναι δύσκολο να προσπεραστεί αυτό το κομμάτι με όποιον τρόπο καθώς δεν είναι εύκολος τρόπος να γίνει η ταυτοποίηση με Captcha αυτοματοποιημένα, καθώς είναι φτιαγμένα για αυτόν ακριβώς τον λόγο, δηλαδή να αποτρέψουν χρήστες/προγραμματιστές να χρησιμοποιήσουν ιστοσελίδες προγραμματιστικά με αυτοματοποίηση. Επίσης οι Headless Browser, συγκεκριμένα η HtmlUnit, λογω του γεγονότος ότι κάνει ουσιαστικά προσομοίωση προγράμματος περιήγησης σε Java, καθιστά πολλές λειτουργίες αρκετά αργές. Κύριο πρόβλημα στην αργοπορία είναι η χρήση JavaScript Engine σε Java, δηλαδή μια "μηχανή" (engine) η οποία μπορεί να εκτελέσει κώδικα JavaScript. Η μηχανή αυτή πρέπει να είναι συμβατή με Java, το οποίο καθιστά την μεγαλύτερη καθυστέρηση λόγω της φύσης της Java. Έχουν γίνει βέβαια προσπάθειες για βελτίωση σε αυτόν τομέα, που θα μπορούσε να χρησιμοποιηθεί διαφορετική JavaScript Engine απο αυτή που προσφέρει η OpenJDK, όπως η GraalVM.

Τέλος, τελευταίο μειονέκτημα είναι το γεγονός ότι από τη στιγμή που αναγκαζόμαστε να προσομοιώνουμε τον περιηγητή, εν τέλει εκτελούμε πολλές λειτουργίες που πιθανότατα αποσπούν πόρους χωρίς να μας αφορούν στον σκοπό μας. Για παράδειγμα μια ιστοσελίδα θα μπορούσε όταν την επισκέπτεται ένας χρήστης, να παίζει κάποιο βίντεο το οποίο θα αποσπα πόρους για την "εμφάνισή" του στο παρασκήνιο, αλλα και δικτυακούς πόρους που θα επιβραδύνουν την επικοινωνία.

7.2 Δημιουργία ezDL queries και χρήση τους με HtmlUnit

Όπως περιγράφηκε και στα προηγούμενα δύο κεφάλαια για τους αντίστοιχους Wrappers, θα χρειαστεί να μετατρέψουμε και εδώ το ezDL query που μας δόθηκε σε query κατανοητό για την ιστοσελίδα που θέλουμε να κάνουμε την αναζήτηση. Κάθε ιστοσελίδα μπορεί να διαφέρει. Θα μπορούσαμε να έχουμε μια ιστοσελίδα η οποία περιέχει μια λίστα από πεδία κειμένου όπως κάνει υποστηρίζει ήδη το PerFedPat, δηλαδή να έχει ξεχωριστά πεδία για "Date", "Inventor" κλπ. Η Google Patents είναι μια ιστοσελίδα που λειτουργεί με αυτόν τον τρόπο (υποστηρίζει βέβαια και τον τρόπο που αναλύσαμε στο προηγούμενο κεφάλαιο). Άλλες ιστοσελίδες όμως μπορεί να παρέχουν μόνο μια απλή μπάρα αναζήτησης και οδηγίες για εισαγωγή συγκεκριμένων πεδίων όπως Inventor με κάποια κωδικοποίηση.

Θα χρησιμοποιήσουμε ένα παράδειγμα ιστοσελίδας αναζήτησης πατεντών πέρα από αυτές που δείξαμε. Η ιστοσελίδα αυτή θα έχει ένα συγκεκριμένο τρόπο αναζήτησης που έχει μόνο μια μπάρα αναζήτησης, οδηγίες για τον τρόπο με τον οποίο μπορούμε να γράψουμε μια αναζήτηση, και το κουμπί αναζήτησης "search". Παράδειγμα τέτοιας ιστοσελίδας παρουσιάζεται στο σχήμα 7.2



Σχήμα 7.2: Παράδειγμα ιστοσελίδας που θα γίνει χρήση HtmlUnit

Για αρχή στον κωδικα μας, αυτό που θέλουμε να κάνουμε είναι αφού ξεκινήσουμε τον Headless Browser με τον τρόπο που περιγράψαμε, να συνδεθούμε στην ιστοσελίδα που περιέχει το πεδίο αναζήτησης και το κουμπί "Search". Αυτό θα γίνει πολύ εύκολα με χρήση της μεθόδου "`getPage(String url)`" που καλείτε από το ήδη υπάρχων `webClient` αντικείμενο που ξεκινήσαμε προηγουμένως. Αυτό θα μας επιστρέψει την ιστοσελίδα σε μορφή αντικειμένου `HtmlPage` . Το αντικείμενο αυτό περιέχει όλες τις πληροφορίες από την ιστοσελίδα που θέλουμε και μας παρέχει την δυνατότητα να αλληλεπιδράσουμε μαζί της. Αυτο που χρειαζόμαστε τώρα είναι ότι θα κάναμε στην πραγματικότητα ως χρήστες, δηλαδη να βρούμε την μπάρα αναζήτησης, να κάνουμε κλικ πάνω και να αρχίσουμε να γράφουμε το τι θέλουμε να αναζητήσουμε. Η ίδια διαδικασία μπορεί να γίνει και προγραμματιστικά χάρης των δυνατοτήτων που μας δίνονται. Βρίσκουμε δηλαδή τη μπάρα μέσω xPath ή βρίσκοντας το element μεσω του id του και έπειτα συνεχίζουμε την ενέργεια του. Έπειτα αυτό που μπορούμε να κάνουμε είναι να πούμε τον Headless Browser ουσιαστικά να γράψει για εμάς πάνω στη μπάρα. Όμως η ιστοσελίδα-παράδειγμα, μας δίνει ένα εμπόδιο σε αυτό το σημείο. Όσο γρήγορα και να γράψει κανείς στη μπάρα, η μπάρα μπορεί να δέχεται μόνο μερικά inputs ανα δευτερόλεπτο. Αυτό έχει ως αποτέλεσμα να "κοπούν" κάποια πλήκτρα και να γίνει λάθος αναζήτηση. Μπορούμε ως λύση, να καθυστερήσουμε την εισαγωγή των πλήκτρων στη μπάρα αλλα αυτό σημαίνει πιο αργή αναζήτηση. Η λύση σε αυτή τη περίπτωση είναι να μεταποιήσουμε το Html

της σελίδας και απλά να θέσουμε πολύ απλά το κείμενο της μπάρας με το query που θέλουμε να αναζητήσουμε. Έπειτα βρίσκουμε το κουμπί αναζήτησης με τον ίδιο τρόπο και κάνουμε "Κλικ"

```
HtmlPage page = webClient.getPage( url: "https:/[redacted]search/en/advancedSearch.jsf");
HtmlTextArea searchInput = (HtmlTextArea) page.getElementById( elementId: "advancedSearchForm:advancedSearchInput:input");
searchInput.setText(queryString);
DomElement searchButton = page.getElementById( elementId: "advancedSearchForm:searchButton");
HtmlPage clickPage = searchButton.click();
webClient.waitForBackgroundJavaScript( timeoutMillis: 60000);
```

Σχήμα 7.3: Απόσπασμα κώδικα για άνοιγμα της σελίδας μεσω του Headless Browser

7.3 Διαχείριση και εμφάνιση αποτελεσμάτων μέσω HtmlUnit

Αφού έχουμε εισάγει το query στη μπάρα αναζήτησης, κάνοντας κλίκ, λαμβάνουμε πίσω την ιστοσελίδα ξανα στη μορφή αφού κάνουμε την αναζήτηση. Σε αυτό το σημείο γίνονται διαφορες λειτουργίες (επικοινωνία με server κ.α.) μεσω JavaScript για την ανάκτηση των αποτελεσμάτων. Είμαστε αναγκασμένοι να θέσουμε τον Browser μας σε αναμονή, μέχρι όλα τα Scripts να ολοκληρωθούν. Αυτό ίσως είναι και το αδύναμο σημείο των Headless Browser, δηλαδή οτι υπάρχει γενικά μεγάλη αναμονή για διάφορες λειτουργίες που ίσως δεν μας προσφέρουν κάτι, αλλα είναι απαραίτητες ετσι ώστε να έχουμε σωστά στημένη την ιστοσελίδα και να βρούμε τις πληροφορίες που χρειαζόμαστε. Αφού ολοκληρωθεί η αναμονή αυτή, η ιστοσελίδα τυγχάνει να μας δίνει αργότερα την δυνατότητα να επιλέξουμε ως επιπλέον ρύθμιση τον αριθμό ανα σελίδα των αποτελεσμάτων. Με παρόμοιο τρόπο θα βρούμε το dropdown box που το περιέχει αυτό και θα το θέσουμε στη μέγιστη τιμή (π.χ. 100). Έπειτα η ιστοσελίδα, μετά την επιλογή αυτή θα ανανεωθεί και θα πρέπει να λάβουμε ξανά το περιεχόμενό της.

```
HtmlPage resultPage = (HtmlPage) clickPage.getEnclosingWindow().getTopWindow().getEnclosedPage();
HtmlSelect perPageDropdown = (HtmlSelect) resultPage.getElementById( elementId: "resultListCommandsForm:perPage:input");
perPageDropdown.setSelectedAttribute(perPageDropdown.getOptionByValue("200"), isSelected: true);
webClient.waitForBackgroundJavaScript( timeoutMillis: 60000);
resultPage = (HtmlPage) clickPage.getEnclosingWindow().getTopWindow().getEnclosedPage(); //refresh after selecting 200 results
webClient.waitForBackgroundJavaScript( timeoutMillis: 60000);
```

Σχήμα 7.4: Απόσπασμα κώδικα για αλλαγή του αριθμού αποτελεσμάτων

Αφού γίνει αυτό, λαμβάνουμε πάλι το περιεχόμενο HtmlPage της σελίδας που θα περιέχει πλήρες περιεχόμενο μετά την αναμονή μας. Γνωρίζοντας που βρίσκονται τα αποτελέσματα μέσω των Html Tags τους ή μεσω άλλων έμμεσων τρόπων, θα λάβουμε τη λίστα με τα αποτελέσματα.

```
DomNodeList resultList = resultPage.getElementById( elementId: "resultListForm:resultTable_data").getChildNodes();
```

Σχήμα 7.5: Απόσπασμα κώδικα για εύρεση των αποτελεσμάτων στη σελίδα

Αυτή η λίστα είναι μια λίστα HTML. Θα χρειαστεί να λάβουμε για κάθε "node" της λίστας τα αποτελέσματα με τον κατάλληλο τρόπο. Δυστυχώς στο παράδειγμα μας, το σημείο που παρουσιάζονται κάποια συγκεκριμένα στοιχεία πατεντών δεν είναι πάντα σταθερός και αποτελούν

μέρος elements χωρίς μοναδικά στοιχεία για να τα ξεχωρίσουμε. Για παράδειγμα, ενα αποτέλεσμα μπορεί να μην περιέχει περιγραφή, ή να μη περιέχει τον Inventor. Επίσης ο Inventor θα μπορούσε να βρίσκεται μέσα σε ένα απλό element `<div>` χωρίς κάποια χαρακτηριστικα, αποτελώντας δύσκολη την εύρεσή του, αν υπάρχει. Για αυτό το λόγο, γίνονται κάποιες υποθέσεις παίρνοντας υπόψη κοντινές ενδείξεις για αρχή την ύπαρξή του, όπως αν υπάρχει άλλο div με τη λέξη “Inventor” πριν απο το div με το όνομα του Inventor. Στο σχήμα 7.6 δίνεται το απόσπασμα κώδικα που χρησιμοποιήθηκε για αυτήν την διαδικασία που μόλις περιγράψαμε

Τέλος, αφου λάβουμε τα δεδομένα τα δημιουργούμε για κάθε αποτέλεσμα ένα Document αντικείμενο ετσι ώστε να επιστραφούν αυτά συνολικά ως μια λίστα αποτελεσμάτων κατανοητή από το ezDL και το front-end.

```
for (int i = 0; i < resultList.getLength(); i++) {
    HtmlTableRow tableRow = (HtmlTableRow) resultList.get(i);
    TextDocument document = new TextDocument();

    List<HtmlElement> informationList = tableRow.getElementsByAttribute( elementName: "span",  attributeName: "class",  attributeValue: "trans-section");

    if (informationList.size() >= 1) {
        String title = informationList.get(0).getTextContent();
        document.setTitle(title);
    }

    if (informationList.size() >= 2) {
        String abstractText = informationList.get(1).getTextContent();
        document.setAbstract(abstractText);
    }

    String url = tableRow.getElementsByTagName("a").get(0).getAttribute( attributeName: "href");
    String completeUrl = "https://[redacted]/search/en/" + url;
    document.addDetailURL(new URL(completeUrl));

    List<HtmlElement> informationElementList = tableRow
            .getElementsByAttribute( elementName: "span",  attributeName: "class",  attributeValue: "ps-field--label notranslate");
    for (HtmlElement htmlElement : informationElementList) {

        try {
            HtmlElement informationElement = htmlElement;

            String informationKey = informationElement.getTextContent();
            String informationKeyText = StringUtils.trim(informationKey);


            if ("Int.Class".equals(informationKeyText)) {
                StringList intClassList = new StringList();
                if (informationElement.getNextElementSibling() != null && informationElement.getNextElementSibling().getFirstElementChild() != null) {
                    intClassList.add(informationElement.getNextElementSibling().getFirstElementChild().asText());
                    document.setIPC(intClassList);
                }
            }
            if ("Applicant".equals(informationKeyText)) {
                if (informationElement.getNextElementSibling() != null) {
                    PersonList applicantList = new PersonList();
                    applicantList.add(new Person(informationElement.getNextElementSibling().getTextContent()));
                    document.setApplList(applicantList);
                }
            }
            if ("Inventor".equals(informationKeyText)) {
                if (informationElement.getNextElementSibling() != null) {
                    PersonList applicantList = new PersonList();
                    applicantList.add(new Person(informationElement.getNextElementSibling().getTextContent()));
                    document.setApplList(applicantList);
                }
            }
```

```
                }
                if ("Appl.No".equals(informationKeyText)) {
                    if (informationElement.getNextElementSibling() != null) {
                        document.setAppNum(informationElement.getNextElementSibling().getTextContent());
                    }
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        WrapperDLObject wrapperResult = new WrapperDLObject(document, new SourceInfo(getSourceID(),  detailsInfo: ""));
        wrapperResults.add(wrapperResult);


    }

    return wrapperResults;

} catch (Exception e) {
    e.printStackTrace();
    return new WrapperDLObjectList();
}
```

Σχήμα 7.6: Απόσπασμα κώδικα για εύρεση πληροφοριών κάθε πατέντας

Κεφάλαιο 8ο: Συμπεράσματα ή/και προτάσεις βελτίωσης

Απο την ανάλυση του προγράμματος PerFedPat και την υλοποίηση επί πρόσθετων λειτουργιών, μπορούμε να κατανοήσουμε τα συμπεράσματα που μπορούμε να λάβουμε από όλη την διαδικασία.

Πρώτο συμπέρασμα είναι πως ένα project σαν το PerFedPat, απαιτεί συνεχή συντήρηση και ανάπτυξη κώδικα αν θέλουμε να είναι λειτουργικό πλήρες. Απο τη στιγμή που μιλάμε για υπηρεσίες διαδικτύου, αυτές μπορούν και αλλάζουν τον τρόπο λειτουργίας τους. Για παράδειγμα ο Wrapper της Google Patents πριν 10 χρόνια θα ήταν τελείως διαφορετικός από αυτόν που δημιουργήθηκε εν έτη 2021. Η επικοινωνία που γινόταν τότε του διακομιστή-χρήστη, ήταν τελείως διαφορετική. Στη πορεία αναπτύχθηκαν νεα APIs και νέοι τρόποι αποστολής και λήψης πληροφοριών. Επίσης εάν γίνει η χρήση Headless Browser σε μια ιστοσελίδα, αυτή μπορεί πολύ εύκολα και χωρίς προειδοποίηση να αλλάξει τελείως τον τρόπο με τον οποίο ένας χρήστης αλληλεπιδρά μαζί της, με αποτέλεσμα ο κώδικας που γράφτηκε να μην λειτουργεί. Για αυτούς τους λόγους, ένα project που στηρίζεται από υπηρεσίες διαδικτύου χρειάζεται συνεχής έλεγχο και συντήρηση για την ομαλή λειτουργία του.

Δεύτερο συμπέρασμα είναι πως παρά τις δυσκολίες που αντιμετωπίζουμε σε ένα τέτοιο project, αντιλαμβανόμαστε πως είναι εν τέλη εφικτές και πολύ χρήσιμες οι λειτουργίες που δίνονται. Αντιλαμβανόμαστε πως τα εργαλεία που μας δίνονται κάνουν την δημιουργία παρόμοιου λογισμικού και την συνεχής συντήρησή τους, ευκολότερη.

Οι βελτιώσεις που μπορεί να γίνουν στο μέλλον είναι η “μοντερνοποίηση” του κώδικα και του front-end περιβάλλοντος με σημερινά δεδομένα. Το γεγονός πως το project είναι γραμμένο σε Java 7 η οποία δεν υποστηρίζεται επίσημα πια, είναι κάτι που θα μπορούσε να θεωρηθεί ως εμπόδιο για χρήση σύγχρονων μεθόδων ανάπτυξης λογισμικού. Επίσης η δημιουργία ενός ανοιχτού και εύκολα προσβάσιμου API του PerFedPat (ή ezDL) για την επικοινωνία χρήστη-διακομιστή, θα βοηθούσε στην ευκολότερη ανάπτυξη διεπαφή χρήστη σε διάφορες πλατφόρμες, ακόμα και πλατφόρμες που δεν είναι γραμμένες σε Java, όπως παράδειγμα μια ιστοσελίδα, ή μια mobile εφαρμογή.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία

[1] World Intellectual Property Organization (WIPO), *WIPO Intellectual Property Handbook, 2004*

[2] World Intellectual Property Organization (WIPO), WIPO Guide to Using Patent Information, 2015

[3] World Intellectual Property Organization (WIPO), Guide to the International Patent Classification, 2019

[4] Morgan Kaufmann, The Morgan Kaufmann Series in Data Management Systems, Data Mining (Third Edition), 2012, Pages 1-38

[5] University of Duisburg-Essen, Working Group Information Engineering, "ezDL - Operator Guide", http://gimlet.is.inf.uni-due.de/ezdl/dev/operator-guide.pdf

[6] M. Jordan & T. Beckers, "ezDL - Developer Guide", 2020,

http://gimlet.is.inf.uni-due.de/ezdl/dev/developer-guide.pdf

[7] Y. Daniel Liang, "Εισαγωγή Στον Προγραμματισμό JAVA 10η έκδοση", Εκδόσεις Τζιόλα, 2015

[8] European Patent Office, "Open Patent Services RESTful Web Services Reference Guide", 2020 https://www.epo.org/searching-for-patents/data/web-services/ops.html

Internet Site

[9] European Patent Office, "The European Patent Convention", Rule 42 "Content of the description" 2016. Available: https://www.epo.org/law-practice/legal-texts/html/epc/2016/e/r42.html

[10] Accenture, "Search & Content Analytics Spotlight" Available:

https://web.archive.org/web/20200928092819/https://www.searchtechnologies.com/blog/federated-search

[11] ezDL, A search System and Framework, Available: http://www.ezdl.de/overview/

[12] Apache Maven Project, "What is Maven", https://maven.apache.org/what-is-maven.html

[13] Apache Lucene, https://lucene.apache.org/

[14] MDN Web Docs, Mozilla, XPath, https://developer.mozilla.org/en-US/docs/Web/XPath

[15] What is an API?, C. Siegel & A. Dorairajan API Friends,

https://apifriends.com/api-management/what-is-an-api/

[16] A. R. H. Girdwood, What is a Headless Browser?

https://blog.arhg.net/2009/10/what-is-headless-browser.html

[17] Google, About Google Patents Overview,

https://support.google.com/faqs/answer/6390996?hl=en&ref_topic=6390989

[18] Gargoyle Software Inc., HtmlUnit, https://htmlunit.sourceforge.io/

[19] Oracle, GraalVM, Accelerating Java performance,

https://www.graalvm.org/java/advantages/#accelerating-java-performance

Journal Articles

[20] Yugandhara Patil and Sonal Patil, “Review of Web Crawlers with Specification and Working”, “International Journal of Advanced Research in Computer and Communication Engineering”, Vol. 5, Issue 1, January 2016 , pp 220-222