ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Επιχειρηματολογία και Συλλογισμός στο Υπολογιστικό Νέφος

Γληγόρης Γεώργιος

Εξεταστική επιτροπή:

Αν. Καθηγητής Μιχαήλ Γ. Λαγουδάκης

Καθηγητής Ευριπίδης Πετράκης

Δρ. Νικόλαος Σπανουδάκης (μέλος Ε.ΔΙ.Π. Σχολής Μ.Π.Δ.)

Χανιά, Αύγουστος 2020

TECHNICAL UNIVERSITY OF CRETE

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Argumentation and Reasoning on the Computational Cloud

Gligoris Georgios

Thesis Committee:

Associate Professor Michail G. Lagoudakis

Professor Euripidis Petrakis

Researcher Nikolaos Spanoudakis (TUC Laboratory Teaching Stuff)

Chania, August 2020

Περίληψη

Η σύγχρονη τάση στη σημερινή ψηφιακή εποχή είναι να παρέχονται υπολογιστικές υπηρεσίες στους τελικούς χρήστες μέσω διαδικτύου, ώστε ο διασυνδεδεμένος χρήστης να έχει πλήρη και εύκολη πρόσβαση από παντού, τόσο στις πλέον ενημερωμένες εκδόσεις του λογισμικού, αλλά σε πολλές περιπτώσεις και στα αρχεία του. Η συγκεκριμένη διπλωματική εργασία επιχειρεί την μεταφορά του Gorgias, ενός εργαλείου για λήψη αποφάσεων με βάση την επιχειρηματολογία και τον συλλογισμό, στο υπολογιστικό νέφος.

Με βασική ιδέα τη σύνδεση προγραμμάτων με την μηχανή Prolog μέσω διαδικτύου, με το πέρας της εργασίας αναπτύχθηκε ένα ολοκληρωμένο περιβάλλον στο υπολογιστικό νέφος. Μέσω της υλοποιημένης πλατφόρμας, ο χρήστης έχει την δυνατότητα να δημιουργήσει φάκελους-έργα, να αποθηκεύσει, να επεξεργαστεί και να κατεβάσει αρχεία τοπικά. Επιπρόσθετα, χρησιμοποιώντας το πάνελ της εφαρμογής, αλληλεπιδρά και διαχειρίζεται τη μηχανή Prolog και το εργαλείο Gorgias και θέτει ερωτήματα, στα οποία με βάση τα αρχεία και τους κανόνες που έχει φορτώσει και έπειτα από συλλογιστική πορεία, παίρνει τις σχετικές απαντήσεις.

Η εφαρμογή έχει αναπτυχθεί με τεχνολογίες που βρίσκονται στην αιχμή της τεχνολογίας σήμερα σε ότι αφορά τον τομέα ανάπτυξης εφαρμογών στο διαδίκτυο. Η διαδικτυακή υπηρεσία είναι υλοποιημένη με το Spring boot framework της Java, ενώ για την διεπαφή χρήστη επιλέχθηκε η Angular 7, εργαλεία τα οποία μαζί με την βιβλιοθήκη RxJS παρέχουν μία ολοκληρωμένη εμπειρία χρήσης. Για τις ανάγκες αποθήκευσης δεδομένων χρησιμοποιήθηκε σχεσιακή βάση δεδομένων και τέλος για την εγκατάστασή της επιλέχθηκε η τεχνολογία των containers με την βοήθεια της πλατφόρμας Docker. Πυρήνας της εφαρμογής, όπως είναι φυσικό, είναι η μηχανή Prolog και κατά επέκταση το εργαλείο Gorgias, όπου οι βιβλιοθήκες του χρησιμοποιούνται κατά κόρον σε αυτή.

Ως αποτέλεσμα της εργασίας, η εφαρμογή είναι διαθέσιμη στο διαδίκτυο, προσφέροντας έναν δίαυλο επικοινωνίας με τη μηχανή Prolog και παρέχοντας στον τελικό χρήστη πληθώρα δυνατοτήτων.

Abstract

The current trend in today's digital era is to provide computing services to the end users via the world-wide web, so that the connected user has full and easy access from everywhere, both to the most up-to-date versions of the software, but in many cases also to the user's own files. This thesis attempts the migration of Gorgias, a decision-making tool based on argumentation and reasoning, to the computational cloud.

With the main idea of connecting programs to the Prolog engine via the web, at the completion of this thesis an integrated environment was developed on the computational cloud. Through the implemented platform, the user has the ability to create project folders, save, edit and download files locally. Additionally, using the application panel, the user interacts and manages the Prolog engine and the Gorgias tool and asks questions, which, based on the files and rules uploaded by the user and after a reasoning process, lead to answers.

The application has been developed with state-of-the-art technologies in the domain of web application development. The online service is implemented using Java's Spring boot framework, while for the user's interface the Angular 7 framework has been chosen, both of which together with the RxJS library provide a complete user experience. For the data storage needs a relational database system was used and finally the container's technology was selected for the application's installation with the assistance of the Docker platform. Core of the application is, of course, the Prolog engine and consequently the Gorgias tool, whose libraries are used extensively in it.

As a result of this thesis, the application is available online, providing a communication channel with the Prolog engine and offering to the user a variety of features.

Ευχαριστίες

Με την ολοκλήρωση της παρούσας διπλωματικής εργασίας, θα ήθελα να ευχαριστήσω τον Αναπλ. Καθηγητή κ. Μιχαήλ Λαγουδάκη για την ανάθεση και υποστήριξη που μου παρείχε. Ιδιαίτερες ευχαριστίες στο μέλος Ε.ΔΙ.Π. της σχολής ΜΠΔ κ. Νικόλαο Σπανουδάκη για την πρόταση της εργασίας, την άμεση και πλήρη υποστήριξη και την καταλυτική του βοήθεια σε πολλά κομμάτια αυτής. Επίσης θα ήθελα να ευχαριστήσω τον Καθηγητή κ. Ευριπίδη Πετράκη για τον χρόνο του και τις συμβουλές του.

Ιδιαίτερη μνεία στους φίλους μου Κώστα, Αργύρη, Αλέξανδρο και στις φίλες μου Αγγελική, Πηνελόπη και Ευρύκλεια, που περάσαμε όλα αυτά τα χρόνια παρέα. Τέλος, το μεγαλύτερο ευχαριστώ το οφείλω στην οικογένειά μου, στον πατέρα μου Στάθη, στην μητέρα μου Κατερίνα και στην αδερφή μου Δέσποινα, για την υπομονή και την συνεχή στήριξή τους.

Γνωρίζοντας ότι αποτελεί την εκπλήρωση μιας μεγάλης επιθυμίας και για εκείνον, η διπλωματική μου εργασία είναι αφιερωμένη στον παππού μου, Αθανάσιο Κωβαίο.

Πίνακας περιεχομένων

Πίνακας εικόνων

Κεφάλαιο 1 - Εισαγωγή

1.1 Περίληψη

Η επιχειρηματολογία είναι η διεπιστημονική μελέτη για το πως μπορούν να εξαχθούν συμπεράσματα μέσω συλλογιστικής πορείας. Αποτελεί μια ταχέως αναπτυσσόμενη επιστήμη με μεγάλο φάσμα εφαρμογών που περικλείει από προγράμματα επιχειρήσεων και ασφάλεια δικτύων [1] μέχρι και ιατρικές εφαρμογές.

Η συγκεκριμένη διπλωματική εργασία εμπλέκει διάφορους τομείς και τεχνολογίες. Τα συνδυάζει όλα μαζί αρμονικά και σκοπός της είναι η ανάπτυξη μιας διαδικτυακής εφαρμογής πάνω στο εργαλείο Gorgias [2], ώστε να γίνει ένα πιο προσιτό, πιο εύχρηστο και πιο φορητό εργαλείο. Το Gorgias είναι ένα εργαλείο που κάνει χρήση της επιχειρηματολογίας και του συλλογισμού και χρησιμοποιείται για την λήψη αποφάσεων. Έχει ως βάση του την μηχανή Prolog [3] και μπορεί να ειπωθεί πως αποτελεί ένα framework[1] αυτής. Ο κώδικάς του είναι γραμμένος σε γλώσσα προγραμματισμού Java[2] και η επικοινωνία με την μηχανή Prolog γίνεται μέσω μιας διεπαφής που ονομάζεται JPL [4]. Απαραίτητη προϋπόθεση για την επιτυχή σύνδεση της Java με την Prolog, είναι η Prolog μηχανή να είναι εγκατεστημένη τοπικά. Αυτό το γεγονός αποτελεί και την βάση της παρούσας εργασίας, η οποία ξεκίνησε με στόχο την δημιουργία μιας διαδικτυακής υπηρεσίας, όπου θα καθίσταται δυνατή η χρήση της JPL διεπαφής μέσω διαδικτύου από οποιοδήποτε υπολογιστή. Αυτό συνεπάγεται ότι πέρα από την χρήση της συγκεκριμένης διαδικτυακής υπηρεσίας από το εργαλείο Gorgias, θα είναι δυνατή η χρήση και από οποιονδήποτε χρήστη είτε σε κάποιο πρόγραμμα, είτε μέσω διαδικτύου. Με άλλα λόγια ο καθένας θα μπορεί να χρησιμοποιήσει το συγκεκριμένο εργαλείο και την Prolog, χωρίς να χρειάζεται να τα εγκαταστήσει τοπικά στον υπολογιστή του.

Ίσως η κυριότερη λειτουργία της Prolog είναι η φόρτωση αρχείου στη βάση δεδομένων της και μετέπειτα οι ερωτήσεις προς αυτή. Με αυτό το σκεπτικό δημιουργήθηκε επιπρόσθετα ένα ολοκληρωμένο σύστημα αποθήκευσης και διαχείρισης αρχείων στο υπολογιστικό νέφος, όπου ο χρήστης έχει την δυνατότητα να ανεβάζει αρχεία, να τα αποθηκεύει, να τα φορτώνει στη βάση της Prolog και γενικά να τα διαχειρίζεται. Ο συνδυασμός της έκθεσης της JPL [4] διεπαφής ως διαδικτυακής υπηρεσίας με το σύστημα διαχείρισης αρχείων που δημιουργήθηκε, παρέχουν στον χρήστη ένα ολοκληρωμένο περιβάλλον στο υπολογιστικό νέφος. Με βάση όλα όσα ειπώθηκαν παραπάνω, γίνεται αντιληπτό πως ο σκοπός της εργασίας είναι να μεταφέρει το εργαλείο Gorgias στο υπολογιστικό νέφος.

Στις επόμενες ενότητες θα αναδειχθεί καλύτερα ο στόχος της συγκεκριμένης διπλωματικής εργασίας αλλά και ποια είναι η υφιστάμενη κατάσταση. Στο δεύτερο κεφάλαιο θα εξηγηθούν βασικές έννοιες για την κατανόηση των μετέπειτα κεφαλαίων και γενικά των εργαλείων που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής. Στο τρίτο κεφάλαιο θα παρουσιαστεί η διαδικτυακή υπηρεσία, η δομή της και οι τρόποι χρήσης της, ενώ στο τέταρτο παρουσιάζεται η διεπαφή χρήστη. Με το πέρας της δημιουργίας της εφαρμογής, τέθηκε το ζήτημα της εγκατάστασής της στον εξυπηρετητή. Στο πέμπτο κεφάλαιο αναλύεται η διαδικασία της εγκατάστασης της εφαρμογής και τα εργαλεία που χρησιμοποιήθηκαν για αυτό τον σκοπό. Τέλος, στο έκτο κεφάλαιο παρουσιάζονται τα αποτελέσματα, ενώ στο έβδομο μια σύνοψη και οι προοπτικές του μέλλοντος.

[1] Framework: Ένα σύνολο βιβλιοθηκών και εργαλείων που βοηθάει στην ανάπτυξη λογισμικού
[2] JAVA : Αντικειμενοστρεφής γλώσσα προγραμματισμού

1.2 Υφιστάμενη κατάσταση

Η Prolog είναι μια γλώσσα λογικού προγραμματισμού που χρησιμοποιείται κυρίως στον τομέα της τεχνητής νοημοσύνης. Η υφιστάμενη κατάσταση σε ότι αφορά την Prolog, έχει να κάνει με ολοκληρωμένα περιβάλλοντα γύρω από αυτή την γλώσσα.

1.2.1 Το SWI-Prolog περιβάλλον

Η SWI-Prolog [5] αποτελεί μια υλοποίηση της γλώσσας Prolog και είναι ίσως η δημοφιλέστερη έκδοση της. Περιλαμβάνει ένα πλούσιο σετ από βιβλιοθήκες για λογικό προγραμματισμό, multi-threading[3], unit testing[4], διεπαφές με άλλες γλώσσες προγραμματισμού, κλπ. Τρέχει σε λειτουργικά Unix, Windows, Macintosh και Linux και, από το 1987 που ξεκίνησε μέχρι και σήμερα, αποτελεί την συνηθέστερη επιλογή στην υφιστάμενη κατάσταση σε ότι αφορά την γλώσσα Prolog. Οι τρόποι με τους οποίους ένας χρήστης μπορεί να την χρησιμοποιήσει παρουσιάζονται στη συνέχεια.

Αρχικά εγκαθιστώντας τοπικά το περιβάλλον της SWI-Prolog, το οποίο περιέχει την μηχανή Prolog, μια βάση δεδομένων, αλλά και δικό του κειμενογράφο, ο χρήστης έχει τον έλεγχο της βάσης δεδομένων και των αρχείων, αλλά επιπρόσθετα αποκτά την δυνατότητα να κάνει ερωτήσεις προς την Prolog και παίρνει αντίστοιχα τις απαντήσεις. Αποτελεί ένα πλήρες γραφικό περιβάλλον το οποίο παρέχει ευκολία στον χρήστη και είναι ο πιο κοινός τρόπος για εγκατάσταση τοπικά και εξοικείωση με την SWI-Prolog. Ο χρήστης δημιουργεί αρχεία Prolog χρησιμοποιώντας τον ενσωματωμένο ή έναν κειμενογράφο της αρεσκείας του και έπειτα τα φορτώνει μέσω του περιβάλλοντος της SWI-Prolog.

Ο δεύτερος τρόπος με τον οποίο μπορεί κάποιος να χρησιμοποιήσει την SWI-Prolog, είναι μέσω ενός web-based[5] περιβάλλοντος, του Swish. Το Swish είναι ένα υποσύνολο της SWI-Prolog, το οποίο παρέχει πλούσιες βιβλιοθήκες για προγραμματισμό στο διαδίκτυο, RDF[6] χειρισμούς, πρόσβαση σε βάση δεδομένων, κλπ. Με το περιβάλλον Swish, ο χρήστης δεν χρειάζεται να εγκαταστήσει την Prolog τοπικά, αντιθέτως χρειάζεται απλά και μόνο έναν περιηγητή, ώστε να συνδεθεί στο περιβάλλον μέσω του HTTP πρωτοκόλλου. Μπορεί να ειπωθεί λοιπόν πως το Swish αποτελεί μια online έκδοση της SWI-Prolog, μια εφαρμογή όπου ο χρήστης έχει την δυνατότητα να εξοικειωθεί με την γλώσσα Prolog, να εκτελέσει παραδείγματα και να εξασκηθεί πάνω στην γλώσσα. Επιπρόσθετα, μπορεί να σώσει τον κώδικα του και να τον μοιραστεί μέσω διαδικτύου.

Τέλος, υπάρχει η δυνατότητα χρησιμοποίησης της SWI-Prolog από προγράμματα γραμμένα σε άλλες γλώσσες προγραμματισμού μέσω των διεπαφών που προσφέρει. Για την γλώσσα προγραμματισμού Java υπάρχει η αμφίδρομη διεπαφή JPL, γύρω από την οποία αναπτύχθηκε η διαδικτυακή υπηρεσία για τις ανάγκες τις εργασίας. Εφόσον η διεπαφή JPL είναι μια αμφίδρομη διεπαφή, απαιτείται η εγκατάσταση τοπικά και της Java αλλά και της Prolog. Έχοντας και την Java και την Prolog εγκατεστημένες στο μηχάνημα, μέσω της JPL διεπαφής υπάρχει η δυνατότητα κλήσης της Prolog

[3] Multi-threading: Ύπαρξη πολλών νημάτων μέσα στο πλαίσιο μιας και μόνο διεργασίας
[4] Unit testing: Έλεγχος λογισμικού μέσω δοκιμών σε μονάδες του κώδικα
[5] Web-based: Εφαρμογή ή πρόγραμμα που είναι προσβάσιμο μέσω διαδικτύου
[6] RDF: γλώσσα περιγραφής πληροφοριών που αφορούν πόρους στο διαδίκτυο

από ένα πρόγραμμα Java, αλλά και αντίστοιχα η δυνατότητα κλήσης της Java από ένα πρόγραμμα Prolog, εφόσον όπως αναφέρθηκε και προηγουμένως η συγκεκριμένη διεπαφή είναι αμφίδρομη. Τα μέρη και ο τρόπος με τον οποίο λειτουργεί η διεπαφή, θα αναλυθούν εκτενέστερα στη συνέχεια.

1.2.2 Το εργαλείο Tau Prolog

Το Tau prolog [6] είναι ένας Prolog διερμηνέας αναπτυγμένος εξ ολοκλήρου στην γλώσσα προγραμματισμού Javascript[7]. Οι περισσότεροι διαδικτυακοί διερμηνείς είναι απομακρυσμένοι εξυπηρετητές με μια εγκατεστημένη έκδοση του διερμηνέα πάνω τους. Λαμβάνουν κώδικα, τον εκτελούν και επιστρέφουν τα αποτελέσματα. Ο Tau prolog είναι πλήρως αναπτυγμένος σε Javascript και ο κώδικας αναλύεται και εκτελείται στην πλευρά του χρήστη. Δεν υπάρχει κάποιος εξυπηρετητής που τρέχει από πίσω και όλη η διαδικασία διαδραματίζεται στο μέρος του πελάτη. Διατίθεται ελεύθερα, που σημαίνει ότι ο καθένας μπορεί να τον χρησιμοποιήσει, να τον τροποποιήσει και να μοιραστεί οποιαδήποτε έκδοσή του.

1.3 Κενά που προκύπτουν από την υφιστάμενη κατάσταση

Παραπάνω περιγράφτηκε η υφιστάμενη κατάσταση γύρω από την γλώσσα λογικού προγραμματισμού Prolog, καθώς επίσης αναφέρθηκαν και οι τρόποι με τους οποίους μπορεί κάποιος να κάνει χρήση της γλώσσας. Διαβάζοντας προσεκτικότερα, είναι φυσικό να δημιουργηθούν κάποια ερωτήματα, όπως επίσης και με την πάροδο του χρόνου, αλλά και την εξέλιξη της τεχνολογίας, να προκύψουν νέες απαιτήσεις. Κάθε ένας τρόπος χρήσης της Prolog έχει τα θετικά του, αλλά και τα αρνητικά του. Όπως αναφέρθηκε παραπάνω, αν ο χρήστης επιθυμεί να χρησιμοποιήσει την Prolog σε ένα πρόγραμμα Java έχει τη δυνατότητα να το κάνει, όμως απαραίτητη προϋπόθεση είναι να έχει εγκατεστημένη και την Prolog στο μηχάνημά του. Η JPL διεπαφή είναι μια διεπαφή που χρησιμοποιείται κατά κόρον, όμως γίνεται αντιληπτό ότι δεσμεύει τον χρήστη. Χρησιμοποιώντας τόσο τον πρώτο, όσο και τον τρίτο τρόπο που περιγράφτηκε στην προηγούμενη ενότητα, παρέχεται η δυνατότητα στον χρήστη να έχει τα δικά του αρχεία και να τα τρέχει όπου επιθυμεί (είτε μέσω Java, είτε μέσω SWI-Prolog), πάντα όμως τοπικά.

Την λύση σε αυτό το πρόβλημα ήρθε να την δώσει το Swish, ένα web-based περιβάλλον και υποσύνολο της SWI-Prolog όπως αναφέρθηκε προηγουμένως. Κάνοντας χρήση της Prolog μέσω αυτού του τρόπου, παρέχεται η δυνατότητα στον χρήστη να γράψει και να τρέξει προγράμματα Prolog ανεξαρτήτως μηχανήματος, εφόσον η διαδικασία πραγματοποιείται μέσω διαδικτύου. Θα μπορούσε λοιπόν να ειπωθεί ότι αποτελεί ένα είδος υπολογιστικού νέφους, αφού η Prolog μηχανή είναι εγκατεστημένη σε έναν εξυπηρετητή και παρέχεται πρόσβαση σε αυτή μέσω μίας διαδικτυακής πλατφόρμας.

Και πάλι όμως υπάρχουν κάποια κενά. Αρχικά, ο χρήστης δεν έχει την δυνατότητα να δημιουργήσει λογαριασμό και εφόσον δεν έχει λογαριασμό γίνεται αντιληπτό ότι δεν υπάρχει και η δυνατότητα αποθήκευσης φακέλων και αρχείων στην εφαρμογή. Συνεπώς χάνεται το πλεονέκτημα να έχει τα δικά του αρχεία και κάθε φορά θα πρέπει να δημιουργεί καινούρια. Έπειτα, δεν υπάρχουν βιβλιοθήκες που να δίνουν την δυνατότητα σύνδεσης ενός προγράμματος με την Prolog, χωρίς να είναι απαραίτητη η εγκατάσταση της τοπικά.

[7] Javascript: Διερμηνευμένη γλώσσα προγραμματισμού για διαδικτυακές εφαρμογές

Το συμπέρασμα λοιπόν που προκύπτει είναι ότι κάθε ένας τρόπος έχει τα πλεονεκτήματα και τα μειονεκτήματα του και στη συγκεκριμένη εργασία έγινε μια προσπάθεια εξαγωγής των πλεονεκτημάτων του εκάστοτε τρόπου χρήσης της Prolog και συνδυασμός τους με σκοπό την καλύτερη εμπειρία και διευκόλυνση του χρήστη.

1.4 Τρόπος επίλυσης ζητημάτων μέσω της εφαρμογής

Όπως αναφέρθηκε και παραπάνω, παρατηρώντας τις ελλείψεις που υπάρχουν στην υφιστάμενη κατάσταση, σκοπός της εργασίας είναι να κρατήσει τα πλεονεκτήματα από τον κάθε τρόπο χρήσης της Prolog και επιπροσθέτως να προσπαθήσει, μέσα από τον συνδυασμό τους, να εξαλείψει τα μειονεκτήματα.

Έχοντας ως κύρια ιδέα την χρήση της Prolog μηχανής μέσω Java, χωρίς όμως η εγκατάσταση της τοπικά να κρίνεται απαραίτητη, στην συγκεκριμένη εργασία έγινε η προσπάθεια δημιουργίας μιας διαδικτυακής υπηρεσίας που έχει ως βάση για την επικοινωνία μεταξύ Prolog και Java, την JPL διεπαφή. Σαν προέκταση των παραπάνω, έγινε προσπάθεια να δοθεί στον χρήστη η δυνατότητα να ‘τρέξει’ προγράμματα Gorgias μέσω της διαδικτυακής υπηρεσίας και τελικά η προσπάθεια αυτή στέφθηκε με επιτυχία. Μεταφέροντας την μηχανή Prolog, αλλά και την διεπαφή σε έναν εξυπηρετητή, αυτομάτως αποκτήθηκε το πλεονέκτημα της μη αναγκαίας εγκατάστασης της Prolog τοπικά.

Έπειτα από αυτό το πρώτο βήμα προέκυψαν ερωτήματα σε ότι αφορά την ασφάλεια, την ιδιωτικότητα, αλλά και το προσωπικό αρχείο του εκάστοτε χρήστη. Παρατηρώντας την εμφάνιση των μειονεκτημάτων χρήσης της Prolog μέσω web-based εφαρμογής, προστέθηκαν στην εφαρμογή που αναπτύχθηκε, λειτουργίες τέτοιες ώστε να παρέχεται στον χρήστη ιδιωτικότητα, ασφάλεια, αλλά και η δυνατότητα να έχει τα προσωπικό του αρχεία στο υπολογιστικό νέφος. Θα μπορούσε να ειπωθεί λοιπόν, πως ξεκινώντας με στόχο και ως βάση της εργασίας την εξαγωγή της JPL διεπαφής σαν διαδικτυακή υπηρεσία για την αυτόνομη επικοινωνία μεταξύ Java και Prolog, δημιουργήθηκε ένα περιβάλλον στο υπολογιστικό νέφος, το οποίο παρέχει στον χρήστη περαιτέρω δυνατότητες και λειτουργίες. Πέρα από τις λειτουργίες που αναφέρθηκαν προηγουμένως, όπως η ασφάλεια και η ιδιωτικότητα, είναι σημαντικό να επισημανθεί η δυνατότητα που δίνεται στον χρήστη να τρέξει προγράμματα Gorgias μέσω διαδικτύου, μια δυνατότητα που μέχρι σήμερα δεν υπήρχε σε καμία εφαρμογή. Γίνεται αντιληπτό λοιπόν, πως το εργαλείο Gorgias καθίσταται πιο γνωστό, αλλά και προσιτό, μέσω της εφαρμογής αυτής.

Σε αυτό το κεφάλαιο, δόθηκαν οι απαντήσεις πάνω στο τι υλοποιεί η συγκεκριμένη εργασία και ποια κενά προσπαθεί να καλύψει. Χρησιμοποιήθηκαν επίσης έννοιες, όπως το υπολογιστικό νέφος (Cloud computing), η διεπαφή (Interface), η ασφάλεια, αλλά και γλώσσες προγραμματισμού, όπως η Java και η Prolog και γενικά πλαίσια (frameworks), όπως το Gorgias, τα οποία θα διασαφηνιστούν και θα μελετηθούν αναλυτικότερα στο επόμενο κεφάλαιο.

Κεφάλαιο 2 - Έννοιες, τεχνολογίες και εργαλεία που χρησιμοποιήθηκαν

2.1 Υπολογιστικό νέφος

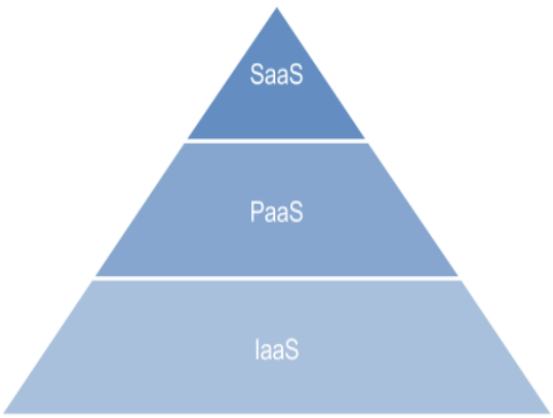
Η παρούσα εργασία μελετάει τον συλλογισμό στο υπολογιστικό νέφος και παρουσιάζει μια εφαρμογή που το υλοποιεί. Αναφέρθηκε στο προηγούμενο κεφάλαιο πως η ιδέα πάνω στην οποία δημιουργήθηκε η εφαρμογή ήταν η έκθεση της JPL διεπαφής [4] σαν διαδικτυακή υπηρεσία με σκοπό την χρησιμοποίηση της γλώσσας Prolog [5] σε προγράμματα Java χωρίς να είναι εγκατεστημένη η Prolog τοπικά. Αποτελεί αυτό από μόνο του μια μορφή υπολογιστικού νέφους; Η απάντηση δίνεται μελετώντας τι είναι το υπολογιστικό νέφος και ποιες είναι οι διάφορες μορφές του.

Υπολογιστικό νέφος [7] ονομάζεται η διανομή των υπολογιστικών υπηρεσιών, όπως παραδείγματος χάρη των εξυπηρετητών, των αποθηκευτικών χώρων, των βάσεων δεδομένων, του λογισμικού και άλλα, μέσω του διαδικτύου. Το υπολογιστικό νέφος χωρίζεται σε τρεις τύπους υπηρεσιών που μπορεί να προσφέρει.

Αρχικά, μπορεί να προσφέρει υποδομή ως υπηρεσία ή όπως συνηθίζεται να λέγεται Infrastructure as a Service (IaaS). Το IaaS είναι μία υπηρεσία στο υπολογιστικό νέφος που προσφέρει υπολογιστική υποδομή, όπως εξυπηρετητές και αποθηκευτικό χώρο. Θα μπορούσε να ειπωθεί με απλά λόγια πως είναι ένα εικονικό κέντρο δεδομένων. Οι υπηρεσίες αυτές μπορούν να χρησιμοποιηθούν για διάφορους σκοπούς, από την φιλοξενία ιστοσελίδων έως και την επεξεργασία μεγάλων δεδομένων. Οι πελάτες μπορούν να εγκαταστήσουν και να χρησιμοποιήσουν οποιοδήποτε λειτουργικό σύστημα και εργαλείο θέλουν στις υποδομές που παίρνουν. Οι μεγαλύτεροι πάροχοι τέτοιων υποδομών είναι ο Microsoft Azure, ο Amazon Web Services και ο Google Compute Engine.

Ένας δεύτερος τύπος υπηρεσιών που προσφέρεται στο υπολογιστικό νέφος είναι η πλατφόρμα ως υπηρεσία ή αλλιώς Platform as a Service (PaaS). Το PaaS αναφέρεται σε πλατφόρμες που παρέχουν runtime περιβάλλοντα για προγραμματισμό, αλλά και διαχείριση εφαρμογών. Χάρη στις υπηρεσίες πλατφόρμας, οι προγραμματιστές μπορούν να αναπτύξουν εφαρμογές χωρίς την ανάγκη των σχετικών υποδομών (εξυπηρετητές, λειτουργικά συστήματα, εργαλεία προγραμματισμού και άλλα). Οι PaaS προμηθευτές παρέχουν ένα ολοκληρωμένο σύστημα υποδομής για την ανάπτυξη εφαρμογών, ενώ οι προγραμματιστές είναι υπεύθυνοι για τον κώδικα και μόνο. Παραδείγματα PaaS προμηθευτών αποτελούν το Windows Azure, OpenShift, AWS Elastic Beanstalk και Heroku.

Ο τρίτος τύπος υπηρεσιών που μπορεί να προσφέρει το υπολογιστικό νέφος είναι το λογισμικό ως υπηρεσία ή αλλιώς Software as a Service (SaaS). Το SaaS επιτρέπει στους χρήστες να χρησιμοποιούν εφαρμογές προσβάσιμες μέσω του υπολογιστικού νέφους. Στην πραγματικότητα, υπηρεσίες email όπως το Gmail και το Hotmail είναι παραδείγματα SaaS υπηρεσιών. Ότι έχει να κάνει με το λογισμικό και υλικό μέρος παρέχεται από τον προμηθευτή, οπότε ο χρήστης δεν χρειάζεται να κάνει καμία εγκατάσταση αλλά αρκεί μόνο να πραγματοποιήσει είσοδο για να χρησιμοποιήσει την εφαρμογή.



Εικόνα 1: Πυραμίδα υπηρεσιών του υπολογιστικού νέφους

Στην Εικόνα 1 απεικονίζονται οι τρεις τύποι υπηρεσιών του υπολογιστικού νέφους, με δομή πυραμίδας. Πράγματι οι IaaS υπηρεσίες αποτελούν την βάση των υπηρεσιών υπολογιστικού νέφους. Με την χρήση PaaS υπηρεσιών αυτόματα ενσωματώνονται και οι υποδομές υλικού και αντίστοιχα με την χρήση SaaS υπηρεσιών ενσωματώνονται οι υποδομές υλικού αλλά και οι υπηρεσίες πλατφόρμας. Θα μπορούσε να ειπωθεί ότι όσο ανεβαίνει κάποιος σε αυτή την πυραμίδα τόσο λιγότερο έλεγχο της εφαρμογής έχει.

Με βάση τα όσα ειπώθηκαν παραπάνω γίνεται αντιληπτό πως η εφαρμογή που αναπτύχθηκε για τις ανάγκες της εργασίας ανήκει στην τρίτη κατηγορία και αποτελεί ένα Software as a Service. Στο Backend τμήμα έχει δημιουργηθεί η διαδικτυακή υπηρεσία, η οποία είναι προσβάσιμη με δύο τρόπους: (α) είτε μέσω μιας διεπαφής χρήστη, γεγονός που ολοκληρώνει την εφαρμογή και τη καθιστά σαν μία υπηρεσία λογισμικού (SaaS), (β) είτε μέσω βιβλιοθηκών που αναπτύχθηκαν και μπορούν να εισαχθούν και να κληθούν από οποιοδήποτε πρόγραμμα. Στην τελευταία αυτή παράγραφο όμως προστέθηκαν νέες έννοιες, οι οποίες είναι αναγκαίο να διασαφηνιστούν, πριν αναλυθεί εκτενέστερα η δομή της εφαρμογής.

2.2 Χρήσιμες έννοιες

2.2.1 Μέρος πελάτη και εξυπηρετητή

Στην πληροφορική οι όροι Frontend και Backend αναφέρονται στον διαχωρισμό των τμημάτων ανάμεσα στο επίπεδο της παρουσίασης και στο επίπεδο της πρόσβασης στα δεδομένα ενός λογισμικού. Σε ένα μοντέλο πελάτη-εξυπηρετητή, ο πελάτης θεωρείται ως το Frontend και ο εξυπηρετητής συνήθως ως το Backend τμήμα της εφαρμογής.

2.2.2 Διεπαφή προγραμματισμού εφαρμογών (API)

Η Διεπαφή Προγραμματισμού Εφαρμογών (Application Programming Interface - API) είναι ένας όρος που μπορεί να χρησιμοποιηθεί για να περιγράψει τα χαρακτηριστικά μιας βιβλιοθήκης ή το πως να επικοινωνήσεις με αυτή. Είναι ουσιαστικά η διεπαφή των προγραμματιστικών διαδικασιών που παρέχουν ένα λειτουργικό σύστημα, μια βιβλιοθήκη ή μια εφαρμογή, προκειμένου να επιτρέπει να γίνονται προς αυτά αιτήσεις από άλλα προγράμματα ή ακόμα και ανταλλαγή δεδομένων. Συνήθως κάθε API (Application Programming Interface) έχει και ένα “API Documentation”, το οποίο αναλύει ποιες συναρτήσεις είναι διαθέσιμες, τον τρόπο με τον οποίο μπορείς να τις καλέσεις, κλπ.

Εξ ορισμού, μια διαδικτυακή υπηρεσία είναι ένα κομμάτι λογισμικού, το οποίο θέτει τον εαυτό του διαθέσιμο μέσω του διαδικτύου και την επικοινωνία με αυτό, συνήθως μέσω XML[8] ή JSON[9]. Ένας πελάτης καλεί την διαδικτυακή υπηρεσία στέλνοντας ένα αίτημα και εκείνη ανταποκρίνεται στέλνοντας μία απάντηση. Η επικοινωνία λαμβάνει χώρα μέσω του διαδικτύου, με το HTTP πρωτόκολλο να είναι το πιο δημοφιλές πρωτόκολλο για την επικοινωνία των δύο συστημάτων.

[8] XML: Γλώσσα σήμανσης που περιέχει σύνολο κανόνων για την ηλεκτρονική κωδικοποίηση κειμένων
[9] JSON: Ανοιχτό μορφότυπο, που μπορεί να διαβαστεί από τον άνθρωπο, για τη μετάδοση αντικειμένων δεδομένων

Αμφότερα, μια διαδικτυακή υπηρεσία και ένα API είναι, όπως περιγράφτηκε και παραπάνω, ένα μέσο επικοινωνίας μεταξύ δύο μηχανών. Η μόνη, άλλα μείζονος σημασίας, διαφορά τους είναι ότι μια διαδικτυακή υπηρεσία θέτει ως "γέφυρα" επικοινωνίας το διαδίκτυο. Ένα API λειτουργεί σαν μια διεπαφή μεταξύ δύο μηχανών που καθορίζει και εκθέτει τις μεθόδους, ώστε να επικοινωνούν οι δύο αυτές μηχανές μεταξύ τους. Όταν αυτή η δράση περιέχει και ανταλλαγή δεδομένων μέσω του διαδικτύου, τότε η διαδικτυακή υπηρεσία έρχεται στο προσκήνιο. Ως εκ τούτου, κάθε διαδικτυακή υπηρεσία είναι μια διεπαφή, αλλά κάθε διεπαφή δεν είναι απαραίτητα και διαδικτυακή υπηρεσία.

2.2.3 Διαφορές μεταξύ SOAP και REST

Simple Object Access Protocol (SOAP) και Representational State Transfer (REST) [8] είναι δύο απαντήσεις στην ίδια ερώτηση: Πώς αποκτώ πρόσβαση σε μια διαδικτυακή υπηρεσία;

Η σύγκριση ωστόσο, SOAP με REST δεν είναι απολύτως σωστή. Και αυτό γιατί το SOAP είναι ένα πρωτόκολλο για πρόσβαση σε μια διαδικτυακή υπηρεσία, ενώ το REST είναι ένα στυλ αρχιτεκτονικής λογισμικού. Είναι αμφότεροι όμως, δύο τρόποι για την απόκτηση πρόσβασης σε μια διαδικτυακή υπηρεσία.

Το SOAP είναι ένα πρωτόκολλο με βάση του την γλώσσα περιγραφής αντικειμένων XML. Η διεπαφή που έχει η διαδικτυακή υπηρεσία περιγράφεται από ένα WSDL[10] έγγραφο. Η ίδια η υπηρεσία περιγράφεται χρησιμοποιώντας την XML και παρέχει όλες τις απαραίτητες πληροφορίες, όπως την κατάλληλη μορφοποίηση των μηνυμάτων, τα πρωτόκολλα μετάδοσης, αλλά και την τοποθεσία για την αλληλεπίδραση με την διαδικτυακή υπηρεσία.

Αντίθετα, το REST είναι ένα αρχιτεκτονικό στυλ που σχεδιάστηκε για διαδικτυακές εφαρμογές. Η κύρια ιδέα είναι πως, αντί να χρησιμοποιούνται περίπλοκοι μηχανισμοί, όπως το SOAP, για την επικοινωνία μεταξύ μηχανών, χρησιμοποιείται το πρωτόκολλο HTTP για να γίνουν οι κλήσεις μεταξύ των μηχανών. Γίνεται αντιληπτό πως και το World Wide Web, το οποίο είναι βασισμένο πάνω στο HTTP πρωτόκολλο, μπορεί να θεωρηθεί ότι χρησιμοποιεί την αρχιτεκτονική REST.

Πέρα από την διαφορά που αναφέρθηκε παραπάνω, υπάρχουν και άλλες διαφορές που πρέπει να ληφθούν υπόψιν, προτού παρθεί η απόφαση με ποιόν τρόπο θα αποκτηθεί πρόσβαση σε μια διαδικτυακή υπηρεσία. Αρχικά το SOAP χρησιμοποιεί διεπαφές ως υπηρεσίες για να εκθέσει επιχειρησιακή λογική, σε αντίθεση με το REST που χρησιμοποιεί το URI. Επίσης το SOAP δέχεται μόνο XML ως διαμόρφωση δεδομένων, ενώ το REST έχει ποικιλία σε ότι αφορά την διαμόρφωση δεδομένων που δέχεται, αφού μπορεί να δεχθεί text, HTML, JSON, XML και άλλα. Έπειτα το SOAP απαιτεί περισσότερους διαθέσιμους πόρους, ενώ το REST είναι πιο ελαφρύ και απαιτεί εμφανώς λιγότερους. Το SOAP βέβαια υπερτερεί σε ασφάλεια έναντι της REST αρχιτεκτονικής. Τέλος, το SOAP υποστηρίζει και το SMTP, αλλά και το HTTP πρωτόκολλο, ενώ η REST αρχιτεκτονική απαιτεί την χρήση του HTTP πρωτοκόλλου και μόνο.

Ένα παράδειγμα που μπορεί να δοθεί για την καλύτερη κατανόηση των όσων αναφέρθηκαν, είναι η παρομοίωση του SOAP πρωτοκόλλου με έναν φάκελο, ενώ του REST απλά με μια καρτ ποστάλ. Σίγουρα η καρτ ποστάλ είναι πιο γρήγορη και φθηνή να σταλθεί, αλλά ο κίνδυνος να χαθεί ή να διαβαστεί από τρίτους είναι σίγουρα μεγαλύτερος από ότι στον φάκελο. Αντίστοιχα, μία καρτ ποστάλ

[10] WSDL: Γλώσσα περιγραφής διεπαφών που περιγράφει την λειτουργικότητα μιας διαδικτυακής υπηρεσίας

μπορεί να σταλθεί μέσα σε έναν φάκελο, αλλά τα βήματα για να την διαβάσει κάποιος είναι περισσότερα.

Σε γενικές γραμμές κάθε ένας από τους δύο τρόπους που αναφέρθηκαν έχει τα μειονεκτήματα και τα πλεονεκτήματά του απέναντι στον άλλον. Η επιλογή του τρόπου πρέπει να είναι ανάλογη των απαιτήσεων που υπάρχουν, αλλά και του εκάστοτε σκοπού. Για τις ανάγκες της συγκεκριμένης εργασίας χρησιμοποιήθηκε η REST αρχιτεκτονική, καθώς υπήρχε η ανάγκη για καλύτερη υποστήριξη από τον περιηγητή και πολλαπλές μορφές δεδομένων, εφόσον η διαδικτυακή υπηρεσία καταναλώνεται από μια διεπαφή χρήστη που επίσης δημιουργήθηκε. Επιπρόσθετα η REST αρχιτεκτονική στην συγκεκριμένη περίπτωση είναι πιο γρήγορη και ευέλικτη, γεγονός που επίσης έπαιξε ρόλο στην επιλογή. Στην επόμενη ενότητα θα αναλυθεί η "γέφυρα" της επικοινωνίας μεταξύ διαδικτυακής υπηρεσίας και πελάτη, δηλαδή το πρωτόκολλο HTTP.

2.2.4 Πρωτόκολλο HTTP

Το HTTP βγαίνει από τα αρχικά του Hypertext Transfer Protocol και σημαίνει πρωτόκολλο μεταφοράς υπερκειμένου. Χρησιμοποιείται για την αποστολή πληροφοριών μεταξύ δύο συστημάτων και χρησιμοποιείται συχνότερα μεταξύ ενός διακομιστή ιστού και ενός οικιακού υπολογιστή. Αν και το HTTP πρωτόκολλο σχεδιάστηκε για χρήση στον ιστό, υποστηρίζει και πιο γενικές λειτουργίες. Οι λειτουργίες αυτές ονομάζονται αιτήσεις και περιλαμβάνουν μεθόδους. Κάθε αίτηση αποτελείται από μία ή περισσότερες γραμμές κειμένου ASCII. Η πρώτη λέξη της πρώτης γραμμής είναι το όνομα της αιτούμενης μεθόδου. Παρακάτω, παρουσιάζονται συνοπτικά οι κύριες μέθοδοι αίτησης του πρωτοκόλλου HTTP:

GET: Η μέθοδος GET ζητά από τον διακομιστή την εκάστοτε σελίδα. Παραδείγματος χάρη GET όνομα_αρχείου HTTP/1.1, όπου το όνομα αρχείου είναι ο πόρος που πρέπει να προσκομιστεί και το 1.1 είναι η έκδοση του πρωτοκόλλου που χρησιμοποιείται.

POST: Η μέθοδος POST όπως και η GET περιέχει μια διεύθυνση URL, αλλά αντί να αιτείται απλά μία σελίδα, μεταφέρει δεδομένα στον διακομιστή, ο οποίος διακομιστής με την σειρά του τα χειρίζεται ανάλογα με το URL.

PUT: Η μέθοδος αυτή είναι αντίστροφη της GET, αντί να διαβάζει την σελίδα, γράφει την σελίδα.

DELETE: Η μέθοδος DELETE καταργεί την σελίδα ή τουλάχιστον δηλώνει ότι ο διακομιστής Ιστού έχει συμφωνήσει να καταργήσει την σελίδα.

Οι παραπάνω μέθοδοι αποτελούν ένα από τα πιο σημαντικά κομμάτια επικοινωνίας ανάμεσα στον εξυπηρετητή και τον πελάτη. Αυτές τις μεθόδους χρησιμοποιεί κατά κόρον και η εφαρμογή που δημιουργήθηκε στο πλαίσιο της εργασίας.

Κάθε αίτηση που στέλνεται όμως λαμβάνει μια απάντηση, η οποία αποτελείται από μια γραμμή κατάστασης και πιθανόν πρόσθετες πληροφορίες. Η γραμμή κατάστασης περιέχει ένα τριψήφιο κωδικό κατάστασης, ο οποίος δηλώνει κατά πόσον εξυπηρετήθηκε η αίτηση και αν δεν εξυπηρετήθηκε, γιατί συνέβη αυτό.

2.3 Γλώσσα προγραμματισμού Prolog

Η Prolog είναι μία δηλωτική γλώσσα λογικού προγραμματισμού, στην οποία τόσο οι δομές δεδομένων της, όσο και ο μηχανισμός λειτουργίας της, βασίζονται στο προτασιακό και κατηγορηματικό λογισμό. Από εκεί άλλωστε προέκυψε και η ονομασία της, εφόσον **Prolog** = **Pro**gramming in **Log**ic.

2.3.1 Δομή ενός προγράμματος Prolog

Σε αυτή την ενότητα θα γίνει μια συνοπτική αναφορά στα τρία κύρια μέρη ενός προγράμματος Prolog [7], τα οποία είναι τα γεγονότα, οι κανόνες, αλλά και οι μεταβλητές.

Γεγονότα

Ένα γεγονός αναπαριστά μία ιδιότητα ενός αντικειμένου ή μία σχέση μεταξύ δύο ή περισσότερων αντικειμένων ή γενικότερα ένα αντικείμενο. Ένα γεγονός μπορεί να περιγραφεί είτε από ένα άτομο, όπως για παράδειγμα room, cat, water κλπ., είτε από έναν ατομικό τύπο, δηλαδή father(nick, alex), είτε από έναν σύνθετο όρο, δηλαδή order(id(10), sound(loud), keyboard).

Κάθε γεγονός τελειώνει με μία τελεία. Η κεφαλή του ατομικού τύπου ονομάζεται κατηγόρημα (predicate). Ένα όρισμα ενός κατηγορήματος μπορεί να είναι είτε ένα άτομο, είτε ένας αριθμός, είτε μία μεταβλητή. Ιδιαίτερα, οι όροι των σύνθετων όρων μπορεί να περιέχουν άλλους σύνθετους όρους. Ο αριθμός τον ορισμάτων που έχει ένα κατηγόρημα ή ένας σύνθετος όρος ονομάζεται τάξη (arity), για παράδειγμα η τάξη του order(id(10), sound(loud), keyboard) είναι τρία.

Μεταβλητές

Κάθε όρισμα ένας κατηγορήματος ή σύνθετου όρου που αρχίζει με κεφαλαίο γράμμα, η Prolog το θεωρεί μεταβλητή. Σε αντίθεση με άλλες γλώσσες προγραμματισμού, οι μεταβλητές στη Prolog δεν έχουν τύπο. Μία μεταβλητή μπορεί να ταυτιστεί με οποιοδήποτε άτομο, κατηγόρημα, σύνθετο όρο ή αριθμό. Δύο μεταβλητές που έχουν το ίδιο όνομα και ανήκουν σε διαφορετικούς κανόνες ή γεγονότα της βάσης δεδομένων δεν σχετίζονται μεταξύ τους. Από τη στιγμή που μία μεταβλητή πάρει μία τιμή μέσα σε έναν κανόνα, τότε αυτή δεν μπορεί να αλλάξει, έτσι σε κάθε χρονική στιγμή μία μεταβλητή μπορεί να θεωρηθεί ότι είτε έχει πάρει τιμή είτε ότι είναι ελεύθερη.

Κανόνες

Ένας κανόνας έχει την μορφή

- K **:-** F1,F2,...,Fn.

Το γεγονός K ονομάζεται κεφαλή του κανόνα, ενώ τα γεγονότα F1, F2, ..., Fn αποτελούν το σώμα του κανόνα. Κάθε κανόνας τελειώνει επίσης με μία τελεία. Για να ισχύσει το γεγονός K θα πρέπει η Prolog να ελέγξει διαδοχικά αν ισχύουν τα F1, F2, ..., Fn.

Το σύμβολο **:-** χωρίζει το σώμα από τη κεφαλή και το γεγονός K ισχύει αν ισχύουν όλα τα γεγονότα του σώματος του κανόνα. Δύο κανόνες μπορούν να έχουν την ίδια κεφαλή.

2.3.2 Χρήση της Prolog

Για να γράψει κανείς ένα πρόγραμμα Prolog πρέπει να χρησιμοποιήσει έναν κειμενογράφο. Μπορεί να χρησιμοποιήσει έναν οποιοδήποτε κειμενογράφο, με την προϋπόθεση ότι το αρχείο θα σώζεται με κατάληξη .pl. Μέσα σε αυτό το αρχείο βρίσκεται η γνώση και η πληροφορία που χρειάζεται η Prolog για να δημιουργήσει την απόκρισή της.

Για να είναι όμως σε θέση να χρησιμοποιήσει τη γνώση αυτή, θα πρέπει τα παραπάνω γεγονότα και οι κανόνες να εισαχθούν στη βάση δεδομένων της (Prolog Knowledge Base). Αυτό πραγματοποιείται φορτώνοντας το αρχείο που περιέχει τους κανόνες και τα γεγονότα στην βάση δεδομένων, με την εντολή consult. Αν το πρόγραμμα που φορτώνεται στην βάση δεδομένων της Prolog δεν έχει λάθη, τότε αποκρίνεται αληθές.

Έχοντας εισαχθεί η γνώση στη βάση δεδομένων της Prolog, υπάρχει η δυνατότητα ερωτήσεων προς αυτή. Οι ερωτήσεις αυτές γίνονται πάντα από τον διερμηνέα της Prolog μετά το σύμβολο ?-. Κάνοντας μια ερώτηση στη Prolog, η Prolog μηχανή ψάχνει στη βάση δεδομένων (ακολουθώντας τη σειρά με την οποία έχουν εισαχθεί τα γεγονότα) και προσπαθεί να βρει απαντήσεις. Επιστρέφει την πρώτη απάντηση που βρίσκει και αφήνει το περιθώριο στον χρήστη αν θέλει να επιστραφούν και άλλες απαντήσεις.

Για να ενημερωθεί η βάση δεδομένων της Prolog και γενικά για την διαχείρισή της, υπάρχουν οι αντίστοιχες εντολές. Πιο συγκεκριμένα οι assert, assertz και asserta προσθέτουν ένα γεγονός στην βάση δεδομένων, ενώ οι εντολές abolish και retract χρησιμοποιούνται για την αφαίρεση του. Με τις εντολές consult, load_files φορτώνεται ένα αρχείο στη βάση, όπως αναφέρθηκε και προηγουμένως, ενώ με την εντολή unload_file αφαιρείται από αυτή. Τέλος, με την εντολή listing μπορεί ο χρήστης να δει όλα τα γεγονότα που έχουν εισαχθεί στη βάση δεδομένων. Οι εντολές αυτές για τον χειρισμό της βάσης αποτελούν παράγοντα στην εφαρμογή που αναπτύχθηκε, αλλά και σημαντικότατο κομμάτι γενικά για την μηχανή Prolog.

Όταν ο χρήστης γράφει ένα πρόγραμμα σε Prolog δεν κάνει τίποτα άλλο από το να εισάγει γεγονότα και κανόνες στη βάση δεδομένων, δηλαδή να εισάγει γνώση σχετική με το χώρο του προβλήματος. Δεδομένης της γνώσης αυτής, μπορεί να ρωτήσει την Prolog αν μία πρόταση είναι αληθής ή ψευδής. Πιο συγκεκριμένα, ο μηχανισμός ελέγχου της Prolog χρησιμοποιεί τη γνώση αυτή για να αποφανθεί αν η ερώτηση που κάνει ο χρήστης είναι αληθής ή ψευδής. Σε αντίθεση με άλλες γλώσσες προγραμματισμού, όπου το πρόγραμμα γράφεται μία φορά και μετά μέσω ενός μεταφραστή (compiler) μετατρέπεται σε γλώσσα μηχανής όπου δεν μπορεί πλέον να αλλαχθεί, στη Prolog το πρόγραμμα (δηλαδή η γνώση που εισάγουμε στη βάση δεδομένων) μπορεί να ενημερωθεί ανά πάσα χρονική στιγμή, είτε από το χρήστη, είτε από το ίδιο το πρόγραμμα. Αυτό είναι πολύ χρήσιμο, όταν δεν είναι εξ αρχής πλήρως γνωστό το πρόβλημα που πρέπει να επιλυθεί.

2.3.4 Το εργαλείο Gorgias

Το Gorgias [2] [9] είναι ένα γενικό πλαίσιο επιχειρηματολογίας που συνδυάζει τις ιδέες της λογικής, της προτίμησης και της απαγωγής με τρόπο που διατηρεί τα οφέλη και των δύο. Μπορεί να αποτελέσει την βάση για τη συλλογιστική, σχετικά με πολιτικές προτιμήσεων, έχοντας ελλιπείς πληροφορίες από δυναμικά περιβάλλοντα. Για παράδειγμα, κατά την αγορά ενός αυτοκινήτου, κάποιος μπορεί να προτιμά ορισμένα χαρακτηριστικά έναντι των άλλων, ή όταν προγραμματίζει τις υποχρεώσεις του, η τήρηση κάποιων προθεσμιών μπορεί να είναι πιο σημαντική έναντι άλλων. Το Gorgias αποτελεί ένα γενικό πλαίσιο με κανόνες και αρχές, που δίνουν απαντήσεις σε ζητήματα όπως τα παραπάνω.

2.3.5 Προαπαιτούμενα για τη χρήση του Gorgias

Το Gorgias είναι ένα framework βασισμένο στην Prolog και για αυτό τον λόγο η εγκατάσταση της SWI-Prolog είναι απαραίτητη για να λειτουργήσει. Εφόσον εγκατασταθεί η SWI-Prolog, για την χρησιμοποίηση του Gorgias και των δυνατοτήτων του, αρκεί να εισαχθούν οι εξής δύο γραμμές στην αρχή κάθε αρχείου:

1. :- compile('../lib/gorgias.pl').
2. :- compile('../ext/lpwf.pl').

Η πρώτη γραμμή φορτώνει το σύστημα του Gorgias, ενώ η δεύτερη φορτώνει μια συλλογή από κανόνες που καθορίζουν την σχέση προεπιλογής μεταξύ των επιχειρημάτων.

2.3.6 Κανόνες του Gorgias

Για να δηλωθούν κανόνες, συγκρούσεις και προτιμήσεις μεταξύ των κανόνων, χρησιμοποιούνται οι όροι της Prolog οι οποίοι σχηματίζονται από σύμβολα κατηγοριοποίησης. Η σύνταξη τέτοιων περιγραφών δίνεται από κανόνες που έχουν την ακόλουθη φόρμα: rule(Label, Head, Body), όπου το κεφάλι (Head) είναι είναι ένα αξίωμα, το σώμα (Body) μια λίστα από αξιώματα και η ετικέτα (Label) είναι ένας σύνθετος όρος, που αποτελείται από ένα όνομα κανόνα και επιλεγμένες μεταβλητές από το κεφάλι και το σώμα. Για παράδειγμα, ο κανόνας rule(r1(x), fly(x), [bird(x)]). δηλώνει ότι κάτι πετάει εφόσον είναι πουλί.

Όταν υπάρχει σύγκρουση μεταξύ κανόνων, η προτεραιότητα μεταξύ σχέσεων ή πιο απλά η προτίμηση μεταξύ των κανόνων δηλώνεται με ένα ειδικό κατηγόρημα το prefer(label1, label2), όπου το label1 και label2 είναι οι ετικέτες των εκάστοτε κανόνων. Συγκεκριμένα το παραπάνω κατηγόρημα δηλώνει ότι ο κανόνας με ετικέτα label1 προτιμάται σε σχέση με τον κανόνα με την ετικέτα label2.

Τέλος, η δήλωση conflict(label1, label2), υποδηλώνει ότι οι κανόνες με ετικέτες label1 και label2 είναι συγκρουόμενοι. Σε πολλές περιπτώσεις αυτό θα είναι αληθές, αν οι ετικέτες label1 και label2 είναι αντίθετα αξιώματα και έρχονται σε σύγκρουση, αλλά επίσης υπάρχει η δυνατότητα να δηλωθούν κανόνες ως συγκρουόμενοι από τον ίδιο τον προγραμματιστή.

2.3.7 Υπολογισμός απόφασης

Γενικά, η κατασκευή μιας λύσης για ένα δεδομένο ερώτημα μπορεί να γίνει σταδιακά, ξεκινώντας από ένα βασικό/αρχικό επιχείρημα και εν συνεχεία προσθέτοντας στο αρχικό επιχείρημα κατάλληλες άμυνες για αυτό. Επικεντρωνόμαστε κυρίως στον υπολογισμό μιας ειδικής κατηγορίας επιχειρημάτων, δηλαδή των αποδεκτών επιχειρημάτων. Διαισθητικά, ένα επιχείρημα είναι αποδεκτό αν αμυνθεί εναντίον κάθε επίθεσης. Πρέπει να σημειωθεί ότι ένα επιχείρημα που επιτίθεται από μόνο του δεν μπορεί να είναι αποδεκτό, επειδή δεν υπάρχει επίθεση ενάντια στο κενό σύνολο.

Ο υπολογισμός ενός αποδεκτού επιχειρήματος είναι μια εμπλοκή δύο φάσεων. Στην πρώτη φάση ένας στόχος μειώνεται σε ένα κλειστό σύνολο που αποδεικνύει το στόχο. Στη συνέχεια, αυτό το αρχικό επιχείρημα επεκτείνεται με κατάλληλες άμυνες για κάθε επίθεση εναντίον του αρχικού συνόλου. Ωστόσο, μετά την ανάπτυξη του αρχικού επιχειρήματος, θα προκύψουν νέες συγκρούσεις και έτσι το σύστημα επαναλαμβάνει ολόκληρη τη διαδικασία έως ότου δεν υπάρξει υπεράσπιση για επίθεση κατά του βασικού επιχειρήματος (δηλαδή αδυναμία εύρεσης ενός αποδεκτού συνόλου) ή μέχρι να μην υπάρξουν άλλες επιθέσεις (δηλαδή την επιτυχή εξαγωγή ενός αποδεκτού επιχειρήματος).

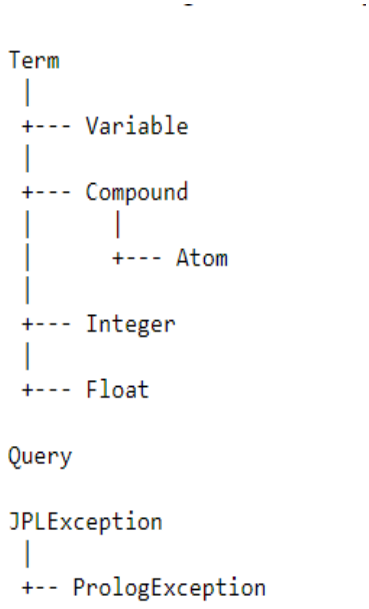
Οι ερωτήσεις που γίνονται προς το σύστημα έχουν την μορφή: prove(Goals, Delta), όπου οι στόχοι (Goals) είναι μία λίστα με θετικά ή αρνητικά αξιώματα, ενώ το Delta είναι ένα αποδεκτό όρισμα για το δεδομένο ερώτημα.

Το Gorgias είναι ένα πολύ χρήσιμο εργαλείο με πολλές δυνατότητες. Έχει την δυνατότητα να λειτουργήσει και να παράξει αποτελέσματα, χωρίς την ανάγκη εισχώρησης δεδομένων πρώτα. Αυτή είναι και η κύρια διαφορά του από άλλες τεχνολογίες τεχνητής νοημοσύνης.

2.3.8 JPL διεπαφή

Το μέρος της εφαρμογής που τρέχει στον εξυπηρετητή και θα παρουσιαστεί στο επόμενο κεφάλαιο, είναι γραμμένο στην γλώσσα προγραμματισμού Java. Για την σύνδεση του Java προγράμματος με την μηχανή Prolog, χρησιμοποιείται η διεπαφή JPL [4].

Η JPL διεπαφή παρέχει ένα σύνολο από κλάσεις που αντιγράφουν τους τύπους δεδομένων και τις συναρτήσεις της SWI-Prolog. Το πακέτο που έχει δημιουργηθεί στον πυρήνα της διεπαφής χρησιμεύει σαν άμεση μετάφραση της Prolog μηχανής. Η διεπαφή JPL αρχικοποιεί την Prolog μηχανή, αν χρειαστεί, μόλις το πρώτο query (η πρώτη ερώτηση προς την μηχανή Prolog) ενεργοποιηθεί. Πρόκειται ουσιαστικά για ένα socket[11] που ανοίγει ανάμεσα στην Java και την SWI-Prolog. Όσο τρέχει το πρόγραμμα, το socket παραμένει ενεργοποιημένο και με την λήξη του προγράμματος, κλείνει και το socket που ενώνει την Java με την SWI-Prolog. Η JPL διεπαφή είναι μια σχετικά μικρή διεπαφή και στην Εικόνα 2 φαίνεται η ιεραρχία των κλάσεών της. Η κλάση Term είναι μια αφηρημένη κλάση και μόνο οι υποκλάσεις της μπορούν να χρησιμοποιηθούν. Οι υποκλάσεις της ωστόσο είναι παρόμοιες



Εικόνα 2: Κλάσεις της JPL διεπαφής

[11] Socket: Δίαυλος αμφίδρομης επικοινωνίας υπολογιστών

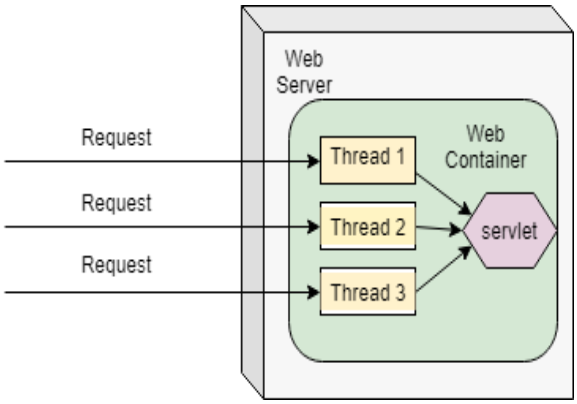
και αντιπροσωπεύουν τους τύπους δεδομένων της SWI-Prolog. Κάθε εικόνα της κλάσης Query, περιέχει και ένα Term, δηλώνοντας έτσι τον στόχο που πρέπει να αποδειχθεί, αλλά και άλλα επιπλέον στοιχεία, αναλόγως την ερώτηση που θα δημιουργηθεί προς την μηχανή Prolog. Τέλος, η κλάση JPLException περιέχει συναρτήσεις και μεθόδους για τον χειρισμό εξαιρέσεων ή επιστροφής σφαλμάτων από την μηχανή Prolog. Στην εφαρμογή και για την σύνδεση με την SWI-Prolog, η JPL έχει καταλυτικό ρόλο. Σε κάθε ερώτηση που προορίζεται για την SWI-Prolog, χρησιμοποιούνται κατάλληλα οι συναρτήσεις και οι μέθοδοι που προσφέρει η διεπαφή και επιτυγχάνεται η επικοινωνία μέσω αυτής της διεπαφής και μόνο.

2.4 Η γλώσσα προγραμματισμού JAVA

Το τμήμα της εφαρμογής που τρέχει στον εξυπηρετητή είναι γραμμένο σε γλώσσα προγραμματισμού Java και πιο συγκεκριμένα δημιουργήθηκε με την βοήθεια ενός γενικού πλαισίου της Java, του Spring. Στις παρακάτω υποενότητες θα γίνει μια περιληπτική αναφορά στο πως λειτουργεί και θα αναδειχθούν τα εργαλεία με τα οποία δημιουργήθηκε η υπηρεσία που τρέχει στο τμήμα του εξυπηρετητή, η οποία θα παρουσιαστεί αργότερα.

2.4.1 JAVA Servlet

Ένα servlet δεν είναι τίποτα άλλο παρά ένα Java πρόγραμμα ή καλύτερα μία Java κλάση, η οποία εκτελείται εσωτερικά της Java εικονικής μηχανής (JVM)[12] σε έναν εξυπηρετητή διαδικτύου. Η πιο συχνή εφαρμογή των Servlets είναι στην επέκταση των δυνατοτήτων ενός εξυπηρετητή, ο οποίος φιλοξενεί εφαρμογές που βασίζονται στο μοντέλο αιτήματος-ανταπόκρισης (request - response). Όπως φαίνεται και στην Εικόνα 3, τα Servlets επικοινωνούν με έναν εξυπηρετητή μέσω ενός container. Δε δημιουργείται ξεχωριστή διεργασία για κάθε αίτημα που καταφθάνει. Αντίθετα για κάθε νέο αίτημα δημιουργείται ένα νέο νήμα και εξυπηρετείται από αυτό. Η Java Virtual Machine (JVM) φορτώνεται μια φορά και παραμένει στη μνήμη.



Εικόνα 3: Web server και Servlet

2.4.2 Spring framework

Το Spring [10] είναι το πιο δημοφιλές framework της Java και χρησιμοποιείται κατά κόρον για την δημιουργία προγραμμάτων και εφαρμογών. Με το Spring framework δημιουργήθηκε και το τμήμα της εφαρμογής που τρέχει στον εξυπηρετητή και παρακάτω θα αναλυθούν έννοιες που θα βοηθήσουν στην καλύτερη κατανόησή του, τα σημαντικότερα κομμάτια του αλλά και το πως δουλεύει.

[12] JVM: Εικονική μηχανή που παρέχει την δυνατότητα στον υπολογιστή να τρέξει Java προγράμματα

2.4.3 Επισημάνσεις του Spring

Στην γλώσσα προγραμματισμού Java, μια επισήμανση (αγγλ. annotation) είναι μια φόρμα από συντακτικά μεταδεδομένα (αγγλ. metadata), η οποία μπορεί να προστεθεί στον Java πηγαίο κώδικα. Κλάσεις, μέθοδοι, παράμετροι και πακέτα μπορούν να περιέχουν επισημάνσεις. Οι επισημάνσεις μπορούν να διαβαστούν από αρχεία πηγαίου κώδικα, αλλά επίσης υπάρχει η δυνατότητα ενσωμάτωσής τους και διαβάσματός τους από αρχεία κλάσεων που δημιουργεί ο μεταγλωτιστής. Το Spring έχει τις δικές του επισημάνσεις, τις οποίες χρησιμοποιεί κατά κόρον, όπως θα δειχθεί και στην συνέχεια. Μερικές από τις πιο δημοφιλείς είναι το @Controller, @RequestMapping, @RequestBody, @ResponseBody κλπ.

2.4.4 Αναστροφή του ελέγχου (Inversion of Control - IoC)

Η αναστροφή του ελέγχου (IoC), είναι μια αρχή στην τεχνολογία λογισμικού με την οποία ο έλεγχος αντικειμένων ή τμημάτων ενός προγράμματος μεταφέρεται σε ένα γενικό πλαίσιο. Χρησιμοποιείται συχνότερα στο πλαίσιο του αντικειμενοστρεφούς προγραμματισμού.

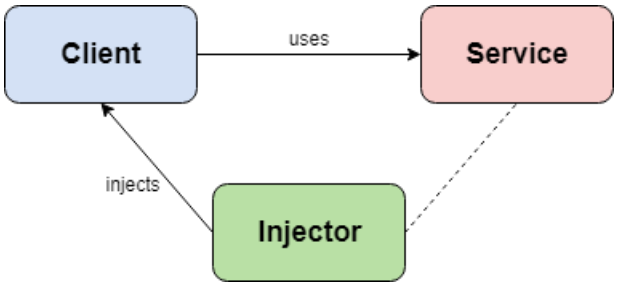
Σε αντίθεση με τον παραδοσιακό προγραμματισμό, στον οποίο ο κώδικας κάνει κλήσεις σε μια βιβλιοθήκη, το IoC απευθύνεται σε ένα γενικό πλαίσιο για να αναλάβει τον έλεγχο της ροής ενός προγράμματος και να κάνει κλήσεις στον κώδικα. Εάν ο προγραμματιστής θέλει να προσθέσει την δική του συμπεριφορά, πρέπει να επεκτείνει τις κλάσεις του πλαισίου στις κλάσεις του προγράμματός του. Η αναστροφή του ελέγχου μπορεί να επιτευχθεί με διάφορους μηχανισμούς, με το σχεδιαστικό μοτίβο Dependency Injection (DI) να αποτελεί έναν από αυτούς.

2.4.5 Το σχεδιαστικό μοτίβο Dependency Injection (DI)

Το Dependency Injection είναι ένα σχεδιαστικό μοτίβο, που υλοποιεί την αναστροφή του ελέγχου (IoC). Πρόκειται για μια διαδικασία που επιτρέπει την δημιουργία εξαρτημένων αντικειμένων εξωτερικά της κλάσης και την παροχή αυτών των αντικειμένων στην κλάση με διαφορετικούς τρόπους. Ιδανικά οι κλάσσεις πρέπει να είναι όσο το δυνατόν ανεξάρτητες η μία από την άλλη κυρίως για λόγους επαναχρησιμότητάς τους. Όπως φαίνεται και στην Εικόνα 4, στο σχεδιαστικό μοτίβο DI εμπλέκονται τρεις τύποι κλάσεων:

1. Κλάση πελάτη (Client class): Αποτελεί την εξαρτημένη κλάση και εξαρτάται από την κλάση υπηρεσιών.
2. Κλάση υπηρεσιών (Service class): Είναι η κλάση που παρέχει την υπηρεσία στην κλάση του πελάτη.
3. Κλάση εισαγωγής (Injector class): Η κλάση εισαγωγής εισάγει το αντικείμενο στην κλάση πελάτη.

Η κλάση πελάτη θέλει να χρησιμοποιήσει ένα αντικείμενο που βρίσκεται εξωτερικά αυτής. Η δυνατότητα αυτή δίνεται μέσω της κλάσης εισαγωγής, η οποία δημιουργεί μια εικόνα της υπηρεσίας που βρίσκεται το αντικείμενο στην κλάση πελάτη και μέσω αυτής της εικόνας παρέχεται η πρόσβαση στο αντικείμενο και τις μεθόδους του.



Εικόνα 4: Dependency Injection

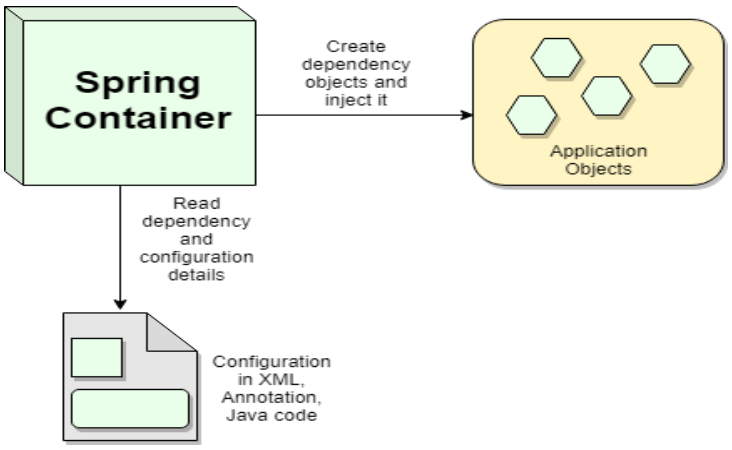
2.4.6 Αντικείμενα του Spring

Τα beans αποτελούν κλειδί για το Spring framework. Ως εκ τούτου, κατανοώντας τι είναι ένα Spring bean και ποια η λειτουργία του, αυτομάτως γίνεται ευκολότερη η κατανόηση του πως λειτουργεί το Sping framework.

Στο Spring, τα αντικείμενα που αποτελούν την ραχοκοκαλιά της εφαρμογής και τα οποία διαχειρίζονται από το Spring IoC container, καλούνται beans. Το Spring έχει δικά του αντικείμενα αλλά επίσης δημιουργούνται και αντικείμενα για την ανάγκη της εκάστοτε εφαρμογής. Ουσιαστικά, το Spring δουλεύει με αυτά τα αντικείμενα και τα διαχειρίζεται μέσω ενός container, του Spring IoC container. Θα μπορούσε να ειπωθεί ότι αποτελεί μια αποθήκη με όλα εκείνα τα εργαλεία που χρειάζεται η εφαρμογή για να λειτουργήσει. Η αποθήκη αυτή έχει ήδη κάποια έτοιμα εργαλεία μέσα, αλλά για κάθε εφαρμογή προστίθενται νέα εργαλεία που εξυπηρετούν τις ανάγκες της.

2.4.7 IoC Container

Το Spring container αποτελεί τον πυρήνα του Spring framework. Το container χρησιμοποιείται για να δημιουργήσει και να διαμορφώσει αντικείμενα. Επίσης, τα Spring IoC Containers χρησιμοποιούνται για να διευθύνουν τον κύκλο ζωής των αντικειμένων αυτών, από την δημιουργία τους μέχρι και την καταστροφή τους. Για τον χειρισμό των αντικειμένων, το Spring Container χρησιμοποιεί σχεδιαστικό μοτίβο Dependency Injection (DI). Το container χρησιμοποιεί μεταδεδομένα τα οποία εξάγει είτε από τον κώδικα, είτε από επισημάνσεις, είτε από ένα XML αρχείο (Εικόνα 5). Το Sping IoC container έχει δύο τύπους. Ο πρώτος είναι το Spring BeanFactory Container το οποίο είναι ένα απλό container που παρέχει βασική υποστήριξη για το μοτίβο σχεδιασμού Depedency Injection. Ο δεύτερος τύπος είναι το ApplicationContext container και αποτελεί τον προχωρημένο τύπο container του Spring. Παρέχει όλες τις λειτουργίες του BeanFactory Container, αφού στην ουσία έχει φτιαχτεί πάνω στο BeanFactory Container, αλλά και κάποιες επιπλέον.



Εικόνα 5: Δημιουργία αντικειμένων μέσω Spring container

2.4.8 Πως λειτουργεί το Spring framework

Όλα ξεκινάνε με τον πελάτη, που είτε πρόκειται για την διεπαφή χρήστη είτε για ένα πρόγραμμα, θα στείλει ένα αίτημα σε ένα συγκεκριμένο URL. Όταν το αίτημα φτάσει στον web container, στην συγκεκριμένη περίπτωση στον Tomcat, εκείνος ψάχνει σε ένα XML αρχείο για να βρει ποιο servlet αντιστοιχεί στο συγκεκριμένο URL που του ζητήθηκε. Μόλις το βρει, απευθύνεται στο servlet, ώστε να επεξεργαστεί το αίτημα. Εφόσον το Spring είναι υλοποιημένο ένα επίπεδο πάνω από το servlet, η ροή κάθε Spring εφαρμογής είναι παρόμοια.

Στο XML αρχείο που ψάχνει ο web container[13], έχει δηλωθεί ο Dispatcher Servlet του Spring, όπου όλα τα αιτήματα αντιστοιχίζονται εκεί. Αυτός με την σειρά του περνάει το αίτημα στον κατάλληλο controller για να το επεξεργαστεί. Πως όμως βρίσκει ποιος controller είναι ο κατάλληλος για το κάθε αίτημα; Με την βοήθεια των επισημάνσεων που αναφέρθηκαν προηγουμένως.

Ο κάθε controller χρησιμοποιεί επιχειρησιακή λογική, καλεί υπηρεσίες και όταν είναι αναγκαίο επικοινωνεί με την βάση δεδομένων για να εξάγει δεδομένα και να δημιουργήσει την απόκριση στο αίτημα που έστειλε ο πελάτης. Εφόσον πρόκειται για μια REST διαδικτυακή υπηρεσία, οι controllers έχουν επισήμανση @ResponseBody, που σημαίνει ότι γράφουν το αποτέλεσμα απευθείας στο σώμα της απάντησης του HTTP και εκείνο επιστρέφεται με την σειρά του σε JSON μορφή. Κατά την επεξεργασία του αιτήματος και για την δημιουργία της απάντησης, χρησιμοποιούνται υπηρεσίες και αντικείμενα μέσω του σχεδιαστικού μοτίβου Dependency Injection, τα οποία δημιουργήθηκαν και διαχειρίζονται από τον IoC Container του Spring.

2.4.9 Hibernate framework

Το Hibernate framework [11] είναι ένα λογισμικό που συνδέει τα αντικείμενα της εφαρμογής με τους πίνακες μιας σχεσιακής βάσης δεδομένων. Η κύρια λειτουργία του είναι η δημιουργία μιας διεπαφής ανάμεσα στη βάση δεδομένων και στο Java πρόγραμμα. Σκοπός αυτής της διεπαφής είναι η χρησιμοποίηση της σχεσιακής βάσης σαν να είναι αντικειμενοστραφής και το επιτυγχάνει αυτό δημιουργώντας αντιστοιχίες μεταξύ εννοιών του αντικειμενοστραφούς προγραμματισμού και των σχέσεων των πινάκων της σχεσιακής βάσης. Το αποτέλεσμα που προκύπτει είναι ο προγραμματιστής

[13] Web container: Το κομμάτι του εξυπηρετητή ιστού που επικοινωνεί με τα Java servlets

να βλέπει μια σχεσιακή βάση δεδομένων σαν αντικειμενοστρεφή και μπορεί να χρησιμοποιεί και να τροποποιεί τα αντικείμενα της εφαρμογής καθαρά σαν αντικείμενα και όχι σαν πίνακες της βάσης δεδομένων. Το Hibernate λοιπόν αναλαμβάνει να κατασκευάσει την κατάλληλη SQL εντολή από την εκάστοτε μέθοδο της Java και να την στείλει στη βάση για υλοποίηση. Στη συνέχεια, πραγματοποιεί την αντίθετη διαδικασία, ώστε να επιστρέψει στο πρόγραμμα τα αποτελέσματα ως αντικείμενα της εφαρμογής. Το Hibernate framework, όπως προκύπτει από τα παραπάνω, τοποθετείται ανάμεσα στην εφαρμογή και στην βάση δεδομένων και η αρχιτεκτονική του αποτελείται από επτά μέρη.

1. **Configuration**: Αποτελεί ένα αρχείο με όλα τα αναγκαία στοιχεία για την σύνδεση της εφαρμογής με τη βάση δεδομένων (URL, πόρτα που ακούει η βάση, κλπ).

2. **Session factory**: Κάθε χρήστης ζητά ένα session αντικείμενο από το session facotry και εκείνο χρησιμοποιεί το configuration αρχείο για την αρχικοποίηση ενός session αντικειμένου.

3. **Session**: Αντιπροσωπεύει την σύνδεση μεταξύ της εφαρμογής και της βάσης δεδομένων κάθε χρονική στιγμή και δημιουργείται από το session factory.

4. **Query**: Επιτρέπει στην εφαρμογή να πραγματοποιεί ερωτήσεις στη βάση για ένα ή παραπάνω αποθηκευμένα αντικείμενα.

5. **First-level cache**: Αντιπροσωπεύει την cache μνήμη την οποία χρησιμοποιεί το session όσο αλληλεπιδρά με την βάση δεδομένων. Όλα τα αιτήματα από το session στη βάση περνάνε από αυτή τη μνήμη.

6. **Transaction**: Ενεργοποιείται για να παρέχει συνοχή και σταθερότητα στα δεδομένα που συναλλάσονται μεταξύ εφαρμογής και βάσης.

7. **Objects**: Τα αντικείμενα της εφαρμογής που πρέπει να αντιστοιχιστούν οι πίνακες της βάσης δεδομένων. Διαμορφώνονται είτε στο configuration αρχείο είτε με την επισήμανση @Entity στις Java κλάσεις.

Πρέπει να επισημανθεί πως για την χρήση του Hibernate διαμορφώθηκαν τα αντίστοιχα beans όπου δηλώθηκαν όλα τα απαραίτητα στοιχεία για την ομαλή λειτουργία του, χρησιμοποιώντας Java διαμόρφωση. Ένα από τα αντικείμενα που δημιουργήθηκαν είναι και το session factory. Οι διεπαφές που σχετίζονται με την σύνδεση εφαρμογής και βάσης δεδομένων, χρησιμοποιούν το session factory για να αλληλεπιδράσουν με το session και να εκτελέσουν λειτουργίες, όπως η αποθήκευση και η ανάκτηση δεδομένων από την βάση.

Στις παραπάνω ενότητες έγινε μια αναφορά στα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του μέρους που τρέχει στον εξυπηρετητή. Παρουσιάστηκε συνοπτικά το Spring framework με το οποίο αναπτύχθηκε το μέρος της εφαρμογής που τρέχει στον εξυπηρετητή, αλλά και το hibernate framework που χρησιμοποιήθηκε για την σύνδεση αυτού με την βάση δεδομένων. Στις επόμενες ενότητες θα παρουσιαστεί το εργαλείο με το οποίο δημιουργήθηκε η διεπαφή χρήστη.

2.5 Angular framework

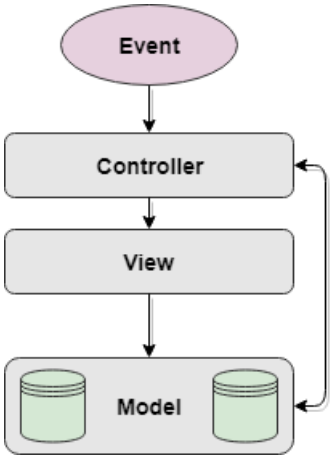
2.5.1 Αρχιτεκτονική της Angular

Όταν, πίσω στο 2013, το Facebook[14] έφερε στο προσκήνιο μια νέα βιβλιοθήκη της Javascript, την React.js, αναπροσδιόρισε τον τρόπο με τον οποίο οι προγραμματιστές έφτιαχναν διεπαφές χρηστών. Η React, που σήμερα αποτελεί το αντίπαλο δέος της Angular, εισήγαγε ένα νέο σχεδιαστικό μοτίβο, μια νέα αρχιτεκτονική, η οποία ονομάζεται αρχιτεκτονική με βάση το component (Component Based Architecture - CBA). Η CBA ενσωματώνει ξεχωριστά κομμάτια μιας μεγαλύτερης διεπαφής χρηστών, σε αυτοσυντηρούμενα και ανεξάρτητα μικροσυστήματα. Η Angular 7, με την οποία δημιουργήθηκε η διεπαφή χρήστη της εφαρμογής, ακολουθεί μια τέτοια αρχιτεκτονική. Η έκδοση 7 είναι ένα σημαντικό στοιχείο εφόσον από την δεύτερη έκδοση της Angular και μετά υπάρχουν ποικίλες αλλαγές σε σχέση με την πρώτη έκδοση. Η πιο σημαντική είναι ότι πλέον δεν ακολουθείται το Model View Controller (MVC) μοντέλο αρχιτεκτονικής, όπως στην AngularJS, αλλά όπως αναφέρθηκε και προηγουμένως ένα μοντέλο αρχιτεκτονικής με βάση το component (CBA). Στις επόμενες ενότητες θα γίνει μια συνοπτική αναφορά στο MVC μοντέλο, θα δειχθεί τι είναι το CBA μοντέλο, αλλά και ποιες είναι οι μεταξύ τους διαφορές.

2.5.2 Αρχιτεκτονική MVC

Το Model View Controller (MVC) [12] αποτελεί ένα αρκετά διαδεδομένο μοτίβο σχεδιασμού λογισμικού και όπως μαρτυρούν και τα αρχικά του, απαρτίζεται από τρία μέρη. Το κομμάτι του μοντέλου (Model), το κομμάτι της παρουσίασης (View) και το κομμάτι του χειριστή (Controller). Είναι δημοφιλές γιατί απομονώνει κάθε κομμάτι από το άλλο και διαχωρίζει τα ζητήματα που προκύπτουν.

Όπως φαίνεται και στην Εικόνα 6, ο controller παραλαμβάνει τα αιτήματα προς την εφαρμογή και έπειτα δουλεύει με το κομμάτι του μοντέλου για την προετοιμασία των δεδομένων που χρειάζονται στην εικόνα. Το κομμάτι της παρουσίασης χρησιμοποιεί τα δεδομένα αυτά για να δημιουργήσει το τελικό αποτέλεσμα που βλέπει ο χρήστης.



Εικόνα 6: MVC αρχιτεκτονική

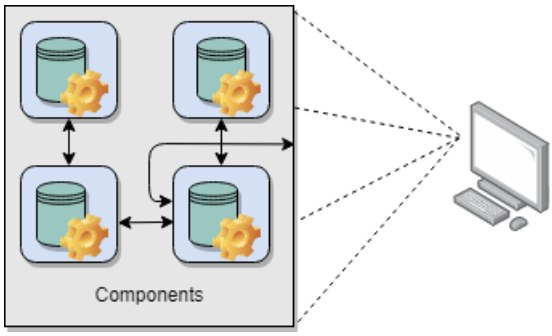
Πιο συγκεκριμένα το κομμάτι του μοντέλου είναι υπεύθυνο για την διαχείριση των δεδομένων της εφαρμογής. Ανταποκρίνεται στα αιτήματα από το κομμάτι της παρουσίασης και στις οδηγίες από τον χειριστή ώστε να αναβαθμιστεί. Το κομμάτι της παρουσίασης (view) αφορά την παρουσίαση των δεδομένων και ενεργοποιείται από τον controller για την παρουσίασή τους. Τέλος, ο controller ανταποκρίνεται στην είσοδο του χρήστη και αλληλεπιδρά με τα υπόλοιπα δύο κομμάτια χρησιμοποιώντας επιχειρησιακή λογική. Πολλές εφαρμογές ακολουθούν το συγκεκριμένο μοντέλο. Το κομμάτι που τρέχει στον εξυπηρετητή και αναλύθηκε στο προηγούμενο κεφάλαιο ακολουθεί τέτοια αρχιτεκτονική με τη μόνη διαφορά ότι ουσιαστικά δεν υπάρχει το κομμάτι της παρουσίασης, αφού αποκρίνεται μόνο με δεδομένα. Η AngularJS είναι ένα MVC framework. Από την Angular 2 μέχρι και την σημερινή εκδοχή της, με την οποία αναπτύχθηκε η διεπαφή χρήστη, η αρχιτεκτονική που ακολουθείται είναι Component-based και θα αναλυθεί εκτενέστερα στις επόμενες ενότητες.

[14] Facebook: Πλατφόρμα κοινωνικής δικτύωσης

2.5.3 Αρχιτεκτονική βασισμένη στο Component (CBA)

Μια από τις τάσεις στην ανάπτυξη εφαρμογών είναι η Component-based αρχιτεκτονική, η οποία αν εφαρμοστεί σωστά προσφέρει την απόλυτη ευελιξία και δυνατότητα κλιμάκωσης. Η αρχιτεκτονική που χρησιμοποιεί η Angular 7 είναι μια τέτοια αρχιτεκτονική (Component Based Architecture - CBA).

Συνεπώς, το ερώτημα που προκύπτει είναι το τι ακριβώς είναι το Component. Το Component είναι ένα μικρό κομμάτι το οποίο αποτελεί μέρος του συνόλου μιας διεπαφής χρήστη. Για παράδειγμα ένα Component στην διεπαφή χρήστη του Facebook είναι το παράθυρο συνομιλίας, το πεδίο για την τοποθέτηση σχολίου, κλπ. Όλα αυτά τα κομμάτια συνυπάρχουν στο ίδιο πλαίσιο, αλλά αλληλεπιδρούν ανεξάρτητα το ένα από το άλλο. Το Component βασίζεται στην ιδέα των αιτημάτων AJAX[15] στα οποία οι κλήσεις προς τον εξυπηρετητή γίνονται απευθείας από την πλευρά του πελάτη, επιτρέποντας την δυναμική ενημέρωση του DOM[16], χωρίς την ανανέωση ολόκληρης της σελίδας. Επειδή το κάθε κομμάτι είναι ανεξάρτητο, μπορεί να ανανεωθεί χωρίς να επηρεάσει τα άλλα κομμάτια και την διεπαφή χρήστη στο σύνολό της.



Εικόνα 7: Component based αρχιτεκτονική

Το κάθε Component έχει την δικιά του δομή, τις δικές του μεθόδους και τις δικές του διεπαφές. Αυτό δίνει την δυνατότητα στους προγραμματιστές να δημιουργήσουν μια διεπαφή χρήστη με ποικίλα διαφορετικά κομμάτια, τα οποία αποτελούν μέρη του συνόλου της εφαρμογής. Μια τέτοια διεπαφή χρήστη, η οποία αποτελείται από πολλαπλά Components, είθισται να λέγεται πως ακολουθεί Component-based αρχιτεκτονική (βλ. Εικόνα 7).

2.5.4 Διαφορές μεταξύ CBA και MVC αρχιτεκτονικών

Η διαφορά ανάμεσα σε μια MVC αρχιτεκτονική και μία CBA, είναι ότι, ενώ η MVC αρχιτεκτονική διαχωρίζει τα καθήκοντα κατακόρυφα, η αρχιτεκτονική βασισμένη στο Component τα διαχωρίζει οριζόντια. Αλλά τι ακριβώς σημαίνει αυτό; Ουσιαστικά, αν χρησιμοποιείται ένα γενικό πλαίσιο, όπως η AngularJs, υπάρχουν HTML πρότυπα που παρουσιάζουν την διεπαφή χρήστη, διαδρομές που καθορίζουν ποια πρότυπα να παρουσιαστούν και διάφορες υπηρεσίες με βοηθητικές μεθόδους. Ακόμα και αν ένα πρότυπο περιέχει διαδρομές, μεθόδους και υπηρεσίες όπως αναφέρθηκε προηγουμένως, όλα αυτά υπάρχουν σε διαφορετικά επίπεδα της αρχιτεκτονικής της εφαρμογής. Ακολουθείται λοιπόν μια κάθετη σειρά επιπέδων, παρόμοια με αυτή της Εικόνας 6.

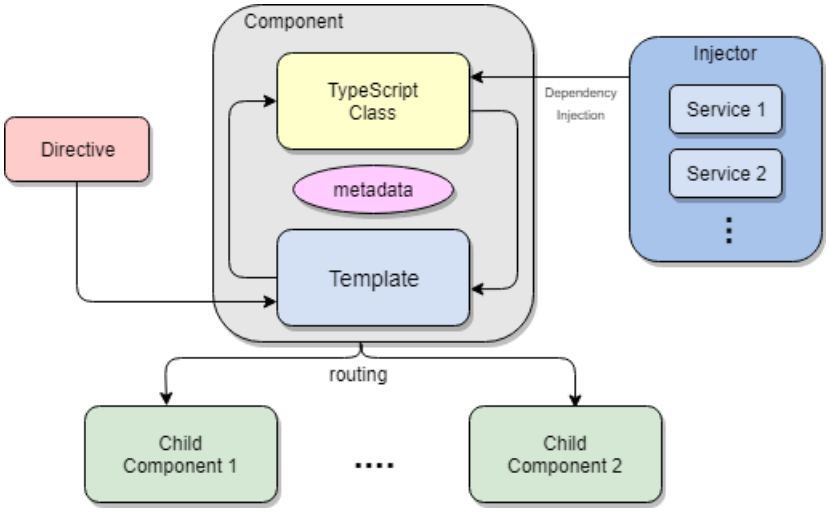
Αντίθετα, στην περίπτωση της αρχιτεκτονικής με βάση το Component, τα καθήκοντα διαχωρίζονται σε μια κοινή βάση από Components. Αυτό σημαίνει ότι ο σχεδιασμός, η επιχειρησιακή λογική, οι υπηρεσίες και οι μέθοδοι υπάρχουν σε ένα ίδιο επίπεδο αρχιτεκτονικής, συνήθως του επιπέδου της παρουσίασης (View layer). Όλα όσα σχετίζονται με ένα Component, ορίζονται στην κλάση του. Η

[15] AJAX: Μέθοδος για την φόρτωση δεδομένων ξεχωριστά από το υπόλοιπο HTML έγγραφο
[16] DOM: Διεπαφή για HTML και XML αρχεία, που συνδέει μια διαδικτυακή σελίδα με μια γλώσσα προγραμματισμού

αρχιτεκτονική με βάση το Component αντιπροσωπεύει μια αρχιτεκτονική που ενθαρρύνει την ελευθερία του προγραμματισμού και την ευελιξία της εφαρμογής σε αντίθεση με την ακαμψία της MVC αρχιτεκτονικής. Το εμπρόσθιο μέρος της εφαρμογής, εκείνο δηλαδή μέσω του οποίου ο χρήστης καταναλώνει τις υπηρεσίες της εφαρμογής, έχει αναπτυχθεί με την Angular 7, ενός framework που ακολουθεί Component-based αρχιτεκτονική. Στις επόμενες ενότητες θα παρουσιαστούν τα σημαντικότερα κομμάτια της Angular 7, έτσι ώστε μετέπειτα που θα αναδειχθούν τα components που δημιουργήθηκαν για την εφαρμογή, να είναι πλήρως κατανοητή η δομή τους.

2.5.5 Τα κύρια κομμάτια της Angular 7

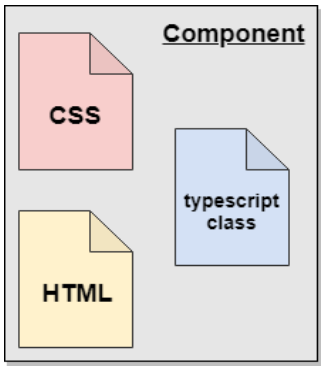
Με όσα ειπώθηκαν και παραπάνω, μια Angular [13] εφαρμογή αποτελείται από πολλά Components. Αυτά τα Components συνδέονται μεταξύ τους μέσω routing ή μέσω selectors. Επιπλέον μπορεί να περιέχουν ένα HTML πρότυπο ή μπορεί να χρησιμοποιούν μία υπηρεσία, την οποία πρέπει πρώτα να εξάγουν μέσα στο Component για να μπορέσουν να την χρησιμοποιήσουν. Η παρακάτω εικόνα (βλ. Εικόνα 8), δείχνει την ραχοκοκαλιά της Angular 7, όπου στη συνέχεια θα παρουσιαστεί συνοπτικά κάθε κομμάτι της.



Εικόνα 8: Τα μέρη της Angular

- Modules

Η Angular αποτελείται από πολλά Modules τα οποία με την σειρά τους αποτελούνται από ένα ή περισσότερα Components, Services και Directives. Ένα Module είναι ένα σύνολο βοηθητικών βιβλιοθηκών οι οποίες εκτελούν παρόμοιες εργασίες και έχει την δυνατότητα να εξάγει μια κλάση, η οποία μπορεί να εισαχθεί σε άλλα Modules. Η ίδια η Angular διαθέτει ποικίλα και μεγάλα Modules, όπως για παράδειγμα το core ή το router module το οποίο εξάγει μεθόδους για την δρομολόγηση στην εφαρμογή και για την χρησιμοποίησή του αρκεί να εισαχθεί στο αντίστοιχο Component.



Εικόνα 9: Περιεχόμενα του Component

- Components

Τα Components είναι τα βασικά δομικά στοιχεία της εφαρμογής. Ένα Component είναι συνήθως μια typescript κλάση, η οποία αλληλοεπιδρά με ένα HTML πρότυπο και ένα CSS αρχείο (βλ. Εικόνα 9). Το HTML πρότυπο είναι υπεύθυνο για την εικόνα του Component, ενώ το CSS αρχείο αναλαμβάνει την διακόσμηση αυτής της εικόνας. Ένα Component έχει ένα μεταδεδομένο (metadata), το οποίο λέει στην Angular ότι η σχετική κλάση πρέπει να θεωρείται ως Component. Τα decorators, όπως ονομάζονται αυτά τα μεταδεδομένα, στην πραγματικότητα είναι JavaScript συναρτήσεις που με κάποιο τρόπο τροποποιούν την κλάση. Γίνεται αντιληπτό πως ένα Component περιέχει επιχειρησιακή λογική και δεδομένα μέσω της Typescript[17] κλάσης και εικόνα μέσω του HTML προτύπου. Αποτελεί δηλαδή ένα MVC μοντέλο από μόνο του.

- Υπηρεσίες

Για την εκμετάλλευση των πλεονεκτημάτων που προσφέρει η Angular και την καλύτερη δομή της εφαρμογής, συγκεκριμένες εργασίες είναι καλό να ανατίθενται σε διαφορετικές υπηρεσίες για την εκτέλεσή τους. Ένα Component μπορεί να καταναλώνει αυτές τις υπηρεσίες για να εκτελέσει κάποια εργασία. Συνήθως ένα component θα πρέπει να ασχολείται με την εμπειρία του χρήστη και την εμφάνιση της εικόνας σε αυτόν και να χρησιμοποιεί τις υπηρεσίες κατάλληλα για την εκτέλεση λειτουργιών. Μια υπηρεσία, πριν χρησιμοποιηθεί στο Component, πρέπει να εισαχθεί στον χειριστή. Αυτό πραγματοποιείται μέσω μίας τεχνικής, για την οποία έγινε αναφορά και σε προηγούμενη ενότητα και ονομάζεται εισαγωγή εξάρτησης (Dependency Injection).

- Το σχεδιαστικό μοτίβο Dependency Injection στην Angular

Ένα από τα βασικά χαρακτηριστικά της Angular είναι η εισαγωγή εξάρτησης. Το Component, όπως αναφέρθηκε παραπάνω, χρησιμοποιεί υπηρεσίες για την εκτέλεση εργασιών και οι υπηρεσίες αυτές εισάγονται στο Component μέσω του εισαγωγέα. Ο εισαγωγέας παρέχει στο Component μια εικόνα της υπηρεσίας την οποία μπορεί να χρησιμοποιήσει. Η εισαγωγή αυτή συνήθως λαμβάνει χώρα στον κατασκευαστή της κλάσης, με την δημιουργία δηλαδή του Component ώστε να χρησιμοποιηθεί η υπηρεσία στη συνέχεια. Η ιδέα της εισαγωγής εξάρτησης είναι να διαχωρίζει τα καθήκοντα σε μικρότερες μονάδες, όπου τα Components θα εξαρτώνται από αυτές τις μονάδες για την εκτέλεση των εργασιών. Με αυτόν τον τρόπο καθιστά την εφαρμογή πιο διαχειρίσιμη και εύκολη στην δοκιμή.

- Directives

Τα Directives (ελλ. οδηγίες) βρίσκονται παντού στην Angular. Ένα Directive είναι ουσιαστικά μια typescript κλάση με μεταδεδομένα. Τα Directives μπορεί να έχουν συνημμένο ένα πρότυπο (template), αλλά είναι επίσης δυνατό να μην έχουν. Το Component αποτελεί ένα παράδειγμα Directive με πρότυπο. Τα είδη οδηγιών στην Angular είναι δύο. Οι οδηγίες δομής (Structural Directives) και οι οδηγίες ιδιοτήτων (Properties Directives). Οι οδηγίες δομής τροποποιούν την δομή ή την διάταξη του DOM, ενώ οι οδηγίες ιδιοτήτων μεταβάλλουν την συμπεριφορά των στοιχείων του. Μερικά παραδείγματα οδηγιών που χρησιμοποιούνται και στην εφαρμογή είναι το *ngFor, *ngIf,

[17] Typescript: Γλώσσα προγραμματισμού που χρησιμοποιείται στην Angular. Πρόκειται για ένα υπερσύνολο της Javascript

ngModel και άλλα. Ουσιαστικά μέσω των Directives επιτυγχάνεται η σύνδεση και αλληλεπίδραση μεταξύ των typescript και HTML αρχείων. Μέσω του *ngFor Directive για παράδειγμα, παρέχεται η δυνατότητα επαναληπτικής διαδικασίας σε ένα HTML αρχείο με μεταβλητές και βήματα που καθορίζονται στο typescript αρχείο, ενώ αντίστοιχα με το *ngIf η δημιουργία συνθήκης για την απόκρυψη ή εμφάνιση HTML κομματιών.

Σε αυτή την ενότητα αναλύθηκαν τα βασικά στοιχεία μιας Angular εφαρμογής. Με βάση την Εικόνα 8 και με την προυπόθεση ότι η Angular ακολουθεί πλέον αρχιτεκτονική που βασίζεται στο component, δείχθηκε ο τρόπος με τον οποίο συνδέονται αυτά τα στοιχεία γύρω από αυτό.

2.5.6 Η βιβλιοθήκη RxJS

Η RxJS [14] αποτελεί μια βιβλιοθήκη του reactive προγραμματισμού. Με τον όρο reactive προγραμματισμός αναφερόμαστε στον προγραμματισμό με βάση την ασύγχρονη ροή δεδομένων. Δημιουργώντας ροές δεδομένων από ένα κλικ σε ένα πεδίο της ιστοσελίδας μέχρι και τα δεδομένα που δέχεται από έναν εξυπηρετητή, τα τρία σημαντικότερα κομμάτια αυτού του τύπου προγραμματισμού και συγκεκριμένα της RxJS βιβλιοθήκης είναι τα Observables, οι Observers και οι Schedulers. Τα Observables εκπέμπουν ροές δεδομένων είτε περιοδικά, είτε μια φορά, ενώ οι Observers είναι οι παρατηρητές που λαμβάνουν τα δεδομένα που εκπέμπονται. Όπως αναφέρθηκε προηγουμένως, η ροή των δεδομένων είναι ασύγχρονη και για αυτό τον λόγο η χρήση ενός διαχειριστή νημάτων είναι απαραίτητη και αυτό τον ρόλο αναλαμβάνουν οι Schedulers. Η βιβλιοθήκη RxJS χρησιμοποιείται κατά κόρον στην διεπαφή χρήστη της εφαρμογής, παρέχοντας μεγαλύτερη ταχύτητα και καλύτερη εμπειρία χρήσης.

Για παράδειγμα, έχοντας δύο ή παραπάνω components ενσωματωμένα σε ένα άλλο component, όπου υπάρχει εξάρτηση μεταξύ αυτών, μέσω του reactive προγραμματισμού και της RxJS βιβλιοθήκης επιτυγχάνεται η σύνδεση και ενημέρωσή τους. Μόλις αλλάξει τιμή ένα από τα components, αυτή η νέα τιμή εκπέμπεται (Observable) και όποιο component είναι εγγραμμένο σε αυτό το Observable θα ειδοποιηθεί και θα λάβει την τιμή αυτόματα. Έπειτα, αν υπάρχει ανάγκη να ενημερώσει τις τιμές του, καλεί τις αντίστοιχες μεθόδους και ενημερώνεται.

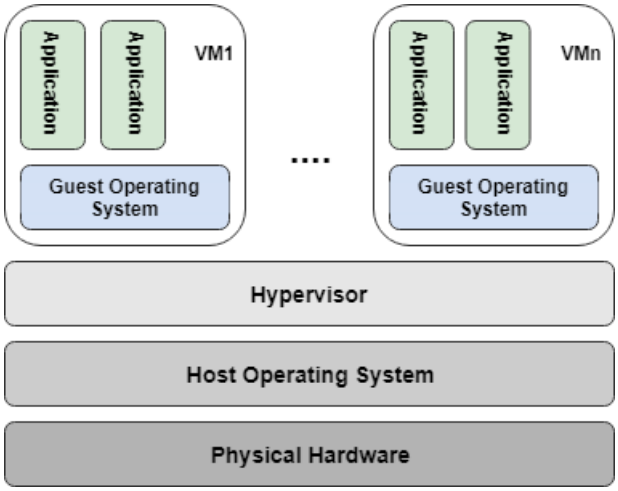
Το παραπάνω παράδειγμα αποτελεί μία από τις πολλές περιπτώσεις, όπου μπορεί να χρησιμοποιηθεί η RxJS βιβλιοθήκη και κατά επέκταση ο reactive προγραμματισμός. Οι τελεστές της RxJS βιβλιοθήκης επίσης, κατέχουν σημαντικό ρόλο και μέσα από την χρησιμοποίηση τους δίνεται η δυνατότητα στον προγραμματιστή να διαμορφώσει την ροή δεδομένων σύμφωνα με τις ανάγκες του. Για παράδειγμα έχει την δυνατότητα να λάβει ένα συγκεκριμένο αριθμό τιμών, τον πρώτο ή τον τελευταίο που εκπέμπεται, να φιλτράρει τις τιμές και άλλα.

2.6 Εικονικοποίηση (Virtualization)

Στην τελευταία ενότητα του δευτέρου κεφαλαίου θα παρουσιαστούν συνοπτικά δύο τεχνολογίες για την εικονικοποίηση, η οποία χρησιμοποιήθηκε για την εγκατάσταση της εφαρμογής. Εξετάζοντας την εικονικοποίηση σε ένα ευρύτερο πλαίσιο, ορίζεται ως η τεχνολογία με την οποία είναι δυνατή η εξομοίωση της λειτουργίας ενός αντικειμένου ή ενός πόρου με την αντίστοιχη λειτουργία του φυσικού αντικειμένου. Με άλλα λόγια, η εικονικοποίηση αναφέρεται στη δημιουργία μιας εικονικής πηγής, όπως για παράδειγμα ενός εξυπηρετητή, ενός λειτουργικού συστήματος, ενός αρχείου ή ενός δικτύου. Η εικονικοποίηση είναι στο προσκήνιο της πληροφορικής για δεκαετίες και σήμερα εφαρμόζεται σε ένα μεγάλο εύρος σε διάφορα επίπεδα συστημάτων, όπως η εικονικοποίηση του επιπέδου του λειτουργικού, του υλικού αλλά και η εικονικοποίηση εξυπηρετητή.

2.6.1 Εικονική μηχανή (Virtual Machine – VM)

Ο πιο κοινός τύπος εικονικοποίησης είναι η εικονικοποίηση υλικού [15]. Σε μια τέτοιου τύπου εικονικοποίηση είναι δυνατό να τρέχουν πολλαπλά λογισμικά συστήματα πάνω σε ένα και μόνο κομμάτι υλικού. Η τεχνολογία της εικονικοποίησης περιλαμβάνει τον διαχωρισμό του φυσικού υλικού και του λογισμικού, εξομοιώνοντας το υλικό χρησιμοποιώντας λογισμικό. Όταν διαφορετικά λειτουργικά συστήματα τρέχουν στην κορυφή ενός πρωταρχικού λειτουργικού συστήματος μέσω της εικονικοποίησης, τότε είναι δυνατή η αναφορά σε αυτό ως εικονική μηχανή.



Εικόνα 10: Αρχιτεκτονική εικονικής μηχανής

Μια εικονική μηχανή μιμείται έναν εξυπηρετητή. Σε έναν τέτοιο εικονικοποιημένο εξυπηρετητή, κάθε εικονική μηχανή που τρέχει πάνω σε αυτόν περιλαμβάνει ένα ολοκληρωμένο λειτουργικό σύστημα με τους κατάλληλους οδηγούς και βιβλιοθήκες καθώς και την εφαρμογή. Κάθε εικονική μηχανή έχει τον δικό της Kernel[18] και το δικό της λειτουργικό σύστημα και μπορεί να χαρακτηριστεί ως πλήρως ανεξάρτητη.

Μια εικονική μηχανή δεν είναι τίποτε άλλο από ένα αρχείο δεδομένων σε έναν φυσικό υπολογιστή, το οποίο μπορεί να μετακινηθεί και να αντιγραφεί σε έναν άλλο υπολογιστή, όπως ακριβώς ένα απλό αρχείο δεδομένων. Οι υπολογιστές σε ένα εικονικό περιβάλλον χρησιμοποιούν δύο τύπους δομών αρχείων. Έναν που καθορίζει το υλικό και έναν ο οποίος καθορίζει τον σκληρό δίσκο.

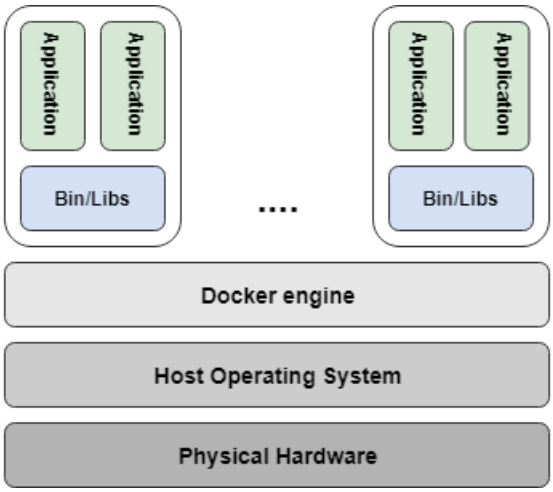
Το λογισμικό της εικονικοποίησης ή αλλιώς ο Hypervisor είναι ένα λεπτό επίπεδο λογισμικού το οποίο λειτουργεί ως διαχειριστής και διαχειρίζεται όλες τις εικονικές μηχανές οι οποίες τρέχουν πάνω στον εξυπηρετητή. Ουσιαστικά πρόκειται για ένα λογισμικό που καθορίζει το πως οι εικονικοί πόροι αντιστοιχίζονται στους φυσικούς πόρους. Όπως φαίνεται και στην Εικόνα 10, το πρώτο επίπεδο αποτελείται από το φυσικό υλικό. Ακολουθεί το λειτουργικό σύστημα (π.χ. Linux) και πάνω σε αυτό λειτουργεί ο Hypervisor, ο οποίος διαχειρίζεται το τελευταίο επίπεδο, δηλαδή τις εικονικές μηχανές. Κάθε εικονική μηχανή αποτελεί ένα κουτί, το οποίο μέσα περιέχει την εφαρμογή και το δικό της λειτουργικό σύστημα.

[18] Kernel: Πυρήνας και θεμέλιο τμήμα ενός λειτουργικού συστήματος

2.6.2 Containerization

Το Containerization [16] είναι μια διαφορετική και πιο ελαφριά λύση από την ολοκληρωτική εικονικοποίηση της μηχανής. Ένα Container είναι μια μονάδα λογισμικού που πακετάρει όλο τον κώδικα και τις εξαρτήσεις που χρειάζεται η εφαρμογή ώστε να λειτουργήσει αρμονικά και να τρέξει σε οποιοδήποτε υπολογιστικό περιβάλλον.

Με τα Containers εικονικοποιείται μόνο το λειτουργικό σύστημα που τρέχει πάνω από το φυσικό εξυπηρετητή. Κάθε Container μοιράζεται αυτό το λειτουργικό σύστημα και τις βιβλιοθήκες του. Τον ρόλο του Hypervisor σε αυτή την περίπτωση τον παίζει μια μηχανή για Containers, όπως το Docker, η οποία είναι εγκατεστημένη πάνω από το λειτουργικό σύστημα (βλ. Εικόνα 11).



Εικόνα 11: Αρχιτεκτονική των containers

Τα κύρια πλεονεκτήματα των Containers είναι το κέρδος και η υψηλή αποδοτικότητα όσο αφορά την μνήμη, τον επεξεργαστή αλλά και τον αποθηκευτικό χώρο που καταναλώνουν. Ένα ακόμα πλεονέκτημα των Containers είναι η φορητότητα. Εφόσον το λειτουργικό σύστημα είναι το ίδιο, ένα Container έχει την δυνατότητα να τρέξει σε οποιοδήποτε σύστημα και σε οποιοδήποτε νέφος, χωρίς να απαιτούνται αλλαγές στον κώδικα. Συνοπτικά οι δύο τύποι container και είναι οι εξής:

- Linux Containers (LXC)

Η αυθεντική τεχνολογία Container είναι τα Linux Containers, γνωστά ως LXC. Τα LXC είναι ένα Linux λειτουργικό σύστημα στο επίπεδο της εικονικοποίησης που παρέχει μεθόδους για να τρέχουν πολλαπλά απομονωμένα συστήματα Linux σε έναν και μόνο πάροχο.

- Docker

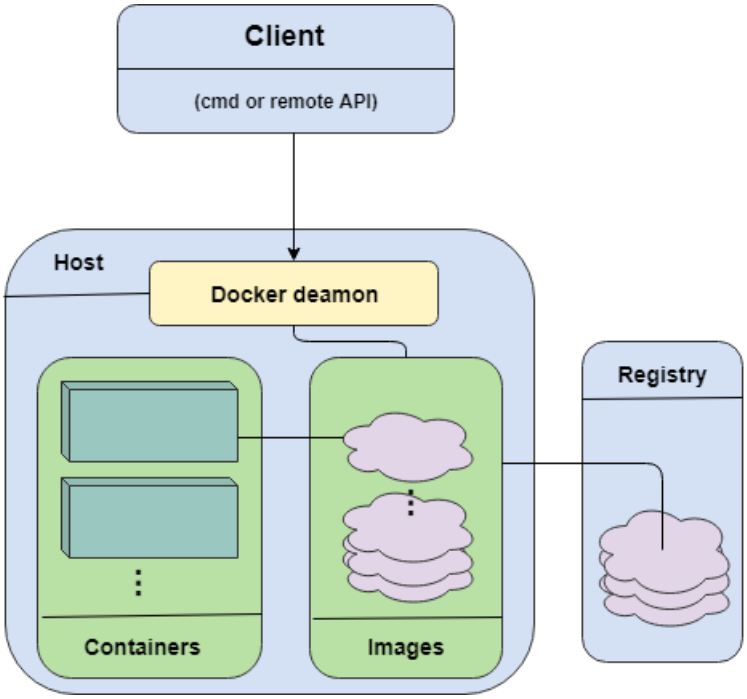
Το Docker ξεκίνησε ως ένα έργο για την δημιουργία LXC containers και παρουσίασε πολλές αλλαγές στο LXC, κάνοντας τα Containers περισσότερο φορητά και ευέλικτα στην χρήση τους. Με την πάροδο του χρόνου μεταμορφώθηκε σε ένα ολοκληρωμένο περιβάλλον Container, το οποίο μπορεί με αποδοτικότητα να δημιουργήσει και να τρέξει Containers.

2.6.3 Docker

Το Docker [17] [18] είναι ένα δημοφιλές, ανοιχτού κώδικα, έργο το οποίο, όπως αναφέρθηκε και στην προηγούμενη ενότητα, είναι βασισμένο πάνω στα Linux containers. Είναι ουσιαστικά μια μηχανή container που χρησιμοποιεί τα χαρακτηριστικά του Linux kernel για την δημιουργία containers πάνω από ένα λειτουργικό σύστημα και αυτοματοποιεί την ανάπτυξη εφαρμογών στο container. Διαθέτει μια αποτελεσματική ροή για την μετακίνηση της εφαρμογής από τους υπολογιστές των προγραμματιστών και τα περιβάλλοντα δοκιμής, στην παραγωγή.

Η ροή αυτή ακολουθήθηκε για την εγκατάσταση της εφαρμογής στον εξυπηρετητή και για την επιτυχή ολοκλήρωσή της χρησιμοποιήθηκαν τα εξής κομμάτια:

1. Docker Client and Daemon
2. Images
3. Docker registries
4. Containers



Εικόνα 12: Τα μέρη του Docker

Στην παραπάνω εικόνα (βλ. Εικόνα 12), απεικονίζονται αυτά τα τέσσερα μέρη του Docker τα οποία θα παρουσιαστούν στη συνέχεια. Το Docker ακολουθεί μια αρχιτεκτονική πελάτη-εξυπηρετητή.

Το docker deamon, που έχει τον ρόλο του εξυπηρετητή, είναι υπεύθυνο για όλες τις ενέργειες που σχετίζονται με τα containers. Το docker deamon, όπως συμβαίνει σε κάθε αρχιτεκτονική πελάτη-εξυπηρετητή, δέχεται αιτήματα/εντολές από τον Docker πελάτη μέσω του CLI ή μιας REST διεπαφής. Ο Docker πελάτης μπορεί να λειτουργεί στον ίδιο πάροχο με τον docker deamon ή σε διαφορετικό από αυτόν που τρέχει το Docker deamon. Ως εκ τούτου, το πρώτο βήμα ήταν η εγκατάσταση της Docker μηχανής και κατά επέκταση του docker deamon στον εξυπηρετητή.

Οι εικόνες αποτελούν τον πυλώνα του Docker, εφόσον τα containers χτίζονται από αυτές. Οι εικόνες μπορούν να διαμορφωθούν με εφαρμογές και να χρησιμοποιηθούν σαν πρότυπα για την δημιουργία containers. Για παράδειγμα, για την δημιουργία ενός container μιας Angular εφαρμογής, αρκεί να προστεθεί ο φάκελος της εφαρμογής στην εικόνα ενός εξυπηρετητή ιστού και έπειτα να δημιουργηθεί το container. Συγκεκριμένα, για την εφαρμογή χρησιμοποιήθηκαν οι εικόνες των Ubuntu, NGINX και MySQL. Σε κάθε μία από αυτές πραγματοποιήθηκε η κατάλληλη τροποποίηση για τις ανάγκες της εφαρμογής και έπειτα πάνω σε αυτές δημιουργήθηκαν τα τρία συνολικά containers.

Η αποθήκη στην οποία υπάρχουν εικόνες Docker, ονομάζεται Docker registry. Χρησιμοποιώντας το Docker registry, παρέχεται η δυνατότητα ανάπτυξης εικόνων, καθώς και η διαμοίρασή τους. Ένα τέτοιο αρχείο μπορεί να είναι είτε ιδιωτικό είτε δημόσιο. Το Docker παρέχει μια υπηρεσία, που ονομάζεται Docker hub και επιτρέπει στον προγραμματιστή να κατεβάσει και να ανεβάσει εικόνες από και προς μια κεντρική τοποθεσία. Αν η αποθήκη του χρήστη είναι δημόσια, όλες οι εικόνες που βρίσκονται μέσα σε αυτή είναι προσβάσιμες από άλλους χρήστες του Docker hub. Ο χρήστης έχει την δυνατότητα να δημιουργήσει και ιδιωτικό αρχείο στο Docker hub. Το docker hub λειτουργεί όπως το git[19], αφού ο προγραμματιστής έχει την δυνατότητα να αναπτύξει εικόνες τοπικά στον υπολογιστή του και έπειτα να τις περάσει στο Docker hub. Η εικόνα που δημιουργήθηκε με βάση το Ubuntu[20] και τρέχει στο παρασκήνιο της εφαρμογής, αποτελεί μια πρωτότυπη εικόνα που συνδέει το Java πρόγραμμα με την SWI-Prolog και μέχρι σήμερα δεν υπήρχε κάτι παρόμοιο.

Τέλος, το container αποτελεί το περιβάλλον εκτέλεσης για το Docker. Τα containers δημιουργούνται από εικόνες και είναι ένα εγγράψιμο επίπεδό τους. Υπάρχει η δυνατότητα ένα container που περιέχει μια εφαρμογή και έχει εισαχθεί στο Docker hub, να χρησιμοποιηθεί σαν εικόνα ώστε πάνω σε αυτό να δημιουργηθούν νέα containers.

2.7 Ανασκόπηση κεφαλαίου

Με την ενότητα της εικονικοποίησης ολοκληρώνεται το δεύτερο κεφάλαιο. Σκοπός του κεφαλαίου αυτού ήταν η η παροχή υποβάθρου για την κατανόηση της εφαρμογής που θα παρουσιαστεί στο επόμενο κεφάλαιο. Στις πρώτες ενότητες του κεφαλαίου αποσαφηνίστηκαν σημαντικές έννοιες πάνω στο διαδίκτυο και έπειτα παρουσιάστηκε η γλώσσα Prolog και το εργαλείο Gorgias. Εν συνεχεία παρουσιάστηκαν τα σημαντικότερα κομμάτια των εργαλείων που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής. Τέλος, έγινε μια αναφορά στις τεχνολογίες που χρησιμοποιούνται σήμερα για την εγκατάσταση εφαρμογών, αλλά και στην τεχνολογία με την οποία πραγματοποιήθηκε η εγκατάσταση της εφαρμογής. Εφόσον όλα τα παραπάνω είναι κατανοητά, στο επόμενο κεφάλαιο θα παρουσιαστεί η εφαρμογή.

[19] Git: Σύστημα ελέγχου εκδόσεων

[20] Ubuntu: Λειτουργικό σύστημα βασισμένο στον πυρήνα Linux

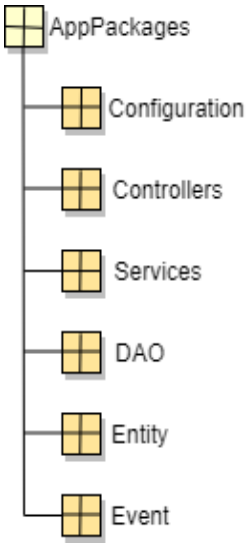
Κεφάλαιο 3 - Ανάπτυξη της εφαρμογής

Η εφαρμογή που αναπτύχθηκε απαρτίζεται από τρία μέρη. Το κυριότερο μέρος είναι το Back End, το μέρος δηλαδή που τρέχει στο παρασκήνιο. Αυτό το μέρος της εφαρμογής αναπτύχθηκε με την χρήση του Spring boot framework και στις επόμενες ενότητες θα παρουσιαστεί η δομή του, τα πακέτα και οι κλάσεις που δημιουργήθηκαν. Στη συνέχεια θα παρουσιασθεί η βάση δεδομένων και πως επιτεύχθηκε η σύνδεση της εφαρμογής με αυτή και τέλος θα παρουσιαστούν οι τρόποι με τους οποίους καταναλώνεται η διαδικτυακή αυτή υπηρεσία.

3.1 Πακέτα και εξαρτήσεις της υπηρεσίας

Για την δημιουργία της διαδικτυακής αυτής υπηρεσίας χρησιμοποιήθηκε το εργαλείο Maven. Το Maven είναι ένα εργαλείο το οποίο εκτελώντας συγκεκριμένα βήματα αναλαμβάνει να ετοιμάσει την εφαρμογή, ώστε να λειτουργήσει αρμονικά. Σε ένα XML αρχείο δηλώνονται όλα τα πακέτα που χρειάζονται να χρησιμοποιηθούν στην εφαρμογή και το Maven αναλαμβάνει να κατεβάσει όλες τις απαραίτητες εξαρτήσεις για να πραγματοποιηθεί η χρήση των πακέτων αυτών.

Σε ότι αφορά την εφαρμογή, στην Εικόνα 13, φαίνονται συνολικά όλα τα πακέτα που δημιουργήθηκαν για τις ανάγκες της εφαρμογής.



Εικόνα 13: Πακέτα της διαδικτυακής υπηρεσίας

Στο πακέτο Configuration υπάρχουν όλες οι κλάσεις που σχετίζονται με την διαμόρφωση της εφαρμογής, κλάσεις που αρχικοποιούν το Spring και γενικά την εφαρμογή. Εκεί διαμορφώνεται το Hibernate που, όπως θα παρουσιαστεί και στην συνέχεια, χρησιμοποιείται για την σύνδεση με την βάση δεδομένων της εφαρμογής. Εκεί επίσης υπάρχει η αντίστοιχη κλάση που διαμορφώνει το Sping Security framework για την ασφάλεια της εφαρμογής και τέλος μέσα στις κλάσεις αυτού του πακέτου δημιουργούνται και όλα τα αντικείμενα τις εφαρμογής.

Στο πακέτο Controllers βρίσκονται όλοι οι χειριστές της εφαρμογής. Για καλύτερη δομή υπάρχουν παραπάνω από ένας χειριστές. Λεπτομέρειες για το τι κάνει ο καθένας θα δοθούν στη συνέχεια.

Το πακέτο services αποτελεί μια διεπαφή με όλες τις μεθόδους των υπηρεσιών της εφαρμογής και την αντίστοιχη κλάση που την υλοποιεί. Επιπρόσθετα, υπάρχει μια κλάση που χρησιμοποιείται ως υπηρεσία στην είσοδο του χρήστη για επαλήθευση στοιχείων.

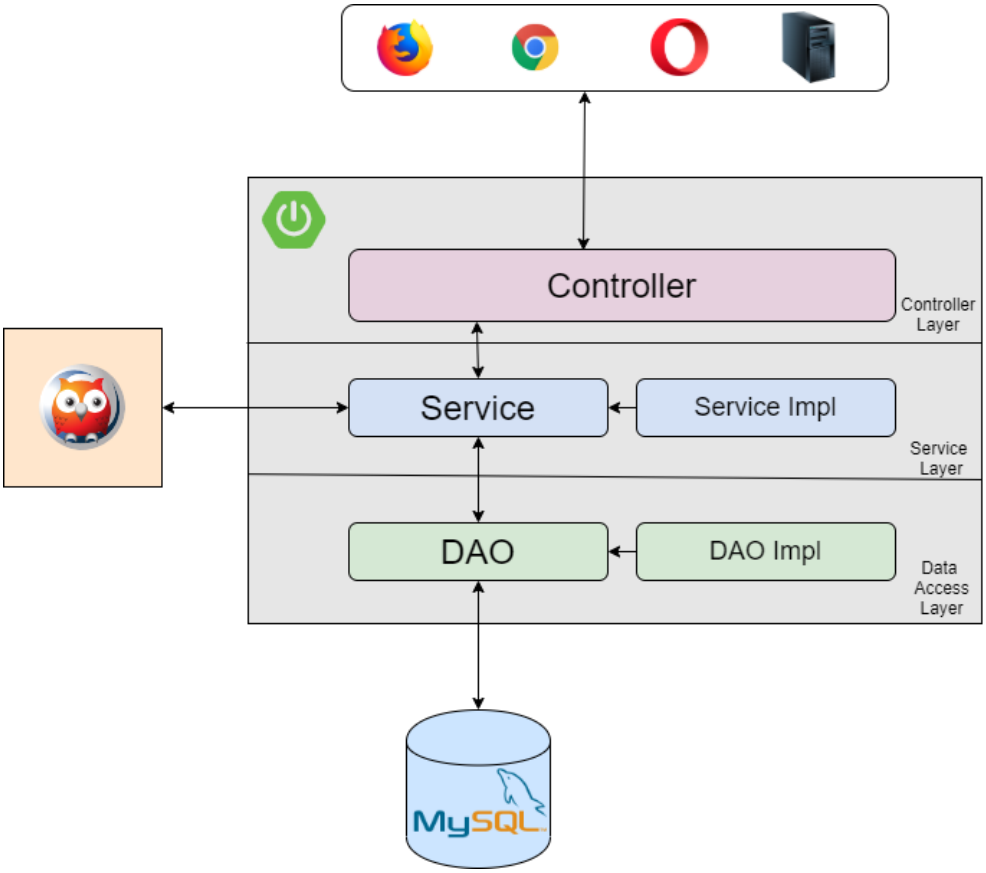
Το πακέτο DAO αποτελείται από κλάσεις που έχουν τον ρόλο του συνδετικού κρίκου μεταξύ της βάσεως δεδομένων και των υπηρεσιών της εφαρμογής. Συνολικά υπάρχουν τέσσερις διεπαφές με μεθόδους σύνδεσης με την βάση δεδομένων και άλλες τέσσερις κλάσεις που τις υλοποιούν. Μια διεπαφή ασχολείται με τα αρχεία, η άλλη με τους φακέλους, άλλη μία περιέχει μεθόδους για τον χρήστη, ενώ υπάρχει και μια διεπαφή με μεθόδους σύνδεσης με την βάση και αφορά τον κλειδάριθμο που δίνεται στον χρήστη κατά την εγγραφή του.

Στο πακέτο Entity, έχουν δημιουργηθεί κλάσεις με τα αντικείμενα της εφαρμογής. Υπάρχουν συνολικά επτά κλάσεις στο πακέτο αυτό, όπου για κάθε κλάση υπάρχει και ο αντίστοιχος πίνακας στη βάση δεδομένων της εφαρμογής.

Τέλος, το πακέτο Event περιέχει κλάσεις, οι οποίες χρησιμοποιούνται για την ανίχνευση γεγονότος, όπως για παράδειγμα μιας νέας εγγραφής χρήστη και αποστολής σε αυτόν email ενεργοποίησης λογαριασμού ή η λήξη μιας συνεδρίας.

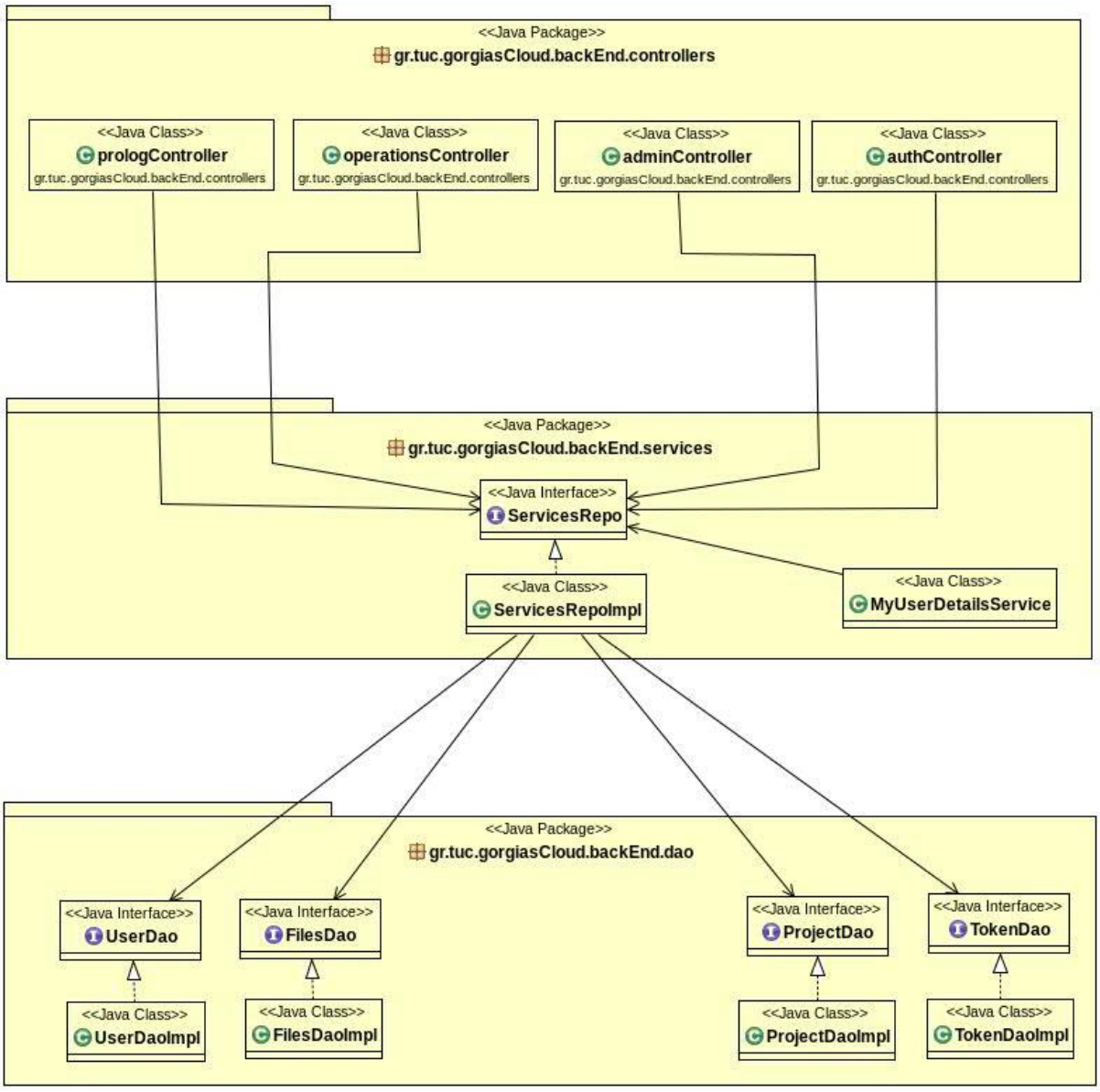
3.2 Η αρχιτεκτονική της εφαρμογής στο μέρος του εξυπηρετητή

Η Εικόνα 14, δείχνει την αρχιτεκτονική που ακολουθεί το μέρος που τρέχει στο παρασκήνιο της εφαρμογής. Όπως γίνεται αντιληπτό, ο πελάτης που είναι είτε ένας περιηγητής διαδικτύου, είτε ένα άλλο πρόγραμμα, όπως παραδείγματος χάρη το Gorgias, στέλνει αιτήματα στην υπηρεσία και ο χειριστής (Controller) αναλαμβάνει να τα διαχειριστεί. Ανάλογα με το αίτημα, καλεί την αντίστοιχη υπηρεσία (Service) που και αυτή με την σειρά της επικοινωνεί με την βάση δεδομένων, αν είναι αναγκαίο, μέσω των DAO (Data Access Object) κλάσεων. Εν συνεχεία, η ροή ακολουθεί αντίθετη διαδρομή και η βάση δεδομένων περνάει τα δεδομένα στο επίπεδο υπηρεσιών μέσω των DAO κλάσεων. Έπειτα, το επίπεδο υπηρεσιών επιστρέφει αυτά τα δεδομένα στον χειριστή, όπου εκεί δημιουργείται και αποστέλνεται η απόκριση της υπηρεσίας. Σε περίπτωση που ο χρήστης αιτηθεί μία υπηρεσία που έχει σχέση με την μηχανή Prolog, τότε ο κατάλληλος χειριστής επικοινωνεί με το επίπεδο υπηρεσιών, καλεί την κατάλληλη υπηρεσία, η οποία επικοινωνεί με την μηχανή Prolog και επιστρέφει στον χειριστή τα αποτελέσματα. Ο χειριστής με την σειρά του επιστρέφει στον πελάτη την απόκριση της υπηρεσίας σε μορφή JSON. Στην δεύτερη περίπτωση και γενικά σε αιτήματα που αφορούν την Prolog και απαιτούν σύνδεση με αυτή, η εφαρμογή δεν αλληλοεπιδρά με την βάση δεδομένων και η ροή των ενεργειών δεν περνάει από τις DAO κλάσεις.



Εικόνα 14: Αρχιτεκτονική της εφαρμογής στο μέρος του εξυπηρετητή

Στην Εικόνα 14 φαίνεται η μεγάλη εικόνα όσο αφορά την αρχιτεκτονική στο μέρος του εξυπηρετητή. Μια γενική ιδέα δηλαδή για την ροή των δεδομένων από την παραλαβή του αιτήματος μέχρι και την απόκριση της υπηρεσίας. Στην Εικόνα 15 φαίνονται πιο συγκεκριμένα πλέον οι κλάσεις και οι διεπαφές που δημιουργήθηκαν για τις ανάγκες τις εφαρμογής. Αρχικά υπάρχει ένα πακέτο που περιέχει τους Controllers. Θα ήταν δυνατό να υπήρχε μόνο ένας χειριστής, όπου όλες οι μέθοδοι και οι αντιστοιχίσεις να πραγματοποιούταν εκεί. Για λόγους καλύτερης δομής και ευχρηστίας όμως προτιμήθηκε η δημιουργία περισσότερων. Ως εκ τούτου, υπάρχει ένας χειριστής, ο οποίος διαχειρίζεται τα αιτήματα που κάνει ο διαχειριστής της εφαρμογής (adminController), ένας χειριστής που διαχειρίζεται αιτήματα που αφορούν την ταυτοποίηση του χρήστη (authController), ένας που διαχειρίζεται αιτήματα που αφορούν διάφορες λειτουργίες της εφαρμογής (operationsController), όπως η δημιουργία νέου φακέλου, η διαγραφή ενός αρχείου, κλπ. και τέλος υπάρχει ο prologController ο οποίος διαχειρίζεται αιτήματα που αφορούν εντολές Prolog, όπως παραδείγματος χάρη μια εντολή για το φόρτωμα ενός αρχείου στη βάση δεδομένων της ή η εισαγωγή ενός νέου κανόνα σε αυτή, κλπ.



Εικόνα 15: Επίπεδα και κλάσεις της διαδικτυακής υπηρεσίας

Όπως φαίνεται Εικόνα 15, όλοι οι χειριστές χρησιμοποιούν υπηρεσίες από την αποθήκη υπηρεσιών που έχει δημιουργηθεί. Το servicesRepo λοιπόν αποτελεί μια διεπαφή την οποία υλοποιεί η κλάση ServicesRepoImpl και εκεί βρίσκονται όλες οι υπηρεσίες που χρησιμοποιούν οι μέθοδοι του κομματιού της εφαρμογής που τρέχει στον εξυπηρετητή.

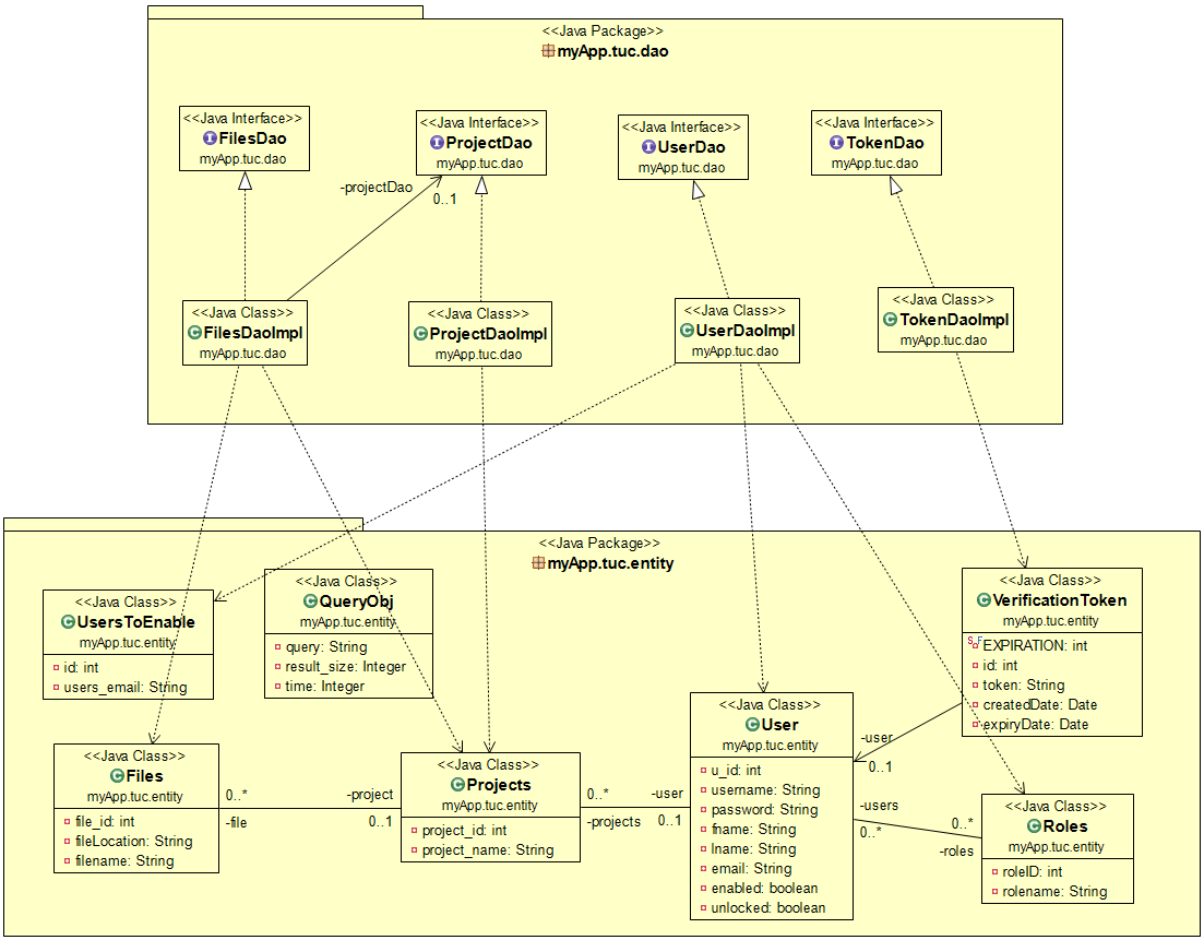
Το τρίτο πακέτο και τελευταίο που φαίνεται στην Εικόνα 15 ονομάζεται DAO και μέσα περιέχει όλες τις αναγκαίες και ικανές διεπαφές, αλλά και κλάσεις που τις υλοποιούν, ώστε να επικοινωνήσει η εφαρμογή με την βάση δεδομένων. Ως εκ τούτου η αλληλεπίδραση της αποθήκης υπηρεσιών με αυτές τις διεπαφές κρίνεται αναγκαία για την εκτέλεση λειτουργιών στη βάση δεδομένων της υπηρεσίας. Οι διεπαφές που υπάρχουν μέσα στο συγκεκριμένο πακέτο είναι τέσσερις. Αρχικά, η UserDao διεπαφή, η οποία περιέχει μεθόδους για αλληλεπίδραση με τη βάση δεδομένων σε ότι έχει να κάνει με τον χρήστη. Έπειτα, η διεπαφή ProjectsDao, μέσω της οποίας η εφαρμογή εκτελεί λειτουργίες, χρησιμοποιώντας τη βάση δεδομένων, που σχετίζονται με τους φακέλους του χρήστη. Στη συνέχεια, υπάρχει η διεπαφή FilesDao με την οποία γίνεται η αλληλεπίδραση με τη βάση για θέματα που αφορούν τα αρχεία του χρήστη. Τέλος, η διεπαφή TokenDao δημιουργήθηκε για την αποθήκευση και τον χειρισμό ενός κλειδάριθμου που δημιουργείται για κάθε νέο χρήστη που εγγράφεται.

Στην Εικόνα 15 απεικονίζονται τρία πακέτα με τις αντίστοιχες κλάσεις και διεπαφές που αποτελούν τα τρία επίπεδα του σχήματος , δηλαδή το επίπεδο του χειρισμού, το επίπεδο των υπηρεσιών και το επίπεδο της σύνδεσης με τη βάση δεδομένων. Στο σχήμα όμως νοείται και η βάση δεδομένων η οποία δεν φαίνεται στο σχήμα. Στην επόμενη ενότητα θα δειχθεί αναλυτικά πως το επίπεδο της σύνδεσης επικοινωνεί με την βάση δεδομένων, ποιες επιπλέον κλάσεις δημιουργήθηκαν καθώς και τι εργαλεία χρησιμοποιήθηκαν, για την σύνδεση αυτή. Στις μετέπειτα ενότητες θα ακολουθηθεί αντίθετη πορεία και θα δειχθεί πιο αναλυτικά το επίπεδο των υπηρεσιών και τέλος το επίπεδο των χειριστών της εφαρμογής.

3.3 Σύνδεση της υπηρεσίας με τη βάση δεδομένων της εφαρμογής

Στην Εικόνα 16, απεικονίζονται οι διεπαφές, οι κλάσεις που υλοποιούνται σε αυτές και πως συνδέονται με τις κλάσεις του πακέτου Entity της εφαρμογής. Φαίνεται λοιπόν πως η κλάση FilesDaoImpl υλοποιεί τις μεθόδους της αντίστοιχης διεπαφής για την εισαγωγή νέου αρχείου, την εξαγωγή της τοποθεσίας αυτού, την διαγραφή ενός αρχείου, καθώς και την επιστροφή όλων των αρχείων ενός φάκελου. Η κλάση ProjectDao υλοποιεί αντίστοιχες μεθόδους για τους φακέλους του χρήστη, όπως παραδείγματος χάρη η δημιουργία ενός νέου φάκελου ή η διαγραφή του. Στην διεπαφή UserDao υπάρχουν μέθοδοι που σχετίζονται με τον χρήστη, όπως η αποθήκευση χρήστη στη βάση δεδομένων, η διαγραφή του, η επιστροφή του ρόλου του (αν δηλαδή είναι απλός χρήστης ή διαχειριστής) κλπ. και υλοποιούνται από την κλάση UserDaoImpl. Τέλος, στην διεπαφή TokenDao υπάρχουν οι μέθοδοι για την αποθήκευση του κλειδάριθμου και την εύρεση αυτού που υλοποιούνται στην κλάση TokenDaoImpl.

Όπως γίνεται αντιληπτό και από την Εικόνα 16, το πακέτο Dao επικοινωνεί με το πακέτο Entity. Σε αυτό το πακέτο υπάρχουν όλες οι κλάσεις, η κάθε μία από τις οποίες αντιστοιχίζεται με έναν πίνακα στη βάση δεδομένων της εφαρμογής. Η κλάση User για παράδειγμα, αντιστοιχεί στον πίνακα user της βάσης δεδομένων και χρησιμοποιείται από την κλάση UserDaoImpl στην υλοποίηση των μεθόδων της. Όλοι οι πίνακες στη βάση δεδομένων έχουν την αντίστοιχη κλάση στην Java, δεν ισχύει όμως και το αντίθετο, αφού υπάρχει και μια κλάση στο πακέτο entity που δεν αντιστοιχεί σε κάποιο πίνακα στη βάση. Πρόκειται για την κλάση QueryObj και έχει δημιουργηθεί για τις ανάγκες χειρισμού εντολών της Prolog, όπως θα αναλυθεί στη συνέχεια.



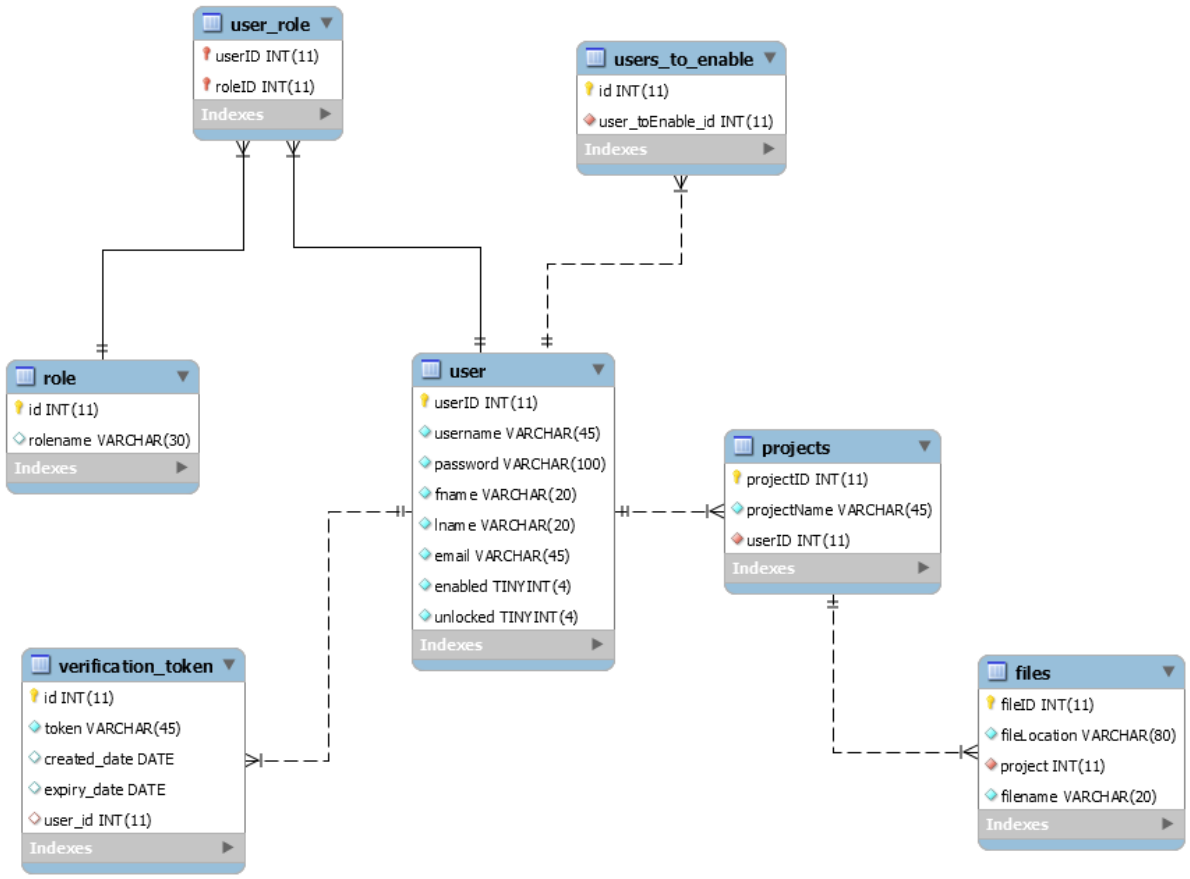
Εικόνα 16: Σύνδεση κλάσεων των πακέτων DAO και Entity

3.3.1 Η χρήση του Hibernate για την σύνδεση με την βάση δεδομένων

Η χρήση του Hibernate framework [11] επιλέχθηκε αντί ενός απλού JDBC (Java DataBase Connector) για την σύνδεση με την βάση δεδομένων της εφαρμογής λαμβάνοντας υπόψιν τα πλεονεκτήματα που έχει. Αρχικά, το Hibernate αυτοματοποιεί βασικές λειτουργίες, όπως η δημιουργία, η ανάγνωση, η αναβάθμιση και η διαγραφή (CRUD - Create Read Update Delete). Ως εκ τούτου, ο κώδικας είναι πιο εύκολο να διαβαστεί και οι γραμμές του σημαντικά λιγότερες, αφού παραδείγματος χάρη για την αποθήκευση ενός αντικειμένου σε έναν πίνακα της βάσης αρκεί να καλεστεί η μέθοδος save() ή για την επικαιροποίηση η μέθοδος update(). Αυτές και άλλες παρόμοιες μεθόδους χρησιμοποιούν και οι διεπαφές της εφαρμογής. Αυτό κάνει και την συντήρηση της εφαρμογής ευκολότερη. Ένα άλλο σημαντικό πλεονέκτημα είναι ότι δεν υπάρχει η ανάγκη ο προγραμματιστής να δημιουργήσει τη βάση δεδομένων, αφού το Hibernate παρέχει την δυνατότητα αυτόματης δημιουργίας της βάσης από τα δεδομένα που διαβάζει. Τέλος, το Hibernate παρέχει καλύτερη δομή στην εφαρμογή, εφόσον την ανεξαρτητοποιεί από την βάση δεδομένων. Ο προγραμματιστής μπορεί εύκολα να αλλάξει την βάση που χρησιμοποιεί, αλλάζοντας μόνο μερικές μεταβλητές στο αρχείο διαμόρφωσης.

3.3.2 Βάση δεδομένων της εφαρμογής

Η βάση δεδομένων που δημιουργήθηκε για την εφαρμογή φαίνεται στη παρακάτω εικόνα (βλ. Εικόνα 17). Χρησιμοποιείται το σύστημα διαχείρισης βάσεων MySQL και όπως γίνεται αντιληπτό από την εικόνα της βάσης, ο κύριος πίνακας είναι αυτός του χρήστη, δηλαδή ο πίνακας user. Υπάρχει ένας πίνακας role για τον ρόλο του χρήστη, καθώς και ο user_role, ο οποίος δημιουργήθηκε για την σύνδεση "πολλά με πολλά" με τον πίνακα user. Ο κάθε χρήστης μπορεί να έχει πολλούς φακέλους και κάθε φάκελος μπορεί να περιέχει πολλά αρχεία, για αυτό δημιουργήθηκαν και οι αντίστοιχοι πίνακες projects και files. Τέλος, ο πίνακας verification_token κρατάει πληροφορίες για τον κλειδάριθμο που αποκτά ο χρήστης κατά την εγγραφή, του ώστε να ενεργοποιήσει τον λογαριασμό του, ενώ ο πίνακας users_to_enable κρατάει όλους τους χρήστες που είναι ενεργοποιημένοι και περιμένουν να πάρουν έγκριση από τον διαχειριστή, ώστε να αρχίσουν να χρησιμοποιούν την υπηρεσία.



Εικόνα 17: Οι πίνακες της βάσης δεδομένων

3.4 Το επίπεδο υπηρεσιών

Το επίπεδο των υπηρεσιών αποτελεί ένα από τα πιο σημαντικά κομμάτια της εφαρμογής, καθώς επικοινωνεί με τον χειριστή, αλλά και με τις DAO κλάσεις. Ως εκ τούτου, ό,τι υπηρεσία χρειαστεί ο εκάστοτε Controller μέσα στις συναρτήσεις που υλοποιεί, απευθύνεται σε αυτό το επίπεδο όπου έχει δημιουργηθεί μια αποθήκη υπηρεσιών. Με την σειρά του, το επίπεδο υπηρεσιών επικοινωνεί με τις DAO διεπαφές, όταν απαιτείται αλληλεπίδραση με την βάση δεδομένων, με την μηχανή Prolog, όταν απαιτείται αλληλεπίδραση με την Prolog, ή σε διαφορετική περίπτωση προσφέρει απευθείας τις υπηρεσίες που του ζητήθηκαν. Ήταν δυνατό να μην δημιουργηθεί το πακέτο DAO και η επικοινωνία με την βάση δεδομένων να γίνεται απευθείας από το επίπεδο υπηρεσιών. Για μεγάλες και πολύπλοκες εφαρμογές όμως αυτό δεν αποτελεί καλή τακτική, αφενός για λόγους δομής και αφετέρου για λόγους επεκτασιμότητας.

Το πακέτο των υπηρεσιών αποτελείται από μια διεπαφή, τις μεθόδους της οποίας υλοποιεί η αντίστοιχη κλάση. Οι μέθοδοι που υλοποιεί αυτή η κλάση χωρίζονται σε τρεις κατηγορίες. Αυτές που αλληλεπιδρούν με τη βάση και συνεπώς προωθούνται στην κατάλληλη κλάση του πακέτου DAO, εκείνες που αλληλεπιδρούν με τη Prolog και εκείνες που εκτελούνται στην κλάση και επιστρέφουν τα αποτελέσματα.

Αυτές οι μέθοδοι και οι συναρτήσεις που τις υλοποιούν αποτελούν τον πυρήνα και το βασικότερο κομμάτι για την λειτουργία της υπηρεσίας. Καλούνται από τις συναρτήσεις των κλάσεων του πακέτου Controllers και στις επόμενες ενότητες, που θα παρουσιαστούν οι σημαντικότερες λειτουργίες της υπηρεσίας, θα γίνουν αναφορές σχεδόν σε όλες αυτές τις μεθόδους, εφόσον είναι τα γρανάζια για να δουλέψει όλο το σύνολο της διαδικτυακής υπηρεσίας.

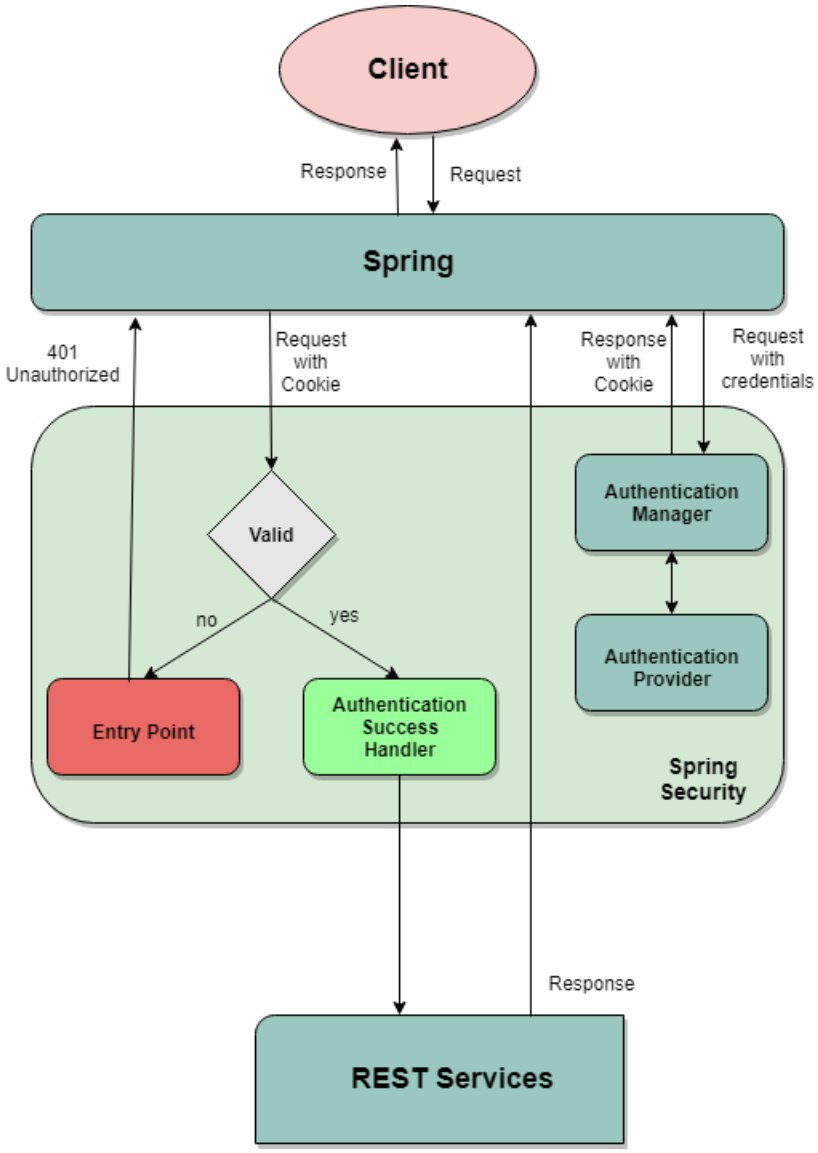
Στο πακέτο των υπηρεσιών υπάρχει άλλη μια κλάση με το όνομα MyUserDetailsService. Αυτή η κλάση δημιουργήθηκε στα πλαίσια της ασφάλειας της εφαρμογής και πριν αναλυθεί το τρίτο και τελευταίο κομμάτι της υπηρεσίας (Controllers), θα αφιερωθεί μια ενότητα για την ανάλυση της ασφάλειας της υπηρεσίας και ποια εργαλεία χρησιμοποιήθηκαν για την υλοποίησή της.

3.5 Πρωτόκολλο ασφάλειας της υπηρεσίας

Ίσως το πιο σημαντικό στοιχείο μιας εφαρμογής είναι η ασφάλεια της. Στην συγκεκριμένη εφαρμογή, για την κατανάλωση των υπηρεσιών της, απαραίτητη προϋπόθεση είναι η εγγραφή σε αυτήν και κατά επέκταση η είσοδος του χρήστη στην εφαρμογή κάθε φορά που επιθυμεί να την χρησιμοποιήσει. Για την υλοποίηση της παραπάνω λειτουργίας και την γενικότερη ασφάλεια της υπηρεσίας στον εξυπηρετητή, δημιουργήθηκε και ακολουθήθηκε ένα πρωτόκολλο ασφαλείας με την βοήθεια του Spring Security [10] [19]. Το Spring Security είναι ένα γενικό πλαίσιο, το οποίο παρέχει την δυνατότητα επεκτασιμότητας του, ώστε να προσαρμοστεί στις ανάγκες της εφαρμογής σε θέματα ταυτοποίησης και παροχής πρόσβασης στον χρήστη. Ουσιαστικά, αποτελεί ένα φίλτρο πριν την παραλαβή κάθε αιτήματος από την υπηρεσία που είτε προωθεί το αίτημα στην υπηρεσία, είτε το επιστρέφει πίσω σε περίπτωση που δεν ταυτοποιηθεί ο χρήστης ή δεν έχει τον κατάλληλο ρόλο για την υπηρεσία που αιτήθηκε. Στην Εικόνα 18 απεικονίζεται η ροή, αλλά και ο ρόλος που κατέχει το Spring Security στην εφαρμογή.

Όταν ο χρήστης αιτηθεί για πρώτη φορά μια υπηρεσία της εφαρμογής, απαραίτητη προϋπόθεση είναι στην επικεφαλίδα του αιτήματος να συμπεριλάβει τα στοιχεία του, δηλαδή το ψευδώνυμό του και τον κωδικό πρόσβασης. Το Spring Security λειτουργεί σαν ένα φίλτρο μέσα στο Spring και πριν κατευθυνθεί το αίτημα στον κατάλληλο χειριστή, περνάει από αυτό το φίλτρο, ώστε να ταυτοποιηθεί ο χρήστης. Αν τα στοιχεία που έδωσε είναι ορθά και είναι ένας ενεργός χρήστης, αλλά επίσης έχει και ρόλο κατάλληλο για την υπηρεσία που ζητάει, τότε περνάει με επιτυχία από το φίλτρο του Spring Security και προωθείται στον αντίστοιχο Controller. Με το που αποκτήσει πρόσβαση ο χρήστης στην υπηρεσία, δημιουργείται μια νέα συνεδρία (Session). Αν ο χρήστης δεν χρησιμοποιήσει την υπηρεσία

και δεν κάνει αποσύνδεση κατά την έξοδό του από αυτή, το Session που δημιουργήθηκε μένει ενεργό για τα επόμενα 30 λεπτά και έπειτα πραγματοποιείται αυτόματη αποσύνδεση. Επίσης, με την αρχική πρόσβαση του χρήστη, η υπηρεσία δημιουργεί και επιστρέφει ένα Cookie, το οποίο μπορεί να στέλνει ο χρήστης στα επόμενα αιτήματά του για να αποκτά πρόσβαση σε αυτή, αντί να στέλνει τα στοιχεία του μαζί με κάθε αίτημα. Ως εκ τούτου, η υπηρεσία δεν μπορεί να χαρακτηριστεί Stateless, εφόσον για την χρησιμοποίησή της χρειάζεται login. Γίνεται αντιληπτό πως, σε μια τέτοια περίπτωση, είναι αναγκαίο η υπηρεσία να αποθηκεύει στην κρυφή μνήμη τα στοιχεία του χρήστη, τα ενεργά Sessions, κλπ. για την αρμονική λειτουργία της. Και αυτό γιατί όσο ο χρήστης χρησιμοποιεί την υπηρεσία και αλληλεπιδρά με την Prolog μηχανή, προσθέτει είτε αρχεία είτε κανόνες στην βάση δεδομένων της Prolog, κάτι που σημαίνει ότι αφενός η εφαρμογή και κατά επέκταση η Prolog πρέπει να έχουν μνήμη και αφετέρου ότι όλα όσα πρόσθεσε στην βάση της Prolog πρέπει να καθαριστούν με την αποσύνδεσή του. Το Spring Security παρέχει την δυνατότητα να εκτελεστεί επιχειρησιακή λογική (business logic), μόλις ο χρήστης στείλει αίτημα για αποσύνδεση και αυτή ακριβώς η δυνατότητα αξιοποιείται ώστε να καθαριστεί η βάση της Prolog.



Εικόνα 18: Διαδικασία ταυτοποίησης χρήστη

Όπως αναφέρθηκε και στην προηγούμενη ενότητα, στο επίπεδο των υπηρεσιών της εφαρμογής έχει δημιουργηθεί και μια κλάση ονόματι MyUserDetailsService. Αυτή η κλάση δημιουργήθηκε στο πλαίσιο της ασφάλειας της εφαρμογής και σκοπός της είναι να φορτώσει τον χρήστη, τα στοιχεία του οποίου δίνονται στην προσπάθεια εισόδου του στην εφαρμογή. Μαζί με την ταυτοποίηση των στοιχείων του, ελέγχει και αν είναι ενεργός, αν έχει ξεκλειδωθεί από τον διαχειριστή και αν ο ρόλος τον οποίο έχει επιτρέπει την πρόσβαση στην υπηρεσία που αιτείται. Αν και μόνο αν ισχύουν όλα τα παραπάνω τότε του επιτρέπει την πρόσβαση, αλλιώς επιστρέφει σφάλμα με κωδικό 401.

3.6 Το επίπεδο χειριστών

Ακολουθώντας αντίθετη φορά, και αφού σε προηγούμενες ενότητες αναλύθηκε το επίπεδο σύνδεσης της εφαρμογής με τη βάση, αλλά και το επίπεδο των υπηρεσιών της εφαρμογής, σε αυτή την ενότητα θα αναλυθεί το αρχικό επίπεδο της υπηρεσίας, μόλις ληφθεί ένα αίτημα, αλλά ταυτόχρονα και το τελευταίο, πριν αποσταλεί η απάντηση από την υπηρεσία.

Ο χειριστής (Controller) αποτελεί ένα από τα τρία κύρια κομμάτια της δομής της υπηρεσίας και είναι στην ουσία ο εγκέφαλός της. Απαρτίζεται από μεθόδους οι οποίες αντιστοιχούν σε συγκεκριμένα URIs μέσα από τα οποία ο χρήστης αποκτά πρόσβαση στις υπηρεσίες της εφαρμογής και την καταναλώνει. Εφόσον πρόκειται για μια REST υπηρεσία δεν επιστρέφει απευθείας κάποια σελίδα. Αντίθετα επιστρέφει αποτελέσματα σε μορφή JSON τα οποία χρησιμοποιούνται κατάλληλα από άλλα προγράμματα. Ένα τέτοιο πρόγραμμα αποτελεί και η διεπαφή χρήστη που δημιουργήθηκε, μέσω της οποίας γίνεται πιο εύχρηστη η κατανάλωση της υπηρεσίας. Ανάλογα με το αίτημα που θα λάβει ο χειριστής καλεί την κατάλληλη μέθοδο η οποία με την σειρά της χρησιμοποιεί τις υπηρεσίες της εφαρμογής, ώστε να επιστρέψει αποτελέσματα ή να εκτελέσει λειτουργίες. Πιο συγκεκριμένα, στο πακέτο Controller έχουν δημιουργηθεί συνολικά τέσσερις χειριστές. Ένας με τα endpoints[21] που αφορούν αιτήματα προς την Prolog, ένας που εξυπηρετεί αιτήματα ασφάλειας και ταυτοποίησης, ένας για τα αιτήματα διαχείρισης αρχείων και φακέλων και τέλος ένας για τις λειτουργίες του διαχειριστή. Οι λειτουργίες που εκτελεί η υπηρεσία και οι υπηρεσίες που προσφέρει γενικά στον χρήστη η εφαρμογή είναι αρκετές. Στις παρακάτω ενότητες θα αναλυθούν οι σημαντικότερες εξ αυτών.

3.6.1 Πρωτόκολλο εγγραφής νέου χρήστη

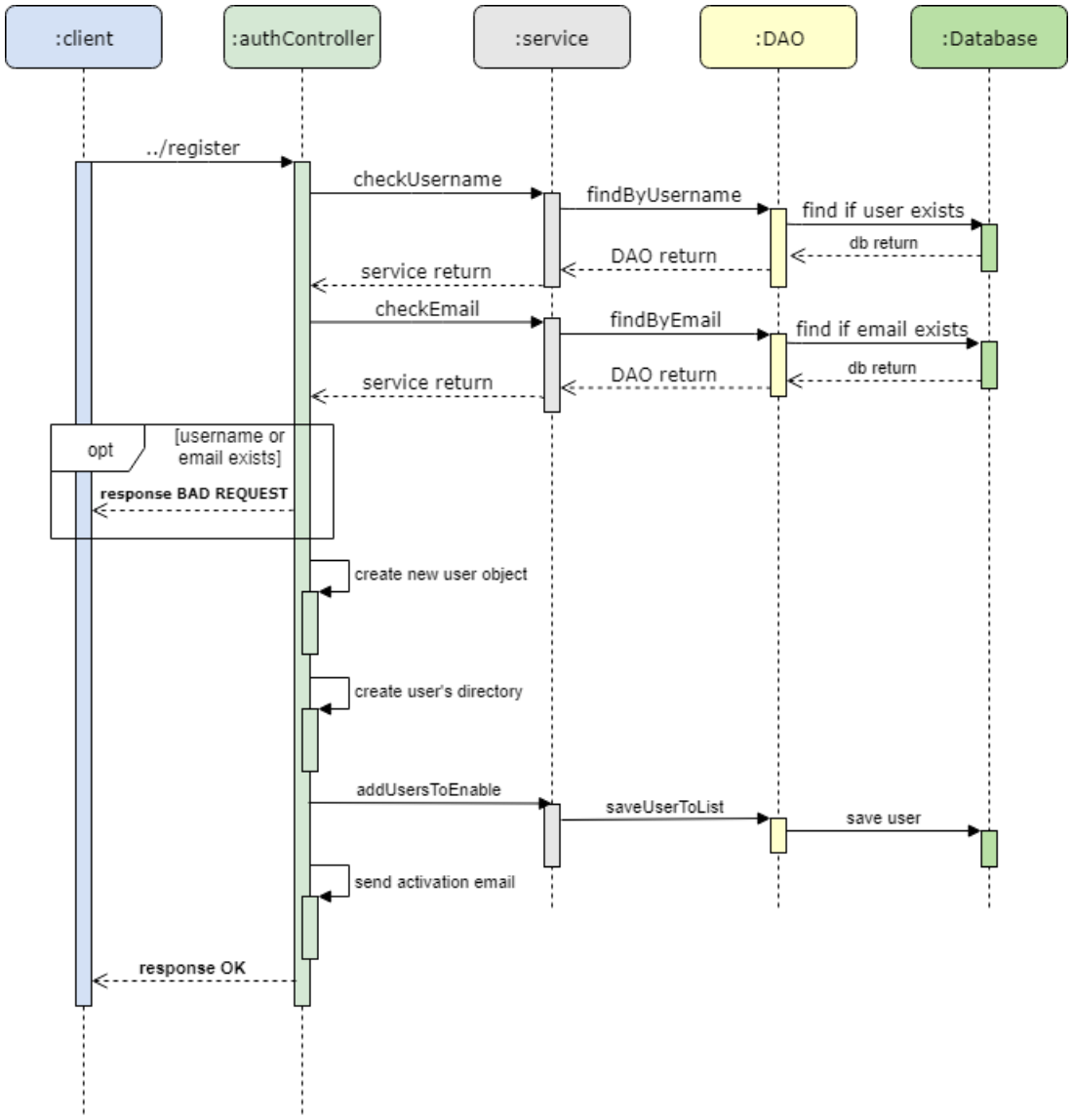
Για να μπορέσει ένας χρήστης να χρησιμοποιήσει την υπηρεσία είτε από το γραφικό περιβάλλον είτε μέσα σε ένα πρόγραμμα, απαραίτητη προϋπόθεση είναι η εγγραφή του στην υπηρεσία. Για λόγους ευχρηστίας καλό είναι να εγγραφεί μέσω της διεπαφής χρήστη της εφαρμογής που παρέχεται. Στην συγκεκριμένη ενότητα θα αναλυθεί η λειτουργία εγγραφής νέου χρήστη στην υπηρεσία, καθώς και η γενικότερη διαδικασία της εγγραφής χρήστη σε αυτή.

Συμπληρώνοντας τα στοιχεία της φόρμας στην διεπαφή χρήστη και πατώντας το κουμπί για εγγραφή, ο χρήστης δεν κάνει τίποτα άλλο από το να καλέσει μια υπηρεσία της εφαρμογής, ένα URI δηλαδή, χρησιμοποιώντας μέθοδο POST και στέλνοντας μαζί τα στοιχεία του. Οι έλεγχοι για την ασφάλεια και την εγκυρότητα των στοιχείων που αποστέλλονται θα αναλυθούν στο επόμενο κεφάλαιο που θα παρουσιασθεί το μέρος που αλληλοεπιδρά με τον χρήστη. Όταν η υπηρεσία στον εξυπηρετητή

[21] Endpoint: Το σημείο επαφής μεταξύ δύο συστημάτων

λαμβάνει ένα νέο αίτημα εγγραφής, αρχικά ελέγχει με την σειρά της για τυχόν λάθη, όπως διπλές εγγραφές ή κενά πεδία. Σε μια τέτοια περίπτωση αποκρίνεται με σφάλμα το οποίο χειρίζεται κατάλληλα η διεπαφή χρήστη.

Αν δεν υπάρχουν τέτοια σφάλματα, το πρώτο που κάνει είναι να δημιουργήσει έναν νέο φάκελο στον γενικό φάκελο της εφαρμογής που υπάρχει στον εξυπηρετητή με το username του νέου χρήστη. Εκεί μέσα θα αποθηκεύονται όλοι οι φάκελοι και τα αρχεία του συγκεκριμένου χρήστη. Εν συνεχεία, θέτει τον χρήστη μη ενεργό και μπλοκαρισμένο στα αντίστοιχα πεδία του πίνακα στη βάση δεδομένων, κωδικοποιεί τον κωδικό πρόσβασης του για λόγους ασφαλείας και τον αποθηκεύει στη βάση δεδομένων. Έπειτα αποστέλλει ένα email στον χρήστη μέσα από το οποίο, επιλέγοντας τον σύνδεσμο που παρέχεται, ενεργοποιεί τον λογαριασμό του. Στην Εικόνα 19, απεικονίζεται το διάγραμμα ενεργειών όπως παρουσιάστηκαν παραπάνω.



Εικόνα 19: Διάγραμμα ενεργειών εγγραφής νέου χρήστη

Για τις ανάγκες της ενεργοποίησης του λογαριασμού του χρήστη, σε κάθε νέα εγγραφή και πιο συγκεκριμένα κατά την διαδικασία αποστολής του email ενεργοποίησης του λογαριασμού, δημιουργείται ένας κλειδάριθμος, ο οποίος αποθηκεύεται στη βάση δεδομένων στον αντίστοιχο πίνακα και αποστέλλεται στο email ενεργοποίησης. Ο κλειδάριθμος αυτός παραμένει ενεργός για μια ημέρα και με το πέρας αυτής ο χρήστης δεν μπορεί να ενεργοποιήσει τον λογαριασμό του. Σε μια τέτοια περίπτωση διαγράφεται ο φάκελος που έχει δημιουργηθεί με το όνομά του, καθώς επίσης και ο ίδιος ο χρήστης από τη βάση δεδομένων και αν θέλει να χρησιμοποιήσει την υπηρεσία πρέπει να κάνει εκ νέου εγγραφή. Αν ο κλειδάριθμος είναι ενεργός και ο χρήστης επιλέξει το αντίστοιχο πεδίο στο email που του έχει αποσταλεί, τότε ο λογαριασμός του ενεργοποιείται. Όταν συμβεί αυτό, το αντίστοιχο πεδίο στη βάση δεδομένων ενημερώνεται και ο χρήστης είναι πλέον ενεργός. Δεν αρκεί όμως αυτό για να μπορέσει να έχει πρόσβαση στην υπηρεσία, αφού αναγκαία συνθήκη είναι να τον εγκρίνει και ο διαχειριστής της εφαρμογής. Αυτό αποτελεί μια επιπλέον λειτουργία για περισσότερη ασφάλεια και καλύτερο έλεγχο της υπηρεσίας.

Όταν λοιπόν ο χρήστης ενεργοποιήσει τον λογαριασμό του, αυτομάτως μπαίνει σε μια λίστα με όλους εκείνους τους χρήστες που περιμένουν την έγκριση τους από τον διαχειριστή. Για τις ανάγκες αυτής της λειτουργίας έχει δημιουργηθεί και ο αντίστοιχος πίνακας στη βάση δεδομένων, όπως δείχθηκε σε προηγούμενη ενότητα. Είναι πλέον αποκλειστική ευθύνη του διαχειριστή της υπηρεσίας να ξεκλειδώσει τον χρήστη και αυτό είναι το τελευταίο βήμα για να μπορέσει ο χρήστης να χρησιμοποιήσει την εφαρμογή. Όταν τον ξεκλειδώσει ο διαχειριστής, τότε θα έχει όλα τα πεδία που ελέγχει η ασφάλεια της υπηρεσίας ενεργά και θα μπορεί δίνοντας τα σωστά στοιχεία να εισέρχεται στην εφαρμογή. Η διαδικασία που περιγράφηκε αποτελεί το πρωτόκολλο εγγραφής νέου χρήστη στην εφαρμογή.

3.6.2 Σύστημα διαχείρησης αρχείων

Μια από τις λειτουργίες που προσφέρει η εφαρμογή και ίσως το στοιχείο εκείνο που είναι ικανό να την χαρακτηρίσει ως μια εφαρμογή υπολογιστικού νέφους, είναι η δυνατότητα ο χρήστης να δημιουργήσει τους δικούς του φακέλους, να ανεβάσει τα δικά του αρχεία εκεί και να τα αποθηκεύσει στον εξυπηρετητή. Έτσι αφενός του παρέχεται η πρόσβαση σε αυτά από παντού, και όχι μόνο τοπικά, και αφετέρου δεν χρειάζεται να ανεβάζει τα αρχεία που θα χρησιμοποιήσει κάθε φορά που κάνει χρήση της εφαρμογής, αντιθέτως αρκεί μόνο να τα φορτώσει κάνοντας χρήση της κατάλληλης Prolog εντολής. Η εφαρμογή παρέχει ένα πλήρες σύστημα διαχείρισης φακέλων και αρχείων και υποστηρίζει λειτουργίες δημιουργίας φακέλων, ανεβάσματος αρχείων αλλά και διαγραφής αυτών. Στις παρακάτω ενότητες θα δειχθούν αναλυτικότερα οι λειτουργίες αυτές.

3.6.3 Δημιουργία νέου φάκελου

Για να μπορέσει ένας χρήστης να ανεβάσει ένα αρχείο, απαραίτητη προϋπόθεση αποτελεί η ύπαρξη ενός φακέλου στον οποίο θα το ανεβάσει. Η συγκεκριμένη λειτουργία υλοποιήθηκε κατά αυτό τον τρόπο με το σκεπτικό της καλύτερης δομής και οργάνωσης του συστήματος φακέλων και αρχείων του χρήστη. Αντί να υπάρχει ένας και μόνο φάκελος όπου θα περιέχει όλα τα αρχεία που ανεβάζει ο χρήστης, παρέχεται η δυνατότητα δημιουργίας φακέλων, όπου κάθε φάκελος περιέχει τα δικά του αρχεία.

Για την δημιουργία ενός νέου φακέλου, ο χρήστης αρκεί να αποστείλει ένα αίτημα στο αντίστοιχο endpoint με μέθοδο POST και παράμετρο το όνομα του φακέλου που επιθυμεί. Βέβαια βασικό στοιχείο για την επιτυχή αποστολή του αιτήματος αποτελεί και η αποστολή είτε των στοιχείων

ταυτοποίησης του χρήστη, είτε του Cookie[22] που του έχει δοθεί στην επικεφαλίδα του αιτήματος, όπως αναλύθηκε στην αντίστοιχη ενότητα για την ασφάλεια της υπηρεσίας.

Στη συνέχεια καλείται η αντίστοιχη μέθοδος στον Controller, η οποία εξάγει τον χρήστη και το όνομα του φακέλου που πρέπει να δημιουργηθεί και καλεί την αντίστοιχη μέθοδο από το επίπεδο των υπηρεσιών. Εκείνη με την σειρά της πραγματοποιεί όλες τις απαραίτητες ενέργειες για την δημιουργία του φακέλου αλλά και την αποθήκευσή του στον αντίστοιχο πίνακα στη βάση δεδομένων. Η ροή είναι η ίδια, όπως αναλύθηκε σε προηγούμενη ενότητα, και αν η δημιουργία του φακέλου είναι επιτυχής, η υπηρεσία αποκρίνεται με HTTP status επιτυχίας, ενώ σε αντίθετη περίπτωση με status σφάλματος. Πρέπει να επισημανθεί πως στη βάση δεδομένων δεν αποθηκεύεται ο φάκελος αυτός καθ' εαυτός αλλά η τοποθεσία του φακέλου. Ο φάκελος δημιουργείται στον φάκελο που υπάρχει στον εξυπηρετητή για τις ανάγκες της εφαρμογής και μέσα στον αντίστοιχο φάκελο του χρήστη.

3.6.4 Προσθήκη αρχείου

Η διαδικασία ανεβάσματος ενός αρχείου, μαζί με την αυτή της εγγραφής νέου χρήστη, αποτελούν τις πιο περίπλοκες λειτουργίες της υπηρεσίας. Και αυτό προκύπτει γιατί για να δουλέψει η λειτουργία του ανεβάσματος ενός αρχείου αρμονικά, χρειάστηκαν να πραγματοποιηθούν έλεγχοι, να γίνουν οι κατάλληλες ενέργειες και να δημιουργηθούν αντίστοιχες υπηρεσίες, που θα αναλυθούν στην ενότητα αυτή.

Ο χρήστης έχει την δυνατότητα να ανεβάσει ένα Prolog αρχείο στέλνοντας ένα POST αίτημα στο αντίστοιχο endpoint. Οι παράμετροι που πρέπει να στείλει στο σώμα του αιτήματος είναι δύο, το αρχείο που επιθυμεί να ανεβάσει και το όνομα του φακέλου στο οποίο θέλει να το ανεβάσει. Η συνάρτηση που δημιουργήθηκε για αυτή την λειτουργία παίρνει τρεις παραμέτρους και την τρίτη την εξάγει από τα στοιχεία που στέλνει ο χρήστης μαζί με το αίτημα και δεν είναι άλλη από το ψευδώνυμο του χρήστη. Έχοντας πλέον το ψευδώνυμο, το όνομα του αρχείου, αλλά και τον φάκελο στο οποίο επιθυμεί ο χρήστης να ανεβάσει το αρχείο, καλεί την αντίστοιχη υπηρεσία για να ανεβάσει το αρχείο και περνάει ως παραμέτρους αυτά τα στοιχεία. Η υπηρεσία αυτή με την σειρά της καλεί την αντίστοιχη υπηρεσία που είναι υπεύθυνη για την επικοινωνία με την βάση δεδομένων της εφαρμογής και την εισαγωγή νέου αρχείου σε αυτή.

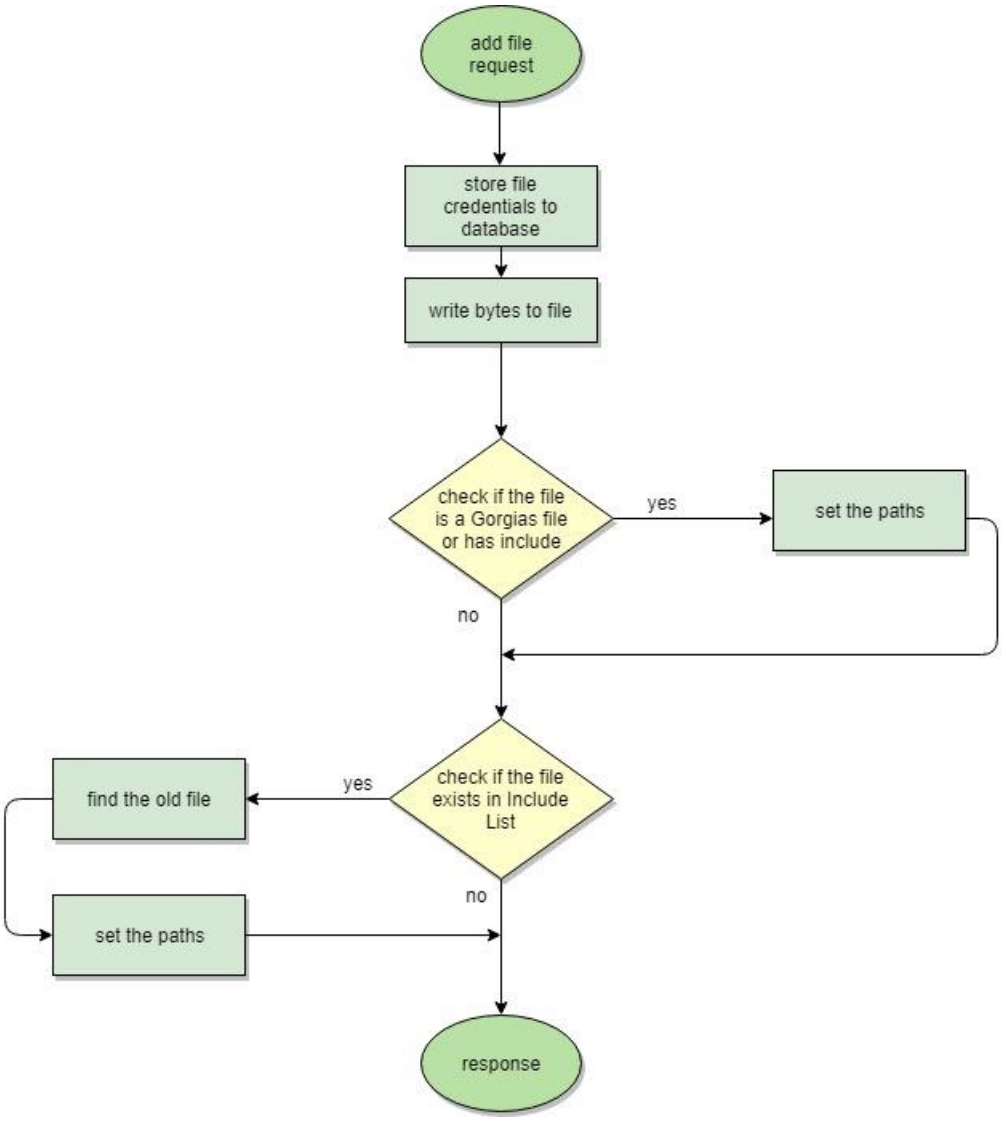
Εφόσον κληθεί η υπηρεσία και αποθηκευτούν οι πληροφορίες του αρχείου στη βάση δεδομένων όπως δείχθηκε, η αρχική συνάρτηση παίρνει τα δεδομένα του αρχείου και τα αποθηκεύει σε μια μεταβλητή τύπου byte[], αλλά και θέτει το κατάλληλο μονοπάτι του αρχείου σε μια μεταβλητή τύπου Path. Έπειτα γράφει τα δεδομένα του αρχείου στο μονοπάτι αυτό. Στην ουσία πρόκειται για έναν φάκελο, ο οποίος υπάρχει στον εξυπηρετητή και εκεί μέσα αποθηκεύονται όλα τα αρχεία των χρηστών. Όπως δείχθηκε και σε προηγούμενη ενότητα, με την εγγραφή κάθε νέου χρήστη δημιουργείται και ένας υποφάκελος μέσα σε αυτόν τον φάκελο με ονομασία το ψευδώνυμο του χρήστη. Κάθε φάκελος που δημιουργεί ο εκάστοτε χρήστης είναι ένας υποφάκελος στον φάκελό του και κάπως έτσι δημιουργείται το μονοπάτι του κάθε αρχείου. Ακόμη όμως δεν έχει εξασφαλιστεί η σωστή λειτουργία του ανεβάσματος ενός αρχείου και αυτό συμβαίνει γιατί πρέπει να πραγματοποιηθούν δύο ακόμα έλεγχοι. Ο πρώτος έλεγχος δεν είναι άλλος από το αν το αρχείο είναι ένα αρχείο Gorgias. Αν όντως πρόκειται για ένα τέτοιο αρχείο, τότε τα δύο μονοπάτια που κάθε αρχείο Gorgias έχει για τον εντοπισμό των αναγκαίων βιβλιοθηκών για την εκτέλεσή του, θα πρέπει να προσαρμοστούν κατάλληλα.

Ο δεύτερος έλεγχος αφορά το κατά πόσο ένα αρχείο φορτώνει στο εσωτερικό του άλλο ή άλλα αρχεία, αν δηλαδή περιέχει εντολή ή εντολές include. Οι δυο αυτοί έλεγχοι πραγματοποιούνται

[22] Cookie: Μικρά αρχεία κειμένου που αποθηκεύονται στον περιηγητή ιστού κατά την πλοήγηση στο διαδίκτυο και περιέχουν πληροφορία

αμέσως μετά την εγγραφή του αρχείου στον κατάλληλο φάκελο. Σύμφωνα και με το διάγραμμα ροής της Εικόνας 20, η αρχική συνάρτηση και με το πέρας της εγγραφής του αρχείου, ελέγχει αν το αρχείο είναι αρχείο Gorgias, αλλά και αν περιλαμβάνει εντολή include στο εσωτερικό του, καλώντας την αντίστοιχη μέθοδο της διεπαφής υπηρεσιών και περνώντας ως ορίσματα το μονοπάτι του αρχείου και το ψευδώνυμο του χρήστη. Σε μια τέτοια περίπτωση καλείται η αντίστοιχη υπηρεσία για την ενημέρωση των μονοπατιών.

Σε περίπτωση που το αρχείο που περιέχεται σε εντολή include δεν έχει ανέβει ακόμα, η ονομασία του αρχείου προστίθεται σε μια λίστα. Σε κάθε νέο ανέβασμα αρχείου του χρήστη, ελέγχεται αν το αρχείο βρίσκεται σε αυτή την λίστα. Σε περίπτωση που βρίσκεται, ενημερώνεται το μονοπάτι του παλιότερου αρχείου, ώστε η εντολή include να περιέχει πλέον το σωστό μονοπάτι και να φορτωθεί το αρχείο. Κάτι τέτοιο δίνει την δυνατότητα στον χρήστη να ανεβάζει αρχεία με ακαθόριστη σειρά. Στην εικόνα που ακολουθεί παρουσιάζεται το διάγραμμα ροής με βάση όσα ειπώθηκαν παραπάνω.



Εικόνα 20: Διάγραμμα ροής για την προσθήκη νέου αρχείου

3.6.5 Διαγραφή φάκελου ή αρχείου

Μία ακόμη λειτουργία που βοηθάει τον χρήστη στην διαχείριση των αρχείων, είναι αυτή της διαγραφής. Ο χρήστης έχει την δυνατότητα στέλνοντας το κατάλληλο αίτημα προς τον εξυπηρετητή, να διαγράψει ένα αρχείο ή και ολόκληρο τον φάκελο. Αρχικά, διαγράφονται τα δεδομένα που υπάρχουν στη βάση δεδομένων και στην συνέχεια τα φυσικά αρχεία που υπάρχουν στο φάκελο του χρήστη.

3.6.6 Λειτουργίες του διαχειριστή

Ο διαχειριστής της εφαρμογής διαθέτει κάποιες επιπλέον λειτουργίες. Η πιο σημαντική είναι πως κάθε χρήστης πρέπει να εγκριθεί από τον διαχειριστή για να μπορέσει να χρησιμοποιήσει την εφαρμογή. Μετά την επιτυχημένη εγγραφή του στην υπηρεσία, εισέρχεται σε έναν πίνακα στην βάση δεδομένων της εφαρμογής, όπου τηρούνται όλοι οι χρήστες που περιμένουν για έγκριση. Με ένα αίτημα στην κατάλληλη υπηρεσία, ο διαχειριστής έχει την δυνατότητα να δει την λίστα με τους χρήστες που περιμένουν να εγκριθούν, αλλά και να εγκρίνει όποιους επιθυμεί. Επίσης μπορεί να διαγράψει έναν χρήστη από την υπηρεσία ή απλά να τον απενεργοποιήσει. Για τις λειτουργίες του διαχειριστή έχει δημιουργηθεί ξεχωριστή κλάση και ξεχωριστός Controller, όπου εκεί μέσα υπάρχουν όλοι οι μέθοδοι που αφορούν τον διαχειριστή της εφαρμογής.

3.6.7 Λειτουργίες Prolog μέσω της υπηρεσίας

Η κύρια ιδέα γύρω από την οποία δημιουργήθηκε η συγκεκριμένη διαδικτυακή υπηρεσία είναι η δυνατότητα χρήσης της Prolog μηχανής και κατά επέκταση του Gorgias, χωρίς να είναι απαραίτητη η εγκατάστασή τους στον υπολογιστή. Συνεπώς, γίνεται αντιληπτό πως η σύνδεση μεταξύ Java και Prolog καθίσταται αναγκαία και στην υπηρεσία αυτό επετεύχθη χάρη στην διεπαφή JPL.

Η JPL είναι μια διεπαφή η οποία ενώνει την Java με την Prolog δίνοντας την δυνατότητα για χρησιμοποίηση της Prolog μέσα σε ένα Java πρόγραμμα. Συνεπώς, μέσα στο πακέτο Controllers της υπηρεσίας υπάρχει ξεχωριστή κλάση ονόματι prologController, η οποία εξυπηρετεί όλα τα αιτήματα που αφορούν την Prolog. Στις υπηρεσίες που καλούνται στις μεθόδους αυτής της κλάσης χρησιμοποιείται κατά κόρον η διεπαφή JPL.

Αρχικά, ο χρήστης στέλνει ένα αίτημα προς την υπηρεσία με POST μέθοδο στο URI '../prolog'. Στο σώμα του αιτήματος έχει προσθέσει ένα αντικείμενο, το Query. Το αντικείμενο αυτό δεν είναι τίποτε άλλο από μια κλάση της Java, η οποία περιέχει την εντολή που δίνει ο χρήστης προς την Prolog μηχανή, τον αριθμό των αποτελεσμάτων που επιθυμεί να του επιστραφούν και τον χρόνο που επιθυμεί να τρέξει η μηχανή Prolog.

Στην αρχή θα πραγματοποιηθεί μια εξαγωγή δεδομένων, τα οποία θα χρησιμοποιηθούν στη συνέχεια. Θα αποθηκευτούν σε μεταβλητές η εντολή, ο αριθμός αποτελεσμάτων, ο χρόνος, αλλά και το username του χρήστη που έστειλε το αίτημα.

Στην συνέχεια θα πραγματοποιηθεί ο πρώτος έλεγχος. Αν η εντολή που έδωσε ο χρήστης είναι η halt ή η listing, τότε δεν εκτελείται και επιστρέφεται μήνυμα πως οι συγκεκριμένες εντολές δεν υποστηρίζονται από την υπηρεσία. Όταν η Prolog τρέχει τοπικά, η εντολή halt χρησιμοποιείται για να κλείσει το γραφικό περιβάλλον της. Στην περίπτωση που χρησιμοποιείται μέσω Java και συγκεκριμένα μέσω της JPL διεπαφής, τερματίζει το πρόγραμμα της Java. Γίνεται αντιληπτό λοιπόν πως για λόγους ασφαλείας αυτή η εντολή δεν εκτελείται και δεν υποστηρίζεται από την υπηρεσία διαδικτύου. Για αντίστοιχους λόγους ασφαλείας δεν εκτελείται και η εντολή listing, αφού η εκτέλεσή της επιστρέφει όλα όσα περιέχει η βάση δεδομένων της μηχανής Prolog.

Έπειτα από τον πρώτο έλεγχο καλείται μια από τις υπηρεσίες της εφαρμογής, η οποία επιστρέφει έναν αριθμό ανάλογα με το ποια εντολή έχει δοθεί από τον χρήστη προς την Prolog. Ανάλογα με την περίπτωση εκτελούνται και οι κατάλληλες ενέργειες. Η υπηρεσία υποστηρίζει το μεγαλύτερο φάσμα των εντολών της Prolog και βεβαίως τις εντολές του Gorgias. Πέρα από τις απλές εντολές, υπάρχουν και κάποιες εντολές που χρειάζονται διαφορετική αντιμετώπιση για να εκτελεστούν σωστά. Σε αυτή την κατηγορία ανήκει η εντολή consult που φορτώνει ένα αρχείο στη βάση της Prolog, η assert για την προσθήκη αξιώματος, η abolish και η unload_file για την αφαίρεση κανόνα και αρχείου αντίστοιχα. Επίσης η εντολή prove του Gorgias, η εντολή write αλλά και οι εντολές findall, bagof και setof της Prolog. Όλες αυτές οι εντολές που αναφέρθηκαν παραπάνω, ως επί των πλείστον αφορούν την διαχείριση της βάσης της Prolog. Για κάθε μια από τις εντολές αυτές επιστρέφεται ξεχωριστό νούμερο και ακολουθείται διαφορετική διαδικασία για την εκτέλεσή τους, ενώ οποιαδήποτε άλλη εντολή εκτελείται με τον ίδιο τρόπο.

Για την consult εντολή, αρχικά ελέγχεται αν το αρχείο που θέλει να φορτώσει ο χρήστης βρίσκεται σε κάποιο φάκελό του, αν έχει δηλαδή ανέβει. Αν ναι, τότε προχωράει στην εκτέλεση της εντολής. Σε διαφορετική περίπτωση επιστρέφει αντίστοιχο μήνυμα σφάλματος. Στην συνέχεια με την βοήθεια της JPL διεπαφής δημιουργείται και εκτελείται η εντολή για την φόρτωση του αρχείου. Να τονιστεί πως χρησιμοποιείται η εντολή load_files της Prolog, αντί για την εντολή consult. Ο λόγος είναι ότι παρατηρήθηκε, πως με την εντολή consult δεν πραγματοποιείται εκκαθάριση της βάσης από το αρχείο στην συνέχεια. Η Prolog επιστρέφει αληθές, αν η φόρτωση του αρχείου ήταν επιτυχημένη και ψευδές σε αντίθετη περίπτωση. Αυτό ακριβώς επιστρέφεται και σαν απάντηση στον χρήστη.

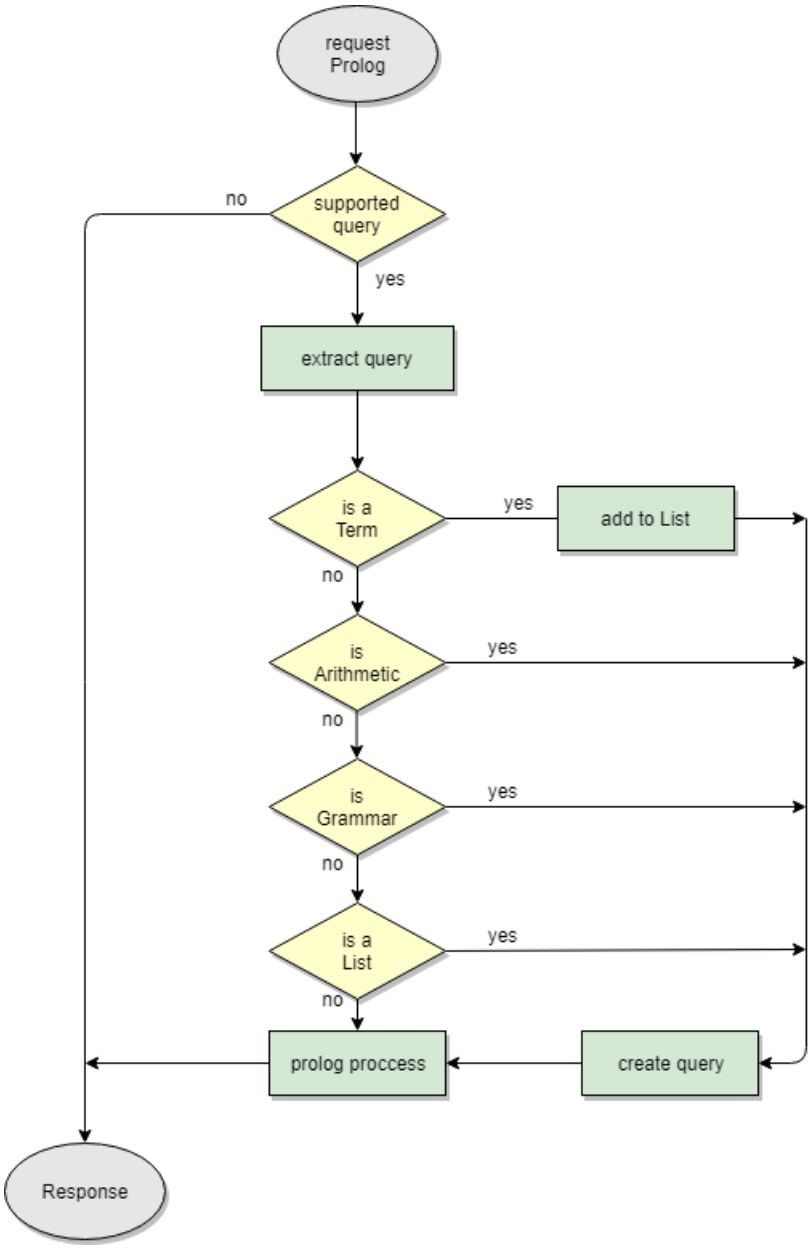
Για την εντολή assert, εφόσον κληθούν κάποιες υπηρεσίες για την εξαγωγή δεδομένων που θα φανούν χρήσιμα στην εκκαθάριση της Prolog βάσης και θα αναλυθούν στην συνέχεια, δημιουργείται η κατάλληλη εντολή και εκτελείται πάλι με την χρήση της JPL διεπαφής. Το αποτέλεσμα και σε αυτή την περίπτωση θα είναι αληθές ή ψευδές και είναι αυτό που επιστρέφεται στον χρήστη.

Έπειτα για τις εντολές abolish**,** unload_file, retract, findall, bagof, setof και prove, χρησιμοποιείται παρόμοια λογική με τις προηγούμενες εντολές. Η διαφορά τους είναι στον σχηματισμό του query που θα αποσταλεί στην μηχανή Prolog. Εφόσον δημιουργηθεί το κατάλληλο query, για τις εντολές consult, assert, abolish, unload_file και retract χρησιμοποιείται η μέθοδος hasSolution της κλάσης Query της διεπαφής JPL. Η μέθοδος αυτή παίρνει ως όρισμα μια συμβολοσειρά, που στην προκειμένη περίπτωση είναι η εντολή, και επιστρέφει μια απάντηση. Αυτός είναι και ο λόγος που χρησιμοποιείται σε αυτές τις εντολές, αφού όλες επιστρέφουν αληθές ή ψευδές σαν αποτέλεσμα. Σε όλες τις περιπτώσεις, ο κώδικας που αφορά την Prolog περικλείεται από try-catch blocks[23], ώστε σε περίπτωση σφάλματος να μην δημιουργηθούν προβλήματα στην σύνδεση και την λειτουργία της Prolog.

Στην περίπτωση των findall, bagof, setof, prove, αλλά και σε περίπτωση που δοθεί οποιαδήποτε άλλη εντολή από τον χρήστη, αμέσως μετά την δημιουργία του query, καλείται η συνάρτηση prologProccess από την αποθήκη υπηρεσιών της εφαρμογής, με ορίσματα την εντολή, τον αριθμό των αποτελεσμάτων και τον χρόνο που θα τρέξει η μηχανή Prolog. Αυτή η συνάρτηση αποτελεί τον πυρήνα, όσο αφορά την λειτουργία και την σύνδεση της Java με την Prolog, και καλείται όταν υπάρχουν πιθανότητες η Prolog να επιστρέψει παραπάνω από ένα αποτελέσματα. Αρχικά, δημιουργείται μια λίστα όπου εκεί αποθηκεύονται όλα τα αποτελέσματα. Εν συνεχεία, αποθηκεύεται σε μια μεταβλητή ο χρόνος του συστήματος την συγκεκριμένη στιγμή. Αν το query έχει λύση η διαδικασία συνεχίζεται, αλλιώς η συνάρτηση επιστρέφει ψευδές. Αν υπάρχει λύση, όσο έχει επόμενη λύση, ο αριθμός των αποτελεσμάτων είναι μέσα στο εύρος των αποτελεσμάτων που έχει ζητήσει ο χρήστης και ο χρόνος είναι μικρότερος από αυτόν που έχει δοθεί, τότε το αποτέλεσμα προστίθεται στην λίστα με τα αποτελέσματα. Η λίστα αυτή περιέχει όλα τα αποτελέσματα και επιστρέφεται

[23] Try-Catch Block: Μπλοκ από εντολές όπου οποιοδήποτε σφάλμα υπάρξει, το catch μπλοκ αναλαμβάνει τον χειρισμό του και δεν διακόπτεται το πρόγραμμα

αρχικά στην καλούσα μέθοδο και στη συνέχεια σε JSON μορφή στον χρήστη που έστειλε το αίτημα. Στη Εικόνα 21 φαίνεται η ροή που ακολουθείται, όπως παρουσιάστηκε σε αυτή την ενότητα.



Εικόνα 21: Διάγραμμα ροής για την επεξεργασία Prolog εντολής

3.6.8 Prolog modules

Όπως αναφέρθηκε στην παραπάνω ενότητα, σε πολλές εντολές η μόνη διαφορά που υπάρχει μεταξύ τους είναι στην δημιουργία του query που θα εκτελεστεί με την βοήθεια της JPL. Η υπόλοιπη διαδικασία είναι η ίδια, αλλά η δομή στο τελικό query που περνάει ως όρισμα είναι διαφορετική. Το ερώτημα που γεννάται είναι ότι εφόσον την εντολή την στέλνει στο σώμα του αιτήματος ο χρήστης, γιατί δεν στέλνεται αυτούσια απευθείας προς την μηχανή Prolog. Η απάντηση είναι ότι κατά αυτόν

τον τρόπο η υπηρεσία απλά δεν θα λειτουργούσε. Η Prolog είναι μια μηχανή που εγκαθίσταται τοπικά και όταν την ξεκινήσει ο χρήστης και φορτώσει αρχεία ή προσθέσει κανόνες κλπ. σε αυτή, αυτά θα προστεθούν στη κοινή βάση δεδομένων της Prolog. Αντίστοιχα, η μηχανή Prolog είναι μονίμως ενεργοποιημένη και κάθε χρήστης που την χρησιμοποιεί μέσω της εφαρμογής, χρησιμοποιεί αυτή την κοινή βάση. Όλοι οι κανόνες, όλα τα αρχεία και γενικά οτιδήποτε φορτώνει ή προσθέτει ο εκάστοτε χρήστης της εφαρμογής, καταλήγουν σε μια κοινή βάση δεδομένων. Γίνεται αντιληπτό πως κατά αυτό τον τρόπο λειτουργίας τα ζητήματα ασφαλείας που προκύπτουν είναι πολλά. Ο χρήστης δεν έχει δικό του χώρο και στην ουσία η βάση δεδομένων της Prolog παίζει τον ρόλο μιας δεξαμενής, όπου όλοι οι χρήστες φορτώνουν ή καταναλώνουν πόρους από εκεί. Όπως αναφέρθηκε προηγουμένως, είναι η φύση της μηχανής Prolog τέτοια που δεν επιτρέπει την δημιουργία διαφορετικών εικόνων ή βάσεων της μηχανής για κάθε χρήστη. Έτσι λοιπόν ο χρήστης έχει πρόσβαση στην ίδια βάση και κατά επέκταση στα αρχεία ή τους κανόνες όλων των χρηστών. Πέρα από την ασφάλεια, τίθεται και το θέμα των σφαλμάτων στις απαντήσεις της Prolog, αφού κανόνες μπορεί να έρχονται σε σύγκρουση.

Την λύση σε αυτό το ζήτημα έδωσε μια λειτουργία της βάσης της Prolog, η λειτουργία των modules. Όταν δημιουργείται ένα module στην Prolog βάση, δημιουργείται ένας χώρος, στον οποίο η πρόσβαση επιτυγχάνεται μόνο αν ο χρήστης θέσει το όνομα του module πριν την εντολή. Για παράδειγμα, η εντολή 'mdl: consult('file.pl').' θα φορτώσει στο module ονόματι mdl το αρχείο file.pl. Πρόσβαση σε αυτό το αρχείο θα επιτυγχάνεται μόνο με το mdl μπροστά από κάθε ερώτηση ή εντολή. Σε αντίθετη περίπτωση η μηχανή Prolog δεν θα βρίσκει το αρχείο. Πρόκειται για έναν τρόπο απομόνωσης, που στην εφαρμογή χρησιμοποιείται για να λύσει το πρόβλημα που παρουσιάστηκε παραπάνω.

Σε κάθε αίτημα που στέλνει ο χρήστης της εφαρμογής, όπως αναφέρθηκε και στην προηγούμενη ενότητα, αποθηκεύεται σε μία μεταβλητή το username του. Αυτή ακριβώς η μεταβλητή μπαίνει ως πρόθεμα πριν την εντολή και ουσιαστικά έτσι δημιουργείται ένα module με όνομα το username του χρήστη. Εφόσον το username του κάθε χρήστη είναι μοναδικό, αντίστοιχα και το κάθε module θα είναι μοναδικό και δεν τίθεται θέμα διπλοτύπων. Αποκτά έναν δικό του χώρο μέσα στην βάση δεδομένων της Prolog και απομονώνεται από τους άλλους χρήστες της εφαρμογής. Σε ορισμένες εντολές το όνομα του module πρέπει να εισαχθεί στην αρχή, όπως στην εντολή prove (π.χ. mdl: prove('...').), ενώ σε αυτές που αφορούν την διαχείριση της βάσης της Prolog προστίθενται συνήθως μετά την εντολή (π.χ. assert('mdl: …').). Αυτός είναι και ο κύριος λόγος που κάθε query έχει διαφορετική δομή, ανάλογα πάντα με την εντολή που έχει δοθεί.

3.6.9 Εκκαθάριση της μηχανής Prolog

Όσο ο χρήστης χρησιμοποιεί την εφαρμογή, προσθέτει ή φορτώνει αρχεία στη βάση δεδομένων της Prolog. Εφόσον η βάση της Prolog είναι κοινή για όλους τους χρήστες και η μηχανή της είναι μονίμως ενεργή, ο αριθμός των αρχείων και των κανόνων που προστίθενται σε αυτή συνεχώς θα αυξάνεται. Ο χρήστης μπορεί βεβαίως να κάνει χρήση αντίστοιχων εντολών για τον καθαρισμό της βάσης, αλλά γίνεται αντιληπτό πως δεν είναι καθόλου καλή τακτική να βασιστεί η εφαρμογή σε αυτό. Για τον καθαρισμό της βάσης από την χρήση ενός χρήστη, εκτελείται η αντίστοιχη επιχειρησιακή λογική, μόλις ο εκάστοτε χρήστης αποσυνδεθεί από την εφαρμογή. Με αυτόν τον τρόπο μπορεί να ειπωθεί πως δημιουργήθηκε μια λειτουργία καθαρισμού αντίστοιχη με τον garbage collector της Java.

Για την υλοποίηση της λειτουργίας καθαρισμού δημιουργήθηκαν δύο multivalue maps (πίνακες αντιστοιχίσεων πολλαπλών τιμών). Πρόκειται για maps στα οποία το κλειδί μπορεί να εμφανιστεί πάνω από μια φορά. Το ένα multivalue map αφορά τους κανόνες που έχουν εισαχθεί από τον χρήστη και περιέχει ως κλειδί το username του χρήστη και ως τιμές το κατηγόρημα (predicate) που έχει εισαχθεί στη βάση και πρέπει να γίνει abolish, αλλά και τον αριθμό τάξεως (arity) το οποίο είναι αναγκαίο για την εντολή abolish. Το δεύτερο multivalue map αφορά τα αρχεία που έχει φορτώσει στη βάση δεδομένων της Prolog ο χρήστης και περιέχει ως κλειδί το username του χρήστη και σαν τιμές

το αρχείο που έχει φορτωθεί από τον χρήστη και τον φάκελο στον οποίο βρίσκεται. Κάθε φορά που ο χρήστης πραγματοποιεί assert ή consult, προστίθεται νέα τιμή στον αντίστοιχο πίνακα με κλειδί το username του και τιμή έναν πίνακα δύο στοιχείων, όπως περιγράφηκαν προηγουμένως. Με την αποσύνδεση του χρήστη, σκανάρονται οι δύο πίνακες και όπου βρεθεί κλειδί με το username του χρήστη πραγματοποιείται abolish ή unload_file, ανάλογα τον πίνακα. Να τονιστεί πως χρησιμοποιείται η εντολή abolish και όχι η retract, γιατί η abolish διαγράφει το predicate οριστικά από την βάση, ενώ η retract θα μπορούσε να ειπωθεί ότι απλά το απενεργοποιεί. Για να καταστεί σαφές, αν ο χρήστης ρωτήσει την Prolog για το κατηγόρημα μετά από εντολή retract σε αυτό, η απάντηση της Prolog θα είναι «ψευδές», ενώ αν ρωτήσει για το κατηγόρημα μετά από την χρήση της εντολής abolish σε αυτό, η Prolog θα αποκριθεί «σφάλμα», αφού το συγκεκριμένο κατηγόρημα δεν θα βρίσκεται στην βάση. Μόλις καθαριστεί η βάση από τις προσθήκες του χρήστη, διαγράφεται το username-κλειδί του χρήστη από τους πίνακες, μαζί βεβαίως με τις τιμές του.

Με αυτόν τον τρόπο καθαρίζεται η βάση της Prolog, μια πολύ σημαντική λειτουργία στην συνολική λειτουργία και αποδοτικότητα της εφαρμογής. Με αυτή την λειτουργία επίσης κλείνει και η ενότητα των λειτουργιών που αφορούν την Prolog.

Στις προηγούμενες ενότητες αναλύθηκε η υπηρεσία που τρέχει στον εξυπηρετητή. Η υπηρεσία αυτή εξυπηρετεί τα αιτήματα που δέχεται και επιστρέφει μια απάντηση. Ένα άλλο σημαντικό στοιχείο είναι με ποιους τρόπους μπορεί ο χρήστης να καταναλώσει την υπηρεσία αυτή, να στείλει δηλαδή ένα αίτημα και να λάβει την απάντηση. Η κατανάλωση αυτής της υπηρεσίας επιτυγχάνεται με δύο τρόπους, ο πρώτος θα αναλυθεί στην επόμενη και τελευταία ενότητα του κεφαλαίου 3, ενώ ο δεύτερος θα παρουσιαστεί στην επόμενη ενότητα.

3.7 Κατανάλωση υπηρεσιών μέσω Spring RestTemplate

Ο πρώτος τρόπος κατανάλωσης της υπηρεσίας είναι μέσω της RestTemplate [20] του Spring [10]. Με την βοήθεια αυτής της διεπαφής που παρέχει το Spring framework, έχει δημιουργηθεί ένα μέρος χρήστη, το οποίο εξάγεται ως jar[24]. Ενσωματώνοντας αυτή την βιβλιοθήκη στο Java πρόγραμμα και εφόσον ο χρήστης είναι εγγεγραμμένος στην υπηρεσία, έχει την δυνατότητα να χρησιμοποιήσει όλες τις κύριες λειτουργίες της υπηρεσίας στο πρόγραμμά του, καλώντας τις συναρτήσεις που προσφέρει η διεπαφή. Πρόκειται ουσιαστικά για μια διεπαφή που έχει δημιουργηθεί και μια κλάση που την υλοποιεί. Η διεπαφή περιέχει τις αφηρημένες μεθόδους, οι οποίες υλοποιούνται στην αντίστοιχη κλάση.

Όπως φαίνεται και στην Εικόνα 22, η διεπαφή περιέχει μεθόδους για είσοδο του χρήστη στην υπηρεσία, για την φόρτωση αρχείου στη βάση της Prolog, για την επιστροφή των φακέλων του χρήστη, για την εντολή prove του Gorgias, για οποιοδήποτε εντολή προς την μηχανή Prolog, αλλά και για την έξοδο από την υπηρεσία. Αντίστοιχα, η κλάση ClientServicesRepoImpl υλοποιεί τις μεθόδους της ClientServiceRepo διεπαφής.

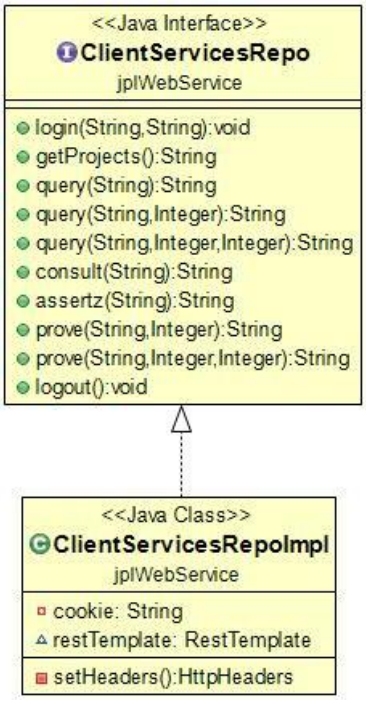
Πιο αναλυτικά, ο χρήστης για την σύνδεση του με την υπηρεσία πρέπει αρχικά να χρησιμοποιήσει στο πρόγραμμά του την συνάρτηση login() με ορίσματα το username και τον κωδικό πρόσβασης του. Τα στοιχεία του αυτά κωδικοποιούνται και στέλνονται στην υπηρεσία για επαλήθευση και σύνδεση με αυτήν. Εφόσον η είσοδος στεφθεί με επιτυχία, εν συνεχεία μπορεί να χρησιμοποιήσει στο πρόγραμμά του οποιοδήποτε συνάρτηση επιθυμεί, χωρίς να χρειάζεται να παρέχει τα στοιχεία του στην εκάστοτε συνάρτηση που θα χρησιμοποιήσει.

[24] Jar: Εκτελέσιμο αρχείο Java που μπορεί να χρησιμοποιηθεί ως βιβλιοθήκη προγράμματος ή ως αυτόνομο πρόγραμμα

Για την πραγματοποίηση αυτής της λειτουργίας χρειάστηκε να δημιουργηθεί μια συνάρτηση στην κλάση ClientServicesRepoImpl που ονομάζεται setHeaders(). Όταν ο χρήστης κάνει επιτυχημένη είσοδο στην υπηρεσία, η υπηρεσία ανταποκρίνεται με ένα μήνυμα. Στις επικεφαλίδες του μηνύματος αυτού περιέχεται και μια επικεφαλίδα που ονομάζεται SET-COOKIE, η οποία έχει μια μοναδική συμβολοσειρά που ταυτοποιεί τον χρήστη και δημιουργείται μετά από την επιτυχή σύνδεσή του. Ο χρήστης πλέον δεν χρειάζεται να στείλει ξανά τα στοιχεία του στα επόμενα αιτήματά του. Αρκεί να στείλει αυτή την συμβολοσειρά στις επικεφαλίδες του αιτήματος και η υπηρεσία θα αναγνωρίσει ότι είναι ήδη συνδεδεμένος.

Πιο συγκεκριμένα, όταν πραγματοποιηθεί η είσοδος του χρήστη στην υπηρεσία, εξάγεται αυτή η συμβολοσειρά και αποθηκεύεται στην μεταβλητή cookie. Η μεταβλητή cookie αρχικά είναι κενή και με την επιτυχή είσοδο του χρήστη παίρνει τιμή. Σε κάθε συνάρτηση που απαιτείται η ταυτοποίηση του χρήστη, εμπεριέχεται και η συνάρτηση setHeaders(). Σκοπός της είναι να θέσει την τιμή της μεταβλητής cookie στην επικεφαλίδα με ονομασία Cookie του αιτήματος που θα σταλθεί. Εφόσον ο χρήστης έχει συνδεθεί επιτυχώς, η μεταβλητή cookie παίρνει την κατάλληλη τιμή και ο χρήστης καταναλώνει την εκάστοτε υπηρεσία χωρίς πρόβλημα. Σε αντίθετη περίπτωση η επικεφαλίδα Cookie δεν θα έχει κάποια τιμή και η υπηρεσία θα ανταποκριθεί με μήνυμα σφάλματος και συγκεκριμένα με κωδικό σφάλματος 401 (ο χρήστης δεν είναι εξουσιοδοτημένος).

Πέρα από την συνάρτηση για την είσοδο στην υπηρεσία, υπάρχει και η συνάρτηση για την ανάκτηση όλων των φακέλων του χρήστη. Αυτή η συνάρτηση στέλνει ένα GET αίτημα προς την υπηρεσία, η οποία αποκρίνεται με μια λίστα σε JSON μορφή με όλους τους φακέλους που έχει δημιουργήσει ο χρήστης. Οι συναρτήσεις consult() και assertz() στέλνουν αιτήματα με την μέθοδο POST για την φόρτωση αρχείου στη βάση της Prolog μηχανής και την πρόσθεση ενός αξιώματος αντίστοιχα. Όπως φαίνεται στην Εικόνα 22, υπάρχουν τρεις μέθοδοι για να στείλει ο χρήστης ένα αίτημα query και δύο μέθοδοι για να στείλει ένα αίτημα prove προς την υπηρεσία και κατά επέκταση προς την Prolog μηχανή. Ο λόγος που δημιουργήθηκαν τόσες συναρτήσεις δεν είναι άλλος από το να δοθούν περισσότερες επιλογές στον χρήστη. Ανάλογα τις προτιμήσεις του μπορεί να καθορίσει για πόση ώρα θα τρέχει η μηχανή Prolog ή πόσα αποτελέσματα θα πάρει ως απάντηση. Αν θέλει να χρησιμοποιήσει τις προκαθορισμένες τιμές, στέλνει απλά την εντολή προς την υπηρεσία, κάνοντας χρήση της συνάρτησης query ή prove, που δέχονται ως μοναδικό όρισμα μια συμβολοσειρά.



Εικόνα 22: Διεπαφή για το μέρος πελάτη

Για τις ανάγκες της δημιουργίας και αποστολής του αιτήματος προς την υπηρεσία, δημιουργήθηκε η κλάση QueryModel. Η κλάση αυτή αποτελείται από τρεις μεταβλητές, την query για την εντολή που θα σταλθεί, την size για τον αριθμό τον απαντήσεων που θα επιστραφεί και την sec για τον χρόνο που θα τρέξει η μηχανή Prolog. Επίσης, έχει τις αντίστοιχες get και set μεθόδους, αλλά και έναν κατασκευαστή.

Η κλάση αυτή χρησιμεύει στην δημιουργία του αντικειμένου που θα σταλθεί στην υπηρεσία. Αυτό το αντικείμενο, πέρα από την εντολή προς την μηχανή Prolog, αποτελείται από το αριθμό των απαντήσεων και τον χρόνο που θα τρέξει η Prolog, όπως αναφέρθηκε προηγουμένως. Δημιουργείται εσωτερικά των συναρτήσεων και παίρνει είτε προκαθορισμένες τιμές, είτε τιμές που θέτει ο χρήστης στέλνοντάς τες ως ορίσματα συνάρτησης. Τέλος, στέλνεται μαζί με το αίτημα προς την υπηρεσία, όπου εκεί εξάγονται οι τιμές του και χρησιμοποιούνται κατάλληλα, όπως αναλύθηκε σε αντίστοιχη ενότητα του προηγούμενου κεφαλαίου.

Λόγω του ότι η επιστροφή της SWI-Prolog προς την Java, σε ότι αφορά τις εντολές που αφορούν τον Gorgias, δεν είναι σε μορφή φιλική προς τον χρήστη, δημιουργήθηκε επιπρόσθετα ένας αποκωδικοποιητής. Αν ο χρήστης επιθυμεί τα αποτελέσματα να είναι εύκολο να διαβαστούν, τότε έχει την δυνατότητα να τα περάσει από τον αποκωδικοποιητή πρώτα. Πρόκειται για μια συνάρτηση που διαμορφώνει κατάλληλα τα αποτελέσματα και είναι στη βούληση του χρήστη αν θα την χρησιμοποιήσει ή όχι.

Πέρα από τις μεθόδους που αφορούν στην αλληλεπίδραση με την μηχανή Prolog, υπάρχουν και μέθοδοι, όπως η addFile, η οποία δίνει την δυνατότητα στον χρήστη να ανεβάσει ένα Prolog αρχείο, περνώντας ως παράμετρο το μονοπάτι όπου βρίσκεται τοπικά το αρχείο που επιθυμεί να ανεβάσει.

Η τελευταία κατά σειρά μέθοδος της διεπαφής ClientServiceRepo είναι η logout(). Αυτή η μέθοδος δημιουργήθηκε για την αποσύνδεση του χρήστη από την υπηρεσία και καλό είναι να καλείται στο πρόγραμμα, όταν δεν χρειάζεται περαιτέρω η κατανάλωση της υπηρεσίας. Αυτό που κάνει η συγκεκριμένη μέθοδος είναι να στείλει ένα GET αίτημα προς την υπηρεσία και συγκεκριμένα την λειτουργία αποσύνδεσης χρήστη. Η υπηρεσία με την σειρά της θα καθαρίσει την βάση της Prolog από την χρήση του χρήστη και θα τερματίσει την συνεδρία που ξεκίνησε, όταν συνδέθηκε αρχικά. Η τιμή του cookie δεν θα ισχύει πλέον και αν ο χρήστης επιθυμεί να χρησιμοποιήσει στο πρόγραμμά του ξανά την υπηρεσία, θα πρέπει να κάνει εκ νέου σύνδεση. Αυτός είναι και ο λόγος που συνίσταται η μέθοδος logout να καλείται στο τέλος του προγράμματος. Αν ο χρήστης δεν την καλέσει σε κανένα σημείο του προγράμματος, τότε η συνεδρία θα παραμείνει ενεργή για τριάντα λεπτά και έπειτα θα πραγματοποιηθεί αυτόματη αποσύνδεση από την υπηρεσία.

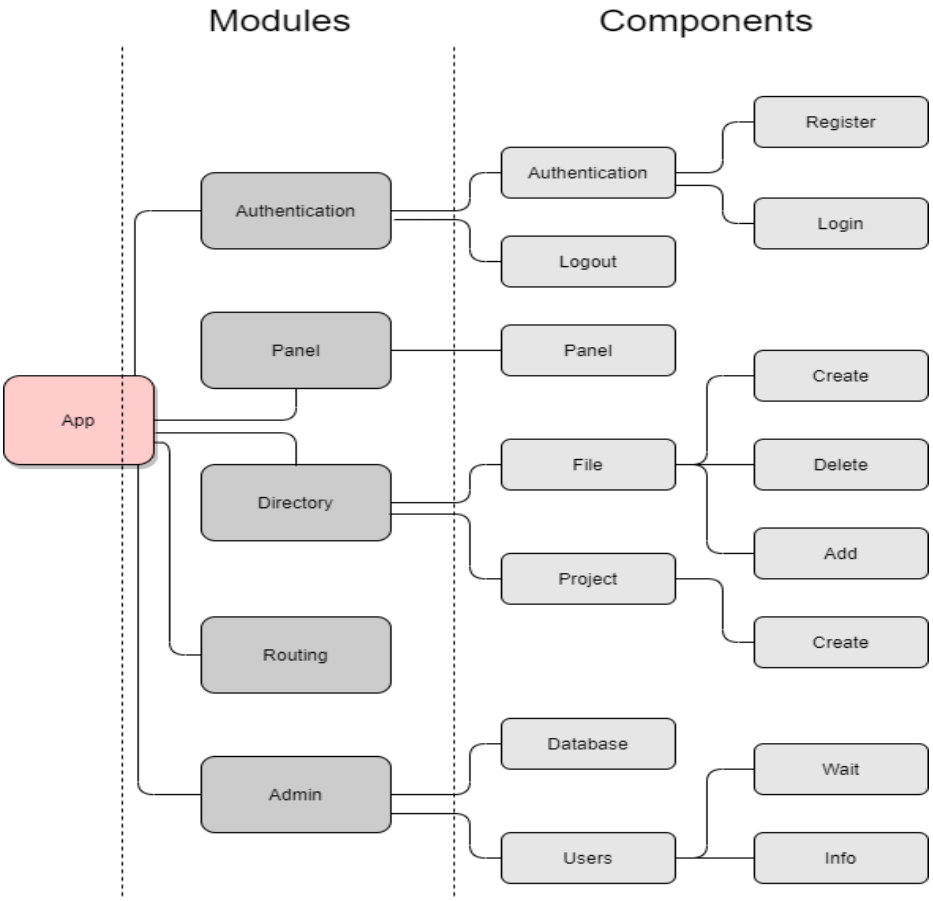
Υπήρχε η δυνατότητα οποιαδήποτε υπηρεσία της εφαρμογής να καταναλώνονταν μέσω αυτού του τμήματος χρήστη που δημιουργήθηκε. Για παράδειγμα να υπήρχαν παρόμοιες συναρτήσεις για την εγγραφή στην υπηρεσία ή για την δημιουργία νέου φακέλου. Εφόσον όμως, όπως θα δειχθεί και στη συνέχεια, υπάρχει ένα ολοκληρωμένο γραφικό περιβάλλον, όπου ο χρήστης μπορεί με ευκολία να πραγματοποιήσει τέτοιου είδους λειτουργίες, κρίθηκε καλύτερο να περιοριστούν οι μέθοδοι της συγκεκριμένης διεπαφής σε εκείνες που έχουν σχέση αμιγώς με την χρήση της Prolog μηχανής.

Κεφάλαιο 4 - Διεπαφή χρήστη

Για τη δημιουργία της διεπαφής χρήστη, εκείνου του τμήματος δηλαδή με το οποίο έρχεται σε επαφή ο χρήστης και αλληλεπιδρά μαζί του, χρησιμοποιήθηκε ένα framework ονόματι Angular [13] [21]. Η Angular αποτελεί μια από τις δημοφιλέστερες πλατφόρμες για την δημιουργία διαδικτυακών υπηρεσιών. Στις επόμενες ενότητες να παρουσιαστεί η διεπαφή χρήστη που δημιουργήθηκε για την εφαρμογή, τα μέρη της καθώς και ο τρόπος που καταναλώνει και αλληλεπιδρά με την υπηρεσία που τρέχει στον εξυπηρετητή.

4.1 Τα components της διεπαφής χρήστη

Στην Εικόνα 23 φαίνονται όλα τα modules και κατά επέκταση τα components που έχουν δημιουργηθεί για την εφαρμογή. Όπως αναφέρθηκε και στην αντίστοιχη ενότητα για την παρουσίαση των δομικών κομματιών που αποτελούν την Angular, η εφαρμογή αποτελείται από modules που με την σειρά τους αποτελούνται από components. Όπως γίνεται αντιληπτό, όλα τα modules της εφαρμογής είναι παιδιά του App module, το οποίο αποτελεί το πρωταρχικό module της εφαρμογής. Από εκεί και πέρα, το Authentication module περιέχει components που αφορούν την ταυτοποίηση, την εγγραφή, την είσοδο και την έξοδο από την εφαρμογή. Το Panel module περιέχει το component του πάνελ, ενώ το Directory module όλα τα components που έχουν σχέση με το σύστημα διαχείρισης των αρχείων του χρήστη. Όλα τα components που έχουν σχέση με τις λειτουργίες που παρέχει η διεπαφή χρήστη στον διαχειριστή βρίσκονται στο Admin module. Το Routing module τέλος καθορίζει τις διαδρομές μεταξύ των components και των URLs και αποτελεί ένα module από μόνο του.



Εικόνα 23: Modules και Components της διεπαφής χρήστη

Για την πλοήγηση ανάμεσα σε διαφορετικά Components και κατά επέκταση σε διαφορετικές σελίδες της εφαρμογής, χρησιμοποιείται ο Router. Ο Router είναι ένα Module, μια βιβλιοθήκη δηλαδή της Angular, που με την χρησιμοποίησή του καθιστά την εφαρμογή ως εφαρμογή μονής σελίδας (Single Page Application - SPA) και αυτό συμβαίνει γιατί δεν χρειάζεται να ανανεώνεται ολόκληρη η σελίδα για την πλοήγηση ανάμεσα στα Components της εφαρμογής.

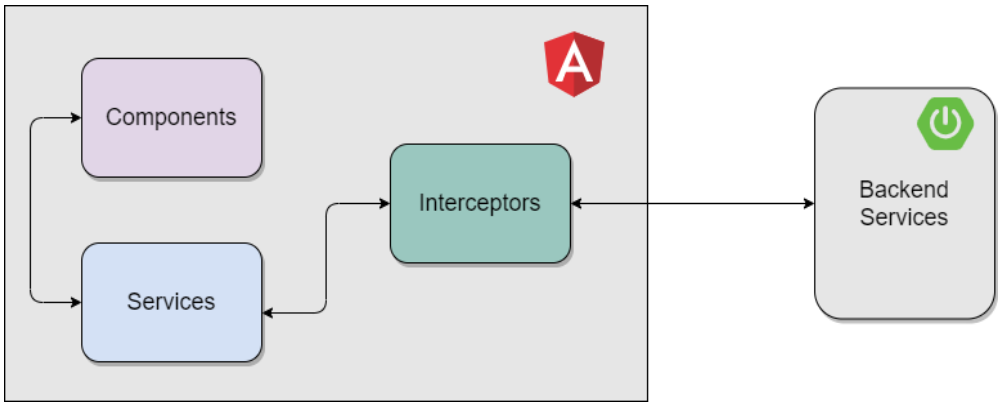
4.2 Βελτιστοποίηση απόδοσης

Για την βελτιστοποίηση της απόδοσης στη διεπαφή χρήστη χρησιμοποιήθηκαν δύο τεχνολογίες. Η πρώτη έχει άμεση σχέση με την προηγούμενη ενότητα. Είναι μια τεχνολογία που την προσφέρει η Angular και ονομάζεται lazy loading. Το lazy loading είναι μια διαδικασία που φορτώνει τα modules δυναμικά. Σε αντίθεση με την φόρτωση όλων των modules κατά την εκκίνηση της εφαρμογής, η εφαρμογή φορτώνει μόνο το module που χρειάζεται κάθε φορά. Για παράδειγμα, όταν ο χρήστης κατευθυνθεί στο πάνελ, τότε θα φορτωθεί το module του πάνελ. Γίνεται αντιληπτό πως κάτι τέτοιο μειώνει τον χρόνο απόκρισης της εφαρμογής και αυξάνει την εμπειρία του χρήστη.

Η δεύτερη τεχνολογία αφορά την cache μνήμη στο κομμάτι του Front End της εφαρμογής. Σε περιπτώσεις που δεν υπάρχουν αλλαγές σε φάκελους ή αρχεία του χρήστη, η εφαρμογή αποθηκεύει τα δεδομένα στην μνήμη cache. Έτσι την επόμενη φορά που ο χρήστης θα ζητήσει έναν φάκελο ή ένα αρχείο, η διεπαφή θα του το σερβίρει από εκεί. Με αυτό τον τρόπο επιτυγχάνεται η μείωση των αιτημάτων προς τον εξυπηρετητή αλλά και ο χρόνος απόκρισης της εφαρμογής.

4.3 Φίλτρα ελέγχου (Interceptors)

Στο τέλος της προηγούμενης ενότητας έγινε αναφορά στη λειτουργία του Router Module, ενός Module για την δρομολόγηση από Component σε Component, χωρίς να χρειάζεται να ανανεώνεται ολόκληρη η σελίδα. Αυτό αμέσως γεννάει κενά ασφαλείας, εφόσον για την χρήση της υπηρεσίας και κατά επέκταση όλων των λειτουργιών της απαιτείται είσοδος χρήστη. Για παράδειγμα πως αντιδρά η εφαρμογή, αν ο χρήστης στείλει URI χωρίς να έχει πραγματοποιήσει είσοδο; Πως ασφαλίζονται τα URIs από τέτοιες περιπτώσεις; Πως στέλνει σε κάθε αίτημα τα στοιχεία του ο χρήστης; Πως διαχειρίζεται η διεπαφή χρήστη μια απάντηση του εξυπηρετητή με κωδικό σφάλματος; Σε όλες αυτές τις ερωτήσεις η απάντηση είναι η ίδια, με την χρήση interceptors. Οι interceptors μπορεί να θεωρηθεί ότι πρόκειται για φίλτρα που παρεμβάλλονται, πριν σταλθεί ένα αίτημα ή πριν παρθεί μια απάντηση και εκτελούν λειτουργίες.



Εικόνα 24: Φίλτρα ελέγχου της διεπαφής χρήστη

Για τις ανάγκες τις εφαρμογής έχουν δημιουργηθεί τρία τέτοια φίλτρα, τα οποία παίζουν πολύ σημαντικό ρόλο στην αρμονική λειτουργία της. Το πρώτο και κυριότερο φίλτρο που δημιουργήθηκε αφορά την είσοδο χρήστη. Με παρόμοια λογική με το μέρος χρήστη μέσω Spring που δημιουργήθηκε, με την επιτυχημένη είσοδο του χρήστη η υπηρεσία στον εξυπηρετητή αποκρίνεται στις επικεφαλίδες της απάντησης με ένα cookie. Αυτή η συμβολοσειρά είναι που ταυτοποιεί τον χρήστη και αποθηκεύεται στον αποθηκευτικό χώρο του περιηγητή (local storage) σαν μία μεταβλητή. Σε κάθε νέο αίτημα προς τον εξυπηρετητή παρεμβάλλεται ο αντίστοιχος interceptor, ελέγχει αν υπάρχει αυτή η μεταβλητή στον αποθηκευτικό χώρο του περιηγητή και αν όντως υπάρχει την προσθέτει στις επικεφαλίδες του αιτήματος και στέλνει το αίτημα. Για κάθε χρήση της υπηρεσίας απαραίτητη προϋπόθεση είναι η είσοδος του χρήστη. Σε περίπτωση που κλείσει το παράθυρο του περιηγητή χωρίς να κάνει αποσύνδεση, αυτομάτως η μεταβλητή διαγράφεται από τον αποθηκευτικό χώρο και απαιτείται εκ νέου είσοδος.

Το δεύτερο φίλτρο που δημιουργήθηκε παρεμβάλλεται σε κάθε αίτημα και προστατεύει την εφαρμογή από την εμφάνιση σελίδων της που απαιτούν είσοδο χρήστη, χωρίς όμως αυτή να έχει πραγματοποιηθεί. Το σκεπτικό είναι παρόμοιο με το αυτό του πρώτου φίλτρου. Όταν ο χρήστης αιτείται μία σελίδα που απαιτεί είσοδο χρήστη, ο interceptor ελέγχει αν στον αποθηκευτικό χώρο υπάρχει η μεταβλητή με το cookie. Σε μια τέτοια περίπτωση, ο χρήστης έχει πραγματοποιήσει επιτυχημένη είσοδο και προωθείται στην σελίδα που αιτήθηκε. Αν όμως δεν υπάρχει cookie στον αποθηκευτικό χώρο του περιηγητή, το φίλτρο λειτουργεί ως φύλακας, απαγορεύοντας την προώθηση του χρήστη στην σελίδα που αιτήθηκε και τον ανακατευθύνει στην αρχική σελίδα, ώστε να κάνει είσοδο ή εγγραφή.

Το τρίτο φίλτρο αφορά την περίπτωση που ο εξυπηρετητής θα απαντήσει με κωδικό σφάλματος. Αυτός ο interceptor παραλαμβάνει την απάντηση του εξυπηρετητή και αντιδρά ανάλογα. Ανάλογα την απάντηση, ανακατευθύνει, αλλάζει τιμές σε μεταβλητές ή μπορεί και να εκτελέσει επιχειρησιακή λογική.

Για την δημιουργία ενός interceptor αρκούν δύο και μόνο βήματα. Αρχικά, η δημιουργία μιας κλάσης, η οποία θα υλοποιεί την HttpInterceptor και θα έχει decorator @injectable, και έπειτα η προμήθεια του interceptor μέσω του AppModule. Οι interceptors κάνουν πολλές δουλειές μέσα στην εφαρμογή και σε πολλές περιπτώσεις δίνουν την λύση γράφοντας αισθητά λιγότερο κώδικα. Χρησιμοποιούνται άλλοτε σαν φύλακες, άλλοτε για να ελέγξουν την ροή της εφαρμογής και γενικά αποτελούν πολύ χρήσιμο εργαλείο για την διαχείριση της εφαρμογής.

4.4 Οι υπηρεσίες της διεπαφής χρήστη

Μια υπηρεσία, όπως αναφέρθηκε και σε προηγούμενη ενότητα, είναι μια κλάση που έχει έναν συγκεκριμένο σκοπό. Στην συγκεκριμένη εφαρμογή υπάρχουν δύο τύποι υπηρεσιών που υλοποιούνται. Ο πρώτος τύπος αφορά τις υπηρεσίες που σκοπός τους είναι η αποστολή αιτήματος για την κατανάλωση μιας υπηρεσίας του εξυπηρετητή. Οι περισσότερες υπηρεσίες δημιουργήθηκαν για αυτό τον σκοπό. Στέλνουν ασύγχρονα αιτήματα προς τον εξυπηρετητή και επιστρέφουν την απόκριση του. Παραδείγματος χάρη, μόλις ο χρήστης πατήσει το κουμπί για είσοδο στην υπηρεσία, η αντίστοιχη κλάση του Component θα εξάγει τα στοιχεία του και θα καλέσει την κατάλληλη υπηρεσία, η οποία με την σειρά της θα στείλει ασύγχρονα το αίτημα στον εξυπηρετητή του Back End. Μέσα στην κλάση ελέγχεται η απάντηση του εξυπηρετητή και αν αποκριθεί με μήνυμα επιτυχίας, με την βοήθεια του router module θα κατευθύνει τον χρήστη στην κατάλληλη σελίδα. Σε διαφορετική περίπτωση δεν θα επιτρέψει την είσοδο στον χρήστη, προειδοποιώντας τον για εσφαλμένα στοιχεία. Αυτό αποτελεί ένα παράδειγμα για το πως λειτουργεί το μέρος της εφαρμογής που αλληλεπιδρά με τον χρήστη και ποια η χρησιμότητα των υπηρεσιών σε αυτό. Για κάθε ενέργεια του χρήστη που

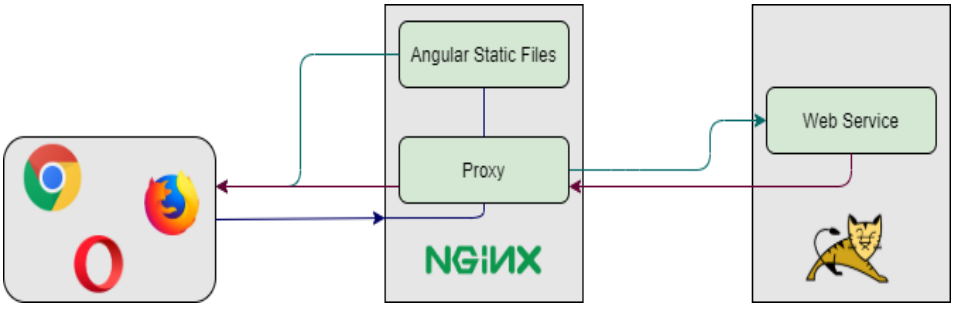
απαιτείται σύνδεση με το μέρος της εφαρμογής που τρέχει στο παρασκήνιο, εκτελείται παρόμοια επιχειρησιακή λογική.

Στην διεπαφή χρήστη έχουν δημιουργηθεί ποικίλες υπηρεσίες. Υπηρεσίες που αποτελούνται από μεθόδους που επικοινωνούν με το Back End ασύγχρονα με την βοήθεια του Http client module της Angular, όπως και υπηρεσίες που διασφαλίζουν ότι οι σελίδες που αφορούν τον διαχειριστή θα είναι διαθέσιμες μόνο σε αυτόν. Υπηρεσίες σχετικά με θέματα ταυτοποίησης και ασφάλειας της εφαρμογής και τέλος υπηρεσίες για την διαχείριση των φακέλων και των αρχείων του χρήστη, αλλά και του πάνελ της διεπαφής χρήστη. Σε όλες αυτές τις υπηρεσίες, αλλά και στα components της διεπαφής, γίνεται χρήση της RxJS βιβλιοθήκης. Όπου υπάρχει ασύγχρονη ροή δεδομένων, έρχονται στο προσκήνιο τα Observables, ενώ σε περιπτώσεις που οι τιμές που επιστρέφονται χρειάζονται διαμόρφωση, χρησιμοποιούνται διάφοροι τελεστές της βιβλιοθήκης.

Ο δεύτερος τύπος υπηρεσιών, είναι οι υπηρεσίες που έχουν σκοπό τον διαμοιρασμό δεδομένων ανάμεσα στα κομμάτια της εφαρμογής. Πιο συγκεκριμένα αυτός ο τύπος υπηρεσιών χρησιμοποιείται για την αλλαγή μιας μεταβλητής ενός component, όχι από την κλάση του component όπου ανήκει, αλλά από την κλάση ενός άλλου component. Λειτουργεί δηλαδή σαν συνδετικός κρίκος ανάμεσα σε δύο ή και παραπάνω components με σκοπό το μοίρασμα των μεταβλητών σε αυτά, όποτε χρειαστεί. Στο συγκεκριμένο κομμάτι γίνεται χρήση του reactive προγραμματισμού και αυτό γιατί εσωτερικά των υπηρεσιών γίνεται χρήση της RxJS βιβλιοθήκης. Μόλις αλλάξει κάτι σε ένα component που επηρεάζει και άλλα components, αυτομάτως μέσω της εκάστοτε υπηρεσίας πυροδοτείται ένα γεγονός. Όποιος Observer έχει κάνει εγγραφή στο Observable που εκπέμπει το γεγονός, ενημερώνεται και αν χρειαστεί ανανεώνει τις μεταβλητές του.

4.5 Ο ρόλος του NGINX εξυπηρετητή

Συνολικά η εφαρμογή και ο τρόπος που έχει αναπτυχθεί και εγκατασταθεί, χρησιμοποιεί δύο εξυπηρετητές. Ο ένας είναι ο Apache Tomcat[25] στον οποίο είναι εγκατεστημένη η διαδικτυακή υπηρεσία και ο άλλος είναι ο NGINX[26] στον οποίο είναι εγκατεστημένη η διεπαφή χρήστη. Ο NGINX εξυπηρετητής για τις ανάγκες της εφαρμογής έχει διαμορφωθεί κατάλληλα, ώστε να λειτουργεί ως εξυπηρετητής μεσολάβησης . Όπως φαίνεται και στην Εικόνα 25, εξυπηρετεί τα αιτήματα που αφορούν στατικά αρχεία, ενώ όποιο αίτημα αφορά κατανάλωση υπηρεσίας του Back End, προωθείται εκεί.

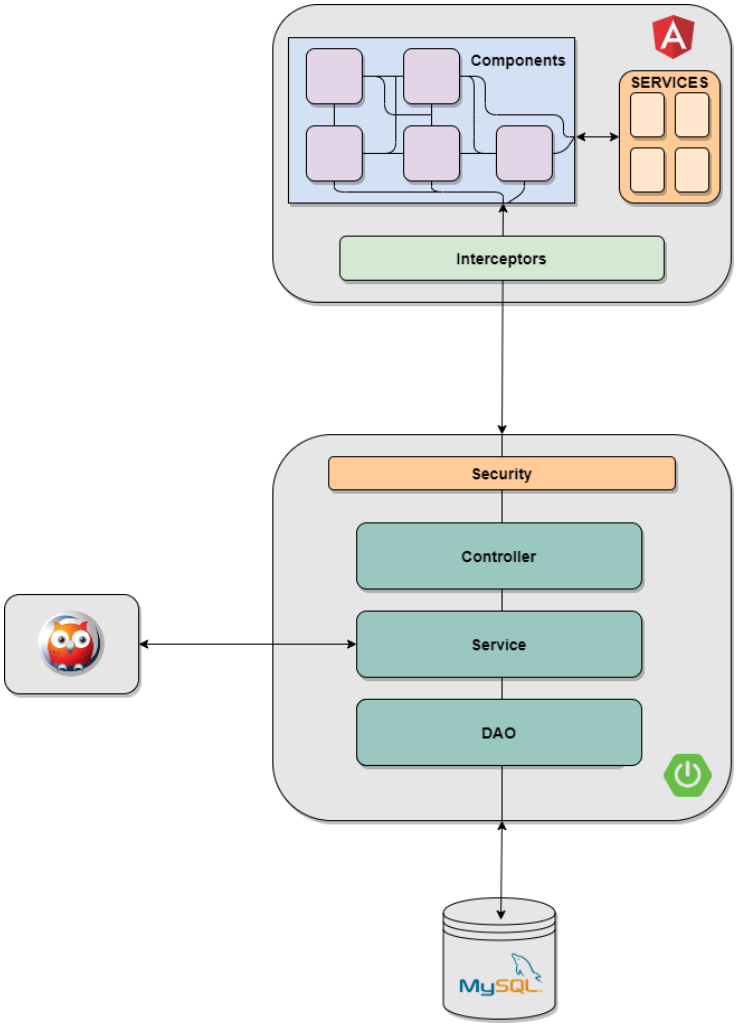


Εικόνα 25: Ο ρόλος του NGINX εξυπηρετητή

[25] Apache Tomcat: http://tomcat.apache.org/
[26] NGINX: https://www.nginx.com/

4.6 Συνολική εικόνα της εφαρμογής

Στις προηγούμενες ενότητες αναλύθηκαν τα εργαλεία που χρησιμοποιήθηκαν για την εφαρμογή, το κομμάτι του εξυπηρετητή, το κομμάτι που αλληλεπιδρά με τον χρήστη και γενικά οι λειτουργίες και το πως δουλεύει η εφαρμογή. Η συνολική εικόνα της εφαρμογής και μετά την παρουσίαση των επιμέρους τμημάτων της, παρουσιάζεται στην παρακάτω εικόνα.



Εικόνα 26: Συνολική εικόνα της εφαρμογής

Για την κύρια κατανάλωση της υπηρεσίας έχει δημιουργηθεί μια διεπαφή χρήστη, η οποία τρέχει σε διαφορετικό εξυπηρετητή από ότι η υπηρεσία. Συγκεκριμένα τρέχει πάνω στον NGINX εξυπηρετητή, ενώ η διαδικτυακή υπηρεσία τρέχει στον Apache Tomcat εξυπηρετητή. Με την έννοια κύρια, εννοείται η κατανάλωση της εφαρμογής μέσω περιηγητή και όχι σε κάποιο πρόγραμμα. Για την κατανάλωση μέσω κάποιου προγράμματος Java έχουν δημιουργηθεί οι αντίστοιχες βιβλιοθήκες, που παρουσιάστηκαν σε προηγούμενη ενότητα. Όπως γίνεται αντιληπτό και από την Εικόνα 26, υπάρχουν τρία ξεχωριστά κομμάτια στην εφαρμογή. Το κομμάτι στην πλευρά του πελάτη (front end), το κομμάτι στην πλευρά του εξυπηρετητή (back end) και το κομμάτι της βάσης δεδομένων, όπου κάθε

κομμάτι ακολουθεί την δικιά του αρχιτεκτονική. Με την υπηρεσία που τρέχει στο παρασκήνιο και συγκεκριμένα στον Apache Tomcat, συνδέεται η μηχανή Prolog. Μέσω της διεπαφής JPL και ενός socket που ανοίγει μεταξύ Java και Prolog, τα αιτήματα που απαιτούν χρήση της Prolog, εφόσον διαμορφωθούν κατάλληλα, προωθούνται εκεί και με την σειρά της η Prolog, μέσω του δίαυλου επικοινωνίας, επιστρέφει τις απαντήσεις. Φίλτρα για την ασφάλεια της εφαρμογής και την ομαλή ροή πληροφοριών έχουν τοποθετηθεί και στην διεπαφή χρήστη (interceptors) και στην διαδικτυακή υπηρεσία (Spring security). Η διεπαφή χρήστη ακολουθεί μια Component based αρχιτεκτονική, όπου το κάθε component αποτελεί ένα κομμάτι του γενικού συνόλου, ενώ η διαδικτυακή υπηρεσία ακολουθεί μια αρχιτεκτονική επιπέδων συγκεκριμένης ροής πληροφοριών.

Με αυτή την ενότητα και την γενική εικόνα της εφαρμογής, ολοκληρώνεται η παρουσίαση όλων των επιμέρους κομματιών και επιπέδων αυτής. Στο επόμενο κεφάλαιο θα αναλυθεί ένα νέο ζήτημα που προκύπτει μετά το πέρας της δημιουργίας της εφαρμογής, το οποίο δεν είναι άλλο από την εγκατάστασή της.

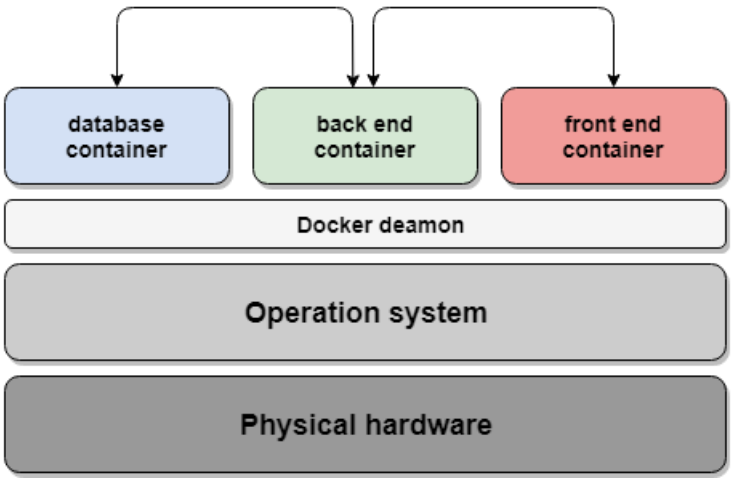
Κεφάλαιο 5 - Εγκατάσταση της εφαρμογής

5.1 Επιλογή τρόπου εγκατάστασης

Η ειδοποιός διαφορά μεταξύ εικονικής μηχανής και container είναι πως η εικονική μηχανή αναπαριστά εικονικοποίηση σε επίπεδο υλικού, ενώ το container αναπαριστά την εικονικοποίηση σε επίπεδο λειτουργικού συστήματος [22]. Αυτός είναι και ο λόγος που το καθιστά πιο ελαφρύ σε σχέση με τις εικονικές μηχανές, Χρησιμοποιώντας την τεχνολογία των containers υπάρχει κέρδος στην αποδοτικότητα και στους πόρους που καταναλώνονται σε σχέση με τους αντίστοιχους πόρους που καταναλώνει μια εικονική μηχανή. Για να ξεκινήσει μια εφαρμογή με την τεχνολογία των containers χρειάζεται αισθητά λιγότερο χρόνο από την αντίστοιχη σε μια εικονική μηχανή, εφόσον δεν χρειάζεται να εκκινείται εκ νέου όλο το λειτουργικό σύστημα, όπως απαιτείται στην εικονική μηχανή. Αντί αυτού το λειτουργικό σύστημα είναι ήδη ενεργό και η εφαρμογή μοιράζεται τους πόρους που χρειάζεται για να εκκινήσει. Ένα πλεονέκτημα της εικονικής μηχανής έναντι των containers είναι ότι παρέχει περισσότερη ασφάλεια. Σαρωτές ασφάλειας και διάφορα εργαλεία παρακολούθησης μπορούν να προστατεύσουν τον hypervisor και το λειτουργικό σύστημα της εικονικής μηχανής, αλλά όχι την εφαρμογή μέσα στο container. Αντίστοιχα ένα μειονέκτημα των containers είναι η έλλειψη απομόνωσης από το λειτουργικό που είναι εγκατεστημένο πάνω στο υλικό και για αυτό γίνεται αντιληπτό πως η πρόσβαση σε ολόκληρο το σύστημα είναι πιο εύκολη μέσω των containers. Αυτό όμως δεν συνεπάγεται ότι η τεχνολογία των containers δεν είναι μια ασφαλής τεχνολογία. Αντιθέτως, πρόκειται για μια πολύ ασφαλή τεχνολογία και σε συνδυασμό με τα πλεονεκτήματα έναντι της εικονικής μηχανής, προτιμάται πλέον από την παραγωγή.

5.2 Η αρχιτεκτονική της εγκατάστασης

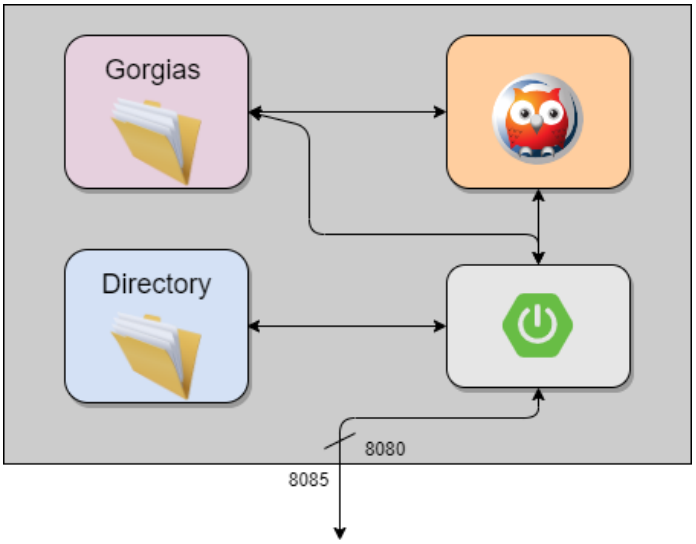
Για την εγκατάσταση της εφαρμογής στον εξυπηρετητή χρησιμοποιήθηκε η τεχνολογία των containers. Η συγκεκριμένη τεχνολογία αποτελεί την υφιστάμενη κατάσταση στο επίπεδο της εικονικοποίησης και προτιμήθηκε από την δημιουργία εικονικής μηχανής για τα πλεονεκτήματα που έχει έναντι αυτής και τα οποία αναλύθηκαν σε προηγούμενη ενότητα. Για την δημιουργία των containers και γενικά για όλο το στήσιμο της εφαρμογής στον εξυπηρετητή χρησιμοποιήθηκε η πλατφόρμα Docker. Με γνώμονα τα όσα ειπώθηκαν στη ενότητα παρουσίασης του εργαλείου Docker, η εικόνα δείχνει την αρχιτεκτονική της εγκατάστασης της εφαρμογής στον εξυπηρετητή.



Εικόνα 27: Αρχιτεκτονική εγκατάστασης της εφαρμογής

Όπως φαίνεται και στην Εικόνα 27, πάνω στο λειτουργικό σύστημα του εξυπηρετητή έχει εγκατασταθεί το Docker deamon. Έχουν δημιουργηθεί συνολικά τρεις εικόνες. Μια εικόνα για τη διαδικτυακή υπηρεσία, μια για την βάση δεδομένων και μια για τη διεπαφή χρήστη της εφαρμογής.

Για την δημιουργία μιας εικόνας, η μηχανή Docker διαβάζει ένα αρχείο, ονόματι Dockerfile, το οποίο περιέχει οδηγίες για το πως θα δημιουργηθεί η εικόνα. Για να γίνει ευκολότερα κατανοητό, ένα container μπορεί να παρομοιαστεί με έναν υπολογιστή με λειτουργικό σύστημα Linux. Σε αυτό τον υπολογιστή, υπάρχει η δυνατότητα διαμόρφωσής του ανάλογα με τις ανάγκες της εφαρμογής. Για το container της διαδικτυακής υπηρεσίας, που αποτελεί και τον πυρήνα της εφαρμογής, χρησιμοποιήθηκε η εικόνα του Ubuntu. Μέσα στο αρχείο που δημιουργεί την εικόνα προστέθηκαν εντολές για την εγκατάσταση της Java, του Apache Tomcat και της SWI-Prolog. Στον Apache Tomcat προστέθηκε το war[27] αρχείο, το αρχείο δηλαδή που περιέχει την υπηρεσία. Επιπρόσθετα, δημιουργήθηκε ένας φάκελος, όπου αποθηκεύονται τα αρχεία των χρηστών και τέλος εγκαταστάθηκαν και οι βιβλιοθήκες του Gorgias. Τα βήματα που απέμειναν για να ολοκληρωθεί το Dockerfile ήταν ο ορισμός μεταβλητών περιβάλλοντος για την σύνδεση της SWI-Prolog με την Java και ο ορισμός της θύρας που θα ακούει. Με την ολοκλήρωση του αρχείου και με την χρήση των εντολών ‘docker image build’ και ‘docker run’, δημιουργήθηκε το container του μέρους της εφαρμογής που τρέχει στον εξυπηρετητή. Θα μπορούσε να δημιουργηθεί ξεχωριστή εικόνα και container για την μηχανή Prolog, αλλά στην συγκεκριμένη περίπτωση δεν θα λειτουργούσε η διεπαφή JPL και αυτός είναι ο λόγος που στο container αυτό συμπεριλαμβάνεται και η Prolog μηχανή. Τα περιεχόμενα του container που αναλύθηκε παραπάνω παρουσιάζονται στην Εικόνα 28.



Εικόνα 28: Το container της διαδικτυακής υπηρεσίας

Το δεύτερο container περιέχει την εικόνα της MySQL, όπου έχει προστεθεί η βάση που δημιουργήθηκε για τις ανάγκες της εφαρμογής. Μέσα στο Dockerfile ορίζεται η θύρα 3306 ως η θύρα όπου ακούει το container. Στο μέλλον αν χρειαστεί να αλλάξει η βάση είτε λόγω διαφορετικών αναγκών, είτε για λόγους ταχύτητας, είτε για οποιονδήποτε άλλο λόγο, αυτό μπορεί να γίνει πολύ εύκολα δημιουργώντας ένα νέο container και συνδέοντάς το με τον container της εφαρμογής.

[27] War: Java Web Archive File

Η τρίτη και τελευταία εικόνα αφορά τη διεπαφή του χρήστη. Λόγω του ότι αυτό το κομμάτι δημιουργήθηκε με την βοήθεια της Angular, το συγκεκριμένο container περιέχει την εικόνα του NGINX εξυπηρετητή και επιπρόσθετα έναν φάκελο που περιλαμβάνει τη διεπαφή χρήστη της εφαρμογής. Ο NGINX εξυπηρετητής έχει διαμορφωθεί κατάλληλα, ώστε οποιοδήποτε αίτημα προς την υπηρεσία αφορά το Back end της εφαρμογής, να το προωθεί απευθείας εκεί.

5.3 Σύνδεση μεταξύ των Containers της εφαρμογής

Για την σύνδεση μεταξύ των containers της εφαρμογής χρησιμοποιήθηκε το Docker compose [24], ένα εργαλείο που καθορίζει και τρέχει εφαρμογές πολλαπλών container. Αυτό που επιτυγχάνει είναι να τρέχει πολλαπλά containers, κάνοντάς τα να λειτουργούν ως μια υπηρεσία. Με το Docker compose, χρησιμοποιήθηκε ένα YAML[28] αρχείο, όπου δηλώνονται όλες οι υπηρεσίες της εφαρμογής. Σε αυτό το αρχείο δηλώνονται επίσης μεταβλητές περιβάλλοντος, αντιστοιχίζονται πόρτες και γενικά περιέχει ότι χρειάζεται για να συνδεθούν τα containers το ένα με το άλλο. Έπειτα με μία και μόνο εντολή, τα δημιουργεί και ξεκινά να τρέχει την εφαρμογή με όλα τα κομμάτια της συνδεδεμένα, όπως αυτά καθορίζονται στο αρχείο.

Σε πιο τεχνικές λεπτομέρειες αρχικά ορίζεται η έκδοση του Docker compose εγγράφου. Εν συνεχεία δηλώνονται οι υπηρεσίες. Συγκεκριμένα, έχουν δηλωθεί τρεις υπηρεσίες, μία για την διεπαφή χρήστη, μία για την υπηρεσία στον εξυπηρετητή και μία για την βάση της εφαρμογής. Στο πεδίο image δηλώνεται η εικόνα που θα γίνει build, όπου, εφόσον η εικόνα δεν υπάρχει, παρέχεται επιπρόσθετα και το Dockerfile, ώστε να δημιουργηθεί με βάση αυτό. Στο πεδίο depends_on δηλώνεται η υπηρεσία στην οποία βασίζεται η κάθε υπηρεσία του YAML αρχείου. Συγκεκριμένα, η υπηρεσία της διεπαφής χρήστη βασίζεται στην υπηρεσία που τρέχει στον εξυπηρετητή και εκείνη με την σειρά της στην υπηρεσία της βάσης δεδομένων. Στο πεδίο ports δηλώνονται οι θύρες όπου ακούει και όπου αντιστοιχίζεται το κάθε container. Τέλος, στο πεδίο environment δηλώνονται μεταβλητές για την σύνδεση της βάσης, ενώ τα volumes αποτελούν τον μηχανισμό για την παρουσίαση και αποθήκευση δεδομένων. Με βάση τα παραπάνω ξεκινώντας το Docker compose, δημιουργείται ένα δίκτυο μεταξύ των υπηρεσιών που δηλώθηκαν.

5.4 Ασφάλεια δεδομένων

Εξ' ορισμού, όλα τα αρχεία που εγγράφονται μέσα σε ένα container, αποθηκεύονται σε ένα εγγράψιμο επίπεδο αυτού. Αυτό έχει ως αποτέλεσμα τα δεδομένα να υπάρχουν όσο υπάρχει το container, όπως επίσης και η μεταφορά τους έξω από αυτό να καθίσταται μία δύσκολη διαδικασία. Είναι φανερό πως η αποθήκευση των δεδομένων στο container είναι μια επιλογή που περιέχει μεγάλο ρίσκο.

Όπου υπάρχει ανάγκη στην εφαρμογή για αποθήκευση δεδομένων (αρχεία, δεδομένα βάσης, κλπ.), έχουν χρησιμοποιηθεί volumes. Τα volumes είναι μια επιλογή που παρέχει το Docker για την αποθήκευση δεδομένων τοπικά στο μηχάνημα. Με πιο απλά λόγια τα δεδομένα αντί να αποθηκεύονται στο container, αποθηκεύονται στο σύστημα αρχείων του μηχανήματος. Ως εκ τούτου αν το container για κάποιο λόγο σταματήσει, τα δεδομένα δεν χάνονται. Η εφαρμογή συγκεκριμένα χρησιμοποιεί δύο volumes, ένα για τα αρχεία των χρηστών και ένα για τις εγγραφές της βάσης και με αυτόν τον τρόπο εγγυάται την ασφάλειά τους.

[28] YAML: Γλώσσα περιγραφής δομής δεδομένων

Κεφάλαιο 6 - Αποτελέσματα εργασίας

Στα προηγούμενα κεφάλαια παρουσιάστηκε η αρχιτεκτονική που ακολούθησε κάθε κομμάτι της εφαρμογής, τα εργαλεία με τα οποία δημιουργήθηκαν αυτά, οι υπηρεσίες που παρέχει, αλλά και ο τρόπος με τον οποίο αυτή η εφαρμογή εγκαταστάθηκε και τρέχει. Στα δύο τελευταία κεφάλαια θα παρουσιαστούν τα αποτελέσματα της εργασίας, το τι επιτεύχθηκε αλλά και το πως μπορεί να εξελιχθεί η εφαρμογή στη συνέχεια.

6.1 Κατανάλωση υπηρεσίων σε JAVA πρόγραμμα μέσω βιβλιοθήκης

Το στιγμιότυπο οθόνης στην Εικόνα 29 παρουσιάζει τον τρόπο που ένα πρόγραμμα Java καταναλώνει την υπηρεσία. Εφόσον εισαχθεί η κατάλληλη βιβλιοθήκη στο πρόγραμμα, ο χρήστης χρησιμοποιεί την συνάρτηση login() με ορίσματα τα στοιχεία του για να πραγματοποιήσει είσοδο στην υπηρεσία. Εφόσον η είσοδος του είναι επιτυχημένη, μπορεί να χρησιμοποιήσει οποιαδήποτε συνάρτηση επιθυμεί, να φορτώσει ένα αρχείο στη βάση δεδομένων της Prolog και να κάνει ερωτήσεις προς αυτή, σε όποιο σημείο του κώδικα του προγράμματος επιθυμεί. Για την καλύτερη κατανόηση, στο συγκεκριμένο παράδειγμα χρησιμοποιούνται μόνο συναρτήσεις της βιβλιοθήκης σειριακά. Αρχικά, η συνάρτηση login χρησιμοποιείται για την είσοδο του χρήστη στην υπηρεσία. Εν συνεχεία, η συνάρτηση createProject δημιουργεί ένα νέο φάκελο και η μέθοδος addFile προσθέτει ένα Prolog αρχείο στον φάκελο αυτόν. Έπειτα, με την μέθοδο consult φορτώνεται ένα αρχείο στην βάση της Prolog και με την μέθοδο asserta προστίθεται ένας νέος κανόνας. Στο παράδειγμα της Εικόνας 30 χρησιμοποιείται και η μέθοδος prove της υπηρεσίας, η οποία υλοποιεί την εντολή prove του Gorgias. Συγκεκριμένα, χρησιμοποιείται μαζί με έναν iterator[29] ώστε να εκτυπωθούν όλα τα αποτελέσματα που επιστρέφει η μέθοδος prove. Τέλος, καλούνται οι μέθοδοι unload και logout για την αποφόρτωση του αρχείου από την βάση και την έξοδο του χρήστη από την υπηρεσία.

```
public void gorgiasViaJava() {
    WSClientImpl gorgias = new WSClientImpl();

    gorgias.login("tucuser", "tuc123");

    gorgias.createProject("gorgiasFolder");
    gorgias.addFile("/home/georgegl/Documents/prolog_docs/dina_v11.pl", "gorgiasFolder");
    gorgias.consult("dina_v11.pl","dina");
    gorgias.asserta("rule(f3000,currentYear(2005),[])");
    String _prove = "[selectFund(Fund)],Delta";

    Iterator<String> it = gorgias.prove(_prove, 10).iterator();
    while(it.hasNext()) {
        System.out.println(it.next());
    }
    gorgias.unload("dina_v11.pl","dina");

    gorgias.logout();
}
```

Εικόνα 29: Κατανάλωση διαδικτυακής υπηρεσίας μέσω προγράμματος

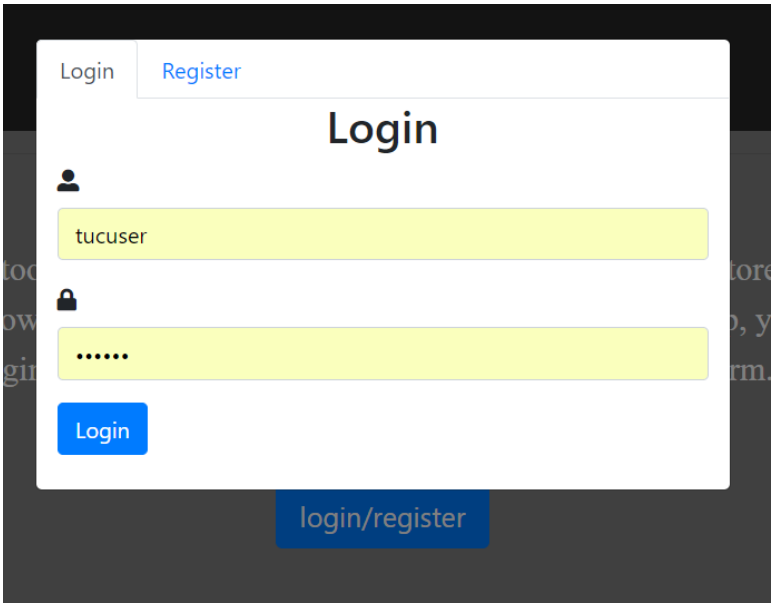
[29] Iterator: Διεπαφή της Java που ανήκει στο framework Collection και επιτρέπει την διάσχιση, πρόσβαση και διαγραφή δεδομένων όσων ανήκουν σε αυτό

6.2 Κατανάλωση υπηρεσιών της εφαρμογής μέσω περιηγητή

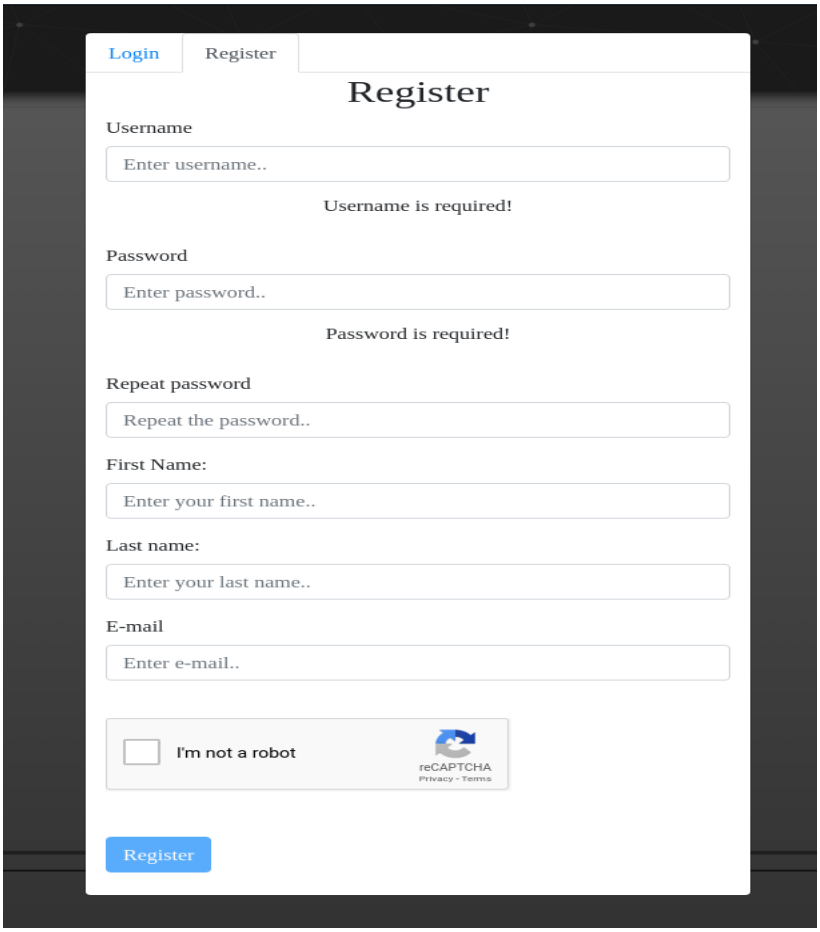
6.2.1 Λειτουργίες εφαρμογής για απλό χρήστη

Με την δημιουργία της διεπαφής χρήστη τα οφέλη είναι πολλαπλά. Αρχικά το εργαλείο Gorgias καθίσταται πιο προσιτό στον χρήστη, ο οποίος έχει έναν δικό του χώρο στο υπολογιστικό νέφος, τον οποίο μπορεί να διαχειριστεί όπως επιθυμεί. Δεν χρειάζεται να εγκαταστήσει απολύτως τίποτα για να φορτώσει τα αρχεία που επιθυμεί και πλέον έχει την δυνατότητα να τρέξει αρχεία Gorgias μέσω του περιηγητή. Η εφαρμογή αυτή βοηθάει και στην ανάπτυξη του εργαλείου Gorgias αυτού καθ'αυτού, εφόσον όποιος δεν είναι εξοικειωμένος με τους υπολογιστές δεν χρειάζεται να ασχοληθεί καθόλου με την εγκατάστασή του. Αρκεί να δημιουργήσει ένα νέο λογαριασμό και να ξεκινήσει να πειραματίζεται με το εργαλείο Gorgias. Στην Εικόνα 30 φαίνεται η σελίδα που υποδέχεται τον χρήστη. Η εφαρμογή απαιτεί εγγραφή στην υπηρεσία για την χρησιμοποίησή της και για αυτό τον λόγο στην αρχική οθόνη ο χρήστης έχει μόνο μία επιλογή. Η επιλογή αυτή είναι το μπλε κουμπί, όπου πατώντας το έχει την δυνατότητα να πραγματοποιήσει είσοδο στην εφαρμογή ή να εγγραφεί σε αυτή.

Εικόνα 30: Αρχική σελίδα της διεπαφής χρήστη



Εικόνα 31: Modal για την είσοδο χρήστη



Εικόνα 32: Modal για την εγγραφή νέου χρήστη

Πατώντας το μπλε κουμπί ανοίγει ένα modal δύο επιλογών που παρέχει την δυνατότητα στον χρήστη για είσοδο ή εγγραφή στην εφαρμογή. Στην Εικόνα 31 απεικονίζεται η πρώτη επιλογή και πιο συγκεκριμένα η είσοδος στην εφαρμογή. Ο χρήστης παραχωρεί τα στοιχεία του και εφόσον επαληθευτούν, εισέρχεται στην εφαρμογή.

Στην Εικόνα 32 απεικονίζεται η φόρμα εγγραφής νέου χρήστη, αλλά και οι έλεγχοι που πραγματοποιούνται κατά την εγγραφή του. Σε περίπτωση που το username υπάρχει ήδη ή οι κωδικοί πρόσβασης δεν ταιριάζουν, παρουσιάζονται οι αντίστοιχες προειδοποιήσεις και δεν επιτρέπεται η εγγραφή. Παρόμοιοι έλεγχοι εκτελούνται για κενά πεδία, αλλά και για διπλότυπα email και σε αυτά τα σημεία υπάρχει ασύγχρονη επικοινωνία της διεπαφής με τον εξυπηρετητή.

Στην Εικόνα 33, φαίνεται η παρουσίαση των φακέλων του χρήστη μέσα από την εφαρμογή. Το εικονίδιο στην πάνω δεξιά πλευρά της οθόνης χρησιμεύει για την δημιουργία νέου φακέλου. Μόλις ο χρήστης το επιλέξει, ανοίγει ένα modal, όπου ο χρήστης έχει την δυνατότητα να δημιουργήσει έναν νέο φάκελο στο προσωπικό χώρο που έχει στην εφαρμογή. Παρόμοιο modal ανοίγει και για να ανεβάσει ένα νέο αρχείο ο χρήστης. Να τονιστεί πως σε αυτό το σημείο πραγματοποιείται ένας ακόμα έλεγχος. Ελέγχεται αν το αρχείο που επιθυμεί να ανεβάσει ο χρήστης είναι ένα αρχείο Prolog. Για την ακρίβεια, στις επιλογές του χρήστη εμφανίζονται μόνο Prolog αρχεία.

.

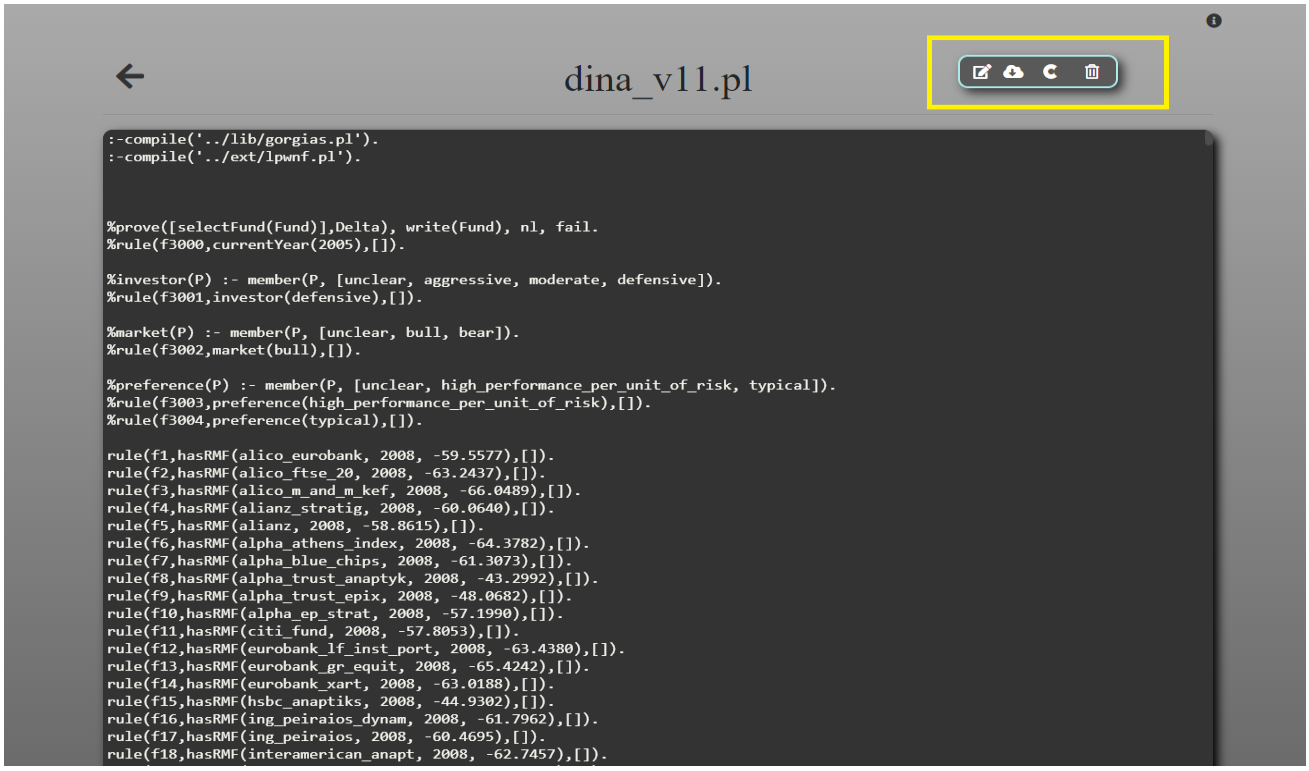
Εικόνα 33: Περιήγηση στους φακέλους χρήστη

Εικόνα 34: Αρχεία φάκελου και επιλογές χρήστη

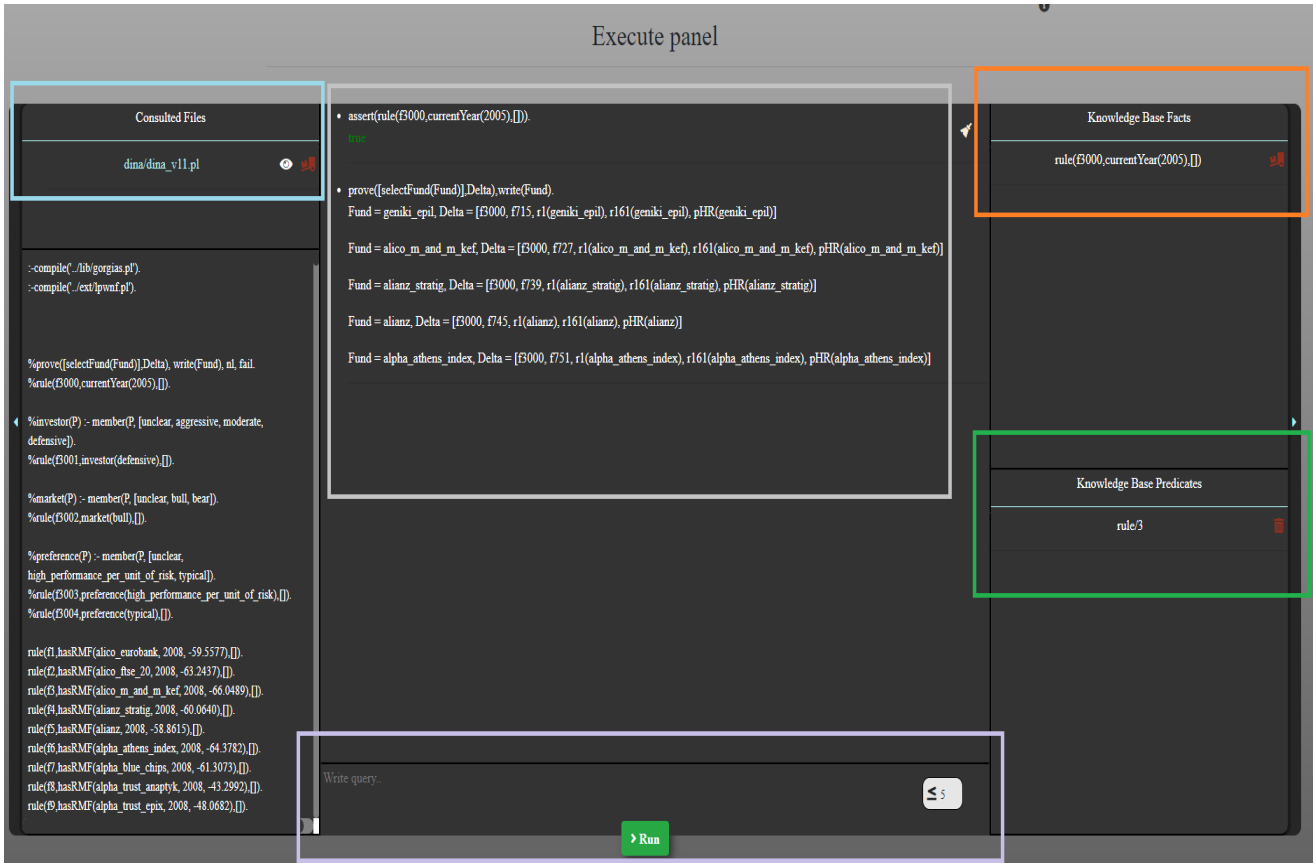
Στην Εικόνα 34, ο χρήστης έχει μπει στον φάκελο dina, όπου στην συγκεκριμένη περίπτωση έχει μόνο ένα αρχείο. Στο ροζ τετράγωνο απεικονίζονται οι επιλογές που παρέχονται στον χρήστη. Ο χρήστης μπορεί να προσθέσει ή να δημιουργήσει νέο αρχείο σε αυτόν τον φάκελο ή να διαγράψει το αρχείο ή και ολόκληρο τον φάκελο, πατώντας στο κατάλληλο εικονίδιο. Επιλέγοντας το εικονίδιο με το βελάκι πάνω αριστερά, οδηγείται πίσω στην σελίδα με όλους τους φακέλους του συνολικά. Σε περίπτωση που επιλέξει το εικονίδιο της διαγραφής ή αυτό της προσθήκης αρχείου, ένα ακόμα modal ανοίγει, το οποίο ρωτάει τον χρήστη αν όντως επιθυμεί την διαγραφή του αρχείου ή αντίστοιχα του δίνει την δυνατότητα να επιλέξει αρχείο για προσθήκη. Αν επιλέξει το εικονίδιο + ωστόσο, μεταφέρεται στον κειμενογράφο της εφαρμογής, όπου εκεί μπορεί να δημιουργήσει το δικό του prolog αρχείο.

Εφόσον ο χρήστης επιλέξει κάποιο αρχείο εμφανίζεται το περιεχόμενό του (βλ. Εικόνα 35). Επιπρόσθετα, πέρα από την επιλογή να ξαναγυρίσει στην σελίδα με τα αρχεία του φακέλου, έχει άλλες τέσσερις επιλογές οι οποίες τονίζονται στο κίτρινο περίγραμμα της εικόνας. Η πρώτη είναι να επιλέξει να επεξεργαστεί το αρχείο και η δεύτερη να το κατεβάσει τοπικά. Έχει επίσης την δυνατότητα να το φορτώσει στην μηχανή Prolog επιλέγοντας το κουμπί C (consult). Αν επιλέξει κάτι τέτοιο, το αρχείο φορτώνεται στη βάση της Prolog και ο χρήστης ανακατευθύνεται στο πάνελ. Τέλος, επιλέγοντας την διαγραφή του αρχείου, ανοίγει ένα modal, το οποίο ρωτάει τον χρήστη για επιβεβαίωση της διαγραφής.

Η Εικόνα 36 δείχνει το πάνελ της εφαρμογής. Στο κεντρικό γκρι τετράγωνο βρίσκεται η περιοχή που απεικονίζονται οι εντολές που δόθηκαν, με την απόκριση της Prolog μηχανής να τοποθετείται από κάτω τους. Αριστερά στο μπλε τετράγωνο φαίνονται όλα τα αρχεία που έχει φορτώσει ο χρήστης. Κάθε αρχείο έχει δύο εικονίδια δεξιά του. Πατώντας το πρώτο, βλέπει τα περιεχόμενα του αρχείου ακριβώς από κάτω, ενώ με το δεύτερο, εξάγει το αρχείο από την βάση της Prolog. Δεξιά στο πορτοκαλί τετράγωνο υπάρχουν όλα τα γεγονότα που έχει προσθέσει και πατώντας στο εικονίδιο που βρίσκεται δεξιά από κάθε γεγονός, το απενεργοποιεί. Στο πράσινο τετράγωνο φαίνονται όλα τα κατηγορήματα που έχουν προστεθεί και η δυνατότητα να διαγράψει ένα κατηγόρημα από την βάση. Στο μοβ τετράγωνο τέλος, απεικονίζεται η περιοχή όπου ο χρήστης γράφει την εντολή, επιλέγει τον αριθμό απαντήσεων που επιθυμεί να λάβει και την τρέχει.



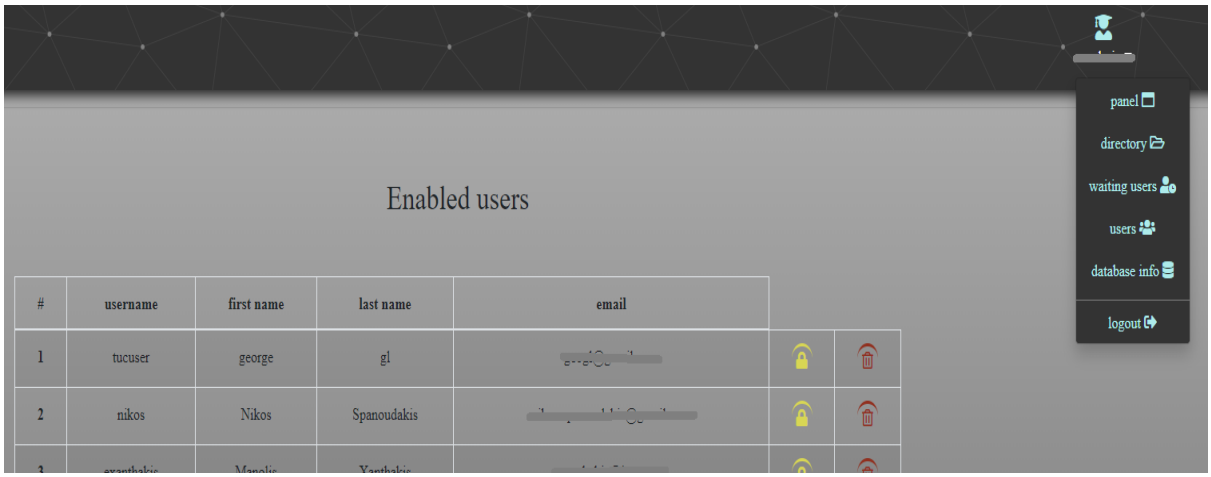
Εικόνα 35: Απεικόνιση περιεχομένου ενός αρχείου του χρήστη



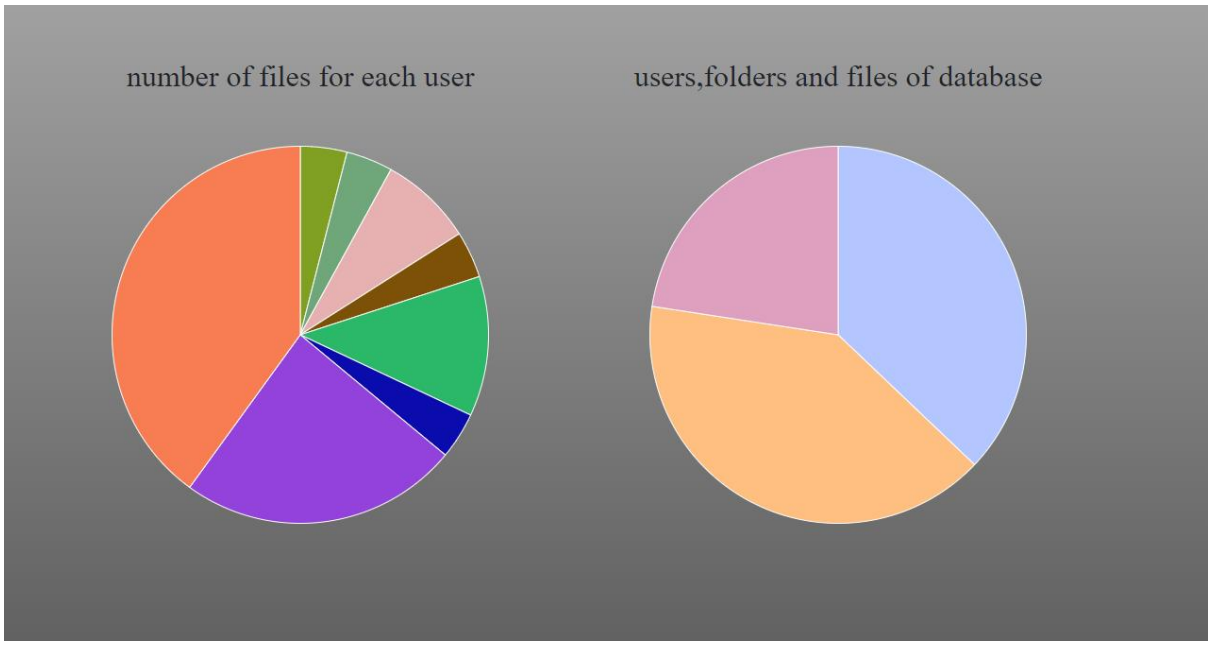
Εικόνα 36: Κονσόλα εντολών και αποτελεσμάτων της εφαρμογής

6.2.2 Λειτουργίες εφαρμογής για τον διαχειριστή

Ο διαχειριστής της εφαρμογής λαμβάνει κάποιες πρόσθετες λειτουργιές. Πιο συγκεκριμένα έχει ξεχωριστή σελίδα μέσα στην διεπαφή χρήστη, όπου του παρέχεται η δυνατότητα να δει γραφικά με την κατάσταση της βάσης δεδομένων (αριθμός αρχείων, φάκελων και χρηστών) και έναν πίνακα με όλους τους χρήστες, όπου μπορεί να κλειδώσει ή και να διαγράψει έναν χρήστη. Τέλος, υπάρχει ένας ακόμα πίνακας όπου επεικονίζονται όλοι οι χρήστες που περιμένουν έγκριση για να χρησιμοποιήσουν την εφαρμογή. Μέσα από αυτόν τον πίνακα αποδέχεται ή απορρίπτει τα αίτηματα των χρηστών.



Εικόνα 37: Πίνακας εγγεγραμένων χρηστών και επιλογές διαχειριστή



Εικόνα 38: Γραφικά στατιστικά της βάσης δεδομένων

Κεφάλαιο 7 - Σύνοψη και μελλοντική εργασία

Σκοπός της εργασίας ήταν η δημιουργία μιας εφαρμογής, η οποία θα καταστήσει το εργαλείο Gorgias περισσότερο προσιτό σε νέους χρήστες. Επιπρόσθετα, θα παρέχει έναν αποθηκευτικό χώρο για οποιοδήποτε χρήστη της εφαρμογής και συνολικά ένα ακόμα βήμα στην ανάπτυξη του Gorgias, αλλά και γενικότερα της Prolog, εφόσον υποστηρίζει το μεγαλύτερο φάσμα των εντολών της. Με βάση όλα όσα αναφέρθηκαν, μέσα από την εφαρμογή παρέχεται η δυνατότητα ανάπτυξης λογικού προγραμματισμού και εξαγωγής δεδομένων και συμπερασμάτων μέσα από συλλογιστική πορεία, στο υπολογιστικό νέφος.

Στις προηγούμενες ενότητες του κεφαλαίου παρουσιάστηκαν τα αποτελέσματα της εργασίας και σίγουρα υπάρχουν οι δυνατότητες για περαιτέρω ανάπτυξή της. Είναι ουσιαστικά μια βάση πάνω στην οποία μπορούν να αναπτυχθούν πολλές ακόμα λειτουργίες, τόσο στην υπηρεσία που τρέχει στο εξυπηρετητή, όσο και στην διεπαφή χρήστη. Η φόρτωση αρχείου χωρίς την ανάγκη αποθήκευσής του, η χρησιμοποίηση Prolog χωρίς να είναι απαραίτητη η εγγραφή στην υπηρεσία και η υποστήριξη περισσότερων εντολών της, είναι μερικές από αυτές. Η κατεύθυνση που μπορεί να ακολουθήσει η εξέλιξη της εφαρμογής δεν περιορίζεται σε κάτι συγκεκριμένο. Από την μια πλευρά υπάρχει η δυνατότητα ανάπτυξης του φάσματος Prolog προγραμμάτων που θα υποστηρίζει, με σκοπό να συμπεριλάβει και άλλους τεχνολογικούς τομείς, όπως εκείνον του σημασιολογικού ιστού. Από την άλλη, υπάρχει η δυνατότα δημιουργίας και άλλων containers με σκοπό την υποστήριξη νέων γλωσσών προγραμματισμού και την εξέλιξη της εφαρμογής σε ένα περιβάλλον ανάπτυξης λογισμικού στο διαδίκτυο.

Όσο αφορά την δομή, η εφαρμογή αποτελείται από τρία τμήματα. Η διαδικτυακή υπηρεσία έχει αναπτυχθεί σε Java με την βοήθεια του Spring framework. Η βάση δεδομένων της εφαρμογής είναι μια MySQL βάση δεδομένων και το κομμάτι που αλληλεπιδρά με τον χρήστη είναι γραμμένο σε Angular 7. Τα τρία αυτά κομμάτια έχουν εγκατασταθεί με τέτοιο τρόπο, ώστε η αντικατάστασή τους να είναι εύκολα εφικτή και η επεκτασιμότητά της εφαρμογής αρκετά υψηλή. Η δομή της εφαρμογής είναι τέτοια που επιτρέπει οι αλλαγές να εφαρμόζονται με σχετική ευκολία. Ο λόγος που συμβαίνει κάτι τέτοιο είναι ότι αποτελείται από κομμάτια-υπηρεσίες που συνδέονται μεταξύ δημιουργώντας ένα δίκτυο και όχι από ένα μεγάλο κομμάτι με όλες τις υπηρεσίες μέσα σε αυτό.

Με την προσθήκη νέων λειτουργιών και όσο η εφαρμογή μεγαλώνει, θα ήταν δυνατό να υιοθετηθεί μία microservices[30] αρχιτεκτονική, όπου πολλές υπηρεσίες θα επικοινωνούσαν μεταξύ τους για να παράξουν την τελική απόκριση. Συγκεκριμένα επίσης κομμάτια και υπηρεσίες της εφαρμογής, όπως η αποστολή emails κατά την εγγραφή νέου χρήστη, θα μπορούσαν να αναπτυχθούν με ένα serverless[31] μοτίβο.

Εν κατακλείδι, η εργασία αυτή αποτελεί μια βάση πάνω στην οποία μπορούν να δημιουργηθούν πολλά και ενδιαφέροντα πράγματα και να αναπτυχθούν νέες λειτουργίες. Ως εκ τούτου, η συγκεκριμένη εφαρμογή, όπως περιγράφηκε αναλυτικά στα προηγούμενα κεφάλαια, αποτελεί μια εφαρμογή που προσφέρει έδαφος για μελλοντική εργασία και ανάπτυξη.

[30] Microservices: Αρχιτεκτονικό στυλ όπου η εφαρμογή αποτελείται από ένα σύνολο υπηρεσιών

[31] Serverless: Μοντέλο υπολογιστικού νέφους όπου ο πάροχος διαχειρίζεται τον εξυπηρετητή και κατανέμει δυναμικά τους πόρους.

Βιβλιογραφία

[1] N. I. Spanoudakis, E. Karafili, A. C. Kakas, E. C. Lupu (2017, May 1).
Argumentation-based security for social good.

[2] Gorgias-B Argumentation tool.
http://gorgiasb.tuc.gr/

[3] Ευάγγελος Κουράκος Μαυρομιχάλης (2006).
Εισαγωγή στη Γλώσσα Προγραμματισμού Prolog.
http://www.icsd.aegean.gr/lecturers/stamatatos/courses/Logic/Prolog/Ch2/ch2_1.html

[4] JPL API.
https://jpl7.org/index

[5] SWI-Prolog.
https://www.swi-prolog.org/

[6] Tau-Prolog.
http://tau-prolog.org/documentation.

[7] R. J. Rafaels, Createspace Independent Publishing (2015, April 1).
Cloud Computing: From Beginning to End Platform.

[8] Upwork Staff (2017, April 19). SOAP vs REST: A look at two different API styles.
https://www.upwork.com/hiring/development/soap-vs-rest-comparing-two-apis/

[9] Kakas Antonis, Moraitis Pavlos, Spanoudakis Nikolaos (2018, December 6).
Gorgias: Applying argumentation.
https://content.iospress.com/articles/argument-and-computation/aac181006

[10] C. Walls, Manning (2018, November 5).
Spring in Action, Fifth Edition.

[11] J. Ottinger, J. Linwood και D. Minter, Apress (2014, April 3).
Beginning Hibernate, Third Edition.

[12] MVC Framework - TutorialsPoint.
https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.html

[13] Angular Official Website.
https://angular.io/

[14] Nwose Lotanna (2019, July 18). Understanding RxJS Observables and why you need them.
https://blog.logrocket.com/understanding-rxjs-observables/

[15] VMWare. Virtual Machine.
https://www.vmware.com/topics/glossary/content/virtual-machine

[16] Docker. What is a Container.
https://www.docker.com/resources/what-container

[17] J. Nickoloff, Manning (2019, December 10).
Docker in Action, Second Edition.

[18] Docker.
https://docs.docker.com/

[19] Spring Security.
https://spring.io/projects/spring-security

[20] Baeldung (2020, July 7). The guide to RestTemplate.
https://www.baeldung.com/rest-template

[21] A. Freeman, Apress (2018, October 10).
Pro Angular 6, Third Edition.

[22] PhoenixNAP (2019, April 15). Containers vs Virtual Machines: What's the difference?
https://phoenixnap.com/kb/containers-vs-vms

[23] Overview of Docker Compose.
https://docs.docker.com/compose/

[24] Apache Maven Project.
https://maven.apache.org/