ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

ΣΗΜΑΣΙΟΛΟΓΙΚΗ ΔΙΑΧΕΙΡΙΣΗ ΠΡΟΤΥΠΩΝ ΠΕΡΙΓΡΑΦΗΣ ΜΙΚΡΟΫΠΗΡΕΣΙΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΠΥΡΙΔΩΝ Ι. ΠΑΠΠΑΣ

Επιβλέπων: Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Οκτώβριος 2020

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

ΣΗΜΑΣΙΟΛΟΓΙΚΗ ΔΙΑΧΕΙΡΙΣΗ ΠΡΟΤΥΠΩΝ ΠΕΡΙΓΡΑΦΗΣ ΜΙΚΡΟΫΠΗΡΕΣΙΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΠΥΡΙΔΩΝ Ι. ΠΑΠΠΑΣ

Επιβλέπων: Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 20η Οκτωβρίου 2020.

…….	…….	…….
Εμμανουήλ Βαρβαρίγος	Θεοδώρα Βαρβαρίγου	Συμεών Παπαβασιλείου
Καθηγητής Ε.Μ.Π.	Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2020

……
ΣΠΥΡΙΔΩΝ Ι. ΠΑΠΠΑΣ
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Ο σκοπός και το αντικείμενο της παρούσας διπλωματικής εργασίας είναι ο σχεδιασμός και η ανάπτυξη μιας εφαρμογής στο πλαίσιο του DITAS project, που να παρέχει στον χρήστη τη δυνατότητα της σημασιολογικής αναζήτησης (semantic search). Σχεδιάστηκαν τεχνικές εμπλουτισμού και βαθμολόγησης μέσω των οποίων διαφοροποιείται η σημασιολογική από άλλες μορφές αναζήτησης. Οι τεχνικές αυτές αναλύθηκαν εκτενώς με σκοπό τη βελτιστοποίηση της αναζήτησης. Παράλλλα, υλοποιήθηκε και μια απλή αναζήτηση (simple search) για τη συγκριτική μελέτη της ταχύτητας εκτέλεσης, του πλήθους και της ποιότητας των αποτελεσμάτων. Η ιδέα για το θέμα προέκυψε από την ανάγκη για ποιοτικότερη ανίχνευση πληροφορίας και την αξιοποίησή της μέσω της τεχνολογίας των Δεδομένων ως Υπηρεσία (Data as a Service).

Η υλοποίηση της REST-API εφαρμογής πραγματοποιήθηκε μέσω ενός Spring-Boot standalone microservice. Ο χρήστης εισάγοντας κάποια ορίσματα, λαμβάνει ως αποτελέσματα της αναζήτησης blueprints, που ανακτώνται από μια MongoDb βάση δεδομένων. Η σημασιολογική αναζήτηση εμπλουτίζει τα αποτελέσματα με τη χρήση μιας οντολογίας, η οποία αξιοποιείται μέσω του framework Jena και είναι σχεδιασμένη με την OWL2 γλώσσα. Στόχος είναι η εφαρμογή να είναι ανεξάρτητη του είδους των δεδομένων που επιστρέφει. Εν προκειμένω, τα στοιχεία των blueprints που χρησιμοποιήθηκαν αφορούν ιατρικά δεδομένα. Δημιουργήθηκε επίσης, ένα front-end για εξυπηρέτηση της εφαρμογής με χρήση HTML και Javascript. Τέλος, συγκρίθηκαν τα δύο είδη αναζήτησης όσον αφορά την ταχύτητά τους και την ποσότητα και ποιότητα των blueprints που επιστράφηκαν.

Με την υλοποίηση της εφαρμογής ο χρήστης κατέχει τη δυνατότητα επιλογής της απλής ή της σημασιολογικής αναζήτησης, ενώ καθώς αφήνονται ανοικτά περιθώρια ανάπτυξης και εξέλιξης, παρέχονται ακόμα περισσότερες δυνατότητες στον τομέα αυτό.

Λέξεις Κλειδιά:
DITAS Project, REST, API, σημασιολογική, αναζήτηση, OWL2, MongoDb, Spring-Boot, μικροϋπηρεσία, Δεδομένα ως Υπηρεσία, DaaS, οντολογία, Jena, Java

Abstract

The object of this diploma thesis is the design and development of an application within the DITAS project, which gives the user the capability of the Semantic Search. The semantic search differentiates itself from other types of search because of its enriching and grading techniques. These techniques were analyzed thoroughly in order to optimize the search. Furthermore, a simple search was implemented for the comparative study of the execution time, as well as the quantity and the quality of the search results. The idea for this issue has emerged from the need to accurately search for data and utilize them through the Data as a Service technology.

The implementation of the REST-API application was accomplished using a Spring-Boot standalone microservice. The user entering as an input some arguments, receives as the search's results blueprints that are recovered from a MongoDb database. The semantic search enriches the recovered results through an ontology, which is used by the Jena framework and designed with the OWL2 language. The goal is for the application to be independent to the type of the data returned. The elements of the blueprints in use have to do with medical data. A front-end was also established to serve the application using HTML and Javascript. Finally, the two types of search were compared in terms of their speed and the quantity and quality of blueprints returned.

With the implementation of the application, the user now has the ability to choose between using the simple or the semantic search, while leaving room for growth and development also even more possibilities in this area are provided.

Keywords:

DITAS Project, REST, API, semantic, search, OWL2, MongoDb, Spring-Boot, microservice, Data as a Service, DaaS, ontology, Jena, Java

Ευχαριστίες

Με την παρούσα διπλωματική εργασία ολοκληρώνεται η ακαδημαϊκή μου πορεία στη Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου, η οποία δε θα μπορούσε να διεκπεραιωθεί χωρίς τη βοήθεια των ανθρώπων με τους οποίους συνεργάστηκα και οι οποίοι με στήριξαν σε αυτό το έργο.

Αρχικά, θα ήθελα να ευχαριστήσω την κα. Θεοδώρα Βαρβαρίγου, καθηγήτρια Ε.Μ.Π για την εμπιστοσύνη που μου έδειξε και την ανάθεση μίας πολύ ενδιαφέρουσας διπλωματικής εργασίας, καθώς και τον κ. Αχιλλέα Μαρινάκη, υποψήφιο Διδάκτορα Ε.Μ.Π για τη βοήθεια και τον χρόνο που διέθεσαν καθοδηγώντας με σε όλα τα στάδια της εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου για την όλη στήριξη κατά τη διάρκεια της φοίτησης μου στο πανεπιστήμιο, καθώς και τους φίλους και τους συμφοιτητές μου, που ήταν παρόντες τόσο στα ωραία όσο και στα δύσκολα αυτής της πορείας.

Πίνακας Περιεχομένων

Contents

Πίνακας Εικόνων

Πίνακας Πινάκων

1. ΤΕΧΝΟΛΟΓΙΕΣ - ΕΡΓΑΛΕΙΑ

1.1.1. Javascript

Η Javascript είναι μια ελαφριά, interpreted γλώσσα προγραμματισμού με αντικειμενοστραφείς ιδιότητες. Ο πυρήνας της γλώσσας είναι ενσωματωμένος στη Netscape, τον Internet Explorer, καθώς και σε άλλους περιηγητές ιστού (web browsers) και είναι εμπλουτισμένος με αντικείμενα (objects), που αντιπροσωπεύουν παράθυρα του περιηγητή ιστού και τα περιεχόμενά του. Η client-side πλευρά της Javascript επιτρέπει οι ιστοσελίδες να περιλαμβάνουν εκτελέσιμο περιεχόμενο, με αποτέλεσμα να μη χρειάζεται να χτίζονται με στατική HTML (HyperText Markup Language), αλλά να μπορούν να αλληλεπιδρούν με τον χρήστη, να ελέγχουν τον περιηγητή και δυναμικά να δημιουργούν HTML περιεχόμενο [1] .

Το επίσημο όνομα της Javascript είναι ECMAScript. Λόγω πνευματικών δικαιωμάτων χρειάστηκε να αλλάξει όνομα και σήμερα, η Mozilla είναι από τις λίγες εταιρείες που επιτρέπεται επισήμως να χρησιμοποιεί το όνομα Javascript. Για κοινή χρήση υπάρχουν οι εξής κανόνες :

- Με τον όρο “Javascript” αναφερόμαστε στην γλώσσα προγραμματισμού.
- Με τον όρο “ECMAScript” αναφερόμαστε στην επίσημη προδιαγραφή της γλώσσας. Άρα όταν κάποιος αναφέρεται στις εκδόσεις της γλώσσας, τις ονομάζει ECMAScript. Για παράδειγμα “Η τωρινή έκδοση της Javascript είναι η ECMAScript 5”.[2]

1.1.2. Java

Η Java είναι μια γενικού σκοπού, class-based, αντικειμενοστραφής γλώσσα προγραμματισμού. Σχετίζεται με την C και την C++ αλλά είναι αρκετά διαφορετική, παραβλέποντας διάφορα στοιχεία της C και της C++ και συνδυάζοντας στοιχεία από άλλες γλώσσες. Είναι μια γλώσσα υψηλού επιπέδου και διαθέτει αυτόματη διαχείριση του αποθηκευτικού χώρου (χρησιμοποιώντας κυρίως τον garbage collector) για να αποφύγει πιθανά προβλήματα ασφαλείας. Σχεδιασμένη ως εμπορική γλώσσα και όχι ως ερευνητικού ενδιαφέροντος γλώσσα, ο σχεδιασμός της δεν περιέχει μη δοκιμασμένα χαρακτηριστικά [41] .

Το Java Virtual Machine είναι το στοιχείο υπεύθυνο για την ανεξαρτησία της Java από το hardware και το σύστημα, για το μικρό μέγεθος του compiled κώδικα και για τη δυνατότητα να προστατεύει τους χρήστες από κακόβουλα λογισμικά. Είναι ένα αφηρημένο υπολογιστικό μηχάνημα, το οποίο όπως και ένα αληθινό περιέχει δικό του σύνολο οδηγιών (instruction set) και διαχειρίζεται

διαφορετικά μέρη της μνήμης κατά τη διάρκεια της εκτέλεσης. Το JVM δεν γνωρίζει τίποτα σχετικά με τη γλώσσα προγραμματισμού Java, παρά μόνο μια συγκεκριμένη δυαδική μορφή (format), τη μορφή του αρχείου κλάσης [3] [1] .

Το JDK της Oracle (Java Development Kit) περιλαμβάνει έναν διερμηνέα (compiler) και ένα σύστημα run-time που υλοποιεί το JVM καθαυτό [4] . Από την έκδοση της Java 11 και μετά, το JRE (Java Runtime Environment) δεν προσφέρεται [4] .
Στα πλαίσια της παρούσας διπλωματικής χρησιμοποιήθηκε Java 11.

1.1.2.1. Annotations

Τα annotations, μια μορφή metadata, προσδίδουν πληροφορία σχετικά με ένα πρόγραμμα που δεν είναι μέρος του προγράμματος καθαυτού. Δεν έχουν άμεση επίδραση στη λειτουργία του κώδικα που σχολιάζουν.

Έχουν μια πληθώρα εφαρμογών, μεταξύ των οποίων:

- Πληροφορία για τον μεταγλωτιστή (compiler): Τα annotations μπορεί να χρησιμοποιηθούν από τον compiler για να ανιχνεύσουν σφάλματα ή να καταστείλουν warnings.
- Επεξεργασία κατά τη διάρκεια μεταγλώτισης και ανάπτυξης (deployment): Εργαλεία λογισμικού μπορούν να επεξεργαστούν annotation πληροφορία για να δημιουργήσουν κώδικα, XML αρχεία κ.α.
- Επεξεργασία κατά τη διάρκεια εκτέλεσης: Κάποια annotations μπορούν να εξεταστούν κατά τη διάρκεια της εκτέλεσης. [6]

1.1.2.2. Java beans

Τα Java beans ορίζονται ως :
"Ένα Java Bean είναι ένα επαναχρησιμοποιήσιμο στοιχείο λογισμικού το οποίο μπορεί να υποστεί οπτική επεξεργασία με χρήση εργαλείου δημιουργίας"
Τα builder tools (εργαλεία δημιουργίας) μπορεί να είναι builders ιστοσελίδων, εφαρμογών, GUI, ή ακόμα και εφαρμογών εξυπηρετητών.

Τα Java Beans μπορεί να υποστηρίζουν ποικιλόμορφη λειτουργικότητα. αλλά γενικά τα στοιχεία που τα χαρακτηρίζουν είναι :

- Υποστήριξη για "introspection", ώστε ένα builder tool να μπορεί να αναλύσει πως λειτουργεί το bean.
- Υποστήριξη για "customization", ώστε όταν χρησιμοποιείται ένας application builder, ο χρήστης να μπορεί να προσαρμόσει την εμφάνιση και συμπεριφορά ενός bean.
- Υποστήριξη για "events", ως ένα απλό μέσο επικοινωνίας που μπορεί να χρησιμοποιηθεί για τη σύνδεση των beans.
- Υποστήριξη για "properties", και για παραμετροποίηση και για προγραμματιστική χρήση.

- Υποστήριξη για ”persistence”, ώστε ένα bean να μπορεί να είναι παραμετροποιήσιμο στον application builder και να μπορεί να έχει την παραμετροποιήσιμη κατάστασή του αποθηκευμένη και αργότερα να επαναφορτωθεί.

Τα τρία πιο σημαντικά χαρακτηριστικά των Java Beans είναι το σύνολο των *properties* που εκθέτουν, το σύνολο των *methods* που αφήνουν άλλα components να καλούν και το σύνολο των *events* που ενεργοποιεί. [5]

1.1.3. HTML

Ο Παγκόσμιος Ιστός (Word Wide Web) είναι ένα δίκτυο από πηγές πληροφοριών. Το Δίκτυο βασίζεται σε τρεις μηχανισμούς, ώστε να έχει αυτές τις πηγές διαθέσιμες στο ευρύτερο δυνατό κοινό:

1. Ένα ομοιόμορφο σύστημα ονοματολογίας για την εύρεση των πηγών στο Δίκτυο (π.χ. URIs).
2. Πρωτόκολλα, για πρόσβαση σε ονοματισμένες πηγές (named resources) στο Δίκτυο (π.χ. HTTP).
3. Υπερκείμενο (hypertext), για εύκολη πλοήγηση στις πηγές, π.χ. HTML (Hypertext Markup Language) [7] .

Η HTML δίνει στους συντάκτες τη δυνατότητα να δημοσιεύουν online έγγραφα με κεφαλίδες, κείμενο, πίνακες, λίστες, φωτογραφίες, να περιλαμβάνουν βίντεο, ήχο, άλλες εφαρμογές ή να ανακτούν πληροφορία online μέσω συνδέσμων υπερκειμένου (hypertext links). Προσφέρει επίσης πληθώρα δυνατοτήτων, όπως ο σχεδιασμός φορμών για συναλλαγές με απομακρυσμένες υπηρεσίες, και η αναζήτηση πληροφορίας [7] .

1.1.4. HTTP

Σύμφωνα με το OSI reference model, το Hypertext Transfer Protocol (HTTP) είναι ένα πρωτόκολλο στρώματος εφαρμογής για πληροφοριακά συστήματα τα οποία είναι κατανεμημένα, συνεργατικά και χρησιμοποιούν υπερμέσα (γραφικά, ήχο, βίντεο, απλό κείμενο και υπερσυνδέσμους). Δεν συγκρατεί κάποια πληροφορία (stateless) και είναι ένα γενικό πρωτόκολλο που μπορεί να χρησιμοποιηθεί για διάφορες εργασίες πέρα από τη χρήση του στο hypertext. Ένα χαρακτηριστικό του HTTP είναι η τυπολόγηση και η διαπραγμάτευση της απεικόνισης των δεδομένων, επιτρέποντας στα συστήματα να χτιστούν ανεξάρτητα από τα δεδομένα που μεταφέρονται. Χρησιμοποιείται από τον Παγκόσμιο Ιστό (Word-Wide Web) από το 1990 [8] .

1.1.5. OWL2

Ο Σημασιολογικός Ιστός (Semantic Web) αποτελεί ένα όραμα για το μέλλον του Ίντερνετ, που η πληροφορία που είναι διαθέσιμη σε αυτό δέχεται σαφή ερμηνεία, κάνοντάς το ευκολότερο για τις μηχανές να την επεξεργαστούν και να τη χρησιμοποιήσουν αυτοματοποιημένα. Στηρίζεται στη δυνατότητα της XML (Extensive Markup Language) να ορίζει προσαρμόσιμα πεδία (tags) και της RDF (ή RDFS Resource Description Framework Schema) να παρουσιάζει δεδομένα [9] .

Μια οντολογία είναι μια θεωρία η οποία χρησιμοποιεί συγκεκριμένο λεξιλόγιο για να περιγράψει entities, classes, properties και related functions από κάποια οπτική γωνία. Οι οντολογίες επιτρέπουν στον υπολογιστή να κατανοήσει την πληροφορία αυτούσια. Ερευνητές έχουν αναπτύξει γλώσσες οντολογιών για αναπαράσταση της γνώσης (knowledge), εργαλεία κατασκευής οντολογιών και reasoners για να κάνουν infer τις οντολογίες. Ανάμεσα στις γλώσσες οντολογιών, η OWL και συγκεκριμένα η τελευταία έκδοσή της, OWL 2, είναι η πιο δημοφιλής.

Η OWL 2 Web Ontology Language, ή OWL 2, είναι μια γλώσσα οντολογιών για το Semantic Web με ρητά ορισμένη σημασιολογία. Οι οντολογίες της OWL 2 περιλαμβάνουν κλάσεις (classes), ιδιότητες (properties), άτομα (individuals) και τιμές δεδομένων (data values) και αποθηκεύονται σαν αρχεία Σημασιολογικού Ιστού. Αυτές οι οντολογίες μπορούν να χρησιμοποιηθούν μαζί με πληροφορία διατυπωμένη σε RDF, όπως επίσης οι οντολογίες καθαυτές μπορούν αν ερμηνευθούν ως RDF αρχεία [10] .

1.1.5.1. Reasoner

Ο reasoner είναι ένα πρόγραμμα το οποίο εξάγει λογικά επακόλουθα από ένα σύνολο ρητά καθορισμένων facts ή αξιωμάτων και γενικά προσφέρει αυτοματοποιημένη υποστήριξη για reasoning λειτουργίες, όπως classification, debugging και querying.

Η ποιότητα και η ορθότητα των οντολογιών διαδραματίζει ζωτικό ρόλο στη σημασιολογική αναπαράσταση και στη διάδοση της γνώσης (knowledge). Για τη διασφάλιση της ποιότητας των οντολογιών υπάρχει ανάγκη αντιμετώπισης του inconsistency και του uncertainty στις οντολογίες σε real-world εφαρμογές. Μια inconsistent οντολογία υποδηλώνει την ύπαρξη σφάλματος ή μιας αντίφασης στην οντολογία και ως αποτέλεσμα κάποιες έννοιες δεν μπορούν να διερμηνευτούν σωστά. Το inconsistency έχει ως αποτέλεσμα ψευδή σημασιολογική κατανόηση και αναπαράσταση γνώσης. Το reasoning των οντολογιών μειώνει το redundancy των πληροφοριών στην knowledge base και βρίσκει τις αντιφάσεις στο περιεχόμενο του knowledge.

Ανάμεσα στον μεγάλο αριθμό των διαθέσιμων reasoners, αυτοί οι οποίοι υποστηρίζονται από το Protégé ή το NeOn είναι : Pellet, RACER, FACT++, Snorocket, HermiT, CEL, ELK and SWRL-IQ, TrOWL [10] .

1.1.5.2. Pellet

Στα πλαίσια της παρούσας διπλωματικής εργασίας, θα χρησιμοποιηθεί ο Pellet reasoner. Ο Pellet είναι ένας open source reasoner της Java βασισμένος στο OWL-DL και σχεδιασμένος από το "The Mind Swap" group. Είναι βασισμένος στον tableu αλγόριθμο και υποστηρίζει εκφραστικές περιγραφικές λογικές. Μπορεί να χρησιμοποιηθεί σε οντολογίες μέσω του Jena όπως και του OW-API interface [40] . Επίσης, ο Pellet υποστηρίζει τη διευκρίνηση των bugs. [11]

1.1.6. REST-API

Το REST-API (Representational State transfer) είναι ένα API (Application Programming Interface) με βάση το REST, όπως ορίστηκε από τον Roy Thomas Fielding. Αυτό καθορίζει μια σειρά κανόνων σχετικά με την ανάπτυξη εφαρμογών Web. :

- Client-Server: διαχωρισμός της υπηρεσίας REST η οποία τρέχει σε έναν κεντρικό διακομιστή (server) και του client.
- Stateless: ο διακομιστής δεν διατηρεί κάποια πληροφορία ως προς την κατάσταση της σύνδεσης, αλλά επεξεργάζεται κάθε αίτηση το ίδιο.
- Cache: τα δεδομένα πρέπει να δηλώνονται είτε άμεσα είτε έμμεσα ως προς τη δυνατότητά τους να αποθηκευτούν ή όχι προσωρινά. Στην περίπτωση που διαθέτουν τη δυνατότητα αυτή, ο χρήστης μπορεί να τα ξαναχρησιμοποιήσει χωρίς να επαναλάβει αίτημα στον διακομιστή.
- Uniform Interface: Οι υλοποιήσεις διαχωρίζονται από τις υπηρεσίες που παρέχουν, έχοντας ένα καθολικό και γενικό Interface, το οποίο ενθαρρύνει τη δυνατότητα επέκτασης.
- Layered system: Το ιεραρχικό σύστημα με διάφορα στρώματα αποκρύπτει σε κάθε στρώμα την πληροφορία που βρίσκεται σε άλλα στρώματα.
- Code on demand: Ο διακομιστής μπορεί να αλλάξει ή να αναβαθμίσει τη λειτουργικότητά του με καινούρια έκδοση της εφαρμογής.

Τέλος, γενικά μια διαδικασία σε REST API ορίζεται από το όνομά της και μία HTTP μέθοδο (όπως GET, POST ή DELETE). Το γεγονός ότι μια εφαρμογή είναι Web εφαρμογή, δε σημαίνει απαραιτήτως πως ακολουθεί το αρχιτεκτονικό μοντέλο των REST-API [12] .

Η εφαρμογή στα πλαίσια της παρούσας διπλωματικής εργασίας ακολουθεί αυτό το μοντέλο, καθώς εξυπηρετείται με HTTP GET αιτήματα τα οποία είναι stateless, έχει uniformed interface, είναι ιεραρχική, υπακούει στο μοντέλο χρήστη/πελάτη - διακομιστή, μπορεί να αναβαθμιστεί on-demand και τα δεδομένα γενικά δεν αποθηκεύονται προσωρινά.

1.1.7. Data as a Service (DaaS)

Η διαχείριση και η διάθεση δεδομένων διαδραματίζουν σημαντικό ρόλο στη δομή των μοντέρνων cloud-fog αρχιτεκτονικών. Χωρίς μια αυστηρή, γρήγορη και ντετερμινιστική μέθοδο ανταλλαγής δεδομένων δεν μπορούμε να είμαστε σίγουροι για την απόδοση και την ποιότητα των συναλλαγών και των εφαρμογών [25] . Το πρόβλημα αυτό καλείται να λύσει μια σχετικά νέα τεχνολογία, τα "Δεδομένα ως Υπηρεσία".

Τα Δεδομένα ως Υπηρεσία, αλλιώς DaaS (Data as a Service), όπως όλα τα μέλη της οικογένειας "as a Service", βασίζεται στην ιδέα πως το προϊόν (δεδομένα στην προκείμενη) μπορεί να δοθεί στον χρήστη όταν το ζητήσει, ανεξάρτητα από γεωγραφικούς ή διοικητικούς διαχωρισμούς προμηθευτή και καταναλωτή. Επιπρόσθετα, η εμφάνιση της service-oriented αρχιτεκτονικής κάνει την πλατφόρμα στην οποία τα δεδομένα υπάρχουν να μην έχει σημασία. [23]

Η πρωτοβουλία του DaaS είναι μια επένδυση στο να εδραιωθούν και να οργανωθούν τα παραγωγικά δεδομένα σε ένα μέρος και να είναι διαθέσιμα να υποστηρίξουν νέες, αλλά και υπάρχοντες ψηφιακές προτάσεις. Το DaaS γίνεται ένα καινοτόμο σύστημα με το να εκθέτει τα δεδομένα ως ένα δια-επιχειρησιακό στοιχείο. Τέλος, απελευθερώνει τα δεδομένα από legacy συστήματα και τα ωθεί σε νέες εφαρμογές και ψηφιακά συστήματα, χωρίς την ανάγκη να διαταράσονται υπάρχοντα back-ends.[24]

1.1.7.1. Blueprint (VDC)

Στα πλαίσια του Ditas Project και συγκεκριμένα στην υλοποίηση του Data as a Service, η περιγραφή μιας υπηρεσίας, με σκοπό να είναι διαθέσιμη σε όλους του πιθανούς αγοραστές, γίνεται μέσω του VDC Blueprint, το οποίο δημιουργείται και δημοσιεύεται από τους κατόχους των δεδομένων. Το VDC (Virtual Data Container) ως μεσολαβητής δίνει τη δυνατότητα στους καταναλωτές των δεδομένων απλώς να ορίσουν τα requirements των ζητούμενων δεδομένων και είναι υπεύθυνο για την παροχή τους ενώ αποκρύπτει την πολυπλοκότητα της υποκείμενης δομής. Το Blueprint που αντιστοιχεί σε ένα VDC είναι ένα JSON αρχείο που περιγράφει όλες τις λεπτομέρειες της υλοποίησης (και άλλα στοιχεία όπως διαθεσιμότητα, τιμή κ.λπ.). [25]

1.1.7.2. JSON

Το JSON (JavaScript Object Noattion) είναι ένας μορφότυπος, ο οποίος χρησιμοποιείται για ανταλλαγή δεδομένων μεταξύ όλων των γλωσσών προγραμματισμού. Παρουσιάστηκε αρχικά, το 2001 στο website json.org. Ο ορισμός του JSON επαναδημοσιεύτηκε ως "IETF RFC 4627" ([13]) τον Ιούλιο του 2006. Είναι εμπνευσμένο από την Javascript (ECMAScript) όπως έχει οριστεί στο ECMAScript Language Specification.

Αποτελείται από ένα συντακτικό με άγκιστρα, αγκύλες, άνω και κάτω τελείες και κόμματα, το οποίο είναι χρήσιμο σε διάφορα πλαίσια και εφαρμογές. Συγκεκριμένα, αποτελείται από δύο βασικές

δομές δεδομένων, τα αντικείμενα και τους πίνακες. Τα αντικείμενα αποτελούνται από ζεύγη ονομάτων και τιμών, ενώ οι πίνακες αποτελούνται από ένα σύνολο τιμών. Η τιμή ενός αντικειμένου μπορεί να είναι κείμενο (String), αριθμός, αληθές/ψευδές (boolean) ή και αντικείμενο ή πίνακας [14]
.

1.1.8. MongoDB

Η MongoDB είναι μια ισχυρή και ευέλικτη βάση δεδομένων γενικού σκοπού. Συνδυάζει τη δυνατότητα επέκτασης σε στοιχεία, όπως δευτερεύοντες δείκτες (indexes), range queries, ταξινόμηση, aggregations και geospatial δείκτες.

Η MongoDB δεν είναι μια σχεσιακή βάση δεδομένων, αλλά ανήκει στην οικογένεια των Document Databases. Μια document-oriented βάση δεδομένων αντικαθιστά την έννοια της γραμμής (εγγραφή), με ένα πιο ευέλικτο μοντέλο, το έγγραφο (document). Περιλαμβάνοντας ενσωματωμένα έγγραφα και πίνακες, η προσέγγιση αυτή επιτρέπει πολύπλοκες ιεραρχικές δομές με μία εγγραφή. [15] Αυτές οι βάσεις δεδομένων προσφέρουν το πλεονέκτημα του να αποθηκεύουν όλη την πληροφορία σχετικά με ένα object σε ένα document, χρησιμοποιώντας εμφωλευμένα documents και arrays. Αυτό βελτιώνει την απόδοση, μειώνοντας τον αριθμό των queries που χρειάζονται για να ανακτηθούν όλα τα αιτηθέντα δεδομένα και εξαλείφοντας την ανάγκη για ένωση πινάκων. [25]

Επίσης, δεν υπάρχει στη βάση κάποιο προκαθορισμένο σχήμα. Τα κλειδιά και οι τιμές ενός αρχείου δεν έχουν σταθερό μέγεθος ή τύπο. Χωρίς προκαθορισμένο σχήμα, η προσθήκη ή η αφαίρεση πεδίων ανάλογα με τις ανάγκες της εφαρμογής γίνεται ευκολότερα, χωρίς να χρειαστεί να αλλάξει κάποιος πίνακας. Αυτό κατά συνέπεια καθιστά αποδοτικότερη τη διαδικασία της παραγωγής, διευκολύνοντας τον πειραματισμό και τις δοκιμές [15] .

Στα πλαίσια της παρούσας εργασίας, καθώς τα blueprints που θα επιστρέφονται δεν είναι καθορισμένου μεγέθους, επιλέχθηκε η MongoDb ούσα document-oriented.

1.1.9. Spring

Το Sping Framework είναι ένα open source framework, βασισμένο στη γλώσσα προγραμματισμού Java. Αποτελεί μια χαμηλού φόρτου λύση για τη δημιουργία επιχειρησιακών εφαρμογών. Παρ'όλα αυτά, η διάρθρωσή του, του επιτρέπει τη χρήση μόνο των κομματιών του που χρειάζονται χωρίς τα υπόλοιπα. Ανάμεσα σε αυτά που προσφέρει είναι το model-view-controller (MVC) και η δυνατότητα για Aspect Oriented Programming (AOP). [16]

1.1.9.1. Spring Boot

Το framework του SpringBoot επιτρέπει αυτόματη παραμετροποίηση, δίνοντας τη δυνατότητα στο Spring, έξυπνα, να ανιχνεύσει το είδος της εφαρμογής που δημιουργείται και αυτόματα να

καθορίσει τις απαραίτητες συνιστώσες για να υποστηριχθούν οι ανάγκες της εφαρμογής. Είναι δηλαδή, σχεδιασμένο ώστε να απλοποιεί την ανάπτυξη των εφαρμογών, το οποίο επιτυγχάνει μέσω δυνατοτήτων που προσφέρει, όπως το auto-configuration [17] . Ανάμεσα στα ισχυρότερα προτερήματα του Spring Boot είναι:

- Δημιουργεί stand-alone Spring εφαρμογές.
- Έχει ενσωματωμένους Tomcat, Jetty ή Undertow (δεν υπάρχει ανάγκη για deployment WAR αρχείων).
- Προσφέρει "'starter' dependencies" για να απλοποιήσει τη δημιουργία του configuration.
- Παραμετροποιεί αυτόματα το Spring και 3rd party βιβλιοθήκες (όπου είναι δυνατόν).
- Προσφέρει χαρακτηριστικά έτοιμα για χρήση στην παραγωγή όπως metrics, health checks και externalized παραμετροποίηση.
- Δεν δημιουργεί κώδικα και δεν έχει προαπαιτούμενη XML παραμετροποίηση. [18]

1.1.9.2. Spring Boot Microservice

Τα microservice είναι μια μοντέρνα προσέγγιση λογισμικού, που ο κώδικας της εφαρμογής παραδίδεται σε μικρά, διαχειρίσιμα κομμάτια, ανεξάρτητα από άλλα. Η μικρή τους κλίμακα και η σχετική απομόνωση μπορούν να οδηγήσουν σε πολλά πρόσθετα οφέλη, όπως ευκολότερη συντήρηση, αυξημένη παραγωγικότητα, μεγαλύτερη ανοχή σε σφάλματα [19] .

Στα πλαίσια της παρούσας εργασίας, η εφαρμογή θα υλοποιηθεί ως ένα microservice. Έτσι, αφού ο κώδικας μεταγλωττιστεί, δημιουργείται ένα jar αρχείο που αυτούσιο περιέχει όλη την λειτουργικότητα της εφαρμογής και δεν χρειάζεται να εκτελεστεί σε κάποιον application server.

1.1.10. Apache Jena

Το Apache Jena είναι ένα ελεύθερο, ανοιχτού λογισμικού framework της Java για τον σχεδιασμό εφαρμογών Σημασιολογικού Ιστού και Διασυνδεδεμένων Δεδομένων (Linked Data).
Το Framework περιλαμβάνει:

- Ένα API (Application Programming Interface) για ανάγνωση, επεξεργασία και εγγραφή RDF δεδομένων σε XML, N-Triples και Turtle.
- Ένα API οντολογιών για τη διαχείριση οντολογιών OWL και RDFS.
- Μια διεπαφή, η οποία θα βασίζεται σε κανόνες για συλλογιστική με δεδομένα OWL και RDF.
- Στρατηγικές αποθήκευσης που αποσκοπούν στην αποτελεσματική αποθήκευση μεγάλου αριθμού RDF triplets στη μνήμη ή στον σκληρό δίσκο.
- Διακομιστές που μπορούν να δημοσιεύουν RDF δεδομένα σε άλλες εφαρμογές .

Συγκεκριμένα στην παρούσα εργασία χρησιμοποιείται το Jena Ontology API. Αυτό το εργαλείο, προσφέρει ένα σταθερό interface για ανάπτυξη εφαρμογών με οντολογίες, ανεξάρτητα από τη γλώσσα οντολογίας που χρησιμοποιείται, από την πιο εκφραστική OWL Full, μέχρι και τη λιγότερο RDFS [20] .

1.1.11. Apache JMeter

Το Apache JMeter είναι μια ελεύθερη, ανοιχτού λογισμικού εφαρμογή σχεδιασμένη για λειτουργικά test σε μεγάλο φόρτο της εφαρμογής καθώς και για να μετράει τις επιδόσεις της εφαρμογής. Είχε αρχικά, σχεδιαστεί για test σε εφαρμογές Web, αλλά έχει επεκταθεί και σε άλλα είδη test.

Μπορεί να χρησιμοποιηθεί για να προσομοιώσει μεγάλο φόρτο σε έναν εξυπηρετητή (server), ένα group από servers, ή και σε κάποιο δίκτυο για να ελέγξει τις αντοχές του ή να αναλύσει τη συνολική του επίδοση κάτω από διαφορετικό φόρτο αιτημάτων [21] .

Στα πλαίσια της παρούσας διπλωματικής χρησιμοποιήθηκε για τη διεξαγωγή μεγάλου όγκου τεστ με κύρια τεστ τα performance.

1.1.12. Protégé

Το Protégé είναι ένα ελεύθερο, ανοιχτού λογισμικού framework και editor για οντολογίες, για τη δημιουργία έξυπνων συστημάτων.

Υποστηρίζεται από μια κοινότητα ακαδημαϊκών, κυβερνητικών και εταιρικών χρηστών, οι οποίοι το χρησιμοποιούν για να δημιουργήσουν λύσεις βασιμένες στα δεδομένα, σε ποικίλα πεδία , όπως η βιοϊατρική και το ηλεκτρονικό εμπόριο. Το Protégé υποστηρίζει την OWL 2 [22] .

Στα πλαίσια της παρούσας διπλωματικής χρησιμοποιήθηκε για τον σχεδιασμό της οντολογίας.

2. ΑΡΧΙΤΕΚΤΟΝΙΚΗ

2.1. Εισαγωγή

Για να εκμεταλλευτεί κανείς στο έπακρον δεδομένα που παράγονται από οποιονδήποτε κλάδο, υπάρχει ανάγκη για βελτιστοποίηση της διαδικασίας συλλογής, αποθήκευσης, πρόσβασης, χρήσης και, τελικά, παράδοσής τους στον τελικό χρήστη. Τα Δεδομένα ως Υπηρεσία (Data as a Service) στοχεύουν στο να δημιουργήσουν ένα μοντέλο που βασίζεται σε υπηρεσίες, το οποίο θα καλύψει όλο τον κύκλο ζωής των δεδομένων μέσω κατανεμημένων διασυνδεδεμένων υπηρεσιών και θα αντιμετωπίσει τα προαναφερθέντα ζητούμενα της διαδικασίας.

Οι βελτιώσεις που προτείνονται από το γενικό πλαίσιο του Data as a Service Marketplace είναι:

- Content discovery: Ένα σημασιολογικά ενισχυμένο στοιχείο για εύρεση περιεχομένου που βοηθάει αυτόν που αγοράζει τα δεδομένα να βρεί τα καταλληλότερα όσον αφορά το περιεχόμενο. Επίσης, βοηθά τον προμηθευτή δεδομένων, να τα περιγράψει καλύτερα για να γίνει πιο εύκολο να βρεθούν.
- QoS evaluation: Ένα εργαλείο αξιολόγησης που παράγει στατιστικές αναλύσεις και μετρικές Ποιότητας Υπηρεσιών (Quality of Service) για τις υπηρεσίες που προμηθεύουν τα δεδομένα, για να βοηθήσουν τον αγοραστή όχι μόνο να αξιολογήσει την Ποιότητα των Δεδομένων, αλλά και την Ποιότητα των Υπηρεσιών με την οποία τα δεδομένα διατίθενται.
- DaaS repository scalability: Ένα στοιχείο επεκτασιμότητας που επιτρέπει τη δυναμική επέκταση του repository του DaaS Marketplace, για να διασφαλίζεται η επιχειρησιακή συνέχεια και ανοχή στα σφάλματα.

Η παρούσα εργασία εστιάζει στις βελτιστοποιήσεις στον τομέα του Content discovery.

Για τον λόγο αυτό, υλοποιείται μια σημασιολογική αναζήτηση. Η σημασιολογική αναζήτηση δέχεται ως είσοδο στοιχεία του χρήστη και επιστρέφει αποτελέσματα που να έχουν συνάφεια με αυτά. Αυτό το πετυχαίνει εμπλουτίζοντας τα εισαχθέντα στοιχεία με άλλα σχετικά. Όλες αυτές οι σχέσεις μεταξύ των στοιχείων πραγματοποιούνται με τη συμβολή μιας οντολογίας. Τέλος, αφού ανακτηθούν όλα τα σχετικά αποτελέσματα, ταξινομούνται βάσει της σχετικότητάς τους και παρουσιάζονται στον χρήστη.

Επιπρόσθετα, εκτός της υλοποίησης του semantic search, για συγκριτικούς λόγους υλοποιήθηκε και μια απλή αναζήτηση (simple search). Η semantic και η simple αναζητήσεις συγκρίνονται και τα συμπεράσματα όσον αφορά την ποιότητα και ποσότητα των αποτελεσμάτων είναι εμφανή. Η simple search ασφαλώς είναι η απλούστερη μορφή αναζήτησης που θα μπορούσε να υλοποιηθεί και δεν συγκρίνεται με άλλες, όπως για παράδειγμα την elastic search.

2.2. Σχεδιασμός

Η υλοποίηση μιας πλατφόρμας σημασιολογικής αναζήτησης εξετάζει διάφορες αρχιτεκτονικές επιλογές.

1. Αρχικά, η αναζήτηση θα ακολουθεί το μοτίβο του stand-alone search engine, όπως εξηγείται στο [39] . Τα document που αναζητεί ο χρήστης (blueprints στην προκείμενη), βρίσκονται εκτός της μηχανής αναζήτησης σε μια βάση δεδομένων και η semantic search δεν καλεί κάποια άλλη μηχανή αναζήτησης για να τα αποκτήσει (τα document), αλλά απλώς εκτελεί query στη βάση.
2. Τα ίδια τα document δεν είναι άρρηκτα συνδεδεμένα με κάποια υπάρχουσα οντολογία (ή το αντίστροφο). Αντιθέτως, προσφέρεται η ευελιξία να σχεδιαστεί εκ νέου μια νέα οντολογία με βάση τις ανάγκες της αναζήτησης και τα δεδομένα που θα πρέπει να επιστρέφει.
3. Η διαδικασία της αναζήτησης αποκρύπτεται από τον χρήστη, καθώς δεν υπάρχει διαφορετική συμπεριφορά από την εφαρμογή προς τον χρήστη ανάλογα με το είδος αναζήτησης που θέλει να πραγματοποιήσει.
4. Η δομή της οντολογίας περιλαμβάνει έναν απλό σκελετό που δημιουργεί σχέσεις μεταξύ των στοιχείων. Δεν έχουν οριστεί properties στα στοιχεία της οντολογίας, αλλά έχουν οριστεί σχέσεις υπο-κλάσης (αντίστοιχα υπερ-κλάσης) μεταξύ τους.
5. Η επαύξηση (Εμπλουτισμός Στοιχείων προς αναζήτηση) γίνεται στατικά. Η εφαρμογή με τη βοήθεια της οντολογίας προσπελαύνει τις hard-coded σχέσεις των εισαχθέντων στοιχείων και δημιουργεί μια λίστα με όλα τα στοιχεία που θα αξιοποιηθούν στην αναζήτηση της βάσης.
6. Η αναζήτηση στη βάση είναι επίσης hard-coded. Τα εισαχθέντα στοιχεία μαζί με τα στοιχεία εμπλουτισμού αναζητώνται όλα μαζί και οποιοδήποτε document περιέχει έστω και ένα από αυτά, επιστρέφεται.
7. Η διαδικασία βαθμολόγησης (Βαθμολόγηση στοιχείων και blueprints) επιτρέπει παραμετροποίηση όσον αφορά τις μετρικές. Κάθε στοιχείο έχει έναν δικό του βαθμό ανάλογα με τη σημαντικότητά του ο οποίος παραμετροποιείται.

2.3. Πλοήγηση

Με βάση τα παραπάνω, υλοποιήθηκε η εφαρμογή της παρούσας εργασίας. Η εφαρμογή ακολουθεί την αρχιτεκτονική του REST-API και είναι υλοποιημένη ως ένα Spring-Boot microservice. Επίσης, ο χρήστης μπορεί να χρησιμοποιήσει την εφαρμογή είτε με ένα HTTP GET request, είτε από το υλοποιημένο front-end.

Ο χρήστης πραγματοποιεί μια αναζήτηση με βάση κάποια ορίσματα (tags) και επιθυμεί να του επιστραφεί ένα σύνολο από blueprints ταξινομημένα ως προς τη συνάφειά τους. Τα blueprints αποτελούνται από tags. Στόχος της υλοποίησης είναι όταν ο χρήστης βάζει ως είσοδο κάποια tags, να

μην του επιστρέφονται μόνο blueprints που τα περιέχουν αυτούσια, αλλά και blueprints που περιέχουν tags σχετικά με αυτά που εισήγαγε ο χρήστης, ταξινομημένα με βάση το πόσο ακριβή είναι. Για παράδειγμα, σε μια αναζήτηση με tags *Doctor*, *Smoker*, το blueprint [Doctor, Smoker, Pregnancy] έχει υψηλότερη βαθμολογία (μεγαλύτερη συνάφεια) από το [Doctor, Asian, Female] και άρα θα εμφανιστεί υψηλότερα λόγω της ταξινόμησης.

Επίσης, για τη συγκριτική μελέτη της σημασιολογικής αναζήτησης υλοποιήθηκε και η απλή αναζήτηση και ένα front-end στο οποίο μπορεί ο χρήστης να τις αξιοποιήσει αμφότερες.

Συμπερασματικά :

Η **simple αναζήτηση** επιστρέφει τα blueprints που περιέχουν μόνο τα tags που έχουν εισαχθεί. Αντίστοιχα, και η ταξινόμηση που συμβαίνει στην προκειμένη περίπτωση αφορά μόνο το πλήθος των tags που περιέχει κάθε blueprint, δηλαδή, ένα blueprint που περιέχει 2 tags από αυτά που εισήγαγε ο χρήστης έχει μικρότερο βαθμό από ένα που περιέχει 3 από τα tags.

Για την υλοποίηση της **semantic αναζήτησης** σχεδιάστηκε μια οντολογία η οποία ορίζει τις σχέσεις μεταξύ των στοιχείων που δομούν τα blueprints. Οι σχέσεις αυτές είναι σχέσεις μεταξύ κλάσεων (κλάση-υποκλάση-υπερκλάση) όπως και individuals που ανήκουν σε κάποια κλάση. Για παράδειγμα, το *Doctor* είναι υπο-κλάση του στοιχείου *Person*, όπως και το *FamilyDoctor* είναι υποκλάση του πεδίου *Doctor*. Έτσι, τα αρχικά, tags που εισήγαγε ο χρήστης εμπλουτίζονται με άλλα μέσω των σχέσεων που προκύπτουν από την οντολογία.

Η εύρεση αυτών των επιπρόσθετων tags γίνεται ακολουθώντας μια σειρά κανόνων. Έτσι, για παράδειγμα αν ο χρήστης αναζητήσει το tag *Doctor*, θα του επιστραφούν και blueprint που περιλαμβάνουν και το tag *Person* (ως υπερ-κλάση του *Doctor*). Διευρύνεται δηλαδή, το πλήθος των εισαχθέντων tags με άλλα tags τα οποία έχουν κάποια σχέση με τα εισαχθέντα (όπως διευρύνεται και το πλήθος των επιστρεφόμενων blueprints).

Η semantic αναζήτηση πέρα από τη διεύρυνση των όρων πρέπει να υλοποιεί και το σύστημα ταξινόμησης-βαθμολόγησης που αναφέρθηκε προηγουμένως. Για παράδειγμα, ο *Neurologist* έχει superclass την κλάση *Specialist*, που έχει superclass την κλάση *Doctor*. Μια αναζήτηση με tag το *Neurologist* θα πρέπει να επιστρέφει blueprints που περιλαμβάνουν ως στοιχείο το *Neurologist* με υψηλότερη βαθμολογία από αυτά που περιέχουν τον *Specialist,* που θα έχουν επίσης υψηλότερη βαθμολογία από αυτά που περιέχουν τον *Doctor*.

Τέλος, πέρα από τη συνάφεια των tags σημαντικό ρόλο στη βαθμολόγηση ενός blueprint κατέχει ο αριθμός των στοιχείων του. Όταν ο χρήστης εισάγει 5 tags, είναι πιο πιθανό να περιμένει ένα blueprint με 5 μερικά ταιριάσματα απ'ότι ένα blueprint με 2 ακριβή ταιριάσματα.

Η semantic αναζήτηση εν τέλει:

- **Εμπλουτίζει** τα tags με τα οποία γίνεται αναζήτηση στη βάση των blueprints με γνώμονα την οντολογία.
- Εμπλουτίζει **ιεραρχικά**. Σε κάθε tag είτε το εισήγαγε ο χρήστης, είτε προέκυψε μέσω της οντολογίας αντιστοιχίζεται ένας βαθμός.

- **Βαθμολογεί** το κάθε blueprint. Η βαθμολογία αυτή προκύπτει μέσω μιας φόρμουλας που προσμετράει τον βαθμό του κάθε tag ξεχωριστά, αλλά και το πλήθος των ταιριασμάτων.

Στο κεφάλαιο 3. ΑΝΑΛΥΣΗ γίνεται η θεωρητική ανάλυση πίσω από τα βασικά κομμάτια της υλοποίησης. Συγκεκριμένα, μελετάται η διαδικασία του εμπλουτισμού των tags (3.1. Εμπλουτισμός Στοιχείων προς αναζήτηση), όπως και η βαθμολογική ιεραρχία τους και η προκύπτουσα φόρμουλα βαθμολόγησης των blueprints (3.2. Βαθμολόγηση στοιχείων και blueprints).

Στο κεφάλαιο 4. Υ αναλύεται το κομμάτι της υλοποίησης για το Υλοποίηση Front-End, την Υλοποίηση Οντολογίας και το Υλοποίηση Back-End. Η ανάλυση αυτή γίνεται για κάθε ξεχωριστή κλάση/αρχείο που έχουν άμεσο αντίκτυπο στην εκτέλεση της εφαρμογής.

Στη συνέχεια, εξετάζονται τα 5. ΑΠΟΤΕΛΕΣΜΑΤΑ της εκτέλεσης της semantic αναζήτησης συγκριτικά με την simple αναζήτηση. Αυτά είναι είτε ποσοτικά (5.1. Ποσοτικός έλεγχος), δηλαδή, αφορούν για παράδειγμα πλήθος αποτελεσμάτων και ταχύτητα εκτέλεσης, είτε ποιοτικά (5.2. Ποιοτικός έλεγχος).

Τέλος, συνοψίζονται τα ΣΥΜΠΕΡΑΣΜΑΤΑ, αναλύονται Μελλοντικές επεκτάσεις και παρατίθεται ο ΚΩΔΙΚΑΣ.

2.4. Επιλογή εργαλείων

Για την υλοποίηση της εφαρμογής και συγκεκριμένα της semantic αναζήτησης αρχικά, επιλέχθηκε η χρήση της Java μέσω του Spring Boot framework. Η απλότητα, αλλά και οι δυνατότητες που προσφέρονται από το Spring-Boot το καθιστούν εύκολη επιλογή όσον αφορά τις Java εφαρμογές. Επίσης, χρησιμοποιήθηκε μια REST-API αρχιτεκτονική ως απλή web-αρχιτεκτονική.

Η βάση επιλέχθηκε να είναι μη-σχεσιακή, δηλαδη να είναι μια NoSQL βάση και συγκεκριμένα η MongoDB.

Για τον σχεδιασμό της οντολογίας χρησιμοποιήθηκε η OWL2 μέσω του εργαλείου Protégé. Η οντολογία έγινε reason με τον Pellet (Reasoner) και αξιοποιείται στην εφαρμογή μέσω του framework Apache Jena.

Επίσης, για τον σχεδιασμό του front-end σχεδιάστηκε μια απλή σελίδα με τη βοήθεια HTML και Javascript, η οποία στέλνει HTTP GET αιτήματα στο standalone microservice.

Τέλος, για τις μετρήσεις αξιοποιήθηκε το Apache JMeter.

3. ΑΝΑΛΥΣΗ

3.1. Εμπλουτισμός Στοιχείων προς αναζήτηση

Όπως αναφέρθηκε και στην ΑΡΧΙΤΕΚΤΟΝΙΚΗ, ο εμπλουτισμός γίνεται στατικά. Για κάθε εισαχθέν tag ακολουθείται μια διαδικασία με την οποία βρίσκονται σχετικά με αυτό tags (συγγενικά του). Αφού βρεθούν αυτά τα tags, μαζί με τα εισαχθέντα, αποτελούν το σύνολο των tags που θα περάσουν στο query για τη βάση (σελ. 20).

Με την εισαγωγή ενός tag από τον χρήστη, ελέγχεται αν υπάρχει στην οντολογία και αν είναι Class ή Individual. Αν δεν είναι τίποτα από τα δύο, σημαίνει πως δεν περιέχεται στην οντολογία και προφανώς αγνοείται. Αναλόγως τι από τα δύο είναι, υπάρχει μια σειρά κανόνων που ανακτώνται οι "συγγενείς" του στοιχείου αυτού και βαθμολογούνται (Βαθμολόγηση). Οι συγγενείς αυτοί είναι τα στοιχεία που θα εμπλουτίσουν την αναζήτηση. Οι κανόνες με τους οποίους επιλέγονται οι συγγενείς είναι τοποθετημένοι με σειρά φθίνουσας σημασίας. Ο συγγενής ενός στοιχείου που ικανοποιεί τον πρώτο κανόνα, βαθμολογείται υψηλότερα από οποιονδήποτε συγγενή, οποιουδήποτε στοιχείου που ικανοποιεί τον δεύτερο κανόνα κ.λπ.

Οι κανόνες εντοπίζουν δύο διαφορετικές περιπτώσεις, το εισαχθέν στοιχείο να είναι individual ή class:

3.1.1. Διαδικασία εμπλουτισμού με βάση εισαχθέν-tag individual

Κατά σειρά από αυτά με υψηλότερη βαθμολογία σε αυτά με τη χαμηλότερη, οι κανόνες για ένα individual θα επιστρέψουν:

1) Την κλάση στην οποία ανήκει το individual και όχι οποιαδήποτε υπερ-κλάση αυτής.

2) Τα siblings του individual, δηλαδή, τα individuals της κλάσης που ανήκει το εισαχθέν individual και όχι τα individuals οποιουδήποτε subclass της κλάσης αυτής.

3) Τέλος, όλες τις υπερ-κλάσεις της κλάσης του individual και όχι μόνο την άμεση υπερ-κλάση αλλά και υπερ-κλάσεις αυτής.

Πίνακας 3-1 : Κανόνες εμπλουτισμού Individual

Συγγένεια
Individual (input)
Direct class του individual
Direct siblings του individual
Όλες οι superclasses της κλάσης του individual

Για παράδειγμα μια αναζήτηση με tag *Asian* θα είναι:

Πίνακας 3-2 : Παράδειγμα εμπλουτισμού για το tag Asian (Individual)

Relationship	**tags**
Individual	Asian
Class of individual	Ethnicity
Sibling of individual	American_Indian_or_Alaska_Native, Black_or_African_American, Caucasian_or_White, Hispanic_/_Latino, Native_Hawaiian_or_Other_Pacific_Islander
Superclasses της κλάσης του individual	Demographics

3.1.2. Διαδικασία εμπλουτισμού με βάση εισαχθέν-tag class

Κατά σειρά από αυτά με υψηλότερη βαθμολογία σε αυτά με τη χαμηλότερη, οι κανόνες για ένα class θα επιστρέψουν:

1) Τις άμεσες υπο-κλάσεις της κλάσης και όχι υπο-κλάσεις των υπο-κλάσεων.

2) Τα individuals της κλάσης και όχι τα individuals οποιουδήποτε subclass της κλάσης.

3) Τέλος, όλες τις υπερ-κλάσεις της κλάσης και όχι μόνο την άμεση υπερ-κλάση αλλά και υπερ-κλάσεις αυτής.

Πίνακας 3-3 : Κανόνες εμπλουτισμού Class

Συγγένεια
Class (input)
Direct subclass του class
Direct individual του class
Όλα τα superclasses του class

Για παράδειγμα μια αναζήτηση με tag *Doctor* θα είναι:

Πίνακας 3-4 : Παράδειγμα εμπλουτισμού για το tag Doctor (Class)

Relationship	**tags**
Class	Doctor
Direct subclass of class	FamilyDoctor, Specialist

Individual of class	
Superclass του class	Person

3.1.3. Σύνοψη Εμπλουτισμού

Εν τέλει τα tags που θα προκύψουν για αναζήτηση στη βάση με εισαχθέντα tags τα *Asian* και *Doctor* θα είναι :

Simple search : {Asian, Doctor}

Semantic search : {Asian, Ethnicity, American_Indian_or_Alaska_Native, Black_or_African_American, Caucasian_or_White, Hispanic_/_Latino, Native_Hawaiian_or_Other_Pacific_Islander , Demographics, Doctor, FamilyDoctor, Specialist, Person}

Είναι εμφανές πως στην $2^{η}$ περίπτωση, η αναζήτηση θα είναι εκτενέστερη αφού θα χρησιμοποιηθούν τα εισαχθέντα tags, αλλά και αυτά που προέκυψαν από τη διαδικασία του εμπλουτισμού. Με αυτόν τον τρόπο, θα υπάρχει πολύ μεγαλύτερο πλήθος από blueprints που θα επιστραφούν στον χρήστη.

Προκύπτει έτσι η ανάγκη ταξινόμησής τους με βάση το πόσο σχετικά είναι με τα αποτελέσματα που επιζητά ο χρήστης. Για τον λόγο αυτό, δημιουργήθηκε το σύστημα βαθμολόγησης των blueprints το οποίο στηρίζεται στη συγγένεια των tags που εμπλουτίζουν την αναζήτηση με τα αρχικά, tags και τον βαθμό της συγγένειας αυτής.

3.2. Βαθμολόγηση στοιχείων και blueprints

Πέρα από τον εμπλουτισμό της αναζήτησης με καινούρια στοιχεία (όσον αφορά την semantic αναζήτηση), υπάρχει ανάγκη ταξινόμησης και βαθμολόγησης ανεξάρτητα από το είδος της αναζήτησης.

Όπως αναφέρθηκε και στην Βαθμολόγηση στοιχείων και blueprints, η βαθμολόγηση είναι παραμετροποιήσιμη. Αυτό σημαίνει πως ο βαθμός της κάθε σχέσης μπορεί πολύ απλά να αλλάξει και κατά συνέπεια να βαθμολογηθούν με διαφορετικό τρόπο τα blueprints που επιστρέφονται.

Η βαθμολόγηση χωρίζεται σε δύο κομμάτια:

1) Τη βαθμολόγηση του κάθε στοιχείου ξεχωριστά ανάλογα με τη συγγένεια που έχει με το εισαχθέν στοιχείο

2) Τη βαθμολόγηση του κάθε blueprint με βάση τους επιμέρους βαθμούς των στοιχείων του.

3.2.1. Βαθμολόγηση κάθε στοιχείου με βάση τη συγγένεια με το tag

Οι υπάρχοντες κανόνες εμπλουτισμού είναι υπεύθυνοι και για την ανάθεση ενός ιεραρχικού συστήματος βαθμολόγησης. Αφού οι κανόνες ορίσουν με ποια στοιχεία θα εμπλουτιστούν τα ήδη υπάρχοντα στοιχεία (Εμπλουτισμός), ελέγχουν τα στοιχεία αυτά, δηλαδή, τι είδους σχέση έχουν με τα εισαχθέντα στοιχεία-tags και τους αντιστοιχίζουν έναν βαθμό, που ο μεγαλύτερος βαθμός αντιστοιχεί και σε ακριβέστερη συσχέτιση. Οι κανόνες που υλοποιήθηκαν εντοπίζουν δύο διαφορετικές περιπτώσεις, το εισαχθέν tag να είναι Class ή Individual.

Individual : Αν το εισαχθέν tag είναι Individual, το tag ενός blueprint που είναι ίδιο με το εισαχθέν έχει βαθμό 100, αν είναι ίδιο με την κλάση στην οποία ανήκει το εισαχθέν individual, θα έχει βαθμό 80 και αντίστοιχα, αν είναι με sibling του εισαχθέντος (δηλαδή, individual της κλάσης του εισαχθέντος), θα έχει βαθμό 71 και τέλος, αν είναι ίδιο με κάποια υπερ-κλάση της κλάσης του εισαχθέντος (superclass), με βαθμό 59.

Πίνακας 3-5 : Βαθμολογία ανά κανόνα εμπλουτισμού για individual-tag

Relationship	**Rank**
Individual	100
Class of individual	80
Sibling of individual	71
Superclass of individual's class	59

Έτσι, το Individual *Asian* (βαθμός 100) θα εμπλουτιστεί με το στοιχείο *Ethnicity* ως κλάση στην οποία ανήκει (βαθμός 80), με τα *American_Indian_or_Alaska_Native*, *Hispanic_/_Latino*, *Black_or_African_American*, *Caucasian_or_White*, *Native_Hawaiian_or_Other_Pacific_Islander* ως siblings του (βαθμός 71) και με το *Demographics* ως υπερκλάση του *Ethnicity* (βαθμός 59).

Πίνακας 3-6 : Βαθμολογία και στοιχεία εμπλουτισμού για εισαχθέν tag Asian

Relationship	**rank**	**tags**
individual	100	Asian
Class of individual	80	Ethnicity
Sibling of individual	71	American_Indian_or_Alaska_Native, Black_or_African_American, Caucasian_or_White, Hispanic_/_Latino, Native_Hawaiian_or_Other_Pacific_Islander
Direct superclass of individual's class	59	Demographics

Class : Αν το εισαχθέν tag είναι Class, το tag ενός blueprint που είναι ίδιο με το εισαχθέν, έχει βαθμό 100, αν είναι ίδιο με υπο-κλάση (subclass) του εισαχθέντος, θα έχει βαθμό 80 και αντίστοιχα αν είναι ίδιο με individual του εισαχθέντος, θα έχει βαθμό 71 και τέλος, αν είναι ίδιο με κάποια υπερ-κλάση του εισαχθέντος (superclass), με βαθμό 61.

Πίνακας 3-7 : Βαθμολογία ανά κανόνα εμπλουτισμού για class-tag

Relationship	**rank**
Class	100
Direct subclass of class	80
Individual of class	71
Superclass of class	61

Έτσι, το tag *Doctor* (βαθμός 100) θα εμπλουτιστεί με τα στοιχεία *FamilyDoctor* και *Specialist* ως subclasses (βαθμοί 80) και με το στοιχείο *Person* ως superclass (βαθμός 61). Αντίστοιχα, το tag *SmokingStatus* (βαθμός 100) θα εμπλουτιστεί με τα *Ex-Smoker*, *Never_Smoker*, *Smoker* ως individuals (βαθμοί 71).

Πίνακας 3-8 : Βαθμολογία και στοιχεία εμπλουτισμού για εισαχθέν tag Doctor

Relationship	**rank**	**tags**
Class	100	Doctor
Direct subclass of class	80	FamilyDoctor, Specialist
Individual of class	71	
Direct superclass of class	61	Person

3.2.2. Βαθμολόγηση του blueprint με βάση τους βαθμούς των tags

Αφού εμπλουτιστεί το πλήθος των στοιχείων που θα αναζητηθεί στη βάση και βαθμολογηθεί το κάθε στοιχείο ξεχωριστά, μένει το κομμάτι της βαθμολόγησης των blueprints. Αρχικά, ο βαθμός του κάθε blueprint θα πρέπει να εξαρτάται από τα επιμέρους στοιχεία, οπότε προκύπτει ως το άθροισμά τους:

$$\sum_{i=0}^{n} Gi$$

Gi: Ο βαθμός του κάθε tag που περιέχεται στο blueprint. Ασφαλώς, αν κάποιο στοιχείο του blueprint δεν ανήκει στο σύνολο των tags (εμπλουτισμένο και μη), δεν προσμετράται.

Το πλήθος των tags όμως, θα πρέπει να διαδραματίζει σημαντικό ρόλο στη βαθμολόγηση του blueprint. Ένα blueprint που έχει πέντε αντιστοιχίες ασφαλώς θα πρέπει να προτιμάται από κάποιο που έχει μόνο μία. Όμως, δεν θα πρέπει μόνο το πλήθος των αντιστοιχιών να καθορίζει απόλυτα τη βαθμολογία ενός blueprint. Δηλαδή, ένα blueprint που περιέχει 3 ακριβή ταιριάσματα (περιέχει 3 tags από αυτά που εισήχθησαν) θα πρέπει να προτιμάται από ένα που περιέχει 4 μερικά (περιέχει 4 tags συγγενικά με αυτά που εισήχθησαν). Για αυτήν την κανονικοποίηση του βαθμού, δεν χρησιμοποιείται στη φόρμουλα μόνο το άθροισμα των βαθμών όλων των στοιχείων, αλλά το άθροισμα αυτό θα διαιρείται με το πλήθος των στοιχείων που περιέχει το blueprint. Έτσι, θα προτιμηθεί ένα blueprint με 3 ακριβή ταιριάσματα από ένα με 4 μερικά.

$$\frac{\sum_{i=0}^{n} Gi}{N}$$

Gi: Ο βαθμός του κάθε tag που περιέχεται στο blueprint. Ασφαλώς, αν κάποιο στοιχείο του blueprint δεν ανήκει στο σύνολο των tags (εμπλουτισμένο και μη), δεν προσμετράται.
N: Το πλήθος των στοιχείων του blueprint.

Για παράδειγμα αντί να βαθμολογηθούν 2 blueprint ως:

$$100 + 100 + 100 < 80 + 80 + 80 + 80$$

Προκύπτει:

$$\frac{100 + 100 + 100}{3} > \frac{80 + 80 + 80 + 80}{4}$$

Όμως, αν σε μια simple αναζήτηση βρεθεί ένα blueprint με 1 στοιχείο που να ταιριάζει με ένα εισαχθέν στοιχείο (άρα βαθμός=100/1=100) και ένα blueprint με 2 στοιχεία που να ταιριάζουν με δύο εισαχθέντα στοιχεία (άρα Rank=(100+100)/2=100), θα έχουν την ίδια βαθμολογία. Ουσιαστικά όμως, το 2ο blueprint ταιριάζει πολύ καλύτερα από το πρώτο καθώς κατάφερε να ταιριάξει βάσει 2 tags. Έτσι, για να προσδίδεται μεγαλύτερη βαθμολογία σε μεγαλύτερο αριθμό ταιριασμάτων (χωρίς ωστόσο, να απαλειφθεί η κανονικοποίηση) πρέπει όσο αυξάνεται το μέγεθος του blueprint, να μην αυξάνεται αναλογικά ο παρανομαστής. Επιλέχθηκε η χρήση της ρίζας για αυτόν τον λόγο. Προκύπτει η τελική φόρμουλα βαθμολόγησης των blueprint ως εξής:

Πίνακας 3-9 : Αλγόριθμος υπολογισμού βαθμού blueprint

$$\frac{\sum_{i=0}^{n} Gi}{\sqrt{N}}$$

Gi: Ο βαθμός του κάθε tag που περιέχεται στο blueprint. Ασφαλώς, αν κάποιο στοιχείο του blueprint δεν ανήκει στο σύνολο των tags (εμπλουτισμένο και μη), δεν προσμετράται
N: Το πλήθος των στοιχείων του blueprint.

Έτσι, ενώ το blueprint με 1 στοιχείο θα έχει πάλι βαθμό 100, αυτό με τα 2 στοιχεία θα έχει βαθμό 141,421, ενώ αυτό με τα 3 θα έχει 173,205.

3.2.3. Διαδικασία βαθμολόγησης

Επίσης, όσον αφορά τη διαδικασία της βαθμολόγησης υπάρχουν οι εξής κανόνες λογικής που ακολουθούνται :

- Κάθε εισαχθέν tag δημιουργεί ένα **tag group**, μια ομάδα από tags δηλαδή, που περιλαμβάνουν το ίδιο καθώς και άλλα tags που προκύπτουν με εμπλουτισμό με βάση αυτό.
- Για κάθε blueprint χρησιμοποιείται μόνο **μια φορά ένα tag group**.
- Ο βαθμός που προκύπτει στο κάθε blueprint, προκύπτει από την **καταλληλότερη επιλογή στοιχείου** από κάθε tag group, άρα δεν μπορεί να αξιοποιηθεί μια greedy λογική βαθμολόγησης που το 1ο (ή και μεγαλύτερο) ταίριασμα είναι και το καλύτερο.

Έστω αναζήτηση με tags *Gynecologist*, *Doctor* και το blueprint

FamilyDoctor, Doctor, Psychiatrist, MedicalImagingTest, OralTest

Από το tag *Gynecologist* προκύπτουν τα εξής στοιχεία:

Gynecologist(grade 100), Person(61), Doctor(61), Specialist(61)

Από το tag *Doctor* προκύπτουν τα εξής στοιχεία:

Doctor(grade 100), Specialist(80), FamilyDoctor(80), Person(61)

Για τον βαθμό του blueprint θα χρησιμοποιηθούν τα

$$\frac{(\text{Doctor}(61) + \text{FamilyDoctor}(80))}{\sqrt{5}} = 63.057117$$

Δεν θα χρησιμοποιηθεί δηλαδή, 2 φορές το tag group του *Doctor* (*Doctor*(100)+*FamilyDoctor*(80)), ούτε θα χρησιμοποιηθεί το tag *Doctor* από το *Doctor* tag group, καθώς αυτό δεν θα επέστρεφε το ιδανικό αποτέλεσμα.
(παράδειγμα διαδικασίας εκτέλεσης βαθμολόγησης: Πίνακας 4-14 : Παράδειγμα ακολουθιακής εκτέλεσης dynamicGrade)

3.2.4. Υπολογισμός των μετρικών

Η επιλογή των βαθμών, των συγγενών του tag που επιστρέφονται, αλλά και της καθαυτής ιεραρχίας των κανόνων είναι κάτι παραμετροποιήσιμο και εξαρτάται από τις ανάγκες και τον σκοπό του προβλήματος. Στα πλαίσια της παρούσας διπλωματικής, οι μετρικές επιλέχθηκαν ώστε να ικανοποιούν κάποιους λογικούς κανόνες.

Όσον αφορά τη βαθμολογία των tags, αφού επιλέχθηκαν οι σχέσεις με τις οποίες θα εμπλουτίζονται τα tags (για παράδειγμα sub-classes του δοσμένου class ή siblings του δοσμένου individual) και ταξινομήθηκαν από τα πιο σημαντικά (μεγάλος βαθμός) στα λιγότερα σημαντικά, έπρεπε να επιλεχθούν και οι τελικοί βαθμοί που θα τοποθετηθούν. Για τον λόγο αυτό αρχικά, η πλήρης αντιστοίχιση εισαχθέντος-tag και blueprint-tag για λόγους ευκολίας ορίστηκε 100. Οι λογικοί κανόνες με τους οποίους προκύπτουν οι βαθμοί εξαρτώνται από την υλοποίηση, αρκεί να έχουν ως αποτέλεσμα βαθμούς σε γνησίως φθίνουσα σειρά. Οι κανόνες οι οποίοι επιλέχθηκαν για την επιλογή των μετρικών αντιστοιχούν στις παρακάτω σχέσεις:
(Θεωρείται πως το εισαχθέν tag έχει βαθμό 100, tag συγγένειας πρώτου βαθμού έχουν βαθμό $\boldsymbol{a}$, 2^{ου} βαθμού έχουν βαθμό $\boldsymbol{b}$ και 3^{ου} βαθμού έχουν βαθμό $\boldsymbol{c}$.)

1) Σε αναζήτηση με δοσμένα 2 στοιχεία-tags, ένα blueprint που περιέχει 1 tag που ταιριάζει ακριβώς με ένα από τα εισαχθέντα, να έχει χαμηλότερο βαθμό από ένα blueprint που έχει 2 tags που προκύπτουν από συγγένεια με τα εισαχθέντα (την 2^{η} κατά σειρά συγγένεια).

$$\frac{1 \times \mathbf{100}}{\sqrt{1}} < \frac{2 \times \boldsymbol{b}}{\sqrt{2}}$$

$$\boldsymbol{b} > 70.71$$

2) Σε αναζήτηση με δοσμένα 5 στοιχεία-tags, ένα blueprint που περιέχει 3 ακριβή ταίρια να έχει χαμηλότερο βαθμό από ένα που έχει tags που προκύπτουν από συγγένεια (1^{ου} βαθμού).

$$\frac{5 \times \boldsymbol{a}}{\sqrt{5}} > \frac{3 \times \mathbf{100}}{\sqrt{3}}$$

$$\boldsymbol{a} > 77.46$$

3) Σε αναζήτηση με δοσμένα 3 στοιχεία-tags, ένα blueprint που περιέχει 3 tags που προκύπτουν με συγγένεια (1ου βαθμού) να έχει υψηλότερο βαθμό από ένα blueprint που περιέχει 1 ακριβές ταίρι και 2 με συγγένεια (ένα 1ου και ένα 3ου βαθμού) **όταν** και τα 3 δοσμένα tags παραπέμπουν σε class της οντολογίας και χαμηλότερο βαθμό όταν παραπέμπουν σε individual.

Για class:

$$\frac{3 \times \boldsymbol{a}}{\sqrt[3]{3}} > \frac{\boldsymbol{100} + \boldsymbol{c} + \boldsymbol{a}}{\sqrt[3]{3}}$$

Για individual:

$$\frac{3 \times \boldsymbol{a}}{\sqrt[3]{3}} < \frac{\boldsymbol{100} + \boldsymbol{c} + \boldsymbol{a}}{\sqrt[3]{3}}$$

Βάσει των κανόνων αυτών, επιλέγονται οι παρακάτω τιμές, όπως ορίζονται στον κώδικα:

```
public final static int DEFAULT_PARENT_TAG_RANK = 100;

public final static int RANK_CLASS_SUBCLASS = 80;
public final static int RANK_CLASS_INDIVIDUALS = 71;
public final static int RANK_CLASS_SUPERCLASSES = 61;

public final static int RANK_IND_DIR_CLASS = 80;
public final static int RANK_IND_SIBLINGS = 71;
public final static int RANK_IND_SUPERCLASSES = 59;
```

3.2.5. Σύνοψη Βαθμολόγησης

Η διαδικασία της βαθμολόγησης είναι άρρηκτα συνδεδεμένη με τη διαδικασία εμπλουτισμού. Αφού επιλεχθούν οι σχέσεις εμπλουτισμού, παραμένει μόνο η αντιστοίχιση των κατάλληλων μετρικών στην κάθε σχέση. Οι σχέσεις (λογικοί κανόνες) τις οποίες θα ικανοποιεί ο κάθε βαθμός θα πρέπει να είναι αποτέλεσμα προσεκτικής ανάλυσης, καθώς καθορίζουν σημαντικά το αποτέλεσμα της ταξινόμησης.

3.3. Σύνοψη ανάλυσης

Συμπερασματικά, τα αποτελέσματα τα οποία θα επιστραφούν στον χρήστη (και η σειρά τους) εξαρτώνται από:

1. Τους **κανόνες εμπλουτισμού** (δηλαδή, το τι θα επιστρέφεται από τη βάση)
2. Την επιλογή **λογικών κανόνων** για τους βαθμούς που αντιστοιχίζονται στους κανόνες εμπλουτισμού

3. Τους **βαθμούς** που αντιστοιχίζονται στους κανόνες εμπλουτισμού (όπως προκύπτουν από τους λογικούς κανόνες)
4. Τον **σχεδιασμό της οντολογίας**

Ο σχεδιασμός της οντολογίας θα εξεταστεί στο κεφάλαιο Υλοποίηση Οντολογίας, παρ'όλα αυτά εξαρτάται κυρίως από την εμπειρία του ontology expert. Είναι επιλογή του ontology-expert πώς θα σχεδιάσει την οντολογία και κατά συνέπεια ποιες σχέσεις θα αξιοποιήσει στο σύστημα εμπλουτισμού και με ποια σειρά σημασίας. Θα ήταν συνετό όμως, οι παραπάνω παράγοντες που διαμορφώνουν την ποιότητα των αποτελεσμάτων να είναι απόρροια συλλογικής σκέψης και μελέτης.

Είναι προφανές λοιπόν, πως η επιλογή των κανόνων εμπλουτισμού, η ταξινόμησή τους κατά σειρά σημασίας, οι βαθμοί που τους αντιστοιχούν, ακόμα και η φόρμουλα καθαυτή που ορίζεται για τη βαθμολόγηση των blueprints εξαρτώνται από την εκάστοτε υλοποίηση.

Για παράδειγμα, για τις ανάγκες άλλου προβλήματος μπορεί στην φόρμουλα να αξιοποιούνταν η $3^{η}$ ρίζα αντί της $2^{ης}$ με αποτέλεσμα να αυξανόνταν η εξάρτηση από το πλήθος των tags.

Μπορεί επίσης να χρησιμοποιούνταν μια ταξινόμιση με διαφορετικούς βαθμούς, όπου για παράδειγμα το superclass μιας κλάσης θα είχε υψηλότερη βαθμολογία από το αντίστοιχο subclass ή μπορεί να μην επιστρέφονταν καν τα subclasses.

Τέλος, μπορεί το query στη βάση να σχεδιαστεί διαφορετικά (βλ. MongoApp.java). Αντί να χρησιμοποιούνται όλα τα tags, δηλαδή, και αυτά που εισήχθησαν και αυτά που προέκυψαν από τον εμπλουτισμό, μπορεί να χρησιμοποιηθούν μόνο τα δεύτερα. Μπορεί επίσης το query αντί να αναζητά blueprints που να περιέχουν οποιοδήποτε εκ του συνόλου των tags (εισαχθέντα και από εμπλουτισμό), να αναζητά blueprints που να περιέχουν ακριβώς ένα tag από κάθε tag group.

4. ΥΛΟΠΟΙΗΣΗ

4.1. Υλοποίηση Front-End

Στόχος της διπλωματικής εργασίας είναι η δημιουργία μιας REST-API εφαρμογής. Συνεπώς όσον αφορά το κομμάτι του Front-End δημιουργήθηκε ένα απλό web page για ευκολία χρήσης των δύο μορφών αναζήτησης. Χρησιμοποιήθηκε ένα βασικό index.html αρχείο διακοσμημένο με ένα stylesheet αρχείο και η όποια επεξεργασία στο front-end γίνεται με τη χρήση ενός javascript αρχείου. Πάνω στο index.html χρησιμοποιήθηκε ένα έτοιμο Bootstrap θέμα, το Grayscale [26] .

4.1.1. Index.html

Παρατηρώντας το index.html αρχείο, φαίνεται πως η οθόνη χωρίζεται σε δύο οριζόντια κομμάτια, ενώ το δεύτερο οριζόντιο σε τρία κατακόρυφα. Το πρώτο οριζόντιο κομμάτι καλωσορίζει τον χρήστη στη σελίδα.

Στο δεύτερο κομμάτι, το πρώτο μέρος περιλαμβάνει ένα text-box με όνομα *Tag 1* και από κάτω δύο ζεύγη κουμπιών, ένα για προσθήκη περιεχομένου, ένα για αφαίρεση και δύο για αναζήτηση, simple και semantic. Το δεύτερο κομμάτι γεμίζει έναν πίνακα δυναμικά με στοιχεία μετά από μια simple αναζήτηση στη βάση. Το τρίτο κομμάτι αντίστοιχα, γεμίζει έναν πίνακα δυναμικά με στοιχεία μετά από μια semantic αναζήτηση στη βάση.
Παρακάτω φαίνονται τα 3 κουτιά και τα stylesheet τους:

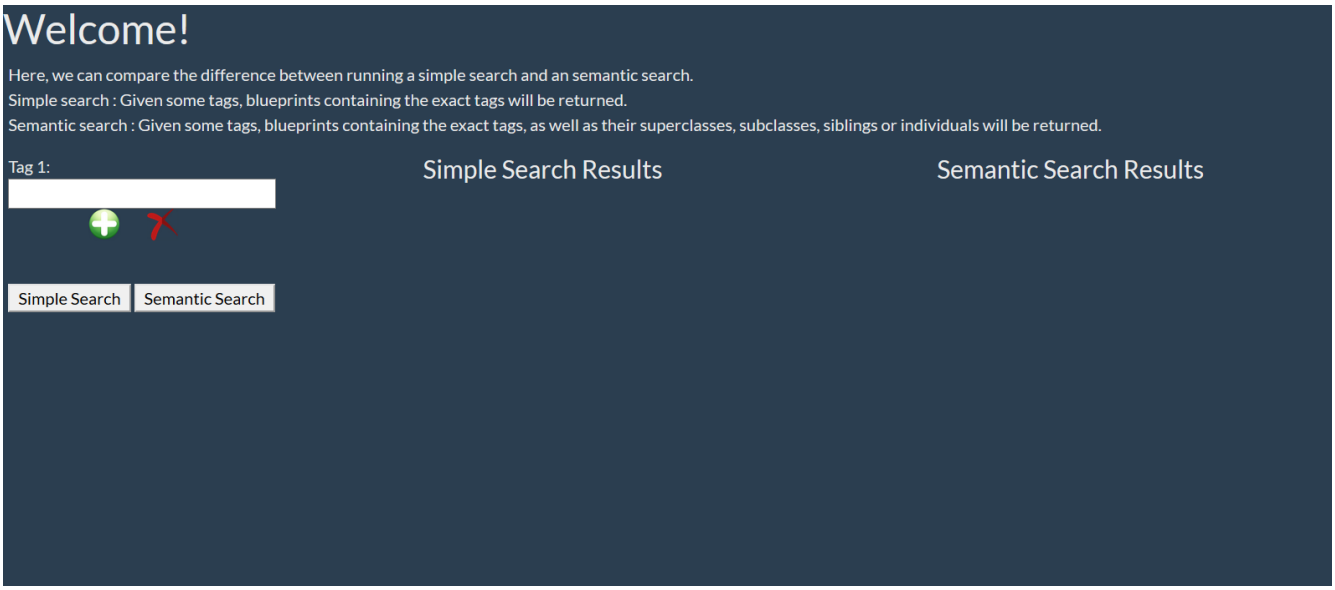
Πίνακας 4-1 : Stylesheet για το front-end

.first {
width: 20%;
float:left; /* add this */
}
	.second {
	width: 39%;
	float: left; /* add this */
	margin-left : 5px;
	font-size: 0.8rem;
	}
		.third {
		width: 39%;
		float: left; /* add this */
		margin-left : 5px;
		font-size: 0.8rem;
		}

 <form id="frm1""> Tag 1: <input type="text" id="tag1" style="width:100%"> </form> <button style="margin- left:30%; width:30px; height:30px; border- style:none;" onclick="addInputTag()" class="btn_add"></button>	<div class="second" id="simple_div"> <h4 style="text-align: center">Simple Search Results</h4> 	<div class="third" id="advanced_div""> <h4 style="text-align: center">Semantic Search Results</h4>

<button style="margin-left:10%; width:30px; height:30px; border-style:none;" onclick="deleteInputTag()" class="btn_rem"></button> <button style="display:inline-block" onclick="simpleSearch()">Simple Search</button> <button style="display:inline-block" onclick="semanticSearch()">Semantic Search</button> 		

Εν τέλει η εικόνα που παρουσιάζεται στον χρήστη είναι η παρακάτω:



Εικόνα 4.1 Αρχική οθόνη

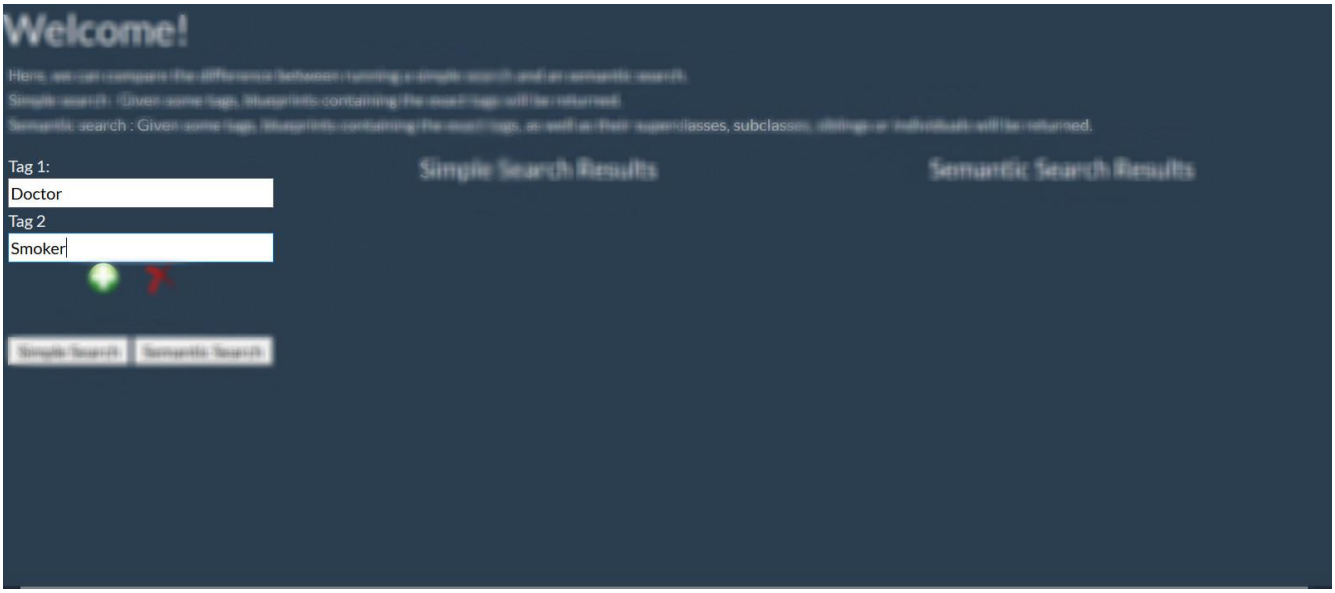
Με τη βοήθεια του javascript αρχείου, ο χρήστης μπορεί να κάνει τις ακόλουθες ενέργειες:
Πατώντας το κουμπί προσθήκης περιεχομένου, ο χρήστης μπορεί να δημιουργήσει καινούρια πεδία-tags για να τα γεμίσει. Μπορεί να έχει το πολύ 7 πεδία-tags.

Εικόνα 4.2 Κουμπί προσθήκης περιεχομένου

Πατώντας το κουμπί αφαίρεσης περιεχομένου, ο χρήστης μπορεί να διαγράψει πεδία-tags. Πάντα θα υπάρχει 1 πεδίο-tag.

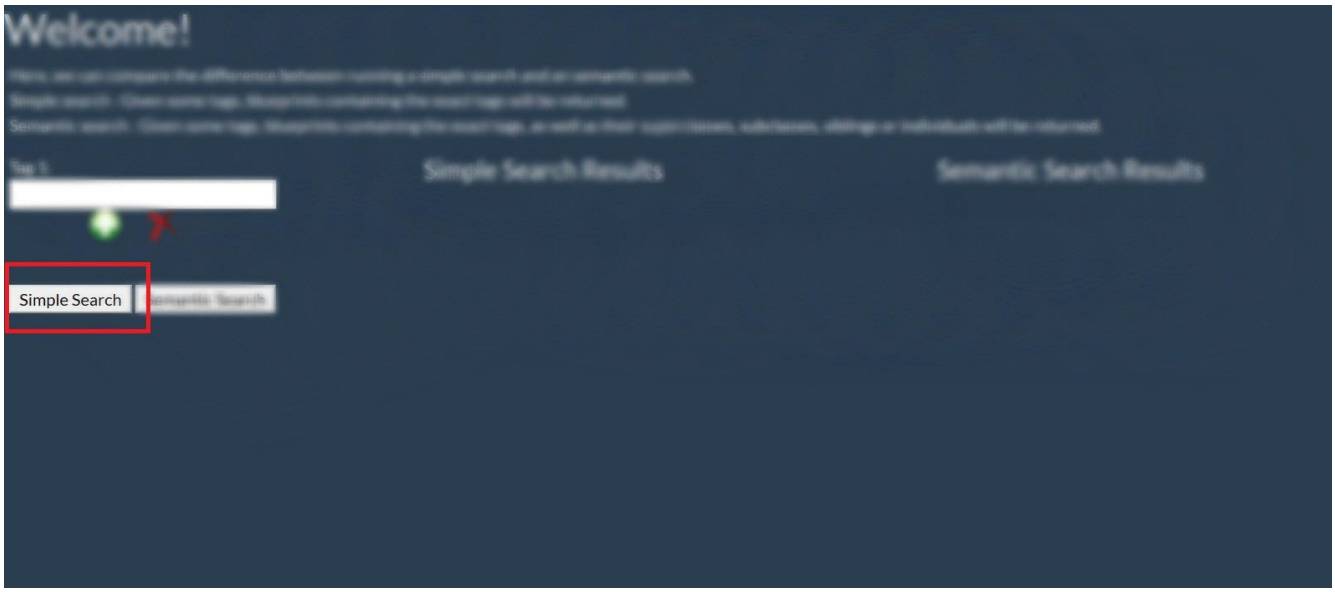
Εικόνα 4.3 Κουμπί αφαίρεσης περιεχομένου

Μπορεί να εισάγει σε κάποιο πεδίο Tag κάποιο στοιχείο που θα ήθελε να αναζητήσει.

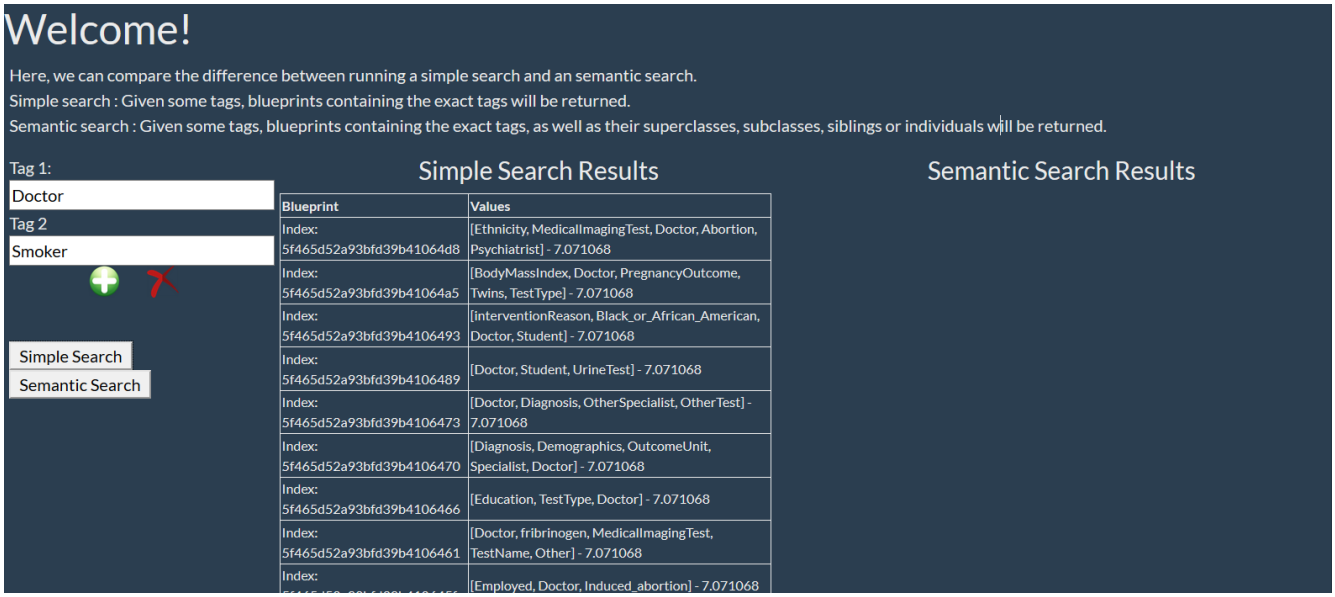


Εικόνα 4.4 Πεδία εισαγωγής tags

Πατώντας το κουμπί για απλή αναζήτηση, ξεκινά η διαδικασία του simple search (MyController.java) με βάση τα tags που έχει βάλει ο χρήστης. Όλα τα πεδία-tags πρέπει να περιέχουν τιμή.



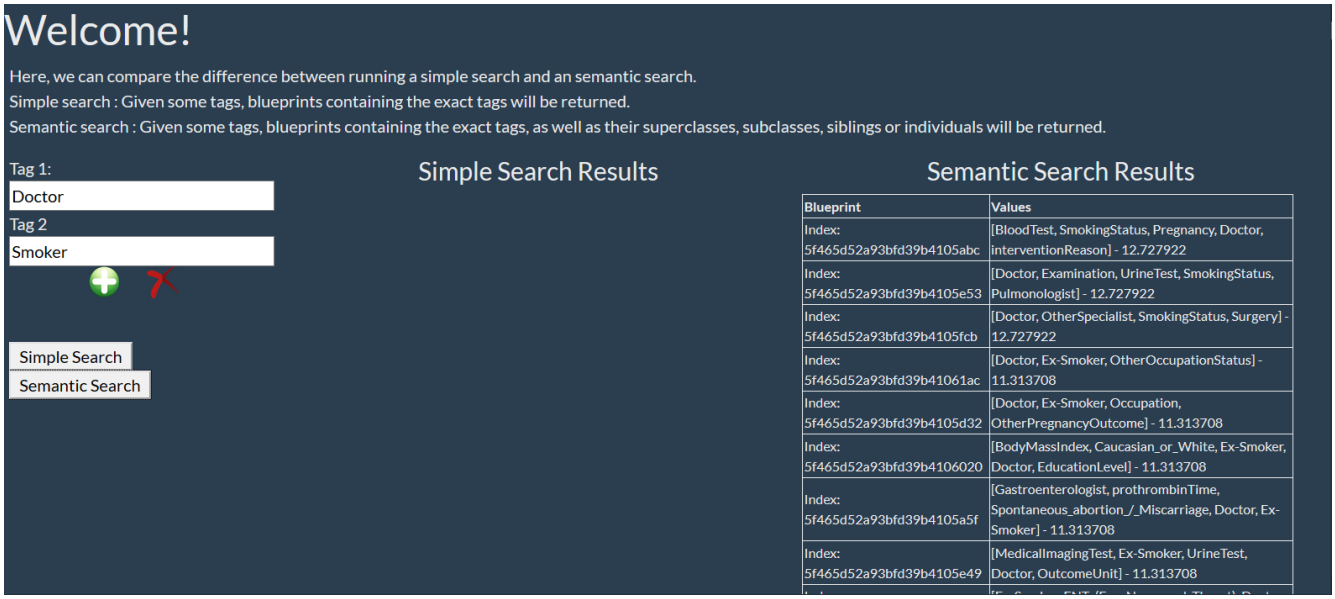
Εικόνα 4.5 Εκτέλεση simple search



Εικόνα 4.6 Αποτελέσματα εκτέλεσης simple search

Πατώντας το κουμπί για semantic αναζήτηση, ξεκινά η διαδικασία του semantic search (MyController.java) με βάση τα tags που έχει βάλει ο χρήστης. Όλα τα πεδία-tags πρέπει να περιέχουν τιμή.

Εικόνα 4.7 Εκτέλεση semantic search



Εικόνα 4.8 Αποτελέσματα εκτέλεσης semantic search

4.1.2. Javascript.js

Στο αρχείο αυτό υλοποιούνται 5 απλές συναρτήσεις για εξυπηρέτηση του index.html
addInputTag: Έχοντας κρατήσει τον τωρινό αριθμό από πεδία-tags, όταν καλείται αυτή η συνάρτηση αν τα πεδία-tags είναι λιγότερα από 7, προσθέτει 1.

deleteInputTag: Έχοντας κρατήσει τον τωρινό αριθμό από πεδία-tags, όταν καλείται αυτή η συνάρτηση αν τα πεδία-tags είναι περισσότερα από 1, αφαιρεί τα πεδία που έχουν προστεθεί, δηλαδή, ένα κενό
, το πεδίο-tag και το όνομα του πεδίου.

addTable: Με την κλήση αυτής της συνάρτησης περνάνε ως παράμετροι 2 μεταβλητές. Ένα αρχείο json και ένα id. Το id αυτό, αναφέρεται στο πεδίο του index.html ( ) που θα φιλοξενήσει έναν πίνακα. Ο πίνακας που θα δημιουργηθεί έχει δύο στήλες, μία περιέχει ένα αναγνωριστικό για κάθε document-blueprint που έχει επιστραφεί από τη βάση και η άλλη περιέχει το document-blueprint. Και οι δύο στήλες προκύπτουν από το json αρχείο που έχει περάσει ως παράμετρος.

simpleSearch: Με την κλήση αυτής της συνάρτησης, ανακτώνται οι τιμές των πεδίων-tags και με βάση αυτά, καλείται το endpoint/simple. Ανεξάρτητα από το πλήθος των πεδίων-tag που θα έχει γεμίσει ο χρήστης, η συνάρτηση καλεί το endpoint με επτά tags, όπου όσα πεδία δεν έχουν συμπληρωθεί αποκτούν κενή τιμή (αλλά ορίζονται στο http request). Στη συνέχεια, εκτελείται fetch στο endpoint, και η απάντηση του, ένα αρχείο json, περνάει ως παράμετρος σε κλήση της addTable μαζί με το κατάλληλο αναγνωριστικό για το πεδίο του index.html.

Το Fetch API, που χρησιμοποιείται για την κλήση του back-end endpoint, είναι ένα εύχρηστο Javascript interface για HTTP Requests. Συγκεκριμένα, γίνεται το fetch στο back-end

```
const response = await fetch('http://localhost:8090/simple'+fetcher);
```

και όταν επιστραφεί η απάντηση, αυτή επεξεργάζεται ως json

```
const advancedJson = await response.json();
```

semanticSearch: Με την κλήση αυτής της συνάρτησης, ανακτώνται οι τιμές των πεδίων-tags και με βάση αυτά, καλείται το endpoint/semantic. Ανεξάρτητα από το πλήθος των πεδίων-tag που θα έχει γεμίσει ο χρήστης, η συνάρτηση καλεί το endpoint με επτά tags, όπου όσα πεδία δεν έχουν συμπληρωθεί αποκτούν κενή τιμή (αλλά ορίζονται στο http request) Στη συνέχεια, εκτελείται fetch στο endpoint, και η απάντηση του, ένα αρχείο json περνάει ως παράμετρος σε κλήση της addTable μαζί με το κατάλληλο αναγνωριστικό για το πεδίο του index.html.

Αντίστοιχα με το simpleSearch, και εδώ η επικοινωνία με το back-end γίνεται με fetch

```
onst response = await fetch('http://localhost:8090/semantic'+fetcher);
```

και όταν επιστραφεί η απάντηση, αυτή επεξεργάζεται ως json

```
const advancedJson = await response.json();
```

4.2. Υλοποίηση Οντολογίας

Η οντολογία χρησιμοποιήθηκε στην απλούστερή της μορφή χωρίς να οριστούν σχέσεις μεταξύ των κλάσεων ή και των individuals. Οι σχέσεις οι οποίες χρησιμοποιήθηκαν είναι αυτές που ορίζουν μια κλάση ως υπο-κλάση (subclass) μιας άλλης κλάσης (αντίστοιχα η δεύτερη κλάση ως υπερ-κλάση της πρώτης κλάσης) και ένα individual ως παιδί μιας κλάσης.

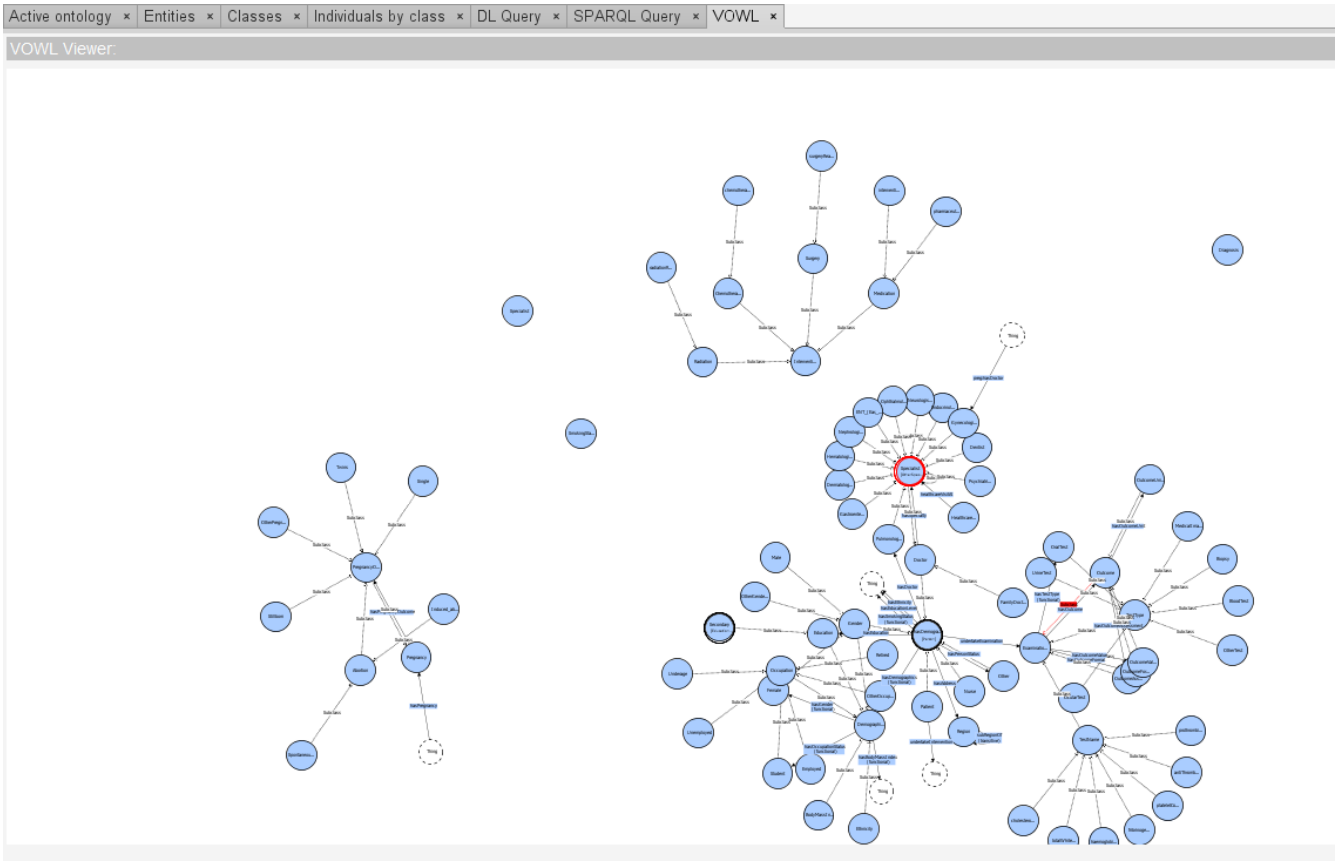
Έτσι, διαμορφώνεται ένας "σκελετός" με βάση τον οποίο προκύπτουν συγγένειες μεταξύ των στοιχείων. Όπως αναφέρθηκε και νωρίτερα, η οντολογία στηρίζεται σε ιατρικά δεδομένα και συγκεκριμένα σε blueprints του e-health κομματιού του DITAS project.

Με βάση τα δοσμένα blueprints του project, απομονώθηκαν στοιχεία και δημιουργήθηκαν με τη βοήθεια του Protégé classes και individuals τα οποία στοχεύουν να εμπλουτίσουν την αναζήτηση του χρήστη.

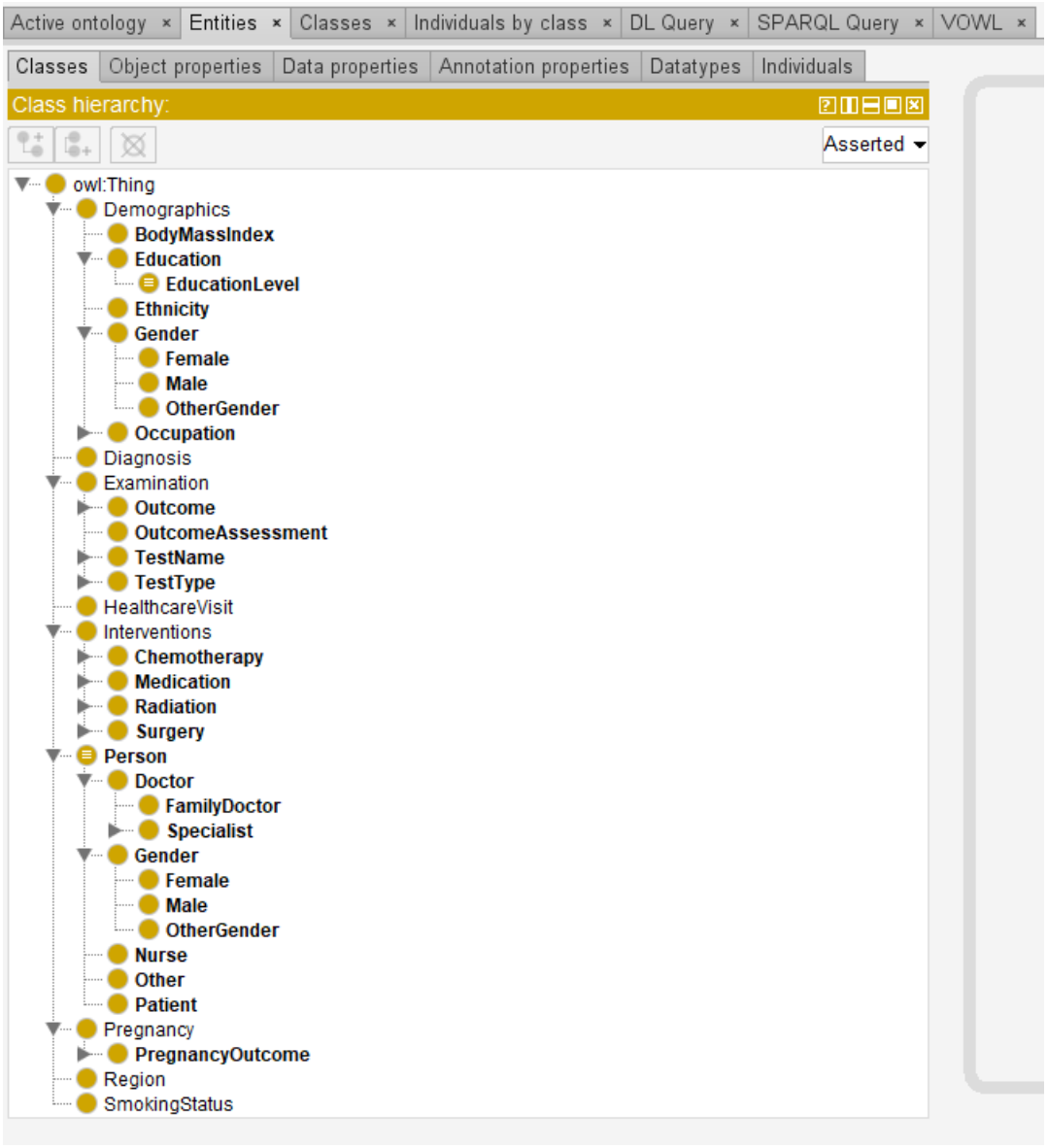
Το πιο σημαντικό χαρακτηριστικό της εφαρμογής αυτής είναι πως η αναζήτηση είναι ontology-agnostic. Η υλοποίηση της εφαρμογής δηλαδή, είναι ανεξάρτητη των δεδομένων που επιστρέφονται από τη βάση ή από την οντολογία που χρησιμοποιείται για να προκύψουν συγγένειες μεταξύ στοιχείων. Η εφαρμογή μπορεί να δεχθεί σαν είσοδο διαφορετικές οντολογίες, ανάλογα με τα στοιχεία της βάσης δεδομένων, όπως θα τις έχει σχεδιάσει ο εκάστοτε ontology expert.

Τέλος, η οντολογία πέρα από τα στοιχεία που περιλαμβάνει δεν πρέπει να περιέχει λογικά λάθη. Για τον λόγο αυτό θα χρησιμοποιηθεί ένας reasoner, ο Openllet reasoner (1.1.5.1. Reasoner). Ο Openllet είναι ένας reasoner της OWL 2 γραμμένος σε Java και σχεδιάστηκε πάνω στον Pellet reasoner [37] .

Παρακάτω φαίνεται μια γραφική αναπαράσταση της οντολογίας χρησιμοποιώντας το VOWL plugin του Protégé, όπως και τα entities σε δενδρική δομή.



Εικόνα 4.9 : Οπτική αναπαράσταση της οντολογίας με Protégé-VOWL



Εικόνα 4.10 : Δενδρική αναπαράσταση της οντολογίας με Protégé

4.2.1. Υλοποιημένη οντολογία

Στη συνέχεια φαίνεται όλη η υλοποιημένη οντολογία. Τα στοιχεία που βρίσκονται στη δεύτερη στήλη είναι sub-class του τελευταίου στοιχείου της πρώτης στήλης, τα στοιχεία της τρίτης sub-class του τελευταίου στοιχείου της δεύτερης και αντίστοιχα στην τέταρτη στήλη. Τα individuals μιας κλάσης τοποθετούνται στην ίδια γραμμή με την κλάση που ανήκουν αλλά σε διπλανή στήλη. Οι κλάσεις της οντολογίας σημειώνονται με ● ενώ με τα individuals με ◆. Έτσι, για παράδειγμα η κλάση "Demographics" έχει sub-class το "BodyMassIndex" που έχει individuals τα "normalWeight", "obesity", "overweight", "underweight".

Πίνακας 4-2 : Υλοποιημένη οντολογία

● Demographics			
	● BodyMassIndex	◆ normalWeight ◆ obesity ◆ overWeight ◆ underWeight	
	● Education		
		● EducationLevel	◆ OtherEducationLevel ◆ Primary ◆ Secondary ◆ Tertiary
	● Ethnicity	◆ American_Indian_or_Alaska_Native ◆ Asian ◆ Black_or_African_American ◆ Caucasian_or_White ◆ Hispanic_/_Latino ◆ Native_Hawaiian_or_Other_Pacific_Islander	
	● Gender		
		● Female	
		● Male	
		● OtherGender	
	● Occupation		
		● Employed	
		● OtherOccupationStatus	
		● Retired	
		● Student	
		● Underage	
		● Unemployed	
● Diagnosis			
● Examination			
	● Outcome		
		● OutcomeFormat	
		● OutcomeUnit	
		● OutcomeValue	
	● OutcomeAssessment		
	● TestName		

		antiThrombin	
		cholesterol	
		fribrinogen	
		haemoglobin	
		plateletCount	
		prothrombinTime	
		totalWhiteCellCount	
	TestType		
		Biopsy	
		BloodTest	
		MedicalImagingTest	
		OcularTest	
		OralTest	
		OtherTest	
		UrineTest	
HealthcareVisit			
Interventions			
	Chemotherapy		
		chemotherapyReason	
	Medication		
		interventionReason	
		pharmaceuticalDrug	
	Radiation		
		radiationReason	
	Surgery		
		surgeryReason	
Person			
	Doctor		
		FamilyDoctor	
		Specialist	
			''ENT_(Ear,_Nose,_and_Throat)''
			Dentist
			Dermatologist
			Endocrinologist
			Gastroenterologist
			Gynecologist
			Hematologist
			Nephrologist
			Neurologist
			Ophthalmologist
			OtherSpecialist
			Psychiatrist
			Pulmonologist
	Gender		
		Female	
		Male	
		OtherGender	
	Nurse		
	Other		
	Patient		

Pregnancy			
	PregnancyOutcome		
		Abortion	
			Induced_abortion
			Spontaneous_abortion_/_Miscarriage
		OtherPregnancyOutcome	
		Single	
		Stillborn	
		Twins	
Region			
SmokingStatus	Ex-Smoker Never_Smoker Smoker		

4.3. Υλοποίηση Back-End

Το Back-End υλοποιήθηκε με χρήση του Spring Boot. Αφού το standalone Spring Boot Microservice του Spring Boot εκκινήσει κανονικά, εξυπηρετεί αιτήματα στο localhost στην πόρτα που έχει οριστεί στο application properties, δηλαδή, στην προκείμενη, την 8090. Οι κλήσεις που εξυπηρετεί το back-end είναι δύο, το simpleSearch και το semanticSearch.

Και στις δύο περιπτώσεις, στην JenaReasoner.java κλάση επεξεργάζονται τα περασμένα ως παράμετροι πεδία-tags και δημιουργείται ένα TagArray.java που για κάθε ένα από αυτά περιλαμβάνει ένα tagGroup το οποίο περιλαμβάνει τα ίδια και (στην περίπτωση του semantic search) όσα προκύπτουν μέσω του εμπλουτισμού για κάθε ένα από αυτά. Επίσης, και στις δύο περιπτώσεις, αφού αναλυθούν τα πεδία, γίνεται μια κλήση στη βάση της MongoDB, η οποία επιστρέφει όλες τις εγγραφές-documents που περιέχουν κάποιο από τα tags. Τέλος, τα αποτελέσματα βαθμολογούνται και ταξινομούνται με βάση αυτή τη βαθμολογία και επιστρέφονται στο front-end.

Η μόνη, αλλά σημαντική διαφορά του semanticSearch με το simpleSearch, είναι το γεγονός πως πριν γίνει κλήση στη βάση, χρησιμοποιείται η υλοποιημένη οντολογία για την εύρεση tags σχετικών με τα ήδη υπάρχοντα με αποτέλεσμα την εμπλουτισμένη αναζήτηση στη βάση.

4.3.1. DiplomaServiceApplication.java

Αυτή η κλάση είναι η κλάση που περιέχει την main method για την εκτέλεση του Spring-Boot standalone.

Το **@SpringBootApplication** annotation της κλάσης υποδεικνύει μια configuration κλάση που δηλώνει μία ή περισσότερες @Bean μεθόδους. Το annotation αυτό είναι ένα βολικό annotation ισοδύναμο με τη δήλωση των[27] :

- **@EnableAutoConfiguration**: Υποδεικνύει στο Spring Boot να αρχίσει να προσθέτει beans βάσει των ρυθμίσεων του classpath, άλλα beans και διάφορες property ρυθμίσεις. Για παράδειγμα, αν το spring-webmvc είναι στο classpath, αυτό το annotation σημειώνει την εφαρμογή ως web εφαρμογή και ενεργοποιεί στοιχειώδεις συμπεριφορές [28] . Εν τέλει, ενεργοποιεί τον μηχανισμό του auto-configuration του Spring Application Conext, προσπαθώντας να μαντέψει και να ρυθμίσει beans που πιθανώς θα χρειαστούν [29] .
- **@ComponentScan**: Υποδεικνύει στο Spring να κοιτάξει για άλλα components, configurations και services στο com/example package [28] .
- **@Configuration**: Σημειώνει την κλάση ως πηγή για ορισμούς των beans για το application context [28] .

4.3.2. AsyncConfiguration.java

Η AsyncConfiguration ορίζει και παραμετροποιεί το Multi-threading της εφαρμογής. Συγκεκριμένα η κλάση είναι ορισμένη ως:

```
@Configuration
@EnableAsync
public class AsyncConfiguration {
```

Το annotation **@Configuration** υποδηλώνει πως η κλάση είναι πηγή για bean ορίσματα για το application context.

Το **@EnabledAsync** ενεργοποιεί τη δυνατότητα του Spring να εκτελεί **@Async** μεθόδους σε ένα thread pool στο παρασκήνιο. Έτσι, όταν καλείται η μέθοδος μια κλάσης με το @Async annotation, δημιουργείται (σύμφωνα με την παραμετροποίηση όπως φαίνεται και παρακάτω) ένα καινούριο thread για την εξυπηρέτησή της.

Η κλάση επίσης προσαρμόζει τον Executor ορίζοντας ένα νέο bean. Η μέθοδος ονομάζεται taskExecutor, αφού αυτό είναι το όνομα της μεθόδου που ψάχνει το Spring [32] .

Στην προκείμενη περίπτωση, περιορίζεται ο αριθμός των threads που εκτελούνται ταυτόχρονα στα 20 και το μέγεθος της ουράς (queue) στα 300

```
executor.setCorePoolSize(20);
executor.setMaxPoolSize(20);
executor.setQueueCapacity(300);
```

Ο **ThreadPoolExecutor** θα προσαρμόσει αυτόματα το μέγεθος του pool σύμφωνα με τα όρια που έχουν τεθεί από το **corePoolSize** και **maximumPoolSize**. Όταν υποβάλλεται μια νέα διεργασία

στην μέθοδο execute(java.lang.Runnable) και τρέχουν λιγότερα από *corePoolSize* threads, δημιουργείται ένα καινούριο thread για να εξυπηρετήσει το αίτημα, ακόμα και αν άλλα ενεργά thread είναι αδρανή. Αν υπάρχουν περισσότερα από *corePoolSize* αλλά λιγότερα από *maximumPoolSize* threads που τρέχουν, καινούριο thread θα δημιουργηθεί μόνο αν η ουρά (queue) είναι γεμάτη.

Θέτοντας στα *corePoolSize* και *maximumPoolSize* την ίδια τιμή, δημιουργείται ένα thread pool καθορισμένου μεγέθους. [33]

4.3.3. MyController.java

Η MyController, είναι η κλάση του Controller που εξυπηρετεί το Front-End.

Το **@RestController** είναι ένα composed annotation το οποίο συνδυάζει τα **@Controller** και **@ResponseBody** για να υποδηλώσει έναν controller του οποίου η κάθε μέθοδος κληρονομεί το **@ResponseBody** annotation και συνεπώς γράφει απευθείας στο response body [34] . Δηλώνει επίσης, στο Spring πως πρέπει να γυρνάει το επιστρεφόμενο String απευθείας πίσω σε αυτόν που έκανε το αίτημα [35] .

Είναι επίσης, γνωστό ως stereotype annotation. Παραθέτει στοιχεία για όποιον διαβάζει τον κώδικα και για το Spring πως η κλάση (που είναι annotated) έχει έναν συγκεκριμένο ρόλο. Σε αυτή την περίπτωση η κλάση είναι ένας web @Controller οπότε το Spring θα το λάβει υπόψιν όταν επεξεργάζεται εισερχόμενα web αιτήματα [35] .

Στη συνέχεια πάνω από κάθε endpoint (simple για simple search και semantic για semantic search) έχουν προστεθεί τα annotations :

```
@CrossOrigin(origins = "*")
@RequestMapping(value = "/simple", method = RequestMethod.GET)
```

και

```
@CrossOrigin(origins = "*")
@RequestMapping(value = "/semantic", method = RequestMethod.GET)
```

αντίστοιχα.

Το Cross-Origin Resource Sharing (CORS) είναι ένας μηχανισμός που χρησιμοποιεί πρόσθετες HTTP επικεφαλίδες ώστε οι φυλλομετρητές να δώσουν στην web εφαρμογή που τρέχει σε ένα origin, πρόσβαση σε δεδομένα από άλλο origin. Η εφαρμογή κάνει cross-origin HTTP αιτήματα όταν αιτείται κάποιον πόρο με διαφορετικό origin (domain, πρωτόκολλο, πόρτα) από το δικό του. Για λόγους ασφαλείας, οι φυλλομετρητές περιορίζουν cross-origin HTTP αιτήματα από script, όπως στην περίπτωση του front-end που γίνεται χρήση του Fetch API [31] . Με το `@CrossOrigin(origins = "*")` annotation, επιτρέπεται η πρόσβαση από οποιδήποτε origin.

Το **@RequestMapping** annotation προσφέρει “routing” πληροφορία. Ενημερώνει το Spring πως κάθε HTTP request με path “/” πρέπει να συνδεθεί με τις "/simple" και "/semantic" μεθόδους.[2] Με τη χρήση αυτού του annotation, προσδιορίζεται το path το οποίο πρέπει να κληθεί από το front end για την υλοποίηση του κάθε service του controller το οποίο ορίζεται ως RequestMethod.GET.

Έτσι αν κάποιος σε έναν φυλλομετρητή ή μέσω ενός app καλέσει το
`localgost:8090/simple`
θα γίνει μια κλήση προς το standalone που θα ανακατευθυνθεί στην υπηρεσία
`public String simple`

ενώ αντίστοιχα το
`localgost:8090/semantic`
στην
`public String semantic`

Αμφότερες οι υπηρεσίες-μέθοδοι, είναι απαραίτητο να κληθούν από το front-end με 7 παραμέτρους, με ονόματα tag1,tag2,tag3,tag4,tag5,tag6,tag7. Έτσι, ενδεικτικές κλήσεις θα είναι οι:

http://127.0.0.1:8090/simple?tag1=Doctor&tag2=&tag3=&tag4=&tag5=&tag6=&tag7=
http://127.0.0.1:8090/semantic?tag1=Doctor&tag2=&tag3=&tag4=&tag5=&tag6=&tag7=

όπου ενώ χρησιμοποιείται μόνο ένα όρισμα-tag, το *tag1=Doctor*, ορίζονται και όλα τα υπόλοιπα με κενή τιμή.

Αφού πραγματοποιηθεί η κλήση, οι παράμετροι μπαίνουν σε έναν πίνακα και γίνεται η κλήση της αντίστοιχης μεθόδου (simpleSearch ή semanticSearch) ενός **JenaReasoner** object. Αφού, όπως φαίνεται και παρακάτω, οι μέθοδοι simpleSearch και semanticSearch του **JenaReasoner** είναι ορισμένες ως **@Async**, κάθε request του front-end δημιουργεί καινούριο thread.

4.3.4. TagObject.java

Το **TagObject** είναι μια βοηθητική κλάση για το **JenaReasoner**. Χρησιμοποιείται για την αποθήκευση κάθε tag, το οποίο θα ανήκει μέσα σε ένα tag group, και περιλαμβάνει το όνομα του, *name,* και τον βαθμό που θα του καταχωρηθεί, *rank*.

```
public String name ;
public int rank;
```

4.3.5. TagArray.java

```
public ArrayList<ArrayList<TagObject>> tagGroup = new ArrayList<ArrayList<TagObject>>();
public int numberOfArgs = 0;
public boolean[] isUsed = new boolean[7];
```

Για κάθε αναζήτηση, αφού δημιουργείται καινούριο **JenaReasoner** object, δημιουργείται και καινούριο **TagArray** object. Αυτό περιέχει πληροφορίες για τα εισαχθέντα από τον χρήστη tags. Αποτελείται από ένα Array από Arrays (tagGroup) που περιέχουν στοιχεία **TagObject**, όπως επίσης τον αριθμό των εισαχθέντων tags (numberOfArgs) και το βοηθητικό Boolean array isUsed.

Στην περίπτωση του simpleSearch, το εμφωλευμένο Array περιλαμβάνει μόνο ένα στοιχείο, αυτό που τοποθετήθηκε με την κλήση του endpoint.

Στην περίπτωση του semanticSearch, το εμφωλευμένο Array με πρώτο στοιχείο αυτό που τοποθετήθηκε με την κλήση του endpoint (parent), γεμίζει όλα τα συγγενικά στοιχεία αυτού όπως προκύπτουν από τον εμπλουτισμό (βλ. 3.1. Εμπλουτισμός Στοιχείων προς αναζήτηση) μέσω της οντολογίας.

Συνεπώς, το tagGroup περιέχει arrays, που το κάθε array αντιπροσωπεύει ένα group από tags με πρώτο αυτό του χρήστη και (στην περίπτωση του semantic search) συγγενείς του tag αυτού. Για παράδειγμα, αν το endpoint κληθεί με παραμέτρους *Smoker*, *Doctor*. Προκύπτουν:

Πίνακας 4-3 : SimpleSearch : argArray για input tags Smoker, Doctor

argArray	
tagGroup1	Smoker
tagGroup2	Doctor

Πίνακας 4-4 : SemanticSearch : argArray για input tags Smoker, Doctor

argArray	
tagGroup1	Smoker, SmokingStatus, Ex-Smoker Never_Smoker
tagGroup2	Doctor, FamilyDoctor, Specialist, Person

Όπου το tag Smoker (Individual) έχει ως Class το *SmokingStatus* και ως siblings τα *Ex-Smoker*, *Never_Smoker* και το tag *Doctor* (Class) έχει ως subclasses τα *FamilyDoctor* και *Specialist* ενώ ως superclass το *Person*.

Τα επιμέρους στοιχεία του κάθε tagGroup είναι τύπου **TagObject** (όπως έχει οριστεί στο TagObject.java). Έτσι, το κάθε tag-συγγενής συνοδεύεται από το όνομα και τον βαθμό του. Σημαντικό κομμάτι της υλοποίησης είναι το γεγονός πως το κάθε group είναι εξαρχής ταξινομημένο σε φθίνουσα σειρά με βάση τον βαθμό των **TagObjects**. Αυτό επιτυγχάνεται μέσα στην semanticSearch, μέθοδο της **JenaReasoner** κλάσης, και είναι απαραίτητο για την dynamicGrade μέθοδο.

Κατά συνέπεια στην περίπτωση του εισαχθέντος tag που είναι individual, το tag group του θα έχει την μορφή (για όσα από τα παρακάτω στοιχεία υπάρχουν):

Πίνακας 4-5 : Μορφή στοιχείων του tag group ενός individual (semantic search)

tagGroup	Individual	Direct Class	Sibling 1	Sibling 2	Sibling 3	.....	Superclass 1	Superclass 2	Superclass 3	.....

και αντίστοιχα στην περίπτωση του εισαχθέντος tag που είναι class, το tag group του θα έχει τη μορφή (για όσα από τα παρακάτω στοιχεία υπάρχουν)

Πίνακας 4-6 : Μορφή στοιχείων του tag group ενός class (semantic search)

tagGroup	Class	Direct Subclass	Individual 1	Individual 2	Individual 3	...	Superclass 1	Superclass 2	Superclass 3	..

Αντιθέτως, στην περίπτωση του simple search, το tag group θα έχει τη μορφή:

Πίνακας 4-7 : Μορφή στοιχείων του tag group σε simple search

tagGroup	Class/Individual									

Το κάθε νέο εμφωλευμένο Array (tagGroup) αρχικοποιείται με την κλήση της addParentTag που εισάγεται το αρχικό tag (π.χ *Smoker* ή *Doctor*) και ο βαθμός του. Στη συνέχεια, τα υπόλοιπα tags εισάγονται με χρήση της addChildTag με βάση το index του στοιχείου πατέρα (π.χ index του *Smoker* είναι 0 ενώ του *Doctor* είναι 1).

Το πρώτο στοιχείο του κάθε εμφωλευμένου Array επιστρέφεται με την κλήση της getParent μεθόδου, ενώ όλα τα στοιχεία μπορούν να προσπελαστούν με χρήση των:

```
public int getChildRankByIndex(int parentIndex, int index)
```
και
```
public String getChildNameByIndex(int parentIndex, int index)
```

που επιστρέφουν το όνομα και τον βαθμό του στοιχείου αντίστοιχα, με βάση το tag group στο οποίο βρίσκεται και το ποιο κατά σειρά στοιχείο ζητείται.

Έτσι, εν τέλει τα tag groups για tags *Smoker*,*Doctor* είναι :

Πίνακας 4-8 : Tag groups για tags Smoker και Doctor.

tagGroup1	Smoker 100	SmokingStatus 80	Ex-Smoker 71	Never_smoker 71
tagGroup2	Smoker 100	FamilyDoctor 80	Specialist 80	Person 61

με τους αντίστοιχους βαθμούς όπως προκύπτουν από την Βαθμολόγηση στοιχείων και blueprints.

Η **sizeOfArgs** μέθοδος επιστρέφει τον αριθμό των εμφωλευμένων Arrays, δηλαδή, το πλήθος των εισαχθέντων tags.

Το **isUsed** array χρησιμοποιείται στην private static float dynamicGrade (String blueprint, TagArray tagGroups) και θα αναλυθεί περαιτέρω εκεί.

4.3.6. ResultDocument.java

```
public String doc;
public float grade;
public String id;
```

Η κλάση **ResultDocument** αντιπροσωπεύει τα στοιχεία ενός blueprint τα οποία επιστρέφονται από το microservice στο front-end. Αυτό το object περιέχει 3 στοιχεία :
doc: Το οποίο περιέχει το blueprint όπως περιέχεται στην MongoDb.
grade: Το οποίο περιέχει τον βαθμό του blueprint όπως προέκυψε μετά τις διαδικασίες στην JenaReasoner κλάση.
id: Το id που έχει δοθεί στο object μέσα στην MongoDb. Αυτό επιστρέφεται κυρίως για να φανεί η μοναδικότητα του blueprint, αλλά και για να μπορεί κάποιος να αναζητήσει άμεσα το blueprint στη βάση με το id του.

Οι μέθοδοι που υλοποιούνται στην κλάση αποτελούν τον constructor του ResultDocument object, καθώς και get/set μεθόδους.

4.3.7. MongoApp.java

Η κλάση αυτή είναι υπεύθυνη για τη σύνδεση με τη βάση της MongoDb και το query σε αυτήν και παράλληλα επεξεργάζεται τα documents που επιστρέφονται πριν με τη σειρά της τα επιστρέψει στην κλάση **JenaReasoner**, η οποία την κάλεσε.

4.3.7.1. Δημιουργία σύνδεσης

Αρχικά, η κλάση ορίζεται ως:

```
@Async
public class MongoApp {
```

συνεπώς, κάθε object τύπου MongoApp αντιστοιχεί και σε διαφορετικό thread.

Στη συνέχεια, ορίζονται οι απαραίτητες παράμετροι, οι οποίες θα χρειαστούν για τη σύνδεση με την MongoDb:

```
static final String MONGO_DATABASE_NAME = "myDatabase";
static final String MONGO_COLLECTION_NAME = "blueprints";
static final String MONGO_IP = "localhost";
static final String MONGO_PORT = "27017";
```

Το όνομα της βάσης είναι **myDatabase** και χρησιμοποιείται το collection **blueprints**. Η βάση είναι προσβάσιμη μέσω του **localhost** (127.0.0.1) στην πόρτα **27017**.

Ο constructor της κλάσης δημιουργεί έτσι μια σύνδεση με την MongoDb, η οποία τερματίζεται μετά την εκτέλεση του query στη βάση και την πρώιμη επεξεργασία των blueprints που επιστρέφονται.

```
mongoClient.close();
```

Τα παραπάνω σε συνδυασμό με το `@Async` annotation κάνουν κάθε MongoApp object να βρίσκεται σε ξεχωριστό thread, το οποίο έχει τη δική του σύνδεση με τη βάση της MongoDb.

4.3.7.2. Query

Με την κλήση της μεθόδου `public void query(TagArray tag_array)` γίνεται το query στη βάση της MongoDb. Η κλάση ανακτά τα blueprints που περιέχουν οποιοδήποτε από όλα τα στοιχεία που υπάρχουν στο **TagArray** object, είτε πρόκειται για τα αρχικά, που αναζήτησε ο χρήστης (εισαχθέντα), είτε για συγγενείς τους, που προστέθηκαν αργότερα (από εμπλουτισμό).
Για να το πετύχει αυτό, το query κάνει find στη βάση επιστρέφοντας τα blueprints που έχουν στα tags τους κάποιο από τα στοιχεία του **TagArray**. Συνεπώς, για κάθε στοιχείο **TagArray** φτιάχνεται ένα μεμονωμένο BasicDBObject object στην μορφη [{"tags": "*TagArrayName*"}.

```
for( ArrayList<TagObject> y : tag_array.argArray) {
	for (TagObject h : y) {
		BasicDBObject tmpquery = new BasicDBObject("tags",h.getName());
		advQuery.add(tmpquery);
	}
}
```

Για παράδειγμα για input tag *Doctor*, προκύπτει το tagArray:
Doctor → FamilyDoctor → Specialist → Person

Οπότε, για να βρεθούν τα blueprints που περιέχουν οποιοδήποτε από τα παραπάνω 4 tags, προκύπτει ο τελικός κορμός του query με τη βοήθεια του $or ορίσματος:

{"$or": [{"tags": "Doctor"}, {"tags": "FamilyDoctor"}, {"tags": "Specialist"}, {"tags": "Person"}]}

και το query που εν τέλει θα τρέξει στην MongoDB θα έχει την μορφή:

db.blueprints.find({"$or": [{"tags": "Doctor"}, {"tags": "FamilyDoctor"}, {"tags": "Specialist"}, {"tags": "Person"}]})

Στη συνέχεια, χρησιμοποιούνται οι μεταβλητές `List<String> ids` και `List<String> blueprints` για την αποθήκευση του κάθε blueprint και του id του. Για παράδειγμα, δύο από τα blueprints που θα επιστραφούν για το tag *Doctor* μέσω της semantic αναζήτησης θα είναι τα:

Πίνακας 4-9 : Παράδειγμα blueprints όπως επιστρέφονται από την MongoDb

{
"_id" :
ObjectId("5f465d52a93bfd39b4105997"),
"tags" : [
"Unemployed",
"surgeryReason",
"FamilyDoctor"
]
}
	{
	"_id" :
	ObjectId("5f465d52a93bfd39b4105969"),
	"tags" : [
	"UrineTest",
	"Stillborn",
	"Specialist",
	"Male",
	"Twins"
	]
	}

Από αυτά τα blueprints, προκύπτουν τα arrays `List<String> ids` και `List<String> blueprints`:

Πίνακας 4-10 : List blueprints όπως προκύπτει από το παράδειγμα των επιστραφέντων blueprints

blueprints	
"[Unemployed,surgeryReason,FamilyDoctor]"	"[UrineTest,Stillborn,Specialist,Male,Twins]"

Πίνακας 4-11 : List ids όπως προκύπτει από το παράδειγμα των επιστραφέντων blueprints

ids	
"5f465d52a93bfd39b4105997"	"5f465d52a93bfd39b4105969"

τα οποία ανακτώνται από την **retrieveAndGradeBlueprint** της **JenaReasoner** κλάσης για περαιτέρω επεξεργασία.

4.3.8. JenaReasoner.java

Το object **JenaReasoner** με τη βοήθεια των **TagObject**, **TagArray**, **ResultDocument**, **MongoApp** υλοποιεί όλη την λογική και για τα δύο είδη αναζήτησης.

4.3.8.1.1. public static List<String> removeNs (String thestring, String myns)

Κάθε στοιχείο αποθηκεύεται στην οντολογία έχοντας ως πρόθεμα το namespace του. Η μέθοδος αυτή, «καθαρίζει» όλα τα στοιχεία που επιστρέφει η οντολογία, Class ή Individual, αφαιρώντας τους οποιοδήποτε namespace προηγείται. Τα namespaces που αφαιρεί είναι τα:

```
String owlns = "http://www.w3.org/2002/07/owl#";
String rdfns = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
String rdfns2 = "http://www.w3.org/2000/01/rdf-schema#";
```

τα οποία είναι συνδεδεμένα με την OWL και το RDF. Αφαιρεί επίσης, το namespace της υλοποιημένης οντολογίας καθαυτής:
http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#

Έτσι, για παράδειγμα από το στοιχείο
http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Person
θα παραμείνει μόνο το
Person

Πολυπλοκότητα
Η πολυπλοκότητα της μεθόδου είναι O(1).

4.3.8.1.2. public static List<String> ontologyIterator (final Iterator<?> I, String myns)

Η μέθοδος αυτή, με όρισμα έναν Iterator, επιστρέφει ένα ArrayList με όλα τα στοιχεία του Iterator. Αν μέσα στα στοιχεία που θα περαστούν, είναι τα στοιχεία της οντολογίας Thing ή/και Nothing τα αγνοεί. Επίσης, τα στοιχεία επιστρέφονται χωρίς το namespace (περιέχεται η μέθοδος public static List<String> removeNs (String thestring, String myns)).

Πολυπλοκότητα

Η πολυπλοκότητα της μεθόδου είναι O(N).

4.3.8.2. Αναζήτηση

4.3.8.2.1. public static void simpleSearch(final String[] args,boolean ignoreNs)

Η μέθοδος simpleSearch είναι μία από τις 2 κεντρικές μεθόδους που καλούνται από τον MyController.java.

Κλήση της μεθόδου

Αρχικά, το **@Async** annotation επιτρέπει τη δημιουργία μιας ασύγχρονης μεθόδου. Έτσι, για κάθε κλήση της μεθόδου δημιουργείται και καινούριο thread.

Όπως κλήθηκε από τον **MyController**, η **simpleSearch** καλείται με όρισμα έναν πίνακα από arguments-tags. Στη συνέχεια, γίνεται η επεξεργασία των arguments-tags, αφού η κλήση του controller γίνεται με όλα τα πεδία να έχουν κάποια τιμή, έστω και κενό string. Διαχωρίζονται λοιπόν, τα arguments που έχουν τιμή με αυτά που δεν έχουν και προστίθενται στο object: `TagArray tagsArray` το οποίο περιλαμβάνει τα tagGroups (στην περίπτωση του simple search κάθε tagGroup θα έχει μόνο ένα στοιχείο-tag, το εισαχθέν).

```
tagsArray.addParentTag(val);
```

retrieveAndGradeBlueprint

Στη συνέχεια, γίνεται η ανάκτηση-βαθμολόγηση-ταξινόμηση των blueprints με την κλήση της private static List<ResultDocument> retrieveAndGradeBlueprint (TagArray tag_array, int numberOfInputTags) και τα αποτελέσματα εκχωρούνται σε μια λίστα τύπου ResultDocument.java.

```
List<ListResultDocument> ret2 = retrieveAndGradeBlueprint(tagsArray,numOfTags);
```

JSON

Τέλος, δημιουργείται ένα String το οποίο περιέχει σε JSON μορφή τα αποτελέσματα που εν τέλει θα θέσουν την μεταβλητή simple_search_blueprints για τον χρήστη.

```
setSimple_search_blueprints(jsoned);
```

Το τελικό json αποτελείται από 2 στοιχεία-arrays, το blueprint και το id. Τα arrays αυτά αντιστοιχίζονται ένα προς ένα, δηλαδή, το πρώτο id αντιστοιχεί στο πρώτο blueprint και ομοιώς τα υπόλοιπα μεταξύ τους. Έτσι, εν τέλει, προκύπτει ένα JSON της μορφής:

```
{
"blueprint" : ["Single, Outcome, Doctor] - 57.735027 " , " [Male, Doctor, Dermatologist] - 57.735027 "],
"id" : ["5f465d52a93bfd39b4105e54"," 5f465d52a93bfd39b4105a2e"]
}
```

το οποίο υποδεικνύει πως το blueprint [Single, Outcome, Doctor] με id 5f465d52a93bfd39b4105e54 έχει rank 57.735027, και το blueprint [Male, Doctor, Dermatologist] με id 5f465d52a93bfd39b4105a2e έχει rank 57.735027.

Πολυπλοκότητα

Για αριθμό εισαχθέντων tags A (μέχρι και 7 στην υλοποίηση), που το κάθε ένα θα δημιουργήσει ένα tag Group μεγέθους M και θα υπολογιστεί ο βαθμός blueprints με αριθμό tags B (μέχρι και 7 στην υλοποίηση) και πλήθος N, η πολυπλοκότητα της simple αναζήτησης προκύπτει:

Πέρα από το κομμάτι της επεξεργασίας των εισαχθέντων tags, που έχει πολυπλοκότητα O(A) και το κομμάτι της private static List<ResultDocument> retrieveAndGradeBlueprint (TagArray tag_array, int numberOfInputTags), που έχει πολυπλοκότητα O(NMB) (όπως φαίνεται και παρακάτω), η simple και η semantic αναζήτηση δεν έχουν άλλα κοινά. Εν τέλει, προκύπτει η πολυπλοκότητα της simple αναζήτησης O(NMB). Καθώς, όμως, τα tag group στην simple αναζήτηση περιέχουν μόνο 1 όρισμα, το εισαχθέν tag, M=1 και άρα η πολυπλοκότητα καταλήγει O(NB) .

4.3.8.2.2. public static void semanticSearch

Η μέθοδος semanticSearch είναι μία από τις 2 κεντρικές μεθόδους που καλούνται από τον Controller.

Κλήση της μεθόδου

Αρχικά, το **@Async** annotation επιτρέπει τη δημιουργία μιας ασύγχρονης μεθόδου. Έτσι, για κάθε κλήση της μεθόδου δημιουργείται και καινούριο thread.

Ακριβώς αντίστοιχα με την **simpleSearch**, η **semanticSearch** καλείται με όρισμα έναν πίνακα από arguments-tags. Και σε αυτή την μέθοδο επεξεργάζονται-διαχωρίζονται τα arguments που έχουν τιμή από αυτά που δεν έχουν και προστίθενται στα `List<String> inputTags` και `TagArray tagsArray`, το οποίο περιλαμβάνει τα tagGroups (στην περίπτωση του semantic search κάθε tagGroup θα έχει το εισαχθέν στοιχείο-tag αλλά και όσα θα προκύψουν μέσω του εμπλουτισμού).

```
inputTags.add(val);
tagsArray.addParentTag(val);
```

Εμπλουτισμός με βάση την οντολογία

Το στάδιο αυτό, ένα από τα πιο βασικά στάδια του semantic search, περιλαμβάνει τον εμπλουτισμό των tags-arguments, που έχουν έρθει ως input με άλλα tags που συνδέονται με τα πρώτα μέσω της υλοποιημένης οντολογίας (βλ. Εμπλουτισμός Στοιχείων προς αναζήτηση).

Κάθε στοιχείο-tag, εφόσον περιέχεται στην οντολογία μπορεί να είναι ένα class ή ένα individual. Αναλόγως ποιο από τα δύο είναι, υπάρχει μια σειρά κανόνων, όπου ανακτώνται οι

συγγενείς του στοιχείου αυτού και βαθμολογούνται. Οι κανόνες είναι τοποθετημένοι με σειρά φθίνουσας σημασίας (βλ. Βαθμολόγηση στοιχείων και blueprints). Έτσι, ο συγγενής ενός στοιχείου που ικανοποιεί τον πρώτο κανόνα, βαθμολογείται υψηλότερα από οποιονδήποτε συγγενή, οποιουδήποτε στοιχείου που ικανοποιεί τον δεύτερο κανόνα. Οι κανόνες και οι βαθμολογίες είναι:

Πίνακας 4-12 : Κανόνες, βαθμολογίες και μέθοδοι που χρησιμοποιούνται για τον εμπλουτισμό ενός individual (π.χ Asian)

relationship	rank	Method used
individual	100	tagInd
Class of individual	80	directClasses = TagInd.listOntClasses
Sibling of individual	71	directClasses.listInstances(true)
Superclass of individual's class	61	DirectClasses.listSuperClasses(false)

Πίνακας 4-13 : Κανόνες, βαθμολογίες και μέθοδοι που χρησιμοποιούνται για τον εμπλουτισμό ενός class (π.χ Doctor)

Relationship	rank	Method used
Class	100	tagClass
Subclass of class	80	TagClass.listSubClasses(true)
Individual of class	71	tagClass.listInstances(true)
Superclass of class	59	tagClass.listSuperClasses(false)

Το αποτέλεσμα κάθε μεθόδου του Jena framework (βλ. Apache Jena), που τρέχει πάνω σε κάποιο OntClass ή Individual, επιστρέφει μια λίστα από στοιχεία-συγγενείς, τα οποία προσπελαύνονται με τη βοήθεια της μεθόδου ontologyIterator. Έτσι, αρχικά, χρησιμοποιείται το framework της Jena για όλα τα απαραίτητα στοιχεία-συγγενείς και μετά το κάθε ένα τοποθετείται με τον αντίστοιχο βαθμό στο tag group του (αυτό που έχει ως πατέρα το εισαχθέν στοιχείο-tag).

Individual

Στην περίπτωση του Individual: το individual ανήκει σε μία κλάση (1η συγγένεια). Έτσι, θα έχει και siblings, δηλαδή, individuals της κλάσης στην οποία ανήκει (2η συγγένεια), όπως και υπερκλάσεις, δηλαδή, κλάσεις στις οποίες ανήκει η κλάση του individual (3η συγγένεια). Έτσι, αφού βρεθεί η κλάση στην οποία ανήκει το individual, εκτελείται ο ontologyIterator για την εύρεση των siblings και των superclasses.

Άρα, αρχικά, βρίσκεται η κλάση στην οποία ανήκει το individual (όπου το `true` υποδηλώνει πως θα επιστραφεί μόνο η κλάση στην οποία ανήκει το individual και όχι υπερ-κλάσεις αυτής):

```
OntClass directClass = tagInd.getOntClass(true);
```

```
tagsArray.addChildTag(index, removeNs(directClass.toString(),mynamespace),
RANK_IND_DIR_CLASS);
```

Αφού εκχωρηθεί στο tagsArray, επιστρέφονται όλα τα instances της κλάσης, άρα siblings του individual (όπου το **true** υποδηλώνει πως θα επιστραφούν μόνο τα Instances της directClass και όχι Instances υποκλάσεών της):

```
List<String> tmp2 = ontologyIterator(directClass.listInstances(true),mynamespace);
```

και κάθε sibling εισέρχεται στο tagsArray και εκχωρείται η βαθμολογία του.

```
for (String m : tmp2) {
	tagsArray.addChildTag(index, m, RANK_IND_SIBLINGS);
}
```

Και αντίστοιχα για τα superclasses της κλάσης στην οποία ανήκει το individual (όπου το **false** υποδηλώνει πως δεν θα επιστραφούν μόνο οι άμεσες υπερ-κλάσεις, αλλά και υπερ-κλάσεις των υπερ-κλάσεων):

```
tmp2 = ontologyIterator(directClass.listSuperClasses(false),mynamespace);
for (String m : tmp2) {
	tagsArray.addChildTag(index, directClass.toString(), RANK_IND_DIR_SUPERCLASS);
}
```

Class

Πιο απλά στην περίπτωση του class: το class έχει κάποια subclasses από τα οποία επιστρέφονται μόνο αυτά που είναι 1ου βαθμού, δεν είναι δηλαδή, subclass ενός subclass του class. Στη συνέχεια, επιστρέφονται τα individuals του class ($2^{η}$ συγγένεια) και τέλος, όλες οι υπερκλάσεις, δηλαδή, κλάσεις στις οποίες ανήκει το class αλλά και οι υπερ-κλάσεις τους ($3^{η}$ συγγένεια). Έτσι, εκτελείται ο ontologyIterator για την εύρεση των παραπάνω συγγενών του class.

Ανάκτηση των υπο-κλάσεων της κλάσης και την εκχώρηση στον tagsArray μαζί με τη βαθμολογία τους (όπου το **true** υποδηλώνει πως θα επιστραφούν μόνο τα άμεσα subclasses του class και όχι τα subclasses των subclasses):

```
List<String> tmp = ontologyIterator(tagClass.listSubClasses(true),mynamespace);
for (String m : tmp) {
	tagsArray.addChildTag(index, m, RANK_CLASS_SUBCLASS);
}
```

Ανάκτηση των individuals (όπου το **true** υποδηλώνει πως θα επιστραφούν μόνο τα individuals του tag Class και όχι τα individuals των υποκλάσεών του):

```
tmp = ontologyIterator(tagClass.listInstances(true),mynamespace);
for (String m : tmp) {
	tagsArray.addChildTag(index, m, RANK_CLASS_INDIVIDUALS);
}
```

Ανάκτηση των υπερ-κλάσεων (όπου το **false** υποδηλώνει πως θα επιστραφούν και τα superclasses των superclasses του class, δηλαδή, όχι μόνο τα άμεσα superclasses του):

```
tmp = ontologyIterator(tagClass.listSuperClasses(false),mynamespace);
for (String m : tmp) {
	tagsArray.addChildTag(index, m, RANK_CLASS_DIR_SUPERCLASS);
}
```

Κάθε φορά που βρίσκεται κάποιο καινούριο tag (π.χ η υπο-κλάση ενός tag), δημιουργείται ένα **TagObject** και προστίθεται στο **TagArray** object: `tagsArray`. Αυτό το **TagObject** έχει ένα index αναλόγως με το ποιο είναι το αρχικό tag από το οποίο προέκυψε, το όνομά του και το rank του.

retrieveAndGradeBlueprint

Στη συνέχεια, αφού ολοκληρωθεί η διαδικασία του εμπλουτισμού των tags, γίνεται η ανάκτηση-βαθμολόγηση-ταξινόμηση των blueprints με την κλήση της private static List<ResultDocument> retrieveAndGradeBlueprint (TagArray tag_array, int numberOfInputTags) και τα αποτελέσματα εκχωρούνται σε μια λίστα τύπου ResultDocument.java.

```
List<ListResultDocument> ret2 = retrieveAndGradeBlueprint(tagsArray,numOfTags);
```

JSON

Τέλος, δημιουργείται ένα String, το οποίο περιέχει σε JSON μορφή τα αποτελέσματα που εν τέλει θα θέσουν την μεταβλητή semantic_search_blueprints για τον χρήστη.

```
setSemantic_search_blueprints(jsoned);
```

Το τελικό json αποτελείται από 2 στοιχεία-arrays, το blueprint και το id. Τα arrays αυτά αντιστοιχίζονται ένα προς ένα, δηλαδή, το πρώτο id αντιστοιχεί στο πρώτο blueprint και ομοιώς τα υπόλοιπα μεταξύ τους. Έτσι, τελικά, προκύπτει ένα JSON της μορφής:

{
"blueprint" : ["Single, Outcome, Doctor] - 57.735027 " , " [Male, Doctor, Dermatologist] - 57.735027 "],
"id" : ["5f465d52a93bfd39b4105e54"," 5f465d52a93bfd39b4105a2e"]
}

το οποίο υποδεικνύει πως το blueprint [Single, Outcome, Doctor] με id 5f465d52a93bfd39b4105e54 έχει rank 57.735027, και το blueprint [Male, Doctor, Dermatologist] με id 5f465d52a93bfd39b4105a2e έχει rank 57.735027.

Πολυπλοκότητα

Για αριθμό εισαχθέντων tags A (μέχρι και 7 στην υλοποίηση), όπου το κάθε ένα θα δημιουργήσει ένα tag Group μεγέθους M και θα υπολογιστεί ο βαθμός blueprints με αριθμό tags B (μέχρι και 7 στην υλοποίηση) και πλήθος N, η πολυπλοκότητα της semantic αναζήτησης προκύπτει:

Πέρα από το κομμάτι της επεξεργασίας των εισαχθέντων tags, που έχει πολυπλοκότητα O(A) και το κομμάτι της private static List<ResultDocument> retrieveAndGradeBlueprint (TagArray tag_array, int numberOfInputTags), που έχει πολυπλοκότητα O(NMB), όπως φαίνεται και παρακάτω, η semantic αναζήτηση έχει επιπλέον και το κομμάτι εμπλουτισμού των tagGroups και του reasoning της οντολογίας. Επίσης, σημαντική διαφορά είναι πως το πλήθος των blueprints N της semantic αναζήτησης είναι κατά πολύ μεγαλύτερο από αυτό του πλήθους της simple αναζήτησης, καθώς έχουν επιστραφεί blueprints, που περιέχουν πολλά περισσότερα tags, τα οποία περιέχονται στα tag groups.

Ενώ, πρακτικά οι διεργασίες, που πραγματοποιούνται πάνω στην οντολογία, δεν επηρεάζουν ιδιαίτερα την πολυπλοκότητα της semantic search, αποτελούν έναν σημαντικό παράγοντα καθυστέρησης (Σύνοψη ποσοτικού ελέγχου), αφού η καθυστέρηση της semantic search φαίνεται να είναι υπερδιπλάσια της simple search, καθώς αυξάνεται ο χρόνος εκτέλεσης ακόμα και κατά 100ms.

4.3.8.3. Βαθμολόγηση

4.3.8.3.1. private static List<ResultDocument> retrieveAndGradeBlueprint (TagArray tag_array, int numberOfInputTags)

Η retrieveAndGradeBlueprint, όπως υποδηλώνει και το όνομά της, επιστρέφει από τη βάση τα blueprints που χρειάζονται και τα βαθμολογεί.

Ανάκτηση Blueprints από MongoDb

Η retrieveAndGradeBlueprint ανακτά τα blueprints που περιέχουν οποιοδήποτε από όλα τα tags τα οποία υπάρχουν στο **TagArray** object, είτε πρόκειται για τα αρχικά, που αναζήτησε ο χρήστης, είτε για συγγενείς τους, που προστέθηκαν αργότερα.
Για να το πετύχει αυτό χρησιμοποιεί την MongoApp.java κλάση:

```
MongoApp mongoQuery = new MongoApp();
mongoQuery.query(tag_array);
blueprintsString = mongoQuery.getBlueprints();
idsString = mongoQuery.getIds();
```

Με την εκτέλεση των παραπάνω εντολών δημιουργείται και κλείνει ένα καινούριο thread για την εξυπηρέτηση ενός νέου connection στην MongoDb, εκτελείται το query και επιστρέφονται τα blueprints και τα id τους στα αντίστοιχα blueprintsString και idsString arrays.

Προετοιμασία για βαθμολόγηση blueprints

Το κάθε blueprint που έχει ανακτηθεί από την MongoDb θα μπει στον πίνακα `List<String> blueprintsString` και το id του blueprint στον πίνακα `List<String> idsString`.

Κάθε blueprint θα βαθμολογηθεί

```
float dynamic_grade = dynamicGrade(blueprintsString.get(blueprintIndex), tag_array);
```

ο βαθμός αυτός θα διαιρεθεί με την τετραγωνική ρίζα του πλήθους των στοιχείων του blueprint

```
int numbOfBluepTags = blueprintsString.get(blueprintIndex).split(",").length;
float grade = (float) (dynamic_grade / (Math.pow(numbOfBluepTags, 1.0/ROOT_TO_DIVIDE_GRADE
)));
```

και θα δημιουργηθεί ένα καινούριο στοιχείο στην `ArrayList<ResultDocument> result = new ArrayList<ResultDocument>();` λίστα με τα **ResultDocument** objects, που θα περιέχει το blueprint, τον βαθμό του και το id του

```
result.add(newResultDocument(blueprintsString.get(blueprintIndex),
grade,idsString.get(blueprintIndex)));
```

Τέλος, ο πίνακας που περιέχει τα **ResultDocument** objects θα ταξινομηθεί με κλήση της sort
`sort (result,0,result.size()-1);`

Πολυπλοκότητα

Για αριθμό εισαχθέντων tags A (μέχρι και 7 στην υλοποίηση), όπου το κάθε ένα θα δημιουργήσει ένα tag Group μεγέθους M και θα υπολογιστεί ο βαθμός blueprints με αριθμό tags B (μέχρι και 7 στην υλοποίηση) και πλήθος N, η πολυπλοκότητα προκύπτει (όπως φαίνεται και παρακάτω βάσει της πολυπλοκότητας της dynamicGrade) : N*O(dynamicGrade)= O(NMB).

4.3.8.3.2. private static float dynamicGrade (String blueprint, TagArray tagGroups)

Ο σχεδιασμός με τον οποίο βαθμολογείται το κάθε blueprint ακολουθεί τη λογική του δυναμικού προγραμματισμού. Αυτό γίνεται καθώς πρέπει να αντιμετωπιστούν διάφορες προκλήσεις που προκύπτουν κατά τη βαθμολόγηση.
Για παράδειγμα, έστω πως με εισαχθέντα tags

Doctor, Smoker, Nurse

πρέπει να βαθμολογηθούν τα blueprint

FamilyDoctor, Smoker, Ex-Smoker
Person, OtherPregnancyOutcome, Nurse, FamilyDoctor

Στην περίπτωση του semantic search, το **TagArray** για τα εισαχθέντα tags και τους αντίστοιχους βαθμούς τους θα είναι:
Doctor (100)→ Specialist (80)→ FamilyDoctor (80)→ Person (61)
Smoker (100)→ SmokingStatus (80)→ Never_Smoker (71)→ Smoker (71)→ Ex-Smoker (71)
Nurse (100)→ Person (61)

(Η επανεμφάνιση του Smoker μέσα στο TagArray δεν ελέγχεται για λόγους απλότητας και πολυπλοκότητας και δεν επηρεάζει καθόλου το αποτέλεσμα.)

Από κάθε tag group, δηλαδή, εισαχθέν tag και αυτά που προκύπτουν από αυτό tag, θα πρέπει να χρησιμοποιείται μόνο ένα tag για τη βαθμολόγηση του blueprint. Έτσι, δεν γίνεται να χρησιμοποιηθεί και το *Smoker* και το *Ex-Smoker* για το blueprint 1) του παραδείγματος.

Επίσης, πρέπει να γίνει η σωστή επιλογή για το από ποιο tag group θα βαθμολογηθεί το κάθε tag. Για παράδειγμα στο blueprint 2) το tag *Person* μπορεί να βαθμολογηθεί και από το tag group του *Doctor* και από το tag group του *Nurse*.

Αν βαθμολογηθεί από το tag group του *Doctor*

Grade= 61 (Person, Doctor group) + 100 (Nurse, Nurse group) =161

Αν βαθμολογηθεί από το tag group του *Nurse*

Grade= 61 (Person, Nurse group) + 80 (FamilyDoctor, Doctor group) = 141

Προκύπτει λοιπόν, η ανάγκη εύρεσης του ιδανικού συνδυασμού για μέγιστη βαθμολογία. Η διαδικασία που εκτελεί η dynamicGrade είναι η εξής:

Δεδομένου ενός blueprint, η διαδικασία ψάχνει για ταύτιση κάποιου tag του blueprint με κάποιο tag από τα tag groups. Ταυτόχρονα, το tag group το οποίο εξετάζεται κάθε φορά πρέπει να μην χρησιμοποιείται ήδη για τη βαθμολόγηση κάποιου άλλου στοιχείου του blueprint. Αυτό ελέγχεται με το

```
if (tagGroups.isUsed[tagGroupIndex])
	continue;
```

Όταν βρεθεί το tag (και το tag group) που ταυτίζεται με κάποιο tag του blueprint, τότε το tag group τίθεται ως used `tagGroups.isUsed[tagGroupIndex]=true;`. Ο βαθμός του tag (από το tag group) εισέρχεται σε μια μεταβλητή που κρατάει τον βαθμό του blueprint, `float tmpgrade = tagRank;`, το tag αφαιρείται από το blueprint και καλείται ξανά η dynamicGrade, προσθέτοντας τον βαθμό που επιστρέφει στην μεταβλητή με τον προηγούμενο βαθμό `tmpgrade += dynamicGrade(tmpBlueprint , tagGroups);`.

Τέλος, ο βαθμός που παραμένει είναι ο μέγιστος

```
if (tmpgrade>maxgrade)
	maxgrade=tmpgrade;
```

Για παράδειγμα:

Για τη βαθμολόγηση του 1) [FamilyDoctor, Smoker, Ex-Smoker] με tags *Doctor*, *Smoker* και tagGroups:

Doctor (100)→ Specialist (80)→ FamilyDoctor (80)→ Person (61)

Smoker (100)→ SmokingStatus (80)→ Never_Smoker (71)→ Smoker (71)→ Ex-Smoker (71)

Προκύπτει ο παρακάτω πίνακας ακολουθιακής εκτέλεσης. Η κολώνα "**Επίπεδο nesting**" καθορίζει το πόσο "βαθιά" βρίσκεται η διαδικασία με τις εμφωλευμένες κλήσεις της dynamicGrade. Η κολώνα "**Ενέργεια**" τις ενέργειες που γίνονται σε εκείνο το βήμα της εκτέλεσης, η "**blueprint**" το blueprint που βρίσκεται σαν όρισμα στην εκάστοτε dynamicGrade, το "**tmp-grade**" την τιμή της μεταβλητής tmp-grade σε εκείνο το βήμα της εκτέλεσης και τα 2 groups υποδηλώνουν την τιμή του array isUsed για το *Doctor* tagGroup και το *Smoker* tagGroup αντίστοιχα.
(βλ. Πίνακας 3-5 : Βαθμολογία ανά κανόνα εμπλουτισμού για individual-tag, Πίνακας 3-7 : Βαθμολογία ανά κανόνα εμπλουτισμού για class-tag και κανόνες βαθμολόγησης)

Πίνακας 4-14 : Παράδειγμα ακολουθιακής εκτέλεσης dynamicGrade(tags={*Doctor,Smoker*})

Επίπεδο nesting	**Ενέργεια**	**blueprint**	**tmp-grade**	**Doctor Group**	**Smoker Group**
0	Αρχική κλήση dynamicGrade([FamilyDoctor, Smoker, Ex-Smoker])	[FamilyDoctor, Smoker, Ex-Smoker]	0	Not used	Not used
0	Αντιστοίχιση FamilyDoctor Doctor Group → used	[FamilyDoctor, Smoker, Ex-Smoker]	0+80 =80	Used	Not used
1	Κλήση dynamicGrade([Smoker, Ex-Smoker])	[Smoker, Ex-Smoker]	0	Used	Not used
1	Αντιστοίχιση Smoker Smoker Group → used	[Smoker, Ex-Smoker]	0+100= 100	Used	Used
2	Κλήση dynamicGrade([Ex-Smoker])	[Ex-Smoker]	0	Used	Used
2	Επιστροφή(0) tmpgrade=0+100 Smoker Group → Not used	[Smoker, Ex-Smoker]	100	Used	Not used
1	tmpgrade (100) >maxgrade (0) maxgrade=100	[Smoker, Ex-Smoker]	100	Used	Not used
1	Αντιστοίχιση με 2ο Smoker Smoker Group → used	[Smoker, Ex-Smoker]	0+71 =71	Used	Used
2	Κλήση dynamicGrade([Ex-Smoker])	[Ex-Smoker]	0	Used	Used
1	Επιστροφή(0) tmpgrade=0+71 Smoker Group → not used	[Smoker, Ex-Smoker]	71	Used	Not used
1	tmpgrade (71) <maxgrade (100) maxgrade=100	[Smoker, Ex-Smoker]	0	Used	Not used
1	Εύρεση αντιστοίχισης με Ex-Smoker Smoker Group → used	[Smoker, Ex-Smoker]	0+71 =71	Used	Used
2	Κλήση dynamicGrade([Smoker])	Smoker]	0	Used	Used
1	Επιστροφή(0) tmpgrade=0+71 Smoker Group → not used	[Smoker, Ex-Smoker]	71	Used	Not used

1	tmpgrade (71) <maxgrade (100) maxgrade=100	[Smoker, Ex-Smoker]	0	Used	Not used
0	Επιστροφή(100) tmpgrade=100+80 Doctor Group → not used	[FamilyDoctor, Smoker, Ex-Smoker]	80	Used	Not used
0	tmpgrade (180) >maxgrade (0) maxgrade=180	[FamilyDoctor, Smoker, Ex-Smoker]	180	Not used	Not used
0	Αντιστοίχιση Smoker Smoker Group → used	[FamilyDoctor, Smoker, Ex-Smoker]	0+100=100	Not used	Used
1	Κλήση dynamicGrade([FamilyDoctor, Ex-Smoker])	[FamilyDoctor, Ex-Smoker]	0	Used	Not used
1	Αντιστοίχιση FamilyDoctor Doctor Group → used	[FamilyDoctor, Ex-Smoker]	0+80=80	Used	Used
2	Κλήση dynamicGrade([Ex-Smoker])	[Ex-Smoker]	0	Used	Used
1	Επιστροφή(0) tmpgrade=0+80 Doctor Group → Not used	[FamilyDoctor, Ex-Smoker]	80	Used	Not used
1	tmpgrade (80) >maxgrade (0) maxgrade=80	[FamilyDoctor, Ex-Smoker]	80	Used	Not used
0	Επιστροφή(0) tmpgrade=100+80 Smoker Group → Not used	[FamilyDoctor, Smoker, Ex-Smoker]	180	Not used	Not used
0	tmpgrade (180) >maxgrade (0) maxgrade=180	[FamilyDoctor, Smoker, Ex-Smoker]	180	Not used	Not used
0	Αντιστοίχιση 2ου Smoker Smoker Group → used	[FamilyDoctor, Smoker, Ex-Smoker]	0+71=71	Not used	Used
1	Κλήση dynamicGrade([FamilyDoctor, Ex-Smoker])	[FamilyDoctor, Ex-Smoker]	0	Used	Not used
1	Αντιστοίχιση FamilyDoctor Doctor Group → used	[FamilyDoctor, Ex-Smoker]	0+80=80	Used	Used
2	Κλήση dynamicGrade([Ex-Smoker])	[Ex-Smoker]	0	Used	Used
1	Επιστροφή(0) tmpgrade=0+80 Doctor Group → Not used	[FamilyDoctor, Ex-Smoker]	80	Used	Not used
1	tmpgrade (80) >maxgrade (0) maxgrade=80	[FamilyDoctor, Ex-Smoker]	80	Used	Not used
0	Επιστροφή(0) tmpgrade=71+80 Smoker Group → Not used	[FamilyDoctor, Smoker, Ex-Smoker]	151	Not used	Not used

0	tmpgrade (151)<maxgrade (180) maxgrade=180	[FamilyDoctor, Smoker, Ex-Smoker]	180	Not used	Not used
0	Αντιστοίχιση Ex-Smoker Smoker Group → used	[FamilyDoctor, Smoker, Ex-Smoker]	0+71 =71	Not used	Used
1	Κλήση dynamicGrade([FamilyDoctor, Ex-Smoker])	[FamilyDoctor, Ex-Smoker]	0	Used	Not used
1	Αντιστοίχιση FamilyDoctor Doctor Group → used	[FamilyDoctor, Ex-Smoker]	0+80=80	Used	Used
2	Κλήση dynamicGrade([Ex-Smoker])	[Ex-Smoker]	0	Used	Used
1	Επιστροφή(0) tmpgrade=0+80 Doctor Group → Not used	[FamilyDoctor, Ex-Smoker]	80	Used	Not used
1	tmpgrade (80)>maxgrade (0) maxgrade=80	[FamilyDoctor, Ex-Smoker]	80	Used	Not used
0	Επιστροφή(0) tmpgrade=71+80 Smoker Group → Not used	[FamilyDoctor, Smoker, Ex-Smoker]	151	Not used	Not used
0	tmpgrade (151) >maxgrade (180) maxgrade=180	[FamilyDoctor, Smoker, Ex-Smoker]	180	Not used	Not used
-	Επιστροφή(180)	-	-	-	-

Σημείωση

Ο διαχωρισμός των tags μέσα στην blueprint μεταβλητή γίνεται με τη βοήθεια String operations. Συγκεκριμένα, για τον διαχωρισμό των tags που βρίσκονται μέσα στο blueprint αφαιρούνται τα brackets και χωρίζονται μεταξύ τους με βάση το κόμμα.

```
tmpBlueprint = tmpBlueprint.replace("[", "");
tmpBlueprint = tmpBlueprint.replace("]", "");
String[] elementdoc = tmpBlueprint.split(", ");
```

Επίσης, για την εμφωλευμένη κλήση του blueprint, εφόσον πρέπει να αφαιρεθεί το tag που ταίριαξε, εκτελούνται τα παρακάτω:

```
tmpBlueprint = tmpBlueprint.replace("["+stringIter+", ", "[");
tmpBlueprint = tmpBlueprint.replace(", "+stringIter+"]", "]");
tmpBlueprint = tmpBlueprint.replace(", "+stringIter+", ", ", ");
tmpBlueprint = tmpBlueprint.replace("["+stringIter+"]", "");
```

Για αυτούς τους λόγους, θα πρέπει να **μην** χρησιμοποιούνται στοιχεία της οντολογίας που να περιέχουν κόμμα “,” ή bracket “[“ , ”]” (βλ. Υλοποίηση Οντολογίας).

Πολυπλοκότητα

Για αριθμό εισαχθέντων tags A (μέχρι και 7 στην υλοποίηση), όπου το κάθε ένα θα δημιουργήσει ένα tag Group μεγέθους M και θα υπολογιστεί ο βαθμός ενός blueprint με αριθμό tags B (μέχρι και 7 στην υλοποίηση) η πολυπλοκότητα προκύπτει: O(A*M*B), αφού για κάθε στοιχείο του blueprint θα προσπελαστούν όλα τα tagGroups και όλα τα tags αυτών. Καθώς, όμως, το A ανεξάρτητα από την υλοποίηση θα είναι αρκετά μικρό, η πολυπλοκότητα εν τέλει είναι O(MB).

4.3.8.4. Ταξινόμηση

Αφού ολοκληρωθεί η βαθμολόγηση των Result Documents, αυτά ταξινομούνται για να επιστραφούν. Η ταξινόμηση γίνεται με χρήση της quicksort, η οποία αποτελείται από δύο μέρη, το sort και το partition. Τα Result Documents ταξινομούνται με βάση τον βαθμό του κάθε Document.

4.3.8.4.1. private static void sort(ArrayList<ResultDocument> res, int low, int high)

Η sort μέθοδος καλείται αναδρομικά, αφού καθορίσει το pivot στοιχείο με βοήθεια της partition, ξανακαλεί τον εαυτό της για να ταξινομήσει τα δύο υπο-μέρη του πίνακα.

```
sort(res, low, pi-1);
sort(res, pi+1, high);
```

4.3.8.4.2. private static int partition(ArrayList<ResultDocument> res, int low, int high)

Η partition μέθοδος αρχικά, επιλέγει ως pivot στοιχείο, το τελευταίο στοιχείο του υποπίνακα

```
float pivot = res.get(high).getGrade();
```

και στη συνέχεια για κάθε στοιχείο ελέγχει αν είναι μικρότερο ή μεγαλύτερο του pivot και τα ανα-διατάσσει. Τέλος, επιστρέφει το pivot στοιχείο.

Πολυπλοκότητα

Η quicksort αναδρομικά χωρίζει έναν πίνακα σε 2 μέρη με βάση ένα pivot στοιχείο. Όποιο στοιχείο του πίνακα είναι μικρότερο από το pivot στοιχείο, θα μεταφερθεί στο πρώτο μισό του πίνακα, ενώ όποιο είναι μεγαλύτερο, στο δεύτερο μισό. Έτσι, καθώς και αυτά τα 2 μέρη χωρίζονται σε άλλα υπομέρη μέχρι να μην χωρίζεται άλλο ο πίνακας, η quicksort ανα-διατάσει όλο τον πίνακα σε αύξουσα σειρά.

Η αναδιάταξη αυτή έχει πολυπλοκότητα O(N*lognN). Αυτό προκύπτει ως εξής: Αν υποθέσουμε πως πάντα το pivot στοιχείο είναι το χειρότερο δυνατό, δηλαδή, όχι η διάμεσος αλλά το μεγαλύτερο ή μικρότερο στοιχείο, τότε για κάθε διαχωρισμό του πίνακα (N διαχωρισμοί) θα πρέπει να προσπελαστούν όλα τα στοιχεία, άρα O(N^2).

Καθώς, όμως, αν επιλέγεται τυχαία το στοιχείο είναι απίθανο να συμβεί κάτι τέτοιο, μπορούμε να θεωρήσουμε πως κατά μέσο όρο θα επιλέγεται ως pivot στοιχείο κάποιο κοντά στο μέσον του πίνακα, τότε οι διαχωρισμοί που θα χρειαστεί να γίνουν είναι logN. Αυτό προκύπτει, καθώς ο πίνακας

μεγέθους N, για να διαχωριστεί σε μοναδιαία κομμάτια, θα χρειαστεί να προσπελαστεί ολόκληρος logN φορές (ταυτόσημο με το ύψος δυαδικού-δένδρου με N φύλλα).

5. ΑΠΟΤΕΛΕΣΜΑΤΑ

Με την ολοκλήρωση της υλοποίησης φαίνεται η διαφορά ανάμεσα στην simple αναζήτηση και την semantic αναζήτηση. Στα πλαίσια της παρούσας διπλωματικής εργασίας, όπως έχει αναφερθεί και νωρίτερα, χρησιμοποιήθηκε μια οντολογία βασισμένη σε ιατρικά δεδομένα. Επίσης, δημιουργήθηκαν 3000 blueprints στη βάση της MongoDb που περιέχουν όρους που απαντώνται στην οντολογία. Το κάθε blueprint, έχει τυχαίο αριθμό στοιχείων-tags από 3 έως και 7.
Αρχικά, θα εξεταστούν διαφορές σε όγκο δεδομένων και ταχύτητα εκτέλεσης.

5.1. Ποσοτικός έλεγχος

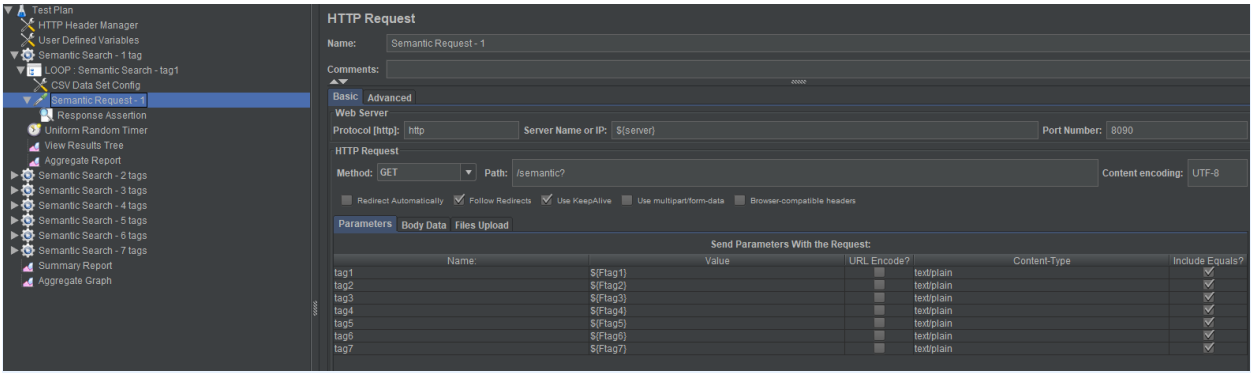
5.1.1. JMETER

Όσον αφορά τα ποσοτικά αποτελέσματα, αξίζει να εξεταστεί η διαφορά στη διάρκεια εκτέλεσης των δύο αναζητήσεων, αλλά και τη διαφορά στον όγκο των blueprints που επιστρέφονται.

Με τη βοήθεια του JMETER πραγματοποιήθηκαν μετρήσεις πάνω σε διάφορες μεθόδους της οντολογίας. Συγκεκριμένα, σχεδιάστηκαν δύο jmx αρχεία, ένα για την εκτέλεση 7000 simple αναζητήσεων και ένα για την εκτέλεση 7000 semantic αναζητήσεων.

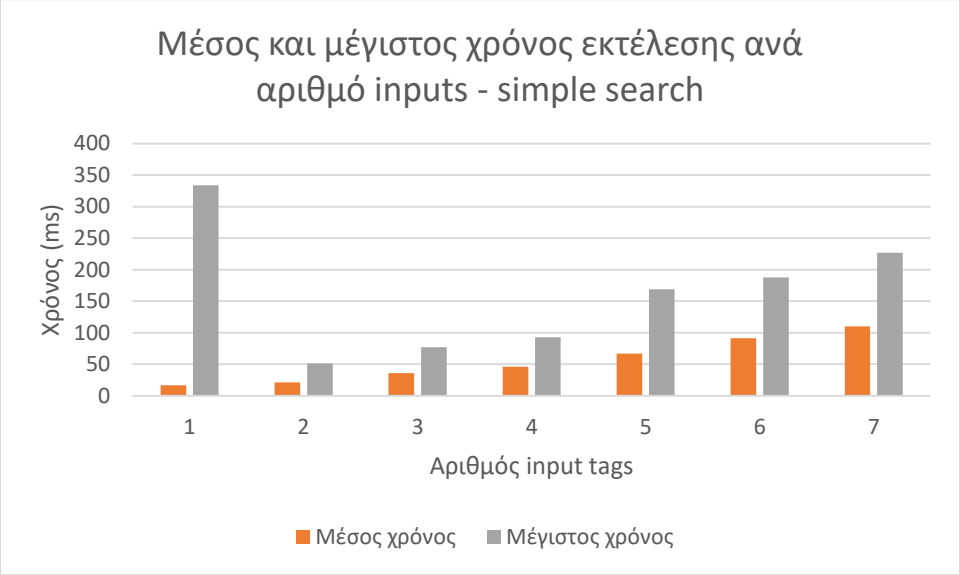
Οι 7000 αυτές αναζητήσεις χωρίστηκαν σε 7 thread groups των 1000 αναζητήσεων και για τα δύο αρχεία. Το $1^{ο}$ group αναζητήσεων περιέχει αναζητήσεις με 1 όρισμα, το $2^{ο}$ group αναζητήσεων περιέχει αναζητήσεις με 2 ορίσματα, το $3^{ο}$ με 3 κλπ. Το κάθε group εκτελείται λίγο μετά το προηγούμενο, ώστε τα logs να μπορούν εύκολα να ομαδοποιηθούν για κάθε group εκτέλεσης.

Επίσης, το κάθε thread group εκτελεί τις αναζητήσεις με 2 threads χωρίς κάποια καθυστέρηση. Εκτελούνται δηλαδή, 2 threads στο jmeter, όπου το κάθε ένα θα εκτελέσει 500 αναζητήσεις για κάθε thread group.

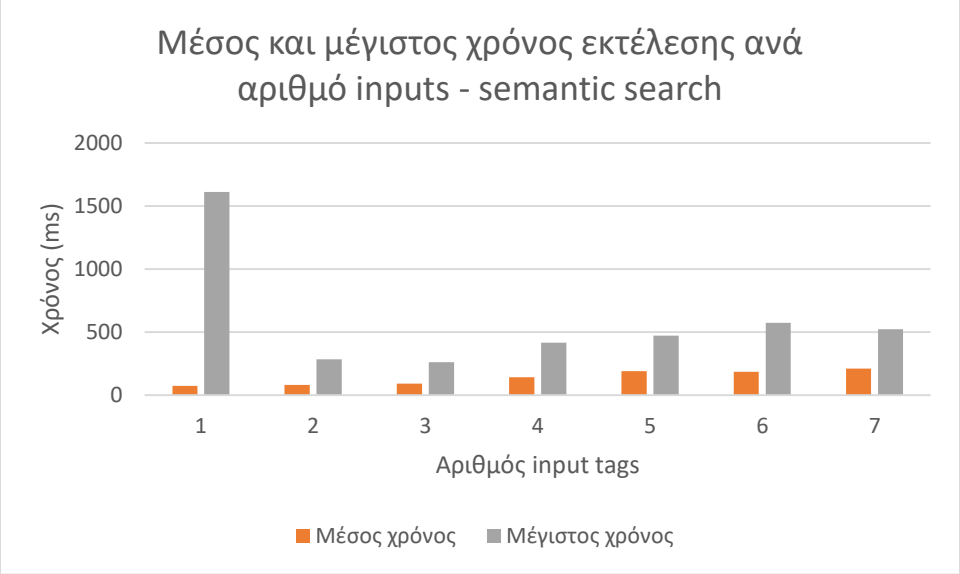


Εικόνα 5.1 Jmeter

Με την εκτέλεση του JMETER παρατηρήθηκε όντως η semantic search να διαρκεί μεγαλύτερο χρονικό διάστημα κατά μέσο όρο. Επίσης, παρατηρούνται ιδιαίτερα μεγάλες max τιμές. Αυτό οφείλεται κυρίως στο γεγονός πως αν η εφαρμογή παραμείνει ανενεργή για αρκετή ώρα, η δημιουργία connection με τη βάση της MongoDb και το query καθυστερούν παραπάνω. Επίσης, ένα από τα χαρακτηριστικά της Java είναι πως ο κώδικας έχει καλύτερη επίδοση όσο περισσότερο εκτελείται. Για αυτό τον λόγο, όταν εκτελούνται διάφορα benchmarks υπολογίζεται ένα μικρό warm-up period που να δίνει στον compiler τη δυνατότητα να παράξει ιδανικό κώδικα [36] . Παρ'όλα αυτά, οι όποιοι υπολογισμοί και γραφικές έγιναν αγνοώντας το κομμάτι του warm-up, δηλαδή, συμπεριλαμβάνοντας αυτές τις αρχικές καθυστερήσεις, καθώς υπάρχει περίπτωση η εφαρμογή να έχει περιόδους που να έχει "κρυώσει".



Εικόνα 5.2 Simple search Jmeter αποτελέσματα



Εικόνα 5.3 Semantic search Jmeter αποτελέσματα

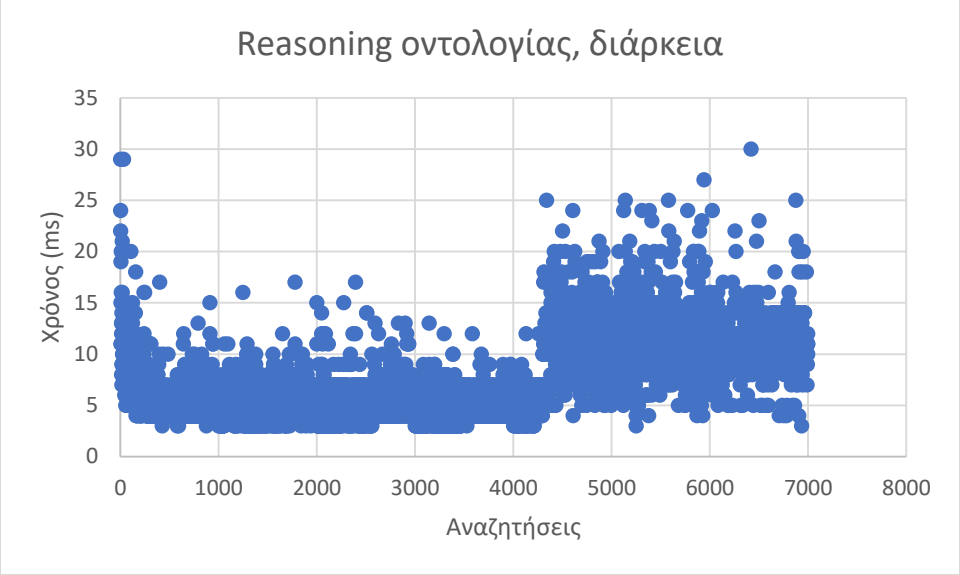
Η διαφορά στον χρόνο εκτέλεσης των δύο αναζητήσεων επικεντρώνεται σε 3 κομμάτια:

- Ενέργειες πάνω στην οντολογία (εμπλουτισμός και reasoning)
- Ανάκτηση και βαθμολόγηση blueprints (retrieveAndGradeBlueprints)
- Δημιουργία JSON String

5.1.2. Ενέργειες πάνω στην οντολογία

Οι ενέργειες που πραγματοποιούνται πάνω στην οντολογία περιλαμβάνουν το διάβασμα και το reasoning της οντολογίας,

```
model = ModelFactory.createOntologyModel(PelletReasonerFactory.THE_SPEC);
model.read(myont);
```



Εικόνα 5.4 Χρονική διάρκεια του reasoning της οντολογίας για κάθε ένα από τα 7000 request

καθώς και την εύρεση των συγγενών των tags

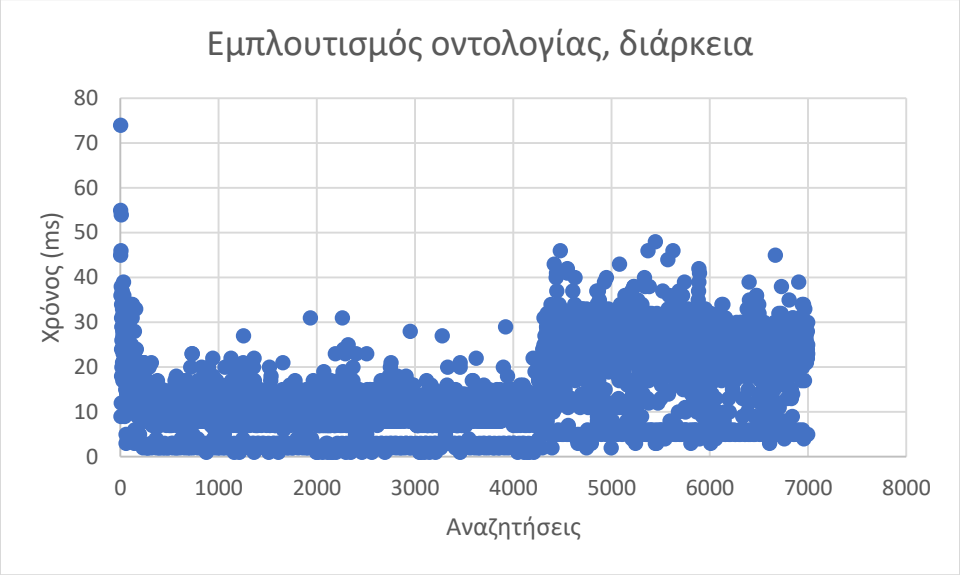
```
OntClass tagClass = model.getOntClass(mynamespace + tagIter);
```

και συγκεκριμένα στην περίπτωση που το tag είναι **individual**

```
Individual tagInd = model.getIndividual(mynamespace+tagIter);
OntClass directClass = tagInd.getOntClass(true);
List<String> tmp2 = ontologyIterator(directClass.listInstances(true),mynamespace);
tmp2 = ontologyIterator(directClass.listSuperClasses(false),mynamespace);
```

ενώ στην περίπτωση που το tag είναι **Class**

```
List<String> tmp = ontologyIterator(tagClass.listSubClasses(true),mynamespace);
tmp = ontologyIterator(tagClass.listSuperClasses(false),mynamespace);
tmp = ontologyIterator(tagClass.listInstances(true),mynamespace);
```



Εικόνα 5.5 Χρονική διάρκεια τουεμπλουτισμού της οντολογίας για κάθε ένα από τα 7000 request

Η διαδικασία της εισαγωγής και του reasoning της οντολογίας προέκυψε περίπου 8ms κατά μέσο όρο, ενώ η διαδικασία εμπλουτισμού των tags κατά μέσο όρο 16ms. Και οι δύο διαδικασίες που πραγματοποιούνται στην οντολογία, δεν προσθέτουν μεγάλη καθυστέρηση, ενώ και σε περιπτώσεις μεγαλύτερων οντολογιών οι διαφορές θα είναι μικρές.

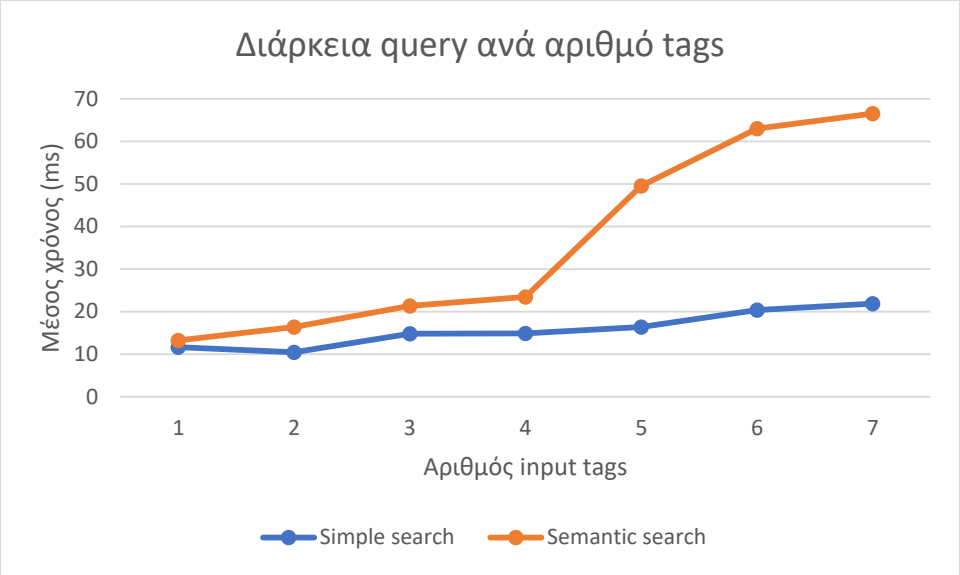
5.1.3. Ανάκτηση και βαθμολόγηση blueprints

Αισθητή είναι η διαφορά στην μέθοδο private static List<ResultDocument> retrieveAndGradeBlueprint (TagArray tag_array, int numberOfInputTags). Αυτή αποτελείται από 3 κύριες υπο-μεθόδους/διαδικασίες :

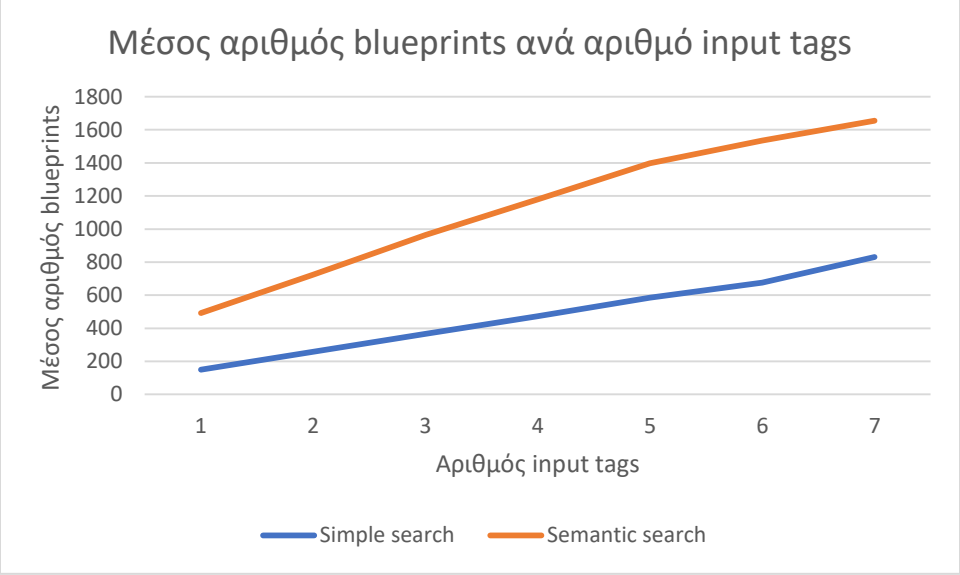
5.1.3.1. MongoDb query :

Στη διαδικασία του query, καθώς υπάρχει πολύ μεγαλύτερος αριθμός tags που αναζητώνται στη σημασιολογική αναζήτηση, αντίστοιχα θα επιστραφούν πολύ περισσότερα αποτελέσματα. Συγκεκριμένα, ενώ στην simple αναζήτηση επιστρέφονται κατά μέσο όρο 477 blueprints σε 15ms χρόνο, στην semantic επιστρέφονται 1135 blueprints σε 36ms χρόνο. Αντίθετα, το γεγονός ότι το query φαίνεται να διαρκεί περισσότερο, δεδομένου ότι επιστρέφει περισσότερα αποτελέσματα, είναι κάτι αναπόφεκτο αλλά θεμιτό, δεδομένου ότι στα πλαίσια των στόχων του προβλήματος που εξετάζεται είναι η επιστροφή περισσότερων blueprints.

Επίσης, η μείωση της αύξησης του αριθμού των αποτελεσμάτων που επιστρέφονται μετά τα 5 tags, όπως φαίνεται στο διάγραμμα Εικόνα 5.7, οφείλεται στο ότι στα 6 tags έχουν επιστραφεί ήδη παραπάνω από τις μισές εγγραφές της βάσης (παραπάνω από 1600/3000). Αν η βάση δεδομένων είχε περισσότερες εγγραφες, λογικά η αύξηση των επιστρεφόμενων αποτελεσμάτων θα ήταν με τον ίδιο ρυθμό.



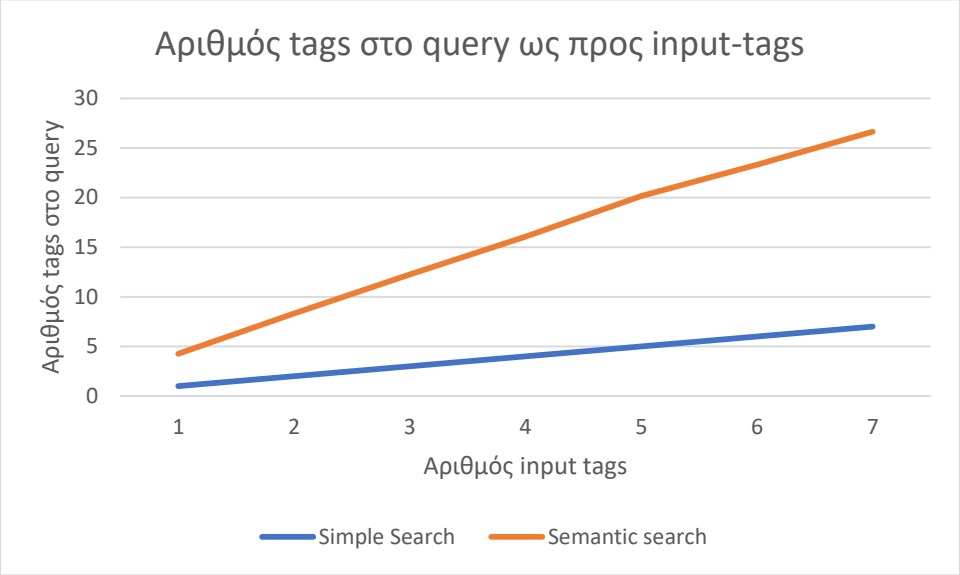
Εικόνα 5.6 Μέση χρονική διάρκεια εκτέλεσης select query για 1000 request ανά αριθμό tags



Εικόνα 5.7 Μέσος αριθμός blueprints που επιστρέφονται από το query για 1000 request ανά αριθμό tags

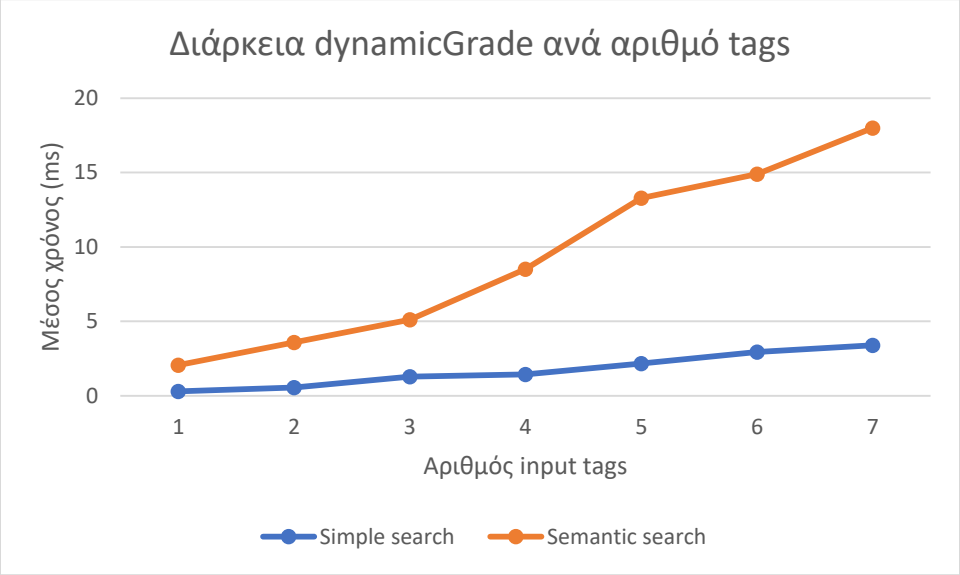
5.1.3.2. dynamicGrade :

Στη διαδικασία του dynamicGrade παρουσιάζεται διαφορά στο πλήθος των blueprints, αλλά και στον αριθμό των tags που προσπελαύνονται για να βαθμολογηθεί το κάθε blueprint. Συγκεκριμένα, ενώ στην simple αναζήτηση υπάρχουν κατά μέσο όρο 4 tags, δηλαδή, όσα έμπαιναν ως input εξαρχής, στην semantic υπάρχουν 16.



Εικόνα 5.8 Μέσος αριθμός tags που περιέχονται στο query για 1000 request ανά αριθμό tags

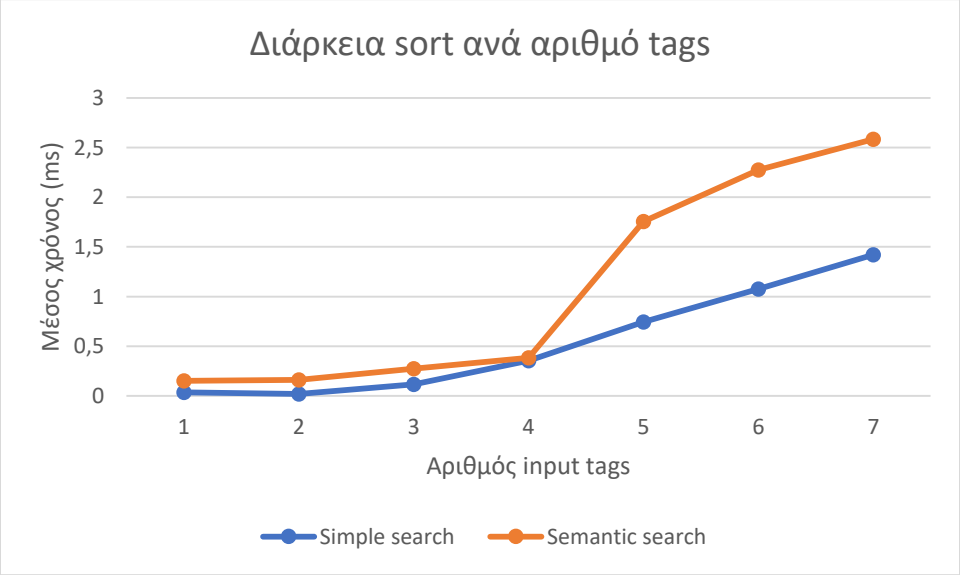
Η dynamicGrade ως εκ τούτου, ενώ στην simple αναζήτηση διαρκεί κατά μέσο όρο 2ms (η εκτέλεση του dynamicGrade σε όλα τα blueprints), στην semantic αναζήτηση διαρκεί 9ms. Γενικά δηλαδή, η dynamicGrade όσο αυξάνονται τα tags, αυξάνεται εκθετικά, αν και για μικρές οντολογίες ή και βάσεις δεδομένων (όπως στο παράδειγμα) δεν επηρεάζει αρκετά τη χρονική διάρκεια εκτέλεσης.



Εικόνα 5.9 Μέση διάρκεια εκτέλεσης αθροίσματος των dynamicGrade για 1000 request ανά αριθμό tags

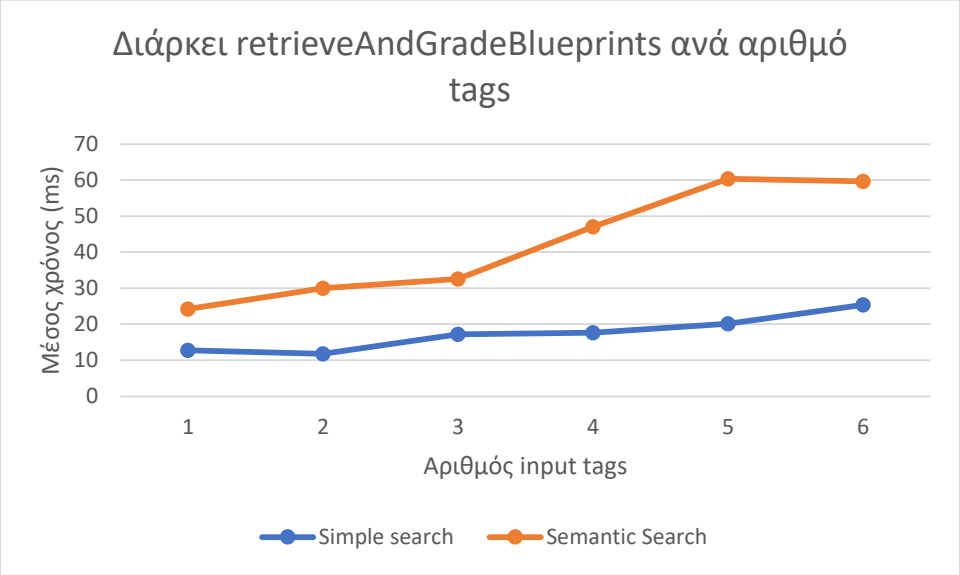
5.1.3.3. sort :

Στη διαδικασία του sort η μόνη διαφορά έγκειται στον αριθμό των blueprints που έχουν επιστραφεί. Φαίνεται πως στην semantic αναζήτηση, το μεγαλύτερο πλήθος blueprints κάνει την αναζήτηση να καθυστερεί κατά 1ms (κατά μέσο όρο), έναντι της simple αναζήτησης που καθυστερεί κατά 0.5ms (κατά μέσο όρο). Πρακτικά η ταξινόμηση διαρκεί το διπλάσιο στην semantic αναζήτηση, ουσιαστικά όμως, (στο παράδειγμα) ο όγκος των δεδομένων είναι τόσο μικρός που δεν έχει καμία σημασία.



Εικόνα 5.10 Μέση χρονική διάρκεια ταξινόμησης για 1000 request ανά αριθμό tags

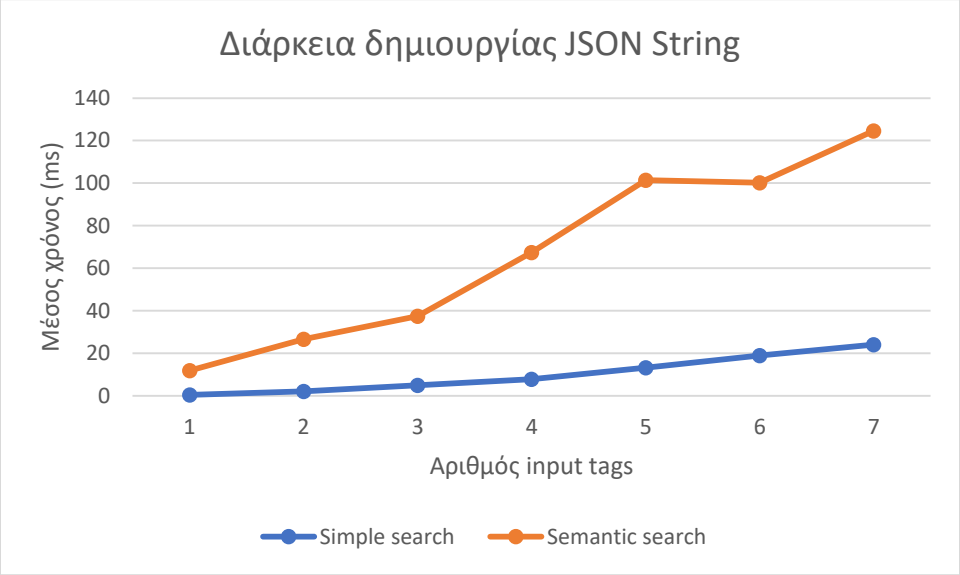
Εν τέλει, όλη η διαδικασία της retrieveAndGradeBlueprint στην simple αναζήτηση διαρκεί κατά μέσο όρο 28ms, ενώ στην semantic αναζήτηση κατά μέσο όρο 63ms. Επίσης, φαίνεται πως η μέθοδος δεν επηρεάζεται ούτε από τον αλγόριθμο της βαθμολόγησης, ούτε από το sorting, αλλά κυρίως από το query στην MongoDb. Βέβαια, όπως φαίνεται και στο γράφημα, όσο αυξάνεται ο αριθμός των tags, τόσο αυξάνεται και η διαφορά μεταξύ των δύο αναζητήσεων, όπου στα 6 και στα 7 tags η διαφορά είναι διπλάσια, με την simple μέθοδο να διαρκεί μέχρι και 30ms, ενώ η semantic μέχρι και 60ms.



Εικόνα 5.11 Μέση χρονική διάρκεια εκτέλεσης του query, των dynamicGrade και του sort για 1000 request ανά αριθμό tags

5.1.4. Δημιουργία JSON

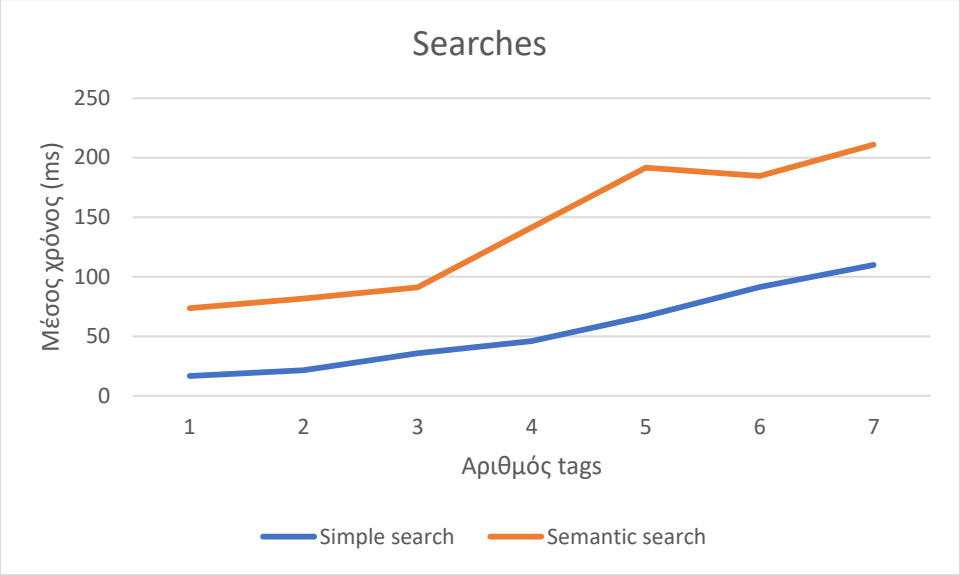
Τέλος, σημαντική είναι η διαφορά της διαδικασίας της δημιουργίας της String μεταβλητής που περιέχει το JSON, η οποία και στις δύο μεθόδους προσθέτει αρκετή καθυστέρηση και διαρκεί στην περίπτωση της simple αναζήτησης 10ms κατά μέσο όρο, ενώ στην περίπτωση της semantic αναζήτησης 67ms. Μάλιστα, στην περίπτωση των 7 tags η διαφορά φτάνει στο σημείο η jsonize να προσθέτει 125ms καθυστέρηση κατά μέσο όρο στα blueprints της semantic αναζήτησης, ενώ μόλις 24ms στα blueprints της simple.



Εικόνα 5.12 Μέση χρονική διάρκεια μετατροπής αποτελεσμάτων σε JSON μορφή για 1000 request ανά αριθμό tags

Συνοψίζοντας και αθροίζοντας τη διάρκεια των διαδικασιών, δηλαδή, reasoning οντολογίας, εμπλουτισμός οντολογίας, retrieveAndGradeBlueprint και JSON creation, η διαφορά μεταξύ simple αναζήτησης και semantic αναζήτησης είναι πως η simple αναζήτηση εκτελείται στα 55ms κατά μέσο όρο, ενώ η semantic στα 139ms κατά μέσο όρο. Οι τιμές αυτές δεν είναι ενδεικτικές, καθώς περιλαμβάνουν το ακραίο σενάριο του ενός tag. Φαίνεται δηλαδή, πως από τα 4 tags και πάνω, υπάρχει ραγδαία αύξηση της καθυστέρησης, η οποία αγγίζει μέχρι και τα 100ms διαφορά (κατά μέσο όρο).

Βέβαια, όπως ειπώθηκε παραπάνω, η κύρια διαφορά έγκειτα στο query που πραγματοποιείται στη βάση δεδομένων και στη διαδικασία της δημιουργίας του JSON-string. Παρ'όλα αυτά, για μια βάση δεδομένων των 3000 στοιχείων και για αναζητήσεις μέχρι 7 tags δεν είναι μεγάλη διάρκεια εκτέλεσης (λιγότερο από 1 δευτερόλεπτο).



Εικόνα 5.13 Μέση συνολική χρονική διάρκεια εκτέλεσης των αναζητήσεων για 1000 request ανά αριθμό tags

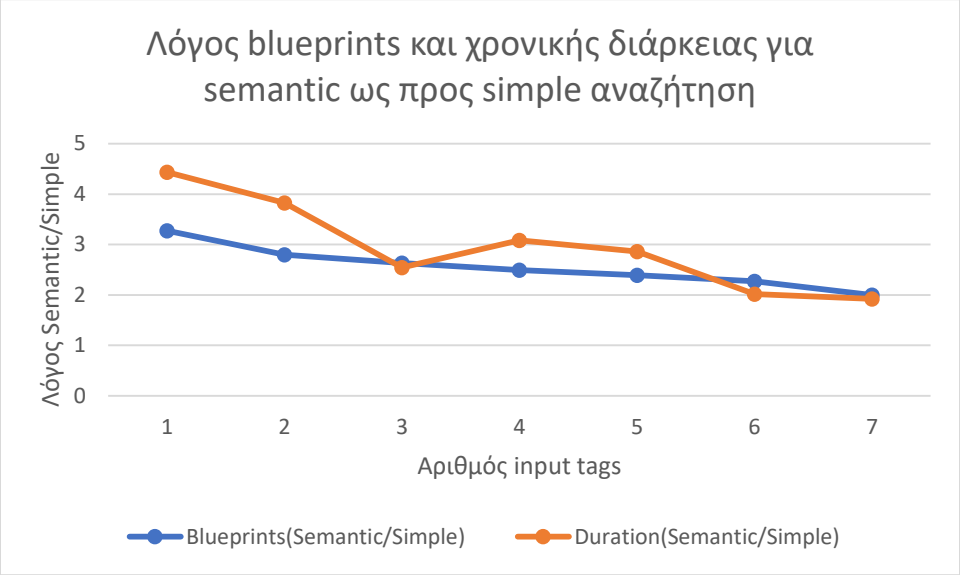
5.1.5. Σύνοψη ποσοτικού ελέγχου

Εν τέλει, οι διαδικασίες που αφορούν την επεξεργασία της οντολογίας επιφέρουν σχετικά μικρή καθυστέρηση, ενώ ο μεγαλύτερος όγκος από tags, που συνεπάγεται μεγαλύτερο όγκο blueprints, επιφέρει μια καθυστέρηση και στα πλαίσια της ανάκτησης, αλλά και στα πλαίσια της επεξεργασίας, όπου φαίνεται πως η διαδικασία που αυξάνει περισσότερο την καθυστέρηση της εκτέλεσης είναι η jsonize. Πέραν της όποιας βελτιστοποίησης μπορεί να επιτευχθεί στον τομέα της ταχύτητας, στόχος της semantic αναζήτησης είναι να προσφέρει μεγαλύτερο (εμπλουτισμένο) και πιο αντιπροσωπευτικό (βαθμολογημένο) σύνολο από blueprints.

Επίσης, εφόσον η πολυπλοκότητα της εκτέλεσης όλης της διαδικασίας δεν είναι μεγάλη, αλλά η καθυστέρηση φαίνεται να αυξάνεται αναλογικά ακόμα και για βάση δεδομένων μεγέθους 100.000 blueprints, η εκτέλεση δεν θα ξεπεράσει τα 10 δευτερόλεπτα πράγμα που την κάνει real-time εφαρμογή. Για να επιτευχθεί περαιτέρω optimization θα μπορούσε να περιοριστεί η διαδικασία της δημιουργίας του JSON-string (που είναι η μεγαλύτερη χρονικά) στο να επιστρέφει τελικά τα πρώτα 10, 20 ή ακόμα και 100 αποτελέσματα.

Τέλος, τα αποτελέσματα δείχνουν πως τα blueprints που επιστρέφονται είναι πολλαπλάσια και μάλιστα η αύξηση των αποτελεσμάτων είναι περίπου ίση της καθυστέρησης που προστίθεται, ιδίως όσο αυξάνεται ο αριθμός των tags που εισάγονται, δηλαδή, :

$$\frac{\text{Αριθμός\_blueprints\_με\_semantic\_search}}{\text{Αριθμός\_blueprints\_με\_simple\_search}} \approx \frac{\text{διάρκεια\_εκτέλεσης\_με\_semantic\_search}}{\text{διάρκεια\_εκτέλεσης\_με\_simple \_search}}$$



Εικόνα 5.14 Λόγος μέσου πλήθου blueprints και μέσης διάρκειας semantic και simple search για 1000 request ανά αριθμό tags

Από τα 5 input tags και πάνω, η διάρκεια φαίνεται να είναι μεγαλύτερη. Αυτό οφείλεται κυρίως στο γεγονός πως ενώ η dynamicGrade αυξάνεται σε διάρκεια όσο αυξάνονται τα input tags, ταυτόχρονα η ποικιλομορφία της βάσης που χρησιμοποιήθηκε περιορίζεται, καθώς επιστρέφονται περισσότερα από τα μισά blueprints της βάσης.

Η διάρκεια εκτέλεσης της semantic search φαίνεται να είναι κατά μέσο όρο 2,96 φορές μεγαλύτερη της simple, ενώ τα blueprints που επιστρέφονται 2,55 φορές περισσότερα.

Αξίζει να σημειωθεί επίσης εδώ, πως όσο αυξάνεται ο αριθμός των εισαχθέντων tags, αυξάνονται και τα blueprints που επιστρέφονται σε βαθμό που στα 5-6 tags, επιστρέφονται τα μισά documents της βάσης. Αν η βάση είχε μεγαλύτερο πλήθος από documents, άρα και μεγαλύτερη ποικιλομορφία, ο λόγος $\text{Αριθμός\_blueprints\_με\_semantic\_search}/\text{Αριθμός\_blueprints\_με\_simple\_search}$ θα είχε μεγαλύτερη τιμή, καθώς θα αυξανόντουσαν περαιτέρω τα documents της semantic αναζήτησης.

5.2. Ποιοτικός έλεγχος

Η semantic αναζήτηση βελτιώνει την simple αναζήτηση σε 2 σημαντικούς τομείς. Εμπλουτίζει τα αποτελέσματα βάσει συγγένειας των δοσμένων tags, η οποία προκύπτει από μια οντολογία και τα βαθμολογεί επιστρέφοντας τα πιο σχετικά πρώτα. Ήδη είναι εμφανής η διαφορά των blueprints που επιστρέφονται στις 2 περιπτώσεις (Εικόνα 5.7 Μέσος αριθμός blueprints που επιστρέφονται από το query για 1000 request ανά αριθμό tags)

Δύο μεγέθη για τη μέτρηση της ποιότητας των αποτελεσμάτων μιας αναζήτησης είναι η ***ακρίβεια*** και η ***ακρίβεια_ανάκλησης,*** όπως ορίζονται παρακάτω:

$$ακρίβεια = \frac{Αριθμός\_ανακτώμενων\_σχετικών\_documents}{Αριθμός\_ανακτώμενων\_documents}$$

$$ακρίβεια\_ανάκλησης = \frac{Αριθμός\_ανακτώμενων\_σχετικών\_documents}{Αριθμός\_σχετικών\_documents}$$

Ασφαλώς και τα δύο μεγέθη ιδανικά θα πρέπει να είναι ίσα με το 1. Με την ***ακρίβεια*** φαίνεται τι ποσοστό των ανακτώμενων documents είναι σχετικά με την ουσία της αναζήτησης, ενώ με την ***ακρίβεια_ανάκλησης*** πόσα από όλα τα σχετικά documents της βάσης ανακτώνται. Ο παράγοντας που θα καθορίσει το μέγεθος της *ακρίβειας_ανάκλησης* είναι το πόσο καλή και πλήρης είναι η υλοποιημένη οντολογία, ώστε για κάθε tag να εμφανίζονται όλα τα σχετικά σε αυτό. Η ***ακρίβεια*** αυξάνεται μέσω της βαθμολόγησης. Όσο καλύτεροι είναι οι κανόνες της βαθμολόγησης και καταλληλότερα επιλεγμένες οι μετρικές, τόσο αυξάνεται και η *ακρίβεια*.

Ακολουθούν κάποια παραδείγματα εκτέλεσης που επιδεικνύουν τη διαφορά μεταξύ του simple search και του semantic search.

5.2.1. Πλήθος αποτελεσμάτων

Στον Πίνακα 11-1 φαίνεται πως για το tag *Endocrinologist*, επιστρέφεται μόνο δύο αποτέλεσματα με το simple search, ενώ πολύ περισσότερα (στον πίνακα έχουν τοποθετηθεί τα πρώτα 10) με το semantic search.

Αντίστοιχα στον Πίνακα 11-2 φαίνεται πως για τα tag *Endocrinologist* και *fribrinogen (σ.σ.μια γλυκοπρωτεΐνη που δημιουργείται στο συκώτι)* επιστρέφονται μόνο τρία tags με το simple search, ενώ πάλι το semantic search επιστρέφει μεγάλο αριθμό (πάλι στον πίνακα εμφανίζονται τα πρώτα 10 εκ των 634).

Πίνακας 5-1 : Παράδειγμα-1 όπου semantic αναζήτηση επιστρέφει πολυπληθέστερα αποτελέσματα από την simple

Tags						
Endocrinologist						
Αποτελέσματα						
Simple				**Semantic**		
[Endocrinologist, Pregnancy, Female, BloodTest] - 50.0 [Chemotherapy, Endocrinologist, chemotherapyReason, Neurologist, OtherTest] - 44.72136				[Endocrinologist, Pregnancy, Female, BloodTest] - 50.0 [Chemotherapy, Endocrinologist, chemotherapyReason, Neurologist, OtherTest] - 44.72136 [Specialist, SmokingStatus, Female] - 35.218365 [Doctor, Examination, Intervention] - 35.218365 [Doctor, TestName, Outcome] - 35.218365		

	[Radiation, Specialist, Employed] - 35.218365 [Ethnicity, Doctor, OtherTest] - 35.218365 [radiationReason, Specialist, Induced_abortion] - 35.218365 [Education, haemoglobin, Doctor] - 35.218365 [Specialist, TestName, Diagnosis] - 35.218365 …. …. ….

Πίνακας 5-2 : Παράδειγμα-2 όπου semantic αναζήτηση επιστρέφει πολυπληθέστερα αποτελέσματα από την simple

Tags						
fribrinogen	Endocrinologist					
Αποτελέσματα						
Simple				Semantic		
[BodyMassIndex, Person, fribrinogen] - 57.735027 [Endocrinologist, Pregnancy, Female, BloodTest] - 50.0 [Chemotherapy, Endocrinologist, chemotherapyReason, Neurologist, OtherTest] - 44.72136				[BodyMassIndex, Person, fribrinogen] - 92.95339 [Doctor, Examination, Intervention] - 70.43673 [TestName, surgeryReason, Person] - 70.43673 [Specialist, TestName, Diagnosis] - 70.43673 [Doctor, TestName, Outcome] - 70.43673 [Examination, Employed, Person] - 70.43673 [Doctor, TestName, Outcome, OutcomeAssessment] - 61.0 [TestName, Gynecologist, Biopsy, Specialist] - 61.0 [Examination, OtherSpecialist, OtherTest, Doctor] - 61.0 [radiationReason, HealthcareVisit, TestName, Doctor] - 61.0 …. ….		

Είναι προφανές από τα 2 παραδείγματα πως η simple αναζήτηση δεν είναι αρκετή, καθώς επιστρέφει ελάχιστα αποτελέσματα και αυτά μη-αντιπροσωπευτικά της αναζήτησης. Επίσης, και τα 2 μεγέθη μέτρησης του ποιοτικού ελέγχου έχουν υψηλότερη τιμή στην περίπτωση του semantic search. Μάλιστα, ενώ ο πραγματικός αριθμός των αποτελεσμάτων που επιστρέφει η semantic search είναι 409 και 634 αντίστοιχα, στον τύπο εισάχθηκε ο αριθμός 10, αφού τα αποτελέσματα αυτά ταξινομούνται βάσει του βαθμού τους, άρα τα πρώτα 10 είναι και πιθανώς πιο κοντά σε αυτό που ζητούσε ο χρήστης. Επίσης, πέρα των σχετικών αυτών μεγεθών (ακρίβεια και ακρίβεια_ανάκλησης) μεγάλη είναι η διαφορά και στο πλήθος των αποτελεσμάτων, 2 στην simple έναντι 409 στην semantic και 3 στην simple έναντι 634 στην semantic.

Παράδειγμα1

$$ακρίβεια\_simple = \frac{2}{2} \qquad ακρίβεια\_ανάκλησης\_simple = \frac{2}{relev\_docs}$$

$$ακρίβεια\_semantic = \frac{10}{10} \qquad ακρίβεια\_ανάκλησης\_semantic = \frac{10}{relev\_docs}$$

relev_docs≥10

Παράδειγμα2

$$ακρίβεια\_simple = \frac{1}{3} \qquad ακρίβεια\_ανάκλησης\_simple = \frac{1}{relev\_docs}$$

$$ακρίβεια\_semantic = \frac{10}{10} \qquad ακρίβεια\_ανάκλησης\_semantic = \frac{7}{relev\_docs}$$

relev_docs≥7

5.2.2. Ποιότητα αποτελεσμάτων

Όπως προαναφέρθηκε όμως, ακόμα και αν επιστραφούν αρκετά αποτελέσματα, δεν σημαίνει πως είναι αντιπροσωπευτικά αυτού που αναζητούσε ο χρήστης.

Για παράδειγμα, στον πίνακα 11-3 ενώ ο χρήστης αναζητά blueprints σχετικά με επεμβάσεις σε έγκυους γυναίκες, καθώς και την ειδικότητα του γιατρού που την εκτέλεσε, παίρνει τυχαία αποτελέσματα, όπως [Specialist, SmokingStatus, Female] - 115.470055 και [haemoglobin, Female, Specialist] - 115.470055. Αντιθέτως με τη semantic search επιστρέφονται πολύ πιο αντιπροσωπευτικά αποτελέσματα. Αξίζει να σημειωθεί, πως ακόμα και το 19ο στοιχείο του semantic search στο παράδειγμα έχει υψηλότερο βαθμό από το 1ο στοιχείο του simple search. Επίσης, η simple search επέστρεψε 487 έναντι 1994 αποτελεσμάτων της semantic.

Αντίστοιχα στον πίνακα 11-4, ενώ ο χρήστης αναζητά blueprints που περιέχουν δημογραφικά χαρακτηριστικά ασθενών ανάλογα με το αν έχουν υπάρξει καπνιστές, που επισκέπτονται οικογενειακούς γιατρούς για τη διενέργεια κάποιας εξέτασης, παίρνει τυχαία αποτελέσματα όπως [Demographics, Psychiatrist, TestType] - 115.470055 [Nephrologist, TestType, TestName, HealthcareVisit] - 100.0. Αντιθέτως, με το semantic search επιστρέφονται πολύ πιο αντιπροσωπευτικά αποτελέσματα. Αξίζει να σημειωθεί, πως και σε αυτήν την περίπτωση μέχρι και το 16ο στοιχείο του semantic search, έχει υψηλότερο βαθμό από το 1ο στοιχείο του simple search. Επίσης, η simple search επέστρεψε 591 έναντι 1860 αποτελεσμάτων της semantic.

Πίνακας 5-3 : Παράδειγμα-1 όπου semantic αναζήτηση επιστρέφει ακριβέστερα αποτελέσματα από την simple

Tags						
Pregnancy	Surgery	Specialist	Female			

Αποτελέσματα	
Simple	Semantic
[Specialist, SmokingStatus, Female] - 115.470055 [haemoglobin, Female, Specialist] - 115.470055 [Endocrinologist, Pregnancy, Female, BloodTest] - 100.0 [Gynecologist, Pregnancy, Female, BloodTest] - 100.0 [Person, Gynecologist, Surgery, Pregnancy] - 100.0 [HealthcareVisit, Specialist, OutcomeAssessment, Pregnancy] - 100.0 [Female, Specialist, Occupation, plateletCount] - 100.0 [Female, Other, Gender, Pregnancy] - 100.0 [Unemployed, Surgery, Female, Spontaneous_abortion_/_Miscarriage, Single] - 89.44272 [Specialist, Dentist, OutcomeValue, Biopsy, Pregnancy] - 89.44272	[Person, Gynecologist, Surgery, Pregnancy] - 170.5 [Person, Gynecologist, surgeryReason, PregnancyOutcome] - 150.5 [Gynecologist, Pregnancy, Female, BloodTest] - 140.0 [Endocrinologist, Pregnancy, Female, BloodTest] - 140.0 [OtherSpecialist, surgeryReason, PregnancyOutcome] - 138.56407 [Nephrologist, Surgery, Gynecologist, PregnancyOutcome] - 130.0 [Female, PregnancyOutcome, Dermatologist, Twins] - 130.0 [Patient, Gynecologist, Surgery, PregnancyOutcome] - 130.0 [Demographics, PregnancyOutcome, Neurologist] - 127.594406 [interventionReason, Surgery, Female, MedicalImagingTest, Neurologist] - 125.21981 ....

Πίνακας 5-4 : Παράδειγμα-2 όπου semantic αναζήτηση επιστρέφει ακριβέστερα αποτελέσματα από την simple

Tags						
HealthcareVisit	FamilyDoctor	Demographics	TestType	SmokingStatus		

Αποτελέσματα	
Simple	Semantic
[Demographics, Psychiatrist, TestType] - 115.470055 [Diagnosis, TestType, interventionReason, SmokingStatus] - 100.0 [OcularTest, TestType, Demographics, Black_or_African_American] - 100.0 [Nephrologist, TestType, TestName, HealthcareVisit] - 100.0 [SmokingStatus, Caucasian_or_White, EducationLevel, FamilyDoctor] - 100.0 [Asian, FamilyDoctor, Retired, Demographics] - 100.0 [TestType, FamilyDoctor, antiThrombin, Retired] - 100.0	[HealthcareVisit, BodyMassIndex, BloodTest, Smoker] - 165.5 [Never_Smoker, BodyMassIndex, HealthcareVisit, BloodTest] - 165.5 [HealthcareVisit, BodyMassIndex, BloodTest, Ex-Smoker] - 165.5 [HealthcareVisit, Doctor, Ethnicity, OtherTest] - 160.5 [OutcomeValue, BodyMassIndex, Ex-Smoker, FamilyDoctor, MedicalImagingTest] - 148.0277

[American_Indian_or_Alaska_Native, Region, TestType, FamilyDoctor] - 100.0 [Patient, HealthcareVisit, TestType, Ophthalmologist] - 100.0 [Demographics, cholesterol, Outcome, FamilyDoctor] - 100.0	[FamilyDoctor, Examination, Smoker, Ethnicity, Interventions] - 139.53064 [Education, TestType, Doctor] - 139.14142 [Examination, Occupation, Person, HealthcareVisit, Region] - 135.0585 [BloodTest, Ex-Smoker, BodyMassIndex] - 133.3679 [OutcomeAssessment, UrineTest, HealthcareVisit, Education] - 130.0 ....

5.2.3. Σύνοψη ποιοτικού ελέγχου

Η διαφορά της σημασιολογικής από την απλή αναζήτηση είναι εμφανής. Ο χρήστης «πληρώνει» μεγαλύτερο πλήθος και ποιότητα αποτελεσμάτων με ταχύτητα εκτέλεσης. Μάλιστα, όπως φαίνεται και στην Εικόνα 5.14 Λόγος μέσου πλήθου blueprints και μέσης διάρκειας semantic και simple search για 1000 request ανά αριθμό tags), η διαφορά στην ταχύτητα εκτέλεσης είναι ανάλογη της διαφοράς του πλήθους των αποτελεσμάτων και αν συμψηφιστεί και η ποιότητα των αποτελεσμάτων (πράγμα δύσκολα μετρήσιμο) ο χρόνος που «πληρώνει» αντιστοιχεί σε μια πολύ καλύτερη αναζήτηση. Βέβαια, αν γινόταν η αντίστοιχη σύγκριση της semantic αναζήτησης με κάποια άλλη αναζήτηση, όπως η elastic, θα φαινόταν μικρότερη διαφορά ως προς τον χρόνο εκτέλεσης, αλλά και πάλι διαφορά στην ποιότητα των αποτελεσμάτων (αυτή την φορά τα η κάθε αναζήτηση θα είχε διαφορετικά αποτελέσματα ανάλογα με το είδος του προβλήματος).

6. ΣΥΜΠΕΡΑΣΜΑΤΑ

Τα οφέλη που προκύπτουν από τη χρήση μιας semantic αναζήτησης, όπως εξετάστηκαν στα αποτελέσματα και συγκεκριμένα στο 5.2. (Ποιοτικός έλεγχος), είναι η δυνατότητα για επιστροφή μεγαλύτερου πλήθους αποτελεσμάτων και η ποιοτικότερη βαθμολόγησή τους.

Ένα ακόμα όφελος είναι η δυνατότητα κατακερματισμού της διαδικασίας υλοποίησης. Η εφαρμογή είναι ontology-agnostic, δηλαδή, υλοποιεί τη σημασιολογική αναζήτηση (και την απλή) χωρίς να επηρεάζεται από το είδος των δεδομένων που επιστρέφει (παρά μόνο το format τους). Έτσι, για την ποιότητα των αποτελεσμάτων υπεύθυνος είναι ο ontology expert, καθώς ο ακριβής σχεδιασμός της οντολογίας θα επιφέρει αντίστοιχα ακριβή αποτελέσματα.

Επίσης, αν η σημασιολογική αναζήτηση δεν συγκριθεί με μια απλή, αλλά (για παράδειγμα) με την ελαστική (Elastic Search), παρουσιάζει και κάποια ακόμα θετικά στοιχεία. Η Elastic Search υπολογίζει για κάθε αποτέλεσμα έναν βαθμό με έναν αλγόριθμο (αντίστοιχα η semantic : Πίνακας 3-9 : Αλγόριθμος υπολογισμού βαθμού blueprint), ο οποίος περιέχει στοιχεία, όπως inverse document frequency (idf : αν ένα tag εμφανίζεται σε πολλά documents αξιοποιείται λιγότερο) ή και term

frequency (tf : πόσες φορές εμφανίζεται ένα tag σε ένα document), τα οποία δεν χρειάζονται στην προκείμενη υλοποίηση.

Παρ'όλα αυτά, ένας ακόμα προτεινόμενος σχεδιασμός είναι η υλοποίηση της σημασιολογικής αναζήτησης πάνω στην ελαστική αναζήτηση. Μια υλοποίηση δηλαδή, η οποία θα εμπλουτίζει τα εισαχθέντα ορίσματα βασισμένη σε μια οντολογία και θα πραγματοποιεί την αναζήτηση στη βάση με τη χρήση της ελαστικής αναζήτησης.

Συνοψίζοντας η σημασιολογική αναζήτηση προσφέρει:

- **Μεγαλύτερο όγκο αποτελεσμάτων**: Ο εμπλουτισμός των εισαχθέντων στοιχείων με άλλα παρεμφερή προσφέρει περισσότερα αποτελέσματα.
- **Ποιοτικότερη ταξινόμηση αποτελεσμάτων**: Η διαδικασία βαθμολόγησης ταξινομεί βάσει σχετικότητας τα αποτελέσματα.
- **Ontology-agnostic**: Η υλοποιημένη εφαρμογή δεν είναι σχεδιασμένη πάνω σε μια συγκεκριμένη οντολογία ή τύπο δεδομένων.
- **Αξιοποίηση άλλων αναζητήσεων**: Η σημασιολογική αναζήτηση μπορεί να βασίζει την αναζήτησή της στη βάση δεδομένων πάνω σε άλλες, όπως η elastic search.

6.1. Μελλοντικές επεκτάσεις

Μια υλοποίηση που εφαρμόζει τη σημασιολογική αναζήτηση έχει διάφορους τομείς στους οποίους μπορεί να βελτιωθεί στηριζόμενη πάνω στην παρούσα.

Αρχικά, μπορεί η (hard-coded) διαδικασία εμπλουτισμού να γίνει παραμετροποιήσιμη. Να επιλέγεται δηλαδή, στο configuration ποιες συγγένειες θα αξιοποιούνται και με τι βαθμό. Επίσης, μπορεί να ορίζεται ο αριθμός των αποτελεσμάτων που θα εμφανίζονται τελικά.

Πιο ουσιώδης επέκταση θα ήταν η συνδυαστική/υβριδική αξιοποίηση της σημασιολογικής με την ελαστική αναζήτηση, όπου η σημασιολογική αναζήτηση εμπλουτίζει τα tags και η ελαστική καλείται να εκτελέσει την αναζήτηση στη βάση δεδομένων. Ουσιώδης επέκταση θα ήταν και η περαιτέρω δυνατότητα παραμετροποίησης των queries που πραγματοποιούνται στη βάση δεδομένων.

Επίσης, η δυνατότητα προσφοράς στον χρήστη προτάσεων για αναζήτηση (Για παράδειγμα για είσοδο "Docttor" → "Did you mean : Doctor?"), η δυνατότητα για feedback από τον χρήστη και

ανάλυση πάνω στα επιστρεφόμενα αποτελέσματα αλλά και στην ίδια την οντολογία θα μπορούσε να επιφέρει εκπληκτικά αποτελέσματα.

7. ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Flanagan, D., & Like, W. S. (2006). JavaScript: The Definitive Guide, 5th.

[2] Rauschmayer, A. (2014). Speaking JavaScript: an in-depth guide for programmers. " O'Reilly Media, Inc.".

[3] Lindholm, T., Yellin, F., Bracha, G., & Buckley, A. (2014). The Java virtual machine specification. Pearson Education.

[4] JDK 11 Release Notes, Important Changes, and Information. Διαθέσιμο online: https://www.oracle.com/java/technologies/javase/jdk-11-relnote.html. [Πρόσβαση 26 Σεπτεμβρίου 2020].

[5] JavaBeans Spec. Διαθέσιμο online: https://www.oracle.com/java/technologies/javase/javabeans-spec.html [Πρόσβαση 26 Σεπτεμβρίου 2020].

[6] The Java™ Tutorials. Διαθέσιμο online : https://docs.oracle.com/javase/tutorial/java/annotations/. [Πρόσβαση 09 2020].

[7] HTML 4.01 Specification, W3C, Μάρτιος 2018. Διαθέσιμο online : https://www.w3.org/TR/html401/intro/intro.html. [Πρόσβαση 09 2020].

[8] RFC 2616, ietf, . Διαθέσιμο online: https://www.ietf.org/rfc/rfc2616.txt. [Πρόσβαση 09 2020].

[9] OWL Web Ontology Language, W3C, 12 09 2009. Διαθέσιμο online: https://www.w3.org/TR/2004/REC-owl-features-20040210/. [Πρόσβαση 09 2020].

[10] OWL 2 Web Ontology Language, W3C, 12 2012. Διαθέσιμο online: https://www.w3.org/TR/owl2-primer/. [Πρόσβαση 09 2020].

[11] Pellet, W3C, 01 2011. Διαθέσιμο online: https://www.w3.org/2001/sw/wiki/Pellet. [Πρόσβαση 09 2020].

[12] Fielding, R. T., & Taylor, R. N. (2000). *Architectural styles and the design of network-based software architectures* (Vol. 7). Irvine: University of California, Irvine.

[13] The JavaScript Object Notation (JSON) Data Interchange Format, Διαθέσιμοonline: https://tools.ietf.org/id/draft-ietf-jsonbis-rfc7159bis-04.html#RFC4627. [Πρόσβαση 09 2020].

[14] R. du Rhône, The JSON Data Interchange Syntax, 2nd Edition ed., Ecma International, 2017.

[15] Chodorow, K. (2013). *MongoDB: the definitive guide: powerful and scalable data storage*. " O'Reilly Media, Inc.".

[16] Johnson, R., Hoeller, J., Donald, K., Sampaleanu, C., Harrop, R., Risberg, T., ... & Templier, T. (2004). The spring framework-reference documentation. *interface*, *21*, 27.

[17] Walls, C. (2016). Spring Boot in action. Manning Publications.

[18] Spring Boot, Διαθέσιμο online: https://spring.io/projects/spring-boot. [Πρόσβαση 09 2020].

[19] Microservices, Srping, Διαθέσιμο online: https://spring.io/microservices. [Πρόσβαση 09 2020].

[20] Jena Ontology API, Διαθέσιμο online: https://jena.apache.org/documentation/ontology/. [Πρόσβαση 09 2020].

[21] Apache JMeter™, APACHE, Διαθέσιμο online: https://jmeter.apache.org/. [Πρόσβαση 09 2020].

[22] Protégé, Stanford University, Διαθέσιμο online: https://protege.stanford.edu/. [Πρόσβαση 09 2020].

[23] Semantic Scholar, Διαθέσιμο online: https://www.semanticscholar.org/topic/Data-as-a-service/396764?fbclid=IwAR1l6PCB3KXaUenMaT69wYOr0NMIJrbbB0w2Wm_BpozCO_iRNibFXr0WyaU. [Πρόσβαση 16 09 2020].

[24] MongoDb, Διαθέσιμο online: https://www.mongodb.com/initiatives/data-as-a-service?fbclid=IwAR20KvdfcPQVZKh3wl-HmbGqDtXrF85JAxfFscCmTdkxaTG11f3SgULfUzY. [Πρόσβαση 16 09 2020].

[25] Psomakelis, E., Nikolakopoulos, A., Marinakis, A., Psychas, A., Moulos, V., Varvarigou, T., & Christou, A. (2020). A Scalable and Semantic Data as a Service Marketplace for Enhancing Cloud-Based Applications. Future Internet, 12(5), 77.

[26] Start Bootstrap, Διαθέσιμο online: https://startbootstrap.com/themes/grayscale. [Πρόσβαση 17 09 2020].

[27] docs.spring.io, Spring, Διαθέσιμο online: https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/autoconfigure/SpringBootApplication.html. [Πρόσβαση 18 09 2020].

[28] spring.io, Διαθέσιμο online: https://spring.io/guides/gs/spring-boot/. [Πρόσβαση 18 09 2020].

[29] docs.spring.io, Spring, Διαθέσιμο online: https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/autoconfigure/EnableAutoConfiguration.html. [Πρόσβαση 18 09 2020].

[30] Building an Application with Spring Boot, Διαθέσιμο online: https://spring.io/guides/gs/spring-boot/. [Πρόσβαση 18 09 2020].

[31] developer.mozilla.org, Διαθέσιμο online: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS. [Πρόσβαση 18 09 2020].

[32] Creating Asynchronous Methods, Spring, Διαθέσιμο online: https://spring.io/guides/gs/async-method/. [Πρόσβαση 18 09 2020].

[33] docs.oracle.com, Oracle, Διαθέσιμο online: https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ThreadPoolExecutor.html. [Πρόσβαση 18 09 2020].

[34] Web on Servlet Stack, Spring, Διαθέσιμο Online: https://docs.spring.io/spring/docs/5.2.8.RELEASE/spring-framework-reference/web.html#mvc. [Πρόσβαση 18 09 2020].

[35] Spring Boot Reference Guide, Spring, Διαθέσιμο online: https://docs.spring.io/spring-boot/docs/2.0.0.M3/reference/htmlsingle/. [Πρόσβαση 18 09 2020].

[36] Oaks, S. (2014). Java Performance: The Definitive Guide: Getting the Most Out of Your Code. " O'Reilly Media, Inc.".

[37] GitHub, Διαθέσιμο online: https://github.com/Galigator/openllet. [Πρόσβαση 18 09 2020].

[38] GitHub, Διαθέσιμο online: https://github.com/DITAS-Project/blueprint/tree/master/ehealth/abstract. [Πρόσβαση 18 09 2020].

[39] Mangold, C. (2007). A survey and classification of semantic search approaches. *International Journal of Metadata, Semantics and Ontologies*, *2*(1), 23-34.

[40] Parsia, B., & Sirin, E. (2004, November). Pellet: An owl dl reasoner. In *Third international semantic web conference-poster* (Vol. 18, p. 13). Publishing.

[41] Lindholm, T., Yellin, F., Bracha, G., & Buckley, A. (2013). *The Java Virtual Machine Specification, Java SE 7 Edition: Java Virt Mach Spec Java_3*. Addison-Wesley.

8. ΚΩΔΙΚΑΣ

8.1. Front-End

8.1.1. Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script type="text/javascript" src="startbootstrap-grayscale-gh-pages/js/sandbox.js"></script>
    <link type="text/css" rel="stylesheet" href="startbootstrap-grayscale-gh-
pages/bootstrap.min.css">

</head>
<body>
    <style>
        .wrapper {
    width: 100%;
    margin-left : 5px;
    margin-right : 5px;
    overflow: hidden;  /*add this to contain floated children */
}
.first {
    width: 20%;
    float:left; /* add this */
}
.second {
    width: 39%;
    float: left; /* add this */
    margin-left : 5px;
    font-size: 0.8rem;

}
.third {
    width: 39%;
    float: left; /* add this */
    margin-left : 5px;
    font-size: 0.8rem;
}
.btn_add {
        background: url(startbootstrap-grayscale-gh-pages/assets/img/add.png) no-repeat;
        background-size : 100%;
        float: left;
}
.btn_rem {
        background: url(startbootstrap-grayscale-gh-pages/assets/img/remove.png) no-repeat;
        background-size : 100%;
```

```
        float: left;
        margin-left : 5px
}
    </style>
    <title>Eeeela</title>
    <h1>Welcome!</h1>
<div class="wrapper">
    <p>Here, we can compare the difference between running a simple search and an semantic
search.<br>
        Simple search   : Given some tags, blueprints containing the exact tags will be returned.
<br>
        Semantic search : Given some tags, blueprints containing the exact tags, as well as their
superclasses, subclasses, siblings or individuals will be returned.
    </p>


<div class="first">
    <form id="frm1"">
        Tag 1: <input type="text" id="tag1"  style="width:100%"><br>

      </form>
    <button style="margin-left:30%; width:30px; height:30px; border-style:none;"
onclick="addInputTag()" class="btn_add"></button>
    <button style="margin-left:10%; width:30px; height:30px; border-style:none;"
onclick="deleteInputTag()" class="btn_rem"></button><br><br><br>

    <button style="display:inline-block" onclick="simpleSearch()">Simple Search</button>
    <button style="display:inline-block" onclick="semanticSearch()">Semantic Search</button><br><br>



</div>
<div class="second" id="simple_div">
    <h4 style="text-align: center">Simple Search Results</h4>
</div>

<div class="third" id="advanced_div"">
    <h4  style="text-align: center">Semantic Search Results</h4>
</div>


</div>
</body>
</html>
```

8.1.2. Sandbox.js

```
console.log('Inside Javascript');

var tagcnt=1;

const semanticSearch = async (tag1) => {
        console.log("SemanticSearch[Start]");

        if (!document.getElementsByTagName) return;

    var tag = [];
    var fetcher = "?";
    for (j=0; j<6; j++){
```

```
        if (document.getElementById('tag'+(j+1))==null)
                break;
        if (document.getElementById('tag'+(j+1)).value==""){
                window.alert("Fill all the tags or remove them!");
                return;
        }
        tag[j] = document.getElementById('tag'+(j+1)).value;
    }
    for (w=0; w<j; w++){
        if (w>0)
                fetcher += "&";
        fetcher += "tag"+(w+1)+"="+tag[w];
        }
    for (w=j; w<7; w++)
        fetcher += "&tag"+(w+1)+"=";

    console.log('Fetching advanced search : /semantic'+fetcher);
    const response = await fetch('http://localhost:8090/semantic'+fetcher);
    const advancedJson = await response.json(); //extract JSON from the http response
    // do something with myJson

    console.log('Putting all the values in the table');
    addTable(advancedJson, "advanced_div");

        console.log("SemanticSearch[End]");
}

const simpleSearch = async () => {
        console.log("SimpleSearch[Start]");
    if (!document.getElementsByTagName) return;

    var tag = [];
    var fetcher = "?";
    for (j=0; j<6; j++){
        if (document.getElementById('tag'+(j+1))==null)
                break;
        if (document.getElementById('tag'+(j+1)).value==""){
                window.alert("Fill all the tags or remove them!");
                return;
        }
        tag[j] = document.getElementById('tag'+(j+1)).value;
    }
    for (w=0; w<j; w++){
        if (w>0)
                fetcher += "&";
        fetcher += "tag"+(w+1)+"="+tag[w];
        }
    for (w=j; w<7; w++)
        fetcher += "&tag"+(w+1)+"=";

    console.log('Fetching simple search : /simple'+fetcher);
    const response = await fetch('http://localhost:8090/simple'+fetcher);
    const simpleJson = await response.json(); //extract JSON from the http response
    // do something with myJson

    console.log('Putting all the values in the table');
    addTable(simpleJson, "simple_div");
    console.log("SimpleSearch[End]");
}

function addTable(thejson, table_id) {
        console.log("AddTable[Start]");
```

```
    var myTableDiv = document.getElementById(table_id)
    var myobj = document.getElementById(table_id+"1");
    if (myobj)
        myobj.remove();
    var table = document.createElement('TABLE')
    table.width = '95%';
    var tableBody = document.createElement('TBODY')

    table.border = '1';
    table.id = table_id+'1';
    table.appendChild(tableBody);

    var heading = new Array();
    heading[0] = "Blueprint"
    heading[1] = "Values"


    //TABLE COLUMNS
    var tr = document.createElement('TR');
    tableBody.appendChild(tr);
    for (i = 0; i < heading.length; i++) {
        var th = document.createElement('TH')
        th.appendChild(document.createTextNode(heading[i]));
        tr.appendChild(th);

    }

    //TABLE ROWS
    for (var i in thejson.blueprint) {
        var tr = document.createElement('TR');

        var td = document.createElement('TD')
        td.appendChild(document.createTextNode("Index: "+thejson.id[i]));
        tr.appendChild(td);

        var td = document.createElement('TD')
        td.appendChild(document.createTextNode(thejson.blueprint[i]));
        tr.appendChild(td);

        tableBody.appendChild(tr);
    }

    myTableDiv.appendChild(table);

    console.log("AddTable[End]");

}

function addInputTag(){
        console.log("AddInputTag[Start] with number of tags = "+tagcnt);
        var container = document.getElementById("frm1");

        if (tagcnt>=7)
                return;
        //while (container.hasChildNodes()){
                //container.removeChild(container.lastChild);
        //}
        tagcnt++;
        container.appendChild(document.createTextNode("Tag "+tagcnt));
        var input = document.createElement("input");
        input.type="text";
        input.id="tag"+tagcnt;
        input.style="width:100%";
```

```
        container.appendChild(input)

        container.appendChild(document.createElement("br"));

        console.log("AddInputTag[End] with number of tags = "+tagcnt);


}

function deleteInputTag(){
        console.log("DeleteInputTag[Start] with number of tags = "+tagcnt);
        var container = document.getElementById("frm1");
        if (tagcnt<2)
                return;
        tagcnt--;

        //br
        container.removeChild(container.lastChild);
        //input
        container.removeChild(container.lastChild);
        //text
        container.removeChild(container.lastChild);
        console.log("DeleteInputTag[End] with number of tags = "+tagcnt);
}
```

8.2. Ontology

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#"
     xml:base="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:obda="https://w3id.org/obda/vocabulary#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
    <owl:Ontology rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20"/>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->




    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasAddress -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#hasAddress">
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Demographics"/>
```

```
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Person"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Region"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasBodyMassIndex
-->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasBodyMassIndex">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Demographics"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Person"/>
        <rdfs:range>
            <owl:Class>
                <owl:oneOf rdf:parseType="Collection">
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#normalWeight"/>
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#obesity"/>
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#overWeight"/>
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#underWeight"/>
                </owl:oneOf>
            </owl:Class>
        </rdfs:range>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasDemographics -
->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasDemographics">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Person"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Demographics"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasDoctor -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasDoctor">
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Patient"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Person"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Dentist"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Dermatologist"/>
```

```
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Doctor"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Endocrinologist"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#FamilyDoctor"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Gastroenterologist"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Hematologist"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Nephrologist"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Neurologist"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Ophthalmologist"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OtherSpecialist"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Psychiatrist"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Pulmonologist"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#ENT_(Ear,_Nose,_and_Throat)"/>
        <rdfs:range>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Gynecologist"/>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasGender"/>
                        <owl:allValuesFrom rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Female"/>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </rdfs:range>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasEducation -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasEducation">
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Demographics"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Person"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Education"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasEducationLevel -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasEducationLevel">
```

```
        <rdfs:subPropertyOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#hasEducation"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Education"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Person"/>
        <rdfs:range>
            <owl:Class>
                <owl:oneOf rdf:parseType="Collection">
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#OtherEducationLevel"/>
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Primary"/>
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Secondary"/>
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Tertiary"/>
                </owl:oneOf>
            </owl:Class>
        </rdfs:range>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasEthnicity -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#hasEthnicity">
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Demographics"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Person"/>
        <rdfs:range>
            <owl:Class>
                <owl:oneOf rdf:parseType="Collection">
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#American_Indian_or_Alaska_Native"/>
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Asian"/>
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Black_or_African_American"/>
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Caucasian_or_White"/>
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Native_Hawaiian_or_Other_Pacific_Islander"/>
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Hispanic_/_Latino"/>
                </owl:oneOf>
            </owl:Class>
        </rdfs:range>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasGender -->
```

```
    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasGender">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AsymmetricProperty"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Demographics"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Person"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Female"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Male"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OtherGender"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasOccupationStatus -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasOccupationStatus">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Demographics"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Person"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Employed"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OtherOccupationStatus"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Retired"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Student"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Underage"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Unemployed"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasOutcome -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasOutcome">
        <rdfs:subPropertyOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#undertakeExamination"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Examination"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Outcome"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasOutcomeAssessment -->
```

```
    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasOutcomeAssessment">
        <rdfs:subPropertyOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#undertakeExamination"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Outcome"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OutcomeAssessment"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasOutcomeFormat -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasOutcomeFormat">
        <rdfs:subPropertyOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasOutcome"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Examination"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Outcome"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OutcomeFormat"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasOutcomeUnit -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasOutcomeUnit">
        <rdfs:subPropertyOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasOutcome"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Examination"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Outcome"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OutcomeUnit"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasOutcomeValue -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasOutcomeValue">
        <rdfs:subPropertyOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasOutcome"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Examination"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Outcome"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OutcomeValue"/>
```

```
    </owl:ObjectProperty>


    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasPersonStatus -
->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#hasPersonStatus">
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Person"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Doctor"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Nurse"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Other"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Patient"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasPregnancy -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#hasPregnancy">
        <rdfs:domain>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Person"/>
                    <owl:Restriction>
                        <owl:onProperty
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasGender"/>
                        <owl:allValuesFrom
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Female"/>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </rdfs:domain>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Pregnancy"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#hasPregnancyOutcome -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#hasPregnancyOutcome">
        <rdfs:subPropertyOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#hasPregnancy"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Pregnancy"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#PregnancyOutcome"/>
    </owl:ObjectProperty>
```

```
    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasSmokingStatus
-->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#hasSmokingStatus">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Person"/>
        <rdfs:range>
            <owl:Class>
                <owl:oneOf rdf:parseType="Collection">
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Ex-Smoker"/>
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Never_Smoker"/>
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Smoker"/>
                </owl:oneOf>
            </owl:Class>
        </rdfs:range>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasSpecialty -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#hasSpecialty">
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Doctor"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Specialist"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasTestType -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#hasTestType">
        <rdfs:subPropertyOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#undertakeExamination"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Examination"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Biopsy"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#BloodTest"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#MedicalImagingTest"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#OcularTest"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#OralTest"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#OtherTest"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#UrineTest"/>
    </owl:ObjectProperty>
```

```
    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#healthcareVisitAt
-->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#healthcareVisitAt">
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#HealthcareVisit"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Specialist"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#subRegionOf -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#subRegionOf">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Region"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Region"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#undertakeExamination -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#undertakeExamination">
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Patient"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Person"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Examination"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#undertakeIntervention -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#undertakeIntervention">
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Patient"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Person"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#preg:hasDoctor --
>

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#preg:hasDoctor">
        <rdfs:subPropertyOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasDoctor"/>
        <rdfs:domain>
```

```
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Patient"/>
                    <owl:Restriction>
                        <owl:onProperty
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasGender"/>
                        <owl:allValuesFrom
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Female"/>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </rdfs:domain>
        <rdfs:range rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Gynecologist"/>
    </owl:ObjectProperty>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->




    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Abortion -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Abortion">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#PregnancyOutcome"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Biopsy -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Biopsy">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#TestType"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#BloodTest -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#BloodTest">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#TestType"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#BodyMassIndex -->
```

```
    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#BodyMassIndex">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Demographics"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Chemotherapy -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Chemotherapy">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Interventions"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Demographics -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Demographics"/>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Dentist -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Dentist">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Specialist"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Dermatologist -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Dermatologist">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Specialist"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Diagnosis -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Diagnosis"/>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Doctor -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Doctor">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Person"/>
```

```
        <owl:disjointWith
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Nurse"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Education -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Education">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Demographics"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#EducationLevel --
>

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#EducationLevel">
        <owl:equivalentClass>
            <owl:Class>
                <owl:oneOf rdf:parseType="Collection">
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#OtherEducationLevel"/>
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Primary"/>
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Secondary"/>
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Tertiary"/>
                </owl:oneOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Education"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Employed -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Employed">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Occupation"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Endocrinologist -
->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Endocrinologist">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Specialist"/>
    </owl:Class>
```

```
    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Ethnicity -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Ethnicity">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Demographics"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Examination -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Examination"/>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#FamilyDoctor -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#FamilyDoctor">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Doctor"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Female -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Female">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Gender"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Gastroenterologist -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Gastroenterologist">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Specialist"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Gender -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Gender">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Demographics"/>
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Person"/>
    </owl:Class>
```

```
    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Gynecologist -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Gynecologist">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Specialist"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#HealthcareVisit -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#HealthcareVisit"/>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Hematologist -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Hematologist">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Specialist"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Induced_abortion -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Induced_abortion">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Abortion"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Interventions -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Interventions"/>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Male -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Male">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Gender"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#MedicalImagingTest -->
```

```
    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#MedicalImagingTest">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#TestType"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Medication -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Medication">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Interventions"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Nephrologist -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Nephrologist">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Specialist"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Neurologist -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Neurologist">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Specialist"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Nurse -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Nurse">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Person"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Occupation -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Occupation">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Demographics"/>
    </owl:Class>
```

```
    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OcularTest -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OcularTest">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#TestType"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Ophthalmologist -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Ophthalmologist">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Specialist"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OralTest -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OralTest">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#TestType"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Other -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Other">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Person"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OtherGender -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OtherGender">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Gender"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OtherOccupationStatus -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OtherOccupationStatus">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Occupation"/>
    </owl:Class>
```

```
    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#OtherPregnancyOutcome -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#OtherPregnancyOutcome">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#PregnancyOutcome"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OtherSpecialist -
->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#OtherSpecialist">
        <owl:equivalentClass>
            <owl:Restriction>
                <owl:onProperty
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#hasSpecialty"/>
                <owl:qualifiedCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:qualifiedCardinality>
                <owl:onClass
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Specialist"/>
            </owl:Restriction>
        </owl:equivalentClass>
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Specialist"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OtherTest -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#OtherTest">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#TestType"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Outcome -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Outcome">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Examination"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OutcomeAssessment
-->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#OutcomeAssessment">
```

```
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Examination"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OutcomeFormat -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#OutcomeFormat">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Outcome"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OutcomeUnit -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#OutcomeUnit">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Outcome"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OutcomeValue -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#OutcomeValue">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Outcome"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Patient -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Patient">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Person"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Person -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Person">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Person"/>
                    <owl:Restriction>
                        <owl:onProperty
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#hasDemographics"/>
                        <owl:allValuesFrom
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Demographics"/>
```

```
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#hasGender"/>
                        <owl:allValuesFrom>
                            <owl:Class>
                                <owl:unionOf rdf:parseType="Collection">
                                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Female"/>
                                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Male"/>
                                    <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OtherGender"/>
                                </owl:unionOf>
                            </owl:Class>
                        </owl:allValuesFrom>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Pregnancy -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Pregnancy"/>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#PregnancyOutcome
-->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#PregnancyOutcome">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Pregnancy"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Psychiatrist -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Psychiatrist">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Specialist"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Pulmonologist -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Pulmonologist">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Specialist"/>
    </owl:Class>
```

```
    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Radiation -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Radiation">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Interventions"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Region -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Region"/>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Retired -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Retired">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Occupation"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Single -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Single">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#PregnancyOutcome"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#SmokingStatus -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#SmokingStatus"/>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Specialist -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Specialist">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Doctor"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Stillborn -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Stillborn">
```

```
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#PregnancyOutcome"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Student -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Student">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Occupation"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Surgery -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Surgery">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Interventions"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#TestName -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#TestName">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Examination"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#TestType -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#TestType">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Examination"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Twins -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Twins">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#PregnancyOutcome"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Underage -->
```

```
    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Underage">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Occupation"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Unemployed -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Unemployed">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Occupation"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#UrineTest -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#UrineTest">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#TestType"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#antiThrombin -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#antiThrombin">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#TestName"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#chemotherapyReason -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#chemotherapyReason">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Chemotherapy"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#cholesterol -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#cholesterol">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#TestName"/>
    </owl:Class>
```

```
    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#fribrinogen -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#fribrinogen">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#TestName"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#haemoglobin -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#haemoglobin">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#TestName"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#interventionReason -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#interventionReason">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Medication"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#pharmaceuticalDrug -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#pharmaceuticalDrug">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Medication"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#plateletCount -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#plateletCount">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#TestName"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#prothrombinTime -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#prothrombinTime">
        <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#TestName"/>
    </owl:Class>
```

```
    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#radiationReason -
->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#radiationReason">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Radiation"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#surgeryReason -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#surgeryReason">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Surgery"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#totalWhiteCellCount -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#totalWhiteCellCount">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#TestName"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#ENT_(Ear,_Nose,_and_Throat) -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#ENT_(Ear,_Nose,_and_Throat)">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Specialist"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Spontaneous_abortion_/_Miscarriage -->

    <owl:Class rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Spontaneous_abortion_/_Miscarriage">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Abortion"/>
    </owl:Class>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////
    //
    // Individuals
    //
    ///////////////////////////////////////////////////////////////////////////////////
```

```
     -->


    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#American_Indian_or_Alaska_Native -->

    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#American_Indian_or_Alaska_Native">
        <rdf:type rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Ethnicity"/>
    </owl:NamedIndividual>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Asian -->

    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Asian">
        <rdf:type rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Ethnicity"/>
    </owl:NamedIndividual>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Black_or_African_American -->

    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Black_or_African_American">
        <rdf:type rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Ethnicity"/>
    </owl:NamedIndividual>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Caucasian_or_White -->

    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Caucasian_or_White">
        <rdf:type rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Ethnicity"/>
    </owl:NamedIndividual>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Ex-Smoker -->

    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Ex-Smoker">
        <rdf:type rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#SmokingStatus"/>
    </owl:NamedIndividual>



    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-
20#Native_Hawaiian_or_Other_Pacific_Islander -->

    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Native_Hawaiian_or_Other_Pacific_Islander">
```

```
        <rdf:type rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Ethnicity"/>
    </owl:NamedIndividual>


    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Never_Smoker -->

    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Never_Smoker">
        <rdf:type rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#SmokingStatus"/>
    </owl:NamedIndividual>


    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OtherEducationLevel -->

    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OtherEducationLevel">
        <rdf:type rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#EducationLevel"/>
    </owl:NamedIndividual>


    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Primary -->

    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Primary">
        <rdf:type rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#EducationLevel"/>
    </owl:NamedIndividual>


    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Secondary -->

    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Secondary">
        <rdf:type rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#EducationLevel"/>
    </owl:NamedIndividual>


    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Smoker -->

    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Smoker">
        <rdf:type rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#SmokingStatus"/>
    </owl:NamedIndividual>


    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Tertiary -->

    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Tertiary">
        <rdf:type rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#EducationLevel"/>
```

```
    </owl:NamedIndividual>


    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#normalWeight -->

    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#normalWeight">
        <rdf:type rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#BodyMassIndex"/>
    </owl:NamedIndividual>


    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#obesity -->

    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#obesity">
        <rdf:type rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#BodyMassIndex"/>
    </owl:NamedIndividual>


    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#overWeight -->

    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#overWeight">
        <rdf:type rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#BodyMassIndex"/>
    </owl:NamedIndividual>


    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#underWeight -->

    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#underWeight">
        <rdf:type rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#BodyMassIndex"/>
    </owl:NamedIndividual>


    <!-- http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Hispanic_/_Latino
-->

    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Hispanic_/_Latino">
        <rdf:type rdf:resource="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-
ontology-20#Ethnicity"/>
    </owl:NamedIndividual>


    <!--
    ///////////////////////////////////////////////////////////////////////////////////
    //
    // General axioms
    //
    ///////////////////////////////////////////////////////////////////////////////////
     -->
```

```
    <rdf:Description>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AllDisjointClasses"/>
        <owl:members rdf:parseType="Collection">
            <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Female"/>
            <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#Male"/>
            <rdf:Description
rdf:about="http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#OtherGender"/>
        </owl:members>
    </rdf:Description>
</rdf:RDF>



<!-- Generated by the OWL API (version 4.5.9.2019-02-01T07:24:44Z) https://github.com/owlcs/owlapi
-->
```

8.3. Back-End

8.3.1. Application.properties

```
server.port=8090
logging.file.path=C:/Users/spappas/Desktop/diplomaLogs
```

8.3.2. pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-
4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.3.0.RELEASE</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.diploma</groupId>
	<artifactId>diploma-service</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>diploma-service</name>
	<description>Spring boot service for Spyros' diploma thesis</description>

	<properties>
		<java.version>11</java.version>
```

```
 </properties>

 <dependencies>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-data-rest</artifactId>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-web-services</artifactId>
        </dependency>

        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-test</artifactId>
                <scope>test</scope>
                <exclusions>
                        <exclusion>
                                <groupId>org.junit.vintage</groupId>
                                <artifactId>junit-vintage-engine</artifactId>
                        </exclusion>
                </exclusions>
        </dependency>
        <dependency>
  <groupId>org.apache.jena</groupId>
  <artifactId>jena-core</artifactId>
  <version>[3.10.0,3.10.99]</version>
</dependency>
<dependency>
  <groupId>org.apache.jena</groupId>
  <artifactId>jena-arq</artifactId>
  <version>[3.10.0,3.10.99]</version>
</dependency>
<dependency>
  <groupId>net.sourceforge.owlapi</groupId>
  <artifactId>owlapi-distribution</artifactId>
  <version>[5.1.9,)</version>
</dependency>
<dependency>
  <groupId>org.jgrapht</groupId>
  <artifactId>jgrapht-ext</artifactId>
  <version>1.1.0</version>
</dependency>
<dependency>
  <groupId>org.antlr</groupId>
  <artifactId>antlr-runtime</artifactId>
  <version>[3.5.2,3.5.99]</version>
</dependency>
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>[1.7.25,1.7.99]</version>
</dependency>
<dependency>
```

```
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-log4j12</artifactId>
  <version>[1.7.25,1.7.99]</version>
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>jcl-over-slf4j</artifactId>
  <version>[1.7.25,1.7.99]</version>
</dependency>
<dependency>
  <groupId>log4j</groupId>
  <artifactId>log4j</artifactId>
  <version>[1.2.17,1.2.99]</version>
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-simple</artifactId>
  <version>[1.7.25,1.7.99]</version>
</dependency>
<dependency>
  <groupId>org.jfree</groupId>
  <artifactId>jfreechart</artifactId>
  <version>1.5.0</version>
</dependency>
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-math3</artifactId>
  <version>[3.6.1,3.6.99]</version>
</dependency>
<dependency>
  <groupId>gnu.getopt</groupId>
  <artifactId>java-getopt</artifactId>
  <version>[1.0.13,)</version>
</dependency>
<dependency>
  <groupId>org.eclipse.jetty</groupId> <!-- Used for the examples. -->
  <artifactId>jetty-server</artifactId>
  <version>9.4.18.v20190429</version>
</dependency>

<!-- Dependencies of dependencies -->
<dependency>
   <groupId>org.apache.thrift</groupId>
   <artifactId>libthrift</artifactId>
   <version>[0.12.0,)</version>
</dependency>
<dependency>
   <groupId>com.fasterxml.jackson.core</groupId>
   <artifactId>jackson-core</artifactId>
   <version>2.10.3</version>
</dependency>
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-annotations</artifactId>
  <version>2.10.3</version>
</dependency>
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>2.10.3</version>
</dependency>
<dependency>
   <groupId>com.github.jsonld-java</groupId>
```

```
         <artifactId>jsonld-java</artifactId>
         <version>0.12.3</version>
      </dependency>
      <dependency>
         <groupId>com.google.inject</groupId>
         <artifactId>guice</artifactId>
         <version>4.2.2</version>
      </dependency>
      <dependency>
         <groupId>com.google.guava</groupId>
         <artifactId>guava</artifactId>
         <version>27.1-jre</version>
      </dependency>
      <dependency>
        <groupId>commons-io</groupId>
         <artifactId>commons-io</artifactId>
         <version>2.6</version>
      </dependency>
      <dependency>
         <groupId>javax.xml.bind</groupId>
         <artifactId>jaxb-api</artifactId>
         <version>2.3.0</version>
      </dependency>
      <dependency>
               <groupId>com.github.galigator.openllet</groupId>
               <artifactId>openllet-owlapi</artifactId>
               <version>2.6.5</version>
       </dependency>
       <dependency>
               <groupId>com.github.galigator.openllet</groupId>
               <artifactId>openllet-jena</artifactId>
               <version>2.6.5</version>
       </dependency>
       <dependency>
        <groupId>org.mongodb</groupId>
        <artifactId>mongo-java-driver</artifactId>
        <version>3.12.5</version>
    </dependency>
</dependencies>
       <build>
               <plugins>
                       <plugin>
                               <groupId>org.springframework.boot</groupId>
                               <artifactId>spring-boot-maven-plugin</artifactId>
                       </plugin>
               </plugins>
       </build>

</project>
```

8.3.3. Logback.xml

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

        <appender name="consoleAppender" class="ch.qos.logback.core.ConsoleAppender">
                <encoder>
                                        <PatternLayout noConsoleNoAnsi="true" disableAnsi="true"
pattern="%style{%d} %style{[%t]} %style{%-5level:} %style{%msg%n%throwable{short}}"/>
                               </encoder>
```

```
                <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
                        <level>debug</level>
                </filter>
        </appender>

        <appender name="dailyRollingFileAppender"
class="ch.qos.logback.core.rolling.RollingFileAppender">
                <File>C:/Users/spappas/Desktop/diplomaLogs/diploma.log</File>
                <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
                        <FileNamePattern>C:/Users/spappas/Desktop/diplomaLogs/diploma.%d{yyyy-MM-
dd}.%i.log.gz</FileNamePattern>
                                        <timeBasedFileNamingAndTriggeringPolicy

       class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
                                                <maxFileSize>80MB</maxFileSize>
                                        </timeBasedFileNamingAndTriggeringPolicy>
                        <maxHistory>300</maxHistory>
                </rollingPolicy>
                                <encoder>
                                        <PatternLayout noConsoleNoAnsi="true" disableAnsi="true"
pattern="%style{%d} %style{[%t]} %style{%-5level:} %style{%msg%n%throwable{short}}"/>
                                </encoder>
                <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
                        <level>debug</level>
                </filter>
        </appender>



        <root level="info">
                <appender-ref ref="consoleAppender" />
                <appender-ref ref="dailyRollingFileAppender" />
        </root>

              <logger name="com.diploma" level="debug" additivity="false">
               <appender-ref ref="consoleAppender" />
               <appender-ref ref="dailyRollingFileAppender" />
           </logger>


</configuration>
```

8.3.4. AsyncConfiguration.java

```
package com.diploma.diplomaservice;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.scheduling.annotation.EnableAsync;
import org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor;
import java.util.concurrent.Executor;

@Configuration
@EnableAsync
```

```
public class AsyncConfiguration {
    private static final Logger LOGGER = LoggerFactory.getLogger(AsyncConfiguration.class);

    @Bean (name = "taskExecutor")
    public Executor taskExecutor() {

        LOGGER.debug("Creating Async Task Executor");

        final ThreadPoolTaskExecutor executor = new ThreadPoolTaskExecutor();
        //sets the core number of threads in the pool
        executor.setCorePoolSize(20);
        //sets the maximum number of threads allowed
        executor.setMaxPoolSize(20);
        //sets the queue capacity
        executor.setQueueCapacity(100);

        executor.setThreadNamePrefix("Async Thread-");

        executor.initialize();

        return executor;

    }

}
```

8.3.5. DiplomaServiceApplication.java

```
package com.diploma.diplomaservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DiplomaServiceApplication {

        public static void main(String[] args) {
                SpringApplication.run(DiplomaServiceApplication.class, args);
        }


}
```

8.3.6. MyController.java

```
package com.diploma.diplomaservice;
import java.net.UnknownHostException;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
```

```
import org.springframework.web.bind.annotation.RestController;

import com.diploma.searcher.JenaReasoner;



@RestController
public class MyController {

    private static final Logger LOGGER = LoggerFactory.getLogger(MyController.class);

    //CrossOrigin anotation gives the front end the ability to use the fetch method
       @CrossOrigin(origins = "*")
       @RequestMapping(value = "/simple", method = RequestMethod.GET)
       public String simple(@RequestParam(name = "tag1") String tag1,
                       @RequestParam(name = "tag2") String tag2, @RequestParam(name = "tag3") String
tag3,
                       @RequestParam(name = "tag4") String tag4, @RequestParam(name = "tag5") String
tag5,
                       @RequestParam(name = "tag6") String tag6, @RequestParam(name = "tag7") String
tag7)
                                       throws UnknownHostException {

               long startTime = System.nanoTime();
               LOGGER.info("[MyController] simpleSearch : START");

               JenaReasoner simpleSearchJena = new JenaReasoner();

               String[] theargs = new String[7];
               theargs[0] = tag1;
               theargs[1] = tag2;
               theargs[2] = tag3;
               theargs[3] = tag4;
               theargs[4] = tag5;
               theargs[5] = tag6;
               theargs[6] = tag7;
               simpleSearchJena.simpleSearch(theargs);

               long endTime = System.nanoTime();
               LOGGER.info("[MyController] simpleSearch : END. Elapsed time : "+(endTime-
startTime)/1000000+"ms");

               return simpleSearchJena.getSimple_search_blueprints();

       }

    //CrossOrigin anotation gives the front end the ability to use the fetch method
       @CrossOrigin(origins = "*")
       @RequestMapping(value = "/semantic", method = RequestMethod.GET)
       public String semantic(@RequestParam(name = "tag1") String tag1,
                       @RequestParam(name = "tag2") String tag2, @RequestParam(name = "tag3") String
tag3,
                       @RequestParam(name = "tag4") String tag4, @RequestParam(name = "tag5") String
tag5,
                       @RequestParam(name = "tag6") String tag6, @RequestParam(name = "tag7") String
tag7)
                                       throws UnknownHostException {

               long startTime = System.nanoTime();
               LOGGER.info("[MyController] semanticSearch : START");

               JenaReasoner semanticSearchJena = new JenaReasoner();
```

```
                String[] theargs = new String[7];
                theargs[0] = tag1;
                theargs[1] = tag2;
                theargs[2] = tag3;
                theargs[3] = tag4;
                theargs[4] = tag5;
                theargs[5] = tag6;
                theargs[6] = tag7;
                semanticSearchJena.semanticSearch(theargs);

                long endTime = System.nanoTime();
                LOGGER.info("[MyController] semanticSearch : END. Elapsed time : "+(endTime-
startTime)/1000000+"ms");

                return semanticSearchJena.getSemantic_search_blueprints();

        }

}
```

8.3.7. JenaReasoner.java

```
package com.diploma.searcher;


import java.net.UnknownHostException;


import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import openllet.jena.PelletReasonerFactory;

import org.apache.jena.ontology.Individual;
import org.apache.jena.ontology.OntClass;
import org.apache.jena.ontology.OntModel;
import org.apache.jena.rdf.model.ModelFactory;
import org.springframework.scheduling.annotation.Async;

import com.mongodb.DB;
import com.mongodb.DBCollection;


public class JenaReasoner
{
        static final int ROOT_TO_DIVIDE_GRADE =2;


        public final static int DEFAULT_PARENT_TAG_RANK = 100;

        public final static int RANK_CLASS_SUBCLASS = 80;
        public final static int RANK_CLASS_INDIVIDUALS = 71;
        public final static int RANK_CLASS_SUPERCLASSES = 61;
```

```
        public final static int RANK_IND_DIR_CLASS = 80;
        public final static int RANK_IND_SIBLINGS = 71;
        public final static int RANK_IND_SUPERCLASSES = 59;


        private List<String> inputTags ;
        private TagArray tagsArray;

        //the variables that are returned to the controller and the front-end
        private static String simple_search_blueprints;
        private static String semantic_search_blueprints;

        // path that the ontology resides
        final static String myont = "C:\\Users\\spappas\\Diploma-Ontology.owl";

        final static String mynamespace =
"http://www.semanticweb.org/spappas/ontologies/2020/3/untitled-ontology-20#";

        // the variable refering to the ontology
        OntModel model;

        private static final Logger LOGGER = LoggerFactory.getLogger(JenaReasoner.class);


/* ============================================================================
 * ============================================================================
 * ============================================================================
 */

        public static String getSimple_search_blueprints() {
                return simple_search_blueprints;
        }

        public static void setSimple_search_blueprints(String default_search_blueprints) {
                JenaReasoner.simple_search_blueprints = default_search_blueprints;
        }

        public static String getSemantic_search_blueprints() {
                return semantic_search_blueprints;
        }

        public static void setSemantic_search_blueprints(String advanced_search_blueprints) {
                JenaReasoner.semantic_search_blueprints = advanced_search_blueprints;
        }


        /*
         * Implements a basic search in the mongodb database based on the user's tags.
         * There is a query in the database returning documents containing at least one of the tags.
         * Then, the grader grades the documents before returning the to the user
         */
        @Async
        public void simpleSearch(final String[] args)
        {
                LOGGER.info("simpleSearch : Start");

                // -----------------     START OF INPUT     -----------------

                if (args.length > 0)
         {
             String tmpargs = "";
             tagsArray = new TagArray();
             for (String val:args) {
```

```
            if (val.isBlank()) {
                    continue;
            }
                    tmpargs=tmpargs+val+" ";
                 tagsArray.addParentTag(val);
            }
            LOGGER.info("The command line arguments are:"+tmpargs);
        }
        else
            LOGGER.error("No command line arguments found.");

            // -----------------     END OF INPUT     -----------------

            /* -----------------     START OF  BLUEPRINTS RETRIEVE, GRADE AND SORT     -----------
------ */

            long startTime = System.nanoTime();
            LOGGER.info("simpleSearch retrieveAndGradeBlueprint : START");
            List<ResultDocument> ret = retrieveAndGradeBlueprint(tagsArray);
            long endTime = System.nanoTime();
            LOGGER.info("simpleSearch retrieveAndGradeBlueprint : END. Elapsed time : "+(endTime-
startTime)/1000000+"ms");

            LOGGER.info("simpleSearch : END");

            /* -----------------     END OF  BLUEPRINTS RETRIEVE, GRADE AND SORT     -------------
---- */

            /* -----------------     START OF JSON CREATION     ----------------- */

            /*
             * Given an input of
             * - blueprint1: [Doctor, Smoker, Patient] with a grade of 19 and an id of
5f465d52a93bfd39b4105f84 (as stored in MongoDb)
             * - blueprint2: [Nurse, Gynecologist, Female] with a grade of 21 and an id of
5f465d52a93bfd39b41061d9 (as stored in MongoDb)
             * The result JSON will be :
             * {
             *             "blueprint" : ["[Doctor, Smoker, Patient] - 19.0","[Nurse,
Gynecologist, Female] - 21.0"],
             *             "id" :["5f465d52a93bfd39b4105f84","5f465d52a93bfd39b41061d9"]
             * }
             * */

            long startJson = System.nanoTime();

            String jsoned = "{\"blueprint\" : [";
            String jsonedId = "";
            jsoned += "\""+ret.get(ret.size()-1).getDoc()+" - "+ret.get(ret.size()-
1).getGrade()+"\"";
            jsonedId += "\""+ret.get(ret.size()-1).getId()+"\"";

            for (int i=(ret.size()-2); i>=0; i--) {
                    jsoned += ",\""+ret.get(i).getDoc()+" - "+ret.get(i).getGrade()+"\"";
                    jsonedId += ",\""+ret.get(i).getId()+"\"";
            }
            jsoned += "], \"id\" : [";
            jsoned += jsonedId;
            jsoned += "]}";

            setSimple_search_blueprints(jsoned);

            long endJson = System.nanoTime();
```

```
                LOGGER.info("[simpleSearch] jsonize : json: "+jsoned);
                LOGGER.info("[simpleSearch] jsonize : END. Elapsed time : "+(endJson-
startJson)/1000000+"ms");

        }

        /*
         * Implements a semantic search in the mongodb database based on the user's tags.
         * Brings back the
         * There is a query in the database returning documents containing at least one of the tags.
         * Then, the grader grades the documents before returning the to the user
         */
        @Async
        public void semanticSearch(final String[] args) throws UnknownHostException
        {
                long semanticStart = System.nanoTime();
                LOGGER.info("[semanticSearch] : Start");

                // -----------------     START OF ONTOLOGY INPUT+REASONING     -----------------


                LOGGER.info("[semanticSearch] : Reading and reasoning the ontology");
                long ReasstartTime = System.nanoTime();
                //create an empty ontology model using Pellet spec
                model = ModelFactory.createOntologyModel(PelletReasonerFactory.THE_SPEC);
                // read the file
                model.read(myont);

                long ReasendTime = System.nanoTime();
                LOGGER.info("[semanticSearch] : Ontology reasoning. Elapsed time : "+(ReasendTime-
ReasstartTime)/1000000+"ms");

                // -----------------     END OF ONTOLOGY INPUT+REASONING     -----------------


                // -----------------     START OF INPUT     -----------------

                // a variable containing the tags as inputted by the user
                //managing the args. If an arg is empty (val.isBlank(), do not add it in the inputTags
or tagsArray
                inputTags = new ArrayList<String>();
                if (args.length > 0)
        {
            String tmpargs = "";
            tagsArray = new TagArray();
            for (String val:args) {
               if (val.isBlank()) {
                       continue;
               }
                       tmpargs=tmpargs+val+" ";
                inputTags.add(val);
                tagsArray.addParentTag(val);
            }
            LOGGER.info("[semanticSearch] : The command line arguments are:"+tmpargs);
        }
        else
            LOGGER.error("[semanticSearch] : No command line arguments found.");

                // -----------------     END OF INPUT     -----------------

                // -----------------     ONTOLOGY ENRICHING OF THE TAGSARRAY     -----------------
```

```
        LOGGER.info("[semanticSearch] : Enriching from ontology : Start");
        long ontStartTime = System.nanoTime();

                int index = 0;
                for (String tagIter : inputTags) {
                        LOGGER.debug("[semanticSearch] : "+inputTags+"  iter:"+tagIter);
                        OntClass tagClass = model.getOntClass(mynamespace + tagIter);
                        String forLog = "";
                        if (tagClass==null) {
                                /* -----------------     INDIVIDUAL     -----------------

                                 * If one of the tags is an individual get in the following order:
                                 * 1) blueprints with the individual
                                 * 2) blueprints with the direct class of the individual
                                 * 3) blueprints with the siblings of the individual (individuals in
the same class)
                                 * 4) blueprints with the direct superclass of the individual's class
                                 * */



                                LOGGER.debug("[semanticSearch] : "+tagIter+" is NOT a Class");

                                Individual tagInd = model.getIndividual(mynamespace+tagIter);

                                // if tagClass==null, it means that the element is NOT a class (and not
an individual). It may not exist in general though
                                if (tagInd==null) {
                                        LOGGER.debug("[semanticSearch] : "+tagIter+" is NOT a Class");
                                        break;
                                }


                                // ------------->  THE CLASS OF THE INDIVIDUAL: i.e individual Asian --
-> Class Ethnicity
                                OntClass directClass = tagInd.getOntClass(true);

                                tagsArray.addChildTag(index,
removeNs(directClass.toString(),mynamespace), RANK_IND_DIR_CLASS);
                                LOGGER.debug("[semanticSearch] : ["+tagIter+"] | Found the direct class
"+directClass.toString());


                                // -------------> THE SIBLINGS OF THE INDIVIDUAL: i.e individual Asian
---> individual Caucasian_or_White
                                List<String> tmp2 =
ontologyIterator(directClass.listInstances(true),mynamespace);
                                for (String m : tmp2) {
                                        tagsArray.addChildTag(index, m, RANK_IND_SIBLINGS);
                                        forLog = forLog+m+" ";
                                }
                                LOGGER.debug("[semanticSearch] : ["+tagIter+"] |
"+directClass.toString()+"--individuals-->"+forLog);


                                // -------------> THE SUPERCLASSES OF THE INDIVIDUAL'S CLASS : i.e
individual Asian --. class Ethnicity --> superclass Demographics
                                tmp2 =
ontologyIterator(directClass.listSuperClasses(false),mynamespace);
                                forLog = "";
                                for (String m : tmp2) {
                                        //store the superclasses (false means not only direct) in the
tagsArray
```

```
                                tagsArray.addChildTag(index, directClass.toString(),
RANK_IND_SUPERCLASSES);
                                forLog = forLog+m+" ";
                        }
                        LOGGER.debug("[semanticSearch] : ["+tagIter+"] | "+directClass+"--
superclasses-->"+forLog);


                        index++;
                }
                else {
                        /* -----------------     CLASS     -----------------
                         * If one of the tags is a Class get in the following order:
                         * 1) blueprints with the Class
                         * 2) blueprints with the direct subclass of the Class
                         * 3) blueprints with the superclass of the Class
                         * 4) blueprints with the individuals of the Class
                         * */

                        LOGGER.debug("[semanticSearch] : tagIter: ["+tagIter+"] is a
class!!!!");

                        // -------------> SUBCLASSES OF THE CLASS : i.e Doctor ---> Specialist
                        //store the subclass(es) in the tagsArray
                        List<String> tmp =
ontologyIterator(tagClass.listSubClasses(true),mynamespace);
                        forLog="";
                        for (String m : tmp) {
                                tagsArray.addChildTag(index, m, RANK_CLASS_SUBCLASS);
                                forLog = forLog+m+" ";
                        }
                        LOGGER.debug("[semanticSearch] : "+tagClass+"--subclasses-->"+forLog);


                        // -------------> INDIVIDUALS OF THE CLASS : i.e class Doctor --->
individual Neurologist
                        //store the siblings in the tagsArray
                        tmp = ontologyIterator(tagClass.listInstances(true),mynamespace);
                        forLog="";
                        for (String m : tmp) {
                                tagsArray.addChildTag(index, m, RANK_CLASS_INDIVIDUALS);
                                forLog=forLog+m+" ";
                        }
                        LOGGER.debug("[semanticSearch] : "+tagClass+"--individuals-->"+forLog);


                        // -------------->  SUPERCLASSES OF THE CLASS : i.e class Doctor --->
superclass Person
                        //store the superclasses (false means not only direct) in the tagsArray
                        tmp = ontologyIterator(tagClass.listSuperClasses(false),mynamespace);
                        forLog="";
                        for (String m : tmp) {
                                tagsArray.addChildTag(index, m, RANK_CLASS_SUPERCLASSES);
                                forLog=forLog+m+" ";
                        }
                        LOGGER.debug("[semanticSearch] : "+tagClass+"--superclasses--
>"+forLog);

                        index++;
                }

        }
```

```
            /* -----------------     END OF ONTOLOGY ENRICHING     ----------------- */

        long ontEndTime = System.nanoTime();
        LOGGER.info("[semanticSearch] : End of Ontology enriching. Elapsed time : "+(ontEndTime-
ontStartTime)/1000000+"ms");

        /* -----------------     RETRIEVE BLUEPRINTS, GRADE THEM AND SORT THEM     -----------------
*/

            /*
             * for each tag, there will be a ranking
             * given the tags-ranking table, retrieveFromMongo will use logic AND queries meaning
:
             * I will have a limit, how many results do i want??
             * if i want n-results do until n : use the combo of tag1, tag2,...tag7 with the
highest rank
             * i.e tag1=Doctor(rank 10) --> superClass:Person(rank 4) --> subClass:Specialist(rank
3)
             *        tag2=Patient(rank 10) --> superClass:Person(rank 4)
             *     tag3=Female(rank 10) --> superClass:Gender(rank 4)
             * */

            long startTime = System.nanoTime();
            LOGGER.debug("[semanticSearch] :  retrieveAndGradeBlueprint : START");
            List<ResultDocument> ret2 = retrieveAndGradeBlueprint(tagsArray);
            long endTime = System.nanoTime();
            LOGGER.debug("[semanticSearch] :  retrieveAndGradeBlueprint : END. Elapsed time :
"+(endTime-startTime)/1000000+"ms");

            /* -----------------     END OF  BLUEPRINTS RETRIEVE, GRADE AND SORT     -------------
---- */

            /* -----------------     START OF JSON CREATION     ----------------- */

            /*
             * Given an input of
             * - blueprint1: [Doctor, Smoker, Patient] with a grade of 19 and an id of
5f465d52a93bfd39b4105f84 (as stored in MongoDb)
             * - blueprint2: [Nurse, Gynecologist, Female] with a grade of 21 and an id of
5f465d52a93bfd39b41061d9 (as stored in MongoDb)
             * The result JSON will be :
             * {
             *             "blueprint" : ["[Doctor, Smoker, Patient] - 19.0","[Nurse,
Gynecologist, Female] - 21.0"],
             *             "id" :["5f465d52a93bfd39b4105f84","5f465d52a93bfd39b41061d9"]
             * }
             * */
            long jsonStartTime = System.nanoTime();

            LOGGER.debug("[semantic] : Jsonize Start");

            String jsoned = "{\"blueprint\" : [";
            String jsonedId = "";
            jsoned += "\""+ret2.get(ret2.size()-1).getDoc()+" - "+ret2.get(ret2.size()-
1).getGrade()+"\"";
            jsonedId += "\""+ret2.get(ret2.size()-1).getId()+"\"";

            for (int i=(ret2.size()-2); i>=0; i--) {
                    jsoned += ",\""+ret2.get(i).getDoc()+" - "+ret2.get(i).getGrade()+"\"";
                    jsonedId += ",\""+ret2.get(i).getId()+"\"";
            }
            jsoned += "], \"id\" : [";
            jsoned += jsonedId;
```

```
		jsoned += "]}";

		LOGGER.info("[semantic] : Jsonize end. Result :"+jsoned);

		/* -----------------     END OF JSON CREATION     ----------------- */


		//setAdvanced_search_blueprints(jsonize(ret2));
		setSemantic_search_blueprints(jsoned);
		long jsonEndTime = System.nanoTime();
		LOGGER.info("[semanticSearch] :  jsonize : END. Elapsed time : "+(jsonEndTime-
jsonStartTime)/1000000+"ms");

		long semanticEnd = System.nanoTime();
		LOGGER.info("[semanticSearch] :  THE END!!Elapsed time : "+(semanticEnd-
semanticStart)/1000000+"ms");
	}

	/*
	 * Iterate through the ontology, ignoring Thing and Nothing and removing the ns
	 */
	private static List<String> ontologyIterator(final Iterator<?> i, String myns)
	{
		LOGGER.debug("[ontologyIterator] : Start");
		List<String> ret = new ArrayList<String>();

		if (i.hasNext()) {
			while (i.hasNext()) {
				String tmp = i.next().toString();
				if (myns!=null)
					tmp=removeNs(tmp,myns);

				//ignore Thing and Nothing
				if (tmp.equals("Thing") || tmp.equals("Nothing"))
					continue;

				ret.add(tmp);
			}
		}

		return ret;
	}

	/*
	 * Every element of the owl model has a url(ns) in front of it.
	 * Removing it in order to use them
	 * */
	private static String removeNs(String thestring, String myns){
		LOGGER.debug("[removeNs] : remove from "+thestring);

		String owlns = "http://www.w3.org/2002/07/owl#";
		String rdfns = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
		String rdfns2 = "http://www.w3.org/2000/01/rdf-schema#";

		thestring = thestring.replaceAll(owlns, "");
		thestring = thestring.replaceAll(rdfns, "");
		thestring = thestring.replaceAll(rdfns2, "");
		thestring = thestring.replaceAll(myns, "");

		return thestring;

	}
```

```
	/* First part queries mongodb database for each tag there is
	 * Second part, calls GradeDocument to grade each blueprint by its rank
	 */
	private static List<ResultDocument> retrieveAndGradeBlueprint(TagArray tag_array) {

		LOGGER.info("[retrieveAndGradeBlueprint] : Start");

		List<String> blueprintsString = new ArrayList<String>();
		List<String> idsString = new ArrayList<String>();

		ArrayList<ResultDocument> result = new ArrayList<ResultDocument>();

		LOGGER.info("[retrieveAndGradeBlueprint] : Querying MongoDb");
		long startQueryTime = System.nanoTime();
		MongoApp mongoQuery = new MongoApp();

		mongoQuery.query(tag_array);

		blueprintsString=mongoQuery.getBlueprints();
		idsString=mongoQuery.getIds();
		long endQueryTime = System.nanoTime();
		LOGGER.info("[retrieveAndGradeBlueprint] : Querying MongoDb finished. Elapsed time :
"+(endQueryTime-startQueryTime)/1000000+"ms");


		LOGGER.debug("[retrieveAndGradeBlueprint] : blueprints returned:
(size="+blueprintsString.size()+") "+blueprintsString);
		if (blueprintsString.size()==0) {
			LOGGER.error("[retrieveAndGradeBlueprint] : Zero blueprints, returning...");
			return null;
		}

		/* retrieve document for each tag Group
		 * for every blueprint, GradeDocument returns a calculated gradefloat
		 * every pair of blueprint-grade is added to the result(ResultDocument) list
		 * grade = calculatedGrade / N^(1/2), where N: number of inputs
		*/
		long startTime = System.nanoTime();
		LOGGER.info("[retrieveAndGradeBlueprint] : Grading all the blueprints
(size:"+blueprintsString.size()+")");
		for (int blueprintIndex=0; blueprintIndex<blueprintsString.size(); blueprintIndex++) {
			float dynamic_grade = dynamicGrade(blueprintsString.get(blueprintIndex),
tag_array);

			//Splitting the blueprint with the comma, there is an array whose size is
equals to the number of blueprint tags
			int numbOfBluepTags = blueprintsString.get(blueprintIndex).split(",").length;

			float grade = (float) (dynamic_grade / (Math.pow(numbOfBluepTags,
1.0/ROOT_TO_DIVIDE_GRADE )));

			result.add(new ResultDocument(blueprintsString.get(blueprintIndex),
grade,idsString.get(blueprintIndex)));
			LOGGER.debug("[retrieveAndGradeBlueprint] : blueprint
("+blueprintsString.get(blueprintIndex)+") has a grade
("+dynamic_grade+"/root"+ROOT_TO_DIVIDE_GRADE+"("+numbOfBluepTags+")="+grade);
		}
		long endTime = System.nanoTime();
		LOGGER.info("[retrieveAndGradeBlueprint] : Grading ended. Elapsed time : "+(endTime-
startTime)/1000000+"ms. Elapsed per blueprint : "+((endTime-
startTime)/1000000)/blueprintsString.size()+"ms/bp");

		startTime = System.nanoTime();
```

```
                sort (result,0,result.size()-1);
                endTime = System.nanoTime();
                LOGGER.info("[retrieveAndGradeBlueprint] : Sorting ended. Elapsed time : "+(endTime-
startTime)/1000000+"ms");


                return result;
        }

        /*
         * Given the blueprint, there is a recursion of the dynamicGrade where the point is to find
the max grade.
         * ATTENTION : 1) Always take the highest rank tag from a tagGroup 2) Never take more than 1
tags from the same tagGroup
         * i.e : tagGroup1) Doctor(rank15)-Specialist(rank7) tagGroup2) Specialist(rank15) ,
blueprint:"Smoker,Doctor,Specialist
         * the grade is 15(Doctor)+15(Specialist) and NOT :
         *             -15(Doctor)+7(Specialist)
         *             -15(Doctor)+7(Specialist)+15(Specialist)
         */
        private static float dynamicGrade(String blueprint, TagArray tagGroups){
                /*
                 * Find the first tag in the first tagGroup that matches a blueprint-element.
                 * Keep the grade and re-call the dynamicGrade without the element in the blueprint
and the tagGroup=isUsed(true)
                 * When the deepest call of dynamicGrade ends, recursively add the grades.
                 * Continue with the rest tags in the rest tagGroups until you find the highest grade
                 */

                LOGGER.debug("[dynamicGrade] : blueprint : "+blueprint);

                float maxgrade =0;
                String tmpBlueprint = blueprint;
                // Since the blueprint String is like "[FamilyDoctor, Doctor, Patient]", remove the
brackets so it is "FamilyDoctor, Doctor, Patient"
                // and can be split to each tag separately
                tmpBlueprint = tmpBlueprint.replace("[", "");
                tmpBlueprint = tmpBlueprint.replace("]", "");

                //at this point, tmpBlueprint is like : "Doctor, Smoker, Patient"
                // for each group of tags i.e 1) Doctor-->Person(SuperClass)-->Specialist(Subclass) 2)
Patient-->Person(SuperClass)
                for (int tagGroupIndex=0; tagGroupIndex<tagGroups.sizeOfArgs(); tagGroupIndex++) {
                        // for each tag in the group i.e Doctor, Person, Specialist
                        if (tagGroups.isUsed[tagGroupIndex])
                                continue;
                        for (int tagIndex=0; tagIndex<tagGroups.tagGroup.get(tagGroupIndex).size();
tagIndex++) {

                                String tagName = tagGroups.getChildNameByIndex(tagGroupIndex,
tagIndex);
                                int tagRank = tagGroups.getChildRankByIndex(tagGroupIndex, tagIndex);

                                //if the blueprint contains the tagName AND the tagGroup has not been
already used
                                if (tmpBlueprint.contains(tagName) && tagName!="") {
                                        tmpBlueprint = tmpBlueprint.replace("[", "");
                                        tmpBlueprint = tmpBlueprint.replace("]", "");

                                        //To access each tag of the blueprint, it is split removing the
comma and the space ", " and put into a String array
                                        String[] elementdoc = tmpBlueprint.split(", ");
```

```
                                        //after splitting, use again the original form of the blueprint
in the replacement procedure
                                        tmpBlueprint = blueprint;
                                        //find which element of the blueprint is equal with the tagName
                                        for (String stringIter : elementdoc) {
                                                if (stringIter.contentEquals(tagName)) {
                                                        tagGroups.isUsed[tagGroupIndex]=true;

                                                        //remove the tag-element from the blueprint
                                                        //case [FamilyDoctor, Doctor, Patient], removing
the FamilyDoctor --> [Doctor, Patient]
                                                        tmpBlueprint =
tmpBlueprint.replace("["+stringIter+", ", "[");

                                                        //case [FamilyDoctor, Doctor, Patient], removing
the Patient --> [FamilyDoctor, Doctor]
                                                        tmpBlueprint = tmpBlueprint.replace(",
"+stringIter+"]", "]");

                                                        //case [FamilyDoctor, Doctor, Patient], removing
the Doctor --> [FamilyDoctor, Patient]
                                                        tmpBlueprint = tmpBlueprint.replace(",
"+stringIter+", ", ", ");


                                                        //case [Doctor], removing the Doctor --> []
                                                        tmpBlueprint =
tmpBlueprint.replace("["+stringIter+"]", "");


                                                        //recursively call dynamicGrade for the rest of
the blueprint
                                                        float tmpgrade = tagRank;
                                                        tmpgrade +=  dynamicGrade(tmpBlueprint ,
tagGroups);
                                                        //found the grade until know. if it is greater
than the max hold it
                                                        //either way, continue
                                                        tagGroups.isUsed[tagGroupIndex]=false;
                                                        tmpBlueprint = blueprint;
                                                                if (tmpgrade>maxgrade)
                                                                        maxgrade=tmpgrade;
                                                        break;
                                                }
                                        }
                                }
                        }
                }

                LOGGER.debug("[dynamicGrade] : blueprint : "+blueprint+", Returning :"+maxgrade);
                return maxgrade;
        }

        /*
         * Implementing quicksort
         */
    private static void sort(ArrayList<ResultDocument> res, int low, int high)
    {
                LOGGER.debug("[sort] : from "+low+" to "+high);

        if (low < high)
        {
            /* pi is partitioning index, arr[pi] is
              now at right place */
```

```
            int pi = partition(res, low, high);

            // Recursively sort elements before
            // partition and after partition
            sort(res, low, pi-1);
            sort(res, pi+1, high);
        }
    }

    /*
     * Part of the quicksort implementation
     */
    private static int partition(ArrayList<ResultDocument> res, int low, int high)
    {
               LOGGER.debug("[partition] : from "+low+" to "+high);

        float pivot = res.get(high).getGrade();
        int i = (low-1); // index of smaller element
        for (int j=low; j<high; j++)
        {
            // If current element is smaller than the pivot
            if (res.get(j).getGrade() < pivot)
            {
                i++;

                // swap arr[i] and arr[j]
                Collections.swap(res, i, j);
            }
        }

        // swap arr[i+1] and arr[high] (or pivot)
        Collections.swap(res, i+1, high);

        return i+1;
    }

}
```

8.3.8. MongoApp.java

```
package com.diploma.searcher;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.scheduling.annotation.Async;

import com.mongodb.BasicDBObject;
import com.mongodb.DB;
import com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import com.mongodb.DBObject;
```

```
import com.mongodb.MongoClient;
import com.mongodb.MongoClientOptions;

@Async
public class MongoApp {

	static final String MONGO_DATABASE_NAME = "myDatabase";
    static final String MONGO_COLLECTION_NAME = "blueprints";
	static final String MONGO_IP = "localhost";
	static final String MONGO_PORT = "27017";

	private final Logger LOGGER = LoggerFactory.getLogger(JenaReasoner.class);
	private DB database ;
	private DBCollection collection ;

	MongoClientOptions options;
	MongoClient mongoClient;
	List<String> ids;
	List<String> blueprints;




	public List<String> getIds() {
		return ids;
	}

	public void setIds(List<String> ids) {
		this.ids = ids;
	}

	public List<String> getBlueprints() {
		return blueprints;
	}

	public void setBlueprints(List<String> blueprints) {
		this.blueprints = blueprints;
	}

	// constructor : creates the connection with the Mongo database PER THREAD
	public MongoApp () {

		this.blueprints = new ArrayList<String>();
		this.ids = new ArrayList<String>();

		LOGGER.debug("[MongoApp] : Setting the connection");
		this.options = MongoClientOptions.builder().socketKeepAlive(false).build();
		this.mongoClient = new MongoClient(MONGO_IP+":"+MONGO_PORT, options);
		this.database = mongoClient.getDB(MONGO_DATABASE_NAME);
		this.collection = database.getCollection(MONGO_COLLECTION_NAME);

		LOGGER.debug("[MongoApp] : Constructor return");

	}

	/* To query mongodb, a list of basicDBobjects is needed.
	 * Each element of the array list is like "tags" : "Doctor" or "tags" : "Patient"
	 * After adding them, with the use of $or it will be like :
	 * find({ $or : [{"tags":"Doctor"}, {"tags":"Patient"}]})
	 */
	public void query(TagArray tag_array){
		LOGGER.debug("[MongoApp] : returner");
		List<BasicDBObject> advQuery = new ArrayList<BasicDBObject>();
```

```
		BasicDBObject final_query = new BasicDBObject();

		for( ArrayList<TagObject> y : tag_array.tagGroup) {
			for (TagObject h : y) {
				BasicDBObject tmpquery = new BasicDBObject("tags",h.getName());
				advQuery.add(tmpquery);
			}
		}
		final_query.put("$or", advQuery);
		LOGGER.debug("[MongoApp] : Final_query : "+final_query.toString());

		LOGGER.debug("[MongoApp] : Querying DB");
		long startTime = System.nanoTime();
		DBCursor cursor2 = collection.find(final_query);
		long endTime = System.nanoTime();

		Iterator<DBObject> it2 = cursor2.iterator();

		// -----------------     END OF QUERY     -----------------

		//blueprints is a list containing each blueprint as its element
		LOGGER.debug("[MongoApp] : End of Query, processing");

		while(it2.hasNext()) {
			DBObject tmp = it2.next();
			this.blueprints.add(tmp.get("tags").toString());
			this.ids.add(tmp.get("_id").toString());
		}
		LOGGER.debug("[MongoApp] : Query and process finished. Elapsed time : "+(endTime-
startTime)/1000000+"ms.");

		// closes the connection with Mongo for the current thread.
		mongoClient.close();
		LOGGER.debug("[MongoApp] : mongoClient.close()");
		}
}
```

8.3.9. ResultDocument.java

```
package com.diploma.searcher;

public class ResultDocument {

	public String doc;
	public float grade;
	public String id;

	public String getId() {
		return id;
	}

	public void setId(String id) {
```

```
		this.id = id;
	}


	public String getDoc() {
		return doc;
	}
	public void setDoc(String doc) {
		this.doc = doc;
	}
	public float getGrade() {
		return grade;
	}

	public void setGrade(float grade) {
		this.grade = grade;
	}

	public ResultDocument (String doc, float grade2, String id) {
		this.doc = doc;
		this.grade = grade2;
		this.id = id;
	}


}
```

8.3.10. TagArray.java

```
package com.diploma.searcher;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class TagArray {
	/*
	 * tagGroup represents the group that is created from each tag which is :
	 * - the tag alone, if the simple search is used
	 * - tag's superclass/subclass/individuals/siblings/class if the semantic search is used
	 * i.e tag: Doctor, Patient
	 * simpleSearch
	 *             tagGroup[0]=[Doctor]
	 *             tagGroup[1]=[Patient]
	 * semanticSearch
	 *             tagGroup[0]=[Doctor,Person,Specialist,FamilyDoctor]
	 *             tagGroup[1]=[Patient,Person]
	 * */

	public ArrayList<ArrayList<TagObject>> tagGroup = new ArrayList<ArrayList<TagObject>>();
	public int numberOfArgs = 0; //number of tagGroups
	public boolean[] isUsed = new boolean[7];

	// creates a new tagGroup and the parent-tag (the one the user inputted)
	public void addParentTag(String parent) {
		ArrayList<TagObject> array = new ArrayList<TagObject>();
		array.add(new TagObject(parent,JenaReasoner.DEFAULT_PARENT_TAG_RANK));
		tagGroup.add(array);
		numberOfArgs++;
```

```
	}

	//inserts a relevant tag in the tagGroup
	public void addChildTag(int parentIndex, String child, int rank) {
		tagGroup.get(parentIndex).add(new TagObject(child,rank));
	}

	public int getChildRankByIndex(int parentIndex, int index) {
		return tagGroup.get(parentIndex).get(index).getRank();
	}

	public String getChildNameByIndex(int parentIndex, int index) {
		return tagGroup.get(parentIndex).get(index).getName();
	}

	public int sizeOfArgs() {
		return numberOfArgs;
	}
}
```

8.3.11. TagObject.java

```
package com.diploma.searcher;

public class TagObject{
	/*
	 * A tagObject is the object created for any inputted or enriched tag.
	 * */
	public String name ;
	public int rank;

	public TagObject(String name, int rank) {
		this.name = name;
		this.rank = rank;
	}

	public String getName() {
		return name;
	}


	public void setName(String name) {
		this.name = name;
	}


	public int getRank() {
		return rank;
	}

	public void setRank(int rank) {
		this.rank = rank;
	}



}
```