Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης
Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Τίτλος:

Ανάπτυξη διαδικτυακού συστήματος γνώσης με δυνατότητα συνεργατικής ενημέρωση γνώσης με χρήση οντολογιών και διαχείριση αβεβαιότητας

Μιχαήλ Γιαννούλης (ΑΜ:4014)

Επιβλέπων Καθηγητής: Εμμανουήλ Μαρακάκης

Επιτροπή Αξιολόγησης: Εμμανουήλ Μαρακάκης, Χαρίδημος Κονδυλάκης, Νικόλαος Παπαδάκης

Ημερομηνία Παρουσίασης: 17/10/2017

Abstract

The aim of this thesis is to study, the design and the development of a modern online knowledge system for experts. The purpose of the system is to provide valid knowledge for accurate decision support. Our knowledge system, in order to achieve its purpose, it provides to experts, an integrated CRUD knowledge management system where knowledge can be collaboratively updated in real time. It also provides a mechanism called Rule Conflict Detection for detecting contradictory rules, thus it ensures the correctness of the KB. In addition, an expert can create entities and their attributes representing objects on which the domain expert would like to enable decision-making. These entities, which are defined by experts, belong to the expert's knowledge domain. Example of entities could be patient, tree, planet, etc. and possible attributes body temperature, yellowish leaves, atmospheric pressure, etc. respectively. The system via its inference engine has the ability to derive decisions on these entities with a degree of certainty. The derivations are based either on the knowledge base that the expert has constructed with his/her collaborators or the knowledge base that has been constructed from all the experts in the specific knowledge domain. The ambition of this system is to provide to multiple domain experts a powerful decision support system which embodies the knowledge of others experts and his/her own knowledge as well as.

Σύνοψη

Σκοπός της πτυχιακής αυτής εργασίας είναι η μελέτη, η σχεδίαση και η ανάπτυξη ενός σύγχρονου διαδικτυακού συστήματος γνώσης για εμπειρογνώμονες. Σκοπός του συστήματος είναι να παρέχει έγκυρη γνώση με στόχο τη μέγιστη ακρίβεια εξαγωγής αποφάσεων. Το σύστημα γνώσης για να πετύχει το σκοπό του παρέχει στον εμπειρογνώμονα ένα ολοκληρωμένο σύστημα διαχείρισης γνώσης CRUD. Η ενημέρωση της βάσης γνώσης μπορεί να γίνει και συνεργατικά σε πραγματικό χρόνο. Επιπλέον διαθέτει μηχανισμό εντοπισμού και εμπόδιση αντίφασης στη γνώση του συστήματος ο οποίος ονομάζεται Rule Conflict Detection. Όσον αφορά τον στόχο του συστήματος, παρέχεται η δυνατότητα στον εμπειρογνώμονα να δημιουργήσει οντότητες καθώς και τα γνωρίσματά τους. Οι οντότητες ανήκουν στο πεδίο γνώσης του εμπειρογνώμονα, ο οποίος καθορίζει την υπόστασή τους για παράδειγμα ασθενής, δέντρο, πλανήτης, κτλ. και τα χαρακτηριστικά τους για παράδειγμα θερμοκρασίας σώματος, κιτρινισμένα φύλλα, ατμοσφαιρική πίεση, κτλ. αντίστοιχα. Το σύστημα μέσω της μηχανής συλλογισμών που διαθέτει εξάγει αποφάσεις πάνω στις οντότητες αυτές οι οποίες αποφάσεις συνοδεύονται με βαθμό βεβαιότητας. Για την εξαγωγή των αποφάσεων χρησιμοποιήθηκε είτε η βάση γνώσης που ο εμπειρογνώμονας με τους συνεργάτες του έχει κατασκευάσει, είτε η βάση γνώσης που κατασκευάστηκε από όλους τους εμπειρογνώμονες του πεδίου γνώσης του. Η φιλοδοξία του συστήματος μας είναι να αποκτήσει ο κάθε εμπειρογνώμονας ανεξάρτητα της γεωγραφικής θέσης στην οποία βρίσκεται και της οικονομικής του κατάστασης ένα σύστημα στήριξης των αποφάσεων του το οποίο εμπεριέχει τη γνώση πολλών άλλων εμπειρογνωμόνων του αντίστοιχου πεδίου καθώς και τη δική του γνώση.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

Εισαγωγή

Η Γνώση είναι η μετεξέλιξη στον νου της καταγεγραμμένης πληροφορίας και εμπειρίας που παρέχει την δυνατότητα καθοδήγησης στη λήψη αποφάσεων. Ο άνθρωπος ως νοήμων καλείται να λάβει καθημερινά μεγάλο αριθμό αποφάσεων. Αποφάσεις απλοϊκές έως και ιδιαίτερα σημαντικές όπου λάθη από εξωγενείς παράγοντες, αμάθεια ή ημιμάθεια μπορεί να κοστίσουν ανθρώπινες ζωές. Σε μία κρίσιμη λοιπόν κατάσταση χρειάζεται να εξάγει αποφάσεις με ταχύτητα και ακρίβεια ακόμα και αν η λύση βρίσκεται εκτός του πεδίο γνώσεων του.

1.1. Παράθεση Προβλήματος

Στο σημασιολογικό ιστό, στα κοινωνικά δίκτυα και στο διαδίκτυο γενικότερα υπάρχει μεγάλη ποσότητα πληροφορίας και γνώσης ώστε να μπορεί να λυθεί κάθε μορφής επιλύσιμο πρόβλημα. Η μεγάλη διαθεσιμότητα της πληροφορίας οφείλεται κυρίως στο πνεύμα συνεργασίας που διακατέχει τον άνθρωπο, την ανάγκη του να προσφέρει αλλά και στους ευφυείς τρόπους εκμαίευσης της γνώσης του από σύγχρονα συστήματα λογισμικού. Μηχανές αναζήτησης όπως η «*Google*» και εφαρμογές όπως το «*Wiki*» δεν θα είχαν εξελιχθεί στον βαθμό που είναι τώρα αν δεν ίσχυαν τα προαναφερθέντα. Παρ' όλα αυτά η λήψη απόφασης από τον άνθρωπο εξακολουθεί να είναι εξαιρετικά δύσκολη σε περιπτώσεις όπου απαιτείται η επεξεργασία μεγάλου όγκου πληροφορίας η οποία σε αρκετές περιπτώσεις απαιτεί αναδρομική επεξεργασία.

Υπάρχει λοιπόν περισσότερο από ποτέ η ανάγκη για αντικατάσταση του ανθρώπινου τρόπου λήψης αποφάσεων και η εξάλειψη όλων των προβλημάτων με τα οποία συνδέεται ο τρόπος αυτός, με ένα αυτοματοποιημένο τρόπο. Ο αυτοματοποιημένος τρόπος χρειάζεται να δείχνει την αναγκαία εμπιστοσύνη στον άνθρωπο εμπειρογνώμονα κατά την διαδικασία εκμαίευσης γνώσης.

1.2. Σκοπός και Στόχοι της Πτυχιακής Εργασίας

Στόχος αυτής της πτυχιακής εργασίας είναι η δημιουργία ενός συστήματος γνώσης. Σκοπός του συστήματος αυτού, είναι να δώσει τη δυνατότητα σε εμπειρογνώμονες να αποτυπώσουν τις γνώσεις τους συνεργατικά ή ατομικά σε μορφή (if-then) κανόνων. Παράλληλα δίνεται η δυνατότητα στον κάθε εμπειρογνώμονα του συστήματος να καταχωρίσει οντότητες και γνωρίσματα αυτών. Στη συνέχεια ο χρήστης εμπειρογνώμονας μπορεί να επιλέξει ατομική ή δημόσια εξαγωγή αποφάσεων πάνω σε μία οντότητα. Έτσι το σύστημα πραγματοποιεί πυροδότηση μέσω αναδρομικών αλγορίθμων όλων των κανόνων εμπειρογνώμονα (προσωπική βάση γνώσης) ή όλων των κανόνων του συστήματος (δημόσια βάση γνώσης) για το συγκεκριμένο γνωστικό πεδίο στο οποίο ανήκει η οντότητα. Έτσι εξάγεται μία ή περισσότερες τελικές αποφάσεις οι οποίες εμπεριέχουν αβεβαιότητα.

Επιπλέον είναι αναγκαίο να τονίσουμε ότι το πεδίο ορισμού γνώσης του συστήματος αλλά και το πεδίο ορισμού ειδικοτήτων εμπειρογνωμόνων, εξαρτάται αποκλειστικά και μόνο από το εύρος των οντολογιών που χρησιμοποιεί το σύστημα. Συνεπώς η επέκταση λέξεων-φράσεων στις οντολογίες, συνεπάγεται επέκταση τόσο στις ειδικότητες όσο και στα όρια γνώσης. Τέλος η λέξη που επιλέχθηκε ως ονομασία του συστήματος είναι το «*COSMOS*». Η λέξη αυτή προέρχεται από την Ελληνική λέξη «*Κόσμος*» που σημαίνει το σύμπαν, η οποία έχει την ίδια σημασία και στην Αγγλική γλώσσα. Το σύστημά μας σχετίζεται άμεσα με την σημασία της λέξης αυτής τόσο στο επίπεδο αφαιρετικότητας (υποστήριξη πολλών και διαφορετικών πεδίων γνώσης) όσο και στην επεκτασιμότητα (συνεχή επέκταση των πεδίων γνώσης, της γνώσης και των εμπειρογνωμόνων) που το χαρακτηρίζουν.

1.3. Παραδείγματα Χρήσης του Συστήματος COSMOS

Υπάρχουν διάφορες περιπτώσεις χρήσης του συστήματος COSMOS, όπως συμβουλευτικό σύστημα σε δημόσιο ή ιδιωτικό νοσοκομείο, συνεργατικό σύστημα ερευνητή, ως έμπειρο σύστημα εξαγωγής απόφασης σε κάποιο ερευνητικό εργαστήριο, ακόμα και για να προτείνει συνταγές μαγειρικής με βάση τα διαθέσιμα υλικά. Πιο συγκεκριμένα:

- *Η χρήση του συστήματος στον τομέα της Ιατρικής:* Το σύστημα παρέχει τη δυνατότητα σε κάποιο γιατρό να καταχωρήσει την γνώση του υπό μορφή (if-then) κανόνων πάνω σε οποιοδήποτε πεδίο της Ιατρικής. Στη συνέχεια μπορεί να συνεργαστεί με άλλους γιατρούς οποιασδήποτε ειδίκευσης με στόχο την ενημέρωση και βελτίωση της γνώσης που καταχώρησε. Παράλληλα ο γιατρός μπορεί να εισάγει στο σύστημα τους ασθενείς του και την

κλινική τους εικόνα, ως οντότητες και γνωρίσματα αυτών αντίστοιχα. Τέλος παρέχεται στον γιατρό η δυνατότητα εκτέλεσης ατομικής ή δημόσιας διάγνωσης πάνω σε οποιοδήποτε ασθενή (οντότητα) επιθυμεί. Ως αποτέλεσμα διάγνωσης παράγεται μία ή περισσότερες αποφάσεις ως προς την κατάσταση του ασθενή, συνοδευόμενες από κάποιο βαθμό βεβαιότητας ως προς την ισχύ των αποφάσεων αυτών.

- *Η χρήση του συστήματος σε κάποιο ερευνητικό εργαστήριο:* Το σύστημα παρέχει τη δυνατότητα σε ερευνητές να εισάγουν την γνώση τους υπό μορφή (if-then) κανόνων. Η γνώση αυτή θα μπορούσε να είναι κανόνες για αξιολόγηση εργαστηριακών μετρήσεων (dataset). Στη συνέχεια ερευνητές από διάφορα εργαστήρια του ευρύτερου πεδίου γνώσης, μπορούν να συνεργαστούν μεταξύ τους για την ενημέρωση και τη βελτίωση των κανόνων του συστήματος. Παράλληλα ο κάθε ερευνητής μπορεί να καταχωρήσει στο σύστημα τη μελέτη πάνω στην οποία εργάζεται και τις μετρήσεις αυτής, ως οντότητα με γνωρίσματα. Τέλος παρέχεται στον ερευνητή η δυνατότητα εκτέλεσης ατομικής ή δημόσιας εξαγωγής αποφάσεων πάνω σε οποιαδήποτε μελέτη (οντότητα) επιθυμεί. Ως αποτέλεσμα εξαγωγής αποφάσεων παράγεται μία ή περισσότερες αποφάσεις αξιολόγησης μελέτης συνοδευόμενες από κάποιο βαθμό βεβαιότητας ως προς την ισχύ των αποφάσεων αυτών.
- *Η χρήση του συστήματος στην εκπαίδευση:* Το σύστημα παρέχει τη δυνατότητα σε κάποιο εκπαιδευτικό να καταχωρήσει (if-then) κανόνες. Οι κανόνες αυτοί θα μπορούσαν να αναφέρονται στον τρόπο αξιολόγησης μαθητών. Στη συνέχεια ο εκπαιδευτικός μπορεί να συνεργαστεί με άλλους συναδέλφους του, για την βελτίωση των κανόνων. Παράλληλα μπορεί να καταχωρήσει στο σύστημα τους μαθητές του και την επίδοσή τους σε διαγωνίσματα ως οντότητες με γνωρίσματα. Τέλος παρέχεται στον εκπαιδευτικό η δυνατότητα εκτέλεσης ατομικής ή δημόσιας εξαγωγής αποφάσεων πάνω σε οποιοδήποτε μαθητή (οντότητα) επιθυμεί. Ως αποτέλεσμα εξαγωγής αποφάσεων παράγεται μία ή περισσότερες αποφάσεις αξιολόγησης φοιτητή συνοδευόμενες από κάποιο βαθμό βεβαιότητας ως προς την ισχύ των αποφάσεων αυτών.
- *Η χρήση του συστήματος σε δημόσιες υπηρεσίες:* Το σύστημα παρέχει τη δυνατότητα σε κάποιο υπάλληλο εφορίας να καταχωρήσει (if-then) κανόνες. Οι κανόνες αυτοί θα μπορούσαν να αναφέρονται σε οικονομικά και κοινωνικά κριτήρια ενός πολίτη για την απόκτηση επιδομάτων. Στη συνέχεια ο εφοριακός μπορεί να συνεργαστεί με άλλους συναδέλφους του για την ενημέρωση και τη βελτίωση των κανόνων που καταχώρησε. Παράλληλα μπορεί να καταχωρήσει στο σύστημα τους πολίτες και την κοινωνική-οικονομική τους κατάσταση, ως οντότητες με γνωρίσματα. Τέλος παρέχεται στον εφοριακό η δυνατότητα εκτέλεσης ατομικής ή δημόσιας εξαγωγής αποφάσεων πάνω σε οποιοδήποτε πολίτη (οντότητα) επιθυμεί. Ως αποτέλεσμα εξαγωγής αποφάσεων παράγεται μία ή περισσότερες αποφάσεις χορήγησης επιδομάτων και κοινωνικών βοηθημάτων που δικαιούται ο πολίτης συνοδευόμενες από κάποιο βαθμό βεβαιότητας ως προς την ισχύ των αποφάσεων αυτών.

1.4. Δομή Εργασίας

Στο κεφάλαιο 1 παρουσιάζεται το πρόβλημα το οποίο πυροδότησε την ανάγκη δημιουργίας αυτής της πτυχιακής εργασίας, όπως επίσης και εισαγωγική αναφορά σκοπών και στόχων του συστήματος αυτού. Στο κεφάλαιο 2 αναλύονται ορισμένες βασικές έννοιες που πρέπει να συμπεριληφθούν στις γνώσεις του αναγνώστη για ομαλή ανάγνωση της πτυχιακής εργασίας. Στο κεφάλαιο 3 παρουσιάζεται η αρχιτεκτονική του συστήματος σε υψηλό αφαιρετικό επίπεδο. Στο κεφάλαιο 4 αναλύονται όλα τα τμήματα της αρχιτεκτονικής του κεφαλαίου 3. Στο κεφάλαιο 5 αναδεικνύεται η διεπαφή του συστήματος μέσα από ένα ολοκληρωμένο παράδειγμα το οποίο πλαισιώνεται από πλήθος λειτουργιών. Στο κεφάλαιο 6 παρουσιάζονται παραπλήσια συστήματα με αυτό που πραγματεύεται η πτυχιακή εργασία όπως επίσης και σύγκρισή του με αυτά. Στο κεφάλαιο 7 ενημερώνουμε τον αναγνώστη για μελλοντικά σχέδια βελτίωσης και επέκτασης του συστήματος. Στο κεφάλαιο 8 υπάρχει η απαραίτητη βιβλιογραφία στην οποία παραπέμπεται ο αναγνώστης για πληρέστερη ενημέρωση του σε διάφορα θέματα της πτυχιακής όπου η μη κατανόησή τους θα μπορούσε να αποσυντονίσει τον αναγνώστη. Τέλος, στο κεφάλαιο 9 γίνεται αναφορά σε ορισμένους αλγόριθμους παραθέτοντας το κώδικα υλοποίησής τους όπως επίσης και μία σύντομη περιγραφή του σκοπού των αλγορίθμων αυτών.

2. Θεωρητικό Υπόβαθρο

2.1. Συστήματα Γνώσης

«***Σύστημα Γνώσης (Σ.Γ.)***» [1] είναι ένα πρόγραμμα το οποίο εκτελεί συλλογισμούς και χρησιμοποιεί γνώση για να λύσει σύνθετα προβλήματα. Η απόκτηση γνώσης γίνεται από διάφορες πηγές γνώσης όπως βιβλία, εμπειρογνώμονες και άλλες. Η γνώση αναπαρίσταται χρησιμοποιώντας διάφορες μεθόδους αναπαράστασης γνώσης όπως είναι οι κανόνες παραγωγής, η λογική, τα σημασιολογικά δίκτυα και άλλες. Για τον χαρακτηρισμό ενός συστήματος, ως σύστημα γνώσης πρέπει να έχει τουλάχιστον τρία βασικά υποσυστήματα: μία βάση γνώσης, μια μηχανή συλλογισμών και ένα περιβάλλον εργασίας χρήστη με διάφορα εργαλεία υποστήριξης για δημιουργία της βάσης γνώσης και επικύρωσης της. Η βάση γνώσης αντιπροσωπεύει γεγονότα για τον κόσμο του προβλήματος και σχέσεις μεταξύ αυτών των γεγονότων.

2.2. Οντολογίες

Ο όρος «***Οντολογία (Ontology)***» [2] προέρχεται από τη φιλοσοφία. Στο συγκεκριμένο πλαίσιο χρησιμοποιείται ως το όνομα μιας υποπεριοχής της φιλοσοφίας, και συγκεκριμένα της μελέτης της φύσης της ύπαρξης, δηλαδή του κλάδου της μεταφυσικής που ενδιαφέρεται για τον προσδιορισμό, με όσο το δυνατό γενικότερους όρους, των ειδών των πραγμάτων που υπάρχουν αληθινά, καθώς και του τρόπου περιγραφής του. Για παράδειγμα η παρατήρηση πως ο κόσμος αποτελείται από συγκεκριμένα αντικείμενα που μπορούν να ομαδοποιηθούν σε αφηρημένες κατηγορίες (κλάσεις) με βάση κοινές ιδιότητες, είναι μια τυπική οντολογική δέσμευση.

Γενικά μια οντολογία περιγράφει τυπικά ένα πεδίο ενδιαφέροντος. Συνήθως, μια οντολογία αποτελείται από μια πεπερασμένη λίστα όρων και τις σχέσεις μεταξύ αυτών. Οι όροι υποδηλώνουν σημαντικές έννοιες (κλάσεις αντικειμένων) του πεδίου. Τα μέλη του προσωπικού, οι φοιτητές, τα μαθήματα κτλ. Οι οντολογίες είναι χρήσιμες τόσο για την οργάνωση και πλοήγηση σε ιστότοπους όσο και για την βελτίωση της ακρίβειας των αναζητήσεων στον Ιστό. Η τεχνητή νοημοσύνη έχει μεγάλη παράδοση στην ανάπτυξη και τη χρήση των οντολογιών.

Υπάρχουν διάφοροι οργανισμοί που ασχολούνται με την παραγωγή οντολογιών. Ένας από αυτούς είναι το BioPortal [3], ο οποίος χρησιμοποιήθηκε ως πηγή άντλησης οντολογιών για τα πεδία της Ιατρικής και της επιστήμης υπολογιστών.

2.3. Αναπαράσταση Γνώσης

Η περιοχή της «***αναπαράστασης γνώσης (knowledge representation)***» μελετά τους τρόπους οργάνωσης της γνώσης την οποία χρησιμοποιεί στους συλλογισμούς του ένα ευφυές σύστημα. Μερικές από τις πιο σημαντικές μεθόδους αναπαράστασης γνώσης που χρησιμοποιούνται σε συστήματα τεχνητής νοημοσύνης είναι η λογική, τα σημασιολογικά δίκτυα, τα πλαίσια και οι κανόνες παραγωγής ή κανόνες. Η αναπαράσταση γνώσης περιλαμβάνει την συντακτική και σημασιολογική τυποποίηση γνώσης του πεδίου ενός προβλήματος. Βασικός τομέας εφαρμογής της αναπαράστασης γνώσης είναι οι εφαρμογές της τεχνητής νοημοσύνης και ιδιαίτερα τα συστήματα γνώσης που ενδιαφέρουν αυτή τη πτυχιακή εργασία. Η αναπαράσταση γνώσης που επιλέχθηκε για το σύστημα γνώσης που κατασκευάστηκε κατά την πτυχιακή αυτή εργασία, είναι «if-then» κανόνες η σύνταξη των οποίων ορίζεται ως εξής:

If Προϋπόθεση then Συμπέρασμα.

2.4. Θεωρία Βεβαιότητας

Η «***Θεωρία Βεβαιότητας (Certainty Theory CT)***» μετράει το βαθμό εμπιστοσύνης, ή το βαθμό αληθείας, κάποιου γεγονότος ή συμπεράσματος κανόνα. Αυτή η εμπιστοσύνη καθορίζεται από τον εμπειρογνώμονα με ευρετικό τρόπο με βάση την εμπειρία του στο πεδίο του προβλήματος. Η εμπιστοσύνη ή η απιστία του εμπειρογνώμονα για κάποιο γεγονός ή κάποιο κανόνα εκφράζεται με παράγοντες βεβαιότητας. Κάθε παράγοντας βεβαιότητας είναι ένας αριθμός μέσα σε ένα εύρος με το αριστερό άκρο να αντιστοιχεί στο απόλυτο ψευδές (απόλυτη απιστία) ενώ το δεξί άκρο στο απόλυτο

αληθές (απόλυτη εμπιστοσύνη). Ένας παράγοντας βεβαιότητας δεν είναι πιθανότητα ώστε το άθροισμα όλων των ισχυρισμών για κάποιο γεγονός να είναι 1. Οι παράγοντες βεβαιότητας χρησιμοποιήθηκαν για πρώτη φορά στο έμπειρο σύστημα (expert system) MYCIN [4].

2.5. Spark Java Micro Framework

Το «***Spark Java Micro Framework***» είναι ένα μικρό πλαίσιο [5] για τη δημιουργία διαδικτυακών εφαρμογών στην Java 8 χρησιμοποιώντας τα Java 8 lambda expressions, με αρκετά εύκολο τρόπο. Είναι απλό, ελαφρύ και δημιουργήθηκε για το γρήγορη ανάπτυξη διαδικτυακών εφαρμογών. Έχει εμπνευστεί από το Sinatra [6] και δεν ακολουθεί την κλασσική αρχιτεκτονική Model-View-Controller (MVC) όπως άλλα framework (π.χ. Spring MVC). Το spark προεπιλεγμένα τρέχει στον Jetty ο οποίος είναι ο server που διαθέτει, όμως θα μπορούσε χωρίς πρόβλημα να τρέχει και σε άλλους server όπως τον Apache Tomcat [7].

2.6. SWI Prolog

Η «***SWI Prolog***» είναι μια ανοιχτού κώδικα υλοποίηση της δηλωτικής γλώσσας προγραμματισμού Prolog [8], από μία ομάδα του πανεπιστημίου στο Άμστερνταμ. Χρησιμοποιείται τόσο σε on-desktop εφαρμογές όσο και σε εφαρμογές διαδικτύου, καθώς πέραν της πλούσιας βιβλιοθήκης της, παρέχει και δικό της server καθιστώντας την υλοποίηση γρήγορη και αρκετά απλή. Το διαδικτυακό πλαίσιο στο οποίο βασίζεται είναι το Definite Clause Grammar [9] το οποίο αποτελεί ένα τρόπο έκφρασης γραμματικής για γλώσσες λογικού προγραμματισμού όπως η Prolog.

2.7. Free Marker Template

Το «***Free Marker Template***» είναι μια μηχανή προτύπου που βασίζεται στην Java και επικεντρώνεται στην αρχιτεκτονική του λογισμικού Model-View-Controller (MVC). Αν και χρησιμοποιείται κυρίως για την ανάπτυξη εφαρμογών Web Servlet, μπορεί να χρησιμοποιηθεί για οποιοδήποτε άλλο είδος εξόδου κειμένου, όπως η δημιουργία CSS, πηγαίου κώδικα Java κλπ. Σε αντίθεση με το JSP, δεν εξαρτάται από την αρχιτεκτονική Servlet ή από το HTTP. Έτσι μπορεί να χρησιμοποιηθεί και για εργασίες εκτός του Web. Το FreeMarker είναι Ελεύθερο Λογισμικό.

2.8. Json Web Token

Το «***JSON Web Token (JWT)***» είναι ένα ανοικτό πρότυπο (RFC 7519) που ορίζει έναν συμπαγή και αυτόνομο τρόπο για την ασφαλή μετάδοση πληροφοριών μεταξύ server-client ως αντικείμενο JSON [10]. Αυτές οι πληροφορίες μπορούν να επαληθευτούν επειδή υπογράφονται ψηφιακά. Τα JWTs μπορούν να υπογραφούν χρησιμοποιώντας ένα μυστικό κλειδί (με τον αλγόριθμο HMAC) ή ένα ζεύγος δημόσιου / ιδιωτικού κλειδιού χρησιμοποιώντας τον (ασύμμετρο αλγόριθμο) RSA. Το JWT αποτελείται από κεφαλή «*Header*», φορτίο «*Payload*» και υπογραφή «*Signature*». Η κεφαλή περιλαμβάνει το είδος του token (JWT) και τον αλγόριθμο που χρησιμοποιήθηκε (HMAC SHA256 or RSA). Το payload περιλαμβάνει πληροφορία όπως το όνομα του χρήστη στον οποίο αναφέρεται το token, χρονική διάρκεια ισχύος του token αυτού κτλ. Το signature περιλαμβάνει το αποτέλεσμα των ψηφιακά υπογεγραμμένων header και payload.

3. Αρχιτεκτονική του Συστήματος

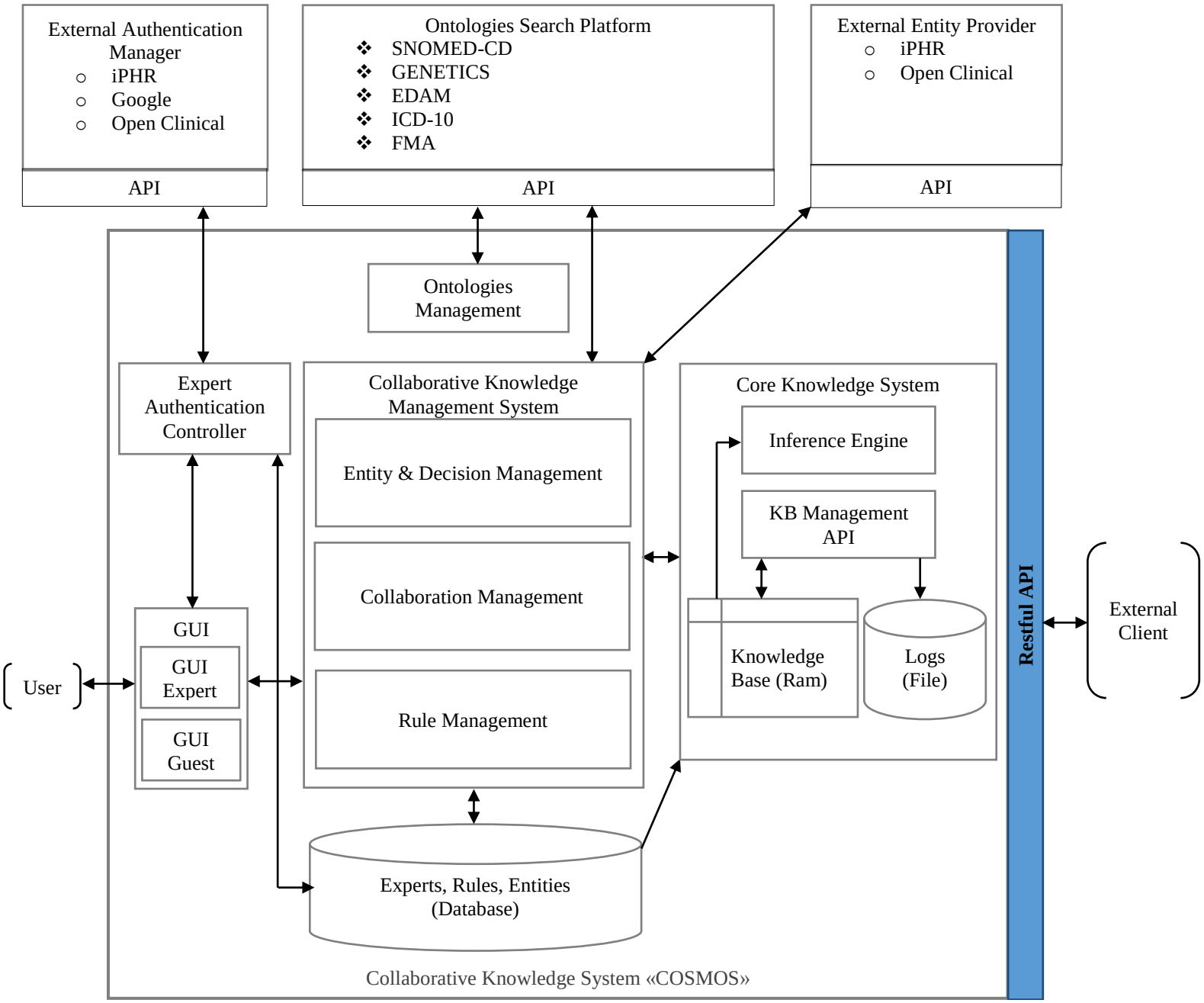

Figure 1 - Αρχιτεκτονική Συστήματος COSMOS

Στην εικόνα 1 βλέπουμε τη γραφική αναπαράσταση ολόκληρου του συστήματος όπως επίσης και όλα τα επιμέρους υποσυστήματα, τους χρήστες αλλά και τα εξωτερικά συστήματα με τα οποία συνεργάζεται. Η παρουσίαση γίνεται σε υψηλό επίπεδο ώστε ο αναγνώστης να καταλάβει πως αλληλοεπιδρούν τα διάφορα τμήματα του συστήματός μας. Παρακάτω γίνεται σύντομη περιγραφή των επιμέρους τμημάτων της αρχιτεκτονικής του συστήματος:

- «***Χρήστης (User)***»: Φυσικό πρόσωπο (άνθρωπος) που χρησιμοποιεί το σύστημα μέσω του γραφικού περιβάλλον χρήστη.
- «***Εξωτερικός πελάτης του συστήματος (External Client)***»: Φυσικό πρόσωπο (άνθρωπος) ή σύστημα (συσκευή) το οποίο επικοινωνεί με το σύστημά μας μέσω του Restful API.

- «***Εξωτερικός Διαχειριστής Ταυτότητας Χρήστη (External Authentication Manager***): Εξωτερικό σύστημα ταυτοποίησης χρηστών. Είναι υπεύθυνο για την ταυτοποίηση των μη καταχωρημένων εμπειρογνωμόνων στο σύστημά μας.
- «***Εξωτερικός Πάροχος Οντοτήτων (External Entity Provider***): Εξωτερικό σύστημα που παρέχει ένα σύνολο οντοτήτων (π.χ. ασθενείς, δέντρα, φοιτητές, κτλ..) του εμπειρογνώμονα, που ενδεχομένως να διαχειρίζονταν στην πλατφόρμα μέσω της οποίας ταυτοποιήθηκε η ειδικότητά του.
- «***Πλατφόρμα Εύρεσης Οντολογιών (Ontologies Search Platform***): Πλατφόρμα που περιλαμβάνει οντολογίες από διάφορα πεδία γνώσης (π.χ. Ιατρική, Επιστήμης Υπολογιστών, κτλ..), προσφέροντας δυνατότητα ερωτήσεων για εντοπισμό ελαστικής ομοιότητας (Elastic Match) αλλά και άντλησης λέξεων – φράσεων.
- Εντός του συστήματος COSMOS περιλαμβάνονται τα εξής:
    - «***Γραφικό Περιβάλλον Χρήστη (GUI)***»: Σύνολο γραφικών στοιχείων που εμφανίζονται στον χρήστη μέσω του διαδικτύου, για την αλληλεπίδρασή του με τις λειτουργίες του συστήματος.
    - «***Σύστημα ελέγχου Ταυτοποίησης Εμπειρογνώμονα (Expert Authentication Controller)»:*** Σύστημα υπεύθυνο για την εξακρίβωση ειδικότητας νέων εμπειρογνωμόνων και για την εξουσιοδότηση ήδη πιστοποιημένων χρηστών του συστήματός μας.
    - «***Σύστημα Διαχείρισης Οντολογιών (Ontologies Management)»:*** Σύστημα υπεύθυνο για την δημιουργία, επέκταση και διαγραφή οντολογιών σε μία εξωτερική πλατφόρμα εύρεσης οντολογιών.
    - «***Σύστημα Διαχείρισης Συνεργατικής Γνώσης (Collaborative Knowledge Management System)»:*** Το σύστημα αυτό είναι υπεύθυνο για την διαχείριση όλων των ενεργειών που γίνονται από τους χρήστες πάνω στις λειτουργίες του συστήματος. Οι ενέργειες αυτές ομαδοποιούνται σε τρείς κατηγορίες: κανόνες (μέσω του Rule Management), οντότητες (μέσω του Entity & Decision Management) και συντάκτες (μέσω του Collaboration Management).
    - «***Πυρήνας Συστήματος Γνώσης (Core Knowledge System)»:*** Το σύστημα αυτό είναι υπεύθυνο για τη διαδικασία εξαγωγής αποφάσεων πάνω σε οντότητες όπως επίσης και για την διαχείριση των εκτελέσιμων κανόνων.
    - «***Βάση Δεδομένων (Database)»:*** Σχεσιακή βάση διατήρησης (εκτελέσιμών ή προσχέδιων) κανόνων, εμπειρογνωμόνων και οντοτήτων.
    - «***Representational state transfer Application Programming Interface*** (***Restful API)»:*** Αποτελεί το παράθυρο επικοινωνίας με τους εξωτερικούς πελάτες του συστήματός μας, προσφέροντας υπηρεσίες απομακρυσμένης άντλησης και καταχώρησης πληροφοριών.

Όσον αφορά την επικοινωνία μεταξύ αυτών των οντοτήτων στο σχήμα ορίζονται δύο ειδών βέλη. Βέλη μονής κατεύθυνσης που δεν περιλαμβάνει feedback και διπλής κατεύθυνσης που αναφέρεται σε αμφίδρομη επικοινωνία. Παρακάτω ακολουθεί σύντομη περιγραφή των επικοινωνιών αυτών καθώς στο επόμενο κεφάλαιο θα υπάρξει περεταίρω ανάλυση:

- Ο User επικοινωνεί με το COSMOS μέσω του GUI που παρέχει το ίδιο το σύστημα.
- Ο Client επικοινωνεί με το COSMOS μέσω της διεπαφής που του παρέχει το Restful API.
- Το υποσύστημα Expert Authentication Controller βρίσκεται σε επικοινωνία τόσο με το GUI καθώς ένας χρήστης του συστήματος μπορεί να αιτηθεί είσοδο, όσο και με το External Authentication Manager καθώς ο αιτών για είσοδο μπορεί να είναι νέος χρήστης απαιτώντας ταυτοποίηση ειδικότητας. Σε κάθε περίπτωση είναι αναγκαία η επικοινωνία με την Database για εξακρίβωση (και καταχώρηση) στοιχείων.
- Το Collaborative Knowledge Management System βρίσκεται σε επικοινωνία με το GUI καθώς ένας πιστοποιημένος χρήστης μπορεί να επικοινωνήσει με το σύστημα αυτό μέσω του GUI Expert. Ενώ ένας μη πιστοποιημένος χρήστης μπορεί να επικοινωνήσει μέσω του GUI Guest. Επιπλέον βρίσκεται σε επικοινωνία με το Ontologies Search Platform για άντληση λέξεων-φράσεων σε λειτουργίες όπου απαιτείται φυσική γλώσσα. Επιπρόσθετα

και μόνο στη περίπτωση συνδεδεμένου εμπειρογνώμονα επικοινωνεί και με το External Entity Provider για άντληση των οντοτήτων που ενδεχομένως να διαχειρίζονταν στην πλατφόρμα από την οποία προέρχεται και η οποία χρησιμοποιήθηκε για την ταυτοποίηση του. Η επικοινωνία του Collaborative Knowledge Management με το Core Knowledge System έχει στόχο την ανταλλαγή όλων των απαραίτητων δεδομένων κατά τη διαδικασία της εξαγωγής αποφάσεων. Τέλος βρίσκεται σε επικοινωνία με τη Database καθώς απαιτείται η πρόσβαση στα δεδομένα που πρόκειται να διαχειριστεί.

- Το Core Knowledge System επικοινωνεί αυτόματα με τη Database, για επαναφορά γνώσης στη βάση γνώσης του συστήματος. Δηλαδή την άντληση μόνο των εκτελέσιμων κανόνων από τη βάση δεδομένων. Η επικοινωνία ενεργοποιείται αυτόματα κάθε φορά που το σύστημα από offline μεταβαίνει σε online.
- Το σύστημα Ontologies Management επικοινωνεί με την Ontologies Search Platform με στόχο τον εμπλουτισμό των λέξεων-φράσεων των οντολογιών που παρέχονται από την πλατφόρμα αυτή.

4. Λεπτομερής Περιγραφή του Συστήματος και της Υλοποίησης του

Το σύστημα COSMOS είναι υλοποιημένο χρησιμοποιώντας δύο βασικές τεχνολογίες, την Java (έκδοση 8) και την SWI Prolog (έκδοση 7.4.2) όπου και οι δύο αυτές τεχνολογίες εφαρμόστηκαν με τέτοιο τρόπο ώστε να μοιράσουν το σύστημα σε δύο κύρια μέρη, παρέχοντάς του modular δυνατότητες καθώς χρησιμοποιούν διαφορετικού τύπου server σε διαφορετικά port.

Πιο συγκεκριμένα, στην πλευρά της Java οι κλάσεις και οι συναρτήσεις δημιουργήθηκαν με χρήση του Spark micro framework και είναι προσβάσιμες μέσω του Jetty Web Server. Ενώ στην πλευρά της Prolog, τα κατηγορήματα και οι μέθοδοι που δημιουργήθηκαν με τη χρήση βιβλιοθηκών της SWI Prolog είναι προσβάσιμα μέσω του SWI server.

Ομαδοποιώντας λοιπόν τα προαναφερθέντα υποσυστήματα (κεφάλαιο 3), με βάση τη τεχνολογία που υλοποιήθηκαν και τον server στον οποίο τρέχουν, χωρίζονται ως εξής:

- JAVA 8 (Jetty Server)
  - Expert Authentication Controller
  - GUI
  - Collaborative Knowledge Management System
  - Ontologies Management
  - Restful API
- SWI PROLOG 7.4.2 (SWI Server)
  - Core Knowledge System

Η Java εξαετίας του υψηλού επιπέδου και των ποιοτικών βιβλιοθηκών της παρέχει στο σύστημα σταθερότητα και επεκτασιμότητα. Για το λόγο αυτό επιλέχθηκε για τη διαχείριση βάσεων δεδομένων και επικοινωνίας μεταξύ εσωτερικών υποσυστημάτων του Cosmos και εξωτερικών συστημάτων όπως φαίνονται στην αρχιτεκτονική του συστήματός μας.

Από την άλλη πλευρά, η SWI Prolog είναι μια γλώσσα ανάπτυξης εφαρμογών τεχνητής νοημοσύνης με ενσωματωμένο μηχανισμό συλλογισμών, τον οποίο αξιοποιήσαμε για να υλοποιήσουμε τη διαδικασία συλλογισμών του συστήματός μας. Επίσης η βάση γνώσης του συστήματος έχει υλοποιηθεί σε SWI Prolog.

4.1. Γραφική Διεπαφή

Η «Γραφική Διεπαφή» (Graphical User Interface) του συστήματος απευθύνεται σε όλους τους χρήστες όπου μέσω ενός ηλεκτρονικού υπολογιστή ή tablet συνδεδεμένο στο διαδίκτυο επιθυμούν να έχουν τη μέγιστη δυνατή ποιότητα των υπηρεσιών που προσφέρει το σύστημα. Επειδή η παρουσίαση των λειτουργιών του συστήματος απαιτεί αρκετά μεγάλο αριθμό pixel, καλό θα ήταν η ανάλυση της συσκευής ή η οθόνη του υπολογιστή να είναι κατ' ελάχιστον High Definition ready (HD) δηλαδή 1280 * 720 pixels. Ιδανικά προτείνεται Full High Definition (FHD) δηλαδή 1920 * 1080 pixels.

Αν και το σύστημα ενσωματώνει καθολικά τον διαχωρισμό μεταξύ client side και server side, το GUI έχει υλοποιηθεί χρησιμοποιώντας τον Apache Free Marker template engine, δίνοντας έτσι μία νέα διάσταση μεταξύ της Java και του παραδοσιακού HTML5 κώδικα. Επίσης το GUI παρουσιάζει αρκετά δυναμικά χαρακτηριστικά, καθώς έχει χρησιμοποιηθεί το μεγαλύτερο μέρος της βιβλιοθήκης JQUERY UI [11]. Επιπλέον χρησιμοποιήθηκε η βιβλιοθήκη CHARTJS [12] για κατασκευή και αλληλεπίδραση με διαγράμματα αλλά και το CYTOSCAPE JS [13] για κατασκευή και αλληλεπίδραση με γράφους.

Ο αριθμός των γραφικών διεπαφών που παρέχονται στον χρήστη σχετίζεται άμεσα με το επίπεδο πιστοποίησής του. Για το λόγο αυτό θα χωρίσουμε το «GUI» σε δύο επιμέρους τμήματα: α) στο «***Γραφική Διεπαφή Εμπειρογνώμονα***» (GUI Expert) και β) «***Γραφική Διεπαφή μη Ταυτοποιημένου Χρήστη***» (GUI Guest).

4.1.1. Γραφική Διεπαφή Εμπειρογνώμονα

Η «Γραφική Διεπαφή Εμπειρογνώμονα» (GUI Expert) αποτελεί το σύνολο διεπαφών του ταυτοποιημένου χρήστη (εμπειρογνώμονα), στον οποίο παρέχεται η μέγιστη πρόσβαση στις λειτουργίες και γραφικές διεπαφές του συστήματος αντίστοιχα.

4.1.2. Γραφική Διεπαφή Μη Ταυτοποιημένου Χρήστη

Η «Γραφική Διεπαφή μη Ταυτοποιημένου Χρήστη» (GUI Guest) αποτελεί το σύνολο γραφικών διεπαφών του μη συνδεδεμένου χρήστη, στον οποίο παρέχεται η ελάχιστη πρόσβαση στις λειτουργίες και γραφικές διεπαφές του συστήματος αντίστοιχα.

4.2. Ελεγκτής Ταυτότητας Εμπειρογνώμονα

Ο «Ελεγκτής Ταυτότητας Εμπειρογνώμονα» (Expert Authentication Controller) αποτελεί το τμήμα του συστήματος το οποίο είναι υπεύθυνο για α) την ταυτοποίηση ενός νέου εμπειρογνώμονα, β) την εξουσιοδότηση του ήδη καταχωρημένου εμπειρογνώμονα, γ) την απόρριψη εισόδου σε μη πιστοποιημένους χρήστες αλλά και δ) την απόρριψη εισόδου σε περίπτωση ήδη συνδεδεμένου εμπειρογνώμονα.

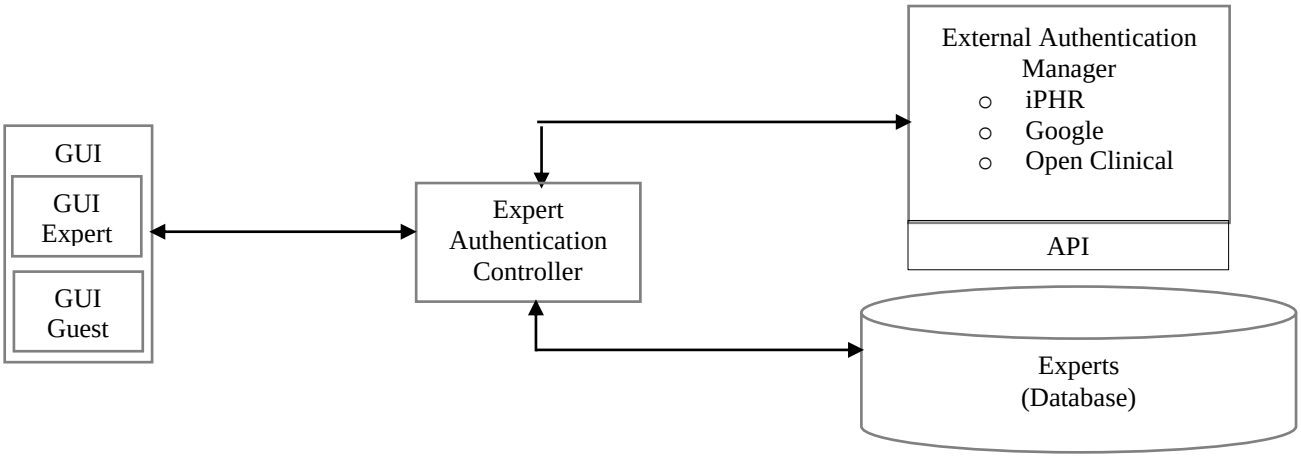

Figure 2 - Διαγραμματική Αναπαράσταση Υποσυστήματος Expert Authentication Controller

Αρχικά ο χρήστης επιλέγει τη πλατφόρμα μέσω της οποία επιθυμεί να πραγματοποιήσει είσοδο στο σύστημα, δηλαδή να ταυτοποιηθεί. Στη συνέχεια πατώντας το κουμπί της ολοκλήρωσης, τα στοιχεία αυτά στέλνονται στο τμήμα Expert Authentication Controller. Στο τμήμα αυτό πραγματοποιείται επικοινωνία με την βάση δεδομένων (εμπειρογνωμόνων) του συστήματος ελέγχοντας αν υπάρχει καταχωρημένος εμπειρογνώμονας με τα στοιχεία αυτά και αν είναι ήδη συνδεδεμένος. Η είσοδος θα του επιτραπεί μόνο στην περίπτωση που είναι ήδη καταχωρημένος αλλά δεν είναι ήδη συνδεδεμένος στο σύστημα. Αλλιώς εάν δεν εντοπίσει τον εμπειρογνώμονα στην βάση δεδομένων τότε στέλνει αίτημα ταυτοποίησης χρήστη στον External Authentication Manager ο οποίος με την σειρά του εάν τον αναγνωρίσει ή όχι θα επιστρέψει ανάλογο μήνυμα σε μορφή JSON. Στην περίπτωση λοιπόν που επιστραφεί μήνυμα επιτυχούς ταυτοποίησης τότε γίνεται καταχώριση του χρήστη ως νέος εμπειρογνώμονας στην βάση δεδομένων (εμπειρογνωμόνων) του συστήματος και τον εξουσιοδοτεί άμεσα επιτρέποντάς του είσοδο. Αλλιώς απορρίπτει το αίτημά του για είσοδο. Σε κάθε περίπτωση ο χρήστης ενημερώνεται μέσω του GUI για το όποιο συμπέρασμα κατέληξε το τμήμα Expert Authentication Controller.

4.3. Διαχειριστής Συνεργατικής Γνώσης

Το τμήμα «Διαχειριστής Συνεργατικής Γνώσης» (Collaborative Knowledge Management System «CKMS») είναι υπεύθυνο για τη διαχείριση των *κανόνων*, *εμπειρογνωμόνων* αλλά και των *οντοτήτων* με στόχο τόσο την ποσοτική όσο και την ποιοτική βελτίωση της γνώσης του συστήματος.

Ομαδοποιεί τρία κύρια υποσυστήματα καθ' ένα εκ' των οποίων διαχειρίζεται διαφορετικού τύπου λειτουργίες συστήματος. Τα τρία κύρια αυτά υποσυστήματα είναι τα εξής: α) «***Διαχειριστής Κανόνων***» (Rule Management), β) «***Διαχειριστής Συνεργασίας Εμπειρογνωμόνων***» (Collaboration Management) και γ) «***Διαχειριστής Οντοτήτων και Αποφάσεων***» (Entity & Decision Management).

4.3.1. Διαχειριστής Κανόνων

Ο «***Κανόνας***» (Rule) για το σύστημα είναι ένα σύνολο από συνθήκες λογικά ή/και χρονικά συνδεδεμένες μεταξύ τους με στόχο την ικανοποίηση ενός αποτελέσματος. Η αναπαράσταση γνώσης που χρησιμοποιείται στους κανόνες είναι σε «if-then» μορφή.

- Κάθε συνθήκη ή συμπέρασμα είναι μία λογική έκφραση στην οποία έχει καταχωρηθεί κάποιος παράγοντας βεβαιότητας/αλήθειας. Επίσης σε κάθε συνθήκη μπορεί να οριστεί χρονική διάρκεια ύπαρξης της. Οι συνθήκες στους κανόνες έχουν τη εξής μορφή: ***(Λογική-έκφραση, Παράγοντας-βεβαιότητας, Λίστα-Χρόνου)*** όπου η ***Λογική-έκφραση*** έχει τη μορφή ***(Όρος, Σχεσιακός-τελεστής, Τιμή)*** όπου ο «**Όρος**» μπορεί να είναι μεταβλητή, σταθερά ή κάποιος άλλος όρος***, ο Παράγοντας-βεβαιότητας*** είναι μια αριθμητική τιμή στο διάστημα [1,100] και η ***Λίστα-Χρόνου*** έχει τη μορφή ***[Χρόνος1,..,ΧρόνοςΚ]*** όπου ***Χρόνος1,..,ΧρόνοςΚ*** είναι χρονικές εκφράσεις όπως χθες, σήμερα, 3 ημέρες, κτλ. Για παράδειγμα, η αναπαράσταση της πρότασης, «Ο Χ έχει πυρετό για 3 ημέρες, ΠΒ=60», παριστάνεται ως εξής: ((πυρετός, =, 'true'), 60, ['3 ημέρες']). Περισσότεροι σύνθετοι κανόνες υπάρχουν στο παράρτημα «Παραδείγματα κανόνων από τη ΒΓ». . Ο παράγοντας βεβαιότητας μιας λογικής έκφρασης είτε καθορίστηκε αρχικά από τον εμπειρογνώμονα ή υπολογίστηκε μέσω της Θεωρίας βεβαιότητας.
- Ο όρος ***λογική διασύνδεση*** αναφέρεται ως η σύνδεση των συνθηκών (condition1,..,conditionN) με συνδυασμούς των λογικών τελεστών "and" ("Λ") και "or" ("V"). Η αναπαράσταση μιας λογικής διασύνδεσης έχει την εξής μορφή. ***(Λογική-έκφραση, Παράγοντας-βεβαιότητας, Λίστα-Χρόνου) and/or … and/or (Λογική-έκφραση, Παράγοντας-βεβαιότητας, Λίστα-Χρόνου).*** Ένα παράδειγμα λογικής διασύνδεσης και η αναπαράσταση της ακολουθούν. «Ο Χ είχε αιμόπτυση χθες και σήμερα έχει πυρετό 39^{o}, ΠΒ=67 και ΠΒ=80 αντίστοιχα». Παριστάνεται για τον Χ ως εξής: ((αιμόπτυση, =, 'true'), 67, ['χθες']) **and** ((θερμοκρασία σώματος, =, '39^{o}'), 80, ['σήμερα']). Ένα άλλο πιο σύνθετο παράδειγμα είναι το εξής: «Ο Χ είχε αιμόπτυση χθες και σήμερα είτε έχει πυρετό για 2 ώρες ή ήταν καλά και πήγε στη δουλειά του , ΠΒ=67, ΠΒ=78 και ΠΒ=80 αντίστοιχα. ((αιμόπτυση, =, 'true'), 67, ['χθες']) **and** ((πυρετός, =, 'true'), 80, ['σήμερα για 2 ώρες']). Περισσότερα παραδείγματα λογικών διασυνδέσεων υπάρχουν στο παράρτημα «Παραδείγματα κανόνων από τη ΒΓ».

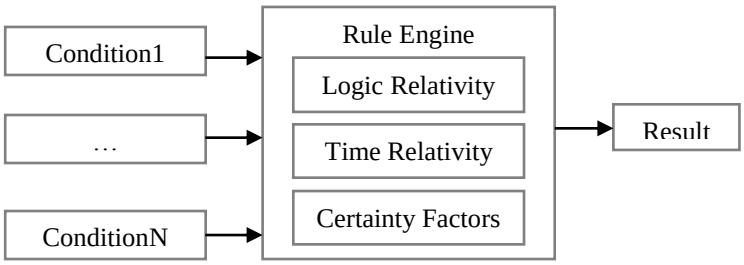

Figure 3 - Διαγραμματική Αναπαράσταση Κανόνα

Ο «***Παράγοντας Βεβαιότητας***» (Certainty Factor «CF») εκφράζει το βαθμό βεβαιότητας (πιστότητας και απιστίας) για την αλήθεια κάθε συνθήκης (Condition) της προϋπόθεσης του κανόνα και του συμπεράσματος του κανόνα όταν ισχύει η προϋπόθεση του. Η θεωρία βεβαιότητας στηρίζεται στη χρήση παραγόντων βεβαιότητας. Βάση της θεωρίας βεβαιότητας ο κάθε ισχυρισμός (γεγονός, κανόνας) έχει ένα παράγοντα βεβαιότητας ο οποίος είναι ένα πραγματικός αριθμός στο διάστημα [-1,1]. Η θεωρία βεβαιότητας υποστηρίζεται από μια μέθοδο μετάδοσης της βεβαιότητας καθώς οι κανόνες πυροδοτούνται στη φάση των συλλογισμών. Για το πεδίο τιμών [-1,1], θεωρείται το αριστερό άκρο ως η περίπτωση μέγιστης απιστίας ενώ το δεξιό άκρο θεωρείται ως η περίπτωση μέγιστης εμπιστοσύνης.

Ο παράγοντας βεβαιότητας στο σύστημά μας ορίζεται στο διάστημα [1,100], όπου το αριστερό άκρο (ένα) είναι η περίπτωση ελάχιστης βεβαιότητας/αληθείας της συνθήκης, ενώ το δεξί άκρο (εκατό) είναι η περίπτωση της μέγιστης βεβαιότητας/αληθείας της συνθήκης. Δηλαδή για ένα παράγοντα βεβαιότητας ισχύει ότι, η μέγιστη τιμή αβεβαιότητας δεν πρέπει να είναι μεγαλύτερη ή ίση από τη μέγιστη τιμή βεβαιότητας, ή ότι η ελάχιστη τιμή βεβαιότητας δεν πρέπει να είναι μηδέν (0%).

Με τον τρόπο αυτό το σύστημα εξασφαλίζει ότι, κάθε ισχυρισμός ο οποίος χαρακτηρίζεται από ένα παράγοντα βεβαιότητας έχει τουλάχιστον 1% βαθμό αλήθειας.

Το πεδίο τιμών δεν ορίζεται στο [0.1, 1] αλλά ορίζεται στο [1,100] καθώς ο ποσοστιαίος ορισμός βεβαιότητας κρίθηκε περισσότερο φιλικός προς τον χρήστη.

- «***Condition Certainty Factor (Condition CF)***»: Είναι ο παράγοντας βεβαιότητας της συνθήκης ενός κανόνα και ορίζεται από τον εμπειρογνώμονα κατά τη διαδικασία ενημέρωσης ή δημιουργίας κανόνα. Ο παράγοντας βεβαιότητας της συνθήκης διατηρείται ο ίδιος και στη διαδικασία συλλογισμών. Έτσι ευνοείται ο παράγοντας βεβαιότητας συνθήκης του κύριου κανόνα σε περίπτωση Pipeline.
- «***Evidence Certainty Factor (Evidence CF)***»: Είναι ο παράγοντας βεβαιότητας της προϋπόθεσης ενός κανόνα και ορίζεται αυτόματα από το σύστημα. Αποτελεί min και max συνδυασμό των Condition CF, ανάλογα με τον λογικό τελεστή που συνδέονται οι συνθήκες. Εάν συνδέονται με "and" τότε υπολογίζει το min των παραγόντων βεβαιότητας. Εάν συνδέονται με "or" τότε υπολογίζει το max των παραγόντων βεβαιότητας.

π.χ. Εάν P1 and (P2 or P3) Τότε Evidence CF = min(P1CF, max(P2CF,P3CF)).
{P| P = παράγοντας βεβαιότητας συνθήκης}

- ***«Rule Certainty Factor (Rule CF)***»: Είναι ο παράγοντας βεβαιότητας (ολόκληρου) του κανόνα και ορίζεται από τον εμπειρογνώμονα που δημιουργεί τον κανόνα.
- «***Result Certainty Factor (Result CF)***»: Είναι ο παράγοντας βεβαιότητας του αποτελέσματος ενός κανόνα, το οποίο ορίζεται από το σύστημα μέσω του παρακάτω τύπου:

Παράγοντας βεβαιότητας προϋπόθεσης κανόνα(CF Evidence) ***** Παράγοντας βεβαιότητας κανόνα (CF Rule)
= Παράγοντας βεβαιότητας Αποτελέσματος (CF Result).

Ο «***Εντοπισμός Σύγκρουσης Κανόνα***» (Rule Conflict Detection «RCD»), είναι ένας μηχανισμός υλοποιημένος σε java ο οποίος χρησιμοποιεί πίνακες αλήθειας [14] για τους λογικούς τελεστές and, or, με στόχο τη μετατροπή του κανόνα σε κανονική διαζευκτική μορφή (Disjunctive Normal Form DNF) [15].

p	*q*	*p* ∧ *q*
T	T	T
T	F	F
F	T	F
F	F	F

Figure 4 - Πίνακας αληθείας λογικού τελεστή "and", με p και q να είναι οι προϋποθέσεων του κανόνα.

p	*q*	*p* ∨ *q*
T	T	T
T	F	T
F	T	T
F	F	F

Figure 5 - Πίνακας αλήθειας λογικού τελεστή "or", με p και q να είναι οι συνθήκες του κανόνα.

Η διαζευκτική κανονική μορφή δύο η περισσοτέρων συνθηκών ορίζεται ως η διάζευξη των συζεύξεων των συνθηκών αυτών.

π.χ. $DNF(X_1, \ldots, X_n) = (X_1 \wedge X_2) \vee (X_3 \wedge X_4 \wedge X_5) \vee \ldots \vee X_n$.
{X| X = Συνθήκη}

Σκοπός χρήσης του RCD είναι:

- Εντοπίζει συντακτικά λάθη στη προϋπόθεση ενός κανόνα. Συντακτικό λάθος σε ένα κανόνα θεωρείται όταν συνθήκες της προϋπόθεσης του κανόνα δεν είναι λογικά συνδεδεμένες μεταξύ τους. Εάν λοιπόν κατά τη διαδικασία μετατροπής της προϋπόθεσης του κανόνα σε DNF μορφή, εμφανιστεί κάποιο error, τότε ο μηχανισμός επιστρέφει μήνυμα εντοπισμού συντακτικού λάθους.
- Εντοπίζει συγκρούσεις μεταξύ δύο ή περισσοτέρων κανόνων. Σύγκρουση θεωρείται όταν τουλάχιστον δύο κανόνες έχουν το ίδιο αποτέλεσμα, το ίδιο σύνολο (ή υποσύνολο) DNF και οι συνθήκες από τις οποίες παράγεται το DNF να είναι ίδιες ή αντίθετες μεταξύ τους.

  Για τον εντοπισμό της σχέσης μεταξύ των προϋποθέσεων δύο κανόνων χρησιμοποιούνται δύο τεχνικές:
  - «Exact Match»: το σύστημα εντοπίζει εάν δύο συνθήκες είναι ακριβώς ίδιες ή ακριβώς αντίθετες. Για παράδειγμα οι συνθήκες «θερμοκρασία σώματος > 37 Celsius» και «θερμοκρασία σώματος =< 37 Celsius» είναι ακριβώς αντίθετες, ενώ οι συνθήκες «θερμοκρασία σώματος == 36.6 Celsius» και «θερμοκρασία σώματος == 36,6 Celsius» είναι ακριβώς ίδιες.
  - «Elastic Numeric Similarity»: το σύστημα εντοπίζει εάν είναι ***κοντά*** δύο μαθηματικές εκφράσεις (συνθήκες προϋποθέσεων) μεταξύ τους. Ο μαθηματικός τύπος που χρησιμοποιήθηκε για το σκοπό αυτό αποτελεί συνδυασμό του τύπου απόστασης μεταξύ δύο σημείων (point distance) με τον τύπο της κανονικοποίησης (normalize). Η κανονικοποίηση χρειάζεται για την παραγωγή αποτελέσματος «*Numeric Similarity*» στο διάστημα [0,1]:

    Numeric Similarity = round (abs (ConditionValue1 – ConditionValue2) / (ConditionValue1 + ConditionValue2))

    Εάν Numeric Similarity = 0, τότε 100% match. Αλλιώς Εάν Numeric Similarity = 1.0 τότε 100% different. Αλλιώς Εάν Numeric Similarity =< 0.10 τότε είναι κοντά.

    Για παράδειγμα η συνθήκη «θερμοκρασίας σώματος > 37 Celsius» είναι κοντά στην συνθήκη «θερμοκρασία σώματος > 36.95 Celsius» καθώς η απόσταση της τιμής «37 Celsius» με την τιμή «36.95 Celsius» είναι μικρότερη από (Numeric Similarity) 0.10.

  Υπάρχουν δύο βαθμίδες συγκρούσεων, «Soft Conflict» και «Hard Conflict»:
  - ***Soft Conflict***: θεωρείται σύγκρουση όπου δύο κανόνες έχουν ίδιο αποτέλεσμα και ακριβώς ίδιο σύνολο (ή υποσύνολο) DNF. Έστω ότι έχουμε δύο κανόνες, τους K1, K2. Οι κανόνες αυτοί έχουν ίδιο συμπέρασμα και τις παρακάτω συνθήκες.
    K1: Συνθήκη_X or Συνθήκη_Y.
    K2: Συνθήκη_X or (Συνθήκη_Y and Συνθήκη_Z).
  - Τότε οι κανόνες K1, K2 έρχονται σε Soft Conflict. Αφού οι δύο αυτοί κανόνες έχουν κοινό υποσύνολο DNF την συνθήκη_X, η οποία αρκεί για να τους πυροδοτήσει.***Hard Conflict***: θεωρείται σύγκρουση όπου δύο κανόνες έχουν ίδιο αποτέλεσμα και ακριβώς αντίθετο σύνολο (ή υποσύνολο) DNF. Έστω ότι έχουμε δύο κανόνες, τους K1, K2. Οι κανόνες αυτοί έχουν ίδιο συμπέρασμα και τις παρακάτω συνθήκες.
    K1: Συνθήκη_X or Συνθήκη_Y.
    K2: not(Συνθήκη_X) or (Συνθήκη_Y and Συνθήκη_Z).
    Τότε οι κανόνες K1 και K2 έρχονται σε Hard Conflict. Εφόσον και οι δύο αυτοί κανόνες έχουν αντίθετο υποσύνολο DNF την Συνθήκη_X και not(Συνθήκη_X).

  Το Soft Conflict αν και δεν είναι επιβλαβές για το σύστημα καθώς προβλέπεται από τη Θεωρία βεβαιότητας, δηλώνει καταχώρηση ήδη υπάρχουσας γνώσης. Για το λόγο αυτό το σύστημα επιτρέπει την δημιουργία κανόνα που βρίσκεται σε Soft Conflict, όμως με «Warning Message». Από την άλλη πλευρά το Hard Conflict δεν είναι διαχειρίσιμο μέσω της θεωρίας βεβαιότητας καθώς φέρνει σε σύγκρουση τη γνώση του συστήματος (αντίφαση). Για το λόγο αυτό το σύστημα δεν επιτρέπει την καταχώρηση κανόνα που βρίσκεται σε Hard Conflict, εμφανίζοντας «Danger Message».

Το σύστημα «Διαχειριστής Κανόνων» (Rule Management) αποτελείται από πέντε κύριες λειτουργίες για τον χρήστη τύπου εμπειρογνώμονα, εκ' των οποίων οι τέσσερεις πρώτες αποτελούν το CRUD system [16].

- **Create Rule**

Η λειτουργία «Create Rule» επιτρέπει την δημιουργία κανόνα και αποθήκευσή του στη βάση δεδομένων. Για το σκοπό αυτό περιλαμβάνει τις εξής επικοινωνίες:

1. Επικοινωνία με το Ontologies Search Platform για τον ορισμό προϋποθέσεων, αποτελέσματος και βάσης γνώσης του κανόνα.
2. Επικοινωνία με βάση δεδομένων για έλεγχο συγκρούσεων μεταξύ κανόνων μέσω του RCD.
3. Επικοινωνία με τη βάση δεδομένων για καταχώριση (Insert) κανόνα.

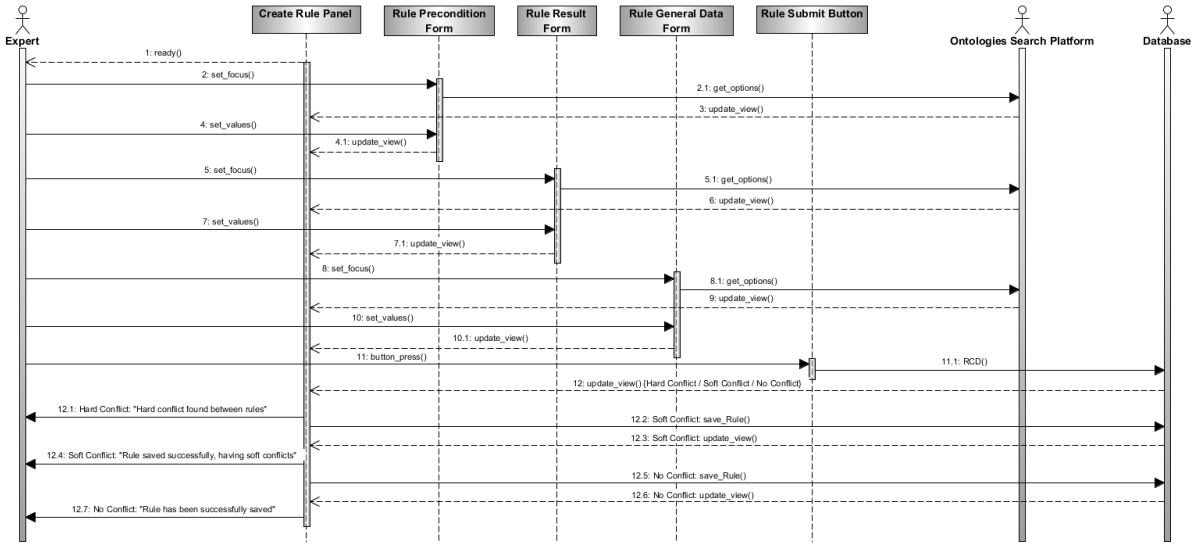

Figure 6 Δημιουργία κανόνα - Διάγραμμα ροής

Στην εικόνα 6, παρουσιάζονται υπό μορφή διαγράμματος ροής τα βήματα που απαιτούνται για την δημιουργία και καταχώρηση κανόνα. Αρχικά ο εμπειρογνώμονας επικοινωνεί μέσω της γραφικής διεπαφής με το «*Create Rule Panel*» για να συμπληρώσει τη φόρμα των προϋποθέσεων «*Rule Precondition Form*», τη φόρμα του αποτελέσματος «*Rule Result Form*» και επιπλέον τη φόρμα με τα χαρακτηριστικά κανόνα «*Rule General Data*» (π.χ. βάση γνώσης κανόνα, περιγραφή κανόνα, έκδοση κανόνα, κτλ.). Παράλληλα τα πεδία της εστιασμένης φόρμας, ενημερώνονται με προτεινόμενες λέξεις-φράσεις από τη πλατφόρμα οντολογιών «*Ontologies Search Platform*». Ο εμπειρογνώμονας επιλέγει καταχώρηση κανόνα, πατώντας το κουμπί «*Rule Submit Button*». Η ενέργεια αυτή πυροδοτεί την μέθοδο «*RCD()*» η οποία ελέγχει για πιθανή σύγκρουση του τρέχον κανόνα με άλλους κανόνες της βάσης δεδομένων. Η μέθοδος αυτή επιστρέφει το κατάλληλο μήνυμα (π.χ. "*Rule saved successfully, having soft conflicts*") προς τον εμπειρογνώμονα, εκτελώντας αντίστοιχα την κατάλληλη ενέργεια (π.χ. καταχώρηση κανόνα στη βάση δεδομένων «*Save_Rule()*»).

- **Read Rule**

Η λειτουργία «Read Rule» επιτρέπει την προβολή κανόνων για την οποία απαιτείται άντληση αυτών από τη βάση δεδομένων. Υπάρχουν δύο επίπεδα προβολής κανόνων:

1. ***Private Rules***: προβολή όλων των κανόνων που ο εμπειρογνώμονας είναι editor ή owner.
2. ***Public Rules***: προβολή όλων των κανόνων που ο εμπειρογνώμονας είναι editor ή owner ή τίποτα από τα δύο.

- **Update Rule**

Η λειτουργία «Update Rule» επιτρέπει την επεξεργασία κάποιου ήδη καταχωρημένου κανόνα της βάσης δεδομένων και την ενημέρωση κατάστασης του κανόνα αυτού.

Η επεξεργασία κάποιου ήδη καταχωρημένου κανόνα μπορεί να γίνει με δύο τρόπους:

1. ***Private Update***: ο εμπειρογνώμονας ενημερώνει μόνος του τον κανόνα στον οποίο είναι owner ή editor.
2. Collaborative Update: ο εμπειρογνώμονας μαζί με ένα σύνολο από συνεργάτες διαφορετικών ή ίδιων ειδικοτήτων ενημερώνουν τον κανόνα στον οποίο συνεργάζονται. Η ενημέρωση αυτή μπορεί να γίνει σε πραγματικό ή μη πραγματικό χρόνο.

Η επεξεργασία κάποιου ήδη καταχωρημένου κανόνα απαιτεί τις παρακάτω επικοινωνίες:

1. Επικοινωνία με τη βάση δεδομένων για άντληση των στοιχείων του κανόνα προς επεξεργασία.
2. Επικοινωνία με το Ontologies Search Platform για τον ορισμό προϋποθέσεων, αποτελέσματος και βάσης γνώσης του κανόνα.
3. Επικοινωνία με βάση δεδομένων για έλεγχο συγκρούσεων μεταξύ κανόνων μέσω του μηχανισμού RCD.
4. Επικοινωνία με τη βάση δεδομένων για ενημέρωση (Update) κανόνα.

Η ενημέρωση κατάστασης ενός ήδη καταχωρημένου κανόνα, αποτελεί την επιλογή ενός εκ' των δύο διαφορετικών καταστάσεων:

- ✓ ***Draft:*** η κατάσταση στην οποία ο κανόνας καθίσταται μη εκτελέσιμος (προσχέδιο).
- ✓ ***Executable:*** η κατάσταση στην οποία ο κανόνας καθίσταται εκτελέσιμος.

Είναι στην κρίση του εμπειρογνώμονα πότε θα αλλάξει ένα κανόνα από draft σε executable ή και το αντίστροφο. Σε κάθε περίπτωση η εναλλαγή αυτή επηρεάζει τη βάση γνώσης του συστήματος. Όταν ένας κανόνας είναι σε κατάσταση draft, τότε διαγράφεται (εάν υπάρχει ήδη) από τη βάση γνώσης. Όταν ένας κανόνας είναι σε κατάσταση executable, τότε αποθηκεύεται (εάν δεν υπάρχει ήδη) στη βάση γνώσης.

- **Delete Rule**

Η λειτουργία «Delete Rule» επιτρέπει την οριστική διαγραφή κάποιου κανόνα από τη βάση δεδομένων του συστήματος. Εάν ο κανόνας αυτός βρίσκονταν σε κατάσταση τύπου executable, τότε ο κανόνας διαγράφεται και από την βάση γνώσης του συστήματος.

- **Clone Rule**

Η λειτουργία «Clone Rule» επιτρέπει την αντιγραφή ενός κανόνα κάποιου εμπειρογνώμονα. Ο νέος κανόνας που θα παραχθεί, καταχωρείται στην βάση δεδομένων του συστήματος ως κανόνας τύπου κλώνος. Οι κανόνες τύπου κλώνος αποκτούν μία επιπλέον λειτουργία που ονομάζεται «***Commit Rule***». Η λειτουργία αυτή επιτρέπει στον κανόνα τύπου κλώνο να προτείνει την τρέχουσα έκδοσή του ως βελτιωτική έκδοση στον κανόνα από τον οποίο αντιγράφηκε. Η λειτουργία αυτή λοιπόν παρέχει έμμεση συνεργασία μεταξύ εμπειρογνωμόνων. Δηλαδή οι εμπειρογνώμονες μπορούν να προτείνουν αλλαγές σε ίδιο κανόνα, χωρίς να απαιτείται να γίνουν collaborators.

4.3.2. Διαχειριστής Συνεργασίας Εμπειρογνωμόνων

Ο «***Ιδιοκτήτης κανόνα***» (Rule Owner) ονομάζεται ο εμπειρογνώμονας που έχει στην προσωπική του βάση γνώσης, τουλάχιστον ένα κανόνα που ο ίδιος δημιούργησε μέσω της λειτουργίας Create Rule. Οι διαθέσιμες ενέργειες του owner πάνω σε ένα κανόνα είναι:

- Private Update.
- Delete Rule.
- Invite an Editor.
- Remove an Editor.
- Forum.
- Collaborative Update.

Ο «***Συντάκτης κανόνα***» (Rule Editor) ονομάζεται ο εμπειρογνώμονας που έχει στην προσωπική του βάση γνώσης, τουλάχιστον ένα κανόνα στον οποίο κάποιος άλλος εμπειρογνώμονας είναι owner. Οι ενέργειες του editor πάνω σε ένα κανόνα είναι:

- Private Update.
- Leave Rule.
- Forum.
- Collaborative Update.

«***Συνεργάτες***» (Collaborators) χαρακτηρίζονται δύο ή περισσότεροι εμπειρογνώμονες που βρίσκονται στον ίδιο κανόνα ως editor ή owner. Δύο εμπειρογνώμονες μπορούν να γίνουν συνεργάτες με δύο τρόπους. Είτε ο ενδιαφερόμενος εμπειρογνώμονας στέλνει Join Request στον κανόνα, είτε ο κανόνας (μέσω του Owner) στέλνει Invite Request στον εμπειρογνώμονα.

Το «***Expert Knowledge Domain***» αποτελεί το πεδίο ορισμού των γνώσεων του εμπειρογνώμονα.

Το «***Expert Root Knowledge Domain***» αποτελεί το πεδίο ορισμού του Expert Knowledge Domain, δηλαδή το ευρύτερο πεδίο ορισμού στο οποίο ανήκει το πεδίο ορισμού εμπειρογνώμονα. Εάν για παράδειγμα το Expert Knowledge Domain είναι το Oncology, τότε το Expert Root Knowledge Domain είναι το Medicine.

Το σύστημα «Διαχειριστής Συνεργασίας Εμπειρογνωμόνων» (Collaboration Management) ομαδοποιεί όλες εκείνες τις λειτουργίες οι οποίες περιβάλλονται γύρο από τον εμπειρογνώμονα. Οι λειτουργίες αυτές είναι οι εξής:

- Editor Invite Request

Η λειτουργία «Editor Invite Request» επιτρέπει σε ένα εμπειρογνώμονα ο οποίος είναι owner σε ένα κανόνα, να καλέσει κάποιον άλλο εμπειρογνώμονα ως συνεργάτη (τύπου editor) στον κανόνα αυτό. Βασική προϋπόθεση συνεργασίας είναι να έχουν και οι δύο εμπειρογνώμονες το ίδιο Root Knowledge Domain. Η λειτουργία αυτή απαιτεί ενημέρωση της βάση δεδομένων του συστήματος.

- Editor Join Request

Η λειτουργία «Join Request» επιτρέπει σε ένα εμπειρογνώμονα να αιτηθεί editor σε κάποιο κανόνα. Βασική προϋπόθεση της λειτουργίας αυτής είναι ο αιτών εμπειρογνώμονας και ο κανόνας να έχουν το ίδιο Root Knowledge Domain. Η λειτουργία αυτή απαιτεί ενημέρωση της βάση δεδομένων του συστήματος.

- Leave Collaboration

Η λειτουργία «Leave Collaboration» επιτρέπει σε ένα editor κάποιου κανόνα να διαγράψει τον κανόνα από την προσωπική του βάση γνώσης, παύοντας παράλληλα να είναι συνεργάτης με τους υπόλοιπους εμπειρογνώμονες στον κανόνα αυτό. Η λειτουργία αυτή απαιτεί ενημέρωση της βάση δεδομένων του συστήματος.

- Remove Collaborator

Η λειτουργία «Remove Collaborator» επιτρέπει σε ένα owner κάποιου κανόνα να διαγράψει κάποιον συνεργάτη του από τον κανόνα αυτό. Η λειτουργία αυτή απαιτεί ενημέρωση της βάση δεδομένων του συστήματος.

- NC-Mail & C-MAIL

Η λειτουργία «NC-Mail & C-Mail» επιτρέπει στους εμπειρογνώμονες που έχουν ίδιο Root Knowledge Domain να επικοινωνήσουν με γραπτό λόγο σε πραγματικό χρόνο. Αποτελείται από δύο μηχανισμούς:

  - NC-Mail (Non Collaborative-Mail): είναι ο μηχανισμός ανταλλαγής μνημάτων μεταξύ μη συνεργαζόμενων εμπειρογνωμόνων.
  - C-Mail (Collaborative-Mail): είναι ο μηχανισμός ανταλλαγής μηνυμάτων μεταξύ συνεργαζόμενων εμπειρογνωμόνων.

Η λειτουργία αυτή απαιτεί ενημέρωση της βάση δεδομένων του συστήματος.

- Forum

Όταν ένας κανόνας δημιουργείται, παράλληλα ενεργοποιείται και η λειτουργία «Forum» για τον κανόνα αυτό. Η λειτουργία αυτή παρέχει στους συνεργαζόμενους εμπειρογνώμονες τις εξής δυνατότητες:

  - Δημιουργία νέας δημοσίευσης «Create Post»: Ο εμπειρογνώμονας έχει τη δυνατότητα να εκφράσει τη γνώμη του, να αναφέρει προβλήματα, να διοργανώσει event ή οτιδήποτε άλλο επιθυμεί γράφοντάς το ως δημοσίευση στο forum του κανόνα.
  - Διαγραφή Δημοσίευσης «Delete Post»: Ο εμπειρογνώμονας και ιδιοκτήτης κάποιας δημοσίευσης μπορεί να τη διαγράψει ανά πάσα στιγμή.
  - Δημοτικότητα «Popularity»: Μία δημοσίευση μπορεί να αξιολογηθεί από τους υπόλοιπους συνεργάτες, επιλέγοντας συμφωνώ ή διαφωνώ πάνω σε αυτή. Ο δημιουργός της δημοσίευσης δε επιτρέπεται να αξιολογήσει τον εαυτό του.

Η λειτουργία αυτή απαιτεί ενημέρωση της βάση δεδομένων του συστήματος.

4.3.3. Διαχειριστής Οντοτήτων και Αποφάσεων

Ως «***Οντότητα***» (Entity) ορίζεται ένα αντικείμενο με διακριτή και ανεξάρτητη ύπαρξη. Σ' αυτή τη πτυχιακή «***Οντότητα***» (Entity), είναι η αναπαράσταση μιας έννοιας ή μια αυτόνομη ύπαρξη με υλική (στον πραγματικό κόσμο) ή θεωρητική υπόσταση (συμβατική ύπαρξη) , που περιλαμβάνει

ένα σύνολο από γνωρίσματα. Η οντότητα ορίζεται στο Expert Root Knowledge Domain του εμπειρογνώμονα που τη δημιούργησε. Παραδείγματα οντοτήτων, *ασθενής, μαθητής, φορολογούμενος*, κτλ.

«***Χαρακτηριστικό ή γνώρισμα οντότητας***» (Entity Attribute) είναι μία μαθηματική ή θεωρητική έκφραση η οποία υποστηρίζει χρονικό προσδιορισμό (schedule). Για παράδειγμα, η οντότητα *ασθενής* μπορεί να έχει τα εξής χαρακτηριστικά, θερμοκρασία σώματος, καρδιακοί παλμοί, λευκά αιμοσφαίρια, κτλ., η οντότητα *μαθητής* μπορεί να έχει τα εξής χαρακτηριστικά, μέσος όρος βαθμολογίας εξαμήνου, αριθμός μητρώου, μαθήματα εξαμήνου, σύνολο περασμένων μαθημάτων, κτλ. , η οντότητα *φορολογούμενος* μπορεί να έχει τα εξής χαρακτηριστικά, ποσό παρακράτησης φόρου σύνταξης, ποσό οφειλών προς το δημόσιο, συνολικό ποσό εσόδων ανά χρόνο, σύνολο ακινήτων, κάτοχος φορολογικής δήλωσης για το 2017, κτλ.

«***Εξαγωγή Απόφασης***» (Decision Derivation) είναι η διαδικασία εκτέλεσης συλλογισμών πάνω σε μία οντότητα, κατά την οποία συνδυάζονται τα χαρακτηριστικά της οντότητας (entity attributes), με ένα σύνολο κανόνων.

Το σύστημα «Διαχειριστής Οντοτήτων και Αποφάσεων» (Entity & Decision Management) ομαδοποιεί όλες εκείνες τις λειτουργίες οι οποίες περιβάλλονται γύρο από «Entity». Συγκεκριμένα περιλαμβάνει τις εξής λειτουργίες:

- **Create/Import Entity**

  Η λειτουργία «Create/Import Entity» μπορεί να γίνει με δύο τρόπους:
  - *Εισαγωγή Οντότητας «Import Entity»:* Εάν ο εμπειρογνώμονας διαχειρίζεται κάποιου είδους οντότητες στην πλατφόρμα μέσω της οποίας ταυτοποιήθηκε, τότε το σύστημα COSMOS κάνει αυτόματο συγχρονισμό καταχωρώντας όλες τις οντότητες της εξωτερικής πλατφόρμας στη βάση δεδομένων του συστήματος.
  - *Δημιουργία οντότητας «Create Entity»:* Ο εμπειρογνώμονας μπορεί ανά πάσα στιγμή να δημιουργήσει μία νέα οντότητα. Η δημιουργία οντότητας περιλαμβάνει δήλωση ονομασίας και κατάστασης οντότητας. Επίσης απαιτεί επικοινωνία με τη βάση δεδομένων για καταχώρηση της νέας αυτής οντότητας.
- **Entity Status**

  Το κάθε entity μπορεί να χαρακτηριστεί από μία κατάσταση την οποία ορίζει ο εμπειρογνώμονας ιδιοκτήτης του αντικειμένου αυτού. Οι καταστάσεις στις οποίες μπορεί να βρεθεί μία οντότητα είναι τρείς: «Good», «Warning», «Danger». Η αλλαγή της κατάστασης απαιτεί ενημέρωση της βάσης δεδομένων του συστήματος.
- **Delete Entity**

  Η λειτουργία «Delete Entity» αναφέρεται στην οριστική διαγραφή μίας οντότητας. Μπορεί να εφαρμοστεί αποκλειστικά και μόνο στις οντότητες οι οποίες δημιουργήθηκαν μέσω της πλατφόρμας COSMOS και από τον εμπειρογνώμονα που τα δημιούργησε. Για τη διαγραφή των entities τα οποία έγιναν import από κάποια εξωτερική πλατφόρμα, μπορούν να διαγραφούν μόνο μέσω της πλατφόρμας αυτής. Η διαγραφή οντότητας απαιτεί ενημέρωση της βάση δεδομένων του συστήματος.
- **Create/Import Entity Attribute**

  Η λειτουργία «Create/Import Entity Attribute» μπορεί να γίνει με δύο τρόπους:
  - *Εισαγωγή γνωρίσματος οντότητας «Import Entity Attribute»:* Οι οντότητες που έγιναν import από εξωτερικό σύστημα ενδεχομένως να είχαν κάποια γνωρίσματα. Αυτά τα γνωρίσματα καταχωρούνται μέσω αυτής της λειτουργίας στην βάση δεδομένων του συστήματος.
  - *Δημιουργία γνωρίσματος οντότητας «Create Entity Attribute»:* Ο εμπειρογνώμονας μπορεί να δημιουργήσει χαρακτηριστικά οντοτήτων (entity attributes) τόσο σε οντότητες οι οποίες προήλθαν από import όσο και σε οντότητες τις οποίες δημιούργησε ο ίδιος. Σε κάθε περίπτωση απαιτείται ενημέρωση της βάσης δεδομένων του συστήματος.
- **Delete Entity Attribute**

  Η λειτουργία «Delete Entity Attribute» αναφέρεται στην οριστική διαγραφή γνωρίσματος οντότητας. Μπορεί να εφαρμοστεί αποκλειστικά μόνο σε γνωρίσματα τα οποία δημιουργήθηκαν μέσω της πλατφόρμας COSMOS από τον εμπειρογνώμονα ο οποίος τις δημιούργησε. Για τη διαγραφή των entity attributes τα οποία έγιναν import από κάποια εξωτερική πλατφόρμα, μπορούν να διαγραφούν μόνο μέσω της πλατφόρμας αυτής. Η διαγραφή γνωρίσματος οντότητας απαιτεί ενημέρωση της βάσης δεδομένων του συστήματος.

- **Decision Derivation for Entity**

Η λειτουργία «Decision Derivation for Entity» αποτελεί το σημείο έναρξης και αρχικοποίησης της διαδικασίας εξαγωγής αποφάσεων. Υπάρχουν δύο είδη εξαγωγής απόφασης:

  - *Ιδιωτική Εξαγωγή Αποφάσεων «Private Decision Derivation»:* ονομάζεται η εξαγωγή απόφασης που χρησιμοποιεί ως γνώση όλους τους κανόνες στους οποίους ο εμπειρογνώμονας είναι owner ή editor.
  - *Δημόσια Εξαγωγή Αποφάσεων «Public Decision Derivation»:* ονομάζεται η εξαγωγή απόφασης που χρησιμοποιεί ως γνώση όλους κανόνες του συστήματος που ορίζονται στο ίδιο Root Knowledge Domain με αυτό του εμπειρογνώμονα.

Το Entity & Decision Management επικοινωνεί με το Core Knowledge System, στέλνοντάς του όλα τα απαραίτητα δεδομένα για την έναρξη της εξαγωγής αποφάσεων. Η αρχικοποίηση αυτή περιλαμβάνει:

  - Λίστα κανόνων με βάση το είδος της εξαγωγής αποφάσεων (Private/Public) που ζητήθηκε.
  - Δυσδιάστατη λίστα γνωρισμάτων (Entity Attributes) της οντότητας (Entity) προς εξαγωγή αποφάσεων.
  - Αναγνωριστικό εμπειρογνώμονα (Expert ID) ο οποίος αιτήθηκε την εξαγωγή αποφάσεων.

Στη συνέχεια το Core Knowledge System εκτελεί συλλογισμούς με χρήση της μηχανής συλλογισμών που διαθέτει και στέλνει πίσω στο Entity & Decision Management δύο λίστες. Η πρώτη λίστα περιλαμβάνει τα αναγνωριστικά των κανόνων που πυροδοτήθηκαν κατά τους συλλογισμούς του συστήματος. Ενώ η δεύτερη λίστα περιλαμβάνει τις αποφάσεις που εξάχθηκαν με χρήση των κανόνων της πρώτης λίστας. Οι αποφάσεις αυτές εμπεριέχουν κάποιον βαθμό βεβαιότητας. Τέλος απαιτείται ενημέρωση της βάσης δεδομένων του συστήματος.

4.4. Βάση Δεδομένων

Η «Βάση Δεδομένων» (Database) αποτελεί τη μακροχρόνια μνήμη του συστήματος. Έχει υλοποιηθεί σε PostgreSQL και ανήκει στην κατηγορία object-relational database management system [17]. Επιλέχθηκε σχεσιακή βάση δεδομένων για την σταθερότητα, ασφάλεια και ακεραιότητα που προσφέρει, κατά τη διαδικασία καταχώρησης ή ενημέρωσης δεδομένων. Οι Locking Mechanism και Queues είναι δύο προεπιλεγμένες τεχνικές που συμβάλουν στην διασφάλιση των παραπάνω.

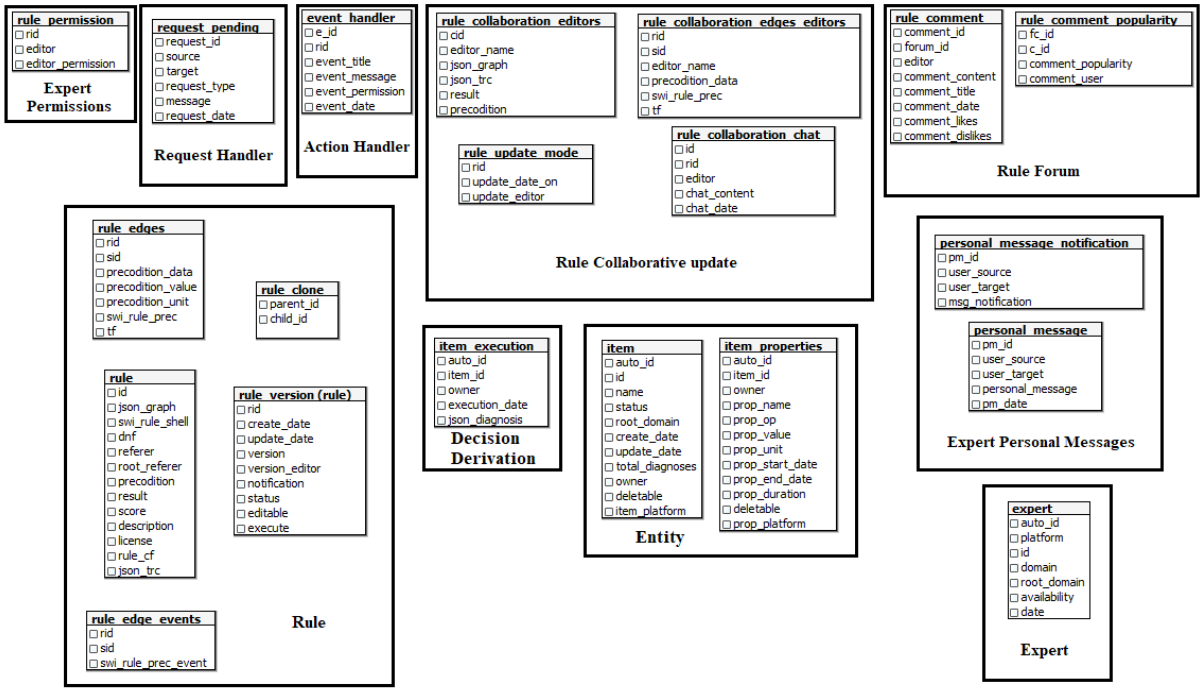

Figure 7 – Παρουσίαση όλων των πινάκων της βάσης δεδομένων σε ομάδες βάση λειτουργίας.

Στην εικόνα 6 παρουσιάζονται εννιά (9) ομάδες πινάκων. Οι ***«Rule», «Entity»*** και ***«Expert»*** αποτελούν τις κύριες ομάδες. Κύριες χαρακτηρίζονται οι ομάδες των οποίων, οι εισερχόμενες αναφορές είναι περισσότερες από τις εξερχόμενες:

- **Rule**

Η ομάδα «Rule» περιλαμβάνει όλους τους πίνακες που συνθέτουν ένα κανόνα. Είναι απαραίτητη τόσο στη κατασκευή όσο και στην ενημέρωση ενός κανόνα. Ο κανόνας αποθηκεύεται σε δύο μορφές:

  - Δεδομένα: όλα τα στοιχεία του κανόνα αποθηκεύονται σε μορφή απλού κειμένου (plain text). Τα στοιχεία αυτά χρησιμεύουν τόσο στην εσωτερική επικοινωνία μεταξύ υποσυστημάτων, όσο και στις υπηρεσίες Restful API.
  - Ανενεργή Γνώση: σύνολο των απαραίτητων στοιχείων (προϋπόθεση, αποτέλεσμα και παράγοντες βεβαιότητας) κανόνα για την μετέπειτα χρήση του ως γνώση του συστήματος. Τα στοιχεία αυτά αποθηκεύονται σε μορφή Prolog κατηγορημάτων. Τα κατηγορήματα αυτά χαρακτηρίζονται γνώση του συστήματος, μόνο όταν φορτωθούν στην βάση γνώσης, έπειτα από επικοινωνία με το Core Knowledge System.

Χαρακτηριστικό της ομάδας αυτής είναι ότι δεν απαιτεί αναφορά σε κάποια άλλη ομάδα πινάκων.

- **Expert**

Η ομάδα «Expert» περιλαμβάνει έναν πίνακα, ο οποίος διατηρεί τα στοιχεία κάθε εμπειρογνώμονα του συστήματος. Χαρακτηριστικό της ομάδας αυτής είναι ότι δεν απαιτεί αναφορά σε κάποια άλλη ομάδα πινάκων.

- **Entity**

Η ομάδα «Entity» περιλαμβάνει όλους τους πίνακες που συνθέτουν μία ολοκληρωμένη οντότητα. Δηλαδή οντότητα με σύνολο από γνωρίσματα. Η ομάδα αυτή απαιτεί αναφορά στην ομάδα Expert.

- **Decision Derivation**

Η ομάδα «Decision Derivation» περιλαμβάνει ένα πίνακα ο οποίος διατηρεί ιστορικό αποφάσεων για κάθε οντότητα κάποιου εμπειρογνώμονα. Η ομάδα αυτή κάνει αναφορά τόσο στην ομάδα Expert όσο και στην ομάδα Entity.

- **Expert Permissions**

Η ομάδα «Expert Permissions» διατηρεί πληροφορίες για το ποιος εμπειρογνώμονας είναι editor ή owner σε ποιο κανόνα. Η ομάδα αυτή κάνει αναφορά τόσο στην ομάδα των Expert όσο και στην ομάδα των Rule.

- **Request Handler**

Η ομάδα «Request Handler» διατηρεί όλα τα ενεργά αιτήματα του συστήματος μεταξύ εμπειρογνωμόνων και κανόνων. Ενεργό είναι το αίτημα το οποίο δεν έχει υποστεί αποδοχή ή απόρριψη. Υποστηρίζονται τρία είδη αιτημάτων:

  - ***Join Request:*** Ο εμπειρογνώμονας ζητά να μπει ως editor σε ένα κανόνα. Η κατεύθυνση αποστολής αιτήματος είναι από εμπειρογνώμονα προς κανόνα.
  - ***Invite Editor Request:*** Ο κανόνας (μέσω του Owner) ζητά να προσθέσει ένα εμπειρογνώμονα στους συνεργάτες. Η κατεύθυνση αποστολής αιτήματος είναι από κανόνα προς εμπειρογνώμονα.
  - ***Commit Rule Request:*** Ο κανόνας κλώνος (μέσω του Owner) ζητά να ενημερώσει ένα κανόνα. Η κατεύθυνση αποστολής αιτήματος είναι από κανόνα προς κανόνα.

Η ομάδα αυτή κάνει αναφορά τόσο στην ομάδα Expert όσο και στην ομάδα Rule.

- **Action Handler**

Η ομάδα «Action Handler» διατηρεί όλα τα μηνύματα ειδοποίησης, των ενεργειών που πραγματοποιεί κάποιος εμπειρογνώμονας. Τα μηνύματα αυτά είναι ορατά τόσο στον ίδιο τον εμπειρογνώμονα όσο και στους συνεργάτες του. Απαιτείται αναφορά στην ομάδα Expert και στην ομάδα Rule.

- **Rule Collaborative Update**

Η ομάδα «Rule Collaborative Update» διατηρεί όλα τα στοιχεία συνεργατικής ενημέρωσης κανόνων. Τα στοιχεία αυτά περιλαμβάνουν για κάθε κανόνα, τον (συνεργάτη) εμπειρογνώμονα και την δική του έκδοση κανόνα. Γίνεται αναφορά τόσο στην ομάδα Expert όσο και στην ομάδα Rule.

- **Rule Forum**

Η ομάδα «Rule Forum» διατηρεί τις δημοσιεύσεις που κάνουν οι εμπειρογνώμονες στο forum ενός κανόνα αλλά και τη δημοτικότητα αυτών. Γίνεται αναφορά τόσο στην ομάδα Expert όσο και στην ομάδα Rule.

- **Expert Personal Messages**

Η ομάδα «Expert Personal Message» περιλαμβάνει πίνακες που εξασφαλίζουν την προσωπική επικοινωνία μεταξύ των εμπειρογνωμόνων. Επίσης εξασφαλίζουν την άμεση ενημέρωση του εμπειρογνώμονα (παραλήπτη) κατά την αποστολή νέου μηνύματος προς αυτόν. Γίνεται αναφορά στην ομάδα Expert.

4.5. Πυρήνας Συστήματος Γνώσης

Ο «Πυρήνας Συστήματος Γνώσης» (Core Knowledge System) αποτελεί βασικό πυλώνα του συστήματος και είναι υλοποιημένος εξ' ολοκλήρου σε SWI Prolog. Χρησιμοποιείται τόσο για την μεταφορά εκτελέσιμης γνώσης από τη Database (μακροχρόνια) στη Knowledge Base (βραχυχρόνια) μνήμη του συστήματος. Όσο και για την διαδικασία της εξαγωγής αποφάσεων.

Αποτελείται από δύο τμήματα λειτουργιών «Μηχανή Συλλογισμών» (Inference Engine) και «Διαχειριστής Βάσης Γνώσης» (KB Management) και δύο τμήματα αποθήκευσης «Βάση Γνώσης» (Knowledge Base) και «Αρχείο Καταγραφής» (Log File).

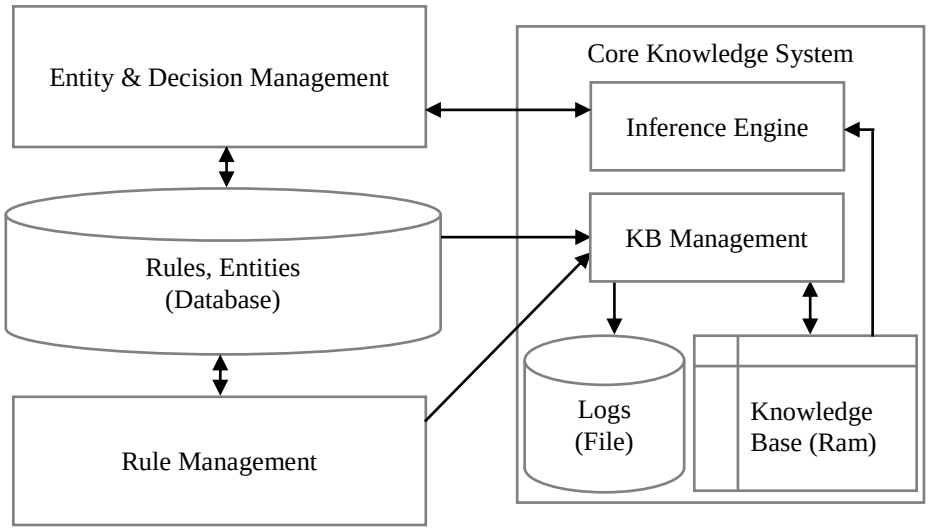

Figure 8 - Σχηματική αναπαράσταση του «Knowledge System» και των επικοινωνιών.

4.5.1. Διαχειριστής Βάσης Γνώσης

Ο «Διαχειριστής Βάσης Γνώσης» (KB Management) είναι το υποσύστημα το οποίο αναλαμβάνει την αποθήκευση ή διαγραφή εκτελέσιμων κανόνων από το τη βραχυχρόνια μνήμη (Knowledge Base) του συστήματος. Η λειτουργία αυτή πυροδοτείτε από το υποσύστημα Rule Management, με στόχο την ετοιμότητα της γνώσης που (εν δυνάμει) θα συμπεριληφθεί κατά τη διαδικασία της εξαγωγής αποφάσεων. Όταν ο SWI Server από κατάσταση offline (π.χ. για λόγους συντήρησης του συστήματος) μεταβαίνει σε κατάσταση online, το KB Management επικοινωνεί με τη βάση δεδομένων για επαναφορά της εκτελέσιμης γνώσης στη βάση γνώσης. Κάθε ενέργεια που εκτελείται στο KB Management παράγει δεδομένα καταγραφής της ενέργειας αυτής τα οποία αποθηκεύονται χρονολογικά ορισμένα στα «Log Files» του συστήματος.

4.5.2. Μηχανή Συλλογισμών

Η «Μηχανή Συλλογισμών» (Inference Engine) έχει υλοποιηθεί χρησιμοποιώντας την μηχανή εκτέλεσης κατηγορημάτων (Execution Engine) και τις βιβλιοθήκες (Libraries) που παρέχει η SWI Prolog. Η μηχανή συλλογισμών αναλύεται σε ένα αναδρομικό αλγόριθμο, σκοπός του οποίου είναι η πυροδότηση συγκεκριμένου εύρους κανόνων της βάσης γνώσης. Η πυροδότηση των κανόνων αυτών βασίζεται στην χρήση των γνωρισμάτων μίας οντότητας ως έγκυρα γεγονότα. Ο αλγόριθμος προσπαθεί να ενοποιήσει τα γεγονότα αυτά με τις συνθήκες των κανόνων.. Το εύρος των κανόνων όπως και η οντότητα ορίζεται έπειτα από επικοινωνία με το Entity & Decision Management.

Μία εξαγωγή απόφασης μπορεί να καταλήξει ως προϋπόθεση ενός άλλου κανόνα, δημιουργώντας έτσι αλυσίδες κανόνων «Pipelines» που στόχο έχουν την εξαγωγή όσο το δυνατό περισσότερο έγκυρων αποφάσεων πυροδοτώντας λιγότερους κανόνες. Επιπλέον η μηχανή συλλογισμών μπορεί να καταλήξει σε ίδια απόφαση παραπάνω από μία φορά. Για το λόγο αυτό χρησιμοποιείται η μέθοδος αντιμετώπισης σύγκρουσης ιδίων αποφάσεων της Θεωρίας Βεβαιότητας. Η μέθοδος αυτή ορίζει ότι, εάν δύο κανόνες με διαφορετικές συνθήκες έχουν ίδιο αποτέλεσμα, τότε οι παράγοντες βεβαιότητας του αποτελέσματος των κανόνων αυτών συνδυάζονται βάση του παρακάτω τύπου:

NewResultCF = (ResultCF1 + ResultCF2) – (ResultCF1 * ResultCF2)

Συνεπώς ο παραπάνω τύπος μειώνει στη μία φορά την κάθε πολλαπλής εμφάνισης απόφαση, ενώ ταυτόχρονα ορίζει νέο επαυξημένο παράγοντα βεβαιότητας (NewResultCF) για την απόφαση αυτή. Το τελικό αποτέλεσμα όλων των αποφάσεων μιας οντότητας επιστρέφεται πίσω στο Entity & Decision Management.

Ο αλγόριθμος της μηχανής συλλογισμών παίρνει ως **Είσοδο** δύο λίστες. Μία λίστα που αποτελείται από αναγνωριστικά εκτελέσιμων κανόνων (π.χ. [R1, R23, R2, R45]). Η λίστα αυτή ονομάζεται «***Executable Rule List***» και καθορίζεται από το είδος εξαγωγής απόφασης (private ή public) που επίλεξε ο εμπειρογνώμονας. Μία δισδιάστατη λίστα γνωρισμάτων της μορφής [[Name1, Value1, Unit1, Duration1, Distance Relevance1], …, [NameN, ValueN, UnitN, DurationN, Distance RelevanceN]]. Η λίστα αυτή ονομάζεται «***Entity Attribute List***», με κάθε εσωτερική της λίστα να αντιστοιχεί σε ένα γνώρισμα της εξεταζόμενης οντότητας. Έπειτα ο αλγόριθμος παράγει μία νέα λίστα με ονομασία «***Initial Executable Rule List***», ως αντίγραφο της «Executable Rule List». Παρακάτω παρουσιάζεται ο αλγόριθμος της μηχανής συλλογισμών σε ψευδογλώσσα, ενώ στο Παράρτημα Α παρουσιάζεται ο ίδιος αλγόριθμος σε μορφή SWI Prolog:

ΒΗΜΑ 1.

Για κάθε γνώρισμα της λίστας γνωρισμάτων «Entity Attribute List» **κάνε**:

- **Εάν** υπάρχει συνθήκη κανόνα που ενοποιείται με το γνώρισμα αυτό λογικά μέσω του κατηγορήματος «*rule_prec*» **και** χρονικά μέσω του κατηγορήματος «*rule_prec_events*», **Τότε** καταχώρησε το [Rule ID, Rule Condition ID] (αναγνωριστικό) της συνθήκης αυτής σε μία δισδιάστατη λίστα ικανοποιημένων συνθηκών «***Triggered Prec List***».

Αλλιώς true.

Συνέχισε στο βήμα 2.

ΒΗΜΑ 2.

Εάν η «Triggered Prec List» είναι άδεια **και** η «Executable Rule List» είναι ίδια με την «Initial Executable Rule List», **Τότε** τερμάτισε χωρίς απόφαση.

Αλλιώς Εάν η «Triggered Prec List» είναι άδεια **και** η «Executable Rule List» δεν είναι ίδια με την «Initial Executable Rule List», **Τότε** πήγαινε στο βήμα 4.

Αλλιώς συνέχισε.

Εάν δεν υπάρχει κανόνας ο οποίος να πυροδοτείται μέσω της ενοποίησης του κατηγορήματος «*rule_shell*» με κάποιο συνδυασμό των συνθηκών που περιλαμβάνει η λίστα «Triggered Prec List», **Τότε** true και συνέχισε στο βήμα 3.

Αλλιώς το αποτέλεσμα του πρώτου κανόνα που ικανοποιείται, δηλαδή το «*Result*» του κατηγορήματος «*rule_shell*», καταχωρείται στη λίστα γνωρισμάτων «***Entity Attribute List***»

:= «Entity Attribute List» + [NameResult, ValueResult, UnitResult, DurationResult, Distance Relevance Result] **και** ο κανόνας διαγράφεται από τη λίστα εκτελέσιμων κανόνων «***Executable Rule List***» := «Executable Rule List» - [Rule ID] **και** καθαρίζεται η λίστα «Triggered Prec List»:= [ ]. Συνέχισε στο βήμα 3.

ΒΗΜΑ 3.

Επανάλαβε τα βήμα 1 και βήμα 2, **Μέχρις ότου** να μην υπάρξει πυροδότηση κανόνα (βλ. βήμα 2) για ακριβώς δύο συνεχόμενες επαναλήψεις των βημάτων 1 και 2. Συνέχισε στο βήμα 4.

ΒΗΜΑ 4.

Παραγωγή της λίστας «***Executed Rule List***» ως αποτέλεσμα της αφαίρεσης των λιστών «New Executable Rule List» και «Initial Executable Rule List». Η νέα αυτή λίστα περιλαμβάνει όλους τους κανόνες που πυροδοτήθηκαν από τον αλγόριθμο. Δηλαδή, «Executed Rule List» := «Initial Executable Rule List» - «Executable Rule List».

ΒΗΜΑ 5.

Για κάθε κανόνα της λίστας «Executed Rule List» **κάνε**:

- Βρες το αποτέλεσμα κανόνα (Result) και τον παράγοντα βεβαιότητας αποτελέσματος (Result_CF), μέσω του κατηγορήματος «rule_shell» που αντιστοιχεί στο κανόνα αυτό.
- Καταχώρησε τα δύο αυτά στοιχεία, σε μία δισδιάστατη λίστα της μορφής [[Result1, Result_CF1],…, [ResultN, Result_CFN]]. Η νέα αυτή λίστα ονομάζεται «***Decision Derivation List***». Δηλαδή «Decision Derivation List» := «Decision Derivation List» + [[Rule Result, Rule Result CF]].

Συνέχισε στο βήμα 6.

ΒΗΜΑ 6.

Συγχώνευση διπλών εμφανίσεων των «*Result*» που περιλαμβάνει η «Decision Derivation List», με χρήση της θεωρίας βεβαιότητας. Μία ενδεχόμενη συγχώνευση, επηρεάζει τόσο τα «*Result*» όσο και τα «*Result_CF*» της λίστας αυτής. Συνέχισε στο βήμα 7.

ΒΗΜΑ 7.

Η **Έξοδος** του αλγορίθμου περιλαμβάνει, μία λίστα κανόνων που εκτελέστηκαν «Executed Rule List» **και** μία δισδιάστατη λίστα αποφάσεων - παραγόντων βεβαιότητας «Decision Derivation List».

Για την καλλίτερη κατανόηση του παραπάνω αλγορίθμου από τον αναγνώστη, ακολουθεί ένα απλό αλλά πλήρες σενάριο. Στο σενάριο αυτό υπάρχει ένας (εμπειρογνώμονας) παθολόγος, ο οποίος έχει στη προσωπική του βάση γνώσης τους παρακάτω πέντε κανόνες.

- **R1**: **Εάν** {C1} θερμοκρασία σώματος > 36.8 Celsius, **Τότε** πυρετός {Result_CF = 100%}.
- **R2**: **Εάν** {C1} θερμοκρασία σώματος < 36.3 Celsius, **Τότε** υποθερμία {Result_CF = 100%}.
- **R3**: **Εάν** {C1}πυρετός **και** {C2}ξηρός βήχας με διάρκεια περισσότερο ή ίσο από 26 ώρες **και (**{C3}ρινική συμφόρηση **ή** {C4}ρινική καταρροή**) και** {C5}μυϊκός πόνος μία μέρα μετά τον πυρετό, **Τότε** κρυολόγημα{Result_CF = 75%}.
- **R4**: **Εάν** {C1} πυρετός με διάρκεια περισσότερο από 20 ώρες **και (**{C2} φλύκταινες στο δέρμα με εμφάνιση τρείς ώρες μετά από την έναρξη πυρετού **ή** {C3} θολές στο δέρμα**), Τότε** ανεμοβλογιά {Result_CF = 80%}.
- **R5**: **Εάν** {C1} πυρετός **και** {C2} θολές στο δέρμα **Τότε** ανεμοβλογιά {Result_CF = 67%}**.**

Εκτός από τους παραπάνω κανόνες, ο παθολόγος αυτός διαχειρίζεται μία οντότητα με ονομασία «Ασθενής Γιώργος Γ.». Η οντότητα αυτή έχει τα παρακάτω τέσσερα γνωρίσματα.

- ✓ **Attr1**: θερμοκρασία σώματος είναι 39.1 Celsius από τις 10/20/2017 03:10:43 έως τις 10/22/2017 03:8:43.
- ✓ **Attr2**: ξηρός βήχας από τις 10/20/2017 23:10:43.
- ✓ **Attr3**: θολές στο δέρμα.
- ✓ **Attr4**: πονοκέφαλος από τις 10/20/2017 22:10:03 έως τις 10/21/2017 08:10:32.

Έστω λοιπόν ότι ο παθολόγος ζητά από το σύστημα να εξάγει αποφάσεις για τη συγκεκριμένη οντότητα, με χρήση μόνο της γνώσης που αυτός έχει στη προσωπική του βάση γνώσης. Δηλαδή επιλέγει Private Decision Derivation για την οντότητα «Γιώργος Γ. Ασθενής». Παρακάτω ακολουθεί η αντίδραση του αλγορίθμου μηχανής συλλογισμού, στην απαίτηση του παθολόγου για ιδιωτική εξαγωγή αποφάσεων. Σημείωση, η υπογράμμιση «_» ως στοιχείο σε λίστα σημαίνει μη ορισμένη τιμή (Undefined).

Ο αλγόριθμος της μηχανής συλλογισμών δέχεται ως είσοδο δύο λίστες. Την «***Executable Rule List***» = [R1, R2, R3, R4, R5] και την «***Entity Attribute List***» = [[θερμοκρασία σώματος, 39.1, Celsius, 180000, 0], [ξηρός βήχας, true, _, _, 72000], [θολές στο δέρμα, true, _, _, _], [πονοκέφαλος, true, _, _, 68440]]. Επίσης ο αλγόριθμος δημιουργεί την «***Initial Executable Rule List***» = [R1, R2, R3, R4, R5]. Η εκτέλεση του αλγορίθμου σύμφωνα με το συγκεκριμένο σενάριο, πραγματοποιείται ως εξής:

ΒΗΜΑ 1.

Είσοδος:

- Executable Rule List
- Entity Attribute List

Ενέργεια / Έξοδος:

- **Triggered Prec List** = [[R1, C1], [R3, C2], [R4, C3], [R5, C2]].

ΒΗΜΑ 2.

Είσοδος:

- Executable Rule List
- Triggered Prec List

Ενέργεια / Έξοδος

- **Entity Attribute List** = [[θερμοκρασία σώματος, 39.1, Celsius, 180000, 0], [ξηρός βήχας, true, _, _, 72000], [θολές στο δέρμα, true, _, _, _], [πονοκέφαλος, true, _, _, 68440], [πυρετός, true, _, _, _]].
- **Executable Rule List** = [~~R1~~, R2, R3, R4, R5]. % Διαγραφή κανόνα R1 λόγω πυροδότησής του.
- **Triggered Prec List** = [].

ΒΗΜΑ 3.

Είσοδος:

- Initial Executable Rule List
- Executable Rule List

Ενέργεια / Έξοδος:

- Επιστροφή στο βήμα 1.

ΒΗΜΑ 1.

Είσοδος:

- Executable Rule List
- Entity Attribute List

Ενέργεια / Έξοδος:

- **Triggered Prec List** = [[R3, C1], [R3, C2], [R4, C1], [R4, C3], [R5, C1], [R5, C2]].

ΒΗΜΑ 2.

Είσοδος:

- Executable Rule List
- Triggered Prec List

Ενέργεια / Έξοδος:

- **Entity Attribute List** = [[θερμοκρασία σώματος, 39.1, Celsius, 180000, 0], [ξηρός βήχας, true, _, _, 72000], [θολές στο δέρμα, true, _, _, _], [πονοκέφαλος, true, _, _, 68440], [πυρετός, true, _, _, _], [ανεμοβλογιά, true, _, _, _]].
- **Executable Rule List** = [R2, R3, ~~R4~~, R5]. % Διαγραφή κανόνα R4 λόγω πυροδότησής του.
- **Triggered Prec List** = [].

ΒΗΜΑ 3.

Είσοδος:

- Initial Executable Rule List
- Executable Rule List

Ενέργεια / Έξοδος:

- Επιστροφή στο βήμα 1.

ΒΗΜΑ 1.

Είσοδος:

- Executable Rule List
- Entity Attribute List

Ενέργεια / Έξοδος:

- **Triggered Prec List** = [[R3, C1], [R3, C2], [R5, C1], [R5, C2]].

ΒΗΜΑ 2.

Είσοδος:

- Executable Rule List
- Triggered Prec List

Ενέργεια / Έξοδος:

- **Entity Attribute List** = [[θερμοκρασία σώματος, 39.1, Celsius, 180000, 0], [ξηρός βήχας, true, _, _, 72000], [θολές στο δέρμα, true, _, _, _], [πονοκέφαλος, true, _, _, 68440], [πυρετός, true, _, _, _], [ανεμοβλογιά, true, _, _, _], [ανεμοβλογιά, true, _, _, _]].
- **Executable Rule List** = [R2, R3, ~~R5~~]. % Διαγραφή κανόνα R5 λόγω πυροδότησής του.
- **Triggered Prec List** = [].

ΒΗΜΑ 3.

Είσοδος:

- Initial Executable Rule List
- Executable Rule List

Ενέργεια / Έξοδος:

- Επιστροφή στο βήμα 1.

ΒΗΜΑ 1.

Είσοδος:

- Executable Rule List
- Entity Attribute List

Ενέργεια / Έξοδος:

- **Triggered Prec List** = [[R3, C1], [R3, C2]].

ΒΗΜΑ 2.

Είσοδος:

- Executable Rule List
- Triggered Prec List

Ενέργεια / Έξοδος:

- Μετακίνηση στο βήμα 3.

ΒΗΜΑ 3.

Είσοδος:

- Initial Executable Rule List
- Executable Rule List

Ενέργεια / Έξοδος:

- Συνέχισε στο βήμα 4.

ΒΗΜΑ 4.

Είσοδος:

- Initial Executable Rule List % [R1, R2, R3, R4, R5]
- Executable Rule List % [R2, R3]

Ενέργεια / Έξοδος:

- **Executed Rule List** = [R1, R4, R5].

ΒΗΜΑ 5.

Είσοδος:

- Executed Rule List

Ενέργεια / Έξοδος:

- **Decision Derivation List** = [[πυρετός, 100], [ανεμοβλογιά, 80], [ανεμοβλογιά, 67]].

ΒΗΜΑ 6.

Είσοδος:

- Decision Derivation List

Ενέργεια / Έξοδος:

- **Final Decision Derivation List** = [[πυρετός, 100], [ανεμοβλογιά, 93.4]]

ΒΗΜΑ 7.

Είσοδος: τίποτα.
Ενέργεια / Έξοδος:

- Executed Rule List.
- Final Decision Derivation List.

4.5.3. Βάση Γνώσης

Η «Βάση Γνώσης» (Knowledge Base) περιλαμβάνει όλους τους εκτελέσιμους κανόνες σε μορφή Prolog κατηγορημάτων. Οι κανόνες αυτοί χρησιμοποιούνται ως έγκυρη γνώση, στους συλλογισμούς του συστήματος κατά την διαδικασία της εξαγωγής αποφάσεων. Η βάση γνώσης αποτελεί τη βραχυχρόνια μνήμη του συστήματος, καθώς βρίσκεται στην μνήμη τυχαίας προσπέλασης «Random Access Memory (RAM)» [18] του SWI server. Η επιλογή χρήσης RAM αντί βάσης δεδομένων ή Prolog (.PL) αρχείο, έχει στόχο τη διασφάλιση της μέγιστης ταχύτητας προσπέλασης και ακεραιότητας δεδομένων αντίστοιχα. Με τον τρόπο αυτό λοιπόν μειώνεται ο χρόνος των δαπανηρών αναδρομών, κερδίζοντας ταχύτητα τόσο στην πυροδότηση κανόνων όσο και στην έρευνα για αλυσιδωτή εξαγωγή αποφάσεων.

4.5.4. Αρχείο Καταγραφής

Το «Αρχείο Καταγραφής» (Log File) είναι αρχείο κειμένου το οποίο διατηρεί αποτελέσματα ενεργειών σε χρονολογική σειρά. Τα αποτελέσματα αυτά αφορούν ενέργειες που πραγματοποιούνται στην επικοινωνία του Rule Management με το KB Management. Όταν ο εμπειρογνώμονας αλληλοεπιδράσει για πρώτη φορά με το Core Knowledge System (μέσω του CKMS), δημιουργείται ένα Log File με ονομασία το expert id του. Έπειτα κάθε φορά που ο εμπειρογνώμονας επικοινωνεί με το KB Management (μέσω του Rule Management) το αρχείο αυτό ενημερώνεται αυτόματα. Όταν ο SWI Server γίνει για πρώτη φορά online, δημιουργείται ένα Log File, το οποίο διατηρεί πληροφορίες για την συχνότητα online και offline του συστήματος.

4.6. Διαχειριστής Οντολογιών

Το τμήμα «Διαχειριστής Οντολογιών» (Ontologies Management) αποτελεί ένα σύνολο από αναλυτές σημασιολογικού ιστού (Semantic Web Parsers) υλοποιημένο σε Java. Οι αναλυτές αυτοί ειδικεύονται στις τεχνολογίες XML, OWL, RDF, JSON και CSV. Στόχος του τμήματος είναι, η ανάλυση οντολογιών σημασιολογικού ιστού και η αρχικοποίηση ή ενημέρωση του Ontologies Search Platform με τις οντολογίες αυτές. Το Ontologies Management δεν είναι *"on the fly"* προσβάσιμο μέσω της διαδικτυακής διεπαφής, αλλά μέσω Integrated Development Environment [19]. Η ανάλυση νέων οντολογιών και η ενημέρωση της πλατφόρμας εύρεσης οντολογιών αποτελεί χρονοβόρα και δαπανηρή διαδικασία για τους πόρους του συστήματος. Εξαετίας της φύσης των οντολογιών (π.χ. XML) μπορεί να απαιτείται προγραμματιστική επέκταση του τμήματος αυτού.

4.7. Δίαυλος Επικοινωνίας με Εξωτερικά Συστήματα

Ο «Δίαυλος Επικοινωνίας με Εξωτερικά Συστήματα» (Restful API) έχει στόχο την εξουσιοδότηση των client που επιθυμούν απομακρυσμένη πρόσβαση σε λειτουργίες του συστήματος. Η εξουσιοδότηση αυτή επιτυγχάνεται με την διανομή Json Web Token. Τα JWT δημιουργούνται και επαληθεύονται με χρήση της Java βιβλιοθήκης nimbus [20].

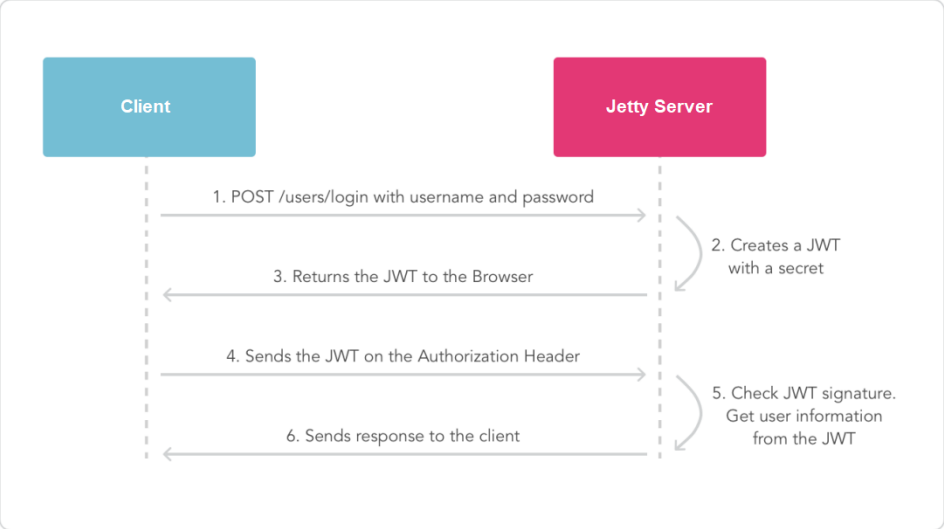

Figure 9 - Διάγραμμα ροής διαδικασίας απόκτησης και χρήσης JWT από τον client.

Στην εικόνα 8 παρουσιάζεται η επικοινωνία μεταξύ Client και Jetty Server. Η επικοινωνία αυτή περιλαμβάνει δύο λειτουργίες:

- ***Χορήγηση JWT***
  - Βήμα 1: ο client στέλνει τα στοιχεία του στον server.
  - Βήμα 2: ο server χρησιμοποιώντας τα στοιχεία του client, παράγει ένα προσωποποιημένο JWT με χρονική διάρκεια ισχύος μίας εβδομάδας.
  - Βήμα 3: ο server στέλνει το παραχθέν JWT στον client.
- ***Επαλήθευση JWT***
  - Βήμα 4: ο client στέλνει το JWT (που του έχει χορηγηθεί) στον server.
  - Βήμα 5: ο server πιστοποιεί την γνησιότητα του JWT και έπειτα ταυτοποιεί τον αιτών client μέσω αυτού.
  - Βήμα 6: ο server ενημερώνει τον client με μήνυμα ανάλογο με το αποτέλεσμα του βήματος πέντε.

Παρακάτω παρουσιάζονται σε μορφή πίνακα οι υπηρεσίες που παρέχονται μέσω του Restful API για τους Client και οι απαιτήσεις αυτών.

URL	HTTP REQUEST METHOD	REQUEST HEADER	REQUEST BODY	JWT AUTHORIZATION	DESCRIPTION
/api/v1.0/login	POST		❖ username ❖ platform		Αποστολή στοιχείων λογαριασμού για απόκτηση jwt.
/api/v1.0/rules/executable	GET	Authorization		✔	Άντληση ID όλων των εκτελέσιμων κανόνων.

Figure 10 - Πίνακας παρουσίασης υπηρεσιών που προσφέρονται μέσω του Restful API.

5. Σενάρια Επίδειξης Βασικών Λειτουργιών του Συστήματος

5.1. Δημιουργία Λογαριασμού Εμπειρογνώμονα.

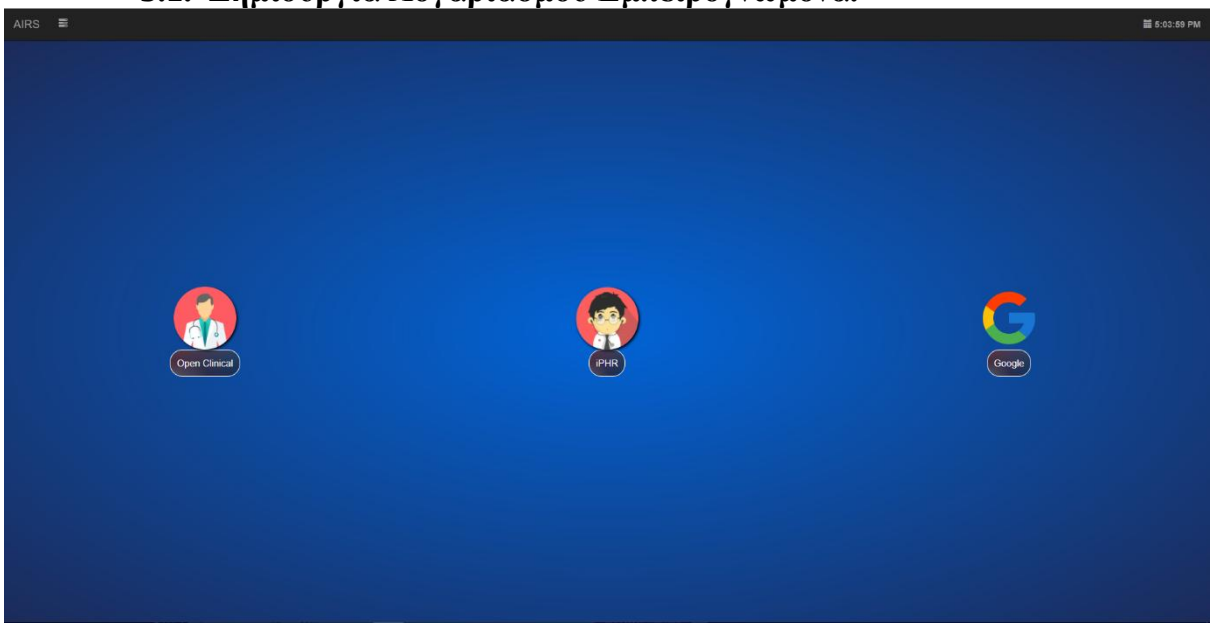

Figure 11 - Διαθέσιμες πλατφόρμες ταυτοποίησης εμπειρογνώμονα.

Αρχικά ο χρήστης επιλέγει μία από τις τρείς πλατφόρμες συνεργάτες του συστήματος μέσω της οποίας επιθυμεί να δημιουργήσει λογαριασμό εμπειρογνώμονα. Η επιλογή πλατφόρμας δεσμεύει καθώς οι πλατφόρμες «openClinical» και «iPHR» παρέχουν πεδία γνώσης με ρίζα το «Medicine» σε αντίθεση με την πλατφόρμα της Google που δεν έχει περιορισμό στην ρίζα πεδίου γνώσης.

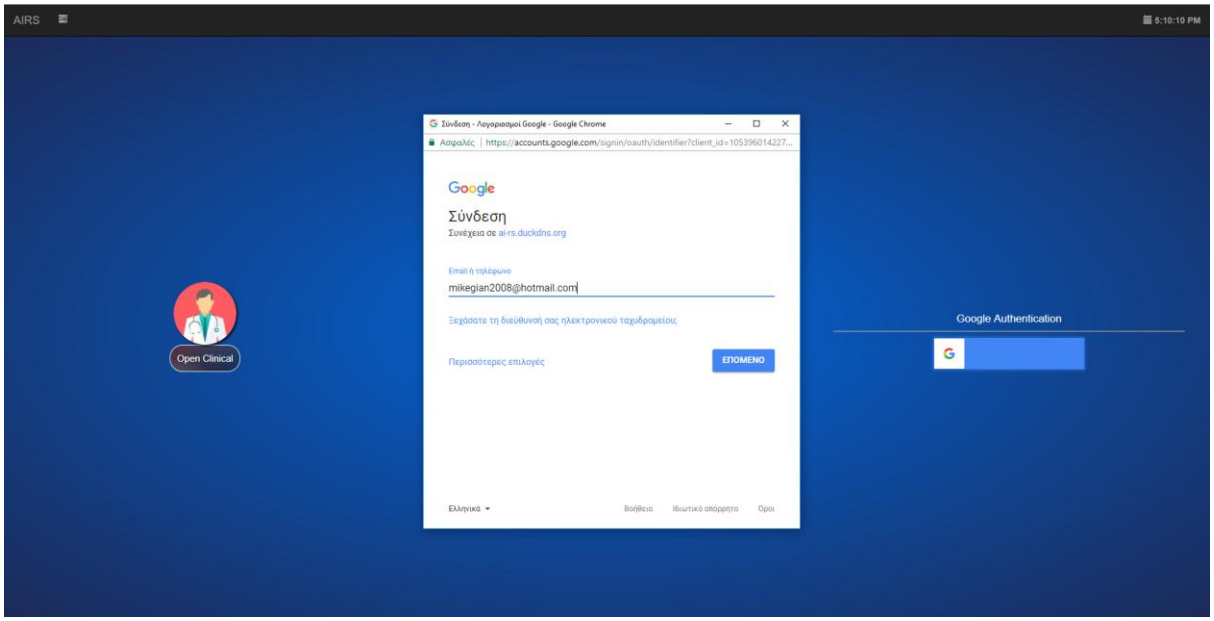

Figure 12 - Επιλογή πλατφόρμας Google για είσοδο στο σύστημα.

Έστω λοιπόν ότι ο ενδιαφερόμενος χρήστης επιθυμεί ταυτοποίηση μέσω της πλατφόρμας Google. Πατώντας λοιπόν πάνω στο εικονίδιο της Google θα του εμφανιστεί μία φόρμα συμπλήρωσης στοιχείων της Google, στην οποία πρέπει να εισάγει email και password.

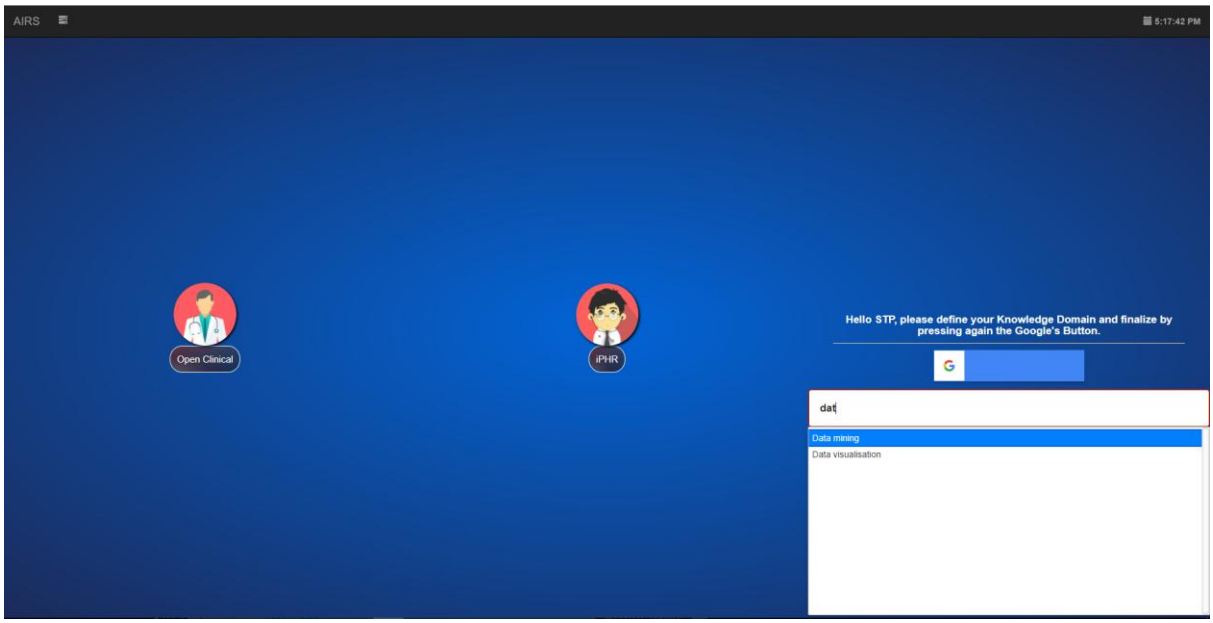

Figure 13 – Επιλογή Πεδίου Γνώσης Εμπειρογνώμονα.

Εφόσον συμπληρώσει σωστά τα στοιχεία του, η πλατφόρμα της Google τον αναγνωρίζει, όμως το COSMOS δεν τον εντόπισε στους εμπειρογνώμονες του συστήματος. Γι' αυτό το λόγο του εμφανίζει ένα πεδίο για να εισάγει το πεδίο γνώσης του ώστε να τον καταχωρήσει ως εμπειρογνώμονα στο σύστημα. Εισάγοντας ως πεδίο γνώσης το «Data mining» και πατώντας για τελευταία φορά πάνω στο κουμπί της Google, πραγματοποιείται καταχώρηση στο σύστημα και τον εξουσιοδοτεί με είσοδο.

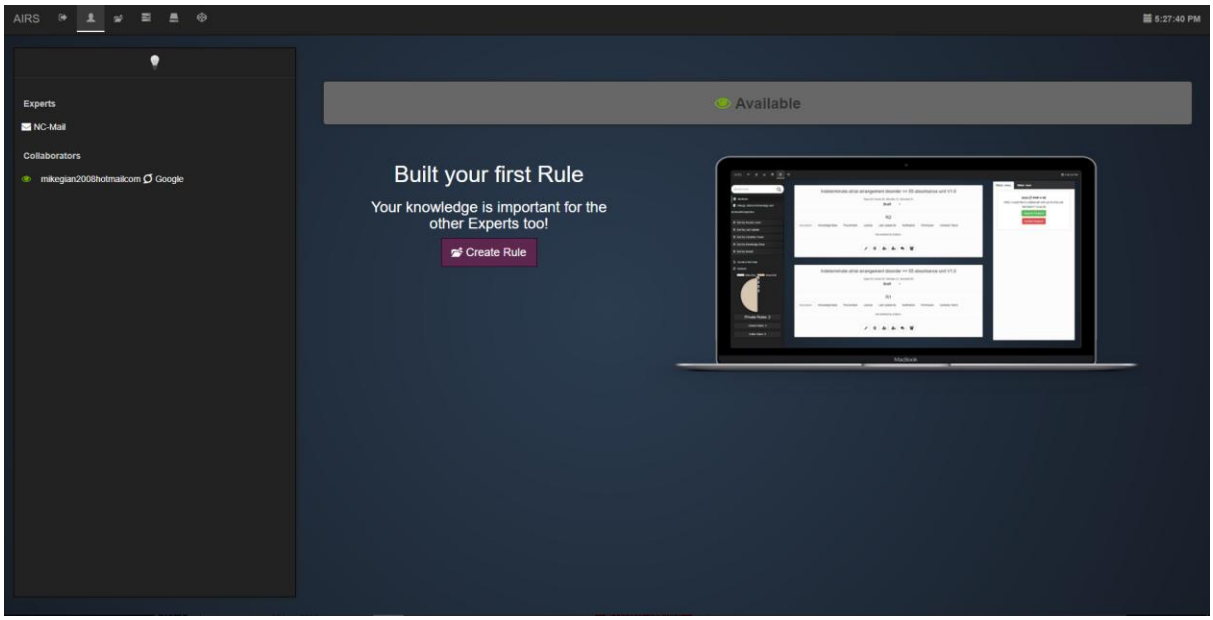

Figure 14 – Κεντρικό μενού επικοινωνίας και προβολής ενεργειών Εμπειρογνώμονα.

Με την πραγματοποίηση εισόδου εμπειρογνώμονα τον μεταφέρει στο «Expert Actions». Στην αριστερή πλευρά περιλαμβάνονται λειτουργίες επικοινωνίας με άλλους εμπειρογνώμονες του συστήματος, όπως επίσης και στοιχεία λογαριασμού εμπειρογνώμονα τα οποία γίνονται ορατά εάν πατήσουμε πάνω στο λογότυπο του συστήματος. Στη δεξιά πλευρά θα εμφανίζονται ενέργειες του ίδιου του εμπειρογνώμονα αλλά και των συνεργατών του πάνω στους κανόνες που διαχειρίζεται.

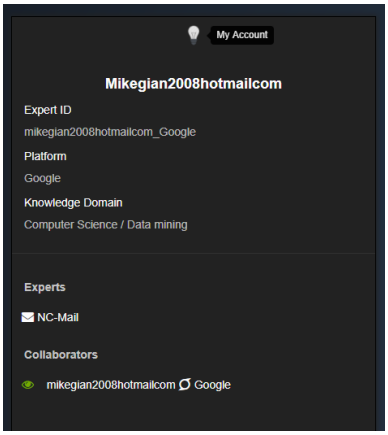

Figure 15 - Στοιχεία λογαριασμού συνδεδεμένου εμπειρογνώμονα.

Στην εικόνα 14 παρουσιάζονται τα στοιχεία λογαριασμού του τρέχοντος εμπειρογνώμονα μέσα στα οποία περιλαμβάνεται και το πεδίο γνώσης στο οποίο ανήκει «Data mining» ενώ πριν από αυτό παρουσιάζεται και η ρίζα του πεδίου γνώσης «Computer Science» στο οποίο βρίσκεται. Τέλος είναι σημαντικό να αναφέρουμε ότι μόνο εμπειρογνώμονες ίδιας ρίζας και ανεξάρτητα πεδίου γνώσης μπορούν να συνεργαστούν μεταξύ τους.

5.2. Δημιουργία και προβολή κανόνα στη προσωπική βάση γνώσης.

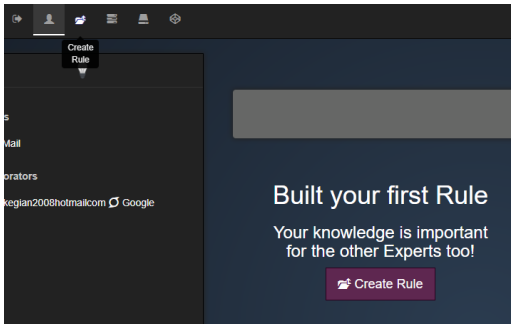

Figure 16 - Κουμπί μετάβασης στη δημιουργία κανόνα.

Επιλέγοντας το εικονίδιο δημιουργίας κανόνα πάνω αριστερά ή ως νέος εμπειρογνώμονας αυτό που βρίσκεται στο κέντρο μπορούμε να μεταβούμε στη διεπαφή δημιουργίας κανόνα.

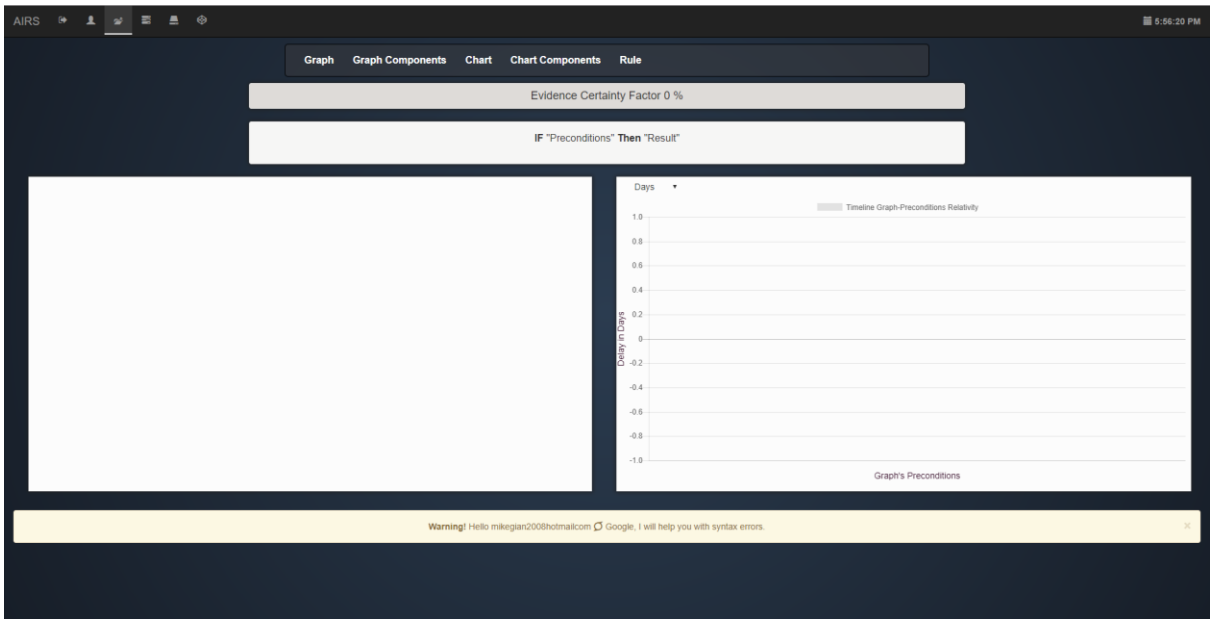

Figure 17 - Διεπαφή δημιουργίας κανόνα.

Στην εικόνα 16 παρουσιάζεται η διεπαφή δημιουργίας κανόνα. Στην πάνω πλευρά διακρίνουμε μενού επιλογών για ενέργειες πάνω στον Γράφο ή πάνω στο διάγραμμα ή πάνω στον κανόνα. Ακριβώς από κάτω συναντάμε το πλαίσιο παρουσίασης του παράγοντα βεβαιότητας των προϋποθέσεων, ενώ αμέσως μετά παρουσιάζεται η αναπαράσταση κανόνα σε if-then μορφή. Στη

δεξιά πλευρά βλέπουμε ένα άδειο δυσδιάστατο διάγραμμα line chart στο οποίο θα αποτυπωθεί η χρονική διασύνδεση των προϋποθέσεων. Στην αριστερή πλευρά βλέπουμε έναν άδειο Γράφο στον οποίο θα αποτυπωθεί η λογική διασύνδεση των προϋποθέσεων. Τέλος στην κάτω πλευρά παρουσιάζεται ένα alert box, το οποίο μας ειδοποιεί για οποιοδήποτε λάθος κάνουμε είτε σε επίπεδο συντακτικού, είτε μη επιτρεπόμενη ενέργεια, είτε για την επιτυχή δημιουργία κανόνα.

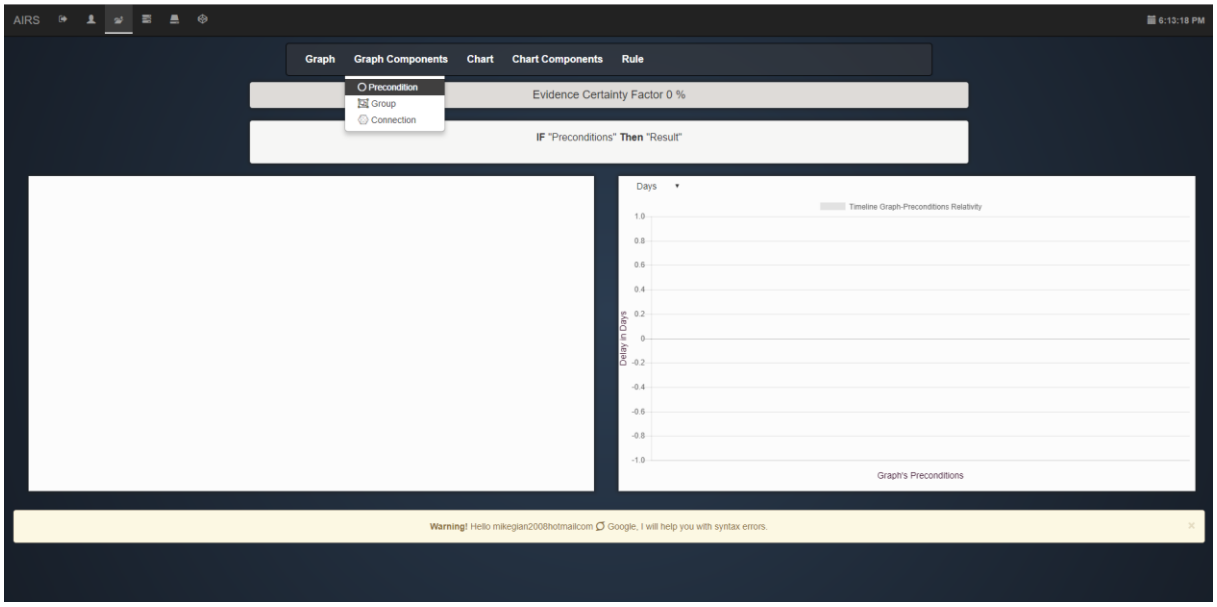

Figure 18 - Επιλογή "Precondition" από μενού ενεργειών.

Για τη έναρξη δημιουργίας προϋπόθεσης κανόνα επιλέγουμε «Precondition» από το «Graph Components» του μενού ενεργειών. Με την ενέργεια αυτή θα εμφανιστεί η παρακάτω φόρμα συμπλήρωσης στοιχείων τα οποία θα αποτελέσουν την προϋπόθεση.

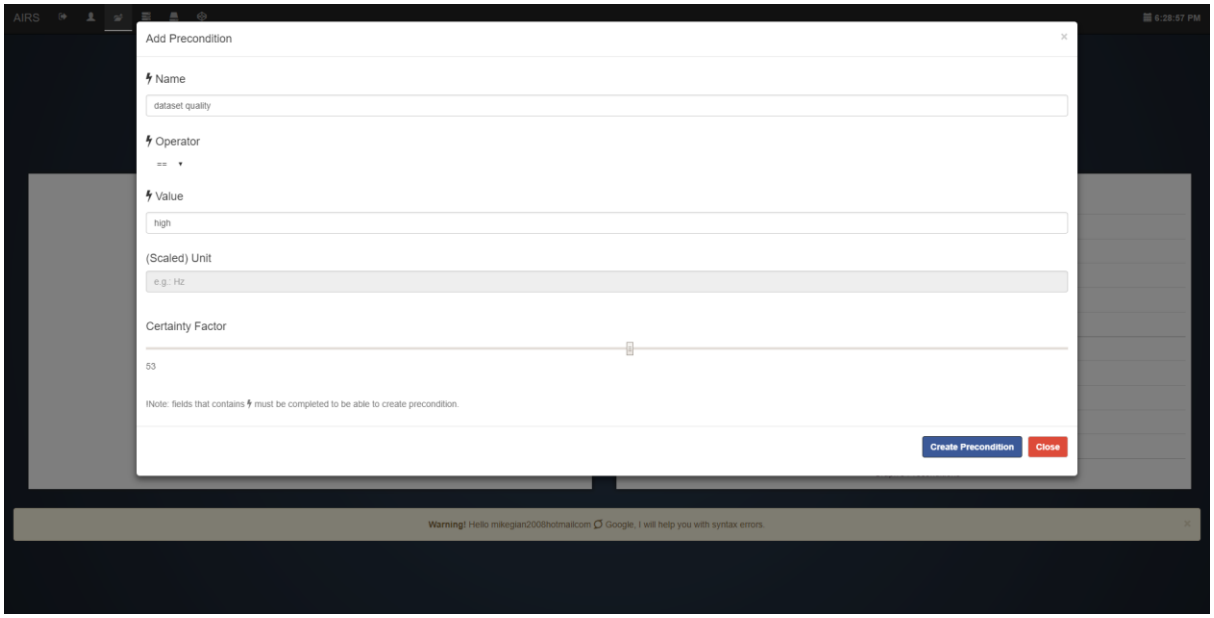

Figure 19 – Φόρμα δημιουργίας προϋπόθεσης και συμπλήρωση πεδίων.

Στη συνέχεια ο εμπειρογνώμονας πρέπει να συμπληρώσει τη φόρμα αυτή ορίζοντας τουλάχιστον το όνομα, τελεστή και τιμή προϋπόθεσης. Όσον αφορά τον παράγοντα βεβαιότητας της προϋπόθεσης, μπορεί είτε να τον αφήσει 50% (by default) είτε να τον παραμετροποιήσει ανάλογα με το επίπεδο βεβαιότητας αληθείας.

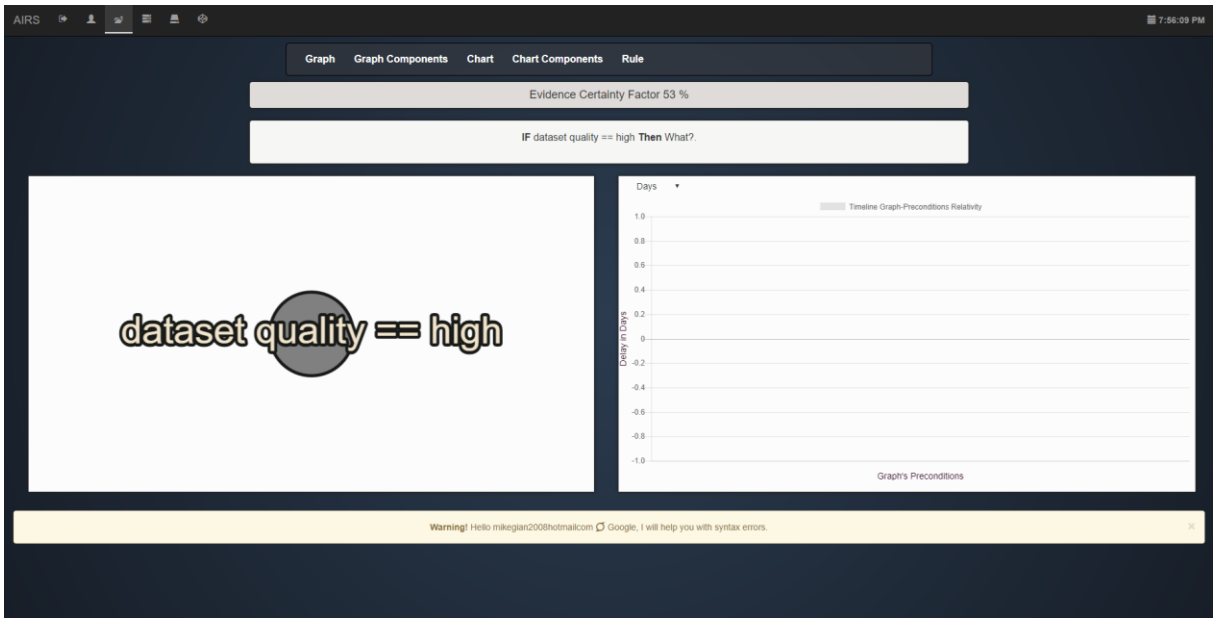

Figure 20 - Αποτέλεσμα δημιουργίας προϋπόθεσης.

Αποτέλεσμα δημιουργίας της προϋπόθεσης μία σειρά από αλλαγές. Καθώς επηρεάστηκε το «Evidence Certainty Factor» με βάση τη θεωρία βεβαιότητας, η if-then αναπαράσταση του κανόνα, όπως επίσης και ο Γράφος στον οποίο προστέθηκε ένας νέος κόμβος που έχει το ρόλο της προϋπόθεσης.

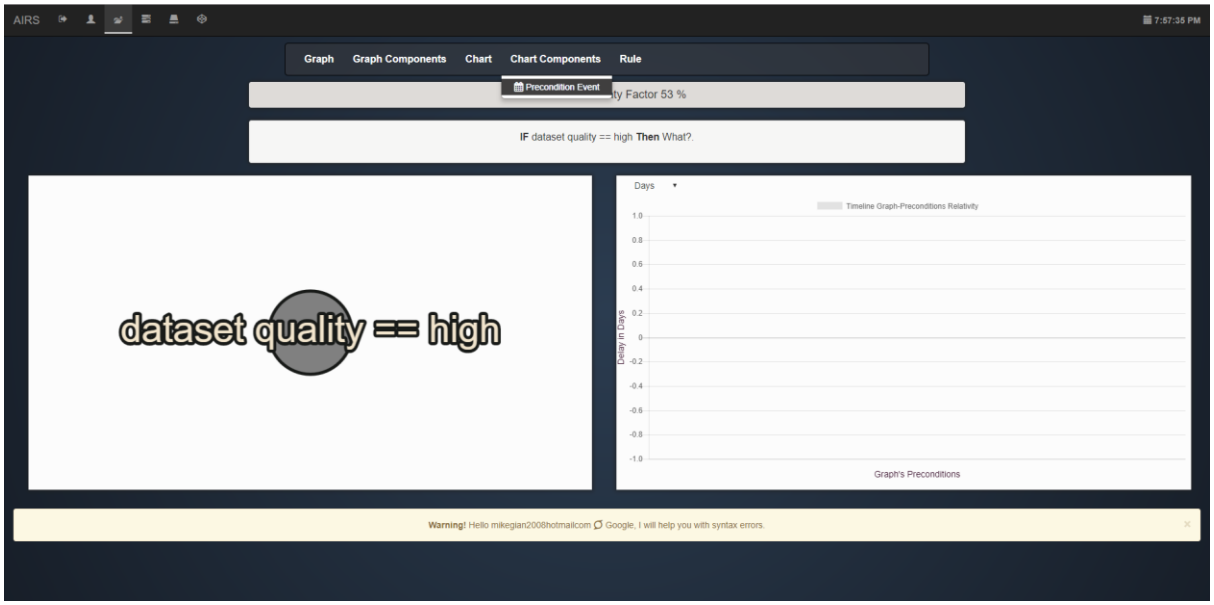

Figure 21 - Επιλογή «Precondition Event» για ορισμό χρονοδιαγράμματος προϋπόθεσης από το μενού ενεργειών.

Επόμενο αλλά μη απαραίτητο βήμα είναι ο χρονικός ορισμός της προϋπόθεσης. Αρχικά επιλέγουμε το «Precondition Event» από τις επιλογές του «Chart Components» του μενού ενεργειών. Με την ενέργεια αυτή θα εμφανιστεί η παρακάτω φόρμα συμπλήρωσης στοιχείων τα οποία θα αποτελέσουν τον πρώτο χρονικό ορισμό προϋπόθεσης του κανόνα.

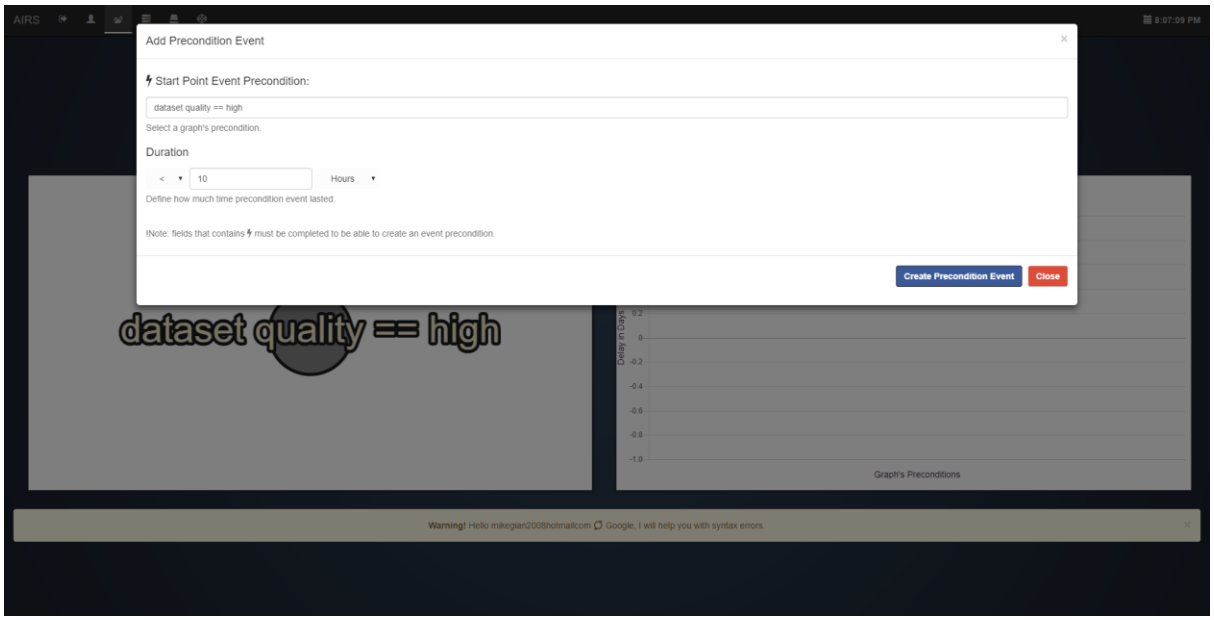

Figure 22 - Φόρμα χρονικού ορισμού προϋπόθεσης και συμπλήρωση πεδίων.

Καθώς το σύστημα γνωρίζει ότι δεν έχουμε ορίσει χρονικά κάποια άλλη προϋπόθεση του κανόνα μας ζητάει προσαρμοσμένα πεδία. Στο πεδίο «Start Point Event Precondition» ορίζουμε την προϋπόθεση ενώ στο «Duration» την χρονική διάρκεια που κράτησε από την στιγμή που ξεκίνησε.

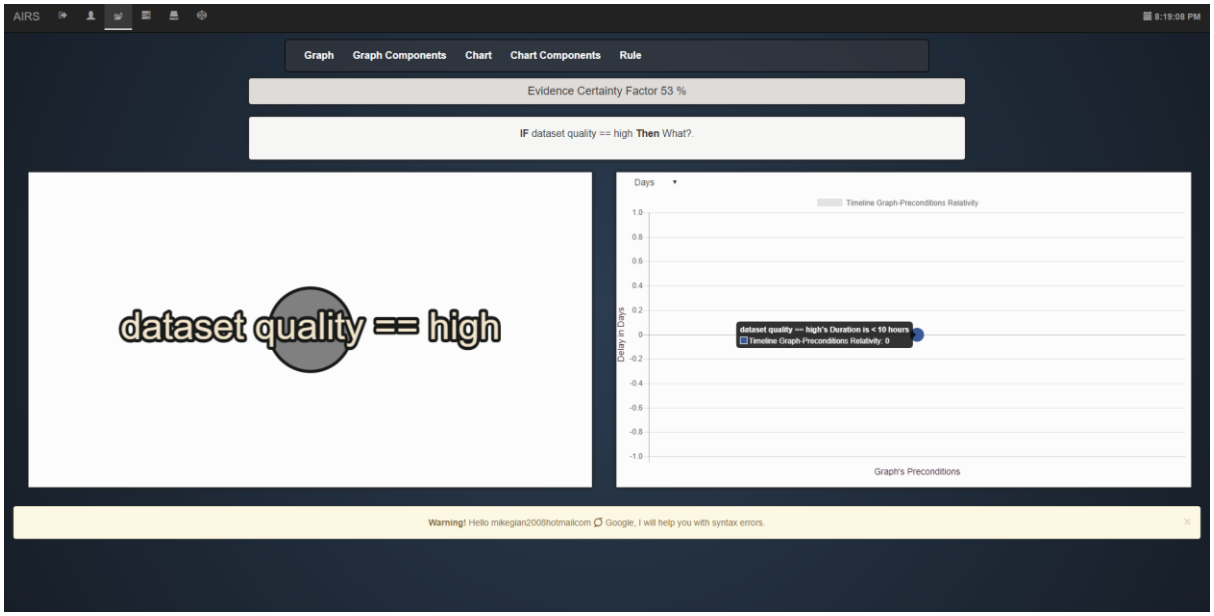

Figure 23 - Αποτέλεσμα χρονικού ορισμού προϋπόθεσης.

Αποτέλεσμα του χρονικού ορισμού της προϋπόθεσης είναι η δημιουργία σημείου πάνω στο χρονοδιάγραμμα. Έως το σημείο αυτό λοιπόν έχουμε ορίσει το εξής: «dataset quality είναι high με παράγοντα βεβαιότητας 53% για λιγότερο από 10 ώρες».

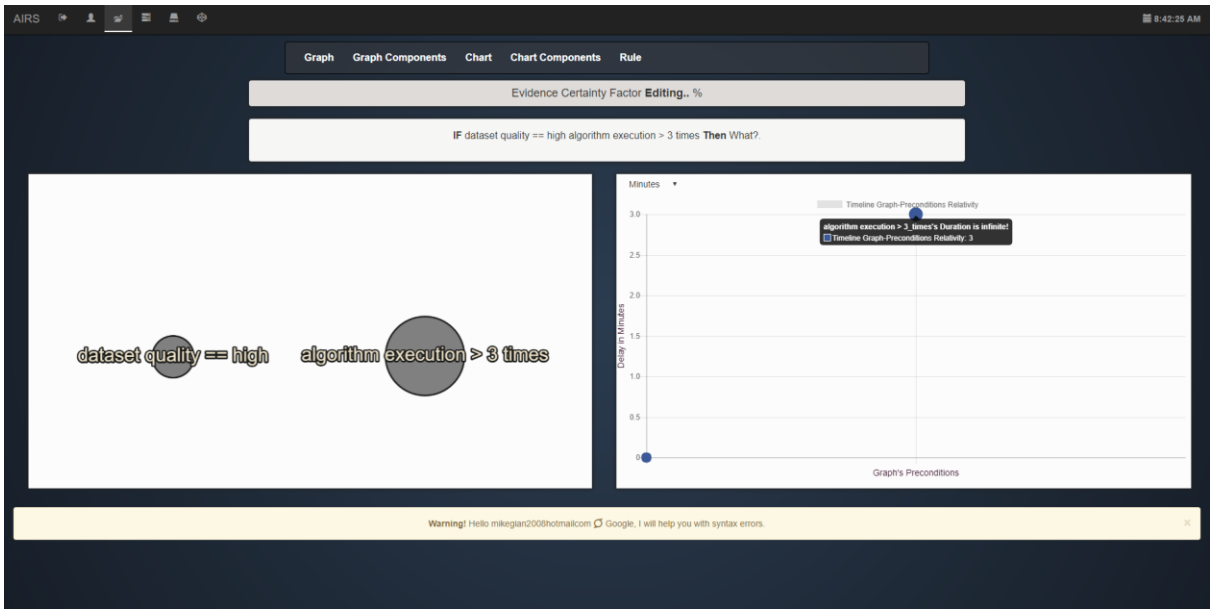

Figure 24 - Προσθήκη νέας προϋπόθεσης και χρονικά ορισμένη ακολουθώντας την ίδια διαδικασία.

Εάν λοιπόν θέλαμε να ορίσουμε την προϋπόθεση «algorithm execution μεγαλύτερο από 3 φορές με παράγοντα βεβαιότητας 100% και ξεκίνησε 3 λεπτά μετά την έναρξη του dataset quality == high» τότε ακολουθώντας την ίδια διαδικασία θα παραγόταν το αποτέλεσμα της εικόνας 23.
Ενώ οι συνθήκες μεμονωμένα είναι σωστές δεν ισχύει το ίδιο ως σύνολο, καθώς δεν υπάρχει λογική διασύνδεση μεταξύ αυτών. Έτσι βλέπουμε ότι ο «Evidence Certainty Factor» είναι σε «Editing..» αφού το σύστημα γνωρίζει ότι ακόμα δεν έχουμε ολοκληρώσει τον κανόνα.

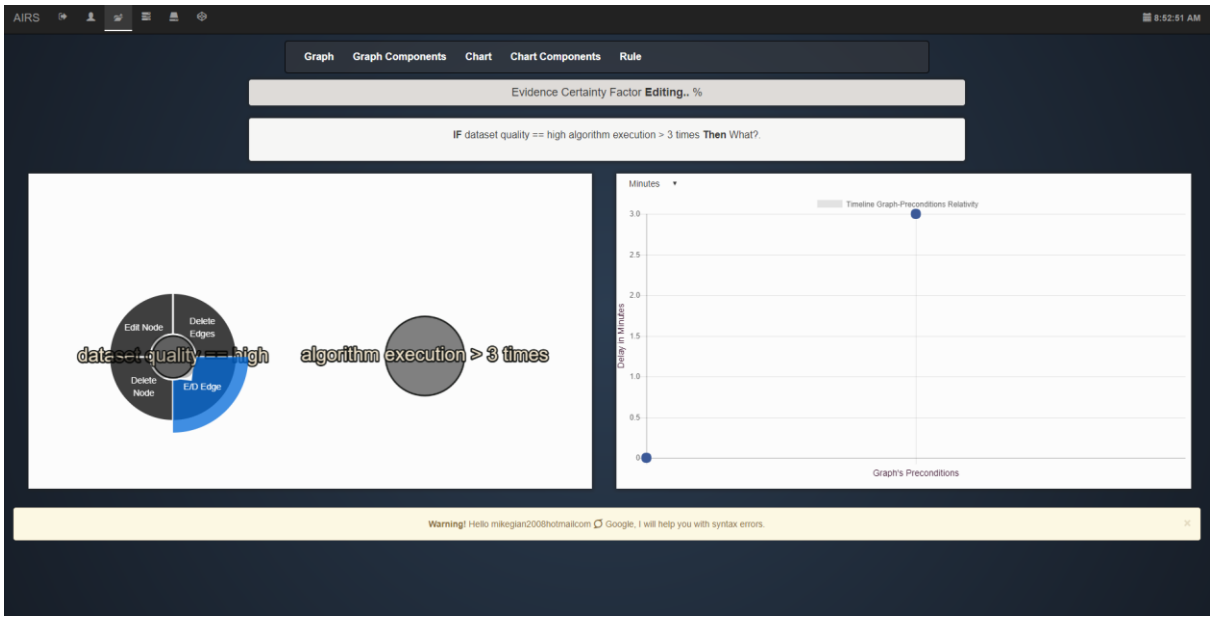

Figure 25 - Ενεργοποίηση λογικής διασύνδεσης προϋποθέσεων μέσα από το Γράφο

Πατάμε λοιπόν «δεξί κλικ» πάνω σε κάποιο κόμβο (προϋπόθεση) του γράφου και επιλέγουμε «E/D Edge». Έπειτα πατώντας «αριστερό κλικ» πάνω σε κάποιο κόμβο και διατηρώντας το πάτημα μετακινούμε τον κέρσορα προς τον κόμβο που θέλουμε να συνδεθεί λογικά μαζί του. Τέλος επιλέγουμε πάνω στην ακμή (λογική διασύνδεση) τον λογικό τελεστή που θέλουμε (AND ή OR) για να καθορίσουμε το είδος της σύνδεσης.

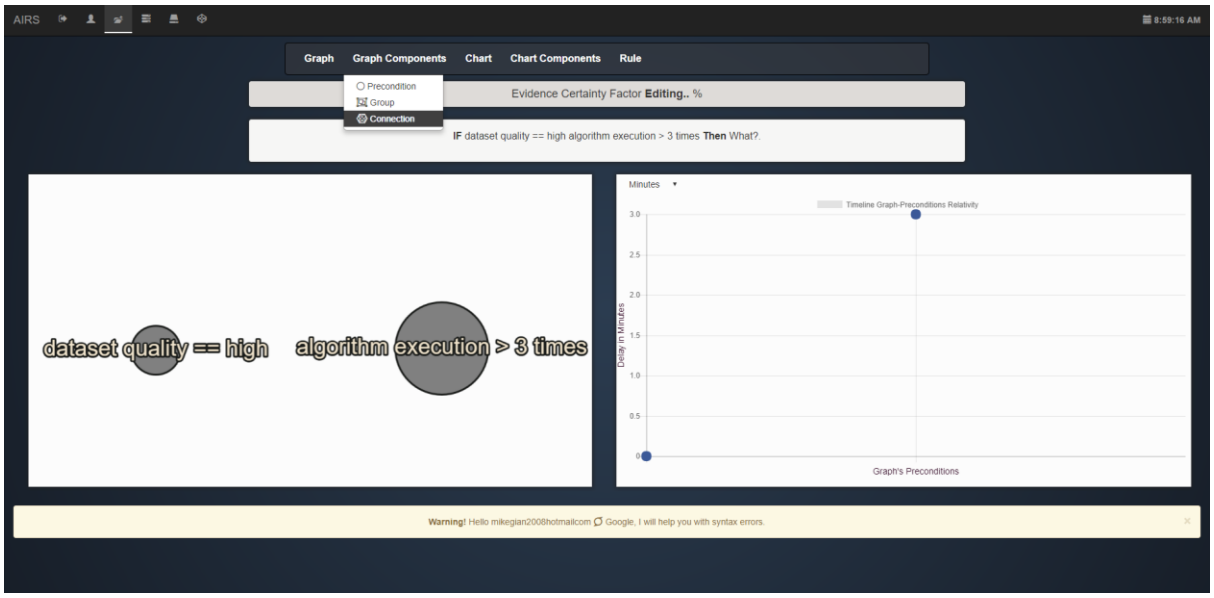

Figure 26 - Δεύτερος τρόπος δημιουργίας λογικής διασύνδεσης προϋποθέσεων

Εναλλακτικά μπορούμε να δημιουργήσουμε λογική διασύνδεση επιλέγοντας «Connection» από το «Graph Components» του μενού ενεργειών. Έτσι θα εμφανιστεί η παρακάτω φόρμα συμπλήρωσης πεδίων.

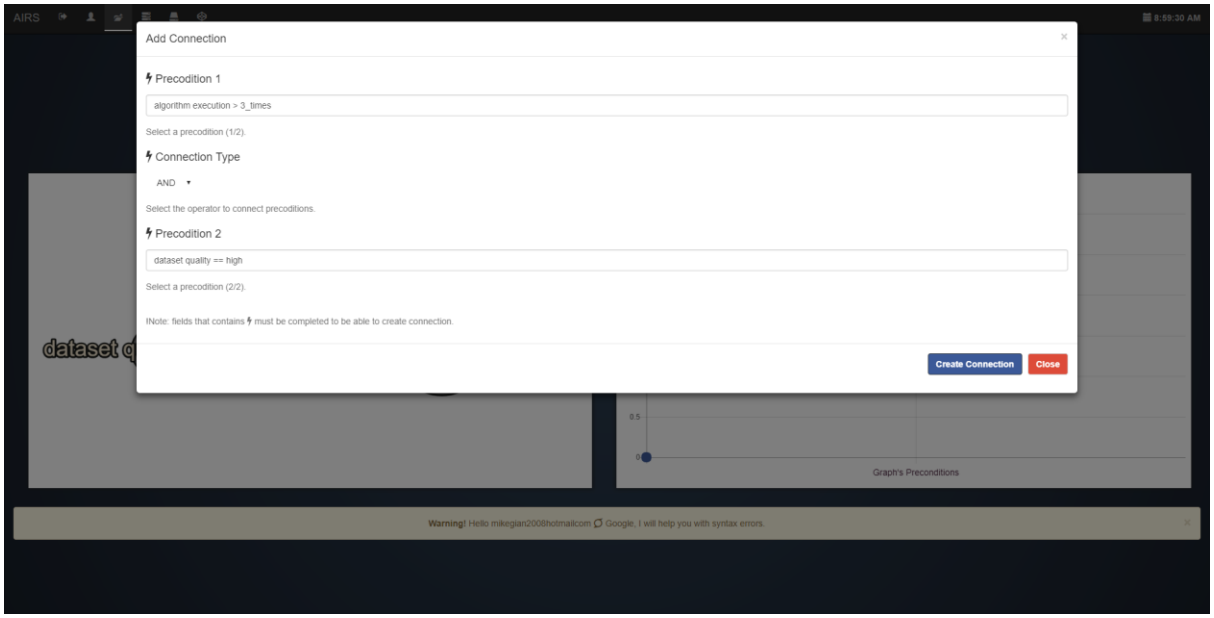

Figure 27 - Φόρμα δημιουργίας λογικής διασύνδεσης με βάση τον δεύτερο τρόπο.

Στη φόρμα αυτή ορίζουμε τις δύο συνθήκες που θέλουμε να συνδεθούν λογικά μεταξύ τους και τον λογικό τελεστή της σύνδεσης αυτής.

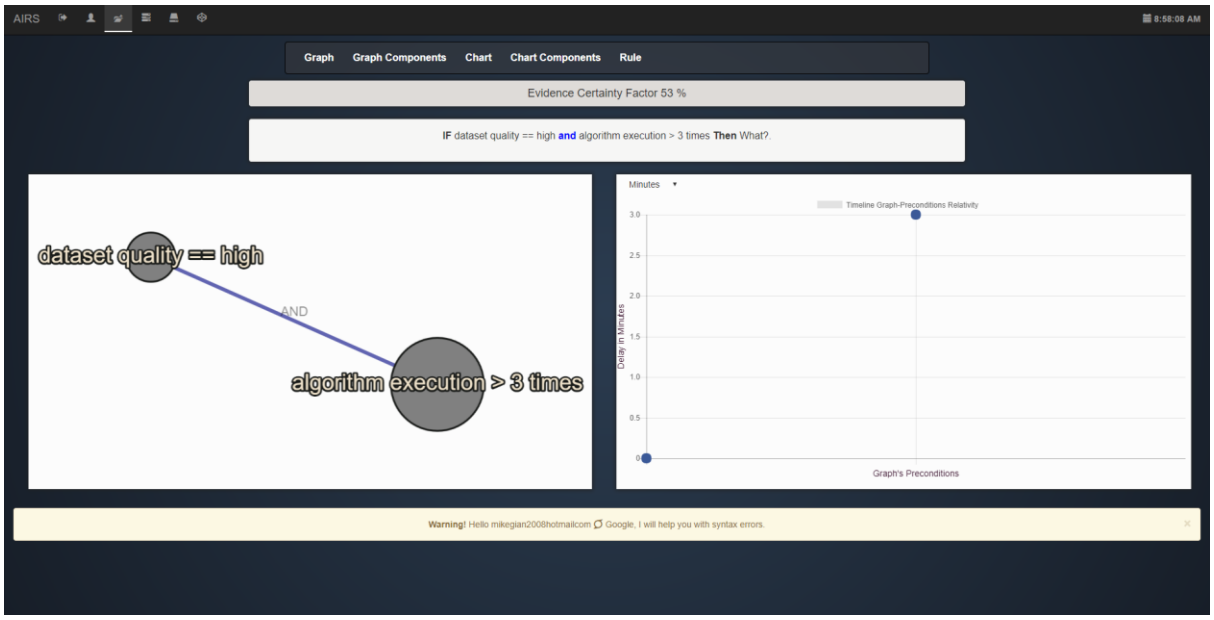

Figure 28 - Αποτέλεσμα λογικής διασύνδεσης (είτε με τον πρώτο είτε με τον δεύτερο τρόπο).

Όποιο από τους δύο τρόπους και να επιλέξουμε, το αποτέλεσμα που θα προκύψει φαίνεται στην εικόνα 27. Στην εικόνα αυτή μπορούμε να δούμε πλέον τις δύο συνθήκες να έχουν επηρεαστεί τόσο πάνω στον γράφο όσο και στην if-then αναπαράσταση ενώ ταυτόχρονα έχει επηρεαστεί και ο παράγοντας βεβαιότητας προϋποθέσεων φτάνοντας το 53%. Εφόσον πλέον είμαστε ικανοποιημένοι με τις συνθήκες του κανόνα, τώρα θα περάσουμε στο αποτέλεσμα αυτού.

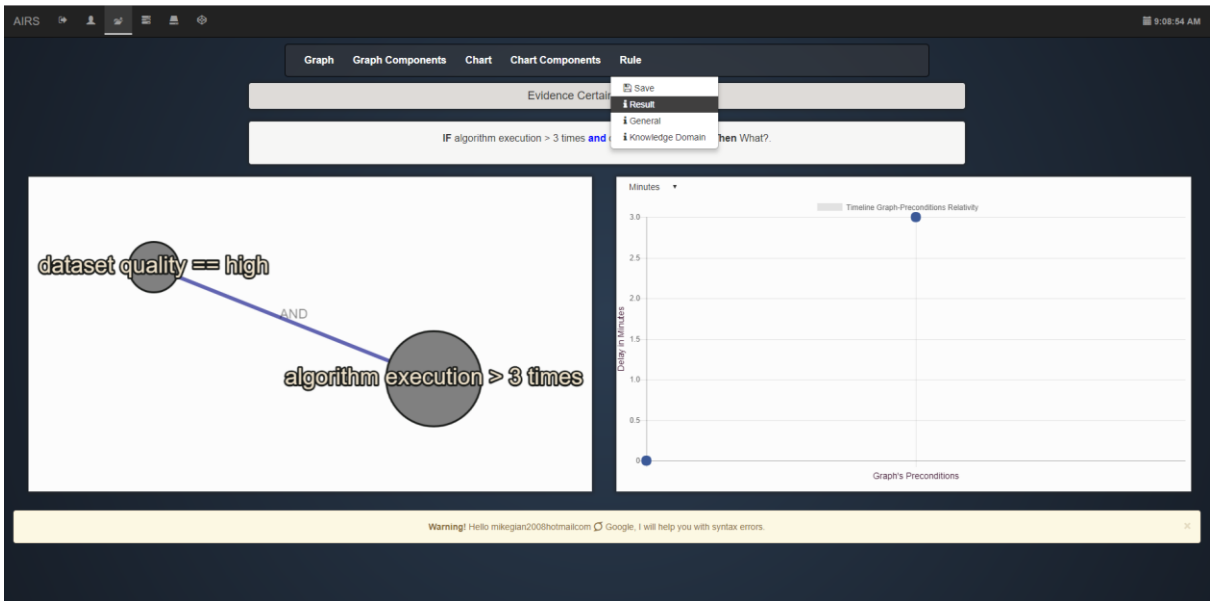

Figure 29 - επιλογή "Result" του "Rule" από μενού ενεργειών.

Επιλέγοντας το «Result» από τις επιλογές του «Rule» στο μενού ενεργειών, θα εμφανιστεί η παρακάτω φόρμα συμπλήρωσης στοιχείων για την δήλωση αποτελέσματος κανόνα.

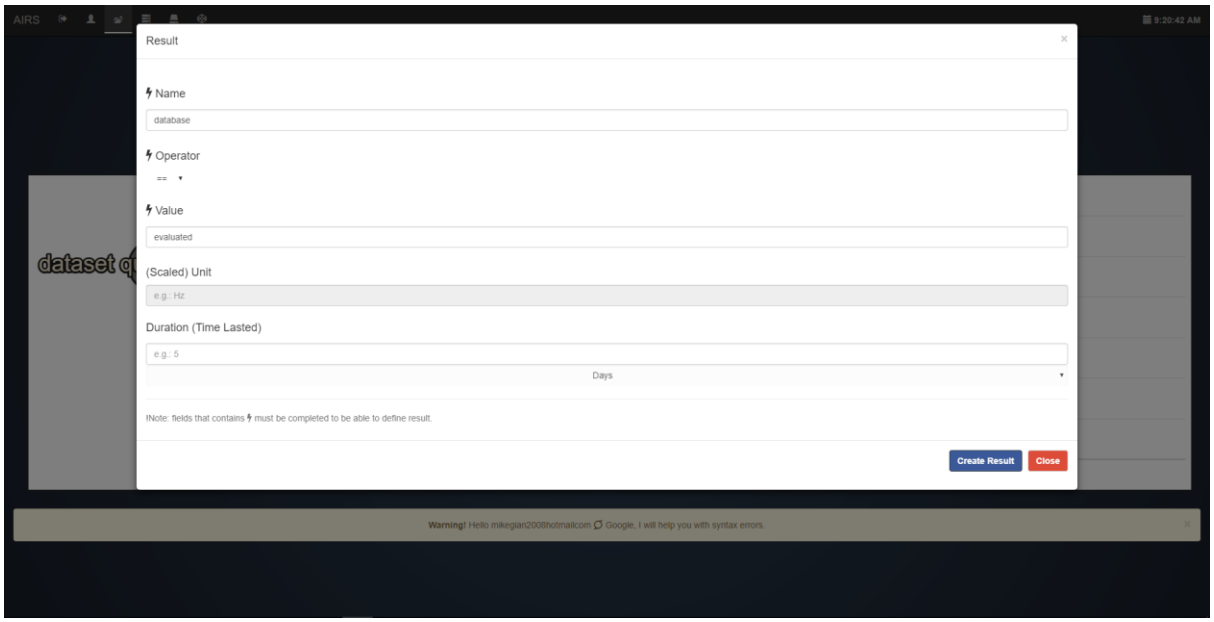

Figure 30 - Φόρμα δημιουργίας αποτελέσματος κανόνα και συμπλήρωση στοιχείων.

Όπως φαίνεται και από τη φόρμα συμπλήρωσης πεδίων της εικόνας 29, το αποτέλεσμα του κανόνα ορίζεται με τον ίδιο τρόπο που ορίζονται οι συνθήκες αυτού. Όπως έχουμε ήδη αναφέρει το σύστημα υποστηρίζει pipelines μεταξύ κανόνων στους συλλογισμούς του, δηλαδή το αποτέλεσμα του κανόνα που ορίσαμε, μπορεί να ορίζεται ως προϋπόθεση σε κάποιο άλλο κανόνα. Η μόνη διαφορά είναι ότι η χρονική διάρκεια του αποτελέσματος μπορεί να οριστεί άμεσα.

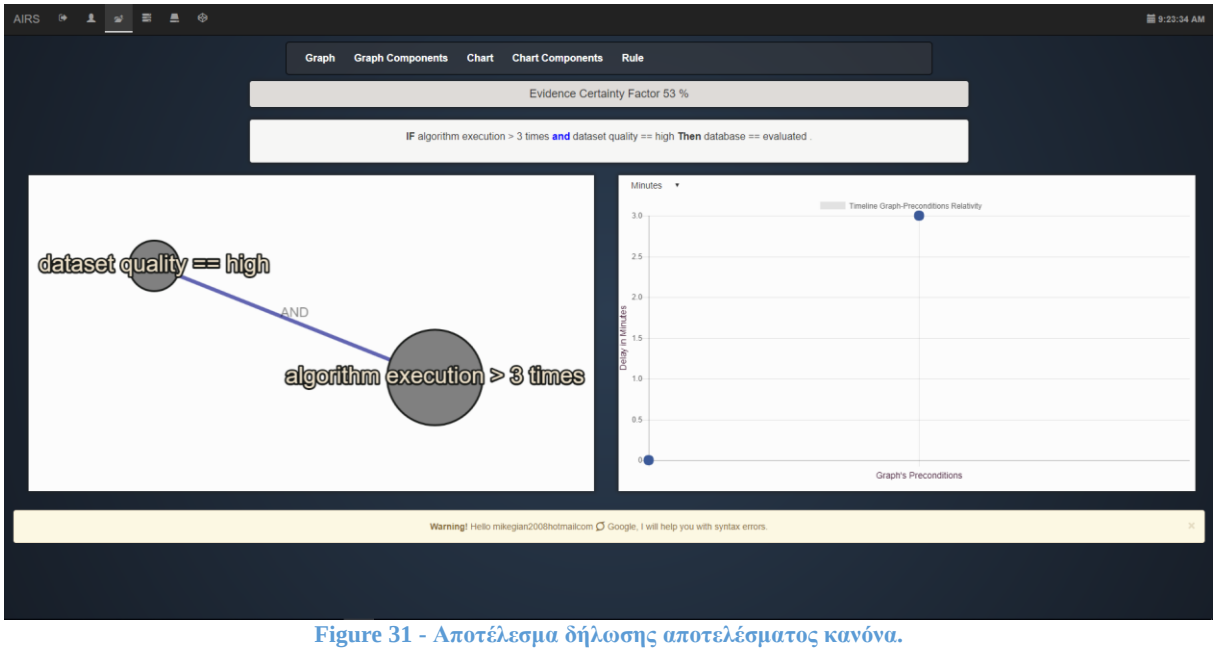

Figure 31 - Αποτέλεσμα δήλωσης αποτελέσματος κανόνα.

Η αναπαράσταση του κανόνα σε μορφή if-then έχει πάρει την τελική της μορφή, όμως χρειαζόμαστε δύο ακόμα ενέργειες για την πλήρη μορφή κανόνα έτσι όπως ορίζεται στο σύστημα. Οι ενέργειες αυτές είναι η δήλωση βάσης γνώσης στην οποία ανήκει ο κανόνας και πληροφορίες του κανόνα ως σύνολο.

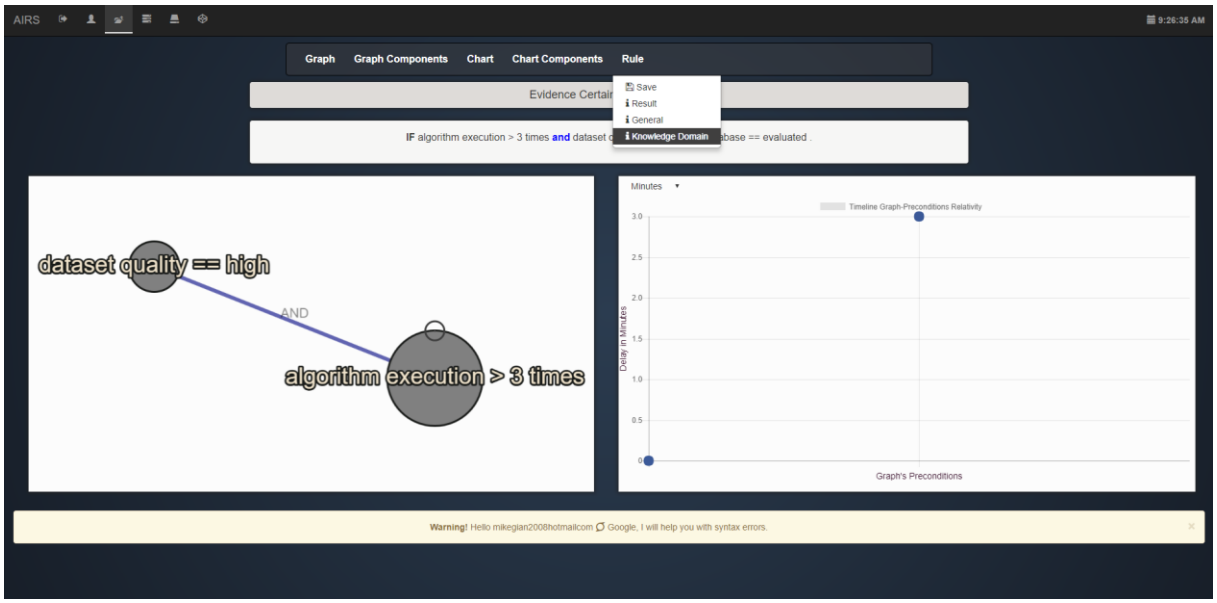

Figure 32 - Επιλογή «Knowledge Domain» από το «Rule» του μενού ενεργειών.

Όσον αφορά τη δήλωση της βάσης γνώσης στην οποία ανήκει ο κανόνας, επιλέγουμε «Knowledge Domain» από το «Rule» στο μενού ενεργειών. Έτσι θα εμφανιστεί η παρακάτω φόρμα προς συμπλήρωση.

Figure 33 – Επιλογή «Data Visualization» ως βάση γνώσης στην οποία ανήκει ο κανόνας.

Στην φόρμα αυτή βλέπουμε ένα πεδίο επιλογής βάσης γνώσης, όπου ο εμπειρογνώμονας επιλέγει το «Data visualization» αντί της προεπιλεγμένης τιμής «Data mining» βάση του πεδίου γνώσης του.

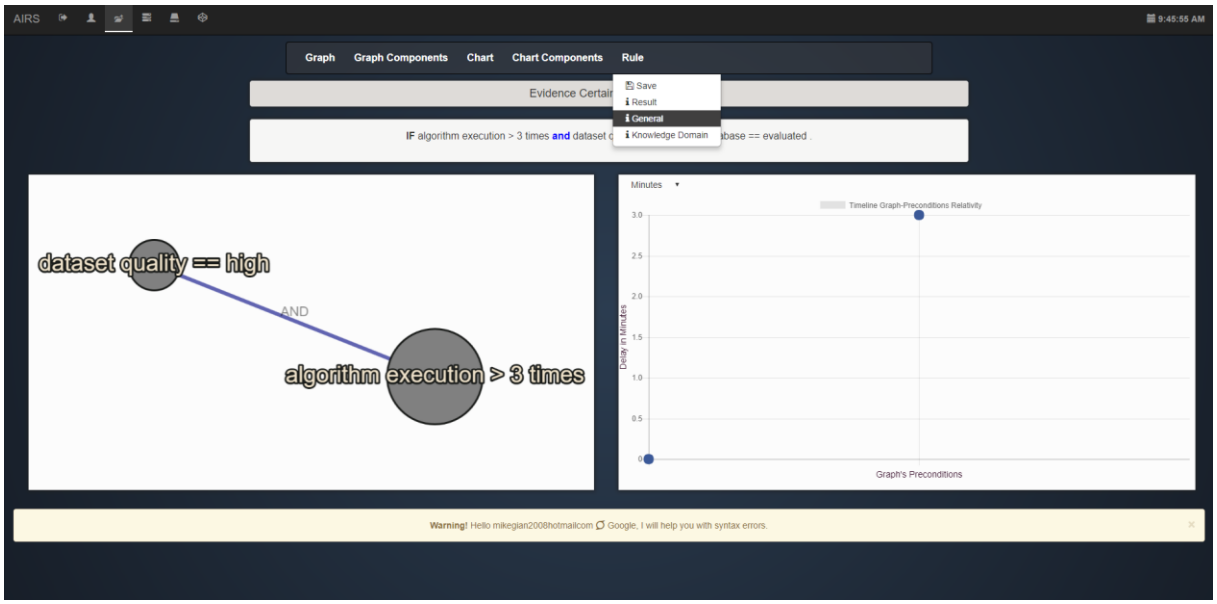

Figure 34 – Επιλογή «General» από το «Rule» του μενού ενεργειών.

Όπως αναφέραμε, η τελευταία ενέργει έχει να κάνει με την δήλωση διαφόρων πληροφοριών του κανόνα στο σύνολο. Για το σκοπό αυτό επιλέγουμε το «General» από το «Rule» του μενού ενεργειών, εμφανίζοντας την παρακάτω φόρμα συμπλήρωσης στοιχείων.

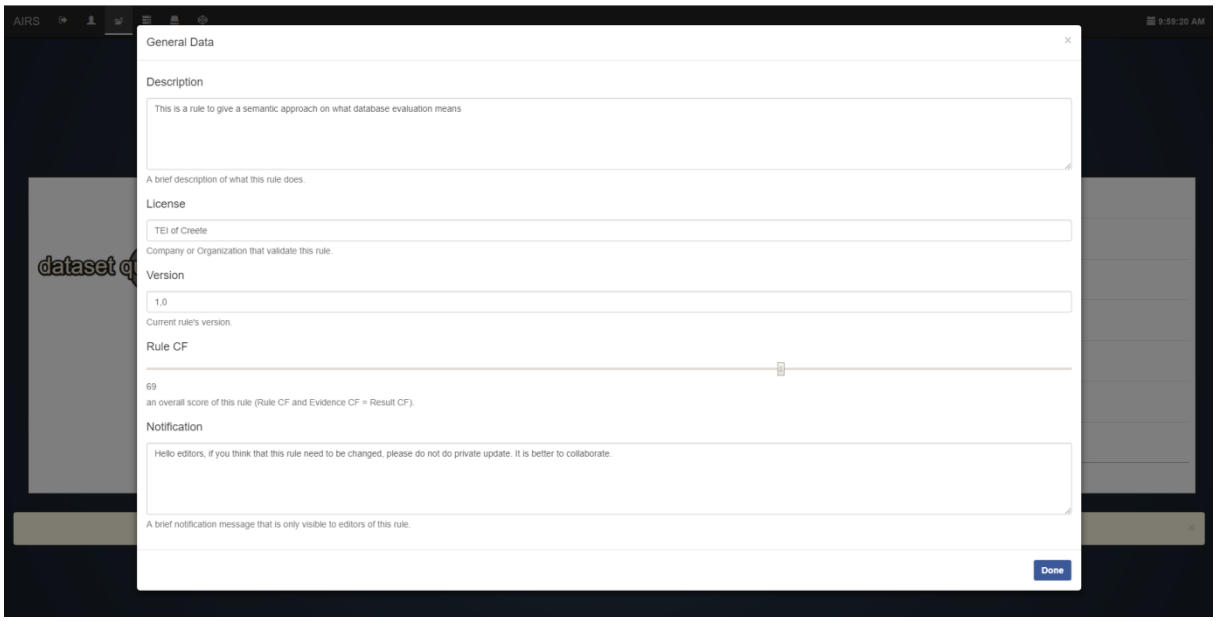

Figure 35 - Φόρμα συμπλήρωσης πληροφοριών του κανόνα ως σύνολο.

Η φόρμα αυτή περιλαμβάνει αρκετά στοιχεία, εκ' των οποίων η συμπλήρωση του «Rule CF» είναι πολύ σημαντική καθώς έχει ρόλο στον παράγοντα βεβαιότητας του αποτελέσματος του κανόνα. Το σύστημα ως προεπιλεγμένη τιμή ορίζει στο 50%. Όσον αφορά τα υπόλοιπα στοιχεία της φόρμας, αν και δεν έχουν ρόλο στην διαδικασία συλλογισμών, είναι σημαντικά για τη σημασιολογική πλευρά του κανόνα ώστε να μπορέσει κάποιος άλλος εμπειρογνώμονας να δει το κανόνα και να καταλάβει άμεσα πάνω σε τι πραγματεύεται «Description», την εγκυρότητα του κανόνα «License», την έκδοση του κανόνα «Version» ενώ μπορεί να συνοδεύεται και από κάποιο σχολιασμό ο οποίος θα είναι διακριτός μόνο προς τον ίδιο και τους συνεργάτες του μέσω του «Notification».

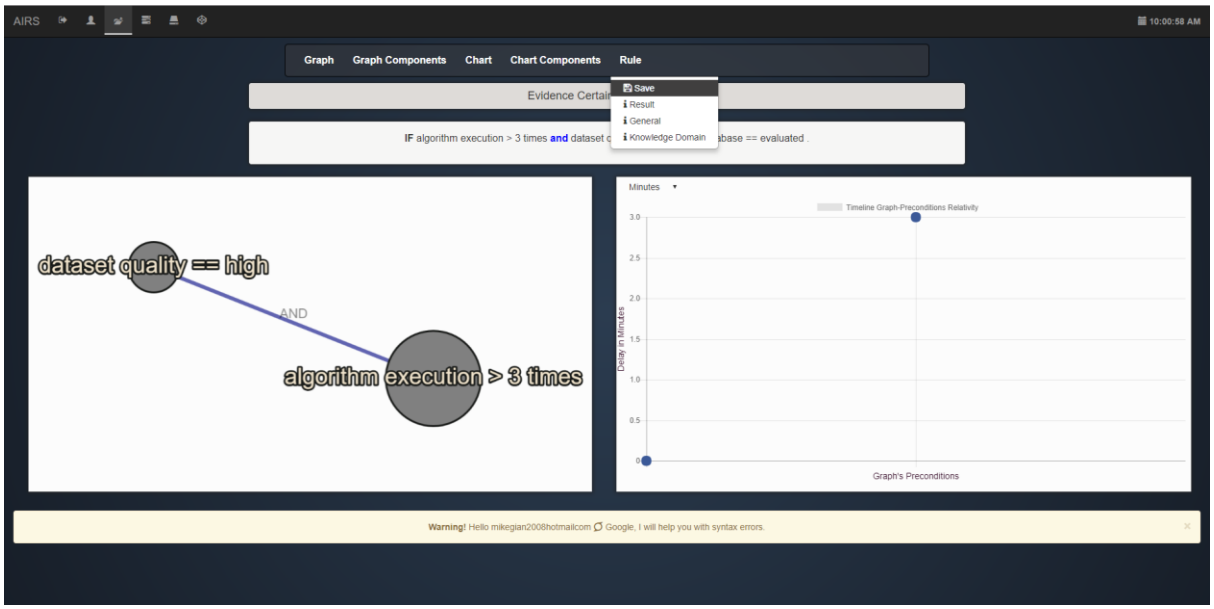

Figure 36 – Επιλογή «Save» από το «Rule» στο μενού ενεργειών.

Πλέον ο κανόνας έχει πάρει την τελική του μορφή. Σειρά λοιπόν έχει η αποθήκευσή του στην βάση γνώσης στην οποία τον ορίσαμε, για να γίνει αυτό επιλέγουμε «Save» από το «Rule» στο μενού ενεργειών.

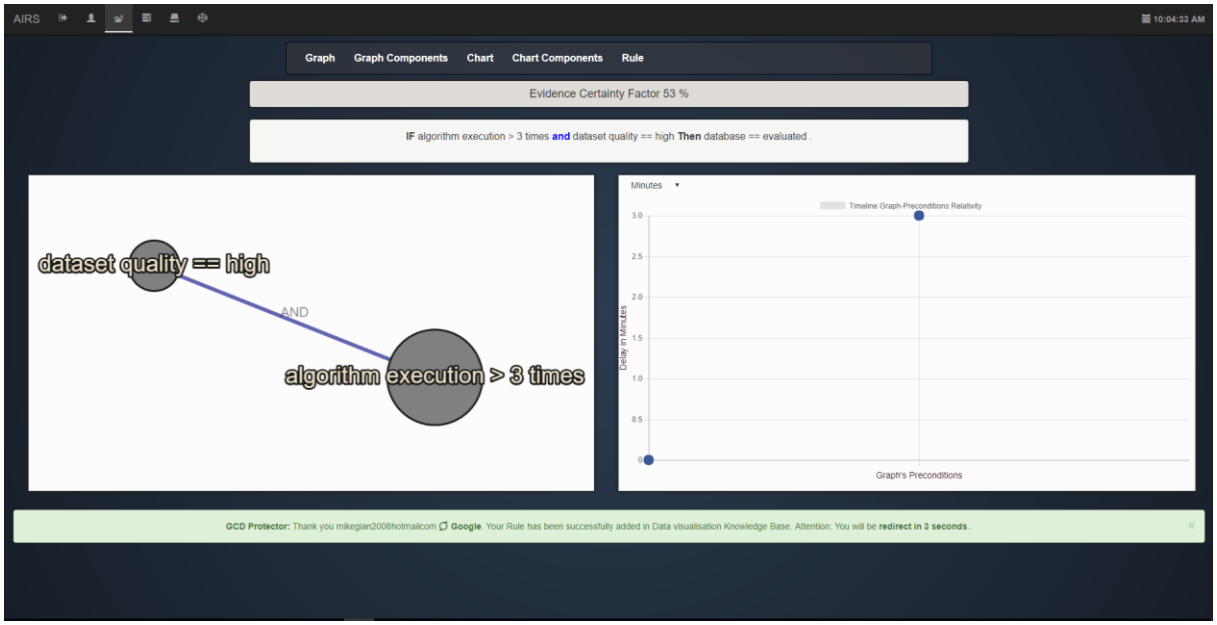

Figure 37 - Αποτέλεσμα επιτυχής αποθήκευσης κανόνα στην βάση γνώσης που επιλέχθηκε.

Τέλος ο «GCD Protector» ενημερώνει τον εμπειρογνώμονα ότι η αποθήκευση του κανόνα πραγματοποιήθηκε με επιτυχία και ότι θα τον μεταφέρει στην προσωπική του βάση κανόνων αυτόματα σε τρία δευτερόλεπτα.

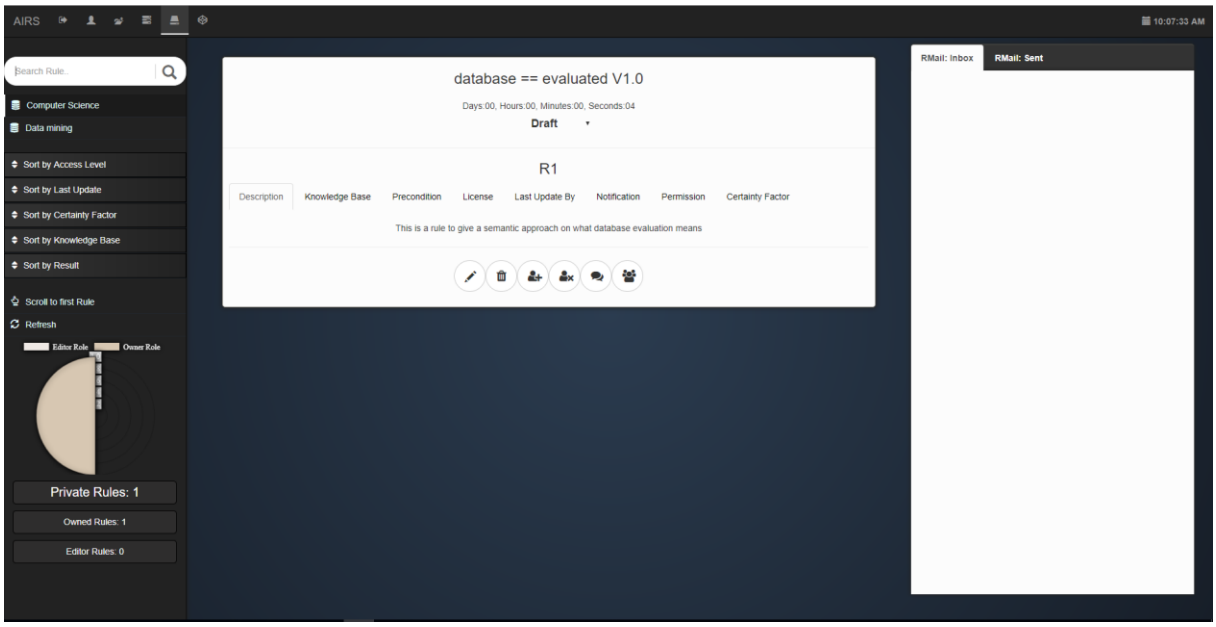

Figure 38 - Προσωπική βάση κανόνων εμπειρογνώμονα με επιλεγμένη βάση γνώσης την «Computer Science».

Στην εικόνα 37 μπορούμε να δούμε την προσωπική βάση κανόνων του εμπειρογνώμονα doctor με προεπιλεγμένη βάση κανόνων παρουσίασης την «Computer Science».
Στην αριστερή πλευρά έχουμε επιλογή προβολής προσωπικής βάσης κανόνων «Computer Science» ή «Data mining», μηχανισμό ταξινόμησης κανόνων παραπλήσιο στο order by και group by της MySQL, διάγραμμα που ενημερώνει ποσοτικά και ποιοτικά στοιχεία των κανόνων που ο εμπειρογνώμονας είναι editor ή owner.
Στο κέντρο είναι όλοι οι προσωπικοί κανόνες (του Computer Science στο στιγμιότυπο αυτό).
Στη δεξιά πλευρά είναι το «R-Mail» μηχανισμός που περιλαμβάνει όλα τα μηνύματα (inbox/sent) που αφορούν κανόνες που είμαστε owner.

5.3. Αίτημα Συνεργασίας πάνω στο κανόνα αυτό από εμπειρογνώμονα και αποδοχή του αιτήματος αυτού.

Για να αναδείξουμε την συνεργασία μεταξύ δύο εμπειρογνωμόνων διαφορετικών πλατφορμών θα δημιουργήσουμε δύο λογαριασμούς εμπειρογνωμόνων, ο ένας θα είναι μέσω του iPHR και ο άλλος θα είναι μέσω της Google, καθώς για να μπορούν να συνεργαστούν πρέπει να έχουν ίδια ρίζα πεδίου γνώσης. Αφού το iPHR υποστηρίζει μόνο «Medicine» ενώ η Google υποστηρίζει και «Medicine», η συνεργασία μεταξύ τους μπορεί να γίνει με ρίζα το «Medicine». Έστω λοιπόν ότι έχουμε δημιουργήσει λογαριασμό ογκολόγου με «Expert_ID: doctor» μέσω της πλατφόρμας iPHR ο οποίος είναι κάτοχος κανόνα και ένα λογαριασμό δερματολόγου μέσω της πλατφόρμας Google με «Expert_ID: mikegian2008hotmailcom».

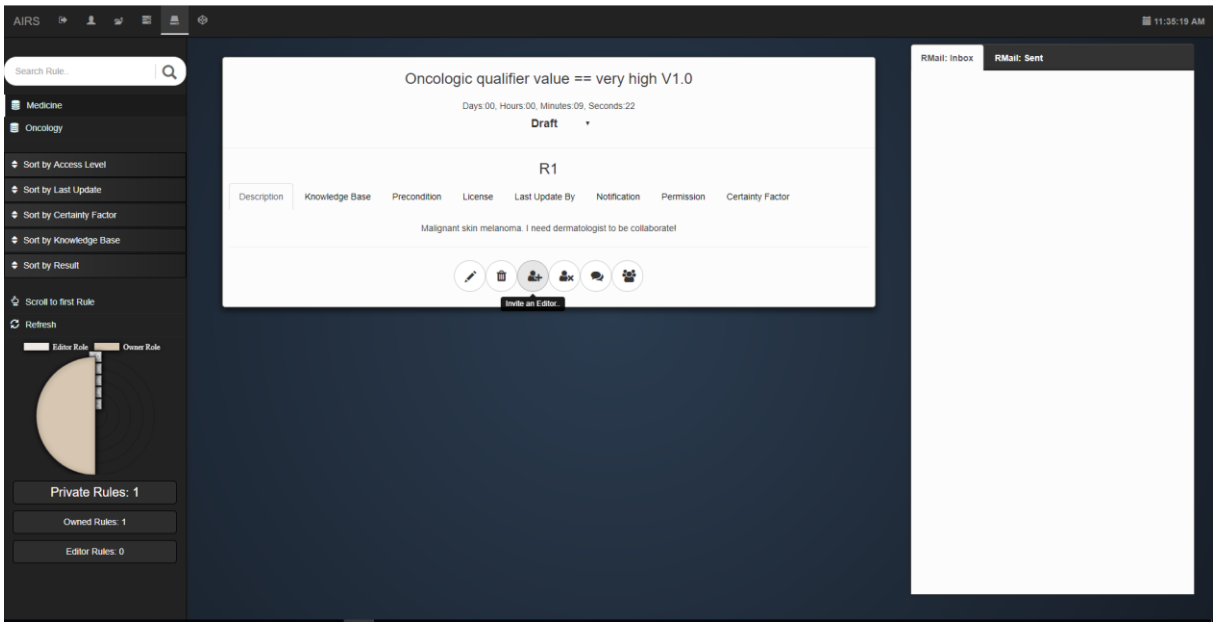

Figure 39 - Πάτημα κουμπιού για κάλεσμα συνεργασίας κάποιου εμπειρογνώμονα στον κανόνα R1.

Ο doctor μέσω της προσωπικής του βάσης γνώσης μπορεί να πατήσει το εικονίδιο με tooltip «Invite an Editor..» πάνω στον κανόνα που επιθυμεί και να εντοπίσει κάποιον εμπειρογνώμονα του συστήματος ώστε να του στείλει αίτημα συνεργασίας πάνω στον κανόνα αυτό.

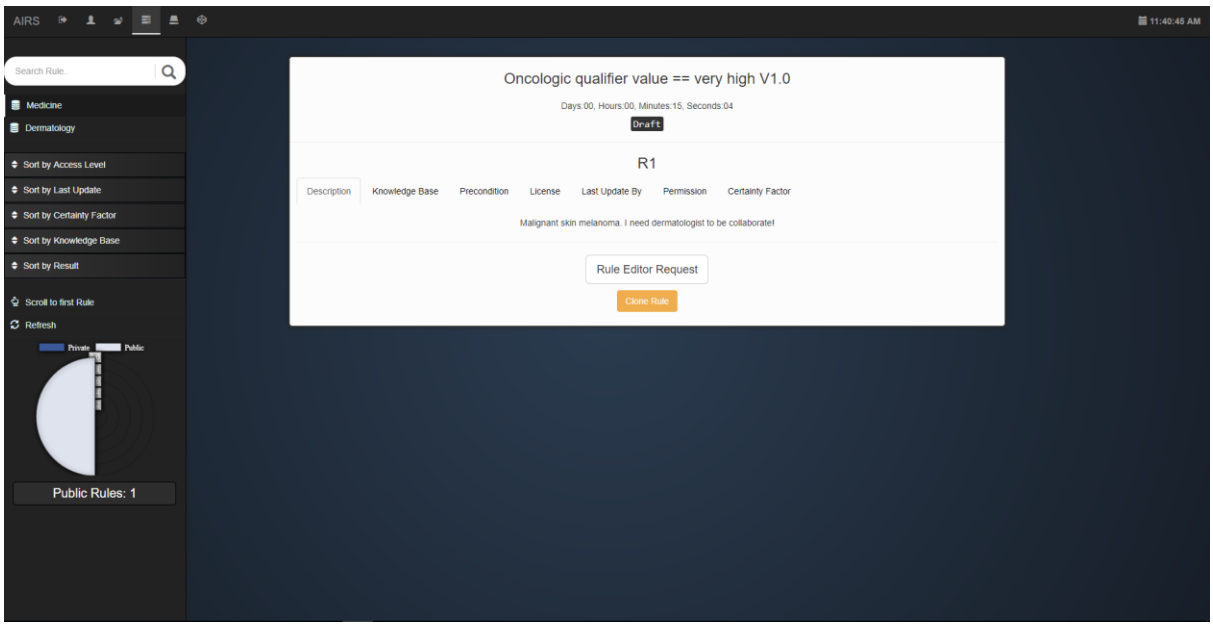

Figure 40 - Πάτημα κουμπιού Rule Editor Request, για αποστολή αιτήματος στον κανόνα R1 για συνεργασία.

Αντίστοιχα ο mikegian2008hotmailcom μπορεί να πάει στη δημόσια βάση κανόνων και να εντοπίσει τον κανόνα του doctor. Εκεί μπορεί να πατήσει «Rule Editor Request» για τη δημιουργία και αποστολή αιτήματος στον δημιουργό του κανόνα «R1» (doctor), για συνεργασία πάνω στον κανόνα αυτό.

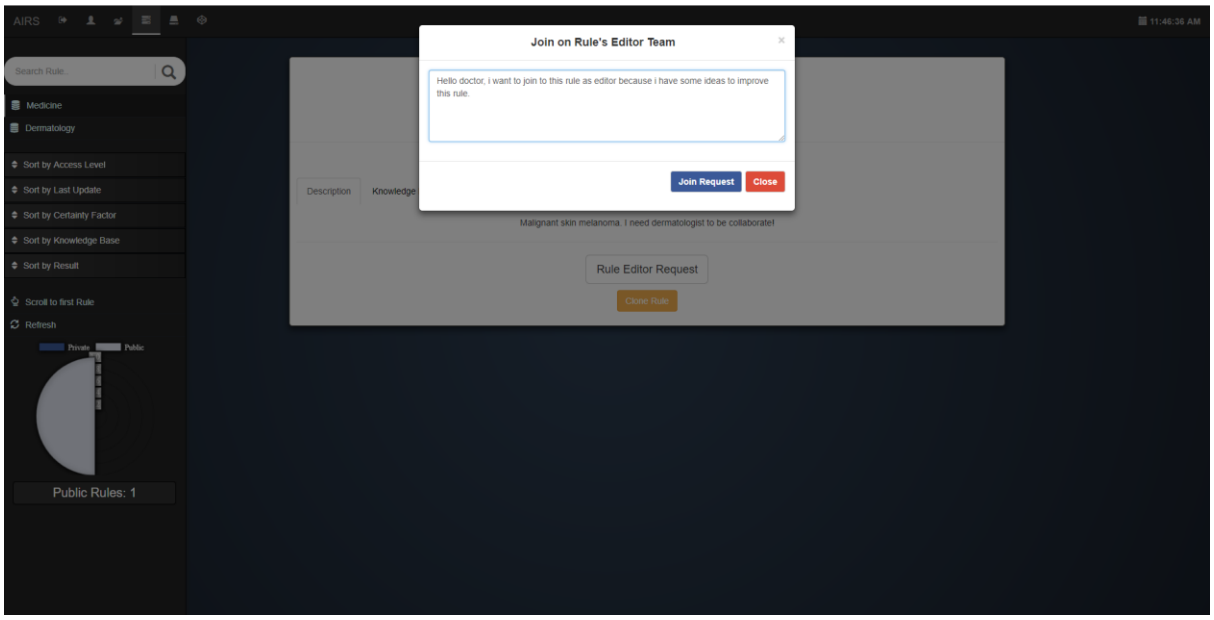

Figure 41 - Σύνταξη αιτήματος για συνεργασία στον κανόνα R1.

Η σύνθεση αιτήματος μπορεί να περιλαμβάνει συγκεκριμένο μήνυμα που θέλει ο αιτών εμπειρογνώμονας ή το default μήνυμα του συστήματος.

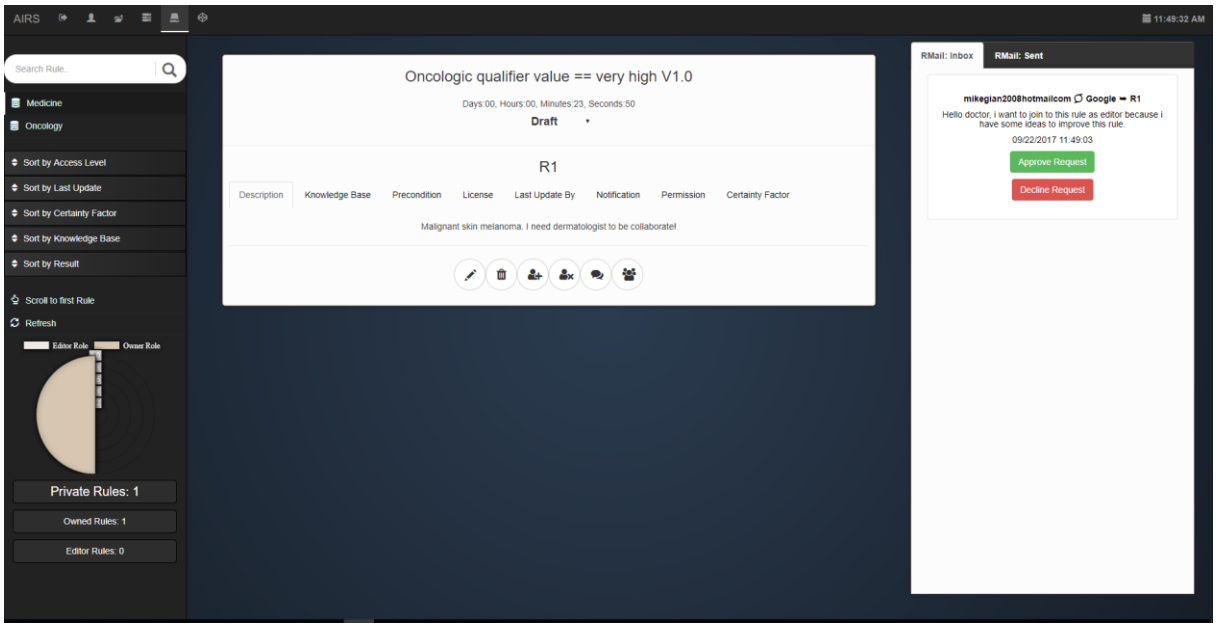

Figure 42 - Εισερχόμενο μήνυμα στην προσωπική βάση κανόνων του doctor, με αποστολέα τον δερματολόγο mikegian2008hotmailcom για συνεργασία πάνω στον κανόνα R1.

Ο doctor λαμβάνει λοιπόν ένα μήνυμα από τον mikegian2008hotmailcom για συνεργασία πάνω στον κανόνα R1 στην προσωπική βάση κανόνων, όπου πατώντας «Approve Request» μπορεί να κάνει αποδοχή του αιτήματος αυτού.

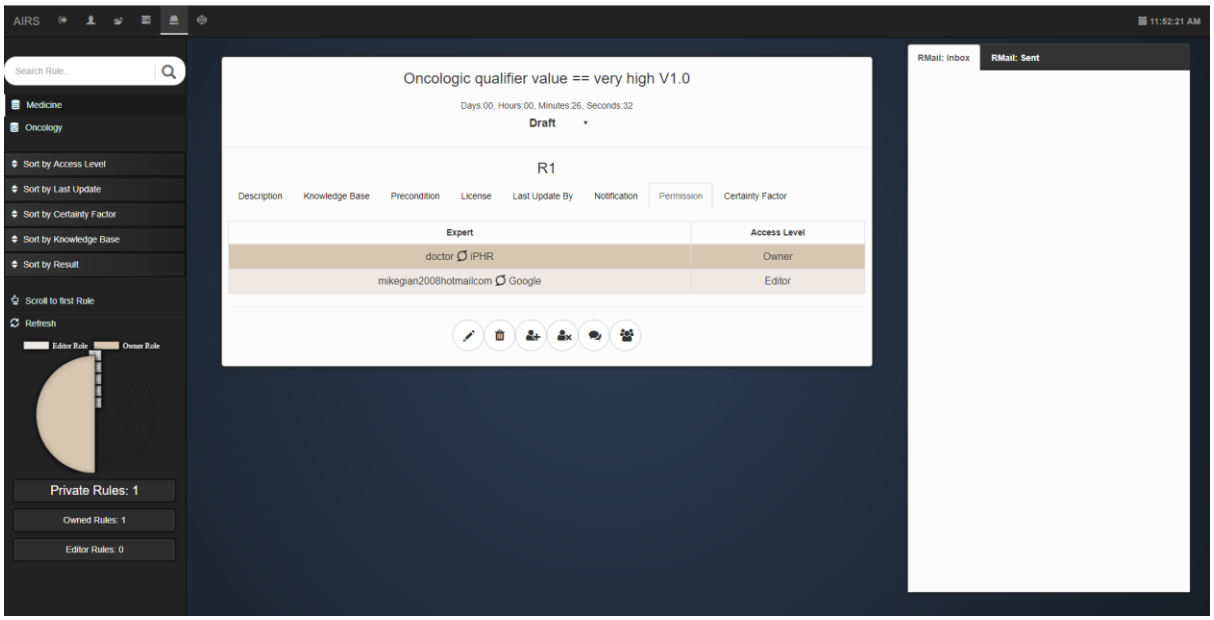

Figure 43 - Προβολή προσωπικής βάσης κανόνων του doctor με εστίαση στο πεδίο «Permission» του κανόνα R1.

Έτσι οι δύο αυτού εμπειρογνώμονες έγιναν συνεργάτες πάνω στον συγκεκριμένο κανόνα, με τον doctor στον ρόλο του «Owner» και τον mikegian2008hotmailcom στον ρόλο του «Editor».

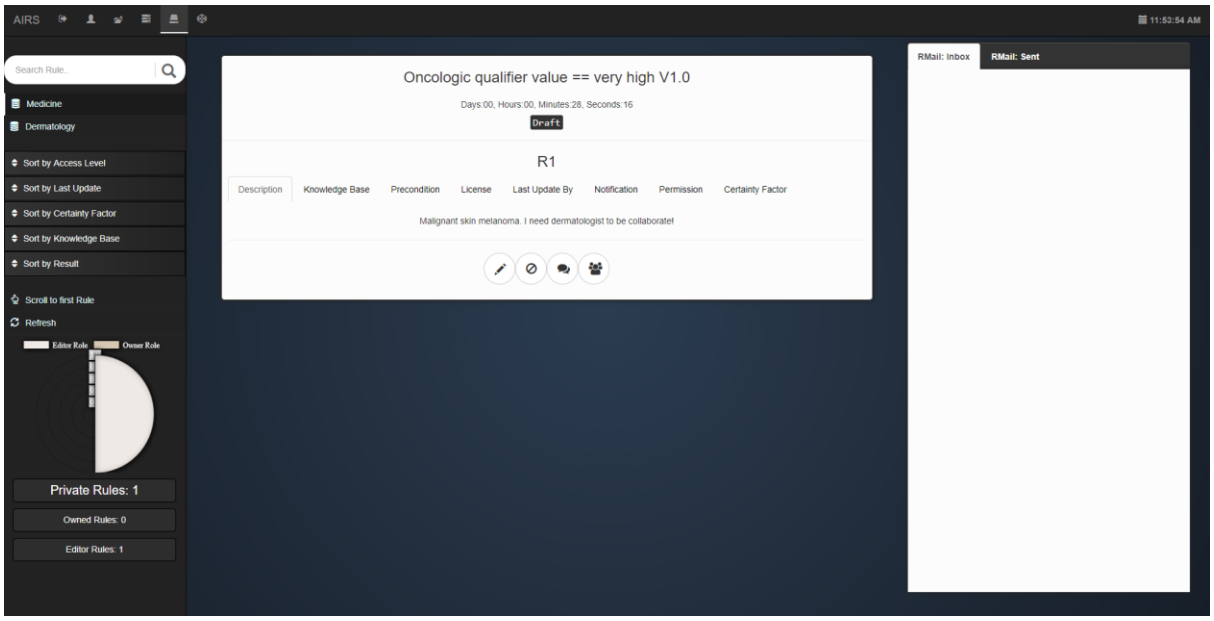

Figure 44 - Προβολή προσωπικής βάσης κανόνων του mikegian2008hotmailcom.

Τέλος ο κανόνας αυτός εμφανίζεται στην προσωπική βάση κανόνων του mikegian2008hotmailcom έχοντας πλέον δυνατότητες «Editor» πάνω σε αυτόν.

5.4. Συνεργατική ενημέρωση του κανόνα.

Έχοντας τον κανόνα της προηγούμενης ενότητας στον οποίο συνεργάτες είναι ο ογκολόγος doctor της πλατφόρμας iPHR και ο δερματολόγος mikegian2008hotmailcom της πλατφόρμας Google. Έστω ότι οι δύο αυτοί εμπειρογνώμονες επιλέγουν την λειτουργία «Collaborative Update» που βρίσκεται πάνω στην παρουσίαση του κανόνα στις προσωπικές τους βάσεις γνώσεις.

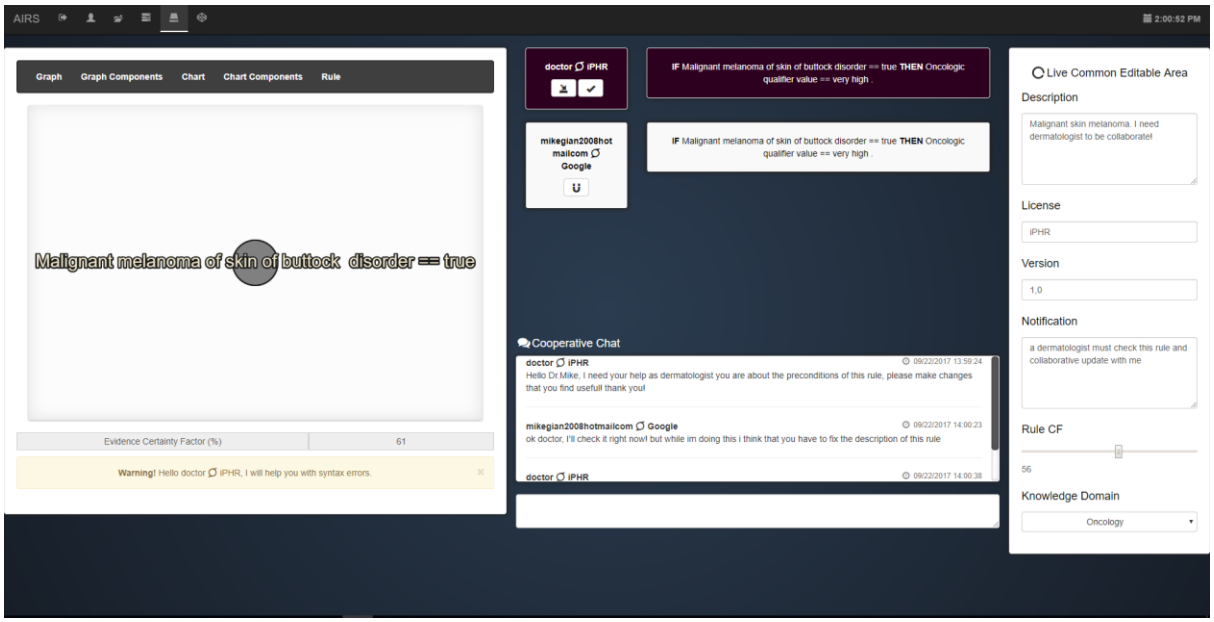

Figure 45 - Συνεργατική Επεξεργασία κανόνα R1, με τρέχον εμπειρογνώμονα τον doctor.

Έπειτα μεταφέρονται σε μία νέα σελίδα, η οποία περιλαμβάνει όλες τις λειτουργίες για αποτελεσματική συνεργασία μεταξύ τους. Στην εικόνα 44 παρουσιάζονται οι λειτουργίες αυτές, έτσι όπως εμφανίζονται στον doctor, χωρίζοντας την οθόνη σε τρία μέρη. Το κέντρο αποτελείται από συστατικά προβολής πραγματικού χρόνου, εκεί βρίσκονται όλοι οι συνεργάτες μαζί με τις δικές τους εκδοχές του ίδιου κανόνα σε if-then αναπαράσταση, όπου με σκούρο κόκκινο φαίνεται ο κανόνας του τρέχον εμπειρογνώμονα. Ακριβώς από κάτω βρίσκεται το «Cooperative Chat» μέσω του οποίου μπορούν να συνομιλήσουν. Συγκεκριμένα εδώ ο doctor ζητάει από τον mikegian2008hotmailcom βελτίωση των προϋποθέσεων του κανόνα. Η δεξιά πλευρά αποτελείται από ένα σύνολο κοινών επεξεργαζόμενων πεδίων σε πραγματικό χρόνο. Τα πεδία αυτά αναφέρονται στα γενικά στοιχεία που συνοδεύουν τον κανόνα όπως επίσης και στον ορισμό του πεδίου γνώσης που βρίσκεται ο κανόνας. Η αριστερή πλευρά παρέχει όλες τις λειτουργίες για επεξεργασία κανόνα, με κάθε ενέργεια του εμπειρογνώμονα στην πλευρά αυτή να μην είναι ορατή εκεί αλλά το αποτέλεσμά της συγχρονίζεται με την «if-then» αναπαράσταση του κέντρου.

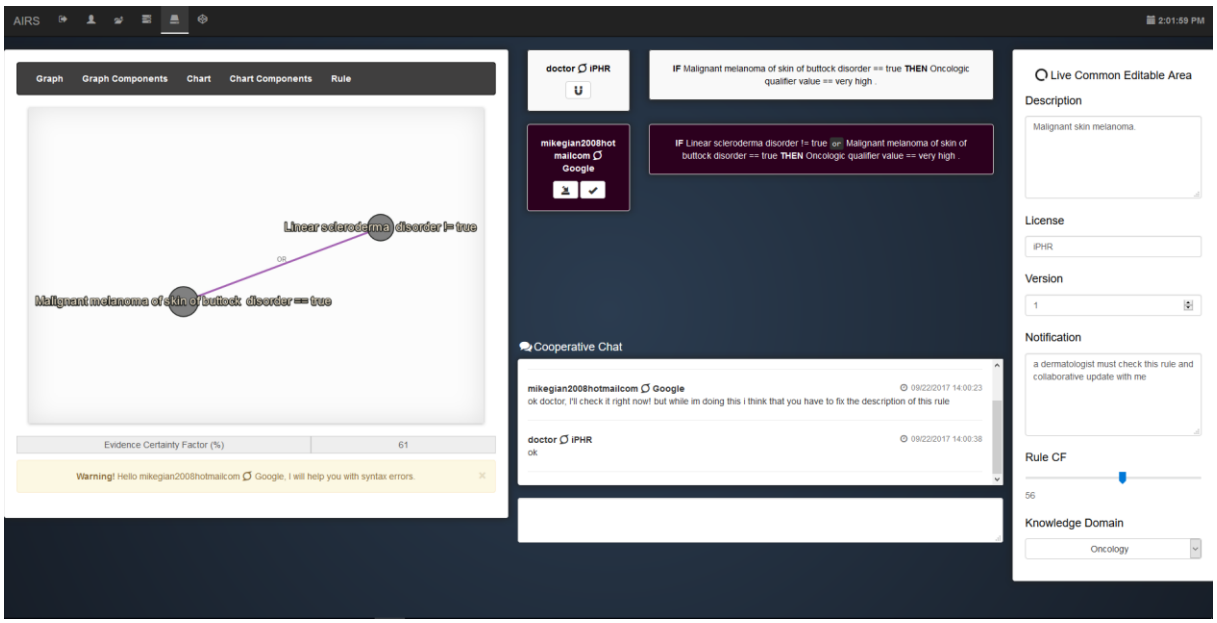

Figure 46 - Συνεργατική επεξεργασία κανόνα με τρέχον εμπειρογνώμονα τον mikegian2008hotmailcom.

Στην εικόνα 45 παρουσιάζονται οι λειτουργίες με τρέχον εμπειρογνώμονα τον mikegian2008hotmailcom, ο οποίος παραμετροποίησε τις συνθήκες στην δική του εκδοχή κανόνα. Συνεπώς οι αλλαγές που έκανε αυτός πάνω στο Γράφο του, συγχρονίζονται αυτόματα με την δική του «if-then» αναπαράσταση που είναι ορατή προς όλους τους συνεργάτες. Ενώ ταυτόχρονα ο doctor άλλαξε το «Description» του κανόνα.

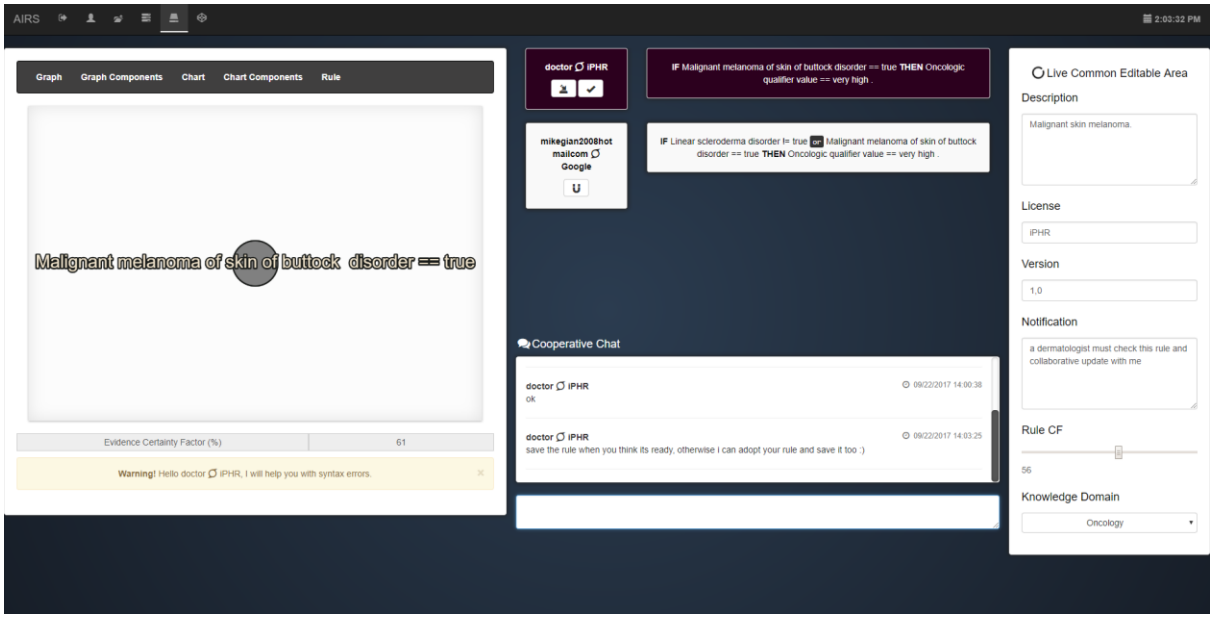

Figure 47 - Αποστολή μηνύματος από doctor μέσω του «Cooperative Chat».

Στη συνέχεια ο (τρέχον) εμπειρογνώμονας doctor ζητάει από τον mikegian2008hotmailcom να αποθηκεύσει τον κανόνα εάν νομίζει ότι οι συνθήκες είναι εντάξει.

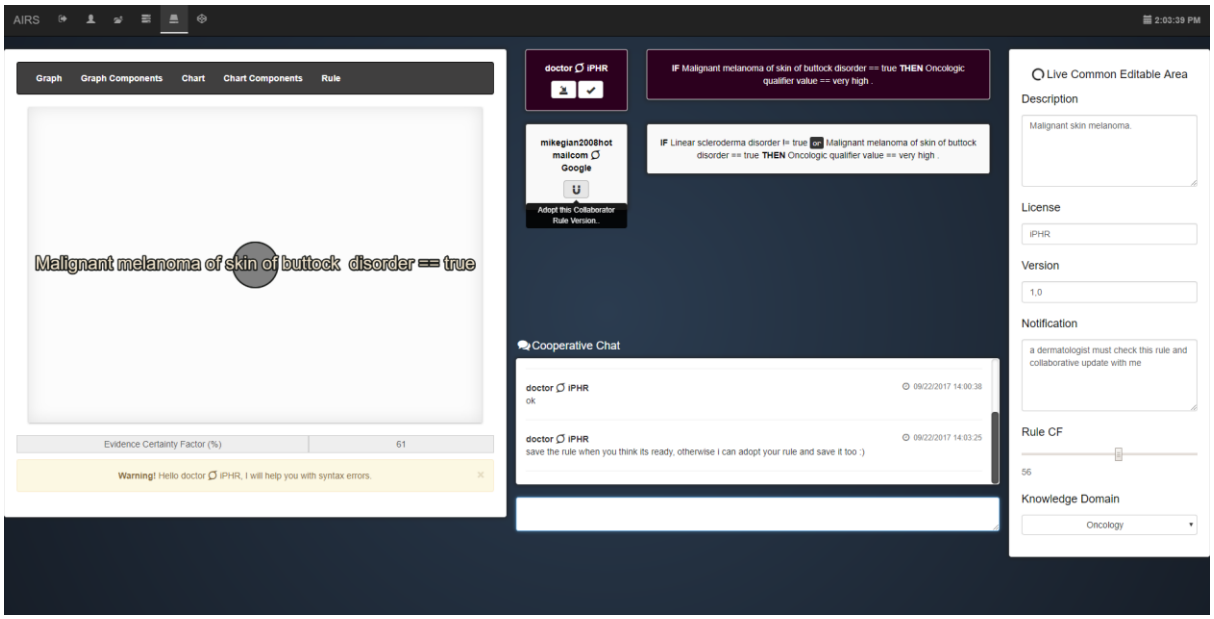

Figure 48 - Παρουσίαση δυνατότητας υιοθέτησης ανά πάσα στιγμή έκδοσης κανόνα πατώντας το κουμπί με τον «μαγνήτη σύμβολο».

Βέβαια όπως φαίνεται στην εικόνα 47, θα μπορούσε ο ίδιος οποιαδήποτε στιγμή να πατήσει το κουμπί του εικονιδίου «μαγνήτη» κάτω από το όνομα του mikegian2008hotmailcom ώστε να υιοθετήσει τον κανόνα του και να τον κάνει save ο ίδιος.

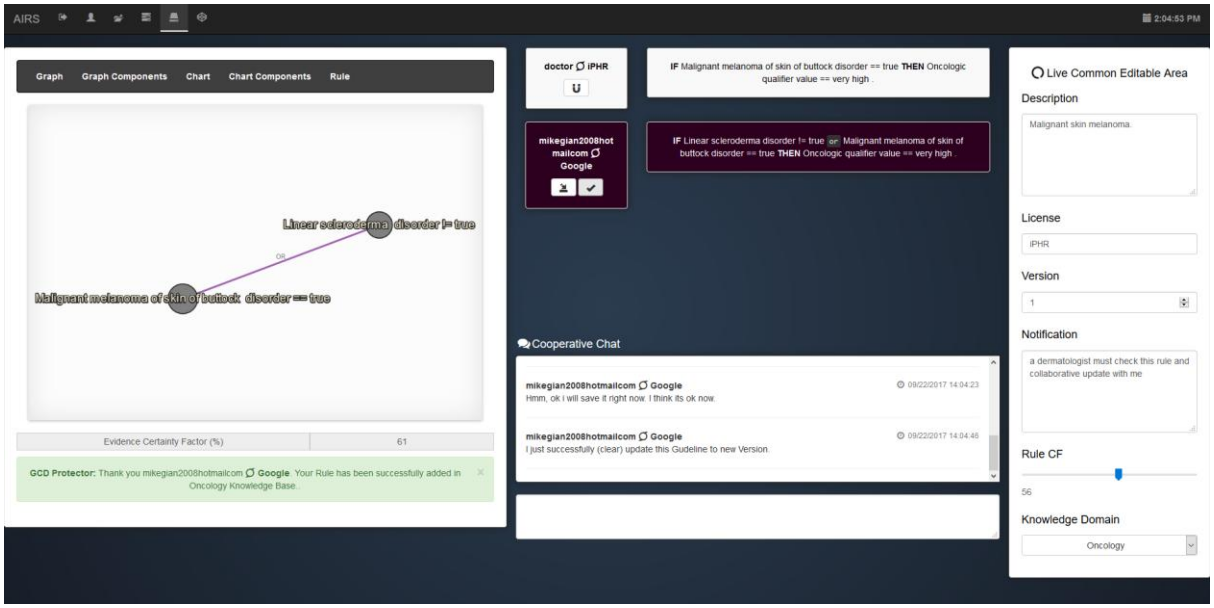

Figure 49 - Πάτημα κουμπιού με το «check σύμβολο» για αποθήκευση τρέχουσας έκδοσης του mikegian2008hotmailcom και των στοιχείων του «Live Common Editable Area».

Ο (τρέχον) εμπειρογνώμονας mikegian2008hotmailcom πατάει το κουμπί «✔» που βρίσκεται κάτω από το όνομά του και δίπλα από την if-then αναπαράσταση της εκδοχής του κανόνα του, ενημερώνοντας τον κανόνα με τα στοιχεία τόσο της δικής του εκδοχής αλλά και τα γενικά στοιχεία κανόνα στη δεξιά πλευρά.

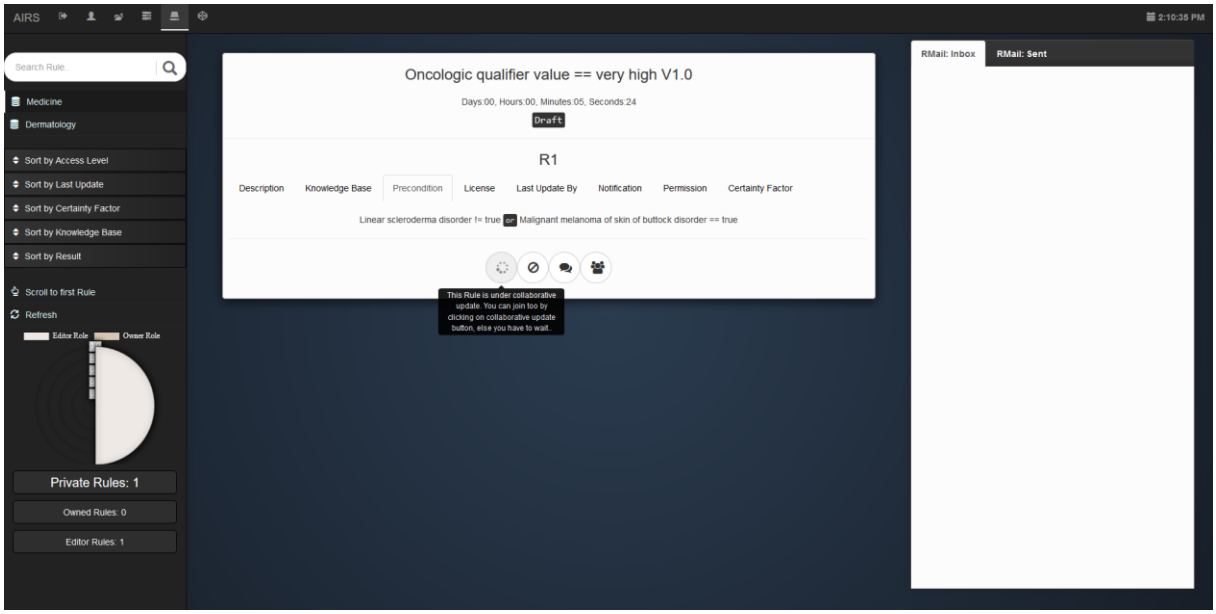

Figure 50 - Προσωπική βάση κανόνων του mikegian2008hotmailcom με το κουμπί για τη λειτουργία «Private Update» να είναι disabled.

Στην εικόνα 49 φαίνεται ο ενημερωμένος κανόνας μέσα από τη προσωπική βάση κανόνων του mikegian2008hotmailcom, με επιλεγμένη την καρτέλα «Precondition» στην οποία φαίνονται οι νέες συνθήκες. Επίσης παρατηρούμε απενεργοποιημένο το (focused) κουμπί του «Private Update» καθώς ο εμπειρογνώμονας doctor είναι ακόμα μέσα στο collaborative update.

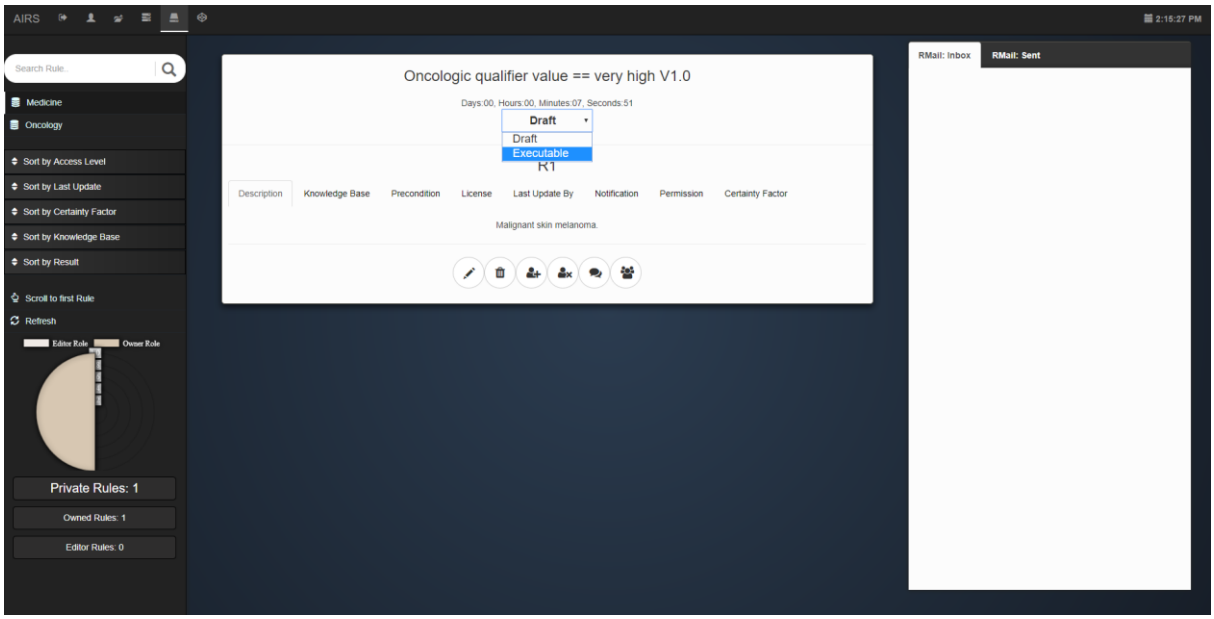

Figure 51 - Προσωπική βάση κανόνων doctor και μετατροπή κανόνα R1 από draft σε executable.

Τέλος ο doctor μετατρέπει τον κανόνα από draft σε executable καθώς είναι πλέον ικανοποιημένος από τις συνθήκες αυτού. Έτσι δίνει τη δυνατότητα τόσο στον ίδιο όσο και στους συνεργάτες του να τον χρησιμοποιήσουν κατά τη διαδικασία εξαγωγής αποφάσεων οντότητας.

5.5. Δημιουργία Οντότητας και Γνωρίσματος αυτής.

Στην ενότητα δημιουργίας οντότητας και γνωρίσματος αυτής, ο εμπειρογνώμονας doctor θα δημιουργήσει μία οντότηταστην οποία θα περάσει ως γνώρισμα κάποια από τις συνθήκες του κανόνα που έχει υπό την κατοχή του, με στόχο την μετέπειτα εξαγωγή αποφάσεων της οντότητας αυτής.

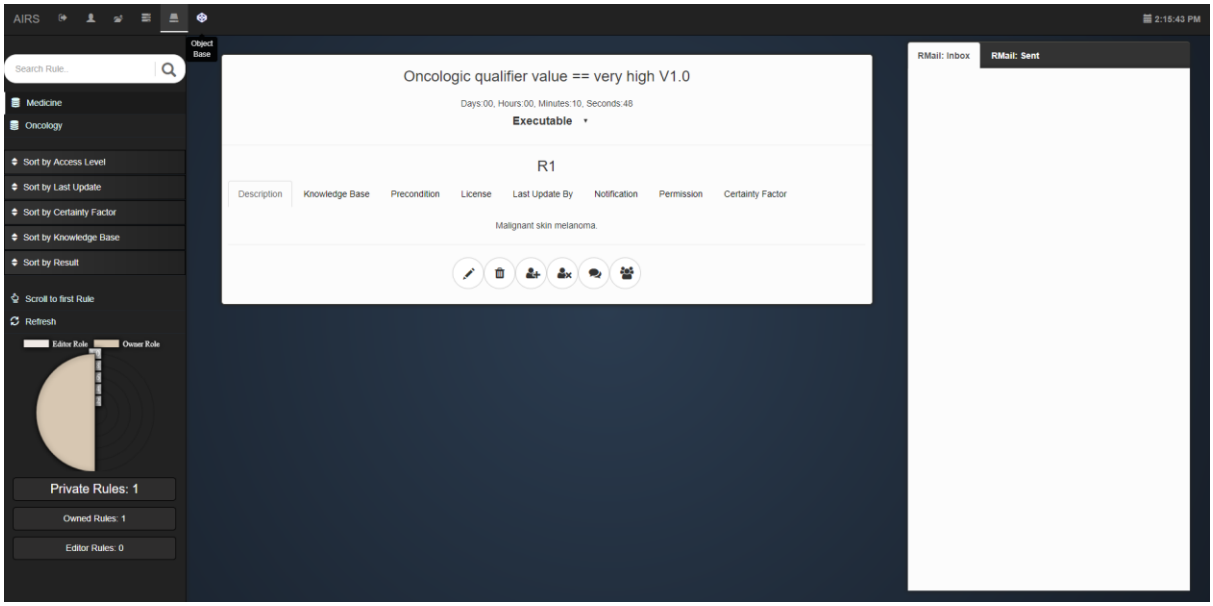

Figure 52 – Προσωπική βάση κανόνων doctor και πάτημα κουμπιού «Object Base» για μετάβαση στην βάση οντοτήτων.

Αρχικά ο doctor επιλέγει το κουμπί «Entity Base» που βρίσκεται στην πάνω πλευρά της οθόνης για τη μεταφορά του στη σελίδα οντοτήτων.

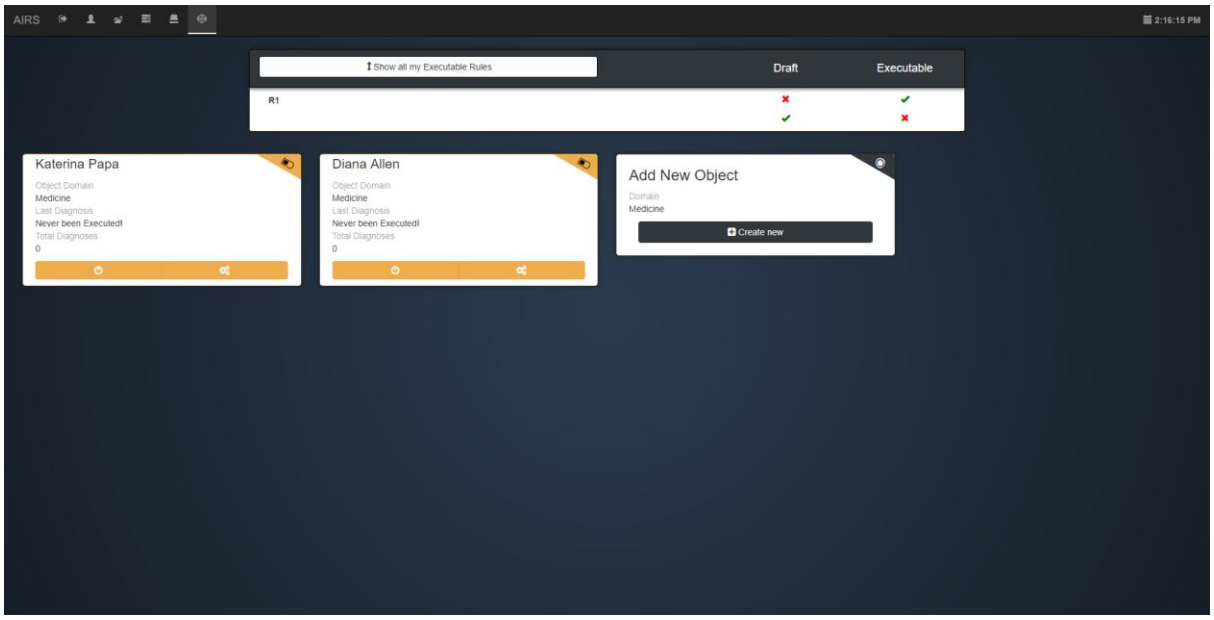

Figure 53 - Βάση οντοτήτων με αυτόματη δημιουργία δύο οντοτήτων εξαετίας της επικοινωνίας του συστήματος με την πλατφόρμα από την οποία προέρχεται ο doctor.

Εκεί ο doctor παρατηρεί ότι του εμφανίζονται δύο γνώριμες οντότητες από την πλατφόρμα την οποία προέρχεται. Οι οντότητες αυτές αντιστοιχούν στους δύο ασθενείς που διαχειρίζεται στην πλατφόρμα iPHR ενώ τα στοιχεία των ασθενών αυτών (μετρήσεις, αλλεργίες, κτλ..) καταχωρούνται ως γνωρίσματα πάνω σε αυτές τις οντότητες. Ο doctor πέραν του αυτόματου συγχρονισμού οντοτήτων, μπορεί να δημιουργήσει και εδώ οντότητες πατώντας δεξί κλικ στο κουμπί «Create New» της καρτέλας «Add New Entity» εμφανίζοντάς του την παρακάτω φόρμα συμπλήρωσης πεδίων.

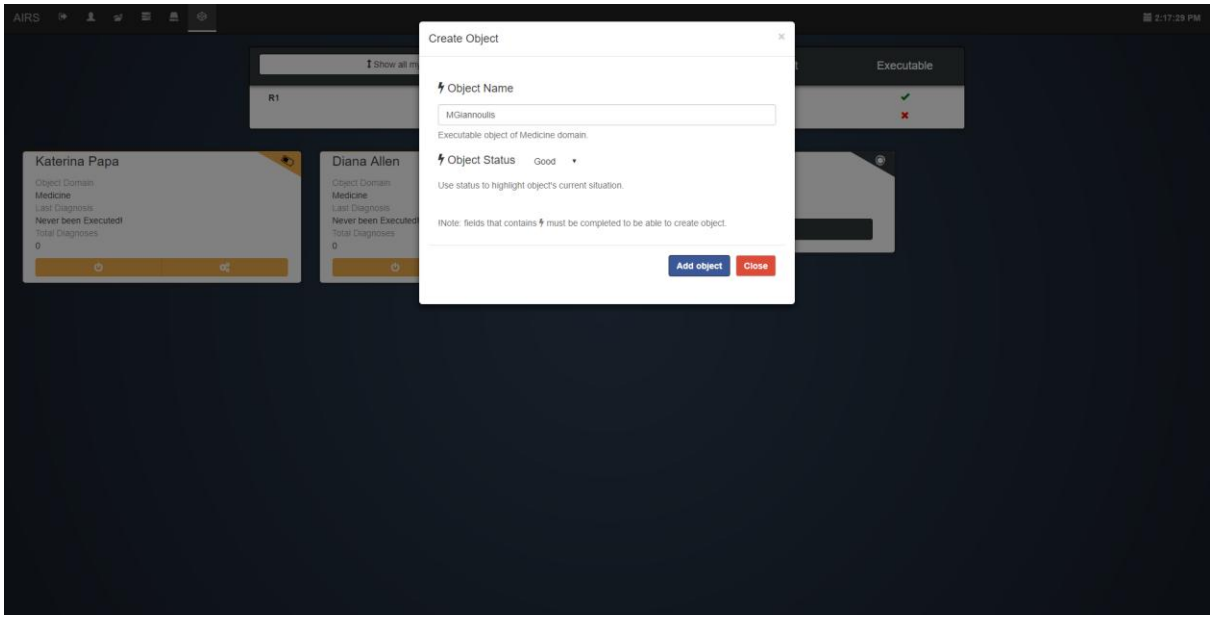

Figure 54 - Φόρμα δημιουργίας νέας οντότητας.

Στη φόρμα αυτή ορίζει το όνομα της οντότητας όπως και την κατάσταση στην οποία βρίσκεται. Όπου πατώντας «Add Entity» δημιουργεί την οντότητα αυτή.

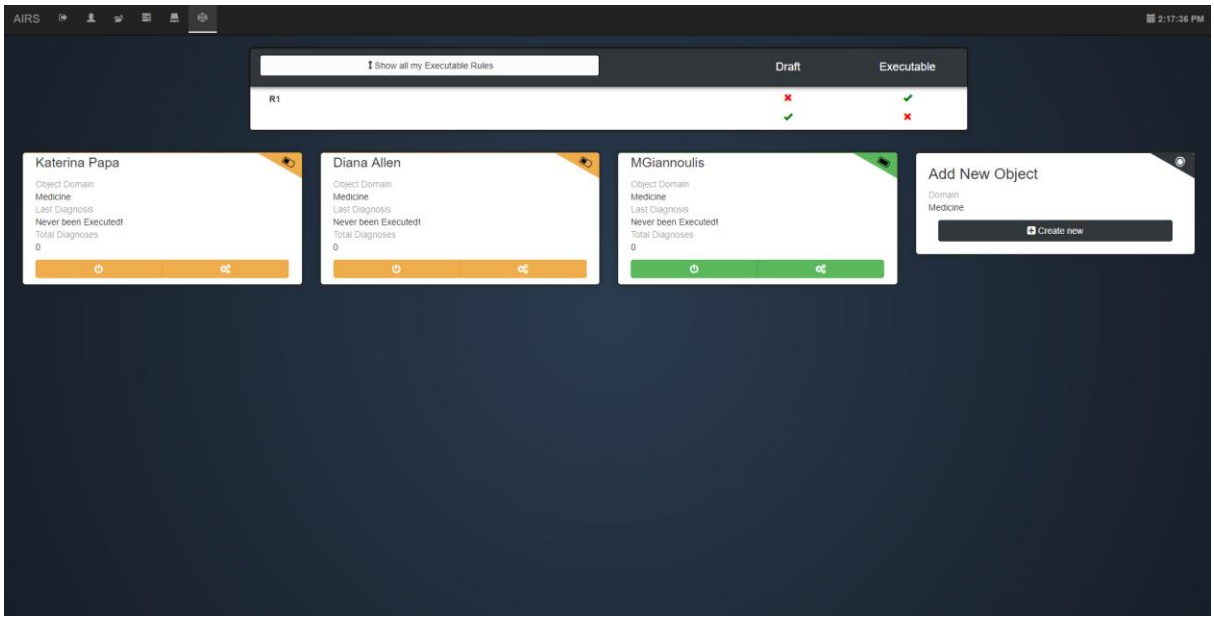

Figure 55 - Αποτέλεσμα δημιουργίας νέας οντότητας.

Στην εικόνα 54 φαίνεται το αποτέλεσμα δημιουργίας οντότητας με ονομασία «MGiannoulis» και κατάσταση «Good».

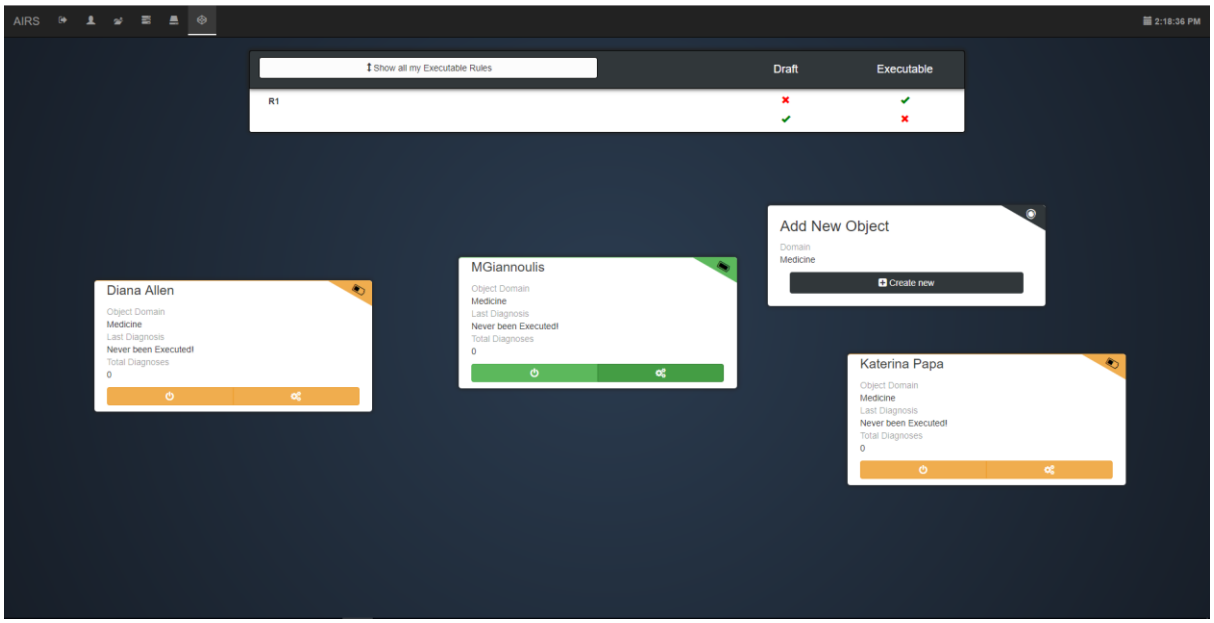

Figure 56 - Πάτημα στο (δεξί) κουμπί επεξεργασίας γνωρισμάτων της οντότητας «MGiannoulis».

Στην καρτέλα της οντότητας αυτής υπάρχουν δύο κουμπιά. Το αριστερό είναι για διαδικασία εξαγωγής αποφάσεων, ενώ το κουμπί δεξιά είναι για τη διαδικασία επεξεργασία γνωρισμάτων αυτού. Πατώντας το δεξί κουμπί εμφανίζεται η παρακάτω φόρμα.

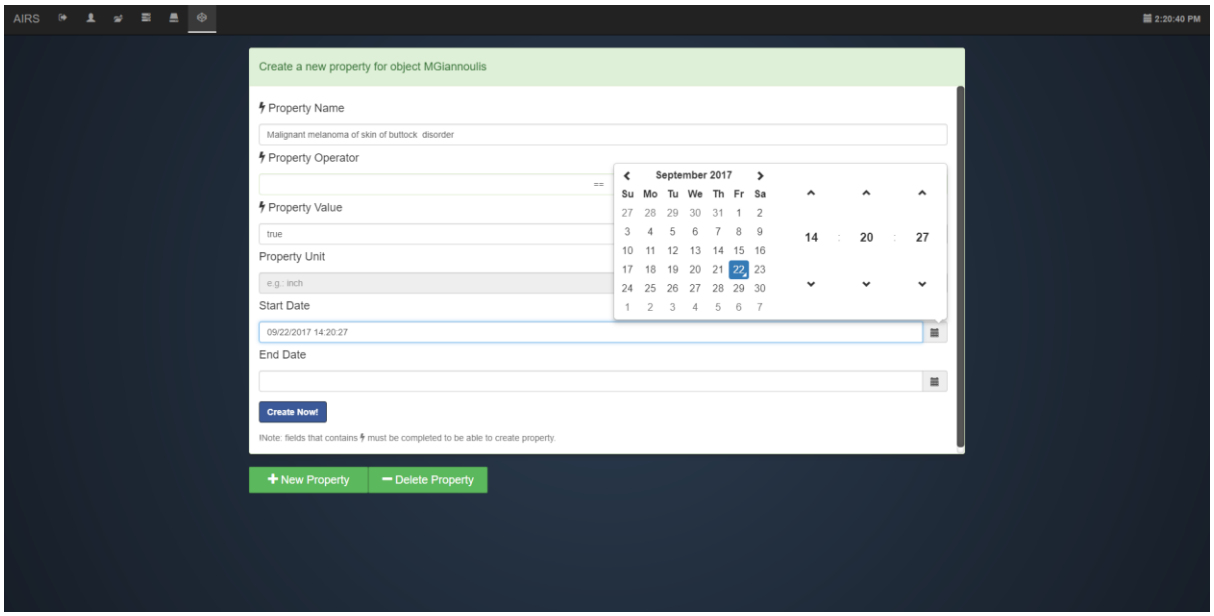

Figure 57 - φόρμα δημιουργίας νέου γνωρίσματος της οντότητας «MGiannoulis».

Στη φόρμα αυτή ο doctor συμπληρώνει τα πεδία ορίζοντας το πρώτο γνώρισμα της οντότητας αυτής. Η δήλωση γνωρίσματος θυμίζει την δήλωση προϋπόθεσης κανόνα όμως εδώ αντί για «distance from start point» και «duration» έχουμε ημερομηνίες.

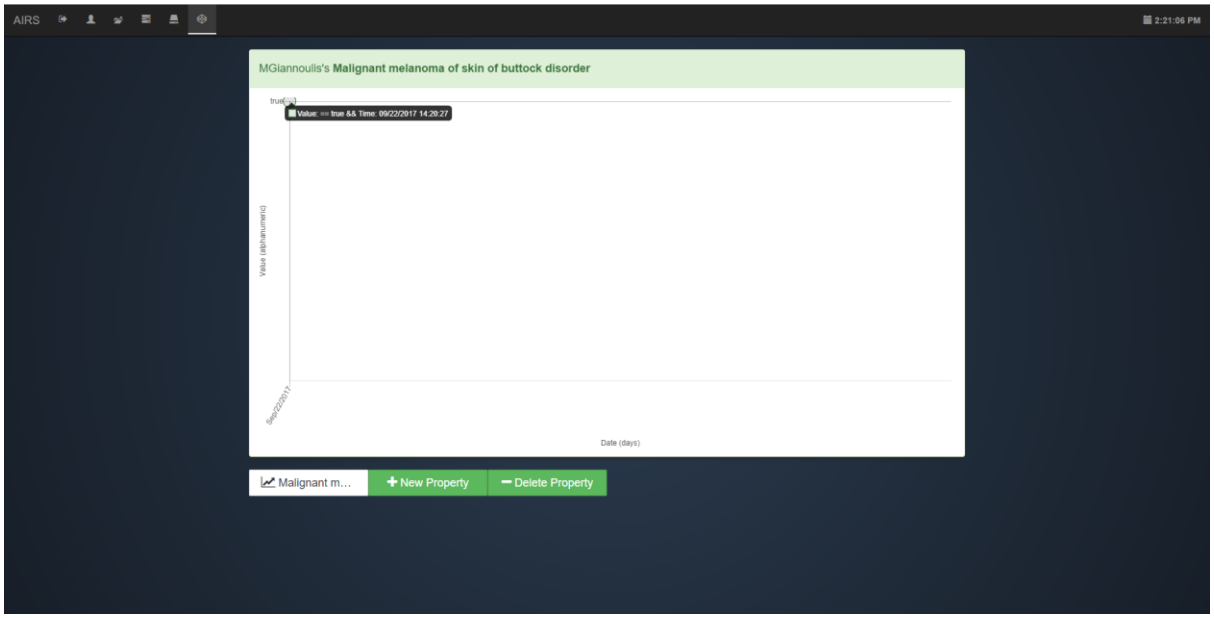

Figure 58 - Γραφική αναπαράσταση γνωρισμάτων της οντότητας «MGiannoulis».

Όλα τα γνωρίσματα μίας οντότητας παρουσιάζονται σε μορφή δισδιάστατου ή μονοδιάστατου γραφήματος, ανάλογα με τη φύση του γνωρίσματος. Στην συγκεκριμένη περίπτωση όπως φαίνεται και στην εικόνα 57, το γνώρισμα παρουσιάζεται σε δυσδιάστατο γράφημα με οριζόντιο άξονα ημερομηνία και κατακόρυφο άξονα τιμή.

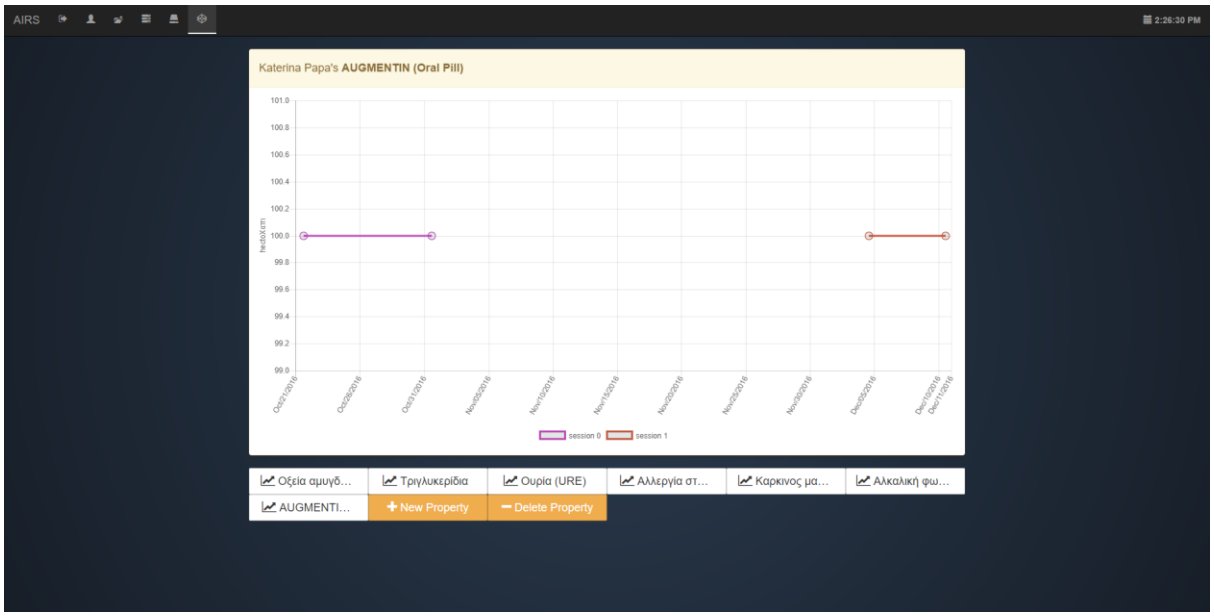

Figure 59 - Γραφική αναπαράσταση γνωρισμάτων της οντότητας «Katerina Papa».

Στην εικόνα 58 παρουσιάζονται επτά γνωρίσματα μίας άλλης οντότητας του doctor το «Katerina Papa», όπου τα γνωρίσματα αυτά όπως και η ίδια η οντότητα δημιουργήθηκαν έπειτα από αυτόματο συγχρονισμό με την εξωτερική πλατφόρμα iPHR.

5.6. Εξαγωγή αποφάσεων του οντότητας.

Ο doctor έχοντας τα κατ' ελάχιστον απαραίτητα:

- Ένα κανόνα.
- Μία οντότητα με ένα γνώρισμα.

πλέον μπορεί να εκτελέσει εξαγωγή αποφάσεων πάνω στην οντότητα αυτή.

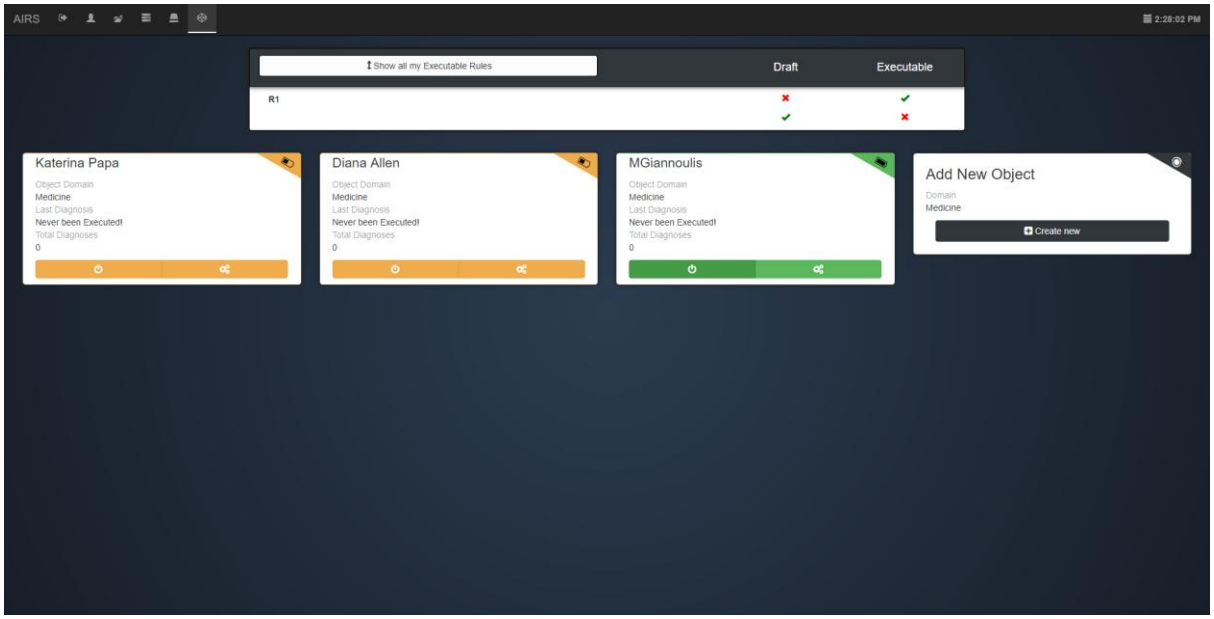

Figure 60 - Πάτημα στο κουμπί για εμφάνιση καρτέλας εξαγωγής αποφάσεων της οντότητας «MGiannoulis».

Αρχικά επιλέγει το αριστερό κουμπί που βρίσκεται πάνω στην καρτέλα της οντότητας «MGiannoulis», ώστε να ανοίξει την καρτέλα της εξαγωγής αποφάσεων.

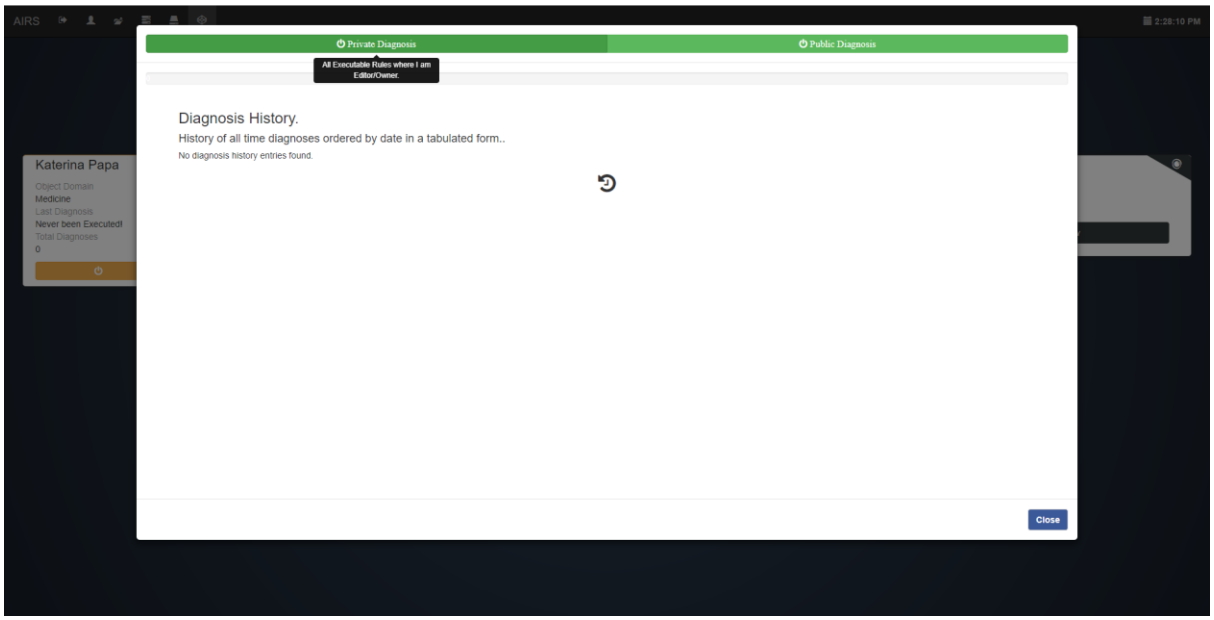

Figure 61 - Επιλογή «Private Decision Derivation» της καρτέλας εξαγωγής αποφάσεων.

Στην καρτέλα αυτή επιλέγοντας «Private Decision Derivation» δηλώνει στο σύστημα ότι οι συλλογισμοί του κατά την εξαγωγή απόφασης να βασιστούν πάνω στους κανόνες που έχει ο ίδιος δημιουργήσει ή είναι συντάκτης σε αυτούς.

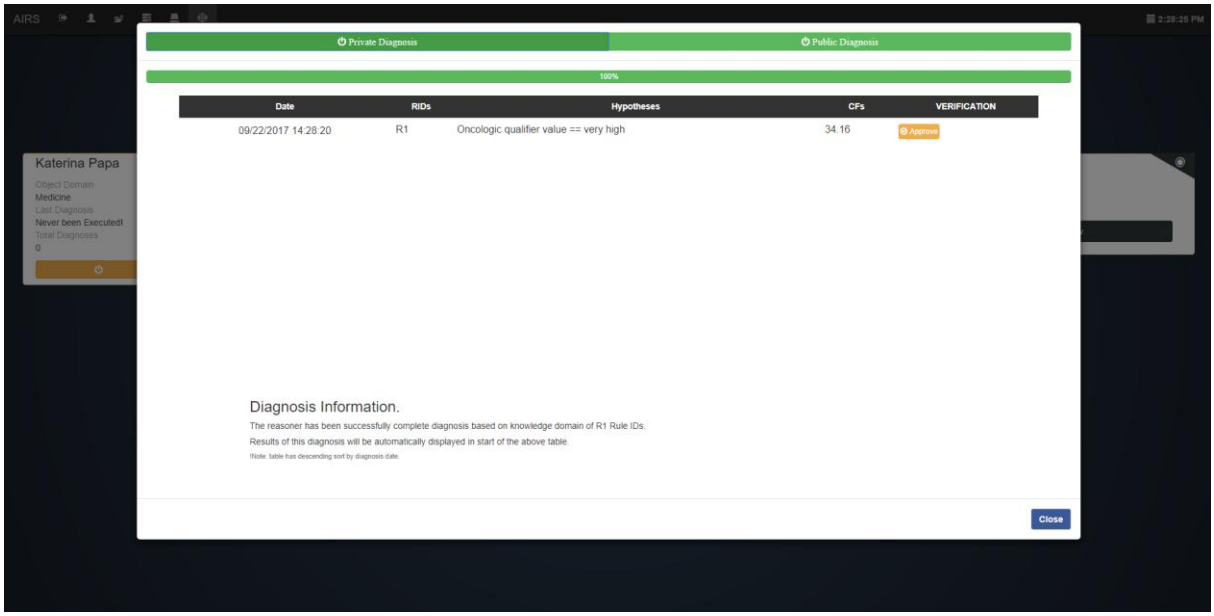

Figure 62 - Εξαγωγή απόφασης από την «Private Decision Derivation» της οντότητας.

Η απόφαση που προέκυψε από το σύστημα μπορεί να γίνει αποδεκτή από τον εμπειρογνώμονα (πατώντας πάνω στο κουμπί «Approve») ή να αγνοηθεί.

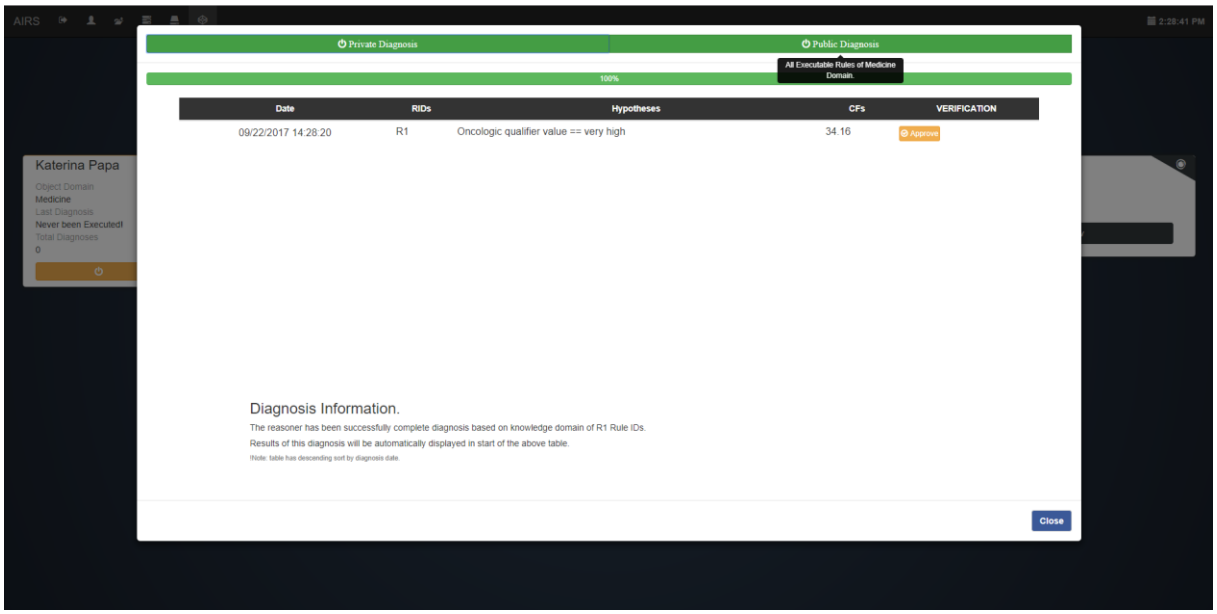

Figure 63 - Επιλογή «Public Decision Derivation» της καρτέλας εξαγωγή απόφασης.

Ο doctor μπορεί επίσης να επιλέξει και «Public Decision Derivation», μέσω της οποίας δηλώνει στο σύστημα ότι οι συλλογισμοί του να βασιστούν σε όλους τους κανόνες που ορίζονται στην ίδια ρίζα πεδίου γνώσης με το δικό του, δηλαδή στο «Medicine».

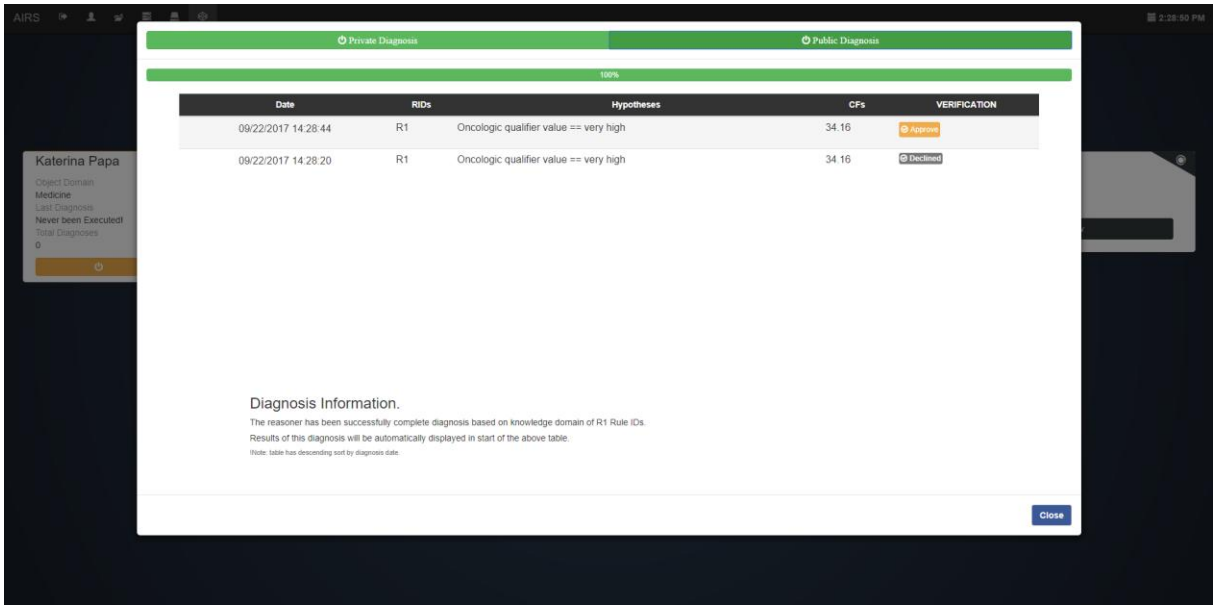

Figure 64 - Εξαγωγή απόφασης από την «Public Decision Derivation» της οντότητας.

Το αποτέλεσμα της προηγούμενης εξαγωγής αποφάσεων εφόσον δεν έγινε approve φαίνεται ως «declined» ενώ η νέα απόφαση που προέκυψε έχει πάρει τη θέση της. Όπως είναι προφανές και τα δύο είδη εξαγωγής απόφασης έβγαλαν τα ίδια αποτελέσματα καθώς στο παράδειγμά μας, ο κανόνας του doctor είναι ο μοναδικός κανόνας του συστήματος.

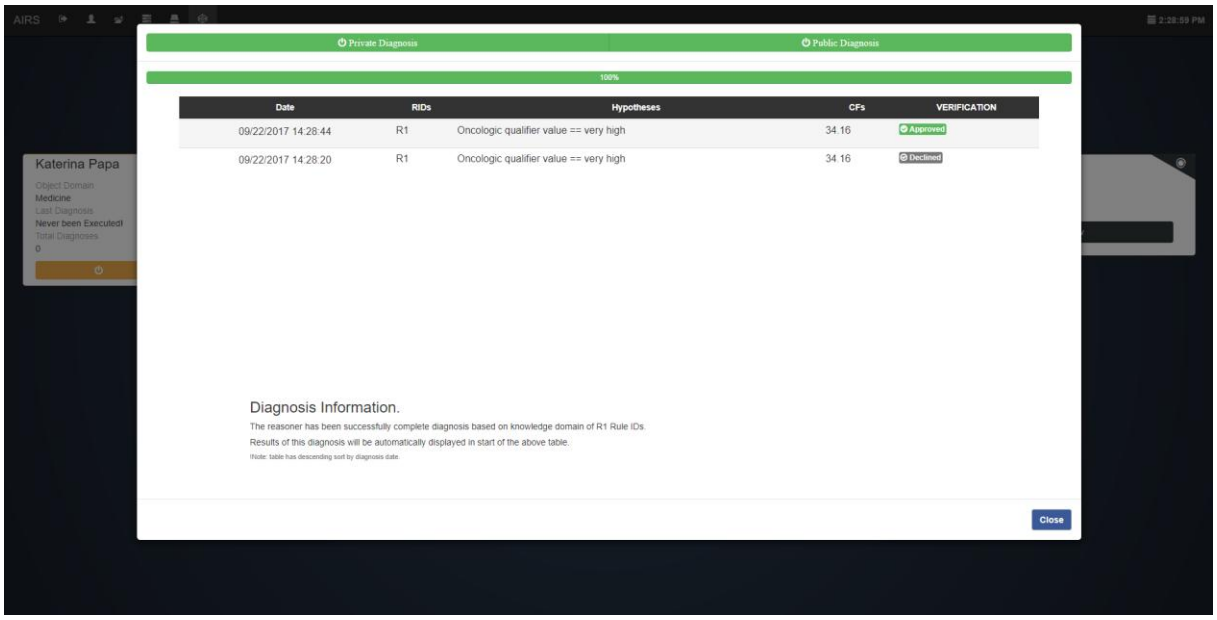

Figure 65 - Αποδοχή απόφασης της τελευταίας εξαγωγής απόφασης της οντότητας.

Έστω ότι ο doctor αποδέχεται την απόφαση του συστήματος πατώντας το κουμπί «Approve». Τότε το σύστημα δημιουργεί αυτόματα ένα νέο γνώρισμα πάνω στην οντότητα αυτή με τιμές βασισμένες στην approved απόφαση.

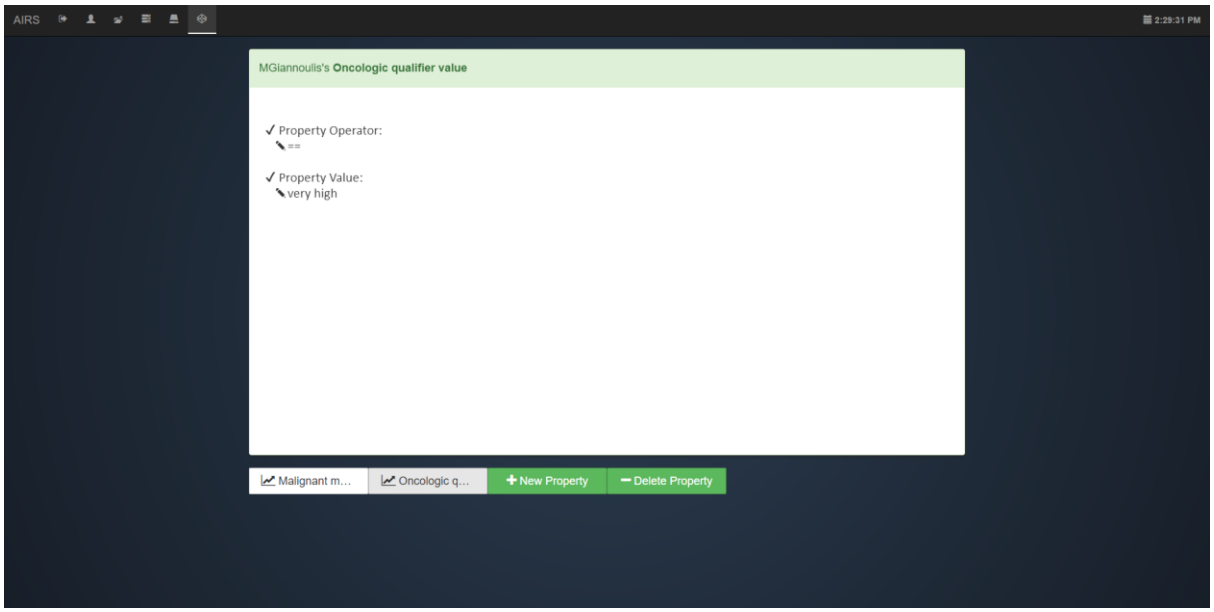

Figure 66 - Προσθήκη απόφασης ως ιδιότητα της οντότητας.

Ο doctor πηγαίνει ξανά στα γνωρίσματα της οντότητας «MGiannoulis» και παρατηρεί ότι το σύστημα όντως πρόσθεσε ένα νέο γνώρισμα το οποία αντιστοιχεί στο αποτέλεσμα της εξαγωγής αποφάσεων. Με τον τρόπο αυτό το σύστημα την επόμενη φορά που θα κάνει εξαγωγή απόφασης θα συμπεριλάβει και το αποτέλεσμα αυτό στα γνωρίσματα του αντικειμένου.

6. Παρόμοια Συστήματα και Σύγκριση με το Σύστημα μας

Η έννοια των συστημάτων γνώσης βρίσκει εφαρμογή για πρώτη φορά την δεκαετία του εξήντα (1960). Από τότε έως σήμερα έχουν δημοσιευτεί δεκάδες αξιόλογα συστήματα, ενώ παραμένει άγνωστος ο αριθμός συστημάτων που έχουν δημιουργηθεί συνολικά. Η ενότητα αυτή έχει στόχο την παρουσίαση μερικών παρόμοιων συστημάτων αλλά και τη ποιοτική σύγκρισή τους με το COSMOS.

6.1. Παρόμοια Συστήματα

Για να χαρακτηριστεί ένα σύστημα παρόμοιο με το δικό μας, πρέπει να έχει κατ' ελάχιστον δύο βασικά χαρακτηριστικά:

- ✓ Rule Management (δυνατότητα δημιουργίας και ενημέρωσης κανόνων).
- ✓ Inference Engine (Μηχανή συλλογισμών).

Για την έρευνα και μελέτη δημοσιεύσεων παρόμοιων συστημάτων χρησιμοποιήθηκαν οι παρακάτω σύνδεσμοι:

- scholar.google.gr (μηχανή αναζήτησης)
- dblp.uni-trier.de (μηχανή αναζήτησης)
- openclinical.org (πλατφόρμα)

Το συμπέρασμα από την έρευνα αυτή είναι ότι ενώ υπάρχουν αρκετές αξιόλογες και πολλά υποσχόμενα δημοσιεύσεις πάνω στα συστήματα γνώσης και την εξαγωγή απόφασης, λίγες από αυτές αναφέρονται σε παρόμοια συστήματα. Οι περισσότερες από αυτές αναφέρονται σε θεωρητικές ή τεχνικές προσεγγίσεις χωρίς κάποια υλοποίηση συστήματος. Ακολουθεί λοιπόν η παρουσίαση των συστημάτων αυτών, ταξινομημένα αλφαβητικά:

1. APACHE III (1991)

Το «***APACHE III***» είναι ένα προγνωστικό σύστημα το οποίο παρουσιάστηκε το 1991 και ανήκει στην κατηγορία Intensive Care Unit (ICU) scoring rule [21]. Σκοπός του συστήματος είναι να προβλέψει τον κίνδυνο του ατόμου να πεθάνει σε νοσοκομείο. Η πρόβλεψη αυτή βασίζεται στην σύγκριση του ιατρικού ιστορικού ασθενή με 18.000 περιστατικά. Τα περιστατικά αυτά είναι αποθηκευμένα στη βάση δεδομένων του συστήματος, έχοντας κατά μέσο όρο 95% ακρίβειας πρόγνωσης. Για την πρόγνωση χρησιμοποιεί έναν score based μηχανισμό.

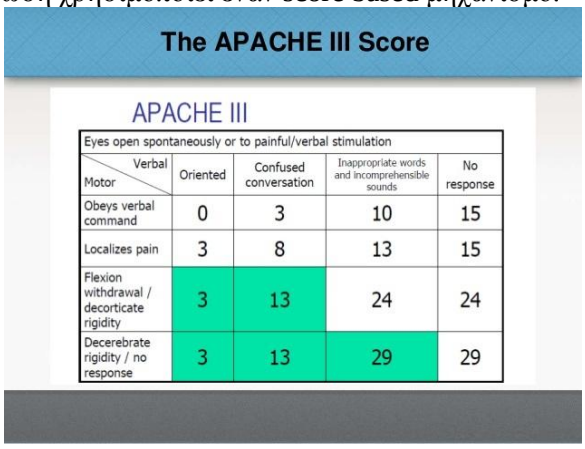

Figure 67 - APACHE III example table of scoring mechanism in action.

2. LISA (2001)

Το «***LISA***» είναι ένα σύστημα κλινικής πληροφόρησης (Clinical Information) και υποστήριξης αποφάσεων (Decision Support System) για συνεργατική περίθαλψη στην παιδική οξεία λεμφοβλαστική λευχαιμία. Ασχολείται πρωτίστως με την παροχή υποστήριξης κατά την περίοδο θεραπείας ασθενή, όπου πρέπει να γίνονται εβδομαδιαίες αποφάσεις για τη δοσολογία φαρμάκου.

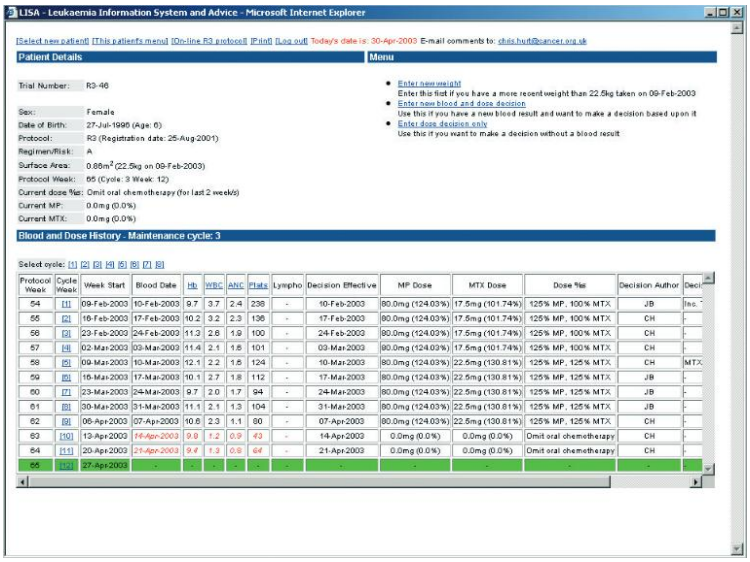

Figure 68 - Στιγμιότυπο της κλινικής διεπαφής για τον προγραμματισμό παρέμβασης για την παρεμβολή και τη συμβουλή λευχαιμίας (LISA), που δείχνει τις λεπτομέρειες του ασθενούς και το ιστορικό αίματος / δοσολογίας.

Οι κανόνες ρύθμισης της δόσης από το δοκιμαστικό πρωτόκολλο MRC εφαρμόστηκαν χρησιμοποιώντας τη γλώσσα μοντελοποίησης κατευθυντήριων γραμμών (Guideline Modeling Language) PROforma και οι συστάσεις (Recommendations) παρέχονται σε κλινική ρύθμιση από τον μηχανισμό έγκρισης TALLIS PROforma [22].
Αποτελείται λοιπόν από δύο κύρια μέρη:

1. Μια κεντρική σχεσιακή βάση δεδομένων της Oracle που περιέχει όλες τις πληροφορίες για τους ασθενείς (προγράμματα φαρμάκων, αποτελέσματα αίματος και τοξικότητας, συνταγογραφούμενες δόσεις κ.λπ.). Αυτές οι πληροφορίες είναι προσβάσιμες από επαγγελματίες υγείας από διαφορετικούς τομείς σε διαφορετικές τοποθεσίες.
2. Μια διαδικτυακή υπό μονάδα υποστήριξης αποφάσεων, η οποία υλοποιείται χρησιμοποιώντας την τεχνολογία ανάπτυξης της κατευθυντήριας γραμμής PROforma. Είναι σχεδιασμένη να παρέχει συμβουλές σχετικά με τις προσαρμογές της δόσης στη θεραπεία της οξείας λεμφοβλαστικής λευχαιμίας στην παιδική ηλικία.

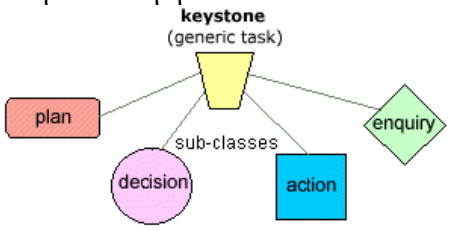

Figure 69 - Proforma language task types.

3. ISABEL (2002)

Το «***Isabel***» είναι ένα διαδικτυακό σύστημα υποστήριξης κλινικών αποφάσεων διάγνωσης που δημιουργήθηκε το 2001 από τους γιατρούς για να προσφέρει υποστήριξη στη λήψη αποφάσεων διάγνωσης στον τομέα της παιδιατρικής.

Figure 70 - ISABEL Interface.

Το Isabel χρησιμοποιεί το λογισμικό επεξεργασίας φυσικής γλώσσας της Autonomy σε αντίθεση με τις τυπικές αναζητήσεις λέξεων-κλειδιών. Η Isabel αποτελείται από μια ιδιόκτητη βάση δεδομένων ιατρικού περιεχομένου με πάνω από 11.000 διαγνώσεις και 4.000 φάρμακα. Υποστηρίζει πρότυπα όπως η συμβολοσειρά ερωτήσεων και το HL7 για την πλήρη ενσωμάτωση API χρησιμοποιώντας JSON ή XML.

4. PUFF (1983)

Το «***PUFF***» είναι ένα έμπειρο σύστημα που ερμηνεύει τα δεδομένα δοκιμής της λειτουργίας των πνευμόνων και εμφανίζεται το 1983. Οι συλλογισμοί του βασίζονται στην προσέγγιση backward chaining [23], χρησιμοποιώντας περίπου 400 κανόνες.

```
RULE811
-------

If:   1) A: The mmf/mmf-predicted ratio is between 35 and 45, and
         B: The fvc/fvc-predicted ratio is greater than 88, or
      2) A: The mmf/mmf-predicted ratio is between 25 and 35, and
         B: The fvc/fvc-predicted ratio is less than 88
Then: 1) There is suggestive evidence (.5) that the degree of
         obstructive airways disease as indicated by the MMF
         is moderate, and
      2) It is definite (1.0) that the following is one of the
         findings about the diagnosis of obstructive airways
         disease:  Reduced mid-expiratory flow indicates
         moderate airway obstruction.

PREMISE: [$AND ($OR (SAND (BETWEEN* (VAL1 CNTXT MMF) 35 45)
                          (GREATERP* (VAL1 CNTXT FVC) 88))
                    (SAND (BETWEEN* (VAL1 CNTXT MMF) 25 35)
                          (LESSP* (VAL1 CNTXT FVC) 80]
ACTION:  (DO-ALL (CONCLUDE CNTXT DEG-MMF MODERATE TALLY 588)
                 (CONCLUDETEXT CNTXT FINDINGS-OAD
                               (TEXT $MMF/FVC2) TALLY 1888))
```

Figure 70 – Παράδειγμα κανόνα του συστήματος PUFF.

Το PUFF ήταν το πρώτο σύστημα που αναπτύχθηκε χρησιμοποιώντας το EMYCIN (Essential MYCIN, van Melle, 1979). Το οποίο περιέλαβε τα χαρακτηριστικά αυτού:

- ✓ διερμηνέα κανόνων (rule interpreter).
- ✓ Εξήγηση (επεξήγηση).
- ✓ απόκτηση γνώσεων (knowledge acquisition).

Όταν δεν μπορεί να ολοκληρώσει την τιμή κάποιας παραμέτρου (συνθήκης), τότε το σύστημα την ζητάει από τον πνευμονολόγο. Στο τέλος ο πνευμονολόγος εξετάζει την αναφορά του PUFF και αν είναι απαραίτητο την τροποποιεί ηλεκτρονικά προτού την εκτυπώσει. Στο 85% των περιπτώσεων η αναφορά δεν χρειάζεται τροποποίηση. Οι τροποποιήσεις συχνά έχουν μορφή σχολίων.

PRESBYTERIAN HOSPITAL OF PMC
CLAY AND BUCHANAN, BOX 7999
SAN FRANCISCO, CA. 94128
PULMONARY FUNCTION LAB

WT 48.8 KG, HT 161 CM, AGE 69 SEX F
REFERRAL DX-

**********TEST DATE 05/13/88

	PREDICTED (+/-SD)	OBSER(%PRED)	POST DILATION OBSER(%PRED)
INSPIR VITAL CAP(IVC)L	2.7	2.3 (86)	2.4 (90)
RESIDUAL VOL (RV) L	2.8	3.8 (188)	3.0 (148)
TOTAL LUNG CAP (TLC) L	4.7	6.1 (138)	5.4 (115)
RV/TLC %	43.	62.	56.
FORCED EXPIR VOL(FEV1) L	2.2	1.5 (68)	1.6 (73)
FORCED VITAL CAP (FVC) L	2.7	2.3 (86)	2.4 (90)
FEV1 /FVC %	73.	65.	67.
PEAK EXPIR FLOW (PEF) L/S	7.1	1.8 (25)	1.9 (26)
FORCED EXP FLOW 25-75% L/S	1.8	8.7 (39)	8.7 (39)
AIRWAY RESIST(RAW) (TLC= 6.1)	0.0(0.0)	1.5	2.2
D F CAP-HGB=14.5 (TLC= 4.8)	24.	17.4 (72)	(74%IF TLC = 4.7)

****

INTERPRETATION: ELEVATED LUNG VOLUMES INDICATE OVERINFLATION. IN ADDITION, THE RV/TLC RATIO IS INCREASED, SUGGESTING A MODERATELY SEVERE DEGREE OF AIR TRAPPING. THE FORCED VITAL CAPACITY IS NORMAL. THE FEV1/FVC RATIO AND MID-EXPIRATORY FLOW ARE REDUCED AND THE AIRWAY RESISTANCE IS INCREASED, SUGGESTING MODERATELY SEVERE AIRWAY OBSTRUCTION. FOLLOWING BRONCHODILATION, THE EXPIRED FLOWS SHOW MODERATE IMPROVEMENT. HOWEVER, THE RESISTANCE DID NOT IMPROVE. THE LOW DIFFUSING CAPACITY INDICATES A LOSS OF ALVEOLAR CAPILLARY SURFACE, WHICH IS MILD.

CONCLUSIONS: THE LOW DIFFUSING CAPACITY, IN COMBINATION WITH OBSTRUCTION AND A HIGH TOTAL LUNG CAPACITY IS CONSISTENT WITH A DIAGNOSIS OF EMPHYSEMA. ALTHOUGH BRONCHODILATORS WERE ONLY SLIGHTLY USEFUL IN THIS ONE CASE, PROLONGED USE MAY PROVE TO BE BENEFICIAL TO THE PATIENT.

PULMONARY FUNCTION DIAGNOSIS:
1. MODERATELY SEVERE OBSTRUCTIVE AIRWAYS DISEASE.
EMPHYSEMATOUS TYPE.

Figure 71 - Παράδειγμα Εξαγωγής απόφασης του συστήματος PUFF.

5. Therapy Edge HIV ® (2001)

Το «***Therapy Edge HIV***» είναι ένα διαδικτυακό σύστημα ηλεκτρονικής ιατρικής εγγραφής (EMR) το οποίο ασχολείται με την θεραπεία του HIV.

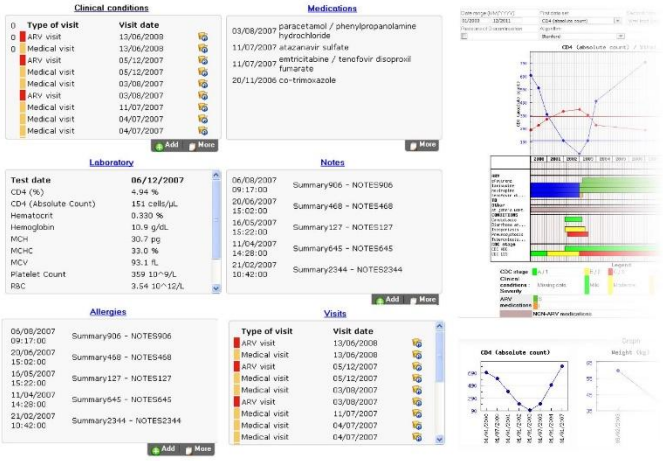

Figure 72 - Therapy Edge Interface.

Χρησιμοποιεί μηχανή συλλογισμών και μία βάση γνώσης για την αξιολόγηση της τρέχουσας κατάστασης του ασθενούς και τη δημιουργία εναλλακτικών επιλογών θεραπείας. Η βάση γνώσης του συστήματος είναι επεκτάσιμη και αποτελείται από κατευθυντήριες γραμμές (guidelines) οι οποίες υποστηρίζουν χρονικά δεδομένα. Τέλος το σύστημα διαθέτει API για επικοινωνία με εξωτερικά συστήματα, παρέχοντας πληροφορία σε μορφή XML.

6.2. Συγκριτική Αξιολόγηση του Συστήματος μας

Έπειτα από την παρουσίαση παραπλήσιων συστημάτων, σειρά έχει η ποιοτική σύγκριση αυτών με το σύστημά μας. Ακολουθεί ένας πίνακας στον οποίο αναδεικνύουμε εν συντομία συγκλίσεις και αποκλίσεις μεταξύ όλων των συστημάτων.

Knowledge System Name	Knowledge Base Management	Knowledge Conflict Detection	Knowledge Time Support	Evidence Measurement Method	Inference Engine Method	System Domain Expert	WEB GUI	GUI Features	Restful API	Cross Platform Support
APACHE III	Private only	✘	✘	Score Based Mechanism	Scoring rule	AID	✘	⋆	✘	✘
ISABEL	Private only	✘	✓	✘	Pattern-recognition algorithms	AID	✓	⋆⋆⋆⋆⋆	✓	✘
LISA	Private only	✘	✘	✘	Proforma	AID	✓	⋆⋆	✘	✘
PUFF	Private only	✘	✘	Certainty Theory	Backward Chaining	AID	✘	⋆	✘	✘
Therapy Edge HIV	Private only	✘	✓	✘	✓	AID	✓	⋆⋆⋆	✓	✘
COSMOS	Real time Collaborative & Private	RCD	✓	Certainty Theory	Forward Channing	ALL	✓	⋆⋆⋆⋆⋆	JWT	✓

Figure 73 - Πίνακας ποιοτικής σύγκρισης συστημάτων.

Στον παραπάνω πίνακα εντοπίζουμε δέκα χαρακτηριστικά τα οποία επιλέχθηκαν με βάση την αναγκαιότητα της σημερινής εποχής από ένα σύστημα γνώσης. Πιο αναλυτικά στον πίνακα αυτό παρατηρούμε τα εξής:

❖ **Knowledge Base Management**

Όλα τα συστήματα παρέχουν δυνατότητα ατομικής ενημέρωσης βάσης γνώση. Όμως το σύστημα COSMOS παρέχει την επιπλέον δυνατότητα συνεργασίας σε πραγματικό χρόνο μεταξύ των εμπειρογνωμόνων, για τη συνεργατική ενημέρωση της βάσης γνώσης.

❖ **Knowledge Conflict Detection**

Το σύστημα COSMOS σε αντίθεση με τα υπόλοιπα συστήματα, παρέχει μηχανισμό εντοπισμού γνώσης που έρχεται σε αντίφαση με την υπόλοιπη γνώση του συστήματος. Η εύρεση αυτή επιτυγχάνεται μέσω του μηχανισμού RCD που αναπτύχθηκε κατά τη δημιουργία του συστήματος για το σκοπό αυτό.

❖ **Knowledge Time Support**

Τα συστήματα ISABEL, Therapy Edge HIV και COSMOS παρέχουν στον εμπειρογνώμονα τη δυνατότητα εισαγωγής χρονικών ιδιοτήτων στη γνώση του συστήματος. Κάτι που δεν υποστηρίζεται από τα υπόλοιπα συστήματα.

❖ **Evidence Measurement Method**

Το σύστημα APACHE III χρησιμοποιεί έναν score based μηχανισμό βασισμένο σε πιθανότητες. Αν και δεν είναι τόσο γενικός όσο η θεωρία βεβαιότητας, είναι ιδανικός για τον σκοπό του συστήματος. Τα συστήματα PUFF και COSMOS χρησιμοποιούν θεωρία βεβαιότητας, ενώ τα συστήματα ISABEL, LISA και Therapy Edge δεν ορίζουν κάποια συγκεκριμένη τεχνική.

❖ **Inference Engine Method**

Κάθε σύστημα χρησιμοποιεί διαφορετική τεχνική στην μηχανή συλλογισμών του. Η κάθε τεχνική επιλέχθηκε ως συνισταμένη διαφόρων παραγόντων, όπως ο τρόπος αναπαράστασης γνώσης, ποσότητα γνώσης και το είδος της γνώσης. Συνεπώς δεν υπάρχει εξ' ορισμού η καλλίτερη τεχνική. Το σύστημα COSMOS επίλεξε την τεχνική «Forward Chaining» [24] λαμβάνοντας υπόψιν τέσσερεις βασικούς παράγοντες:

- Αναπαράσταση γνώσης σε μορφή if-then κανόνα.

- Εξαγωγή αποφάσεων πάνω σε γνωστά γνωρίσματα οντότητας.
- Υποστήριξη μεγάλου αριθμού κανόνων από διαφορά πεδία γνώσης.
- Υποστήριξη αλυσιδωτών συλλογισμών κανόνων (pipeline).

❖ **System Domain Expert**

Όλα τα προαναφερθέντα παρόμοια συστήματα έχουν κατασκευαστεί να εξυπηρετούν ένα πολύ συγκεκριμένο πεδίο γνώσης στον τομέα της Ιατρικής. Το APACHE III ως προγνωστικό θανάτου, το ISABEL για παιδιατρική, το LISA για λευχαιμία, το PUFF για πνευμονολογία, το Therapy Edge για το HIV. Το COSMOS από την άλλη, μπορεί να φιλοξενήσει εμπειρογνώμονες και γνώση από οποιοδήποτε πεδίο γνώσης σε οποιοδήποτε τομέα (π.χ. Επιστήμη Υπολογιστών, Γεωλογία, Κοσμολογία, Ιατρική, κτλ.) καθώς ο μοναδικός επεκτάσιμος περιορισμός είναι το εύρος οντολογιών που γνωρίζει το σύστημα.

❖ **WEB GUI**

Όλα τα συστήματα παρέχουν διεπαφή για τον τελικό χρήστη του συστήματος, όμως τα ISABEL, LISA, Therapy Edge και COSMOS παρέχουν τις λειτουργίες τους μέσω του διαδικτύου.

❖ **GUI Features**

Η διεπαφή των συστημάτων PUFF και APACHE III παίρνουν την ελάχιστη βαθμολογία του ενός αστεριού καθώς όχι μόνο δεν παρέχουν online δυνατότητες αλλά και αυτές είναι πολύ περιορισμένες. Η διεπαφή του LISA είναι επίσης περιορισμένη όμως παίρνει ακόμα ένα αστεράκι καθώς είναι προσβάσιμη μέσω του διαδικτύου. Η διεπαφή του συστήματος Therapy Edge παίρνει τρία αστέρια καθώς παρέχεται online, προσφέροντας παράλληλα περισσότερες λειτουργίες από τα προαναφερθέντα. Τέλος τα συστήματα ISABEL και COSMOS παίρνουν το μέγιστο αριθμό αστεριών καθώς παρέχουν ποιότητα και ποσότητα στις λειτουργίες τους με σύγχρονες τεχνικές.

❖ **Restful API**

Είναι πολύ αναγκαίο στη σημερινή εποχή να υπάρχει η δυνατότητα επικοινωνίας ενός συστήματος, π.χ. συστήματος γνώσης, με άλλα υποστηρικτικά συστήματα στο έργο του, π.χ. βάσεις δεδομένων. Το σύστημα COSMOS γνωρίζοντας την ανάγκη αυτή όχι απλά διαθέτει Restful API αλλά το έχει υλοποιήσει με την τρέχουσα τεχνολογία (state of the art) καθώς χρησιμοποιεί το Json Web Token (JWT) για την εξουσιοδότηση των clients. Τα συστήματα ISABEL και Therapy Edge διαθέτουν επίσης Restful API παρέχοντας πληροφορίες σε μορφή JSON και XML, χωρίς όμως να είναι γνωστή η τεχνολογία μέσω της οποίας υλοποιήθηκε.

❖ **Cross Platform Support**

Το COSMOS παρέχει δυνατότητα χρήσης του συστήματος από εμπειρογνώμονες διαφόρων πλατφορμών, συγχρονίζοντας αυτόματα τα προσωπικά τους στοιχεία και οντότητες που ενδεχομένως να διαχειρίζονται. Με τον τρόπο αυτό οι εμπειρογνώμονες ανεξάρτητα πλατφόρμας, αποκτούν ένα κοινό σημείο αναφοράς για συνεργασία και επικοινωνία.

7. Συμπεράσματα και Μελλοντικές Επεκτάσεις

7.1. Συμπεράσματα

Το COSMOS εξαιτίας του τρόπο διαχείρισης και χρήσης της γνώσης δίνει μεγαλύτερη αξία σε αυτή, καθώς η γνώση ενός εμπειρογνώμονα μπορεί να γίνει άμεσα γνώση πολλών. Το COSMOS επενδύει πάνω στη συνεργασία μεταξύ εμπειρογνωμόνων, προσφέροντας πλούσιες δυνατότητες (π.χ. Cross Platform, Realtime Collaboration, Forum, Mail, κτλ.) με στόχο την παραγωγή έγκυρης γνώσης. Επιπλέον το COSMOS δίνει μία νέα διάσταση στη σχέση εμπειρογνώμονα – σύστημα γνώσης, καθώς με τη χρήση σύγχρονης τεχνολογίας μπορεί ο ένας μπορεί να συμβουλεύει και να συμβουλεύεται από τον άλλο σε πραγματικό χρόνο. Η ταχύτητα και η υποστήριξη πολύπλοκων συλλογισμών (pipelines) είναι δύο ακόμα χαρακτηριστικά του συστήματος COSMOS με τα οποία ο άνθρωπος είναι αδύνατο να συγκριθεί.

Τέλος το σύστημα γνώσης COSMOS εξαιτίας των σύγχρονων διαδικτυακών χαρακτηριστικών που διαθέτει, το καθιστούν προσβάσιμο τόσο από τον υπολογιστή όσο και από το τάμπλετ του κάθε εμπειρογνώμονα ανά πάσα στιγμή.

7.2. Μελλοντικές Επεκτάσεις

Το σύστημα γνώσης που παρουσιάστηκε και αναλύθηκε κατά την πτυχιακή αυτή εργασία είναι λειτουργικό έχοντας περάσει με επιτυχία το στάδιο ελέγχου δυσλειτουργιών. Στο σημείο αυτό έχει περιθώρια για επέκταση και βελτιώσεις, με ορισμένες από αυτές παρουσιάζονται παρακάτω ταξινομημένες σε φθίνουσα σειρά προτεραιότητας:

1. Επέκταση χρονικού προσδιορισμού συνθήκης κανόνα.
2. Δημιουργία φίλτρων αναζήτησης εμπειρογνωμόνων του συστήματος.
3. Επέκταση φίλτρων αναζήτησης κανόνων.
4. Επέκταση των οντολογιών που γνωρίζει το σύστημα.
5. Επέκταση του Restful API για υποστήριξη περισσότερων υπηρεσιών.
6. Επέκταση συνεργασίας με περισσότερες πλατφόρμες πιστοποίησης εμπειρογνώμονα
7. Επέκταση γραφικής διεπαφής για καλλίτερη απόδοση σε οθόνες με ανάλυση μικρότερη από High Definition.

Βιβλιογραφία

[1] Μ. Μαρακάκης, Προγραμματισμός σε Λογική για Τεχνητή Νοημοσύνη, Ηράκλειο Κρήτης: Εκδώσεις Νέων Τεχνολογιών, 2014.

[2] F. v. H. Grigoris Antoniou, Εισαγωγή στο Σημασιολογικό Ιστό, Αθήνα: Κλειδάριθμος, 2009.

[3] Ontology, National Center for Biomedical, «Bioportal,» 2005. [Ηλεκτρονικό]. Available: https://bioportal.bioontology.org/. [Πρόσβαση 24 09 2017].

[4] E. H. Shortlife, Computer Based Medical Consultation MYCIN, New York: ELSEVIER, 1976.

[5] I. F. G. G. Schaible, A Flexible Micro Protocol Framework, Springer, 2004.

[6] M. S. Sudeep Agarwal, Learning Sinatra, PACKT, 2016.

[7] «Tomcat,» APACHE, 26 03 2009. [Ηλεκτρονικό]. Available: http://tomcat.apache.org/. [Πρόσβαση 24 09 2017].

[8] C. S. M. William F. Clocksin, Programming in Prolog, Springer, 2003, p. 314.

[9] M. Triska, «Prolog DCG Primer,» [Ηλεκτρονικό]. Available: https://www.metalevel.at/prolog/dcg. [Πρόσβαση 24 09 2017].

[10] «JSON,» 2000. [Ηλεκτρονικό]. Available: http://www.json.org/. [Πρόσβαση 24 09 2017].

[11] S. Gonzalez, jQuery UI in Action, MANNING, 2014, p. 384.

[12] «CHART JS,» 2014. [Ηλεκτρονικό]. Available: http://www.chartjs.org/. [Πρόσβαση 24 09 2017].

[13] M. F. C. T. L. G. H. Y. D. O. S. G. D. Bader, «Cytoscape.js: a graph theory library for visualisation and analysis,» *OXFORD ACADEMIC*, p. 311, 15 01 2016.

[14] J. E. Whitesitt, Boolean Algebra and Its Applications, New York: Dover Publications, 1995.

[15] D. Haussler, «On The Learnability Of Disjunctive Normal Form,» Kluwer Academic Publishers, Boston, 1995.

[16] M. P. a. F. R. Macario Polo, «Reflective Persistence (Reflective CRUD: Reflective Create, Read, Update & Delete),» University of Castilla, Ciudad Real Spain, 2001.

[17] A. Silberschatz, Database system concepts, New York: McGraw-Hill, 1997, pp. McGraw-Hill.

[18] J. L. Hennessy, Computer Architecture, San Francisco CA: Morgan Kaufmann Publishers, 2003.

[19] M. K. M. T. Fensel, The Web Service Modeling Toolkit - An Integrated Development Environment for Semantic Web Services, Australia: Springer, 2007.

[20] «Nimbus JWT Library,» [Ηλεκτρονικό]. Available: https://github.com/ Connect2id/Nimbus-JWT/ wiki/ Nimbus-JWT. [Πρόσβαση 24 09 2017].

[21] E. S. Epstein, «A Scoring Rule System for Probability Forecasts or Ranked Categories,» p. 3, 1969 05 26.

[22] N. J. C. L. A. R. R. T. P. John Fox, PROforma: a general technology for clinical decision support systems, ELSEVIER, 1997.

[23] W. A. S. George F. Luger, Artificial Intelligence, Harlow: ADDISON-WESLEY, 1999, p. 18.

[24] F. K. Fahiem Bacchus, «Using Temporal Logic to Control Search in a Forward Chaining Planner,» p. 153, 01 01 1995.

Παράρτημα Α - Παραδείγματα Κανόνων της Βάσης Γνώσης

/* ====================== Περιγραφή μερών κανόνα βάσης γνώσης ====================== */

Ένας κανόνας της βάσης γνώσης αποτελείται από τρία κατηγορήματα. Τα ***rule_shell***, ***rule_prec*** και ***rule_prec_events***. Η σχέση μεταξύ του κατηγορήματος rule_shell με το rule_prec είναι ένα προς πολλά. Η σχέση μεταξύ του rule_prec με το κατηγόρημα rule_prec_events είναι επίσης ένα προς πολλά. Συνεπώς ένας κανόνας, θα μπορούσε να έχει πολλές συνθήκες, οι οποίες θα επαναλαμβάνονται σε διαφορετικά χρονικά σημεία.

/* ------------ Rule Shell ------------ */

rule_shell(Rule_ID, Prec_List, Result, Result_CF)

1. Rule_ID: αναγνωριστικό του κανόνα (e.g: 1).
2. Prec_List: λίστα όλων των συνθηκών του κανόνα. (e.g: [A,B,C,D]. Η πυροδότηση ενός κανόνα εξαρτάτε από το συνδυασμό των θέσεων της λίστας αυτής που θα ενοποιήσουμε με "true").
3. Result: Αποτέλεσμα κανόνα και χρονικός ορισμός (e.g: Όνομα_Τελεστής_Τιμή_ΜονάδαΜέτρησης_ΧρονικήΔιάρκεια).
4. Result_CF: Παράγοντας Βεβαιότητας Αποτελέσματος.

Αναγνωριστικό του κατηγορήματος «rule_shell» είναι το Rule_ID.

/* ------------ Rule Prec ------------ */

rule_prec(Rule_ID, Prec_ID, Prec_Name, Prec_OP_Value, Prec_Unit)

1. Rule_ID: αναγνωριστικό του κανόνα (e.g: 1).
2. Prec_ID: μοναδικός αριθμός στο πεδίο ορισμού του κανόνα Rule_ID (e.g: 4, όπου αυτό σημαίνει ότι είναι η συνθήκη_4 του κανόνα Rule_ID).
3. Prec Name: Ονομασία συνθήκης (e.g: Weight).
4. Prec_OP_Value: Τελεστής και τιμή συνθήκης (e.g: >= 100).
5. Prec_Unit: Μονάδα μέτρησης συνθήκης (e.g: gram).

Αναγνωριστικό του κατηγορήματος «rule_prec» είναι το [Rule_ID, Prec_ID].

/* ------------ Rule Prec ------------ */

rule_prec_event(Rule_ID, Rule_Prec, DistanceFromStartPoint, Duration)

1. Rule_ID: αναγνωριστικό του κανόνα (e.g: 1).
2. Prec_ID: μοναδικός αριθμός στο πεδίο ορισμού του κανόνα Rule_ID (e.g: 4, όπου αυτό σημαίνει ότι είναι η συνθήκη_4 του κανόνα Rule_ID).
3. DistanceFromStartPoint: δηλώνει την καθυστέρηση να έναρξης μίας συνθήκης σε σχέση με τις υπόλοιπες.
4. Duration: Χρονική διάρκεια που κράτησε η συνθήκη Rule_Prec.

Το κατηγόρημα «rule_prec_event» έχει ως αναγνωριστικό τρία πεδία [Rule_ID, Prec_ID, DistanceFromStartPoint], με τα δύο πρώτα πεδία της λίστας να δείχνουν το «rule_prec» στο οποίο αναφέρονται.

/* ====================== Περιγραφή του κανόνα 1 ====================== */

Εάν ο Χ έχει τα εξής γνωρίσματα: Weight > 85 kilogram ή (Difficulty transferring weight finding == true και Penicillin G Product != true), όπου το "Difficulty transferring weight finding == true" έχει διάρκεια περισσότερο από 5 μήνες ενώ το "Penicillin G Product != true" έχει διάρκεια 8 ώρες και ξεκίνησε μία μέρα μετά το " Difficulty transferring weight finding == true".
Τότε ισχύει για το Χ η εξής απόφαση: "Abdomen hyper resonant finding == true", με χρονική διάρκεια 10 μέρες.

/* ====================== Αναπαράσταση του κανόνα 1 στη βάση γνώσης ====================== */

Για την αναπαράσταση του παραπάνω κανόνα στη βάση γνώσης χρειαζόμαστε τρία είδη κατηγορημάτων. Rule Shell, Rule Prec, Rule Prec Events.

/* ------------ Κέλυφος Κανόνα ------------ */

```
rule_shell(1,[C,B,A],'Abdomen hyper resonant finding_==_true_null_864000',33):-
 (
   A == true,
   B == true;
   C == true
).
```

/* Το κέλυφος του κανόνα 1 θα πυροδοτηθεί εάν η συνθήκη 1 είναι αληθής και η συνθήκη 2 ή συνθήκη 3 είναι επίσης αληθής. Δηλαδή εάν ισχύει ότι Condition 1 == true && (Condition 2 == true || Condition 3 == true). */

/* ------------ Συνθήκη_1 κανόνα ------------ */

```
rule_prec(1,1,'Weight',A,gram):-
 (
  number(A),
  A > 85000;
  A == '> 85000'
 ).
```

/* Η συνθήκη 1 του κανόνα 1 θα πυροδοτηθεί εάν το γνώρισμα με το οποίο θα ενοποιηθεί είναι αριθμός μεγαλύτερο του 85000 γραμμάρια ή αλφαριθμιτικό «> 85000» γραμμάρια. */

/* ------------ Συνθήκη_2 κανόνα ------------ */

```
rule_prec(1,2,'Difficulty transferring weight finding',A,_):-
 (
  atom(A),
```

```
  A==true;
  A == '==true'
 ).
/* Η συνθήκη 2 του κανόνα 1 θα πυροδοτηθεί εάν το γνώρισμα με το οποίο θα ενοποιηθεί είναι το «Dificulty transferring weight finding» και είναι «true» ή είναι «== true». */

/* ---------- Συνθήκη_3 κανόνα ---------- */
rule_prec(1,3,'Penicillin G product',A,_):-
 (
  atom(A),
  A\=true;
  A == '!=true'
 ).
/* Η συνθήκη 3 του κανόνα 1 θα πυροδοτηθεί εάν το γνώρισμα με το οποίο θα ενοποιηθεί είναι το «Penicillin G product» και είναι διαφορετικό του «true» ή είναι «!= true». */

/* ---------- Χρονικός Ορισμός Συνθήκης 3 κανόνα ---------- */
rule_prec_event(1, 3, 86400, A):-
      number(A),
      A == 28800.
/* Ο χρονικός ορισμός αυτός της συνθήκης 3 του κανόνα 1 θα πυροδοτηθεί, εάν η συνθήκη 3 διήρκησε 28800 δευτερόλεπτα και ξεκίνησε 86400 δευτερόλεπτα μετά από τις συνθήκες που ξεκίνησαν τη χρονική στιγμή 0. */

/* ---------- Χρονικός Ορισμός Συνθήκης 2 κανόνα ---------- */
Rule_prec_event(1, 2, 0, A):-
      Number(A),
      A > 13148715.
/* Ο χρονικός ορισμός αυτός της συνθήκης 2 του κανόνα 1 θα πυροδοτηθεί, εάν η συνθήκη 2 διήρκησε περισσότερο από 13148715 δευτερόλεπτα και ξεκίνησε τη χρονική στιγμή 0. */
```

Παράρτημα Β - SWI Prolog Μηχανή Συλλογισμών

```
/* ====================== Αρχικοποίηση Αλγορίθμου μηχανής συλλογισμών ====================== */
```
Ο αλγόριθμος αρχικοποιείται με δύο λίστες. Η μία λίστα περιλαμβάνει τα γνωρίσματα της οντότητας πάνω στην οποία καλείται να εξάγει αποφάσεις. Η άλλη λίστα περιλαμβάνει όλους τους εκτελέσιμους κανόνες που μπορούν να λάβουν μέρος στη διαδικασία συλλογισμών της συγκεκριμένης εξαγωγής απόφασης.

```
/* ====================== Ανάλυση μηχανής συλλογισμών σε μορφή κώδικα ====================== */

/* =-=-=-=-=-=-=-=-=-=-=-=-=-=-= Περιγραφή SWI Reasoner (Predicate) =-=-=-=-=-=-=-=-=-=-=-=-=-=-=
```
Το κατηγόρημα αυτό είναι υπεύθυνο για την κλήση του κατηγορήματος «Reasoner Pipeline», από το οποίο παραλαμβάνει μία λίστα με τους κανόνες που δεν εκτελέστηκαν κατά τη διαδικασία συλλογισμών. Έπειτα χρησιμοποιεί την λίστα αυτή για την παραγωγή μίας νέας λίστας με όλους τους κανόνες που εκτελέστηκαν. Στη συνέχεια καλεί το κατηγόρημα «Hypothesis Certainty Factor Merger», από το οποίο παραλαμβάνει δύο λίστες. Μία λίστα με αποφάσεις και μία λίστα με παράγοντες βεβαιότητας αυτών. Επιστρέφει τρείς λίστες. Μία λίστα με αναγνωριστικά κανόνων που εκτελέστηκαν. Μία λίστα με αποτελέσματα κανόνων. Μία λίστα με παράγοντες βεβαιότητας των αποτελεσμάτων κανόνων.
Είσοδος κατηγορήματος:
a) RID Domain: RID_Domain.
b) All SWI Item Properties: SWI_IProperties.
Έξοδος κατηγορήματος:
a) List of RIDs took part in diagnosis: SWI_Final_TRIDs.
b) List of Diagnosis results: SWI_Final_TResults.
c) List of Diagnosis result CFs: SWI_Final_TCFs.
```
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= */
swi_reasoner(RID_Domain,IProperties,SWI_Final_TRIDs,SWI_Final_TResults,SWI_Final_TCFs) :-
    reasoner_pipeline(RID_Domain, IProperties,RID_Domain,NoTriggeredRIDomain),
    sub(RID_Domain,NoTriggeredRIDomain,SWI_Final_TIDs),
    cf_hypothesis_merger(SWI_Final_TIDs,[],SWI_Final_TResults,[],SWI_Final_TCFs),
    id_to_rid(SWI_Final_TIDs,SWI_Final_TRIDs).

/* =-=-=-=-=-=-=-=-=-=-= Περιγραφή Hypothesis Certainty Factor Merger (Predicate) =-=-=-=-=-=-=-=-=-=-=
```
Το κατηγόρημα αυτό είναι υπεύθυνο για την εύρεση των αποτελεσμάτων των κανόνων που εκτελέστηκα και τους παράγοντες βεβαιότητας αυτών. Έπειτα μέσω της θεωρίας βεβαιότητας πραγματοποιεί συγχώνευση ενδεχόμενων πολλαπλών εμφανίσεων αποτελεσμάτων κανόνων. Επιστρέφει δύο λίστες. Μία λίστα με αποτελέσματα κανόνων καθαρισμένα από διπλές εμφανίσεις. Μία λίστα με παράγοντες βεβαιότητας των αποτελεσμάτων αυτών.
Είσοδος κατηγορήματος:
a) RID Domain: RID_Domain.
b) Temp Rule Hypothesis List: Temp_Hypotheses.
c) Temp Rule Hypothesis CF List: Temp_CFs.
Έξοδος κατηγορήματος:
a) Merged hypothesis list of requested RID Domain: Hypotheses.
b) Merged hypothesis CF List of all Hypothesis: CFs.
=-= */
cf_hypothesis_merger([],Hypotheses,Hypotheses,CFs,CFs).
cf_hypothesis_merger([RID|Next_RID],Temp_Hypotheses,Hypotheses,Temp_CFs,CFs):-
    RID \= [],
    clause(rule_shell(RID,_,NewResult,NewCF),_),
    (
        /* IF the Result of current Rule is already in list: */
        (
            member(NewResult,Temp_Hypotheses),
            indexOf(Temp_Hypotheses,NewResult,Pos),
            nth1(Pos,Temp_CFs,OldCF),
            /* "100" because the range is 0-100 */
            Increased_CF is ((NewCF/100) + (OldCF/100) - ((NewCF/100) * (OldCF/100))) * 100,
            replaceInThePosition(Temp_CFs,Pos,Increased_CF,NewCFs),
            cf_hypothesis_merger(Next_RID,Temp_Hypotheses,Hypotheses,NewCFs,CFs)
        );
        /* IF the Result of current Rule is Not already in list: */
        (
            append(Temp_Hypotheses,[NewResult],New_Temp_Hypotheses),
            append(Temp_CFs,[NewCF],New_Temp_CFs),
            cf_hypothesis_merger(Next_RID,New_Temp_Hypotheses,Hypotheses,New_Temp_CFs,CFs)
)
).

/* =-=-=-=-=-=-=-=-=-=-=-=-= Περιγραφή Reasoner Pipeline (Predicate) =-=-=-=-=-=-=-=-=-=-=-=-=
```
Το τμήμα αυτό κώδικα, είναι υπεύθυνο για την αναδρομική κλήση του κατηγορήματος «Reasoner Extender», έως ότου δεν υπάρχουν περιθώρια πυροδότησης (περισσότερων) κανόνων. Επιστρέφει μία λίστα με τα αναγνωριστικά των κανόνων που δεν χρησιμοποιήθηκαν στη διαδικασία συλλογισμών.


```

```
Είσοδος κατηγορήματος:
    a) RID Domain: RID_Domain.
    b) All SWI Item Properties: SWI_IProperties.
    c) Temporary List RID Result list for recursion: Temp_NOTRDomain.
Έξοδος κατηγορήματος:
    a) List of Rule IDs that did not take part in reasoning: NOTRDomain.
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= */
reasoner_pipeline(RID_Domain,SWI_IProperties,Temp_NOTRDomain,NOTRDomain):-
    RID_Domain \= [],
    reasoner_extender(RID_Domain, SWI_IProperties,[],Decremented_RID_Domain, Augmented_SWI_IProperties),
    (
        (
            \+equal(RID_Domain,Decremented_RID_Domain),
            reasoner_pipeline(Decremented_RID_Domain,Augmented_SWI_IProperties,Decremented_RID_Domain,NOTRDomain)
        );
        (
            NOTRDomain = Temp_NOTRDomain
        )
    );true.

/* =-=-=-=-=-=-=-=-=-=-=-=-= Περιγραφή Reasoner Extender (Predicate) =-=-=-=-=-=-=-=-=-=-=-=-=
    Το κατηγόρημα αυτό, εκτελεί το κατηγόρημα «Reasoner Meta Prec» για να βρει όλες τις συνθήκες κανόνων που πυροδοτούνται. Έπειτα εκτελεί το κατηγόρημα «Reasoner Meta Shell» για να βρει όλους τους κανόνες που πυροδοτούνται (με βάση τις συνθήκες που πυροδοτήθηκαν). Εάν πυροδοτηθεί κάποιος κανόνας, τότε το αποτέλεσμά του καταχωρείται στη λίστα γνωρισμάτων, το αναγνωριστικό του διαγράφεται από τη λίστα εκτελέσιμων κανόνων και επιστρέφει τις δύο αυτές λίστες. Δηλαδή επιστρέφει τη νέα λίστα γνωρισμάτων και τη νέα λίστα εκτελέσιμων κανόνων. Αλλιώς επιστρέφει τις δύο αυτές λίστες χωρίς να τις αλλάξει.
Είσοδος κατηγορήματος:
    a) RID Domain: RID_Domain.
    b) All SWI Item Properties: SWI_IProperties.
    c) Empty List Variable will be used for comparison: Temp_RID_Domain.
Έξοδος κατηγορήματος:
    a) Decremented Rule ID Domain by the Triggered Rule ID: New_RID_Domain.
    b) Incremented SWI Item Properties by the result of triggered Rule ID: New_SWI_IProperties.
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= */
reasoner_extender(New_RID_Domain,New_SWI_IProperties,Old_RID_Domain,New_RID_Domain,New_SWI_IProperties):-
    equal(New_RID_Domain,Old_RID_Domain).

reasoner_extender(RID_Domain,SWI_IProperties,Temp_RID_Domain,New_RID_Domain,New_SWI_IProperties):-
    \+equal(RID_Domain,Temp_RID_Domain),
    /* Find all triggerable preconditions */
    reasoner_Meta_Prec(RID_Domain,SWI_IProperties,[],TPrecs),
    /* Find the first triggerable rule (from rule domain) based on triggerable preconditions */
    reasoner_Meta_Shell(RID_Domain,TPrecs,TRID),
    (
        /* IF triggerable rule id found */
        (
            number(TRID),
            clause(rule_shell(TRID,_,TRule_Result,_),_),
            /* TRule_Result = name_op_value_unit_duration */
            atomic_list_concat(LATRule_Result,'_',TRule_Result),
            /* LATRule_Result = [name,op,value,unit,duration] */
            LATRule_Result = [LRName,LROP,LRValue,LRUnit,LRDuration],
            atomize(LRName,ALRName),
            atomize(LROP,ALROP),
            numerize(LRValue,NLRValue),
            atomize(LRUnit,Temp_ALRUnit),
            atomize(LRDuration,Temp_ALRDuration),
            (
                (Temp_ALRUnit == 'null',ALRUnit = '');
                (ALRUnit = Temp_ALRUnit)
            ),
            (
                (Temp_ALRDuration == 'null',ALRDuration = '');
                (numerize(LRDuration,ALRDuration))
            ),
            /* !Critical: RESULT DSP = UNDEFINED = '' */
            New_IProperty = [ALRName,ALROP,NLRValue,ALRUnit,ALRDuration,''],
            (
                /* IF rule's result is already an item property: */
                (
                    member(New_IProperty,SWI_IProperties),
```

```
                    New_SWI_IProperties = SWI_IProperties
                );
                /* IF rule's result is not already an item property: */
                (
                    append(SWI_IProperties,[New_IProperty],New_SWI_IProperties)
                )
            ),
            delete(RID_Domain,TRID,New_RID_Domain),
            reasoner_extender(New_RID_Domain,New_SWI_IProperties,RID_Domain,New_RID_Domain,New_SWI_IProperties)
        );
        /* IF there is no triggerable rule id */
        (reasoner_extender(RID_Domain,SWI_IProperties,RID_Domain,New_RID_Domain,New_SWI_IProperties))
    ).

/* =-=-=-=-=-=-=-=-=-=-=-=-=-= Περιγραφή Reasoner Meta Shell (Predicate) =-=-=-=-=-=-=-=-=-=-=-=-=
    Το κατηγόρημα αυτό είναι υπεύθυνο για την εύρεση ενός κανόνα που ικανοποιείται. Ικανοποίηση κανόνα σημαίνει, ενοποίηση του κατηγορήματος «rule_shell» μα κάποιο συνδυασμό συνθηκών της λίστας ικανοποιημένων συνθηκών. Εάν ικανοποιηθεί κάποιος κανόνας, τότε το κατηγόρημα επιστρέφει το αναγνωριστικό του κανόνα που ικανοποιήθηκε. Αλλιώς επιστρέφει «empty».
Είσοδος κατηγορήματος:
    a) RID Domain: [RID|Next_RID].
    b) All Triggerable Preconditions: Triggered_Preconditions.
Έξοδος κατηγορήματος:
    a) First Triggerable Rule Found: RID.
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= */
reasoner_Meta_Shell([],_,_).
reasoner_Meta_Shell([RID|Next_RID],Triggered_Preconditions,New_RID):-
    RID \= [],
    /* Get Rule's shell precondition Map e.g: [P1,P2,P3,P4] */
    clause(rule_shell(RID,Shell_Precs,_,_),_),
    /* Building Triggered Precondition Map e.g: [true,P2,true,P3,P4] */
    meta_Shell_Map(RID,Triggered_Preconditions,Shell_Precs,New_Shell_Precs),
    (
        /* Triggered Preconditions are enought to trigger the Rule! */
        (
            rule_shell(RID,New_Shell_Precs,_,_),
            New_RID = RID,
            reasoner_Meta_Shell([],Triggered_Preconditions,New_RID)
        );
        /* Continue until find a triggerable Rule (or end of Rules) */
        (
            reasoner_Meta_Shell(Next_RID,Triggered_Preconditions,New_RID)
        )
    ).

/* Predicate meta_Shell_Map  Body */
meta_Shell_Map(_,[],EQUAL,EQUAL).
meta_Shell_Map(RID,[[CRID,CSID]|Next_RSet],Shell_Precs,Final_RPList):-
    (
        (
            CRID == RID,
            replaceInThePosition(Shell_Precs, CSID, 'true', New_Shell_Precs),
            meta_Shell_Map(RID,Next_RSet,New_Shell_Precs,Final_RPList)
        );
        (
            meta_Shell_Map(RID,Next_RSet,Shell_Precs,Final_RPList)
        )
    ).

/* =-=-=-=-=-=-=-=-=-=-=-=-=-= Περιγραφή Reasoner Meta Precondition (Predicate) =-=-=-=-=-=-=-=-=-=-=-=-=
        Το κατηγόρημα αυτό προσπαθεί για κάθε όρισμα της λίστας γνωρισμάτων, να ικανοποιήσει κάποια συνθήκη κάποιου κανόνα της λίστας εκτελέσιμων κανόνων τόσο λογικά όσο και χρονικά. Η λογική ικανοποίηση (της συνθήκης) βασίζεται στην επιτυχή πυροδότηση του κατηγορήματος «rule_prec». Η χρονική ικανοποίηση (της συνθήκης) βασίζεται στην επιτυχή πυροδότηση του κατηγορήματος «rule_prec_events». Όταν μία συνθήκη ικανοποιείται τότε το αναγνωριστικό της αποθηκεύεται σε μία λίστα ικανοποιημένων συνθηκών. Το κατηγόρημα επιστρέφει είτε μία λίστα από αναγνωριστικά (ικανοποιημένων) συνθηκών, είτε μία άδεια λίστα.
Είσοδος κατηγορήματος:
    a) List of Rule IDs that define the domain of Precondition to search for: RID_Domain.
    b) Two dimensional list of Item's Properties: SWI_IProperties.
    c) Initialize the list that will contain all triggered preconditions: Matched_Precs.
Έξοδος κατηγορήματος:
    a)  Final Matched Preconditions (Sorted): Final_Matched_Precs.
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= */
```

```
reasoner_Meta_Prec(_,[],Completed,Completed_Sorted):-
    sort(Completed,Completed_Sorted).
reasoner_Meta_Prec(RID_Domain, [IProp|Next_IProp],Matched_Precs,Final_Matched_Precs):-
    IProp = [PName,POp,PValue,PUnit,PDuration,SPDistance],

    /* !Critical: property duration = UNDEFINED = 'fail' */
    ((PDuration \='',numerize(PDuration,RMP_Duration));(RMP_Duration = 'fail')),
    /* !Critical: property dsp = UNDEFINED = -1 */
    ((SPDistance \='',numerize(SPDistance,RMP_SPDistance));(RMP_SPDistance is -1)),
    /* Try to match a precondition per loop */
    aggregate_all(bag([RID,SID]), meta_rule_prec(RID_Domain,PName,POp,PValue,PUnit,RMP_Duration,RMP_SPDistance,RID,SID),
    Triggered_Precs),
    (
        /* IF a new [RID,SID]: */
        (
            \+member(Triggered_Precs,Matched_Precs),
            append(Matched_Precs,Triggered_Precs,New_Matched_Precs),
            reasoner_Meta_Prec(RID_Domain,Next_IProp,New_Matched_Precs,Final_Matched_Precs)
        );
        /* IF [RID,SID] already exists: */
        (
            reasoner_Meta_Prec(RID_Domain,Next_IProp,Matched_Precs,Final_Matched_Precs)
        )
    ).

meta_rule_prec(RID_Domain,PName,POp,PValue,PUnit,PDuration,SPDistance,Return_RID,Return_SID):-
    (
        /* Case 1.0 Find a triggerable "rule_prec" */
        (
            /* Case 1.1 body solution 1: */
            (
                POp == '==',
                numerize(PValue,New_P_Value),
                rule_prec(Return_RID,Return_SID,PName,New_P_Value,PUnit)
            );
            /* Case 1.2 body solution 2: */
            (
                POp \= '==',
                big_decimal_number_to_atom(PValue,Atom_PValue),
                atom_concat(POp,' ',POp_white_space),atom_concat(POp_white_space,Atom_PValue,PV_Prec),
                rule_prec(Return_RID,Return_SID,PName,PV_Prec,PUnit)
            )
        ),
        /* Case 2.0 Find a triggerable "rule_prec_event" */
        (
            /* Case 2.1 predicate exists: */
            (
                /* Distance from start point (Disabled). */
                clause(rule_prec_event(Return_RID,Return_SID,_,_),_),
                rule_prec_event(Return_RID,Return_SID,_,PDuration)
            );
            /* Case 2.2 predicate does not exist: */
            (
                \+clause(rule_prec_event(Return_RID,Return_SID,_,_),_),true)
            )
        ),
        /* Case 3.0 already member of my executable rule domain */
        member(Return_RID,RID_Domain)
    ).
```

Παράρτημα Γ - SWI Prolog Διαχειριστής Βάσης Γνώσης

```
/* ================================================================================
                        1.  LOAD PREDICATES IN LOCAL MEMMORY (!Requirements: _)
 ================================================================================ */
load_Predicates_Memory(SCounter,PCounter,PECounter):-
 /* RETRACT ALL PREDICATES */
 retract_predicates('*',_,_),
 /* OPEN SQL CONNECTION */
 connect_SQL,
 /* SELECT RULE SHELL PREDICATES FROM SQL */
 select_rule_shell_SQL('*',_,RS_rslt),
 /* SELECT RULE PREC PREDICATES FROM SQL */
 select_rule_prec_SQL('*',_,RP_rslt),
 /* SELECT RULE PREC EVENTS PREDICATES FROM SQL */
 select_rule_prec_events_SQL('*',_,RPE_rslt),
 /* CLOSE SQL CONNECTION */
 disconnect_SQL,
 /* LOAD FETCHED RULE SHELL PREDICATES IN LOCAL MEMMORY */
 load_Predicates(RS_rslt,0,SCounter),
 /* LOAD FETCHED RULE PREC PREDICATES IN LOCAL MEMMORY */
 load_Predicates(RP_rslt,0,PCounter),
 /* LOAD FETCHED RULE PREC EVENT PREDICATES IN LOCAL MEMMORY */
 load_Predicates(RPE_rslt,0,PECounter).

/* ================================================================================
                        2.  INSERT A PREDICATE IN LOCAL MEMMORY (!Requirements: Rule ID)
 ================================================================================ */
add_Predicates_Memory(RULE_ID,SWI_Response):-
 /* RETRACT THE REQUESTED ID FOR SAFETY REASONS */
 retract_predicates('RULE_ID',RULE_ID,_),
 /* OPEN SQL CONNECTION */
 connect_SQL,
 /* SELECT RULE SHELL PREDICATES FROM SQL */
 select_rule_shell_SQL('RULE_ID',RULE_ID,RS_rslt),
 /* SELECT RULE PREC PREDICATES FROM SQL */
 select_rule_prec_SQL('RULE_ID',RULE_ID,RP_rslt),
 /* SELECT RULE PREC EVENTS PREDICATES FROM SQL */
 select_rule_prec_events_SQL('RULE_ID',RULE_ID,RPE_rslt),
 /* CLOSE SQL CONNECTION */
 disconnect_SQL,
 /* LOAD FETCHED RULE SHELL PREDICATES IN LOCAL MEMMORY */
 load_Predicates(RS_rslt,0,SCounter),
 /* LOAD FETCHED RULE PREC PREDICATES IN LOCAL MEMMORY */
 load_Predicates(RP_rslt,0,PCounter),
 /* LOAD FETCHED RULE PREC EVENT PREDICATES IN LOCAL MEMMORY */
 load_Predicates(RPE_rslt,0,_),
 (
  /* THE PRECONDITION HAS BEEN SUCCESSFULLY LOADED */
  (
      SCounter > 0,
      PCounter > 0,
      SWI_Response = 'success'
     );
     /* THE PRECONDITION FAILED TO BE LOAD */
     (
      SWI_Response = 'error'
     )
 ).

/* ================================================================================
                        3.  REMOVE A PREDICATE FROM LOCAL MEMMORY (!Requirements: Rule ID)
 ================================================================================ */
remove_Predicates_Memory(RID,SWI_Response):-
 /* RETRACT PREDICATES BY RULE ID */
 retract_predicates('RULE_ID',RID,SWI_Response).

/* _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_ */

 /* ========================================
        1. SQL - GET ALL RULE SHELL PREDICATES
```

```
   ============================================ */
select_rule_shell_SQL(Search_By, Search_Value, RSLT) :-
 (
  /* CASE 1. Search for all executable rules. */
     (
       Search_By == '*',
       SQL_Query = 'SELECT swi_rule_shell FROM rule INNER JOIN rule_version ON id = rule_version.rid WHERE execute = \'executable\''
     );
  /* CASE 2. Search for specific executable Rule. */
     (
       Search_By == 'RULE_ID',
       atomize(Search_Value,ASearch_Value),
       atom_concat('SELECT swi_rule_shell FROM rule INNER JOIN rule_version ON id = rule_version.rid WHERE execute = \'executable\'
AND
       id = ',ASearch_Value,SQL_Query)
     )
 ),
 findall(Rshell,odbc_query(ks_db,SQL_Query,row(Rshell)),RSLT).

/* ============================================
            2. SQL - GET ALL RULE PREC PREDICATES
   ============================================ */
select_rule_prec_SQL(Search_By, Search_Value, RSLT) :-
 (
  /* CASE 1. Search for all executable rules. */
     (
       Search_By == '*',
       SQL_Query = 'SELECT swi_rule_prec FROM rule_edges INNER JOIN rule_version ON rule_edges.rid = rule_version.rid WHERE execute =
       \'executable\''
     );
  /* CASE 2. Search for specific executable Rule. */
     (
       Search_By == 'RULE_ID',
       atomize(Search_Value,ASearch_Value),
       atom_concat('SELECT swi_rule_prec FROM rule_edges INNER JOIN rule_version ON rule_edges.rid = rule_version.rid WHERE execute
=
       \'executable\' AND rule_edges.rid = ',ASearch_Value,SQL_Query)
     )
 ),
 findall(RPrec,odbc_query(ks_db,SQL_Query,row(RPrec)),RSLT).


/* ============================================
        3. SQL - GET ALL RULE PREC EVENT PREDICATES
   ============================================ */
select_rule_prec_events_SQL(Search_By, Search_Value, RSLT) :-
 (
  /* CASE 1. Search for all executable rules. */
     (
       Search_By == '*',
       SQL_Query = 'SELECT swi_rule_prec_event FROM rule_edge_events INNER JOIN rule_version ON rule_edge_events.rid =
       rule_version.rid WHERE execute = \'executable\''
     );
  /* CASE 2. Search for specific executable Rule. */
     (
       Search_By == 'RULE_ID',
       atomize(Search_Value,ASearch_Value),
       atom_concat('SELECT swi_rule_prec_event FROM rule_edge_events INNER JOIN rule_version ON rule_edge_events.rid =
       rule_version.rid WHERE execute = \'executable\' AND rule_edge_events.rid = ',ASearch_Value,SQL_Query)
     )
 ),
 findall(RPrecEvnt,odbc_query(ks_db,SQL_Query,row(RPrecEvnt)),RSLT).

/* _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_ */

/* ==============================================
     1. LOAD PREDICATES IN MEMMORY - INNER PREDICATE
   ============================================== */
load_Predicates([],EQUAL,EQUAL).
load_Predicates([PREDICATE|NEXT_PREDICATE],Inner_Incr,Counter):-
 PREDICATE \= [],
```

```
 termize(PREDICATE,TERM_FIRST_PREDICATE),
 asserta(TERM_FIRST_PREDICATE),
 NewCounter is Inner_Incr + 1,
 load_Predicates(NEXT_PREDICATE,NewCounter,Counter).

/* ================================================
            2. UNLOAD PREDICATES FROM MEMMORY
  ================================================ */
retract_predicates(Request_Type, Request_Value, Response):-
 (
  /* IF PREDICATES EXIST */
  (
      /* IF REQUEST TYPE IS ALL PREDICATES */
      (
      Request_Type == '*',
      /* RETRACT RULE SHELL */
     retractall(rule_shell(_,_,_,_)),
     /* RETRACT RULE PREC */
     retractall(rule_prec(_,_,_,_,_)),
     /* RETRACT RULE PREC EVENT */
      retractall(rule_prec_event(_,_,_,_)),
      Response = 'success'
      );
      /* IF REQUEST TYPE IS RULE ID */
      (
      Request_Type == 'RULE_ID',
      /* RETRACT RULE SHELL */
      retractall(rule_shell(Request_Value,_,_,_)),
      /* RETRACT RULE PREC */
      retractall(rule_prec(Request_Value,_,_,_,_)),
      /* RETRACT RULE EVENT PREC */
      retractall(rule_prec_event(Request_Value,_,_,_)),
      Response = 'success'
      )
     );
     /* ELSE IF PREDICATES DOES NOT EXIST */
     (
      Response = 'error'
     )
 ).
```

Παράρτημα Δ - Παραχθείσα Βιβλιοθήκη Κατηγορημάτων SWI Prolog.

```
/*
=============================================================================================================
                                                Array / List Manipulation
=============================================================================================================
*/

/* ------------------------------------------------------------------------------------
        Replace the value of specific position in a list (Start point = 1)

        e.g: replaceInThePosition(+List,+Position,+Element,-NewList).
 ------------------------------------------------------------------------------------ */
replaceInThePosition([_|T],1,E,[E|T]).
replaceInThePosition([H|T],P,E,[H|R]) :- P > 1, NP is P-1, replaceInThePosition(T,NP,E,R).

/* ------------------------------------------------------------------------
                    Check if two lists are equal
                      e.g: equal(+List1,+List2).
 ------------------------------------------------------------------------ */
isSubset([],_).
isSubset([H|T],Y):-
  member(H,Y),
  select(H,Y,Z),
  isSubset(T,Z).
equal(X,Y):-
  is_list(X),
     is_list(Y),
  isSubset(X,Y),
  isSubset(Y,X).

/* ------------------------------------------------------------------------
                    Substract between two list
            e.g: sub(+ParentList,+ChildList,-NewList).
 ------------------------------------------------------------------------ */
sub(List,[],List).
sub(List,[X|Sub],Rem) :- select(X,List,Rem0), sub(Rem0,Sub,Rem).

/* ------------------------------------------------------------------------------------------
      Find (the first appear of "!") position of element in list (start point = 1)
                        e.g: indexOf(+List,+Element,-Position).
 ------------------------------------------------------------------------------------------ */
indexOf([Element|_], Element, 1):- !.
indexOf([_|Tail], Element, Index):- indexOf(Tail, Element, Index1), !, Index is Index1+1.

/*
=============================================================================================================
                                                ASCII Manipulation
=============================================================================================================
*/
 /* ============================================================================
                                    Atomize/2 Prolog Predicate
   ============================================================================ */
atomize(Any_Input,Atom_Output):-
(
     /*CASE 1: INPUT IS NOT AN ATOM */
     (
       \+(atom(Any_Input)),
       term_to_atom(Any_Input,Atom_Output)
      );
      /* CASE 2: INPUT IS AN ATOM */
      (
       atom(Any_Input),
       Atom_Output = Any_Input
      )
).

 /* ============================================================================
```

```
                                        termize/2 Prolog Predicate
   ================================================================================== */
  termize(Any_Input,Term_Output):-
(
      /* Case 1: Input is not a term */
      (
        \+(atom(Any_Input)),
        Term_Output = Any_Input
      );
       /* Case 2: Input is a term */
       (
         atom(Any_Input),
         term_to_atom(Term_Output,Any_Input)
       )
).

 /* ================================================================================
                                        numerize/2 Prolog Predicate

                                IF possible to be numeric then do it, ELSE do it atom!
  ================================================================================ */
numerize(Any_Input,Numeric_Output):-
(
      /* Case 1: IF numeric input */
      (
            number(Any_Input),
            Numeric_Output = Any_Input
      );
      /* Case 2.0. IF non numeric input */
      (
            \+number(Any_Input),
            atomize(Any_Input,Atomized_Input),
            (
                  /* Case 2.1 IF an atom number */
                  (
                        atom_number(Atomized_Input,Numeric_Output),
                        number(Numeric_Output)
                  );
                   /* Case 2.2 IF an atom alpha */
                   (
                        Numeric_Output = Atomized_Input
                  )
            )
      )
).

/*
=============================================================================================================
                                                SQL Queries
 ============================================================================================================
*/
/* ================================================================================
 ================================  CREATE CONNECTION  ================================
 Δημιουργία Σύνδεσης με βάση δεδομένων:
 Database Name: ks_db
 DNS Name: ks_db
 Username: cosmos
 Password: a123456789
*/
connect_SQL :- odbc_connect('ks_db', _, [user('cosmos'), alias(ks_db), password('a123456789'), open(once)]).

/* ================================================================================
 ================================  Destroy Connection  ================================
                    Αποσύνδεση απο την βάση δεδομένων
*/
disconnect_SQL:- odbc_disconnect(ks_db).

/*
=============================================================================================================
              BIG DECIMAL NUMBER -> ATOM ( 3.242555849230000000 -> '3.24255584923' or 3.000 -> '3.0' or 3 -> '3' )
```

```
================================================================================================================================
*/
big_decimal_number_to_atom(Numeric_Value,New_Atom_Value):-
 (
     (
          number(Numeric_Value),
          format(atom(Atom_Value), '~100f', [rational(Numeric_Value)]),
          regex('(0*$)', i,Atom_Value, Captures),[Chars_Capture] = Captures,atom_chars(XXX,Chars_Capture),
          atom_length(XXX,XXX_Length),
          sub_atom(Atom_Value,_, _, XXX_Length, S),
          ((XXX_Length > 0,Temp_var = S);(XXX_Length =< 0,Temp_var = Atom_Value)),
          ((atomic_list_concat(G,.,Temp_var),G = [New_Atom_Value|T], T =['']);(New_Atom_Value = Temp_var))
     );
     (
          \+number(Numeric_Value),
          atomize(Numeric_Value,New_Atom_Value)
     )
 ).

/*
================================================================================================================================
                                                RULE LIST ID -> RULE LIST RID ( e.g: [3,19,5] -> [R3,R19,R5] ).

================================================================================================================================
*/
id_to_rid([],[]).
id_to_rid([H1|T1],[H2|T2]):- atom_concat('R', H1, H2), id_to_rid(T1,T2).

/* ========================================================================
                                    Date Manipulation
 ======================================================================== */
     /*                          βρίσκει τα ΤΡΕΧΟΝ ημέρα,μήνας,έτος                          */
     /*                                  1ο όρισμα η ημέρα                                  */
     /*                                  2ο όρισμα ο μήνας                                  */
     /*                                  3ο όρισμα το έτος                                  */
  date_month_year(Second,Minute,Hour,Day,Month,Year) :-
    get_time(Stamp),
    /* μετατρέπει το Time Stamp σε DateTime ,ενώ το "Local" είναι το TimeZone*/
    stamp_date_time(Stamp, DateTime, local),
    date_time_value(second, DateTime, Second),
    date_time_value(minute, DateTime, Minute),
    date_time_value(hour, DateTime, Hour),
    date_time_value(day, DateTime, Day),
    date_time_value(month, DateTime, Month),
    date_time_value(year, DateTime, Year).
```

Παράρτημα Ε - Σχήμα Βάσης Δεδομένων

```
/* ========================================================== */
/*                          CREATE USER TABLES                */
/* ========================================================== */

/*-------------------- EXPERTS TABLE ---------------------*/
CREATE TABLE expert(
     id TEXT NOT NULL,
      auto_id SERIAL UNIQUE,
      platform TEXT NOT NULL,
      domain TEXT NOT NULL,
      root_domain TEXT NOT NULL,
      availability TEXT,
      date TEXT,
      primary key(id)
 );

/*-------------------- Personal Message ---------------------*/
CREATE TABLE personal_message(
      user_source TEXT,
      user_target TEXT,
      pm_date TEXT,
      pm_id SERIAL UNIQUE,
      personal_message TEXT,
      PRIMARY KEY(user_source, user_target,pm_date),
      FOREIGN KEY(user_source) REFERENCES expert(id),
      FOREIGN KEY(user_target) REFERENCES expert(id)
);

/*-------------------- Personal Message Notification ---------------------*/
CREATE TABLE personal_message_notification(
      user_source TEXT,
      user_target TEXT,
      pm_id SERIAL UNIQUE,
      msg_notification INTEGER,
      PRIMARY KEY(user_source, user_target),
      FOREIGN KEY(user_source) REFERENCES expert(id),
      FOREIGN KEY(user_target) REFERENCES expert(id)
);


/* ========================================================== */
/*                          CREATE ITEM TABLES                */
/* ========================================================== */

CREATE TABLE item(
      auto_id SERIAL UNIQUE,
      id TEXT UNIQUE, /* mikegian2008@hotmail.com */
      owner TEXT NOT NULL, /* mgian_iPHR */
      name TEXT NOT NULL, /* e.g: Mgian*/
      status TEXT NOT NULL, /* critical, warning, good */
      root_domain TEXT NOT NULL, /* (root must be same as expert's knowledge root) patients */
      create_date TEXT NOT NULL, /* 28-01-2019 */
      update_date TEXT NOT NULL,
      total_diagnoses TEXT NOT NULL,
      deletable VARCHAR(5) NOT NULL,
      item_platform TEXT,
      PRIMARY KEY(id, owner),
      FOREIGN KEY(owner) REFERENCES expert(id)
);

CREATE TABLE item_properties(
      item_id TEXT NOT NULL,
      owner TEXT NOT NULL,
      prop_name TEXT NOT NULL,
      prop_start_date TEXT NOT NULL,
      auto_id SERIAL UNIQUE,
      prop_op TEXT NOT NULL,
      prop_value TEXT NOT NULL,
```

```
    prop_unit TEXT,
    prop_end_date TEXT,
    prop_duration TEXT, /* !important: SECONDS */
    deletable VARCHAR(5) NOT NULL,
    prop_platform TEXT NOT NULL,
    PRIMARY KEY(item_id, owner, prop_name,prop_start_date),
    FOREIGN KEY(item_id) REFERENCES item(id),
    FOREIGN KEY(owner) REFERENCES expert(id)
);


/* =============== ===============*/
CREATE TABLE item_execution(
    item_id TEXT NOT NULL,
    owner TEXT NOT NULL,
    execution_date TEXT NOT NULL,
    auto_id SERIAL UNIQUE,
    json_diagnosis TEXT NOT NULL, /* {RIDs,Hypotheses, CFs, Approves} */
    PRIMARY KEY(item_id,owner,execution_date),
    FOREIGN KEY(item_id) REFERENCES item(id),
    FOREIGN KEY(owner) REFERENCES expert(id)
);

/* ========================================================== */
/*                          CREATE RULE TABLES                */
/* ========================================================== */

/*---------------------- RULE TABLE ----------------------*/
CREATE TABLE rule(
    id INTEGER NOT NULL,
    json_graph TEXT NOT NULL,
    swi_rule_shell TEXT NOT NULL,
    dnf TEXT NOT NULL,
    referer TEXT NOT NULL,
    root_referer TEXT NOT NULL,
    precodition TEXT NOT NULL,
    result TEXT NOT NULL,
    score FLOAT NOT NULL,
    description TEXT,
    license TEXT,
    rule_cf TEXT,
    json_trc TEXT,
    PRIMARY KEY(id)
);

/*---------------------- RULE EDGES TABLE (Precondition Table) ----------------------*/
CREATE TABLE rule_edges(
    rid INTEGER NOT NULL,
    sid INTEGER NOT NULL,
    precodition_data TEXT,
    precodition_value TEXT,
    precodition_unit TEXT,
    swi_rule_prec TEXT,
    tf FLOAT ,
    PRIMARY KEY(rid,sid),
    FOREIGN KEY(rid) REFERENCES rule(id)
);

/*---------------------- RULE EDGES TIME EVENTS TABLE (Precondition's time relativity Table) ----------------------*/
CREATE TABLE rule_edge_events(
    rid INTEGER NOT NULL,
    sid INTEGER NOT NULL,
    swi_rule_prec_event TEXT NOT NULL, /* Duration in seconds + Distance from Start Point in seconds!*/
    PRIMARY KEY(rid,sid,swi_rule_prec_event),
    FOREIGN KEY(rid) REFERENCES rule(id)
);

/*---------------------- RULE VERSION TABLE ----------------------*/
CREATE TABLE rule_version(
    rid INTEGER NOT NULL,
    create_date TEXT NOT NULL,
```

```
    update_date TEXT NOT NULL,
    version FLOAT NOT NULL,
    version_editor TEXT NOT NULL,
    notification TEXT,
    status TEXT, /* clone or original */
    editable TEXT,
    execute TEXT,
    primary key(rid),
    FOREIGN KEY(rid) REFERENCES rule(id),
    FOREIGN KEY(version_editor) REFERENCES expert(id)
);

/*---------------------- RULE PERMISSION TABLE ----------------------*/
CREATE TABLE rule_permission(
    rid INTEGER NOT NULL,
    editor TEXT NOT NULL,
    editor_permission INTEGER NOT NULL, /* 0 = owner, 2 = editor */
    primary key(rid,editor),
    FOREIGN KEY(rid) REFERENCES rule(id),
    FOREIGN KEY(editor) REFERENCES expert(id)
);

/*--------------------- Panding Requests ---------------------*/
CREATE TABLE request_pending(
    request_id SERIAL UNIQUE,
    source TEXT,
    target TEXT,
    request_type TEXT,
    message TEXT,
    request_date TEXT,
    primary key(source, target, request_type)
);

/*---------------------- Rule Clones ----------------------*/
CREATE TABLE rule_clone(
    parent_id INTEGER,
    child_id INTEGER,
    primary key(parent_id,child_id),
    FOREIGN KEY(parent_id) REFERENCES rule(id)
);

/*---------------------- RULE UPDATE MODE ----------------------*/
CREATE TABLE rule_update_mode(
    rid INTEGER,
    update_date_on TEXT,
    update_editor TEXT,
    primary key(rid),
    FOREIGN KEY(rid) REFERENCES rule(id)
);

/*--------------------- Forum ---------------------*/
CREATE TABLE rule_comment(
    comment_id SERIAL UNIQUE,
    forum_id INTEGER,
    comment_date TEXT,
    editor TEXT,
    comment_content TEXT,
    comment_title TEXT,
    comment_likes INTEGER,
    comment_dislikes INTEGER,
    primary key(forum_id, comment_id, comment_date),
    FOREIGN KEY(forum_id) REFERENCES rule(id),
    FOREIGN KEY(editor) REFERENCES expert(id)
);

/*--------------------- LIKE DISLIKE COMMENTS ---------------------*/
CREATE TABLE rule_comment_popularity(
    fc_id INTEGER NOT NULL,
    c_id INTEGER NOT NULL,
    comment_popularity TEXT,
    comment_user TEXT,
```

```
    primary key(fc_id, c_id, comment_user, comment_popularity),
    FOREIGN KEY(fc_id) REFERENCES rule(id),
    FOREIGN KEY(c_id) REFERENCES rule_comment(comment_id),
    FOREIGN KEY(comment_user) REFERENCES expert(id)
);


/*-------------------- COLLABORATION CHAT ---------------------*/
CREATE TABLE rule_collaboration_chat(
    rid INTEGER,
    editor TEXT,
    chat_date TEXT,
    chat_content TEXT,
    id SERIAL UNIQUE,
    primary key(rid, editor, chat_date),
    FOREIGN KEY(rid) REFERENCES rule(id),
    FOREIGN KEY(editor) REFERENCES expert(id)
);

/*-------------------- COLLABORATION RULE DATA ---------------------*/
CREATE TABLE rule_collaboration_data(
    cid INTEGER NOT NULL,
    description TEXT,
    license TEXT,
    version TEXT,
    notification TEXT,
    referer TEXT,
    cf TEXT,
    PRIMARY KEY(cid),
    FOREIGN KEY(cid) REFERENCES rule(id)
);

/*-------------------- COLLABORATION RULE EDITORS ---------------------*/
CREATE TABLE rule_collaboration_editors(
    cid INTEGER NOT NULL,
    editor_name TEXT NOT NULL,
    json_graph TEXT,
    json_trc TEXT,
    result TEXT,
    precodition TEXT,
    PRIMARY KEY(cid, editor_name),
    FOREIGN KEY(cid) REFERENCES rule(id),
    FOREIGN KEY(editor_name) REFERENCES expert(id)
);

/*-------------------- EVENT (VIEW) HANDLER ---------------------*/
CREATE TABLE event_handler(
    e_id SERIAL UNIQUE,
    rid INTEGER NOT NULL,
    event_title TEXT,
    event_message TEXT,
    event_permission TEXT, /* owner or editor level*/
    event_date TEXT,
    PRIMARY KEY(rid,event_message,event_date)
);
```

Παράρτημα ΣΤ – Οδηγίες Εγκατάστασης του Συστήματος COSMOS

ΣΤ.1. Προετοιμασία εγκατάστασης του συστήματος

Προτού περάσουμε στην εγκατάσταση του συστήματος σε κάποιο ηλεκτρονικό υπολογιστή, απαιτείται η εγκατάσταση ενός συνόλου από προγράμματα:

1. Πρόγραμμα Αποσυμπίεσης κατάληξης «*rar*». Προτεινόμενα προγράμματα για το σκοπό αυτό είναι το WinRAR (https://www.win-rar.com/download.html) ή PowerISO (https://www.poweriso.com/download.php).
2. Προγράμματα για την εγκατάσταση της Σχεσιακής Βάσης Δεδομένων του συστήματος αλλά και τη διαχείριση αυτής. Προτεινόμενα για το σκοπό αυτό είναι το PostgreSQL (https://www.postgresql.org/) μαζί με το pgAdmin (https://www.pgadmin.org/). Επιπλέον, κατά την εγκατάσταση της PostgreSQL θα πρέπει να δημιουργηθεί ένας χρήστης με username: "***cosmos***" και password: "***a123456789***". Τέλος θα χρειαστεί να δημιουργηθεί μία βάση δεδομένων (όπου ο παραπάνω χρήστης να είναι owner) με ονομασία «***ks_db***». Το σχήμα της βάσης αυτής θα οριστεί στην επόμενη ενότητα.
3. Εγκατάσταση του ODBC Driver για την επικοινωνία της SWI Prolog με την βάση δεδομένων. Στη περίπτωση της PostgreSQL απαιτείται (https://odbc.postgresql.org/).
4. Java 8 (http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html) για την αλληλεπίδραση με το Java Server του συστήματος και Java IDE για την επεξεργασία του Java κώδικα του συστήματος. Προτεινόμενος IDE είναι το InteliJIDEA (https://www.jetbrains.com/idea/download/ - section=windows) ή NetBeans (https://netbeans.org /downloads/).
5. SWI Prolog (http://www.swi-prolog.org/download/stable) για την έναρξη και επεξεργασία του SWI Server του συστήματος.

ΣΤ.2. Εγκατάσταση Συστήματος

Το CD που περιλαμβάνει την πτυχιακή αυτή εργασία, περιλαμβάνει επίσης ένα συμπιεσμένο αρχείο με ονομασία «*COSMOS – The Knowledge System.rar*». Αποσυμπιέζοντας το αρχείο αυτό, σε κάποιο εύκολα προσβάσιμο σημείο του υπολογιστή π.χ. «C:\», παράγεται ένας φάκελος με ονομασία «*COSMOS – The Knowledge System*». Ο φάκελος αυτός εμπεριέχει τέσσερεις φακέλους «*Java Server*», «*SWI Server*», «*Solr Server*», «*SQL Server*» όπως φαίνεται στη παρακάτω εικόνα.

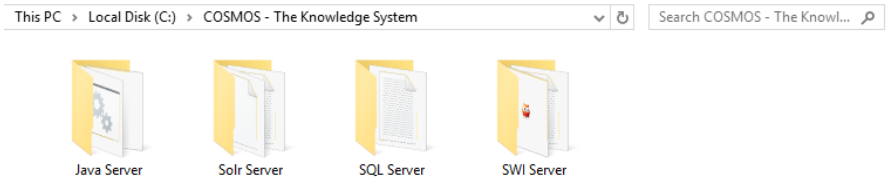

Figure 74 – Όλοι οι φάκελοι που περιλαμβάνει το αποσυμπιεσμένο αρχείο.

Ακολουθούν πέντε ενέργειες για την εγκατάσταση του συστήματος σε κάποιο υπολογιστή. Οι ενέργειες αυτές απαιτούν τόσο την ολοκλήρωση των βημάτων της προηγούμενης ενότητας όσο και την παραπάνω αποσυμπίεση:

6. ***Δημιουργία σχήματος της βάσης δεδομένων «ks_db»***

Αρχικά ξεκινάμε το πρόγραμμα το pgAdmin4 και επιλέγουμε τη βάση δεδομένων «ks_db» που βρίσκεται στην αριστερή πλευρά. Έπειτα επιλέγουμε από το μενού επιλογών «Tools» ─ «Query Tool».

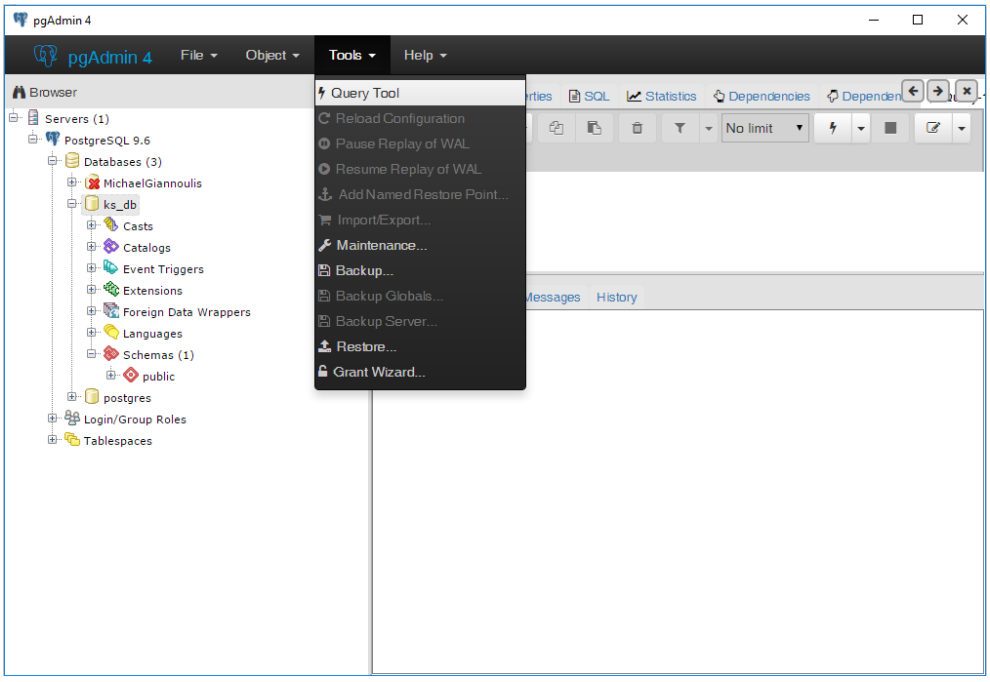

Figure 75 – pgAdmin4 – Επιλογή «Query Tool» για τη βάση δεδομένων «ks_db».

Στη συνέχεια ανοίγουμε το φάκελο «SQL Server» (που παράχθηκε κατά την αποσυμπίεση). Εκεί θα δούμε ένα αρχείο τύπου txt, με ονομασία «SQL_SCHEMA_ks_db». Ανοίγουμε το αρχείο αυτό (με κάποιο notepad) και κάνουμε αντιγραφή όλες τις γραμμές που περιέχει (ctrl+a, ctrl+c).

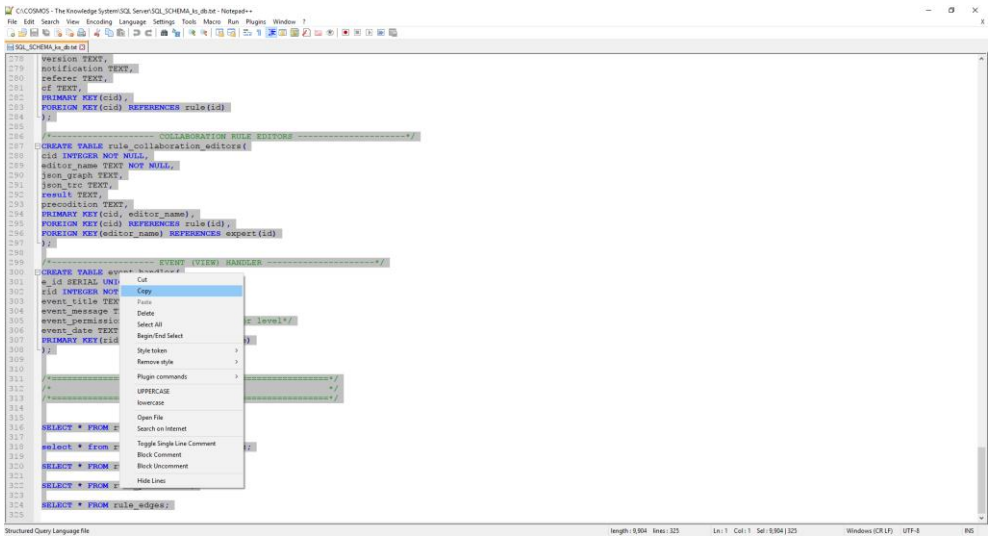

Figure 76 – Αντιγραφή SQL κώδικα σχήματος για τη βάση δεδομένων «ks_db».

Έπειτα επιστρέφουμε πίσω στο pgAdmin4 και κάνουμε επικόλληση (ctrl+v) τον κώδικα, στο παράθυρο που άνοιξε μέσω της επιλογής "Query Tool". Τέλος πατάμε το κουμπί «Execute/Refresh» και τρέχουμε τον κώδικα αυτό.

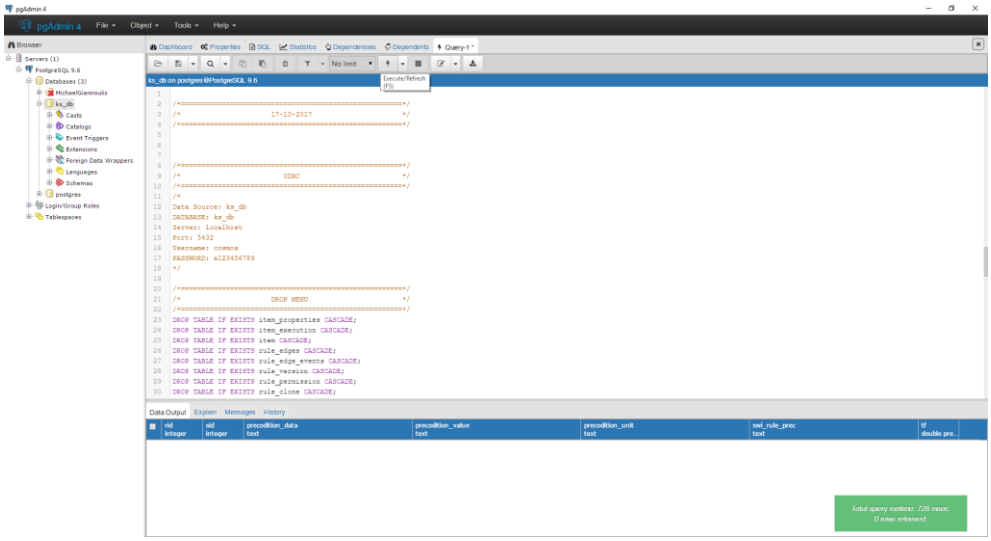

Figure 77 – pgAdmin4 – Επικόλληση κώδικα για τη δημιουργία του σχήματος της βάσης «ks_db».

7. *Δημιουργία DSN στον PostgreSQL ODBC Driver*

Αρχικά ανοίγουμε το πρόγραμμα «ODBC Data Sources (64-bit) ή (32-bit)» το οποίο στο λειτουργικό σύστημα Windows, βρίσκεται στο «*C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Administrative Tools*», επιλέγουμε «System DSN» και πατάμε το κουμπί «Add…».

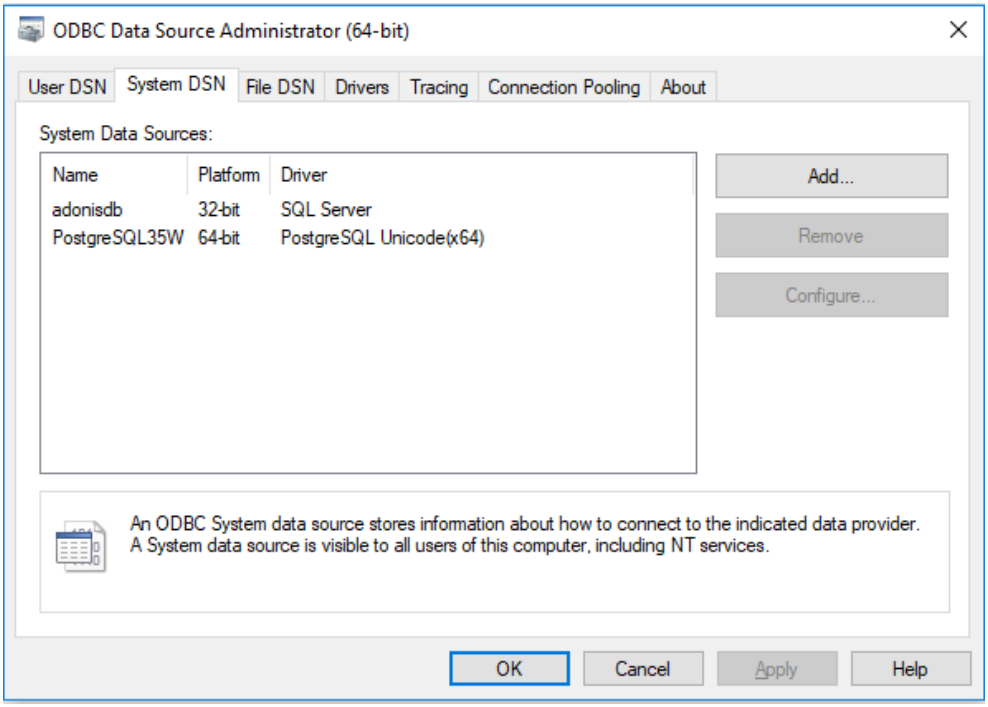

Figure 78 – ODBC Driver (64 bit) - System DSN.

Στη συνέχεια από το μενού επιλογών που θα εμφανιστεί επιλέγουμε «PostgreSQL Unicode(x64)» και πατάμε το κουμπί «Finish».

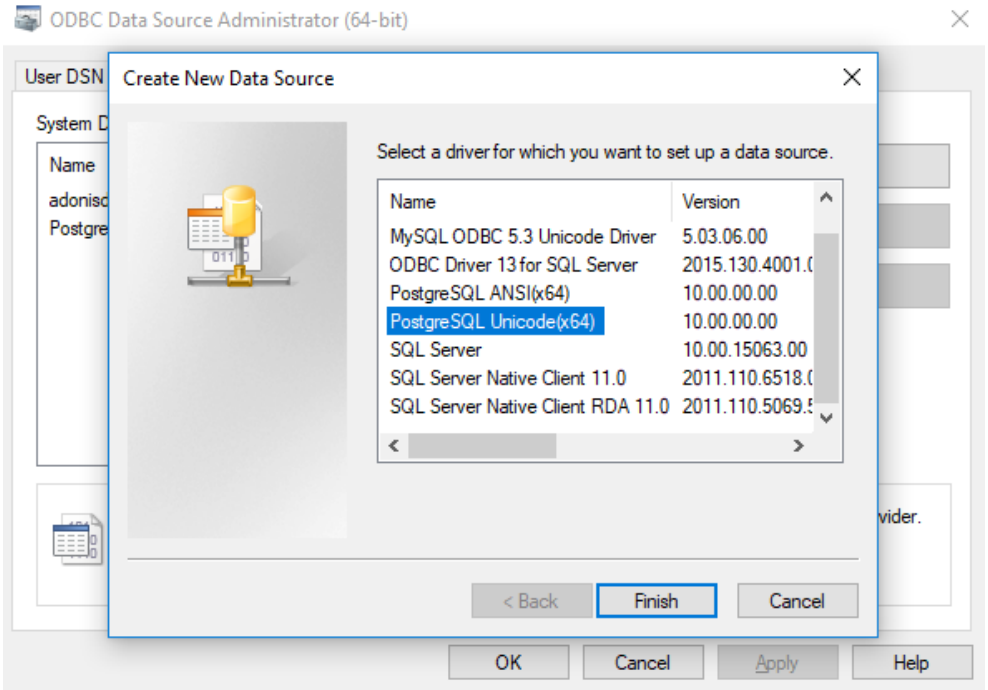

Figure 79 – ODBC Driver (64 bit) - Create New Data Source.

Έπειτα θα μας εμφανίσει σε νέα καρτέλα, μία φόρμα με ορισμένα πεδία που πρέπει να συμπληρώσουμε ως εξής:

- Data Source: ks_db
- Database: ks_db
- Server: 127.0.0.1
- Username: cosmos
- Port: 5432
- Password: a123456789

Τέλος πατάμε το κουμπί «Save» και κλείνουμε το παράθυρο. Το βήμα αυτό θα δημιουργήσει ένα «Data Source Name» με ονομασία «ks_db», το οποίο χρειάζεται η SWI Prolog για την επικοινωνία της με τη βάση δεδομένων του συστήματος.

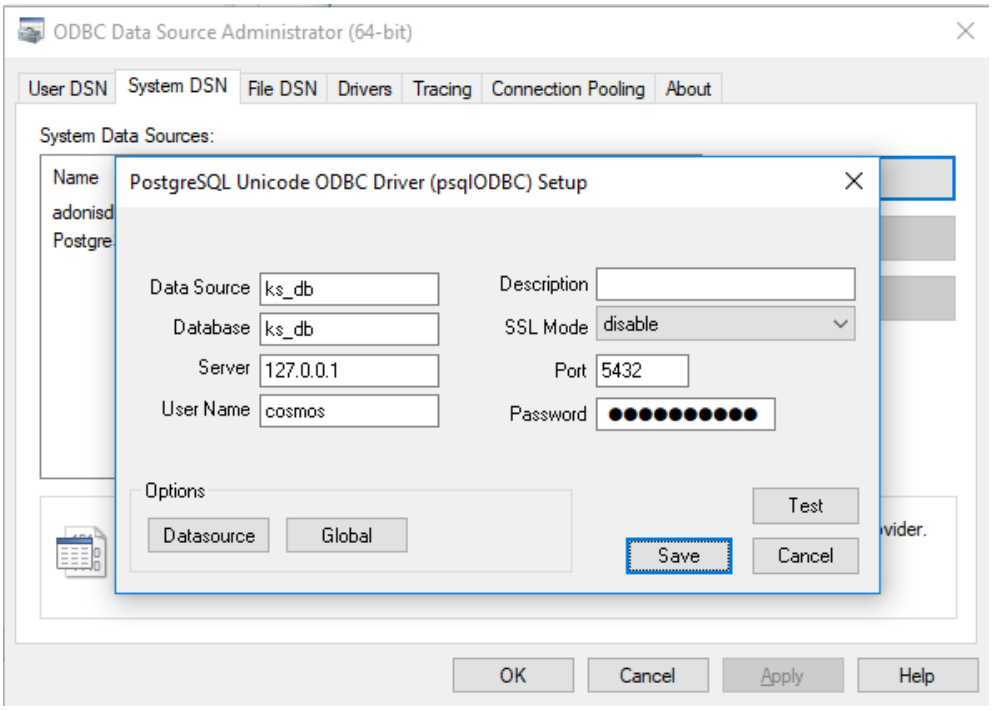

Figure 80 – ODBC Driver (64bit) – φόρμα συμπλήρωσης πεδίων για δημιουργία DSN.

8. *Εγκατάσταση SOLR Server*

Ο φάκελος «*Solr Server*» αντιπροσωπεύει την «*πλατφόρμα εύρεσης οντολογιών*». Αρχικά ανοίγουμε το command line των Windows και εντοπίζουμε το φάκελο «bin» που βρίσκεται εντός του φακέλου «Solr Server».

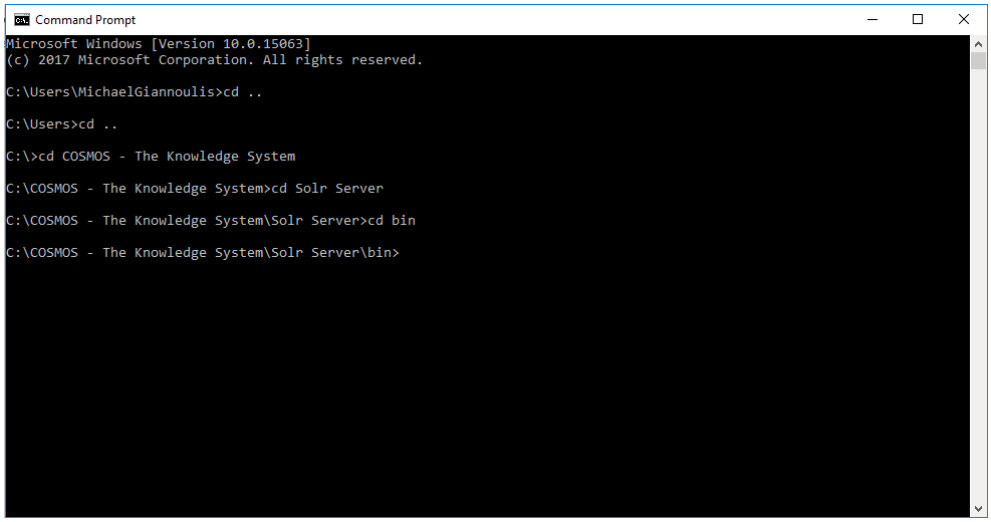

Figure 81 – CMD - Εύρεση μονοπατιού Solr Server \ bin.

Στη συνέχεια γράφουμε την εντολή «solr.cmd start» και πατάμε το κουμπί «enter». Ο solr server πλέον τρέχει με επιτυχία στο port 8983 (test: http://127.0.0.1:8983/). Εάν θέλουμε να κλείσουμε τον solr server, τότε αντί για «solr.cmd start» πληκτρολογούμε «solr stop –p 8983».

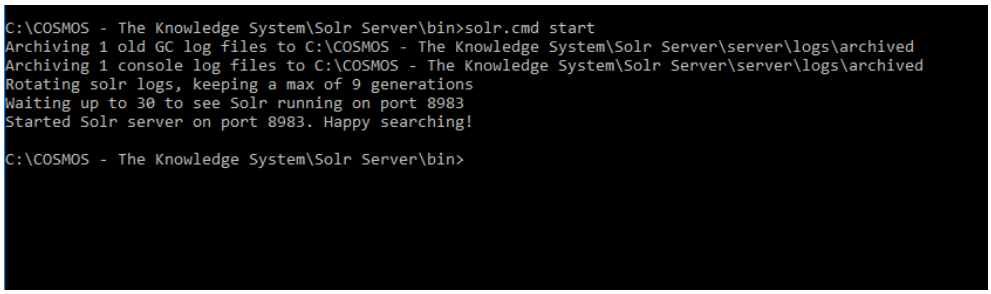

Figure 82 – CMD - Έναρξη Solr Server στο port 8983.

9. ***Εγκατάσταση Java (Jetty) Server***

Ανοίγουμε το φάκελο «Java Server» και τρέχουμε το αρχείο «start-jar.cmd» για την έναρξη του server ή τρέχουμε το αρχείο «stop-jar.cmd» για το κλείσιμο του server. Ο server τρέχει στο port 8082 (test: http://127.0.0.1:8082/).

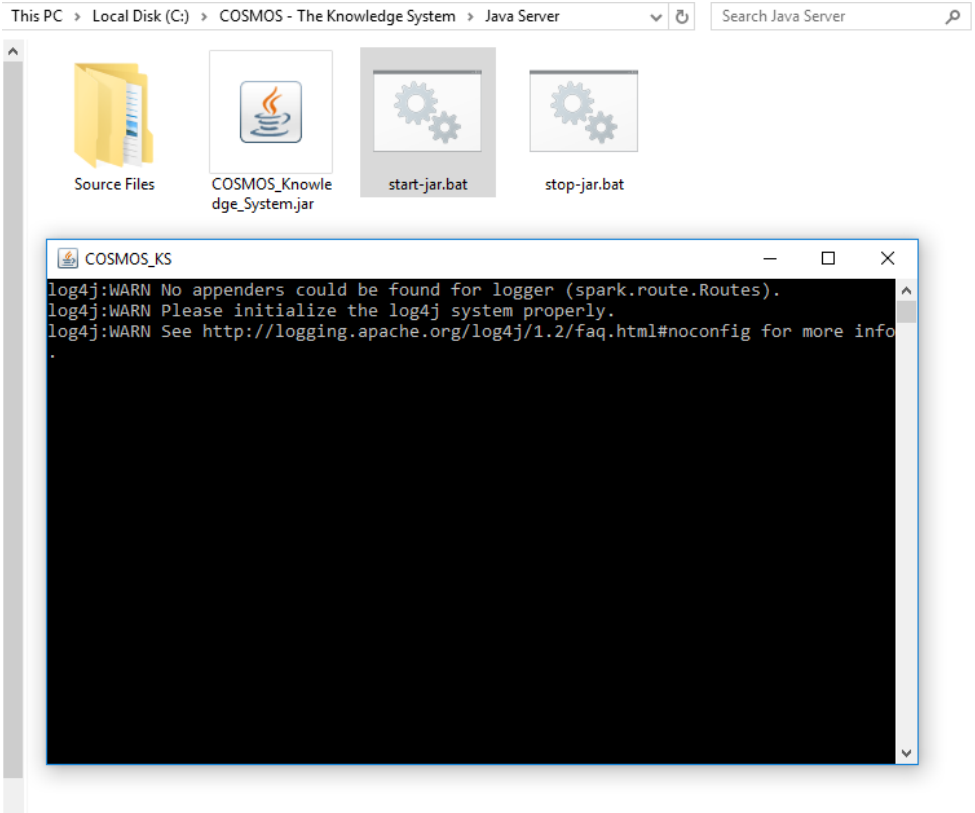

Figure 83 – Έναρξη Java (Jetty) Server στο port 8082.

10. ***Εγκατάσταση SWI Server***

Αρχικά ανοίγουμε το φάκελο «SWI Server» και εκτελούμε το «reasoner.pl». Εξαετίας μίας third-party βιβλιοθήκης που χρησιμοποιούμε για τα regex, απαιτείται εγκατάσταση αυτής γράφοντας την εντολή «pack_install(regex).». Έπειτα θα προκύψει ένας μικρός διάλογος με το σύστημα στον οποίο η ακολουθία των απαντήσεών μας είναι «Y», «1», «Y», έτσι όπως φαίνεται στην εικόνα 84. Εφόσον όλα κυλήσουν όλα ομαλά κλείνουμε το «reasoner.pl».

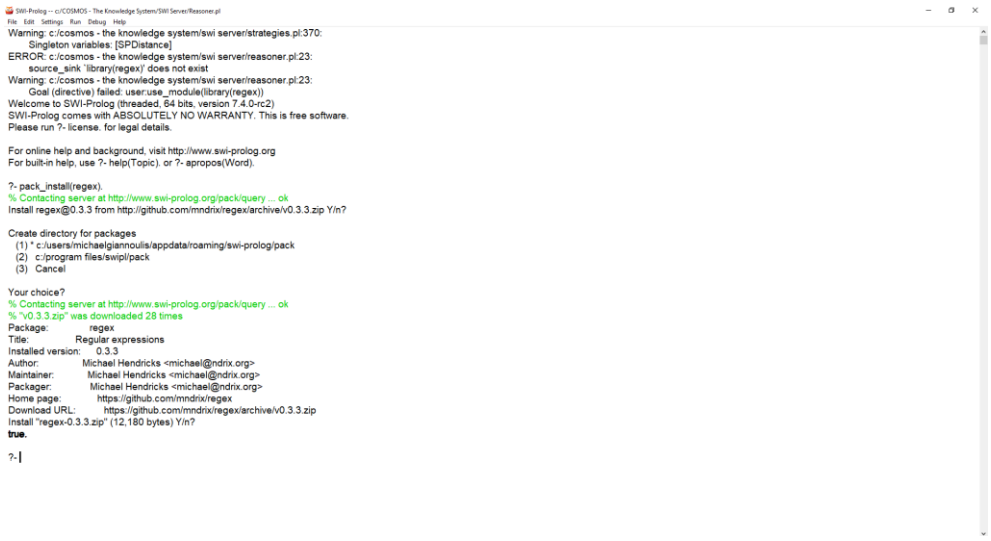

Figure 84 – SWI Server - Εγκατάσταση βιβλιοθήκης "regex".

Ανοίγουμε το αρχείο «reasoner.pl» και γράφουμε την εντολή «reasoner». Με το τρόπο αυτό ο SWI Server ξεκινάει στο port 8085. Για το κλείσιμο του server αρκεί να γράψουμε την εντολή «halt».

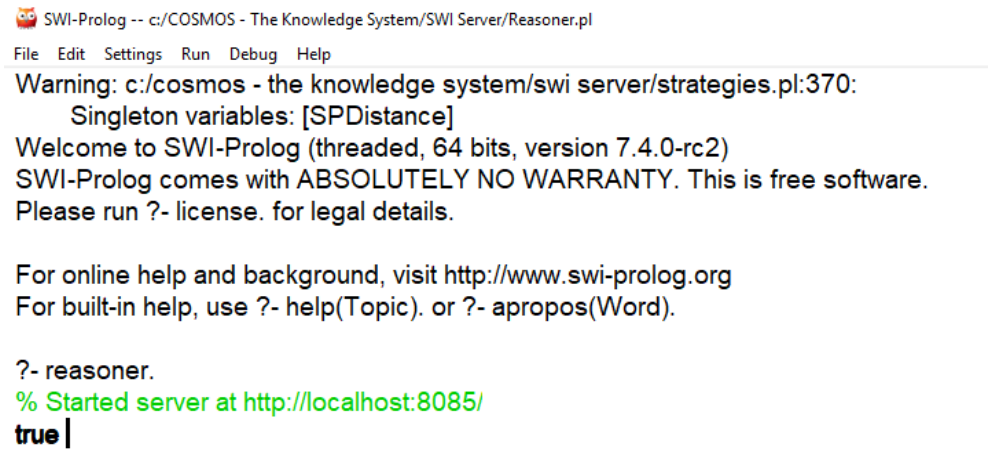

Figure 85 – Έναρξη του SWI Server στο port 8085.

Το σύστημα COSMOS είναι πλέον προσβάσιμο τοπικά μέσω browser (Chrome, Mozilla etc.) στο url: http://127.0.0.1:8082/. Αντί για localhost θα μπορούσαμε να συνδεθούμε με την IPV4 του υπολογιστή. Για παράδειγμα http://192.168.1.11:8082/. Στη περίπτωση όμως που θέλαμε απομακρυσμένη πρόσβαση μέσω κάποιου dns (π.χ. cosmos.gr), θα μπορούσαμε να αποκτήσουμε ένα (έως πέντε) dns δωρεάν από το DuckDNS https://www. duckdns.org/.