ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ ΜΕ ΒΑΣΗ ΤΑ ΣΥΝΔΕΔΕΜΕΝΑ ΑΝΟΙΚΤΑ ΔΕΔΟΜΕΝΑ

Διπλωματική Εργασία

Μουζά Ευαγγελία

Θεσσαλονίκη, Ιούνιος 2019

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ ΜΕ ΒΑΣΗ ΤΑ ΣΥΝΔΕΔΕΜΕΝΑ ΑΝΟΙΚΤΑ ΔΕΔΟΜΕΝΑ

Μουζά Ευαγγελία

Πτυχίο Μηχανικού Πληροφορικής και Τηλεπικοινωνίων, Πανεπιστήμιο Δυτικής Μακεδονίας, 2014

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Ταμπούρης Ευθύμιος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την …/6/2019

Ονοματεπώνυμο	Ονοματεπώνυμο	Ονοματεπώνυμο
.....	.....	.....

Μουζά Ευαγγελία

.....

Ευχαριστίες

Θα ήθελα να ευχαριστήσω των καθηγητή Ταμπούρη Ευθύμιο ο οποίος επέβλεψε την διπλωματική μου εργασία και με βοήθησε να την ολοκληρώσω με τη συμβολή και καθοδήγησή του.

Επίσης, θα ήθελα να ευχαριστήσω την οικογένεια μου για την πνευματική υποστήριξη που μου προσέφερε καθ' όλη τη διάρκεια των σπουδών μου και ειδικότερα την περίοδο της συγγραφής της παρούσας διπλωματικής καθώς επίσης και τους συμφοιτητές και φίλους μου για τις όμορφες στιγμές που περάσαμε μέσα και έξω από τη σχολή.

Μουζά Ευαγγελία

Θεσσαλονίκη, Ιούνιος 2019

Περίληψη

Καθημερινά, δημόσιοι φορείς παράγουν, συγκεντρώνουν και διαθέτουν ένα μεγάλο εύρος δεδομένων και πληροφοριών σχετικά με τις υπηρεσίες τους σε διαφορετικούς τομείς της δημόσιας ζωής. Η διάθεση αυτή των δεδομένων γίνεται πλέον και μέσω δικτυακών πυλών ηλεκτρονικής διακυβέρνησης, ωστόσο τις περισσότερες φορές είναι αδόμητη και δεν μπορεί να αναγνωστεί από τις μηχανές (ηλεκτρονικούς υπολογιστές).

Τα τελευταία χρονιά αυξάνεται ολοένα και περισσότερο και στη χωρά μας η δημοσίευση Συνδεδεμένων Ανοικτών Δεδομένων από τους Δημοσίους φορείς μας και είναι αυξανομένη και η ανάγκη της ανάγνωσης και κατ' επέκταση επεξεργασίας αυτών των δεδομένων από τους πολίτες.

Η παρούσα διπλωματική εργασία έχει ως κύριο στόχο την δημιουργία εφαρμογών που θα βοηθήσουν προς αυτή την κατεύθυνση. Συγκεκριμένα, οι εφαρμογές αυτές είναι εφαρμογές που επικοινωνούν με τα αποθετήρια στα οποία βρίσκονται Συνδεδεμένα Ανοικτά Δεδομένα δημοσίων υπηρεσιών ενός πιλοτικού γράφου του Πανεπιστήμιου Μακεδονίας και εξάγουν ερωτήματα που ενδιαφέρουν τους πολίτες σε μορφή αναγνώσιμη από τον καθένα.

Συνοψίζοντας τα βασικά στοιχεία αυτής της διπλωματικής εργασίας, επικεντρώνονται σε τρεις θεματικούς άξονες. Κατά τον πρώτο άξονα, αναλύεται το θεωρητικό υπόβαθρο πάνω στο οποίο στηρίζεται η παρούσα εργασία. Ο δεύτερος άξονας αναφέρεται στις τεχνολογίες και στις μεθόδους που χρησιμοποιήθηκαν ώστε να ληφθούν τα αποτελέσματα. Ο τρίτος άξονας περιστρέφεται γύρω από την περιγραφή και την ανάλυση των εφαρμογών.

Λέξεις κλειδιά: *Ανοικτά Συνδεδεμένα Δεδομένα, δημοσιές υπηρεσίες, CPSV-AP*

Abstract

Every day, public services produce, collect and publish a wide range of data and information on their services in different areas of public life. This data is now available through eGovernment gateways, in the form of Linked Open Data, but it is often unstructured.

In recent years, the publication of Linked Open Data by our Public Services has been increasing and the need to read and process those data by citizens is increasing.

The main goal of this Diploma Thesis is to create applications that will help in this direction. In particular, these applications are applications that communicate with the repositories in which the Linked Open Data of public services of a pilot project of the University of Macedonia and export questions that are of interest to the citizens in a form readable by each one.

The basic elements of this Diploma Thesis focus on three main topics. The first topic analyses the theoretical background on which this paper is based. The second topic refers to the technologies and methods used to create the applications. The third topic revolves around the description and analysis of the applications.

Keywords: *Linked Open Data, public services, CPSV-AP*

Περιεχόμενα

Πίνακας Εικόνων

Πίνακας Πινάκων

1. ΕΙΣΑΓΩΓΗ

Ανοικτά Δεδομένα των Δημοσίων Υπηρεσιών είναι όλες οι πληροφορίες που συλλέγονται από τους φορείς και διατίθενται στο ευρύ κοινό για λογούς διαφάνειας, ενημέρωσης η δημοσίου συμφέροντος. Παραδείγματα είναι διοικητικά έγγραφα όπως εγκύκλιοι και Υπουργικές αποφάσεις, πολεοδομικές και οικονομικές μελέτες, γεωργικά δεδομένα όπως χάρτες, στατιστικά δεδομένα και ερευνητικά δεδομένα.

Εξ' ορισμού τα ανοικτά δεδομένα πρέπει να είναι προσβάσιμα, δίχως περιορισμούς για το κοινό. Με αυτόν τον τρόπο ενισχύεται σημαντικά η διαφάνεια στο δημόσιο όπως επίσης μειώνεται και το κόστος από την αποδοτικότερη λειτουργία του συνόλου των δημοσίων υπηρεσιών, καθώς οι δημόσιοι φορείς αποκτούν πλέον άμεση πρόσβαση σε δεδομένα, δυνατότητα που δεν είχαν πριν.

Οι πολίτες, οι οργανισμοί και οι επιχειρήσεις έχουν πρόσβαση επομένως σε κυβερνητικές διαδικτυακές πύλες και μπορούν να ενημερωθούν για τις δημόσιες υπηρεσίες που ενδιαφέρονται. Ωστόσο, ακόμη και στην ίδια χώρα, οι δημόσιες υπηρεσίες τεκμηριώνονται με βάση διαφορετικά περιφερειακά ή τοπικά μοντέλα. Επομένως, κάποιες φορές οι πληροφορίες που αναζητούνται μπορεί να είναι αντικρουόμενες και λόγω αυτού να μην θεωρούνται χρήσιμες. Επίσης, η δημοσίευση των δεδομένων στις κυβερνητικές διαδικτυακές πύλες γίνεται σε ακατέργαστη μορφή με αποτέλεσμα να γίνεται δύσκολη η ανάκτηση και η χρησιμοποίησή τους από τους πολίτες, τους οργανισμούς και τις επιχειρήσεις.

Για τον λόγο αυτό δημιουργήθηκε από την Ευρωπαϊκή Ένωση με τη συνδρομή των κρατών - μελών της το μοντέλο δεδομένων Core Public Service Vocabulary (CPSV) το οποίο είναι ένα απλοποιημένο, επαναχρησιμοποιήσιμο και επεκτάσιμο μοντέλο δεδομένων που απεικονίζει τα θεμελιώδη χαρακτηριστικά μιας υπηρεσίας του δημοσίου φορέα. Η δημιουργία του CPSV έχει στόχο να προσφέρει μια ανεξάρτητη τεχνολογία, με την οποία οι πολίτες ή διάφοροι οργανισμοί θα μπορούν να συνδυάσουν πληροφορίες που αφορούν την ίδια δημόσια υπηρεσία από διαφορετικές σελίδες

ηλεκτρονικής διακυβέρνησης ή διαφορετικά μοντέλα δημοσίων υπηρεσιών, σε εθνικό και σε διεθνές επίπεδο.

Στην Ελλάδα, ο φορέας που χρησιμοποιεί το μοντέλο Core Public Service Vocabulary και συγκεκριμένα το μοντέλο δεδομένων Core Public Service Vocabulary Application Profile, είναι η Περιφέρεια Ηπείρου (http://www.opendataepirus.gr/en/).

1.1 Αντικείμενο διπλωματικής εργασίας

Όπως αναφέρθηκε, η δημοσίευση των δεδομένων στις κυβερνητικές διαδικτυακές πύλες γίνεται σε ακατέργαστη μορφή με αποτέλεσμα να γίνεται δύσκολη η ανάκτηση και η χρησιμοποίησή τους.

Η παρούσα διπλωματική εργασία έχει ως κύριο στόχο την δημιουργία εφαρμογών που θα βοηθήσουν ώστε οι πολίτες, οι οργανισμοί και οι επιχειρήσεις να μπορούν να αναζητούν πληροφορίες που τους ενδιαφέρουν από τα Συνδεδεμένα Ανοικτά Δεδομένα των δημοσίων υπηρεσιών σε μορφή αναγνώσιμη, ώστε και να μπορούν να τις κατανοήσουν και να μπορούν να τις χρησιμοποιήσουν. Συγκεκριμένα, οι εφαρμογές αυτές είναι εφαρμογές που επικοινωνούν με τα αποθετήρια στα οποία βρίσκονται τα Συνδεδεμένα Ανοικτά Δεδομένα των δημοσίων υπηρεσιών και εξάγουν ερωτήματα που ενδιαφέρουν τους πολίτες σε μορφή αναγνώσιμη από τον καθένα.

Οι εφαρμογές που αναπτύχθηκαν είναι δύο:

- η πρώτη είναι η εφαρμογή που επικοινωνεί με το αποθετήριο των δεδομένων που θέλουμε να αναζητήσουμε. Η επικοινωνία εδώ γίνεται μέσω ερωτημάτων. Η εφαρμογή μας θέτει τα ερωτήματα, που είναι ερωτήματα σχετικά με τις υπηρεσίες που προσφέρουν οι δημόσιοι φορείς, στο αποθετήριο και παίρνει τα αποτελέσματα της εκτέλεσης των ερωτημάτων.
- η δεύτερη είναι η εφαρμογή που επικοινωνεί με την παραπάνω και είναι μια εφαρμογή ιστού. Σε αυτή την εφαρμογή, ο χρήστης μπορεί να επιλέξει από τα ερωτήματα αυτό που επιθυμεί (π.χ. απαραίτητα δικαιολογητικά για την διεκπεραίωση μιας δημόσιας υπηρεσίας) και να δει τις πληροφορίες που επιθυμεί.

1.2 Οργάνωση κεφαλαίων

Η παρούσα διπλωματική εργασία οργανώνεται σε πέντε κεφάλαια. Συγκεκριμένα:

Στο πρώτο κεφάλαιο πραγματοποιείται μια εισαγωγή σχετικά με το θέμα που πραγματεύεται η εργασία, δηλαδή την ανάπτυξη δυο εφαρμογών για την ανάδειξη των Συνδεδεμένων Ανοικτών Δεδομένων Δημοσίων Υπηρεσιών.

Στο δεύτερο κεφάλαιο παρουσιάζεται το θεωρητικό υπόβαθρο στο οποίο βασίζεται η παρούσα διπλωματική εργασία. Περιγράφονται οι εννοείς των Διασυνδεδεμένων Δεδομένων, των Ανοικτών Συνδεδεμένων Δεδομένων, των γράφων RDF, της SPARQL, δίνεται η περιγραφή των REST-ful εφαρμογών καθώς επίσης και αναλυτική περιγραφή του μοντέλου Core Public Service Vocabulary.

Στο τρίτο κεφάλαιο περιγράφονται οι τεχνολογίες και οι μέθοδοι που χρησιμοποιήθηκαν για την ανάπτυξη των εφαρμογών. Έτσι, παρουσιάζονται όλα τα προγράμματα και οι τεχνολογίες στις οποίες στηρίχθηκε η εργασία.

Στο τέταρτο κεφάλαιο γίνεται περιγραφή και ανάλυση της παρούσας εργασίας. Συγκεκριμένα παρουσιάζονται και αναλύονται λεπτομερώς τα βήματα που ακολουθήθηκαν, ο κώδικας που αναπτύχθηκε καθώς επίσης και στιγμιότυπα και των δύο εφαρμογών.

Στο πέμπτο κεφάλαιο αναλύονται τα συμπεράσματα που προέκυψαν κατά την ανάπτυξη των εφαρμογών καθώς και ορισμένες πιθανές μελλοντικές επεκτάσεις.

Τέλος, αναγράφονται όλες οι βιβλιογραφικές πήγες που χρησιμοποιήθηκαν κατά την συγγραφή της παρούσας εργασίας και παρατίθεται ο κώδικας των εφαρμογών.

2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

Στο συγκεκριμένο κεφάλαιο παρουσιάζεται το θεωρητικό υπόβαθρο της παρούσας διπλωματικής εργασίας. Στόχος του κεφαλαίου είναι να πραγματοποιήσει μια πλήρη ανασκόπηση των βασικών χαρακτηριστικών των Συνδεδεμένων Ανοικτών Δεδομένων, των εφαρμογών REST καθώς επίσης και μια πλήρη ανασκόπηση των Core Public Service Vocabulary.

2.1 Ηλεκτρονική διακυβέρνηση

Η ηλεκτρονική διακυβέρνηση ορίζεται ως η αξιοποίηση των τεχνολογιών πληροφοριών και επικοινωνιών στις δημόσιες διοικήσεις, σε συνδυασμό με οργανωτικές αλλαγές και νέες δεξιότητες, ώστε να βελτιωθούν η παροχή δημοσίων υπηρεσιών και οι δημοκρατικές διαδικασίες καθώς και να ενισχυθεί η υποστήριξη των πολιτικών που ασκεί το δημόσιο. Η ηλεκτρονική διακυβέρνηση αποτελεί καταλύτη για να διευκολυνθεί η καλύτερη και αποτελεσματικότερη διοίκηση. Βελτιώνει τη διαμόρφωση και την εφαρμογή των πολιτικών που ασκεί το Δημόσιο και βοηθά τον δημόσιο τομέα να αντιμετωπίσει τις αλληλοσυγκρουόμενες απαιτήσεις για την παροχή περισσότερων και καλύτερων υπηρεσιών με λιγότερους πόρους.

Η ηλεκτρονική διακυβέρνηση βοηθά να καταστεί ο δημόσιος τομέας περισσότερο ανοικτός, με λιγότερους αποκλεισμούς και υψηλότερη παραγωγικότητα. Αποτελεί προϋπόθεση ώστε ο δημόσιος τομέας να είναι προετοιμασμένος για το μέλλον.

Η χρηστή διακυβέρνηση υπό αυτή την έννοια -που πρέπει να επιτευχθεί με τον κατάλληλο συνδυασμό τεχνολογιών πληροφοριών και επικοινωνιών, οργανωτικής καινοτομίας και βελτίωσης των δεξιοτήτων ("ηλεκτρονική διακυβέρνηση")- μπορεί επίσης να εφαρμοστεί σε άλλες υπηρεσίες, όπως π.χ. στην υγεία, την εκπαίδευση και τις δημόσιες μμεταφορές. (Επιτροπή των Ευρωπαϊκών Κοινοτήτων, 2008)

2.2 Διασυνδεδεμένα Δεδομένα

Καθημερινά οι οργανισμοί, τα ιδρύματα κ.τ.λ. παράγουν και συλλέγουν έναν μεγάλο όγκο δεδομένων (ιατρικά δεδομένα, δεδομένα που καταγράφουν καθημερινά οι εφαρμογές που χρησιμοποιούν οι χρήστες, ακόμη και δεδομένα από δημόσιες υπηρεσίες), τα οποία διαφέρουν από ίδρυμα σε ίδρυμα κι από οργανισμό σε οργανισμό ανάλογα με την λειτουργία τους. Για μεγάλο χρονικό διάστημα τα δεδομένα είναι διαθέσιμα σε τρίτους που επιθυμούν να τα επεξεργαστούν ή διατίθενται έναντι κάποιου χρηματικού ποσού (Zikopoulos & Eaton, 2011). Τα τελευταία χρόνια ξεκινούν με αργούς ρυθμούς διεθνείς οργανισμοί και δημόσιες κρατικές υπηρεσίες την ελεύθερη διάθεση των δεδομένων αυτών. Γίνονται αρκετές προσπάθειες για την καθιέρωση και ευρεία χρήση των Ανοιχτών Δεδομένων για δημόσια χρήση (W3C, 2019).

Σύμφωνα με τον ορισμό της Ευρωπαϊκής Επιτροπής: «ανοιχτά δεδομένα είναι όσα διατίθενται ελεύθερα προς επαναχρησιμοποίηση, η οποία περιλαμβάνει χρήση των δεδομένων για σκοπούς που προβλέπονται ή όχι από τον πρωτότυπο δημιουργό» (Ευρωπαϊκή Επιτροπή, 2011). Αν και το όλο εγχείρημα είναι ακόμη μακριά από την καθιέρωσή του, ωστόσο τα τελευταία χρόνια παρατηρείται αύξηση των ελευθέρων δεδομένων που διατίθενται καθώς δίνεται μεγαλύτερη προσοχή σε σχέση με το παρελθόν. Βάσει μελετών που έχουν πραγματοποιηθεί παρατηρείται το γεγονός ότι τα δεδομένα που προέρχονται από κυβερνητικές υπηρεσίες και οργανισμούς, σε μεγάλο ποσοστό επαναχρησιμοποιούνται ή διατίθενται στο κοινό για περεταίρω έρευνα και μελέτη.

Ο σημασιολογικός Ιστός και οι δυνατότητες που παρέχει βοήθησαν στο να μπορούν να μοιράζονται τα διάφορα ιδρύματα τα ανοιχτά διασυνδεμένα δεδομένα που παράγουν με την χρήση του διαδικτύου. Πλέον η χρήση του διαδικτύου είναι στην καθημερινότητα όλων, έτσι και η σωστή αξιοποίηση των μέσων από τα ιδρύματα και τους οργανισμούς που τους παρέχει το διαδίκτυο σε επικοινωνιακό και τεχνικό επίπεδο δημιουργεί νέες προοπτικές για την ορθότερη χρήση και εκμετάλλευση των δημοσίων δεδομένων. Για να γίνουν όλα όσα αναφέρθηκαν θα πρέπει εκτός από την διάθεση των δεδομένων να συνδυάζονται και με ανοιχτές πολιτικές για την πρόσβαση σε αυτά. Καταλήγουμε ότι με την ύπαρξη αλλά και τη δυνατότητα πρόσβασης σε «ανοιχτά δεδομένα» (open data) μπορεί να πραγματοποιηθεί η σύνδεση μεταξύ τους.

Τα Ανοιχτά Συνδεδεμένα Δεδομένα (Linked Open Data) που εξετάζονται στο πλαίσιο της παρούσας εργασίας συνιστούν δεδομένα ελεύθερα προσβάσιμα σε όλους, χωρίς τεχνικούς ή νομικούς

περιορισμούς, με άδειες χρήσης και όρους χρήσης που υπόκεινται μόνο στους νόμους περί επαναχρησιμοποίησης πληροφοριών του δημόσιου τομέα. Τα Ανοιχτά Συνδεδεμένα Δεδομένα δεν έχουν περιορισμούς, εφόσον παραμένουν άθικτα και δεν συνιστούν αντικείμενο εκμετάλλευσης, αναφέρεται η πηγή τους και καταγράφεται η ημερομηνία της τελευταίας επικαιροποίησης τους. Έχοντας γίνει κατανοητή η δύναμη και τα πλεονεκτήματα που προσφέρει η διασύνδεση της πληροφορίας, φορείς, κυβερνήσεις αλλά και μεμονωμένα άτομα αρχίζουν να δημοσιεύουν τα δικά τους δεδομένα στο Διαδίκτυο (Berners, 2006).

2.2.1 Βασικές Αρχές

Για να θεωρηθούν τα δεδομένα ανοιχτά θα πρέπει να ακολουθούν κάποιες βασικές αρχές.

Αυτές είναι οι ακόλουθες (Tim & Carl M, 2007):

- Να είναι πλήρη. Θα πρέπει τα δημόσια δεδομένα να μοιράζονται χωρίς να δεσμεύονται ή να περιορίζονται από κάποιον παράγοντα (ως δημόσια δεδομένα ορίζονται όλα τα δεδομένα τα οποία δεν υπόκεινται σε αναγκαίο περιορισμό λόγω προσωπικής φύσης ή διαβάθμισης).
- Να είναι πρωτογενή. Πρωτογενή δεδομένα ορίζονται όσα συλλέγονται κατά την παραγωγή τους και να δεν υπόκεινται σε αλλαγές ή συγχωνεύσεις.
- Θα πρέπει τα δεδομένα να είναι διαθέσιμα κατά την παραγωγή τους ώστε να διατηρούν την χρησιμότητά τους (δηλαδή δεδομένα που παράγονται σήμερα για παράδειγμα από έναν μετεωρολογικό σταθμό, είναι χρήσιμα τη δεδομένη στιγμή και όχι μετά από μια βδομάδα).
- Θα πρέπει τα δεδομένα να είναι διαθέσιμα σε όσο μεγαλύτερο κοινό γίνεται για να αξιοποιούνται στο μέγιστο.
- Θα πρέπει τα δεδομένα να είναι σε κατάλληλη μορφή ώστε να διευκολύνει την επεξεργασία τους από τις υπάρχουσες τεχνολογίες.
- Τα δεδομένα θα πρέπει να είναι διαθέσιμα προς το ευρύ κοινό που εκφράζει το ενδιαφέρον του για να τα χρησιμοποιήσει, χωρίς να υπάρχει καμία διάκριση και χωρίς τη συλλογή προσωπικών δεδομένων. Τέλος, δεν θα πρέπει να εμποδίζεται η επαναχρησιμοποίησή τους. Τα δεδομένα πρέπει να είναι ανοιχτά και να μην έχουν τροποποιηθεί ή παραχθεί με συγκεκριμένη μορφή την οποία μπορεί να επεξεργαστεί αποκλειστικά μια οντότητα ή να

υπόκεινται σε πνευματικά δικαιώματα.

- Τα ανοιχτά δεδομένα δεν θα προστατεύονται από πνευματικά δικαιώματα, πατέντες, λογότυπα κτλ. Μόνο κατά περίπτωση θα τίθενται λογικοί περιορισμοί όπως η προστασία προσωπικών δεδομένων, δηλαδή να μην περιέχονται προσωπικά στοιχεία π.χ. ονόματα ή άλλα προσωπικά χαρακτηριστικά συμπεριλαμβανομένων και γεωγραφικών στιγμάτων (Dulong de Rosnay & Janssen, 2014).
- Θα πρέπει τα δεδομένα να είναι επιθεωρίσημα.

2.2.2 Βαθμολόγηση

Με τα χρόνια μαζεύονται πάρα πολλά ανοιχτά δεδομένα τα οποία με κάποιον τρόπο πρέπει να αξιολογούνται βοηθώντας να μην σπαταλάται χρόνος αναζήτησης μεταξύ πολλών δεδομένων. Ο Tim Berners-Lee έρχεται να αναπτύξει ένα σύστημα βαθμολογίας των ανοιχτών διασυνδεδεμένων δεδομένων. Το σύστημα αυτό το ονόμασε σύστημα πέντε αστεριών και βασίζεται στην αξιολόγηση της ποιότητας των δεδομένων ως προς την ευκολία χρήσης τους, το κόστος, την συνδεσιμότητα του κ.α. (Jayg, 2019). Ως παράδειγμα αναφέρεται η βαθμολογία με ένα αστέρι που μπορεί να δηλώνει δεδομένα χαμηλής προσβασιμότητας και με 5 αστέρια δεδομένα υψηλής προσβασιμότητας και δια-συνδεδεμένα. Όσα περισσότερα αστέρια λάβουν τα ανοιχτά διασυνδεδεμένα δεδομένα, το data set των δεδομένων γίνεται πιο ισχυρό, με αποτέλεσμα να είναι πιο εύκολο για τους «καταναλωτές δεδομένων» να το χρησιμοποιούν (Janowicz, Hitzler, Adams, Kolas & Vardeman II, 2014).

1. ★ Διαθέσιμο στον ιστό (ανεξάρτητα από τη μορφή), αλλά με ανοικτή άδεια (ανοιχτά δεδομένα)
2. ★★ Διατίθεται ως δομημένα δεδομένα, σε μηχαναγνώσιμη μορφή (π.χ. excel αντί για σάρωση εικόνας ενός πίνακα)
3. ★★★ Όπως τα (2), συν μη κατοχυρωμένη μορφή (π.χ. CSV αντί excel)
4. ★★★★ Όλα τα παραπάνω, με επιπλέον τη χρήση ανοικτών προτύπων από το W3C (RDF και SPARQL) για να εντοπιστούν οι πληροφορίες, έτσι ώστε οι χρήστες να μπορούν να τις επισημαίνουν
5. ★★★★★ Όλα τα παραπάνω, συν τη σύνδεση των δεδομένων με δεδομένα άλλων παρόχων

για την παραγωγή περιεχομένου

2.3 Εργαλεία και τεχνικές

Η λειτουργία του Σημασιολογικού Ιστού δεν αφορά μόνο στην εισαγωγή των δεδομένων στο διαδίκτυο, αλλά και στη δημιουργία δεσμών μέσω των οποίων οι μηχανές μπορούν να εξερευνήσουν τον ιστό των δεδομένων. Όπως ο ιστός του υπερκειμένου βασίζεται στην σύνδεση έγγραφων υπερκειμένου γραμμένων σε HTML, ο ιστός δεδομένων βασίζεται στη σύνδεση μεταξύ αυθαίρετων πράξεων που περιγράφονται από το RDF (Bizer, Heath & Berners-Lee, 2009). Τα URI αναγνωρίζουν οποιοδήποτε είδος αντικειμένου ή έννοιας. Είτε για το HTML είτε RDF το URL ακολουθεί τα παρακάτω:

- Χρήση URI ως ονόματα.
- Χρήση URI HTTP έτσι ώστε οι χρήστες να μπορούν να αναζητούν τα ονόματα.
- Όταν κάποιος ψάχνει ένα URI, παρέχει χρήσιμες πληροφορίες χρησιμοποιώντας τα πρότυπα όπως το RDF την SPARQL.
- Ένας σύνδεσμος μπορεί να έχει συνδέσμους προς άλλα URI.

Μπορεί να «φαίνεται απλό», αλλά στην πραγματικότητα, υπάρχει ένας εκπληκτικά μεγάλος όγκος δεδομένων που δεν συνδέεται μεταξύ τους (Shah et al, 2014). Ακολουθούν τεχνικές ανάκτησης της πληροφορίας.

2.3.1 RDF

Ο όρος RDF κάνει την εμφάνισή του ως πρόταση το 1995 από την Κοινοπραξία W3CIncubator του Παγκόσμιου Ιστού με σκοπό να επεκτείνει την ήδη υπάρχουσα πλατφόρμα του, την Platform for Internet Content Selection, που βοηθά στην επιλογή περιεχομένου από το Διαδίκτυο. Η πλατφόρμα PICS αναπτύχθηκε για την περιγραφή περιεχομένων ιστοσελίδων και για την αξιολόγησή τους. Η πλατφόρμα αναλάμβανε την μεταφορά των πληροφοριών, οι οποίες ανταλλάσσονταν μεταξύ ενός web server και ενός web browser και αξιολογούνταν ανάλογα με τις εκάστοτε απαιτήσεις. Αυτή η λειτουργία έφερε στην επιφάνεια την ανάγκη που εμφάνιζε ο παγκόσμιος ιστός για τον περιορισμό

και την αξιολόγηση των μεγάλου πλήθους περιεχομένων του (Klyne & Carroll, 2004).

Μέχρι τότε όλες οι πληροφορίες που ανταλλάσσονταν στον ιστό ήταν κατανοητές μόνο από τους ανθρώπους αλλά όχι για τους υπολογιστές, που η μονή τους δραστηριότητα ήταν να μεταφέρουν και να εμφανίζουν την πληροφορία χωρίς να έχουν γνώση του είδους της. Σε αυτό το σημείο κάνει την εμφάνισή της η ιδέα της δημιουργίας του RDF που θα βοηθούσε να γίνει κατανοητή η ανταλλαγή πληροφορίων τόσο από τους ανθρώπους όσο και από τις μηχανές.

Το Φεβρουάριο του 2004 η Κοινοπραξία του Παγκόσμιου Ιστού ανακοίνωσε την τελική έγκριση και την καθιέρωση του RDF ως πρότυπο. Η καθιέρωση του RDF και η χρήση του σε προϊόντα και υπηρεσίες, μεταβαίνει σε ένα στάδιο όπου θα μπορεί να γίνεται αναζήτηση σε δομημένη πληροφορία στον Παγκόσμιο Ιστό (Klyne & Carroll, 2004).

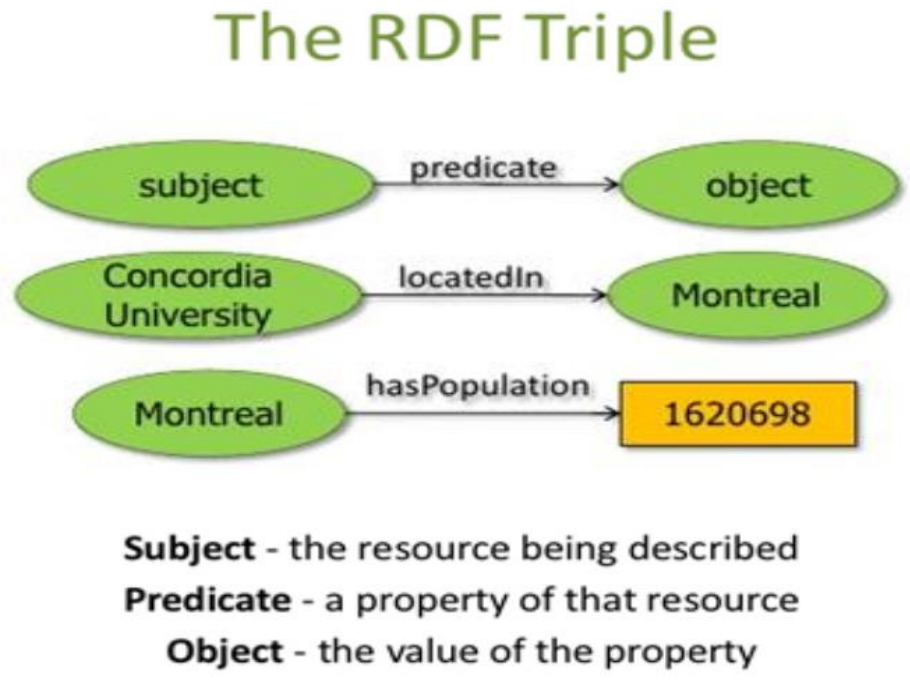
Το πρότυπο RDF βασίζεται στην χρήση τριάδων. Δηλαδή, αναπαριστά τα δεδομένα σε τριάδες γνωστές ως triples. Οι τριάδες ακολουθούν την μορφή «υποκείμενο, κατηγόρημα, αντικείμενο». Αρχικά το αντικείμενο και το υποκείμενο είναι URI’s, τα οποία είναι μια ή πολλές οντότητες και μια τιμή αντίστοιχα. Το κατηγόρημα είναι ένα URL του οποίου η αρμοδιότητα είναι να προσδιορίζει την σχέση ανάμεσα στο αντικείμενο και το υποκείμενο. Δεν είναι αυθαίρετο και βασίζεται σε υπάρχοντα χρησιμοποιούμενα λεξιλόγια (vocabularies). Τα λεξιλόγια περιέχουν πλήθος από URIs, όπως για παράδειγμα το FOAF, το DCAT και το SKOS. Τα λεξικά χρησιμοποιούνται με στόχο να προσδιορίζουν είδη συσχετίσεων σε συγκεκριμένα θεματικά πεδία (Klyne & Carroll, 2004). τα οποία θα αναλυθούν περεταίρω στην συνέχεια για να γίνει κατανοητή η δομή του RDF.

Exemplos

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
<foaf:Person rdf:nodeID="harry">
        <foaf:name>Harry Osborn</foaf:name>
        <rdfs:seeAlso rdf:resource="http://www.osborn.com/harry.rdf"/>
</foaf:Person>
<foaf:Person>
        <foaf:name>Peter Parker</foaf:name>
        <foaf:knows rdf:nodeID="harry"/>
        <foaf:knows>
                <foaf:Person>
                        <foaf:name>Aunt May</foaf:name>
                </foaf:Person>
        </foaf:knows>
</foaf:Person>
</rdf:RDF>
```

http://www.xml.com/pub/a/2004/02/04/foaf.html

Εικόνα 1:Foaf rdf Πηγή: http://www.xml.com



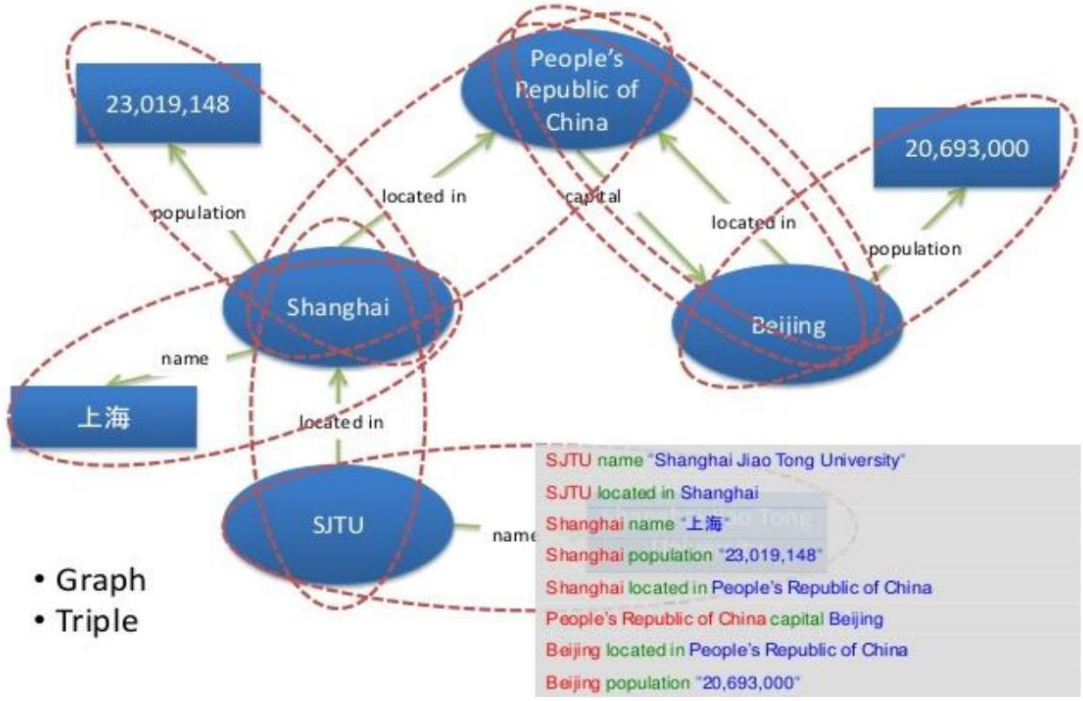
Εικόνα 2:Αναπαράσταση τριπλέτας RDF Πηγή: https://www.slideshare.net/marin_dimitrov/rdf-sparql-and-semantic-repositories

Αναλυτικότερα τα κύρια στοιχεία του RDF μοντέλου είναι:

Resources [πόροι]: Ως resource θεωρείται ένα αντικείμενο στο οποίο αναφερόμαστε, π.χ. βιβλίο, ταινία, τραγούδι κλπ. Για να αναφερθούμε σε ένα resource θα πρέπει να γίνει χρήση ενός URI το οποίο προσδιορίζει μοναδικά το resource.
Properties [ιδιότητες]: Είναι οι ιδιότητες και οι σχέσεις που θα χρησιμοποιηθούν για να περιγράψουν ένα resource [πόρο], π.χ. τίτλος, διάρκεια, καλλιτέχνης κ.τ.λ. και αυτά είναι URLs.
Statements [δηλώσεις]: Τα RDF statements μπορούν να γίνουν με τρεις διαφορετικούς τρόπους:

1. Κάνοντας χρήση τριπλέτων triples [εικόνα 2]
2. Με τη χρήση κατευθυνόμενων γράφων [εικόνα 3]
3. Κάνοντας χρήση σύνταξης που μοιάζει πολύ με την XML.

Μεταφράζοντας τον Γράφο θα παρατηρήσουμε ότι οι κόμβοι αντιστοιχούν σε υποκείμενα και αντικείμενα και οι σύνδεσμοι που έχουν ανάμεσά τους αναφέρονται στα κατηγορήματα. Ο Γράφος που σχηματίζεται αναπαριστά έναν χώρο δεδομένων (global data space) όπου μπορεί να πραγματοποιηθεί αναζήτηση πληροφορίας, μέσω της επίσκεψης των κόμβων και των ακμών του γράφου. Καθώς πληθαίνουν οι τριάδες έχουμε πολλές οντότητές που μπορούν να συνδέονται και μεταξύ τους, δηλαδή μια τριάδα με μια άλλη τριάδα, σχηματίζοντας έτσι έναν κατευθυντικό Γράφο (Manola & Miller, 2014).



Εικόνα 3:Παράδειγμα γράφου Πηγή: https://www.slideshare.net/marin_dimitrov/rdf-sparql-and-semantic-repositories

2.3.2 SPARQL

Η SPARQL είναι ένα σύνολο από βασικές αρχές των ανοιχτών διασυνδεδεμένων δεδομένων. Αποτελεί ένα σύνολο προδιαγραφών που παρέχει γλώσσες και πρωτόκολλα ερωτήσεων για τον χειρισμό αρχείων RDF αντίστοιχη της SQL για τις βάσεις δεδομένων. Οι διαφορές τους οφείλονται κυρίως στην φύση των δεδομένων καθώς δεν υπάρχουν πλέον πίνακες με στήλες, αλλά έχουμε ιδιότητες που αντιστοιχούν σε οντότητες, δηλαδή έναν Γράφο όπου τα δεδομένα συνδέονται μεταξύ τους μέσω των τριάδων. Το πρώτο πρότυπο της SPARQL κάνει την εμφάνισή του τον Ιανουάριο του 2008.

Μέσα σε ένα σύνολο (μοντέλο) δεδομένων RDF (Resource Description Framework), δηλαδή δεδομένα από πηγές του διαδικτύου, η SPARQL καλείται να διαχειριστεί τα δεδομένα αυτού του είδους μέσω ερωτημάτων (queries) και ένα εξάγει συγκεντρωτικά αποτελέσματα π.χ. αναφορές με μεγαλύτερη ευκολία από την κλασσική SQL ή την NnoSQL αν και οι δύο αυτές γλώσσες ερωτημάτων συνδέονται μεταξύ τους. Τα ερωτήματα αυτά μπορούν να είναι: από απλά ταιριάσματα στον Γράφο

που σχηματίζουν τα δεδομένα των αρχείων RDF έως πιο σύνθετες εκφράσεις (Christian & Andreas, 2009).

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name (COUNT(?friend) AS ?count)
WHERE {
    ?person foaf:name ?name .
    ?person foaf:knows ?friend .
} GROUP BY ?person ?name
```

Εικόνα 4:Ερώτημα SPARQL Πηγή: https://repository.kallipos.gr/bitstream/11419/1344/1/06_chapter_5.pdf

Για παράδειγμα, στην Εικόνα 4 βλέπουμε ένα ερώτημα SPARQL το οποίο μοιάζει πάρα πολύ με την SQL, οπότε όσοι είναι εξοικειωμένοι με την SQL ευκολά εξοικειώνονται και με την SPARQL. Το ερώτημα αυτό ανακτά τα ονόματα των ατόμων και το πλήθος των φίλων που περιέχουν τα αρχεία RDF. Οι μεταβλητές εμφανίζονται με ? ή $.

```
# Δηλώσεις προθεμάτων
PREFIX foo: <http://example.com/resources/>
...
# Δηλώσεις συνόλων δεδομένων
FROM ...
# Πρόταση αποτελέσματος
SELECT ...
# Μοτίβο ερωτήματος
WHERE {
    ...
}
# Μετατροπείς ερωτημάτων
ORDER BY ...
```

Εικόνα 5:Βασική δομή ενός ερωτήματος SPARQL Πηγή: https://wordlift.io/blog/en/entity/sparql/

Η SPARQL προσφέρει επιλογές για την μορφή που θα έχουν τα δεδομένα που θα μας επιστρέψει μετά την εκτέλεση του ερωτήματος. Οι επιλογές αυτές είναι:

- Extensible Markup Language – XML

```xml
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
 <head>
   <variable name="name"/>
   <variable name="count"/>
 </head>
 <results>
   <result>
     <binding name="name">
       <literal>Alice</literal>
     </binding>
     <binding name="count">
```

Εικόνα 6:Μορφή δεδομένων σε XML Πηγή: https://repository.kallipos.gr/bitstream/11419/1344/1/06_chapter_5.pdf

- JavaScript Object Notation- JSON

```json
{
  "head": {
    "vars": [ "name" , "count" ]
  } ,
  "results": {
    "bindings": [
      {
        "name": { "type": "literal" , "value": "Alice" } ,
        "count": { "datatype": "http://www.w3.org/2001/XMLSchema#integer"
↪  , "type": "typed-literal" , "value": "3" }
      } ,
      {
        "name": { "type": "literal" , "value": "Bob" } ,
        "count": { "datatype": "http://www.w3.org/2001/XMLSchema#integer"
↪  , "type": "typed-literal" , "value": "1" }
      } ,
      {
        "name": { "type": "literal" , "value": "Charlie" } ,
        "count": { "datatype": "http://www.w3.org/2001/XMLSchema#integer"
↪  , "type": "typed-literal" , "value": "1" }
      }
```

Εικόνα 7:Μορφή δεδομένων σε JSON Πηγή: https://repository.kallipos.gr/bitstream/11419/1344/1/06_chapter_5.pdf

- comma-separated values - CSV και tab-separated values – TSV

CSV

```
name,count
Alice,3
Bob,1
Charlie,1
```

TSV

```
?name<TAB>?count
"Alice"<TAB>3
"Bob"<TAB>1
"Charlie"<TAB>1
```

Εικόνα 8: Μορφή δεδομένων σε CSV, TSV Πηγή: https://repository.kallipos.gr/bitstream/11419/1344/1/06_chapter_5.pdf

2.4 LOD Cloud

Μετά την καθιέρωση των ανοιχτών διασυνδεδεμένων δεδομένων δεν άργησαν αρκετοί οργανισμοί και επιχειρήσεις να ενδιαφερθούν να διαθέσουν ελευθέρα τα δεδομένα τους. Το πιο σημαντικό παράδειγμα της αποδοχής των Συνδεδεμένων Δεδομένων (Linked Data) είναι το Σύννεφο των Ανοιχτών Διασυνδεδεμένων Δεδομένων (Linked Open Data Cloud). Το W3C ξεκινάει τον Ιανουάριο του 2007 ένα ανοιχτό πρόγραμμα το Linked 32 Open Data Project μέσω την ομάδας SWEO.

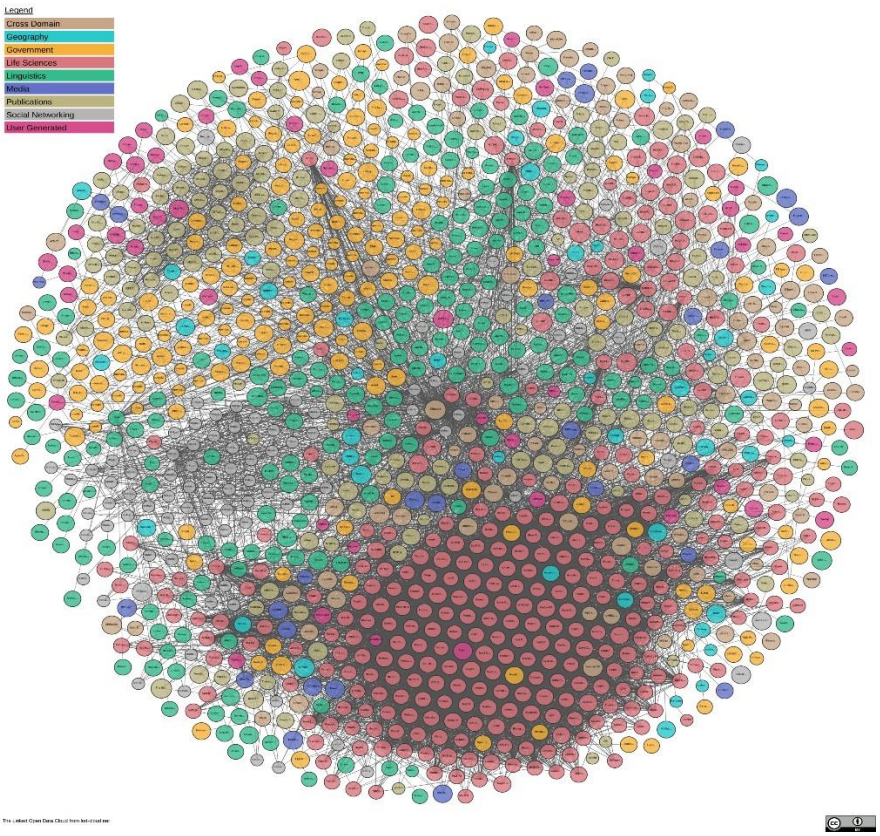
Στόχος ήταν να δημοσιευθούν όσο το δυνατόν περισσότερα συνδεδεμένα δεδομένα τα οποία είναι διαθέσιμα και διαθέτουν ανοικτές άδειες και θα μετατρέπονται σε RDF και θα δημοσιεύονται στο Διαδίκτυο (Bizer, Heath & Berners-Lee, 2009). Τα πρώτα βήματα τα έκαναν ερευνητές και προγραμματιστές από ερευνητικά εργαστήρια πανεπιστήμιων. Όταν όμως άρχισε να γίνεται γνωστό το έργο τους και να γίνονται εμφανή τα οφέλη, ακολούθησαν και άλλοι μεγάλοι οργανισμοί, όπως το BBC, η Thomson Reuters και η μεγαλύτερη Βιβλιοθήκη στην Αμερική, αυτή του Κογκρέσου. Το έργο ήταν ελεύθερο και μπορούσε οποιοσδήποτε να ανεβάσει τα δεδομένα του αρκεί να τηρούσαν τους βασικούς κανόνες για ανοιχτά διασυνδεδεμένα δεδομένα.

Όλα αυτά τα δεδομένα μπορούν να δημιουργήσουν Γράφους. Αυτοί οι Γράφοι έχουν κάποιους δεσμούς μεταξύ των δεδομένων που συνδέονται. Τα πιο έντονα τόξα αντιστοιχούν σε μεγαλύτερο

αριθμό συνδέσεων. Το περιεχόμενο του Σύννεφου μπορεί να περιέχει πάσης φύσεως δεδομένα, όπως (Bizer, Heath & Berners-Lee, 2009):

- **γεωγραφικές περιοχές**
- **ανθρώπους**
- **επιχειρήσεις**
- **βιβλία**
- **επιστημονικές εκδόσεις**
- **ταινίες**
- **μουσική**
- **τηλεοπτικά και ραδιοφωνικά προγράμματα**
- **γονίδια**
- **πρωτεΐνες**
- **φάρμακα και κλινικές δοκιμές**
- **διαδικτυακές κοινότητες**
- **στατιστικά δεδομένα**
- **αποτελέσματα απογραφών και σχόλια**

Στην Εικόνα 9 αποτυπώνεται το διάγραμμα του LOD Cloud. Απεικονίζονται όλα τα σύνολα δεδομένων (datasets). Κάθε χρώμα συμβολίζει και ένα διαφορετικό πεδίο στον Ιστό Δεδομένων. Οι συνδέσεις των δεδομένων στο διάγραμμα απεικονίζονται ως βέλη, η φορά των οποίων καθορίζει από ποιο dataset σε ποιο πάει η ενέργεια.



Εικόνα 9: Μορφή LOD Cloud Πηγή: https://lod-cloud.net/

2.5 Πρακτικές Ανοιχτών διασυνδεδεμένων δεδομένων

Για να είναι ορθά τα διασυνδεδεμένα δεδομένα θα πρέπει να ακολουθούν κάποιους κανόνες, οι οποίοι είναι οι ακόλουθοι (Bizer, Heath & Berners-Lee, 2009):

- Χρήση URIs
- Μετατροπή Δεδομένων σε RDF
- Χρήση οντολογιών
- Πλατφόρμα φιλοξενίας δεδομένων
- Διασύνδεση με άλλα Δεδομένα
- Χρήση της SPARQL

2.6 RESTful API

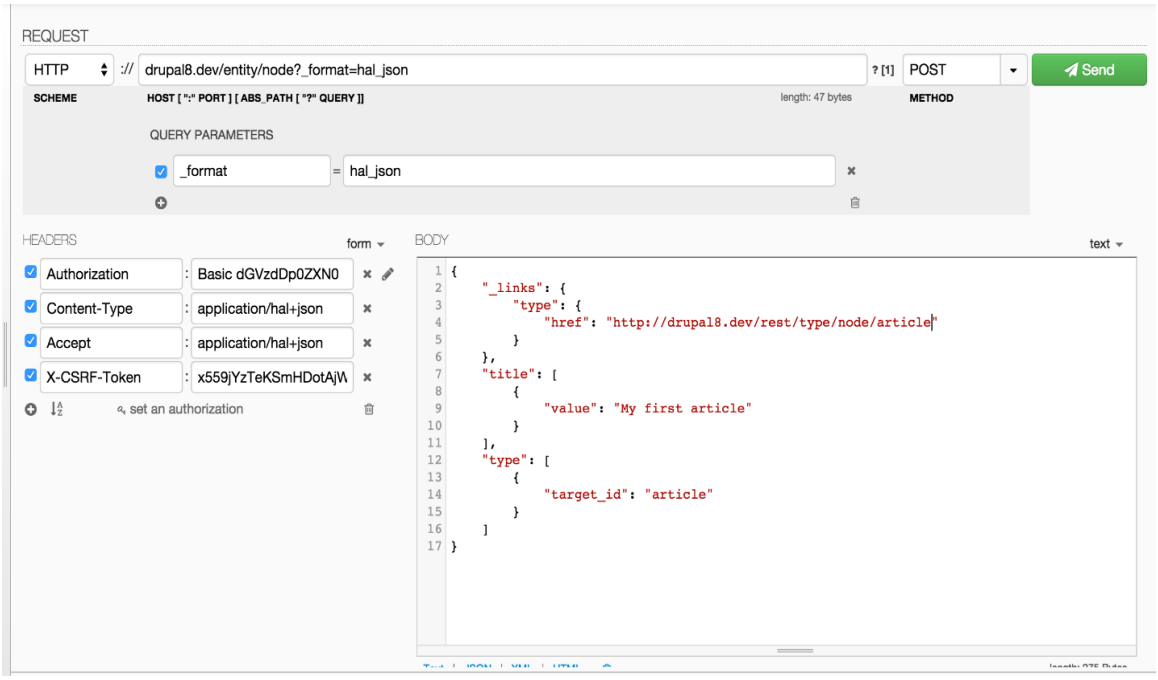
Το πρωτόκολλο REST έκανε την εμφάνισή του το 2000. Δημιουργός του ο Roy Fielding, που αποτελούσε την ακαδημαϊκή του διατριβή η οποία ονομάζεται «Architectural Styles and the Design of Network-based Software Architectures». Το REST προέρχεται από τα ακρωνύμια των λέξεων Representational State Transfer. Αποτελείται από ένα σύνολο αρχών που επικεντρώνονται στο πώς θα σχεδιαστεί μια διαδικτυακή εφαρμογή με βάση τους πόρους που υπάρχουν στην διάθεσή μας, όπως τα δεδομένα ενός συστήματος. Οποιαδήποτε μεταβολή καταγράφεται στο σύστημα, π.χ. η πραγματοποίηση μιας ενέργειας που μεταφέρεται στο σύστημα με την βοήθεια του πρωτοκόλλου HTTP από διάφορους clients. Οι clients δεν περιορίζονται από τη γλώσσα προγραμματισμού στην οποία έχουν υλοποιηθεί (Rouse, 2019).

2.6.1 Τα βασικά χαρακτηριστικά του REST

1. Χρησιμοποιεί HTTP αιτήματα με την βοήθεια των οποίων επικοινωνεί ο χρήστης με την δικτυακή υπηρεσία.

Το REST για να σχεδιαστεί βασίζεται στην σχέση του με τις λειτουργίες CRUD (create, read, update, delete) και των HTTP μεθόδων POST/GET/PUT/DELETE όπου αντιστοιχούν ως εξής:

- Με την POST δημιουργούμε νέους πόρους στον Server(Create)
- Με την GET ανακτούμε δεδομένα(Read).
- Με την PUT κάνουμε ενημέρωση σε ένα υπάρχον πόρο(Update)
- Με την DELETE διαγράφουμε έναν πόρο(Delete) (Rouse, 2019).



Εικόνα 10:Παράδειγμα POST Πηγή: https://www.drupal.org/docs/8/core/modules/rest/3-post-for-creating-content-entities

Για να σχεδιάσουμε μια υπηρεσία REST πρέπει να ακολουθήσουμε δύο βήματα:

1. Θα πρέπει να ξέρουμε τι θα διαθέτει η υπηρεσία μας στον πελάτη-χρήστη.
2. Ποιες από τις μεθόδους GET, POST, PUT και DELETE θα χρειαστούμε και ποιες θα είναι οι ενέργειες που θα κάνει η κάθε μέθοδος στους πόρους μας.

2. Είναι stateless

Με τον όρο stateless αναφερόμαστε στο ότι όποια ενέργεια γίνει δεν μένει σε κάποια μνήμη και εκτελείται μια και μόνο φορά. Αυτό που μεταφράζεται και ως έλλειψη κατάστασης σημαίνει ότι οποιαδήποτε κλήση σε μια υπηρεσία REST δεν θα συνδέεται με κάποια προγενέστερή της. Θα πρέπει να εξασφαλίζεται ότι οι κλήσεις είναι ανεξάρτητες μεταξύ τους. Ο server δεν θα πρέπει να έχει μνήμη και να θυμάται τι έγινε σε κάποια προηγούμενη κλήση τη στιγμή που επεξεργάζεται την τρέχουσα.

3. Απεικονίζει τη δομή των καταλόγων σαν URIs.

Ο σχηματισμός των URI μιας υπηρεσίας REST πρέπει να είναι κατανοητός όταν τον διαβάζει κάποιος και να καταλαβαίνει την ενέργεια στην οποία αποσκοπεί χωρίς να χρειάζεται τεκμηρίωση. Για παράδειγμα, τα URL σε ένα φόρουμ συζητήσεων θα μπορούσαν να έχει την ακόλουθη μορφή:

Εικόνα 11:Κανόνες αποδεκτών URL Πηγή: http://www.myservice.org/discussion/topics/{topic}

4. Μεταφέρει δεδομένα με χρήση XML, JSON ή συνδυασμό τους.

Κατά την δημιουργία της υπηρεσίας μπορούμε να παρέχουμε και τους δύο τρόπους στον πελάτη. Το μόνο που έχει να κάνει είναι να μας το δηλώσει στο HTTP Accept header, όπου η τιμή του θα είναι ένας τύπος MIME. Θα μπορεί να έχει μία από τις τρεις ακόλουθες τιμές, JSON (application/json), XML (application/xml) και XHTML(application/xhtml+xml*)* (Yates, Beal, Keenan, McLaren, Pignatelli & Ritchie, 2015).

2.7 Core Vocabularies

Τα Core Vocabularies είναι απλουστευμένα, επαναχρησιμοποιήσιμα και επεκτάσιμα μοντέλα δεδομένων που καταγράφουν τα θεμελιώδη χαρακτηριστικά μιας οντότητας με τρόπο ουδέτερο ως προς το περιβάλλον. Οι δημόσιες διοικήσεις μπορούν να χρησιμοποιήσουν και να επεκτείνουν τα βασικά λεξιλόγια στα ακόλουθα πλαίσια:

- **Ανάπτυξη νέων συστημάτων:** τα Core Vocabularies μπορούν να χρησιμοποιηθούν ως

προεπιλεγμένο σημείο εκκίνησης για το σχεδιασμό των εννοιολογικών και λογικών μοντέλων δεδομένων στα νεοϊδρυθέντα συστήματα πληροφοριών.

- **Ανταλλαγή πληροφοριών μεταξύ συστημάτων:** τα Core Vocabularies μπορούν να αποτελέσουν τη βάση ενός συγκεκριμένου μοντέλου δεδομένων που χρησιμοποιείται για την ανταλλαγή δεδομένων μεταξύ των υφιστάμενων συστημάτων πληροφοριών.
- **Ενσωμάτωση δεδομένων:** Τα Core Vocabularies μπορούν να χρησιμοποιηθούν για την ενσωμάτωση δεδομένων που προέρχονται από διαφορετικές πηγές δεδομένων και δημιουργούν ένα δίκτυο δεδομένων (PwC EU Services, 2014).

2.8 Το Ευρωπαϊκό μοντέλο Core Public Service Vocabulary

Το Core Public Service Vocabulary (**CPSV)** είναι ένα απλοποιημένο, επαναχρησιμοποιήσιμο και επεκτάσιμο μοντέλο δεδομένων που συλλαμβάνει τα βασικά χαρακτηριστικά μιας υπηρεσίας που προσφέρεται από τη δημόσια διοίκηση (PwC EU Services, 2016).

2.8.1 Χρησιμότητα του Core Public Service Vocabulary

Ακόμη και στην ίδια χώρα, οι δημόσιες υπηρεσίες τεκμηριώνονται βάσει διαφορετικών εθνικών, περιφερειακών ή τοπικών μοντέλων δημόσιας υπηρεσίας. Επιπλέον, οι περιγραφές δημόσιων υπηρεσιών που παρέχονται μέσω δικτυακών πυλών ηλεκτρονικής διακυβέρνησης είναι συνήθως αδόμητες και μη αναγνώσιμες από μηχανές. Αυτή η κατακερματισμένη άποψη της έννοιας της δημόσιας υπηρεσίας και η απουσία μηχανικά αναγνώσιμων περιγραφών της δημόσιας υπηρεσίας επηρεάζουν την ποιότητα και την αποδοτικότητα της παροχής δημόσιων υπηρεσιών, αυξάνουν το διοικητικό φόρτο και καθιστούν την παροχή δημόσιων υπηρεσιών πιο δαπανηρή.

Αυτό αποτελεί σημαντικό εμπόδιο για την ενιαία αγορά.

Επί του παρόντος είναι αδύνατο ή πολύ δύσκολο:

- Η αναζήτηση σε διάφορες πύλες ηλεκτρονικής διακυβέρνησης για δημόσιες υπηρεσίες που σχετίζονται ή ενδέχεται να αντιμετωπίσουν την ίδια ανάγκη.
- Η εύρεση των σωστών πληροφορίων σχετικά με μια συγκεκριμένη δημόσια υπηρεσία,

ειδικά σε ένα διασυνοριακό περιβάλλον με διαφορετικές κυβερνητικές δομές και διαφορετικά μοντέλα δημόσιας υπηρεσίας.

- Η συγκέντρωση πληροφοριών από διάφορα εθνικά, περιφερειακά και τοπικά συστήματα ηλεκτρονικής διακυβέρνησης ή συνδυασμό υφιστάμενων υπηρεσιών για τη δημιουργία νέων.
- Η δημιουργία περιγραφών δημόσιων υπηρεσιών αναγνώσιμες από μηχανήματα, οι οποίες θα είναι επαναχρησιμοποιήσιμες (ακολουθώντας το υπόδειγμα Open Government Linked Data) και θα επέτρεπαν λειτουργίες όπως η αυτοματοποιημένη ανακάλυψη και σύνθεση των υπηρεσιών (PwC EU Services, 2012).

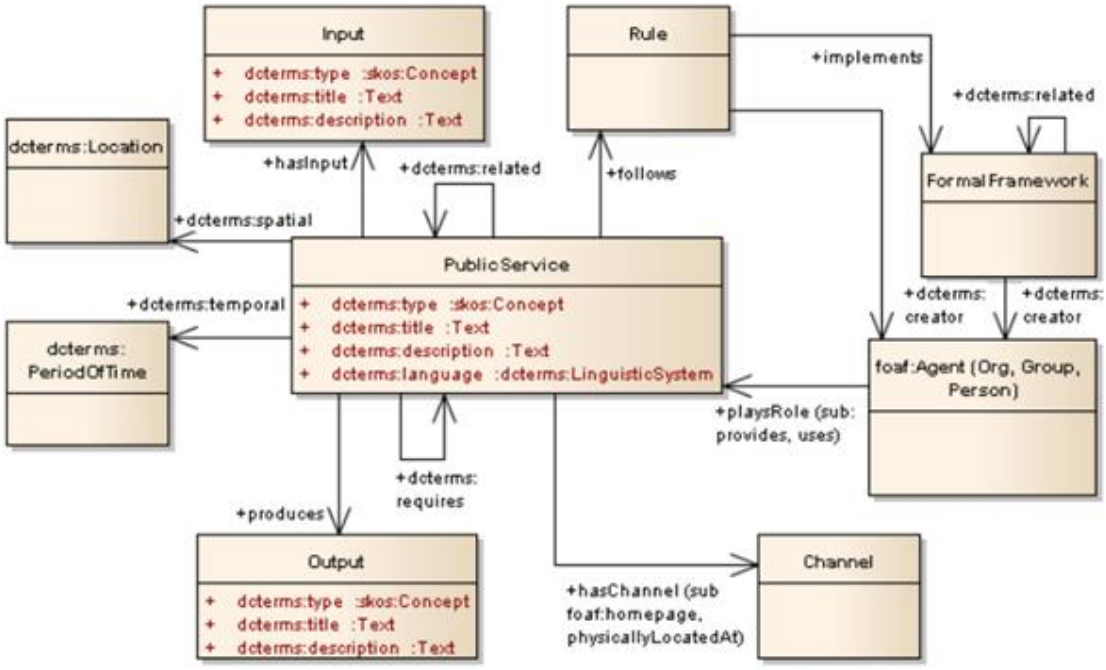
2.8.2 Το όραμα του Core Public Service Vocabulary

Το CPSV αποσκοπεί να προσφέρει μια τεχνολογία ανεξάρτητη και μία γενική εκπροσώπηση μιας υπηρεσίας που παρέχεται από τη δημόσια διοίκηση. Το λεξιλόγιο θα αναδειχθεί ως κοινός παρονομαστής των υφιστάμενων εθνικών, περιφερειακών και τοπικών μοντέλων δημόσιων υπηρεσιών, που θα επιτρέψει την απρόσκοπτη ανταλλαγή υπηρεσιών και πληροφοριών σε διάφορα συστήματα ηλεκτρονικής διακυβέρνησης (PwC EU Services, 2012).

2.8.3 Το μοντέλο

Όπως κάθε βασικό λεξιλόγιο, το CPSV δεν παρέχει και δεν μπορεί να παράσχει όλους τους απαραίτητους όρους για να περιγράψει κάθε δημόσια υπηρεσία σε όλα τα πλαίσια. Αντίθετα, παρέχει ένα θεμέλιο που, όταν χρησιμοποιείται, παρέχει ένα κοινό επίπεδο διαλειτουργικότητας.

Το λεξιλόγιο συνοψίζεται στην Εικόνα 12 και στο ακόλουθο κείμενο:



Εικόνα 12:Διάγραμμα UML για το CPSV Πηγή: https://joinup.ec.europa.eu/sites/default/files/document/2013-05/Piloting%20the%20Core%20Public%20Service%20Vocabulary.pdf

Στην καρδιά του μοντέλου είναι η ίδια η δημόσια υπηρεσία. Αυτή πιθανότατα θα έχει ένα όνομα, μια περιγραφή και, σε πολλές περιπτώσεις, θα είναι συγκεκριμένου τύπου. Για μεγαλύτερη διαλειτουργικότητα, οι τύποι υπηρεσιών θα πρέπει να δίδονται ως τιμές από έναν κατάλογο όπως ο κατάλογος υπηρεσιών που χρησιμοποιείται σε πολλές χώρες της ΕΕ. Η υπηρεσία είναι πιθανό να είναι διαθέσιμη μέσω πολλαπλών καναλιών, συμπεριλαμβανομένης μιας τοποθεσίας Web, μίας ή περισσοτέρων φυσικών τοποθεσιών και ούτω καθεξής. Η γενική ιδιότητα `has:Channel` συνδέει την υπηρεσία με κάθε τέτοιο κανάλι. Το CPSV ισχυρίζεται ότι η ιδιότητα `foaf:homepage` είναι μία υποκατηγορία της `has:Channel` και δημιουργεί την υποκατηγορία `physicallyAvailableAt` η οποία συνδέει την υπηρεσία στο `dcterms:Location`. Η υπηρεσία μπορεί να γίνει διαθέσιμη σε διάφορες γλώσσες οι οποίες μπορούν να καθοριστούν χρησιμοποιώντας την ιδιότητα `dcterms:language`.

Μια υπηρεσία συνήθως απαιτεί κάποιο είδος εισόδου. Σε περίπτωση έκδοσης άδειας οδήγησης, αυτό θα είναι η απόδειξη ότι έχει περάσει τη δοκιμασία οδήγησης. Πολλές υπηρεσίες θα απαιτούν κάποιο είδος απόδειξης ταυτότητας και ούτω καθεξής. Ομοίως, η έξοδος θα ποικίλει ανάλογα με την συγκεκριμένη υπηρεσία, αλλά συνήθως θα υπάρχει ένα έγγραφο ή άλλο αντικείμενο που είναι η έξοδος.

Οι δημόσιες υπηρεσίες ρυθμίζονται από ένα σύνολο κανόνων. Αυτοί θα καθοριστούν συνήθως από έναν ενιαίο οργανισμό και θα εφαρμόσουν σε ένα συνδυασμό νομοθεσίας και πολιτικής. Δεν είναι καθήκον του CPSV να διαμορφώνει λεπτομερείς σχέσεις μεταξύ πολιτικών και νομοθεσίας, ωστόσο τα `dcterms:related` μπορούν να χρησιμοποιηθούν για τη σύνδεση τέτοιων αντικειμένων και αξίζει να σημειωθεί ότι οι κατάλογοι ελεγχόμενων τύπων υπηρεσιών είναι πιθανόν οι ίδιοι να παρέχουν συμβουλές και συνδέσμους με τα σχετικά έγγραφα τα οποία εξουσιοδοτούν ή απαιτούν την παροχή της υπηρεσίας.

Η ιδιότητα `dcterms:Agent` αντιπροσωπεύει οποιοδήποτε άτομο, ομάδα ή οργανισμό που παίζει κάποιο ρόλο στην υπηρεσία.

Οι βασικοί ρόλοι είναι οι `provides` και `users` και συγκεκριμένες ιδιότητες αντικειμένων παρέχονται για αυτές ως συντομεύσεις. Ωστόσο, υπάρχουν διάφοροι ρόλοι που μπορούν να παιχτούν κατά την παροχή ή τη χρήση μιας υπηρεσίας. Ως εκ τούτου, παρέχεται η ιδιότητα `hasRole` (PwC EU Services, 2013).

2.9 Core Public Service Vocabulary Application Profile (CPSV-AP)

Το CPSV-AP είναι ένα μοντέλο δεδομένων που έχει αναπτυχθεί στο πλαίσιο μιας ομάδας εργασίας για την περιγραφή των δημόσιων υπηρεσιών.

Η κύρια εστίαση της έκδοση 1.00 του CPSV-AP ήταν η περιγραφή των δημόσιων υπηρεσιών και των επιχειρηματικών εκδηλώσεων.

Το 2016, το πεδίο εφαρμογής του μοντέλου δεδομένων επεκτάθηκε σταδιακά ώστε να καλύψει κάθε είδους δημόσιας υπηρεσίας. Το έργο αυτό επικεντρώθηκε τελικά στη βελτίωση και την εναρμόνιση της παροχής πληροφοριών σχετικά με τις δημόσιες υπηρεσίες σε καθιερωμένες πύλες ηλεκτρονικής διακυβέρνησης, υιοθετώντας μια οπτική γωνία για τους πολίτες και τις επιχειρήσεις.

Η έκδοση 2.0 αναθεωρήθηκε οδηγώντας στη δημοσίευση του CPSV-AP v2.1. Η ενημέρωση βρίσκει το κίνητρό της στην εμπειρία της εφαρμογής της έκδοσης 2.0 του CPSV-AP με την προσθήκη της

έννοιας του καταλόγου για την κάλυψη των μεταδεδομένων της προέλευσης των πηγών περιγραφών δημοσιών υπηρεσιών που συλλέγονται σε κοινή βάση δεδομένων (αποκεντρωμένη προσέγγιση).

Η τελευταία ενημέρωση είναι το CPSV-AP 2.2.

Το CPSV-AP θεωρήθηκε ως ένα πρώτο βήμα για τη δημιουργία ενός μοντέλου για την περιγραφή δημόσιων υπηρεσιών που σχετίζονται με επιχειρηματικές και συμβάντα ζωής, για να διευκολυνθεί η δημιουργία καταλόγων υπηρεσιών προσανατολισμένων στις επιχειρήσεις και τους πολίτες (PwC EU Services, 2016).

2.9.1 Περιπτώσεις Χρήσης

Το CPSV-AP έχει σχεδιαστεί για να καλύπτει τις περιπτώσεις χρήσης που περιγράφονται παρακάτω. Πρόκειται για τροποποιημένες εκδόσεις των περιπτώσεων χρήσης που προκάλεσαν την ανάπτυξη του αρχικού CPSV-AP, λαμβάνοντας υπόψη τα γεγονότα ζωής των πολιτών καθώς και τις επιχειρηματικές εκδηλώσεις. Αν και το βασικό κίνητρο παραμένει το ίδιο, το πεδίο εφαρμογής είναι ευρύτερο από το αρχικό σύνολο.

- **<u>Περίπτωση χρήσης 1: Βελτίωση της εύρεσης πληροφοριών σχετικά με τις δημόσιες υπηρεσίες</u>**

Σε πολλές χώρες υπάρχουν διαφορετικά τοπικά και περιφερειακά ηλεκτρονικά σημεία ενιαίας εξυπηρέτησης (Points of Single Contact -PSCs). Αυτές οι εθνικές, περιφερειακές ή τοπικές υπηρεσίες μπορεί να έχουν διαφορετικούς τρόπους για τη διάθεση πληροφοριών σχετικά με τις δημόσιες υπηρεσίες και το γεγονός της επιχείρησης ή της ζωής στην οποία ανήκουν.

Ο τρόπος που είναι οργανωμένες οι πληροφορίες ποικίλει ανάλογα με τον δημιουργό τους. Δηλαδή, πληροφορίες των δημοσίων υπηρεσιών είναι συχνά δομημένες σύμφωνα με τις προδιαγραφές του παρόχου ή μια ορισμένη οργανωτική δομή της εκάστοτε δημόσιας διοίκησης. Έτσι δημιουργείται κενό ανάμεσα στην προσφερόμενη πληροφορία και αυτήν που περιμένει να βρει μια επιχείρηση ή ένας πολίτης δομημένα.

Με βάση τα παραπάνω, είναι χρήσιμο να υπάρχει μια ενιαία ψηφιακή πύλη για πληροφορίες

σχετικά με γεγονότα και συναφείς δημόσιες υπηρεσίες, ιδίως στο πλαίσιο της διασυνοριακής παροχής υπηρεσιών. Ένα κοινό μοντέλο δεδομένων όπως το CPSV-AP, καθιστά δυνατή την ευέλικτη ανταλλαγή και ενοποίηση των διαφορετικών περιγραφών δημόσιας υπηρεσίας και διευκολύνει τη δημοσίευση αυτών των πληροφοριών στην ενιαία ψηφιακή πύλη.

- **Περίπτωση χρήσης 2: Δημιουργία ενοποιημένου ευρωπαϊκού καταλόγου δημοσίων υπηρεσιών**

Προϋπόθεση για την ενιαία αγορά της ΕΕ είναι η ελεύθερη κυκλοφορία αγαθών, υπηρεσιών και κεφαλαίων σε ολόκληρη την ΕΕ. Το CPSV-AP έχει σχεδιαστεί για να διευκολύνει τα παραπάνω σε όλα τα επίπεδα. Η χρήση ενός κοινού μοντέλου δεδομένων, όπως το CPSV-AP για την περιγραφή δημόσιων υπηρεσιών, επιτρέπει την ευέλικτη ανταλλαγή και ενσωμάτωση των περιγραφών υπηρεσιών μεταξύ των εθνικών / περιφερειακών αρχών. Με αυτό τον τρόπο, το κοινό μοντέλο δεδομένων λειτουργεί ως γέφυρα, μια κοινή γλώσσα που καθιστά δυνατή τη χαρτογράφηση όλων των διαφορετικών τρόπων περιγραφής των δημόσιων υπηρεσιών και των επιχειρηματικών γεγονότων για την ομαδοποίησή τους σε μία κοινή βάση.

- **Περίπτωση χρήσης 3: Διαχείριση χαρτοφυλακίων δημοσίων υπηρεσιών**

Στις περισσότερες χώρες, η κυριότητα και η διαχείριση των δημόσιων υπηρεσιών κατανέμεται μεταξύ των διαφόρων δημόσιων διοικήσεων και οδηγεί σε διαφορετικούς τρόπους διαχείρισης του κύκλου ζωής τους. Αυτό καθιστά δύσκολη την πλήρη προβολή των δημόσιων υπηρεσιών που προσφέρονται στο πλαίσιο ενός κράτους μέλους και την υιοθέτηση μιας ολιστικής προσέγγισης για τη διαχείρισή τους και τον τρόπο ομαδοποίησης των δημόσιων υπηρεσιών.

Η διαχείριση χαρτοφυλακίου δημόσιων υπηρεσιών επιτρέπει σε μια δημόσια διοίκηση να εφαρμόζει μια ολιστική και συστηματική διαχείριση των επενδύσεών της στην παροχή δημόσιας υπηρεσίας προκειμένου να βελτιστοποιήσει την κάλυψη των αναγκών των πολιτών και των επιχειρήσεων έναντι της συνολικής αξίας των επενδύσεών τους.

Οι πλήρεις, επαναχρησιμοποιούμενες, αναγνώσιμες από μηχανές περιγραφές δημόσιων υπηρεσιών και τα γεγονότα με τα οποία ομαδοποιούνται θα διευκολύνουν τη μέτρηση και την ποσοτικοποίηση του κόστους και των οφελών τους και θα επιτρέψουν τη σύγκρισή τους, την

αξιολόγηση, την παρακολούθηση, τη διαχείριση και τη συνεχή βελτίωση τους (PwC EU Services, 2016).

2.9.2 Υποχρεωτικές και προαιρετικές κλάσεις και ιδιότητες του CPSV-AP

Για να υποδείξουν τις ελάχιστες απαιτήσεις συμμόρφωσης με το CPSV-AP, οι κλάσεις και οι ιδιότητες ταξινομούνται ως υποχρεωτικές ή προαιρετικές. Μια ελάχιστη εφαρμογή του CPSV-AP παρέχει τουλάχιστον πληροφορίες σχετικά με τις υποχρεωτικές ιδιότητες των υποχρεωτικών τάξεων. Οι προαιρετικές τάξεις μπορούν ακόμα να έχουν υποχρεωτικές ιδιότητες για τις οποίες πρέπει να παρέχονται πληροφορίες όταν η συγκεκριμένη κλάση χρησιμοποιείται στην περιγραφή των δημόσιων υπηρεσιών και των επιχειρηματικών εκδηλώσεων.

- ***Υποχρεωτική κλάση (mandatory class):*** ένας δέκτης δεδομένων πρέπει να είναι σε θέση να επεξεργάζεται πληροφορίες σχετικά με τις περιπτώσεις της κλάσης και ένας αποστολέας πρέπει να παρέχει πληροφορίες σχετικά με τις περιπτώσεις της.
- ***Προαιρετική κλάση (optional class):*** ένας δέκτης πρέπει να είναι σε θέση να επεξεργάζεται πληροφορίες σχετικά με τις περιπτώσεις της κλάσης και ένας αποστολέας μπορεί να παράσχει τις πληροφορίες, αλλά δεν είναι υποχρεωμένος να το πράξει.
- ***Υποχρεωτική ιδιότητα (mandatory property)***: ο αποδέκτης των δεδομένων θα πρέπει να μπορεί να επεξεργάζεται πληροφορίες και ο αποστολέας θα πρέπει να παρέχει πληροφορίες για την ιδιότητα της κλάσης. Σε περίπτωση που η ιδιότητα ανήκει σε προαιρετική κλάση, ο αποδέκτης των δεδομένων θα πρέπει να μπορεί να επεξεργάζεται και ο αποστολέας θα πρέπει να παρέχει πληροφορία για την ιδιότητα της κλάσης.
- ***Προαιρετική ιδιότητα (optional property):*** ο αποδέκτης των δεδομένων θα πρέπει να μπορεί να επεξεργάζεται πληροφορία για την ιδιότητα, ενώ ο αποστολέας μπορεί αλλά δεν είναι υποχρεωμένος να παρέχει πληροφορία για αυτήν (PwC EU Services, 2016).

2.9.3 Προθέματα και χώρος ονομάτων

Το μοντέλο CPSV-AP χρησιμοποιεί λεξιλόγια για τις συνδέσεις των δεδομένων των δημοσίων υπηρεσιών. Τα προθέματα (prefixes) και ο χώρος ονομάτων (namespaces) των λεξιλογίων αυτών

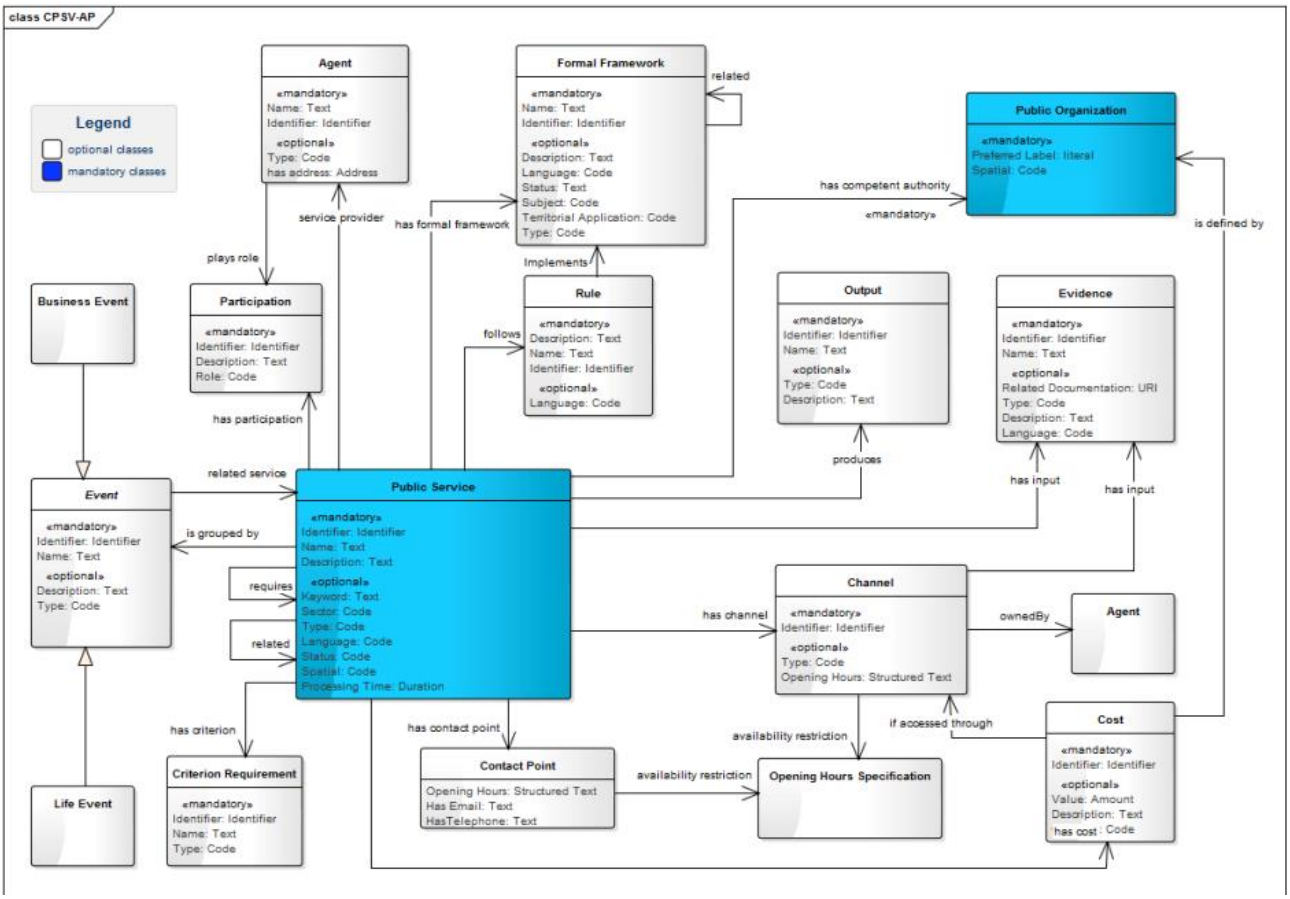
φαίνονται παρακάτω, αντιστοίχως (PwC EU Services, 2016):

Prefix	Namespace
cv	http://data.europa.eu/m8g/
cpsv	http://purl.org/vocab/cpsv#
adms	http://www.w3.org/ns/adms#
eli	http://data.europa.eu/eli/ontology#
dct	http://purl.org/dc/terms/
dcat	http://www.w3.org/ns/dcat#
skos	http://www.w3.org/2004/02/skos/core#
schema	https://schema.org/
locn	http://www.w3.org/ns/locn#
foaf	http://www.w3.org/2004/02/skos/core#

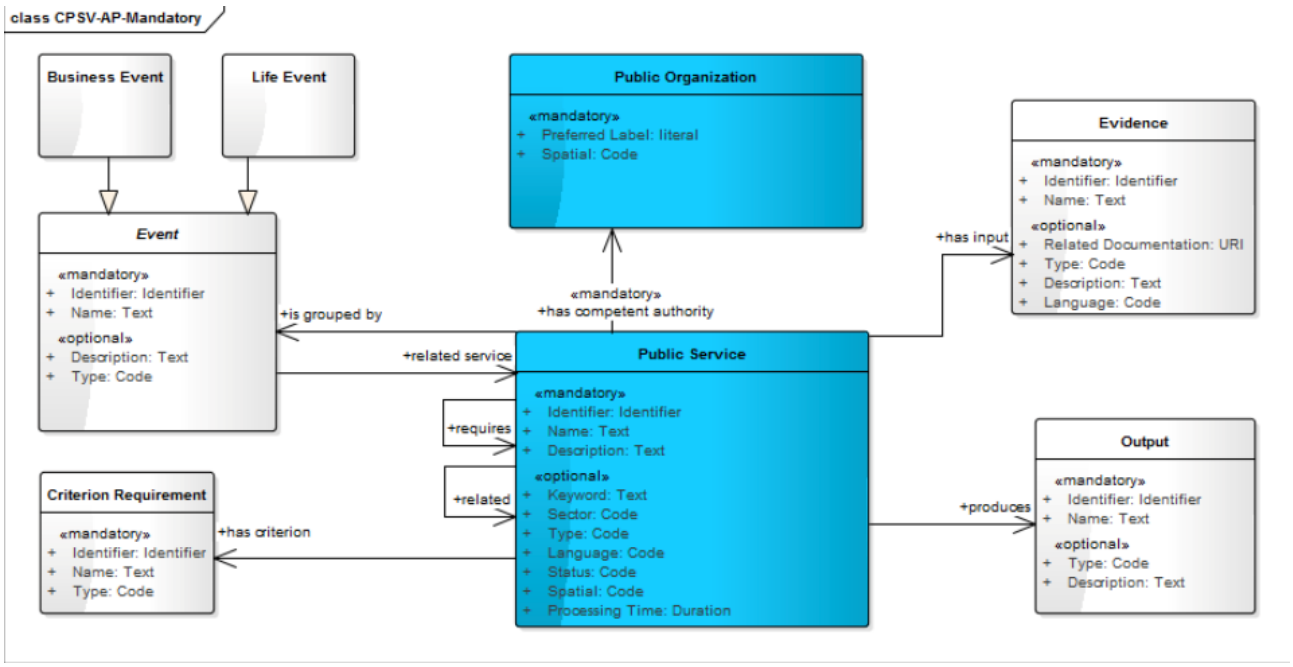
Πίνακας 1:Προθέματα και χώρος ονομάτων

2.9.4 Διαγράμματα UML του CPSV-AP

Η Εικόνα 13 δείχνει το πλήρες προφίλ του CPSV-AP, τις σχέσεις μεταξύ των κλάσεων και τις σχέσεις μεταξύ τον ιδιοτήτων. Η Εικόνα 14 δείχνει τις κλάσεις και τις ιδιότητες που καθορίζουν την ίδια την υπηρεσία: τις απαραίτητες εισροές, τις πιθανές εξόδους, την υπεύθυνη δημόσια αρχή και τα συμβάντα που ενεργοποιούν τη χρήση της υπηρεσίας (PwC EU Services, 2016)..



Εικόνα 13:Γραφική απεικόνιση των σχέσεων μεταξύ των τάξεων και των ιδιοτήτων του CPSV-AP Πηγή: https://joinup.ec.europa.eu/solution/core-public-service-vocabulary-application-profile/distribution/cpsv-ap-specification-v20-pdf



Εικόνα 14: Οι κλάσεις και ιδιότητες στο CPSV-AP που ορίζουν την ίδια την υπηρεσία Πηγή: https://joinup.ec.europa.eu/solution/core-public-service-vocabulary-application-profile/distribution/cpsv-ap-specification-v20-pdf

2.9.5 Public Service Class

Αυτή η κλάση αντιπροσωπεύει την ίδια την Δημόσια Υπηρεσία. Μια δημόσια υπηρεσία είναι ένα υποχρεωτικό ή διακριτικό σύνολο πράξεων που εκτελούνται ή μπορούν να εκτελεστούν από δημόσιο οργανισμό ή για λογαριασμό του. Οι υπηρεσίες μπορεί να είναι προς όφελος ενός ατόμου, μιας επιχείρησης ή άλλης δημόσιας αρχής (PwC EU Services, 2016).

Όνομα κλάσης	Υποχρεωτική/Προαιρετική	URI
Public Service	Υποχρεωτική	cpsv:PublicService

Οι ακόλουθες υποενότητες ορίζουν τις ιδιότητες της κλάσης δημόσιας υπηρεσίας.

- **Identifier:** Αυτή η ιδιότητα αντιπροσωπεύει ένα αναγνωριστικό για τη δημόσια υπηρεσία. Ορίζεται με την ιδιότητα dct:Identifier

Ιδιότητα	URI	Είδος	Πληθικότητα
Identifier	dct:identifier	URI	1...1

- **Name:** Αυτή η ιδιότητα αντιπροσωπεύει το επίσημο όνομα της Δημόσιας Υπηρεσίας. Ορίζεται με την ιδιότητα dct:title.

Ιδιότητα	URI	Είδος	Πληθικότητα
Name	dct:title	Text	1...1

- **Description:** Αυτή η ιδιότητα αντιπροσωπεύει ένα ελεύθερο κείμενο για την περιγραφή της Δημόσιας Υπηρεσίας. Η περιγραφή είναι πιθανό να είναι το κείμενο που βλέπουν οι δυνητικοί χρήστες της Δημόσιας Υπηρεσίας σε οποιονδήποτε κατάλογο δημόσιας υπηρεσίας. Οι δημόσιες διοικήσεις ενθαρρύνονται να συμπεριλάβουν ένα εύλογο επίπεδο λεπτομέρειας στην περιγραφή. Ορίζεται με την ιδιότητα dct:description.

Ιδιότητα	URI	Είδος	Πληθικότητα
Description	dct:description	Text	1...1

- **Keyword:** Αυτή η ιδιότητα αντιπροσωπεύει μια λέξη-κλειδί, έναν όρο ή μια φράση που περιγράφει τη Δημόσια Υπηρεσία. Ορίζεται με την ιδιότητα dcat:keyword

Ιδιότητα	URI	Είδος	Πληθικότητα
Keyword	dcat:keyword	Text	0...n

- **Sector:** Αυτή η ιδιότητα αντιπροσωπεύει τη βιομηχανία ή τον τομέα για τον οποίο η δημόσια υπηρεσία σχετίζεται με ή προορίζεται για. Για παράδειγμα: περιβάλλον, ασφάλεια, στέγαση. Σημειώστε ότι μια ενιαία δημόσια υπηρεσία μπορεί να σχετίζεται με πολλούς τομείς. Ορίζεται με την ιδότητα cv:sector.

Ιδιότητα	URI	Είδος	Πληθικότητα
Sector	cv:sector	Concept	0...n

- **Type:** Αυτή η ιδιότητα αντιπροσωπεύει τον τύπο δημόσιας υπηρεσίας για παράδειγμα

κοινωνική προστασία, υγεία, ψυχαγωγία, πολιτισμό και θρησκεία, τις οικονομικές υποθέσεις, κ.α.. Ορίζεται με την ιδιότητα dct:type.

Ιδιότητα	URI	Είδος	Πληθικότητα
Type	dct:type	Concept	0...n

- **Language:** Αυτή η ιδιότητα αντιπροσωπεύει τη γλώσσα ή τις γλώσσες στις οποίες είναι διαθέσιμη η Δημόσια Υπηρεσία. Αυτό θα μπορούσε να είναι μία γλώσσα ή πολλές γλώσσες, για παράδειγμα σε χώρες με περισσότερες από μία επίσημες γλώσσες. Ορίζεται με την ιδιότητα dct:language.

Ιδιότητα	URI	Είδος	Πληθικότητα
Language	dct:language	Concept	0...n

- **Is Grouped By:** Αυτή η ιδιότητα συνδέει την κλάση Public Service με την κλάση Event. Δηλώνει ότι πολλές Δημόσιες Υπηρεσίες μπορεί να συνδέονται με ένα συγκεκριμένο Γεγονός ή πολλά διαφορετικά. Ορίζεται με την ιδιότητα cv:isGroupedBy.

Ιδιότητα	URI	Είδος	Πληθικότητα
Is Grouped By	cv:isGroupedBy	Event	0...n

- **Has Competent Authority:** Αυτή η ιδιότητα συνδέει μια δημόσια υπηρεσία με έναν δημόσιο οργανισμό, ο οποίος είναι ο υπεύθυνος για την παροχή της δημόσιας υπηρεσίας. Το αν ο συγκεκριμένος δημόσιος οργανισμός παρέχει άμεσα ή εξωτερικά δημόσια υπηρεσία δεν έχει σημασία. Ο Δημόσιος Οργανισμός που είναι η Αρμόδια Αρχή της Υπηρεσίας είναι αυτός που είναι τελικά υπεύθυνος για τη διαχείριση και την παροχή της δημόσιας υπηρεσίας. Ο όρος Αρμόδια Αρχή ορίζεται στην οδηγία 2006/123/EC ως εξής: *"Κάθε φορέας ή αρχή που έχει εποπτικό ή ρυθμιστικό ρόλο σε ένα κράτος μέλος σε σχέση με δραστηριότητες παροχής υπηρεσιών, συμπεριλαμβανομένων, ιδίως, των διοικητικών αρχών, συμπεριλαμβανομένων των δικαστηρίων που ενεργούν ως τέτοια, των επαγγελματικών φορέων και των επαγγελματικών ενώσεων ή άλλων επαγγελματικών οργανώσεων οι οποίες, την άσκηση της νομικής αυτονομίας τους, ρυθμίζουν συλλογικά την πρόσβαση σε δραστηριότητες παροχής*

υπηρεσιών ή την άσκησή τους.”. Ορίζεται με την ιδιότητα cv:hasCompetentAuthority.

Ιδιότητα	URI	Είδος	Πληθικότητα
Has Competent Authority	cv:hasCompetentAuthority	Public Organisation	0...n

- **Has Input:** Αυτή η ιδιότητα συνδέει μια Δημόσια Υπηρεσία με μία ή περισσότερες περιπτώσεις της κλάσης Evidence. Μια συγκεκριμένη δημόσια υπηρεσία μπορεί να απαιτήσει την παρουσία ορισμένων τεκμηρίων για την παράδοση. Ορίζεται με την ιδιότητα cv:hasInput.

Ιδιότητα	URI	Είδος	Πληθικότητα
Has Input	cv:hasInput	Evidence	0...n

- **Has Formal Framework:** Η ιδιότητα αυτή συνδέει μια Δημόσια Υπηρεσία με ένα Τυπικό Πλαίσιο. Δείχνει το τυπικό πλαίσιο (π.χ. νομοθεσία) στο οποίο η δημόσια υπηρεσία σχετίζεται, λειτουργεί ή έχει τη νομική της βάση. Ορίζεται με την ιδιότητα cv:hasFormalFramework.

Ιδιότητα	URI	Είδος	Πληθικότητα
Has Formal Framework	cv:hasFormalFramework	Formal Framework	0...n

- **Produces:** Συνδέει μια δημόσια υπηρεσία με μία ή περισσότερες περιπτώσεις της κλάσης Output που περιγράφει το πραγματικό αποτέλεσμα της εκτέλεσης μιας συγκεκριμένης δημόσιας υπηρεσίας. Οι εξόδοι μπορεί να είναι οποιοσδήποτε πόρος, για παράδειγμα ένα έγγραφο, ή οτιδήποτε άλλο παράγεται ως αποτέλεσμα της εκτέλεσης της Δημόσιας Υπηρεσίας.

Ιδιότητα	URI	Είδος	Πληθικότητα
Produces	cpsv:produces	Output	0...n

- **Spatial:** Μια Δημόσια Υπηρεσία είναι πιθανό να είναι διαθέσιμη σε μία περιορισμένη Χωρική περιοχή, πιθανόν μια περιοχή που ανήκει σε μία συγκεκριμένη δημόσια αρχή. Ο

χωρικός αυτός περιορισμός δεν αφορά την αρμοδιότητα ή την ταχύτητα λειτουργίας της υπηρεσίας. Η τιμή της υποχρεωτικής αυτής ιδιότητας προέρχεται από το Publications Office's Metadata Registry.

Ιδιότητα	URI	Είδος	Πληθικότητα
Spatial	dct:spatial	Concept	0...n

- **Has Channel:** Η ιδιότητα αυτή συνδέει την κλάση Public Service με την κλάση Channel και περιγράφει το μέσο ή το ανάλι με το οποίο κάποιος χρησιμοποιεί ή αλληλεπιδρά με μια Δημόσια Υπηρεσία όπως μέσω μιας online υπηρεσίας, μέσω τηλεφώνου ή με φυσική παρουσία σε γραφείο εξυπηρέτησης.

Ιδιότητα	URI	Είδος	Πληθικότητα
Has Channel	cv:hasChannel	Channel	0...n

- **Has Cost:** Η ιδιότητα αυτή συνδέει την κλάση Public Service με την κλάση Cost. Υποδεικνύει το κόστος που σχετίζεται με την εκτέλεση Δημόσιας Υπηρεσίας για τον πολίτη ή την επιχείρηση που σχετίζεται με την εκτέλεση της συγκεκριμένης Δημόσιας Υπηρεσίας.

Ιδιότητα	URI	Είδος	Πληθικότητα
Has Cost	cv:hasCost	Cost	0...1

2.9.6 Evidence Class

Αυτή η κλάση ορίζεται στο λεξιλόγιο Core Criterion and Core Evidence (CCCEV) ως οποιοσδήποτε πόρος που μπορεί να τεκμηριώσει ή να υποστηρίξει κάποια κριτήρια. Περιέχει πληροφορίες που αποδεικνύουν ότι μία προϋπόθεση κριτηρίου υπάρχει ή είναι αληθές.

Αποδεικτικά στοιχεία μπορεί να είναι οποιοσδήποτε πόρος - έγγραφο, οτιδήποτε χρειάζεται για την εκτέλεση της Δημόσιας Υπηρεσίας. Στο πλαίσιο των δημόσιων υπηρεσιών, τα αποδεικτικά στοιχεία είναι συνήθως διοικητικά έγγραφα ή συμπληρωμένα έντυπα αίτησης. Μια συγκεκριμένη δημόσια υπηρεσία μπορεί να απαιτήσει την παρουσία ορισμένων αποδεικτικών στοιχείων ή συνδυασμών αποδεικτικών στοιχείων προκειμένου να ολοκληρωθεί (PwC EU Services, 2016).

Όνομα κλάσης	Υποχρεωτική/Προαιρετική	URI
Evidence	Προαιρετική	cv:Evidence

Οι ακόλουθες υποενότητες ορίζουν τις ιδιότητες της κλάσης.

- **Identifier:** Η ιδιότητα αυτή αποτελεί αναγνωριστικό της κλάσης Evidence.

Ιδιότητα	URI	Είδος	Πληθικότητα
Identifier	dct:identifier	URI	1...1

- **Name:** Η ιδιότητα αντιπροσωπεύει το επίσημο όνομα της κλάσης.

Ιδιότητα	URI	Είδος	Πληθικότητα
Name	dct:title	Text	1...1

2.9.7 Output Class

Τα αποτελέσματα μπορεί να είναι κάθε πόρος, έγγραφο, οτιδήποτε παράγεται από τη Δημόσια Υπηρεσία (PwC EU Services, 2016).

Όνομα κλάσης	Υποχρεωτική/Προαιρετική	URI
Output	Προαιρετική	cv:Output

Οι ακόλουθες υποενότητες ορίζουν τις ιδιότητες της κλάσης.

- **Identifier:** Η ιδιότητα αυτή αποτελεί αναγνωριστικό της κλάσης Output.

Ιδιότητα	URI	Είδος	Πληθικότητα
Identifier	dct:identifier	URI	1...1

- **Name:** Η ιδιότητα αντιπροσωπεύει το επίσημο όνομα της κλάσης.

Ιδιότητα	URI	Είδος	Πληθικότητα
Name	dct:title	Text	1...1

2.9.8 Cost Class

Η κλάση κόστος αντιπροσωπεύει τα κόστη που σχετίζονται με την εκτέλεση μιας Δημόσιας Υπηρεσίας που ένας χρήστης της πρέπει να πληρώσει προκειμένου να την χρησιμοποιήσει (PwC EU Services, 2016).

Όνομα κλάσης	Υποχρεωτική/Προαιρετική	URI
Cost	Προαιρετική	cv:Cost

Οι ακόλουθες υποενότητες ορίζουν τις ιδιότητες της κλάσης.

- **Identifier:** Η ιδιότητα αυτή αποτελεί αναγνωριστικό της κλάσης Cost.

Ιδιότητα	URI	Είδος	Πληθικότητα
Identifier	dct:identifier	URI	1...1

- **Value:** Η ιδιότητα αυτή αντιπροσωπεύει την αξία που πρέπει να πληρωθεί, δηλαδή το Κόστος.

Ιδιότητα	URI	Είδος	Πληθικότητα
Value	cv:value	Number	0...1

- **Currency:** Αυτή η ιδιότητα αντιπροσωπεύει το νόμισμα στο οποίο πρέπει να καταβληθεί το κόστος.

Ιδιότητα	URI	Είδος	Πληθικότητα
Currency	cv:currency	Concept	0...1

2.9.9 Channel Class

Η κλάση αυτή περιγράφει το κανάλι δια μέσου του οποίου ένας πολίτης χρησιμοποιεί την Δημόσια Υπηρεσία (PwC EU Services, 2016).

Όνομα κλάσης	Υποχρεωτική/Προαιρετική	URI
Channel	Προαιρετική	cv:Channel

Οι ακόλουθες υποενότητες ορίζουν τις ιδιότητες της κλάσης.

- **Identifier:** Η ιδιότητα αυτή αποτελεί αναγνωριστικό της κλάσης Channel.

Ιδιότητα	URI	Είδος	Πληθικότητα
Identifier	dct:identifier	URI	1...1

- **Type:** Η ιδιότητα αυτή περιγράφει τον Τύπο του καναλιού.

Ιδιότητα	URI	Είδος	Πληθικότητα
Type	dct:type	Concept	0...1

- **Opening Hours:** Αυτή η ιδιότητα αντιπροσωπεύει τις κανονικές ώρες λειτουργίας ενός καναλιού.

Ιδιότητα	URI	Είδος	Πληθικότητα
Opening Hours	schema:openingHours	Text	0...n

2.9.10 Formal Framework Class

Αυτή η ιδιότητα αντιπροσωπεύει τη νομοθεσία, την πολιτική ή τις πολιτικές που βρίσκονται πίσω από τους κανόνες που διέπουν την υπηρεσία. Ο ορισμός και οι ιδιότητες ανταποκρίνονται στους

κανόνες που θέτονται στο «Συμπεράσματα του Συμβουλίου για τη θέσπιση του Αναγνωριστικού Ευρωπαϊκής ομοθεσίας (European Legislation Identifier - ELI) (PwC EU Services, 2016).

Όνομα κλάσης	Υποχρεωτική/Προαιρετική	URI
Formal Framework	Προαιρετική	cpsv:FormalFramework

Οι ακόλουθες υποενότητες ορίζουν τις ιδιότητες της κλάσης.

- **Identifier:** Η ιδιότητα αυτή αποτελεί αναγνωριστικό της κλάσης Formal Framework.

Ιδιότητα	URI	Είδος	Πληθικότητα
Identifier	dct:identifier	URI	1...1

- **Name:** Η ιδιότητα αντιπροσωπεύει το επίσημο όνομα της κλάσης.

Ιδιότητα	URI	Είδος	Πληθικότητα
Name	dct:title	Text	1...1

2.9.11 Public Organisation Class

Η υποχρεωτική αυτή κλάση, που ορίζεται στο Core Public Organisation Vocabulary αφορά το Δημόσιο Οργανισμό στον οποίο ανήκει η Δημόσια Υπηρεσία που διεκπεραιώνεται. Βασίζεται σε μεγάλο βαθμό στο W3C[1]. (PwC EU Services, 2016)

Όνομα κλάσης	Υποχρεωτική/Προαιρετική	URI
Public Organisation	Υποχρεωτική	cv:PublicOrganisation

Στο πλαίσιο του CPSV-AP, οι ακόλουθες ιδιότητες είναι υποχρεωτικές:

- preferred label
- spatial

[1] https://www.w3.org/TR/vocab-org/

Ιδιότητα	URI	Είδος	Πληθικότητα
Preferred label	skos:altLabel	Text	1...1
Spatial	dct:spatial	Concepr	1...1

2.9.12 Γνωστές χρήσεις του CPSV-AP

- Το **Βέλγιο** χρησιμοποίησε το CPSV-AP ως κοινό λεξιλόγιο στο πλαίσιο ενός πιλοτικού προγράμματος για την εναρμόνιση των δεδομένων δημόσιων υπηρεσιών από διάφορες περιφερειακές πηγές και τη συγκέντρωσή τους σε ένα κοινό σύστημα που μπορεί να απεικονιστεί σε μια δικτυακή πύλη που βασίζεται σε χρήστες.

- Το **Υπουργείο Οικονομικών της Εσθονίας** δημιούργησε μία επέκταση του CPSV για την κάλυψη των τοπικών αναγκών, καθώς και για την κάλυψη του κύκλου ζωής των δημόσιων υπηρεσιών. Δημιουργήθηκαν νέες κατηγορίες και ιδιότητες για την κάλυψη πληροφοριών που σχετίζονται με την ασφάλεια, την αξιολόγηση και την υποκείμενη υπηρεσία Web που υποστηρίζει την παροχή δημόσιας υπηρεσίας. Το εκτεταμένο CPSV αποτελεί επίσης τη βάση για το εσθονικό πλαίσιο για τη δυναμική διαχείριση των χαρτοφυλακίων δημόσιας υπηρεσίας.

- Η **Φινλανδία** χρησιμοποίησε το CPSV-AP ως έμπνευση για τη δημιουργία ενός εθνικού μοντέλου δεδομένων. Επαναχρησιμοποίησαν κλάσεις και ιδιότητες του CPSV-AP και προσάρμοσαν το μοντέλο στις ανάγκες τους. Επιπλέον, πραγματοποίησαν έναν πιλότο με την Εσθονία, για τον οποίο χρησιμοποίησαν το CPSV-AP και εργαλεία για τη δημιουργία ενός διασυνοριακού καταλόγου δημόσιων υπηρεσιών, καθώς και έναν δικτυακό τόπο με γνώμονα τον χρήστη για την απεικόνιση των δεδομένων.

- **Ο Οργανισμός Ψηφιοποίησης στην Ιταλία** έχει αναπτύξει το CPSV-AP_IT, ένα εθνικό μοντέλο δεδομένων που επεκτείνεται στο CPSV-AP για να συμπεριλάβει τα χαρακτηριστικά για κάθε χώρα. Αυτή τη στιγμή εφαρμόζουν το μοντέλο τους σε έναν εθνικό κατάλογο δημοσίων υπηρεσιών. Για το σκοπό αυτό, συλλέγουν περιγραφές δημόσιων υπηρεσιών από διαφορετικές πηγές σε ολόκληρη τη χώρα.

- Η **Ισπανία** εργάζεται σε έναν πιλότο με την **Πορτογαλία**, για τον οποίο χρησιμοποιεί το CPSV-AP και άλλα εργαλεία για τη δημιουργία ενός διασυνοριακού καταλόγου δημόσιων υπηρεσιών, καθώς και μια ιστοσελίδα με επίκεντρο τον χρήστη για την απεικόνιση των δεδομένων.

- **Η Περιφέρεια Ηπείρου στην Ελλάδα** χρησιμοποίησε το CPSV-AP για να διαμορφώσει ένα υποσύνολο του καταλόγου δημόσιων υπηρεσιών και χρησιμοποίησε την προτεινόμενη διαδικασία για να τις μετατρέψει σε συνδεδεμένα δεδομένα (PwC EU Services, 2016).

3. ΜΕΘΟΔΟΙ ΚΑΙ ΤΕΧΝΟΛΟΓΙΕΣ

Στο συγκεκριμένο κεφάλαιο θα παρουσιαστούν οι μέθοδοι και οι τεχνολογίες που χρησιμοποιήθηκαν στην παρούσα διπλωματική εργασία. Συγκεκριμένα, θα παρουσιαστούν: το εργαλείο Apache Tomcat, το εργαλείο Apache Jena, το πλαίσιο ιστού Spring MVC, η τεχνολογία Java Server Pages και η τεχνολογία Bootstrap.

3.1 Apache Tomcat

Το Apache Tomcat είναι ένα εργαλείο διακομιστή (server) ανοιχτού κώδικα που αναπτύχθηκε από το Apache Software Foundation. Είναι ένα από τα πολλά προιόντα ανοιχτού κώδικα που σχετίζονται με το Apache που χρησιμοποιούνται από επαγγελματίες πληροφορικής για διάφορα καθήκοντα.
Το Apache Tomcat επιτρέπει την υλοποίηση Java Servlets και JavaServer Pages (JSP) για την προώθηση ενός αποτελεσματικού περιβάλλοντος διακομιστή Java. Οι χρήστες μπορούν επίσης να αποκτήσουν πρόσβαση σε πόρους για διαμόρφωση και να χρησιμοποιήσουν επεκτάσιμη γλώσσα σήμανσης (XML) για τη διαμόρφωση των έργων τους (Apache Tomcat, 2019) .

3.2 Apache Jena

Το Apache Jena είναι ένα Java Framework για την κατασκευή εφαρμογών Σημασιολογικού Ιστού. Παρέχει ένα προγραμματιστικό περιβάλλον για RDF, OWL και SPARQL και περιλαμβάνει μία μηχανή συμπερασμάτων βασισμένη σε κανόνες.
Παρέχει μία ελαφριά, κλιμακούμενη αποθήκευση και παρέχει και επίπεδο αναζήτησης SPARQL.
Το SDB είναι ένα υποσύστημα βάσης δεδομένων SPARQL για το Jena. Παρέχει αποθήκευση μεγάλης κλίμακας και ερωτήματα συνόλων δεδομένων RDF χρησιμοποιώντας συμβατικές βάσεις δεδομένων. Παρέχει εργαλεία για εξισορρόπηση φορτίου, ασφάλεια, ομαδοποίηση, δημιουργία αντιγράφων ασφαλείας και διαχείριση της εγκατάστασης (W3C, 2019).

3.3 Spring MVC

Το Spring MVC είναι ένα framework ιστού. Έχει σχεδιαστεί ώστε να παρέχει υψηλή αποδοτικότητα και ευελιξία από την πρώτη μέρα και είναι συμβατό με όλες τις τεχνολογίες Java καθώς επίσης και με πολλά frameworks ανοιχτού κώδικα.

3.3.1 Χαρακτηριστικά

Βασίζεται σε κορυφαίες τεχνολογίες της Java όπως Servlets και JSP και μπορεί να αναπτυχθεί σε οποιονδήποτε Servlet όπως ο Tomcat.

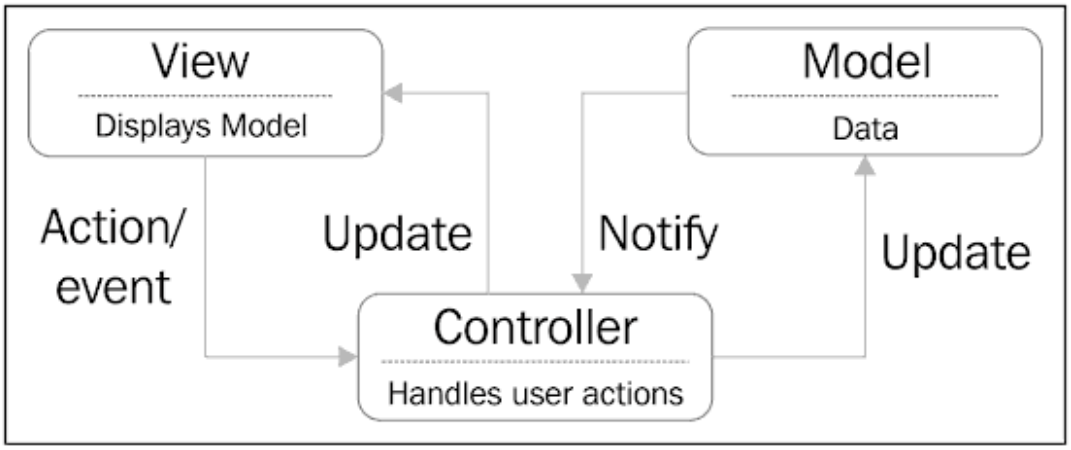
Είναι βασισμένο στο αρχιτεκτονικό πρότυπο Model-View-Controller (MVC), με σαφή διαχωρισμό των ανησυχιών χρησιμοποιώντας απλούς σχολιασμούς και ετικέτες XML ονομάτων.

Υποστηρίζει ένα μεγάλο σύνολο τεχνολογιών όπως: JSP, Thymeleaf, PDF, Excel.

Ευέλικτη χαρτογράφηση URL με αυτόματη μετατροπή αίτησης και απόκρισης σε διάφορες μορφές όπως JSON, XML, HTML (Kunjumohamed, Sattari, Bretet & Warin, 2016).

3.3.2 Model-View-Controller

Το MVC είναι ένα καθιερωμένο αρχιτεκτονικό πρότυπο που χρησιμοποιείται ευρέως για την κατασκευή διαδραστικών εφαρμογών ιστού και επιτραπέζιων υπολογιστών. Διαχωρίζει την εφαρμογή σε τρία κύρια στοιχεία τα οποία στην ουσία αναπαριστούν επίπεδα, διαχωρίζει τις ανησυχίες που μπορεί να υπάρχουν μεταξύ αυτών των τριών και καθορίζει πως θα επικοινωνούν μεταξύ τους.



Εικόνα 15: Το πρότυπο MVC Πηγή: Kunjumohamed, Sattari, Bretet & Warin, 2016

Στην Εικόνα 15 φαίνεται πως λειτουργεί το πρότυπο MVC. Το Model αναπαριστά δεδομένα, το View απεικονίζει το Model και το Controller διαχείριζεται τις ενέργεις χρήσης. Model μπορούν να είναι οποιαδήποτε δεδομένα καθώς επίσης και αυτά που βρίσκονται αποθηκευμένα σε βάσεις δεδομένων. Το Controller συμπεριφέρεται ως ενδιάμεσος μεταξύ των View και Model. Συνήθως έχει ένα σύνολο χειριστών για κάθε γεγονός που δημιουργείται όταν ο χρήστης αλληλεπιδρά μαζί του (Kunjumohamed, Sattari, Bretet & Warin, 2016) .

3.4 JavaServer Pages (JSP)

Οι JavaServerPages (JSP) είναι μία πρότυπη τεχνολογία Java που επιτρέπει τη δημιουργία δυναμικών σελίδων για τις εφαρμογές ιστού της Java. Το JSP είναι γραμμένο πάνω από τις προδιαγραφές του Java Servlet (Java server). Οι δύο τεχνολογίες συνήθως λειτουργούν μαζί. Από τη σκοπιά της κωδικοποίησης, η πιο εμφανής διαφορά μεταξύ τους είναι ότι με τους servers γράφεται πρώτα ο κώδικας Java και στη συνέχεια ενσωματώνεται ο κώδικας της πλευράς του πελάτη (όπως HTML) σε αυτόν τον κώδικα, ενώ με το JSP ξεκινά πρώτα το σενάριο του πελάτη και έπειτα ενσωματώνονται ετικέτες JSP για να γίνει η σύνδεση με Java.

Οι σελίδες JSP είναι σχετικά γρήγορες και εύκολες στην κατασκευή τους και αλληλεπιδρούν άψογα με τους εξυπηρετητές Java σε έναν server όπως ο Tomcat (Tyson, 2019).

3.5 Bootstrap

Το Bootstrap δημιουργήθηκε από τους Mark Otto, Jacob Thornton στο Twitter και κυκλοφόρησε ως ένα προιόν ανοιχτού κώδικα τον Αύγουστο του 2011 στο GitHub.

Είναι ένα ελεύθερο front-end framework για ταχύτερη και ευκολότερη ανάπτυξη ιστού. Περιλαμβάνει πρότυπα σχεδίασης HTML και CSS για τυπογραφία, πλοήγηση, φόρμες, κουμπιά, πίνακες και πολλά άλλα καθώς επίσης και προαιρετικά πρόσθετα της JavaScript. Επίσης, δίνει τη δυνατότητα εύκολης και υπεύθυνης δημιουργίας σχεδιών που ανταποκρίνοται στις ανάγκες του κάθε σχεδιαστή (responsive web design[2]).

3.5.1 Πλεονεκτήματα της χρήσης Bootstrap:

- **Easy to use:** οποιοσδήποτε με βασική γνώση HTML και CSS μπορεί να το χρησιμοποιήσει
- **Responsive features:** το CSS ανταποκρίνεται σε κινητά, tablet, υπολογιστές
- **Mobile-first approach:** τα στυλ για κινητά αποτελούν μέρος του κεντρικού framework
- **Browser compaatibility:** είναι συμβατό με όλα τα προγράμματα περιήγησης (Chrome, Firefox, Internet Explorer, Safari, Opera) (W3C, 2019).

[2] responsive web design: αφορά στη δημιουργία ιστοτόπων οι οποίοι αυτομάτως προσαρμόζονται ώστε να φαίνονται καλά σε όλες τις συσκευές, από κινητά έως μεγάλους σταθερούς υπολογιστές.

4. ΠΕΡΙΓΡΑΦΗ ΚΑΙ ΑΝΑΛΥΣΗ

Στο συγκεκριμένο κεφάλαιο γίνεται η περιγραφή των εφαρμογών που αναπτύχθηκαν στα πλαίσια της παρούσας διπλωματικής εργασίας. Συγκεκριμένα, παρουσιάζονται και αναλύονται λεπτομερώς οι εφαρμογές που δημιουργήθηκαν, τα βήματα που ακολουθήθηκαν καθώς επίσης και τμήματα του κώδικα που δημιουργήθηκε για την ολοκλήρωση της κάθε μιας.

4.1 Δεδομένα

Στα πλαίσια αυτής της διπλωματικής εργασίας, μελετήθηκαν δεδομένα της πύλης Ο Οδηγός του Πολίτη της Περιφέρειας Ηπείρου[3]. Τα δεδομένα αυτά είναι δεδομένα περιγραφών δημοσίων υπηρεσιών που έχουν μετατραπεί σε RDF δεδομένα. Είναι δημοσιευμένα ως Ανοικτά Συνδεδεμένα Δεδομένα στο αποθετήριο γράφων του Πανεπιστήμιου Μακεδονίας (http://data.dai.uom.gr:8890/) και η ανάκτησή τους γίνεται με τη χρήση ερωτημάτων SPARQL.
Το SPARQL Endpoint που χρησιμοποιήθηκε για την δημιουργία των ερωτημάτων είναι: http://195.251.218.39:8890/sparql.

4.1.1 Τα δεδομένα σύμφωνα με τις προδιαγραφές του CPSV-AP

Παρακάτω περιγράφονται κάποιες υποχρεωτικές και προαιρετικές κλάσεις και ιδιότητες του CPSV-AP βάσει των οποίων περιγράφονται τα δεδομένα τα οποία χρησιμοποιήθηκαν ώστε να θέσουμε τα ερωτήματα στο SPARQL Endpoint (http://195.251.218.39:8890/sparql).

4.1.1.1 Public Service:Identifier

Το Identifier (αναγνωριστικό) της κλάσης Public Service μπορεί να είναι της μορφής: http://data.dai.uom.gr:8890/PublicServices/id/ps/ps0001.

[3] http://www.politis.gov.gr/

Το παραπάνω URI αναλύεται ως εξής:

- domain: http://data.dai.uom.gr:8890/PublicServices, όπου
    - http://data.dai.uom.gr:8890 : αποθετήριο γράφων του Πανεπιστήμιου Μακεδονίας
    - PublicServices : δηλώνει ότι είναι αφορά Δημόσιες Υπηρεσίες
- type: id, μοναδικό αναγνωριστικό του πόρου (identifier)
- concept: ps, συντομογραφία του public service
- reference: ps0001, αποτελεί αναφορά σε δημόσια υπηρεσία.

4.1.1.2 Public Service:HasInput, Evidence:Identifier

Το Identifier (αναγνωριστικό) της ιδιοτητας HasInput και της κλασης Evidence μπορεί να είναι της μορφής: http://data.dai.uom.gr:8890/PublicServices/id/doc/identity.

Το παραπάνω URI αναλύεται ως εξής:

- domain: http://data.dai.uom.gr:8890/PublicServices, όπου
    - http://data.dai.uom.gr:8890 : αποθετήριο γράφων του Πανεπιστήμιου Μακεδονίας
    - PublicServices : δηλώνει ότι είναι αφορά Δημόσιες Υπηρεσίες
- type: id, μοναδικό αναγνωριστικό του πόρου (identifier)
- concept: doc, ο τύπος του εγγράφου (ταυτότητα, πιστοποιητικό, άδεια, κτλ.)
- reference: identity, το είδος του αποδεικτικού (Evidence)

4.1.1.3 PublicService:Produces, Output:Identifier

Το Identifier (αναγνωριστικό) της ιδιότητας Produces και της κλάσης Output μπορεί να είναι της μορφής: http://data.dai.uom.gr:8890/PublicServices/id/doc/licence0001.

- domain: http://data.dai.uom.gr:8890/PublicServices, όπου
    - http://data.dai.uom.gr:8890 : αποθετήριο γράφων του Πανεπιστήμιου Μακεδονίας
    - PublicServices : δηλώνει ότι είναι αφορά Δημόσιες Υπηρεσίες
- type: id, μοναδικό αναγνωριστικό του πόρου (identifier)
- concept: doc, ο τύπος του εγγράφου (ταυτότητα, πιστοποιητικό, άδεια, κτλ.)

- reference: license0001, αποτέλεσμα (Output) που παράγει (Produces) η διεκπεραίωση των δημοσίων υπηρεσιών, π.χ. άδεια, βεβαίωση, κτλ.

4.1.1.4 PublicService:HasCost, Cost:Identifier

Το Identifier (αναγνωριστικό) της ιδιότητας HasCost και της κλάσης PublicService μπορεί να είναι της μορφής: http://data.dai.uom.gr:8890/PublicServices/id/cost/cost0001.

- domain: http://data.dai.uom.gr:8890/PublicServices, όπου
  - http://data.dai.uom.gr:8890 : αποθετήριο γράφων του Πανεπιστήμιου Μακεδονίας
  - PublicServices : δηλώνει ότι είναι αφορά Δημόσιες Υπηρεσίες
- type: id, μοναδικό αναγνωριστικό του πόρου (identifier)
- concept: cost, το κόστος για την διεκπεραίωσης της δημόσιας υπηρεσίας
- reference: cost0001, τιμή του κόστους για την διεκπεραίωσης της δημόσιας υπηρεσίας

4.2 Ερωτήματα SPARQL

Για την ανάκτηση των δεδομένων χρησιμοποιήθηκαν ερωτήματα SPARQL. Το SPARQL Endpoint που χρησιμοποιήθηκε για την δημιουργία των ερωτημάτων είναι το http://195.251.218.39:8890/sparql. Στην Εικόνα 16 φαίνεται το περιβάλλον του Endpoint.

Εικόνα 16: Περιβάλλον του SPARQL Endpoint

4.2.1 Αριθμός εγγράφων που απαιτείται για κάθε υπηρεσία

Ένα ερώτημα που μπορούμε να θέσουμε είναι αυτό που επιστρέφει ως αποτέλεσμα τον αριθμό των εγγράφων που απαιτείται για την διεκπεραίωση κάθε δημόσιας υπηρεσίας. Το ερώτημα αυτό σε SPARQL είναι το εξής:

```
prefix dct:<http://purl.org/dc/terms/>
prefix cpsv:<http://purl.org/vocab/cpsv#>

select distinct ?PSname (count(?number) as ?evidence)
where { ?x dct:title ?PSname. ?x cpsv:hasInput ?number. }
```

Το ερώτημα αυτό το εκτελούμε στο Endpoint, όπως φαίνεται στην Εικόνα 17 και βλέπουμε το αποτέλεσμα της εκτέλεσης στην Εικόνα 18.

Εικόνα 17: Πρώτο ερώτημα SPARQL

PSname	evidence
Announcement of the commencement of an electrical engineer professional activities	45
Announcement of the commencement of a mechanical engineer professional activities	36
Announcement of the commencement of a construction machinery assistant operator professional activities	45
Approval of application for the participation in exams for obtaining a construction machinery operator license	54
Matching the old license of a construction machinery operator with a new type of license	18
Initial issuing of driving license of AM, A1, A2, A, B category	63
Upgrading of valid driving license of AM, A1, A2, A, B, BE category	54
Upgrading of valid driving license of B category to C1 or C or D1 or D category, with or without a Certificate of Professional Competence (C.P.C.)	27
Upgrading of valid driving license of C1, C, D1, D categories to C1E, CE, D1E, DE categories, respectively and within these categories, with or without a Certificate of Professional Competence (C.P.C.)	45
Renewing of driving license of every category	135
Issuing of license for the establishment and the operation of a Creative Center for Children (C.C.C.) With Disabilities (W.D) for profit	90
Issuing of license for the establishment and the operation of a non-profit Creative Center for Children (C.C.C.) With Disabilities (W.D.)	90
Issuing of license for the establishment and the operation of a Creative Center for Children (C.C.C.) for profit	90
Issuing of license for the establishment and the operation of a non-profit Creative Center for Children (C.C.C.)	90
Issuing of Certificate of Professional Competence (C.P.C.) of Initial Training	153
Reissuing of driving license of categories restricted for health or retirement reasons	180
Conversion of American, Canadian, Australian, Japanese, South Korean and South African driving license into Greek driving license	189
Issuing of certified copy of driving license due to loss, theft, damage or alteration	153
Exchanging of driving license issued by a Member State of E.E. with a greek driving license of equivalent category	153
Replacement of driving license	153
Limitation of categories of valid driving license for health or retirement reasons	180
Issuing of special driving license of passenger vehicle for public use (taxi and special leasing vehicle)	189
Upgrading of valid special driving license of passenger vehicle for public use (taxi and special leasing vehicle)	18
Issuing of certified copy of special driving license of passenger vehicle for public use (taxi and special leasing vehicle) due to loss, theft or damage	126
Adding of code 96 in valid driving license of B category	45

Εικόνα 18: Αποτέλεσμα εκτέλεσης πρώτου ερωτήματος

4.2.2 Απαιτούμενα δικαιολογητικά για κάθε υπηρεσία

Ένα ερώτημα που μπορούμε να θέσουμε είναι αυτό που επιστρέφει ως αποτέλεσμα τα απαιτούμενα δικαιολογητικά για την διεκπεραίωση κάθε δημόσιας υπηρεσίας. Το ερώτημα αυτό σε SPARQL είναι το εξής:

```
prefix dct:<http://purl.org/dc/terms/>
prefix cpsv:<http://purl.org/vocab/cpsv#>

select distinct ?PSname ?evidence
where {?x dct:title ?PSname. ?x cpsv:hasInput ?evidence. }
```

Το ερώτημα αυτό το εκτελούμε στο Endpoint, όπως φαίνεται στην Εικόνα 19 και βλέπουμε το αποτέλεσμα της εκτέλεσης στην Εικόνα 20.

Εικόνα 19: Δεύτερο ερώτημα SPARQL

PSname	evidence
Announcement of the commencement of an electrical engineer professional activities	http://data.dai.uom.gr:8890/PublicServices/id/doc/Identity
Announcement of the commencement of an electrical engineer professional activities	http://data.dai.uom.gr:8890/PublicServices/id/doc/BachelorDegree
Announcement of the commencement of an electrical engineer professional activities	http://data.dai.uom.gr:8890/PublicServices/id/doc/ProfessionalLicense
Announcement of the commencement of an electrical engineer professional activities	http://data.dai.uom.gr:8890/PublicServices/id/doc/ResidencePermit
Announcement of the commencement of an electrical engineer professional activities	http://data.dai.uom.gr:8890/PublicServices/id/doc/ReceiptPayment
Announcement of the commencement of a mechanical engineer professional activities	http://data.dai.uom.gr:8890/PublicServices/id/doc/Identity
Announcement of the commencement of a mechanical engineer professional activities	http://data.dai.uom.gr:8890/PublicServices/id/doc/BachelorDegree
Announcement of the commencement of a mechanical engineer professional activities	http://data.dai.uom.gr:8890/PublicServices/id/doc/ResidencePermit
Announcement of the commencement of a mechanical engineer professional activities	http://data.dai.uom.gr:8890/PublicServices/id/doc/ReceiptPayment
Announcement of the commencement of a construction machinery assistant operator professional activities	http://data.dai.uom.gr:8890/PublicServices/id/doc/Identity
Announcement of the commencement of a construction machinery assistant operator professional activities	http://data.dai.uom.gr:8890/PublicServices/id/doc/BachelorDegree
Announcement of the commencement of a construction machinery assistant operator professional activities	http://data.dai.uom.gr:8890/PublicServices/id/doc/ProfessionalLicense
Announcement of the commencement of a construction machinery assistant operator professional activities	http://data.dai.uom.gr:8890/PublicServices/id/doc/ResidencePermit
Announcement of the commencement of a construction machinery assistant operator professional activities	http://data.dai.uom.gr:8890/PublicServices/id/doc/ReceiptPayment
Approval of application for the participation in exams for obtaining a construction machinery operator license	http://data.dai.uom.gr:8890/PublicServices/id/doc/BachelorDegree
Approval of application for the participation in exams for obtaining a construction machinery operator license	http://data.dai.uom.gr:8890/PublicServices/id/doc/ReceiptPayment
Approval of application for the participation in exams for obtaining a construction machinery operator license	http://data.dai.uom.gr:8890/PublicServices/id/doc/Certificate0003
Approval of application for the participation in exams for obtaining a construction machinery operator license	http://data.dai.uom.gr:8890/PublicServices/id/doc/DrivingLicenseB
Approval of application for the participation in exams for obtaining a construction machinery operator license	http://data.dai.uom.gr:8890/PublicServices/id/doc/HealthCertificate
Approval of application for the participation in exams for obtaining a construction machinery operator license	http://data.dai.uom.gr:8890/PublicServices/id/doc/Certificate0018
Matching the old license of a construction machinery operator with a new type of license	http://data.dai.uom.gr:8890/PublicServices/id/doc/ReceiptPayment
Matching the old license of a construction machinery operator with a new type of license	http://data.dai.uom.gr:8890/PublicServices/id/doc/ExpiredLicense
Initial issuing of driving license of AM, A1, A2, A, B category	http://data.dai.uom.gr:8890/PublicServices/id/doc/Identity
Initial issuing of driving license of AM, A1, A2, A, B category	http://data.dai.uom.gr:8890/PublicServices/id/doc/DrivingLicenseB
Initial issuing of driving license of AM, A1, A2, A, B category	http://data.dai.uom.gr:8890/PublicServices/id/doc/HealthCertificate
Initial issuing of driving license of AM, A1, A2, A, B category	http://data.dai.uom.gr:8890/PublicServices/id/doc/DrivingLicenseAM

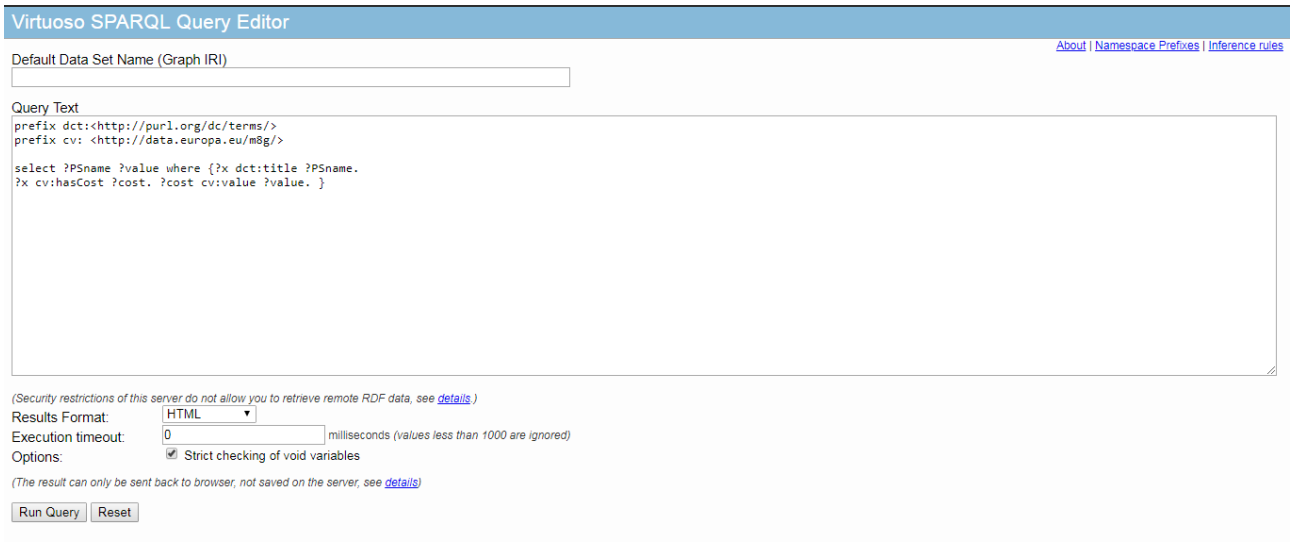
Εικόνα 20: Αποτέλεσμα εκτέλεσης δεύτερου ερωτήματος

4.2.3 Κόστος διεκπεραίωσης κάθε δημόσιας υπηρεσίας

Ένα ερώτημα που μπορούμε να θέσουμε είναι αυτό που επιστρέφει ως αποτέλεσμα το κόστος που απαιτείται για την διεκπεραίωση κάθε δημόσιας υπηρεσίας. Το ερώτημα αυτό σε SPARQL είναι το εξής:

```
prefix dct:<http://purl.org/dc/terms/>
prefix cv: <http://data.europa.eu/m8g/>

select ?PSname ?value where {?x dct:title ?PSname.
?x cv:hasCost ?cost. ?cost cv:value ?value. }
```

Το ερώτημα αυτό το εκτελούμε στο Endpoint, όπως φαίνεται στην Εικόνα 21 και βλέπουμε το αποτέλεσμα της εκτέλεσης στην Εικόνα 22.



Εικόνα 21: Τρίτο ερώτημα SPARQL

PSname	value
Announcement of the commencement of an electrical engineer professional activities	17.0
Announcement of the commencement of an electrical engineer professional activities	17.0
Announcement of the commencement of an electrical engineer professional activities	17.0
Announcement of the commencement of an electrical engineer professional activities	17.0
Announcement of the commencement of an electrical engineer professional activities	17.0
Announcement of the commencement of an electrical engineer professional activities	17.0
Announcement of the commencement of an electrical engineer professional activities	17.0
Announcement of the commencement of an electrical engineer professional activities	17.0
Announcement of the commencement of an electrical engineer professional activities	17.0
Announcement of the commencement of an electrician professional activities	17.0
Announcement of the commencement of an electrician professional activities	17.0
Announcement of the commencement of an electrician professional activities	17.0
Announcement of the commencement of an electrician professional activities	17.0
Announcement of the commencement of an electrician professional activities	17.0
Announcement of the commencement of an electrician professional activities	17.0
Announcement of the commencement of an electrician professional activities	17.0
Announcement of the commencement of an electrician professional activities	17.0
Announcement of the commencement of an electrician professional activities	17.0
Issuing of a copy of a childbirth registration certificate	17.0
Issuing of a copy of a childbirth registration certificate	17.0
Issuing of a copy of a childbirth registration certificate	17.0
Issuing of a copy of a childbirth registration certificate	17.0
Announcement of the commencement of an electrical engineer professional activities	17.0
Announcement of the commencement of an electrical engineer professional activities	17.0
Announcement of the commencement of an electrical engineer professional activities	17.0
Announcement of the commencement of an electrical engineer professional activities	17.0

Εικόνα 22: Αποτέλεσμα εκτέλεσης τρίτου ερωτήματος

4.3 SPARQL Service

Ο σκοπός της παρούσας διπλωματικής ήταν η κατασκευή μιας εφαρμογής ιστού που θα παρουσιάζει τα δεδομένα των δημοσιών υπηρεσιών που υπήρχαν ως Ανοικτά Συνδεδεμένα Δεδομένα στους χρήστες που θα το επιθυμούσαν, δηλαδή στους πολίτες ή ακόμη και στις ίδιες τις δημόσιες υπηρεσίες.

Για να πραγματοποιηθεί το παραπάνω, έπρεπε να δημιουργηθεί αρχικά μια εφαρμογή που θα επικοινωνεί με το Endpoint που βρίσκονται τα δεδομένα μας, θα θέτει τα ερωτήματα που επιθυμούμε και έπειτα θα τα στέλνει στην εφαρμογή Web.

Η εφαρμογή αυτή είναι μια εφαρμογή REST-ful η οποία περιέχει τα παραπάνω δεδομένα. Ουσιαστικά πρόκειται για μια εφαρμογή που επικοινωνεί με το SPARQL Endpoint μέσω μεθόδων, συγκεκριμένα εντολών GET, και δίνει πρόσβαση σε οποίον επιθυμεί να καταναλώσει αυτά τα δεδομένα.

4.3.1 Αρχιτεκτονική συστήματος

Για την κατασκευή του Service χρησιμοποιήθηκαν οι παρακάτω τεχνολογίες:

- Apache Maven, που χρησιμοποιήθηκε ως επέκταση (plug-in) στο Eclipse.
- Spring MVC
- Spring REST
- Apache Jena

Συγκεκριμένα, τα βήματα που έπρεπε να ακολουθηθούν ώστε να γίνει σωστά το SPARQL service είναι τα εξής:

Βήμα 1: Αρχικά, για κάθε ερώτημα SPARQL έπρεπε να γίνει “κλήση” στο SPARQL Endpoint. Το εργαλείο που χρησιμοποιήθηκε γι’ αυτό είναι το Apache Jena.

Βήμα 2: Η κάθε απάντηση που λαμβανόταν από το Endpoint για κάθε ένα από τα ερωτήματα, έπρεπε να μετατραπεί σε REST service χρησιμοποιώντας Spring Rest. Τα αποτελέσματα που

παράγονται από αυτή τη μετατροπή είναι παρόμοια με τα αποτελέσματα που θα παραγόταν σαν JSON.

Η εφαρμογή είναι βασισμένη στο Maven. Η συγγραφή του κώδικα, δηλαδή, έγινε εξ' ολοκλήρου σε αυτό.

Ο web server που χρησιμοποιήθηκε σε όλη τη διάρκεια της ανάπτυξης των εφαρμογών για την δοκιμή του κώδικα είναι ο Apache Tomcat.

4.3.2 Περιγραφή

Παρακάτω, παρουσιάζεται αναλυτικά η διαδικασία που ακολουθήθηκε καθώς επίσης και μέρος του κώδικα που δημιουργήθηκε για την κατασκευή του SPARQL Service.

Για τη διαμόρφωση του SPARQL Service έπρεπε, όπως αναφέρθηκε παραπάνω, να δηλωθεί το Endpoint που βρίσκονται τα δεδομένα στα οποία θα τεθούν τα ερωτήματα. Το Endpoint, όπως φαίνεται και στον παρακάτω κώδικα, είναι το http://195.251.218.39:8890/sparql. Το όνομα της μεταβλητής που το περιέχει είναι SPARQL_END_POINT για να γίνεται κατανοητό σε τι αναφέρεται η μεταβλητή.

```
public interface Configuration {
	String SPARQL_END_POINT = "http://195.251.218.39:8890/sparql";
}
```

Στο παρακάτω τμήμα κώδικα φαίνεται η δήλωση των ερωτημάτων που είχαμε θέσει στην προηγουμένη ενότητα. Τα ονόματα που δοθήκαν στις μεταβλητές είναι δηλωτικά των ερωτημάτων. Επομένως για το ερώτημα που επιστρέφει τα απαραίτητα δικαιολογητικά που απαιτούνται για την διεκπεραίωση των δημοσιών υπηρεσιών το όνομα της μεταβλητής είναι: REQUIRED_DOC_SPARQL_QUERY. Για το ερώτημα που επιστρέφει τον αριθμό των δικαιολογητικών που απαιτούνται για την διεκπεραίωση κάθε δημοσίας υπηρεσίας το όνομα της μεταβλητής είναι: NUMBER_OF_DOC_SPARQL_QUERY. Για το ερώτημα που επιστρέφει το κόστος της διεκπεραίωσης κάθε δημοσίας υπηρεσίας το όνομα της μεταβλητής είναι: VALUE_OF_SERVICE_SPARQL_QUERY.

```
String REQUIRED_DOC_SPARQL_QUERY = "PREFIX  dct:  <http://purl.org/dc/terms/>\n"+
                                   "PREFIX  cpsv: <http://purl.org/vocab/cpsv#>\n"+
                                   "SELECT  distinct ?PSname (COUNT(?number)
AS ?evidence)\n"+
                                   "WHERE\n"+
                                   "{ ?x  dct:title      ?PSname ;\n"+
                                      "cpsv:hasInput  ?number\n"+
                                   "}\n"+
                                   "GROUP BY ?PSname";

String NUMBER_OF_DOC_SPARQL_QUERY = "PREFIX  dct:  <http://purl.org/dc/terms/>\n"+
                                   "PREFIX  cpsv: <http://purl.org/vocab/cpsv#>\n"+
                                     "SELECT DISTINCT  ?PSname ?evidence\n"+
                                     "WHERE\n"+
                                     "{ ?x  dct:title      ?PSname ;\n"+
                                         "cpsv:hasInput  ?evidence\n"+
                                      "}";

String VALUE_OF_SERVICE_SPARQL_QUERY = "PREFIX  dct:
<http://purl.org/dc/terms/>\n"+
                                   "PREFIX  cv:   <http://data.europa.eu/m8g/>\n"+
                                     "SELECT  ?PSname ?value\n"+
                                     "WHERE\n"+
                                     "{ ?x     dct:title   ?PSname ;\n"+
                                               "cv:hasCost  ?cost .\n"+
                                               "?cost  cv:value    ?value\n"+
                                     "}";}
```

Για να δημιουργηθεί ο σκελετός της εφαρμογής μας σε Spring Rest έπρεπε να δημιουργηθούν τα παρακάτω αντικείμενα (beans). Ένα για κάθε ένα από τα ερωτήματα που αναφέρθηκαν προηγουμένως. Το πρώτο αντικείμενο είναι για τα απαραίτητα δικαιολογητικά που απαιτούνται για την διεκπεραίωση των δημοσιών υπηρεσιών, το δεύτερο είναι για τον αριθμό των δικαιολογητικών που απαιτούνται για την διεκπεραίωση κάθε δημοσίας υπηρεσίας και το τρίτο είναι για το κόστος της διεκπεραίωσης κάθε δημοσίας υπηρεσίας.

Αντικείμενο για τα απαραίτητα δικαιολογητικά που απαιτούνται για την διεκπεραίωση των δημοσιών υπηρεσιών:

```
@JsonPropertyOrder({"psname","evidence"})
public class NumberOfDocQueryResultBean {

      private String PSname;
      private String evidence;


      public String getPSname() {
            return PSname;
      }
```

```
	public void setPSname(String pSname) {
		PSname = pSname;
	}
	public String getEvidence() {
		return evidence;
	}
	public void setEvidence(String evidence) {
		this.evidence = evidence;
	}
}
```

Αντικείμενο για τον αριθμό των δικαιολογητικών που απαιτούνται για την διεκπεραίωση κάθε δημοσίας υπηρεσίας:

```
@JsonPropertyOrder({"psname","evidence"})
public class RequiredDocQueryResultBean {

	private String PSname;
	private String evidence;


	public String getPSname() {
		return PSname;
	}
	public void setPSname(String pSname) {
		PSname = pSname;
	}
	public String getEvidence() {
		return evidence;
	}
	public void setEvidence(String evidence) {
		this.evidence = evidence;
	}
}
```

Αντικείμενο για το κόστος της διεκπεραίωσης κάθε δημοσίας υπηρεσίας:

```
@JsonPropertyOrder({"psname","value"})
public class ValueOfServiceQueryResultBean {

	private String PSname;
	public String getPSname() {
		return PSname;
	}
	public void setPSname(String pSname) {
		PSname = pSname;
	}
	public String getValue() {
		return value;
	}
	public void setValue(String value) {
		this.value = value;
```

```
	}
	private String value;
}
```

Το @JsonPropertyOrder είναι ένα σχόλιο (annotation) που χρησιμοποιείται για να καθορίσει την σειρά με την οποία θα εμφανιστούν τα δεδομένα μετά την εκτέλεση των αντικειμένων. Οι ιδιότητες που βρίσκονται μέσα στο σχόλιο, είναι αυτές που θα εμφανιστούν πρώτες σε σειρά. Στο πρώτο αντικείμενο, θα εμφανιστούν πρώτες οι ιδιότητες: psname (όνομα της δημόσιας υπηρεσίας) και evidence (αποδεικτικά). Στο δεύτερο αντικείμενο θα εμφανιστούν πρώτες οι ιδιότητες: psname και evidence. Στο τρίτο αντικείμενο θα εμφανιστούν πρώτες οι ιδιότητες: psname και value (κόστος).

Με το Spring, για να χτιστεί μια εφαρμογή REST-ful, θα πρέπει οι αιτήσεις HTTP να χειρίζονται από έναν ελεγκτή (controller). Ο controller επιστρέφει τα δεδομένα στο σώμα απόκρισης του HTTP. Στο παρακάτω τμήμα κώδικα φαίνεται ο controller που χρησιμοποιήθηκε για τα παραπάνω ερωτήματα.

```
@RequestMapping("/rest/api")
@RestController
public class SparqlServiceController {


	@RequestMapping(method = RequestMethod.GET, value = "/required-doc")
	@ResponseBody()
	public ResponseEntity<List<Object>> getSparqlQuery1(){
		List<Object> result = null;
		try{
			result = sparqlEngine.executeRequiredDocSparqlQuery();
			return new ResponseEntity<List<Object>>(result, HttpStatus.OK);
		}catch(Exception e){
			System.out.println("exception occured : "+e);
			//ErrorBean eb= new ErrorBean();
			eb.setErrorCode("-1");
			result = new ArrayList<Object>();
			result.add((Object)eb);
			eb.setErrorMessage("Internal Server Error Occurred");
			return new ResponseEntity<List<Object>>(result, HttpStatus.OK);
		}
	}

	@RequestMapping(method = RequestMethod.GET, value = "/number-of-doc")
	@ResponseBody()
	public ResponseEntity<List<Object>> getSparqlQuery2(){
		List<Object> result = null;
		try{
			result = sparqlEngine.executeNumberOfDocSparqlQuery();
			return new ResponseEntity<List<Object>>(result, HttpStatus.OK);
		}catch(Exception e){
			System.out.println("exception occured : "+e);
			eb.setErrorCode("-1");
			result = new ArrayList<Object>();
			result.add((Object)eb);
			eb.setErrorMessage("Internal Server Error Occurred");
```

```
                return new ResponseEntity<List<Object>>(result, HttpStatus.OK);
            }
        }

        @RequestMapping(method = RequestMethod.GET, value = "/value-of-service")
        @ResponseBody()
        public ResponseEntity<List<Object>> getSparqlQuery3(){
            List<Object> result = null;
            try{
                result = sparqlEngine.executeValueOfServiceSparqlQuery();
                return new ResponseEntity<List<Object>>(result, HttpStatus.OK);
            }catch(Exception e){
                System.out.println("exception occured : "+e);
                eb.setErrorCode("-1");
                result = new ArrayList<Object>();
                result.add((Object)eb);
                eb.setErrorMessage("Internal Server Error Occurred");
                return new ResponseEntity<List<Object>>(result, HttpStatus.OK);
            }
        }
}
```

Οι παράμετροι της μεθόδου HTTP δεν είναι προκαθορισμένοι και αν δεν καθοριστεί μια τιμή θα αντιστοιχιστούν σε οποιοδήποτε αίτημα HTTP. Αυτό γίνεται με το σχόλιο @RequestMapping. Για κάθε ένα ερώτημα θέτουμε την μέθοδο σε GET (@RequestMapping(method = RequestMethod.GET, ..).

Το σχόλιο @ResponseBody χρησιμοποιείται για να επιστραφούν τα δεδομένα απευθείας στο σώμα απόκρισης του HTTP.

4.3.3 Εκτέλεση

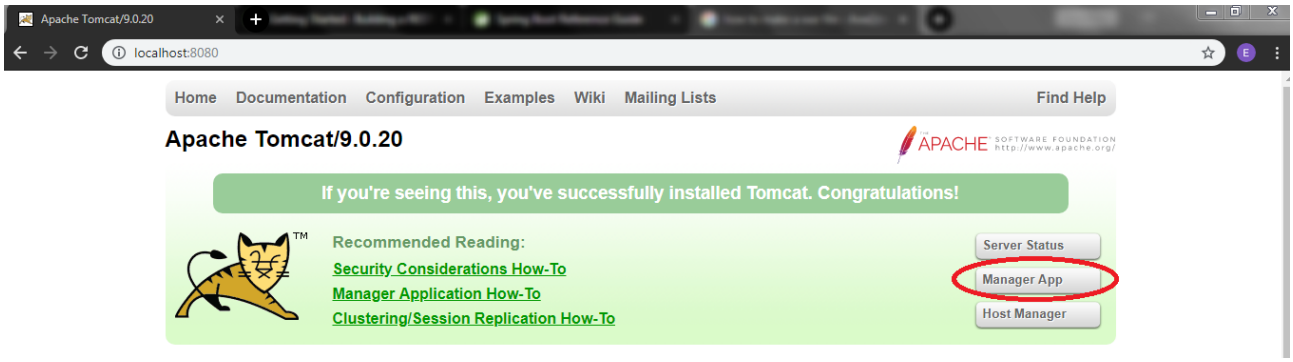
Αφού δημιουργήθηκε ο κώδικας για την ανάπτυξη της εφαρμογής, έπρεπε να γίνει εκτελέσιμος. Παρακάτω παρουσιάζονται τα αποτελέσματα της εκτέλεσης.

Αρχικά, δημιουργήθηκε το αρχείο war της εφαρμογής ώστε να τοποθετηθεί στον Tomcat Server.

Το αρχείο war δημιουργήθηκε με την εντολή:

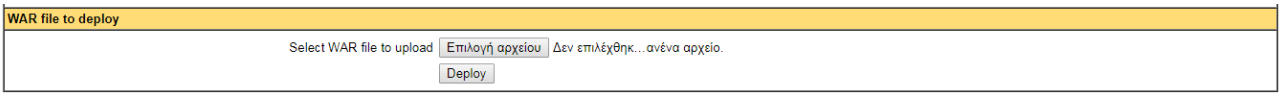
```
jar -cvf sparqlservice.war *
```

➔ Στον Tomcat Server, γίνεται σύνδεση στο Manager App, όπως φαίνεται στην Εικόνα 23.



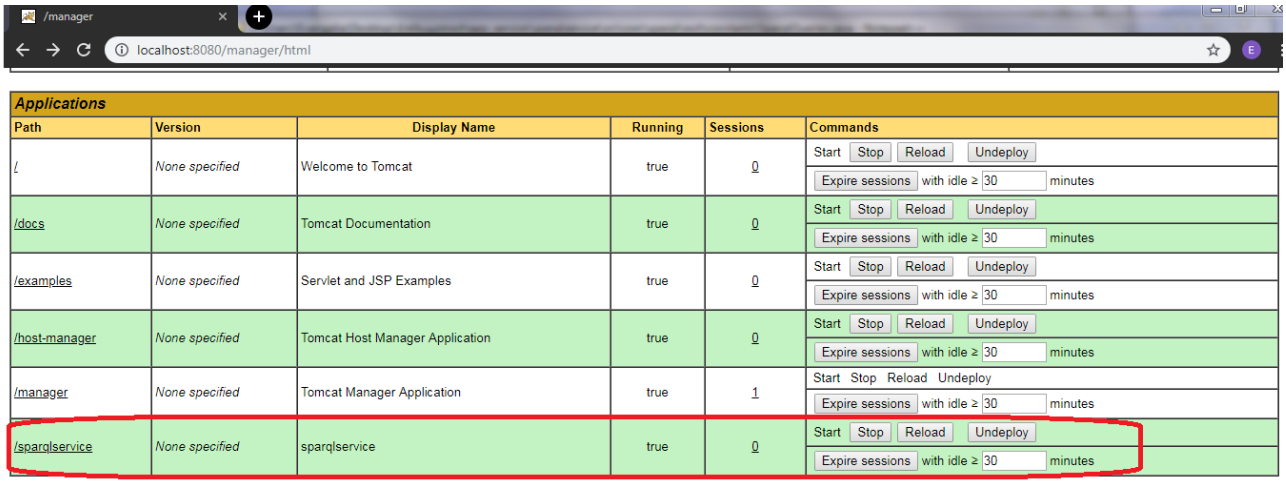
Εικόνα 23: Σύνδεση στον manager του Tomcat

➔ Στο Web Application του Manager, γίνεται η εισαγωγή του αρχείου war που δημιουργήθηκε πριν όπως φαίνεται στην Εικόνα 24.



Εικόνα 24: Εισαγωγή war file

➔ Αφού γίνει η εισαγωγή του αρχείου, θα εμφανιστεί στον manager του Tomcat όπως φαίνεται στην Εικόνα 25.

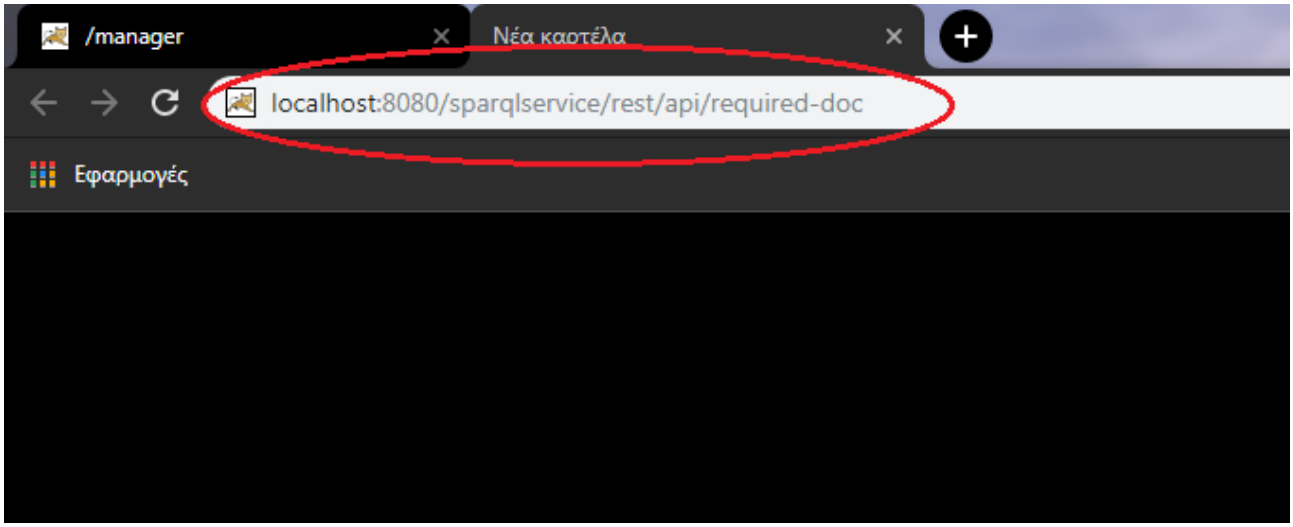


Εικόνα 25: Η εφαρμογή στον manager του Tomcat

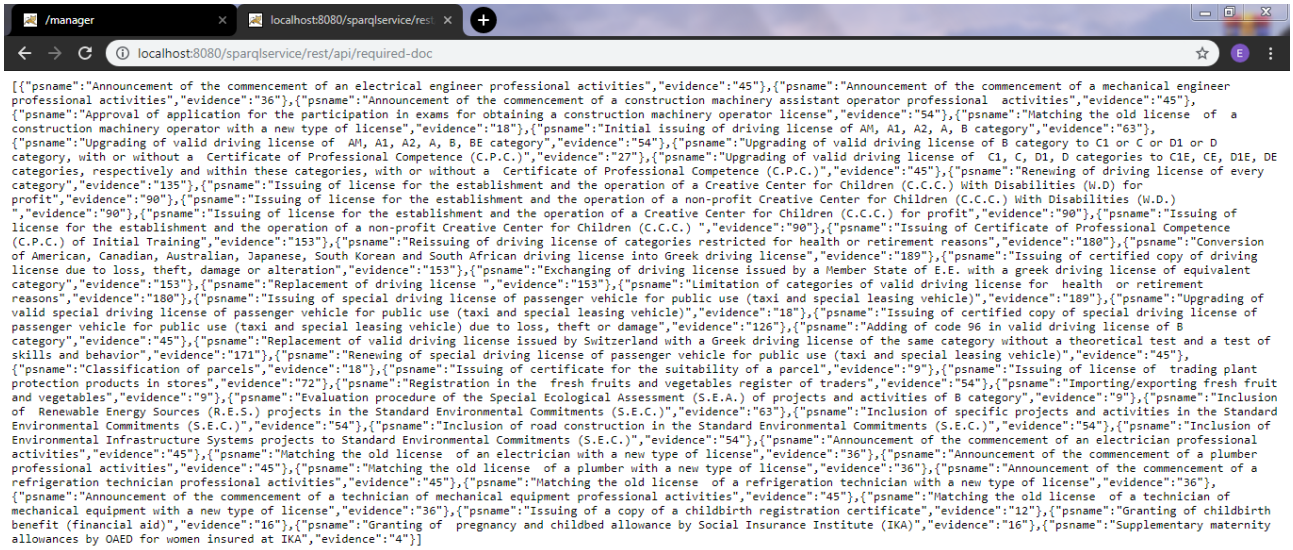
Για να ελέγξουμε την εφαρμογή θα πρέπει να το κάνουμε για κάθε ερώτημα ξεχωριστά χρησιμοποιώντας το URI που βρίσκεται η εφαρμογή μας, δηλαδή το:

localhost:8080/sparqlservice/rest/api/ και το όνομα για το κάθε ερώτημα.

Για το πρώτο ερώτημα χρησιμοποιούμε το localhost:8080/sparqlservice/rest/api/required-api όπως φαίνεται στην Εικόνα 26. Το αποτέλεσμα που βλέπουμε φαίνεται στην Εικόνα 27.

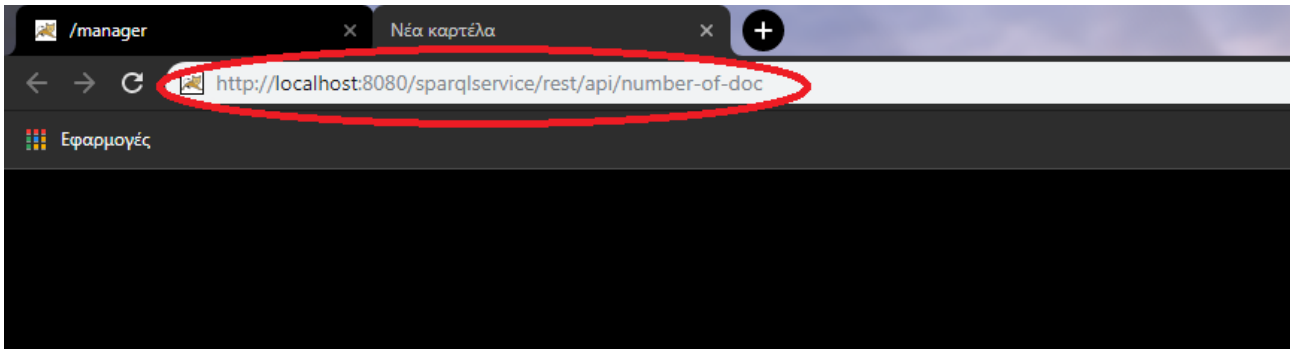


Εικόνα 26:Τύπος πρώτου ερωτήματος στον manager

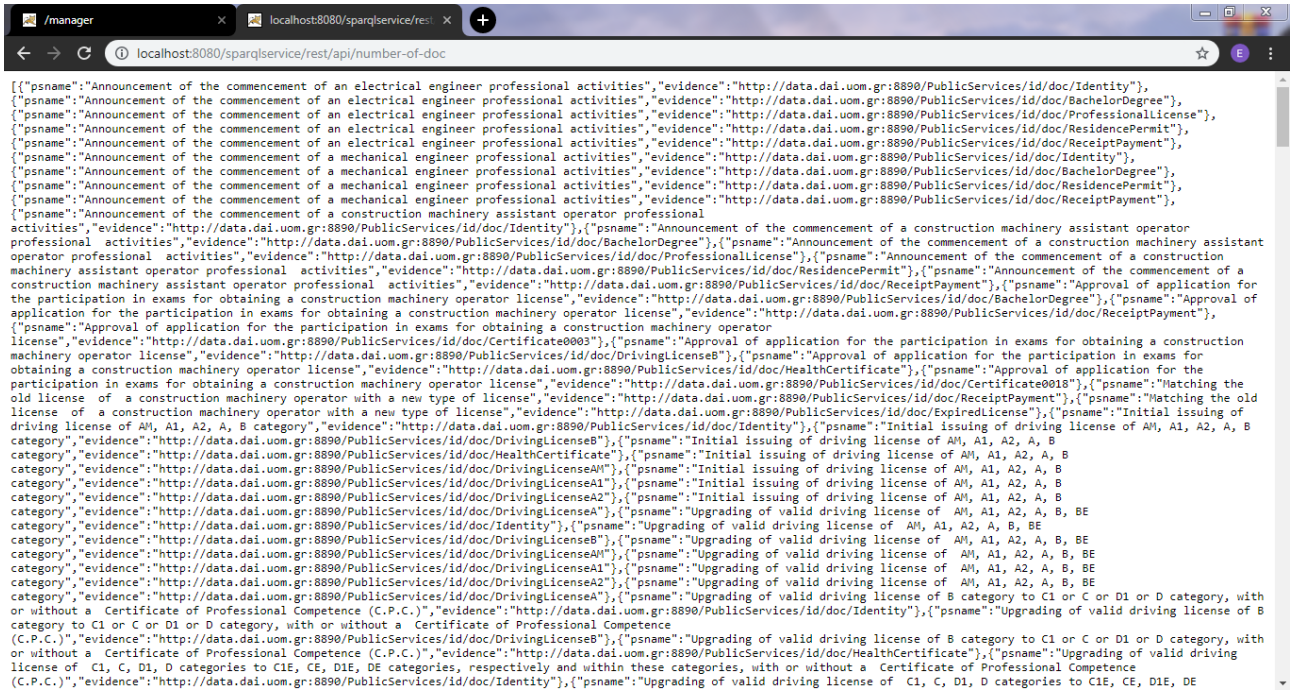


Εικόνα 27: Αποτέλεσμα πρώτου ερωτήματος

Για το δεύτερο ερώτημα χρησιμοποιούμε το localhost:8080/sparqlservice/rest/api/number-of-doc όπως φαίνεται στην Εικόνα 28. Το αποτέλεσμα που βλέπουμε φαίνεται στην Εικόνα 29.

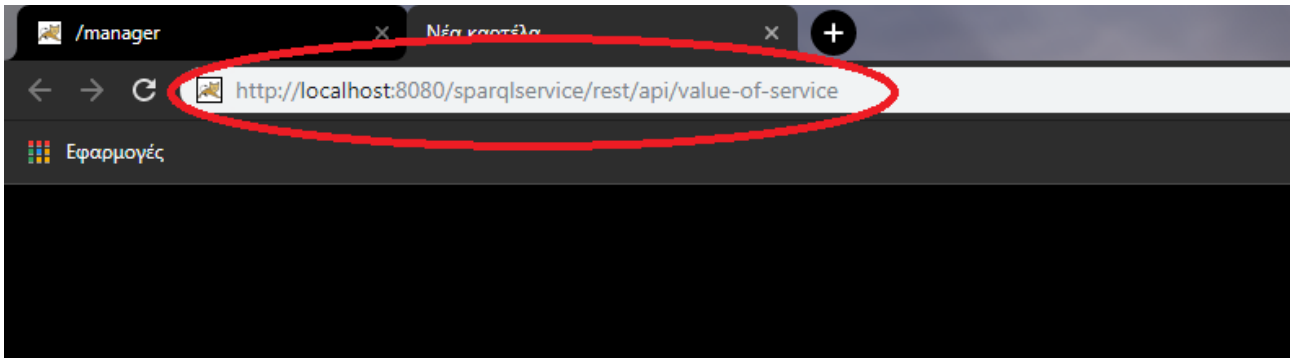


Εικόνα 28: Τύπος δεύτερου ερωτήματος στον manager

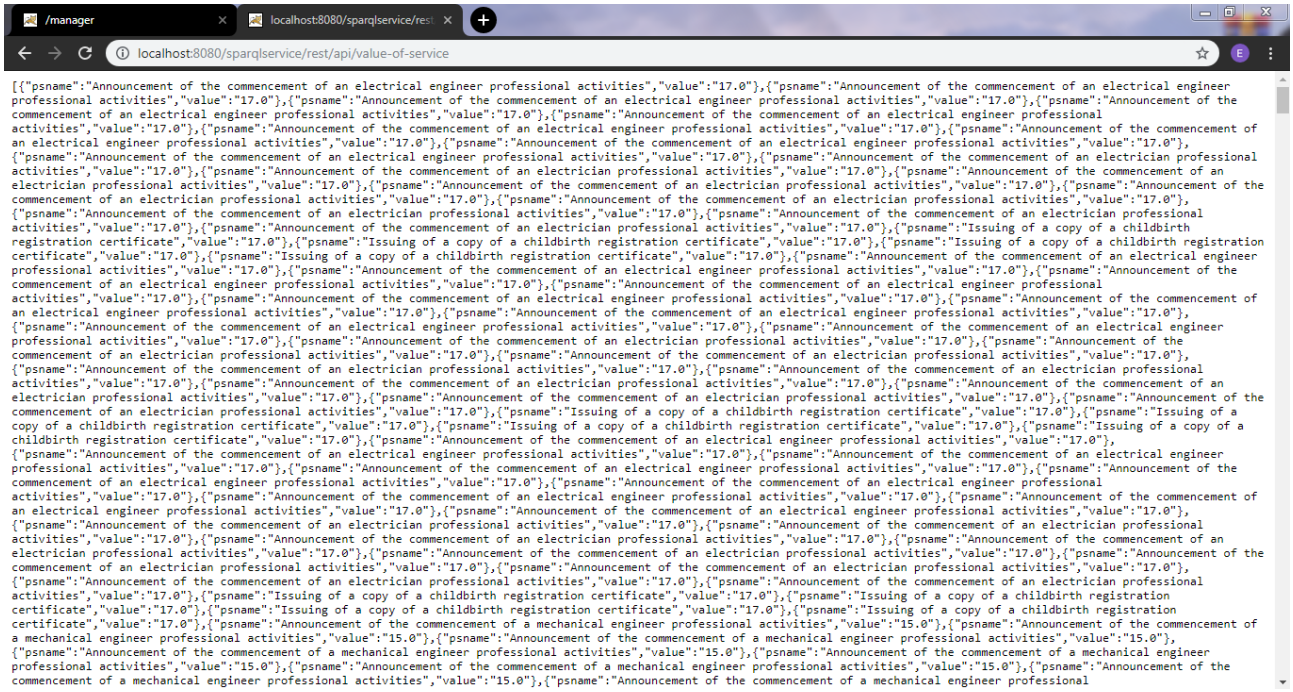


Εικόνα 29: Αποτέλεσμα δεύτερου ερωτήματος

Για το τρίτο ερώτημα χρησιμοποιούμε το localhost:8080/sparqlservice/rest/api/value-of-service όπως φαίνεται στην Εικόνα 30. Το αποτέλεσμα που βλέπουμε φαίνεται στην Εικόνα 31.



Εικόνα 30: Τύπος τρίτου ερωτήματος στον manager



Εικόνα 31: Αποτέλεσμα τρίτου ερωτήματος

4.4 SPARQL Web Application

Αφού δημιουργήθηκε το SPARQL Service, μπορεί να δημιουργηθεί και το SPARQL Web Application (Sparql WebApp). Όπως αναφέρθηκε προηγουμένως, το Sparql WebApp θα επικοινωνεί με το Service ώστε να καταναλώνονται τα δεδομένα των δημοσίων υπηρεσιών από οποίον επιθυμεί.

4.4.1 Αρχιτεκτονική συστήματος

Για την κατασκευή του Service χρησιμοποιήθηκαν οι παρακάτω τεχνολογίες:

- Apache Maven, που χρησιμοποιήθηκε ως επέκταση (plug-in) στο Eclipse.
- Spring MVC
- Spring REST
- Spring MVC, JSP (Front-End)
- Bootstrap (responsive)
- HTML, CSS, JQuery

Η εφαρμογή είναι βασισμένη στο Maven. Η συγγραφή του κώδικα, δηλαδή, έγινε εξ' ολοκλήρου σε αυτό.
Ο web server που χρησιμοποιήθηκε σε όλη τη διάρκεια της ανάπτυξης των εφαρμογών για την δοκιμή του κώδικα είναι ο Apache Tomcat.
Για τη δημιουργία της σελίδας χρησιμοποιήθηκε η τεχνολογία JSP καθώς επίσης και οι Bootstrap, CSS και JQuery. Με αυτόν τον τρόπο προσαρμόστηκε το στυλ της σελίδας ώστε να είναι responsive (Bootstrap), προσαρμόστηκε ώστε να ακολουθεί ένα συγκεκριμένο στυλ (CSS) και προσαρμόστηκε ώστε να έχει μια καλά δομημένη μορφή (JQuery).

4.4.2 Περιγραφή

Παρακάτω, παρουσιάζεται αναλυτικά η διαδικασία που ακολουθήθηκε καθώς επίσης και μέρος του κώδικα που δημιουργήθηκε για την κατασκευή του Sparql WebApp.

Αρχικά, έπρεπε να δημιουργηθεί η σύνδεση του Sparql WebApp με το SPARQL Service. Για να γίνει αυτό έπρεπε να καταναλωθούν μέσα στην εφαρμογή τα Endpoints που δημιουργήθηκαν στο προηγούμενο στάδιο, και των οποίων τα αποτελέσματα βλέπουμε στις Εικόνες 27, 29 ,31.
Όπως φαίνεται στον παρακάτω κώδικα, τα ονόματα των μεταβλητών που ανατέθηκαν τα Endpoints, είναι και πάλι δηλωτικά των ονομάτων των ερωτημάτων που θα τεθούν στο κάθε Endpoint. Επομένως, για το ερώτημα που επιστρέφει τα απαραίτητα δικαιολογητικά που απαιτούνται για την διεκπεραίωση των δημοσιών υπηρεσιών το όνομα της μεταβλητής είναι: REQUIRED_DOC_REST_ENDPOINT. Για το ερώτημα που επιστρέφει τον αριθμό των δικαιολογητικών που απαιτούνται για την διεκπεραίωση κάθε δημοσίας υπηρεσίας το όνομα της μεταβλητής είναι: NUMBER_OF_DOC_REST_ENDPOINT. Για το ερώτημα που επιστρέφει το κόστος της διεκπεραίωσης κάθε δημοσίας υπηρεσίας το όνομα της μεταβλητής είναι:

VALUE_OF_SERVICE_REST_ENDPOINT.

```
public interface RestEndpoints {

	String REQUIRED_DOC_REST_ENDPOINT =
"http://localhost:8080/sparqlservice/rest/api/required-doc";
	String NUMBER_OF_DOC_REST_ENDPOINT =
"http://localhost:8080/sparqlservice/rest/api/number-of-doc";
	String VALUE_OF_SERVICE_REST_ENDPOINT =
"http://localhost:8080/sparqlservice/rest/api/value-of-service";

}
```

Στη συνέχεια ακολουθεί η περιγραφή της κατασκευής του Spring Model View Controller που χρησιμοποιήθηκε για την δημιουργία του Sparql WebApp.

Αρχικά, έπρεπε να δοθούν οι πληροφορίες σχετικά με την εφαρμογή και τη διαμόρφωσή της σε ένα αρχείο με όνομα pom.xml. Ο κώδικας του συγκεκριμένου αρχείου φαίνεται παρακάτω.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>SparqlWebApp</groupId>
  <artifactId>SparqlWebApp</artifactId>
  <version>1.0</version>
  <packaging>war</packaging>
  <name>sparqlwebapp</name>
  <build>
    <sourceDirectory>src</sourceDirectory>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.5.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
      <plugin>
        <artifactId>maven-war-plugin</artifactId>
        <version>3.0.0</version>
        <configuration>
          <warSourceDirectory>WebContent</warSourceDirectory>
        </configuration>
      </plugin>
    </plugins>
  </build>
  <dependencies>
  <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
```

```
            <version>3.1.0</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>5.1.7.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.1.7.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>5.1.7.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>5.1.7.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
            <version>5.1.7.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>javax.servlet.jsp.jstl</groupId>
            <artifactId>jstl-api</artifactId>
            <version>1.2</version>
        </dependency>


    </dependencies>
</project>
```

Μπορούμε να δούμε ότι στο συγκεκριμένο κώδικα δίνονται ουσιαστικά πληροφορίες σχετικά με τις τεχνολογίες οι οποίες χρησιμοποιήθηκαν. Στο τμήμα του κώδικα με το όνομα <dependency> δημιουργούνται οι εξαρτήσεις που θα χρησιμοποιηθούν στην εφαρμογή μας και αυτό αποσκοπεί στο να μην χρειάζεται να προσδιορίσουμε αργότερα τις βιβλιοθήκες που θα χρειαστούν για τις εξαρτήσεις αυτές.

Στη συνέχεια, δημιουργήθηκε ο ελεγκτής (Controller). Ο Controller, όπως προαναφέρθηκε, είναι αυτός που χειρίζεται τις αιτήσεις HTTP. Για να δημιουργηθεί ο Controller χρησιμοποιείται το σχόλιο @Controller, που σημειώνει την κλάση ως Ελεγκτή, το σχόλιο @RequestMapping, που χρησιμοποιείται για να καθορίσει την κλάση στην οποία είναι θα γίνει ο έλεγχος.

```
@Controller
public class HomeController {
```

```
	@RequestMapping("/")
	public ModelAndView home(){
		return new ModelAndView("home","apiBean", new ApiBean());
	}

	@RequestMapping(value="/getApiResult", method=RequestMethod.POST)
	public String getApiResult( @ModelAttribute("apiBean")ApiBean, ModelMap model ){
		System.out.println("api name: "+apiBean.getApiName());
		apiBean.setApiResult("");

		if("requiredDocApi".equalsIgnoreCase(apiBean.getApiName())){
			apiBean.setApiResult(restClient.getRequiredDoc());

		}else if("numberOfDocApi".equalsIgnoreCase(apiBean.getApiName())){
			apiBean.setApiResult(restClient.getNumberOfDoc());

		}else if("valueOfServiceApi".equalsIgnoreCase(apiBean.getApiName())){
			apiBean.setApiResult(restClient.getValueOfService());
		}
		return "home";
	}

}
```

Αφού δημιουργήθηκε και ο Controller έπρεπε να κατασκευαστεί και η σελίδα που θα εμφανίζει τα αποτελέσματα μας. Η σελίδα είναι μια JSP σελίδα (Java Server Page) και ο κώδικάς της φαίνεται παρακάτω.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>
	<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
	<title>sparql WebApp</title>

	<script src="${pageContext.request.contextPath}/js/jquery-3.4.1.min.js"></script>
	<script src="${pageContext.request.contextPath}/js/popper.min.js"></script>

    <link rel="stylesheet"
href="${pageContext.request.contextPath}/css/bootstrap.min.css" />
    <script src="${pageContext.request.contextPath}/js/bootstrap.min.js"></script>

	<link rel="stylesheet" href="${pageContext.request.contextPath}/css/style.css" >
	<link href="https://fonts.googleapis.com/css?family=Open+Sans" rel="stylesheet">
	<script type="text/javascript">
		function onApiClick(id) {
			if (id == "requiredDocApi") {
				document.getElementById('requiredDocApi').style="background-
color:#218838 !important";
```

```
                        document.getElementById('numberOfDocApi').style="background-
color:#269abc !important";

      document.getElementById('valueOfServiceApi').style="background-
color:#269abc !important";

                        $("#errorMsg").hide();
                        $("#executeBtn").removeAttr("disabled");
                        $("#apiName").val("requiredDocApi");

                  } else if (id == "numberOfDocApi") {

                        document.getElementById('numberOfDocApi').style="background-
color:#218838 !important";
                        document.getElementById('requiredDocApi').style="background-
color:#269abc !important";

      document.getElementById('valueOfServiceApi').style="background-
color:#269abc !important";

                        $("#errorMsg").hide();
                        $("#executeBtn").removeAttr("disabled");
                        $("#apiName").val("numberOfDocApi");
                  } else {


      document.getElementById('valueOfServiceApi').style="background-
color:#218838 !important";
                        document.getElementById('numberOfDocApi').style="background-
color:#269abc !important";
                        document.getElementById('requiredDocApi').style="background-
color:#269abc !important";

                        $("#errorMsg").hide();
                        $("#executeBtn").removeAttr("disabled");
                        $("#apiName").val("valueOfServiceApi");
                  }
            }

            function invokeApi(){
                  $("#invokeApi").click();
            }

            function clearResult(){
                  $("#apiResult").val("");
            }

      </script>

</head>

<body style="background-color:#dae0e5 !important;">

      <div class="text-center" style="color: white !important;">
            <h1 style="line-height: 100px">SPARQL WEBAPP</h1>
      </div>

    <form:form method="POST"
                        action="${pageContext.request.contextPath}/getApiResult"
                        modelAttribute="apiBean">
            <div class="container">
```

```
                    <div class="row">
                          <div class="col-sm-6"
                                style="border-right: double; margin-top: 20px">
                                <h1>Select API</h1>

                                <a href="#" name="requiredDocApi" id="requiredDocApi"
onclick="onApiClick('requiredDocApi')">
                                     <b>1.Required Doc</b><br> Documents required to
perform a service
                                </a><br>
                                <a href="#" id="numberOfDocApi" name="numberOfDocApi"
onclick="onApiClick('numberOfDocApi')">
                                    <b>2.Number of Doc</b><br>How many documents
requires each public service as Imported to be processed
                                </a><br>
                                <a href="#" id="valueOfServiceApi"
name="valueOfServiceApi" onclick="onApiClick('valueOfServiceApi')">
                                    <b>3. Value of Service</b><br>What is the cost of
handling each described public service
                                </a><br>

                          </div>
                          <form:hidden path="apiName" id="apiName" />
                          <div class="col-sm-4">

                                <input type="submit" value="Execute" id="invokeApi"
                                      style="display: none;">
                                <button name="executeBtn" id="executeBtn" type="submit"
class="btn" onClick="invokeApi();" disabled="true">Execute</button>


                          </div>

                          <div class="col-sm-2" style="margin-top:175px;">

                                <span style="color:red;" id="errorMsg"
name="errorMsg"><b>Please Select an API</b></span>

                          </div>
                          <h1>Result</h1>
                          <a href="#" name="requiredDocApi" class="btn btn-primary"
id="requiredDocApi" onclick="clearResult()" style="margin:10px;background-color:
#337ab7 !important;">
                                    Clear Result
                                </a>
                          <form:textarea class="form-control col-md-12 "
path="apiResult" disabled="true" />

                    </div>
              </div>
        </form:form>

        <script type="text/javascript">

            console.log("selected API : "+valueOfServiceApi);

            if($('#apiResult').val() != ""){
                console.log("data not empty");
                      var tmpData = JSON.parse($('#apiResult').val());
                      var formattedData = JSON.stringify(tmpData, null, '\t');
                      $('#apiResult').text(formattedData);
```

```
            }
        </script>
</body>
</html>
```

Για την σελίδα χρησιμοποιήθηκε CSS βασισμένο σε Bootstrap το οποίο περιλαμβάνει και επεκτάσεις JavaScript. Όπως φαίνεται στον παραπάνω κώδικα, οι επεκτάσεις αυτές είναι επεκτάσεις jQuery και Popper.js και εισήχθησαν στο κομμάτι της κεφαλίδας (<head>) με την μορφή:

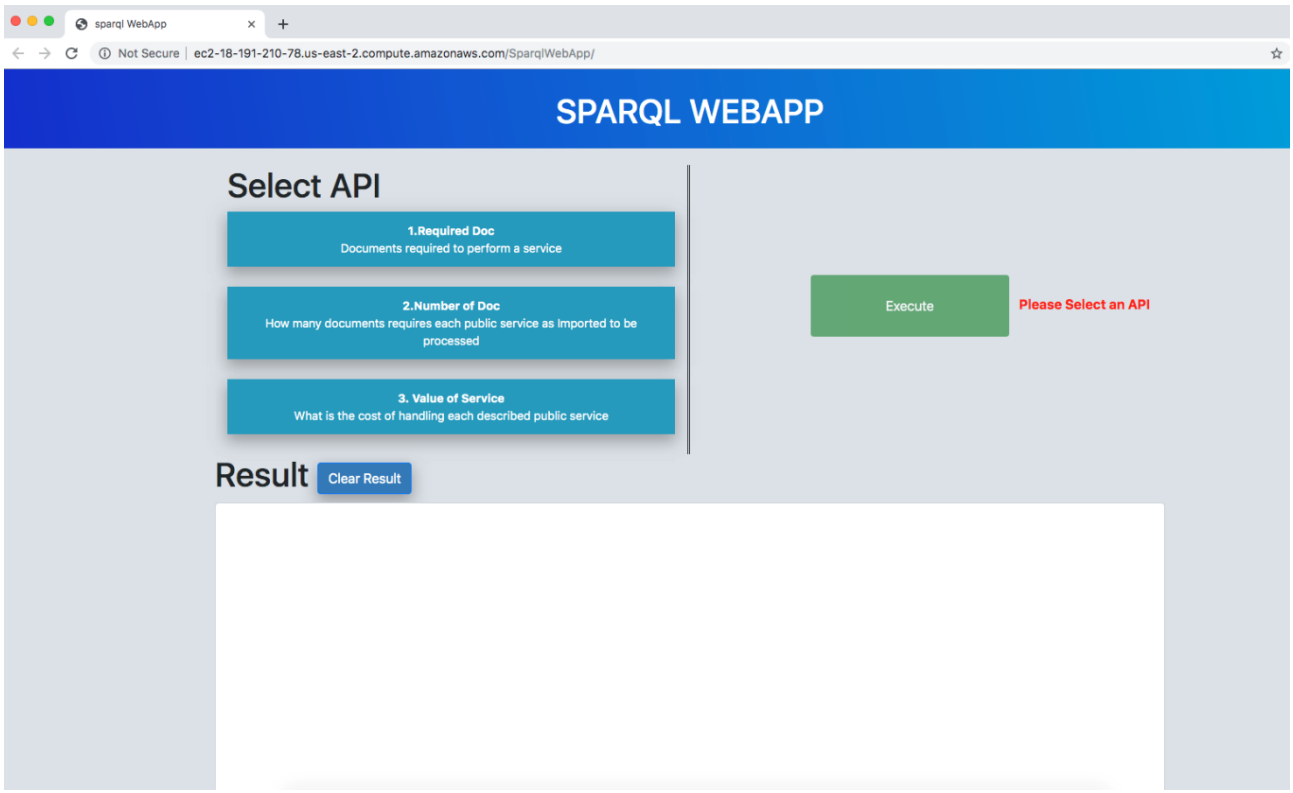
```
<script src="${pageContext.request.contextPath}/js/jquery-3.4.1.min.js"></script>
<script src="${pageContext.request.contextPath}/js/popper.min.js"></script>
```

όπου χρησιμοποιήθηκε το μονοπάτι (path) στο οποίο βρίσκονται για να δηλωθούν.
Ο κώδικας του Bootstrap CSS καθώς επίσης και των jQuery, Popper.js βρίσκονται στο Παράρτημα Β στο τέλος της διπλωματικής.

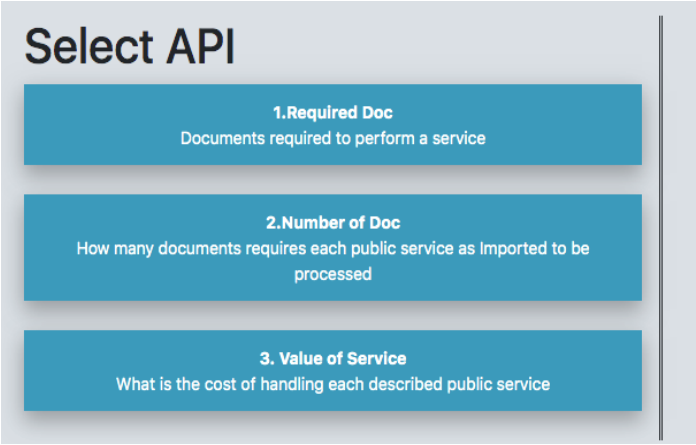
4.4.3 Παρουσίαση

Για να γίνει απολυτά κατανοητό το πως λειτουργεί ο παραπάνω κώδικας, ακολουθούν στιγμιότυπα της εφαρμογής μετά την ολοκλήρωσή της.

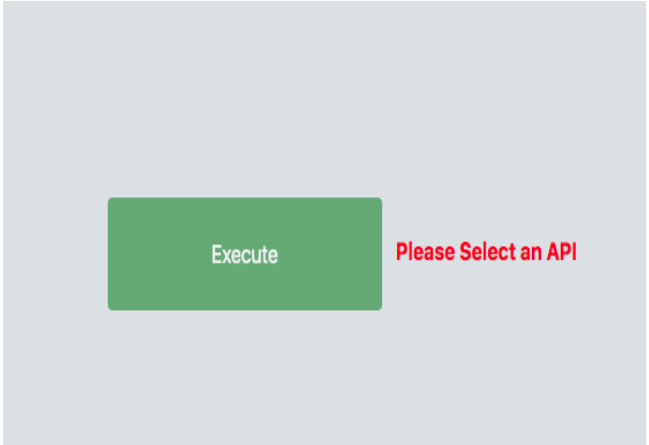
Στην Εικόνα 32 φαίνεται το περιβάλλον του Sparql WebApp. Στο πάνω τμήμα της σελίδας στο κέντρο υπάρχει ο τίτλος (SPARQL WEBAPP). Στα αριστερά φαίνονται τα ερωτήματα, δεξιά υπάρχει η επιλογή της εκτέλεσης των ερωτημάτων και στο κάτω μέρος της σελίδας βρίσκεται το τμήμα στο οποίο παρουσιάζονται τα αποτελέσματα των ερωτημάτων μας.



Εικόνα 32:Το περιβάλλον του Sparql WebApp



Εικόνα 34: Μενού ερωτημάτων



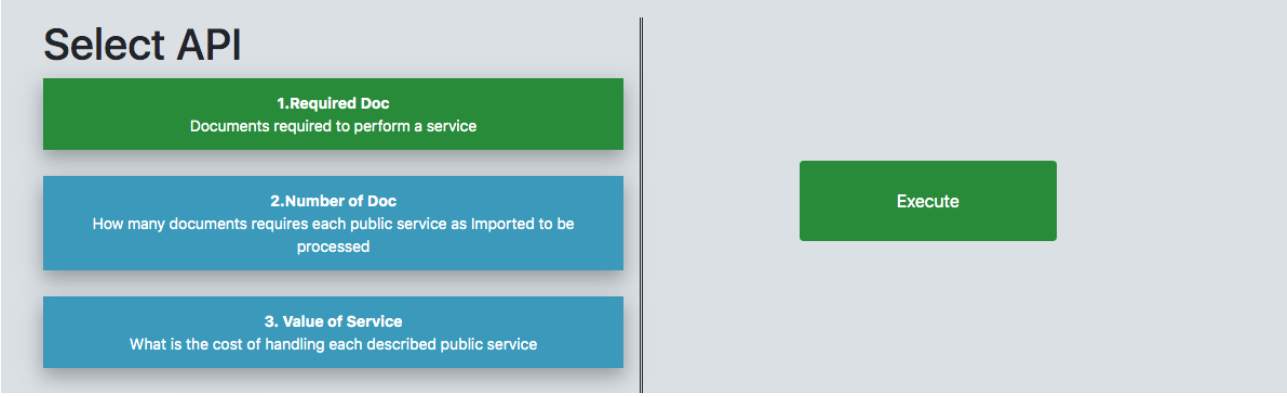
Εικόνα 33: Κουμπί εκτέλεσης

Στην Εικόνα 33 φαίνεται το μενού από το οποίο μπορούμε να επιλέξουμε το ερώτημα το οποίο θέλουμε να εκτελέσουμε. Κάθε ερώτημα έχει ένα όνομα (1. Required Doc) και την περιγραφή του (Documents required to perform a service).

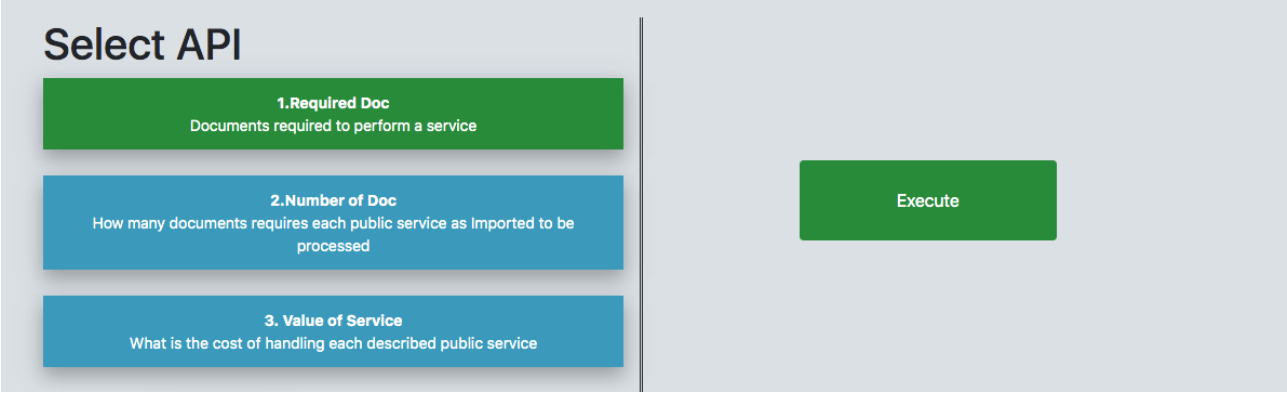
Στην Εικόνα 34 φαίνεται το κουμπί το οποίο χρησιμοποιείται για να εκτελέσουμε το ερώτημα που έχουμε επιλέξει από το μενού. Το συγκεκριμένο κουμπί έχει το όνομα Execute. Διπλά από το κουμπί εμφανίζεται το μήνυμα “Please Select an API” όταν δεν έχουμε επιλέξει κανένα ερώτημα.

Στην Εικόνα 35 βλέπουμε πως όταν επιλέγουμε ένα ερώτημα, το μήνυμα “Please Select an API” δεν εμφανίζεται πλέον.

Στην Εικόνα 35 και στην Εικόνα 36 μπορούμε να δούμε ότι όταν ένα ερώτημα επιλέγεται, αλλάζει χρώμα. Από μπλε που είναι όταν δεν είναι επιλεγμένο, γίνεται πράσινο.

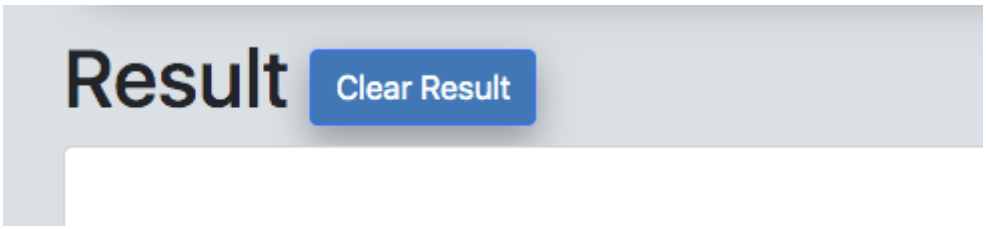


Εικόνα 35: Επιλογή ερωτήματος

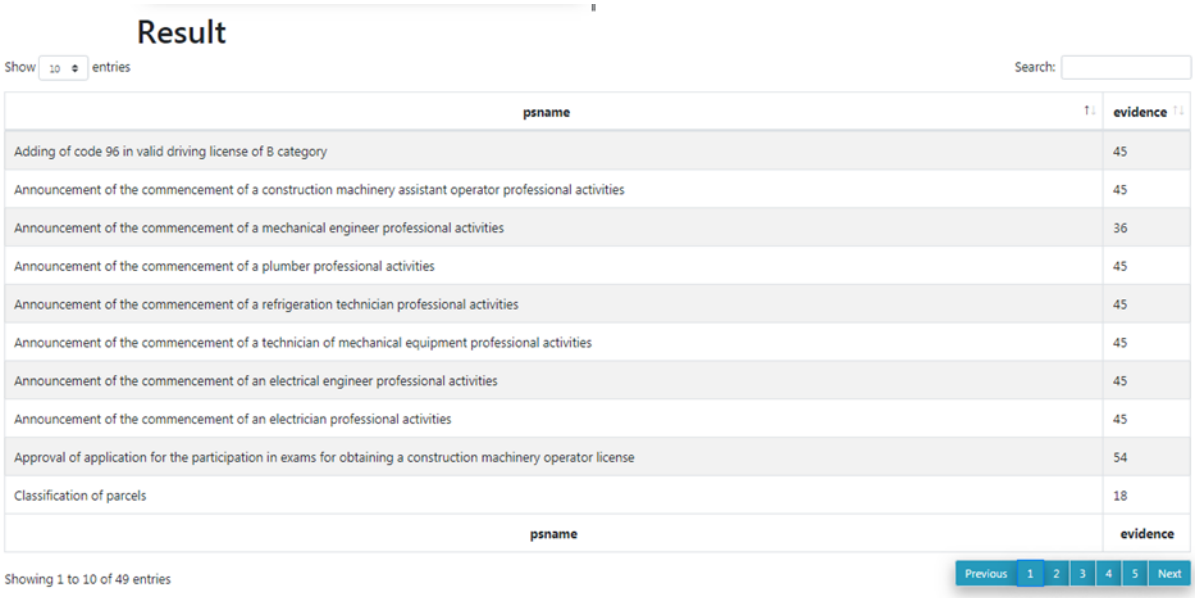


Εικόνα 36: Επιλεγμένο ερώτημα

Στην Εικόνα 37 φαίνεται άλλη μια επιλογή του περιβάλλοντος του Sparql WebApp. Αυτή η επιλογή είναι το Clear Result. Με την συγκεκριμένη επιλογή “καθαρίζουμε” το τμήμα στο οποίο εμφανίζονται τα αποτελέσματα της εκτέλεσης ενός ερωτήματος.



Εικόνα 37: Κουμπί Clear Result


Result

Show 10 entries

Search:

psname	evidence
Adding of code 96 in valid driving license of B category	45
Announcement of the commencement of a construction machinery assistant operator professional activities	45
Announcement of the commencement of a mechanical engineer professional activities	36
Announcement of the commencement of a plumber professional activities	45
Announcement of the commencement of a refrigeration technician professional activities	45
Announcement of the commencement of a technician of mechanical equipment professional activities	45
Announcement of the commencement of an electrical engineer professional activities	45
Announcement of the commencement of an electrician professional activities	45
Approval of application for the participation in exams for obtaining a construction machinery operator license	54
Classification of parcels	18
psname	evidence

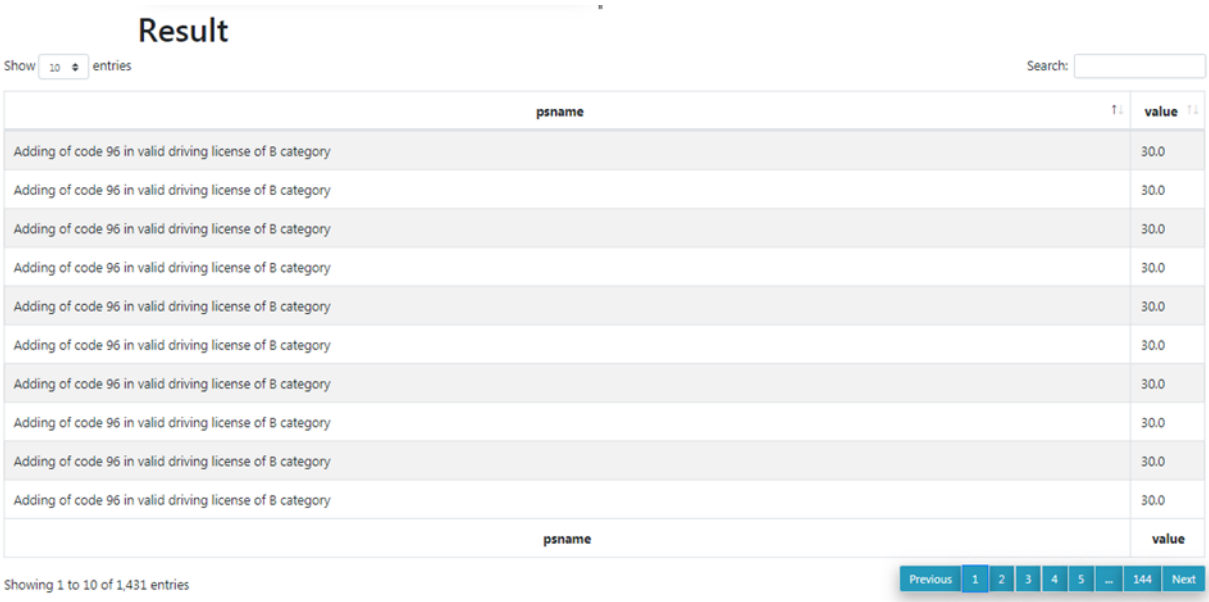
Showing 1 to 10 of 49 entries

Previous 1 2 3 4 5 Next

Εικόνα 38: Αποτέλεσμα εκτέλεσης πρώτου ερωτήματος στο Sparql WebApp

Στην Εικόνα 38 φαίνεται το αποτέλεσμα που μας δίνεται μετά την εκτέλεση του πρώτου ερωτήματος το οποίο είναι αυτό που μας δίνει τον αριθμό των δικαιολογητικών που απαιτούνται για την διεκπεραίωση κάθε δημόσιας υπηρεσίας.

Στην Εικόνα 39 φαίνεται το αποτέλεσμα που μας δίνεται μετά την εκτέλεση του τρίτου ερωτήματος το οποίο είναι αυτό που μας δίνει το κόστος που απαιτείται για την διεκπεραίωση κάθε δημόσιας υπηρεσίας.



Εικόνα 39: Αποτέλεσμα εκτέλεσης τρίτου ερωτήματος στο Sparql WebApp

Στις Εικόνες 38 και 39 βλέπουμε ότι τα αποτελέσματα εμφανίζονται σε μορφή πίνακα. Υπάρχει η δυνατότητα επιλογής του αριθμού των αποτελεσμάτων που θα εμφανιστούν (στην περίπτωση αυτή βλέπουμε ότι είναι 10) . Επίσης, υπάρχει στο κάτω μέρος του πίνακα, δυνατότητα πλοήγησης μεταξύ των σελίδων με τα αποτελέσματα.

Το κυριότερο χαρακτηριστικό που βλέπουμε στις Εικόνες 38 και 39 είναι το μενού αναζήτησης που εμφανίζεται στα αποτελέσματα (Search menu). Με αυτό, δίνεται η δυνατότητα στον χρήστη να αναζητήσει για ποια υπηρεσία θα ήθελε να δει το κόστος διεκπεραίωσής της ή για ποια υπηρεσία θα ήθελε να δει τα δικαιολογητικά τα οποία πρέπει να προσκομίσει για την διεκπεραίωσής της.

5. ΕΠΙΛΟΓΟΣ

Στο παρόν κεφάλαιο παραθέτετε ένα σύνολο παρατηρήσεων που έγιναν αντιληπτές κατά τη διάρκεια εκπόνησης της συγκεκριμένης εργασίας. Παράλληλα, θα γίνει αναφορά ορισμένων πιθανών μελλοντικών επεκτάσεων που μπορούν να πραγματοποιηθούν.

5.1 Συμπεράσματα

Η συγκέντρωση και η δημοσίευση των δεδομένων των Δημοσίων Υπηρεσιών με την τεχνική των συνδεδεμένων δεδομένων, βοηθάει στην πιο εύκολη ανάκτηση δεδομένων από τους πολίτες, τους οργανισμούς, τις επιχειρήσεις, ακόμη και από τους ιδίους τους δημοσίους φορείς. Οι δημόσιοι φορείς θα πρέπει να συγκεντρώσουν κι άλλα δεδομένα και να τα μετατρέψουν κι αυτά σε Ανοικτά Συνδεδεμένα Δεδομένα ώστε να υπάρχει μεγάλος όγκος δεδομένων για να μπορούν να καλυφθούν οι ανάγκες , που μεγαλώνουν ολοένα και περισσότερο με την πάροδο των χρόνων.

Στην παρούσα διπλωματική εργασία καταφέραμε να δημιουργήσουμε εφαρμογές που βοηθούν στην ανάδειξη των συνδεδεμένων δεδομένων των δημοσίων υπηρεσιών ώστε να είναι αναγνώσιμα και διαχειρίσιμα από όποιον επιθυμεί να τα καταναλώσει. Οι τεχνικές που ακολουθήθηκαν είναι τεχνικές που μπορούν εύκολα να ακολουθηθούν και να πετύχουν. Επίσης, η χρησιμότητα των εφαρμογών που δημιουργήθηκαν θεωρείται μεγάλη, αφού στη χώρα μας δεν υπάρχουν αντίστοιχες.

Αναλυτικότερα, τα δεδομένα που χρησιμοποιήθηκαν και καταναλώνονται στις εφαρμογές που δημιουργήθηκαν, είναι Ανοικτά Συνδεδεμένα Δεδομένα που δημιουργήθηκαν με βάση το μοντέλο CPSV-AP. Το συγκεκριμένο μοντέλο μπορεί να αξιοποιηθεί και από άλλες εφαρμογές για τον ίδιο σκοπό, αφού αποτελεί βάση για την ανταλλαγή πληροφοριών μιας και είναι η “γλώσσα” που απαιτείται για τον κοινό προσδιορισμό των δημοσίων συνδεδεμένων δεδομένων σε ευρωπαϊκό

επίπεδο. Επίσης, το μοντέλο μπορεί να χρησιμοποιηθεί εύκολα διότι παρέχονται αναλυτικές λεπτομέρειές του από την PwC EU Services.

Για τη δημιουργία των εφαρμογών χρησιμοποιήθηκε ως βάση η τεχνολογία Spring (Spring MVC, Spring REST). Η συγκεκριμένη τεχνολογία επιλέχθηκε γιατί βοηθάει στο να ενωθούν διαφορετικά συστατικά μεταξύ τους. Στο πλαίσιο αυτής της εργασίας, τα διαφορετικά συστατικά είναι: τα ερωτήματα SPARQL που έπρεπε να ανακτηθούν και έπειτα να καταναλωθούν, η δημιουργία ενός RESTful service και έπειτα η δημιουργία μιας εφαρμογής ιστού. Με την ενασχόληση με τη συγκεκριμένη τεχνολογία, παρατηρήθηκε ότι δίνεται μια εξαιρετική υποστήριξη για RESTful υπηρεσίες και στο τέλος ο κώδικας είναι “καθαρός”, αφού δεν υπάρχουν πολλές εξαρτήσεις και κλήσεις μέσα στις κλάσεις λόγω του ότι παρέχονται αρκετές που καταναλώνουν απευθείας τους πόρους και επίσης παρέχονται σχόλια (annotation) που στέλνουν αμέσως τις αποκρίσεις στους χρήστες.

5.2 Μελλοντικές επεκτάσεις

Στα πλαίσια αυτής της διπλωματικής εργασίας, μελετήθηκαν δεδομένα της πύλης Ο Οδηγός του Πολίτη της Περιφέρειας Ηπείρου. Τα δεδομένα αυτά είναι δεδομένα περιγραφών δημοσίων υπηρεσιών που έχουν μετατραπεί σε RDF δεδομένα. Είναι δημοσιευμένα ως Ανοικτά Συνδεδεμένα Δεδομένα στο αποθετήριο γράφων του Πανεπιστήμιου Μακεδονίας (http://data.dai.uom.gr:8890/). Παρακάτω παρουσιάζονται μερικές πιθανές επεκτάσεις που θα μπορούσαν να γίνουν στις εφαρμογές.

Αρχικά, θα μπορούσαν να δημιουργηθούν ερωτήματα και κατ’ επέκταση εφαρμογές για όλα τα πιθανά σενάρια χρήσης των δεδομένων των δημοσίων υπηρεσιών και όχι μόνο για τα δεδομένα του Οδηγού του Πολίτη της Περιφέρειας Ηπείρου. Μπορεί τα δεδομένα να είναι παρόμοια για όλους τους Οδηγούς όλων των Περιφερειών, όμως, όπως έχει αναφερθεί, κάθε περιφέρεια θα μπορούσε να έχει ορίσει διαφορετικά τα δεδομένα της.

Επίσης, θα μπορούσε μελλοντικά να υπάρξει συνεργασία μεταξύ των φορέων ώστε να δημιουργηθεί μια εφαρμογή ιστού, οπού θα παρουσιάζονται τα δεδομένα με τη μορφή που τα βλέπουμε στην εφαρμογή ιστού που έχουμε δημιουργήσει σε αυτή την εργασία, ώστε να είναι

ευκολά προσβάσιμα από όλους.

Τέλος, μια επέκταση θα μπορούσε να γίνει στον τρόπο που εμφανίζονται τα αποτελέσματα στον χρήστη. Θα μπορούσαν, για παράδειγμα, να εμφανίζονται τα δεδομένα σε μορφή πίνακα αντί για τη μορφή JSON που εμφανίζονται τώρα. Αυτό θα βοηθήσουμε στην πιο εύκολη αναζήτηση, από την πλευρά του χρήστη, των δεδομένων που απαιτούνται.

Βιβλιογραφικές αναφορές

Armstrong, P., Steward, M. & Ward, A. (2013). *Case study: using linked data to integrate resources from cultural heritage institutions across Canada*. Available at: https://or2013.net/sites/or2013.net/files/Integrating%20Resources%20with%20Linked%20Data/index.pdf.

Berners-Lee, T. (2006). *Linkeddata - designissues*. Ανακτήθηκε Φεβρουάριος 2019. www.w3.org/designissues/linkeddata.html.

Bizer, C. & Schultz, A. (2009). *The berlin SPARQL benchmark. International Journal of Semantic Web and Information Systems*, 5(2), 1–24.

Bizer, C. (2009). *The Emerging Web of Linked Data*. *IEEE Intelligent Systems*, 24(5), 87-92.

Bizer, C., Heath, T. & Berners-Lee, T. (2009*). Linked Data-The Story So Far*. *International Journal on Semantic Web and Information Systems,* 5(3), 1-22.

Bizer, C., Jentzsch, A. & Cyganiak, R. (2011). *The State of the LOD Cloud.*

Bleecker, J. (2005). *A Design Approach for the Geospatial Web*. *O'Reilly Network*. Ανακτήθηκε Ιανουάριος 2019. http://www.oreillynet.com/pub/a/network/2005/06/07/geospatialweb.html

Casanova, M. A., Breitman, K., Brauner, D.F. & Marins, A. (2007). *Database conceptual schema matching. Computer* (LONGBEACH), 40102, 104.

Gruber, T. (1993). A translation approach to port able onto logy specifications. *Knowledge Acquisition,* 5,199-22.

Dulong de Rosnay, M., Janssen, K. (2014) . Legal and Institutional Challenges for Opening Data across Public Sectors: Towards Common Policy Solutions. Journal of Theoretical and Applied Electronic Commerce Research (JTAER), Universidad de Talca – Chile. Special Issue "Transparency and Open Data Policies", 9 (3),:1-14.Gur N. R., Sanchez, L. D. & Kauppinen, T. (2012). Gi systems for public health with an ontology-based approach: Proceedings of the Agile. International Conferenceon Geographic Information Science, Avignon.

Janowicz, K., Hitzler, P., Adams, B., Kolas, D. & Vardeman II, C. (2014). *Five stars of linked data vocabulary use. Semantic Web,* 5(3), 173–176.

jayg, (2015). *5 Star Open Data*. Ανακτήθηκε Φεβρουάριος 2019. 5stardata.info/en/

Klyne, G. & Carroll, J. (2004*). Resource Description Framework (RDF): Concepts and Abstract Syntax*.

Manola, F. & Miller, E. (2014). *RDF Primer.*

O'Reilly, T. & Malamud, C. (2007). *Open Government*. Working Group.

Rouse, M., (2019). *RESTful API*. Ανακτήθηκε Φεβρουάριος 2019 https://searchmicroservices.techtarget.com/definition/RESTful-API

Shah, K., Fouzia, J., Rahman, S., Iftikhar, A., (2014). *Linked Open Data: Towards the Realization of Semantic Web-A Review.* Indian Journal of Science and Technology. 7. 745-764.

Yates, A., Beal, K., Keenan, S., McLaren, W., Pignatelli, M. & Ritchie, GRS. (2015). *The Ensembl REST API: Ensembl Data for Any Language*. *Bioinformatics,* 31(1), 143-5.

Zikopoulos, P., Eaton, C. (2011). *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data (1st ed.).* McGraw-Hill Osborne Media.

Semantic Interoperability Courses: Course Modyle 2 core vocabularies, 2014

PwC EU Services. (2019). *Core Vocabularies*. Ανακτήθηκε Μάρτιος 2019 https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/core-vocabularies

PwC EU Services. (2016). *Core Public Service Vocabulary*. Ανακτήθηκε Μάρτιος 2019 https://joinup.ec.europa.eu/solution/core-public-service-vocabulary/about

PwC EU Services. (2016). *Piloting the Core Public Service Vocabulary.* Ανακτήθηκε Μάρτιος 2019 https://joinup.ec.europa.eu/sites/default/files/document/2013-05/Piloting%20the%20Core%20Public%20Service%20Vocabulary.pdf

PwC EU Services. (2019). *Core Public Service Vocabulary Application Profile.* Ανακτήθηκε Μάρτιος 2019 https://joinup.ec.europa.eu/release/core-public-service-vocabulary-application-profile-v20

Apache Tomcat Welcome. *Apache Tomcat*. Ανακτήθηκε Φεβρουάριος2019 http://tomcat.apache.org/

W3C. (2019). *Apache Jena*. Ανακτήθηκε Φεβρουάριος2019 https://www.w3.org/2001/sw/wiki/Apache_Jena

Shameer Kunjumohamed, Hamidreza Sattari, Alex Bretet, Geoffroy Warin. (2016). *Spring MVC: Designing Real-World Web Applicarions*.

Matthew Tyson. (2019). *What is JSP? Introduction to JavaServer Pages*. Ανακτήθηκε Φεβρουάριος2019 https://www.javaworld.com/article/3336161/what-is-jsp-introduction-to-javaserver-pages.html

W3C. (2019). *Bootstrap Get Started.* Ανακτήθηκε Φεβρουάριος2019 https://www.w3schools.com/bootstrap/bootstrap_get_started.asp

Παράρτημα Α' - Κώδικας εφαρμογής SPARQL Service

Error Bean

```
package com.sparql.rest.beans;

import org.springframework.stereotype.Component;

@Component("eb")
public class ErrorBean {

	private String errorCode;
	public String getErrorCode() {
		return errorCode;
	}
	public void setErrorCode(String errorCode) {
		this.errorCode = errorCode;
	}
	public String getErrorMessage() {
		return errorMessage;
	}
	public void setErrorMessage(String errorMessage) {
		this.errorMessage = errorMessage;
	}
	private String errorMessage;
}
```

Number of Doc Query Result Bean

```
import com.fasterxml.jackson.annotation.JsonPropertyOrder;

@JsonPropertyOrder({"psname","evidence"})
public class NumberOfDocQueryResultBean {

	private String PSname;
	private String evidence;


	public String getPSname() {
		return PSname;
	}
	public void setPSname(String pSname) {
		PSname = pSname;
	}
	public String getEvidence() {
		return evidence;
	}
	public void setEvidence(String evidence) {
		this.evidence = evidence;
	}
}
```

Required Query Bean

```
package com.sparql.rest.beans;

import com.fasterxml.jackson.annotation.JsonPropertyOrder;
```

```
@JsonPropertyOrder({"psname","evidence"})
public class RequiredDocQueryResultBean {

	private String PSname;
	private String evidence;


	public String getPSname() {
		return PSname;
	}
	public void setPSname(String pSname) {
		PSname = pSname;
	}
	public String getEvidence() {
		return evidence;
	}
	public void setEvidence(String evidence) {
		this.evidence = evidence;
	}
}
```

Value of Service Bean

```
package com.sparql.rest.beans;


import com.fasterxml.jackson.annotation.JsonPropertyOrder;

@JsonPropertyOrder({"psname","value"})
public class ValueOfServiceQueryResultBean {

	private String PSname;
	public String getPSname() {
		return PSname;
	}
	public void setPSname(String pSname) {
		PSname = pSname;
	}
	public String getValue() {
		return value;
	}
	public void setValue(String value) {
		this.value = value;
	}
	private String value;
}
```

Rest Service Configuration

```
package com.sparql.rest.beans;


import com.fasterxml.jackson.annotation.JsonPropertyOrder;

@JsonPropertyOrder({"psname","value"})
public class ValueOfServiceQueryResultBean {

	private String PSname;
```

```
	public String getPSname() {
		return PSname;
	}
	public void setPSname(String pSname) {
		PSname = pSname;
	}
	public String getValue() {
		return value;
	}
	public void setValue(String value) {
		this.value = value;
	}
	private String value;
}
```

Rest Service Initializer

```
package com.sparql.rest.configuration;

import
org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitia
lizer;

public class SparqlRestServiceIntitializer extends
AbstractAnnotationConfigDispatcherServletInitializer{

	@Override
	protected Class<?>[] getRootConfigClasses() {
		return new Class[] {SparqlRestServiceConfiguration.class};
	}

	@Override
	protected Class<?>[] getServletConfigClasses() {
		return null;
	}

	@Override
	protected String[] getServletMappings() {
		return new String[] { "/" };
	}
}
```

Configuration

```
package com.sparql.rest.constants;

public interface Configuration {

	String SPARQL_END_POINT = "http://195.251.218.39:8890/sparql";
}
```

SPARQL Queries

```
package com.sparql.rest.constants;

public interface SparqlQueries {

	String REQUIRED_DOC_SPARQL_QUERY = "PREFIX  dct:  <http://purl.org/dc/terms/>\n"+
				"PREFIX  cpsv: <http://purl.org/vocab/cpsv#>\n"+
				"SELECT  distinct ?PSname (COUNT(?number)
AS ?evidence)\n"+
				"WHERE\n"+
				"{ ?x  dct:title      ?PSname ;\n"+
				   "cpsv:hasInput  ?number\n"+
				"}\n"+
				"GROUP BY ?PSname";

	String NUMBER_OF_DOC_SPARQL_QUERY = "PREFIX  dct:
<http://purl.org/dc/terms/>\n"+
				"PREFIX  cpsv: <http://purl.org/vocab/cpsv#>\n"+
				  "SELECT DISTINCT  ?PSname ?evidence\n"+
				  "WHERE\n"+
				  "{ ?x  dct:title      ?PSname ;\n"+
				      "cpsv:hasInput  ?evidence\n"+
				   "}";

	String VALUE_OF_SERVICE_SPARQL_QUERY = "PREFIX  dct:
<http://purl.org/dc/terms/>\n"+
				"PREFIX  cv:   <http://data.europa.eu/m8g/>\n"+
				  "SELECT  ?PSname ?value\n"+
				  "WHERE\n"+
				  "{ ?x     dct:title   ?PSname ;\n"+
				            "cv:hasCost  ?cost .\n"+
				            "?cost  cv:value    ?value\n"+
				  "}";

}
```

Controller

```
package com.sparql.rest.controllers;

import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;

import com.sparql.rest.beans.ErrorBean;
import com.sparql.rest.util.SparqlQueryEngine;

@RequestMapping("/rest/api")
@RestController
public class SparqlServiceController {

	@Autowired
```

```
SparqlQueryEngine sparqlEngine;

@Autowired
ErrorBean eb;


@RequestMapping(method = RequestMethod.GET, value = "/required-doc")
@ResponseBody()
public ResponseEntity<List<Object>> getSparqlQuery1(){
	List<Object> result = null;
	try{
		result = sparqlEngine.executeRequiredDocSparqlQuery();
		return new ResponseEntity<List<Object>>(result, HttpStatus.OK);
	}catch(Exception e){
		System.out.println("exception occured : "+e);
		//ErrorBean eb= new ErrorBean();
		eb.setErrorCode("-1");
		result = new ArrayList<Object>();
		result.add((Object)eb);
		eb.setErrorMessage("Internal Server Error Occurred");
		return new ResponseEntity<List<Object>>(result, HttpStatus.OK);
	}

}

@RequestMapping(method = RequestMethod.GET, value = "/number-of-doc")
@ResponseBody()
public ResponseEntity<List<Object>> getSparqlQuery2(){
	List<Object> result = null;
	try{
		result = sparqlEngine.executeNumberOfDocSparqlQuery();
		return new ResponseEntity<List<Object>>(result, HttpStatus.OK);
	}catch(Exception e){
		System.out.println("exception occured : "+e);
		eb.setErrorCode("-1");
		result = new ArrayList<Object>();
		result.add((Object)eb);
		eb.setErrorMessage("Internal Server Error Occurred");
		return new ResponseEntity<List<Object>>(result, HttpStatus.OK);
	}

}

@RequestMapping(method = RequestMethod.GET, value = "/value-of-service")
@ResponseBody()
public ResponseEntity<List<Object>> getSparqlQuery3(){
	List<Object> result = null;
	try{
		result = sparqlEngine.executeValueOfServiceSparqlQuery();
		return new ResponseEntity<List<Object>>(result, HttpStatus.OK);
	}catch(Exception e){
		System.out.println("exception occured : "+e);
		eb.setErrorCode("-1");
		result = new ArrayList<Object>();
		result.add((Object)eb);
		eb.setErrorMessage("Internal Server Error Occurred");
		return new ResponseEntity<List<Object>>(result, HttpStatus.OK);
	}

}
```

```
}
```

SPARQL Query Engine

```
package com.sparql.rest.util;

import java.util.ArrayList;
import java.util.List;

import org.apache.jena.query.QueryExecution;
import org.apache.jena.query.QueryExecutionFactory;
import org.apache.jena.query.QuerySolution;
import org.apache.jena.query.ResultSet;
import org.springframework.stereotype.Component;

import com.sparql.rest.beans.RequiredDocQueryResultBean;
import com.sparql.rest.beans.NumberOfDocQueryResultBean;
import com.sparql.rest.beans.ValueOfServiceQueryResultBean;
import com.sparql.rest.constants.Configuration;
import com.sparql.rest.constants.SparqlQueries;

@Component("sparqlEngine")
public class SparqlQueryEngine {

       private ResultSet executeSparqlQuery(String service, String query){
         QueryExecution e = QueryExecutionFactory.sparqlService(service,query);
         return e.execSelect();
    }

       public List<Object> executeRequiredDocSparqlQuery() throws Exception{

             List<Object> query1ResultList = new ArrayList<Object>();

             ResultSet result = executeSparqlQuery(Configuration.SPARQL_END_POINT ,
SparqlQueries.REQUIRED_DOC_SPARQL_QUERY);
             List<String> names = result.getResultVars();
         while (result.hasNext()){
             RequiredDocQueryResultBean resultBean = new RequiredDocQueryResultBean();
             QuerySolution s = result.nextSolution();

       resultBean.setPSname(s.get(names.get(0)).asLiteral().getValue().toString());

       resultBean.setEvidence(s.get(names.get(1)).asLiteral().getValue().toString());
             query1ResultList.add((Object)resultBean);

         }
         System.out.println("query1ResultList size : "+query1ResultList.size());
         return query1ResultList;

       }

   public List<Object> executeNumberOfDocSparqlQuery() throws Exception{

             List<Object> query2ResultList = new ArrayList<Object>();

             ResultSet result = executeSparqlQuery(Configuration.SPARQL_END_POINT ,
SparqlQueries.NUMBER_OF_DOC_SPARQL_QUERY);
             List<String> names = result.getResultVars();
         while (result.hasNext()){
             NumberOfDocQueryResultBean resultBean = new NumberOfDocQueryResultBean();
             QuerySolution s = result.nextSolution();
             resultBean.setPSname(s.get(names.get(0)).toString());
```

```
            resultBean.setEvidence(s.get(names.get(1)).toString());
            query2ResultList.add((Object)resultBean);

        }
        System.out.println("query2ResultList size : "+query2ResultList.size());
        return query2ResultList;

      }

   public List<Object> executeValueOfServiceSparqlQuery() throws Exception{

            List<Object> query3ResultList = new ArrayList<Object>();

            ResultSet result = executeSparqlQuery(Configuration.SPARQL_END_POINT ,
SparqlQueries.VALUE_OF_SERVICE_SPARQL_QUERY);
            List<String> names = result.getResultVars();
       while (result.hasNext()){
            ValueOfServiceQueryResultBean resultBean = new
ValueOfServiceQueryResultBean();
            QuerySolution s = result.nextSolution();
            resultBean.setPSname(s.get(names.get(0)).toString());
            resultBean.setValue(s.get(names.get(1)).toString());
            query3ResultList.add((Object)resultBean);

       }
       System.out.println("query3ResultList size : "+query3ResultList.size());
       return query3ResultList;

      }

}
```

Παράρτημα Β' – Κώδικας SPARQL Web Application

Api Bean

```
package com.sparql.webapp.beans;

public class ApiBean {

	private String apiName;
	private String apiResult;

	public String getApiResult() {
		return apiResult;
	}

	public void setApiResult(String apiResult) {
		this.apiResult = apiResult;
	}

	public String getApiName() {
		return apiName;
	}

	public void setApiName(String apiName) {
		this.apiName = apiName;
	}
}
```

Application Context Configuration

```
package com.sparql.webapp.configuration;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.view.InternalResourceViewResolver;

@Configuration
@ComponentScan("com.sparql.webapp.*")
public class ApplicationContextConfig {

    @Bean(name = "viewResolver")
    public InternalResourceViewResolver getViewResolver() {
        InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
        viewResolver.setPrefix("/WEB-INF/pages/");
        viewResolver.setSuffix(".jsp");
        return viewResolver;
    }
}
```

Spring Web application Initializer

```
package com.sparql.webapp.configuration;

import javax.servlet.FilterRegistration;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRegistration;
```

```
import org.springframework.web.WebApplicationInitializer;
import org.springframework.web.context.ContextLoaderListener;
import org.springframework.web.context.support.AnnotationConfigWebApplicationContext;
import org.springframework.web.filter.CharacterEncodingFilter;
import org.springframework.web.servlet.DispatcherServlet;

public class SpringWebAppInitializer implements WebApplicationInitializer {

    @Override
    public void onStartup(ServletContext servletContext) throws ServletException {
        AnnotationConfigWebApplicationContext appContext = new
AnnotationConfigWebApplicationContext();
        appContext.register(ApplicationContextConfig.class);

        ServletRegistration.Dynamic dispatcher =
servletContext.addServlet("SpringDispatcher",
                new DispatcherServlet(appContext));
        dispatcher.setLoadOnStartup(1);
        dispatcher.addMapping("/");


        ContextLoaderListener contextLoaderListener = new
ContextLoaderListener(appContext);

        servletContext.addListener(contextLoaderListener);


        // Filter.
        FilterRegistration.Dynamic fr = servletContext.addFilter("encodingFilter",
CharacterEncodingFilter.class);

        fr.setInitParameter("encoding", "UTF-8");
        fr.setInitParameter("forceEncoding", "true");
        fr.addMappingForUrlPatterns(null, true, "/*");
    }

}
```

Web MVC Configuration

```
package com.sparql.webapp.configuration;

import java.nio.charset.Charset;
import java.util.Arrays;
import java.util.List;

import org.springframework.context.annotation.Configuration;
import org.springframework.http.MediaType;
import org.springframework.http.converter.HttpMessageConverter;
import org.springframework.http.converter.StringHttpMessageConverter;
import
org.springframework.web.servlet.config.annotation.DefaultServletHandlerConfigurer;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

@Configuration
@EnableWebMvc
```

```
public class WebMvcConfig extends WebMvcConfigurerAdapter {

    private static final Charset UTF8 = Charset.forName("UTF-8");

    // Config UTF-8 Encoding.
    @Override
    public void configureMessageConverters(List<HttpMessageConverter<?>> converters) {
        StringHttpMessageConverter stringConverter = new StringHttpMessageConverter();
        stringConverter.setSupportedMediaTypes(Arrays.asList(new MediaType("text",
"plain", UTF8)));
        converters.add(stringConverter);


    }

    // Static Resource Config
    // equivalents for <mvc:resources/> tags
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {

registry.addResourceHandler("/css/**").addResourceLocations("/css/").setCachePeriod(315
56926);

registry.addResourceHandler("/img/**").addResourceLocations("/img/").setCachePeriod(315
56926);

registry.addResourceHandler("/js/**").addResourceLocations("/js/").setCachePeriod(31556
926);
    }

    // equivalent for <mvc:default-servlet-handler/> tag
    @Override
    public void configureDefaultServletHandling(DefaultServletHandlerConfigurer
configurer) {
        configurer.enable();
    }

}
```

Rest Endpoints

```
package com.sparql.webapp.constants;

public interface RestEndpoints {

      String REQUIRED_DOC_REST_ENDPOINT =
"http://localhost:8080/sparqlservice/rest/api/required-doc";
      String NUMBER_OF_DOC_REST_ENDPOINT =
"http://localhost:8080/sparqlservice/rest/api/number-of-doc";
      String VALUE_OF_SERVICE_REST_ENDPOINT =
"http://localhost:8080/sparqlservice/rest/api/value-of-service";

}
```

Controller

```
package com.sparql.webapp.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;

import com.sparql.webapp.beans.ApiBean;
import com.sparql.webapp.rest.consumer.SparqlRestConsumer;

@Controller
@EnableWebMvc
public class HomeController {

	@Autowired
	SparqlRestConsumer restClient;

	@RequestMapping("/")
	public ModelAndView home(){
		return new ModelAndView("home","apiBean", new ApiBean());
	}

	@RequestMapping(value="/getApiResult", method=RequestMethod.POST)
	public String getApiResult( @ModelAttribute("apiBean")ApiBean apiBean, ModelMap
model ){
		System.out.println("api name: "+apiBean.getApiName());
		apiBean.setApiResult("");

		if("requiredDocApi".equalsIgnoreCase(apiBean.getApiName())){
			apiBean.setApiResult(restClient.getRequiredDoc());

		}else if("numberOfDocApi".equalsIgnoreCase(apiBean.getApiName())){
			apiBean.setApiResult(restClient.getNumberOfDoc());

		}else if("valueOfServiceApi".equalsIgnoreCase(apiBean.getApiName())){
			apiBean.setApiResult(restClient.getValueOfService());
		}
		return "home";
	}

}
```

Rest Consumer

```
package com.sparql.webapp.rest.consumer;

import org.springframework.stereotype.Component;
import org.springframework.web.client.RestTemplate;

import com.sparql.webapp.constants.RestEndpoints;

@Component("restClient")
public class SparqlRestConsumer {
```

```
	public String getRequiredDoc(){

		RestTemplate restClient = new RestTemplate();
		String response =
restClient.getForObject(RestEndpoints.REQUIRED_DOC_REST_ENDPOINT, String.class);
		System.out.println("response : "+response);

		return response;
	}

	public String getNumberOfDoc(){

		RestTemplate restClient = new RestTemplate();
		String response =
restClient.getForObject(RestEndpoints.NUMBER_OF_DOC_REST_ENDPOINT, String.class);
		System.out.println("response : "+response);

		return response;
	}

	public String getValueOfService(){

		RestTemplate restClient = new RestTemplate();
		String response =
restClient.getForObject(RestEndpoints.VALUE_OF_SERVICE_REST_ENDPOINT, String.class);
		System.out.println("response : "+response);

		return response;
	}

}
```

Java Server Page

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>
	<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
	<title>sparql WebApp</title>

	<script src="${pageContext.request.contextPath}/js/jquery-3.4.1.min.js"></script>
	<script src="${pageContext.request.contextPath}/js/popper.min.js"></script>

    <link rel="stylesheet"
href="${pageContext.request.contextPath}/css/bootstrap.min.css" />
    <script src="${pageContext.request.contextPath}/js/bootstrap.min.js"></script>

	<link rel="stylesheet" href="${pageContext.request.contextPath}/css/style.css" >
	<link href="https://fonts.googleapis.com/css?family=Open+Sans" rel="stylesheet">
	<script type="text/javascript">
		function onApiClick(id) {
			if (id == "requiredDocApi") {
```

```
                        document.getElementById('requiredDocApi').style="background-
color:#218838 !important";
                        document.getElementById('numberOfDocApi').style="background-
color:#269abc !important";

      document.getElementById('valueOfServiceApi').style="background-
color:#269abc !important";

                        $("#errorMsg").hide();
                        $("#executeBtn").removeAttr("disabled");
                        $("#apiName").val("requiredDocApi");

                  } else if (id == "numberOfDocApi") {

                        document.getElementById('numberOfDocApi').style="background-
color:#218838 !important";
                        document.getElementById('requiredDocApi').style="background-
color:#269abc !important";

      document.getElementById('valueOfServiceApi').style="background-
color:#269abc !important";

                        $("#errorMsg").hide();
                        $("#executeBtn").removeAttr("disabled");
                        $("#apiName").val("numberOfDocApi");
                  } else {


      document.getElementById('valueOfServiceApi').style="background-
color:#218838 !important";
                        document.getElementById('numberOfDocApi').style="background-
color:#269abc !important";
                        document.getElementById('requiredDocApi').style="background-
color:#269abc !important";

                        $("#errorMsg").hide();
                        $("#executeBtn").removeAttr("disabled");
                        $("#apiName").val("valueOfServiceApi");
                  }
            }

            function invokeApi(){
                  $("#invokeApi").click();
            }

            function clearResult(){
                  $("#apiResult").val("");
            }

      </script>

</head>

<body style="background-color:#dae0e5 !important;">

      <div class="text-center" style="color: white !important;">
            <h1 style="line-height: 100px">SPARQL WEBAPP</h1>
      </div>

    <form:form method="POST"
                        action="${pageContext.request.contextPath}/getApiResult"
```

```
                        modelAttribute="apiBean">
            <div class="container">
                  <div class="row">
                        <div class="col-sm-6"
                              style="border-right: double; margin-top: 20px">
                              <h1>Select API</h1>

                              <a href="#" name="requiredDocApi" id="requiredDocApi"
onclick="onApiClick('requiredDocApi')">
                                    <b>1.Required Doc</b><br> Documents required to
perform a service
                              </a><br>
                              <a href="#" id="numberOfDocApi" name="numberOfDocApi"
onclick="onApiClick('numberOfDocApi')">
                                    <b>2.Number of Doc</b><br>How many documents
requires each public service as Imported to be processed
                              </a><br>
                              <a href="#" id="valueOfServiceApi"
name="valueOfServiceApi" onclick="onApiClick('valueOfServiceApi')">
                                    <b>3. Value of Service</b><br>What is the cost of
handling each described public service
                              </a><br>

                        </div>
                        <form:hidden path="apiName" id="apiName" />
                        <div class="col-sm-4">

                              <input type="submit" value="Execute" id="invokeApi"
                                    style="display: none;">
                              <button name="executeBtn" id="executeBtn" type="submit"
class="btn" onClick="invokeApi();" disabled="true">Execute</button>


                        </div>

                        <div class="col-sm-2" style="margin-top:175px;">

                              <span style="color:red;" id="errorMsg"
name="errorMsg"><b>Please Select an API</b></span>

                        </div>
                        <h1>Result</h1>
                        <a href="#" name="requiredDocApi" class="btn btn-primary"
id="requiredDocApi" onclick="clearResult()" style="margin:10px;background-color:
#337ab7 !important;">
                              Clear Result
                        </a>
                        <form:textarea class="form-control col-md-12 "
path="apiResult" disabled="true" />

                  </div>
            </div>
      </form:form>

      <script type="text/javascript">

          console.log("selected API : "+valueOfServiceApi);

          if($('#apiResult').val() != ""){
              console.log("data not empty");
                    var tmpData = JSON.parse($('#apiResult').val());
```

```
                    var formattedData = JSON.stringify(tmpData, null, '\t');
                    $('#apiResult').text(formattedData);
            }

        </script>

</body>

</html>
```

Web.xml (for spring mvc configuration)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID" version="3.1">
  <display-name>SparqlWebApp</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>

</web-app>
```

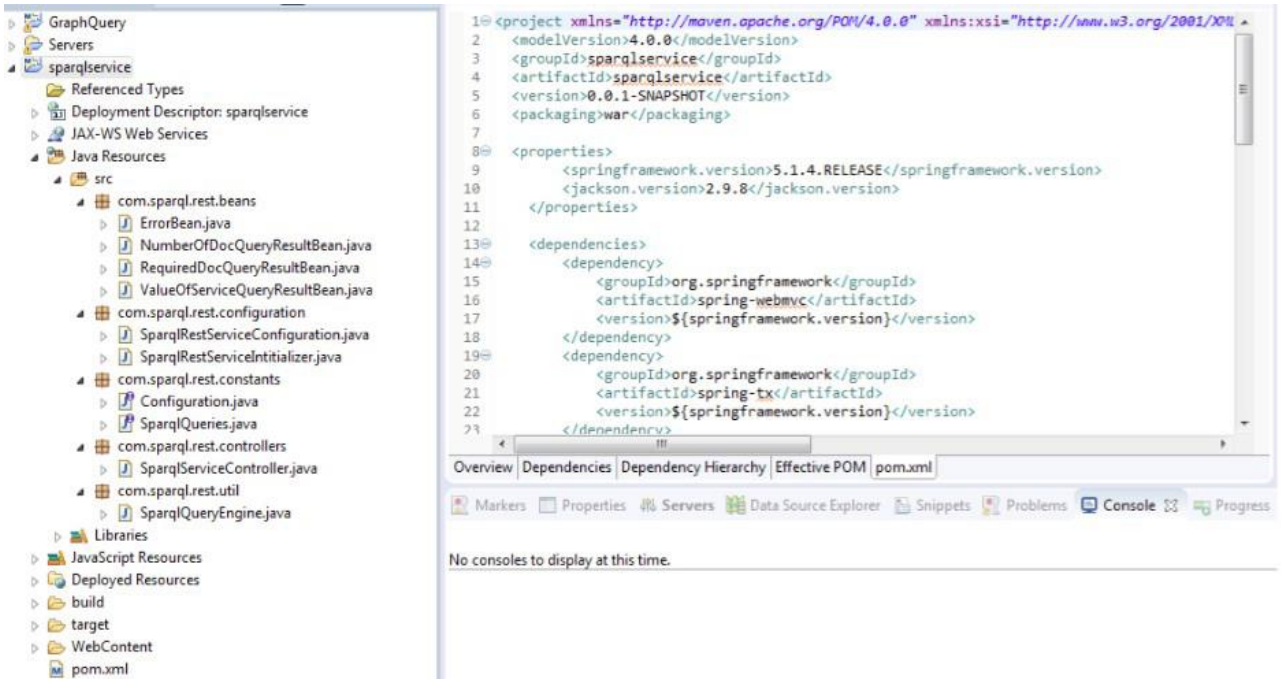
SPARQL Service Technical Specification

Τεχνολογίες που χρησιμοποιήθηκαν:

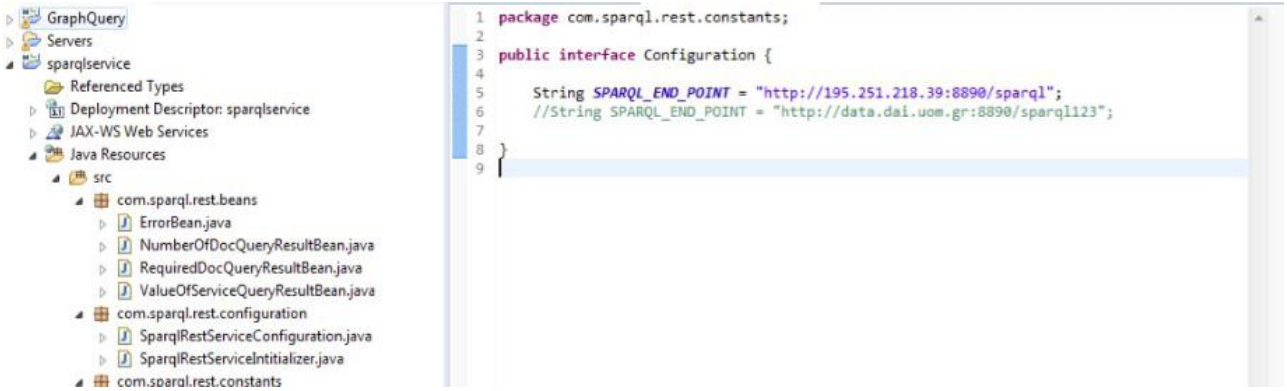
- Maven
- Spring MVC
- Spring REST
- Apache Jena

Περιγραφή

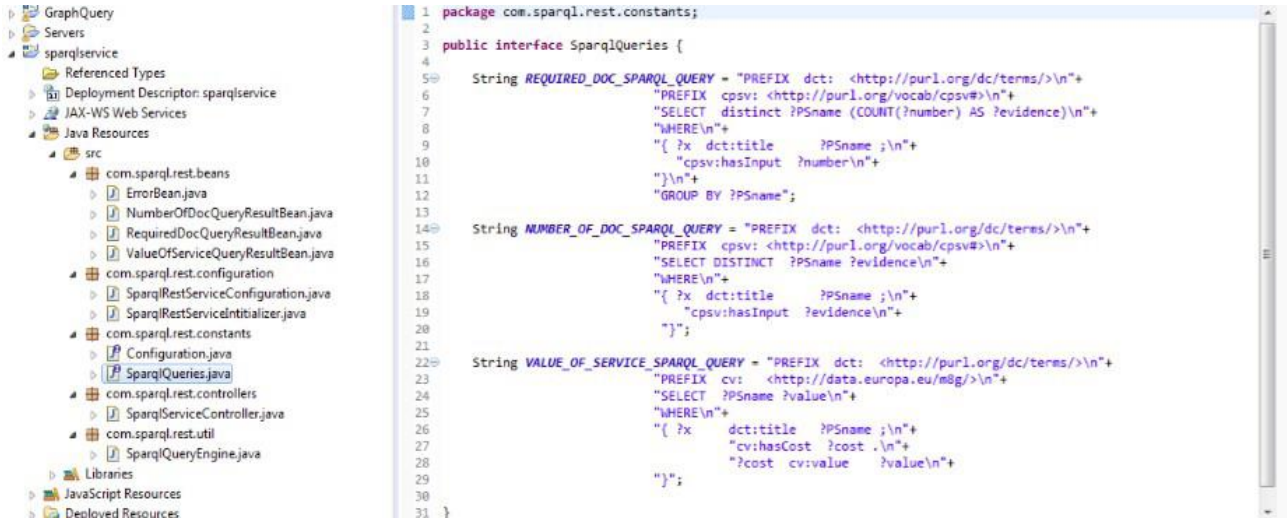
- Maven-based. Όλες οι εξαρτήσεις βρίσκονται στο Maven.
- Χρησιμοποιείται ένα αρχείο με το όνομα pom.xml



➔ Το SPARQL Endpoint διαμορφώνεται σε μία κλάση με όνομα Configuration.java

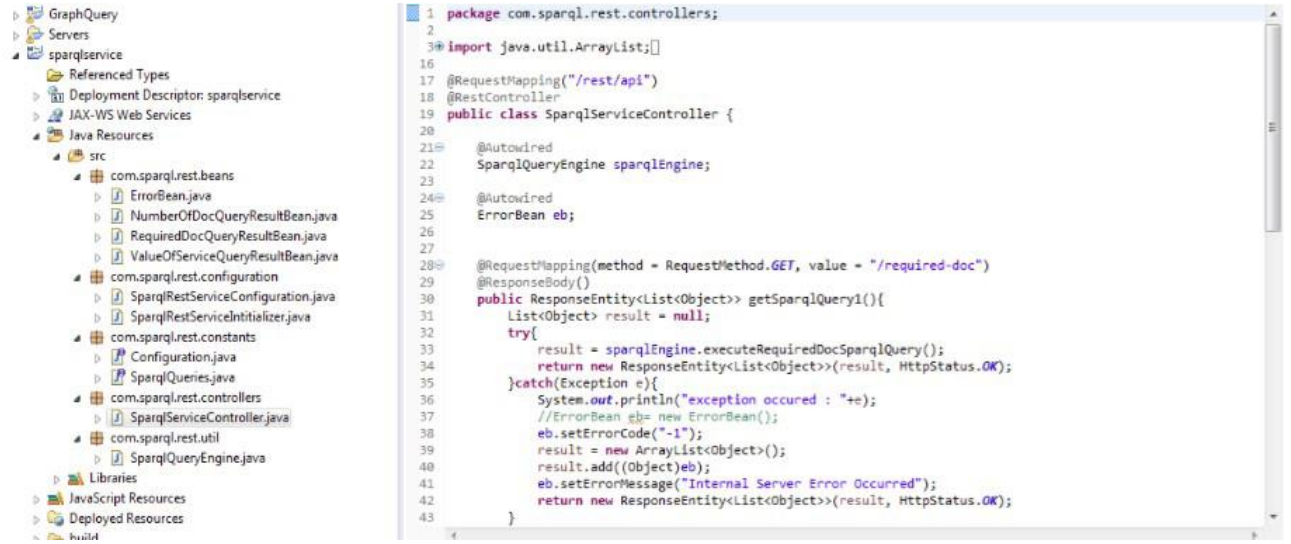


➔ Τα ερωτήματα SPARQL διαμορφώνονται σε μία κλάση με όνομα SparqlQueries.java



➔ Για να γίνει η κλήση στα δεδομένα για το κάθε ερώτημα χρησιμοποιείται η κλάση SparqlQueryEngine.java

➔ Χρησιμοποιείται Spring REST για να μετατραπεί η απάντηση (response) των ερωτημάτων sparql σε εφαρμογή με την κλάση SparqlServiceController.java



SPARQL Web Application Technical Specification

Τεχνολογίες που χρησιμοποιήθηκαν:

- Maven
- Spring MVC
- Spring REST
- Spring MVC, JSP (Front-End)
- Bootstrap (responsive)
- HTML, CSS, JQuery

Περιγραφή

➔ Maven-based. Όλες οι εξαρτήσεις βρίσκονται στο Maven.

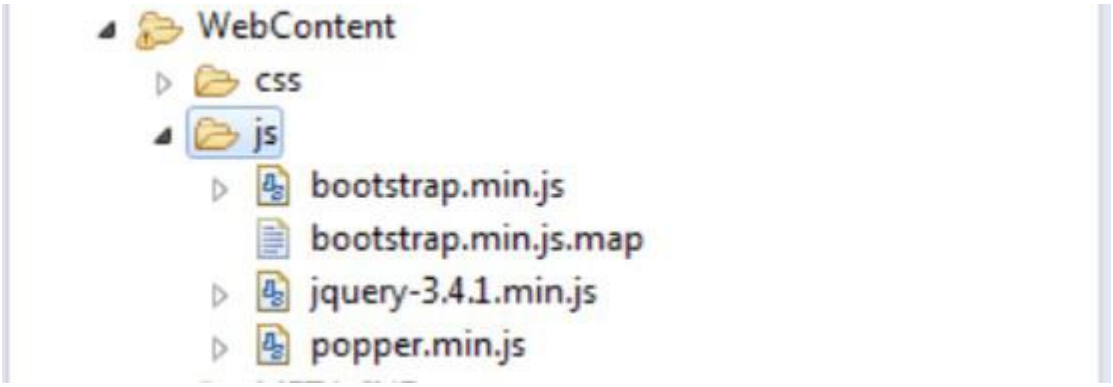
➔ Χρησιμοποιείται ένα αρχείο με το όνομα pom.xml

➔ Το αρχείο home.jsp περιέχει το απαραίτητο CSS και JS.

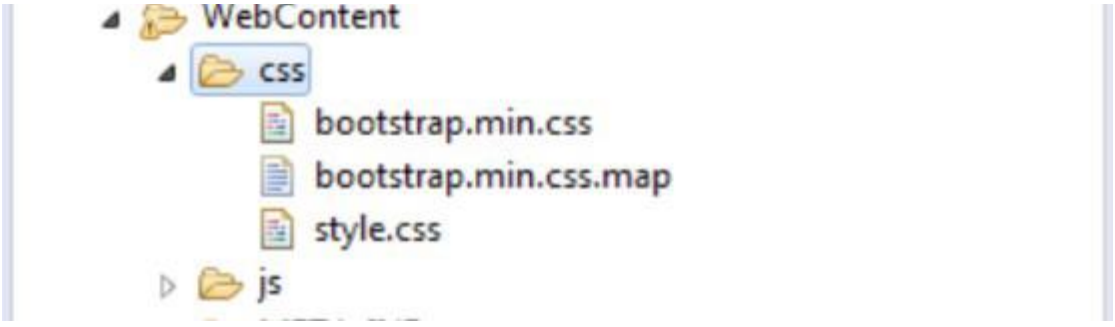
home.jsp

```
    pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>sparql WebApp</title>

    <script src="${pageContext.request.contextPath}/js/jquery-3.4.1.min.js"></script>
    <script src="${pageContext.request.contextPath}/js/popper.min.js"></script>

    <link rel="stylesheet" href="${pageContext.request.contextPath}/css/bootstrap.min.css" />
    <script src="${pageContext.request.contextPath}/js/bootstrap.min.js"></script>

    <link rel="stylesheet" href="${pageContext.request.contextPath}/css/style.css" >
    <link href="https://fonts.googleapis.com/css?family=Open+Sans" rel="stylesheet">
    <script type="text/javascript">
        function onApiClick(id) {
            if (id == "requiredDocApi") {
                document.getElementById('requiredDocApi').style="background-color:#218838 !important";
                document.getElementById('numberOfDocApi').style="background-color:#269abc !important";
                document.getElementById('valueOfServiceApi').style="background-color:#269abc !important";

                $("#errorMsg").hide();
                $("#executeBtn").removeAttr("disabled");
                $("#apiName").val("requiredDocApi");

            } else if (id == "numberOfDocApi") {
```

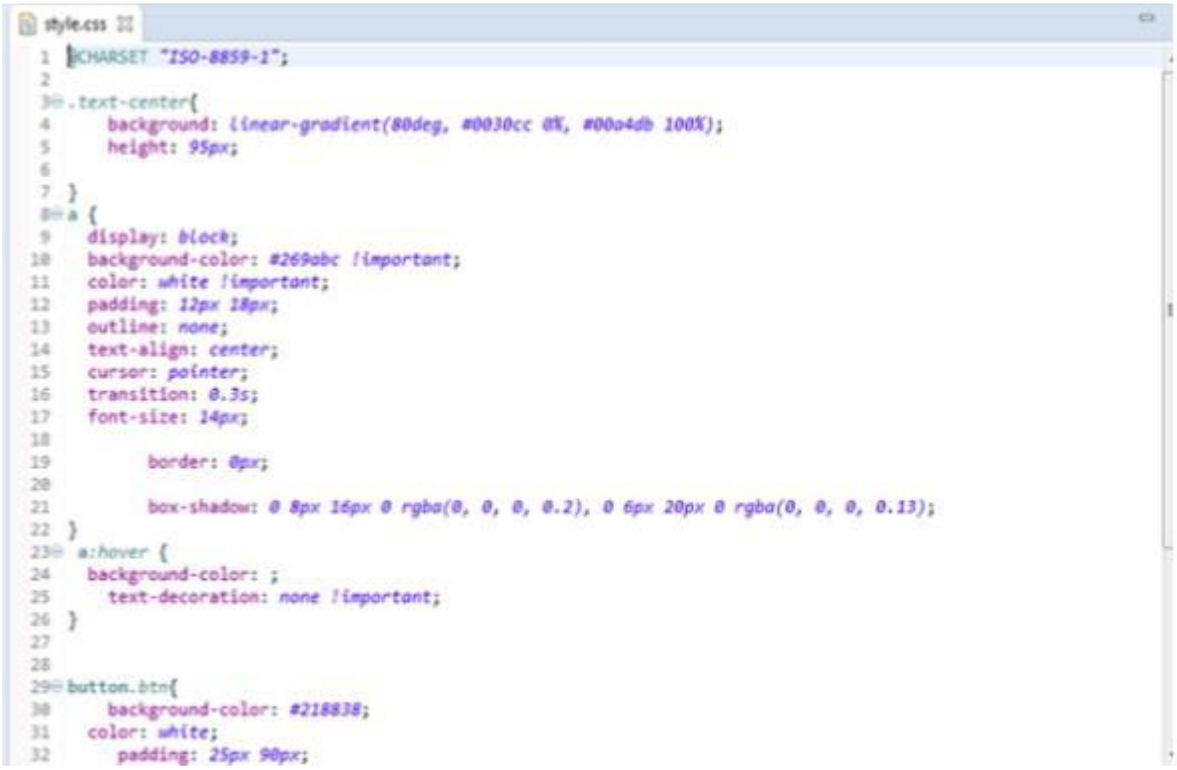
➔ Τα αρχεία JS βρίσκονται μέσα στον φάκελο WebContent στον υποκατάλογο js



➔ Τα αρχεία CSS βρίσκονται μέσα στον φάκελο WebContent στον υποκατάλογο css

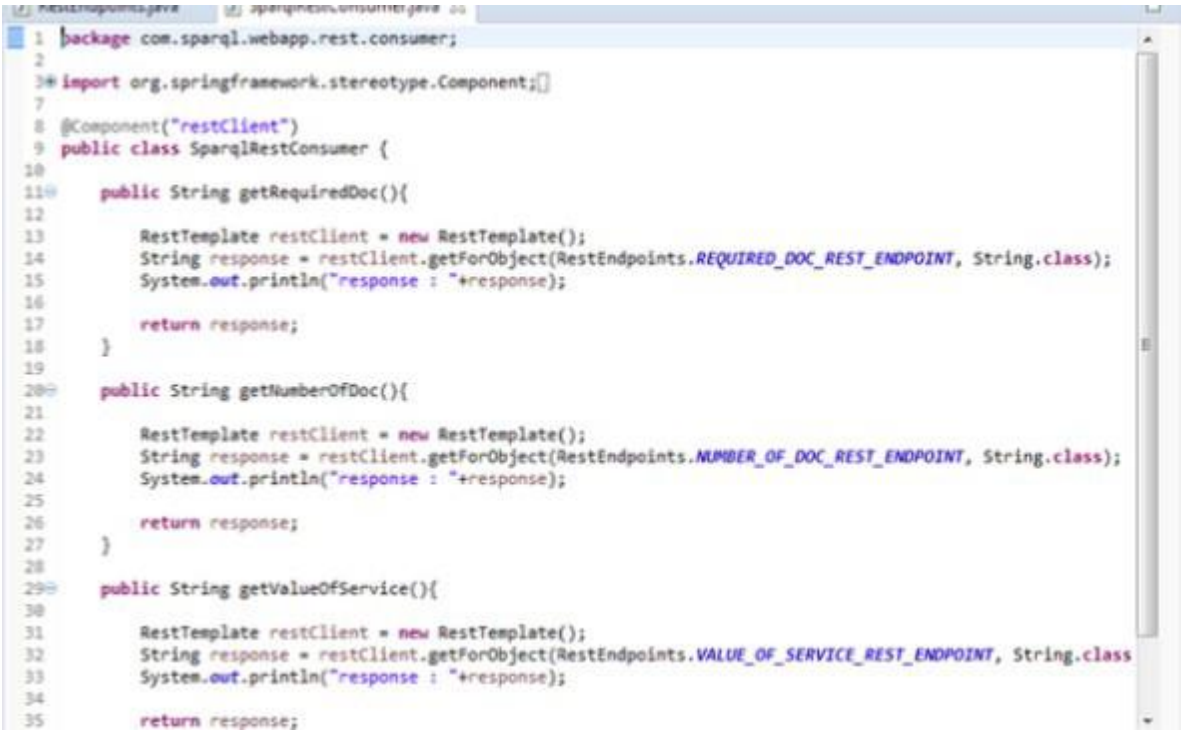


➔ Το αρχείο style.css δημιουργήθηκε για να προσαρμόσει το στυλ του web application



➔ Τα sparql endpoinds δηλώνονται στο αρχείο RestEndPoints.java

➔ Τα δεδομένα των endpoints καταναλώνονται στο αρχείο SparqlRestConsumer.java



```java
package com.sparql.webapp.rest.consumer;

import org.springframework.stereotype.Component;

@Component("restClient")
public class SparqlRestConsumer {

    public String getRequiredDoc(){

        RestTemplate restClient = new RestTemplate();
        String response = restClient.getForObject(RestEndpoints.REQUIRED_DOC_REST_ENDPOINT, String.class);
        System.out.println("response : "+response);

        return response;
    }

    public String getNumberOfDoc(){

        RestTemplate restClient = new RestTemplate();
        String response = restClient.getForObject(RestEndpoints.NUMBER_OF_DOC_REST_ENDPOINT, String.class);
        System.out.println("response : "+response);

        return response;
    }

    public String getValueOfService(){

        RestTemplate restClient = new RestTemplate();
        String response = restClient.getForObject(RestEndpoints.VALUE_OF_SERVICE_REST_ENDPOINT, String.class
        System.out.println("response : "+response);

        return response;
```

➔ Χρησιμοποιείται JSON.parse() ώστε τα δεδομένα να εμφανίζονται σε καλά δοδημένη μορφή στο αρχείο home.jsp

```
99
100                 <div class="col-sm-2" style="margin-top:175px;">
101
102                     <span style="color:red;" id="errorMsg" name="errorMsg"><b>Please Select an API</b></spa
103
104                 </div>
105                 <h1>Result</h1>
106                 <a href="#" name="requiredDocApi" class="btn btn-primary" id="requiredDocApi" onclick="clea
107                     Clear Result
108                 </a>
109                 <form:textarea class="form-control col-md-12 " path="apiResult" disabled="true" />
110
111             </div>
112         </div>
113     </form:form>
114
115     <script type="text/javascript">
116
117         console.log("selected API : "+valueOfServiceApi);
118
119         if($('#apiResult').val() != ""){
120             console.log("data not empty");
121             var tmpData = JSON.parse($('#apiResult').val());
122             var formattedData = JSON.stringify(tmpData, null, '\t');
123             $('#apiResult').text(formattedData);
124         }
125
126     </script>
127
128 </body>
129
130 </html>
```