Σχολή Θετικών Επιστημών και Τεχνολογίας

Μεταπτυχιακή Εξειδίκευση στα Πληροφοριακά Συστήματα

Διπλωματική Εργασία

Δημοσίευση διοικητικών διαδικασιών του Δημοσίου ως Ανοικτά Συνδεδεμένα Δεδομένα (Linked Open Data): Η Περίπτωση του Εθνικού Μητρώου Διοικητικών Διαδικασιών - Mitos.gov.gr

Κωνσταντίνος Ι. Ντούτσος-Οικονόμου

Επιβλέπων καθηγητής: Ευθύμιος Ταμπούρης

Αθήνα, Σεπτέμβριος 2023

Δημοσίευση διοικητικών διαδικασιών του Δημοσίου ως Ανοικτά Συνδεδεμένα Δεδομένα (Linked Open Data):

Η Περίπτωση του Εθνικού Μητρώου Διοικητικών Διαδικασιών - Mitos.gov.gr

Κωνσταντίνος Ι. Ντούτσος-Οικονόμου

Επιτροπή Επίβλεψης Πτυχιακής / Διπλωματικής Εργασίας

Επιβλέπων Καθηγητής:

Ταμπούρης Ευθύμιος

Καθηγητής του Πανεπιστημίου Μακεδονίας

Συν-Επιβλέπουσα Καθηγήτρια:

Χριστοπούλου Ελένη

Μέλος ΣΕΠ του Ελληνικού Ανοικτού Πανεπιστημίου

Μέλος ΕΔΙΠ του τμήματος Πληροφορικής του Ιονίου Πανεπιστημίου

Αθήνα, Σεπτέμβριος 2023

«Στην οικογένεια μου, μα πάνω από όλους και όλα σε εμένα που ποτέ δεν σταμάτησα ποτέ να κυνηγάω πάντα κάτι παραπάνω.»

Περίληψη

Η διάθεση των διαδικασιών του δημοσίου ως Ανοικτά Συνδεδεμένα Δεδομένα είναι κρίσιμη για την προαγωγή της διαφάνειας, την ενίσχυση της αποτελεσματικότητας και την επέκταση της δημοκρατικής συμμετοχής, δημιουργώντας παράλληλα νέες ευκαιρίες για την καινοτομία.

Η παρούσα διπλωματική εργασία με τίτλο "Δημοσίευση Διοικητικών Διαδικασιών του Δημοσίου ως Ανοικτά Συνδεδεμένα Δεδομένα" διερευνά την μετατροπή και δημοσίευση διαδικασιών του Εθνικού Μητρώου Διοικητικών Διαδικασιών (ΕΜΔΔ) "Mitos" σε ανοικτά συνδεδεμένα δεδομένα κατά προτυποποίηση & μοντελοποίηση CPSV-AP (Core Public Service Vocabulary Application Profile). Το ΕΜΔΔ παρουσιάζει δομημένες περιγραφές δημοσίων υπηρεσιών, με πληροφορίες για τα βήματα, την νομοθεσία, το κόστος, τις προϋποθέσεις και άλλα στοιχεία, καθώς και δικαιολογητικά, τα οποία είναι διαθέσιμα μέσω API ή της Βιβλιοθήκης Σημασιολογικών Δεδομένων.

Η μελέτη επικεντρώνεται στην περίπτωση του Εθνικού Μητρώου Διοικητικών Διαδικασιών, το οποίο αποτελεί μια πλούσια πηγή δεδομένων, η οποία μέχρι στιγμής παραμένει αναξιοποίητη. Με τη χρήση σύγχρονων μεθοδολογιών και προσαρμογών από τη βιβλιογραφία, αναδεικνύουμε τις διαδικασίες και τα βήματα, τα οποία είναι απαραίτητα για την αποτελεσματική δημοσίευση αυτών των δεδομένων.

Λέξεις – Κλειδιά

Δημόσια Ανοικτά Δεδομένα, Επεξεργασία Ανοικτών Συνδεδεμένων Δεδομένων, Εθνικό Μητρώο Διαδικασιών Δημοσίου (ΕΜΔΔ), mitos.gov.gr, Core Public Service Vocabulary(CPSV), Core Public Service Vocabulary – Application Profile (CPSV-AP)

Public sector processes publication as linked open data.

The National Registry of Public Services case study.

Konstantinos Ntoutsos-Oikonomou

Abstract

The provision of public procedures as Open Linked Data is critical for promoting transparency, enhancing efficiency, and extending democratic participation, while simultaneously creating new opportunities for innovation.

The present thesis titled "Publishing Public Administrative Procedures as Open Linked Data" investigates the conversion and publication of procedures from the National Registry of Administrative Procedures (NRAP) "Mitos" into open linked data according to CPSV-AP standardization & modeling. The NRAP presents structured descriptions of public services, with information on the steps, legislation, cost, prerequisites, and various other elements, as well as documentation, which are available through API or Semantic Data Library.

The study, therefore, focuses on the case of the National Registry of Administrative Procedures, which is a rich source of data that has so far remained untapped. Using contemporary methodologies and adaptations from literature, we highlight the procedures and steps necessary for the effective publication of these data.

Keywords

Linked Open Data, National, ΕΜΔ, National Registry of Administrative Public Services, mitos.gov.gr, Core Public Service Vocabulary (CPSV), Core Public Service Vocabulary – Application Profile (CPSV-AP)

Περιεχόμενα

Κατάλογος Εικόνων / Σχημάτων

Συντομογραφίες & Ακρωνύμια

Ακολουθούν κάποια παραδείγματα:

ΔΕ	Διπλωματική Εργασία
ΕΑΠ	Ελληνικό Ανοικτό Πανεπιστήμιο
ΘΕ	Θεματική Ενότητα
ΠΕ	Πτυχιακή Εργασία
ΠΣ	Πρόγραμμα Σπουδών
ΣΥΝ	Συντονιστής/Συντονίστρια
ΕΜΔΔ	Εθνικό Μητρώο Διοικητικών Διαδικασιών
LOD	Linked Open Data
CMS	Content Management System
DB	Database
RDF	Resource Description Framework
W3C	World Wide Web Consortium
CPSV	Core Public Service Vocabulary
CPSV-AP	Core Public Service Vocabulary – Application Profile
REST	Representational State Transfer (Protocol)
SMW	Semantic Media Wiki
WSL	Windows Subsystem (for) Linux

1. Εισαγωγή

1.1 Περιγραφή Προβλήματος

Το Εθνικό Μητρώο Διοικητικών Διαδικασιών «ΜΙΤΟΣ» προσφέρει προς ανάγνωση μία πληθώρα πληροφοριών οι οποίες είναι αναρτημένες σε ένα Σύστημα Διαχείρισης Περιεχόμενου (Wiki - CMS) τροποποιημένο για να μπορεί να υποδέχεται Σημασιολογικά Δεδομένα. Οι πληροφορίες αυτές αφορούν τις διαδικασίες του δημόσιου τομέα και μεταξύ άλλων περιέχουν στοιχεία όπως τρόπος, χρόνος και κόστος υλοποίησης των διαδικασιών αυτών.

Παρόλο την αυξανόμενη αναγνώριση των πλεονεκτημάτων των Ανοικτών Συνδεδεμένων Δεδομένων για βελτιστοποίηση της διαφάνειας αλλά και της αποτελεσματικότητας των δημοσίων διαδικασιών, η μετατροπή τους σε LOD (Linked Open Data) παραμένει μέχρι και σήμερα μια σημαντική πρόκληση.

Σκοπός μας είναι να μετατρέψουμε τα δεδομένα αυτά που προσφέρονται στο ΜΙΤΟΣ σε Ανοικτά Συνδεδεμένα, καθώς και να κατανοήσουμε τα πλεονεκτήματα αυτών αλλά και τις δυσκολίες που προκύπτουν.

1.2 Αντικείμενο και στόχοι της μελέτης

Η παρούσα διπλωματική εργασία αποσκοπεί στην μετατροπή των δεδομένων αυτών από το Εθνικό Μητρώο Διοικητικών Διαδικασιών «ΜΙΤΟΣ» σε Ανοικτά Συνδεδεμένα Δεδομένα μοντελοποιημένα κατά CPSV-AP (Core Public Service Vocabulary Application Profile).

Με γνώμονα την αξιοποίηση όλων των διαθέσιμων εργαλείων και υπηρεσιών με σκοπό την βελτιστοποίηση της ποιότητας των δεδομένων αλλά και την ευχρηστία των δεδομένων τόσο για τους κυβερνητικούς οργανισμούς όσο και για τους πολίτες, οι επιμέρους στόχοι μας είναι η συλλογή των δεδομένων αυτών μέσω του API, η μοντελοποίησης τους, η επεξεργασίας τους και η δημοσίευσης τους στο Διαδίκτυο και τον Σημαντικό ιστό.

1.3 Περιεχόμενα της μελέτης

Το δεύτερο κεφάλαιο πραγματεύεται το υλικό κατανόησης, ουσιαστικά αναπτύσσονται βασικές βιβλιογραφικές έννοιες προκειμένου να μπορεί να κατανοήσει ο αναγνώστης εις βάθος το θεωρητικό υπόβαθρο τόσο των τεχνολογιών και των μεθόδων που πρόκειται να χρησιμοποιηθούν, όσο και των εργαλείων.

Το τρίτο κεφάλαιο περιέχει την ανάλυση της αρχιτεκτονικής. Παρατίθεται η δομή, ο τρόπος και οι μέθοδοι με τις οποίες επιτελούμε το τεχνικό μέρος της διπλωματικής εργασίας. Σε αυτό το κεφάλαιο συναντάμε την ανάπτυξη των τρόπων μέσω των οποίων εξάγουμε τα δεδομένα, τα αποθηκεύουμε, τα μετασχηματίζουμε και στην συνέχεια τα τοποθετούμε στο RDF Triple store προκειμένου να δημοσιευθούν.

Στο τέταρτο κεφάλαιο γίνεται η τεχνική ανάπτυξη του έργου. Αναφερόμαστε τεχνικά στις διαδικασίες, τους τρόπους, μελετάμε τα blocks του κώδικα που έχουμε χρησιμοποιήσει και βλέπουμε αναλυτικά τα βήματα της λειτουργίας της προτεινόμενης αρχιτεκτονικής σε δράση.

Στο πέμπτο και τελευταίο κεφάλαιο γίνεται αναφορά στα συμπεράσματα που προέκυψαν κατά την διάρκεια εκπόνησης της διπλωματικής εργασίας, αλλά και τις πιθανές ευκαιρίες που υπάρχουν προς εκμετάλλευση σε μελλοντικές έρευνες.

1.4 Μεθοδολογία

Η μεθοδολογία μετατροπής δεδομένων σε ανοικτά συνδεδεμένα δεδομένα υπάρχει και είναι διαθέσιμη στην παγκόσμια βιβλιογραφία εδώ και αρκετά χρόνια. Η πρόσφατη βιβλιογραφία, όντας εκτενέστερη και αναλυτικότερη, επεκτείνεται σε πιο συγκεκριμένες κατευθύνσεις και εντάσσει υπό-περιπτώσεις θεματικών, οι οποίες αναφέρονται στο πεδίο.

Με τις δημοσιεύσεις *'Συνδεδεμένα Δεδομένα: Μια ευκαιρία για τις ελληνικές Βιβλιοθήκες' (2015, Σπανός)* και *'On using CPSV-AP to publish public service descriptions as linked open data' (2018, Γέροντας, Ταμπούρης, Λαζοπούλου, Ταραμπάνης)* έχουμε μια πιο ξεκάθαρη εικόνα τόσο στο τεχνικό κομμάτι της υλοποίησης όσο και για τη σημασία της θεωρητικής συζήτησης γύρω από τη θεματική.

Μέσα από μια σειρά 7 προγραμματισμένων βημάτων μπορούμε να δημοσιεύσουμε τις διαδικασίες που επιθυμούμε σε ανοικτά συνδεδεμένα δεδομένα.

Τα 7 βήματα αυτά διαρθρώνονται σε:

- Κατανόηση του συγκείμενου και προετοιμασία των διαδικασιών προς επιλογή
- Σύνθεση των πολιτικών σχεδιασμού URI για τις διαδικασίες
- Μοντελοποίηση των δεδομένων
- Παραγωγή Συνδεδεμένων Δεδομένων
- Επαλήθευση των δεδομένων αυτών
- Δημοσίευση των διαδικασιών σαν LOD
- Αναγνώριση και Αξιοποίηση ευκαιριών για εκμετάλλευση των δεδομένων

Ως προς την συλλογή των δεδομένων χρησιμοποιούμε την προγραμματιστική διεπαφή (API) του ΜΙΤΟΣ (https://api.mitos.gov.gr/info/#/)

Ουσιαστικά αυτή η μεθοδολογία μας επιτρέπει να εξάγουμε τα δεδομένα από την βάση και αυτοματοποιώντας την διαδικασία μπορούμε να έχουμε πάντα διαθέσιμα τα πιο σύγχρονα και ενημερωμένα δεδομένα από το Εθνικό Μητρώων Διαδικασιών.

Σκοπός μας είναι να έχουμε μία πλήρης λειτουργική μετασχηματιστική ροή εργασιών (transformation workflow) από το σημείο 0, το σημείο παραγωγής των δεδομένων αυτών μέχρι το τελικό, το οποίο είναι το σημείο κατανάλωσης και αξιοποίησης τους.

Εικόνα 1. Γενικό Σχήμα διαδικασίας της ροής των δεδομένων από την αρχή έως το τέλος. Η πλήρη ανάλυση του θα γίνει σε μεταγενέστερο κεφάλαιο.

Για να εξαχθούν οι διαδικασίες από την βάση δεδομένων του μέσω API, πραγματοποιούνται τα 2 ακόλουθα GET Requests:

1) Την πρώτη φορά εξάγουμε τα δεδομένα για Χ αριθμών διαδικασιών (π.χ Τα μοναδικά αναγνωριστικά – id – για 50 διαδικασίες)

Εμφάνιση όλων των διαδικασιών

```
curl -X GET "API_BASE_URL/v1/services"
```

2) Ενώ την επόμενη φορά εκτελούμε την κλήση με τις παραμέτρους του ID της διαδικασίας.

Εμφάνιση συγκεκριμένης διαδικασίας βάση του 6ψήφιου ID της

```
curl -X GET "API_BASE_URL/v1/services/id/{id}"
```

Αφού εξαχθούν τα αποτελέσματα σκοπός μας είναι κατά τα πρότυπα μετατροπής των δεδομένων σε LOD να κατανοήσουμε το συγκείμενο, να ετοιμάσουμε και να επιλέξουμε τις διοικητικές διαδικασίες που θέλουμε να μοντελοποιήσουμε.

Στην συγκεκριμένη ενότητα πρέπει να είμαστε σε θέση να κατανοήσουμε όλες τις λεπτομέρειες εκείνες που διέπουν τις διαδικασίες.

Η προσεκτική μελέτη όλων των πεδίων είναι ιδιαίτερα σημαντική για την πλήρη κατανόηση της λειτουργίας τους αλλά και του σκοπού/ρόλου τους. Κατανοώντας εις βάθος τις λειτουργίες αυτές θα είμαστε σε θέση να κρίνουμε κατά την επιλογή των δεδομένων ποιες είναι οι επιθυμητές αυτές πληροφορίες που θέλουμε να αξιοποιήσουμε.

Η επιλογή των διαδικασιών γίνεται με βάση μια πληθώρα κριτηρίων. Συνήθως τα βασικότερα κριτήρια στα οποία στρέφουμε την προσοχή μας είναι **α) η δημοτικότητα μιας διαδικασίας** και **β) η διαθεσιμότητα καλά σχεδιασμένων, οργανωμένων πληροφοριών και περιγραφών της διαδικασίας αυτής.** Αυτά είναι σημαντικά προκειμένου να προκύψουν αποτελέσματα υψηλής ποιότητας προς δημοσίευση, τα οποία με τη σειρά τους να μπορούν να αξιοποιηθούν καταλλήλως.

Στην συνέχεια περνάμε στην σύνθεση των πολιτικών σχεδιασμού URIs για τις διαδικασίες.

Μας απασχολούν τα πρότυπα τα οποία σχετίζονται με το μοντέλο CPSV-AP καθώς διαμορφώνουμε τα δεδομένα μας σύμφωνα με αυτό.
Με βάση τις γενικότερες οδηγίες στην Ευρωπαϊκή Ένωση και τους εταίρους της το μοντέλο σχεδίασης URI που ακολουθείτε έχει την εξής μορφή:

http:// {domain}/ {type}/ {concept}/ {reference}

Προσπαθώντας να αναλύσουμε την σχετική αρχιτεκτονική για την δομή των persistent URIs θα πρέπει ουσιαστικά να επιστρατεύσουμε όλες τις πληροφορίες που μας δίνονται βάση και της βιβλιογραφία αλλά και των αντίστοιχων εφαρμογών που έχουν επιτευχθεί κατά καιρούς και σε άλλες χώρες μέλη.

Βάση όλων αυτών των πληροφοριών αλλά και κοινών παραδοχών και παροτρύνσεων είναι γνωστό ότι μια προτεινόμενη μορφή είναι η ακόλουθη (αναφορά σε διπλωματική Λαζοπ.):

Persistent URI Format:

http(s):// { HOST } ή { DOMAIN } / { type } / { concept } / { reference }

Αναφέρουμε ένα παράδειγμα που σχετίζεται με την υποχρεωτική ιδιότητα της κλάσης Public Service (identifier).

Persistent URI:

http(s)://eap.gr:8990/PublicServices/id/ps/ps0001.

Στο κομμάτι του {HOST} ή {DOMAIN} βλέπουμε ότι είναι το endpoint στο οποίο έχει στηθεί το αποθετήριο τριπλετών μας (RDF Store) ενώ το Public Services υποδηλώνει ότι τα URIs αυτά απευθύνονται σε Διαδικασίες (Διοικητικές στην περίπτωση μας) του Δημοσίου.

Το **type: id** παρατηρούμε ότι εκφράζει τον τύπο του πόρου (resource) ο οποίος μεταφράζεται στον μοναδικό αναγνωριστικό (identifier) που αφορά την διαδικασία.

Το **concept: ps** αναφέρεται στον τύπο της έννοιας στον πραγματικό κόσμο, στην προκειμένη περίπτωση της διαδικασίας. Ολογράφως θα πρέπει να αποτυπωθεί ως Public Service ξανά, ωστόσο για λόγους χρηστικότητας, εξοικονόμησης χώρου και χρόνου επιλέγεται μία συντομογραφία που αποτελείται από τα αρχικά γράμματα των 2 λέξεων, “Public Service”.

Στο τελευταίο κομμάτι του URI το **reference: ps0001** αναφέρεται η κάθε δημόσια διαδικασία. Ουσιαστικά αποτελεί μια μονοσήμαντη και μοναδική περιγραφή της διαδικασίας

Σχετικά με τη μοντελοποίηση των δεδομένων, το πρώτο πράγμα που πρέπει να αναγνωρίσουμε είναι σε τι κατάσταση βρίσκονται.

Αν τα δεδομένα μας είναι «RAW» (πρωτογενή), δηλαδή δεδομένα τα οποία δεν έχουν υποστεί καμία επεξεργασία, δεν είναι δομημένα και οργανωμένα, τότε η δουλειά μας σε αυτό το στάδιο είναι ελαφρώς διαφορετική.

Θα πρέπει ξεκινώντας από το 0 να δομήσουμε τα δεδομένα αυτά κατά τα πρότυπα του CPSV-AP, κάτι το οποίο έχει πολύ μεγαλύτερο χρονικό ορίζοντα και βάθος καθώς καλούμαστε να παράγουμε και θεμελιώδη δουλειά προκειμένου να συνεχίσουμε την μοντελοποίηση.

Αν τα δεδομένα μας είναι δομημένα, τότε αυτό που πρέπει να κάνουμε είναι να αντιστοιχηθούν στις ιδιότητες και τα concepts του CPSV-AP.

Σε κάθε περίπτωση ωστόσο ο κύριος όγκος της δουλειάς είναι η εξαγωγή meta-πληροφοριών και αντιστοίχιση τους στις ιδιότητες του CPSV-AP V3.1

Η εξαγωγή των πληροφοριών αυτών μπορεί να βασίζεται είτε σε αυτόματα workflows (σαν την περίπτωση που μελετάμε στην παρούσα διπλωματική) είτε σε ημι-αυτόματα που περιλαμβάνουν ενδιάμεσα βήματα, υπεύθυνα για την επαλήθευση της εγκυρότητας αυτών λόγω σημασιολογικών δυσκολιών των διαδικασιών. Χρησιμοποιούνται διάφορα εργαλεία για την μοντελοποίηση των δεδομένων αυτών, αναφερόμενα ως mapping tools.

Το επόμενο βήμα σχετίζεται με την παραγωγή συνδεδεμένων δεδομένων και την διασύνδεση τους.

Σε αυτό το σημείο, οι διαδικασίες οι οποίες είναι συμμορφούμενες κατά CPSV-AP μετατρέπονται σε συνδεδεμένα δεδομένα. Σαν τελικό σκοπό έχουμε πάντα την δυνατότητα να διασυνδεθούν με άλλα data-set στον σημαντικό ιστό.

Στη συγκεκριμένη περίπτωση θα μελετήσουμε τη μετατροπή και τη μοντελοποίηση των δεδομένων με τη χρήση εργαλείων αλλά και διάφορων προγραμματιστικών τρόπων (custom scripts).

Τα παραγόμενα δεδομένα συνήθως τα εξάγουμε σε αρχεία με μορφές RDF/Turtle. Αυτές οι μορφές είναι κατανοητές και από ανθρώπους και από υπολογιστές και συνήθως είναι και οι πιο χρήσιμες. Ωστόσο μπορούμε να εξάγουμε τα δεδομένα σε πληθώρα μορφών όπως CSV,JSON,XML κ.α

Ακολούθως είναι η επαλήθευση των εξαγόμενων αποτελεσμάτων.

Σε αυτό το σημείο ελέγχεται η εγκυρότητα των δεδομένων (σε συντακτικό επίπεδο) καθώς και αν τυχόν υπάρχει παραβίαση των προτύπων του CPSV-AP.

Μετά την επαλήθευση για την εγκυρότητα των εξαγόμενων αποτελεσμάτων τα linked data set (είτε το RDF αρχείο, είτε τα Turtles) μεταμορφώνονται σε ένα RDF Store και γίνεται δημόσια γνωστό σαν RDF γράφος μέσω ενός SPARQL endpoint.

2. Βιβλιογραφική Ανασκόπηση

2.1 Εισαγωγή

Η μετατροπή των διαδικασιών του Δημοσίου σε Ανοικτά Συνδεδεμένα Δεδομένα (ΑΣΔ) κατά βάση προτύπου CPSV-AP αλλά και η δημοσίευση τους στον Παγκόσμιο Ιστό χρήζει μελέτης της υπάρχουσας βιβλιογραφίας.

Για την κατανόηση της σημασίας και της χρηστικότητας αυτών θα πρέπει να εξερευνήσουμε βιβλιογραφικά, τα Ανοικτά Συνδεδεμένα Δεδομένα, το Πρωτόκολλο CPSV-AP και τους επιμέρους ορισμούς και ορολογίες που διέπουν τις έννοιες αυτές.

2.2 Η Ηλεκτρονική Διακυβέρνηση στη Δημόσια Διοίκηση

Η επιτακτικότητα της στέρεης και λειτουργικής Ηλεκτρονικής Διακυβέρνησης στην Δημόσια Διοίκηση είναι αναγκαία. Τα ξεκάθαρα πλέον πλεονεκτήματα, όπως:

- Ενδυνάμωση του περιβάλλοντος ελέγχου
- Περαιτέρω διεύρυνση της διαφάνειας στη Δημόσια Διοίκηση
- Καταπολέμηση της διαφθοράς
- Αδιαμφισβήτητη βελτιστοποίηση της διεκπεραίωσης διαδικασιών

είναι ορισμένα από τα σημεία-κλειδιά στον μετασχηματισμό του συνόλου της παροχής υπηρεσιών από το κράτος και στη βελτιστοποίηση τους, ώστε αυτό (σ.σ κράτος) να είναι εναρμονισμένο με όλες τις ευρωπαϊκές οδηγίες προκειμένου να μπορείτε και να θεωρείται ισάξιο σε ποιότητα παροχής υπηρεσιών κράτος-μέλος της Ε.Ε.

Το 2012 η Δημόσια Διαβούλευση του Σχεδίου Δράσης για την Ανοικτή Διακυβέρνηση (Open Government Partnership)[1] έδωσε την δυνατότητα προκειμένου να καταστεί ρεαλιστική, η στοχοθέτηση και η νομοθέτηση με τρόπο τέτοιο ώστε να δημιουργηθεί ένα σύγχρονο πλαίσιο για την εύρεση, διαχείριση και επεξεργασία δεδομένων. Μια σύγχρονη κοινωνική επιταγή και ένας στόχος όλων των χωρών της Ε.Ε.

Δημιουργήθηκε έτσι το πρόγραμμα της Δι@υγειας[2], το GeoData[3], η κυβερνητική Πύλη ΕΡΜΗΣ[4] και ένα σύνολο από διάφορες πλατφόρμες σε Δημοτικό και Περιφερειακό επίπεδο, οι οποίες αποτέλεσαν σημαντικές πρωτοβουλίες και βήματα προκειμένου να δημιουργηθεί εν τέλει το GOV.gr, σαν ένα οικοσύστημα, το οποίο αποτελεί τη Βίβλο του μοντέρνου ψηφιακού μετασχηματισμού, εμπεριέχοντας πληθώρα υπηρεσιών, πληροφοριών και δεδομένων προς αξιοποίηση από τους πολίτες, άλλους οργανισμούς αλλά και επιχειρήσεις.

1. http://www.opengov.gr/ogp/?p=9 - Προσπελάστηκε Ιανουάριο του 2023
2. https://diavgeia.gov.gr/ - Προσπελάστηκε Ιανουάριο του 2023
3. https://www.capital.gr/epikairotita/1121638/upeka-eleutheri-diathesi-geoxorikon-dedomenon-meso-tou-geodata-gov-gr/ - Προσπελάστηκε Ιανουάριο του 2023
4. https://opengov.gr/ogp/?p=52 – Προσπελάστηκε Ιανουάριο του 2023

2.3 Σημασιολογικός Ιστός

Ο Σημασιολογικός Ιστός, ένα όραμα που προτάθηκε από τον Tim Berners-Lee, τον εφευρέτη του Παγκόσμιου Ιστού, στοχεύει στη δημιουργία ενός Ιστού δεδομένων που μπορούν να υποβληθούν σε επεξεργασία άμεσα και έμμεσα από μηχανές (Berners-Lee, Hendler & Lassila, 2001). Επεκτείνει τον υπάρχοντα Ιστό, όπου τα μη δομημένα έγγραφα διασυνδέονται μέσω υπερσυνδέσμων, σε έναν Ιστό, όπου τα δομημένα δεδομένα μπορούν να διασυνδεθούν και να γίνουν πιο χρήσιμα και ουσιαστικά.

Το κλειδί για τον Σημασιολογικό Ιστό είναι η χρήση προτύπων, όπως το Πλαίσιο Περιγραφής Πόρων (RDF), η Γλώσσα Οντολογίας Ιστού (OWL), το Πρωτόκολλο SPARQL και η Γλώσσα Ερωτήσεων RDF (SPARQL). Το RDF χρησιμεύει ως μια γενική μέθοδος για την αποσύνθεση οποιουδήποτε συνόλου δεδομένων σε τριάδες, τα οποία μπορούν να διασυνδεθούν και να δημοσιευτούν στον Ιστό. Το OWL χρησιμοποιείται για να εκφράσει σύνθετες σχέσεις μεταξύ αυτών των στοιχείων δεδομένων και να συνάγει νέα δεδομένα, ενώ το SPARQL παρέχει έναν ισχυρό μηχανισμό αναζήτησης για την εξαγωγή σημαντικών πληροφοριών από τον ιστό δεδομένων (Antoniou & van Harmelen, 2004).

Στον Σημασιολογικό Ιστό, οι πληροφορίες έχουν σαφώς καθορισμένο νόημα, επιτρέποντας καλύτερα στους υπολογιστές και στους ανθρώπους να συνεργάζονται (Berners-Lee & Hendler, 2001). Τα Συνδεδεμένα Ανοικτά Δεδομένα (LOD) είναι μια πρακτική δημοσίευσης δομημένων δεδομένων ώστε να μπορούν να διασυνδεθούν και να γίνουν πιο χρήσιμα. Τα LOD (θα αναλυθούν εκτενώς στην συνέχεια) επιτρέπουν τη σύνδεση δομημένων πληροφοριών, δημιουργώντας έναν παγκόσμιο ιστό δεδομένων (Heath & Bizer, 2011).

Το GraphDB, το οποίο αναπτύχθηκε από την Ontotext, είναι μια ισχυρή και επεκτάσιμη βάση δεδομένων RDF, η οποία υποστηρίζει αυτές τις τεχνολογίες του Σημασιολογικού Ιστού, καθιστώντας το ένα αποτελεσματικό εργαλείο για την αποθήκευση, την αναζήτηση και τη δημοσίευση δεδομένων Σημασιολογικού Ιστού. Παρέχοντας ένα τελικό σημείο SPARQL, το GraphDB επιτρέπει τη δημοσίευση δεδομένων RDF στον Ιστό και επιτρέπει σε άλλες εφαρμογές και χρήστες να αλληλεπιδρούν με αυτά τα δεδομένα μέσω HTTP, συμβάλλοντας αποτελεσματικά στην υλοποίηση του οράματος του Σημασιολογικού Ιστού (Bishop et al., 2011).

2.4 Διασυνδεδεμένα Δεδομένα

Τα Συνδεδεμένα δεδομένα είναι ένα σύνολο από τις καλύτερες πρακτικές (best-practises) που αφορούν τη δημοσίευση αλλά και τη συνδεσιμότητα δομημένων δεδομένων στο διαδίκτυο, δίνοντας μας τη δυνατότητα να αναζητήσουμε προγραμματιστικά και να συνδυαστούν με τρόπο τέτοιο, ώστε να είναι αναγνωρίσιμα από υπολογιστές και μηχανές. (Berners-Lee, 2006). Ουσιαστικά χτίζουν πάνω στις θεμελιώδεις αρχές του διαδικτύου αλλά και τις επεκτείνουν, δημιουργώντας ένα διασυνδεδεμένο και σημασιολογικά πλούσιο τοπίο δεδομένων.

Οι αρχές των διασυνδεδεμένων δεδομένων, όπως είναι γνωστό κατά την βιβλιογραφία έχουν προταθεί και διατυπωθεί από τον Tim Berners-Lee, τον δημιουργό του World Wide Web consortium, σε μία ανακοίνωση με τίτλο "Linked Data - Design Issues" (Berners-Lee, 2006).

2.4.1 Οι αρχές των Συνδεδεμένων Δεδομένων

Οι θεμελιώδες αρχές των ΣΔ εμπεριέχουν

1. Τη χρήση URIs (Uniform Resource Identifiers),
2. Μοναδικές ακολουθίες χαρακτήρων που προσδιορίζουν έναν λογικό ή φυσικό πόρο,
3. Την υιοθέτηση των HTTP URIs για την δυνατότητα πρόσβασης στα δεδομένα,
4. Τη χρήση του RDF σαν πρότυπο μοντελοποίησης των δεδομένων,
5. και τέλος τη διασύνδεση συνόλων δεδομένων για τη διευκόλυνση της ανακάλυψης και της ενσωμάτωσης δεδομένων (Berners-Lee, 2006).

2.4.2 Σημασιολογικές Τεχνολογίες και Συνδεδεμένα Δεδομένα

Οι σημασιολογικές τεχνολογίες, όπως το RDF (Resource Description Framework), το OWL (Web Ontology Language) και το SPARQL (ένα γλώσσα ερωτημάτων για το RDF), διαδραματίζουν κρίσιμο ρόλο στην υλοποίηση των Διασυνδεδεμένων Δεδομένων.

Αυτές οι τεχνολογίες παρέχουν έναν τυποποιημένο τρόπο για να αναπαραστήσουμε, να αναρωτηθούμε και να συλλογιστούμε για τα δεδομένα, επιτρέποντας στις μηχανές να κατανοούν και να επεξεργάζονται τα δεδομένα σημασιολογικά (Bizer, Heath, & Berners-Lee, 2009; W3C, 2014).

2.4.3 Εφαρμογές και διάφορα Uses Cases

Τα Διασυνδεδεμένα Δεδομένα έχουν εφαρμογή σε διάφορους τομείς, πχ στις επιστήμες υγείας, στις κοινωνικές επιστήμες, στην πολιτιστική κληρονομιά μας και έχουν χρησιμοποιηθεί από οργανισμούς, από μέσα μαζικής ενημέρωσης μέχρι και απο κυβερνήσεις.

Επίσημα παραδείγματα περιλαμβάνουν το Linked Open Data (LOD) cloud, το οποίο περιέχει χιλιάδες διασυνδεδεμένα σύνολα δεδομένων που καλύπτουν διάφορα θέματα (Schmachtenberg, Bizer, & Paulheim, 2014), το έργο Bio2RDF, το οποίο ενσωματώνει βιοϊατρικά δεδομένα από πολλαπλές πηγές (Belleau κ.ά., 2008), και το DBpedia, το οποίο εξάγει δομημένα δεδομένα από τη Βικιπαίδεια και τα δημοσιεύει ως Διασυνδεδεμένα Δεδομένα (Auer κ.ά., 2007).

2.5 Ανοικτά Συνδεδεμένα Δεδομένα – Linked Open Data

Τα Διασυνδεδεμένα Ανοιχτά Δεδομένα (Linked Open Data - LOD) αποτελούν ένα υποσύνολο των Διασυνδεδεμένων Δεδομένων που επικεντρώνεται στην παροχή ελεύθερα διαθέσιμων, προσβάσιμων και επαναχρησιμοποιήσιμων δεδομένων κάτω από ανοιχτές άδειες. Συνδυάζει τις αρχές των Διασυνδεδεμένων Δεδομένων με το κίνημα των Ανοιχτών Δεδομένων, με στόχο τη δημιουργία ενός παγκόσμιου, διασυνδεδεμένου χώρου δεδομένων, ο οποίος ενθαρρύνει την ανταλλαγή δεδομένων, τη διαλειτουργικότητα και τη συνεργασία.

Το κίνημα των LOD έχει αποκτήσει σημαντική ώθηση από την ίδρυσή του το 2007, όταν ο Tim Berners-Lee, ο εφευρέτης του Παγκόσμιου Ιστού, περιέγραψε το όραμά του για ένα παγκόσμιο "Τεράστιο Παγκόσμιο Γράφημα" (Giant Global Graph - GGG) δεδομένων (Berners-Lee, 2007). Το LOD cloud (σύννεφο), μια οπτική αναπαράσταση των διασυνδεδεμένων συνόλων δεδομένων, τα οποία δημοσιεύονται ως Διασυνδεδεμένα Ανοιχτά Δεδομένα, έχει αυξηθεί σημαντικά κατά τη διάρκεια των τελευταίων ετών, περιλαμβάνοντας χιλιάδες σύνολα δεδομένων από διάφορους τομείς (Schmachtenberg, Bizer, & Paulheim, 2014).

Μερικά αξιοσημείωτα παραδείγματα πρωτοβουλιών LOD περιλαμβάνουν:

DBpedia: Ένα έργο που εξάγει δομημένες πληροφορίες από τη Βικιπαίδεια και τις δημοσιεύει ως Διασυνδεδεμένα Ανοιχτά Δεδομένα (Auer et al., 2007).

Europeana: Μια πλατφόρμα που παρέχει πρόσβαση σε εκατομμύρια ψηφιοποιημένα αντικείμενα από ευρωπαϊκές βιβλιοθήκες, μουσεία και αρχεία, δημοσιεύοντας τα ως Διασυνδεδεμένα Ανοιχτά Δεδομένα (Haslhofer & Isaac, 2011).

Η Βρετανική Βιβλιοθήκη και η Εθνική Βιβλιοθήκη της Γαλλίας: Και οι δύο θεσμοί έχουν δημοσιεύσει τα βιβλιογραφικά τους δεδομένα ως Διασυνδεδεμένα Ανοιχτά Δεδομένα, προωθώντας την πρόσβαση και την επαναχρησιμοποίηση αυτών των πολύτιμων πληροφοριών (Miller, 2011; Tillet, 2011).

Καθιστώντας τα δεδομένα ανοικτά και διασυνδεδεμένα, το κίνημα των LOD προάγει την καινοτομία, την έρευνα και νέες εφαρμογές που αξιοποιούν τη συνδυασμένη γνώση του 1

Συνολικά, το κίνημα των Διασυνδεδεμένων Ανοιχτών Δεδομένων συμβάλλει στη δημιουργία ενός πιο ανοιχτού, διασυνδεδεμένου και καινοτόμου ψηφιακού κόσμου, όπου οι πληροφορίες και οι γνώσεις μπορούν να αξιοποιηθούν ευρέως για την ανάπτυξη και τη βελτίωση της ανθρώπινης ζωής.

2.6 Core Public Service Vocabulary (CPSV)

Το CPSV, ή Core Public Service Vocabulary, είναι ένα μοντέλο δεδομένων που αναπτύχθηκε από το πρόγραμμα ISA² (Interoperability Solutions for European Public Administrations, Businesses and Citizens) της Ευρωπαϊκής Επιτροπής. Το CPSV στοχεύει στην τυποποίηση της περιγραφής των δημοσίων υπηρεσιών που προσφέρουν διάφοροι κυβερνητικοί οργανισμοί σε όλη την Ευρωπαϊκή Ένωση, βελτιώνοντας έτσι τη δια λειτουργικότητά, την ανακαλυψιμότητα και την προσβασιμότητα των πληροφοριών των δημόσιων υπηρεσιών.

Το CPSV παρέχει ένα κοινό πλαίσιο για την περιγραφή των δημοσίων υπηρεσιών χρησιμοποιώντας ένα σύνολο τυποποιημένων ιδιοτήτων και κλάσεων. Εφαρμόζοντας αυτό

το κοινό λεξιλόγιο, οι κυβερνητικοί οργανισμοί μπορούν να μοιράζονται πιο αποτελεσματικά πληροφορίες σχετικά με τις δημόσιες υπηρεσίες τους, επιτρέποντας στους πολίτες και τις επιχειρήσεις να εντοπίζουν και να κατανοούν τις διαθέσιμες υπηρεσίες.

Το Core Public Service Vocabulary είναι μέρος ενός ευρύτερου συνόλου βασικών λεξιλογίων που αναπτύχθηκαν από το πρόγραμμα ISA², το οποίο περιλαμβάνει επίσης το Core Person Vocabulary, το Core Business Vocabulary και το Core Location Vocabulary. Αυτά τα βασικά λεξιλόγια έχουν σχεδιαστεί για να διευκολύνουν την ανταλλαγή δεδομένων και τη δια-λειτουργικότητά μεταξύ των δημόσιων διοικήσεων σε ολόκληρη την Ευρώπη.

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

Το Core Public Service Vocabulary Application Profile (CPSV-AP) είναι μια επέκταση του μοντέλου δεδομένων Core Public Service Vocabulary (CPSV), ή οποία αναπτύχθηκε στο πλαίσιο του προγράμματος ISA² (Interoperability Solutions for European Public Administrations, Businesses, and Citizens) της Ευρωπαϊκής Επιτροπής. Το CPSV-AP στοχεύει να παρέχει μια κοινή προσέγγιση στην περιγραφή των δημοσίων υπηρεσιών που προσφέρουν διάφοροι κρατικοί οργανισμοί σε ολόκληρη την Ευρωπαϊκή Ένωση, προωθώντας τη δια λειτουργικότητά, την ανακαλυψιμότητα και την προσβασιμότητα των πληροφοριών των δημόσιων υπηρεσιών (ISA², 2020).

Στοιχεία του CPSV-AP:

Το CPSV-AP επεκτείνει το μοντέλο δεδομένων CPSV παρέχοντας μια σειρά από τυποποιημένες ιδιότητες και κλάσεις για να περιγράψει τις δημόσιες υπηρεσίες με πιο λεπτομερή τρόπο. Περιλαμβάνει έννοιες όπως "Δημόσια Υπηρεσία", "Δημόσιος Οργανισμός", "Πράκτορας", "Κανόνας", "Κριτήρια", "Αποδεικτικά Στοιχεία", "Κανάλι" και "Συμβάν Ζωής", μεταξύ άλλων. Χρησιμοποιώντας αυτούς τους τυποποιημένους όρους, οι κυβερνητικοί οργανισμοί μπορούν να περιγράψουν συνεπώς τις δημόσιες υπηρεσίες τους, επιτρέποντας πιο αποτελεσματική κοινή χρήση πληροφοριών και ανακάλυψη (ISA², 2020).

Εφαρμογές και Περιπτώσεις Χρήσης:

Το CPSV-AP έχει υιοθετηθεί από αρκετές ευρωπαϊκές χώρες για να βελτιώσουν την ανακαλυψιμότητα και την προσβασιμότητα των δημόσιων υπηρεσιών. Για παράδειγμα, η φλαμανδική κυβέρνηση του Βελγίου έχει εφαρμόσει το CPSV-AP για να περιγράψει τις δημόσιες υπηρεσίες της στη φλαμανδική πύλη ανοικτών δεδομένων, παρέχοντας ένα ενιαίο σημείο πρόσβασης για τους πολίτες και τις επιχειρήσεις για να εντοπίζουν και να κατανοούν τις διαθέσιμες υπηρεσίες (Vlaanderen, 2018). Παρομοίως, η ισπανική κυβέρνηση έχει υιοθετήσει το CPSV-AP στον κατάλογο δημοσίων υπηρεσιών της (AgenSynd, 2019).

Συμπέρασμα:

Το Core Public Service Vocabulary Application Profile (CPSV-AP) διαδραματίζει κρίσιμο ρόλο στην ενίσχυση της δια-λειτουργικότητας και την ευκολία εύρεσης των δημόσιων υπηρεσιών σε ολόκληρη την Ευρωπαϊκή Ένωση. Παρέχοντας ένα τυποποιημένο πλαίσιο για την περιγραφή των δημόσιων υπηρεσιών, το CPSV-AP επιτρέπει στις κυβερνήσεις να μοιράζονται πιο αποτελεσματικά πληροφορίες σχετικά με τις υπηρεσίες τους, επωφελούμενοι έτσι τόσο οι πολίτες όσο και οι επιχειρήσεις. Καθώς περισσότερες χώρες υιοθετούν το CPSV-AP, θα γίνει ολοένα και πιο σημαντικό να διασφαλίζεται ότι οι περιγραφές των δημοσίων υπηρεσιών παραμένουν ενημερωμένες και περιεκτικές, ξεκλειδώνοντας το πλήρες δυναμικό του κοινόχρηστου τοπίου δεδομένων.

2.8 Προγραμματιστικές Διεπαφές Εφαρμογών API's τύπου REST

Το API (Application Programming Interface), ή Διασύνδεση Προγραμματισμού Εφαρμογών, είναι ένα σύνολο κανόνων, πρωτοκόλλων και εργαλείων, το οποίο επιτρέπει σε διαφορετικές εφαρμογές λογισμικού να επικοινωνούν μεταξύ τους. Τα API επιτρέπουν στους προγραμματιστές να αλληλοεπιδρούν με εξωτερικές υπηρεσίες, συστήματα ή συστατικά, παρέχοντας μια καλά ορισμένη διεπαφή για την ανταλλαγή δεδομένων και λειτουργιών. Αυτό επιτρέπει την κατασκευή εφαρμογών λογισμικού πάνω από υπάρχουσες υπηρεσίες, πηγές δεδομένων και πλατφόρμες, μειώνοντας το χρόνο και την προσπάθεια ανάπτυξης (T. M. Mikkonen & Taivalsaari, 2014).

Τα API μπορούν να καταταχθούν σε διαφορετικούς τύπους, όπως τα Web API, τα API λειτουργικών συστημάτων και τα API βιβλιοθηκών ή πλαισίων. Τα Web API, ιδιαίτερα, έχουν αποκτήσει ιδιαίτερη σημασία με την ευρεία υιοθέτηση του διαδικτύου και την ανάγκη για δια-λειτουργικές υπηρεσίες ιστού (Pautasso, Zimmermann, & Leymann, 2008).

Μερικά γνωστά παραδείγματα API περιλαμβάνουν:

Google Maps API: Επιτρέπει στους προγραμματιστές να ενσωματώσουν λειτουργίες του Google Maps στις εφαρμογές τους, όπως την εμφάνιση χαρτών, την παροχή οδηγιών και την αναζήτηση τοποθεσιών (Google, χ.χ.).

Twitter API: Επιτρέπει στους προγραμματιστές να έχουν πρόσβαση και να αλληλεπιδρούν με τα δεδομένα του Twitter, όπως τη λήψη tweets, την ανάρτηση ενημερώσεων και την αναζήτηση για συγκεκριμένες λέξεις-κλειδιά ή hashtags (Twitter, χ.χ.).

Τα API διαδραματίζουν κρίσιμο ρόλο στη σύγχρονη ανάπτυξη λογισμικού, καθώς διευκολύνουν την ένταξη διαφόρων συστημάτων και υπηρεσιών, προωθώντας τη δια-λειτουργικότητά, τη προσαρμοστικότητα και την επαναχρησιμοποίηση λειτουργιών.

2.9 Resource Description Framework – Πλαίσιο Περιγραφής Πόρων

Το RDF, ή Resource Description Framework, είναι ένα πρότυπο μοντέλο για την ανταλλαγή δεδομένων στον ιστό, το οποίο αναπτύχθηκε από τον Παγκόσμιο Οργανισμό Προτύπων Ιστού (W3C). Το RDF παρέχει έναν ευέλικτο και επεκτάσιμο τρόπο αναπαράστασης πληροφοριών σχετικά με πόρους, επιτρέποντας την εύκολη κοινή χρήση, συγχώνευση και επαναχρησιμοποίηση δεδομένων σε διαφορετικές εφαρμογές και τομείς (Klyne & Carroll, 2004).

Χρησιμοποιεί τριπλέτες, που αποτελούνται από εκφράσεις υποκειμένου-ρήμα-αντικειμένου, για να αναπαραστήσει τα δεδομένα ως γράφημα διασυνδεδεμένων κόμβων και ακμών. Κάθε τριάδα περιγράφει μια σχέση μεταξύ δύο πόρων ή μεταξύ ενός πόρου και μιας λεξιλογικής τιμής. Στο RDF, οι πόροι προσδιορίζονται από URI (Uniform Resource

Identifiers), που παρέχουν παγκοσμίως μοναδικά και μόνιμα αναγνωριστικά για στοιχεία δεδομένων (Klyne & Carroll, 2004).

Το RDF είναι ένα κύριο συστατικό του Semantic Web, μιας επέκτασης του παραδοσιακού Ιστού και στοχεύει να καταστήσει τα δεδομένα πιο αναγνώσιμα και κατανοητά από τις μηχανές. Υπηρετεί ως βάση για άλλες τεχνολογίες του Semantic Web, όπως το RDFS (RDF Schema) και το OWL (Web Ontology Language), τα οποία παρέχουν επιπλέον εκφραστικότητα και δυνατότητες συλλογισμού (W3C, 2014).

Μερικές χρήσεις του RDF περιλαμβάνουν:

Ενσωμάτωση δεδομένων: Το RDF μπορεί να χρησιμοποιηθεί για την ένταξη δεδομένων από διαφορετικές πηγές, επιτρέποντας την δια λειτουργικότητά και την ανταλλαγή δεδομένων μεταξύ διαφορετικών εφαρμογών (Heath & Bizer, 2011).

Αναπαράσταση γνώσης: Το RDF μπορεί να χρησιμοποιηθεί για τη μοντελοποίηση πολύπλοκων δομών γνώσης, όπως οντολογίες και ταξινομίες, υποστηρίζοντας τη σημασιολογική λογική συλλογισμού και συμπερασματολογίας (Allemang & Hendler, 2011).

2.10 Semantic MediaWiki (SMW)

Το Semantic MediaWiki (SMW) είναι μια επέκταση της ευρέως χρησιμοποιούμενης πλατφόρμας MediaWiki, η οποία τροφοδοτεί ιστότοπους όπως το Wikipedia. Το SMW βελτιώνει την παραδοσιακή πλατφόρμα MediaWiki εισάγοντας σημασιολογικές δυνατότητες, επιτρέποντας στους χρήστες να δομούν και να προσθέτουν σημειώσεις στα δεδομένα εντός των σελίδων wiki, καθιστώντας τα πιο προσβάσιμα και αναγνώσιμα από μηχανές (Krötzsch, Vrandečić, & Völkel, 2006).

Το SMW επιτρέπει στους χρήστες να προσθέτουν σημασιολογικές σημειώσεις στις σελίδες wiki χρησιμοποιώντας μια απλή σύνταξη σήμανσης, επιτρέποντας τη δημιουργία σχέσεων μεταξύ σελίδων και την ανάθεση ιδιοτήτων στα στοιχεία δεδομένων. Οι σημασιολογικές σημειώσεις μπορούν στη συνέχεια να χρησιμοποιηθούν για τη δημιουργία δυναμικού

περιεχομένου, όπως λίστες, πίνακες και χάρτες, με βάση τα δομημένα δεδομένα που αποθηκεύονται στο wiki (Krötzsch, Vrandečić, & Völkel, 2006).

Διαχείριση γνώσης: Το SMW μπορεί να χρησιμοποιηθεί ως εργαλείο συνεργατικής διαχείρισης γνώσης, επιτρέποντας στους χρήστες να δημιουργούν, δομούν και μοιράζονται πληροφορίες εντός μιας οργάνωσης ή κοινότητας (Berners-Lee, 2006).

Ενσωμάτωση δεδομένων: Το SMW μπορεί να χρησιμοποιηθεί για την ενσωμάτωση και εναρμόνιση δεδομένων από διαφορετικές πηγές, διευκολύνοντας την ανταλλαγή δεδομένων και την διαλειτουργικότητα μεταξύ διαφορετικών συστημάτων (Krötzsch, Vrandečić, & Völkel, 2006).

Ενσωματώνοντας σημασιολογικές δυνατότητες στην πλατφόρμα MediaWiki, το Semantic MediaWiki παρέχει μια ισχυρή και ευέλικτη λύση για τη διαχείριση και την κοινή χρήση δομημένων δεδομένων στον ιστό.

2.11 SPARQL

Το SPARQL, ή SPARQL Protocol and RDF Query Language, είναι μια τυποποιημένη γλώσσα ερωτημάτων και πρωτόκολλο που αναπτύχθηκε από το World Wide Web Consortium (W3C) για την εξαγωγή και επεξεργασία δεδομένων RDF (Resource Description Framework). Το SPARQL επιτρέπει στους χρήστες να ανακτούν και να επεξεργάζονται δεδομένα που είναι αποθηκευμένα σε μορφή RDF, εκτελώντας διάφορους τύπους ερωτημάτων, όπως SELECT, CONSTRUCT, DESCRIBE και ASK (Prud'hommeaux & Seaborne, 2008).

Το SPARQL είναι ένα βασικό συστατικό του Semantic Web, καθώς παρέχει ένα μέσο πρόσβασης και ερωτήματος για τα δομημένα δεδομένα που αναπαρίστανται χρησιμοποιώντας RDF. Υποστηρίζει λειτουργίες αντιστοίχισης μοτίβων, φιλτραρίσματος και συγχώνευσης, επιτρέποντας στους χρήστες να εκτελούν περίπλοκα ερωτήματα πάνω σε δεδομένα RDF (Pérez, Arenas, & Gutierrez, 2006).

Μερικές χρήσεις του SPARQL περιλαμβάνουν:

Ενοποίηση δεδομένων: Το SPARQL μπορεί να χρησιμοποιηθεί για το ερώτημα και την ενσωμάτωση δεδομένων από πολλαπλές πηγές δεδομένων RDF, διευκολύνοντας την

ανταλλαγή δεδομένων και τη δια-λειτουργικότητα μεταξύ διαφορετικών συστημάτων (Bizer, Heath, & Berners-Lee, 2009).

Σημασιολογική αναζήτηση: Το SPARQL μπορεί να χρησιμοποιηθεί για τη δημιουργία μηχανών αναζήτησης σημασιολογικών δεδομένων, επιτρέποντας στους χρήστες να εκτελούν περίπλοκα ερωτήματα πάνω σε δομημένα δεδομένα, όπως η αναζήτηση οντοτήτων με βάση τις ιδιότητες και τις σχέσεις τους (Arenas, Cuenca Grau, Kharlamov, Marciuska, & Zheleznyakov, 2016).

3. Ανάλυση και σχεδίαση της Αρχιτεκτονικής του έργου

3.1 Εισαγωγή

Καθώς προχωράμε στην εξερεύνηση της δημοσίευσης των Δημόσιων Υπηρεσιών ως Συνδεδεμένα Ανοικτά Δεδομένα (LOD), η αρχιτεκτονική της προτεινόμενης λύσης παίζει καθοριστικό ρόλο. Αυτό το κεφάλαιο με τίτλο «Ανάλυση και Σχεδιασμός της Αρχιτεκτονικής», εμβαθύνει σχολαστικά σε αυτό το αναπόσπαστο μέρος της έρευνάς μας.

Είναι αδιαμφισβήτητο γεγονός ότι η αρχιτεκτονική οποιουδήποτε συστήματος διαχείρισης δεδομένων δεν χρησιμεύει μόνο ως το προσχέδιο για το σύνολο της εγκατάστασης, αλλά επίσης υπαγορεύει την αποτελεσματικότητα, την ευελιξία και την επεκτασιμότητα του. Για το σύστημά μας, είναι υψίστης σημασίας η δημιουργία μιας αρχιτεκτονικής που μπορεί να χειριστεί με επιτυχία μεγάλο όγκο πληροφοριών δημόσιας υπηρεσίας, διασφαλίζοντας παράλληλα ότι είναι σωστά συνδεδεμένη και ανοιχτή για δημόσια χρήση.

Σε αυτό το κεφάλαιο, θα σχεδιάσουμε μια αρχιτεκτονική που ενσωματώνει τις βασικές αρχές του LOD και επιτρέπει την απρόσκοπτη ροή δεδομένων από τη συλλογή έως τη δημοσίευση. Ο σχεδιασμός μας έχει επηρεαστεί από τις βασικές απαιτήσεις της δια-λειτουργικότητας των δεδομένων, της διαφάνειας και της προσβασιμότητας, που αποτελούν όλα χαρακτηριστικά γνωρίσματα του LOD.

Η γενική ροή εργασιών της αρχιτεκτονικής μας, όπως περιγράφεται λεπτομερώς σε αυτό το κεφάλαιο, περιλαμβάνει διάφορα στάδια: συλλογή πληροφοριών, επεξεργασία τους, μεταμόρφωσή τους, κατασκευή τους και τελικά δημοσίευσή τους. Αυτή η συστηματική διαδικασία διασφαλίζει ότι τα ακατέργαστα δεδομένα δεν απορρίπτονται απλώς στο δημόσιο τομέα, αλλά επιμελούνται, επεξεργάζονται, συνδέονται και παρουσιάζονται με ουσιαστικό τρόπο, ενισχύοντας έτσι τη χρησιμότητα και τη συνάφειά τους.

Το κεφάλαιο ξεκινά με μια επισκόπηση της αρχιτεκτονικής API και των σημαντικών χαρακτηριστικών της. Στη συνέχεια, προχωρά σε μια σταδιακή ανάλυση ολόκληρης της ροής εργασίας, περιγράφοντας τις περίπλοκες διαδικασίες που εμπλέκονται σε κάθε βήμα. Οι προκλήσεις και οι λύσεις που σχετίζονται με κάθε στάδιο θα συζητηθούν επίσης σε βάθος για να δοθεί μια ρεαλιστική εικόνα της διαδικασίας και να επισημανθούν τα μέτρα που λαμβάνονται για τον μετριασμό πιθανών ζητημάτων.

Μέχρι το τέλος αυτού του κεφαλαίου, οι αναγνώστες θα αποκτήσουν μια πλήρη κατανόηση της αρχιτεκτονικής και της ροής εργασίας που εμπλέκονται στη μετατροπή των δεδομένων δημόσιας υπηρεσίας σε LOD. Αυτή η κατανόηση θα χρησιμεύσει ως βάση για το επόμενο κεφάλαιο, όπου θα εμβαθύνουμε στην τεχνική υλοποίηση και αξιολόγηση της προτεινόμενης λύσης μας.

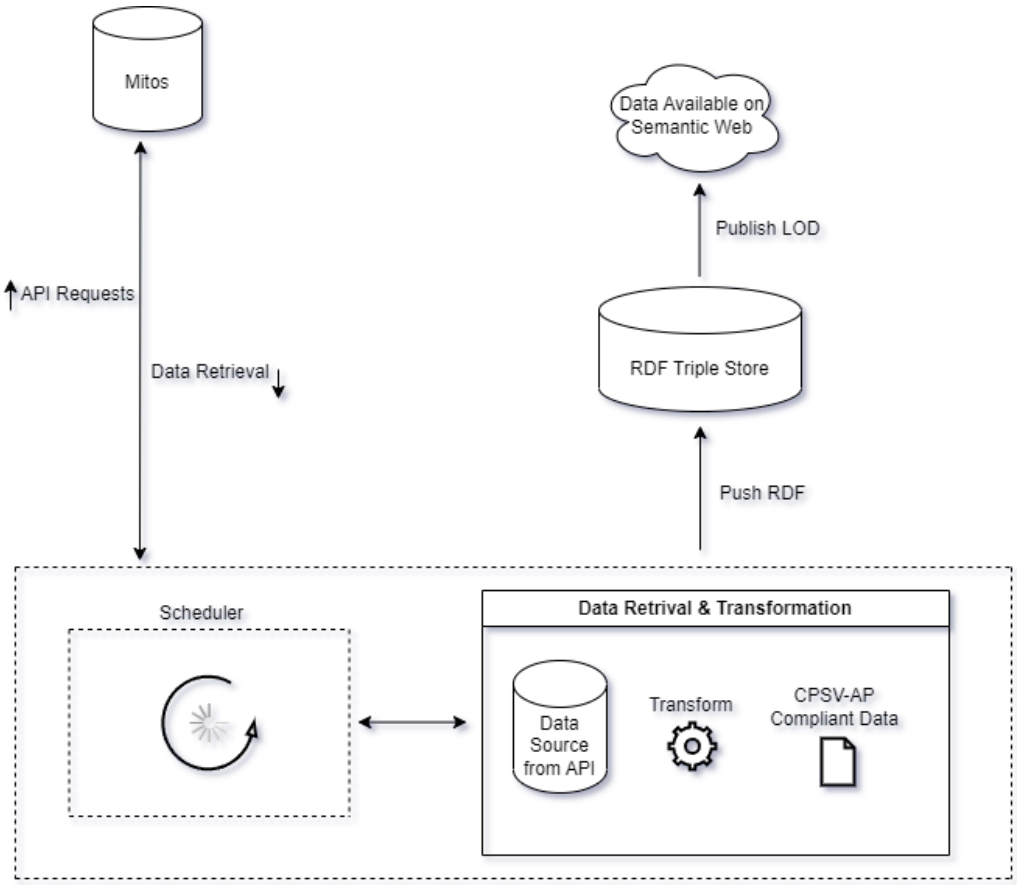
Στην ουσία, η «Ανάλυση και Σχεδιασμός της Αρχιτεκτονικής» αποτελεί τη ραχοκοκαλιά της διπλωματικής εργασίας, καθώς εξηγεί το σχέδιο της προσέγγισής μας και τους τρόπους υπό τους οποίους καθίστανται τα δεδομένα δημόσιας υπηρεσίας πιο πολύτιμα, προσβάσιμα και χρήσιμα σε ένα ευρύ φάσμα χρηστών.

3.2 Ανάλυση της Αρχιτεκτονικής

Η βασική λειτουργία της αρχιτεκτονικής μας στηρίζεται στα δεδομένα που υπάρχουν για να επιτελεστούν όλες οι απαραίτητες λειτουργίες προκειμένου να υλοποιηθεί το έργο.

Όπως έχει προαναφερθεί τα δεδομένα αυτά προέρχονται από εδραιωμένες καλές πρακτικές ή από τη βιβλιογραφία που έχει συνδράμει σημαντικά στην μελέτη του συγκεκριμένου κλάδου.

Έτσι λοιπόν, αφού έχει προηγηθεί αυτή η πραγμάτευση τόσο στο υλικοτεχνικό όσο και στο θεωρητικό κομμάτι της υλοποίησης μπορούμε με ακρίβεια και ασφάλεια να οδηγηθούμε στην θεώρηση του μοντέλου, το οποίο θα είναι προδιαγεγραμμένο έτσι ακριβώς ώστε να εξυπηρετήσει τις ανάγκες μας.



Εικόνα 2. Γενικό Μοντέλο ροής της αρχιτεκτονικής

Αυτό το μοντέλο λοιπόν, είναι μια κατανεμημένη αρχιτεκτονική που εκμεταλλεύεται κυρίως τα API, διάφορες γλώσσες προγραμματισμού, τις τεχνολογίες του σημασιολογικού διαδικτύου και τις υπηρεσίες *ενορχήστρωσης (Orchestration). Ο στόχος είναι η απόκτηση δεδομένων, ο μετασχηματισμός, η αποθήκευση και η δυνατότητα ερώτησης (querying) στο Σημασιολογικό Διαδίκτυο. Ας αναλύσουμε κάθε συστατικό και τις αλληλεπιδράσεις τους.

Αλληλεπιδράσεις με το API του Mitos: Το σύστημα ξεκινά μια διαδικασία ανάκτησης δεδομένων αποστέλλοντας αιτήματα HTTP GET στη βάση δεδομένων Mitos μέσω του API του. Το API είναι ο τυποποιημένος τρόπος επικοινωνίας μεταξύ διαφορετικών συστατικών λογισμικού, εξασφαλίζοντας ότι τα δεδομένα μοιράζονται με δομημένο και προβλέψιμο τρόπο.

***Επεξήγηση Όρου:** Ο όρος ενορχήστρωση στην πληροφορική παραπέμπει στην αυτοματοποίηση των διαδικασιών, των ρυθμίσεων και της εποπτείας υπολογιστικών συστημάτων και λογισμικού (Thomas*

Erl. Service-Oriented Architecture: Concepts, Technology, and Design (The Pearson Service Technology Series from Thomas Erl) ISBN: 0-13-185858-0

Μετασχηματισμός Δεδομένων: Κατά την λήψη των δεδομένων από το API του Mitos, χρησιμοποιούνται scripts γραμμένα σε Python 3 για να μετασχηματίσουν τα δεδομένα σε συμβατά με CPSV-AP (Core Public Service Vocabulary Application Profile) δεδομένα. Το CPSV-AP είναι ένα τυποποιημένο μοντέλο δεδομένων που επιτρέπει την περιγραφή δημόσιων υπηρεσιών. Αυτό εξασφαλίζει τη συνέπεια των δεδομένων και τη συμβατότητα, ειδικά για οντότητες που πρέπει να μοιράζονται δεδομένα δημόσιων υπηρεσιών. Η RDFlib, μια βιβλιοθήκη της Python, χρησιμοποιείται για την εργασία με το RDF (Resource Description Framework), ένα τυποποιημένο μοντέλο για την ανταλλαγή δεδομένων στο Διαδίκτυο.

RDF Triple Store: Τα μετασχηματισμένα και μοντελοποιημένα δεδομένα αποθηκεύονται στη συνέχεια σε ένα RDF Triple Store. Το RDF Triple Store είναι μια βάση δεδομένων που κατασκευάστηκε ειδικά για την αποθήκευση και ανάκτηση τριπλέτων RDF. Μια "τριπλέτα" σε αυτό το πλαίσιο αναφέρεται στην ατομική οντότητα δεδομένων στο RDF, η οποία αποτελείται από ένα υποκείμενο-ρήμα-αντικείμενο. Η αποθήκευση δεδομένων με αυτόν τον τρόπο καθιστά τα δεδομένα έτοιμα για δημοσίευση στο Σημασιολογικό Διαδίκτυο, επιτρέποντας την κατανόηση, τον διαμοιρασμό και την επαναχρησιμοποίηση τους σε διάφορα πεδία εφαρμογής, επιχείρησης και κοινότητας.

Διαδικασία «ενορχήστρωσης»: Αυτό είναι ένα κρίσιμο μέρος του συστήματος που παρακολουθεί την διαδικασία από την αρχή ως το τέλος, εξασφαλίζοντας την ομαλή ενσωμάτωση και εκτέλεση όλων των διαδικασιών. Παρακολουθεί, καταγράφει και επιβλέπει τα γεγονότα, παρέχοντας προειδοποιήσεις που επιτρέπουν την προληπτική διόρθωση σφαλμάτων και επίσης βοηθά στη βελτίωση του συνολικού συστήματος μέσω της παρακολούθησης της απόδοσης και της βελτιστοποίησης.

Συνολικά, αυτή η αρχιτεκτονική δείχνει μια σύγχρονη προσέγγιση στην επεξεργασία και τον μετασχηματισμό των δεδομένων, ακολουθώντας τα πρότυπα του διαδικτύου, διασφαλίζοντας τη συνέπεια των δεδομένων και επιτρέποντας αποτελεσματική παρακολούθηση σφαλμάτων.

3.3 Συλλογή Δεδομένων και Διαχείριση

Όπως αναλύσαμε και αναφέραμε προηγουμένως, είναι δεδομένο ότι στην ταχέως εξελισσόμενη ψηφιακή σφαίρα, η αρχιτεκτονική λογισμικού διαδραματίζει κρίσιμο ρόλο στον τρόπο απόκτησης, αποθήκευσης και διαχείρισης των πληροφοριών.

Το Mitos.gov.gr προσφέρει ένα ισχυρό API που επιτρέπει στους χρήστες να αντλούν δεδομένα μέσω αιτημάτων REST τύπου GET. Αυτά τα δεδομένα μπορούν να ενσωματωθούν σε εφαρμογές λογισμικού, βελτιώνοντας τη λειτουργικότητα και βελτιώνοντας το εύρος και την αποτελεσματικότητα της διαχείρισης δεδομένων.

Η προσέγγισή μας περιλαμβάνει την εξαγωγή των διαθέσιμων δεδομένων από τα τελικά σημεία (endpoints) του API χρησιμοποιώντας καλά δομημένα αιτήματα GET. Στη συνέχεια, τα δεδομένα που ανακτώνται αποθηκεύονται και οργανώνονται συστηματικά, ανοίγοντας το δρόμο για μεταγενέστερη επεξεργασία και ανάλυση. Ένα κεντρικό στοιχείο της μελέτης μας περιλαμβάνει τον εντοπισμό αποτελεσματικών σχημάτων αποθήκευσης που διασφαλίζουν την προσβασιμότητα και την ακεραιότητα των δεδομένων. Αυτή η διαδικασία είναι αναπόσπαστο μέρος της δημιουργίας ενός αξιόπιστου συστήματος διαχείρισης δεδομένων, το οποίο μπορεί να χειριστεί μεγάλους όγκους δεδομένων διατηρώντας παράλληλα την ποιότητα των δεδομένων.

Επιπλέον, η μελέτη διερευνά μεθόδους διασφάλισης της συνέπειας, της ασφάλειας και της επεκτασιμότητας των δεδομένων στην αρχιτεκτονική του λογισμικού μας. Εξετάζεται ο αντίκτυπος αυτών των παραγόντων στην απόδοση του συστήματος καθώς και τεχνικές βελτιστοποίησης της απόδοσης δεδομένων και των χρόνων ανάκτησης.

3.4 Μετατροπή Δεδομένων και Συνδεσιμότητα

Ο πλούτος των διαθέσιμων δεδομένων σήμερα απαιτεί ισχυρές και αποτελεσματικές διαδικασίες μετασχηματισμού δεδομένων για να μεγιστοποιηθεί η χρησιμότητά τους. Η παρούσα διατριβή εστιάζει στη μεθοδολογία και την υλοποίηση ενός συστήματος που μετατρέπει δεδομένα από αρχεία CSV σε RDF (Πλαίσιο Περιγραφής Πόρων) Turtles χρησιμοποιώντας την γλώσσα προγραμματισμού Python και την βιβλιοθήκη RDFLib.

Η διαδικασία ξεκινά με την απόκτηση δεδομένων από τα τερματικά σημεία (endpoints) του API που παρέχονται από το Mitos.gov.gr, όπως περιγράψαμε προηγούμενος, τα οποία αποθηκεύονται σε καλά δομημένα αρχεία CSV. Τα αρχεία CSV προσφέρουν μια ευρέως αποδεκτή, απλή και κατανοητή μορφή, καθιστώντας τα ιδανικά για αρχική αποθήκευση δεδομένων. Ωστόσο, ενώ αυτή η μορφή είναι εύκολα αναγνώσιμη από ανθρώπους και μηχανές, υστερεί σε ικανότητες σημασιολογικής αναπαράστασης και σύνδεσης, τις οποίες αντιμετωπίζει η μορφή RDF.

Για να διευκολυνθεί ο μετασχηματισμός δεδομένων από CSV σε RDF Turtles, χρησιμοποιείται η βιβλιοθήκη RDFLib, ένα εργαλείο Python που έχει σχεδιαστεί για εργασία με RDF. Αυτή η βιβλιοθήκη επιτρέπει την ανάλυση, τη σειριοποίηση και τον χειρισμό δεδομένων RDF. Στη μεθοδολογία μας, ορίζουμε χώρους ονομάτων και δημιουργούμε ένα γράφημα RDF για την αποθήκευση των δεδομένων μας. Τα δεδομένα CSV διαβάζονται σειρά προς σειρά, δημιουργώντας URIRefs για κάθε θέμα με βάση το αντίστοιχο αναγνωριστικό σειράς. Τριπλές (θέμα-κατηγόρημα-αντικείμενο) προστίθενται στη συνέχεια στο γράφημα, όπου τα κατηγορήματα ορίζονται με βάση τα πεδία CSV και τα αντικείμενα με βάση τις αντίστοιχες τιμές τους.

Αυτή η υλοποίηση είναι κρίσιμη καθώς παρέχει μια διαδρομή για τη μετάφραση των δεδομένων που είναι αποθηκευμένα σε παραδοσιακή μορφή πίνακα (CSV) σε μια πιο πλούσια σημασιολογικά, συνδεδεμένη μορφή δεδομένων (RDF). Ο μετασχηματισμός σε RDF Turtles επιτρέπει την καλύτερη ενσωμάτωση με άλλα σύνολα δεδομένων, βελτιωμένες δυνατότητες σημασιολογικής αναζήτησης και μια πιο ισχυρή κατανόηση του πλαισίου και των σχέσεων των δεδομένων.

3.5 Ενορχήστρωση, Αυτοματοποίηση και logs

Η χρησιμότητα του Apache Airflow σε μια αρχιτεκτονική λογισμικού που βασίζεται σε δεδομένα είναι ανεκτίμητη. Βελτιώνει τις σύνθετες διαδικασίες που εμπλέκονται στην ανάκτηση, αποθήκευση, μετασχηματισμό και δημοσίευση Συνδεδεμένων Ανοικτών Δεδομένων (LOD). Η δυνατότητα αυτοματοποίησης αυτών των εργασιών χρησιμοποιώντας το Apache Airflow μειώνει την πιθανότητα σφαλμάτων, ενισχύει την αποτελεσματικότητα και επιτρέπει την επεκτασιμότητα.

Η αρχιτεκτονική μας χρησιμοποιεί τα Κατευθυνόμενα Ακυκλικά Γραφήματα (DAG) του Apache Airflow για να μοντελοποιήσει τη γραμμή δεδομένων. Κάθε κόμβος στο DAG αντιπροσωπεύει μια διαφορετική εργασία που εμπλέκεται στον κύκλο ζωής των δεδομένων, όπως η ανάκτηση δεδομένων από το API του Mitos.gov.gr, η αποθήκευση αυτών των δεδομένων, η μετατροπή τους από CSV σε RDF Turtles και τελικά η δημοσίευση τους ως LOD. Αυτή η δομημένη, αυτοματοποιημένη ακολουθία λειτουργιών ελαχιστοποιεί τη χειροκίνητη παρέμβαση και βελτιώνει τη συνολική ακεραιότητα των δεδομένων.

Επιπλέον, το Apache Airflow προσφέρει ένα ισχυρό σύστημα καταγραφής, το οποίο καταγράφει την κατάσταση κάθε εργασίας σε πραγματικό χρόνο. Αυτό είναι επωφελές για σκοπούς αντιμετώπισης προβλημάτων και εντοπισμού σφαλμάτων, καθώς βοηθά στην έγκαιρη επίλυση προβλημάτων που ενδέχεται να προκύψουν. Το σύστημα καταγραφής ενισχύει επίσης τη διαφάνεια και την ιχνηλασιμότητα στη διοχέτευση δεδομένων, καθιστώντας ευκολότερη την παρακολούθηση και τον έλεγχο των διαδικασιών.

Η προσαρμοστικότητα και ο προγραμματισμός του Apache Airflow είναι επίσης σημαντικά χαρακτηριστικά. Επιτρέπουν την προσαρμογή και τη βελτιστοποίηση των ροών εργασίας, διασφαλίζοντας ότι η αρχιτεκτονική μπορεί να ανταποκριθεί αποτελεσματικά στους μεταβαλλόμενους όγκους δεδομένων και να διατηρήσει την ευρωστία κατά τη διάρκεια πιθανών αστοχιών.

Ουσιαστικά, το Apache Airflow προσφέρει μια ισχυρή πλατφόρμα για τη διαχείριση πολύπλοκων ροών εργασίας δεδομένων, ενισχύοντας την απόδοση, την αξιοπιστία και τη διαφάνεια. Οι δυνατότητές του στην ενορχήστρωση εργασιών, την αυτοματοποίηση και την καταγραφή είναι ζωτικής σημασίας στο πλαίσιο μιας αρχιτεκτονικής λογισμικού που βασίζεται σε δεδομένα, μεταμορφώνοντας τον τρόπο διαχείρισης και χρήσης των δεδομένων.

3.6 Διοικητικές Διαδικασίες και Ανοικτά Συνδεδεμένα Δεδομένα

Η σχέση μεταξύ των δημόσιων υπηρεσιών και των Συνδεδεμένων Ανοικτών Δεδομένων (LOD) είναι ουσιαστική στην ψηφιακή εποχή. Οι δημόσιες υπηρεσίες παράγουν τεράστιο όγκο δεδομένων, τα οποία, όταν μετατρέπονται σε LOD, μπορούν να προσφέρουν

αυξημένη διαφάνεια, να προωθήσουν την καινοτομία και να ενισχύσουν τη συμμετοχή των πολιτών. Ωστόσο, για την πλήρη αξιοποίηση αυτών των πλεονεκτημάτων, οι κατάλληλοι μηχανισμοί αποθήκευσης και δημοσίευσης είναι ζωτικής σημασίας. Σε αυτό το σημείο εμπλέκεται το GraphDB, το οποίο λειτουργεί ως αποτελεσματική λύση για την αποθήκευση και τη διαχείριση του LOD στην αρχιτεκτονική μας.

Το GraphDB είναι μια εξαιρετικά επεκτάσιμη, έτοιμη για επιχειρήσεις και οργανισμούς βάση δεδομένων σημασιολογικών γραφημάτων, η οποία χειρίζεται αποτελεσματικά πολύπλοκα δεδομένα σημασιολογικών γραφημάτων. Στην αρχιτεκτονική μας, χρησιμεύει ως ένα ισχυρό σύστημα αποθήκευσης για τις μεταμορφωμένες χελώνες RDF. Αυτό το σύστημα παρέχει ένα σημασιολογικό πλαίσιο, ενισχύοντας έτσι τη δομή, την ολοκλήρωση και την ανάκτηση των δεδομένων. Επιπλέον, οι δυνατότητες του GraphDB, όπως η εξαγωγή συμπερασμάτων και η γρήγορη φόρτωση δεδομένων, το καθιστούν εξαιρετική επιλογή για τη διαχείριση μεγάλων όγκων LOD.

Εκτός από την αποθήκευση, το GraphDB παίζει επίσης ζωτικό ρόλο στη δημοσίευση του LOD. Παρέχει ένα τελικό σημείο για το οποίο μπορεί να αναζητηθεί χρησιμοποιώντας το Πρωτόκολλο και Γλώσσα ερωτημάτων SPARQL αλλά και να χαρτογραφηθεί με την χρήση του RDF, διευκολύνοντας την προσβασιμότητα των δεδομένων. Οι δημόσιες υπηρεσίες μπορούν να χρησιμοποιήσουν αυτή τη δυνατότητα για τη διάδοση των δεδομένων τους, επιτρέποντας στους ενδιαφερόμενους φορείς, τους προγραμματιστές και τους πολίτες να έχουν πρόσβαση και να κάνουν χρήση των δεδομένων. Αυτή η δημόσια προσβασιμότητα προάγει τη διαφάνεια και επιτρέπει την καινοτομία, καθώς τα δεδομένα μπορούν να χρησιμοποιηθούν για τη δημιουργία νέων εφαρμογών ή για τη βελτίωση των υπαρχουσών.

Η χρήση του GraphDB στην αρχιτεκτονική μας αποτελεί την επιτομή της δυνατότητας συγχώνευσης δημόσιων υπηρεσιών με το LOD. Εξασφαλίζει ότι τα δεδομένα που παράγονται από τις δημόσιες υπηρεσίες μπορούν να αποθηκευτούν, να διαχειρίζονται και να δημοσιεύονται αποτελεσματικά. Αυτό το σύστημα δεν είναι μόνο αξιόπιστο και αποτελεσματικό, αλλά ενισχύει επίσης τη χρηστικότητα των δεδομένων, καθιστώντας το πολύτιμο εργαλείο για την προώθηση της καινοτομίας και της διαφάνειας στις δημόσιες υπηρεσίες.

4. Τεχνική Υλοποίηση έργου

4.1 Ανάλυση του API

Το Web Service για την εξαγωγή / παροχή δεδομένων από το Εθνικό Μητρώο Διοικητικών Διαδικασιών ακολουθεί τις αρχιτεκτονικές ιδιότητες και την δομή REST (REpresentational State Transfer) υπό τις προδιαγραφές του openAPI 3.0.

Η λειτουργία του χαρακτηρίζεται από την επιστροφή δεδομένων σχετικά με τις διοικητικές διαδικασίες σε ερωτήματα / αιτήματα HTTP, τα οποία οι χρήστες αποστέλλουν κατάλληλα δομημένα και διαμορφωμένα.

Διέπετε από 5 HTTP GET μεθόδους που είναι υπεύθυνες για να επιστρέψουν στα αιτήματα των χρηστών το κατάλληλο αποτέλεσμα με βάση τις παραμέτρους που έχουν θέσει οι προαναφερόμενοι.

Μέθοδος /Services:

Η μέθοδος αυτή είναι υπεύθυνη για την επιστροφή πληροφοριών για όλες τις διαδικασίες.

Μέθοδος /Services/ask

Η μέθοδος αυτή είναι υπεύθυνη για την αναζήτηση ιδιότητας στις διαδικασίες

Μέθοδος /Services/id/{id}

Η μέθοδος αυτή είναι υπεύθυνη για την επιστροφή πληροφοριών της συγκεκριμένης διαδικασίας που έχουμε αναζητήσει με βάση τον κωδικό της

π.χ 190794

Μέθοδος /Services/name/{name}

Η μέθοδος αυτή είναι υπεύθυνη για την επιστροφή πληροφοριών της συγκεκριμένης διαδικασίας που έχουμε αναζητήσει με βάση το όνομα της

π.χ Erasmus+ στους τομείς της εκπαίδευσης

Μέθοδος /Services/uuid/{uuid}

Η μέθοδος αυτή είναι υπεύθυνη για την επιστροφή πληροφοριών της συγκεκριμένης διαδικασίας που έχουμε αναζητήσει με βάση του μοναδικού αναγνωριστικούς της

π.χ 79d7a540-9a89-4f17-8701-21ce6c6bc456

Κάθε μέθοδος μπορεί να κληθεί με διάφορες παραμέτρους για να μας επιστραφεί το κατάλληλο αποτέλεσμα.

Για τις μεθόδους / {id} / {name} / {uuid}

έχουμε 2 κοινές παραμέτρους, την **BPMN** και την **English**.

Η BPMN μας επιστρέφει την το XML του διαγράμματος BPMN της διαδικασίας, ενώ η ENGLISH μας επιστρέφει την διαδικασία στα αγγλικά.

Η τελευταία (μη-κοινή) παράμετρος είναι ξεχωριστή για κάθε μία από τις μεθόδους και απευθύνεται στο κλειδί αναγνώρισης της διαδικασίας. όνομα, κωδικός και μοναδικό αναγνωριστικό της διαδικασίας.

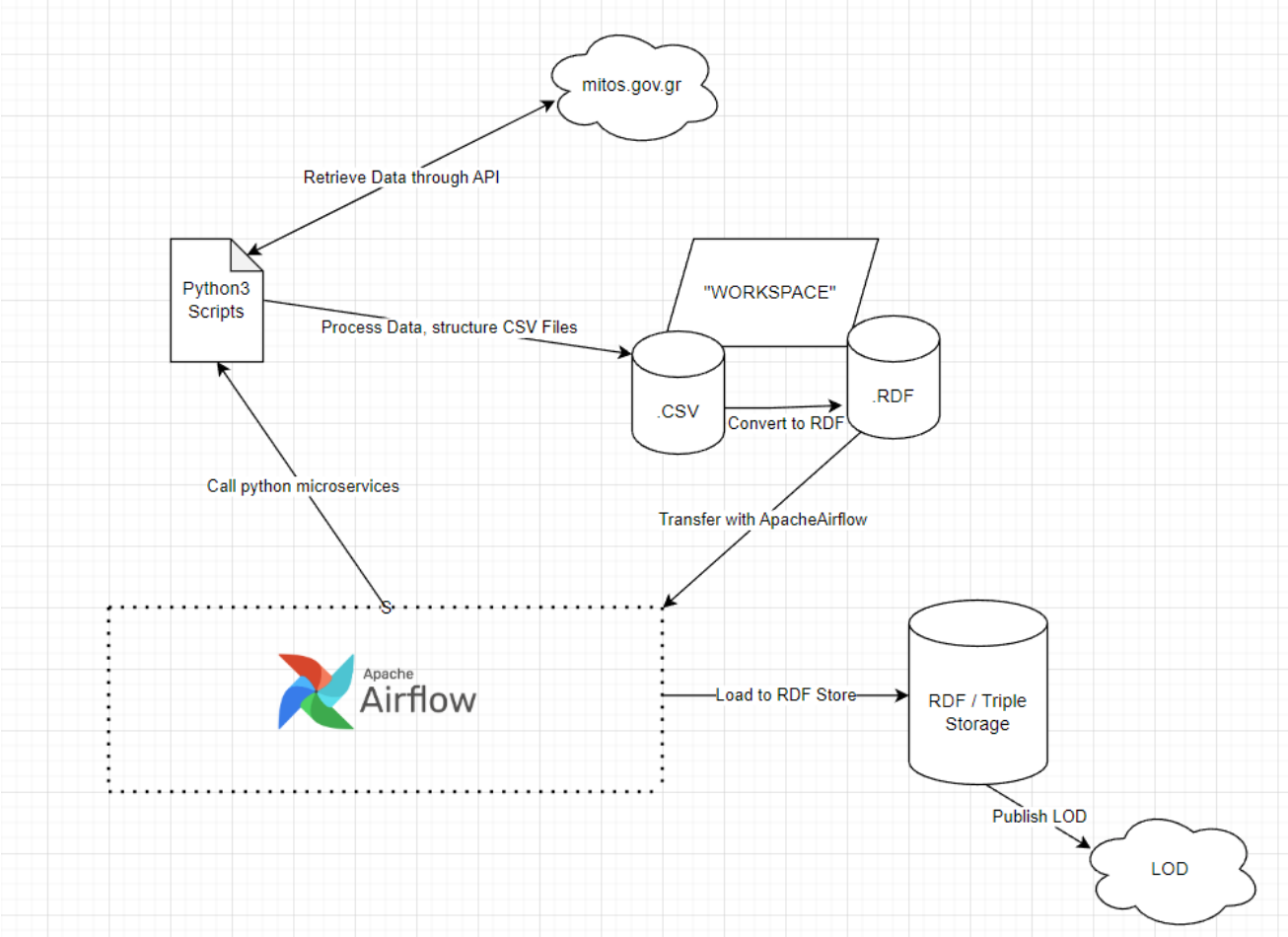
Στην μέθοδο /ask παρατηρούμε 3 επιπλέον παραμέτρους, την params, την namespace και την properties.

Στην params μπορούμε να δώσουμε ως “κλειδί” την ιδιότητα της διαδικασίας και το αποτέλεσμα της που ψάχνουμε. *π.χ Process source::EU-GO*

Στην namespace και στην properties να συγκεκριμενοποιήσουμε ποιες από τις ιδιότητες θέλουμε να μας επιστραφούν (αν όχι όλες).

(Εισαγωγή screenshots για το σχεματικό)

4.2 Ανάλυση Αρχιτεκτονικής του Workflow



Εικόνα 3. Ειδικό Μοντέλο ροής της αρχιτεκτονικής με αναφορά σε τεχνολογίες

Στην παραπάνω εικόνα αποτυπώνεται η οπτικοποίηση της αρχιτεκτονικής σε σχηματική αναπαράσταση. Στην συγκεκριμένη σχηματική αναπαράσταση είναι παρούσες οι διαδικασίες οι οποίες είναι υπεύθυνες για να φέρουν εις πέρας τις διεργασίες, οι οποίες είναι απαραίτητες για να επιτευχθεί ο τελικός μας σκοπός.

Συλλογή, Μελέτη, Επεξεργασία, Μοντελοποίηση, Μεταφορά και Αποθήκευση, Δημοσίευση είναι οι βασικές διαδικασίες οι οποίες συναντούμε όταν μελετάμε το σχηματικό.

Η διαδικασία του να αναλύσουμε την κάθε μία ξεχωριστά είναι μια απαραίτητη προϋπόθεση προκειμένου να μπορεί να αποσαφηνιστεί η σημασία για τους αναγνώστες της παρούσας διπλωματικής εργασίας.

Συνολικά, η αρχιτεκτονική που περιγράφουμε φαίνεται να είναι μια ισχυρή και αποτελεσματική λύση για εξαγωγή δεδομένων, μετασχηματισμό, αποθήκευση και δημοσίευση. Η αξιοποίηση της Python για εξαγωγή και μετασχηματισμό δεδομένων είναι μια καλή επιλογή λόγω της εκτεταμένης υποστήριξης των βιβλιοθηκών και της αναγνωσιμότητάς της. Τα REST API παρέχουν ένα επεκτάσιμο και τυπικό μέσο εξαγωγής δεδομένων. Η αποθήκευση των πρωτογενών δεδομένων σε αρχεία csv χρησιμεύει ως ένα χρήσιμο ενδιάμεσο στάδιο για επιθεώρηση και πιθανή διόρθωση σφαλμάτων. Επιπλέον, η μετατροπή των δεδομένων σε μορφές RDF ευθυγραμμίζεται καλά με τις αρχές του Σημασιολογικού Ιστού.

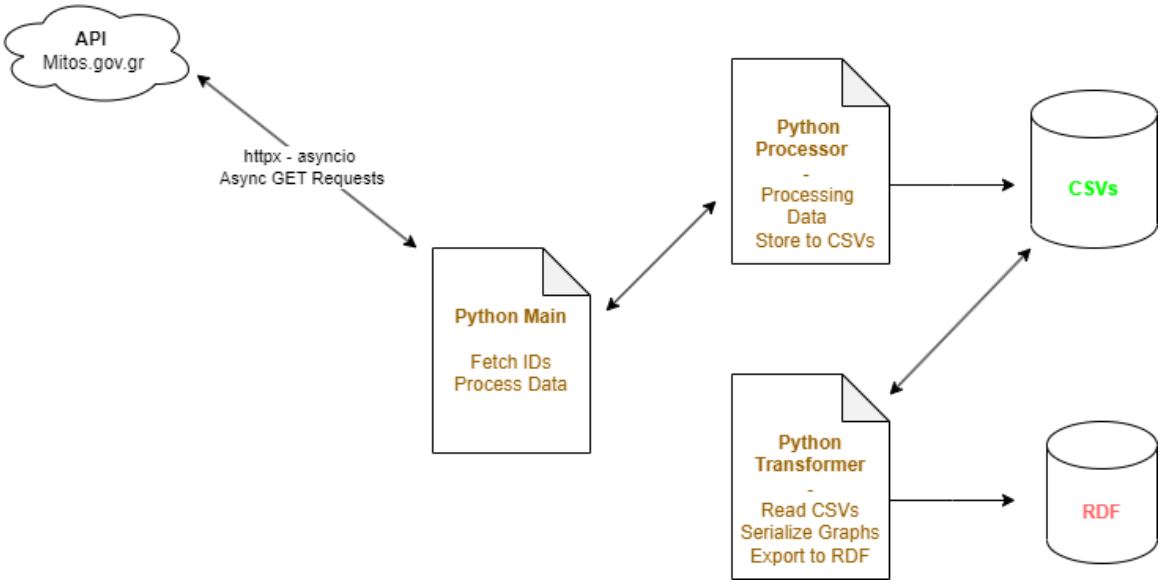
Η επιλογή του GraphDB ως μέσο αποθήκευσης RDF είναι εξαιρετική, λαμβάνοντας υπόψη την απόδοσή του, την επεκτασιμότητα και την υποστήριξή του για το SPARQL, το οποίο είναι βασικό για την αναζήτηση δεδομένων και την ανάκτηση σε σημασιολογικό πλαίσιο. Η ικανότητα του GraphDB να εκθέτει ένα τελικό σημείο SPARQL μας επιτρέπει επίσης να δημοσιεύσουμε αποτελεσματικά τα δεδομένα μας στον Σημασιολογικό Ιστό.

Η ενορχήστρωση ολόκληρης της διαδικασίας με το Apache Airflow είναι μια λογική προσέγγιση. Το Airflow προσφέρει ισχυρές δυνατότητες προγραμματισμού και παρακολούθησης που είναι ζωτικής σημασίας σε έναν αγωγό δεδομένων. Η ικανότητά του να ορίζει σύνθετες ροές εργασίας ως κώδικα και η υποστήριξή του για τη δημιουργία κατευθυνόμενων ακυκλικών γραφημάτων (DAGs) εργασιών μας επιτρέπουν να διαχειριζόμαστε αποτελεσματικά τις εξαρτήσεις και τη σειρά εκτέλεσης.

4.3 Συλλογή Δεδομένων μέσω API

Σε αυτό το σημείο θα δούμε την χρήση python script ως ένα ευέλικτο και ισχυρό εργαλείο για εξαγωγή και μετατροπή δεδομένων από τα τελικά σημεία JSON API σε μορφή CSV. Χρησιμοποιεί πρωτίστως τις ενσωματωμένες βιβλιοθήκες της Python, όπως «**requests**», «**json**» και «**csv**», γεγονός που εξασφαλίζει ευρεία συμβατότητα και ευκολία στη χρήση αλλά και τις «**httpx**» και «**asyncio**» που μας προσφέρουν τις λειτουργίες ασύγχρονων λειτουργιών στο προγραμματισμό.

Η βασική αρχή της αρχιτεκτονικής ως λογική εκτέλεσης των διαδικασιών περιγράφεται από το εξής μοντέλο.



Εικόνα 4. Μοντέλο ροής της εκτέλεσης των διαδικασιών της αρχιτεκτονικής

Στον πυρήνα του μοντέλου βρίσκονται οι τρεις κύριες λειτουργίες του:

1. Η **main** λειτουργία γραμμένη σε python,
2. Ο **Processor** γραμμένος σε python
3. Και ο **Transformer**, επίσης γραμμένος σε python.

Η **main** είναι υπεύθυνη για την επιτέλεση του βασικού κορμού της διαδικασίας.

Ακολουθεί ο κώδικας και παρακάτω η ανάλυση αυτού

```
import asyncio
import httpx
import csv
from processorMain import extract_data_for_csv, save_to_csv

BASE_URL = "https://api.digigov.grnet.gr/v1/services"

async def fetch_ids(page=1, limit=100):
    async with httpx.AsyncClient() as client:
```

```
        params = {
            "page": page,
            "limit": limit
        }
        resp = await client.get(BASE_URL, params=params)
        resp.raise_for_status()

        # Check if response is JSON
        if resp.headers.get('content-type') == 'application/json':
            json_data = resp.json()
            # Extract IDs from the nested 'data' key
            if 'data' in json_data and isinstance(json_data['data'], list):
                return [item['id'] for item in json_data['data']]
            else:
                print("Unexpected JSON structure:", json_data)
                return []
        else:
            print("Received non-JSON response:", resp.text)
            return []

async def fetch_data_from_url_async(url):
    """Fetches data from the given URL using httpx asynchronously."""
    async with httpx.AsyncClient() as client:
        resp = await client.get(url)
        resp.raise_for_status()
        return resp.json()

async def main_exec_async(ids, keys, output_filename):
    base_url = "https://api.digigov.grnet.gr/v1/services/"

    all_extracted_data = []

    for id_ in ids:
        full_url = base_url + id_
        print(f"Fetching data from URL: {full_url}")  # Debug statement
        data = await fetch_data_from_url_async(full_url)

        extracted_rows = extract_data_for_csv(data, keys)

        all_extracted_data.extend(extracted_rows)

    print(f"Saving {len(all_extracted_data)} rows to {output_filename}")
    save_to_csv(output_filename, keys, all_extracted_data)

async def process_csvs_async(ids):
    #print("Processing CSV1...")  # Debug statement
```

```
    properties =
["id","uuid","status","estimated_implementation_time","provided_language","
org_owner","provision_org","output_type","cost_min","cost_max","life_events
","alternative_titles","official_title","description","last_updated"]
    await main_exec_async(ids, properties, 'ProcessGeneral.csv')

    #print("Processing CSV2...")  # Debug statement
    properties2 =
["id","conditions_name","conditions_num_id","conditions_type"]
    await main_exec_async(ids, properties2, 'ProcessConditions.csv')

    #print("Processing CSV3...")  # Debug statement
    properties3 =
["id","evidence_description","evidence_num_id","evidence_owner","evidence_r
elated_url","evidence_type"]
    await main_exec_async(ids, properties3, 'ProcessEvidence.csv')

    print("Processing CSV4...")  # Debug statement
    properties4 = ["id","rule_ada","rule_description"]
    await main_exec_async(ids, properties4, 'ProcessRules.csv')

async def main_async():
    ids = await fetch_ids()
    print("Fetched IDs:", ids);
    await process_csvs_async(ids)

if __name__ == "__main__":
    asyncio.run(main_async())
```

Δομικά, ο σκελετός της είναι ο ακόλουθος:

1) Imports

```
import asyncio
import httpx
import csv
from processorMain import extract_data_for_csv, save_to_csv
```

asyncio – Μας επιτρέπει την σύνταξη κώδικα που μπορεί να εκτελείται παράλληλα / ταυτόχρονα χρησιμοποιώντας σύνταξη με χρήση της async/await

httpx – Ασύγχρονος HTTP Client, που μας επιτρέπει να κάνουμε κλήσεις (ασύγχρονες).

csv – Βιβλιοθήκη που μας επιτρέπει να διαβάζουμε και να γράφουμε σε αρχεία CSV

processorMain – το Import της δεύτερη μας λειτουργίας που περιέχει τον κώδικα της επεξεργασίας των δεδομένων και της αποθήκευσης τους.

2) Constants

BASE_URL: Το βασικό URL για το endpoint του API του Mitos

```
BASE_URL = "https://api.digigov.grnet.gr/v1/services"
```

3) Functions

1. **fetch_ids(page=1, limit=100)**

   Η συγκεκριμένη συνάρτηση είναι μια ασύγχρονη συνάρτηση η οποία είναι υπεύθυνη για την λήψη των IDs των διαδικασιών. Αποστέλλει ένα GET Request με συγκεκριμένες παραμέτρους σελιδοποίησης (αριθμός σελίδα και όριο αποτελεσμάτων), κοιτάει το επιστρεφόμενο αποτέλεσμα σε μορφή JSON και παραλαμβάνει τα IDs των διαδικασιών.

2. **Fetch_data_from_url_async(url)**

   Ασύγχρονη συνάρτηση που λαμβάνει δεδομένα από δοθέν URL με την χρήση της httpx. Ουσιαστικά λαμβάνει τα δεδομένα για την κάθε διαδικασία σε μορφή JSON όπως αυτά επιστρέφονται από το API.

3. **main_exec_async(ids, keys, output_filename)**
   Συνάρτηση που επεξεργάζεται τα δεδομένα για τα προσφερόμενα σε αυτήν Ids. Λαμβάνει τα δεδομένα για κάθε ID, εξάγει τα σχετικά με την κλήση δεδομένα, τα δεδομένα δηλαδή που περιγράφονται από την δήλωση των κλειδιών (keys) και τα αποθηκεύει σε CSV αρχεία.
4. **process_csvs_async(ids)**
   Επεξεργάζεται τα δεδομένα για πολλαπλά CSV αρχεία βάση των διαφορετικών ιδιοτήτων (ή κλειδιών) και τα αποθηκεύει.
   Οι ιδιότητες ή κλειδιά για κάθε CSV και τα ονόματα των παραγόμενων αρχείων CSV είναι ήδη προ-διατυπωμένα από τον προγραμματιστή.

5. **main_async()**

   Η κύρια συνάρτηση στο script της python που ενορχηστρώνει όλη την διαδικασία συνολικά

Ο python **processor** είναι υπεύθυνος προκειμένου να επεξεργαστεί και να αποθηκεύσει τα δεδομένα. Ουσιαστικά περιέχει τον κώδικα που είναι οι βασικές λειτουργίες ενσωματωμένες μέσα στις μεγαλύτερες συναρτήσεις, που καλούνται από την προηγούμενη λειτουργία που αναλύσαμε.

Ακολουθεί ο κώδικας και παρακάτω η ανάλυση αυτού

```
import requests
import csv

def fetch_data_from_url(url):
    """Fetches data from the given URL."""
    response = requests.get(url)
    response.raise_for_status()
    return response.json()

def sanitize_data(data):
    """Remove or replace problematic characters from data."""
    if isinstance(data, str):
        # Replace newline characters with a space (or remove them
altogether)
        data = data.replace('\n', ' ')
        data = data.replace('*', ' ')  # Replace with space
        return data
    return data

def find_key_recursive(data, target_key):
    """Recursively search for all instances of a key in a nested dictionary
or list."""
    if isinstance(data, dict):
        if target_key in data:
            yield data[target_key]

        for key, value in data.items():
            yield from find_key_recursive(value, target_key)

    elif isinstance(data, list):
```

```
        for item in data:
            yield from find_key_recursive(item, target_key)


def save_to_csv(filename, headers, all_data):
    """Saves the extracted data to a CSV file with a custom delimiter."""
    with open(filename, 'w', newline='') as csvfile:
        csvwriter = csv.writer(csvfile, delimiter='*')  # Setting custom
delimiter here
        csvwriter.writerow(headers)
        for data_row in all_data:
            sanitized_row = [sanitize_data(data) for data in data_row]
            csvwriter.writerow(sanitized_row)

def extract_data_for_csv(data, keys):
    """Further refined function to extract data for CSV based on provided
keys.
    Handles any key that corresponds to a list of dictionaries, ensuring
each dictionary results in a separate row along with corresponding data.
    """
    # Helper function to fetch all values for a key
    def get_all_values(data, key):
        return list(find_key_recursive(data, key))

    # First, gather all the data for each key
    all_data = {key: get_all_values(data, key) for key in keys}

    # Find the max length of data among all keys to determine number of
rows
    max_rows = max(len(values) for values in all_data.values())

    # Construct the rows
    extracted_rows = []
    for i in range(max_rows):
        row = []
        for key in keys:
            values = all_data[key]
            if i < len(values):
                row.append(values[i])
            else:
                # If a key doesn't have a value for this row, append its
most recent value
                row.append(values[-1] if values else None)
        extracted_rows.append(row)

    return extracted_rows
```

```
def main_exec(keys, output_filename):
    """Modified main function to handle multiple keys and output
filenames."""
    # Base URL and IDs (assuming these remain constant across all scripts)
    base_url = "https://api.digigov.grnet.gr/v1/services/"

    all_extracted_data = []

    # Fetch and extract data for each ID
    for id_ in ids:
        full_url = base_url + id_
        print(f"Fetching data from URL: {full_url}")  # Debug statement
        data = fetch_data_from_url(full_url)
        extracted_rows = extract_data_for_csv(data, keys)
        all_extracted_data.extend(extracted_rows)

     # Debug statement
    print(f"Saving {len(all_extracted_data)} rows to {output_filename}")

    # Save to CSV
    save_to_csv(output_filename, keys, all_extracted_data)

if __name__ == "__main__":
    process_csvs()
```

Δομικά, ο σκελετός του script είναι ο ακόλουθος:

Imports

```
import requests
import csv
```

Functions

fetch_data_from_url(url)

Παραλαμβάνει τα δεδομένα από το URL που έχει δοθεί ως παράμετρος εκτέλεσης, ελέγχει και επιστρέφει τα αντίστοιχα μηνύματα σφάλματος αν αποτύχει το Request, απαντάει με τα δεδομένα σε μορφή JSON.

sanitize_data(data)

Η συγκεκριμένη συνάρτηση είναι υπεύθυνη να καθαρίζει τα δεδομένα από προβληματικούς χαρακτήρες που μπορεί να υπάρξουν μέσα σε περιγραφές ή μεταβλητές μεγάλου όγκου που μπορεί να οδηγήσουν σε πρόβλημα κατά την διαδικασία γραμμογράφησης.

find_key_recursive(data, target_key)

Αναζητεί όλες τις εμφανίσεις των δηλωθέν κλειδιών σε ένθετα και μη λεξικά και λίστες.

save_to_csv(filename, headers, all_data)

Αποθηκεύει τα εξαγόμενα δεδομένα στο αρχείο CSV. Χρησιμοποιούμε custom delimiter (*) τον αστερίσκο γιατί δεν εμφανίζεται στα αποτελέσματα ώστε να μπορέσουμε να τα αποθηκεύσουμε στα αρχεία CSV. Επίσης ‘καθαρίζει’ κάθε δεδομένο και κατασκευάσει την γραμμογράφηση προκειμένου να μην δημιουργηθεί θέμα κατά την μεταφορά στο αρχείο.

extract_data_for_csv(data, keys)

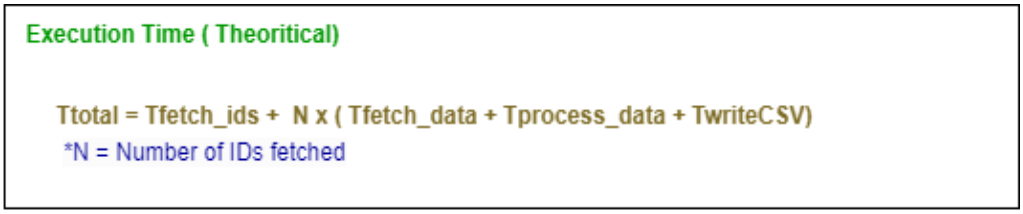
Εξάγει τα δεδομένα για το αρχείο CSV βάση των δοθέντων κλειδιών. Είναι σχεδιασμένη έτσι ώστε να μπορεί να χρησιμοποιεί τα κλειδιά για τις αντίστοιχες λίστες των λεξικών. Για κάθε κλειδί, μαζεύει όλα τα απαραίτητα δεδομένα και έπειτα κατασκευάσει την γραμμογράφηση για το CSV.

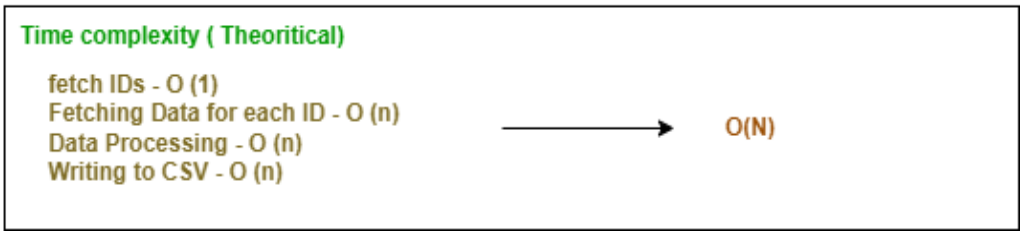
main_exec(keys, output_filename)

Είναι υπεύθυνη για την εκτέλεση των λειτουργιών που λαμβάνουν και εξάγουν δεδομένα για μια δηλωθέν λίστα IDs βάση των δοθέντων κλειδιών, καθώς και την αποθήκευση τους στα αρχεία CSV.

Σημαντική ανάλυση που επίσης χρήζει αναφοράς είναι αυτή του κόστους της εκτελεσιμότητας των διαδικασιών.

Μελετώντας τις προγραμματιστικές πράξεις που εκτελούν οι διαδικασίες καταλήγουμε στο θεωρητικό συμπέρασμα του ότι η πολυπλοκότητα μας είναι γραμμικού επιπέδου.

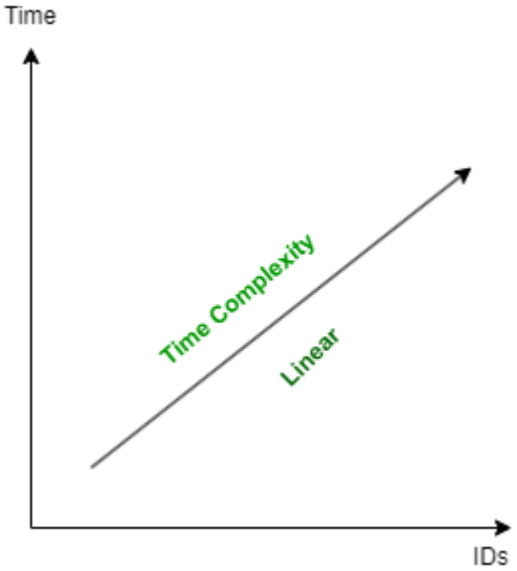




Εικόνες 5 & 6 . Θεωρητικοί Χρόνοι Εκτέλεσης και Πολυπλοκότητες.

Επομένως όσον αυξάνεται ο αριθμός των ID των διαδικασιών που θέλουμε να επεξεργαστούμε, τόσο θα αυξάνεται και ο χρόνος εκτέλεσης.

Προφανώς, όπως αναφέραμε προηγούμενος αποτελεί την θεωρητική προσέγγιση, διότι το πραγματικά αποτέλεσμα επηρεάζεται από μία σειρά μεταβλητών που δεν είναι μετρήσιμες πάντα όπως, το δίκτυο το οποίο επιτελείται η διαδικασία, το μηχάνημα που την επιτελεί και τις δυνατότητες του (π.χ επεξεργαστική ισχύ), την συνολικότερη υποδομή του Infrastructure που φιλοξενεί το API κ.ά.



Εικόνα 7. Σχηματική αναπαράσταση της πολυπλοκότητας σε συσχέτιση με τα ID και τον χρόνο εκτέλεσης.

Ο κώδικας έχει σχεδιαστεί με γνώμονα την αρθρωτότητα (modularity) και τη λειτουργικότητα, καθιστώντας τον προσαρμόσιμο σε διαφορετικά API και δομές δεδομένων. Ωστόσο, το σύνολο των ιδιοτήτων που στοχεύει το σενάριο είναι κωδικοποιημένο, γεγονός που μπορεί να περιορίσει την προσαρμοστικότητά του. Για να βελτιωθεί η χρηστικότητα, η λίστα των ιδιοτήτων θα μπορούσε να εξωτερικευτεί, επιτρέποντας στους χρήστες να καθορίσουν ποιες ιδιότητες θέλουν να εξαγάγουν χωρίς να τροποποιήσουν το ίδιο το σενάριο.

Όσον αφορά την αναγνωσιμότητα, ο κώδικας είναι καλά δομημένος και σαφώς κατακερματισμένος σε διακριτές λειτουργίες, ενισχύοντας τη δυνατότητα συντήρησης και την ευκολία κατανόησης. Επιπλέον, το σενάριο περιλαμβάνει δηλώσεις εκτύπωσης που ενημερώνουν τον χρήστη για την πρόοδο και την έξοδο της διαδικασίας εξαγωγής δεδομένων.

Συνολικά, ο κώδικας είναι φιλικός προς το χρήστη, λειτουργικός και προσαρμόσιμος, καθιστώντας τον ένα αποτελεσματικό εργαλείο για την εξαγωγή δεδομένων από JSON API και τη μετατροπή σε μορφή CSV.

4.4 Προετοιμασία & Μετατροπή Δεδομένων σε RDF

4.4.1 Προετοιμασία Διαδικασιών

Στην παρούσα διπλωματική εργασία έχουμε εργαστεί με την τελευταία έκδοση του λεξιλογίου CPSV-AP, v3.0.0.

Αρχικά, μετά από έρευνα και λεπτομερή μελέτη καταλήξαμε στους ακόλουθους συνδυασμούς προετοιμασίας των services προς επεξεργασία.

Παρουσιάζουμε τις αναφορές και τις αντιστοιχήσεις μεταξύ της διαθέσιμης πληροφορίας και του λεξιλογίου οι οποίες νοηματικά χωρίζονται διάφορα γκρουπ.

Τα γκρουπ τα οποία ερευνούμε είναι:

General *- Γενικές Πληροφορίες Διαδικασίας*

Conditions *– Προϋποθέσεις Διαδικασίας*

Evidences *– Δικαιολογητικά που αφορούν την Διαδικασία*

Rules *– Νομοθεσία που αφορά την Διαδικασία*

General

Διαθέσιμη Πληροφορία μέσω API	**CPSV:AP**
ID	identifier cpsv::PublicService
STATUS	skos:Concept
ESTIMATED_IMPLEMENTATION_TIME	processing_time
PROVIDED_LANGUAGE	dcterms:language
OUTPUT_TYPE	cpsv:output / cpsv:produces
COST_MIN	cv:Cost
COST_MAX	cv:Cost

ALTERNATIVE_TITLES	skos:altLabel
OFFICIAL_TITLE	dct:title
DESCRIPTION	dct:description
LIFE_EVENTS	cv:LifeEvent

Το πεδίο **LIFE_EVENTS** περιγράφεται από την κλάση cv:LifeEvent και συνδέεται με την διαδικασία στο cpsv::PublicService μέσω του property isGroupedBy.

Το πεδίο **ID** αφορά τον μοναδικό κωδικό της διαδικασίας και παίρνει θέση identifier στην κλάση cpsv::PublicService.

Το πεδίο **OFFICIAL_TITLE** αφορά τον επίσημο τίτλο της διαδικασίας, επομένως περιγράφεται από το dct:title στην cpsv::PublicService

Το πεδίο **PROVIDED_LANGUAGE** εξηγεί τις προσφερόμενες γλώσσες για την διαδικασία. Περιγράφεται από το dcterms:language property και ανήκει στην cpsv::PublicService.

Τα πεδία **COST_MIN & COST_MAX** αναφέρονται αντίστοιχα στην ελάχιστη και μέγιστη χρέωση που υπάρχει για την διαδικασία. Περιγράφονται από την κλάση cv:Cost και συνδέονται στην cpsv::PublicService μέσω του hasCost property.

Το πεδίο **OUTPUT_TYPE** εξηγεί τα εξαρχόμενα της διαδικασίας και περιγράφεται από την cpsv:Output και συνδέεται με την cpsv::PublicService μέσω του cpsv:Produces

Το πεδίο **ALTERNARTIVE_TITLES** αφορά τους εναλλακτικούς τίτλους της διαδικασίας και περιγράφεται μέσω του skos:altLabel property στην cpsv::PublicService

Το πεδίο **DESCRIPTION** αφορά την περιγραφή της διαδικασίας, και περιγράφεται μέσω του dct:description στην cpsv::PublicService

Το πεδίο **STATUS** αφορά την κατάσταση της διαδικασίας στο σύστημα και περιγράφεται από την skos:Concept ενώ συνδέεται με την cpsv:PublicService μέσω του isClassifiedBy property.

Το πεδίο **ESTIMATED_IMPLEMENTATION_TIME** αφορά τον χρόνο υλοποίησης της διαδικασίας και περιγράφεται από την cpsv:processingTime που βρίσκεται στην cpsv::PublicService.

Conditions

Διαθέσιμη Πληροφορία μέσω API	**CPSV:AP**
CONDITIONS_NAME	dct:title
CONDITIONS_NUM_ID	cv:identifier
CONDITIONS_TYPE	processing_time

Το πεδίο **CONDITIONS_NAME** αναφέρεται στο όνομα της συγκεκριμένης προϋπόθεσης της διαδικασίας, το **CONDITIONS_NUM_ID** στον αύξων αριθμό της σειράς εμφάνισης της προϋπόθεσης, ενώ το **CONDITIONS_TYPE** στον τύπο της κατηγοριοποίησης της προϋπόθεσης.

Evidences

Διαθέσιμη Πληροφορία μέσω API	**CPSV:AP**
EVIDENCE_DESCRIPTION	Identifier cpsv::PublicService
EVIDENCE_NUM_ID	cv:identifier
EVIDENCE_OWNER	processing_time
EVIDENCE_RELATED_URL	dcterms:language
EVIDENCE_TYPE	cpsv:output / cpsv:produces

Το πεδίο **EVIDENCE_DESCRIPTION** περιγράφει αναλυτικά το κάθε δικαιολογητικό, το **EVIDENCE_NUM_ID** υποδεικνύει τον αύξων αριθμό της σειράς εμφάνισης, το

EVIDENCE_OWNER αναφέρεται στην οντότητα του υπεύθυνου κατάθεσης του δικαιολογητικού, το **EVIDENCE_TYPE** περιέχει τον κωδικό είδους του δικαιολογητικού ενώ το **EVIDENCE_RELATED_URL** περιέχει τον σχετικό υπερ-σύνδεσμο που αφορά το δικαιολογητικό.

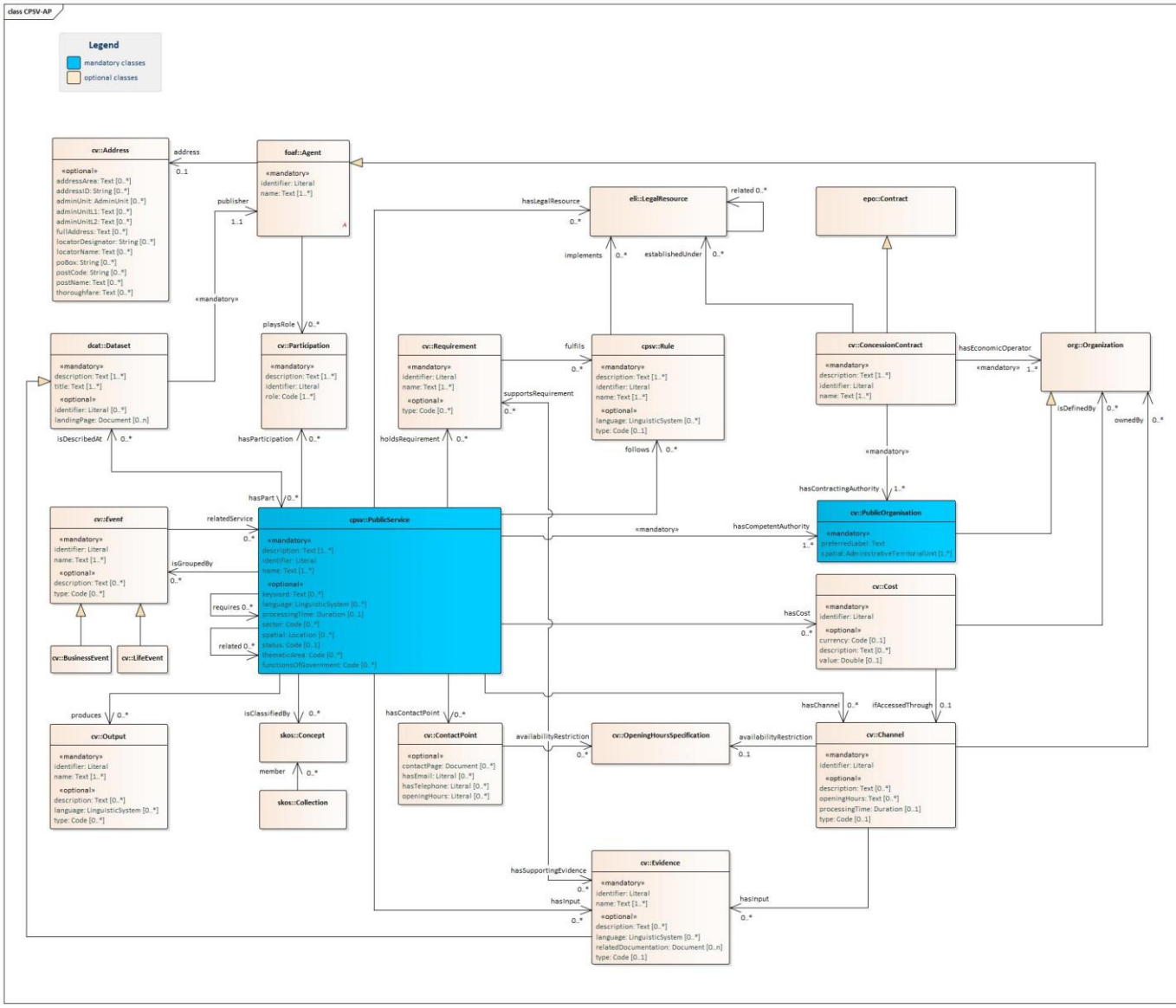
Rules

Διαθέσιμη Πληροφορία μέσω API	**CPSV:AP**
RULE_ADA	dct:identifier
RULE_DESCRIPTION	dct:description

Το πεδίο **RULE_ADA** περιέχει τον αριθμό διαδικτυακής ανάρτησης της σχετικής νομοθεσίας και αποτελεί μοναδικό αναγνωριστικό ενώ το πεδίο **RULE_DESCRIPTION** την περιγράφει αναλυτικά.

Σε αυτό το σημείο είναι σημαντικό να αναφέρουμε ότι η κλάση που περιγράφει την νομοθεσία είναι η LegalResource από το λεξιλόγιο «eli» που χρησιμοποιείται για να περιγράψει μετα-δεδομένα που αφορούν νομικό και ρυθμιστικό περιεχόμενο, ενώ συνδέεται με την cpsv::PublicService μέσω του property cv:hasLegalResource.

Τα δικαιολογητικά τα ορίζουμε με την κλάση cv:Evidence ενώ χρησιμοποιείται το property cv:holdsRequirement για να τα συνδέσει με την cpsv::PublicService.



Εικόνα 8. Διάγραμμα UML της έκδοσης 3.0.0 του CPSV-AP.
Πηγή: https://semiceu.github.io/CPSV-AP/releases/3.0.0/html/overview.jpg

4.4.2 Μετατροπή Δεδομένων

Το παρεχόμενο script έχει σχεδιαστεί για να εκτελεί έναν μετασχηματισμό από μια μορφή δεδομένων CSV σε RDF (Πλαίσιο Περιγραφής Πόρων) χρησιμοποιώντας τη σύνταξη Turtle, η οποία είναι μια κοινή διαδικασία για τη δημιουργία Συνδεδεμένων Δεδομένων. Το RDF είναι ένα τυπικό μοντέλο για την ανταλλαγή δεδομένων στον Ιστό. Επιτρέπει μια δομημένη, αλλά ευέλικτη αναπαράσταση δεδομένων.

Εκτός από την διπλωματική εργασία για να γίνει η ανάλυση του και να υπάρχει συνοχή στην αναγνωσιμότητα , το εν λόγω script του ‘transformer’ όπως και όλο το υλικό βρίσκεται και ηλεκτρονικά σε public repository στο GitHub εδώ.

Το script ξεκινάει εισάγοντας τις απαραίτητες βιβλιοθήκες όπως «csv» για την ανάλυση αρχείων CSV και «rdflib» για το χειρισμό RDF. Στη συνέχεια ορίζει τη διαδρομή προς το αρχείο εισόδου CSV και το επιθυμητό αρχείο εξόδου Turtle.

Μετά από αυτό, το script καθορίζει χώρους ονομάτων. Ένας χώρος ονομάτων στο RDF είναι μια συλλογή ιδιοτήτων και κλάσεων RDF που χρησιμοποιούνται συχνά μαζί και μοιράζονται το ίδιο πρόθεμα URI. Σε αυτό το σενάριο, ο χώρος ονομάτων BASE αντιπροσωπεύει τη βασική διεύθυνση URL για τους πόρους που περιγράφονται, ενώ οι χώροι ονομάτων 'cv', 'eli' και 'cpsv' είναι προσαρμοσμένοι για συγκεκριμένα λεξιλόγια που χρησιμοποιεί το σενάριο. Οι χώροι ονομάτων «DCTERMS», «RDF», «OWL», «SKOS», «RDFS» και «FOAF» είναι τυπικοί για γνωστά λεξιλόγια RDF.

Δημιουργείται ένα νέο γράφημα RDF 'g' και οι χώροι ονομάτων δεσμεύονται σε αυτό το γράφημα. Αυτό σημαίνει ότι όταν το γράφημα είναι σειριακό σε ένα αρχείο, οι χώροι ονομάτων θα συμπεριληφθούν και οι ιδιότητες και οι κλάσεις από αυτούς τους χώρους ονομάτων θα συντομεύονται χρησιμοποιώντας τα προθέματά τους.

Στη συνέχεια, το σενάριο διαβάζει το αρχείο CSV εισόδου χρησιμοποιώντας ένα DictReader, το οποίο αντιστοιχίζει τις πληροφορίες που διαβάζονται σε ένα λεξικό όπου τα κλειδιά είναι τα ονόματα των στηλών και οι τιμές είναι οι τιμές από κάθε γραμμή. Το σενάριο χρησιμοποιεί τη στήλη 'id' για να δημιουργήσει μοναδικά URIRefs για τα θέματα των τριπλών που θα προσθέσει στο γράφημα.

Στο βρόχο πάνω από τις σειρές του αρχείου CSV, το σενάριο προσθέτει ένα σύνολο τριπλετών στο γράφημα για κάθε σειρά. Κάθε τριπλέτα αποτελείται από ένα υποκείμενο (το URIRef με βάση το αναγνωριστικό της σειράς), ένα κατηγορούμενο (μια ιδιότητα από έναν από τους χώρους ονομάτων) και ένα αντικείμενο (η αντίστοιχη τιμή από τη σειρά, μετασχηματισμένη σε Literal ή URIRef ανάλογα με τις ανάγκες).

Το script προσθέτει επίσης ένα rdf:type τριπλό για κάθε σειρά, υποδεικνύοντας ότι το θέμα είναι τύπου "PublicService" από τον χώρο ονομάτων "cpsv".

Τέλος, το script σειριοποιεί το γράφημα (δηλαδή το μετατρέπει σε αναπαράσταση κειμένου) σε μορφή RDF/XML και το αποθηκεύει στο αρχείο εξόδου. Η μορφή 'turtle' θα μπορούσε επίσης να χρησιμοποιηθεί εδώ, αλλά το σενάριο χρησιμοποιεί 'xml' για ευρύτερη συμβατότητα. Αυτό το σειριακό αρχείο μπορεί να κοινοποιηθεί ή να δημοσιευτεί.

Επίσης άλλες εφαρμογές μπορούν να το αναλύσουν ξανά σε ένα γράφημα RDF για περαιτέρω επεξεργασία ή αναζήτηση.

Παράθεση κώδικα python υπεύθυνο για την λειτουργία αυτή

```
import pandas as pd
import json

base_url = 'https://mitos.gov.gr:8890'

def is_valid(value):
    return bool(value) and str(value).lower() != 'nan'

def preprocess_life_events(life_events_data):
    mapping = {}
    current_id = 0

    for event_str in life_events_data:
        event_str = str(event_str)  # Convert event_str to string if it
isn't already
        # Cleaning up the string and splitting by comma
        events = event_str.replace("[", "").replace("]",
"").strip().split(',')

        for event in events:
            event = event.strip().replace("'", "")  # Removing quotes and
spaces

            # If the event is not in the mapping, add it with a new ID
            if event not in mapping:
                mapping[event] = current_id
                current_id += 1
    return mapping


def preprocess_conditions(df):
    """
    Processes the conditions dataframe to create a JSON mapping with
conditions_name as the primary key.
    Assigns an incrementing id value for each unique condition, starting
from 1.
```

```
    """
    json_mapping = {}
    condition_id_counter = 1  # Initialize the counter

    # Drop duplicate rows based on the 'conditions_name' column
    df_unique_conditions = df.drop_duplicates(subset='conditions_name')

    for _, row in df_unique_conditions.iterrows():
        conditions_name = row["conditions_name"]
        json_mapping[conditions_name] = {
            "id": condition_id_counter,
            "conditions_num_id": int(row["conditions_num_id"]) if not
pd.isnull(row["conditions_num_id"]) else None,
            "conditions_type": row["conditions_type"] if not
pd.isnull(row["conditions_type"]) else None
        }
        condition_id_counter += 1  # Increment the counter

    return json_mapping



def generate_triples(resource_id, uuid, title, description,
provided_language, cost_min, cost_max, output_type, life_events,
alternative_titles, df_conditions, df_evidences, df_rules):
    triples = []
    triples.append(f"<{base_url}/id/ps/{resource_id}> a cpsv:PublicService
;")
    triples.append(f'  dct:identifier "{uuid}" ;')
    triples.append(f'  dct:title "{title}" ;')
    if is_valid(alternative_titles):
        triples.append(f'  skos:altLabel """{alternative_titles}""" ;')
    if is_valid(description):
        triples.append(f'  dct:description """{description}""" ;')

    # Check if cost_min and cost_max are not null and add the cv:hasCost
triple
    if not pd.isna(cost_min) and not pd.isna(cost_max):
        triples.append(f' cv:hasCost
<{base_url}/PublicServices/id/cost/cost_max/cost{resource_id}> ;')
        triples.append(f' cv:hasCost
<{base_url}/PublicServices/id/cost/cost_min/cost{resource_id}> ;')

    # Check provided_language and add the corresponding triple(s), if not
empty
    if pd.notna(provided_language):
        languages = provided_language.split(',')
```

```
        for language in languages:
            language = language.strip()  # Remove any leading/trailing
whitespace
            triples.append(f' dct:language "{language}" ;')

    if df_conditions is not None:
        matched_conditions = df_conditions[df_conditions['id'] ==
resource_id]['conditions_name']
    for condition_name in matched_conditions:
        if condition_name in conditions_mapping:
            condition_id = conditions_mapping[condition_name]["id"]  #
Adjusted this line to access the id within the dictionary
            triples.append(f'
cv:holdsRequirement<{base_url}/PublicServices/id/requirement/requirement{co
ndition_id}> ;')

    # Check if life_events is filled, and if so, add the cv:isGroupedBy
triple
    condition_found_evidences = resource_id in df_evidences['id'].values
    if condition_found_evidences:
        triples.append(f' cv:hasInput
<{base_url}/PublicServices/id/evidence/evidence{resource_id}> ;')

    # Check if life_events is filled, and if so, add the cv:isGroupedBy
triple
    condition_found_rules = resource_id in df_rules['id'].values
    if condition_found_rules:
        triples.append(f' cv:hasLegalResource
<{base_url}/PublicServices/id/rule/rule{resource_id}> ;')

    # Add the cv:isGroupedBy triple based on the life_event mapping
    if pd.notna(life_events):
        life_events_list = life_events.replace("[", "").replace("]",
"").strip().split(',')
        for life_event in life_events_list:
            life_event = life_event.strip().replace("'", "")
            if life_event in life_events_mapping:
                event_id = life_events_mapping[life_event]
                triples.append(f'
cv:isGroupedBy<{base_url}/PublicServices/id/event/event{event_id}> ;')

    # Check if output_type is filled, and if so, add the cpsv:produces
triple
    if pd.notna(output_type):
        triples.append(f' cpsv:produces
<{base_url}/PublicServices/id/praxis/praxis{resource_id}> ;')
```

```
    triples.append(".")
    return triples

def append_with_check(triples_list, triple):
    """
    Append the given triple to the triples list.
    Before appending, ensure that the last appended triple (if any) ends
with a dot.
    """
    if triples_list and triples_list[-1].endswith(" ."):
        triples_list[-1] = triples_list[-1].replace(" ;", " .")

    triples_list.append(triple)

def append_additional_triples(triples_list, resource_id, cost_min,
cost_max, output_type, life_events, df_conditions, df_evidences, df_rules):
    """
    This function appends additional triples based on the conditions for a
given resource_id.

    Parameters:
    - resource_id: The ID of the resource for which to append the
conditions.
    - df_conditions: DataFrame containing the conditions data.
    - conditions_mapping: Dictionary mapping condition names to their IDs
from the JSON file.
    - base_url: Base URL for constructing the triples.

    Returns:
    - List of triples.
    """

    triples = []

    # Add new lines for cost_min and cost_max if available
    if not pd.isna(cost_min):
        triples.append(f"
<{base_url}/PublicServices/id/cost/cost_min/cost{resource_id}> a cv:Cost
;")
        triples.append(f"   cv:value {cost_min} ;")
        triples.append('    cv:currency "Euro" ;')
        triples.append('    dct:description "Min Cost" .')

    if not pd.isna(cost_max):
        triples.append(f"
<{base_url}/PublicServices/id/cost/cost_max/cost{resource_id}> a cv:Cost
;")
```

```
        triples.append(f"    cv:value {cost_max} ;")
        triples.append('    cv:currency "Euro" ;')
        triples.append('    dct:description "Max Cost" .')

    if not pd.isna(output_type):
        triples.append(f"
<{base_url}/PublicServices/id/praxis/praxis{resource_id}> a cv:Output ;")
        if is_valid(output_type):
            triples.append(f'    dct:title "{output_type}" .')


    # Check if the resource_id is present in the ProcessConditions.csv and
append cv:holdsRequirement triple
    matching_evidences = df_evidences[df_evidences['id'] == resource_id]
    for _, row in matching_evidences.iterrows():
        append_with_check(triples, f'
<{base_url}/PublicServices/id/evidence/evidence{resource_id}> a cv:Evidence
;')
        if is_valid(row["evidence_num_id"]):
            append_with_check(triples, f'    cv:identifier
"{row["evidence_num_id"]}" ;')
        if is_valid(row["evidence_description"]):
            append_with_check(triples, f'    cv:name
"{row["evidence_description"]}" ;')
        if is_valid(row["evidence_type"]):
            append_with_check(triples,(f'    dct:type
"{row["evidence_type"]}" ;'))
        if is_valid(row["evidence_related_url"]):
           append_with_check(triples, f'    cv:relatedDocumentation
"{row["evidence_related_url"]}" .')

    # Check if the resource_id is present in the ProcessConditions.csv and
append cv:holdsRequirement triple
    matching_rules = df_rules[df_rules['id'] == resource_id]
    for _, row in matching_rules.iterrows():
        triples.append(f'
<{base_url}/PublicServices/id/rule/rule{resource_id}> a eli:LegalResource
;')
        if is_valid(row["rule_ada"]):
            triples.append(f'    dct:identifier "{row["rule_ada"]}" ;')
        if is_valid(row["rule_description"]):
            triples.append(f'    dct:description "{row["rule_description"]}"
.')

    triples_list.append(triples)

# Read the CSV file using the pipe ('|') delimiter
```

```
csv_file_path = 'ProcessGeneral.csv'
df = pd.read_csv(csv_file_path, delimiter='*')

# Load the ProcessConditions.csv data
df_conditions = pd.read_csv("ProcessConditions.csv", delimiter='*')
df_conditions.columns = ['id', 'conditions_name', 'conditions_num_id',
'conditions_type']
df_conditions['id'] = df_conditions['id'].astype(int)

# Load the ProcessEvidences.csv data
df_evidences = pd.read_csv("ProcessEvidence.csv", delimiter='*')
df_evidences.columns = ['id', 'evidence_description', 'evidence_num_id',
'evidence_owner', 'evidence_related_url', 'evidence_type']
df_evidences['id'] = df_evidences['id'].astype(int)

# Load the ProcessEvidences.csv data
df_rules = pd.read_csv("ProcessRules.csv", delimiter='*')
df_rules.columns = ['id', 'rule_ada', 'rule_description']
df_rules['id'] = df_rules['id'].astype(int)

# Preprocess the life_events column and generate the mapping
life_events_mapping = preprocess_life_events(df['life_events'].tolist())
with open('life_events_mapping.json', 'w', encoding='utf-8') as json_file:
    json.dump(life_events_mapping, json_file, ensure_ascii=False, indent=4)

# Preprocess the conditions and generate the mapping
conditions_mapping = preprocess_conditions(df_conditions)
with open('conditions_mapping.json', 'w', encoding='utf-8') as json_file:
    json.dump(conditions_mapping, json_file, ensure_ascii=False, indent=4)

triples_list = [generate_triples(row['id'], row['uuid'],
row['official_title'], row['description'], row['provided_language'],
row['cost_min'], row['cost_max'], row['output_type'], row['life_events'],
row['alternative_titles'], df_conditions, df_evidences, df_rules ) for
index, row in df.iterrows()]

# Append additional triples
for index, row in df.iterrows():
    append_additional_triples(triples_list, row['id'], row['cost_min'],
row['cost_max'], row['output_type'], row['life_events'], df_conditions,
df_evidences, df_rules)

# Append triples for each unique condition from the mapping
added_conditions = set()
triples_conditions = []
for condition_name, condition_data in conditions_mapping.items():
    condition_id = condition_data["id"]
```

```
    if condition_id not in added_conditions:
        triples_conditions.append(f'
<{base_url}/PublicServices/id/requirement/requirement{condition_id}> a
cv:Requirement ;')
        if is_valid(condition_data["conditions_num_id"]):
            triples_conditions.append(f'   cv:identifier
"{condition_data["conditions_num_id"]}" ;')
        if is_valid(condition_name):
            triples_conditions.append(f'   dct:title "{condition_name}" ;')
        if is_valid(condition_data["conditions_type"]):
            triples_conditions.append(f'   dct:type
"{condition_data["conditions_type"]}" .')
        added_conditions.add(condition_id)

triples_list.append(triples_conditions)

# Append triples for each unique life event from the mapping
triples_lifeEvents = []
for life_event, event_id in life_events_mapping.items():
    triples_lifeEvents.append(f'
<{base_url}/PublicServices/id/event/event{event_id}> a cv:LifeEvent ;')
    triples_lifeEvents.append(f'   cv:name "{life_event}" .')
triples_list.append(triples_lifeEvents)

# Flatten the list of lists into a single list
triples = [triple for sublist in triples_list for triple in sublist]

# Write the triples to the TTL file
with open("generated_data.ttl", "w") as f:
    f.write("@prefix schema: <https://schema.org/> .\n")
    f.write("@prefix eli: <http://data.europa.eu/eli/ontology#> .\n")
    f.write("@prefix cv: <http://data.europa.eu/m8g/> .\n")
    f.write("@prefix adms: <http://www.w3.org/ns/adms#> .\n")
    f.write("@prefix dct: <http://purl.org/dc/terms/> .\n")
    f.write("@prefix skos: <http://www.w3.org/2004/02/skos/core#> .\n")
    f.write("@prefix dcat: <http://www.w3.org/ns/dcat#> .\n")
    f.write("@prefix locn: <http://www.w3.org/ns/locn#> .\n")
    f.write("@prefix foaf: <http://xmlns.com/foaf/0.1/> .\n")
    f.write("@prefix cpsv: <http://purl.org/vocab/cpsv#> .\n")
    f.write("\n")
    f.write("\n".join(triples))
```

4.5 Ο ρόλος του Apache Airflow σαν Orchestrator

Η χρησιμότητα του Apache Airflow σε μια αρχιτεκτονική λογισμικού, η οποία βασίζεται σε δεδομένα είναι ανεκτίμητη. Βελτιώνει τις σύνθετες διαδικασίες που εμπλέκονται στην ανάκτηση, αποθήκευση, μετασχηματισμό και δημοσίευση Συνδεδεμένων Ανοικτών Δεδομένων (LOD). Η δυνατότητα αυτοματοποίησης αυτών των εργασιών χρησιμοποιώντας το Apache Airflow μειώνει την πιθανότητα σφαλμάτων, ενισχύει την αποτελεσματικότητα και επιτρέπει την επεκτασιμότητα.

Η αρχιτεκτονική χρησιμοποιεί τα Κατευθυνόμενα Ακυκλικά Γραφήματα (DAG) του Apache Airflow για να μοντελοποιήσει τη γραμμή δεδομένων. Κάθε κόμβος στο DAG αντιπροσωπεύει μια διαφορετική εργασία που εμπλέκεται στον κύκλο ζωής των δεδομένων, όπως η ανάκτηση δεδομένων από το API του Mitos.gov.gr, η αποθήκευση αυτών των δεδομένων, η μετατροπή τους από CSV σε RDF Turtles και τελικά η δημοσίευση του LOD. Αυτή η δομημένη, αυτοματοποιημένη ακολουθία λειτουργιών ελαχιστοποιεί τη χειροκίνητη παρέμβαση και βελτιώνει τη συνολική ακεραιότητα των δεδομένων.

Επιπλέον, το Apache Airflow προσφέρει ένα ισχυρό σύστημα καταγραφής που καταγράφει την κατάσταση κάθε εργασίας σε πραγματικό χρόνο. Αυτό είναι επωφελές για σκοπούς αντιμετώπισης προβλημάτων και εντοπισμού σφαλμάτων, καθώς βοηθά στην έγκαιρη επίλυση προβλημάτων που ενδέχεται να προκύψουν. Το σύστημα καταγραφής ενισχύει επίσης τη διαφάνεια και την ιχνηλασιμότητα στη διοχέτευση δεδομένων, καθιστώντας ευκολότερη την παρακολούθηση και τον έλεγχο των διαδικασιών.

Η προσαρμοστικότητα και ο προγραμματισμός του Apache Airflow είναι επίσης σημαντικά χαρακτηριστικά. Επιτρέπουν την προσαρμογή και τη βελτιστοποίηση των ροών εργασίας, διασφαλίζοντας ότι η αρχιτεκτονική μπορεί να ανταποκριθεί αποτελεσματικά στους μεταβαλλόμενους όγκους δεδομένων και να διατηρήσει την ανθεκτικότητά της κατά τη διάρκεια πιθανών *αστοχιών κατά την εκτέλεση των διεργασιών και διαφόρων προγραμμάτων.

Ουσιαστικά, το Apache Airflow προσφέρει μια ισχυρή πλατφόρμα για τη διαχείριση πολύπλοκων ροών εργασίας δεδομένων, ενισχύοντας την απόδοση, την αξιοπιστία και τη διαφάνεια. Οι δυνατότητές του στην ενορχήστρωση εργασιών, την αυτοματοποίηση και την καταγραφή είναι ζωτικής σημασίας στο πλαίσιο μιας αρχιτεκτονικής λογισμικού που βασίζεται σε δεδομένα, μεταμορφώνοντας τον τρόπο διαχείρισης και χρήσης των δεδομένων.

**Επεξήγηση Όρου: Ως αστοχία λογισμικού μπορεί να περιγραφεί οποιαδήποτε ανεξήγητη συμπεριφορά ή η παραγωγή μη-ορθών αποτελεσμάτων σε ένα υπολογιστικό σύστημα.*

[1] Parnas, D. L., & Madey, J. (1995). 41-61. https://doi.org/10.1016/0167-6423(95)00005-I

[2] Chillarege, R., Bhandari, I. S., Chaar, J. K., Halliday, M. J., Moebus, D. S., Ray, B. K., & Wong, M. Y. (1992). Orthogonal defect classification-a concept for in-process measurements. IEEE Transactions on Software engineering, 18(11), 943-956. https://doi.org/10.1109/32.177364

4.6 Αποθήκευση και μεταφορά δεδομένων

Ως κρίσιμο στοιχείο στην αρχιτεκτονική μας, το GraphDB διαδραματίζει σημαντικό ρόλο στη διαχείριση και τον χειρισμό των δεδομένων RDF (Resource Description Framework). Αυτό είναι ιδιαίτερα σημαντικό στη ροή μας, όπου τα δεδομένα συλλέγονται αρχικά σε μορφή CSV, μετατρέπονται σε σύνταξη RDF Turtle και στη συνέχεια πρέπει να αποθηκευτούν με τρόπο που να διευκολύνει τα προηγμένα ερωτήματα και την ανάλυση.

Το GraphDB είναι επίσης συμβατό με τα πρότυπα του W3C. Μπορεί να χειριστεί μεγάλους όγκους σύνθετων σημασιολογικών ερωτημάτων στα αποθηκευμένα δεδομένα RDF, κάτι που είναι ωφέλιμο για περαιτέρω εξερεύνηση δεδομένων και εξαγωγή συμπερασμάτων. Επιπλέον, η βασική μηχανή βάσης δεδομένων του GraphDB είναι ικανή να συλλογίζεται σε κλίμακα, επιτρέποντας έτσι τη δημιουργία νέων σημασιολογικών γεγονότων με βάση προϋπάρχοντα δεδομένα - ένα χαρακτηριστικό που θα μπορούσε να βελτιώσει τις γνώσεις που προέρχονται από τα δεδομένα σας.

Στην αρχιτεκτονική μας, μετά τη μετατροπή των δεδομένων CSV σε μορφή RDF Turtle, τα δεδομένα RDF που προκύπτουν μπορούν να φορτωθούν στο GraphDB. Αυτή η διαδικασία μπορεί να αυτοματοποιηθεί χρησιμοποιώντας το REST API του GraphDB, όπου τα

δεδομένα RDF μπορούν να προστεθούν σε ένα αποθετήριο κάνοντας ένα αίτημα POST. Τα δεδομένα μπορούν επίσης να φορτωθούν χειροκίνητα χρησιμοποιώντας τη διεπαφή ιστού Workbench του GraphDB. Και στις δύο περιπτώσεις, ο χρήστης μπορεί να καθορίσει τη μορφή των εισερχόμενων δεδομένων.

Μόλις τα δεδομένα φορτωθούν στο GraphDB, αποθηκεύονται σε ένα triplestore, όπου κάθε κομμάτι δεδομένων (ή "statement") αποθηκεύεται ως τριπλό με τη μορφή {subject, predicate, object}. Αυτή η δομή αποθήκευσης επιτρέπει την αποτελεσματική αναζήτηση με χρήση της SPARQL, της τυπικής γλώσσας για την αναζήτηση δεδομένων RDF. Η ενσωμάτωση του GraphDB στην αρχιτεκτονική σας επιτρέπει έτσι εξελιγμένες λειτουργίες δεδομένων και προσφέρει μια σταθερή βάση για εφαρμογές Συνδεδεμένων Ανοικτών Δεδομένων.

4.7 Δημοσίευση των Ανοικτών Συνδεδεμένων Δεδομένων στον Σημασιολογικό Ιστό

Τα Συνδεδεμένα Ανοικτά Δεδομένα (LOD) αποτελούν αναπόσπαστο μέρος του οράματος του Σημασιολογικού Ιστού, το οποίο στοχεύει στη μετατροπή του Ιστού σε μια παγκόσμια βάση δεδομένων. Στην αρχιτεκτονική μας, το GraphDB διαδραματίζει κεντρικό ρόλο όχι μόνο στην αποθήκευση και διαχείριση των δεδομένων RDF, αλλά και στη δημοσίευση αυτών των δεδομένων στον Σημασιολογικό Ιστό.

Η βάση δεδομένων RDF της GraphDB, σε συνδυασμό με την ικανότητά της να υποστηρίζει το πρωτόκολλο SPARQL και τη γλώσσα ερωτημάτων RDF (SPARQL), παρέχει έναν αποτελεσματικό και αποδοτικό μηχανισμό για τη δημοσίευση και την κοινή χρήση δεδομένων RDF στον Ιστό. Εκθέτοντας ένα τελικό σημείο SPARQL, το GraphDB επιτρέπει σε εξωτερικές εφαρμογές και χρήστες να υποβάλλουν ερωτήματα στα δεδομένα μέσω HTTP, καθιστώντας ουσιαστικά τα δεδομένα μέρος του Σημασιολογικού Ιστού.

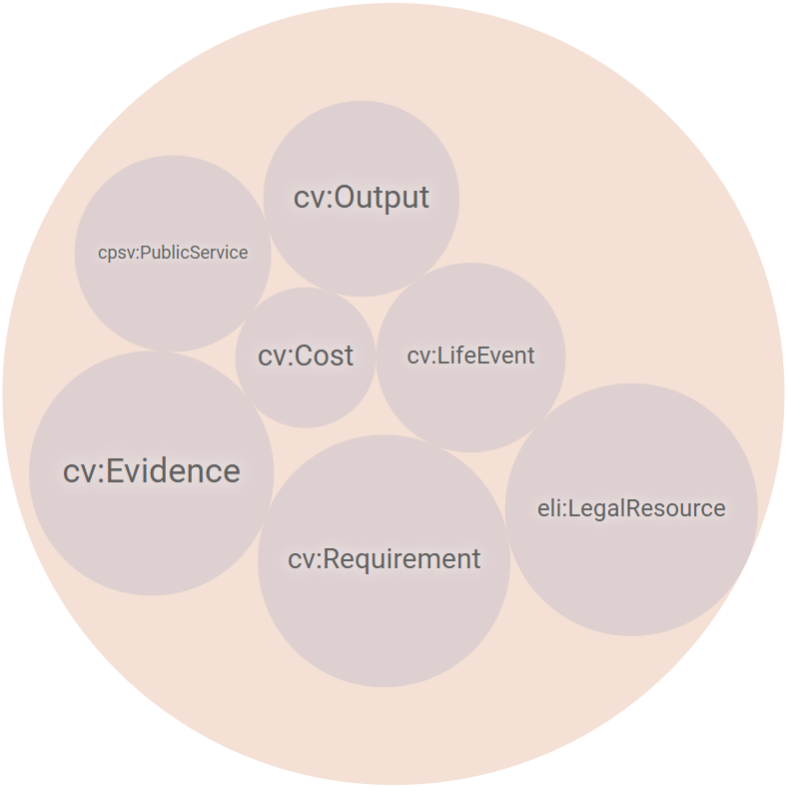
Όταν τα δεδομένα φορτώνονται στο GraphDB, γίνονται προσβάσιμα σε οποιοδήποτε εργαλείο ή εφαρμογή που μπορεί να αλληλοεπιδράσει με τα τελικά σημεία (endpoints) SPARQL. Αυτό το άνοιγμα βρίσκεται στην καρδιά της έννοιας LOD. Σημαίνει ότι όχι μόνο μπορούμε να συνδέσουμε τα δεδομένα με άλλα σύνολα δεδομένων, αλλά και άλλα άτομα μπορούν επίσης να συνδέσουν τα δεδομένα τους με τα δικά μας, οδηγώντας σε έναν ιστό διασυνδεδεμένων συνόλων δεδομένων. Μια τέτοια ρύθμιση όχι μόνο αυξάνει την

ορατότητα και την εμβέλεια των δεδομένων σας, αλλά επιτρέπει επίσης τον εμπλουτισμό τους με συνδέσεις με άλλα σύνολα δεδομένων.

Η υποστήριξη του GraphDB για πρότυπα RDF και η συμβατότητά του με διάφορες μορφές δεδομένων, διασφαλίζει την ευελιξία και τη δια-λειτουργικότητα που απαιτείται για τις εφαρμογές του Σημασιολογικού Ιστού. Επιπλέον, οι δυνατότητες συμπερασμάτων του GraphDB σημαίνουν ότι μπορεί να αντλήσει αυτόματα νέα δεδομένα από τα υπάρχοντα δεδομένα με βάση προκαθορισμένους κανόνες. Αυτά τα συναγόμενα δεδομένα μπορούν επίσης να εκτεθούν στον Σημασιολογικό Ιστό, ενισχύοντας περαιτέρω τον πλούτο του LOD σας.

Συμπερασματικά, χρησιμοποιώντας το GraphDB στην αρχιτεκτονική μας, δημιουργούμε αποτελεσματικά μια ισχυρή πλατφόρμα για τη διαχείριση, την αναζήτηση και τη δημοσίευση Συνδεδεμένων Ανοικτών Δεδομένων, συμβάλλοντας έτσι στην υλοποίηση του οράματος του Σημασιολογικού Ιστού.

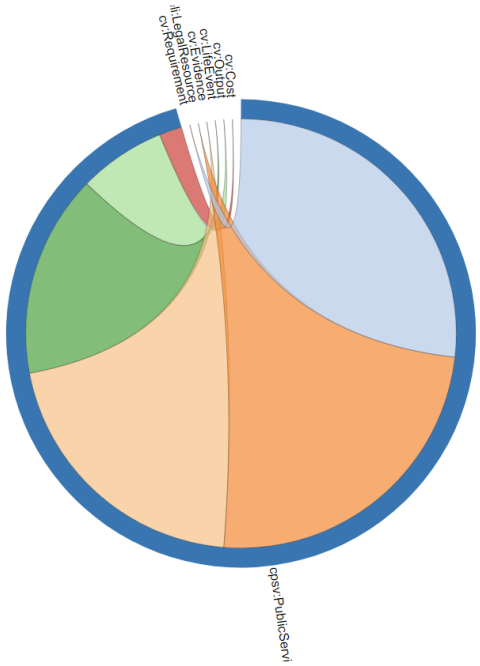
Ακολουθούν οι γραφικές αναπαραστάσεις από τα δημοσιευμένα δεδομένα και ένα κομμάτι αυτών.



Εικόνα 9. Η Ιεραρχία των κλάσεων.

Μας δείχνει την ιεραρχία των RDF κλάσεων με βάση τον αριθμό των εμφανίσεων τους. Οι μεγαλύτεροι κύκλοι είναι και οι πατρικές κλάσεις, οι μικρότεροι κύκλοι είναι οι υπό-κλάσεις

Class	Links
cpsv:PublicService	1K →
cv:Requirement	405 ←
eli:LegalResource	371 ←
cv:Evidence	314 ←
cv:LifeEvent	231 ←
cv:Output	99 ←
cv:Cost	26 ←



Εικόνες 10 & 11. Οι Σχέσεις των κλάσεων.

Στην πρώτη εικόνα απεικονίζεται ο πίνακας που αναφέρει το πόσα Links (Συνδέσμους) περιέχει η κάθε κλάση που την αναφέρει. Ενώ η δεύτερη εικόνα το αναπαριστά σχηματικά.

Προφανώς για λόγους οικονομίας και μεγέθους δεν μπορεί να αποτυπωθεί πλήρως στο παρών έγγραφο αναλυτικά το αρχείο .TTL, επομένως διατίθεται online μέσω Github.

4.8 Οδηγός Εγκατάστασης και Επεξήγησης της Τεχνικής Υποδομής και των Προγραμμάτων

4.8.1 Εισαγωγή στο WSL2 & εγκατάσταση του VirtualEnv

Για να μπορέσουμε να εγκαταστήσουμε τα προγράμματα που θέλουμε να χρησιμοποιήσουμε αλλά και να στήσουμε την υποδομή μας πρέπει να έχουμε στη διάθεση μας έναν web server, ο οποίος θα μπορέσει να φιλοξενήσει όλα τα απαραίτητα για να την υλοποίηση της αρχιτεκτονική μας.

Οι επιλογές που έχουμε στην διάθεση μας είναι 2.

Εξ ορισμού θα πρέπει να χρησιμοποιηθούν τεχνολογίες ως επί το πλείστον OpenSource.

Έτσι λοιπόν η επιλογή μας θα είναι να χρησιμοποιήσουμε έναν Linux webserver.

Στην περίπτωση μας θα δούμε την χρήση ενός WSL2 (Windows Subsystem for Linux 2*.

**σ.σ ο αριθμός 2 αναφέρεται στην έκδοση*

Το υποσύστημα Windows για Linux 2 (WSL2), που εισήχθη από τη Microsoft, αποτελεί έμβλημα της εξελισσόμενης σχέσης της εταιρείας με το λογισμικό ανοιχτού κώδικα και της αυξανόμενης δέσμευσής της στις ανάγκες των προγραμματιστών και της κοινότητας Linux. Αν και η αρχική έκδοση, το WSL1, εφάρμοσε ένα επίπεδο συμβατότητας για την εκτέλεση δυαδικών εκτελέσιμων αρχείων Linux στα Windows 10, το WSL2 κάνει ένα βήμα μπροστά ενσωματώνοντας έναν πλήρη πυρήνα Linux σε ένα ελαφρύ περιβάλλον εικονικής μηχανής.

Το WSL2 είναι ένα υβριδικό υπολογιστικό μοντέλο που χρησιμοποιεί έναν πραγματικό πυρήνα Linux για να διευκολύνει την εκτέλεση δυαδικών αρχείων Linux, να αυξάνει την απόδοση του συστήματος αρχείων και να προσφέρει πλήρη συμβατότητα κλήσεων συστήματος. Αυτό είναι δυνατό λόγω ενός υποσυνόλου χαρακτηριστικών Hyper-V, της τεχνολογίας εικονικοποίησης (virtualization) της Microsoft, η οποία επιτρέπει στο WSL2 να τρέχει την ελαφριά εικονική μηχανή του με ελάχιστη κατανάλωση πόρων. Γεφυρώνει τις εγγενείς διαφορές μεταξύ των περιβαλλόντων Windows και Linux, απλοποιώντας έτσι την εμπειρία του προγραμματιστή.

Η ενσωμάτωση ενός πλήρους πυρήνα Linux επιτρέπει στο WSL2 να εκτελεί εγγενώς container Docker, χωρίς την ανάγκη πρόσθετου λογισμικού εικονικοποίησης. Ως αποτέλεσμα, ενισχύει την ικανότητα των Windows 10 ως πλατφόρμα για την ανάπτυξη, τη δοκιμή και την εκτέλεση εφαρμογών που στοχεύουν σε πολλαπλά περιβάλλοντα. Επιπλέον, μειώνει τα γενικά έξοδα εγκατάστασης και συντήρησης ξεχωριστών συστημάτων ή VM για διαφορετικές πλατφόρμες.

Τα container είναι ελαφριά, αυτόνομα, εκτελέσιμα πακέτα λογισμικού που περιλαμβάνουν όλα όσα χρειάζονται για την εκτέλεση ενός λογισμικού, συμπεριλαμβανομένου του κώδικα, ενός χρόνου εκτέλεσης, βιβλιοθηκών, μεταβλητών περιβάλλοντος και αρχείων διαμόρφωσης. Τα κοντέινερ είναι απομονωμένα μεταξύ τους και δεσμεύουν το δικό τους λογισμικό, βιβλιοθήκες και αρχεία διαμόρφωσης. μπορούν να επικοινωνούν μεταξύ τους μέσω καλά καθορισμένων καναλιών. Όλα αυτά γίνονται για να διασφαλιστεί ότι το λογισμικό λειτουργεί με συνέπεια, ανεξάρτητα από το περιβάλλον στο οποίο εκτελείται, καθιστώντας το ιδανικό για ανάπτυξη, αποστολή και ανάπτυξη εφαρμογών.

Πέρα από την πρακτική χρησιμότητα, η στρατηγική σημασία του WSL2 δεν πρέπει να υποτιμάται. Η υποστήριξη της Microsoft για το Linux στο κυρίαρχο λειτουργικό της σύστημα αντικατοπτρίζει μια στρατηγική στροφή προς την υιοθέτηση οικοσυστημάτων ανοιχτού κώδικα, αναγνωρίζοντας την αξία τους σε σύγχρονα περιβάλλοντα ανάπτυξης και τροφοδοτώντας ένα ευρύτερο φάσμα προγραμματιστών. Σηματοδοτεί ένα σημείο καμπής στον τρόπο λειτουργίας της Microsoft, επιδεικνύοντας την προθυμία να παρέχει στους προγραμματιστές τα εργαλεία που χρειάζονται, ακόμα κι αν αυτό σημαίνει συνδυασμό προηγούμενων ανταγωνιστικών πλατφορμών.

Ο αντίκτυπος του WSL2 εκτείνεται πέρα από μεμονωμένους προγραμματιστές και επηρεάζει και τις επιχειρήσεις. Μειώνει τα εμπόδια εισόδου για λειτουργίες μικτού περιβάλλοντος, ενθαρρύνει ένα πιο ποικιλόμορφο αναπτυξιακό οικοσύστημα και υποστηρίζει πιο ευέλικτες, platform-agnostic* λειτουργίες. Ως εκ τούτου, ανοίγει έναν δρόμο για τα Windows να είναι μια πιο ελκυστική και ευέλικτη πλατφόρμα για προγραμματιστές, οι οποίοι πρέπει να εργαστούν με τεχνολογίες ανοιχτού κώδικα και εφαρμογές πολλαπλών πλατφορμών.

*Επεξήγηση Όρου: Ο όρος platform-agnostic, αναφέρεται σε λογισμικό ή εφαρμογές οι οποίες τρέχουν σε οποιοδήποτε λειτουργικό σύστημα ή πλατφόρμα χωρίς να χρειάζονται ιδιαίτερη μεταχείριση προκειμένου να λειτουργήσουν. Έχουν σχεδιαστεί ουσιαστικά με τρόπο τέτοιο που να μην περιορίζονται από το να χρειάζονται να εκτελεστούν σε κάποιο συγκεκριμένο περιβάλλον.

Συνοπτικά, το WSL2 δεν είναι απλώς ένα τεχνικό κατόρθωμα, αλλά και μια στρατηγική κίνηση που ευθυγραμμίζεται με τις τρέχουσες τάσεις στη βιομηχανία λογισμικού. Με την ενσωμάτωση του Linux με τα Windows, η Microsoft προσφέρει μια ολοκληρωμένη πλατφόρμα που εξυπηρετεί τις ανάγκες των προγραμματιστών στον κόσμο ανάπτυξης πολλαπλών περιβαλλοντικών επιπτώσεων. Δείχνει πώς ένας γίγαντας λογισμικού μπορεί να εξελιχθεί, υιοθετώντας μια πιο ανοιχτή προσέγγιση για να διατηρήσει και να επεκτείνει τη βάση χρηστών του στο δυναμικό τοπίο της ανάπτυξης λογισμικού.

(Θεωρούμε ως μη-σημαντικό να αναφερθούμε στην διαδικασία για το πως στήνεται καθώς υπάρχει πληθώρα πληροφοριών ελεύθερη και δωρεάν στο διαδίκτυο για το συγκεκριμένο θέμα)

Το πρώτο βήμα που έχουμε να υλοποιήσουμε αφού ετοιμάσουμε λοιπόν το WSL2 είναι να αρχικοποιήσουμε το Εικονικό μας περιβάλλον (Virtual Environment) που θα δουλέψουμε.

Σαν root χρήστης επιτελούμε τα ακόλουθα και με την σειρά εκτέλεσης αυτό που κάνουμε είναι, [1] να μεταφερθούμε στο home directory του server μας,

[2] να εγκαταστήσουμε το πακέτο virtualenv, το οποίο είναι υπεύθυνο για το εικονικό περιβάλλον.

[3] να το δημιουργήσουμε το περιβάλλον αυτό

[4] να το ενεργοποιήσουμε

```
[1] cd ~
[2] pip install virtualenv
[3] virtualenv airflow_env
[4] source airflow_env/bin/activate
```

Στη συνέχεια δημιουργούμε έναν φάκελο που τον ονομάζουμε airflow. Αφού δημιουργήσουμε τον φάκελο τότε τον αποθηκεύουμε σε μία global μεταβλητή προκειμένου να μπορούμε να επαναχρησιμοποιήσουμε την διαδρομή (folder path) εύκολα.

```
[1] nano ~/.bashrc
[2] AIRFLOW_HOME=/c/Users/[YourUsername]/airflow
```

4.8.2 Εγκατάσταση & Αρχικοποίηση του Apache Airflow

Με την εγκατάσταση του εικονικού μας περιβάλλοντος και αφού είναι ενεργοποιημένο, μπορούμε να προχωρήσουμε στην εγκατάσταση του Apache Airflow.

```
[1] pip install apache-airflow
```

Μετά την επιτυχή εγκατάσταση του Apache Airflow μπορούμε να αρχικοποιήσουμε (initialize) την βάση δεδομένων του.

```
[2] airflow db init
```

Δημιουργούμε έναν φάκελο, ο οποίος θα είναι αυτός που θα χρησιμοποιηθεί για να αποθηκεύσουμε όλα τα script μας και έναν χρήστη ο οποίος θα είναι το account που θα χρησιμοποιήσουμε για να μπορούμε να κάνουμε login στο UI του Airflow.

```
[3] airflow users create --username admin –password admin –firstname admin
–lastname admin –role Admin –email youremail@email.com
```

```
[4] airflow users list – Μας επιτρέπει να δούμε τους δημιουργημένους
χρήστες και να επιβεβαιώσουμε την ύπαρξη τους.
```

Εκτελούμε τον scheduler και ξεκινάμε τον webserver εκτελώντας τις εντολές 5 & 6.

```
[5] airflow scheduler
// Εκκίνηση scheduler
```


```
[6] airflow webserver
// Εκκίνηση webserver
```

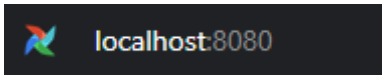
Σε περίπτωση που η default πόρτα που τρέχει ο Apache Airflow (8080) είναι ήδη σε χρήση, μπορούμε να την αλλάξουμε με την εντολή 7.

```
[7] airflow webserver –port <port number>
```

Τέλος, μπορούμε να πραγματοποιήσουμε σύνδεση μέσω του UI στο admin panel με τα στοιχεία χρήστη που μόλις δημιουργήσαμε.

```
[2023-06-26 12:17:46 +0300] [1935] [INFO] Listening at: http://0.0.0.0:8080 (1935)
```

σ.σ . Η διεύθυνση σύνδεσης στο διαχειριστικό θα είναι localhost:<port>



5. Συμπεράσματα – Προτάσεις για μελλοντική έρευνα

Συμπερασματικά, η έρευνά μας επιβεβαιώνει τη σημασία του μετασχηματισμού των δημόσιων υπηρεσιών σε Συνδεδεμένα Ανοικτά Δεδομένα (LOD). Η μετάβαση επιτρέπει βελτιωμένη δια-λειτουργικότητα και προσβασιμότητα δεδομένων, ενισχύοντας ένα περιβάλλον διαφάνειας, συνεργασίας και συμμετοχής των πολιτών. Επιπλέον, ενδυναμώνει τους πολίτες, τις επιχειρήσεις και άλλες κυβερνητικές οντότητες παρέχοντάς τους κρίσιμες πληροφορίες σε δομημένη, μηχανικά αναγνώσιμη μορφή, ανοίγοντας το δρόμο για πιο ενημερωμένη λήψη αποφάσεων και καινοτόμες λύσεις.

Επιπλέον, διαπιστώσαμε ότι η χρήση των LOD σε δημόσιες υπηρεσίες υποστηρίζει την κυβερνητική αποτελεσματικότητα και προωθεί την εξοικονόμηση κόστους, καθώς διευκολύνει την επαναχρησιμοποίηση και την κοινή χρήση δεδομένων σε ένα ευρύ φάσμα τμημάτων και φορέων. Τα ευρήματά μας υπογραμμίζουν το γεγονός ότι τα LOD δεν είναι απλώς ένα προαιρετικό πρόσθετο, αλλά μια θεμελιώδης αλλαγή στη διαχείριση των δημόσιων δεδομένων, η οποία μπορεί να απελευθερώσει ουσιαστική και αναντικατάστατη κοινωνική, οικονομική και πολιτική αξία. Ως εκ τούτου, η υιοθέτηση των LOD θα πρέπει να θεωρείται προτεραιότητα στις στρατηγικές εκσυγχρονισμού των δημόσιων υπηρεσιών στο πλαίσιο της ψηφιακής εποχής και της κοινωνίας της πληροφορίας.

Στην τεχνική υλοποίηση ενός τέτοιου έργου υπάρχουν διάφορα ευρήματα που χρήζουν αξιολόγησης. Αναλύοντας τα ένα-προς-ένα καταλήγουμε στα εξής:

Α) Η δομή της πληροφορίας στο Ηλεκτρονικό Αποθετήριο “ΜΙΤΟΣ” αν και είναι έτοιμη προς αξιοποίηση, συχνά παρατηρείται ότι η δομή της δεν είναι σε ώριμο στάδιο. Η σύνθεση των δεδομένων όπως βρίσκονται στην παρούσα κατάσταση, καθιστά δύσκολη και δυσνόητη την διαδικασία αντιστοίχισης των πεδίων για να περιγράφουνε με το λεξιλόγιο CPSV-AP. Αιτία αυτού του γεγονότος, αποτελεί η αρχική υποδομή και αρχιτεκτονική στο εν λόγω έργο, η οποία δεν είναι πλήρως εναρμονισμένη με τα πρότυπα.

Αποτέλεσμα αυτής της έλλειψης είναι η ανάκυψη νέων ζητημάτων προς συζήτηση σε οποιαδήποτε προσπάθεια γίνει η οποία αφορά την αντιστοίχιση των τύπων αυτών των

πεδίων λόγω της δυσνόητης περιγραφής τους και χρήστης τους από την Ελληνική Κυβέρνηση, τους πολίτες, τις επιχειρήσεις και τους φορείς.

Β) Το API που προσφέρεται αυτή την στιγμή είναι και αυτό σε πρώιμο στάδιο και αυτό προκύπτει από τα εξής:

1. Η επιστροφή όλων των δεδομένων στις κλήσεις των ερωτημάτων γίνεται σε μορφή κειμένου (τύπου String). Ενδεχομένως σε συγκεκριμένες υλοποιήσεις η εν λόγω τεχνική να μην καθιστά πρόβλημα, ωστόσο παραβιάζει τις αρχές της ακεραιότητας των δεδομένων και των διεθνώς αναγνωρισμένων βέλτιστων πρακτικών καθώς ο τύπος επιστροφής των δεδομένων πρέπει να είναι ο φυσικός τύπος τους.

2. Το infrastructure κομμάτι της αρχιτεκτονικής που είναι υπεύθυνο για το Web Service αν και σε ‘πιλοτικό’ επίπεδο δεν μπορεί να ανταποκριθεί σε ρεαλιστικές συνθήκες και ανάγκες. Παράδειγμα αυτού αποτελεί το γεγονός το ότι το Service ενδέχεται να ‘πέσει’, τεχνικά δηλαδή να προκληθεί outage και να κοπεί η πρόσβαση σε όλους, αν κάποιος

ενδιαφερόμενος επιτελέσει κλήσεις χωρίς φίλτρα για περιορισμό αποτελεσμάτων, γεγονός που δημιουργεί μεγάλο προβληματισμό.

3. Η πληροφορία δεν καθαρίζεται σωστά όταν έρχεται από την βάση και προσφέρεται μέσω API. Αυτό έχει ως αποτέλεσμα πολλά responses να έχουν ‘σκουπίδια’ στις απαντήσεις όπως (ειδικές χαρακτήρες *, ; , \n) οι οποίοι μπορούν να δημιουργήσουν θέμα. Ο καθαρισμός θα έπρεπε να μπορεί να επιτελείτε σε επίπεδο service και όχι να μετατεθείτε η ευθύνη στον αποδέκτη της πληροφορίας.

4. Οι συναρτήσεις και οι μέθοδοι είναι ποσοτικά λίγες. Μπορεί κάτι τέτοιο να είναι λογικό, καθώς το API βρίσκεται σε πρώιμο στάδιο όντας μια νέα υπηρεσία, ωστόσο στο άμεσο μέλλον θα πρέπει να εξερευνηθούν και να γίνουν δημόσιες και άλλες λειτουργίες.

Γ) Αν και υπάρχουν ισχυρά εργαλεία διαθέσιμα για συγκεκριμένες εργασίες, όπως το Apache Jena για το χειρισμό RDF, το SPARQL για τα ερώτημα, το Virtuoso και το GraphDB για την αποθήκευση δεδομένων RDF, το συνολικό οικοσύστημα εξακολουθεί να έχει περιθώρια για βελτίωση. Πολλά εργαλεία δεν διαθέτουν ολοκληρωμένη τεκμηρίωση, δεν είναι φιλικά προς τους νέους χρήστες ή δεν συντηρούνται επαρκώς.

Είναι σημαντικό για την κοινότητα ανοιχτού κώδικα να συνεχίσει την ανάπτυξη σε αυτούς τους τομείς για να διασφαλίσει ότι το μετασχηματιστικό δυναμικό των RDF και των LOD μπορεί να υλοποιηθεί πλήρως, χωρίς να αποτρέπουν τη χρήση τους λόγω τέτοιων δυσκολιών.

Ένα από τα θέματα που θα μπορούσαν μελλοντικά να ερευνηθούν είναι το εξής:

Η δημιουργία visualization ώστε να μπορούν οι χρήστες να 'ταξιδέψουν' στα ανοικτά δημόσια δεδομένα και να βλέπουν τις συσχετίσεις των οντοτήτων αυτών, προσφέροντας έτσι μια μοντέρνα δια δραστική εμπειρία.

Το API του ΜΙΤΟΣ δεν είναι το μοναδικό που απευθύνεται σε συγκεκριμένα πεδία. Πεδία όπως οι φορείς, που χρησιμοποιούνται από το ίδιο είναι διαθέσιμα και με πολλές περισσότερες πληροφορίες μέσω Τρίτων APIs του δημοσίου. https://api.digigov.grnet.gr/v1/organizations

Αυτό πρακτικά σημαίνει ότι όλη η πληροφορία που υπάρχει διαθέσιμη μέσων των APIs του δημοσίου, μπορεί να οργανωθεί δημιουργώντας ένα knowledge Graph που ο χρήστης θα

μπορεί να κάνει traverse και search εικονικά στον browser του, είτε μέσω κειμένου είτε μέσω διαδράσεων (interactions).

Μερικά οφέλη που θα προκύψουν από την ολοκλήρωση μίας τέτοιας μελέτης είναι η εξοικείωση του ατόμου με χώρους, εργαλεία και τεχνολογίες όπως:

- Η μελέτη της διαδικασίας εξαγωγής και μετασχηματισμού των Δεδομένων με Συνδέσεις για οπτικοποίηση τους.
- Η εξερεύνηση της γκάμας των διαθέσιμων τεχνικών οπτικοποίησης για διάφορους τύπους δεδομένων.
- Οι τύποι εργαλείων οπτικοποίησης δεδομένων με συνδέσεις που είναι διαθέσιμες επί του παρόντος.
- Προσεγγίσεις στην οπτικοποίηση του σύννεφου των Linked Open Data (LOD Public Cloud).
- Η χρήση των πινάκων ελέγχου για να παρέχουν περιληπτικές πληροφορίες για ένα σύνολο δεδομένων.
- Πώς η σημασιολογία μπορεί να χρησιμοποιηθεί για την οδήγηση της αναζήτησης και την εμφάνιση των αποτελεσμάτων της αναζήτησης.
- Η επαφή με εργαλεία που μπορούν να χρησιμοποιηθούν για την αναζήτηση σημασιολογικών δεδομένων.
- Πώς τα δεδομένα μπορούν να συγκεντρωθούν και να αναλυθούν στατιστικά.
- Πώς η μηχανική μάθηση μπορεί να χρησιμοποιηθεί για τον εντοπισμό των προτύπων σε ένα σύνολο δεδομένων και να εξαχθούν συμπεράσματα για αυτά.

Καταλήγοντας, η παρούσα διπλωματική εργασία επιβεβαιώνει έντονα τη χρήση αυτοματοποιημένων ροών εργασίας για την εξαγωγή, τον μετασχηματισμό, την επικύρωση και τη δημοσίευση των Δημόσιων Υπηρεσιών ως Συνδεδεμένα Ανοικτά Δεδομένα.

Τα ευρήματά μας καταδεικνύουν ότι η αυτοματοποιημένη προσέγγιση όχι μόνο ενισχύει την αποτελεσματικότητα της διαδικασίας, αλλά εξασφαλίζει επίσης υψηλότερο επίπεδο ακρίβειας και συνέπειας δεδομένων σε σύγκριση με τις μη αυτόματες διαδικασίες.

Επιπλέον, οι αυτοματοποιημένες ροές εργασίας αποτελούν μια επεκτάσιμη λύση, η οποία είναι απαραίτητη για τη διαχείριση μεγάλου όγκου δεδομένων δημόσιας υπηρεσίας.

Η αυτοματοποίηση αυτών των διαδικασιών μειώνει τη δυνατότητα ανθρώπινου λάθους και ταυτόχρονα επιτρέπει συνεχείς ενημερώσεις σε πραγματικό χρόνο, αυξάνοντας έτσι τη συνάφεια και την επικαιρότητα των δεδομένων.

Δεδομένων αυτών των πολυάριθμων πλεονεκτημάτων, η υλοποίηση αυτοματοποιημένων ροών εργασίας θα πρέπει να αποτελεί πρωταρχικό κριτήριο στις προσπάθειες μετατροπής δεδομένων δημόσιων υπηρεσιών σε Συνδεδεμένα Ανοικτά Δεδομένα.

Αυτός ο μετασχηματισμός θα μπορούσε να λειτουργήσει ως καταλύτης για τη λήψη αποφάσεων βάσει δεδομένων, την καινοτομία και την αυξημένη διαφάνεια στις δημόσιες υπηρεσίες.

Βιβλιογραφία

Ακολουθούν οι βιβλιογραφικές αναφορές (πηγές) της Εργασίας.

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.

Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., Auer, S., & Sack, H. (2016). Quality assessment for Linked Data: A survey. Semantic Web, 7(1), 63-93.

Allemang, D., & Hendler, J. (2011). Semantic web for the working ontologist: effective modeling for advanced semantic web applications. Morgan Kaufmann.

Parycek, P., & Sachs, M. (2011). Linked open government data for improving public services. Government Information Quarterly, 28(2), 166-175.

Hogan, A., Harth, A., & Passant, A. (2011). Weaving the pedantic web. Linked data for the working ontologist, 233-255.

Tassoni, F., Domingue, J., & Salvadores, M. (2014). Linked data for open and distance learning. IEEE Transactions on Learning Technologies, 7(3), 246-259.

Lopes, N., Gonçalves, R., & de Melo, G. (2018). Linked data and public services in smart cities. In Smart Cities and Smart Spaces (pp. 55-78). Springer.

McNaughton, J. (2014). Linked data and government documents. Government Information Quarterly, 31(1), 94-101.

Davies, J., & Fensel, D. (2010). Linked data-a geographical perspective. Semantic Web, 1(2), 129-134.

van Harmelen, F., Aroyo, L., & Stash, N. (2014). Open data: powering the future of the knowledge society. Amsterdam University Press.

Σπανός (2015). Συνδεδεμένα Δεδομένα: Μια ευκαιρία για τις ελληνικές Βιβλιοθήκες

Γέροντας, Ταμπούρης, Λαζοπούλου, Ταραμπάνης (2018). CPSV-AP to publish public service descriptions as linked open data

Gerontas, Zeginis, Promikyridis, Andros, Tamboyris, Vibor Cipan, Tarabanis (2022). Enhancing Core Public Service Vocabulary to Enable Public Service Personalization

OpenRefine.org (https://openrefine.org/). Ανακτήθηκε 3/ 2023

Apache Airflow (https://airflow.apache.org/). Ανακτήθηκε 4/2023

Υπεύθυνη Δήλωση Συγγραφέα:

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν.1599/1986, η παρούσα εργασία αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης.