|  |  |
| --- | --- |
|  | Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης  Σχολή Θετικών Επιστημών  **Τμήμα Πληροφορικής** |

Πτυχιακή/Διπλωματική Εργασία

«Διαλειτουργική αναπαράσταση και αναζήτηση δεδομένων από συσκευές παρακολούθησης δραστηριότητας»

(Cross-functional representation and retrieval of data from activity trackers)

«ΑΝΔΡΕΑΣ ΝΑΛΜΠΑΝΤΗΣ»

**ΑΕΜ**: 3699 **Ροή**: Πληροφοριακά Συστήματα

**Επιβλέπων Καθηγητής:**

**Γεώργιος Μεδίτσκος**

**Επίκουρος Καθηγητής Τμήματος Πληροφορικής ΑΠΘ**

**Θεσσαλονίκη 2023**

Περίληψη

Η παρούσα πτυχιακή εργασία εστιάζει στην έννοια του Σημασιολογικού Ιστού και αποσκοπεί στη συλλογή δεδομένων από συσκευές παρακολούθησης δραστηριότητας και τη χρήση διαφόρων τεχνολογιών του Σημασιολογικού Ιστού για την εξαγωγή συμπερασμάτων από αυτά. Αρχικά, πραγματοποιήθηκε η συλλογή δεδομένων από ένα ανοιχτό dataset που περιλάμβανε δεδομένα από συσκευές δραστηριότητας Fitbit, τα οποία παρουσιάζονταν σε μορφή CSV αρχείων. Αφού συλλέχθηκαν τα δεδομένα, έπρεπε να κατασκευαστεί μία OWL οντολογία με τις ανάλογες κλάσεις και ιδιότητες για την αναπαράσταση τους.

Για την ενσωμάτωση αυτών των δεδομένων στην οντολογία, αναπτύχθηκε ένα πρόγραμμα σε Python, με την χρήση της βιβλιοθήκης owlready2, το οποίο παρείχε αυτοματοποιημένη τη διαδικασία δημιουργίας της οντολογίας, συμπεριλαμβάνοντας τις απαραίτητες κλάσεις και ιδιότητες. Η οντολογία βασίστηκε και χρησιμοποίησε κλάσεις και ιδιότητες από τις οντολογίες SSN και SOSA.

Έπειτα από την κατασκευή της οντολογίας, αναπτύχθηκε ένα πρόγραμμα που διάβαζε τα CSV αρχεία και αποθήκευε τα δεδομένα στην οντολογία σε μορφή RDF τριπλετών, αντιστοιχίζοντάς τα με τις κατάλληλες ιδιότητες και κλάσεις. Αυτό το πρόγραμμα υλοποιήθηκε επίσης σε Python.

Στη συνέχεια, πραγματοποιήθηκε η επικύρωση των δεδομένων χρησιμοποιώντας την γλώσσα επικύρωσης RDF γράφων, SHACL. Δημιουργήθηκαν SHACL κανόνες για τον έλεγχο όλων των στιγμιότυπων (instances) των κλάσεων της οντολογίας, με σκοπό να διασφαλιστεί η ορθότητα τους.

Τέλος, τα δεδομένα αποθηκεύτηκαν σε μια βάση GraphDB. Προκειμένου να επιτευχθεί η οπτικοποίηση των δεδομένων και η εξαγωγή συμπερασμάτων, αναπτύχθηκε μια web εφαρμογή που συνδέεται με τη βάση δεδομένων. Η εφαρμογή επιτρέπει στους χρήστες να προβάλουν τα δεδομένα, να παρακολουθούν τις διακυμάνσεις τους κατά τη διάρκεια ορισμένων χρονικών περιόδων και να παρατηρούν τις αλλαγές που συμβαίνουν, προκειμένου να εξάγουν τα δικά τους συμπεράσματα.

Με την εφαρμογή αυτή, επιτυγχάνεται η ολοκλήρωση της εργασίας και η ανάπτυξη μιας συστηματικής προσέγγισης για τη συλλογή, επεξεργασία και ανάλυση δεδομένων από fitness trackers με τη χρήση τεχνολογιών του Σημασιολογικού Ιστού. Τα αποτελέσματα και τα συμπεράσματα που προκύπτουν από αυτήν την προσέγγιση μπορούν να έχουν εφαρμογές στον τομέα της υγείας και της ευεξίας.

Abstract

The present thesis focuses on the concept of the Semantic Web and aims to collect data from activity monitoring devices and utilize various Semantic Web technologies to extract insights from them. Initially, data was collected from an open dataset containing Fitbit activity device data presented in CSV file format. Once the data was collected, an OWL ontology had to be constructed with the corresponding classes and properties to represent it.

To integrate this data into the ontology, a Python program was developed using the library owlready2, which automated the ontology creation process, including the necessary classes and properties in it. The ontology was based on and utilized classes and properties from the SSN and SOSA ontologies.

Following the ontology construction, a program was developed to read the CSV files and store the data in the ontology in the form of RDF triples, mapping them to the appropriate properties and classes. This program was also implemented in Python.

Subsequently, the data was validated using the RDF graph validation language, SHACL. SHACL rules were created to check all the instances of the ontology classes, ensuring their correctness.

Finally, the data was stored in a GraphDB database. To visualize the data and extract insights, a web application was developed that connects to the database. The application allows users to view the data, track fluctuations during specific time periods, and observe changes that occur, enabling them to draw their own conclusions.

With this application, the completion of the work and the development of a systematic approach to data collection, processing, and analysis from fitness trackers using Semantic Web technologies are achieved. The results and insights obtained from this approach can have applications in the fields of health and well-being.

Ευχαριστίες

Πριν την παρουσίαση των αποτελεσμάτων της παρούσας εργασίας, αισθάνομαι την υποχρέωση να ευχαριστήσω ορισμένους από τους ανθρώπους με τους οποίους συνεργάστηκα και έπαιξαν πολύ σημαντικό ρόλο στην πραγματοποίησή της.

Θα ήθελα να ευχαριστήσω τον Επίκουρο Καθηγητή του Τμήματος Πληροφορικής του ΑΠΘ, κύριο Γεώργιο Μεδίτσκο για την βοήθεια και την καθοδήγηση που μου προσέφερε καθ’ όλη την περίοδο εκπόνησης της εργασίας. Επίσης, θα ήθελα να ευχαριστήσω τους γονείς μου Νίκο και Κορίνα για την σημαντική υποστήριξη που μου παρείχαν σε όλη την διάρκεια των προπτυχιακών σπουδών μου.

<7/1/2023>

Ανδρέας Ναλμπάντης

Περιεχόμενα

[ΚΕΦΑΛΑΙΟ 1: Εισαγωγή 17](#_Toc139115040)

[1.1 Γενική Ιδέα 19](#_Toc139115041)

[1.2 Ενότητες Εργασίας 20](#_Toc139115042)

[ΚΕΦΑΛΑΙΟ 2: Σημασιολογικοσ Ιστοσ 21](#_Toc139115043)

[2.1 Εισαγωγή 23](#_Toc139115044)

[2.1.1 Ιστορική Αναδρομή 24](#_Toc139115045)

[2.1.2 Βασικές Έννοιες του Σημασιολογικού Ιστού 25](#_Toc139115046)

[2.1.3 Διασυνδεδεμένα Δεδομένα (Linked Data) 26](#_Toc139115047)

[2.1.3.1 Ανοιχτά Διασυνδεδεμένα Δεδομένα 26](#_Toc139115048)

[2.2 Resource Description Framework (RDF) 27](#_Toc139115049)

[2.2.1 Σύνταξη τους RDF 29](#_Toc139115050)

[2.3 RDF Schema (RDFS) 30](#_Toc139115051)

[2.3.1 Λεξιλόγιο της RDFS 31](#_Toc139115052)

[2.3.1.1 Κλάσεις 31](#_Toc139115053)

[2.3.1.2 Ιδιότητες 31](#_Toc139115054)

[2.4 Web Ontology Language (OWL) 34](#_Toc139115055)

[2.4.1 Διαφορές με RDFS 34](#_Toc139115056)

[2.4.2 Τύποι Ιδιοτήτων 35](#_Toc139115057)

[Αντίστροφες Ιδιότητες 35](#_Toc139115058)

[Συμμετρικές Ιδιότητες 35](#_Toc139115059)

[Μεταβατικές Ιδιότητες 35](#_Toc139115060)

[Συναρτησιακές Ιδιότητες 36](#_Toc139115061)

[Αντίστροφες Συναρτησιακές Ιδιότητες 36](#_Toc139115062)

[2.4.3 Ισοδυναμία 37](#_Toc139115063)

[Ισοδυναμία Κλάσεων 37](#_Toc139115064)

[Ισοδυναμία Ιδιοτήτων 37](#_Toc139115065)

[Ομοιότητα Στιγμιότυπων 38](#_Toc139115066)

[2.4.4 Περιορισμοί 38](#_Toc139115067)

[owl:someValuesFrom 38](#_Toc139115068)

[owl:allValuesFrom 39](#_Toc139115069)

[owl:hasValue 39](#_Toc139115070)

[2.5 SPARQL 39](#_Toc139115071)

[2.5.1 Δομή ενός SPARQL ερωτήματος 40](#_Toc139115072)

[2.5.2 SELECT και WHERE 41](#_Toc139115073)

[2.5.3 CONSTRUCT 42](#_Toc139115074)

[2.5.4 ASK / DESCRIBE 43](#_Toc139115075)

[2.5.5 Εντολή FILTER 44](#_Toc139115076)

[2.5.6 Εντολή LIMIT 44](#_Toc139115077)

[2.5.7 Εντολή ORDER BY 45](#_Toc139115078)

[2.5.8 Μερικά παραδείγματα 45](#_Toc139115079)

[2.6 Σημασιολογικές Βάσεις Δεδομένων 47](#_Toc139115080)

[2.6.1 Graph Databases 47](#_Toc139115081)

[2.6.1.1 Labeled Property Graph (LPG) 48](#_Toc139115082)

[2.6.1.2 Περιπτώσεις Χρήσης 48](#_Toc139115083)

[2.6.2 RDF Triplestore 49](#_Toc139115084)

[2.6.2.1 Περιπτώσεις Χρήσης 49](#_Toc139115085)

[ΚΕΦΑΛΑΙΟ 3: ΣΥΣΚΕΥΕΣ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 51](#_Toc139115086)

[3.1 Εισαγωγή 53](#_Toc139115087)

[3.1.1.1 Βασικές κατηγορίες fitness trackers 53](#_Toc139115088)

[3.1.2 Smartwatches 54](#_Toc139115089)

[3.2 SSN – SOSA: Βασικές Δυνατότητες και Δομές 56](#_Toc139115090)

[3.2.1 Τμηματοποίηση 56](#_Toc139115091)

[3.2.1.1 Vertical Segmentation 57](#_Toc139115092)

[3.2.1.2 Horizontal Segmentation 57](#_Toc139115093)

[3.2.2 Συντακτικό των SSN/SOSA 58](#_Toc139115094)

[3.2.2.1 Παρατηρήσεις (Observations) 58](#_Toc139115095)

[3.2.2.2 Ενεργοποιήσεις (Actuations) 60](#_Toc139115096)

[3.2.2.3 Δειγματοληψία (Sampling) 61](#_Toc139115097)

[3.2.2.4 Χαρακτηριστικά Ενδιαφέροντος και Ιδιότητες 62](#_Toc139115098)

[3.2.2.5 Διαδικασίες και Αποτελέσματα 63](#_Toc139115099)

[3.2.2.6 Συστήματα και Ανάπτυξη τους 64](#_Toc139115100)

[ΚΕΦΑΛΑΙΟ 4: Υλοποίηση 66](#_Toc139115101)

[4.1 Περιγραφή Του Προβλήματος 67](#_Toc139115102)

[4.1.1 Επιλογή Dataset 67](#_Toc139115103)

[4.2 Εργαλεία – Frameworks που Χρησιμοποιήθηκαν 68](#_Toc139115104)

[4.3 Αρχιτεκτονική 69](#_Toc139115105)

[4.4 Οντολογία 69](#_Toc139115106)

[4.4.1 Περιγραφή Οντολογίας 69](#_Toc139115107)

[4.4.2 Δημιουργία οντολογίας με owlready2 71](#_Toc139115108)

[4.5 Μετασχηματισμός Δεδομένων 72](#_Toc139115109)

[4.6 Επικύρωση Αποτελεσμάτων 77](#_Toc139115110)

[4.7 Αποθήκευση Δεδομένων 81](#_Toc139115111)

[4.8 Dashboards και παραδείγματα ερωτημάτων 84](#_Toc139115112)

[ΚΕΦΑΛΑΙΟ 5: Συμπεράσματα Και Μελλοντικές Εργασίες 90](#_Toc139115113)

[5.1 Συμπεράσματα και Προβληματισμοί 91](#_Toc139115114)

[5.2 Προτασεις για Μελλοντικη Εργασια 92](#_Toc139115115)

[ΠΑΡΑΡΤΗΜΑ I: Αναφορές 93](#_Toc139115116)

[ΠΑΡΑΡΤΗΜΑ II: Ακρώνυμα 96](#_Toc139115117)

[ΠΑΡΑΡΤΗΜΑ III: Οδηγίες Εκτέλεσησ 98](#_Toc139115118)

Λίστα Σχημάτων

[Εικόνα 1 - Διαδίκτυο: Δίκτυο Διασυνδεδεμένων Υπολογιστών 23](#_Toc139114941)

[Εικόνα 2 – The Semantic Web Stack 25](#_Toc139114942)

[Εικόνα 3 – 5 star deployment scheme for Open Data 27](#_Toc139114943)

[Εικόνα 4 – Δομή τους RDF triple 28](#_Toc139114944)

[Εικόνα 5 – Παράδειγμα τους RDF graph 29](#_Toc139114945)

[Εικόνα 6 – RDF/XML syntax 29](#_Toc139114946)

[Εικόνα 7 – Turtle syntax 30](#_Toc139114947)

[Εικόνα 8 – Turtle syntax με την εφαρμογή των κανόνων 30](#_Toc139114948)

[Εικόνα 9 – SPARQL Endpoint της DBpedia 40](#_Toc139114949)

[Εικόνα 10 – Παράδειγμα SPARQL ερωτήματος 41](#_Toc139114950)

[Εικόνα 11 – Παράδειγμα ερωτήματος SELECT – WHERE 41](#_Toc139114951)

[Εικόνα 12 – Αποτελέσματα ερωτήματος 42](#_Toc139114952)

[Εικόνα 13 – Παράδειγμα CONSTRUCT ερωτήματος 42](#_Toc139114953)

[Εικόνα 14 – Παράδειγμα CONSTRUCT ερωτήματος 43](#_Toc139114954)

[Εικόνα 15 – Αποτελέσματα ερωτήματος 43](#_Toc139114955)

[Εικόνα 16 – Παράδειγμα ASK ερωτήματος 43](#_Toc139114956)

[Εικόνα 17 – Παράδειγμα DESCRIBE ερωτήματος 44](#_Toc139114957)

[Εικόνα 18 – Παράδειγμα χρήσης FILTER 44](#_Toc139114958)

[Εικόνα 19 – Παράδειγμα χρήσης LIMIT 44](#_Toc139114959)

[Εικόνα 20 – Παράδειγμα χρήσης ORDER BY 45](#_Toc139114960)

[Εικόνα 21 – Παράδειγμα LPG 48](#_Toc139114961)

[Εικόνα 22 – Apple Watch Series 8 55](#_Toc139114962)

[Εικόνα 23 – Οι οντολογίες SSN και SOSA με τους τα κάθετα και οριζόντια modules τους 58](#_Toc139114963)

[Εικόνα 24 – Κλάσεις και σχέσεις των SSN/SOSA σχετικά με τα Observations 58](#_Toc139114964)

[Εικόνα 25 – Κλάσεις και ιδιότητες των SSN/SOSA σχετικά με Actuations 60](#_Toc139114965)

[Εικόνα 26 – Κλάσεις και σχέσεις των SSN/SOSA σχετικά με Sampling 61](#_Toc139114966)

[Εικόνα 27 – Κλάσεις και ιδιότητες των SSN/SOSA σχετικά με FoT και Properties 62](#_Toc139114967)

[Εικόνα 28 – Κλάσεις και ιδιότητες των SSN/SOSA σχετικά με Procedures 63](#_Toc139114968)

[Εικόνα 29 – Κλάσεις και ιδιότητες των SSN/SOSA σχετικά με Results 63](#_Toc139114969)

[Εικόνα 30 – Κλάσεις και ιδιότητες των SSN/SOSA σχετικά με τα Systems και το Deployment 64](#_Toc139114970)

[Εικόνα 31 – Δείγμα από το επιλεγμένο dataset 68](#_Toc139114971)

[Εικόνα 32 – Δείγμα από τα δεδομένα ύπνου των χρηστών 70](#_Toc139114972)

[Εικόνα 33 – Δημιουργία τους οντολογίας 71](#_Toc139114973)

[Εικόνα 34 – Προσθήκη κλάσεων και ιδιοτήτων 72](#_Toc139114974)

[Εικόνα 35 – Αρχείο main.py 72](#_Toc139114975)

[Εικόνα 36 – Συνάρτηση loadData 73](#_Toc139114976)

[Εικόνα 37 – Χρήση pandas για το διάβασμα των αρχείων 73](#_Toc139114977)

[Εικόνα 38 – Διάσχιση του αρχείου ανά γραμμή 74](#_Toc139114978)

[Εικόνα 39 – Δημιουργία στιγμιότυπων τύπου Person 74](#_Toc139114979)

[Εικόνα 40 – Κλήση τους κατάλληλης συνάρτησης για το κάθε αρχείο 75](#_Toc139114980)

[Εικόνα 41 – Συνάρτηση loadWeightLogInfo() 76](#_Toc139114981)

[Εικόνα 42 – Συνάρτηση setDateTime() 77](#_Toc139114982)

[Εικόνα 43 – Shape για την κλάση Person 78](#_Toc139114983)

[Εικόνα 44 – Shape για την κλάση Observation 79](#_Toc139114984)

[Εικόνα 45 – Shape για την κλάση Result 79](#_Toc139114985)

[Εικόνα 46 – Validation τους οντολογίας με pyshacl 80](#_Toc139114986)

[Εικόνα 47 – Αποτελέσματα του validation 80](#_Toc139114987)

[Εικόνα 48 – Εντοπισμός λάθους από validation 81](#_Toc139114988)

[Εικόνα 49 – GraphDB Workbench home page 82](#_Toc139114989)

[Εικόνα 50 – Πληροφορίες για το ενεργό repository 82](#_Toc139114990)

[Εικόνα 51 – SPARQL page του GraphDB Workbench 83](#_Toc139114991)

[Εικόνα 52 – GraphDB 83](#_Toc139114992)

[Εικόνα 53 – Δημιουργία του server με Express 85](#_Toc139114993)

[Εικόνα 54 – Αρχική σελίδα τους εφαρμογής 86](#_Toc139114994)

[Εικόνα 55 – Αρχική όψη τους σελίδας Dashboards 86](#_Toc139114995)

[Εικόνα 56 – SPARQL query που στέλνεται στην βάση 87](#_Toc139114996)

[Εικόνα 57 – Πίνακας με τα αποτελέσματα ερωτήματος 87](#_Toc139114997)

[Εικόνα 58 – Γράφημα για SedentaryMinutes 88](#_Toc139114998)

[Εικόνα 59 – Γράφημα για LightlyActiveMinutes 88](#_Toc139114999)

[Εικόνα 60 – SPARQL ερώτημα που εκτελείται 89](#_Toc139115000)

[Εικόνα 61 – Πίνακας αποτελεσμάτων 89](#_Toc139115001)

[Εικόνα 62 – Γράφημα αποτελεσμάτων 89](#_Toc139115002)

1. Εισαγωγή

Εισαγωγή

# Γενική Ιδέα

Αντικείμενο της παρούσας εργασίας είναι η χρήση τεχνολογιών του Σημασιολογικού Ιστού για την επεξεργασία και την αναπαράσταση δεδομένων από συσκευές παρακολούθησης δραστηριότητας, ή αλλιώς fitness trackers.

Ο Σημασιολογικός Ιστός είναι μια έννοια που αναφέρεται στην επέκταση του παγκόσμιου ιστού, πέρα από την απλή απεικόνιση πληροφοριών, προσδίδοντας σημασιολογική σημασία στα δεδομένα. Αυτό σημαίνει ότι οι πληροφορίες που περιέχονται στον ΣΙ είναι δομημένες και κατανοητές από τους υπολογιστές, επιτρέποντας τους να κατανοούν το περιεχόμενο και το νόημα των πληροφοριών πέρα από απλές αναζητήσεις κειμένου. Ο ΣΙ αποτελεί σημαντικό στοιχείο για τη βελτίωση της επίδοσης και της αποδοτικότητας των λειτουργιών του ιστού, καθώς μπορεί να βελτιώσει την ακρίβεια και την αποτελεσματικότητα των αναζητήσεων στον ιστό, να διευκολύνει την αυτόματη κατηγοριοποίηση και ανάλυση πληροφοριών, καθώς και να παρέχει πλούσια και εξατομικευμένη εμπειρία στους χρήστες.

Τα fitness trackers είναι φορητές συσκευές που χρησιμοποιούνται για την παρακολούθηση και τη μέτρηση της σωματικής δραστηριότητας και της υγείας. Συνήθως φοριούνται στον καρπό ή στο σώμα και συλλέγουν δεδομένα όπως αριθμός βημάτων, καρδιακή συχνότητα, ύπνος, κατανάλωση θερμίδων και άλλες παραμέτρους υγείας. Τα δεδομένα αυτά συνήθως συγχρονίζονται με μια εφαρμογή σε smartphone ή τον υπολογιστή του χρήστη για να παρουσιαστούν και να αναλυθούν. Τα fitness trackers μπορούν να συμβάλλουν σημαντικά στη παρακολούθηση αλλά και την βελτίωση της υγείας των χρηστών καθώς μπορούν να παρέχουν πολύτιμες πληροφορίες για τη φυσική τους κατάσταση, να παρακινήσουν την άσκηση και την υγιεινή ζωή, και να βοηθήσουν στην παρακολούθηση και διαχείριση της υγείας και των γυμναστικών στόχων. Η αναπαράσταση των δεδομένων από τα επιλεγμένα fitness trackers στην εργασία θα δίνει την δυνατότητα στον χρήστη να θέτει ερωτήματα πάνω στα δεδομένα, ώστε να βλέπει την πρόοδο των χρηστών και να εξάγει τα συμπεράσματα του σχετικά με την χρήση των συσκευών αυτών.

Τα δεδομένα που χρησιμοποιούνται εξήχθησαν από ένα ανοιχτό dataset με δεδομένα χρηστών που χρησιμοποίησαν κάποια συσκευή παρακολούθησης δραστηριότητας. Μετά την εξαγωγή τους, τα δεδομένα πρέπει να μετασχηματιστούν από το μοντέλο του dataset σε γράφους γνώσης RDF, ώστε να αποθηκευτούν σε μια σημασιολογική βάση δεδομένων. Για την μετατροπή αυτή υλοποιείται μια οντολογία, συμβατή με το μοντέλο SSN-SOSA, που να μπορεί να αναπαραστήσει με κατάλληλο τρόπο τα δεδομένα.

Για την οπτικοποίηση των δεδομένων και την παρουσίαση τους στον χρήστη, κατασκευάστηκε μία web εφαρμογή που επιτρέπει στον χρήστη να θέτει τα ερωτήματα του πάνω στα δεδομένα. Η διαδικασία αυτή γίνεται μέσω της γλώσσας SPARQL που καθιστά δυνατή την επικοινωνία των χρηστών με σημασιολογικές βάσεις.

Γενικότερα, ο σκοπός της εργασίας είναι η κατανόηση της έννοιας του Σημασιολογικού Ιστού και η εκμάθηση των βασικών τεχνολογιών της με στόχο να είναι δυνατή η ανάλυση πραγματικών δεδομένων και η εξαγωγή πραγματικών και χρήσιμων συμπερασμάτων.

# Ενότητες Εργασίας

Η εργασία δομείται σε κεφάλαια ως εξής:

* Στο Κεφάλαιο 2 περιγράφεται λεπτομερώς η έννοια του Σημασιολογικού Ιστού. Αναλύεται η προέλευση του, ο λόγος που τον χρειαζόμαστε, ο τρόπος χρήσης και διάφορες εφαρμογές του, καθώς και οι βασικές τεχνολογίες του, RDF, OWL, SPARQL και οι σημασιολογικές βάσεις δεδομένων, που χρησιμοποιήθηκαν και στα πλαίσια της εργασίας.
* Στο Κεφάλαιο 3, αρχικά περιγράφεται ο όρος των συσκευών παρακολούθησης δραστηριότητας ή “fitness trackers”. Αναλύεται ο τρόπος λειτουργίας καθώς και οι βασικοί τύποι τους. Στη συνέχεια αναλύονται με λεπτομέρεια οι οντολογίες SSN και SOSA. Παρουσιάζονται κάποια βασικά στοιχεία του συντακτικού τους, όπως παρατηρήσεις (observations) και χαρακτηριστικά ενδιαφέροντος (features of interest), τα οποία χρησιμοποιούνται αργότερα και στην υλοποίηση.
* Στο Κεφάλαιο 4 παρουσιάζεται ο τρόπος υλοποίησης όλων των σταδίων της εργασίας. Αρχικά περιγράφεται το πρόβλημα και παρουσιάζονται τα εργαλεία και οι τεχνολογίες που χρησιμοποιήθηκαν για την επίλυση του, καθώς και τα δεδομένα. Στην συνέχεια, αναλύεται η οντολογία που κατασκευάστηκε καθώς και ο τρόπος κατασκευής της με python. Έπειτα, αναλύεται ο μηχανισμός μετασχηματισμού των δεδομένων, ο οποίος διαβάζει τα δεδομένα από τα csv αρχεία, κάνει τις κατάλληλες αλλαγές και τα αποθηκεύει ως τριπλέτες στην οντολογία. Ακολουθεί ο μηχανισμός επικύρωσης των δεδομένων, όπου αναλύονται οι κανόνες που δημιουργήθηκαν σε SHACL για την επικύρωση της οντολογίας και η αποθήκευση των δεδομένων σε μία βάση δεδομένων GraphDB. Τέλος, περιγράφεται η web εφαρμογή που δημιουργήθηκε για την οπτικοποίηση των αποτελεσμάτων, η οποία υλοποιήθηκε με χρήση τεχνολογιών όπως Angular, Node.js και Express.js. Ο κώδικας και όλα τα αρχεία που χρησιμοποιήθηκαν για την υλοποίηση όλων των τμημάτων της εργασίας είναι διαθέσιμα σε αυτό το GitHub repository: <https://github.com/AndreasNbt/FitnessData>
* Στο Παράρτημα I παρουσιάζονται αλφαβητικά η βιβλιογραφία και οι δικτυακοί τόποι που αναφέρονται στην εργασία.
* Στο Παράρτημα IΙ παρουσιάζονται τα ακρωνύμια τα οποία χρησιμοποιούνται σε αυτή την εργασία για την διευκόλυνση του αναγνώστη.

1. Σημασιολογικοσ Ιστοσ

ΣΗΜΑΣΙΟΛΟΓΙΚΟΣ ΙΣΤΟΣ

# Εισαγωγή

Το Διαδίκτυο είναι ένα παγκόσμιο δίκτυο διασυνδεδεμένων υπολογιστών, οι οποίοι επικοινωνούν μεταξύ τους χρησιμοποιώντας κοινά πρωτόκολλα επικοινωνίας. Οι χρήστες του Διαδικτύου μπορούν εύκολα και γρήγορα να περιηγηθούν σε μια τεράστια βάση πληροφοριών, να αποστείλουν και να λάβουν αρχεία, να κάνουν χρήση της ηλεκτρονικής αλληλογραφίας, και γενικά να χρησιμοποιήσουν ένα πλήθος πολυάριθμων υπηρεσιών που έχουν στη διάθεσή τους. [1]

Ο Παγκόσμιος Ιστός (World Wide Web – WWW) είναι μία από τις βασικές υπηρεσίες του Διαδικτύου. Αποτελείται από το σύνολο των ιστοσελίδων που περιέχονται στους υπολογιστές του Διαδικτύου, οπότε είναι ένα σύνολο πολυμεσικών αρχείων που αποτελούνται από κείμενο, εικόνες, βίντεο, ήχο και όλα τα άλλα μέσα που χρησιμοποιούνται για την κατασκευή των σημερινών ιστοσελίδων. Αυτά τα μέσα είναι πολύ χρήσιμα για τους ανθρώπους, που μπορούν να τα επεξεργαστούν και να εκτελέσουν διάφορες εργασίες με αυτά, αλλά δεν δίνουν πολλές δυνατότητες στους υπολογιστές. Οι υπολογιστές μπορούν να τα προβάλλουν, να τα απαριθμήσουν, να τα μεταφέρουν αλλά μέχρι εκεί. [2]

Diagram, engineering drawing

Description automatically generated

Εικόνα 1 - Διαδίκτυο: Δίκτυο Διασυνδεδεμένων Υπολογιστών

Ο ΣΙ αποτελεί επέκταση του σημερινού ιστού και έχει ως στόχο να κάνει τα δεδομένα πιο κατανοητά και πιο χρήσιμα για τους υπολογιστές. Ένας τέτοιος ιστός θα διευρύνει τις δυνατότητες του σημερινού ιστού, κάνοντας τον πιο δυναμικό και επεκτείνοντας βασικές λειτουργίες όπως η αναζήτηση και η διασύνδεση δεδομένων, με την χρήση της σημασιολογίας. Η αναζήτηση δε θα περιορίζεται πλέον μόνο στην εύρεση λέξεων κλειδιών, αλλά θα μπορεί να γίνει πιο σημασιολογική, ψάχνοντας για συνώνυμα, λαμβάνοντας υπόψη τα συμφραζόμενα και τον σκοπό του ερωτήματος.

## Ιστορική Αναδρομή

Ένα Σημασιολογικό Δίκτυο αποτελεί μια βάση γνώσης που αντιπροσωπεύει τις σημασιολογικές σχέσεις μεταξύ των εννοιών που ανήκουν σε αυτό. Η έννοια του ΣΔ διαμορφώθηκε στις αρχές της δεκαετίας του 1960 από ερευνητές όπως ο γνωστικός επιστήμονας Allan M. Collins, ο γλωσσολόγος M. Ross Quillian και η ψυχολόγος Elizabeth F. Loftus ως μορφή αναπαράστασης σημασιολογικά δομημένης γνώσης. Όταν εφαρμόζεται στο πλαίσιο του σύγχρονου Διαδικτύου, επεκτείνει το δίκτυο των συνδεδεμένων, αναγνώσιμων από τον άνθρωπο, ιστοσελίδων εισάγοντας μεταδεδομένα [3], τα οποία είναι δεδομένα που γίνονται κατανοητά από τον υπολογιστή και περιγράφουν τις σελίδες και το πώς σχετίζονται μεταξύ τους. Αυτό επιτρέπει στους αυτοματοποιημένους πράκτορες να έχουν πιο έξυπνη πρόσβαση στον Ιστό και να εκτελούν περισσότερες και πιο πολύπλοκες εργασίες για λογαριασμό των χρηστών. Η έννοια αυτή έδωσε την βάση για την μετέπειτα θεμελίωση του Σημασιολογικού Ιστού.

Ο όρος «Σημασιολογικός Ιστός» επινοήθηκε από τον Tim Berners-Lee, τον εφευρέτη του Παγκόσμιου Ιστού και διευθυντή της Κοινοπραξίας του Παγκόσμιου Ιστού (“W3C”)[[1]](#footnote-1), η οποία επιβλέπει την ανάπτυξη των προτεινόμενων προτύπων του Σημασιολογικού Ιστού. Ορίζει τον ΣΙ ως «έναν ιστό δεδομένων που μπορεί να υποβληθεί σε επεξεργασία άμεσα και έμμεσα από μηχανές». [4]

Ο Berners-Lee είχε οραματιστεί τον Σημασιολογικό Ιστό τουλάχιστον το 1994, μόλις λίγα χρόνια αφότου άρχισε να αναπτύσσει το WWW το 1989. Αποκάλυψε την ιδέα του για τον Σημασιολογικό Ιστό στο Πρώτο Διεθνές Συνέδριο WWW, που πραγματοποιήθηκε το 1994, το οποίο είχε ως αποτέλεσμα τη δημιουργία του W3C.

Όπως το είδε ο Berners-Lee, τα δύο κλειδιά για την ανάπτυξη ενός πραγματικά χρήσιμου αποθετηρίου πληροφοριών απαιτούσαν τη συμπερίληψη μεταδεδομένων ή πληροφοριών σχετικά με τις πληροφορίες που βρίσκονται στον Ιστό, που θα μπορούσαν να διαβαστούν και να «κατανοηθούν» από τις μηχανές, καθώς και την επισύναψη «αξιών» σε υπερσυνδέσμους που θα μπορούσαν να χρησιμοποιήσουν οι υπολογιστές για να κατευθύνουν αναζητήσεις.

Με την δημιουργία του W3C ξεκίνησαν να αναπτύσσονται όλα τα εργαλεία και οι τεχνολογίες που θα χρησιμοποιούνταν έπειτα για την υλοποίηση του ΣΙ.

Το 1998 κυκλοφόρησε η πρώτη έκδοση της XML[[2]](#footnote-2) (eXtensible Markup Language). Η XML παρέχει μια στοιχειώδη σύνταξη για τη δομή του περιεχομένου μέσα στα έγγραφα. Παρέχει την δυνατότητα για την προσθήκη σημασιολογίας στα αντικείμενα του εγγράφου, ωστόσο δεν τη συσχετίζει με την πραγματική έννοια του περιεχομένου που περιέχεται μέσα, υπάρχει απλά για την διευκόλυνση του αναγνώστη και την ύπαρξη δομής στο έγγραφο. Η XML δεν αποτελεί πλέον απαραίτητο συστατικό των τεχνολογιών του ΣΙ στις περισσότερες περιπτώσεις, καθώς υπάρχουν εναλλακτικά συντακτικά, όπως το Turtle [[3]](#footnote-3).

Το 1999 κυκλοφόρησε η πρώτη έκδοση της RDF[[4]](#footnote-4) (Resource Description Framework). Αποτελεί το θεμελιώδες πρότυπο του ΣΙ. Είναι μια απλή γλώσσα που χρησιμοποιείται για την περιγραφή διασυνδεδεμένων δεδομένων. Ένα μοντέλο που βασίζεται σε RDF μπορεί να αναπαρασταθεί με μία ποικιλία συντακτικών, τους RDF/XML ή Turtle. Αργότερα, δημιουργήθηκαν και άλλα πρότυπα βασισμένα στην RDF, τα οποία επέκτειναν όλο και περισσότερο τους δυνατότητες και την εκφραστικότητα τους γλώσσας, τους η RDFS[[5]](#footnote-5) (RDF-Schema) και η OWL[[6]](#footnote-6) (Web Ontology Language). Στην εικόνα 2 βλέπουμε την δομή του ΣΙ όσον αφορά τους τεχνολογίες που χρησιμοποιούνται για να τον υλοποιήσουν.

Diagram

Description automatically generated

Εικόνα 2 – The Semantic Web Stack

## Βασικές Έννοιες του Σημασιολογικού Ιστού

Ο Σημασιολογικός Ιστός (ή Ιστός των Δεδομένων, τους γίνεται γνωστός τα τελευταία χρόνια) ακολουθεί διαφορετικές αρχές σχεδιασμού, οι οποίες μπορούν να συνοψιστούν ως εξής:

* Καθιστά δομημένα και ημι-δομημένα δεδομένα διαθέσιμα σε τυποποιημένες μορφές στον Ιστό.
* Καθιστά, όχι μόνο τα σύνολα δεδομένων, αλλά και τα μεμονωμένα δεδομένα-στοιχεία και τους σχέσεις τους προσβάσιμες στον Ιστό και,
* Περιγράφει την σημασιολογία αυτών των δεδομένων με φορμαλισμό, ώστε η σημασιολογία που επιδιώκεται να είναι επεξεργάσιμη από τους μηχανές. [5]

Οι αρχές σχεδιασμού που αναφέρθηκαν επιτυγχάνονται με τη χρήση διάφορων βασικών τεχνολογιών που χαρακτηρίζουν τον ΣΙ και ταυτόχρονα τον διαφοροποιούν από τον υπάρχοντα Ιστό.

Ο ΣΙ χρησιμοποιεί:

* Γραφήματα με ετικέτα (labeled graphs) [6] ως μοντέλο για τα αντικείμενα και τους σχέσεις τους. Τα αντικείμενα αποτελούν τους κόμβους του γραφήματος και οι ακμές τους σχέσεις μεταξύ τους. Για την αναπαράσταση αυτών των γράφων χρησιμοποιείται η γλώσσα RDF.
* Uniform Resource Identifiers, ή αλλιώς URIs [7], τα οποία είναι αναγνωριστικά που καθορίζουν μοναδικά ένα αντικείμενο του σημασιολογικού ιστού.
* Οντολογίες ως μοντέλο για την τυπική αναπαράσταση τους επιδιωκόμενης σημασιολογίας των δεδομένων. Τεχνολογίες τους η RDF Schema (RDFS) ή OWL χρησιμοποιούνται για την κατασκευή των οντολογιών, χρησιμοποιώντας URIs για την αναπαράσταση των τύπων και των ιδιοτήτων τους.

## Διασυνδεδεμένα Δεδομένα (Linked Data)

Αποτελεί θεμελιώδη έννοια του Σημασιολογικού Ιστού. Αποτελούν δομημένα δεδομένα τα οποία συνδέονται μεταξύ τους, ώστε να γίνονται πιο χρήσιμα με την εκτέλεση σημασιολογικών ερωτημάτων. Βασίζονται σε τυπικές τεχνολογίες τους HTTP, RDF και URI, και τους χρησιμοποιούν ώστε να γίνεται διαμοιρασμός των πληροφοριών με τρόπο που να μην εξυπηρετεί μόνο τους χρήστες, αλλά και τους υπολογιστές. Μέρος του οράματος των ΔΔ είναι να γίνει το Διαδίκτυο μια παγκόσμια βάση δεδομένων.

Η έννοια επινοήθηκε και πάλι από τον Tim Berners-Lee, ο οποίος ανέφερε και τέσσερις βασικές αρχές σχετικά με την χρήση των διασυνδεδεμένων δεδομένων [8]:

1. Πρέπει να γίνεται χρήση των URI για την ονομασία και την αναγνώριση μεμονωμένων πόρων.
2. Θα πρέπει να χρησιμοποιούνται HTTP URIs για να επιτρέπεται η αναζήτηση και η ερμηνεία αυτών των πόρων.
3. Χρήσιμες πληροφορίες σχετικά με τον πόρο που προσδιορίζεται θα πρέπει να γίνονται διαθέσιμες μέσω ανοιχτών προτύπων τους RDF και SPARQL.
4. Κατά την δημοσίευση δεδομένων στον Ιστό, τους οι αναφορές σε τους πόρους θα πρέπει να γίνονται χρησιμοποιώντας τα HTTP URIs των πόρων.

### Ανοιχτά Διασυνδεδεμένα Δεδομένα

Τα Ανοικτά Διασυνδεδεμένα Δεδομένα (ΑΔΔ) είναι ΔΔ που κυκλοφορούν με ανοιχτή άδεια, η οποία δεν εμποδίζει την επαναχρησιμοποίησή τους δωρεάν.

Ο Tim Berners-Lee πρότεινε ένα σχέδιο 5 αστέρων για την βαθμολόγηση των ανοιχτών δεδομένων που δημοσιεύονται στον Ιστό, με την υψηλότερη βαθμολογία να αποτελούν τα ΑΔΔ. Παρακάτω παρουσιάζονται παραδείγματα για κάθε βαθμό:

* 1 αστέρι: Δεδομένα που είναι διαθέσιμα στον Ιστό σε οποιαδήποτε μορφή, για παράδειγμα PDF.
* 2 αστέρια: Δεδομένα που είναι διαθέσιμα στον Ιστό σε οποιαδήποτε δομημένη μορφή, για παράδειγμα σε μορφή αρχείου Microsoft Excel (.xls).
* 3 αστέρια: Δεδομένα που είναι διαθέσιμα στον Ιστό σε μη ιδιόκτητη δομημένη μορφή, για παράδειγμα CSV αντί για Excel.
* 4 αστέρια: Δεδομένα που ακολουθούν τα πρότυπα τους W3C, τους χρήση RDF και URIs.
* 5 αστέρια: Όλα τα προηγούμενα, με επιπλέον προσθήκη συνδέσμων προς άλλες πηγές ΑΔΔ.

Στην εικόνα 3 δίνεται και μία οπτική απεικόνιση αυτού του σχεδίου. Τους φαίνεται, η μορφή των δεδομένων αλλάζει και γίνεται πιο δομημένη όσο ανεβαίνουμε επίπεδο.

Timeline

Description automatically generated

Εικόνα 3 – 5 star deployment scheme for Open Data[[7]](#footnote-7)

# Resource Description Framework (RDF)

Η RDF αποτελεί το θεμελιώδες μοντέλο για την αναπαράσταση και τη διαχείριση δεδομένων στον ΣΙ. Έχει χαρακτηριστικά που διευκολύνουν την αναπαράσταση κατανεμημένης πληροφορίας καθώς και τη συγχώνευση δεδομένων ακόμα και αν τα υποκείμενα σχήματα τους διαφέρουν. Επίσης, καθιστά εύκολη την εξέλιξη των σχημάτων με την πάροδο του χρόνου χωρίς να απαιτείται αλλαγή των υπαρχόντων δεδομένων.

Επεκτείνει τη δομή σύνδεσης του Ιστού, δίνοντας ένα URI στη σχέση που συνδέει δύο πόρους, καθώς και στους δύο πόρους που συνδέονται. Η διαδικασία αυτή αποτελεί τον βασικό τρόπο αναπαράστασης των δεδομένων της RDF και έχει ως αποτέλεσμα τη δημιουργία μίας RDF τριπλέτας (RDF triple). Η τριπλέτα είναι η ατομική οντότητα δεδομένων της RDF. Όπως υποδηλώνει το όνομα της, είναι ένα σύνολο τριών οντοτήτων που κωδικοποιεί μια δήλωση σχετικά με κάποια σχέση μεταξύ σημασιολογικών δεδομένων.

Αποτελείται από τρία μέρη:

* Υποκείμενο (subject)
* Κατηγόρημα (predicate)
* Αντικείμενο (object)

Text

Description automatically generated with medium confidence

Εικόνα 4 – Δομή τους RDF triple

Η RDF στην ουσία είναι ένας κατευθυνόμενος γράφος. Κάθε τριπλέτα είναι μια ακμή που συνδέει το subject με το object μέσω του predicate. Παράδειγμα ενός RDF γράφου φαίνεται στην εικόνα 5. Όλοι οι πόροι που είναι διαθέσιμοι στην οντολογία βρίσκονται στα μπλε πλαίσια και συνδέονται μεταξύ τους μέσω των ακμών. Έτσι, δημιουργούνται διάφορα RDF triples, με το subject και το object να καθορίζονται από την κατεύθυνση των ακμών και με predicate το label της αντίστοιχης ακμής. Τα πράσινα πλαίσια αποτελούν σταθερές τιμές, όπως συμβολοσειρές ή ημερομηνίες. Για παράδειγμα, μία τριπλέτα που μπορούμε να εξάγουμε από τον παρακάτω γράφο, κοιτώντας στο πάνω αριστερά μέρος του, είναι η:

**:The\_Beatles rdf:type :Band**

Ο λόγος που μπορούμε να εξάγουμε αυτή τη τριπλέτα είναι επειδή οι δύο κόμβοι :The\_Beatles και :Band συνδέονται μέσω μίας ακμής η οποία έχει label rdf:type. Οπότε, το predicate είναι το rdf:type, και κοιτώντας την κατεύθυνση της ακμής, μπορούμε να συμπεράνουμε ότι το subject είναι ο πόρος :The\_Beatles και το object ο πόρος :Band.

Diagram

Description automatically generated

Εικόνα 5 – Παράδειγμα τους RDF graph[[8]](#footnote-8)

## Σύνταξη τους RDF

Υπάρχουν διάφορα συντακτικά για την σειριοποίηση ενός RDF γράφου σε κείμενο. Ένας τρόπος είναι η σειριοποίηση βασισμένη σε XML, όπως φαίνεται στην εικόνα 6:

Graphical user interface, text, application, email

Description automatically generated

Εικόνα 6 – RDF/XML syntax

Όπως φαίνεται όμως, δεν είναι και τόσο φιλική για τον άνθρωπο, για αυτό έχει αντικατασταθεί πλήρως με τη σειριοποίηση Turtle.

Στην Turtle, η κάθε τριπλέτα καταλαμβάνει μία γραμμή, παραθέτοντας το subject, το predicate και το object χωρισμένα με κενό και προσθέτοντας μια τελεία (“.”) στο τέλος. Παράδειγμα χρήσης του Turtle syntax φαίνεται στην εικόνα 7, όπου βλέπουμε την σύνταξη 10 τριπλετών.

Graphical user interface, text, application

Description automatically generated

Εικόνα 7 – Turtle syntax

Ένα μειονέκτημα που βλέπουμε στην απεικόνιση αυτή είναι η επανάληψη όρων. Η επανάληψη των subjects και των predicates για κάθε τριπλέτα περιπλέκει το συντακτικό και μειώνει την αναγνωσιμότητα του. Για αυτόν τον λόγο, η Turtle περιέχει κάποιες συντακτικές λειτουργίες που μειώνουν τους επαναλήψεις των ίδιων όρων και την κάνουν πιο ευανάγνωστη:

* Αν δύο τριπλέτες έχουν κοινό subject, τότε η πρώτη τριπλέτα τερματίζεται με ερωτηματικό (‘;’) και το subject τους επόμενης τριπλέτας μπορεί να παραλειφθεί.
* Αν δύο τριπλέτες έχουν κοινό subject και predicate τότε μπορούν να γραφτούν ως μία τριπλέτα, χωρίζοντας τα objects με κόμμα, χωρίς να γίνεται επανάληψη του subject και του predicate.
* Το **rdf:type** χρησιμοποιείται για να δηλώσει τον τύπο τους πόρου, και μπορεί να αντικατασταθεί με το γράμμα a.

Εφαρμόζοντας τους τους κανόνες προκύπτει ένα πιο απλό και ευανάγνωστο αποτέλεσμα.

Graphical user interface, application, Word

Description automatically generated

Εικόνα 8 – Turtle syntax με την εφαρμογή των κανόνων

# RDF Schema (RDFS)

Είναι μια σημασιολογική επέκταση της RDF. Επεκτείνει το βασικό λεξιλόγιο της RDF δίνοντας έτσι περισσότερες δυνατότητες μοντελοποίησης και παρέχει μηχανισμούς για την περιγραφή ομάδων σχετικών πόρων καθώς και των σχέσεων μεταξύ αυτών των πόρων.

Το σύστημα κλάσεων και ιδιοτήτων της RDFS είναι παρόμοιο με τα συστήματα των αντικειμενοστραφών γλωσσών προγραμματισμού όπως η Java. Η RDFS διαφέρει από πολλά τέτοια συστήματα στο ότι αντί να ορίζει μια κλάση ως προς τις ιδιότητες που μπορεί να έχουν τα στιγμιότυπά τους, περιγράφει ιδιότητες ως προς τις κατηγορίες πόρων στις οποίες εφαρμόζονται. Αυτός είναι ο ρόλος των μηχανισμών domain και range που θα περιγραφούν, οι οποίοι θυμίζουν τους όρους του πεδίου ορισμού και πεδίου τιμών από τα μαθηματικά.

Για παράδειγμα, θα μπορούσαμε να ορίσουμε την ιδιότητα eg:author ώστε να έχει domain ένα eg:Document και range eg:Person, ενώ ένα κλασικό αντικειμενοστραφές σύστημα μπορεί συνήθως να ορίζει μια κλάση eg:Book με ένα χαρακτηριστικό που ονομάζεται eg:author τύπου π.χ.: eg:Person.

## Λεξιλόγιο της RDFS

### Κλάσεις

**rdfs:Resource**

* Όλα τα αντικείμενα που περιγράφονται με RDF καλούνται πόροι (resources) και ανήκουν στην κλάση Resource.
* Είναι η κλάση που περιλαμβάνει τα πάντα. Όλες οι άλλες κλάσεις είναι υποκλάσεις της.

**rdfs:Class**

* Περιλαμβάνει όλους τους πόρους που είναι κλάσεις.
* Μία κλάση προσδιορίζει ένα σύνολο ομοειδών οντοτήτων, δηλαδή οντότητες που μοιράζονται τις ίδιες ιδιότητες και χαρακτηριστικά.
* Χρήση του **rdf:type** για να δώσω τύπο σε έναν πόρο.

**rdf:Property**

* Η κλάση των RDF ιδιοτήτων.

### Ιδιότητες

**rdf:type**

* Αποτελεί στιγμιότυπο της rdf:Property και χρησιμοποιείται για να δηλώσει ότι ένας πόρος είναι στιγμιότυπο μίας κλάσης.
* Μια τριπλέτα της μορφής **R rdf:type C** δηλώνει ότι το **C** είναι στιγμιότυπο της rdfs:Class και ότι το **R** είναι στιγμιότυπο της κλάσης **C**.
* Ανήκει στο βασικό λεξιλόγιο της RDF.

**rdfs:domain**

* Αποτελεί στιγμιότυπο της rdf:Property και εφαρμόζεται σε ιδιότητες. Χρησιμοποιείται για να δηλώσει ότι οι πόροι που έχουν κάποια ιδιότητα είναι στιγμιότυπα μίας ή παραπάνω κλάσεων.
* Μια τριπλέτα της μορφής **P rdfs:domain C** δηλώνει ότι:
  + **P** είναι στιγμιότυπο της rdf:Property.
  + **C** είναι στιγμιότυπο της rdfs:Class.
  + Οι πόροι στους οποίους αντιστοιχούν τα subjects των τριπλετών που έχουν ως predicate το **P** είναι στιγμιότυπα της κλάσης **C**.
* Όταν μια ιδιότητα P έχει περισσότερα από ένα rdfs:domain, τότε οι πόροι που συμβολίζονται από τα subjects των τριπλετών με predicate P είναι στιγμιότυπα όλων των κλάσεων που δηλώνονται από τα rdfs:domain.

**rdfs:range**

* Αποτελεί στιγμιότυπο της rdf:Property και χρησιμοποιείται για να δηλώσει ότι οι τιμές μιας ιδιότητας είναι στιγμιότυπα μίας ή περισσότερων κλάσεων.
* Μια τριπλέτα τους μορφής **P rdfs:range C** δηλώνει ότι:
  + **P** είναι στιγμιότυπο τους rdf:Property.
  + **C** είναι στιγμιότυπο τους rdfs:Class.
  + Οι πόροι τους οποίους αντιστοιχούν τα objects των τριπλετών που έχουν ως predicate το **P** είναι στιγμιότυπα τους κλάσης **C**.
* Όταν μια ιδιότητα P έχει περισσότερα από ένα rdfs:range, τότε οι πόροι που συμβολίζονται από τα αντικείμενα των τριπλετών με κατηγόρημα P είναι στιγμιότυπα όλων των κλάσεων που δηλώνονται από τα rdfs:range.
* Επίσης, το range μπορεί να είναι και κάποιος πρωτογενής τύπος δεδομένων, όπως αριθμός ή συμβολοσειρά.

Οι δύο αυτές ιδιότητες ουσιαστικά αντιστοιχούν στις έννοιες του πεδίου ορισμού και πεδίου τιμών που είναι γνωστές από τα μαθηματικά.

Έστω ότι έχουμε τις τριπλέτες:

**:takesCourse rdfs:domain :Student**

**:takesCourse rdfs:range :Course**

**:Andreas :takesCourse :SemanticWeb**

Με βάση τη σημασιολογία των ιδιοτήτων domain και range, εξάγονται αυτόματα οι ακόλουθες τριπλέτες ως συμπεράσματα:

**:Andreas rdf:type :Student**

**:SemanticWeb rdf:type :Course**

Τέλος, πρέπει να σημειωθεί ότι τα domain και range δεν αποτελούν περιορισμούς, καθώς στην RDFS δεν γίνεται να απαγορευτεί σε κάποιο στιγμιότυπο να είναι μέλος μιας κλάσης. Χρησιμοποιούνται απλά για την περιγραφή του σχήματος και δεν μπορούν να εγγυηθούν την εγκυρότητα του αποτελέσματος.

**rdfs:subClassOf**

* Χρησιμοποιείται για να δηλώσει ότι μία κλάση είναι υποκλάση μίας άλλης, οπότε και όλα τα στιγμιότυπα της υποκλάσης θα ανήκουν και στην ανώτερη κλάση.
* Μπορούμε λοιπόν να χρησιμοποιήσουμε αυτή την ιδιότητα για να δημιουργήσουμε μία ιεραρχία μεταξύ των κλάσεων.
* Μια τριπλέτα της μορφής **C1 rdfs:subClassOf C2** δηλώνει ότι:
  + **C1** και **C2** είναι στιγμιότυπα της rdfs:Class.
  + Η κλάση **C1** είναι υποκλάση της **C2**, άρα όλα τα στιγμιότυπα της **C1** θα είναι και στιγμιότυπα της **C2**.
* Είναι μεταβατική ιδιότητα.
* Για παράδειγμα, αν έχουμε τους ακόλουθες τριπλέτες:

**:MobaGame rdfs:subClassOf :MultiplayerGame**

**:MultiplayerGame rdfs:subClassOf :Videogame**

**:LeagueOfLegends rdf:type :MobaGame**

Τότε θα προκύψουν τα ακόλουθα ως συμπέρασμα της μεταβατικότητας:

**:LeagueOfLegends rdf:type :MultiplayerGame**

**:LeagueOfLegends rdf:type :Videogame**

**rdfs:subPropertyOf**

* Αντίστοιχα, χρησιμοποιείται για να δηλώσει ότι μία ιδιότητα είναι υπο-ιδιότητα μίας άλλης. Ως αποτέλεσμα, οι πόροι που συσχετίζονται μέσω κάποιας ιδιότητας, θα συσχετίζονται και μέσω της ανώτερης ιδιότητας.
* Για παράδειγμα:
  + :brother rdfs:subPropertyOf :relative
  + Άρα, οι πόροι που συσχετίζονται με την ιδιότητα :brother θα συσχετίζονται και με την ιδιότητα :relative.
* Είναι επίσης μεταβατική ιδιότητα.

**rdfs:label**

* Χρησιμοποιείται για να δώσει ένα όνομα σε κάθε πόρο.
* Οι πόροι στον ΣΙ καθορίζονται από τα URIs, τα οποία συνήθως δεν είναι φιλικά για τον άνθρωπο.

# Web Ontology Language (OWL)

Η OWL είναι μία γλώσσα για την δημιουργία και αναπαράσταση οντολογιών στον ΣΙ. Αποτελεί πρότυπο του W3C και έχει σχεδιαστεί έτσι ώστε να μπορεί να χρησιμοποιηθεί με την RDF και την RDFS, για να επιτρέπει τη δημιουργία πιο περίπλοκων, εκφραστικών και κλιμακωτών οντολογιών.

Μια οντολογία είναι μια επίσημη περιγραφή των εννοιών και των σχέσεων σε έναν συγκεκριμένο τομέα. Η OWL παρέχει ένα πλούσιο σύνολο δομών για την αναπαράσταση οντολογιών, το οποίο συμπεριλαμβάνει έννοιες όπως κλάσεις, ιδιότητες, στιγμιότυπα και τις σχέσεις μεταξύ τους. Η OWL παρέχει επίσης μια επίσημη σημασιολογία, βασισμένη στη λογική περιγραφής, η οποία επιτρέπει την αυτοματοποιημένη συλλογιστική και τον έλεγχο λογικής συνέπειας των οντολογιών.

## Διαφορές με RDFS

Ενώ τόσο η RDFS όσο και η OWL χρησιμοποιούνται για την μοντελοποίηση και την αναπαράσταση γνώσης στον ΣΙ, υπάρχουν ορισμένες βασικές διαφορές μεταξύ των δύο γλωσσών:

1. **Εκφραστικότητα**: Η OWL είναι πιο εκφραστική από την RDFS, πράγμα που σημαίνει ότι μπορεί να μοντελοποιεί πιο σύνθετες σχέσεις μεταξύ οντοτήτων. Δίνει περισσότερες και πιο προηγμένες δυνατότητες μοντελοποίησης εννοιών, όπως ο καθορισμός σύνθετων ιεραρχιών κλάσεων, η δυνατότητα ορισμού περιορισμών, και πολλά άλλα.
2. **Λεξιλόγιο**: Η OWL έχει μεγαλύτερο λεξιλόγιο από την RDFS, με περισσότερες προκαθορισμένες κλάσεις και ιδιότητες διαθέσιμες. Αυτό διευκολύνει τη δημιουργία οντολογιών χωρίς να χρειάζεται να οριστεί κάθε έννοια και ιδιότητα από την αρχή.
3. **Συλλογιστική**: Η OWL υποστηρίζει πιο προηγμένες δυνατότητες συλλογιστικής από την RDFS, συμπεριλαμβανομένων πιο εξελιγμένων δυνατοτήτων εξαγωγής συμπερασμάτων. Αυτό σημαίνει ότι τα συστήματα που βασίζονται σε OWL μπορούν να εξάγουν καλύτερα και πιο περίπλοκα συμπεράσματα από τα δεδομένα έναντι των συστημάτων που βασίζονται σε RDFS.
4. **Τυπική Σημασιολογία**: Η OWL έχει μία επίσημη και καλά ορισμένη σημασιολογία που επιτρέπει την ύπαρξη αυτοματοποιημένης συλλογιστικής και ελέγχου λογικών ασυνεπειών, γεγονός το οποίο διασφαλίζει ότι οι οντολογίες είναι απαλλαγμένες από αντιφάσεις και ασυνέπειες. Η RDFS, από την άλλη, δεν έχει επίσημη σημασιολογία και είναι λιγότερο κατάλληλη για αυτοματοποιημένο συλλογισμό και ελέγχους ασυνεπειών.
5. **Περιπτώσεις Χρήσης**: Η OWL χρησιμοποιείται συχνά για πιο σύνθετες εφαρμογές που απαιτούν πιο προηγμένες δυνατότητες μοντελοποίησης και συλλογισμού, όπως βιοϊατρικές οντολογίες ή για την ενοποίηση επιστημονικών δεδομένων. Από την άλλη, η RDFS χρησιμοποιείται κυρίως για απλούστερες εφαρμογές που απαιτούν βασικές δυνατότητες μοντελοποίησης, όπως η περιγραφή και η κατηγοριοποίηση πόρων του ιστού.

## Τύποι Ιδιοτήτων

Το λεξιλόγιο της OWL μας επιτρέπει να ορίσουμε ιδιότητες που μοντελοποιούν διάφορες βασικές μαθηματικές σχέσεις. Οι ιδιότητες αυτές αυξάνουν κατά πολύ τις δυνατότητες συμπερασμού και δίνουν περισσότερο βάθος στην οντολογία.

Αντίστροφες Ιδιότητες

Χρήση τους ιδιότητας **owl:inverseOf** για να δηλώσουμε ότι δύο ιδιότητες είναι αντίστροφες.

Στα μαθηματικά, η αντιστροφή μιας συνάρτησης f είναι μια συνάρτηση για την οποία ισχύει ότι, αν f(x)=y, τότε f-1(y)=x. Ομοίως στην OWL, η αντιστροφή μιας ιδιότητας είναι μια άλλη ιδιότητα που αντιστρέφει τη φορά της πρώτης.

Η έννοια αυτή γίνεται εύκολα κατανοητή με ένα παράδειγμα. Αν έχουμε τους παρακάτω τριπλέτες:

**:Tolkien :wrote :LordOfTheRings .**

**:wrote owl:inverseOf :writtenBy .**

Τότε θα καταλήξουμε μέσω συμπερασμού στο ότι

**:LordOfTheRings :writtenBy :Tolkien**

Συμμετρικές Ιδιότητες

Το owl:inverseOf συσχετίζει μία ιδιότητα με κάποια άλλη. Η περίπτωση που αυτές οι δύο ιδιότητες είναι ίδιες είναι τόσο συχνή ώστε να υπάρχει επιπλέον λεξιλόγιο για να την μοντελοποιεί: **owl:SymmetricProperty**. Αποτελεί πτυχή μιας ιδιότητας και εκφράζεται στην OWL ως κλάση.

Εκφράζουμε ότι μια ιδιότητα είναι συμμετρική όπως εκφράζουμε και την ιδιότητα μέλους σε μία κλάση. Δηλαδή:

**:P rdf:type owl:SymmetricProperty .**

Παράδειγμα συμμετρικής ιδιότητας:

**bio:married rdf:type owl:SymmetricProperty**

Καθώς θέλουμε να υπάρχει η συσχέτιση married μεταξύ και των δύο προσώπων ως subjects. Οπότε, αν

**:x bio:married :y,** τότε ισχύει και

**:y bio:married :x.**

Χωρίς αυτή τη δυνατότητα, θα έπρεπε να δηλώσουμε ότι η ιδιότητα bio:married είναι αντίστροφη του εαυτού της.

Μεταβατικές Ιδιότητες

Η έννοια της μεταβατικότητας στα μαθηματικά περιγράφεται ως εξής:

* Αν A => B και B => C, τότε A => C

Στην OWL η έννοια αυτή περιγράφεται με το δομικό στοιχείο **owl:TransitiveProperty.**

**:P rdf:type owl:TransitiveProperty .**

Τυπικό παράδειγμα μεταβατικής ιδιότητας αποτελεί η σχέση προγόνου-απογόνου. Στην OWL λοιπόν, θα μπορούσε να μοντελοποιηθεί προσθέτοντας στην οντολογία την τριπλέτα:

**bio:hasAncestor rdf:type owl:TransitiveProperty .**

Για περισσότερη ευελιξία θα μπορούσαμε να δηλώσουμε και την αντίστροφη ιδιότητα:

**bio:hasAncestor owl:inverseOf bio:ancestorOf .**

Οπότε, αν

**:x bio:hasAncestor :y,** και

**:y bio:hasAncestor :z,** τότε ισχύει και

**:x bio:hasAncestor :z**.

Συναρτησιακές Ιδιότητες

Βασίζεται στην έννοια της συνάρτησης στα μαθηματικά, όπου κάθε συνάρτηση για μία τιμή της εισόδου μας δίνει μόνο μία τιμή ως έξοδο. Έτσι και στην OWL, μια συναρτησιακή ιδιότητα μπορεί να έχει μία μόνο τιμή για κάθε μεμονωμένο στιγμιότυπο.

**:P rdf:type owl:FunctionalProperty .**

Συχνά παραδείγματα αποτελούν ιδιότητες όπως hasMother ή hasBirthplace οι οποίες λαμβάνουν μία μόνο τιμή για κάθε στιγμιότυπο.

Μια επιπλέον δυνατότητα που δίνουν οι συναρτησιακές ιδιότητες, η οποία τις καθιστά πολύ χρήσιμες στην OWL, είναι η εύρεση ομοιότητας. Στην περίπτωση που μία συναρτησιακή ιδιότητα παίρνει δύο τιμές για κάποιο στιγμιότυπο, η OWL θα θεωρήσει ότι οι δύο αυτές τιμές είναι ισοδύναμες. Για παράδειγμα, αν έχουμε τις παρακάτω τριπλέτες:

**lit:Tolkien fam:hasMother bio:MabelTolkien .**

**lit:Tolkien fam:hasMother bio:MabelSuffield .**

**fam:hasMother rdf:type owl:FunctionalProperty .**

Τότε από τον ορισμό της συναρτησιακής ιδιότητας μπορούμε να συμπεράνουμε ότι τα δύο objects είναι ισοδύναμα:

**bio:MabelTolkien owl:sameAs bio:MabelSuffield .**

Αντίστροφες Συναρτησιακές Ιδιότητες

Έχουν ρόλο αντίστοιχο του πεδίου κλειδιού σε μία σχεσιακή βάση δεδομένων, δηλαδή αντιστοιχεί μία τιμή μίας ιδιότητας σε μία μόνο οντότητα. Στη περίπτωση όμως που βρεθούν δύο οντότητες που μοιράζονται την ίδια τιμή σε μία αντίστροφη συναρτησιακή ιδιότητα, η OWL δε θα εμφανίσει κάποιο μήνυμα λάθους, αλλά θα θεωρήσει ότι οι δύο οντότητες είναι ισοδύναμες. Οπότε, όπως οι συναρτησιακές ιδιότητες, και οι αντίστροφες τους μπορούν να χρησιμοποιηθούν για την διαπίστωση ομοιότητας και είναι συνήθως το κύριο δομικό στοιχείο που χρησιμοποιείται για τον λόγο αυτό.

**:P rdf:type owl:InverseFunctionalProperty .**

Συχνό παράδειγμα κάποιας αντίστροφης συναρτησιακής ιδιότητας αποτελούν οι αναγνωριστικοί αριθμοί (αριθμός ταυτότητας, ΑΦΜ, κτλ.) καθώς ένα ID αντιστοιχεί σε ένα μοναδικό πρόσωπο.

## Ισοδυναμία

Στην παρουσίαση των συναρτησιακών ιδιοτήτων έγινε αναφορά στην έννοια της ισοδυναμίας μεταξύ πόρων. Όπως έχουμε πει, η RDF παρέχει έναν καθολικό τρόπο ταυτοποίησης των πόρων με τη χρήση των URIs. Αυτό καθιστά δυνατή την αναφορά σε μία απλή οντότητα με κατανεμημένο τρόπο. Αν όμως θέλουμε να συγχωνεύσουμε πληροφορίες από πολλαπλές πηγές, μπορεί η ίδια οντότητα να αναφέρεται με διαφορετικά URIs. Στη περίπτωση αυτή θα θέλαμε να συμπεράνουμε κάπως ότι οι δύο αυτές οντότητες είναι ισοδύναμες. Η OWL δίνει αυτή τη δυνατότητα με μία πληθώρα εννοιών ισοδυναμίας.

Ισοδυναμία Κλάσεων

Όταν δύο κλάσεις είναι γνωστό ότι έχουν πάντα τα ίδια μέλη, τότε λέμε ότι είναι ισοδύναμες. Η έννοια της ισοδυναμίας είναι κάτι που θα μπορούσαμε να ορίσουμε και στην RDFS, απλά με έναν πιο έμμεσο τρόπο, χρησιμοποιώντας την ιδιότητα rdfs:subClassOf. Για παράδειγμα, οι παρακάτω δύο τριπλέτες μας δίνουν το συμπέρασμα ότι οι κλάσεις Analyst και Researcher είναι ισοδύναμες:

**:Analyst rdfs:subClassOf :Researcher .**

**:Researcher rdfs:subClassOf :Analyst .**

Πράγματι θα είναι ισοδύναμες καθώς όλα τα στιγμιότυπα της κλάσης Analyst θα γίνονται και στιγμιότυπα τους κλάσης Researcher και αντίστροφα. Η OWL δίνει έναν άμεσο τρόπο για τον ορισμό της ισοδυναμίας μεταξύ κλάσεων, με το δομικό στοιχείο **owl:equivalentClass**. Οπότε, το παραπάνω παράδειγμα με χρήση της OWL θα γραφόταν:

**:Analyst owl:equivalentClass :Researcher .**

Ισοδυναμία Ιδιοτήτων

Με παρόμοιο τρόπο λειτουργεί και η ισοδυναμία μεταξύ ιδιοτήτων. Στην RDFS μπορούσαμε να την ορίσουμε έμμεσα με αντίστοιχο τρόπο χρησιμοποιώντας την ιδιότητα rdfs:subPropertyOf. Για παράδειγμα:

**:bought rdfs:subPropertyOf :purchased**

**:purchased rdfs:subPropertyOf :bought**

Στην OWL θα μπορούσαμε να το ορίσουμε άμεσα χρησιμοποιώντας την ιδιότητα **owl:equivalentProperty**

**:bought owl:equivalentProperty :purchased**

Ομοιότητα Στιγμιότυπων

Η ισοδυναμία κλάσεων και ιδιοτήτων που αναφέρθηκαν, παρόλο που προσφέρουν πιο εύληπτους τρόπους για την έκφραση αυτών των σχέσεων, δεν δίνουν επιπλέον εκφραστικές δυνατότητες σε σχέση με την RDFS. Απλά παρέχουν έναν πιο σύντομο και ευανάγνωστο τρόπο παρουσίασης τους.

Πέρα όμως από την ισοδυναμία μεταξύ κλάσεων και ιδιοτήτων, χρειάζεται και ένας τρόπος για να εκφράσουμε την ισοδυναμία μεταξύ των ίδιων των αντικειμένων. Αυτό είναι ιδιαίτερο χρήσιμο όταν προσπαθούμε να συνδυάσουμε πληροφορίες από διαφορετικές πηγές, όπου μπορεί να γίνεται αναφορά στην ίδια οντότητα με διαφορετικά URIs. Η RDFS δεν παρέχει κάποιο δομικό στοιχείο για να το επιτύχουμε αυτό. Για αυτό η OWL μας παρέχει ένα επιπλέον δομικό στοιχείο με το οποίο μπορούμε να θέσουμε δύο στιγμιότυπα ως ισοδύναμα, το **owl:sameAs**.

**:x owl:sameAs :y**

Με τη χρήση αυτής της ιδιότητας, για κάθε τριπλέτα στην οποία υπάρχει το :x, μπορούμε να εξάγουμε την ίδια τριπλέτα, αντικαθιστώντας το :x με το :y και αντίστροφα, καθώς είναι συμμετρική ιδιότητα.

## Περιορισμοί

Οι περιορισμοί αποτελούν ένα από τα σημαντικότερα δομικά στοιχεία της OWL, καθώς μας επιτρέπουν να περιγράφουμε και να ορίσουμε νέες κλάσεις, συσχετίζοντας τες με άλλες κλάσεις που έχουμε ήδη μοντελοποιήσει. Αυτό αυξάνει σημαντικά τις δυνατότητες μοντελοποίησης που διαθέτουμε και μας επιτρέπει να δημιουργήσουμε εύκολα πιο περίπλοκες και εξειδικευμένες κλάσεις.

Είναι μία ειδική περίπτωση κλάσεων που ορίζεται από τη περιγραφή των στιγμιότυπων που θα περιέχουν. Ο ορισμός τους γίνεται με τη χρήση των λέξεων-κλειδιών **owl:Restriction** και **owl:onProperty**, όπου το τελευταίο δηλώνει την ιδιότητα της οποίας η τιμή μας ενδιαφέρει. Έπειτα, ακολουθεί ο τύπος που θέλουμε να είναι ο περιορισμός.

Υπάρχουν 3 βασικοί τύποι περιορισμών:

owl:someValuesFrom

Ο περιορισμός owl:someValuesFrom θυμίζει την έννοια του υπαρξιακού ποσοδείκτη. Παράγει περιορισμούς του τύπου:

«Όλα τα στιγμιότυπα για τα οποία τουλάχιστον μία τιμή τους ιδιότητας P προέρχεται από την κλάση C» [6]

Ο ορισμός ενός περιορισμού αυτού του τύπου στην OWL γίνεται με αυτό τον τρόπο:

**[ a owl:Restriction;**

**owl:onProperty :P;**

**owl:someValuesFrom :C** .]

Για παράδειγμα, έστω ότι θέλουμε να ορίσουμε μία νέα κλάση που να περιλαμβάνει όλους τους νικητές grand slam στο τένις, και έχουμε υλοποιήσει μία ιδιότητα hasWon με range τη κλάση Tournament. Ως υποκλάση του Tournament, υπάρχει η κλάση GrandSlam. Άρα για οποιοδήποτε στιγμιότυπο, αν έστω μία τιμή της ιδιότητας hasWon είναι τύπου GrandSlam, θέλουμε αυτό το στιγμιότυπο να ανήκει στην κλάση.

Με τη χρήση περιορισμών και της ισοδυναμίας, αυτό θα υλοποιούταν ως εξής:

**:GrandSlamWinner owl:equivalentClass**

**[ a owl:Restriction;**

**owl:onProperty :hasWon;**

**owl:someValuesFrom :GrandSlam . ]**

owl:allValuesFrom

Αντίστοιχα, ο περιορισμός owl:allValuesFrom αντιστοιχεί στην έννοια του καθολικού ποσοδείκτη.

Παράγει περιορισμούς του τύπου:

«Όλα τα στιγμιότυπα για τα οποία όλες οι τιμές της ιδιότητας P προέρχονται από την κλάση C» [6]

Ορίζεται ως εξής:

**[ a owl:Restriction;**

**owl:onProperty :P;**

**owl:allValuesFrom :C** .]

Αν ένα στιγμιότυπο είναι μέλος τους περιορισμού αυτού του τύπου, τότε μπορούμε να συμπεράνουμε, για κάθε τριπλέτα που περιγράφει το στιγμιότυπο με την ιδιότητα :P, ότι η τιμή της ιδιότητας είναι στιγμιότυπο της κλάσης :C.

owl:hasValue

Παράγει περιορισμούς του τύπου:

«Όλα τα στιγμιότυπα που έχουν την τιμή A για την ιδιότητα P» [6]

Θα μπορούσε να θεωρηθεί ως μία ειδική περίπτωση του someValuesFrom, στην οποία η κλάση C είναι ένα μοναδιαίο σύνολο {Α}. [2]

Ορίζεται ως εξής:

**[ a owl:Restriction;**

**owl:onProperty :P;**

**owl:hasValue :A** .]

# SPARQL

Η SPARQL είναι η γλώσσα ερωτημάτων (query language) που χρησιμοποιείται για την πρόσβαση σε δεδομένα RDF. Σε μία «κανονική» βάση δεδομένων θα χρησιμοποιούσαμε ένα ερώτημα SQL για να ανακτήσουμε δεδομένα. Αντίστοιχα, χρησιμοποιούμε την SPARQL για να ανακτήσουμε δεδομένα που είναι αποθηκευμένα σε μία σημασιολογική βάση δεδομένων, που αποθηκεύει τα δεδομένα ως τριπλέτες.

Οι βάσεις αυτές είναι γνωστές και ως triplestore ή RDF store και αποθηκεύουν τα δεδομένα ως τριπλέτες, με τη μορφή subject-predicate-object που έχει αναφερθεί. Κάθε triplestore παρέχει και έναν τρόπο πρόσβασης στα δεδομένα του, ώστε να μπορούν να χρήστες να θέτουν ερωτήματα, που ονομάζεται endpoint. Ένα SPARQL Endpoint είναι πάντα σε θέση να λαμβάνει και να επεξεργάζεται αιτήματα πρωτοκόλλου SPARQL. Παράδειγμα ενός endpoint φαίνεται στην εικόνα 9, όπου βλέπουμε το endpoint που δίνει η DBpedia [7]. Εκεί μπορούμε να γράψουμε και να εκτελέσουμε ερωτήματα πάνω στα δεδομένα της βάσης δεδομένων που διαθέτει η DBpedia.

Graphical user interface, text, application, email

Description automatically generated

Εικόνα 9 – SPARQL Endpoint της DBpedia[[9]](#footnote-9)

## Δομή ενός SPARQL ερωτήματος

Κάθε SPARQL ερώτημα περιλαμβάνει:

* Έναν τροποποιητή λύσης (solution modifier), ή κεφαλή
* Το σώμα του ερωτήματος

Ο τροποποιητής λύσης παρέχει τη βάση για τον καθορισμό του τύπου του ερωτήματος και του τρόπου παρουσίασης των αποτελεσμάτων. Το σώμα του ερωτήματος περιλαμβάνει μία συλλογή από μοτίβα δηλώσεων RDF, γραμμένα σε Turtle, με βάση τα οποία θα επιλεχθούν οι αντίστοιχες τριπλέτες που είναι αποθηκευμένες στη βάση.

Στη εικόνα 10 φαίνεται ένα παράδειγμα ενός πολύ απλού ερωτήματος. Η πρώτη σειρά αποτελεί τον solution modifier, όπου καθορίζουμε τον τύπο του ερωτήματος (εδώ είναι SELECT), και το πως θέλουμε να φαίνονται τα αποτελέσματα (εδώ απλά εμφανίζουμε τη μεταβλητή ?author). Το σώμα του ερωτήματος είναι η εντολή WHERE μαζί με ότι περιλαμβάνει. Στη περίπτωση αυτή απλά ζητάμε το πεδίο dbo:author του πόρου [<http://dbpedia.org/resource/The\_Lord\_of\_the\_Rings>](http://dbpedia.org/resource/The_Lord_of_the_Rings).

A picture containing graphical user interface

Description automatically generated

Εικόνα 10 – Παράδειγμα SPARQL ερωτήματος

## SELECT και WHERE

Τα ερωτήματα SELECT αποτελούν συνήθως μία απλή μορφή ερωτήματος και χρησιμοποιούνται για την επιλογή και την αναπαράσταση δεδομένων από τη βάση. Θυμίζουν αρκετά τα ερωτήματα SELECT τους SQL και χρησιμοποιούνται για τον ίδιο σκοπό.

Έχουν δύο μέρη:

1. Μια σειρά ερωτηματικών λέξεων, ή μεταβλητών.
2. Ένα πρότυπο γράφου ερώτησης.

Στην εικόνα 11 φαίνεται ένα απλό παράδειγμα SELECT ερωτήματος. Μετά το SELECT ακολουθούν δύο μεταβλητές, οι οποίες θα περιλαμβάνουν τα αποτελέσματα που θα εμφανιστούν μόλις εκτελεστεί το ερώτημα. Η εντολή WHERE υποδεικνύει τον πρότυπο γράφο ερώτησης, που είναι οτιδήποτε περιέχεται μέσα τους αγκύλες. Εκεί, μέσω RDF δηλώσεων με σύνταξη παρόμοια της Turtle, επιλέγονται από τη βάση τα objects όλων των τριπλετών που επαληθεύουν αυτές τις δηλώσεις. Για παράδειγμα, από την πρώτη δήλωση μέσα στη WHERE, αποθηκεύονται στη μεταβλητή ?author τα objects όλων των τριπλετών με subject το URI που δίνεται και predicate το dbo:author, και αντίστοιχα στη δεύτερη δήλωση με predicate το dbo:publisher. Εφόσον στη θέση του object έχουμε μία μεταβλητή, δεν μας ενδιαφέρει τι βρίσκεται ως object, οποιαδήποτε τιμή και να έχει απλά αποθηκεύεται στην αντίστοιχη μεταβλητή και μετά επιστρέφεται μέσω της SELECT.

Text

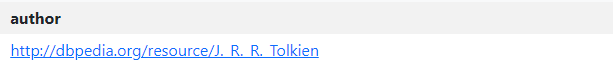
Description automatically generated

Εικόνα 11 – Παράδειγμα ερωτήματος SELECT – WHERE

Εκτελώντας λοιπόν αυτό το ερώτημα επιστρέφονται 2 στήλες, μία για κάθε μεταβλητή που υπάρχει στη SELECT.

Graphical user interface

Description automatically generated with medium confidence



Εικόνα 12 – Αποτελέσματα ερωτήματος

## CONSTRUCT

Σε αντίθεση με την SELECT που απλά επιστρέφει σε μορφή πίνακα τα αποτελέσματα που ικανοποιούν τον πρότυπο γράφο, η εντολή CONSTRUCT εντοπίζει με αντίστοιχο τρόπο τις τιμές και τις τοποθετεί σε έναν νέο πρότυπο γράφο για την κατασκευή RDF τριπλετών. Με αυτό τον τρόπο μπορούμε να αλλάξουμε την μορφή του αποτελέσματος και να προσθέσουμε επιπλέον πληροφορίες που θα θέλαμε να υπάρχουν. Χρησιμοποιείται συχνά όταν θέλουμε να ορίσουμε νέες σχέσεις μεταξύ των δεδομένων που έχουμε στην βάση ή όταν θέλουμε να δημιουργήσουμε κανόνες. Για παράδειγμα, έστω ότι στην οντολογία τους έχουμε την ιδιότητα :hasParent για να μοντελοποιήσουμε την έννοια του γονέα και θα θέλαμε να προσθέσουμε την ιδιότητα :hasGrandparent. Οπότε, αν είχαμε τις τριπλέτες

**:x :hasParent :y,** και

**:y :hasParent :z**, τώρα θα θέλουμε να έχουμε και την τριπλέτα

**:x :hasGrandparent :z.**

Αυτό μπορεί να γίνει εύκολα με τη χρήση ενός CONSTRUCT ερωτήματος. Στο WHERE κομμάτι του ερωτήματος θα εντοπίζονται όλοι οι πόροι μεταξύ των οποίων υπάρχουν οι επιθυμητές :hasParent σχέσεις, και στο CONSTRUCT τμήμα θα προστίθεται η ιδιότητα :hasGrandparent μεταξύ των αντίστοιχων πόρων. Στην εικόνα 13 βλέπουμε το ερώτημα που θα εκτελούσαμε:

Graphical user interface

Description automatically generated with low confidence

Εικόνα 13 – Παράδειγμα CONSTRUCT ερωτήματος

Στην εικόνα 14 παρουσιάζεται ένα ακόμα απλό παράδειγμα CONSTRUCT ερωτήματος. Στο κομμάτι της SELECT, επιλέγονται οι τιμές της ιδιότητας dbo:author και dbo:publisher που σχετίζονται με τον πόρο Lord of the Rings. Έπειτα, μέσα στη CONSTRUCT δημιουργείται νέος πρότυπος γράφος στον οποίο προστίθενται labels στους επιλεγμένους πόρους με τη χρήση του rdfs:label. Έτσι προσθέτουμε επιπλέον πληροφορία στα δεδομένα μας.

Graphical user interface, text, application, email

Description automatically generated

Εικόνα 14 – Παράδειγμα CONSTRUCT ερωτήματος

Τα αποτελέσματα των CONSTRUCT ερωτημάτων είναι τριπλέτες.

Text

Description automatically generated

Εικόνα 15 – Αποτελέσματα ερωτήματος

## ASK / DESCRIBE

Ένα ερώτημα τύπου ASK χρησιμοποιείται για την εκτέλεση ερωτημάτων που δίνουν απάντηση ναι ή όχι. Η απάντηση που δίνει θα είναι true ή false. Για παράδειγμα, έστω ότι θέλουμε να βρούμε αν υπάρχει κάποιο έργο που να προηγείται του Lord of the Rings? Στην εικόνα 16 φαίνεται το πως θα θέταμε αυτό το ερώτημα στην SPARQL με τη χρήση της εντολής ASK. Αν η μεταβλητή ?any έχει πάρει έστω μία τιμή, τότε η απάντηση θα είναι true, αλλιώς θα είναι false.

A picture containing diagram

Description automatically generated

Εικόνα 16 – Παράδειγμα ASK ερωτήματος

Από την άλλη, ένα ερώτημα τύπου DESCRIBE επιστρέφει πληροφορίες σχετικά με τους πόρους που ζητάμε. Εντοπίζει όλες τις τριπλέτες που παρέχουν πληροφορίες σχετικές με τους τους πόρους και τις επιστρέφει. Για παράδειγμα, το παρακάτω ερώτημα της εικόνας 17 θα επιστρέψει όλες τις πληροφορίες που θα μπορέσει να βρει σχετικά με τον author του Lord of the Rings.

A picture containing text

Description automatically generated

Εικόνα 17 – Παράδειγμα DESCRIBE ερωτήματος

## Εντολή FILTER

Μας επιτρέπει να φιλτράρουμε τα αποτελέσματα του ερωτήματος μας. Με τη FILTER μπορούμε να ορίσουμε έναν Boolean έλεγχο (ή περισσότερους), μέσω του οποίου θα προσδιορίζεται αν η αντίστοιχη γραμμή θα συμπεριληφθεί ή όχι στα αποτελέσματα. Αυτό μας επιτρέπει να θέτουμε πιο λεπτομερή ερωτήματα καθώς και να απορρίπτουμε εύκολα τους γραμμές που δεν μας ενδιαφέρουν. Για παράδειγμα, έστω ότι από τα σχόλια του Lord of the Rings, θέλουμε μόνο αυτά που είναι στα αγγλικά:

A picture containing text

Description automatically generated

Εικόνα 18 – Παράδειγμα χρήσης FILTER

## Εντολή LIMIT

Χρησιμοποιείται όταν θέλουμε να εμφανίζεται ένας συγκεκριμένος αριθμός γραμμών στα αποτελέσματα. Για παράδειγμα, αν ένα ερώτημα επιστρέφει πολλά αποτελέσματα και εμείς θέλαμε μόνο τα πρώτα 3, θα χρησιμοποιούσαμε την εντολή LIMIT 3.

A picture containing graphical user interface

Description automatically generated

Εικόνα 19 – Παράδειγμα χρήσης LIMIT

A screenshot of a computer

Description automatically generated

## Εντολή ORDER BY

Χρησιμοποιείται όταν θέλουμε να ταξινομήσουμε τα αποτελέσματα, με βάση τις τιμές κάποιας μεταβλητής. Μπορεί να συνδυαστεί με την εντολή LIMIT ώστε να μπορούμε να παίρνουμε τις μεγαλύτερες ή μικρότερες τιμές του συνόλου των αποτελεσμάτων.

Text

Description automatically generated

Εικόνα 20 – Παράδειγμα χρήσης ORDER BY

Στο παράδειγμα αυτό, τα αποτελέσματα ταξινομούνται με αύξουσα αλφαβητική σειρά βάση της μεταβλητής ?subject, και έπειτα κρατούνται μόνο τα πρώτα 3. Αν μετά τις μεταβλητές προστεθεί η λέξη-κλειδί **DESC**, τότε η ταξινόμηση θα είναι φθίνουσα.

## Μερικά παραδείγματα

Έστω ότι ψάχνουμε να βρούμε τα ονόματα και τον τόπο γέννησης όλων των τενιστών που γεννήθηκαν μετά το 1990 και έχουν κερδίσει το Wimbledon. Θα χρειαστούμε τις ιδιότητες rdf:type, για να ελέγξουμε αν ο τύπος είναι TennisPlayer, και έπειτα τις ιδιότητες dbp:name, dbp:birthPlace, dbp:birthDate και dbp:wimbledonresult. Επίσης, θα χρειαστεί να φιλτράρουμε τις μεταβλητές birthDate και wimbledonresult ώστε να ικανοποιούνται οι συνθήκες του ερωτήματος.

Text, letter

Description automatically generated

Προσθέτοντας στο τέλος την εντολή LIMIT 5 για να πάρουμε τα 5 πρώτα αποτελέσματα και εκτελώντας το ερώτημα εμφανίζονται τα ακόλουθα:

Background pattern

Description automatically generated

Έστω ότι ψάχνουμε να βρούμε τα ονόματα και τον πληθυσμό των πέντε χωρών που έχουν ως νόμισμα το ευρώ και έχουν το μικρότερο human development index. Θα χρειαστούμε και πάλι το rdf:type για να προσδιορίσουμε αρχικά ότι ο πόρος είναι χώρα και έπειτα τις ιδιότητες dbp:commonName, dbo:currencyCode, dbp:populationEstimate και dbo:humanDevelopmentIdex ώστε να πάρουμε τα στοιχεία που θέλουμε. Για να πάρουμε τις πέντε χώρες με το μικρότερο HDI, αρχικά ταξινομούμε τα αποτελέσματα σε αύξουσα σειρά με βάση το HDI με την εντολή ORDER BY και έπειτα παίρνουμε μόνο τα πρώτα 5 με την εντολή LIMIT.

Text, letter

Description automatically generated

Τα αποτελέσματα:

Graphical user interface

Description automatically generated with medium confidence

# Σημασιολογικές Βάσεις Δεδομένων

Τα δεδομένα του ΣΙ έχουν διαφορετική μορφή από τα δεδομένα του κανονικού Ιστού. Για τον λόγο αυτό χρειάζονται βάσεις που να ειδικά διαμορφωμένες ώστε να μπορούν να αποθηκεύσουν διασυνδεδεμένα δεδομένα σε μορφή τριπλετών.

## Graph Databases

Μια βάση δεδομένων γραφήματος είναι ένας τύπος βάσεων δεδομένων που χρησιμοποιεί δομές γραφημάτων για σημασιολογικά ερωτήματα, όπως κόμβους, ακμές και ιδιότητες για την αποθήκευση και την αναπαράσταση δεδομένων. Μία βασική έννοια αυτού του συστήματος είναι η έννοια της σχέσης (relationship) μεταξύ των διάφορων οντοτήτων της βάσης. Οι βάσεις δεδομένων γραφημάτων έχουν κατασκευαστεί ειδικά για την αποθήκευση και την πλοήγηση μέσω σχέσεων. Χρησιμοποιούν κόμβους για να αποθηκεύουν οντότητες δεδομένων, και ακμές για την αποθήκευση των σχέσεων μεταξύ των οντοτήτων. Μια ακμή έχει πάντα έναν κόμβο έναρξης, έναν κόμβο τέλους, έναν τύπο και μία κατεύθυνση και μπορεί να περιγράφει οποιονδήποτε τύπο σχέσης όπως σχέση γονέα-παιδιού, σχέσεις ιδιοκτησίας, ιεραρχίας και πολλά άλλα. Δεν υπάρχει όριο στον αριθμό και το είδος των σχέσεων στις οποίες μπορεί να περιέχεται ένας κόμβος. Το γεγονός αυτό προσφέρει μεγάλη ευελιξία και εκφραστικότητα. Ανήκουν στην ευρύτερη κατηγορία των NoSQL βάσεων [8].

Λίγο πιο αναλυτικά για τα στοιχεία των γράφων:

* Οι κόμβοι (nodes) αναπαριστούν οντότητες ή στιγμιότυπα όπως ανθρώπους, επιχειρήσεις, λογαριασμούς, ή οποιοδήποτε άλλο αντικείμενο το οποίο μπορεί να τεθεί υπό παρατήρηση. Αποτελούν την αντίστοιχη έννοια μίας εγγραφής ή μίας γραμμής που θα συναντούσαμε σε μία σχεσιακή βάση δεδομένων.
* Οι ακμές (edges), για τις οποίες χρησιμοποιούνται και οι όροι γράφος (graph) και σχέση (relationship), είναι οι γραμμές που συνδέουν τους κόμβους μεταξύ τους και αναπαριστούν τις σχέσεις που υπάρχουν ανάμεσα τους. Η εξέταση των συνδέσεων και των διασυνδέσεων μεταξύ των στοιχείων ενός γραφήματος μας δίνει τα συμπεράσματα με τα οποία μπορούμε να απαντήσουμε τα ερωτήματα μας, και να τα χρησιμοποιήσουμε έπειτα για να βελτιώσουμε πιθανόν το προϊόν ή την υπηρεσία μας. Οι ακμές μπορούν να είναι κατευθυνόμενες ή μη κατευθυνόμενες. Σε έναν μη κατευθυνόμενο γράφο, μία ακμή που συνδέει δύο κόμβους έχει μία μοναδική σημασία. Σε έναν κατευθυνόμενο γράφο, οι ακμές μπορούν να έχουν διαφορετικές σημασίες, ανάλογα με τη κατεύθυνση τους. Οι ακμές αποτελούν την πιο βασική έννοια των graph databases, καθώς προσφέρουν δυνατότητες που δεν εφαρμόζονται άμεσα σε ένα σχεσιακό μοντέλο (relational model) [9] ή σε ένα μοντέλο αποθήκευσης εγγράφων (document-store model) [10].
* Οι ιδιότητες (properties) είναι πληροφορίες που σχετίζονται με τους κόμβους. Για παράδειγμα, αν το Lord of the Rings ήταν ένας κόμβος, θα μπορούσε να ήταν συσχετισμένο με ιδιότητες όπως βιβλίο, ταινία, τριλογία, μεταξύ άλλων.

### Labeled Property Graph (LPG)

Στις βάσεις δεδομένων τύπου LPG το γράφημα αποτελείται από κόμβους και σχέσεις (ακμές). Κάθε κόμβος ή σχέση έχει ένα ID, μία ή περισσότερες ετικέτες (labels) που περιγράφουν τον τύπο ή την κλάση του, ένα σύνολο από ιδιότητες που περιέχουν τιμές και αναπαρίστανται ως key-value, καθώς και ένα αντίστοιχο κλειδί για να επιτρέπεται η αναφορά στον αντίστοιχο κόμβο ή σχέση. Οι σχέσεις σε ένα LPG έχουν πάντα έναν αρχικό κόμβο και έναν τελικό κόμβο, καθώς και κατεύθυνση, καθιστώντας τον γράφο κατευθυνόμενο. Δύο κόμβοι λοιπόν συνδέονται πάντα με μία σχέση, σχηματίζοντας έτσι την τελική δομή του γραφήματος.

Στην εικόνα 21 φαίνεται ένα παράδειγμα ενός LPG. Όπως φαίνεται, κάθε κόμβος και ακμή έχει ένα ID καθώς και ένα label που δίνει μία περιγραφή του τύπου του. Οι κόμβοι περιγράφουν έναν άνθρωπο, οπότε το label μας δίνει την πληροφορία Person. Επίσης, κάθε κόμβος και ακμή έχουν ένα σύνολο από ιδιότητες που αναπαρίστανται ως ζεύγη key-value, και μας δίνουν περισσότερες πληροφορίες για την οντότητα που αναπαρίσταται.

Graphical user interface, application

Description automatically generated

Εικόνα 21 – Παράδειγμα LPG

### Περιπτώσεις Χρήσης

Εντοπισμός Απατών (Fraud Detection)

Οι GDB είναι ικανές για πολύ αποτελεσματικό εντοπισμό και πρόληψη απατών. Μέσω της χρήσης των σχέσεων, η επεξεργασία των οικονομικών συναλλαγών και των συναλλαγών αγοράς μπορεί να γίνεται σε σχεδόν πραγματικό χρόνο. Τα ερωτήματα εκτελούνται γρήγορα πάνω στα γραφήματα, δίνοντας μας την δυνατότητα να εντοπίσουμε γρήγορα περιπτώσεις πιθανής απάτης. Για παράδειγμα, όταν ένας αγοραστής χρησιμοποιεί το ίδιο email και την ίδια πιστωτική κάρτα που περιλαμβάνονται σε μία γνωστή υπόθεση απάτης. Επίσης, οι GDB επιτρέπουν την εύκολη αναγνώριση μοτίβων μέσα στο σύνολο των σχέσεων, όπως πολλά άτομα που σχετίζονται με μία προσωπική διεύθυνση email ή πολλά άτομα που μοιράζονται την ίδια IP αλλά κατοικούν σε διαφορετικές φυσικές διευθύνσεις.

Μηχανές Συστάσεων (Recommendation Engines)

Οι GDB είναι μία καλή επιλογή για εφαρμογές συστάσεων. Σε ένα γράφημα, μπορούν να αποθηκευτούν σχέσεις μεταξύ διαφορετικών κατηγοριών πληροφοριών όπως τα ενδιαφέροντα πελατών, οι φίλοι ή το ιστορικό αγορών. Έτσι μπορούν για παράδειγμα να γίνονται προτάσεις προϊόντων σε έναν χρήστη με βάση τα προϊόντα που αγοράζονται από τους χρήστες οι οποίο έχουν παρόμοιο ιστορικό. Επίσης, μπορούν να εντοπιστούν άτομα που έχουν έναν κοινό φίλο αλλά δεν γνωρίζονται μεταξύ τους και να γίνει έτσι μία σύσταση φιλίας. Οι στρατηγικές αυτές χρησιμοποιούνται από όλες τις μεγάλες εταιρείες που χρησιμοποιούν μηχανές συστάσεων, όπως social media websites (facebook, Instagram, twitter), entertainment websites όπως Netflix, YouTube, και πολλά άλλα.

## RDF Triplestore

Τα RDF triplestores αποτελούν μία υποκατηγορία των graph databases. Είναι ένας εξειδικευμένος τύπος βάσης δεδομένων που έχει σχεδιαστεί ειδικά για την αποθήκευση, ανάκτηση και διαχείριση RDF τριπλετών. Παρέχουν έναν μηχανισμό αποθήκευσης και οργάνωσης των δεδομένων με τρόπο που επιτρέπει την αποτελεσματική αναζήτηση και ανάκτηση πληροφοριών. Όπως έχουμε αναφέρει, το RDF είναι ένα μοντέλο δεδομένων που χρησιμοποιείται για την περιγραφή πόρων στον Ιστό χρησιμοποιώντας δηλώσεις της μορφής υποκείμενο-κατηγόρημα-αντικείμενο, γνωστές και ως RDF triples. Κάθε τριπλέτα αντιπροσωπεύει μία σχέση μεταξύ δύο πόρων, με το κατηγόρημα να περιγράφει το είδος της σχέσης. Μπορούν να χρησιμοποιηθούν για μία ποικιλία εφαρμογών, όπως για την κατασκευή γραφημάτων γνώσης (knowledge graphs), για σημασιολογική αναζήτηση, και για την ανάπτυξη εφαρμογών διασυνδεδεμένων δεδομένων.

Τα triplestores υποστηρίζουν την εκτέλεση σημασιολογικών ερωτημάτων στα δεδομένα τους με τη χρήση της SPARQL. Τα ερωτήματα SPARQL μπορούν να χρησιμοποιηθούν για την αναζήτηση μοτίβων στα δεδομένα RDF, για φιλτράρισμα και συγκέντρωση των δεδομένων, και για την εκτέλεση σύνθετων ερωτημάτων σε πολλαπλά σύνολα δεδομένων.

### Περιπτώσεις Χρήσης

Μερικές από τις πιο συχνές περιπτώσεις χρήσης των RDF triplestores είναι οι ακόλουθες:

* **Εφαρμογές Σημασιολογικού Ιστού**: Τα triplestores χρησιμοποιούνται συχνά στην ανάπτυξη εφαρμογών ΣΙ που χρησιμοποιούν δεδομένα για να περιγράψουν και να συσχετίσουν διαφορετικές έννοιες στον Ιστό. Αποθηκεύοντας τα RDF δεδομένα σε ένα triplestore, οι προγραμματιστές μπορούν να υποβάλλουν ερωτήματα και να χειριστούν τα δεδομένα πιο αποδοτικά, και να δημιουργήσουν εφαρμογές που παρέχουν πιο έξυπνη λειτουργία αναζήτησης και ανακάλυψης.
* **Διαχείριση Γνώσης**: Τα triplestores είναι επίσης χρήσιμα για την διαχείριση της γνώσης ενός οργανισμού. Αποθηκεύοντας δεδομένα σχετικά με προϊόντα, υπηρεσίες, υπαλλήλους και άλλους πόρους σε ένα triplestore, οι οργανισμοί μπορούν να δημιουργήσουν πιο αποτελεσματικά συστήματα αναζήτησης και προτάσεων, καθώς και καλύτερα εργαλεία συνεργασίας.
* **Internet Of Things (IoT)**: Καθώς ο αριθμός των συνδεδεμένων συσκευών συνεχίζει να αυξάνεται, υπάρχει μία αυξανόμενη ανάγκη διαχείρισης και αναζήτησης δεδομένων από πολλαπλές πηγές. Τα triplestores μπορούν να χρησιμοποιηθούν για την αποθήκευση και τη διαχείριση δεδομένων του IoT, επιτρέποντας στους προγραμματιστές να δημιουργούν εφαρμογές που παρέχουν πληροφορίες σχετικά με την απόδοση συσκευών, μοτίβα χρήσης, και άλλες σημαντικές μετρικές. [11]
* **Ενσωμάτωση Δεδομένων (Data Integration)**: Τα triplestores είναι ιδιαίτερα χρήσιμα για την ενοποίηση δεδομένων από διαφορετικές πηγές. Αποθηκεύοντας τα δεδομένα σε μία κοινή μορφή, όπως η RDF, οι προγραμματιστές μπορούν εύκολα να συνδυάσουν και να συγκρίνουν δεδομένα από διαφορετικές πηγές, δημιουργώντας έτσι πιο ισχυρές εφαρμογές ανάλυσης και επιχειρηματικής ευφυΐας.

1. ΣΥΣΚΕΥΕΣ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ

ΣΥΣΚΕΥΕΣ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ

# Εισαγωγή

Οι συσκευές παρακολούθησης δραστηριότητας, γνωστές και ως fitness trackers, είναι φορητές ηλεκτρονικές συσκευές που σχεδιάστηκαν για να παρακολουθούν και να μετρούν διάφορες πτυχές της σωματικής δραστηριότητας και της υγείας. Αυτές οι συσκευές φοριούνται συνήθως στον καρπό, αλλά μπορούν επίσης να φορεθούν ως κολιέ, καρφίτσες ή ενσωματωμένες, όπως οι έξυπνοι αισθητήρες ρούχων. Kάνουν χρήση διάφορων αισθητήρων για να συλλέξουν δεδομένα σχετικά με τη δραστηριότητα του χρήστη και την υγεία του. Επιπλέον, υποστηρίζουν τη συμβατότητα με εφαρμογές κινητών τηλεφώνων ή υπολογιστών για τη συγχρονισμό και ανάλυση των δεδομένων, την αδιάβροχη σχεδίαση για χρήση κατά τη διάρκεια άσκησης ή προπόνησης στο νερό, την διάρκεια ζωής της μπαταρίας για επαρκή λειτουργία, και την ακρίβεια και αξιοπιστία των μετρήσεων που πραγματοποιούν.

Τα είδη των δεδομένων που συλλέγονται μπορεί να περιλαμβάνουν:

* Βήματα και απόσταση: Οι συσκευές καταγράφουν τον αριθμό των βημάτων που κάνει κάποιος και την απόσταση που διανύει.
* Καρδιακή συχνότητα: Ορισμένα fitness trackers μπορούν να μετρούν την καρδιακή συχνότητα κατά τη διάρκεια της ημέρας και κατά τη διάρκεια της άσκησης.
* Κατανάλωση θερμίδων: Με βάση τη δραστηριότητα και τα βιομετρικά δεδομένα, οι συσκευές εκτιμούν την κατανάλωση θερμίδων του χρήστη.
* Ύπνος: Ορισμένες συσκευές παρακολουθούν την ποιότητα και τη διάρκεια του ύπνου, καταγράφοντας τον αριθμό των ωρών ύπνου και τα επίπεδα αναζωογόνησης.
* Επίπεδα οξυγόνου στο αίμα: Ορισμένα προηγμένα fitness trackers μετρούν τα επίπεδα οξυγόνου στο αίμα για να παρέχουν πληροφορίες σχετικά με την αντοχή και την καρδιοαναπνευστική επίδοση.

Τα fitness trackers έχουν γίνει δημοφιλή για την παρακολούθηση και τη βελτίωση της φυσικής κατάστασης, της υγείας και του ευζωικού τρόπου ζωής. Παρέχουν πληροφορίες και κίνητρα για την επίτευξη γυμναστικών στόχων, την παρακολούθηση της προόδου, τη βελτίωση της υγείας, τη διατήρηση του υγιούς βιονομικού ρυθμού, και την ενσωμάτωση της φυσικής δραστηριότητας στην καθημερινή ζωή.

### Βασικές κατηγορίες fitness trackers

Υπάρχουν διάφορα είδη fitness trackers που καλύπτουν διάφορες ανάγκες και προτιμήσεις:

* Βηματομετρητές (Step Trackers): Αυτοί οι fitness trackers μετρούν τον αριθμό των βημάτων του χρήστη κατά τη διάρκεια τους ημέρας. Χρησιμοποιούν ενσωματωμένους αισθητήρες επιτάχυνσης για να ανιχνεύσουν της κινήσεις του σώματος των χρηστών και να υπολογίσουν τον αριθμό των βημάτων τους.
* Καρδιοπαλμομετρητές (Heart Rate Trackers): Αυτοί οι fitness trackers μετρούν την καρδιακή συχνότητα. Χρησιμοποιούν αισθητήρες καρδιακών παλμών, είτε μέσω της επαφής με το δέρμα του χρήστη είτε μέσω οπτικής ανίχνευσης, για να παρέχουν πληροφορίες σχετικά με την καρδιακή δραστηριότητα, την κατανάλωση θερμίδων και την αντοχή.
* Παρακολούθηση ύπνου (Sleep Trackers): Αυτοί οι fitness trackers παρακολουθούν την ποιότητα και τη διάρκεια του ύπνου. Χρησιμοποιούν αισθητήρες για να ανιχνεύσουν την κίνηση, τη συχνότητα καρδιακών παλμών και τους παραμέτρους για να αξιολογήσουν την ποιότητα του ύπνου των χρηστών.
* Γυμναστήριο/Προπόνηση (Gym/Workout Trackers): Αυτοί οι fitness trackers είναι ειδικά σχεδιασμένοι για την παρακολούθηση και την καταγραφή της προπόνησης και της γυμναστικής των χρηστών. Μπορούν να ανιχνεύσουν ασκήσεις, να μετρήσουν την κατανάλωση θερμίδων, να παρακολουθούν τον καρδιακό ρυθμό και να παρέχουν συμβουλές για τη βελτίωση της απόδοσης.
* Πολυλειτουργικά (Multisport Trackers): Αυτοί οι fitness trackers είναι προηγμένοι και προορίζονται για πολλαπλές δραστηριότητες και αθλήματα. Μπορούν να παρακολουθήσουν τη δραστηριότητα σε διάφορες αθλητικές δραστηριότητες, όπως τρέξιμο, κολύμβηση, ποδηλασία, γυμναστική και άλλα.

Αυτά είναι μερικά από τα κύρια είδη fitness trackers που βρίσκονται στην αγορά. Καθένα από αυτά έχει τους δικές του μοναδικές λειτουργίες και δυνατότητες, επιτρέποντάς τους χρήστες να προσαρμόσουν την επιλογή τους βάση των ατομικών τους αναγκών και στόχων φυσικής κατάστασης. Βέβαια, καθώς ο τομέας των fitness tracker έχει γνωρίσει μεγάλη ανάπτυξη τα τελευταία χρόνια, οι καινούριες συσκευές που κυκλοφορούν υποστηρίζουν συνήθως όλες τις λειτουργίες που αναφέρθηκαν και παραπάνω, υποστηρίζοντας μέχρι και λειτουργίες ενός smartphone.

## Smartwatches

Ένα smartwatch είναι με λίγα λόγια ένας φορητός υπολογιστής σε μορφή ρολογιού. Αποτελούν το επικρατέστερο είδος συσκευών παρακολούθησης δραστηριότητας. Η δημοφιλία τους τα τελευταία χρόνια έχει φτάσει σε τέτοιο βαθμό ώστε οι όροι fitness tracker και smartwatch να θεωρούνται ταυτόσημοι.

Τα σύγχρονα smartwatches παρέχουν μία απλή διεπαφή οθόνης αφής όπου ο χρήστης θα μπορεί να βλέπει όλα τα δεδομένα που έχουν αναφερθεί καθώς και να εκτελεί διάφορες καθημερινές λειτουργίες. Σχεδόν για όλα τα smartwatches πλέον, υπάρχει η δυνατότητα σύνδεσης με κάποια εφαρμογή, ανάλογα με το brand του ρολογιού, η οποία παρέχει επιπλέον δυνατότητες καθώς και μακροπρόθεσμη βιοπαρακολούθηση του χρήστη και καταγραφή των μετρικών που τον ενδιαφέρουν. Ενώ τα πρώιμα μοντέλα μπορούσαν να εκτελέσουν βασικές λειτουργίες, όπως υπολογισμούς, ένδειξη ώρας, μεταφράσεις και απλά παιχνίδια, τα νεότερα μοντέλα μοιάζουν όλο και περισσότερο στα smartphones καθώς περιέχουν εφαρμογές, λειτουργικό σύστημα και δυνατότητα σύνδεσης με Wi-Fi και Bluetooth. Ορισμένα μοντέλα μπορούν να λειτουργήσουν και ως φορητές συσκευές για την αναπαραγωγή πολυμέσων όπως αρχείων ήχου και βίντεο μέσω ακουστικών Bluetooth. Επίσης, πολλά σύγχρονα μοντέλα διαθέτουν και λειτουργίες κινητής τηλεφωνίας, όπως πραγματοποίηση κλήσεων και αποστολή μηνυμάτων.

Στην εικόνα 22 φαίνεται το Apple Watch Series 8, το οποίο είναι ένα από τα καλύτερα smartwatches της αγοράς. Μπορεί να εκτελέσει όλες τις βασικές λειτουργίες που αναφέρθηκαν με πολύ αποτελεσματικό τρόπο, και παρέχει επιπλέον δυνατότητες όπως παρακολούθηση της θερμοκρασίας σώματος του χρήστη και ανίχνευση συγκρούσεων. Όσο η δημοφιλία των smartwatch παραμένει τόσο υψηλή, έτσι θα συνεχίσουν και να κυκλοφορούν ολοένα και νεότερες εκδόσεις οι οποίες θα διευρύνουν ακόμα περισσότερο τον ήδη αυξημένο αριθμό εργασιών που θα μπορούν να εκτελούν.



Εικόνα 22 – Apple Watch Series 8

# SSN – SOSA: Βασικές Δυνατότητες και Δομές

Η οντολογία SSN (Semantic Sensor Network) είναι μια οντολογία που χρησιμοποιείται για την περιγραφή και την αναπαράσταση αισθητήρων, παρατηρητών και συναφών δεδομένων στα πλαίσια του ΣΙ και του Internet Of Things (IoT). Η οντολογία SSN παρέχει ένα σύνολο από έννοιες και σχέσεις που επιτρέπουν την περιγραφή των αισθητήρων, των μετρήσεων, των παρατηρήσεων και των συμβάντων που σχετίζονται με αυτούς. Επιτρέπει την περιγραφή της τοποθεσίας των αισθητήρων, των μεγεθών που μετρούν, των μονάδων μέτρησης, της χρονικής σήμανσης και άλλων παραμέτρων που συνδέονται με τη συλλογή αισθητηριακών δεδομένων. Η οντολογία SSN είναι χρήσιμη για την ανάπτυξη εφαρμογών ΙοΤ, όπου η περιγραφή και η αλληλεπίδραση με αισθητήρες και παρατηρητές είναι σημαντική. Η οντολογία SOSA (Sensor, Observation, Sample, and Actuator) επεκτείνει την οντολογία SSN και παρέχει έννοιες για την περιγραφή των δειγμάτων και των ενεργοποιητών (actuators). Πέρα από τους αισθητήρες και τις παρατηρήσεις, η οντολογία SOSA περιγράφει τις διαδικασίες λήψης δειγμάτων από τους αισθητήρες, καθώς και τις ενέργειες που μπορούν να εκτελεστούν από τους ενεργοποιητές. Επιτρέπει επίσης την περιγραφή των παραμέτρων των δειγμάτων, των μεθόδων λήψης, των ενεργειών και άλλων σχετικών πληροφοριών.

Οι οντολογίες SSN και SOSA χρησιμοποιούνται κυρίως στο πλαίσιο του ΙοΤ, των έξυπνων πόλεων και των εφαρμογών που αφορούν την παρακολούθηση περιβαλλοντικών συνθηκών, την αισθητηριακή ανάλυση και την αυτοματοποίηση διαδικασιών. Βοηθούν στη δημιουργία ενός κοινού πλαισίου για την ανταλλαγή και την επεξεργασία δεδομένων αισθητήρων και παρατηρητών, προωθώντας την αλληλεπίδραση και την ανάπτυξη νέων εφαρμογών στον τομέα του ΙοΤ.

## Τμηματοποίηση

Η τμηματοποίηση οντολογίας είναι μια κοινή μέθοδος που χρησιμοποιείται στη μηχανική οντολογιών. Με την τεχνική αυτή, η οντολογία χωρίζεται σε μικρότερα τμήματα, καθένα από τα οποία εξειδικεύεται σε ένα μικρό αριθμό θεμάτων. Στοχεύει στο να παρέχει τους χρήστες οντολογιών τη γνώση που απαιτούν, μειώνοντας όσο το δυνατόν περισσότερο το εύρος σε αυτό που είναι απολύτως απαραίτητο για μια δεδομένη περίπτωση χρήσης. Η οντολογία SSN προσφέρει διάφορα υποσύνολα οντολογιών τα οποία εξυπηρετούν διαφορετικούς σκοπούς και διαφορετικές ομάδες χρηστών. Για παράδειγμα, η οντολογία SOSA έχει ως στόχο να προσφέρει δυνατότητες σημασιολογικού εμπλουτισμού σε αποθετήρια δεδομένων των οποίων η διαχείριση δεν γίνεται από τους μηχανικούς της οντολογίας αλλά από ένα ευρύτερο κοινό. Μπορούν να διακριθούν δύο κύριες κατηγορίες σπονδυλοποίησης οντολογίας.

Η πρώτη κατηγορία περιλαμβάνει εκείνες τις προσεγγίσεις που εστιάζουν στον συνδυασμό των ήδη υπαρχόντων οντολογιών μέσω ενσωμάτωσης και χαρτογράφησης οντολογιών, συνήθως μέσω δηλώσεων owl:import. Η εισαγωγή στην OWL έχει κατεύθυνση από μια εξαρτημένη οντολογία σε μια οντολογία εξάρτησης. Στην ουσία εισάγεται όλη η γνώση της οντολογίας μέσα σε μία άλλη. Αν και η εισαγωγή είναι μεταβατική, η γνώση διαδίδεται μόνο προς μία κατεύθυνση. Η οντολογία στην οποία γίνεται η εισαγωγή λαμβάνει όλη τη σημασία των εισαγόμενων όρων που χρησιμοποιούνται, συμπεριλαμβάνοντας όλα τα αξιώματα που σχετίζονται με την έννοια αυτών των όρων. Αντίθετα, η εισαγόμενη οντολογία δεν αποκτά καμία επιπλέον γνώση.

Η δεύτερη κατηγορία περιλαμβάνει τις προσεγγίσεις που στοχεύουν στο διαχωρισμό και την εξαγωγή τμημάτων οντολογιών ως ενότητες (modules). Το κύριο χαρακτηριστικό ενός module είναι ότι είναι αυτοτελές, δηλαδή αποτυπώνει πλήρως την σημασία των εισαγόμενων όρων που χρησιμοποιούνται συμπεριλαμβάνοντας όλα τα αξιώματα που σχετίζονται με την έννοια αυτών των όρων. Αυτό σημαίνει ότι το αποτέλεσμα ορισμένων συλλογιστικών εργασιών, όπως η υπαγωγή ή η απάντηση ερωτήματος εντός ενός module θα πρέπει να είναι εφικτό και να καταλήγει στις ίδιες απαντήσεις χωρίς την ανάγκη πρόσβασης σε άλλα modules.

Η τμηματοποίηση των οντολογιών SSN και SOSA χρησιμοποιεί την πρώτη προσέγγιση συνθέτοντας την οντολογία σε υπο-ενότητες που χρησιμοποιούν δηλώσεις owl:import. Διακρίνονται δύο μέθοδοι ανάλογα με την κατευθυντικότητα τους κατάτμησης: μια κάθετη κατάτμηση (vertical segmentation) και μια οριζόντια κατάτμηση (horizontal segmentation).

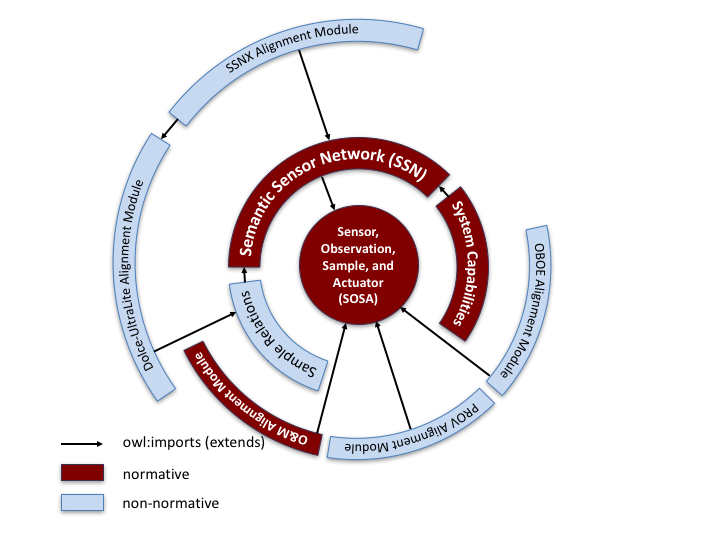
### Vertical Segmentation

Σε αυτή τη μέθοδο κατάτμησης τα κάθετα modules βασίζονται το ένα πάνω στο άλλο χρησιμοποιώντας δηλώσεις owl:import. Τα modules χαμηλότερου επιπέδου είναι ανεξάρτητα από εκείνα ανώτερων επιπέδων και λογικά συνεπείς από μόνα τους. Από την άλλη, τα modules των ανώτερων επιπέδων εξαρτώνται από modules κατώτερων επιπέδων, τα οποία εισάγουν με owl:import, και δεν μπορούν να σταθούν μόνα τους σημασιολογικά.

Για παράδειγμα, το Dolce-UltraLite Alignment module εισάγει την οντολογία SSN η οποία αντίστοιχα εισάγει την οντολογία SOSA. Ωστόσο, αντίστροφα, ούτε η SOSA ούτε η SSN εισάγουν αυτό το module. Στην πραγματικότητα, το SOSA ως πυρήνας, δεν εισάγει τις οντολογίες, γεγονός που το καθιστά πραγματικά ανεξάρτητο από τα υπόλοιπα modules που προσθέτουν περισσότερη εκφραστικότητα και περαιτέρω οντολογικές δεσμεύσεις στην ελαφριά σημασιολογία του SOSA.

### Horizontal Segmentation

Τα modules που βρίσκονται στο ίδιο οριζόντιο επίπεδο ενδέχεται να εξαρτώνται το ένα από το άλλο, δηλαδή μπορεί να βασίζονται στην εισαγωγή ενός άλλου module σε αυτά, το οποίο βρίσκεται στο ίδιο επίπεδο. Στην εικόνα 23 φαίνεται η τμηματοποίηση των οντολογιών SSN και SOSA. Βλέπουμε ότι χωρίζεται σε επίπεδα τα οποία εμπλουτίζουν αυτά που είναι ανώτερων επιπέδων, τους περιγράφηκε στο vertical segmentation, και τα modules του κάθε επιπέδου εξαρτώνται το ένα από το άλλο.



Εικόνα 23 – Οι οντολογίες SSN και SOSA με τους τα κάθετα και οριζόντια modules τους

## Συντακτικό των SSN/SOSA

### Παρατηρήσεις (Observations)

Στην εικόνα 24 φαίνονται οι βασικές κλάσεις και ιδιότητες των οντολογιών SSN και SOSA που σχετίζονται με τα Observations. Τα αξιώματα της SOSA φαίνονται με πράσινο χρώμα ενώ αυτά που ανήκουν αποκλειστικά στην SSN με μπλε.

A picture containing chart

Description automatically generated

Εικόνα 24 – Κλάσεις και σχέσεις των SSN/SOSA σχετικά με τα Observations

Πιο αναλυτικά, κάποιες από τις έννοιες που φαίνονται στην εικόνα:

sosa:ObservableProperty

Μια παρατηρήσιμη ιδιότητα ενός χαρακτηριστικού ενδιαφέροντος (feature of interest, FoT). Για παράδειγμα, το ύψος ενός κτιρίου. Στην περίπτωση αυτή, η παρατηρήσιμη ιδιότητα είναι το ύψος και το feature of interest είναι το κτίριο. Αποτελεί υποκλάση της γενικότερης κλάσης **ssn:Property.**

sosa:Observation

Παρατήρηση (Observation) καλούμε την διαδικασία (Procedure) προσέγγισης ή υπολογισμού της τιμής μίας ιδιότητας ενός feature of interest. Δηλαδή ο υπολογισμός της τιμής ενός **sosa:ObservableProperty**. Για παράδειγμα, ο υπολογισμός του ύψους ενός κτιρίου το οποίο αποτελεί FoT είναι ένα Observation.

Όπως φαίνεται στην εικόνα 22, συνδέεται με κάποιον αισθητήρα (Sensor) για να φαίνεται πως εκτελέστηκε το Observation και από τι, με ένα ObservableProperty για να περιγράψει τι είναι η εκτίμηση του αποτελέσματος, και με ένα FoT για να περιγράψει το χαρακτηριστικό με το οποίο σχετίζεται η ιδιότητα που μας ενδιαφέρει.

* **sosa:observedProperty**: σχέση που συνδέει ένα Observation με την ιδιότητα που παρατηρήθηκε. Επομένως, το domain της ιδιότητας περιλαμβάνει την κλάση sosa:Observation και το range την κλάση sosa:ObservableProperty.

sosa:Sensor

Συσκευή, πράκτορας (agent, ενδεχομένως και άνθρωπος), ή λογισμικό που υλοποιεί ή έχει κάποιο ρόλο στην υλοποίηση ενός Procedure. Ένας Sensor ανταποκρίνεται σε ένα ερέθισμα (Stimulus), δηλαδή μία αλλαγή στο περιβάλλον, ή σε δεδομένα εισόδου που προέρχονται από τα αποτελέσματα προηγούμενων Observation. Αποτελεί υποκλάση της **ssn:System**.

Ιδιότητες που σχετίζονται με αυτή τη κλάση:

* **sosa:observes**: σχέση μεταξύ ενός Sensor και ενός ObservableProperty που είναι ικανό να ανιχνεύσει. Το domain της ιδιότητας περιέχει την κλάση sosa:Sensor και το range την κλάση sosa:ObservableProperty. Επίσης, αποτελεί υπο-ιδιότητα της ssn:forProperty.
* **sosa:isObservedBy**: αντίστροφη ιδιότητα της sosa:observes.

ssn:Stimulus

Ένα γεγονός του πραγματικού κόσμου που πυροδοτεί έναν Sensor. Οι ιδιότητες που σχετίζονται με το Stimulus μπορεί να διαφέρουν από το τελικό ObservableProperty που θα γίνει αντικείμενο παρατήρησης. Είναι το συμβάν, όχι το αντικείμενο, που ενεργοποιεί τον Sensor.

Ιδιότητες που σχετίζονται με αυτή τη κλάση:

* **ssn:isProxyFor**: σχέση μεταξύ ενός Stimulus και της ιδιότητας για την οποία αποτελεί proxy. Για παράδειγμα, μια αύξηση ή μείωση της ταχύτητας των περιστρεφόμενων κυπέλλων σε έναν αισθητήρα ανέμου χρησιμεύει ως proxy για την ταχύτητα του ανέμου.
* **ssn:wasOriginatedBy**: σχέση μεταξύ ενός Observation και του Stimulus το οποίο το προκάλεσε.
* **ssn:detects**: σχέση μεταξύ ενός Sensor και του Stimulus που ανιχνεύει ο Sensor.

### Ενεργοποιήσεις (Actuations)

Στην εικόνα 25 φαίνονται οι βασικές κλάσεις και ιδιότητες των οντολογιών SSN και SOSA που σχετίζονται με τα Actuations.

A picture containing graphical user interface

Description automatically generated

Εικόνα 25 – Κλάσεις και ιδιότητες των SSN/SOSA σχετικά με Actuations

sosa:ActuableProperty

Μία ενεργοποιήσιμη ιδιότητα ενός FoT. Για παράδειγμα, ένας ενεργοποιητής παραθύρου ενεργεί αλλάζοντας την κατάσταση του παραθύρου μέσω ανοιχτού και κλειστού. Η δυνατότητα του παραθύρου να ανοίγει και να κλείνει είναι η ActuatableProperty του.

Αποτελεί υποκλάση τους ιδιότητας ssn:Property.

sosa:Actuation

Εκτελεί μία διαδικασία (Procedure) η οποία έχει ως αποτέλεσμα κάποια αλλαγή στη τρέχουσα κατάσταση του κόσμου, χρησιμοποιώντας έναν ενεργοποιητή (Actuator). Για παράδειγμα, το αυτόματο κλείσιμο του παραθύρου άμα η θερμοκρασία του δωματίου πέσει κάτω από 20 βαθμούς κελσίου. Η πράξη αποτελεί το Actuation και η συσκευή που κλείνει το παράθυρο αποτελεί τον Actuator.

Ιδιότητες που σχετίζονται με αυτή τη κλάση:

* **sosa:actsOnProperty**: σχέση μεταξύ ενός Actuation και της ιδιότητας την οποία επηρεάζει.
* **sosa:isActedOnBy**: αντίστροφη ιδιότητα της sosa:actsOnProperty

sosa:Actuator

Η συσκευή που εφαρμόζει ένα Actuation.

Ιδιότητες που σχετίζονται με αυτή τη κλάση:

* **sosa:madeActuation**: σχέση μεταξύ ενός Actuator και του Actuation που εφαρμόζει.
* **sosa:madeByActuator**: αντίστροφη ιδιότητα της sosa:madeActuation.

### Δειγματοληψία (Sampling)

Στην εικόνα 26 φαίνονται οι βασικές κλάσεις και ιδιότητες των οντολογιών SSN και SOSA που σχετίζονται με Sampling.

A picture containing graphical user interface

Description automatically generated

Εικόνα 26 – Κλάσεις και σχέσεις των SSN/SOSA σχετικά με Sampling

sosa:Sample

Τα δείγματα είναι συνήθως υποσύνολα ή αποσπάσματα από το FoT ενός Observation. Χρησιμοποιούνται σε περιπτώσεις όπου οι παρατηρήσεις δε μπορούν να γίνουν άμεσα σε ολόκληρο το FoT, είτε γιατί δεν είναι δυνατόν να παρατηρήσουμε ολόκληρο το εύρος του χαρακτηριστικού ή γιατί είναι πιο βολικό να χρησιμοποιηθεί κάποιος διακομιστής μεσολάβησης (proxy). Είναι επομένως τεχνουργήματα μίας στρατηγικής παρατήρησης και συνήθως δεν έχουν κάποια σημαντική λειτουργία πέρα από τον ρόλο τους στην διαδικασία τους παρατήρησης. Τα χαρακτηριστικά τους συνήθως παρουσιάζουν μικρό ενδιαφέρον.

Η κλάση sosa:Sample αποτελεί υποκλάση των sosa:FeatureOfInterest και sosa:Result. Ιδιότητες που σχετίζονται με αυτή τη κλάση:

* **sosa:hasSample**: σχέση μεταξύ ενός FoT και του δείγματος που χρησιμοποιείται για την αναπαράσταση του. Άρα, το domain της ιδιότητας περιλαμβάνει την κλάση sosa:FeatureOfInterest και το range την κλάση sosa:Sample.
* **sosa:isSampleOf**: αντίστροφη ιδιότητα της sosa:hasSample.

sosa:Sampling

Εκτέλεση ενός Procedure το οποίο έχει ως στόχο την δημιουργία ή τον μετασχηματισμό ενός ή περισσότερων Samples. Για παράδειγμα, επιλογή ενός υποσυνόλου του πληθυσμού ή λήψη αίματος από κάποιον ασθενή.

sosa:Sampler

Οποιαδήποτε συσκευή χρησιμοποιείται για την εκτέλεση κάποιου Sampling. Ιδιότητες που σχετίζονται με αυτή τη κλάση:

* **sosa:madeSampling**: σχέση μεταξύ ενός Sampler και του Sampling που εκτέλεσε. Άρα, το domain της ιδιότητας περιλαμβάνει την κλάση sosa:Sampler και το range αντίστοιχα την κλάση sosa:Sampling.
* **sosa:madeBySampler**: αντίστροφη ιδιότητα της sosa:madeSampling

### Χαρακτηριστικά Ενδιαφέροντος και Ιδιότητες

Στην εικόνα 27 φαίνονται οι βασικές κλάσεις και ιδιότητες των οντολογιών SSN και SOSA που σχετίζονται με FoT και properties. Βλέπουμε ότι η έννοια του ObservableProperty που αναφέρθηκε παραπάνω αποτελεί υποκλάση της Property και ότι τα properties συνδέονται με τα FoT με τη χρήση κάποιων ιδιοτήτων.

Diagram, text

Description automatically generated

Εικόνα 27 – Κλάσεις και ιδιότητες των SSN/SOSA σχετικά με FoT και Properties

sosa:FeatureOfInterest

Το αντικείμενο του οποίου κάποια ιδιότητα εκτιμάται ή υπολογίζεται και τη διάρκεια ενός Observation. Για παράδειγμα, αν υπολογίζεται το ύψος ενός κτιρίου, το ύψος αποτελεί το ObservableProperty ενώ το κτίριο αποτελεί το FoT.

Ιδιότητες που σχετίζονται με αυτή τη κλάση:

* **sosa:hasFeatureOfInterest**: σχέση μεταξύ ενός Observation και της οντότητας την οποία επιλέξαμε για την παρατήρηση κάποιας ιδιότητας τους, ή μεταξύ μίας δειγματοληψίας (Sampling) και της οντότητας που πήραμε ως δείγμα.
* **sosa:isFeatureOfInterestOf**: αντίστροφη ιδιότητα της sosa:hasFeatureOfInterest

ssn:Property

Μία ιδιότητα κάποιας οντότητας. Μία πτυχή της οντότητας που είναι εγγενής και δεν μπορεί να υπάρξει χωρίς την οντότητα.

Ιδιότητες που σχετίζονται με τη κλάση:

* **ssn:hasProperty**: σχέση μεταξύ μίας οντότητας με μία ιδιότητα τους.
* **ssn:isPropertyOf**: αντίστροφη ιδιότητα της ssn:hasProperty
* **ssn:forProperty**: σχέση μεταξύ κάποιας πτυχής μίας οντότητας με μία ιδιότητα. Για παράδειγμα, μεταξύ ενός Sensor και των ιδιοτήτων που μπορεί να παρατηρήσει.

### Διαδικασίες και Αποτελέσματα

Στις εικόνες 28 και 29 βλέπουμε κάποιες βασικές έννοιες των οντολογιών SSN και SOSA σχετικές με τις διαδικασίες (procedures) και τα αποτελέσματα τους (results).

A picture containing diagram

Description automatically generated

Εικόνα 28 – Κλάσεις και ιδιότητες των SSN/SOSA σχετικά με Procedures

Diagram

Description automatically generated

Εικόνα 29 – Κλάσεις και ιδιότητες των SSN/SOSA σχετικά με Results

sosa:Procedure

Ένα πρωτόκολλο, ένα σχέδιο, τους αλγόριθμος ή μία υπολογιστική μέθοδος η οποία καθορίζει, μεταξύ άλλων, το πως να γίνεται ένα Observation, ή πως να δημιουργηθεί ένα Sample. Μία διαδικασία είναι επαναχρησιμοποιήσιμη και μπορεί να εμπλέκεται σε πολλά Observations και Samplings. Εξηγεί τα βήματα που πρέπει να γίνουν για να φτάσουμε σε επιθυμητά και χρήσιμα αποτελέσματα.

Ιδιότητες που σχετίζονται με αυτή τη κλάση:

* **sosa:usedProcedure**: σχέση μεταξύ ενός επαναχρησιμοποιήσιμου Procedure και της πράξης για την οποία χρησιμοποιήθηκε, όπως ένα Observation ή ένα Sampling.
* **ssn:implements**: σχέση μεταξύ μιας οντότητας που εφαρμόζει ένα Procedure με κάποιο εκτελέσιμο τρόπο και του Procedure (αλγόριθμος, διαδικασία ή μέθοδος).
* **ssn:implementedBy**: αντίστροφη ιδιότητα της ssn:implements

ssn:Input

Κάθε πληροφορία που παρέχεται σε ένα Procedure για να τη χρησιμοποιήσει

* **ssn:hasInput**: σχέση μεταξύ ενός Procedure και του Input του.

ssn:Output

Κάθε πληροφορία που παράγεται από ένα Procedure.

* **ssn:HasOutput**: σχέση μεταξύ ενός Procedure και του Output του.

sosa:Result

Το αποτέλεσμα ενεργειών όπως Observation και Sampling.

* **sosa:hasResult**: σχέση μεταξύ μίας ενέργειας (Observation, Sampling, …) και του Result που αυτή δίνει.
* **sosa:isResultOf**: αντίστροφη ιδιότητα της sosa:hasResult.
* **sosa:hasSimpleResult**: η απλή τιμή που επιστρέφεται από μία ενέργεια, όπως Observation. Για παράδειγμα, η τιμή 15 ή η τιμή true.
* **sosa:resultTime**:είναι η χρονική στιγμή που ολοκληρώθηκε μια ενέργεια. Το range της ιδιότητας περιλαμβάνει τιμές τύπου xsd:dateTime.

### Συστήματα και Ανάπτυξη τους

Στην εικόνα 30 βλέπουμε κάποιες βασικές έννοιες των οντολογιών SSN και SOSA σχετικά με τα συστήματα (systems) και την ανάπτυξη τους (deployment). Όπως φαίνεται, οι οντότητες που έχουμε περιγράψει (Sensor, Actuator, Sampler) αποτελούν υποκλάση του System, το οποίο τίθεται σε ανάπτυξη μέσω του Deployment και γίνεται host σε κάποιο Platform.

Diagram

Description automatically generated with medium confidence

Εικόνα 30 – Κλάσεις και ιδιότητες των SSN/SOSA σχετικά με τα Systems και το Deployment

sosa:Platform

Αποτελεί μία οντότητα η οποία «φιλοξενεί» τους οντότητες, τους Sensors, Samplers, και άλλα Platforms. Για παράδειγμα, ένα όχημα, τους δορυφόρος, ένα τηλέφωνο, ένας άνθρωπος ή ένα ζώο, θα μπορούσαν να συμπεριφέρονται ως Platforms (τεχνικά ή βιολογικά) για Sensors ή Actuators.

Ιδιότητες που σχετίζονται με αυτή τη κλάση:

* **sosa:hosts**: σχέση μεταξύ ενός Platform και της οντότητας την οποία φιλοξενεί.
* **sosa:isHostedBy**: αντίστροφη ιδιότητα της sosa:hosts

ssn:System

Είναι μία μονάδα αφαίρεσης για τα κομμάτια υποδομής που εφαρμόζουν Procedures. Ένα σύστημα μπορεί να περιέχει κάποια στοιχεία, που καλούνται τα υποσυστήματα του, και τα οποία είναι άλλα συστήματα.

* **ssn:hasSubSystem**: σχέση μεταξύ ενός System και ενός από τα στοιχεία του.

ssn:Deployment

Περιγράφει την ανάπτυξη ενός ή περισσότερων Systems για κάποιον συγκεκριμένο σκοπό. Το Deployment μπορεί να γίνει σε κάποιο Platform. Για παράδειγμα, η εγκατάσταση ενός Sensor που μετράει την θερμοκρασία σε έναν τοίχο.

Ιδιότητες που σχετίζονται με αυτή τη κλάση:

* **ssn:deployedSystem**: σχέση μεταξύ ενός Deployment και του Platform πάνω στο οποίο έχουν αναπτυχθεί τα Systems.
* **ssn:hasDeployment**: σχέση μεταξύ ενός System και ενός Deployment, δεδομένου ότι το System έχει αναπτυχθεί στο συγκεκριμένο Deployment.
* **ssn:deployedOnPlatform**: σχέση μεταξύ ενός Deployment και του Platform στο οποίο έχουν αναπτυχθεί τα Systems.
* **ssn:inDeployment**: αντίστροφη ιδιότητα της ssn:deployedOnPlatform.

1. Υλοποίηση

ΥΛΟΠΟΙΗΣΗ

# Περιγραφή Του Προβλήματος

Στόχος της εργασίας είναι η συλλογή και η αναπαράσταση δεδομένων από fitness trackers, με τρόπο που να επιτρέπει τους χρήστες να θέτουν ερωτήματα και να εξάγουν συμπεράσματα από αυτά. Τα δεδομένα που συλλέγονται αρχικά πρέπει να μετασχηματισθούν σε RDF τριπλέτες και να ενσωματωθούν σε μία οντολογία, η οποία έχει κατασκευαστεί για την αναπαράσταση τους. Έπειτα, οι τριπλέτες πρέπει να αποθηκευτούν σε μία σημασιολογική βάση δεδομένων, όπως η GraphDB. Τέλος, πρέπει να κατασκευαστεί μία web εφαρμογή με dashboards, η οποία θα συνδεθεί με τ ην βάση και θα παρουσιάζει τα δεδομένα στον χρήστη και θα του επιτρέπει να θέσει ερωτήματα πάνω σε αυτά. Ο κώδικας βρίσκεται στο παρακάτω Github repository και οδηγίες για την εκτέλεση του μπορούν να βρεθούν στο Παράρτημα ΙΙΙ ή στο README αρχείο του repository.

<https://github.com/AndreasNbt/FitnessData>

## Επιλογή Dataset

Το πρώτο βήμα για την εκπόνηση της εργασίας ήταν η εύρεση του συνόλου δεδομένων που θα χρησιμοποιηθεί. Το σύνολο αυτό θα έπρεπε να περιλαμβάνει μία ικανοποιητική ποσότητα δεδομένων από κάποιο fitness tracker, που θα επιτρέπει την εξαγωγή ουσιαστικών συμπερασμάτων.

Τελικά, το dataset που επιλέχθηκε είναι αυτό που βρίσκεται στην ακόλουθη διεύθυνση: <https://www.kaggle.com/datasets/arashnic/fitbit>. Περιλαμβάνει δεδομένα από 30 διαφορετικούς χρήστες κάποιας συσκευής Fitbit, οι οποίοι έδωσαν την άδεια για την καταγραφή των δεδομένων τους για μία συγκεκριμένη χρονική περίοδο, για μία έρευνα που διεξαγόταν από την Amazon Mechanical Turk. Έγινε παρατήρηση και καταγραφή διάφορων δεδομένων όπως αριθμός βημάτων, αριθμός θερμίδων που κάηκαν, καρδιακοί παλμοί, δεδομένα ύπνου, διάφορες μετρικές όπως βάρος, BMI, και άλλα. Επίσης, χωρίζονται σε δεδομένα ανά μέρα, ανά ώρα, ανά λεπτό, μέχρι και ανά δευτερόλεπτο, μόνο για την μετρική των καρδιακών παλμών.

Στην εικόνα 31 φαίνεται ένα μικρό τμήμα από τα δεδομένα. Η πρώτη στήλη περιέχει το ID του χρήστη τον οποίο αφορούν οι μετρήσεις. Στη δεύτερη στήλη βρίσκεται η ημερομηνία και η ώρα που συλλέχθηκαν τα δεδομένα. Στις υπόλοιπες στήλες βρίσκονται τα δεδομένα που μας ενδιαφέρουν. Για παράδειγμα, στο συγκεκριμένο αρχείο υπάρχει το βάρος του χρήστη, το ποσοστό λίπους και το BMI. Οι δύο τελευταίες στήλες δεν περιλαμβάνουν δεδομένα που μας ενδιαφέρουν οπότε όταν γίνεται η φόρτωση των αρχείων στο πρόγραμμα που είναι υπεύθυνο για τον μετασχηματισμό των δεδομένων, απλά διαγράφονται.

A picture containing text, screenshot, number, font

Description automatically generated

Εικόνα 31 – Δείγμα από το επιλεγμένο dataset

# Εργαλεία – Frameworks που Χρησιμοποιήθηκαν

Για την υλοποίηση χρησιμοποιήθηκαν διάφορα εργαλεία και τεχνολογίες, ανάλογα με τις ανάγκες του κάθε τμήματος της εργασίας. Αρχικά, για τον σχεδιασμό της οντολογίας, χρησιμοποιήθηκε το Protégé, το οποίο είναι ένα εργαλείο για την δημιουργία οντολογιών, που βασίζεται στις γλώσσες RDF και OWL. Αναλυτική περιγραφή αυτών των τεχνολογιών δόθηκε στο κεφάλαιο 2 της εργασίας. Χρησιμοποιούνται για αναπαράσταση δεδομένων στον σημασιολογικό ιστό καθώς και για την δημιουργία οντολογιών. Περαιτέρω ανάλυση της οντολογίας θα γίνει παρακάτω. Για τον μετασχηματισμό των δεδομένων από τα .csv αρχεία και την φόρτωση τους στην οντολογία υλοποιήθηκε ένα πρόγραμμα με τη γλώσσα προγραμματισμού Python. Το πρόγραμμα αυτό διαβάζει όλα τα αρχεία του dataset και κάνει τους κατάλληλους μετασχηματισμούς σε αυτά. Είναι υπεύθυνο επίσης για την κατασκευή της οντολογίας και την φόρτωση όλων των δεδομένων σε αυτήν. Για την επικύρωση των δεδομένων, χρησιμοποιείται η γλώσσα SHACL. Μέσω αυτής ορίστηκε ένα σύνολο από κανόνες τους οποίους πρέπει να ικανοποιεί η παραγόμενη οντολογία, έτσι ώστε να είναι έγκυρη. Τέλος, τα παραγόμενα δεδομένα αποθηκεύονται σε μία βάση GraphDB.

# Αρχιτεκτονική

A screenshot of a diagram

Description automatically generated with low confidence

# Οντολογία

Για την αναπαράσταση των δεδομένων στον Σημασιολογικό Ιστό απαιτούνταν η δημιουργία μίας οντολογίας με τις κατάλληλες κλάσεις και ιδιότητες για την αναπαράσταση τους. Η οντολογία δημιουργήθηκε αρχικά χειροκίνητα χρησιμοποιώντας το Protégé. Επιπλέον, υπάρχει η δυνατότητα και αυτόματης δημιουργίας της από το python πρόγραμμα που κατασκευάστηκε για τον μετασχηματισμό των δεδομένων, με χρήση της βιβλιοθήκης owlready2. Η οντολογία βασίζεται στο μοντέλο των SSN και SOSA, οπότε χρησιμοποιεί κάποιες κλάσεις και ιδιότητες που αναλύθηκαν παραπάνω.

## Περιγραφή Οντολογίας

Όπως αναφέρθηκε, το dataset περιέχει παρατηρήσεις δεδομένων από διάφορους ανθρώπους, οι οποίες γινόντουσαν ανά μέρα, ανά ώρα ή ανά λεπτό, ανάλογα με το είδος των δεδομένων. Οπότε, χρειάζονταν αρχικά 3 βασικές κλάσεις για την αναπαράσταση των δεδομένων:

* **Person**, για την αναπαράσταση των ανθρώπων,
* **Observation**, για την αναπαράσταση των παρατηρήσεων, και
* **Result**, για την αναπαράσταση των αποτελεσμάτων των παρατηρήσεων.

Η κλάση **Person** αποτελεί υποκλάση τους κλάσης **FeatureOfInterest**, καθώς σύμφωνα με τον ορισμό της, οι άνθρωποι σε αυτή τη περίπτωση αποτελούν τα χαρακτηριστικά ενδιαφέροντος των οποίων οι ιδιότητες τίθενται υπό παρατήρηση. Αυτές είναι οι κλάσεις που χρειάζονται για την αναπαράσταση των δεδομένων. Επιπλέον χρειάζονται και κάποιες ιδιότητες, ώστε να γίνουν οι κατάλληλες συσχετίσεις μεταξύ των δεδομένων:

* **hasFeatureOfInterest:** ιδιότητα που συσχετίζει ένα Observation με το FeatureOfInterest το οποίο παρατηρείται. Έχει ως domain την κλάση Observation και ως range την κλάση FeatureOfInterest.
* **isFeatureOfInterestOf:** αντίστροφη ιδιότητα της hasFeatureOfInterest**.**
* **observedPerson:** συσχετίζει ένα Observation με το αντίστοιχο στιγμιότυπο τύπου Person το οποίο παρατηρείται. Καθώς η κλάση Person είναι υποκλάση της FeatureOfInterest, έτσι και η ιδιότητα observedPerson αποτελεί υπο-ιδιότητα της hasFeatureOfInterest. Έχει ως domain την κλάση Observation και ως range την κλάση Person.
* **hasResult:** συσχετίζει ένα Observation με το αντίστοιχο Result του. Έχει ως domain την κλάση Observation και ως range την κλάση Result.
* **personId, observationId, resultId:** μοναδικό ID για κάθε στιγμιότυπο αυτών των κλάσεων. Για την κλάση Person, το ID παίρνεται από την πρώτη στήλη των αρχείων του dataset. Για τις κλάσεις Observation και Result είναι ένας αύξων αριθμός.

Επιπλέον, με την κλάση Result συσχετίζονται και κάποιες ιδιότητες οι οποίες περιλαμβάνουν τις τιμές όλων των παρατηρήσεων. Οι ιδιότητες αυτές αποτελούν datatype properties και έχουν ως domain την κλάση Result και ως range τους τύπους string, integer, decimal, ανάλογα με τις τιμές. Για παράδειγμα, στην εικόνα 31, οι στήλες C-F περιέχουν τιμές που αφορούν τα WeightKg, WeightPounds, Fat, και BMI. Για την αναπαράσταση αυτών των τιμών χρησιμοποιούνται οι ιδιότητες **hasWeightKg**, **hasWeightPounds**, **hasFat**, και **hasBMI**. Αντίστοιχα, υπάρχουν και άλλες παρόμοιες ιδιότητες για τα υπόλοιπα δεδομένα.

Στην εικόνα 32 φαίνεται ένα δείγμα από ένα άλλο αρχείο του dataset, που περιλαμβάνει δεδομένα των χρηστών σχετικά με τον ύπνο. Στην στήλη B φαίνεται η μέρα που έγινε η παρατήρηση, στην στήλη C υπάρχει η ιδιότητα TotalSleepRecords, που δείχνει πόσες φορές κοιμήθηκε ο χρήστης μέσα στη μέρα, στη στήλη D η ιδιότητα TotalMinutesAsleep, δηλαδή τα συνολικά λεπτά όπου ο χρήστης κοιμόταν και τέλος η ιδιότητα TotalTimeInBed, δηλαδή τα συνολικά λεπτά όπου ο χρήστης βρισκόταν στο κρεβάτι του. Για την αναπαράσταση αυτών των τιμών δημιουργήθηκαν οι ιδιότητες **hasTotalSleepRecords**, **hasTotalMinutesAsleep**, και **hasTotalTimeInBed**.

A screenshot of a data

Description automatically generated with medium confidence

Εικόνα 32 – Δείγμα από τα δεδομένα ύπνου των χρηστών

## Δημιουργία οντολογίας με owlready2

Η οντολογία αρχικά δημιουργήθηκε με το Protégé. Αργότερα όμως, για μεγαλύτερη ευελιξία και για εξοικονόμηση χρόνου, υλοποιήθηκε και με python χρησιμοποιώντας την βιβλιοθήκη owlready2[[10]](#footnote-10). Μέσω του API της επιτρέπει την δημιουργία και την φόρτωση οντολογιών, την δημιουργία κλάσεων, ιδιοτήτων και στιγμιότυπων, και γενικότερα υποστηρίζει όλες τις λειτουργίες που παρέχει η OWL. Η διαδικασία είναι αρκετά απλή. Στην εικόνα 33 φαίνεται η συνάρτηση **loadOntology()** όπου γίνεται η δημιουργία της οντολογίας. Για να γίνει αυτό χρησιμοποιείται η συνάρτηση get\_ontology(), η οποία δέχεται μία παράμετρο, το IRI της οντολογίας ή, αν πρόκειται για τοπικό αρχείο, το path του αρχείου. Δίνοντας ένα path ως παράμετρο, δημιουργείται ένα νέο αρχείο μίας OWL οντολογίας στην αντίστοιχη τοποθεσία.

A screen shot of a computer code

Description automatically generated with low confidence

Εικόνα 33 – Δημιουργία τους οντολογίας

Αφού δημιουργηθεί η οντολογία, πρέπει να προστεθούν οι κλάσεις και οι ιδιότητες. Στην εικόνα 34 φαίνεται ένα μέρος της συνάρτησης **addClassesAndProperties()**, η οποία δέχεται ως παράμετρο την οντολογία και προσθέτει σε αυτήν τις κλάσεις και τους ιδιότητες που αναφέρθηκαν παραπάνω. Οι κλάσεις δημιουργούνται στο πάνω μέρος με χρήση της συνάρτησης **new\_class()** του **types** module. Ως πρώτη παράμετρος δίνεται το όνομα της κλάσης και ως δεύτερη παράμετρος η κλάση της οποίας θα αποτελεί υποκλάση. Για παράδειγμα, η κλάση Person είναι υποκλάση της FeatureOfInterest, οπότε η κλάση FeatureOfInterest δίνεται ως δεύτερη παράμετρος. Έπειτα γίνεται η δημιουργία των ιδιοτήτων. Δημιουργούνται όπως μία κλάση στην python, με χρήση του **class** keyword. Το σύμβολο ‘>>’ αποτελεί συντόμευση για την δήλωση του domain και του range μίας ιδιότητας. Αν ως παράμετρος δοθεί μία άλλη ιδιότητα, αυτό σημαίνει ότι η ιδιότητα που θα δημιουργηθεί θα είναι υπο-ιδιότητα της. Παρακάτω δημιουργούνται και όλες οι υπόλοιπες ιδιότητες, με τον ίδιο τρόπο.

A picture containing text, screenshot, font

Description automatically generated

Εικόνα 34 – Προσθήκη κλάσεων και ιδιοτήτων

# Μετασχηματισμός Δεδομένων

Αφού κατασκευάζεται η οντολογία και δημιουργούνται όλες οι απαραίτητες κλάσεις και ιδιότητες, το επόμενο βήμα είναι ο μετασχηματισμός των δεδομένων από τα csv αρχεία και η φόρτωση τους στην οντολογία. Στην εικόνα 35 φαίνεται το αρχείο main.py του project. Αρχικά, καλεί την συνάρτηση loadOntology(), της οποίας η περιγραφή έγινε παραπάνω, η οποία δημιουργεί την οντολογία, προσθέτει όλες τις απαραίτητες κλάσεις και ιδιότητες, και έπειτα την επιστρέφει. Έπειτα, καλείται η συνάρτηση loadData(), η οποία δέχεται ως παράμετρο την οντολογία που δημιουργήθηκε και είναι υπεύθυνη για τον μετασχηματισμό των δεδομένων και την δημιουργία όλων των απαραίτητων στιγμιοτύπων της οντολογίας.

A screen shot of a computer code

Description automatically generated with low confidence

Εικόνα 35 – Αρχείο main.py

Στην εικόνα 36 φαίνεται η συνάρτηση loadData(), από την οποία ξεκινάει η διαδικασία του μετασχηματισμού των δεδομένων. Αρχικά, δημιουργείται ένας κενός πίνακας people, ο οποίος θα περιλαμβάνει όλα τα id των ανθρώπων που έχουν εισαχθεί στην οντολογία. Αυτό θα χρειαστεί ώστε να αναγνωρίζονται τα κοινά id που υπάρχουν σε διαφορετικά αρχεία και να μην δημιουργούνται επιπλέον στιγμιότυπα τύπου Person με τα ίδια id. Η μεταβλητή filenum είναι μία βοηθητική μεταβλητή που χρησιμοποιείται αργότερα για την αναγνώριση του κάθε αρχείου. Αντίστοιχα και οι μεταβλητές observationCount και resultCount, οι οποίες χρησιμοποιούνται για την μέτρηση των observations και results αντίστοιχα, και για τον ορισμό των id τους. Μετά από αυτές τις δηλώσεις, ακολουθεί μία επαναληπτική δομή, η οποία καλεί την συνάρτηση loadFileData() για κάθε αρχείο του dataset και ενημερώνει κατάλληλα την μεταβλητή filenum. Η συνάρτηση, αφού κάνει τον μετασχηματισμό των δεδομένων του αρχείου, επιστρέφει τις ενημερωμένες τιμές των μεταβλητών observationCount και resultCount. Το files είναι ένας πίνακας που έχει δημιουργηθεί, ο οποίος περιέχει όλα τα paths των αρχείων που χρησιμοποιούνται.

A picture containing text, multimedia software, screenshot

Description automatically generated

Εικόνα 36 – Συνάρτηση loadData

Στις παρακάτω εικόνες φαίνεται ο κώδικας της συνάρτησης loadFileData(). Η συνάρτηση αυτή δέχεται 4 παραμέτρους, την οντολογία, τον πίνακα people, το path του αρχείου που θα διαβαστεί, και το filenum. Αρχικά, διαβάζονται τα περιεχόμενα του αρχείου με τη χρήση της συνάρτησης read\_csv() από την βιβλιοθήκη pandas[[11]](#footnote-11) και τα κενά κελιά συμπληρώνονται με 0.

A picture containing text, screenshot, font

Description automatically generated

Εικόνα 37 – Χρήση pandas για το διάβασμα των αρχείων

Ακολουθεί μία επαναληπτική δομή, όπου τα δεδομένα του αρχείου διαβάζονται ανά γραμμή. Για κάθε γραμμή, δημιουργείται ένα νέο στιγμιότυπο των κλάσεων Observation και Result, τα οποία συνδέονται μεταξύ τους μέσω της ιδιότητας hasResult. Επιπλέον, θέτονται τα ids αυτών των στιγμιοτύπων με την χρήση των αντίστοιχων ιδιοτήτων και των μεταβλητών observationCount και resultCount, των οποίων οι τιμές στη συνέχεια αυξάνονται κατά 1.

A picture containing text, screenshot, font

Description automatically generated

Εικόνα 38 – Διάσχιση του αρχείου ανά γραμμή

Έπειτα, αφού έχουμε τα δεδομένα της κάθε γραμμής, πρέπει να εξεταστεί κάθε κελί ξεχωριστά. Όλα τα αρχεία έχουν μία σταθερή και συνεπή μορφή, δηλαδή στη πρώτη στήλη βρίσκεται πάντα το id του ανθρώπου, στη δεύτερη στήλη η ημερομηνία, και έπειτα ακολουθούν τα υπόλοιπα δεδομένα. Για αυτό χρησιμοποιείται η μεταβλητή column\_index, ώστε να καταλαβαίνει το πρόγραμμα σε ποια στήλη του αρχείου βρίσκεται και να εκτελεί τις κατάλληλες ενέργειες.

Στην εικόνα 39 φαίνονται οι ενέργειες που εκτελεί το πρόγραμμα για τα δεδομένα τους πρώτης στήλης. Αρχικά, τα δεδομένα διαβάζονται ανά στήλη και αποθηκεύονται στη μεταβλητή data. Αν το column\_index είναι ίσο με 0, σημαίνει ότι βρισκόμαστε στη πρώτη στήλη και στη μεταβλητή data είναι αποθηκευμένο αυτή τη στιγμή το id του χρήστη. Πρέπει αρχικά να ελέγξουμε αν αυτός ο χρήστης έχει ήδη εισαχθεί στην οντολογία. Οπότε ψάχνουμε να βρούμε αν το id του χρήστη υπάρχει ήδη στον πίνακα people. Αν υπάρχει, τότε βρίσκουμε το αντίστοιχο στιγμιότυπο της κλάσης Person χρησιμοποιώντας τη συνάρτηση instances(), και θέτουμε την ιδιότητα observedPerson χρησιμοποιώντας ως subject το στιγμιότυπο τύπου Observation που κατασκευάστηκε παραπάνω και ως object τον Person που μόλις πήραμε. Αν δεν υπάρχει, τότε δημιουργούμε ένα νέο στιγμιότυπο τύπου Person και θέτουμε το id του χρησιμοποιώντας την μεταβλητή data. Θέτουμε με τον ίδιο τρόπο την ιδιότητα observedPerson και προσθέτουμε το id στον πίνακα people ώστε να μη δημιουργηθεί μελλοντικά επιπλέον στιγμιότυπο για τον ίδιο χρήστη.

A picture containing text, screenshot, font

Description automatically generated

Εικόνα 39 – Δημιουργία στιγμιότυπων τύπου Person

Για τα δεδομένα που βρίσκονται τους υπόλοιπες στήλες, θέτονται οι αντίστοιχες ιδιότητες. Καθώς κάθε αρχείο περιλαμβάνει διαφορετικά δεδομένα, έχει δημιουργηθεί μία διαφορετική συνάρτηση για κάθε αρχείο. Για να γίνει η κλήση της σωστής συνάρτησης χρησιμοποιούμε την μεταβλητή filenum που αναφέρθηκε παραπάνω. Χρησιμοποιώντας ένα match-case statement τους python, καλείται η αντίστοιχη συνάρτηση ανάλογα με την τιμή τους μεταβλητής, όπως φαίνεται στην εικόνα 40.

A picture containing text, screenshot, font

Description automatically generated

Εικόνα 40 – Κλήση τους κατάλληλης συνάρτησης για το κάθε αρχείο

Για παράδειγμα, έστω ότι διαβάζουμε το πρώτο αρχείο, το οποίο είναι το weightLogInfo. Δείγμα του αρχείου φαίνεται στην εικόνα 31. Τα δεδομένα που περιέχει το αρχείο, πέρα από τα id των χρηστών, είναι η ημερομηνία και ώρα της παρατήρησης, βάρος σε kg, βάρος σε pounds, fat, και bmi. Οπότε, πρέπει να γίνει χρήση των ιδιοτήτων της οντολογίας που αντιστοιχούν σε αυτά τα δεδομένα. Στην εικόνα 41 φαίνεται η συνάρτηση loadWeightLogInfo(). Δέχεται ως παραμέτρους την μεταβλητή column\_index ώστε να μπορεί να καθορίσει σε ποια στήλη βρίσκεται για την χρησιμοποιήσει την κατάλληλη ιδιότητα, το αντίστοιχο στιγμιότυπο result που δημιουργήθηκε παραπάνω, καθώς και τα δεδομένα που είναι αποθηκευμένα στη μεταβλητή data. Χρησιμοποιώντας άλλο ένα match-case, ελέγχει την τιμή του column\_index και δημιουργεί μία καινούρια τριπλέτα. Για παράδειγμα, αν το column\_index είναι ίσο με 1, σημαίνει ότι βρισκόμαστε στη δεύτερη στήλη, όπου είναι αποθηκευμένη η ημερομηνία και η ώρα της παρατήρησης.

A screen shot of a computer program

Description automatically generated with low confidence

Εικόνα 41 – Συνάρτηση loadWeightLogInfo()

Για την αποθήκευση της ημερομηνίας και της ώρας συγκεκριμένα, καλείται η συνάρτηση setDateTime(), η οποία φαίνεται στην εικόνα 42. Η ιδιότητα resultTime, η οποία χρησιμοποιείται για την αναπαράσταση αυτής της πληροφορίας στην οντολογία, έχει ως range τον τύπο δεδομένων datetime, που δίνεται από την βιβλιοθήκη owlready2. Για την δημιουργία ενός νέου datetime object χρειάζεται η πληροφορία που διαβάζεται από το αρχείο, η οποία είναι η ημερομηνία και η ώρα ενωμένη ως μία συμβολοσειρά, να χωριστεί στα επιμέρους τμήματα της (έτος, μήνας, μέρα, ώρα, λεπτό), τα οποία στη συνέχεια θα αποθηκευτούν στις αντίστοιχες μεταβλητές. Αφού γίνει αυτό, δημιουργείται στο τέλος η επιθυμητή τριπλέτα με την ιδιότητα resultTime δίνοντας ως object ένα νέο datetime με ορίσματα τις μεταβλητές που δημιουργήθηκαν. Για κάποια αρχεία χρειάστηκε η υλοποίηση κάποιων επιπλέον ελέγχων, καθώς μπορεί να είχαν μόνο ημερομηνία και όχι ώρα, ή μπορεί η ώρα να μην ήταν σε εικοσιτετράωρη μορφή αλλά να περιλάμβανε «PM» ή «AM». Στη περίπτωση αυτή θα θέλαμε να μην υπάρχουν αυτά τα σύμβολα και η ώρα να αποθηκεύεται σε εικοσιτετράωρη μορφή.

A screen shot of a computer screen

Description automatically generated with low confidence

Εικόνα 42 – Συνάρτηση setDateTime()

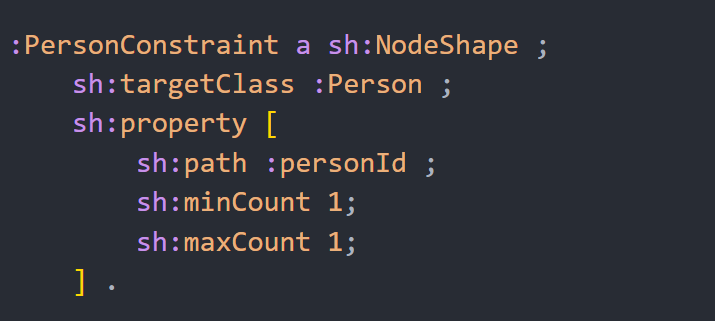
Με τον ίδιο τρόπο έχουν υλοποιηθεί συναρτήσεις και για τα υπόλοιπα αρχεία. Επιπλέον, ενημερώνεται σε κάθε επανάληψη και η τιμή της μεταβλητής column\_index ώστε να δείχνει πάντα στη σωστή στήλη. Η διαδικασία αυτή επαναλαμβάνεται για κάθε γραμμή κάθε αρχείου έως ότου να διαβαστούν όλα τα δεδομένα του dataset και να δημιουργηθούν όλες οι επιθυμητές τριπλέτες. Τέλος, όπως φαίνεται στην εικόνα 35, η οντολογία αποθηκεύεται σε ένα αρχείο Data.owl, το οποίο θα χρησιμοποιηθεί για την επικύρωση και την αποθήκευση των αποτελεσμάτων.

# Επικύρωση Αποτελεσμάτων

Αφού έχει κατασκευαστεί η οντολογία, πριν γίνει η αποθήκευση της στη βάση πρέπει πρώτα να επικυρωθεί ώστε να είμαστε βέβαιοι ότι είναι συνεπής και δεν περιέχει λάθη. Για τον σκοπό αυτό χρησιμοποιείται η γλώσσα περιορισμών SHACL[[12]](#footnote-12). Η SHACL χρησιμοποιείται για την επικύρωση RDF γράφων ελέγχοντας αν ικανοποιούν κάποιους προκαθορισμένους κανόνες. Αυτοί οι κανόνες παρέχονται ως σχήματα (shapes) και ως δομές που εκφράζονται στη μορφή ενός γράφου RDF. Οι γράφοι RDF που χρησιμοποιούνται με αυτόν τον τρόπο ονομάζονται «γράφοι σχημάτων» στο SHACL και οι γράφοι RDF που επικυρώνονται έναντι ενός γράφου σχημάτων ονομάζονται «γράφοι δεδομένων».

Αρχικά, πριν ξεκινήσει η διαδικασία της επικύρωσης, χρειάζεται να καθοριστεί το σύνολο των κανόνων που θα χρησιμοποιηθεί. Στην οντολογία μας υπάρχουν 3 βασικές κλάσεις των οποίων τα στιγμιότυπα θέλουμε να ακολουθούν τους κανόνες, οι κλάσεις Person, Observation, και Result. Οπότε, δημιουργήθηκαν 3 σχήματα, ένα για κάθε κλάση, όπου το καθένα περιλαμβάνει κάποιους κανόνες για κάποιες από τις ιδιότητες των κλάσεων.

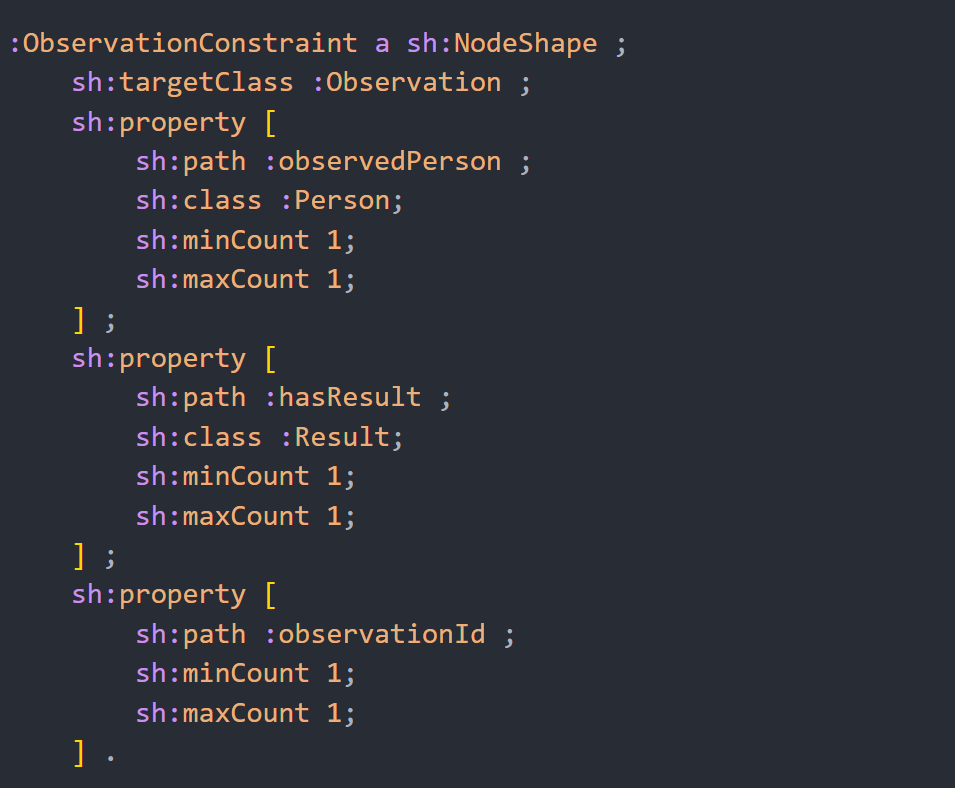
Στην εικόνα 43 φαίνεται ο περιορισμός που δημιουργήθηκε για την κλάση Person. Έχει μέσα έναν κανόνα ο οποίος λέει ότι κάθε στιγμιότυπο τύπου Person θα πρέπει να ανήκει σε μία ακριβώς τριπλέτα που έχει ως predicate την ιδιότητα personId. Με άλλα λόγια, κάθε Person έχει ένα μοναδικό id. Το sh:NodeShape αποτελεί υποκλάση του γενικότερου sh:Shape και δηλώνει ότι το σχήμα που δημιουργούμε αφορά τα στιγμιότυπα μίας συγκεκριμένης κλάσης του RDF γράφου. Το sh:targetClass δηλώνει την κλάση την οποία αφορά ο περιορισμός. Επίσης, μέσα στο sh:property μπλοκ, το sh:path δηλώνει την ιδιότητα που θέλουμε να επαληθεύσουμε, σε αυτή τη περίπτωση το personId.



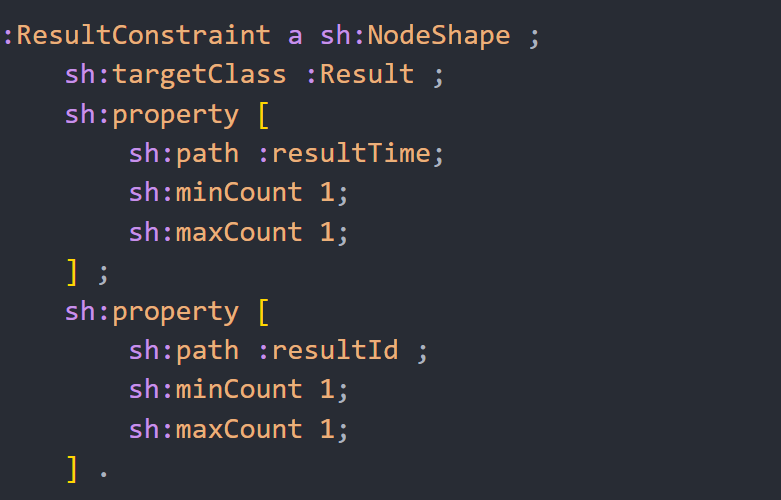
Εικόνα 43 – Shape για την κλάση Person

Στην εικόνα 44 φαίνεται το shape που δημιουργήθηκε για την κλάση Observation, το οποίο περιλαμβάνει 3 κανόνες. Ο πρώτος αφορά την ιδιότητα observedPerson και λέει ότι κάθε στιγμιότυπο τύπου Observation θα πρέπει να συνδέεται με ακριβώς ένα στιγμιότυπο τύπου Person μέσω της observedPerson. Ο δεύτερος αντίστοιχα λέει ότι κάθε Observation θα πρέπει να συνδέεται με ακριβώς ένα στιγμιότυπο τύπου Result μέσω της ιδιότητας hasResult. Ο τελευταίος λέει ότι κάθε Observation θα πρέπει να έχει ακριβώς ένα id. Το sh:class, που βρίσκεται μέσα στο sh:property μπλοκ σε έναν από τους κανόνες, δηλώνει ότι η τιμή της αντίστοιχης ιδιότητας θα πρέπει να είναι του τύπου που δηλώνεται δίπλα από το sh:class.

Τέλος, στην εικόνα 45 φαίνεται το shape που δημιουργήθηκε για την κλάση Result, το οποίο περιλαμβάνει 2 κανόνες. Ο πρώτος λέει ότι κάθε Result πρέπει να έχει ακριβώς μία ημερομηνία-ώρα, μέσω της ιδιότητας resultTime, και ο δεύτερος ότι κάθε Result θα πρέπει να έχει ακριβώς ένα id.

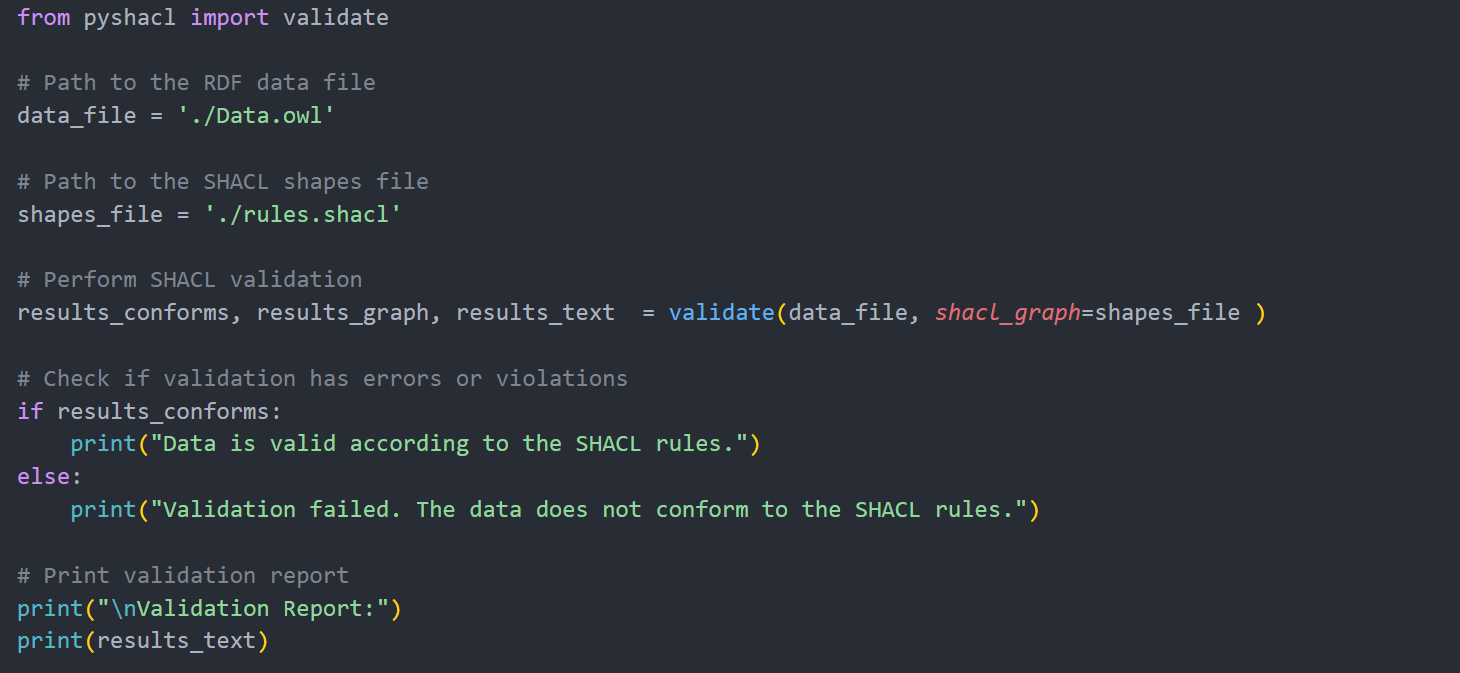


Εικόνα 44 – Shape για την κλάση Observation

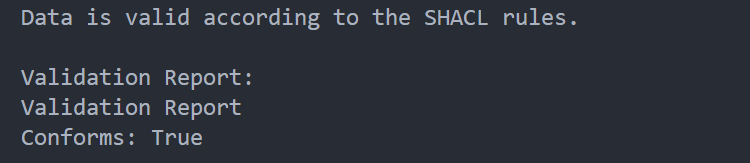


Εικόνα 45 – Shape για την κλάση Result

Αφού έχει οριστεί το σύνολο των κανόνων, το επόμενο βήμα είναι η επικύρωση της οντολογίας. Στην εικόνα 46 φαίνεται το πρόγραμμα που υλοποιήθηκε σε python για να εκτελέσει τη διαδικασία τους επικύρωσης, με χρήση της βιβλιοθήκης pyshacl. Αρχικά δηλώνονται τα δύο αρχεία που απαιτούνται για την επικύρωση, το αρχείο των δεδομένων (data\_file), το οποίο είναι η οντολογία που κατασκευάσαμε, και το αρχείο με τα shapes, δηλαδή τους κανόνες (shapes\_file). Έπειτα, καλείται η συνάρτηση validate() που εκτελεί την επικύρωση. Τέλος, ελέγχεται αν η επικύρωση ολοκληρώθηκε χωρίς λάθη μέσω της μεταβλητής results\_conforms και εμφανίζονται οι ανάλογες πληροφορίες στον χρήστη. Εκτελώντας το πρόγραμμα, η οντολογία επαληθεύεται επιτυχώς, τους φαίνεται στην εικόνα 47.

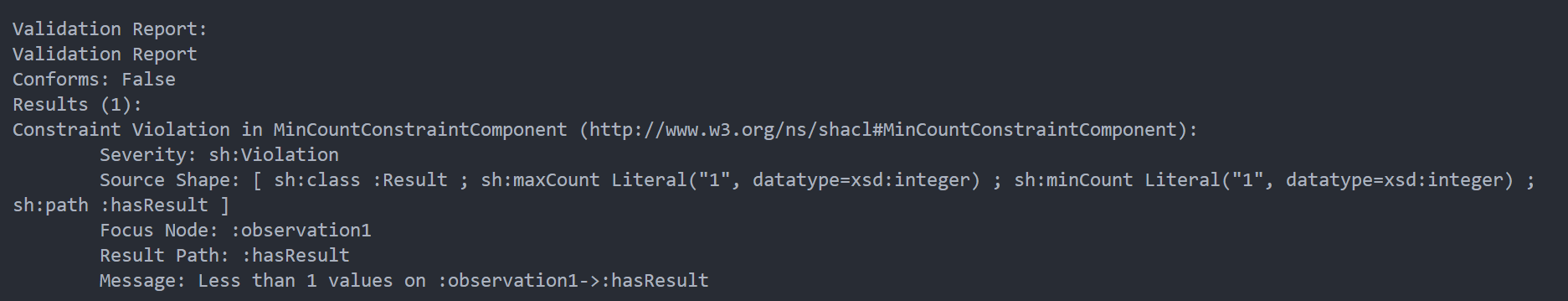


Εικόνα 46 – Validation τους οντολογίας με pyshacl



Εικόνα 47 – Αποτελέσματα του validation

Για να είμαστε απόλυτα σίγουροι ότι το validation γίνεται σωστά, παρακάτω παρουσιάζεται ένα σενάριο όπου έγινε ένα εσκεμμένο λάθος στην οντολογία για να δούμε αν θα εντοπιστεί κατά τη διαδικασία της επικύρωσης. Στο σενάριο αυτό αφαιρέθηκε η ιδιότητα hasResult από ένα στιγμιότυπο της κλάσης Observation. Το λάθος εντοπίζεται κατά τη διαδικασία του validation και τα αποτελέσματα που δίνονται φαίνονται στην εικόνα 48. Όπως φαίνεται, υπάρχει ένα λάθος στο observation1 (focus node) που αφορά την ιδιότητα hasResult (result path). Από κάτω δίνεται και η περιγραφή του λάθους η οποία λέει στην ουσία ότι δεν ικανοποιείται το κριτήριο του να υπάρχει τουλάχιστον ένα hasResult στο συγκεκριμένο στιγμιότυπο.



Εικόνα 48 – Εντοπισμός λάθους από validation

# Αποθήκευση Δεδομένων

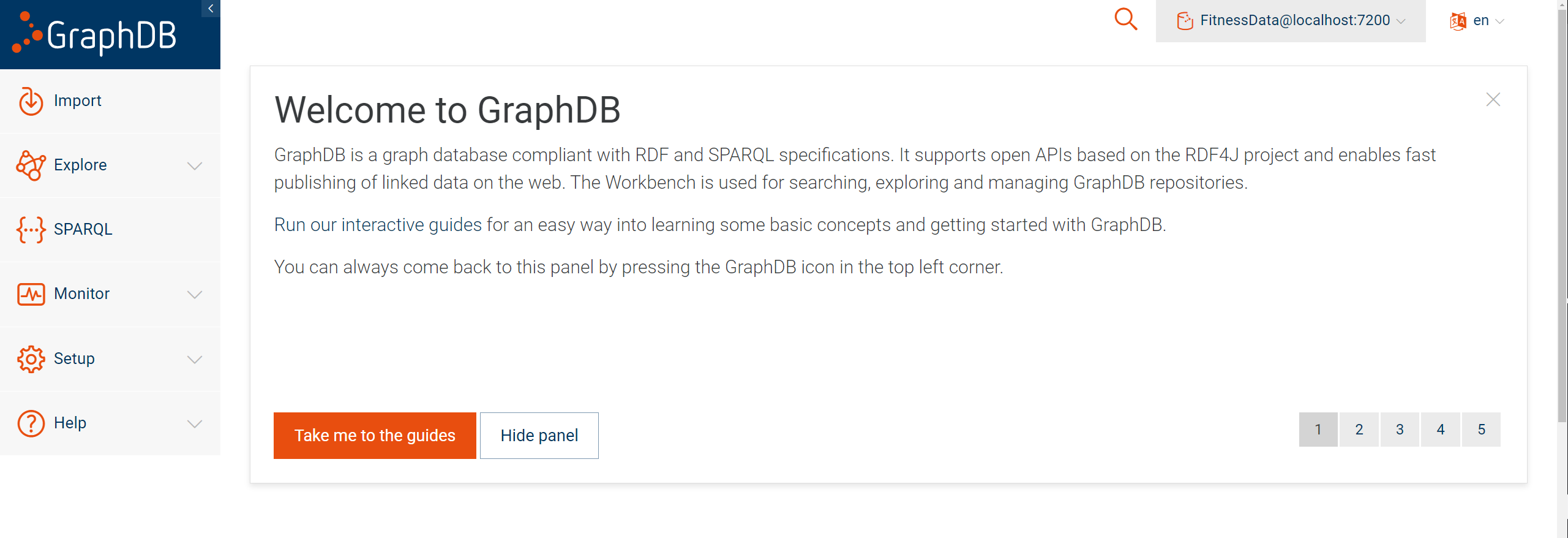
Η αποθήκευση των δεδομένων έγινε σε μία βάση δεδομένων GraphDB. Το GraphDB είναι ένα πανίσχυρο σύστημα βάσης δεδομένων γράφων που παρέχει εξαιρετική λειτουργικότητα για την αποθήκευση, την ανάκτηση και την ερώτηση πολύπλοκων δεδομένων γράφων. Είναι βασισμένη σε RDF και SPARQL και προσφέρει προηγμένες δυνατότητες για τη διαχείριση γράφων δεδομένων, την ανίχνευση συσχετίσεων και την ανάλυση της γνώσης. Με την υποστήριξη ευέλικτων σχημάτων, ανεξάρτητης από τη γλώσσα αναζήτησης και προηγμένων λειτουργιών αποθήκευσης, το GraphDB αποτελεί ένα ισχυρό εργαλείο για την ανάπτυξη και την εκτέλεση εφαρμογών που απαιτούν αποτελεσματική διαχείριση γράφων δεδομένων. Η δομή των γράφων αντιπροσωπεύει τα δεδομένα με κόμβους και ακμές, όπου οι κόμβοι αναπαριστούν οντότητες και οι ακμές αναπαριστούν τις σχέσεις μεταξύ τους. Το GraphDB προσφέρει πολλά πλεονεκτήματα, όπως ευελιξία στην αναπαράσταση πολύπλοκων σχέσεων, αποτελεσματική αναζήτηση με βάση τις σχέσεις και ανάλυση δεδομένων και ανακάλυψη γνώσης μέσω ερωτημάτων στο γράφο.

Οι κύριοι τρόποι και σενάρια χρήσης τους GraphDB περιλαμβάνουν:

* Κοινωνικά δίκτυα και συστήματα συστάσεων: Η GraphDB μπορεί να χρησιμοποιηθεί για την ανάλυση και την εξαγωγή γνώσης από κοινωνικά δίκτυα και τη δημιουργία συστάσεων για χρήστες.
* Επιστημονική έρευνα και βιοπληροφορική: Η GraphDB μπορεί να χρησιμοποιηθεί για την ανάλυση γενετικών δεδομένων, την ανακάλυψη γενετικών συνδέσεων και την ανάλυση βιολογικών δικτύων.
* Γνωσιακή βάση για εφαρμογές AI: Η GraphDB μπορεί να χρησιμοποιηθεί για την αποθήκευση και την ανάκτηση γνώσης που απαιτείται από εφαρμογές τεχνητής νοημοσύνης όπως συστήματα συστάσεων, ερωταπαντήσεων και αυτόματης κατανόησης γλώσσας.

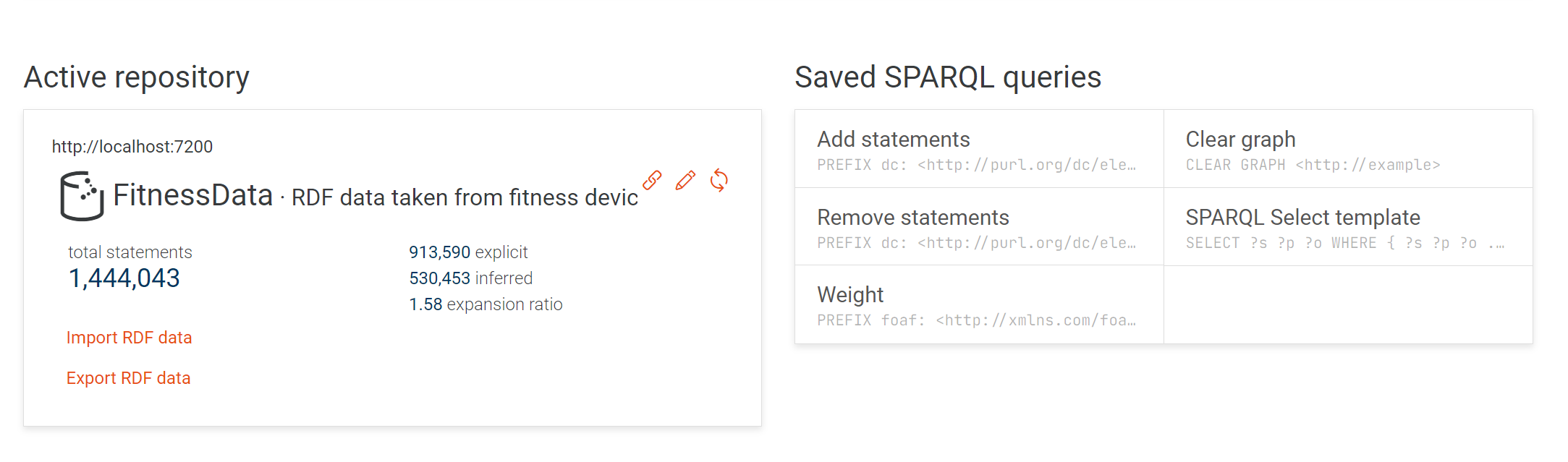
Ένα άλλο σημαντικό πλεονέκτημα του GraphDB είναι το GraphDB Workbench, το οποίο χρησιμοποιήθηκε και στα πλαίσια της εργασίας. Το GraphDB Workbench είναι ένα γραφικό περιβάλλον διαχείρισης και ανάπτυξης για το GraphDB. Αποτελεί ένα διαδικτυακό εργαλείο που παρέχει μια φιλική προς τον χρήστη διεπαφή για να εκτελεί ερωτήματα, να διαχειρίζεται γράφους, να προβάλλει τα αποτελέσματα και να επεξεργάζεται τα δεδομένα. Με το GraphDB Workbench, οι χρήστες μπορούν να εκτελούν SPARQL ερωτήματα για να ανακτήσουν δεδομένα από τον γράφο, να δημιουργούν και να επεξεργάζονται γράφους, να διαχειρίζονται οντότητες και σχέσεις, και να εκτελούν αναλύσεις γράφων. Επιπλέον, παρέχει εργαλεία για την ανακάλυψη της γνώσης που περιέχεται στο γράφο, για οπτικοποιήσεις, διαγράμματα και γεωγραφικές προβολές. Το GraphDB Workbench είναι σχεδιασμένο για να είναι εύχρηστο και να παρέχει μια ολοκληρωμένη εμπειρία διαχείρισης για τους χρήστες του GraphDB. Επιτρέπει την αποτελεσματική εξερεύνηση και ανάλυση των δεδομένων γράφου, καθιστώντας την ανάπτυξη και τη διαχείριση γράφων πιο εύκολη και αποτελεσματική διαδικασία.

Στην εικόνα 49 φαίνεται η αρχική σελίδα του GraphDB Workbench. Αρχικά, ο χρήστης βλέπει κάποιες πληροφορίες σχετικά με το GraphDB και το Workbench, του παρέχονται κάποιες πληροφορίες σχετικά με τις βασικές λειτουργίες που μπορούν να εκτελεστούν, καθώς και σύνδεσμοι προς κάποια guides που παρέχονται από τους developers. Στο αριστερά μέρος της σελίδας υπάρχουν όλοι οι σύνδεσμοι για την πλοήγηση του χρήστη προς τις υπόλοιπες σελίδες της εφαρμογής.



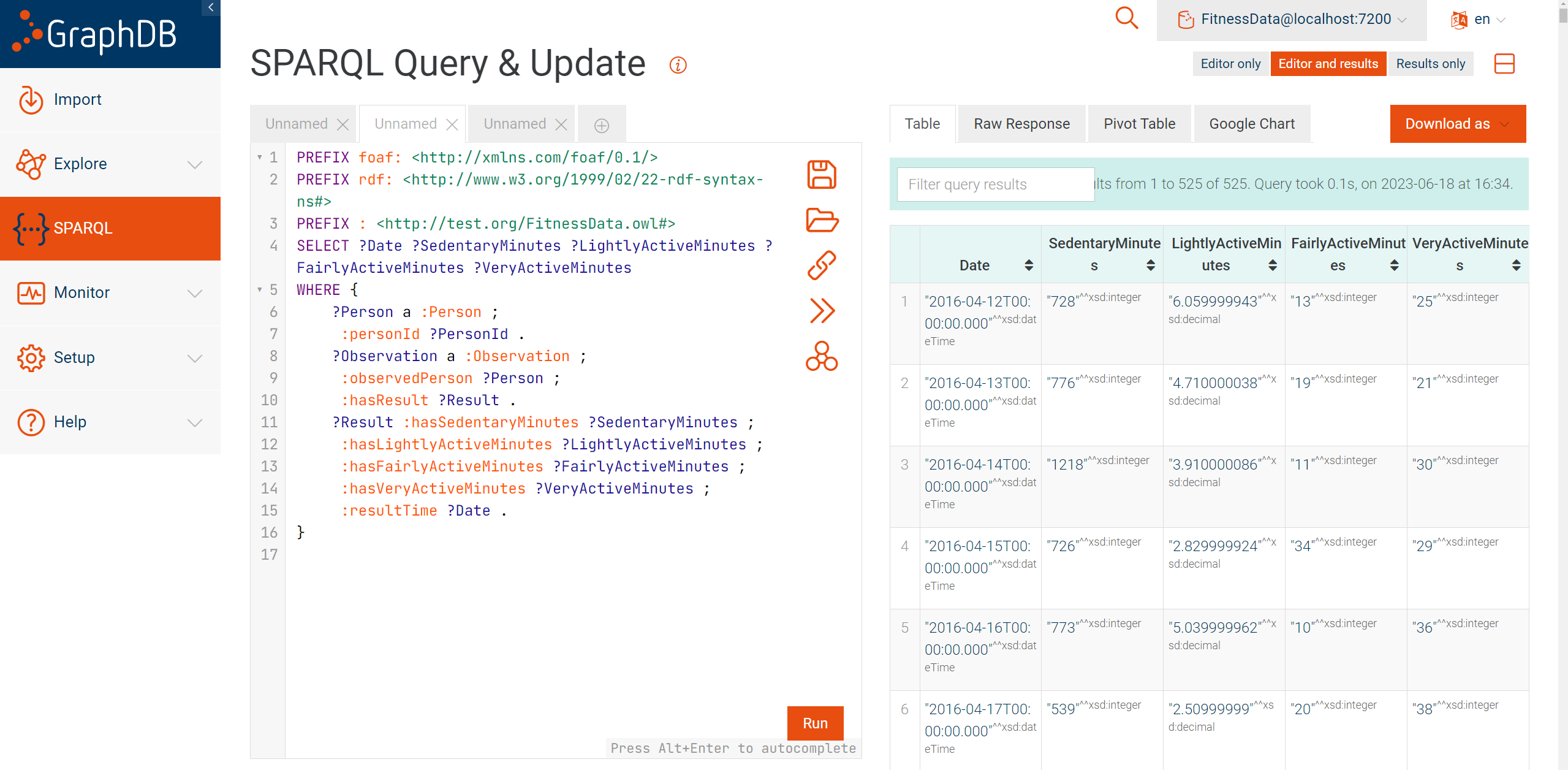
Εικόνα 49 – GraphDB Workbench home page

Επίσης, στο κάτω μέρος της σελίδας παρέχονται κάποιες πληροφορίες σχετικά με το ενεργό repository, το οποίο φαίνεται και στο dropdown menu στο πάνω δεξιά της σελίδας, μέσω του οποίου μπορεί να γίνει και εναλλαγή σε κάποιο άλλο repository. Επιπλέον, υπάρχουν και κάποια shortcuts για την εκτέλεση κάποιων αποθηκευμένων ερωτημάτων πάνω στο repository. Αυτές οι πληροφορίες φαίνονται στην εικόνα 50. Αριστερά βρίσκεται το ενεργό repository, για το οποίο βλέπουμε το όνομα του (FitnessData), την τοποθεσία στην οποία είναι ενεργός ο server (<http://localhost:7200>), τον αριθμό τριπλετών που περιέχει (total statemenets) καθώς και πόσες από αυτές είναι σαφώς ορισμένες (explicit) και πόσες είναι αποτέλεσμα συμπερασμού (inferred). Παρέχονται επίσης και σύνδεσμοι για την εισαγωγή και την εξαγωγή δεδομένων. Δεξιά βρίσκονται κάποια αποθηκευμένα ερωτήματα.



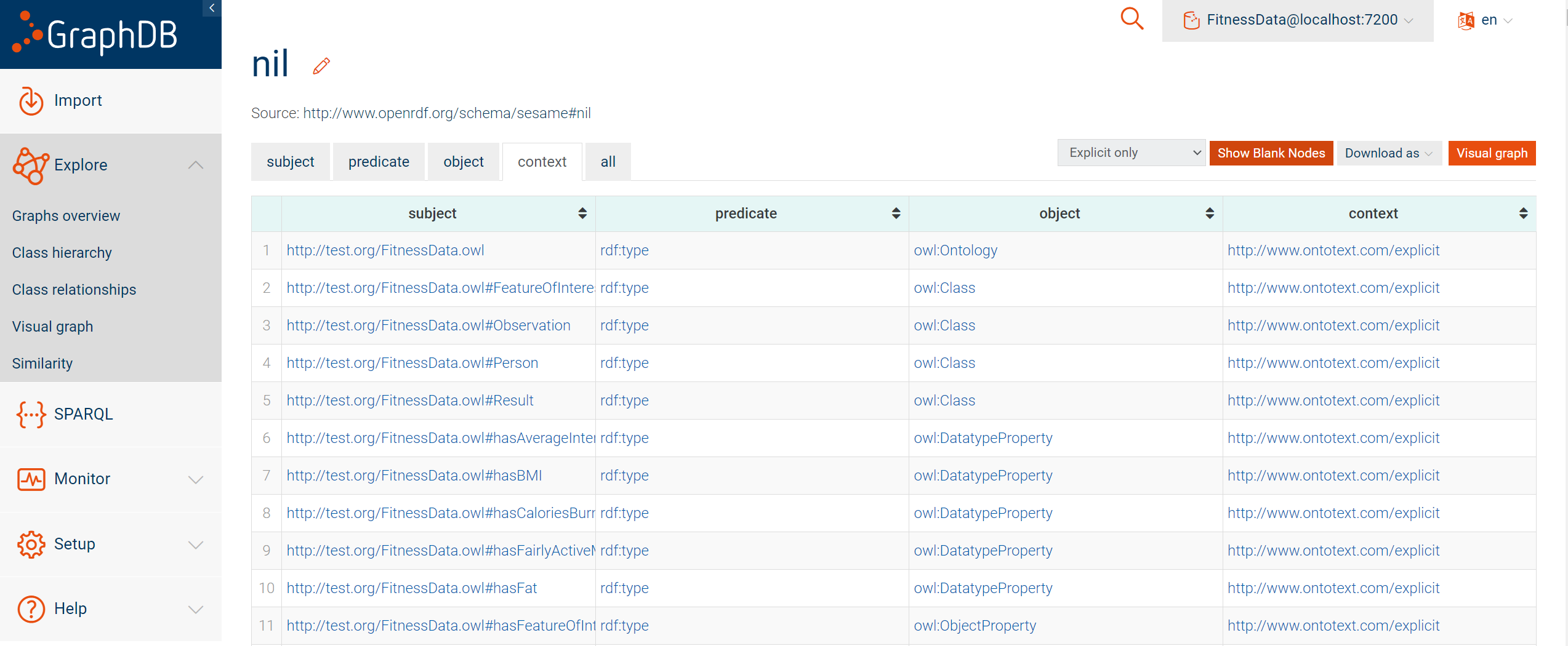
Εικόνα 50 – Πληροφορίες για το ενεργό repository

Όπως αναφέρθηκε, το GraphDB Workbench δίνει την δυνατότητα για την δημιουργία και την εκτέλεση ερωτημάτων πάνω στην βάση δεδομένων. Στην εικόνα 51 φαίνεται η σελίδα που παρέχεται για αυτές τις λειτουργίες, στην οποία ο χρήστης μπορεί να πλοηγηθεί μέσω της μπάρας στο αριστερό μέρος της σελίδας. Στο αριστερό μέρος είναι ο editor για την εκτέλεση του ερωτήματος και δεξιά φαίνονται τα αποτελέσματα του. Επίσης, δίνονται διάφορες επιλογές στον χρήστη για την αλλαγή της δομής της σελίδας μέσω των κουμπιών στο πάνω δεξιά μέρος, όπως εμφάνιση μόνο του editor ή εμφάνιση μόνο των αποτελεσμάτων.



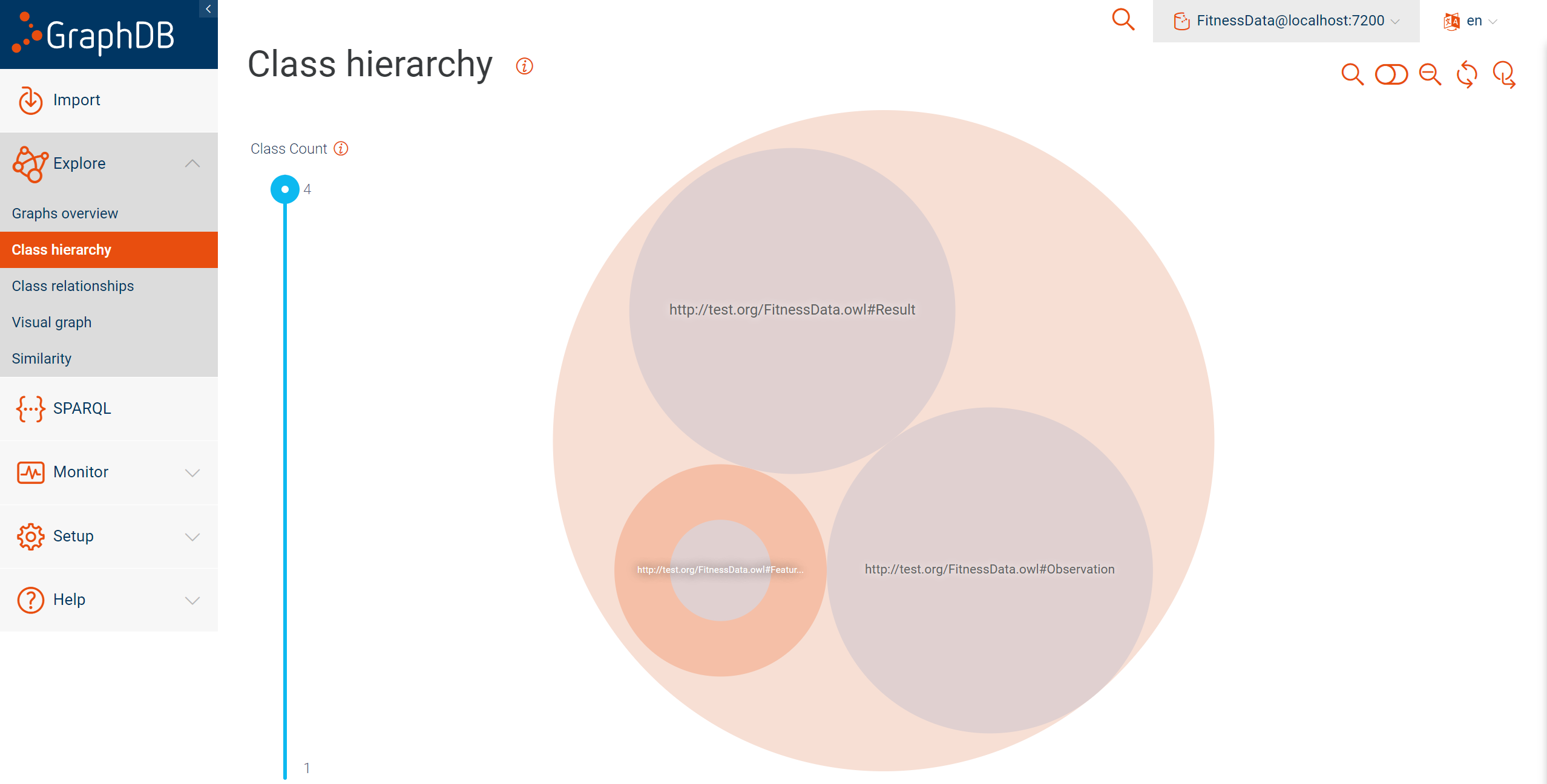
Εικόνα 51 – SPARQL page του GraphDB Workbench

Επιπλέον, το GraphDB Workbench παρέχει διάφορους τρόπους για την οπτικοποίηση του repository. Μέσω του Explore μενού ο χρήστης μπορεί να δει τα δεδομένα της βάσης του, σε μορφή πίνακα ή σε μορφή γραφημάτων, καθώς και κάποιες άλλες πληροφορίες για τα δεδομένα όπως ιεραρχία κλάσεων και αριθμός συνδέσμων μεταξύ των κλάσεων. Στην εικόνα 52 φαίνεται ένα κομμάτι από τα δεδομένα του repository σε μορφή πίνακα. Ο χρήστης μπορεί να μεταβεί σε αυτή τη σελίδα μέσω του συνδέσμου ‘Graphs overview’, επιλέγοντας το γράφημα που τον ενδιαφέρει.



Εικόνα 52 – GraphDB

Έπειτα, στην εικόνα 53 φαίνεται η ιεραρχία των κλάσεων του γράφου, η οποία είναι προσβάσιμη μέσω του συνδέσμου ‘Class hierarchy’. Ο χρήστης μπορεί να μεταβάλλει τον αριθμό των κλάσεων που θέλει να εμφανίζονται μέσω του slider στο αριστερό μέρος. Κάνοντας κλικ σε κάποια κλάση, ο χρήστης μπορεί να δει τα στιγμιότυπα της. Όπως φαίνεται λοιπόν, οι δύο πιο σημαντικές κλάσεις της οντολογίας είναι οι κλάσεις Observation και Result. Επίσης, η κλάση Person βρίσκεται μέσα στη κλάση FeatureOfInterest καθώς αποτελεί υποκλάση της.



# Dashboards και παραδείγματα ερωτημάτων

Για την οπτικοποίηση των αποτελεσμάτων και την δυνατότητα εξαγωγής συμπερασμάτων από αυτά, δημιουργήθηκε μία web εφαρμογή μέσω της οποίας ο χρήστης μπορεί να διαλέξει την κατηγορία των δεδομένων που θέλει να δει, για τον άνθρωπο που τον ενδιαφέρει, και να παρατηρήσει την εξέλιξη που είχε μέσα σε μία συγκεκριμένη χρονική περίοδο. Η εφαρμογή συνδέεται με τη βάση δεδομένων στην οποία αποθηκεύτηκαν τα δεδομένα και στέλνει σε αυτήν κάποια προκαθορισμένα ερωτήματα, ανάλογα με τις επιλογές του χρήστη και το τι θέλει να δει.

Για το backend της εφαρμογής χρησιμοποιήθηκαν οι τεχνολογίες Node.js και Express.js, που αποτελούν εργαλεία βασισμένα σε JavaScript. Το Node.js αποτελεί ένα JavaScript runtime environment, που επιτρέπει την εκτέλεση JavaScript κώδικα εκτός του περιβάλλοντος κάποιου περιηγητή (browser). Το Express.js είναι ένα ελαφρύ πλαίσιο εφαρμογής ιστού (web framework) για το Node.js. Παρέχει ένα απλό και ευέλικτο σύνολο χαρακτηριστικών για την κατασκευή ιστοσελίδων και API. Το Express.js βοηθά στην χειρισμό των αιτημάτων HTTP, τη δρομολόγηση, τη μεσαία λογική (middleware) και τις κοινές εργασίες ανάπτυξης ιστού. Με τη χρήση αυτών των τεχνολογιών, δημιουργείται ένας server που τρέχει τοπικά στον υπολογιστή του χρήστη, γίνεται η σύνδεση με τη βάση δεδομένων, και στέλνονται τα ερωτήματα και λαμβάνονται οι απαντήσεις από τη βάση. Στην εικόνα 53 φαίνεται ο κώδικας του αρχείου server.js, στο οποίο χρησιμοποιείται η Express.js για την ρύθμιση του middleware, των API routes, και για την δημιουργία του server.

A screen shot of a computer program

Description automatically generated with low confidence

Εικόνα 53 – Δημιουργία του server με Express

Για το frontend της εφαρμογής, χρησιμοποιήθηκε η Angular. Η Angular είναι ένα web framework ανοιχτού κώδικα που χρησιμοποιείται για την δημιουργία σύγχρονων εφαρμογών ιστού[[13]](#footnote-13). Είναι βασισμένη στην γλώσσα προγραμματισμού TypeScript και ακολουθεί την αρχιτεκτονική του πελάτη-εξυπηρετητή (client-server). Η Angular προσφέρει ένα πλήρες σύνολο εργαλείων και λειτουργιών που καλύπτουν την ανάπτυξη, τον δομημένο προγραμματισμό, την διαχείριση κατάστασης, τη δρομολόγηση, την ανάλυση και τις διάφορες πτυχές της ανάπτυξης εφαρμογών. Επιπλέον, παρέχει έναν ισχυρό μηχανισμό δέσμευσης δεδομένων και μια ευέλικτη δομή για την ανάπτυξη προσαρμόσιμων και επαναχρησιμοποιήσιμων συνιστωσών. Με την χρήση της Angular, οι προγραμματιστές μπορούν να αναπτύξουν ευέλικτες, αποδοτικές και ποιοτικές εφαρμογές ιστού που λειτουργούν χωρίς προβλήματα σε διάφορες πλατφόρμες και συσκευές. Επιπλέον, η μεγάλη κοινότητα της Angular παρέχει υποστήριξη, πόρους και επεκτατικές δυνατότητες, καθιστώντας το ένα ισχυρό εργαλείο για την ανάπτυξη προηγμένων εφαρμογών ιστού.

Η εφαρμογή που κατασκευάστηκε αποτελείται από δύο σελίδες, μία αρχική σελίδα (about page) η οποία δίνει κάποιες βασικές πληροφορίες σχετικά με το dataset καθώς και την δυνατότητα λήψης του, και μία σελίδα στην οποία γίνεται η οπτικοποίηση των δεδομένων, με χρήση πινάκων και γραφημάτων, και στην οποία ο χρήστης μπορεί να δει τα δεδομένα που τον ενδιαφέρουν (dashboards page). Με το που ανοίγει ο χρήστης την εφαρμογή, βρίσκεται στην αρχική σελίδα, η οποία φαίνεται στην εικόνα 54. Σε αυτή τη σελίδα ο χρήστης μπορεί να κάνει λήψη του dataset πατώντας το κουμπί “Download”, να δει κάποιες βασικές πληροφορίες που αφορούν το dataset, και να μεταβεί στη σελίδα dashboards, είτε μέσω του κουμπιού “View Dashboards” στο κάτω μέρος της σελίδας, είτε από τη μπάρα πλοήγησης στη κορυφή της σελίδας.

A screenshot of a fitness tracker

Description automatically generated with medium confidence

Εικόνα 54 – Αρχική σελίδα τους εφαρμογής

Η αρχική όψη της σελίδας dashboards, προτού εκτελέσει κάποια ενέργεια ο χρήστης φαίνεται στην εικόνα 55. Ο χρήστης επιλέγει τον άνθρωπο και την κατηγορία δεδομένων που τον ενδιαφέρει, καθώς και την επιθυμητή χρονική περίοδο, μέσω των τριών μενού που φαίνονται. Όταν πατήσει το submit κουμπί, στέλνεται ένα SPARQL ερώτημα στην βάση δεδομένων μέσω του backend, και λαμβάνεται η απάντηση, η οποία παρουσιάζεται στον χρήστη μέσω πινάκων και γραφημάτων.

A screenshot of a computer

Description automatically generated with medium confidence

Εικόνα 55 – Αρχική όψη τους σελίδας Dashboards

Τα ερωτήματα που στέλνονται στη βάση είναι προκαθορισμένα και η επιλογή του σωστού ερωτήματος για αποστολή γίνεται βάση των επιλογών του χρήστη. Για παράδειγμα, αν ο χρήστης επέλεξε την κατηγορία Activity, για τον person1, στη χρονική περίοδο 4/1/2016 – 4/14/2016, το ερώτημα που εκτελείται είναι αυτό που φαίνεται στην εικόνα 56. Μέσα στον κώδικα, υπάρχει ένα προκαθορισμένο ερώτημα για κάθε επιλεγμένη κατηγορία, στο οποίο ενσωματώνονται το id του ανθρώπου που επιλέγει ο χρήστης καθώς και οι ημερομηνίες μέσω μεταβλητών. Έτσι, το ερώτημα τελικά διαμορφώνεται δυναμικά ανάλογα με το τι θέλει να δει ο χρήστης.

A screen shot of a computer code

Description automatically generated with medium confidence

Εικόνα 56 – SPARQL query που στέλνεται στην βάση

Αφού εκτελείται το ερώτημα και λαμβάνονται τα αποτελέσματα, γίνεται η παρουσίαση τους στην σελίδα dashboards. Αρχικά, εμφανίζεται ένας πίνακας που περιλαμβάνει τα αποτελέσματα του ερωτήματος, όπως φαίνεται στην εικόνα 57. Ο χρήστης μπορεί να περιηγηθεί στον πίνακα καθώς και να μεταβάλλει τον αριθμό των γραμμών ανά σελίδα μέσω του paginator στο κάτω μέρος του πίνακα, ανάλογα με τον αριθμό των αποτελεσμάτων που επιστρέφει το κάθε ερώτημα.

A screenshot of a computer

Description automatically generated with medium confidence

Εικόνα 57 – Πίνακας με τα αποτελέσματα ερωτήματος

Για την καλύτερη οπτικοποίηση των αποτελεσμάτων και για ευκολότερη παρατήρηση της εξέλιξης των τιμών, δημιουργούνται και γραφήματα. Κάθε γράφημα έχει στον οριζόντιο άξονα την ημέρα παρατήρησης, δηλαδή την στήλη Date του πίνακα, και στον κατακόρυφο την τιμή μίας από τους υπόλοιπες στήλες. Για παράδειγμα, για αυτό το ερώτημα θα δημιουργηθούν 4 γραφήματα, ένα για κάθε στήλη που περιλαμβάνεται στην απάντηση, εκτός της Date. Παραδείγματα γραφημάτων για τις τιμές των SedentaryMinutes και LightlyActiveMinutes φαίνονται στις εικόνες 58 και 59. Τα γραφήματα είναι τύπου γραμμής, η οποία δείχνει την μεταβολή της τιμής της αντίστοιχης μεταβλητής δια της επιλεγμένης χρονικής περιόδου. Στην περίπτωση που δεν υπάρχουν δεδομένα για το ερώτημα του χρήστη, παρουσιάζεται ένα ενημερωτικό μήνυμα στην θέση που θα εμφανιζόταν ο πίνακας.

A picture containing text, screenshot, line, diagram

Description automatically generated

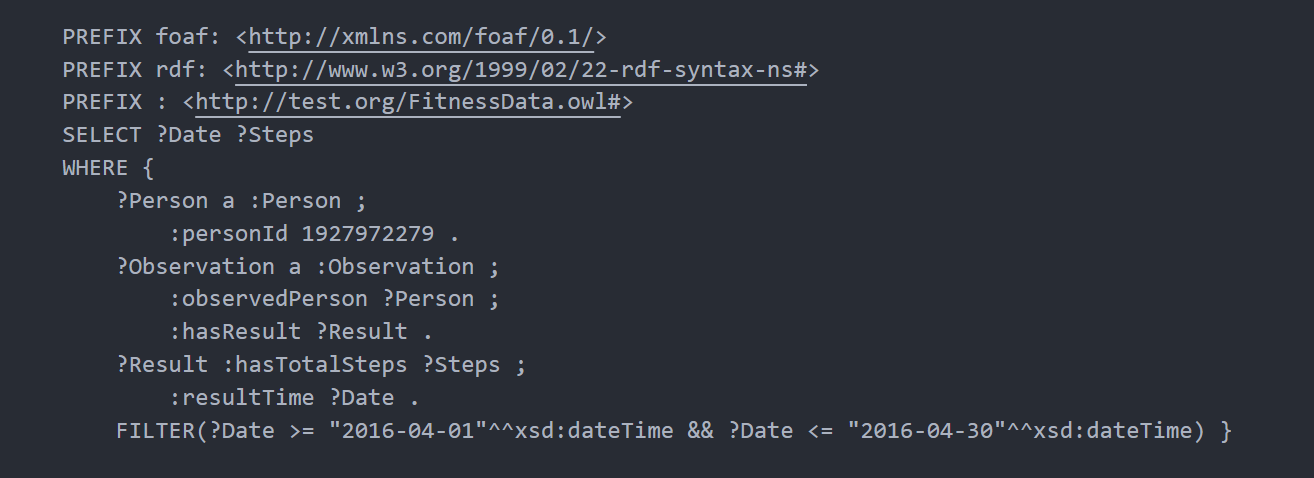
Εικόνα 58 – Γράφημα για SedentaryMinutes

A picture containing text, line, screenshot, plot

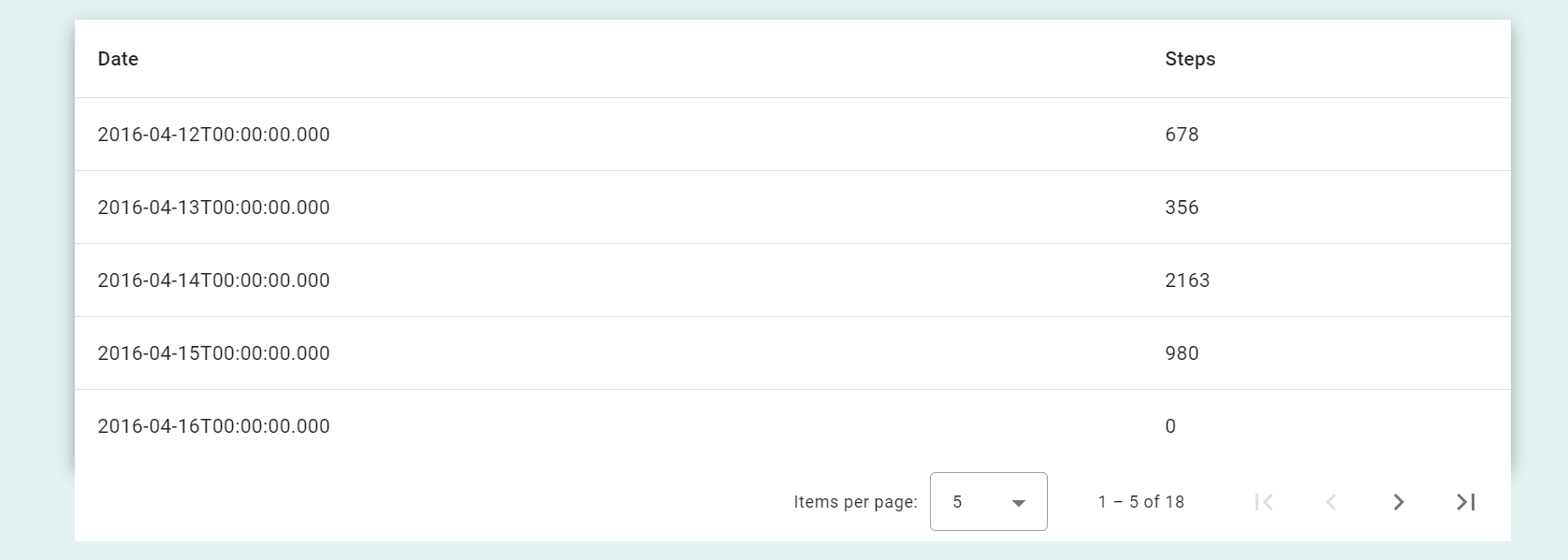
Description automatically generated

Εικόνα 59 – Γράφημα για LightlyActiveMinutes

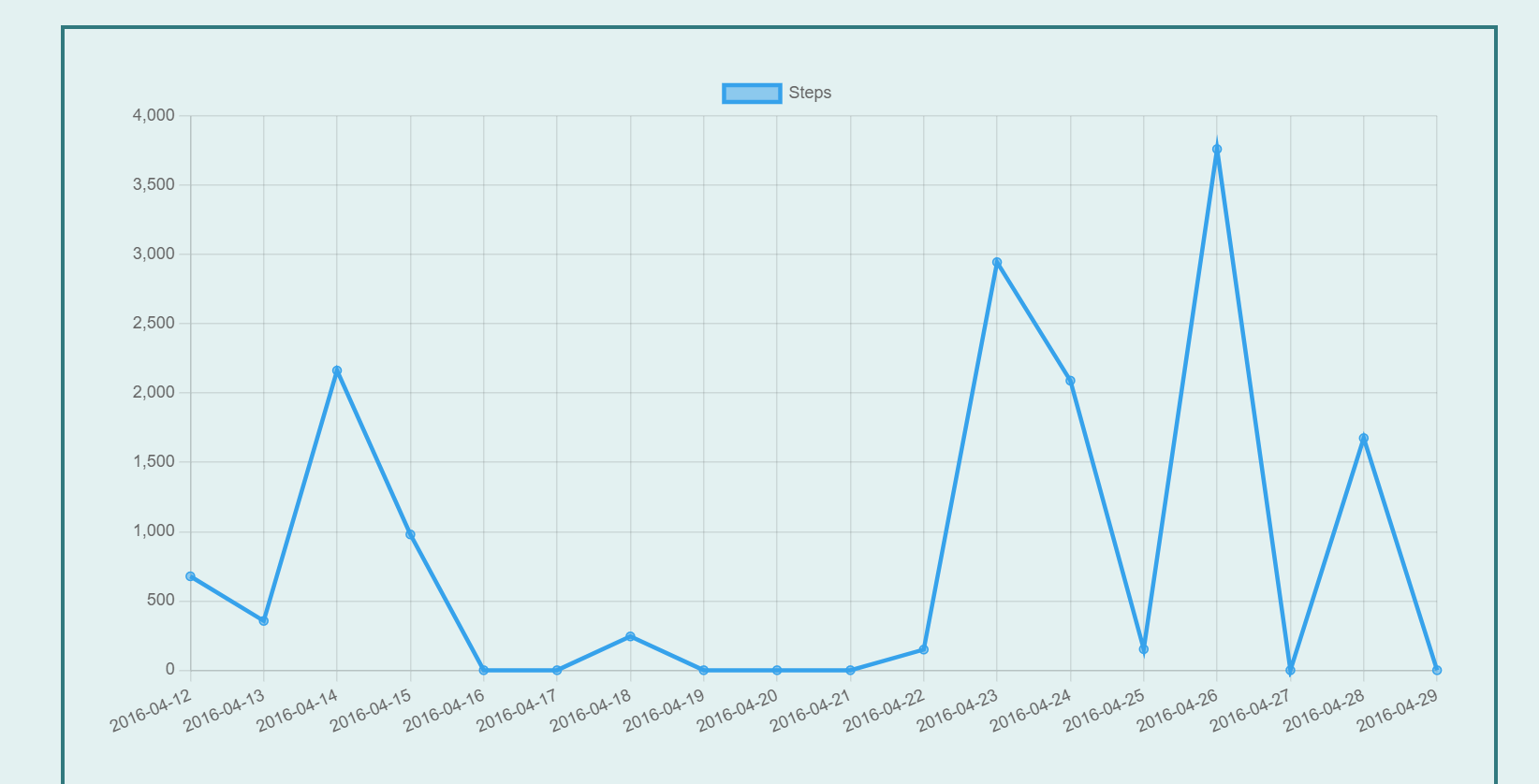
Ακολουθεί η παρουσίαση ενός ακόμα παραδείγματος. Έστω ότι ο χρήστης επιλέγει από την λίστα των ανθρώπων τον person2, από τις κατηγορίες το steps και ημερομηνίες από 4/1/2016 μέχρι 4/30/2016. Το ερώτημα που εκτελείται φαίνεται στην εικόνα 60, και τα αποτελέσματα που βλέπει ο χρήστης φαίνονται στις εικόνες 61 και 62. Επιστρέφονται 18 εγγραφές, από τις οποίες αρχικά στον πίνακα φαίνονται οι 5. Ο χρήστης μπορεί να μεταβεί στις υπόλοιπες μέσω του paginator στο κάτω μέρος του πίνακα.



Εικόνα 60 – SPARQL ερώτημα που εκτελείται



Εικόνα 61 – Πίνακας αποτελεσμάτων



Εικόνα 62 – Γράφημα αποτελεσμάτων

1. Συμπεράσματα Και Μελλοντικές Εργασίες

Συμπεράσματα – Προτάσεις για Μελλοντικη Εργασια

# Συμπεράσματα και Προβληματισμοί

Σκοπός αυτής της πτυχιακής εργασίας ήταν η δημιουργία εργαλείων που θα μπορούσαν να λάβουν δεδομένα που προέρχονται από fitness trackers, να τα μετασχηματίσουν, να τα αποθηκεύσουν σε έναν RDF γράφο, και στη συνέχεια να τα αναπαραστήσουν στον χρήστη με κατανοητό τρόπο, δίνοντας του έτσι την δυνατότητα να βγάλει τα συμπεράσματα του από αυτά. Κύριος στόχος λοιπόν των εργαλείων είναι να αποτελέσουν ένα μέσο οπτικοποίησης των δεδομένων για τον χρήστη, επιτρέποντας του να παρατηρήσει τις μεταβολές που έγιναν στα δεδομένα ανά την χρονική περίοδο παρατήρησης τους, χρησιμοποιώντας τεχνολογίες του Σημασιολογικού Ιστού.

Στο πλαίσιο εκπόνησης της εργασίας κρίθηκε απαραίτητη η μελέτη διάφορων όρων και εργαλείων, καθώς οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση όλων των εργαλείων της εργασίας δεν είναι ιδιαίτερα γνωστές. Αρχικά, έπρεπε να γίνει μια γενικότερη μελέτη για τον Σημασιολογικό Ιστό, τι είναι, ποια προβλήματα προσπαθεί να διορθώσει, ποιες τεχνολογίες χρησιμοποιεί, ώστε να μπορέσουν και όλες οι έννοιες που περιέχονται στην εργασία να παρουσιαστούν με κατανοητό τρόπο. Επίσης, έγινε μία σύντομη έρευνα σχετικά με τις συσκευές παρακολούθησης δραστηριότητας ή fitness trackers, καθώς αποτελούν ένα σημαντικό κομμάτι της εργασίας, εστιάζοντας περισσότερο στο πως λειτουργούν, γιατί χρησιμοποιούνται, τις λειτουργίες και δυνατότητες τους, και τους διάφορους τύπους τους.

Έπειτα, όσον αφορά το κομμάτι της υλοποίησης, έπρεπε να γίνει μελέτη κάποιων βασικών τεχνολογιών που χρησιμοποιούνται στον Σημασιολογικό Ιστό, όπως οι γλώσσες RDF, RDFS και OWL, που χρησιμοποιήθηκαν για την δημιουργία της οντολογίας, η γλώσσα SPARQL που χρησιμοποιείται για την εκτέλεση ερωτημάτων, καθώς και οι σημασιολογικές βάσεις δεδομένων, όπως είναι η GraphDB, που χρησιμοποιούνται για την αποθήκευση των δεδομένων RDF. Επίσης, έγινε μελέτη και των οντολογιών SSN και SOSA, πάνω στις οποίες βασίστηκε η οντολογία της εργασίας, καθώς και στη γλώσσα επικύρωσης RDF γράφων, SHACL, η οποία χρησιμοποιήθηκε για την επικύρωση του παραγόμενου γράφου.

Η μελέτη και η εκμάθηση αυτών των τεχνολογιών δεν ήταν ιδιαίτερα δύσκολη και δεν αντιμετώπισα κάποιο σοβαρό εμπόδιο, καθώς το documentation που υπάρχει στο διαδίκτυο είναι αρκετά καλό, και επίσης με βοήθησε και το αντίστοιχο μάθημα που παρακολούθησα στη σχολή μου, το μάθημα «Σημασιολογικός Ιστός και Ευφυείς Εφαρμογές», μέσω του οποίου ήρθα πρώτη φορά σε επαφή με την έννοια του Σημασιολογικού Ιστού και ξεκίνησα την ενασχόληση μου με κάποιες από αυτές τις τεχνολογίες. Ένα γενικότερο πρόβλημα που αντιμετώπισα όμως κατά την διάρκεια της υλοποίησης ήταν η διόρθωση σφαλμάτων. Ο λόγος που είχα αυτό το πρόβλημα είναι ότι δεν υπάρχει τόσο μεγάλο community για αυτές τις τεχνολογίες όπως για παράδειγμα θα υπήρχε για άλλες γλώσσες προγραμματισμού, όπως Java. Έτσι, υπήρχαν κάποια σημεία στα οποία ίσως να είχα κολλήσει λιγάκι παραπάνω, όπως όταν έφτιαχνα τους κανόνες επικύρωσης του RDF γράφου, χρησιμοποιώντας την SHACL, καθώς δεν υπήρχαν πολλά παραδείγματα στο διαδίκτυο, ή αντίστοιχες ερωτήσεις από τους χρήστες σε σελίδες όπως stackoverflow, οι οποίες συνήθως αποτελούν μεγάλη βοήθεια.

Το τελευταίο κομμάτι της εργασίας ήταν η δημιουργίας της web εφαρμογής για την οπτικοποίηση των δεδομένων, όπου χρησιμοποιήθηκαν διάφορες τεχνολογίες όπως Angular για το frontend, και Node.js και Express.js για το backend και την επικοινωνία με την βάση. Μια δυσκολία που αντιμετώπισα σε αυτό το κομμάτι ήταν στα πρώτα στάδια, στο θέμα της επιλογής των τεχνολογιών που θα χρησιμοποιούσα. Αυτό που με βοήθησε στην επιλογή τους Angular ήταν το γεγονός ότι έκανα την πρακτική μου άσκηση την ίδια περίοδο με την εκπόνηση της πτυχιακής, κατά την οποία ασχολούμουν με την Angular, οπότε υπήρχε ήδη μία εξοικείωση.

Συμπερασματικά, η διαδικασία εκπόνησης της πτυχιακής κύλησε ομαλά χωρίς να υπάρξει κάποιο σημαντικό πρόβλημα, και όλα τα παραγόμενα εργαλεία έχουν την επιθυμητή λειτουργία και εκτελούν σωστά τον σκοπό τους. Παρόλα αυτά όμως, υπάρχουν σημεία τα οποία επιδέχονται βελτιώσεις.

# Προτασεις για Μελλοντικη Εργασια

Παρόλο που η εφαρμογή έχει την επιθυμητή λειτουργία χωρίς να παρουσιάζει προβλήματα, υπάρχουν κάποια σημεία τα οποία μπορούν να βελτιωθούν. Αρχικά, θα μπορούσε να προστεθεί περισσότερη πληροφορία στην οντολογία. Δεν έγινε χρήση όλων των δεδομένων που υπήρχαν στο αρχικό dataset, καθώς δεν απαιτούταν τόση πολλή πληροφορία στα πλαίσια της εργασίας και θα γινόταν δυσκολότερη η διαδικασία ελέγχου και συντήρησης της. Οπότε, υπάρχουν ακόμα αρκετά δεδομένα τα οποία δεν χρησιμοποιούνται και θα μπορούσαν να προστεθούν στην οντολογία, ώστε να παρέχει περισσότερη πληροφορία. Επίσης, ένα άλλο τμήμα που έχει περιθώρια βελτίωσης είναι η web εφαρμογή για την οπτικοποίηση των δεδομένων. Στην εφαρμογή θα μπορούσαν να προστεθούν περισσότερες δυνατότητες, όπως δυνατότητα προβολής της μεταβολής των δεδομένων ανά ώρα, για τα δεδομένα όπου υπάρχει αυτή η πληροφορία στην οντολογία, δυνατότητα επιλογής και προβολής περισσότερων και διαφορετικού τύπου γραφημάτων, δυνατότητα προβολής στατιστικών για την μεταβολή των δεδομένων που θα υπολογίζονται λαμβάνοντας υπόψιν τα δεδομένα όλων των ανθρώπων, καθώς ίσως και τη δυνατότητα εκτέλεσης SPARQL ερωτημάτων άμεσα από τον χρήστη πάνω στη βάση, αν ενδιαφέρεται για κάτι πολύ συγκεκριμένο που δεν θα παρέχεται άμεσα από την εφαρμογή. Η εφαρμογή όπως είναι τώρα είναι πολύ απλή, καθώς δεν απαιτούταν κάτι περισσότερο και πιο περίπλοκο στα πλαίσια της εργασίας, αλλά υπάρχουν πολλές περισσότερες δυνατότητες που θα μπορούσαν να προστεθούν σε αυτήν.

1. Αναφορές

Βιβλιογραφία

[1] The Internet Explained, Vincent Zegna & Mike Pepper, Sonet Digital, November 2005, pp. 1–7.

[2] McPherson, Stephanie Sammartino (2009). Tim Berners-Lee: Inventor of the World Wide Web

[3] National Information Standards Organization (NISO) (2001). Understanding Metadata (PDF)

[4] Βerners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. A New Form of Web Content That Is Meaningful to Computers Will Unleash a Revolution of New Possibilities. Scientific American, 284, 1-5.

[5] Grigoris Antoniou, Paul Growth, Frank van Harmelen, Rinke Hoekstra. A Semantic Web Primer, 3rd Edition.

[6] Harary, F. “Labeled Graphs.” Graph Theory. Reading, MA: Addison-Wesley, pp. 10 and 178-180, 1994.

[7] Bray, Tim; Hollander, Dave; Layman, Andrew; Tobin, Richard, eds. (2006-08-16). “Namespaces in XML 1.1 (Second Edition)”. World Wide Web Consortium. 2.2 Use of URIs as Namespace Names.

[8] Linked Data: Evolving the Web into a Global Data Space (2011) by Tom Heath and Christian Bizer, Synthesis Lectures on the Semantic Web: Theory and Technology, Morgan & Claypool

[9] Dean Allemang, Jim Hendler. Ο Σημασιολογικός Ιστός για τους Δημιουργούς Οντολογιών.

[10] Haller, Armin; Janowicz, Krzysztof; Cox, Simon J.D.; Lefrançois, Maxime; Taylor, Kerry; Le Phuoc, Danh; Lieberman, Joshua; García-Castro, Raúl; Atkinson, Rob; Stadler, Claus (2018). “The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation”

[11] McCreary, Dan; Kelly, Ann (2013). Making Sense of NoSQL: A guide for managers and the rest of us.

[12] Codd, E. F. (1970). “A Relational Model of Data for Large Shared Data Banks”. Communications of the ACM. 13 (6): 377–387.

[13] Geoffrey James (1985). Document Databases.

[14] Acharjya, D.P.; Geetha, M.K., eds. (2017). Internet of Things: Novel Advances and Envisioned Applications. Springer. P. 311.

1. Ακρώνυμα

Ελληνικοί Οροι

|  |  |
| --- | --- |
| **Ακρώνυμο** | **Επεξήγηση** |
| ΣΙ | Σημασιολογικός Ιστός |
| ΣΔ | Σημασιολογικό Δίκτυο |
| ΔΔ | Διασυνδεδεμένα Δεδομένα |
| ΑΔΔ | Ανοικτά Διασυνδεδεμένα Δεδομένα |

Αγγλικοί Οροι

| **Ακρώνυμο** | **Επεξήγηση** |
| --- | --- |
| RDF | Resource Description Framework |
| RDFS | Resource Description Framework – Schema |
| OWL | Web Ontology Language |
| XML | eXtensible Markup Language |
| URI | Uniform Resource Identifier |
| HTTP | HyperText Transfer Protocol |
| LPG | Labeled Property Graph |
| GDB | Graph Databases |
| SSN | Semantic Sensor Network |
| SOSA | Sensor, Observation, Sample, and Actuator |
| IoT | Internet of Things |

1. Οδηγίες Εκτέλεσησ

**Προαπαιτήσεις**

Πριν εκτελέσετε το πρόγραμμα, βεβαιωθείτε ότι έχετε εγκατεστημένες τις εξής προαπαιτήσεις:

* Python: Η Python απαιτείται για την εκτέλεση των Python scripts σε αυτό το αποθετήριο. Μπορείτε να κατεβάσετε την τελευταία έκδοση της Python από την επίσημη ιστοσελίδα της Python: [python.org](https://www.python.org/).
* Node.js: Το Node.js απαιτείται για την εκτέλεση του Angular project που περιλαμβάνεται σε αυτό το αποθετήριο. Μπορείτε να κατεβάσετε και να εγκαταστήσετε το Node.js από την επίσημη ιστοσελίδα του Node.js: [nodejs.org](https://nodejs.org/en).
* GraphDB: Το GraphDB είναι μια προϋπόθεση για αυτό το έργο. Ακολουθήστε τα παρακάτω βήματα για να εγκαταστήσετε το GraphDB:
  + Κατεβάστε το GraphDB από την επίσημη ιστοσελίδα της Ontotext: [graphdb.ontotext.com](https://graphdb.ontotext.com/)
  + Επιλέξτε το κατάλληλο πακέτο εγκατάστασης για το λειτουργικό σύστημά σας και ακολουθήστε τις οδηγίες εγκατάστασης που παρέχονται στην τεκμηρίωση του GraphDB.

**Εγκατάσταση**

Για να εγκαταστήσετε τις απαιτούμενες εξαρτήσεις για αυτό το έργο, ακολουθήστε αυτά τα βήματα:

* Κλωνοποιήστε το αποθετήριο στον υπολογιστή σας χρησιμοποιώντας την ακόλουθη εντολή: **git clone https://github.com/AndreasNbt/FitnessData.git**
* Στη συνέχεια, πλοηγηθείτε στον φάκελο του αποθετηρίου: **cd directory**
* Εγκαταστήστε όλες τις απαιτούμενες εξαρτήσεις για τα python scripts:
  + **pip install pandas**
  + **pip install pyshacl**
  + **pip install owlready2**
* Εγκαταστήστε τις εξαρτήσεις του Node.js εκτελώντας την ακόλουθη εντολή: **npm install**

**Οδηγίες εκτέλεσης**

* Για να εκτελέσετε το πρόγραμμα που διαβάζει και μετασχηματίζει τα δεδομένα και δημιουργεί την οντολογία, εκτελέστε το script main.py με την ακόλουθη εντολή: **python main.py**
  + υπάρχει ήδη ένα αρχείο Data.owl στο αποθετήριο που περιέχει την έξοδο του main.py, οπότε δεν χρειάζεται να το εκτελέσετε εκτός αν θέλετε να κάνετε κάποια αλλαγή ή δοκιμή.
* Για να επικυρώσετε την οντολογία, εκτελέστε το script validation.py με την ακόλουθη εντολή: **python validation.py**
* Στη συνέχεια, πρέπει να δημιουργήσετε ένα τοπικό αποθετήριο GraphDB και να αποθηκεύσετε το αρχείο Data.owl. Για να το κάνετε αυτό, ανοίξτε το GraphDB Workbench ανοίγοντας την εφαρμογή GraphDB Desktop που έχετε εγκαταστήσει προηγουμένως.
* Στη συνέχεια, μεταβείτε στη σελίδα Import και προσθέστε το αρχείο Data.owl που υπάρχει στο αποθετήριο κάνοντας κλικ στο κουμπί "Upload RDF files" κοντά στο πάνω μέρος της σελίδας. Όταν το αρχείο εμφανιστεί παρακάτω, κάντε κλικ στο κουμπί Import στα δεξιά του.
* Τέλος, για να εκτελέσετε την web εφαρμογή, πρώτα μεταβείτε στον φάκελο Dashboards: **cd ./Dashboards**
* Πρώτα, πρέπει να εκτελέσετε τον τοπικό διακομιστή που θα στέλνει αιτήματα στη βάση δεδομένων. Για να το κάνετε αυτό, μεταβείτε στον φάκελο backend με την ακόλουθη εντολή: **cd ./src/backend**. Στη συνέχεια ξεκινήστε τον διακομιστή με την εντολή: **node server.js**
* Έπειτα, ανοίξτε ένα νέο τερματικό, μεταβείτε στον φάκελο Dashboards και ξεκινήστε το Angular project με την εντολή: **ng serve.** Αυτό θα εκτελέσει το project και θα ανοίξει ένα παράθυρο περιήγησης που θα εμφανίζει την αρχική σελίδα.

1. <https://www.w3.org/about/> [↑](#footnote-ref-1)
2. <https://www.w3.org/standards/xml/> [↑](#footnote-ref-2)
3. <https://www.w3.org/TR/turtle/> [↑](#footnote-ref-3)
4. <https://www.w3.org/TR/PR-rdf-syntax/> [↑](#footnote-ref-4)
5. <https://www.w3.org/TR/rdf-schema/> [↑](#footnote-ref-5)
6. <https://www.w3.org/TR/owl-features/> [↑](#footnote-ref-6)
7. <https://en.wikipedia.org/wiki/Linked_data> [↑](#footnote-ref-7)
8. <https://docs.stardog.com/tutorials/rdf-graph-data-model> [↑](#footnote-ref-8)
9. <https://dbpedia.org/sparql> [↑](#footnote-ref-9)
10. <https://owlready2.readthedocs.io/en/latest/> [↑](#footnote-ref-10)
11. <https://pandas.pydata.org/docs/> [↑](#footnote-ref-11)
12. <https://www.w3.odrg/TR/shacl/> [↑](#footnote-ref-12)
13. <https://en.wikipedia.org/wiki/Single-page_application> [↑](#footnote-ref-13)