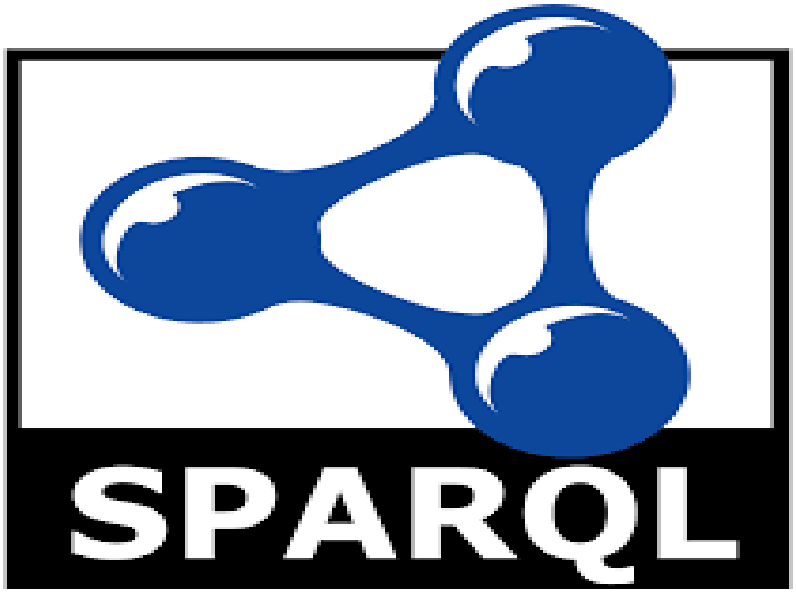
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

<<οπτικοποίηση Sparql βιβλιοθήκης σε Javascript>>



Του φοιτητή
Τσουρέλα Μάριος
Αρ.Μητρώου 164769

Επιβλέπων
Χαράλαμπος Μπράτσας
Επίκουρος Καθηγητής

Ημερομηνία 06/01/2025

Τίτλος Δ.Ε. Οπτικοποίηση Sparql βιβλιοθήκης σε Javascript

Κωδικός Δ.Ε.23329

Ονοματεπώνυμο φοιτητή Τσουρέλας Μάριος

Ονοματεπώνυμο εισηγητή Χαράλαμπος Μπράτσας

Ημερομηνία ανάληψης Δ.Ε.13-11-2023

Ημερομηνία περάτωσης Δ.Ε.13-11-2025

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

<<Στους αγαπημένους μου ανθρώπους>>

Πρόλογος

Η επιλογή της διπλωματικής εργασίας με θέμα την οπτικοποίηση Sparql βιβλιοθήκης σε Javascript προέκυψε από το ενδιαφέρον μου για τον σημασιολογικό ιστό και την εγγενής ανάγκης προσκόλλησεις νοήματος στα δεδομένα. Η Sparql είναι μια από τα πιο βασικά εργαλεία που κάποιος πρέπει να εξοικειωθεί προκειμένου να κάνει το επόμενο βήμα στην ενασχόλησή του με τον σημασιολογικό ιστό. Τα RDF δεδομένα προσφέρουν μοναδικές δυνατότητες για την ανάλυσή τους αλλά και την διαχείρισή τους. Η ενσωμάτωση αυτών με ένα διαδραστικό περιβάλλον που μπορεί να προσφέρει η JavaScript μου έδωσε την έμπνευση να εξερευνήσω ακόμα περισσότερο τις τεχνικές οπτικοποίησης αυτών των πολύπλοκων δεδομένων. Παράλληλα, ήταν μια αφορμή να αναπτύξω τις δυνατότητές μου στο Frontend αλλά και στην ανάπτυξη Backend. Συνολικά η εργασία αυτή χρειάζεται ένα ολοκληρωμένο σύνολο ικανοτήτων, από τις πιο απλές δεξιότητες μέχρι πιο περίπλοκες, προκειμένου να γίνει πραγματικότητα. Οι δεξιότητες αυτές είναι αναγκαίες για κάθε νέο που θέλει να ασχοληθεί στο τομέα της πληροφορικής.

Περίληψη

Η παρούσα εργασία έχει ως βασικό σκοπό την οπτικοποίηση δεδομένων SPARQL με τη χρήση JavaScript, συνδυάζοντας τεχνολογίες αιχμής για τη δημιουργία μιας εφαρμογής που διευκολύνει την κατανόηση και την παρουσίαση σύνθετων δεδομένων. Το ταξίδι προς την ολοκλήρωσή της περιλαμβάνει τρία βασικά μέρη, τα οποία συνεισέφεραν καθοριστικά στην επίτευξη του στόχου. Το πρώτο μέρος επικεντρώνεται στην κατανόηση του SPARQL και του Σημασιολογικού Ιστού. Ο Σημασιολογικός Ιστός, ως εξέλιξη του παραδοσιακού Παγκόσμιου Ιστού, στοχεύει στη δομημένη οργάνωση των δεδομένων, καθιστώντας τα διαλειτουργικά και εύκολα προσβάσιμα μέσω κοινών προτύπων και τεχνολογιών. Η σημασία του SPARQL έγκειται στην ικανότητά του να ανακτά και να διαχειρίζεται δεδομένα από πολύπλοκα δίκτυα γνώσης, δημιουργώντας νέες προοπτικές για την ανάλυση και την αξιοποίηση πληροφοριών. Το δεύτερο μέρος αφορά το Backend, όπου χρησιμοποιήθηκαν το Node.js και το NestJS για την ανάπτυξη μιας κλιμακούμενης και ασφαλούς υποδομής. Οι modular αρχές και η δομημένη αρχιτεκτονική που προσφέρει το NestJS απλοποίησαν τη δημιουργία REST APIs, επιτρέποντας την αποδοτική επικοινωνία μεταξύ του frontend και των βάσεων δεδομένων. Στο τρίτο μέρος, το Frontend έπαιξε καθοριστικό ρόλο στη βελτίωση της εμπειρίας χρήστη. Με τη σωστή σχεδίαση και τη χρήση σύγχρονων εργαλείων, δημιουργήθηκε μια διαδραστική και φιλική διεπαφή που καθιστά τα δεδομένα προσβάσιμα, ευανάγνωστα και εύκολα κατανοητά. Συνδυάζοντας αυτές τις τρεις πτυχές, η εργασία όχι μόνο έθεσε τις βάσεις για την περαιτέρω αξιοποίηση του SPARQL και του Σημασιολογικού Ιστού, αλλά παράλληλα ενίσχυσε τις γνώσεις και τις δεξιότητές μου στην ανάπτυξη σύγχρονων και ολοκληρωμένων εφαρμογών. Όλως ο κώδικας της εργασίας βρίσκεται στο Github το link είναι : https://github.com/MarTsou98/Sparql_Visual_Graph

<<Visualization of Sparql Library using Javascript>>

<<Tsourelas Marios>>

Abstract

The main purpose of this project was to visualize SPARQL data using JavaScript, combining cutting-edge technologies to create an application that simplifies the understanding and presentation of complex data. The journey to its completion consists of three key parts, each of which played a vital role in achieving the goal.

The first part focuses on understanding SPARQL and the Semantic Web. The Semantic Web, as an evolution of the traditional World Wide Web, aims to organize data in a structured way, making it interoperable and easily accessible through common standards and technologies. The importance of SPARQL lies in its ability to retrieve and manage data from complex knowledge networks, opening new possibilities for data analysis and utilization.

The second part revolves around the Backend, where Node.js and NestJS were employed to develop a scalable and secure infrastructure. The modular principles and structured architecture provided by NestJS facilitated the creation of REST APIs, enabling efficient communication between the frontend and the databases.

The third part highlights the role of the Frontend in enhancing the user experience. Through thoughtful design and the use of modern tools, a dynamic and user-friendly interface was created, making data accessible, readable, and easy to comprehend.

By combining these three aspects, the project not only set the foundation for further exploration of SPARQL and the Semantic Web but also strengthened my knowledge and skills in developing modern, integrated applications.

Ευχαριστίες

Σε αυτό το σημείο θέλω να ευχαριστήσω την οικογένεια μου για την οικονομική και ηθική υποστήριξη που μου έδωσαν. Θέλω να ευχαριστήσω τους φίλους μου που μου έδωσαν δύναμη να συνεχίσω την εκπόνηση της εργασίας. Στο κύριο Χαράλαμπο Μπράτσα που δέχτηκε και με βοήθησε στην εκπόνηση της εργασίας, αλλά και στον Λάζαρο Ιωαννίδη που με βοήθησε με την καθοδήγησή του στο να ολοκληρωθεί αυτή η εργασία.

Περιεχόμενα

Κατάλογος σχημάτων

Συντομογραφίες

SPARQL	SPARQL Protocol and RDF Query Language
RDF	Resource Description Framework
RDFS	Resource Description Framework schema
W3C	World Wide Web Consortium
NLP	Natural Language Processing
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation for Linked Data
LLM	Large Language Models
OWL	Web Ontology Language
URIs	Uniform Resource Identifiers
SKOS	Simple Knowledge Organization System
API	Application Programming Interface
RDF4J	RDF for Java
R2RML	RDB to RDF Mapping Language
npm	Node Package Manager
REST API	Representational State Transfer Application Programming Interface
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
SVG	Scalable Vector Graphics

Κεφάλαιο 1ο: Sparql

εικόνα 1.1 SPARQL Logo

1.1 Εισαγωγή

Ο όρος σημασιολογία προέρχεται από τις λέξεις σημασία και λόγος. Η Σημασιολογία είναι ο κλάδος της γλωσσολογίας, της φιλοσοφίας και της επιστήμης υπολογιστών που ασχολείται με τη μελέτη της σημασίας. Στο ευρύτερο πλαίσιο, εξετάζει πώς οι λέξεις, οι φράσεις, τα σύμβολα και τα νοήματα συσχετίζονται και πώς αποκτώνται, ερμηνεύονται και χρησιμοποιούνται από ανθρώπους ή μηχανές, ανάλογα με την ειδικότητα η σημασιολογία αναφέρεται σε διαφορετικούς τομείς [1].

Ο τομέας της γλωσσολογίας αναλύει την σημασία των λέξεων, των προτάσεων και των κειμένων. Αναρωτιέται, ο τομέας της γλωσσολογίας, το ποια είναι η σημασία μια λέξης ή φράσης, το πως καταφέρνει ο συνδυασμός των λέξεων να παράγουν πολυσύνθετα νοήματα και ποια είναι μεταβολή της σημασίας ανάλογα με την πρόθεση ή και το πλαίσιο του πομπού. Υπάρχουν λέξεις που παρόλο που το νόημα τους είναι κοινό στις περισσότερες γλώσσες και πολιτισμούς, η σημασία τους καταλήγει να είναι διαφορετική. Παράδειγμα στα ελληνικά η λέξη “σπίτι” σημαίνει κυρίως το κτίριο όπου ζει κάποιος ενώ στην αγγλική γλώσσα περιγράφει όχι μόνο το κτήριο αλλά και την οικογενειακή θαλπωρή. Ακόμα αναλύονται οι συσχετίσεις μεταξύ των συνωνύμων, όπως όμορφος και ωραίος, αλλά και τις έννοιες των αντωνυμιών, καλός και κακός [2].

Στο τομέα της Φιλοσοφίας η σημασιολογία ασχολείται με το τι σημαίνουν τα σύμβολα και οι προτάσεις και το πως σχετίζονται με την πραγματικότητα. Αναρωτιέται, ο τομέας της φιλοσοφίας, πως αποκτά νόημα μια λέξη, για παράδειγμα πως αποκτά νόημα μια λέξη όπως η ελευθερία. Ποια είναι η συσχέτιση μεταξύ της γλώσσα και της πραγματικότητας, παράδειγμα από ποια σημείο ένα φυτό αναφέρεται σαν δέντρο και όχι σαν θάμνος. Ποιος είναι ο ρόλος του υποκείμενου στην ερμηνεία της σημασίας, για παράδειγμα πως εξηγείται η φράση αυτός είναι καλός άνθρωπος, τι θεωρείται καλός [1].

Στην επιστήμη των ηλεκτρονικών υπολογιστών η σημασιολογία έχει πρωταγωνιστικό ρόλο σε πεδία όπως η τεχνητή νοημοσύνη, η μηχανική μάθησης και ο Σημασιολογικός Ιστός. Πολλές από τις πιο προηγμένες τεχνολογίες έχουν ανάγκη την σημασιολογία, με τρανταχτό παράδειγμα αυτό ενός Virtual

Assistant, όπως η Siri της εταιρείας Apple, μεταξύ άλλων όπως η Αλέξα της Amazon και η Google Assistant. Ο τρόπος που χρησιμοποιείται η σημασιολογία στους Virtual Assistants είναι μέσω της NLP (Natural Language Processing), που περιλαμβάνει τα εξής κομμάτια: την ανάλυση της πρότασης, την συσχέτιση των δεδομένων και την εκτέλεση της κατάλληλης ενέργειας. Για παράδειγμα, έστω ότι ο χρήστης δίνει μια εντολή όπως “Παίξε το αγαπημένο μου τραγούδι”, ο Virtual Assistant πρέπει να κατανοήσει την δομή της πρότασης: υποκείμενο, εσύ (δηλαδή ο Virtual Assistant), ρήμα, παίξε, αντικείμενο, τραγούδι. Πρέπει επίσης να ερμηνεύσει την λέξη "αγαπημένο" βάσει του χρήστη. Η λέξη "αγαπημένο" χρήζει σημασιολογική κατανόηση, αφού η αγάπη είναι εντελώς υποκειμενική. Αυτή η λέξη μπορεί να σχετίζεται με το ιστορικό του χρήστη, μπορεί όμως να σχετίζεται και με το ποια τραγούδια έχει σημειώσει ως αγαπημένα, ή μπορεί να είναι ένας συνδυασμός αυτών των δύο. Έπειτα, ο Virtual Assistant αναζητά τα δεδομένα από διάφορες πηγές, όπως την προσωπική του βιβλιοθήκη ή κάποιας streaming υπηρεσίας όπως το Spotify ή το Youtube. Πράττει τους σημασιολογικούς συνδέσμους, όπως το πόσες φορές εμφανίστηκε κάποιο τραγούδι, ποια είναι η βαθμολογία του. Έφοσον κατανοήσει το αίτημα του χρήστη, ο Virtual Assistant εκκινεί το σύστημα αναπαραγωγής ήχου με το τραγούδι που έκρινε ότι είναι το αγαπημένο του χρήστη [3].

Υπάρχουν και πολλές προκλήσεις που έρχονται αντιμέτωποι οι Virtual Assistants. Η φυσική γλώσσα είναι γεμάτη με αμφίβολες έννοιες. Η πολυπλοκότητα της γλώσσας δημιουργεί πολλά προβλήματα. Για παράδειγμα, ακολουθώντας το προηγούμενο παράδειγμα, μια φράση όπως “Παιξε Summer” είναι άκρως υποκειμενική. Το “Παιξε Summer” μπορεί να εννοεί ένα τραγούδι που ονομάζεται *Summer*, ένα τραγούδι που έχει ως κεντρικό θέμα την λέξη *Summer*, μια λίστα τραγουδιών που ονομάζονται *Summer*, ή ακόμα και το πρώτο τραγούδι που σχετίζεται με την λέξη *Summer*. Επίσης είναι αναγκαίο ο Virtual Assistant να αντιλαμβάνεται και το πολιτισμικό αλλά και το συναισθηματικό υπόβαθρο του χρήστη. Φυσικά, δεν μπορεί να παραμερίσουμε το γεγονός ότι μια φράση μπορεί να ειπωθεί με πολλούς τρόπους. Για παράδειγμα, η φράση “παίξε μουσική” και “Βάλε μουσική” έχουν ακριβώς το ίδιο νόημα. Αυτά τα προβλήματα βελτιώνονται με την χρήση LLM (Large Language Models), από την προσωποποίηση που κάνουν στον χρήστη και το πλαίσιο που δημιουργούνται τα αιτήματα [3].

Η διαλειτουργικότητα που μπορεί να επιτευχθεί μέσω της σημασιολογίας μπορεί να βελτιώσει την αποτελεσματικότητα και την ποιότητα της ζωής μας. Με την χρήση της διαλειτουργικότητας που γίνεται εφικτή από τη σημασιολογία, μπορούν να συσχετιστούν και να ενοποιηθούν δεδομένα μεταξύ διαφορετικών συστημάτων. Έστω, ότι υπάρχουν δεδομένα ασθενών αποθηκευμένα σε διαφορετικές βάσεις δεδομένων ακόμα και διαφορετικά συστήματα, όπως εργαστηριακά αποτελέσματα, φαρμακευτικές αγωγές, προηγούμενες διαγνώσεις. Ένας γιατρός μπορεί να λάβει μια ενοποιημένη απάντηση σε μια συνδυαστική ερώτηση δίχως να ξοδεύει χρόνο στο να δημιουργεί αυτός τις συσχετίσεις. Αυτό έχει ως αποτέλεσμα ο γιατρός να δέχεται μια πλήρη εικόνα ενός ασθενή, αποφεύγονται τα λάθη και οι επικαλύψεις των δεδομένων, και όλα αυτά δίχως να χρειάζεται ο ανασχεδιασμός των συστημάτων [1].

Η σημασιολογία πρωταγωνιστεί ιδιαίτερα όταν εφαρμόζεται σε τεράστιας κλίμακας συστήματα διαχείρισης και ανταλλαγής δεδομένων, όπως ο Σημασιολογικός Ιστός. Ο Σημασιολογικός Ιστός επιδιώκει να δώσει στα δεδομένα νόημα και την ικανότητα της διαλειτουργικότητας, επιτρέποντας και στους ανθρώπους αλλά και στους υπολογιστές να χρησιμοποιούν τα δεδομένα με εξυπνότερο και πιο αποτελεσματικό τρόπο. Ο Σημασιολογικός Ιστός δίνει τη δυνατότητα για τη δημιουργία συνδεδεμένων δεδομένων, μετατρέποντάς τα σε πιο κατανοητά, και μπορεί να υπάρχει συνεργασία μεταξύ συστημάτων και χρηστών [2].

1.2 Βασικές έννοιες του Σημασιολογικού ιστού

Ο Σημασιολογικός Ιστός είναι ο επόμενος εξελικτικός κόμβος του Παγκόσμιου Ιστού, ο στόχος του είναι να μετατρέψει τον Ιστό από έναν χώρο κυρίως πληροφοριών και δεδομένων που κατανοούνται από ανθρώπους σε έναν χώρο όπου οι πληροφορίες και τα δεδομένα είναι κατανοητά από ανθρώπους αλλά και από ηλεκτρονικούς υπολογιστές. Αυτό γίνεται πραγματικότητα με μια σειρά από θεμελιώδεις έννοιες και τεχνολογίες που επιτρέπουν την οργάνωση, την επεξεργασία και την αποστολή/αποδοχή δεδομένων με εννοιολογικό τρόπο. Οι έννοιες όπως τριάδες RDF, οντολογία, Linked Data παρέχουν πλαίσιο για την αναπαράσταση και κατανόηση των δεδομένων με μεγαλύτερη ακρίβεια. Αυτές οι έννοιες δημιουργούν ένα δίκτυο διασυνδεδεμένων δεδομένων που όχι μόνο μπορεί και κατηγοριοποιεί, οργανώνει τις πληροφορίες, αλλά και ενισχύει τη διαλειτουργικότητα μεταξύ διαφορετικών συστημάτων και εφαρμογών. Όλες αυτές οι έννοιες συνεργάζονται αρμονικά για να μετατρέψουν τον Ιστό σε ένα πιο έξυπνο, κατανοητό και αποτελεσματικό εργαλείο τόσο για ανθρώπους όσο και για ηλεκτρονικούς υπολογιστές[4].

Ξεκινώντας, μια βασική έννοια και απαραίτητη να αναφερθεί είναι αυτή των δομημένων δεδομένων. Τα δομημένα δεδομένα αφορούν πληροφορίες που οργανώνονται βάση με μια διάρθρωση ή μορφή. Μερικά απλά παραδείγματα είναι ένας πίνακας, μια βάση δεδομένων ή ακόμα και ένα αρχείο JSON. Συνήθως είναι αποθηκευμένα σε μορφή ενός πίνακα με σειρές και στήλες για την ευκολότερη επεξεργασία, αναζήτηση και ανάλυση. Η δομή τους είναι ξεκάθαρη και δίνεται στους ηλεκτρονικούς υπολογιστές η δυνατότητα να εκτελούν υπολογισμούς και αναλύσεις με ακρίβεια και ταχύτητα. Ακόμα η οργάνωση των δομημένων δεδομένων είναι ιδανικά για αυτοματοποιήσεις, κάνοντας την εξαγωγή των δεδομένων, την κατηγοριοποίηση και την επαναχρησιμοποίηση των πληροφοριών δυνατή[5].

Στη συνέχεια, ένα αναπόσπαστο κομμάτι του σημασιολογικού ιστού είναι η δομή των τριάδων RDF. Οι τριάδες τύπου RDF είναι βασική μονάδα αναπαράστασης δεδομένων στο σημασιολογικό ιστό. Κάθε τριάδα τύπου RDF περιλαμβάνει τρία στοιχεία: το υποκείμενο, πιο γνωστό ως subject, το ρήμα, πιο γνωστό ως predicate, και το αντικείμενο, πιο γνωστό ως object, τα οποία συνθέτουν μια ακριβή δήλωση ή σχέση μεταξύ δύο αντικειμένων. Για παράδειγμα, έστω ότι έχουμε μια τριάδα τύπου RDF, "Η Μαρία (υποκείμενο) έχει (ρήμα) γάτα (αντικείμενο)". Στο συγκεκριμένο παράδειγμα το υποκείμενο “η Μαρία”, το ρήμα “έχει” δηλώνει τη σχέση της με το αντικείμενο, την “γάτα”. Αυτή η δομή επιτρέπει την οργάνωση και διασύνδεση δεδομένων από άλλες πηγές, πραγματοποιώντας τη διαλειτουργικότητα, την ευκολότερη αναζήτηση και τις εξαγωγές πληροφοριών. Οι τριάδες τύπου RDF δίνουν τη δυνατότητα στους ηλεκτρονικούς υπολογιστές να καταλαβαίνουν και να επεξεργάζονται τα δεδομένα έτσι ώστε να μπορεί να δημιουργηθεί ένα δίκτυο δεδομένων[6].

Οι οντολογίες είναι ένας ακόμα από τους βασικούς πυλώνες του σημασιολογικού ιστού. Οι οντολογίες παρέχουν ένα κοινό λεξιλόγιο για την αναπαράσταση δεδομένων και τις μεταξύ τους σχέσεις. Μέσω των οντολογιών, οι πληροφορίες αποκτούν δομή και οι μηχανές μπορούν να κατανοήσουν τη σημασία τους. Οι οντολογίες επιτρέπουν τη μοντελοποίηση σύνθετων εννοιών και των σχέσεών τους, καθιστώντας δυνατή την αναζήτηση, τη συσχέτιση και την ανάλυση δεδομένων με τρόπο που υπερβαίνει την απλή λέξη-προς-λέξη αντιστοιχία. Οι οντολογίες μπορούν και ενισχύουν τη διαλειτουργικότητα, μεταξύ διαφορετικών συστημάτων, δίνοντας τη δυνατότητα να ενοποιούνται τα δεδομένα από διαφορετικές πηγές σε μια κοινή μορφή, αυξάνοντας την αποτελεσματικότητα[7].

Η SPARQL αποτελεί την κύρια γλώσσα ερωτημάτων που χρησιμοποιείται για την εκτέλεση ερωτημάτων σε δεδομένα που αποθηκεύονται σε μορφή RDF. Οι χρήστες με τη χρήση της SPARQL, μπορούν να “τραβάνε”, να αναλύουν και να διασυνδέουν δεδομένα που είναι οργανωμένα σε τριάδες, δίνοντας τη δυνατότητα να εξάγουν πολύπλοκες πληροφορίες από RDF βάσεις δεδομένων. Η γλώσσα

SPARQL παρέχει εργαλεία για τη διαχείριση, αναζήτηση, τη χρήση φίλτρων, συνδυασμό διαφορετικών πηγών δεδομένων, ακόμα και την επεξεργασία των απαντήσεων των ερωτημάτων. Αυτό την καθιστά ένα πανίσχυρο εργαλείο για τη διαχείριση συνδεδεμένων δεδομένων και την ανάλυση στον σημασιολογικό ιστό[8].

Τα URIs είναι μοναδικά αναγνωριστικά ονόματα, και χρησιμοποιούνται για την αναγνώριση πόρων στον Ιστό, δίνοντας τη δυνατότητα να υπάρχει σαφής αναφορά σε δεδομένα. Κάθε ένα URI αντιπροσωπεύει μόνο έναν πόρο, ανεξαρτήτως του τι ή ποιος είναι αυτός. Ένα URI μπορεί να είναι μια ιστοσελίδα μέχρι μια πληροφορία σε μια βάση δεδομένων. Η σημαντικότητα των URI είναι ύψιστης, ειδικά στο σημασιολογικό ιστό, διότι επιτρέπουν σε πόρους να συνδέονται και να εντοπίζονται μεταξύ τους, με τέτοιο τρόπο ώστε να μπορεί να το καταλάβει και ένας άνθρωπος αλλά και ένας ηλεκτρονικός υπολογιστής. Η χρήση των URI εγγυάται ότι τα δεδομένα διατηρούν τη μοναδικότητά τους και μπορούν να εντοπιστούν από παγκόσμια κλίμακα, επιτρέποντας τη διαλειτουργικότητα και την αναπαράσταση των πληροφοριών στο διαδίκτυο[9].

Μια ακόμη από τις βασικές έννοιες του σημασιολογικού ιστού είναι αυτή των Λεξιλογίων. Τα λεξιλόγια, ή πιο γνωστά Vocabularies, αποτελούν σύνολα από ορισμούς, έννοιες και σχέσεις που υπάρχουν στον σημασιολογικό ιστό, δίνοντας τη δυνατότητα να αναπαριστούν τα δεδομένα με κατανοητό τρόπο. Μέσα από τα λεξιλόγια, οι έννοιες και οι σχέσεις που ενθυλακώνουν τα δεδομένα αποκτούν σαφήνεια, αλλά και βοηθούν στη διασύνδεση και την ανταλλαγή των δεδομένων. Τα λεξιλόγια καθορίζουν χαρακτηριστικά και ιδιότητες στα δεδομένα ενώ παράλληλα πραγματοποιείται ενοποίηση αυτών[10].

Η διαλειτουργικότητα, γνωστή ως Interoperability, είναι η δυνατότητα να μοιράζονται, αναλύονται και να χρησιμοποιούνται δεδομένα από διαφορετικά συστήματα και εφαρμογές, ανεξαρτήτως περιορισμών σχετικά με τη μορφή ή την πηγή τους. Αυτό επιτυγχάνεται μέσω των RDF και της SPARQL, αφού επιτρέπουν την ανταλλαγή και επεξεργασία των δεδομένων από διαφορετικές πλατφόρμες. Έτσι είναι δυνατόν να συνδεθούν και να ενοποιηθούν δεδομένα από πολλές πηγές, βελτιώνοντας τη συνεργασία μεταξύ των πηγών και κάνοντας πιο εύκολη την ανάπτυξη των εφαρμογών που λειτουργούν σε εντελώς διαφορετικά περιβάλλοντα. Επομένως, η διαλειτουργικότητα είναι μια από τις θεμελιώδεις αρχές για τη δημιουργία ενός σημασιολογικού ιστού[11].

Στην συνέχεια η αυτοματοποιημένη συλλογιστική, γνωστή ως Automated Reasoning, ονομάζεται η διαδικασία κατά την οποία ηλεκτρονικοί υπολογιστές χρησιμοποιούν λογικούς κανόνες και αλγόριθμους με σκοπό να εξάγουν νέες πληροφορίες, βασισμένοι στις ήδη υπάρχουσες σχέσεις και πληροφορίες που υπάρχουν. Στον σημασιολογικό ιστό είναι δυνατό, μέσω της συλλογιστικής, να εξάγουμε νέα δεδομένα από τις ήδη αποθηκευμένες τριαδες τύπου RDF, καταφέρνοντας έτσι να δημιουργούνται συστήματα ικανά να κατανοούν και να επεξεργάζονται πληροφορίες με σημασιολογικό σκοπό. Μέσω της αυτοματοποιημένης συλλογιστικής μπορούμε να αποκαλύψουμε μυστικές σχέσεις, να βελτιωθεί η ακρίβεια των αναζητήσεων και να ενισχυθεί η ποιότητα των δεδομένων. Η αυτοματοποιημένη συλλογιστική είναι κρίσιμη για την ανάπτυξη προηγμένων εφαρμογών, όπως η τεχνητή νοημοσύνη [12].

Η σημασιολογική αναφορά, γνωστή ως Semantic Annotation, είναι η διαδικασία κατά την οποία προστίθενται ετικέτες σε δεδομένα, με σκοπό να ενισχυθεί το νόημά τους, ώστε οι ηλεκτρονικοί υπολογιστές να μπορούν να τους κατανοήσουν. Μέσω αυτών των ετικετών τα δεδομένα εμπλουτίζονται με πληροφορίες, κάνοντας τις έννοιες πιο σαφείς και κατανοητές από συστήματα, επιτρέποντας μια βελτιωμένη επεξεργασία. Με αυτόν τον τρόπο δημιουργούνται επιπλέον επίπεδα νοήματος που διευκολύνουν την αναζήτηση, την ανάλυση και τη συσχέτιση των δεδομένων. Έτσι

μπορεί να δημιουργηθεί ένα ισχυρό και ευέλικτο οικοσύστημα δεδομένων, κάνοντας τις πληροφορίες πολύ πιο προσβάσιμες, επαναχρησιμοποιήσιμες, τόσο για τους ανθρώπους όσο και για τους ηλεκτρονικούς υπολογιστές [13].

Τέλος, μια από τις πιο σημαντικές πτυχές του Σημασιολογικού Ιστού είναι αυτή του Linked Data. Με τον όρο Linked Data ονομάζουμε την πρακτική σύνδεσης των δεδομένων από διαφορετικές πηγές, έτσι ώστε να μπορούν να συνδέονται και να αναγνωρίζονται μεταξύ τους μέσω του Διαδικτύου. Η φιλοσοφία του Linked Data βασίζεται στην ιδέα ότι τα δεδομένα δεν πρέπει να είναι απομονωμένα σε ξεχωριστές βάσεις δεδομένων, αλλά αντίθετα να συνδέονται με άλλες σχετικές πληροφορίες, δημιουργώντας ένα δίκτυο δεδομένων που είναι εύκολα προσβάσιμο και αναγνωρίσιμο από ανθρώπους και ηλεκτρονικούς υπολογιστές. Υπάρχουν τέσσερις βασικές αρχές του Linked Data. Τα δεδομένα πρέπει να είναι προσβάσιμα μέσω URIs, δηλαδή μια μοναδική αναφορά σε αυτό το αντικείμενο. Δεύτερον, οι πληροφορίες πρέπει να είναι σε μορφή RDF, δηλαδή subject - predicate - object. Τρίτον, οι διασυνδέσεις μεταξύ των τριπλετών RDF πρέπει να συνδέονται μέσω URIs. Τέλος, τα δεδομένα πρέπει να διαθέτουν links προς άλλες πηγές στο διαδίκτυο. Με αυτόν τον τρόπο κάθε δεδομένο στο Linked Data μπορεί να συνδέεται με άλλα δεδομένα που υπάρχει συσχέτιση μεταξύ τους [14].

1.3 πρότυπα και εργαλεία του Σημασιολογικού ιστού.

Ο Σημασιολογικός Ιστός βασίζεται σε μια σειρά από πρότυπα και εργαλεία που επιτρέπουν την αποτελεσματική αναπαράσταση, οργάνωση και διασύνδεση δεδομένων. Τα πρότυπα αυτά και τα εργαλεία διασφαλίζουν ότι είναι εφικτή η διασύνδεση των δεδομένων μεταξύ τους, με τέτοιο τρόπο ώστε οι ηλεκτρονικοί υπολογιστές να μπορούν να κατανοούν τις σχέσεις μεταξύ της πληροφορίας και να παρέχουν αυτοματοποιημένα αποτελέσματα και αναλύσεις. Ιστορικά, ο Σημασιολογικός Ιστός οραματίστηκε από τον ίδιο τον δημιουργό του Παγκοσμίου Ιστού, Tim Berners-Lee. Η ιδέα του πρωτοεμφανίστηκε για πρώτη φορά στα τέλη της δεκαετίας του 1990. Το 2001, το όραμα αυτό περιγράφηκε εκτενώς στο άρθρο *"The Semantic Web"* που δημοσιεύθηκε στο περιοδικό *Scientific American*, το οποίο έθεσε τις βάσεις για τις τεχνολογίες του RDF και των οντολογιών. Η W3C υποστήριξε πλήρως την ιδέα του Σημασιολογικού Ιστού, η οποία ανέπτυξε τα σχετικά πρότυπα και εργαλεία για αυτήν, καθιστώντας τη διαλειτουργικότητα και την αυτοματοποιημένη επεξεργασία δεδομένων εφικτή [9].

1.2 Λογότυπο του W3C

Το πρότυπο RDF είναι θεμέλιο του Σημασιολογικού Ιστού, και πρόκειται για ένα μοντέλο

αναπαράστασης δεδομένων που περιγράφει τις πληροφορίες με τη μορφή τριάδων υποκειμένου - ρήματος - αντικειμένου, πιο γνωστό ως *subject - predicate - object*. Ένα απλό παράδειγμα μιας τριάδας είναι το βιβλίο (υποκείμενο - subject), έχει τίτλο (αντικείμενο - predicate), "Ο Μικρός Πρίγκιπας" (αντικείμενο - object). Το RDF μπορεί και οργανώνει τα δεδομένα με τέτοιο τρόπο ώστε να μπορούν να τα καταλάβουν και οι άνθρωποι αλλά και οι ηλεκτρονικοί υπολογιστές. Το RDF επιτρέπει την ακριβή αναπαράσταση και τη διασύνδεση των πληροφοριών. Οι τριάδες αυτές μπορούν να περιγράφουν οποιοδήποτε είδος πληροφορίας, όπως ιδιότητες αντικειμένων, σχέσεις μεταξύ τους ή ακόμα και πολυσύνθετες αφηγήσεις μεταξύ αυτών. Η ικανότητα του RDF να είναι ευέλικτο, το καθιστά ικανό να προστεθούν νέες τριάδες δίχως να απαιτείται αλλαγή στο βασικό σχήμα των δεδομένων [11].

Μια από τις πιο σημαντικές πτυχές του Σημασιολογικού Ιστού είναι αυτή της γλώσσας ερωτημάτων SPARQL. Η SPARQL είναι ειδικά φτιαγμένη έτσι ώστε να μπορεί να διαχειριστεί και να επεξεργαστεί τον χειρισμό, την εξαγωγή και την εισαγωγή δεδομένων από RDF βάσεις δεδομένων. Η γλώσσα ερωτημάτων SPARQL (SPARQL Protocol and RDF Query Language) παρέχει έναν ισχυρό και ευέλικτο τρόπο για να ανακτούν, να επεξεργάζονται και να συνδυάζουν δεδομένα που είναι αποθηκευμένα σε μορφή RDF και δίνει στους χρήστες τη δυνατότητα να γράφουν σύνθετα ερωτήματα που καταφέρνουν να αναζητήσουν πληροφορίες από πολλά τελικά σημεία, πιο γνωστά ως *endpoints*, μπορώντας έτσι να αξιοποιήσουν την ευχέρεια των τριάδων RDF και των σχέσεων που δημιουργούνται μεταξύ τους [8].

Το πρότυπο RDFS πρόκειται για μια επέκταση του RDF. Αυτό που προσθέτει στο RDF πρότυπο είναι επιπρόσθετες σημασιολογικές δυνατότητες, επιτρέποντας την οργάνωση των δεδομένων σε ιεραρχίες και τη δημιουργία ταξινομήσεων και ιδιοτήτων. Το πρότυπο RDFS προσφέρει εργαλεία για τον ορισμό κατηγοριών (classes) και σχέσεων (properties) μεταξύ δεδομένων, επιτρέποντας την αναπαράσταση πιο πολύπλοκων και δομημένων πληροφοριών. Μέσω του RDFS, μπορούμε να ορίσουμε ιεραρχίες κατηγοριών και να προστεθούν κανόνες σχετικά με την κληρονομικότητα και τη συνέχεια των δεδομένων. Οι υποκατηγορίες κληρονομούν αυτόματα όλες τις ιδιότητες της ανώτερης κατηγορίας. Έτσι βοηθά στη δημιουργία καλύτερης οργάνωσης και ευκολότερης διαχείρισης μεγάλων και πολυσύνθετων δεδομένων, ενώ ταυτόχρονα ενισχύει τη δυνατότητα των ηλεκτρονικών υπολογιστών να κατανοούν και να επεξεργάζονται με έναν πιο εννοιολογικό τρόπο [10].

Το πρότυπο OWL είναι ένας ακόμα κεντρικός πυλώνας του Σημασιολογικού Ιστού, και η χρησιμότητά του είναι η δημιουργία οντολογιών, δηλαδή σύνθετων και τυποποιημένων μοντέλων δεδομένων που περιγράφουν τις σχέσεις μεταξύ των όρων. Το πρότυπο OWL παρέχει μια ισχυρότερη και πιο εκφραστική γλώσσα από το RDFS, επιτρέποντας την αναπαράσταση πολύπλοκων εννοιών και λογικών σχέσεων. Με τη χρήση του προτύπου OWL, δίνεται η δυνατότητα δηλώσεις λεπτομερών περιορισμών. Συλλογισμοί και λογικοί κανόνες υποστηρίζονται από το πρότυπο OWL, έτσι οι ηλεκτρονικοί υπολογιστές καταφέρνουν να εξάγουν νέα δεδομένα από υπάρχουσες πληροφορίες. Αυτό καθιστά το πρότυπο OWL, απαραίτητο εργαλείο για τη δημιουργία εφαρμογών που σχετίζονται με γνώση. Με τη χρήση του OWL, οι εφαρμογές μπορούν να διαχειρίζονται και να επεξεργάζονται πολύπλοκα δίκτυα γνώσης με αυξημένη ακρίβεια και συνέπεια, καθιστώντας το ιδιαίτερα χρήσιμο σε τομείς όπως η τεχνητή νοημοσύνη, η βιοϊατρική και η διαχείριση μεγάλων δεδομένων [8].

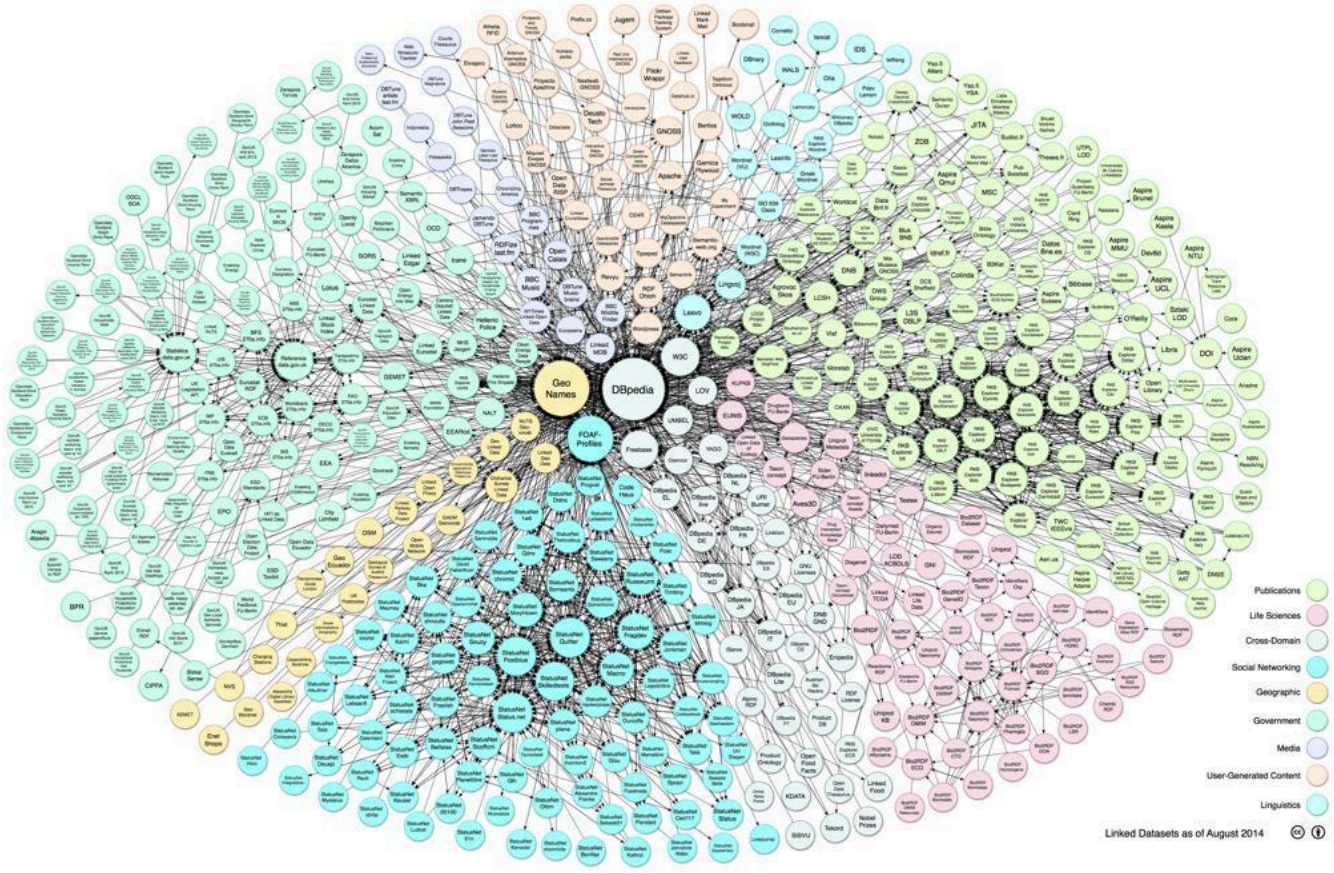
Ένα ακόμα πρότυπο είναι αυτό του SKOS. Το SKOS πρόκειται για ένα πρότυπο ειδικά φτιαγμένο για την οργάνωση, την ανταλλαγή συστημάτων γνώσης, όπως θησαυροί, ταξινομεί και λεξιλόγια. Με αυτό επιτρέπεται η διαλειτουργικότητα μεταξύ διαφορετικών συστημάτων οργάνωσης γνώσης, παρέχοντας ένα κοινό μοντέλο για την αναπαράσταση και την ανταλλαγή εννοιών. Μέσω του SKOS, μπορούν να οργανωθούν οι έννοιες σε ιεραρχία, να συσχετιστούν με σχέσεις όπως γενικότερος όρος, ειδικότερος

όρος, σχετικός όρος καθώς και επικαλούνται ορισμοί, ετικέτες και άλλα παραδείγματα. Το SKOS κάνει πιο εύκολη την ενσωμάτωση διαφορετικών θησαυρών και λεξιλογίων σε ένα ενιαίο πλαίσιο δεδομένων, ενισχύοντας τη συνεργασία μεταξύ συστημάτων [10].

Τέλος, ένα ακόμα πρότυπο πολύ βασικό είναι το JSON-LD. Το JSON-LD προσφέρει ένα απλό και κατανοητό τρόπο ενσωμάτωσης συνδεδεμένων δεδομένων σε εφαρμογές που χρησιμοποιούν τη δημοφιλή μορφή JSON. Έχει σχεδιαστεί για να συνδυάζει την ευχρηστία του JSON με τις δυνατότητες του RDF, επιτρέποντας την περιγραφή δομημένων δεδομένων με σαφήνεια και ακρίβεια. Μέσω του JSON-LD, οι προγραμματιστές μπορούν να ενσωματώσουν σημασιολογικές πληροφορίες στα δεδομένα τους, κάνοντάς τα κατανοητά τόσο από ανθρώπους όσο και από ηλεκτρονικούς υπολογιστές [11].

Για να υποστηρίξει την υλοποίηση αυτών των προτύπων και να διευκολύνει τη μετάβαση στον Σημασιολογικό Ιστό, η W3C έχει αναπτύξει μερικά εργαλεία. Αυτά τα εργαλεία παρέχουν στους προγραμματιστές και στους ερευνητές τα μέσα για τη δημιουργία, τη διαχείριση και την επεξεργασία δεδομένων σύμφωνα με τις αρχές και τα πρότυπα του Σημασιολογικού Ιστού, κάνοντας την αξιοποίηση των συνδεδεμένων δεδομένων πιο απλή και αποτελεσματική.

Ένα από τα πιο γνωστά εργαλεία είναι το Protégé, ιδανικό για τη δημιουργία και την επεξεργασία οντολογιών. Το Protégé είναι ένα από τα πιο διαδεδομένα και ισχυρά εργαλεία για τη δημιουργία και την επεξεργασία οντολογιών, καθιστώντας το θεμελιώδες στον χώρο του Σημασιολογικού Ιστού. Αναπτύχθηκε από το Πανεπιστήμιο του Stanford και παρέχει στους χρήστες ένα φιλικό περιβάλλον για τη μοντελοποίηση σύνθετων δεδομένων και τη διαχείριση γνώσης. Υποστηρίζει πρότυπα όπως το OWL και το RDF, επιτρέποντας την κατασκευή οντολογιών που μπορούν να ενσωματωθούν απρόσκοπτα σε εφαρμογές του Σημασιολογικού Ιστού. Με δυνατότητες όπως συλλογισμούς, ενσωματωμένους ελεγκτές συνέπειας και μια πλούσια συλλογή πρόσθετων (plugins), το Protégé διευκολύνει την ανάλυση, τη διάχυση και την επαναχρησιμοποίηση δεδομένων. Επιπλέον, η ευελιξία του το καθιστά κατάλληλο για διάφορους τομείς, από την τεχνητή νοημοσύνη και τη βιοϊατρική έως τη διαχείριση επιχειρηματικής γνώσης, ενισχύοντας τη διαλειτουργικότητα και τη συνεργασία μεταξύ συστημάτων [15].



1.3 Ο ιστός των Linked Data

Ένα ακόμα εργαλείο ιδανικό για την ανάπτυξη εφαρμογών που βασίζονται στον σημασιολογικό ιστό και το Linked Data, ονομάζεται Apache Jena. Πρόκειται για ένα εργαλείο ανοιχτού κώδικα, δημιουργημένο από την εταιρεία Hewlett-Packard Laboratories, στις αρχές της δεκαετίας 2000, και το 2012 έγινε επίσημο έργο του Apache. Το Apache Jena καταφέρνει να προσφέρει μια συλλογή εργαλείων και APIs για τη διαχείριση RDF δεδομένων, την δημιουργία οντοτήτων, συλλογισμούς αλλά και την χρήση της γλώσσας SPARQL. Υποστηρίζονται τα πρότυπα RDF, RDFS και OWL, δίνοντας τη δυνατότητα να μοντελοποιηθούν και να επεξεργαστούν πολυσύνθετα δεδομένα. Χρησιμοποιείται κυρίως σε εφαρμογές που απαιτούν προηγμένη διαχείριση των δεδομένων, όπως τεχνητή νοημοσύνη, επιχειρηματική ανάλυση και βιοπληροφορική [16].

Το RDF4J είναι ένα ακόμα ισχυρό εργαλείο ανοιχτού κώδικα, ιδανικό για την αποθήκευση, την επεξεργασία και την ικανότητα να κάνει κάποιος query σε RDF δεδομένα. Παρέχει πλήρη υποστήριξη για τη γλώσσα SPARQL, επιτρέποντας την ανάκτηση, την επεξεργασία και τη διαχείριση RDF δεδομένων, δίνοντας τη δυνατότητα για σύνδεση με εξωτερικά endpoints. Καθιστά το RDF4J ιδανικό για να ενσωματωθεί σε ήδη δημιουργημένες εφαρμογές, αλλά μπορεί επίσης να χρησιμοποιηθεί αυτόνομα. Λόγω της ευελιξίας και της επεκτασιμότητάς του, το RDF4J είναι ευρέως διαδεδομένο σε έργα που απαιτούν προηγμένη διαχείριση δεδομένων σημασιολογικού ιστού [17].

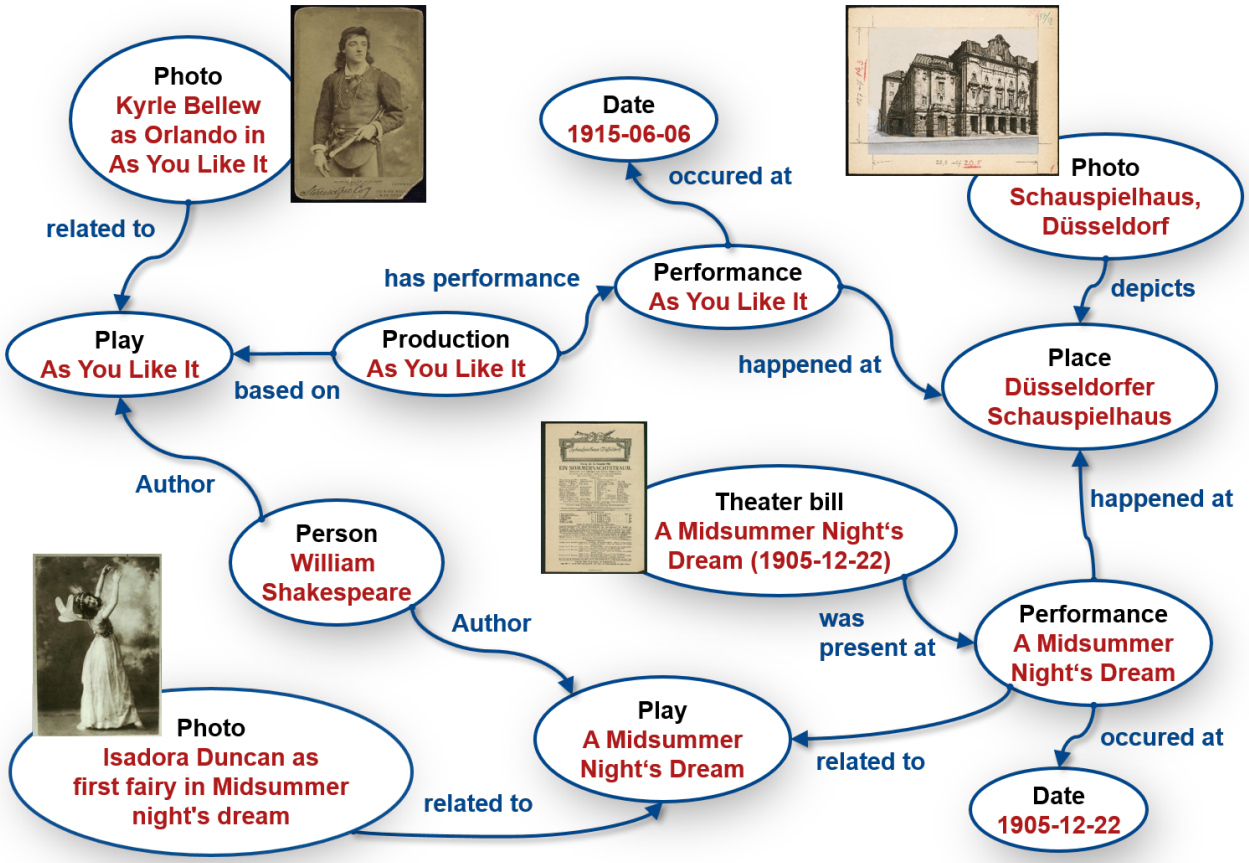
Το GraphDB είναι RDF βάση δεδομένων, που σχεδιάστηκε για την οργάνωση, αποθήκευση και την πρόσβαση σε συνδεδεμένα δεδομένα. Υποστηρίζεται η γλώσσα SPARQL, επιτρέποντας σύνθετες ερωτήσεις και αναλύσεις, και ειδικεύεται στη διαχείριση δεδομένων που βασίζεται στο πρότυπο RDF. Μπορεί να δημιουργεί αυτόματα νέες σχέσεις μέσω λογικής συμπερασματολογίας, εξάγοντας νέα

γνώση από πληροφορίες που ήδη υπάρχουν στη βάση δεδομένων. Η υψηλή απόδοση και η ικανότητά του να κλιμακώνεται την καθιστούν την ιδανική βάση δεδομένων σε τομείς όπως η τεχνητή νοημοσύνη, η ανάλυση μεγάλων δεδομένων και οι έξυπνες βάσεις δεδομένων. Μια παρόμοια βάση δεδομένων είναι η AllegroGraph [18].

Τέλος, το OpenRefine είναι ένα ευέλικτο εργαλείο για το σχεδιασμό αλλά και το καθαρισμό, την επεξεργασία και τη διασύνδεση των δεδομένων με σημασιολογικά δίκτυα και λεξιλόγια. Το OpenRefine βελτιώνει την ποιότητα των δεδομένων, αναγνωρίζοντας και επιδιορθώνοντας σφάλματα, επιτρέπει τη δομημένη οργάνωση και τη μετατροπή δεδομένων στο πρότυπο RDF. Οι χρήστες μπορούν να συνδέσουν τα δεδομένα με λεξιλόγια και βάσεις γνώσεων όπως η DBPedia, ενισχύοντας τη διαλειτουργικότητα. Το εργαλείο είναι ιδιαίτερα δημοφιλές στην επιστήμη των δεδομένων λόγω της εύκολης χρήσης και των προηγμένων δυνατοτήτων του [19].

1.4 Σημασιολογικός Ιστός και η χρησιμότητά του

Ο **σημασιολογικός ιστός** πρόκειται για ένα από τα πιο φιλόδοξα και καινοτόμα εγχειρήματα στο κλάδο των ηλεκτρονικών υπολογιστών. Ο νέος και καινοτόμος τρόπος οργάνωσης και διαχείρισης δεδομένων, υπερβαίνει τα όρια του παραδοσιακού ιστού, μετατρέποντας το διαδίκτυο σε μια πλατφόρμα διασυνδεδεμένης γνώσης. Ο σημασιολογικός ιστός δεν περιορίζεται σε μια απλή συλλογή πληροφοριών, αλλά χρησιμοποιεί τα πρότυπα και τα εργαλεία για να δώσει νόημα σε αυτά, έτσι ώστε να μπορούν να κατανοηθούν και από ανθρώπους αλλά και ηλεκτρονικούς υπολογιστές. Η χρησιμότητά του δεν περιορίζεται μόνο στην κατανόηση των δεδομένων, αλλά και στην εξαγωγή νέων πληροφοριών, στην συνεργασία και την καινοτομία. Με εργαλεία όπως την **SPARQL**, **OWL** και **RDF**, ο σημασιολογικός ιστός μπορεί να δημιουργήσει το περιβάλλον για την ανάπτυξη εφαρμογών που μπορούν να λύσουν σύγχρονα προβλήματα και να προσφέρουν λύσεις σε διάφορους τομείς της ζωής των ανθρώπων [20].



Εικόνα 1.5 Παράδειγμα σημασιολογικού ιστού.

Ο Σημασιολογικός Ιστός μπορεί να προσφέρει σημαντικά οφέλη στον τομέα της υγείας, αφού επιτρέπει τη διασύνδεση και αξιοποίηση ιατρικών δεδομένων από ποικίλες και ετερογενείς πηγές. Αυτό περιλαμβάνει δεδομένα από ιστορικά ασθενών, κλινικές μελέτες, γενετικές πληροφορίες, καθιστώντας εφικτό τον σχηματισμό μιας πλήρους και ολοκληρωμένης εικόνας της ψυχοσωματικής κατάστασης κάθε ασθενούς. Η δυνατότητα να συνδέονται δεδομένα από διαφορετικά συστήματα επιτρέπει στους γιατρούς, τους ερευνητές και τις φαρμακευτικές εταιρείες να έχουν πρόσβαση σε μια πιο ολοκληρωμένη κατανόηση του ασθενή, της πάθησης ή του φαρμάκου, καταφέρνοντας έτσι να γίνονται πιο αποτελεσματικές οι διαγνώσεις και είναι εφικτή η προσαρμογή των θεραπευτικών στρατηγικών στις ατομικές ανάγκες του κάθε ασθενούς. Επιπλέον, μέσω της δυνατότητας διαλειτουργικότητας που προσφέρει ο σημασιολογικός ιστός, επιταχύνεται η συνεργασία μεταξύ νοσοκομείων, ερευνητικών κέντρων και φαρμακευτικών εταιριών. Ο σκοπός αυτής της ανταλλαγής γνώσεων γίνεται για την δημιουργία πιο ολοκληρωμένων και ακριβών θεραπειών. Αυτή η συνεργασία προάγει την εξατομικευμένη ιατρική, όπου οι θεραπείες και οι διαγνωστικές διαδικασίες προσαρμόζονται με ακρίβεια στις ανάγκες του κάθε ασθενούς. Επιπλέον, μέσω της αυτοματοποιημένης συλλογιστικής, αναγνωρίζονται συσχετίσεις που προηγουμένως δεν είχαν βρεθεί, και σχέσεις μεταξύ των δεδομένων που ενδέχεται να είναι δύσκολο να εντοπιστούν με κλασσικές μεθόδους. Έτσι επιτρέπεται την αναγνώριση νέων παρενεργειών σε φάρμακα, είτε θετικών είτε αρνητικών, και τη βελτίωση των θεραπειών μέσα από την κατανόηση των άγνωστων αλληλεπιδράσεων μεταξύ των διαφόρων παραμέτρων της υγείας. Εν τέλει, ο Σημασιολογικός Ιστός συμβάλλει όχι μόνο στη βελτίωση της κλινικής πρακτικής, αλλά και στην ενίσχυση της ιατρικής έρευνας, με ταχύτερη ανακάλυψη νέων θεραπευτικών επιλογών και πιο αποτελεσματική διαχείριση των ασθενών [21].

Ακόμα και ο επιχειρηματικός κόσμος μπορεί να βγει κερδισμένος από τη χρήση και την εξέλιξη του σημασιολογικού ιστού. Μπορεί, ο σημασιολογικός ιστός, να προσφέρει σημαντικές δυνατότητες μέσω της

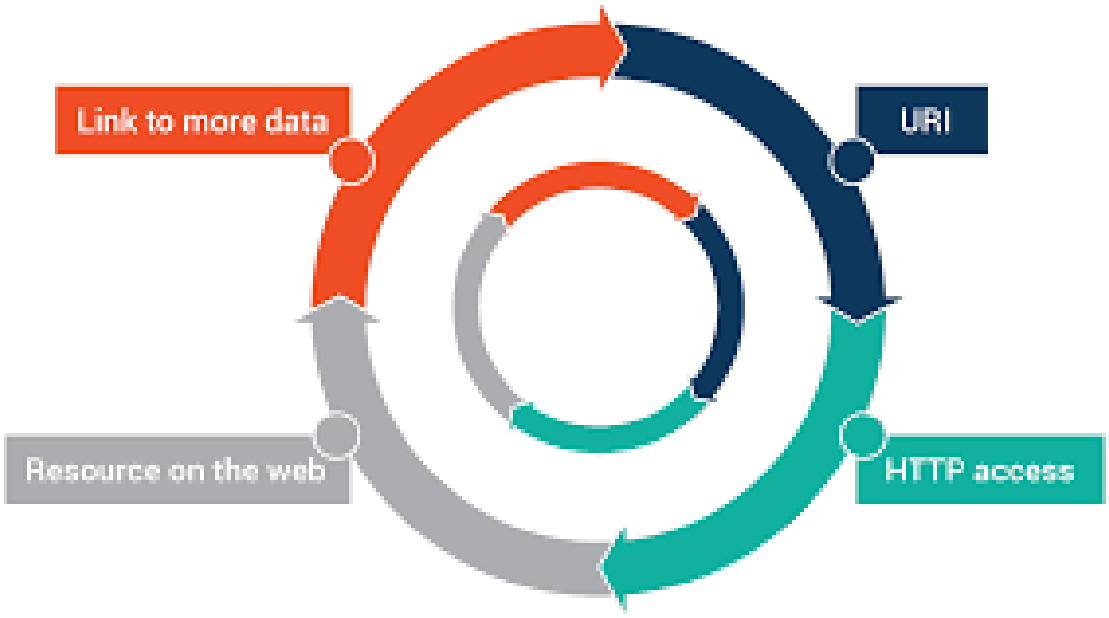
οργάνωσης και της ανάλυσης των δεδομένων. Λόγω των συνδεδεμένων δεδομένων, οι επιχειρήσεις μπορούν να αποκτήσουν βαθύτερη κατανόηση της αγοράς, να βρουν πρότυπα αλλά και να γίνουν προβλέψεις για μελλοντικές τάσεις της αγοράς. Έτσι με αυτόν τον τρόπο επιτρέπεται η πιο στοχευμένη λήψη αποφάσεων, όπως την ανάπτυξη νέων προϊόντων, την αναπροσαρμογή της στρατηγικής μάρκετινγκ αλλά και τη διαχείριση εφοδιαστικής αλυσίδας. Επιπλέον, η αυτοματοποίηση πολύπλοκων διαδικασιών μέσω της διαλειτουργικότητας των συστημάτων και της χρήσης σημασιολογικών τεχνολογιών μειώνει το κόστος και αυξάνει την αποτελεσματικότητα, δίνοντας ανταγωνιστικό πλεονέκτημα στις εταιρείες [22].

1.5 Linked Data και Resource Description Framework (RDF)

Η έννοια του Linked Data και το RDF αποτελούν δύο από τις πιο βασικές έννοιες του σημασιολογικού ιστού. Τα δύο αυτά στοιχεία πρωταγωνιστούν στο ρόλο της διαχείρισης, της διασύνδεσης και της περιγραφής δεδομένων. Μέσω αυτών προσφέρεται η δυνατότητα να δημιουργηθεί ένας ιστός όπου τα δεδομένα έχουν δομή, και είναι εμπλουτισμένα με έννοιες και σημασία. Μέσω των Linked Data και του RDF τα δεδομένα μετατρέπονται σε πολύ πιο προαβάσιμα [23].

1.5.1 Ορισμός Linked Data

Το Linked Data πρόκειται για μια σειρά αρχών και πρακτικών με στόχο να διασυνδεθούν τα δομημένα δεδομένα σε μέγεθος ιστού. Η έννοια αυτή επιτρέπει τη δημοσίευση δεδομένων έτσι ώστε να είναι δυνατή η εύκολη αναζήτηση, πρόσβαση και επεξεργασία όχι μόνο από μηχανές αλλά και από ανθρώπους. Το Linked Data αξιοποιεί τα URI για να μπορέσει να αναγνωρίσει πόρους και το RDF για την σωστή αναπαράσταση των δεδομένων. Με τη χρήση του Linked Data οι πληροφορίες δεν περιορίζονται σε απομονωμένα σύνολα δεδομένων αλλά συνδέονται και με άλλα δεδομένα, δημιουργώντας έτσι ένα νέο δίκτυο γνώσης [24].



σχήμα 1.5: Κύκλος των Linked Data

1.5.2 Βασικές αρχές του Linked Data

Οι βασικές αρχές του Linked Data διατυπώθηκαν από τον ίδιο τον Tim Berners-Lee. Αρχικά, η χρήση URI προκειμένου να αναγνωριστούν οι πόροι. Δηλαδή, κάθε οντότητα ή πληροφορία στον ιστό αποκτά ένα μοναδικό αναγνωριστικό URI, επιτρέποντας την αναφορά και την αναγνώρισή της. Ακόμα, η χρήση HTTP προκειμένου να γίνει η πρόσβαση στα URI. Εννοώντας ότι τα URI πρέπει να είναι προσβάσιμα μέσω του πρωτοκόλλου HTTP, επιτρέποντας την ανάκτηση πληροφοριών. Στη συνέχεια, η παροχή πληροφοριών σε μορφή RDF, είναι

μια από τις βασικές αρχές του Linked Data. Όταν κάποιος ανακτάει ένα URI, πρέπει να επιστρέφονται δομημένα δεδομένα σε μορφή RDF. Πρέπει ακόμα να σιγουρευτεί ο συσχετισμός των δεδομένων με άλλα URIs. Δηλαδή τα δεδομένα πρέπει να περιέχουν συνδέσμους προς άλλα δεδομένα στον ιστό, προκειμένου να δημιουργείται ένα διασυνδεδεμένο δίκτυο πληροφοριών. Αυτές είναι οι βασικές αρχές του Linked Data. Μέσω αυτών μπορούμε να δημιουργήσουμε έναν ανοιχτό ιστό δεδομένων, όπου οι πληροφορίες από διαφορετικές πηγές συνδυάζονται και αξιοποιούνται εύκολα [25].

1.5.3 RDF

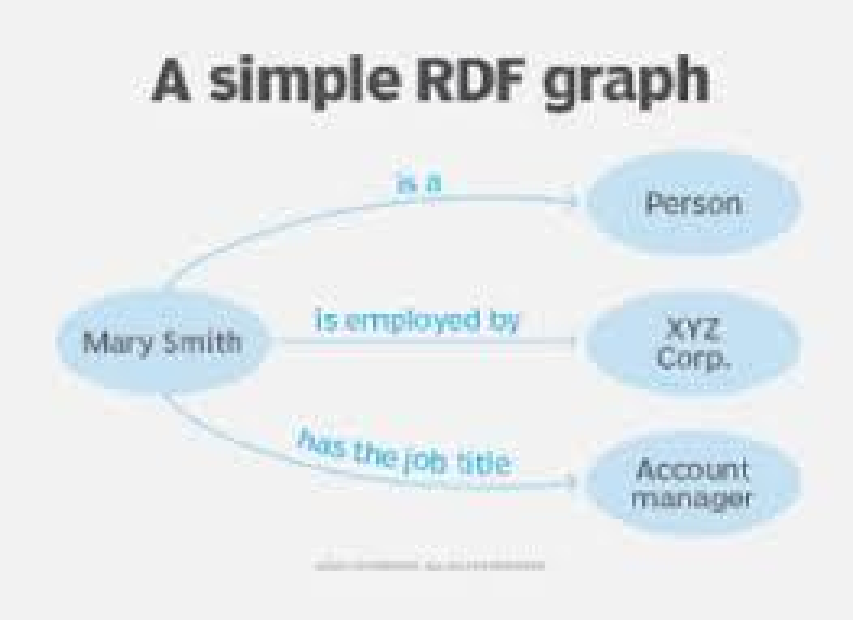
Το RDF είναι ένα πρότυπο για την αναπαράσταση των δεδομένων και την διασύνδεση τους στον σημασιολογικό ιστό. Το RDF βασίζεται στην ιδέα της περιγραφής των πόρων, μέσω μιας τριπλέτας η οποία αποτελείται από:

1. Το Υποκείμενο (Subject): Ο πόρος που περιγράφεται
2. Το ρήμα (Predicate): Η ιδιότητα ή η σχέση που αποδίδεται στο υποκείμενο.
3. Το αντικείμενο (Object): Η τιμή της ιδιότητας ή ένας άλλος πόρος που σχετίζεται με το υποκείμενο.

Μέσω αυτής της τριπλέτας, επιτρέπεται η φομημένη περιγραφή των δεδομένων με απλό και κατανοητό τρόπο. Ένα παράδειγμα ενός RDF σε μορφή τριπλέτας.

- Υποκείμενο: http://example.org/person#John
- Κατηγόρημα: foaf:knows
- Αντικείμενο: http://example.org/person#Mary

Επεξηγηματικά, η παραπάνω τριπλέτα δηλώνει ότι ο άνθρωπος John γνωρίζει την Mary. Χρησιμοποιείται η ιδιότητα foaf:knows από το λεξιλόγιο FOAF(Friend of a Friend).



σχήμα 1.6 Απλό παράδειγμα RDF δεδομένου

1.5.4 Συσχέτιση μεταξύ Linked Data και RDF

Η συσχέτιση μεταξύ των Linked Data και RDF είναι θεμελιώδης. Το RDF αποτελεί την βάση για την υλοποίηση των αρχών του Linked Data. Πιο συγκεκριμένα, το Linked Data ορίζει τις αρχές για τη διασύνδεση και δημοσίευση των δεδομένων στο ιστό. Το RDF παρέχει τον μηχανισμό για την αναπαράσταση των δεδομένων με δομή και σημασιολογικά πλούσια μορφή [26].

Η RDF και η χρήση της επιτρέπει τη δημιουργία τριπλετών που συνδέουν τα δεδομένα. Αυτά μπορούν να αναζητηθούν και να συνδιαστούν μέσω της γλώσσας ερωτηματων SPARQL. Ακόμα, επιτρέπει τη διασύνδεση δεδομένων από διαφορετικές πηγές, κάτι που είναι βασικό για τη δημιουργία Linked Data. Για παράδειγμα, ένα σύνολο δεδομένων για πόλεις μπορεί να περιλαμβάνει τριπλέτες RDF με σκοπό να ορίσουν σχέσεις όπως city:locatedIn country:Italy, δηλαδή πόλεις στην χώρα Ιταλία. Αυτά τα δεδομένα μπορούν να συνδεθούν με σύνολα Linked Data για να παρέχουν περισσότερες πληροφορίες, όπως για παράδειγμα πληροφορίες για την Ιταλία. Έτσι, το Linked Data χρησιμοποιεί το RDF για να δημιουργήσει έναν διασυνδεδεμένο ιστό δεδομένων. Οι πληροφορίες συνδυάζονται και επαναχρησιμοποιούνται με ευκολία από διαφορετικές εφαρμογές και οργανισμούς [27].

1.5.5 Συμπέρασμα

Συμπερασματικά το Linked Data και το RDF συνεργάζονται αρμονικά. Ο σκοπός τους είναι να διασυνδέσουναι και να διαχειριστούν τα δομημένα δεδομένα στο σημασιολογικό ιστό. Οι βασικές αρχές του Linked Data καθορίζουν την δημοσίευση διασυνδεδεμένων δεδομένων, ενώ το RDF παρέχει το αναγκαίο πλαίσιο για την αναπαράστασή τους. Αυτό επιτρέπει την δημιουργία ενός καθολικού δικτύου γνώσης, όπου τα δεδομένα είναι προσβάσιμα, έχουν σημασία και μπορούμε να τα αναζητήσουμε, με αποτέλεσμα να είναι εφικτή η ανάπτυξη πιο έξυπνων και διαλειτουργικών εφαρμογών.

1.6 Η SPARQL

Η SPARQL πρόκειται για την γλώσσα ερωτημάτων που χρησιμοποιείται στο σημασιολογικό ιστό. Πρόκειται για μια πανίσχυρη και ευέλικτη γλώσσα ερωτημάτων που χρησιμοποιείται για την αναζήτηση και την ανάκτηση δεδομένων που βρίσκονται σε μορφή RDF. Έκανε την πρώτη της εμφάνιση από τον οργανισμό W3C το έτος 2008 με σκοπό να γίνει η παγκόσμια γλώσσα ερωτημάτων για εγγραφές σε RDF, δίνοντας τη δυνατότητα να ενώσει και να συνδυάσει πληροφορίες από διαφορετικές πηγές και επίσης μπορεί να επιστρέψει τα αποτελέσματα σε πολλές και διαφορετικές μορφές [28]. Με βάση τα παραπάνω, θα εξεταστούν αναλυτικότερα αυτά τα χαρακτηριστικά της SPARQL, τα οποία την καθιστούν απαραίτητο εργαλείο για την ενασχόληση με δεδομένα τύπου RDF [29].

```
SELECT *
WHERE{
    ?item wdt:P31 wd:Q3305213 .
    ?item wdt:P170 wd:Q34661 .
    ?item wdt:P1476 ?title .
    ?item wdt:P18 ?image .
    ?item wdt:P571 ?inception .
    SERVICE wikibase:label {
        bd:serviceParam
        wikibase:language "[AUTO_LANGUAGE],en".
    }
}
LIMIT 15
```

σχήμα 1.7 Παράδειγμα Σύνταξης Sparql

Αρχικά όπως προαναφέρθηκε, μέσω της SPARQL υποστηρίζετε η πολυμορφία των δεδομένων. Πιο συγκεκριμένα μπορεί να λειτουργήσει με δεδομένα αποθηκευμένα σε διαφορετικά φορμάτ. Ένα από τα πιο σύνηθες RDF/XML, βασίζεται σε XML για την αναπαράσταση των RDF. Ένα τυπικό παράδειγμα RDF/XML αρχείου περιλαμβάνει την εξής δομή:

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

<rdf:Description rdf:about="http://example.org/resource">

<example:property xmlns:example="http://example.org/ontology">Value</example:property>

</rdf:Description>

</rdf:RDF>

Αυτή η μορφή αρχείου δείχνει την χρήση των τριπλετών RDF σε XML. Πιο συγκεκριμένα το <rdf:RDF> υποδηλώνει το βασικό(root) στοιχείο του εγγράφου RDF/XML. Στην συνέχεια το xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" δηλώνει το χώρο των ονομάτων(namespace) RDF, το οποίο είναι βασικό και απαραίτητο για την σωστή ανάγνωση των RDF στοιχείων. Η χρήση του <rdf:Description> είναι να περιγράφει έναν πόρο (resource). Το χαρακτηριστικό rdf:about="http://example.org/resource" αναφέρεται στην συγκεκριμένη διεύθυνση (URI) του πόρου που περιγράφηκε. Το στοιχείο <example:property> προσδίδει μια ιδιότητα(Predicate) στο πόρο, και xmlns:example="http://example.org/ontology" αποδίδει τον χώρο των ονομάτων όπου ανήκει αυτή η ιδιότητα. Και τέλος η τιμή του στοιχείου Value είναι το Object της τριπλέτας τύπου RDF. Συνοψίζει την τιμή που αποδίδεται στην ιδιότητα example:property για το συγκεκριμένο resource.

Η δομή της παραπάνω τριπλέτας τύπου RDF αποδομείται ώς εξής. Το Subject της τριπλέτας είναι http://example.org/resource. Το Predicate είναι http://example.org/ontology/property, και το Object είναι το Value. Έτσι αναπαριστά μια RDF τριπλέτα σε XML, και καταφέρετε να περιγράψει

πληροφορίες στον σημασιολογικό ιστό.

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
>
  <owl:DatatypeProperty rdf:about="http://example.org/autology#ref">
    <rdfs:domain rdf:resource="https://www.buildingsmart-tech.org/
                               ifcXML/IFC4/final#Entity"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#IDREF"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="http://example.org/autology#href">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#anyURI"/>
    <rdfs:domain rdf:resource="https://www.buildingsmart-tech.org/
                               ifcXML/IFC4/final#Entity"/>
  </owl:DatatypeProperty>
  <owl:Class rdf:about="https://www.buildingsmart-tech.org/
                        ifcXML/IFC4/final#Entity"/>
</rdf:RDF>
```

σχήμα 1.8 RDF/XML παράδειγμα

Μια ακόμη μορφή φορμάτ ονομάζεται Turtle, και πρόκειται για ένα πιο ευαναγνωστο φορμάτ αποτύπωσεις RDF τριπλετών. Το φορμάτ Turtle χρησιμοποιεί μια απλούστερη σύνταξη για να το καταφέρει αυτό, καταφέρνοντας έτσι να είναι ιδιαίτερα χρήσιμο για την ανθρώπινη κατανόηση μιας τριπλέτας.

```
@prefix query: <http://purl.org/queryall/query:> .
@prefix provider: <http://purl.org/queryall/provider:> .
@prefix profile: <http://purl.org/queryall/profile:> .
@prefix : <http://example.org/> .

:myquery a query:Query ;
   query:inputRegex "(.*)" ;
   profile:profileIncludeExcludeOrder
      profile:excludeThenInclude .

:myprovider a provider:Provider ;
   provider:resolutionStrategy provider:proxy ;
   provider:resolutionMethod provider:httpgeturl ;
   provider:isDefaultSource "true"^^<http://www.w3.org/2001/
XMLSchema#boolean> ;
   provider:endpointUrl "http://myhost.org/${input_1}" ;
   provider:includedInQuery :myquery ;
   profile:profileIncludeExcludeOrder
      profile:excludeThenIncludae .
```

σχήμα 1.9 Παράδειγμα Turtle

Ένα παράδειγμα Turtle μπορεί να είναι το εξής:

@prefix ex: <http://example.org/ontology/> .

<http://example.org/resource> ex:property "Value" .

Μονομιάς ο καθένας μπορεί να καταλάβει ότι η αριθμός των γραμμών, που σε σχέση με το παράδειγμα RDF/XML είναι λιγότερος, το καθιστά πολύ πιο ευανάγνωστο. Συγκεκριμένα, @prefix υποδηλώνει έναν προθέτη, δηλαδή, προκειμένου να μειώση την επανάληψη μακριών URIs. Στο παραπάνω παράδειγμα το ex: ανιστοιχεί στο URI http://example.org/ontology/, και μπορεί να χρησιμοποιηθεί μόνο ο προθέτης ex προκειμένου να ονοματίσουμε το παραπάνω URI. Περαιτέρω, το Subject είναι http://example.org/resource , αντιπροσωπεύει τον πόρο που περιγράφεται, ακριβώς όπως το προηγούμενο παράδειγμα. Το predicate ,στην προκειμένη περίπτωση, είναι το ex:property, και εδώ είναι εμφανής η χρήση του προθέτη. Ακόμα και στην προκειμένη περίπτωση που είναι μόνο ένας ο προθέτης και μόνο μια η γραμμή που χρησιμοποιείται, μπορεί κάποιος να αντιληφθεί το πόσο πιο ευανάγνωστο είναι. Συνεχίζοντας το Object παραμένει το "Value" και εξακολουθεί να είναι η τιμή που συνδέεται με την ιδιότητα ex:property.
Σε σύγκριση με το RDF/XML, το φορμάτ Turtle είναι πιο συνοπτικό και φιλικό προς τον άνθρωπο, ειδικά για άτομα που έχουν μια εξοικείωση με δεδομένα τύπου RDF. Ακόμα υπάρχει ένα συνήθεις φορματ αποτύπωσεις των τριπλετών εν ονόματη N-Triples. Πρόκειται για ένα φορμάτ που βασίζεται σε γραμμές κειμένου, όπου η κάθε γραμμή αντιπροσωπεύει μια τριάδα. Αυτός ο τρόπος χρησιμοποιείται, συχνότερα, για την αποθήκευση ή την αποστολή μεγάλων σύνολο δεδομένων. Ένα απλό παράδειγμα μιας N-triple είναι:

<http://example.org/alice> <http://example.org/hasAge> "30" .

<http://example.org/alice> <http://example.org/livesIn> <http://example.org/NewYork> .

Το Subject στο παραπάνω παράδειγμα είναι <http://example.org/alice>. Ο υποκείμενος πόρος που αναφέρεται πρόκειται για την alice. Δηλαδή είναι ένα URI που προσδιορίζει την 'alice'. Όπως φαίνεται, και οι δύο γραμμές έχουν κοινό Object. Το predicate, στην πρώτη γραμμή είναι http://example.org/hasAge, και περιγράφει κάποιο χαρακτηριστικό, ή προσδίδει την ιδιότητα. Στην συγκεκριμένη γραμμή είναι hasAge και μεταφράζεται ώς "Έχει ηλικία", άρα είναι χαρακτηριστικό της Alice που συνδέει την ηλικία της με την τιμή της. Παράλληλα στην δεύτερη γραμμή το Predicate είναι <http://example.org/livesIn> . Σε αυτή την το Predicate δηλώνει την σχέση της alice με τον τόπο όπου ζει, δηλαδή "κατοικεί", και το διασυνδέει με την πόλη την οποία μένει. Το Object στην πρώτη γραμμή είναι το "30", και υποδηλώνει την ηλικία της alice που είναι 30 ετών. Στην δεύτερη γραμμή, το Object είναι <http://example.org/NewYork> και ταυτόχρονα πρόκειται για ένα URI, και αναφερεται στην Νέα Υόρκη, όπου είναι ο τόπος που διανέμει. Άρα, μέσω τον δύο αυτών γραμμών, μπορούμε και συνθέτουμε την εξής πληροφορία: Η alice είναι 30 ετών και Η alice ζει στην Νέα Υόρκη.

Αυτοί είναι τρεις από τους πιο συνήθεις τρόπους αναπαραστάσεις RDF δεδομένων. Σχετικά με το φορματ RDF/XML, βασίζεται στο XML, και παρέχει έναν τυποποιημένο τρόπο ανταλλαγής RDF δεδομένων, αλλά καθιστάται δυσκολότερη στην ανάγνωσει από ανθρώπους λόγω της πολυπλοκότητας της σύνταξης του XML. Από την άλλη το φορμάτ Turtle παρέχει μια πιο ανθρώπινη προσέγγιση στο θέμα της αναγνωσιμότητας, αφού είναι μια πιο συμπυκνωμένη μορφή αποτύπωσεις των δεδομένων RDF, και καταφέρνει έτσι να είναι πιο φιλική προς τον χρήστη. Τέλος, τα N-Triples είναι ένα απλό φορμάτ που καταγράφει τα δεδομένα RDF ως μια σειρά από τριπλέτες, και παρέχει μια λιτή

αναπαράσταση, χωρίς να απαιτεί προηγμένες γνώσεις για την σύνταξή της. Είναι ειδικά διαμορφωμένη για την ανταλλαγή μεγάλων συνόλων δεδομένων και την αποθήκευσή τους. Κάθε μια από αυτές τις μορφές έχει τα πλεονεκτήματα και τα μειονεκτήματά της, αλλά η SPARQL μπορεί και παρέχει την ικανότητα να εργάζεται με την πολυμορφία των δεδομένων.

Ένα ακόμα βασικό χαρακτηριστικό της SPARQL είναι αυτό της επεκτασιμότητας και της διαλειτουργικότητας που παρέχει, δίνοντας την δυνατότητα να της εύκολης ενσωμάτωσης και ανταλλαγής δεδομένων από διαφορετικές πηγές και διαφορετικά συστήματα.

Μέσω της SPARQL μπορεί να επιτευχθεί η διαλειτουργικότητα με διάφορους μηχανισμούς. Το γεγονός ότι πρόκειται για μια ενιαία γλώσσα ερωτημάτων την καθιστά αμέσως ικανεί να καταφέρει την διαλειτουργικότητα σε διάφορα συστήματα και διαφορετικές πηγές. Αυτό συμβαίνει διεθνές πρότυπα και κανόνες, που έχουν εισαχθεί από την W3C, όσο αναφορά την εκτέλεση ερωτημάτα σε δεδομένα τύπου RDF. Η γλώσσα SPARQL είναι κοινή για όλες της βάσεις δεδομένων τύπου RDF, και οποιοδήποτε σύστημα υποστηρίζει την γλώσσα ερωτημάτων SPARQL μπορεί να διασυνδεθεί με κάθε ένα άλλο σύστημα που χρησιμοποιεί τα ίδια πρότυπα RDF. Έτσι προσφέρεται ένα κοινό πλαίσιο για την αλληλεπίδραση με άλλα δεδομένα.

Ακόμα η διαλειτουργικότητα γίνεται πραγματικότητα μέσω της χρήσης των ανοικτών προτύπων. Πρότυπα όπως του RDF και το OWL για την αναπαράσταση και την περιγραφή των δεδομένων, επιτρέπει την ενοποιήσει δεδομένων από εξωτερικές πηγές, ακόμα και όταν αυτές έχουν διαφορετικά σχήματα και μοντέλα δεδομένων. Ένα απλό παράδειγμα αναπαράστασης δεδομένων με πρότυπο OWL είναι :

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

        xmlns:owl="http://www.w3.org/2002/07/owl#">


  <!-- Οντολογία για Θηλαστικά και Κατοικίδια -->

  <owl:Ontology rdf:about="http://example.org/ontology/animals"/>


  <!-- Ορισμός Κλάσεων -->

  <owl:Class rdf:about="#Animal"/>

  <owl:Class rdf:about="#Mammal">

    <rdfs:subClassOf rdf:resource="#Animal"/>

  </owl:Class>

  <owl:Class rdf:about="#Pet">

    <rdfs:subClassOf rdf:resource="#Animal"/>
```

```
</owl:Class>

<!-- Ιδιότητες -->
<owl:ObjectProperty rdf:about="#hasOwner">
  <rdfs:domain rdf:resource="#Pet"/>
  <rdfs:range rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
</owl:ObjectProperty>

<!-- Ιδιότητα κατοικίδιου ζώου -->
<owl:DatatypeProperty rdf:about="#isDomesticated">
  <rdfs:domain rdf:resource="#Animal"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
</owl:DatatypeProperty>

<!-- Άτομα -->
<rdf:Description rdf:about="#Dog">
  <rdf:type rdf:resource="#Mammal"/>
  <rdf:type rdf:resource="#Pet"/>
  <isDomesticated>true</isDomesticated>
</rdf:Description>

<rdf:Description rdf:about="#Cat">
  <rdf:type rdf:resource="#Mammal"/>
  <rdf:type rdf:resource="#Pet"/>
  <isDomesticated>true</isDomesticated>
</rdf:Description>
```

</rdf:RDF>

Στο συγκεκριμένο παράδειγμα δημιουργούμε τις κλάσεις Animal, Mammal και την κλάση Pet. Οι κλάσεις Mammal και Pet είναι Animal. Προσδίδονται και οι ιδιότητες hasOwner στο κατοικίδιο Pet, ο οποίος είναι τύπου foaf:Person, ακόμα προσδίδεται η ιδιότητα isDomesticated είναι μια λογική ιδιότητα, αληθής ή ψευδής, που εξηγεί αν ένα ζώο είναι εξημερωμένο ή όχι. Τέλος ορίζονται και δύο άτομα ένα Dog και ένα Cat τα οποία είναι Mammals και Pets, και τα δύο είναι εξημερωμένα, και πηγάζει το συμπέρασμα αυτό από την ιδιότητα isDomesticated που είναι true, αληθής. Αυτός ο τρόπος αναπαράστασης των δεδομένων μπορεί να ανακτηθούν και να επεξεργαστούν από την γλώσσα ερωτημάτων SPARQL. Έτσι τα δεδομένα από εξωτερικές πηγές ή συστήματα ενοποιούνται και μπορεί να γίνει ερώτηση σε αυτά, λόγω του κοινού συνόλου κανόνων, που παρέχει η γλώσσα ερωτημάτων SPARQL.

Η SPARQL υποστηρίζει την δυνατότητα εξωτερικών συνδέσεων ή "ομοσπονδιακών ερωτημάτων", περισσότερα γνωστά ως Federated Queries. Επιτρέπει στους χρήστες να εκτελούν ερωτήματα που διασυνδέει δεδομένα από πολλές πηγές. Αυτό γίνεται πραγματικότητα μέσω της δυνατότητας αναφοράς σε εξωτερικά endpoints ή server SPARQL σε ένα μόνο ερωτήματα. Ας υποθέσουμε ένα σενάριο με δύο διαφορετικές βάσεις δεδομένων. Η πρώτη βάση δεδομένων περιέχει πληροφορίες για βιβλία και για τους συγγραφείς των εν λόγω βιβλίων. Στην δεύτερη βάση δεδομένων περιέχεται πληροφορίες για τους συγγραφείς και την χώρα την οποία γεννήθηκαν. Ο στόχος του σεναρίου είναι να βρούμε τα βιβλία από την πρώτη βάση δεδομένων, σε συνδυασμό μαζί με την χώρα που γεννήθηκε ο συγγραφέας από την δεύτερη βάση δεδομένων. Παρακάτω παρατίθεται ένα snippet κώδικα SPARQL.

```
PREFIX ex: <http://example.org/schema/>

PREFIX dbo: <http://dbpedia.org/ontology/>


SELECT ?book ?author ?country WHERE {

  SERVICE <http://repositoryA.org/sparql> {

    ?book ex:hasAuthor ?author .

  }

  SERVICE <http://repositoryB.org/sparql> {

    ?author dbo:birthPlace ?country .

  }

}
```

Παρακάτω θα εξηγηθεί περαιτέρω η σύνταξη της SPARQL, αλλά το αποτέλεσμα αυτής της ερώτησης είναι, ο συνδυασμός των δεδομένων από τις δύο βάσεις δεδομένων και καταφέρνει να επιστρέφει την λίστα βιβλίων μαζί με την χώρα όπου γεννήθηκε ο συγγραφέας. Αυτό είναι ένα παράδειγμα που πραγματοποιείται διαλειτουργικότητα, μέσω της SPARQL, αφού καταφέρνουμε να πάρουμε δεδομένα

από δύο διαφορετικές πηγές.

Ένα ακόμα τρόπος που η SPARQL διασφαλίζει την διαλειτουργικότητα, είναι η μετατροπή δεδομένων. Η SPARQL δίνει την δυνατότητα να χρησιμοποιούθουναι εργαλεία που μετατρέπουν τα δεδομένα (data transformation tools). Για παράδειγμα R2RML μετατρέπει δεδομένα από μη RDF σχήματα , όπως μια βάση δεδομένων SQL, σε μορφή RDF. Ένα παράδειγμα χρήσεις R2RML είναι:

Έστω πως υπάρχει μια βάση δεδομένων SQL και περιέχεται σε αυτών ένας πίνακας Books με τις παρακάτω στήλες

1. BookID: Μοναδικό αναγνωριστικό του βιβλίου.
2. Title: Τίτλος του βιβλίου.
3. Author: Συγγραφέας του βιβλίου.

Και ο σκοπός είναι να μετατραπούν τα δεδομένα σε RDF, σύμφωνα με το μοντέλο.

1. Τα βιβλία ως οντότητες με URL.
2. Το Title και το Author ως ιδιότητες.

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .

@prefix ex: <http://example.org/schema/> .

@prefix d: <http://example.org/data/> .


# Mapping για τον πίνακα Books

<#BooksMapping> a rr:TriplesMap ;

   rr:logicalTable [ rr:tableName "Books" ] ;


   # Δημιουργία των URIs για κάθε βιβλίο

   rr:subjectMap [

      rr:template "http://example.org/book/{BookID}" ;

      rr:class ex:Book ;

   ] ;


   # Ιδιότητα Title

   rr:predicateObjectMap [

      rr:predicate ex:title ;
```

```
    rr:objectMap [ rr:column "Title" ] ;
] ;

# Ιδιότητα Author
rr:predicateObjectMap [
    rr:predicate ex:author ;
    rr:objectMap [ rr:column "Author" ] ;
] .
```

Επεξηγηματικά, το rr:logicalTable δηλώνει τον πίνακα Books από την βάση δεδομένων SQL, ως πηγη δεδομένων. Το rr:subjectMap καθορίζει ότι το URI για κάθε μια εγγραφή από τον πίνακα Books χρησιμοποιεί το BookID, αλλά και ότι το URI κατασκευάζεται με το πρότυπο http://example.org/book/{BookID}. Η χρησιμότητα του rr:predicateObjectMap, είναι να ορίζει τις ιδιότητες ex:title και ex:author για κάθε μια εγγραφή όπου τα δεδομένα αντλούνται από τις στήλες Title και Author αντίστοιχα. Με αυτόν τό τρόπο παράγονται δεδομένα σε μορφή RDF. Έτσι επιτυγχάνεται η διαλειτουργικότητα.

Ένας ακόμα τρόπος που γίνεται πραγματικότητα η διαλειτουργικότητα, είναι με την χρήση των οντολογιών. Μέσω αυτών ορίζονται οι έννοιες και οι σχέσεις των δεδομένων. Η κατηγοριοποίηση και οι κοινοί ορισμοί δεδομένων, επιτρέπουν σε διαφορετικά συστήματα να ανταλλάσουν δεδομένα και να διασυνδέοντε αποτελεσματικά. Όπως προαναφέρθηκε η SPARQL είναι ικανή να επεξεργαστεί αυτές τις οντολογίες, που η σημασία της είναι άκρως σημαντική στην ύπαρξης της διαλειτουργικότητας.

Τέλος η χρήση των ανοικτών προτύπων όπως το RDF για την αναπαράσταση και περιγραφή των δεδομένων, επιτρέπει την ενοποίηση των δεδομένων από εξωτερικές πηγές, ακόμα και αν τα σχήματα που ακολουθούνται από τα δεδομένα είναι διαφορετικά. Μέσω της SPARQL γίνεται εφικτό να κάνουμε ερωτήσεις από διάφορους παρόχους και να ενοποιηθούν, αφού το σύνολο των κανόνων που χρησιμοποιούνται είναι κοινή για όλους.

Ένα ακόμα από τα βασικά χαρακτηριστικά που παρέχει η χρήση της SPARQL είναι η ικανότητα της επεκτασιμότητας. Η επεκτασιμότητα ονομάζουμε την ικανότητα ενός συστήματος, μιας τεχνολογίας ή μιας εφαρομής να προσαρμόζετε όταν αλλάζει το μέγεθος και η πολυπλοκότητα των δεδομένων που επεξεργάζονται από αυτήν. Αυτό συμβαίνει όταν αυξάνεται ο αριθμός των χρηστών ή οι απαιτήσεις των συστημάτων.

Ένας από τους τρόπους που καταφέρνεται η επεκτασιμότητα μέσω της SPARQL είναι, το γεγονός ότι η SPARQL υποστηρίζει την χρήση των ομοσπονδιακών ερωτημάτων. Επιτρέπεται έτσι η αναζήτηση και η συλλογή δεδομένων από διαφορετικές πηγές RDF ταυτόχρονα. Έτσι αντί να χρειάζεται μια μοναδική κεντρική βάση δεδομένων, οι εφαρμογές μπορούν να ρωτούν διάφορες βάσεις δεδομένων σε πραγματικό χρόνο. Ένα παράδειγμα κώδικα SPARQL είναι:

```
SELECT ?book ?author ?price WHERE {
  SERVICE <http://example1.org/sparql> {
```

```
    ?book ex:hasAuthor ?author .
  }
  SERVICE <http://example2.org/sparql> {
    ?book ex:hasPrice ?price .
  }
}
```

Ακόμα ένας τρόπος που επιτυγχάνεται η επεκτασιμότητα είναι το γεγονός ότι τα περισσότερα συστήματα SPARQL διαθέτουν βελτιστοποιητες ερωτημάτων. Ο σκοπός των βελτιστοποιητών ερωτημάτων είναι αναλύσουνε το ερώτημα και να κάνουν καλύτερο το τρόπο εκτέλεσης τους. Οι βελτιστοποιητες έχουν την ικανότητα να προσαρμόζονται ανάλογα με τον όγκο των δεδομένων και την αρχιτεκτονική της βάσης δεδομένων, κάνοντας μικρότερο τον χρόνο που χρειάζεται για την εκτέλεση και την χρήση των πόρων. Έστω ένα παράδειγμα:

Υπάρχει ένα σύνολο RDF δεδομένων που περιέχει δεδομένα για βιβλία, συγγραφείς και κατηγορίες. Το σύνολο έχει δεκάδες χιλιάδες εγγραφές και η βάση SPARQL υποστηρίζει την βελτιστοποίηση ερωτημάτων. Έτσι θα έμοιαζε μια η δομή δεδομένων:

```
@prefix ex: <http://example.org/schema/> .

# Τριπλές για βιβλία
<http://example.org/book1> ex:hasTitle "SPARQL Basics" ;
                           ex:hasAuthor <http://example.org/author1> ;
                           ex:inCategory <http://example.org/category1> .

<http://example.org/book2> ex:hasTitle "Advanced SPARQL" ;
                           ex:hasAuthor <http://example.org/author2> ;
                           ex:inCategory <http://example.org/category2> .

# Τριπλές για συγγραφείς
<http://example.org/author1> ex:hasName "Alice" .
<http://example.org/author2> ex:hasName "Bob" .
```

```
# Τριπλές για κατηγορίες

<http://example.org/category1> ex:categoryName "Data Science" .

<http://example.org/category2> ex:categoryName "Semantic Web" .
```

Παρακάτω υπάρχει ένα snippet κώδικα , δίχως την χρήση της βελτιστοποίησης:

```
PREFIX ex: <http://example.org/schema/>

SELECT ?bookTitle ?authorName WHERE {

  ?book ex:hasTitle ?bookTitle .

  ?book ex:hasAuthor ?author .

  ?author ex:hasName ?authorName .

  ?book ex:inCategory ?category .

  ?category ex:categoryName "Data Science" .

}
```

Παρακάτω υπάρχει ένα snippet κώδικα , με την χρήση της βελτιστοποίησης:

```
PREFIX ex: <http://example.org/schema/>

SELECT ?bookTitle ?authorName WHERE {

  ?category ex:categoryName "Data Science" .

  ?book ex:inCategory ?category .

  ?book ex:hasTitle ?bookTitle .

  ?book ex:hasAuthor ?author .

  ?author ex:hasName ?authorName .

}
```

Ποιες είναι οι αλλαγές μεταξύ των δύο snippets. Αρχικά ο περιορισμός “?category ex:categoryName "Data Science"” Εκτελείται πρώτος. Μονο οι σχετικές κατηγορίες και βιβλία φιλτράρονται πριν αυτές συνδυαστούν με τους συγγραφείς. Ο αριθμός των ενδιάμεσων δεδομένων μειώνεται δραστικα.

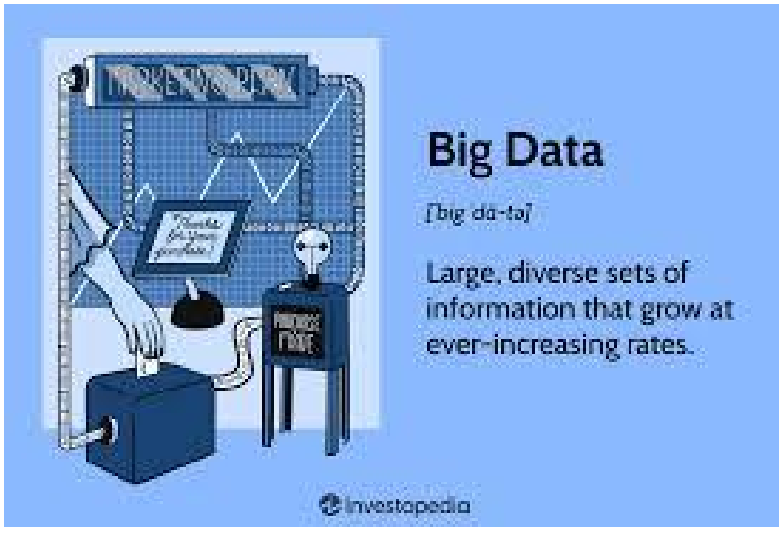
Μια βάση δεδομένων, όπως το Virtuoso, δίνει την δυνατότητα να εφαρμόσει αυτόματα την αναδιάταξη των συνθηκών της ερώτησης. Αυτό γίνεται δυνατό με τα ευρετήρια τριπλετών, χρησιμοποιώντας προκαθορισμένα ευρετήρια για πιο ταχύς αναζήτηση στις τριπλέτες. Αλλά επίσης υπολογίζεται το κόστος εκτέλεσης κάθε μέρους του ερωτήματος και επιλέγεται η πιο αποδοτική σειρά. Έτσι η

βελτιστοποίηση μειώνει τον χρόνο επεξεργασίας και οι εφαρμογές μπορούν να διαχειρίζονται μεγαλύτερους όγκους δεδομένων. Με αυτόν τον τρόπο επιτυγχάνεται η επεκτασιμότητα των εφαρμογών.

Η η χρήση ευρετηρίων, πιο γνωστό ως index, πρόκειται για έναν κρίσιμο μηχανισμό όσον αφορά το θέμα της επεκτασιμότητας, και στην γρηγορότερη εκτέλεση ερωτημάτων SPARQL. Γίνεται πιο εμφανείς αυτό όταν οι βάσεις δεδομένων όταν οι τριπλέτες σε αυτή, ξεπερνούν σε αριθμό εκατομμύρια ή δισεκατομμύρια εγγραφές. Το indexing πρόκειται για την διαδικασία δημιουργίας μιας δομής των δεδομένων που βοηθάει στην ταχύτερη πρόσβαση σε συγκεκριμένα δεδομένα, χωρίς να χρειάζεται να γίνει αναζήτηση σε όλη την βάση δεδομένων. Υπάρχουν πέντε τύποι ευρετηριών.

Αρχικά υπάρχουν τα Ευρετήρια τριπλετών. Αυτές οι RDF τριπλέτες αποθηκεύονται σε όλα τα πιθανά σχήματα ταξινόμησης. Δηλαδή, ταξινομούνται SPO (Subject, Predicate, Object), POS (Predicate, Object, Subject) και OSP (Object, Subject, Predicate). Με αυτό τον τρόπο επιτρέπετε ταχύτερη εύρεση συγκεκριμένων δεδομένων ανάλογα με το ποια ήταν η ερώτηση. Στην συνέχεια υπάρχουν τα Ευρετήρια Bitmap. Χρησιμοποιεί δυαδικές αναπαραστάστάσεις προκειμένου να αποθηκεύσει τα δεδομένα. Κάθε χαρακτηριστικό αντιστοιχεί σε ένα bit, όπου το 1 υποδηλώνει ότι υπάρχει μια τριπλέτα που επιβεβαιώνει το χαρακτηριστικό αυτό. Είναι ιδιαίτερα αποδοτικό για το φιλτράρισμα των δεδομένων και την εκτέλεση πιο σύνθετων ερωτήσεων, που έχουν πολλές συνθήκες. Τα ευρετήρια γραφημάτων, εστιάζουν στην δομή του RDF γράφου. Είναι ιδανικό για την εκτέλεση ερωτημάτων που βασίζονται σε σχέσεις μεταξύ οντοτήτων, για παράδειγμα εύρεση όλων των κόμβων που συνδέονται με ένα συγκεκριμένο κόμβο. Συνεχίζοντας έχουμε τα προθετικά ευρετήρια. Χρησιμοποιούνται για την ταχύτερη αναζήτηση URIs ή λέξεων που αρχίζουν με συγκεκριμένο πρόθεμα. Είναι χρήσιμο όταν τα ερωτήματα περιλαμβάνουν strings ή συγκεκριμένες namespaces. Και τέλος, τα στατιστικά ευρετήρια. Κάποιες βάσεις δεδομένων, για παράδειγμα το Virtuoso, δημιουργούν στατιστικά στοιχεία για τις τριπλέτες.

Ένας ακόμα τρόπος που επιτυγχάνεται η επεκτασιμότητα είναι, η υποστήριξη τεχνολογιών BIG DATA που παρέχει η SPARQL. Με τον όρο BIG DATA ονομάζουμε τα πολύ μεγάλα και σύνθετα δεδομένα που δεν είναι δυνατόν να επεξεργαστούν ή να αναλυθούν με παραδοσιακούς τρόπους, μεθόδους και εργαλεία επεξεργασίας δεδομένων. Το γεγονός ότι η SPARQL μπορεί να ενσωματωθεί με συστήματα όπως, Apache Jena TDB, Blazegraph , Virtuoso και Hadoop μπορεί να επεξεργαστεί και να αναλυθούν BIG DATA.



σχήμα 1.10 Σύντομος Ορισμός του Big Data

Πιο συγκεκριμένα το Apache Jena TDB πρόκειται για μια βιβλιοθήκη και βάση δεδομένων που υποστηρίζεται η διαχείριση RDF δεδομένων και την εκτέλεση SPARQL ερωτημάτων. Πιο αναλυτικά, επιτρέπεται η τοπική αποθήκευση RDF δεδομένων, η χρήση ευρετηρίων σε όλες τις μορφές τους SPO, POS και OSP, υποστηρίζετε η SPARQL και κυρίως είναι σχεδιασμένη για τοπική χρήση με έμφαση στην απόδοση. Η τεχνολογία Apache Jena TDB είναι ιδανική για την ανάπτυξη, το testing για μικρομεσαίων εφαρμογών που δεν απαιτούν επεξεργασία σε κλίμακα.

Από την άλλη το Blazegraph είναι μια δημοφιλής RDF βάση δεδομένων που υποστηρίζει την SPARQL και παρέχει την δυνατότητα να κάνει επεξεργασία σε κλίμακα. Είναι κατάλληλο για εφαρμογές που χρειάζονται υψηλή απόδοση όπως συστήματα συστήματος ή ανάλυσης συνδεδεμένων δεδομένων. Η πιο γνωστή χρήση που γίνεται η Blazegraph είναι αυτή της Wikidata. Τα χαρακτηριστικά που την κάνουν ιδανική για να πληρή τα προηγούμενα ικανότητες είναι το γεγονός ότι μπορεί και χρησιμοποιεί τεχνικές Scale-out, επιτρέποντας την προσθήκη κόμβων για την διαχείριση BIG DATA και μπορεί και χρησιμοποιεί την κάρτα γραφικών, βελτιώνοντας την απόδοση της σε πολύπλοκα ερωτήματα. Επίσης είναι σχεδιασμένο για μεγάλα γραφήματα RDF και μπορεί να υποστηρίξει δισεκατομμύρια τριπλέτες.

Ακόμα το Virtuoso είναι ένα από τα πιο συνήθεις συστήματα διαχείρισης δεδομένων τύπου RDF. Χρησιμοποιείται για εφαρμογές συνδεδεμένων δεδομένων και δημόσιες βάσεις δεδομένων. Η πιο γνωστή από αυτές είναι η DBPedia. Το Virtuoso συνδιάζει λειτουργίες SQL αλλά και SPARQL. Παρόμοια με το Blazegraph υποστηρίζονται τεχνικές για Scale-out. Δίνεται όμως και η δυνατότητα της συμπίεσης των δεδομένων προκειμένο να επιτευχθεί η μειωμένη ανάγκη πόρων. Τέλος δίνετε μπορεί να εφαρμοστούν οι στατιστικές βελτιστοποιητές για την πιο αποδοτική στρατηγική εκτέλεσης ερωτημάτων.

Τέλος δίνετε και η δυνατότητα της χρήσης SPARQL πάνω από Hadoop, έτσι γίνεται εφικτή η επεξεργασία των BIG DATA. Η επεξεργασία κλίμακας γίνετε μέσω του Hadoop HDFS προκειμένου να αποθηκευτούν τα δεδομένα τύπου RDF αλλά και την παράλληλη επεξεργασία μέσω MapReduce. Υπάρχουν διάφορα frameworks, που επιτρέπουν την εκτέλεση SPARQL πάνω από Hadoop. Και τέλος το γεγονός ότι υποστηρίζετε η σύνθεση δεδομένων από διάφορες πηγές την καθιστά κατάλληλη για αναλύσεις BIG DATA όπου τα RDF δεδομένα συνδυάζονται και με άλλες μορφές δεδομένων, όπως για παράδειγμα JSON ή CSV.

Κάθε μια από αυτές τις τεχνολογίες έχουν διαφορετικές δυνατότητες και περιορισμούς. Η επιλογή εξαρτάται ανάλογα με την ανάγκη που προκύπτει από το μέγεθος των δεδομένων, την ανάλυσης που χρειάζονται και της απαιτούμενης απόδοσης. Πάραυτα, με την χρήση αυτών τεχνολογιών μπορεί να επιτευχθεί η επεκτασιμότητα.

Ακόμα ένας από τους τρόπους η επεκτασιμότητα γίνεται εφικτή μέσω της χρησης της τεχνολογίας SPARQL, είναι το γεγονός ότι υποστηρίζεται η Incremental Updates. Δηλαδή, η SPARQL υποστηρίζει την προσθαφαίρεση δεδομένων μέσω SPARQL UPDATE, δίχως να χρειάζεται οι βάσεις δεδομένων να επαναφορτόσουν μεγάλους όγκους δεδομένων, ενημερώνοντάς τα σε πραγματικός χρόνο. Ένα παράδειγμα ενός snippet:

```
INSERT DATA {

  <http://example.org/book3> ex:hasAuthor "John Doe" .

}
```

Η επεκτασιμότητα καταφέρνετε και μέσω των κατανεμημένων περιβαλλόντων, πιο γνωστά ως cloud computing. Με αυτόν τον τρόπο προσφέρονται πολλά πλεονεκτήματα για εφαρμογές μεγάλης κλίμακας. Ο τρόπος που μπορεί και λειτουργεί το cloud computing είναι μέσω της κατανεμημένης αποθήκευσης, δηλαδή αποθηκεύονται τα RDF δεδομένα σε κατανεμημένα συστήματα αρχείων, και έτσι επιτρέπεται η διατήρηση τεράστιων γραφημάτων, οι οποιοί δεν μπορούν να χωρέσουν σε έναν μόνον server. Παράλληλα υπάρχει και η κατανεμημένη επεξεργασία, όπου τα ερωτήματα σπάνε σε μικρότερα κομμάτια και να επεξεργαστούν παράλληλα σε πολλούς κόμβους. Με την χρήση του Cloud computing, μπορεί να επιτευχθεί η κλιμακωσιμότητα. Με τον όρο κλιμακωσιμότητα ονομάζουμε την ικανότητα ενός συστήματος να αυξομειώνεται αυτόματα οι υποδομές που χρησιμοποιούνται ανάλογα με τις απαιτήσεις. Ένα μικρό παράδειγμα είναι η χρήση περισσότερων κόμβων όταν υπάρχει μεγαλύτερος όγκος ερωτημάτων ή τα δεδομένα αυξάνονται. Τέλος, ελαχιστοποιείται η επικοινωνία μεταξύ κόμβων, με την χρήση τοπικής αποθήκευσης δεδομένων για να μειωθεί ο χρόνος μετάδοσης.

Τα πλεονεκτήματα της SPARQL στο cloud computing είναι πολλά. Μερικά από αυτά είναι η Υποστήριξη Μεγάλων Γραφημάτων, δηλαδή, είναι ικανά να αποθηκεύουν δισεκατομμύρια τριπλέτες τύπου RDF. Η απόδοση του συστήματος είναι αυξημένη, και καταφέρνετε αυτό μέσω των παράλληλων εκτελέσεων πολύπλοκων ερωτημάτων. Η χρήση της δυναμικής κλιμάκωσης. δηλαδή η ικανότητα να μεταβάλετε η χρήση των υποδομών ανάλογα με τον φόρτο της εργασίας. Είναι αρκετά ευέλικτο στην υλοποίηση αφού υπάρχουν αρκετές ιδιωτικές εταιρείες που παρέχουν αυτήν την δυνατότητα, όπως Amazon και Google. Και τέλος η χρήση σύνθετων ερωτήσεων από πολλαπλές πηγές το καθιστούν κατάλληλο για την επιτευξη της επεκτασιμότητας.

Τέλος ένα από τα πιο βασικές πτυχές που επιτυγχάνεται η επεκτασιμότητα είναι το γεγονός ότι υποστηρίζονται τα εξωτερικά δεδομένα μέσω συνδεδεμένων δεδομένων. Μέσω της SPARQL δίνετε πρόσβαση σε εξωτερικά σύνολα δεδομένων και γίνεται εφικτή η ενσωμάτωσή τους, ενισχύοντας την κλιμακωσιμότητα, την διαλειτουργικότητα και την επεκτασιμότητα.
Με τους παραπάνω τρόπους καταφέρνετε η επεκτασιμότητα και η διαλειτουργικότητα, δύο πολύ σημαντικές πτυχές της SPARQL. Επιτρέπουν την αποτελεσματική διαχείριση και αξιοποίηση μεγάλων όγκων δεδομένων σε διαφορετικά περιβάλλοντα, την ενσωματωση ετερογενών πηγών πληροφοριών. Προωθείτε η συνεργασία μεταξύ συστημάτων που βασίζονται σε διαφορετικά δεδομένα. Εν ολίγοις, η SPARQL είναι ένα πανίσχυρο εργαλείο για την κατασκευή εφαρμογών και συστημάτων που χρειάζεται ανακάλυψη γνώσης, διασύνδεση δεδομένων και ανάλυση αυτών.

1.7 Η Σύνταξη της SPARQL.

Η SPARQL είναι μια εξειδικευμένη γλώσσα ερωτημάτων που σχεδιάστηκε για την αναζήτηση και την διαχείριση των δεδομένων που είναι αποθηκευμένα σε μορφή RDF. Παρομοίως με τις παραδοσιακές γλώσσες ερωτημάτων για σχεσιακες βάσεις δεδομένων, όπως η SQL, η SPARQL επιτρέπει την εξαγωγή και διαχειρισμό των δεδομένων. Το ιδιαίτερο χαρακτηριστικό της SPARQL είναι η ικανότητα να εκτελεί ερωτήματα σε γραφήματα δεδομένων, τα οποία είναι δομημένα ώς υποκείμενο-ρήμα-αντικείμενο, πιο γνωστό ως subject-predicate-object.

Η σύνταξη της SPARQL είναι δομημένη και ευέλικτη. Επιτρέπεται, σε αυτήν, η εκτέλεση απλών και πολυσύνθετων ερωτημάτων που περιλαμβάνουν επιλογές δεδομένων, φιλτραρισματα, ταξινόμησης και συνένωση πληροφοριών από διαφορετικές πηγές. Λόγω της χρήσης μοτίβων γραφήματος, μπορεί και εντοπίζονται δεδομένα που ανταποκρίνονται στα κριτήρια που επιβεβαιώνουν τα διασυνδεδεμένα δεδομένα με αποτέλεσμα να είναι ιδανικό για την δημιουργία εφαρμογών στο σημασιολογικό ιστό.

Η εντολή SELECT στην SPARQL αποτελεί τον ακρογωνιαίο λίθο της εκτέλεσης ερωτημάτων. Ο

σκοπός της εντολής SELECT είναι η ανάκτηση δεδομένων από μια RDF βάση δεδομένων. Χρησιμοποιείται για την συγκεκριμένη επιλογή μεταβλητών που περιέχουν πληροφορίες από την RDF τριπλέτα. Το αποτέλεσμα που επιστρέφετε είναι σε μορφή πίνακα. Γενικά μοιάζει με την σύνταξη της SELECT στην γλώσσα ερωτημάτων SQL.

Ένα παράδειγμα της εντολής SELECT:

```
SELECT ?person ?name

WHERE {

  ?person rdf:type foaf:Person .

  ?person foaf:name ?name .

}
```

Ο σκοπός της παραπάνω ερώτησης είναι, το παραπάνω ερώτημα ανακτά όλους τους πόρους τύπου foaf:Person και τα ονόματά τους (foaf:name).

Επεξηγηματικά, η βασική δομή της εντολής SELECT περιλαμβάνει τρια διαφορετικά κομμάτια.

1. Μεταβλητές: Οι μεταβλητές που θέλουμε να ανακτήσουμε προσδιορίζονται με το πρόθεμα ?.
2. WHERE clause: Χρησιμοποιείται για τον καθορισμό των μοτίβων γραφήματος που πρέπει να ταιριάξουν τα δεδομένα.
3. Προαιρετικά φίλτρα: Μέσω της εντολής FILTER, μπορούμε να περιορίσουμε τα αποτελέσματα.

Υπάρχουν ακόμα τρόποι ώστε να επεκταθεί η χρήση της SELECT. Ένας τρόπος είναι LIMIT και OFFSET.
Παράδειγμα:

```
SELECT ?person ?name

WHERE {

  ?person rdf:type foaf:Person .

  ?person foaf:name ?name .

}

LIMIT 10 OFFSET 5
```

Ένας από του βασικούς λόγους χρήσης του OFFSET και LIMIT είναι ο περιορισμός του πλήθους των αποτελεσμάτων. Ένα RDF γράφημα αποτελεσμάτων μπορεί να περιέχει τεράστιο όγκο δεδομένων, και η ανάκτηση των αποτελεσμάτων μπορεί να αποβεί χρονοβόρα αλλά και να καταναλώνει υπερβολικούς πόρους. Με το LIMIT, μπορούμε να περιορίσουμε τον αριθμό των επιστρεφόμενων εγγραφών, καταφερνοντας να γίνει έτσι πιο αποδοτικό. Για παράδειγμα, έστω ότι έχουμε εκατομμύρια

επιστρεφόμενες εγγραφές, με την χρήση του LIMIT 10 δίνουμε την οδηγία να επιστραφούν μόνο οι 10 πρώτες εγγραφές, μειόνοντας την χρήση μνήμης και τον χρόνο που χρειάζεται.

Η OFFSET χρησιμοποιείται σε συνδυασμό με το LIMIT για να επιστρέψει σελιδοποιημένα τα δεδομένα. Αυτό είναι χρήσιμο για εφαρμογές, που απαιτούν να εμφανιστούν τα δεδομένα σε σελίδες, όπως οι ιστότοποι. Για παράδειγμα:

Για την προβολή αποτελεσμάτων σελίδας:

- Πρώτη σελίδα: LIMIT 10 OFFSET 0
- Δεύτερη σελίδα: LIMIT 10 OFFSET 10
- Τρίτη σελίδα: LIMIT 10 OFFSET 20

Μέσω αυτής της τεχνικής διασφαλίζεται ότι οι χρήστες μπορούν να περιηγηθούν σε τεράστια σύνολα δεδομένων εύκολα και αποδοτικά.

Σε βάση δεδομένων με μεγάλη ζήτηση, η επιστροφή όλων των αποτελεσμάτων μπορεί να υπερφόρτωση τον server. Μέσω της χρήσης του LIMIT και OFFSET μειώνεται ο αριθμός των δεδομένων που επεξεργάζεται ο server ανά ερωτήματα, προσφεροντας πιο σταθερή απόδοση.

Όταν ο χρήστης χρειάζεται να αναζητήση μόνο ένα μικρό τμήμα ενός γραφήματος, για παράδειγμα τις 50 πιο πρόσφατες εγγραφές, το LIMIT είναι εξαιρετικά χρήσιμο προκειμένου να επιστραφούν μόνο οι σχετικές πληροφορίες. Για παράδειγμα:

```
SELECT ?book ?author

WHERE {

  ?book rdf:type ex:Book .

  ?book ex:hasAuthor ?author .

}

ORDER BY DESC(?book)

LIMIT 50
```

Στο συγκεκριμένο παράδειγμα, ανακτούμε μόνο τα 50 πιο πρόσφατα βιβλία.

Ακόμα μέσω της χρήσης του LIMIT και OFFSET μπορούμε να παρέχουμε μια καλύτερη εμπειρία στο χρήστη. Οι εφαρμογές που αλληλεπιδρούν με τον τελικό χρήστη, όταν εμφανίζονται όλα τα αποτελέσματα ταυτόχρονα μπορεί να είναι δυσνόητη. Για παράδειγμα, αν ο χρήστης κάνει μια ερώτηση και επιστραφούν 200000 εγγραφές, ο χρήστης δεν θα μπορεί να διαβάσει τα αποτελέσματα. Ένω αν κάνει μια σταδιακή παρουσίαση των αποτελεσμάτων, ο χρήστης θα μπορέσει να διαβάσει τα αποτελέσματα.

Συμπερασματικά, το LIMIT και OFFSET είναι απαραίτητα για την βελτιστοποίηση των ερωτημάτων SPARQL, εξασφαλίζοντας την αποδοτικότητα, την ευκολία στην χρήση αλλά και καλύτερη διαχείριση

μεγάλων συνόλων δεδομένων. Μέσω της χρήσεις του LIMIT και OFFSET επιτρέπει στις εφαρμογές να είναι κλιμακούμενες και πιο ευέλικτες κάνοντας την SPARQL κατάλληλη για πολυσύνθετα περιβάλλοντα δεδομένων.

Μια ακόμη εντολή στη SPARQL είναι αυτή της DISTINCT. Η εντολή DISTINCT, χρησιμοποιείται με σκοπό να μην εμφανίζονται διπλότυπες εγγραφές, στα αποτελέσματα ενός ερωτήματος. Είναι ιδιαίτερη σημαντική σε περιπτώσεις που η βάση δεδομένων περιέχει πλεονάζουσες πληροφορίες, και με την χρήση της DISTINCT καταφέρετε η ανάκτηση που συμβαίνει να είναι αυτή μοναδικών τιμών. Για παράδειγμα:

```
SELECT DISTINCT ?name

WHERE {

  ?person foaf:name ?name .

}
```

Επεξηγηματικά, είναι εμφανές ότι το DISTINCT βρίσκεται στην αρχή της εντολής SELECT και πριν από τις μεταβλητές. Αυτό που καταφέρνει αυτή η εντολή είναι να εξαγει όλες τις τιμές της μεταβλητής ?name από το RDF γράφημα. Οι διπλότυπες εγγραφές απορρίπτονται, διασφαλίζοντας ότι κάθε όνομα, ?name εμφανίζεται μόνο μία φορά στην επιστροφή των αποτελεσματων.

Η χρήση του DISTINCT καταφέρνει την αποφυγεί πλεονάζοντας δεδομένα. Οι βάσεις δεδομένων μπορεί να περιέχουν διπλότυπες πληροφορίες για διαφορετικούς λόγους. Ένας από αυτούς είναι η επανάληψη πόρων ή πλεονασματικές τριπλέτες. Ακόμα βελτιστοποιείται η ανάλυση των ερωτημάτων. Είναι ιδανικό για ερωτήματα που κάνουν στατιστική ή αναλυτική επεξεργασία των δεδομένων, αφού μειώνει τον όγκο που πρεπει να επεξεργαστεί, εστιάζοντας στις μοναδικές τιμές. Τέλος σε συνδυασμό με άλλες εντολές όπως η GROUP BY επιτρέπεται η δημιουργία συνοπτικών αναφορών με μοναδικά αποτελέσματα.

Μια ακόμη εντολή όπου χρησιμοποιείται στην SPARQL, είναι αυτή της ORDER BY. Ο ρόλος της είναι ταξινομεί τα αποτελέσματα ενός ερωτήματος βάσει μιας ή περισσότερων μεταβλητών. Είναι απαραίτητη όταν προσπαθούμε να παρουσιάσουμε δεδομένα με μια σειρά, π.χ. αλφαβητική, αριθμητική ή χρονική. Ένα παράδειγμα σύνταξης της εντολής ORDER BY είναι:

```
SELECT ?name

WHERE {

  ?person foaf:name ?name .

}

ORDER BY ASC(?name)
```

Επεξηγηματικά το παραπάνω ερώτημα επιστρέφει όλα τα ονοματα ταξινομημένα αλφαβητικά σε αύξουσα σειρά. Το γεγονός ότι είναι σε αύξουσα σειρά μπορούμε να το καταλάβουμε από ASC(?name). Σε αντίθεση περίπτωση, αν θέλαμε κάνουμε φθίνουσα σειρά, θα χρησιμοποιήσουμε την

DESC().

Δίνεται η δυνατότητα να γίνουν σύνθετες ταξινομησης. Δηλαδή να γίνουν ταξινομήσεις βάσει πολλών κριτηρίων. Για παράδειγμα:

```
SELECT ?name ?age

WHERE {

  ?person foaf:name ?name .

  ?person foaf:age ?age .

}

ORDER BY ASC(?age) DESC(?name)
```

Επεξηγηματικά, η ταξινόμηση, γενικά, εφαρμόζεται με την σειρά που υποδηλώνονται στα κριτήρια. Δηλαδή τα δεδομένα ταξινομούνται πρώτα κατά ηλικία σε αύξουσα σειρά, εάν δύο άτομα έχουν την ίδια ηλικία, τότε ταξινομούνται κατα όνομα σε φθίνουσα σειρά. Ένα από τα πλεονεκτήματα που προσφέρει η χρήση των σύνθετων ταξινομήσεων είναι το γεγονός ότι η παρουσίαση των δεδομένων είναι βελτιωμένη. Διευκολύνει την ανάλυση των δεδομένων, για παράδειγμα η ταξινόμηση βάσει ημερομηνία ή τιμής μπορεί να βοηθήσει στην δημιουργία αναφορών. Ακόμα μπορεί να εφαρμοστεί σε όλους τους τύπους δεδομένων RDF και συνδυάζεται με άλλες εντολές όπως το OFFSET και LIMIT για να επιστρέψουν ταξινομημένα υποσύνολα δεδομένων. Τέλος μπορεί να καθοριστεί η προτεραιότητα στην ιεράρχηση των αποτελεσμάτων, βοηθώντας να βρεθούν στον εντοπισμό των σημαντικότερων δεδομένων. Εν κατακλείδι η εντολή ORDER BY είναι απαραίτητο εργαλείο στην SPARQL για την οργάνωση των αποτελεσμάτων.

Μια τελευταία λειτουργία στην εντολή SELECT είναι η GROUP BY. Ο ρόλος της είναι να ομαδοποιεί τα αποτελέσματα των ερωτημάτων και την εκτέλεση πράξεων όπως COUNT, SUM, AVG. Για παράδειγμα :

```
SELECT ?city (COUNT(?person) AS ?population)

WHERE {

  ?person dbo:birthPlace ?city .

}

GROUP BY ?city

HAVING (?population > 1000)
```

Επεξηγηματικά, η παραπάνω ερώτηση αναζητά πόλεις (?city) όπου άτομα (?person) έχουν γεννηθεί. Το COUNT χρησιμοποιείται για να μετρήσει τον αριθμό των ατόμων που έχουν γεννηθεί σε κάθε πόλη. Η GROUP BY ?city ομαδοποιεί τα αποτελέσματα ανά πόλη. Ενώ η HAVING (?population > 1000) φιλτράρει τις πόλεις ώστε να επιστραφούν μόνο όσες έχουν πληθυσμό πάνω από 1000 άτομα.

Η GROUP BY μπορεί να συνδυαστεί με άλλες εντολές. Μια από αυτές είναι η ORDER BY. Για

παράδειγμα:

```
SELECT ?city (COUNT(?person) AS ?population)

WHERE {

  ?person dbo:birthPlace ?city .

}

GROUP BY ?city

ORDER BY DESC(?population)
```

Το παραπάνω ερώτημα ταξινομεί τις πόλεις κατά πληθυσμό σε φθίνουσα σειρά.

Ακόμα μπορεί να συνδυαστεί με την εντολή LIMIT προκειμένου να περιοριστεί ο αριθμός των ομαδοποιημένων αποτελεσμάτων. Για παράδειγμα :

```
SELECT ?city (COUNT(?person) AS ?population)

WHERE {

  ?person dbo:birthPlace ?city .

}

GROUP BY ?city

ORDER BY DESC(?population)

LIMIT 5
```

Επεξηγηματικά, επιστρέφει τις 5 πόλεις με τον μεγαλύτερο πληθυσμό.

Συμπερασματικά η εντολή GROUP BY και οι αθροιστικές λειτουργίες είναι απαραίτητα εργαλεία στη SPARQL για την ανάλυση και ομαδοποίηση δεδομένων RDF. Παρέχετε ευελιξία και δύναμη για την συγκέντρωση στατιστικών στοιχείων και εξαγωγή δεδομένων που δεν είναι προφανείς στα αρχικά δεδομένα. Η σωστή χρήση μπορεί να προσφέρει βαθύτερη κατανόηση των δεδομένων.

Δύο ακόμα εντολές της SPARQL είναι DESCRIBE και CONSTRUCT. Πρόκειται για δύο εργαλεία εξαγωγής και αναπαράστασης δεδομένων από RDF γραφήματα με διαφορετικούς τρόπους. Χρησιμοποιούνται για την ανάκτηση πληροφοριών, διαφέρουν μεταξύ τους ως προς την λειτουργικότητά τους, παρέχοντας συγκεκριμένες δυνατότητες για συγκεκριμένες περιπτώσεις χρήσης.

Αρχικά η εντολή DESCRIBE χρησιμοποιείται για να επιστρέψει μια περιγραφή ενός πόρου ή πόρων από ένα σύνολο RDF δεδομένων. Η περιγραφή που επιστρέφεται καθορίζετε από την πηγή SPARQL ή την εφαρμογή, και συνήθως αντιστοιχεί σε τριπλέτα που είναι σχετικά με τον πόρο. Αυτό είναι χρήσιμο για την ανακάλυψη δεδομένων όταν δεν γνωρίζετε εξ αρχής η δομή της βάσεις. Για

παράδειγμα:

```
DESCRIBE ?person

WHERE {

?person rdf:type foaf:Person .

}
```

Επεξηγηματικά, ο στόχος του παραπάνω ερωτήματος είναι να επιστρέψει πληροφορίες για όλους τους πόρους που τύπου Person. Η απάντηση περιλαμβάνει όλες τις τριπλέτες που έχουν ως subject τον ?person ή και άλλα σχετικά δεδομένα ανάλογα με την πηγή SPARQL. Ένα από τα πιο σημαντικά χαρακτηριστικά είναι η ευελιξία στην περιγραφή, αφού η απάντηση που δέχεται ο χρήστης εξαρτάται από την υλοποίηση της endpoint. Αλλά ακόμα ένα από τα πιο βασικά χαρακτηριστικά είναι ότι μέσω του DESCRIBE μπορούμε να ανακαλύψουμε και να εξερευνήσουμε δεδομένα όταν δεν ξέρουμε εξ αρχής τις ιδιότητες των πόρων.

Συνεχίζοντας, η εντολή CONSTRUCT επιτρέπει την δημιουργία RDF γραφήματος εξ ολοκλήρου βασισμένο στα ήδη υπάρχοντα δεδομένα. Ο κάθε χρήστης έχει την δυνατότητα να καθορίσει ένα μοτίβο, πιο γνωστό template, που περιγράφει την δομή του νέου γραφήματος. Αυτό την καθιστά ιδανική την μετασχηματιστική αναπαράσταση των δεδομένων η και την εξαγωγή συγκεκριμένων τριπλετών. Ένα παράδειγμα της CONSTRUCT είναι:

```
CONSTRUCT {

  ?person foaf:name ?name .

  ?person foaf:age ?age .

}

WHERE {

  ?person rdf:type foaf:Person .

  ?person foaf:name ?name .

  OPTIONAL { ?person foaf:age ?age . }

}
```

Επεξηγηματικά, το παραπάνω ερώτημα δημιουργεί ένα νέο RDF γράφημα που περιέχει τριπλέτες μόνο για το όνομα και την ηλικία των πόρων τύπου foaf:Person. Ταυτόχρονα, αν δεν υπάρχει πληροφορία για την ηλικία foaf:age, δεν επιστρέφεται η αντιστοιχεί τριπλέτα. Αυτό είναι πολύ χρήσιμο γιατί δημιουργεί εξατομικευμένη RDF γραφήματα για εξαγωγή ή επεξεργασία των δεδομένων. Μέσω αυτού η αναπαράσταση των δεδομένων είναι πολύ πιο ευέλικτη, και να προσαρμόζεται στις ανάγκες του χρήστη. Μέσω του OPTIONAL επιτρέπει την δημιουργία πλήρων ή ελλιπών τριπλετών.

Εν κατακλείδι οι εντολές DESCRIBE και CONSTRUCT είναι ισχυρά εργαλεία της SPARQL για την ανάκτηση και την διαχείριση δεδομένων RDF. Το DESCRIBE είναι απλό και χρήσιμο για ανακάλυψη

δεδομένων, ενώ η χρήση του CONSTRUCT παρέχει την ευελιξία, την μετασχηματιστική επεξεργασία και την δημιουργία νέων γράφημάτων. Είναι σημαντικό να γίνεται η σωστή επιλογή της κατάλληλης εντολής όπου αυτή εξαρτάται από τις ανάγκες του ερωτήματος, και του σκοπού που αποσκοπεί αυτή κάθε φορά.

Μια ακόμη επέκταση της SPARQL είναι αυτή των FILTER. Η εντολή FILTER χρησιμοποιείται για τον περιορισμο αποτελεσμάτων ενός ερωτήματος βάσει συγκεκριμένων συνθηκών. Θυμίζει την εντολή WHERE από την γλώσσα ερωτημάτων SQL και μπορεί να εφαρμόσει λογικές εκφράσεις, συγκρισεις μεταξύ αριθμών, Strings ή ακόμα και κανονικές εκφράσεις. Ένα παράδειγμα:

```
SELECT ?name ?age

WHERE {

  ?person foaf:name ?name .

  ?person foaf:age ?age .

  FILTER (?age > 30)

}
```

Επεξηγηματικά η εντολή FILTER εφαρμόζει τη συνθήκη ?age > 30. Τα αποτελέσματα που επιστρέφονται είναι μόνο εκείνα τα οποία η ηλικία είναι μεγαλύτερη από 30.

Μια από τις χρήσεις της εντολής FILTER είναι να περιορίζει τις αριθμητικές τιμές. Αυτά τα δεδομένα φιλτράρονται με βάση τιμές, όπως για παράδειγμα η ηλικία, οι ημερομηνίες ή διάφορες τιμές. Μπορούν να χρησιμοποιηθούν κανονικές εκφράσεις, ιδανικό για την εύρεση τιμών που ταιριάζουν σε συγκεκριμένα μοτίβα. Για παράδειγμα:

```
FILTER regex(?name, "^A", "i")
```

Επεξηγηματικά το παραπάνω παράδειγμα επιστρέφει μόνο ονόματα που ξεκινούν με το κεφαλαίο γράμμα Α.

Μπορούμε να κάνουμε και συνδυασμούς λογικών των συνθηκών. Επιτρέπεται η χρήση την λογικής έκφρασης AND/OR. Για παράδειγμα:

```
FILTER (?age > 30 && ?age < 50)
```

Επεξηγηματικά επιστρέφονται μόνο τα δεδομένα με ηλικία μεγαλύτερη του 30 και μικρότερη του 50.

Η εντολή OPTIONAL χρησιμοποιείται για να συμπεριληφθούν προαιρετικά δεδομένα στα αποτελέσματα ενός ερωτήματος. Σε περίπτωση που τα προαιρετικά δεδομένα δεν υπάρχουν για κάποια οντότητα, το αποτέλεσμα επιστρέφεται κανονικά, αλλά η μεταβλητή που σχετίζεται με το OPTIONAL θα έχει κενή τιμή. Για παράδειγμα:

```
SELECT ?name ?email
```

```
WHERE {
  ?person foaf:name ?name .
  OPTIONAL { ?person foaf:mbox ?email . }
}
```

Επεξηγηματικά, στο παραπάνω ερώτημα επιστρέφετε το όνομα κάθε ατόμου από το γράφημα. Η OPTIONAL δίνει την δυνατότητα να επιστραφούν και το email, εάν αυτό υπάρχει. Σε περιπτωση που ένα άτομο δεν έχει email στο γράφημα, το πεδίο email παραμένει null.

Ο λόγος που χρησιμοποιείται η OPTIONAL είναι για να εμπλουτιστούν τα αποτελέσματα. Αρκετά χρήσιμο όταν κάποια δεδομένα είναι προαιρετικά ή τα δεδομένα είναι ελλειπής και θέλουμε να πάρουμε τις οντότητες αυτές. Εξασφαλίζεται ότι η απουσία ενός κομματιού μιας τριπλέτας δεν αποκλείει το αποτέλεσμα.

Μια ακόμη εντολή είναι αυτή της UNION. Ο ρόλος της UNION είναι να συνενώσει διαφορετικά πρότυπα ερωτημάτων. Παρέχετε μέσω αυτής η δυνατότητα εκτέλεσης πολλαπλών μοτίβων, όπου τα διαφορετικά αποτελέσματα συνενωνονται σε ένα ενιαίο σύνολο. Για παράδειγμα:

```
SELECT ?name ?location
WHERE {
  { ?person foaf:name ?name . ?person dbo:birthPlace ?location . }
  UNION
  { ?person foaf:name ?name . ?person dbo:deathPlace ?location . }
}
```

Επεξηγηματικά, η UNION συνδυάζει δύο διαφορετικά μοτίβα. Το πρώτο μοτίβο επιστρέφει την τοποθεσίας γέννησης ενός ατόμου, birthPlace. Ενώ το δεύτερο μοτίβο επιστρέφει την τοποθεσία θανάτου ενός ατόμου, deathPlace. Τα αποτελέσματα αυτά συνενωνονται και παρουσιάζονται μαζί.

Η χρήση της UNION είναι να παραχθούν αποτελέσματα που προέρχονται από διαφορετικά μοτίβα ή πηγές. Επιτρέπεται μέσω αυτής της εντολής να συνδιάζονται δεδομένα που σχετίζονται με διαφορετικές ιδιότητες ή σχέσεις. Έτσι μπορούμε να εκτελέσουμε διαφορετικά ερωτήματα μόνο με ένα ερώτημα.

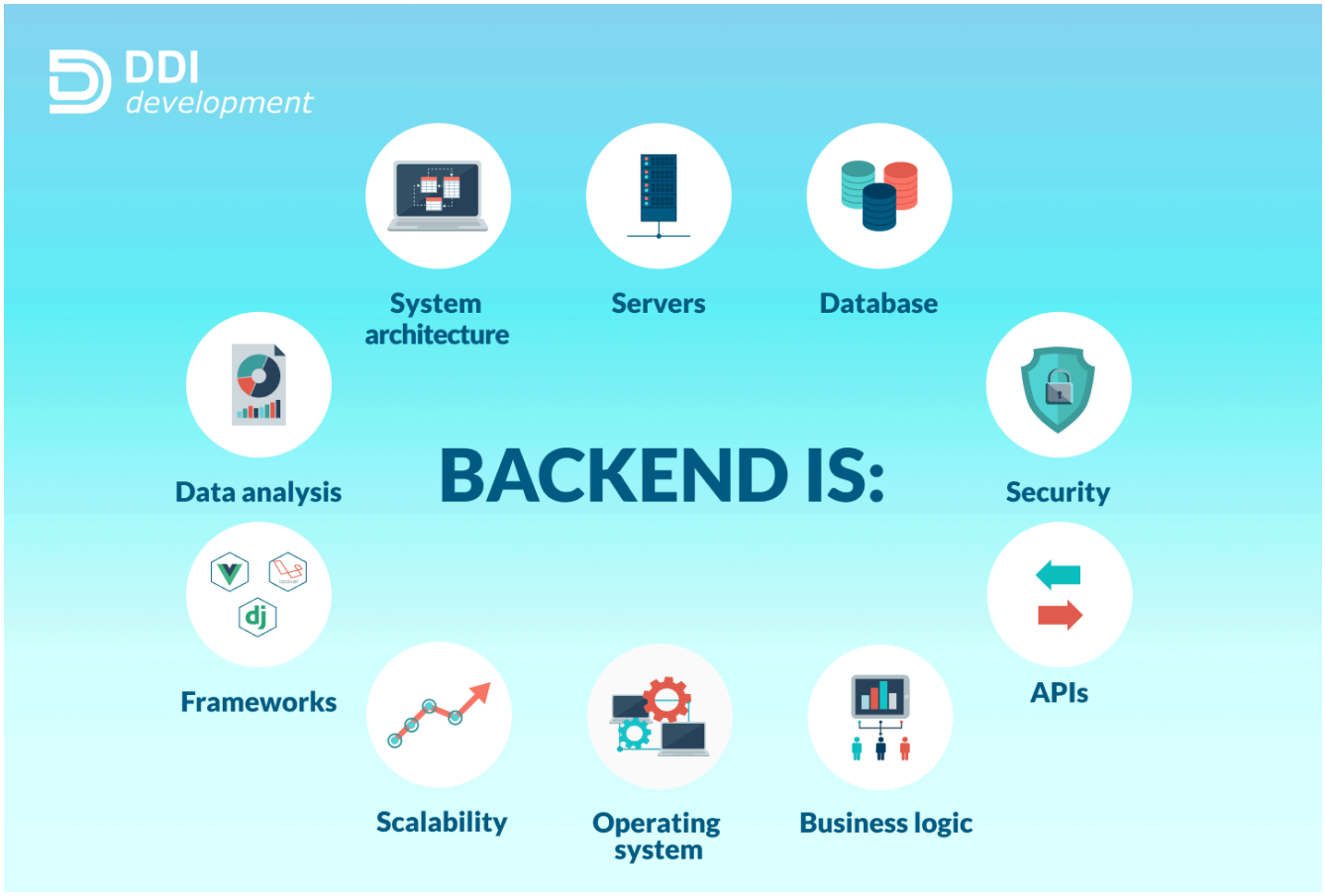
1.8 Συμπέρασμα

Εν κατακλείδι η σύνταξη της SPARQL είναι ειδικά σχεδιασμένη για να προσφέρει μια ισχυρή και ευέλικτη πρόσβαση σε δεδομένα τύπου RDF. Η SPARQL επιτρέπει ερωτήσεις που μπορούν να ανακτήσουν, να φιλτράρουν, να συνενώσουν και να επεξεργαστούν διασυνδεδεμένα δεδομένα στο

σημασιολογικό ιστό. Οι κεντρικοί πυλώνες των ερωτημάτων στην SPARQL είναι η SELECT, ASK, CONSTRUCT και DESCRIBE. Μέσω αυτών μπορεί να καλυφθεί ένα μεγάλο φάσμα αναγκών. Ταυτόχρονα οι FILTER, OPTIONAL και UNION αυξάνουν την ευελιξία και την προσαρμοστικότητα των ερωτημάτων. Εν τέλει η σύνταξη της SPARQL συνδιάζει την απλότητα και την έκφραση, επιτρέποντας να εκτελούνται ερωτήσεις με ακρίβεια και αποδοτικότητα. Πρόκειται για ένα θεμελιώδες εργαλείο για την ανάκτηση και αξιοποίηση της πληροφορίας.

Κεφάλαιο 2ο: Back-end, NestJS

2.1 Εισαγωγή



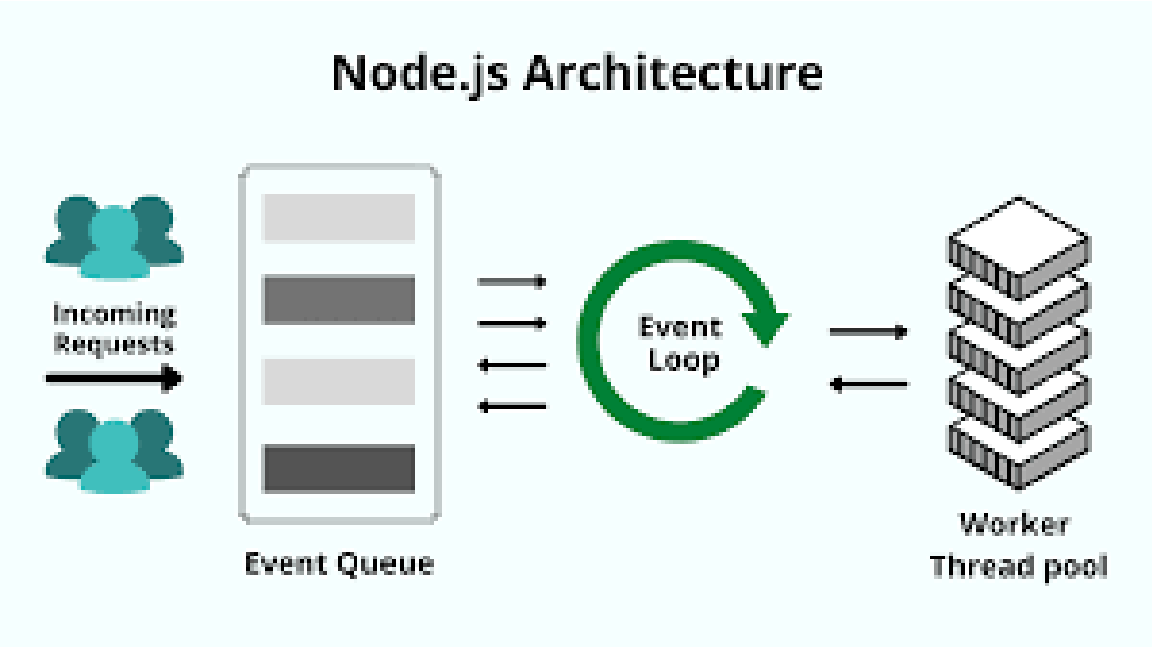
σχήμα 2.1 Το Backend είναι

Η ανάπτυξη του λογισμικού είναι μια συνεχώς εξελισσόμενη επιστήμη. Επηρεάζεται άμεσα από τις νέες τεχνολογίες και εργαλεία που αναδύονται καθημερινά. Ένας από τους πιο κρίσιμους τομείς της ανάπτυξης λογισμικού είναι η ανάπτυξη Backend εφαρμογών. Με τον όρο Backend εννοούμε το τμήμα της εφαρμογής που είναι πίσω από την σκηνή, με σκοπό να διαχειρίζεται δεδομένα, λογική αλλά και την επικοινωνία με την βάση δεδομένων ή άλλες εξωτερικές υπηρεσίες [31], [32]. Υπό αυτό το πλαίσιο, η πλατφόρμα Node.js και το

framework NestJS παίζουν πρωταγωνιστικό ρόλο, αφού επιτρέπεται μέσω αυτών η δημιουργία αποδοτικών, επεκτάσιμων και σύγχρονων backend συστημάτων [35], [36].

Το backend υποστηρίζει τις περισσότερες διαδικτυακές εφαρμογές που χρησιμοποιούνται καθημερινά. Συμμετέχει από εφαρμογές όπως τα Social Media, διαδικτυακές πλατφόρμες κοινωνικής δικτύωσης, έως τις ηλεκτρονικές αγορές [6]. Οι τεχνολογίες του backend επιτρέπουν σε αυτήν να διαχειριστούν τα δεδομένα, την ασφάλεια των εφαρμογών και τη δημιουργία των API, μεταξύ άλλων. Σήμερα, στις εφαρμογές, είναι βασικό χαρακτηριστικό η ταχύτητα, η ασφάλεια και η επεκτασιμότητα των εφαρμογών. Οι Node.js και NestJS μπορούν να προσφέρουν αυτές τις απαραίτητες δυνατότητες [35], [37].

Πιο συγκεκριμένα, το Node.js πρόκειται για μια πλατφόρμα JavaScript που φέρνει επανάσταση στην ανάπτυξη του backend, επιτρέποντας τη χρήση της ίδιας γλώσσας τόσο στο frontend όσο και στο backend [31]. Το NestJS αξιοποιεί τη δυναμική του Node.js, εισάγοντας μια δομή βασισμένη σε αρχές της ανάπτυξης λογισμικού. Πιο συγκεκριμένα, η αρθρωτή αρχιτεκτονική (modular architecture), η ευελιξία και η υποστήριξη της γλώσσας TypeScript καθιστούν το NestJS ιδανική επιλογή για την κατασκευή εφαρμογών που είναι αποδοτικές, ευανάγνωστες και συντηρήσιμες [36], [38].



σχήμα 2.2 Αρχιτεκτονική του Node.js

Η ανάπτυξη σύγχρονου κώδικα backend βασίζεται σε αρχιτεκτονικές όπως το REST API, οι οποίες επιτρέπουν την επικοινωνία μεταξύ συστημάτων με τρόπο απλό και αποτελεσματικό [6]. Ένα REST API προσφέρει διαφανή, προβλέψιμη και κατανοητή αλληλεπίδραση μεταξύ πελατών και διακομιστών. Το NestJS παρέχει εξαιρετικά εργαλεία για την υλοποίησή τους [36]. Επιπλέον, δίνεται η δυνατότητα επικοινωνίας με εξειδικευμένες υπηρεσίες, όπως εξωτερικά SPARQL Endpoints [8]. Δημιουργούνται νέες προκλήσεις και απαιτούνται εργαλεία που μπορούν να ανταποκριθούν σε πολυσύνθετες ανάγκες διαχείρισης δεδομένων [37].

Στη συνέχεια αυτού του κεφαλαίου, θα εξετάσουμε αρχικά το Node.js και το NestJS, αναλύοντας τις βασικές αρχές τους, τις διαφορές τους, καθώς και τα πλεονεκτήματά τους. Στη συνέχεια, θα αναλυθούν βασικές έννοιες που σχετίζονται με την ανάπτυξη εφαρμογών, όπως η REST API, η σύνδεση με εξωτερικά SPARQL Endpoints και ο ρόλος της JSON [10]. Θα δοθεί έμφαση στην γλώσσα προγραμματισμού TypeScript, η οποία αποτελεί τον βασικό μοχλό πίσω από την NestJS. Θα πραγματοποιηθεί μια σύγκριση της TypeScript με την παραδοσιακή JavaScript, με σκοπό να αναδειχθούν οι λόγοι που η TypeScript έχει αποκτήσει τόσο δημοφιλία στις σύγχρονες εφαρμογές [22]. Θα εξεταστούν οι ομοιότητες αλλά και οι διαφορές τους, παρέχοντας ένα συμπέρασμα που καταδεικνύει τη σημασία της TypeScript στην ανάπτυξη λογισμικού.

Ένα ακόμα θέμα που θα αναλυθεί είναι η χρήση του JSON, το οποίο αποτελεί το κύριο μέσο ανταλλαγής δεδομένων μεταξύ backend και frontend [24]. Θα εξεταστεί ο τρόπος που η JSON ενσωματώνεται και χρησιμοποιείται σε εφαρμογές NestJS και πώς επιτρέπει και διευκολύνει την επικοινωνία μεταξύ συστημάτων [36].

Εν κατακλείδι, η μελέτη των εργαλείων αυτών και των σχετικών τεχνολογιών προσφέρει όχι μόνο κατανόηση σε θεωρητικό επίπεδο, αλλά και πρακτική γνώση, απαραίτητη για την υλοποίηση πραγματικών εφαρμογών. Η ανάπτυξη λογισμικού γίνεται όλο και πιο πολύπλοκη και πολυσύνθετη, και η εξοικείωση με σύγχρονα frameworks, όπως το NestJS, είναι αναγκαία για την αντιμετώπιση των σύγχρονων προκλήσεων και την αξιοποίηση των ευκαιριών που παρουσιάζονται [23], [38].

2.2.1 Χαρακτηριστικά, χρήσεις, πλεονεκτήματα και μειονεκτήματα της Node.js

Το Node.js αποτελεί μια ανοιχτού κώδικα, cross-platform πλατφόρμα εκτέλεσης JavaScript. Μέσω αυτής επιτρέπεται η ανάπτυξη εφαρμογών από το μεριά του server. Δημιουργήθηκε το 2009, και βασίστηκε στην V8 μηχανή της Javascript της Google [30]. Το Node.js δίνει την δυνατότητα να χρησιμοποιείται η JavaScript, που συνήθως χρησιμοποιείται για client-side, προκειμένου να αναπτυχθεί ένα πλήρης λειτουργικό backend εφαρμογή.

Τα κύρια χαρακτηριστικά του Node.js είναι τέσσερα. Αρχικά έχει μη μπλοκαριστική αρχιτεκτονική και ασύγχρονη φύση, είναι Single-threaded event loop, παρέχεται ευρεία υποστήριξη βιβλιοθηκών μέσω του npm [32] αλλά και η διαλειτουργικότητα που παρέχει.

Με τον όρο μη μπλοκαριστική αρχιτεκτονική και ασύγχρονη φύση εννοούμε ότι το Node.js βασίζεται σε ένα event-driven μοντέλο προγραμματισμού. Εξασφαλίζεται έτσι ότι οι λειτουργίες δεν μπλοκάρουν το σύστημα. Επιτρέπεται έτσι η ταυτόχρονη διαχείριση πολλών αιτημάτων χωρίς να υπάρξει σημαντική επίπτωση στην απόδοσή του [31]. Με τον όρο Single-threaded event loop εννοούμε ότι μπορεί και χρησιμοποιεί μόνο έναν πυρήνα για την εκτέλεση του event loop. Υπάρχουν βιβλιοθήκες, όπως η libuv, που μπορούν να αξιοποιήσουν παρασκηνιακές λειτουργίες για τη διαχείριση πιο απαιτητικών εργασιών [31]. Μπορεί να παρέχεται ευρεία υποστήριξη βιβλιοθηκών μέσω του npm, όπου npm είναι ο μεγαλύτερος διαχειριστής πακέτων στον κόσμο και παρέχεται πρόσβαση σε χιλιάδες βιβλιοθήκες και modules. Αυτά μπορούν να χρησιμοποιηθούν για την ταχύτερη ανάπτυξη εφαρμογών [32]. Το Node.js συνεργάζεται με διάφορες βάσεις δεδομένων, όπως SQL και NoSQL, και ενσωματώνεται εύκολα με APIs τρίτων.

Μια από τις πιο συνήθεις χρήσεις του Node.js είναι η ανάπτυξη RESTful APIs. Παρέχεται έτσι

ταχύτητα αλλά και η δυνατότητα χειρισμού πολλαπλών αιτημάτων. Έτσι είναι ιδανικό για APIs που απαιτούν υψηλή ταυτόχρονη χρήση. Ακόμα μια από τις χρήσεις του Node.js είναι Real-time εφαρμογές. Δηλαδή εφαρμογές που λειτουργούν σε πραγματικό χρόνο, όπως chat εφαρμογές, ειδοποιήσεις και παιχνίδια, και αυτό γίνεται δυνατό μέσω του WebSocket για αμφίδρομη επικοινωνία [33]. Υπάρχουν επίσης πλατφόρμες όπως το AWS Lambda, που μπορούν να παρέχουν Serverless αρχιτεκτονικές. Έτσι οι προγραμματιστές μπορούν να χρησιμοποιούν το Node.js για μικρο-υπηρεσίες δίχως την ανάγκη χρήσης ενός Server [34].

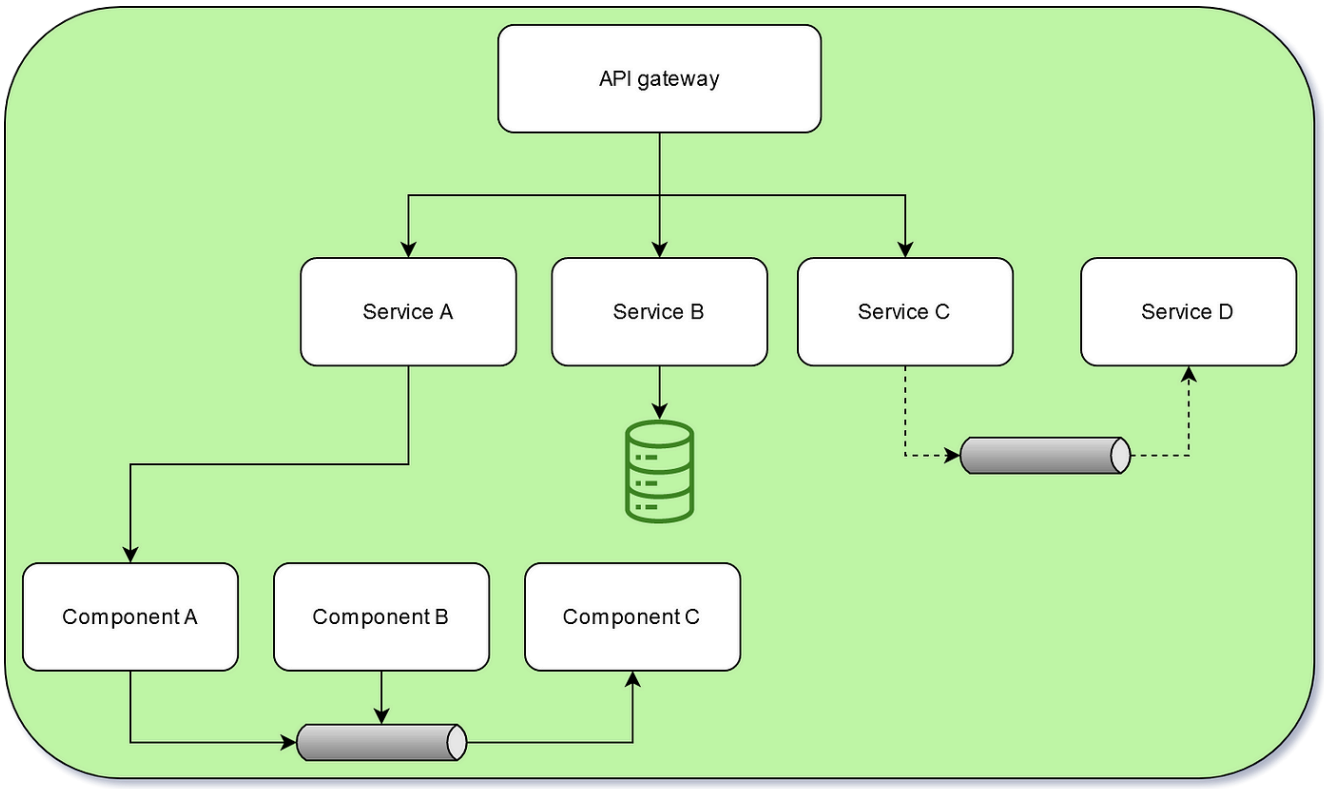
Ένα από τα πλεονεκτήματα της Node.js είναι αυτό της ταχύτητας. Λόγω της μηχανής V8, οι λειτουργίες εκτελούνται εξαιρετικά γρήγορα [30]. Ακόμα ένα από τα πλεονεκτήματα είναι η ίδια η κοινότητα. Η κοινότητα προγραμματιστών του Node.js παρέχει συνεχώς υποστήριξη και δημιουργεί νέα εργαλεία προκειμένου να αντιμετωπίσουν τα προβλήματα που αναδύονται. Το Node.js είναι αρκετά ευέλικτο, σε σημείο που να μπορεί να χρησιμοποιηθεί για μικρές εφαρμογές έως γιγάντιες πλατφόρμες. Τέλος, το γεγονός ότι οι προγραμματιστές μπορούν να χρησιμοποιούν την ίδια γλώσσα σε όλα της τα μέρη, και στο frontend και στο backend, το καθιστά αρκετά εύκολο στη χρήση.

Φυσικά όπως κάθε εργαλείο το Node.js έχει και μειονεκτήματα. Ένα από αυτά είναι το γεγονός ότι είναι single-threaded. Ενώ αυτό το μοντέλο είναι αποτελεσματικό για I/O-bound εργασίες, δεν είναι ιδανικό για εφαρμογές που απαιτούν εντατική επεξερησία από την κεντρική μονάδα επεξεργασίας. Ακόμα σε ορισμένες περιπτώσεις, η χρήση των πολλαπλών callbacks μπορεί να οδηγήσει σε πολύπλοκο και δύσκολο στην διαχείριση κώδικα. Αυτό ονομάζετε στους κύκλους του Node.js ώς Callback hell. Μπορεί να αποφευχθεί αυτό με την χρήση σωστών εργαλείων όπως promises.

Επι του συνόλου, το Node.js αποτελεί βασικό εργαλείο για έναν σύγχρονο προγραμματιστή. Η συμβολή στην μεταμόρφωση της διαδικτυακής ανάπτυξης είναι εξαιρετικά σημαντική, αφού επιτρέπεται μέσω αυτού η ταχύτερη και πιο αποτελεσματική δημιουργία εφαρμογών.

2.2.2 NestJS

Το NestJS είναι ένα framework για την ανάπτυξη backend εφαρμογών. Είναι δημιουργημένο πάνω στο Node.js και είναι σχεδιασμένο με γνώμονα τη χρήση της γλώσσας TypeScript. Δημιουργήθηκε με σκοπό να καλύψει την ανάγκη για οργάνωση, modularity και την ικανότητα ο κώδικας να είναι επεκτάσιμος. Προσφέρει στους προγραμματιστές μια δομή που κάνει πιο εύκολη την ανάπτυξη και την συντήρηση εφαρμογών μεγάλης κλίμακας [35][37].



σχήμα 2.3 Παράδειγμα Application με NestJS

Ένα από τα κύρια χαρακτηριστικά του NestJS είναι η επικέντρωσή της στην γλώσσα TypeScript. Παρόλο που το NestJS μπορεί και υποστηρίζει την χρήση Javascript, είναι σχεδιασμένο με την TypeScript στο νου. Προσφέρονται μέσω αυτής τύποι, types, και επαυξημένη ασφάλεια στον κώδικα. Ακόμα το υποστηρίζει την αρθρωτή αρχιτεκτονική, Modular Architecture. Το NestJS βασίζεται σε modules, επιτρέποντας στους προγραμματιστές να οργανώνουν τον κώδικα σε ξεχωριστά κομμάτια. Έτσι είναι πιο εύκολα επεκτάσιμα και μπορούν να επαναχρησιμοποιηθούν. Αυτό, είναι βασισμένο σε αρχές του Angular, και χρησιμοποιεί δομές εμπνευσμένες από την Angular, όπως decorators, π.χ. @Controllers, @Module. Ακόμα μπορούν να γίνουν Dependency Injection (DI) καθιστώντας το οικείο σε προγραμματιστές που ξέρουν να προγραμματίζουν σε Angular [35][36]. Το NestJS αξιοποιεί τις δυνατότητες και τις βιβλιοθήκες του Node.js, όπως το Express. Υποστηρίζει εύκολα REST APIs, GraphQL APIs και real-time εφαρμογές μέσω WebSockets. Προσφέρετε έτσι μια ευελιξία [36][37]. Τέλος το NestJS ενσωματώνει εργαλεία και βέλτιστες πρακτικές για την δημιουργία unit tests και integration tests [36].

Η χρήση του NestJS παρέχει κάποια πλεονεκτήματα. Ένα από αυτά είναι η οργανωμένη δομή που έχει. Μέσω αυτής διευκολύνεται η ανάπτυξη μεγάλων projects. Το πολλαπλά πρωτόκολλα επικοινωνίας που υποστηρίζει, όπως REST, GraphQL, WebSockets, επιτρέπει την δημιουργία πολλών και διαφορετικών εφαρμογών. Όπως και με το Node.js, υπάρχει εκτεταμένη κοινότητα προγραμματιστών που συνεισφέρουν στο NestJS, παρέχοντας υποστήριξη και πρόσθετα πακέτα [31][32][33]. Τέλος, μέσω των Dependency Injection, μπορούν να διαχειριστούν οι εξαρτήσεις και να προσφερθεί έτσι μεγαλύτερη καθαρότητα και επαναχρησιμοποίηση του κώδικα [35][36].

Η χρήση του NestJS έρχεται και με μερικά μειονεκτήματα. Ένα από αυτά είναι ότι για προγραμματιστές που δεν είναι εξοικειωμένοι με την χρήση της TypeScript ή του Angular, η

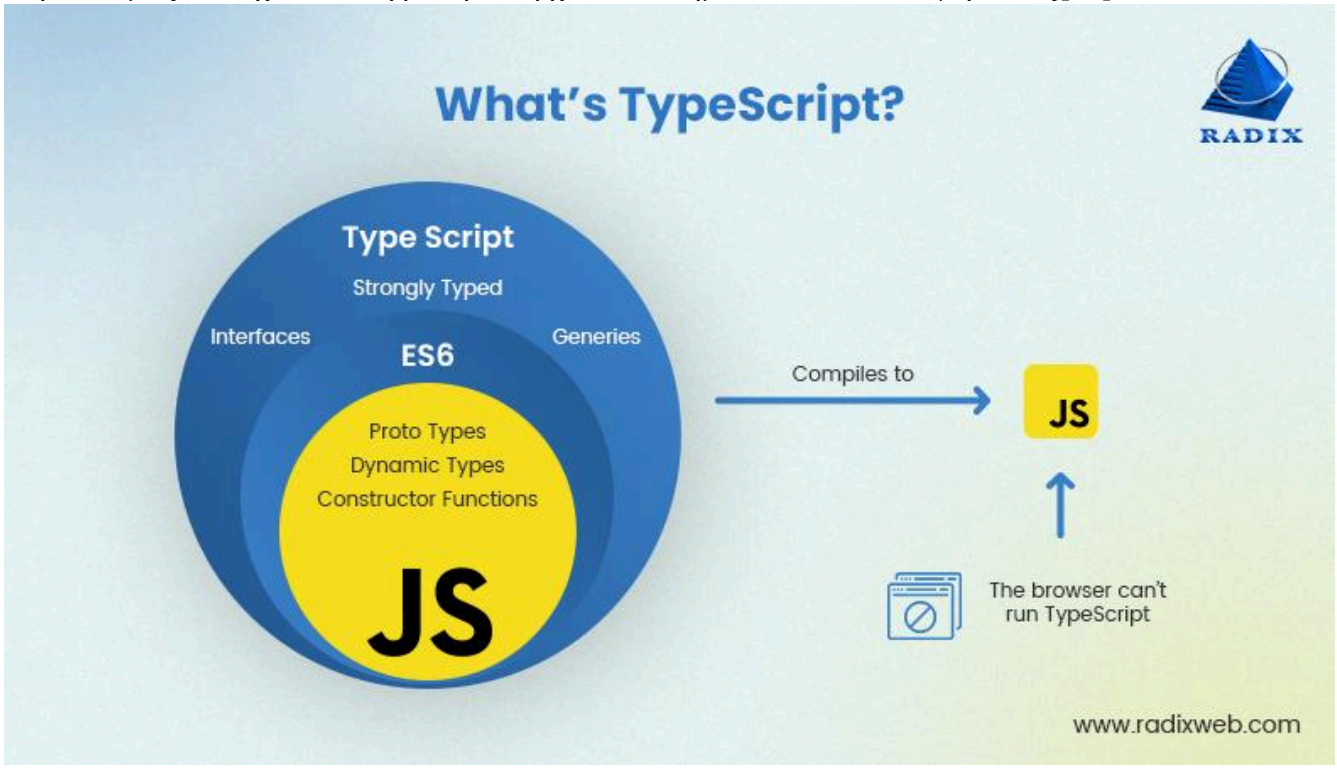
κατανόηση της αρχιτεκτονικής του NestJS μπορεί να αποβεί δύσκολη. Ακόμα, το NestJS μπορεί να είναι υπερβολικό για απλές εφαρμογές, λόγω της δομής και της πολυπλοκότητας της, έτσι τα μικρά projects μπορούν να πάθουν υπερφόρτωση [35][37].

Το NestJS χρησιμοποιείται για την δημιουργία και συντήρηση εταιρικών εφαρμογών. Το NestJS είναι ιδανικό για μεγάλα projects με πολλαπλές ομάδες ανάπτυξης. Αλλά το modular σύστημα του NestJS το καθιστά κατάλληλο για την ανάπτυξη μικρο-υπηρεσιών (microservices). Τέλος, είναι ιδανικό για εφαρμογές chat ή gaming με υποστήριξη WebSockets [35][36].

Εν κατακλείδι, το NestJS είναι ένα πανίσχυρο εργαλείο για την δημιουργία backend εφαρμογών. Ιδιαίτερα όταν η διαχείριση μεγάλης κλίμακας κώδικα είναι προτεραιότητα. Η TypeScript, η φιλοσοφία των modular και η υποστήριξη σύγχρονων προτύπων ανάπτυξης το καθιστούν ιδανική επιλογή για προγραμματιστές που επιδιώκουν να οργανώσουν, ασφαλίσουν και να δημιουργήσουν έναν επεκτάσιμο κώδικα [35][36][37].

2.2.3.1 Ομοιότητες και Διαφορές μεταξύ TypeScript και JavaScript

Η γλώσσα TypeScript και η γλώσσα JavaScript είναι δύο γλώσσες προγραμματισμού που είναι στενά συνδεδεμένες. Η γλώσσα TypeScript αποτελεί ένα υπερσύνολο της γλώσσας JavaScript. Η γλώσσα Typescript επεκτείνει τις δυνατότητες της Javascript δίχως να αναιρεί καμμία από τις λειτουργικότητες. Παρόλο όμως που έχουν κοινή βάση, υπάρχουν και σημεία που είναι διαφορετικές[40].



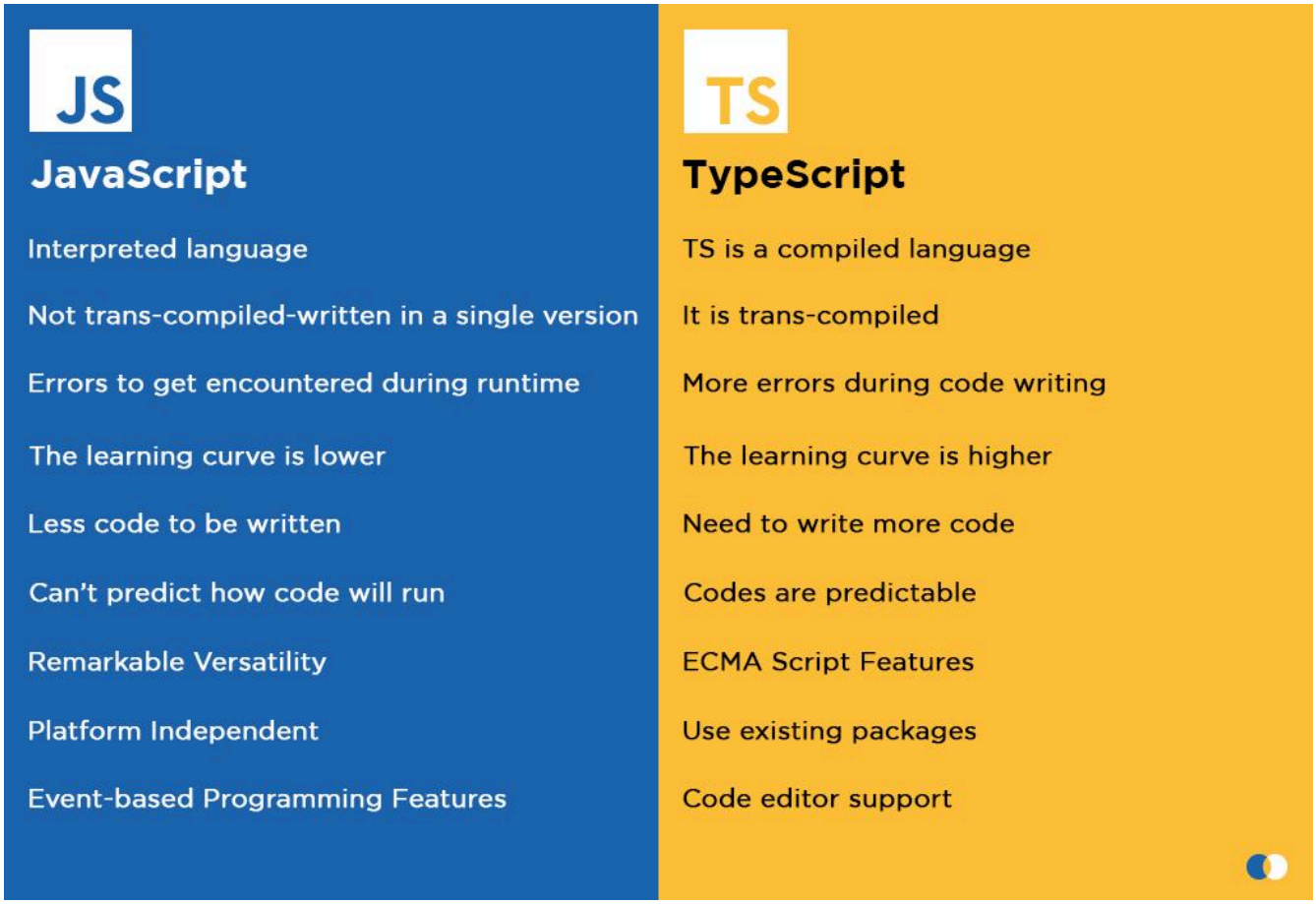
σχήμα 2.4 Υπερσύνολο Typescript

Μια από τις κυριότερες ομοιότητες μεταξύ της γλώσσας Typescript και της γλώσσας Javascript είναι η ίδια η σύνταξη των δύο γλωσσών. Η σύνταξη της γλώσσας TypeScript βασίζεται εξ ολοκλήρου στην σύνταξη της γλώσσας JavaScript. Ο οποιοσδήποτε κώδικα γραμμένος σε γλώσσα JavaScript είναι έγκυρος TypeScript κώδικας. Η δήλωση των μεταβλητών, οι βρόγχοι , οι συναρτήσεις και οι κλάσεις ακολουθούν την ίδια σύνταξη. Για παράδειγμα:

```
// Ισχύει και στις δύο γλώσσες

const greet = (name) => {

    console.log(`Hello, ${name}`);

};
```

Ακόμα ένα κοινό χαρακτηριστικό είναι ότι και οι δύο γλώσσες υποστηρίζουν το ES6/ESNext. Και οι δύο γλώσσες υποστηρίζουν τις σύγχρονες δυνατότητες της ECMAScript, όπως arrow functions, destruction, spread/rest operators και async/await. Η γλώσσα TypeScript μεταγλωττίζετε σε JavaScript, που αυτό σημαίνει ότι ο κώδικας εν τέλει εκτελείται από περιβάλλονται που υποστηρίζουν την γλώσσα JavaScript. Ένα από αυτά τα περιβάλλοντα είναι η Node.js, και όλοι οι browsers. Τέλος και στις δύο γλώσσες παρέχεται υποστήριξη από εργαλεία ανάπτυξης, όπως το Visual Studio Code που παρέχει δυνατότητα αυτόματης συμπλήρωσης (IntelliSense) και debugging.

Βέβαια εκτός από ομοιότητες μεταξύ των δύο γλωσσών υπάρχουν και διαφορές. Μια από αυτές και ίσως η πιο σημαντική, είναι η στατική τυποποίηση. Με τον όρο αυτό εννοούμε ότι οι τύποι δεδομένων, όπως string, number, boolean, καθορίζονται κατά την φάση της συγγραφής του κώδικα΄. Μειώνονται τα λάθη έτσι κατά την εκτέλεση.



σχήμα 2.5 Διαφορές μεταξύ της JavaScript και της TypeScript

Για παράδειγμα :

```
// TypeScript

function add(a: number, b: number): number {

   return a + b;

}



// JavaScript (δεν υπάρχουν τύποι)

function add(a, b) {

   return a + b;

}
```

Επεξηγηματικά στην TypeScript δηλώνετε το type, ενώ στην JavaScript δεν δηλώνεται.

Μια ακόμη γιγενής διαφορά μεταξύ της γλώσσας TypeScript και της γλώσσας JavaScript, είναι ότι η TypeScript πρέπει να μεταγλωτιστεί σε JavaScript προτού εκτελεστεί. Το ίδιο δεν συμβαίνει για την JavaScript, αφού τα περιβάλλοντα που ενεργεί, εκτελεί απευθείας τις εντολές.

Επίσεις μια από τις ειδοποιές διαφορές είναι ότι η TypeScript υποστηρίζει την χρήση προηγμένων εννοιών. Πρόκειται για πρόσθετα χαρακτηριστικά που δεν υπάρχουν στην JavaScript. Ένα από αυτά είναι τα Interfaces και Enums ώστε να μπορεί να γίνεται καλύτερη οργάνωση και χρήση του κώδικα. Ακόμα τα Generics, προσφέρουν την δυνατότητα να επαναχρησιμοποιήσετε ο κώδικας με διαφορετικούς τύπους. Τέλος τα Decorators, προσθέτουν μεταδεδομένα σε κλάσεις και μεθόδους. Για παράδειγμα:

```
interface User {

   id: number;

   name: string;

}



const user: User = { id: 1, name: "John" };
```

Επεξηγηματικά, ορίζεται ένα interface με το όνομα User, το οποίο περιγράφει τη δομή ενός αντικειμένου. Το id είναι τύπου αριθμός, ενώ το name είναι τύπου string. Δημιουργείται μια σταθερά user που είναι τύπου User. Το αντικείμενο πρ΄πρέπει να περιέχει όλες τις ιδιότητες που ορίζονται στο User και να ταιριάζουν στους τύπους. Το id του αντικειμένου έχει τιμή 1, αριθμός. Το name έχει τιμή John, string. Σε περίπτωση που κάποια από τις απαιτούμενες ιδιότητες δεν είχε τιμή ή ήταν λάθος τύπου, θα εμφανιζόταν σφάλμα κατά την μεταγλώττιση, εξασφαλίζοντας την σωστή χρήση του κώδικα. Αυτό δεν συμβαίνει με την JavaScript.

Λόγο της παλαιότητας και της μεγαλύτερης ιστορίας της JavaScript, υπάρχει πολύ περισσότερη υποστήριξη, από την κοινότητα. Πάραυτα η TypeScript αναπτύσσεται ραγδαία, και υπάρχει πολύ υποστήριξη από την κοινότητα.

Τέλος , η γλώσσα TypeScript απαιτεί από τους προγραμματιστές να εξοικειωθούν με νέες έννοιες. Για παράδειγμα η τυποποίηση και η μεταγλώττιση, κάτι που μπορεί να παραληφθεί στην JavaScript.

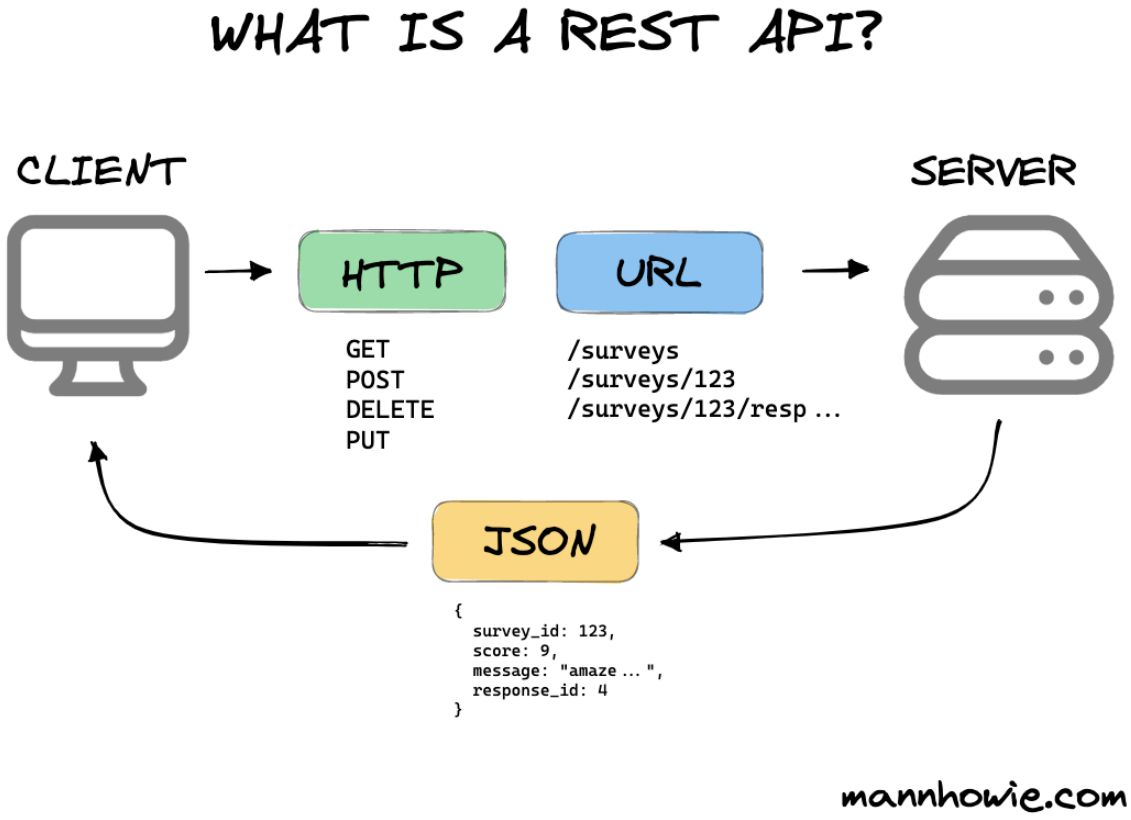
2.2.3.2 Συμπερασμα

Συμπερασματικά η γλώσσα JavaScript και η γλώσσα TypeScript είναι άρρητα συνδεδεμένες. Κάθε μια από αυτές εξυπηρετούν διαφορετικές ανάγκες. Η JavaScript είναι ιδανική για γρήγορη ανάπτυξη εφαρμογών και υπάρχει αμέτρητα εργαλεία και βιβλιοθήκες για να γίνει αυτό. Η γλώσσα TypeScript από την άλλη, μπορεί και προσφέρει μια αυξημένη ασφάλεια και δομή, καταλήγοντας να γίνεται η διαχείριση μεγάλων έργων ευκολότερη και μη επιρρεπής σε σφάλματα. Για μικρή κλίμακας εφαρμογές ή για εφαρμογές που απαιτούν ταχύτητα, προτείνετε η JavaScript. Αντίθετα όταν πρόκειται για μεγάλα έργα ή όταν υπάρχει ομάδα ανάπτυξης, η γλώσσα TypeScript είναι καλύτερη επιλογή λόγω της σαφήνειας και της συντήρησης που προσφέρει.

2.3 REST API

2.3.1 Εισαγωγή

Το REST API είναι ένα αρχιτεκτονικό στυλ σχεδιασμού που χρησιμοποιείται για την δημιουργία web υπηρεσιών. Επιτρέπετε μέσω αυτού να επικοινωνούν μεταξύ τους συστήματα HTTP. Πρόκειται για έναν από τους πιο διαδεδομένους τρόπους κατασκευής APIs. Προσφέρει απλότητα, ευελιξία και ευρεία υποστήριξη από διάφορες τεχνολογίες. Το REST API αποτελείται από μερικές βασικές αρχές[41].



σχημα 2.6 Τι είναι το API

2.3.2 Βασικές αρχές του REST API

Μια από τις βασικές αρχές του REST API, είναι ότι όλα αναφέρονται ως “πόροι” και αναπαρίστανται μέσω μοναδικών URLs. Για παράδειγμα https://api.example.com/users αναπαριστά μια λίστα χρηστών, ενώ https://api.example.com/users/1 αναφέρεται σε έναν συγκεκριμένο χρήστη. Ακόμα το REST API αξιοποιεί τις μεθόδους του HTTP για να καθορίσει τις ενέργειες που θα σε έναν πόρο[41]. Παρακάτω παρατίθενται οι ενέργειες:

- GET: Ανάκτηση δεδομένων.
- POST: Δημιουργία νέου πόρου.
- PUT: Ενημέρωση ενός υπάρχοντος πόρου.
- DELETE: Διαγραφή ενός πόρου

Μια ακόμη βασική αρχη του REST API, είναι ότι κάθε αίτημα προς το API είναι ανεξάρτητο και περιέχει όλες τις πληροφορίες που χρειάζονται για να ολοκληρωθεί. Ο server δεν διατηρεί την κατάσταση του πελάτη, αυτό ονομάζεται Statelessness. Ακόμα, μια ακόμη βασική αρχή είναι ότι τα δεδομένα που επιστρέφονται από ένα REST API συνήθως είναι σε μορφή JSON. Αυτό συμβαίνει λόγο της ευκολίας ανάγνωσης και ανάλυσης από τις εφαρμογές[41]. Μια ακόμη μορφή που συνηθίζεται είναι το XML. Τέλος, τα REST API χρησιμοποιεί HTTP status codes για να περιγράψει την έκβαση κάθε αιτήματος, παρακάτω παρατίθενται τα status codes:

- 200: Επιτυχία.
- 201: Δημιουργήθηκε (για POST αιτήματα).
- 400: Κακή αίτηση (Bad Request).
- 401: Μη εξουσιοδοτημένο (Unauthorized).
- 404: Πόρος δεν βρέθηκε (Not Found).
- 500: Εσωτερικό σφάλμα διακομιστή (Internal Server Error).

2.3.3 Παραδείγματα REST API

Παρακάτω παρατίθενται μερικά παραδείγματα χρήσης REST API:

GET https://api.example.com/users

Response:

```
[
   { "id": 1, "name": "John Doe" },
   { "id": 2, "name": "Jane Doe" }
]
```

2.3.4 Πλεονεκτήματα του REST API

Το REST API παρέχει μερικά βασικά πλεονεκτήματα. Ένα από αυτό είναι η απλότητα που προσφέρει. Χρησιμοποιώντας το πρωτόκολλο HTTP, το οποίο είναι ευρέως κατανοητό και υποστηριζόμενο. Ακόμα η χρήση του REST API προσδίδει μια ευελιξία[41]. Ένα REST API μπορεί να χρησιμοποιηθεί από οποιοδήποτε client-side, όπως για παράδειγμα browser ή mobile app, ανεξαρτήτως πλατφόρμας. Επίσης ο server και ο client λειτουργούν ανεξάρτητα ο ένας από τον άλλον. Δηλαδή υπάρχει μια ανεξαρτησία, επιτρέποντας την ανάπτυξη του καθενός, δίχως να επηρεάζεται η άλλη μεριά. Τέλος η λόγω της απλότητάς τους, τα REST APIs είναι ευκολα κλιμακώσιμο σε μεγαλύτερη εφαρμογή.

2.3.5 Μειονεκτήματα του REST API

Τα REST API έχει και μερικά μειονεκτήματα. Ένα από τα μειονεκτήματα του REST API είναι η απουσία της τυποποιησης για πολύπλοκες σχέσεις. Μερικές φορές είναι δύσκολο να εκφράσουμε πολύπλοκες σχέσεις δεδομένων με την χρήση του REST. Ακόμα ένα από τα μειονεκτήματα είναι ότι υπάρχει κατακερματισμός των αιτημάτων. Δηλαδή οι εφαρμογές που απαιτούν πολλά αιτήματα για να αντλήσουν όλα τα απαραίτητα δεδομένα, μερικές φορές παρουσιάζουν χαμηλότερη απόδοση.

2.3.6 Συμπέρασμα

Τα REST APIs αποτελούν έναν απλό, αποδοτικό και αποδεκτό τρόπο επικοινωνίας μεταξύ δύο συστημάτων ή εφραμογων. Παρόλου που υπάρχουν μερικά μειονεκτήματα, η ευελιξία και η εύκολη υλοποίησή του το καθιστά ιδανικό για πολλές εφαρμογές στον ιστό και όχι μόνο.

2.4 Χρήση του Node.js-NestJS για Σύνδεση σε SPARQL Endpoint

2.4.1 Εισαγωγή

Κάθε εφαρμογή που χρειάζεται να έχει πρόσβαση σε RDF δεδομένα ή που αλληλεπιδρά με τον Σημασιολογικό Ιστό, είναι ανάγκη να συνδεθεί με ένα SPARQL Endpoint. Το NestJS, προσφέρει ένα οργανωμένο περιβάλλον για την ανάπτυξη backend εφαρμογών, ιδανικό για σύνδεση με εξωτερικά SPARQL Endpoints. Παρακάτω θα παρουσιαστεί ο τρόπος σύνδεσης με ένα SPARQL Endpoint χρησιμοποιώντας το NestJS, μαζί με τα πλεονεκτήματα και τα μειονεκτήματα αυτής της προσέγγισης. Χρησιμοποιώντας την καθαρή αρχιτεκτονική που προσφέρει η NestJS, μπορεί κανείς να δημιουργήσει εύκολα επεκτάσιμες εφαρμογές, οργανώνοντας τη λογική σε modules και services.

2.4.2 Τρόπος Σύνδεσης σε SPARQL Endpoint με NestJS

Στην παρών εργασία χρησιμοποιείται η NestJS για την σύνδεση σε SPARQL Endpoint. Ο παρακάτω κώδικας είναι υλοποιεί την σύνδεση μέσω ενός HTTP Post αιτήματος. Ο κώδικας είναι γραμμένο σε γλώσσα TypeScript.

```
import {
  Body,
  Controller,
  Post,
  HttpException,
  HttpStatus,
} from '@nestjs/common';
import axios from 'axios';


@Controller()
export class AppController {
  @Post('query-sparql')
  async querySparql(@Body() body: { endpoint: string; query: string }) {
    const { endpoint, query } = body;


    // Basic validation to ensure that both endpoint and query are provided
    if (!endpoint || !query) {
```

```
    throw new HttpException(
      'Endpoint and query are required.',
      HttpStatus.BAD_REQUEST,
    );
  }

  try {
    // Encode the query and create the full SPARQL endpoint URL
    const encodedQuery = encodeURIComponent(query);
    const url = `${endpoint}?query=${encodedQuery}`;

    // Make a GET request to the SPARQL endpoint, specifying that we want JSON
    const response = await axios.get(url, {
      headers: {
        Accept: 'application/sparql-results+json', // Request JSON format
      },
    });

    // Log the response to see what's being returned
    console.log('Response from SPARQL endpoint:', response.data);

    // Return the response data to the client
    return response.data;
  } catch (error) {
    console.error('Error querying SPARQL endpoint:', error.message);
    throw new HttpException(
```

```
        'Failed to query the SPARQL endpoint. Please check the query and endpoint.',
        HttpStatus.INTERNAL_SERVER_ERROR,
      );
    }
  }
}
```

Επεξηγηματικά το πρώτο βήμα είναι η λήψη δεδομένων από τον Χρήστη. Αυτό γίνετε με την παρακάτω γραμμή κώδικα.

```
@Body() body: { endpoint: string; query: string }
```

Τα δεδομένα endpoint, αφορά την διεύθυνση του SPARQL Endpoint που θα ζητήσει πληροφορίες. Το query πρόκειται για το ίδιο το ερώτημα που κάνει ο χρήστης, μέσω του σώματος ,body, του POST αιτήματος. Το Body decorator παρέχεται από την NestJS και χρησιμοποιείται για να χαρτογραφήσει τα δεδομένα του αιτήματος σε μεταβλητές.

Στην συνέχεια το δεύτερο βήμα είναι ότι επικυρώνεται η είσοδος. Αναφέρετε στο if statement:

```
if (!endpoint || !query) {
  throw new HttpException(
    'Endpoint and query are required.',
    HttpStatus.BAD_REQUEST,
  );
}
```

Εδώ πραγματοποιείται ένας βασικός έλεγχος για την ύπαρξη των παραμέτρων και query. Σε περίπτωση που η παράμετροι δεν πληρούνται για παράδειγμα, για παράδειγμα αν μια από τις δύο λείπει, επιστρέφεται μια εξαίρεση HttpException με μήνυμα σφάλματος και κωδικό κατάστασης 400 Bad Request.

Το επόμενο βήμα είναι η κωδικοποίηση του SPARQL ερωτήματος. Αυτό συμβαίνει στις παρακάτω δύο γραμμές.

```
const encodedQuery = encodeURIComponent(query);
const url = `${endpoint}?query=${encodedQuery}`;
```

Το ερώτημα κωδικοποιείται με την χρήση της encodeURIComponent. Αυτό συμβαίνει προκειμένου οι ειδικοί χαρακτήρες, όπως για παράδειγμα τα κενά ή τα σύμβολα, να μην προκαλέσουν κάποιο ανεπιθύμητο πρόβλημα κατά την αποστολή του ως παράμετρο URL. Η τελική διεύθυνση του SPARQL Endpoint κατασκευάζεται προσθέτοντας την κωδικοποιημένη ερώτηση του χρήστη.

Το τέταρτο βήμα είναι το αίτημα GET στον SPARQL Endpoint. Ο κώδικας που αναφέρεται η προηγούμενη πρόταση είναι:

```
const response = await axios.get(url, {

  headers: {

    Accept: 'application/sparql-results+json', // Request JSON format

  },

});
```

Το παραπάνω σύνολο εντολών, εκτελεί ένα GET αίτημα προς το τελικό URL, χρησιμοποιώντας το HTTP client axios. Πολύ σημαντικό είναι ότι ορίζεται το header :

Accept:application/sparql-results+json

Αυτό έχει ως αποτέλεσμα να ζητηθεί η απάντηση σε μορφή JSON. Η απάντηση σε μορφή JSON είναι κατάλληλη για επεξεργασία από την εφαρμογή.

Στο τελευταίο βήμα, γίνεται η επιστροφή ή διαχείρη σφάλματος. Αν το αίτημα ολοκληρωθεί επιτυχώς, τα δεδομένα που επιστρέφονται από το SPARQL Endpoint προωθούνται στον πελάτη. Η γραμμή που γίνεται η επιστροφή είναι :

```
return response.data;
```

Σε περίπτωση αποτυχίας, το σφάλμα καταγράφεται και επιστρέφεται εξαίρεση με μήνυμα και κωδικό κατάστασης 500. Η γραμμές που οφείλονται στην διαχείριση του σφάλματος είναι :

```
catch (error) {

  console.error('Error querying SPARQL endpoint:', error.message);

  throw new HttpException(

    'Failed to query the SPARQL endpoint. Please check the query and endpoint.',

    HttpStatus.INTERNAL_SERVER_ERROR,

  );

}
```

Έτσι η εφαρμογή NestJS συνδέεται αποτελεσματικά με ένα εξωτερικό SPARQL Endpoint. Επίσης παρέχεται κατάλληλη διαχείριση αιτημάτων αλλά και των σφαλμάτων που μπορούν να προκύψουν.

2.4.3 Τα πλεονεκτήματα της χρήσης NestJS για σύνδεση σε SPARQL Endpoint.

Η χρήση της NestJS για την σύνδεση σε SPARQL Endpoint παρέχει μια σειρά από πλεονεκτήματα. Αυτά ενισχύουν την αποδοτικότητα, την επεκτασιμότητα και την ασφάλεια των εφαρμογών. Ως ένα σύγχρονο framework για την ανάπτυξη backend εφαρμογών με Node.JS, το NestJS διευκολύνει την ανάπτυξη RESTful APIs και την διαχείριση αιτημάτων σε εξωτερικά πόρους, όπως ένα SPARQL Endpoint.

Το NestJS προσφέρει μια modular και επεκτάσιμη αρχιτεκτονική. Αυτό κάνει την διαχείριση της σύνδεσης με το εξωτερικό SPARQL Endpoint μέσω controllers και services. Εξασφαλίζεται έτσι ότι ο κώδικας παραμένει καθαρός και εύκολα διαχειρίσιμος. Αυτό είναι πολύ κρίσιμο προκειμένου να μπορεί να γίνει συντήρηση και ανάπτυξη μεγάλων εφαρμογών.

Ακόμα η ύπαρξη της ενσωματωμένης υποστήριξης για τον μηχανισμό HttpException, επιτρέπει την σωστή διαχείριση σφαλμάτων με κατάλληλα HTTP status codes. Βελτιώνεται η εμπειρία του χρήστη και ο κώδικα καθιστά πιο αξιόπιστος. Παρέχονται σαφή μηνύματα σφάλματος όταν μια εργασία δεν ανταποκρίνεται όπως πρέπει.

Η χρήση της γλώσσας TypeScript στο NestJS προσφέρει ισχυρή τυποποίηση. Αυτό έχει ως αποτέλεσμα να μειωθεί η πιθανότητα κάποιου λάθους κατά την ανάπτυξη. Εξασφαλίζουν υψηλότερη σταθερότητα και καλύτερη αποφυγή προβλημάτων, που προκύψουν λόγω μη επαρκούς ελέγχου δεδομένων κατά την εκτέλεση της εφαρμογής.

Η modular φύση του NestJS επιτρέπει την επέκταση της εφαρμογές, αρκετά εύκολα. Μπορεί να επεκταθούν με επιπλέον λειτουργίες, όπως η προσθήκη caching, αυθεντικοποίησης ή άλλες διαδικασίες επεξεργασίας των δεδομένων. Άρα η εφαρμογή αποκτά μια ευελιξία και είναι έτοιμη να υποστηρίξει νέες απαιτήσεις καθώς αναπτύσσεται.

Τέλος, NestJS υποστηρίζει πλήρως τις σύγχρονες τεχνικές ανάπτυξης. Για παράδειγμα το η εντολές async/await μπορούν και διαχειρίζονται ασύγχρονα αιτήματα. Αυτή η ικανότητα είναι κρίσιμγ φια την αποτελεσματικότερη εκτέλεση αιτημάτων δικτύων. Ειδικά στον σημασιολογικό ιστό, με την χρήση SPARQL queries, που απαιτείται αναμονή για την απόκριση από εξωτερικές πηγές δεδομένων.

Εν κατακλείδι η χρήση του NestJS για την σύνδεση με SPARQL Endpoints είναι μια σύγχρονη και αποτελεσματική λύση. Συνδυάζεται η οργανωμένη αρχιτεκτονική, η ισχυρή τυποποίηση και η δυνατότητα της επέκτασης της εφαρμογής. Τα πλεονεκτήματα που προσφέρονται, όπως η διαχείριση των εξαιρέσεων, η χρήση της γλώσσας TypeScript και οι ασύγχρονες λειτουργίες που υποστηρίζονται, καθιστούν το NestJS ιδανικό για τέτοιες εφαρμογές. Εξασφαλίζεται η αξιοπιστία, η ευελιξία και η υψηλή απόδοση. Οι προγραμματιστές επωφελούνται από ένα framework που απλοποιεί την διαδικασία της συνδεσης με SPARQL Endpoint.

2.4.4 Τα μειονεκτήματα της χρήσης NestJS για σύνδεση σε SPARQL Endpoint.

Η χρήση του NestJS για τη σύνδεση με SPARQL Endpoints είναι μια δημοφιλής επιλογή για την ανάπτυξη σύγχρονων backend. Προσφέρει πλειάδα πλεονεκτημάτων, υπάρχουν και ορισμένα σημεία που μπορεί να αποτελέσουν προκλήσεις ανάλογα με την εφαρμογή. Στην παρούσα ενότητα, αναλύονται τα βασικότερα μειονεκτήματα της χρήσης του NestJS για τη σύνδεση σε SPARQL Endpoints, βοηθώντας στην καλύτερη κατανόηση των δυνατοτήτων και περιορισμών του.

Το NestJS, λόγω της αρχιτεκτονικής του, μπορεί να είναι υπερβολικά περίπλοκο για μικρές εφαρμογές

ή απλές υλοποιήσεις σύνδεσης με SPARQL Endpoints. Υπάρχουν frameworks όπως το Express.js, που μπορεί να είναι πιο απλά και να προσφέρουν πιο άμεσες λύσεις. Άρα να είναι καλύτερα για περιπτώσεις που απαιτούνται μόνο βασικές λειτουργίες.

Ακόμα αν και το NestJS είναι ένα ισχυρο framework, ταυτόχρονα και η καμπύλη εκμάθησης είναι απότομη και απαιτητική. Ειδικά για προγραμματιστές που δεν έχουν εμπειρία με TypeScript. Αυτό μπορεί να οδηγήσει τις νέες ομάδες να χρειαστούν περισσότερο χρόνο για την ανάπτυξη της εφαρμογής.

Τέλος ένα ακόμα από τα αρνητικά της NestJS είναι ότι συχνά βασίζεται σε εξωτερικές βιβλιοθήκες. Για παράδειγμα το axios χρησιμοποιείται για αιτήματα HTTP. Παρόλο που αυτό παρέχει ευελιξία, προσθέστε ένας επιπλέον βαθμός εξάρτησης, αυξάνοντας τις πιθανότητας δημιουργίας προβλημάτων. Αυτό μπορεί να συμβεί αν μια εξωτερική βιβλιοθήκη γίνει deprecated ή παρουσιαστεί κάποιο κενό ασφαλείας.

Τα παραπάνω μειονεκτήματα δεν είναι καθοριστικά. Είναι σημαντικό να ληφθούν υπόψη κατά της επιλογή του NestJS για την ανάπτυξη εφαρμογών που συνδέονται με SPARQL Endpoints. Ειδικά σε έργα με περιορισμένη κλίμακα ή πόρους.

2.5 JSON και SPARQL

2.5.1 Εισαγωγή

Η JSON και η SPARQL είναι δύο εργαλεία για την διαχείριση και την ανταλλαγή δεδομένων. Συχνα χρησιμοποιούνται σε εφαρμογές που αφορούν τον σημασιολογικό ιστό και τα διασυνδεδεμένα δεδομένα.

Η JSON είναι ένα ελαφρύ και ευαναγνωστο από τον άνθρωπο πρότυπο μορφοποίηση δεδομένων. Χρησιμοποιείται ευρέως από πολλές εφαρμογές και συστήματα, για την ανταλλαγή δεδομένων. Είναι πολύ ευέλικτο και εξασφαλίζει την διαλειτουργικότητα, και την καθιστά κατάλληλη για APIs και εφαρμογές του διαδικτύου [42].

Ένα από τα βασικά χαρακτηριστικά της JSON είναι απλότητά της. Διαθέτει ευανάγνωστη δομή με κλειδιά και τιμές, π.χ., { "key": "value" }. Ακόμα το γεγονός ότι η JSON χρησιμοποιείται από διάφορες γλώσσες του προγραμματισμού, διευκολύνει την διαλειτουργικότητα. Τέλος η JSON μπορεί να αναλυθεί και να διαχειριστεί εύκολα. Εργαλεία όπως το JSON.parse και το JSON.stringify, είναι κατάλληλα για την διαχείριση και την ανάλυσή. Εν κατακλείδι η JSON χρησιμεύει κυρίως στη διαχείριση δομημένων δεδομένων. Αποτελεί ιδανικό τρόπο παρουσίασης των αποτελεσμάτων SPARQL. Η απλότητά της είναι ιδανική για επεξεργασία από προγράμματα και εφαρμογές[42].

2.5.2 Σχέση JSON και SPARQL

Η SPARQL χρησιμοποιείται συχνά για την εξαγωγή δεδομένων από RDF βάσεις δεδομένων. Τα αποτελέσματα χρειάζονται κατάλληλη μορφοποίηση προκειμένου να αξιοποιηθούν από εφαρμογές. Η JSON είναι ο ιδανικός τρόπος παρουσίασης των αποτελεσμάτων. Η SPARQL υποστηρίζει εγγενώς την εξαγωγή αποτελεσμάτων σε μορφή JSON μέσω των headers Accept: application/sparql-results+json. Έτσι είναι πολύ πιο εύκολη η επεξεργασία και η ανάλυση των δεδομένων από εφαρμογές. Παρακάτω παρατίθεται ένα παράδειγμα SPARQL αποτελέσματος σε μορφή JSON:

```
{
```

```
 "head": {
  "vars": ["name", "birthdate"]
 },
 "results": {
  "bindings": [
   {
    "name": { "type": "literal", "value": "Albert Einstein" },
    "birthdate": { "type": "literal", "value": "1879-03-14" }
   },
   {
    "name": { "type": "literal", "value": "Marie Curie" },
    "birthdate": { "type": "literal", "value": "1867-11-07" }
   }
  ]
 }
}
```


Επεξηγηματικά, η JSON χωρίζεται σε δύο τμήματα. Το πρώτο τμήμα, το head, περιέχει τις μεταβλητές που καθορίζονται στην SPARQL ερώτηση. Επιστρέφονται ως μέρος του αποτελέσματος. Στο παραπάνω παράδειγμα, οι μεταβλητές είναι το name, και το birthday. Έτσι επιτρέπεται στις εφαρμογές, μέσω της λίστας αυτής, να γνωρίζουν ποια δεδομένα έχουν επιστραφεί. Το δεύτερο τμήμα είναι αυτά των results. Περιέχει τα αποτελέσματα της ερώτησης ως ένας πίνακας αντικειμένων. Το bindings είναι ένας πίνακας όπου κάθε στοιχείο του αντιπροσωπεύει μια εγγραφή του αποτελέσματος. Το τρίτο τμήμα είναι η κάθε εγγραφή αυτή καθεαυτή. Αποτελείται από τις μεταβλητές και τις τιμές τους, όπως τις επιστρέφει το SPARQL query. Κάθε μεταβλητή περιέχει δύο βασικά πεδία, το πρώτο είναι το type, δηλαδή ο τύπος της τιμής, και το value που αντιπροσωπεύει την πραγματική τιμή της μεταβλητής.

Παρακάτω παρατίθεται ένα παράδειγμα από την εργασία:


```
Fetched classes : Array(50)
        0:
                class: {type: 'uri', value: 'http://dbpedia.org/ontology/Company'}
                classLabel: {type: 'literal', xml:lang: 'en', value: 'company'}
                subclass: {type: 'uri', value: 'http://dbpedia.org/ontology/Publisher'}
```

.

.

.

Επεξηγηματικά, το class πρόκειται για το URI που αναπαριστά την κλάση ontology. Στο παράδειγμα http://dbpedia.org/ontology/Company, που αντιστοιχεί στην έννοια "Εταιρεία". Το classLabel είναι μια ετικέτα της κλάσης, προκειμένου να είναι ευανάγνωστη σε ανθρώπους. Είναι τύπου literal, που δηλώνει ότι πρόκειται για μια λεκτική τιμή. Το xml:lang: 'en' δηλώνει ότι η γλώσσα είναι στα Αγγλικά. Το subClass είναι το URI μιας από τις υποκλάση της κλάσης Company. Στο παράδειγμα: http://dbpedia.org/ontology/Publisher, που δηλώνει ότι η "Publisher" (Εκδότης) είναι υποκλάση της "Company".

2.5.3 Πλεονεκτήματα και μειονεκτήματα συνδυασμού JSON και SPARQL

Ο συνδυασμός JSON και SPARQL παρέχει μερικά πολύ σημαντικά πλεονεκτήματα στην δημιουργία μια εφαρμογής ανάκτησης δεδομένων RDF. Ένα από αυτά είναι ότι η JSON διευκολύνει τη διασύνδεση εφαρμογών που χρειάζονται τα αποτελέσματα SPARQL, και γίνετε δυνατόν αυτό, από την ευρέως διαδεδομένη μορφή της JSON. Ακόμα τα δεδομένα JSON μπορούν να αναλυθούν γρήγορα και να χρησιμοποιηθούν σε γλώσσες όπως η JavaScript, και συνεπώς της TypeScript. Τέλος οι περισσότερες εφαρμογές και τα περισσότερα APIs προτιμούν την χρήση JSON ως μορφή ανταλλαγής δεδομένων. Άρα είναι ιδανικό για την παρουσίαση αποτελεσμάτων SPARQL.

Παρά την χρησιμότητα του συνδιασμού JSON και SPARQL, υπάρχουν και ορισμένα μειονεκτήματα που πρέπει να ληφθούν υπόψη. Αρχικά τα δεδομένα SPARQL, όταν μετατρέπονται σε JSON, περιέχουν πολυεπίπεδη δομή. Αυτό μπορεί να δυσκολέψει την επεξεργασία των δεδομένων, ειδικά αν η εφαρμογή δεν σχεδιάστηκε για να διαχειρίζεται τέτοιου είδους πολυπλοκότητας. Ακόμη η μετατροπή των δεδομένων RDF σε JSON συχνά οδηγεί σε πλεονασμό πληροφοριών. Για παράδειγμα μερικά URIs μπορεί να επαναλαμβάνονται. Αυτό μπορεί να αυξήσει το μέγεθος των δεδομένων επηρεάζοντας έτσι την απόδοση του συστήματος, ειδικά όταν τα σύνολα δεδομένων είναι πολύ μεγάλα. Τέλος, ενώ η JSON παρουσιάζει τα αποτελέσματα με φιλικό τρόπο, δεν αποτυπώνει πλήρως τη σημασιολογική πληροφορία των RDF δεδομένων. Για παράδειγμα αυτό μπορεί να συμβεί με τους τύπους σχέσεων ή και το context. Δυσκολεύονται έτσι πιο σύνθετες σημασιολογικες αναλύσεις ή μετασχηματισμούς.

Ο συνδυασμός JSON και SPARQL αποτελεί ένα ισχυρό εργαλείο για τη διαχείριση, εξαγωγή και παρουσίαση δεδομένων του σημασιολογικού ιστού. Παρέχονται πλεονεκτήματα όπως η ευκολία ενσωμάτωσης, ευελιξίας στην επεξεργασία η ευρεία υποστήριξη, που την καθιστά ιδανική για πολλές περιπτώσεις. Ωστόσο, υπάρχουν και μειονεκτήματα, όπως η πολυπλοκότητα της δομής της JSON, η αύξηση των δεδομένων και η απώλεια της σημασίας του περιεχομένου. Εν κατακλείδι η επιλογή της JSON για την παρουσίαση του SPARQL εξαρτάται από τις ανάγκες του συστήματος. Σε περιπτώσεις όπως η φιλική παρουσίαση και η γρήγορη ενσωμάτωση είναι προτεραιότητες, είναι μια εξαιρετική επιλογή. Αντίθετα, εργα που απαιτούν υψηλή ακρίβεια στην έννοια των δεδομένων, χρειάζονται και συμπληρωματικές προσεγγίσεις.

2.6 NestJS και SPARQL: Γνωστές εφαρμογές και χρήσεις

Το NestJS είναι ένα από τα πιο δημοφιλής frameworks για την ανάπτυξη backend εφαρμογών. Η ίδια η αρχιτεκτονική, διευκολύνει την επεκτασιμότητα και τη συντήρηση. Υποστηρίζονται πολλά εργαλεία και προσεγγίσεις όπως Dependency Injection, REST, και GraphQL. Παράλληλα η SPARQL χρησιμοποιείται για την εξαγωγή και την επεξεργασία RDF δεδομένων, σε εφαρμογές σημασιολογικού

ιστού. Ο συνδυασμός του NestJS με την SPARQL είναι ιδιαίτερα χρήσιμος σε εφαρμογές που βασίζονται σε γνώση και δεδομένα. Αυτά περιγράφονται μέσω μοντέλων RDF. Παρακάτω ακολουθούν γνωστές περιπτώσεις εφαρμογών και χρήσεων.

Ένας από τις πιο βασικές εφαρμογές είναι αυτή των Knowledge Graph APIs. Πολλές εταιρείες και οργανισμοί αναπτύσουν APIs για αναζήτηση βασισμένο σε knowledge graphs. Από τους πιο γνωστούς οργανισμούς είναι η Wikidata και η DBpedia. Αυτοί οι οργανισμοί επιλέγουν την NestJS για το backend. Αυτό που προσφέρει η NestJS είναι τα RESTful APIs για την πρόσβαση σε RDF δεδομένων. Δίνετε η διαχείριση σύνθετων αιτήσεων προς SPARQL endpoints, με δυνατότητα caching αλλά και validation. Ακόμα η ολοκληρωμένες βιβλιοθήκες που υπάρχουν, διευκολύνουν την δημιουργία ερωτημάτων SPARQL.

Ακόμα γνωστή εφαρμογή είναι αυτή του Semantic Search. Δηλαδή το Semantic Search αφορά την αναζήτηση με βάση τη σημασιολογική έννοια. Το NestJS μπορεί να συνδεθεί με SPARQL endpoints προκειμένου να γίνει η αναζήτηση δεδομένων από RDF βάσεις, όπως το Wikidata ή το MusicBrainz. Δίνετε η δυνατότητα για ανάπτυξη εργαλείων ανάλυσης ενσωματώνοντας την φυσική γλώσσα με την SPARQL, προκειμένου να γίνει εξαγωγή δεδομένων από πολυδιάστατα knowledge graphs.

Επίσεις πολύ γνωστές είναι οι εφαρμογές με σκοπό την διαχείριση της πολιτιστικής κληρονομιάς. Πολλά έργα στο τομέα της πολιτιστικής κληρονομιάς, όπως για παράδειγμα Europeana, βασίζονται σε RDF και SPARQL, προκειμένου να γίνει η διασύνδεση και την οργάνωση των δεδομένων. Η NestJS λειτουργεί ως μια γέφυρα για να γίνει η παροχή SPARQL endpoints σε τελικούς χρήστες μέσω φιλικών διεπαφών προς τον χρήστη. Ακόμα έχουν δημιουργηθεί εργαλεία οπτικοποίησης των δεδομένων μέσω APIs. Τέλος ενοποιούνται τα δεδομένα από πολλαπλές RDF πηγές.

Τέλος, υπάρχουν εφαρμογές IoT όπως και εφαρμογές Smart Cities. Οι έξυπνες πόλεις και οι IoT πλατφόρμες πολύ συχνά χρησιμοποιούν RDF δεδομένα. Αυτό συμβαίνει προκειμένου να γίνεται η οργάνωση και η ανάλυση αυτών. Με την χρήση του NestJS, μπορούν να δημιουργηθούν APIs, προκειμένου να γίνεται η συλλογή και η ανάλυση των SPARQL-based δεδομένων από IoT συσκευές. Τα συστήματα παρακολούθησης και reporting βασίζονται σε δεδομένα RDF.

Εν κατακλείδι το NestJS, παρέχει μια ισχυρή υποστήριξη μέσω για τη modular ανάπτυξη και την διαχείριση δεδομένων. Συνδυάζεται ιδανικά με την SPARQL σε εφαρμογές γνώσης και σημασιολογικής αναζήτησης. Ο συνδυασμός αυτός παρέχει μια ευελιξία, ταχύτητα και δυνατότητα διαχείρισης μεγάλου όγκο δεδομένων.

2.7 Συμπέρασμα

Η ανάπτυξη backend εφαρμογών με χρήση του NestJS αναδεικνύει τη δύναμη και την ευελιξία των σύγχρονων εργαλείων ανάπτυξης. Ειδικά όταν συνδυάζεται με την ισχύ της Node,js και τη δομημένη προσέγγιση της TypeScript. Η σύγκριση της TypeScript και Javascript, κατανοούμε πως η προσθήκης στατικής τυποποίησης βελτιώνει την ποιότητα και τη συντηρησιμότητα του κώδικα. Παράλληλα το NestJS μέσω της αρχιτεκτονική του παρέχει την ιδανική βάση για την δημιουργία REST APIs. Το JSON διευκολύνει τη διαχείριση και τη ανταλλαγή των δεδομένων. Οι εφαρμογές του NestJS εκτείνονται και σε άλλους τομείς και συστήματα, όπως semantic search ή APIs για την διαχείριση και την διάθεση δεδομένων RDF. Συνολικά, το NestJS συνδυάζει τα καλύτερα χαρακτηριστικά της σύγχρονης ανάπτυξης backend. Προσφέρεται η ευελιξία, επεκτασιμότητα και ισχυρά εργαλεία για την ανάπτυξη εφαρμογών, που εξυπηρετούν σύνθετες ανάγκες δεδομένων και υπηρεσιών.

Κεφάλαιο 3ο: Frontend

3.1 Εισαγωγή

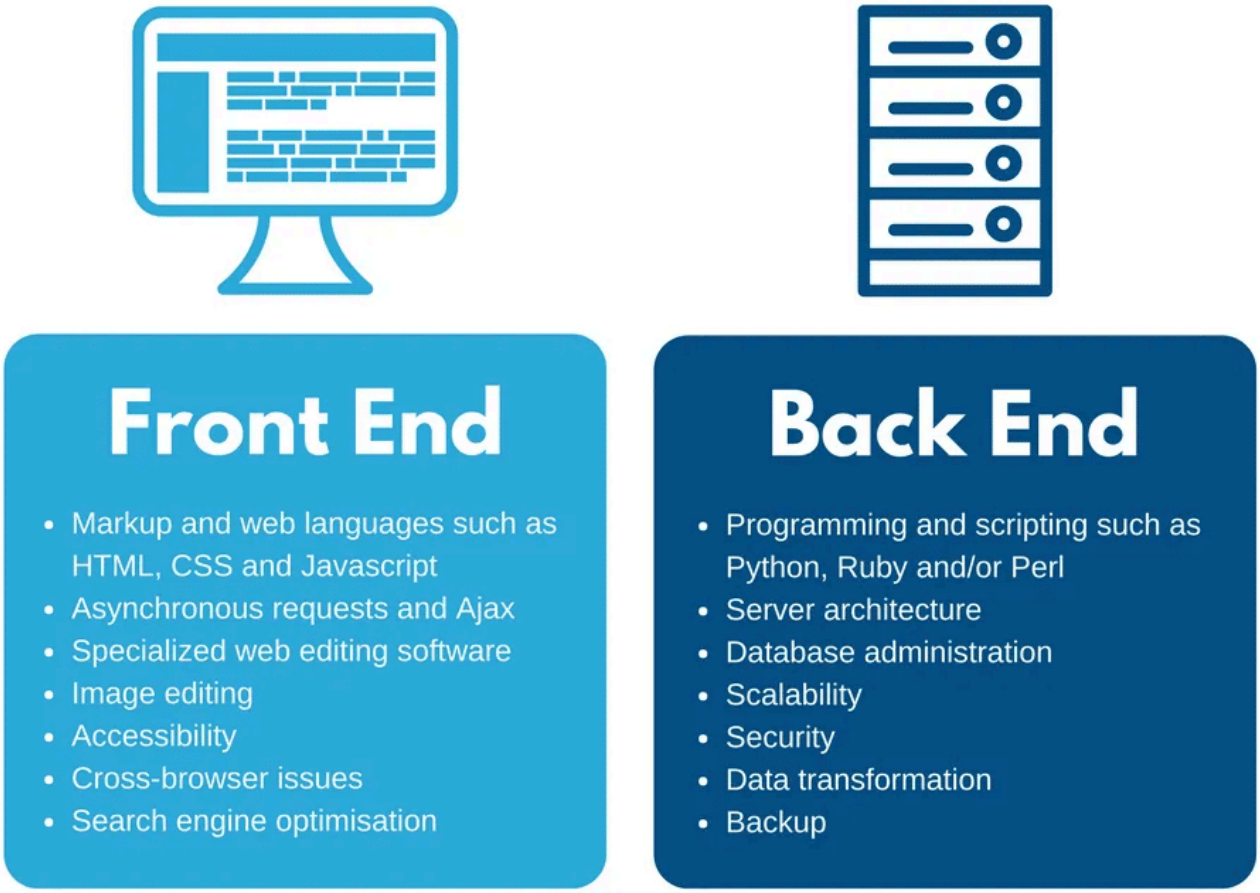
Το frontend είναι ένας από του πιο σημαντικές πτυχές κάθε διαδικτυακής εφαρμογής. Αποτελεί το σημείο αλληλεπίδρασης του χρήστη με την εσωτερική λογική της εφαρμογής. Στην παρούσα εργασία, ο σχεδιασμός του frontend είχε ως στόχο να παρέχει ένα φιλικό προς τον χρήστη περιβάλλον, προκειμένου να του δίνετε εύκολα η δυνατότητα να εκτελούνται SPARQL ερωτήματα και αυτά να οπτικοποιηθούν, σε μορφή ενός γράφου. Μέσω της βιβλιοθήκης D3.js και της Nest.js, η εφαρμογή καταφέρνει να απεικονισει σύνθετες σχέσεις και δεδομένα.

Γενικά, ο σκοπός αλλά και ο ρόλος του frontend είναι να επιτρέπεται η μετάφραση των τεχνικών λειτουργιών του backend σε εμπειρίες που είναι διαδραστικές για τον τελικό χρήστη. Η ανάπτυξη χωρίζεται σε τρεις τομείς, πρώτων η χρήση της HTML για την δομή του περιεχομένου και την σειρά της απεικόνισης, το CSS που ευθύνεται για την οπτικό στυλ της εφαρμογής, και τέλος την JavaScript που διαχειρίζεται την διαδραστικότητα και την δυναμική φόρτωση δεδομένων. Στην παρούσα εργασία η οπτικοποίηση των RDF δεδομένων γίνεται μέσω γραφημάτων που στοχεύουν στο διευκολυνθεί η κατανόηση σύνθετων σχέσεων, και κυριότερα αυτών των σχέσεων μεταξύ κλάσεων και υποκλάσεων [99][100].

Η εργασία αυτή προσπαθεί να ενσωματώσει τις σύγχρονες τάσεις, δυνατότητες και καινοτομίες στην ανάπτυξη του frontend. Μέσω της χρήσης της βιβλιοθήκης D3.js, περισσότερα για αυτή παρακάτων, τα δεδομένα αναπαρίστανται σε κόμβους και ακμές, δίνοντας έτσι στον χρήστη ένα οπτικό περιβάλλον που κάνε πιο εύκολη την ανάλυση και την κατανόηση των σχέσεων [101]. Έχει δημιουργηθεί ένα ευέλικτο Query Builder, το οποίο οι χρήστες μέσω των επιλογών τους να δημιουργήσουν αλλά και να εκτελέσουν τα δικά τους εξειδικευμένα ερωτήματα. Ακόμα οι χρήστες μπορούν να κάνουν διάφορες αποθηκεύσεις στον τοπικό τους υπολογιστή, είτε των δεδομένων που δέχονται από την εξωτερική βάση δεδομένων είτε την ίδια την ερώτηση που έγινε στην εξωτερική βάση δεδομένων. Τέλος δίνετε και η δυνατότητα να γίνει λήψη ενός στιγμιότυπου του γράφου, όπως είναι εκείνη την στιγμή.

Η σημασία του frontend είναι πολύ σημαντική και επιβεβαιώνεται σε μελέτες. Αναδεικνύουν την θετική επίδραση στην απόδοση και την χρηστικότητα των εφαρμογών. Ειδικά όσον αφορά την οπτικοποίηση δεδομένων βελτιώνει την κατανόηση πολύπλοκων συνόλων δεδομένων, και ενισχύεται η αντίληψη και η λήψη αποφάσεων[99][102]. Διευκολύνετε και η δημιουργία δυναμικών και επεκτάσιμων συστημάτων οπτικοποίησης[101] .

Εν κατακλείδι, το frontend οποιαδήποτε σύγχρονης εφαρμογής, δεν περιορίζεται σε στατικές απεικονισης αλλά προσφέρονται εργαλεία ανάλυσης και διαχείρισης δεδομένων. Η λειτουργικότητα του το καθιστά ένα από τα πιο κρίσιμα στοιχεία για την διεπαφή με τον τελικό χρήστη, ειδικά σε εφαρμογές ανάλυσης των δεδομένων, όπως η παρούσα εργασία. Οι διαδραστικές δυνατότητες εξασφαλίζουν ότι η εφαρμογή είναι προσβάσιμη και χρηστική σε κάθε τύπο χρήστη, ανεξάρτητα του επιπέδου των τεχνικών γνώσεων που έχει.

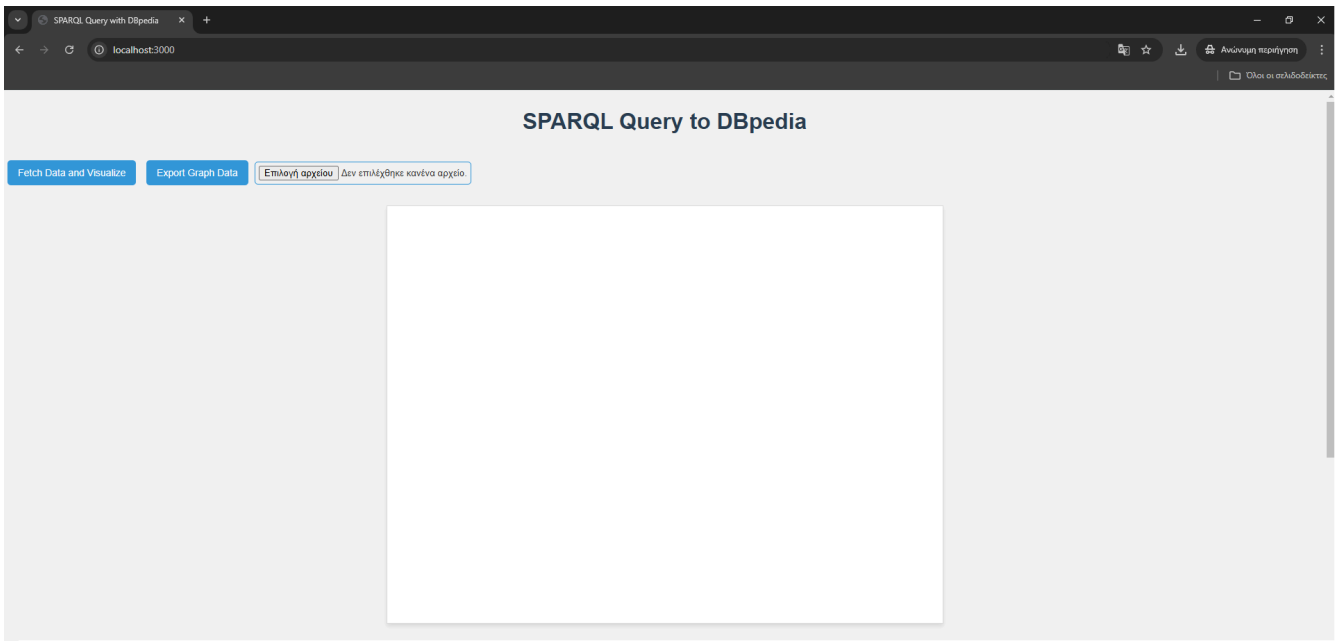


σχήμα 3.1 Διαφορές μεταξύ frontend και Backend

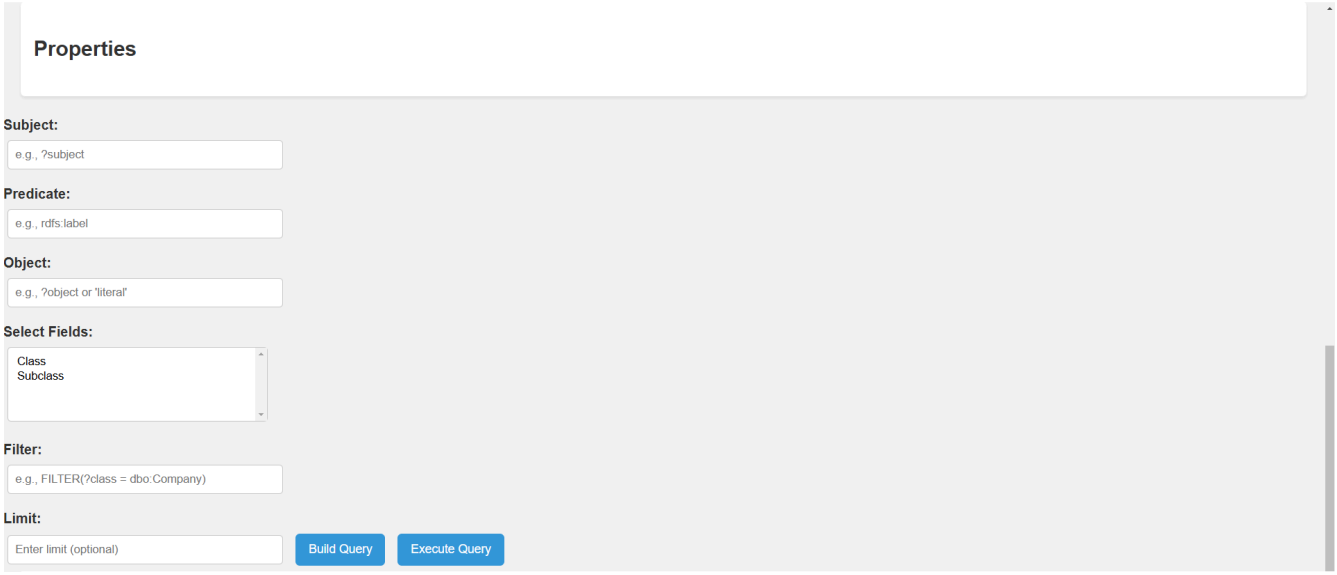
3.2 HTML

3.2.1 Γενικά

Η HTML πρόκειται για μια βασική γλώσσα σήμανσης που χρησιμοποιείται για την κατασκευή ιστοσελίδων και εφαρμογών στο διαδίκτυο. Χρησιμοποιείται για να δώσει μια δομή στο περιεχόμενο των σελίδων. Μέσω αυτής καθορίζετε η διάταξη και η μορφή των επιμέρων χαρακτηριστικών της σελίδας. Η HTML δεν είναι γλώσσα προγραμματισμού, γιατί δεν περιλαμβάνει κάποια λογική ροή ή υπολογισμούς, επάνω της βασίζονται όλες οι δυναμικές ιστοσελίδες.



σχήμα 3.2.1 Frontend της παρούσας εργασίας



σχήμα 3.2.2 Συνέχεια του ίδιου Frontend

Εν συντομία η HTML αποτελείται από στοιχεία, πιο γνωστά ως tags. Αυτά περιβάλλουν το περιεχόμενο και το διαμορφώνουν κατάλληλα. Αυτά τα tags χρησιμοποιούνται για να ορίσουν ότι μπορεί κάποιος να αντικρίσει στο διαδίκτυο. Κάθε tag έχει συγκεκριμένη δομή και σημασία, και είναι μείζων ζήτημα η σωστή χρήση προκειμένου να γίνει ομαλή εμφάνιση και λειτουργία μια ιστοσελίδας. Για παράδειγμα, το tag <h1> χρησιμοποιείται για τις κύριες επικεφαλίδες, ενώ το tag <a> χρησιμοποιείται για την δημιουργία υπερσυνδέσμων.

Παραδειγματα:

<h1>This is an h1 tag</h1>

<a>This is an a tag

Η HTML δημιουργήθηκε από τον Tim Berners-Lee το 1991, και έχει εξελιχθεί πολλές φορές με πολλές αναθεωρήσεις και εκδόσεις. Για περισσότερες πληροφορίες για την ιστορία του παγκόσμιου ιστού μπορείτε να ανατρέξετε στην πηγή [105]Η τελεταία είναι η έκδοση HTML5, η οποία προσφέρει δυνατότητα υποστήριξεις βίντεο και ήχου χωρίς πρόσθετα plugin, βελτιωμένη υποστήριξη για εφαρμογές web και καλύτερη συμβατότητα με κινητές συσκευές. Για περισσότερες πληροφορίες σχετικά με την HTML5 και τις νέες δυνατότητες παρατίθενται η επίσημη ιστοσελίδα του W3C, και τα MDN Web Docs [103][104]. Η HTML δεν είναι ικανή από μόνη να κατασκευάσει ιστοσελίδες. Συνδυάζεται και με άλλες τεχνολογίες όπως η CSS και η JavaScript, όπου θα αναλυθούν παρακάτω.

3.2.2 HTML κώδικας της εργασίας

Η παρούσα εργασία ασχολείται με την απεικόνιση των SPARQL δεδομένων σε μορφή γράφου. Ο κώδικας είναι οργανωμένος με την παραδοσιακή δομή μιας ιστοσελίδας. Αποτελείται από το tag <head>, και το tag <body>. Το head περιέχει βασικά μεταδεδομένα, όπως η γλώσσα και η κωδικοποίηση των χαρακτήρων. Δηλώνονται τα αρχεία JavaScript που χρησιμοποιούνται για την λειτουργία της ιστοσελίδας. Στο body περιλαμβάνεται η βασική δομή, δηλαδή οι τίτλοι, τα κουμπια τα πεδία εισόδου και οι περιοχές για την εμφάνιση των αποτελεσμάτων. Αυτά είναι τα στοιχεία που ο χρήστης αλληλεπιδρά.

Οι χρήστες πατώντας το κουμπί "Fetch Data and Visualize" ανακτούν τα δεδομένα από τον server και γίνετε απεικόνιση σε μορφή γραφήματος. Με το πάτημα του κουμπιού "Export Graph Data" επιτρέπεται η εξαγωγή των δεδομένων του γραφήματος, κατεβάζοντας τα δεδομένα σε μορφή JSON. Το κουμπί "Take Screenshot" παρέχει την δυνατότητα να ληφθεί ένα στιγμιότυπο της τρέχουσας κατάστασης του γράφου και να αποθηκευτεί τοπικά στο υπολογιστή, ενώ το το κουμπί "Import Graph Button", επιτρέπει την εισαγωγή ενός γραφήματος από ένα αρχείο JSON. Η ιστοσελίδα περιλαμβάνει και πεδία εισόδου για την εισαγωγή των στοιχείων ερωτήματος SPARQL. Αυτά τα πεδία περιλαμβάνουν το υποκείμενο, Subject , το ρήμα ,Predicate, και το αντικείμενο, object. Δίνετε και η επιλογή επιλογής φίλτρων και ορίων για τα αποτελέσματα του ερωτήματος. Με το κουμπί "Build Query", ο χρήστης δημιουργεί το ερώτημα SPARQL, χρησιμοποιώντας τα προαναφερθέν πεδία. Με το κουμπί "Execute Query" το ερώτημα εκτείλετε και τα αποτελέσματα εμφανίζονται στην περιοχή <pre id="queryResults">. Η περιοχή <svg id="graph"> είναι το μέρος που χρησιμοποιείται για την απεικόνιση των δεδομένων σε μορφή γράφου. Η περιοχή   εμφανίζει τις ιδιότητες των δεδομένων που αναρτώνται. Γεμίζει δυναμικά ανάλογα τα αποτελέσματα του ερωτήματος SPARQL. Ο κώδικας περιλαμβάνει διάφορα αρχεία JavaScript το κάθε ένα υπεύθυνο για την λογική και αλληλεπίδραση με την ιστοσελίδα.

Εν κατακλείδι η HTML είναι η θεμελιώδης γλώσσα σήμανσης που χρησιμοποιείται για τη δημιουργία και δομή των ιστοσελίδων στο διαδίκτυο. Στον κώδικα όπου χρησιμοποιήθηκε για την συγγραφή αυτής της εργασίας λειτουργεί ώς το πλαίσιο για την διαμόρφωση της ιστοσελίδας. Παρέχονται τα βασικά στοιχεία που απαιτούνται για να υπάρχει αλληλεπίδραση με τον χρήστη και την εμφάνιση των δεδομένων.

3.3 CSS

Το CSS, είναι μια γλώσσα σχεδίασης που χρησιμοποιείται για την διαμόρφωση της εμφάνισης μιας ιστοσελίδας. Ορίζουμε μέσω αυτού, την παρουσίαση των στοιχείων HTML, όπως το χρώμα, τη γραμματοσειρά, τα περιθώρια, τα διαστήματα και άλλες στυλιστικές επιλογές. Είναι ένα βασικό και ισχυρό εργαλείο που επιτρέπει να αποσυνδεθεί η δομή της ιστοσελίδας από το πως φαίνεται, κάνοντάς την ανάπτυξη των ιστοσελίδων πιο ευέλικτη και ευανάγνωστη. Ο παρακάτω κώδικας παρέχει μια σειρά από κανόνες CSS που έχουν σχεδιαστεί με σκοπό να προσφέρουν καθαρό, εθαναγνωστό και ελκυστικό περιβάλλον για τον χρήστη. Εστιάζετε η διαμόρφωση των στοιχείων όπως το σώμα της σελίδας, τα κουμπιά, οι φόρμες και οι περιοχές προεπισκόπησης ή αποτελεσμάτων.

σχήμα 3.3 CSS Λογότυπο

Παρακάτω παρατίθεται ο κώδικας:

```
/* General body styling */

body {

    font-family: Arial, sans-serif;

    line-height: 1.6;

    margin: 0;

    padding: 0;

    background-color: #f4f4f4;

    color: #333;
```

```
}

/* Header */
h1 {
    text-align: center;
    margin-top: 20px;
    color: #2c3e50;
}

/* Button Styling */
button {
    background-color: #3498db;
    color: white;
    border: none;
    padding: 10px 15px;
    font-size: 14px;
    cursor: pointer;
    margin: 5px;
    border-radius: 5px;
}

button:hover {
    background-color: #2980b9;
}

/* Input and Select Styling */
```

```
input[type="text"],
input[type="number"],
select {
    padding: 8px;
    margin: 5px;
    border: 1px solid #ccc;
    border-radius: 4px;
    width: 300px;
}

input[type="file"] {
    margin: 10px 0;
}

/* Label Styling */
label {
    display: block;
    margin-top: 10px;
    font-weight: bold;
}

/* Graph Container */
#graph-container {
    display: flex;
    justify-content: center;
    align-items: center;
```

```
    margin: 20px 0;
}

#graph {
    border: 1px solid #ddd;
    background-color: #fff;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

/* Query Preview and Results */
#queryPreview,
#queryResults {
    background-color: #ecf0f1;
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 5px;
    margin: 10px;
    font-family: monospace;
    white-space: pre-wrap;
    word-wrap: break-word;
}

#queryPreview {
    margin-top: 20px;
}
```

```
#prop_div {
    margin: 20px;
    padding: 15px;
    background: #ffffff;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    border-radius: 5px;
}

/* Button Container */
#buttonContainer {
    text-align: center;
}

/* File Input Button */
input[type="file"] {
    padding: 5px;
    border: 1px solid #3498db;
    border-radius: 5px;
    cursor: pointer;
}

/* Make the layout more responsive */
@media (max-width: 768px) {
    input[type="text"],
    input[type="number"],
    select {
```

```
        width: 100%;
    }

    #graph {
        width: 100%;
    }

    button {
        width: 100%;
        font-size: 16px;
    }

    #prop_div {
        margin: 10px;
    }
}
```

Πιο συγκεκριμένα, ορίζεται μια γραμματοσειρά (Arial, sans-serif), και μια απόχρωση για το background (#f4f4f4). Το χρώμα του κειμένου είναι (#333). Κάθε επικεφαλίδα με ετικέτα h1 έχει κεντραριστεί και το χρώμα της είναι (#2c3e50). Τα κουμπιά έχουν μπλε χρώμα (#3498db) με ένα εφε όταν ο χρήστης βάλει τον κέρσορα πάνω, κάνοντάς την λίγο πιο έντονο μπλε (#2980b9). Γωνίες των κουμπιών είναι στρογγυλεμένες, και το μέγεθός τους προσαρμόζεται για άνετη αλληλεπίδραση. Τα πεδία εισαγωγής κειμένου, αριθμών και επιλογών, έχουν μαλακά περιγράμματα και ευρύχωρο σχεδιασμό 300px πλάτος. Το container του γραφήματος έχει κεντραριστεί και περιβάλλεται από ένα πλαίσιο. Με την χρήση των media queries, η διάταξη και τα στοιχεία προσαρμόζονται στις μικρότερες οθόνες.

Ο παραπάνω κώδικας έχει σκοπό να αξιοποιήσετε τις δυνατότητες του CSS με σκοπό να κατασκευάσει μια καλαίσθητη και λειτουργική σελίδα. Η χρήση του CSS ενισχύει την οπτική ταυτότητα μιας εφαρμογής. Βελτιώνεται μέσω αυτού και η εμπειρία του χρήστη. Ο συνδυασμός της αισθητικής και τις ανταποκρισιμότητας στις συσκευές κάνει τη σελίδα προσβάσιμη και ευχάριστη σε όλους τους χρήστες.

3.4 Λειτουργικότητες σε JavaScript

Στην παρακάτω υποενότητα, θα εστιάσουμε την προσοχή μας στην λειτουργικότητα που υλοποιείται μέσω της JavaScript και της σύνδεσης με την οπτικοποίηση δεδομένων που προέρχονται από την βάση

δεδομένων DBpedia. Η χρήση της JavaScript γίνετε προκειμένου να επιτευχθεί η αλληλεπίδραση με τα δεδομένα, την κατασκευή των γραφημάτων και την διαχείριση των εισόδων και εξόδων των χρηστών. Πρωταγωνιστικό ρόλο έχει η βιβλιοθήκη D3.js, αφού χρησιμοποιείται για την οπτικοποίηση των δεδομένων σε δυναμικά γραφήματα.

σχήμα 3.4 JavaScript λογότυπο

3.4.1 Οπτικοποίηση Δεδομένων

Η οπτικοποίηση των δεδομένων αποτελεί το πιο βασικό κομμάτι της εργασίας αυτής. Σκοπός είναι μέσω αυτής οι χρήστες να μπορούν να αντιληφθούν τις σχέσεις με δεδομένα από διάφορες βάσεις δεδομένων, έτσι ώστε να είναι κατανοητή μέσω γραφημάτων.

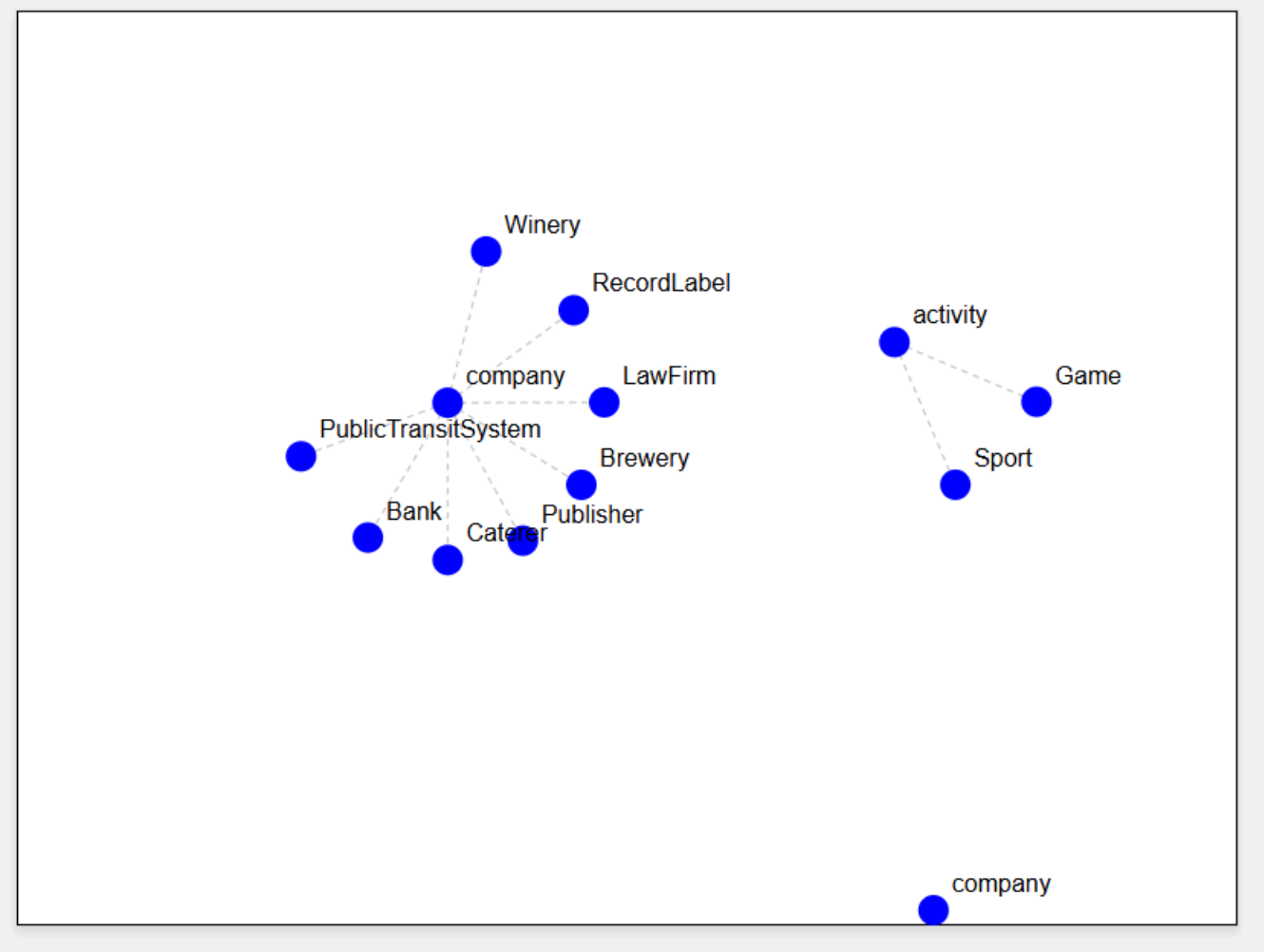
3.4.1.1 Χρήση D3.js

Η βιβλιοθήκη D3.js είναι ένα από τα πιο ισχυρά εργαλεία για την οπτικοποίηση δεδομένων στον παγκόσμιο ιστό. Έχει γραφτεί σε JavaScript και επιτρέπεται να δημιουργηθούν δυναμικά διαδραστικά γραφήματα αλλά και άλλες μορφές αναπαράστασης των δεδομένων. Αυτό γίνεται δυνατότητα μέσω της χρησης του SVG και άλλων web-based τεχνολογιών. Στον κώδικα της εργασίας, χρησιμοποιείται για για την οπτικοποίηση των δεδομένων που προκύπτουν από το ερώτημα SPARQL. Η D3.js δημιουργεί διασυνδεμένων κόμβων, nodes, και γραμμών , links , τα οποία αντιπροσωπεύουν τις κλάσεις και τις υποκλάσεις. Δημιουργείται έτσι ένα γραφικό μοντέλο των σχέσεων μεταξύ αυτών. Η διαδικασία περιλαμβάνει την κατασκευή ενός γραφήματος, όπου κάθε κλάση αντιστοιχεί σε έναν κόμβο και ο σχέσεις υποκλάσης μετατρέπονται σε συνδέσεις μεταξύ κόμβων. Έτσι η μέθοδος επιτρέπει στους χρήστες να οπτικοποιήσουν τις σχέσεις με έναν πιο ευανάγνωστος και κατανοητό

τρόπο.

3.4.1.2 Οπτικοποίηση JSON response

Τα δεδομένα που ανακτώνται μέσω των SPARQL ερωτημάτων επιστρέφονται σε μορφή JSON. Στο κώδικα της εργασίας, οι απαντήσεις αναλύονται σε JSON και στην χρησιμοποιούνται για την δημιουργία ενός γραφήματος. Ο τρόπος που επεξεργάζονται και οπτικοποιούνται τα δεδομένα από το JSON είναι κρίσιμη για την επιτυχία της εφαρμογής. Η D3.js χρησιμοποιεί τα JSON δεδομένα για να δημιουργήσεις δυναμικά τους κόμβους και τις συνδέσεις, αναπαριστώντας τις κλάσεις και τις υποκλάσεις από το αποτέλεσμα. Η σωστή διαχείριση του JSON response και η οπτικοποίηση του επιτρέπει στο χρήστες να αντιληφθούν τις συσχετίσεις με πιο κατανοητό τρόπο.



σχήμα 3.5 Οπτικοποίηση Κλάσεων-υποκλάσεων από την πηγή DBpedia

3.4.1.3 Η σημασία των Nodes

Οι κόμβοι, nodes, αποτελούν τα θεμελιώδη στοιχεία του γραφήματος και αναπαριστούν τις κλάσεις και τις υποκλάσεις. Κάθε ένας κόμβος έχει δύο χαρακτηριστικά, πρώτον το id το οποίο είναι το URI της κλάσης, και την ετικέτες, label, που παρέχει μια πιο κατανοητή περιγραφή της κλάσης. Στον κώδικα, τα nodes δημιουργούνται από την επεξεργασία των δεδομένων JSON και αντιπροσωπεύουν τις κεντρικές οντότητες. Η σημασία των κόμβων είναι ότι αποτελούν τα σημεία αναφοράς στις σχέσεις

του γραφήματος. Κάθε ένας κόμβος έχει με άλλους μέσω των συνδέσεων (links), και αυτή διασύνδεση επιτρέπει στους χρήστες να κατανοήσουν τις διάφορες σχέσεις μεταξύ των κλάσεων.

3.4.1.4 Η σημασία των Lines

Οι γραμμές, lines ή links, του γραφήματος συνδέουν δύο κόμβους. Αναπαριστάται έτσι η σχέση μεταξύ των κλάσεων και των υποκλάσεων. Στον κώδικα, οι σχέσεις υποκλάσης αναπαρίστανται με links που συνδέουν την κλάση με τις υποκλάσεις της. Αυτές οι γραμμές έχουν υψίστει σημασία αφού παρέχουν την αναγκαία σύνδεση μεταξύ των κόμβων, επιτρέποντας στους χρήστες αν καταλαβουν την ιεραρχία και την κληρονομικότητα που μπορεί να υπάρχει. Η σωστή παρουσίαση αυτών των σχέσεων μέσω γραμμών, διευκολύνει τη διαδικασία της ανάλυσης των δεδομένων, και συμβάλλει στην περαιτέρω κατανόηση.

3.4.1.5 Λειτουργίες του γραφήματος

Το γράφημα, που δημιουργεί ο κώδικας της εργασίας έχει και μερικές λειτουργίες και δυνατότητες. Η βασικότερη είναι η αλληλεπίδραση που μπορεί να έχει ο χρήστης. Κρατώντας πατημένο το κουμπί του ποντικιού μπορεί να μετακινήσει τους κόμβους και οι διασυνδεδεμένοι με γραμμές άλλοι κόμβοι να ακολουθήσουν την πορεία του. Μπορεί έτσι κάποιος να δει περισσότερες ή λιγότερες πληροφορίες. Ακόμα η εξαγωγή δεδομένων του γραφήματος σε μορφή JSON είναι χρήσιμη για αποθήκευση ή για μελλοντική ανάλυση. Ακόμα δίνετε η δυνατότητα να εισάγουμε γραφήματα από εξωτερικά αρχεία JSON, επιτρέποντας τη φόρτωση δεδομένων για ανάλυση ή επεξεργασία. Αυτές οι λειτουργίες βοηθούν στην διαχείριση δεδομένων και στην ανάλυση της δομής του συνόλου. Καθιστά επίσης το γράφημα ακόμα πιο κατανοητο για τον χρήστη.

3.4.2 Import-Export JSON

3.4.2.1 Εισαγωγή

Στην παρών εργασία δίνετε ακόμη η δυνατότητα της εξαγωγής αλλά και τις εισαγωγής δεδομένων σε μορφή JSON. Αυτή η λειτουργία είναι ιδιαίτερα χρήσιμη όταν πρόκειται για την μεταφορά των δεδομένων μεταξύ των συστημάτων, δίχως την ανάγκη της ανάκτησης των δεδομένων από τον εξωτερική πηγή. Αυτό φυσικά γεννάει τον προορισμό ότι τα δεδομένα δεν είναι up to date. Μπορούν να δημιουργηθούν ακόμα αντίγραφα ασφαλείας, αλλά και να ανακτηθούν δεδομένα. Παρακάτω θα περιγραφούν οι λειτουργίες της εξαγωγής και της εισαγωγής δεδομένων, μέσω του ImportData.js και του ExportData.js αντίστοιχα.

3.4.2.2 Εισαγωγή Δεδομένων (Import)

Οι δύο βασικές λειτουργίες του ImportData.js υλοποιούνται μέσω των μεθόδων importGraphDataFromJson και setupImportButtonListener. Αυτές οι δύο μέθοδοι συνεργάζονται προκειμένου να δίνετε η δυνατότητα να φορτώνονται τα δεδομένα από αρχεία JSON στην εφαρμογή. Όταν ο χρήστης πατήσει το κουμπί “Επιλογή αρχείου”, εμφανίζονται τα τοπικά αρχεία του χρήστη. Παρέχεται μηχανισμός διαχείρισης των σφαλμάτων, για περιπτώσεις που ο χρήστης δεν επιλέξει καποιο αρχείο ή τα δεδομένα είναι μη έγκυρα.

Μέσω του FileReader γίνεται η ανάγνωση του περιεχομένου αρχείου ως κείμενο. Τα δεδομένα μετατρέπονται έπειτα απο JSON σε αντικείμενο JavaScript μέσω της JSON.parse. Εφόσων τα δεδομένα είναι έγκυρα, τότε εκτελείται η callback μέθοδος για να συνεχίσει η επεξεργασία.

Παρακάτω παρατίθεται ο κώδικας.

```
export function importGraphDataFromJson(event, callback) {
    const file = event.target.files[0];
    if (file) {
        const reader = new FileReader();
        reader.onload = function(e) {
            try {
                const jsonData = JSON.parse(e.target.result);
                console.log("Imported data:", jsonData);
                if (Array.isArray(jsonData)) {
                    callback(jsonData);
                } else {
                    console.error('Imported data is not an array');
                }
            } catch (error) {
                console.error('Error parsing the file:', error);
            }
        };
        reader.readAsText(file);
    } else {
        console.error('No file selected');
    }
}
```

Η λειτουργία της μεθόδο setupImportButtonListener είναι να ορίσει ένα ακροατή γεγονότων, event listener. Το κουμπί όπου ορίζεται ο event listener έχει συγκεκριμένο ID. Έτσι επιτρέπεται η απρόσκοπτη σύνδεση των λειτουργιών εισαγωγής με το User Interface.

Παρακάτω είναι κώδικας.

```
export function setupImportButtonListener(buttonId, callback) {
```

```
    const importButton = document.getElementById(buttonId);
    if (importButton) {
        importButton.addEventListener('change', function(event) {
            importGraphDataFromJson(event, callback);
        });
    } else {
        console.error('Import button not found');
    }
}
```

Με τη χρήση αυτών των συναρτήσεων, η εφαρμογή διασφαλίζει εύκολη και ασφαλή εισαγωγή δεδομένων από JSON αρχεία, με σκοπό να δημιουργηθεί ένας γράφος που απεικονίζει τα δεδομένα και συνδέσεις τους.

3.4.2.3 Εξαγωγή Δεδομένων (Export)

η Εξαγωγή των δεδομένων υλοποιείται με την χρήση της μεθόδου exportGraphDataToJson. Αυτή η μέθοδος παίρνει ως είσοδο δεδομένα, τα μετατρέπει σε μορφή JSON και τα αποθηκεύει σε ένα αρχείο. Τα δεδομένα που έχουν μετατραπεί σε μορφή JSON έχουν στοίχιση 2 κενών. Δημιουργείται ένα Blob, που περιέχει το JSON. Ο χρήστης κατεβάζει το αρχείο, και έπειτα γίνεται ο καθαρισμός των πόρων.

Παρακάτω είναι ο κώδικας

```
 export function exportGraphDataToJson(data, filename = 'graphData.json') {
    const json = JSON.stringify(data, null, 2);
    const blob = new Blob([json], { type: 'application/json' });
    const url = URL.createObjectURL(blob);
    const link = document.createElement('a');
    link.href = url;
    link.download = filename;
    link.click();
    URL.revokeObjectURL(url);
}
```

Έτσι ο χρήστης, μπορεί να κατεβάσει τα δεδομένα που έχει εξάγει σε δομή JSON, μόνο με ένα πάτημα του κουμπιού “Export Graph Data”.

σχήμα 3.6 Παράδειγμα εξαγωγής δεδομένων

3.4.2.4 Συμπέρασμα.

Η υλοποίηση αυτών των μεθόδων, εισαγωγής και εξαγωγής δεδομένων σε μορφή JSON δίνει μια ευελιξία στην εφαρμογή. Η εξαγωγή των δεδομένων επιτρέπει την δημιουργία των αντιγράφων ασφαλείας και τη μεταφορά της πληροφορίας. Παράλληλα η εισαγωγή των δεδομένων διευκολύνει την απεικόνιση των δεδομένων. Με την κατάλληλη διαχείριση των σφαλμάτων και το φιλικό προς τον χρήστη, αυξάνεται η χρηστικότητα και η αξιοπιστία της εφαρμογής. Τέλος, το γεγονός ότι χρησιμοποιείται το JSON ως φόρμα δεδομένων, εξασφαλίζει την ευρεία χρήση συμβατότητας και ευκολία επεξεργασίας από άλλες πλατφόρμες και συστήματα.

3.4.3 Screenshot

Η εφαρμογή δίνει την δυνατότητα να γίνει λήψη ενός στιγμιοτύπου. Αποτελεί ένα χρήσιμο χαρακτηριστικό που επιτρέπει στο χρήστες να αποθηκεύσω μια εικόνα του γραφήματος. Αυτό γίνεται πραγματικότητα με την χρήση της βιβλιοθήκης JavaScript dom-to-image. Η μέθοδος takeScreenshot κάνει λήψη ενός στοιχείου HTML και την αποθηκεύει ως ένα αρχείο εικόνας τύπου JPEG. Η διαδικασία που ακολουθείται είναι, πρώτα, αναγνωρίζει την το στοιχείο με ID graph-container, το οποίο περιέχει το γράφημα.Έπειτα, γίνεται ένας έλεγχος προκειμένου να διαπιστωθεί αν υπάρχει το στοιχείο, και σε περίπτωση αποτυχίας εμφανίζεται ένα μήνυμα σφάλματος. Στην συνέχεια Με την χρήση της domtoimage.toJpeg, το στοιχείο μετατρέπεται σε εικόνα JPEG. Σε αυτή την εικόνα ορίζετε ένας λευκό φόντος μεσω της παραμέτρου bgcolor. Τέλος δημιουργείται ένας προσωρινός σύνδεσμος <a>, προκειμένου να χρησιμοποιηθεί για την λήψη της εικόνας με όνομα graph_screenshot.jpeg.

Παρακάτω παρατίθεται ο κώδικας που χρησιμοποιήθηκε:

```
function takeScreenshot() {
  const graphElement = document.getElementById('graph-container');


  if (!graphElement) {
    console.error('Graph container not found!');
    return;
  }


  // Capture screenshot as JPEG with white background
  domtoimage.toJpeg(graphElement, {
      quality: 0.9,  // JPEG format with 90% quality
      bgcolor: '#ffffff'  // Set background color to white
  })
    .then((dataUrl) => {
```

```
      const link = document.createElement('a');
      link.href = dataUrl;
      link.download = 'graph_screenshot.jpg';
      link.click();
    })
    .catch((error) => {
      console.error('Error capturing screenshot:', error);
    });
}
```

Για να εκτελεστεί η μέθοδος όταν ο χρήστης πατήσει το κουμπί, προστίθεται και ένας event listener. Το κουμπί καλεί την μέθοδο takeScreenshot κάθε φορά που κάνει κλικ το κουμπί ο χρήστης.

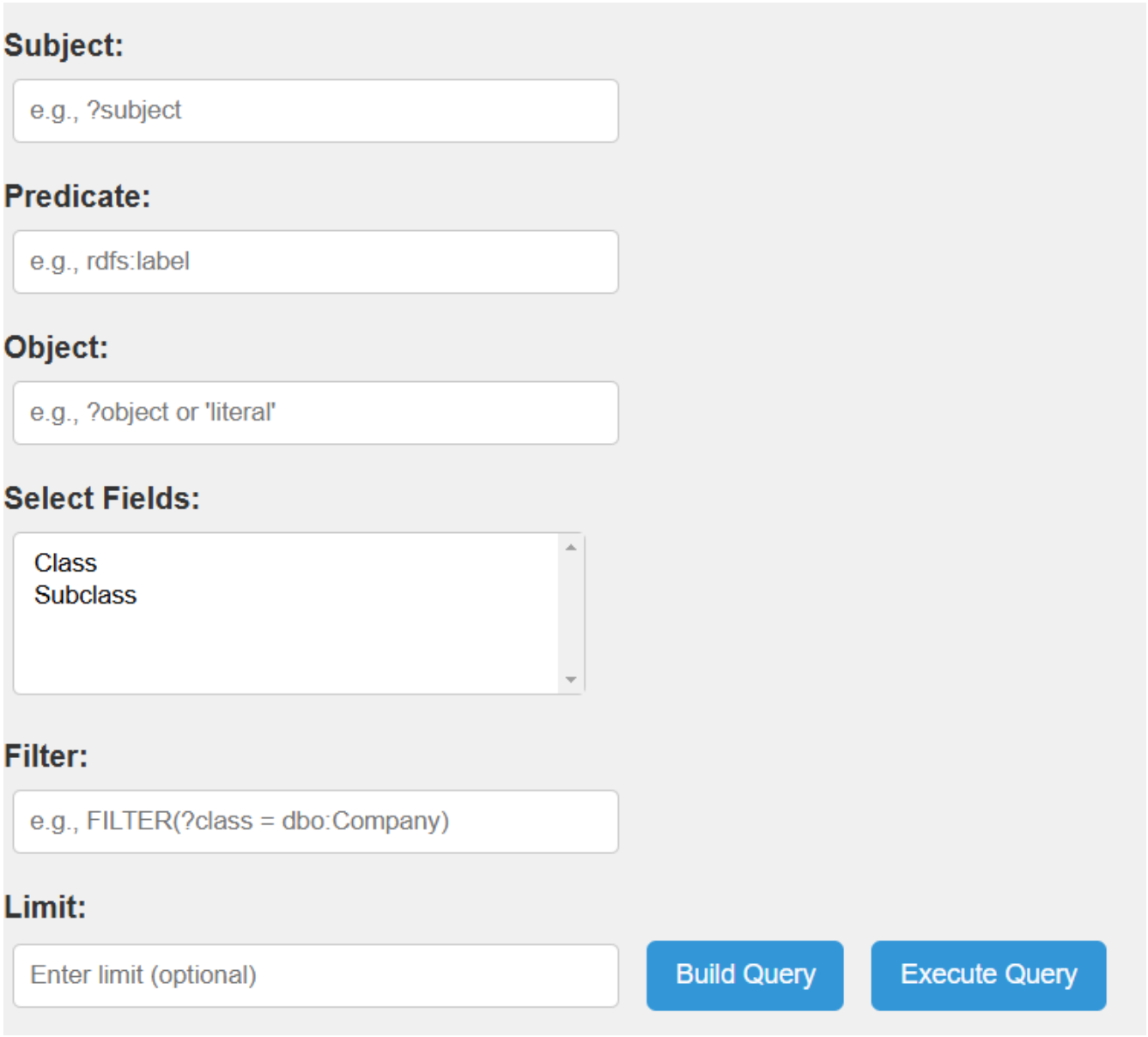
Παρακάτω παρατίθεται ο κώδικας που χρησιμοποιήθηκε:

```
document.getElementById('screenshotButton').addEventListener('click', takeScreenshot);
```

Τα πλεονεκτήματα που προσδίδει αυτή η λειτουργία είναι ότι οι χρήστες μπορούν να αποθηκεύσουν απευθείας την οπτική απεικόνιση του γραφήματος. Το στιγμιότυπο αποτελεί έναν γρήγορο τρόπο διατήρησης της τρέχουσας κατάσταστασης του γραφήματος. Η χρήση της μορφής JPEG εξασφαλίζει συμβατότητα με όλες τις συσκευές και εφαρμογές προβολής εικόνας. Έτσι αναβαθμίζεται η εμπειρία του χρήστη, κάνοντας την εφαρμογή πιο πρακτική και φιλική προς αυτόν.

3.4.4 Query Builder

Ακόμη στην παρών εφαρμογή δίνετε στον χρήστη η δυνατότητα να χρησιμοποιήσει το εργαλείο Query Builder. Πρόκειται για ένα εργαλείο που επιτρέπει στον χρήστη να δημιουργει SPARQL ερωτήματα. Το Query Builder αποτελείται από πεδία, που περιμένουν από τον χρήστη να τα συμπληρώσει και να σχηματιστεί στην εσωτερική λογική η SPARQL ερώτηση. Αποτελείται από τα βασικά πεδία Subject, Predicate και Object. Ακόμα ο χρήστης επιλέγει αν πρόκειται για κλάση ή υποκλάση, αν θέλει μπορεί να συμπληρώσει και κάποιο Φίλτρο αλλά και το Limit. Δίνεται ακόμα η δυνατότητα να αποθηκευτεί και η ερώτηση που γίνεται αλλά ακόμα και η απάντηση που επιστρέφετε.



σχήμα 3.7 Query Builder της εργασίας

3.4.4.1 Πεδία

Τα πεδία που συμπληρώνει ο χρήστης έχουν πρωταγωνιστικό ρόλο στην δημιουργία του ερωτήματος. Η μέθοδος formatVariable διασφαλίζει ότι κάθε πεδίο είναι σωστα διαμορφωμενο, προσθέτοντας το συμβολο ? όπου απαιτείται. Αυτές οι μεταβλητές συγκεντρώνονται και χρησιμοποιούνται για να δημιουργηθεί το ερώτημα SPARQL.

Παρακάτω δίνεται ο κώδικας:

```
function formatVariable(input) {

  const trimmedInput = input.trim();

  return trimmedInput.startsWith('?') ? trimmedInput : `?${trimmedInput}`;
```

```
}
```

3.4.4.3 Build Query – Execute Query

Η διαδικασία δημιουργίας και εκτέλεσης των ερωτημάτων χωρίζεται σε δύο μέρη.

Πρώτα είναι το στάδιο του Build Query. Η μέθοδος buildQuery δημιουργεί το ερώτημα SPARQL με βάση τα πεδία, τα φίλτρα αλλά και το όριο των αποτελεσμάτων. Τα επιλεγμένα πεδία συνδέονται με ένα SELECT DISTINCT. Συνδυάζονται ακόμα τα φίλτρα, από το αντίστοιχο πεδίο, και εάν υπάρχει, προστίθενται και ο περιορισμός του πλήθους των αποτελεσμάτων, με μια πρόταση LIMIT.

Παρακάτω παρατίθεται ο κώδικας.

```
function buildQuery() {
  const selectedFields = Array.from(document.getElementById('selectFields').selectedOptions)
                        .map(option => formatVariable(option.value))
                        .join(' ');
  const filter = document.getElementById('queryFilter').value.trim();
  const limit = document.getElementById('queryLimit').value.trim();

  let query = `
    SELECT DISTINCT ${selectedFields} WHERE {
      ${selectedFields.includes('?class') ? '?class a owl:Class .' : ''}
      ${selectedFields.includes('?subclass') ? '?subclass rdfs:subClassOf ?class .' : ''}
      ${filter ? filter : ''}
    }
    ${limit ? `LIMIT ${limit}` : ''}
  `;

  document.getElementById('queryPreview').textContent = query.trim();
}
```

Το επόμενο στάδιο είναι αυτό του Execute Query. Η μέθοδος executeQuery εκτελεί το ερώτημα μέσω ενός POST αιτήματος στον SERVER. Τα αποτελέσματα εμφανίζονται στον χρήστη, σε μορφή λίστας, και αν πράγματι επιστραφούν δεδομένα δημιουργούνται δύο κουμπιά, ένα αποθήκευσης του Query και ένα άλλο αποθήκευσης των Data.

3.4.4.4 Save Query – Save Data

Η εφαρμογή δίνει την επιλογή αποθήκευσης του Query και του Data, μέσω των δύο αντίστοιχων κουμπιών.

Παρακάτω παρατίθεται ο κώδικας για το Save Query:

```
function saveQuery(query) {
  const blob = new Blob([query], { type: 'text/plain' });
  const link = document.createElement('a');
  link.href = URL.createObjectURL(blob);
  link.download = 'query.txt';
  link.click();
  URL.revokeObjectURL(link.href);
}
```

Παρακάτω παρατίθεται ο κώδικας για το Save Data:

```
function saveData(data) {
  const jsonData = JSON.stringify(data, null, 2);
  const blob = new Blob([jsonData], { type: 'application/json' });
  const link = document.createElement('a');
  link.href = URL.createObjectURL(blob);
  link.download = 'data.json';
  link.click();
  URL.revokeObjectURL(link.href);
}
```

Πρέπει να σημειωθεί ότι τα κουμπιά αποθήκευσης δημιουργούνται δυναμικά. Προκειμένου να εμφανίζονται και να διασφαλίζουν ότι είναι διαθέσιμα μόνο όταν υπάρχουν δεδομένα προς αποθήκευση.

Παρακάτω παρατίθεται ο κώδικας της δημιουργίας των δύο κουμπιών.

```
// Create "Save Query" button
  const saveQueryButton = document.createElement('button');
  saveQueryButton.id = 'saveQueryButton';
  saveQueryButton.innerText = 'Save Query';
  saveQueryButton.addEventListener('click', () => saveQuery(query));
  buttonContainer.appendChild(saveQueryButton);

  // Create "Save Data" button
  const saveDataButton = document.createElement('button');
  saveDataButton.id = 'saveDataButton';
  saveDataButton.innerText = 'Save Data';
  saveDataButton.addEventListener('click', () => saveData(data));
  buttonContainer.appendChild(saveDataButton);
}
```

3.4.4.5 Συμπέρασμα

Το Query Builder είναι ένα εργαλείο για την δημιουργία και εκτελεση SPARQL ερωτημάτων. Παρέχει ένα απλό και εύχρηστο περιβάλλον για την δημιουργία ερωτημάτων και την εκτέλεσή τους. Δίνετε η δυνατότητα να διαχειρίζονται τα αποτελέσματα. Η αποθήκευση των ερωτημάτων και των δεδομένων διευκολύνει τους χρήστες, διασφαλίζοντας την ευελιξία και την αποτελεσματικότητά της.

3.5 Επίλογος.

Το κεφάλαιο εστίασε στο Frontend και στην χρήση της JavaScript. Είχε σκοπό να αναδείξει την ανάπτυξη της διαδραστικότητας και των λειτουργιών των Web εφαρμογών. Αναγράφεται η βασική δομή της HTML και της βελτίωσης της αισθητικής μέσω του CSS, μέχρι την υλοποίηση των προηγμένων λειτουργιών με JavaScript. Ενσωματώνονται εργαλεία όπως το D3.js προκειμένου να γίνει οπτικοποίηση των δεδομένων. Μπορούν να γίνει εξαγωγή και εισαγωγή αρχείων μορφής JSON, η δυνατότητα αποθήκευσης στιγμιοτύπων, και να δημιουργηθούν SPARQL Queries με την βοήθεια του Query Builder.

Εν κατακλείδι οι τεχνολογίες που υπάρχουν στην παρούσα εργασία ενσωματώνουν ισχυρά εργαλεία και βιβλιοθήκες για την δημιουργία μιας φιλικής προς τον χρήστη, χωρίς να αφαιρείται η λειτουργικότητα. Ο συνδιασμος των τεχνολογιών προσφέρει βάσεις για την ανάπτυξη εφαρμογών Web που ανταποκρίνονται στις απαιτήσεις της σύγχρονης εποχής. Δίνεται ιδιαίτερη έμφαση στην διαδραστικότητα, στην χρησιμότητα, στην ευχρηστία και την αποτελεσματική διαχείριση των δεδομένων.

4. Συμπέρασμα.

Η παρών εργασία ανέλυσε και παρουσίασε τις σύγχρονες τεχνολογίες και εργαλεία που χρησιμοποιούνται για την ανάπτυξη εφαρμογών Σημασιολογικού Ιστου. Δίνετε περισσότερη έμφαση στην οπτικοποίηση και την διαχείριση των δεδομένων. Από την εργασία προκύπτουν μερικά συμπεράσματα.

Αρχικά η SPARQL αναδεικνύεται ως ένα από τα πιο βασικά εργαλεία για την εξαγωγή και την διαχείριση RDF δεδομένων. Προσφέρει δυνατότητες όπως η εκτέλεση ερωτημάτων και παρουσίαση σχέσεων των δεδομένων. Ο Σημασιολογικός Ιστός, μαζί με τα πρότυπα του Linked Data, παρέχουν τις βάσεις για δομημενη και κατανοητή μορφή δεδομένων στο διαδίκτυο, είτε από ανθρώπους αλλά και είτε από μηχανές.

Ακόμα το NestJS, όπου είναι βασισμένο στο Node.js, αποτελεί μια πανίσχυρη επιλογή για την ανάπτυξη APIs και την επικοινωνία με εξωτερικές πηγές SPARQL. Η TypeScript προσφέρει ασφάλεια και σταθερότητα, παρόλου που υπάρχουν διαφορές με την JavaScript.

Οι τεχνολογίες του Front-end, όπως η D3.js, προσφέρουν δυνατότητες οπτικοποίησης, κάνοντας έτσι πιο κατανοητά τα σύνθετα δεδομένα για τους χρήστες. Επιπλέον οι λειτουργίες όπως import και export, η λήψη στιγμιότυπων και η δυναμική κατασκευή ερωτημάτων συμβάλλουν στην ομαλή και εύχρηστη χρήση της εφαρμογής από τους χρήστες.

Τέλος ο συνδυασμός όλων των τεχνολογιών, δημιοτφεί μια πληρος εφαρμογή που είναι διαλειτουργική μεταξύ backend και frontend, ενώ παράλληλα αξιοποιεί τις δυνατότητες του Σημασιολογικού Ιστού.

Βιβλιογραφία:

[1] Berners-Lee, T., et al. (2001). *The Semantic Web: A New Form of Web Content That Is Meaningful to Computers*. Scientific American.
[2] Gruber, T. R. (1993). *A Translation Approach to Portable Ontology Specifications*. Knowledge Acquisition.
[3] Manning, C. D., & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

[4] Berners-Lee, T. (2001). *The Semantic Web*. Scientific American, 284(5), 34–43.

[5] Rodríguez, M. A., & Groza, T. (2007). *Structured data in the Web*. IEEE Internet Computing, 11(4), 48–55.

[6] Shadbolt, N., Berners-Lee, T., & Hall, W. (2006). *The semantic web revisited*. IEEE Intelligent Systems, 21(3), 96–101.

[7] Gruber, T. R. (1993). *A translation approach to portable ontology specifications*. Knowledge Acquisition, 5(2), 199–220.

[8] Prud'hommeaux, E., & Seaborne, A. (2008). *SPARQL Query Language for RDF*. World Wide Web Consortium (W3C). Retrieved from https://www.w3.org/TR/rdf-sparql-query/

[9] Berners-Lee, T., & Fischetti, M. (2001). *The Semantic Web*. Scientific American, 284(5), 34-43. https://www.scientificamerican.com/article/the-semantic-web/

[10] McGuinness, D. L., & Harmelen, F. v. (2004). *OWL Web Ontology Language Overview*. W3C Recommendation. Retrieved from https://www.w3.org/TR/owl-features/

[11] Klyne, G., & Carroll, J. (2004). *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation. Retrieved from https://www.w3.org/TR/rdf-concepts/

[12] Fensel, D., & van Harmelen, F. (2001). The Semantic Web: An Introduction. Springer. https://doi.org/10.1007/978-3-642-56735-9

[13] G. Klyne, D. Beckett, and J. McCarroll (2004). Semantic Web Standards. Journal of Web Semantics, 1(1), 1-5. https://doi.org/10.1016/j.websem.2004.03.001

[14] Berners-Lee, T., & Fischetti, M. (2001). The Semantic Web. Scientific American, 284(5), 34-43. https://www.scientificamerican.com/article/the-semantic-web/

[15] Protégé official website: https://protege.stanford.edu/
Noy, N. F., & McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. Stanford Knowledge Systems Laboratory.

[16] Apache Jena official website: https://jena.apache.org/
Araujo, R. A., & Nascimento, M. (2018). A survey of the Apache Jena framework. *Journal of Computing and Information Technology*, 26(4), 255-267.

[17] RDF4J official website: https://rdf4j.org/
López, V., & García, D. (2016). RDF4J: A framework for working with RDF data in Java. *International Journal of Computer Science and Information Technologies*, 7(1), 118-123.

[18] GraphDB official website: https://www.ontotext.com/products/graphdb/
Fensel, D., & van Harmelen, F. (2011). *The Semantic Web: An Introduction*. Springer.

[19] OpenRefine official website: https://openrefine.org/
He, Q., & Zhang, Z. (2013). OpenRefine: A tool for data cleaning and transformation. Proceedings of the 2013 International Conference on Data Mining and Big Data, 134-139.

[20] Berners-Lee, T., & Fischetti, M. (2001). Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web. HarperSanFrancisco.

[21] Shadbolt, N., Berners-Lee, T., & Hall, W. (2006). The Semantic Web Revisited. IEEE Intelligent Systems, 21(3), 96-101.

[22] Alani, H., & Hitzler, P. (2017). The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management. Wiley.

[23] Berners-Lee, T. (2006). Linked Data. W3C.

[24] Bizer, C., & Seaborne, A. (2004). D2RQ - Database to RDF Mapping. In International Semantic Web Conference (ISWC).

[25] Berners-Lee, T., & Fischetti, M. (2001). Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor. Harper San Francisco.

[26] Bizer, C., & Cyganiak, R. (2009). The Resource Description Framework (RDF) and its Applications. Springer.

[27] Harris, S., & Seaborne, A. (2013). SPARQL 1.1 Query Language. W3C.

[28] Prud'hommeaux, E., & Seaborne, A. (2008). SPARQL Query Language for RDF. W3C Recommendation.

[29] Harris, S., & Seaborne, A. (2013). SPARQL 1.1 Query Language. W3C.

[30] V8 JavaScript Engine. (2021). Google. Retrieved from https://v8.dev/
Αυτή η αναφορά παρέχει μια πλήρη εξήγηση για την V8 μηχανή της Google, η οποία χρησιμοποιείται στο Node.js για τη γρήγορη εκτέλεση κώδικα JavaScript.

[31] Node.js Official Documentation. (2021). Node.js Foundation. Retrieved from https://nodejs.org/en/docs/

Η επίσημη τεκμηρίωση του Node.js, η οποία περιλαμβάνει πληροφορίες για την αρχιτεκτονική του, τις ασύγχρονες λειτουργίες και τη χρήση του event loop.

[32] npm Documentation. (2021). npm, Inc. Retrieved from https://docs.npmjs.com/
Η τεκμηρίωση του npm, του μεγαλύτερου διαχειριστή πακέτων για JavaScript, που παρέχει πρόσβαση σε χιλιάδες βιβλιοθήκες και modules για το Node.js.

[33] WebSocket API. (2021). Mozilla Developer Network. Retrieved from https://developer.mozilla.org/en-US/docs/Web/API/WebSocket
Το άρθρο αυτό εξηγεί την τεχνολογία WebSocket, η οποία είναι κεντρική για την ανάπτυξη real-time εφαρμογών με το Node.js.

[34] AWS Lambda: Serverless Computing. (2021). Amazon Web Services. Retrieved from https://aws.amazon.com/lambda/
Η επίσημη σελίδα του AWS Lambda, η οποία παρέχει serverless αρχιτεκτονικές και χρησιμοποιεί το Node.js για την ανάπτυξη μικρο-υπηρεσιών.

[35] Casciaro, M. (2014). *Node.js Design Patterns*. Packt Publishing.

[36] NestJS Documentation. Retrieved from https://docs.nestjs.com/

[37] Ihrig, C. (2015). *Pro Node.js for Developers*.

[38] Myśliwiec, K. (2021). *Mastering NestJS*. Packt Publishing.

[40] https://www.programmingcube.com/typescript-vs-javascript/

[41] https://restfulapi.net/

[42]https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Scripting/JSON

[43].IEEE Xplore, *Barriers in Front-End Web Development*, διαθέσιμο στο: IEEE

[44].*A Comparative Analysis of Modern Frontend Frameworks*, IEEE Conference Publication, διαθέσιμο στο: IEEE Xplore

[45].*Frontend Development Trends in 2023*, Computer.org, διαθέσιμο στο: Computer.org

[46].*Comparison and Analysis of Popular Frontend Frameworks and Libraries*, IEEE, διαθέσιμο στο: IEEE Xplore

[47]. W3C (World Wide Web Consortium). "HTML5." https://www.w3.org/html/

[48]. MDN Web Docs. "HTML: Hypertext Markup Language." https://developer.mozilla.org/en-US/docs/Web/HTML

[49]. Berners-Lee, T. "The World Wide Web." https://www.w3.org/History/19921103-hypertext/hypertext/WWW/News/1992/Jan/WWW.html