ΕΘΝΙΚΟΜΕΤΣΟΒΙΟΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝΜΗΧΑΝΙΚΩΝ
ΚΑΙΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ, ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Σημασιολογικός Ιστός & Διασυνδεδεμένα Δεδομένα σε Διεπαφές χρηστών & Λογικές Υπηρεσίες

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ιωάννης Κ. Βαλαδάκης

Επιβλέπων: **Δημήτριος Ασκούνης**
Καθηγητής ΕΜΠ

Αθήνα, Νοέμβριος 2019

ΕΘΝΙΚΟΜΕΤΣΟΒΙΟΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝΜΗΧΑΝΙΚΩΝ
ΚΑΙΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ, ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Σημασιολογικός Ιστός & Διασυνδεδεμένα Δεδομένα σε Διεπαφές χρηστών & Λογικές Υπηρεσίες

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ιωάννης Κ. Βαλαδάκης

Επιβλέπων: **Δημήτριος Ασκούνης**
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 21 Νοεμβρίου 2019.

.....	.....	.....
ΔΗΜΗΤΡΙΟΣ ΑΣΚΟΥΝΗΣ	ΨΑΡΡΑΣ ΙΩΑΝΝΗΣ	ΔΟΥΚΑΣ ΧΑΡΗΣ
Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Επίκουρος Καθηγητής ΕΜΠ

Αθήνα, Νοέμβριος 2019

.....

Βαλαδάκης Κ. Ιωάννης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Σε ένα περιβάλλον όπου ολοένα και περισσότερες ανάγκες για αυτοματοποίηση και ψηφιοποίηση υπηρεσιών και λειτουργιών γίνονται εφικτές βάση προγραμματιστικής ισχύς και τεχνογνωσίας, το παρόν και το μέλλον της διαβίωσης των δεδομένων δύναται να παρουσιαστεί ως σύνθεση κατ' ανάγκη ανεξάρτητων υπολογιστικών λύσεων που έχουν κατά ένα μέρος υλοποιηθεί από μια υπηρεσία ή έναν οργανισμό. Οι ψηφιοποιημένες καθημερινές ανάγκες έχουν πλέον αποτελέσει ένα αναπόσπαστο κομμάτι της ζωής μας. Κάτι το οποίο πριν λίγα χρόνια χαρακτηρίζονταν ως και φανταστικό έχει καταλήξει πλέον ένας απλός χειρισμός σε μια διαπροσωπεία.

Μεγάλοι οργανισμοί και εταιρείες πληροφορικής παρέχουν υπηρεσίες και δεδομένα τα οποία αποτελούν πηγές ανάγκης σε θέματα παρατήρησης και αλληλεπίδρασης με τους τελικούς χρήστες. Ένα αχανές διαδίκτυο με υπηρεσίες και πληροφορίες που προσφέρονται από προγραμματιστικές λογικές εταιρειών, αυτοματοποιημένων μονάδων και διασυνδεδεμένων υπολογιστών μέσω διεπαφών στους χρήστες έρχεται να επεκταθεί από την τεχνολογία του σημασιολογικού ιστού ο οποίος στοχεύει στη μοντελοποίηση της πληροφορίας ώστε αυτή να γίνει διεθνώς αναγνωρίσιμη από υπολογιστικές μονάδες.

Στην παρούσα μελέτη εξετάζεται και επεκτείνεται η χρήση του Σημασιολογικού Ιστού και των λειτουργιών του σε διαπροσωπείες και υπηρεσίες που συμβάλουν στην καθημερινότητα μας. Τίθεται το πρόβλημα της λειτουργίας των διασυνδεδεμένων δεδομένων σαν ανεξάρτητες μονάδες πληροφόρησης και λειτουργίας τους. Υποδεικνύεται ένας τρόπος με τον οποίο αυτές οι μονάδες μπορούν να αποκτήσουν χαρακτηριστικά που εξελίσσονται από μόνα τους η αναπαράγονται από τους χειρισμούς του χρήστη. Κύριος σκοπός της εργασίας είναι να παρουσιάσει ένα framework το οποίο διαχειρίζεται τα διασυνδεδεμένα δεδομένα και προσδίδει στον τελικό χρήστη ιδιότητες που εξυπηρετούν στην μοντελοποίηση και στην επέκτασή τους με έναν αυτοματοποιημένο τρόπο.

Πιο συγκεκριμένα θα αναπτύξουμε κάποιες παραγράφους με μια εγκυκλοπαιδική αναφορά σε βασικά στοιχεία του σημασιολογικού ιστού και πως η πληροφορία εκφράζεται στις υπάρχουσες τεχνολογίες. Θα περιγράψουμε κάποια από τα ιδιαίτερα χαρακτηριστικά των διασυνδεδεμένων δεδομένων και θα παρουσιάσουμε τα πλεονεκτήματα της αυτοματοποίησης και χρησιμοποίησης τους από τεχνολογίες του web. Θα γίνει μια αναφορά των λογικών αναγκών και εκφράσεων της καθημερινής ζωής και τρόπους με τους οποίους μπορεί να υλοποιηθούν υπολογιστικά. Θα εκφραστούν οι προβληματισμού σαν αντίκτυπο στις διεπαφές του χρήστη στο διαδίκτυο και πως αυτές μπορούν να εκφραστούν σε συλλογές από χαρακτηρισμένα δεδομένα.

Λέξεις κλειδιά: Σημασιολογικός Ιστός, Διασυνδεδεμένα Δεδομένα, Ανεξάρτητα διαλειτουργικά Δομικά στοιχεία Web, Λογικές Υπηρεσίες σε RDF, Front-End framework, Δυναμική προσέγγιση και διαχείριση δομικών στοιχείων, Τεχνολογίες Semantics σε διαπροσωπείες Χρηστών

Abstract

Semantic Data has been a part of web to structure the information provided from end points of any type and interact with users via interfaces. The Web through the ages has changed its form to be handled and managed by automated procedures and systems that use it. In current document we will mention some of the technologies provided in now days and how they apply on Linked Data Applications. In Chapter 1 we will mention some of the technologies that are in use from today's environments on computer science, linked data design issues and standards, their lifecycle on the web and some advantages they offer on automated procedures and artificial intelligence. Chapter 2 describes analog needs and logics and how they can be implemented by user interfaces and programmers. Also, it describes formats of JSON-LD, CSV & TSV and XML that can be used to collect and process the Linked Data. At the end we describe some issues on a programming approach. On Chapter 3 we discuss and present the architectural strategies and patters of a solution that uses Semantic technologies, and how they apply on a User Interface using JavaScript objects and functions. Chapter 4 provides an analysis of an implementation of a Books library example providing information gathered from DBpedia endpoint as a triplestore using schema.org entities and classes. It analysis basic views provided from a flux implementation and presents the basic components and functionality of gathering handling and constructing information of Linked Data. On Chapter 5 the document analyses the approach we offered and provides a conclusion of the generic implementation we presented. Finally, on Chapter 6 we mention expands and future attachments that can be added to the generic solution we offered. Target of current document is to analyze the usage of Open Data on the web and how they can be managed with a generic solution on an interface. It also describes a solution using flux architectural pattern of Facebook that complies with semantic infrastructure and needs and has been implemented for a thesis purpose.

Keywords: Semantic Web Components, Dynamic representation Front-End Framework, Logic Services and Business Logic to RDF, Semantics on User Interfaces, Semantics Front-End tools and framework, Linked Data representation Algorithm, Open Data dynamic Handling.

Περιεχόμενα

Κατάλογος Εικόνων

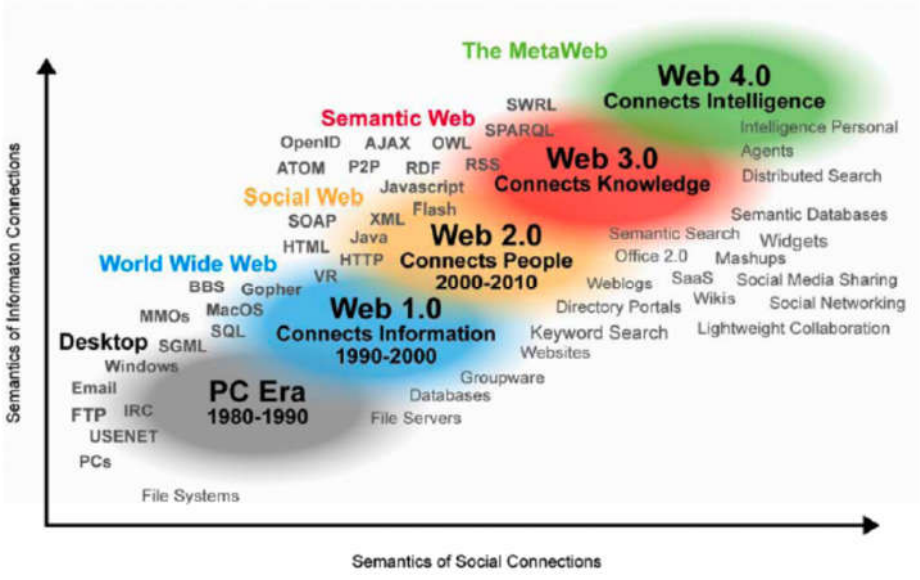
Διαγράμματα

1.Εισαγωγή

Το **Διαδίκτυο** ένα παγκόσμιο πλέγμα διασυνδεδεμένων υπολογιστών, υπηρεσιών και πληροφοριών που παρέχει στους χρήστες του ανταλλαγή δεδομένων μεταξύ οποιουδήποτε διασυνδεδεμένου υπολογιστή. Σαν επικοινωνιακό δίκτυο η τεχνολογία του είναι κυρίως βασισμένη στην διασύνδεση επιμέρους δικτύων ανά τον κόσμο και σε πολυάριθμα πρωτόκολλα επικοινωνίας. Ξεκινώντας από την κυβέρνηση των Ηνωμένων πολιτειών στα 60's ως μια προσπάθεια σταθερού δικτύου επικοινωνίας, έφτασε σε μόλις 50 χρόνια να αποτελεί ένα από τα πιο ποικιλόμορφα δίκτυα παροχής υπηρεσιών. Μεταφορά πολυμέσων, αρχείων, τηλεφωνία, διαδικτυακή τηλεόραση, ραδιόφωνο, social networks, εφημερίδες, blogs είναι μερικές από τις καθημερινές χρήσεις του και όχι μόνο.

Ο **παγκόσμιος ιστός** από την άλλη που στην καθημερινότητα μας παρερμηνεύεται και ταυτίζεται με το διαδίκτυο είναι στην πραγματικότητα μια υπηρεσία του. Ορίζεται ως ένα ανοιχτό σύστημα διασυνδεδεμένων πληροφοριών και πολυμεσικού περιεχομένου, που επιτρέπει στους χρήστες του Διαδικτύου να αναζητήσουν πληροφορίες μεταβαίνοντας από ένα έγγραφο στο άλλο. Κάθε δίκτυο-δομική μονάδα του διαδικτύου αποτελείται από συνδεδεμένους υπολογιστές σε τοπικό επίπεδο, που στις μέρες μας λειτουργούν και ως διεπαφές υπηρεσιών από συστήματα εταιριών και οργανισμών. Αυτά τα δίκτυα με τη σειρά τους συνδέονται σε ευρύτερα δίκτυα, όπως εθνικά και υπερεθνικά στο ευρύτερο μοναδικό δίκτυο στον κόσμο τον παγκόσμιο ιστό στο οποίο η πληθώρα δυνατοτήτων υπολογιστικής λογικής εξυπηρετεί σε συλλογικό βαθμό τις ανάγκες του πλανήτη συμπεριλαμβάνοντας τόσο τα γήινα δίκτυα, όσο και τα δίκτυα των δορυφόρων της και άλλων διαστημικών συσκευών που είναι συνδεδεμένα σε αυτό[1].

Η τεχνολογία του ιστού καθιστά δυνατή την δημιουργία "υπερκειμένων", μία διασύνδεση δηλαδή πάρα πολλών μη ιεραρχημένων στοιχείων που παλαιότερα ήταν απομονωμένα. Τα στοιχεία αυτά μπορούν να πάρουν και άλλες μορφές πέραν της μορφής του γραπτού κειμένου, όπως εικόνας και ήχου.

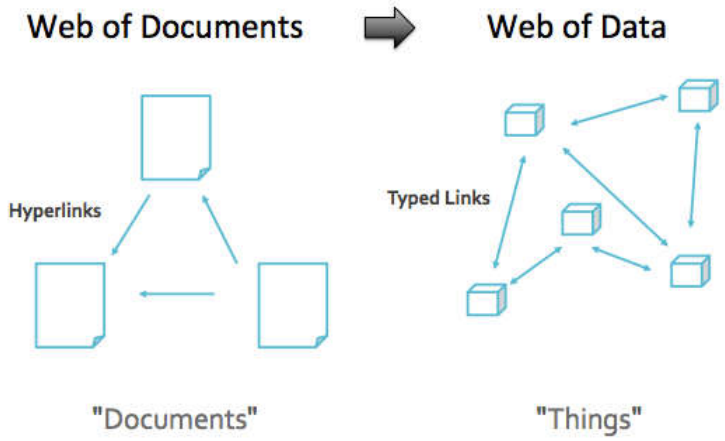


Εικόνα 1: Εξέλιξη Web & Τεχνολογίες

*«Ο αρχικός Παγκόσμιος Ιστός (**WEB 1.0**) συνέδεε δεδομένα και πληροφορίες που μπορούσαν να παρέχουν στο διαδίκτυο μόνο όσοι γνώριζαν HTML (Hyper Text Markup Language). Ο χρήστης του διαδικτύου ήταν παθητικός δέκτης των πληροφοριών. Αυτό άλλαξε με την έλευση του **WEB 2.0**, του διαδραστικού ιστού. Οι χρήστες του διαδικτύου μπορούν πλέον να αλληλοεπιδρούν μεταξύ τους, να ανταλλάσσουν πληροφορίες, να διαμορφώνουν το περιεχόμενο μιας ιστοσελίδας. Ο Web 2.0 συνδέει άτομα μεταξύ τους και είναι αποτέλεσμα κυρίως του εύκολου τρόπου κατασκευής ιστοσελίδων. Τα πρώτα WEB 2.0 εργαλεία είναι τα ιστολόγια (blogs), στα οποία ο χρήστης του διαδικτύου μπορεί να αφήνει σχόλια. Άλλα παραδείγματα WEB 2.0 εργαλείων είναι τα wikis και οι ιστοσελίδες κοινωνικής δικτύωσης.»*

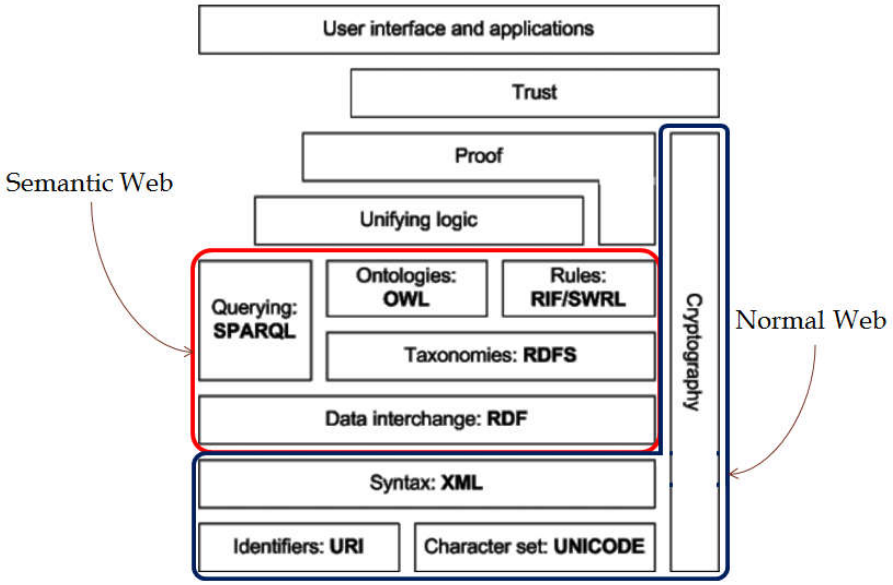
1.1 Semantic Web

Ο Σημασιολογικός Ιστός[2] (Web 3.0) είναι μια επέκταση του σημερινού Ιστού, υπόσχεται να φέρει δομή στο ουσιαστικό περιεχόμενο των ιστοσελίδων όπως παρουσιάζει ο σκοπός δημιουργίας του. Η λογική πίσω από αυτή την τεχνογνωσία είναι η δημοσιευμένη πληροφορία να περιέχει μεταδεδομένα, τα οποία όχι μόνο θα είναι κοινά για όλους, αλλά θα μπορούν να «κατανοούνται» και από μηχανές, οι οποίες θα βοηθήσουν στην καλύτερη συλλογή και επεξεργασία τους.



Εικόνα 2:Web of documents & Web of things

Ο Σημασιολογικός Ιστός βασίζεται σε τεχνολογίες που ήδη υπάρχουν (URI και XML) αλλά και σε νέες τεχνολογίες (RDF, RDFS, OWL, κα.), οι οποίες αναπτύσσονται με την βοήθεια της κοινότητας λογισμικού. Δεδομένου ότι ο νέος Ιστός σκοπεύει να είναι μια μεγάλη βάση όπου δεδομένα από διαφορετικά πεδία θα συνδέονται μεταξύ τους, αναμένεται να παίξει μεγάλο ρόλο στη ζωή μας. Το σημασιολογικό διαδίκτυο, όπως αρχικά οραματίστηκε, είναι ένα σύστημα το οποίο επιτρέπει στις μηχανές να "καταλαβαίνουν" τα πολύπλοκα ανθρώπινα αιτήματα βασισμένα στο νόημά τους. Τέτοιου είδους "κατανόηση" απαιτεί από τις συσχετιζόμενες πηγές πληροφοριών να έχουν μία σημασιολογική δομή.



Εικόνα 3: Semantics as Webs Middleware

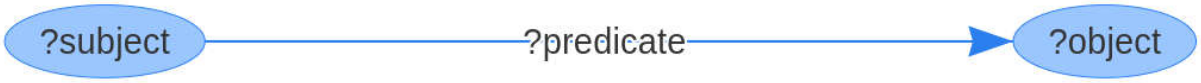
Ο όρος "Σημασιολογικό Διαδίκτυο" αναφέρεται συχνά σε μορφές και τεχνολογίες που μπορούν να το υποστηρίξουν. Η συλλογή των δομών δεδομένων που είναι συνδεδεμένα υποστηρίζονται από τις τεχνολογίες που παρέχουν μία τυπική περιγραφή εννοιών, όρων και σχέσεων μέσα σε ένα δεδομένο domain γνώσεων.

Αυτές οι τεχνολογίες έχουν καθοριστεί ως W3C τυποποίηση και περιέχουν:

- Resource Description Framework (RDF), μία γενική μέθοδο περιγραφής πληροφοριών
- RDF Schema (RDFS)
- Simple Knowledge Organization System (SKOS)
- SPARQL, ως μία RDF γλώσσα ερωτημάτων
- Notation3 (N3), σχεδιάστηκε με βάση την ανθρώπινη ικανότητα να διαβάζουμε
- N-Triples, μία μορφή για αποθήκευση και μετάδοση πληροφοριών
- Turtle (λιτή τριπλή γλώσσα RDF)
- Web Ontology Language (OWL), οικογένεια απεικόνισης γλωσσών γνώσης
- Rule Interchange Format (RIF), ένα πλαίσιο κανόνων γλωσσών του διαδικτύου για την υποστήριξη εναλλασσόμενων κανόνων.

1.1.1 Resource Description Framework (RDF)

Το RDF[3] είναι ένα πρότυπο μοντέλο για την ανταλλαγή δεδομένων στο Σημασιολογικό Ιστό, διαθέτει χαρακτηριστικά που διευκολύνουν τη συγχώνευση δεδομένων, ακόμη και αν τα υποκείμενα σχήματα διαφέρουν. Υποστηρίζει ειδικά την εξέλιξη των σχημάτων με την πάροδο του χρόνου χωρίς να απαιτείται η αλλαγή δεδομένων στα στοιχεία που συλλέγονται από τους χρήστες του διαδικτύου. Το RDF επεκτείνει τη δομή σύνδεσης του Ιστού χρησιμοποιώντας τα URI για να ονομάσει τις σχέσεις μεταξύ των πραγμάτων καθώς και τα δύο άκρα μοντέλα του συνδέσμου (αυτό συνήθως αναφέρεται ως "τριπλέτα").

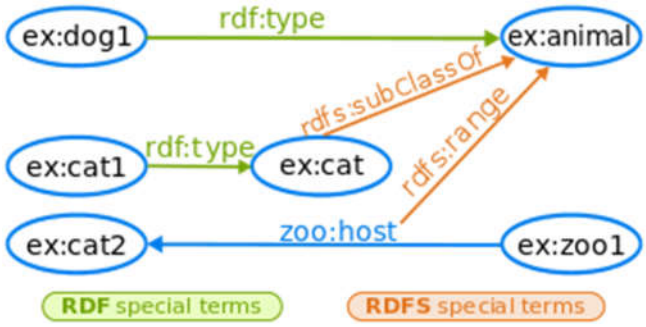


Εικόνα 4: A triple Sample

Χρησιμοποιώντας αυτό το απλό μοντέλο, επιτρέπει την ανάμειξη, έκθεση και κοινή χρήση δομημένων και ημιδομημένων δεδομένων σε διάφορες εφαρμογές. Αυτή η δομή διασύνδεσης μορφοποιεί ένα Γράφο όπου οι ακμές του αναπαριστούν την ονομασμένη σύνδεση 2 πόρων – κόμβων και οι κόμβοι του είναι στοιχεία IRIs, κενοί κόμβοι ή άλλα λεκτικά τύπων δεδομένων που χρησιμοποιούνται για να εκφράσουν περιγραφές των πόρων.

1.1.2 RDF Schema (RDFS)

Το RDFS[4] είναι ένα σύνολο από κλάσεις με ιδιότητες κατασκευασμένο έτσι ώστε να χρησιμοποιεί την αναπαράσταση γνώσης του RDF μοντέλου των δεδομένων και να προσφέρει περιγραφή σε βασικά στοιχεία τα οποία αναπαριστούν οντότητες ή αλλιώς λεξικά τα οποία στοχεύουν σε μια ευρύτερη μοντελοποίηση του εκάστοτε RDF πόρου. Οι πόροι οι οποίοι αποτελούν και ένα κέντρο πληροφόρησης είναι κατασκευασμένοι έτσι ώστε να μπορούν να διαχειριστούν και να μεταδώσουν δομές τριπλέτας με γλώσσα ερωτημάτων SPARQL. Οι υλοποιήσεις RDFS είναι και RDFS κλάσεις με συσχετισμένες ιδιότητες και άλλες ωφέλιμες ιδιότητες κατασκευασμένες σε RDF λεξικά.



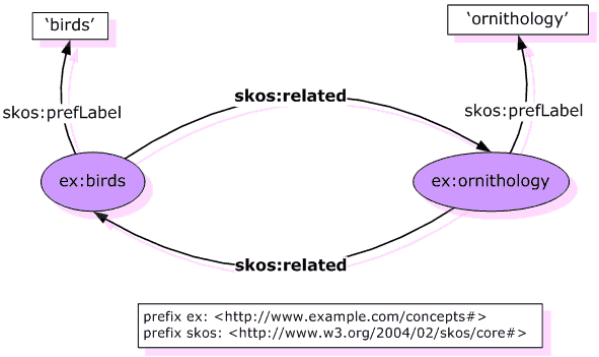
Εικόνα 5: Rdf Example

1.1.3 Simple Knowledge Organizations System (SKOS)

Το SKOS[5] είναι ένας τομέας ο οποίος σχεδιάστηκε για να αναπτύσσει προδιαγραφές και πρότυπα τα οποία στηρίζουν συστήματα οργάνωσης γνώσης. Παρέχει ένα κοινό τρόπο αναπαράστασης των συστημάτων αυτών (Knowledge Organization Systems) χρησιμοποιώντας το RDF. Η κωδικοποίηση της πληροφορίας σε RDF επιτρέπει στην πληροφορία να μεταδίδεται σε υπολογιστικές εφαρμογές με διαλειτουργικό τρόπο επιτρέποντας σε αυτά τα συστήματα να χρησιμοποιούνται σε περιγραφικές υλοποιήσεις και αποκεντρωμένων μεταδεδομένων εφαρμογές (distributed, decentralized metadata applications).

Η W3C ανακοίνωσε ένα νέο στάνταρ που υλοποίησε μια παγκόσμια γέφυρα μεταξύ των συστημάτων γνώσεων περιλαμβάνοντας εγκυκλοπαιδικούς θησαυρούς, ταξινομήσεις, επικεφαλίδες θεμάτων,

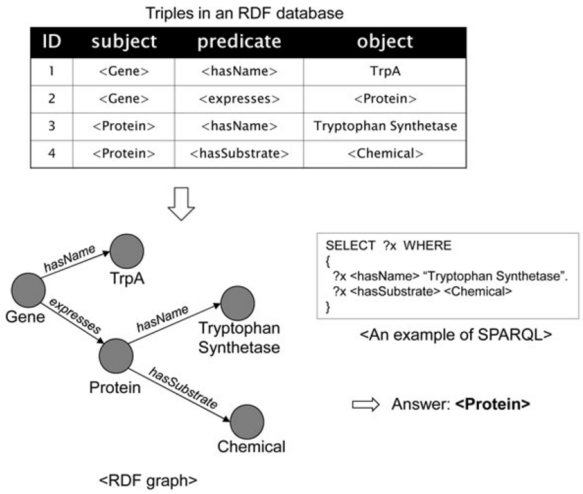
ταξινομίες, folksonomies, και την κοινότητα διασυνδεδεμένων, φέρνοντας προνόμια και στις δύο. Οι βιβλιοθήκες, τα μουσεία, οι εφημερίδες, οι κυβερνητικοί πόροι, οι επιχειρήσεις, οι πόροι κοινωνικής δικτύωσης και άλλες κοινότητες που διαχειρίζονται μεγάλες συλλογές βιβλίων, ιστορικών αντικειμένων, ειδησεογραφικών δελτίων, επιχειρηματικών γλωσσαρίων και άλλων αντικειμένων μπορούν πλέον να χρησιμοποιήσουν το SKOS για να αξιοποιήσουν τα διασυνδεδεμένα δεδομένα.



Εικόνα 6: Skos Example

1.1.4 SPARQL query language for RDF

Το RDF όπως προαναφέραμε είναι ένας κατευθυνόμενος Γράφος δεδομένων με ονοματισμένες κορυφές για την αναπαράσταση των πληροφοριών στο διαδίκτυο. Οι προσδιορισμοί αυτοί καθορίζουν την σύνταξη και τη σημασιολογία της γλώσσας ερωτήσεων SPARQL[6] για το RDF. Οι δυνατότητές της είναι ποικίλες καθώς δύναται να περιγράφει ποικιλόμορφες πηγές δεδομένων είτε τα δεδομένα αυτά προσφέρονται σε μορφές RDF άμεσα είτε περιγραφόμενα από κάποιο middleware. Μια από αυτές είναι να ρωτάει για πληροφορίες σε απαιτούμενα ή προαιρετικά μοτίβα γραφημάτων μαζί με τις συζεύξεις τους και τις διαχωριστικές τους δυνατότητες. Υποστηρίζει τη δοκιμή επέκταση τιμών και έχει την ικανότητα να περιορίζει τις ερωτήσεις σύμφωνα με τις πηγές RDF γραφημάτων. Τα αποτελέσματα των ερωτημάτων SPARQL μπορούν να είναι σύνολα αποτελεσμάτων ή γραφήματα RDF.



Εικόνα 7: Sparql definition

1.1.5 Notation3

Γνωστή και ως N3[7] η notation 3 είναι ένας ισχυρισμός και μια λογική γλώσσα υπερσύνολο του RDF. Επεκτείνει το RDF μοντέλο δεδομένων προσθέτοντας μια φόρμουλα μεταβλητών, λογικών υλοποιήσεων και συναρτησιακών προθεμάτων καθώς προμηθεύει συμφώνως με τη σύνταξη των εκφράσεων ένα εναλλακτικό RDF/XML.

Στόχοι της γλώσσας είναι:

- Να βελτιστοποιήσει την έκφραση των δεδομένων και τη λογική της ίδιας της γλώσσας
- Να επιτρέψει στο RDF να εκφραστεί
- Να επιτρέψει στους κανόνες να ολοκληρωθούν ομαλά με το RDF
- Να επιτρέψει παραθέσεις τ.ω. οι δηλώσεις δηλώσεων να είναι δυνατές
- Να είναι αναγνώσιμη, φυσική όσο και συμμετρική όσο είναι δυνατόν

Η γλώσσα το πετυχαίνει αυτό με τα ακόλουθα χαρακτηριστικά:

- Συντομογραφία των URI χρησιμοποιώντας προθέματα τα οποία είναι συνδεδεμένα με τοπωνύμια (@prefix) σαν την XML
- Επανάληψη ενός αντικειμένου για το ίδιο το αντικείμενο που περιγράφει με το πρόθεμα του κόμμα «,».
- Επανάληψη ενός προθέματος για το ίδιο το υποκείμενο χρησιμοποιώντας το semicolon «;»
- Bnode σύνταξη με βεβαιότητα ιδιοτήτων χρησιμοποιώντας μεταξύ των ιδιοτήτων το [and]
- Τυποποιείται αφήνοντας ένα N3 Γράφο να αναφέρεται μεταξύ άλλων N3 Γράφων χρησιμοποιώντας το {and}
- Μεταβλητές και ποσοτικοποιήσεις για να επιτρέψει στους κανόνες κ.α. να εκφραστούν
- Απλή και συνεπής γραμματική.

Παράδειγμα RDF – Notation 3

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://en.wikipedia.org/wiki/Tony_Benn">
    <dc:title>Tony Benn</dc:title>
    <dc:publisher>Wikipedia</dc:publisher>
  </rdf:Description>
</rdf:RDF>
```

```
@prefix dc: <http://purl.org/dc/elements/1.1/>.

<http://en.wikipedia.org/wiki/Tony_Benn>
  dc:title "Tony Benn";
  dc:publisher "Wikipedia".
```

Εικόνα 8: Notation 3 syntax

1.1.6 N-Triples

Είναι μια μορφή για την αποθήκευση και την μετάδοση δεδομένων. Πρόκειται για μια γραμματοσειρά μορφής σειριοποίησης απλού κειμένου για γραφήματα RDF και ένα υποσύνολο της μορφής Turtle. Το N-triples δεν πρέπει να μπερδεύεται με την Notation3 η οποία είναι ένα υπερσύνολο της Turtle.
Σχεδιάστηκε σαν μια απλούστερη μορφή του N3 και της Turtle, έτσι ώστε να είναι ευκολότερο για το λογισμικό να αναλύει και να παράγει. Ωστόσο, επειδή δεν υπάρχουν μερικές από τις συντομεύσεις που παρέχονται από στειροποιήσεις του RDF μπορεί να είναι δύσκολο να τυποποιηθούν μεγάλος όγκος δεδομένων με το χέρι και δύσκολο να διαβαστούν.

- RDF/XML

```
<rdf:RDF xmlns="http://xmlns.com/foaf/0.1/"
         xmlns:dc="http://purl.org/dc/terms/"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
  <Document rdf:about="http://www.w3.org/2001/sw/RDFCore/ntriples/">
    <dc:title xml:lang="en-US">N-Triples</dc:title>
    <maker>
      <Person rdf:nodeID="art">
        <name>Art Barstow</name>
      </Person>
    </maker>
    <maker>
      <Person rdf:nodeID="dave">
        <name>Dave Beckett</name>
      </Person>
    </maker>
  </Document>
</rdf:RDF>
```

- N-Triples

```
<http://www.w3.org/2001/sw/RDFCore/ntriples/> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ↵
   <http://xmlns.com/foaf/0.1/Document> .
<http://www.w3.org/2001/sw/RDFCore/ntriples/> <http://purl.org/dc/terms/title> "N-Triples"@en-US .
<http://www.w3.org/2001/sw/RDFCore/ntriples/> <http://xmlns.com/foaf/0.1/maker> _:art .
<http://www.w3.org/2001/sw/RDFCore/ntriples/> <http://xmlns.com/foaf/0.1/maker> _:dave .
_:art <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
_:art <http://xmlns.com/foaf/0.1/name> "Art Barstow".
_:dave <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
_:dave <http://xmlns.com/foaf/0.1/name> "Dave Beckett".
```

Εικόνα 9: Rdf & N-Triples

1.1.7 Turtle Syntax

Terse RDF Triple Language (Turtle)[8] είναι μια σύνταξη και μια δομή αρχείου η οποία εκφράζει το RDF μοντέλο δεδομένων. Είναι παρόμοια με τη SPAQL και την RDF γλώσσα ερωτήσεων. Το RDF αντιπροσωπεύει πληροφορίες χρησιμοποιώντας τη δομή της τριπλέτας η οποία περιλαμβάνει ένα θέμα, ένα πρόθεμα και ένα αντικείμενο. Κάθε στοιχείο στην τριπλέτα εκφράζεται ως ένα URI του ιστού και η turle παρέχει έναν τρόπο ομαδοποίησης τριών URI για να σχηματίσει την εκάστοτε τριπλέτα και παρέχει τρόπους για να συντομεύσει τέτοιες πληροφορίες όπως για παράδειγμα την εξαγωγή κοινών τμημάτων URI.

Όπως περιεγράφηκε και στις παραπάνω ενότητες ή σύνταξη Turtle είναι δυνατή και σε RDF/XML όπως και σε N3. Το παρακάτω σχήμα περιγράφει έναν RDF Γράφο από 4 τριπλέτες το οποίο είναι διαθέσιμο και σε N-triples σύνταξη.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://example.org/stuff/1.0/> .

<http://www.w3.org/TR/rdf-syntax-grammar>
  dc:title "RDF/XML Syntax Specification (Revised)" ;
  ex:editor [
    ex:fullname "Dave Beckett";
    ex:homePage <http://purl.org/net/dajobe/>
  ] .
```

```
<http://www.w3.org/TR/rdf-syntax-grammar> <http://purl.org/dc/elements/1.1/title> "RDF/XML
Syntax Specification (Revised)" .
<http://www.w3.org/TR/rdf-syntax-grammar> <http://example.org/stuff/1.0/editor> _:bnode .
_:bnode <http://example.org/stuff/1.0/fullname> "Dave Beckett" .
_:bnode <http://example.org/stuff/1.0/homePage> <http://purl.org/net/dajobe/> .
```

Εικόνα 10: Turtle Syntax

Ο MIME[9] τύπος της Turtle είναι «text/Turtle» και η κωδικοποίηση των χαρακτήρων του περιεχομένου της είναι πάντα UTF-8.

1.1.8 OWL – Web Ontology Language

Η OWL[10] είναι και αυτή με τη σειρά της μια γλώσσα αναπαράστασης γνώσης για τη συγγραφή οντοτήτων οι οποίες είναι ένας επίσημος τρόπος για να χαρακτηρίζονται ταξινομίες και καταταγμένα δίκτυα, και να ορίζουν ουσιαστικά τη δομή γνώσης σε διάφορους πόρους. Οι οντότητες αυτές στο διαδίκτυο έχουν μια ουσιώδη διαφορά με αυτές που παρουσιάζονται προγραμματιστικά από τον αντικειμενοστραφή προγραμματισμό. Όταν μια οντότητα περιγράφεται με σκοπό να περιγράψει μια πληροφορία στο διαδίκτυο δύναται να αλλάζει και να εξελίσσεται συχνά και τυπικά. Οι οντότητες αυτές είναι πολύ πιο ευέλικτες αφού είναι κατασκευασμένες να παρουσιάσουν πληροφορίες στο διαδίκτυο που πηγάζουν από ετερογενής πόρους ή άλλες πηγές πληροφόρησης.

OWL2 Functional Syntax

```
Ontology(<http://example.org/tea.owl>
  Declaration( Class( :Tea ) )
)
```

OWL2 XML Syntax

```
<Ontology ontologyIRI="http://example.org/tea.owl" ...>
  <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
  <Declaration>
    <Class IRI="Tea"/>
  </Declaration>
</Ontology>
```

Manchester Syntax

```
Ontology: <http://example.org/tea.owl>
Class: Tea
```

RDF/XML syntax

```
<rdf:RDF ...>
    <owl:Ontology rdf:about=""/>
    <owl:Class rdf:about="#Tea"/>
</rdf:RDF>
```

RDF/Turtle

```
<http://example.org/tea.owl> rdf:type owl:Ontology .
:Tea  rdf:type            owl:Class .
```

Εικόνα 11: Owl Examples

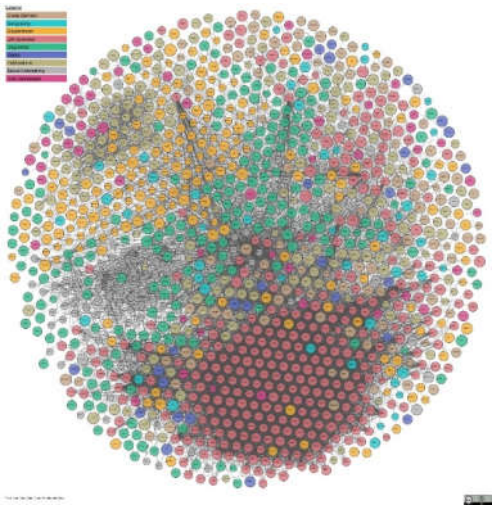
1.1.9 Rule Interchange Format (RIF)

Η ομάδα εργασίας του Rule Interchange Format[11] στελεχώθηκε από το World Wide Web Consortium το 2005 για να δημιουργήσει ένα στάνταρ για την ανταλλαγή κανόνων μεταξύ των συστημάτων, συγκεκριμένα μεταξύ των Web κανόνων των μηχανών. Το RIF επικεντρώθηκε στην ανταλλαγή περισσότερο παρά στο να προσπαθήσει να δημιουργήσει μια απλή μια προς όλες γλώσσα επειδή, σε αντίθεση με τα άλλα Semantics Web στάνταρ όπως το RDF, την OWL και την SPARQL, ήταν άμεσα ορατό ότι μια απλή γλώσσα δεν θα ικανοποιούσε τις ανάγκες των πολλών παραδειγμάτων για την χρήση της αναπαράστασης γνώσης σε επιχειρηματικά μοντέλα. Από μόνο του όμως ένα σύστημα ανταλλαγής κανόνων σήμανε αποθαρρυντικό. Τα γνωστά συστήματα κανόνων εμπίπτουν σε τρεις ευρείες κατηγορίες: πρώτης τάξης, λογικού προγραμματισμού και κανόνες δράσης. Αυτά τα παραδείγματα μοιάζουν λίγο στον τρόπο σύνταξης και σημασιολογίας. Παρόλα αυτά υπάρχουν μεγάλες διαφορές μεταξύ τους.

Η προσέγγιση αυτής της ομάδας εργασίας ήταν να σχεδιάσει μια οικογένεια από γλώσσες, τις επονομαζόμενες διαλέκτους (dialects), με αυστηρή σύνταξη και σημασιολογία. Η οικογένεια των διαλέκτων RIF προορίζεται να είναι ομοιόμορφη και επεκτάσιμη. Η ομοιομορφία RIF σημαίνει ότι οι διάλεκτοι αναμένεται να μοιράζονται όσο το δυνατόν περισσότερο την υπάρχουσα συντακτική και σημασιολογική σκευή. Η επεκτασιμότητα εδώ σημαίνει ότι θα πρέπει να είναι δυνατόν για τους εξειδικευμένους εμπειρογνώμονες να ορίσουν μια νέα διάλεκτο RIF ως μια συντακτική επέκταση σε μια υπάρχουσα διάλεκτο RIF, με νέα στοιχεία που αντιστοιχούν στην επιθυμητή πρόσθετη λειτουργικότητα. Αυτές οι νέες διαλέκτους RIF θα ήταν μη τυποποιημένες όταν καθορίστηκαν, αλλά θα μπορούσαν ενδεχομένως να γίνουν πρότυπα.

Λόγω της έμφασης στην αυστηρότητα, η μορφή λέξεων στο όνομα του RIF είναι κάπως υποτιμητική. Το RIF στην πραγματικότητα παρέχει κάτι παραπάνω από μια απλή μορφή. Ωστόσο, η έννοια του μορφοτύπου είναι απαραίτητη για τον τρόπο με τον οποίο πρόκειται να χρησιμοποιηθεί το RIF. Τελικά, το μέσο ανταλλαγής μεταξύ διαφορετικών συστημάτων κανόνων είναι η XML, μια μορφή ανταλλαγής δεδομένων. Βασικό στην ιδέα πίσω από την ανταλλαγή κανόνων μέσω του RIF είναι ότι διαφορετικά συστήματα θα παρέχουν συντακτικές αντιστοιχίσεις από τις μητρικές τους γλώσσες σε διαλέκτους RIF και ανάποδα. Αυτές οι αντιστοιχίσεις απαιτούνται για τη διατήρηση της σημασιολογίας και επομένως τα σύνολα κανόνων μπορούν να μεταδοθούν από το ένα σύστημα στο άλλο, με την προϋπόθεση ότι τα συστήματα μπορούν να μιλάνε μέσα από μια κατάλληλη διάλεκτο, την οποία υποστηρίζουν και οι δύο.

1.2 Διασυνδεδεμένα Δεδομένα



Εικόνα 12: Linked Data Ecosystem

Ο όρος **Linked Data** (διασυνδεδεμένα δεδομένα) περιγράφει μία μέθοδο δημοσιοποίησης δομημένων δεδομένων ώστε να είναι αλληλένδετα και να γίνουν πιο χρήσιμα. Η μέθοδος αυτή στηρίζεται στις γνωστές τεχνολογίες του Ιστού, όπως HTTP και URIs, αλλά αντί να τις χρησιμοποιεί για να εξυπηρετεί ιστοσελίδες για τους ανθρώπινους αναγνώστες, τις επεκτείνει ώστε να ανταλλάσσουν πληροφορίες με τρόπο που να μπορούν να διαβαστούν αυτόματα από τους υπολογιστές. Αυτό επιτρέπει δεδομένα από διαφορετικές πηγές να συνδέονται και να μπορούν να αναζητηθούν.

Ο στόχος του W3C Semantic Web Education and Outreach ομάδας Linking Open Data community project είναι να επεκταθεί ο Ιστός με κοινά στοιχεία δημοσιεύοντας διάφορα σύνολα δεδομένων ως RDF στον Ιστό

και θέτοντας RDF συνδέσεις μεταξύ των δεδομένων από διαφορετικές πηγές. Τον Οκτώβριο του 2007 τα σύνολα δεδομένων αποτελούνταν από παραπάνω από δύο δισεκατομμύρια RDF τριάδες, οι οποίες ήταν αλληλένδετες με πάνω από δύο εκατομμύρια RDF συνδέσεις. Από το Σεπτέμβριο του 2011 αυτό το μέγεθος αυξήθηκε σε 31 δισεκατομμύρια RDF τριάδες, αλληλένδετες με περίπου 504 εκατομμύρια RDF συνδέσεις.

1.2.1 Αρχές Σχεδίασης

Ο Τιμ Μπέρνερς Λι[12] σκιαγράφησε τέσσερις αρχές των Linked Data στο Design Issues: Linked Data σημείωμά του, παραφρασμένες παρακάτω:

- Χρησιμοποίηση των URIs για να προσδιοριστούν πράγματα.
- Χρησιμοποίηση των HTTP URIs έτσι ώστε να μπορούν να αναφερθούν σ' αυτά τα πράγματα και να μπορούν να αναζητηθούν ("dereferenced") από ανθρώπους και πράκτορες χρηστών.
- Δώσε χρήσιμες πληροφορίες σχετικά με το θέμα όταν το URI του ορίζεται εκ νέου χρησιμοποιώντας τυποποιημένους τύπους όπως RDF/XML.
- Συμπερίληψη συνδέσεων σε άλλα σχετικά URIs με τα εκτεθειμένα δεδομένα ώστε να βελτιωθεί η ανακάλυψή τους από άλλες σχετικές πληροφορίες στον Ιστό.

Ο Tim Berners-Lee έκανε μια παρουσίαση των linked data στο TED 2009 συνέδριο. Σε αυτό, επανέλαβε τις αρχές των linked data, όπως επίσης και τους τρεις "εξαιρετικά απλούς" κανόνες:

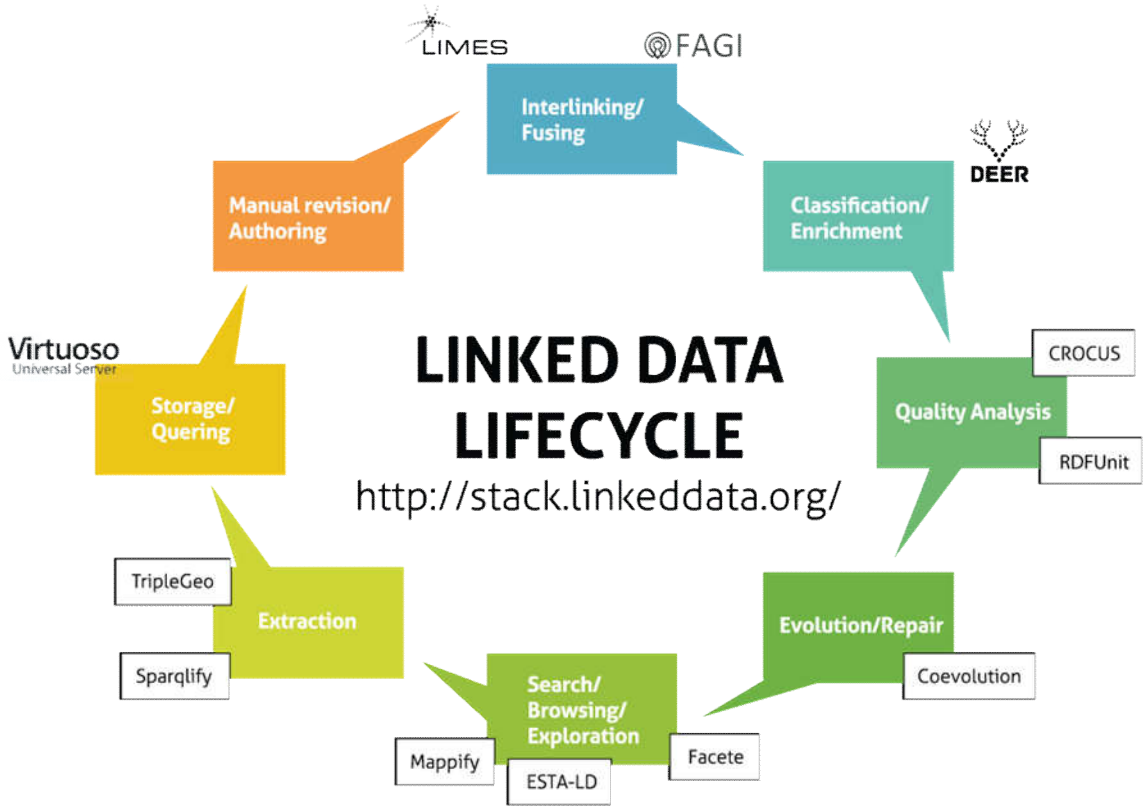
- Όλα τα είδη των εννοιολογικών πραγμάτων έχουν τώρα ονόματα που ξεκινούν με HTTP.
- Παίρνω πίσω σημαντικές πληροφορίες. Θα πάρω πίσω κάποια δεδομένα σε μια τυποποιημένη μορφή η οποία είναι χρήσιμα δεδομένα τα οποία κάποιος μπορεί θα ήθελε να ξέρει σχετικά με αυτό το πράγμα, σχετικά με αυτό το γεγονός.
- Παίρνω πίσω τέτοια πληροφορία που περιέχει όχι μόνο το ύψος κάποιου και το βάρος και που έχει γεννηθεί, αλλά και σχέσεις. Και όταν έχει σχέσεις, κάθε φορά που εκφράζει μια σχέση, τότε το άλλο πράγμα με το οποίο είναι συνδεδεμένο, του δίνεται ένα από αυτά τα ονόματα που ξεκινάν με HTTP.
- Σημειώστε ότι παρόλο που ο δεύτερος κανόνας αναφέρει τις "τυποποιημένες μορφές", δεν απαιτεί κάποιο συγκεκριμένο πρότυπο, όπως RDF/XML.

Τα δεδομένα που οι χρήστες ή οι εφαρμογές εκθέτουν στο διαδίκτυο και ακολουθούν τους παραπάνω κανόνες ακολουθούν ένα σύστημα αξιολόγησης σε σχέση με τη συμβατότητά τους στα κατασκευασμένα πρότυπα και κανόνες των διασυνδεδεμένων δεδομένων. Ο τρόπος βαθμολόγησης ακολουθεί τις παρακάτω αρχές δίνοντας αστέρια ανάλογά με την πληρότητα των κανόνων.

(*)	Τα δεδομένα προσφέρονται ανοικτά στο web.
(**)	Τα δεδομένα είναι δομημένα και αναγνωρίσιμα από τον υπολογιστή
(***)	Έκδοση των δεδομένων σε μη συγκεκριμένες δομές αλλά σε διεθνώς αναγνωρισμένες πχ . Plaintext, CSV vs Excel.
(****)	Η δημοσιοποίηση των δεδομένων θα πρέπει να έχει χαρακτηριστεί με τα πρότυπα του W3C. Θα πρέπει τα δεδομένα να έχουν ταυτοποιηθεί με συνέπεια, να είναι δυνατή η σύνδεση με άλλα.
(*****)	Τα δεδομένα είναι διασυνδεδεμένα με άλλους παρόχους. (triplets)

1.2.2 Κύκλος Ζωής Διασυνδεδεμένων Δεδομένων

Έχουν προταθεί διάφοροι τρόποι κατανόησης του κύκλου ζωής των διασυνδεδεμένων δεδομένων και των συστατικών τους βημάτων. Ο Sören Auer[13] πρότεινε έναν οκταετή κύκλο ζωής. Αυτός συνεπάγεται την αναζήτηση ή εύρεση πηγών από τις οποίες μπορούν να εξάγονται και να αποθηκεύονται δεδομένα, διασυνδέοντας αυτές τις πηγές δεδομένων, καθορίζοντας την ποιότητα του νέου συνόλου δεδομένων, διατηρώντας τα δεδομένα καθώς εξελίσσονται και καθιστώντας τα διαθέσιμα ως πηγή δεδομένων για περαιτέρω περιήγηση, αναζήτηση ή εξερεύνηση.



Εικόνα 13: Linked Data Lifecycle

Τα οκτώ στάδια αυτά είναι του κύκλου που προτάθηκαν είναι :

Storage. Η διαχείριση δεδομένων RDF είναι ακόμα δύσκολη σε σχέση με αυτή των σχεσιακών δεδομένων. Στόχος είναι να κλείσουμε αυτό το χάσμα απόδοσης χρησιμοποιώντας τεχνολογία columnstore, δυναμική βελτιστοποίηση ερωτημάτων, προσαρμοστική προσωρινή αποθήκευση συνδέσεων, βελτιστοποιημένη επεξεργασία γραφημάτων και δυνατότητα κλιμάκωσης cluster / cloud.

Authoring. Τα LOD2[14] διευκολύνουν τη δημιουργία πλούσιων βάσεων σημασιολογικής γνώσης με την τεχνολογία Semantic Wiki, το WYSIWYM[15] παράδειγμα (What You See Is What You Mean) και τη διανομή κοινωνικών πληροφοριών, τη σημασιολογική συνεργασία και τεχνικές δικτύωσης.

Interlinking. Η δημιουργία και η διατήρηση links με (ημι-) αυτοματοποιημένο τρόπο εξακολουθεί να αποτελεί μείζονα πρόκληση και κρίσιμη σημασία για τον καθορισμό της συνοχής και τη διευκόλυνση της ενσωμάτωσης των δεδομένων. Επιδιώκουμε προσεγγίσεις σύνδεσης που αποδίδουν υψηλή ακρίβεια και ανάκληση, οι οποίες επιβεβαιώνονται αυτόματα ή με ανατροφοδότηση του τελικού χρήστη.

Classification. Τα Linked Data στο διαδίκτυο είναι κατά κύριο λόγο γραμμές από στιγμιότυπα δεδομένων. Για την ολοκλήρωση των δεδομένων, τη σύντηξη, την αναζήτηση και πολλών άλλων εφαρμογών, παρόλα αυτά, χρειαζόμαστε αυτές τις γραμμές καταχωρήσεων να είναι συνδεδεμένες και να ολοκληρώνονται με οντολογίες ανώτερων επιπέδων.

Quality. Η ποιότητα του περιεχομένου στο Data Web ποικίλλει, καθώς η ποιότητα του περιεχομένου στον ιστό εγγράφων διαφέρει. Τα LOD2 αναπτύσσουν τεχνικές για την αξιολόγηση της ποιότητας με βάση χαρακτηριστικά όπως προέλευση, πλαίσιο, κάλυψη ή δομή.

Evolution/Repair. Τα δεδομένα στον Ιστό είναι δυναμικά. Πρέπει να διευκολύνουμε την εξέλιξη των δεδομένων διατηρώντας σταθερά τα πράγματα. Οι αλλαγές και οι τροποποιήσεις στις βάσεις γνώσεων, στα λεξιλόγια και στις οντολογίες πρέπει να είναι διαφανείς και παρατηρήσιμες. Το LOD2 αναπτύσσει επίσης μεθόδους εντοπισμού προβλημάτων στις βάσεις γνώσεων και αυτόματα προτείνει στρατηγικές επισκευής.

Search/Browsing/Exploration. Για πολλούς χρήστες, το Data Web εξακολουθεί να είναι αόρατο. Τα LOD2 αναπτύσσουν τεχνικές αναζήτησης, περιήγησης, εξερεύνησης και οπτικοποίησης για διάφορα είδη Linked Data (δηλ. Χωρικών, χρονικών, στατιστικών), τα οποία καθιστούν το Data Web λογικό για πραγματικούς χρήστες.

Αυτά τα στάδια του κύκλου ζωής, ωστόσο, δεν πρέπει να αντιμετωπιστούν μεμονωμένα, αλλά με τη διερεύνηση μεθόδων που διευκολύνουν την αμοιβαία γονιμοποίηση των προσεγγίσεων που αναπτύσσονται για την επίλυση προκλήσεων.

Ως αποτέλεσμα αυτής της αλληλεξάρτησης την οποία παρουσιάζουν τα διασυνδεδεμένα δεδομένα και οι υπολογιστικές τους προσεγγίσεις, θα πρέπει να επιδιώξουμε τη δημιουργία ενός κύκλου βελτίωσης για βάσεις γνώσεων στο Data Web. Η βελτίωση μιας βάσης γνώσεων σε σχέση με μία πτυχή (π.χ. μια νέα ευθυγράμμιση με έναν άλλο κόμβο διασύνδεσης) ενεργοποιεί μια σειρά πιθανών περαιτέρω βελτιώσεων (π.χ. πρόσθετες αντιστοιχίσεις εμφάνισης). Η πρόκληση είναι να αναπτυχθούν τεχνικές που επιτρέπουν την εκμετάλλευση αυτών των αμοιβαίων γονιμοποιήσεων στο διανεμημένο δίκτυο Web of Data. Μια δυνατότητα είναι ότι διάφοροι αλγόριθμοι κάνουν χρήση κοινών λεξιλογίων για τη δημοσίευση αποτελεσμάτων των ενεργειών χαρτογράφησης, συγχώνευσης, επιδιόρθωσης ή εμπλουτισμού.

Μετά τη δημοσίευση μιας υπηρεσίας τα νέα ευρήματά του σε ένα από αυτά τα ευρέως κατανοητά λεξιλόγια, οι μηχανισμοί κοινοποίησης (όπως το Semantic Pingback[16]) μπορούν να ειδοποιήσουν σχετικές άλλες υπηρεσίες ή το πρωτότυπο εκδότη δεδομένων, ότι υπάρχουν νέες προτάσεις βελτίωσης. Δεδομένης της σωστής διαχείρισης των στοιχείων προέλευσης, οι προτάσεις βελτίωσης μπορούν αργότερα (μετά την αποδοχή από τον εκδότη) να γίνουν μέρος του αρχικού συνόλου δεδομένων.

1.3 Πλεονεκτήματα αυτοματοποίησης Διασυνδεδεμένων δεδομένων

Οι πηγές δεδομένων και οι πόροι οι οποίοι προσφέρουν τόσο πληροφορίες όσο και δεδομένα για την υλοποίηση επιχειρησιακών λογικών και προγραμμάτων - υπηρεσιών αυτοματοποίησης προσφέρουν κατά ένα σημείο κοινά στοιχεία και πληροφορίες. Με το web of things η εναλλακτικά με το διαδίκτυο των πραγμάτων οι πληροφορίες αυτές παίρνουν μορφή τέτοια ώστε να καθίσταται εφικτή η αντίληψη των μηχανών στα δεδομένα. Σημαντικά πλεονεκτήματα και αυτοματοποιήσεις πολλών μορφών έρχονται στις μέρες μας και παρουσιάζονται σε πληροφορικές λύσεις και τεχνολογίες όχι μόνο σε επιχειρησιακά κομμάτια αλλά και σε καθημερινές ανάγκες που πολλές φορές δεν γίνονται αντιληπτές στους καθημερινούς χρήστες του διαδικτύου. Τέτοιου είδους χαρακτηριστικά γνωρίσματα εκφράζουν καινούργιες ιδιότητες και υπηρεσίες οι οποίες παίρνουν μέρος στη ζωή μας και δίνουν καινούργιες δυνατότητες σε αυτοματοποιημένα συστήματα.

1.3.1 Semantic Artificial Intelligence

Η σημασιολογική τεχνητή νοημοσύνη είναι μια προσέγγιση που έρχεται με τεχνικά και οργανωτικά πλεονεκτήματα. Είναι κάτι περισσότερο από έναν ακόμη αλγόριθμο μάθησης μηχανής. Πρόκειται για στρατηγική AI που βασίζεται σε τεχνικά και οργανωτικά μέτρα, τα οποία υλοποιούνται καθ' όλη τη διάρκεια του κύκλου ζωής των δεδομένων.

Μια σημασιολογική πλατφόρμα τεχνητής νοημοσύνης παρέχει τη βάση για την ανάπτυξη της τεχνητής νοημοσύνης σε επίπεδο επιχείρησης. Πρόσφατα, πολλές εταιρείες έχουν αρχίσει να μαθαίνουν πώς να ενισχύουν τις ψηφιακές υπηρεσίες, τα προϊόντα και τις διαδικασίες με τεχνητή νοημοσύνη με την κατασκευή πρωτοτύπων. Μια οργάνωση έργου με σαφώς καθορισμένο αποτέλεσμα περιορίζει τον κίνδυνο σπατάλης πόρων και υποστηρίζει μια σταδιακή καμπύλη μάθησης.

Η σημασιολογική τεχνητή νοημοσύνη συνδυάζει επιμελώς επιλεγμένες μεθόδους και εργαλεία που λύνουν τις πιο συνηθισμένες περιπτώσεις χρήσης όπως ταξινόμηση και σύσταση με πολύ ακριβή τρόπο. Ένα γράφημα σημασιολογικής γνώσης χρησιμοποιείται στην καρδιά μιας σημασιολογικής ενισχυμένης αρχιτεκτονικής AI, το οποίο παρέχει μέσα για μια πιο αυτοματοποιημένη διαχείριση της ποιότητας των δεδομένων. Αύξηση της ποιότητας των δεδομένων επιτυγχάνεται επίσης λόγω του γεγονότος ότι οι εμπειρογνώμονες του αντικειμένου χωρίς δεξιότητες στον τομέα των μαθηματικών ή εφαρμοσμένης μηχανικής λογισμικού μπορούν να κατανοήσουν τις λογικές που ακολουθούν την επεξεργασία δεδομένων και μπορούν να συνεισφέρουν με γνώσεις που αφορούν συγκεκριμένους τομείς. Ο συνδυασμός συμβατικών μεθοδολογιών AI (π.χ. Deep learning) με σημασιολογικές τεχνολογίες επιτρέπει επίσης να καθιερωθεί μια καλύτερη κατανομή εργασίας μεταξύ διαφορετικά εξειδικευμένων φορέων.

Τα σημασιολογικά εμπλουτισμένα δεδομένα χρησιμεύουν ως βάση για την καλύτερη ποιότητα δεδομένων και παρέχουν περισσότερες επιλογές για την εξαγωγή χαρακτηριστικών. Αυτό οδηγεί σε μεγαλύτερη ακρίβεια της πρόβλεψης και της ταξινόμησης που υπολογίζεται από αλγόριθμους μηχανικής μάθησης. Με τα γραφήματα γνώσης, μπορεί να χρησιμοποιηθεί ένα προηγμένο μοντέλο δεδομένων που επιτρέπει την ερμηνεία και την επαναχρησιμοποίηση δεδομένων σε διάφορα περιβάλλοντα. Τα σημασιολογικά εμπλουτισμένα δεδομένα χρησιμεύουν ως βάση για την καλύτερη ποιότητα δεδομένων και παρέχουν περισσότερες επιλογές για την εξαγωγή χαρακτηριστικών. Αυτό οδηγεί σε μεγαλύτερη ακρίβεια της πρόβλεψης και της ταξινόμησης που υπολογίζεται από αλγόριθμους μηχανικής μάθησης.

Τα συνδεδεμένα δεδομένα που βασίζονται στα πρότυπα του W3C μπορούν να χρησιμεύσουν ως πλατφόρμα δεδομένων σε επίπεδο επιχείρησης και βοηθούν στην παροχή δεδομένων κατάρτισης για την εκμάθηση μηχανών με πιο οικονομικό τρόπο. Αντί να δημιουργούν σύνολα δεδομένων ανά εφαρμογή ή περίπτωση χρήσης, δεδομένα υψηλής ποιότητας μπορούν να εξαχθούν από ένα γράφημα γνώσης ή μια σημασιολογική λίμνη δεδομένων. Μέσω αυτής της προσέγγισης που βασίζεται στα πρότυπα, μπορούν επίσης να συνδεθούν αυτόματα εσωτερικά δεδομένα και εξωτερικά δεδομένα και μπορούν να χρησιμοποιηθούν ως ένα πλούσιο σύνολο δεδομένων για κάθε εργασία εκμάθησης μηχανών.

Η σημασιολογική AI επιδιώκει να παρέχει μια υποδομή για να ξεπεράσει τις ασυμμετρίες πληροφόρησης μεταξύ των προγραμματιστών, συστημάτων AI και άλλων ενδιαφερομένων, συμπεριλαμβανομένων των καταναλωτών και των πολιτικών. Η σημασιολογική AI οδηγεί τελικά σε διακυβέρνηση AI που λειτουργεί σε τρία επίπεδα: τεχνικά, ηθικά, και στο νομικό επίπεδο.

Είναι ο συνδυασμός μεθόδων που προέρχονται από το συμβολικό AI και το στατιστικό AI. Η καλή εμφάνιση σημαίνει ότι για μια δεδομένη περίπτωση χρήσης, διάφοροι ενδιαφερόμενοι, όχι μόνο οι επιστήμονες δεδομένων, αλλά και οι ιδιοκτήτες διεργασιών ή εμπειρογνώμονες, επιλέγουν από τις διαθέσιμες μεθόδους και εργαλεία και αναπτύσσουν συνεργατικά ροές εργασίας που είναι πιθανότατα κατάλληλες για την αντιμετώπιση σε ένα βασικό πρόβλημα.

Οι περισσότεροι αλγόριθμοι εκμάθησης μηχανών λειτουργούν καλά είτε με κείμενο είτε με δομημένα δεδομένα, αλλά αυτοί οι δύο τύποι δεδομένων συνδυάζονται σπάνια για να χρησιμεύσουν ως σύνολο. Τα μοντέλα σημασιολογικών δεδομένων μπορούν να γεφυρώσουν αυτό το κενό. Οι σύνδεσμοι και οι σχέσεις μεταξύ αντικειμένων επιχειρήσεων και δεδομένων όλων των μορφών όπως XML, σχεσιακά δεδομένα, CSV και επίσης αδόμητο κείμενο μπορούν να διατεθούν για περαιτέρω ανάλυση. Αυτό μας επιτρέπει να συνδέουμε δεδομένα ακόμη και σε ετερογενείς πηγές δεδομένων για να προσφέρουμε αντικείμενα δεδομένων ως σύνολα δεδομένων εκπαίδευσης που αποτελούνται από πληροφορίες από δομημένα δεδομένα και κείμενο ταυτόχρονα.

1.3.2 Linked data catalogs

Τα data catalogs[17], markets ή repositories είναι πλατφόρμες που παρέχουν πρόσβαση σε ένα ευρύ φάσμα συνεισφερόντων συνόλων δεδομένων. Βοηθούν τους καταναλωτές δεδομένων στην εύρεση και πρόσβαση σε νέα σύνολα δεδομένων. Οι κατάλογοι γενικά προσφέρουν σχετικά μεταδεδομένα σχετικά με το σύνολο δεδομένων. Η πλατφόρμα ανοιχτού κώδικα CKAN για παράδειγμα μπορεί να χρησιμοποιηθεί για τη διαχείριση και την παροχή πρόσβασης σε μεγάλο αριθμό συνόλων δεδομένων. Το CKAN[18] συνιστάται για ένα μεγάλο ίδρυμα που θέλει να διαχειριστεί την πρόσβαση σε μια σειρά δεδομένων. Το DataHub[19] είναι ένας δημόσιος κατάλογος δεδομένων στον οποίο μπορούν να συνεισφέρουν τα σύνολα δεδομένων. Οι παραπάνω κατάλογοι δεδομένων είναι πλατφόρμες που παρέχουν πρόσβαση σε ένα ευρύ φάσμα συνόλων δεδομένων από διαφορετικούς τομείς.
Το CKAN είναι μια πλατφόρμα ανοικτού κώδικα για την ανάπτυξη ενός καταλόγου για μια σειρά συνόλων δεδομένων. Το CKAN μπορεί να χρησιμοποιηθεί από μια οργάνωση για την εσωτερική διαχείριση των συνόλων δεδομένων. Αυτά τα σύνολα δεδομένων δεν χρειάζεται να είναι διαθέσιμα στο κοινό ως μέρος του Cloud Opening Data Linking.

Το CKAN διαθέτει διάφορα εργαλεία για τους εκδότες δεδομένων που υποστηρίζουν:

- Συλλογή δεδομένων
- Δημιουργία μεταδεδομένων

- Μηχανισμοί πρόσβασης στο σύνολο δεδομένων
- Ενημέρωση του συνόλου δεδομένων
- Παρακολούθηση της πρόσβασης στο σύνολο δεδομένων

Το DataHub είναι ένας κατάλογος δεδομένων που λειτουργεί σε κοινότητα και περιέχει περισσότερα από 5.000 σύνολα δεδομένων. Το DataHub υλοποιείται χρησιμοποιώντας την πλατφόρμα CKAN και μπορεί να χρησιμοποιηθεί για την εύρεση δημόσιων συνόλων δεδομένων. Στο The DataHub, τα σύνολα δεδομένων μπορούν να οργανωθούν σε ομάδες με το καθένα να έχει τα δικά του δικαιώματα χρήστη. Οι ομάδες μπορεί να βασίζονται σε θέματα (π.χ. αρχαιολογικά σύνολα δεδομένων) ή σύνολα δεδομένων σε μια συγκεκριμένη γλώσσα ή να προέρχονται από μια συγκεκριμένη χώρα.

Τα δεδομένα είναι το καύσιμο της ψηφιακής οικονομίας και το υποκείμενο περιουσιακό στοιχείο κάθε εφαρμογής τεχνητής νοημοσύνης. Η σημασιολογική AI αντιμετωπίζει την ανάγκη για ερμηνευτικά και ουσιαστικά δεδομένα και παρέχει τεχνολογίες για τη δημιουργία τέτοιου είδους δεδομένων από την αρχή ενός κύκλου ζωής δεδομένων. Οι εταιρείες διαθέτουν και παράγουν συνεχώς δεδομένα, τα οποία διανέμονται σε διάφορα συστήματα βάσεων δεδομένων. Όταν πρόκειται για την εφαρμογή νέων περιπτώσεων χρήσης, απαιτούνται συνήθως πολύ συγκεκριμένα δεδομένα.

Υπάρχουν ερωτήσεις, εάν αυτά τα δεδομένα είναι διαθέσιμα και αν ναι, πού. Σε πολλές περιπτώσεις, πολύτιμα δεδομένα θα μπορούσαν ακόμη να εξαχθούν αυτομάτως, εάν συνδέονται διάφορες πηγές δεδομένων. Οι εφαρμογές συνήθως εξελίσσονται και απαιτούν πρόσθετα δεδομένα από κάπου αλλού. Η δημιουργία δεδομένων για μια συγκεκριμένη εφαρμογή δεν σημαίνει ότι οι ροές εργασιών δεδομένων στο σύστημα πηγής θα αντικατασταθούν. Αυτό μπορεί να οδηγήσει σε αλληλεπικάλυψη δεδομένων σε μια προέλευση σφάλματος σε έναν οργανισμό. Αυτά τα λίγα παραδείγματα ήδη εξηγούν την πολυπλοκότητα της ευέλικτης διαχείρισης δεδομένων. Δεν αποτελεί καθόλου τεχνική ευθύνη, αλλά καταδεικνύει τη σημασία ενός κεντρικού πλαισίου διακυβέρνησης δεδομένων για την ψηφιοποίηση μιας επιχείρησης, συμπεριλαμβανομένων των προϊόντων και των υπηρεσιών της.

1.3.3 Linked data and commercial search engines

Οι μηχανές αναζήτησης συλλέγουν πληροφορίες σχετικά με τους πόρους του διαδικτύου, προκειμένου να εμπλουτίσουν τον τρόπο εμφάνισης των αποτελεσμάτων αναζήτησης. Τα Snippets είναι οι λίγες γραμμές κειμένου που εμφανίζονται κάτω από έναν σύνδεσμο αποτελεσμάτων αναζήτησης για να δώσουν στον χρήστη μια καλύτερη αίσθηση του τι μπορεί να βρεθεί σε αυτήν τη σελίδα και του τρόπου με τον οποίο σχετίζεται με το ερώτημα. Τα εμπλουτισμένα αποσπάσματα (Snippet) παρέχουν πιο λεπτομερείς πληροφορίες, κατανοώντας κάτι σχετικά με το περιεχόμενο της σελίδας που εμφανίζεται στα αποτελέσματα αναζήτησης. Για παράδειγμα, ένα πλούσιο απόσπασμα ενός εστιατορίου ενδέχεται να εμφανίσει μια κριτική πελάτη ή πληροφορίες μενού. Ένα εμπλουτισμένο απόσπασμα για ένα μουσικό άλμπουμ μπορεί να παρέχει μια καταχώρηση κομματιών με έναν σύνδεσμο για κάθε μεμονωμένο κομμάτι.

Τα εμπλουτισμένα αποσπάσματα δημιουργούνται από τα δομημένα δεδομένα που ανιχνεύονται σε μια ιστοσελίδα. Τα δομημένα δεδομένα που βρίσκονται σε μια ιστοσελίδα μπορούν να εκπροσωπούνται χρησιμοποιώντας το RDFa ως μορφή σήμανσης και schema.org ως λεξιλόγιο. Το schema.org είναι μια συνεργασία μεταξύ της Google, της Microsoft και του Yahoo! να αναπτύξει ένα σχήμα σήμανσης που μπορεί να χρησιμοποιηθεί από τις μηχανές αναζήτησης για την παροχή πλουσιότερων αποτελεσμάτων.

Το schema.org παρέχει μια συλλογή σχημάτων για την περιγραφή διαφορετικών τύπων πόρων, όπως:

- Δημιουργικό έργο: Βιβλίο, ταινία, ηχογράφηση μουσικής, ...
- Ενσωματωμένα αντικείμενα εκτός κειμένου
- Εκδήλωση
- Υγεία και ιατρικά είδη
- Οργάνωση
- Τόπος, τοπική επιχείρηση, εστιατόριο
- Προϊόν, προσφορά, αθροιστική προσφορά
- Επανεξέταση, συνολική βαθμολογία

Τα δεδομένα που αντιπροσωπεύονται χρησιμοποιώντας το schema.org αναγνωρίζονται από έναν αριθμό μηχανών αναζήτησης όπως το Bing, το Google, το Yahoo! και Yandex. Το schema.org προσφέρει επίσης έναν μηχανισμό επέκτασης που ένας εκδότης μπορεί να χρησιμοποιήσει για να προσθέσει πιο εξειδικευμένες έννοιες στα λεξιλόγια. Ο σκοπός του schema.org δεν είναι να παρέχει μια οντολογία ανώτατου επιπέδου, αλλά δημιουργεί κεντρικά σχήματα κατάλληλα για πολλές κοινές καταστάσεις και που μπορεί επίσης να επεκταθεί για να περιγράψει τα πράγματα με περισσότερες λεπτομέρειες.

Το γράφημα γνώσεων Google χρησιμοποιεί δομημένα δεδομένα από την Freebase για να εμπλουτίσει τα αποτελέσματα αναζήτησης. Για παράδειγμα, μια αναζήτηση για τους Beatles θα μπορούσε να περιλαμβάνει δεδομένα σχετικά με την μπάντα και την ιδιότητα μέλους της. Στο στιγμιότυπο μπορεί να φανεί σε αυτό που ονομάζεται παράθυρο αποσαφήνισης στα δεξιά των αποτελεσμάτων αναζήτησης. Αυτές οι πρόσθετες πληροφορίες θα μπορούσαν να βοηθήσουν τον χρήστη να αποσαφηνίσει τις εναλλακτικές έννοιες των όρων αναζήτησης. Το γράφημα γνώσεων Google μπορεί επίσης να επιτρέπει στους χρήστες να έχουν πρόσβαση σε σχετικές απευθείας ιστοσελίδες που διαφορετικά θα ήταν ένα ή περισσότερα βήματα πλοήγησης μακριά από τα αποτελέσματα αναζήτησης. Για παράδειγμα, η αναζήτηση ενός άλμπουμ Beatles θα μπορούσε να παρέχει συνδέσμους που δίνουν άμεση πρόσβαση στα κομμάτια που περιέχονται στο άλμπουμ.

Τα παραπάνω παραδείγματα χρησιμοποιούν γραφήματα δεδομένων που συνδέουν τις οντότητες για να εμπλουτίσουν τα αποτελέσματα αναζήτησης. Το Open Graph Protocol[20], που αναπτύχθηκε αρχικά από το Facebook, μπορεί να χρησιμοποιηθεί για να ορίσει ένα κοινωνικό γράφημα μεταξύ ανθρώπων και μεταξύ ανθρώπων και αντικειμένων. Το Open Graph Protocol μπορεί να χρησιμοποιηθεί για να εκφράσει φιλικές σχέσεις μεταξύ ανθρώπων και σχέσεις μεταξύ ανθρώπων και αντικειμένων που τους αρέσουν: μουσική που ακούν, βιβλία που έχουν διαβάσει, ταινίες που έχουν παρακολουθήσει. Αυτοί οι δεσμοί μεταξύ ενός αντικειμένου και ενός ατόμου εκφράζονται κάνοντας κλικ σε κουμπιά Facebook "like" που μπορούν να προστεθούν από εκδότες σε ιστότοπους εκτός του τομέα Facebook. Το RDFa που είναι ενσωματωμένο στη σελίδα παρέχει μια τυπική περιγραφή του στοιχείου "like". Το Open Graph Protocol υποστηρίζει την περιγραφή πολλών τομέων, όπως μουσική, βίντεο, άρθρα, βιβλία, ιστοτόπους και προφίλ χρηστών.

Το Open Graph Protocol μπορεί να χρησιμοποιηθεί για την έκφραση διαφορετικών τύπων ενεργειών για διαφορετικούς τύπους περιεχομένου. Για παράδειγμα, ένας χρήστης μπορεί να δηλώσει ότι θέλει να παρακολουθήσει, ή να δώσει μια βαθμολογία για μια ταινία τηλεοπτικού προγράμματος. Για ένα παιχνίδι, ένας χρήστης μπορεί να καταγράψει ένα επίτευγμα ή ένα υψηλό σκορ. Αυτό το κοινωνικό γράφημα των ανθρώπων και των αντικειμένων μπορεί στη συνέχεια να χρησιμοποιηθεί στην Αναζήτηση Γραφημάτων Facebook.

2.Πρόβλημα- θέμα προς μελέτη

Τα δεδομένα στο διαδίκτυο προσφέρονται σε διάφορες δομές και σχήματα τα οποία ένας προγραμματιστής καλείται να μεταφράσει σε τεχνολογίες και τεχνοτροπίες αλληλεπίδρασης συστημάτων και πληροφοριών καθώς και τη μοντελοποίηση τους για να παρουσιάσει μια κατ' ανάγκη διαπροσωπεία υπηρεσίας. Τα προβλήματα που καλείται να λύσει πέραν της λογικής είναι:

- Τα κατάλληλα πρότυπα επικοινωνίας και οι κανόνες των frameworks που θα χρησιμοποιήσει. (Data manipulation & transformation - Build dependencies & Connectivity - Data & Processes Flow)
- Η εμφάνιση και η δόμηση της πληροφορίας σύμφωνα με τις ανάγκες της λογικής του σκοπού υλοποίησης (template - style – component context).
- Τις επιτρεπτές καταστάσεις συστήματος και του δέντρου αναπαράστασης δεδομένων και εσωτερικών καταστάσεων για την απεικόνιση της πληροφορίας. (Logic state - DOM tree[21] - Data representation)
- Μηχανισμό παρακολούθησης & ανανέωσης του συστήματος από χειρισμούς (action – event system).

Στο επίπεδο του χρήστη οι ανάγκες περιορίζονται σε κανονικοποιημένες μορφές πληροφορίας και αποτελέσματα χειρισμών. Basic UX / UI principles:

- Ότι και να συμβεί στο σύστημα ο χρήστης για να αλληλοεπιδράσει χρειάζεται μια κατάσταση.
- Αμεσότητα Αποτελεσμάτων
- Ευκολίας στο χειρισμό Ιδιοτήτων
- Οπτικά χαρακτηρισμένο περιβάλλον
- Ιδιότητες κ Αλληλεπίδραση με άλλα συστήματα χρήσης είτε από ανάγκη (Global Services and Application Sharing), είτε από επιλογή (Business Logic Extend).

Παρόλη τη γενικοποίηση των παραπάνω ζητημάτων και μιας χαώδεις ανάλυσης, βάση λογικής της εκάστοτε υλοποίησης, οι τεχνοτροπίες και τα εργαλεία μεγάλων οργανισμών και ομάδων λύνουν πολλά από αυτά τα ζητήματα. Έτσι η ανάγκη για μια υλοποίηση μιας υπηρεσίας ανάγεται σε επιλογές τεχνολογιών που εξυπηρετούν σκοπούς σε ζητήματα παροχής αναγκών και υπηρεσιών οι οποίες αλλάζουν χαρακτηριστικά στην εκάστοτε υλοποίηση αλλά παραμένουν κατά ένα σημείο, η μια λογική, κοινές σε κάποια σημεία συσχέτισης.

Στη μελέτη μας για την ανάλυση και την απεικόνιση δεδομένων από το Semantic Web γενικοποιήσαμε τη διαδικασία την οποία μπορεί να ληφθεί και να αναπαρασταθεί μια πληροφορία σε ένα User Interface που προσφέρει μια εξειδικευμένη μοντελοποίηση και τον χειρισμό της σύμφωνα με τα χαρακτηρισμό της οντότητας σε ένα End Point και πιο συγκεκριμένα στο σύνδεσμο της DBPedia. Στο Semantic Web τα δεδομένα ακολουθούν το πρότυπο της λεγόμενης τριπλέτας χαρακτηρίζοντας δεδομένα και τις σχέσεις μεταξύ τους σαν πράξεις αποτελεσμάτων.

Για παράδειγμα η εκάστοτε οντότητα έχει κάποια χαρακτηρισμένα τεχνικά χαρακτηριστικά που καλούνται properties και μεταφράζονται σαν attributes αντικειμένων. Αυτά είναι και τα χαρακτηριστικά ενός μοντέλου - αντικειμένου ή όπως μπορεί να εκφραστεί σαν μια ψηφιακή οντότητα. Στην πραγματικότητα τα properties μεταφράζονται άμεσα λογικά με ότι μπορεί να περιγράψει ένα υποκείμενο - αντικείμενο (μια οντότητα, κάτι δηλαδή χαρακτηρισμένο) και χαρακτηριστικά που περιγράφουν την διαφορετικότητα και τη μοναδικότητα μιας έκφρασης συσχέτισης. Σε επίπεδο λογικής κάθε αντικείμενο μπορεί και περιγράφεται με ιδιότητες οι οποίες μπορεί να είναι πράξεις

αποτελεσμάτων. Κάθε λειτουργία που μπορεί να περιγράφει ένα αντικείμενο αποτελεί άξιο χαρακτηριστικό συμπεριφοράς στον ψηφιακό κόσμο. Εκτός από τα δομικά χαρακτηριστικά που απαρτίζουν την κάθε οντότητα ένα σύνολο λειτουργιών μπορεί επίσης να αποτελέσει ένα χαρακτηριστικό αυτής όσον αναφορά το επίπεδο παρουσίασης σε ένα ψηφιακό περιβάλλον.

Κατά αυτόν τον τρόπο κάθε λειτουργία που μπορεί να προσαρτηθεί σε μια ψηφιακή οντότητα μπορεί να αποτελέσει δέσμιο χαρακτηριστικό όσον αναφορά την περιγραφή του σε μια γενικοποιημένη μοντελοποίηση. Με βάση τα παραπάνω μια ψηφιακή αναπαράσταση δεν κρίνεται μόνο από τα χαρακτηριστικά που μπορούν να περιγράψουν ένα αντικείμενο ή μια υπηρεσία αλλά και τις αναγωγές στον τρόπο λειτουργίας της αφού και αυτές αποτελούν μέρος της ψηφιακής περιγραφής του. Στα διασυνδεδεμένα δεδομένα κάθε συμπερασμός ή χαρακτηριστικό που μπορεί να αποτελέσει μέρος μιας λογικής αναπαράστασης μπορεί να περιγραφεί και να χαρακτηριστεί μέσω συνδέσμων και να απεικονίσει αυτές τις λειτουργίες με χαρακτηριστικά του συγκεκριμένου μοντέλου. Έτσι η μοντελοποίηση οντοτήτων σε έναν υπολογιστικό κόσμο αλλά και οντότητες που σχετίζονται με το διαδίκτυο των πραγμάτων μπορούν να σχηματίσουν στοιχεία που αποτελούν εντολές για αυτά όσον αναφορά την συμπεριφορά τους και τη διασύνδεσή τους με το τελικό χρήστη.

Μια επιχειρησιακή λογική από την άλλη μπορεί να μεταφραστεί άμεσα σε μια χαρακτηρισμένη ιδιότητα η οποία μπορεί να αποτελέσει λειτουργία σχετιζόμενη με μια οντότητα που έχει ονομαστεί και περιγραφεί. Με αυτόν τον τρόπο οι περιγραφές λογικών εκφράσεων που αποτελούν στοιχεία λειτουργίας και περιγράφουν με κανόνες μοντέλα μπορούν να αντικατοπτριστούν σε ψηφιακά χαρακτηριστικά που περιγράφουν μια μοντελοποιημένη οντότητα με μια μόνο ιδιότητα που μπορεί να κρύβει στον τρόπο λειτουργίας της έναν αυστηρά ορισμένο τρόπο αλληλεπίδρασης με τους μηχανισμούς χειρισμού της. Εξάλλου ο σκοπός δημιουργίας του σημασιολογικού ιστού είναι να περιγράψει μια βάση γνώσης χαρακτηρίζοντας τις οντότητες που μοντελοποιούνται στο διαδίκτυο με σκοπό αυτές να είναι διαχειρίσιμες όχι μόνο από χρήστες αλλά και από αυτοματοποιημένους μηχανισμούς υπολογιστικών συστημάτων. Το πρόβλημα που δημιουργείται εδώ είναι η μοντελοποίηση αυτή να είναι διαθέσιμη και διαχειρίσιμη σε κάθε είδους σύστημα που την περιγράφει και αν όχι να αυτό να είναι σε θέση να επεκτείνεται εύκολα έτσι ώστε να είναι ικανό να ανταποκρίνεται σε αυτή.

Το τελικό σημείο το οποίο κάθε οντότητα περιγράφεται στον παγκόσμιο ιστό γίνεται μέσω διαπροσωπειών και αποτυπώνει μια τελική εικόνα στους χειριστές του με λειτουργίες και επιλογές που περιγράφονται είτε από τις ίδιες τις διαπροσωπείες είτε από διασυνδεδεμένα συστήματα που τις υλοποιούν. Κάθε λογική υπηρεσία και μονάδα η οποία περιγράφεται μπορεί και αποτελεί ένα προϊόν αλληλεπίδρασης μεταξύ συστημάτων και ανθρώπων. Οι άνθρωποι ή αλλιώς τελικοί χρήστες δεν είναι σε θέση να γνωρίζουν τις υποδομές και τις λογικές που κρύβονται πίσω από τα υπολογιστικά συστήματα και πολλές φορές δεν τους ενδιαφέρει. Αυτό που αποτελεί χαρακτηριστικό προϊόν επεξεργασίας είναι χαρακτηρισμένα δομικά στοιχεία τα οποία είναι σε θέση να αλληλοεπιδράσουν και να χειριστούν όχι μόνο αυτοί αλλά και τώρα πιά και οι μηχανές. Το πρόβλημα το οποίο λοιπόν δημιουργείται είναι πως αυτές οι ψηφιακές οντότητες μπορούν να μοντελοποιηθούν με έναν επεκτάσιμο και δυναμικό τρόπο ώστε κάθε προϊόν του διαδικτύου αλλά και γενικότερα του ψηφιακού κόσμου να είναι σε θέση να γίνεται αντιληπτό τόσο από τις μηχανές όσο και από τους χρήστες στο κοινό τους σημείο αλληλεπίδρασης, σε μια δηλαδή διαπροσωπεία.

2.1 Στην πραγματική ζωή.

Μια έκφραση μπορεί να είναι ένας κανόνας, μια μεταφορά, ένας χαρακτηρισμός και πολλά άλλα πράγματα άξια αναφοράς. Το νόημα αυτής κάθε φορά είναι “μια περιγραφή χαρακτηρίζεται”, τι είναι

όμως μια περιγραφή και πως χαρακτηρίζεται. Στον αντικειμενοστραφή προγραμματισμό είναι τα Objects - Methods στο καθημερινό μας λόγο τα Subjects - Verbs, στα UIs Modules - Actions, γενικά σε όλα αυτά παρατηρείται μια αλληλεπίδραση. Και μια αλληλεπίδραση είναι μια αμοιβαία υλική ή ηθική επίδραση. Γενικευμένα μια πράξη μεταξύ δύο abstract αναφορών. Abstract ναι αλλά χαρακτηρισμένες βάση κοινών ιδιοτήτων. Οι οποίες ιδιότητες είναι και οι ορισμοί αλλά και οι άρσεις καταστάσεων και συμπεριφορών που είναι εφικτοί και ορίζονται από κανόνες. Στην πληροφορική οι οντότητες χαρακτηρίζονται γενικευμένα και δίνουν νόημα βάση ενός συνόλου υποσυνόλων λογικών ή ψηφιακών οντοτήτων, των ενεργειών και των περιορισμών τους και αυτό τις καθιστά τα λεγόμενα μοντέλα.

Μιλάμε ξανά και ξανά για ενέργειες και περιορισμούς αλλά ας υπολογίσουμε με μια υποτυπώδη μοντελοποίηση τη διαφορά του κλειστού και του ανοιχτού κόσμου. Κάτι το οποίο έχει μια ιδιότητα στον κλειστό κόσμο ανάγει σε κάτι υπαρκτό ή τουλάχιστον λεκτικά εκφρασμένο το οποίο είναι τεκμηριωμένο ή αληθές. Κάτι αντίστοιχο περιγράφεται σε μια υπολογιστική μονάδα σαν ένα σύνολο ορισμένων κανόνων και καταστάσεων μέσω μιας γλώσσας ή γραμματικής.

Οι κανόνες λοιπόν αλλά και οι περιορισμοί θα πετύχουν μια αντιστοιχία χαρακτηρισμού της εκάστοτε ψηφιακής ανάλυσης με ένα σύνολο ιδιοτήτων και καταστάσεων ώστε αυτό να γίνει αντιληπτό σαν πράξη αποτελέσματος. Παράδειγμα ένα αυτοκίνητο είναι χαρακτηρισμένο στον πραγματικό και κλειστό κόσμο σαν όχημα αλλά δεν πετάει ακόμα. Ένα αυτοκίνητο δεν πετάει και είναι όχημα και αυτό είναι αληθές στον κλειστό κόσμο και στην τωρινή πραγματικότητα και έτσι έχει εκφραστεί να λειτουργεί στις υπολογιστικές μηχανές που το περιγράφουν. Αν λοιπόν στο μέλλον το αυτοκίνητο δεχθεί τροποποίηση ώστε να πετάει έτσι και τα συστήματα που το περιγράφουν θα πρέπει να εξελιχθούν για να μπορέσουν να περιγράψουν την νέα του ιδιότητα αλλά αυτό δεν θα αλλάξει κάποια χαρακτηριστικά που ήδη το περιγράφουν είτε σαν αντιληπτή ψηφιακή συσχέτιση με μια πραγματικότητα είτε σαν γενικοποιημένη έννοια βλέπε μάρκα, μοντέλο.

Στην αναπαράσταση λοιπόν των Linked Data και της δια σύνδεσής τους, ένα αντικείμενο (subject - object) περιγράφεται βάση των πεδίων αναφοράς του (obj attributes - properties) και των σχέσεων με άλλες οντότητες αφήνοντας μια συσχέτιση μεταξύ οντοτήτων σαν χαρακτηρισμούς άλλων αναφορών πολλές φορές και στο ίδιο πεδίο συσχέτισης (attribute value as Array[] or same object attributes with different values). Αυτό περιορίζει τα προβλήματα λογικής και τα ανάγει σε σχέσεις αλληλεπίδρασης και έκφρασης. Σκεφτείτε μια υπηρεσία η οποία προσφέρεται για ένα σκοπό η οποία έχει χαρακτηριστικά άξια αναφοράς ένα υποσύνολο χαρακτηριστικών διεθνώς εκφρασμένων καθώς επίσης και ειδικώς εκφρασμένων.

Αν για παράδειγμα ένα κατάστημα διανομής χρησιμοποιήσει ένα υπολογιστικό σύστημα για την καταγραφή και παρακολούθηση παραγγελιών, την ενδιαφέρει για παράδειγμα το όνομα και η διεύθυνση της παραγγελίας τα οποία είναι διεθνώς χαρακτηρισμένα δεδομένα αλλά την ενδιαφέρει και ένας Α/Α για παράδειγμα παραγγελίας ο οποίος είναι ειδικώς εκφρασμένη πληροφορία κ αποσκοπεί για παράδειγμα στη ταύτιση με ένα σύστημα πχ νομικού ελέγχου. Επίσης ένα αντικείμενο GEO (γεωγραφικού περιεχομένου) που ενδέχεται να εμπεριέχεται σε μια τέτοιου είδους πληροφορία ή ακόμα και να σχηματίζεται σαν έξτρα πληροφορία που είναι ευρέως ή ειδικώς ορισμένο σε μια υπηρεσία πχ gmaps είναι και διεθνώς και ειδικώς εκφρασμένο αν είναι επιθυμητή κάποια αλληλεπίδραση με το μοντέλο. Η ίδια η υπηρεσία από την άλλη είναι αυτή με την σειρά της μια οντότητα στον ψηφιακό κόσμο η οποία μπορεί και χαρακτηρίζεται επίσης από τα υπολογιστικά συστήματα και αποτελεί ένα υπερμοντέλο συνδεδεμένων εννοιών που την απαρτίζουν. Για παράδειγμα μια παραγγελία μπορεί να χαρακτηριστεί ως ένα αντικείμενο που περιέχει τα προϊόντα, το

κατάστημα, τον πελάτη, των ώρα αλλά ακόμα και λειτουργίες όπως η διαδικασία της παραγγελιοληψίας και πληρωμής που αποτελούν επίσης χαρακτηριστικά της.

Με αυτό το παράδειγμα παρατηρούμε ότι εκτός από τον χαρακτηρισμό των οντοτήτων που αποτελούν τα στοιχεία μιας οντότητας, ότι και οι διαδικασίες στις οποίες μπορεί να λάβει μέρος αυτή η οντότητα αποτελούν χαρακτηριστικά της που θα πρέπει να περιγράφονται. Όπως αναφέραμε η διαδικασία της πληρωμής είναι μια λειτουργία που μπορεί να διασπαστεί σε αρκετά ενδιάμεσα στάδια όπως η online πληρωμή, η διασύνδεση με ένα τραπεζικό σύστημα που κάνει την πίστωση μέσο τραπέζης και η τελική ενημέρωση του πελάτη ότι αυτή ήλθε εις πέρας. Με βάση αυτά τα χαρακτηριστικά λοιπόν, που αποτελούν γενικότερες δομικές μονάδες που απαρτίζουν μια εκφρασμένη υπολογιστικά οντότητα, τα διασυνδεδεμένα δεδομένα είναι σε θέση να περιγράψουν στα πληροφοριακά συστήματα λειτουργίες οι οποίες μπορούν να περιγραφούν με συνδέσμους (IRIs).

Σχεδιασμός τέτοιων πληροφοριακών συστημάτων ή υπηρεσιών πρέπει κατά βάση να πληρούν τις αρχές επεκτασιμότητας και από διεθνώς ορισμένης αποτύπωσης του αντικειμένου που εξυπηρετούν και από ειδικών - εξατομικευμένων αναγκών είτε αυτές περιγράφουν δομικά χαρακτηριστικά είτε ολόκληρες λογικές που αντικατοπτρίζονται σε δομικές μονάδες του κάθε αντικειμένου. Όχι μόνο γιατί αυτές είναι οι αρχές σχεδιασμού όπως γενικευμένα έχουν οριστεί στην επιστήμη της πληροφορικής, αλλά επειδή ενδέχεται τα ίδια να είναι επεκτάσιμες και ανεξάρτητες οντότητες ως προς τις ανάγκες που υλοποιήθηκαν καθώς αυτές αλλάζουν σύμφωνα με τις εξαρτήσεις αλληλεπίδρασης μεταξύ υπηρεσιών και όχι μόνο. Αυτό το γεγονός της επεκτασιμότητας καθίσταται καίριο χαρακτηριστικό στις μέρες μας καθώς όλο και μεγαλώνουν ή μεταβάλλονται οι σχέσεις αλληλεπίδρασης μεταξύ END to END πληροφοριακών υλοποιήσεων οι οποίες είναι γνωστοποιημένες είτε από τεχνολογικά πρότυπα είτε από ευρέως χρησιμοποιημένα προκαθορισμένα μοντέλα οργανισμών και εταιρειών.

2.2 Διεπαφές Χρηστών

Το τελικό σημείο αναπαράστασης των υπηρεσιών που διατίθενται στο διαδίκτυο είναι προϊόν παραγωγής των διεπαφών. Σε αυτές υλοποιούνται και παρουσιάζονται λογικές και λειτουργίες με μεθόδους ερωτήσεων και αυτόματων χειρισμών. Οι διεπαφές χαρακτηρίζουν τη λειτουργία ενός συστήματος σε επιτρεπτές καταστάσεις στις οποίες κάθε άλλο σύστημα η χειριστής μπορεί να αλληλοεπιδράσει. Αυτές υλοποιούν προγραμματιστικές λογικές και εκδίδουν συνδέσμους και προσβάσιμα σημεία, όσων αναφορά τα APIs, και γραφικά με χειρισμούς, όσων αναφορά τα User Interfaces. Οι τρόποι επικοινωνίας και τα πρωτόκολλα που χρησιμοποιούνται σε ευρεία κλίμακα στις μέρες μας είναι οι μέθοδοι REST[22] και SOAP[23] σε ότι έχει να κάνει με αλληλεπίδραση συστημάτων και τη συγκομιδή πληροφοριών. Με αυτά επιτυγχάνεται ένας προσδιορισμός λειτουργιών και έκδοσης δεδομένων στα περισσότερα συστήματα. Οι λογικές υπηρεσίες και η διάθεση των δεδομένων σύμφωνα με μια κατάσταση παρουσιάζεται στον τελικό χρήστη μέσω μιας διαπροσωπείας η οποία αποκρύπτει αλλά και παρουσιάζει λογικές αλληλεπίδρασης με συστήματα διεπαφών που χαρακτηρίζουν τέτοιου τύπου λειτουργίες.

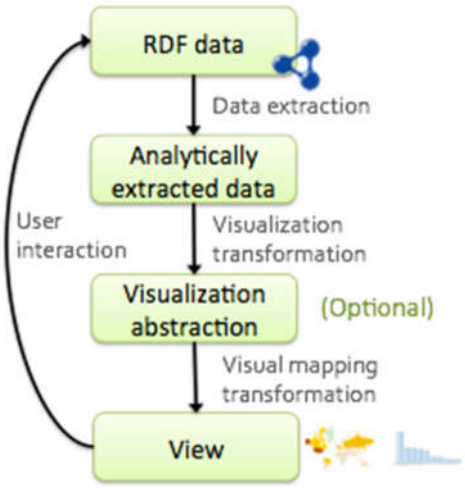
Ο συνδυασμός των semantic τεχνολογιών και των υπηρεσιών που υλοποιούνται από τις διεπαφές προσδίδει επιπλέον χαρακτηριστικά και ιδιότητες σε υπολογιστικές προσεγγίσεις μιας υπηρεσίας ή εφαρμογής. Το REST, το οποίο προτάθηκε από τον Roy Fielding στην διατριβή του[24], βρίσκεται ανάμεσα στις διασημότερες και πιο σημαντικές επιλογές, και αυτό κυρίως λόγω της απλότητας της πρότασής του έχει εμπνεύσει ένα τεράστιο αριθμό διαφορετικών εφαρμογών την τελευταία δεκαετία.

Υπάρχουν πολλοί τρόποι σύμφωνα με τους οποίους θα μπορούσαν να διατεθούν τα συνδεδεμένα δεδομένα, για παράδειγμα, μέσω ενός κατακερματισμού δεδομένων ή ενός τελικού σημείου SPARQL. Σε αρχικό κεφάλαιο παρουσιάστηκε η παροχή δεδομένων σε μορφή αναγνώσιμη από μηχανές όπως RDF / XML ή Turtle. Τα αποτελέσματα ενός ερωτήματος SPARQL θα μπορούσαν να παρασχεθούν σε μορφή όπως το JSON που θα μπορούσε τότε να χρησιμοποιήσει ένας προγραμματιστής λογισμικού για την ανάπτυξη μιας εφαρμογής. Σε επόμενο κεφάλαιο θα αναφέρουμε μορφές στις οποίες τα δεδομένα μπορούν να είναι κατανοητές σε εφαρμογές σε μηχανικά αναγνώσιμη μορφή, αλλά και πώς μπορούν να παρουσιαστούν τα συνδεδεμένα δεδομένα για χρήση από ένα ανθρώπινο κοινό.

Η εφαρμογή τεχνικών οπτικοποίησης στα δεδομένα RDF μέσο μιας διεπαφής παρέχει συναρπαστικούς τρόπους επικοινωνίας δεδομένων. Ένας χρήστης μπορεί να ξεκινήσει τη διαδικασία εισάγοντας ένα ερώτημα αναζήτησης. Η σημασιολογία των πηγών που επιστρέφονται από την αναζήτηση μπορεί στη συνέχεια να χρησιμοποιηθεί για την κατασκευή μιας συνεκτικής παρουσίασης των πληροφοριών. Οι περιγραφές κειμένων των πόρων RDF μπορούν να ενσωματωθούν με συνδέσμους προς άλλους πόρους και άλλα μέσα, όπως εικόνες. Μπορούν να συγκεντρωθούν και να παρουσιαστούν σε διαφορετικές μορφές όπως σε ένα χάρτη (δείχνοντας για παράδειγμα όπου απελευθερώθηκαν) ή και ένα διάγραμμα ράβδων (που δείχνει για παράδειγμα τον αριθμό των μεμονωμένων εκδόσεων κάθε εργασίας).

Οι απεικονίσεις έχουν ορισμένα πλεονεκτήματα ως προς τον τρόπο με τον οποίο επικοινωνούν τα δεδομένα. Καθιστούν δυνατή την περιγραφή μιας ιστορίας με τα δεδομένα, δίνοντάς της κάποια σημασία και ερμηνεία. Διευκολύνουν επίσης τους ανθρώπους να εντοπίζουν μοτίβα στα δεδομένα, όπως αλλαγές με την πάροδο του χρόνου ή μεταξύ διαφορετικών τοποθεσιών. Οι απεικονίσεις μπορούν επίσης να αποκαλύψουν διαφορές μεταξύ συνόλων δεδομένων που μπορεί να μην είναι εμφανείς από απλές περιγραφικές στατιστικές, όπως ο μέσος όρος και η παραλλαγή ενός συνόλου τιμών. Όλα αυτά είναι υλοποιήσιμα και τροποποιούνται ανάλογα με τις ανάγκες της κάθε εφαρμογής στον τελικό χρήστη μέσω μιας πλατφόρμας ή μιας κοινής διεπαφής.

Οι τεχνικές οπτικοποίησης συνδεδεμένων δεδομένων[25] στοχεύουν στην παροχή γραφικών παραστάσεων ορισμένων πληροφοριών που ενδιαφέρουν μέσα σε ένα σύνολο δεδομένων. Πρέπει να επιλεγούν απεικονίσεις που ταιριάζουν με τον τύπο δεδομένων, για παράδειγμα εάν πρόκειται για αριθμητικά δεδομένα ή πληροφορίες θέσης. Επίσης, πρέπει να επιλεγούν οπτικοποιήσεις που ταιριάζουν με την εργασία που προσπαθεί να εκτελέσει ο χρήστης, φέρνοντας στο προσκήνιο τους τύπους δεδομένων και μοτίβα στα δεδομένα με τα οποία επιθυμούν να εργαστούν. Αυτό αποτελεί και το κύριο χάσμα που επιθυμούμε να γεφυρώσουμε στην παρούσα διατριβή. Έναν αυτοματοποιημένο τρόπο αναπαράστασης προκαθορισμένων στοιχείων και λογικών υπηρεσιών που είναι διαδεδομένες στο ευρύ κοινό και είναι σε θέση να γίνει αντιληπτός από μια διεπαφή η οποία είναι ικανή να περιγράψει τις δομικές μονάδες με ένα τρόπο όχι μόνο φιλικό προς το χρήστη αλλά, ευκόλως επεκτάσιμο και από μηχανές.



Εικόνα 14: Linked Data Visualization

Το παραπάνω σχήμα απεικονίζει τον τρόπο με τον οποίο πρέπει να μετασχηματιστούν τα ακατέργαστα δεδομένα RDF για την παραγωγή οπτικοποιήσεων. Πρώτον, τα δεδομένα ενδιαφέροντος εξάγονται από το σύνολο δεδομένων (Data extraction). Η εκτέλεση ενός ερωτήματος SPARQL κάνει μια τέτοιου είδους διεργασία εφικτή. Δεύτερον, τα δεδομένα πρέπει να μετασχηματιστούν ώστε να μπορούν να εμφανιστούν με τις προβλεπόμενες μεθόδους απεικόνισης (Visualization Transformation). Ένα απλό παράδειγμα θα ήταν να εξαγάγουμε μια αριθμητική τιμή από μια συμβολοσειρά έτσι ώστε να μπορεί τότε να απεικονιστεί σε ένα διάγραμμα ράβδων. Τρίτον, τα δεδομένα πρέπει να αντιστοιχίζονται στις δομές της απεικόνισης (Visual Mapping Transformation). Για παράδειγμα, η αριθμητική τιμή θα μπορούσε να αντιστοιχιστεί στον άξονα y ενός διαγράμματος ράβδων. Η άποψη που προκύπτει από αυτή τη διαδικασία μπορεί να μην είναι μια στατική εικόνα. Μπορεί επίσης να παρέχει τρόπους με τους οποίους ο χρήστης μπορεί να αλληλοεπιδρά με τα δεδομένα, με μεγέθυνση ή κλικ για να ενεργοποιήσει περαιτέρω απεικονίσεις.

Πέραν των απλών αυτών χειρισμών διασυνδεδεμένα δεδομένα που περιγράφουν πολύπλοκες λογικές μέσω των στοιχείων τους μπορούν να χαρακτηριστούν σε δομικά στοιχεία που παρουσιάζονται στον χρήστη. Με αυτόν τον τρόπο σύνθετες λειτουργίες μπορούν να πάρουν μορφή η οποία να παρουσιάζεται με μια πιο φιλική προσέγγιση προς το χρήστη και να γίνει αν δεν είναι ευρέως διαδεδομένη στο κοινό. Για παράδειγμα η διαδικασίας login σε ένα σύστημα να περιγράφεται μέσω μιας οντότητας όπως μια επέκταση της Person της DBPedia με μια ιδιότητα External Login που χειρίζεται τους λογαριασμούς του χρήστη στους λογαριασμούς του στο διαδίκτυο κάνοντας για παράδειγμα χρήση των cookies του browser. Σύνθετες λειτουργίες λογικών έχουν τη δυνατότητα κατά βάθος να περιγραφούν από υπολογιστικά συστήματα με τη χρήση ιδιοτήτων διασυνδεδεμένων δεδομένων με συστήματα που υποστηρίζουν τεχνολογίες semantic και αναπαριστούν τις λειτουργίες τους σε συνδέσμους.

Πολλές φορές οι επεκτάσεις μοντέλων και οι περιγραφές γενικότερων χαρακτηριστικών αποτελούν κύριο μέλημα σε μια υλοποίηση. Πλαίσια που αποτελούν ήδη ολοκληρωμένες λειτουργίες ή μικρότερα χαρακτηριστικά που αποτελούν μέρος μιας ευρύτερης οντότητας κρίνονται αναγκαία στο σχηματισμό νέων χαρακτηριστικών που τα περιγράφουν ή τα χρησιμοποιούν για να υλοποιήσουν νέες ή ανασχηματισμένες ανάγκες. Τέτοιου τύπου χαρακτηριστικά προβλήματα λύνουν υλοποιήσεις οι οποίες

εκμεταλλεύονται κοινά χαρακτηριστικά και αποφεύγουν την επαναχρησιμοποίηση υπαρχόντων υλοποιημένων στοιχείων.

2.2.1 Adaptive User Interface- Adaptive Presentation

Με τον αυξανόμενο αριθμό δημοσιευμένων δομημένων δεδομένων, ο ιστός κινείται προς την κατεύθυνση της εμφάνισης ενός πλούσιου οικοσυστήματος μηχανικών κατανοητών δεδομένων. Τα σημασιολογικά δομημένα δεδομένα διευκολύνουν μια σειρά από σημαντικές πτυχές της διαχείρισης πληροφοριών, όπως η ανάκτηση πληροφοριών, η αναζήτηση, η απεικόνιση, η προσαρμογή, η εξατομίκευση και η ολοκλήρωση. Παρά τα οφέλη αυτά, οι συνδεδεμένες εφαρμογές δεδομένων (LDA)[26] δεν έχουν ακόμη υιοθετηθεί από τη μεγάλη κοινότητα προγραμματιστών ιστού εκτός του τομέα του Σημασιολογικού Ιστού και, κατ' επέκταση, από τους τελικούς χρήστες στον Ιστό. Η χρήση των σημασιολογικών δεδομένων εξακολουθεί να είναι αρκετά περιορισμένη και τα περισσότερα από τα δημοσιευμένα δεδομένα Linked Data δημιουργούνται από σχετικά μικρό αριθμό εκδοτών, γεγονός που δείχνει εμπόδια για την ευρεία χρήση των Συνδεδεμένων Δεδομένων.

Το σημερινό κενό επικοινωνίας ανάμεσα στους προγραμματιστές του Semantic Web και τους σχεδιαστές της εμπειρίας χρήστη (UX), που οφείλονται στην ανάγκη να φέρουν γνώση του Σημασιολογικού Ιστού, εμποδίζει την εκσυγχρονισμένη ροή βέλτιστων πρακτικών από την κοινότητα UX σε ανάπτυξη περιβάλλοντος διεπαφές χρήστη (UI). Η προκύπτουσα έλλειψη υιοθεσίας και τυποποίησης συχνά καθιστά τα σημερινά LDA (Linked Data Applications) ασυμβίβαστα με τις προσδοκίες των χρηστών και ωθούν περισσότερο χρόνο ανάπτυξης και κόστος για τους προγραμματιστές των LDA. Σε αυτή την περίπτωση, δαπανάται περισσότερο χρόνος για τον επανασχεδιασμό των υφιστάμενων UI παρά για την εστίαση στην καινοτομία και τη δημιουργία περίπλοκων LDA. Αυτή η εργασία παρουσιάζει την ιδέα των στοιχείων Adaptive Linked Data-driven Web Components μαζί με την εφαρμογή ανοιχτού κώδικα για την ανάπτυξη ευέλικτων και επαναχρησιμοποιήσιμων UIs του Σημασιολογικού Ιστού. Τα Web Components αποτελούν ένα σύνολο προτύπων W3C που επιτρέπουν τη δημιουργία προσαρμοσμένων, επαναχρησιμοποιήσιμων γραφικών στοιχείων ή συστατικών στοιχείων χρήστη σε έγγραφα Ιστού και σε εφαρμογές Web.

Το Resource Description Framework (RDF), από την άλλη πλευρά, παρέχει ένα κοινό μοντέλο δεδομένων που επιτρέπει τη δημιουργία, την κοινή χρήση και την ενοποίηση με δομημένο τρόπο σε διαφορετικές εφαρμογές. Συνδεδεμένα στοιχεία δεδομένων που βασίζονται σε δεδομένα είναι ένα είδος στοιχείων του Web που χρησιμοποιούν το μοντέλο δεδομένων RDF για την απεικόνιση του περιεχομένου τους και των προδιαγραφών τους (δηλαδή μεταδεδομένων σχετικά με το στοιχείο). Τα Linked Data web components υποστηρίζονται από ένα σύνολο προκαθορισμένων βασικών στοιχείων του Web, το καθένα από τα οποία αντιπροσωπεύει ένα διαμέρισμα του μοντέλου δεδομένων RDF στο Web UI. Έτσι, η φύση του Σημασιολογικού Ιστού ενός LDA μπορεί να ενθυλακωθεί σε LD components, επιτρέποντας έτσι στους σχεδιαστές UX και στους προγραμματιστές Ιστού έξω από την κοινότητα του Σημασιολογικού Ιστού να συμβάλουν σε LDAs. Τα συστατικά μέρη (components) παρέχουν επίσης στους τρέχοντες προγραμματιστές του Σημασιολογικού Ιστού έναν μηχανισμό για την επαναχρησιμοποίηση των υπαρχόντων στοιχείων του διαδικτύου στα LDA τους. Επιπλέον, εκμεταλλεύονται την ισχύ και την ευελιξία του μοντέλου δεδομένων RDF για την περιγραφή και την κοινή χρήση των πόρων ώστε να παρέχουν έναν μηχανισμό για την προσαρμογή των διεπαφών Web βάσει της σημασίας των δεδομένων και των κανόνων που ορίζονται από το χρήστη.

Με αυτή την προοπτική σχεδιασμού οι προγραμματιστές και οι σχεδιαστές διεπαφών καλούνται να σχεδιάσουν και να υλοποιήσουν τις ανάγκες των επιχειρησιακών λογικών σε ανεξάρτητες μονάδες και συστατικά μέρη με ιδιότητες οι οποίες τα χαρακτηρίζουν και τα φέρουν σε εφαρμογή ανεξάρτητα με την υπόλοιπη υλοποίηση. Κατά αυτόν τον τρόπο οι components που συντάσσουν όχι μόνο χαρακτηρίζονται από προσωπικές ιδιότητες και λειτουργίες αλλά καθίστανται και επαναχρησιμοποιήσιμοι έτσι ώστε να αποφεύγεται κάθε φορά η υλοποίηση τους. Τα θετικά χαρακτηριστικά που έρχονται να λάβουν θέση εδώ είναι όχι μόνο η εξατομίκευση μιας ορισμένης μονάδας που παρέχει ιδιότητες σε ένα περιβάλλον χρήστη αλλά και η επαναχρησιμοποίησή τους σε διαφορετικά σημεία του κώδικα. Επίσης αξιόλογη αναφορά έχει η κληρονομικότητα και η επέκτασή τους με νέες ιδιότητες.

2.2.2 Boilerplate code

Στην επιστήμη της πληροφορικής η έννοια του boilerplate έχει να κάνει με επαναχρησιμοποιήσιμα κομμάτια κώδικα τα οποία συμπεριλαμβάνονται σε πολλά σημεία του κώδικα με ελάχιστες η καθόλου αλλαγές. Πολλές φορές μια οντότητα αποτελείται από χαρακτηριστικά άξια αναφοράς και διαχείρισης από τον τελικό χρήστη η ένα άλλο κομμάτι κώδικα. Το πρόβλημα που έρχεται να λάβει χώρο σε αυτή την περίπτωση έχει να κάνει με πιθανές αλλαγές στο ενδεχόμενο μέλλον είτε κατ' ανάγκη από εξέλιξη ή αλλαγή πληροφοριακών συστημάτων, είτε από ενδεχόμενη ανάγκη παρουσίασης νέων λειτουργιών στον τελικό χρήστη σε μια πληροφοριακή λογική.

Στο κομμάτι του προγραμματισμού και στην υλοποίηση πληροφοριακών λογικών οι σχεδιαστές λογισμικού και οι προγραμματιστές διασπούν τέτοιου είδους επαναχρησιμοποιήσιμα κομμάτια σε μικρότερης εμβέλειας κομμάτια κώδικα προσδίδοντας τους ιδιότητες και λειτουργίες έτσι ώστε αυτά να έχουν την δυνατότητα χρησιμοποίησης από διαφορετικά σημεία του κώδικα και από διαφορετικά προγράμματα.

Μια ενδεχόμενη αλλαγή σε τέτοιου τύπου λειτουργίες ενδέχεται να επιφέρει επαναπρογραμματισμό και αναδιαμόρφωση του κώδικα σε σημεία του προγράμματος που επηρεάζονται είτε άμεσα, είτε έμμεσα με αποτέλεσμα οι προγραμματιστές λογισμικού να πρέπει να αλλάξουν τις εκάστοτε υλοποιήσεις τους ξεχωριστά. Ο σχεδιασμός και η διάθεση προγραμμάτων και κομματιών αυτών σε κώδικα ανοικτού λογισμικού και όχι μόνο, έχει δημιουργήσει την ανάγκη της υλοποίησης σε πιο γενετικού τύπου υλοποιήσεις έτσι ώστε να γίνεται δυνατή η υλοποίηση της χρησιμοποίησης αυτού του τύπου των κομματιών κώδικα σε διαφορετικές υλοποιήσεις και συστήματα που τα έχουν ανάγκη με μικρές παραμετροποιήσεις και αρχεία να εκφράζουν πιθανώς ειδικού τύπου αλλαγές στην λειτουργία τους.

Όσον αναφορά την υλοποίηση εφαρμογών με μεταδεδομένα και διασυνδεδεμένα δεδομένα οι ανάγκες για μοντελοποίηση και χρησιμοποίηση είναι μια βάναυση διαδικασία καθώς πολλά από τα δεδομένα που διατίθενται χρησιμοποιούν από κοινού ίδια κομμάτια κώδικα και περιγραφής σε πολλά στιγμιότυπα. Η ανάγκη του boilerplate μπορεί να ελαττωθεί σε υψηλό επίπεδο με μηχανισμούς μεταπρογραμματισμού οι οποίοι μπορούν να διαχειριστούν και να επεξεργαστούν αυτόματα από υπολογιστές και πληροφοριακά συστήματα. Επίσης τέτοιου τύπου κομμάτια κώδικα και υλοποιήσεις επαναχρησιμοποιήσεων περιχυτών και λειτουργειών μπορεί να αντιμετωπίσει τέτοιου είδους προβλήματα με τον σχεδιασμό δυναμικών υλοποιήσεων ρίχνοντας όλο το βάρος της χρησιμοποίησης και παραμετροποίησης σε αρχεία ή μηχανισμούς οι οποίοι χρησιμοποιούν μοντέλα και γεννήτριες μοντέλου-κώδικα, εξαλείφοντας την ανάγκη για χειροκίνητο κώδικα.

2.3 Μορφές Διασυνδεδεμένων Δεδομένων

Τα Linked Data είναι ένας τρόπος για να δημιουργηθεί ένα δίκτυο δεδομένων βασισμένων σε πρότυπα που μπορούν να ερμηνευτούν από τη μηχανή σε διάφορα έγγραφα και τοποθεσίες Web. Επιτρέπει σε μια εφαρμογή να ξεκινά από ένα κομμάτι Συνδεδεμένων Δεδομένων και να ακολουθεί ενσωματωμένους συνδέσμους σε άλλα τμήματα Συνδεδεμένων Δεδομένων που φιλοξενούνται σε διαφορετικές τοποθεσίες στον ιστό.

Μέσο ερωτημάτων SPARQL σε Endpoints που επιτρέπουν την διανομή πληροφοριών και Διασυνδεδεμένων δεδομένων ένας προγραμματιστής έχει την δυνατότητα να λάβει αποτελέσματα πληροφορίας σε μορφές κατανοητές και διαχειρίσιμες από ήδη υπάρχων τεχνολογίες όπως JSON, CSV & XML οι οποίες χρησιμοποιούνται και διαχειρίζονται από συστήματα που επικοινωνούν μεταξύ τους με πρωτόκολλα που τις υποστηρίζουν.

2.3.1 JSON Format

Το JSON είναι μια χρήσιμη μορφή σειριοποίησης δεδομένων και μηνυμάτων. Αυτή τη προδιαγραφή ορίζει το JSON-LD, μια μορφή που βασίζεται σε JSON για τη σειριοποίηση των Συνδεδεμένων Δεδομένων. Η σύνταξη έχει σχεδιαστεί για να ενσωματώνεται εύκολα σε αναπτυγμένα συστήματα που χρησιμοποιούν ήδη το JSON και παρέχει μια ομαλή πορεία αναβάθμισης από το JSON σε JSON-LD. Αποτελεί πρωτίστως στόχο να είναι ένας τρόπος για τη χρήση των Συνδεδεμένων Δεδομένων σε περιβάλλοντα προγραμματισμού που βασίζονται στο Web, για την ανάπτυξη διαλειτουργικών υπηρεσιών Web και για την αποθήκευση των Συνδεδεμένων Δεδομένων σε μηχανές αποθήκευσης με βάση το JSON.

Το JSON-LD είναι μια ελαφριά σύνταξη για τη σειριοποίηση των Συνδεδεμένων Δεδομένων σε JSON. Ο σχεδιασμός του επιτρέπει την ερμηνεία των υφιστάμενων JSON ως Linked Data με ελάχιστες αλλαγές. Εφόσον το JSON-LD είναι 100% συμβατό με το JSON, ο μεγάλος αριθμός των JSON επεξεργαστών και βιβλιοθηκών που είναι διαθέσιμα σήμερα μπορεί να επαναχρησιμοποιηθεί. Εκτός από όλα τα χαρακτηριστικά που παρέχει η JSON, το JSON-LD εισάγει:

- ένα μηχανισμό καθολικής αναγνώρισης για αντικείμενα JSON μέσω της χρήσης IRI,
- ένα τρόπο για την αποσαφήνιση των κλειδιών που μοιράζονται μεταξύ διαφορετικών εγγράφων JSON με τη χαρτογράφηση τους σε IRI μέσω ενός πλαισίου,
- ένα μηχανισμό στον οποίο μια τιμή σε ένα αντικείμενο JSON μπορεί να αναφέρεται σε ένα αντικείμενο JSON σε διαφορετική τοποθεσία στο Web,
- τη δυνατότητα σχολιασμού των συμβολοσειρών με τη γλώσσα τους,
- ένα τρόπο συσχετισμού των τύπων δεδομένων με τιμές όπως ημερομηνίες και ώρες,
- και μια δυνατότητα έκφρασης ενός ή περισσοτέρων κατευθυνόμενων γραφημάτων, όπως ενός κοινωνικού δικτύου, σε ένα ενιαίο έγγραφο.

Το JSON-LD έχει σχεδιαστεί για να είναι άμεσα χρησιμοποιήσιμο ως JSON, χωρίς γνώση του RDF. Είναι επίσης σχεδιασμένο για χρήση ως RDF, αν είναι επιθυμητό, για χρήση με άλλες τεχνολογίες συνδεδεμένων δεδομένων όπως το SPARQL. Οι προγραμματιστές που απαιτούν κάποια από τις διευκολύνσεις που αναφέρονται παραπάνω ή χρειάζονται σειριοποίηση ενός RDF Graph ή RDF Dataset σε μια σύνταξη βασισμένη σε JSON θα βρουν το JSON-LD ενδιαφέρον. Οι χρήστες που σκοπεύουν να

χρησιμοποιήσουν το JSON-LD με εργαλεία RDF θα διαπιστώσουν ότι μπορεί να χρησιμοποιηθεί ως άλλη σύνταξη RDF, όπως η TURTLE.

Η σύνταξη έχει σχεδιαστεί έτσι ώστε να μην ενοχλεί τα ήδη αναπτυσσόμενα συστήματα που λειτουργούν με JSON, αλλά να παρέχουν μια ομαλή πορεία αναβάθμισης από το JSON σε JSON-LD. Δεδομένου ότι το σχήμα τέτοιων δεδομένων ποικίλλει, το JSON-LD διαθέτει μηχανισμούς για την αναμόρφωση των εγγράφων σε μια δομή που προσδιορίζει τον προσδιορισμό και απλοποιεί την επεξεργασία τους.

2.3.2 CSV & TSV Formats

Οι μορφές CSV (τιμές διαχωρισμένες με κόμματα) και TSV (τιμές διαχωρισμένες σε καρτέλες) παρέχουν απλές μορφές εύκολης επεξεργασίας για τη μετάδοση των πινακοποιημένων δεδομένων. Υποστηρίζονται ως μορφές δεδομένων εισαγωγής σε πολλά εργαλεία, ιδιαίτερα σε υπολογιστικά φύλλα.

Η μορφή SPARQL αποτελεσμάτων σε CSV αποτελέσματα αποτελεί μια απώλεια κωδικοποίησης ενός πίνακα αποτελεσμάτων. Αυτή η μορφή δεν κωδικοποιεί όλες τις λεπτομέρειες για κάθε όρο RDF στα αποτελέσματα, αλλά απλώς δίνει μια συμβολοσειρά χωρίς να υποδηλώνει τον τύπο του όρου (IRI, Literal, Literal με τύπο δεδομένων, Literal με γλώσσα ή κενό κόμβο). Αυτό καθιστά απλό να καταναλώνονται δεδομένα, όπως κείμενο και αριθμούς, σε εφαρμογές χωρίς να χρειάζεται να κατανοηθούν οι λεπτομέρειες του RDF. Σε ορισμένες εφαρμογές, οι εικασίες σχετικά με τα στοιχεία που είναι υπερσύνδεσμοι γίνονται πραγματιστικά, για παράδειγμα, υποθέτοντας ότι τα strings που αρχίζουν με "http: //" είναι σύνδεσμοι.

Η μορφή SPARQL αποτελεσμάτων σε TSV αποτελεσμάτων κωδικοποιεί τις λεπτομέρειες των όρων RDF στον πίνακα αποτελεσμάτων χρησιμοποιώντας τη σύνταξη που χρησιμοποιούν οι SPARQL και η TURTLE. Μια εφαρμογή που λαμβάνει ένα σύνολο αποτελεσμάτων κωδικοποιημένων TSV μπορεί να διαιρέσει κάθε γραμμή σε στοιχεία της σειράς αποτελεσμάτων και να εξαγάγει όλες τις λεπτομέρειες που επιθυμεί να επεξεργαστεί σχετικά τους όρους RDF με απλή επεξεργασία συμβολοσειρών χωρίς έναν πλήρη αναλυτή XML ή JSON που απαιτούν οι μορφές αποτελεσμάτων SPARQL.

2.3.3 XML Format

Ένα έγγραφο Αποτελεσμάτων SPARQL είναι ένα έγγραφο XML που είναι έγκυρο σε σχέση είτε με το RELAX NG XML Schema είτε με το W3C XML Schema. Ξεκινά με στοιχείο εγγράφου SPARQL στο "*http://www.w3.org/2005/SPARQL-results# namespace*". Μέσα στο στοιχείο SPARQL υπάρχουν δύο επιμέρους στοιχεία, μια κεφαλή και ένα στοιχείο αποτελεσμάτων (είτε αποτελέσματα, είτε boolean) τα οποία πρέπει να εμφανίζονται με αυτή τη σειρά.

Το στοιχείο κεφαλής είναι το πρώτο παιδικό στοιχείο του στοιχείου *<SPARQL>*. Για ένα αποτέλεσμα ερώτησης μεταβλητής δέσμευσης, η κεφαλή (head) πρέπει να περιέχει μια ακολουθία στοιχείων που περιγράφουν το σύνολο των ονομάτων μεταβλητής ερωτήματος σε μια ακολουθία λύσεων. Η σειρά των ονομάτων μεταβλητών στην ακολουθία είναι η σειρά των ονομάτων μεταβλητών που δίδονται στο όρισμα της εντολής SELECT στο ερώτημα SPARQL. Εάν χρησιμοποιείται το στοιχείο SELECT *, η σειρά των ονομάτων είναι απροσδιόριστη. Στο εσωτερικό του στοιχείου κεφαλής, η διατεταγμένη ακολουθία των επιλεγμένων μεταβλητών ονομάτων χρησιμοποιείται για να δημιουργήσει κενά υποκείμενα στοιχεία μεταβλητά με το όνομα της μεταβλητής ως τιμή ενός ονόματος χαρακτηριστικού. Για ένα

αποτέλεσμα ερωτήματος τύπου boolean, δεν απαιτούνται στοιχεία μέσα στην κεφαλή και η μεταβλητή δεν πρέπει να υπάρχει.

Για οποιοδήποτε αποτέλεσμα ερωτήματος, η κεφαλή μπορεί επίσης να περιέχει στοιχεία συνδέσμου παιδιών με ένα χαρακτηριστικό href που περιέχει ένα σχετικό URI που παρέχει μια σύνδεση με κάποια επιπλέον μεταδεδομένα σχετικά με τα αποτελέσματα των ερωτημάτων. Το σχετικό URI επιλύεται σε σχέση με το βασικό URI βάσης, το οποίο είναι συνήθως το URI εγγράφου μορφής αποτελεσμάτων ερωτήματος. Τα στοιχεία σύνδεσης πρέπει να εμφανίζονται μετά από οποιαδήποτε μεταβλητά στοιχεία που υπάρχουν.

Το δεύτερο παιδικό στοιχείο του *<SPARQL>* πρέπει να εμφανίζεται μετά τη κεφαλή είναι είτε αποτέλεσμα είτε boolean. Είναι γραμμένο ακόμη και αν τα αποτελέσματα του ερωτήματος είναι κενά. Το στοιχείο αποτελεσμάτων περιέχει την πλήρη αλληλουχία των αποτελεσμάτων ερωτήματος. Για κάθε λύση ερωτήματος στα αποτελέσματα των ερωτημάτων, προστίθεται ένα αποτέλεσμα αποτελεσμάτων παιδιού.

Κάθε στοιχείο αποτελέσματος αντιστοιχεί σε μια λύση ερωτήματος σε ένα αποτέλεσμα και περιέχει στοιχεία παιδιού (σε καμία συγκεκριμένη σειρά) για κάθε μεταβλητή ερωτήματος που εμφανίζεται στη λύση. Χρησιμοποιείται για την καταγραφή του τρόπου με τον οποίο οι μεταβλητές επερωτήσεων συνδέονται με τους όρους RDF. Κάθε δέσμευση μέσα σε μια λύση γράφεται ως στοιχείο δεσμευτικό ως παιδί του αποτελέσματος με το όνομα μεταβλητής ερωτήματος ως τιμή του χαρακτηριστικού ονόματος. Η τιμή μιας δεσμευτικής μεταβλητής ερωτήματος, η οποία είναι ένας όρος RDF, συμπεριλαμβάνεται ως το περιεχόμενο της δέσμευσης ως εξής:

- **RDF URI Reference *U***
  <binding><uri>*U*</uri></binding>
- **RDF Literal *S***
  <binding><literal>*S*</literal></binding>
- **RDF Literal *S* with language *L***
  <binding><literal xml:lang="*L*">*S*</literal></binding>
- **RDF Typed Literal *S* with datatype URI *D***
  <binding><literal datatype="*D*">*S*</literal></binding>
- **Blank Node label *I***
  <binding><bnode>*I*</bnode></binding>

Ένα παράδειγμα λύσης ερωτήματος που κωδικοποιείται σε αυτή τη μορφή έχει ως εξής:

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">

  <head>
    <variable name="x"/>
    <variable name="hpage"/>
    <variable name="name"/>
    <variable name="age"/>
    <variable name="mbox"/>
    <variable name="friend"/>
  </head>

  <results>

    <result>
      <binding name="x">
        <bnode>r2</bnode>
      </binding>
      <binding name="hpage">
        <uri>http://work.example.org/bob/</uri>
      </binding>
      <binding name="name">
        <literal xml:lang="en">Bob</literal>
      </binding>
      <binding name="age">
        <literal datatype="http://www.w3.org/2001/XMLSchema#integer">30</literal>
      </binding>
      <binding name="mbox">
        <uri>mailto:bob@work.example.org</uri>
      </binding>
    </result>

    ...
  </results>

</sparql>
```

Εικόνα 15: Linked Data XML Format

2.4 Παρουσίαση προβλήματος

Σύμφωνα με όλα όσα αναφέραμε στην παρούσα εργασία θα προσπαθήσουμε να υλοποιήσουμε ένα σύστημα λειτουργίας ανεξάρτητων διαλειτουργικών components που έρχονται σε συνεργασία μεταξύ τους και εκμεταλλεύονται τις λειτουργίες και την πληροφορία των διασυνδεδεμένων δεδομένων. Σκοπός είναι να υλοποιηθούν ανεξάρτητοι λειτουργικοί components που χαρακτηρίζουν μορφές πληροφορίας ως ανάγκες ενός τελικού χρήστη, να δρουν ανεξάρτητα και να επικοινωνούν μεταξύ τους για να προσφέρουν προαπαιτούμενες λειτουργίες. Στόχος μας είναι να εκφράσουμε έναν ανεξάρτητο τρόπο διαχείρισης των διασυνδεδεμένων δεδομένων ώστε η διαπροσωπεία που θα παρουσιάσουμε να είναι ικανή να δέχεται στοιχεία από End-points και hypermedia endpoints αναγνωρίζοντας το είδος της πληροφορίας και να μπορεί να τα ανάγει σε δομικές μονάδες που εκφράζουν τα χαρακτηριστικά τους στον χρήστη. Επιπλέον να υπάρξει μια διαχείριση των μεταδεδομένων και των στοιχείων που συντελούν πληροφορία για τα διασυνδεδεμένα δεδομένα έτσι ώστε οι web crawlers και αυτοματοποιημένα συστήματα στο διαδίκτυο να διαχειρίζονται την πληροφορία χωρίς να απαιτείται από τον προγραμματιστή ιδιαίτερη επεξεργασία σε ενδεχόμενες αλλαγές και προσθήκες στοιχείων που αποτελούν την πληροφορία. Το χαρακτηριστικό που ευελπιστούμε να προσδώσουμε στην εφαρμογή είναι να το σύστημα να μπορεί μέσω IRIs να περιγράψει όχι μόνο χαρακτηριστικά των μοντέλων που επιθυμεί να αναπαραστήσει αλλά και να χαρακτηρίσει λειτουργίες που περιγράφονται με αυτά ορίζοντας ένα σύστημα γράφου που απεικονίζει προγραμματιστικές λογικές. Συγκεκριμένα με τον ορισμό ψευδογράφου από αρχείο οι νέες ιδιότητες που θέλουμε να προσδώσουμε στην εφαρμογή είναι υπηρεσίες που απεικονίζονται μέσω IRIs και χαρακτηρίζονται σαν ιδιότητες στα μεταδεδομένα. Με τις υπηρεσίες να είναι αυτές που κατά βάθος χαρακτηρίζουν μια εφαρμογή σκοπό έχουμε να περιγράψουμε τη λογική σύνθετων υπηρεσιών σε IRIs τα οποία με τον δυναμικό resolver που θα κατασκευαστεί αυτές να έχουν όχι μόνο τη δυνατότητα να διαχειρίζονται από την εφαρμογή αλλά και να αντανακλώνται σε ένα σύνδεσμο που θα εκφράζει λειτουργίες και θα εξυπηρετεί προγραμματιστικές λογικές. Στόχος είναι να καταφέρουμε να απεικονίσουμε σύνθετες λογικές με τη

μορφή συνδέσμων και να δώσουμε τη δυνατότητα αυτοί να απεικονίζονται σε μορφές τριπλέτας ώστε να καθίσταται δυνατή η περιγραφή και η λειτουργία τους από διασυνδεδεμένα δεδομένα.

Για το λόγο αυτό η επεξεργασία και η παρουσίαση των δεδομένων που επεξεργάζεται κάθε interface σε μια τέτοιου είδους πληροφορία θα πρέπει να πληροί την ανάγκη της αυτοματοποίησης ήδη υπάρχων λειτουργιών όπως η προσθήκη, η αναδημοσίεση και η δημιουργία νέων στοιχείων όπως αυτές καθορίστηκαν από τον κύκλο ζωής τους παραπάνω. Αυτές οι λειτουργίες θα πρέπει όχι μόνο να αυτοματοποιηθούν από τις διεπαφές αλλά και να ικανοποιούν τις αρχές ώστε ένας χρήστης χωρίς να αντιλαμβάνεται την λειτουργία και τις αρχές μιας υλοποίησης να μπορεί με ενέργειες που πραγματοποιεί από επιλογές του να δομεί στοιχεία πληροφορίας που γίνονται αντιληπτά από αυτοματοποιημένα συστήματα. Η προσωπική διαχείριση και ο τρόπος με τον οποίο ο κάθε χρήστης αντιλαμβάνεται και ικανοποιεί τις ανάγκες του για πληροφόρηση αποτελεί ξεχωριστό προϊόν για συστήματα γνώσης και επεξεργασίας. Κατά αυτόν τον τρόπο και σεβόμενοι τη διαφορετικότητα κάθε σύστημα καθίσταται ικανό να διευρύνει τον τρόπο με τον οποίο αναγνωρίζει και αναζητά τα δεδομένα. Ένα υπολογιστικό σύστημα που υλοποιείται κατά αυτόν τον τρόπο θα πρέπει να είναι ικανό να επεκτείνει τον τρόπο λειτουργίας του από μόνο του χωρίς ιδιαίτερες αλλαγές. Έτσι η ανάγκη για αναδιορισμό χαρακτηριστικών και επαναπρογραμματισμό μεγαλώνει με τους σχεδιαστές λογισμικού να αναδιαμορφώνουν και να εξελίσσουν μια υλοποίηση ολοένα και περισσότερο. Στόχος στην εποχή μας είναι τα συστήματα να έχουν την ικανότητα να πράττουν πολλές από αυτές τις ανάγκες από μόνα τους και να αναδιαμορφώνουν πολλές φορές κομμάτια κώδικα αυτοματοποιημένα εκφράζοντάς τα με έναν δυναμικό τρόπο.

Το πρόβλημα που καλούμαστε να ανταποκριθούμε είναι η κατασκευή ενός framework και μιας διαδικασίας αποτίμησης δεδομένων με έναν δυναμικό τρόπο ώστε η λειτουργία του κατά τη σύνδεση σε κάθε τελικό σημείο να ανταποκρίνεται χωρίς να απαιτούνται αλλαγές και παραμετροποιήσεις κάθε φορά που αλλάζει μια λογική οντότητα. Η διασύνδεση με εξωτερικά συστήματα κάθε φορά ποικίλει ως προς τα δεδομένα τα οποία προσφέρονται και με βάση τους πόρους τους οποίους παρέχονται από κάθε API. Έτσι η ανάγκη αναπροσδιορισμού και επέκταση του κώδικα κάθε φορά κρίνεται αναγκαία από προγραμματιστές και σχεδιαστές. Στις μέρες μας υπάρχουν συστήματα πληροφορικής τα οποία ανταποκρίνονται σε κάθε είδους πληροφορία όταν ερωτηθούν κατά βάση με ένα resource που σχηματίζεται. Οι τεχνολογίες των REST calls προσφέρουν μια πληθώρα από Interfaces τα οποία απαντούν και προσφέρουν σε κάθε ερώτημα διαφορετική δομή αντικειμένων αλλά με προκαθορισμένα και χαρακτηρισμένα πρότυπα όπως το JSON και το XML. Στόχος είναι να καταφέρουμε να εκφράσουμε την κάθε συλλογή με ένα δυναμικού τύπου reasoning το οποίο συμπεριφέρεται με ένα κοινό τρόπο κυρίως σε διασυνδεδεμένα δεδομένα και ύστερα σε αντικείμενα που μπορεί να αναγνωρίσει. Τα διασυνδεδεμένα δεδομένα έχουν το χαρακτηριστικό να περιγράφονται σαν οντότητες οι οποίες έχουν την δυνατότητα να ορίζονται με rdf διαγράμματα. Για να επιτύχουμε λοιπόν τη διαδικασία αυτόματης αναγνώρισης σε μια generic διεπαφή αρκεί να εκφράσουμε αυτόν τον χαρακτηρισμό με κοινές και χαρακτηρισμένες ιδιότητες και να υλοποιήσουμε μια προκαθορισμένη λειτουργία για «άγνωστα» χαρακτηριστικά τα οποία να μπορούν να επεξεργαστούν χειριστές πέραν των προγραμματιστών με ειδίκευση στο σημασιολογικό ιστό.

Βασικό χαρακτηριστικό μιας τέτοιας υλοποίησης είναι ο χειρισμός των μεταδεδομένων και των γράφων που παρέχουν γνώση και πληροφορία. Η δημοσίευση αυτών σε triplestores και ο ορισμός νέων οντοτήτων από μονάδες που πραγματοποιούν τη συγκεκριμένη λειτουργία με ένα αυτοματοποιημένο τρόπο είναι τα κύρια χαρακτηριστικά που θα μας απασχολήσουν και αποτελούν μια δομή αντίληψης και χειρισμού. Μην ξεχνάμε ότι όλη η δομή της πληροφορίας και ο τρόπος που γίνεται αντιληπτός από τις μηχανές δεν είναι μόνο η αποθήκευση μιας πληροφορίας και η διάθεση αλλά και ο ορισμός της ως

οντότητα ή ο συνδυασμός της από ορισμούς μικρότερων οντοτήτων. Το πρόβλημα λοιπόν που καλείται η τεχνολογία να λύσει είναι να προσφέρει τέτοιες ιδιότητες στους χρήστες του διαδικτύου έτσι ώστε εκμεταλλευόμενη τους χειρισμούς του κάθε χειριστή να μπορεί και να εκμεταλλεύεται τις πληροφορίες δημιουργώντας γνώση. Με αυτόν τον τρόπο η διαφορετικότητα αποτύπωσης της κάθε ανθρώπινης λογικής και ανάγκης θα εκφράζεται σαν μια δόμηση της πληροφορίας προς άλλους χρήστες και θα αποτελεί μια καινούργια ή επέκταση υπάρχουσας βάση για τις μηχανές.

Σε κάθε σχεδιασμό ενός πληροφοριακού συστήματος κατά την ανάλυση και ανάπτυξη των λειτουργιών σχεδιάζονται οι ανάγκες και παρουσιάζονται μέσω μιας διεπαφής στον τελικό χρήστη. Μια ολοκληρωμένη λύση που εξυπηρετεί τις ανάγκες του και εμπλουτίζεται με λειτουργίες και γραφικά αποκρύπτει την υποδομή και χρησιμοποίηση της τεχνογνωσίας που έχει χρησιμοποιηθεί και δίνει όχι μόνο μια προσιτή προσέγγιση αλλά πολλές φορές και μια ικανοποίηση κατά τον χειρισμό της σε ανάγκες που διαφέρουν στον κάθε χειριστή ξεχωριστά.

Στο παρόν κεφάλαιο παρουσιάσαμε από μια γενικοποιημένη σκοπιά τα προβλήματα που καλείται να αντιμετωπίσει ένας προγραμματιστής ή σχεδιαστής συστημάτων διασυνδεδεμένων δεδομένων για να περιγράψει και να αξιοποιήσει τις δυνατότητες και τις πληροφορίες που διανέμονται ελεύθερα στο διαδίκτυο. Μιλήσαμε για διεπαφές χρηστών και τις κοινές τους επαναλαμβανόμενες λειτουργίες και πως αυτές με τον καιρό υλοποιούνται ξανά και ξανά για να παρέχουν νέες υπηρεσίες και εφαρμογές στις διεπαφές χρηστών και άλλων υπολογιστικών συστημάτων. Παρουσιάσαμε επίσης τους τρόπους και κοινούς φορμαλισμούς με τους οποίους τα διασυνδεδεμένα δεδομένα διανέμονται και διατίθενται από εξωτερικά συστήματα.

Στο επόμενο κεφάλαιο παρουσιάζεται από τεχνολογικής σκοπιάς το αντικείμενο και πως τα διασυνδεδεμένα δεδομένα έρχονται σε συνεργασία με τις υπάρχων υλοποιήσεις να φέρουν ένα στρώμα παρουσίασης στον τελικό χρήστη. Πριν προχωρήσουμε σε μια τέτοιου είδους ανάλυση και τεχνική επεξήγηση αρχιτεκτονικών και σχεδιασμού αξίζει να περιγράψουμε πως σχεδιάζεται και υλοποιείται μια υπηρεσία από τους μηχανικούς λογισμικού σχετικά με το πρόβλημα που καλείται να λύσει και την υπηρεσία που ενδέχεται να προσφέρει στους χρήστες – χειριστές της. Κάθε ανάλυση ξεκινάει από γενικές ανάγκες οι οποίες περνούν από επιπλέον στάδια διαχωρισμού και διάσπασης σε επιμέρους στοιχεία.

2.4.1 Πρώτη ανάλυση και καταγραφή

Για την υλοποίηση και καταγραφή των απαιτήσεων και των ιδιοτήτων διασυνδεδεμένων δεδομένων σκοπός της παρούσας εργασίας είναι η υλοποίηση μιας ηλεκτρονικής βιβλιοθήκης με σκοπό να παρουσιάσει στον χρήστη αναγκαίες πληροφορίες και λειτουργίες οι οποίες αντλούνται από SPARQL endpoints που προσφέρουν σε RDF διαγράμματα την μορφή της πληροφορίας. Απώτερος σκοπός είναι να καταγραφούν τα προβλήματα στην μετάδοση και τη διάθεση τέτοιου είδους πληροφορίας, να παρουσιαστεί μια προσέγγιση ανεξάρτητων user interface components με πληροφορίες που υλοποιούν και διανέμουν στον τελικό χρήστη λειτουργίες και υπηρεσίες. Αυτοί πρέπει να είναι ικανοί να λειτουργούν αυτόνομα, να επικοινωνούν και να αλληλοεπιδρούν μεταξύ τους για να στελεχώσουν μια διεπαφή και να δώσουν την ικανότητα στον χρήστη να επεξεργάζεται και να εμπλουτίζει τα δεδομένα με τον δικό του τρόπο. Τέλος να παρουσιαστεί μια υπολογιστική προσέγγιση η οποία στελεχώνεται από ανεξάρτητα υπολογιστικά κομμάτια τα οποία πλαισιώνουν την εφαρμογή μιας διαπροσωπείας και αλληλοεπιδρώντας με το διαδίκτυο να παρέχουν πληροφορία στο χρήστη από παρόμοια δεδομένα. Επίσης σκοπός μας είναι να μπορούμε να περιγράψουμε γενικότερες υπηρεσίες που αποτελούν ανάγκες του χρήστη με τις σχετικές οντότητες που περιγράφονται μέσω συνδέσμων σε συστήματα που

τις υλοποιούν. Αυτά θα πρέπει να είναι σε θέση να αναγνωρίζουν το είδος της πληροφορίας που εισέρχεται στο σύστημα και να καθορίζουν τον τρόπο λειτουργίας τους με στόχο να υλοποιούν οντότητες διεθνώς αναγνωρισμένες και μοντελοποιημένες από εξωτερικά υπολογιστικά συστήματα και συνδέσμους που παρέχουν δομημένη και εκφρασμένη πληροφορία. Σκοπός της κατασκευής είναι αυτά να είναι αυτολειτούργητα, αυτοματοποιημένα και κατά προσέγγιση δυναμικά ώστε να χρησιμοποιούνται καθώς λειτουργούν από τους προγραμματιστές ανεξάρτητα και επαναλαμβανόμενα με μικρές τροποποιήσεις και παραμετροποιήσεις.

Σαν πρώτο στάδιο καταγραφής των απαιτήσεων και των αναγκών χώρο έρχεται να λάβει μια τεχνογνωσία της τεχνολογίας λογισμικού που καταγράφει απλοποιημένα τις ανάγκες του τελικού χρήστη και τις λειτουργίες που ενδέχεται να κριθούν αναγκαίες από τη μεριά του. Φυσικά και αναφερόμαστε στα γνωστά case diagrams της UML[27] που με απλή καταγραφή των αναγκών σχηματίζουν μια πρώτη εικόνα καταγραφής των απαραίτητων λειτουργιών που ενδέχεται να υλοποιηθούν.

Στο παράδειγμά μας για την ηλεκτρονική βιβλιοθήκη σκοπός είναι να καταγράψουμε τις βασικές ανάγκες ενός χειριστή καθώς και τις ιδιότητες των αντικειμένων που κρίνονται απαραίτητες να χρησιμοποιηθούν. Η καταγραφή αυτή των αναγκών γίνεται με σκοπό αυτές να μπορούν να συνδυαστούν ώστε να φανούν χρήσιμες και να παρουσιάσουν μια ολοκληρωμένη εικόνα. Γίνεται επίσης μια ομαδοποίηση και αναφορά σε γενικότερες έννοιες και ανάγονται από μεριάς αναγκών τελικού χρήστη σε ιδιότητες και αναφορές του σχεδιαστή.

2.4.2 Καταγραφή απλοποιημένων αναγκών

Μια πρώτη ανάλυση μιας ηλεκτρονικής βιβλιοθήκης όσον αναφορά τις ανάγκες του τελικού χρήστη σε ένα use case diagram έχει να κάνει με απλοϊκά ζητήματα και όσο το δυνατόν πιο απλά εκφρασμένα ζητούμενα. Σκοπός ενός τέτοιου διαγράμματος είναι να καταγράψει σε επίπεδο αναγκών προτάσεις και ζητήματα τα οποία ενδέχεται να παρουσιάσει η εφαρμογή και να μοντελοποιήσει τις ανάγκες σε επίπεδα παρουσίασης μιας διεπαφής.

Σε μια ηλεκτρονική βιβλιοθήκη ο χρήστης έχει την ανάγκη καθ’ αυτή να βρίσκει πληροφορίες άμεσα σχετικά με συγγράμματα και άρθρα που τον απασχολούν, να δέχεται λεπτομέρειες σχετικά με αυτά και να συγκεντρώνει μια γενικότερη μορφή πληροφόρησης που εκπληρώνουν τις ανάγκες του για τα συγγράμματα και τα κείμενα που ενδέχεται να αναζητήσει και να ενημερωθεί χρησιμοποιώντας την πλατφόρμα. Ανάγκες που μπορεί να παρουσιαστούν σε μια πλατφόρμα ηλεκτρονικής βιβλιοθήκης δεν αφορούν μόνο κείμενα και το περιεχόμενό τους αλλά και άλλες οντότητες που περιγράφονται από τα διασυνδεδεμένα δεδομένα όπως αυτές των συγγραφέων, τοπογραφικές πληροφορίες σχετικά με την ανεύρεση συγγραμμάτων, events που σχετίζονται με κοινωνικά δίκτυα και σελίδες, πληροφορίες σχετικά με εκδοτικούς οίκους και άλλα. Επίσης λογικές υπηρεσίες που μπορεί να σχετίζονται με τις συγκεκριμένες οντότητες όπως η ενοικίαση κάποιου βιβλίου, η παραγγελία από συνδέσμους που τα προσφέρουν και άλλες όπως η δημοσίευση ή αναζήτηση του συνδέσμου που παρακολουθείται σε κάποιο μέσο κοινωνικής δικτύωσης.

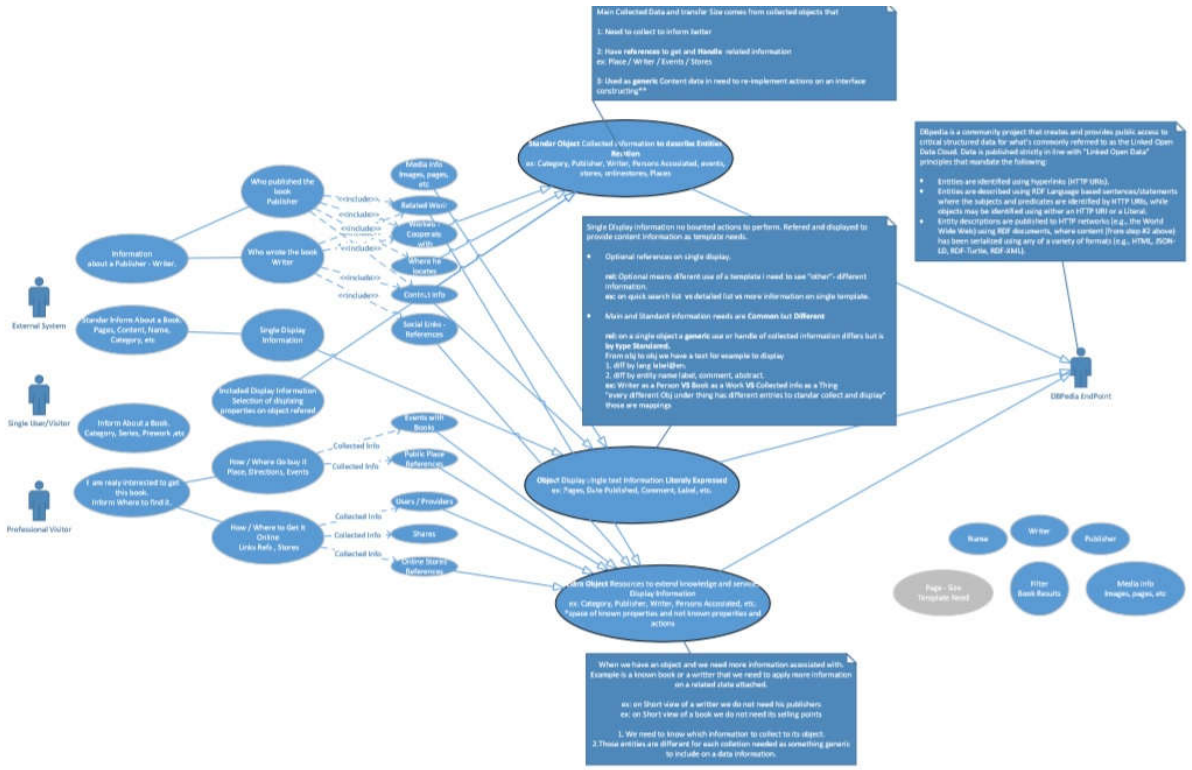
Για να υπάρξει μια καταγραφή των βασικών πληροφοριών των οντοτήτων που ενδέχεται να παρουσιαστούν και να χρησιμοποιηθούν πρέπει να λάβουμε υπ’ όψη μας όχι μόνο τον τελικό χρήστη που ενδέχεται να χρησιμοποιήσει την πλατφόρμα αλλά και άλλα endpoints που ενδέχεται να χρησιμοποιήσουν τους πόρους και τη συλλογή πληροφοριών. Κύρια μορφή αυτοματοποίησης για τα δεδομένα που προσφέρονται σε μια τέτοια υπηρεσία είναι τα μεταδεδομένα. Στον κόσμο των

διασυνδεδεμένων δεδομένων όπως παρουσιάσαμε στα προηγούμενα κεφάλαια τα μεταδεδομένα είναι ένα σημαντικό κομμάτι για τον κύκλο ζωής των διασυνδεδεμένων δεδομένων καθώς και για την χρησιμοποίηση τους από μηχανισμούς αυτοματοποίησης και πράκτορες που υλοποιούνται από μηχανές και υπολογιστές. Σαν κύρια εισαγωγή αναγκών στο σύστημά μας αφήσαμε να εννοηθεί ο τελικός χρήστης και οι ανάγκες του αλλά στην πραγματικότητα η λειτουργικότητα που εκφράζει τα διασυνδεδεμένα δεδομένα είναι στις ιδιότητες των αντικειμένων τους και στον τρόπο συσχέτισής τους με άλλες οντότητες που εκφράζονται από συνδέσεις και hypperlinks.

Σαν μια πρώτη καταγραφή των αναγκών της πλατφόρμας μας μπορούμε να παρουσιάσουμε ιδέες και ερωτήματα του χρήστη σε μια γενική μορφή η οποία με τη διαδικασία της ανάλυσης θα προχωρήσει σε λεπτομερέστερες ανάγκες και οντοτικοποίηση. Στο πρώτο στάδιο του διαγράμματος έχουμε ανάγκες όπως :

- Οι πληροφορίες σχετικά με των συγγραφέα ή τον εκδότη
- Βασικές πληροφορίες των συγγραμμάτων
- Πληροφορίες σχετικά με τα βιβλία, τις σειρές, τις κατηγορίες
- Την ανάγκη για πληροφόρηση σχετικά με τη διάθεση του βιβλίου

Οι παραπάνω ανάγκες αναλύονται σε πιο εξιδανικευμένες όπως ο διαχωρισμός της πληροφόρησης σχετικά με τις οντότητες του συγγραφέα και του εκδότη. Βασικές πληροφορίες που παρουσιάζονται σχετικά με το σύγγραμμα. Πληροφορίες σχετικά με το που μπορεί κάποιος να αγοράσει το βιβλίο όπως τοποθεσίες, οδηγίες πρόσβασης και events. Πως μπορεί κάποιος να το βρει σε διαδικτυακά μαγαζιά και σελίδες με αναφορές, ακόμα και ηλεκτρονικά καταστήματα.



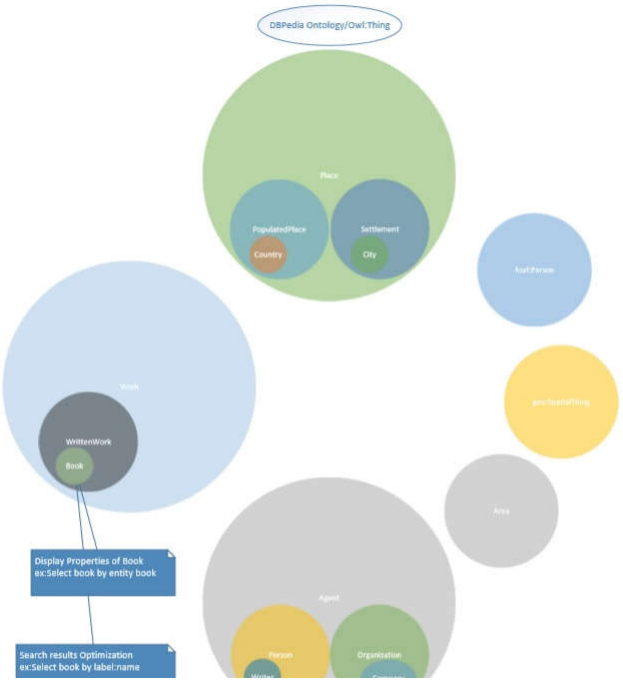
Διάγραμμα 1: Book Example Use case

Η παραπάνω ανάλυση επεκτείνεται σε ακόμα πιο λεπτομερή κομμάτια σχετικά με τις οντότητες των εκδοτών και των συγγραφέων. Σχετικές πληροφορίες που ενδέχεται να απασχολήσουν τον χρήστη είναι αυτές των media, άλλες όπως σχετικές εικόνες, σχετικοί σύνδεσμοι, σχετική βιβλιογραφία του συγγραφέα, πληροφορίες καταγωγής και τοποθέτησης, επικοινωνιακά στοιχεία, σχετικοί σύνδεσμοι με social media, αναφορές σε ιστοσελίδες καθώς και λοιπές πλατφόρμες πληροφόρησης. Ανάγκες ακόμα σχετικές με την διάθεση και τη συσχέτιση των αναφερόμενων οντοτήτων σε καταστήματα, δημόσια κτήρια, βιβλιοθήκες και άλλα προσωποκεντρικά και τοπογραφικά δεδομένα που σχετίζονται με τη διάθεση όπως καταστήματα και εκδοτικοί οίκοι. Οι πληροφορίες όπως αναφέραμε με μέσα κοινωνικής δικτύωσης και online διάθεσης συγγραμμάτων καθώς και σχετικές πληροφορίες χρηστών που τις διαθέτουν, όπως επίσης και διαδικτυακοί σύνδεσμοι που σχετίζονται με τέτοιου είδους πληροφόρηση χρήζουν επίσης αναγκαίες.

Με μια υποτυπώδη ανάλυση τέτοιου είδους αναγκών και λειτουργιών που ενδέχεται να χρησιμοποιήσει μια εφαρμογή παρατηρούμε μια γενικοποίηση όσων αναφορά τη μεριά του προγραμματιστή- σχεδιαστή και τα διασυνδεδεμένα δεδομένα που διατίθενται. Με τον τρόπο με τον οποίο είναι κατασκευασμένοι οι σύνδεσμοι στις τριπλέτες οι τιμές τους έχουν το πλεονέκτημα να περιγράφουν με λεκτικά κάποιες ιδιότητες και άλλες με συνδέσμους. Σύμφωνα με την παραπάνω ανάλυση ορατή γίνεται σε βασικές ανάγκες περιγραφής η επιλογή τέτοιων ιδιοτήτων τόσο με απλά λεκτικά όσο και με συνδέσμους που περιγράφουν πιο ολοκληρωμένες οντότητες και αντικείμενα. Η διάθεση αυτών σε γλώσσα SPARQL έχει να κάνει με την επιλογή φίλτρων και εμφωλευμένων ιδιοτήτων που διατίθενται από σχέσεις συνδέσμους. Έτσι η παραπάνω ανάλυση έχει να κάνει με τρεις τύπους γενικευμένων αναγκών. Σε μια εφαρμογή δηλαδή μπορούμε να περιγράψουμε την παραπάνω ανάλυση με ιδιότητες λεκτικών, βασικών αντικειμένων εκφρασμένων σε IRIs και άλλων λοιπών αντικειμένων που σχηματίζονται για την ανάγκη της εφαρμογής από συλλογή πληροφοριών από συστήματα και βάσεις δεδομένων.

Για την κατασκευή ενός UI της ηλεκτρονικής μας βιβλιοθήκης, βασικές οντότητες της DBpedia που περιγράφονται από την owl:Thing είναι αναγκαίο να προσαρτηθούν για να περιγράψουν τα αντικείμενά μας όπως αναλύσαμε παραπάνω. Αυτές πολλές φορές χαρακτηρίζουν υποκλάσεις πραγμάτων και άλλες γενικότερες έννοιες που χρειάζεται να χρησιμοποιηθούν με ερωτήματα SPARQL ώστε να στελεχωθεί η πληροφορία. Για την συγκεκριμένη εργασία επιλέξαμε να περιγράψουμε οντότητες που προσδίδουν χαρακτηριστικά στις ανάγκες που καταγράφηκαν παραπάνω. Αυτές όπως φαίνεται και στα παρακάτω διαγράμματα Venn όντας υποκλάσεις της γενικότερης οντότητας owl:Thing είναι:

1. Work
    a. Written Work
        i. Book
2. Agent
    a. Person
        i. Writer
    b. Organization
        i. Company



Διάγραμμα 2: Entities Venn Diagram

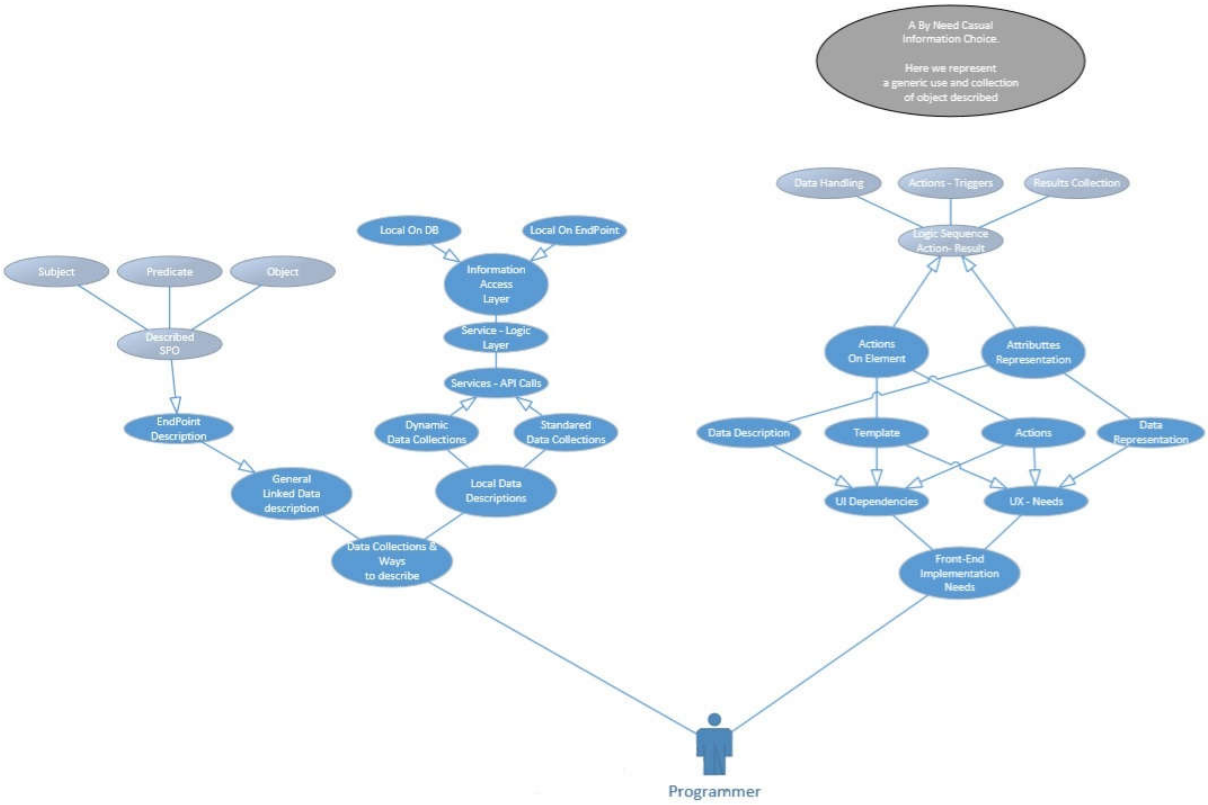
3. Foaf:person
4. Geo:SpatialThing
5. Area
6. Place
    a. Populate Place
        i. Country
    b. Settlement
        i. City

2.4.3 Προγραμματιστική σκοπιά

Σχετικά με την ανάλυση που προηγήθηκε, μια πλατφόρμα ηλεκτρονικής βιβλιοθήκης από προγραμματιστική σκοπιά θα πρέπει να είναι ικανή να χρησιμοποιήσει τις πληροφορίες που εκδίδονται από τα διασυνδεδεμένα δεδομένα καθώς και άλλες πηγές που παρουσιάζονται από API συστημάτων που είναι ήδη υλοποιημένα. Όπως για παράδειγμα APIs από πλατφόρμες κοινωνικής δικτύωσης, συστήματα τα οποία παρέχουν πληροφορίες σχετικά με άλλες ηλεκτρονικές βιβλιοθήκες, ακόμα και ένα back end σύστημα που υλοποιεί προγραμματιστικές ανάγκες που επεκτείνουν την παρουσίαση δεδομένων ή ακόμα υλοποιούν μια αγορά ή ένα σύστημα profiling για τους χρήστες της πλατφόρμας.

Οι ανάγκες σε μια τέτοια υπηρεσία ποικίλουν και ανάλογα με το επίπεδο των πληροφοριών και των λειτουργιών που ενδέχεται να χρησιμοποιηθούν, δημιουργούν ανάγκες όπως επικοινωνία με άλλα συστήματα, εξατομίκευσης λειτουργίας για τους χρήστες της και όχι μόνο. Από τη μεριά του προγραμματιστή παρουσιάζεται το πρόβλημα της σύνθεσης των πληροφοριών και της κατασκευής λειτουργιών σε μια διεπαφή η οποία θα είναι λειτουργική και φιλική προς τον τελικό χρήστη. Μια σχετική υλοποίηση περιορίζει της ανάγκες παρουσίασης σε ένα UI το οποίο δεν χρειάζεται να υλοποιεί κάποιου είδους business λογική αλλά να δέχεται και να παρουσιάζει αφού συνδυάσει αντικείμενα.

Κατά την κατασκευή ενός front end το οποίο εκμεταλλεύεται τους πόρους των ανοικτών δεδομένων ο προγραμματιστής καλείται να μετατρέψει αυτές τις ανάγκες σε μια συλλογή δεδομένων και να τα εκφράσει σε αντικείμενα προσδίδοντάς τους ιδιότητες μέσω συναρτήσεων οι οποίες και υλοποιούν τις λειτουργίες της πλατφόρμας. Στο παρακάτω σχεδιάγραμμα παρουσιάζεται από τη μεριά του προγραμματιστή ένα διάγραμμα που εκφράζει τις ανάγκες ενός τέτοιου συστήματος, της σύνθεσης των δεδομένων καθώς και τις ανάγκες παρουσίασης σε μια διεπαφή.



Διάγραμμα 3: Programmer Analysis Use Case

Το πρόβλημα που καλείται να λύσει ένας προγραμματιστής και παρουσιάζουμε σε αυτή την εργασία είναι ένας τρόπος διαχείρισης των διασυνδεδεμένων δεδομένων από τελικά σημεία SPARQL, η πρόσμιξή τους με πληροφορίες από άλλα συστήματα και η τελική τους διάθεση. Για την διαχείριση, τη διάθεση και την παρουσίαση της πληροφορίας των Linked Data απαιτείται όχι μόνο η λύση προβλημάτων όπως το parsing των πληροφοριών και η μετάφρασή τους σε δεδομένα που μπορεί να χρησιμοποιήσει κάποιος σε μια διεπαφή αλλά και ένας γενικότερος τρόπος διαχείρισής τους. Να αναφέρουμε ότι ο όγκος των διαφορετικών δεδομένων και του πλήθους διαφορετικών ιδιοτήτων ποικίλει αλλά υπάρχει ένα κοινό σημείο αναφοράς. Αυτό όπως αναφέραμε και παραπάνω είναι ότι τα δεδομένα αυτά διατίθενται εκφρασμένα σε literals και IRIs. Εκμεταλλευόμενοι αυτή την ομοιογένεια θα παρουσιάσουμε αργότερα έναν τρόπο έτσι ώστε αυτού του τύπου τα δεδομένα να μεταμορφώνονται σε javascript objects με έναν γενετικό τρόπο και να καθιστούν τη χρησιμοποίησή τους θέμα configuration και υλοποίησης templates που εξυπηρετούν σκοπούς προβολής και χειρισμών.

3. Αντικείμενο υπό διερεύνηση

Κάθε εφαρμογή όχι μόνο έχει διαφορετικό σκοπό αλλά και διαφορετικές αρχιτεκτονικές αποφάσεις όσον αφορά τον τρόπο πρόσβασης, επεξεργασίας και αποθήκευσης των δεδομένων. Σε αυτή την ενότητα παρουσιάζουμε τα γενικά αρχιτεκτονικά πρότυπα των εφαρμογών των συνδεδεμένων δεδομένων και τις διαφορετικές αποφάσεις σχεδιασμού που σχετίζονται με αυτά τα πρότυπα.

Ο όρος αρχιτεκτονική λογισμικού περιγράφει τα στοιχεία ενός συστήματος λογισμικού και τις σχέσεις μεταξύ αυτών των στοιχείων. Για ένα web based σύστημα τα στοιχεία θα μπορούσαν να περιλαμβάνουν λειτουργικές μονάδες λογισμικού (modules), βάσεις δεδομένων και διακομιστές web. Ορισμένα τμήματα της αρχιτεκτονικής μπορεί να είναι στοιχεία παλαιού τύπου προερχόμενα από προηγούμενα συστήματα. Οι σχέσεις μεταξύ των στοιχείων ενός συστήματος υποδεικνύουν ποια στοιχεία επικοινωνούν κατά τη λειτουργία του συστήματος και τους μηχανισμούς με τους οποίους πραγματοποιείται αυτή η επικοινωνία. Παράλληλα με τον προσδιορισμό της δομής του συστήματος, η αρχιτεκτονική λογισμικού ορίζει επίσης ένα σύνολο σχεδιαστικών πρακτικών που πρέπει να ακολουθηθούν για να δημιουργηθεί και να διατηρηθεί η αρχιτεκτονική.

Ένα σημαντικό αρχιτεκτονικό μοτίβο που χρησιμοποιείται στην ανάπτυξη του συστήματος είναι η πολυεπίπεδη αρχιτεκτονική (multi-tier architecture)[28]. Μια πολυεπίπεδη αρχιτεκτονική χωρίζει τη λειτουργικότητα σε μια σειρά από στρώματα από χαμηλού επιπέδου αποθήκευσης δεδομένων μέχρι τα συστατικά στοιχεία αλληλεπίδρασης χρήστη. Κάθε στρώμα είναι υπεύθυνο για ξεχωριστές ενέργειες της εφαρμογής και η διασύνδεσή τους αποτελεί την κύρια λύση. Με αυτή την εφαρμογή τα διαφόρου επιπέδου στρώματα επικοινωνούν μεταξύ τους αλλά υλοποιούνται ξεχωριστά. Αυτό τα κάνει εύκολα επεκτάσιμα και ξεχωριστά διαχειρίσιμα. Αυτή η αρχιτεκτονική χρησιμοποιείται συνήθως για πολλά είδη εφαρμογών στον ιστό. Δεδομένου ότι πολλές εφαρμογές συνδέσμων δεδομένων είναι επίσης εφαρμογές ιστού, τείνουν να συμμορφώνονται με αυτήν την αρχιτεκτονική προσέγγιση.

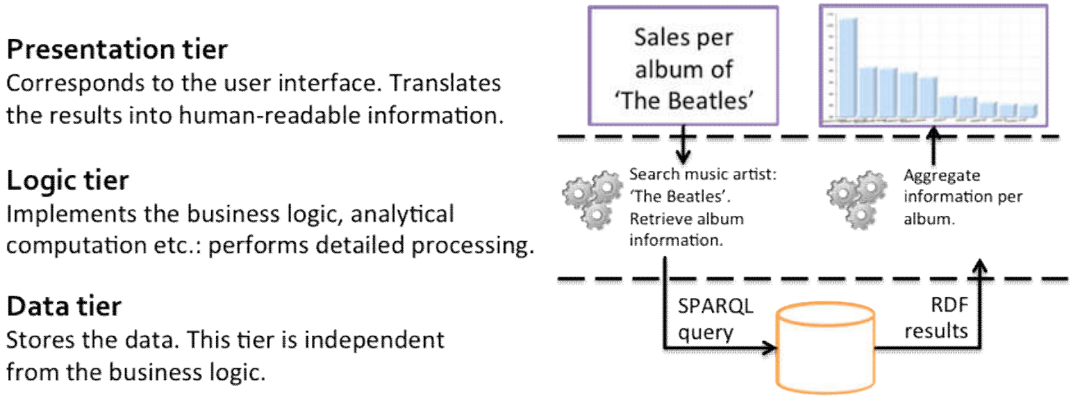
Ένα σημαντικό πλεονέκτημα της κλιμακωτής αρχιτεκτονικής είναι ότι διαχωρίζει λογικά τη λειτουργικότητα του συστήματος σε μια σειρά από στρώματα και καθορίζει την επικοινωνία μεταξύ αυτών των στρωμάτων. Αυτός ο διαχωρισμός καθιστά πολύ πιο εύκολη την αντικατάσταση ενός στρώματος της αρχιτεκτονικής ή την επαναχρησιμοποίηση ενός στρώματος της υπάρχουσας σε μια νέα εφαρμογή. Για παράδειγμα, μια εφαρμογή μπορεί να έχει ένα επίπεδο που να είναι αφιερωμένο στην αποθήκευση δεδομένων. Αυτή η λειτουργικότητα μπορεί να παρέχεται από μια συγκεκριμένη βάση δεδομένων ή ένα triplestore. Το στρώμα αποθήκευσης μπορεί να επαναχρησιμοποιηθεί σε μια εναλλακτική εφαρμογή, να αντικατασταθεί ή να επεκταθεί από ένα άλλο στρώμα αποθήκευσης που παρέχει την ίδια λειτουργικότητα και επικοινωνία με άλλα επίπεδα.

3.1 Αρχιτεκτονική Σημασιολογικών υλοποιήσεων

Η πιο συχνή αρχιτεκτονική πολλών επιπέδων είναι η αρχιτεκτονική τριών επιπέδων. Κατ' αρχάς, ένα επίπεδο παρουσίασης το οποίο παρέχει μια διεπαφή που μπορεί να δεχθεί την είσοδο του χρήστη και να καταστήσει τα αποτελέσματα σε μορφή αναγνώσιμη από άνθρωπο. Δεύτερον, μια λογική βαθμίδα που υλοποιεί την επιχειρησιακή λογική της εφαρμογής. Σε αυτό το επίπεδο η υλοποίηση λαμβάνει τα διαθέσιμα δεδομένα και αναλύσεις και τα μετατρέπει για να καλύψει τις ανάγκες του χρήστη. Τρίτον, μια βαθμίδα δεδομένων η οποία αποθηκεύει τα υποκείμενα δεδομένα σε μια μορφή ανεξάρτητη από την επιχειρησιακή λογική που εφαρμόζεται σε αυτήν στην εφαρμογή.

Στην παρακάτω εικόνα φαίνεται ένα παράδειγμα μουσικής βιβλιοθήκης ως αρχιτεκτονική τριών επιπέδων. Το επίπεδο παρουσίασης χειρίζεται τα ερωτήματα και τις αλληλεπιδράσεις του χρήστη και τις επιστρεφόμενες εξόδους, συμπεριλαμβανομένων των αποτελεσμάτων κειμένου και των οπτικοποιήσεων. Το λογικό επίπεδο μετατρέπει τα ερωτήματα χρήστη σε ερωτήματα SPARQL και συγκεντρώνει τα αποτελέσματα RDF. Το επίπεδο δεδομένων είναι υπεύθυνο για την αποθήκευση και στην περίπτωση αυτή χρησιμοποιείται ένα triplestore.

Μια σημαντική πτυχή που πρέπει να σημειωθεί στην περίπτωση εφαρμογών συνδεδεμένων δεδομένων είναι ότι η διαχωριστική γραμμή μεταξύ της βαθμίδας δεδομένων και της λογικής βαθμίδας ενδέχεται να μην είναι τόσο ξεκάθαρη. Εάν μια σχεσιακή βάση δεδομένων χρησιμοποιείται ως στρώμα αποθήκευσης, τότε όλη η επεξεργασία των δεδομένων πέρα από την επιστροφή των αποτελεσμάτων σε ένα ερώτημα βάσης δεδομένων γίνεται στο λογικό επίπεδο. Εντούτοις, τα triplestores είναι ικανά να εκτελούν διάφορους τύπους reasoning και επομένως σε ορισμένες περιπτώσεις ένα σημαντικό μέρος της επιχειρησιακής λογικής μπορεί να μεταφερθεί μέσα στο triplestore. Για λόγους απόδοσης, είναι επωφελές να γίνεται το reasoning σε χαμηλότερο δυνατό επίπεδο.

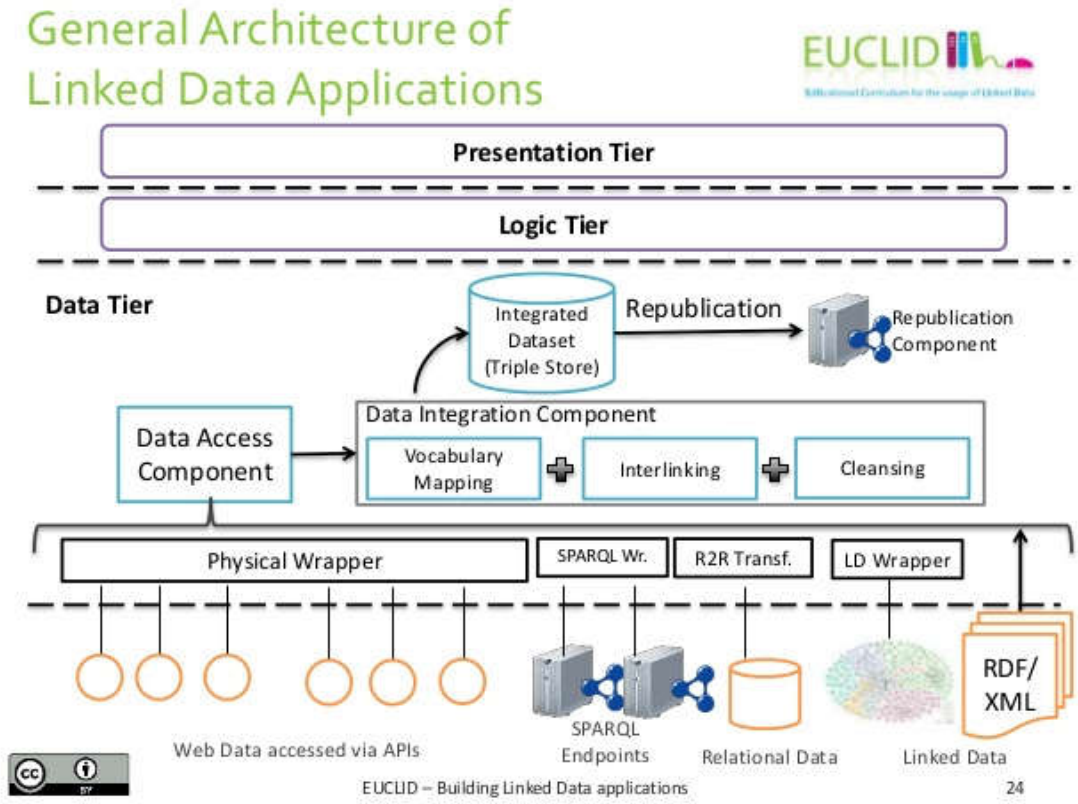


Εικόνα 16: Semantics 3-tier Logic

Το στρώμα παρουσίασης παράγει τα είδη απεικονίσεων που είδαμε στο Κεφάλαιο 2. Το λογικό στρώμα επεξεργάζεται δεδομένα για παρουσίαση. Το στρώμα δεδομένων καθώς εφαρμόζει και μερικώς από τη λογική κάνει πολύ περισσότερη δουλειά παρά απλή αποθήκευση δεδομένων. Τα δεδομένα που αποθηκεύονται στο στρώμα δεδομένων μπορούν να καταναλωθούν από διάφορες πηγές. Αυτά μπορεί να καταναλώνονται από τα SPARQL endpoints ή RDF dumps. Οι Wrappers μερικές φορές απαιτείται να μετατρέψουν τα δεδομένα σε κατάλληλη μορφή. Όπως περιγράφεται παρακάτω, το R2RML[29] μπορεί να χρησιμοποιηθεί για τη μετατροπή δεδομένων από σχεσιακή βάση δεδομένων σε RDF.

Επίσης, όπως αναφέραμε στο Κεφάλαιο 2, τα ανακτημένα δεδομένα μπορεί να χρησιμοποιούν διαφορετικά λεξιλόγια στο επίπεδο σχήματος και μπορεί επίσης σε επίπεδο παρουσίασης να έχουν διαφορετικές αναφορές σε στοιχεία που αναφέρονται στο ίδιο πράγμα. Οι γλώσσες όπως το SKOS μπορούν να χρησιμοποιηθούν για την έκφραση σχέσεων μεταξύ των λεξιλογίων και η ιδιότητα του **owl: sameAs** μπορεί να χρησιμοποιηθεί για τη συσχέτιση πόρων. Επίσης θα μπορούσαν να χρησιμοποιηθούν εργαλεία όπως το πλαίσιο SILK για τον εντοπισμό και την έκφραση σχέσεων μεταξύ των συνόλων δεδομένων.

Μπορεί επίσης να απαιτείται κάποιος καθαρισμός δεδομένων, για παράδειγμα, για τον εντοπισμό αμφισημιών μεταξύ των ονομάτων των πόρων μέσα στα σύνολα δεδομένων και για την επίλυση αυτών των ασαφειών. Επομένως, πρέπει να πραγματοποιηθεί χαρτογράφηση λεξιλογίου, διασύνδεσης και καθαρισμός για την παραγωγή ενός συνεκτικού συνόλου δεδομένων. Εκτός από τη χρήση του λογικού επιπέδου, μπορεί να παρέχεται άμεση δυνατότητα αναδημοσίευσης των δεδομένων.



Εικόνα 17: Semantics Application Tier

3.2 Αρχιτεκτονικά Patterns

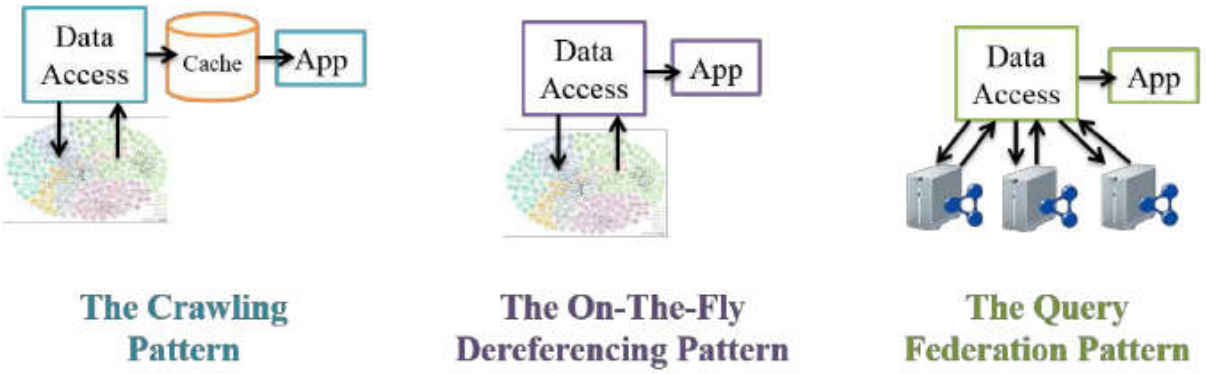
Τα δεδομένα που καταναλώνονται και ενσωματώνονται σε μια εφαρμογή μπορούν να προσεγγιστούν με διάφορους τρόπους. Μπορούν να εντοπιστούν τρία κύρια αρχιτεκτονικά πρότυπα.

Πρώτον, υπάρχει το **crawling pattern** (μοτίβο ανίχνευσης), στο οποίο τα δεδομένα φορτώνονται εκ των προτέρων. Τα δεδομένα μπορεί επίσης να χρειάζεται να μετασχηματιστούν όπως περιγράφεται παραπάνω. Τα δεδομένα διαχειρίζονται σε ένα triplestore, έτσι ώστε να μπορούν να είναι προσβάσιμα αποτελεσματικά. Το μειονέκτημα αυτού του προτύπου είναι ότι τα δεδομένα ενδέχεται να μην είναι ενημερωμένα.

Δεύτερον, υπάρχει το μοτίβο **On-The-Fly Dereferencing Pattern** (απελευθερωμένο). Εδώ, οι διευθύνσεις URI αποκλείονται τη στιγμή που η εφαρμογή απαιτεί τα δεδομένα. Αυτό το πρότυπο

ανακτά τα ενημερωμένα δεδομένα, αλλά η απόδοση επηρεάζεται όταν η εφαρμογή πρέπει να απελευθερώσει πολλά URIs.

Τρίτον, υπάρχει το **Federated Query Pattern** (Ομοσπονδιακό Πρότυπο ερωτημάτων) στο οποίο υποβάλλονται σύνθετα ερωτήματα σε ένα σταθερό σύνολο πηγών δεδομένων. Αυτή η προσέγγιση επιτρέπει στις εφαρμογές να λειτουργούν με τα τρέχοντα δεδομένα που ανακτώνται απευθείας από τις πηγές. Ωστόσο, η εξεύρεση βέλτιστων σχεδίων εκτέλεσης ερωτημάτων σε μεγάλο αριθμό πηγών είναι ένα πολύπλοκο πρόβλημα. Αυτό το τρίτο μοτίβο σε συγκεκριμένες καταστάσεις μπορεί να προσφέρει έναν τρόπο πρόσβασης σε ενημερωμένα δεδομένα με επαρκείς χρόνους απόκρισης.



Εικόνα 18: Linked Data Patterns

3.3 Στρώμα Δεδομένων

Στο επίπεδο δεδομένων τα νέα Συνδεδεμένα Δεδομένα ενδέχεται να καταναλωθούν από ένα SPARQL endpoint σε RDF. Όπως αναφέρθηκε παραπάνω, εάν τα δεδομένα βρίσκονται σε άλλη μορφή όπως CSV, τότε ο wrapper θα χρησιμοποιηθεί για τη μετατροπή των δεδομένων σε RDF. Οι εφαρμογές Συνδεδεμένων Δεδομένων μπορούν να εφαρμόσουν μια αρχιτεκτονική Mediator-Wrapper για να αποκτήσουν πρόσβαση σε ετερογενείς πηγές, στις οποίες οι wrappers είναι χτισμένοι γύρω από κάθε πηγή δεδομένων, ώστε να παρέχουν μια ενιαία προβολή των ανακτηθέντων δεδομένων.

Η Αρχιτεκτονική Mediator-Wrapper θα μπορούσε να χρησιμοποιηθεί με οποιοδήποτε από τα τρία αρχιτεκτονικά πρότυπα (crawling pattern, on-the-fly dereferencing pattern, federated query pattern).
Τα πιο κατάλληλα πρότυπα θα εξαρτηθούν από διάφορους παράγοντες όπως ο αριθμός των πηγών που πρέπει να έχουν πρόσβαση, ο τρόπος με τον οποίο πρέπει να είναι ενημερωμένα τα δεδομένα και η ταχύτητα απόκρισης που απαιτείται από το επίπεδο δεδομένων. Υπάρχουν διαθέσιμα διάφορα εργαλεία που μπορούν να χρησιμοποιηθούν κατά την υλοποίηση ενός στοιχείου πρόσβασης δεδομένων.

Οι Linked Data Crawlers είναι προγράμματα ανίχνευσης ιστού που έχουν σχεδιαστεί για τη συλλογή δεδομένων RDF. Οι βιβλιοθήκες συνδεδεμένων δεδομένων υποστηρίζουν την πρόσβαση και τη μετάβαση των Συνδεδεμένων Δεδομένων. Οι βιβλιοθήκες SPARQL παρέχουν ένα API για πρόσβαση στα SPARQL endpoints. Οι ομοσπονδιακοί μηχανισμοί SPARQL παρέχουν ένα ενιαίο σημείο πρόσβασης για πολλές ετερογενείς πηγές δεδομένων. Τέλος, τα API μηχανών αναζήτησης όπως το Sindice υποστηρίζουν σημασιολογικές αναζητήσεις που επιστρέφουν έγγραφα RDF.

Mechanisms	Tools (Examples)
Linked Data Crawlers	LDspider https://code.google.com/p/ldspider/ Slug https://code.google.com/p/slug-semweb-crawler/
Linked Data Client Libraries	Semantic Web Client Library http://wifo5-03.informatik.uni-mannheim.de/bizer/ng4j/semwebclient/ The Tabulator http://www.w3.org/2005/ajar/tab Moriarty https://code.google.com/p/moriarty/
SPARQL Client Libraries	Jena Semantic Web Framework http://jena.apache.org/
Federated SPARQL Engines	ANAPSID https://github.com/anapsid/anapsid FedX http://www.fluidops.com/fedx/ SPLENDID https://code.google.com/p/rdffederator/
Search Engine APIs	Sindice http://sindice.com/developers/api Uberblic http://uberblic.com/

Εικόνα 19: Semantics Mechanisms & Tools

Το ολοκληρωμένο σύνολο δεδομένων μπορεί στη συνέχεια να διατηρηθεί σε ένα τοπικό triplestore. Απαιτείται ένα triplestore εκτός εάν τα δεδομένα ανακτηθούν επί τόπου και απορρίπτονται αμέσως μετά την δημιουργία μιας απάντησης στο αίτημα του χρήστη. Ορισμένα εμπορικά ή ελεύθερα RDF triplestores είναι διαθέσιμα, συμπεριλαμβανομένων των OWLIM[30], Jena TDB[31], Cumulus[32], Allegro Graph[33], Virtuoso Universal Server[34] και RDF3x[35]. Όπως περιγράφεται, τα τελικά σημεία SPARQL και τα RDF dumps μπορούν να χρησιμοποιηθούν για τη διάθεση δεδομένων RDF. Τα δεδομένα μπορούν επίσης να διατίθενται μέσω API ή χρησιμοποιώντας τη λειτουργικότητα που παρέχεται από το επιλεγμένο application framework.

3.3.1 Vocabulary Mapping, Interlinking, Cleansing

Εκτός από τη χρήση των υφιστάμενων λεξιλογίων, ο συντάκτης ή ο συντηρητής ενός συνόλου δεδομένων θα πρέπει να διερευνήσει πώς οι οντότητες του συνόλου δεδομένων μπορούν να συνδεθούν με οντότητες σε άλλα σύνολα δεδομένων. Αυτό ακολουθεί την αρχή των Συνδεδεμένων Δεδομένων διασύνδεσης με άλλους κόμβους URI, ώστε ο χρήστης να μπορεί να ανακαλύψει περισσότερα πράγματα. Οι συνδέσεις RDF μεταξύ οντοτήτων σε διαφορετικά σύνολα δεδομένων μπορούν να καθοριστούν σε δύο επίπεδα: το επίπεδο παρουσίας και το επίπεδο σχήματος. Στις συνδέσεις επιπέδου ενότητας μπορούν να γίνουν σύνδεσμοι μεταξύ μεμονωμένων οντοτήτων (π.χ. ανθρώπων, τόπων, αντικειμένων) χρησιμοποιώντας τις ιδιότητες **rdfs:seeAlso** και **owl:sameAs**. Η ιδιότητα **owl:sameAs** χρησιμοποιείται για να δηλώσει ότι δύο αναφορές URI αναφέρονται πραγματικά στο ίδιο πράγμα. Η ιδιότητα ***rdfs:seeAlso*** συν δεικνύει ότι περισσότερες σχετικές πληροφορίες μπορούν να βρεθούν ακολουθώντας τον σύνδεσμο.

Στο επίπεδο σχήματος, το οποίο περιέχει το λεξιλόγιο που χρησιμοποιείται για την ταξινόμηση των αντικειμένων σε επίπεδο παρουσίας, μπορεί να εκφραστεί ένας αριθμός σχέσεων χρησιμοποιώντας το RDFS, OWL και το SKOS Mapping vocabulary. Οι ιδιότητες RDFS **rdfs:subPropertyOf** και **rdfs:subClassOf** μπορούν να χρησιμοποιηθούν για να δηλώσουν σχέσεις μεταξύ δύο ιδιοτήτων ή δύο κλάσεων από διαφορετικά λεξιλόγια.

Το OWL παρέχει επίσης ευρήματα για να δηλώσει ότι δύο κατηγορίες ή δύο ιδιότητες έχουν το ίδιο νόημα, για παράδειγμα:

mo:MusicArtist - owl:equivalentClass - ex:musician .
foaf:made - owl:equivalentProperty - ex:creatorOf.

Το πρώτο από αυτές τις τριπλέτες σημαίνει ότι όλες οι περιπτώσεις μιας από αυτές τις τάξεις είναι επίσης περιπτώσεις του άλλου. Το δεύτερο σημαίνει ότι εάν δύο πόροι συνδέονται με μία από τις ιδιότητες τότε συνδέονται επίσης από την άλλη ιδιότητα. Οι ιδιότητες χαρτογράφησης SKOS μπορούν επίσης να χρησιμοποιηθούν για να εκφράσουν την ευθυγράμμιση μεταξύ εννοιών από διαφορετικά λεξιλόγια.

Η διαδικασία ανίχνευσης δεσμών μεταξύ συνόλων δεδομένων είναι γνωστή ως ανακάλυψη συνδέσμων. Τα σύνολα δεδομένων είναι ετερογενή όσον αφορά τα λεξιλόγια, τη μορφή και την αναπαράστασή τους. Αυτό καθιστά τη διαδικασία της ανακάλυψης συνδέσεων πολύ πιο πολύπλοκη. Ο καθορισμός του αν δύο οντότητες από διαφορετικά σύνολα δεδομένων αναφέρονται στο ίδιο πράγμα είναι ένα παράδειγμα αυτού που είναι γνωστό ως πρόβλημα επίλυσης οντότητας. Δύο τύποι ασάφειας μπορούν να κάνουν τη διαδικασία πιο δύσκολη. Οι διφορούμενες ονομασίες μπορεί να προκύψουν από τα τυπογραφικά λάθη ή τη χρήση διαφορετικών γλωσσών ή ομώνυμων για να περιγράψουμε κάτι. Οι διαρθρωτικές αμφισημίες προκύπτουν από οντότητες που έχουν πιθανώς ασυνεπείς σχέσεις με άλλες οντότητες στα αντίστοιχα σύνολα δεδομένων τους. Αυτά επιλύονται χρησιμοποιώντας τεχνικές οντολογίας και αντιστοίχισης σχήματος.

Οι αντιστοιχίσεις μεταξύ των συνόλων δεδομένων (είτε σε επίπεδο στιγμιότυπου ή σχήματος) μπορούν να ανακαλυφθούν και να εκφραστούν τόσο με το χέρι όσο και αυτόματα. Η χειροκίνητη σύγκριση ζευγών οντοτήτων από διαφορετικά σύνολα δεδομένων δεν είναι πρακτική για μεγαλύτερα σύνολα δεδομένων. Το SILK είναι ένα εργαλείο που μπορεί να χρησιμοποιηθεί για να ανακαλύψει και να εκφράσει σχέσεις μεταξύ συνόλων δεδομένων.

Το SKOS *(Simple Knowledge Organization System)* είναι ένα μοντέλο δεδομένων για την έκφραση και τη σύνδεση των Συστημάτων Οργάνωσης της Γνώσης, όπως τα συστήματα θησαυρών, ταξινομικών και ταξινομήσεων. Το SKOS εκφράζεται ως RDF τριπλέτες.

3.3.2 R2R – SKOS

Για τα δεδομένα που είναι αποθηκευμένα σε πολλούς πίνακες σε μια σχεσιακή βάση δεδομένων χρειαζόμαστε έναν πιο εκφραστικό τρόπο για τον ορισμό των αντιστοιχιών σε μια ομάδα δεδομένων RDF. Το R2RML (Σχεσιακή Βάση Δεδομένων με τη Γλώσσα Χαρτογράφησης RDF) μπορεί να χρησιμοποιηθεί για να εκφράσει τις αντιστοιχίσεις μεταξύ μιας σχεσιακής βάσης δεδομένων και του RDF που μπορεί στη συνέχεια να χειριστεί μια μηχανή R2RML. Το R2RML μπορεί να χρησιμοποιηθεί για τη δημοσίευση RDF από σχεσιακές βάσεις δεδομένων με δύο τρόπους.

Πρώτον, τα δεδομένα θα μπορούσαν να μετατραπούν σε παρτίδες RDF dumps. Τα οποία θα μπορούν στη συνέχεια να φορτωθούν σε ένα triplestore RDF. Το endpoint του triplestore θα μπορούσε στη συνέχεια να χρησιμοποιηθεί για την εκτέλεση ερωτημάτων SPARQL έναντι του συνόλου δεδομένων RDF. Δεύτερον, η μηχανή R2RML θα μπορούσε να χρησιμοποιηθεί για να μεταφράσει τα ερωτήματα SPARQL σε ερωτήματα SQL που μπορούν να εκτελεστούν κατά της σχεσιακής βάσης δεδομένων.

Το 2012, το W3C έκανε δύο συστάσεις για χαρτογράφηση μεταξύ σχεσιακών βάσεων δεδομένων και RDF. Η πρώτη πρόταση ορίζει μια άμεση αντιστοίχιση μεταξύ της βάσης δεδομένων και του RDF. Αυτό δεν επιτρέπει τη χαρτογράφηση του λεξιλογίου ή τη διασύνδεση, απλώς η δημοσίευση περιεχομένου

βάσης δεδομένων σε μορφή RDF χωρίς πρόσθετο μετασχηματισμό. Η σύσταση άμεσης χαρτογράφησης δεν έχει σημασία εδώ, καθώς επιθυμούμε επίσης να χαρτογραφήσουμε αντικείμενα στη βάση δεδομένων (για παράδειγμα καλλιτέχνες μουσικής) σε υπάρχοντα URI, παρόλο που αυτά τα URI δεν περιλαμβάνονται στην ίδια τη βάση δεδομένων. Η δεύτερη σύσταση είναι το R2RML που παρέχει ένα μέσο εκχώρησης οντοτήτων στη βάση δεδομένων σε κατηγορίες και χαρτογράφηση αυτών των οντοτήτων σε τριπλέτες subject-predicate-object. Επιτρέπει επίσης την κατασκευή νέων URI για οντότητες και τη διασύνδεσή τους με το υπόλοιπο γράφημα RDF.

3.4 Logic and Presentation layers

Όταν τα ολοκληρωμένα δεδομένα είναι διαθέσιμα σε πραγματικό χρόνο ή σε ένα triplestore, μπορούν να χρησιμοποιηθούν και να έχουν πρόσβαση από τα επίπεδα λογικής και παρουσίασης. Όπως αναφέρθηκε παραπάνω, κάποια από τη λογική της εφαρμογής μπορεί να υλοποιηθεί στο επίπεδο δεδομένων με τη συλλογιστική πάνω στο triplestore. Άλλες μορφές επεξεργασίας που δεν μπορούν αργότερα να υλοποιηθούν στα δεδομένα πραγματοποιούνται στο λογικό στρώμα. Αυτό μπορεί να συνεπάγεται την εφαρμογή μεθόδων στατιστικής ή μηχανικής μάθησης για την εξαγωγή συμπερασμάτων από τα δεδομένα. Μια πληρέστερη επεξεργασία σύμφωνα με τις ανάγκες της εκάστοτε εφαρμογής που αφορά διαδικασίες σύνθεσης και αποτίμησης λογικών οντοτήτων ως προς το χρήστη η ένα σύστημα γίνεται σε αυτό το επίπεδο που κατά βάση προσφέρει από μεριάς υλοποίησης μια πιο υψηλού επιπέδου επεξεργασία των δεδομένων από αυτή του στρώματος αποθήκευσης. Μορφές διασυνδεδεμένων δεδομένων που χαρακτηρίζουν αξιόλογες οντότητες σε σχέση με τις ανάγκες του χειριστή αλλά και διαδικασίες οι οποίες πρέπει να συνταχθούν ώστε να διατεθούν στο στρώμα παρουσίασης είναι οι κύριες λειτουργίες του στρώματος. Άλλες μορφές επιχειρησιακής λογικής, όπως οι ροές εργασίας, υλοποιούνται επίσης σε αυτό το επίπεδο. Τέτοιου είδους διαδικασίες αποσκοπούν ώστε η επεξεργασία των δεδομένων να περνά από στάδια τα οποία ακολουθούν μια λογική ροή παρασκευής εννοιών και συμπερασμάτων που αντιπροσωπεύουν στοιχεία και ιδιότητες αντικειμένων ή των ενδιάμεσων σταδίων τους. Αυτή η επεξεργασία αποσκοπεί όχι μόνο για την τροποποίηση αλλά και για ενδιάμεσα στάδια διάθεσης σε άλλα συστήματα, ή επικοινωνίας με αυτά και γενικότερου τύπου λειτουργίες που υλοποιεί ένα σύστημα που περιγράφει μια λογική διαδικασία σε στάδια.

Τέλος, το στρώμα παρουσίασης εμφανίζει τις πληροφορίες στον χρήστη σε διάφορες μορφές, συμπεριλαμβανομένου του κειμένου, των διαγραμμάτων ή άλλου τύπου τεχνικών απεικόνισης. Η τελική μορφή των δεδομένων είναι κατασκευασμένη έτσι ώστε οι διαπροσωπείες χρηστών να παρουσιάζουν με δομημένη την πληροφορία οντότητες και να τις διαθέτουν στον χρήστη. Σε αυτό το στάδιο δεν συνεπάγεται ότι κάθε μορφή θα αποσκοπεί σε μια απλή παρουσίαση αλλά και σε αντικείμενα που μπορεί να είναι χρήσιμα για μια εφαρμογή. Το τελικό στρώμα παρουσίασης είναι μια υποτιθέμενη αναπαράσταση που έχει βάση σε μια οντότητα που χρίζεται αναγκαία από ένα σύστημα το οποίο τη ζητά και επιστρέφεται σε αυτό. Αυτή η διαδικασία στα περισσότερα συστήματα γίνεται όπως αναφέραμε και παραπάνω με APIs σε εφαρμογές χρηστών και συστήματα τα οποία τις χρησιμοποιούν. Σημασία έχει να κατανοηθεί ότι αυτό το στάδιο είναι το τελικό στάδιο διάθεσης της πληροφορίας που αναζητάτε και σχηματίζεται μετά την συγκομιδή των πληροφοριών από οντότητες και της επεξεργασίας που υφίστανται από το ενδιάμεσο στάδιο της επεξεργασίας.

3.5 Αρχιτεκτονική προσέγγιση Υλοποίησης

Στο προηγούμενο κεφάλαιο έγινε μια ανάλυση αναγκών των οντοτήτων και των λειτουργιών μιας διεπαφής ηλεκτρονικής βιβλιοθήκης. Παρουσιάστηκαν οι ανάγκες από μεριάς των χειριστών και του προγραμματιστή. Στο παρόν κεφάλαιο έγινε μια αναφορά σχετικά με τις αρχιτεκτονικές των προγραμματιστικών υλοποιήσεων και των αρχιτεκτονικών patterns που ακολουθούνται όταν πρόκειται να υλοποιηθεί ένα υπολογιστικό μοντέλο. Αναφορά έγινε επίσης στους τρόπους με τους οποίους μια πληροφορία μεταφράζεται σε SPARQL ερωτήματα και πραγματοποιείται η σύνθεση τέτοιων ερωτημάτων για την άντληση της πληροφορίας από SPARQL End points.

Σύμφωνα με την αναφορά των 3-tier applications που ακολουθούνται έγιναν ορατά τρία βασικά επίπεδα. Με την καταγραφή αυτών ένας κύκλος εργασιών μετάφρασης των λειτουργιών του προγράμματος που υλοποιείται, άντληση πληροφοριών, σύνθεση δεδομένων και διάθεσης στην εφαρμογή παρατηρήθηκε. Για τους σκοπούς της εργασίας παρουσιάζουμε μια αρχιτεκτονική τριών επιπέδων δυναμικής προσέγγισης που διαθέτει διασυνδεδεμένα δεδομένα σε μια διεπαφή. Το αρχιτεκτονικό Pattern που ακολουθούμε είναι αυτό του on the fly. Τα actions του χρήστη στην διεπαφή μας προκαλούν ενέργειες συγκέντρωσης πληροφοριών και φιλτραρίσματος σε οντότητες διασυνδεδεμένων δεδομένων της DBPedia. Οι οντότητες αυτές, όπως αναφέραμε σε προηγούμενο κεφάλαιο, ποικίλουν και με την επέκταση της εφαρμογής πληθαίνουν σχηματίζοντας ερωτήματα συλλογής και φιλτραρίσματος πληροφορίας. Σημαντική παρατήρηση είναι ότι κατά την συγκέντρωση μιας οντότητας νέες μικρότερες οντότητες οι οποίες αποτελούν μέρος της δόμησης ενός αντικειμένου προκύπτουν οι οποίες είναι απαραίτητες για την συλλογή της καθολικής πληροφορίας.

Με βάση την αρχιτεκτονική τριών επιπέδων που παρουσιάζουμε διαχωρίζουμε τις ανάγκες της υλοποίησης σε 3 επίπεδα παράγοντας τον ίδιο κύκλο εργασιών όπως αναφέραμε. Σε πρώτη φάση μια ενέργεια γίνεται trigger η οποία έχει σκοπό την άντληση μιας σύνθετης πληροφορίας με εμφωλευμένες οντότητες που την απαρτίζουν. Για την κατασκευή μιας γενετικού τύπου γεννήτριας ερωτημάτων τέτοιου τύπου εκμεταλλευόμαστε το γεγονός ότι τα properties των αντικειμένων που θέλουμε να δημιουργήσουμε είναι είτε literals που παρουσιάζουν μια πληροφορία άμεσα, είτε τύπου αντικειμένου που η τιμή τους είναι ένα IRI. Για παράδειγμα ας αναλύσουμε ένα σύγγραμμα. Για την περισυλλογή ιδιοτήτων όπως ο τίτλος, μια σύντομη περιγραφή και ο αριθμός των σελίδων διατίθενται άμεσα από τις τιμές των properties rdfs:label, rdfs:comment, dbpedia:ontology/numbersOfPages. Άλλες ιδιότητες όπως ο εκδότης, ο συγγραφέας και η θεματολογία του βιβλίου ή η κατηγορία που ανήκει έρχονται σε μορφή Uri από properties όπως dbpedia:ontology/publisher, author , literalyGenre. Με αυτή την ιδιότητα εξιδανικεύουμε έναν δυναμικό τρόπο παραγωγής ερωτημάτων άντλησης πληροφορίας κατασκευάζουμε δυναμικά URL ερωτημάτων που στοχεύουν στο endpoint της DBPedia για την άντληση δεδομένων και επιλέγουμε τα χαρακτηριστικά που επιθυμούμε να είναι ορατά στο χρήστη. Κάθε αντικείμενο της JavaScript που θέλουμε να κατασκευάσουμε έχει properties που έχουν τιμές λεκτικά ή αντικείμενα που δημιουργούν νέα ερωτήματα SPARQL για την περισυλλογή των στοιχείων τους.

Όπως είναι διακριτό για την κατασκευή δεδομένων σε μοντέλα της JavaScript χρειάζονται πολλές ερωτήσεις τέτοιου τύπου που η κάθε μια εξ' αυτών αντιστοιχεί σε κάθε ένα Property αντικειμένου που θέλουμε να χρησιμοποιήσουμε. Τέτοιου τύπου ερωτήματα μπορούν να γίνουν ασύγχρονα για την συγκέντρωση της πληροφορίας και να συντεθούν σε ένα αντικείμενο μετά την περάτωσή τους για να αποτελέσουν το τελικό μοντέλο που θα γίνει διαχειρίσιμο από την εφαρμογή. Για την υλοποίηση η ανάλυση σε διάσπαση απλών ερωτημάτων που θα συγκροτήσουν την πληροφορία, η κατασκευή ερωτημάτων SPARQL καθώς και η αντίστροφη διαδικασία της σύνθεσης ενός μοντέλου

αναπαράστασης αναφέρονται σαν διαδικασίες που υλοποιούν τη λογική προσέγγιση της εφαρμογής και ανήκουν στο Logic Layer.

Οι διαδικασίες σύνδεσης στο απομακρυσμένο σημείο και υποβολής του ερωτήματος καθώς και οι ενδιάμεσες καταστάσεις ανάλυσης σε βήματα που στελεχώνουν την επεξεργασία των απλών μοντέλων που επιστρέφονται από τα τελικά σημεία και πλαισιώνουν το Data Layer της εφαρμογής. Για την σύνθεση πληροφορίας από άλλα τελικά σημεία και από απομακρυσμένα συστήματα πληροφοριών, κύριο χαρακτηριστικό είναι τα rest ερωτήματα τα οποία πραγματοποιούνται μέσω APIs. Στην εφαρμογή μας απουσιάζει η λογική της αποθήκευσης των δεδομένων καθώς η διάθεση γίνεται από πληροφορίες που υπάρχουν στον ιστό. Για να γίνει διακριτό το επίπεδο των δεδομένων στο χαμηλό στρώμα επεξεργασίας αυτών χρησιμοποιούμε αρχεία τα οποία είναι αποθηκευμένα σε μορφή αναγνωρίσιμη από τις τεχνολογίες των semantics εκφρασμένα σε turtle μορφή. Το πλεονέκτημα εδώ είναι ότι η πληροφορίες που συλλέγονται με αυτόν τον τρόπο προέρχονται και αυτές από ασύγχρονα ερωτήματα του διαδικτύου και επιστρέφουν τους ίδιους τύπους δεδομένων με αυτούς των ερωτημάτων SPARQL. Για τις ανάγκες της εφαρμογής θα χρησιμοποιήσουμε τύπους δεδομένων JSON και JSON-LD το οποίο επεκτείνει τη δομή του JSON με κεφαλίδες, όπως αναφέρθηκε παραπάνω και δίνει μορφή στα δεδομένα που θέλουμε να χρησιμοποιήσουμε.

Σαν τελικό στάδιο αυτό της αναπαράστασης. Το Presentation Layer της εφαρμογής μας είναι αυτό το οποίο παρουσιάζει στον τελικό χρήστη την πληροφορία ολοκληρωμένη μετά από το στάδιο της σύνθεσης και προσδίδει και τις λειτουργίες που ενεργοποιούν το στρώμα της λογικής της εφαρμογής μέσω actions από JavaScript events. Με αυτόν τον τρόπο αφήνουμε την διαχείριση της πληροφορίας στον χρήστη με μια οπτική αναπαράσταση μέσω templates που ορίζει αλλά και περιορίζει τις λειτουργίες που γίνονται διαθέσιμες από τα αντικείμενα που συλλέγονται. Το Presentation Layer δεν είναι τίποτε παραπάνω από μια διαχείριση js αντικειμένων που παρουσιάζονται μέσω templates εμπλουτίζοντάς τα με events που ενεργοποιούν τις λειτουργίες συγκομιδής και διάθεσης νέων πληροφοριών. Η ιδέα της παρασκευής δομικών στοιχείων που επεξεργάζονται και παρουσιάζουν διασυνδεδεμένα δεδομένα που υπακούν στη δική τους ξεχωριστή διαχείριση και υλοποιούν λογικές οργανισμών και υπηρεσιών.[36]

3.5.1 Ανάλυση Αντικειμένων και ιδιοτήτων

Όσον αναφορά την ανάλυση των αντικειμένων που ενδέχεται να χρησιμοποιηθούν σε μια εφαρμογή, πρέπει να καθορίσουμε τις ανάγκες παρουσίασης από τη σκοπιά του UX. Σε κάθε εφαρμογή προκύπτουν λειτουργίες διαχείρισης πλήθους δεδομένων και των λεπτομερειών τους. Η ομαδική διαχείριση και η λεπτομερής είναι δύο τύποι ερωτημάτων που θα υλοποιήσουμε. Αυτές δεν διαφέρουν κατά πολύ μεταξύ τους και μπορούμε να πούμε ότι η μια γεννά την άλλη καθώς για παράδειγμα θα θέλαμε να δούμε στις λεπτομέρειες ενός συγγραφέα όλα τα συγγράμματα του. Είναι και οι δύο διαδικασίες περισυλλογής πληροφοριών αλλά διαφέρουν στον τρόπο χρησιμοποίησης τους από το χρήστη και δημιουργίας ερωτημάτων πληροφοριών στα τελικά σημεία. Άμεσα διαχωρίζουμε τους δυο τύπους ερωτημάτων που σχηματίζονται σε μια τέτοια περίπτωση. Η μια έχει να κάνει με τις λεπτομέρειες του συγγραφέα σαν οντότητα και η άλλη με τη συλλογή των συγγραμμάτων με φίλτρο τον συγκεκριμένο συγγραφέα.

Σαν πρώτη φάση οι ανάγκες διάθεσης και παρακολούθησης των αντικειμένων σε οντότητες συγγραμμάτων, συγγραφέων και οργανισμών είναι σε πλειάδες που αποτελούνται από μεγάλες

συλλογές αλλά με περιορισμένες πληροφορίες για το κάθε αντικείμενο. Για παράδειγμα η παρουσίαση μιας λίστας αποτελεσμάτων δεν είναι απαραίτητο να περιέχει πολλές πληροφορίες για το σύγγραμμα και μπορεί να υλοποιηθεί σε ένα καθολικό ερώτημα. Από την άλλη κάποιες από αυτές τις ιδιότητες ενδέχεται να χρησιμοποιηθούν σαν φίλτρα για να δώσουν μορφή σε μια διεπαφή. Αυτές με την σειρά τους αποτελούν ξεχωριστά ερωτήματα τα οποία συγκεντρώνουν ένα πλήθος διαφορετικών εγγραφών και συλλογής ξεχωριστών αντικειμένων. Σε αυτού του τύπου τα ερωτήματα μας ενδιαφέρει να δημιουργήσουμε και να διαχειριστούμε collections από αντικείμενα φιλτραρισμένα με πεπερασμένες ιδιότητες και να τα παρουσιάσουμε στον χρήστη ομαδοποιημένα ή ανά σελίδες.

Οι ιδιότητες των αντικειμένων σε μορφές ομαδικής παρουσίασης διαχωρίζονται σε δύο. Αυτές που παρουσιάζονται μαζί με τη συλλογή της κύριας περισυλλογής, όπως ο τίτλος ενός βιβλίου, η εικόνα ή ενδεχομένως μια κατηγορία βιβλίου όσον αναφορά τα συγγράμματα και αυτές της συλλογής πληροφοριών σχετικά με το φιλτράρισμα και με τις ιδιότητες που υπάρχουν από κοινού. Όπως για παράδειγμα τις χώρες συνολικά που αναφέρονται σε όλα τα συγγράμματα, τις γλώσσες, τις κατηγορίες, τους εκδότες κα. Οι τελευταίες είναι ερωτήματα SPARQL τα οποία αναφέρονται σε ολόκληρη την συλλογή του end point με distinct δεδομένα συλλογής, αλλά μικρό εύρος από properties που στελεχώνουν την πληροφορία. Οι παραπάνω διαδικασίες ομαδικής συλλογής μπορεί να αναπτυχθούν σε διαφορετικά ερωτήματα περισυλλογής πληροφοριών.

Για τις ανάγκες της λεπτομερής παρουσίασης μιας οντότητας όπως περιγράψαμε παραπάνω γίνεται αντιληπτή μια εξάρτηση σε σχέση με τις αλληλεπιδράσεις που δύναται να παρουσιαστούν σε μια διεπαφή η οποία μπορεί να υλοποιηθεί με events στο επίπεδο παρουσίασης. Σε κάθε περίπτωση η ανάγκη που δημιουργείται είναι η γνώση των properties των αντικειμένων. Το schema.org με τις ιδιότητές του προσφέρει την πληροφορία που είναι αναγκαία για την ύπαρξη οντοτήτων και χαρακτηριστικών τέτοιων αντικειμένων. Για την διαχείριση της κάθε οντότητας ο προγραμματιστής θα πρέπει να γνωρίζει ποιες ιδιότητες των διασυνδεδεμένων δεδομένων θέλει να επεξεργαστεί και να παρουσιάσει καθώς και ποιες από αυτές θέλει να επεκτείνει με σκοπό τη συλλογή ένθετων αντικειμένων που περιέχουν στοιχεία από διαφορετικά αντικείμενα που με τη σειρά τους δημιουργούν επιπλέον ερωτήματα.[37]

Για την κατασκευή των ερωτημάτων που θέλουμε να παρουσιάσουμε και μιας generic διαχείρισης θα κάνουμε χρήση των ιδιοτήτων των διασυνδεδεμένων δεδομένων μέσο παραμετροποίησης επιλέγοντας ποιες από αυτές θέλουμε να διαχειριστούμε μέσω configuration αρχείων. Περαιτέρω ανάλυση θα γίνει σε επόμενο κεφάλαιο όπου θα παρουσιαστούν και modules που υλοποιούν τις προαναφερθείσες λειτουργίες. Η κύρια εστίαση για την κατασκευή και την συγκομιδή των απαραίτητων πληροφοριών έχει στόχο τον εμπλουτισμό των σημερινών αρχιτεκτονικών προσανατολισμένων προς υπηρεσίες (SOA) με σημασιολογικούς φορμαλισμούς, παρέχοντας έτσι υπηρεσίες Σημασιολογικού Ιστού ως επαναχρησιμοποιήσιμα και κλιμακούμενα συστατικά λογισμικού.[38] Έχουν γίνει επίσης μερικές απόπειρες δημιουργίας δομικών στοιχείων Σημασιολογικού Ιστού με την ενσωμάτωση των υφιστάμενων εξαρτημάτων που βασίζονται στο Web.[39] [40]

3.5.2 Οπτικοποίηση αντικειμένων

Σύμφωνα με όσα αναφέρθηκαν σχετικά με την ανάλυση των αντικειμένων και με το στρώμα παρουσίασης η διαδικασία που τα δεδομένα παρουσιάζονται στον χρήστη μέσω μιας διεπαφής έχει να κάνει καθαρά με τις τεχνολογίες front – end. Στη τελική τους μορφή μετά από τη διαδικασία της συλλογής της πληροφορίας τα αντικείμενα θα είναι της μορφής αυτής των JavaScript αντικειμένων. Τα

τρία βασικά χαρακτηριστικά που χαρακτηρίζουν αυτά τα αντικείμενα είναι οι ιδιότητες με τις τιμές τους και οι συναρτήσεις που καθορίσουν τη λειτουργία τους.

Τα properties των αντικειμένων της JavaScript μπορούν να έχουν τιμές από primitive τύπους που χαρακτηρίζουν κάποιες ιδιότητες κυρίως λεκτικές αλλά να είναι και τύποι αντικειμένων με περισσότερη πληροφορία και λειτουργικότητα. Όπως έχουμε αναφέρει σχετικά με του τύπους επιστροφής των ιδιοτήτων των διασυνδεδεμένων δεδομένων αυτοί μπορεί να είναι literals ή URIs. Για να πετύχουμε μια αντιστοίχιση θα θεωρήσουμε ότι τα Literals αντιμετωπίζονται ως primitive types και τα URIs ως ένθετα αντικείμενα τα οποία για τη συλλογή της πληροφορίας θα κατασκευάζουμε ερωτήματα SPARQL τα οποία θα επιστρέφουν ξεχωριστά την ιδιαίτερη αυτή συλλογή. Με αυτόν τον τρόπο το πρόβλημα για την παρουσίαση θα γενικοποιηθεί σε πρόβλημα ασύγχρονης περισυλλογής και σύνθεσης στο τελικό αντικείμενο.

Με τη χρήση ξεχωριστών templates που υλοποιούν την κάθε οντότητα αλλά και templates για τη διαχείριση των ένθετων αντικειμένων το πρόβλημα της παρουσίασης καθίσταται εύκολο και θέμα configuration της κάθε οντότητας που θα αντιστοιχεί στο εκάστοτε απαραίτητο δομικό στοιχείο για το σχηματισμό της.[41] Αντιμετωπίζοντας και παρουσιάζοντας την κάθε ιδιότητα σαν ξεχωριστό αντικείμενο με το δικό του template για την παρουσίαση προσδίδουμε ιδιότητες ξεχωριστά. Αυτό μπορεί να υλοποιηθεί με παραμετροποίηση ιδιοτήτων ανά επίπεδο παρουσίασης αλλά και με ορισμένες ιδιότητες που υλοποιούνται στον κάθε component που χαρακτηρίζει το αντικείμενο που περιγράφουμε.

Για τις ανάγκες της παρουσίασης στην υλοποίησή μας, παρουσιάζουμε δύο κύριους τύπους διαπροσωπειών που αποτελούνται από μικρότερους ανεξάρτητους components που χρησιμοποιούν τις ιδιότητες κάθε αντικειμένου ξεχωριστά και εκμεταλλεύονται τα κύρια χαρακτηριστικά των διασυνδεδεμένων δεδομένων γενικά.

Η πρώτη από αυτές είναι μια διαπροσωπεία που αναγνωρίζει τα properties κάθε οντότητας και αναζητά στον σύνδεσμο καταχωρίσεις σύμφωνα με φίλτρα. Η συλλογή έχει να κάνει με αντικείμενα ίδιου τύπου που παρουσιάζονται σελιδοποιημένα. Στην ίδια διαπροσωπεία υλοποιείται μια διαδικασία περισυλλογής διακριτών ιδιοτήτων που χαρακτηρίζουν την αναζήτηση και χρησιμοποιούνται σαν φίλτρα για την βασική αναζήτηση. Οι components που υλοποιούν την συγκεκριμένη διεπαφή χωρίζονται σε αυτούς που υλοποιούν τις ιδιότητες της συλλογής σαν φίλτρα και αλληλοεπιδρούν με τον κύριο component που υλοποιεί την διαδικασία της αναζήτησης. Τα περιφερειακά αυτά components θα τα ονομάζουμε facets και θα φροντίσουμε να έχουν μεθόδους παρουσίασης που εφαρμόζονται σε κάθε τύπου ιδιότητα ενός αντικειμένου έτσι ώστε να μπορούν να λειτουργήσουν για κάθε τύπου συλλογή. Με αυτόν τον τρόπο θα δώσουμε μια ιδιότητα generic προσέγγισης στις ιδιότητες αντικειμένων των συλλογών με τις οποίες θα μπορούμε να καθορίσουμε τα φίλτρα στις παραγόμενες SPARQL ερωτήσεις για τη διαδικασία της αναζήτησης.

Η δεύτερη διαπροσωπεία που θα παρουσιάσουμε θα είναι αυτή της παρουσίασης των λεπτομερειών μιας οντότητας. Αυτή θα απαρτίζεται από μικρότερα components τα οποία υλοποιούν τις ιδιότητες μιας συγκεκριμένης οντότητας. Μέσω configuration αρχείων που θα καθορίζουν τον τρόπο εμφάνισης της κάθε ιδιότητας θα πετύχουμε το template binding, ή καλύτερα component binding, το οποίο και θα δίνει ιδιότητες στο κάθε property μέσω ανεξάρτητων κλήσεων και συναρτήσεων. Σκοπός είναι κάθε ιδιότητα αντικειμένου που περιγράφεται στο συγκεκριμένο επίπεδο να είναι σε θέση να περιγράψει και να υλοποιήσει όχι μόνο μια τιμή αλλά και μια ολόκληρη λογική διαχείριση που θα κρύβεται πίσω από ένα εκφρασμένο χαρακτηριστικό. Σε αυτή την κατάσταση η διαδικασία παρουσίασης χωρίζεται σε

μικρότερες ανάγκες που χρησιμοποιούν components για να παρουσιάσουν τις ιδιότητες του αντικειμένου και μπορούν να υλοποιηθούν βάση του δέντρου της οντολογίας που προσφέρει ο σύνδεσμος. Σε αυτό το επίπεδο κάθε ιδιότητα αντιμετωπίζεται σαν ξεχωριστό αντικείμενο και μπορεί να λειτουργήσει ξεχωριστά εκμεταλλευόμενοι τις ιδιότητες των ασύγχρονων κλίσεων για την περισυλλογή των λεπτομερειών και του rendering των τεχνολογιών της JavaScript.

Με βάση την παραπάνω λογική διαχωρίζοντας τις ιδιότητες των αντικειμένων σε ξεχωριστές μονάδες αυτοδιαχειριζόμενες και αυτολειτούργητες προσδίδουμε στην εφαρμογή μας μια τύπου γνώση η οποία μπορεί να λειτουργήσει για κάθε τύπου αντικείμενο χωρίς πραγματικά να γνωρίζει τις εξαρτήσεις από άλλα αντικείμενα και να καθιστά την κάθε δομική μονάδα ανεξάρτητη. Σκοπός είναι να υλοποιήσουμε τέτοιες ανεξάρτητες μονάδες που με τη σύνθεσή τους να μπορούν να δώσουν μια γενικότερη εικόνα σε επίπεδο παρουσίασης στον τελικό χρήστη.

3.5.3 Στρώμα λογικής και Σύνθεσης Πληροφορίας

Στο Logic Layer της εφαρμογής της εφαρμογής μας κυριαρχούν 2 βασικές ιδιότητες, αυτή της ανάλυσης της πληροφορίας και η αντίστροφη της σύνθεσης. Η γενετική προσέγγιση που παρουσιάζουμε έχει να κάνει με το πώς ένα αντικείμενο μεταμορφώνεται σε ξεχωριστές απλοποιημένες οντότητες και δημιουργεί ερωτήματα σε γλώσσα SPARQL και απομακρυσμένα συστήματα μέσω REST calls. Η διαδικασία της ανάλυσης εξαρτάται αποκλειστικά από τα properties των αντικειμένων των διασυνδεδεμένων δεδομένων και αυτών που σχετίζονται με δεδομένα περισυλλογής από external APIs.

Στη συνέχεια θα μελετήσουμε και θα αναλύσουμε τις δύο αυτές λειτουργίες ξεχωριστά και θα παρουσιάσουμε ένα τρόπο σύνθεσης των δύο τύπου κλήσεων σε ένα καθολικό αντικείμενο που επιστρέφεται στο στρώμα παρουσίασης. Πριν όμως προχωρήσουμε σκόπιμο είναι να εξετάσουμε τις ανάγκες αυτής της προσέγγισης. Για τις ανάγκες συλλογής δεδομένων ένας διαχωρισμός που γίνεται διακριτός είναι αυτός των διαφορετικού τύπου ερωτημάτων που πραγματοποιούνται. Τα μεν διασυνδεδεμένα δεδομένα προέρχονται από τελικά σημεία SPARQL τα οποία μπορούν να επιστρέψουν την πληροφορία σε JSON-LD με την πραγματοποίηση ενός ερωτήματος. Για αυτού του τύπου τα δεδομένα αναγκαία είναι η σύνταξη ενός SPARQL ερωτήματος, ενώ για τα λοιπά δεδομένα που προέρχονται από External APIs απαραίτητη είναι η διασύνδεση με END points και η γνώση του τύπου της πληροφορίας που θα επιστρέψει το Resource. Και τα δύο υλοποιούνται μέσω HTTP calls στα οποία απαραίτητες είναι η επικεφαλίδες ερωτημάτων η γνώση του URL, οι ιδιότητες της ταυτοποίησης στοιχείων αν αυτές απαιτούνται και οι παράμετροι για κάθε τύπου ερώτημα.

Για τη σύνταξη των ερωτημάτων των διασυνδεδεμένων δεδομένων απαραίτητη είναι η σύνταξη ενός SPARQL ερωτήματος καθώς και ο τύπος επιστροφής των δεδομένων. Ένα end point όπως αυτό της DBPedia δέχεται REST κλήσεις όπως κάθε κοινό endpoint με παραμέτρους query και format τα οποία παρσάρονται μέσω ενός URI. Έτσι λοιπόν για την διαδικασία της περισυλλογής δεδομένων απαιτείται η σύνταξη ενός τέτοιου τύπου ερωτήματος. Για τις ανάγκες της σύνταξης βάση της προηγούμενης ανάλυσης υλοποιούμε services τα οποία παράγουν ερωτήματα επεξεργαζόμενα τους γράφους των δεδομένων για κάθε ανάγκη.

Σύμφωνα με την καταγραφή των αντικειμένων παραπάνω για να παρουσιάσουμε την αναζήτηση και διαχείριση μιας συλλογής αντικειμένων υπάρχει η ανάγκη για υλοποίηση λειτουργιών όπως:

- Αναζήτηση με search term λεκτικό
- Επιλογή οντοτήτων που αποτελούν μέρος της συλλογής
- Μέτρηση πλήθους αποτελεσμάτων

- Σελιδοποίηση
- Επιλογή ιδιοτήτων της συλλογής ως φίλτρα

Η υλοποίηση των παραπάνω μπορεί να γίνει με services τα οποία κατασκευάζουν ένα ερώτημα select της SPARQL τροποποιώντας κατάλληλα το select και το where part με ιδιότητες όπως αυτές του filter και του Optional. Αντίστοιχα απλοποιημένα ερωτήματα μπορούν να συνταχθούν για τις διαδικασίες της μέτρησης αποτελεσμάτων. Οι ανάγκες για την υλοποίηση ενός τέτοιου service είναι να γνωρίζει τα properties και τα resources των αντικειμένων που επεξεργάζεται και να δημιουργεί δυναμικά ένα string με το αντίστοιχο ερώτημα για να δημιουργηθεί η κατάλληλη παράμετρος που είναι απαραίτητη σε ένα GET ερώτημα της DBPedia. Παρόμοια λειτουργία υλοποιείται και για ερωτήματα τύπου POST και DELETE. Σημαντική προϋπόθεση για την υλοποίηση τέτοιων services είναι η ανάλυση σε όσο το δυνατόν πιο απλά ερωτήματα τα οποία σε συνεργασία μεταξύ τους μπορούν να υλοποιήσουν τις ανάγκες της κάθε εφαρμογής. Σημαντικό είναι να εκμεταλλευτούμε την ιδιότητα των ασύγχρονων διαφορετικών κλήσεων και να διασπάσουμε τις ανάγκες της πληροφόρησης σε όσο το δυνατόν πιο απλά ερωτήματα.

Σχετικά με την επιστροφή των ερωτημάτων αναζήτησης σε end points διασυνδεδεμένων δεδομένων τα ίδια τα δεδομένα επιστρέφονται σε JSON-LD με ιδιότητες που χαρακτηρίζονται από το head της επιστροφής. Με αυτόν τον τρόπο μπορεί και προστίθεται μια λογική επεξεργασία των δεδομένων και δίνεται μια μορφή σε κάθε σύνδεσμο που επιστρέφεται. Κοινός με την επιστροφή δομημένων αποτελεσμάτων το σύστημα που είναι σε θέση να αντιλαμβάνεται τι να χρησιμοποιήσει και να παρουσιάσει προσδίδει λειτουργίες σύνθετες όσον αναφορά τα δεδομένα που επεξεργάζεται. Για την διαχείριση αυτής της πληροφορίας και την επεξεργασία πριν τα στρώματα της παρουσίασης και διάθεσης στους τελικούς components της διαπροσωπείας, όπως και για λόγους ενεργοποίησης events σε άλλους διασυνδεδεμένους components της εφαρμογής τα δεδομένα πρέπει να υποστούν κάποιου είδους επεξεργασία. Αυτό το utilization process το οποίο είναι και υπεύθυνο για την προ επεξεργασία των δεδομένων όπως αυτά επιστρέφονται από την εκάστοτε κλήση διαβάζει τα configuration files στα οποία αναφέρονται τα properties των διασυνδεδεμένων δεδομένων που έχουν οριστεί και αντιστοιχεί τον τρόπο παρουσίασής τους σε βασικά components τα οποία υλοποιούν ξεχωριστές λειτουργίες για κάθε τύπο πληροφορίας.

Αντίστοιχα services που υλοποιούν τις λειτουργίες της διάθεσης και επεξεργασίας των δεδομένων που σχετίζονται με external APIs και back end συστήματα είναι αναγκαία για μια διαφορετικού τύπου επεξεργασία δεδομένων από αυτή των ανοικτών δεδομένων. Τα services αυτά υλοποιούν ασύγχρονα promises που αναφέρονται σε REST κλήσεις και αποσπούν την απαραίτητη πληροφορία και κατά την επιστροφή τους πραγματοποιούν τον αντίστοιχο μετασχηματισμό σε οντότητες που είναι γνωστές για το περιβάλλον που υλοποιούμε μέσω μιας utilization διαδικασίας. Λόγω των ασύγχρονων τεχνολογιών και της διαδικασίας του rendering δεν χρήζει απαραίτητη η πρόσμιξη των διαφορετικών στοιχείων μεταξύ τους τόσο όσο η ανάλυση του κάθε template που υποστηρίζει μικτά δεδομένα σε ξεχωριστές διαδικασίες ανανέωσης και επεξεργασίας της πληροφορίας. Με αυτόν τον τρόπο επιτυγχάνεται μια ξεχωριστή διαχείριση της πληροφορίας σε κύκλους επεξεργασίας και παρουσίασης εκτός αν απαιτείται η δημοσίευση τιμών κατά σε διασυνδεδεμένα δεδομένα οπότε και χρήζει απαραίτητος ο ορισμός ενός rdf γραφήματος.

3.5.4 Αναπαραγωγή ερωτημάτων και Διαχείριση δεδομένων

Για την συγκομιδή των δεδομένων και τη διαχείριση πόρων σχετικά με την πραγματοποίηση ερωτημάτων περί περισυλλογής δεδομένων, σύμφωνα με όσα αναφέραμε και είναι γνωστό απαιτείται η σύνδεση με απομακρυσμένα σημεία τα οποία προσφέρουν την πληροφορία. Η διασύνδεση και η υποβολή ερωτήσεων πραγματοποιείται με HTTP calls. Αυτές οι κλίσεις όσον αναφορά τα διασυνδεδεμένα δεδομένα προϋποθέτει τις αντίστοιχες παραμέτρους στα τελικά σημεία ώστε αυτά να προσδώσουν την αντίστοιχη πληροφορία μέσω APIs που διαθέτουν σε διάφορες μορφές δεδομένων όπως αναφέραμε.

Για τη δυναμική προσέγγιση και την παραγωγή των πηγαίων ερωτημάτων που θέλουμε να παρουσιάσουμε στο Layer της εφαρμογής που εκτίθεται στο web και πραγματοποιεί τις διασυνδέσεις με συστήματα πηγαίας πληροφόρησης, η υλοποίησή μας θα πρέπει να πληροί τις προϋποθέσεις που συζητήθηκαν σχετικά με το στρώμα των δεδομένων παραπάνω και να διαχειρίζεται τα rdf διαγράμματα και τις κλήσεις τις οποίες θα πραγματοποιούν τη περισυλλογή των δεδομένων. Για την δημιουργία SPARQL ερωτημάτων απαραίτητος είναι ο ορισμός ενός γράφου αναπαράστασης δεδομένων ή η χρησιμοποίηση ενός γνωστού όπως αυτός της Schema.org.

Το στρώμα διαχείρισης αυτό θα πρέπει να έχει γνώση της κατάστασης των διασυνδεδεμένων δεδομένων και να είναι ικανό να περισυλλέγει και να διαθέτει στην εφαρμογή μας δεδομένα από διάφορα end points ανά το web. Για τον λόγο αυτό θα πρέπει να είναι ικανό να επεξεργάζεται τα δεδομένα και να έρχεται σε συνεργασία με το layer της λογικής όπως περιγράψαμε για να διαθέτει αυτά σε μορφή που είναι επεξεργάσιμη από την εφαρμογή. Κατά τις κλίσεις των απομακρυσμένων σημείων που υποστηρίζουν ερωτήματα SPARQL απαραίτητος είναι ο ορισμός του ονόματος του γράφου που θα χρησιμοποιηθεί καθώς επίσης και τα prefixes που στελεχώνουν τις οντότητες του ερωτήματος.

Για την γενετική υλοποίηση που παρουσιάζουμε σχετικά με ερωτήματα στο σύνδεσμο της DBPedia και τη χρησιμοποίηση του γράφου αναπαράστασης του schema.org τα prefixes που χρησιμοποιούμε για τη σύνταξη των ερωτημάτων είναι:

- PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
- PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
- PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
- PREFIX owl: <http://www.w3.org/2002/07/owl#>
- PREFIX dcterms: <http://purl.org/dc/terms/>
- PREFIX void: <http://rdfs.org/ns/void#>
- PREFIX foaf: <http://xmlns.com/foaf/0.1/>
- PREFIX skos: http://www.w3.org/2004/02/skos/core#

Τα ερωτήματα που θα μας απασχολήσουν βάση της ανάλυσης που προηγήθηκε για τη διαχείριση δεδομένων σε λίστες είναι αυτά της περισυλλογής από resources με φίλτρα, η καταμέτρηση των αποτελεσμάτων, η αναζήτηση με παράμετρο λεκτικού, η αναζήτηση με προκαθορισμένες οντότητες καθώς και η λεπτομερής αναζήτηση ενός resource.

Πέραν της ανάλυσης και της συγκομιδής πληροφοριών σχετικά με τα δεδομένα το στρώμα επικοινωνίας με το web είναι απαραίτητο να επεξεργάζεται τα δεδομένα με τρόπο ώστε να

εξαλείφονται οι αμφισημασίες και να επιτυγχάνεται μια αντιστοίχιση σχήματος σχετικά με τις οντότητες του web που προϋπάρχουν. Για τις ανάγκες της σύνθεσης της πληροφορίας που επεξεργάζεται μια διεπαφή χρήζει επίσης αναγκαία η διαδικασία του republication σε rdf διάγραμμα για την περιγραφή του συνόλου των δεδομένων που επεξεργάζεται η εφαρμογή. Για να περιγράψουμε την αντιστοίχιση αυτή με υπάρχων διαγράμματα rdf και να καταφέρουμε να ορίσουμε τις νέες οντότητες που γενικεύουν τη λειτουργία των δεδομένων της εφαρμογής μας θα ορίσουμε ένα γράφο οντοτήτων σε σύνταξη turtle. Με την κατασκευή αυτού του γράφου θα είμαστε σε θέση να προσθέσουμε νέα λειτουργικότητα στα ερωτήματα SPARQL που θα συντάσσονται με την ικανότητα να περιγράψουμε σαν οντότητες του ερωτήματος νέα στοιχεία που θα μας απασχολούν και να βοηθήσουμε στη διαδικασία του vocabulary matching χρησιμοποιώντας προ υπάρχων ορισμένες οντολογίες και ιδιότητες. Η κατασκευή του ψευδογράφου αυτού rdf μπορεί να γίνει με την υλοποίηση ενός vocabulary σε ένα rdf dump το οποίο συμπεριλαμβάνεται στην εφαρμογή. Για την κατασκευή και τον ορισμό μπορούμε να δημιουργήσουμε ένα αρχείο .tll με τριπλέτες που ορίζουν τα στοιχεία της εφαρμογής μας και τα αντιστοιχούν με Object Properties, Datatype Properties και Classes της owl. Οι ιδιότητες αυτές θα είναι σε θέση να δώσουν μορφή στα δεδομένα που επεξεργαζόμαστε και θα χρήσει το σύστημα ικανό να αντιληφθεί και να σχηματίσει διαχειρίσιμες οντότητες που θα παρουσιάζονται στον χρήστη. Αυτή η μορφή λογικής θα πρέπει να είναι επεκτάσιμη και στην πραγματικότητα να δίδεται στο σύστημα από εξωτερικές πηγές που τη διαχειρίζονται, αλλά για τους σκοπούς της εργασίας και της μελέτης του χαρακτηρισμού των δεδομένων με την αποφυγή της σύνδεσης σε κάποιο εξωτερικό σύστημα η κατασκευή θα γίνει πειραματικά με αρχείο το οποίο θα διαχειρίζεται η ίδια η εφαρμογή όσον αναφορά την επέκταση και τη σύνταξή του.

Με τη σύνταξη του λεξιλογίου αυτού θα είμαστε σε θέση να αντιστοιχίσουμε άμεσα τις οντότητες που χρησιμοποιούν οι διεπαφές με ιδιότητες των ερωτημάτων και στοιχεία του web που θα συλλεχθούν. Θα πετύχουμε επίσης την άμεση αναπαραγωγή στοιχείων που πλαισιώνουν την εφαρμογή μας σε στοιχεία διασυνδεδεμένων δεδομένων που μπορούν να διασυνδεθούν με υπάρχων υλοποιήσεις. Τα δεδομένα που επιστρέφονται από τις εκάστοτε κλίσεις της εφαρμογής θα επεξεργάζονται μέσω Utilization Processes και θα διατίθενται στους αντίστοιχους μηχανισμούς ενημέρωσης της εφαρμογής με αποτέλεσμα να ενημερώνονται οι διαπροσωπείες που στελεχώνουν την εφαρμογή για το περιβάλλον του τελικού χρήστη.

4. Μεθοδολογία- Υλικό

Στην αναπαράσταση των δεδομένων βάση των αναγκών κάθε μοντέλο μπορεί να προκαλέσει αλλαγές στην ίδια την αναπαράσταση ή κατάσταση ενός UI και να μεταβάλει την κατάσταση του εκάστοτε συστήματος σε περισσότερο από ένα σημεία. Αυτά λόγω της μεταβολής τους θα πρέπει να αναπαρασταθούν ξανά και να παρουσιάσουν χειρισμούς και πληροφορία η οποία μπορεί και να μη διαφέρει αλλά σίγουρα μεταβάλλεται.

Θέλοντας να αναπαραστήσουμε τα διασυνδεδεμένα δεδομένα σαν μοντέλα υπο-μοντέλων που δρουν και λειτουργούν αυτόνομα (θυμηθείτε μεταφορικά υλοποιούμε τον κόσμο και ενδεχόμενες εξελίξεις του), πρέπει να σεβαστούμε την αρχή της γενικότητας προγραμματιστικά (abstraction is a rule). Τα Linked Data λειτουργούν και παρέχουν αυτόνομα αλλά και ανεξάρτητα πληροφορία και λειτουργίες επεκτάσιμες και γνωστές. Έχουν σαν κύριο χαρακτηριστικό μια δομή (ίσως αλλά και μεταβλητή) η οποία ανάγεται σε σχέσεις μεταξύ τους και περιγράφεται με συνδέσμους αλληλεξάρτησης (type on an Endpoint).

Ο όγκος του προβλήματος επέρχεται στις λέξεις “μεταβαλλόμενες δια-συνδεδεμένες αλληλεξαρτήσεις” και όταν περισσότερο από μια μεταβαλλόμενες αλληλεξαρτήσεις περιγράφουν προγραμματιστικά αίρουν και παραλλάσουν κατά το δοκούν προγραμματιστικές λογικές, και κανόνες υλοποίησης και παρουσίασης. Για το λόγο αυτό η γενετική προσέγγιση που θα παρουσιάσουμε με σκοπό την υλοποίηση μιας ηλεκτρονικής βιβλιοθήκης θα υλοποιηθεί με όσο το δυνατόν ανεξάρτητα κομμάτια τα οποία θα στελεχώσουν την τελική εφαρμογή. Αυτά τα στοιχεία έχουν σκοπό να δρουν ανεξάρτητα και να επικοινωνούν μεταξύ τους μέσω events έτσι ώστε η υλοποίηση τους να διασπαστεί και κοινές λειτουργίες και να αποφευχθεί ο επαναπρογραμματισμός και η υλοποίηση της κάθε οντότητας.[42]

Βασιζόμενοι στις μοντελοποιήσεις των λειτουργιών τέτοιου είδους πληροφορίας όπου όλο και περισσότερες δυνατότητες και λειτουργίες περιγράφουν μια οντότητα και αυτή με τη σειρά της ένα σύνολο ή μια μεγαλύτερη οντότητα παρατηρούμε το χαρακτηριστικό ότι κάποιες ιδιότητες είναι γνωστές ακόμα και κοινές. Για να εκμεταλλευτούμε αυτό το γεγονός δίνοντας μορφή στην κάθε ιδιότητα αντικειμένου που περιγράφεται από το κάθε μοντέλο θα πρέπει αυτό να είναι παρόμοιο με το γράφο των διασυνδεδεμένων δεδομένων (entity graph) και να είναι αντιληπτό από τις τεχνολογίες του ιστού υποστηρίζοντας τις αρχές του SW. Κατά αυτόν τον τρόπο το εκάστοτε μοντέλο θα είναι επεκτάσιμο, και ευμενές στην διασύνδεση του με άλλα χαρακτηρισμένα πρότυπα και τεχνολογίες του σημασιολογικού ιστού και να καθίσταται ικανό να εκθέσει τα δεδομένα που χρησιμοποιεί και δημοσιεύει με τις τεχνολογίες των μεταδεδομένων και των μικροδεδομένων (microdata).[43]

Αυτό σημαίνει 2 πράγματα:

1. Θα πρέπει να υπάρχει ένας σκελετός από σταθερά περιγραφικά στοιχεία (attributes – entities – ontologies) ώστε να χαρακτηρίζονται οι λειτουργίες σε κάθε μοντέλο που περιγράφουν και να επεκτείνονται χρήζοντας απλά ένα abstract κομμάτι λογικής και λειτουργίας.
2. Κάθε κομμάτι να μπορεί να αναπαρασταθεί και να χειριστεί ξεχωριστά για το λόγο ότι μπορεί και αντιπροσωπεύει μια ενδεχόμενη οντότητα και πιθανό μέρος σε μια πιο γενική περιγραφή με χαρακτηριστικά που κληρονομούνται και διαχειρίζονται χωριστά.

Στην ενότητα αυτή θα παρουσιάσουμε την υλοποίηση καθώς και τις λειτουργίες της περιγράφοντας τους components που την πλαισιώνουν. Θα αναλύσουμε τα στάδια επεξεργασίας των δεδομένων και πως μπορούμε να εκμεταλλευτούμε τα κοινά τους στοιχεία στις ενδιάμεσες διαδικασίες. Θα

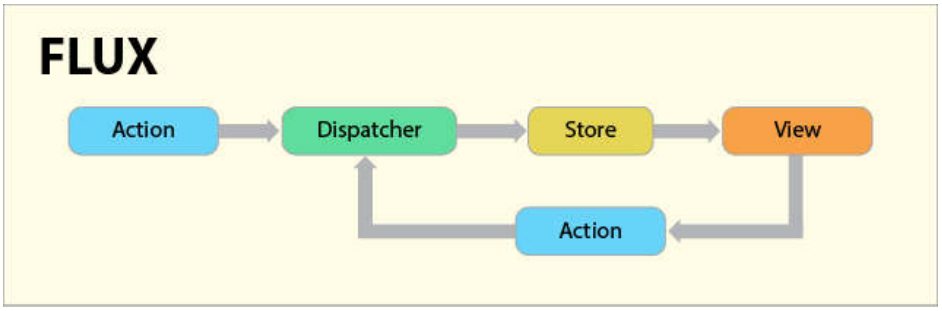
αναλύσουμε την εφαρμογή σε μικρότερα ανεξάρτητα ζητήματα που παρουσιάζουν μια ομοιογένεια καθώς και τον τρόπο επικοινωνίας αυτών. Θα κάνουμε μια διάκριση των τεχνολογιών που χρησιμοποιήθηκαν και θα παρουσιάσουμε το διάγραμμα ροής της πληροφορίας στα ενδιάμεσα στάδια από την περισυλλογή στην ανάλυση και στην τελική διάθεση και σύνθεση της διαπροσωπείας.

4.1 Υπολογιστική προσέγγιση & Γενικοποιημένη Μοντελοποίηση

Σε μια τελική προσέγγιση μιας υπολογιστικής υλοποίησης βρίσκεται ο τελικός χρήστης και οι ανάγκες του. Σε αυτό το επίπεδο δεν γίνεται αντιληπτή η εκάστοτε λειτουργία και ο σκελετός δόμησης μιας εφαρμογής αλλά οι ιδιότητες και μια μορφή φιλικής οπτικοποίησης λειτουργιών και επιλογών. Όλα τα σύγχρονα User Interfaces αποτελούν μια δομή πληροφορίας με επιλογές και χειρισμούς από τον τελικό χρήστη. Στην κατηγορία των Διασυνδεδεμένων δεδομένων οι πληροφορίες αυτές όπως περιγράψαμε γίνονται εν μέρη κατανοητές και διαχειρίζονται από μηχανισμούς αυτόματης συμπλήρωσης κώδικα και μηχανών που με τη σειρά τους προσφέρουν λειτουργίες αυτοματισμού και επικοινωνίας με άλλα πληροφοριακά συστήματα και όχι μόνο με τους τελικούς χρήστες και τους χειρισμούς τους.

Υπάρχουν διάφορα αρχιτεκτονικά πρότυπα τα οποία σχετίζονται με την υλοποίηση διεπαφών για τον τελικό χρήστη προσδίδοντας χειρισμούς οι οποίοι μεταξύ των άμεσα ενδιαφερομένων συντελούν σε καταστάσεις συστημάτων. Μια εφαρμογή που απευθύνεται στον τελικό χρήστη επιφέρει ιδιότητες από Actions τα οποία μεταβάλλουν την κατάσταση του UI και παραλλάσουν τους components που την στελεχώνουν. Για την υλοποίηση της κάθε μονάδας που αποτελεί ένα δομικό στοιχείο της διαπροσωπείας που είναι ορατή στον τελικό χειριστή διαχωρίζουμε την λειτουργία σε μικρότερα κομμάτια τα οποία λειτουργούν ανεξάρτητα και με τη σύνθεσή τους στελεχώνουν και δομούν μεγαλύτερης εμβέλειας σε λειτουργείες οντότητες και διαπροσωπείες.

Για την υλοποίηση της εφαρμογής μας επιλέξαμε να χρησιμοποιήσουμε την τεχνολογία front- end του flux.[44] Αυτή με τη μονής κατεύθυνσης ροή δεδομένων υπακούει σε actions τα οποία γίνονται trigger από κάθε component και μέσω του Dispacher ενημερώνει το context της εφαρμογής και με events ανανεώνει το τελικό στάδιο αναπαράστασης που υπάρχουν τα templates που είναι υπεύθυνα για το binding των δεδομένων. Η τεχνολογία είναι ευρέως διαδεδομένη και γνωστή από το μηχανισμό του facebook και επιτρέπει στις δομικές μονάδες να αντιμετωπίζονται ξεχωριστά.



Διάγραμμα 4: Flux flow Diagram

Με μια ροή δεδομένων μονής κατεύθυνσης, επιτυγχάνει μια προβλέψιμη κατάσταση εφαρμογής. Όταν η ροή δεδομένων είναι μονής κατεύθυνσης, οι αλλαγές στο στρώμα προβολής της εφαρμογής θα ενεργοποιήσουν μια ενέργεια στο επίπεδο δεδομένων. Αυτές οι αλλαγές θα αντικατοπτριστούν στην προβολή. Η προβολή δεν επηρεάζει άμεσα τα δεδομένα της εφαρμογής αλλά δίνει τη δυνατότητα

χειρισμών οι οποίοι ανακλούν πάλι στον Dispatcher ο οποίος είναι υπεύθυνος να ανανεώσει το Store που με την σειρά του θα κάνει εμφανείς τις αλλαγές στο επίπεδο προβολής.

Για την καταγραφή των λειτουργιών και τη διάσπαση σε μικρότερες οντότητες της διεπαφής η βιβλιοθήκη μας θα υλοποιεί components διασυνδεδεμένων δεδομένων για 3 βασικές οντότητες. Αυτές των συγγραμμάτων, των εκδοτών και των συγγραφέων καθώς και οντότητες που τις απαρτίζουν και θα προσφέρει λειτουργικότητα και ενέργειες σχετικά με τις ιδιότητες των αντικειμένων αυτών. Πριν προχωρήσουμε στην ανάλυση της υλοποίησης και για να δώσουμε μια πρώτη εικόνα της εφαρμογής θα προσπαθήσουμε να περιγράψουμε και να αναλύσουμε τις καταστάσεις του συστήματος και να τις αναγάγουμε σε μικρότερες οντότητες που την απαρτίζουν και της δίνουν μετά την σύνδεσή τους μια τελική κατάσταση.

4.1.1 Ανάλυση δέντρου καταστάσεων της εφαρμογής

Για την υλοποίηση της πλατφόρμας ενημέρωσης σχετικά με συγγράμματα που αναφέρονται στο σύνδεσμο της DBPedia, το User Interface μας έχει ένα γράφο καταστάσεων που αντικατοπτρίζει τα routes της εφαρμογής που σχετίζονται με τα διασυνδεδεμένα δεδομένα. Στην προσπάθεια να περιγράψουμε τις οντότητες Book, Writer και Publisher κατασκευάσαμε 2 τύπους διαπροσωπειών για την κάθε μια. Αυτοί έχουν να κάνουν με την αναζήτηση των καταχωρίσεων σε πλήθος δεδομένων ίδιας τάξης (Browse Collection) και στις λεπτομέρειες τις κάθε μιας εξ' αυτών (Entity Details). Η κάθε μια κατάσταση μπορεί να περιέχει τις οντότητες της άλλης και να υπάρχει μια τύπου πρόσμιξη των δεδομένων που την στελεχώνουν είτε για τις ανάγκες πληροφόρησης είτε για τις ανάγκες φίλτρων και προσθήκης ιδιοτήτων στον χρήστη που αλληλοεπιδρά με την εφαρμογή. Στη συνέχεια θα αναλύσουμε κάθε μια από αυτές με τα events ανακατεύθυνσης σε άλλο στάδιο της εφαρμογής καθώς και τα events που πλαισιώνουν την εφαρμογή.

Σαν πρώτο στάδιο υπάρχουν οι 3 καταστάσεις του συστήματος οι οποίες υλοποιούν την ομαδική αναζήτηση και αποτελούνται από την κύρια οντότητα που την περιγράφει. Αυτές είναι:

- Book List : /Books
- Writers List: /Writers
- Publisher List: /Publisher

Οι καταστάσεις αυτές του συστήματος υλοποιούν μια συλλογή των κύριων αντικειμένων στις οποίες αναφέρονται και χρησιμοποιούν τα properties των συλλογών τους για ανάγκες παρουσίασης φίλτρων σε SPARQL ερωτήματα τα οποία υλοποιούνται με ανεξάρτητους components που ενεργοποιούν νέα events περισυλλογής διασυνδεδεμένων δεδομένων από το Web. Στην εφαρμογή μας οι components αυτοί που συγκεντρώνουν και παρουσιάζουν το πλήθος των ιδιοτήτων σε ολόκληρη τη συλλογή ονομάζονται facets και συλλέγουν τις ιδιότητες από ολόκληρο το φάσμα των δεδομένων που περιγράφονται στην αναζήτηση παρουσιάζοντας στοιχεία σχετικά με το συνολικό πλήθος των δεδομένων. Για παράδειγμα οι χώρες που αναφέρονται στα συγγράμματα, οι γλώσσες στις οποίες διατίθενται, οι κατηγορίες στις οποίες ανήκουν κ.α.

Οι καταστάσεις αυτές όπως θα αναλύσουμε παράγουν ερωτήματα σχετικά με την συλλογή δεδομένων συγγραμμάτων, συγγραφέων και εκδοτών. Ο κύριος component που υλοποιείται για να παρουσιάσει αυτή τη συλλογή αλληλοεπιδρά με τα facets τα οποία τον πλαισιώνουν και επανεκτιμάτε η ερώτηση αναζήτησης. Τα facets είναι ξεχωριστά υλοποιημένοι components οι οποίοι δημιουργούν events τα οποία επηρεάζουν το context του store που περιγράφει τη συλλογή. Σκοπός της χρησιμοποίησης τους

είναι η υλοποίηση λογικών ερωτήσεων όπως αναζήτησε όλα τα συγγράμματα που έχουν εκδοθεί σε μια συγκεκριμένη χώρα, αναζήτησε τα συγγράμματα που έχουν εκδοθεί στην Αγγλική γλώσσα, αναζήτησε τα συγγράμματα που έχει εκδώσει ο συγκεκριμένος οίκος και γενικότερα ερωτήματα που μπορούν να εκφραστούν με τις ιδιότητες που περιέχουν τα εκάστοτε δεδομένα καθώς τα facets υλοποιούν συλλογές από δεδομένα που χαρακτηρίζονται σαν properties του βασικού αντικειμένου που αναπαρίσταται.

Οι τρείς καταστάσεις παρουσίασης και λεπτομερούς ανάλυσης για την κάθε οντότητα χαρακτηρίζονται από τις καταστάσεις συστήματος με τη σήμανση Detail και είναι :

- Book's Detail: /Books/Detail/:id
- Writer's Detail : /Writers/Detail/:id
- Publisher's Detail: /Publishers/Detail/:id

Οι καταστάσεις αυτές παρουσίασης λεπτομερειών μιας οντότητας όπως παρουσιάζεται στο παρακάτω διάγραμμά σε αντίθεση με αυτές που περιγράφουν τις λίστες περισυλλογής δεδομένων είναι η παρουσίαση ενός resource πληροφορίας η οποία αναλύεται σε συλλογή από properties ενός προκαθορισμένου δεδομένου και παρουσιάζει τα δομικά του στοιχεία με τη χρήση components που περιγράφουν το κάθε property σαν μονάδα ξεχωριστής διαχείρισης.

Για παράδειγμα κατά την επιλογή ενός Book από μια λίστα δεδομένων που παρουσιάστηκε από μια αναζήτηση, μια ανακατεύθυνση σε ένα διαφορετικό State της εφαρμογής πραγματοποιείται (αυτό των detail) και περισυλλέγεται η οντότητα του βιβλίου σε επίπεδο resource. Το ερώτημα της SPARQL που πραγματοποιείται επιστρέφει δεδομένα από το σύνδεσμο της DBPedia σχετικά με ένα συγκεκριμένο σύγγραμμα και περιέχει όλες τις πληροφορίες που ορίζονται από Properties και Values του συγκεκριμένου αντικειμένου. Για την αναπαράσταση της πληροφορίας ορίζονται διαφορετικά δομικά στοιχεία δηλαδή components που περιγράφουν ιδιότητες που κατά την περισυλλογή επιστράφηκαν σε μορφή συνδέσμου με ένα URI, όπως για παράδειγμα αυτή του Author που περιγράφει τον συγγραφέα, της κατηγορίας του συγγράμματος, μιας γεωγραφικής τοποθεσίας και οτιδήποτε μπορεί να αποτελέσει περιγραφή με τη χρήση ενός συνδέσμου στις οντότητες που παρουσιάζονται σαν ιδιότητες ή τιμές.

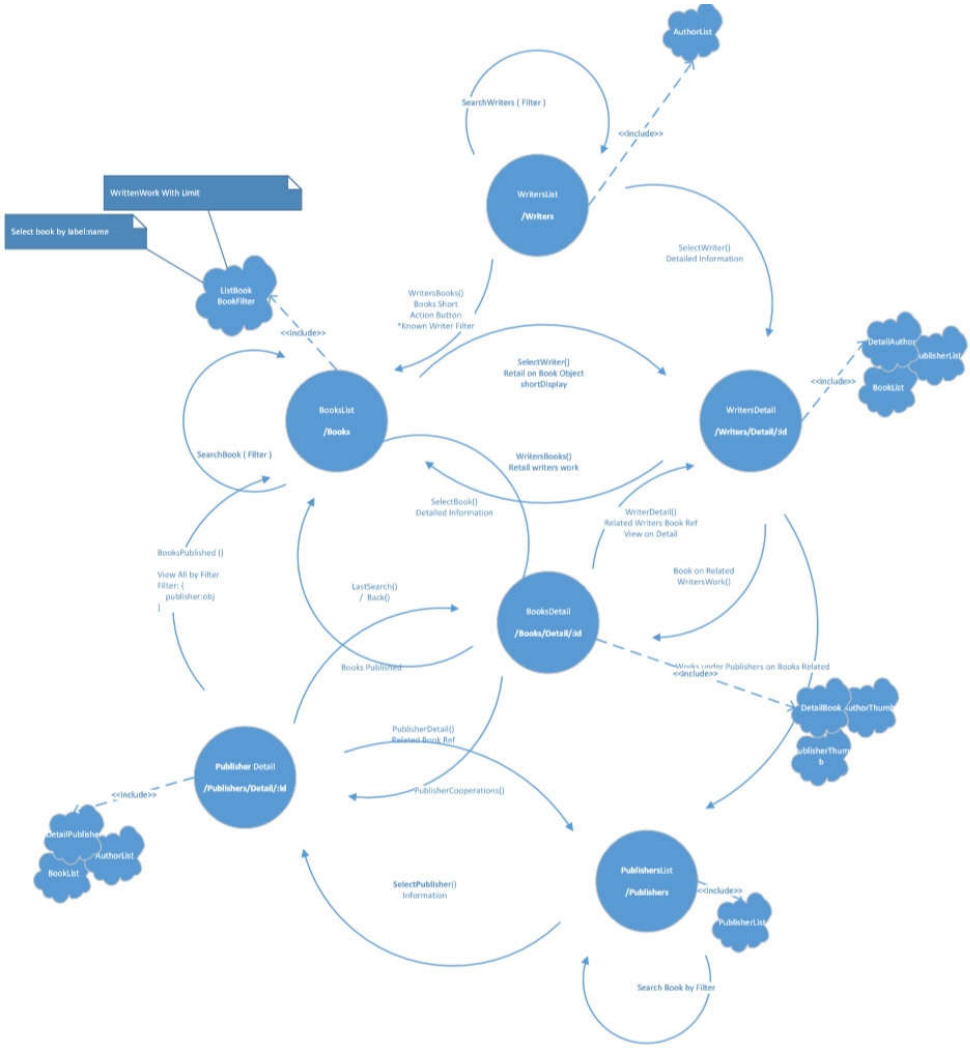
Μια εκτενή ανάλυση και παρουσίαση, μιλώντας για μια ιδιότητα που εκφράζεται με τη χρήση συνδέσμου στην λεπτομερή ανάλυση, μπορεί να υλοποιηθεί σαν ξεχωριστός component που με τη σειρά του δημιουργεί ένα ερώτημα περισυλλογής και ξεχωριστής αναπαράστασης. Η υλοποίηση μιας οντότητας που περιγράφεται από μια ιδιότητα διασυνδεδεμένων δεδομένων δεν πρέπει να ορίζεται καθολικά και μπορεί να διαφέρει σύμφωνα με τις ανάγκες της εφαρμογής μεταξύ των καταστάσεων. Για παράδειγμα ο Author μπορεί να παρουσιαστεί σε μορφή Author Detail σαν resource , σαν οντότητα στη συλλογή της λίστας των authors και σαν τιμή ενός συγκεκριμένου Property του βιβλίου. Με βάση αυτά μόλις παρουσιάσαμε μια ανάγκη ενός συνδέσμου με διαφορετικούς τρόπους αναπαράστασης δύναται να διαφέρει της επεξεργασίας και της παρουσίασής της. Η επίλυση του προβλήματος αυτού έχει τόσο να κάνει με το επίπεδο που παρουσιάζεται μια ιδιότητα και επιφέρει διαφορετικούς τρόπους όσον έχει και με το σχεδιασμό ή τους χειρισμούς της μέσα στην εφαρμογή.

Με το ζήτημα της αναπαράστασης και διαχείρισης των οντοτήτων θα παρουσιάσουμε ένα τρόπο προκαθορισμού της λειτουργίας σε σχέση με το επίπεδο της παρουσίασης αργότερα. Αλλά πριν τελειώσουμε με το διάγραμμα καταστάσεων να πούμε ότι οι σχετικές οντότητες που παρουσιάζονται στην εφαρμογή έχουν να κάνουν με events που διαφέρουν σε σχέση με τον τρόπο με τον οποίο παρουσιάζονται αυτές μέσα στην εφαρμογή και με τον τρόπο με τον οποίο υλοποιούνται από τα

templates σε κάθε κατάσταση συστήματος. Παρόλα αυτά το επίπεδο το οποίο χαρακτηρίζει την επεξεργασία της οντότητας μπορεί να καθοριστεί σε διάφορα στάδια πριν το binding της πληροφορίας και να παραμετροποιηθεί δυναμικά από configuration files που ορίζουν το πώς θα παρουσιάζεται σε κάθε επίπεδο.

Στο διάγραμμα αναπαράστασης καταστάσεων παρουσιάζονται με κυκλικές οντότητες οι καταστάσεις του συστήματος που αναφέραμε. Με τα βέλη οι πιθανές αλλαγές καταστάσεων από χειρισμούς του χρήστη και από events που περιγράφονται. Οι διακεκομμένες γραμμές υποδηλώνουν τη σχέση με τα αντικείμενα που στελεχώνουν την κάθε κατάσταση και με σύννεφα οι βασικοί components που ενδέχεται να τις πλαισιώσουν. Στο διάγραμμα δεν αναφέρουμε τους απλούς components οι οποίοι υλοποιούν το κάθε property αντικειμένου που περιγράφεται αλλά αυτούς που λογικά αποτελούν συνδυασμένη λογική και μπορούν να αναλυθούν ακόμα περισσότερο όπως θα δούμε στο κεφάλαιο του implementation αργότερα.

Σκοπός του γραφήματος είναι να περιγράψει τις βασικές καταστάσεις του συστήματος και τις πιθανές ανάγκες ομαδικής παρουσίασης από properties αντικειμένων που είναι ενδιαφέρον να παρουσιαστούν στον χρήστη σε κάθε κατάσταση και σχετίζονται με την οντότητα που αναπαρίσταται. Παρατηρώντας λοιπόν την σύνθεση γίνονται ορατές οι κοινές οντότητες που περιγράφονται και όπως είπαμε παραπάνω δύναται να διαφέρουν σε επίπεδο παρουσίασης ανάλογα με την κατάσταση που περιγράφεται. Για παράδειγμα η ιδιότητα του Author που βλέπουμε να εμφανίζεται στις καταστάσεις του WriterList σαν κύριο αντικείμενο περισυλλογής, στην κατάσταση του Writer Details που παρουσιάζεται σαν βασική οντότητα με λεπτομέρειες, στην κατάσταση του Publisher Details περιγράφοντας τους συνεργαζόμενους συγγραφείς με τον σχετικό εκδότη και στην κατάσταση του Book List σαν property της βασικής οντότητας Book που όπως θα δούμε χρησιμοποιείται σαν facet.



Διάγραμμα 5: UI state Diagram

Σχετικά με τα events τα οποία κυριαρχούν στο παραπάνω διάγραμμα υπάρχουν events τα οποία αντιστοιχούν στην ίδια κατάσταση και events τα οποία αλλάζουν την κατάσταση του συστήματος. Αλλάζοντας την κατάσταση του συστήματος οι βασικοί components που πλαισιώνουν την εκάστοτε κατάσταση αλλάζουν αλλά δύναται να παρουσιάσουν με τις ίδιες δομικές μονάδες το περιεχόμενο της πληροφορίας ως ένα επίπεδο. Αυτό που διαφέρει μεταξύ της αναπαράστασης της κάθε οντότητας που περιγράφεται είναι οι ιδιότητες που στελεχώνουν την πληροφορία αλλά αυτές δεν παύει να είναι properties των διασυνδεδεμένων δεδομένων που υλοποιούνται με τους δύο βασικούς τρόπους που αναφέρουμε ξανά και ξανά, δηλαδή ως literals ή ως IRLs.

Φυσικά και κάθε κατάσταση περιγράφει πολύ περισσότερες λειτουργίες που μπορεί να αποσκοπούν στις ανάγκες ενός χρήστη. Εδώ γίνεται μια αναφορά όσων από αυτές αποτελούν πηγαία ερωτήματα λογικής με σκοπό να γίνει κατανοητή η υποτυπώδη λειτουργία που χαρακτηρίζεται μεταξύ των αλλαγών των καταστάσεων του συστήματος που δύναται να προέλθουν από ενέργειες του χρήστη της εφαρμογής.

4.1.2 Triggered and Action Events

Οι διαπροσωπείες γενετικού τύπου που υλοποιούμε προσφέρουν στο χρήστη – χειριστή actions με τα οποία λογικές εκφράσεις παίρνουν μορφή υπολογιστικής διαχείρισης μέσω action events. Στις καταστάσεις του συστήματος που περιγράφονται παραπάνω γίνονται διακριτές συναρτήσεις που διαφέρουν από λογικής πλευράς της εφαρμογής αλλά σχετικά με τον τρόπο συγκομιδής των πληροφοριών όπως θα δούμε δεν διαφέρουν και τόσο.

Αναφορικά και για να αναλύσουμε τις καταστάσεις μια προς μια έχουμε:
Στη κατάσταση Booklist:

- Search Book with SearhTerm
- Search Book with Country
- Search Book with Language
- Search Book with Author
- Search Book with Category
- Search Book with other Property
- Select Book – View Book Details of current book
- Select Writer – View Writer Details of selected Writer
- Select Publisher – View Publisher Details of selected Publisher

Στην κατάσταση Writers List :

- Search Writer with Name
- Search Writer with birthdate
- Search Writer with Birth Place
- Search Writer with Country
- Search Writer with genre
- Search Writer with other Property
- Select Writer – View Writer Details
- Select Books – View Books of current Writer

Στην κατάσταση Publishers List :

- Search Publisher with Term
- Search Publisher with company
- Search Publisher with Products
- Select Publisher – view Publisher Details

Στην κατάσταση Book Detail:

- View Books on Category (literary Genre)
- View Writer Details
- View Publisher Details
- Add / Edit/ Delete Book’s Property (admin access)
- Search events, point of sale
- Buy online / Order
- Share / Search on social networks

Στην κατάσταση Writer Detail:

- View Books on Writer
- View Publisher Details
- Add / Edit/ Delete Writer's Property (admin access)
- Find Person on media
- Contact / Direct communication

Στην κατάσταση Publisher Detail:

- View Books on Publisher
- View Publishers on Industry
- View Writers Cooperated
- Add / Edit/ Delete Publisher's Property (admin access)
- See point of sales
- View other publishes
- Contact

Όλες οι παραπάνω συναρτήσεις μπορούν να γενικευθούν σε βασικούς τύπους ερωτημάτων που παράγονται όπως αναζήτηση πλήθους αποτελεσμάτων με παράμετρο ή παραμέτρους και λεπτομερής συλλογή πληροφορίας με συγκεκριμένο resource. Οι δύο αυτές κατηγορίες ερωτημάτων μπορούν να συντάξουν ερωτήματα SPARQL με αντίστοιχα φίλτρα και να δώσουν αποτελέσματα πληροφόρησης είτε παραμένοντας σε καταστάσεις ομαδικής αναζήτησης μιας οντότητας είτε περισυλλέγοντας τις λεπτομέρειες μιας πληροφορίας σχετικά με ένα resource. Ο διαχωρισμός αυτός που κάνουμε σχετικά με τις καταστάσεις λιστοειδής αναπαράστασης και λεπτομερής αναπαράστασης γίνεται γιατί όπως είπαμε θα παρουσιάσουμε δυο κύριες δυναμικές διαπροσωπείες οι οποίες θα στελεχώσουν το κύριο σώμα της εφαρμογής μας.

Η επιλογή αυτή σχετικά με την ομαδοποίηση και τη συγκέντρωση πληροφοριών πολλαπλών δεδομένων βάση φίλτρων μπορεί να υλοποιηθεί εκμεταλλευόμενοι τις οντότητες των δεδομένων και αναπαριστώντας τις ως φίλτρα αφού στελεχώνουν την πληροφορία που θέλουμε να συλλέξουμε. Η δε λεπτομερής αναπαράσταση υλοποιείται με βάση το γεγονός και την ανάγκη παρακολούθησης μιας συγκεκριμένης οντότητας που αποτελεί πηγαία μορφή πληροφόρησης και χρήζει άξια διαχείρισης. Και στις δυο καταστάσεις η κατηγοριοποίηση των λειτουργειών και τον ενεργειών που δύναται να χαρακτηρίσουν το σύστημά μας απλοποιούνται σε ανάγκες ανακατεύθυνσης στον γράφο των καταστάσεων του συστήματος και επεξεργασίας των δεδομένων που παρουσιάζονται στην εκάστοτε κατάσταση. Αυτές με κύριο κορμό την διαχείριση των δεδομένων αναλύονται σε διαδικασίες που υποστηρίζονται από CRUD operations και Interface presentation actions είτε αυτά πρόκειται να υλοποιούνται από διασυνδεδεμένα δεδομένα είτε από δεδομένα που προέρχονται από εξωτερικά συστήματα.

Events τύπου όπως search entities with filters κάνουν το σύστημα να παραμείνει στην κατάσταση στην οποία βρίσκεται πραγματοποιώντας μια ερώτηση σχετικά με τη διάθεση της ίδιας οντότητας με καινούργια φίλτρα. Άλλα events όπως select Writer() , select Book() όχι μόνο παροτρύνουν το σύστημα να αλλάξει κατάσταση αλλά να πραγματοποιήσει και ερώτηση περισυλλογής δεδομένων σε επίπεδα resource. Αυτό υλοποιείτε προγραμματιστικά με μια ανακατεύθυνση σε καινούργια κατάσταση συστήματος και πραγματοποίηση της εκάστοτε ερώτησης συγκομιδής πληροφοριών που θα συντάξουν το αντικείμενο που παρουσιάζεται.

Η σύνταξη του αντίστοιχου SPARQL ερωτήματος πραγματοποιείται από τα trigger events του UI είτε από events που ενεργοποιούνται αυτόματα από το bind δεδομένων σε συγκεκριμένους components. Για παράδειγμα μια κλήση σε καινούργια δεδομένα μπορεί να προκληθεί από ένα action του χρήστη κατά την επιλογή οντοτήτων στους components των facets που αλληλοεπιδρούν με τον component που παρουσιάζει την αναπαράσταση ενός collection ενώ μια άλλη να δημιουργήσει μια κλήση λεπτομερειών ενός resource όπως όταν αναπαριστούμε τα book details και έχουμε ένα ξεχωριστό component για να παρουσιάσουμε το resource του Author που καταφθάνει σε ένα URL σαν property της οντότητας book και είναι αναγκαία η περισυλλογή λεπτομερέστερης πληροφορίας.

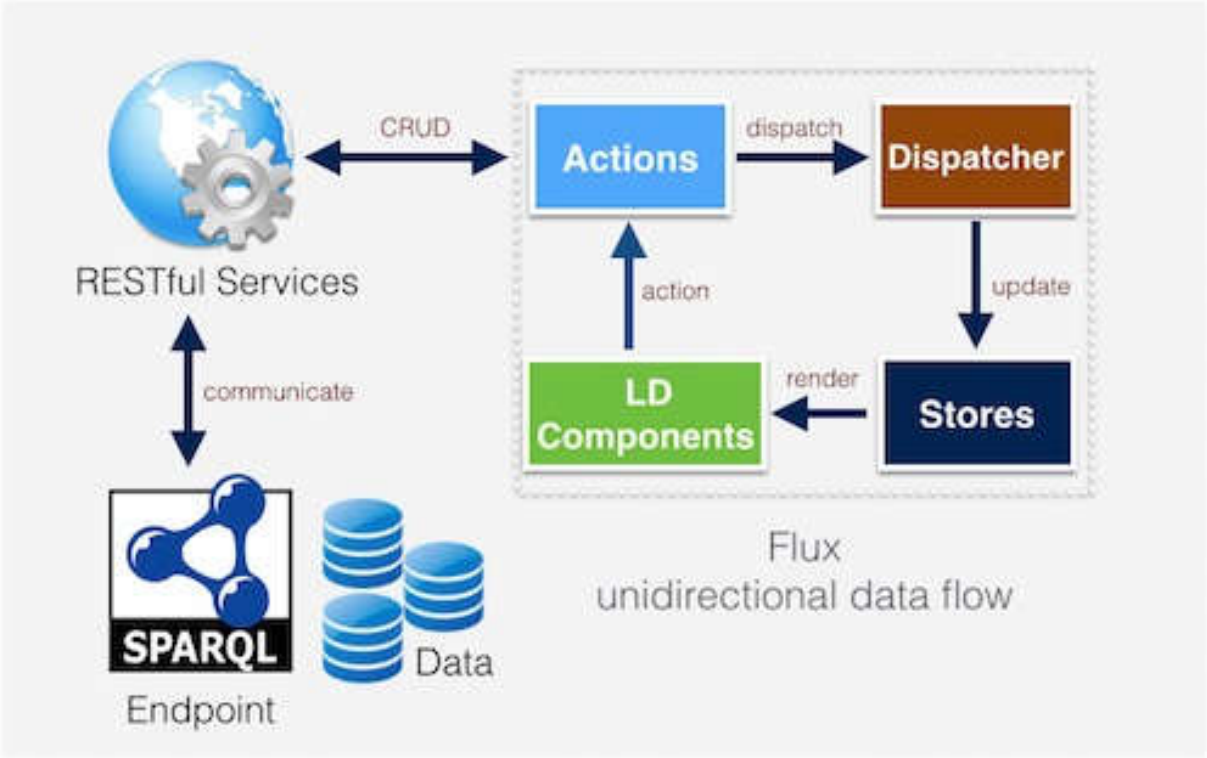
Η λειτουργικότητα της διαδικασίας σύνταξης, πραγματοποίηση ερώτησης, απόσπασης πληροφορίας και οπτικής προβολής στο τελικό template μπορεί και υλοποιείτε ξεχωριστά αφού μπορούμε να διαχωρίσουμε τη λειτουργία σε ξεχωριστούς components ανάλογα με την λογική. Στη κατασκευή και στη στελέχωση μιας κατάστασης του συστήματος σημασία δεν έχουν οι αλληλοεπιδράσεις όσο ο διαχωρισμός και ο ορισμός των events σε ανεξάρτητα κομμάτια που θα πλαισιώσουν τη διαπροσωπεία. Διαδικασίες επεξεργασίας των δεδομένων μπορούν να υλοποιηθούν από διαφορετικά δομικά στοιχεία που περιγράφουν το κάθε property του αντικειμένου που προβάλλεται και διαχειρίζεται με έναν ξεχωριστό τρόπο.

Με αυτόν τον τρόπο η λειτουργικότητα που επιθυμούμε να προσδώσουμε μέσω ενεργειών του χρήστη στην υλοποίησή μας καθίσταται ένα γεγονός κατασκευής ξεχωριστών components που μπορούν να συντάξουν την σελίδα που είναι κάθε φορά ορατή στον χρήστη. Κάθε ξεχωριστή δομική μονάδα που αποτελεί την ορατή διαπροσωπεία κάθε φορά έχει τους δικούς της χειρισμούς και ιδιότητες και από εκεί και πέρα όλα χρήζουν ανάγκες σύνταξης και παραμετροποίησης της σελίδας που παρουσιάζεται.

4.2 User Interface

Το τελικό στάδιο αναπαράστασης των δεδομένων σε μια διαπροσωπεία αλληλεπίδρασης με τον τελικό χρήστη είναι αυτό το οποίο κρίνει πολλές φορές το αν μια εφαρμογή είναι αντιπροσωπευτική των προσδοκιών της εφαρμογής. Πολλές υλοποιήσεις κρύβουν πολύπλοκες λογικές και λειτουργίες και δεν κεντρίζουν το ενδιαφέρον του τελικού χρήστη λόγω γραφικών.

Η υλοποίησή μας χρησιμοποιεί τα συστατικά του ReactJS του Facebook, την αρχιτεκτονική Flux, το πλαίσιο Fluxible[45] του Yahoo! για ισομορφικές εφαρμογές Web (δηλ. Το τρέξιμο του κώδικα εξαρτημάτων τόσο στον εξυπηρετητή όσο και στον πελάτη) και το πλαίσιο Semantic-UI[46] για ευέλικτα θέματα UI. Οι κύριοι λόγοι για τους οποίους επιλέξαμε τα στοιχεία React έναντι άλλων λύσεων Web Components (π.χ. Polymer, AngularJS, EmberJS κ.λπ.) ήταν η ωριμότητα και η συντηρησιμότητα της τεχνολογίας, η υποστήριξη πολλαπλών πλατφορμών, ο αριθμός των εργαλείων / συστατικών και την αποδοτικότητα της υποκείμενης εικονικής DOM προσέγγισης.



Διάγραμμα 6: Flux unidirectional Data Flow

Η εφαρμογή ακολουθεί την αρχιτεκτονική Flux, η οποία αποκλείει τη λειτουργία MVC (Μοντέλο-Προβολή-Ελεγκτής) υπέρ μιας μονοδιάστατης ροής δεδομένων. Όταν ένας χρήστης αλληλοεπιδρά με ένα στοιχείο React, το στοιχείο προωθεί μια ενέργεια μέσω κεντρικού αποστολέα, στα διάφορα stores που διατηρούν τα δεδομένα της εφαρμογής και την επιχειρηματική λογική και ενημερώνει όλα τα εξαρτήματα που επηρεάζονται. Η αλληλεπίδραση του στοιχείου με τα τελικά σημεία SPARQL για την ανάκτηση και την ενημέρωση των Συνδεδεμένων Δεδομένων πραγματοποιείται μέσω της κλήσης των RESTful υπηρεσιών σε ενέργειες.

Πριν προχωρήσουμε στην ανάλυση της υλοποίησης και στην περιγραφή της διαδικασίας του κύκλου ανάλυσης και σκόπιμο είναι να αναδείξουμε τις δυο βασικές διαπροσωπείες οι οποίες αποτελούν την κύρια διαχείριση και παρουσίαση των διασυνδεδεμένων δεδομένων. Με αυτόν τον τρόπο θα γίνει πιο κατανοητή η υλοποίηση μας, οι καταστάσεις του συστήματος σε επίπεδο παρουσίασης αλλά και παράλληλα θα δοθεί μια πρώτη εικόνα σε σχέση με όσα αναφέραμε παραπάνω για τις αλληλεξαρτήσεις αλλά και τα δομικά στοιχεία που θα αναλύσουμε.

Με τη βοήθεια των εικόνων στη συνέχεια θα αναλύσουμε τον κύκλο επεξεργασίας και τη ροή δεδομένων καθώς και τους components και τις συναρτήσεις που απαρτίζουν το κάθε στάδιο της flux υλοποίησης.

4.2.1 Αναζήτηση πλήθους δεδομένων - Browse Collection

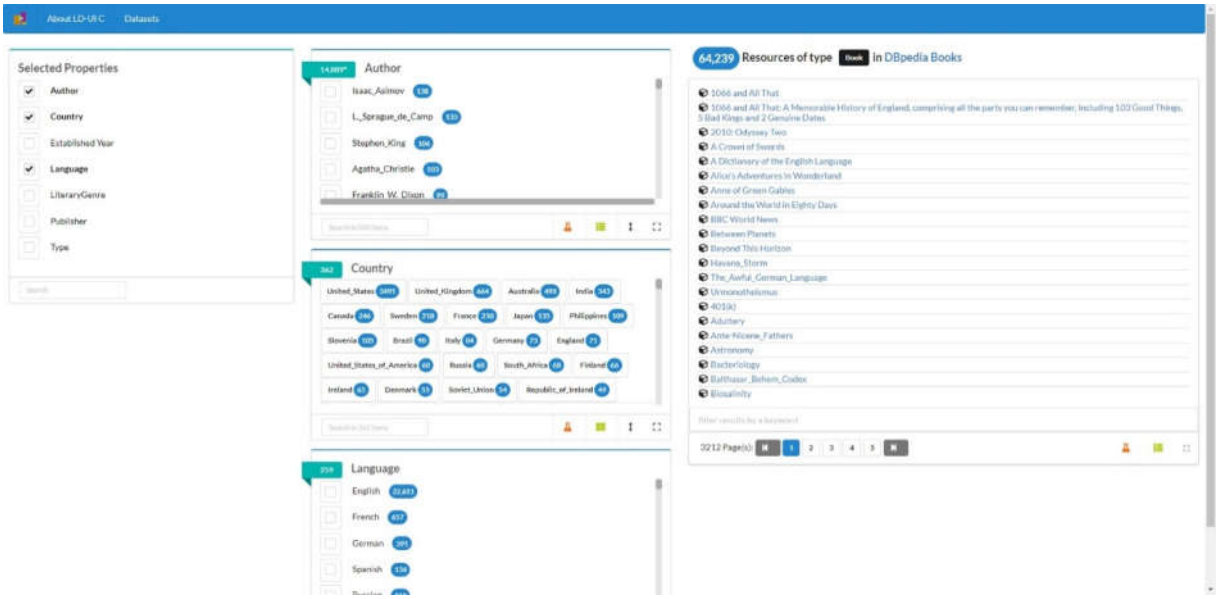
Στην ανάλυση που προηγήθηκε παρουσιάσαμε την ανάγκη για την ομαδική αναζήτηση και παρουσίαση κοινών δεδομένων ίδιου τύπου. Στην παρακάτω εικόνα φαίνεται να υλοποιείται μια διαπροσωπεία η οποία υλοποιεί αυτή την ανάγκη σχετικά με την οντότητα των συγγραμμάτων. Παρόμοια υλοποίηση

όμως χρησιμοποιείται και για τις οντότητες των Writers και Publishers. Η σχετική σελίδα του συστήματος παρουσιάζει τα δεδομένα της συλλογής που προέρχονται σχετικά με την αναζήτηση των συγγραμμάτων από το resource της DBPedia.

Αποτελείται από 3 βασικά μέρη που γίνονται διακριτά και στην εικόνα. Οι components που υλοποιούν αυτή την σελίδα είναι ξεχωριστά κομμάτια κώδικα που πλαισιώνουν την εφαρμογή με ξεχωριστά events και υλοποιούν το πρότυπο flux όπως περιγράψαμε παραπάνω. Αργότερα στην ανάλυση της υλοποίησης θα περιγράψουμε αναλυτικά τα συστατικά στοιχεία καθώς και τη ροή της πληροφορίας μέσω του flux μέσα στα στρώματα που διαχωρίζονται στην εφαρμογή μαζί με τα events που δημιουργούνται.

Σκοπός της αναφοράς του Browsing Collection εδώ είναι να αναλύσει την οθόνη που συλλέγει τα δεδομένα ίδιου τύπου και τα παρουσιάζει σαν επιλογές στον χρήστη. Στην οθόνη λοιπόν παρουσιάζονται τα τρία βασικά μέρη και γίνονται ορατά σαν στήλες. Στην αριστερότερη στήλη παρουσιάζονται τα properties της οντότητας των βιβλίων που χαρακτηρίζουν κάθε καταχώρηση και είναι κοινά για όλα τα συγγράμματα και γενικότερα για κάθε οντότητα συλλογής την οποία στην υλοποίησή μας ονομάζουμε Dataset. Αυτά τα properties είναι ο Author δηλαδή ο συγγραφέας του συγγράμματος, η χώρα που έχει εκδοθεί, η χρονολογία που δημοσιεύτηκε, η γλώσσα του συγγράμματος, το λογοτεχνικό είδος, ο εκδότης και ο τύπος.

Ο component παρουσιάζει τις ιδιότητες αυτές οι οποίες στελεχώνουν την κάθε οντότητα (dataset) και έχουν οριστεί στο configuration file από τον προγραμματιστή. Έτσι αποκρύπτει όσες ιδιότητες δεν είναι αναγκαίο να εμφανίζονται στο χρήστη φιλτράροντας οπτικά ιδιότητες του εκάστοτε αντικειμένου που χρήζουν αξιόλογες για την διαχείριση. Η υλοποίηση αυτού του δομικού στοιχείο έχει την ικανότητα να διαβάζει την οντότητα που επεξεργάζεται η σελίδα και να συλλέγει τις ιδιότητες του αντικειμένου έστω και αν υπάρχει σε μια μόνο καταχώρηση. Συγκεκριμένα έχει δοθεί μια λειτουργία η οποία διαβάζει το dataset που χαρακτηρίζει η κάθε οντότητα και εμφανίζει όλες τις ιδιότητες που θα ανακαλύψει ότι υπάρχουν στο εκάστοτε endpoint. Αυτή παίρνει την μορφή ενός enumerator ο οποίος κατά το rendering προσαρμόζεται έτσι ώστε να δείχνει όλες τις τιμές ή τις τιμές που έχουν οριστεί από τον προγραμματιστή. Η default λειτουργία του έχει να κάνει με αυτές τις ιδιότητες που είναι τροποποιημένες για να απαλείφονται πληροφορίες που δεν είναι χρήσιμες αλλά εύκολα τροποποιείται έτσι ώστε να διαβάζει όλα τα Properties μιας συλλογής που επεξεργάζεται κατά την διαδικασία του routing της εφαρμογής σχετικά με την επιλογή του dataset.

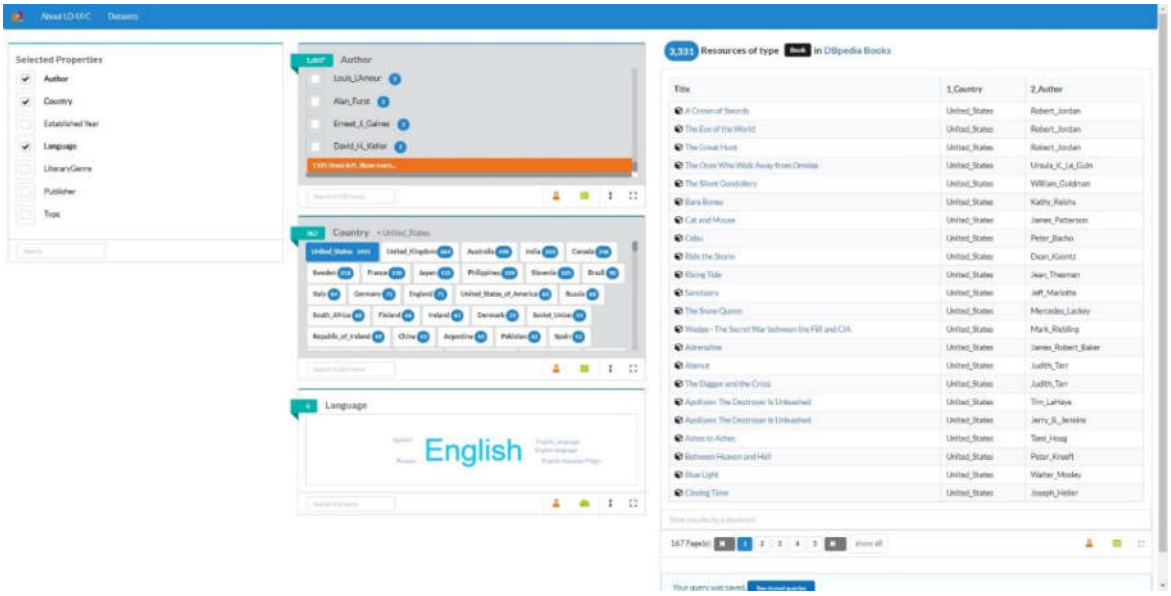


Εικόνα 20: UI Browse State

Στην κεντρική στήλη που εμφανίζεται στη συγκεκριμένη σελίδα παρουσιάζονται τα επιλεγμένα properties της αριστερής στήλης τα οποία εμφανίζουν τις διαφορετικές τιμές ολόκληρου του πλήθους των εγγραφών στον σύνδεσμο που πραγματοποιείται η ερώτηση. Αυτά όπως αναφέραμε παραπάνω ονομάζουμε facets. Σκοπός τους είναι να παρουσιάσουν τις ιδιότητες των οντοτήτων σε τιμές και προσφέρουν το πλήθος των καταχωρήσεων αυτών σε badges, ή μέσω του hover event σε άλλες επιλογές παρουσίασης, προσφέροντας στον τελικό χρήστη την ιδιότητα της καταμέτρησης του πλήθους των αποτελεσμάτων που περιέχουν αυτή την πληροφορία.

Δεξιότερα υπάρχει ένας component που παρουσιάζει τις καταχωρήσεις των δεδομένων σχετικά με μια ερώτηση με φίλτρα έχει αποδοθεί σαν SPARQL ερώτημα στο Virtuoso triple store. Σε αυτή την οντότητα που υλοποιείται τα δεδομένα παρουσιάζονται σελιδοποιημένα για τις ανάγκες του πλήθους και της χρονικής καθυστέρησης συγκομιδής των πληροφοριών. Ο συγκεκριμένος component αποτελεί το βασικό στοιχείο αναφοράς των συλλογών των αντικειμένων και όπως θα δούμε έχει αρκετές ιδιότητες.

Στην παρακάτω εικόνα είναι διακριτό ένα στιγμιότυπο το οποίο ο χρήστης έχει πραγματοποιήσει κάποια actions σχετικά με τους 3 βασικούς components που υλοποιεί η διεπαφή. Αναφορικά έχει επιλέξει να εμφανίζονται οι ιδιότητες του συντάκτη, της χώρας και της γλώσσας. Με αυτόν τον τρόπο έχει ενεργοποιήσει 3 facets αυτών των επιλογών στην κεντρική κολώνα. Λόγω του πλήθους των διαφορετικών αποτελεσμάτων που μπορεί να συλλέξει ένα ερώτημα οι facets υλοποιούνται με τη λογική του autocomplete.



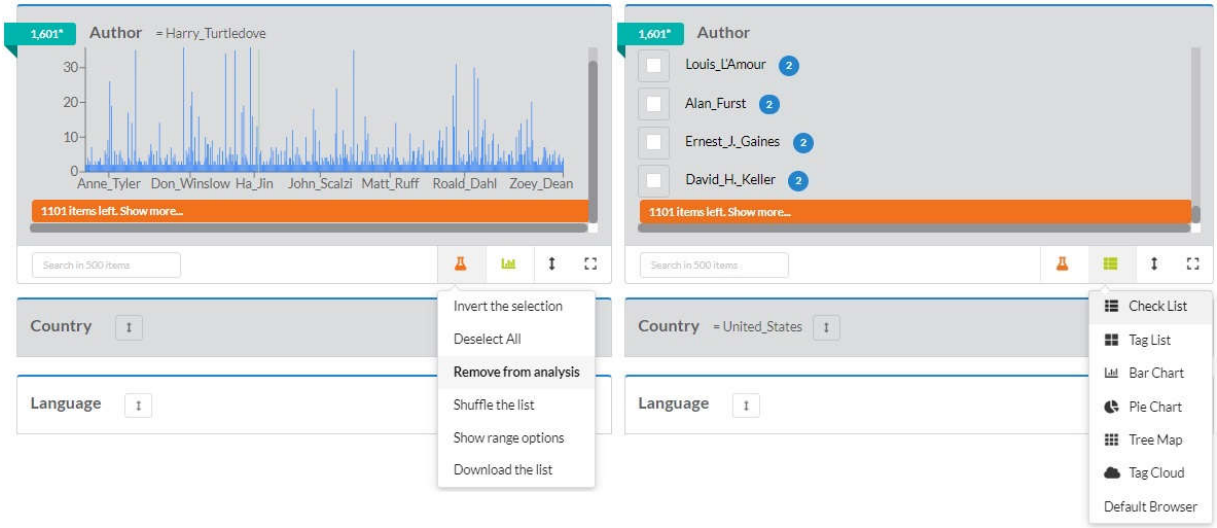
Εικόνα 21: UI Facets Options

Δηλαδή φέρνουν κάποια αποτελέσματα στον χρήστη τα οποία μπορεί να διαχειριστεί και παρουσιάζουν την επιλογή να φορτώσει περισσότερα όταν το dropdown του component που τα παρουσιάζει φτάσει στο τέλος του. Αυτό γίνεται διακριτό στο facet που υλοποιεί τον συντάκτη στην παραπάνω εικόνα και σαν ιδιότητα του SPARQL query που δημιουργείται υλοποιείται μέσω της Limit. Παρόλα αυτά υπάρχει το πλήθος των καταχωρήσεων της κεντρικής οντότητας που ερωτάται και φαίνεται σε ένα tag επάνω αριστερά στον κάθε component. Κάθε ιδιότητα του αντικειμένου που παρουσιάζεται στα facets μπορεί να προσαρτηθεί οπτικά στην αναπαράσταση των δεδομένων δεξιότερα και να δώσει μια έξτρα ανάλυση των δεδομένων σχετικά με την πληροφόρηση του χρήστη.

Μια άλλη ιδιότητα είναι αυτή της επιλογής του κάθε στοιχείου η οποία λειτουργεί σαν φίλτρο και ανατροφοδοτεί το σύστημα με ένα διαφορετικό ερώτημα περισυλλογής ανανεώνοντας το περιεχόμενο της σελίδας όχι εξολοκλήρου αλλά με events που ανανεώνουν το κάθε δομικό στοιχείο ξεχωριστά. Η επιλογή της οντότητας είναι με διαφορετικό χρωματισμό όπως αυτό φαίνεται στο facet που υλοποιεί την ιδιότητα Country που παρουσιάζεται στο κεντρικό facet με επιλεγμένη την τιμή «United-States».

Με το event που ενεργοποιείται το σύστημα κατά την επιλογή πραγματοποιεί μια διαδικασία κατασκευής ενός καινούργιου ερωτήματος και υποβολής του στο σύνδεσμο που έχει παραμετροποιηθεί να παρέχει τα δεδομένα. Έτσι συλλέγοντας τα καινούργια δεδομένα προκαλείται μια αλλαγή στο Store το οποίο υλοποιεί την σελίδα μετά από μια διαδικασία επεξεργασίας όπως θα αναλύσουμε παρακάτω. Κατά την αλλαγή της καταστάσεως του Store που πραγματοποιείται η σελίδα και οι components που την περιστοιχίζουν ανανεώνουν την κατάσταση των δεδομένων τους και εμφανίζουν τα καινούργια αποτελέσματα. Αυτή η διαδικασία του rendering είναι ανεξάρτητη για κάθε δομικό στοιχείο λειτουργώντας μόνο όπου αυτή είναι απαραίτητο να πραγματοποιηθεί και η τεχνολογία του React αποδεδειγμένα την υλοποιεί με τον πιο γρήγορο τρόπο. Κατά την αλλαγή των δεδομένων βάση φίλτρων δεν αλλάζουν μόνο οι καταχωρήσεις που παρουσιάζονται στην αναπαράσταση της κύριας πληροφορίας αλλά και τα δεδομένα που εμφανίζονται από τα υπόλοιπα facets αλλάζοντας τις καταχωρήσεις και τα αθροίσματα που φαίνονται.

Λόγω του πλήθους ομοειδών αποτελεσμάτων που συλλέγονται από ένα ερώτημα στα facets δίνονται κάποιες εξαιρετικές ιδιότητες που είναι κοινές και μπορούν να προσδώσουν κάποια εξτρά λειτουργικότητα στον χρήστη συντάσσοντας απλά components που υλοποιούν την οπτική τους ιδιότητα και παρουσιάζονται αλλά και ιδιότητες όπως της στελέχωσης πληροφορίας και διαχείρισης των δεδομένων. Στο footer του κάθε συστατικού στοιχείου υπάρχουν ένα input με το οποίο μπορούμε να εφαρμόσουμε μια λεκτική αναζήτηση σχετικά με τα δεδομένα που συλλέγουν, και τέσσερα action buttons από options που μπορούν να εφαρμοστούν σε κάθε τέτοιου τύπου δεδομένα.



Εικόνα 22: UI Facets Actions

Αναλυτικά αυτά είναι:

- Actions

Στα actions εμφανίζεται μια λίστα επιλογών στην οποία ο χειριστής μπορεί να αντιστρέψει την επιλογή που έχει κάνει σαν φίλτρα επιλέγοντας όλες τις άλλες εκτός από αυτές που έχει επιλέξει ήδη. Αυτή έχει εφαρμογή σαν φίλτρο not in στην ερώτηση που σχηματίζεται. Η ιδιότητα του καθαρισμού της επιλογής των φίλτρων που ήδη έχουν πραγματοποιηθεί. Η ιδιότητα Add/ Remove from Analysis που έχει σαν αποτέλεσμα να εμφανίζεται και να εξαφανίζεται το συγκεκριμένο property του αντικειμένου στη συλλογή της πληροφορίας δεξιότερα, όπως φαίνεται στον πίνακα των αποτελεσμάτων σε προηγούμενη εικόνα. Την ιδιότητα του shuffle της λίστας αποτελεσμάτων. Την ιδιότητα εμφάνισης κάποιων έξτρα φίλτρων σχετικά με τις τιμές που δείχνει και επηρεάζεται από τον τύπο των δεδομένων που εμφανίζει αλλά και από παραμετροποίηση του συγκεκριμένου facets από το αρχείο με τα configuration που θα αναλύσουμε στην συνέχεια. Τέλος υπάρχει η ιδιότητα συλλογής της λίστας τιμών που έχει συλλέξει και το κατέβασμά του σε αρχείο csv όπου ο χρήστης μπορεί να συλλέξει και να επεξεργαστεί με την εξαγωγή του.

- Change the Browser

Η λίστα επιλογών που εμφανίζεται σε αυτή την επιλογή είναι μια λίστα από components που μπορούν να εφαρμοστούν και να αλλάξουν τον τρόπο με τον οποίο παρουσιάζονται τα δεδομένα. Οι τιμές που εμφανίζονται όπως και στη παραπάνω εικόνα αντιστοιχούν σε διαφορετική αναπαράσταση όπως αυτή της εικόνας παρακάτω. Λόγω της παρόμοιας λειτουργίας που προσδίδει ο τύπος της πληροφορίας υλοποιήσαμε τα facets να αναπαρίστανται σε check lists, tag lists , bar charts, pie charts, tree map και tags cloud. Η επιλογή της προκαθορισμένης αναπαράστασης όπως φαίνεται σαν τελευταίο στοιχείο επιλογής διαβάζει από το configuration file των facets την τιμή που έχει οριστεί και αν δεν έχει τροποποιηθεί κάποια παράμετρος εμφανίζει τα δεδομένα σε μορφή checklist

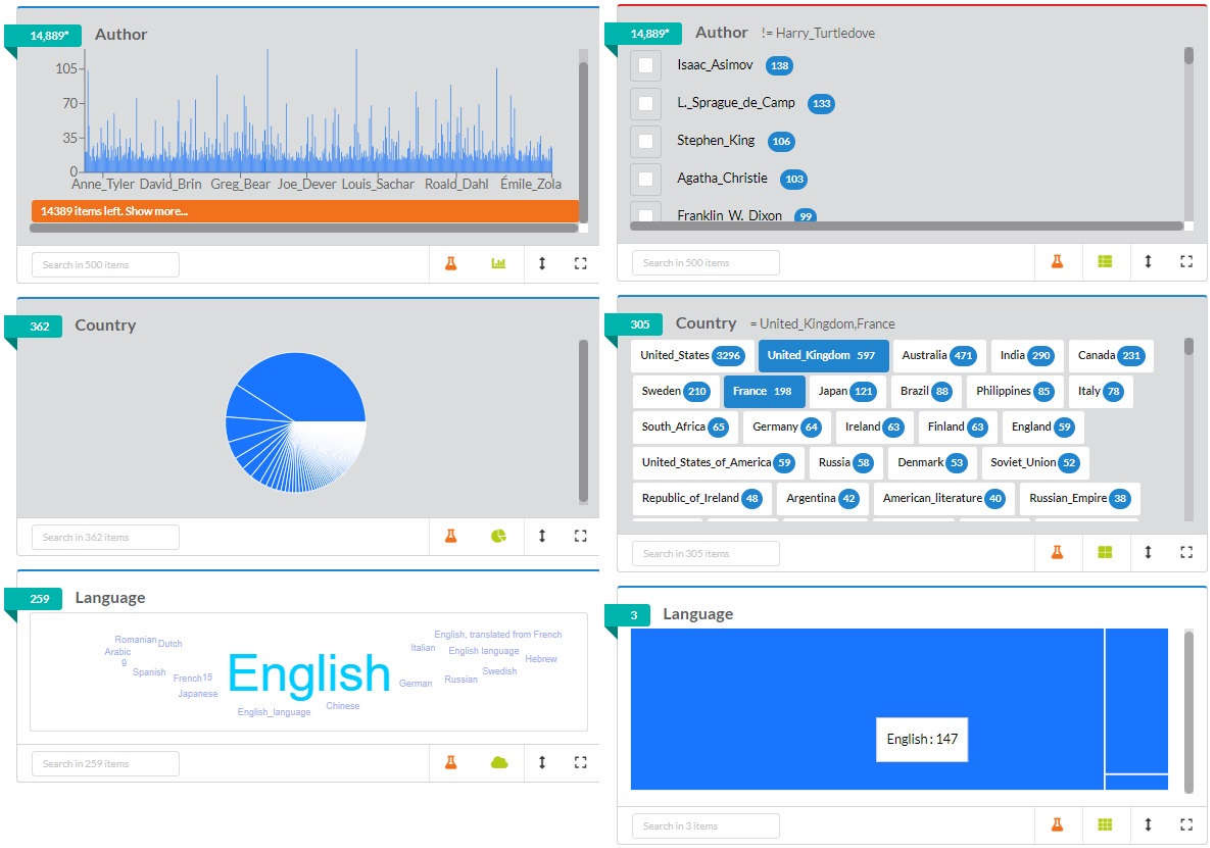
- Vertical Collapse

Το vertical Collapse είναι μια ιδιότητα που εφαρμόζεται στο συγκεκριμένο δομικό στοιχείο και το κάνει να φαίνεται μόνο με την επικεφαλίδα του και τα στοιχεία που είναι επιλεγμένα σαν φίλτρα όπως φαίνεται παραπάνω στα properties της χώρας και της γλώσσας.

- Expand Facet

Τέλος η ιδιότητα του expand κάνει τον συγκεκριμένο component να μεγαλώνει σε ανάλυση και να καλύπτει τις 2 αριστερότερες κολώνες που αναπαριστούν τα selected properties και τα facets.

Στην εικόνα που ακολουθεί γίνονται ορατοί οι τρόποι με τους οποίους μπορούμε να αναπαραστήσουμε τα δεδομένα που συλλέγονται και υλοποιούνται από τα facets. Επάνω αριστερά φένεται το Propery του author που αναπαρίσταται σε ένα bar chart και έχει γκρί φόντο γιατί έχει συμπεριθηφθεί στον πίνακα δεδομένων της αναζήτησης. Επίσης παρουσιάζεται με πορτοκαλί χρώμα η επιλογή Show More μαζί με το πλήθος των αποτελεσμάτων που δεν έχουν εμφανιστεί. Δεξιά αυτού είναι η ίδια οντότητα σε list item στην οποία έχει εφαρμοστεί η επιλογή του revert selection και η λειτουργία του facet υλοποιεί το φίλτρο της ερώτησης συλλογής με not in σε πολλαπλές τιμές.

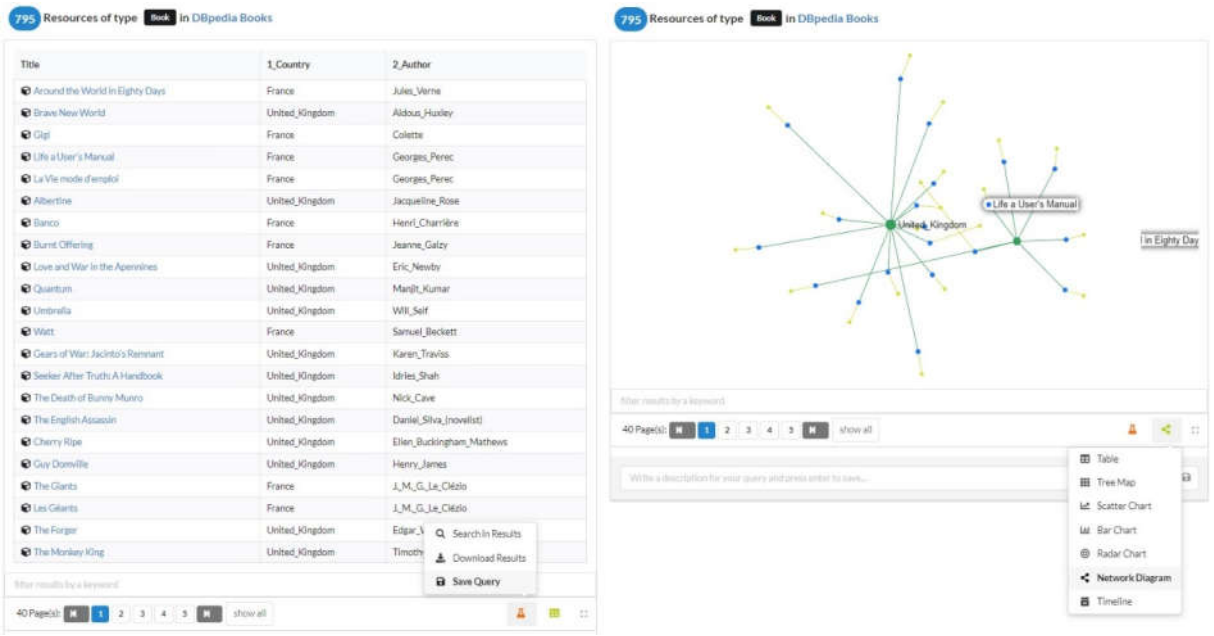


Εικόνα 23: UI Facets View Options

Στις επιλογές του country γίνονται ορατές αριστερά η επιλογή σε Pie chart όπου με ένα hover mouse event εμφανίζεται η τιμή και το πλήθος της συλλογής και δεξιότερα σε μορφή tag list. Τέλος στις κατώτερες επιλογές αυτές των γλωσσών είναι εμφανή τα στοιχεία σαν tag cloud και tree map.

Στις εικόνες όπως παρατηρείται παρακάτω παρουσιάζεται ο component που εφαρμόζονται τα στοιχεία της αναζήτησης και η παρουσίαση των δεδομένων. Αυτός είναι ο βασικός component που παρουσιάζει

τις κεντρικές οντότητες τις οποίες ορίζει το εκάστοτε dataset. Αριστερά στην εικόνα φαίνονται τα αποτελέσματα αναζήτησης σελιδοποιημένα σε μορφή πίνακα με στήλες το title του βιβλίου και ενσωματωμένα στις δεξιότερες η πληροφορία του country και Author που έχουν προσαρτηθεί από τις επιλογές των facets με το action “add to analysis”. Δεξιά στην εικόνα παρουσιάζεται ένα Network diagram το οποίο αποτελεί μια διαφορετικού είδους απεικόνισης της πληροφορίας.



Εικόνα 24: Browse Results Options

Και αυτά τα δεδομένα έχουν κοινές ιδιότητες όπως και τα facets. Αποτελούν καταχωρήσεις διασυνδεδεμένων δεδομένων οι οποίες έχουν προέλθει από ένα καθολικό ερώτημα το οποίο διαχειρίζεται συνολικά το view. Κάθε ιδιότητα που αναπαρίσταται είτε στον πίνακα είτε σε οποιαδήποτε μορφή παρουσίασης αποτελεί παράγωγο configuration επιλογής σε επίπεδο dataset. Υπάρχει δηλαδή η ιδιότητα να χαρακτηρίσουμε τις επικεφαλίδες του πίνακα με οποιοδήποτε λεκτικό το οποίο το αντιστοιχούμε σε οποιοδήποτε property της οντότητας της οποίας έχουμε κατ’ επιλογήν να εμφανίζεται στον ορισμό του κάθε dataset ξεχωριστά. Η σελιδοποίηση των δεδομένων γίνεται με σκοπό να αποφευχθεί ο όγκος της πληροφορίας για λόγους χρονικής καθυστέρησης αλλά και οπτικούς. Παρόλα αυτά υπάρχει η επιλογή να εμφανίσουμε όλα τα αποτελέσματα όπως φαίνεται από το κουμπί show all δίπλα από το pagination στο footer του component.

Στον συγκεκριμένο component εφαρμόζεται αναζήτηση λεκτικού στα ήδη προσαρτημένα δεδομένα και αφορά την κύρια κολώνα που αποτελεί την πηγαία πληροφόρηση. Υλοποιείται μέσω ενός input επάνω από το footer και προσφέρει την διαχείριση εικονικού φίλτρου στην ήδη υπάρχον συλλογή και σελίδα. Έχουν προστεθεί επίσης ιδιότητες επιπλέον διαχείρισης από action που υλοποιούν λειτουργίες όπως η αναζήτηση μέσο λεκτικού η οποία πραγματοποιεί ένα καινούργιο SPARQL ερώτημα κατά την πληκτρολόγηση του enter. Επίσης υπάρχει η δυνατότητα αποθήκευσης των δεδομένων που έχουν συλλεχθεί και η εξαγωγή τους σε αρχείο. Τέλος υπάρχει η δυνατότητα αποθήκευσης του συγκεκριμένου ερωτήματος η οποία για την ώρα αποθηκεύεται σε global μεταβλητή και είναι ενεργή όσο παραμένει το context της εφαρμογής αλλά μπορεί να υλοποιηθεί η αποθήκευσή της σε σύνδεσμο η σε αρχείο. Η δεύτερη λίστα επιλογών είναι αυτή του View και όπως και στους facet components

αλλάζει τον τρόπο με τον οποίο αναπαρίστανται τα δεδομένα. Όπως παρατηρείτε στην εικόνα παραπάνω με την εμφάνιση του network diagram που έχει επιλεχθεί. Τέλος υπάρχει το πλήκτρο toggle παρουσίασης που μεγεθύνει τον συγκεκριμένο component να εμφανίζεται σε ολόκληρο το παράθυρο.

Με βάση λοιπόν όσα αναφέραμε σε προηγούμενα κεφάλαια για τις ιδιότητες και τους κοινούς τύπου που διατίθενται τα δεδομένα είναι πολύ εύκολο όπως παρουσιάζεται να υλοποιηθούν δομικά στοιχεία της εφαρμογής που έχουν κοινές ιδιότητες όσον αναφορά την συμπεριφορά και την παρουσίαση. Η εφαρμογή μας είναι τροποποιημένη έτσι ώστε να παρουσιάζει και να επεξεργάζεται τα διασυνδεδεμένα δεδομένα με έναν τελείως γενετικό τρόπο εμφανίζοντας στον χρήστη τα resources που τροποποιούνται με χαρακτηριστικά που εμπλουτίζονται σε κάθε σημείο εμφάνισης. Η προκαθορισμένη λειτουργία για οντότητες και στοιχεία που δεν αναφέρονται στα configuration αρχεία διαχειρίζονται ως πεδία string τα οποία αναγράφουν τις τιμές των URLs που επιστρέφονται.

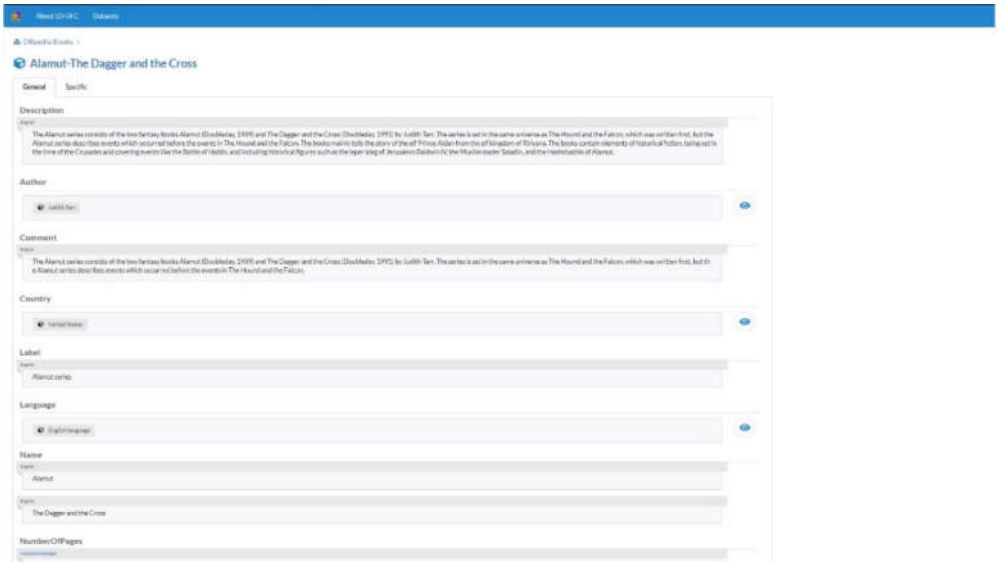
4.2.2 Λεπτομερής Αναπαράσταση - Entity Details

Κατά την επιλογή μιας καταχώρησης από τα δεδομένα που επιστρέφονται από μια αναζήτηση, όπως παρουσιάστηκε παραπάνω, μια οντότητα από αυτές που παρουσιάζει το dataset γίνεται το κύριο αντικείμενο που θέλει να παρατηρήσει ο χρήστης. Ένα event αλλαγής κατάστασης πραγματοποιείται και μια καινούργια σελίδα σε νέο tab ξεκινάει να λειτουργεί με το νέο context της εφαρμογής. Η λεπτομερής αναπαράσταση όπως αναφέρθηκε παραπάνω είναι αυτή που στελεχώνει την λεπτομερής παρουσίαση μιας συλλογής προκαθορισμένου δεδομένου σαν αποτέλεσμα ενός SPARQL ερωτήματος.

Σε αυτό το view υλοποιείται στον βασικό component κάθε ερώτηση η οποία περιγράφει ένα resource του triplestore. Στις βασικές καταστάσεις όπως περιγράψαμε παραπάνω υλοποιούνται αυτές των details των οντοτήτων που περιγράφει το σύστημά μας. Η κύρια παρουσίαση έχει να κάνει με τις παρουσιάσεις τον ιδιοτήτων της κάθε οντότητας. By default, η λειτουργία έχει να κάνει με τον διαχωρισμό της πληροφορίας όπως αυτή καταφθάνει από τον εκάστοτε σύνδεσμο. Τα properties των αντικειμένων μπορούν να οριστούν και να τροποποιηθούν μέσω του configuration file σχετικά με το ποιόν component θα χρησιμοποιήσουν για την αναπαράστασή τους και άλλες ιδιότητες όπως το λεκτικό που θα εμφανίζεται για το κάθε property, σε ποια κατηγορία θα ανήκει και με τι templates θα γίνεται διαδικασία της προβολής της αναζήτησης και της επεξεργασίας. Σε επόμενο κεφάλαιο θα παρουσιάσουμε την ανάλυση της παραμετροποίησης και θα γίνει πιο κατανοητή η λειτουργικότητα των views που περιγράφουμε καθώς και ο διαχωρισμός των συστατικών της σελίδας που αποτελεί την λεπτομερή αναπαράσταση. Η λειτουργία αυτή σχετικά με το πώς θα εμφανίζεται κάθε ιδιότητα στις σχετικές δομικές μονάδες που υλοποιούνται κάνει το κάθε στοιχείο του δεδομένου που παρέχεται μια ξεχωριστή λειτουργία και το παρουσιάζει με ένα δομικό στοιχείο που έχει τη δυνατότητα να προσαρτηθεί κάθε τύπου λογική που αποκρύπτεται μέσω ενός κουμπιού ή ενός πιο πλούσιου template που την παρουσιάζει. Για παράδειγμα η αγορά ενός συγγράμματος σε αυτό το σημείο μπορεί να υλοποιηθεί με ένα custom property στο λεξιλόγιο που ορίζουμε με και σε συνδυασμό με ένα εξωτερικό σύστημα να υλοποιήσει την αγοραπωλησία. Η επέκταση ενός ήδη υπάρχοντος συνδέσμου που αποτελεί μέρος της πληροφορίας θα μπορούσε να αντιστοιχιστεί με ένα δομικό στοιχείο το οποίο κατά τον σχηματισμό του να συνδέεται με ένα εξωτερικό σύστημα και παρουσιάζει τα σημεία πώλησης ή διάθεσης. Νέες ιδιότητες σχετικά με τις οντότητες που παρουσιάζονται στη λεπτομερή αναπαράσταση μπορούν να προσαρτηθούν από ένα χειριστή admin. Με αυτόν τον τρόπο και με τις ιδιότητες των δομικών στοιχείων να επεξεργάζονται events κατά τον σχηματισμό της πληροφορίας κάθε λογική υπηρεσία που επιθυμεί ο διαχειριστής να προσαρτήσει στην εφαρμογή καθίσταται εφικτή με το σχηματισμό του αντίστοιχου δομικού στοιχείου που έχει αντιστοιχιστεί στο επίπεδο configuration.

Ένα resource αποτελείται από τα properties που το απαρτίζουν και τιμές οι οποίες καταφθάνουν σε λεκτικά και συνδέσμους όπως περιγράψαμε. Η ιδιομορφία της κάθε τιμής που επιστρέφεται με ένα ερώτημα μας επιτρέπει να διαχειριστούμε τα δεδομένα με έναν κοινό τρόπο και να προσδώσουμε ιδιότητες που μπορούν να εφαρμοστούν από κοινού. Η λεπτομερής αναπαράσταση προσφέρει μια generic προσέγγιση για κάθε δεδομένο συνδέσμου όσον αναφορά την παρουσίαση και την επεξεργασία. Αυτό που διαφέρει είναι το τι αντιπροσωπεύει η κάθε τιμή και πως αυτή μπορεί να παρουσιαστεί στον χρήστη. Έτσι βασικό στοιχείο στο πρόγραμμά μας είναι να μπορούμε να διαχειριστούμε αλλά και να παρουσιάσουμε κάθε ιδιότητα με ένα ξεχωριστό component ώστε οι τιμές που παρουσιάζονται να πάρουν μια μορφή πιο φιλική ως προς τον χρήστη.

Στην εικόνα που ακολουθεί αναλύεται ένα σύγγραμμα που έχει προέλθει από επιλογή στην σελίδα της ομαδικής αναζήτησης που περιγράψαμε προηγουμένως. Κατά την επιλογή μιας καταχώρησης ένα event αναζήτησης λεπτομέρειας ενεργοποιείται και σχηματίζεται μια ερώτηση που συνδέεται με το end point που έχει οριστεί στο σύστημά μας για την συγκέντρωση της πληροφορίας του resource. Στο καινούργιο tab που ενεργοποιείται η κατάσταση του συστήματος μεταβάλλεται και αποδίδεται το ερώτημα περισυλλογής σε μια άλλη οθόνη, αυτή η οποία περιγράφει το κάθε property καθώς και την τιμή του. Εύκολα όλες οι τιμές σε αυτή την οθόνη μπορούν να αναπαρασταθούν με τον ίδιο τρόπο αφού είτε πρόκειται για URIs, είτε για λεκτικά μπορούν να σχηματίσουν την πληροφορία με ένα component.



Εικόνα 25: Resources Detail View

Στο view μπορούμε να παρατηρήσουμε την ύπαρξη των tabs. Αυτή υποδηλώνει την κατηγορία την οποία έχουμε ορίσει από το configuration file για την κάθε οντότητα να παρουσιάζεται. Σκοπός της δήλωσης κατηγορίας είναι να αποκρύπτει τις οντότητες που δεν επιθυμεί ο προγραμματιστής να εμφανίζονται στον χρήστη. Η προκαθορισμένη λειτουργία αναιρεί τα tabs και σχηματίζει σε ολόκληρη την σελίδα τις ιδιότητες του αντικειμένου τη μια κάτω από την άλλη.

Πολλές από αυτές τις ιδιότητες διαφέρουν μεταξύ τους ως προς τον τύπο της τιμής την οποία παρουσιάζουν. Κάθε ιδιότητα αναγνωρίζεται από το σύστημα έτσι ώστε όταν περιέχει λεκτικό να σχηματίζεται στο Interface με έναν component που περιγράφει την ιδιότητα με μια ταμπέλα και την

τιμή σε ένα πλαίσιο τιμών. Τέτοιου τύπου ιδιότητες γίνονται διακριτές στην παραπάνω εικόνα όπως το Description , Comment, Label, Name. Όπως είναι φυσικό για κάθε ιδιότητα μπορεί να υπάρχει πλήθος τιμών καθώς ένα αντικείμενο μπορεί να περιέχει παραπάνω από μια φορές την ίδια ιδιότητα με διαφορετικές τιμές. Αυτές παρουσιάζονται η μια κάτω από την άλλη με την ίδια επικεφαλίδα όπως για παράδειγμα αυτή του Label. Διακριτή επίσης γίνεται και η χρήση ενός άλλου component που περιγράφει ιδιότητες όταν αυτές αντιστοιχούν σε τιμές URIs. Αυτές οι ιδιότητες μπορεί κατά την συλλογή τους να περιέχουν περεταίρω πληροφορία και να υπάρχει η ανάγκη για τη συγκομιδή της. Αυτού του είδους οι ιδιότητες περιγράφονται στα διασυνδεδεμένα δεδομένα όπως είπαμε σαν σύνδεσμοι και εμφανίζονται σαν αντικείμενα. Τέτοιες πληροφορίες στην παραπάνω εικόνα αποτελούν το Author, Country και Language.

Κατά την επιλογή τους από τον χειριστή το σύστημα αλλάζει το περιεχόμενο της σελίδας σε μια αντίστοιχη κατάσταση λεπτομέρειας που περιγράφει τον παραπάνω σύνδεσμο. Αυτό υποδηλώνει την κοινή επεξεργασία τύπου δεδομένων ομοειδών στοιχείων και τη λειτουργικότητα που έχει προσφέρει η εφαρμογή μας σαν με ένα κοινό τρόπο διαχείρισης κάθε στοιχείου που αποτελεί μια δομή πληροφορίας. Με αυτή την ενέργεια ένας νέος κύκλος περισυλλογής, επεξεργασίας αποτελεσμάτων και οπτικής αναπαράστασης ξεκινά και σαν αποτέλεσμα μια ανακατεύθυνση σε μια λεπτομερής αναπαράσταση πραγματοποιείται.

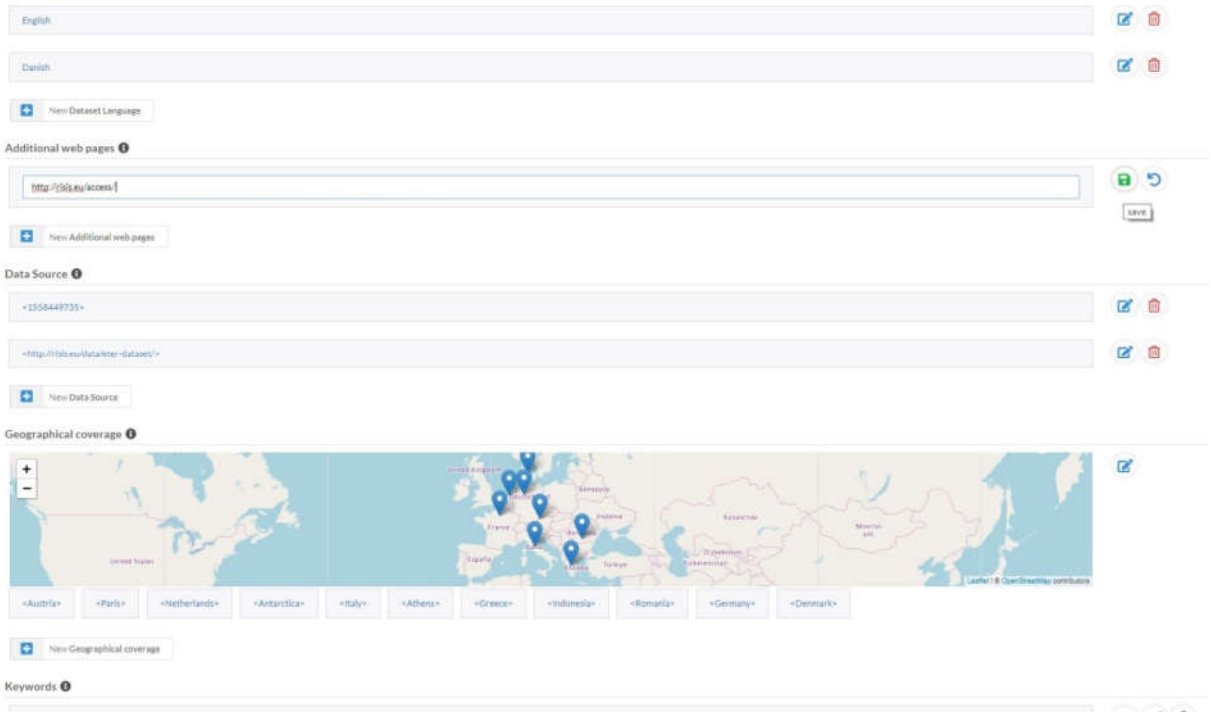
Για τις ανάγκες παρουσίασης και διευκόλυνσης του χρήστη ώστε σε κάθε παρουσίαση εμφολευμένης πληροφορίας να μην ανακατευθύνεται και να χάνει τη συγκεκριμένη παρουσίαση έχει υλοποιηθεί μια διαφορετική λειτουργεία. Στις ιδιότητες που αποτελούν αντικείμενα δεδομένων και περιγράφονται σαν σύνδεσμοι εμφανίζεται στα δεξιά ένα μπλε πλήκτρο με την ετικέτα «Show details». Αυτού η λειτουργία είναι toggle και με την ενεργοποίησή του ένας κύκλος συγκομιδής πληροφορίας όπως οι παραπάνω ενεργοποιείται με αποτέλεσμα να εμφανίζονται οι λεπτομέρειες του αντικειμένου που προέκυψε από την συλλογή σε ένα collapsible panel όπως φαίνεται στη παρακάτω εικόνα.

Με αυτή την ιδιότητα της γενικές επεξεργασίας ανοικτών δεδομένων κάθε τύπου πληροφορία που μπορεί να επιστραφεί από ένα σύστημα σε μορφή JSON-LD και να αποτελέσει στοιχείο πληροφόρησης ή επεξεργασίας μπορεί εύκολα να προσαρτηθεί και να αναγνωριστεί από την εφαρμογή. Η λειτουργία αυτή έχει να κάνει με την προκαθορισμένη λειτουργία μιας άγνωστης από το σύστημα ιδιότητας που αντιμετωπίζεται με τον ίδιο τρόπο κάθε φορά. Ο χειριστής της εφαρμογής μπορεί να αλλάξει τον τρόπο και το δομικό στοιχείο που παρουσιάζεται κάθε λεπτομέρεια της πληροφορίας και να χρησιμοποιήσει τις ενέργειες που είναι ήδη υλοποιημένες καθώς αποτελούν ξεχωριστό κομμάτι υλοποίησης.

Εικόνα 26: Resource Show More Functionality

Το Panel αυτό παρουσιάζει σαν επικεφαλίδα το αντικείμενο που περιεγράφηκε και στο κορμό του οι νέες οντότητες που συγκεντρώθηκαν. Αμέσως γίνεται ορατό σε αυτή την περίπτωση η επαναχρησιμοποίηση υλοποιημένων components αφού για τις ανάγκες παρουσίασης χρησιμοποιείται ο ίδιος component ο οποίος χρησιμοποιείται και για την παρουσίαση στα tabs και χρησιμοποιεί τις επικεφαλίδες σαν sections και τις τιμές σε εμφολευμένους components που περιγράφουν την κάθε οντότητα όπως έχει σεταριστεί.

Για τις ανάγκες της επεξεργασίας των δεδομένων δημιουργήσαμε ένα dataset με ιδιότητες διασυνδεδεμένων δεδομένων που επιτρέπει την προσθήκη νέων τιμών μιας ιδιότητας και την επεξεργασία αυτών. Για να κάνουμε χρήση στις ιδιότητες εισαγωγής επεξεργασίας και διαγραφής οι ιδιότητες του dataset στο configuration file πρέπει να οριστούν σαν Editable και να καθοριστεί ο component που θα χρησιμοποιείται ώστε αυτό να γίνει εφικτό. Στην παρακάτω εικόνα γίνεται ορατό αυτό το dataset καθώς και οι επιλογές που ενεργοποιούνται στη σελίδα λεπτομερειών. Σε κάθε group ιδιοτήτων εμφανίζεται η επιλογή εισαγωγής νέας καταχώρησης με ένα πλήκτρο με την ένδειξη «+» και την ονομασία New “Property Name” στο τέλος κάθε section. Στα δεξιά κάθε ιδιότητας γίνονται εμφανή τα πλήκτρα του Edit και Delete. Κατά την ενεργοποίηση της επεξεργασίας μιας πληροφορίας δεδομένων ο component που παρουσιάζει την ιδιότητα αλλάζει και τον διαδέχεται αυτός της επεξεργασίας που είναι πολλές φορές ίδιος με αυτόν της εισαγωγής. Στη διάρκεια επεξεργασίας μιας καταχώρησης τα πλήκτρα δεξιότερα αλλάζουν και αντικαθίστανται με δύο της αποθήκευσης και της επιστροφής.



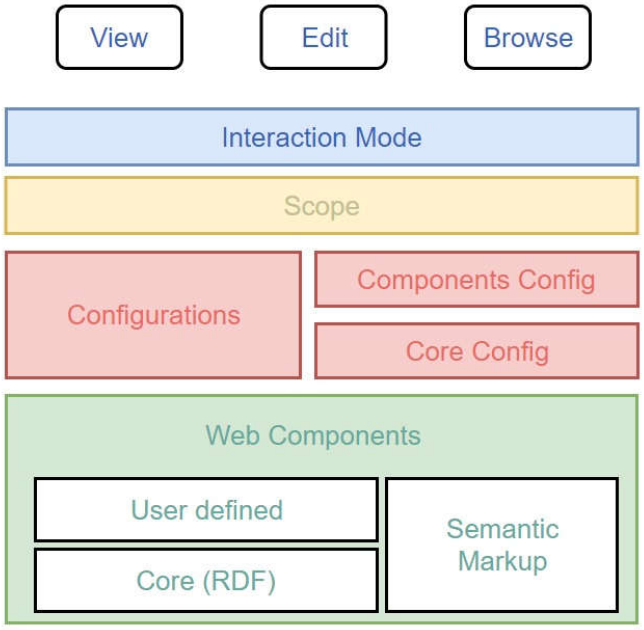
Εικόνα 27: Resources View Edit Delete Options

Κατά τις τρείς αυτές λειτουργίες της εισαγωγής, επεξεργασίας μετά την αποθήκευση και διαγραφής αντίστοιχα SPARQL ερωτήματα δημιουργούνται στο σύνδεσμο που έχει οριστεί και υλοποιούνται με τις εντολές Modify, Insert, Delete όπως ορίζει η σύνταξη της γλώσσας. Με τον τρόπο που δημιουργείται η διεπαφή και διαχωρίζεται σε ξεχωριστά sections για τις ανάγκες της κάθε οντότητας μας δίνεται η δυνατότητα να ορίσουμε για κάθε ιδιότητα επεξεργασίας και προβολής ένα ξεχωριστό component που τις υλοποιεί και χρησιμοποιείται. Έτσι για την δυναμική διαχείριση ομοειδών στοιχείων έχουμε υλοποιήσει components όπως αναφέρθηκαν παραπάνω για την παρουσίαση λεκτικών και συνδέσμων και για την διαδικασία της εισαγωγής και της επεξεργασίας inputs στα οποία ο χρήστης μπορεί να επεξεργαστεί την ιδιότητα είτε αυτή είναι λεκτικό είτε σύνδεσμος. Στην εικόνα παραπάνω μπορούμε να παρατηρήσουμε το περιεχόμενο του Geographical coverage το οποίο για τις ανάγκες παρουσίασης έχει παραμετροποιηθεί έτσι ώστε κατά την προβολή να μεταβιβάζει τα δεδομένα σε ένα component που τα αναγνωρίζει σαν συντεταγμένες και να τις παρουσιάζει σε ένα χάρτη.

Με την ίδια λογική και κάθε ιδιότητα δομείται σαν ξεχωριστό χαρακτηριστικό και αποκτά με το δικό της template παρουσίαση και επεξεργασία. Σε αυτό το σημείο βλέπουμε ότι αν και κάθε ιδιότητα των διασυνδεδεμένων δεδομένων αποτυπώνεται με κοινό τρόπο μπορεί να αποκτήσει δικά της χαρακτηριστικά και ιδιότητες δίνοντας στον χρήστη μια άλλη διάσταση σχετικά με το τι εκφράζει. Για τις διαδικασίες εισαγωγής νέων στοιχείων σε μια οντότητα ο εξειδικευμένος χρήστης ή προγραμματιστής το μόνο που έχει να κάνει είναι να ορίσει την νέα ιδιότητα και να την αντιστοιχίσει με τα δομικά στοιχεία που θα την παρουσιάζουν και θα την επεξεργάζονται και ίσως να δημιουργήσει νέα στοιχεία σε επίπεδο object όπως θα δούμε παρακάτω.

4.3 Χαρακτηρισμός Ιδιοτήτων και δεδομένων

Προκειμένου να εξορθολογιστεί η διαδικασία ανάπτυξης UI σε LDAs, προτείνουμε μια αρχιτεκτονική των προσαρμοστικών Web components για διασυνδεδεμένα δεδομένα που εμπλουτίζονται από το μοντέλο δεδομένων RDF. Η προτεινόμενη αρχιτεκτονική αντιμετωπίζει την επαναχρησιμοποίηση και την ευελιξία του LDA με την ενσωμάτωση Web components βασισμένων στο RDF. Αυτοί όπως αναλύσαμε έχουμε σκοπό να περιγράφουν και να διαχειρίζονται τα δεδομένα ενός πόρου και τις ιδιότητες των αντικειμένων με τέτοιο τρόπο ώστε να δρουν σαν ανεξάρτητες μονάδες.



Εικόνα 28: UI Representation Framework

Στο τελικό επίπεδο βρίσκεται ο χρήστης και η αλληλεπίδρασή του με το περιβάλλον της εφαρμογής. Σε αυτόν προσφέρονται οι ιδιότητες View, Edit, Browse. Πίσω από την τελική αναπαράσταση κρύβεται ένα στρώμα από πεδία δράσης και παραμετροποιήσεις από βασικές οντότητες οι οποίες πλαισιώνουν την εφαρμογή και τροποποιούνται να λειτουργούν με διαφορετικό τρόπο για την εκάστοτε διαπροσωπεία που σχηματίζεται και απαρτίζει την τελική εφαρμογή που διατίθεται στο χρήστη.

Τα πεδία δράσης δύναται να αλλάζουν από σημείο σε σημείο στην εφαρμογή και ανάλογα με τα δεδομένα τα οποία παρουσιάζουν. Για αυτόν τον λόγω η συνεργασία τους με το στρώμα παραμετροποίησης προσδίδει στην εφαρμογή μια λειτουργικότητα κάθε φορά που όχι μόνο παρουσιάζει ξεχωριστά αλλά και περιορίζει λειτουργίες όπου αυτές δεν είναι χρήσιμες ή δεν πρέπει να παρουσιάζονται στον χρήστη. Εξιδανικεύεται έτσι μια λειτουργία κάθε φορά ώστε να μπορεί να ελεγχθεί από τον προγραμματιστή το εύρος των δεδομένων επεξεργασίας.

Στο επίπεδο των παραμετροποιήσεων υπάρχουν δύο κύριες ενότητες οι οποίες κρίνεται αναγκαίο να επεξεργαστούν. Τα βασικά configuration που αποτελούν και τη βασική συμπεριφορά του συστήματος. Αυτά για παράδειγμα είναι παραμετροποιήσεις καταστάσεων του συστήματος (routes), διασύνδεση με εξωτερικά συστήματα και στοιχεία που είναι απαραίτητα για την λειτουργεία της εφαρμογής, όπως ο

σύνδεσμος που μπορεί να αποθηκεύονται τα προσωρινά triplestores, οι παραμετροποιήσεις του χρήστη, ο σύνδεσμος για το export των αρχείων, τα εξωτερικά συστήματα και η διασύνδεσή τους κ.α. Επίσης σε αυτό το επίπεδο παραμετροποίησης βρίσκονται οι συμπεριφορές των δομικών στοιχείων που χαρακτηρίζουν την συμπεριφορά τους και τον τρόπο με τον οποίο επεξεργάζονται τα δεδομένα και συνθέτουν τους ανεξάρτητους components που συνθέτουν την παρουσίαση από δομικά στοιχεία που διαχειρίζονται τα διασυνδεδεμένα δεδομένα αλλά και άλλου τύπου δεδομένα που προέρχονται από απομακρυσμένες πηγές.

Τα χαρακτηριστικά των διασυνδεδεμένων δεδομένων μπορεί να αλλάζουν σε κάθε view και να παρουσιάζονται με διαφορετικά templates που εξυπηρετούν τις ιδιότητες που θέλουμε να προσδώσουμε σε κάθε χρήση ανάλογα με την εκάστοτε υλοποίηση. Η δόμηση κάθε διαπροσωπείας αποτελείται από μικρότερα μέρη τα οποία μπορεί να εκφράζουν τον ίδιο τύπο δεδομένων. Σε πολλά σημεία λοιπόν δημιουργείται η ανάγκη για επαναχρησιμοποίηση των components που έχουμε υλοποιήσει. Σε άλλα σημεία του προγράμματος η προκαθορισμένη συμπεριφορά μπορεί να χρήζει ανάξια επεξεργασίας η να μην εξυπηρετεί τις ανάγκες της εφαρμογής. Για παράδειγμα αν όλα τα στοιχεία που χρησιμοποιούνται και παρουσιάζονται από templates που τα εκφράζουν ως λεκτικά, η τιμή μιας ημερομηνίας δεν θα ήταν εύκολο να διαχειριστεί.

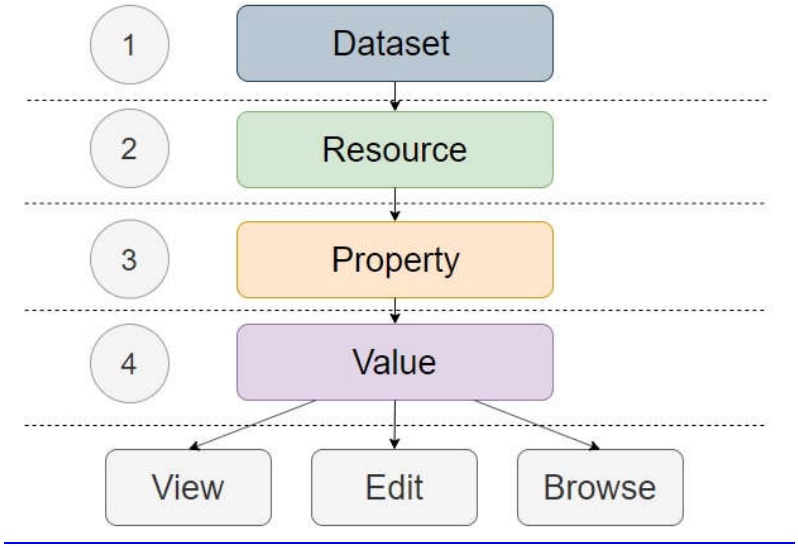
Αμέσως κρίνεται αναγκαία η προσαρμογή των ιδιοτήτων που χαρακτηρίζουν μια μορφή που είναι ευρέως γνωστή και έχουν υλοποιηθεί από προγραμματιστές components που τις εκφράζουν ευρέως και έχουν ήδη χαρακτηρίσει στις περισσότερες διεπαφές στο web ως προς την παρουσίαση και το χειρισμό τους. Κοινά παραδείγματα αναφοράς είναι αυτά των γεωγραφικών δεδομένων, της ημερομηνίας, αριθμητικές τιμές, δεδομένα θερμοκρασίας κλπ. Τα στοιχεία αυτά όπως είπαμε μπορούν να εφαρμοστούν σε διάφορα σημεία της εφαρμογής και χρήζει αναγκαία η παραμετροποίησή τους σε διαφορετικά επίπεδα αν αυτά τα δομικά στοιχεία θέλουμε να έχουν μια γενικότερη διαχείριση. Για αυτόν τον λόγω εφαρμόσαμε ένα διάγραμμα ροής των δεδομένων στην εφαρμογή μας έτσι ώστε να δίνεται η δυνατότητα κάθε ιδιότητα να παραμετροποιείται στον τρόπο χρήσης της σε κάθε επίπεδο που χρησιμοποιείται.

Από κοινού τα δεδομένα σε κάθε στρώμα επεξεργασίας έχουν κοινούς web components που χρησιμοποιούνται και τα επεξεργάζονται και αυτό δίνει μια δυναμική ιδιότητα στην εφαρμογή μας. Η εφαρμογή είναι έτσι κατασκευασμένη ώστε αν δε γνωρίζει τον τύπο δεδομένων ή ιδιότητας να δρα και να συμπεριφέρεται με έναν προκαθορισμένο τρόπο. Για δεδομένα που έχουν οριστεί από τον προγραμματιστή αλλάζει τον τρόπο με τον οποίο χειρίζεται το εκάστοτε στοιχείο με βάση τέσσερα επίπεδα που θα μελετήσουμε στη συνέχεια.

Οι components αυτοί είτε ορίζονται από τον χρήστη είτε από το σύστημα και για προκαθορισμένες λειτουργίες και επεξεργασία οφείλουν να συμμορφώνονται με τις αρχές των διασυνδεδεμένων δεδομένων και να προσφέρουν ένα Semantic Markup.

4.3.1 Επίπεδα παρουσίασης δεδομένων

Όπως απεικονίζεται στο Σχήμα, υπάρχουν τέσσερα επίπεδα βασικών στοιχείων στην εφαρμογή. Κάθε βασική συνιστώσα περιγράφει τις ενέργειες που απαιτούνται για την ανάκτηση και την ενημέρωση των δεδομένων που βασίζονται σε γραφήματα και παρέχει μια βάση για τα εξαρτήματα που ορίζονται από το χρήστη να αλληλοεπιδρούν με τα συνδεδεμένα δεδομένα σε τρεις τρόπους: προβολή, επεξεργασία και περιήγηση.



Διάγραμμα 7: Data Flow Direction

Η ροή δεδομένων στο σύστημα ξεκινά από τη συνιστώσα Dataset που χειρίζεται όλα τα συμβάντα που σχετίζονται με ένα σύνολο πόρων κάτω από ένα όνομα γραφήματος που προσδιορίζεται από ένα URI. Αν σε αυτό το σημείο τα δεδομένα μπορούν να οριστούν με components τα οποία να περιγράφουν την διαχείρισή τους, τότε από τη πρώτη φορά που συλλέγονται αναπαρίστανται και συμπεριφέρονται με αυτόν τον τρόπο. Σκοπός της συνιστώσας δεν είναι μόνο να διαχειρίζεται ομαδικά τα δεδομένα σε αυτό το επίπεδο αλλά να χρησιμοποιείται έτσι ώστε μεγαλύτερα collections από πόρους να αποκτούν ιδιότητες και ιδιαίτερα χαρακτηριστικά γνωρίσματα συμπεριφοράς στην εφαρμογή.

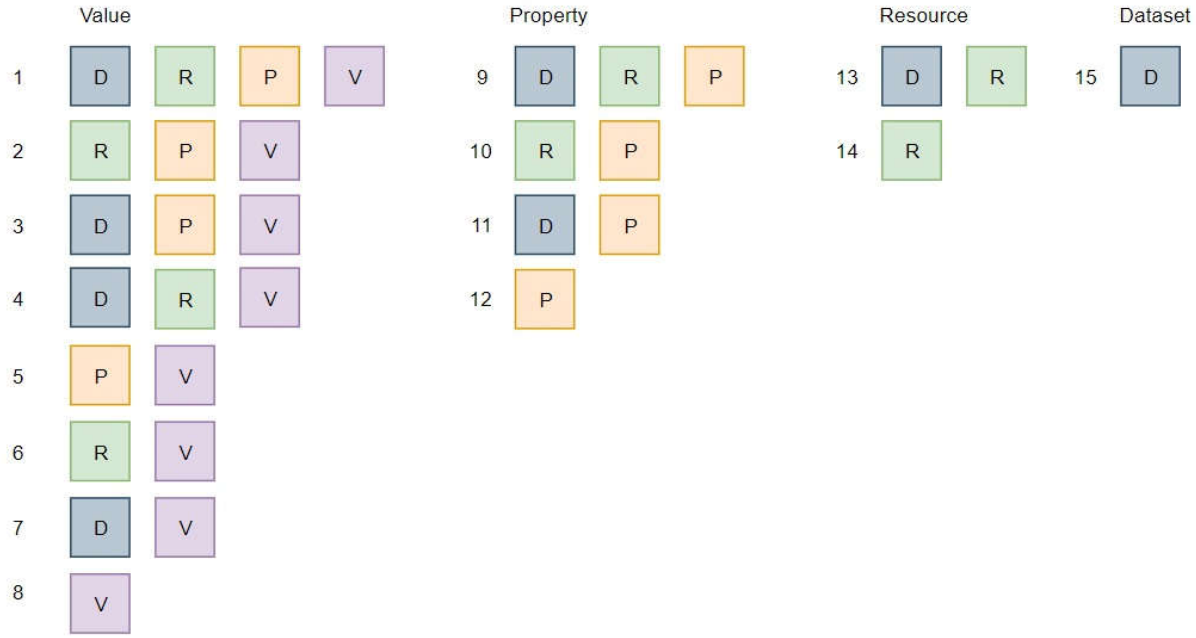
Το επόμενο επίπεδο είναι το στοιχείο Resource που προσδιορίζεται από ένα URI και υποδεικνύει τι περιγράφεται στην εφαρμογή. Ένας πόρος περιγράφεται από ένα σύνολο ιδιοτήτων τις οποίες χειρίζεται το στοιχείο της ιδιότητας. Οι ιδιότητες μπορούν να είναι είτε μεμονωμένες είτε συγκεντρωτικές όταν συνδυάζουν πολλαπλές λειτουργίες ενός πόρου (π.χ. ένα στοιχείο που συνδυάζει ιδιότητες γεωγραφικού μήκους και γεωγραφικού πλάτους, ιδιότητες ημερομηνίας έναρξης και λήξης για ένα εύρος ημερομηνιών κ.λπ.).

Κάθε ιδιότητα δημιουργείται από μια μεμονωμένη τιμή ή πολλαπλές τιμές σε περίπτωση συνολικού αντικειμένου. Οι τιμές των ιδιοτήτων ελέγχονται από το στοιχείο Value. Με τη σειρά τους, τα στοιχεία της ιδιότητας επικαλούνται διαφορετικά στοιχεία για την προβολή, επεξεργασία και περιήγηση στις τιμές των ιδιοτήτων.

Τα στοιχεία του προγράμματος προβολής, επεξεργασίας και περιήγησης είναι τερματικά στη ροή δεδομένων με μοναδική κατεύθυνση, όπου προσαρμοσμένα συστατικά που παράγονται από το χρήστη μπορούν να συνδεθούν στο σύστημα. Οι αλληλεπιδράσεις των χρηστών με τα εξαρτήματα των διασυνδεδεμένων δεδομένων ελέγχονται από ένα σύνολο διαμορφώσεων που ορίζονται σε ένα ή περισσότερα επιλεγμένα επίπεδα συνιστωσών γνωστά ως πεδία.

4.3.2 Πεδία δράσης και παραμετροποιήσεις

Τα δομικά στοιχεία Web παρέχουν μια ευέλικτη προσέγγιση για την προσαρμογή του περιβάλλοντος. Ένα πλαίσιο (context) μπορεί να είναι ένας συγκεκριμένος τομέας ενδιαφέροντος, μια συγκεκριμένη απαίτηση χρήστη ή και τα δύο. Προκειμένου να είναι δυνατή η προσαρμογή και η εξατομίκευση, η προσέγγιση μας εκμεταλλεύεται τις έννοιες Πεδίο εφαρμογής και διαμόρφωση. Ένα πεδίο ορίζεται ως μια ιεραρχική μετάθεση των συνιστωσών δεδομένων, πόρων, ιδιοτήτων και τιμής (Dataset, Resource, Property and Value components). Κάθε πεδίο εφαρμογής παρέχει ένα συγκεκριμένο επίπεδο εξειδίκευσης σε ένα δεδομένο πλαίσιο που κυμαίνεται από 1 (πιο συγκεκριμένο) έως 15 (λιγότερο συγκεκριμένο).



Εικόνα 29: View Properties Configuration

Τα πεδία καθορίζονται χρησιμοποιώντας είτε τα URI των ονομασμένων γραφημάτων, πόρων και ιδιοτήτων είτε προσδιορίζοντας τους τύπους πόρων και τους τύπους δεδομένων. Μια παραμετροποίηση ορίζεται ως μια ρύθμιση που επηρεάζει τον τρόπο με τον οποίο ερμηνεύονται και αποδίδονται τα στοιχεία LDA και Web (π.χ. αναπαράγουν ένα συγκεκριμένο στοιχείο για μια συγκεκριμένη ιδιότητα RDF ή επιβάλλουν ένα στοιχείο για να εμφανίζουν τα URI μιας σελίδας της Wikipedia για τους πόρους DBpedia).

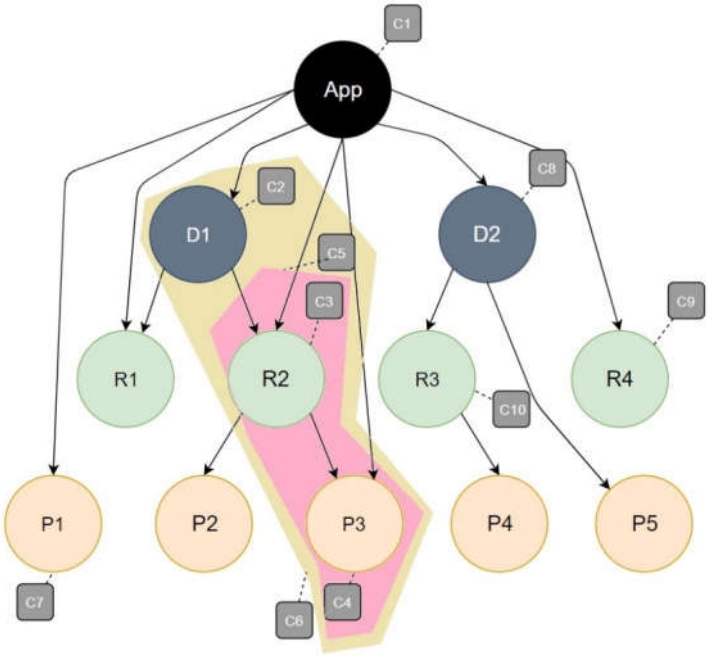
Η προσαρμογή του UI αντιμετωπίζεται με τη διασταύρωση των παραμετροποιήσεων για πεδία (scopes), με τη συμπλήρωση των διαμορφώσεων και την αντικατάστασή τους όταν εντοπίζεται ένα

ειδικότερο εφαρμοστέο πεδίο. Όπως φαίνεται στον αλγόριθμο παρακάτω στην χειρότερη περίπτωση όταν χρησιμοποιούνται τα πεδία DRPV και το UI υποτίθεται ότι θα καταστήσει τα στοιχεία της αξίας, και τα 15 πεδία πρέπει να διασταυρωθούν για την προσαρμογή:

```
1  InitialConfig = {initial application
       configuration}
2  Context = [array of scopes with the
       corresponding configuration objects]
3  Config = InitialConfig
4  for (i = 15; i < 1; i--) {
5      Config.compareWith(Context[i]) {
6          Config.addMissingAttributes()
7          Config.overwriteExistingAttributes()
8      }
9  }
```

Εικόνα 30: Representation Algorithm

Στο παρακάτω σχήμα γίνεται ορατό ένα παράδειγμα του υπερ-γράφου διαμόρφωσης της εφαρμογής που περιέχει πεδία με το μέγιστο βάθος DRP. Το γράφημα ορίζει μια γενική διαμόρφωση για την εφαρμογή ως C1. Υπάρχουν διαμορφώσεις που ορίζονται για το φάσμα δεδομένων D1 ως C2, για το πεδίο πόρων R2 ως C3 και για το πεδίο ιδιότητας P2 ως C4. Υπάρχουν επίσης διαμορφώσεις για το πεδίο RP R2P2 ως C5 και για το πεδίο DRP D1R2P2 ως C6.



Διάγραμμα 8: View Configuration Example

Ας υποθέσουμε ότι έχουμε μια ρύθμιση με το ακόλουθες τιμές για τα πεδία και τις διαμορφώσεις:

- D1= <http://myendpoint.org/users>

- R2= type foaf:Person
- P2= rdfs:label
- C1={{viewer:'basic'},{attr1:1},{attr2:3}}
- C2={{attr1:0},{attr3:2}}
- C3={{attr3:1},{attr4:4},{attr5:1}}
- C4={{attr5:2},{attr6:1}}
- C5={{viewer:'contact'},{attr3:5},{attr7:6}}
- C6={{attr3:8},{attr7:1},{attr8:3}}

Με τις παραπάνω ρυθμίσεις, όταν ένα στοιχείο ιδιότητας για rdfs:label αποδίδεται χωρίς το σύνολο δεδομένων και τον πόρο πλαίσιο, η διαμόρφωση θα είναι:

{{viewer: 'basic'}, {attr1: 1}, {attr2: 3}, {attr5: 2}, {attr6: 1}}

Όταν ο component της ιδιότητας γίνει render μέσα στο περιβάλλον πόρων του τύπου foaf:Person, οι ρυθμίσεις για το πρόγραμμα προβολής και attr5 αντικαθίστανται και νέες ρυθμίσεις για attr3, attr4 και attr7 προστήθενται:

{{viewer:'contact'},{attr1:1},{attr2:3},{attr3:5},{attr4:4}, {attr5:1},{attr6:1},{attr7:6}}

Όταν παρέχεται το πρόσθετο πλαίσιο του συνόλου δεδομένων ως <http://myendpoint.org/users>, attr3 και attr7 αντικαθίστανται και μια νέα ρύθμιση για το attr8 προστίθεται:

{{viewer:'contact'},{attr1:0},{attr2:3},{attr3:8},{attr4:4}, {attr5:1},{attr6:1},{attr7:1},{attr8:3}}

Τα πεδία μπορούν επίσης να καθοριστούν ανά χρήστη, διευκολύνοντας την έκδοση και την επαναχρησιμοποίηση των διαμορφώσεων που αφορούν το χρήστη. Οι διαμορφώσεις που καθορίζονται από το χρήστη παρέχουν διαφορετικές προβολές σε στοιχεία και ως εκ τούτου δεδομένα, βάσει των διαφορετικών προσώπων που ασχολούνται με αυτά.

Εκτός από την προσαρμογή λεπτομερών components, η εφαρμογή μας παρέχει ένα λεπτομερή έλεγχο πρόσβασης στα δεδομένα μέσω των πεδίων των στοιχείων. Για παράδειγμα, ένας προγραμματιστής εφαρμογών μπορεί να περιορίσει την πρόσβαση σε μια συγκεκριμένη ιδιότητα συγκεκριμένου πόρου σε ένα συγκεκριμένο σύνολο δεδομένων και σε μια συγκεκριμένη λειτουργία αλληλεπίδρασης.

4.3.3 Σημασιολογική σήμανση των Web Components

Η έμφυτη υποστήριξη του RDF σε components της εφαρμογής επιτρέπει την αυτόματη δημιουργία σημασιολογικής σήμανσης στο επίπεδο UI. Χαμηλές σημασιολογικές τεχνικές όπως το RDFa, τα Mircodata και το JSON-LD μπορούν να ενσωματωθούν στα βασικά στοιχεία για να εκθέσουν δομημένα δεδομένα σε τρέχουσες μηχανές αναζήτησης που είναι σε θέση να αναλύουν σημασιολογική σήμανση.

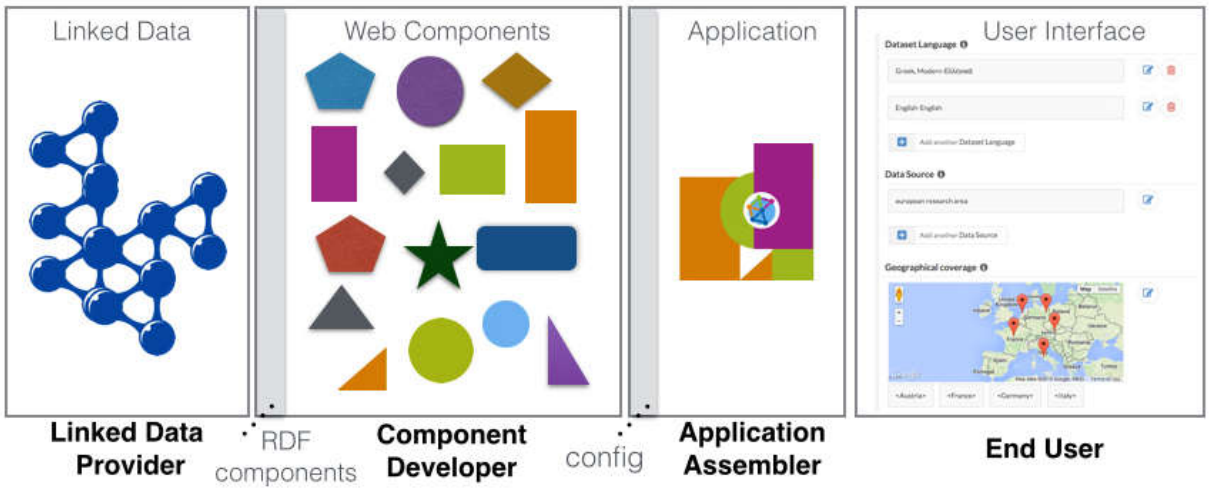
Για παράδειγμα, ένα συστατικό στοιχείο που δημιουργήθηκε με βάση τις οντολογίες Good Relations ή Schema.org, μπορεί να εκθέσει αυτόματα τα δεδομένα του προϊόντος ως αποσπάσματα Google Rich για προϊόντα τα οποία θα παρέχουν καλύτερη προβολή των δεδομένων στα αποτελέσματα αναζήτησης στο Web (δηλ. SEO).

Εκτός από τον αυτόματο σχολιασμό των δεδομένων που παρέχονται από τα συστατικά στοιχεία του διαδικτυακού τόπου, η προσέγγιση προσφέρει ημι-αυτόματη σήμανση των στοιχείων του Web με τη δημιουργία μεταδεδομένων συστατικών στοιχείων. Τα μεταδομένα των components αποτελούνται από δύο κατηγορίες σήμανσης:

- Αυτόματη σήμανση που παράγεται από την ανάλυση του περιεχομένου του πακέτου των components και μεταδεδομένα σχετικά με το στοιχείο και τις εξαρτήσεις του. Περιλαμβάνει γενικά μεταδεδομένα, όπως όνομα, περιγραφή, έκδοση, αρχική σελίδα, συγγραφέα καθώς και τεχνικά μεταδεδομένα στο χώρο αποθήκευσης πηγών συνιστωσών και εξαρτήσεις.
- Χειροκίνητη σήμανση που δημιουργείται από συγγραφείς των components, η οποία εκθέτει μεταδεδομένα όπως επίπεδο στοιχείου (σύνολο δεδομένων, πόρο, ιδιότητα, τιμή), λεπτομερειακότητα (ατομική, συγκεντρωτική), λειτουργία (προβολή, επεξεργασία, περιήγηση) και προδιαγραφές παραμέτρων διαμόρφωσης.

Παρόμοια με τη σήμανση περιεχομένου, η σήμανση Component μπορεί να χρησιμοποιήσει κοινώς γνωστές οντολογίες όπως το Schema.org για να βελτιώσει την ορατότητα των web components και να επιτρέψει στους συναρμολογητές εφαρμογών να κατανοήσουν καλύτερα την προβλεπόμενη χρήση και τις δυνατότητες ενός δεδομένου στοιχείου.

4.3.4 Τα ενδιαφερόμενα μέρη και ο κύκλος ζωής



Εικόνα 31: Κύκλος ζωής Components

Όπως φαίνεται στην παραπάνω εικόνα ο κύκλος ζωής των components στην υλοποίησή μας περιλαμβάνει τέσσερις βασικούς τύπους ενδιαφερομένων:

- Linked Data Provider.

Δεδομένου ότι η προσέγγιση επικεντρώνεται κυρίως σε εφαρμογές συνδεδεμένων δεδομένων, η παροχή δεδομένων συμβατών με το RDF αποτελεί ουσιαστική φάση ανάπτυξης. Υπάρχουν διαφορετικά στάδια στην παροχή συνδέσμων δεδομένων,[47] όπως η εξαγωγή δεδομένων, η αποθήκευση, η

διασύνδεση, ο εμπλουτισμός, η ανάλυση ποιότητας και η επισκευή, τα οποία πρέπει να ληφθούν υπόψη από τους επιστήμονες δεδομένων και τους εμπειρογνώμονες των συνδεδεμένων δεδομένων. Μόλις τα δεδομένα και τα σχήματα παρασχεθούν στο σύστημα, αυτό μπορεί να φέρει μια αμοιβαία αξία στους παρόχους συνδεδεμένων δεδομένων για την καλύτερη κατανόηση και επεξεργασία των δεδομένων όταν χρειάζεται.

Για παράδειγμα, στην περίπτωση των γεωγραφικών συντεταγμένων, μια συνιστώσα χάρτη μπορεί να επιτρέπει στους παρόχους δεδομένων να επεξεργάζονται εύκολα τα δεδομένα (π.χ. διφορούμενες οντότητες) μέσα σε ένα συγκεκριμένο γεωγραφικό όριο με οπτικό τρόπο. Τα Actions Stores RESTful Services Endpoints φέρνουν σε επικοινωνία τον CRUD Data Dispatcher ο οποίος επισπεύδει τη μονής κατεύθυνσης FLUX ροή δεδομένων στο framework της υλοποίησης.

- Component Developer.

Οι προγραμματιστές εξαρτημάτων είναι σχεδιαστές του UX και προγραμματιστές Ιστού που συμμετέχουν στην κατασκευή εξαρτημάτων. Υπάρχουν δύο τύποι δομικών στοιχείων Web που αναπτύσσονται σε αυτό το βήμα:

α) Κύρια στοιχεία τα οποία αφηγούνται το υποκείμενο μοντέλο δεδομένων RDF. Αυτά τα στοιχεία είναι ενσωματωμένα στο σύστημα, ωστόσο μπορούν ακόμα να αντικατασταθούν από προγραμματιστές που έχουν επάρκεια στο Σημασιολογικό Ιστό και τα Συνδεδεμένα Δεδομένα.

β) Community-driven components τα οποία εκμεταλλεύονται τα βασικά στοιχεία. Αυτά τα στοιχεία είτε δημιουργούνται από το μηδέν είτε αναμιγνύονται και αντικαθιστούν τα υπάρχοντα στοιχεία Web που βρίσκονται στον Ιστό.

- Application Assembler.

Το κύριο καθήκον των συναρμολογητών εφαρμογών είναι να προσδιορίσουν τα σωστά στοιχεία και διαμορφώσεις για την εφαρμογή, και να τα συνδυάσουν κατά τρόπο που να ανταποκρίνεται στις απαιτήσεις της εφαρμογής. Μέσα στο σύστημα στοιχείων, τα μεταδεδομένα που παρέχονται από κάθε στοιχείο Web διευκολύνουν την ανακάλυψη σχετικών στοιχείων. Έχοντας κοινόχρηστα λεξιλόγια στα Linked Open Data επιτρέπει στους συναρμολογητές όχι μόνο να επαναχρησιμοποιούν τα εξαρτήματα αλλά και να επαναχρησιμοποιούν τις υπάρχουσες διαμορφώσεις και πεδία που δημοσιεύονται στον Ιστό.

Για παράδειγμα, αν υπάρχει ήδη μια κατάλληλη διαμόρφωση για το πεδίο RP που χρησιμοποιεί foaf:Person ως τύπος πόρων και dcterms:description ως URI ιδιότητας, ο assembler μπορεί να ξαναχρησιμοποιήσει αυτή τη διαμόρφωση μέσα στην εφαρμογή του.

- Τελικός χρήστης.

Οι τελικοί χρήστες βιώνουν την εργασία με τα συστατικά μέρη για να επιτύχουν στόχους σε έναν συγκεκριμένο τομέα εφαρμογής. Ως εκ τούτου, μπορούν να ζητήσουν την ανάπτυξη νέων εξαρτημάτων ή διαμορφώσεων για την εκπλήρωση των απαιτήσεών τους και αναμένεται να παρέχουν ανατροφοδότηση σχετικά με τα υπάρχοντα στοιχεία.

4.4 Implementation

Για την κατασκευή της πλατφόρμας υλοποίησης δυναμικής παρουσίασης και επεξεργασίας διασυνδεδεμένων δεδομένων κατασκευάσαμε μια υλοποίηση η οποία διαχειρίζεται διασυνδεδεμένα δεδομένα με έναν δυναμικό τρόπο καθώς οι οντότητες που θέλουμε να περιγράψουμε για το παράδειγμα της ηλεκτρονική βιβλιοθήκης απαιτεί τη διαχείριση και την επεξεργασία διασυνδεδεμένων δεδομένων πολλών τύπων. Οι κύριες οντότητες που μας απασχόλησαν είναι αυτές των συγγραμμάτων, των συντακτών και των εκδοτών. Κατά την υλοποίηση και τη συγκομιδή πληροφοριών αλλά και σύνθεσης των δεδομένων το εύρος τον τύπων των διασυνδεδεμένων δεδομένων μεγάλωνε με αποτέλεσμα όλο και περισσότερες οντότητες να χρήζουν άξιες διαχείρισης και επεξεργασίας. Πέραν των κύριων οντοτήτων παρουσιάστηκε η ανάγκη για τον χειρισμό κάθε ιδιότητας αντικειμένου που χαρακτηρίζεται σαν οντότητα με ένα σύνδεσμο.

Με τη ραγδαία αύξηση των αναγκών και του πλήθους των στοιχείων η απόκρυψη και διανομή συγκεκριμένης πληροφορίας προς τον χρήστη αποτέλεσε το εύκολο κομμάτι της υλοποίησης. Η επαναχρησιμοποίηση του κώδικα χαρακτηρισμένων στοιχείων έγινε αναγκαία και ο τρόπος διαχείρισης των δεδομένων αποτέλεσε το κύριο κομμάτι της υλοποίησης με σκοπό να αποφευχθεί το boilerplate. Με την υλοποίηση παρατηρήσαμε εκτός από τις βασικές οθόνες που αποτελούν την εφαρμογή, όπως το index και το about, ότι οι σελίδες που χρησιμοποιήθηκαν για την διαχείριση της πληροφορίας των διασυνδεδεμένων δεδομένων μπορούσε να εφαρμοστεί και στα τρία στοιχεία που θα αποτελούσαν την εφαρμογή.

Με κύριο γνώμονα την αναζήτηση πληροφοριών και τη παρουσίασή τους στον χρήστη και για να καθίσταται το γεγονός της επέκτασης της εφαρμογής εύκολο σε ενδεχόμενες αλλαγές στο μέλλον από παρόχους οι οποίοι διαθέτουν αυτά τα δεδομένα κατασκευάσαμε μια εφαρμογή που παρέχει ένα δυναμικό τρόπο απεικόνισης και επεξεργασίας. Για την υλοποίηση χρησιμοποιήσαμε το interface του react-cli. Με την πλατφόρμα αυτή το build της εφαρμογής σε σχέση με τις ανάγκες του να συμπεριλάβει ήδη υλοποιημένους components που διατίθενται από την κοινότητα ανοικτού κώδικα, έγινε ευκολότερο.

Για τις ανάγκες της υλοποίησης έγιναν Import βιβλιοθήκες από κώδικα οι οποίες μας βοήθησαν να σχηματίσουμε την εφαρμογή μας όσον αναφορά το γραφιστικό αλλά και το κομμάτι του προγραμματισμού για την αναζήτηση πληροφοριών. Χρησιμοποιήθηκαν components οι οποίοι υλοποιούν τον flexible react, τις ασύγχρονες κλήσεις, το core system της javascript, την τεχνολογία react, τη διαχείριση json, την διαχείριση cookies και για τους σκοπούς του προγραμματισμού το babel και το webpack. Για την κατασκευή της εφαρμογής υλοποιήθηκαν components με την αρχιτεκτονική flux που ακολουθεί ένα μοντέλο μονής κατεύθυνσης ροής δεδομένων. Με βάση αυτή την αρχιτεκτονική υλοποίηση υλοποιήθηκαν components οι οποίοι διαχωρίστηκαν σε Actions, Stores και Dispatchers.

Για να επιτρέψει την εκκίνηση των LDA UIs, η εφαρμογή μας παρέχει ένα ολοκληρωμένο πλαίσιο που συνδυάζει τα ακόλουθα κύρια στοιχεία:

- Ένα σύνολο υπηρεσιών RESTful Web που επιτρέπουν βασικές λειτουργίες CRUD στα Linked Data χρησιμοποιώντας SPARQL queries.
- Ένα σύνολο βασικών στοιχείων που ονομάζονται Reactors που υλοποιούν βασικά στοιχεία Συνδεδεμένων Δεδομένων μαζί με τις αντίστοιχες ενέργειες και stores.
- Ένα σύνολο προεπιλεγμένων στοιχείων που επιτρέπουν τη βασική προβολή, επεξεργασία και περιήγηση των Συνδεδεμένων Δεδομένων.
- Ένα σύνολο ελάχιστων βιώσιμων διαμορφώσεων που βασίζονται στον τύπο δεδομένων και ιδιοτήτων από τα συνήθως χρησιμοποιούμενα λεξιλόγια (π.χ. foaf, dcterms και SKOS).
- Βασική προσθήκη ελέγχου πρόσβασης που επιτρέπει τον περιορισμό της πρόσβασης ανάγνωσης / εγγραφής στα δεδομένα.

Η εφαρμογή συμμορφώνεται με την Αρχιτεκτονική Microservices,[48] όπου τα υπάρχοντα εξαρτήματα ReactJS μπορούν να επεκταθούν με συμπληρωματικές υπηρεσίες LD. Σε αντίθεση με την κεντρική μονολιθική αρχιτεκτονική, η αρχιτεκτονική microservices επιτρέπει την τοποθέτηση των κύριων λειτουργιών του LDA σε χωριστές αποσυνδεδεμένες υπηρεσίες και κλίμακα, διανέμοντας αυτές τις υπηρεσίες σε διακομιστές, αναπαράγοντας ανάλογα με τις ανάγκες. Αυτό το αρχιτεκτονικό στυλ συμβάλλει επίσης στην ελαχιστοποίηση της ανακατανομής ολόκληρης της εφαρμογής όταν ζητούνται αλλαγές σε στοιχεία. Η σημασιολογική σήμανση των δεδομένων υποστηρίζεται εγγενώς στο framework, ενσωματώνοντας τις σχολιασμούς των Microdata μέσα στα στοιχεία του διαδικτυακού τόπου.

4.4.1 Δομικά Στοιχεία

Για την εφαρμογή υλοποιήθηκαν κάποια κύρια βασικά συστατικά τα οποία χρήζουν άξια αναφοράς προκειμένου να κατανοηθεί η λειτουργία της εφαρμογής και να περιγραφεί η λειτουργία της. Σχετικά με την υλοποίηση θα αναφέρουμε εδώ κάποια από αυτά με τις ιδιότητές τους και αρχεία τα οποία τα αποτελούν. Δεν θα προβούμε σε ανάλυση όλων των δομικών στοιχείων που αποτελούν την εφαρμογή καθώς το πλήθος των αρχείων λόγω της διάσπασης σε μικρότερα δομικά στοιχεία που απαρτίζουν κάθε κομμάτι υλοποίησης εξελίσσεται σε μεγάλη ανάλυση αλλά θα αναφέρουμε και θα περιγράψουμε τα δομικά στοιχεία που αποτελούν την εφαρμογή όσον αναφορά τα διασυνδεδεμένα δεδομένα ώστε να είμαστε σε θέση ο αναγνώστης να αντιληφθεί ένα κύκλο επεξεργασίας.

- Configs

Αρχικά μια χρήσιμη αναφορά είναι αυτή των configuration files της εφαρμογής. Τα αρχεία που αποτελούν τον συγκεκριμένο φάκελο είναι αρχεία που καθορίζουν την λειτουργεία της εφαρμογής και την παραμετροποίηση των components που χρησιμοποιούν τα διασυνδεδεμένα δεδομένα.

Για αρχή υπάρχει το αρχείο **general.js** το οποίο διαχειρίζεται μεταβλητές που είναι αναγκαίες για την εφαρμογή και πρέπει να είναι ορατές από οποιοδήποτε σημείο. Περιέχει μεταβλητές που ορίζουν τους τίτλους της εφαρμογής, τα URLs από γράφους δεδομένων (που είναι αποθηκευμένα σε τελικά σημεία) που είναι αποθηκευμένα τα configuration, καθώς η εφαρμογή μπορεί να εξάγει αντικείμενα configuration με τη μορφή μεταδεδομένων. Επίσης τα mappings που χρησιμεύουν για την εισαγωγή δεδομένων που έχουν γίνει export από το σύστημα, το path για τον φάκελο με το upload των

δεδομένων και άλλες Boolean μεταβλητές οι οποίες επιτρέπουν και αποτρέπουν λειτουργίες της διαπροσωπίας.

Στη συνέχεια υπάρχει το **routes.js** το οποίο είναι ένα αρχείο που αναφέρει και παραμετροποιεί τις καταστάσεις του συστήματος για μετάβαση. Ένα ιδιαίτερο χαρακτηριστικό είναι ότι κάθε κατάσταση που ορίζεται επεξεργάζεται τον handler της κατάστασης με require τα απαραίτητα components για την υλοποίηση προκαθορισμένων διαδικασιών. Επίσης ορίζει action events για κάθε αντικείμενο τα οποία εκτελούν ένα event στο context της κατάστασης όταν μια αλλαγή του συστήματος πραγματοποιηθεί. Αυτά είναι events ανακατεύθυνσης με actions τα οποία έχουν συμπεριληφθεί και αλλάζουν το context της εφαρμογής είτε με την επιλογή του context dispatch, είτε με την επιλογή executeAction (προκαθορισμένες μέθοδοι του Fluxible Context που γίνεται import από την εφαρμογή με το πακέτο flexible-router) διαχειρίζονται την αλλαγή των καταστάσεων της εφαρμογής. Με αυτόν τον τρόπο επιτυγχάνονται τα event της δρομολόγησης με κάποιο action περισυλλογής δεδομένων στις περιπτώσεις ομαδικής αναζήτησης και της λεπτομερούς απεικόνισης.

Το **server.js** είναι ένα άλλο configuration file το οποίο παραμετροποιεί την πόρτα στην οποία θα δημοσιευτεί η εφαρμογή, τροποποιεί τα SPAQL End points που χρησιμοποιούν τα services με αντικείμενα που ορίζουν τον host, port , path, graph Name και endpointType. Επίσης ορίζει αντικείμενα όπως το dbpedia Lookup Service, dbpedia Spotlight Service και google Recaptcha Service που χρησιμοποιούνται από την εφαρμογή.

Το **facets.js** το οποίο είναι ένας configurator που παραμετροποιεί τα facets της σελίδας ομαδικής αναζήτησης. Σε αυτόν κάθε αντικείμενο ορίζεται με το resource που αναλύει και περιέχει 2 στοιχεία. Μια λίστα τιμών στην οποία περιλαμβάνονται όλα τα resource που περιγράφονται και ένα αντικείμενο config το οποίο κάθε στοιχείο που έχει αναφερθεί μπορεί να αποκτήσει ιδιότητες όπως ο component που θα γίνει Browse και View, σε ποια θέση στην λίστα θα εμφανίζεται, αν θα εμφανίζεται και σε ποια κατηγορία (ιδιότητα η οποία το κάνει να εμφανίζεται σε μια λίστα accordion στην επιλογή των properties στην σελίδα της αναζήτησης), αν θα μπορεί να προσαρτηθεί στην ανάλυση, αν θα εμφανίζει την ερώτηση που παράγεται σε log της εφαρμογής, αν θα είναι ενεργοποιημένα τα extra φίλτρα και άλλα στοιχεία που χαρακτηρίζουν την λειτουργεία του. Για τις ανάγκες της εφαρμογής ορίζεται και ένα αντικείμενο με resource τη τιμή generic το οποίο αν δεν έχει εφαρμοστεί κάποια τροποποίηση χρησιμοποιείται από κοινού για μια προκαθορισμένη λειτουργία των facets.

```
resource: {
    'generic': {
        usePropertyCategories: 1,
        propertyCategories: ['overview', '
            legalAspects', 'technicalAspects'],
        resourceReactor: ['Resource'],
        shortenURI: 1
    }
},
property: {
    'generic': {
        propertyReactor: ['IndividualProperty'],
        objectReactor: ['IndividualObject'],
        objectIViewer: ['BasicIndividualView'],
        objectIEditor: ['BasicIndividualInput']
    },
    'http://purl.org/dc/terms/language': {
        allowNewValue: 1,
        label: ['Dataset Language'],
        category: ['overview'],
        hint: ['The language of the dataset.
            Resources defined by the Library of
            Congress (http://id.loc.gov/vocabulary/
            iso639-1.html, http://id.loc.gov/
            vocabulary/iso639-2.html) SHOULD be
            used.'],
        objectIViewer: ['LanguageView'],
        objectIEditor: ['LanguageInput'],
        defaultValue: ['http://id.loc.gov/vocabulary
            /iso639-1/en']
    },
    'http://purl.org/dc/terms/spatial': {
         label: ['Geographical coverage'],
         category: ['overview'],
         hint: ['The geographical area covered by
              the dataset.'],
         allowNewValue: 1,
         objectReactor: ['AggregateObject'],
         objectAViewer: ['DBpediaGoogleMapView'],
         objectIViewer: ['BasicDBpediaView'],
         asWikipedia: 1,
         objectAEditor: ['BasicAggregateInput'],
         objectIEditor: ['DBpediaInput'],
         lookupClass: ['Place']
     },
    'http://purl.org/dc/terms/subject': {
        category: ['overview'],
        label: ['Keywords'],
        hint: ['Tags a dataset with a topic.'],
        allowNewValue: 1,
        objectIEditor: ['DBpediaInput'],
        objectIViewer: ['BasicDBpediaView'],
        asWikipedia: 1
    },
    'http://purl.org/dc/terms/license': {
        category: ['legalAspects'],
        label: ['License'],
        allowNewValue: 1,
        objectIViewer: ['BasicOptionView'],
        objectIEditor: ['BasicOptionInput'],
        options: [
            {label: 'Open Data Commons Attribution
                License', value: 'http://www.
                opendatacommons.org/licenses/by/'},
            {label: 'Creative Commons Attribution-
                ShareAlike', value: 'http://
                creativecommons.org/licenses/by-sa/
                3.0/'}
        ],
        allowUserDefinedValue: 1
    }
}
```

Τέλος ένα άλλο αρχείο είναι αυτό του **reactor.js** στο οποίο ένα στιγμιότυπο φαίνεται στη διπλανή εικόνα. Το συγκεκριμένο configuration file είναι υπεύθυνο για τη διαχείριση των πόρων, των ιδιοτήτων και του συνόλου των δεδομένων όπως περιγράψαμε στην ενότητα για τα επίπεδα παρουσίασης και τον αλγόριθμο σχεδιασμού προσδίδοντας ιδιότητες σε κάθε επίπεδο παρουσίασης. Στο αρχείο μπορούν να τροποποιηθούν κατάλληλα οι σύνδεσμοι σε επίπεδο dataset, resource και property. Για κάθε επίπεδο παρουσίασης και παραμετροποίησης είναι υλοποιημένη

Εικόνα 32: Reactor Config Example

μια συνιστώσα generic η οποία κάνει δυνατή την default συμπεριφορά των αντικειμένων τα οποία επεξεργάζονται. Από εκεί και πέρα ανάλογα με το επίπεδο το οποίο ορίζονται τα αντίστοιχα configurations για κάθε σύνδεσμο μπορεί να παραμετροποιηθεί ο reactor που τα επεξεργάζεται. Ορίζει τους components που αποτελούν την συμπεριφορά του συστήματος σε διαδικασίες οπτικής προβολής και επεξεργασίας. Ορίζει μια ετικέτα που χαρακτηρίζει την ιδιότητα και αντικαθιστά το όνομα της ιδιότητας. Περιέχει ένα hint το οποίο εμφανίζεται με μια ένδειξη δίπλα από το property του αντικειμένου. Ορίζει αν επιτρέπεται η επεξεργασία της ιδιότητας και υλοποιεί και κάποιες άλλες ιδιότητες κυρίως οπτικού περιεχομένου που αποτελούν βασικό κομμάτι της συμπεριφοράς στον component όταν η πληροφορία γίνει render.

- Stores

Τα stores που υλοποιούν την εφαρμογή μας είναι αυτά τα οποία διαχειρίζονται και τα δεδομένα της εφαρμογής και είναι υπεύθυνα για το render της πληροφορίας στις τιμές που ορίζονται από κάθε component που υλοποιεί τις κεντρικές σελίδες.

Το **RouteStore** είναι για αρχή το Store το οποίο διαβάζει από το configuration αρχείο τις καταστάσεις του συστήματος.

Το **ApplicationStore** είναι αυτό το οποίο διαχειρίζεται τον τίτλο στης εφαρμογής και κάνει το σύστημα να εναλλάσσεται σε κατάσταση loading mode on – off.

Το **DatasetStore** είναι αυτό το οποίο διαχειρίζεται τα αποτελέσματα της αναζήτησης στον κύριο component σελιδοποίησης. Περιέχει handlers οι οποίοι με events ανανεώνουν την λίστα των δεδομένων και το σύνολο των αποτελεσμάτων. Για την λειτουργεία του, διατηρεί μεταβλητές που χαρακτηρίζουν το graphname που είναι απαραίτητο για τη δημιουργία ενός SPARQL ερωτήματος και είναι υπεύθυνο να δημοσιεύσει μεταδεδομένα που σχηματίζουν το αντικείμενο που περιγράφεται. Διατηρεί το datasetURI που χρησιμοποιείται για να δηλώσει τον σύνδεσμο που είναι αποθηκευμένος ο γράφος που αναδημοσιεύει τα δεδομένα. Ένα άλλο στοιχείο είναι τα resources που διατίθενται σαν properties των αντικειμένων που επεξεργάζονται, και το resourceQuery το οποίο έχει σχηματιστεί για να φέρει τα δεδομένα από το σύνδεσμο που ερωτήθηκε. Υλοποιεί ένα config στοιχείο που μεταβιβάζει τα δεδομένα που έχουν προέλθει από το τον αλγόριθμο αναπαράστασης και μια ιδιότητα total που περιέχει το σύνολο των δεδομένων. Τέλος μια error ιδιότητα που μεταβιβάζει την κατάσταση σφάλματος για την συμπεριφορά του συστήματος όταν προκληθεί κάποιο λάθος κατά τη συλλογή.

Το **FacetedBrowserStore** είναι το Store το οποίο διαχειρίζεται τα facets της εφαρμογής τα οποία αλληλοεπιδρούν με τα αποτελέσματα της αναζήτησης. Υλοποιεί συναρτήσεις που είναι συνδεδεμένες με handlers για ιδιότητες όπως :

1. Της φόρτωσης των Master facets και των facets που υλοποιούν resources
2. Φόρτωσης περισσότερων αποτελεσμάτων
3. Φόρτωσης του μετρητή των αποτελεσμάτων
4. Χειρισμό των effects που αλληλοεπιδρούν μεταξύ τους αλλά και με την συλλογή των πληροφοριών
5. Φόρτωσης των configuration που τα τροποποιούν και του καθαρισμού τους.

Διατηρεί επίσης μεταβλητές που αντιπροσωπεύουν τα αντικείμενα των facets και πλήθους, resources τα οποία προέρχονται σαν αποτέλεσμα αναζήτησης από φίλτρα, το άθροισμα των καταχωρήσεων, τη

σελίδα που αλληλοεπιδρά με τον κύριο component, το resourceQuery που έχει σχηματιστεί για την συγκομιδή στην ερώτηση που αντιπροσωπεύει και προσαρτάτε στην κύρια ερώτηση, τα φίλτρα κάθε facet που έχουν εφαρμοστεί, το configuration του dataset καθώς και το configuration του component που τα σχηματίζει και μια κατάσταση error.

Το **ResourceStore** το οποίο είναι υπεύθυνο για τη λεπτομερή αναπαράσταση. Υλοποιεί μια συνάρτηση όταν το resource που ερωτηθεί ολοκληρωθεί και γίνεται bind στον handler. Συγκρατεί την πληροφορία του ονόματος του γράφου, το datasetURI που υλοποιεί, το resourceURI που υλοποιεί και το resource Type που είναι απαραίτητα για τα μεταδεδομένα που αναδημοσιεύει. Διαθέτει μια μεταβλητή για τον τίτλο που εμφανίζεται επάνω από τα tabs, την κατηγορία την οποία δείχνει αναφερόμενοι στο ενεργό tab (καθώς όπως είπαμε το κάθε property μπορεί να αντιστοιχιστεί σε μια κατηγορία), μια λίστα με τα properties που αναπαριστά και μια λίστα με τα αντίστοιχα property paths που ορίζονται για το κάθε config αντικείμενο που τροποποιεί το component που συντάσσει την σελίδα. Τέλος ένα αντικείμενο error που καθορίζει την συμπεριφορά του σε περίπτωση σφάλματος.

Τέλος το **IndividualObjectStore** το οποίο με την σειρά του υλοποιεί μια συνάρτηση που κάνει ανανέωση των δεδομένων συνδεδεμένη με έναν handler. Αυτή με ένα payload που καλείται ανανεώνει τα Object Properties και Types ανάλογα με το payload.ObjectURI που δηλώνει το property του κάθε αντικειμένου. Διατηρεί δύο αντικείμενα τα οποία περιλαμβάνουν τα objectProperties και τα objectTypes με δείκτες τα URI των ιδιοτήτων και τιμές τα value κάθε ιδιότητας τα οποία στελεχώνουν τα μεταδεδομένα και τις τιμές.

- Actions

Ο φάκελος αυτός περιλαμβάνει όλα τα events που αποτελούν είδος χειρισμού στην εφαρμογή και χρησιμοποιούνται από τους components, οι οποίοι απαρτίζουν την εφαρμογή, με τη διαδικασία του Import. Οι συναρτήσεις που υλοποιούν τα actions είναι κυρίως συναρτήσεις που δέχονται σαν Input το context της εφαρμογής ένα payload και μια συνάρτηση done. Με αυτόν τον τρόπο αποσπούν από το context της εφαρμογής συναρτήσεις και πραγματοποιούν λειτουργίες οι οποίες ορίζονται από τα services. Στο αποτέλεσμα της κάθε κλήσης χρησιμοποιούν το context.dispatch με προκαθορισμένα λεκτικά τα οποία έχουν οριστεί σαν handlers στα Stores της εφαρμογής και σε διάφορους components με αποτέλεσμα να ανανεώνεται το εκάστοτε Store και να ενεργοποιούνται συναρτήσεις. Τα actions υλοποιούν λειτουργίες της φόρτωσης των Datasets, Resources, Facets, lookup στο σύνδεσμο της DBPedia, count Resources, create & delete Resource, Property, Individual Object και γενικά τα περισσότερα event του χρήστη αλλά και της εφαρμογής. Σχετικά με τα events όπως αναφέραμε υπάρχουν αυτά που προκαλούνται από τον χρήστη αλλά και από components που κατά το bind των δεδομένων καλούν services για την ανάκτηση δεδομένων.

- Components

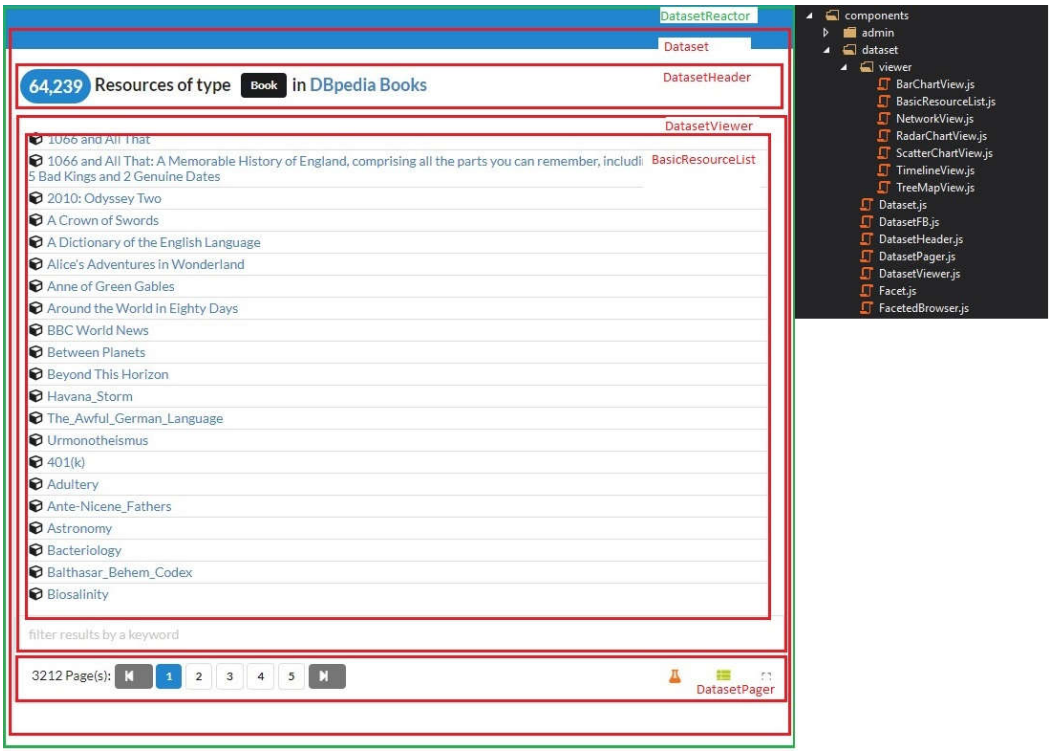
Στο συγκεκριμένο directory υλοποιούνται όλα τα Components της εφαρμογής. Πρόκειται για components που κάνουν extend τον βασικό React.Component. Στο directory αυτό υπάρχουν άλλα directories που υλοποιούν components για την λειτουργία της εφαρμογής και έχουν ομαδοποιηθεί σύμφωνα με την λειτουργία τους σε φακέλους. Κύρια directories τα οποία περιέχουν τους components κατηγοριοποιημένους και θα αναλύσουμε είναι οι dataset, resource, property, object και reactors.

Για αρχή υπάρχουν οι βασικοί components που έχουν υλοποιηθεί για να εμφανίζουν τις σελίδες της εφαρμογής. Στο top επίπεδο της εφαρμογής υπάρχει ο Application ο οποίος κατασκευάζει ένα element

που περιέχει τους components – React element **Nav** που υλοποιεί το Navigation bar της εφαρμογής και ένα **Handler** που τροποποιείται ανάλογα με το route της εφαρμογής όπως περιγράψαμε παραπάνω στο routes configuration αρχείο.

Στις καταστάσεις συστήματος της ομαδικής αναζήτησης και την λεπτομερούς ο Handler παίρνει τις τιμές των reactors **DatasetReactor** & **ResourceReactor** που είναι υλοποιημένοι στον φάκελο reactors μέσα στο directory των components. Εκτός από αυτούς τους δύο reactors έχουν υλοποιηθεί ο **PropertyReactor** και ο **ObjectReactor** οι οποίοι περικλείουν components που συντάσσουν τις ιδιότητες των αντικειμένων στη λεπτομερής παρουσίαση.

Ο **DatasetReactor** είναι ένας component ο οποίος περιέχει τις πληροφορίες που μεταβιβάζονται από το **DatasetStore** και τις μεταβιβάζει στο Component **Dataset** που είναι υλοποιημένος σε ένα directory dataset μέσα στο components. Αυτός είναι και ο κύριος components που αποτελεί τη λεπτομερή αναζήτηση. Στον ίδιο φάκελο υπάρχουν οι components που υλοποιούν τον **header** του που αποτελεί το την επικεφαλίδα της παρουσίασης δεδομένων στην αναζήτηση, τον **pager** που υλοποιεί το κάτω μέρος της κεντρικής αναζήτησης και ο **viewer** ο οποίος αναπαριστά το κύριο σώμα. Όπως είδαμε στο κεφάλαιο της αναζήτησης πλήθους υπάρχουν πολλοί τρόποι αναπαράστασης των δεδομένων που καθορίζονται από μια επιλογή που προέρχεται από μια λίστα επιλογής στον pager. Οι components που υλοποιούν αυτή την αναπαράσταση ορίζονται μέσα στον φάκελο viewer και είναι οι **BarChart**, **BasicResourceList**, **Network**, **ScatterChart**, **Timeline** και **TreeMap**. Το ποια από τις επιλογές θα γίνει ορατή στον χρήστη το καθιστά ο **DatasetViewer**.



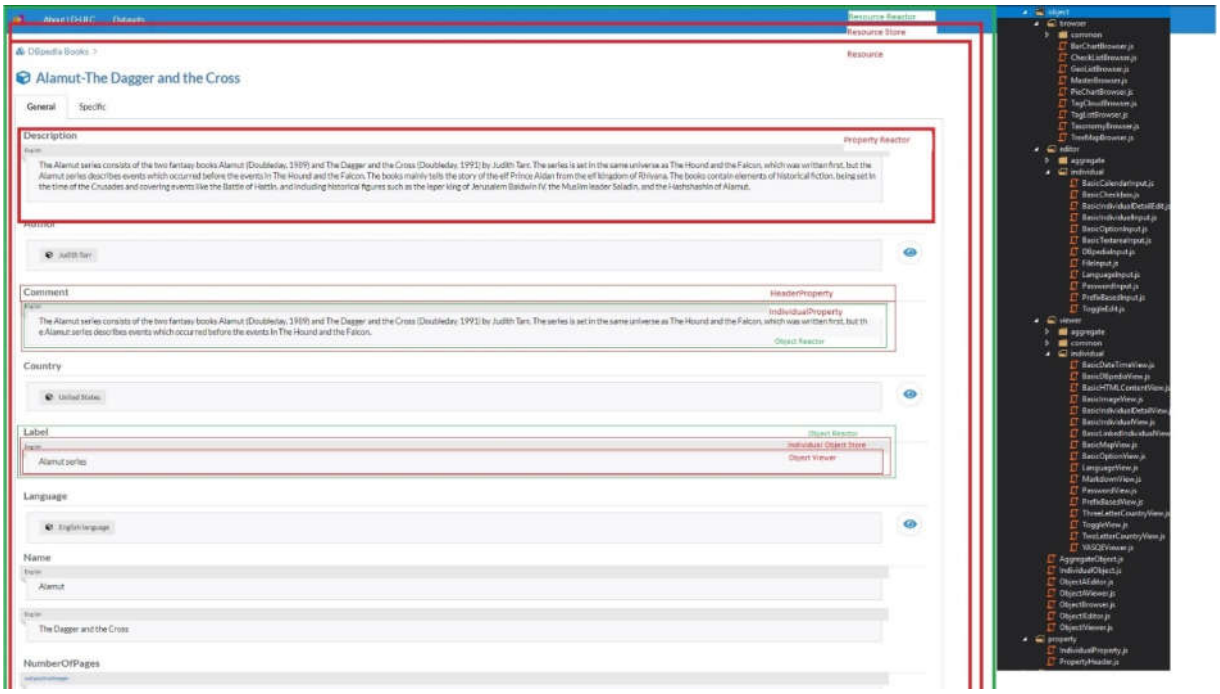
Εικόνα 33: Dataset Components

Ο **Resource Reactor** είναι ένας component ο οποίος περιέχει τις πληροφορίες που μεταβιβάζονται από το **ResourceStore** και τις μεταβιβάζει στο Component **Resource** που είναι υλοποιημένος σε ένα

directory resource μέσα στο components. Ο τελευταίος κατασκευάζει μια λίστα από Property Reactors και αν έχουν οι ιδιότητες κατηγορία από το resource configuration level κατασκευάζει tabs ανάλογα με τις κατηγορίες των δεδομένων. Ο Property Reactor υλοποιεί λειτουργίες όπως handle create, update, detail και delete με τη χρήση των actions που έχει κάνει import. Για το bind της πληροφορίας χρησιμοποιεί τους **IndividualProperty** και **HeaderProperty** στο directory.

Μέσα στο **IndividualProperty** υλοποιούνται λίστες με elements από τον component **ObjectReactor** που με τη σειρά του υλοποιεί ένα **IndividualObjectStore** το οποίο σχηματίζει τα πεδία με τους **ObjevtIEditor** και **ObjectIviewer** οι οποίοι βάση configuration χρησιμοποιούν components που έχουν υλοποιηθεί για να παρουσιάζουν τις τιμές σε επίπεδα browser, editor και viewer.

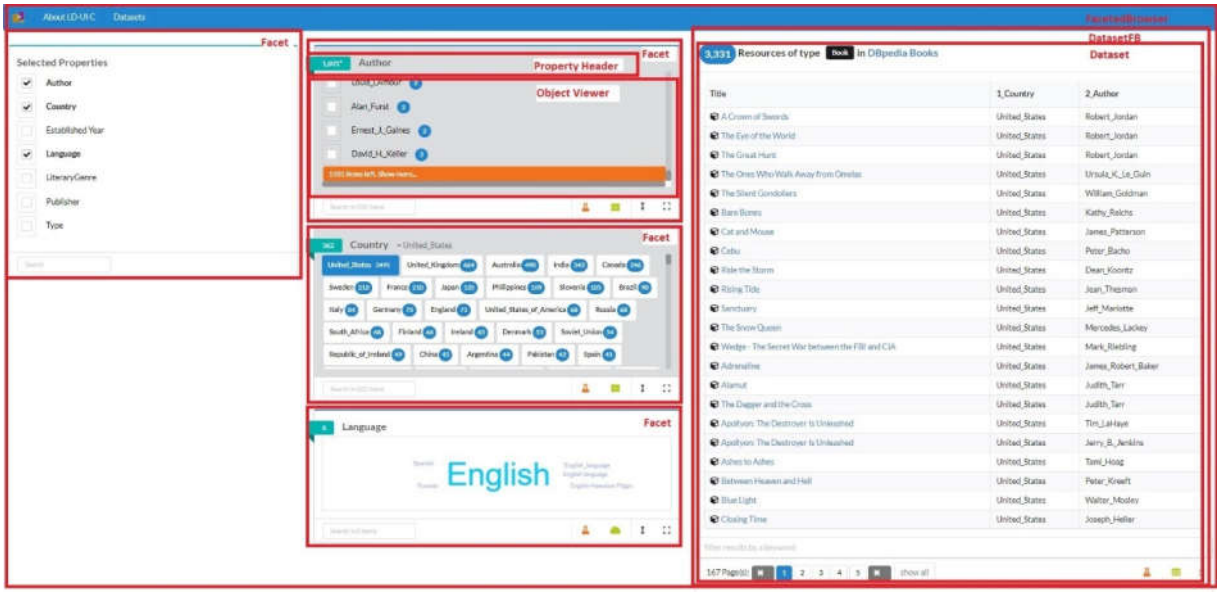
Οι components που ορίζονται σε φακέλους browser, editor, viewer μέσα στο directory "components/object/" υλοποιούν elements που διαχειρίζονται κάθε είδους τιμή και ανάλογα με το πώς έχουν οριστεί από το reactor.js για κάθε ιδιότητα επιλέγονται να σχηματίσουν την πληροφορία. Επειδή κάθε ιδιότητα που υλοποιεί τα διασυνδεδεμένα δεδομένα ποικίλει ανάλογα και για τις μορφές της επεξεργασίας της μπορεί να αντιστοιχεί και να διαχειρίζεται από διαφορετικά elements της react το πλήθος αυτών τον αρχείων είναι μεγάλο. Αναφορικά εδώ θα πούμε ότι για τον browser έχουν υλοποιηθεί barcharts, checklist, geolist, piechart, tagcloud, taglist, taxonomy, treemap, για τον editor, textbox, textarea, calendar, file, language, password, toggleedit και για τον viewer basicdate, html, image, individual, linkedindividual, map , option, language, markdown, password, two & tree country letter οι οποίοι αντιστοιχούνται ημιαυτόματα ανάλογα αλλά και με τροποποιήσεις.



Εικόνα 34: Detail View Components

Τέλος στο directory με τα datasets υπάρχει ο **FacetBrowser** ο οποίος είναι ένας components που στελεχώνει το κύριο σώμα της σελίδας λεπτομερούς αναζήτησης. Αυτός υλοποιεί τις τρεις στήλες οι οποίες είναι εμφανείς στη σελίδα αναζήτησης. Στην αριστερότερη υλοποιεί έναν **Facet** component που είναι μια λίστα δεδομένων από τις ιδιότητες του αντικειμένου που έχει συλλεγεί και παρουσιάζεται

στην αναζήτηση. Στην κεντρική στήλη χρησιμοποιεί μια λίστα από Facet components οι οποίοι εμφανίζονται κατ' επιλογήν από την αριστερότερη στήλη των ιδιοτήτων και στην δεξιά στήλη υλοποιεί τον **DatasetFB** που είναι ένα component παρόμοιος με αυτόν του **Dataset** με τις ίδιες ιδιότητες. Κάθε Facet αποτελείται από τρία μέρη. Στην επικεφαλίδα χρησιμοποιείται ένας Property Header ίδιος με αυτόν της λεπτομερούς κατάστασης, ένας ObjectViewer που παραμετροποιείται έτσι ώστε να εμφανίζει όλα τα δυνατά templates που υλοποιούνται για την κάθε οντότητα από το configuration file για προκαθορισμένη λειτουργία και αργότερα με την επιλογή από την λίστα του pager που αποτελεί και το τρίτο κομμάτι. Εδώ παρατηρούμε μια ιδιότητα κοινού χαρακτηριστικού καθώς το κάθε τι που παρουσιάζεται σαν αντικείμενο του facet είναι μια ιδιότητα αντικειμένου των διασυνδεδεμένων δεδομένων και μπορεί να ακολουθήσει την ίδια υλοποίηση με τις ιδιότητες που παρουσιάζονται στην λεπτομερή αναζήτηση και παρουσιάζουν πλήθος δεδομένων. Αυτό είναι και ένα σημείο διαφορετικής παραμετροποίησης στο configuration αρχείο και ορίζεται σαν χαρακτηριστικό που διαφέρει κατά το στρώμα παρουσίασης. Έτσι σιγά σιγά γίνεται αντιληπτή η ανάγκη κοινών χαρακτηριστικών και ο διαφορετικός τρόπος χειρισμού.



Εικόνα 35: Browser Components

- Services

Στο φάκελο των services είναι υλοποιημένα αντικείμενα της javascript τα οποία είναι υπεύθυνα για την αλληλοεπίδραση του περιβάλλοντος με εξωτερικά end points. Τα services που υλοποιούν λειτουργίες των διασυνδεδεμένων δεδομένων είναι αυτά του **datasetservice** που ορίζει μια λειτουργία read, του **resourceservice** που υλοποιεί λειτουργίες του read, create, update και delete, του **facetservice** που υλοποιεί και αυτό με τη σειρά του μια read καθώς και άλλα services που διαχειρίζονται λειτουργίες του administrator, του import και του dbpedia για λειτουργίες lookup. Αυτές οι κλάσεις λειτουργούν με το πακέτο του request-promise που έχει συμπεριληφθεί στην εφαρμογή και πραγματοποιούν τις ασύγχρονες κλήσεις. Κατά την απάντηση τα δεδομένα πραγματοποιούν μια σειρά από διαδικασίες επεξεργασίας που σε συνδυασμό με τα configuration file τροποποιούν τα δεδομένα έτσι ώστε αυτά να αποκτήσουν μια σειρά από ιδιότητες που είναι χρήσιμες για την αναπαράστασή τους από τους components που τις εμφανίζουν.

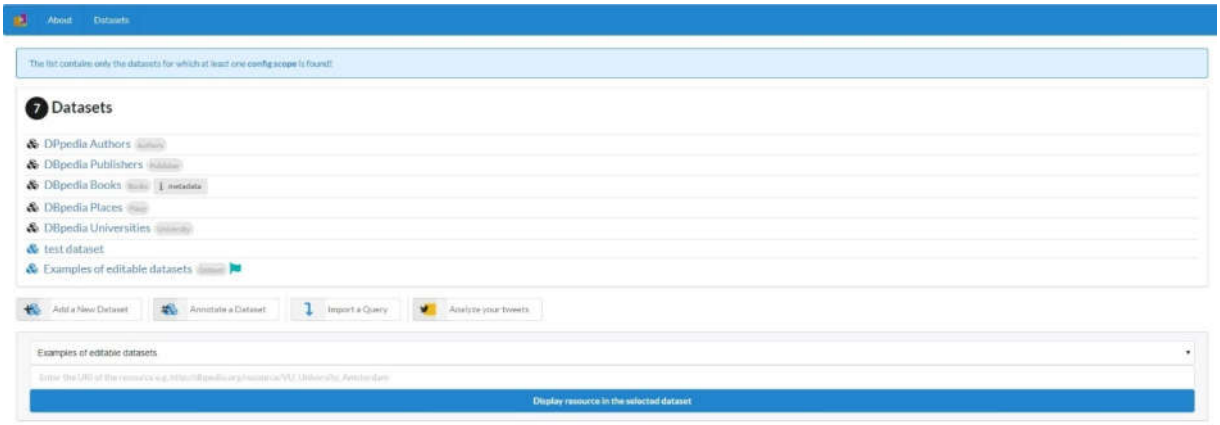
Αυτές οι διαδικασίες επεξεργασίας υλοποιούνται από κλάσεις με συναρτήσεις που ορίζονται στον φάκελο utils της εφαρμογής μέσα στο directory των services. Κλάσεις με παρόμοια ονόματα υλοποιούνται για να χρησιμοποιούνται από τα αντίστοιχα services καθώς και μια dynamicHelper η οποία μέσω ενός config.ttl σχηματίζει ένα γράφο δεδομένων ψευδοαντικειμένων που ορίζονται για τις απαιτήσεις της εφαρμογής στην επεξεργασία δυναμικού τύπου δεδομένων. Αυτός ο γράφος ορίζει στοιχεία που αποτελούν αντικείμενα όπως το configuration και χρησιμοποιείται για να μπορεί ο χρήστης να εξάγει τα configuration σε οντότητες τύπων που χρησιμοποιεί η εφαρμογή μας σαν τριπλέτες αναγνωρισμένες από ένα triplestore.

Τέλος κάτω από το ίδιο directory αυτό των services βρίσκεται ένας φάκελος ο οποίος ονομάζεται SPARQL και υλοποιεί κλάσεις των DatasetQuery , FacetQuery , ResourceQuery , DBPedia και Admin οι οποίες περιέχουν συναρτήσεις που δέχονται σαν παραμέτρους το graphname, το resourceURI, propertyURI, τιμές και τύπους αντικειμένων και κατασκευάζουν τα ερωτήματα SPARQL σε μορφή strings που αργότερα ενσωματώνονται από τα services σε URIs τα οποία στοχεύουν στο endpoint της DBPedia.

4.4.2 Περιγραφή λειτουργίας δομικών στοιχείων και Ενεργειών

Μετά από την περιγραφή των δομικών στοιχείων της εφαρμογής είμαστε σε θέση να περιγράψουμε την λειτουργεία των κύριων σελίδων και την λειτουργεία που προσφέρουν στη δημοσιοποίηση των διασυνδεδεμένων δεδομένων και στον τρόπο με τον οποίο η εφαρμογή δρα ανεξάρτητα για κάθε οντότητα. Ο κύκλος που αναφέρθηκε σχετικά με την λειτουργεία της flux αρχιτεκτονικής καθιστά την εφαρμογή μας τόσο λειτουργική όσον αναφορά τα δομικά στοιχεία που αναφέραμε παραπάνω και επίσης ικανή να διαχειρίζεται τα δομικά στοιχεία που την αποτελούν με μια ανεξαρτησία και λειτουργικότητα που μπορεί εύκολα να επεκταθεί. Κάθε κατάσταση του συστήματος αντιστοιχίζεται σε μια διαπροσωπεία η οποία δομείται από ένα Store και περιλαμβάνει τους δικούς της components που ανανεώνουν το σύνολο των δεδομένων τους με events.

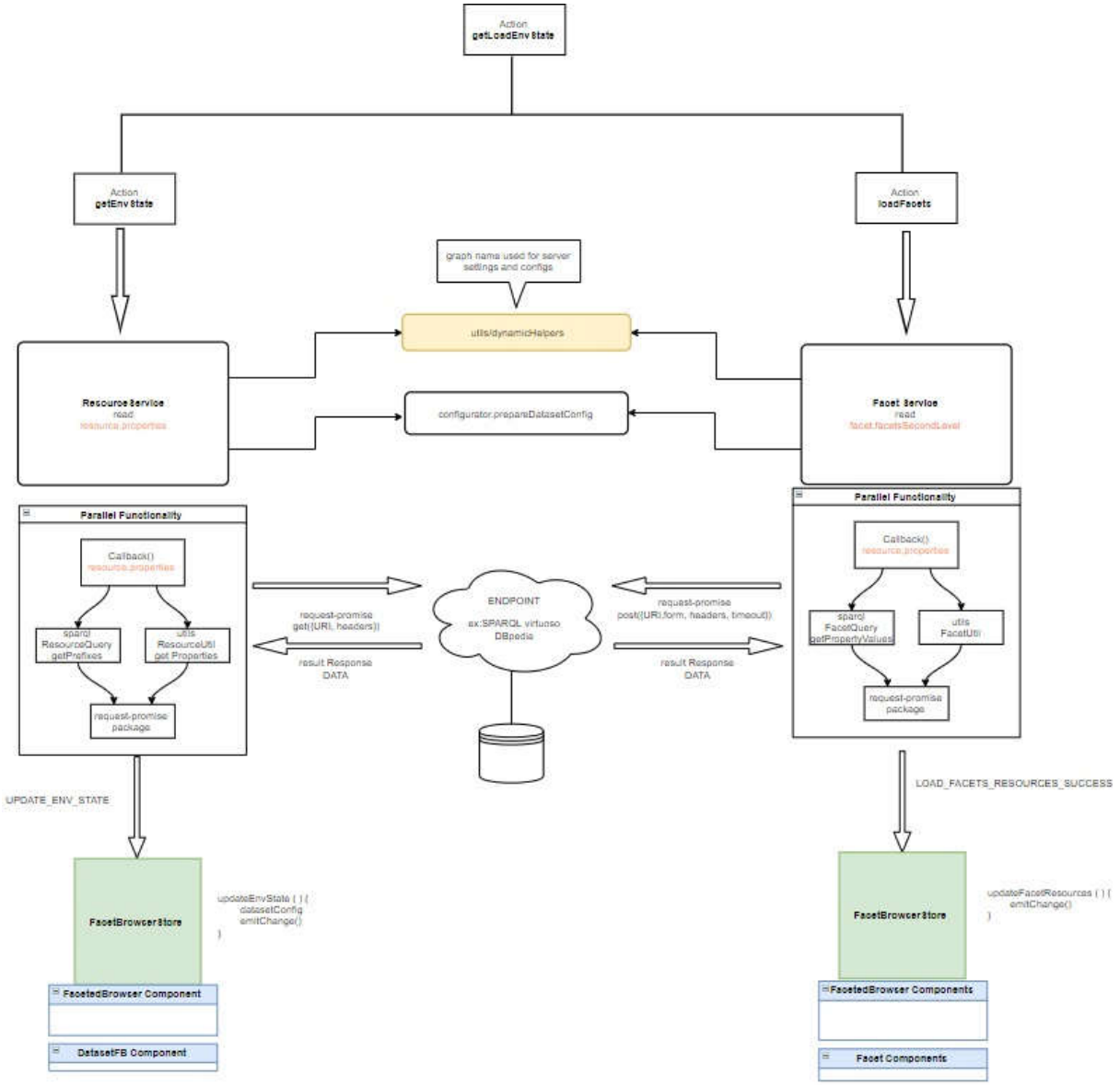
Για αρχή ο χρήστης μπορεί να επιλέξει μεταξύ διαφόρων datasets που έχουν προκαθοριστεί από τον προγραμματιστή ή τον administrator της εφαρμογής και να αντλήσει πληροφορίες. Η οθόνη αυτή όπως φαίνεται στην παρακάτω εικόνα είναι η οθόνη των datasets. Αναφέρονται όσα από αυτά έχουν οριστεί στο reactor.js configuration αρχείο. Το συγκεκριμένο αρχείο να αναφέρουμε εδώ ότι τροποποιείται κατά τις διαδικασίες του add new dataset με επιπλέον resources τα οποία κατά την διαδικασία εισαγωγής επιλέγονται και προστίθενται μαζί με τις ιδιότητές τους, όπως και οι components οι οποίοι θα τα σχεδιάσουν από μια σελίδα administration.



Εικόνα 36: Dataset Selections View

Στην οθόνη φαίνονται οι ετικέτες που έχουν καθοριστεί κατά την διαδικασία του import και σαν URI ανακατεύθυνσης, μια κατάσταση του route configuration που αποτελείται από το ID που έχει δώσει η εφαρμογή στο συγκεκριμένο dataset και το resource το οποίο έχει οριστεί στο config αρχείο. Έτσι το Route Store της εφαρμογής παίρνει τις παραμέτρους και σχηματίζει ένα URI της μορφής "/browse/:id?/:stateURI?" καλώντας μια συνάρτηση executeAction του Store και ανοίγοντας μια νέα καρτέλα που υλοποιεί τον FacetBrowser.

Ο Component του FacetBrowser όπως αναφέρθηκε παραπάνω είναι μια κατάσταση του συστήματος που περιέχει τρεις κολώνες στα αριστερά υλοποιεί ένα facet το οποίο περιέχει όλα τα properties του συνδέσμου που έχουν οριστεί στο reactor.js σαν ιδιότητες του συγκεκριμένου συνδέσμου και εμφανίζει μια λίστα με τις επιλογές αυτές στον χρήστη με ιδιότητα των επιλεγμένων από αυτούς σαν facets που παρουσιάζουν πληροφορία στην μεσαία στήλη. Κατά την επιλογή της κατάστασης του συστήματος ένα action ενεργοποιείται με την ονομασία getLoadEnvState (get load environment State) το οποίο ενεργοποιεί δύο άλλα actions στο σύστημα αυτά τον getEnvState & loadFacets. Το κάθε ένα χρησιμοποιεί τα services που είναι υλοποιημένα για να πραγματοποιήσει τα ερωτήματα στο web για την ανάκτηση της πληροφορίας. Πριν την κλήση το service χρησιμοποιεί το URI που έχει κατασκευαστεί και με την βοήθεια του dynamicHelper, ο οποίος είναι μια συνάρτηση που αντιστοιχεί στο Logic της εφαρμογής, και κατασκευάζει τις προκαθορισμένες τιμές για την κλήση. Αναλυτικά σχηματίζει το με τη βοήθεια του general configuration file τον host , port , path , protocol , username & password επιστρέφοντας τα αντίστοιχα αντικείμενα σε μια μέθοδο, αυτή του configurator. Η αντίστοιχη μέθοδος του configurator διαβάζει το αρχείο configuration reactor και σύμφωνα με την προκαθορισμένη λειτουργία ή με επιλογές που έχει ορίσει ο προγραμματιστής ή ο διαχειριστής της εφαρμογής εφαρμόζει ανάλογα με τις τιμές που έχουν δοθεί σαν παραμέτρους το αντίστοιχο ερώτημα και τον γράφο τον δεδομένων. Στην συνέχεια σαν αποτέλεσμα αυτής της διαδικασίας υλοποιείται με παραμέτρους η κλήση Get όπως φαίνεται στο παρακάτω process στο τελικό σημείο που έχει οριστεί. Η ασύγχρονη κλήση με το αποτέλεσμά της και τις μεταβλητές που έχουν κατασκευαστεί από τον Configurator μεταβιβάζονται με το αντίστοιχο μήνυμα UPDATE_ENV_STATE στο Store.

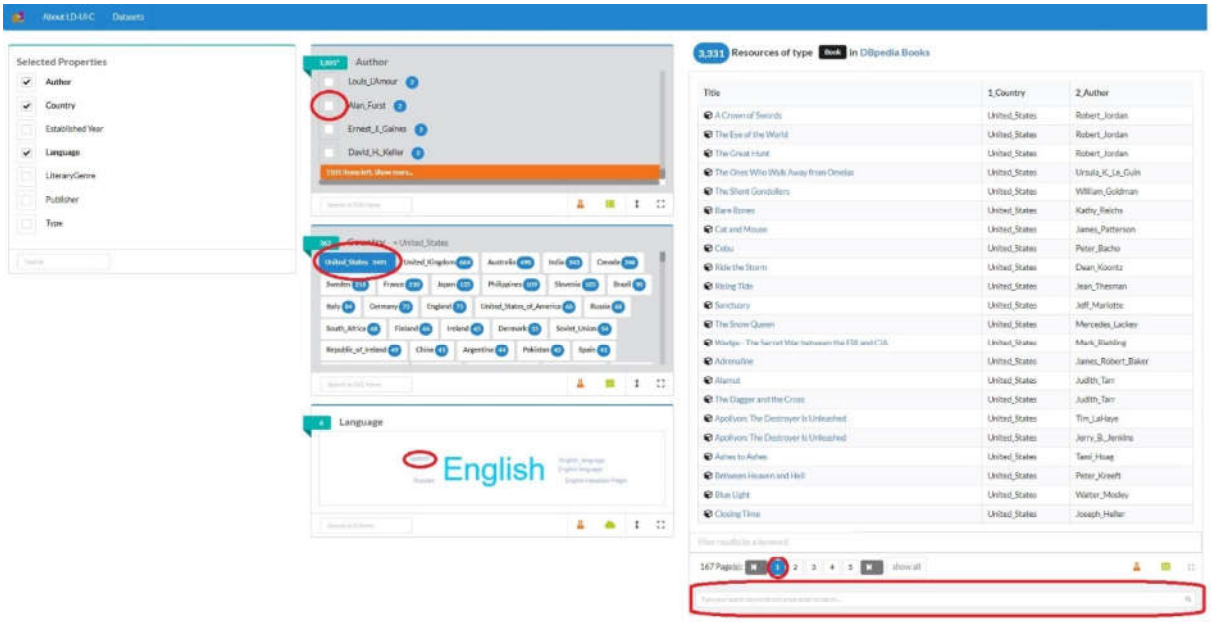


Διάγραμμα 9: Load Environment Process

Στο συγκεκριμένο μήνυμα υπακούει μια συνάρτηση που μεταβιβάζει τα δεδομένα και τις αλλαγές στο DatasetFB Component για να παρουσιάσει τα δεδομένα. Οι ιδιότητες και οι τιμές του αντικειμένου που επεξεργάζεται ο συγκεκριμένος component περνούν στους components που υλοποιεί με την μονής κατεύθυνσης ροή δεδομένων του flux. Οι αντίστοιχοι components που υλοποιούνται από τον συγκεκριμένο περιχυτή είναι όπως αναφέραμε και στο προηγούμενο κεφάλαιο η επικεφαλίδα που διατηρεί πληροφορίες του dataset, το κύριο σώμα που εμφανίζει τα δεδομένα και ο pager που απαρτίζει την σελιδοποίηση. Η επιλογή του αντίστοιχου template για την εμφάνιση των αποτελεσμάτων αποτελεί την πληροφορία του config που μεταβιβάζεται από τον configurator.

Μια παρόμοια λειτουργία είναι αυτή της συγκομιδής της πληροφορίας που υλοποιείται από την διαδικασία του event load Facets. Αυτή τη φορά χρησιμοποιείται ένα άλλο service αυτό των facets.

Παρόμοια διαδικασία ακολουθείται και σε αυτήν την υπηρεσία όπου με την βοήθεια του dynamic helper και του configurator σχηματίζεται ένα αντικείμενο config από το αρχείο config facets.js. Ο παραμετροποιητής αυτή τη φορά κατασκευάζει ένα διαφορετικό ερώτημα και ένα αντικείμενο με δείκτες τα resources του κάθε property της οντότητας ώστε κάθε facet στην συνέχεια βάση του property που υλοποιεί να αλλάζει ξεχωριστά τα δεδομένα του. Κατά την επιστροφή των δεδομένων ένα event με όνομα LOAD_FACETS_RESOURCES_SUCCESS αντικατοπτρίζει της αλλαγές σε μια συνάρτηση του FacetBrowser Component ο οποίος με το αντικείμενο που έχει σχηματιστεί από τον configurator και τα αποτελέσματα αλλάζει το περιεχόμενο των facets που υλοποιεί. Η τελική εικόνα της σελίδας μετά την εφαρμογή των πεδίων παραμετροποίησης και των δεδομένων είναι αυτή της ομαδικής αναζήτησης όπως φαίνεται παρακάτω.



Εικόνα 37: Browser Search Action Events

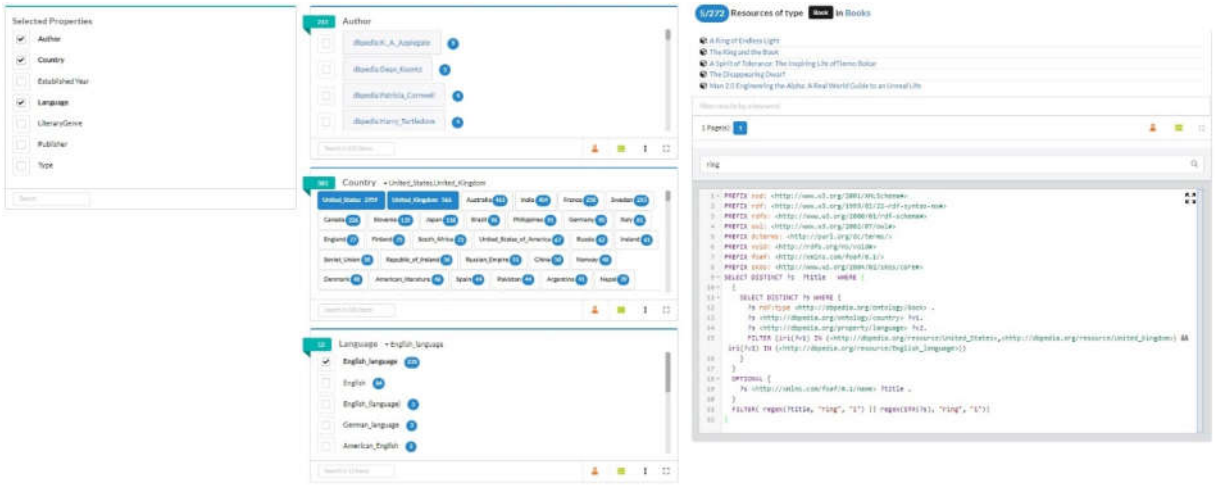
Στην εικόνα σηματοδοτούνται με κόκκινες ενδείξεις παραδείγματα από events στο User Interface που ενδέχεται να ενεργοποιήσουν νέους κύκλους περισυλλογής πληροφοριών στο συγκεκριμένο context της εφαρμογής. Κατά την επιλογή δεδομένων από τα facets όπως αυτά φαίνονται στην μεσαία στήλη και events όπως του Invert selection ένας νέος κύκλος με το action load facets πραγματοποιείται με τις επιλεγμένες οντότητες αυτή την φορά να αποτελούν φίλτρα που κατά την διαδικασία παρασκευής του ερωτήματος μεταφράζονται σε SPARQL strings με την ιδιότητα Filter και γίνονται append στο where cause της ερώτησης σύμφωνα με το property μου μεταφέρεται από τις επιλεγμένες ιδιότητες. Σε αυτή την περίπτωση το service υλοποιεί την λειτουργία του με την επιλογή side effects. Κατά την επιστροφή των αποτελεσμάτων που φορτώνονται και απασχολούν τα facets τα resources που επιστρέφονται ανανεώνονται στο context της συνάρτησης που είναι συνδεδεμένο με τη λίστα αναζήτησης.

Στη δεξιά στήλη της παρουσίασης των resources που έχουν φορτωθεί σαν αποτέλεσμα της ερώτησης στο τελικό σημείο δύο events κυριαρχούν. Αυτό της επιλογής σελίδας και αυτό της αναζήτησης με λεκτικό. Κατά την αναζήτηση με λεκτικό ένα νέο action ενεργοποιείται με όνομα searchInDataset και λειτουργία παρόμοια με αυτή του handler με το όνομα LOAD_FACETS_RESOURCES_SUCCESS. Ένας

κύκλος εργασίας και συγκομιδής πραγματοποιείται με αποτέλεσμα την ίδια λειτουργία που περιγράψαμε. Στη σελιδοποίηση μια αντίστοιχη συνάρτηση έχει γίνει bind από τον FacetBrowser η οποία ενεργοποιεί μια διαδικασία παρόμοια με αυτή του load facet με mode second η οποία στον handler έχει ακριβώς την ίδια λειτουργία με κωδικό επιστροφής event όπως περιεγράφηκε παραπάνω.

Με την υλοποίηση των παραπάνω παρατηρούμε ότι όλα τα events μπορούν να υλοποιηθούν από διαφορετικά actions που μπορούν από παραμετροποίηση των handlers πότε να αλλάξουν και πότε να κρατήσουν την ίδια λειτουργία σε ένα πεδίο εφαρμογής. Αυτό αποτελεί ένα κομμάτι υπολογιστικής λογικής το οποίο προσφέρει ιδιότητες στα δομικά στοιχεία που αποτελούν την λειτουργία της εφαρμογής ανεξάρτητα αλλά και συλλογικά. Το context της εφαρμογής υλοποιείται κυρίως από τον FacetBrowser το οποίο διατηρεί όλα τα δεδομένα που αντιστοιχούν στην κατάσταση και τα διαμοιράζει σε components οι οποίοι λειτουργούν με τα δικά τους events και λογική που μπορεί να έχει πρόσβαση σε ευρύτερα κομμάτια αλλά και στο καθολικό context της διαπροσωπείας μέσω των handlers.

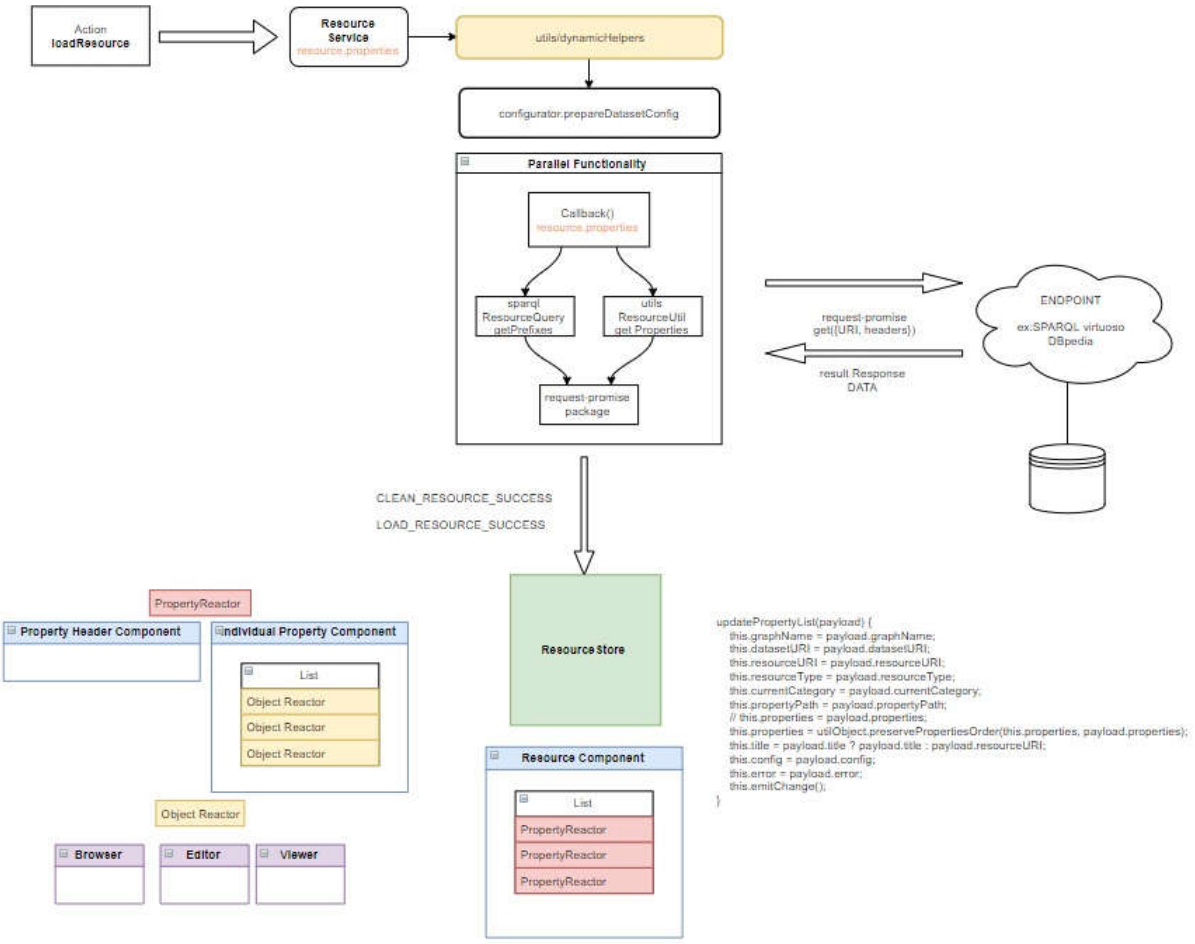
Ένα αποτέλεσμα μιας προκαθορισμένης ερώτησης που αποδίδεται από τον χειρισμό του interface είναι αυτό της παρακάτω εικόνας, όπου έχουμε ορίσει σαν headers βασικά prefixes και από τα facets έχουμε επιλέξει δύο χώρες αυτές των Ηνωμένων Πολιτειών και του Ηνωμένου Βασιλείου και σαν Optional οντότητα το λεκτικό που έχουμε ορίσει να περιέχει τον τίτλο του βιβλίου με ένα τιμή «Ring».



Εικόνα 38: Browse results Query

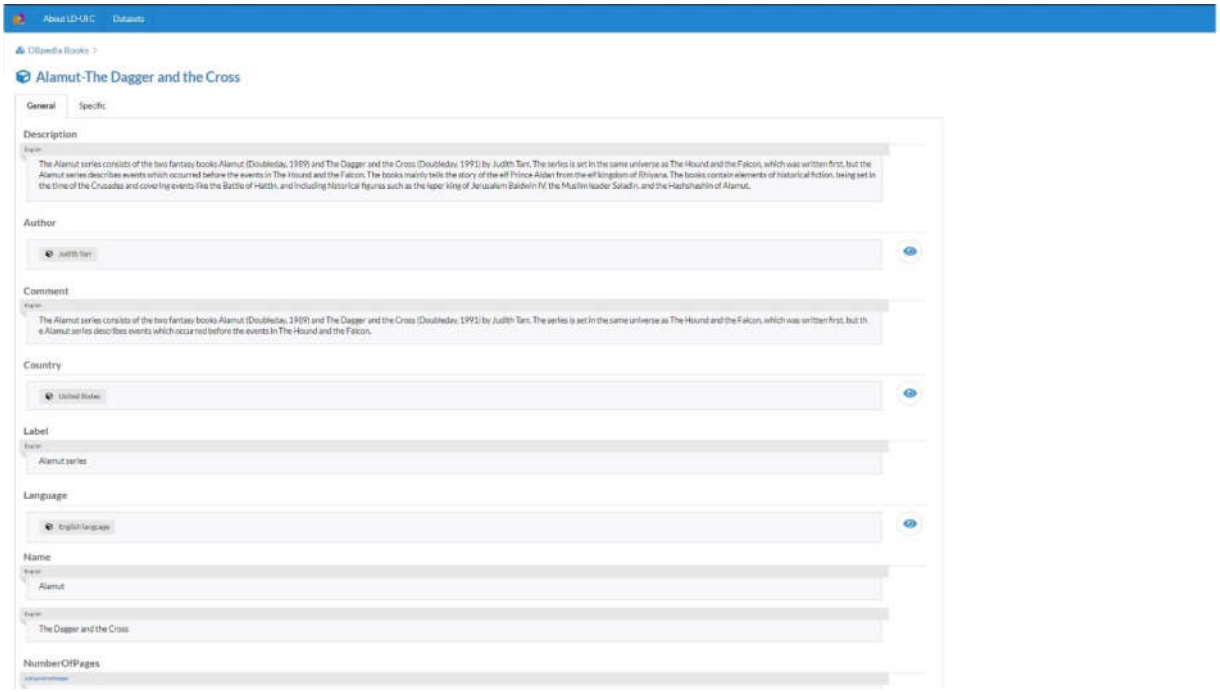
Στη συνέχεια κατά την επιλογή ενός resource από τα αποτελέσματα της αναζήτησης ένα event ανακατεύθυνσης πραγματοποιείται στη κατάσταση resource του συστήματος. Αυτή από τον route handler ενεργοποιεί ένα action αυτό του loadResource το οποίο με μια παρόμοια διαδικασία δημιουργεί ένα νέο tab με URI τις παραμέτρους του dataset και του resource. Το Action αυτό καλεί το Service του Resource το οποίο με την βοήθεια του dynamic helper και του Configurator κάνει χρήση του sparql/ResourceQuery και του /util/ResourceUtil στη συνάρτηση callback του Configurator. Το query με τις παραμέτρους που δέχεται από την ανακατεύθυνση παράγει τον γράφο και το ερώτημα που χρειάζεται για να σταλεί στο τελικό σημείο. Με την επιστροφή της κλίσης η διαδικασία του utilization παίρνει θέση και ένα αντικείμενο μεταβιβάζεται στο callback της συνάρτησης. Το Resource store στη συνέχεια δέχεται το callback και εκτελεί την συνάρτηση που αντιστοιχεί στο name του handler 'LOAD_RESOURCE_SUCCESS': 'updatePropertyList'. Τα δεδομένα καθώς και τα configuration που έχουν σχηματιστεί μεταβιβάζονται στον Resource Component που υλοποιεί μια λίστα από Property Reactors.

Αυτοί με την σειρά τους περιέχουν στο render δύο components όπως περιγράψαμε και παραπάνω. Ο Property Header αναλαμβάνει να σχηματίσει την επικεφαλίδα με την ιδιότητα που παρουσιάζει και ο Individual Component το κύριο σώμα. Αυτός σύμφωνα με το configuration αρχείο και την κατάσταση στην οποία βρίσκεται το κάθε resource property, καθώς ο χειριστής μπορεί να έχει επιλέξει να κάνει edit τη συγκεκριμένη ιδιότητα, αναλαμβάνει από το config object που του έχει σταλεί να σχηματίσει το τελικό template και θα του αποτιμηθεί η τιμή ανάλογα με την κατάσταση και με τον component που του έχει οριστεί.



Διάγραμμα 10: Detail View Actions Flow

Η τελική εικόνα του view είναι αυτή της λεπτομερής παρουσίασης που εμφανίζει σε κάθε tab τα properties του αντικειμένου και τις τιμές τους ανάλογα με το πώς αυτά έχουν οριστεί από το αρχείο configuration reactor.js. Σε κάθε property του αντικειμένου εκτός από το δομικό στοιχείο που αναλαμβάνει να το σχεδιάσει στις καταστάσεις Browse, Edit, View ορίζεται και μια κατηγορία η οποία αντιστοιχεί σε ποιο tab θα εμφανιστεί.



Εικόνα 39: Detail View Example

Στη σελίδα της λεπτομερούς αναζήτησης όπως περιγράψαμε και σε ενότητα παραπάνω, αν η τιμή του property που περιγράφεται αποτελεί αντικείμενο επεξεργασίας εμφανίζεται στα αριστερά η ένδειξη expand η οποία περισυλλέγει τα δεδομένα που αποτελούν και τα εμφανίζει αναλυτικά. Η ιδιότητα του showDetails υλοποιείται με το event loadObjectProperties. Αυτό ενεργοποιεί ένα κύκλο εργασιών του resource service με την ίδια ακριβώς διαδικασία όπως και με την φόρτωση του resource της σελίδας. Κατά την επιστροφή της κλήσης το Individual Object Store είναι υπεύθυνο να ανανεώσει το περιεχόμενο των δεδομένων με το LOAD_OBJECT_PROPERTIES_SUCCESS το οποίο με το URI που έχει σχηματίσει την ερώτηση ανανεώνει τα properties της σελίδας με αυτό το κλειδί. Κατά αυτόν τον τρόπο ανανεώνεται μόνο το συγκεκριμένο component που υλοποιεί την ιδιότητα χωρίς να αλλάζει το περιεχόμενο της σελίδας. Παρόμοιες διαδικασίες υλοποιούνται και κατά τις διαδικασίες edit και delete με αντίκτυπο το store του Individual Object χωρίς να αλλάζει το περιεχόμενο της σελίδας ολοκληρωτικά. Σε αυτές τις περιπτώσεις αντίστοιχα SPARQL ερωτήματα υλοποιούνται και στο callback της συνάρτησης του service ένα event ενεργοποιείται στον αντίστοιχο handler. Εδώ παρατηρούμε μια ουσιαστική λειτουργία της flux αρχιτεκτονικής στην οποία με την διάσπαση του context της κατάστασης σε μικρότερα που ενεργοποιούνται οι λειτουργίες τους από τους handlers έχουν τη δυνατότητα να επεξεργάζονται μικρότερα κομμάτια της εφαρμογής ανεξάρτητα, χωρίς να απαιτείται πρόσμιξη των διαδικασιών. Αυτό είναι ένα σημαντικό πλεονέκτημα αλλά για να υλοποιηθεί μια τέτοια προγραμματιστική προσέγγιση πρέπει να υπάρχει ο κατάλληλος σχεδιασμός.

Για να κλείσουμε τη διαδικασία της περιγραφής της λειτουργίας, θα παρουσιάσουμε τώρα τη διαδικασία με την οποία ένας administrator χρήστης της εφαρμογής μπορεί να εισάγει ένα καινούργιο dataset στην εφαρμογή και να εφαρμόσει νέα resources μέσα σε αυτό. Στη σελίδα της επιλογής των dataset για την εφαρμογή υπάρχει η επιλογή Add a New Dataset.

Εικόνα 40: Dataset Actions

Με την επιλογή αυτή για την εισαγωγή ενός νέου dataset ο χρήστης μεταβαίνει σε μια κατάσταση του συστήματος η οποία φαίνεται παρακάτω και παροτρύνεται να εισάγει μια ετικέτα σχετικά με την οποία θα περιγράφεται το dataset που επιθυμεί να σχηματίσει καθώς και ένα URL με το αντίστοιχο endpoint το οποίο θα είναι υπεύθυνο να προσκομίσει τις πληροφορίες όταν δημιουργηθούν ερωτήματα. Μια σχετική εικόνα φαίνεται παρακάτω στην οποία ο χρήστης έχει και την δυνατότητα να εισάγει τα δεδομένα από ένα csv αρχείο. Κατά την εφαρμογή και την ολοκλήρωση των απαραίτητων πεδίων ή μέσω της εισαγωγής, ένας νέος αριθμός κλειδί δημιουργείται για τον ορισμό του γράφου που αποτελεί μοναδικό αναγνωριστικό μαζί με την ετικέτα που έχει ορίσει ο χρήστης. Εάν το URL παραμείνει κενό ένα action «createEmptyDataset» καλείται αλλιώς το «createFromExistingDataset». Κατά τη διαδικασία εισαγωγής νέου dataset, νέες καταχωρίσεις προστίθενται στο αρχείο reactor.js που αποτελεί το βασικό configuration της εφαρμογής. Με τη σειρά του ο dynamicHelper διαβάζει το URL του endpoint που έχει προκαθοριστεί από το server config αρχείο για το τοπικό triplestore ή επιλέγεται αυτό που έχει προκαθορίσει ο χρήστης. Στη συνέχεια παρασκευάζεται ένα SPARQL ερώτημα «INSERT» και υποβάλλεται στο συγκεκριμένο resource. Κατά την επιτυχία ένα event ανακατεύθυνσης ενεργοποιείται στη σελίδα της ομαδικής αναζήτησης.

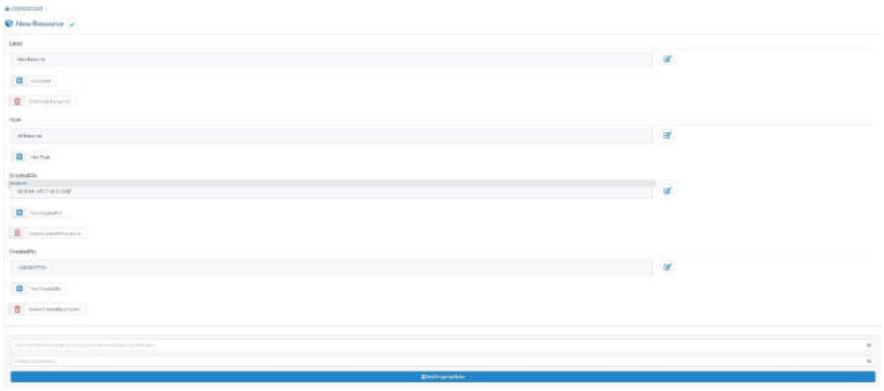
Εικόνα 41: Import Dataset Option

Το επόμενο στάδιο είναι αυτό της ομαδικής αναζήτησης σε λειτουργία edit. Αυτό παρουσιάζει τα resources τα οποία έχουν συνδεθεί με το συγκεκριμένο dataset, και δίνει την ιδιότητα στον χρήστη να εισάγει νέα resources με την επιλογή που φαίνεται κάτω από τον pager της συνάρτησης. Να πούμε εδώ ότι η λειτουργία αυτή υλοποιείται και στα dataset που έχει ήδη περισυλλέξει με τίς επιλογές readOnly:false και AllowNewValue :true.

Εικόνα 42: Import resource Option

Με την επιλογή του add new Resource ο χρήστης μεταβαίνει στην κατάσταση της λεπτομερούς αναζήτησης στην οποία τα πεδία που υπάρχουν ήδη είναι τροποποιημένα να εμφανίζονται σε λειτουργία που υλοποιεί το edit και έχουν καθοριστεί σε ένα dump.ttl. Σε κάθε property αντικειμένου

είναι ενεργές οι επιλογές της δημιουργίας νέας τιμής κάτω από κάθε συγκεκριμένη ιδιότητα και επεξεργασία όπως φένεται στα αριστερά κάθε καταχώρησης.



Εικόνα 43: Edit resource Options

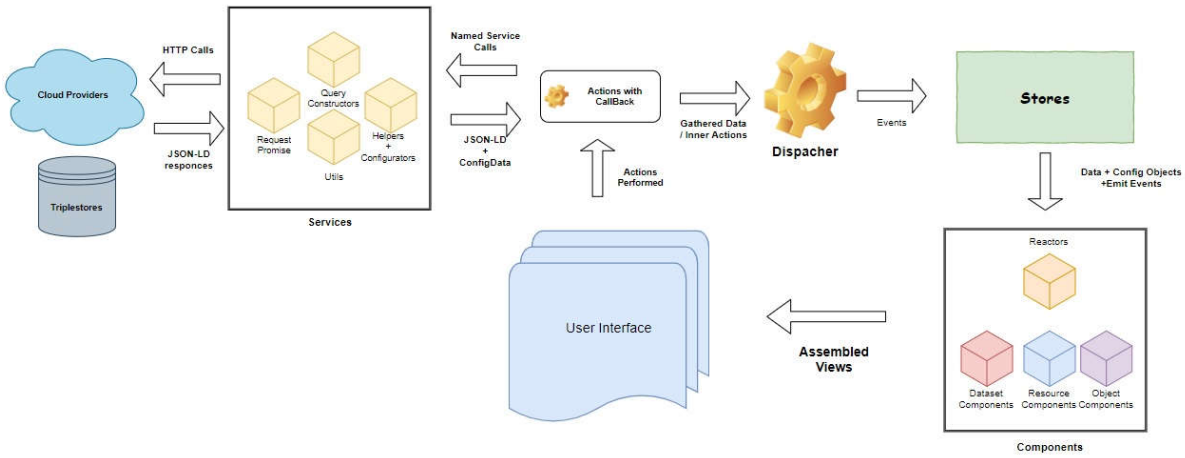
Το πλήκτρο της επεξεργασίας αλλάζει την τιμή που αναπαραριστά το individual Object και από την κατάσταση view ξεκινάει μια κατάσταση edit με τον αντίστοιχο ή τον προκαθορισμένο αν δεν έχει οριστεί editor να τη θέση του component. Κάτω ακριβώς από κάθε ιδιότητα εμφανίζεται μια επιλογή στον χρήστη για την διαγραφή ολόκληρου του property. Τέλος στο κάτω μέρος της σελίδας δίνεται το δικαίωμα στον χρήστη να προσθέσει ένα καινούργιο property καθορίζοντας δύο πεδία. Αυτά είναι το url του property που επιθυμεί να εισάγει και το value που του αντιστοιχεί. Με αυτόν τον τρόπο ο χρήστης μπορεί να χειριστεί και να δημιουργήσει νέα διασυνδεδεμένα δεδομένα σε κόμβο του διαδικτύου που έχει ορίσει κατά την εισαγωγή του dataset. Για τις λειτουργίες αυτής της διαδικασίας και των προκαθορισμένων τιμών που εμφανίζει το σύστημα έχουμε αναπτύξει ένα dump.ttl αρχείο με προκαθορισμένες τιμές που περιγράφει σε turtle γλώσσα ιδιότητες που αντιστοιχούν σε διασυνδεδεμένα δεδομένα κλάσεων και properties που έχουν υλοποιηθεί από τη schema.org. Κατά την εισαγωγή νέου dataset και των resources σε αυτόν παράγεται ένα JSON-LD αρχείο το οποίο μπορεί να αποτελέσει μέθοδο εισαγωγής σε ένα triplestore καθώς και ένα ttl αρχείο για να περιγράψει τον γράφο δεδομένων.

Με τη σύνδεση σε ένα πραγματικό endpoint οι συναρτήσεις που υλοποιούν τα sparql ερωτήματα για εισαγωγή και διαγραφή και έχουν υλοποιηθεί να αλληλεπιδρούν με το virtuoso της DBpedia μπορούν να χρησιμοποιηθουν για αντίστοιχες διαδικασίες. Αυτό όμως αφήνεται σαν επέκταση της εφαρμογής λόγω του ότι η διαδικασία της υλοποίησης αποτελεί καθαρά το front-end κομμάτι χωρίς να υπάρχει η σύνδεση με κάποιο διαχειρίσιμο triplestore. Παόλα αυτά με τη χρήση του vocabulary.ttl και του mappings.ttl που κάνει χρήση η εφαρμογή οι διαδικασίες επεξεργασίας προσθήκης και διαγραφής στοιχείων που αποτελούν ιδιότητες και τιμές παράγουν ερωτήματα με ένα resource που ορίζεται στο general configuration αρχείο και έχουν την μορφή http//:myresource.

Από την άλλη όταν ο χρήστης κατά την επιλογή τροποποίησης ενός υπάρχων dataset που αντιστοιχεί σε σύνδεσμο ενεργό με παρόμοιες διαδικασίες στα στρώματα της εφαρμογής παράγει SPARQL ερωτήματα που υποβάλονται σε πραγματικό χρόνο στο triplestore που έχει αντιστοιχιστεί υλοποιόντας διαδικασίες προσθήκης και διαγραφής.

4.4.3 Τυποποίηση διαδικασίας επεξεργασίας διασυνδεδεμένων δεδομένων

Σχετικά με όσα αναφέραμε για τις ροές εργασιών και τη μονή κατεύθυνσης ροή δεδομένων μπορούμε να διαχωρίσουμε τις παρακάτω ενότητες στην υλοποίησή μας. Στο κύριο σώμα της εφαρμογής ο χρήστης μπορεί από το User Interface να αλληλοεπιδρά με δεδομένα που του διατίθενται από την εφαρμογή. Τα στρώματα παρουσίασης με την υλοποίηση του React Framework και του Flexible-UI προσφέρουν διαδικασίες rendering με γραφικά φιλικά προς τον χρήστη και προκαθορισμένα components που υλοποιούν βασικούς χαρακτηρισμούς (bars, date pickers, switches etc). Στις σελίδες που παρουσιάζονται στον χρήστη δίνεται η ιδιότητα της αλληλεπίδρασης με ενέργειες που έχουν υλοποιηθεί και αποτελούν ένα ξεχωριστό κομμάτι της εφαρμογής. Αναφέρουμε το κομμάτι των actions σαν μια οντότητα που μπορεί να λειτουργήσει αυτόνομα και να προσαρτηθεί σε διαφορετικά κομμάτια της εφαρμογής καθώς κάθε action δέχεται το context της εφαρμογής, ένα payload δεδομένων και μια συνάρτηση επιστροφής. Με αυτόν τον τρόπο μπορεί να εκτελέσει διαφορετικές διεργασίες καλώντας τα services που προστίθενται στο context να μεταβιβάσει τα δεδομένα και να χρησιμοποιήσει τους dispatcher για να ενεργοποιήσει τα events που ανακλώνται στα Stores της εφαρμογής. Τέλος με την προκαθορισμένη συνάρτηση που δίνεται σαν εισαγωγή έχει τη δυνατότητα να ενεργοποιήσει ένθετα events που μπορούν να αποτελέσουν προκαθορισμένες διαδικασίες από τον εκάστοτε component που τα υλοποιεί.



Εικόνα 44: General Framework Flow

Κατά την διαδικασία της κλήσης στα services της εφαρμογής ξεκινάει ένας κύκλος διαδικασιών που χωρίζεται σε διαφορετικές ενότητες που εύκολα μπορούν να επεκταθούν και να αποτελέσουν κύρια σημεία επεξεργασίας για την εφαρμογή. Με την κλήση ενός service ένα πακέτο από συναρτήσεις βοηθούς προσφέρουν τις κοινές ιδιότητες για την προετοιμασία των δεδομένων και της σύνδεσης με τα απομακρυσμένα σημεία. Συγκεκριμένα οι helpers παράγουν και σχηματίζουν με τα δεδομένα που λαμβάνουν από τα actions όλες τις απαραίτητες πληροφορίες για να υλοποιηθεί η απομακρυσμένη κλήση. Διαβάζουν τα configuration αρχεία και τροποποιούν τους headers, τα paths, τα ports και τα πρωτόκολλα που απαιτούνται. Επίσης στις προκαθορισμένες κλήσεις για διασυνδεδεμένα δεδομένα, διαχειρίζονται τη σύνδεση στα triplestores με τα απαραίτητα options και προετοιμάζουν τον γράφο που πρόκειται να χρησιμοποιηθεί. Επιπλέον προετοιμάζουν μέσω διαδικασιών τη προκαθορισμένη δυναμική συμπεριφορά που ενδέχεται να ακολουθηθεί μέσω της χρήσης των configurators.

Οι configurators αποτελούν μια ξεχωριστή διαδικασία. Με τη λειτουργία τους να επεξεργάζονται και να διαχειρίζονται τα config αρχεία παράγουν μια δυναμική επεξεργασία και κατασκευή αντικειμένων που

θα χρησιμοποιηθούν για τη διάθεση και την ανάθεση των πληροφοριών που θα διατεθούν κατά την επιστροφή των κλήσεων. Η δυναμική υλοποίησή που έχουμε κατασκευάσει ακολουθεί τον αλγόριθμο που παρουσιάσαμε σε προηγούμενο κεφάλαιο και διαθέτει τα επιστρεφόμενα δεδομένα σε γενικότερους τύπους αντικειμένων διατηρώντας χρήσιμες πληροφορίες σχετικά με τα resources που θα αποσπαστούν από τα απομακρυσμένα σημεία.

Στη συνέχεια κατά την ολοκλήρωση των διεργασιών αυτών σειρά έρχονται να πάρουν οι υλοποίησες των συναρτήσεων που κατασκευάζουν τα SPARQL ερωτήματα. Το σύστημα σε αυτό το σημείο έχει τον προκαθορισμένο γράφο που έχει προκύψει κατά την διαδικασία του configuration και αναγνωρίζοντας τις οντότητες που έχουν φτάσει σε αυτό το σημείο από το pre-construction κατασκευάζουν strings με τα ερωτήματα ώστε αυτά να χρησιμοποιηθούν κατά την κλήση. Αφού έχει σχηματιστεί ο γράφος και το ερώτημα ανάλογα με τη διαδικασία που θα πραγματοποιηθεί με το πακέτο του request-promise που έχει συμπεριληφθεί στην υλοποίηση από το npm build, πραγματοποιείται μια κλήση στο εκάστοτε απομακρυσμένο σημείο. Η κλήση περιέχει της παραμέτρους της ερώτησης και τα δεδομένα που απαιτούνται και η επιστροφή της όσον αναφορά τα διασυνδεδεμένα δεδομένα γίνεται με JSON-LD.

Το σύνολο των δεδομένων τώρα πια που έχει συλλεχθεί περνάει από ένα καινούργιο κύκλο επεξεργασίας. Αυτός ο κύκλος ονομάζεται utilization process. Σε αυτή τη διαδικασία τα δεδομένα επεξεργάζονται σύμφωνα με τα αρχεία του configuration και σχηματίζονται τα απαραίτητα αντικείμενα σύμφωνα με τους components που τα υλοποιούν, επιλέγοντας τους κατάλληλους για το rendering. Σε αυτό το σημείο σχηματίζονται τα αντικείμενα που είναι ικανά να διαχειριστούν τα facets και οι components που διαχειρίζονται τα resources. Σύμφωνα με τον αλγόριθμο που παρουσιάσαμε το κάθε resource περνάει και τα 15 στάδια επεξεργασίας πραγματοποιώντας μια διαδικασία override. Η εφαρμογή αυτής της διαδικασίας είναι εύκολο να υλοποιηθεί με την JavaScript και τις ιδιότητες του extend καθώς τα δεδομένα σχηματίζουν αντικείμενα με δείκτες τα resources που τα αποτελούν και τιμές ένθετα αντικείμενα που αναγνωρίζονται από τους components που τα υλοποιούν. Κάθε αντικείμενο που αποτελεί τιμή για τα resources που υλοποιεί, δεν προσβάλει την κύρια απάντηση του ερωτήματος και αποτελεί καθαρά και μόνο σχηματισμό ορισμένων ιδιοτήτων που αναγνωρίζουν οι components. Τα δεδομένα που συλλέγονται διατηρούνται σε τιμές όπως το resources, resourceURI, totals, counts κλπ που έχουν σχηματιστεί πριν την ερώτηση και είναι προκαθορισμένα για την εφαρμογή όσον αναφορά τη συγκομιδή των linked data. Οι επικεφαλίδες των απαντήσεων στα JSON-LD αποτελούν ένα καθοριστικό σχήμα για το vocabulary matching που έχει οριστεί χρησιμοποιώντας και τις τιμές των ιδιοτήτων sameAs της owl και seeAlso του rdfs. Σε αυτό το σημείο μια κύρια λειτουργία έρχεται να λάβει χώρο καθώς εδώ σχηματίζονται τα αντίστοιχα microdata που αποτελούν κύριο χαρακτηριστικό αναγνώρισης δεδομένων για τις semantic τεχνολογίες των δεδομένων. Με βάση τα headers που επιστρέφονται από τα JSON-LD και με τη βοήθεια των αντικειμένων που σχηματίζονται κατά την διαδικασία του utilization η διάσπαση σε οντότητες συντηρεί και στοιχεία για αυτού του είδους τα δεδομένα ανάλογα με το resource ή καλύτερα την ιδιότητα που τα χαρακτηρίζει.

Στη συνέχεια τα δεδομένα που έχουν σχηματιστεί σαν αποτέλεσμα μαζί με τα αντικείμενα που έχουν σχηματιστεί περνούν πίσω στο callback του action το οποίο χρησιμοποιεί το context dispatch. Μέσω των dispatchers τώρα πιά αυτά διατίθενται στο εκάστοτε store που τα διατηρεί. Αυτά υπακούν και στα ονομασμένα events που έχουν αντιστοιχιστεί με τους dispatchers και υλοποιούν το κάθε σήμα με μια συνάρτηση όπου τα δεδομένα επεξεργάζονται αντίστοιχα και περνούν μέσω του emit στους components που απαρτίζουν τη σελίδα. Μέχρι αυτό το σημείο παρατηρείται μια διαδικασία με την οποία ο κάθε προγραμματιστής μπορεί να παραλλάξει κατά την κρίση του το κάθε event και να επεξεργαστεί στα stores ή να διασπάσει τη λογική ενημερώνοντας και άλλα σημεία της εφαρμογής.

Το επόμενο στάδιο είναι οι reactors που μπορούν και αναγνωρίζουν τα αντίστοιχα αντικείμενα που έχουν δημιουργηθεί και να διασπούν τη συλλογή με μια configurable μέθοδο σχηματισμού ανάλογα με τις τιμές που συναντούν. Από αυτό το σημείο και μετά η εφαρμογή διαλέγει τους components που θα χρησιμοποιήσει για να σχηματίσει τα resources. Στην ουσία οι components διαβάζουν λεκτικά τα οποία έχουν αντιστοιχιστεί στο κάθε επίπεδο παρουσίασης, αντιστοιχιζόμενοι με τα resources, και διαλέγουν τους αντίστοιχους components που θα υλοποιήσουν. Η όλη αυτή διαδικασία της δυναμικής εφαρμογής μπορεί εύκολα να τροποποιηθεί και να επεκταθεί σε κάθε στάδιο έτσι ώστε να προσδοθεί παραπάνω δυναμικότητα και ευελιξία στο τελικό UI που θα παρουσιαστεί στον χρήστη.

Όπως παρατηρεί κάποιος στον όλο κύκλο της διαδικασίας που περιγράψαμε, δίνεται η δυνατότητα σε έναν προγραμματιστή να επέμβει σε πολλά σημεία της εφαρμογής και να χρησιμοποιήσει είτε κομμάτια υλοποιημένα είτε να κατασκευάσει νέα και να προσαρτήσει μια διαφορετική λειτουργία εκμεταλλευόμενος τις δυναμικές διαδικασίες του rendering. Το framework υλοποιεί κύρια κομμάτια που μπορεί να αποτελέσουν προκαθορισμένες διαδικασίες επεξεργασίας διασυνδεδεμένων δεδομένων. Η προϋπόθεση να επεκτείνει κάποιος την λειτουργία του δεν έχει να κάνει τόσο με τη γνώση των τεχνολογιών των semantics. Μπορεί εύκολα ένας κοινός προγραμματιστής να παράγει καινούργιος components και να τους συμπεριλάβει στη λύση. Σε δεδομένα άγνωστα από τον ορισμό των configuration που έχουν να κάνουν με διαδικασίες επεξεργασίας των resources που επιστρέφονται το ίδιο το σύστημα προσφέρει μια προκαθορισμένη λειτουργία που ορίζεται ως generic και μπορεί να επεξεργαστεί άγνωστα αντικείμενα καθώς και ενδεχόμενες προσθήκες στο μέλλον.

5. Δεδομένα που συλλέχθηκαν

Στην διαδικασία υλοποίησης μιας εφαρμογής Διασυνδεδεμένων Δεδομένων, έγιναν γνωστές και παρατηρήθηκαν οι τρόποι διασύνδεσης και οι ιδιότητες μιας LDA εφαρμογής. Όπως παρουσιάστηκε η υλοποίηση μιας τέτοιας εφαρμογής έκανε σαφής τις ανάγκες για μοντελοποίηση των ανοικτών δεδομένων και τους τρόπους με τους οποίους γίνεται υπολογιστικά μια υλοποίηση σε συνδυασμό με το περιβάλλον του τελικού χρήστη και πως παρουσιάζονται σε αυτόν τα δεδομένα.

Η προτεινόμενη λύση βασισμένη σε components δίνει έμφαση στην επαναχρησιμοποίηση και τον διαχωρισμό των ανησυχιών, όσον αφορά την ανάπτυξη, εφαρμογών συνδεδεμένων δεδομένων. Ο μηχανισμός προσαρμογής UI με βάση το RDF στοχεύει στην καλύτερη προσαρμογή και εξατομίκευση με βάση το νόημα των δεδομένων. Επιπλέον, η χρήση τυποποιημένων στοιχείων Web φιλοδοξεί να φέρει καλύτερη επικοινωνία μεταξύ σχεδιαστών UX και προγραμματιστών Σημασιολογικού Ιστού, προκειμένου να επαναχρησιμοποιηθούν οι καλύτερες πρακτικές UI σε εφαρμογές Linked Data. Υποστηρίζεται ότι η γεφύρωση του χάσματος μεταξύ των τεχνολογιών Semantic Web και Web Components φέρνει αμοιβαία οφέλη και για τις δύο πλευρές. Από τη μία πλευρά, οι τεχνολογίες του Σημασιολογικού Ιστού παρέχουν υποστήριξη για την ανακάλυψη, τη διαλειτουργικότητα, την ενσωμάτωση και την προσαρμογή στον Παγκόσμιο Ιστό. Ενώ από την άλλη τα Web Components φέρνουν τα πλεονεκτήματα της τυποποίησης UI, της επαναχρησιμοποίησης, της δυνατότητας αντικατάστασης και της ενσωμάτωσης σε τρέχουσες εφαρμογές Σημασιολογικού Ιστού.

Όπως αποδείχτηκε και αναφέρθηκε σε προηγούμενα κεφάλαια μια υλοποίηση κώδικα με χρησιμοποίηση των κατάλληλων frameworks για διασυνδεδεμένα δεδομένα που υλοποιούν τις βασικές αρχές και ιδιότητες των διασυνδεδεμένων δεδομένων μπορεί να έρθει σε συνεργατικές ολοκληρωμένες λύσεις με υπάρχοντα περιβάλλοντα χρήστη και μηχανισμούς διεκπεραίωσης πληροφοριών προσφέροντας ολοκληρωμένες λύσεις για κάθε οργανισμό και της λογικής που επιθυμεί να εκφράσει στο κοινό του τις κατάλληλες πληροφορίες και χειρισμούς με μικρές αλλαγές και παραμετροποιήσεις.

Το γενετικό κομμάτι παρουσίασης και υλοποίησης των λειτουργιών έρχεται να λύσει στην πραγματικότητα τα χέρια των σχεδιαστών λογισμικού σε ότι έχει να κάνει με την επαναχρησιμοποίηση και υλοποίηση κομματιών κώδικα για κοινά σημεία και να το περιορίσει σε σχεδιασμό εξιδανικευμένων στοιχείων που απαρτίζουν μικρότερα κομμάτια της εκάστοτε εφαρμογής. Με τα rdf dumps που χρησιμοποιήθηκαν παρουσιάστηκε η ευκολία μοντελοποίησης και χειρισμού των διασυνδεδεμένων δεδομένων με αποτέλεσμα να γίνει ορατή η ευκολία πρόσμιξης υπάρχων υλοποιήσεων σε συνδυασμό με τα endpoints που παρέχουν πληροφορία με δομές rdf και ήδη υπάρχων δομημένες πληροφορίες στο διαδίκτυο.

Λειτουργίες που απαιτούν την δημοσιοποίηση και χειρισμό των δεδομένων καθίστανται ευκολότερες αφού οι κοινότητα ανοικτού κώδικα έχει εξασφαλίσει υπάρχων ανεξάρτητα γενετικά και λειτουργικά ολοκληρωμένα components που μπορούν εύκολα να προσαχθούν σε μια υπάρχων υλοποίηση. Με αυτόν τον τρόπο ένας προγραμματιστής και ένας μηχανικός λογισμικού που επιθυμεί να επεκτείνει την εφαρμογή του μπορεί να χρησιμοποιεί τα ανοικτά δεδομένα και να ανάγει την ανάγκη υλοποίησης σε χρησιμοποίηση των κατάλληλων components που εξυπηρετούν κάθε είδους ανάγκη σε ότι έχει να κάνει με τον κύκλο ζωής, τις κοινές λειτουργίες καθώς και τα πρότυπα που χρησιμοποιούνται σε σχέση με τη λειτουργεία των διασυνδεδεμένων δεδομένων.

Με τη διάσπαση των λειτουργιών σε μικρότερες λογικές μονάδες παρατηρήσαμε ότι κάθε υλοποίηση μπορεί εύκολα να επεκταθεί και να διαμορφωθεί ώστε να αποτελέσει μια πιο ολοκληρωμένη λύση στην πάροδο του χρόνου. Με την αρχιτεκτονική που χρησιμοποιήσαμε η υλοποίηση αποτέλεσε σχεδιασμό και κατασκευή μεθόδων και λειτουργιών σε μικρού εύρους λογική. Η αρχιτεκτονική του flux και της μονής κατεύθυνσης του binding έδωσε μια ευελιξία σε σχέση με την πρόσμιξη και τη μεταξύ τους επίδραση ενεργειών και δομικών στοιχείων. Παρατηρήθηκε ότι κάθε δομική μονάδα μπορεί να υπακούει και να επιλέγει τα event που την ενδιαφέρουν με αποτέλεσμα να γίνεται ευκολότερη η μεταβολή της και η τροποποίησή της.

Η διαδικασία της ανάλυσης της πληροφορίας και η ομοιογένεια επεξεργασίας του αλγορίθμου αναπαράστασης προσέδωσε χαρακτηριστικά ομοιογένειας στα διασυνδεδεμένα δεδομένα. Η χρήση της ιδιότητας αυτού του τύπου των στοιχείων να περιγράφονται με παρόμοιο τρόπο και να διατίθενται από οργανισμούς που τα χαρακτηρίζουν έκανε το σύστημά μας μετά από ένα σημείο να μπορεί να επεκταθεί με μόνο απλή παραμετροποίηση. Σαφώς και στην υλοποίηση χρειάζονται προσαρμογές που επεκτείνουν τη λειτουργία του σχετικά με τη σύνδεση σε απομακρυσμένα σημεία και APIs τα οποία διαθέτουν μια διαφορετική πληροφορία που δεν είναι χαρακτηρισμένη από διασυνδεδεμένα δεδομένα αλλά αυτό αποτελεί ένα κομμάτι ξεχωριστής υλοποίησης που μπορεί εύκολα να προσαχθεί στις δομικές μονάδες που την στελεχώνουν. Η ανεξαρτησία του τρόπου λειτουργίας των δομικών μονάδων με τη διαδικασία υπακοής σε events των handlers τις καθιστά ευκόλως επεκτάσιμες και λειτουργικές. Εύκολα ένας προγραμματιστής μπορεί να ενσωματώσει επιπλέον λειτουργικότητα με τη χρήση νέων δομικών στοιχείων ή ακόμα και αντιγραφής των υπαρχόντων με επεκτάσεις, καθώς μπορεί να κάνει χρήση των actions και των handlers με τον τρόπο που τον ενδιαφέρει.

Σημαντικό πλεονέκτημα αυτού του τύπου της αρχιτεκτονικής είναι ότι κάθε στοιχείο μπορεί να αποσπαστεί και να αποτελέσει λειτουργία ενός άλλου με την χρήση ενός event ανανέωσης του περιεχομένου του. Όσον αναφορά την επέκταση των διασυνδεδεμένων δεδομένων από τις πηγές του web το σύστημα χρήζει ανεξάρτητο καθώς μπορεί και συμπεριφέρεται με ένα δυναμικό τρόπο σε κάθε είδους ιδιότητα στοιχείου και της τιμής του. Σκοπός ήταν σε ενδεχόμενες αλλαγές σχετικά με την δόμηση τέτοιων στοιχείων το σύστημα να μπορεί εύκολα να ανταπεξέρχεται και να προσαρμόζεται. Με το δυναμικό αυτό χειρισμό των Linked Data που παρουσιάσαμε κάθε ιδιότητα μπορεί να παραμετροποιηθεί ή και να αφαιρεθεί αφού κατά τις διαδικασίες του rendering και του utilization process θα αγνοηθεί αν δεν την αναγνωριστεί ή θα αντιμετωπιστεί με την προκαθορισμένη λειτουργία.

Όσον αναφορά τη διάθεση και την επεξεργασία των δεδομένων από web crawlers και μηχανισμούς αναζήτησης οι components που αναλαμβάνουν το rendering της πληροφορίας διατηρούν και το property που έχει περισυλλεγεί με αποτέλεσμα να προσαρτώνται στην εφαρμογή microdata που παρέχουν αυτού του είδους την πληροφορία. Με αυτόν τον τρόπο οι αυτοματοποιημένοι μηχανισμοί μπορούν να διαχειριστούν την απαραίτητη πληροφορία και να προσφέρουν την μορφή αυτοματισμού την οποία προσφέρουν.

Η εφαρμογή μπορεί να χρησιμοποιήσει ξεχωριστά end points για τη συγκομιδή των πληροφοριών καθώς με τη βοήθεια των helpers αντιστοιχεί τα δεδομένα που επιθυμεί να συλλέξει από ένα resource που χαρακτηρίζει τον τύπο του αντικειμένου που επεξεργάζεται αρκεί αυτά να επιστρέφονται σε μια μορφή JSON-LD. Το resource αυτό στη πρωταρχική συνιστώσα του dataset αποτελεί και το endpoint που ορίζεται από ένα αντικείμενο στο αρχείο που υλοποιεί τους servers. Με αυτόν τον τρόπο κάθε οντότητα που είναι ορισμένη να αναπαριστά το σύστημα μπορεί να περισυλλέγεται από διαφορετικά σημεία στο web αρκεί αυτά να έχουν οριστεί. Εύκολα με ενδεχόμενη παραμετροποίηση η λειτουργία αυτή μπορεί να επεκταθεί και να αλλάζει το αρχείο που αποτελεί την κύρια πηγή παραμετροποίησης ή

και επέκτασης των συνδέσμων με έναν αυτοματοποιημένο τρόπο ή να προσφέρεται σαν αποτέλεσμα λειτουργίας που επιστρέφεται από ένα σύστημα Back-End στην υλοποίηση.

Κατά τη διαδικασία αποτίμησης της περισυλλεγμένης πληροφορίας οι οντότητες που έχουν χαρακτηριστεί μπορούν να κληρονομήσουν χαρακτηριστικά από ήδη ορισμένες τιμές που αποτελούν τα resources στο αρχείο του reactor. Με αυτό τον τρόπο η σύνδεση σε ένα τελείως άγνωστο end point το οποίο μπορεί να μεταβιβάσει την απάντηση σε μια μορφή JSON-LD μπορεί μέσω των κεφαλών ή της ιδιότητας sameAs της owl όχι μόνο να χρησιμοποιηθεί από την εφαρμογή αλλά και τα δεδομένα που επιστρέφονται από αυτό να χειριστούν από το framework με ήδη υλοποιημένους Components. Αυτό το στοιχείο λειτουργίας που εξασφαλίζει το utilization process είναι και παράλληλα μια μορφή αυτόματης λογικής καθώς η αναγνώριση του resource μπορεί να οριστεί μια μόνο φορά και να χρησιμοποιείται από κάθε σύνδεση σε διαφορετικό πόρο. Από την άλλη αυτό αποτελεί και ένα μεγάλο κίνδυνο σε ενδεχόμενες επιθέσεις κακόβουλου λογισμικού. Αυτό γιατί μπορεί ένα συγκεκριμένο δεδομένο που παρέχεται να προσαρτήσει συνδέσμους που η εφαρμογή δεν είναι σε θέση να αναγνωρίσει αλλά θα του επιτρέψει να αποτυπωθεί μέσω της δυναμικής ιδιότητας. Αυτό γιατί το σύστημα είναι υλοποιημένο να επεξεργάζεται resources που αποτυπώνονται με τη μορφή συνδέσμων αλλά δεν είναι σε θέση να επεξεργάζεται το URI που σχηματίζεται αναγνωρίζοντας την προέλευσή του ή τη λειτουργία του. Ο σχεδιασμός και η υλοποίηση μας, αφορά τις διαδικασίες συλλογής και αποτύπωσης των δεδομένων με ένα δυναμικό τρόπο και όχι να επεξεργάζεται την προέλευση και την αξιοπιστία ενός πόρου που του διατήθεται.

Σχετικά με τις κυβερνοεπιθέσεις και την προσθήκη κακόβουλου λογισμικού το rendering html κώδικα που μπορεί να προσαρτηθεί με την χρήση ενός και μόνο URL που είναι προκαθορισμένο να αναπαρίσταται από επιλεγμένους Components είναι κάτι το οποίο ένας προγραμματιστής είναι δύσκολο να προβλέψει. Έτσι μια λειτουργία imaging η preview ενός συνδέσμου μπορεί να αποτελέσει ναι μεν μια σπουδαία λειτουργία όσον αναφορά την λειτουργικότητα μιας εφαρμογής αλλά και δύσκολα ελέγχεται είτε από προγράμματα είτε από μηχανισμούς που την αποτρέπουν. Έτσι ο ορισμός Components που υλοποιεί τέτοιες λειτουργίες πρέπει να γίνεται με σύνεση και να αποφεύγονται αγνώστου τύπου resources κατά τη διαδικασία του configuration.

6. Συμπεράσματα και μελλοντικά ζητήματα

Αυτή η εργασία εισαγάγει την έννοια και τις βασικές αρχές των Συνδεδεμένων Δεδομένων, μαζί με επισκοπήσεις τεχνολογιών υποστήριξης όπως URIs, HTTP, JSON-LD και RDF. Συλλογικά, αυτές οι τεχνολογίες και οι αρχές επιτρέπουν ένα στυλ δημοσίευσης που μετατρέπει δεδομένα στο ίδιο το επίπεδο του Ιστού. Ένα μοναδικό χαρακτηριστικό που εκθέτει τα Συνδεδεμένα Δεδομένα στις αυστηρές και απεριόριστες δυνατότητες του Παγκοσμίου Ιστού. Αυτή η απόλυτη ολοκλήρωση με τον Ιστό, η οποία υποστηρίζεται από ανοικτά πρότυπα που βασίζονται στην κοινότητα, εξυπηρετεί επίσης μια προστατευτική λειτουργία για τους εκδότες δεδομένων που ενδιαφέρονται για τη μελλοντική προστασία των περιουσιακών τους στοιχείων.

Τα Linked Data έχουν υιοθετηθεί από σημαντικό αριθμό εκδοτών δεδομένων οι οποίοι, συλλογικά, έχουν κατασκευάσει έναν ιστό δεδομένων εντυπωσιακής κλίμακας. Με αυτόν τον τρόπο, έχουν αποδείξει τη σκοπιμότητά τους ως μια προσέγγιση για τη δημοσίευση δεδομένων στο Web και την αυξανόμενη ωριμότητα των πλατφορμών λογισμικού και των εργαλείων που υποστηρίζουν αυτό. Η ανάπτυξη εργαλείων ενημερώνεται πάντοτε όχι μόνο από τα πρότυπα και τις προδιαγραφές αλλά από τις βέλτιστες πρακτικές που αναπτύσσονται και υιοθετούνται στη σχετική κοινότητα.

Ίσως η μόνη πτυχή των Συνδεδεμένων Δεδομένων να ξεπεράσει την ποικιλομορφία των σεναρίων δημοσίευσης θα είναι οι τρόποι με τους οποίους τα Συνδεδεμένα Δεδομένα καταναλώνονται από τον Ιστό. Αυτό θέτει ένα αρχικό θεμέλιο πάνω στο οποίο μπορούν να τοποθετηθούν διαφορετικές αρχιτεκτονικές για κατανάλωση Συνδεδεμένων Δεδομένων. Μεταξύ αυτής της ποικιλομορφίας, η οποία πρέπει να επικροτηθεί, θα υπάρχουν αναπόφευκτα ορισμένες κοινές απαιτήσεις και συστατικά λογισμικού που αντικατοπτρίζουν τα βασικά χαρακτηριστικά και τις δυνατότητες των Συνδεδεμένων Δεδομένων στο ανοικτό, χαοτικό και αντιφατικό περιβάλλον του Ιστού.

Οι μεγάλες πολυεθνικές εταιρείες αντιμετωπίζουν παρόμοιες προκλήσεις με αυτές που αντιμετωπίζονται στο πλαίσιο των Συνδεδεμένων Δεδομένων. Διατηρούν χιλιάδες ανεξάρτητα εξελισσόμενες βάσεις δεδομένων μεταξύ των τμημάτων, των θυγατρικών και των νεοαποκτηθέντων εταιρειών και αγωνίζονται για την αξιοποίηση του δυναμικού των στοιχείων τους. Επομένως, όπως οι κλασσικές τεχνολογίες ιστού έχουν υιοθετηθεί ευρέως μέσα στα intranets, τα Linked Data έχουν τη δυνατότητα να χρησιμοποιηθούν ως ελαφριά, ενσωματωμένη τεχνολογία ενσωμάτωσης δεδομένων σε μεγάλους οργανισμούς. Σε αντίθεση με τις κλασικές αποθήκες δεδομένων που απαιτούν δαπανηρή επένδυση εκ των προτέρων για τη μοντελοποίηση ενός σφαιρικού σχήματος, οι τεχνολογίες Linked Data επιτρέπουν στις εταιρείες να δημιουργήσουν χώρους δεδομένων με σχετικά μικρή προσπάθεια. Δεδομένου ότι χρησιμοποιούνται αυτοί οι χώροι δεδομένων, οι εταιρείες μπορούν να επενδύσουν βήμα προς βήμα στη δημιουργία συνδέσμων δεδομένων, κοινών λεξιλογίων ή αντιστοίχισης σχημάτων μεταξύ των πηγών για να επιτρέψουν βαθύτερη ολοκλήρωση.

Είναι επίσης ενδιαφέρον να σημειώσουμε ότι οι μεγάλες εταιρίες στο διαδίκτυο είναι ήδη απασχολημένες με την κατασκευή τέτοιων χώρων δεδομένων. Το Google, το Yahoo και το Facebook άρχισαν να συνδέουν όλα τα δεδομένα χρήστη, γεωγραφικά και λιανικά και άρχισαν να χρησιμοποιούν αυτούς τους χώρους δεδομένων στις εφαρμογές τους. Σε αντίθεση με τον ανοιχτό ιστό, ο οποίος είναι προσβάσιμος σε όλους, αυτοί οι αναδυόμενοι χώροι δεδομένων ελέγχονται από μεμονωμένες εταιρείες οι οποίες αποφασίζουν επίσης πώς αξιοποιούνται οι χώροι δεδομένων - προς όφελος ολόκληρης της κοινωνίας ή όχι. Ως εκ τούτου, παρόλο που σήμερα μπορεί να είναι ευκολότερο να βελτιωθεί ένα προφίλ στο Facebook ή να ανεβάσετε μια βάση δεδομένων στους πίνακες σύντηξης

Google, αντί να δημοσιεύσετε τα συνδεδεμένα δεδομένα, η προσπάθεια μπορεί να δαπανηθεί σωστά καθώς συμβάλλει σε ένα στοιχείο δημόσιων δεδομένων ικανό να επηρεάσει βαθιά το μέλλον ως άφιξη του ίδιου του ιστού.

Ως μελλοντικό μας σχέδιο, σκοπός είναι να δημιουργηθεί υποδομή cloud για την κοινή χρήση και επαναχρησιμοποίηση των επαναχρησιμοποιούμενων components και των διαμορφώσεων προκαθορισμένων λειτουργιών – συστημάτων καθώς και εξαρτήματα Web χωρίς την ανάγκη εγκατάστασης κάποιου πλαισίου που θα κάνει λειτουργικές της εφαρμογές βάση εγκατάστασής του σε μια υλοποίηση. Κατά αυτόν τον τρόπο το web θα έχει την δυνατότητα να χρησιμοποιείται και να επεκτείνεται από μηχανές αυτόματου συμπερασμού οι οποίοι θα κατανοούν πλήρως τη δομή της πληροφορίας ή και ακόμα από τελικούς χρήστες που δεν θα έχουν την εμπειρία σχεδίασης εφαρμογών σημασιολογικού ιστού χρησιμοποιώντας προκαθορισμένα πρότυπα και δομές οι οποίες θα εκφράζουν λειτουργίες που θα είναι κατανοητές και από τις δυο πλευρές. Αυτού του τύπο η κατεύθυνση για μελλοντική έρευνα είναι η ανάπτυξη μηχανισμών για την αυτόματη διαμόρφωση και σύνθεση στα στοιχεία του ιστού βάσει της σημασιολογικής σήμανσης που παρέχεται.

Με την κατασκευαστική προσέγγιση και τη χρησιμοποίηση προκαθορισμένων αρχείων επεξεργασίας στα στάδια της παραμετροποίησης, σύνδεσης και αναγνώρισης κοινών στοιχείων στόχος μας είναι σε ενδεχόμενη επέκταση να καθίσταται εφικτή η παραμετροποίηση και ο καθορισμός αυτών από υλοποιήσεις εξωτερικών συστημάτων χωρίς αυτό να επιδρά άμεσα στην εφαρμογή. Συγκεκριμένα τα στάδια στα οποία μπορεί να αποτελέσουν επέκταση είναι ο ορισμός του χειρισμού των resources που αποτελούν την διαδικασία του rendering, το mapping των συνδέσμων του utilization process, η διαχείριση των πόρων συγκομιδής και ο τρόπος αναδημοσίευσης δεδομένων καινούριων συνόλων στοιχείων. Η υλοποίηση χρησιμοποιεί την τεχνολογία του JSON-LD και αναγνωρίζει τα rdf που σαν τεχνολογίες μπορούν και αναπαριστούν με επιτυχία τα δεδομένα στην εφαρμογή. Η διαδικασία αυτή του δυναμικού rendering αναγνωρίζει του πόρους που συλλέγονται από τα στοιχεία του web μέσο ενός αρχείου.

Ενδεχόμενη επέκταση μπορεί να αποτελέσει η σύνδεση σε μια υπηρεσία που σχηματίζει ένα τέτοιο τύπο αρχείου το οποίο δεν αποτελεί προσθήκη στην υλοποίηση αλλά αποτέλεσμα μιας υπηρεσίας που έχει υλοποιηθεί σε σύνδεσμο του διαδικτύου. Endpoints που χειρίζονται μια τέτοια διαδικασία μπορούν να χρησιμοποιηθούν από εφαρμογές οργανισμών ή κοινών πόρων του διαδικτύου χωρίς να αποτελούν μέρος της λύσης. Με αυτόν τον τρόπο η επέκταση της διαδικασίας επεξεργασίας των συλλεγμένων πληροφοριών μπορεί να αποτελέσει ξεχωριστό κομμάτι, τροποποιήσιμο σε ενδεχόμενες αλλαγές των δεδομένων ακόμα και από αυτοματοποιημένους μηχανισμούς.

Οι διαδικασίες του mapping και της αναδημοσίευσης έχουν υλοποιηθεί σε rdf dumps τα οποία μπορούν κάλλιστα να ανανεωθούν σε οποιαδήποτε χρονική στιγμή. Οι διαδικασίες αυτές χρησιμοποιούνται από το σύστημα σε διαφορετικά στάδια. Η πρώτη είναι υπεύθυνη να αντιστοιχίσει τα δεδομένα περισυλλογής χρησιμοποιώντας τα IRIs που διατίθενται για να αναγνωρίσει έναν κοινό τρόπο απεικόνισης σε τιμές που έχουν οριστεί. Μια άλλη χρησιμοποίηση του mapping είναι κατά την διαδικασία της δημοσίευσης και κατασκευής νέων στοιχείων τα οποία έχουν σχηματιστεί από το περιβάλλον του χρήστη και χρησιμοποιούν ένα λεξιλόγιο σε μορφή αρχείου για το σχηματισμό του γράφου δεδομένων. Σε αυτό το σημείο η εφαρμογή καθιστά έναν τρόπο τοπικής επεξεργασίας των στοιχείων τα οποία αντιστοιχεί σε δυναμικά prefixes που θα αποτελέσουν των γράφο και χρησιμοποιούνται για τα microdata όσον αναφορά τη δημοσίευση. Εύκολα ο χειρισμός αυτών των λειτουργιών μπορεί να προέλθει από υπηρεσίες εταιρειών και servers που παρέχουν αντίστοιχη πληροφόρηση. Κατά αυτόν τον τρόπο η διαδικασίες μπορούν να αποτελέσουν μια δυναμικής

προσέγγισης υλοποίηση καθώς η τοπική διαχείριση γίνεται με δυναμικό τρόπο. Σαφώς ένα σύστημα το οποίο αναγνωρίζει της οντότητες όπως αυτές προσφέρονται από το διαδίκτυο μπορεί να αποτελέσει κύρια πηγή για τον σχηματισμό και τις απαιτήσεις που μπορεί να αποτελέσουν μέρος της εφαρμογής. Ένα ενδεχόμενο Hypermedia μπορεί να εκφραστεί με κανόνες σε ένα rdf dump το οποίο σχηματίζει τα απαραίτητα δεδομένα που θα χρειαστούν και να αποτελέσουν την περιγραφή του στο σύστημα. Με τον χαρακτηρισμό νέων οντοτήτων και αντικειμένων που αποτελούν μορφή κοινώς διαδεδομένη το σύστημα θα είναι σε θέση να περιγράψει όλο και περισσότερα δεδομένα και υπηρεσίες τα οποία αποτελούν εξελισσόμενες πηγές πληροφόρησης για τον χρήστη στη διαπροσωπεία.

Ο χαρακτηρισμός των τύπων των δεδομένων και των λογικών υπηρεσιών σε rdf διαγράμματα και η περιγραφή τους από IRIs τα οποία μπορούν να αποτελέσουν τις τιμές τους για συγκεκριμένες ιδιότητες αντιμετωπίζεται από το σύστημα δυναμικής αναπαράστασης ως μια οντότητα που αποτελεί προϊόν ανακατεύθυνσης στον χρήστη με αποτέλεσμα μια νέα καρτέλα να δημιουργείται με την ανακατεύθυνση στον συγκεκριμένο σύνδεσμο. Αυτή η λειτουργία μπορεί να αποτελέσει ενδεχομένως μια επέκταση σε ένα σύστημα που παρέχει αυτοματοποιημένα την πληροφορία. Η λειτουργία περισσότερης πληροφορίας σχετικά με ένα συγκεκριμένο αντικείμενο στη λεπτομερής παρουσίασης θα μπορούσε να επεκταθεί με τη διασύνδεση σε ένα Hypermedia API[49] το οποίο να προσφέρει επιλογές για διαδικασίες που σχετίζονται με το συγκεκριμένο αντικείμενο και να το εμπλουτίσει με λειτουργίες που το περιγράφουν και ταυτίζονται με αυτό. Για παράδειγμα ο χαρακτηρισμός μιας οντότητας να περιγράφεται από ένα δομικό στοιχείο που εκτελεί λειτουργίες θα μπορούσε να αποτελέσει μια πιο ολοκληρωμένη λειτουργία σε αντικείμενα που είναι χαρακτηρισμένα με ιδιότητες προκαθορισμένες στο web ή υπηρεσίες που διατίθενται από εξωτερικά συστήματα. Στο παράδειγμα της ηλεκτρονικής βιβλιοθήκης ας πούμε το Interface θα μπορούσε να χαρακτηρίσει με μια ιδιότητα την αγορά ενός βιβλίου από μια υπηρεσία ορίζοντας ένα χαρακτηριστικό στο αντικείμενο με έναν υπερσύνδεσμο ο οποίος θα ήταν υπεύθυνος με το context της εφαρμογής που είναι εκφρασμένο στην συγκεκριμένη κατάσταση να εκμεταλλευτεί έτσι ώστε να εκτελέσει την συγκεκριμένη αγορά. Η υλοποίηση χαρακτηριστικών εκφρασμένων σε υπερσυνδέσμους δεν είναι ανάγκη να περιορίζεται σε event ανακατεύθυνσης στον χρήστη αλλά μπορεί να κρύβει μια πιο σύνθετη υπηρεσία την οποία εκφράζει με ένα δομικό στοιχείο στον χρήστη σύμφωνα με τις ανάγκες του. Η λειτουργία αυτή θα μπορούσε να υλοποιηθεί απλά με ένα νέο component ο οποίος θα καθοριστεί στο configuration να σχηματίζεται στην διαπροσωπεία και να υπακούει σε events που περισυλλέγουν την πληροφορία. Οι τεχνολογίες του web σήμερα υλοποιούνται από συστήματα τα οποία εκθέτουν τις λειτουργίες του εκφρασμένες σε URLs και δίνουν τις δυνατότητες στον client κάθε εφαρμογής να πραγματοποιήσει λειτουργίες σύμφωνα με τις ανάγκες του. Απαραίτητα συστατικά για την υλοποίηση τέτοιου τύπου διαδικασιών είναι η γνώση του συνδέσμου και οι λεπτομέρειες του αντικειμένου που αλληλοεπιδρά. Η δυνατότητα του δυναμικού rendering που προσφέρει η εφαρμογή μπορεί και την καθιστά ικανή να ενσωματώσει τέτοιες λειτουργίες με την υλοποίηση ξεχωριστών δομικών στοιχείων που χαρακτηρίζονται από ιδιότητες των αντικειμένων. Έτσι κάθε δομικό στοιχείο περισυλλέγοντας τις απαραίτητες πληροφορίες που διατίθενται στο context της υλοποίησης μπορεί να λειτουργήσει ανεξάρτητα και να ενσωματωθεί στην υπάρχων υλοποίηση εύκολα προσδίδοντας λειτουργίες που αντιπροσωπεύουν αλλά και εμπλουτίζουν τη κάθε οντότητα που χαρακτηρίζεται από το σύστημα.

Με την υλοποίηση πιο λεπτομερών components που εξυπηρετούν κάθε είδους ανάγκη ενδεχόμενης οντότητας όπως για παράδειγμα ένα αντικείμενο που αντιπροσωπεύει μια εκφρασμένη λειτουργία σε ένα IoT περιβάλλον η επέκταση της διαπροσωπείας μπορεί να αποτελέσει ένα δυναμικό τρόπο χειρισμού και αναπαράστασης κάθε διασυνδεδεμένου συστήματος. Αυτό που αρκεί για την υλοποίηση είναι να χαρακτηρίζονται οι ιδιότητες και οι λειτουργίες της με τη δομή των διασυνδεδεμένων δεδομένων. Ο ορισμός αυτός σε JSON-LD και rdf διαγράμματα μπορεί να εμπλουτίσει ένα User

Interface με δομικά στοιχεία τα οποία η λειτουργία τους θα περιγράφεται από συνδέσμους. Η έκδοση λειτουργιών ενός Server σε URLs που θα αποτελούν ενδεχόμενη χρησιμοποίηση της αντιπροσωπευτικής οντότητας μπορεί να αποτελέσει ενδεχομένως επέκταση σε χειρισμούς των sensors που χρησιμοποιούνται και σαν αποτέλεσμα η διεπαφή να αποτελέσει μέσο διαχείρισης. Καθώς τα πάντα στο web μπορούν να αντιστοιχιστούν σε μια REST υπηρεσία και να χαρακτηριστούν σαν αντικείμενα που αναγνωρίζει ένα σύστημα με τις τεχνολογίες των semantics, ένα δυναμικά χαρακτηρισμένο περιβάλλον χρήστη μπορεί να αποτελέσει μια λύση προκαθορισμένων διαδικασιών και ενεργειών που συμβάλουν στην περιγραφή λειτουργιών μέσο πόρων που εξυπηρετούν αυτού του τύπου τις ιδιότητες. Οι διαδικασίες που μπορούν να περιγραφούν μέσω συνδέσμων είναι αμέτρητες καθώς κάθε σύστημα γίνεται expose στο διαδίκτυο. Όλα τα σύγχρονα APIs διαθέτουν λειτουργίες για κάθε τύπου αντικείμενο το οποίο μπορεί να εκφραστεί μέσο υπερσυνδέσμων.

Η τεχνολογία flux που χρησιμοποιήθηκε για τη δυναμική αναπαράσταση μπορεί και περιορίζει την ανανέωση του context των δεδομένων σε συγκεκριμένα δομικά στοιχεία που όπως περιεγράφηκε μπορούν όχι μόνο να αλλάζουν το περιεχόμενό τους χωρίς να επηρεάζουν την υπόλοιπη εφαρμογή αλλά και να υλοποιούν γεγονότα που αφορά μόνον αυτά. Η ανεξαρτησία των δομικών στοιχείων μπορεί να προσαρμόσει μια λειτουργία της εκάστοτε οντότητας ξεχωριστά και η οποία προκαθορίζεται από μια κατάσταση στην οποία μεταβαίνει το συγκεκριμένο context που αντιπροσωπεύει. Έτσι ανεξάρτητα από το υπόλοιπο σώμα της εφαρμογής κάθε οντότητα μπορεί να εφαρμόσει τις καταστάσεις στις οποίες μεταβαίνει με τον διορισμό των δομικών στοιχείων που την περιγράφουν. Ενδεχόμενες αλλαγές που προκύπτουν στην κατάσταση του αντικειμένου που περιγράφεται μπορεί να αντιστοιχιστεί με δομικά στοιχεία που την απεικονίζουν αρκεί αυτά να έχουν οριστεί. Αυτό αποτελεί μια πολύ μεγάλη ιδιότητα επικεντρωμένης διαχείρισης σε ένα συγκεκριμένο χαρακτηριστικό που αποτελεί το κάθε αντικείμενο. Αν όχι μόνο το κάθε αντικείμενο αλλά και η ιδιότητα μπορεί και αποκτά ένα χαρακτηριστικό να μεταβαίνει σε καταστάσεις που διαχειρίζονται αυτόνομα, το κάθε δομικό στοιχείο μπορεί να αποτελέσει μια ολοκληρωτική διαχείριση εκφρασμένη μέσα από ένα απλό μοντέλο JSON-LD. Σκεφτείτε κάθε δομική μονάδα της εφαρμογής να αλληλοεπιδρά με ένα server ο οποίος μπορεί και περιγράφει όχι μόνο τις λειτουργίες του αλλά και τις καταστάσεις στις οποίες μεταβαίνει το σύστημα μετά από την πραγματοποίηση μιας ενέργειας. Αυτό θα μπορούσε να αποτελέσει και ένα συγκεκριμένο τρόπο διαχείρισης σε επίπεδο ιδιότητας το οποίο να δρα σε συγκεκριμένο πεδίο εφαρμογής και να προσδίδει στην εφαρμογή επιπλέον λειτουργικότητα και χαρακτηριστικά ανεξάρτητα διαχειρίσιμα από μέρος της εφαρμογής.

Μια τέτοια υλοποίηση που μπορεί να προσφέρει τις ενέργειες και τις καταστάσεις στις οποίες μπορεί να μεταβεί ένα σύστημα με τις τεχνολογίες των semantics είναι η προσέγγιση των Deep Graphs. Το πρότυπο Deep Graphs, το οποίο βασίζεται στις τεχνολογίες του σημασιολογικού ιστού, ουσιαστικά αποτελεί έναν επίσημο τρόπο να περιγράφει κανείς Restful Web APIs τα οποία βασίζονται πάνω στο JSON-LD και πηγαίνει πέρα από την σημερινή κατάσταση με το να προτείνει την δυνατότητα συντονισμού πολλαπλών clients και με το να αναλύει και να αποφασίζει σχετικά με νέες δραστηριότητες ουσιαστικά σε πραγματικό χρόνο. Στοχεύει στο να δείξει ότι οι δραστηριότητες έχοντας έξυπνους agents οι οποίοι θα περιέχουν όλη την πληροφορία που χρειάζονται για να πετύχουν έναν στόχο, και έχοντας APIs τα οποία εκθέτουν την πληροφορία τους με μια εμπλουτισμένη πλούσια

σημασιολογία, το καθιστά δυνατό στους πρώτους να ακολουθήσουν αυτόματα υπερσυνδέσμους, με τον ίδιο τρόπο περίπου που το κάνουμε και εμείς οι άνθρωποι. Αυτό συμβαίνει με το αντίστοιχο SWRL που ο server στέλνει. Παρόλα αυτά, αυτό που ουσιαστικά υλοποιεί τη ροή εργασιών που ο client πρέπει να ακολουθήσει είναι ένα FSM (Final State Machine). Οι Hypermedia web υπηρεσίες έχουν από τη φύση τους την έννοια της αυτόματης μηχανής πολύ στενά δεμένη μαζί τους, όπως τα οραματίστηκε ο Fielding αρχικά στην διατριβή του. Το SWRL επιλέχθηκε σαν γλώσσα σειριοποίησης για αυτά τα FSMs επειδή είναι ένα W3C πρότυπο και συνδυάζει την λογική των κανόνων με την OWL. Στην δική μας προσέγγιση, οι SWRL κανόνες είναι μία σειρά από βήματα, που μπορεί ο client να ακολουθήσει. Κάθε φορά που μία κατάσταση γίνεται ΑΛΗΘΗΣ μετά από μία συναλλαγή, ο client ανακαλύπτει την επόμενη κατάσταση που θα ακολουθήσει μέσα από τον hypermedia χάρτη του API. Στέλνει το επόμενο αίτημα στον server και εξαρτάται πλέον από τον ίδιο να βγάλει τα συμπεράσματά του πάνω σε αυτά. Μια τέτοιου είδους προσθήκη στο framework που παρουσιάζουμε μπορεί να προσαρμοστεί στο επίπεδο που περιγράφεται το κάθε δομικό στοιχείο από ένα συγκεκριμένο component και να επιφέρει αλλαγές σε συγκεκριμένο context της σελίδας λεπτομερούς παρουσίασης καθώς η αλλαγή του component μπορεί να γίνει δυναμικά και να αποτελέσει μια συγκεκριμένη αλλαγή. Μια τέτοιου είδους επέκταση μπορεί εύκολα να τροποποιηθεί και να είναι λειτουργική καθώς η προσέγγιση των Deep Graphs περιέχει όχι μόνο το state των δεδομένων και τα απαραίτητα μεταδεδομένα αλλά και τις τεχνολογίες των semantics που απαιτούνται για την λειτουργία της εφαρμογής μας. Ένα event το οποίο ενδέχεται να προσαρτηθεί μπορεί να περιέχει από μόνο του το απαραίτητο SPARQL ερώτημα αλλά και την κατάσταση του FSM αντιστοιχισμένη με τον component που θα την υλοποιήσει. Έτσι μια ολόκληρη υλοποίηση ενεργειών του χρήστη μπορεί να προσαρτηθεί χωρίς ο χρήστης να μεταβαίνει σε διαφορετικές διαδρομές της εφαρμογής και να αποτελέσει κομμάτι υλοποίησης σε επίπεδο Object σύμφωνα με την ανάλυση παρουσίασης που έγινε παραπάνω. Σαφώς και μια τέτοια παραμετροποίηση χρήζει επέκταση διαφορετικών σημείων της εφαρμογής αλλά με το δυναμικό rendering που παρουσιάζουμε και με την υλοποίηση κάθε ιδιότητας στο δικό της πεδίο εφαρμογής μια σειρά από ξεχωριστές λειτουργίες τέτοιου τύπου μπορούν να αποτελέσουν συγκεκριμένο κομμάτι παρουσίασης κάτω από μια ιδιότητα διασυνδεδεμένων δεδομένων που έχει οριστεί, και να αποτελέσει μια σειρά από λογικές λειτουργίες που ενσωματώνονται σε επίπεδο παρουσίασης των ιδιοτήτων της κάθε οντότητας.

Το κάθε σημείο λογικής που διαχωρίζεται στην εφαρμογή μπορεί να αποτελέσει κομβικό σημείο επέκτασης. Η απόκρυψη της λειτουργίας πίσω από μια σειρά κανόνων και προγραμματιστικής ακολουθίας καταστάσεων μπορεί να εφαρμοστεί παρουσιάζοντας στον χρήστη μια έξτρα λειτουργικότητα. Πίσω από μια προσθήκη ιδιότητας ενός μεταδεδομένου που χαρακτηρίζεται με το rdf και την owl, η επέκταση της λειτουργίας μπορεί να πάρει τεράστιες διαστάσεις περιγράφοντας με components και actions στο χρήστη τις απαραίτητες πληροφορίες για την αλληλεπίδραση. Για παράδειγμα μπορούμε να εφαρμόσουμε μια λειτουργία προσφοράς σημείων αγοραπωλησίας στον χρήστη με μια ιδιότητα χαρακτηρισμένη κάτω από ένα property που ορίζει ο rdf γράφος για παράδειγμα με ένα resource webc:BuyBook το οποίο περιγράφεται με ένα URI σαν ιδιότητα στη λεπτομερή παρουσίαση ενός Book. Με την ιδιότητα show more να παρουσιάζονται σε αυτών μια σειρά από λειτουργίες σχετικά με την αγορά και επιλογές με actions για τις αντίστοιχες διαδικασίες. Με την επιλογή τους ο χρήστης είναι ικανός να αλλάζει το context που παρουσιάζει το συγκεκριμένο resource

και να αλληλοεπιδρά με ένα Hypermedia API με την λογική των Deep Graphs που εκθέτει συγκεκριμένες πηγές και διαδικασίες ορισμένες σε ένα JSON-LD που αναγνωρίζεται από την εφαρμογή. Στη συνέχεια ο χρήστης μέσω των ενεργειών που του προσφέρονται να εκτελεί την επιθυμητή ενέργεια και να μεταβαίνει σε μια διαφορετική κατάσταση αλλάζοντας τον συγκεκριμένο component που την περιγράφει και έτσι να πλοηγείτε και να αλληλοεπιδρά σε μια λογική καταστάσεων που επιδρά σε συγκεκριμένο context της εφαρμογής.

Η υλοποίηση και ο προκαθορισμός των λειτουργιών που ενδέχεται να προσαρτηθούν σε μια υλοποίηση είναι στη φαντασία και στη λογική που θέλει να ακολουθήσει η κάθε εταιρεία ή ο προγραμματιστής. Σημαντικό είναι ότι με το δυναμικό rendering και την επεξεργασία της προσέγγισής μας η κάθε επέκταση καθίσταται εφικτή και σημαντικό κομμάτι σε αυτή αποτελεί η αρχιτεκτονική του flux που επιτρέπει στην εφαρμογή να διασπά τον κύκλο ζωής μιας διεργασίας σε διαφορετικά αυτόνομα κομμάτια που ενεργούν ξεχωριστά από τη συνολική λειτουργία. Η διαδικασία της παρουσίασης μπορεί εύκολα να επεκταθεί σε όλα τα σημεία αναπαράστασης της εφαρμογής με την υλοποίηση περαιτέρω δομικών στοιχείων που περιγράφουν κάθε τύπου λειτουργία. Σημαντικό είναι ένα σύγχρονο front-end να είναι σε θέση να επεξεργάζεται και να αναλύει τις τεχνολογίες του σημασιολογικού ιστού και αυτό προσπαθήσαμε να εκφράσουμε στην παρούσα εργασία με ένα δυναμικό τρόπο και μια ανεξαρτησία στα επίπεδα παρουσίασης της πληροφορίας.

Βιβλιογραφία

- *Reification in knowledge representation is the process of turning a predicate into an object. Hunt, Matthew (1996). "Notes on Semantic Nets and Frames" (PDF). Retrieved 15 June 2016.*

- *Managing the Life-Cycle of Linked Data with the LOD2 Stack https://link.springer.com/content/pdf/10.1007/978-3-642-35173-0_1.pdf*

- *Platforms/Frameworks , Editors and Validatorshttp://linkeddata.org/tools*

- *Khalili, A., and Auer, S. User interfaces for semantic authoring of textual content: A systematic literature review. Web Semantics: Science, Services and Agents on the World Wide Web 22 (Oct. 2013), 1–18.*

- *Pietriga, E., Bizer, C., Karger, D., and Lee, R. Fresnel: A browser-independent presentation vocabulary for RDF. In The Semantic Web - ISWC 2006, I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. M. Aroyo, Eds., no. 4273 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 2006, pp. 158–171.*

- Auer, S., Doehring, R., and Dietzold, S. LESS - template-based syndication and presentation of linked data. In The Semantic Web: Research and Applications, L. Aroyo, G. Antoniou, E. Hyv¨onen, A. t. Teije, H. Stuckenschmidt, L. Cabral, and T. Tudorache, Eds., no. 6089 in Lecture Notes in Computer Science

- Schlegel, K., Weißgerber, T., Stegmaier, F., Granitzer, M., and Kosch, H. Balloon Synopsis: A jQuery plugin to easily integrate the Semantic Web in a website. CEUR Workshop Proceedings 1268 (Oct. 2014).

- *Pellegrini, T., Mireles, V., Steyskal, S., Panasiuk, O., Fensel, A., &Kirrane, S. (2018). Automated Rights Clearance Using Semantic Web Technologies: The DALICC Framework. In Semantic Applications (pp. 203-218). Springer Vieweg, Berlin, Heidelberg. 13) Dumontier, M., Wilcke, X., Bloem, P., & de Boer, V. (2017): "The knowledge graph as the default data model for learning on heterogeneous knowledge." Data Science, 1(2), 39-57*

- *Dietzold, S., Hellmann, S., Peklo, M.: Using javascriptrdfa widgets for model/view separation inside read/write websites. In Proceedings of the 4th Workshop on Scripting for the Semantic Web. (2008)*

- *Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Sheets, D.: Tabulator: Exploring and analyzing linked data on the semantic web. In Proceedings of the 3rd International Semantic Web User Interaction Workshop (Vol. 2006). (2006)*

- *Quilitz, B., Leser, U.: Querying distributed RDF data sources with SPARQL. In The Semantic Web: Research and Applications, pp. 524-538. Springer Berlin Heidelberg (2008)*

- *Broekstra, J., Kampman, A., Van Harmelen, F.: Sesame: A generic architecture for storing and querying rdf and rdf schema. In The Semantic WebISWC, pp. 54-68. Springer Berlin Heidelberg (2002)*

References

[1] https://www.w3.org/Help/#webinternet
[2] T. Berners-Lee, J. Hendler and O. Lassila (2001) "The Semantic Web". Scientific American vol. 284 number 5, pp 34-43. Available on-line at http://www.scientificamerican.com/article.cfm?id=the-semantic-web.
[3] O. Lassila and R. Swick (1999) "Resource Description Framework (RDF) Model and Syntax Specification". Published on-line at http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.
[4] D. Brickley and R. V. Guha, "RDF Vocabulary Description Language 1.0: RDF Schema," *W3C Recommendation*, 2004.
[5] Alistair Miles, STFC Rutherford Appleton Laboratory / University of Oxford Sean Bechhofer, University of Manchester "SKOS Simple Knowledge Organization System Reference" Published on-line at https://www.w3.org/TR/skos-reference/
[6] E. Prud'hommeaux and A. Seaborne (2008) "SPARQL Query Language for RDF". Published on-line at http://www.w3.org/TR/rdf-sparql-query/.
[7] https://www.w3.org/TeamSubmission/n3/
[8] Turtle - Terse RDF Triple Language, Eric Prud'hommeaux, Gavin Carothers. World Wide Web Consortium, 10 July 2012. This version is http://www.w3.org/TR/2012/WD-turtle-20120710/. The latest version is http://www.w3.org/TR/turtle/
[9] *"MIME Media Types: text/turtle". Internet Assigned Numbers Authority (IANA). 28 March 2011. Retrieved 27 November 2011.*
[10] W3C OWL Working Group, "OWL 2 Web Ontology Language," *W3C Recommendation*, 2009. [Online]. Available: http://www.w3.org/TR/owl2- overview/.
[11] D. Reynolds, "OWL 2 RL in RIF (Second Edition)," *W3C Working Group Note*, 2013. [Online]. Available: http://www.w3.org/TR/2013/NOTE-rif-owl- rl-20130205
[12] T. Berners-Lee, "Linked Data," *Design Issues for the World Wide Web*, 2006. [Online]. Available: http://www.w3.org/DesignIssues/LinkedData.html.
[13] Sören Auer, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Amrapali Zaveri (2015) "Introduction to Linked Data and its Lifecycle on the Web". Introduction to Linked Data and its Lifecycle on the Web
[14] LOD2 Project, available at http://lod2.eu/
[15] Sauer, C.: WYSIWIKI - Questioning WYSIWYG in the Internet Age. In: Wikimania (2006)
[16] Adding social dimension to the Linked Data Web [Online]. Available: http://aksw.org/Projects/SemanticPingback.html
[17] Emmanuel Pietriga, Hande Gözükan, Caroline Appert, Marie Destandau, Šejla Čebirić, François Goasdoué, Ioana Manolescu Browsing Linked Data Catalogs with LODAtlas [Online]. Available: https://hal.inria.fr/hal-01827766/document
[18] CKAN, the world's leading Open Source data portal platform https://ckan.org/
[19] DataHub A Data Ecosystem for Individuals, Teams and People https://datahub.csail.mit.edu/www/
[20] "The Open Graph protocol," 2010. [Online]. Available: http://opengraphprotocol.org/. [Accessed: 07-Jul-2010].
[21] **Document Object Model** (**DOM**) https://en.wikipedia.org/wiki/Document_Object_Model
[22] Representational state transfer https://en.wikipedia.org/wiki/Representational_state_transfer
[23] Simple Object Access Protocol https://en.wikipedia.org/wiki/SOAP
[24] Fielding, Roy Thomas. "Architectural styles and the design of network-based software architectures." 2000
[25] Brunetti, J. M., Auer, S., Garc´ıa, R., Kl´ımek, J., and Necask ˇ y, M. ´ Formal linked data visualization model. In Proceedings of International Conference on Information Integration and Web-based Applications & Services (New York, NY, USA, 2013), IIWAS '13, ACM, p. 309:309–309:318.
[26] Hausenblas, M. (2009). Linked Data Applications. Technical Report, DERI, Galway.
[27] Use Case Diagrams https://www.uml-diagrams.org/use-case-diagrams.html
[28] Multitier architecture https://en.wikipedia.org/wiki/Multitier_architecture
[29] R2RML: RDB to RDF Mapping Language https://www.w3.org/TR/r2rml/

[30] http://www.ontotext.com/owlim
[31] TDB a component of Jena for RDF storage and query http://jena.apache.org/documentation/tdb/
[32] CumulusRDF https://code.google.com/archive/p/cumulusrdf/
[33] AllegroGraph, high-performance, persistent graph database https://franz.com/agraph/allegrograph/
[34] Data Virtualization platform, Virtuoso https://virtuoso.openlinksw.com/
[35] RDF storage and retrieval system https://code.google.com/archive/p/rdf3x/
[36] Hai H. Wang, Nick Gibbins, Terry Payne, Alina Patelli, and Yangang Wang. A survey of semantic web services formalisms. Concurrency and Computation: Practice and Experience, 27(15):4053--4072, 2015. 10.1002cpe.3481
[37] Martin G. Skjæveland. Sgvizler: A javascript wrapper for easy visualization of sparql result sets. In *9th Extended Semantic Web Conference (ESWC2012)*, May 2012
[38] Hai H. Wang, Nick Gibbins, Terry Payne, Alina Patelli, and Yangang Wang. A survey of semantic web services formalisms. *Concurrency and Computation: Practice and Experience*, 27(15):4053--4072, 2015. 10.1002cpe.3481
[39] Máire Casey and Claus Pahl. Web components and the semantic web. *Electr. Notes Theor. Comput. Sci.*, 82(5):156--163, 2003
[40] Olaf Hartig, Martin Kost, and Johann Christoph Freytag. Designing component-based semantic web applications with DESWAP. In Bizer and Joshi
[41] Ramón Hervás and José Bravo. Towards the ubiquitous visualization: Adaptive user-interfaces based on the semantic web. *Interacting with Computers*, 23(1):40--56, 2011
[42] Dominic Cooney. Introduction to web components, 2014. [Online].
Available: http://www.w3.org/TR/components-intro/
[43] I. Hickson, "HTML Microdata," *W3C Working Group Note*, 2013. [Online]. Available: http://www.w3.org/TR/2013/NOTE-microdata-20131029/.
[44] Flux Application architecture for building user interfaces [Online]. Available: https://facebook.github.io/flux/
[45] Fluxible Build isomorphic Flux applications https://fluxible.io/
[46] Semantic UI User Interface is the language of the web https://semantic-ui.com
[47] Sören Auer, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, and Amrapali Zaveri. Introduction to linked data and its lifecycle on the web. In *Proceedings of the 9th International Conference on Reasoning Web: Semantic Technologies for Intelligent Data Access*, RW'13, pages 1--90, Berlin, Heidelberg, 2013. Springer-Verlag.
[48] James Lewis and Martin Fowler. Microservices, 2014. http://martinfowler.com/articles/microservices.html.
[49] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," University of California, Irvine, 2000.