ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΕΥΦΥΗΣ ΕΦΑΡΜΟΓΗ ΤΗΣ ΑΡΧΗΣ ΕΛΑΧΙΣΤΟΠΟΙΗΣΗΣ ΔΕΔΟΜΕΝΩΝ

ΤΟΥΜΠΕΛΗΣ ΣΤΥΛΙΑΝΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Γεώργιος Λιουδάκης
Διδάσκων Π.Δ. 407/80

Λαμία, Οκτώβριος 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΕΥΦΥΗΣ ΕΦΑΡΜΟΓΗ ΤΗΣ ΑΡΧΗΣ ΕΛΑΧΙΣΤΟΠΟΙΗΣΗΣ ΔΕΔΟΜΕΝΩΝ

ΤΟΥΜΠΕΛΗΣ ΣΤΥΛΙΑΝΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Γεώργιος Λιουδάκης
Διδάσκων Π.Δ. 407/80

Λαμία, Οκτώβριος 2022

UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

INTELLIGENT APPLICATION OF THE PRINCIPLE OF DATA MINIMIZATION

TOUMPELIS STYLIANOS

FINAL THESIS

ADVISOR

Giorgios Lioudakis

Adjust Professor

Lamia, October 2022

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία: 3/11/2022

Ο Δηλ.

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση
του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων
σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΠΕΡΙΛΗΨΗ

Το ζήτημα της ιδιωτικότητας και προστασίας δεδομένων καθίσταται όλο και περισσότερο νομοθετημένο και είναι ένα από τα κρίσιμα ζητήματα που απασχολούν τις σύγχρονες κοινωνίες. Ο βαθμός συλλογής και επεξεργασίας που αυξάνεται ραγδαία με την εξέλιξη της τεχνολογίας αποτελεί τον κυριότερο λόγο. Για την αντιμετώπιση αυτών των ζητημάτων η Ευρωπαϊκή Ένωση ψήφισε τον Γενικό Κανονισμό Προστασίας Δεδομένων. Μεταξύ των βασικών αρχών του Κανονισμού είναι και η αρχή ελαχιστοποίησης δεδομένων.

Στόχος αυτής της πτυχιακής εργασίας είναι να παρουσιάσει μία τεχνική λύση στην εφαρμογή της αρχής ελαχιστοποίησης. Στο πλαίσιο αυτό εισάγει ένα καινοτόμο σύστημα το οποίο εκμεταλλεύεται τις τεχνολογίες σημασιολογικού ιστού προκειμένου να παίρνονται σε πραγματικό χρόνο αποφάσεις για το ποια είναι τα δεδομένα που κατά περίπτωση ικανοποιούν την αρχή της ελαχιστοποίησης, δηλαδή είναι τα κατάλληλα, συναφή και αναγκαία για τους σκοπούς για τους οποίους υποβάλλονται σε επεξεργασία.

.

ABSTRACT

The issue of privacy and data is becoming increasingly legislated and is one of the critical issues that concern modern societies. The degree and processing that has been collected with the development of technology is the main reason. To address these issues, the European Union passed the General Data Protection Regulation. Among the basic principles of the Regulation is the principle of data minimization.

The aim of this thesis is to present a technical solution to the application of the minimization principle. In this context, it introduces an innovative system that exploits semantic web technologies in order to make real-time decisions about which data satisfy the minimization principle, i.e. are appropriate, relevant and necessary for the purposes for which they are submitted to processing.

ΕΥΧΑΡΙΣΤΙΕΣ

Η ολοκλήρωση της παρούσας πτυχιακής εργασίας θα ήταν αδύνατη χωρίς την υποστήριξη του καθηγητή μου, Λιουδάκη Γεώργιου. Του εκφράζω ένα βαθύ ευχαριστώ για όλη τη βοήθεια που μου προσέφερε. Θέλω, επίσης, να ευχαριστήσω βαθιά την μητέρα μου Ειρήνη και την γιαγιά μου Ευαγγελία οι οποίες υπήρξαν πάντα ένα ανεκτίμητο στήριγμα για μένα και στις οποίες οφείλω όλη τη διαδρομή των σπουδών μου, μέχρι σήμερα. Τέλος, θέλω να ευχαριστήσω πολύ τα αδέρφια μου και τους κοντινούς μου ανθρώπους, για την άμετρη συμπαράσταση που έδειξαν καθόλη τη διάρκεια των προπτυχιακών μου σπουδών.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΙΝΑΚΑΣ ΣΧΗΜΑΤΩΝ

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή

Το ζήτημα της ιδιωτικότητας και προστασίας δεδομένων απασχολεί όλο και περισσότερο τις σύγχρονες κοινωνίες, με αποτέλεσμα, μεταξύ άλλων, να καθίσταται όλο και περισσότερο αντικείμενο νομοθέτησης [1]. Όπως αναφέρεται χαρακτηριστικά «η ιδιωτικότητα είναι η δικαιολογητική βάση για την απαγόρευση της κρατικής παρέμβασης στον ιδιωτικό και οικογενειακό βίο ενώ εμφανίζεται σαν το δικαιοπολιτικό θεμέλιο για την ελευθερία της σκέψης, τις προσωπικές επιλογές του ατόμου για κάθε πτυχή της ιδιωτικής του ζωής. Στο σημείο αυτό συνδέεται και με τον περιορισμό της χρήσης δεδομένων προσωπικού χαρακτήρα» [2]. Η μετάβαση στην ψηφιακή εποχή, όπου η οικονομία βασίζεται στις πληροφορίες και την ψηφιοποίησή τους, το δικαίωμα της ιδιωτικότητας θίγεται σε μεγάλο βαθμό. Επιπλέον η εξέλιξη των τεχνολογιών και του Διαδικτύου έχει οδηγήσει στην λεγόμενη «δεδομενοποίηση» των κοινωνιών, δηλαδή στην αυξανόμενη μετατροπή πτυχών της καθημερνής ζωής των ανθρώπων σε δεδομένα μέσα από την συλλογή, επεξεργασία και αποθήκευση αυτών.

Απάντηση στην εξέλιξη των τεχνολογιών και «δεδομενοποίηση» αποτέλεσε ο Γενικός Κανονισμός Προστασίας Δεδομένων για την προστασία των φυσικών προσώπων έναντι της επεξεργασίας των δεδομένων προσωπικού χαρακτήρα και για την ελεύθερη κυκλοφορία των δεδομένων αυτών [3]. Ο Κανονισμός θέσπισε κάποιες αρχές που διέπουν την επεξεργασία δεδομένων προσωπικού χαρακτήρα, ανάμεσα σε αυτές είναι ότι τα δεδομένα προσωπικού χαρακτήρα πρέπει να είναι «επαρκή, συναφή και περιορισμένα σε τι είναι απαραίτητο σε σχέση με τους σκοπούς για τους οποίους υποβάλλονται σε επεξεργασία». Αυτό αποτελεί την «αρχή της ελαχιστοποίησης δεδομένων».

Η αρχή της ελαχιστοποίησης δεδομένων προβλέπει ότι ένας υπεύθυνος επεξεργασίας δεδομένων πρέπει να περιορίζει τη συλλογή προσωπικών πληροφοριών σε ό,τι είναι άμεσα σχετικό και απαραίτητο για την επίτευξη ενός συγκεκριμένου σκοπού. Με άλλα λόγια, οι υπεύθυνοι επεξεργασίας δεδομένων θα πρέπει να συλλέγουν μόνο τα προσωπικά δεδομένα που πραγματικά χρειάζονται.

Η παρούσα πτυχιακή εργασία έχει ως στόχο να δώσει μια τεχνική λύση στην εφαρμογή της αρχής ελαχιστοποίησης μέσα από ένα καινοτόμο σύστημα που αποφασίζει ποια είναι τα ελάχιστα δεδομένα που μπορούν να εξυπηρετήσουν συγκεκριμένους σκοπούς. Στο πλαίσιο αυτό το σύστημα χρησιμοποιεί ένα σημασιολογικό μοντέλο που εκφράζει τους τύπους προσωπικών δεδομένων, δημιουργώντας έτσι μια οντολογία, και τις συσχετίσεις τους με διάφορες υπηρεσίες στις οποίες μπορεί να αυτά ζητηθούν για την εξυπηρέτησή τους. Προκειμένου να γίνει λήψη αποφάσεων σε πραγματικό χρόνο χρησιμοποιείται ένας αλγόριθμος που εκμεταλλεύεται τις συσχετίσεις του μοντέλου και αποφασίζει εάν αυτά είναι τα ελάχιστα ή αποφασίζει εάν θα χρειαστεί να φιλτράρει τα δεδομένα που δόθηκαν ώστε να βρει μέσα από αυτά τα ελάχιστα δεδομένα για την εξυπηρέτηση της υπηρεσίας.

(Υποκεφάλαιο 1.1) Διάρθρωση της Διατριβής

Στο Κεφάλαιο 2 πραγματοποιείται η επισκόπηση κάποιων βασικών εννοιών και τεχνολογιών που αποτέλεσαν το υπόβαθρο για την εργασία. Συγκεκριμένα, πρώτα μελετάται η σχετική νομοθεσία, με έμφαση στον Γενικό Κανονισμό Προστασίας Δεδομένων και την αρχή της ελαχιστοποίησης. Στη συνέχεια, παρουσιάζονται οι σημασιολογικές τεχνολογίες με έμφαση στο RDF που αποτελεί το βασικό τεχνολογικό εργαλείο.

Στο Κεφάλαιο 3 παρουσιάζεται η περιγραφή του συστήματος ελαχιστοποίησης δεδομένων. Πιο συγκεκριμένα γίνεται ανάλυση του μοντέλου RDF, που περιγράφει τις σχέσεις δεδομένων προσωπικού χαρακτήρα βάσει του οποίου το σύστημα λαμβάνει αποφάσεις, και παρουσιάζεται μία αναλυτική ανάλυση της αρχιτεκτονικής του και της ροής που ακολουθεί στην εκτέλεσή του.

Πυρήνα του συστήματος αποτελεί ο αλγόριθμος ελαχιστοποίησης δεδομένων που παρουσιάζεται στο Κεφάλαιο 4. Το Κεφάλαιο εστιάζει στην αναλυτική περιγραφή και ανάλυση του αλγόριθμου που είναι αυτός που δίνει ως αποτέλεσμα ποια είναι τα ελάχιστα δεδομένα σε κάθε περίπτωση ανάλογα με τον υποκείμενο σκοπό.

Στο Κεφάλαιο 5 περιγράφεται η αναλυτικά η υλοποίηση του συστήματος σε γλώσσα προγραμματισμού Java.

Στο Κεφάλαιο 6 παρουσιάζεται ένα παράδειγμα ενός σεναρίου του πραγματικού κόσμου μέσα στο οποίο θα μπορούσε να γίνει χρήση του συστήματος και στην συνέχεια γίνεται μία τεχνική υλοποίηση του σεναρίου, πώς δηλαδή θα λειτουργήσει το σύστημα στο ίδιο αυτό σενάριο που περιγράφεται.

Τέλος, το Κεφάλαιο 7 συνοψίζει κάποια συμπεράσματα, καθώς και τις προοπτικές για μελλοντική εργασία.

ΚΕΦΑΛΑΙΟ 2 Υπόβαθρο

(Υποκεφάλαιο 2.1) Υποκείμενη Νομοθεσία

(Ενότητα 2.1.1) Ιστορική Αναδρομή

Η ιδιωτικότητα αναγνωρίστηκε ως ένα από τα θεμελιώδη δικαιώματα του ανθρώπου από τα Ηνωμένα Έθνη το 1948, με το Άρθρο 12 της Οικουμενικής Διακήρυξης για τα Ανθρώπινα Δικαιώματα [4]. Η ανάπτυξη και η εξέλιξη των τεχνολογιών παρακολούθησης, συλλογής, επεξεργασίας και αρχειοθέτησης πληροφοριών οδήγησαν στην υιοθέτηση σχετικής νομοθεσίας σε πολλές χώρες του κόσμου, φτάνοντας στις μέρες μας σε παγκόσμια σχεδόν κάλυψη. Ο πρώτος σύγχρονος νόμος προστασίας δεδομένων υιοθετήθηκε από το Hesse της Δυτικής Γερμανίας το 1970 [5] και ο πρώτος εθνικός νόμος προστασίας προσωπικών δεδομένων το 1973 στην Σουηδία. Την επόμενη χρονιά το 1974, το κογκρέσο των ΗΠΑ υιοθέτησε το «Νόμο περί Ιδιωτικότητας» (Privacy Act) [6] ο οποίος ρύθμισε τη συλλογή, τη διατήρηση, τη χρήση και τη διάδοση προσωπικών πληροφοριών που διατηρούνται σε συστήματα αρχείων από ομοσπονδιακές υπηρεσίες.

Ορόσημο στην ιδιωτικότητα και την προστασία των δεδομένων αποτέλεσαν οι «Κατευθυντήριες Οδηγίες για την Προστασία της Ιδιωτικότητας και τη Διασυνοριακή Ροή των Προσωπικών Δεδομένων» (Guidelines on the Protection of Privacy and Transborder Flows of Personal Data) [7] που εξέδωσε ο Οργανισμός Οικονομικής Συνεργασίας και Ανάπτυξης (ΟΟΣΑ) το 1980, οι οποίες έχοντας συμβουλευτική και όχι νομοθετική φύση, είχαν σκοπό να εναρμονίσουν τις νομοθεσίες που ακολούθησαν γύρω από μία κοινή βάση σχετικά με τα προστασία των προσωπικών δεδομένων στα κράτη-μέλη του Οργανισμού.

Αργότερα, το 1995 η Ευρωπαϊκή Κοινότητα εξέδωσε την Οδηγία 95/46/ΕΚ «για την προστασία των φυσικών προσώπων έναντι της επεξεργασίας δεδομένων προσωπικού χαρακτήρα και για την ελεύθερη διακίνηση των δεδομένων αυτών» [8], βασικός στόχος της οποίας ήταν η προστασία των θεμελιωδών δικαιωμάτων και ελευθεριών των φυσικών προσώπων όσον αφορά τις δραστηριότητες επεξεργασίας και τη διασφάλιση της ελεύθερης κυκλοφορίας δεδομένων προσωπικού χαρακτήρα μεταξύ κρατών-μελών.

Η Οδηγία 95/46/ΕΚ καταργήθηκε στις 25 Μαΐου του 2018 με την εφαρμογή του Γενικού Κανονισμού Προστασίας Δεδομένων (GDPR - 2016/679) της Ευρωπαϊκής Ένωσης σε όλα τα κράτη-μέλη, και αποτελεί τη μεγαλύτερη αλλαγή στην νομοθεσία περί προστασίας των δεδομένων και συντέλεσε στην ουσιαστικότερη και αποτελεσματικότερη προστασία της ιδιωτικότητας [3].

(Ενότητα 2.1.2) Κανονισμός (ΕΕ) 2016/679 (GDPR)

Ο Γενικός Κανονισμός Προστασίας Δεδομένων (GDPR – 2016/679) [3] αποτελεί ένα κοινό πλαίσιο ρυθμίσεων για τον τρόπο με τον οποίο συλλέγονται, επεξεργάζονται, φυλάσσονται, διακινούνται, αξιοποιούνται αλλά και καταστρέφονται, δεδομένα προσωπικού χαρακτήρα των πολιτών της Ευρωπαϊκής Ένωσης, ανεξαρτήτως του τόπου διαμονής τους, τόσο σε ηλεκτρονική όσο και σε φυσική μορφή.

Το αντικείμενο και οι στόχοι του Κανονισμού είναι η θέσπιση κανόνων που αφορούν την προστασία των φυσικών προσώπων έναντι της επεξεργασίας δεδομένων προσωπικού χαρακτήρα και την ελεύθερη κυκλοφορία των δεδομένων προσωπικού χαρακτήρα.

Ο Κανονισμός εφαρμόζεται στην επεξεργασία δεδομένων προσωπικού χαρακτήρα στο πλαίσιο δραστηριοτήτων ενός υπευθύνου επεξεργασίας ή εκτελούντος την επεξεργασία στην Ένωση, ανεξάρτητα από το κατά πόσο η επεξεργασία πραγματοποιείται εντός της Ένωσης, αλλά και ανεξάρτητα από το εάν ο υπεύθυνος επεξεργασίας ή εκτελών την επεξεργασία βρίσκεται εγκατεστημένος στην Ένωση. Μια πολύ βασική πτυχή του Κανονισμού είναι ο προσδιορισμός των βασικών αρχών που διέπουν την επεξεργασία δεδομένων προσωπικού χαρακτήρα, οι οποίες παρουσιάζονται στην επόμενη ενότητα, καθώς και τις απαραίτητες προϋποθέσεις προκειμένου η όποια επεξεργασία να είναι σύννομη. Πέραν αυτών, ο Κανονισμός ορίζει τις προϋποθέσεις που αφορούν τη συγκατάθεση των υποκειμένων των δεδομένων και την εγκυρότητά της, αλλά και σειρά δικαιωμάτων των υποκειμένων των δεδομένων, όπως είναι η διαφανής ενημέρωση, τα δικαιώματα πρόσβασης, διόρθωσης, διαγραφής, εναντίωσης και περιορισμού της επεξεργασίας, το δικαίωμα στη φορητότητα, και δικαιώματα σε σχέση με την αυτοματοποιημένη ατομική λήψη αποφάσεων, περιλαμβανομένης της κατάρτισης προφίλ.

Ο Κανονισμός περιλαμβάνει σειρά κανόνων που αφορούν τις υποχρεώσεις των υπευθύνων επεξεργασίας και των εκτελούντων την επεξεργασία, συμπεριλαμβανομένων του ορισμού Υπευθύνου Επεξεργασίας Δεδομένων, της τήρησης αρχείων δραστηριοτήτων επεξεργασίας, της διενέργειας εκτίμησης αντικτύπου προστασίας δεδομένων, καθώς και της λήψης κατάλληλων οργανωτικών και τεχνικών μέτρων για την ασφάλεια των δεδομένων και της επεξεργασίας τους.

(Ενότητα 2.1.3) Βασικές Αρχές του Γενικού Κανονισμού Προστασίας Δεδομένων

Ο κανονισμός θέτει επτά θεμελιώδεις αρχές σχετικά με τη συλλογή και επεξεργασία των δεδομένων, καθώς και τη νομική βάση πάνω στην οποία μπορούν να λάβει χώρα. Πιο συγκεκριμένα, οι βασικές αρχές είναι οι εξής:

1. Νομιμότητα, δικαιοσύνη και διαφάνεια

Ο λόγος συλλογής δεδομένων πρέπει να καθοριστεί σε νομική βάση. Ο GDPR προβλέπει διάφορους λόγους, όπως η επεξεργασία με βάση τη συγκατάθεση, το δημόσιο συμφέρον ή τα συμφέροντα του νόμου. Πρέπει να υπάρχει διαφάνεια, για παράδειγμα, σχετικά με τα δεδομένα που συλλέγονται, για τι σκοπό, για ποιον και για πόσο καιρό θα διατηρηθούν. Αυτές οι πληροφορίες πρέπει να είναι γραμμένες όσο το δυνατόν πιο ξεκάθαρα σε μια εύκολα κατανοητή γλώσσα.

2. Περιορισμός σκοπού

Τα δεδομένα προσωπικού χαρακτήρα συλλέγονται για καθορισμένους, σαφείς και νόμιμους σκοπούς και δεν μπορούν να υποστούν περαιτέρω επεξεργασία με τρόπο ασυμβίβαστο με αυτούς τους σκοπούς. Τουλάχιστον ένας από τους λόγους που αναφέρονται για τη συλλογή δεδομένων πρέπει να πληρούνται ώστε να του επιτραπεί η επεξεργασία. Αυτό σημαίνει ότι στο πρέπει να προσδιορίζεται για ποιους σκοπούς συλλέγονται δεδομένα. Ορισμένες άλλες ειδικές προϋποθέσεις ενδέχεται να ισχύουν. Ωστόσο, σε ορισμένες περιπτώσεις, τα δεδομένα μπορούν να υποβληθούν σε επεξεργασία για νέους σκοπούς που είναι συμβατοί με τον αρχικούς με την προϋπόθεση ότι το υποκείμενο των δεδομένων συναινεί για τον νέο σκοπό.

3. Ελαχιστοποίηση δεδομένων

Τα προσωπικά δεδομένα θα πρέπει να είναι επαρκή, σχετικά και να περιορίζονται στα απαραίτητα. Θα πρέπει να προσδιοριστεί ο ελάχιστος αριθμός προσωπικών δεδομένων που απαιτούνται για την επεξεργασία, πριν από οποιαδήποτε συλλογή δεδομένων. Ως εκ τούτου, τα δεδομένα θα πρέπει να είναι επαρκή, σχετικά ή να έχουν σαφή σύνδεση με τον σκοπό και να περιορίζονται στην εκπλήρωση των σκοπών τα οποία υποβάλλονται σε επεξεργασία.

4. Ακρίβεια

Τα δεδομένα πρέπει να είναι ακριβή και να διατηρούνται ενημερωμένα. Υπάρχει υποχρέωση για τα δεδομένα ο υπεύθυνος επεξεργασίας να διασφαλίζει προληπτικά την ακρίβεια των δεδομένων και εάν κάποιο από αυτά είναι ανακριβές, λανθασμένο ή παραπλανητικό, τότε είτε να το διαγράφει είτε να το διορθώνει χωρίς καθυστέρηση. Σε ορισμένες περιπτώσεις το ο υπεύθυνος επεξεργασίας μπορεί να βασιστεί στις απαιτήσεις των υποκειμένων των δεδομένων, για παράδειγμα εάν θέλει η διεύθυνσή τους να ενημερωθεί στη βάση δεδομένων.

5. Περιορισμός αποθήκευσης

Τα προσωπικά δεδομένα θα πρέπει να διατηρούνται για καθορισμένο χρονικό διάστημα και όχι περισσότερο από όσο χρειάζεται. Όταν ο σκοπός διατήρησης των δεδομένων δεν είναι πλέον σχετικός ή είναι ξεπερασμένος τα δεδομένα θα πρέπει να διαγραφούν ή να ανωνυμοποιηθούν.

6. Ακεραιότητα και εμπιστευτικότητα

Τα προσωπικά δεδομένα θα πρέπει να διατηρούνται ασφαλή. Πρέπει να ληφθούν τα κατάλληλα μέτρα για να εξασφαλιστεί η ασφάλεια των δεδομένων, συμπεριλαμβανομένης της προστασίας από μη εξουσιοδοτημένη ή παράνομη επεξεργασία και από τυχαία απώλεια, καταστροφή ή ζημιά.

7. Ευθύνη

Η αρχή της λογοδοσίας καθιστά τον υπεύθυνο επεξεργασίας δεδομένων υπεύθυνο και συνεπώς υπεύθυνο για τη συμμόρφωση με τον GDPR, λαμβάνοντας όλα τα απαραίτητα μέτρα.

(Ενότητα 2.1.4) Αρχή Ελαχιστοποίησης Δεδομένων

Αντικείμενο του συστήματος που αναπτύχθηκε στο πλαίσιο της παρούσας πτυχιακής εργασίας αποτελεί η αρχή ελαχιστοποίησης δεδομένων.

Οι απαιτήσεις της αρχής συνοψίζονται στο άρθρο 5 παράγραφος 1 στοιχείο γ) [3] του Γενικού Κανονισμού Προστασίας Δεδομένων το οποίο ορίζει ότι «[τα δεδομένα προσωπικού χαρακτήρα] είναι κατάλληλα, συναφή και περιορίζονται στο αναγκαίο για τους σκοπούς για τους οποίους υποβάλλονται σε επεξεργασία («ελαχιστοποίηση των δεδομένων»)». Επομένως, θα πρέπει να προσδιορίζονται τα κατάλληλα προσωπικά δεδομένα που χρειάζονται. Θα πρέπει να δίνεται επαρκής αλλά ελάχιστη πληροφορία.

Στο ερώτημα ποια δεδομένα είναι κατάλληλα, συναφή και περιορισμένα η απάντηση εξαρτάται από τον σκοπό για τον οποίο συλλέγονται και επεξεργάζονται τα προσωπικά δεδομένα. Για να αξιολογηθεί εάν διατίθεται ο ελάχιστος όγκος προσωπικών δεδομένων, πρέπει πρώτα να γίνεται ξεκάθαρος ο σκοπός για τον οποίο χρειάζονται. Επομένως, τα προσωπικά δεδομένα που θα διατεθούν δεν πρέπει να είναι περισσότερα από όσα απαιτούνται για την εκπλήρωση του σκοπού καθώς και να μην περιλαμβάνουν μη σχετικές λεπτομέρειες. Εάν διατίθενται περισσότερα δεδομένα από όσα είναι πραγματικά απαραίτητα για τον σκοπό, πιθανόν να είναι παράνομο (καθώς οι περισσότερες νόμιμες βάσεις έχουν ένα στοιχείο αναγκαιότητας) αφού παραβιάζεται η αρχή της ελαχιστοποίησης δεδομένων.

Εν κατακλείδι, σύμφωνα με την αρχή ελαχιστοποίησης δεδομένων πρέπει να συλλέγονται τα ελάχιστα επαρκή και σχετικά προσωπικά δεδομένα που χρειάζονται για συγκεκριμένους σκοπούς.

Οι απαιτήσεις της αρχής ελαχιστοποίησης δεδομένων του Γενικού Κανονισμού Προστασίας Δεδομένων, αποτέλεσαν την βάση στην σχεδίαση της εφαρμογής ελαχιστοποίησης δεδομένων που κατασκευάστηκε στην παρούσα πτυχιακή.

(Υποκεφάλαιο 2.2) Σημασιολογικές Τεχνολογίες

(Ενότητα 2.2.1) Σημασιολογικός Ιστός

Σημασιολογικός Ιστός όπως όρισε ο δημιουργός του Παγκόσμιου Ιστού Tim Berners-Lee είναι «ένας ιστός από δεδομένα που μπορούν να επεξεργαστούν απ' ευθείας από υπολογιστές και προγράμματα». Ο Σημασιολογικός Ιστός επινοήθηκε ως μία επέκταση του Παγκόσμιου Ιστού από τον ίδιο τον Tim Berners-Lee με σκοπό να βελτιώσει τον τρόπο με τον οποίο χρησιμοποιούνται, αναζητούνται και επεξεργάζονται πληροφορίες σε αυτό, μετατρέποντας σταδιακά τον υπάρχοντα Παγκόσμιο Ιστό από ιστό εγγράφων (Web of Documents) σε ιστό δεδομένων (Web of Data). Σε αυτό τον Ιστό (Web) συνδέονται μεταξύ τους πληροφορίες και περιεχόμενο ιστού εμπλουτισμένο και δικτυωμένο από μηχανικά-αναγνώσιμα σημασιολογικά μεταδεδομένα όπου η ανταλλαγή πληροφοριών βελτιστοποιείται επιτρέποντας στις μηχανές να διακρίνουν τα νοήματα (σημασιολογικό περιεχόμενο) και να τα επεξεργάζονται [9]. Για την μετατροπή του Παγκόσμιου Ιστού σε έναν ιστό δεδομένων, έχουν προταθεί μια σειρά σχετικών τεχνολογιών που μπορούν να χρησιμοποιηθούν για την κωδικοποίηση γνώσης έτσι ώστε να μπορεί να επεξεργαστεί από υπολογιστές. Πιο συγκεκριμένα, κάποια από τα βασικά θέματα στα οποία στηρίζεται το όραμα του Σημασιολογικού Ιστού και οι τεχνολογίες του είναι η κατασκευή αφηρημένων μοντέλων που αναπαριστούν πτυχές του πραγματικού κόσμου, η ανάπτυξη μηχανών συλλογισμού για την εξαγωγή χρήσιμων συμπερασμάτων από κωδικοποιημένη γνώση και η δυνατότητα ανταλλαγής ετερογενούς πληροφορίας σε ευρεία κλίμακα. Σήμερα, ο Σημασιολογικός Ιστός αποτελεί ένα ιδιαίτερο πεδίο της Επιστήμης των Υπολογιστών που αντλεί και συνδυάζει γνώση προερχόμενη από άλλα περισσότερο ώριμα πεδία, όπως η τεχνητή νοημοσύνη, οι βάσεις δεδομένων και τα δίκτυα υπολογιστών, και του οποίου οι βασικές τεχνολογίες χρησιμοποιούνται και σε εφαρμογές και περιβάλλοντα εκτός του Παγκόσμιου Ιστού.

Ο Σημασιολογικός Ιστός μπορεί να επιτευχθεί μόνο αν το περιεχόμενο που υπάρχει στον Ιστό αποκτήσει τυπικό αυστηρό νόημα, κωδικοποιημένο σε κάποιο μορφότυπο ικανό για επεξεργασία από υπολογιστές [10]. Για τη διαδικασία αυτή δεν έχει αναπτυχθεί μια σειρά κατάλληλων τεχνολογιών και προτύπων, γνωστών και ως τεχνολογιών Σημασιολογικού Ιστού, από το W3C (World Wide Web Consortium), τον οργανισμό που είναι υπεύθυνος για την προώθηση ανοικτών προτύπων για τον Παγκόσμιο Ιστό. Οι τεχνολογίες Σημασιολογικού Ιστού που χρησιμοποιήθηκαν περιγράφονται αναλυτικά στις επόμενες ενότητες αυτού του κεφαλαίου.

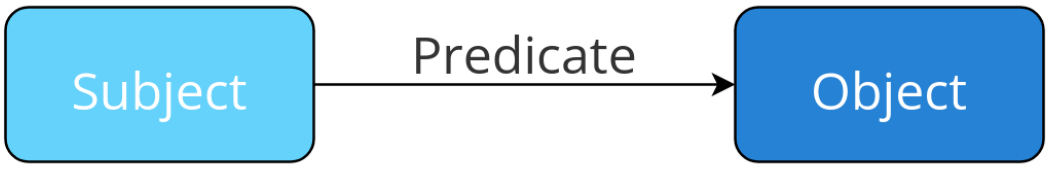
(Ενότητα 2.2.2) Μοντέλο RDF

Το Resource Description Framework (RDF) είναι ένα πρότυπο για την περιγραφή δομημένων πληροφοριών Σημασιολογικού Ιστού και ανταλλαγής δεδομένων, καθώς και για την κωδικοποίηση σημασιολογικών σχέσεων μεταξύ αυτών των δεδομένων, έτσι ώστε αυτές οι σχέσεις να μπορούν να ερμηνευτούν υπολογιστικά. Τα δεδομένα αυτά ονομάζονται πόροι (resources). Οι πόροι μπορεί να είναι οτιδήποτε, συμπεριλαμβανομένων εγγράφων, ανθρώπων, φυσικών αντικειμένων και αφηρημένων εννοιών. Το RDF προορίζεται για καταστάσεις στις οποίες πληροφορίες πρέπει να υποβάλλονται σε επεξεργασία από εφαρμογές. Το RDF παρέχει ένα μοντέλο για την έκφραση αυτών των πληροφοριών, ώστε να μπορεί να ανταλλάσσεται μεταξύ εφαρμογών χωρίς απώλεια νοήματος. Το μοντέλο αυτό είναι βασισμένο σε γράφο, και συγκεκριμένα σε κατευθυνόμενο γράφο, επομένως αποτελείται από κόμβους και ακμές. Οι ακμές του RDF γράφου εκφράζονται ως

τριάδες(triples) ή αλλιώς προτάσεις(statements), ενώ οι κόμβοι εκφράζουν τους πόρους του μοντέλου [11].

(Ενότητα 2.2.3) Μοντέλο RDF: Τριάδες

Μια πρόταση RDF εκφράζει μια σχέση μεταξύ δύο πόρων. Το υποκείμενο(subject) και το αντικείμενο (object) αντιπροσωπεύουν τους δύο πόρους που σχετίζονται. Το κατηγόρημα (predicate) παρουσιάζει τη σχέση τους. Η σχέση διατυπώνεται με κατευθυντικό τρόπο (από υποκείμενο σε αντικείμενο) και επειδή αποτελούνται από τρία στοιχεία ονομάζονται τριάδες(triple) [11]. Μία πρόταση έχει πάντα την δομή που παρουσιάζεται στο Σχήμα 1. Μια RDF τριάδα ουσιαστικά δηλώνει μια συσχέτιση, η οποία υποδεικνύεται από το κατηγόρημα, μεταξύ των αντικειμένων ή εννοιών που υποδηλώνονται από το υποκείμενο και το αντικείμενο της τριάδας. Συχνά, το κατηγόρημα μιας RDF πρότασης αναφέρεται και ως *ιδιότητα* (property).



Σχήμα 1: Παράδειγμα τριάδας RDF

Μερικά παραδείγματα τριάδων (triples):

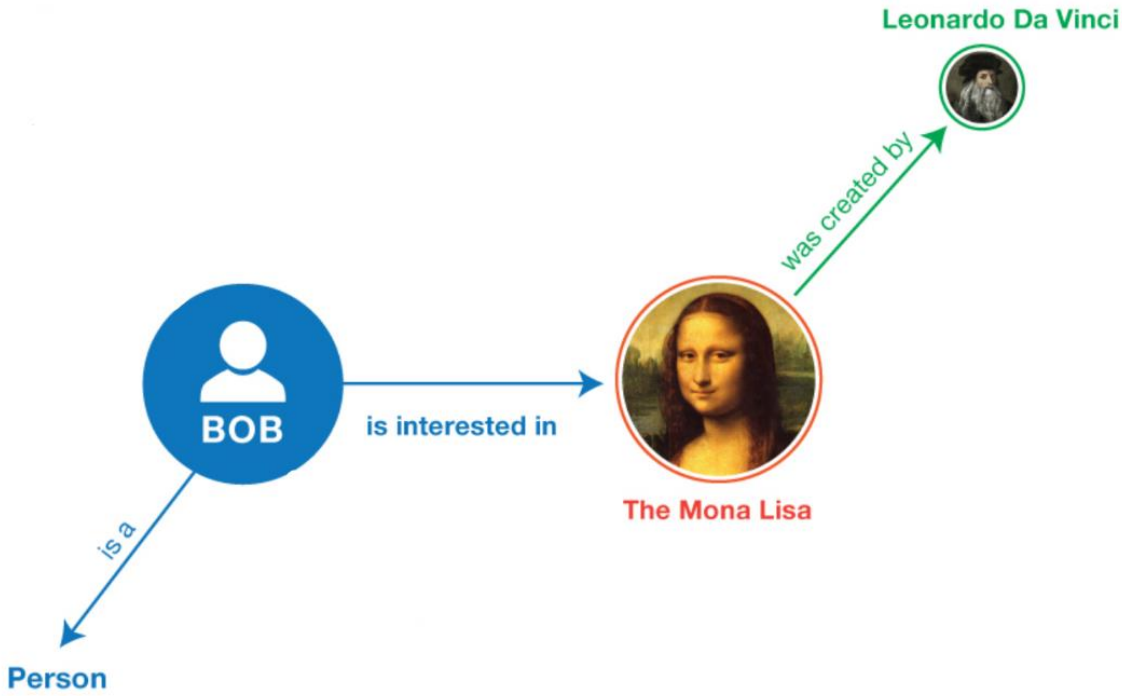
```
<Bob> <is a> <person>.

<Bob> <is interested in> <the Mona Lisa>.

<the Mona Lisa> <was created by> <Leonardo da Vinci>.
```

Ο ίδιος πόρος μπορεί να αναφέρεται συχνά σε πολλαπλές τριάδες. Στο παραπάνω παράδειγμα, ο Bob είναι το υποκείμενο δύο τριάδων και η Mona Liza είναι το αντικείμενο μίας τριάδας και το υποκείμενο μίας άλλης τριάδας. Η δυνατότητα να υπάρχει ο ίδιος πόρος ως υποκείμενο μίας τριάδας και ως αντικείμενο μίας άλλης καθιστά δυνατή την εύρεση σχέσεων μεταξύ τριάδων, κάτι που είναι σημαντικό μέρος των δυνατότητων του RDF.

Το Σχήμα 2 δείχνει το RDF γράφο που προκύπτει από το παράδειγμα τριάδας που δόθηκε παραπάνω.



Σχήμα 2: Παράδειγμα RDF [11]

(Ενότητα 2.2.4) Μοντέλο RDF: IRIs

Όλοι οι πόροι (δηλαδή οι κόμβοι και οι ιδιότητες) στο RDF μοντέλο προσδιορίζονται από ένα αναγνωριστικό, γνωστό και ως IRI που είναι συντομογραφία του "International Resource Identifier" [11]. Χρησιμοποιούνται για την αναγνώριση πόρων όπως έγγραφα, άτομα, φυσικά αντικείμενα και αφηρημένες έννοιες. Τα IRI είναι αναγνωριστικά, επομένως μπορεί να χρησιμοποιηθεί ξανά αυτό το IRI για να αναγνωρίσουν το ίδιο πράγμα. Για παράδειγμα, το ακόλουθο IRI χρησιμοποιείται ως ιδιότητα RDF για να δηλώσει μια σχέση γνωριμίας μεταξύ ανθρώπων: foaf:knows.

Τα IRI σε ένα μοντέλο RDF συχνά ξεκινούν με μια κοινή υποσυμβολοσειρά που είναι γνωστή ως namespace IRIs. Ορισμένα namespace IRIs συσχετίζονται με ένα σύντομο ονομαστικό γνωστό ως πρόθεμα(prefix).

Είναι σύνηθες να συντομεύονται τα IRI που ξεκινούν με τα namespace IRIs χρησιμοποιώντας ένα namespace prefix για να διευκολύνεται η αναγνωσιμότητα.

(Ενότητα 2.2.5) Μοντέλο RDF: Πολλαπλοί Γράφοι

Το μοντέλο RDF παρέχει έναν μηχανισμό για την ομαδοποίηση των δηλώσεων σε παραπάνω από έναν γράφο και τη συσχέτιση αυτών των γράφων μεταξύ τους με IRIs [11].

Για παράδειγμα, οι δηλώσεις στο παραπάνω παράδειγμα θα μπορούσαν να ομαδοποιηθούν σε δύο επώνυμους γράφους.

Ο πρώτος γράφος θα περιέχει τις δηλώσεις:

<Bob> <is a> <person>.

<Bob> <is interested in> <the Mona Lisa>.

Ο δεύτερος γράφος θα περιέχει τις δηλώσεις:

<the Mona Lisa> <was created by> <Leonardo da Vinci>.

Μεταξύ τους ενώνονται με την τριάδα: <Bob> <is interested in> <the Mona Lisa>.

(Ενότητα 2.2.6) Μοντέλο RDF: RDF Schema (RDFS)

Το RDF μοντέλο παρέχει δυνατότητα έκφρασης απλών δηλώσεων για κάποιο αντικείμενο ή έννοια, με τη χρήση προσδιορισμένων ιδιοτήτων και τιμών δεδομένων, πράγμα που δεν αρκεί για την περιγραφή και μοντελοποίηση αντικειμένων συγκεκριμένου γνωστικού πεδίου ή κλάσεων οντοτήτων με κοινά χαρακτηριστικά. Έτσι το RDF μοντέλο καλύπτεται σε αυτό από το RDF Schema το οποίο παρέχει ένα λεξιλόγιο που επιτρέπει τον ορισμό ορολογικής γνώσης και λεξιλογίων συγκεκριμένης θεματολογίας για τον ορισμό και την περιγραφή των κλάσεων κι ιδιοτήτων που συνδέουν RDF πόρους. Πιο συγκεκριμένα το RDF Schema επιτρέπει τον ορισμό κλάσεων στις οποίες οι πόροι ενός RDF μοντέλου μπορούν να ομαδοποιηθούν σε κλάσεις και οι πόροι που ανήκουν σε

αυτήν ονομάζονται *στιγμιότυπα* (instances) αυτής της κλάσης. Επιπλέον, επιτρέπει τον ορισμό ιδιοτήτων ειδικού σκοπού, οι οποίοι αποτελούν ειδικούς πόρους που εμφανίζονται στη θέση του κατηγορήματος σε RDF τριάδες και συνδέουν δύο πόρους μεταξύ τους.

(Ενότητα 2.2.7) Οντολογία και Γλώσσα OWL

Ο όρος **οντολογία** (ontology) είναι μία βασική έννοια του Σημασιολογικού. Ο όρος αυτός έχει φιλοσοφική προέλευση και αναφέρεται στο *λόγο περί του όντος* ή στην *επιστήμη του όντος*, τη φιλοσοφική αναζήτηση που εξετάζει τις αρχές της ύπαρξης και συγκρότησης του *Όντος*, μελετά τη φύση και την ουσία των *Όντων*. Στην επιστήμη των υπολογιστών και την επιστήμη της πληροφορίας, μια οντολογία παρουσιάζει μια αναπαράσταση, επίσημη ονομασία και ορισμό κατηγοριών, ιδιοτήτων και σχέσεων μεταξύ των εννοιών, των δεδομένων και των οντοτήτων που τεκμηριώνουν έναν, πολλούς ή όλους τους τομείς του λόγου. Πιο απλά, μια οντολογία είναι ένας τρόπος εμφάνισης των ιδιοτήτων μιας θεματικής περιοχής και του τρόπου με τον οποίο σχετίζονται, ορίζοντας ένα σύνολο εννοιών και κατηγοριών που αντιπροσωπεύουν το θέμα. Οι οντολογίες μπορούν να διακριθούν μεταξύ τους με βάση το αντικείμενο της εννοιολογικής σύλληψης που περιγράφουν [12] Στα πλαίσια της παρούσας πτυχιακής μας ενδιαφέρουν μόνο οι οντολογίες αναπαράστασης γνώσης, οι οποίες συγκεντρώνουν τους βασικούς μηχανισμούς

που χρησιμοποιούνται σε μια γλώσσα αναπαράστασης γνώσης, όπως το RDF Schema το οποίο δίνει την δυνατότητα, ως ένα βαθμό, της μοντελοποίησης οντολογιών.

Εκεί που το RDF Schema είναι ελλιπής συμπληρώνει η **OWL** (Web Ontology Language) [13], μία γλώσσα που έχει προταθεί από το W3C και αποτελεί την επιλογή για μοντελοποίηση οντολογιών. Τα βασικότερα στοιχεία που χρησιμοποιεί η OWL είναι οι οντότητες(entities), οι εκφράσεις(expressions) και τα αξιώματα(axioms). Τέλος, κάθε οντολογία OWL μπορεί να παρουσιαστεί και σαν ένας γράφος RDF

(Ενότητα 2.2.8) Γράφος Γνώσης (Knowledge Graph)

Ένας **γράφος γνώσης** (knowledge graph) [14] είναι ένας γράφος που αντιπροσωπεύει ένα δίκτυο οντοτήτων—δηλαδή αντικείμενα, γεγονότα, καταστάσεις ή έννοιες—και απεικονίζει τις σχέσεις μεταξύ τους. Οι οντολογίες αντιπροσωπεύουν τη ραχοκοκαλιά της τυπικής σημασιολογίας ενός γράφου γνώσης. Μπορούν να θεωρηθούν ως η βάση δεδομένων του γράφου και διασφαλίζουν μια κοινή κατανόηση των δεδομένων και των σημασιών τους. Μόλις έχουμε μια οντολογία, αν προσθέσουμε συγκεκριμένα δεδομένα, μπορούμε να δημιουργήσουμε ένα γράφο γνώσης. Ουσιαστικά, δημιουργείται ένα γράφος γνώσης χρησιμοποιώντας οντολογίες και δεδομένα αυτών. Συχνά, συνδυάζονται πολλές, ανεξάρτητα ανεπτυγμένες οντολογίες για να ληφθεί ένα γράφημα γνώσης που αντιπροσωπεύει πλήρως έναν τομέα ενδιαφέροντος.

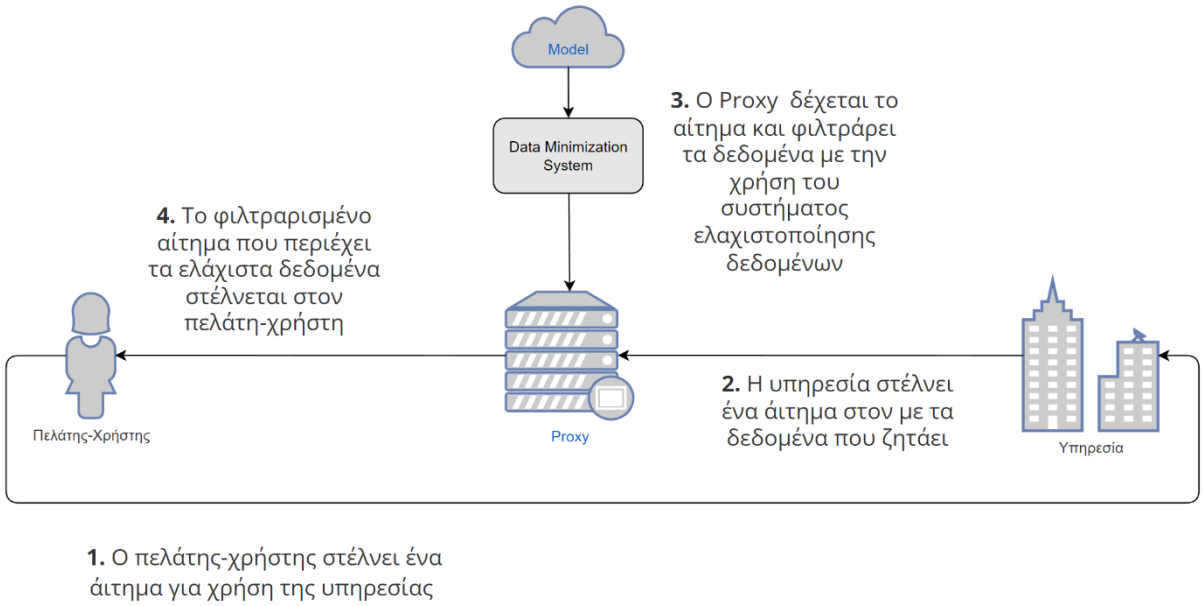
Στην παρούσα πτυχιακή, η επιλογή ανάπτυξης ενός γράφου γνώσης βασίστηκε στην ιδέα της αναπαράστασης δεδομένων σημασιολογικά, επιτρέποντας έτσι την λήψη αποφάσεων με «έξυπνο» τρόπο αφού κάθε αναζήτηση ψάχνει δυναμικά τις συσχετίσεις που εκφράζονται μεταξύ των δεδομένων επιστρέφοντας αποτελέσματα πάντα σχετικά με την αναζήτηση. Ο γράφος αυτός υλοποιήθηκε με ένα μοντέλο RDF (περισσότερες λεπτομέρειες στο Υποκεφάλαιο 3.2: Μοντέλο Δεδομένων Συστήματος του Κεφαλαίου 3).

ΚΕΦΑΛΑΙΟ 3 Περιγραφή συστήματος

(Υποκεφάλαιο 3.1) Γενική Επισκόπηση

Στο πλαίσιο της παρούσας πτυχιακής αναπτύχθηκε σύστημα που επιτελεί το σύνολο των λειτουργιών που αποφασίζει ποια θα είναι τα ελάχιστα προσωπικά δεδομένα που θα χρειαστεί να δώσει ένας πελάτης σε μία υπηρεσία. Στόχος του συστήματος είναι να ελαχιστοποιήσει τα δεδομένα που θα δώσει ένας πελάτης σε μία υπηρεσία. Έστω ότι μία υπηρεσία δίνει στο σύστημα έναν τύπο που αντιστοιχεί σε αυτήν και ένα σύνολο από δεδομένα που ζητάει από τον client. Το σύστημα έχει κατασκευάσει ένα μοντέλο με όλους τους τύπους των προσωπικών δεδομένων ενός πελάτη καθώς και τους τύπους υπηρεσιών για διάφορες περιπτώσεις. Το σύστημα παίρνει τον τύπο της υπηρεσίας και τα δεδομένα και αποφασίζει ,βάση του μοντέλου, αν αυτά είναι τα ελάχιστα δεδομένα που θα μπορούσε να λάβει από τον πελάτη. Όταν βρεθούν τα ελάχιστα δεδομένα τα ζητάει από τον πελάτη.

Το σύστημα έχει πολλούς πιθανούς τρόπος λειτουργίας, ένας από αυτούς απεικονίζεται στο Σχήμα 3 και είναι ότι το σύστημα ελαχιστοποίησης δεδομένων λειτουργεί ως proxy.



Σχήμα 3: Δομή Συστήματος σε λειτουργία proxy

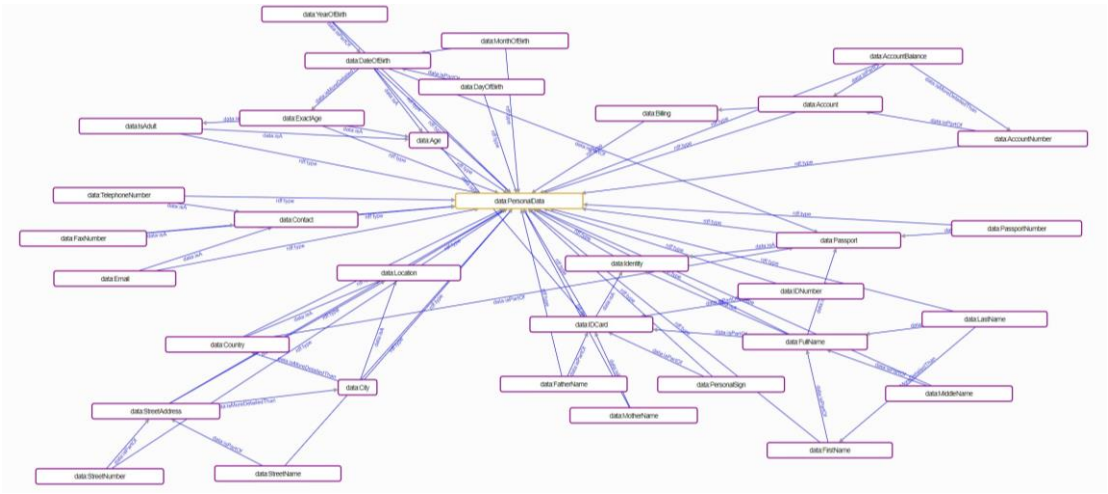
(Υποκεφάλαιο 3.2) Μοντέλο Δεδομένων Συστήματος

Στα πλαίσια της παρούσας πτυχιακής κατασκευάστηκε ένα μοντέλο RDF, που αναπαριστά έναν γράφο γνώσης ο οποίος παρουσιάζει τις συσχετίσεις μεταξύ των τύπων προσωπικών δεδομένων και των τύπων υπηρεσιών με αυτούς.

Πιο συγκεκριμένα δημιουργήθηκαν δύο γράφοι οι οποίοι συσχετίζονται μεταξύ τους. Ένα γράφος PersonalDataGraph και ένας γράφος Services. Μέσα στον γράφο έχουν δοθεί κάποια namespace αναγνωριστικά τα οποία ξεχωρίζουν πότε τα στοιχεία του γράφου αποτελούν τύπους δεδομένων ή υπηρεσιών. Για τον λόγο αυτό έχουμε data: για τους τύπους προσωπικών δεδομένων και service: για τους τύπους των υπηρεσιών.

(Ενότητα 3.2.1) PersonalDataGraph

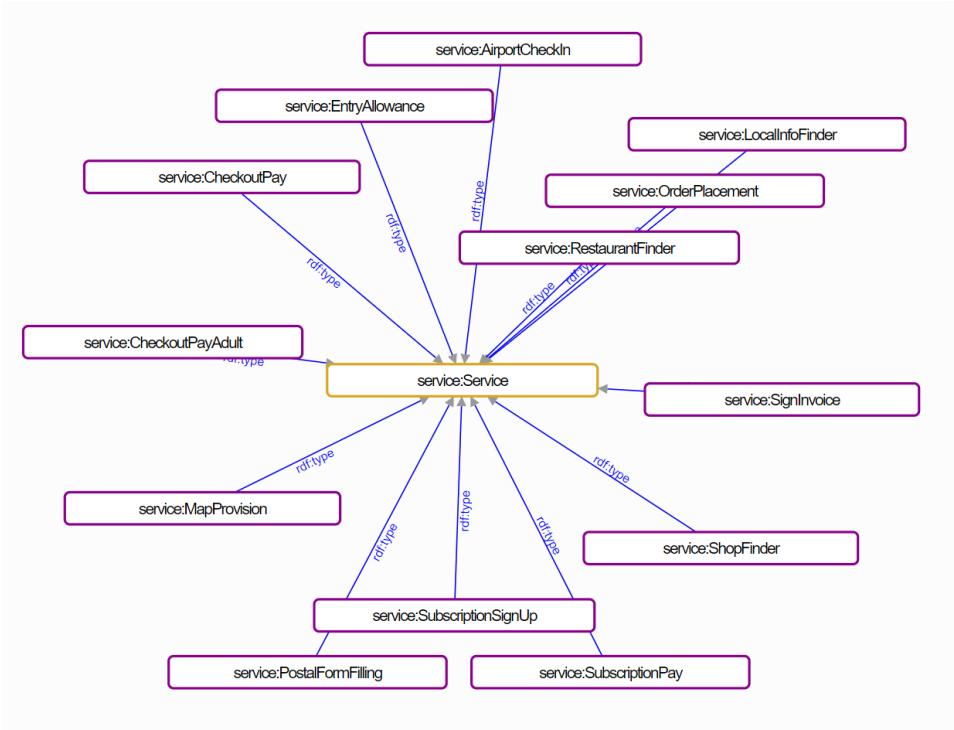
Οι συσχετίσεις που εκφράζονται μέσα στο PersonalDataGraph (Σχήμα 4) ασχολούνται με το επίπεδο λεπτομέρειας των τύπων προσωπικών δεδομένων. Όλοι οι τύποι προσωπικών δεδομένων αποτελούν τύπους της κλάσης PersonalData και συσχετίζονται ως rdf.type του RDF Schema. Οι τύποι προσωπικών δεδομένων Location, Age, Contact, Identity και Billing είναι του ευρύτερου τύπου της PersonalData και οι συχετίσεις αυτών με άλλα δεδομένα ορίζονται ως isA πχ. (isA(City, Location)). Άλλες συσχετίσεις είναι του τύπου isPartOf και isMoreDetailedThan. Η λογική που ακολουθεί την isPartOf είναι ότι το υποκείμενο αποτελεί ένα μέρος του αντικείμενου μίας δήλωσης, ενώ η σχέση isMoreDetailedThan περιγράφει ένα δεδομένο που είναι πιο λεπτομερές από ένα άλλο.Για παράδειγμα, το δεδομένο του τύπου «YearOfBirth» είναι ένα μέρος του «DateOfBirth» (isPartOf(YearOfBirth, DateOfBirth)), ενώ το δεδομένο του τύπου «StreetAddress» είναι περισσότερο λεπτομερής από το «City» isMoreDetailedThan(StreetAddress, City).



Σχήμα 4: Personal Data Graph

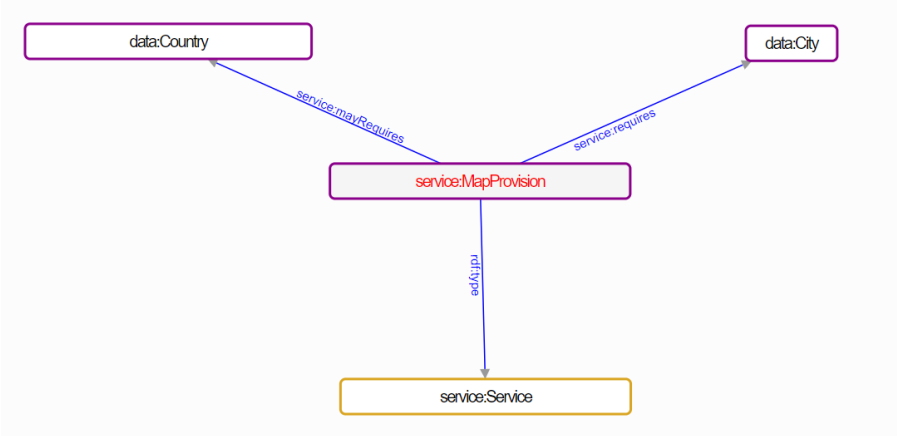
(Ενότητα 3.2.2) ServicesGraph

Στο γράφο ServicesGraph τύποι υπηρεσιών αποτελούν τύπους της κλάσης Services και συσχετίζονται ως rdf.type του RDF Schema.



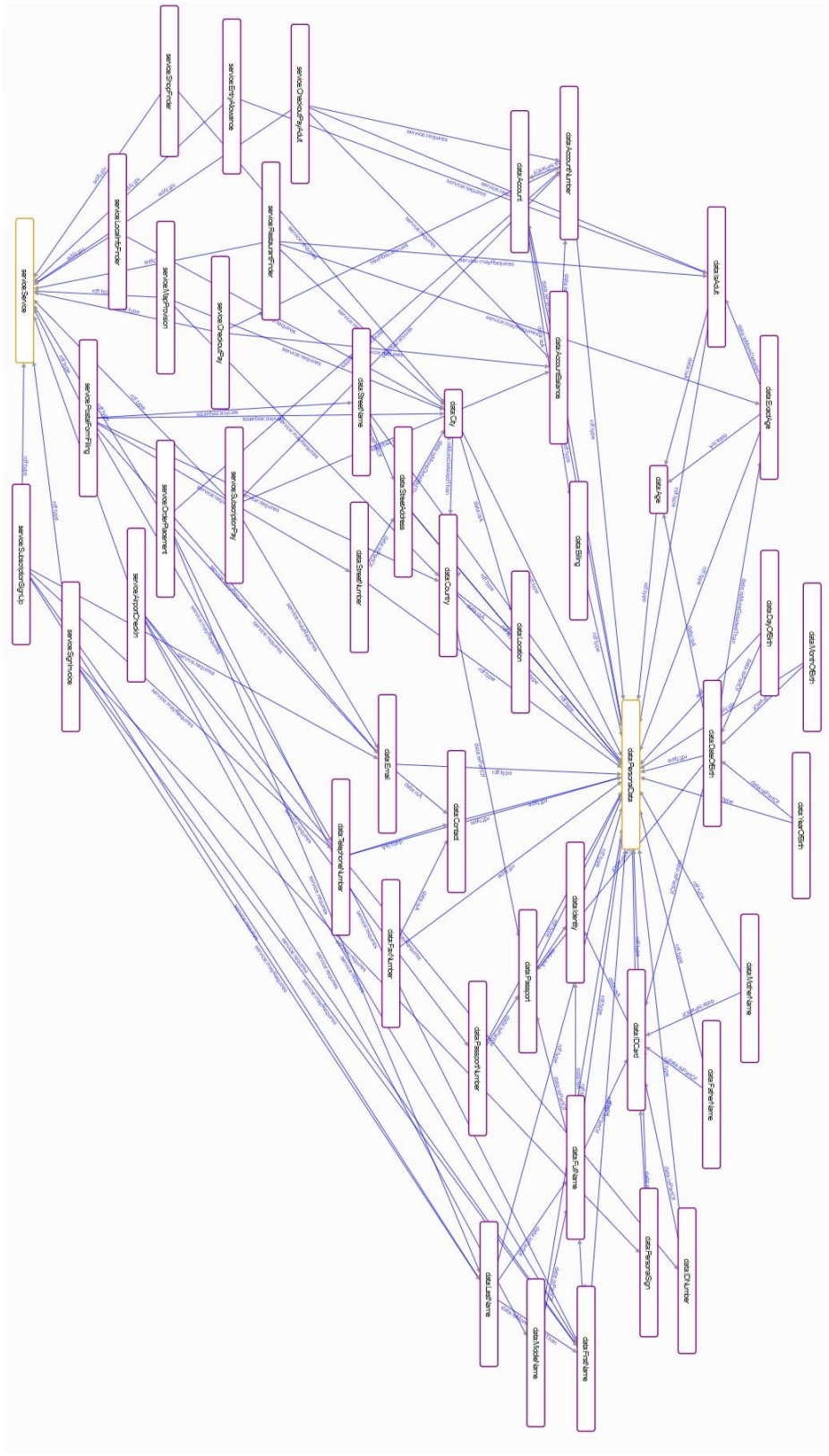
Σχήμα 5: Services Graph

Οι δύο γράφοι συνδέονται με συσχετίσεις που δείχνουν ποια είναι τα ελάχιστα δεδομένα που χρειάζεται ή ενδεχομένως να χρειάζεται μία υπηρεσία. Οι συσχετίσεις αυτές είναι προσδιορίζονται ως requires και mayRequired. Για παράδειγμα, για μία υπηρεσία του τύπου «MapProvision» τα ελάχιστα δεδομένα που χρειάζεται είναι «City» και ενδεχομένως να χρειάζεται «Country» (requires(MapProvision, City) και mayRequired(MapProvision,Country)).



Σχήμα 6: Παράδειγμα συσχέτισης Service με Personal Data

Το μοντέλο RDF που αποτελεί το γράφο γνώσης παρουσιάζεται στο Σχήμα 7.



Σχήμα 7: Το μοντέλο RDF που χρησιμοποιείται στο σύστημα

(Υποκεφάλαιο 3.3) Ανάλυση Συστήματος

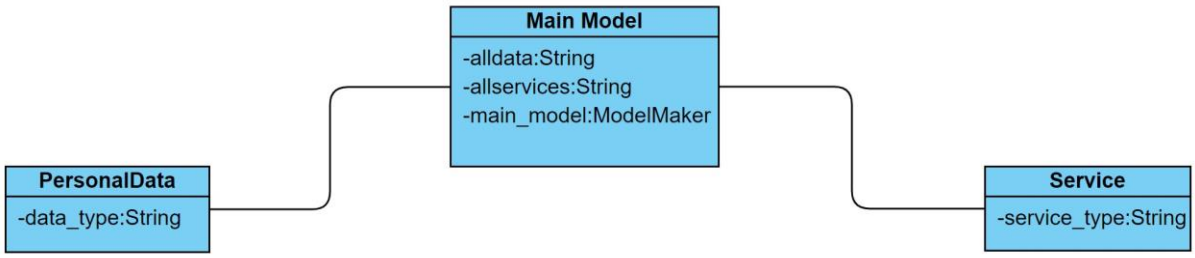
(Ενότητα 3.3.1) Δομή Συστήματος

Το σύστημα ελαχιστοποίησης δεδομένων αποτελείται από δύο κύριες λειτουργίες. Η πρώτη λειτουργία κατασκευάζει ένα γράφο γνώσης ο οποίος αποτελεί το μοντέλο που εκφράζει τις συσχετίσεις των τύπων προσωπικών δεδομένων με διάφορους τύπους υπηρεσιών. Η δεύτερη λειτουργία είναι ο αλγόριθμος που πραγματοποιεί την ελαχιστοποίηση δεδομένων(ο αλγόριθμος εξηγείται αναλυτικά στο Κεφάλαιο 4).

Το σύστημα υλοποιήθηκε σε Java και αποτελείται απο πέντε κλάσεις, τις *MainModel*, *ModelMaker, DecisionMaker, PersonalData* και *Service* με κύρια κλάση την *MainModel*.

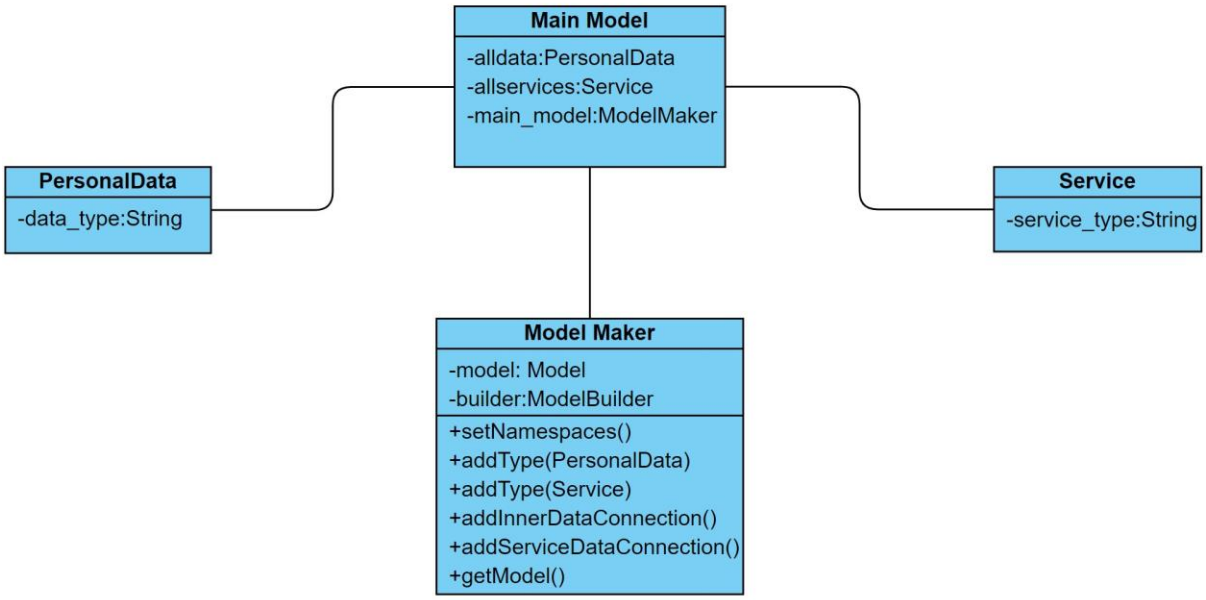
Στην κλάση *MainModel* δημιουργείται το μοντέλο με τους τύπους δεδομένων και υπηρεσιών.

Η PersonalData είναι η κλάση λογισμικού που αναπαριστά-αντιπροσωπεύει τους τύπους των δεδομένων και χρησιμοποιείται για την διαχείριση αυτών, αντίστοιχα η κλάση Service είναι μόνο για την δημιουργία τύπων υπηρεσιών.



Σχήμα 8: Διάγραμμα Κλάσης 1 από 3

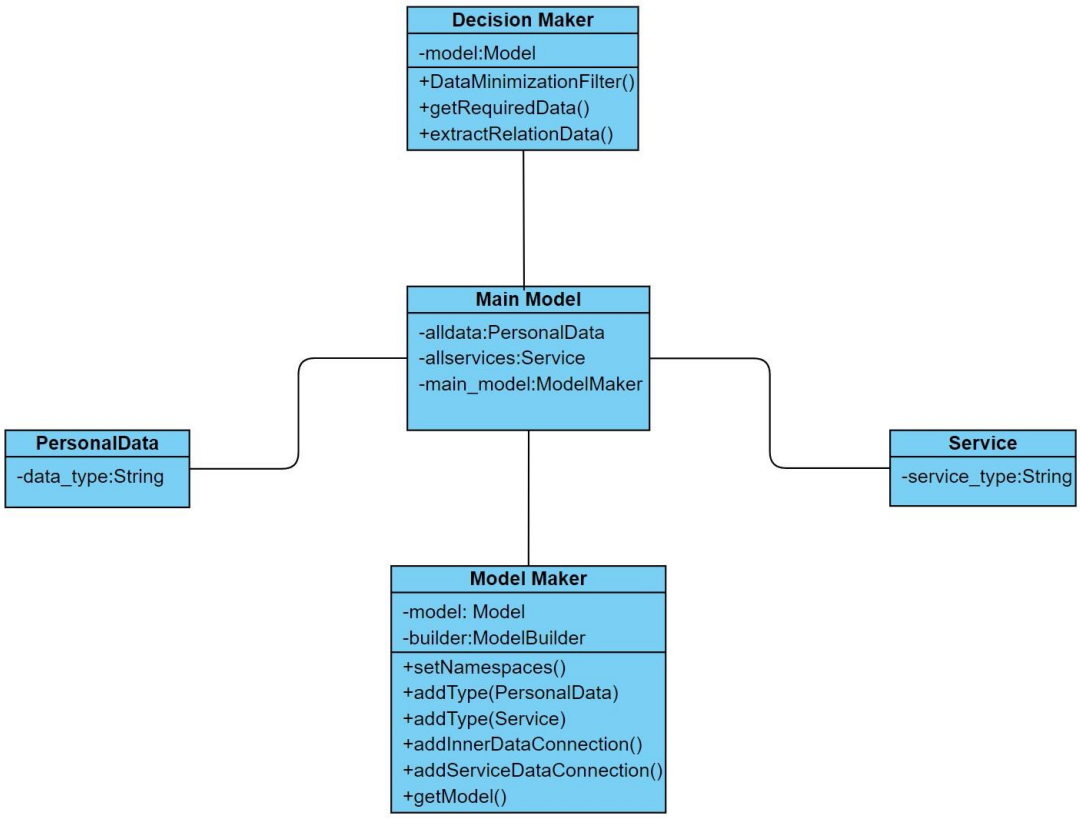
Στην κλάση *MainModel* δημιουργείται το μοντέλο της κλάσης *ModelMaker*. Στην *ModelMaker* πραγματοποιούνται όλες οι συσχετίσεις που παρουσιάζονται στο μοντέλο.



Σχήμα 9: Διάγραμμα Κλάσης 2 από 3

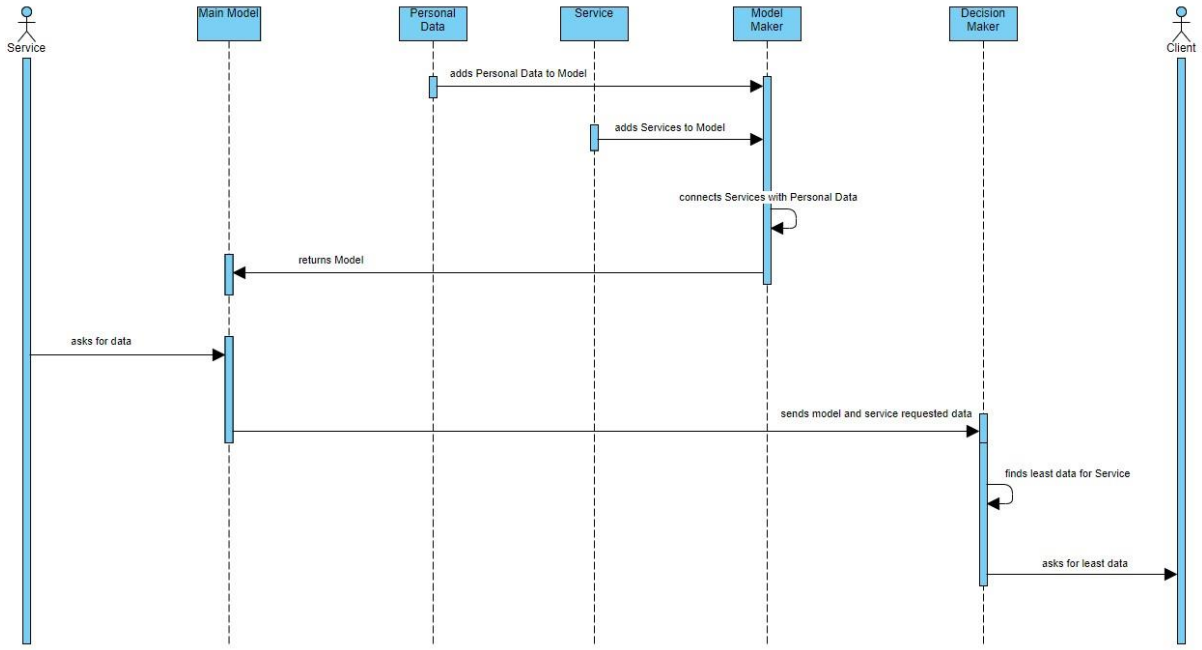
Η δεύτερη λειτουργία του συστήματος είναι η εφαρμογή του αλγόριθμου ελαχιστοποίησης δεδομένων που εξηγείται παρακάτω.

Η πάρση των αποφάσεων γίνεται στην κλάση *DecisionMaker*. Μέσα στην κλάση *DecisionMaker* έχει αναπτυχθεί ο αλγόριθμος που αναζητεί και αποφασίζει για τα ελάχιστα δεδομένα που θα επιστρέψει το σύστημα. Η αναζήτηση γίνεται βάση του μοντέλου που αναπτύχθηκε στο πρώτο στάδιο λειτουργίας του συστήματος.



Σχήμα 10: Διάγραμμα Κλάσης 3 από 3

(Ενότητα 3.3.2) Λειτουργία Συστήματος



Σχήμα 11: Διάγραμμα ροής της λειτουργίας του συστήματος

Στην κλάση Model Maker κατασκευάζονται τύποι υπηρεσιών και προσωπικών δεδομένων της κλάσης Personal Data και Services, προθέτονται στο μοντέλο όπου δημιουργούνται συσχετίσεις μεταξύ τους. Το μοντέλο αυτό επιστρέφεται στην Main Model η οποία δέχεται κάποια δεδομένα που ζητάει έναν τύπος υπηρεσίας. Τα δεδομένα που ζήτησε η υπηρεσία στέλνονται στην κλάση Decision Maker στην οποία εφαρμόζεται ελαχιστοποίηση αυτών και παράγει τα ελάχιστα δεδομένα που χρειάζεται να δώσει στην υπηρεσία και ζητώνται από τον πελάτη.

ΚΕΦΑΛΑΙΟ 4 Αλγόριθμος Ελαχιστοποίησης Δεδομένων (Data Minimization Algorithm)

(Υποκεφάλαιο 4.1) Σύντομη Περιγραφή

Ο αλγόριθμος ελαχιστοποίησης δεδομένων αναλαμβάνει τον ρόλο του υπεύθυνου επεξεργασίας δεδομένων προκειμένου να περιορίσει την συλλογή προσωπικών πληροφοριών. Βασική του λειτουργία είναι να επεξεργαστεί τα δεδομένα που του δίνονται και να επιστρέψει τα ελάχιστα δεδομένα που χρειάζονται για την εξυπηρέτηση μίας υπηρεσίας. Η εύρεση των ελάχιστων δεδομένων γίνεται με αναζήτηση κατά πλάτος σε ένα γράφο γνώσης που περιγράφει τις σχέσεις ενός τύπου μίας υπηρεσίας και των ελάχιστων δεδομένων που χρειάζεται.

(Υποκεφάλαιο 4.2) Αλγόριθμος

(Ενότητα 4.2.1) Ψευδοκώδικας

Algorithm: Data Minimization Filter

Input: service_type and service_requested_data

Output: least_data needed for type of service

1) **initialize** required_data[ ];
2) **for** the **model** statements which statement.relation == “requires” **and**
   statement.subject == service_type **do**:
3) required_data[ ].add(statement data);
4) **endfor**

5) **initialize** total_found_data[ ] ;
6) **initialize** least_data[ ] ;

7) **for** each data in service_requested_data **do**:
8) **for** the submodel of data statements in which statement.relation == “isPartOf” **or**
   statement.relation == “isMoreDetailedThan”
   :
9) total_found_data[ ].add(statement data);
10) **endfor**

11) **if** total_found_data contains required_data **do**:

```
12)          least_data[ ].add(data);
13)     endif

14) endfor

15) return least_data[ ] ;
```

(Ενότητα 4.2.2) Εξήγηση Ψευδοκώδικα

Ο αλγόριθμος δέχεται έναν τύπο μίας υπηρεσίας και μία λίστα με τα δεδομένα που ζητήθηκαν για επεξεργασία και επιστρέφει τα ελάχιστα δεδομένα που χρειάζεται η υπηρεσία. Σύμφωνα με το μοντέλο model ψάχνει να βρει ποιά ελάχιστα δεδομένα χρειάζεται η υπηρεσία και τα αποθηκεύει σε έναν πίνακα required_data. Αυτά τα ελάχιστα δεδομένα εξάγονται αναλύοντας τις σχέσεις που συνδέουν τα δεδομένα με τις υπηρεσίες και είναι του τύπου “requires” (γραμμή 1 – 4).

Ο πίνακας total_found_data αποθηκεύει το σύνολο των δεδομένων που σχετίζονται άμεσα με τα αρχικά δεδομένα που δόθηκαν στον αλγόριθμο και ο πίνακας least_data αποθηκεύει τα ελάχιστα δεδομένα που χρειάζονται (γραμμή 5 – 6).

Για κάθε δεδομένο service_requested_data (γραμμή 7 - 14) χρησιμοποιείται ένα υπομοντέλο που έχει πατέρα αυτό το δεδομένο (γραμμή 8). Μέσα στο υπομοντέλο γίνεται αναζήτηση κατά πλάτος για να βρεθούν τα αμέσως επόμενα σχετιζόμενα δεδομένα που συνδέονται με τις σχέσεις “isPartOf” ή “isMoreDetailedThan” (γραμμή 8 -10). Η αναζήτηση γίνεται επαναληπτικά για όλες τις σχέσεις του υπομοντέλου. Όλα τα δεδομένα που κρατούνται μέσα από την αναζήτηση αποθηκεύονται στο total_found_data (γραμμή 9). Έπειτα ελέγχεται ποια δεδομένα από αυτά που κρατήθηκαν κατά την αναζήτηση είναι τα ελάχιστα δεδομένα που χρειάζεται ο τύπος της υπηρεσίας και αποθηκεύονται στον πίνακα least_data (γραμμή 11 – 13).

Όταν γίνει η αναζήτηση για όλα τα δεδομένα service_requested_data επιστρέφεται ο πίνακας least_data που περιέχει τα ελάχιστα δεδομένα (γραμμή 15).

ΚΕΦΑΛΑΙΟ 5 Προγραμματιστική Υλοποίηση Συστήματος

Η υλοποίηση του συστήματος έγινε σε γλώσσα προγραμματισμού Java. Τα εργαλεία που χρησιμοποιήθηκαν μέσα στο σύστημα είναι το Eclipse RDF4J™, ένα Java API.

(Υποκεφάλαιο 5.1) RDF4J

Το [1] Eclipse RDF4J™ είναι ένα ανοιχτού κώδικα framework της Java για διαχείριση δεδομένων RDF. Το framework περιλαμβάνει ανάλυση, αποθήκευση, εξαγωγή συμπερασμάτων και αναζήτηση τέτοιων δεδομένων. Μέσω ενός API το framework προσφέρει μια μεγάλη κλίμακα εργαλείων στους προγραμματιστές για να αξιοποιήσουν τη δύναμη του RDF και των σχετικών προτύπων. Ο πυρήνας του πλαισίου RDF4J είναι το RDF Model API ,που ορίζεται στο πακέτο org.eclipse.rdf4j.model. Αυτό το API ορίζει πώς αναπαρίστανται τα δομικά στοιχεία του RDF (δηλώσεις, IRI και μοντέλα). Για την προγραμματιστική προσέγγιση του συστήματος χρησιμοποιήθηκαν συναρτήσεις του RDF4J για την κατασκευή και την διαχείριση του μοντέλου και θα εξηγηθούν με περισσότερη λεπτομέρεια στο επόμενο κεφάλαιο.

(Υποκεφάλαιο 5.2) Μοντέλο RDF

(Ενότητα 5.2.1) Δημιουργία Τύπων

Η κεντρική κλάση του συστήματος είναι η `MainModel`. Μέσα στην `MainModel` κατασκευάζετε ένα instance της κλάσης `ModelMaker main_model` που θα είναι ο κατασκευαστής του μοντέλου RDF. Το Model είναι μια επέκταση της προεπιλεγμένης κλάσης Java Collection java.util.Set<Statement>. Στο RDF4J, ένα μοντέλο είναι απλώς μια συλλογή δηλώσεων RDF (RDF statements).

Συμπεριλαμβάνεται στην διεπαφή `org.eclipse.rdf4j.model.Model`

```
public class MainModel {

    public static void main(String[] args) {

        ModelMaker main_model = new ModelMaker();
```

Η `ModelMaker` φτιάχνει έναν builder και ένα model. Ο `builder` είναι της κλάσης `ModelMaker` που αποτελεί εργαλείο της RDF4J και λειτουργεί ως χτίστης όπως εξάλλου λέει και το όνομα του, και «χτίζει» το μοντέλο κάθε φορά που προστίθεται κάτι. Χτίζει πάνω στο model.

[1]Για περισσότερες πληροφορίες σχετικά με το API μπορείτε να επισκεφτείτε την ιστοσελίδα: https://rdf4j.org/

Συμπεριλαμβάνεται στην διεπαφή `org.eclipse.rdf4j.model.util.ModelBuilder`

```
private static ModelBuilder builder = new ModelBuilder();
private static Model model;
```

Στο `main_model` καλείται μία συνάρτηση `setNamespaces()` μέσα στην οποία φτιάχνονται κάποια namespaces που θα χρησιμοποιούν μέσα στους γράφους.

Η μέθοδος `setNamespaces()` του `ModelMaker` που χρησιμοποεί την συνάρτηση της RDF4J για να θέσουμε κάποια αναγνωριστικά στο μοντέλο:

```
public void setNamespaces() {

        model = builder.setNamespace("d", "data:").setNamespace("s",
"service:").build();

}
```

Κάθε προσωπικό δεδομένο είναι και ένα instance της κλάσης `PersonalData` η οποία κρατάει ως μοναδική πληροφορία τον τύπο του δεδομένου.

```
public class PersonalData{



    public String data_type;



    public PersonalData(String type) {

        this.data_type = type;

    }



}
```

Όλα τα προσωπικά δεδομένα αποθηκεύονται μέσα σε μία λίστα που ονομάζεται `alldata`.

```
List<PersonalData> alldata = new ArrayList<PersonalData>();



            alldata.add(new PersonalData("Identity"));

                alldata.add(new PersonalData("FullName"));

                    alldata.add(new PersonalData("FirstName"));

                    alldata.add(new PersonalData("MiddleName"));

                    alldata.add(new PersonalData("LastName"));
```

```
    alldata.add(new PersonalData("IDCard"));
        alldata.add(new PersonalData("IDNumber"));
        alldata.add(new PersonalData("PersonalSign"));
        alldata.add(new PersonalData("FatherName"));
        alldata.add(new PersonalData("MotherName"));

    alldata.add(new PersonalData("Passport"));
        alldata.add(new PersonalData("PassportNumber"));


alldata.add(new PersonalData("Contact"));
    alldata.add(new PersonalData("TelephoneNumber"));
    alldata.add(new PersonalData("Email"));
    alldata.add(new PersonalData("FaxNumber"));\


alldata.add(new PersonalData("Location"));
    alldata.add(new PersonalData("StreetAddress"));
        alldata.add(new PersonalData("StreetName"));
        alldata.add(new PersonalData("StreetNumber"));
    alldata.add(new PersonalData("City"));
    alldata.add(new PersonalData("Country"));


alldata.add(new PersonalData("Age"));
    alldata.add(new PersonalData("IsAdult"));
    alldata.add(new PersonalData("ExactAge"));
    alldata.add(new PersonalData("DateOfBirth"));
```

```
                alldata.add(new PersonalData("DayOfBirth"));

                alldata.add(new PersonalData("MonthOfBirth"));

                alldata.add(new PersonalData("YearOfBirth"));


            alldata.add(new PersonalData("Billing"));

                alldata.add(new PersonalData("Account"));

                alldata.add(new PersonalData("AccountNumber"));

                alldata.add(new PersonalData("AccountBalance"));
```

Τα δεδομένα προθέτονται όλα στο γράφο `PersonalDataGraph` που αποτελεί μέρος του μοντέλου καλώντας την `addType()` του `ModelMaker` που παίρνει σαν παράμετρο κάθε ένα από τα δεδομένα.

```
for (PersonalData pd : alldata) {

     main_model.addType(pd);

}
```

Η μέθοδος `addType()` η οποία δέχεται μόνο `PersonalData`:

```
public void addType(PersonalData data) {

       model = builder.namedGraph("d:PersonalDataGraph")

                      .add("d:"+data.data_type, RDF.TYPE, "d:PersonalData")

                      .build();

}
```

Η μέθοδος `nameGraph()` είναι συνάρτηση του RDF4J και προσθέτει επώνυμους γράφους RDF μέσα στο μοντέλο.

Η μέθοδος `add()` είναι συνάρτηση του RDF4J που προσθέτει ένα statement μέσα στο μοντέλο. Εφόσον κάθε statement είναι μία τριάδα, τα ορίσματα που παίρνει είναι στο πρώτο πεδίο το subject, στο δεύτερο predicate και στο τρίτο το object. Στην περίπτωση της `addType()` στην στην `add()` το predicate θα είναι του τύπου rdf.type. Για την χρήση λεξιλογίου RDF όπως γίνεται εδώ, η RDF4J παρέχει την διεπαφή `org.eclipse.rdf4j.model.vocabulary.RDF`.

Κάθε τύπος υπηρεσίας είναι και ένα instance της κλάσης `Service`, η οποία κρατάει ως μοναδική πληροφορία τον τύπο της.

```
public class Service {


    public String service_type;


    public Service(String type) {

        this.service_type = type;

    }


}
```

Όλοι οι τύποι των υπηρεσιών αποθηκεύονται μέσα σε μία λίστα που ονομάζεται `alldata`.

```
List<Service> allservices = new ArrayList<Service>();

            allservices.add(new Service("SignInvoice"));

            allservices.add(new Service("LocalInfoFinder"));

            allservices.add(new Service("MapProvision"));

            allservices.add(new Service("ShopFinder"));

            allservices.add(new Service("RestaurantFinder"));

            allservices.add(new Service("OrderPlacement"));

            allservices.add(new Service("EntryAllowance"));

            allservices.add(new Service("CheckoutPay"));

            allservices.add(new Service("CheckoutPayAdult"));

            allservices.add(new Service("AirportCheckIn"));

            allservices.add(new Service("SubscriptionPay"));

            allservices.add(new Service("SubscriptionSignUp"));

            allservices.add(new Service("PostalFormFilling"));
```

Τα δεδομένα προθέτονται όλα στο γράφο `ServicesGraph` που αποτελεί μέρος του μοντέλου καλώντας την `addType()` του `ModelMaker` που παίρνει σαν παράμετρο κάθε τύπους υπηρεσιών.

```
for (Service srvc : allservices) {

     main_model.addType(srvc);

}
```

Η μέθοδος `addType()` η οποία δέχεται μόνο Service.

```
public void addType(Service service) {

       model = builder.namedGraph("s:ServicesGraph")

                      .add("s:"+service.service_type, RDF.TYPE, "s:Service")

                      .build();

}
```

Όπως περιγράφηκε για την `addType()` των προσωπικών δεδομένων έτσι κι εδώ.

(Ενότητα 5.2.2) Σύνδεση Τύπων

Στην συνέχεια πραγματοποιούνται οι συνδέσεις αρχικά μεταξύ των δεδομένων μέσα στον γράφο `PersonalDataGraph` και έπειτα μεταξύ του γράφου `PersonalDataGraph` και του `ServicesGraph`.

Όλες οι συνδέσεις των δεδομένων γίνονται με την `addInnerDataConnection()` που παίρνει σαν παράμετρο τα προσωπικά δεδομένα που θα συσχετιστούν και τον τύπο της συσχέτισης. Οι συσχετίσεις μεταξύ των δεδομένων είναι *IsA*, *isPartOf* και *isMoreDetailedThan*. (Ο λόγος επιλογής των συσχετίσεων περιγράφεται στην λογική της υλοποίησης του συστήματος).

Η μέθοδος `addInnerDataConnection()`

```
    public void addInnerDataConnection(PersonalData data1, PersonalData data2, String
connection) {

        model = builder.namedGraph("d:PersonalDataGraph")

                        .add("d:"+data2.data_type, "d:"+connection, "d:"+data1.data_type)

                        .build();

    }
```

Μέσα στην κλάση `MainModel`

```
    //Identity -> FullName, IdCard, Passport

    main_model.addInnerDataConnection(alldata.get(0), alldata.get(1), "isA");

    main_model.addInnerDataConnection(alldata.get(0), alldata.get(5), "isA");

    main_model.addInnerDataConnection(alldata.get(0), alldata.get(10), "isA");



    //Passport -> PassportNumber

    main_model.addInnerDataConnection(alldata.get(10), alldata.get(11), "isPartOf");



    //FullName -> First,Middle,Last Name

    main_model.addInnerDataConnection(alldata.get(1), alldata.get(2), "isPartOf");

    main_model.addInnerDataConnection(alldata.get(1), alldata.get(3), "isPartOf");

    main_model.addInnerDataConnection(alldata.get(1), alldata.get(4), "isPartOf");



    //First -> LastName
```

```
    main_model.addInnerDataConnection(alldata.get(2), alldata.get(4),
"isMoreDetailedThan");


    //IDCard -> IDNumber, Father,MotherName, PersonalSign

    main_model.addInnerDataConnection(alldata.get(5), alldata.get(6), "isPartOf");

    main_model.addInnerDataConnection(alldata.get(5), alldata.get(7), "isPartOf");

    main_model.addInnerDataConnection(alldata.get(5), alldata.get(8), "isPartOf");

    main_model.addInnerDataConnection(alldata.get(5), alldata.get(9), "isPartOf");


    //IDCard, Passport <- FullName

    main_model.addInnerDataConnection(alldata.get(5), alldata.get(1), "isPartOf");

    main_model.addInnerDataConnection(alldata.get(10), alldata.get(1), "isPartOf");


    //Contact -> Telephone, Email, Fax

    main_model.addInnerDataConnection(alldata.get(12), alldata.get(13), "isA");

    main_model.addInnerDataConnection(alldata.get(12), alldata.get(14), "isA");

    main_model.addInnerDataConnection(alldata.get(12), alldata.get(15), "isA");


    //Location -> Country, City, StreetAddress

    main_model.addInnerDataConnection(alldata.get(16), alldata.get(17), "isA");

    main_model.addInnerDataConnection(alldata.get(16), alldata.get(20), "isA");

    main_model.addInnerDataConnection(alldata.get(16), alldata.get(21), "isA");


    //StreetAddress -> StreetNumber,Name

    main_model.addInnerDataConnection(alldata.get(17), alldata.get(18), "isPartOf");

    main_model.addInnerDataConnection(alldata.get(17), alldata.get(19), "isPartOf");


    //City <- Country

    main_model.addInnerDataConnection(alldata.get(21), alldata.get(20),
"isMoreDetailedThan");
```

```
    //StreetAddress <- City

    main_model.addInnerDataConnection(alldata.get(20), alldata.get(17),
"isMoreDetailedThan");



    //Age -> IsAdult, ExactAge, DateOfBirth

    main_model.addInnerDataConnection(alldata.get(22), alldata.get(23), "isA");

    main_model.addInnerDataConnection(alldata.get(22), alldata.get(24), "isA");

    main_model.addInnerDataConnection(alldata.get(22), alldata.get(25), "isA");



    //DateOfBirth -> DayOfBirth, MonthOfBirth, YearOfBirth

    main_model.addInnerDataConnection(alldata.get(25), alldata.get(26), "isPartOf");

    main_model.addInnerDataConnection(alldata.get(25), alldata.get(27), "isPartOf");

    main_model.addInnerDataConnection(alldata.get(25), alldata.get(28), "isPartOf");



    //IDCard, Passport <- DateOfBirth

    main_model.addInnerDataConnection(alldata.get(5), alldata.get(25), "isPartOf");

    main_model.addInnerDataConnection(alldata.get(10), alldata.get(25), "isPartOf");



    //Passport <- Country

    main_model.addInnerDataConnection(alldata.get(10), alldata.get(21), "isPartOf");



    //ExactAge <- IsAdult

    main_model.addInnerDataConnection(alldata.get(23), alldata.get(24),
"isMoreDetailedThan");



    //DateOfBirth <- ExactAge

    main_model.addInnerDataConnection(alldata.get(24), alldata.get(25),
"isMoreDetailedThan");



    //Billing -> Account
```

```
    main_model.addInnerDataConnection(alldata.get(29), alldata.get(30), "isA");



    //Account -> AccountNumber, AccountBalance

    main_model.addInnerDataConnection(alldata.get(30), alldata.get(31), "isPartOf");

    main_model.addInnerDataConnection(alldata.get(30), alldata.get(32), "isPartOf");



    //AccountBalance <- AccountNumber

    main_model.addInnerDataConnection(alldata.get(31), alldata.get(32),
"isMoreDetailedThan");
```

Οι συνδέσεις των υπηρεσιών γίνονται με την `addServiceDataConnection()` που παίρνει σαν παράμετρο τα προσωπικά δεδομένα και τις υπηρεσίες που θα συσχετιστούν και τον τύπο της συσχέτισης.

Οι συσχετίσεις μεταξύ των τύπων υπηρεσιών και των δεδομένων είναι *required*, *mayRequires*. (Ο λόγος επιλογής των συσχετίσεων περιγράφεται στην λογική της υλοποίησης του συστήματος).

Η μέθοδος `addInnerDataConnection()`:

```
    public void addServiceDataConnection(Service service, PersonalData data, String
connection) {

            model = builder.add("s:"+service.service_type, "s:"+connection,
"d:"+data.data_type)

                        .build();

    }
```

Μέσα στην κλάση `MainModel`

```
    //SignInvoice -> PersonalSign, First,Last Name

    main_model.addServiceDataConnection(allservices.get(0), alldata.get(7), "requires");

    main_model.addServiceDataConnection(allservices.get(0), alldata.get(2), "mayRequires");

    main_model.addServiceDataConnection(allservices.get(0), alldata.get(4), "mayRequires");

    //LocalInfoFinder -> City

    main_model.addServiceDataConnection(allservices.get(1), alldata.get(20), "requires");

    //MapProvision -> City, Country

    main_model.addServiceDataConnection(allservices.get(2), alldata.get(20), "requires");

    main_model.addServiceDataConnection(allservices.get(2), alldata.get(21),
"mayRequires");

    //ShopFinder -> City

    main_model.addServiceDataConnection(allservices.get(3), alldata.get(20), "requires");

    //RestaurantFinder -> City, IsAdult, ExactAge

    main_model.addServiceDataConnection(allservices.get(4), alldata.get(20), "requires");

    main_model.addServiceDataConnection(allservices.get(4), alldata.get(23),
"mayRequires");

    main_model.addServiceDataConnection(allservices.get(4), alldata.get(24),
"mayRequires");

    //OrderPlacement -> First,Last Name, Email, AccountInfo

    main_model.addServiceDataConnection(allservices.get(5), alldata.get(2), "requires");

    main_model.addServiceDataConnection(allservices.get(5), alldata.get(4), "requires");

    main_model.addServiceDataConnection(allservices.get(5), alldata.get(14), "requires");

    main_model.addServiceDataConnection(allservices.get(5), alldata.get(31),
"mayRequires");

    //EntryAllowance -> IsAdult

    main_model.addServiceDataConnection(allservices.get(6), alldata.get(23), "requires");
```

```
    //CheckoutPay -> AccountNumber, AccountBalance

    main_model.addServiceDataConnection(allservices.get(7), alldata.get(31), "requires");

    main_model.addServiceDataConnection(allservices.get(7), alldata.get(32), "requires");

    //CheckoutPayAdult -> IsAdult, AccountNumber, AccountBalance

    main_model.addServiceDataConnection(allservices.get(8), alldata.get(23), "requires");

    main_model.addServiceDataConnection(allservices.get(8), alldata.get(31), "requires");

    main _model.addServiceDataConnection(allservices.get(8), alldata.get(32), "requires");

    //AirportCheckIn -> PassportNumber, FirstName, LastName, IDNumber

    main_model.addServiceDataConnection(allservices.get(9), alldata.get(11), "requires");

    main_model.addServiceDataConnection(allservices.get(9), alldata.get(2), "requires");

    main_model.addServiceDataConnection(allservices.get(9), alldata.get(3), "requires");

    main_model.addServiceDataConnection(allservices.get(9), alldata.get(6),
"mayRequires");

    //SubscriptionPay -> AccountNumber, AccountBalance, Email

    main_model.addServiceDataConnection(allservices.get(10), alldata.get(31), "requires");

    main_model.addServiceDataConnection(allservices.get(10), alldata.get(32), "requires");

    main_model.addServiceDataConnection(allservices.get(10), alldata.get(14),
"mayRequires");

    //SubscriptionSignUp -> FirstName, LastName, Email

    main_model.addServiceDataConnection(allservices.get(11), alldata.get(2), "requires");

    main_model.addServiceDataConnection(allservices.get(11), alldata.get(4), "requires");

    main_model.addServiceDataConnection(allservices.get(11), alldata.get(14), "requires");

    main_model.addServiceDataConnection(allservices.get(11), alldata.get(13),
"mayRequires");

    //PostalFormFilling -> Country, City, StreetName, StreetNumber, Email, Telephone

    main_model.addServiceDataConnection(allservices.get(12), alldata.get(21), "requires");
```

```
    main_model.addServiceDataConnection(allservices.get(12), alldata.get(20), "requires");

    main_model.addServiceDataConnection(allservices.get(12), alldata.get(18), "requires");

    main_model.addServiceDataConnection(allservices.get(12), alldata.get(19), "requires");

    main_model.addServiceDataConnection(allservices.get(12), alldata.get(14),
"mayRequires");

    main_model.addServiceDataConnection(allservices.get(12), alldata.get(13),
"mayRequires");
```

Με την σύνδεση των δεδομένων με τα τους τύπους των υπηρεσιών ολοκληρώνεται η κατασκευή του μοντέλου RDF.

(Υποκεφάλαιο 5.3) Εφαρμογή Ελαχιστοποίησης Δεδομένων

Στην κλάση `MainModel` δημιουργείται ένα instance της κλάσης `DesicionMaker` με το όνομα filter το οποίο θα αναλάβει να φιλτράρει τα δεδομένα που θα δοθούν ώστε να επιστρέψει τα ελάχιστα. Ο `DesicionMaker` παίρνει κατά την αρχικοποίηση του το μοντέλο που κατασκευάστηκε στην `MainModel`.

```
DesicionMaker filter = new DesicionMaker(main_model.getModel());
```

Ο constructor του `DesicionMaker`.

```
public DesicionMaker(Model model) {

    this.model = model;



}
```

Το σύστημα καλεί την `DataMinimizationFilter()` η οποία παίρνει τον τύπο της υπηρεσίας και τα δεδομένα που ζητήθηκαν.

```
filter.DataMinimizationFilter(service_type, service_required_data);
```

Η συνάρτηση `DataMinimizationFilter` με τον αλγόριθμο ελαχιστοποίησης.

```
public void DataMinimizationFilter(String service_type, ArrayList<String>  service_required_data){


    ArrayList<String> least_data = new ArrayList<String>();


    //data that is required and mayrequired for Service
    ArrayList<String> is_required_data = new ArrayList<String>();
    ArrayList<String> may_required_data = new ArrayList<String>();


    //get data for service
    is_required_data = getRequiredData(service_type, true);
    may_required_data = getRequiredData(service_type, false);



    //search if data that is given is the ones that require. If its ok, its removed for list
    for (String i: service_required_data) {
```

```
    if (is_required_data.contains(i)) {

        is_required_data.remove(i);



    }
else if (may_required_data.contains(i)) {

        may_required_data.remove(i);



   }

}



//Search all the data and the're relations for the required data and print it

for (String x: service_required_data) {



    //the reason for using two arrays is to keep track of the data that have been searched so
far so that its not searched again

    ArrayList<String> total_found_data = new ArrayList<String>();

    ArrayList<String> current_data = new ArrayList<String>();



    //add first data to search its relations

    current_data.add(x);



    //when all the relations of data is search the list will be empty

    while (!current_data.isEmpty()) {



        //extract relation data from current data

        current_data = extractRelationData(current_data, total_found_data);



        //save new extracted data to total data found

        total_found_data.addAll(current_data);



        //whenever a data that is required is found is being printed
```

```
            for (String i: current_data) {
                if (is_required_data.contains(i)) {
                    least_data.add(i);

                }
                else if (may_required_data.contains(i)) {
                    least_data.add(i);
                }

            }

        }

    }

}
```

Η λειτουργία του κώδικα και η λογική πίσω από την υλοποίηση εξηγούνται στην ενότητα που ασχολείται αποκλειστικά με τον αλγόριθμο.

Πρακτικά στην για την προγραμματιστική υλοποίηση η `DataMinimizationFilter()` καλεί άλλες δύο συναρτήσεις την `getRequiredData()`, η οποία παίρνει τον τύπο της υπηρεσίας και την επιλογή εάν θα ψάξει για *requires* ή *mayRequires* σχέσεις στο μοντέλο και επιστρέφει μία λίστα με τα δεδομένα που βρήκε ότι χρειάζονται, και την `extractRelationData()`, με παράμετρούς τα δεδομένα που θα ψάξει και τα συνολικά δεδομένα που έχουν βρεθεί μέχρι στιγμής ώστε να μην τα ξαναψάξει, και επιστρέφει κάθε φορά μία λίστα με τα καινούργια δεδομένα μέσα από αναζήτηση κατά πλάτος.

`getRequiredData()`:

```
//Extracts the required Data for the Service

//parameters: (Service name, option for two different functionalities (true [requires], false [may requires])

private ArrayList<String> getRequiredData(String Service, boolean opt) {


        ArrayList<String> servicedata = new ArrayList<String>();
```

```
            //if subject is the service and object is a data and its relation is requires
its added to list

            if (opt) {

                for (Statement st: model.filter(iri(Service), iri("service:requires"),
null)) {

                    servicedata.add(st.getObject().toString());

                }



            //if subject is the service and object is a data and its relation is
mayrequires its added to list

            } else {

                for (Statement st: model.filter(iri(Service), iri("service:mayRequires"),
null)) {

                    servicedata.add(st.getObject().toString());

                }

            }



        //returns the data needed for the service

        return servicedata;



    }
```

extractRelationData():

```
    //Searches and extracts the data that its requires for the service from the data that
is given

    //parameters: (data that will search in depth, data that is found so far ,so that it
doesn't search same data that have already been searched)

    private ArrayList<String> extractRelationData(ArrayList<String> data, ArrayList<String>
founddata) {



        //save all new data found here
```

```
        ArrayList<String> newdata = new ArrayList<String>();


        //iterates through the model to search data related to the data provided
        for (String i: data) {


            //if data is partOf or isA its added to list
            for (Statement st: model.filter(null, iri("data:isA"), iri(i))) {
                newdata.add(st.getSubject().toString());
            }
            for (Statement st: model.filter(null, iri("data:isPartOf"), iri(i))) {
                newdata.add(st.getSubject().toString());
            }


            //if data is moredetailedthan its added to list
            for (Statement st: model.filter(iri(i), iri("data:isMoreDetailedThan"), null))
{
                newdata.add(st.getObject().toString());
            }
        }


        //removes duplicate data with data that is already been found
        newdata.removeAll(founddata);
        return newdata;
    }
```

Η συνάρτηση `iri()` είναι συνάρτηση του RDF4J και χρησιμοποιήθηκε για να μετατρέψουμε τα String που δίνονταναι μέσα στις συναρτήσεις σε IRI προκειμένου να τα αναγνωρίσει το μοντέλο Περιλαμβάνεται στην διεπαφή `org.eclipse.rdf4j.model.util.Values.iri`

Τέλος, η συνάρτηση η συνάρτηση `filter()` είναι αυτή που επιστρέφει το υπομοντέλο του στοιχείου του μοντέλου που θέλουμε. Δηλαδή επιστρέφει μόνο τα statements που είναι συνδεμένα με το στοιχείο αυτό.

Με την επιστροφή των ελάχιστων δεδομένων τελειώνει η εκτέλεση του συστήματος και επομένως η προγραμματιστική υλοποίηση του.

ΚΕΦΑΛΑΙΟ 6 Παράδειγμα Εκτέλεσης Συστήματος

(Υποκεφάλαιο 6.1) Σενάριο Εκτέλεσης

Έστω ότι ένα άτομο θέλει επισκεφτεί ένα ανταλλακτήριο νομισμάτων στο οποίο για να του επιτραπεί η είσοδος πρέπει να είναι άνω των 18 ετών. Η άδεια της εισόδου στην ιστοσελίδα είναι μία υπηρεσία έστω «EntryAllowance» και τα ελάχιστα δεδομένα που χρειάζεται η υπηρεσία είναι να γνωρίζει εάν το άτομο είναι ενήλικας, επομένως έστω το δεδομένο «IsAdult». Για την είσοδο στο ανταλλακτήριο θα του ζητηθεί η ταυτότητα του, έστω «IDCard». Το πρόβλημα εδώ είναι ότι η ταυτότητα δεν αποτελεί το ελάχιστο πιθανό δεδομένο που μπορεί να αναδείξει για να του επιτραπεί είσοδος αφού περιέχει πολλά προσωπικά δεδομένα τα οποία είναι αχρείαστα για την υπηρεσία. Ταυτόχρονα όμως σύμφωνα με το μοντέλο των προσωπικών δεδομένων το δεδομένο «IsAdult» είναι ένα μέρος του «IDCard». Το σύστημα που αναπτύχθηκε φροντίζει ότι το άτομο κατά την είσοδο του δεν θα του ζητηθεί το δεδομένο «IDCard» που ζητάει αρχικά η υπηρεσία αλλά θα αποφασίσει ότι για την υπηρεσία «EntryAllowance» θα επιστρέψει το δεδομένο «IsAdult» που είναι το ελάχιστο δεδομένο που χρειάζεται και αποσπάτε μέσα από το «IDCard».

(Υποκεφάλαιο 6.2) Υλοποίηση Σεναρίου

Θεωρώντας ότι το σύστημα λειτουργεί με proxy όπως αναφέρθηκε κι στο 3.1 Γενική Επισκόπηση του Κεφαλαίου 3, η υλοποίηση του σεναρίου θα ακολουθεί την ροή που περιγράφεται παρακάτω:

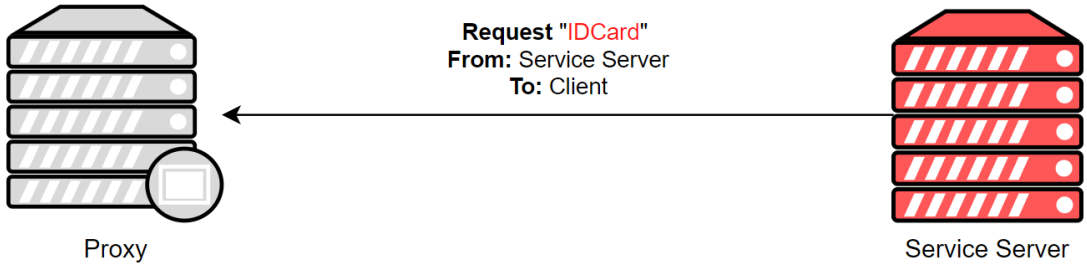
1. Αρχικά, ο χρήστης στέλνει ένα αίτημα(request) στην ιστοσελίδα του ανταλλακτηρίου για να την επισκεφτεί, το αίτημα http GET ζητάει την ιστοσελίδα εισόδου από τον διακομιστή της υπηρεσίας(Service Server).

2. Ο διακομιστής της υπηρεσίας στέλνει ένα αίτημα http GET στον χρήστη μέσα στο οποίο ζητάει το δεδομένο IDCard

Το αίτημα http post και θα έχει την μορφή:

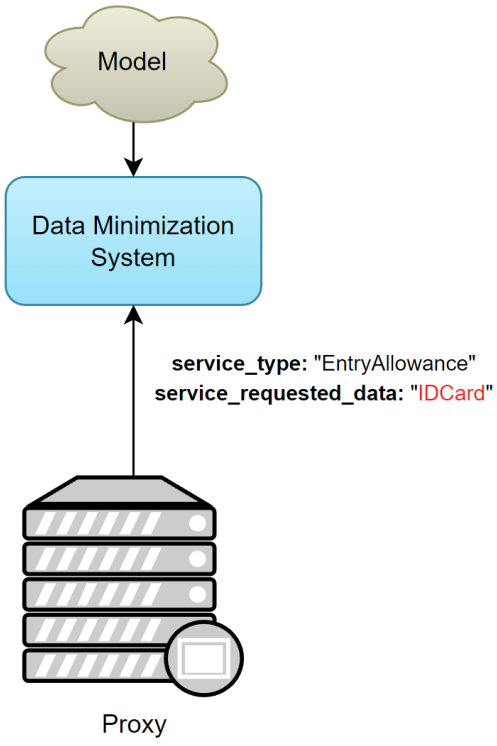
```
GET /exchange.html?service=”EntryAllowance”&data=”IDCard” HTTP/1.1
Host: www.exchange.entry.service.com
Content-Type: application/www-form-urlencoded
Accept-Language: en-us
```

3. Το αίτημα δεν θα σταλθεί στον χρήστη καθώς πρώτα θα πάει στον proxy.



Σχήμα 12: Αναπαράσταση αιτήματος υπηρεσίας

4. Ο proxy στέλνει δεδομένα του αιτήματος που πήρε από την υπηρεσία στο σύστημα ελαχιστοποίησης δεδομένων. Επομένως το σύστημα θα λάβει τον τύπο της υπηρεσίας EntryAllowance και τα δεδομένα που ζητάει IDCard.

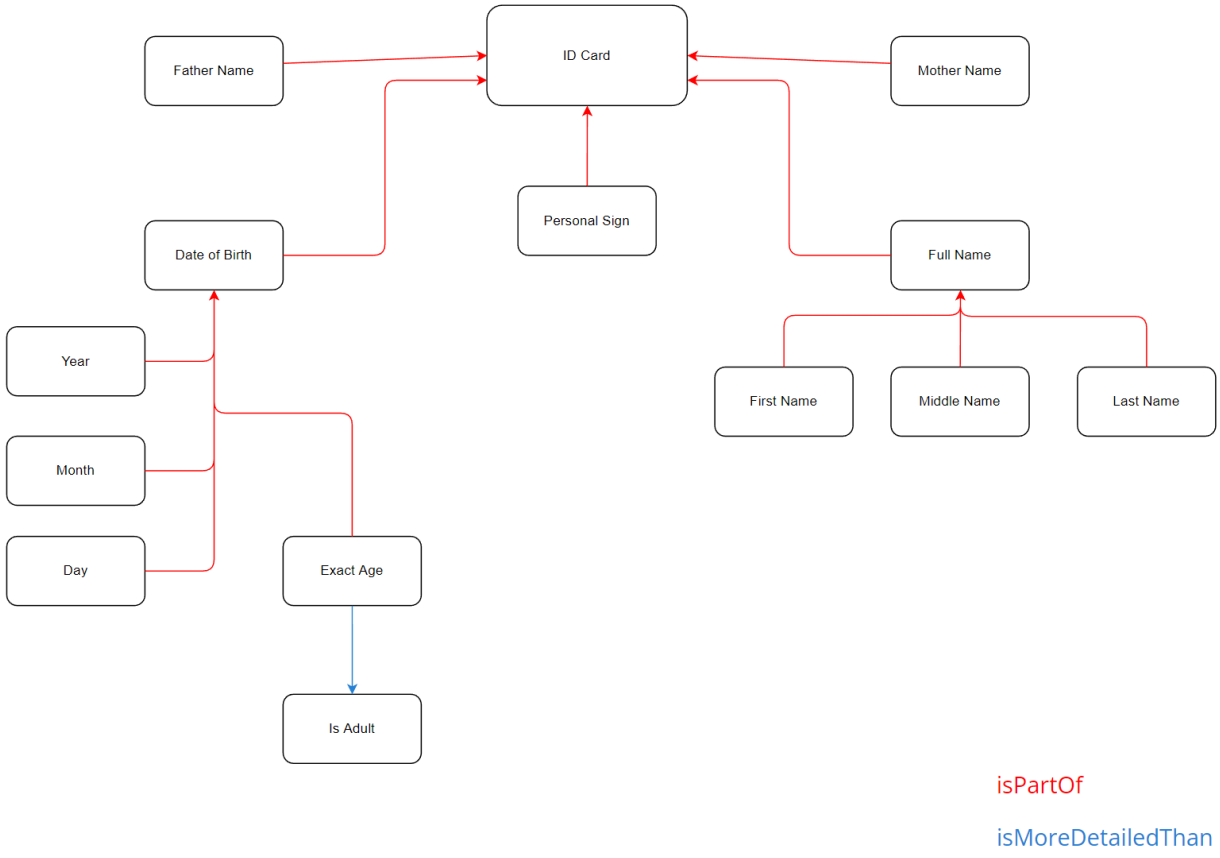


Σχήμα 13: Τα στοιχεία που ζήτησε η υπηρεσία δίνονται στο σύστημα ελαχιστοποίησης δεδομένων

5. Με είσοδο service_type: EntryAllowance τον τύπος υπηρεσίας, και service_requested_data: IDCard το δεδομένο.

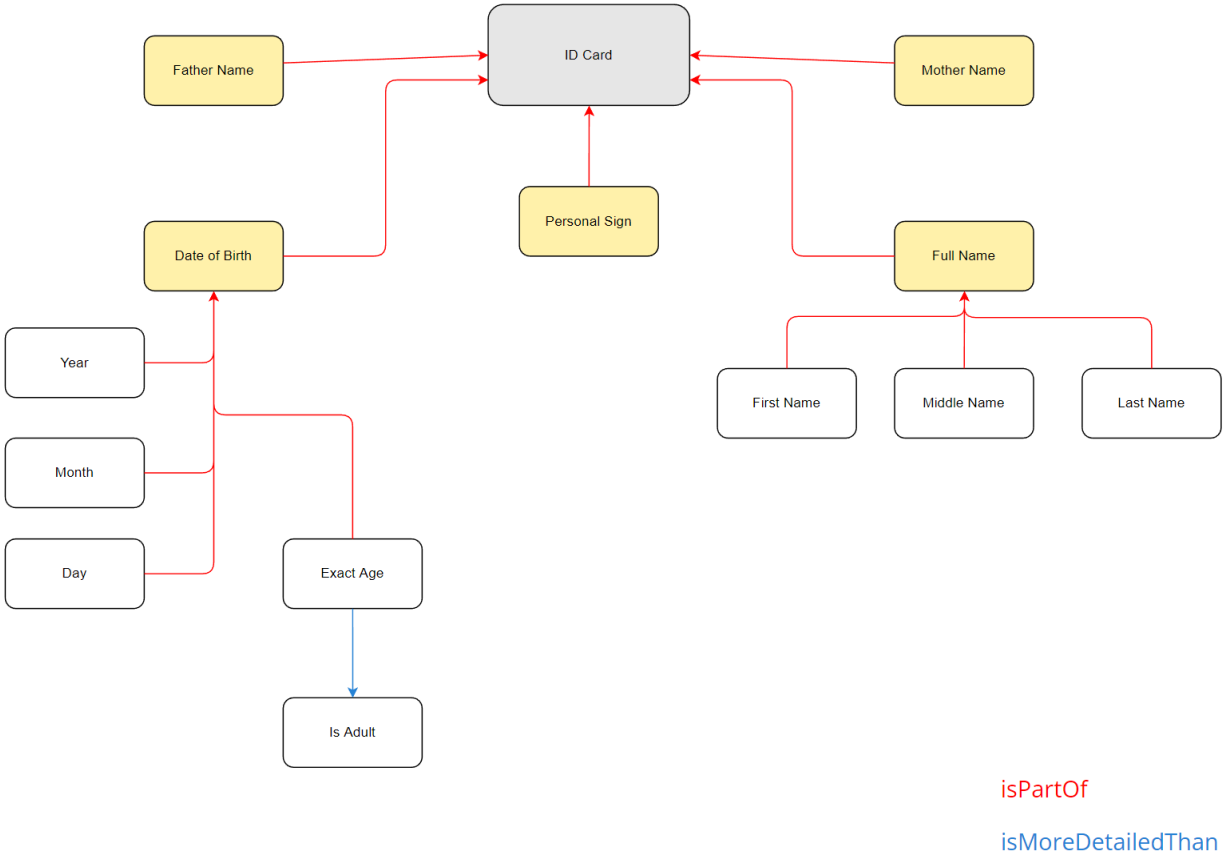
6. Μέσα από το μοντέλο βρίσκει ότι το ελάχιστο δεδομένο που χρειάζεται ο τύπος της υπηρεσίας είναι το δεδομένο IsAdult

7. Για κάθε ένα από τα δεδομένα που δόθηκαν μέσω του service_requested_data, στην προκειμένη περίπτωση για το IDCard ψάχνει μέσα στο υπομοντέλο του, μία αναπαράσταση αυτού απεικονίζεται στο Σχήμα 14.



Σχήμα 14: Διάγραμμα ροής της λειτουργίας του συστήματος

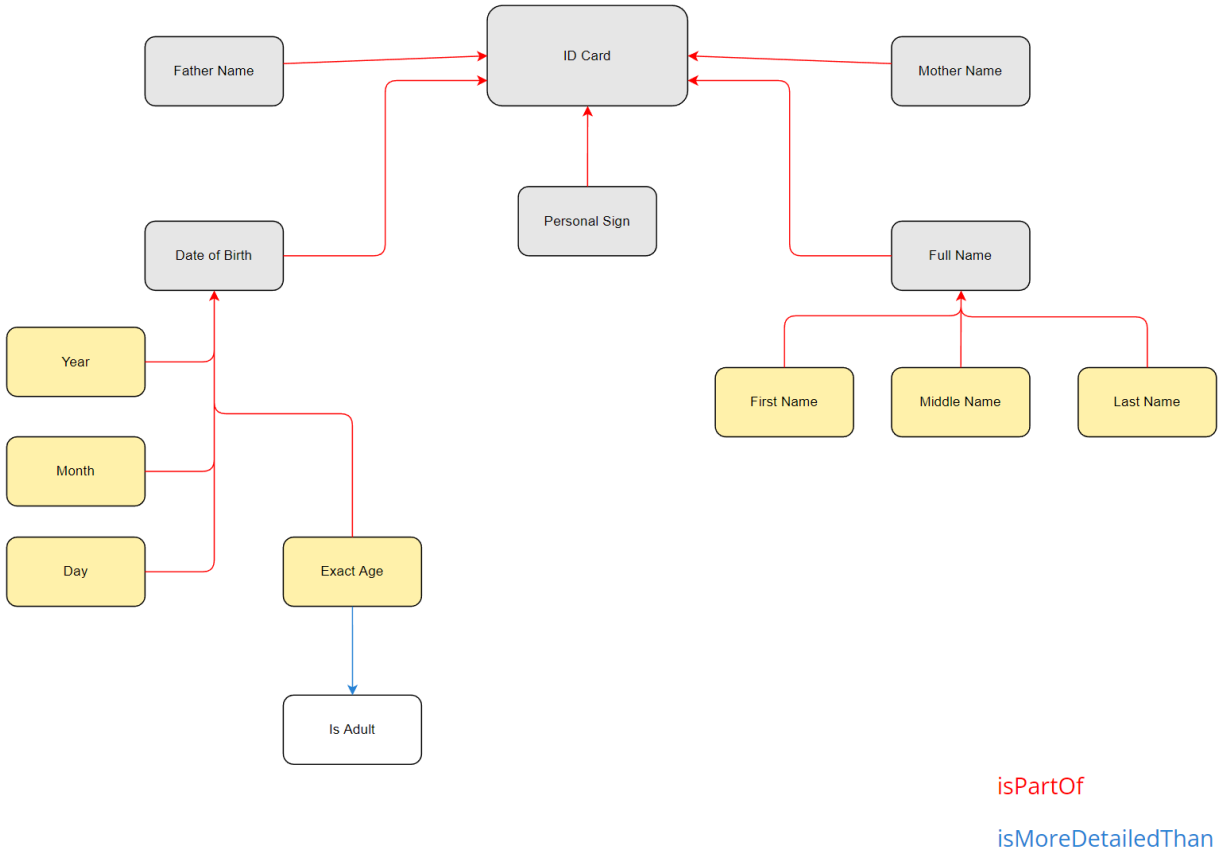
8. Στην πρώτη αναζήτηση ψάχνει ποια δεδομένα έχουν σχέσεις isPartOf ή isMoreDetailedThan με το δεδομένο ID Card. Από το υπομοντέλο μπορούμε να δούμε ότι τα αποτελέσματα της πρώτης αναζήτησης είναι τα: FatherName, Mother Name, Personal Sign, Full Name και Date Of Birth. Στο Σχήμα 15 τονίζονται με κίτρινο χρώμα.



Σχήμα 15: Αποτελέσματα πρώτης αναζήτησης

9. Ελέγχει εάν από τα δεδομένα που βρήκε είναι κάποιο από τα ελάχιστα της υπηρεσίας. Εδώ δεν είναι κανένα από αυτά που βρέθηκαν επομένως συνεχίζει την αναζήτηση

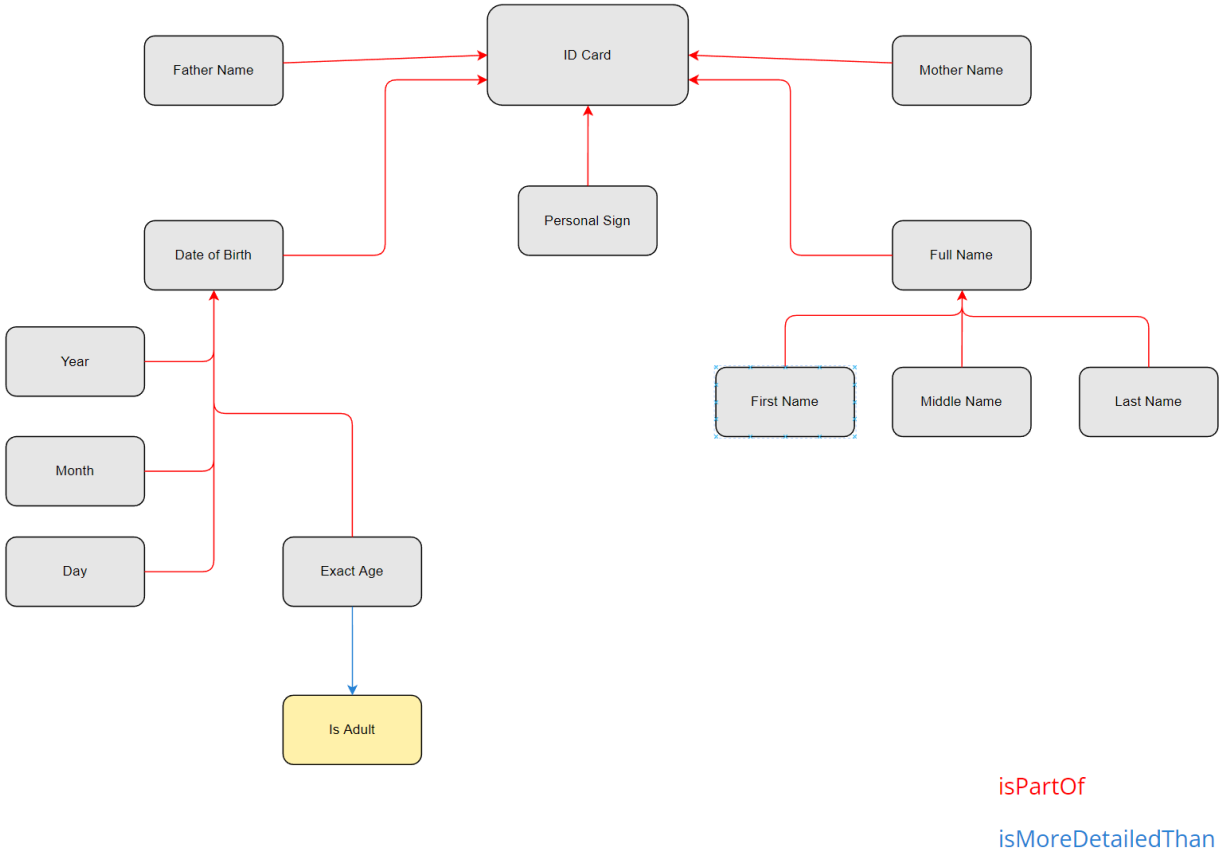
10. Στην επόμενη αναζήτηση ψάχνει για τις σχέσεις των Father Name, Mother Name, Personal Sign, Full Name και Date Of Birth. Εδώ βλέπουμε ότι τα δεδομένα Father Name, Mother Name, Personal Sign δεν έχουν άλλες συνδέσεις. Επομένως ο αλγόριθμος θα προκύψει με τα δεδομένα Year, Month, Day, Exact Age που βρέθηκαν μέσα από το Date Of Birth και τα First Name, Middle Name και Last Name μέσα από το Full Name. Στο Σχήμα 16 με κίτρινο τονίζονται ποια είναι τα δεδομένα που προκύπτουν από την δεύτερη αναζήτηση. Με γκρι χρώμα χρωματίζονται τα δεδομένα που έχουν ελεγχθεί.



Σχήμα 16: Αποτελέσματα δεύτερης αναζήτησης

11. Ελέγχει εάν από τα δεδομένα που βρήκε είναι κάποιο από τα ελάχιστα της υπηρεσίας. Εδώ πάλι δεν είναι κανένα από αυτά που βρέθηκαν επομένως συνεχίζει την αναζήτηση.

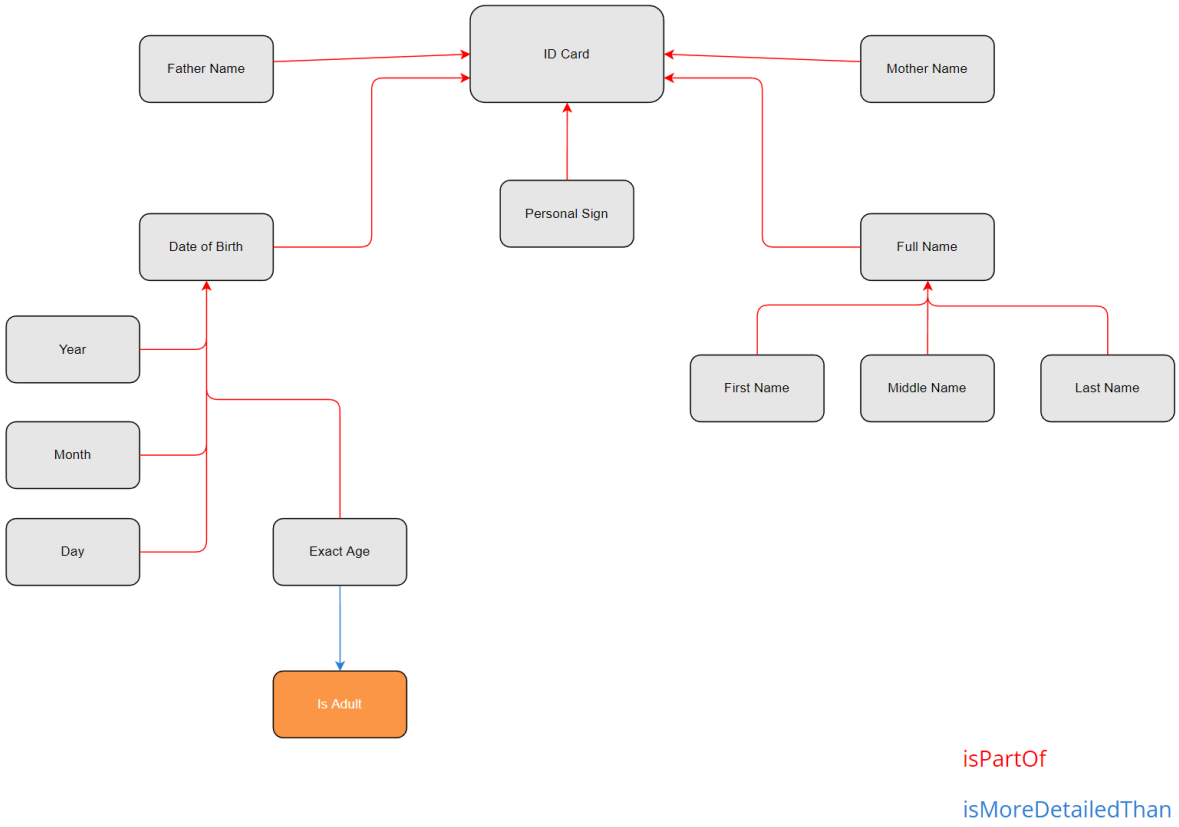
12. Στην επόμενη αναζήτηση ψάχνει για τις σχέσεις των Year, Month, Day, Exact Age που βρέθηκαν μέσα από το Date Of Birth και τα First Name, Middle Name και Last Name που βρέθηκαν μέσα από το Full Name. Το μοναδικό δεδομένο που βρίσκει είναι το IsAdult καθώς τα άλλα δεδομένα δεν έχουν κάποια άλλη σχέση. Στο σχήμα 7.6 με κίτρινο τονίζεται το μοναδικό δεδομένο που προκύπτει από την τρίτη αναζήτηση. Με γκρι χρώμα χρωματίζονται τα δεδομένα που έχουν ελεγχθεί.



Σχήμα 17: Αποτελέσματα τρίτης αναζήτησης

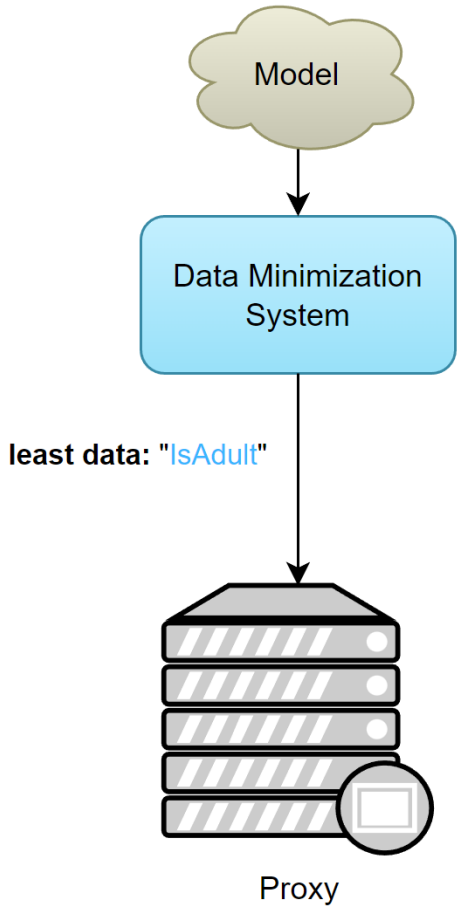
13. Ελέγχει εάν από τα δεδομένα που βρήκε είναι κάποιο από τα ελάχιστα της υπηρεσίας. Διαπιστώνει ότι το δεδομένο που βρήκε είναι αυτό που αποτελεί το ελάχιστο δεδομένο για τον τύπο της υπηρεσίας.

14. Επειδή από το IsAdult δεν έχει κάποια άλλη ανεξερεύνητη σχέση μέσα στο υπομοντέλο, τερματίζει η αναζήτηση κι τα ελάχιστα δεδομένα που βρέθηκαν, στην περίπτωση μας είναι το δεδομένο του τύπου IsAdult. Στο Σχήμα 18 με πορτοκαλί χρώμα τονίζεται το δεδομένο που συμπέρανε το σύστημα ότι είναι τα ελάχιστα.



Σχήμα 18: Αποτέλεσμα αλγόριθμου ελαχιστοποίησης δεδομένων

15. Το αποτέλεσμα της ελαχιστοποίησης δίνονται στο proxy ώστε να φτιάξει το νέο αίτημα για τον χρήστη



Σχήμα 19: Επιστροφή ελάχιστων δεδομένων από το σύστημα ελαχιστοποίησης δεδομένων

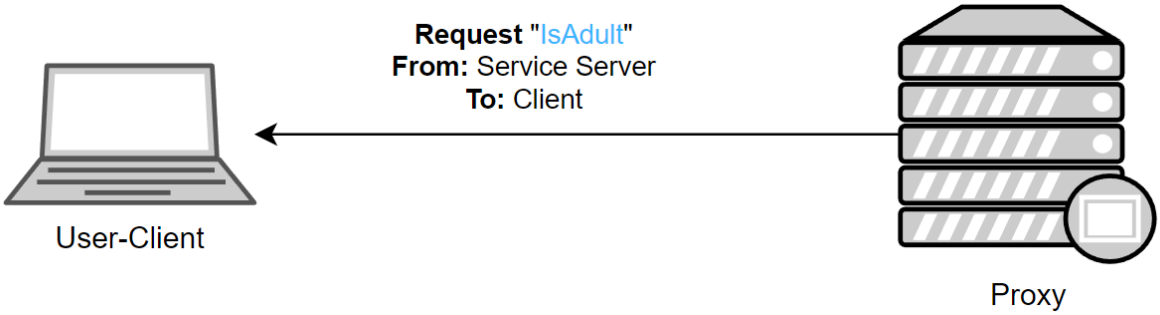
16. Το νέο αίτημα που θα σταλεί στον χρήστη περιέχει τα επιθυμητά ελάχιστα δεδομένα.

    Το νέο αίτημα http και θα έχει την μορφή:

```
GET/exchange.html?service=”EntryAllowance”&data=”IsAdult” HTTP/1.1
Host: www.exchange.entry.service.com
Content-Type: application/www-form-urlencoded

Accept-Language: en-us
```

17. Το αίτημα στέλνεται στον χρήστη-πελάτη



Σχήμα 20: Ο proxy στέλνει το φιλτραρισμένο αίτημα στον χρήστη

ΚΕΦΑΛΑΙΟ 7 Συμπεράσματα – Μελλοντική Εργασία

Βάση και κίνητρο για την πραγματοποίηση της παρούσας πτυχιακής αποτέλεσε η αναδυόμενη τάση, τόσο νομική όσο και τεχνολογική στην επεξεργασία και αποθήκευση δεδομένων προσωπικού χαρακτήρα, με στόχο την εφαρμογή της αρχής ελαχιστοποίησης δεδομένων μέσα από ένα σύστημα.

Βασικές συνεισφορές της προτεινόμενης λύσης προς αυτή την κατεύθυνση αποτέλεσαν τεχνολογίες σημασιολογικού ιστού που μας επιτρέπουν να περιγράψουμε τύπους προσωπικών δεδομένων, τύπους διαφόρων υπηρεσιών και κυρίως τις σημασιολογικές σχέσεις μεταξύ αυτών, έτσι ώστε αυτές οι σχέσεις να μπορούν να ερμηνευτούν υπολογιστικά. Δημιουργώντας ένα σύστημα που έχει ένα μοντέλο RDF που ερμηνεύει έναν γράφο γνώσης με τις συσχετίσεις των δεδομένων και έναν αλγόριθμο που πραγματοποιεί αναζήτηση στο μοντέλο εξετάζοντας τις συσχετίσεις του μοντέλου που αποτελούν κανόνες προκειμένου να αποσπάσει ελάχιστα δεδομένα που χρειάζεται για την εξυπηρέτηση κάθε υπηρεσίας λαμβάνοντας υπόψιν τα δεδομένα που δόθηκαν. Τέλος, συμπεραίναμε ότι πολλά από τα ελάχιστα δεδομένα μπορούν να αποσπαστούν μέσα από τα αρχικά δεδομένα που δίνονται στο σύστημα.

Η υλοποίηση του συστήματος και οι κανόνες που θέτει το μοντέλο έχουν προδιαγραφεί με έναν ιδιόκτητο τρόπο με μία χαρτογράφηση των σχέσεων μεταξύ τύπων υπηρεσιών και τύπων προσωπικών δεδομένων. Οπότε θα ήταν επιθυμητό οι κανόνες αυτοί να υιοθετηθούν σε κάποια άλλα πρότυπη γλώσσα προδιαγραφής ελέγχου πρόσβασης και χρήσης όπως είναι η XCML, για μία πιο ισχυρή προσέγγιση για την καταγραφή τύπων δεδομένων και των σχέσεων μεταξύ αυτών, και όχι απλά μία συσχέτιση rdf.

Επιπλέον θα μπορούσε να γίνει ενσωμάτωση με ένα proxy HTTP (όπως αναφέρεται στο παράδειγμα εκτέλεσης στο Κεφάλαιο 6.2) ώστε το σύστημα να γίνει πιο ευέλικτο. Ταυτόχρονα το proxy θα επιτρέψει την δημιουργία ενός API ώστε να έχει πρόσβαση σε πολλούς πόρους ή δυνατότητες του ενσωματωμένου τελικού σημείου HTTP ώστε να γίνεται η ελαχιστοποίηση.

Αντίστοιχα, θα μπορούσε το σύστημα να λειτουργήσει ως ένα plug-in σε browser που θα τρέχει τον αλγόριθμο και να ρυθμίζονται τα δεδομένα που μπορεί να δώσει στην ανάλογη υπηρεσία που χρησιμοποιείται ώστε ο χρήστης να μπορεί να κατεβάσει απλά κι εύκολα και να απολαμβάνει την λειτουργία του.

Λαμβάνοντας υπόψιν την λειτουργία του συστήματος θα μπορούσαμε να επικεντρώσουμε σε αλλαγές πάνω στο μοντέλα δεδομένων. Πιο συγκεκριμένα να εμπλουτιστούν τα μοντέλα δεδομένων κι υπηρεσιών ούτως ώστε να ανταποκρίνονται σε πραγματικές ανάγκες του πραγματικού κόσμου φτιάχνοντας τους γράφους έτσι ώστε να περιλαμβάνουν χιλιάδες εγγραφές με πιο συνηθισμένους τύπους δεδομένων και ακόμα μεγαλύτερο αριθμό υπηρεσιών.

Τέλος θα μπορούσε να γίνει συνεργασία με κάποιον δικηγόρο ούτε ώστε να φτιαχτούν κανόνες και συσχετίσεις νομικού πλαισίου που να ανταποκρίνονται στην νομοθεσία περί προστασίας δεδομένων και την επεξεργασία και αποθήκευση αυτών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] S. C. Kokott J., «The dinstiction between privacy and data protection in the jurisprudence of the CJEU and the ECtHR,» *International Data Privacy* , 2013.

[2] Π. Ε., Προσωπικά Δεδομένα Προστασία GDPR (1η Έκδοση), Αθήνα: Εκδόσεις Παπαδόπουλος, 2018.

[3] Τ. Ε. Κ. Κ. Τ. Σ. Τ. Ε. ΕΝΩΣΗΣ, *ΚΑΝΟΝΙΣΜΟΣ (ΕΕ) 2016/679 ΤΟΥ ΕΥΡΩΠΑΪΚΟΥ ΚΟΙΝΟΒΟΥΛΙΟΥ ΚΑΙ ΤΟΥ ΣΥΜΒΟΥΛΙΟΥ της 27ης Απριλίου 2016 για την προστασία των φυσικών προσώπων έναντι της επεξεργασίας των δεδομένων προσωπικού χαρακτήρα και για την ελεύθερη κυκλοφορία των δεδομένων αυτών,* 2016.

[4] Η. Έθνη, *Οικουμενική Διακήρυξη για τα Ανθρώπινα Δικαιώματα.,* Δεκέμβριος 1948.

[5] ". 1. O. 1. Geseĵ- und VerordnungsblaĴ für das Land Hessen - Teil I - Nr. 4.

[6] U.S.C, *Public Law No. 93-579, 88 Stat. 1897,* 31 December 1974.

[7] O. f. E. C.-o. a. Development, *OECD guidelines on the protection of privacy and transborder flows of personal data.,* 1980.

[8] E. P. a. Council, «Directive 95/46/EC of the European Parliament and of the Council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data,» *Official Journal of the European Communities,* November 1995.

[9] J. H. a. O. L. T. Berners-Lee, «The Semantic Web,» *Scientific American,* May 2001.

[10] M. K. a. S. R. P. Hitzler, Foundations of Semantic Web Technologies, FL, USA, 2009.

[11] V. U. A. Y. R. B. Guus Schreiber, «RDF 1.1 Primer,» W3C Working Group Note, 24 June 2014. [Ηλεκτρονικό]. Available: https://www.w3.org/TR/rdf11-primer/.

[12] O. C.-G. a. M. F.-L. A. Gomez-Perez, Ontological Engineering(1st ed.), USA: Secaucus, 2003.

[13] P. F. P.-S. a. B. P. B. Motik, «OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax,» October 2009. [Ηλεκτρονικό]. Available: http://www.w3.org/TR/owl2-syntax/.

[14] L. Ehrlinger και W. Wöß, «Towards a Definition of Knowledge Graphs,» σε *12th International Conference on Semantic Systems*, 2016.