ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ

ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΚΑΙ ΥΠΗΡΕΣΙΕΣ

Κατεύθυνση: Δικτυοκεντρικά Πληροφοριακά Συστήματα

ΔΙΑΛΕΙΤΟΥΡΓΙΚΟΤΗΤΑ ΣΥΣΤΗΜΑΤΩΝ ΔΙΑΧΕΙΡΙΣΗΣ ΙΑΤΡΙΚΩΝ ΦΑΚΕΛΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΩΝ ΠΡΟΤΥΠΩΝ FHIR ΚΑΙ epSOS Patient Summary

Χάρης-Μάριος Βρεττός – ΜΕ14021

Επιβλέπων: Δημοσθένης Κυριαζής, Επίκουρος Καθηγητής

Περιεχόμενα

Κατάλογος Σχημάτων

Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον Επίκουρο Καθηγητή του Τμήματος «Ψηφιακών Συστημάτων» κ. Δημοσθένη Κυριαζή και τον κ. Ανδρέα Μενύχτα, για την αμέριστη βοήθειά τους και καθοδήγησή καθ' όλη την διάρκεια της εκπόνησης της πτυχιακής μου.

Επίσης θα ήθελα να ευχαριστήσω τους γονείς μου, που κατά τον 1.5 χρόνο φοίτησης μου στο τμήμα «Ψηφιακών Συστημάτων» του Πανεπιστημίου Πειραιά μου προσέφεραν οικονομική αλλά πάνω απ' όλα ψυχολογική στήριξη για να μπορέσω να ολοκληρώσω τις σπουδές μου.

Συντομογραφίες

ΕΛΛΗΝΙΚΕΣ

ΠΔΙΠΑ - Περίληψη Δεδομένων Ιατροφαρμακευτικής Περίθαλψης Ασθενή

ΑΜΚΑ – Αριθμός Μητρώου Κοινωνικής Ασφάλισης

ΛΑΤΙΝΙΚΕΣ

FHIR - Fast Healthcare Interoperability Resources

epSOS - European Patients Smart Open Services

PS - Patient Summary

REST - Representational State Transfer

JSON - JavaScript Object Notation

XML - eXtensible Markup Language

NCP - National Contact Point

API - Application Programming Interface

HTTP - HyperText Transfer Protocol

CSS – Cascading StyleSheets

DSTU – Draft Standard for Trial Use

EMR – Electronic Medical Records

UDDI – Universal Description, Discovery and Integration

WSDL – Web Service Definition Language

SOAP – Simple Object Access Protocol

URI – Uniform Resource Identifier

1. Εισαγωγή

Τη σημερινή εποχή η τεχνολογία χρησιμοποιείται παντού, κάνοντας τη ζωή μας πολύ πιο εύκολη και απλή, από τα πιο απλά πράγματα, όπως την παρασκευή καφέ μέχρι και τα πιο σύνθετα όπως τα ταξίδια του ανθρώπου στο διάστημα. Ένας τομέας στον οποίο η τεχνολογία έχει βοηθήσει σε πολύ μεγάλο βαθμό είναι αυτός της Υγείας. Εκτός του ότι χρησιμοποιείται ως εργαλείο στα χέρια των γιατρών για να σώσουν εκατομμύρια ζωές καθημερινά σε όλο τον κόσμο, γίνεται επίσης και μέσο για τους υπαλλήλους που εργάζονται στον τομέα της Υγείας και τους ασθενείς να έχουν μια πιο ευχάριστη εργασιακή εμπειρία και εξυπηρέτηση αντίστοιχα. Η καταγραφή των ιατρικών δεδομένων του ασθενούς σε ηλεκτρονική μορφή συνέβαλε σε αυτό.

Για πάρα πολλά χρόνια η καταγραφή και αποθήκευση των ιατρικών δεδομένων γινόταν σε χάρτινη μορφή κάνοντας την διαχείριση τους μια χρονοβόρα και κουραστική διαδικασία. Φυσικά δεν είναι λίγες οι περιπτώσεις στο παρελθόν που οι ιατρικοί φάκελοι των ασθενών είχαν καταστραφεί από φυσικές καταστροφές, όπως φωτιές και πλημμύρες ή είχαν κλαπεί από τις κλινικές δημιουργώντας πολλά προβλήματα. Πέρα από τις φυσικές καταστροφές που μπορούσαν να υποστούν, η διαχείριση τους ήταν και αυτή ένα δύσκολο εγχείρημα. Είναι σαφές ότι οι φάκελοι αυτοί θα πρέπει ανά τακτά χρονικά διαστήματα να ενημερώνονται με αποτελέσματα ιατρικών εξετάσεων του ασθενούς, ασθένειες, εγχειρίσεις και γενικά με ιατρικά δεδομένα που μπορεί να βοηθήσουν τους γιατρούς που εξετάζουν τον ασθενή να βγάλουν μια όσο το δυνατόν πιο ακριβή διάγνωση. Η αλλαγή και ενημέρωση των χάρτινων ιατρικών φακέλων σε χιλιάδες ασθενείς είναι μια αρκετά χρονοβόρα διαδικασία και το ποσοστό λανθασμένης καταχώρισης είναι αρκετά μεγάλο. Το μεγαλύτερο πρόβλημα όμως είναι το γεγονός ότι οι χάρτινοι ιατρικοί φάκελοι φυλάσσονται σε ένα μόνο μέρος και δεν υπάρχει πρόσβαση από τρίτους. Παραδείγματος χάρη σε περίπτωση που κάποιος ταξίδευε στο εξωτερικό ή ακόμα και σε μια άλλη πόλη της Ελλάδας και πάθαινε κάτι δεν θα μπορούσε ο θεράπων ιατρός να έχει πρόσβαση στα ιατρικά δεδομένα του, με αποτέλεσμα να κάνει λανθασμένη ή ακόμα και αργοπορημένη διάγνωση που στην χειρότερη περίπτωση θα μπορούσε να στοιχίσει ακόμα και τη ζωή στον ασθενή.

Για την επίλυση όλων των παραπάνω προβλημάτων είδη από το 1960, άρχισαν να εμφανίζονται οι πρώτοι ηλεκτρονικοί ιατρικοί φάκελοι, δηλαδή η ψηφιοποιημένη μορφή των ιατρικών δεδομένων του ασθενούς.

Σήμερα, σύμφωνα με την έκθεση του Bloomberg News[1] οι δέκα χώρες που έχουν το μεγαλύτερο ποσοστό υιοθέτησης ηλεκτρονικών ιατρικών φακέλων είναι η Νορβηγία(98%), η Ολλανδία(98%), το Ηνωμένο Βασίλειο(97%), η Νέα Ζηλανδία(97%), η Αυστραλία(92%), η Γερμανία(82%), οι Ηνωμένες Πολιτείες(69%), η Γαλλία(67%), ο Καναδάς(56%) και η Ελβετία(41%).

Έτσι λοιπόν, τα κλινικά δεδομένα του ασθενούς δεν μπορούν να καταστραφούν από φυσικές καταστροφές καθώς είναι καταχωρημένα σε κάποιο απομακρυσμένο διακομιστή. Ακόμα, λύνοντας το πρόβλημα της περιορισμένης πρόσβασης από τρίτους, από τη στιγμή που δεν είναι τοπικά αποθηκευμένα τα δεδομένα μπορούν και άλλοι γιατροί να έχουν πρόσβαση πέρα από αυτόν που συνήθως υποθάλπει τον ασθενή σε μια συγκεκριμένη κλινική. Τέλος όσον αφορά τη διαχείριση των φακέλων, δημιουργήθηκαν και κάποια συστήματα που την κάνουν πολύ πιο εύκολη και γρήγορη, μειώνοντας ταυτόχρονα το ποσοστό λανθασμένης εκχώρησης στη βάση σώζοντας ακόμα και ζωές εξαιτίας πιθανής λανθασμένης καταγραφής στοιχείων.

Σε κάθε περίπτωση όμως με την επίλυση κάποιου προβλήματος μπορεί να παρουσιαστούν και πολλά άλλα. Στην περίπτωση των συστημάτων διαχείρισης ηλεκτρονικών ιατρικών φακέλων, μπορεί να λύνονται πολλά προβλήματα σχετικά με τη διαχείριση, τη μείωση του κόστους, τη μείωση του ποσοστού λάθους εκχώρησης αλλά ταυτόχρονα δημιουργείται ένα μείζων θέμα, η διαλειτουργικότητα μεταξύ τέτοιων συστημάτων.

Σκοπός λοιπόν της εν λόγω διπλωματικής εργασίας είναι να προτείνει μια προσέγγιση για την επίλυση αυτού του προβλήματος, ώστε να γίνει πιο εύκολος ο διαμοιρασμός των ιατρικών δεδομένων των ασθενών τόσο σε εθνικό όσο και σε διεθνές επίπεδο.

Η προσέγγιση στο πρόβλημα που προτείνεται χρησιμοποιεί δύο βασικά στοιχεία. Το πρώτο είναι το ΠΔΙΔΑ(PS)[2] που αποτελείται από ένα πρωτυποποιημένο σύνολο από σημαντικά κλινικά δεδομένα απαιτούμενα για να διασφαλιστεί μια ασφαλή ιατροφαρμακευτική περίθαλψη. Το δεύτερο είναι το FHIR[3], το οποίο περιλαμβάνει ένα σύνολο από ιατρικούς πόρους, που

αντιπροσωπεύουν κλινικές έννοιες, βασιζόμενοι στις δομές δεδομένων JSON[4] και XML[5] αλλά και ένα RESTful πρωτόκολλο βασισμένο στο HTTP[6]. Παραδείγματος χάρη ο πόρος του FHIR, Patient περιέχει στοιχεία για την χαρτογράφηση σημαντικών πληροφοριών για τον ασθενή όπως όνομα, επώνυμο, στοιχεία επικοινωνίας και πολλά άλλα. Αυτά τα δύο συνιστούν τον πυρήνα πίσω από την ιδέα.

Ως αναφορά την υλοποίησή της, τα ιατρικά δεδομένα αποθηκεύονται σε μια απομακρυσμένη βάση δεδομένων η οποία είναι σχεδιασμένη σύμφωνα με το ΠΔΙΔΑ. Για την λειτουργικότητα, έχει υλοποιηθεί ένα Web Service[7] το οποίο παίρνει ως είσοδο πόρους FHIR και παράγει ως έξοδο πόρους FHIR. Πιο συγκεκριμένα παίρνει τα δεδομένα που υπάρχουν στα πεδία τους και τα αποθηκεύει στα κατάλληλα πεδία στη βάση. Για την αντίστροφη δουλειά το πρόγραμμα παίρνει τα δεδομένα από την βάση, τα βάζει στα κατάλληλα πεδία του πόρου και τα επιστρέφει στο χρήστη που έκανε την αίτηση για τα συγκεκριμένα δεδομένα.

Όλη αυτή η διαδικασία έχει ως σκοπό να χρησιμοποιείται το FHIR ως κοινός συνδετικός κρίκος για την ανταλλαγή ιατρικών δεδομένων ώστε να λυθεί το πρόβλημα της διαλειτουργικότητας μεταξύ των συστημάτων διαχείρισης ηλεκτρονικών ιατρικών φακέλων.

Εν τέλει μέσα στη διπλωματική, για την επίδειξη της λειτουργίας που συζητήθηκε παραπάνω, έχουν ετοιμαστεί και δύο ακόμα προγράμματα, μια εφαρμογή Android και ένα Java πρόγραμμα. Η εφαρμογή Android συνδέεται με το σύστημα διαχείρισης ηλεκτρονικών ιατρικών φακέλων HealthVault της Microsoft και χρησιμοποιεί το πιο πάνω FHIR Web Service για να περάσει τα δεδομένα που έχει αποθηκεύσει ο χρήστης στη κεντρική βάση δεδομένων. Η Java εφαρμογή παίρνει κάποια εργαστηριακά δεδομένα της Βιοιατρικής μέσω ενός έτοιμου REST Web Service και τα αποθηκεύει στη βάση χρησιμοποιώντας το FHIR Web Service.

2. Ανασκόπηση Βιβλιογραφίας

Σε αυτό το κεφάλαιο, θα αναλυθούν όλες οι τεχνολογίες, συστήματα και πρότυπα που χρησιμοποιήθηκαν για την επίλυση του προβλήματος της διαλειτουργικότητας των συστημάτων διαχείρισης ηλεκτρονικών ιατρικών φακέλων.

2.1. epSOS

Το epSOS[8] είναι ένα μεγάλης κλίμακας πρόγραμμα πάνω στην ηλεκτρονική υγεία, με προϋπολογισμό 36.5 εκατομμυρίων ευρώ που «έτρεξε» πιλοτικά για 6 χρόνια(1^{η} Ιουλίου 2008 – 31^{η} Ιουνίου 2014) και περιλαμβάνει σαράντα πέντε ιδρύματα από είκοσι πέντε διαφορετικές ευρωπαϊκές χώρες(είκοσι δύο εντός ευρωπαϊκής ένωσης και τρείς εκτός ευρωπαϊκής ένωσης). Στόχος του προγράμματος είναι η βελτίωση της ποιότητας και ασφάλειας της ιατροφαρμακευτικής περίθαλψης των πολιτών όταν ταξιδεύουν σε ξένα Ευρωπαϊκά κράτη. Επίσης στοχεύει στην ανάπτυξη μιας ICT(Information and Communication Technology) υποδομής που θα επιτρέπει την ασφαλή πρόσβαση σε ιατρικά δεδομένα ασθενών μεταξύ διαφόρων ευρωπαϊκών συστημάτων υγείας. Αυτή η υποδομή ωφελεί και τους ασθενείς αλλά και τους γιατρούς στην αναζήτηση αλλά και προσφορά αντίστοιχα υπηρεσιών υγείας.

Με την συνένωση διαφορετικών υποδομών ηλεκτρονικής υγείας μεταξύ κρατών, η αρχιτεκτονική epSOS επιτρέπει την ανταλλαγή δεδομένων ακόμα και εκτός συνόρων που είναι άλλωστε και ο πρωτεύον στόχος του προγράμματος. Το epSOS μπορεί να κάνει μια σημαντική συνεισφορά στην ασφάλεια των ασθενών ελαττώνοντας τα ιατρικά λάθη και παρέχοντας γρήγορη πρόσβαση σε έγγραφα με ιατρικά δεδομένα αλλά και σε συνταγογραφημένα φάρμακα από το εξωτερικό. Σε περιπτώσεις έκτακτης ανάγκης, αυτά τα έγγραφα παρέχουν ιατρικές πληροφορίες που θα μπορούσαν να σώσουν ζωές και να μειώσουν το ποσοστό λάθους κατά την διάρκεια διαγνωστικών διαδικασιών.

Σε πρώτη φάση δοκιμάστηκαν οι εξής υπηρεσίες σε πολυεθνικό επίπεδο, η υπηρεσία ΠΔΙΔΑ και η υπηρεσία ηλεκτρονικής συνταγογράφησης με τη χρήση συστημάτων ePrescription.

Ηλεκτρονική Συνταγογράφηση – ePrescription

Η υπηρεσία ePrescription περιλαμβάνει την ηλεκτρονική συνταγογράφηση(ePrescribing) και την ηλεκτρονική διανομή(eDispensing).

- Το ePrescribing περιλαμβάνει την ηλεκτρονική συνταγογράφηση φαρμάκων με τη χρήση λογισμικού από νομικά εξουσιοδοτημένους γιατρούς και την ηλεκτρονική μεταφορά της συνταγής σε κάποιο φαρμακείο όπου στην συνέχεια θα μπορέσει το φάρμακο να διανεμηθεί στον ασθενή.
- Το eDispensing ορίζεται ως η ηλεκτρονική ανάκτηση της συνταγής γιατρού και η διανομή του φαρμάκου στον ασθενή όπως ορίζεται από την φάση του ePrescription. Από την στιγμή που το φάρμακο παραληφθεί από τον πελάτη, ο διανομέας πρέπει να αναφέρει τις πληροφορίες σχετικά με την ενέργεια της διανομής χρησιμοποιώντας το λογισμικό ePrescription.

Περίληψη Δεδομένων Ιατροφαρμακευτικής Περίθαλψης Ασθενούς – Patient Summary

Το ΠΔΙΠΑ, είναι μια πρωτυποποιημένη συλλογή από βασικά ιατρικά δεδομένα που περιλαμβάνει τις πιο σημαντικές ιατρικές λεπτομέρειες ώστε να διασφαλιστεί η ασφαλής ιατροφαρμακευτική περίθαλψη. Αυτή η «περιληπτική» έκδοση των δεδομένων του ασθενή δίνει στους γιατρούς τις απαραίτητες πληροφορίες ώστε να μπορέσουν να παρέχουν την απαραίτητη ιατροφαρμακευτική περίθαλψη σε προγραμματισμένες ή απρογραμμάτιστες καταστάσεις(π.χ. ατύχημα). Παρόλο που αυτά τα δεδομένα έχουν ως σκοπό να βοηθήσουν τους γιατρούς σε απρογραμμάτιστες περιπτώσεις, μπορούν να χρησιμοποιηθούν και για προγραμματισμένη ιατροφαρμακευτική περίθαλψη (π.χ. σε περίπτωση μετακόμισης του πελάτη).

Αυτή η περίληψη των ιατρικών δεδομένων του πελάτη περιλαμβάνει τα ακόλουθα δεδομένα.

- Γενικές πληροφορίες για τον ασθενή(π.χ. όνομα, επίθετο, φύλλο)

- Μια περίληψη που αποτελείται από τα πιο σημαντικά κλινικά δεδομένα(π.χ. αλλεργίες, εμφυτεύματα, προβλήματα υγείας, σημαντικές εγχειρίσεις κατά την διάρκειας των τελευταίων 6 μηνών)
- Μια λίστα με τα φάρμακα που παίρνει την παρούσα φάση ο ασθενής
- Πληροφορίες σχετικά με το ίδιο το ΠΔΙΔΑ(π.χ. από ποιον δημιουργήθηκε και ενημερώθηκε)

Οι παραπάνω υπηρεσίες είναι αυτές που δοκιμάστηκαν κατά την πιλοτική λειτουργεία. Πέρα από αυτές τις δύο, υπάρχουν και άλλες υπηρεσίες που αναλύθηκαν και παρουσιάζονται παρακάτω. Οι υπηρεσίες με πράσινη επικεφαλίδα θωρήθηκαν εφικτές για δοκιμή και προετοιμάζονται από τις χώρες μέλη για πιλοτική λειτουργεία ενώ με κόκκινη αυτές που θεωρήθηκαν ανέφικτες για δοκιμή και έτσι δεν συμπεριλήφθηκαν.

Πρόσβαση σε Ιατρικά Δεδομένα από τον Ασθενή

Θα παρέχει στους ασθενείς πρόσβαση στα είδη υπάρχοντα ΠΔΙΔΑ τους, με ή χωρίς την παρουσία γιατρού είτε σε σημείο που παρέχεται ιατροφαρμακευτική περίθαλψη, είτε αλλού.

Έντυπο φαρμακευτικής επισκόπησης (ΕΦΕ) – Medication Realted Overview(MRO)

Το έντυπο φαρμακευτικής επισκόπησης περιέχει πληροφορίες που μπορεί να χρειαστούν κατά τη διάρκεια της συνταγογράφησης και της διανομής των φαρμάκων στον ασθενή σε μια ξένη χώρα.

Άλλες χρήσιμες πληροφορίες που μπορεί να περιέχει η επέκταση του ΕΦΕ είναι σχετικά με αλλεργίες και δυσανεξίες.

Έκθεση ιατροφαρμακευτικής περίθαλψης(ΕΙΠ) – Healthcare Encounter Report(HCER)

Η συγκεκριμένη υπηρεσία αποσκοπεί στην ενημέρωση της χώρας στην οποία είναι ασφαλισμένος ο ασθενής αφότου η ΠΔΙΔΑ του έχει αιτηθεί λόγω κάποιου γεγονότος ιατροφαρμακευτικής περίθαλψης που έλαβε μέρος στην ξένη χώρα που επισκέφτηκε. Η πληροφόρηση της χώρας στην οποία είναι ασφαλισμένος ο ασθενής, γίνεται φυσικά με τη συγκατάθεση του ίδιου.

Ενσωμάτωση της Ευρωπαϊκής Κάρτας Ασφάλισης Υγείας(ΕΕΚΑ) - Integration of the European Health Insurance Card(IEHIC)

Στην δεύτερη φάση του, το epSOS ανέλυσε και αξιολόγησε αν και πως πρέπει να χρησιμοποιηθούν τα ΕΕΚΑ δεδομένα όχι μόνο για υποβολή αιτήσεων αποζημίωσης αλλά και για αναγνώριση του ίδιου του ασθενούς, ωφελώντας έτσι τους χρήστες απλοποιώντας έτσι πολλές διαδικασίες.

Ενσωμάτωση του αριθμού 112

Το 1991, η Ευρωπαϊκή Ένωση καθιέρωσε το 112 ως παγκόσμιο αριθμό έκτακτης ανάγκης για όλα τα κράτη μέλη της. Αυτός ο αριθμός παρέχει εικοσιτετράωρη πρόσβαση σε υπηρεσίες έκτακτης ανάγκης σε όλη την Ευρώπη και τις χώρες που ανήκουν στην Ευρωπαϊκή Οικονομική ζώνη. Πολλές γειτονικές χώρες έχουν ή θα υιοθετήσουν την «υπηρεσία 112» όπως η Ουκρανία, η Τουρκία κ.α.

Το epSOS αξιολόγησε πως το 112 θα μπορούσε να ενσωματωθεί και αυτό στις είδη υπάρχουσες υπηρεσίες που προσφέρει το epSOS. Ο στόχος είναι να επιτρέψει στις Ευρωπαϊκές υπηρεσίες έκτακτης ανάγκης και πιο συγκεκριμένα σε αυτές που σχετίζονται με την Υγεία να έχουν πρόσβαση σε δεδομένα του ΠΔΙΠΑ νόμιμα και με ασφάλεια για την βελτίωση της ποιότητας ιατροφαρμακευτικής περίθαλψης.

2.1.1. Τεχνικό Υπόβαθρο

Η αρχιτεκτονική epSOS βασίζεται στην υπηρεσιοστραφή αρχιτεκτονική. Οι υπηρεσίες που παρέχονται από το epSOS έχουν υλοποιηθεί ως Web Services και περιγράφονται από τη γλώσσα WSDL(Web Services Description Language)[9]. Η επικοινωνία μεταξύ παρόχου και πελάτη ξεκινάει πάντα από την πλευρά του πελάτη. Κάθε χώρα που συμμετέχει στο πρόγραμμα παρέχει αυτές τις υπηρεσίες μέσω ενός NCP που ενεργεί ως πάροχος υπηρεσίας σε άλλες χώρες που συμμετέχουν στο πρόγραμμα και ως «πύλη» για αυτούς που τις χρησιμοποιούν.

2.2. FHIR

Το FHIR είναι ένα πρότυπο που περιγράφει μοντέλα δεδομένων ή αλλιώς πόρους(resources) και συμπεριλαμβάνει ένα API που χρησιμοποιείται για την διαχείριση αυτών των πόρων αλλά και την ανταλλαγή ηλεκτρονικών ιατρικών φακέλων. Αυτό το πρότυπο δημιουργήθηκε από την HL7(Health Level Seven International)[10], οργανισμός που φτιάχνει τέτοια πρότυπα σχετικά με την Υγεία.

Το FHIR, βασίζεται σε προηγούμενα πρότυπα της HL7 όπως τις εκδόσεις HL7 2.x και HL7 3.x. Σε αντίθεση όμως με τους προκατόχους του είναι πολύ πιο εύκολο στην χρήση του καθώς χρησιμοποιεί ένα σύνολο καινούργιων τεχνολογιών που περιλαμβάνουν ένα HTTP REST[11] πρωτόκολλο, HTML[12], CSS[13] για καλύτερη εμπειρία από πλευράς χρήστη, μια επιλογή μεταξύ των δομών δεδομένων JSON και XML, το OAuth[14] για εξουσιοδότηση και ATOM[15] ως αναφορά την παρουσίαση αποτελεσμάτων.

Στόχος του FHIR, είναι να διευκολύνει την διαλειτουργικότητα μεταξύ παλαιότερων συστημάτων υγείας και να κάνει πιο εύκολο τον διαμοιρασμό ιατρικών δεδομένων σε παρόχους υπηρεσιών υγείας αλλά και σε μεμονωμένα άτομα, σε ένα σύνολο συσκευών όπως κινητά, υπολογιστές και tablet αλλά και να επιτρέψει σε προγραμματιστές να υλοποιήσουν ιατρικού τύπου εφαρμογές που μπορούν να ενσωματωθούν σε είδη υπάρχοντα συστήματα.

Το FHIR παρέχει μια διαφορετική προσέγγιση, «εγγραφοστραφή», που εκθέτει διακριτά στοιχεία δεδομένων ως υπηρεσίες. Παραδείγματος χάρη, βασικά κλινικά στοιχεία όπως ασθενείς,

διαγνωστικές αναφορές, συνταγογράφηση φαρμάκων κ.α. μπορούν να ανακτηθούν και να διαχειριστούν μέσω ξεχωριστών URL. Το FHIR, υποστηρίχθηκε από την American Medical Informatics Association και από εταιρίες όπως η Cerner που εκτίμησαν το πόσο ανοιχτό και επεκτάσιμο είναι.

2.2.1. Προτυποποίηση

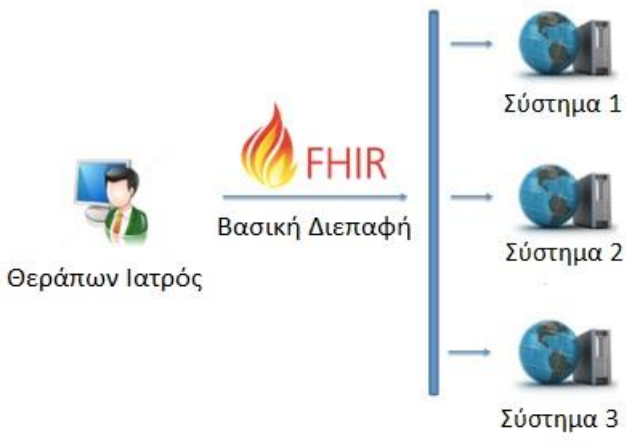
Το Φεβρουάριο του 2014 η HL7 εξέδωσε την πρώτη έκδοση του FHIR, DSTU[16] 1. Αυτή τη στιγμή το FHIR βρίσκεται στην έκδοση DSTU 2.1 και το 2017 αναμένεται το ολοκληρωμένο πρότυπο.

2.2.2. Υλοποιήσεις

Αυτή τη στιγμή υπάρχουν ανοιχτού κώδικα υλοποιήσεις πάνω στο FHIR που περιλαμβάνουν δομές δεδομένων, διακομιστές και άλλα εργαλεία σε μια ποικιλία από γλώσσες. Στην JAVA η υλοποίηση ανοιχτού κώδικα που υπάρχει ονομάζεται HAPI-FHIR[17].

Λόγω του ότι το FHIR έχει βασιστεί πάνω στο HL7 και χρησιμοποιεί HTTP πρωτόκολλο, τα μηνύματα που ανταλλάσσονται μπορούν να διαβαστούν από διάφορα συστήματα για συλλογή δεδομένων σε πραγματικό χρόνο. Με αυτή τη λογική, οι οργανισμοί υγείας θα μπορούσαν να μαζεύουν δεδομένα που περιέχονται σε μηνύματα FHIR, καθώς αυτά μηνύματα περνούν μέσα από το δίκτυο. Αυτά τα δεδομένα μπορούν να αποθηκευτούν σε κάποια κεντρική βάση και στη συνέχεια να συσχετιστούν με άλλες πληροφορίες για την επίτευξη κάποιου ή κάποιων αποτελεσμάτων.

Φυσικά για να λειτουργήσουν όλα αυτά και να υπάρξει όφελος από τη χρήση του FHIR θα πρέπει να χρησιμοποιηθεί κάποιο αναγνωριστικό ασθενή ώστε να γίνει η σύνδεση του ασθενή στα διάφορα συστήματα.



Σχήμα 1: Βασική Λειτουργία FHIR

2.3. Hapi – FHIR

Το Hapi-FHIR είναι ένας ανοιχτού κώδικα, αντικειμενοστραφές HL7 2.x επεξεργαστής κώδικα (parser) για την γλώσσα προγραμματισμού Java. Το Hapi-FHIR δεν έχει καμία συσχέτιση με την HL7. Το λογισμικό αναπτύχθηκε από το University Health Network στο Τορόντο του Καναδά και συμμορφώνεται απλά με τα πρότυπά της HL7. Τη στιγμή εκπόνησης της εργασίας η έκδοση του Hapi-FHIR είναι η 2.2.

2.4. Microsoft HealthVault

Το HealthVault[18] είναι ένα web based σύστημα της Microsoft που αποθηκεύει και διατηρεί πληροφορίες σχετικά με την υγεία. Ξεκίνησε να λειτουργεί τον Οκτώβριο του 2007 και απευθύνεται τόσο σε απλούς πολίτες όσο και σε επαγγελματίες που ανήκουν στον χώρο της υγείας. Τον Ιούνιο του 2010 επέκτεινε τις υπηρεσίες του και στο Ηνωμένο Βασίλειο.

2.4.1. Γενική Ιδέα

Το HealthVault, όπως αναφέρθηκε και πιο πάνω, διατηρεί πληροφορίες σχετικά με την υγεία του ασθενούς. Η πρόσβαση σε κάποιο μητρώο υγείας ασθενή γίνεται μέσω ενός λογαριασμού HealthVault, ο οποίος ταυτόχρονα μπορεί να είναι εξουσιοδοτημένος για να έχει πρόσβαση σε μητρώα πολλών ατόμων. Παραδείγματος χάρη μπορεί μια μητέρα να θέλει να διαχειρίζεται τα μητρώα υγείας των παιδιών της ή ένας πατέρας να έχει πρόσβαση στο μητρώο υγείας του γιού του ώστε να τον βοηθήσει να αντιμετωπίσει πιθανά θέματα υγείας. Πρόσβαση στο λογαριασμό HealthVault μπορεί να υπάρξει μέσω κάποιου Windows Live ID[19], Facebook ή κάποιων παρόχων OpenID[20].

2.4.2. Εξουσιοδότηση

Κάποιο άτομο μπορεί να αλληλοεπιδράσει με το μητρώο υγείας που έχει στο HealthVault είτε μέσω του κεντρικού ιστότοπου, είτε μέσω κάποιας εφαρμογής που είναι συνδεδεμένη με την πλατφόρμα HealthVault. Όταν ο χρήστης χρησιμοποιήσει για πρώτη φορά την εφαρμογή, θα του ζητηθεί να εξουσιοδοτήσει την εφαρμογή να έχει πρόσβαση σε κάποια δεδομένα που μόνο αυτή θα μπορεί να χρησιμοποιήσει. Επίσης ο χρήστης μπορεί να διαμοιραστεί τον λογαριασμό του με κάποιον άλλο όπως παραδείγματος χάρη έναν γιατρό, τη γυναίκα, έναν γονιό κ.α.

2.4.3. Συσκευές

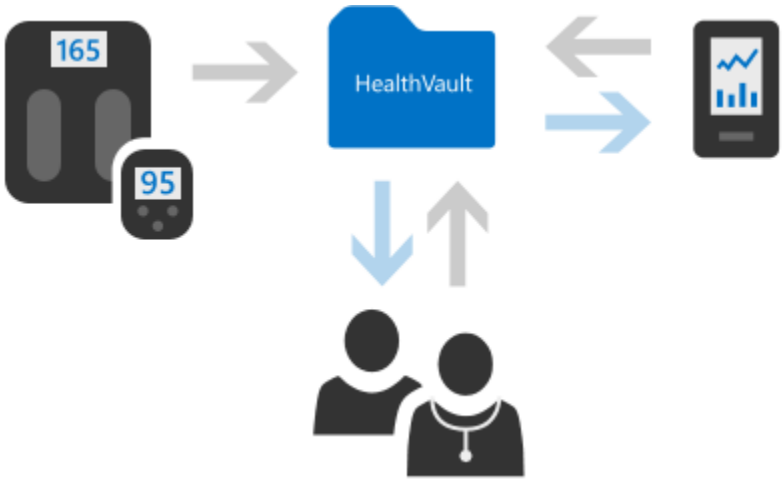
Το Κέντρο Σύνδεσης HealthVault(HealthVault Connection Center), επιτρέπει τη μεταφορά δεδομένων από συσκευές όπως ρολόγια που μετρούν τον παλμό της καρδιάς, μετρητές πίεσης αίματος κ.α. στο HealthVault. Μπορεί επίσης να χρησιμοποιηθεί ώστε να ψάξει και να βρει οδηγούς(drivers) για τις ιατρικές συσκευές.

2.4.4. Διαλειτουργικότητα

Το HealthVault υποστηρίζει έναν αριθμό από μορφές ανταλλαγής δεδομένων συμπεριλαμβανομένων προτύπων βιομηχανίας όπως το Continuity of Care Document[21] και Continuity of Care Record[22]. Αυτή η υποστήριξη σε τέτοια πρότυπα της βιομηχανίας κάνουν εύκολη την ολοκλήρωση του HealthVault με είδη υπάρχοντα συστήματα που διαχειρίζονται ηλεκτρονικούς ιατρικούς φακέλους.

2.4.5. Ανταγωνιστές

Οι κύριοι ανταγωνιστές του HealthVault είναι οι Dossia, World Medical Card, Apples HealthKit and Health και άλλοι. Η Google ανακοίνωσε τον Ιανουάριο του 2012 ότι η πλατφόρμα της, Google Health, θα έβγαινε από την αγορά και παρότρυνε τους χρήστες να μεταφέρουν τα δεδομένα τους στην πλατφόρμα HealthVault μέσα στον επόμενο χρόνο.



Σχήμα 2: Βασική Λειτουργία HealthVault

2.5. Android

Το android είναι ένα λειτουργικό σύστημα, που την τρέχουσα χρονική περίοδο αναπτύσσεται από την Google, βασίζεται στον πυρήνα Linux και έχει σχεδιαστεί για φορητές συσκευές με οθόνη

αφής όπως κινητά και ταμπλέτες. Επίσης ποικίλες εκδόσεις του android χρησιμοποιούνται σε φορητούς υπολογιστές, κονσόλες, βιντεοκάμερες και άλλες ηλεκτρονικές συσκευές. Από το 2015 το λειτουργικό android χρησιμοποιείται σε περισσότερες συσκευές από κάθε άλλο λειτουργικό στον κόσμο όπως iOS και Windows.

Το android αναπτύχθηκε αρχικά από την Android Inc. την οποία η Google εξαγόρασε το 2005. Το android αποκαλύφθηκε το 2007 σε συνδυασμό με την ίδρυση της Open Handset Alliance, μια κοινοπραξία από εταιρίες λογισμικού, hardware και τηλεπικοινωνιών αφοσιωμένες στην προώθηση «ανοιχτών» προτύπων για φορητές συσκευές. Από τον Ιούλιο του 2015 το Google Play, υπηρεσία παροχής εφαρμογών και υπηρεσιών, είχε πουλήσει πάνω από ένα εκατομμύριο εφαρμογές και πάνω από πενήντα εκατομμύρια εφαρμογές είχαν κατέβει σε συσκευές σε όλο τον κόσμο. Μια έρευνα το 2013 έδειξε ότι το 71% των προγραμματιστών αναπτύσσουν εφαρμογές για android, ενώ ένα 40% βρήκαν ότι άτομα που δουλεύουν ως προγραμματιστές βλέπουν την πλατφόρμα android ως προτεραιότητα για τις εφαρμογές τους.

Ο πηγαίος κώδικας κυκλοφορεί από την Google κάτω από άδεια ανοιχτού λογισμικού. Παρόλα αυτά οι περισσότερες συσκευές κυκλοφορούν με ένα συνδυασμό ανοιχτού κώδικα και ιδιόκτητου λογισμικού για την πρόσβαση σε υπηρεσίες της Google.

Το εν λόγω λειτουργικό χρησιμοποιείται από πάρα πολλές εταιρίες τεχνολογίας που χρειάζονται ένα έτοιμο, φτηνό και προσαρμόσιμο λειτουργικό για τις συσκευές τους.

Το android αναπτύσσεται ιδιωτικά από την Google μέχρι οι αλλαγές και ενημερώσεις να είναι έτοιμες για την κυκλοφορία όπου ο πηγαίος κώδικας γίνεται διαθέσιμος στο κοινό.

Σχήμα 3: Android Runtime

2.6. MongoDB

Η MongoDB[23] είναι μια «εγγραφοστραφής» βάση δεδομένων, που δουλεύει σε παραπάνω από μια υπολογιστικές πλατφόρμες. Σε αντίθεση με τις κλασικές σχεσιακές βάσεις δεδομένων, η MongoDB είναι κατηγοριοποιημένη ως NoSQL[24] και χρησιμοποιεί JSON έγγραφα (BSON για τη MongoDB) για την αποθήκευση δεδομένων, κάνοντας ορισμένες εφαρμογές να τρέχουν πιο εύκολα και πιο γρήγορα. Η MongoDB έχει κυκλοφορήσει με ένα συνδυασμό αδειών GNU Affero General Public License και Apache License και είναι ανοιχτού λογισμικού.

Πρώτα αναπτύχθηκε από την εταιρία MongoDB Inc. τον Οκτώβριο του 2007 σαν κομμάτι ενός προϊόντος PAAS(Platform As A Service)[25]. Στη συνέχεια η εταιρία άλλαξε πολιτική και το 2009 στράφηκε προς την ανάπτυξη ενός μοντέλου ανοιχτού λογισμικού με τη MongoDB να παρέχει υποστήριξη και άλλες εμπορικές υπηρεσίες. Από τότε, η MongoDB έχει χρησιμοποιηθεί ως λογισμικό στην πλευρά του διακομιστή από έναν μεγάλο αριθμό ιστοτόπων και υπηρεσιών συμπεριλαμβανομένου του Craiglist, EBay, Foursquare κ.α. Από τον Ιούλιο του 2015, είναι το τέταρτο πιο δημοφιλές σύστημα διαχείρισης δεδομένων και επίσης η πιο δημοφιλής βάση για την αποθήκευση εγγράφων.

2.6.1. Κύρια Χαρακτηριστικά

- Είναι «εγγραφοστραφής»

Αντί να παίρνει ένα αντικείμενο και να το σπάει σε πολλαπλές σχεσιακές δομές, η MongoDB μπορεί να το αποθηκεύσει σε ένα ελάχιστο αριθμό εγγράφων. Παραδείγματος χάριν αντί να γίνει η αποθήκευση των πληροφοριών ενός τίτλου βιβλίου και το όνομα συγγραφέα σε δύο διαφορετικές σχεσιακές δομές, μπορούν να αποθηκευτούν όλα σε ένα μόνο έγγραφο με το όνομα Βιβλίο.

- Ερωτήματα στη βάση

Η MongoDB υποστηρίζει ερωτήματα στη βάση δεδομένων σύμφωνα με κάποιο πεδίο, αναζητήσεις με κάποιο regular expression αλλά και σύμφωνα με κάποια ανώτερη ή κατώτερη τιμή.

- Ευρετήρια

Σε κάθε πεδίο στην MongoDB μπορεί να προστεθεί κάποιο ευρετήριο για πιο γρήγορη αναζήτηση (Τα ευρετήρια είναι εννοιολογικά ίδια με αυτά τον σχεσιακών βάσεων δεδομένων). Δευτερεύοντα ευρετήρια είναι επίσης διαθέσιμα.

- Αντίγραφα δεδομένων

Η MongoDB προσφέρει υψηλή διαθεσιμότητα με συλλογές αντίγραφων δεδομένων. Μια τέτοια συλλογή αποτελείται από ένα ή περισσότερα αντίγραφα δεδομένων. Κάθε αντίγραφο μπορεί ανά πάσα στιγμή να παίξει το ρόλο του κυρίως ή του δευτερεύοντος αντίγραφου. Σε κάθε κυρίως αντίγραφο μπορούν εξ αρχής να εκτελεστούν λειτουργίες διαβάσματος και επεξεργασίας. Επίσης, σε περίπτωση που αποτύχει κάποιο κυρίως αντίγραφο, γίνεται μια διαδικασία εκλογής και αποφασίζεται ποιο από τα δευτερεύοντα αντίγραφα θα πάρει τη θέση του κυρίως αντίγραφου. Να τονιστεί ότι διαδικασίες διαβάσματος μπορούν να γίνουν και σε δευτερεύοντα αντίγραφα.

- Εξισορρόπηση φορτίου

Η MongoDB κλιμακώνεται οριζοντίως χρησιμοποιώντας την διαδικασία sharding. Sharding είναι η διαδικασία κατά την οποία γίνεται αποθήκευση των δεδομένων σε πολλαπλά μηχανήματα. Ο

χρήστης επιλέγει ένα κλειδί shard, το οποίο καθορίζει τον τρόπο με τον οποίο θα κατανεμηθούν οι συλλογές των δεδομένων. Τα δεδομένα χωρίζονται σύμφωνα με το κλειδί shard και διανέμονται σε πολλαπλά shards(συλλογή αντιγράφων που περιέχει ένα υποσύνολο δεδομένων). Η MongoDB μπορεί να τρέχει πάνω από πολλαπλούς διακομιστές, εξισορροπώντας τον φόρτο εργασίας και/ή κρατώντας αντίγραφα των δεδομένων σε περίπτωση δυσλειτουργίας του hardware. Τέλος σε μια βάση που είδη τρέχει είναι πολύ εύκολη η πρόσθεση νέων μηχανημάτων.

- Αποθήκευση δεδομένων

Η MongoDB μπορεί να λειτουργήσει και ως file system, χρησιμοποιώντας τα πλεονεκτήματα της εξισορρόπησης φορτίου και της αντιγραφής δεδομένων που αναλύθηκαν πιο πάνω σε πολλαπλά μηχανήματα για την αποθήκευση δεδομένων.

Αυτή η λειτουργία λέγεται Grid File System και συμπεριλαμβάνεται στου οδηγούς(drivers) της MongoDB. Η MongoDB παρέχει λειτουργίες για διαχείριση αρχείων και περιεχομένου στους προγραμματιστές. Το Grid FS, αντί να αποθηκεύει τα δεδομένα σε ένα έγγραφο, χωρίζει το αρχείο σε μέρη και αποθηκεύει κάθε ένα από αυτά τα μέρη ως διαφορετικό έγγραφο

Σε ένα σύστημα MongoDB με πολλά μηχανήματα, τα αρχεία μπορούν να διανέμονται και να αντιγράφονται σε πολλά μηχανήματα δημιουργώντας έτσι ένα σύστημα που μπορεί να διαχειριστεί μεγάλο όγκο δεδομένων και μπορεί να χειριστεί τυχόν λάθη.

- Συσσωμάτωση(Aggregation)

Το MapReduce[26] μπορεί να χρησιμοποιηθεί για μαζική επεξεργασία δεδομένων και λειτουργίες συσσωμάτωσης. Το MapReduce καθιστά ικανό στο χρήστη στο να ανακτήσει δεδομένα που μόνο ένα ερώτημα της SQL[27] εντολής GROUP BY θα μπορούσε να επιστρέψει.

- Χρήση JavaScript στην πλευρά του διακομιστή

Η JavaScript μπορεί να χρησιμοποιηθεί σε ερωτήματα στην βάση δεδομένων, σε λειτουργίες συσσωμάτωσης(όπως το MapReduce) και να σταλθεί κατευθείαν στη βάση ώστε να εκτελεστεί.

- Sharding

Sharding είναι η διαδικασία αποθήκευσης δεδομένων σε πάνω από ένα μηχανήματα και είναι η προσέγγιση της Mongo στο να αντιμετωπίζει θέματα σχετικά με τον όγκο των πληροφοριών. Όσο το μέγεθος των δεδομένων μεγαλώνει, ένα μόνο μηχάνημα μπορεί να μην είναι αρκετό για να αποθηκεύσει τα δεδομένα ούτε να παρέχει διαδικασίες εγγραφής και ανάγνωσης σε λογικό χρονικό περιθώριο. Το Sharding λύνει το πρόβλημα της οριζόντιας κλιμάκωσης.

2.6.2. Αρχιτεκτονική

Υποστήριξη γλωσσών

Η MongoDB περιέχει οδηγούς(drivers) για μια ποικιλία γλωσσών προγραμματισμού και περιβάλλοντα προγραμματισμού. Επίσης υπάρχει και ένας μεγάλος αριθμός από «μη επίσημους» οδηγούς(drivers) για πολλές γλώσσες προγραμματισμού.

Διαχείριση και Γραφικά Περιβάλλοντα

Η διαχείριση της βάσης γίνεται μέσω εργαλείων γραμμής εντολών όπως το mongo shell, επειδή η MongoDB δεν περιλαμβάνει γραφικό περιβάλλον για τη διαχείρισή της. Κυκλοφορούν όμως λογισμικά με γραφικό περιβάλλον για την διαχείριση και προβολή των δεδομένων.

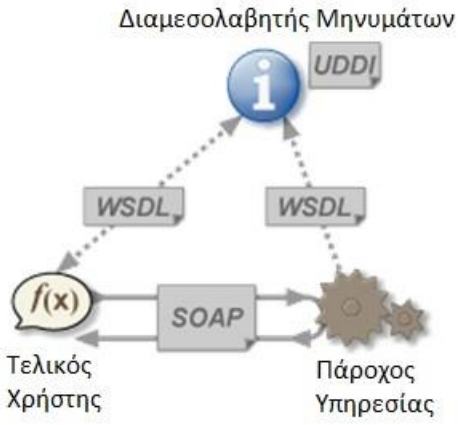
2.7. Web Service

Web Service είναι μια υπηρεσία που προσφέρεται από μια ηλεκτρονική συσκευή σε μια άλλη ηλεκτρονική συσκευή, οι οποίες επικοινωνούν μεταξύ τους μέσω Παγκόσμιου Ιστού. Σε ένα web service, τεχνολογίες ιστού όπως το πρωτόκολλο HTTP, σχεδιασμένες αρχικά για επικοινωνία μεταξύ ανθρώπου-μηχανής, χρησιμοποιούνται για την μεταφορά αναγνώσιμων από μηχανές μορφών αρχείου όπως XML και JSON. Στην πράξη, ένα web service παρέχει μια αντικειμενοστραφή διεπαφή βασισμένη στον Παγκόσμιο Ιστό σε ένα διακομιστή, χρησιμοποιούμενη παραδείγματος χάριν από κάποιον άλλο διακομιστή ιστού ή κινητή εφαρμογή που παρέχει μια διεπαφή στον χρήστη/δέκτη. Μια άλλη εφαρμογή που παρέχεται στον χρήστη μπορεί να είναι ένα mashup, δηλαδή ένας διακομιστής ιστού που χρησιμοποιήσει πολλά web

services από διαφορετικές μηχανές και παρουσιάζει το τελικό περιεχόμενο σε μια μόνο διεπαφή χρήστη.

Το W3C[28] ορίζει ως web service ένα λογισμικό σχεδιασμένο να υποστηρίζει τη διαλειτουργικότητα μεταξύ μηχανών πάνω από το δίκτυο.

Ο όρος Web Services περιγράφει ένα πρωτυποποιημένο τρόπο για την ολοκλήρωση εφαρμογών βασισμένες στον ιστό χρησιμοποιώντας τα πρότυπα XML, SOAP[29], WSDL και UDDI[30]. Η XML χρησιμοποιείται για την «ετικετοποίηση» των δεδομένων, το SOAP για τη μεταφορά, το WSDL για την περιγραφή των υπηρεσιών και το UDDI καταγράφει ποιες υπηρεσίες είναι διαθέσιμες.



Σχήμα 4: Λειτουργία SOAP Web Service

Ένα web service είναι μια μέθοδος επικοινωνίας μεταξύ δύο ηλεκτρονικών μηχανών πάνω από το δίκτυο. Είναι μια λειτουργία λογισμικού που παρέχεται μέσω μιας διεύθυνσης δικτύου, με την υπηρεσία να είναι πάντα έτοιμη να χρησιμοποιηθεί από κάποια μηχανή.

Πολλοί οργανισμοί χρησιμοποιούν πολλαπλά συστήματα για διαχείριση. Αυτά τα συστήματα χρειάζονται συχνά να ανταλλάξουν δεδομένα μεταξύ τους και ένα web service είναι μια μέθοδος επικοινωνίας που επιτρέπει σε δύο συστήματα να ανταλλάσσουν δεδομένα μέσω ιστού. Το σύστημα που κάνει το αίτημα για δεδομένα ονομάζεται (Αιτών Υπηρεσίας)Service Requester και

το λογισμικό που διαχειρίζεται το αίτημα και παρέχει τα δεδομένα ονομάζεται (Πάροχος Υπηρεσίας) Service Provider.

Διάφορα λογισμικά μπορούν να χρησιμοποιήσουν διαφορετικές γλώσσες προγραμματισμού και όπως είναι λογικό γίνεται αναγκαία η χρησιμοποίηση μιας μεθόδου για την ανταλλαγή δεδομένων που δεν στηρίζεται πάνω σε μια συγκεκριμένη γλώσσα προγραμματισμού. Οι περισσότερες γλώσσες προγραμματισμού όμως μπορούν να διαβάσουν και να ερμηνεύσουν ετικέτες XML. Έτσι λοιπόν η XML χρησιμοποιείται από τα web services για ανταλλαγή δεδομένων.

Χρειάστηκαν να οριστούν κανόνες επικοινωνίας μεταξύ διαφορετικών συστημάτων, σχετικά με:

- Πώς ένα σύστημα μπορεί να κάνει αίτηση για δεδομένα από ένα άλλο σύστημα
- Ποιες παράμετροι χρειάζονται να περαστούν κατά την αίτηση των δεδομένων
- Ποια θα ήταν η δομή των παραγόμενων δεδομένων
- Τι μηνύματα λάθους να απεικονιστούν σε περίπτωση που κάποιος κανόνας επικοινωνίας δεν τηρηθεί έτσι ώστε η διαχείριση των λαθών να γίνει πιο εύκολη.

Όλοι αυτοί οι κανόνες επικοινωνίας ορίζονται σε ένα αρχείο που ονομάζεται WSDL και έχει επέκταση .wsdl.

Η διαδικασία αιτήματος δεδομένων από έναν χρήστη σε κάποιο πάροχο έχει ως εξής. Ο Πάροχος Υπηρεσιών στέλνει το αρχείο WSDL στο UDDI. Αυτός που κάνει αίτηση για την υπηρεσία, επικοινωνεί με το UDDI για να μάθει ποιος είναι ο πάροχος των δεδομένων που χρειάζεται, και στη συνέχεια επικοινωνεί με τον Πάροχο Υπηρεσιών χρησιμοποιώντας το πρωτόκολλο SOAP. Ο Πάροχος Υπηρεσιών επικυρώνει το αίτημα υπηρεσίας και στέλνει τα δεδομένα σε μορφή XML, χρησιμοποιώντας SOAP πρωτόκολλο. Αυτό το XML αρχείο ξαναεπικυρώνεται χρησιμοποιώντας ένα XSD αρχείο.

2.8. REST

Στην υπολογιστική, το REST είναι η αρχιτεκτονική λογισμικού του Παγκόσμιου Ιστού. Το συντονισμένο σύνολο περιορισμών του πρωτοκόλλου REST, εφαρμόζεται στον σχεδιασμό των μερών ενός κατανεμημένου συστήματος υπερμέσων που μπορεί να οδηγήσει σε υψηλότερη απόδοση και σε μια πιο εύκολα διαχειρίσιμη αρχιτεκτονική λογισμικού.

Όταν ένα σύστημα συμμορφώνεται με τους περιορισμούς του REST τότε μπορεί να λέγεται RESTful. Τα RESTful συστήματα, συνήθως αλλά όχι πάντα, επικοινωνούν μέσω του πρωτοκόλλου HTTP με τα ρήματα GET, POST, PUT, DELETE κ.τ.λ. που οι φυλλομετρητές χρησιμοποιούν για να ανακτήσουν ιστοσελίδες και να στείλουν δεδομένα σε απομακρυσμένους διακομιστές. Τα REST συστήματα αλληλοεπιδρούν με εξωτερικά συστήματα σαν πόρους που αναγνωρίζονται μέσω URIs. Παραδείγματος χάριν το URI people/tom, που μπορεί να χρησιμοποιηθεί μέσω HTTP ρημάτων όπως DELETE /people/tom.

Το REST ορίστηκε από τον Roy Thomas Fielding στη διδακτορική του εργασία το 2000 «Architectural Styles and the Design of Network-based Software Architectures». Ο Fielding ανέπτυξε την αρχιτεκτονική REST παράλληλα με το HTTP 1.1 του 1996-1999, βασιζόμενος στο είδη υπάρχον HTTP 1.0 του 1996.

2.8.1. Ιδιότητες Αρχιτεκτονικής

- **Απόδοση** – Η αλληλεπίδραση μεταξύ των επιμέρους στοιχείων μπορεί να παίξει καταλυτικό ρόλο την απόδοση που αντιλαμβάνεται ο χρήστης και στην αποδοτικότητα του δικτύου
- **Επεκτασιμότητα** για την υποστήριξη μεγάλου αριθμού στοιχείων και αλληλεπίδραση μεταξύ αυτών
- **Απλότητα** όσον αφορά τις διεπαφές
- **Τροποιησιμότητα** των εξαρτημάτων για να ταιριάξουν με τις μεταβαλλόμενες ανάγκες (ακόμα και όταν η εφαρμογή τρέχει)

- **Μεταφερσιμότητα** των εξαρτημάτων μετακινώντας τον κώδικα με τα δεδομένα
- **Ορατότητα** επικοινωνίας μεταξύ εξαρτημάτων από μέσα υπηρεσιών
- **Αξιοπιστία** είναι η ανοχή σε σφάλματα, σε περίπτωση σφαλμάτων που μπορεί να εμφανιστούν σε εξαρτήματα, συνδέσμους ή δεδομένα

2.8.2. Περιορισμοί Αρχιτεκτονικής

Οι αρχιτεκτονικές ιδιότητες του REST μπορούν να κατανοηθούν εφαρμόζοντας περιορισμούς αλληλεπίδρασης σε εξαρτήματα, συνδέσμους και δεδομένα. Κάποιος μπορεί να χαρακτηρίσει τις εφαρμογές που συμμορφώνονται με τους περιορισμούς του REST ως RESTful. Αν μια εφαρμογή δεν τηρεί κάποιον από τους περιορισμούς δεν μπορεί να χαρακτηρισθεί ως RESTful. Όταν ένα κατανεμημένο σύστημα υπερμέσων συμμορφώνεται με το αρχιτεκτονικό στυλ του REST, του επιτρέπεται να έχει τις επιθυμητές λειτουργίες που περιεγράφηκαν παραπάνω όπως απόδοση, επεκτασιμότητα, απλότητα, τροποποιησιμότητα, ορατότητα, μεταφερσιμότητα και αξιοπιστία.

Οι περιορισμοί του REST είναι:

- Πελάτη-Διακομιστής (Client - Server)

Μια ομοιόμορφη διεπαφή ξεχωρίζει τους πελάτες από τους διακομιστές. Αυτός ο διαχωρισμός σημαίνει ότι για παράδειγμα οι πελάτες δεν ασχολούνται με το κομμάτι της αποθήκευσης δεδομένων, το οποίο παραμένει εσωτερικά στον διακομιστή, ώστε η μεταφερσιμότητα του κώδικα στην πλευρά του πελάτη να είναι όσο το δυνατόν καλύτερη. Οι διακομιστές από την πλευρά τους, δεν ασχολούνται με το κομμάτι της διεπαφής που αλληλοεπιδρά ο χρήστης, ούτε με την κατάστασή του, ώστε να παραμένουν όσο το δυνατόν πιο απλοί και επεκτάσιμοι. Οι διακομιστές και οι πελάτες μπορούν ανά πάσα στιγμή να αντικατασταθούν και να αναπτυχθούν ξεχωριστά, όσο η αλληλεπίδραση μεταξύ τους δεν αλλοιώνεται.

- Χωρίς μνήμη (Stateless)

Η επικοινωνία μεταξύ πελάτη και διακομιστή δεν περιορίζεται επιπλέον από κάποιο πλαίσιο του πελάτη αποθηκευμένο στην πλευρά του διακομιστή. Κάθε αίτημα του πελάτη περιέχει όλες τις

απαραίτητες πληροφορίες για να εκπληρωθεί το αίτημα του και μνήμη για μια συνεδρία υπάρχει μόνο στην πλευρά του πελάτη. Η κατάσταση μιας συνεδρίας μπορεί να μεταφερθεί από τον διακομιστή σε κάποια άλλη υπηρεσία όπως σε μια βάση για να διατηρήσει μια κατάσταση για ένα χρονικό διάστημα και να επιτρέψει την αυθεντικοποίηση. Ο πελάτης ξεκινάει να στέλνει αιτήματα όταν είναι έτοιμος να μεταβεί σε μια καινούργια κατάσταση. Όσο ένα ή περισσότερα αιτήματα είναι σε εκκρεμότητα, ο πελάτης θεωρείται ότι είναι σε μια μετάβαση προς μια νέα κατάσταση. Η αναπαράσταση κάθε κατάσταση μιας εφαρμογής περιέχει συνδέσμους που μπορούν να χρησιμοποιηθούν την επόμενη που ένας πελάτης επιλέγει να ξεκινήσει μια μετάβαση προς μια καινούργια κατάσταση.

- Αποθήκευση στην κρυφή/προσωρινή μνήμη (Cacheable)

Όπως και στον Παγκόσμιο Ιστό, οι πελάτες και οι διαμεσολαβητές μπορούν να αποθηκεύσουν στην προσωρινή τους μνήμη τις απαντήσεις που έχουν λάβει από τον διακομιστή. Οι απαντήσεις αυτές λοιπόν πρέπει να καθορίσουν τον εαυτό τους ως αποθηκεύσιμες στην προσωρινή μνήμη ή όχι για να αποτρέψουν πελάτες να χρησιμοποιήσουν παλιά ή ακατάλληλα δεδομένα σε άλλα αιτήματα. Η καλή διαχείριση της αποθήκευσης στην προσωρινή μνήμη εξαλείφει μερικώς ή πλήρως κάποιες αλληλεπιδράσεις διακομιστή-πελάτη, βελτιώνοντας έτσι την απόδοση και την επεκτασιμότητα.

- Πολυεπίπεδο σύστημα (Multilayered System)

Ένας χρήστης δεν μπορεί σε αρχική φάση να ξέρει αν είναι συνδεδεμένος κατευθείαν στον διακομιστή ή σε κάποιον ενδιάμεσο διακομιστή. Οι ενδιάμεσοι διακομιστές μπορούν να βελτιώσουν την επεκτασιμότητα του συστήματος ενεργοποιώντας την εξισορρόπηση φορτίου και παρέχοντας κοινόχρηστες κοινές μνήμες. Μπορούν επίσης να επιβάλουν την εφαρμογή πολιτικών ασφαλείας.

- Κώδικας σε ζήτηση – Προαιρετικό (Code On Demand)

Οι διακομιστές μπορούν προσωρινά να επεκτείνουν ή να τροποποιήσουν την λειτουργικότητα ενός πελάτη μεταφέροντας εκτελέσιμο κώδικα. Παραδείγματα μπορούν να περιλαμβάνουν Java Applets και κομμάτια κώδικα στην πλευρά του χρήστη σε Javascript.

- Ομοιόμορφη διεπαφή (Uniform Interface)

Ο περιορισμός της ομοιόμορφης διεπαφής είναι βασικός στην σχεδίαση ενός REST Web Service. Η ομοιόμορφη διεπαφή απλοποιεί και «αποσυνδέει» την εφαρμογή επιτρέποντας σε όλα τα μέρη να αναπτυχθούν ανεξάρτητα. Οι τέσσερις περιορισμοί της ομοιόμορφης διεπαφής είναι:

- Αναγνώριση των πόρων

Ο κάθε πόρος στα συστήματα βασιζόμενα στο REST αναγνωρίζεται σε αιτήματα, παραδείγματος χάριν χρησιμοποιώντας URIs[31]. Οι πόροι αυτοί εννοιολογικά είναι διαφορετικοί από τις αναπαραστάσεις που στέλνονται στον πελάτη. Παραδείγματος χάριν ο διακομιστής μπορεί να στείλει δεδομένα από την βάση δεδομένων του ως HTML, XML ή JSON, κανένα από τα οποία δεν είναι η εσωτερική αναπαράσταση της βάσης.

- Επεξεργασία των πόρων μέσα από αυτές τις αναπαραστάσεις

Όταν ένα πελάτης έχει την αναπαράσταση ενός πόρου, συμπεριλαμβανομένου των μεταδεδομένων που είναι προσαρτημένα, έχει αρκετές πληροφορίες ώστε να τροποποιήσει ή να διαγράψει τον πόρο.

- Αυτό-περιγραφόμενα μηνύματα

Κάθε μήνυμα περιλαμβάνει αρκετές πληροφορίες σχετικά με την επεξεργασία του ίδιου του μηνύματος. Για παράδειγμα, ποιος επεξεργαστής κώδικα (parser) πρέπει να επικαλεστεί καθορίζεται από τον τύπο Internet media type (πιο παλιά οριζόταν ως τύπος MIME[32]).

- (Hypermedia As The Engine Of Application State) HATEOAS[33]

Οι πελάτες μεταβαίνουν από μια κατάσταση σε μια άλλη μόνο μέσω ενεργειών που αναγνωρίζονται δυναμικά μέσα σε υπερμέσα από τον διακομιστή (π.χ. υπερσύνδεσμοι μέσα σε υπερκείμενο). Εκτός από μικρά σταθερά σημεία εισόδου στην εφαρμογή, ένας πελάτης δεν υποθέτει ότι κάποια συγκεκριμένη ενέργεια είναι διαθέσιμη για κάποιο συγκεκριμένο πόρο πέραν αυτών των ενεργειών που περιγράφονται από αναπαραστάσεις, που προηγουμένως έχουν παραληφθεί από τον διακομιστή.

2.9. RESTFUL APIs

Τα Web Service APIs που συμφωνούν με τους περιορισμούς της αρχιτεκτονικής REST ονομάζονται RESTful APIs. Τα RESTful APIs που χρησιμοποιούν το πρωτόκολλο HTTP καθορίζονται με τα εξής χαρακτηριστικά

- Ένα βασικό URI(π.χ. http://example.com/resources)
- Ένας τύπος δεδομένων(Internet Media Type) για τα δεδομένα. Συνήθως είναι JSON αλλά μπορεί να είναι και άλλοι τύποι όπως XML, Atom, microformats, application/vnd.collection+json, etc)
- Βασικές HTTP μέθοδοι(OPTIONS, GET, PUT, POST ή DELETE)
- Σύνδεσμοι υπερκειμένου για την αναφορά της κατάστασης
- Σύνδεσμοι υπερκειμένου για την αναφορά στους πόρους που βρίσκονται στους διακομιστές.

2.10. XML

Η XML είναι μια markup γλώσσα που καθορίζει ένα σύνολο κανόνων για την κωδικοποίηση εγγράφων σε μια τέτοια μορφή που μπορεί να είναι ταυτόχρονα αναγνώσιμη και από μηχανές και από ανθρώπους. Καθορίζεται από τις προδιαγραφές της W3C, XML 1.0 και από άλλες σχετικές προδιαγραφές, όλα τα οποία είναι ανοιχτά πρότυπα.

Ο σχεδιαστικός στόχος της XML είναι να δώσει έμφαση στην απλότητα, την γενικότητα και την ευχρηστία στον Παγκόσμιο Ιστό. Είναι μια κειμενική μορφή δεδομένων με ισχυρή υποστήριξη μέσω Unicode για διάφορες ανθρώπινες γλώσσες(Ελληνικά, Αγγλικά κ.τ.λ.). Παρόλο που ο σχεδιασμός της XML εστιάζει σε έγγραφα, χρησιμοποιείται ευρέως για την αναπαράσταση αυθαίρετων δομών δεδομένων όπως σε αυτές που χρησιμοποιούνται στα Web Services.

Πολλά σχηματικά συστήματα(schema systems) υπάρχουν για να βοηθούν στον ορισμό των γλωσσών βασισμένων στην XML, ενώ πολλά APIs έχουν αναπτυχθεί στην βοήθεια επεξεργασίας XML δεδομένων.

2.10.1. Εφαρμογές της XML

Από το 2009, εκατοντάδες μορφές εγγράφων που χρησιμοποιούν σύνταξη XML έχουν αναπτυχθεί, συμπεριλαμβανομένου του RSS[34], Atom, SOAP, XHTML. Οι μορφές που βασίζονται στην XML έχουν υιοθετηθεί από πολλά εργαλεία office όπως Microsoft Office (Office Open XML), OpenOffice.org, LibreOffice (OpenDocument) και το iWork της Apple. Επίσης η XML έχει χρησιμοποιηθεί σαν βασική γλώσσα σε πρωτόκολλα επικοινωνίας όπως το XMPP. Εφαρμογές για το .NET Framework[35] της Microsoft χρησιμοποιούν XML αρχεία για ρυθμίσεις.

Ένα μεγάλο κομμάτι πάνω στο οποίο χρησιμοποιείται η XML είναι η ανταλλαγή δεδομένων πάνω από τον Παγκόσμιο Ιστό. Η IETF RFC 7303 δίνει πρότυπα για την κατασκευή Internet Media Types όταν στέλνονται δεδομένα με την μορφή XML. Επίσης καθορίζει τους τύπους δεδομένων application/xml και text/xml που λέει ότι τα δεδομένα πρέπει να είναι σε μορφή XML. Πάνω στο πρότυπο του text/xml έχει ασκηθεί κριτική σαν πιθανή πηγή προβλημάτων κωδικοποίησης και έχει προταθεί να μην χρησιμοποιείται πλέον.

2.10.2. Βασικά χαρακτηριστικά XML

Σε αυτό το σημείο παρουσιάζονται κάποια βασικά χαρακτηριστικά της XML, μιας και η παρουσίαση όλων θα ήταν ένα ακατόρθωτο εγχείρημα στα πλαίσια αυτής της εργασίας, όντας πάρα πολλά.

- Χαρακτήρες

  Εξ ορισμού ένα XML έγγραφο είναι μια σειρά από χαρακτήρες. Σχεδόν κάθε νόμιμος Unicode χαρακτήρας μπορεί να εμφανιστεί σε ένα XML έγγραφο.

- Επεξεργαστής και εφαρμογή

  Ο επεξεργαστής αναλύει τις ετικέτες και περνάει την πληροφορία που περιέχεται μέσα στις ετικέτες στην εφαρμογή. Οι προδιαγραφές θέτουν το τι πρέπει ένας XML επεξεργαστής να

κάνει και τι όχι, αλλά η εφαρμογή είναι έξω από την εμβέλεια του. Ο επεξεργαστής αυτός ονομάζεται και αλλιώς XML parser.

- Ετικέτες και περιεχόμενο

Οι χαρακτήρες που αποτελούν ένα έγγραφο XML είναι χωρισμένοι σε δύο κατηγορίες, ετικέτες και περιεχόμενο. Συνήθως οι χαρακτήρες που αποτελούν μια ετικέτα ξεκινούν με το χαρακτήρα "<" και τελειώνουν με το χαρακτήρα ">".

- Στοιχείο

Ένα στοιχείο είναι ένα σύνολο χαρακτήρων που ξεκινάει με μια ετικέτα αρχής (π.χ. <section>) και τελειώνει με μια ετικέτα τέλους (π.χ. </section>). Μαζί με το περιεχόμενο που μπορεί να υπάρχει μεταξύ των δύο ετικετών, το σύνολο ονομάζεται στοιχείο. Σε περίπτωση που δεν υπάρχει περιεχόμενο τότε η ετικέτα είναι αυτή, "<section/>".

- Χαρακτηριστικό

Όταν σε μια ετικέτα αρχής υπάρχει και ένα σύνολο χαρακτήρων που μαζί αποτελούν ένα ζευγάρι όνομα/τιμή, το ζευγάρι ονομάζεται χαρακτηριστικό. Στο παρακάτω παράδειγμα το πρασινισμένο μέρος αποτελεί ένα χαρακτηριστικό

- Δήλωση XML

Ένα XML έγγραφο θα πρέπει να ξεκινάει με κάποιες πληροφορίες για αυτό όπως για παράδειγμα με την ετικέτα <?xml version="1.0" encoding="UTF-8"?>

```
<Books>
    <Book ISBN="0553212419">
        <title>Sherlock Holmes: Complete Novels...
        <author>Sir Arthur Conan Doyle</author>
    </Book>
    <Book ISBN="0743273567">
        <title>The Great Gatsby</title>
        <author>F. Scott Fitzgerald</author>
    </Book>
    <Book ISBN="0684826976">
        <title>Undaunted Courage</title>
        <author>Stephen E. Ambrose</author>
    </Book>
    <Book ISBN="0743203178">
        <title>Nothing Like It In the World</title>
        <author>Stephen E. Ambrose</author>
    </Book>
</Books>
```

Σχήμα 5: Έγγραφο XML

2.11. JSON

Η δομή δεδομένων JSON, είναι μια ανοιχτού προτύπου μορφή που χρησιμοποιεί κείμενο αναγνώσιμο και από ανθρώπους πέρα από μηχανές για την μετάδοση δεδομένων που αποτελούνται από ζεύγη χαρακτηριστικού-τιμής. Είναι η κύρια μορφή δεδομένων που χρησιμοποιείται για ασύγχρονη επικοινωνία μεταξύ φυλλομετρητή/διακομιστή(AJAJ[36], Asynchronous JavaScript And JSON), η οποία σε μεγάλο βαθμό αντικαθιστά την XML(AJAX[37], Asynchronous JavaScript And XML).

Παρόλο που αρχικά προήλθε από την JavaScript, η JSON είναι μια δομή δεδομένων ανεξάρτητη από κάποια γλώσσα. Κώδικας για επεξεργασία και παραγωγή δεδομένων σε μορφή JSON είναι διαθέσιμος σε πολλές γλώσσες προγραμματισμού.

Η JSON αρχικά προτάθηκε από τον Douglas Crockford. Σε αυτή την φάση περιγράφεται από δύο ανταγωνιστικά πρότυπα, το RFC 7159 και ECMA-404. Το ECMA περιγράφει μόνο την επιτρεπόμενη σύνταξη ενώ το RFC επίσης παρέχει θεωρήσεις σημασιολογικές και ασφάλειας. Ο επίσημος τύπος δεδομένων στο δίκτυο για το JSON είναι application/json και η επέκταση ενός JSON αρχείου είναι .json.

2.12. Τύποι δεδομένων και σύνταξη

Οι βασικοί τύποι δεδομένων της JSON είναι:

- Αριθμός(Number)
- Σειρά από χαρακτήρες(String)
- Boolean
- Πίνακας(Array)
- Κενό(null)

Οι whitespace χαρακτήρες επιτρέπονται και αγνοούνται μεταξύ συντακτικών στοιχείων (τιμές και στίξη, αλλά όχι μέσα σε μια τιμή που περιέχει μια σειρά από χαρακτήρες). Τέσσερις συγκεκριμένοι χαρακτήρες θεωρούνται ως whitespace: το κενό, το οριζόντιο tab, η line feed και carriage return. Η JSON δεν παρέχει κάποιο συγκεκριμένο συντακτικό για τα σχόλια.

Στις αρχικές εκδόσεις της JSON υποχρεούται ένα έγκυρο JSON έγγραφο να αποτελείται από ένα αντικείμενο ή ένα τύπο πίνακα που μπορούν να περιέχουν άλλους τύπους μέσα τους. Αυτός ο περιορισμός αφαιρέθηκε με το πρότυπο RFC 7158, ώστε ένα JSON έγγραφο να αποτελείται εξ ολοκλήρου από κάθε πιθανή JSON τιμή.

```
- response: {
  - groups: [
    - {
        type: "places",
        name: "Matching Places",
      - items: [
        - {
            id: "4be40357d27a20a1aa75935b",
            name: "Indira Gandhi International Airport (DEL)",
            contact: { },
          - location: {
              address: "Indira Gandhi International Airport",
              city: "New Delhi",
              state: "India",
              lat: 28.55448289261816,
              lng: 77.0896053314209,
              distance: 12291
            },
          - categories: [
            - {
                id: "4bf58dd8d48988d1ed931735",
                name: "Airport",
                pluralName: "Airports",
                icon: "https://foursquare.com/img/categories/travel/airport.png",
              - parents: [
                  "Travel Spots"
                ],
                primary: true
              },
            - {
                id: "4bf58dd8d48988d1eb931735",
                name: "Airport Terminal",
                pluralName: "Airport Terminals",
                icon: "https://foursquare.com/img/categories/travel/airport_terminal.png",
              - parents: [
                  "Travel Spots",
                  "Airports"
                ]
              }
            ],
            verified: true,
          - stats: {
              checkinsCount: 3724,
              usersCount: 1838
            },
          - hereNow: {
              count: 0
            }
          },
        - {
            id: "4b6f9e1ef964a52010f82ce3",
            name: "Gurgaon Delhi Toll Gate",
```


Σχήμα 6: Έγγραφο JSON

2.12.1. Χρήσεις

JSON-RPC

Το JSON-RPC είναι ένα πρωτόκολλο RPC βασισμένο στη JSON, σαν αντικατάσταση του XML-RPC ή του SOAP. Είναι ένα απλό πρωτόκολλο που καθορίζει ένα αριθμό τύπων δεδομένων και εντολών. Το JSON-RPC αφήνει ένα σύστημα να στείλει κοινοποιήσεις (πληροφορίες στον διακομιστή που δεν χρειάζονται απάντηση) και πολλαπλά καλέσματα στον διακομιστή που μπορούν να απαντηθούν ανεξαρτήτως σειράς.

AJAX

Η JSON χρησιμοποιείται συχνά σε τεχνικές Ajax. Με το Ajax υπάρχει η δυνατότητα να γίνει νέο αίτημα σε μια ιστοσελίδα αφού έχει φορτωθεί στον φυλλομετρητή, συνήθως σε αλληλεπίδραση με ενέργειες χρήστη πάνω στην ιστοσελίδα. Σαν μέρος του μοντέλου Ajax, τα καινούργια

δεδομένα μπορούν να ενσωματωθούν στην διεπαφή του χρήστη δυναμικά την ώρα που φτάνουν από τον διακομιστή. Για παράδειγμα, όταν ο χρήστης πληκτρολογεί κάτι σε μια φόρμα αναζήτησης, με κώδικα που βρίσκεται στην πλευρά του πελάτη, αυτό στέλνεται στον διακομιστή και απαντάει με πιθανές επιλογές από την βάση δεδομένων. Όταν αυτό υλοποιήθηκε στα μέσα του 2000, το Ajax χρησιμοποιούσε XML ως δομή για την ανταλλαγή δεδομένων. Τώρα οι περισσότεροι προγραμματιστές χρησιμοποιούν JSON για να περάσουν τα Ajax δεδομένα τους μεταξύ διακομιστή και χρήστη.

2.12.2. Τύπος MIME

Ο επίσημος τύπος δεδομένων MIME για JSON κείμενο είναι application/json. Παρόλο που πολλές σύγχρονες υλοποιήσεις έχουν υιοθετήσει τον επίσημο τύπο MIME, πολλές εφαρμογές συνεχίζουν να παρέχουν υποστήριξη σε παλαιότερους τύπους MIME. Πολλοί πάροχοι υπηρεσιών, φυλλομετρητές, δικτυακές εφαρμογές, βιβλιοθήκες, APIs[38], χρησιμοποιούν, περιμένουν ή αναγνωρίζουν ως «μη επίσημο» τύπο MIME το text/json ή τον τύπο περιεχομένου text/javascript. Παραδείγματα περιλαμβάνουν το Google Search API, Yahoo, Flickr, Facebook API, Lift Framework κ.τ.λ.

3. Προτεινόμενη Λύση

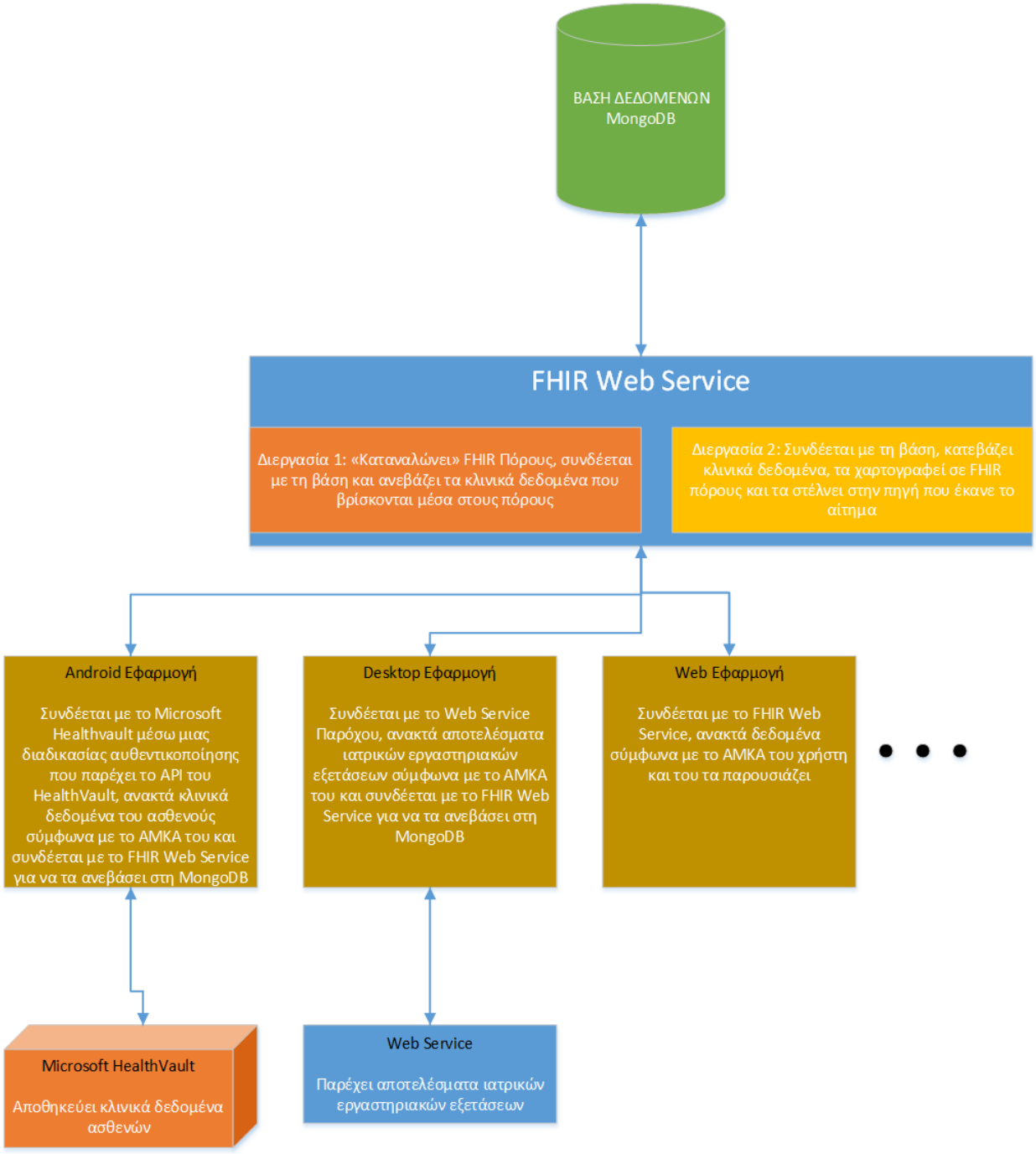
Σε αυτό το κεφάλαιο θα αναλυθεί η αρχιτεκτονική της προτεινόμενης λύσης, τα μοντέλα δεδομένων τα οποία χρησιμοποιήθηκαν και η διαδικασία υλοποίησης.

3.1. Αρχιτεκτονική

Η λύση περιλαμβάνει τρεις βασικές συνιστώσες

- Την βάση δεδομένων
- Το web service το οποίο κάνει τις μετατροπές
- Τις εφαρμογές που χρησιμοποιούν το web service είτε για να ανεβάσουν είτε για να κατεβάσουν δεδομένα

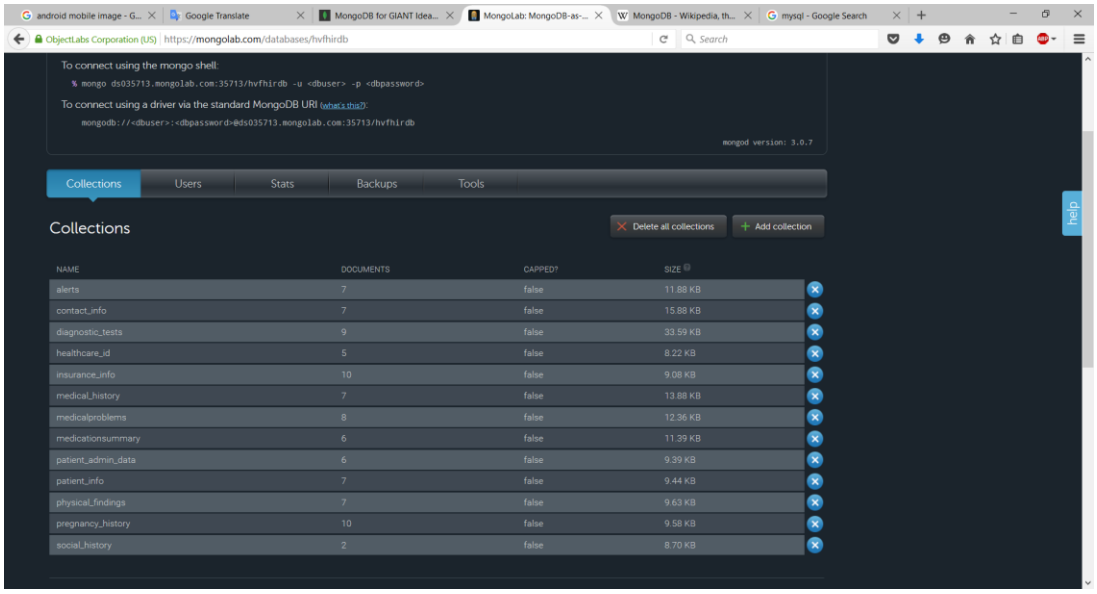
Παρακάτω παρουσιάζεται ένα σχήμα της αρχιτεκτονικής και μερικές επεξηγήσεις πάνω στο σχήμα αλλά και την υλοποίηση.



Σχήμα 7: Αρχιτεκτονική συστήματος

ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

Η βάση δεδομένων που χρησιμοποιείται είναι η MongoDB καθώς τα ιατρικά δεδομένα μπορούν να φτάσουν σε τέτοια μεγέθη ώστε τα συμβατικά σχεσιακά συστήματα βάσεων δεδομένων όπως η MySQL να μην μπορούν να τα διαχειριστούν.



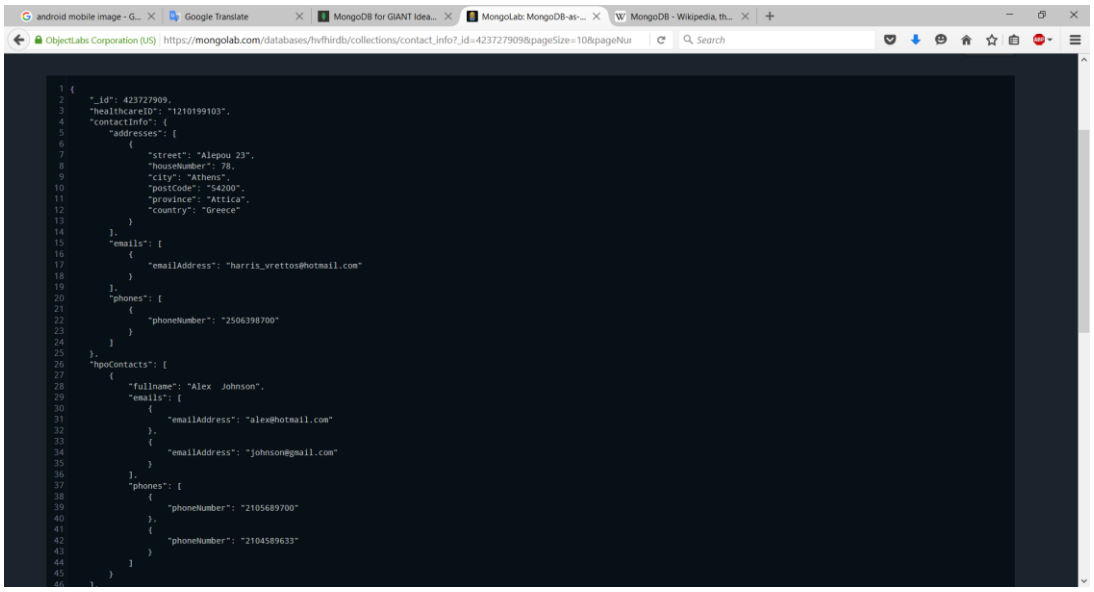
Σχήμα 8: Αναπαράσταση Βάσης Δεδομένων

Όπως μπορούμε να δούμε και στην παραπάνω εικόνα η βάση δεδομένων περιέχει 13 συλλογές (collections) γιατί τόσα είναι τα πεδία στα οποία χωρίζεται το ΠΔΙΠΑ.

- alerts – Περιέχονται στοιχεία που σχετίζονται με τις αλλεργίες του ασθενούς
- contact_info – Περιέχονται στοιχεία με τα οποία μπορεί κάποιος να έρθει σε επαφή με τον ασθενή(email, τηλέφωνο), διευθύνσεις, στοιχεία ανθρώπων που θα μπορούσε να καλέσει ο ασθενής σε περιπτώσεις έκτακτης ανάγκης και στοιχεία επαφής ανθρώπων που επιβλέπουν σε ιατρικό επίπεδο τον ασθενή
- diagnostic_tests – Περιέχονται στοιχεία σχετικά με εργαστηριακά δεδομένα αιματολογικών εξετάσεων
- healthcare_id – Περιέχεται ο ΑΜΚΑ (Αριθμός Μητρώου Κοινωνικής Ασφάλισης) ως αναφορά την Ελλάδα ή κάποιο μοναδικό αναγνωριστικό που χρησιμοποιείται στο εξωτερικό

- insurance_info – Περιέχεται ο αριθμός ιδιωτικής ασφάλισης σε κάποιο πρόγραμμα υγείας
- medical_history – Περιέχονται στοιχεία σχετικά με εμβόλια, εγχειρίσεις και προβλήματα υγείας του ασθενούς όχι αργότερα από έξι μήνες πριν
- medicalproblems – Περιέχονται στοιχεία σχετικά με εγχειρίσεις, προβλήματα υγείας, συσκευές, θεραπείες και πληροφορίες σχετικά με την ανάγκη του ασθενούς να εκτιμάται οι υγεία του από τρίτους, όχι νωρίτερα από έξι μήνες πριν
- medicationsummary – Περιέχονται στοιχεία σχετικά με τα συνταγογραφημένα φάρμακα που παίρνει ο ασθενής και δεν έχουνε περιέλθει στην ημερομηνία λήξης τους
- patient_admin_data – Περιέχονται πληροφορίες σχετικά με τη διαχείριση του ΠΔΙΠΑ του ασθενούς
- patient_info – Περιέχονται πληροφορίες σχετικά με τον ασθενή όπως όνομα, επώνυμο, φύλλο κ.α.
- physical_findings – Περιέχονται πληροφορίες σχετικά με την μέτρηση της πίεσης του ασθενούς
- pregnancy history – Περιέχονται πληροφορίες σχετικά με την εγκυμοσύνη ή εγκυμοσύνες του ασθενούς
- social_history – Περιέχονται πληροφορίες σχετικά με τις καθημερινές συνήθειες του ασθενούς όσον αφορά τον τρόπο ζωής του (κάπνισμα, κατανάλωση αλκοόλ κ.α.)

Αφού επεξηγήθηκαν τι δεδομένα περιέχονται στον κάθε πίνακα θα δούμε την μορφή με την οποία είναι αποθηκευμένα.



Σχήμα 9: Παρουσίαση μια συλλογής(collection) της Mongo

Όπως παρατηρούμε είναι σε μορφή JSON καθώς όπως αναφέρθηκε και σε προηγούμενο κεφάλαιο η Mongo είναι NoSQL βάση και δέχεται/παράγει δεδομένα σε μορφή JSON.

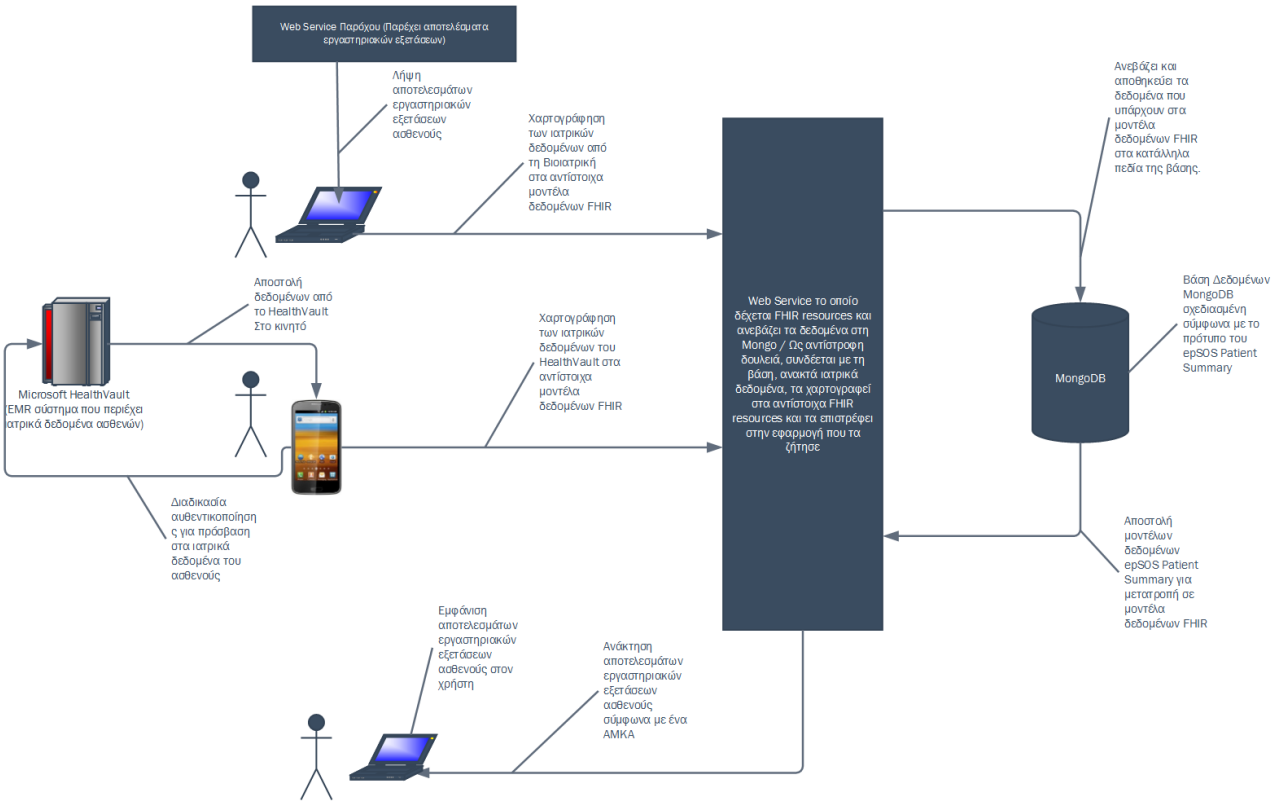
REST WEB SERVICE

Το web service κάνει δύο διεργασίες. Κατά την πρώτη διεργασία εξάγει τα απαραίτητα δεδομένα από τους «πόρους» FHIR, συνδέεται με τη βάση και τα αποθηκεύσει στα κατάλληλα πεδία. Κατά την δεύτερη διεργασία, συνδέεται με τη βάση, παίρνει τα κατάλληλα δεδομένα, τα χαρτογραφεί στους ανάλογους FHIR «πόρους» και τα στέλνει πίσω στην εφαρμογή.

ΕΦΑΡΜΟΓΕΣ

Ουσιαστικά οι εφαρμογές χρησιμοποιούν το web service κάνοντας τις ανάλογες REST αιτήσεις και αυτό αναλαμβάνει να κάνεις τις μετατροπές και τη σύνδεση με τη βάση. Για την ανάγκη της εργασίας έχουν υλοποιηθεί τρεις εφαρμογές. Η πρώτη είναι μια Android εφαρμογή η οποία σε πρώτη φάση, συνδέεται με την πλατφόρμα Microsoft HealthVault μέσω μιας διαδικασίας

αυθεντικοποίησης και ανακτά κλινικά δεδομένα. Στην συνέχεια η εφαρμογή συνδέεται με το FHIR Web Service και αποθηκεύει τα δεδομένα στη MongoDB. Η δεύτερη desktop εφαρμογή σε πρώτη φάση συνδέεται με ένα Web Service κάποιου τρίτου παρόχου και ανακτά δεδομένα ιατρικών εργαστηριακών εξετάσεων. Στην συνέχεια όπως και στην κινητή εφαρμογή συνδέεται με το FHIR Web Service και ανεβάζει τα κλινικά δεδομένα στη βάση. Τέλος η τρίτη web εφαρμογή χρησιμοποιεί το FHIR Web Service και ανακτά κάποια αποτελέσματα ιατρικών εξετάσεων σύμφωνα με ένα ΑΜΚΑ και τα παρουσιάζει στον χρήστη.



Σχήμα 10: Παρουσίαση λειτουργίας λύσης του προβλήματος της διαλειτουργικότητας EMR συστημάτων

3.2. Μοντέλο Δεδομένων

Σε αυτό το κεφάλαιο παρουσιάζονται όλα τα μοντέλα δεδομένων που κατασκευάστηκαν ή χρησιμοποιήθηκαν έτοιμα αλλά και κάποιες μετατροπές μεταξύ αυτών.

3.2.1. FHIR

Το μοντέλο FHIR έχει πάνω από 90 «πόρους»/κλάσεις για την αναπαράσταση των δεδομένων. Για την εκπόνηση της εργασίας χρησιμοποιήθηκαν οι εξής δέκα:

- Patient – Παρέχει πληροφορίες σχετικά με τον ασθενή όπως προσωπικά στοιχεία, στοιχεία επικοινωνίας, διευθύνσεις και επαφές έκτακτης ανάγκης όπως φίλους, γιατρούς κ.α.
- AllergyIntolerance – Παρέχει πληροφορίες σχετικά με ανεπιθύμητες ανταποκρίσεις του οργανισμού όταν αυτός έρθει σε επαφή με κάποια ουσία η οποία είναι μοναδική για το κάθε άτομο ξεχωριστά
- Careplan – Παρέχει πληροφορίες σχετικά με τον τρόπο που προτίθενται οι γιατροί να αντιμετωπίσουν ένα πρόβλημα υγείας για ένα ασθενή ή ομάδα ασθενών για μια συγκεκριμένη χρονική περίοδο
- Condition – Παρέχει πληροφορίες σχετικά με προβλήματα υγείας ή διαγνώσεις προβλημάτων από κάποιον γιατρό
- DetectedIssue – Υποδεικνύει ένα πραγματικό ή δυνητικό κλινικό πρόβλημα με ή μεταξύ κάποιων ενεργών ή προτεινόμενων αντιμετωπίσεων μιας κλινικής πάθησης ενός ασθενή
- Immunization – Περιγράφει το γεγονός παροχής κάποιου εμβολίου όπως αναφέρθηκε από κάποιο ασθενή ή υπεύθυνο υγείας και μπορεί να περιέχει και αντιδράσεις που προκάλεσε ο εν λόγω εμβολιασμός
- Procedure – Παρέχει πληροφορίες σχετικά με ένα γεγονός που «έγινε πάνω» σε ένα ασθενή. Αυτό το γεγονός μπορεί να είναι είτε εγχείρηση, είτε κάποια υπνοθεραπεία, είτε συμβουλευτική με κάποιον ειδικό κ.α.
- Observation – Μετρήσεις ή απλοί ισχυρισμοί που έγιναν για κάποιο ασθενή, συσκευή ή αντικείμενο

- MedicationAdministration – Περιγράφει ένα γεγονός κατά το οποίο ένας ασθενής κατανάλωσε ή του χορηγήθηκε ένα φάρμακο. Αυτό μπορεί να είναι τόσο απλό όσο η κατάποση ενός δισκίου όσο και μιας μακράς διάρκειας έγχυσης, παραδείγματος χάρη αίματος
- Coverage - Πληροφορίες που μπορούν να χρησιμοποιηθούν για την πληρωμή ή επιστροφή προϊόντων φροντίδας υγείας και υπηρεσιών

Από όλα τα πεδία που περιείχαν αυτοί οι «πόροι», δεν χρησιμοποιήθηκαν όλα. Χρησιμοποιήθηκαν μόνο αυτά τα οποία συμπίπτουν με τα πεδία του ΠΔΙΠΑ(epSOS Patient Summary), μοντέλο δεδομένων το οποίο θα αναλυθεί αμέσως μετά.

3.2.2. epSOS ΠΔΙΠΑ – epSOS Patient Summary

Το ΠΔΙΠΑ περιέχει τους εξής 13 κύριους άξονες, σύμφωνα με τους οποίους έχει φτιαχτεί ακριβώς και το μοντέλο:

- Αναγνωριστικό(Identification) – Παρέχει έναν αριθμό που είναι μοναδικός για τον ασθενή στην χώρα του ως αναφορά την ιατροφαρμακευτική περίθαλψη(π.χ. Ελλάδα ΑΜΚΑ)
- Προσωπικές Πληροφορίες(Personal Information) – Παρέχει πληροφορίες σχετικά με το όνομα, το επώνυμο, το φύλλο και την ημερομηνία γέννησης
- Πληροφορίες Επικοινωνίας(Contact Information) – Παρέχει πληροφορίες σχετικά με τη διεύθυνση του ασθενούς(δρόμος, αριθμός δρόμου, πόλη, ταχυδρομικός κώδικας, επαρχία, χώρα), emails, τηλέφωνα και πληροφορίες σχετικά με επαφές εκτάκτου ανάγκης(όνομα, τηλέφωνα και emails)
- Πληροφορίες Ασφάλισης(Insurance Information) – Παρέχει τον αριθμό ασφάλισης του ασθενούς σε κάποιο ιδιωτικό πλάνο ασφάλισης
- Κίνδυνοι(Alerts) – Παρέχει πληροφορίες σχετικά με αλλεργίες που μπορεί να έχει κάποιο ασθενής (περιγραφή, ημερομηνία εμφάνισης, αναγνωριστικό, αιτία, αναγνωριστικό αιτίας) και πληροφορίες σχετικά με πληροφορίες υγείας που είναι σημαντικές και δεν συμπεριλαμβάνονται στις αλλεργίες (περιγραφή, αναγνωριστικό)

- Ιατρικό Ιστορικό(Medical History) – Παρέχει πληροφορίες σχετικά με εμβόλια του ασθενούς (όνομα, ημερομηνία εμβολιασμού, κάποιος μοναδικός αναγνωριστικός αριθμός, εμπορικό όνομα), επεμβάσεις (περιγραφή, αναγνωριστικό, ημερομηνία επέμβασης) και προβλήματα υγείας (περιγραφή, αναγνωριστικό, ημερομηνία που έκανε την εμφάνισή του, ημερομηνία που τέλειωσε, λόγος που εξαφανίστηκε το πρόβλημα) τα οποία είναι πάνω από 6 μηνών
- Ιατρικά Προβλήματα(Medical Problems) – Παρέχει πληροφορίες σχετικά με προβλήματα υγείας (περιγραφή, αναγνωριστικό, ημερομηνία εμφάνισης), ιατρικές συσκευές (περιγραφή, αναγνωριστικό, ημερομηνία εμφύτευσης), επεμβάσεις (περιγραφή, αναγνωριστικό, ημερομηνία επέμβασης), θεραπείες για κάποιο πρόβλημα υγείας (περιγραφή, αναγνωριστικό) και πληροφορίες σχετικά με το αν κάποιος ασθενής έχει την ανάγκη διαρκής εκτίμησης της κατάστασης από τρίτους (περιγραφή, αναγνωριστικό). Όλα αυτά τα προβλήματα έχουν εμφανιστεί το αργότερο έξι μήνες πριν
- Πληροφορίες Φαρμάκων(Medication Summary) – Παρέχει πληροφορίες σχετικά με την συνταγογράφηση φαρμάκων που παίρνει κάποιος ασθενής (ενεργό συστατικό, αναγνωριστικό συστατικού, δύναμη, μορφή φαρμάκου, αριθμός «μονάδων» κατά την λήψη του φαρμάκου, συχνότητα, διάρκεια λήψης φαρμάκου, ημερομηνία έναρξης)
- Κοινωνικό Ιστορικό(Social History) – Παρέχει πληροφορίες σχετικά με καθημερινές συνήθειες του ασθενούς (περιγραφή, χρονικό διάστημα) όπως κάπνισμα, κατάποση αλκοόλ κ.α.
- Ιστορικό Γέννησης(Pregnancy History) – Παρέχει πληροφορίες σχετικά με την επικείμενη γέννα του ασθενούς(ημερομηνία γέννας)
- Φυσιολογικά Ευρήματα(Physical Findings) – Παρέχει πληροφορίες σχετικά με την μέτρηση της πίεσης του αίματος του ασθενούς(τιμές συστολικής, διαστολικής πίεσης αίματος και ημερομηνία μέτρησης)
- Διαγνωστικά Τεστ(Diagnostic Tests) – Παρέχει πληροφορίες σχετικά με αποτελέσματα εργαστηρίου εξετάσεων που έχουν γίνει πάνω στο αίμα(αποτελέσματα, ημερομηνία εξέτασης)

- Δεδομένα Διαχείρισης ΠΔΙΠΑ(Patient Administrative Data) – Παρέχει πληροφορίες σχετικά με το ίδιο το ΠΔΙΠΑ(χώρα, ημερομηνία δημιουργίας, ημερομηνία ενημέρωσης, φύση ΠΔΙΠΑ, συγγραφέας ΠΔΙΠΑ)

Τα παραπάνω δύο μοντέλα είναι αυτά που χρησιμοποιήθηκαν για την επίλυση του προβλήματος της διαλειτουργικότητας. Όμως για την επίδειξη της σωστής λειτουργίας της προτεινόμενης λύσης, κατασκευάστηκαν και δύο εφαρμογές που χρησιμοποιούν και αυτές ένα μοντέλο δεδομένων.

3.2.3. Μοντέλο HealthVault

Το HealthVault χρησιμοποιεί πάνω από 80 τύπους δεδομένων για την αναπαράσταση κλινικών και όχι μόνο δεδομένων. Όπως και στην περίπτωση του FHIR, δεν χρησιμοποιήθηκαν όλοι οι τύποι αλλά ένα μέρος αυτών που συμπίπτουν με τα πεδία του ΠΔΙΠΑ. Η εφαρμογή που υλοποιήθηκε χρησιμοποιεί τους εξής τύπους δεδομένων (από την στιγμή που χρησιμοποιήθηκαν μόνο τα πεδία που συμπίπτουν με αυτά του ΠΔΙΠΑ, το οποίο αναλύθηκε νωρίτερα, θα γίνει απλά η περιγραφή του τύπου δεδομένων και δεν θα αναφερθούν τα πεδία εντός):

- Allergy - Παρέχει στοιχεία σχετικά με αλλεργίες του ασθενή
- Basic Demographic Information – Παρέχει βασικά δημογραφικά στοιχεία σχετικά με τον ασθενή που όμως δεν δίνουν πληροφορίες σχετικά με το ποιος είναι (π.χ. φύλλο)
- Blood Pressure – Παρέχει πληροφορίες σχετικά με μετρήσεις τις πίεσης αίματος του ασθενή
- Condition – Παρέχει πληροφορίες σχετικά με προβλήματα υγείας τους ασθενούς
- Contact – Παρέχει πληροφορίες σχετικά με τις επαφές εκτάκτου ανάγκης του ασθενούς
- Immunization – Παρέχει πληροφορίες σχετικά με εμβολιασμούς που έχει υποστεί ο ασθενής
- Insurance Plan – Παρέχει πληροφορίες σχετικά με την ασφάλεια υγείας του ασθενούς
- Lab Results – Παρέχει πληροφορίες σχετικά με αποτελέσματα εργαστηριακών εξετάσεων του ασθενούς

- Medical Problem – Παρέχει πληροφορίες σχετικά με κάποιο πρόβλημα υγείας και διάγνωσής του
- Medication – Παρέχει πληροφορίες σχετικά με φάρμακα που παίρνει ή έπαιρνε ο ασθενής
- Personal Contact Information – Παρέχει πληροφορίες σχετικά με στοιχεία επικοινωνίας του ασθενούς
- Personal Demographic Information – Παρέχει πληροφορίες σχετικά με την αναγνώριση του ασθενούς (π.χ. όνομα, επώνυμο)
- Pregnancy – Παρέχει πληροφορίες σχετικά με γέννες που έχουν γίνει ή θα γίνουν
- Procedure – Παρέχει πληροφορίες σχετικά με εγχειρίσεις που έχει υποστεί ο ασθενής

3.2.4. Μορφή δεδομένων και μετατροπές

Οι δύο δομές δεδομένων που χρησιμοποιούνται για την αναπαράσταση και ανταλλαγή των δεδομένων είναι JSON και XML. Η XML χρησιμοποιείται μόνο κατά την χρήση του HealthVault ενώ στις υπόλοιπες περιπτώσεις JSON.

XML

Για την ανάκτηση των δεδομένων που έχει αποθηκεύσει ο χρήστης στο HealthVault, παρέχει ένα API που βασίζεται στην XML. Η εφαρμογή «χτυπάει» ασύγχρονα την πλατφόρμα και αυτή επιστρέφει τα δεδομένα που ζήτησε ο χρήστης σε μορφή XML. Στην συνέχεια αυτά τα δεδομένα «χαρτογραφούνται» σε τύπους δεδομένων που παρέχει το HealthVault και μπορούν να χρησιμοποιηθούν από την εφαρμογή. Αυτές οι δύο λειτουργίες που παρουσιάστηκαν βρίσκονται σε μια βιβλιοθήκη που παρέχει το HealthVault.

JSON

Η MongoDB αποθηκεύει τα δεδομένα σε μορφή JSON. Έτσι λοιπόν για το ανέβασμα των δεδομένων από το web service, χρησιμοποιήθηκαν οι κλάσεις και μέθοδοι που παρέχει ο οδηγός(driver) της mongo. Για το κατέβασμα των δεδομένων, πέρα από τον driver της mongo, χρησιμοποιήθηκε η βιβλιοθήκη Jackson που χαρτογραφεί τα δεδομένα της mongo στις κλάσεις του μοντέλου epSOS ΠΔΙΠΑ.

Ως αναφορά τις μετατροπές που γίνονται μεταξύ των μοντέλων το web service αναλαμβάνει να χαρτογραφήσει τα δεδομένα της mongo στις κλάσεις του μοντέλου epSOS ΠΔΙΠΑ και στη συνέχεια να περάσει τα δεδομένα στους «πόρους» FHIR.

3.3. Υλοποίηση

3.3.1. Τεχνολογίες

Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την υλοποίηση της Android εφαρμογής, του FHIR Web Service που κάνει τις μετατροπές και την εφαρμογή που παίρνει δεδομένα από την Βιοιατρική είναι η **Java**.

Για την υλοποίηση της Android εφαρμογής χρησιμοποιήθηκε το **Android Studio** και για τα υπόλοιπά το **IntelliJ IDEA Community.** Και στις 2 περιπτώσεις το build σύστημα για την αυτοματοποίηση διαδικασιών όπως διαχείριση εξαρτήσεων και εισαγωγή τρίτου λογισμικού ήταν το **Gradle.**

Όσον αναφορά την υλοποίηση του FHIR Web Service, χρησιμοποιήθηκε η αρχιτεκτονική **REST** και για την ανάπτυξή του, το **Spring REST MVC**. Για τα HTTP requests στην πλευρά του χρήστη και πάλι χρησιμοποιήθηκε το Spring REST MVC καθώς περιέχει έτοιμες μεθόδους και κλάσεις που κάνουν την διαχείριση τους πολύ εύκολη.

Για την online φιλοξενία των Web Services χρησιμοποιήθηκε ο Tomcat Web Server σε ένα εικονικό μηχάνημα Ubuntu Server.

Για την εφαρμογή Android χρησιμοποιήθηκε το **Spring REST MVC for android development**.

Για την επικοινωνία και την ανταλλαγή δεδομένων με την πλατφόρμα HealthVault, παρέχεται ένα **SDK** που συνυπάρχει μαζί με την κυρίως android εφαρμογή ώστε να μπορεί να χρησιμοποιεί τον κώδικά του.

Για την διαχείριση των FHIR «πόρων», χρησιμοποιήθηκε ένα έτοιμο java API με την ονομασία **Hapi-FHIR** το οποίο δεν έχει κάποια σύνδεση με το FHIR της HL7, αλλά ακολουθεί τα πρότυπά της.

Δύο σημαντικές βιβλιοθήκες που βοήθησαν στην αυτόματη χαρτογράφηση των JSON πεδίων πάνω σε κάποια java μοντέλα δεδομένων ήταν η **Jackson** που παρέχεται έτοιμη με το Spring REST MVC και η **Gson** της Google. Σκοπός αυτών των βιβλιοθηκών είναι να κάνουν πιο εύκολη την διαχείριση αρχείων που περιέχουν δεδομένα σε μορφή JSON

Τέλος για την βάση δεδομένων, όπως αναφέρθηκε και πιο πάνω, χρησιμοποιήθηκε η **MongoDB** και για την διαχείρισή της ο **java Mongo οδηγός(driver)**. Σε αρχική φάση και για λόγους δοκιμής, οι συναλλαγές με τη βάση έγιναν σε τοπικό server εντός του υπολογιστή ενώ στη συνέχεια χρησιμοποιήθηκε η **MongoLab**, μια online MongoDB πλατφόρμα φιλοξενίας(host).

3.3.2. Διαδικασία υλοποίησης και προβλήματα

Σε αρχική φάση υλοποιήθηκε η Android εφαρμογή η οποία συνδέεται με την πλατφόρμα HealthVault και ανακτά δεδομένα που έχει αποθηκεύσει ο χρήστης στον λογαριασμό του. Για αυτή την διαδικασία προσφέρεται μια μικρή βιβλιοθήκη για android η οποία όμως σε αντίθεση με την βιβλιοθήκη για C# που παρέχει η Microsoft, λείπουν πολλές λειτουργίες. Η βιβλιοθήκη δεν περιλάμβανε όλες τις δομές δεδομένων που χρειάζονται για την αναπαράσταση των πληροφοριών που έχει αποθηκεύσει ο χρήστης στην πλατφόρμα, έτσι λοιπόν έπρεπε να κατασκευαστούν. Επίσης είχε και κάποια λάθη που εμπόδιζαν την σωστή λειτουργία του προγράμματος και διορθώθηκαν. Εν τέλει κατασκευάστηκε μια εφαρμογή κατά την οποία ο χρήστης εισάγει τα Windows live ID στοιχεία του για να μπορέσει να συνδεθεί στην εφαρμογή και επιλέγει τα δεδομένα που θέλει να εισάγει στην κεντρική βάση.

Στη συνέχεια υλοποιήθηκε το REST FHIR Web Service το οποίο σε αρχική φάση κάνει τις απαραίτητες μετατροπές και ανεβάζει ιατρικά δεδομένα στη βάση είτε από το HealthVault ή από άλλες πηγές και ως δεύτερη λειτουργία, παίρνει δεδομένα από τη βάση και τα παρέχει σε εφαρμογές που τα ζήτησαν.

Για τη διαδικασία του ανεβάσματος των δεδομένων στη βάση, δημιουργήθηκαν 13 κλάσεις/πόροι σύμφωνα με το μοντέλο ΠΔΙΠΑ, όπου κάθε μία δέχεται ως παράμετρο έναν ή περισσότερους πόρους FHIR. Η κάθε κλάση αναλαμβάνει να συνδεθεί στη βάση και να εισάγει τα δεδομένα τα οποία έλαβε. Σε περίπτωση επιτυχίας στέλνει μήνυμα επιτυχίας(HTTP 200) ή σε περίπτωση αποτυχίας στέλνει το ανάλογο μήνυμα(HTTP 500, HTTP 404 κ.α.). Για να συνδεθεί μια εφαρμογή με το FHIR Web Service και να ανεβάσει δεδομένα, πρέπει να χρησιμοποιήσει ένα από τα 13 URL που αντιστοιχούν σε μια από τις κλάσεις που υλοποιεί την ενέργεια που περιεγράφηκε πιο πάνω.

Από την άλλη πλευρά μπορεί κάποια εφαρμογή να ζητήσει κάποια δεδομένα από την βάση. Σε αυτή την περίπτωση έχουν φτιαχτεί άλλες 10 κλάσεις/πόροι που δέχονται δεδομένα σύμφωνα με το μοντέλο ΠΔΙΠΑ και επιστρέφουν τον ανάλογο «πόρο» FHIR που ζητήθηκε. Όπως αναφέρθηκε και πιο πάνω έχει υλοποιηθεί για τις ανάγκες τις εργασίας ένα τέτοιο μοντέλο, σύμφωνα με το ΠΔΙΠΑ. Η διαδικασία είναι η εξής. Το Web Service συνδέεται με τη βάση και ζητάει κάποια δεδομένα. Εφόσον υπάρχουν δεδομένα που ικανοποιούν το ερώτημα που έγινε, τα δεδομένα χαρτογραφούνται με τη βοήθεια της βιβλιοθήκης Jackson στις ανάλογες κλάσεις του μοντέλου ΠΔΙΠΑ. Έπειτα γίνεται η μετατροπή από ΠΔΙΠΑ σε FHIR και στο τέλος το επιστρέφεται ο ανάλογος «πόρος» FHIR. Και πάλι, επιστρέφεται το ανάλογο HTTP μήνυμα όπως και στην αντίστροφη διαδικασία.

Ένα πρόβλημα που παρουσιάστηκε κατά την διάρκεια υλοποίησης του FHIR Web Service ήταν κατά τη διαδικασία του serialization των FHIR κλάσεων, δηλαδή της διαδικασίας μετατροπής των δεδομένων σε μια σειρά από bytes για την μετάδοση πάνω από το HTTP και του de-serialization των FHIR κλάσεων, δηλαδή την διαδικασία μετατροπής μιας σειράς από bytes σε κάποιο αντικείμενο για να μπορέσει το πρόγραμμα να το χρησιμοποιήσει. Για να αντιμετωπιστεί αυτό το πρόβλημα χρησιμοποιήθηκαν κάποιες λειτουργίες του API του Hapi-FHIR για την μετατροπή των κλάσεων αυτών σε μορφή String ώστε να μπορέσουν στη συνέχεια να γίνουν serialized και de-serialized από το Spring και να μεταδοθούν με επιτυχία.

Από τη στιγμή που επιτεύχθηκε η λειτουργία του FHIR Web Service, έπρεπε να γίνει η σύνδεση της android εφαρμογής με αυτό. Όμως για να γίνει η χρήση του, έπρεπε να εισαχθούν στην εφαρμογή μέσω Gradle οι αντίστοιχες βιβλιοθήκες του FHIR κάτι που παρουσίασε και άλλο πρόβλημα. Κατά τη διάρκεια της διαδικασίας build το πρόγραμμα έβγαζε σφάλματα και δεν

μπορούσε να τρέξει. Το πρόβλημα δεν επιλύθηκε καθώς η βιβλιοθήκη FHIR παρουσίαζε προβλήματα και όταν γινόταν η εισαγωγή του σε κενή εφαρμογή και σύμφωνα με τις πληροφορίες που παρουσίαζε το log το πιθανότερο είναι να υπάρχει πρόβλημα εξαρτήσεων μεταξύ των βιβλιοθηκών. Ως τελευταία ενέργεια, έγινε η προσπάθεια εισαγωγής της βιβλιοθήκης FHIR που ήταν ειδικά για android, αλλά τα ίδια προβλήματα παρέμειναν.

Για την επίλυση του προβλήματος, εν τέλει, δημιουργήθηκε ένα νέο Web Service, ίδιο με το προηγούμενο σε λειτουργία, που δεν χρησιμοποιεί καθόλου FHIR αλλά δέχεται και παρέχει μόνο δεδομένα σύμφωνα με το μοντέλο ΠΔΙΠΑ. Έτσι λοιπόν με την βοήθεια του Spring MVC για android έγιναν οι HTTP κλήσεις για το ανέβασμα των δεδομένων στη βάση. Οι κλήσεις έγιναν ασύγχρονα καθώς μπορούν να γίνουν μόνο εκτός main thread στο Android.

Όλες αυτές οι κλήσεις που γίνονται προς το FHIR Web Service περιέχουν ως παράμετρο ένα αναγνωριστικό υγείας του ασθενούς ώστε να επιστρέφονται στοιχεία που έχουν να κάνουν με μόνο ένα ασθενή.

Επίσης η εργασία περιλαμβάνει μια εφαρμογή που συνδέεται μέσω ενός τρίτου Web Service παρόχου υπηρεσιών υγείας και ανακτά δεδομένα εργαστηριακών εξετάσεων. Αφού τα ανακτήσει, συνδέεται με το Web Service που υλοποιήθηκε στα πλαίσια της εργασίας και ανεβάζει τα δεδομένα. Η σύνδεση με το FHIR Web Service δεν θα επεξηγηθεί καθώς η διαδικασία είναι η ίδια με πριν. Όσον αφορά τη σύνδεση με το Web Service του παρόχου, πάλι χρησιμοποιήθηκε η βιβλιοθήκη Jackson για χαρτογράφηση των JSON δεδομένων σε ένα μοντέλο που υλοποιήθηκε ειδικά για το συγκεκριμένο Web Service. Και εδώ χρησιμοποιείται η αρχιτεκτονική REST ενώ για την ανάκτηση δεδομένων γίνονται HTTP κλήσεις.

Τέλος, υλοποιήθηκε και μια web εφαρμογή που χρησιμοποιεί την αντίστροφη διαδικασία του FHIR Web Service κατά την οποία συνδέεται με τη βάση, τραβάει δεδομένα και τα στέλνει στην εφαρμογή. Η εφαρμογή αποτελείται από δύο απλές διεπαφές. Στην πρώτη διεπαφή, ο χρήστης εισάγει το ΑΜΚΑ του και μόλις πατήσει το κουμπί submit του παρουσιάζονται στην δεύτερη διεπαφή αποτελέσματα σχετικά με εξετάσεις αίματος που είχε εισάγει στο παρελθόν.

Τα δύο Web Services και η βάση δεδομένων στην αρχή δοκιμάστηκαν σε έναν τοπικό διακομιστή εντός του υπολογιστή. Στην συνέχεια όσον αφορά τη βάση δεδομένων δημιουργήθηκε ένας

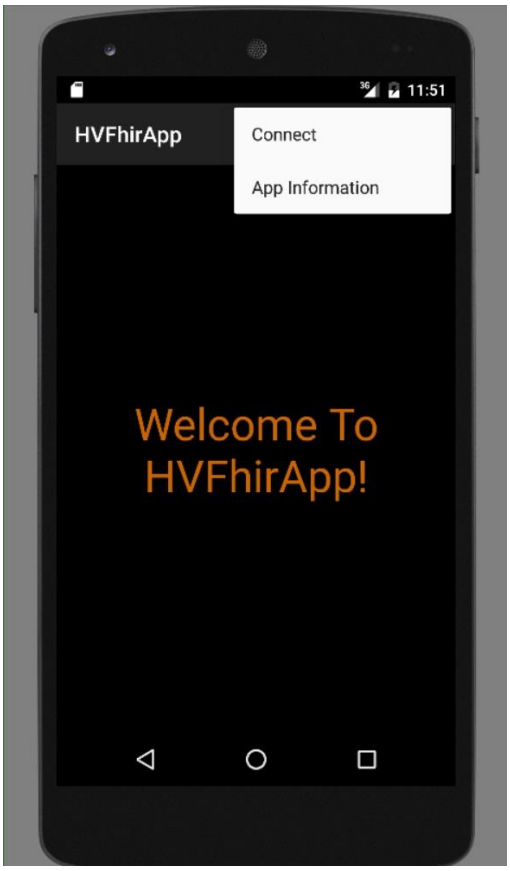
λογαριασμός στο mongolab για την φιλοξενία της MongoDB βάσης. Για τα Web Services, δημιουργήθηκε μια εικονική μηχανή(virtual machine) στον Okeanos που τρέχει Ubuntu Server και εγκαταστάθηκε ένας Tomcat Web Server για να τα φιλοξενήσει. Ο Tomcat Server φιλοξενεί και την web εφαρμογή.

4. Προσέγγιση Προβλήματος

Σε αυτό το κεφάλαιο θα παρουσιαστεί η προτεινόμενη λύση για το πρόβλημα που περιεγράφηκε στα προηγούμενα κεφάλαια της διπλωματικής μέσω ενός σεναρίου χρήσης.

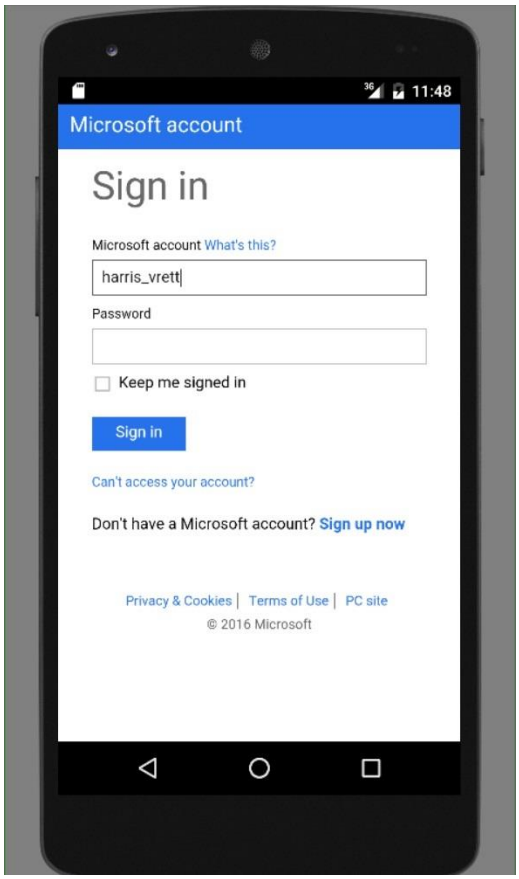
4.1. Εφαρμογή Android

Η πρώτη εικόνα παρουσιάζει τις επιλογές που έχει ο χρήστης όταν θα εισέλθει για πρώτη φορά στην εφαρμογή. Η πρώτη επιλογή είναι να συνδεθεί με το HealthVault και η δεύτερη είναι να δει πληροφορίες σχετικά με εφαρμογή και πώς να τη χρησιμοποιήσει.



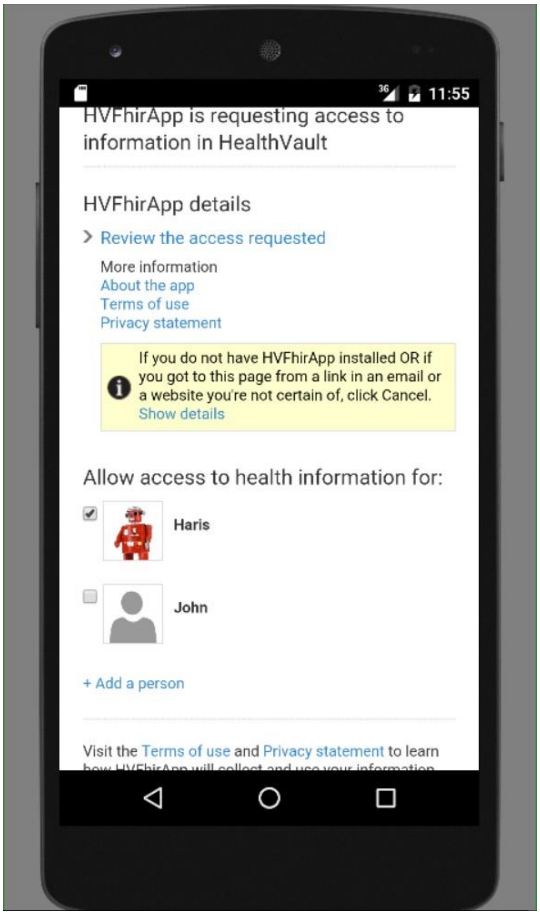
Σχήμα 11: Αρχική Διεπαφή και μενού της εφαρμογής

Σε περίπτωση που πατήσει να συνδεθεί, δίνεται η δυνατότητα στο χρήστη να κάνει login στο HealthVault μέσω διάφορων λογαριασμών που μπορεί να έχει όπως Facebook, Windows Live ID, OpenID, Ubisecure. Σε αυτή την περίπτωση χρησιμοποιείται λογαριασμός Windows.



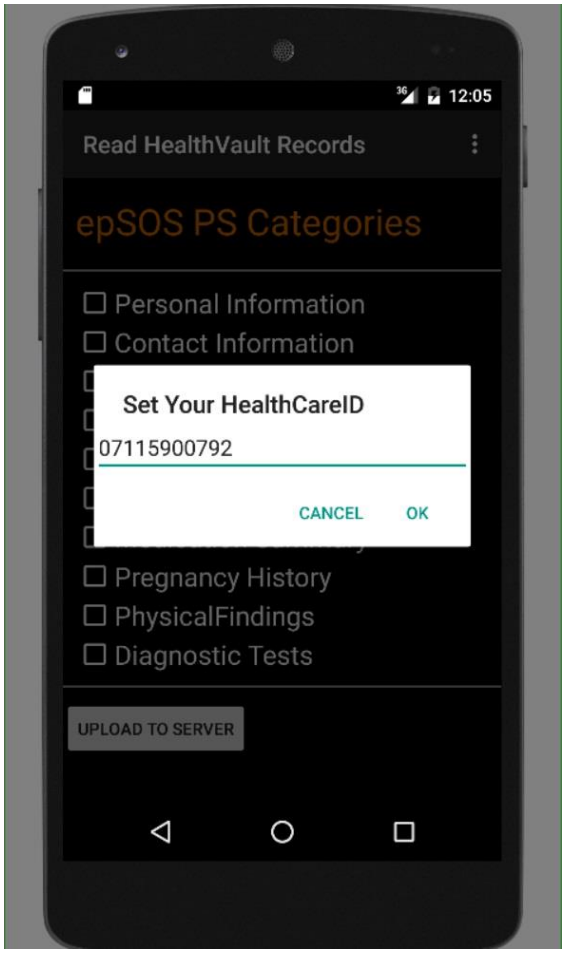
Σχήμα 12: Διεπαφή σύνδεσης της εφαρμογής με το HealthVault

Όπως επισημάνθηκε και πιο πάνω ένας χρήστης μπορεί να έχει πρόσβαση σε κλινικά στοιχεία παραπάνω του ενός ατόμου. Έτσι μετά την επιτυχή σύνδεση του χρήστη πρέπει να επιλέξει σε ποιους λογαριασμούς θα έχει πρόσβαση. Σε αυτή την περίπτωση επιλέγεται να υπάρχει πρόσβαση μόνο στον λογαριασμό Haris.



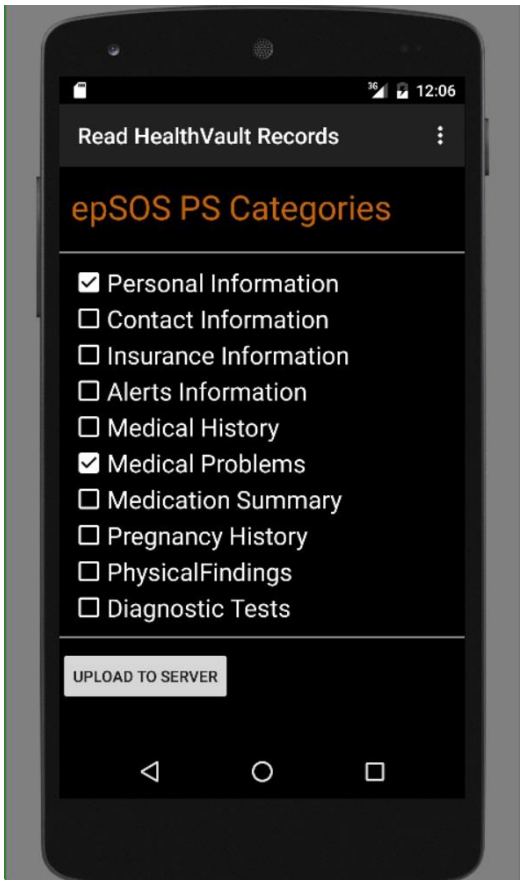
Σχήμα 13: Διεπαφή επιλογής των λογαριασμών που ο κύριος χρήστης θα έχει πρόσβαση

Έπειτα η εφαρμογή πηγαίνει τον χρήστη στην κεντρική εφαρμογή όπου πριν την χρησιμοποιήσει, πρέπει να εισάγει το ΑΜΚΑ του από την αντίστοιχη επιλογή στο κεντρικό μενού.



Σχήμα 104: Εισαγωγή αναγνωριστικού αριθμού υγείας(ΑΜΚΑ για Ελλάδα)

Όπως φαίνεται παρακάτω, η κεντρική διεπαφή αποτελείται από checkboxes και το κάθε πεδίο checkbox αντιπροσωπεύει μια κατηγορία epSOS ΠΔΙΠΑ. Ο χρήστης επιλέγοντας ένα από αυτά ουσιαστικά επιλέγει τα δεδομένα που θέλει να ανεβάσει στην βάση δεδομένων. Σε αυτή την περίπτωση ανεβάζει δεδομένα τα οποία σχετίζονται με τα προσωπικά του στοιχεία και με τα προβλήματα υγείας το τελευταίο εξάμηνο. Αφού τα επιλέξει πατάει το κουμπί UPLOAD TO SERVER και σε περίπτωση επιτυχημένης ή αποτυχημένης σύνδεσης του έρχεται το ανάλογο μήνυμα.

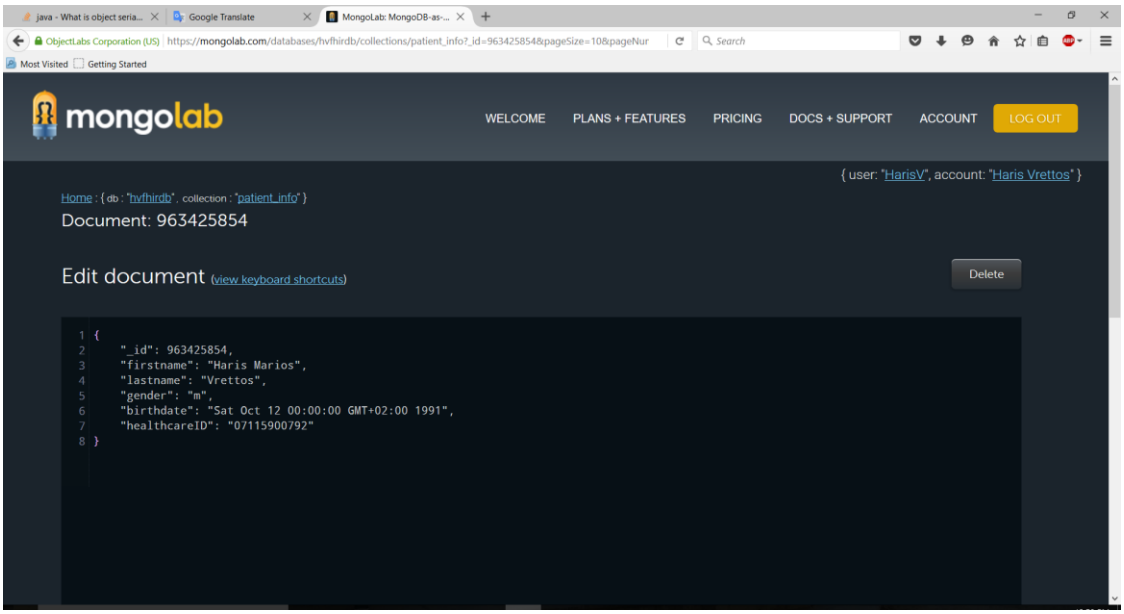


Σχήμα 15: Διεπαφή κύριας λειτουργίας διεπαφής και παρουσίασης των κατηγοριών epSOS

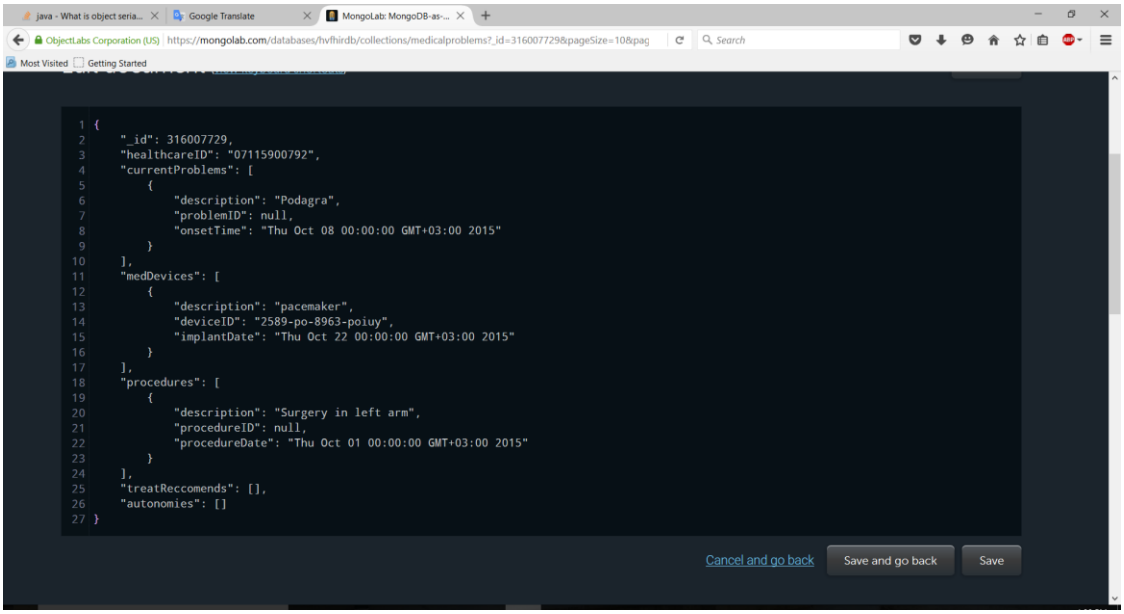
4.2. Αποτελέσματα στη βάση από σύνδεση με Android

Σε αυτή τη φάση θα παρουσιαστούν τα αποτελέσματα της διαδικασίας που παρουσιάστηκε παραπάνω. Οι τρεις παρακάτω εικόνες δείχνουν τα αποτελέσματα που προήλθαν από την χρήση της εφαρμογής. Η πρώτη αφορά δεδομένα σχετικά με τα προσωπικά στοιχεία του χρήστη, η δεύτερη περιέχει κλινικά δεδομένα σχετικά με προβλήματα υγείας του χρήστη το τελευταίο εξάμηνο και η τρίτη εικόνα περιέχει στοιχεία διαχείρισης του ΠΔΙΠΑ. Αυτά τα τρία πεδία του πίνακα ήταν που τροποποιήθηκαν με τις ενέργειες που έγιναν στην Android εφαρμογή. Όπως μπορούμε να παρατηρήσουμε και στους τρείς πίνακες υπάρχει το ΑΜΚΑ του ασθενούς το οποίο

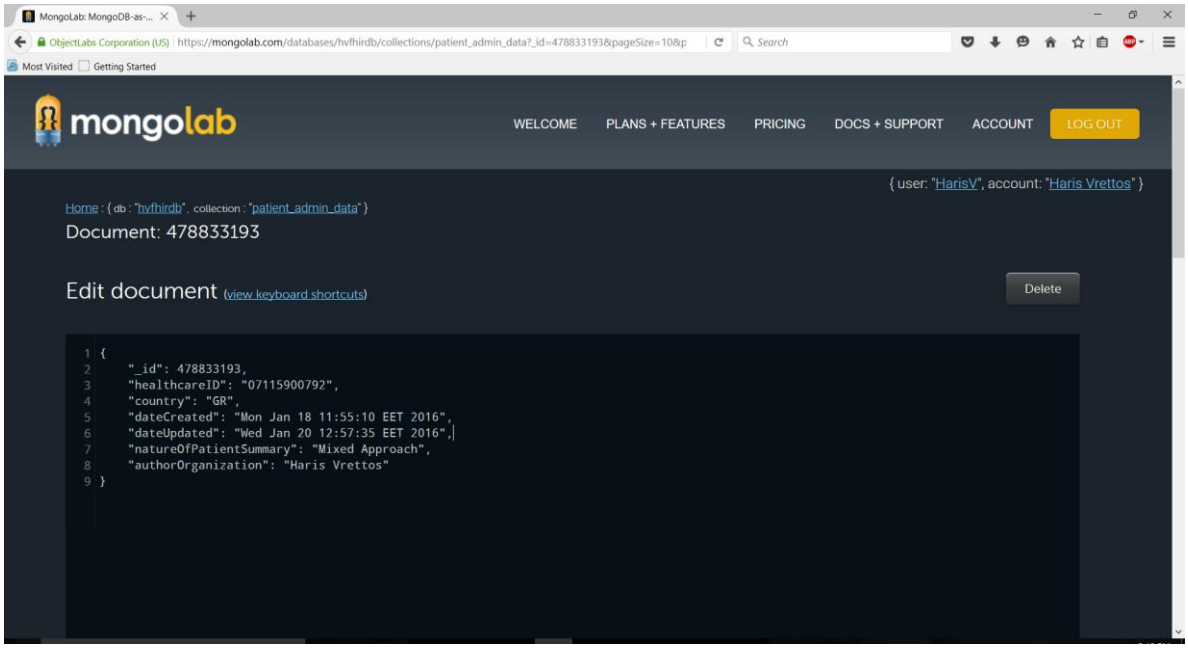
χρησιμεύει ως βασικό χαρακτηριστικό για τα ερωτήματα στην βάση αλλά και τη σύνδεση των διαφορετικών συλλογών(collections) της βάσης μεταξύ τους.



Σχήμα 16: Αποτελέσματα σχετικά με τα προσωπικά στοιχεία του χρήστη



Σχήμα 17: Αποτελέσματα σχετικά με προβλήματα υγείας του χρήστη

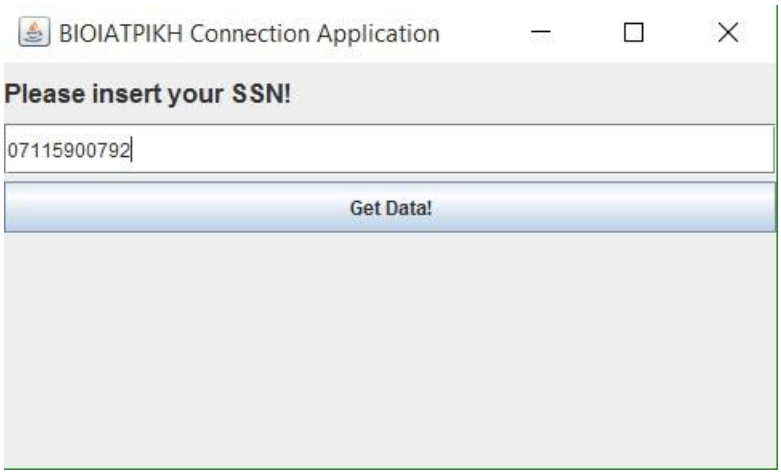


Σχήμα 18: Αποτελέσματα σχετικά με δεδομένα διαχείρισης του ΠΔΙΠΑ

4.3. Desktop JAVA εφαρμογή

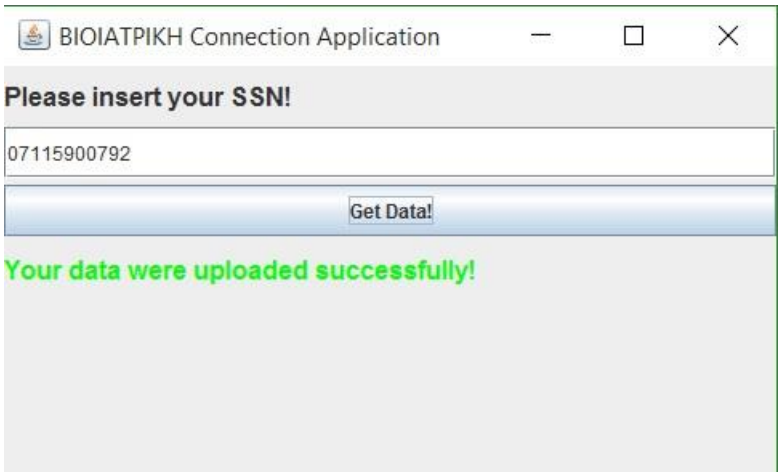
Αφού παρουσιάστηκε η εφαρμογή η οποία συνδέεται με ένα σύστημα διαχείρισης ηλεκτρονικών ιατρικών φακέλων και ανακτά δεδομένα, τώρα θα δούμε την λειτουργία μιας άλλης desktop java εφαρμογής που σχετίζεται με την σύνδεση σε server της Βιοιατρικής μέσω ενός τρίτου Web Service, ανακτά εργαστηριακού τύπου δεδομένα και τα αποθηκεύει στην βάση.

Η παρακάτω εικόνα παρουσιάζει μια πολύ βασική διεπαφή με χρήση της βιβλιοθήκης Swing στην οποία ο χρήστης εισάγει το ΑΜΚΑ του και πατάει το κουμπί για να ανεβάσει τα δεδομένα του.



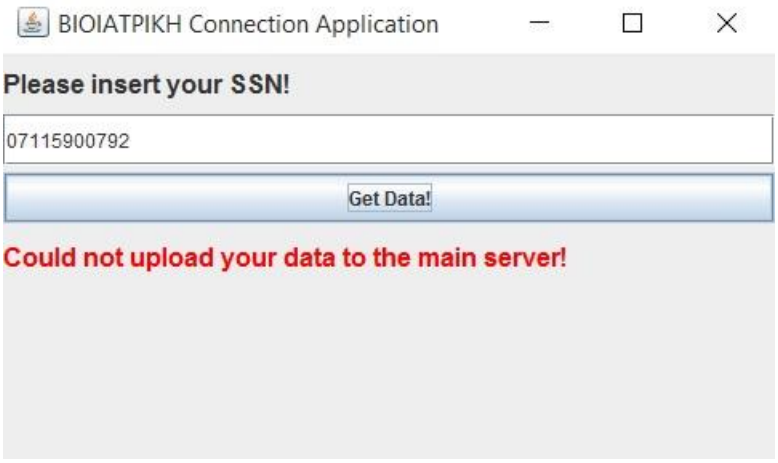
Σχήμα 1911: Βασική διεπαφή της desktop εφαρμογής

Σε περίπτωση επιτυχημένης αποθήκευσης των εργαστηριακών δεδομένων εμφανίζεται στον χρήστη το ακόλουθο μήνυμα.



Σχήμα 20: Αποτέλεσμα επιτυχούς σύνδεσης με τη βάση και αποθήκευση δεδομένων

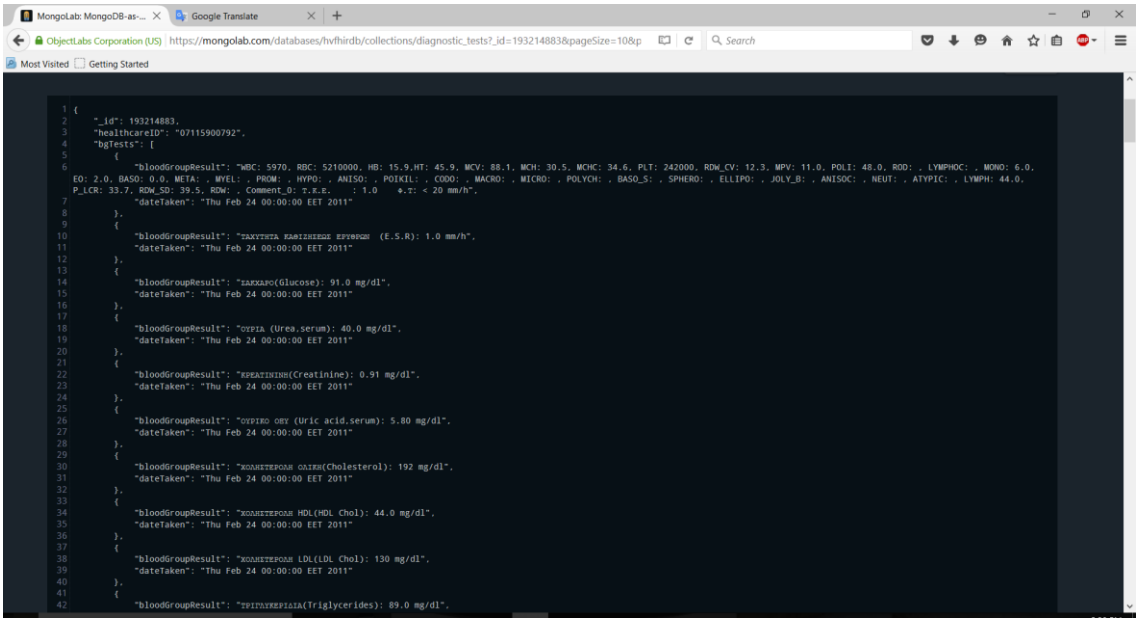
Ενώ σε περίπτωση αποτυχημένης σύνδεσης με τα δύο web service ή αποθήκευσης των δεδομένων η εφαρμογή βγάζει το ακόλουθο μήνυμα.



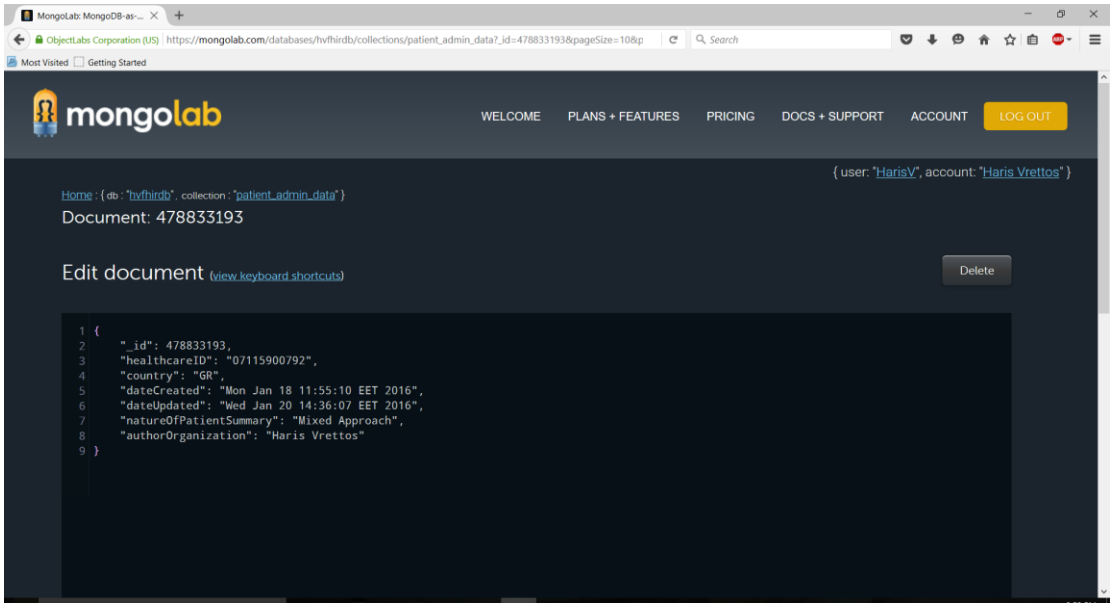
Σχήμα 21: Αποτελέσματα αποτυχημένης σύνδεσης με τη βάση ή αποθήκευσης δεδομένων

4.4. Εμφάνιση αποτελεσμάτων στη σύνδεση με την Desktop εφαρμογή

Στις επόμενες 2 εικόνες μπορούμε να παρατηρήσουμε τις αλλαγές που επήλθαν στην βάση δεδομένων. Στην πρώτη εικόνα παρατηρούμε ότι η συλλογή(collection) diagnostic_tests έχει αλλάξει και έχουν προστεθεί για τον χρήστη τα αποτελέσματα των εξετάσεων και η ημερομηνία που έγινε η εξέταση. Στην δεύτερη εικόνα όπως και σε κάθε αλλαγή της βάσης δεδομένων ενημερώνεται και η συλλογή patient_administrative_data που περιέχει στοιχεία σχετικά με την διαχείριση του ίδιου το ΠΔΙΔΑ.



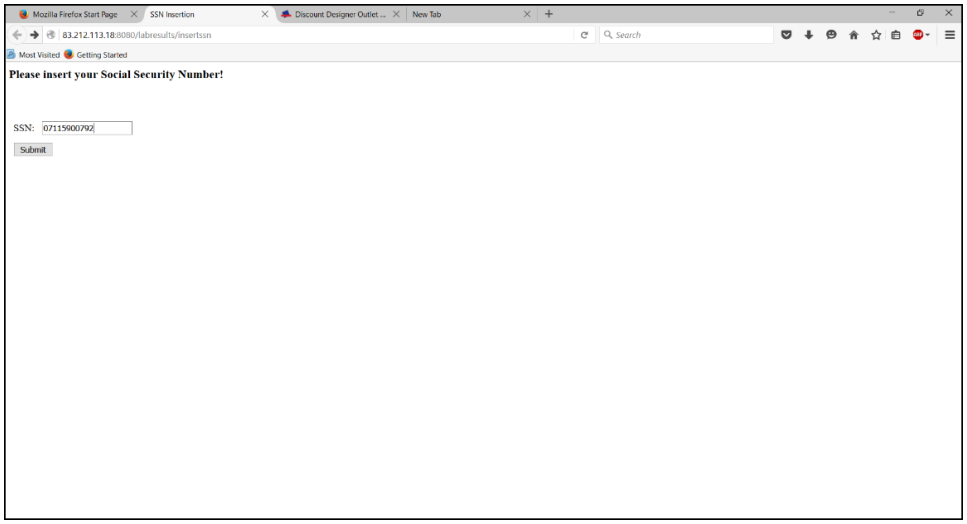
Σχήμα 22: Αποτελέσματα ασθενούς σχετικά με εργαστηριακές εξετάσεις



Σχήμα 23: Αποτελέσματα σχετικά με την διαχείριση του ΠΔΙΠΑ

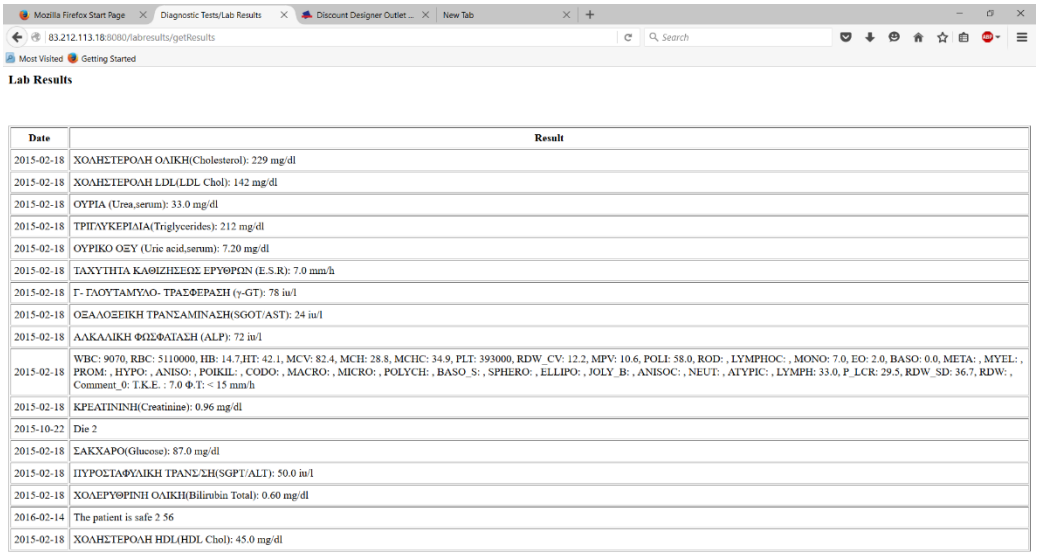
4.5. Web Εφαρμογή που παρουσιάζει τα δεδομένα

Στην παρακάτω εικόνα παρουσιάζεται μια πολύ απλή διεπαφή στην οποία ο χρήστης εισάγει το ΑΜΚΑ του για να του επιστραφούν αποτελέσματα σχετικά με εργαστηριακές εξετάσεις που είχε αποθηκεύσει με την desktop εφαρμογή.



Σχήμα 24: Εισαγωγή ΑΜΚΑ στην Web εφαρμογή

Στην παρακάτω εικόνα παρουσιάζονται τα αποτελέσματα των εξετάσεων αίματος του ασθενούς που πληκτρολόγησε το ΑΜΚΑ



Σχήμα 25: Εμφάνιση εργαστηριακών εξετάσεων

5. Συμπεράσματα

Η διπλωματική αυτή είχε σκοπό ως σκοπό να παρουσιάσει μια λύση στο πρόβλημα της διαλειτουργικότητας μεταξύ συστημάτων διαχείρισης ηλεκτρονικών ιατρικών φακέλων, μέσω των προτύπων epSOS και FHIR. Φυσικά το πόσο σημαντική είναι μια λύση σε ένα πρόβλημα, εξαρτάται και από το βαθμό που χρησιμοποιείται ώστε να επιλύσει το πρόβλημα. Αυτή τη στιγμή η χρήση ηλεκτρονικών ιατρικών φακέλων στα συστήματα υγείας ανά τον κόσμο είναι ποσοστιαία πολύ μικρή. Στην εισαγωγή, σύμφωνα με άρθρο του Bloomberg το 2013, παρουσιάστηκαν οι 10 χώρες που έχουν προχωρήσει σε μεγαλύτερο βαθμό στην ψηφιοποίηση των ιατρικών τους δεδομένων και η 10^{η} χώρα έχει ως ποσοστό 41%. Στις υπόλοιπες χώρες το ποσοστό θα είναι πολύ μικρότερο ενώ σε κάποιες άλλες μηδαμινό. Όπως γίνεται κατανοητό το ποσοστό ψηφιοποίησης των κλινικών δεδομένων έχει άμεσο αντίκτυπο και στην χρήση των συστημάτων διαχείρισης ηλεκτρονικών ιατρικών φακέλων. Όσο μεγαλύτερο είναι, τόσο μεγαλύτερο το ποσοστό χρήσης τέτοιων συστημάτων κάνοντας πιο σημαντική την αξίας των λύσεων σε τέτοια προβλήματα.

Η λύση σε ένα πρόβλημα, όσο μικρό και αν είναι αυτό, δεν παύει να είναι σημαντική καθώς ο κόσμος και η τεχνολογία εξελίσσεται και προσαρμόζεται στις ανάγκες του σήμερα κάνοντας την ανάγκη αυτοματοποίησης διαδικασιών στον τομέα της υγείας πολύ σημαντική.

Σε πιο τεχνικό επίπεδο τα δύο πρότυπα που χρησιμοποιήθηκαν, epSOS ΠΔΙΠΑ και FHIR, είναι καινούργια πρότυπα. Η υπηρεσία του προγράμματος epSOS, ΠΔΙΠΑ, βγήκε μόλις το 2014 από την πιλοτική της λειτουργία ενώ για το FHIR αναμένεται το πλήρες πρότυπο μέσα στο 2017 καθώς μέχρι τώρα τα δύο DSTU που έχουν βγει έχουν χαρακτήρα πειραματικό. Ιδιαίτερα το FHIR που είναι το βασικό εργαλείο στην επίλυση της διαλειτουργικότητας, παρόλο που έχει χρησιμοποιηθεί από μεγάλα ονόματα της πληροφορικής υγείας αναμένεται να χρησιμοποιηθεί ακόμα περισσότερο ώστε να προσφέρει ακόμα περισσότερες λύσεις και σε πολλά άλλα προβλήματα.

Η παρούσα εργασία δεν ξύνει ούτε την επιφάνεια των όσων έχει να προσφέρει το FHIR. Παρέχει πολύ περισσότερους «πόρους» από όσους χρησιμοποιήθηκαν για την αναπαράσταση κλινικών

δεδομένων, ενώ το κυρίως API έχει πάρα πολλές λειτουργίες. Σκοπός ήταν να παρουσιαστεί η βασική ιδέα και όχι να μπορέσει να χρησιμοποιηθεί αυτούσια στην αγορά.

Ως επόμενο βήμα και πάρα πολύ σημαντικό, θα ήταν η υλοποίηση ενός συστήματος ασφάλειας και ως προς τη βάση δεδομένων αλλά και ως προς τη χρήση του FHIR Web Service καθώς αναφερόμαστε σε ευαίσθητα προσωπικά δεδομένα στα οποία θα πρέπει να υπάρχει πρόσβαση μόνο από εξουσιοδοτημένους χρήστες όπως παραδείγματος χάριν τον ίδιο τον ασθενή ή και τον θεράπων ιατρό που επιβλέπει τον ασθενή.

Όπως ειπώθηκε και σε προηγούμενο κεφάλαιο, το FHIR έχει κάποια προβλήματα με την χρήση του στο Android. Λόγω του ότι ένα μεγάλο ποσοστό προγραμματιστών φτιάχνει εφαρμογές πάνω σε αυτό το λειτουργικό για τον τομέα της υγείας με σκοπό την ανταλλαγή κλινικών δεδομένων μπορεί να αποτελέσει τροχοπέδα στην επιλογή του. Έτσι λοιπόν θα πρέπει σε μελλοντικές εκδόσεις να λυθεί αυτό το πρόβλημα ώστε να μπορέσει να χρησιμοποιηθεί με ευκολία σε μελλοντικές εφαρμογές.

Έτσι όπως είναι υλοποιημένο το FHIR Web Service, κατά τη διαδικασία ανεβάσματος των κλινικών δεδομένων του ασθενούς, σε περίπτωση που «χαρτογραφηθούν» περισσότερα δεδομένα από αυτά που ταιριάζουν στο epSOS ΠΔΙΠΑ, τα υπόλοιπα θα χαθούν καθώς δεν θα ανέβουν στην MongoDB. Για να μπορέσει να γίνει εκμετάλλευση και των υπόλοιπων κλινικών δεδομένων, για μελλοντική χρήση τους, υπάρχει η δυνατότητα χρήσης του REST API που παρέχει το HAPI-FHIR σε συνδυασμό με τους FHIR servers που είναι διαθέσιμοι αυτή τη στιγμή σε επίπεδο δοκιμής. Το REST API επιτρέπει το ανέβασμα δεδομένων που βρίσκονται σε πόρους FHIR σε κάποιους servers είτε σε JSON ή XML μορφή με τη χρήση ανάλογων REST αιτημάτων. Έτσι λοιπόν ως μελλοντική υλοποίηση μέσα στο FHIR Web Service, θα μπορούσαν σε αρχική φάση τα κλινικά δεδομένα να αποθηκεύονται με τη χρήση του REST API σε κάποιους FHIR Servers και στη συνέχεια να πραγματοποιηθεί η είδη υλοποιημένη διαδικασία ανεβάσματος των δεδομένων που αντιστοιχούν στο epSOS ΠΔΙΠΑ. Φυσικά θα μπορούσε να γίνει το ανέβασμα όλων των δεδομένων στην MongoDB αλλά με τον παραπάνω τρόπο ο κώδικας μειώνεται εμφατικά καθώς δεν χρειάζεται να δοθούν ονόματα στα κλειδιά στις JSON τιμές χειροκίνητα.

6. Παράρτημα

Ο Παρακάτω κώδικας παρουσιάζει την διαδικασία μετατροπής των δεδομένων που λαμβάνει το web service ώστε να τα περάσει στη βάση και πιο συγκεκριμένα των δεδομένων που σχετίζονται με το Ιατρικό Ιστορικό του Ασθενή.

```
@RestController
@RequestMapping(value="/fhirtoepsos")
public class EPSOSMedicalHistory {

   IParser parser;
   MongoDatabase mDatabase;
   final String COLLECTION = "medical_history";
   final String DATABASE = "hvfhirdb";
   boolean OK = true;

   public EPSOSMedicalHistory() {
      try{
         parser = Server.ctx.newJsonParser();
         mDatabase = Server.mClient.getDatabase(DATABASE);
      }catch(Exception ex){
         OK = false;
      }
   }

   @RequestMapping(value="/medicalhistory/{healthcareID}",
method=RequestMethod.POST)
   public ResponseEntity<String> setMedicalHistory(@RequestBody
MedicalHistoryWrapper mwrapper,
                                    @PathVariable(value="healthcareID") String
healthcareID){

      if(OK){

         List<Immunization> immunizations = new ArrayList<>();
         List<Condition> conditions = new ArrayList<>();
         List<Procedure> procedures = new ArrayList<>();

         for(String s : mwrapper.getImmunizations()){
            immunizations.add((Immunization)parser.parseResource(s));
         }

         for(String s : mwrapper.getConditions()){
            conditions.add((Condition)parser.parseResource(s));
         }

         for(String s : mwrapper.getProcedures()){
            procedures.add((Procedure)parser.parseResource(s));
         }

         List<BasicDBObject> vaccinations = new ArrayList<>();
         List<BasicDBObject> resolvedProblems = new ArrayList<>();
         List<BasicDBObject> procs = new ArrayList<>();
```

```
         if(immunizations.size() > 0){

            for(Immunization immunization : immunizations){
               vaccinations.add(new
BasicDBObject("vaccDisease",immunization.getVaccinationProtocolFirstRep().getTarge
tDiseaseFirstRep().getText()).
                     append("brandName", "").
                     append("vaccID",
immunization.getIdentifierFirstRep().getValue()).
                     append("vaccinationDate",
String.valueOf(immunization.getDate())));
            }
         }

         if(conditions.size() > 0){
            DateTimeDt date, endDate;

            for(Condition con : conditions){

               date = (DateTimeDt)con.getOnset();
               endDate = (DateTimeDt)con.getAbatement();

               resolvedProblems.add(new
BasicDBObject("description",con.getCode().getText()).
                     append("problemID",
con.getCode().getCodingFirstRep().getCode()).
                     append("onsetTime", String.valueOf(date.getValue())).
                     append("endDate", String.valueOf(endDate.getValue())).
                     append("resolutionReason", con.getNotes()));
            }
         }

         if(procedures.size() > 0){
            DateTimeDt date;

            for(Procedure proc : procedures){
               date = (DateTimeDt)proc.getPerformed();

               procs.add(new BasicDBObject("description",
proc.getCode().getText()).
                     append("procedureID",
proc.getCode().getCodingFirstRep().getCode()).
                     append("procedureDate", String.valueOf(date.getValue())));
            }
         }

         try{

            FindIterable<Document> iterable = mDatabase.getCollection(COLLECTION).
                  find(new Document("healthcareID",healthcareID));

            if(iterable.iterator().hasNext()){
               mDatabase.getCollection(COLLECTION).updateOne(new
Document("healthcareID",healthcareID),
                     new Document("$set",new
Document("vaccinations",vaccinations).
                           append("resProblems", resolvedProblems).
```

```
                        append("procedures", procs)));
            }else{
               Boolean idExists = true;
               Random randomID;
               int docID = 0;
               FindIterable<Document> idIterable;

               while(idExists){
                  randomID = new Random();
                  docID = randomID.nextInt(999999999);

                  idIterable = mDatabase.getCollection(COLLECTION).
                        find(new Document("_id",docID));

                  if(!idIterable.iterator().hasNext()){
                     idExists = false;
                  }
               }

               Document medHistory_doc = new
Document("_id",docID).append("healthcareID",healthcareID).
                     append("vaccinations",vaccinations).
                     append("resProblems", resolvedProblems).
                     append("procedures", procs);

               mDatabase.getCollection(COLLECTION).insertOne(medHistory_doc);
            }

            return new ResponseEntity<>("OK", HttpStatus.OK);
         }catch(Exception ex){
            System.out.println(ex.getMessage());
            return new ResponseEntity<>("NOT OK", HttpStatus.BAD_REQUEST);
         }
      }else{
         return new ResponseEntity<>("NOT OK", HttpStatus.BAD_REQUEST);
      }

   }

}
```

Ο παρακάτω κώδικας παρουσιάζει τη διαδικασία σύνδεσης με τη βάση και τοποθέτηση των δεδομένων σε «πόρους» FHIR. Πιο συγκεκριμένα του «πόρου» που σχετίζεται με τον εμβολιασμό του ασθενούς

```
@RestController
@RequestMapping(value="/epsostofhir")
public class FHIRImmunization {

   final String COLLECTION = "medical_history";
   final String DATABASE = "hvfhirdb";
   IParser parser;
   MongoDatabase mDatabase;
   boolean OK = true;
```

```
   public FHIRImmunization() {
      try{
         parser = Server.ctx.newJsonParser();
         mDatabase = Server.mClient.getDatabase(DATABASE);
      }catch(Exception ex){
         OK = false;
      }
   }

   @RequestMapping(value="/immunization/{healthcareID}", method=RequestMethod.GET)
   public @ResponseBody List<String>
setImmunization(@PathVariable(value="healthcareID") String healthcareID){

      String jsonString = "";
      List<String> imms = new ArrayList<>();
      ObjectMapper objectMapper = new ObjectMapper();
      objectMapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES,
false);

      if(OK){
         try{

            FindIterable<Document> iterable = mDatabase.getCollection(COLLECTION).
                  find(new Document("healthcareID", healthcareID));

            if(iterable.iterator().hasNext()){

               for(Document id_doc : iterable){
                  jsonString = id_doc.toJson();
               }

               MedicalHistory medicalHistory = objectMapper.readValue(jsonString,
MedicalHistory.class);

               if(medicalHistory.getVaccinations().size() > 0){
                  SimpleDateFormat sdf = new SimpleDateFormat("EEE MMM dd hh:mm:ss
zzz yyyy");
                  //Date onSetDate = null;

                  for(Vaccination vaccination : medicalHistory.getVaccinations()){
                     Immunization immunization = new Immunization();
                     Date onSetDate = null;
                     //set vaccination onset date
                     try{
                        onSetDate = sdf.parse(vaccination.getVaccinationDate());
                        immunization.setDate(new DateTimeDt(onSetDate));
                     }catch(Exception ex){
                        System.out.println(ex.getMessage());
                        immunization.setDate(new DateTimeDt(onSetDate));
                     }

                     //set disease
                     immunization.addVaccinationProtocol().addTargetDisease(new
CodeableConceptDt().setText(vaccination.getVaccDisease()));
                     immunization.setVaccineCode(new
CodeableConceptDt().setText(vaccination.getBrandName()).
                           addCoding(new
```

```
CodingDt().setCode(vaccination.getVaccID())));

                        //add vaccine to list
                        imms.add(parser.encodeResourceToString(immunization));
                    }
                }
            }
         }catch(Exception ex ){
            System.out.println(ex.getMessage());
         }
      }

      return imms;
   }
}
```

Ο παρακάτω κώδικας παρουσιάζει την σύνδεση της Android εφαρμογής με την πλατφόρμα HealthVault για την επιστροφή δεδομένων σχετικά με τον ασθενή όπως όνομα, επώνυμο κ.α.

```
public class PersonalInfoClass {

    private HealthVaultClient hvClient;
    private Record hvRecord;
    private static PersonalDemographics personalDemographics;
    private static BasicDemographicInformation basicDemographics;
    public static String pers_info_errors = "";


    public PersonalInfoClass(HealthVaultClient hvClient, Record record){
        this.hvRecord = record;
        this.hvClient = hvClient;
    }

    @SuppressWarnings("unchecked")
    public void getPersonalInfo(Context context){

hvClient.asyncRequest(hvRecord.getThingsAsync(ThingRequestGroup2.thingTypeQuery(Pe
rsonalDemographics.ThingType)),
                new PersonalInfoCallback(PersonalInfoCallback.PERS_DEMO_CASE,
context));


hvClient.asyncRequest(hvRecord.getThingsAsync(ThingRequestGroup2.thingTypeQuery(Ba
sicDemographicInformation.ThingType)),
                new
PersonalInfoCallback(PersonalInfoCallback.BASIC_DEMO_CASE,context));
    }

    private PersonalDemographics renderPersonalDemographics(List<Thing2>
piThings){

        for (Thing2 persInfo : piThings) {
            PersonalDemographics persDemo = (PersonalDemographics)
persInfo.getData();
            personalDemographics = persDemo;
            break;
```

```
        }

        return personalDemographics;
    }

    public static PersonalDemographics getPersonalDemographics(){
        return personalDemographics;
    }

    private BasicDemographicInformation renderBasicDemographics(List<Thing2>
piThings){

        for (Thing2 basicInfo : piThings) {
            BasicDemographicInformation baseDemo = (BasicDemographicInformation)
basicInfo.getData();
            basicDemographics = baseDemo;
            break;
        }

        return basicDemographics;
    }

    public static BasicDemographicInformation getBasicDemo(){
        return basicDemographics;
    }

    private class PersonalInfoCallback<Object> implements RequestCallback {

        private ProgressDialog progressDialog;
        public static final int PERS_DEMO_CASE = 1;
        public static final int BASIC_DEMO_CASE = 2;

        private int event;
        private Context context;

        public PersonalInfoCallback(int event,Context context){
            this.event = event;
            this.context = context;
            progressDialog = ProgressDialog.show(context, "Loading", "Handling
Personal Information...", true);
        }

        @Override
        public void onError(HVException exception) {
            switch(event){
                case PERS_DEMO_CASE:
                    personalDemographics = new PersonalDemographics();
                    break;
                case BASIC_DEMO_CASE:
                    basicDemographics = new BasicDemographicInformation();
                    break;
            }
            pers_info_errors = exception.getMessage().toString()+"\n";
            progressDialog.dismiss();
        }

        @Override
        public void onSuccess(java.lang.Object o) {
```

```
            switch(event){
                case PERS_DEMO_CASE:
                    personalDemographics =
renderPersonalDemographics(((ThingResponseGroup2) o).getThing());
                    break;
                case BASIC_DEMO_CASE:
                    basicDemographics =
renderBasicDemographics(((ThingResponseGroup2) o).getThing());
                    break;
            }
            progressDialog.dismiss();
        }
    }

}
```

Τέλος ο παρακάτω κώδικας έχει να κάνει με τη χρήση του Spring από την πλευρά του Client για να κάνει REST HTTP Requests

```
try {
    JsonReader jreader = new JsonReader(new
FileReader("C:/Users/Haris/Desktop/Διπλωματική/bioassist_rest_ws/1.json"));
    BioassistPatient bio = new Gson().fromJson(jreader, new
TypeToken<BioassistPatient>() {}.getType());

    ctx = FhirContext.forDstu2();

    restTemplate = new RestTemplate();

    try {
        ResponseEntity<String> responseEntity =
restTemplate.postForEntity("http://localhost:9001/fhirtoepsos/diagnostictests/" +
bio.getSSN(),
                getResultValuesBiochemical(bio), String.class);

        if(responseEntity.getStatusCode().toString().equals("200")){
            ResponseEntity<String> adminResEntity = restTemplate.

getForEntity("http://localhost:9001/fhirtoepsos/administrativedata/"+bio.getSSN()+
"/admindata?healthcareid="+

"&country="+country+"&nature="+typeOfPS+"&author="+author, String.class);

            if(adminResEntity.getStatusCode().toString().equals("200")){
                result.setForeground(Color.GREEN);
                result.setText("Your data were uploaded successfully!");
            }
        }
    } catch (Exception ex) {
        result.setForeground(Color.RED);
        result.setText("Could not upload your data to the main server!");
    }
}catch(FileNotFoundException fnfex){
    result.setText("Not Such File Exists!");
}
```

7. Αναφορές

[1] Robertson, Jordan. “Top 10 Countries Where Doctors Go Digital” (2013). http://www.bloomberg.com/slideshow/2013-06-25/top-10-countries-where-doctors-go-digital.html

[2] Estelrich Anna, Chronaki Catherine, Cangioli Giorgio, Melgara Marcello. “Converging Patient Sumaries: Finding the Common Denominator between the European Patient Summary and the US-Based Continuity of Care Document.”

[3] Ahier, Brian. “FHIR and the future of Interoperability” (2015)

[4] Crockford, Douglas. "The application/json media type for javascript object notation (json)." (2006).

[5] Bray, Tim, et al. "Extensible markup language (XML)." World Wide Web Consortium Recommendation REC-xml-19980210. http://www. w3. org/TR/1998/REC-xml-19980210 16 (1998).

[6] Fielding, Roy, et al. "Hypertext transfer protocol--HTTP/1.1." (1999): 12.

[7] Erl, Thomas. Service-oriented architecture: a field guide to integrating XML and web services. Prentice Hall PTR, 2004.

[8] Lindén, Frederik. "epsos, smart open services for European patients from strategies to services health as the enabler for cross-border healthcare." Infrastructures for Health Care 23 (2009).

[9] Christensen, Erik, et al. "Web services description language (WSDL) 1.1." (2001).

[10] Dolin, Robert H., et al. "HL7 clinical document architecture, release 2." Journal of the American Medical Informatics Association 13.1 (2006): 30-39.

[11] Battle, Robert, and Edward Benson. "Bridging the semantic Web and Web 2.0 with representational state transfer (REST)." Web Semantics: Science, Services and Agents on the World Wide Web 6.1 (2008): 61-69.

[12] Graham, Ian S. The HTML sourcebook. John Wiley & Sons, Inc., 1995.

[13] Briggs, Owen, et al. Cascading Style Sheets. 2004.

[14] Hardt, Dick. "The OAuth 2.0 authorization framework." (2012).

[15] Nottingham, Mark, and Robert Sayre. "The atom syndication format." (2005).

[16] Dickinson, Gary, Linda Fischetti, and Sam Heard. "HL7 EHR System Functional Model Draft Standard for Trial Use." Health Level 7 (2004).

[17] Kasthurirathne, Suranga N., et al. "Enabling Better Interoperability for HealthCare: Lessons in Developing a Standards Based Application Programing Interface for Electronic Medical Record Systems." Journal of medical systems 39.11 (2015): 1-8.

[18] Sunyaev, Ali, et al. "Evaluation framework for personal health records: Microsoft HealthVault vs. Google Health." System Sciences (HICSS), 2010 43rd Hawaii International Conference on. IEEE, 2010.

[19] Furnell, Steven. "An assessment of website password practices." *Computers & Security* 26.7 (2007): 445-451.

[20] Recordon, David, and Drummond Reed. "OpenID 2.0: a platform for user-centric identity management." Proceedings of the second ACM workshop on Digital identity management. ACM, 2006.

[21] Dolin, Robert H., Gay Giannone, and Gunther Schadow. "Enabling joint commission medication reconciliation objectives with the HL7/ASTM Continuity of Care Document standard." AMIA. 2007.

[22] Ferranti, Jeffrey M., et al. "The clinical document architecture and the continuity of care record: a critical analysis." Journal of the American Medical Informatics Association 13.3 (2006): 245-252.

[23] Banker, Kyle. MongoDB in action. Manning Publications Co., 2011.

[24] Leavitt, Neal. "Will NoSQL databases live up to their promise?." *Computer* 43.2 (2010): 12-14.

[25] Lawton, George. "Developing software online with platform-as-a-service technology." Computer 41.6 (2008): 13-15.

[26] Chu, Cheng, et al. "Map-reduce for machine learning on multicore." Advances in neural information processing systems 19 (2007): 281.

[27] Date, Chris J., and Hugh Darwen. A Guide To Sql Standard. Vol. 3. Reading: Addison-Wesley, 1997.

[28] Brickley, Dan, and Ramanathan V. Guha. "Resource Description Framework (RDF) Schema Specification 1.0: W3C Candidate Recommendation 27 March 2000." (2000).

[29] Curbera, Francisco, et al. "Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI." IEEE Internet computing 6.2 (2002): 86.

[30] Paolucci, Massimo, et al. "Importing the semantic web in UDDI." Web Services, E-Business, and the Semantic Web. Springer Berlin Heidelberg, 2002. 225-236.

[31] Masinter, Larry, Tim Berners-Lee, and Roy T. Fielding. "Uniform resource identifier (URI): Generic syntax." (2005).

[32] Freed, Ned, and Nathaniel Borenstein. "Multipurpose internet mail extensions (MIME) part one: Format of internet message bodies." (1996).

[33] Liskin, Olga, Leif Singer, and Kurt Schneider. "Teaching old services new tricks: adding HATEOAS support as an afterthought." Proceedings of the Second International Workshop on RESTful Design. ACM, 2011. [34]

[34] Board, Advisory RSS. "RSS 2.0 Specification." (2007).

[35] Thai, Thuan, and Hoang Lam. . NET framework essentials. " O'Reilly Media, Inc.", 2003.

[36] David Hurth. “JSON Beats XML, or AJAJ vs AJAX

[37] Garrett, Jesse James. "Ajax: A new approach to web applications." (2005).

[38] Gossman, William E., and Peter J. Hartmaier. "Mobility extended telephone application programming interface and method of use." U.S. Patent No. 6,181,935. 30 Jan. 2001.