ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Ανάπτυξη λογισμικού για χρήση μεταξύ ιατρού και ασθενούς

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της/του

Βροχίδη Πρόδρομου

Επιβλέπων : Καβαλλιεράτου Εργίνα

Μέλη εξεταστικής επιτροπής:

Σάμος, Σεπτέμβριος 2022

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πρόλογος και ευχαριστίες

Πρώτη παράγραφος

Επόμενη παράγραφος....

του

ΒΡΟΧΙΔΗ ΠΡΟΔΡΟΜΟΥ

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πίνακας περιεχομένων

Περίληψη

Στην παρούσα διπλωματική εργασία, περιγράφεται και υλοποιείται η ανάπτυξη μίας εφαρμογής, σκοπός της οποίας είναι η χρήση της σε κάποια κλινική ή ιατρείο, για τον προγραμματισμό ραντεβού μεταξύ ιατρών και ασθενών. Σκοπός της εργασίας, είναι η διευκόλυνση τον χρηστών στους οποίους απευθύνεται, δηλαδή γιατρούς και ασθενείς, ως προς τον προγραμματισμό τον ραντεβού αυτών, αλλά και η απλούστευση και ψηφιοποίηση των απαιτούμενων ενεργειών. Στόχος ήταν η ανάπτυξη ενός φιλικού και εύχρηστου προς κάθε χρήστη λογισμικού. Για τον υλοποίηση των λειτουργιών της χρησιμοποιήθηκε αντικειμενοστραφής προγραμματισμός.

Οι βασικές λειτουργίες της εφαρμογής, είναι ο κανονισμός ραντεβού μεταξύ ιατρών που υπάρχουν στο σύστημα της εφαρμογής και των ασθενών που την χρησιμοποιούν, η αποθήκευση των ραντεβού που έχουν προηγηθεί με κάθε ασθενή και με κάθε γιατρό, αλλά και η διατήρηση ενός ψηφιακού ιατρικού φακέλου για κάθε ασθενή που εγγράφεται στην βάση δεδομένων της εφαρμογής. Η σημαντικότερη διευκόλυνση που μπορεί να προσφέρει η εφαρμογή, είναι ο απλός, εύκολος και γρήγορος προγραμματισμός των ραντεβού, χωρίς την απαίτηση φυσικής παρουσίας, ούτε του ασθενή ούτε και του γιατρού. Ο ψηφιακός ιατρικός φάκελος αποτελεί μια ακόμη πολύ σημαντική προσθήκη, καθώς δίνεται δυνατότητα εύκολης πρόσβασης σε αυτόν από τον ασθενή, χωρίς την χρήση χαρτιού και δίχως την απαίτηση της φυσικής του παρουσίας ώστε να λάβει τις διαγνώσεις που αποτελούν τον φάκελό του.

Τέλος, στο θεωρητικό κομμάτι της διπλωματικής, αρχικά παρουσιάζονται παρόμοια λογισμικά που ήδη χρησιμοποιούνται στο εξωτερικό αλλά και στην Ελλάδα, ακολουθεί μια ανάλυση των τεχνολογιών και των εργαλείων που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής HospitApp και τελικά παρουσιάζονται οι λειτουργίες της εφαρμογής, με μέρος του κώδικα που γράφτηκε γι' αυτή, όπως και μια περιγραφή του τρόπου λειτουργίας της βάσης δεδομένων που χρησιμοποιείται για την αποθήκευση των δεδομένων που απαιτούνται για την λειτουργία της.

Λέξεις Κλειδιά: *Εφαρμογή, Android, Java, Κλινική, Ιατρείο, Ραντεβού*

Abstract

In this thesis, the development of an application is described and implemented with the purpose to be used in a clinic or infirmary, for scheduling appointments between doctors and patients. The main goal is to facilitate the users to whom it is addressed, i.e. doctors and patients, in terms of planning their appointments, but also to simplify and digitize the required actions. The goal was to develop a user-friendly and easy-to-use software. Object-oriented programming was used to implement its functions.

The main functions of the application are the regulation of appointments between doctors stored in the application's system and the patients who use it, the storage of previous appointments with each patient and with each doctor, but also the maintenance of a digital medical file for each patient registering in the application database. The most important facility that the application can offer is the simple, easy and fast scheduling of appointments, without the requirement of physical presence, neither of the patient nor of the doctor. The digital medical file is another very important addition, as it allows easy access to it by the patient, without the use of paper and without the requirement of his physical presence in order to receive the diagnoses that make up his file.

Finally, in the theoretical part of the thesis, similar software that are already used abroad and in Greece are first presented, followed by an analysis of the technologies and tools used to develop the HospitApp application and finally, the functions of the application are presented, with part of the code written for it, as well as a description of how the database is used to store the data required for its operation works.

Keywords: Application, Android, Java, Clinic, Infirmary, Appointment

1 *Εισαγωγή*

Στη σύγχρονη εποχή, είναι γεγονός πως η ραγδαία ανάπτυξη που έχει σημειωθεί στην επιστήμη της Πληροφορικής βοηθάει στις καθημερινές μας ανάγκες, κάνοντας τις πιο εύκολες, γρήγορες και λειτουργικές απ' ότι στο παρελθόν. Στην παρούσα διπλωματική θα γίνει μια έρευνα για το πως η Πληροφορική και συγκεκριμένα η ανάπτυξη λογισμικού, μπορεί να εμπλακεί στον τομέα της υγείας και να διευκολύνει ενέργειες που μέχρι πρότινος απαιτούσαν πολύ περισσότερο χρόνο. Το αντικείμενο της διατριβής θα είναι η ανάπτυξη ενός λογισμικού, προς χρήση ιατρών και ασθενών. Σκοπός της ανάπτυξης του λογισμικού αυτού, είναι η καλύτερη διαχείριση των ραντεβού σε ένα νοσοκομείο, ιατρείο ή κλινική, η παροχή της δυνατότητας διατήρησης αρχείου με όλα τα προηγούμενα αλλά και επερχόμενα ραντεβού κάθε ασθενή με τον αντίστοιχο ιατρό, κλινική ή νοσοκομείο, αλλά και αρχείο με το σύνολο των γνωματεύσεων που έχουν δοθεί στους ασθενείς.

Η ανάπτυξη ενός λογισμικού για την διαχείριση των ραντεβού ασθενών και ιατρών, την καταχώρησή τους σε αυτό αλλά και την καταχώρηση όλων των γνωματεύσεων του ιατρού, θα μπορέσει δυνητικά να βοηθήσει για την ευκολότερη και αμεσότερη επικοινωνία μεταξύ των δύο. Με τα μέχρι τώρα δεδομένα στην χώρα μας, εάν κάποιος είναι άρρωστος και θέλει να επισκεφθεί έναν γιατρό, αυτός ή αυτή πρέπει να επισκεφθεί το νοσοκομείο και να περιμένει μέχρι ο γιατρός να είναι διαθέσιμος. Ο ασθενής περιμένει επίσης σε μια ουρά ενώ έχει κλείσει το ραντεβού του. Εάν ο γιατρός ακυρώσει το ραντεβού για κάποιον έκτακτο λόγο, ο ασθενής δεν μπορεί να γνωρίζει την ακύρωση, εκτός εάν ή έως ότου αυτός ή αυτή επισκεφθεί το νοσοκομείο/ιατρείο. Η δυνατότητα άμεσης κλήσης ραντεβού μεταξύ τους μέσω του λογισμικού και η καταχώρηση του ραντεβού αυτού αλλά και της διάγνωσης που θα προκύψει, μπορούν να βοηθήσουν αισθητά και τον ιατρό αλλά και τον ασθενή. Ο ιατρός απ΄ την μεριά του, θα μπορεί να έχει σε ψηφιακή μορφή (βάση δεδομένων) όλες τις προηγούμενες διαγνώσεις, συνταγές και συνεδρίες που είχε με τον

ασθενή, οργανωμένες και άμεσα προσβάσιμες σε περίπτωση που χρειαστούν για κάποια άλλη διάγνωση του ίδιου ασθενή, διατηρώντας έτσι ένα "αρχείο" για τον καθένα ξεχωριστά. Από την μεριά του ασθενούς, οι διευκολύνσεις που μπορεί να προσφέρει ένα τέτοιο λογισμικό, αφορούν τον κανονισμό ενός ραντεβού, ο οποίος θα μπορεί να γίνεται ψηφιακά και χωρίς την προϋπόθεση της φυσικής του παρουσίας στον ιατρείο, το νοσοκομείο ή το γραφείο του ιατρού με τον οποίο θέλει το ραντεβού, γλιτώνοντας συνεπώς χρόνους αναμονής σε ουρές ή την περίπτωση μιας έκτακτης ακύρωσης του ραντεβού. Επίσης ο ασθενής θα έχει και αυτός στην κατοχή του ένα αρχείο με όλα τα ραντεβού που είχε με τον εκάστοτε ιατρό, τις γνωματεύσεις που πήρε από αυτόν αλλά και μια ψηφιακή ατζέντα με τα επερχόμενα ραντεβού του. Η κατοχή και η δυνατότητα άμεσης ψηφιακής πρόσβασης σε όλες τις ιατρικές γνωματεύσεις και διαγνώσεις που έχει λάβει ο ασθενής, μπορούν να φανούν χρήσιμες και σε περιπτώσεις όπου ο ασθενής βρίσκεται σε κάποιο άλλο ιατρείο, νοσοκομείο ή κλινική, όπου ίσως οι καταχωρήσεις φανούν βοηθητικές στην περίσταση εκείνη, διασφαλίζοντας έτσι χρόνο στον ασθενή.

Ιδιαίτερη βαρύτητα κατά την ανάπτυξη ενός τέτοιου πληροφοριακού συστήματος, θα πρέπει να δοθεί στην ασφάλεια των προσωπικών δεδομένων των χρηστών και κυρίως των ασθενών, αφού θα δημιουργείται ουσιαστικά ένας ψηφιακός ιατρικός φάκελος γι' αυτούς.

Υπάρχουν αρκετές διαφορές μεταξύ ενός απτού ιατρικού φακέλου και ενός ψηφιακού. Τα πλεονεκτήματα ενός απτού/χειρόγραφου ιατρικού φακέλου, περιορίζονται στην μεταφορά του χωρίς την χρήση ηλεκτρονικού υπολογιστή και στο γεγονός ότι δεν απαιτείται κάποια ειδική εκπαίδευση του ιατρικού προσωπικού. Σε αντίθεση με τον απτό, ένας ψηφιακός ιατρικός φάκελος, μπορεί να χρησιμοποιηθεί, από περισσότερους του ενός χρήστες ταυτόχρονα. Αυτό σημαίνει πως αν ο ίδιος ασθενής χρειαστεί να επισκεφθεί παραπάνω από έναν γιατρό μπορεί να δώσει πρόσβαση στον φάκελό του σε όλους και έτσι οι γνωματεύσεις τους να συνταχθούν γρηγορότερα. Ακόμη, κατά την ενημέρωση του φακέλου του ασθενούς από ένα γιατρό σε κάποιο νοσοκομείο μπορεί να εξοικονομηθεί και πάλι χρόνος, λόγω της ψηφιοποίησης της γνωμάτευσης και στην περίπτωση που ο ασθενής χρειάζεται τα αποτελέσματα της εν λόγω γνωμάτευσης για να επισκεφθεί κάποιον άλλον γιατρό άλλης ειδικότητας ή για να κάνει κάποιες εξετάσεις που απαιτούν τα αποτελέσματα αυτής, να μην χρειαστεί η φυσική του παρουσία για την παραλαβή τους και κατά συνέπεια να διευκολυνθεί όλη η διαδικασία. Ένα ακόμη σημαντικό πλεονέκτημα της χρήσης ψηφιακού ιατρικού φακέλου, είναι το ότι δεν απαιτείται κάποιος φυσικός χώρος αποθήκευσης. Τα ιατρικά δεδομένα των ασθενών υπάρχουν ψηφιακά κι έτσι ελαχιστοποιείται και ο κίνδυνος απώλειας των παλαιότερων λόγω έλλειψης χώρου ή κακής συντήρησής τους. Ο μεγαλύτερος κίνδυνος που υπάρχει στην διατήρηση ενός ηλεκτρονικού ιατρικού φακέλου, είναι αυτός της ασφάλειας των προσωπικών δεδομένων και του ιατρικού απορρήτου των ασθενών.

1.1 Τι υπάρχει σήμερα

Σύμφωνα με το άρθρο που δημοσιεύτηκε στο "Διεθνές Αραβικό Συμβούλιο για την Πληροφοριακή" με τίτλο "Medical patient appointments management using smart software system in UAE"[1], τα διαδικτυακά ραντεβού αυξάνονται περισσότερο με την ανάπτυξη της τεχνολογίας και του Διαδικτύου και επομένως η ανάγκη για την δυνατότητα λήψης ραντεβού μέσω λογισμικού και στον τομέα της υγείας γίνεται όλο και πιο επιτακτική. Ο τομέας της υγειονομικής περίθαλψης είναι ένας από τους σημαντικότερους τομείς που χρειάζεται περισσότερη προσοχή λόγω της υψηλής ευαισθησίας, επομένως, η ανάπτυξη τέτοιων συστημάτων είναι πολύ περίπλοκη και πρέπει να είναι ακριβέστερη με πολύ υψηλό απόρρητο. Για την δημιουργία διαδικτυακών συστημάτων διαχείρισης ραντεβού, οι προγραμματιστές επικεντρώθηκαν στα προβλήματα της κανονικής διαδικασίας ραντεβού, ώστε να μπορούν να σχεδιάσουν εφαρμογές και ιστότοπους που θα ήταν λύσεις για αυτά τα προβλήματα. Παρόλο που τα συστήματα αυτά δεν είναι ίδια μεταξύ τους, υπάρχουν κοινά χαρακτηριστικά, όπως και κάποιες διαφορές. Τα ακόλουθα είναι τρία τέτοιου είδους συστήματα που αναπτύχθηκαν από διαφορετικούς ανθρώπους, σε διαφορετικά μέρη σε όλον τον κόσμο:

1.1.1 Online Polyclinic

«Στόχος αυτής της διατριβής ήταν να αναπτυχθεί ένα διαδικτυακό σύστημα ραντεβού και διαχείρισης ιατρικής βάσης δεδομένων, στο Πακιστάν. Ο σκοπός της εφαρμογής αυτής, ήταν να δημιουργήσει ένα σύστημα μέσω του οποίου ένας ασθενής μπορεί εύκολα να συγκρίνει, να επιλέξει και να πραγματοποιήσει ένα ραντεβού με κάποιον γιατρό από το σπίτι. Δεύτερος στόχος ήταν η αντικατάσταση του μη αυτόματου ιατρικού συστήματος διατήρησης αρχείων με το ηλεκτρονικό σύστημα διαχείρισης βάσεων δεδομένων. Η εφαρμογή υλοποιήθηκε με επιτυχία χρησιμοποιώντας γλώσσες προγραμματισμού όπως HTML, CSS και JavaScript στην πλευρά του πελάτη, ενώ PHP και MySQL στην πλευρά του διακομιστή.» [2]

1.1.2 Mr Doc

«Αυτό το σύστημα εντοπίστηκε αφού μια ομάδα πέντε προγραμματιστών εντόπισε τα προβλήματα της μέχρι τότε διαδικασίας λήψης ραντεβού με νοσοκομεία ή γιατρούς και αποφάσισε να βρει λύσεις, παρέχοντας ένα σύστημα που υποστηρίζει το κλείσιμο ραντεβού κάνοντάς το μια διαδικτυακή αυτοματοποιημένη διαδικασία.

Αυτή η εφαρμογή μπορεί να χρησιμοποιηθεί και με άλλα βασισμένα σε ραντεβού συστήματα. Η εφαρμογή για κινητό (android), δέχεται ραντεβού αποθηκεύοντας τις εγγραφές τους στο ημερολόγιο του τηλεφώνου το οποίο είναι συγχρονισμένο με το ημερολόγιο της Google. Ο χρήστης λαμβάνει μια ειδοποίηση την προκαθορισμένη ώρα πριν από το ραντεβού. Επομένως, ένα σύστημα με εύκολα βήματα και μερικές χρήσιμες λειτουργίες, όπως υπενθυμίσεις, τοποθεσία και ειδοποιήσεις είναι ένα εξαιρετικό εργαλείο

για να βοηθήσει όλους τους ανθρώπους στην απλούστευση της διαδικασίας λήψης ραντεβού.» [3]

1.1.3 Medicus

«Το σύστημα αυτό, αναπτύχθηκε από μια ομάδα προγραμματιστών που είχαν την ιδέα να κάνουν τη διαδικασία λήψης ραντεβού ευκολότερη για τους ασθενείς και επικεντρώθηκαν στη μείωση της προσπάθειας που πρέπει να κάνει ένας ασθενής για να κλείσει ραντεβού με γιατρό. «Η κύρια ιδέα αυτής της εργασίας είναι να παρέχει ευκολία και άνεση στους ασθενείς και να επιλύσει επίσης τα προβλήματα που αντιμετωπίζουν οι ασθενείς κατά τη συνάντηση». Στην πραγματικότητα, αυτό το σύστημα παρέχει μια σειρά λειτουργιών όπως υπενθυμίσεις για ραντεβού, εμφάνιση της τοποθεσίας του νοσοκομείου ή της κλινικής και ειδοποιήσεις σχετικά με οποιαδήποτε αλλαγή μπορεί να προκύψει στο ραντεβού, όπως ακύρωση ή αλλαγή ώρας, που βοηθούν τους ασθενείς.»[4]

1.1.4 HU fundacion Jimenez diaz application

Το λογισμικό αυτό χρησιμοποιείται στα νοσοκομεία της Ισπανίας από ασθενείς και γιατρούς, για τον κανονισμό και την διαχείριση των ραντεβού τους. Χρησιμοποιώντας την εφαρμογή, η οποία είναι διαθέσιμη στο Google Play, ο ασθενής μπορεί να κλείσει από αυτήν τα ραντεβού του στο νοσοκομείο με τον γιατρό που επιθυμεί. Ακόμη η εφαρμογή δίνει την δυνατότητα αποθήκευσης αρχείου με όλες τις γνωματεύσεις που έχει λάβει ο ασθενής. Τα ισπανικά νοσοκομεία αποτελούν μια ιδιαίτερη περίπτωση καθώς ο πληθυσμός της Ισπανίας είναι ένας από τους πιο μεγάλους ηλικιακά στην Ευρώπη. Ποσοστό 17,95% του πληθυσμού της είναι άνω των 65 ετών και η μέση ηλικία της χώρας είναι τα 42,7 έτη. Το μέσο προσδόκιμο ζωής είναι τα 81,8 έτη και η χώρα έχει ένα από τα υψηλότερα προσδόκιμα γυναικείας ζωής στην Ευρώπη, τα 84,9 έτη. Καταλαβαίνουμε λοιπόν πως λόγω της μεγάλης μέσης ηλικίας του πληθυσμού της χώρας, η ζήτηση για νοσηλεία, χειρουργική επέμβαση και γενικότερες ιατρικές υπηρεσίες είναι αυξημένη, πράγμα που σημαίνει πως υπάρχει μια αρκετά μεγάλη λίστα αναμονής. Ένα τέτοιο λογισμικό λοιπόν, εκτός της διευκόλυνσης που παρέχει σε ασθενείς και γιατρούς, μπορεί να βοηθήσει και στον καλύτερο προγραμματισμό όλων των ραντεβού σε ένα νοσοκομείο αλλά και στην αποσυμφόρηση των νοσοκομείων από ουρές αναμονής.

1.2 Χρήση νοσοκομειακών λογισμικών στην Ελλάδα

Τρία από τα νοσοκομειακά λογισμικά που χρησιμοποιούνται στην χώρα μας, έχουν να κάνουν με την διατήρηση ηλεκτρονικού ιατρικού αρχείου ή EMR (Electronic Medical Record).

1.2.1 OpenEMR

Εικόνα 1.1 – Λογότυπο του OpenEMR

Το open E.M.R είναι ένα πλήρες πρόγραμμα οργάνωσης και διαχείρισης αρχείων ασθενών. Το λογισμικό αυτό είναι μια ιατρική εφαρμογή λογισμικού η οποία δίνει στους χρήστες, το ιατρικό προσωπικό, το προσωπικό των γραφείων και το νοσηλευτικό προσωπικό, την δυνατότητα να διατήρησης αρχείου με τα ιατρικά δεδομένα των ασθενών. Το λογισμικό ενώνει την ιατρική κλινική διαχείριση και τα ηλεκτρονικά ιατρικά αρχεία. Περιλαμβάνει επίσης κι αλλές δυνατότητες όπως, κανονισμός ραντεβού, κλινικός έλεγχος, εξετάσεις, δυνατότητα αποθήκευσης εικόνων, ειδοποίηση των συμμετεχόντων με mail, fax, δυνατότητα πληρωμών. Αποτελεί λογισμικό ανοιχτού κώδικα και τρέχει στα περισσότερα λειτουργικά συστήματα και διανέμεται δωρεάν από το www.sourceforge.net.

1.2.2 GNU Health

Εικόνα 1.2 – Λογότυπο του GNUHealth

Το GNU Health είναι ένα ελεύθερο λογισμικό (GNU GPL). Ο τρόπος σχεδίασής του είναι κατάλληλος ώστε να είναι δυνατή η εγκατάστασή του σε διάφορα λειτουργικά συστήματα (GNU / Linux, FreeBSD, MS Windows) και τα διαφορετικά συστήματα διαχείρισης βάσεων δεδομένων (PostgreSQL). Η γλώσσα που έχει χρησιμοποιηθεί για τον προγραμματισμό του ειναι η Python και το πλαίσιο Tryton. Οι βασικές λειτουργίες του GNUHealth είναι η εξέταση ασθενούς και λήψη του ιστορικού του ψηφιακά, η διαχείριση ασθενών (δημιουργία νέου ασθενούς, εκτίμηση ιστορικού, κ.τ.λ.), η διαχείριση ιατρών, η διαχείριση εργαστηρίου, η παροχή πληροφοριών φαρμάκων, η διαχείρηση της αποθήκης ιατρο-νοσηλευτικού υλικού και προμηθειών και η οικονομικο-λογιστική διαχείριση του νοσοκομείου. Το σύστημα αυτό χρησιμοποιείται από το νοσοκομείο Παπαγεωργίου στην Αθήνα αλλά και σε άλλες χώρες όπως Φιλιππίνες (Marcelo-Padilla), Αργεντινή (Dr Onativia και San Vicente), Ινδονησία (Mojowarno Christian, Griya Waluya και Griya Husada), Ζάμπια (Victoria), Τανζανία (Masaki International), Νιγηρία (Peerless), Κατάρ (Sharq Medical Center) και Μπανγκλαντές (Chittangong).

1.2.3 EMR module της SAP

Το σύστημα αυτό χρησιμοποιείται από τους ασθενείς διευκολύνοντας την ανταλλαγή πληροφοριών μεταξύ των ιατρών, νοσηλευτών, φαρμακείων και ασθενών με σκοπό την παροχή πληροφοριών και υπηρεσιών στην επικοινωνία των ως άνω ατόμων και τμημάτων των υγειονομικών μονάδων. Οι βασικές λειτουργίες του λογισμικού αυτού είναι η γρήγορη ανάλυση κλινικών και λειτουργικών δεδομένων, η παροχή εξατομικευμένων υπηρεσιών επικοινωνίας στους ασθενείς και η διαχείριση εσόδων και εξόδων της κλινικής μονάδας.

1.3 Στόχος

Αντικείμενο της παρούσας διπλωματικής, είναι η ανάπτυξη μιας εφαρμογής για smartphones, η οποία θα δίνει σε γιατρούς και ασθενείς την δυνατότητα προγραμματισμού ραντεβού μεταξύ τους, την διατήρηση αρχείου με τα προηγούμενα και επερχόμενα ραντεβού τους και την αρχειοθέτηση ενός ψηφιακού ιατρικού φακέλου για κάθε ασθενή, ο οποίος θα περιλαμβάνει τις διαγνώσεις που αυτός έχει λάβει. Βασική λειτουργία της εφαρμογής είναι η μεταφόρτωση αρχείων διαγνώσεων και συνταγογραφήσεων από τους γιατρούς, σε μια βάση δεδομένων, στον προσωπικό φάκελο κάθε ασθενή, στον οποίο πρόσβαση θα έχει μόνο ο τελευταίος. Η εφαρμογή αναπτύχθηκε για λειτουργικό σύστημα Android, με σκοπό την κάλυψη μεγαλύτερου εύρους χρηστών. Στόχος λοιπόν της εργασίας, είναι η διευκόλυνση των χρηστών της εφαρμογής (ασθενών και γιατρών), όσον αφορά τον προγραμματισμό των μεταξύ τους ραντεβού και την εύκολη προσπέλαση προηγούμενων διαγνώσεων.

2 *Τεχνολογίες που χρησιμοποιήθηκαν*

Παρακάτω παρουσιάζονται όλα τα εργαλεία και οι γλώσσες προγραμματισμού που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής.

2.1 Java

Εικόνα 2.1 - Λογότυπο της Java

«Η γλώσσα προγραμματισμού Java™ είναι μια γλώσσα γενικής χρήσης, ταυτόχρονης, βασισμένης σε κλάσεις, αντικειμενοστραφής. Είναι σχεδιασμένη να είναι απλή αρκετά ώστε οι προγραμματιστές να μπορούν να επιτύχουν ευχέρεια στη γλώσσα. Η γλώσσα προγραμματισμού Java σχετίζεται με τη C και τη C++, αλλά είναι οργανωμένη μάλλον διαφορετικά, με διάφορες πτυχές της C και της C++ να έχουν παραλειφθεί και να περιλαμβάνονται μερικές ιδέες από άλλες γλώσσες. Προορίζεται να είναι μια γλώσσα παραγωγής, όχι μια γλώσσα έρευνας, και έτσι, όπως ο C. A. R. Hoare που προτείνεται στην κλασική του εργασία για το γλωσσικό σχέδιο, ο σχεδιασμός της έχει αποφύγει να συμπεριλάβει νέα και μη δοκιμασμένα χαρακτηριστικά.
Η γλώσσα προγραμματισμού Java είναι έντονα πληκτρολογημένη. Αυτή η προδιαγραφή κάνει ξεκάθαρη διάκριση μεταξύ των σφαλμάτων χρόνου μεταγλώττισης που μπορούν και πρέπει να ανιχνευθούν στον χρόνο μεταγλώττισης και αυτά που εμφανίζονται κατά το χρόνο εκτέλεσης. Ο χρόνος μεταγλώττισης συνήθως αποτελείται από την μετάφραση προγραμμάτων σε μια ανεξάρτητη από την μηχανή αναπαράσταση κώδικα byte. Οι δραστηριότητες χρόνου εκτέλεσης περιλαμβάνουν τη φόρτωση και τη σύνδεση των κλάσεων που απαιτούνται για την εκτέλεση ενός προγράμματος, την προαιρετική δημιουργία κώδικα μηχανής και δυναμική βελτιστοποίηση του προγράμματος και την πραγματική εκτέλεση του προγράμματος. Η γλώσσα προγραμματισμού Java είναι μια γλώσσα σχετικά υψηλού επιπέδου. Περιλαμβάνει αυτόματη διαχείριση αποθήκευσης, συνήθως χρησιμοποιώντας έναν συλλέκτη σκουπιδιών, για αποφυγή προβλημάτων ασφάλειας της ρητής κατανομής (όπως στην ελεύθερη C ή τη διαγραφή της C++). Υψηλής απόδοσης garbage-collected υλοποιήσεις, μπορεί να έχουν περιορισμένες παύσεις για την υποστήριξη του προγραμματισμού συστημάτων και εφαρμογών σε πραγματικό χρόνο. Η γλώσσα δεν περιλαμβάνει τυχόν μη ασφαλείς κατασκευές, όπως προσβάσεις σε πίνακα χωρίς έλεγχο ευρετηρίου, αφού τέτοιες μη ασφαλείς κατασκευές θα προκαλούσαν ένα πρόγραμμα να συμπεριφέρεται με απροσδιόριστο τρόπο. Η γλώσσα προγραμματισμού Java συνήθως μεταγλωττίζεται στην κωδικοποιημένη εντολή bytecoded και δυαδική μορφή που ορίζεται στο The Java™ Virtual Machine Specification, Java
Έκδοση SE 7.» [5]

2.1.1 Η εικονική μηχανή της Java

«Η εικονική μηχανή Java είναι ο ακρογωνιαίος λίθος της πλατφόρμας Java. Είναι το στοιχείο της τεχνολογίας που είναι υπεύθυνο για την ανεξαρτησία του υλικού και του λειτουργικού συστήματος, το μικρό μέγεθος του μεταγλωττισμένου κώδικα και την ικανότητά του να προστατεύει τους χρήστες από κακόβουλα προγράμματα. Η εικονική μηχανή της Java είναι μια αφηρημένη υπολογιστική μηχανή. Σαν πραγματικός υπολογιστής μηχανή, έχει ένα σύνολο εντολών και χειρίζεται διάφορες περιοχές μνήμης κατά το χρόνο εκτέλεσης. Είναι αρκετά συνηθισμένο να υλοποιείται μια γλώσσα προγραμματισμού χρησιμοποιώντας μια εικονική μηχανή: η πιο γνωστή εικονική μηχανή μπορεί να είναι η μηχανή P-Code του UCSD Πασκάλ. Η πρώτη πρωτότυπη υλοποίηση της Java Virtual Machine, έγινε στη Sun H Microsystems, Inc., μιμήθηκε το σύνολο εντολών Java Virtual Machine στο λογισμικό φιλοξενείται από μια φορητή συσκευή που έμοιαζε με ένα σύγχρονο Personal Digital Βοηθός (PDA). Οι τρέχουσες υλοποιήσεις της Oracle μιμούνται το εικονική μηχανή της Java σε φορητές συσκευές, επιτραπέζιους

υπολογιστές και συσκευές διακομιστή, αλλά η εικονική μηχανή Java δεν προϋποθέτει κάποια συγκεκριμένη τεχνολογία υλοποίησης, υλικό κεντρικού υπολογιστή ή λειτουργικό σύστημα υποδοχής. Δεν ερμηνεύεται εγγενώς, αλλά μπορεί εξίσου καλά να υλοποιείται με τη μεταγλώττιση του συνόλου εντολών της σε αυτό μιας CPU πυριτίου. Μπορεί επίσης να υλοποιηθεί σε μικροκώδικα ή απευθείας σε πυρίτιο. Η εικονική μηχανή Java δεν γνωρίζει τίποτα για τη γλώσσα προγραμματισμού Java, μόνο μιας συγκεκριμένης δυαδικής μορφής, τη μορφή αρχείου κλάσης. Ένα αρχείο κλάσης περιέχει Java οδηγίες εικονικής μηχανής (ή bytecode) και έναν πίνακα συμβόλων, καθώς και άλλες βοηθητικές πληροφορίες. Για λόγους ασφάλειας, η Java Virtual Machine επιβάλλει ισχυρούς συντακτικούς και δομικούς περιορισμούς στον κώδικα ενός αρχείου κλάσης. Ωστόσο, οποιαδήποτε γλώσσα με λειτουργικότητα που μπορεί να εκφραστεί με όρους έγκυρου αρχείου κλάσης μπορεί να φιλοξενηθεί από την εικονική μηχανή της Java.» [6]

2.2 XML

Εικόνα 2.2 - Λογότυπο της XML

«Η XML αναπτύχθηκε από μια ομάδα εργασίας XML (αρχικά γνωστή ως SGML Editorial Review Board) ιδρύθηκε υπό την αιγίδα της Κοινοπραξίας του Παγκόσμιου Ιστού (W3C) το 1996. Πρόεδρος της ήταν ο Jon Bosak της Sun Microsystems με την ενεργή συμμετοχή μιας Ομάδας Ειδικού Ενδιαφέροντος XML (παλαιότερα γνωστή ως SGML Working Group) που οργανώθηκε επίσης από το W3C. Η ιδιότητα μέλους του XML Working Η ομάδα δίνεται σε παράρτημα. Ο Dan Connolly υπηρέτησε ως επαφή της Ομάδας Εργασίας με το W3C. Οι στόχοι σχεδιασμού για την XML είναι:

1. Η XML θα μπορεί να χρησιμοποιηθεί άμεσα μέσω του Διαδικτύου.
2. Η XML θα υποστηρίζει μεγάλη ποικιλία εφαρμογών.
3. Η XML θα είναι συμβατή με την SGML.

4. Θα είναι εύκολο να γραφτούν προγράμματα που επεξεργάζονται έγγραφα XML.
5. Ο αριθμός των προαιρετικών δυνατοτήτων στο XML πρέπει να διατηρείται στο απόλυτο ελάχιστο, ιδανικά μηδέν.
6. Τα έγγραφα XML πρέπει να είναι ευανάγνωστα από τον άνθρωπο και εύλογα σαφή.
7. Το σχέδιο XML πρέπει να προετοιμαστεί γρήγορα.
8. Ο σχεδιασμός της XML πρέπει να είναι επίσημος και συνοπτικός.
9. Τα έγγραφα XML θα είναι εύκολο να δημιουργηθούν.
10. Η σκληρότητα στη σήμανση XML είναι ελάχιστης σημασίας.

Αυτή η προδιαγραφή, μαζί με τα σχετικά πρότυπα (Unicode [Unicode] και ISO/IEC 10646 [ISO/IEC 10646] για χαρακτήρες, Internet RFC 3066 [IETF RFC 3066] για ετικέτες αναγνώρισης γλώσσας, ISO 639 [ISO 639] για τους κωδικούς ονομάτων γλώσσας και το ISO 3166 [ISO 3166] για τους κωδικούς ονομάτων χωρών), παρέχει όλες τις απαραίτητες πληροφορίες για την κατανόηση της XML Έκδοσης 1.1 και την κατασκευή προγραμμάτων υπολογιστή για την επεξεργασία της.» [7]

2.3 Android Studio

Εικόνα 2.3 - Λογότυπο του Android Studio

«Το Android Studio είναι ένα ολοκληρωμένο προγραμματιστικό περιβάλλον (Integrated Development Environment- IDE) για ανάπτυξη εφαρμογών στην πλατφόρμα Android. Ανακοινώθηκε το 2013 από την Google. Η χρήση του είναι ελεύθερη με την άδεια Apache License 2.0.» Παρέχει στους προγραμματιστές έναν επεξεργαστή κειμένου, δυνατότητα αποσφαλμάτωσης και προσομοιώνει λειτουργίες, όπως η τοποθεσία GPS, η καθυστέρηση δικτύου, οι αισθητήρες κίνησης και η δυνατότητα πολλαπλής αφής. Επιπλέον, παρέχει emulator

δίνοντας στους προγραμματιστές την δυνατότητα δοκιμής των εφαρμογών τους σε διάφορα κινητά, tablets, Android Wear και συσκευές Android TV. [8]

2.4 Firebase

Εικόνα 2.4 - Λογότυπο του Firebase

«Το Firebase είναι ένα πλαίσιο που είναι χρήσιμο για τη δημιουργία φορητών και διαδικτυακών εφαρμογών για επιχειρήσεις που απαιτούν βάση δεδομένων πραγματικού χρόνου που σημαίνει ότι όταν ένας χρήστης ενημερώνει μια εγγραφή στη βάση δεδομένων, η ενημέρωση πρέπει να μεταφέρεται σε κάθε χρήστη στη στιγμή. Παρέχει μια βασική και ενοποιημένη πλατφόρμα σε πολλές εφαρμογές μαζί με μια πλειάδα άλλων λειτουργιών της Google. Το Firebase χειρίζεται το μεγαλύτερο μέρος της εργασίας από την πλευρά του διακομιστή όταν πρόκειται για την ανάπτυξη εφαρμογών. Υπάρχουν πολυάριθμα στοιχεία που καθιστούν το Firebase ένα τόσο ουσιαστικό εργαλείο στην ανάπτυξη από τη σκοπιά ενός προγραμματιστή. Με αυτόν τον τρόπο,
βοηθά στη διατήρηση μιας κατάστασης αρμονίας μεταξύ του προγραμματιστή και του πελάτη, προκαλώντας ελάχιστη καθυστέρηση στην εργασία.»[9] Επίσης είναι μια βάση NoSQL που φιλοξενεί το cloud, το οποίο μας επιτρέπει να αποθηκεύουμε και να συγχρονίζουμε τους χρήστες σε πραγματικό χρόνο. Η βάση δεδομένων Realtime είναι ένα μεγάλο αντικείμενο JSON το οποίο οι προγραμματιστές μπορούν να διαχειριστούν σε πραγματικό χρόνο. [10]

2.4.1 Βάση δεδομένων πραγματικού χρόνου (Realtime database)

«Η βάση δεδομένων σε πραγματικό χρόνο είναι μια βάση δεδομένων που φιλοξενείται σε cloud. Τα δεδομένα αποθηκεύονται ως JSON και συγχρονίζονται συνεχώς σε κάθε συνδεδεμένο πελάτη. Όταν οποιαδήποτε εφαρμογή πολλαπλών πλατφορμών αναπτύσσεται με iOS, Android και JavaScript SDKs, το μεγαλύτερο μέρος των απαιτήσεων του χρήστη, βασίζεται σε ένα

στιγμιότυπο βάσης δεδομένων σε πραγματικό χρόνο και αυτή η παρουσία ενημερώνεται με τα νέα δεδομένα. Αυτή η δυνατότητα επιτρέπει στους προγραμματιστές να παραλείψουν το βήμα της ανάπτυξης μιας βάσης δεδομένων και το Firebase χειρίζεται τα περισσότερα από αυτά για τις εφαρμογές. Παρέχει μια προσαρμόσιμη γλώσσα κανόνων που βασίζεται σε εκφράσεις για να καθορίσει τον τρόπο οργάνωσης των δεδομένων όταν οι πληροφορίες μπορούν να μελετηθούν.»11]

2.5 RSA

Βασική απαίτηση της εφαρμογής, μιας και πρόκειται για λογισμικό που συλλέγει και διατηρεί προσωπικά δεδομένα των χρηστών του, είναι η κρυπτογράφηση των στοιχείων αυτών. Για τον σκοπό αυτό, χρησιμοποιήθηκε ο κρυπταλγόριθμος RSA. Πρόκειται για έναν κρυπταλγόριθμο ασύμμετρου κλειδιού, που δημιούργησαν οι Ρον Ρίβεστ, Άντι Σαμίρ και Λεν Άντλμαν, από τους οποίους πήρε και το όνομά του. Παρακάτω περιγράφονται ο τρόπος κρυπτογράφησης και αποκρυπτογράφησης ενός μηνύματος, με την χρήση του RSA.

2.5.1 Κρυπτογράφηση

Ο κύριος λόγος για τον οποίο ο RSA είναι ασφαλής, είναι η επιλογή μεγάλων πρώτων αριθμών και η δυσκολία παραγοντοποίησής τους. Κατά την εκτέλεση του αλγορίθμου δημιουργούνται δύο κλειδιά, ένα ιδιωτικό (private key) και ένα δημόσιο (public key). Αρχικά, η δημιουργία του ιδιωτικού και του δημοσίου κλειδιού εξαρτώνται από δύο διαφορετικούς τυχαίους, μεγάλους πρώτους αριθμούς, οι οποίοι συχνά έχουν το ίδιο μέγεθος. Αφού επιλεγούν οι δύο αριθμοί, υπολογίζεται το γινόμενό τους:

$$p * q = n$$

Έπειτα υπολογίζεται το γινόμενο:

$$(p - 1) * (q - 1)$$

Στη συνέχεια, επιλέγεται τυχαία και πάλι, ένας πρώτος αριθμός e ο οποίος θα πρέπει να είναι μικρότερος του 1 και πρώτος σε σχέση με το γινόμενο (p - 1) * (q – 1), δηλαδή ο αριθμός e και το γινόμενο αυτό να μην έχουν κανέναν κοινό παράγοντα πέραν της μονάδας. Ο αριθμός e αποτελεί και το δημόσιο κλειδί. Με την παραγωγή του δημοσίου κλειδιού, δίνεται στον αποστολέα μια μονοσήμαντη συνάρτηση με βάση το δημόσιο αυτό κλειδί, με την χρήση της οποίας μπορεί να κρυπτογραφήσει ένα μήνυμα. [12]

2.5.2 Αποκρυπτογράφηση

Για την αποκρυπτογράφηση ενός μηνύματος που έχει κρυπτογραφηθεί με την χρήση του δημοσίου κλειδιού που παράχθηκε από τον RSA, απαιτείται η χρήση ενός μοναδικού ιδιωτικού κλειδιού d. Ο αριθμός d είναι το υπόλοιπο μιας διαίρεσης και ο μαθηματικός τύπος που τον ορίζει είναι ο εξής:

$$d = 1 / e \bmod ((p - 1) * (q - 1))$$

Στον παραπάνω τύπο φαίνεται πως ο αριθμός d εξαρτάται και από την τιμή του δημόσιου κλειδιού αλλά και αυτές των αριθμών p και q. Εάν ο παραλήπτης γνωρίζει τον ιδιωτικό κλειδί d, μπορεί να προχωρήσει στην αποκρυπτογράφηση του μηνύματος του παραλήπτη χρησιμοποιώντας το. [13]

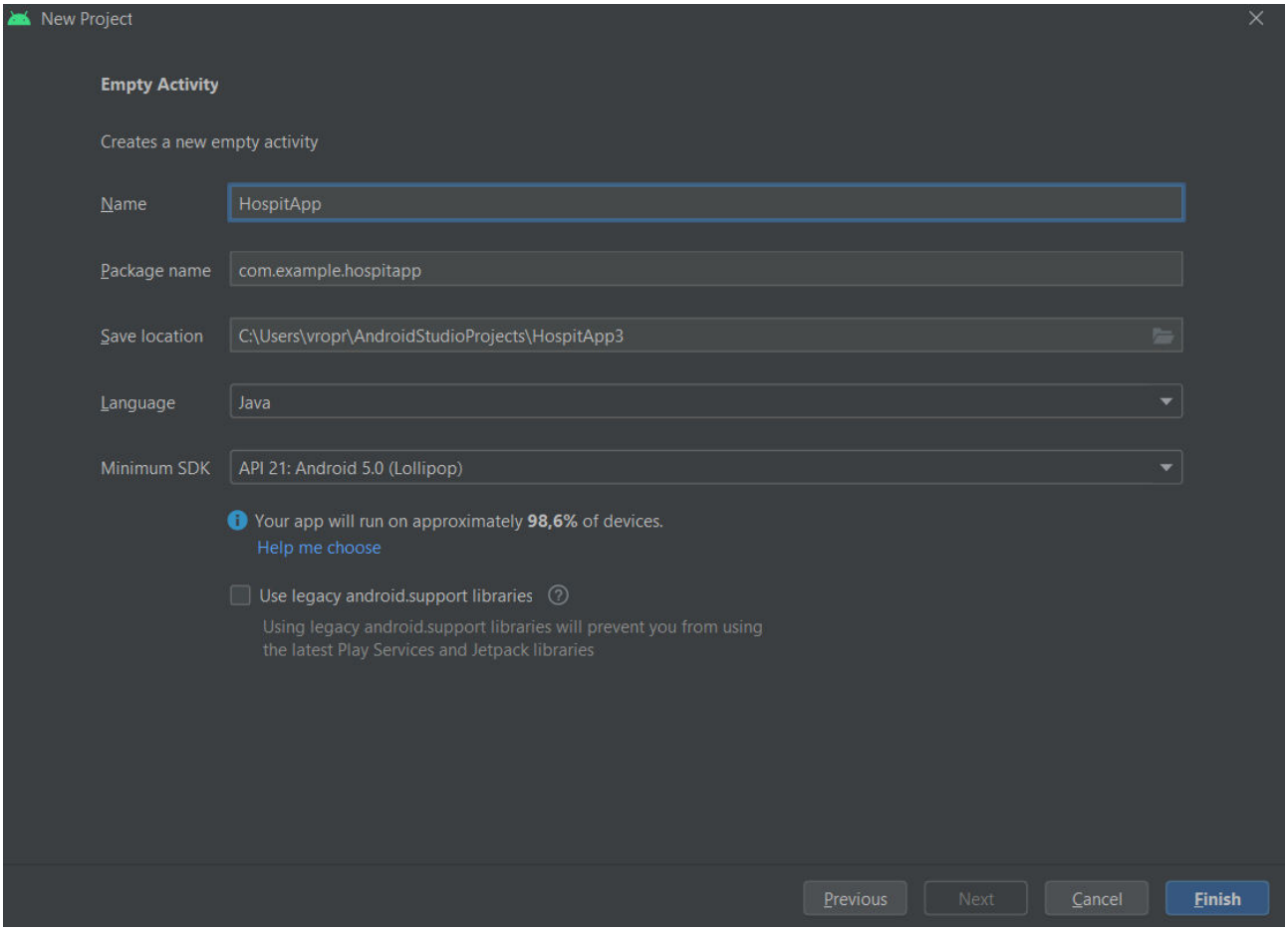
2.5.3 Ασφάλεια RSA

Όσο μεγαλύτεροι είναι οι πρώτοι αριθμοί που χρησιμοποιούνται για την παραγωγή των κλειδιών τόσο ασφαλέστερος θεωρείται αλγόριθμος. Παρόλα αυτά η κακή του χρήση μπορεί να οδηγήσει σε μεγάλες αδυναμίες ασφάλειας. Εξάλλου μέχρι τώρα δεν έχει αποδειχθεί πως ο μόνος παράγοντας που καθορίζει το επίπεδο ασφαλείας του είναι η παραγοντοποίηση των ακεραίων. Ακόμη, υπάρχει πάντα η πιθανότητα να δημιουργηθεί κάποιος αλγόριθμος (αν δεν υπάρχει ήδη) με τον οποίο θα είναι δυνατή η παραγοντοποίηση σε πολυωνυμικό χρόνο. [14]

3 Η εφαρμογή HospitApp

Στο παρόν κεφάλαιο θα γίνει παρουσίαση της εφαρμογής. Παρακάτω παρατίθενται ο πηγαίος κώδικας της εφαρμογής μαζί με σχετικές εικόνες και σχόλια για την ευκολότερη κατανόησή του αλλά και στιγμιότυπα οθόνης (screenshots) από την λειτουργία της.

Όπως αναφέρεται και παραπάνω, η εφαρμογή υλοποιήθηκε με την χρήση του Android Studio. Αρχικά λοιπόν δημιουργήθηκε το project και επιλέχθηκαν το όνομα της εφαρμογής, η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την δημιουργία της, αλλά και η κατώτερη Android έκδοση στην οποία θα τρέχει. Η γλώσσα προγραμματισμού που επέλεξα για την συγκεκριμένη εφαρμογή είναι η Java και το όνομα της εφαρμογής HospitApp. Το όνομα προκύπτει από μια ελαφρώς ανορθόδοξη σύμπτυξη των αγγλικών λέξεων «*hospital» (νοσοκομείο)* και «*application» (εφαρμογή).* Η κατώτερη έκδοση Android στην οποία μπορεί να λειτουργήσει η εφαρμογή είναι η Android Lollipop 5.0. Ο λόγος επιλογής της έκδοσης αυτής είναι πως μπορεί να χρησιμοποιηθεί από αρκετά μεγάλο εύρος συσκευών, σε ποσοστό 98,6% σύμφωνα με το Android Studio.



Εικόνα 3.1 - Αρχικές επιλογές εφαρμογής

Η εφαρμογή προκειμένου να λειτουργεί σωστά, απαιτεί πρόσβαση στο internet αλλά και την δυνατότητα να «διαβάζει» εξωτερικά αρχεία από την μνήμη της συσκευής. Για να είναι αυτό εφικτό, η εφαρμογή θα πρέπει να ζητάει άδεια από τον χρήστη. Παρακάτω φαίνεται ο απαιτούμενος κώδικας που γράφτηκε στο AndroidManifest.xml αρχείο του project.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

Εικόνα 3.2 - Αιτήσεις άδειας για πρόσβαση στις λειτουργίες

- Android.permission.INTERNET - ζητάει δικαίωμα χρήσης internet είτε από δεδομένα κινητής τηλεφωνίας είτε από χρήση του Wi-Fi.

- Android.permission.READ_EXTERNAL_STORAGE - ζητάει δικαίωμα ανάγνωσης από εξωτερικό χώρο αποθήκευσης.

3.1 Activities

Ο προγραμματισμός της εφαρμογής μέσω του Android Studio, είναι βασισμένος σε activities (δραστηριότητες). Για την υλοποίηση κάθε μιας από αυτές, υπάρχουν δύο αρχεία, μια Java κλάση και ένα αρχείο .xml. Στις κλάσεις γίνεται ο προγραμματισμός της κάθε δραστηριότητας ενώ τα xml αρχεία βρίσκονται στον φάκελο app/res/layout και οι εντολές που περιέχουν είναι υπεύθυνες για την δημιουργία του γραφικού περιβάλλοντος. Τα δύο αυτά αρχεία επικοινωνούν μεταξύ τους έτσι ώστε να επιτυγχάνεται ο σκοπός της κάθε δραστηριότητας.

3.2 Βιβλιοθήκες

Στο αρχείο build.gradle(Module.app), υπάρχουν εξ' αρχής οι βασικές βιβλιοθήκες για την λειτουργία της εφαρμογής. Εκτός όμως από αυτές το Android Studio δίνει στον προγραμματιστή την δυνατότητα να προσθέσει και άλλες προϋπάρχουσες βιβλιοθήκες, ώστε να χρησιμοποιηθούν κατά την υλοποίηση.

```
1   plugins {
2       id 'com.android.application'
3   }
4   apply plugin: 'com.google.gms.google-services'
5   android {...}
33
34  dependencies {
35
36      implementation 'androidx.appcompat:appcompat:1.3.1'
37      implementation 'com.google.android.material:material:1.4.0'
38      implementation 'androidx.constraintlayout:constraintlayout:2.1.0'
39      //noinspection GradleCompatible,GradleCompatible
40      implementation 'com.android.support:appcompat-v7:26.1.0'
41      //noinspection GradleCompatible,GradleCompatible
42      implementation 'com.android.support:design:26.1.0'
43      implementation 'androidx.navigation:navigation-fragment:2.3.5'
44      implementation 'androidx.navigation:navigation-ui:2.3.5'
45      implementation 'androidx.annotation:annotation:1.2.0'
46      implementation 'androidx.constraintlayout:constraintlayout-core:1.0.1'
47      implementation 'com.google.firebase:firebase-database:20.0.2'
48      implementation 'com.google.firebase:firebase-storage:20.0.1'
49
50      testImplementation 'junit:junit:4.+'
51      androidTestImplementation 'androidx.test.ext:junit:1.1.3'
52      androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
53  }
```

Εικόνα 3.3 - Εισαγωγή βιβλιοθηκών

Οι βιβλιοθήκες που προστέθηκαν στο project είναι οι εξής:

- 'com.google.firebase:firebase-database:20.0.2'
- 'com.google.firebase:firebase-storage:20.0.1'

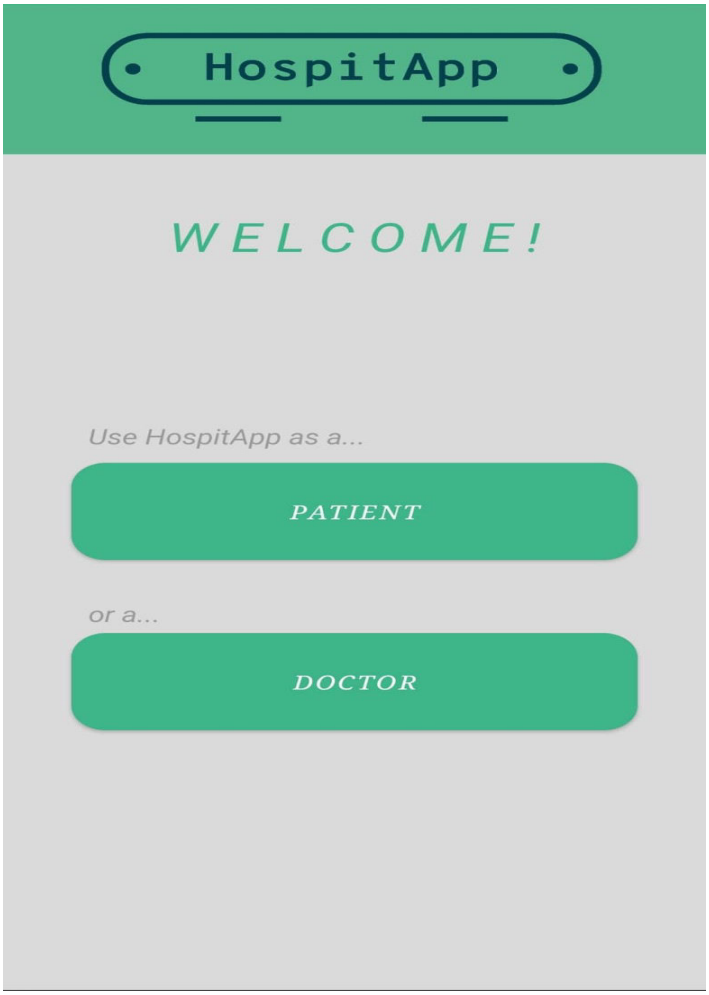
Οι παραπάνω βιβλιοθήκες προστέθηκαν για την χρήση του Firebase. Μέσω αυτών δίνεται στην εφαρμογή η δυνατότητα συγχρονισμού με την βάση δεδομένων.

3.3 Λειτουργίες εφαρμογής

Προκειμένου να γίνει πιο κατανοητή η παρουσίασης των κλάσεων της εφαρμογής, θα παρουσιαστούν 2 διαφορετικά σενάρια, στο ένα εκ των οποίων ο χρήστης θα συνδέεται στην εφαρμογή ως ασθενής, ενώ στο άλλο ως γιατρός.

3.3.1 Εκκίνηση εφαρμογής

Ξεκινώντας, μέσω της κλάσης StartActivity, η εφαρμογή δίνει στον χρήστη την επιλογή για το εάν θα συνδεθεί ως ασθενής ή ως γιατρός. Αφού ο χρήστης επιλέξει, σε επόμενη οθόνη του ζητείται να συνδεθεί. Για την υλοποίηση του γραφικού περιβάλλοντος της κλάσης, χρησιμοποιείται το xml αρχείο activity_start.xml. Η κλάση StartActivity είναι κοινή και για τους γιατρούς και για τους ασθενείς. Ακολουθούν εικόνες της αρχικής οθόνης και screenshots του κώδικα της κλάσης StartActivity.



Εικόνα 3.4 - Αρχική οθόνη εφαρμογής

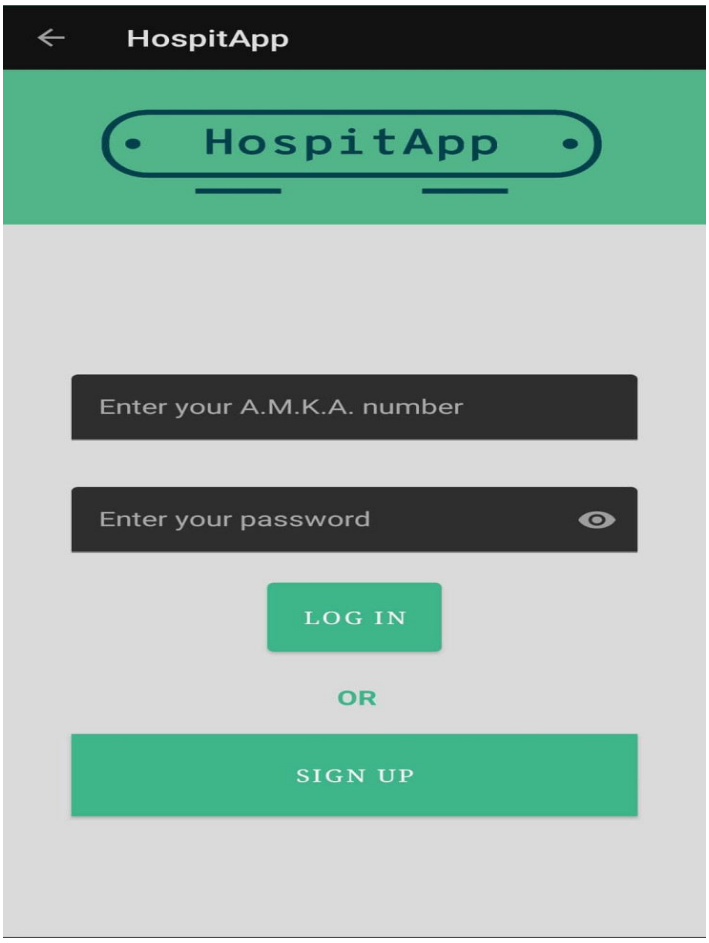
```java
package com.example.hospitapp;

import ...

public class StartActivity extends AppCompatActivity {

    Button patientUse;
    Button doctorUse;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_start);

        patientUse = (Button) findViewById(R.id.patientUse);
        doctorUse = (Button) findViewById(R.id.doctorUse);

        patientUse.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent( packageContext: StartActivity.this, MainActivity.class);
                startActivity(intent);
            }
        });

        doctorUse.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent( packageContext: StartActivity.this, DoctorLogIn.class);
                startActivity(intent);
            }
        });
    }
```

Εικόνα 3.5 - Η κλάση StartActivity

1η περίπτωση: Χρήση από ασθενή

3.3.2 Σύνδεση χρήστη

Αφού ο χρήστης επιλέξει να συνδεθεί ως ασθενής, μεταφέρεται στην επόμενη οθόνη, όπου καλείται είτε να συνδεθεί, εισάγοντας τον A.M.K.A. του και τον κωδικό χρήστη του, εάν έχει ήδη εγγραφεί στο σύστημα, είτε να κάνει εγγραφή δημιουργώντας τον δικό του λογαριασμό. Σε περίπτωση επιλογής σύνδεσης από τον χρήστη, κλάση MainActivity είναι υπεύθυνη για την επαλήθευση των στοιχείων που εισήχθησαν, αναζητώντας και συγκρίνοντάς τα με αυτά που υπάρχουν αποθηκευμένα στην βάση δεδομένων. Ο έλεγχος αυτός γίνεται με τις συναρτήσεις getPatientInfo() και checkAmkaPassword(), καθώς η πρώτη αποθηκεύει σε μεταβλητές τα στοιχεία που εισήγαγε ο χρήστης και τα στέλνει στην δεύτερη η οποία κάνει την σύγκριση και την αναζήτηση στην βάση. Σε περίπτωση που κάποιο από τα απαιτούμενα πλαίσια κειμένου μείνει κενό, ή εάν ο A.M.K.A. ή ο κωδικός χρήστη είναι λανθασμένα, εμφανίζεται κατάλληλο μήνυμα και δεν επιτρέπεται η πρόσβαση στον χρήστη. Στα παρακάτω στιγμιότυπα φαίνεται η οθόνη σύνδεσης - εγγραφής και ο κώδικας της κλάσης .



Εικόνα 3.6 - Οθόνη σύνδεσης ή εγγραφής

```java
package com.example.hospitapp;

import androidx.annotation.NonNull;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import java.io.UnsupportedEncodingException;

public class MainActivity extends AppCompatActivity {
    private
    TextView or;
    Button login, signup;
    TextInputLayout amka, password;
    String amkaStr, passwordStr, passwordOnDB, patientNameStr, patientSurnameStr, patientAmkaStr;
```

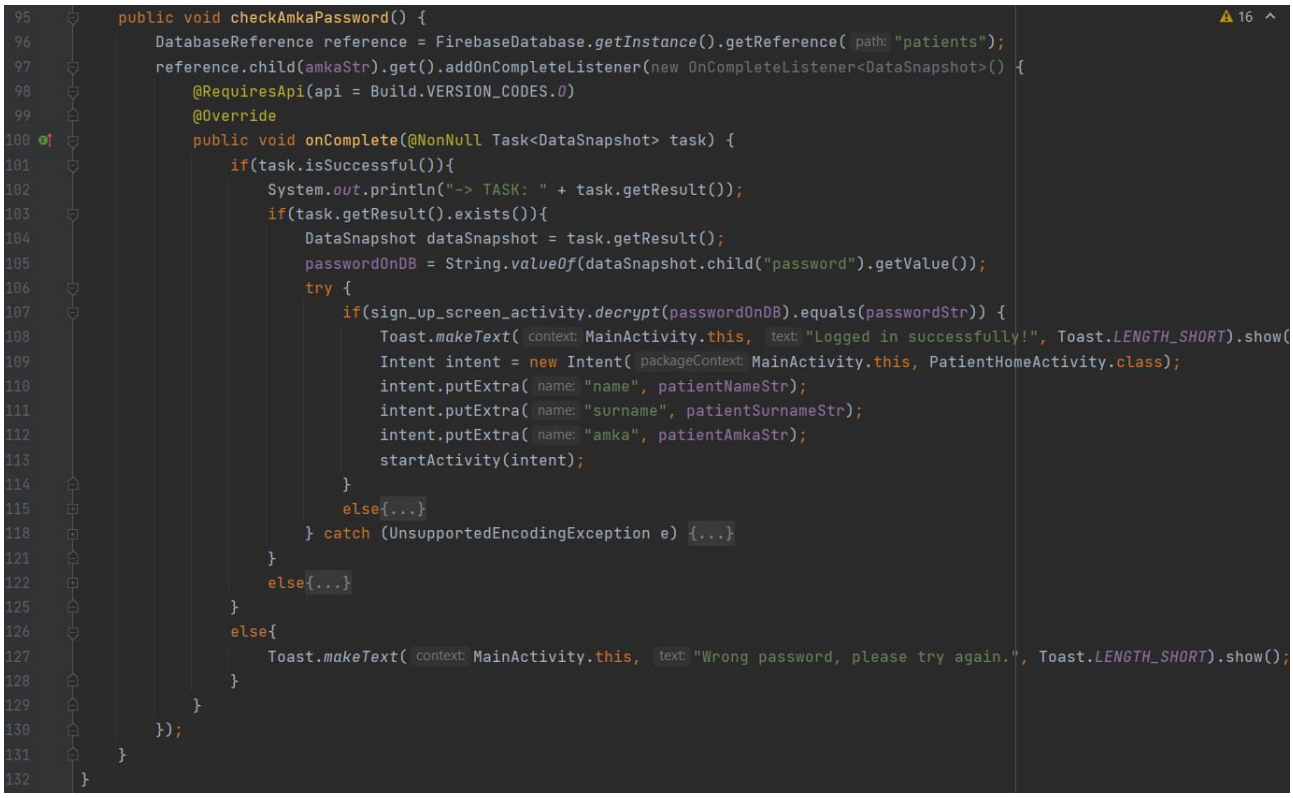
Εικόνα 3.7 - Κλάση MainActivity (1)

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        or = (TextView) findViewById(R.id.or);
        login = (Button) findViewById(R.id.LogInDoc);
        amka = (TextInputLayout) findViewById(R.id.editTextAmka);
        signup = (Button) findViewById(R.id.SignUp);
        password = (TextInputLayout) findViewById(R.id.editTextPassword);

        login.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                amkaStr = amka.getEditText().getText().toString();
                passwordStr = password.getEditText().getText().toString();

                if(amkaStr.isEmpty()||passwordStr.isEmpty())
                    Toast.makeText( context: MainActivity.this,  text: "Please fill all fields.", Toast.LENGTH_SHORT).show();
                else{
                    getPatientInfo();
                    checkAmkaPassword();
                    amka.getEditText().setText("");
                    password.getEditText().setText("");
                }
            }
        });

        signup.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) { openSignUpScreen(); }
        });
```

Εικόνα 3.8 - Κλάση MainActivity (2)

```
    public void openSignUpScreen(){
        Intent intent = new Intent( packageContext: this, sign_up_screen_activity.class);
        startActivity(intent);
    }

    private void getPatientInfo() {
        DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "patients").child(amkaStr);
        reference.get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
            @Override
            public void onComplete(@NonNull Task<DataSnapshot> task) {
                if (task.isSuccessful()) {
                    System.out.println("Task succeed.");
                    if (task.getResult().exists()) {
                        DataSnapshot dataSnapshot = task.getResult();
                        patientNameStr = dataSnapshot.child("name").getValue().toString();
                        patientSurnameStr = dataSnapshot.child("surname").getValue().toString();
                        patientAmkaStr = dataSnapshot.child("amka").getValue().toString();
                    } else {
                        Toast.makeText( context: MainActivity.this,  text: "No patient with such A.M.K.A. exist.", Toast.LENGTH_SHORT).s
                    }
                } else {
                    System.out.println(task.isSuccessful());
                    Toast.makeText( context: MainActivity.this,  text: "Wrong password, please try again.", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
```

Εικόνα 3.9 - Κλάση MainActivity (3), συνάρτηση getPatientInfo()

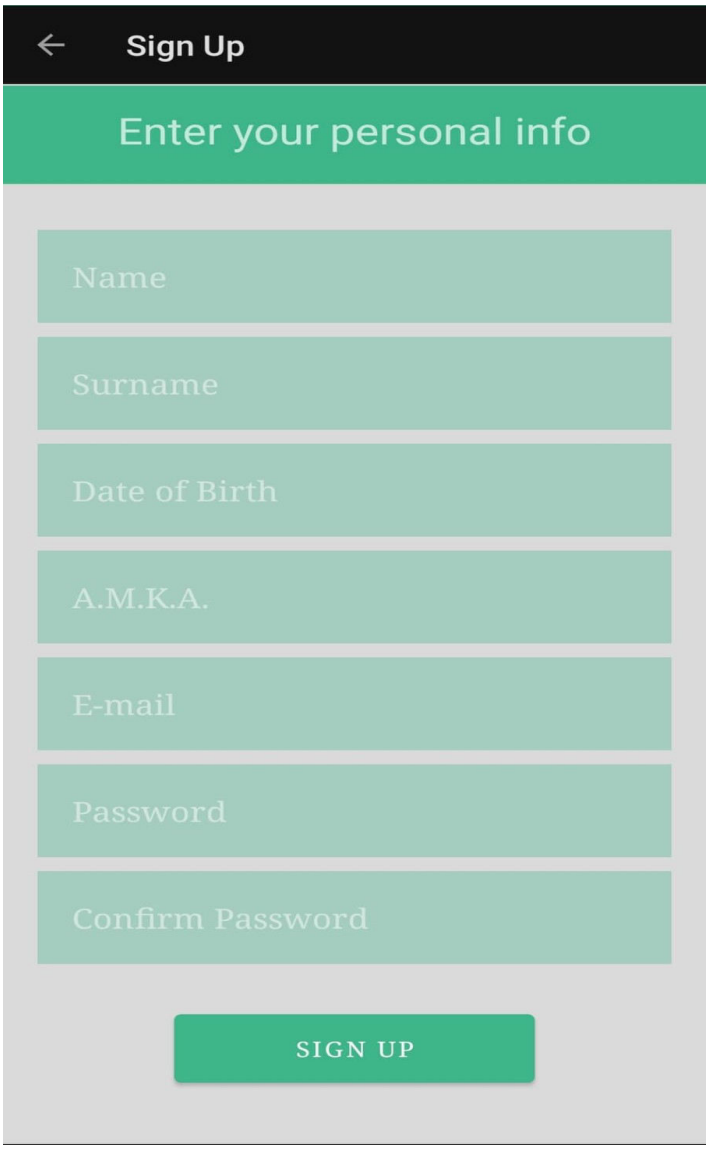


```
95      public void checkAmkaPassword() {                                                    ⚠ 16 ^
96          DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "patients");
97          reference.child(amkaStr).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
98              @RequiresApi(api = Build.VERSION_CODES.O)
99              @Override
100             public void onComplete(@NonNull Task<DataSnapshot> task) {
101                 if(task.isSuccessful()){
102                     System.out.println("-> TASK: " + task.getResult());
103                     if(task.getResult().exists()){
104                         DataSnapshot dataSnapshot = task.getResult();
105                         passwordOnDB = String.valueOf(dataSnapshot.child("password").getValue());
106                         try {
107                             if(sign_up_screen_activity.decrypt(passwordOnDB).equals(passwordStr)) {
108                                 Toast.makeText( context: MainActivity.this,  text: "Logged in successfully!", Toast.LENGTH_SHORT).show(
109                                 Intent intent = new Intent( packageContext: MainActivity.this, PatientHomeActivity.class);
110                                 intent.putExtra( name: "name", patientNameStr);
111                                 intent.putExtra( name: "surname", patientSurnameStr);
112                                 intent.putExtra( name: "amka", patientAmkaStr);
113                                 startActivity(intent);
114                             }
115                             else{...}
118                         } catch (UnsupportedEncodingException e) {...}
121                     }
122                     else{...}
125                 }
126                 else{
127                     Toast.makeText( context: MainActivity.this,  text: "Wrong password, please try again.", Toast.LENGTH_SHORT).show();
128                 }
129             }
130         });
131     }
132 }
```

Εικόνα 3.10 - Κλάση MainActivity (4), συνάρτηση checkAmkaPassword()

3.3.3 Εγγραφή νέου χρήστη

Εάν ο ασθενής επιλέξει να εγγραφεί, τότε οδηγείται στην οθόνη εγγραφής νέου ασθενή μέσω της κλάσης *sign_up_screen_activity*. Σε αυτό το σημείο ζητείται από τον χρήστη να εισάγει κάποια προσωπικά του στοιχεία (όνομα, επώνυμο, ημερομηνία γέννησης, A.M.K.A., e-mail) και να επιλέξει έναν κωδικό ασφαλείας. Η κλάση είναι υπεύθυνη για την κρυπτογράφηση και μεταφόρτωση των στοιχείων του χρήστη στην βάση δεδομένων. Εάν η εισαγωγή των στοιχείων γίνει επιτυχώς, ο χρήστης μεταφέρεται και πάλι στην αρχική οθόνη της εφαρμογής ώστε να συνδεθεί. Παρακάτω παρατίθενται screenshots του κώδικα της κλάσης και της οθόνης εγγραφής.



Εικόνα 3.11 - Οθόνη εγγραφής

```java
package com.example.hospitapp;

import ...

public class sign_up_screen_activity extends AppCompatActivity {

    FirebaseDatabase rootNode;
    DatabaseReference reference;

    TextView name, surname, date, amka, email, password, confPasswd;
    Button signup;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.sign_up_screen);
        getSupportActionBar().setTitle("Sign Up");

        name = (TextView) findViewById(R.id.PatientName);
        surname = (TextView) findViewById(R.id.PatientSurname);
        date = (TextView) findViewById(R.id.PatientDate);
        amka = (TextView) findViewById(R.id.PatientAMKA);
        email = (TextView) findViewById(R.id.PatientEmail);
        password = (TextView) findViewById(R.id.PatientPassword);
        confPasswd = (TextView) findViewById(R.id.ConfirmPassword);
        signup = (Button) findViewById(R.id.Signup);

        signup.setOnClickListener(new View.OnClickListener() {
            @RequiresApi(api = Build.VERSION_CODES.O)
            @Override
            public void onClick(View view) {
                rootNode = FirebaseDatabase.getInstance();
```

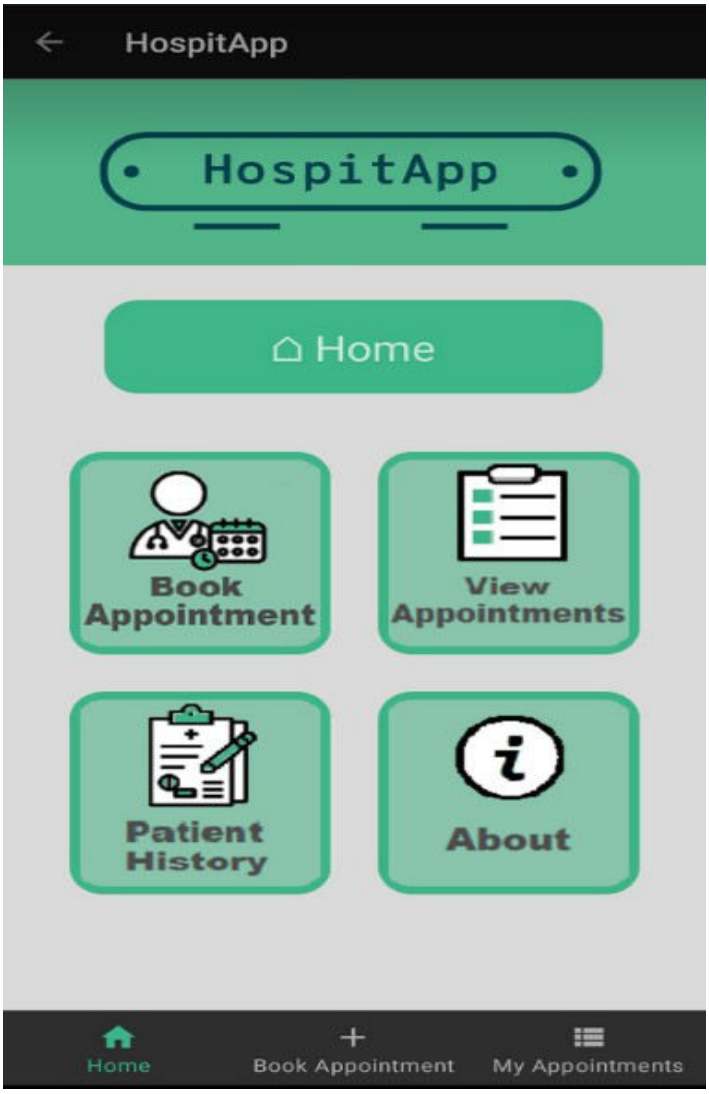
Εικόνα 3.12 - Κλάση sign_up_screen (1)

```java
package com.example.hospitapp;

import ...

public class sign_up_screen_activity extends AppCompatActivity {

    FirebaseDatabase rootNode;
    DatabaseReference reference;

    TextView name, surname, date, amka, email, password, confPasswd;
    Button signup;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.sign_up_screen);
        getSupportActionBar().setTitle("Sign Up");

        name = (TextView) findViewById(R.id.PatientName);
        surname = (TextView) findViewById(R.id.PatientSurname);
        date = (TextView) findViewById(R.id.PatientDate);
        amka = (TextView) findViewById(R.id.PatientAMKA);
        email = (TextView) findViewById(R.id.PatientEmail);
        password = (TextView) findViewById(R.id.PatientPassword);
        confPasswd = (TextView) findViewById(R.id.ConfirmPassword);
        signup = (Button) findViewById(R.id.Signup);

        signup.setOnClickListener(new View.OnClickListener() {
            @RequiresApi(api = Build.VERSION_CODES.O)
            @Override
            public void onClick(View view) {
                rootNode = FirebaseDatabase.getInstance();
```

Εικόνα 3.13 - Κλάση sign_up_screen (2)

3.3.4 Βασικό μενού ασθενή

Μόλις ο ασθενής συνδεθεί επιτυχώς στην εφαρμογή, εμφανίζεται το βασικό μενού της εφαρμογής μέσω της κλάσης *PatientHomeActivity*. Το μενού αυτό περιέχει τρία βασικά buttons, τα οποία οδηγούν στον κανονισμό νέου ραντεβού με κάποιον γιατρό, στην προβολή των προηγούμενων και επερχόμενων ραντεβού του ασθενή και στην προβολή των διαγνώσεων που έχει λάβει ως τώρα και υπάρχουν στην βάση δεδομένων. Μέσω των συναρτήσεων της κλάσης *PatientHomeActivity* και με βάση τις επιλογές του χρήστη, καλούνται οι κατάλληλες κλάσεις για την διεκπεραίωση των επιθυμιών του. Εάν ο χρήστης επιλέξει «Book Appointment», καλείται η συνάρτηση «openBookAppointment», εάν επιλέξει «View Appointments», καλείται η συνάρτηση «openViewAppointments» και αντίστοιχα αν επιλέξει «Patient History», καλείται η openViewPatientHistory. Παρακάτω παρουσιάζεται ο κώδικας υλοποίησης των λειτουργιών της κλάσης *PatientHomeActivity* και στιγμιότυπο της αρχικής οθόνης ασθενή.



Εικόνα 3.14 - Αρχική οθόνη χρήστη – ασθενή

```java
package com.example.hospitapp;

import ...

public class PatientHomeActivity extends AppCompatActivity {

    Button bookAp;
    Button viewAp;
    Button viewHistory;
    String patientNameStr, patientSurnameStr, patientAmkaStr;
    BottomNavigationView bottomNavigationView;

    public PatientHomeActivity() {
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        Bundle extras = getIntent().getExtras();
        patientNameStr = extras.getString( key: "name");
        patientSurnameStr = extras.getString( key: "surname");
        patientAmkaStr = extras.getString( key: "amka");
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_patient_home);
        getSupportActionBar().setTitle("HospitApp");

        bookAp = (Button) findViewById(R.id.bookAppointment);
        viewAp = (Button) findViewById(R.id.myAppointmerts);
        viewHistory = (Button) findViewById(R.id.viewHistory);
        bottomNavigationView = findViewById(R.id.bottom_navigation);

        bottomNavigationView.setSelectedItemId(R.id.nav_home);

```

Εικόνα 3.15 - Κλάση PatientHomeActivity (1)

```
43          bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigat
44              @Override
45              public boolean onNavigationItemSelected(@NonNull MenuItem item) {
46                  switch (item.getItemId()){
47                      case R.id.nav_home:
48                          return  true;
49                      case R.id.nav_app_list:
50                          Intent intent = new Intent(getApplicationContext(), ViewAppointments.class);
51                          intent.putExtra( name: "name", patientNameStr);
52                          intent.putExtra( name: "surname", patientSurnameStr);
53                          intent.putExtra( name: "amka", patientAmkaStr);
54                          startActivity(intent);
55                          overridePendingTransition( enterAnim: 0, exitAnim: 0);
56                          return true;
57                      case R.id.book_app:
58                          Intent intent2 = new Intent(getApplicationContext(), BookAppointment.class);
59                          intent2.putExtra( name: "name", patientNameStr);
60                          intent2.putExtra( name: "surname", patientSurnameStr);
61                          intent2.putExtra( name: "amka", patientAmkaStr);
62                          startActivity(intent2);
63                          overridePendingTransition( enterAnim: 0, exitAnim: 0);
64                          return true;
65                  }
66                  return false;
67              }
68          });
69
70          bookAp.setOnClickListener(new View.OnClickListener() {
71              @Override
72              public void onClick(View view) { openBookAppointment(); }
75          });
76
```

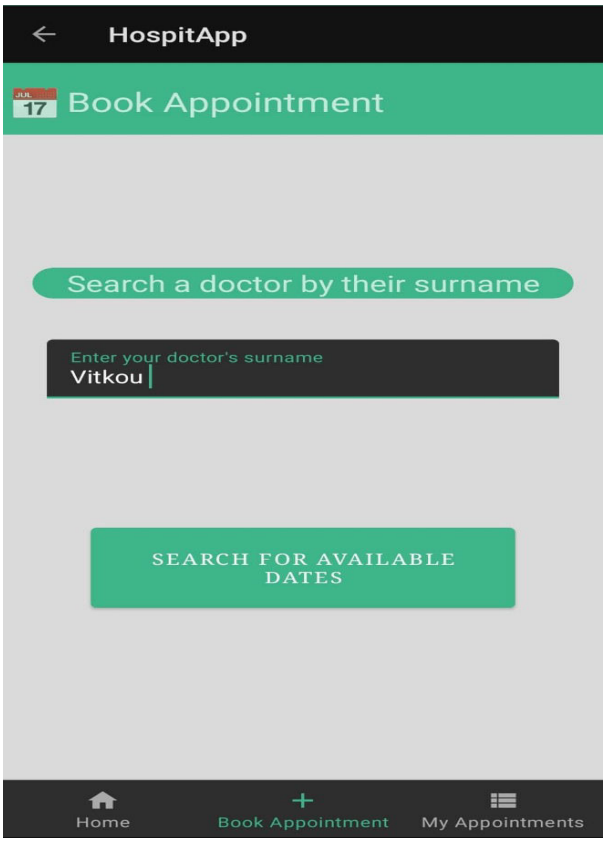
Εικόνα 3.16 - Κλάση PatientHomeActivity (2)

```
77          viewAp.setOnClickListener(new View.OnClickListener(){
78              @Override
79              public void onClick(View view) { openViewAppointments(); }
82          });
83          viewHistory.setOnClickListener(new View.OnClickListener(){
84              @Override
85              public void onClick(View view) { openViewHistory(); }
88          });
89      }
90      @Override
91      public void onBackPressed() {
92          super.onBackPressed(); Toast.makeText( context: PatientHomeActivity.this,  text: "Logged out.", To
93      }
94      public void openBookAppointment() {
95          Intent intent = new Intent( packageContext: PatientHomeActivity.this, BookAppointment.class);
96          intent.putExtra( name: "name", patientNameStr);
97          intent.putExtra( name: "surname", patientSurnameStr);
98          intent.putExtra( name: "amka", patientAmkaStr);
99          startActivity(intent);
100     }
101     public void openViewAppointments() {
102         Intent intent = new Intent( packageContext: PatientHomeActivity.this, ViewAppointments.class);
103         intent.putExtra( name: "amka", patientAmkaStr);
104         startActivity(intent);
105     }
106     public void openViewHistory() {
107         Intent intent = new Intent( packageContext: PatientHomeActivity.this, ViewHistory.class);
108         intent.putExtra( name: "name", patientNameStr);
109         intent.putExtra( name: "surname", patientSurnameStr);
110         intent.putExtra( name: "amka", patientAmkaStr);
111         startActivity(intent);
112     }
113 }
```

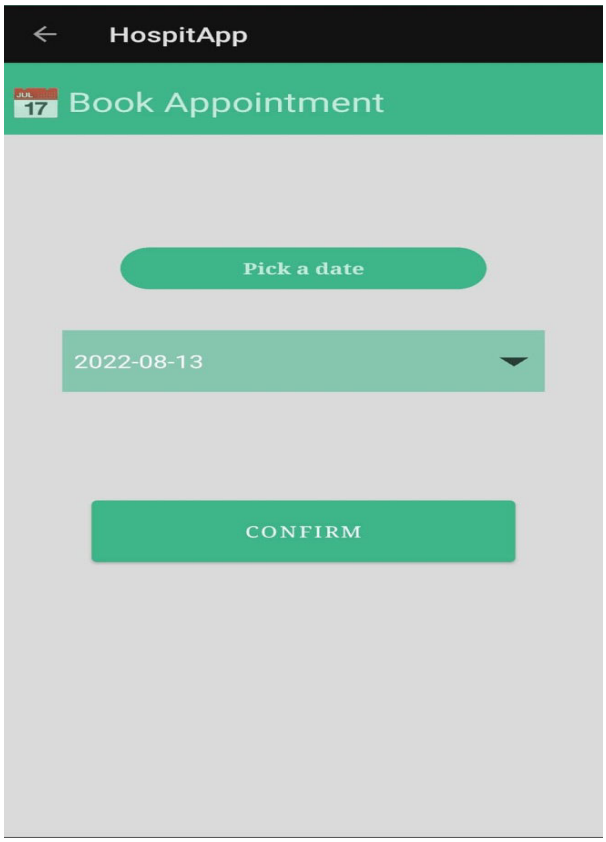
Εικόνα 3.17 - Κλάση PatientHomeActivity (3)

3.3.5 Προγραμματισμός ραντεβού

Στις κλάσεις BookAppointment και DisplayAvailableDates, υλοποιούνται όλες οι λειτουργίες που χρειάζονται για τον προγραμματισμού ενός νέου ραντεβού για τον ασθενή. Αρχικά, ζητείται από τον ασθενή να εισάγει το επώνυμο του ιατρού με τον οποίο θέλει να κλείσει το ραντεβού. Εφόσον ο γιατρός αυτός υπάρχει στην βάση δεδομένων, γίνεται αναζήτηση και εμφάνιση των ημερομηνιών κατά τις οποίες ο γιατρός αυτός είναι διαθέσιμος. Η προσπέλαση στην βάση δεδομένων για την επικύρωση πως ο γιατρός είναι καταχωρημένος, αλλά και για την εύρεση των διαθέσιμων για τον ιατρό ημερομηνιών, γίνεται με την χρήση της συνάρτησης findAvailableDates(). Εάν το επώνυμο υπάρχει στην βάση, τότε με το πάτημα του κουμπιού «*Search for available dates*» ο χρήστης μεταφέρεται σε επόμενη οθόνη ώστε να δει τις διαθέσιμες ημερομηνίες και να επιλέξει αυτή του ραντεβού. Εν συνεχεία, η εφαρμογή ψάχνει στα στοιχεία του γιατρού τις διαθέσιμες ημερομηνίες και τις αποθηκεύει σε έναν πίνακα, ο οποίος στην συνέχεια «γεμίζει» το Spinner που προβάλλεται στον χρήστη ώστε να επιλέξει. Μόλις ο χρήστης επιλέξει και την ημερομηνία που επιθυμεί, η εφαρμογή εμφανίζει τα στοιχεία του επερχόμενου ραντεβού τα οποία αποθηκεύονται και στην βάση δεδομένων. Παράλληλα, η επιλεχθείσα ημερομηνία αντικαθίσταται στην βάση δεδομένων με την λέξη «unavailable» ώστε να μην υπάρχει περιθώριο επιλογής της ίδιας ημερομηνίας για άλλο ραντεβού. Στις παρακάτω εικόνες υπάρχουν στιγμιότυπα της εφαρμογής από τον προγραμματισμό ραντεβού και ο κώδικας των σχετικών κλάσεων.



Εικόνα 3.19 - Αναζήτηση γιατρού



Εικόνα 3.20 - Επιλογή ημερομηνίας

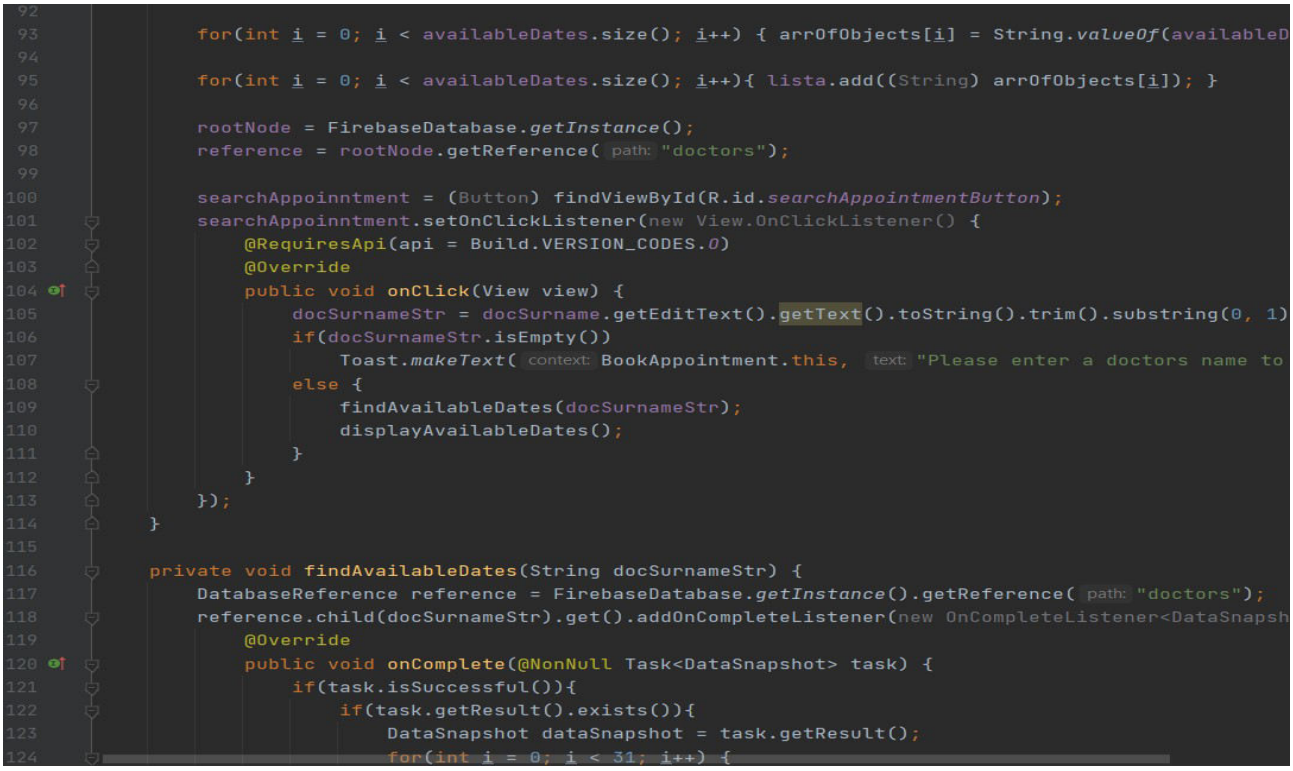
Εικόνα 3.21 - Πληροφορίες για το ραντεβού

```java
package com.example.hospitapp;

import ...

@RequiresApi(api = Build.VERSION_CODES.O)
public class BookAppointment extends AppCompatActivity {
    private static final int DAYS = 31;
    FirebaseDatabase rootNode;
    DatabaseReference reference;
    private Button searchAppoinntment;
    LocalDate start = LocalDate.now();
    List<LocalDate> availableDates;
    ArrayList<String> lista = new ArrayList<~>();
    TextInputLayout docSurname;
    String patientNameStr, patientSurnameStr, patientAmkaStr, docSurnameStr;
    BottomNavigationView bottomNavigationView;
    public BookAppointment() {
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_book_appointment);
        Bundle extras = getIntent().getExtras();
        patientNameStr = extras.getString( key: "name");
        patientSurnameStr = extras.getString( key: "surname");
        patientAmkaStr = extras.getString( key: "amka");

        getSupportActionBar().setTitle("HospitApp");
        bottomNavigationView = findViewById(R.id.bottom_navigation);

        bottomNavigationView.setSelectedItemId(R.id.book_app);
```

Εικόνα 3.22 - Κλάση BookAppointment (1)

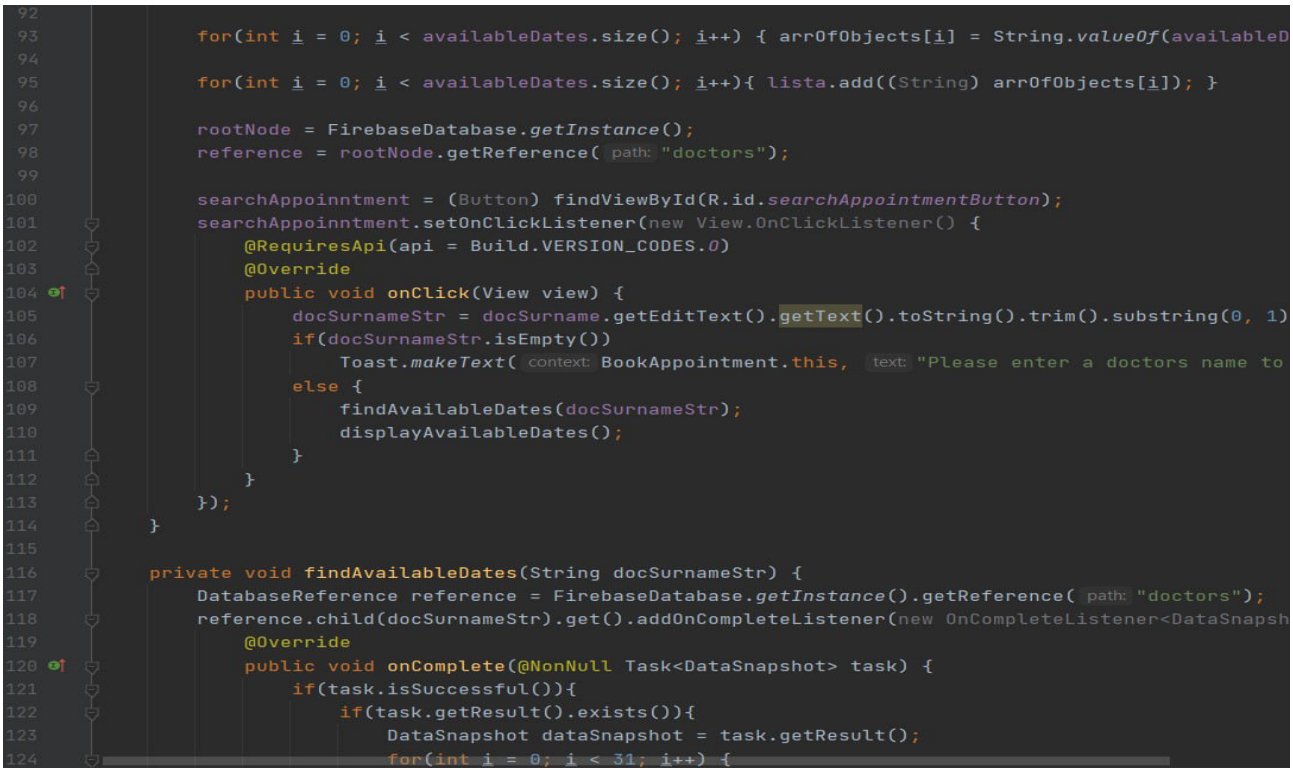
```java
        bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigati
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                switch (item.getItemId()){
                    case R.id.nav_home:
                        Intent intent = new Intent(getApplicationContext(), PatientHomeActivity.clas
                        intent.putExtra( name: "name", patientNameStr);
                        intent.putExtra( name: "surname", patientSurnameStr);
                        intent.putExtra( name: "amka", patientAmkaStr);
                        startActivity(intent);
                        overridePendingTransition( enterAnim: 0, exitAnim: 0);
                        return  true;
                    case R.id.book_app:
                        return true;
                    case R.id.nav_app_list:
                        Intent intent2 = new Intent(getApplicationContext(), ViewAppointments.class)
                        intent2.putExtra( name: "name", patientNameStr);
                        intent2.putExtra( name: "surname", patientSurnameStr);
                        intent2.putExtra( name: "amka", patientAmkaStr);
                        startActivity(intent2);
                        overridePendingTransition( enterAnim: 0, exitAnim: 0);
                        return true;
                }
                return false;
            }
        });

        docSurname = (TextInputLayout) findViewById(R.id.searchSurname);

        LocalDate start = LocalDate.now();
        this.availableDates = (availableDates(start, start.plusDays(31)));
        String[] arrOfObjects = new String[availableDates.size()];
```

Εικόνα 3.23 - Κλάση BookAppointment (2)



```
92
93          for(int i = 0; i < availableDates.size(); i++) { arrOfObjects[i] = String.valueOf(availableD
94
95          for(int i = 0; i < availableDates.size(); i++){ lista.add((String) arrOfObjects[i]); }
96
97          rootNode = FirebaseDatabase.getInstance();
98          reference = rootNode.getReference( path: "doctors");
99
100         searchAppoinntment = (Button) findViewById(R.id.searchAppointmentButton);
101         searchAppoinntment.setOnClickListener(new View.OnClickListener() {
102             @RequiresApi(api = Build.VERSION_CODES.O)
103             @Override
104             public void onClick(View view) {
105                 docSurnameStr = docSurname.getEditText().getText().toString().trim().substring(0, 1)
106                 if(docSurnameStr.isEmpty())
107                     Toast.makeText( context: BookAppointment.this,  text: "Please enter a doctors name to
108                 else {
109                     findAvailableDates(docSurnameStr);
110                     displayAvailableDates();
111                 }
112             }
113         });
114     }
115
116     private void findAvailableDates(String docSurnameStr) {
117         DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "doctors");
118         reference.child(docSurnameStr).get().addOnCompleteListener(new OnCompleteListener<DataSnapsh
119             @Override
120             public void onComplete(@NonNull Task<DataSnapshot> task) {
121                 if(task.isSuccessful()){
122                     if(task.getResult().exists()){
123                         DataSnapshot dataSnapshot = task.getResult();
124                         for(int i = 0; i < 31; i++) {
```

Εικόνα 3.24 - Κλάση BookAppointment (3)



```
92
93          for(int i = 0; i < availableDates.size(); i++) { arrOfObjects[i] = String.valueOf(availableD
94
95          for(int i = 0; i < availableDates.size(); i++){ lista.add((String) arrOfObjects[i]); }
96
97          rootNode = FirebaseDatabase.getInstance();
98          reference = rootNode.getReference( path: "doctors");
99
100         searchAppoinntment = (Button) findViewById(R.id.searchAppointmentButton);
101         searchAppoinntment.setOnClickListener(new View.OnClickListener() {
102             @RequiresApi(api = Build.VERSION_CODES.O)
103             @Override
104             public void onClick(View view) {
105                 docSurnameStr = docSurname.getEditText().getText().toString().trim().substring(0, 1)
106                 if(docSurnameStr.isEmpty())
107                     Toast.makeText( context: BookAppointment.this,  text: "Please enter a doctors name to
108                 else {
109                     findAvailableDates(docSurnameStr);
110                     displayAvailableDates();
111                 }
112             }
113         });
114     }
115
116     private void findAvailableDates(String docSurnameStr) {
117         DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "doctors");
118         reference.child(docSurnameStr).get().addOnCompleteListener(new OnCompleteListener<DataSnapsh
119             @Override
120             public void onComplete(@NonNull Task<DataSnapshot> task) {
121                 if(task.isSuccessful()){
122                     if(task.getResult().exists()){
123                         DataSnapshot dataSnapshot = task.getResult();
124                         for(int i = 0; i < 31; i++) {
```

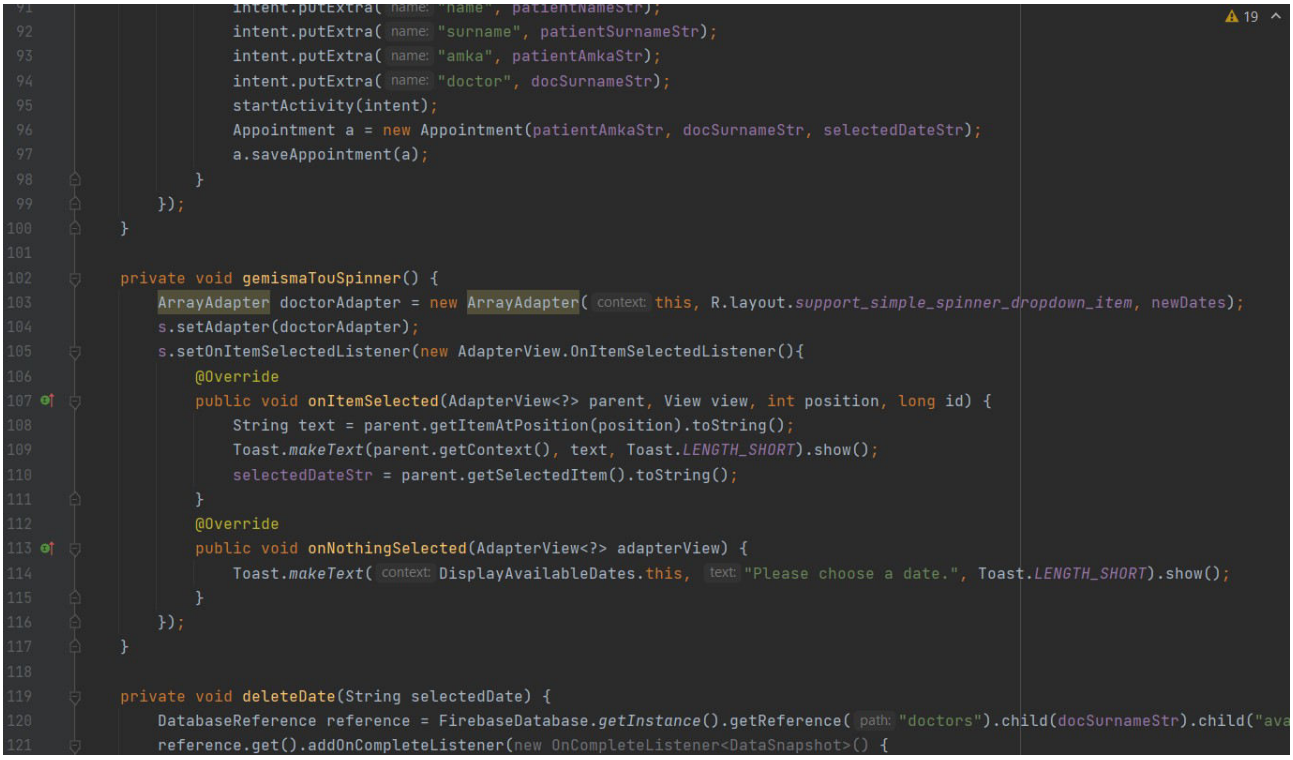
Εικόνα 3.25 - Κλάση BookAppointment (4)

```java
package com.example.hospitapp;

import ...

@RequiresApi(api = Build.VERSION_CODES.O)
public class DisplayAvailableDates extends AppCompatActivity implements AdapterView.OnItemSelectedListener {
    private static final int DAYS = 31;
    Button okButton;
    ArrayList <String> newDates = new ArrayList<>();
    EditText docSurname;
    Spinner s;
    String selectedDateStr, patientNameStr, patientSurnameStr, patientAmkaStr, docSurnameStr;

    @RequiresApi(api = Build.VERSION_CODES.O)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_display_available_dates);
        Bundle extras = getIntent().getExtras();
        patientNameStr = extras.getString( key: "name");
        patientSurnameStr = extras.getString( key: "surname");
        patientAmkaStr = extras.getString( key: "amka");
        docSurnameStr = extras.getString( key: "doctor");

        getSupportActionBar().setTitle("Book Appointment");

        s = (Spinner) findViewById(R.id.dateSpinner);
        okButton = (Button)findViewById(R.id.okButton);
        docSurname = (EditText)findViewById(R.id.searchSurname);
        BookAppointment ba = new BookAppointment();
```

Εικόνα 3.26 - Κλάση DisplayAvailableDates (1)

```java
        //Gemisma Imerominion
        DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "doctors");
        reference.child(docSurnameStr).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
            @Override
            public void onComplete(@NonNull Task<DataSnapshot> task) {
                if (task.isSuccessful()) {
                    System.out.println("Task succeed.");
                    if (task.getResult().exists()) {
                        DataSnapshot dataSnapshot = task.getResult();
                        for (int i = 0; i < DAYS; i++) {
                            newDates.add(String.valueOf(dataSnapshot.child("available").child(String.valueOf(i)).getValue()));
                            if(i==DAYS-1){
                                gemismaTouSpinner();
                            }
                        }
                    } else {
                        Toast.makeText( context: DisplayAvailableDates.this,  text: "No patient with such A.M.K.A. exist.", Toast.LENGTH
                    }
                } else {
                    Toast.makeText( context: DisplayAvailableDates.this,  text: "Wrong password, please try again.", Toast.LENGTH_SHORT)
                }
            }
            });

        //Gemisma tou Spinner
        okButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                deleteDate(selectedDateStr);
                Intent intent = new Intent( packageContext: DisplayAvailableDates.this, AppointmentInfo.class);
                intent.putExtra( name: "selectedDateStr", selectedDateStr);
                intent.putExtra( name: "name", patientNameStr);
```

Εικόνα 3.27 - Κλάση DisplayAvailableDates (2)



```
                intent.putExtra( name: "name", patientNameStr);
                intent.putExtra( name: "surname", patientSurnameStr);
                intent.putExtra( name: "amka", patientAmkaStr);
                intent.putExtra( name: "doctor", docSurnameStr);
                startActivity(intent);
                Appointment a = new Appointment(patientAmkaStr, docSurnameStr, selectedDateStr);
                a.saveAppointment(a);
            }
        });
    }

    private void gemismaTouSpinner() {
        ArrayAdapter doctorAdapter = new ArrayAdapter( context: this, R.layout.support_simple_spinner_dropdown_item, newDates);
        s.setAdapter(doctorAdapter);
        s.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener(){
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                String text = parent.getItemAtPosition(position).toString();
                Toast.makeText(parent.getContext(), text, Toast.LENGTH_SHORT).show();
                selectedDateStr = parent.getSelectedItem().toString();
            }
            @Override
            public void onNothingSelected(AdapterView<?> adapterView) {
                Toast.makeText( context: DisplayAvailableDates.this,  text: "Please choose a date.", Toast.LENGTH_SHORT).show();
            }
        });
    }

    private void deleteDate(String selectedDate) {
        DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "doctors").child(docSurnameStr).child("ava
        reference.get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
```

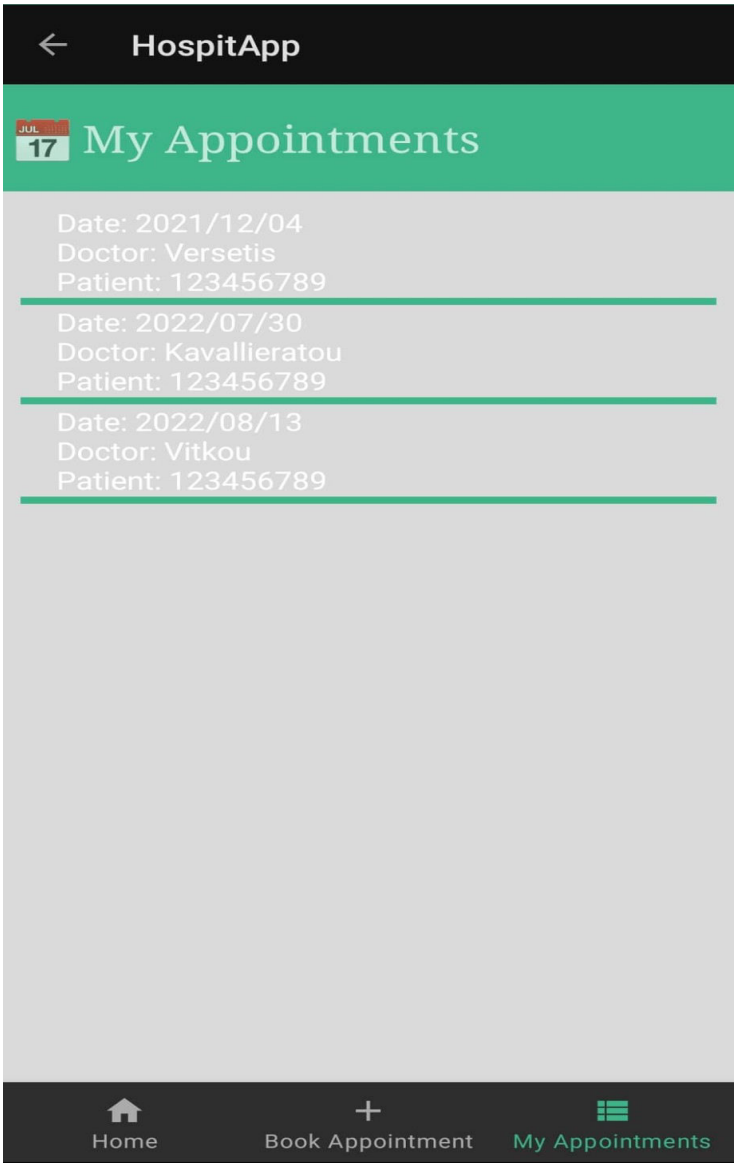
Εικόνα #3.28 - Κλάση DisplayAvailableDates (3)

```
        DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "doctors").child(docSurnameStr).ch
        reference.get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
            @Override
            public void onComplete(@NonNull Task<DataSnapshot> task) {
                if (task.isSuccessful()) {
                    System.out.println("Task succeed.");
                    if (task.getResult().exists()) {
                        DataSnapshot dataSnapshot = task.getResult();
                        for(int i = 0; i < 31; i++) {
                            if (dataSnapshot.child(String.valueOf(i)).getValue().equals(selectedDate)) {
                                reference.child(String.valueOf(i)).setValue("Unavailable");
                            }
                        }
                    } else {
                        Toast.makeText( context: DisplayAvailableDates.this,  text: "No patient with such A.M.K.A. exist.", Toast.LENGT
                    }
                } else {
                    Toast.makeText( context: DisplayAvailableDates.this,  text: "Wrong password, please try again.", Toast.LENGTH_SHORT
                }
            }
        });
    }

    @Override
    public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {}
    @Override
    public void onNothingSelected(AdapterView<?> adapterView) {}
}
```

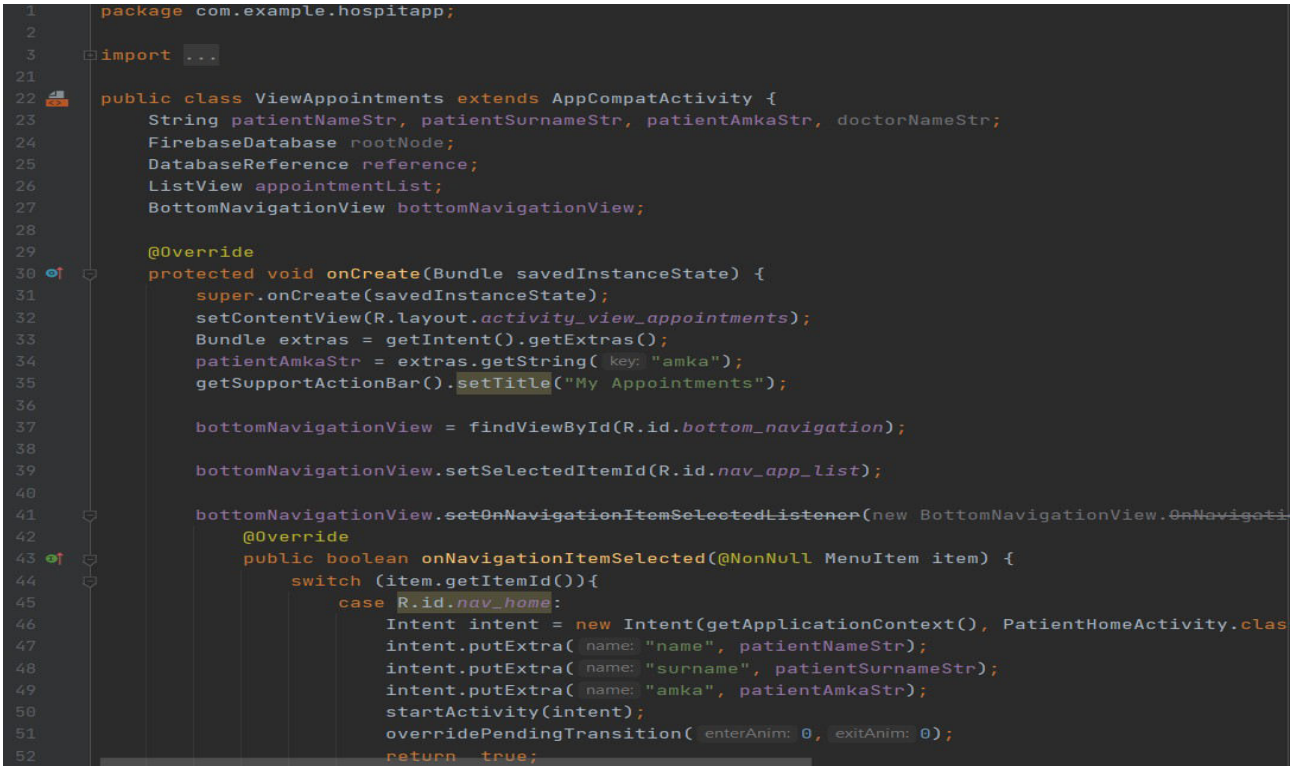
Εικόνα 3.29 - Κλάση DisplayAvailableDates (4)

3.3.6 Προβολή των κανονισμένων ραντεβού

Μια άλλη επιλογή που δίνεται στον χρήστη από το αρχικό μενού, είναι αυτή της προβολής των κανονισμένων ραντεβού του. Η λειτουργία αυτή επιτυγχάνεται μέσω της κλάσης *ViewAppointments.* Στην περίπτωση που ο χρήστης επιλέξει να δει τα προγραμματισμένα ραντεβού του, εμφανίζεται στην οθόνη του μία λίστα με αυτά αλλά και τα στοιχεία του καθενός, δηλαδή η ημερομηνία του ραντεβού, το επώνυμο του ιατρού και το Α.Μ.Κ.Α. του ασθενή. Στο σημείο αυτό, οι συναρτήσεις της κλάσης *ViewAppointments*, βρίσκουν τα στοιχεία όλων των ραντεβού του ασθενή στην βάση δεδομένων και τα μεταφέρουν στην οθόνη του χρήστη. Ακολουθούν στιγμιότυπο της οθόνης προβολής των ραντεβού και ο κώδικας της κλάσης.

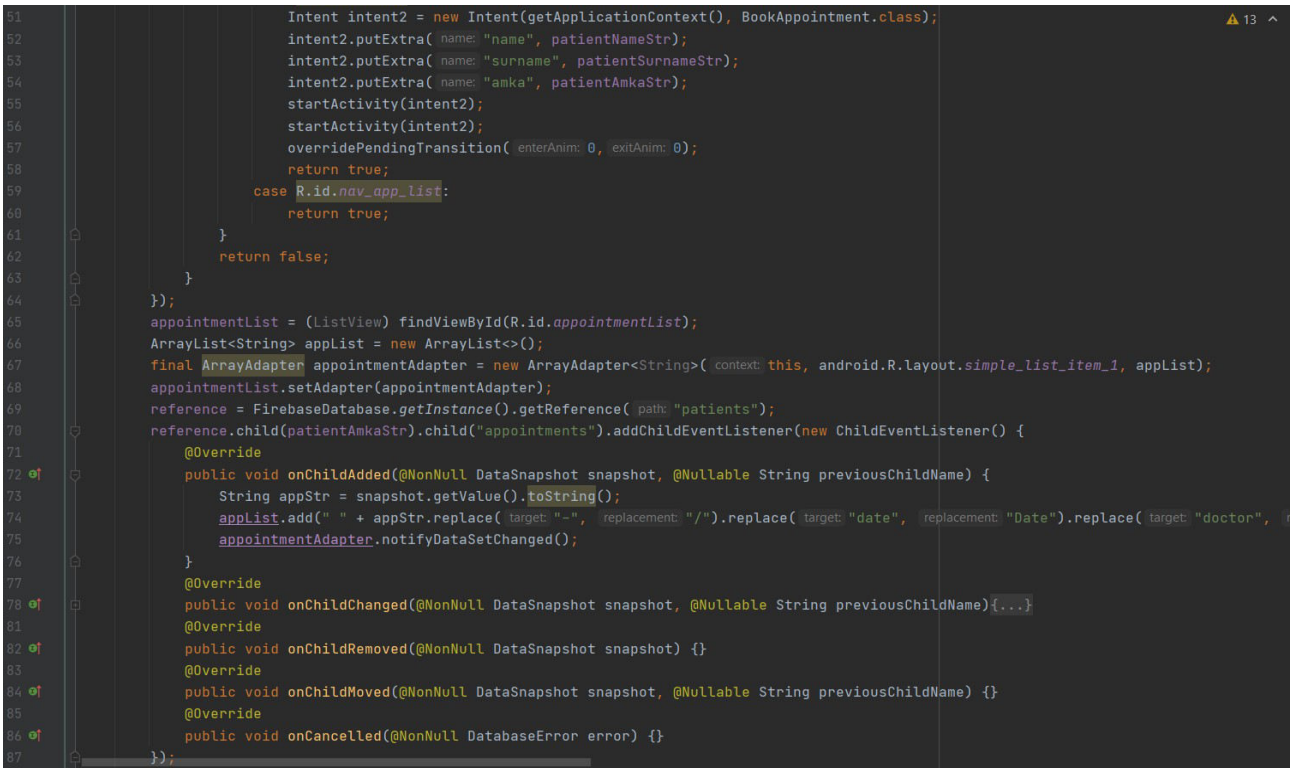


Εικόνα 3.30 - Λίστα ραντεβού ασθενή



```
package com.example.hospitapp;

import ...

public class ViewAppointments extends AppCompatActivity {
    String patientNameStr, patientSurnameStr, patientAmkaStr, doctorNameStr;
    FirebaseDatabase rootNode;
    DatabaseReference reference;
    ListView appointmentList;
    BottomNavigationView bottomNavigationView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_appointments);
        Bundle extras = getIntent().getExtras();
        patientAmkaStr = extras.getString( key: "amka");
        getSupportActionBar().setTitle("My Appointments");

        bottomNavigationView = findViewById(R.id.bottom_navigation);

        bottomNavigationView.setSelectedItemId(R.id.nav_app_list);

        bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigati
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                switch (item.getItemId()){
                    case R.id.nav_home:
                        Intent intent = new Intent(getApplicationContext(), PatientHomeActivity.clas
                        intent.putExtra( name: "name", patientNameStr);
                        intent.putExtra( name: "surname", patientSurnameStr);
                        intent.putExtra( name: "amka", patientAmkaStr);
                        startActivity(intent);
                        overridePendingTransition( enterAnim: 0, exitAnim: 0);
                        return  true;
```

Εικόνα 3.31 - Κλάση ViewAppointments(1)

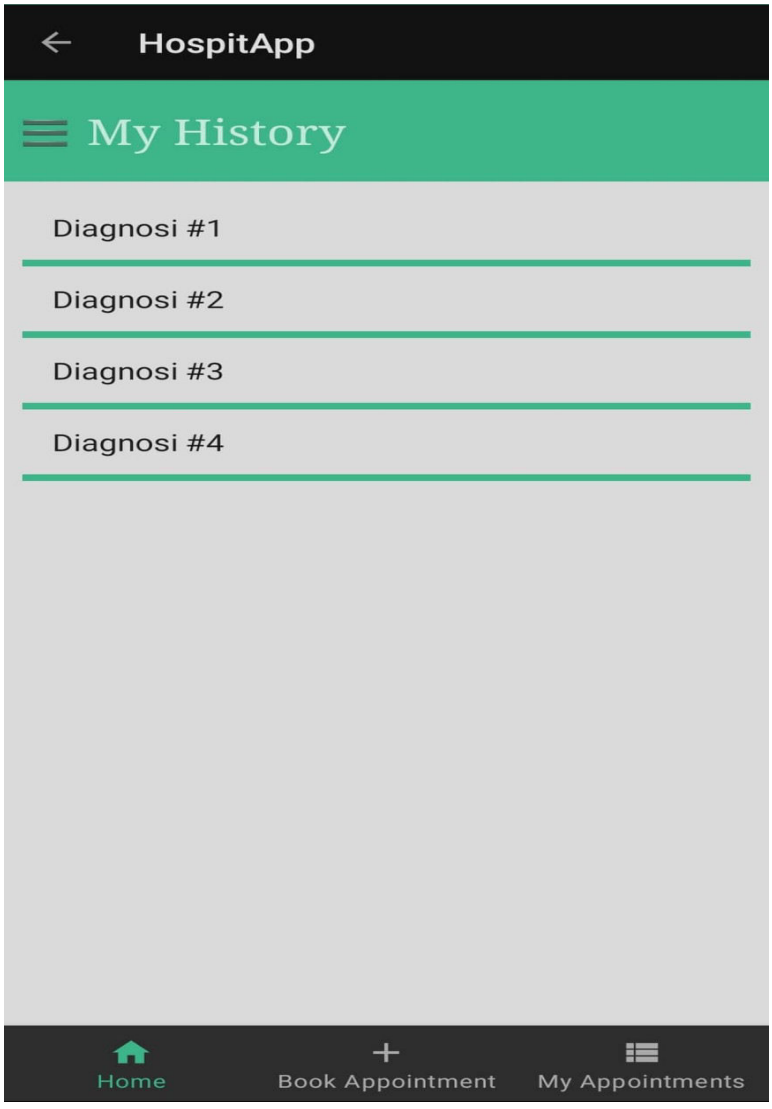


```
                    Intent intent2 = new Intent(getApplicationContext(), BookAppointment.class);
                    intent2.putExtra( name: "name", patientNameStr);
                    intent2.putExtra( name: "surname", patientSurnameStr);
                    intent2.putExtra( name: "amka", patientAmkaStr);
                    startActivity(intent2);
                    startActivity(intent2);
                    overridePendingTransition( enterAnim: 0, exitAnim: 0);
                    return true;
                case R.id.nav_app_list:
                    return true;
            }
            return false;
        }
    });
    appointmentList = (ListView) findViewById(R.id.appointmentList);
    ArrayList<String> appList = new ArrayList<>();
    final ArrayAdapter appointmentAdapter = new ArrayAdapter<String>( context: this, android.R.layout.simple_list_item_1, appList);
    appointmentList.setAdapter(appointmentAdapter);
    reference = FirebaseDatabase.getInstance().getReference( path: "patients");
    reference.child(patientAmkaStr).child("appointments").addChildEventListener(new ChildEventListener() {
        @Override
        public void onChildAdded(@NonNull DataSnapshot snapshot, @Nullable String previousChildName) {
            String appStr = snapshot.getValue().toString();
            appList.add(" " + appStr.replace( target: "-", replacement: "/").replace( target: "date", replacement: "Date").replace( target: "doctor",
            appointmentAdapter.notifyDataSetChanged();
        }
        @Override
        public void onChildChanged(@NonNull DataSnapshot snapshot, @Nullable String previousChildName){...}
        @Override
        public void onChildRemoved(@NonNull DataSnapshot snapshot) {}
        @Override
        public void onChildMoved(@NonNull DataSnapshot snapshot, @Nullable String previousChildName) {}
        @Override
        public void onCancelled(@NonNull DatabaseError error) {}
    });
```

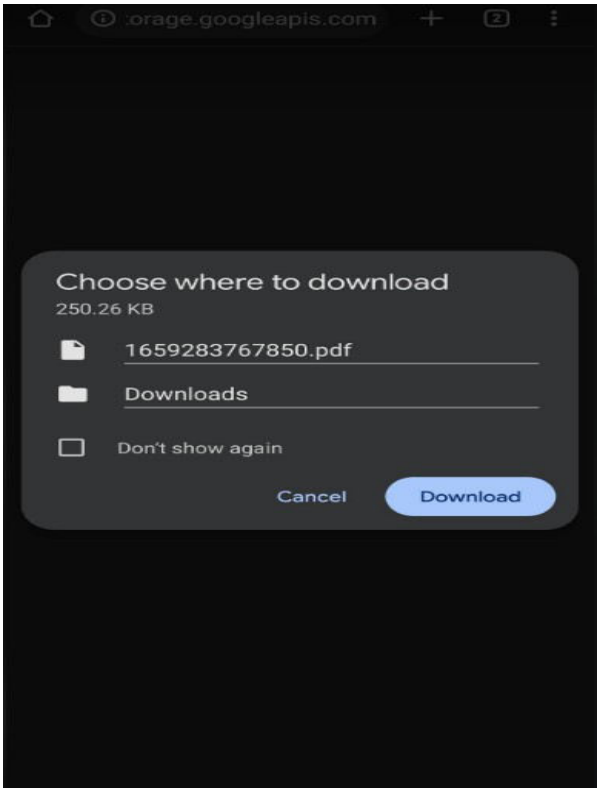
Εικόνα #3.32 - Κλάση ViewAppointments(2)

3.3.7 Λήψη και προβολή των καταχωρημένων διαγνώσεων

Η τρίτη και τελευταία λειτουργία της εφαρμογής, όσον αφορά τον χρήστη, είναι η προβολή ενός ιστορικού με τις διαγνώσεις που έχει λάβει έως τώρα ο ασθενής. Οι διαγνώσεις αυτές έχουν μεταφορτωθεί στην βάση δεδομένων από τους γιατρούς που επισκέφθηκε ο ασθενής, σε μορφή PDF αρχείου. Στην οθόνη του χρήστη εμφανίζεται μία λίστα με τα ονόματα των αρχείων των διαγνώσεων. Όταν ο χρήστης επιλέξει ένα απ' τα στοιχεία της λίστας, τότε η εφαρμογή μέσω της κλάσης «ViewHistory», βρίσκει το αντίστοιχο pdf αρχείο στην βάση και εμφανίζει μήνυμα στον χρήστη ζητώντας άδεια λήψης του αρχείου στον αποθηκευτικό χώρο της συσκευής. Εάν ο χρήστης συμφωνήσει στην λήψη, τότε το αρχείο ανοίγει και ο χρήστης μπορεί να το δει. Ακολουθούν στιγμιότυπα οθόνης των λειτουργιών που περιγράφηκαν στην παρούσα παράγραφο και ο κώδικας της κλάσης «*ViewHistory*».



Εικόνα #3.33 - Λίστα καταχωρημένων διαγνώσεων.



Εικόνα 3.34 - Ερώτηση λειτουργικού συστήματος συσκευής προορισμό αποθήκευσης της διάγνωσης.

```
1   package com.example.hospitapp;
2
3   import ...
26
27  public class ViewHistory extends AppCompatActivity {
28      String patientNameStr, patientSurnameStr, patientAmkaStr;
29      ListView PDFListView;
30      DatabaseReference databaseReference;
31      List<UploadPDF> uploadPDFs;
32
33      @Override
34      protected void onCreate(Bundle savedInstanceState) {
35          Bundle extras = getIntent().getExtras();
36          patientNameStr = extras.getString( key: "name");
37          patientSurnameStr = extras.getString( key: "surname");
38          patientAmkaStr = extras.getString( key: "amka");
39          super.onCreate(savedInstanceState);
40          setContentView(R.layout.activity_view_history);
41          getSupportActionBar().setTitle("HospitApp");
42
43          PDFListView = (ListView) findViewById(R.id.PDFListView);
44          uploadPDFs = new ArrayList<>();
45
46          viewAllFiles();
47          PDFListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
48              @Override
49              public void onItemClick(AdapterView<?> adapterView, View view, int position, long l) {
50                  UploadPDF uploadPDF = uploadPDFs.get(position);
51
52                  Intent intent = new Intent(Intent.ACTION_VIEW);
53                  intent.setData((Uri.parse(uploadPDF.getUrl())));
54                  startActivity(intent);
55                  Intent chooserIntent = Intent.createChooser(intent,  title: "Open Report");
```

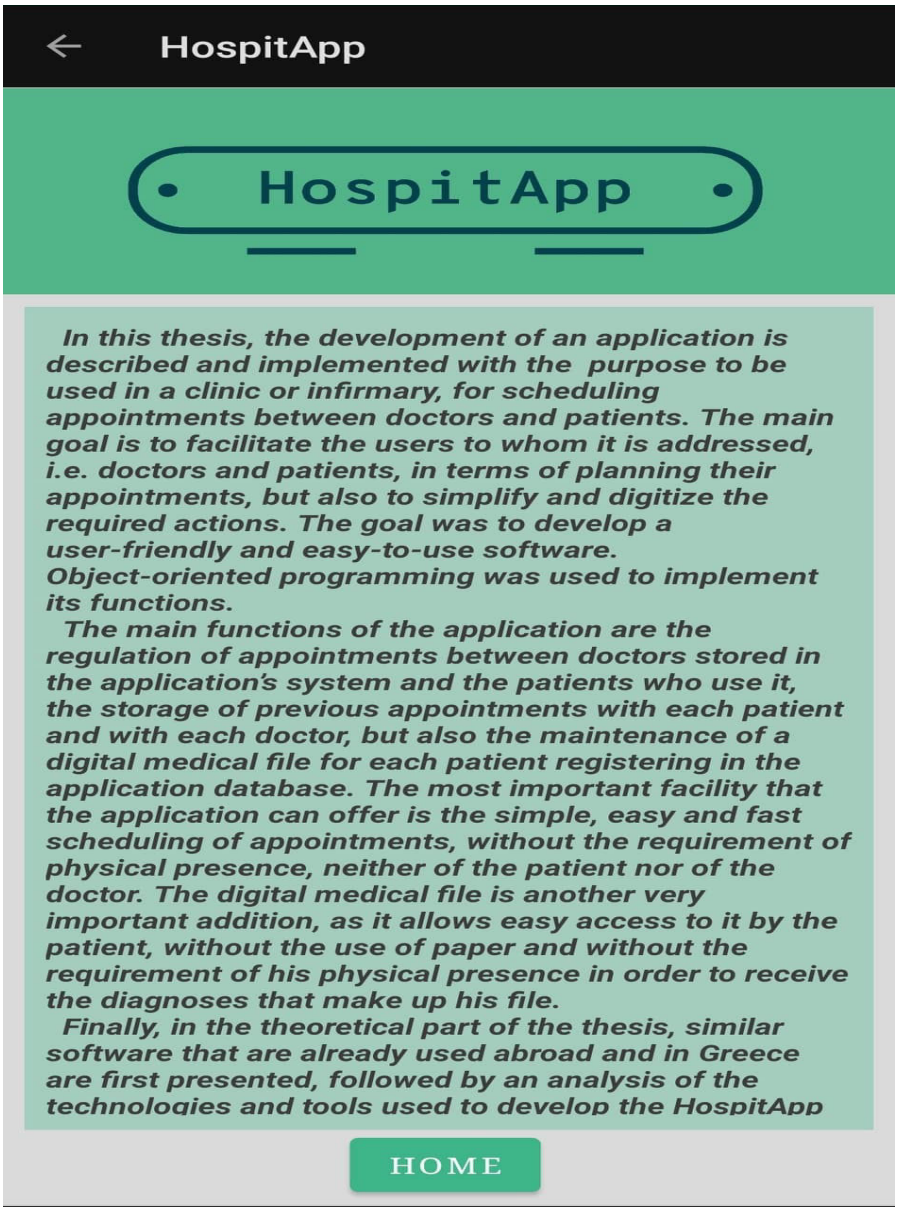
Εικόνα 3.35 - Κλάση ViewHistory (1)

```
                startActivity(chooserIntent);
            }
        });
    }

    private void viewAllFiles() {
        databaseReference = FirebaseDatabase.getInstance().getReference( path: "patients").child(patientAmkaStr).child("diagnoseis");
        databaseReference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {

                for (DataSnapshot postSnapshot : snapshot.getChildren()) {
                    UploadPDF uploadPDF = postSnapshot.getValue(UploadPDF.class);
                    uploadPDFs.add(uploadPDF);
                }

                String[] uploads = new String[uploadPDFs.size()];

                for (int i = 0; i < uploads.length; i++) {
                    uploads[i] = uploadPDFs.get(i).getName();
                }

                ArrayAdapter<String> adapter = new ArrayAdapter<~>(getApplicationContext(), android.R.layout.simple_list_item_1, uploads){};
                PDFListView.setAdapter(adapter);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {

            }
        });
    }
}
```

Εικόνα 3.36 - Κλάση ViewHistory (2)

3.3.8 About

Τελευταία επιλογή του χρήστη ώς ασθενής, είναι η προβολή κάποιον πληροφοριών σχετικά με την εφαρμογή. Επιλέγοντας το κουμπί «*About*», εμφανίζονται στην οθόνη κάποιες πληροφορίες σχετικά με την εφαρμογή. Ακολουθεί στιγμιότυπο της οθόνης «*About*».

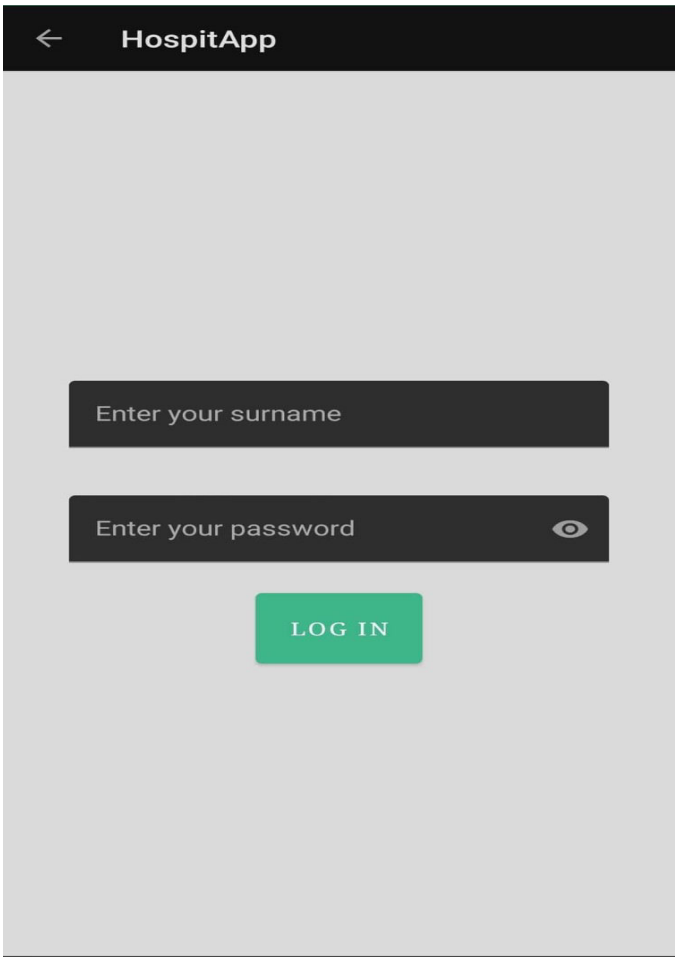


Εικόνα 3.37 – Οθόνη «About»

2η περίπτωση: Χρήση από ιατρό

3.3.9 Σύνδεση ιατρού

Αφού ο χρήστης επιλέξει να συνδεθεί ως γιατρός, μεταφέρεται στην επόμενη οθόνη, όπου καλείται να συνδεθεί, εισάγοντας το επώνυμό του και τον κωδικό χρήστη του. Σε περίπτωση επιλογής σύνδεσης από τον χρήστη, κλάση DoctorLogIn είναι υπεύθυνη για την επαλήθευση των στοιχείων που εισήχθησαν, αναζητώντας και συγκρίνοντάς τα με αυτά που υπάρχουν αποθηκευμένα στην βάση δεδομένων. Ο έλεγχος αυτός γίνεται με τις συναρτήσεις getDoctorInfo() και checkSurnamePassword(), καθώς η πρώτη αποθηκεύει σε μεταβλητές τα στοιχεία που εισήγαγε ο χρήστης και τα στέλνει στην δεύτερη η οποία κάνει την σύγκριση και την αναζήτηση στην βάση. Σε περίπτωση που κάποιο από τα απαιτούμενα πλαίσια κειμένου μείνει κενό, ή εάν το επώνυμο ή ο κωδικός χρήστη είναι λανθασμένα, εμφανίζεται κατάλληλο μήνυμα και δεν επιτρέπεται η πρόσβαση στον χρήστη. Στα παρακάτω στιγμιότυπα φαίνεται η οθόνη σύνδεσης και ο κώδικας της κλάσης.



Εικόνα 3.38 - Σύνδεση ιατρού

```java
package com.example.hospitapp;

import ...

public class DoctorLogIn extends AppCompatActivity {
    private
    Button login;
    TextInputLayout surname, password;
    String surnameStr, passwordStr, passwordOnDB, doctorNameStr, doctorSurnameStr;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_doctor_log_in);

        login = (Button) findViewById(R.id.LogInDoc);
        surname = (TextInputLayout) findViewById(R.id.editTextSurname);
        password = (TextInputLayout) findViewById(R.id.editTextPassword);

        login.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                surnameStr = surname.getEditText().getText().toString().trim();
                passwordStr = password.getEditText().getText().toString();

                if(surnameStr.isEmpty()||passwordStr.isEmpty())
                    Toast.makeText( context: DoctorLogIn.this,  text: "Please fill all fields.", Toast.LEN
                else{
                    getDoctorInfo();
                    checkSurnamePassword();
                    surname.getEditText().setText("");
                    password.getEditText().setText("");
```

Εικόνα 3.39 - Κλάση DoctorLogin (1)

```java
    private void getDoctorInfo() {
        DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "doctors");
        reference.child(surnameStr).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>
            @Override
            public void onComplete(@NonNull Task<DataSnapshot> task) {
                if (task.isSuccessful()) {
                    System.out.println("Task succeed.");
                    if (task.getResult().exists()) {
                        DataSnapshot dataSnapshot = task.getResult();
                        doctorNameStr = dataSnapshot.child("name").getValue().toString();
                        doctorSurnameStr = dataSnapshot.child("surname").getValue().toString();
                    } else {
                        Toast.makeText( context: DoctorLogIn.this,  text: "No doctor with such surname exi
                    }
                } else {
                    System.out.println(task.isSuccessful());
                    Toast.makeText( context: DoctorLogIn.this,  text: "Wrong password, please try again.",
                }
            }
        });
    }

    public void checkSurnamePassword() {
        DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "doctors");
        reference.child(surnameStr).get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>
            @RequiresApi(api = Build.VERSION_CODES.O)
            @Override
            public void onComplete(@NonNull Task<DataSnapshot> task) {
                if(task.isSuccessful()){
                    if(task.getResult().exists()) {
                        DataSnapshot dataSnapshot = task.getResult();
                        passwordOnDB = String.valueOf(dataSnapshot.child("password").getValue());
```

Εικόνα 3.40 - Κλάση DoctorLogin (2)

```
85                      passwordOnDB = String.valueOf(dataSnapshot.child("password").getValue());
86                      if (passwordOnDB.equals(passwordStr)) {
87                          Toast.makeText( context: DoctorLogIn.this,  text: "Logged in successfully!",
88                          Intent intent = new Intent( packageContext: DoctorLogIn.this, DoctorHomeActi
89                          intent.putExtra( name: "name", doctorNameStr);
90                          intent.putExtra( name: "surname", doctorSurnameStr);
91                          startActivity(intent);
92                      }
93                      else {
94                          Toast.makeText( context: DoctorLogIn.this,  text: "CHECK: Wrong password, plea
95                      }
96                  }
97                  else{
98                      Toast.makeText( context: DoctorLogIn.this,  text: "No doctor with such surname exi
99                  }
100             }
101             else{
102                 Toast.makeText( context: DoctorLogIn.this,  text: "Wrong password, please try again.",
103             }
104         }
105     });
106 }
107 }
```

Εικόνα 3.41 - Κλάση DoctorLogin (3)

3.3.10 Βασικό μενού ιατρού

Μόλις ο γιατρός συνδεθεί επιτυχώς στην εφαρμογή, εμφανίζεται το βασικό μενού της εφαρμογής μέσω της κλάσης *DoctorHomeActivity*. Το μενού αυτό περιέχει δύο βασικά buttons, τα οποία οδηγούν στην προβολή των ραντεβού του γιατρού και στην μεταφόρτωση μιας νέας διάγνωσης, στον φάκελο κάποιου ασθενή στην βάση δεδομένων. Μέσω των συναρτήσεων της κλάσης *DoctorHomeActivity* και με βάση τις επιλογές του χρήστη, καλούνται οι κατάλληλες κλάσεις για την διεκπεραίωση των επιθυμιών του. Εάν ο χρήστης επιλέξει «View Appointments», καλείται η συνάρτηση «openViewDoctorAppointments» και αντίστοιχα αν επιλέξει «Upload PDF», καλείται η «UploadDiagnosi» Παρακάτω παρουσιάζεται ο κώδικας υλοποίησης των λειτουργιών της κλάσης DoctorHomeActivity και στιγμιότυπο της αρχικής οθόνης γιατρού.

Εικόνα 3.42 - Βασικό μενού ιατρού

```
package com.example.hospitapp;

import ...

public class DoctorHomeActivity extends AppCompatActivity {
    Button viewAp, uploadPDF;
    String doctorNameStr, doctorSurnameStr;
    BottomNavigationView bottomNavigationView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_doctor_home);
        getSupportActionBar().setTitle("Home");

        Bundle extras = getIntent().getExtras();
        doctorNameStr = extras.getString( key: "name");
        doctorSurnameStr = extras.getString( key: "surname");

        viewAp = (Button) findViewById(R.id.myDoctorAppointmerts);

        bottomNavigationView = findViewById(R.id.bottom_navigation_doc);
        bottomNavigationView.setSelectedItemId(R.id.nav_home_doc);

        bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigati
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                switch (item.getItemId()){
                    case R.id.nav_home_doc:
                        return  true;
                    case R.id.nav_app_list_doc:
                        Intent intent = new Intent(getApplicationContext(), ViewDoctorAppointments.c
                        intent.putExtra( name: "name", doctorNameStr);
                        intent.putExtra( name: "surname", doctorSurnameStr);
```

Εικόνα 3.43 - Κλάση DoctorHomeActivity (1)

```
                        intent.putExtra( name: "surname", doctorSurnameStr);
                        startActivity(intent);
                        overridePendingTransition( enterAnim: 0, exitAnim: 0);
                        return true;
                    case R.id.upload_pdf:
                        Intent intent2 = new Intent(getApplicationContext(), UploadDiagnosi.class);
                        intent2.putExtra( name: "name", doctorNameStr);
                        intent2.putExtra( name: "surname", doctorSurnameStr);
                        startActivity(intent2);
                        overridePendingTransition( enterAnim: 0, exitAnim: 0);
                        return true;
                }
                return false;
            }
        });

        viewAp.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) { openViewDoctorAppointments(); }
        });

        uploadPDF = (Button) findViewById(R.id.uploadDiagnosi);

        uploadPDF.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) { openUploadPDF(); }
        });
    }

    private void openUploadPDF() {
        Intent intent = new Intent( packageContext: DoctorHomeActivity.this, UploadDiagnosi.class);
        intent.putExtra( name: "name", doctorNameStr);
        intent.putExtra( name: "surname", doctorSurnameStr);
```

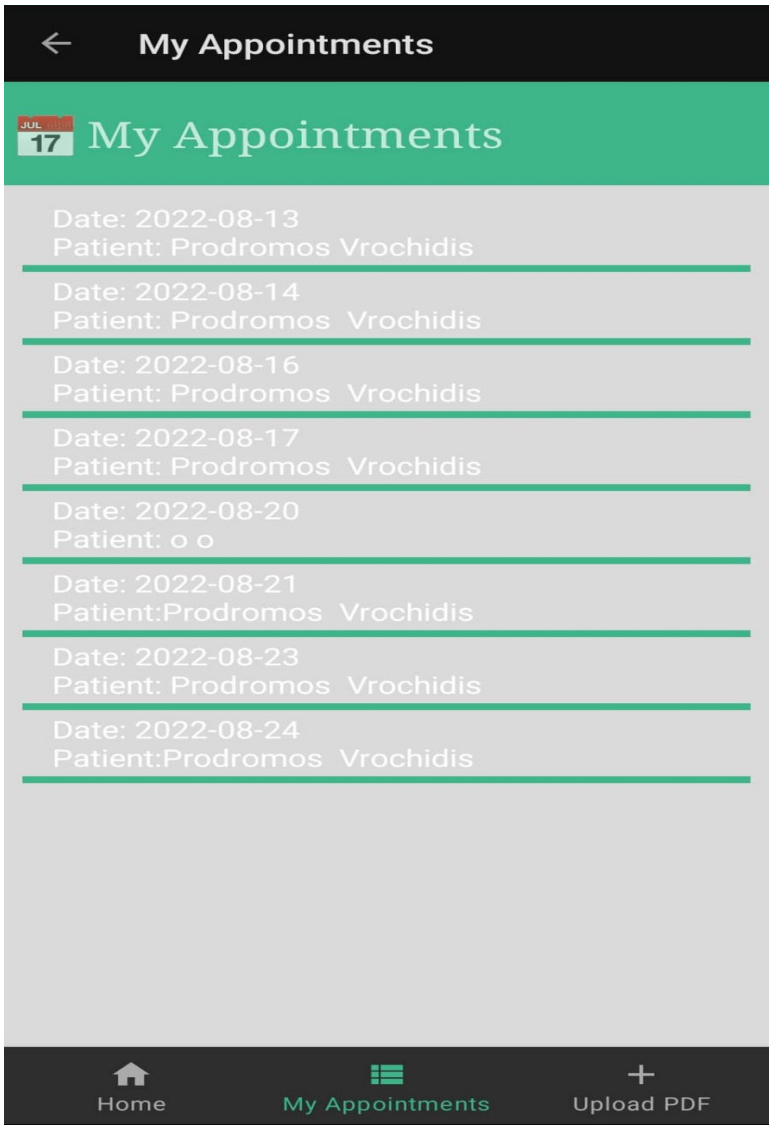
Εικόνα 3.44 - Κλάση DoctorHomeActivity (2)

```
75      private void openUploadPDF() {
76          Intent intent = new Intent( packageContext: DoctorHomeActivity.this, UploadDiagnosi.class);
77          intent.putExtra( name: "name", doctorNameStr);
78          intent.putExtra( name: "surname", doctorSurnameStr);
79          startActivity(intent);
80      }
81
82      private void openViewDoctorAppointments() {
83          Intent intent = new Intent( packageContext: DoctorHomeActivity.this, ViewDoctorAppointments.class);
84          intent.putExtra( name: "name", doctorNameStr);
85          intent.putExtra( name: "surname", doctorSurnameStr);
86          startActivity(intent);
87      }
88  }
```

Εικόνα 3.45 - Κλάση DoctorHomeActivity (3)

3.3.11 Προβολή των κανονισμένων ραντεβού

Η πρώτη επιλογή που δίνεται στον χρήστη από το αρχικό μενού, είναι αυτή της προβολής των κανονισμένων ραντεβού του. Η λειτουργία αυτή επιτυγχάνεται μέσω της κλάσης *ViewDoctorAppointments.* Στην περίπτωση που ο χρήστης επιλέξει να δει τα προγραμματισμένα ραντεβού του, εμφανίζεται στην οθόνη του μία λίστα με αυτά αλλά και τα στοιχεία του καθενός, δηλαδή η ημερομηνία του ραντεβού και το ονοματεπώνυμο του ασθενή. Στο σημείο αυτό, οι συναρτήσεις της κλάσης *ViewDoctorAppointments*, βρίσκουν τα στοιχεία όλων των ραντεβού του ασθενή στην βάση δεδομένων και τα μεταφέρουν στην οθόνη του χρήστη. Ακολουθούν στιγμιότυπο της οθόνης προβολής των ραντεβού και ο κώδικας της κλάσης.

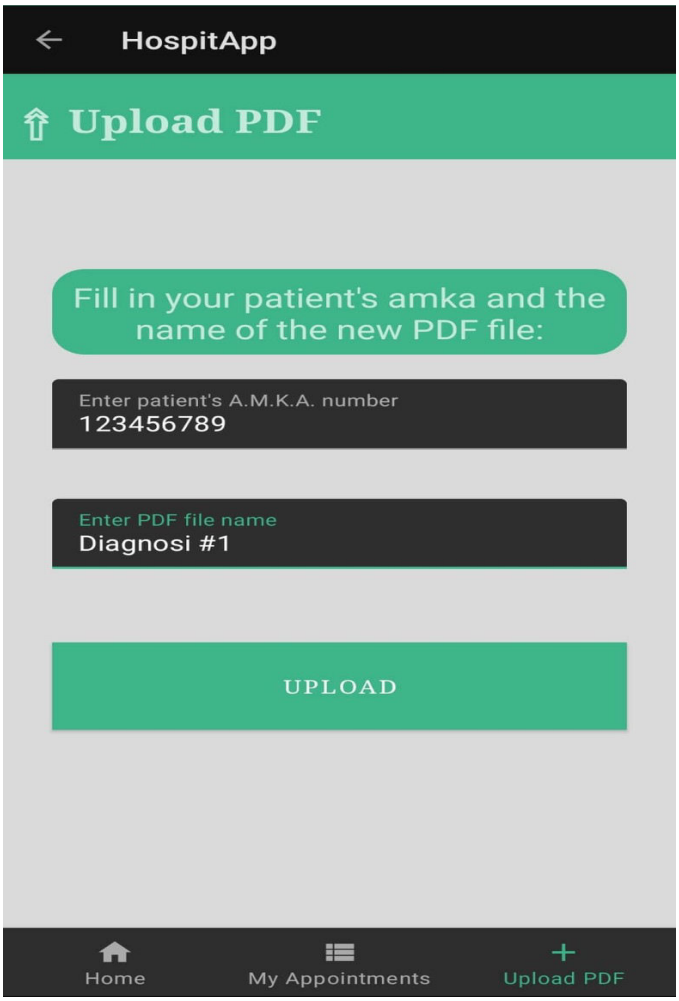


Εικόνα 3.46 - Λίστα ραντεβού γιατρούς

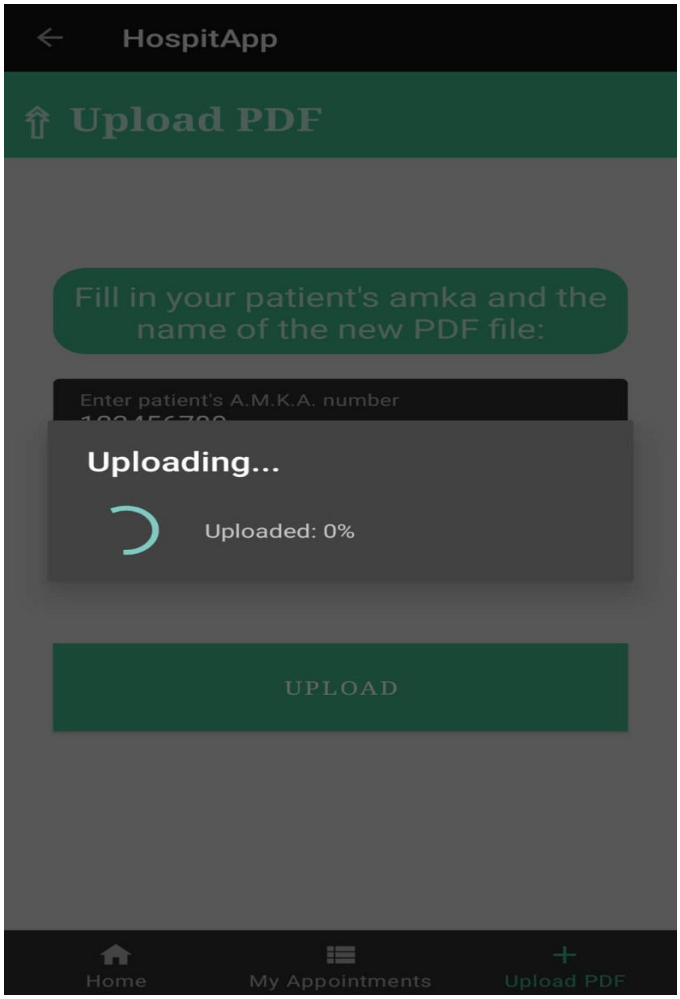
3.3.12 Μεταφόρτωση νέας διάγνωσης

Η δεύτερη επιλογή που δίνεται στον χρήστη από το μενού του γιατρού, είναι αυτή της μεταφόρτωσης μιας νέας διάγνωσης στον φάκελο κάποιου ασθενή στην βάση δεδομένων.

Οι απαραίτητες ενέργειες για την υλοποίηση της λειτουργίας αυτής, γίνονται από την κλάση «*UploadDiagnosi*». Μόλις ο χρήστης επιλέξει «*Upload PDF*», μεταφέρεται σε μία νέα οθόνη, όπου του ζητείται να συμπληρώσει τον A.M.K.A. του ασθενή για τον οποίο έγινε η διάγνωση, αλλά και το όνομα με το οποίο θα αποθηκευτεί αυτή στην βάση δεδομένων. Αφού ο χρήστης επιλέξει «*UPLOAD*», η εφαρμογή τον μεταφέρει στην μνήμη και τα αρχεία της συσκευής του, απ' όπου καλείται να επιλέξει το αρχείο μορφής pdf με την διάγνωση για τον ασθενή. Μόλις το επιλέξει, το αρχείο μεταφορτώνεται στην βάση δεδομένων. Ακολουθούν στιγμιότυπα οθόνης της λειτουργίας αυτής και η κλάση «*UploadDiagnosi*».



Εικόνα 3.47 - Επιλογή ασθενή και ονόματος αρχείου μεταφόρτωση



Εικόνα 3.48 - Αναμονή έως την του αρχείου

```java
package com.example.hospitapp;

import ...

public class UploadDiagnosi extends AppCompatActivity {
    Button uploadBtn;
    TextInputLayout pdfName, patientAMKA;
    String doctorNameStr, doctorSurnameStr, patientAmkaStr;
    StorageReference storageReference;
    DatabaseReference databaseReference;
    BottomNavigationView bottomNavigationView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_upload_diagnosi);

        Bundle extras = getIntent().getExtras();
        doctorNameStr = extras.getString( key: "name");
        doctorSurnameStr = extras.getString( key: "surname");

        uploadBtn = (Button) findViewById(R.id.uploadBtn);
        pdfName = (TextInputLayout) findViewById(R.id.PDF_name);
        patientAMKA = (TextInputLayout) findViewById(R.id.patient_amka);

        bottomNavigationView = findViewById(R.id.bottom_navigation_doc);
        bottomNavigationView.setSelectedItemId(R.id.upload_pdf);

        bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigati
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                switch (item.getItemId()){
```

Εικόνα 3.49 - Κλάση UploadDiagnosi (1)

```java
                        Intent intent2 = new Intent(getApplicationContext(), DoctorHomeActivity.clas
                        intent2.putExtra( name: "name", doctorNameStr);
                        intent2.putExtra( name: "surname", doctorSurnameStr);
                        startActivity(intent2);
                        overridePendingTransition( enterAnim: 0, exitAnim: 0);
                        return  true;
                    case R.id.nav_app_list_doc:
                        Intent intent = new Intent(getApplicationContext(), ViewDoctorAppointments.c
                        intent.putExtra( name: "name", doctorNameStr);
                        intent.putExtra( name: "surname", doctorSurnameStr);
                        startActivity(intent);
                        overridePendingTransition( enterAnim: 0, exitAnim: 0);
                        return true;
                    case R.id.upload_pdf:
                        return true;
                }
                return false;
            }
        });

        uploadBtn.setOnClickListener(new View.OnClickListener() {
            @RequiresApi(api = Build.VERSION_CODES.LOLLIPOP_MR1)
            @Override
            public void onClick(View view) {
                selectPDFFile();
                patientAmkaStr = patientAMKA.getEditText().getText().toString().trim();
                storageReference = FirebaseStorage.getInstance().getReference();
                databaseReference = FirebaseDatabase.getInstance().getReference( path: "patients").chi
            }
        });
    }
```

Εικόνα 3.50 - Κλάση UploadDiagnosi (2)

3.4 Η βάση δεδομένων

Σε αυτό το κεφάλαιο θα περιγραφή η υλοποίηση και λειτουργία της βάσης δεδομένων που χρησιμοποιείται για την λειτουργία της εφαρμογής HospitApp.

3.4.1 Επιλογή Firebase

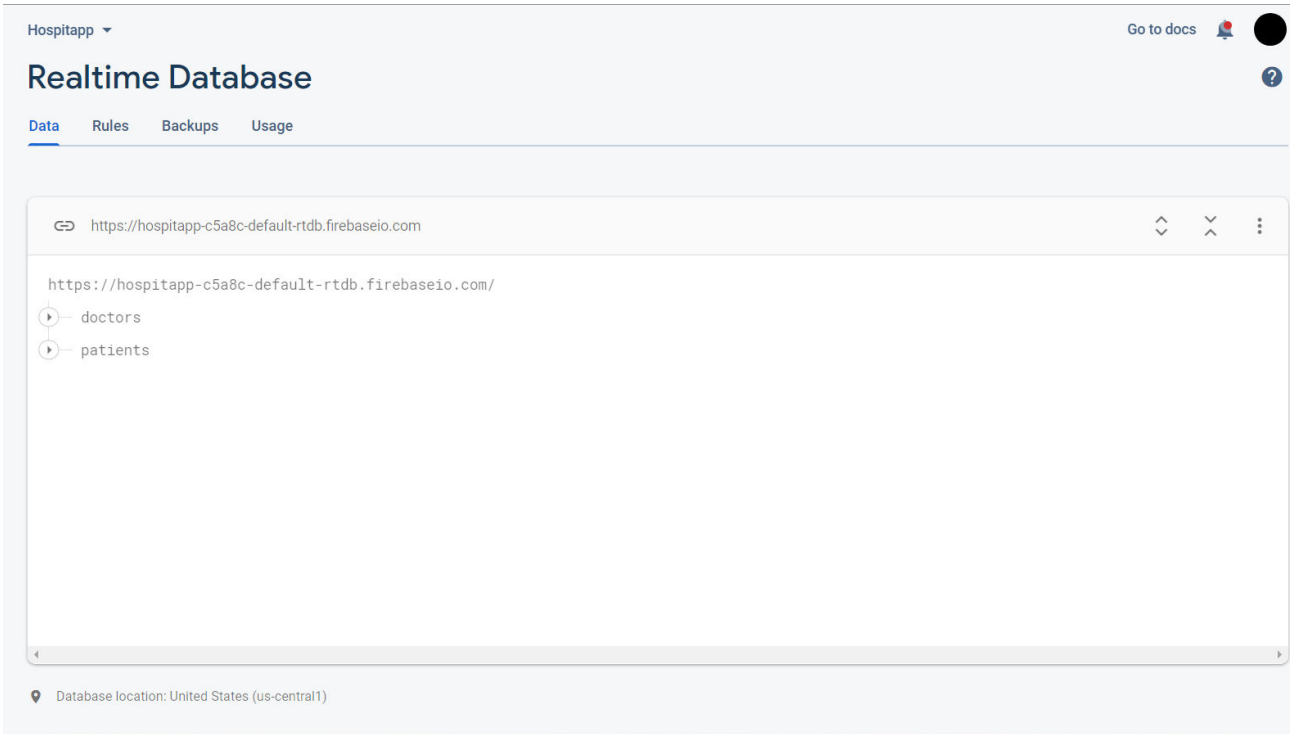
Ο κύριος λόγος για τον οποίο επιλέχθηκε η χρήση μιας πραγματικού χρόνου βάσης δεδομένων (realtime database) μέσω του Firebase, είναι πως επιτρέπει στις εφαρμογές να προσεγγίζουν δεδομένα πολλαπλών πλατφορμών σε πραγματικό χρόνο και οι όποιες αλλαγές γίνονται στην βάση, εμφανίζονται απευθείας και στον χρήστη. Η μεταφορά αρχείων από την συσκευή στην βάση γίνεται εύκολα και γρήγορα. Ένα ακόμη πλεονέκτημα της χρήσης μια realtime database, είναι πως δίνεται η δυνατότητα για την διεκπεραίωση εργασιών χωρίς σύνδεση στο Διαδίκτυο. Ακόμη και εκτός σύνδεσης τα δεδομένα εξακολουθούν να αποθηκεύονται προσωρινά στη μνήμη της συσκευής όταν μόλις η συσκευή συνδεθεί και πάλι στο Διαδίκτυο, ξεκινάει ο συγχρονισμός τους.

Ένα άλλο πλεονέκτημα της χρήσης Firebase είναι οι ασφαλείς και γρήγορες υπηρεσίες «hosting» (φιλοξενίας) του. Το hosting της Firebase υποστηρίζει όλους τους τύπους περιεχομένου, συμπεριλαμβανομένων των εφαρμογών ιστού, δυναμικού και στατικού περιεχομένου.

Ακόμη ο SSL μηδενικής διαμόρφωσης ενισχύει την ασφάλεια της παράδοσης περιεχομένου. Για να διατηρείται ο προσαρμοσμένος τομέας ασφαλής από εξωτερικές απειλές, η χρήση της δωρεάν πιστοποίησης SSL του Firebase είναι επίσης επωφελής. Επίσης, το Firebase CLI βοηθά τους προγραμματιστές να κάνουν την εφαρμογή τους ζωντανή και να εκτελείται μέσα σε δευτερόλεπτα.

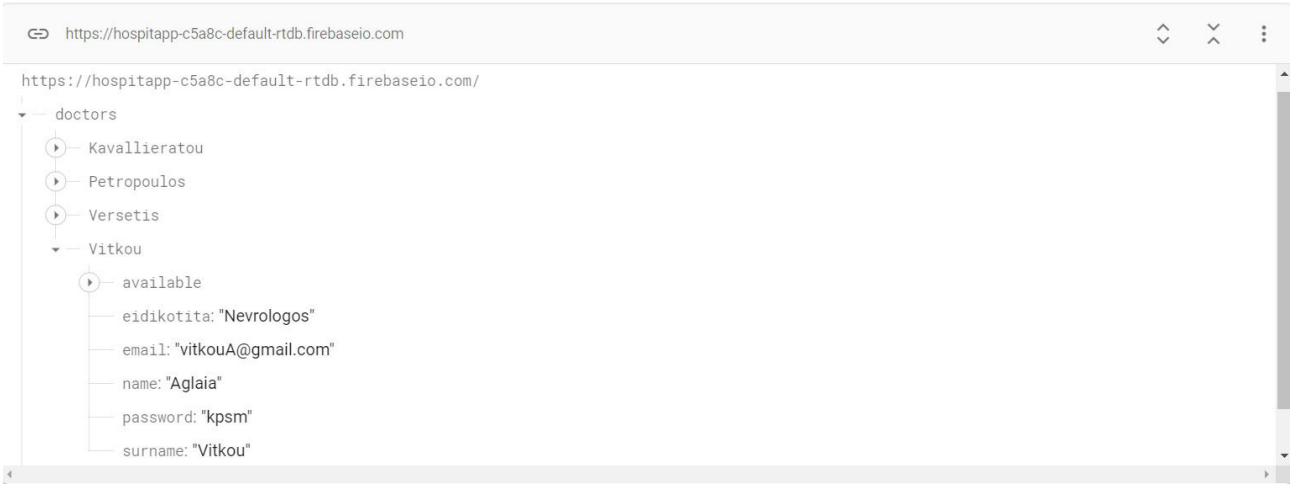
3.4.2 HospitApp realtime database

Μέσω των παρακάτω στιγμιοτύπων, ακολουθεί μια περιγραφή της βάσης δεδομένων της εφαρμογής HospitApp. Παράλληλα με την παράθεση των εικόνων, θα γίνεται και επεξήγηση των λειτουργιών της.



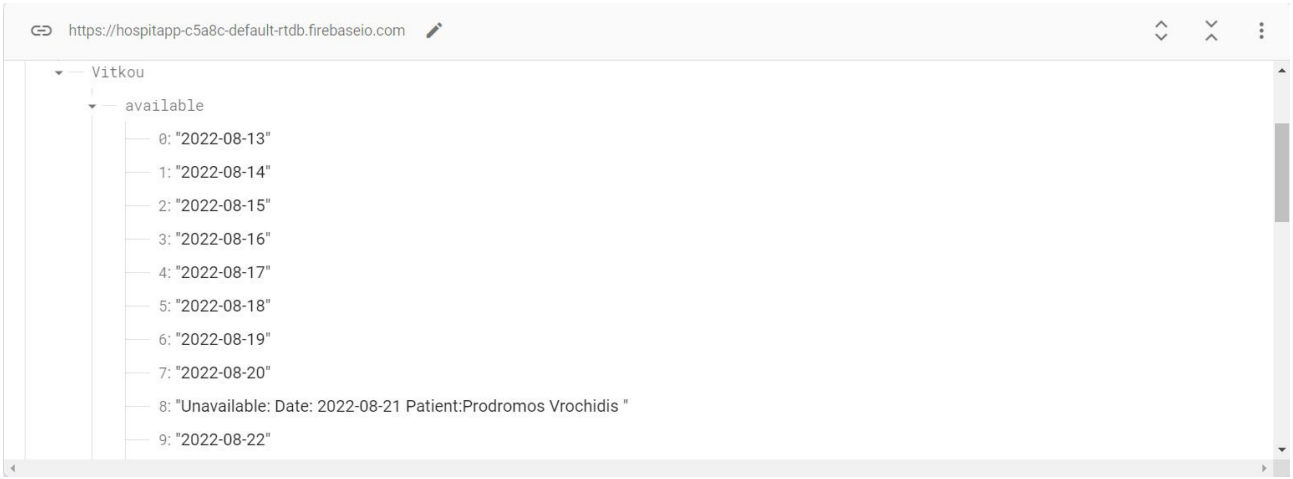
Εικόνα 4.1 - Αρχικοί φάκελοι της βάσης δεδομένων

Στην εικόνα #, φαίνονται οι δύο αρχικοί φάκελοι της βάσης δεδομένων. Ο ένας αφορά τους γιατρούς ενώ ο άλλος τους ασθενείς, που είναι εγγεγραμμένοι στην εφαρμογή. Μέσα περιέχουν εγγραφές γιατρών και ασθενών αντίστοιχα οι οποίες με τη σειρά τους περιέχουν τα στοιχεία της κάθε εγγραφής.



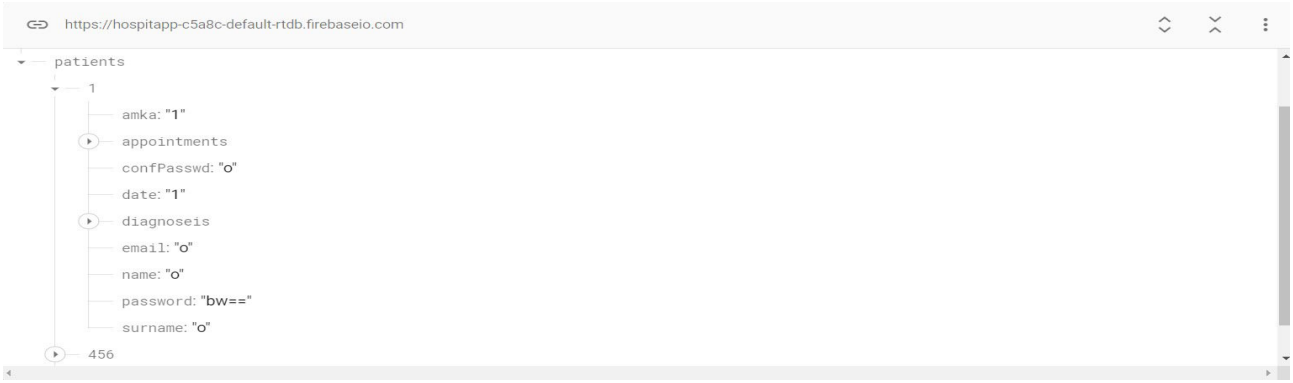
Εικόνα 4.2 - Εγγραφές ιατρών

Στην παραπάνω εικόνα φαίνεται ένα παράδειγμα εγγραφής ιατρού. Κάτω από το επώνυμό του υπάρχουν τα στοιχεία του που έχουν καταχωρηθεί στην εφαρμογή (όνομα, επώνυμο, ειδικότητα, κωδικός χρήστη, και e-mail), αλλά και ένας φάκελος με τις διαθέσιμες προς ραντεβού ημερομηνίες που διαθέτει, για τον τρέχοντα μήνα.



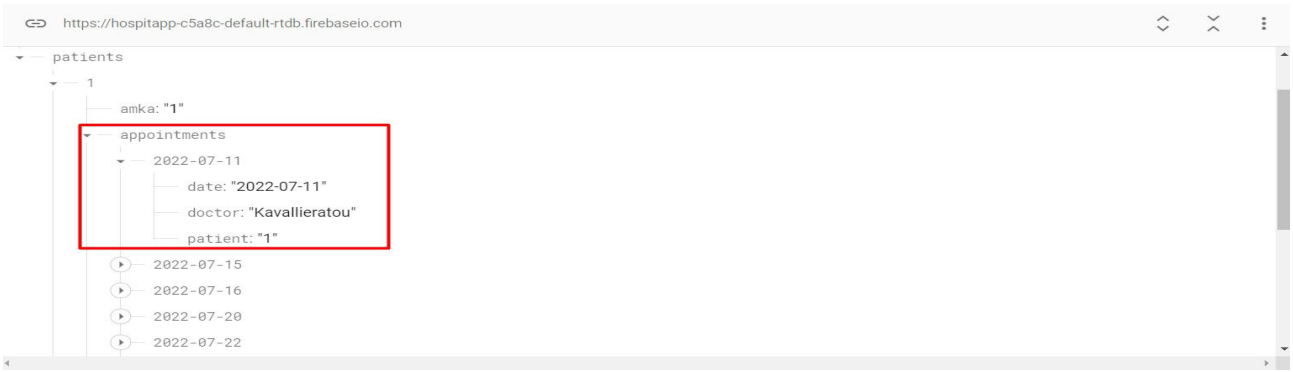
Εικόνα 4.3 - Φάκελος διαθέσιμων ημερομηνιών γιατρού

Όπως φαίνεται και στο παραπάνω στιγμιότυπο από την βάση δεδομένων, στις ημερομηνίες που δεν είναι διαθέσιμες, αναγράφεται και ο ασθενής με τον οποίο ο γιατρός έχει κανονισμένο ραντεβού.

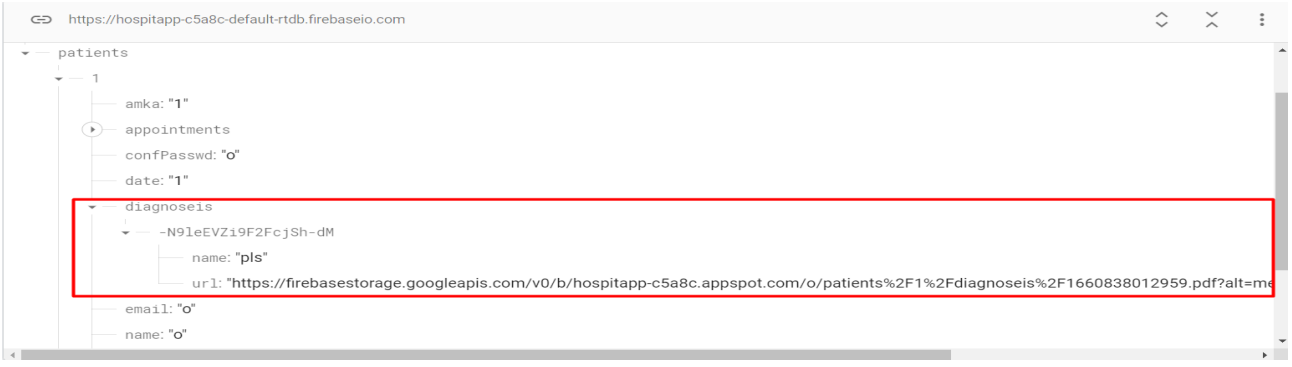


Εικόνα 4.4 - Εγγραφές ασθενών

Στην εικόνα #, φαίνεται ένα παράδειγμα εγγραφής ασθενή. Οι ασθενείς είναι καταχωρημένοι με βάση τον Α.Μ.Κ.Α. τους, και κάτω από το καθένα υπάρχουν τα υπόλοιπα στοιχεία του που έχουν καταχωρηθεί στην βάση δεδομένων (όνομα, επώνυμο, κωδικός χρήστη, e-mail), αλλά και δύο υποφάκελοι. Ο ένας περιέχει τις ημερομηνίες των ραντεβού του ασθενή και για κάθε έναν τα στοιχεία του (ημερομηνία, επώνυμο γιατρού, Α.Μ.Κ.Α. ασθενή). Ο άλλος υποφάκελος περιέχει τις διαγνώσεις που έχουν καταχωρηθεί στην βάση από κάποιον γιατρό, μέσω της εφαρμογής. Για κάθε εγγραφή διάγνωσης, υπάρχει το όνομα της και μια διεύθυνση url μέσω της οποίας μπορεί ο χρήστης της εφαρμογής να κατεβάσει το pdf αρχείο με την διάγνωση.



Εικόνα 4.5 - Εγγραφή ραντεβού ασθενή.



Εικόνα 46 - Εγγραφή διάγνωσης

4 *Αποτελέσματα*

Όπως έχει προαναφερθεί αποτέλεσμα της παρούσας εργασίας, είναι μια πλήρως λειτουργική και εύχρηστη εφαρμογή, η οποία μπορεί να χρησιμοποιηθεί σε μια κλινική ή ιατρείο, για την οργάνωση των ραντεβού των ασθενών και την διατήρηση ψηφιακού ιατρικού φακέλου για κάθε έναν από αυτούς. Απώτερος της εφαρμογής είναι η διευκόλυνση των χρηστών της μέσω της απλούστευσης των διαδικασιών που σχετίζονται με ένα ιατρείο.

Πριν αλλά και κατά την διάρκεια της υλοποίησης, μελετήθηκε ο τρόπος λειτουργίας και τα χαρακτηριστικά άλλων παρόμοιων εφαρμογών που ήδη υπάρχουν. Από αυτές, συγκεντρώθηκαν τα περισσότερα πλεονεκτήματα που εντοπίστηκαν και δόθηκε ιδιαίτερη βαρύτητα στην υιοθέτησή τους από την παρούσα υλοποίηση. Αποτελέσματα λοιπόν της παραπάνω αναζήτησης είναι τα εξής:

- Η HospitApp είναι εφαρμογή με φιλικό και απλό γραφικό περιβάλλον. Αυτό δίνει την δυνατότητα στον χρήστη να περιηγηθεί σε αυτή και να την χρησιμοποιήσει γρήγορα και αποτελεσματικά.
- Η χρήση της εφαρμογής δεν απαιτεί από τον χρήστη γνώσεις Πληροφορικής ή τεχνολογιών γενικότερα.
- Όλες οι λειτουργίες της εφαρμογής εκτελούνται απλά και γρήγορα. Ο χρήστης έχει την δυνατότητα να αξιοποιήσει τις λειτουργίες της ακολουθώντας μικρά, απλά και σαφώς ορισμένα βήματα.
- Απευθύνεται σε αρκετά μεγάλη μερίδα χρηστών smartphones, αφού πρόκειται για android εφαρμογή αλλά και η έκδοση android που χρησιμοποιήθηκε (Android Lollipop 5.0) καλύπτει ένα εύρος συσκευών android της τάξης του 98.6%.
- Τα προσωπικά δεδομένα των χρηστών που αποθηκεύονται στην βάση δεδομένων είναι ασφαλή. Σε πρώτο επίπεδο τα δεδομένα των χρηστών αποθηκεύονται εξ αρχής

κρυπτογραφημένα στην πραγματικού χρόνου βάση δεδομένων του Firebase, με την χρήση του κρυπταλγορίθμου RSA. Πέραν όμως αυτού, τα δεδομένα κάθε αντικειμένου Firebase είναι κρυπτογραφημένα σύμφωνα με το 256-bit Advanced Encryption Standard, και κάθε κλειδί κρυπτογράφησης κρυπτογραφείται με ένα τακτικά περιστρεφόμενο σύνολο βασικών κλειδιών.

- Η εφαρμογή έχει αρκετές προοπτικές εξέλιξης.

Πέραν όμως των πλεονεκτημάτων που έχει η HospitApp, υπάρχουν και κάποια μειονεκτήματα τα οποία εντοπίστηκαν κατά το πέρας της υλοποίησης αλλά και μετά από πολλές δοκιμές της λειτουργίας της:

- Η απουσία διαδικτύου δεν επιτρέπει την ορθή χρήση της εφαρμογής.
- Το γραφικών περιβάλλον της εφαρμογής είναι αρκετά απλό και ίσως να μην «κερδίζει» εύκολα την προσοχή του χρήστη.
- Προς το παρόν δεν υπάρχει η δυνατότητα επιλογής κάποιας άλλης γλώσσας.

5 Συμπεράσματα

5.1 Συμπεράσματα

Στην παρούσα διπλωματική εργασία, αναπτύχθηκε μία android εφαρμογή, έτοιμη προς χρήση σε οποιαδήποτε κλινική ή ιατρείο, κανονισμού και διαχείρισης ραντεβού μεταξύ γιατρών και ασθενών. Στόχος της δημιουργίας της εφαρμογής ήταν η απλούστευση της διαδικασίας κανονισμού ενός ραντεβού ασθενή με την εφαρμογή και η απαλλαγή του από την υποχρέωση της φυσικής του παρουσίας για τον σκοπό αυτό. Ακόμη, η εφαρμογή δίνει την δυνατότητα δημιουργίας και διατήρησης ενός ψηφιακού ιατρικού φακέλου για κάθε ασθενή, στον οποίο μπορούν να υπάρχουν όλες οι διαγνώσεις συνταγογραφήσεις ή άλλα χρήσιμα για τον ασθενή αρχεία, μαζεμένα σε μία βάση δεδομένων πραγματικού χρόνου, ο οποίος απαλλάσσει γιατρούς και ασθενείς από την χρήση χαρτιού.

Κατά την εκπόνηση της εργασίας, διαπιστώθηκε πως η δημιουργία ενός τέτοιου είδους λογισμικού, είναι αρκετά απαιτητική. Προϋποθέτει σε βάθος κατανόηση των στόχων που είναι επιθυμητό να επιτευχθούν στο πέρας της υλοποίησής του αλλά και εκτεταμένη έρευνα του τρόπου λειτουργίας των ζωτικής σημασίας εργαλείων που οδηγούν στην επιτυχή υλοποίησή του. Για την δημιουργία της εφαρμογής χρησιμοποιήθηκαν εξελιγμένες τεχνολογίες και σύγχρονα εργαλεία, τα οποία μελετήθηκαν σε βάθος, δραστηριότητα η οποία με οδήγησε στην εξοικείωσή μου με αυτά και διεύρυνε το ενδιαφέρον μου για περαιτέρω ενασχόληση με την ανάπτυξη εφαρμογών και λογισμικών.

Εν κατακλείδι, οι αρχικοί στόχοι της διπλωματικής εργασίας επιτεύχθηκαν, καθώς υλοποιήθηκε μια πλήρως λειτουργική android εφαρμογή, η οποία δεν στερείται προοπτικών εξέλιξης και αναβάθμισής της. Οι συνεχώς εξελισσόμενες νέες τεχνολογίες που διαρκώς ανακαλύπτονται και αξιοποιούνται κατάλληλα και με σύνεση, αποτελούν ένα από τα σημαντικότερα εργαλεία που κατέχει η ανθρωπότητα στα χέρια της. Εργαλείο του οποίου η χρήση μπορεί να επεκταθεί σε σχεδόν όλους τους τομείς της καθημερινότητάς μας, να διευκολύνει και

να αναβαθμίσει τον τρόπο ζωής μας και να δώσει λύσεις σε προβλήματα που παλαιότερα έμοιαζαν άλυτα.

5.2 Προοπτικές εξέλιξης

Για την εφαρμογή HospitApp, υπάρχουν αρκετές ιδέες που θα μπορούσαν να την αναβαθμίσουν και να την κάνουν περισσότερο λειτουργική.

Μια αρκετά σημαντική αναβάθμιση της εφαρμογής θα ήταν η αλλαγή του τρόπου σύνδεσης σε αυτήν. Αντί για την χρήση του ΑΜΚΑ και ενός κωδικού πρόσβασης θα ήταν αρκετά πιο χρήσιμη η σύνδεση μέσω του gsis.gr, όπου εκεί υπάρχουν ήδη επαληθευμένα τα στοιχεία όλων των πολιτών και πιθανών χρηστών της εφαρμογής. Εναλλακτικά θα μπορούσε η σύνδεση να γίνεται και μέσω άλλης πλατφόρμας όπως το Gmail.

Άλλη μια χρήσιμη προσθήκη θα ήταν μια λειτουργία με την οποία η εφαρμογή θα έκανε αναγνώριση κειμένου στις διαγνώσεις των ασθενών, αναζητώντας λέξεις - κλειδιά για διάφορες παθήσεις έτσι ώστε να μπορεί να τις αρχειοθετεί με βάση αυτές, κάνοντας την προσπέλαση των pdf αρχείων πιο εύκολη και αποτελεσματική.

Επίσης αρκετά χρήσιμη θα ήταν και η προσθήκη υπενθυμίσεων για τα ραντεβού. Προσθήκη που είναι εύκολα υλοποιήσιμη και θα πρόσθετε σε μια πιο ολοκληρωμένη λειτουργία της εφαρμογής.

Μια άλλη ιδέα για την αναβάθμιση της λειτουργίας της εφαρμογής, είναι η επέκταση του κώδικα, με σκοπό την διατήρηση ξεχωριστού αρχείου, αποκλειστικά για τις συνταγές και τα φάρμακα που λαμβάνει ο ασθενής – χρήστης. Παρόλο που η εφαρμογή ως έχει, παρέχει την δυνατότητα αποθήκευσης των συνταγών, ο διαχωρισμός τους από τις διαγνώσεις θα βοηθήσει στην καλύτερη οργάνωση της βάσης δεδομένων και κατ’ επέκταση στην ακόμη ευκολότερη χρήση της εφαρμογής.

Χρήσιμη επίσης, θα ήταν και η μετατροπή της εφαρμογής έτσι ώστε να μπορεί να χρησιμοποιηθεί και σε λειτουργικό σύστημα IOS. Μια τέτοια επέκταση θα είναι αρκετά σημαντική, αφού θα καλύψει ακόμη μεγαλύτερο εύρος χρηστών.

Βιβλιογραφία

[1] Odeh, Ayman, Raghad Abdelhadi, and Hussien Odeh. "Medical patient appointments management using smart software system in UAE." *2019 International Arab Conference on Information Technology (ACIT)*. IEEE, 2019.

[2] M. Khalid, S. Singh, K. Singh, J. Jeevitha and P. Anand, "Medicus: A Doctor Appointment Booking System", *International Journal of Computing and Technology*, vol. 5, no. 4, pp. 48-52, n.d.

[3] S. Malik, N. Bibi, S. Khan, R. Sultana and S. Abdul Rauf, "Mr. Doc: A Doctor Appointment Application System", *International Journal of Computer Science and Information Security*, vol. 14, no. 12, pp. 452-460, 2016.

[4] M. A. R. Y. A. M. TUFAIL, "Tampere University of Applied Sciences", Master's Degree in Information Technology, 2018.

[5] Joy, B., Steele, G., Gosling, J., & Bracha, G. (2000). The Java language specification.

[6] Yellin, Frank, and Tim Lindholm. "The java virtual machine specification." (1996).

[7] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F. (1997). Extensible markup language (XML). *World Wide Web Journal*, *2*(4), 27-66.

[8] https://developer.android.com/studio/intro

[9] Chatterjee, N., Chakraborty, S., Decosta, A., & Nath, A. (2018). Real-time communication application based on android using Google firebase. *Int. J. Adv. Res. Comput. Sci. Manag. Stud*, *6*(4).

[10] firebase.google.com,Firebase Realtime Database, https://firebase.google.com/docs/database/?gclid=EAIaIQobChMI8L3v54SY6AIVxbTtCh2_mAVh EAAYASAAEgLhyPD_BwE

[11] Realtime Database: Chatterjee, N., Chakraborty, S., Decosta, A., & Nath, A. (2018). Real-time communication application based on android using Google firebase. *Int. J. Adv. Res. Comput. Sci. Manag. Stud*, *6*(4).

[12] ΧΡΥΣΟΛΩΡΆΣ, Γεώργιος-Θεόδωρος; ΣΤΡΙΓΚΟΣ, Νομικός-Αντώνιος. Εφαρμογή των Γενετικών Αλγορίθμων στο πρόβλημα RSA (RSAP). 2005.

[13] Milanov, Evgeny. "The RSA algorithm." *RSA laboratories* (2009): 1-11.

[14] Βασίλειος Α. Κάτος, Γεώργιος Χρ. Στεφανίδης, επιμ. (2003). *Τεχνικές Κρυπτογραφίας & Κρυπτανάλυσης*. Θεσσαλονίκη: Ζυγός. ISBN 960-8065-40-2. (κεφ. 6.5.2.: Ασφάλεια του RSA)

Παράρτημα Ι [ΤΙΤΛΟΣ ΠΑΡΑΡΤΗΜΑΤΟΣ Ι]

Στα παραρτήματα συνήθως τοποθετούνται:

1 Παράλληλα αποτελέσματα της εργασίας (όπως π.χ. αποδείξεις θεωρημάτων) τα οποία δεν είναι απαραίτητα για την κατανόηση του κειμένου που προηγήθηκε.

2 Επιλεγμένα κομμάτια κώδικα που δημιουργήθηκαν κατά τη διάρκεια της εργασίας, παρουσιάζουν ιδιαίτερο ενδιαφέρον και αξίζουν την προσοχή του ενδιαφερόμενου αναγνώστη.