Πανεπιστήμιο Θεσσαλίας
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Ευφυής διαχείριση ιατρικών ραντεβού μέσω εφαρμογής κινητής συσκευής (Doctorganize)

Διπλωματική Εργασία

ΘΩΜΑ ΑΘΑΝΑΣΙΟΥ

Επιβλέπων
Μιχαήλ Βασιλακόπουλος
Αναπληρωτής Καθηγητής

Βόλος, Ιούνιος 2019

Πανεπιστήμιο Θεσσαλίας
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Ευφυής διαχείριση ιατρικών ραντεβού μέσω εφαρμογής κινητής συσκευής (Doctorganize)

Διπλωματική Εργασία

ΘΩΜΑ ΑΘΑΝΑΣΙΟΥ

Επιτροπή επίβλεψης

Επιβλέπων	Συνεπιβλέπουσα	Συνεπιβλέπουσα
Μιχαήλ Βασιλακόπουλος	Ελένη Τουσίδου	Ασπασία Δασκαλοπούλου
Αναπληρωτής Καθηγητής	Μέλος Ε.ΔΙ.Π.	Επίκουρη Καθηγήτρια

Βόλος, Ιούνιος 2019

Πανεπιστήμιο Θεσσαλίας
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Ο/Η συγγραφέας αυτής της εργασίας βεβαιώνει ότι κάθε βοήθεια την οποία είχε για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης βεβαιώνει ότι έχει αναφέρει τις όποιες πηγές από τις οποίες έκανε χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται επακριβώς, είτε παραφρασμένες.

University of Thessaly
Faculty of Engineering
Department of Electrical & Computer Engineering

A mobile app for the intelligent management of medical appointments (Doctorganize)

Diploma Thesis

THOMAS ATHANASIOU

Supervisor
Michael Vassilakopoulos
Associate professor

Volos, June 2019

Περίληψη

Οι εφαρμογές Android και γενικότερα οι web applications παίζουν σημαντικό ρόλο στην τεχνολογία και στην ανθρώπινη επικοινωνία. Σε αυτήν τη διπλωματική εργασία θα ασχοληθούμε με τον Ιατρικό-Φαρμακευτικό Τομέα. Συγκεκριμένα είναι μία εφαρμογή Android που έχει ως στόχο την καλύτερη και πιο έξυπνη επικοινωνία Γιατρού και Ασθενή. Οι ασθενείς σπαταλούν πολύ χρόνο για να βρουν γιατρούς που χρειάζονται και ακόμα περισσότερο να κλείσουν κάποιο ραντεβού μαζί τους. Από την πλευρά των γιατρών υπάρχει επίσης αρκετός φόρτος στο να οργανώσουν συναντήσεις με ασθενείς. Η συγκεκριμένη εφαρμογή (Doctorganize) έχει να κάνει με την επίλυση αυτού του προβλήματος. Το layout της εφαρμογής έχει σχεδιαστεί έτσι ώστε να είναι πολύ φιλικό σε κάθε χρήστη, ανεξαρτήτως ηλικίας, με μια μικρή προυπόθεση να ξέρουν να χειρίζονται βασικά ένα smartphone. Μέσω της εφαρμογής οι ασθενείς μπορούν να ψάξουν και να βρουν τους γιατρούς τους οποίους χρειάζονται και να ζητήσουν κάποια συνάντηση. Επίσης μπορούν να βλέπουν ημερομηνίες και ώρες των συναντήσεων με τους γιατρούς ανά πάσα στιγμή. Από την πλευρά των ιατρών μπορούν να κάνουν διαθέσιμες κάποιες ώρες συναντήσεων ώστε ο κάθε ασθενής να μπορεί να επιλέξει εύκολα και γρήγορα τη συνάντησή του. Αυτό λύνει προβλήματα όπως τηλεφωνική επικοινωνία, καθυστερήσεις σε ιατρεία.

Λέξεις Κλειδιά

Λέξη Android, Mobile Application, Ιατροφαρμακευτική Εφαρμογή

Abstract

Mobile and web applications in general play an important role in technology and human communication. In this diploma thesis we will deal with the Medical-Pharmaceutical Sector. Specifically, it is an Android application that aims to better and smarter Doctor and Patient communication. Patients spend a lot of time finding doctors that they need, and even more to make an appointment with them. From the doctors' side, there is also a lot of workload to organize meetings with patients. This particular application (Doctorganize) has to do with solving this problem. The layout of the application is designed to be very friendly to any user regardless of age with a small requirement to know how to basically handle a smartphone. Through the application, patients can look and find the doctors they need and ask for a meeting. They can also see dates and times of meetings with doctors at any time. From the physicians' side, they can make available a few hours of meetings so that each patient can easily and quickly choose his appointment. This solves problems such as telephone communication, delays in medical practices.

Keywords

Android, Mobile Application, Medical and Doctor organization

Στην οικογένειά μου.

Ευχαριστίες

Καταρχήν θα ήθελα να ευχαριστήσω από τα βάθη της καρδιάς μου την οικογένειά μου που με στήριξε ψυχολογικά και πρακτικά για να φτάσω στο σημείο να τελειώσω τις σπουδές μου και να έχω την δυνατότητα να μπορώ να ασχοληθώ με τον τομέα που αγαπώ. Στην συνέχεια θα ήθελα να ευχαριστήσω θερμά τον κύριο Μιχαήλ Βασιλακόπουλο που μου έδωσε την ευκαιρία να υλοποιήσω μία τέτοια εφαρμογή. Η καθοδήγηδή του καθώς επίσης και οι ιδέες που μου έδωσε για την καλύτερη κατανόηση και υλοποίηση της εφαρμογής ήταν κάτι παραπάνω απο βοηθητικές. Επίσης, θα ήθελα να ευχαριστήσω τον κύριο Εμμανουήλ Βάβαλη ο οπόιος έπαιξε καθοριστικό ρόλο στον τομέα του Μηχανικού Λογισμικού που ακολούθησα. Οι οδηγίες, οι συμβουλές και η καθοδήγηση που μου παρείχε με έκαναν πέρα από το να αποκτήσω πολυ σημαντικές γνώσεις σε αυτόν τον τομέα, να έχω και το έναυσμα να τον ακολουθήσω και να έχω την διάθεση να εξελιχθώ περισσότερο. Ύστερα θα ήθελα να ευχαριστήσω τους φίλους μου Νικόλαο Ροζή, Κώστα Χουλιαρά, Χαρίση Πέτρο συγκεκριμένα καθώς ασχοληθήκαμε μαζί με πολλά projects και η συνεργασία μαζί τους έπαιξε τεράστιο ρόλο στην εως τώρα ακαδημαική μου καριέρα. Τέλος, ένα μεγάλο και ιδιαίτερο ευχαριστώ στον Νίλο Ψαθά για την αμέριστη βοήθειά του σε μεγάλος μέρος δυσκολιών που αντιμετώπισα.

Πρόλογος

Η παρούσα διπλωματική εργασία εκπονήθηκε ως τελευταίο βήμα της ακαδημαικής μου διαδρομής για την απόκτηση του διπλώματος του τμήματος Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών του Πανεπιστημίου Θεσσαλίας στην πόλη του Βόλου υπό την επίβλεψη του καθηγητή Μιχαήλ Βασιλακόπουλου. Σαν ιδέα ήταν η άμεση και γρήγορη εξυπηρέτηση ασθενών αλλά και ιατρών σε ότι έχει να κάνει με την διαχείρηση των ραντεβού και των συναντήσεών τους. Το ερέθισμα για να ασχοληθώ με μια τέτοια εφαρμογή ήρθε κατά την ενασχόλησή μου σε κάποια projects μιας φαρμακευτικής εταιρίας.

Περιεχόμενα

Κατάλογος σχημάτων

Κεφάλαιο 1

Εισαγωγή

1.1 Αρχική Ιδέα

Ο προγραμματισμός ήταν από την αρχή της ακαδημαικής μου καριέρας ένας τομέας τον οποίο αγάπησα και ασχολήθηκα πολύ περισσότερο από άλλους. Αρχικά από γλώσσες προγραμματισμού C [33], Java [32] άρχισα να παίρνω βασικές γνώσεις για το τι ακριβώς είναι προγραμματισμός, πως μια σειρά εντολών μπορούν να οδηγήσουν τον υπολογιστή να κάνει κάποιες απλές λειτουργίες, πως ένας υπολογιστής μπορεί να κάνει πολλαπλούς υπολογισμούς και λειτουργείες.

Υλοιποιώντας projects για σκοπούς της σχολής έμαθα πιο σύνθετους από τους βασικούς όρους του προγραμματισμού, όπως αντικειμενοστρέφεια, δυναμική διαχείριση μνήμης, σχεδιασμός design απλών εφαρμογών. Στη συνέχεια με μαθήματα όπως WWW technologies, Πληροφοριακά Συστήματα στον Παγκόσμιο Ιστό καθώς επίσης και ειδικά θέματα τα οποία εμπεριείχαν projects σε τεχνολογίες Django και Android ανέπτυξα τις γνώσεις μου πάνω στο Web and Mobile application Development κάτι που αισθάνθηκα ότι ήταν αυτό που θα ήθελα να κάνω επαγγελματικά για το υπόλοιπο της ζωής μου.

Το Android Development [34] ήταν κάτι που μου κίνησε πραγματικά το ενδιαφέρον και αποφάσισα να υλοποιήσω μια εφαρμογή για την διπλωματική μου εργασία. Κατά τη διάρκεια της υλοποίησης της, εξοικειώθηκά αρκετά με το Android Studio και τους emulators που χρησιμοποιούνται εκεί. Προφανώς υπήρξαν πολλά εμπόδια και δυσκολίες. Παρόλα αυτά με συνεχή αναζήτηση στο διαδίκτυο, ακολουθώντας πολλά tutorials σε YouTube και επικοινωνώντας με άτομα που είχαν παρόμοιες τριβές, όλα λύνονταν σταδιακά και κατάφερα να ολοκληρώσω επιτυχώς την εφαρμογή.

Εξ αιτίας μιας πολύμηνης εργασίας μου σε μια φαρμακευτική εταιρία αποφάσισα σαν στόχο να υλοποιήσω μια τέτοια εφαρμογή με σκοπό την βοήθεια ατόμων στον Ιατροφαρμακευτικό τομέα, δίνοντας την δυνατότητα στους χρήστες (συμπεριλαμβανομένου του εαυτού μου) να μπορούν γρήγορα και εύκολα μέσω μιας απλής εφαρμογής να κανονίζουν ραντεβού που υπό άλλες περιπτώσεις θα ήταν πολυ πιθανό να συναντουσαν προβλήματα.

Στις παρακάτω σελιδες παρουσιάζεται η πλατφόρμα Android και τα στάδια τα οποία ακολούθησα στην ανάπτυξη της εφαρμογής.

1.2 Γενικά για το Android

Σχήμα 1.1: Android Logo from [6]

Το Android [3] είναι λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας το οποίο τρέχει τον πυρήνα του λειτουργικού Linux. Αρχικά αναπτύχθηκε από την Google και αργότερα από την Open Handset Alliance. Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google.Το Android είναι κατά κύριο λόγο σχεδιασμένο για συσκευές με οθόνη αφής, όπως τα έξυπνα τηλέφωνα και τα τάμπλετ, με διαφορετικό περιβάλλον χρήσης για τηλεοράσεις (Android TV), αυτοκίνητα (Android Auto) και ρολόγια χειρός (Android Wear). Παρόλο που έχει αναπτυχθεί για συσκευές με οθόνη αφής, έχει χρησιμοποιηθεί σε κονσόλες παιχνιδιών, ψηφιακές φωτογραφικές μηχανές, συνηθισμένους Η/Υ (π.χ. το HP Slate 21) και σε άλλες ηλεκτρονικές συσκευές.

Το Android είναι το πιο ευρέως διαδεδομένο λογισμικό στον κόσμο. Οι συσκευές με Android έχουν περισσότερες πωλήσεις από όλες τις συσκευές Windows, iOS και Mac OS X μαζί.

Η πρώτη παρουσίαση της πλατφόρμας Android έγινε στις 5 Νοεμβρίου 2007, παράλληλα με την ανακοίνωση της ίδρυσης του οργανισμού Open Handset Alliance, μιας κοινοπραξίας 48 τηλεπικοινωνιακών εταιριών, εταιριών λογισμικού καθώς και κατασκευής hardware, οι οποίες είναι αφιερωμένες στην ανάπτυξη και εξέλιξη ανοιχτών προτύπων στις συσκευές κινητής τηλεφωνίας. Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα του Android υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού. Το λογότυπο για το λειτουργικό σύστημα Android είναι ένα ρομπότ σε χρώμα πράσινου μήλου και σχεδιάστηκε από τη γραφίστρια Ιρίνα Μπλόκ.

Η τελευταία έκδοση καλείται Android 9.0 Pie και φέρνει σημαντικές αλλαγές, όπως το Adaptive Battery και το Adaptive Brightness. Επίσης, στο Android 9.0 Pie υπάρχει ένα νέο σύστημα πλοήγησης με ένα μεμονωμένο κουμπί home.

1.3 Ιστορία του Android

Το λειτουργικό σύστημα Android [5] ιδρύθηκε στο Palo Alto της Καλιφόρνια τον Οκτώβριο του 2003 από τους Andy Rubin, Rich Miner, Nick Sears και Chris White. Ο Rubin χαρακτήρισε το έργο Android ως "τεράστιο δυναμικό για την ανάπτυξη έξυπνων κινητών συσκευών που έχουν μεγαλύτερη επίγνωση της θέσης και των προτιμήσεων του ιδιοκτήτη του". Οι πρώτες προθέσεις της εταιρείας ήταν να αναπτύξει ένα προηγμένο λειτουργικό σύστημα για ψηφιακές φωτογραφικές μηχανές και αυτή ήταν η βάση του πεδίου για τους επενδυτές τον Απρίλιο του 2004. Η εταιρεία αποφάσισε τότε ότι η αγορά των φωτογραφικών μηχανών δεν ήταν αρκετά μεγάλη για τους στόχους της, πέντε μήνες αργότερα είχε εκτρέψει τις προσπάθειές της και ανέβαζε το Android ως λειτουργικό σύστημα χειρός που θα ανταγωνιζόταν τα Symbian και τα Microsoft Windows Mobile.

Το λογότυπο του Android από το 2007 έως το 2014. Ο Ρούμπιν είχε δυσκολία να προσελκύσει επενδυτές από νωρίς και το Android αντιμετώπιζε έξωση από το χώρο γραφείων του. Ο Steve Perlman, ένας στενός φίλος του Ρούμπιν, του έφερε 10.000 δολάρια σε μετρητά σε ένα φάκελο, και σύντομα έπειτα ενσύρθηκε ένα μη ανακοινωθέν ποσό ως χρηματοδότηση σπόρων. Ο Perlman αρνήθηκε ένα μερίδιο στην εταιρεία και δήλωσε ότι "το έκανα επειδή πίστευα στο πράγμα και ήθελα να βοηθήσω τον Andy". Τον Ιούλιο του 2005, η Google απέκτησε το Android Inc. για τουλάχιστον 50 εκατομμύρια δολάρια. Οι βασικοί υπάλληλοί της, όπως οι Rubin, Miner και White, εντάχθηκαν στο Google ως μέρος της εξαγοράς. Δεν γνώριζαν πολλά για το μυστικό Android την εποχή εκείνη, καθώς η εταιρεία παρέσχε λίγες λεπτομέρειες εκτός από το ότι έκανε λογισμικό για κινητά τηλέφωνα. Στο Google, η ομάδα με επικεφαλής τον Rubin ανέπτυξε μια πλατφόρμα κινητών συσκευών που τροφοδοτείται από τον πυρήνα του Linux. Η Google προώθησε την πλατφόρμα στους κατασκευαστές και τους μεταφορείς με την υπόσχεση παροχής ενός ευέλικτου, αναβαθμίσιμου συστήματος. Η Google είχε "παρατάξει μια σειρά από εξαρτήματα υλικού και συνεργάτες λογισμικού και σημείωσε στους μεταφορείς ότι ήταν ανοικτή σε διάφορους βαθμούς συνεργασίας".

Η Google άλλαξε αργότερα τα έγγραφα προδιαγραφών Android για να δηλώσει ότι θα υποστηρίζονται οι "Οθόνες αφής", αν και "το Προϊόν σχεδιάστηκε με την παρουσία διακριτών φυσικών κουμπιών ως υπόθεση, επομένως μια οθόνη αφής δεν μπορεί να αντικαταστήσει πλήρως τα φυσικά κουμπιά". Μέχρι το 2008, τόσο η Nokia όσο και το BlackBerry ανακοίνωσαν έξυπνα τηλέφωνα με βάση το touch για να ανταγωνιστεί το iPhone 3G και η εστίαση του Android τελικά μετατράπηκε σε απλές οθόνες αφής. Το πρώτο εμπορικά διαθέσιμο smartphone που τρέχει το Android ήταν το HTC Dream, επίσης γνωστό ως T-Mobile G1, που ανακοινώθηκε στις 23 Σεπτεμβρίου 2008.

Στις 5 Νοεμβρίου 2007, η Open Handset Alliance, μια κοινοπραξία εταιρειών τεχνολογίας όπως η Google, κατασκευαστές συσκευών όπως HTC, Motorola και Samsung, ασύρματες εταιρείες όπως η Sprint και η T-Mobile και κατασκευαστές chipset όπως η Qualcomm και η Texas Instruments με στόχο την ανάπτυξη της "πρώτης πραγματικά ανοικτής και ολοκληρωμένης πλατφόρμας για κινητές συσκευές". Μέσα σε ένα χρόνο η Open Handset Alliance αντιμετώπισε δύο άλλους ανταγωνιστές ανοιχτής πηγής, το Symbian Foundation και το LiMo Foundation, το τελευταίο αναπτύσσοντας επίσης ένα κινητό λειτουργικό σύστημα όπως το Google. Τον Σεπτέμβριο του 2007, το InformationWeek κάλυψε μια μελέτη Evalueserve που ανέφερε ότι η Google είχε

καταθέσει αρκετές αιτήσεις ευρεσιτεχνίας στον τομέα της κινητής τηλεφωνίας.

Από το 2008, το Android έχει δει πολλές αναβαθμίσεις, οι οποίες έχουν βελτιώσει σταδιακά το λειτουργικό σύστημα, προσθέτοντας νέα χαρακτηριστικά και διορθώνοντας σφάλματα σε προηγούμενες κυκλοφορίες. Κάθε μεγάλη κυκλοφορία ονομάζεται με αλφαβητική σειρά μετά από επιδόρπιο ή ζαχαρούχο σκεύασμα, με τις πρώτες εκδόσεις του Android να ονομάζονται "Cupcake", "Donut", "Eclair" και "Froyo", με αυτή τη σειρά. Κατά την ανακοίνωση του Android KitKat το 2013, η Google εξήγησε ότι "Δεδομένου ότι αυτές οι συσκευές κάνουν τη ζωή μας τόσο γλυκιά, κάθε έκδοση Android ονομάζεται μετά από ένα επιδόρπιο", παρόλο που ένας εκπρόσωπος της Google δήλωσε στο CNN σε συνέντευξή του ότι "Είναι σαν μια εσωτερική ομάδα κάτι που προτιμάμε να είμαστε λίγο - πώς θα πρέπει να πω - λίγο ασυμβίβαστο στο θέμα, θα πω ".

Το 2010, η Google παρουσίασε τη σειρά συσκευών Nexus, μια σειρά στα οποία η Google συνεργάστηκε με διαφορετικούς κατασκευαστές συσκευών για την παραγωγή νέων συσκευών και την εισαγωγή νέων εκδόσεων Android. Η σειρά περιγράφηκε ότι έχει "διαδραματίσει βασικό ρόλο στο ιστορικό του Android εισάγοντας νέες επαναλήψεις λογισμικού και πρότυπα υλικού σε όλο το πλάτος" και έγινε γνωστός για το λογισμικό "free-bloat" με "έγκαιρες ενημερώσεις". Στη διάσκεψη προγραμματιστών τον Μάιο του 2013, η Google ανακοίνωσε μια ειδική έκδοση του Samsung Galaxy S4, όπου, αντί να χρησιμοποιήσει την προσαρμογή του Android, το τηλέφωνο έτρεξε "stock Android" και υποσχέθηκε να λάβει γρήγορα νέες ενημερώσεις συστήματος. Η συσκευή θα γίνει η αρχή του προγράμματος Google Play και θα ακολουθήσει άλλες συσκευές, όπως η έκδοση HTC One Google Play και η έκδοση MotoG Google Play. Το 2015, ο Ars Technica έγραψε ότι "Νωρίτερα αυτή την εβδομάδα, το τελευταίο από τα τηλέφωνα Android για την έκδοση Google Play στην ηλεκτρονική προθήκη της Google αναφέρθηκε ως" δεν είναι πλέον διαθέσιμο προς πώληση "και ότι" Τώρα όλα έχουν φύγει και φαίνεται όπως και το πρόγραμμα έχει ολοκληρωθεί.

Από το 2008 έως το 2013, ο Hugo Barra υπηρέτησε ως εκπρόσωπος των προϊόντων, εκπροσωπώντας το Android σε συνεντεύξεις Τύπου και το Google I / O, το ετήσιο συνέδριο της Google για τους προγραμματιστές. Έφυγε από την Google τον Αύγουστο του 2013 για να ενώσει την κινεζική εταιρεία κατασκευής τηλεφώνων Xiaomi. Λιγότερο από έξι μήνες νωρίτερα, ο τότε διευθύνων σύμβουλος της Google, Larry Page ανακοίνωσε σε blog, ότι ο Andy Rubin είχε μετακομίσει από το τμήμα Android για να αναλάβει νέα έργα στο Google και ότι η Sundar Pichai θα γίνει το νέο Android. Ο ίδιος ο Pichai θα αλλάξει τις θέσεις του και θα γίνει ο νέος CEO της Google τον Αύγουστο του 2015 μετά την αναδιάρθρωση της εταιρίας στο αλφαβητάρι αλφαβήτου, καθιστώντας τον Hiroshi Lockheimer νέο επικεφαλής του Android.

Τον Ιούνιο του 2014, η Google ανακοίνωσε το Android One, ένα σύνολο «μοντέλων αναφοράς υλικού» που θα «επιτρέψουν στους κατασκευαστές συσκευών να δημιουργήσουν εύκολα τηλέφωνα υψηλής ποιότητας με χαμηλό κόστος», σχεδιασμένα για καταναλωτές στις αναπτυσσόμενες χώρες. Τον Σεπτέμβριο, η Google ανακοίνωσε το πρώτο σετ Android One phones για κυκλοφορία στην Ινδία. Ωστόσο, η Recode ανέφερε τον Ιούνιο του 2015 ότι το έργο ήταν "απογοήτευση", αναφέροντας "διστακτικούς καταναλωτές και συνεργάτες κατασκευής" και "σφάλματα από την εταιρεία αναζήτησης που ποτέ δεν έχει σπάσει το υλικό". Τα σχέδια για την επανενεργοποίηση του Android One εμφανίστηκαν τον Αύγουστο του 2015, με την Αφρική να ανακοινώθηκε ως επόμενη θέση για το πρόγραμμα μια εβδομάδα αργότερα. Μια αναφορά από τις πληροφορίες

τον Ιανουάριο του 2017 δήλωσε ότι η Google επεκτείνει το πρόγραμμα Android One με χαμηλό κόστος στις Ηνωμένες Πολιτείες, αν και η The Verge σημειώνει ότι η εταιρεία πιθανότατα δεν θα παράγει τις ίδιες τις ίδιες τις συσκευές.

Η Google εισήγαγε τα smartphone Pixel και Pixel XL τον Οκτώβριο του 2016, τα οποία διατέθηκαν στο εμπόριο ως τα πρώτα τηλέφωνα που έκανε η Google και παρουσίαζαν αποκλειστικά ορισμένες λειτουργίες λογισμικού, όπως ο Βοηθός Google (Google Assistant), πριν από την ευρύτερη ανάπτυξη. Τα τηλέφωνα Pixel αντικατέστησαν τη σειρά Nexus, με μια νέα γενιά τηλεφώνων Pixel που ξεκίνησε τον Οκτώβριο του 2017.

1.4 Ιστορικό Εκδόσεων Android

Παρόλο που το Android [5] είναι ένα προϊόν ελεύθερου λογισμικού, ένα κομμάτι της ανάπτυξης του λογισμικού συνεχίζεται σε ιδιωτικό παρακλάδι. Για να έρθει αυτό το λογισμικό σε κοινή θέαση δημιουργήθηκε ένα παρακλάδι του μόνο ανάγνωσης, εν ονόματι Cupcake. Το Cupcake συνήθως συγχέεται με τον τίτλο μιας ενημέρωσης, σε αντίθεση με όσα δηλώνει η ίδια η Google στην ιστοσελίδα ανάπτυξης του Android: το Cupcake αποτελεί ακόμη ένα έργο σε εξέλιξη, όχι μια επίσημη έκδοση. Αξιοσημείωτες αλλαγές στο λειτουργικό Android θα παρουσιαστούν στο cupcake και περιλαμβάνουν αλλαγές στο σύστημα διαχείρισης των μεταφορτώσεων (download manager), το framework, Bluetooth, το λογισμικό συστήματος, το ραδιόφωνο και το σύστημα τηλεφωνίας, εργαλεία προγραμματισμού , το κυρίως σύστημα και διάφορες εφαρμογές, καθώς και πληθώρες διορθώσεις σφαλμάτων. Στις 7 Αυγούστου 2018, κυκλοφόρησε η επίσημη ενημέρωση έκδοσης 9.0 για το Android. Αποτελείται από πολλά νέα χαρακτηριστικά και βελτιώσεις στο γραφικό περιβάλλον:

- Τη λειτουργία Adaptive Battery, η οποία δίνει προτεραιότητα στην κατανάλωση της μπαταρίας μόνο για τις εφαρμογές και τις υπηρεσίες που χρησιμοποιεί περισσότερο ο χρήστης, κλείνοντας ταυτόχρονα όλες τις υπόλοιπες εφαρμογές που τρέχουν στο background.
- Τη λειτουργία Adaptive Brightness, η οποία μαθαίνει πως να προσαρμόζει την φωτεινότητα της οθόνης ανάλογα με το περιβάλλον.
- Τα App Actions θα σε βοηθούν να πραγματοποιείς την επόμενη ενέργεια σου πιο γρήγορα επειδή θα αντιλαμβάνονται τι ακριβώς θέλεις να κάνεις.
- Η εμφάνιση και η εμπειρία χρήσης γίνεται πιο εύκολη με το νέο σύστημα πλοήγησης.
- Το Android Dashboard, το οποίο θα δείχνει στον χρήστη αναλυτικά πόσο χρόνο ξοδεύει σε συγκεκριμένες εφαρμογές, πόσες φορές, ξεκλείδωσε το smartphone και πόσες ειδοποιήσεις έλαβε κατά τη διάρκεια της ημέρας.
- Το App Timer είναι ένα εργαλείο που θα του επιτρέπει να θέτει χρονικό περιορισμό στην χρήση μιας εφαρμογής, θα τον ειδοποιεί όταν φτάνει στο όριο που έχει θέσει και θα μετατρέπει το εικονίδιο της εφαρμογής σε ασπρόμαυρο για να του υπενθυμίσει

Code name	Version number	Linux kernel version[1]	Initial release date	API level	Ref.
No Codename	1.0	2.1	September 23, 2008	1	[2]
Petit Four	1.1	2.6	February 9, 2009	2	[2]
Cupcake	1.5	2.6.27	April 27, 2009	3	
Donut	1.6	2.6.29	September 15, 2009	4	[3]
Eclair	2.0 – 2.1	2.6.29	October 26, 2009	5 – 7	[4]
Froyo	2.2 – 2.2.3	2.6.32	May 20, 2010	8	[5]
Gingerbread	2.3 – 2.3.7	2.6.35	December 6, 2010	9 – 10	[6]
Honeycomb	3.0 – 3.2.6	2.6.36	February 22, 2011	11 – 13	[7]
Ice Cream Sandwich	4.0 – 4.0.4	3.0.1	October 18, 2011	14 – 15	[8]
Jelly Bean	4.1 – 4.3.1	3.0.31 to 3.4.39	July 9, 2012	16 – 18	[9]
KitKat	4.4 – 4.4.4	3.10	October 31, 2013	19 – 20	[10]
Lollipop	5.0 – 5.1.1	3.16	November 12, 2014	21 – 22	[11]
Marshmallow	6.0 – 6.0.1	3.18	October 5, 2015	23	[12]
Nougat	7.0 – 7.1.2	4.4	August 22, 2016	24 – 25	[13]
Oreo	8.0 – 8.1	4.10	August 21, 2017	26 – 27	[14]
Pie	**9.0**	4.4.107, 4.9.84, and 4.14.42	August 6, 2018	28	[15]
Android Q	10.0			29	
Legend: Old version	Older version, still supported	**Latest version**	Latest preview version		

Σχήμα 1.2: Android Versions and Releases from [5]

1.5 Βιβλιογραφική Ανασκόπηση

Βασική προϋπόθεση για την υλοποίηση μιας τέτοιου είδους εφαρμογής είναι η γνώση βασικών χαρακτηριστηκών της τεχνολογίας Android όπως η αλληλεπίδραση της γλώσσας προγραμματισμού Java και των αρχείων XML, η ένταξη κάποιας βάσης δεδομένων στην εφαρμογή καθώς επίσης και το πως χειρίζεται αυτή η τεχνολογία http requests [10].
Η χρήση της Java για το Android Developing είναι πολύ σημαντική διότι προσφέρει ασφάλεια (Τίποτα δεν εκτελείται έξω από το JVM), είναι αντικειμενοστραφής, είναι πλούσια σε frameworks και classes όπως networking, threading, IO operations και τέλος είναι Open Source [37].
Για την ανάπτυξη της συγκεκριμένης εφαρμογής χρησιμοπιήθηκαν ιδέες από προηγούμενες υλοποιήσεις και έρευνες. Η αρχική ιδέα για το πως οι χρήστες μπορούν να εγγράφονται και να συνδέονται στην εφαρμογή, είτε ως ιατροί είτε ως ασθενείς, ήρθε από το άρθρο [28] των Noorsyahira Ismail, Shahreen Kasim1, Yusmadi Yah Jusoh, Rohayanti Hassan, Ayu Alyani μια ομάδα Μηχανικών Υπολογιστών στην Μαλαισία οι οποίοι ανέπτυξαν μια παρόμοια εφαρμογή.
Οι Alaa Qaffas και Trevor Barker [35] έδειξαν πως μπορεί να οργανωθεί μια εφαρμογή με σκοπούς booking και πως γίνεται ο χρήστης να επιλέγει ημερομηνίες, ώρες κτλ. Επίσης για τις λειτουργίες τις Firebase μελέτησα την εφαρμογή Bon_Voyage [11], που είχε να κάνει με κρατήσεις για δωμάτια ξενοδοχείων. Ιδιαίτερα σημαντικό ήταν το πως θα στηθεί ένα Web Api για την υλοποίηση των ειδοπιήσεων. Όπως αναφέρει ο Andri Heryandi [25], για να υλοποιηθεί μια τέτοια λειτουργία χρειάζεται ένα API(Application Programming Interface) που θα κάνει την εφαρμογή να επικοινωνεί με τα υπάρχοντα πληροφοριακά συστήματα και να μπορεί να στέλνει ειδοποιήσεις. Το API δημιουργείται με το web protocol(http/https) και συμπεριφέρεται σαν ένα πρόσθετο σύσστημα στην εφαρμογή.

Οι εφαρμογές στον Ιατροφαρμακευτικό Τομέα έχουν πολλά χαρακτηριστικά, όπως διαχείριση συναντήσεων, παρακολούθηση φαρμακευτικής αγωγής, οικονομικές προτάσεις κ.ο.κ. Κάποιες από τις πιο διαδεδομένες εφαρμογές στον τομέα αυτόν είναι οι εξής:

- DoctorAnytime
  Το DoctorAnytime είναι διαθέσιμο και μέσω web [14] και μέσω Android Εφαρμογής [13] Το doctoranytime, το οποίο ξεκίνησε να δημιουργείται τον Μάιο του 2012, είναι μία καινοτόμος υπηρεσία στο χώρο της υγείας για εύκολο και γρήγορο κλείσιμο ραντεβού μέσω διαδικτύου (ή και μέσω τηλεφώνου για όσους το επιθυμούν) σε γιατρούς και διαγνωστικά κέντρα σ' όλη την Ελλάδα. Πάνω από 40 πόλεις εξυπηρετούνται από το doctoranytime με τον κύριο όγκο γιατρών και διαγνωστικών κέντρων να βρίσκεται σε Αθήνα και Θεσσαλονίκη, ενώ υποστηρίζονται όλα τα ασφαλιστικά ταμεία.
  Οι διαφορές με την Doctorganize είναι ότι υπάρχουν Ειδικέυσεις Ιατρών με τις οποίες οι ασθενείς μπορούν να κάνουν αναζήτηση καθώς επίσης και τοποθεσίες που προτιμούν οι ασθενείς.

- NowDoctor
  Το NowDoctor είναι μία web application [29] που εξυπηρετεί ραντεβού μεταξύ ιατρών ασθενών. Οι χρήστες αναζητούν γιατρό μέσα από μια εύκολη αναζήτηση.Τα βήματα είναι απλά.

Η ειδικότητα, η περιοχή, ο ασφαλιστικος φορέας με τον οποίο είστε συμβεβλημένος αποτέλουν τα στοιχεία της αναζήτησης. Στην προσωπική σελίδα των ιατρών, οι χρήστες μπορούν να διαβάσουν το αναλυτικό βιογραφικό τους κι ένα προσωπικό τους μήνυμα. Επίσης, έχουν τη δυνατότητα να δουν φωτογραφίες από το ιατρείο και βίντεο με συνέντευξη τους.
Οι διαφορές με το Doctorganize είναι ότι υπάρχει ένα πιο λεπτομερές profile στην πλευρά του ιατρού από το οποίο ο ασθενής μπορεί να δει σημαντικές λεπτομέρειες όπως ειδίκευση, συμβεβλημένα ταμεία, φωτογραφείες του ιατρείου και το βιογραφικό του Ιατρού.

- dr. Pad
  Χρησιμοποιώντας την εφαρμογή διαχείρισης ραντεβού Dr.Pad [15], μπορούν οι Ιατροί να διαχειριστούν όλα τα αρχεία ασθενών τους όπως προσωπικές πληροφορίες, ιατρικές εκθέσεις, φάρμακα, ιστορικό επισκέψεων, κλινικές σημειώσεις, ιστορικό ασθενών και άλλες σημειώσεις.
  Η κύρια διαφορά με το Doctorganize είναι ότι υπάρχει το ιστορικό των ασθενών στην εφαρμογή, κάτι το οποίο δίνει σημαντικές βοήθειες στους Ιατρούς.

- Book My Doc
  Το Book My Doc [9] είναι ένα πολυλειτουργικό ηλεκτρονικό πρόγραμμα για προγραμματισμό ιατρικών ραντεβού για γιατρούς, κλινικές και μεγάλα νοσοκομεία. Αυτό το σύστημα προγραμματισμού ανοιχτού κώδικα βασίζεται σε php με πλαίσιο CodeIgniter. Με την ευκολία διαμόρφωσης και εξατομίκευσης, το Book My Doc είναι το καλύτερο εργαλείο τόσο για τους γιατρούς όσο και για τους ασθενείς. Πρόκειται για ένα ισχυρό πρόγραμμα ιατρικού προγραμματισμού για προγραμματισμό ραντεβού για να καλύψει όλες τις προκλήσεις που μπορεί να αντιμετωπίσει κατά τη διάρκεια του προγραμματισμού ραντεβού και της διαχείρισης νοσοκομείων. Αυτό το ηλεκτρονικό λογισμικό κρατήσεων ραντεβού για το γιατρό είναι φιλικό προς το χρήστη τόσο στα εμπρός όσο και στα πίσω άκρα, πράγμα που σημαίνει ότι η διαδικασία κράτησης ραντεβού και η διαχείριση του Book My Doc είναι εύκολη τόσο για τους γιατρούς όσο και για τους ασθενείς. Για την προσέλκυση νέων ασθενών, η ιατρική πρακτική πρέπει να υιοθετήσει ψηφιοποίηση. Χρησιμοποιώντας το Book My Doc κάνουν μια αρχή για την παροχή ψηφιακής υγειονομικής περίθαλψης.

  Οι διαφορές του με το Doctorganize είναι ότι έχει επικοινωνία με Νοσοκομεία, Κλινικές και μέσω της εφαρμογής μπορούν να πραγματοποιηθούν πληρωμές είτε για επισκέψεις είτε για φάρμακα.

Κεφάλαιο 2

Προγραμματισμός με Android

2.1 Android Studio

Σχήμα 2.1: Android Studio Logo from [8]

Το Android Studio [8] είναι ένα ολοκληρωμένο προγραμματιστικό περιβάλλον (IDE) για ανάπτυξη εφαρμογών στην πλατφόρμα Android. Ανακοινώθηκε στις 16 Μαΐου 2013 στο συνέδριο Google I/O από την Google ProductManager, Katherine Chou. Το Android Studio είναι διαθέσιμο ελεύθερα με την άδεια Apache License 2.0.

Το Android Studio ήταν διαθέσιμο σε πρώιμο στάδιο για προεπισκόπηση ξεκινώντας από την έκδοση 0.1 τον Μάιο του 2013, έπειτα ξεκίνησε το δοκιμαστικό στάδιο από την έκδοση 0.8 που βγήκε τον Ιούνιο του 2014. Η πρώτη σταθερή έκδοση βγήκε το Δεκέμβριο του 2014, με την έκδοση 1.0. Βασισμένο στο λογισμικό της JetBrains' IntelliJ IDEA, το Android Studio σχεδιάστηκε αποκλειστικά για προγραμματισμό Android. Είναι διαθέσιμο για Windows, Mac OS X και Linux, και αντικατέστησε τα Eclipse Android-Development Tools (ADT) ως το κύριο IDE της Google για ανάπτυξη εφαρμογών Android.

2.2 Android SDK

Το Android SDK [7] (κιτ ανάπτυξης λογισμικού) είναι ένα σύνολο εργαλείων ανάπτυξης που χρησιμοποιούνται για την ανάπτυξη εφαρμογών για πλατφόρμα Android. Το κιτ Android SDK περιλαμβάνει τα εξής:

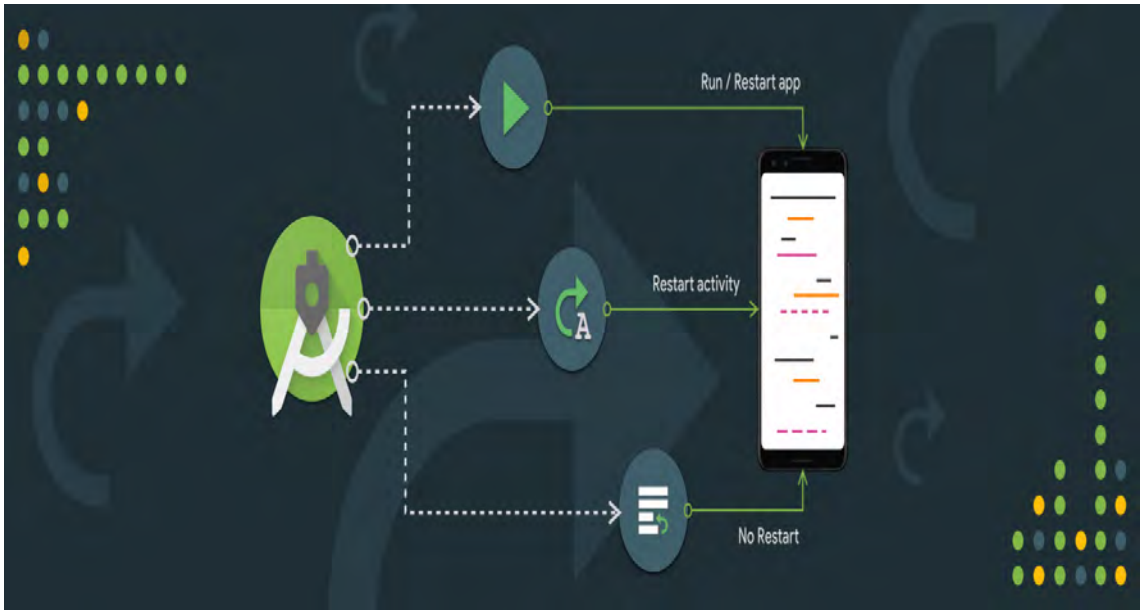
- Απαιτούμενες βιβλιοθήκες
- Debugger
- Έναν εξομοιωτή
- Σχετική τεκμηρίωση για τα Interfaces της εφαρμογής Android (API)
- Δείγμα κώδικα πηγής

- Tutorials για το λειτουργικό σύστημα Android

Κάθε φορά που η Google κυκλοφορεί μια νέα έκδοση του Android, κυκλοφορεί επίσης ένα αντίστοιχο SDK. Για να είναι σε θέση να γράφουν προγράμματα με τις πιο πρόσφατες λειτουργίες, οι προγραμματιστές πρέπει να κατεβάσουν και να εγκαταστήσουν το SDK κάθε εκδόσης για το συγκεκριμένο τηλέφωνο.

Οι πλατφόρμες ανάπτυξης που είναι συμβατές με το SDK περιλαμβάνουν λειτουργικά συστήματα όπως Windows (XP ή νεότερα), Linux (οποιαδήποτε πρόσφατη διανομή Linux) και Mac OS X (10.4.9 ή νεότερη έκδοση). Τα στοιχεία του Android SDK μπορούν να ληφθούν ξεχωριστά. Τα πρόσθετα είναι επίσης διαθέσιμα για λήψη.

2.3 Android Emulators



Σχήμα 2.2: Android Studio Emulators from [4]

2.3.1 Απαιτήσεις και Προτάσεις

Το Android Emulator [4] έχει επιπλέον απαιτήσεις πέρα από τις βασικές απαιτήσεις συστήματος για το Android Studio:

- Εργαλεία SDK 26.1.1 ή νεότερο
- Επεξεργαστή 64-bit
- Windows: CPU με υποστήριξη UG (χωρίς περιορισμούς)
- HAXM 6.2.1 ή νεότερη (συνιστάται HAXM 7.2.0 ή μεταγενέστερη)

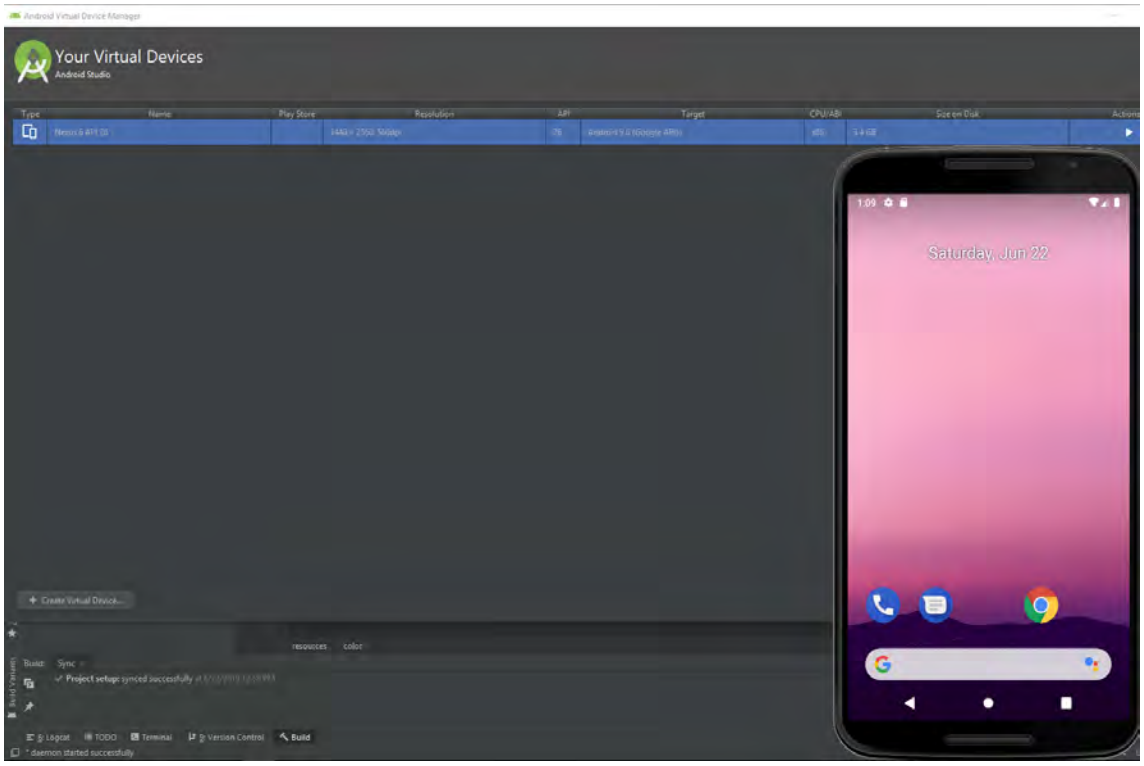
Η χρήση επιτάχυνσης υλικού έχει πρόσθετες απαιτήσεις στα Windows και στο Linux:

- Επεξεργαστής Intel σε Windows ή Linux: επεξεργαστής Intel με υποστήριξη Intel VT-x, Intel EM64T (Intel 64) και λειτουργία Execute Disable (XD) Bit
- Επεξεργαστής AMD σε Linux: Επεξεργαστής AMD με υποστήριξη για AMD Virtualization (AMD-V) και Supplementary Streaming SIMD Extensions 3 (SSSE3).
- Επεξεργαστής AMD σε Windows: έκδοση Android Studio 3.2 ή νεότερη έκδοση και Windows 10 Απριλίου 2018 ή υψηλότερη έκδοση για λειτουργικότητα πλατφόρμας Windows Hypervisor (WHPX).
- Για να εργαστείτε με το Android 8.1 (επίπεδο 27 του API) και τις υψηλότερες εικόνες του συστήματος, μια προσαρμοσμένη κάμερα web πρέπει να έχει τη δυνατότητα να καταγράφει καρέ 720p.

2.3.2 Android Virtual Devices

Κάθε εμφάνιση του Android Emulator χρησιμοποιεί μια εικονική συσκευή Android (AVD) για να καθορίσει την έκδοση Android και τα χαρακτηριστικά του υλικού της προσομοιωμένης συσκευής. Για να δοκιμάσουμε αποτελεσματικά την εφαρμογή μας, θα πρέπει να δημιουργήσουμε ένα αρχείο AVD που θα μοντελοποιεί κάθε συσκευή στην οποία έχει σχεδιαστεί η εκτέλεση της εφαρμογής. Για να δημιουργήσουμε και να διαχειριστούμε AVDs, χρησιμοποιούμε το AVD Manager.

Κάθε AVD λειτουργεί ως ανεξάρτητη συσκευή, με δική του ιδιωτική αποθήκευση δεδομένων χρήστη, κάρτας SD κ.ο.κ. Από προεπιλογή, ο εξομοιωτής αποθηκεύει τα δεδομένα χρήστη, τα δεδομένα της κάρτας SD και την προσωρινή μνήμη σε έναν κατάλογο συγκεκριμένο για αυτό το AVD. Όταν εκκινούμε τον εξομοιωτή, φορτώνει τα δεδομένα χρήστη και τα δεδομένα της κάρτας SD από τον κατάλογο AVD.

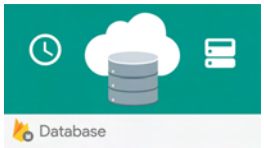


Σχήμα 2.3: AVD Manager

2.4 Firebase Database

2.4.1 Realtime Database

Η Realtime Database [22] είναι μια cloud-hosted βάση δεδομένων NoSQL [36], με υποστήριξη SDK για iOS, Android και το web. Συνεργάζεται εύκολα με τα άλλα εργαλεία της Firebase [26] για έλεγχο ταυτότητας, αποθήκευση αρχείων, αναλυτικά στοιχεία και άλλα. Αυτό το εργαλείο αποθηκεύει δεδομένα σε έγγραφα JSON, οπότε όλα είναι είτε ένα κλειδί είτε μια τιμή. Ο συγχρονισμός δεδομένων χρησιμοποιεί web sockets. Η Realtime Database [24] χειρίζεται επίσης ενημερώσεις όταν μια συσκευή είναι εκτός σύνδεσης, συγχρονίζοντας τις αλλαγές όταν το δίκτυο επανασυνδεθεί.



Σχήμα 2.5: RealTime Database from [21]

Η Realtime Database [21] απαιτεί να γραφεί το μεγαλύτερο μέρος του κώδικα εφαρμογής του συστήματός στον πελάτη. Αυτή η προσέγγιση θα μπορούσε να είναι μια θετική ή αρνητική πτυχή ανάλογα με το πώς το βλέπουμε. Πρέπει να χειριστούμε την περισσότερη λογική στις εφαρμογές των πελατών σας, πράγμα που σημαίνει την αναπαραγωγή κώδικα σε Android, iOS και web, εκτός αν δημιουργήσουμε μια συνάρτηση Cloud Firebase για την αντιμετώπιση των αιτημάτων. Αν το κάνουμε αυτό, όμως, χάνουμε πολλά από τα ενσωματωμένα χαρακτηριστικά του Realtime SDK.

Αυτή η λογική πελάτη απαιτεί επίσης να γράψουμε την επικύρωση δεδομένων μας. Η Realtime

Σχήμα 2.4: Firebase from [22]

Database δεν έχει έννοιες των τύπων δεδομένων. Θα μας επιτρέψει να αποθηκεύσουμε τιμές ως συμβολοσειρές ή αριθμούς ή ένθετα αντικείμενα και συστοιχίες συμβολο σειρών και αριθμών σε οποιοδήποτε πεδίο που θέλουμε. Οι πίνακες, ωστόσο, είναι πραγματικά απλά αντικείμενα με δείκτες που χρησιμοποιούνται ως κλειδιά. Είναι απόλυτα έτοιμο να διαχειριστούμε αυτούς τους τύπους δεδομένων μόνοι μας.

Η Realtime Database προτείνει μείωση των δεδομένων και εξομάλυνση των δεδομένων. Αυτό γίνεται πιο εύκολα από ό, τι γίνεται όταν πρέπει να παρακολουθήσουμε όλες τις τοποθεσίες στις οποίες αντιγράφηκε ένα στοιχείο δεδομένων σε περίπτωση που το στοιχείο ενημερωθεί. Είναι επίσης δύσκολο όταν υποστηρίζουμε ολοκαίνουργια χαρακτηριστικά χωρίς να χρειαστεί να επαναπροσδιορίσουμε μαζικά το υφιστάμενο σχήμα δεδομένων μας.

2.4.2 Firebase Authentication

Σχήμα 2.6: Firebase Authentication Providers from [20]

Οι περισσότερες εφαρμογές πρέπει να γνωρίζουν την ταυτότητα ενός χρήστη. Η γνώση της ταυτότητας ενός χρήστη επιτρέπει σε μια εφαρμογή να αποθηκεύει με ασφάλεια τα δεδομένα χρήστη στο cloud και να παρέχει την ίδια εξατομικευμένη εμπειρία σε όλες τις συσκευές του χρήστη. Η λειτουργία Authentication Firebase [20] παρέχει υπηρεσίες backend, εύχρηστα SDK και έτοιμες βιβλιοθήκες UI για τον έλεγχο της ταυτότητας των χρηστών στην εφαρμογή μας. Υποστηρίζει έλεγχο ταυτότητας χρησιμοποιώντας κωδικούς πρόσβασης, αριθμούς τηλεφώνου, δημοφιλείς παρόχους ομοσπονδιακής ταυτότητας, όπως το Google, το Facebook και το Twitter, και πολλά άλλα.

Το Firebase Authentication ενσωματώνεται με άλλες υπηρεσίες Firebase και αξιοποιεί τα πρό-

τυπα της βιομηχανίας όπως το OAuth 2.0 και το OpenID Connect, έτσι ώστε να μπορεί εύκολα να ενσωματωθεί με το προσαρμοσμένο backend.

Βασικές δυνατότητες

Μπορείτε να συνδεθείτε στους χρήστες στην εφαρμογή Firebase είτε χρησιμοποιώντας το FirebaseUI ως ολοκληρωμένη λύση drop-in auth είτε χρησιμοποιώντας το SDK Authentication Firebase για να ενσωματώσετε με μη αυτόματο τρόπο μία ή περισσότερες μεθόδους σύνδεσης στην εφαρμογή σας.

Για να συνδέσουμε έναν χρήστη στην εφαρμογή μας, λαμβάνουμε πρώτα τα διαπιστευτήρια ελέγχου ταυτότητας από το χρήστη. Αυτά τα διαπιστευτήρια μπορούν να είναι η διεύθυνση ηλεκτρονικού ταχυδρομείου και ο κωδικός πρόσβασης του χρήστη ή ένα διακριτικό OAuth από έναν ομοσπονδιακό πάροχο ταυτότητας. Στη συνέχεια, μεταβιβάζουμε αυτά τα διαπιστευτήρια στο SDK Authentication Firebase. Οι υπηρεσίες backend θα επαληθεύσουν τα διαπιστευτήρια αυτά και θα επιστρέψουν μια απάντηση στον πελάτη.

Μετά την επιτυχή σύνδεση, μπορούμε να αποκτήσουμε πρόσβαση στις βασικές πληροφορίες του προφίλ του χρήστη και να ελέγξουμε την πρόσβαση του χρήστη σε δεδομένα που είναι αποθηκευμένα σε άλλα προϊόντα της Firebase. Μπορούμε επίσης να χρησιμοποιήσουμε το παρεχόμενο διακριτικό ελέγχου ταυτότητας για να επαληθεύσουμε την ταυτότητα των χρηστών στις δικές μας υπηρεσίες back-end.

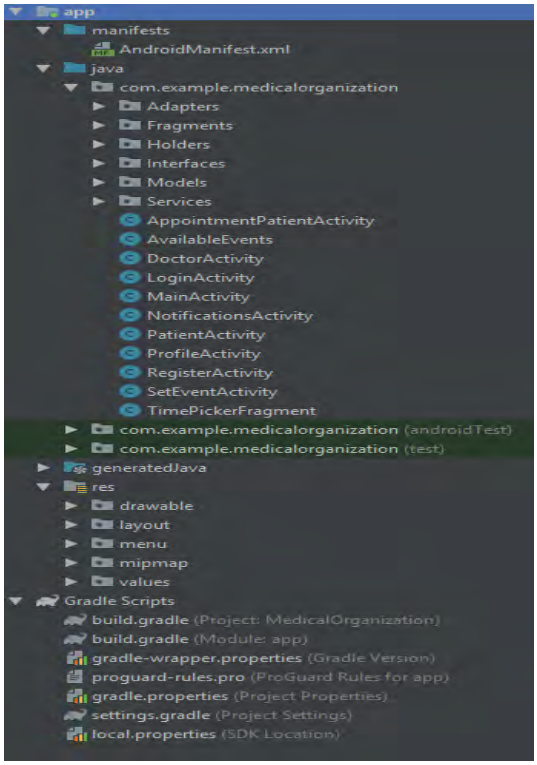
Σημείωση: Από προεπιλογή, οι πιστοποιημένοι χρήστες μπορούν να διαβάζουν και να γράφουν δεδομένα στη βάση δεδομένων Firebase Realtime και Cloud Storage. Μπορούμε να ελέγξετε την πρόσβαση αυτών των χρηστών, τροποποιώντας τους Firebase Realtime Database και τους κανόνες ασφάλειας αποθήκευσης Cloud.

Κεφάλαιο 3

Περιεχόμενα εφαρμογής Android

3.1 Βασικές Πληροφορίες

Στον προγραμματισμό με Android [1] οι εφαρμογές είναι φτιαγμένες σαν ένας συνδυασμός πολλών ανεξάρτητων αρχείων που αλληλεπιδρούν. Υπάρχουν Activities για τις οποίες υπάρχουν και τα αντίστοιχα xml αρχεία, το Android Manifest αρχείο (xml), εικόνες και διανύσματα διάφορων μεγεθών, και χρωμάτων, τα gradle αρχεία και ότι άλλο απαιτεί η υλοποίηση της εφαρμογής. Παρακάτω είναι μια εικόνα που δείχνει την αρχιτεκτονική και την δόμηση των αρχείων:



Σχήμα 3.1: Δομή Αρχείων

3.1.1 Αρχεία XML

Το XML σημαίνει Extensible Markup Language. Η XML είναι μια γλώσσα σήμανσης παρόμοια με την HTML που χρησιμοποιείται για την περιγραφή δεδομένων. Οι ετικέτες XML δεν είναι προκαθορισμένες σε XML. Πρέπει να ορίσουμε τις δικές μας ετικέτες. Το Xml είναι καθαυτό ευανάγνωστο τόσο από άνθρωπο όσο και από μηχανή. Επίσης, είναι επεκτάσιμο και απλό να αναπτυχθεί. Στο Android χρησιμοποιούμε xml για το σχεδιασμό των σχεδιαγραμμάτων μας, επειδή το xml είναι ελαφρύ, ώστε να μην κάνει τη διάταξη μας βαριά. Κάθε οθόνη εφαρμογής Android έχει ορισμένα στοιχεία όπως κουμπί, κείμενο ή εικόνες. Αυτά περιέχονται μέσα στο ViewGroup. Τα σχήματα είναι τα καλύτερα παραδείγματα για τα ViewGroups. Οι διαφορετικοί τύποι διάταξης στο Android είναι Linear Layout, Relative Layout, Absolute Layout, Table Layout and Frame Layout.

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="405dp">
    <LinearLayout
        android:background="@drawable/gradientbackground"
        android:layout_width="match_parent"
        android:layout_height="350dp"
        android:orientation="vertical">
        <ImageView
            android:layout_marginTop="45dp"
            android:layout_gravity="center_horizontal"
            android:layout_width="150dp"
            android:layout_height="wrap_content"
            app:srcCompat="@drawable/ic_users"/>
        <TextView
            android:layout_marginTop="10dp"
            android:layout_gravity="center_horizontal"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/nameTextView"
            android:textColor="#fff"
            android:textStyle="bold"
            android:textSize="21sp"/>
    </LinearLayout>
    <android.support.v7.widget.CardView
        android:layout_width="400dp"
        android:layout_height="120dp"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="275dp">
        <LinearLayout
            android:background="@color/white"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="horizontal"
LinearLayout > RelativeLayout > LinearLayout > ImageView
```

Σχήμα 3.2: Παράδειγμα XML Αρχείου

3.1.2 Αρχείο Manifest.xml

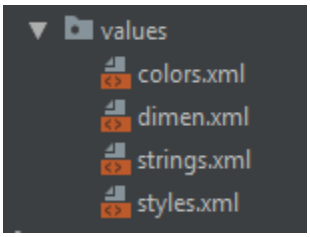
Αυτό το xml [31] χρησιμοποιείται για τον ορισμό όλων των στοιχείων της εφαρμογής μας. Περιλαμβάνει τα ονόματα των πακέτων εφαρμογών, των Activities, των receivers, των υπηρεσιών και των δικαιωμάτων που χρειάζεται η εφαρμογή μας. Για παράδειγμα - Ας υποθέσουμε ότι πρέπει να χρησιμοποιήσουμε το διαδίκτυο στην εφαρμογή μας, τότε πρέπει να καθορίσουμε το δικαίωμα Internet σε αυτό το αρχείο.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.medicalorganization">

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.Light.NoActionBar">
        <activity android:name=".SetEventActivity"></activity>
        <activity android:name=".NotificationsActivity" />
        <activity android:name=".PatientActivity" />
        <activity android:name=".AvailableEvents" />
        <activity android:name=".AppointmentPatientActivity" />
        <activity android:name=".DoctorActivity" />
        <activity android:name=".ProfileActivity" />
        <activity android:name=".RegisterActivity" />
        <activity android:name=".LoginActivity" />
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <service android:name=".Services.MyFirebaseMessagingService">
            <intent-filter>
                <action android:name="com.google.firebase.MESSAGING_EVENT" />
            </intent-filter>
```

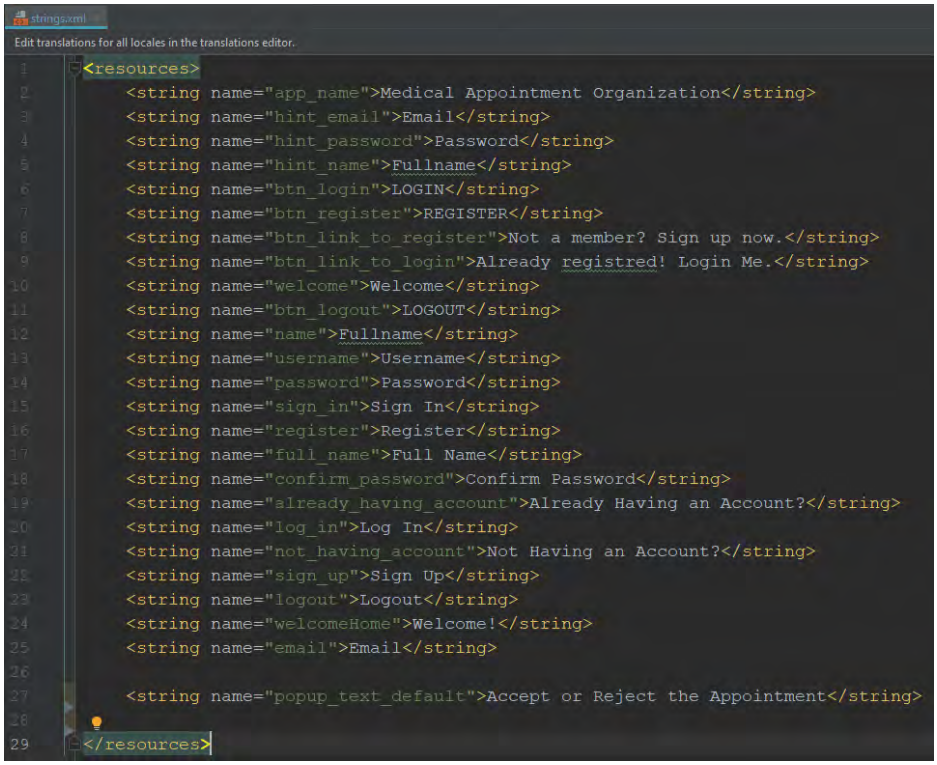
Σχήμα 3.3: AndroidManifest.xml

3.1.3 Strings, Colors, Styles



Σχήμα 3.4: Values

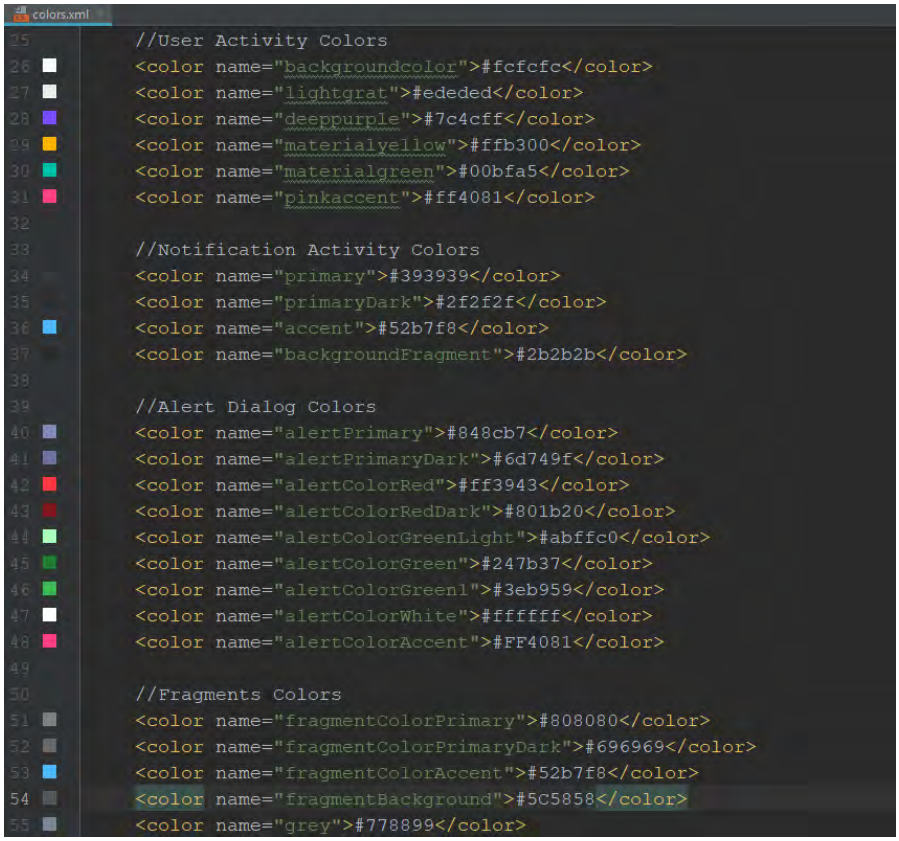
Αυτά τα xml αρχεία χρησιμοποιούνται για να αντικαταστήσουν Hard-Coded Strings, διαφορετικά Styles και Colors που απαιτούνται για την υλοποίηση της εφαρμογής. Παρακάτω είναι εικόνες που δείχνουν τα αρχεία Strings.xml, Styles.xml, Colors.xml



Σχήμα 3.5: Strings.xml

3.1.4 Αρχεία Activities

Εάν έχετε εργαστεί με τη γλώσσα προγραμματισμού C, C ++ ή Java τότε πρέπει να έχετε δει ότι το πρόγραμμά σας ξεκινά από τη λειτουργία main (). Πολύ παρόμοιος τρόπος, το σύστημα Android ξεκινάει το πρόγραμμά του με μια Activity που ξεκινάει με μια κλήση στη μέθοδο επιστροφής κλήσης onCreate (). Υπάρχει μια ακολουθία μεθόδων επανάκλησης που ξεκινούν μια



Σχήμα 3.6: Colors.xml

activity και μια ακολουθία μεθόδων επανάκλησης που κόβουν μια activity όπως φαίνεται στο παρακάτω διάγραμμα κύκλου ζωής μιας activity [17]:

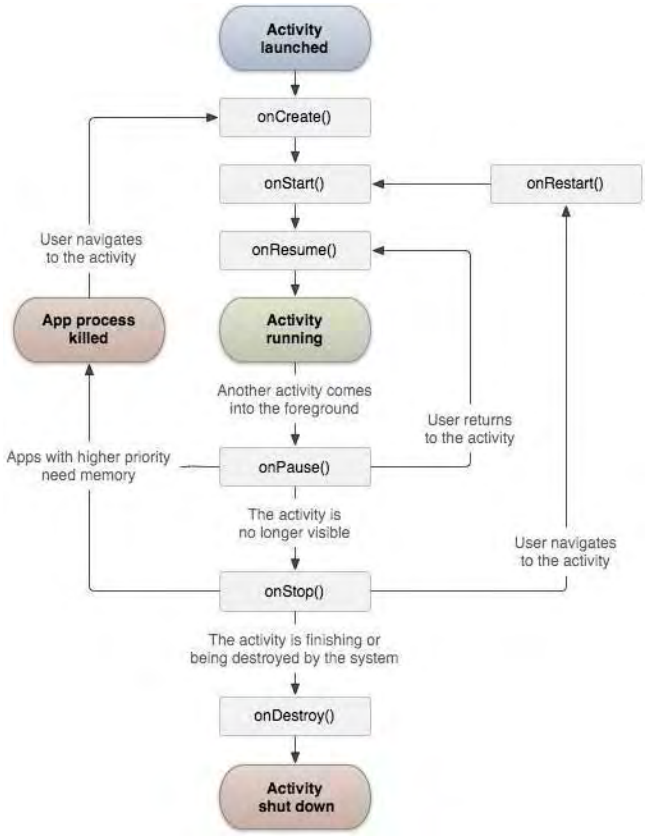
Η κλάση Activity [18] καθορίζει τις ακόλουθες πλάγιες κλήσεις, δηλ. συμβάντα. Δεν χρειάζεται να εφαρμόσουμε όλες τις μεθόδους επανάκλησης. Ωστόσο, είναι σημαντικό να καταλάβουμε το καθένα και να εφαρμόσουμε αυτά που διασφαλίζουν ότι η εφαρμογή μας συμπεριφέρεται όπως περιμένουν οι χρήστες.

Fragments

Η υποκατηγορία FragmentActivity μπορεί να χρησιμοποιήσει την κλάση Fragment για να τροποποιήσει καλύτερα τον κώδικα, να κατασκευάσει πιο εξελιγμένες διεπαφές χρήστη για μεγαλύτερες οθόνες και να βοηθήσει στην κλιμάκωση της εφαρμογής τους μεταξύ μικρών και μεγάλων οθονών.

Activity LifeCycle

Οι Activities στο σύστημα διαχειρίζονται ως στοίβα από Activities. Όταν ξεκινάει μια νέα Activity, συνήθως τοποθετείται στην κορυφή της τρέχουσας στοίβας και γίνεται η τρέχουσα Activity - η προηγούμενη Activity παραμένει πάντοτε κάτω από αυτήν στη στοίβα και δεν θα ξαναγίνει



Σχήμα 3.7: Activities from [17]

πάλι στο προσκήνιο μέχρι να εξέλθει η νέα Activity. Μπορεί να εμφανιστούν στην οθόνη μία ή περισσότερες στοίβες δραστηριοτήτων.
Μια Activity έχει ουσιαστικά τέσσερις καταστάσεις:

- Εάν μια Activity βρίσκεται στο προσκήνιο της οθόνης (στην υψηλότερη θέση της κορυφαίας στοίβας), είναι ενεργή ή τρέχει. Αυτή είναι συνήθως η Activity με την οποία συνεργάζεται ο χρήστης.

- Εάν μια Activity έχει χάσει την εστία, αλλά εξακολουθεί να παρουσιάζεται στο χρήστη, είναι ορατή. Είναι πιθανό, εάν μια νέα μη πλήρους μεγέθους ή διαφανής Activity επικεντρώνεται στην κορυφή της δραστηριότητάς μας, μια άλλη Activity έχει υψηλότερη θέση στη λειτουργία πολλαπλών παραθύρων ή η ίδια η Activity δεν είναι εστιασμένη στην τρέχουσα λειτουργία παραθύρου. Μια τέτοια Activity είναι εντελώς ζωντανή (διατηρεί όλες τις πληροφορίες των μελών και παραμένει συνδεδεμένη με τον διαχειριστή παραθύρων).

- Εάν μια Activity αποκρύπτεται εντελώς από μια άλλη Activity, διακόπτεται ή αποκρύπτεται. Εξακολουθεί να διατηρεί όλες τις πληροφορίες εφαρμογής και μέλους, ωστόσο, δεν είναι πλέον ορατή στον χρήστη, έτσι ώστε το παράθυρό του να είναι κρυμμένο και συχνά θα σκοτωθεί από το σύστημα όταν απαιτείται μνήμη αλλού.

- Το σύστημα μπορεί να αποβάλει τη Activity από τη μνήμη είτε ζητώντας του να τελειώσει είτε απλά να σκοτώσει τη διαδικασία. Όταν εμφανιστεί ξανά στον χρήστη, πρέπει να επανεκκινηθεί πλήρως και να επαναφερθεί στην προηγούμενη κατάσταση.

Starting Activities and Getting Results

```
public class MyActivity extends Activity {
    ...

    static final int PICK_CONTACT_REQUEST = 0;

    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER) {
            // When the user center presses, let them pick a contact.
            startActivityForResult(
                new Intent(Intent.ACTION_PICK,
                new Uri("content://contacts")),
                PICK_CONTACT_REQUEST);
           return true;
        }
        return false;
    }

    protected void onActivityResult(int requestCode, int resultCode,
            Intent data) {
        if (requestCode == PICK_CONTACT_REQUEST) {
            if (resultCode == RESULT_OK) {
                // A contact was picked.  Here we will just display it
                // to the user.
                startActivity(new Intent(Intent.ACTION_VIEW, data));
            }
        }
    }
}
```

Σχήμα 3.8: Intents from [18]

Η μέθοδος startActivity (Intent) χρησιμοποιείται για να ξεκινήσει μια νέα δραστηριότητα, η οποία θα τοποθετηθεί στο επάνω μέρος της στοίβας δραστηριοτήτων. Παίρνει ένα μόνο επιχείρημα, μια πρόθεση, η οποία περιγράφει τη δραστηριότητα που πρόκειται να εκτελεστεί.

Μερικές φορές θέλουμε να πάρουμε ένα αποτέλεσμα πίσω από μια Activity όταν τελειώνει. Για παράδειγμα, μπορούμε να ξεκινήσουμε μια δραστηριότητα που επιτρέπει στον χρήστη να επιλέξει ένα άτομο σε μια λίστα επαφών. όταν τελειώνει, επιστρέφει το άτομο που επιλέχθηκε. Για να το κάνουμε αυτό, καλουμε την έκδοση startActivityForResult (Intent, int) με μια δεύτερη ακέραια παράμετρο που προσδιορίζει την κλήση. Το αποτέλεσμα θα επανέλθει μέσω της μεθόδου onActivityResult (int, int, Intent).

Όταν μια δραστηριότητα εξέρχεται, μπορεί να καλέσει setResult (int) για να επιστρέψει τα δεδομένα πίσω στον γονέα της. Πρέπει πάντα να παρέχει έναν κώδικα αποτελεσμάτων, ο οποίος μπορεί να είναι τα τυπικά αποτελέσματα RESULT_CANCELED, RESULT_OK ή οποιεσδήποτε προσαρμοσμένες τιμές ξεκινούν από το RESULT_FIRST_USER. Επιπλέον, μπορεί προαιρετικά

να επιστρέψει μια πρόθεση που περιέχει οποιαδήποτε πρόσθετα δεδομένα θέλει. Όλες αυτές οι πληροφορίες εμφανίζονται στο Activity.onActivityResult () του γονέα μαζί με το αναγνωριστικό ακέραιου που παρείχε αρχικά.

Εάν μια δραστηριότητα παιδιού αποτύχει για οποιονδήποτε λόγο (όπως συντριβή), η γονική δραστηριότητα θα λάβει ένα αποτέλεσμα με τον κωδικό RESULT_CANCELED.

3.2 Firebase Web API για Notifications

Καθώς οι εφαρμογές εξελίσσονται, είναι όλο και πιο συνηθισμένο να συναντούμε λειτουργίες που συνηθίζεται να συσχετίζονται με μια εγγενή εφαρμογή σε μια web application. Πολλοί ιστότοποι αποστέλλουν ειδοποιήσεις στους χρήστες τους μέσω του προγράμματος περιήγησης για διάφορα συμβάντα που συμβαίνουν στην εφαρμογή ιστού.

3.2.1 Ειδοποιήσεις με Firebase

Η Firebase είναι μια πλατφόρμα που προσφέρει διάφορες υπηρεσίες για mobile and web applications και βοηθά τους προγραμματιστές να δημιουργούν εφαρμογές γρήγορα με πολλά χαρακτηριστικά.

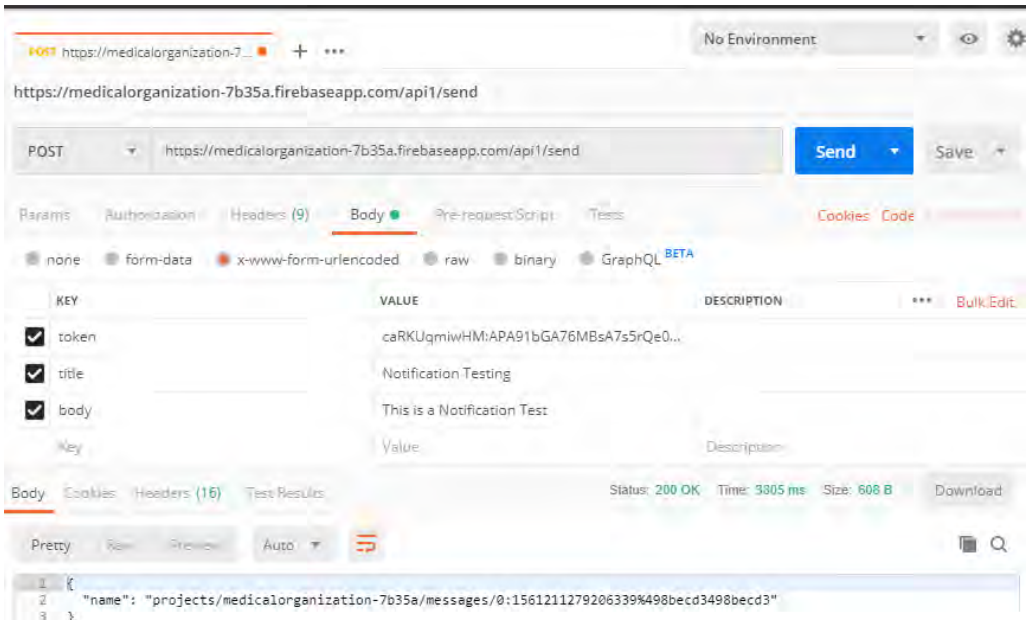
Για να στείλουμε τις ειδοποιήσεις, θα χρησιμοποιήσουμε την υπηρεσία Cloud Messaging [19], η οποία μας επιτρέπει να στέλνουμε μηνύματα σε οποιαδήποτε συσκευή χρησιμοποιώντας αιτήσεις HTTP [27].

3.2.2 Service Workers

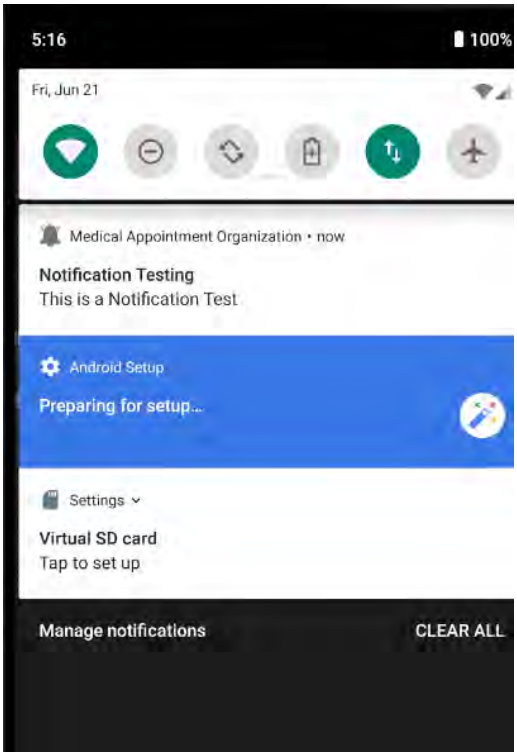
Ένας service worker είναι μια δέσμη ενεργειών στο πρόγραμμα περιήγησής σας εκτελείται στο παρασκήνιο, ξεχωριστά από την ιστοσελίδα ή την εφαρμογή, επιτρέποντας λειτουργίες που δεν απαιτούν αλληλεπίδραση μεταξύ ιστοσελίδας/εφαρμογής και χρήστη. Για να λάβετε το event onMessage, η εφαρμογή μας χρειάζεται έναν service worker. Από προεπιλογή, όταν ξεκινάμε το Firebase, αναζητά ένα αρχείο που ονομάζεται firebase-messaging-sw.js. Αυτός ο service worker θα εισαγάγει βασικά το script που απαιτείται για να εμφανίσει τις ειδοποιήσεις όταν η εφαρμογή μας βρίσκεται στο παρασκήνιο. Πρέπει να προσθέσουμε το firebase-messaging-sw.js στη θέση όπου θα εξυπηρετηθούν τα αρχεία σας.

3.2.3 Στέλνοντας Ειδοποιήσεις

Για να στείλουμε την ειδοποίηση [30], πρέπει να ζητήσουμε από το API του Firebase να το ενημερώσει για το token που θα λάβει ο χρήστης. Για την υλοποίηση της συγκεκριμένης εφαρμογής, αρχικά χρησιμοποίησα την πλατφόρμα Postman. Βασικά, πρέπει να υποβάλουμε ένα αίτημα POST στη διεύθυνση του API της firebase: https://medicalorganization-7b35a.firebaseapp.com/api1/send στέλνοντας ένα JSON [16] στο σώμα αιτήσεων που περιλαμβάνει σίγουρα το token της συσκευής στην οποία θέλω να στείλω την ειδοποίηση, έναν τίτλο και ένα σώμα. Πατώντας Send βλέπουμε ότι η ειδοποίηση έφτασε στην συσκευή (emulator).



(α΄) Postman



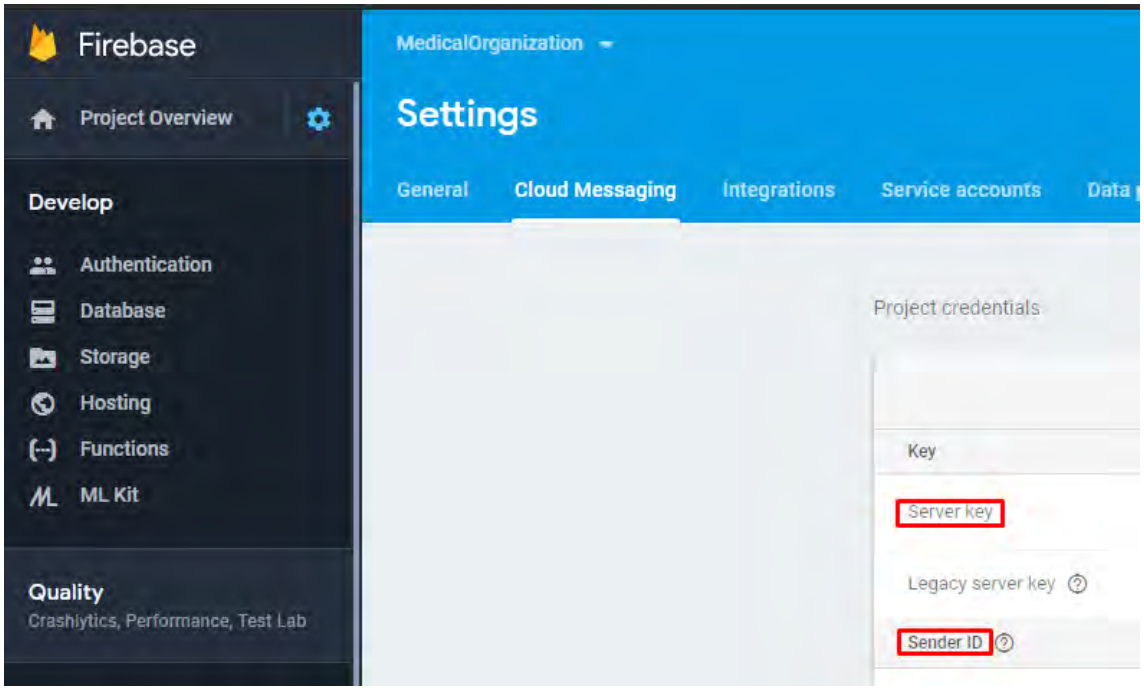
(β΄) Emulator Notification

Σχήμα 3.9: Postman and Notification

3.2.4 Χτίζοντας το Web API

Για την υλοποίηση του Web API ακολούθησα τα εξής βήματα:

- Πήρα τα credentials της εφαρμογής (Server Key, Sender ID) από τις Ρυθμίσεις του Firebase Console.



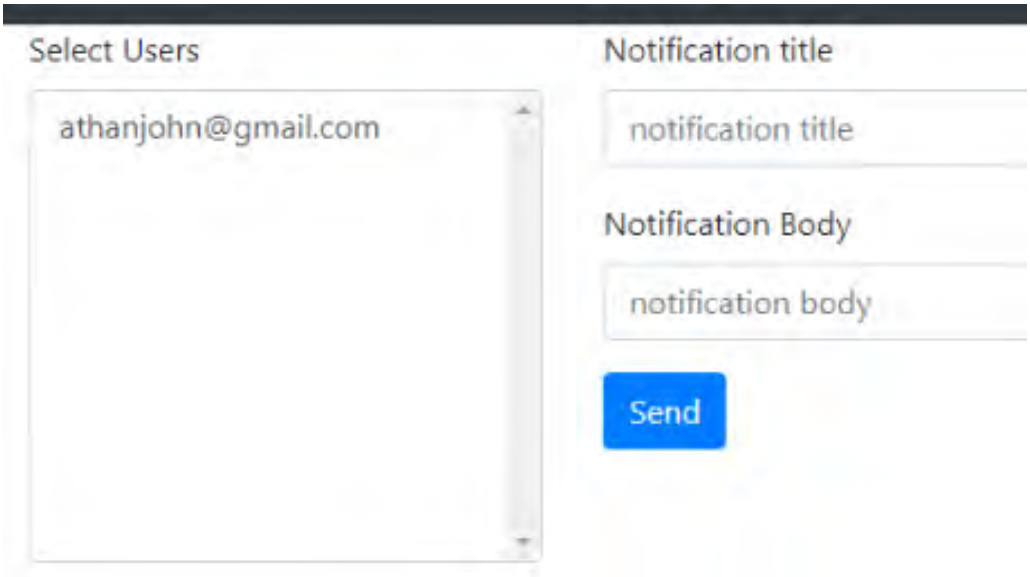
Σχήμα 3.10: Credentials

- Αρχικοποίησα ένα αρχείο HTML με σκοπό να παίρνω κάθε φορά το Token, κάνωντας login με τα στοιχεία κάποιου χρήστη.

Σχήμα 3.11: Index.html

Σχήμα 3.12: Index

- Μετά την επιτυχή σύνδεση του χρήστη υπάρχει νεο HTML αρχείο που δείχνει άλλους users ώστε ο χρήστης να επιλέξει σε ποιόν ή ποιούς θα στείλει την ειδοποίηση.



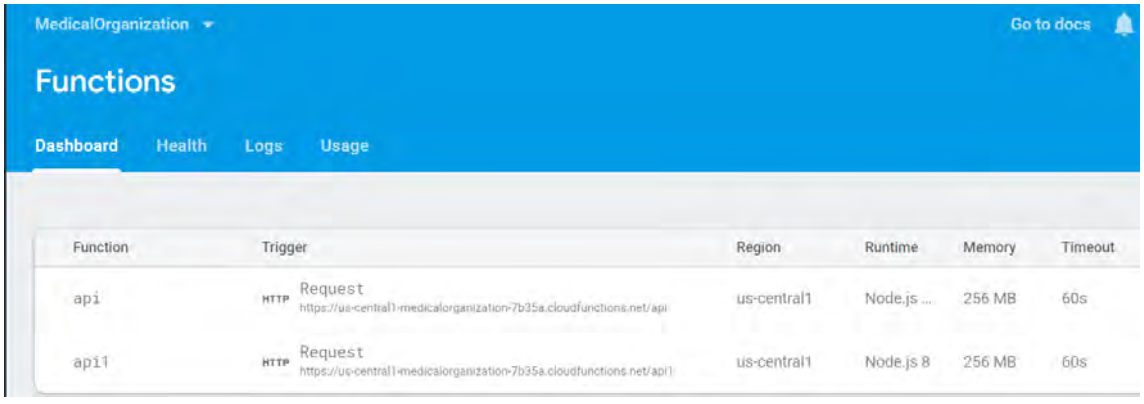
Σχήμα 3.13: Admin Page-After Login

- Υλοποίηση αρχεία scripts(Service Workers) χρησιμοποιώντας Node Javascript [12] για τους σκοπούς που προαναφέρθηκαν. Χρησιμοποιώντας ένα LocalHost από Browser έκανα κάποια tests και έφτασα στο συμπέρασμα ότι οι ειδοποιήσεις στέλνονται και φτάνουν σωστά

- Το μόνο που απέμενε ήταν να γίνει upload το API στην κονσόλα της Firebase ώστε να χειρίζομαι πλέον τις ειδοποιήσεις μέσω της εφαρμογής Android.

```
function getAccessToken(){
    return new Promise(function(resolve, reject){
        var key = require("./service-account.json");
        var jwtClient = new google.auth.JWT(
            key.client_email,
            null,
            key.private_key,
            SCOPES,
            null
        );
        jwtClient.authorize(function(err, tokens){
            if(err){
                reject(err);
                return;
            }
            resolve(tokens.access_token);
        });
    });
}
```

Σχήμα 3.14: Get Token with Node.js

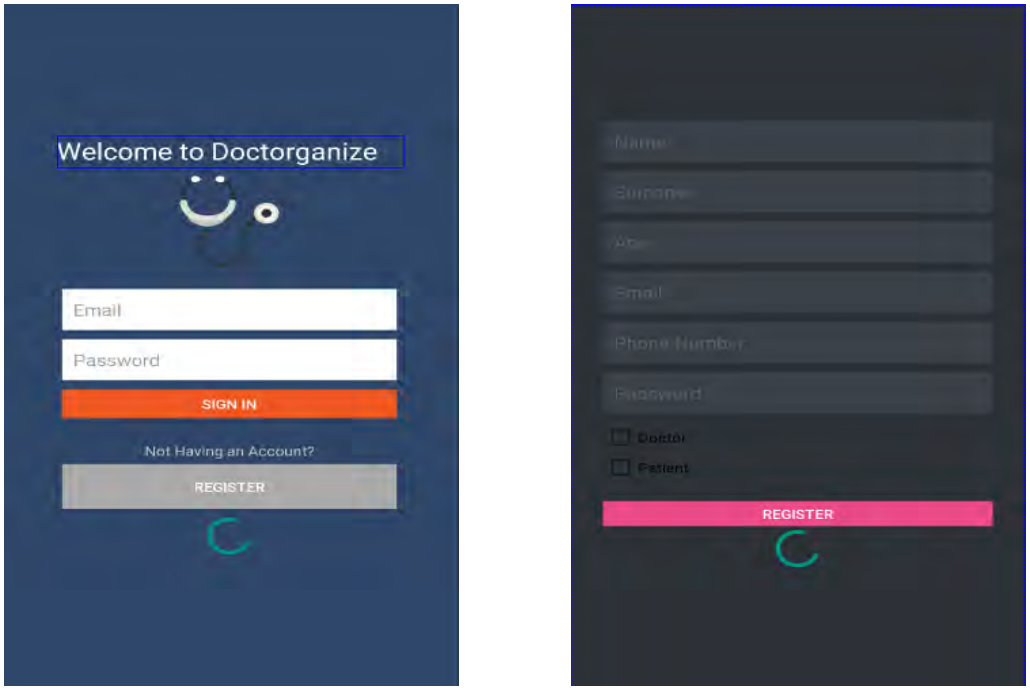


Σχήμα 3.15: API Deployed on Firebase Console

Κεφάλαιο 4

Παρουσίαση της εφαρμογής

4.1 Login and Registration



(α΄) Login (β΄) Register

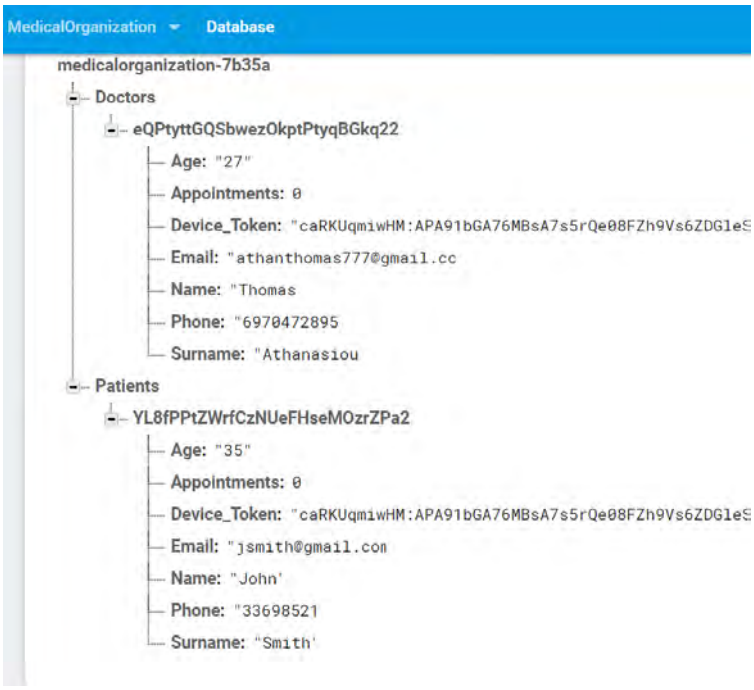
Σχήμα 4.1: Login and Registration

4.1.1 Register

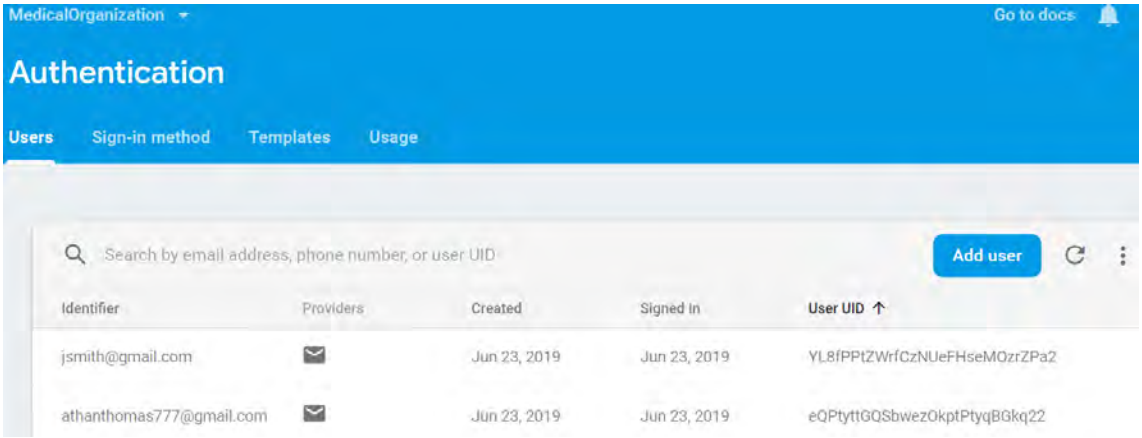
Υπάρχουν 2 είδη χρηστών στην εφαρμογή. Ιατροί και Ασθενείς. Κάθε χρήστης για να συμμετέχει και να χρησιμοποιεί την εφαρμογή πρέπει αρχικά να κάνει εγγραφή. Πρέπει να πληκτρολογήσει σημαντικά στοιχεία όπως ονομ/επώνυμο, ηλικία, mail, αριθμό τηλεφώνου και έναν κωδικό. Στη συνέχεια ανάλογα την ιδιότητα του χρήστη πρέπει να επιλέξει αν είναι Ιατρός ή Ασθενής. Αφού συμπληρωθούν όλα τα στοιχεία και εφόσον δεν υπάρχει χρήστης με το ίδιο email μετά τη διαδικα-

σία Register τα στοιχεία του χρήστη εισάγωνται στην Firebase Realtime Database, καθώς επίσης και στους χρήστες του Authentication Feature της Firebase.

Κάθε χρήστης έχει ένα μοναδικό id. Μετά από την επιτυχή εγγραφή κάποιων χρηστών η βάση δεδομένων δείχνει έτσι:



Σχήμα 4.2: Firebase Realtime Database screenshot



Σχήμα 4.3: Firebase Authentication Feature

4.1.2 Login

Firebase Users

Το Firebase User [23] αντιπροσωπεύει έναν λογαριασμό χρήστη που έχει εγγραφεί στην εφαρμογή. Οι εφαρμογές έχουν συνήθως πολλούς εγγεγραμμένους χρήστες.

Οι παρουσίες χρηστών είναι ανεξάρτητες από τις περιπτώσεις Authentication Firebase, έτσι μπορούμε να έχουμε αρκετές αναφορές σε διαφορετικούς χρήστες μέσα στο ίδιο πλαίσιο και να καλούμε οποιαδήποτε από τις μεθόδους τους.

Sign-in Providers

Για να κάνουμε έναν χρήστη να συνδεθεί στην εφαρμογή μπορούμε να χρησιμοποιήσουμε διάφορους providers όπως email address and password, federated identity providers, and your custom auth system. Επίσης θα μπορούσαμε να χρησιμοποιήσουμε εξωτερικούς providers όπως Google, Facebook, Github και άλλα. Στην συγκεκριμένη εφαρμογή χρησιμοποίησα το email address and password provider.

The Current User

Όταν ένας χρήστης κάνει εγγραφή ή συνδέεται στην εφαρμογή γίνεται αυτόματα ο current User του Auth Instance. Με αυτόν τον τρόπο κρατάμε τα δεδομένα του χρήστη ώστε να μην χρειάζεται να κάνει πάλι σύνδεση στην εφαρμογή, αν η εφαρμογή κλείσει για οποιονδήποτε λόγο.

Firebase ID Tokens

Τα tokens δημιουργούνται από τη Firebase όταν ένας χρήστης συνδεθεί σε μια εφαρμογή. Αυτά είναι JWT που αναγνωρίζουν με ασφάλεια έναν χρήστη σε ένα project Firebase. Περιέχουν βασικές πληροφορίες προφίλ για έναν χρήστη, συμπεριλαμβανομένης της συμβολοσειράς ID του χρήστη, η οποία είναι μοναδική για το project Firebase. Επειδή η ακεραιότητα των αναγνωριστικών μπορεί να επαληθευτεί, μπορούμε να τα στείλουμε σε backend server για να εντοπίσουμε τον χρήστη που είναι συνδεδεμένος αυτήν τη στιγμή.

4.1.3 Υλοποίηση LoginActivity και RegisterActivity

Για την υλοποίηση των LoginActivity και RegisterActivity χρειάζεται να κάνουμε ένα DatabaseReference που να αντιπροσωπεύει τους Ιατρούς και τους Ασθενείς. Στη συνέχεια παίρνουμε κάθε φορά τα στοιχεία που έχει πληκτρολογήσει ο χρήστης (Email and Password) για το Login και (Όνομα, Επώνυμο, Ηλικία, Τηλέφωνο, Email, Password και Ιδιότητα) για το Register, μέσω των EditTexts που αναλογούν σε κάθε στοιχείο.

Παρακάτω φαίνονται κάποια στιγμιότυπα από το Login και το Register Activities καθώς και από τα layouts(xml) τους.

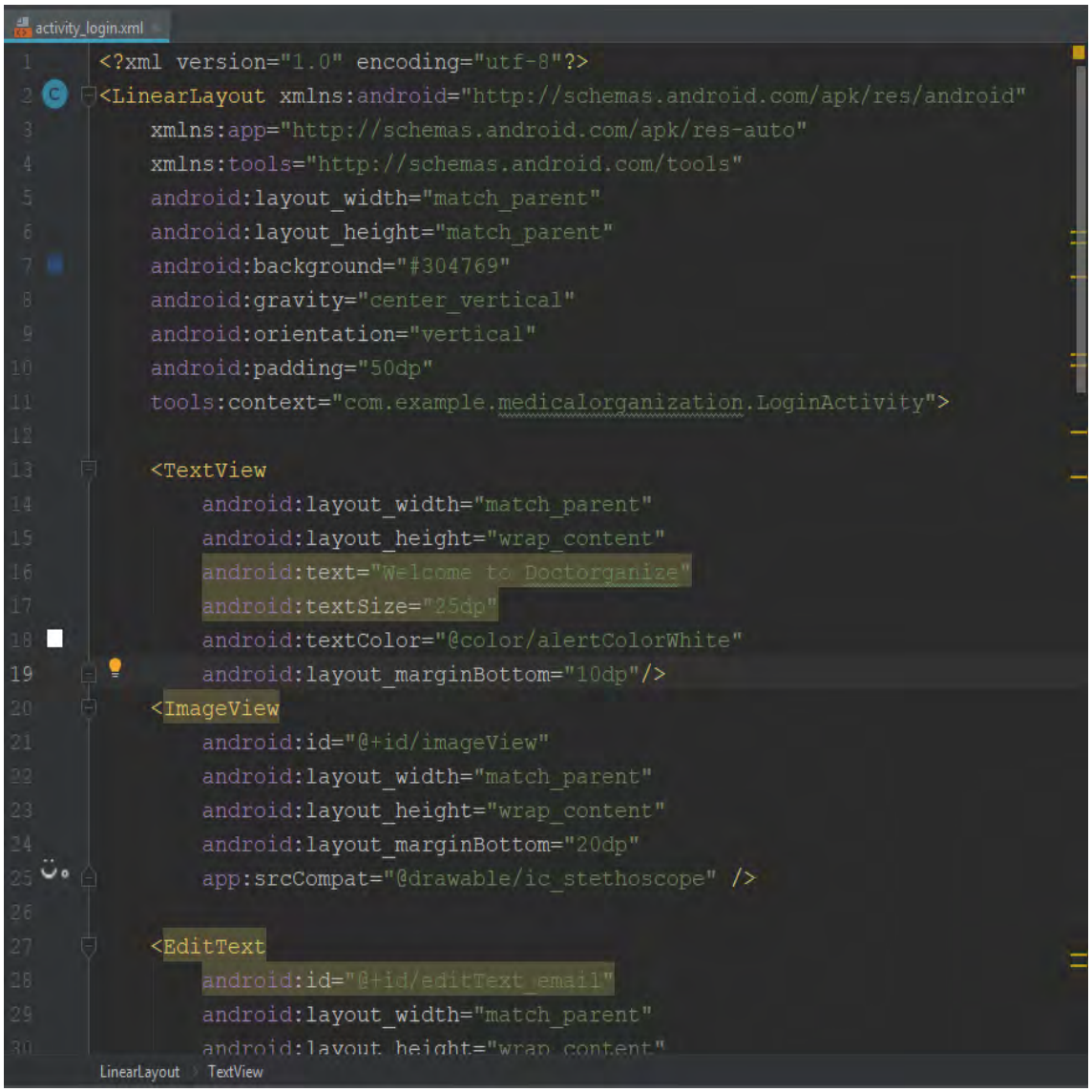
Για να εντάξουμε στην βάση δεδομένων χρήστες πρέπει να φτιάξουμε ανάλογα Model Classes [2]. Οι κλάσεις αυτές δεν διαφέρουν πολύ.

```
public class LoginActivity extends AppCompatActivity {

    private FirebaseDatabase mFirebaseDatabase;
    private DatabaseReference mDoctorsDatabaseReference, mPatientsDatabaseReference;
    private FirebaseAuth mAuth;

    private EditText email, password;
    private ProgressBar progressBar;
    public int identityId;

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

        mFirebaseDatabase = FirebaseDatabase.getInstance();
        mDoctorsDatabaseReference = mFirebaseDatabase.getReference().child("Doctors");
        mPatientsDatabaseReference = mFirebaseDatabase.getReference().child("Patients");
        mAuth = FirebaseAuth.getInstance();
        email =(EditText)findViewById(R.id.editText_email);
        password = (EditText)findViewById(R.id.editText_password);
        progressBar = (ProgressBar)findViewById(R.id.progressbar);
        progressBar.setVisibility(View.GONE);

    }
```

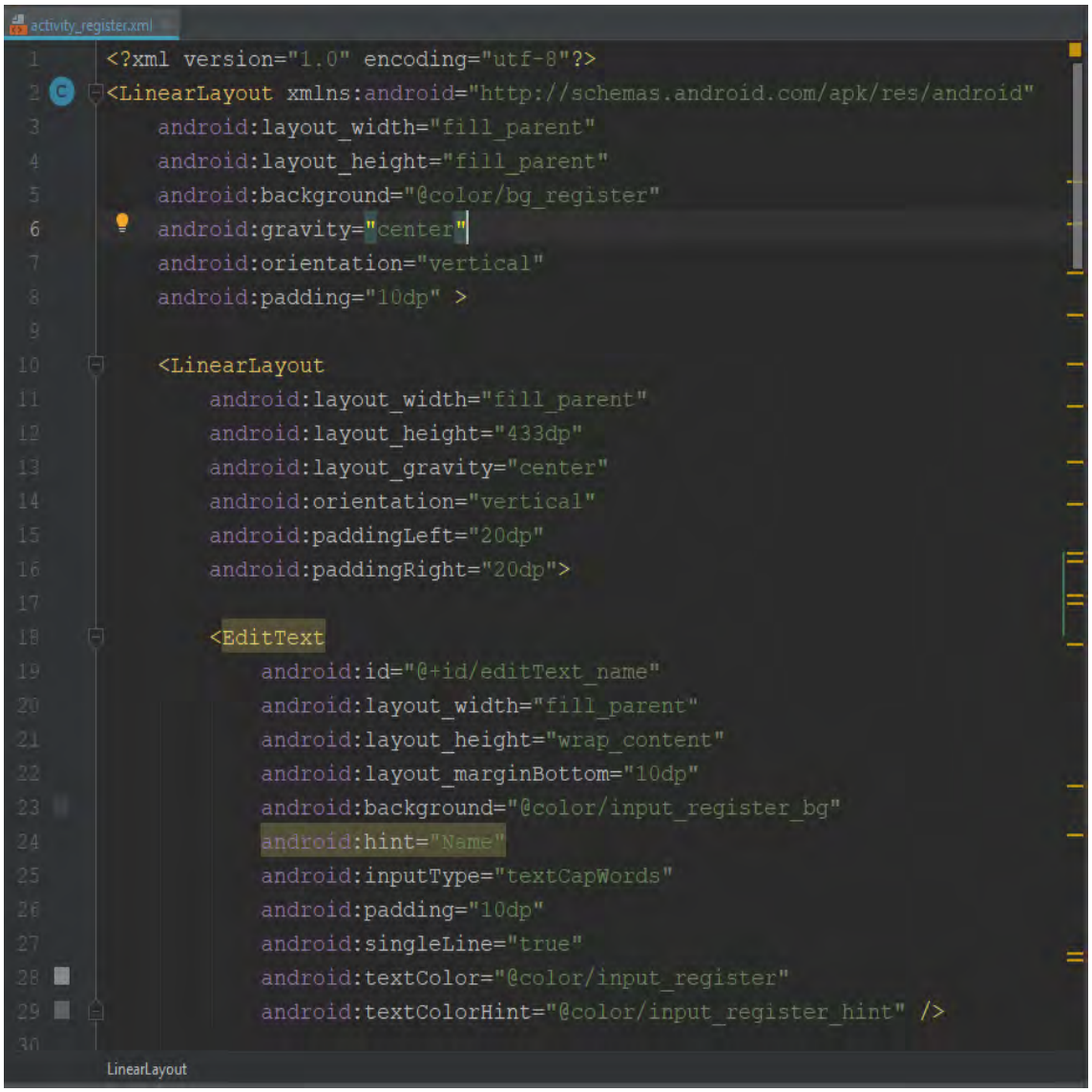
Σχήμα 4.4: Login Activiy

```
public class RegisterActivity extends AppCompatActivity {

    private EditText name, surname, age, password, email, phone;

    private ProgressBar progressBar;
    private CheckBox identity_doctor, identity_patient;
    public int identityId;

    private FirebaseDatabase mFirebaseDatabase;

    private FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        name = (EditText)findViewById(R.id.editText_name);
        surname = (EditText)findViewById(R.id.editText_surname);
        age = (EditText)findViewById(R.id.editText_age);
        email =(EditText)findViewById(R.id.editText_email);
        phone = (EditText)findViewById(R.id.editText_phone);
        password = (EditText)findViewById(R.id.editText_password);
        identity_doctor = (CheckBox)findViewById(R.id.checkBox_doctor);
        identity_patient = (CheckBox)findViewById(R.id.checkBox_patient);
        progressBar = (ProgressBar)findViewById(R.id.progressbar);
        progressBar.setVisibility(View.GONE);

        mFirebaseDatabase = FirebaseDatabase.getInstance();
        mAuth = FirebaseAuth.getInstance();
```

Σχήμα 4.5: Register Activity



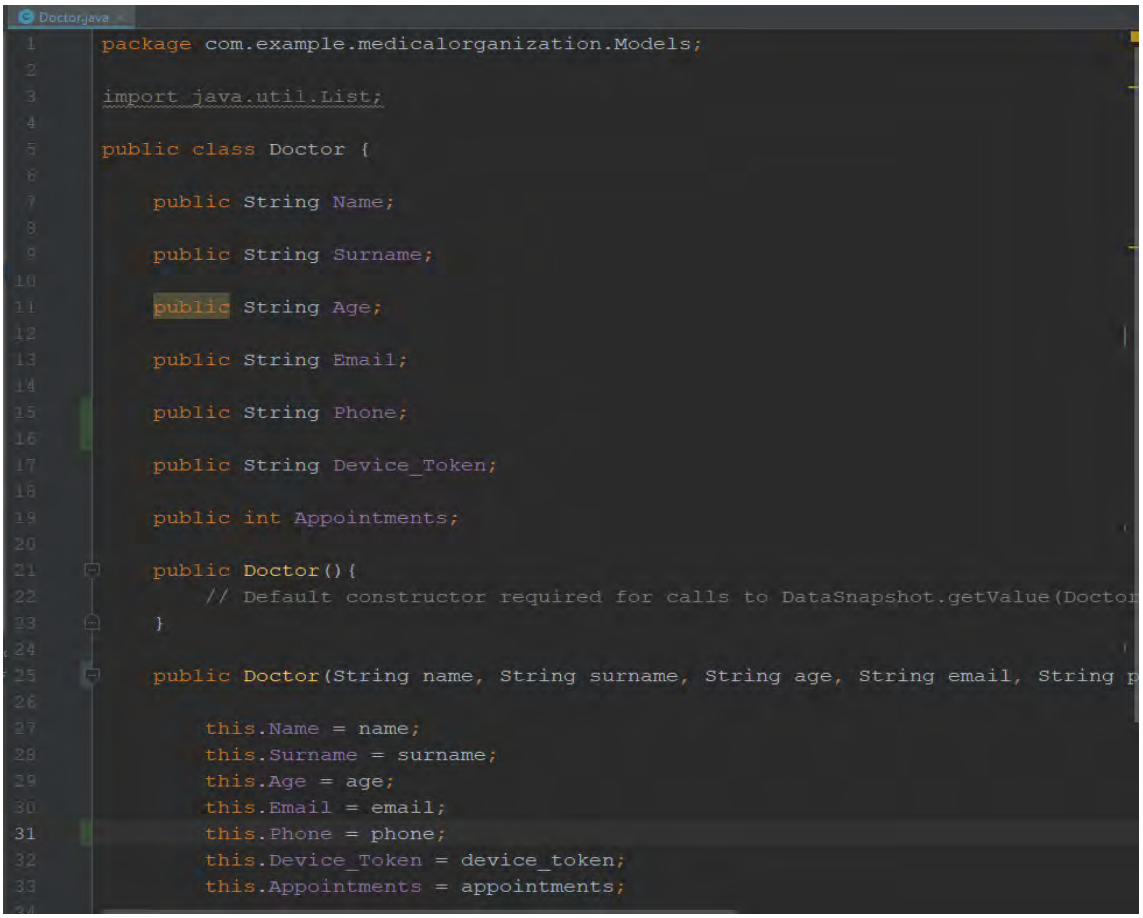
Σχήμα 4.6: Login Layout



Σχήμα 4.7: Register Layout

4.1.4 Model Class Doctor

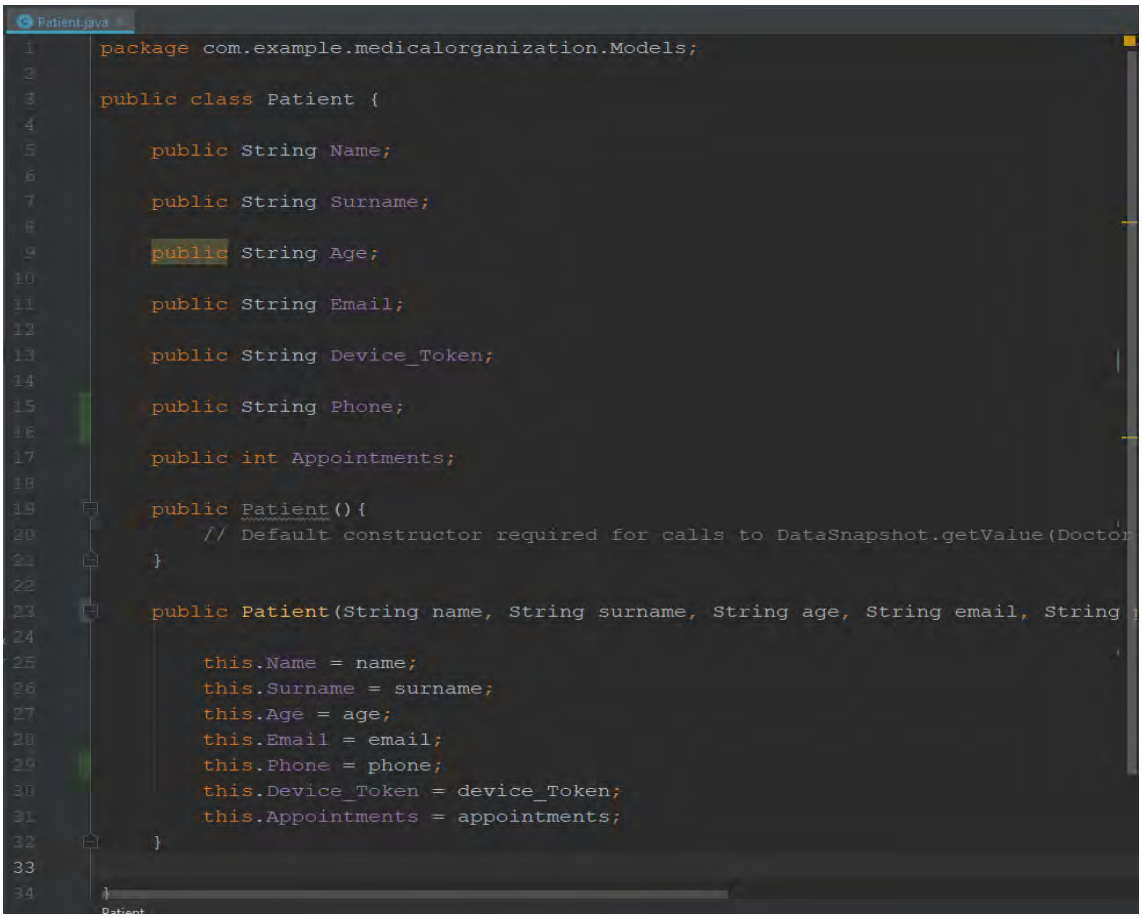
Αυτή η κλάση έχει τις παραμέτρους που χρειάζεται κάθε Ιατρός καθώς και κάποιους Constructors.



Σχήμα 4.8: Doctor Class

4.1.5 Model Class Patient

Αυτή η κλάση έχει τις παραμέτρους που χρειάζεται κάθε Ασθενής καθώς και κάποιους Constructors.



Σχήμα 4.9: Patient Class

Για να κάνουμε μία εγγραφή φτιάχνουμε ανάλογα την ιδιότητα του χρήστη ένα αντικείμενο με βάση τα στοιχεία που έχουμε πάρει από τα layouts. Επίσης πολύ σημαντικό είναι να παίρνουμε το token της συσκευής του χρήστη κάθε φορά που συνδέεται αλλά και όταν κάνει την εγγραφή ώστε να μπορεί να επικοινωνήσει με ειδοποιήσεις με άλλους χρήστες. Το token πρέπει να ανανεώνεται κάθε φορά που ένας χρήστης συνδέεται στην εφαρμογή, έτσι ώστε σε περίπτωση που συνδεθεί από άλλη συσκευή να μην χάσει ειδοποιήσεις που πιθανώς πρέπει να λάβει.

```
//if doctor
Doctor doctor = new Doctor(str_name, str_surname,
        str_age, str_email,
        str_phone, token[0],
        appointments: 0);
```

Σχήμα 4.10: Doctor Object

```
//if Patient
Patient patient = new Patient(str_name,
        str_surname, str_age,
        str_email, str_phone,
        token[0], appointments: 0);
```

Σχήμα 4.11: Patient Object

```
//GET REGISTRATION TOKEN
FirebaseInstanceId.getInstance().getInstanceId()
        .addOnCompleteListener((task) -> {
                if(task.isSuccessful()){
                    //get registration token
                    token[0] = task.getResult().getToken();
                }
                else{
                    Toast.makeText(getApplicationContext(), text: "
                }
        });
```

Σχήμα 4.12: Get Registration Token

4.2 Profile Activity and Layout

Όταν ένας Χρήστης κάνει login αυτόματα μεταφέρεται στην Profile Activity όπου φαίνονται κάποια στοιχεία χρήστη.

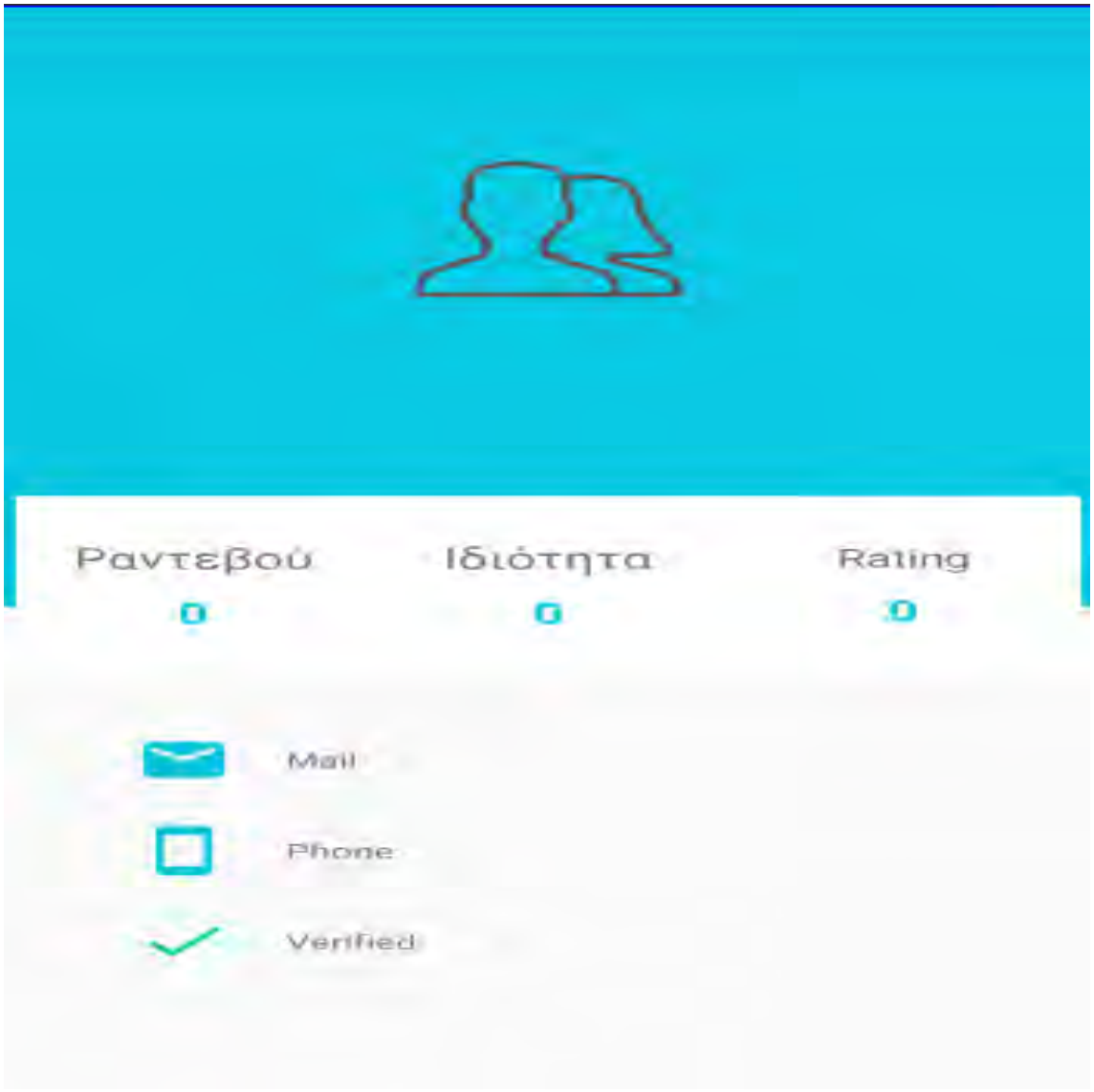
```
package com.example.medicalorganization;

import ...

public class ProfileActivity extends AppCompatActivity {

    private FirebaseAuth mAuth;
    private FirebaseDatabase mFireDatabase;
    private DatabaseReference mDoctorsReference, mPatientReference;
    private TextView name, identity, mail, verifiedEmail, appointments, ratingTextView, ratingTitle, phone;

    @Override
    protected void onCreate(Bundle savedInstanceState) {...}

    private void loadUserInformation() {...}

    private void setDoctorsTextViews(final String id) {...}

    private void averageRating(final String id) {...}

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {...}

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {...}

    @Override
    protected void onStart() {...}

    @Override
    public void onBackPressed() {...}
}
```

Σχήμα 4.13: Profile Activity

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ProfileActivity"
    android:background="@color/graylight"
    android:orientation="vertical">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:background="@color/startblue"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"></android.support.v7.widget.T
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="405dp">
        <LinearLayout
            android:background="@drawable/gradientbackground"
            android:layout_width="match_parent"
            android:layout_height="350dp"
            android:orientation="vertical">
            <ImageView
                android:layout_marginTop="45dp"
                android:layout_gravity="center_horizontal"
                android:layout_width="150dp"
                android:layout_height="wrap_content"
                app:srcCompat="@drawable/ic_users"/>
            <TextView
                android:layout_marginTop="10dp"
```

Σχήμα 4.14: Profile Layout

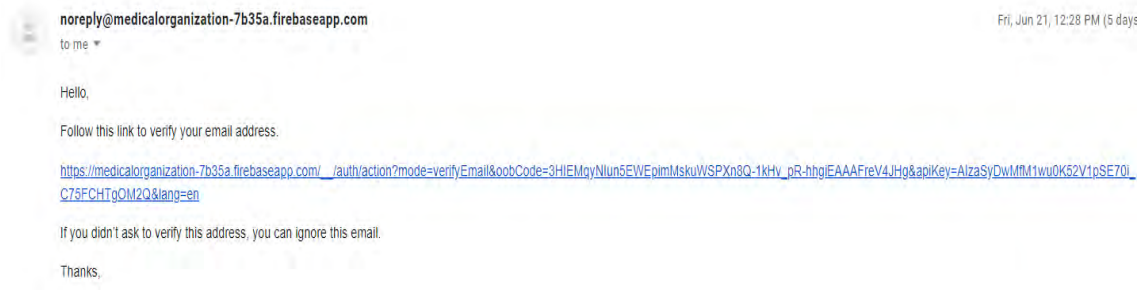
Στην Profile Activity αναγράφονται τα στοιχεία του χρήστη όπως Όνομα, Επώνυμο, Email, Τηλέφωνο, Ραντεβού του χρήστη, Ιδιότητα (Ανάλογα αν είναι Ιατρός η Ασθενής) και βαθμολογία αν είναι Ιατρός.



Σχήμα 4.15: Profile Design

4.2.1 Email Verification

Κάνωντας click στο πεδίο Email στέλνεται ένα mail στον χρήστη για την επιβεβαίωση της διεύθυνσής του.



Σχήμα 4.16: Verification Email

Μόλις ο χρήστης ακολουθήσει το link οδηγείται σε μια σελίδα για την επιτυχή επιβεβαίωση του mail του. Έτσι αν το mail του έχει επιβεβαιωθεί το πεδίο του email στην Profile Activity γίνεται Verified. Το email verification προσφέρεται από την Firebase και χειρίζεται έτσι:

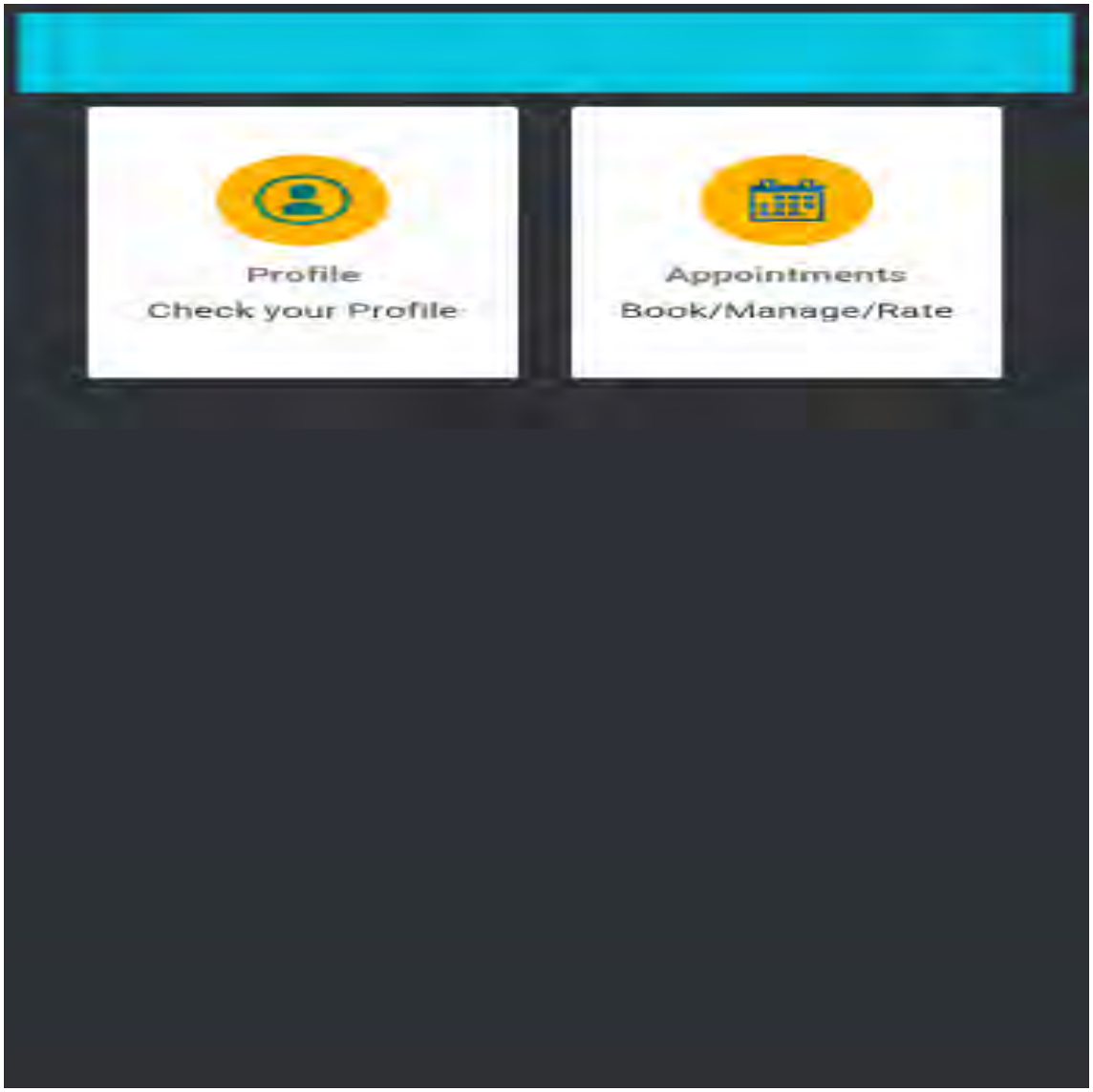
```
//Check if user is verified email address, if not click to verify
if (mAuth.getCurrentUser().isEmailVerified()) {
    verifiedEmail.setText("Email Verified");
} else {
    verifiedEmail.setText("Email not Verified. Click to send Verification Email");
    verifiedEmail.setOnClickListener((v) → {
            mAuth.getCurrentUser().sendEmailVerification().addOnCompleteListener((task) → {
                    Toast.makeText( context: ProfileActivity.this,  text: "Verification email Sent", Toast.LENGTH_SHORT).show();
            });
    });
}
```

Σχήμα 4.17: Email Verification Code

Αναλόγως το αν έχει επιβεβαιωθεί το email του χρήστη το TextView στο layout αλλάζει.

4.3 Η πλευρά του Ασθενή

Όταν ένας χρήστης κάνει Login ή Register τότε αυτόματα οδηγείται στην Profile Activity. Στην πλευρά του ασθενή υπάρχει ένα layout για τις ανάγκες του συγκεκριμένου τύπου χρήστη. Στην πρώτη επιλογή ο χρήστης μπορεί να δει το Profile του και από εκεί ενδεχομένως να επιβεβαιώσει το email του. Στην δεύτερη επιλογή ο χρήστης μπορεί να επιλέξει Ιατρό για να κλείσει νέο ραντεβού, να εξετάσει τι ραντεβού έχει ήδη ζητήσει και έχουν επιβεβαιωθεί και αν ένα ραντεβού έχει ήδη γίνει μπορεί να αξιολογήσει τον αντίστοιχο Ιατρό.



Σχήμα 4.18: Patient Layout

Για τη διευκόλυνση της λειτουργίας της εφαρμογής αντί να χρησιμοποιώ πολλά activities ώστε να αλλάζουν συνεχώς παράθυρα για κάθε λειτουργία, για τους σκοπούς του ασθενή, όταν πατήσει το Appointments Option οδηγείται στην AppointmentPatientActivity. Η συγκεκριμένη activity έχει 3 Fragments.

AppointmentPatientActivity.java

```
23      private ViewPagerAdapter adapter;
24      @Override
25      protected void onCreate(Bundle savedInstanceState) {
26          super.onCreate(savedInstanceState);
27          setContentView(R.layout.activity_appointment_patient);
28
29          mToolbar = (Toolbar)findViewById(R.id.toolbar);
30          setSupportActionBar(mToolbar);
31          getSupportActionBar().setTitle("Medical Organization");
32
33          tabLayout = (TabLayout)findViewById(R.id.tablayout_id);
34          viewPager = (ViewPager)findViewById(R.id.viewpager_id);
35          adapter = new ViewPagerAdapter(getSupportFragmentManager());
36
37          //Add fragment Here
38          adapter.AddFragment(new DoctorsFragment(), title: "Doctors");
39          adapter.AddFragment(new AppointmentsFragment(), title: "My Appointments");
40          adapter.AddFragment(new RateFragment(), title: "Rate");
41
42          viewPager.setAdapter(adapter);
43          tabLayout.setupWithViewPager(viewPager);
44
45          tabLayout.getTabAt( index: 0).setIcon(R.drawable.ic_doctor);
46          tabLayout.getTabAt( index: 1).setIcon(R.drawable.ic_appointment);
47          tabLayout.getTabAt( index: 2).setIcon(R.drawable.ic_star);
48
49
```

Σχήμα 4.19: Appointments Activity with Fragments

4.3.1 Doctor Fragment

Στο πρώτο Fragment ο Ασθενής μπορέι να επιλέξει Ιατρό ώστε να κανονίσει ένα νέο Ραντεβού.

AppointmentPatientActivity.java

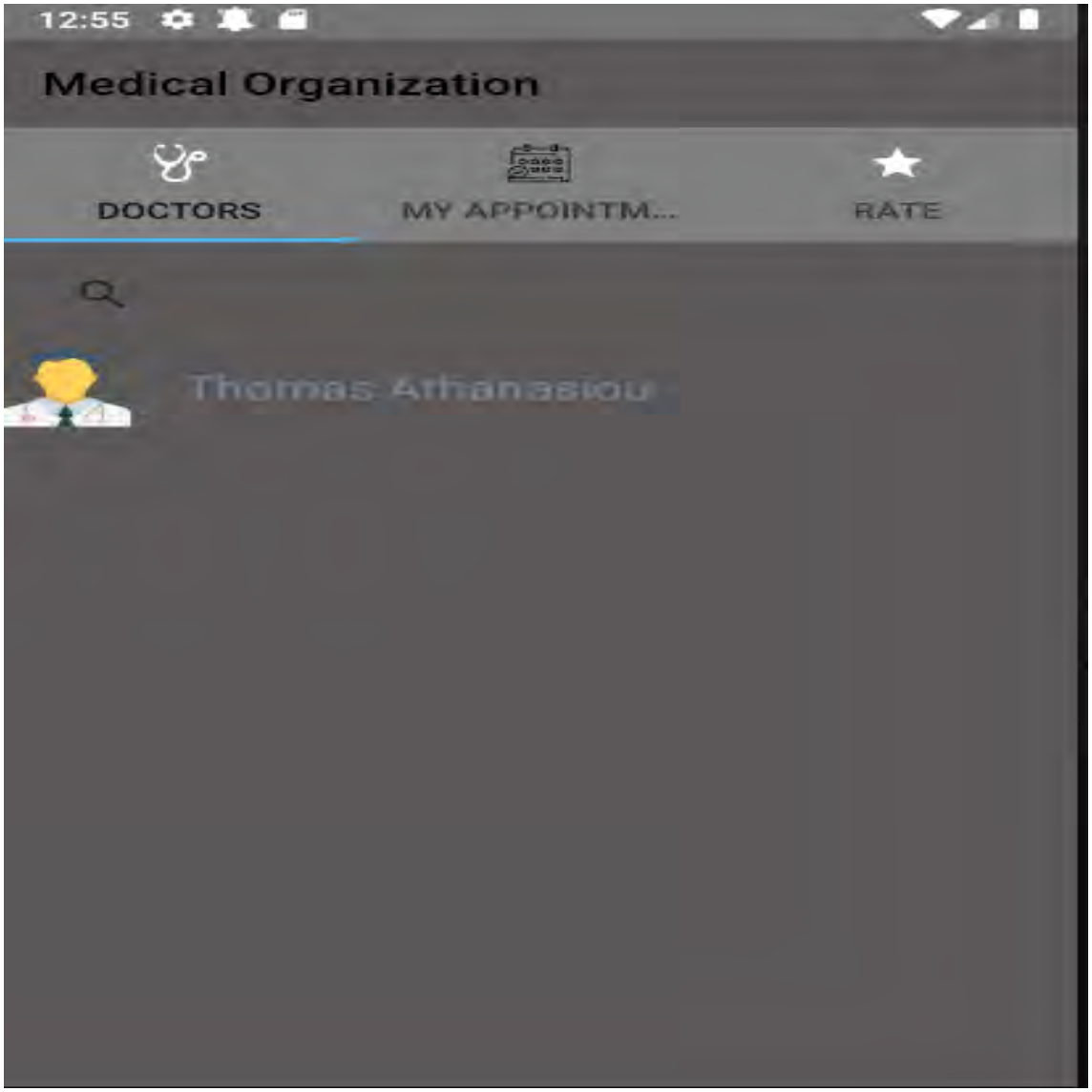
```
23      private ViewPagerAdapter adapter;
24      @Override
25      protected void onCreate(Bundle savedInstanceState) {
26          super.onCreate(savedInstanceState);
27          setContentView(R.layout.activity_appointment_patient);
28
29          mToolbar = (Toolbar)findViewById(R.id.toolbar);
30          setSupportActionBar(mToolbar);
31          getSupportActionBar().setTitle("Medical Organization");
32
33          tabLayout = (TabLayout)findViewById(R.id.tablayout_id);
34          viewPager = (ViewPager)findViewById(R.id.viewpager_id);
35          adapter = new ViewPagerAdapter(getSupportFragmentManager());
36
37          //Add fragment Here
38          adapter.AddFragment(new DoctorsFragment(), title: "Doctors");
39          adapter.AddFragment(new AppointmentsFragment(), title: "My Appointments");
40          adapter.AddFragment(new RateFragment(), title: "Rate");
41
42          viewPager.setAdapter(adapter);
43          tabLayout.setupWithViewPager(viewPager);
44
45          tabLayout.getTabAt( index: 0).setIcon(R.drawable.ic_doctor);
46          tabLayout.getTabAt( index: 1).setIcon(R.drawable.ic_appointment);
47          tabLayout.getTabAt( index: 2).setIcon(R.drawable.ic_star);
48
49
```

Σχήμα 4.20: Doctor Fragment Screenshot

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".Fragments.DoctorsFragment"
    android:orientation="vertical"
    android:background="@color/fragmentBackground">


    <android.support.v7.widget.SearchView
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:id="@+id/searchBar"
        android:padding="10dp"
        android:layout_margin="5dp"
        app:queryHint="Search Doctor"/>

    <android.support.v7.widget.RecyclerView
        android:id="@+id/doctors_list"
        android:scrollbars="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

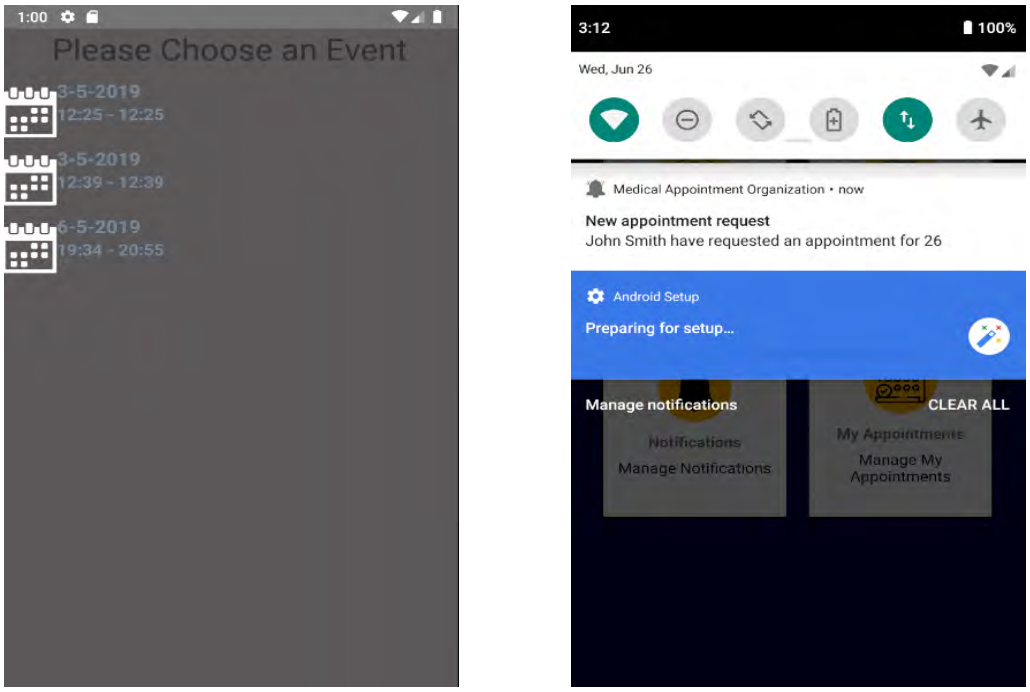
Σχήμα 4.21: Doctor Fragment Screenshot

Υπάρχει όπως φαίνεται ένα SearchBar για να μπορεί ο ασθενής να ψάξει έναν Ιατρό με βάση το όνομά του.



Σχήμα 4.22: Doctor Fragment Emulator

Όταν ο ασθενής επιλέξει έναν από τους διαθέσιμους ιατρούς οδηγείται στα διαθέσιμα ραντεβού που έχει ο συγκεκριμένος Ιατρός. Από εκεί επιλέγει κάποιο ραντεβού και στέλνει ειδοποίηση στον Ιατρό για το αίτημά του.

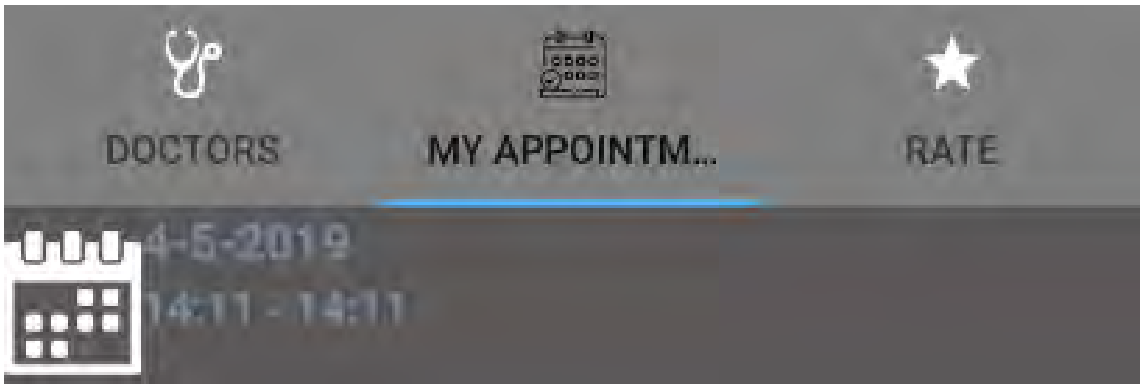


(α΄) Choose Appointment (β΄) Notification on Doctor's Side

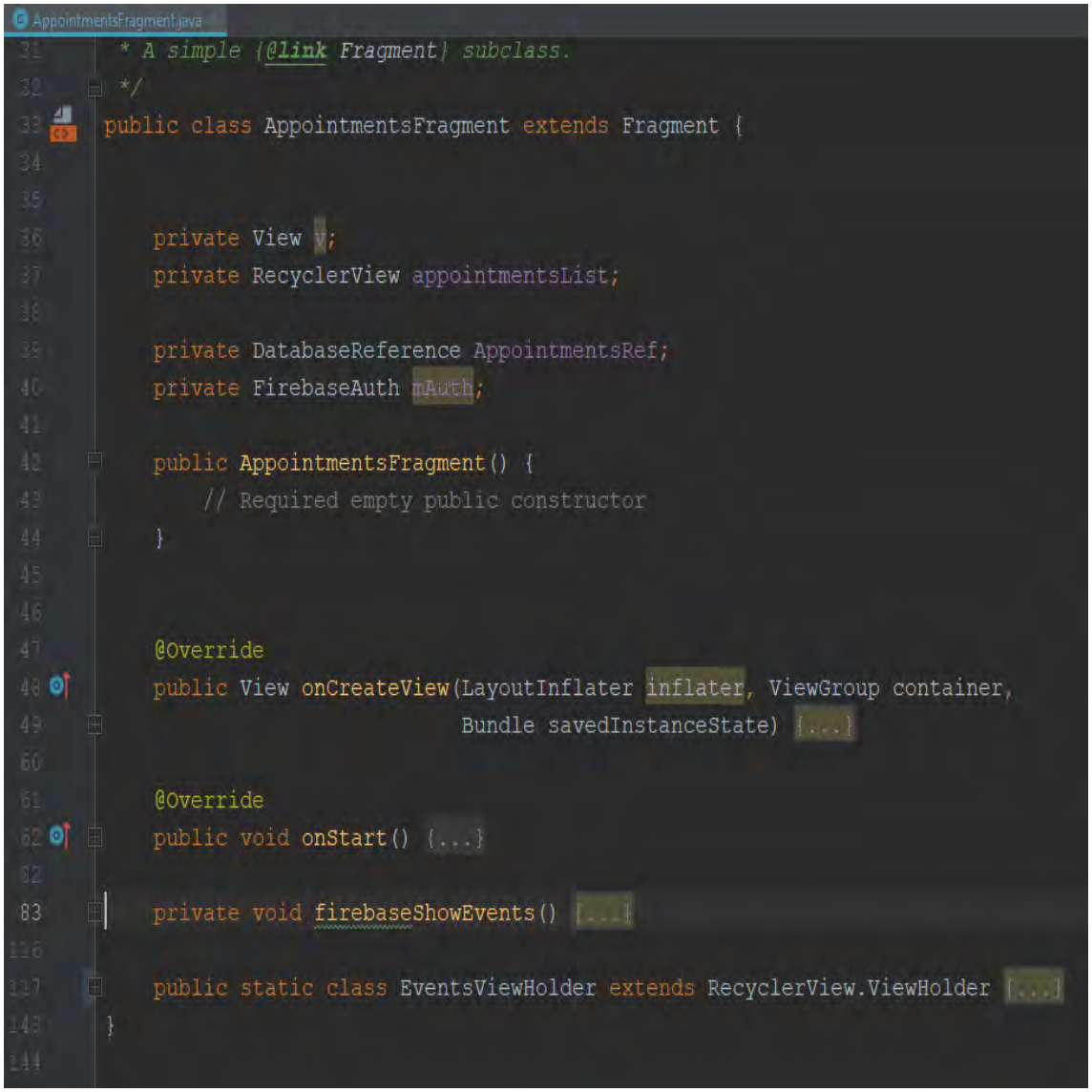
Σχήμα 4.23: Available Appointments, Notification

4.3.2 Appointments Fragment

Αφού ο Ιατρός επιβεβαιώσει το ραντεβού, ο ασθενής μπορεί να το βλέπει στο Appointments Fragment. Αυτό το Fragment έχει μέσω ενός RecyclerView όλα τα επιβεβαιωμένα ραντεβού του ασθενή. Το layout δεν διαφέρει πολύ από αυτό του Doctors Fragment.



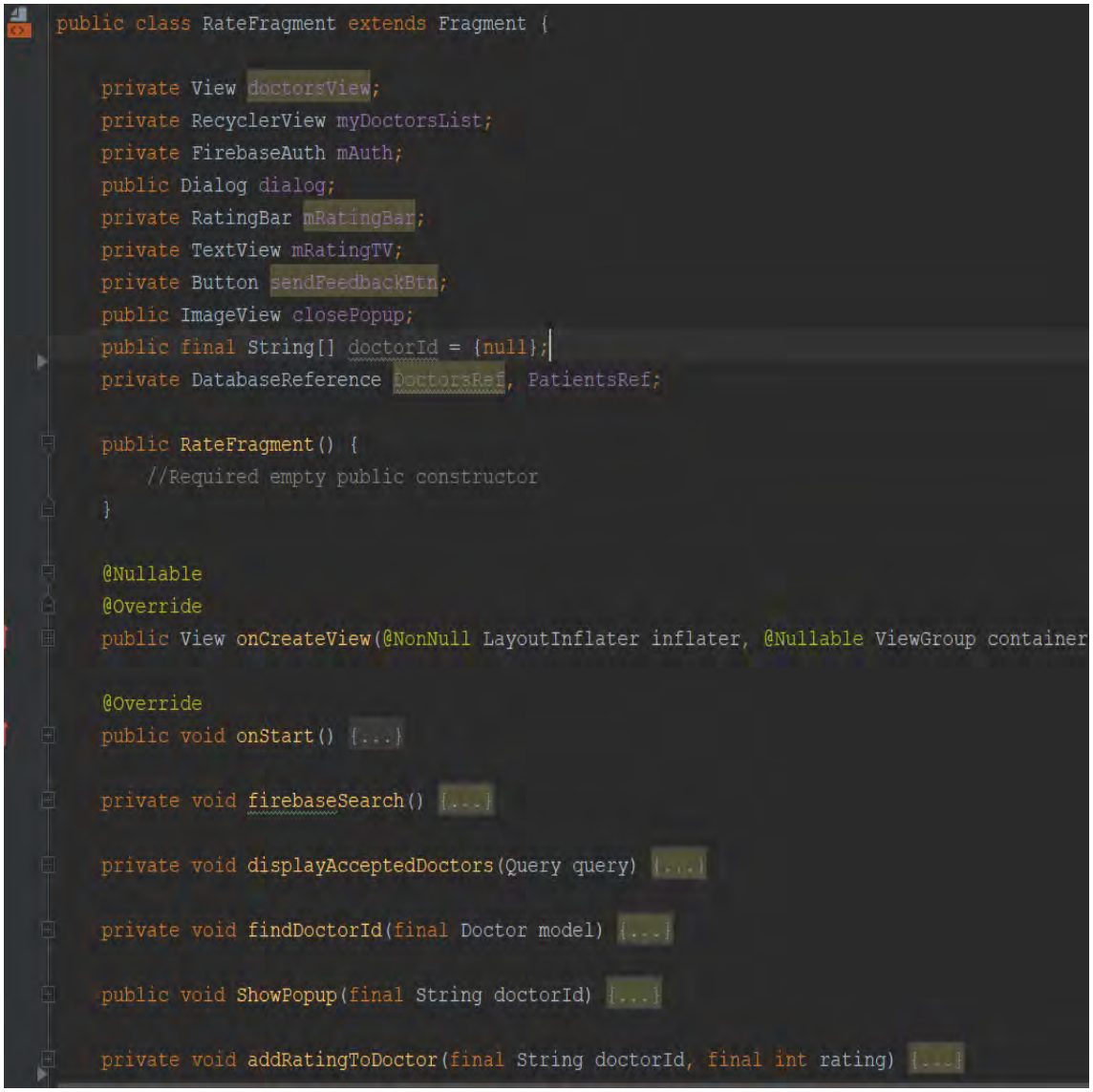
Σχήμα 4.24: Appointment Fragment Emulator



Σχήμα 4.25: Appointment Fragment Class

4.3.3 Rate Fragment

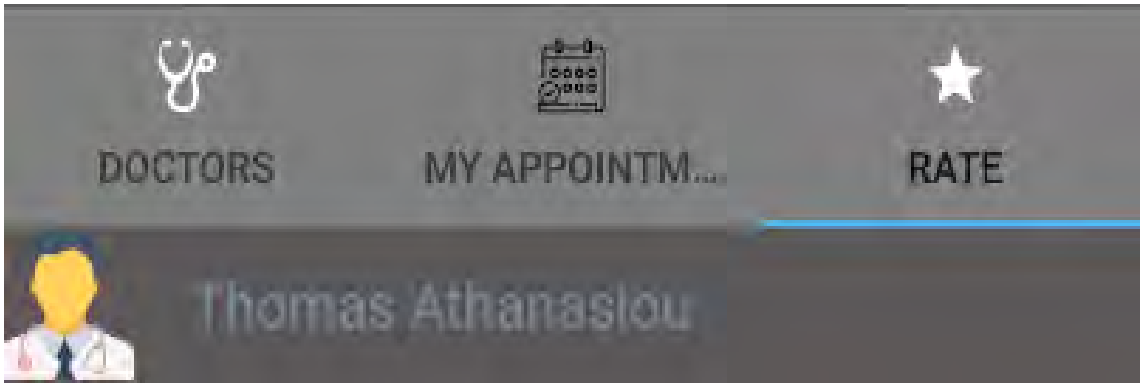
Μετά την επιβεβαίωση του ραντεβού και μετά το πέρας του, ο Ασθενής έχει την δυνατότητα να αξιολογήσει τον ιατρό.



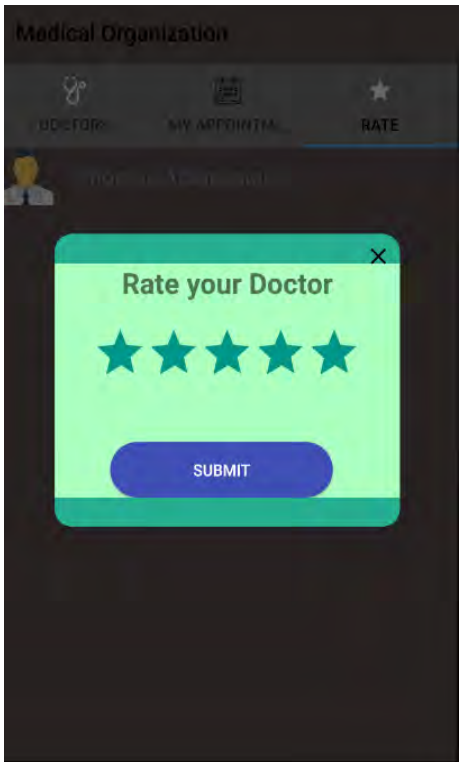
```
public class RateFragment extends Fragment {

    private View doctorsView;
    private RecyclerView myDoctorsList;
    private FirebaseAuth mAuth;
    public Dialog dialog;
    private RatingBar mRatingBar;
    private TextView mRatingTV;
    private Button sendFeedbackBtn;
    public ImageView closePopup;
    public final String[] doctorId = {null};
    private DatabaseReference DoctorsRef, PatientsRef;

    public RateFragment() {
        //Required empty public constructor
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container

    @Override
    public void onStart() {...}

    private void firebaseSearch() {...}

    private void displayAcceptedDoctors(Query query) {...}

    private void findDoctorId(final Doctor model) {...}

    public void ShowPopup(final String doctorId) {...}

    private void addRatingToDoctor(final String doctorId, final int rating) {...}
```

Σχήμα 4.26: Rate Fragment Class

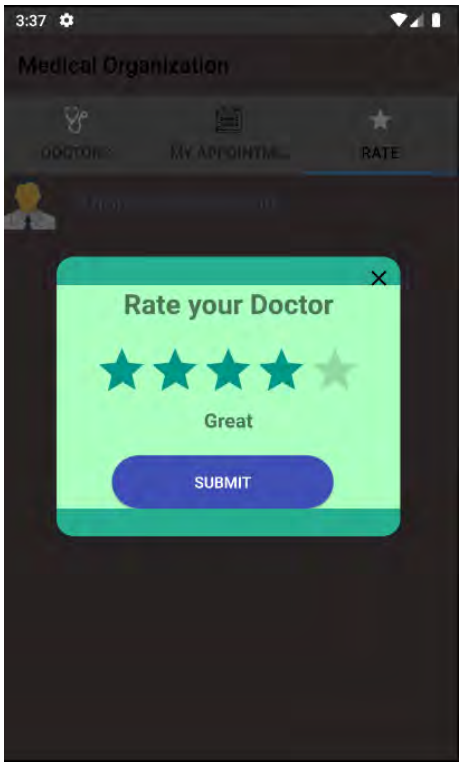
Όταν ο ασθενής επιλέξει τον Ιατρό που πρέπει να αξιολογήσει τότε εμφανίζεται ένας popup διάλογος και από εκεί διαλέγει αστέρια αξιολόγησης.



Σχήμα 4.27: Rate Fragment Emulator



(α΄) Rate Popup 1



(β΄) Rate Popup 2

Σχήμα 4.28: Rate Popup

4.4 Η Πλευρά του Ιατρού

Όπως και στην πλευρά του ασθενή όταν ένας χρήστης είναι Ιατρός, μόλις συνδεθεί στην εφαρμογή οδηγείται στην ProfileActivity. Εκεί έχει τα ίδια στοιχεία με τον ασθενή εκτός από ένα επιπλέον, το Rating. Το Rating ορίζεται με βάση το πόσοι ασθενείς έχουν βαθμολογήσει τον συγκεκριμένο Ιατρό. Ο Ιατρός έχει 4 επιλογές.

- Η πρώτη έχει να κάνει με το Profile του, όπου βλέπει τα στοιχεία του και μπορεί να επιβεβαιώσει το mail του.
- Η δεύτερη είναι για να δημιουργήσει ένα διαθέσιμο ραντεβού για τους ασθενείς.
- Η τρίτη είναι για να ελέγχει τις ειδοποιήσεις που λαμβάνει
- Η τέταρτη για να ελέγχει τα ραντεβού που έχει επιβεβαιώσει και να μπορεί να τα ακυρώσει.

Σχήμα 4.29: Doctor's Activity

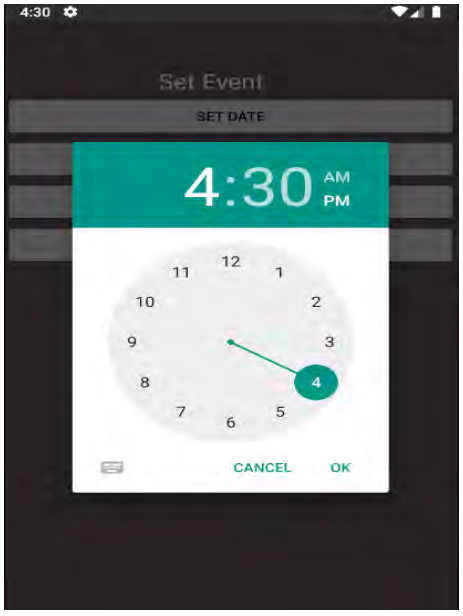
4.4.1 Set Event Activity

Για να δημιουργήσει ένα διαθέσιμο ραντεβού ο Ιατρός πρέπει να επιλέξει ημερομηνία, ώρα έναρξης, ώρα λήξης.

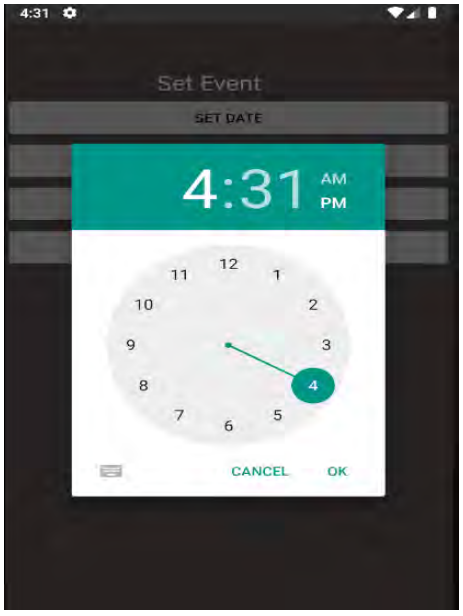
(α΄) Set Event Layout

(β΄) Set Date

Σχήμα 4.30: Set event, Set Date



(α΄) Set Starting Time



(β΄) Set Ending Time

Σχήμα 4.31: Set Time

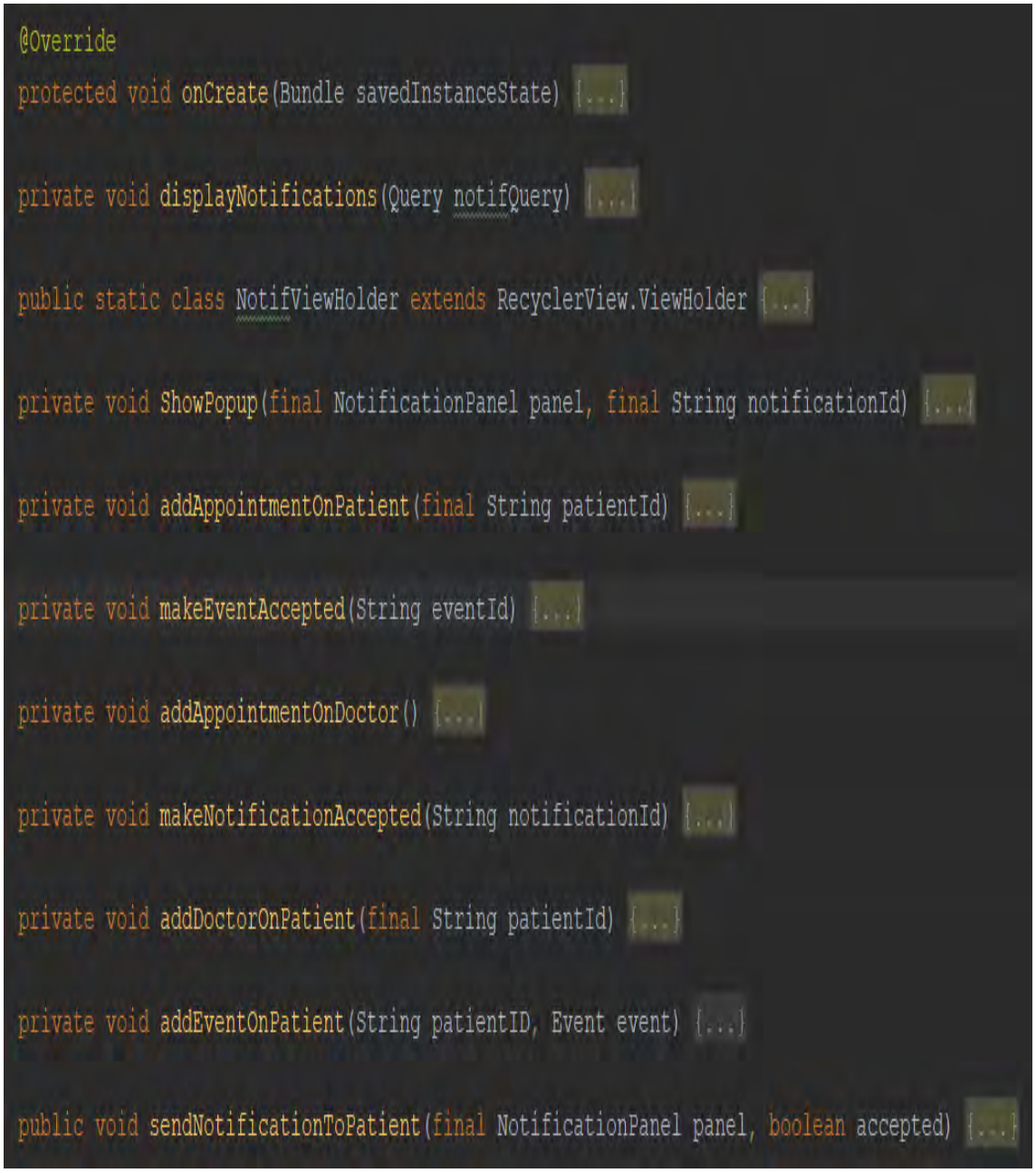
Μόλις τα απαραίτητα στοιχεία (Ημερομηνία, Ώρες) έχουν τεθεί, πατώντας στο Create Event δημιουργείται στην βάση ένα Αντικείμενο τύπου Event. Σε αυτό αποθηκεύονται στοιχεία όπως Ημερομηνία, Ώρα, όνομα Ιατρού, αλλά και μεταβλητές οι οποίες θα χρησιμοποιηθούν μόλις ο ασθενής επιλέξει το συγκεκριμένο event όπως όνομα και token ασθενή, καθώς και μια boolean μεταβλητή για το αν το συγκεκριμένο event έχει γίνει αποδεκτό από τον Ιατρό.

```
package com.example.medicalorganization.Models;

import java.util.Date;

public class Event {


    public java.util.Date startTime;
    public Date endTime;
    public java.util.Date date;
    public String doctorName;
    public String patientName;
    public String patientID;
    public String patientToken;
    public String doctorId;
    public boolean accepted;

    public Event() {}

    public Event(java.util.Date date, java.util.Date startTime, java.util.Date endTime, String doctorName, String patientName
                 String patientID, String patientToken,
                 String doctorId, boolean accepted) {

        this.date = date;
        this.startTime = startTime;
        this.endTime = endTime;
        this.doctorName = doctorName;
        this.patientName = patientName;
        this.patientID = patientID;
        this.patientToken = patientToken;
        this.doctorId = doctorId;
        this.accepted = accepted;
    }
}
```

Σχήμα 4.32: Event Object

4.4.2 Notifications Activity

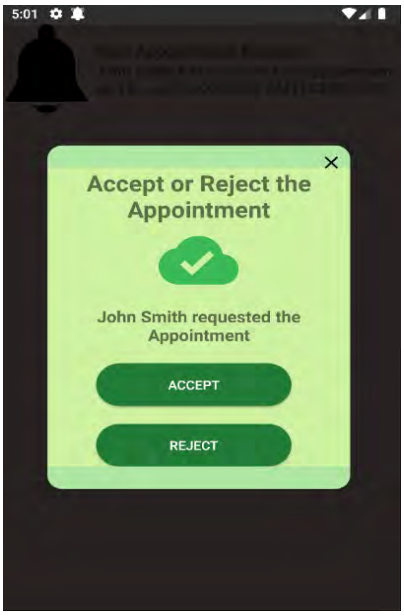
Αυτή η Activity έχει μια σειρά από μεθόδους-λειτουργίες που έχουν να κάνουν με το πως θα αποδεχθεί ένα αίτημα για ραντεβού από ασθενή και πως θα τον ενημερώσει. Αρχικά με RecyclerViewer φαίνονται όλες οι ειδοποιήσεις που έχει λάβει ο Ιατρός.



Σχήμα 4.33: Notifications Activity

Όταν κάνει click πάνω σε μία από αυτές εμφανίζεται ένας popup διάλογος για το αν αποδέχεται ή απορρίπτει την συγκεκριμένη συνάντηση.

(α΄) Notifications Activity on Emulator



(β΄) Notification Popup

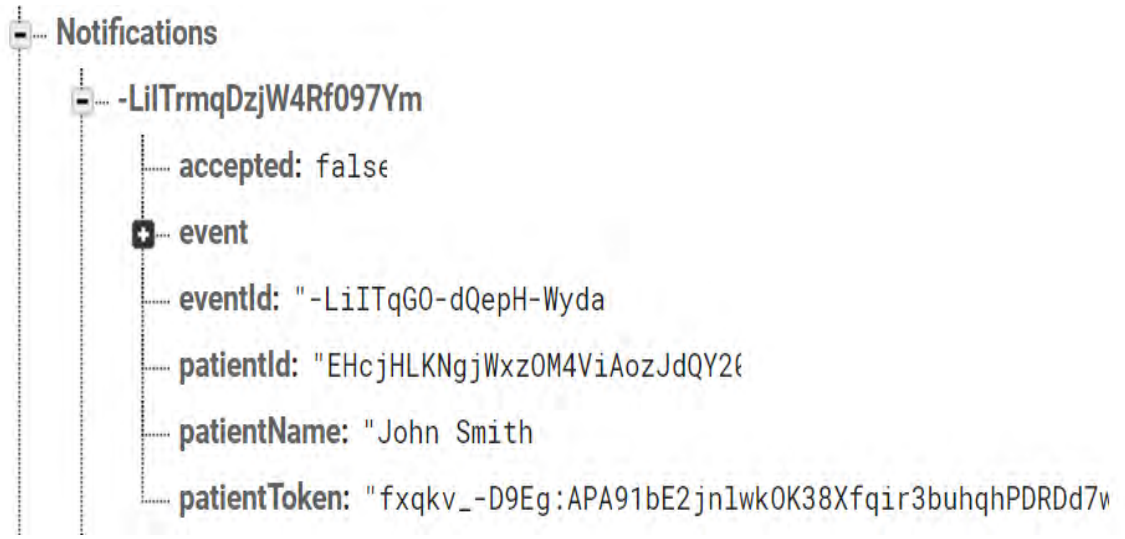
Σχήμα 4.34: Notification Activity and Popup Dialog

Αν ο Ιατρός επιλέξει Accept τότε γίνονται οι παρακάτω λειτουργίες:

- SendNotificationToPatient : Ο ασθενής λαμβάνει ειδοποίηση ότι το ραντεβού του έγινε αποδεκτό.
- AddEventOnPatient: Το αντικείμενο Event καταχωρείται και στην πλευρά του ασθενή.
- AddDoctorOnPatient: Ο Ιατρός καταχωρείται στην βάση που βρίσκεται ο ασθενής ώστε να μπορεί αργότερα να τον αξιολογήσει.
- MakeNotificationAccepted: Κάνει το αντικείμενο της συγκεκριμένης ειδοποίησης αποδεκτό ώστε να μην εμφανιστεί πάλι.
- AddAppointmentOnDoctor: Αυξάνει τον αριθμό των ραντεβού του Ιατρού.
- AddAppointmentOnPatient: Αυξάνει τον αριθμό των ραντεβού του Ασθενή.
- MakeEventAccepted: Κάνει το Event αποδεκτό και μη διαθέσιμο για άλλους ασθενείς.

Αν επιλέξει Reject τότε ο ασθενής λαμβάνει μία ειδοποίηση ότι το ραντεβού του απορρίφθηκε.

Σχήμα 4.35: Event Object on RealTime Database

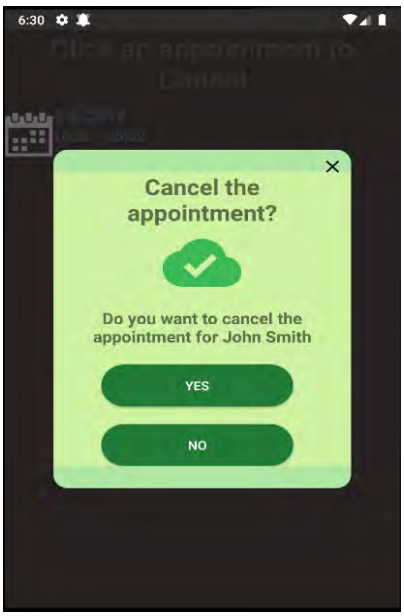


Σχήμα 4.36: Notification Object on RealTime Database

4.4.3 Manage Appointments Activity

Αυτή η Activity δείχνει τα επιβεβαιωμένα ραντεβού του ιατρού. Πατώντας πάνω σε ένα ραντεβού ο Ιατρός μπορεί να ακυρώσει το συγκεκριμένο γεγονός ενημερώνοντας τον ασθενή με μια ειδοποιήση και κάνοντας τις απαραίτητες αλλαγές στην βάση δεδομένων.

(α΄) Manage Appointments Activity Emulator



(β΄) Manage Appointments Popup

Σχήμα 4.37: Manage Appointments Activity and Popup Dialog

```
@Override
protected void onCreate(Bundle savedInstanceState) {...}
@Override
protected void onStart() { super.onStart(); }

private void showPopup(final Event model, final String eventId) {...}

private void makeEventNotAccepted(String eventId) {...}

private void sendNotificationToPatient(Event event) {...}

public static class EventViewHolder extends RecyclerView.ViewHolder{...}
```

Σχήμα 4.38: Manage Appointment Methods

Κεφάλαιο 5

Ανάλυση της Εφαρμογής

5.1 Προγραμματισμός Android με Firebase

5.1.1 Login

Κάθε φορά που ένας χρήστης κάνει Login πρέπει να υπάρχουν References στην βάση που δείχνουν την ιδιότητα του χρήστη(Ιατρός/Ασθενής). Επίσης πρέπει να υπάρχει ένα αντικείμενο Authorization του χρήστη που δηλώνει την ταυτότητά του με βάση το email και το password.

```
mFirebaseDatabase = FirebaseDatabase.getInstance();
mDoctorsDatabaseReference = mFirebaseDatabase.getReference().child("Doctors");
mPatientsDatabaseReference = mFirebaseDatabase.getReference().child("Patients");
mAuth = FirebaseAuth.getInstance();
```

Σχήμα 5.1: Firebase Login

Καλούμε την μέθοδο signInWithEmailAndPassword() της Firebase που ταυτοποιεί τον χρήστη παίρνοντας σαν παραμέτρους το Email και τον κωδικό του. Στην συνέχεια ψάχνουμε στην βάση και στην πλευρά των Ιατρών και στην πλευρά των ασθενών για το συγκεκριμένο Email προσθέτωντας την μέθοδο-Listener addListenerForSingleValueEvent() και στις 2 References. Οι References ανανεώνονται προσθέτοντας το μοναδικό Id κάθε χρήστη, το οποίο παίρνουμε καλώντας την .getCurrentUser().getUid(). Η συγκεκριμένη λειτουργία πραγματοποιείται για την ανανέωση του device_token του κάθε χρήστη σε περίπτωση που συνδεθεί από διαφορετική συσκευή. Για την ανανέωση του token αναλόγως την ιδιότητα του χρήστη καλούμε στο reference του συγκεκριμένου χρήστη την .setValue(). Μετά την επιτυχή ανανέωση του Token οδηγούμε τον χρήστη μέσω Intent στην Profile Activity. Αλλιώς μόνο η signInWithEmailAndPassword() είναι αρκετή για να συνδέσει έναν χρήστη στην εφαρμογή.

```
mAuth.signInWithEmailAndPassword(email.getText().toString().trim(), password.getText().toString().trim()).addOnComplete
        if (task.isSuccessful()){
            progressBar.setVisibility(View.GONE);
            //you have successfully logged
            final String currentUserId = mAuth.getCurrentUser().getUid();
            final String[] token = new String[1];
            //GET REGISTRATION TOKEN
            FirebaseInstanceId.getInstance().getInstanceId()
                    .addOnCompleteListener((task) -> {
                    if(task.isSuccessful()){
                        //get registration token
                        token[0] = task.getResult().getToken();
                    }
                    else{
                        Toast.makeText(getApplicationContext(), text: "Token not generated" + task.getException()
                    }
                });
```

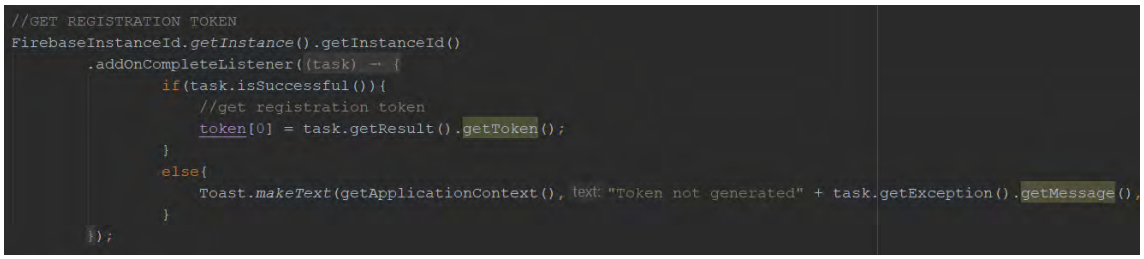
Σχήμα 5.2: Token Generation

```
//set the device tokedatabase if doctor
mDoctorsDatabaseReference.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        if (dataSnapshot.hasChild(currentUserId)){
            mDoctorsDatabaseReference.child(currentUserId).child("Device_Token")
                    .setValue(token[0])
                    .addOnCompleteListener((task) -> {
                            if(task.isSuccessful()){
                                finish();
                                Intent intent = new Intent(getApplicationContext(), DoctorActivity.class);
                                //clear all activities on the top of the stack, when back button is pressed
                                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                startActivity(intent);
                            }
                    });
        }
    }
```

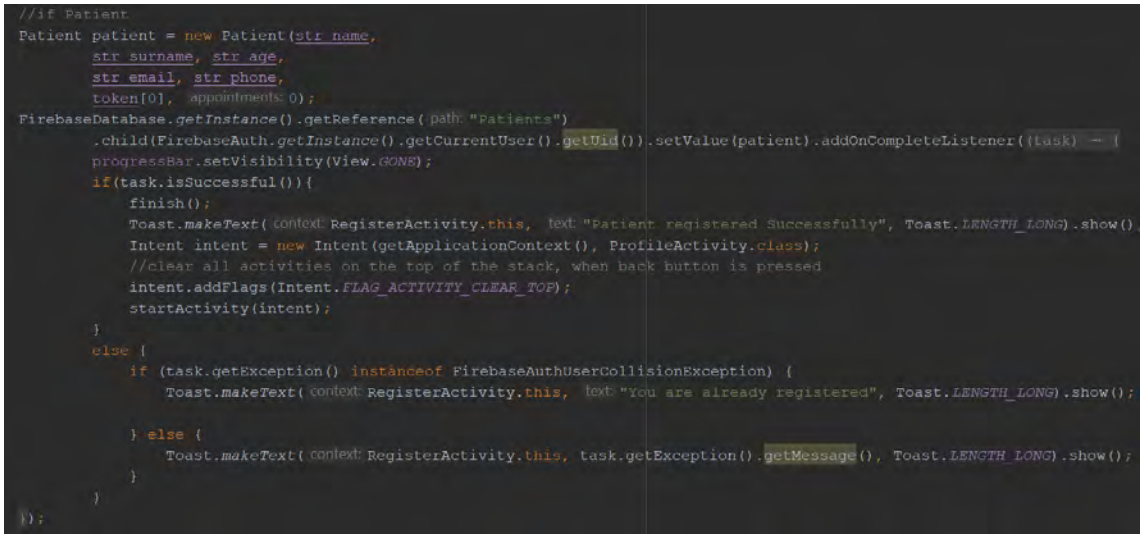
Σχήμα 5.3: Refresh Doctor Token

5.1.2 Register

Για να εγγράψουμε έναν χρήστη στην εφαρμογή κάνουμε την ίδια διαδικασία με το Login αλλά αυτή την φορά καλούμε την createUserWithEmailAndPassword() όπου παίρνει πάλι σαν παραμέτρους Email και Password και αναλόγως την ιδιότητα του χρήστη (Ιατρός/Ασθενής) τον προσθέτει στην βάση μαζί με τα υπόλοιπα στοιχεία του. Προφανώς ένα πολύ σημαντικό στοιχείο είναι πάλι το token. Μόλις ο χρήστης εγγραφεί επιτυχώς οδηγείται στην Profile Activity.



Σχήμα 5.4: Token Generation Register



Σχήμα 5.5: Register Patient

5.1.3 DoctorActivity και PatientActivity

Αναλόγως την ιδιότητα του χρήστη (Ιατρός/Ασθενής) στις λειτουργλιες των Activities ψάχνουμε τον χρήστη όπως στο Login, με την ανάλογη Reference και με το email του και ανοίγουμε αναλόγως τα νέα Activities ή Fragments που επιλέγει ο χρήστης. Για παράδειγμα στην OpenEventActivity() όταν ένας Ιατρός επιλέγει να δημιουργήσει ένα νέο event παιρνάμε στο νέο activity τα στοιχεία του Ιατρού. Στην συγκεκριμένη περίπτωση χρειάζονται σίγουρα το id του Ιατρού και το όνομά του.

```
public void openEventsActivity(View view) {
    final String email = mAuth.getCurrentUser().getEmail();
    mDoctorsReference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            for(DataSnapshot data : dataSnapshot.getChildren()){
                Doctor doctor = data.getValue(Doctor.class);
                if(doctor.Email.equals(email)){
                    startActivity(new Intent(getApplicationContext(), SetEventActivity.class));
                }
            }
        }

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

        }
    });
```

Σχήμα 5.6: Open Event Activity

Από την πλευρά του Ασθενή όταν ανοίγει ένα νέο Activity όπως openAppointmentsActivity πάλι ανατρέχουμε στο Reference των ασθενών ώστε να μπορούμε να πάρουμε τα δεδομένα του συγκεκριμένου ασθενή.

```
public void openAppointmentsActivity(View view) {
    final String email = mAuth.getCurrentUser().getEmail();
    mPatientsReference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            for(DataSnapshot data : dataSnapshot.getChildren()){
                Patient patient = data.getValue(Patient.class);
                if(patient.Email.equals(email)){
                    startActivity(new Intent(getApplicationContext(), AppointmentPatientActivity.class));
                }
            }
        }
```

Σχήμα 5.7: Open Appointments Activity

5.1.4 Fragments

Τα Fragments είναι σαν μικρά Activities ενσωματωμένα σε ένα Activity. Στην πλευρά του Ασθενή ανοίγωντας την AppointmentPatientActivity οδηγούμαστε σε 3 Fragments.

DoctorFragment

Το DoctorFragment είναι για να επιλέξει ο ασθενής έναν Ιατρό ώστε να κλείσει μια συνάντηση. Για να εμφανιστούν οι διαθέσιμοι Ιατροί στην οθόνη χρησιμοποιούμε ένα RecyclerView. Πατώντας πάνω σε έναν Ιατρό ο ασθενής μπορεί να διαλέξει ένα ραντεβού και στη συνέχεια να προχωρήσει. Επίσης υπάρχει ένα SearchBar στο οποίο ο ασθενής μπορεί να ψάξει ιατρό με βάση το όνομά του. Για τη σωστή λειτουργία του SearchBar πρέπει οι Ιατροί στο recyclerView να ανανεώνονται σε κάθε αλλαγή του κειμένου που πληκτρολογεί ο ασθενής. Για αυτόν τον σκοπό καλούμε μέσα στην onQueryTextChange() ένα FirebaseSearch που ψάχνει αναλόγως το κείμενο που δέχεται κάθε φορά τον Ιατρό.

```
firebaseSearch("");
if (searchBar != null) {
    searchBar.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String query) { return false; }

        @Override
        public boolean onQueryTextChange(String newText) {
            firebaseSearch(newText);
            return true;
        }
    });
}
```

Σχήμα 5.8: Search Bar

Σχήμα 5.9: Firebase Search

AppointmentsFragment

Στο AppointmentsFragment υπάρχει ένα RecyclerView που αντλεί από την βάση τα επιβεβαιωμένα ραντεβού που έχει ο ασθενής.

RateFragment

Στο RateFragment υπάρχει ένα RecyclerView που αντλεί τους Ιατρούς με τους οποίους ο ασθενής έχει ήδη κλείσει μια συνάντηση. Πατώντας στον Ιατρό ο ασθενής μπορεί να αξιολογήσει. Αυτό γίνεται με έναν popup Dialog στον οποίο ο ασθενής μπορεί να επιλέξει αστέρια. Στην συνέχεια αφού πατήσει Submit καλείται η addRatingToDoctor() με παραμέτρους το id του ιατρού και την αξιολόγηση σασν ακέραιο. Στην addRatingToDoctor() δημιουργείται ένα αντικείμενο τύπου Rating με πεδία την αξιολόγηση και το όνομα του ασθενή, και στη συνέχεια με ένα Reference στο συγκεκριμένο Ιατρό προστίθεται η αξιολόγηση στον Ιατρό.

Σχήμα 5.10: Rating

```
final String[] patientName = new String[1];
PatientsRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        for(DataSnapshot data : dataSnapshot.getChildren()){
            Patient patient = data.getValue(Patient.class);
            if(patient.Email.equals(mAuth.getCurrentUser().getEmail())) {
                patientName[0] = patient.Name + " " + patient.Surname;
                Rating rate = new Rating(rating, patientName[0]);
                DatabaseReference reference = FirebaseDatabase.getInstance().getReference().child("Doctors")
                        .child(doctorId)
                        .child("Ratings");
                reference.push().setValue(rate);
            }
        }
    }
```

Σχήμα 5.11: Add Rating to Doctor

5.2 Web Api

Για να στέλνονται ειδοποιήσεις στους χρήστες και σε διαφορετικές συσκευές πρέπει να υλοποιηθεί ένα web api και να γίνει deploy στις συναρτήσεις της Firebase. Όταν γίνει αυτό και αφού ελέγξουμε το αν λειτουργεί σωστά μέσω του Postman όπως αναλύθηκε σε προηγούμενο κεφάλαιο, πρέπει να το χρησιμοποιήσουμε και μέσω του πηγαίου κώδικα της εφαρμογής. Αυτό γίνεται με τα εξής βήματα:

- Φτιάχνουμε ένα interface που υλοποιεί το HTTP Post request που χρειαζόμαστε για την αποστολή και παραλαβή των ειδοποιήσεων. Σαν πεδία του είναι φυσικά το token της συσκευής που αποστέλλεται η ειδοποίηση αλλά και ένας τίτλος και ένα σώμα.

```
public interface Api {

    @FormUrlEncoded
    @POST("send")
    Call<ResponseBody> sendNotification(
            @Field("token") String token,
            @Field("title") String title,
            @Field("body") String body
    );
}
```

Σχήμα 5.12: Api Interface

- Για να χρησιμοποιήσουμε το interface πρέπει να κατασκευάσουμε ανάλογα ένα αντικείμενο Retrofit. Το retrofit είναι ένα πολύ καλό εργαλείο για να χειριζόμαστε REST APIs και διάφορα requests όπως get, post, patch κ.α. κάτι που στην περίπτωση της εφαρμογής μας είναι απαραίτητο. Φτιάχνοντας το Retrofit για κλήσεις σε Api περνάμε παραμέτρους το url του web api και έναν GSON converter. Έπειτα για να χρησιμοποιήσουμε το interface καλούμε το αντικείμενο του retrofit με βάση το interface Api που δημιουσργήσαμε. Τέλος η κλήση γίνεται καλώντας την sendNotification με τις ανάλογες παραμέτρους.

```
final Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("https://medicalorganization-7b35a.firebaseapp.com/api1/")
        .addConverterFactory(GsonConverterFactory.create())
        .build();
Api api = retrofit.create(Api.class);
Call<ResponseBody> call = api.sendNotification(panel.patientToken, response,  body: "");

call.enqueue(new Callback<ResponseBody>() {
    @Override
    public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {

        Toast.makeText(getApplicationContext(),  text: "Notification sent to the Patient", Toast.LENGTH_LONG).show();
    }
```

Σχήμα 5.13: Retrofit και Send Notification

Κεφάλαιο 6

Επίλογος και Συμπεράσματα

6.1 Πηγαίος Κώδικας και οδηγίες Εγκατάστασης

Ο πηγαίος κώδικας της εφαρμογής βρίσκεται στο Github στο `https://github.com/thathana1521/MedicalOrganization`. Εδώ υπάρχουν πολλά commits που εξηγούν τι πρόσθετο έβαζα κάθε φορά καθώς και τα προβλήματα που έλυνα.
Τα αρχεία της πλευράς του Server για την υλοποίηση του API βρίσκονται στο `https://github.com/thathana1521/ServerFilesForMedicalAppointmentOrganization`.

Για να εγκαταστήσετε την εφαρμογή στην συσκευή Android σας κατεβάστε το .apk αρχείο από το Github του πηγαίου κώδικα.

Για να τρέξετε την εφαρμογή στον υπολογιστή σας να την τροποποιήσετε ή να την εξελίξετε, πρέπει να κάνετε τα εξής βήματα:

- Κατεβάστε τον κώδικα από τα links του Github.
- Απαραίτητες προϋποθέσεις είναι να έχετε την τελευταία έκδοση του Android Studio και έναν Google Account.
- Από το Android Studio File-> Open Project κάνετε browsing στον φάκελο που κατεβάσατε την εφαρμογή και πατάτε Open.
- Όλες οι εξαρτήσεις που απαιτούνται για την Firebase είναι ήδη εγκατεστημένες στην εφαρμογή.
- Αν θέλετε να δημιουργήσετε νέα εφαρμογή με δικά σας credentials και να έχετε πρόσβαση στο Firebase Console πρέπει να δημιουργήσετε ένα νέο Android Studio Project(File-> New-> New Project...), να αντιγράψετε τα αρχεία που χρειάζεστε και στη συνέχεια να ακολουθήσετε τις επίσημες οδηγίες της Firebase στο `https://firebase.google.com/docs/` και από εκεί Get started with Firebase -> Add Firebase to app -> Android.

- Για το deployment του API πρέπει να ακολουθήσετε τις οδηγίες από `https://firebase.google.com/docs/cloud-messaging` και να κάνετε τις απαραίτητες αλλαγές σε SecretID και ServerKey όπως αναφέρονται στον επίσημο οδηγό της Firebase.

- Τέλος πρέπει στην πλευρά του πηγαίου κώδικα να αλλάξετε το link του API. Αντικαταστήστε το "https://medicalorganization-7b35a.firebaseapp.com/api1/" με το link που θα έχετε από το Firebase Console.

6.2 Επίλογος

Κατά την υλοποίηση της συγκεκριμένης εφαρμογής παρατηρήθηκαν πολλές από τις δυνατότητες του Firebase. Πρόκειται για ένα πολύ σημαντικό και σύγχρονο εργαλείο προσφέροντας λειτουργίες όπως Authentication και RealTime Database που εφαρμόζονται σχετικά εύκολα. Το documentation τους είναι αρκετά αναλυτικό και χρήσιμο και σε συνδυασμό με τα επίσημα tutorials τους δίνουν τη δυνατότητα στους χρήστες να μάθουν εύκολα μια βάση δεδομένων NoSQL και πως να την χρησιμοποιούν.
Το μόνο μειωνέκτημα της Firebase είναι ότι έχει όριο στα transactions, κάτι που στην συγκεκριμένη εφαρμογή δεν αποτέλεσε πρόβλημα διότι δεν υπήρχαν πολύπλοκα Queries.
Η συγκεκριμένη εφαρμογή υλοποιεί ένα απλό σύστημα κρατήσεων, αλλά δεν λύνει παραπάνω προβλήματα από αυτά που έχουν ήδη λυθεί από παρόμοιες εφαρμογές που είναι ήδη στην παραγωγή.

6.3 Συμπεράσματα και Επεκτάσεις

Σαν συνέχεια της υλοποίησης της εφαρμογής ο πρώτος στόχος θα ήταν ένα καλύτερο Interface για την επιλογή των ραντεβού από τους Ιατρούς. Μια εμφάνιση με ένα Calendar View θα ήταν πιο φιλική και πιο άμεση στον χρήστη. Στην συνέχεια θα μπορούσαν να προσθεθούν επιπλέον δεδομένα για τους χρήστες. Δηλαδή, στην πλευρά του Ιατρού πρέπει να υπάρχει ειδίκευση, τοποθεσία Ιατρείου, Φωτογραφίες Ιατρείου, βιογραφικό. Από την πλευρά του ασθενή πρέπει να υπάρχει ιστορικό, φαρμακευτικές αγωγές, τοποθεσία. Επίσης ένα σύστημα πληρωμών είτε για επισκέψεις είτε για φάρμακα θα ήταν πολύ βοηθητικό.

Βιβλιογραφία

[1] Anu Gupta Abhinav Kathuria1. Challenges in Android Application Development: A Case Study. *International Journal of Computer Science and Mobile Computing*, 4(5):6, 2015.

[2] Lisane B. de Brisolara Abilio G. Parada. A model driven approach for Android applications development. *Symposium on Computing System Engineering (SBESC)*, σελίδα 6, 2012.

[3] Android. `https://www.android.com/`.

[4] Android Emulator. `https://developer.android.com/studio/run/emulator`.

[5] Android History and Versions. `https://en.wikipedia.org/wiki/Android_version_history`.

[6] Android Logo. `https://www.android.com/intl/en_hk/`.

[7] Android SDK. `https://www.techopedia.com/definition/4220/android-sdk`.

[8] Android Studio. `https://en.wikipedia.org/wiki/Android_Studio`.

[9] Book my Doctor. `https://codecanyon.net/item/book-my-doctor/13150333`.

[10] Ed Burnette. *Hello, Android: Introducing Google's Mobile Development Platform*. Pragmatic Bookshelf; Third edition, August 7, 2010.

[11] Neil Castellino. Bon Voyage Hotel Booking System. `https://github.com/NeilCastellino/Bon_Voyage`.

[12] Nimesh Chhetri. A Comparative Analysis of Node.js (Server-Side JavaScript). `https://repository.stcloudstate.edu/cgi/viewcontent.cgi?referer=https://www.google.com/&httpsredir=1&article=1004&context=csit_etds`, 2016.

[13] DoctorAnytime Android. `https://play.google.com/store/apps/details?id=doctoranytime.health.fitness.doc.app&hl=en`.

[14] DoctorAnytime Web. https://www.doctoranytime.gr/en.

[15] Dr. Pad. https://play.google.com/store/apps/details?id=us.drpad.drpadapp&hl=en_US.

[16] Wenjie Xu Dunlu Peng, Lidong Cao. Using JSON for Data Exchanging in Web Service Applications. *Journal of Computational Information Systems*, 16:5883–5890, 2011.

[17] Firebase Activities. https://www.tutorialspoint.com/android/android_acitivities.html.

[18] Firebase Activities. https://developer.android.com/reference/android/app/Activityl.

[19] Firebase API for Notifications. https://www.freecodecamp.org/news/how-to-add-push-notifications-to-a-web-app-with-firebase-528a702e13e1/.

[20] Firebase Authentication. https://firebase.google.com/docs/auth/.

[21] Firebase Realtime Database. https://savvyapps.com/blog/firebase-realtime-database-vs-cloud-firestore-for-your-app.

[22] Firebase Realtime Database. https://abhiandroid.com/ui/xml.

[23] Firebase Users. https://firebase.google.com/docs/auth/users.

[24] Pablo López Gabriela Martínez. ADVANCED DATABASES PROJECT: REAL-TIME DATABASES AND FIREBASE. http://cs.ulb.ac.be/public/_media/teaching/infoh415/student_projects/2019/firebase.pdf, 2018.

[25] Andri Heryandi. Developing Application Programming Interface (API) for Student Academic Activity Monitoring using Firebase Cloud Messaging (FCM). *OP Conference Series Materials Science and Engineering*, 407:7, 2018.

[26] Chunnu Khawas. Application of Firebase in Android App Development-A Studys. *International Journal of Computer Applications*, 179(46):49–53, 2018.

[27] Ameya Parker Mithilesh Tarkar. APIs and Restful APIs. *International Journal of Trend in Scientific Research and Development (IJTSRD)*, 2(5):319–322, 2018.

[28] Yusmadi Yah Jusoh Rohayanti Hassan Ayu Alyani Noorsyahira Ismail, Shahreen Kasim1. MEDICAL APPOINTMENT APPLICATION. *Acta Electronica Malaysia*, σελίδα 9, 2017.

[29] NowDoctor. https://www.nowdoctor.gr/.

[30] May H. Riadh. Notification System to Students using an Android Application. *International Journal of Computer Applications*, 130(1):22–27, 2016.

[31] Daiki Chiba Ryo Sato και Shigeki Goto. Detecting Android Malware by Analyzing Manifest Files. *Proceedings of the Asia-Pacific Advanced Network*, 36:23–31, 2013.

[32] Dr. Tejinder Singh. A Survey on Java Programming Language and Methods of Improvisation. *International Journal of Innovations & Advancement in Computer Science*, 6(12):158–162, 2017.

[33] Kawalpreet Singh. Research Paper on C Language. *INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN TECHNOLOGY*, 2(6):7, 2015.

[34] Mahima M Katti Suhas Holla. ANDROID BASED MOBILE APPLICATION DEVELOPMENT and its SECURITY. *International Journal of Computer Trends and Technology*, 3(3):5, 2012.

[35] Alaa Qaffas Trevor Barker. Online Appointment Management System. *https://www.researchgate.net/publication/*266171377*_Online_Appointment_Management_System*, σελίδα 5, 2012.

[36] Meenu Dave Vatika Sharma. SQL and NoSQL Databases. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(8):20–27, 2012.

[37] Şenay Kocakoyun. Developing of Android Mobile Application Using Java and Eclipse: An Application. *INTERNATIONAL JOURNAL OF ELECTRONICS, MECHANICAL AND MECHATRONICS ENGINEERING*, 7(1):1335–1354, 2017.

Συντομογραφίες

βλπ	βλέπε
κ.λπ.	και λοιπά
κ.ο.κ	και ούτω καθεξής