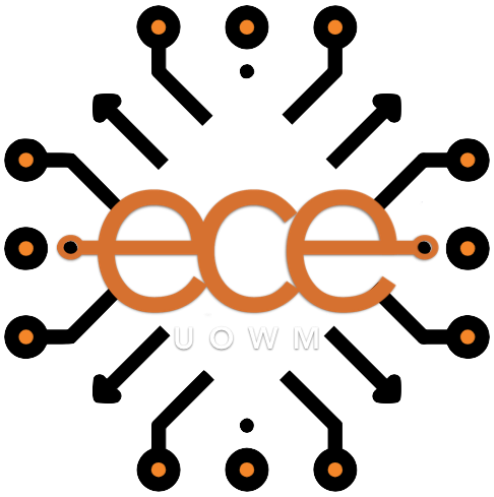


ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΤΜΗΜΑ
ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη Εφαρμογής Διαχείρισης Ζώων Συντροφιάς»

Δήλωση Πνευματικών Δικαιωμάτων

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο“ Ανάπτυξη Εφαρμογής Διαχείρισης Ζώων Συντροφιάς ” καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών πρώην Μηχανικών Πληροφορικής και Τηλεπικοινωνιών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. Παντελή Αγγελίδη αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.

Υπογραφή Φοιτητή:

Γενοβέφα-Φωτεινή Κωνσταντάκου

Ευχαριστίες

«Στους γονείς μου»

Κατάλογος εικόνων

Πρόλογος

Στο σύγχρονο τρόπο ζωής τα κατοικίδια ζώα όλο και περισσότερο μπαίνουν στην καθημερινότητα μας. Η ανάγκη για σωστή φροντίδα και ιατρική περίθαλψη τους όλο και περισσότερο απαιτεί την δική μας ενεργή συμμετοχή. Στο σημείο αυτό η χρήση νέων τεχνολογιών αναδεικνύεται ως ισχυρό εργαλείο, για την ορθή ενημέρωση των ιδιοκτητών των κατοικιδίων για τα διάφορα κτηνιατρικά προβλήματα που εμφανίζονται.

Η παρούσα εργασία συνδυάζει την πρωτοποριακή τεχνολογία των Android εφαρμογών με την κτηνιατρική επιστήμη ώστε να έχουμε καλύτερα αποτελέσματα στην εφαρμογή των μεθόδων θεραπείας.

Επιπλέον, η χρήση ατομικού φακέλου για κάθε κατοικίδιο και η παρακολούθηση της υγείας τους βοηθάει να αποφύγουμε τα όποια μελλοντικά προβλήματα πρόκειται να δημιουργηθούν μέσα στην οικογένεια μας.

Περίληψη

Η παρούσα πτυχιακή εργασία αναφέρεται στο σχεδιασμό και στην ανάπτυξη εφαρμογής για κινητές συσκευές, με σκοπό τη βελτίωση της διαχείρισης ενός κτηνιατρείου με χρήση των νέων τεχνολογιών. Η εφαρμογή από τη μεριά των πελατών επιταχύνει και βελτιώνει της διαδικασία της αναφοράς του προβλήματος και βοηθάει στην καλύτερη εξυπηρέτηση τω πελατών, ώστε να έχουν μια καλύτερη εικόνα για την πορεία της θεραπείας . Επίσης, από τη μεριά των πελατών, μπορούν άμεσα να έρθουν σε επαφή με τον κτηνίατρο. Η εφαρμογή διαθέτει μια τοπική βάση όπου αποθηκεύονται οι χρήστες της εφαρμογής και κρατάμε μια τοπική και μια απομακρυσμένη βάση για τα διαθέσιμα ραντεβού που έχει το κτηνιατρείο ανά ημέρα, αλλά επιπλέον διαθέτει και τα κατάλληλα εργαλεία για να μπορεί να κατευθύνει τους πελάτες να έχουν άμεση ενημέρωση για την πορεία της θεραπείας

Παρουσιάζονται και συγκρίνονται οι αντίστοιχες εφαρμογές που εξυπηρετούν τον ίδιο σκοπό.

Τέλος, γίνεται μια ανάλυση της δομής της εφαρμογής και των τεχνολογιών που χρησιμοποιήθηκαν κατά την διάρκεια της ανάπτυξης.

Summary

This thesis focuses on the design and development of a mobile application aimed at improving the management of a veterinary clinic through the use of new technologies. From the clients' perspective, the application expedites and enhances the problem reporting process, contributing to better customer service and providing clients with a clearer picture of the treatment progress. Additionally, clients can directly contact the veterinarian through the application. The application maintains a local database for its

users and keeps both local and remote databases for available appointments at the clinic on a daily basis. Moreover, it provides tools to guide clients for immediate updates on the treatment progress.

The thesis also presents and compares similar applications serving the same purpose. Finally, an analysis of the application's structure and the technologies utilized during development is conducted.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1: ΕΞΥΠΝΑ ΚΙΝΗΤΑ ΤΗΛΕΦΩΝΑ ΚΑΙ ΕΦΑΡΜΟΓΕΣ

1.1 Εισαγωγή

Σκοπός αυτής της πτυχιακής εργασίας είναι η ανάπτυξη μιας κινητής εφαρμογής για την βελτίωση των λειτουργικών αναγκών ενός κτηνιατρείου. Η εφαρμογή μας σκοπεύει την διευκόλυνση των καθημερινών αναγκών του ενός κτηνιατρείου αλλά τη καθοδήγηση των ιδιοκτητών των κατοικιδίων στην διαδικασία κρατήσεων και στην θεραπεία, προσφέροντας τις δυνατότητες των νέων τεχνολογιών. Η εφαρμογή μας αναπτύχθηκε για το λειτουργικό σύστημα Android.

Οι εφαρμογή αποτελείται από δύο μέρη, με το πρώτο μέρος να περιέχει τις λειτουργίες που αφορούν τον πελάτη, όπως κλείσιμο ραντεβού, 24-ωρη καθοδήγηση από έμπειρ προσωπικό για την καλύτερη και αποτελεσματικότερη εφαρμογή της θεραπείας στα κατοικίδια. Το δεύτερο μέρος περιέχει τις λειτουργίες που αφορούν το κτηνιατρείο, όπως τη λίστα με τα διαθέσιμα ραντεβού της ημέρας ή όπως το ιστορικό για κάθε κατοικίδιο που αφορά το κτηνιατρείο μας.

1.2 Η μεθοδολογία

Ο πρωταρχικός σκοπός της εφαρμογής μας είναι να διευκολύνει την διαδικασία της θεραπείας των κατοικιδίων αλλά και να γίνει καλύτερη η ενημέρωση και καθοδήγηση των ιδιοκτητών κατοικιδίων έτσι ώστε να έχουμε μια βέλτιστη επικοινωνία μεταξύ των πελατών και των κτηνιάτρων . Βοηθάει τους κτηνιάτρους να έχουν μια καλύτερη εικόνα για την πορεία της θεραπείας , να μπορούν να καθοδηγούν και να συμβουλεύουν τους πελάτες για μια αποτελεσματικότερη περίοδο θεραπείας των κατοικιδίων τους.

Παράλληλα θα γίνει έρευνα και μελέτη των παρόμοιων εφαρμογών που υπάρχουν παγκοσμίως οι οποίες εκπληρούν τους ίδιους στόχους και θα προστεθούν επιπλέον λειτουργίες που θα διακρίνουν την εφαρμογή μας από τις υπόλοιπες. Κατά την ανάπτυξη της εφαρμογής θα γίνουν αρκετές βελτιώσεις διότι μπορεί να προκύπτουν καινούριες ανάγκες και να δοθούν καινούριες λύσεις με σκοπό τη βελτίωση των αρχικών σχεδίων. Μετά την ολοκλήρωση της εφαρμογής, αφού ξεπεράστηκαν κάποια προβλήματα, έγινε η αξιολόγηση του συστήματος. Στην αξιολόγηση αναδεικνύονται τα δυνατά όσο και τα αδύναμα σημεία της εφαρμογής μας. Τέλος, ο στόχος μας μετά την αξιολόγηση να προκύψουν παρατηρήσεις και να προετοιμαστεί το έδαφος για μελλοντικές επεκτάσεις.

1.3 Η Δομή της πτυχιακής

Η δομή της πτυχιακής μας δείχνει την διαδικασία και τα στάδια που ακολουθήσαμε κατά την διάρκεια της δημιουργίας της πτυχιακής

- Στο **πρώτο κεφάλαιο** κάνουμε μια εισαγωγή αναφέροντας τον σκοπό της εφαρμογής, ποια συνεισφορά θα έχει στην εκπαιδευτική διαδικασία και ποιες είναι οι δυσκολίες που αντιμετωπίσαμε. Αναλύουμε το τι είναι το λειτουργικό σύστημα Android, κάνουμε μια γενική επισκόπηση και ιστορική αναδρομή των εφαρμογών για έξυπνες συσκευές και κινητά τηλέφωνα (smartphones) και κάνουμε μια αναφορά στο τεχνολογικό υπόβαθρο του λειτουργικού συστήματος.

- Στο **δεύτερο κεφάλαιο** θα αναφερθούμε στις τεχνολογίες βάσεις δεδομένων, θα κάνουμε μια γενική επεξήγηση για το IDE (Android studio) και θα εξηγήσουμε τα πλεονεκτήματα της γλώσσας Java.

- Στο **τρίτο κεφάλαιο** γίνεται η ανάλυση των τεχνολογιών που χρησιμοποιούνται στις κινητές εφαρμογές για τις βάσεις δεδομένων, σύγκριση των διαθέσιμων τεχνολογιών και ο λόγος που επιλέξαμε τη συγκεκριμένη τεχνολογία για την εφαρμογή μας. Επίσης, αναφερόμαστε στις υπηρεσίες που χρησιμοποιούνται για την ταυτοποίηση των χρηστών και προσπέλασης των δεδομένων από έναν διακομιστή.

- Στο **τέταρτο κεφάλαιο** γίνεται η παρουσίαση των βασικών λειτουργιών της εφαρμογής μας με λεπτομέρεια και παρουσιάζεται η τελική μορφή της εφαρμογής. Παρουσιάζονται κάποια στιγμιότυπα οθόνης (screenshots) που πιστοποιούν τη σωστή λειτουργία της εφαρμογής μας. Γίνεται αναλυτική επεξήγηση των σταδίων ανάπτυξης της εφαρμογής. Θα εξηγήσουμε την διαδικασία της εγκατάστασης.

- Στο **πέμπτο κεφάλαιο** θα γίνει επεξήγηση του κώδικα και αναφορά στα βασικά σημεία του ενδιαφέροντος.

- Στο **έκτο κεφάλαιο** θα γίνει μια αναφορά στις αδυναμίες που έχει η εφαρμογή μας και θα αναφερθούμε στις μελλοντικές επεκτάσεις και ιδέες που θα βοηθήσουν στην αποτελεσματικότερη λειτουργία της εφαρμογής μας

1.2.1 Ιστορική αναδρομή στις κινητές συσκευές

Ας ξεκινήσουμε την ιστορική αναδρομή μας με τον Μαρτίν Κούπερ της Motorola, όταν πριν πολλά χρόνια, σε μια αμερικανική μεγαλούπολη κρατούσε στα δύο χέρια του μια συσκευή που έμοιαζε με φορητό ασύρματο. Είχε βάρος 900 γραμμάρια και ύψος 25 εκατοστά. Ήταν το πρώτο σύγχρονο κινητό τηλέφωνο με τον κωδικό Motorola DynaTAC.

Figure 1.2:1-First mobile phone

Τα κινητά τηλέφωνα εκείνης της εποχής δεν έμοιαζαν και πολύ με τα σημερινά έξυπνα κινητά τηλέφωνα. Δεν είχαν ούτε κάμερες για λήψη βίντεο ή φωτογραφιών, ούτε είχαν μικροεπεξεργαστές για διάφορες λειτουργίες που εκτελούν σήμερα. Για να είμαστε πιο ακριβείς, λόγω του μεγέθους τους ήτανε δύσκολο να χαρακτηριστούν ιδιαίτερα ως ‘κινητά’.

Όμως εξαιτίας των αυξημένων απαιτήσεων των χρηστών των κινητών τηλεφώνων, οι κατασκευαστές των κινητών τηλεφώνων αναγκάστηκαν να μειώσουν το μέγεθος και να αυξήσουν τις δυνατότητες που παρέχουν στους χρήστες για να γίνουν όλο και πιο ανταγωνιστικοί στην αγορά εργασίας. Ετσι τα TalkPhones στην πάροδο του χρόνου μετατράπηκαν σε SmartPhones.

Παρακάτω παρουσιάζεται εν συντομία η ιστορική εξέλιξη των κινητών τηλεφώνων και η μετάβαση στα smartphones.

- Περίοδος 1979-1992: Τα κινητά συστήματα έχουν ενσωματωμένα συστήματα για τον έλεγχο της λειτουργίας τους.
- Έτος 1993: Το IBM Simon, θεωρείται το πρώτο smartphone, είχε μια οθόνη αφής ,ηλεκτρονικό ταχυδρομείο και PDA χαρακτηριστικά.
- Έτος 1996: Παρουσιάζεται το Palm Pilot, ένας προσωπικός ψηφιακός βοηθός, ο οποίος περιέχει το Palm OS λειτουργικό σύστημα κινητής τηλεφωνίας.
- Έτος 1996: Τα Windows CE εισάγουν τις φορητές συσκευές PC
- Έτος 1999: Το Nokia S40 OS παρουσιάστηκε επίσημα με το μοντέλο Nokia 7110
- Έτος 2000: Το Symbian έγινε το πρώτο σύγχρονο κινητό λειτουργικό σύστημα σε ένα smartphone με την έναρξη χρήσης του στη συσκευή Ericcson R380.
- Έτος 2002: Η BlackBerry παρουσίασε το πρώτο Smartphone της.
- Έτος 2005: Η Nokia εισήγαγε το Maemo OS για το πρώτο tablet internet N770.
- Έτος 2007 : Η Apple εισήγαγε το Iphone με το λειτουργικό σύστημα iOS, που συνδύασε ‘κινητό τηλέφωνο’ και ΄επικοινωνία στο internet΄.
- Έτος 2007: Το Open Handset Alliance (OHA) δημιουργήθηκε από τις Google , HTC, Sony ,Dell, Intel, Motorola, Samsung ,LG.
- Έτος 2008: Η OHA βγάζει εκδόσεις Android 1.0 με το HTC Dream ως το πρώτο τηλέφωνο Android.
- Έτος 2009: Η Samsung ανακοίνωσε το Bada OS με την εισαγωγή του Samsung S8500.
- Έτος 2010 : Κυκλοφόρησαν Windows Phone OS τηλέφωνα
- Έτος 2012 : Το Lenovo K800 είναι το πρώτο Intel powered smartphone (Android OS).

1.2.1.1 Η ιστορική επισκόπηση του λειτουργικού συστήματος

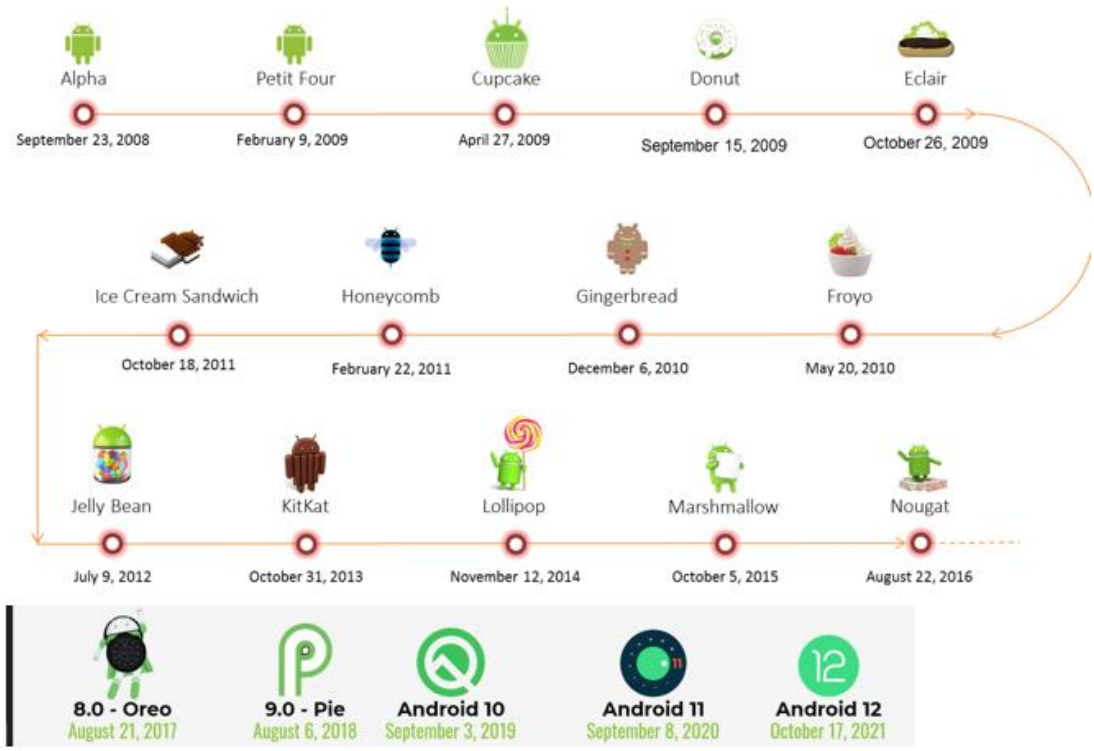


Figure 1.2:2 Η εξελικτική πορεία των εκδόσεων του Android

2.2.1 Android 4.3

Το Android 4.3 (Jelly Bean) βγήκε στην αγορά πριν δέκα περίπου χρόνια, τον Ιούλιο 2013 και έφερνε αρκετά νέα χαρακτηριστικά για εκείνη την εποχή όπως:

- Βελτιώσεις στην απόδοση
- Τρισδιάστατα γραφικά μέσω του OpenGL ES 3.0
- Bluetooth Smart (δηλαδή bluetooth χαμηλής κατανάλωσης ενέργειας κατά τη μεταφορά δεδομένων από εφαρμογές και αισθητήρες)
- Βελτιώσεις στην απόδοση του ήχου
- Γονικό έλεγχο (parental control) για ασφαλή χρήση από ανήλικα τέκνα

Η Google σταμάτησε την υποστήριξη των εκδόσεων 4.1 έως 4.3 τον Αύγουστο του 2021. Την περίοδο εκείνη, ήδη μόλις 1% των Android συσκευών χρησιμοποιούσαν τη συγκεκριμένη έκδοση.

2.2.2 Android 4.4 (KitKat)

Figure 1.2:3-Android KitKat logo

Το Android 4.4, με ψευδώνυμο KitKat παρουσιάστηκε την πρώτη φορά το Σεπτέμβριο του 2013.Το Android 4.4 είχε αρκετές βελτιώσεις στα γραφικά, με εκλεπτυσμένα εικονίδια και γενικά μια επανασχεδιασμένη αρχική οθόνη.

Τα χαρακτηριστικά που προστέθηκαν ήταν:

- Αναλόγως με τις ειδοποιήσεις αλλαγή στην εμφάνιση των μπαρών.
- Βελτιώσεις στο Google Maps.
- Ιδιαίτερες βελτιώσεις στην υποστήριξη ασύρματων συνδέσεων.
- Υποστήριξη για καταγραφή οθόνης μέσω της λειτουργίας screen recording utility

Η Google σταμάτησε την υποστήριξη της έκδοσης 4.4 στις νέες παρουσιάσεις των Google Play Services τον Αύγουστο του 2023.

2.2.3 Android 8.0 (Oreo)

Figure 1.2:4-Android Oreo

Η έκδοση αυτή ήταν η 8η μεγάλη αναβάθμιση του Android. Συνέχισε την μέχρι τότε παράδοση της Google να δίνονται ονόματα "γλυκισμάτων", και έτσι τον Μάρτιο του 2017 που κυκλοφόρησε η έκδοση πήρε το όνομα Oreo από τη γνωστή μάρκα

μπισκότων.

Τα σημαντικότερα χαρακτηριστικά της έκδοσης Oreo ήταν τα εξής:

- **Λειτουργία Picture in Picture (PiP):** Παρέχεται πλέον η δυνατότητα για χρήση δύο εφαρμογών ταυτόχρονα, με την μία από τις δύο να ελαχιστοποιείται σε ένα μικρό τετράγωνο στο κάτω μέρος της οθόνης. Μπορεί πλέον ο χρήστης να παρακολουθεί ένα video στο youtube ενώ ταυτόχρονα να αναζητεί μια επαφή στη λίστα επαφών του τηλεφώνου του.
- **Βελτιωμένη ταχύτητα εκκίνησης της συσκευής.**
- **Νέου τύπου βελτιωμένα notifications** Παράχεται ένας ευκολότερος τρόπος για την διαχείρισή τους.
- **Έξυπνη επιλογή κειμένου:** Παρέχεται εξυπνότερος τρόπος επιλογής κειμένου και λαμβάνεται απόφαση για το ποια εφαρμογή είναι κατάλληλη για αυτή για την εκτέλεση κάποιας ενέργειας (π.χ. Αντιγραφή αριθμού τηλεφώνου)
- Υποστήριξη για περισσότερες οθόνες (multi display control)
- Ο Mediarecorder υποστηρίζει τη μορφή MPEG2_TS που διευκολύνει πολύ την ζωντανή ροή video (streaming).
- **Βελτιωμένη χρήση μπαταρίας:** Με το Android 8.0 γίνεται καλύτερη διαχείριση της μπαταρίας ώστε να αυξάνεται η διάρκειά της. Αυτό επιτυγχάνεται με τον περιορισμό του πλήθους των διεργασιών που τρέχουν στο παρασκήνιο.

2.2.4 Android 9.0 (Pie)

Figure 1.2:5-Android Pie logo

Πρόκειται για την 9η μεγάλη αναβάθμιση του Android. Παρουσιάστηκε για πρώτη φορά τον Αύγουστο 2018. Με τα τελευταία διαθέσιμα στοιχεία από την Google, το 6.6% των κινητών τηλεφώνων με Android, τρέχουν την έκδοση αυτή. Από τα πιο σημαντικά χαρακτηριστικά της είναι τα παρακάτω:

- **Νέα γραμμή πλοήγησης:** Τα τρία εικονίδια πλοήγησης που συναντούσαμε μέχρι τότε στο κάτω μέρος της οθόνης, με τα οποία μπορούσαμε να πάμε στην κεντρική οθόνη, να γυρίσουμε πίσω και να δούμε τις εφαρμογές που χρησιμοποιήθηκαν πρόσφατα, αντικαταστάθηκαν από ένα κουμπί πλοήγησης.
- **Μεγάλες βελτιώσεις στην αισθητική του (νέο material design)**
- **Ακόμα πιο βελτιωμένος έλεγχος μπαταρίας:** Γίνεται χρήση αλγορίθμων

τεχνητής νοημοσύνης για να χρησιμοποιείται με “έξυπνο” τρόπο η μπαταρία της συσκευής**, με τη νέα λειτουργία “Adaptive Battery”**. Το λειτουργικό “μαθαίνει” ποιες εφαρμογές έχουν μεγαλύτερη χρήση και απενεργοποιεί τις υπόλοιπες που τρέχουν στο παρασκήνιο.

- **Έλεγχος φωτεινότητας οθόνης:** Χρησιμοποιείται πλέον τεχνολογία μηχανικής μάθησης (machine learning) για να προσαρμόζεται η φωτεινότητα της συσκευής ανάλογα με το περιβάλλον και τις προτιμήσεις του χρήστη. Αυτό επιτυγχάνεται μέσω της νέας λειτουργίας με το όνομα Adaptive Brightness.

2.2.5 Android 10

Figure 1.2:6-Android 12

Η Google μαζί με την νέα αυτή έκδοση έφερε στους χρήστες και κάποια νέα χαρακτηριστικά, βελτιώσεις και τις πολυσυζητημένες χειρονομίες (gestures). Οι χειρονομίες αντικαθιστούν το παλιό σχεδιασμό των κουμπιών και κάνουν πιο εύκολη την πλοήγηση στο περιβάλλον του λειτουργικού, ιδιαίτερα στα fuul-screen smartphones.

Πλέον για την πλοήγηση του συστήματος υπάρχουν 3 επιλογές:

1. Η πρώτη επιλογή είναι η κλασική πλοήγηση με 3 κουμπιά.
2. Η δεύτερη επιλογή είναι μια παραλλαγή που προσφέρει πλοήγηση με 2 κουμπιά.
3. Η τρίτη επιλογή (με τη χρήση των χειρονομιών) είναι ο νέος τρόπος πλοήγησης στο Android 10.

Οι προεπιλεγμένες χειρονομίες του Android 10 παρουσιάζονται παρακάτω. Εκτελούνται με τον τρόπο που περιγράφουμε και πραγματοποιούν τις ενέργειες που αναφέραμε:

- Κάνοντας Swipe up από το κάτω μέρος της οθόνης , αναλόγως ποια εφαρμογή είναι ανοιχτή μας μεταφέρει στην αρχική οθόνη.

- Με Swipe στα δεξιά ή στα αριστερά από την άκρη της οθόνης ο χρήστης μεταφέρεται στην προηγούμενη σελίδα ή αν υπάρχει κάποια ανοιχτή εφαρμογή κλείνει.
- Η εναλλαγή μεταξύ των ανοιχτών εφαρμογών γίνεται με swipe up και τη διατήρηση του δακτύλου στην οθόνη.

- Η ενεργοποίηση του Google Assistant γίνεται με swipe down στη δεξιά ή αριστερά στη κάτω γωνία της οθόνης.
- Επιπλέον, αν μία εφαρμογή έχει ενεργοποιημένο τον χειρισμό μέσω gestures τότε ο χρήστης αποκτά πρόσβαση στο βασικό μενού κάνοντας swipe από την αριστερή πλευρά με δύο δάχτυλα.
- Ένας ακόμη τρόπος για την εμφάνιση του βασικού μενού είναι το παρατεταμένο πάτημα στην αριστερή πλευρά της οθόνης και swipe up, αυτή η μέθοδος είναι ιδιαίτερα χρήσιμη στον χειρισμό της συσκευής με ένα χέρι.

2.2.6 Android 11

Figure 1.2:7-Android 11

Πρόκειται για την 11η μεγάλη αναβάθμιση του Android. Πρωτοπαρουσιάστηκε τον Σεπτέμβριο 2020. Με τα τελευταία στοιχεία (τέλος 2023), παγκοσμίως, το 17% των Android συσκευών χρησιμοποιούν τη συγκεκριμένη έκδοση.

Ο στόχος της Google με το Android 11 ήταν να βελτιώσει μερικές βασικές πτυχές του λειτουργικού συστήματος που χρειάζονταν κάποιον εκσυγχρονισμό. Μεταξύ αυτών ήταν και οι ειδοποιήσεις. Οι ειδοποιήσεις σε προηγούμενες εκδόσεις Android ήταν πολύ καλές, αλλά στο Android 11, παρουσιάστηκε μια νέα δυνατότητα που κρατά τις σημαντικές ειδοποιήσεις συνομιλίας / μηνυμάτων στην πρώτη γραμμή.

Ονομάστηκε "συννεφάκια συνομιλίας" (chat bubbles) και είναι βασικά μια “αντιγραφή” της δυνατότητας του chat head του Facebook Messenger που υπάρχει εδώ και αρκετά χρόνια. Όταν γίνεται λήψη ενός κειμένου, ή ένα μήνυμα WhatsApp/Viber, ή οτιδήποτε άλλο παρόμοιο, παρέχεται πλέον η δυνατότητα τώρα μετατρέπεται αυτή η τακτική ειδοποίηση σε μια φούσκα συνομιλίας που επιπλέει στο επάνω μέρος της οθόνης μας. Ανεξάρτητα από το τι κάνουμε στο τηλέφωνό μας, τα συννεφάκια συνομιλίας διασφαλίζουν ότι έχουμε πάντα εύκολη πρόσβαση σε αυτήν τη συνομιλία.
Άλλες βελτιώσεις είναι οι εξής:

- **Share menu pinning (καρφίτσωμα του μενού κοινής χρήσης)**

Μπορούμε να καρφώσουμε έως 4 εφαρμογές κοινής χρήσης στην κορυφή του μενού κοινής χρήσης.

- **Σίγαση ειδοποιήσεων κατά την εγγραφή βίντεο.**

Μια ιδιαίτερα χρήσιμη νέα λειτουργία που βοηθά τον χρήστη όταν κάνει καταγραφή video. Στο Android 11 υπάρχει πλέον ένα νέο API κάμερας που επιτρέπει στους χρήστες να θέσουν τις εφαρμογές τους σε σίγαση όταν η κάμερα είναι ανοιχτή.

- **Δυνατότητα ανοικτού Bluetooth κατά τη λειτουργία πτήσης**

Μέχρι και το Android 10, η ενεργοποίηση της λειτουργίας πτήσης απενεργοποιούσε αυτόματα τις ασύρματες συνδέσεις, συμπεριλαμβανομένου του Bluetooth. Αυτό με το Android 11 άλλαξε και πλέον παραμένει ενεργό.

2.2.7 Android 12

Figure 1.2:8-: Android 12 logo

Πρόκειται για την 12η μεγάλη αναβάθμιση του Android. Πρωτοπαρουσιάστηκε τον Οκτώβριο 2021. Με τα τελευταία στοιχεία (τέλος 2023), παγκοσμίως, το 18% των Android συσκευών χρησιμοποιούν τη συγκεκριμένη έκδοση και αποτελεί την δεύτερη πιο χρησιμοποιούμενη έκδοση του συγκεκριμένου λειτουργικού (682 εκατομμύρια συσκευές παγκοσμίως). Το Android 12 έχει να παρουσιάσει δεκάδες νέα πολύ βελτιωμένα χαρακτηριστικά που κάποια από αυτά αναφέρονται παρακάτω:

- **Νέο UI και Material NEXT DESIGN**
- **Βελτιώσεις για μεγάλες οθόνες και lockscreen**
- **Νέα γραφική διεπαφή (UI) ειδοποιήσεων και μενού ρυθμίσεων**
- **Καλύτερα Screenshots κα ευκολότερη κοινοποίηση Wi-Fi**
- **Βελτιώσεις στον ήχο**
- **Αναβαθμίσεις μέσω Google Play**

1.2.2 ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

Στην ενότητα αυτή θα κάνουμε μια σύντομη αναφορά για τα Λειτουργικά συστήματα των έξυπνων κινητών συσκευών (Smart Phones). Θα εξηγήσουμε αναλυτικά τη δομή των λειτουργικών συστημάτων, τα δυνατά και τα αδύνατα σημεία και τις δυνατότητες που παρέχει το κάθε λειτουργικό σύστημα.

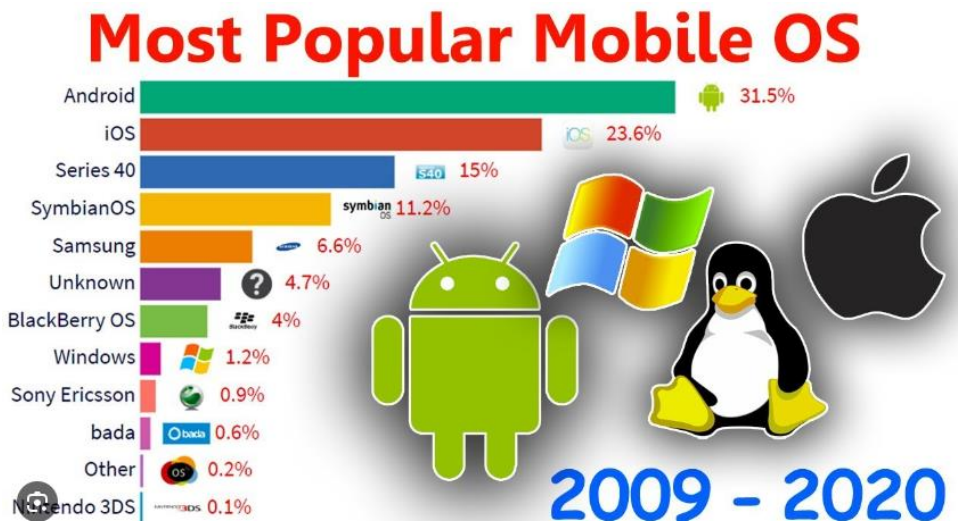


Figure 1.2:9- Τα πιο δημοφιλή λειτουργικά συστήματα κινητών συσκευών (mobile OS) την περίοδο 2009-2020

1.2.2.1 SYMBIAN OS

Το Symbian OS ήταν ένα λειτουργικό σύστημα για φορητές συσκευές που αναπτύχθηκε από την εταιρεία Nokia και πλέον δεν υποστηρίζεται. Αποτελεί εξέλιξη του λειτουργικού συστήματος EPOC της Psion. Το Symbian κυκλοφόρησε το 2001 και ήταν από τα πρώτα "έξυπνα" λογισμικά για κινητά τηλέφωνα. Δημιουργήθηκε με τη γλώσσα προγραμματισμού C++ από τη Symbian Ltd.

Για την εποχή του ήταν αρκετά γρήγορο και σταθερό, κατάφερνε να έχει μια καλή διαχείριση του hardware και επιπλέον έκανε καλή διαχείριση της μπαταρίας.

Figure 1.2:10-Symbian logo

Σε σύγκριση με το λειτουργικό σύστημα iOS του iPHONE της Apple και το Android της Google ,το Symbian θεωρείται απαρχαιωμένο. Αυτό οφείλεται στο γεγονός ότι δεν σχεδιάστηκε αρχικά για τη χρήση στις τόσο δημοφιλή πλέον συσκευές αφής. Το Symbian OS αποτελείτω από πολλά επίπεδα, βιβλιοθήκες λειτουργιών, μηχανές εφαρμογών, διακομιστές, έναν βασικό πυρήνα και ένα επίπεδο διεπαφής υλικού. Το Symbian ήταν το πιο διαδεδομένο λειτουργικό σύστημα κινητής συσκευής μέχρι το

2010, όπου και αντικαταστάθηκε ολοκληρωτικά από το Android.μέχρι το 2010, όπου και αντικαταστάθηκε ολοκληρωτικά από το Android.

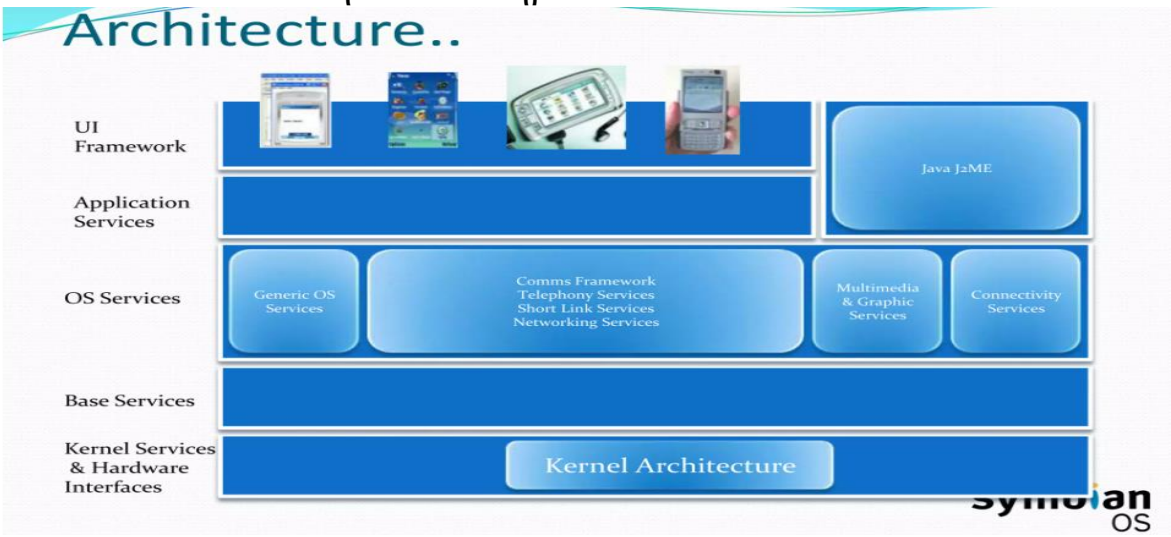


Figure 1.2:11-Mobile Operating System

1.2.2.2 APPLE iOS

Το IOS είναι το δεύτερο πιο διαδεδομένο λειτουργικό σύστημα για έξυπνα τηλέφωνα (28% της παγκόσμιας αγοράς, στοιχεία 2023). Είναι ένα λειτουργικό σύστημα ανεπτυγμένο από την Apple και χρησιμοποιείται αποκλειστικά για συσκευές της (χρησιμοποιείται από τα προϊόντα μόνο για Apple (Iphone, Ipad)).

Η Αρχική του ονομασία ήτανε IPHONE OS, και πρωτοπαρουσιάστηκε το 2007, ενώ για εμπορικούς λόγους αργότερα ονομάστηκε με σκέτο IOS. Είναι ένα "κλειστό" (proprietary) λειτουργικό σύστημα (δηλαδή χωρίς ελεύθερη πρόσβαση στον κώδικά του). Για να δημοσιεύσει κάποιος μια εφαρμογή στο App Store (το online κατάστημα της Apple για εφαρμογές) πρέπει πρώτα να πάρει έγκριση από την Apple. Αυτό το καθιστά πιο ασφαλές, καθώς είναι πολύ πολύ λιγότερο ευάλωτο σε ιούς και σε κακόβουλο λογισμικό.

Figure 1.2:12-IOS UI

Η γλώσσα προγραμματισμού που χρησιμοποιείται για την ανάπτυξη εφαρμογών iOS είναι η Swift ή η Objective-C. Η Apple κάνει συνεχώς αναβαθμίσεις στο λειτουργικό της και τις παρέχει δωρεάν στα περισσότερα έξυπνα κινητά τα οποία έχει βγάλει στην αγορά.

Η αρχιτεκτονική του iOS βασίζεται σε μια πολυεπίπεδη αρχιτεκτονική (layered based architecture) όπου στο ανώτερο επίπεδο, το IOS λειτουργεί ως ενδιάμεσος κρίκος μεταξύ του υλικού και των εφαρμογών. Οι εφαρμογές επικοινωνούν με το hardware μέσω συγκεκριμένων διεπαφών (interfaces) του συστήματος. Αυτές οι διεπαφές καθιστούν απλή την ανάπτυξη εφαρμογών.

Το κατώτερο επίπεδο δίνει τις βασικές υπηρεσίες πάνω στις οποίες βασίζονται όλες οι εφαρμογές ενώ τα ανώτερα επίπεδα παρέχουν τα γραφικά και υπηρεσίες σχετικές με τις διεπαφές. Οι περισσότερες από τις διεπαφές του συστήματος παρέχονται μαζί με ένα ειδικό πακέτο που ονομάζεται πλαίσιο (framework).

Ένα πλαίσιο είναι ένας κατάλογος (φάκελος) ο οποίος κρατάει τις δυναμικά διαμοιραζόμενες βιβλιοθήκες όπως τα .a αρχεία, τα αρχεία επικεφαλίδων (header files), τις εικόνες και τις βοηθητικές εφαρμογές που υποστηρίζουν τη βιβλιοθήκη. Κάθε επίπεδο έχει ένα σύνολο από πλαίσια που είναι χρήσιμα για τους προγραμματιστές.

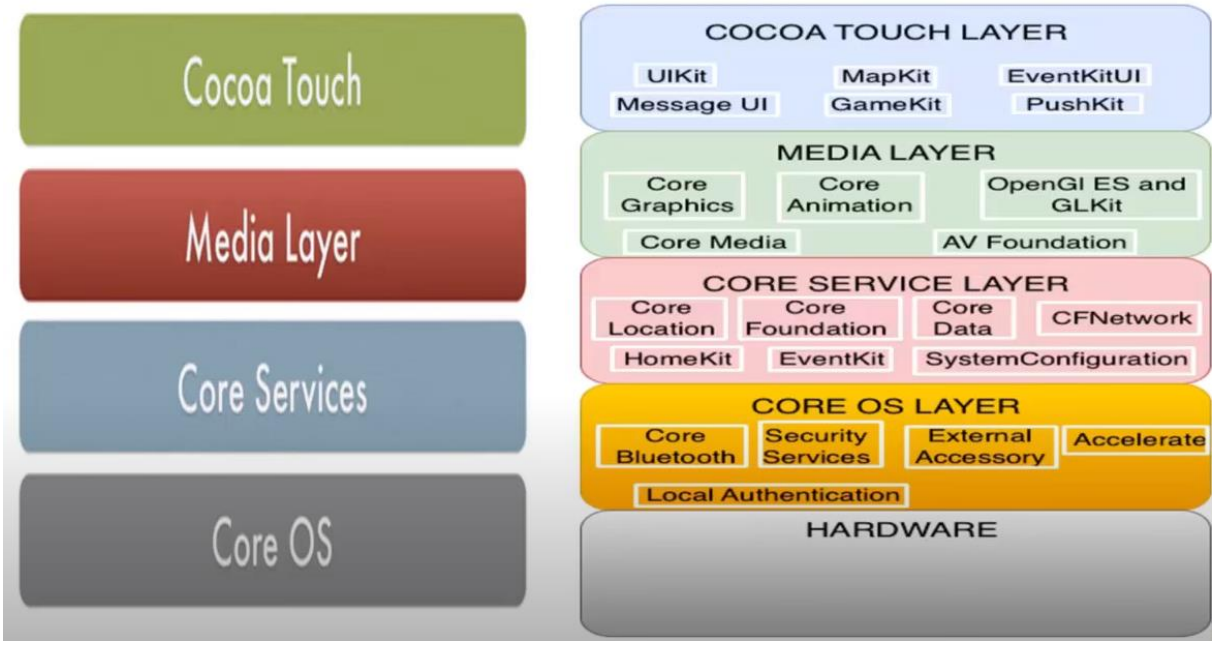


Figure 1.2:13-IOS Architecture

1.2.2.3 "Android Operating System "

Το Android είναι ένα λειτουργικό σύστημα για κινητά (τόσο 32-bit όσο και 64-bit) που βασίζεται σε μια τροποποιημένη έκδοση του πυρήνα του λειτουργικού συστήματος Linux και άλλου λογισμικού ανοιχτού κώδικα, σχεδιασμένο κυρίως για κινητές συσκευές με οθόνη αφής, όπως είναι τα έξυπνα τηλέφωνα (smartphones) και τα tablets. Με τα τελευταία στοιχεία (τέλος του 2023) το Android χρησιμοποιείται από το **70% όλων κινητών συσκευών παγκοσμίως**. Η γλώσσα που χρησιμοποιούν οι προγραμματιστές για την ανάπτυξη των εφαρμογών Android είναι Java ή η Kotlin και μέσω των βιβλιοθηκών που παρέχει η Google μπορούν να δημιουργηθούν ακόμα και πολύ περίπλοκες εφαρμογές. Το λειτουργικό σύστημα Android είναι "ανοιχτού κώδικα" (OpenSource) και είναι διαθέσιμο δωρεάν.

Figure 1.2:14-Android UI

Η Google δημοσιεύει τις καινούριες εκδόσεις του λειτουργικού της συνήθως δίνοντας τους ονόματα γλυκισμάτων όπως:

- Lollipop (γλυφιτζούρι)
- Kitkat (Σοκολατένια μπάρα της Nestle)
- Marshmallow (Ζαχαρωτό)

Το Android λειτουργικό σύστημα έχει ως κέντρο του τον "πυρήνα" (Linux Kernel) και αποτελείται από μια στοίβα λογισμικού. Μια βασική αρχή της αρχιτεκτονικής του Android, είναι η επαναχρησιμοποίηση του κώδικα, πράγμα που παρέχει τη δυνατότητα της δημοσίευσης και διαμοιρασμού των :

- Δραστηριοτήτων
- Υπηρεσιών
- Δεδομένων με άλλες εφαρμογές που τρέχουν επάνω στο λειτουργικό σύστημα.

Χάρη στη δυνατότητα αυτή μπορούμε να κάνουμε επέκταση ή βελτιστοποίηση των ήδη υπαρχόντων εφαρμογών που έχουν αναπτυχθεί, ή μπορούν να δημιουργηθούν νέες από την αρχή.

Το λειτουργικό σύστημα Android δεν είναι απλά ένα ενιαίο λειτουργικό σύστημα, αλλά είναι μια στοίβα λογισμικού που αποτελείται από διάφορα επίπεδα, όπου σε κάθε επίπεδο εκτελείται μια ξεχωριστή λειτουργία. Έτσι έχουμε τις υπηρεσίες διασύνδεσης με τις εφαρμογές και τις κύριες εφαρμογές που είναι (CORE):

- Email client
- Η εφαρμογή διαχείρισης SMS
- Το ημερολόγιο
- Ο browser
- Η εφαρμογή διαχείρισης επαφών (και άλλες βασικές εφαρμογές που είναι προεγκατεστημένες.)

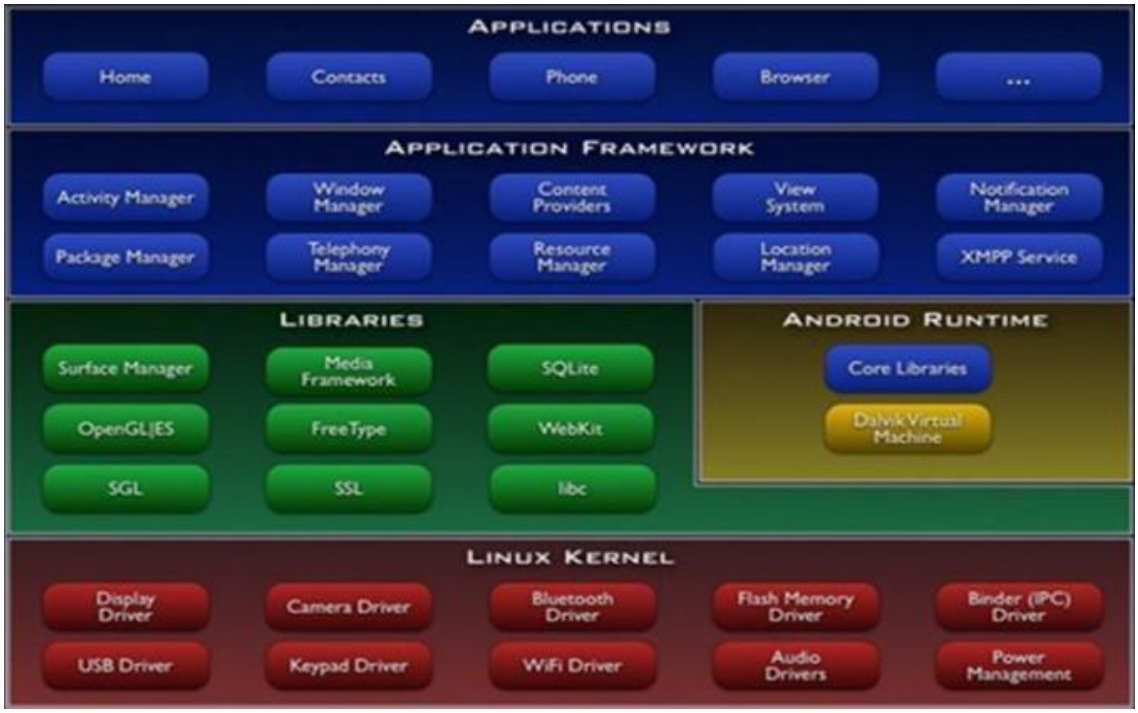


Figure 1.2:15-Τα επίπεδα (layers) android

Η αρχιτεκτονική του λειτουργικού συστήματος Android υποστηρίζει:

- Παράλληλη εκτέλεση πολλαπλών εφαρμογών
- Ισότητα για την εκτέλεση της κάθε εφαρμογής
- Εκτέλεση εφαρμογών χωρίς περιορισμό.

Το Android λειτουργικό σύστημα αποτελείται από πέντε επίπεδα:

[1] Επίπεδο εφαρμογών (Applications)

Στο επίπεδο αυτό μερικές βασικές εφαρμογές που διαθέτει το λειτουργικό σύστημα είναι:

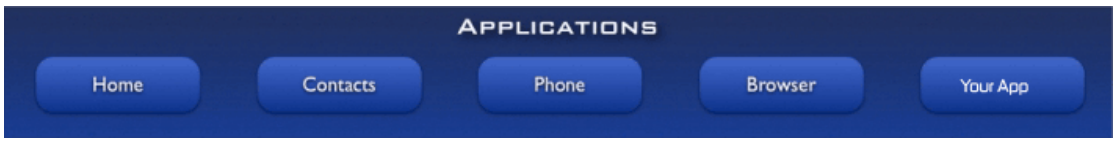


Figure 1.2:16-Application Layer

- E-mail clients
- Αποστολή SMS μηνυμάτων
- Το ημερολόγιο
- Οι Χάρτες (Google Maps)
- Φυλλομετρητής (Browser)
- Πρόγραμμα για επαφές χρηστών

[2] Επίπεδο πλαισίου εφαρμογών (Application Framework)

Figure 1.2:17-Application Framework

Το επίπεδο αυτό μας παρέχει υψηλού επίπεδου δομικές μονάδες που δίνουν τη δυνατότητα στους προγραμματιστές να δημιουργήσουν καινοτόμες εφαρμογές. Στο επίπεδο αυτό οι προγραμματιστές μπορούν να εκμεταλλευτούν τις δυνατότητες του υλικού των κινητών συσκευών (π.χ. όπως ο Χάρτης ή οι αισθητήρες που έχουν οι συσκευές για τον εντοπισμό θέσεων μέσω GPS). Στο επίπεδο αυτό κάθε εφαρμογή που εκτελείται στο λειτουργικό σύστημα μπορεί να χρησιμοποιεί τις δυνατότητες μιας άλλης εφαρμογής.

Τα βασικά δομικά στοιχεία του πλαισίου εφαρμογών είναι:

- **Ο διαχειριστής περιεχομένου (Content Manager)**
  Είναι η δυνατότητα που παρέχει στον προγραμματιστή εφαρμογών να διαμοιράζονται τα δεδομένα από μια εφαρμογή σε μια άλλη. Τέτοια δεδομένα μπορεί να είναι οι επαφές του χρήστη στις βάσεις δεδομένων.
- **Ο διαχειριστής πόρων (Resource Manager)**
  Επιτρέπει την πρόσβαση :
  - *String*
  - *Εικόνες*
  - *Layout Files*
- **Ο διαχειριστής τοποθεσίας (Location Manager)**
  Παρέχει την δυνατότητα μέσω του κατάλληλου υλικού που διαθέτουν οι κινητές συσκευές (π.χ. GPS) για τον καθορισμό της τοποθεσίας επάνω στον χάρτη .
- **Ο διαχειριστής ειδοποιήσεων (Notification Manager)**
  Ο διαχειριστής ειδοποιήσεωνείναι μια υπηρεσία του λειτουργικού συστήματος που διαχειρίζεται την εμφάνιση των διαφόρων ειδοποιήσεων σε μια συσκευή Android. Επιτρέπει στους προγραμματιστές να δημιουργούν και να εμφανίζουν ειδοποιήσεις στους χρήστες για λογαριασμό των εφαρμογών τους. Το API του Notification Manager παρέχει τη δυνατότητα δημιουργίας ειδοποιήσεων και ελέγχου της συμπεριφοράς τους.
  Οι ειδοποιήσεις μπορούν να εμφανίζονται στη σκίαση ειδοποιήσεων, στην οθόνη κλειδώματος και στη γραμμή κατάστασης. Ο διαχειριστής ειδοποιήσεων παρέχει επίσης τη δυνατότητα προσαρμογής των ήχων των ειδοποιήσεων, των μοτίβων δόνησης και των χρωμάτων LED.
- **Ο διαχειριστής δραστηριοτήτων (Activity Manager)**

Ρυθμίζει τη διάρκεια ζωής των διαφόρων εφαρμογών και επιτρέπει στο χρήστη να πλοηγείται στις προηγούμενες εφαρμογές

- **Το σύστημα προβολών (View System)**
  Περιέχει το σύνολο των αντικειμένων που χρησιμοποιούνται στο σχεδιασμό της οπτικής διεπαφής όπου αλληλεπιδρά με τον χρήστη. Μερικά από αυτά είναι τα πεδία εισαγωγής κειμένου, το πλέγμα , οι λίστες.
- **Ικανότητα δικτύωσης (Connectivity Manager)**
  Περιέχει πληροφορίες για τις δυνατές συνδέσεις μιας συσκευής, καθώς και την κατάσταση κάθε σύνδεσης.

[3] Επίπεδο χρόνου εκτέλεσης εφαρμογών (Android Runtime)

Figure 1.2:18-Android Operating System Runtime Layer

Περιέχει ένα σύνολο βασικών βιβλιοθηκών και την **Dalvik Virtual Machine .**

Το Dalvik Virtual Machine (που πλέον δεν υποστηρίζεται) είναι μια εικονική μηχανή διεργασιών (process virtual machine) στο λειτουργικό σύστημα Android η οποία εκτελεί εφαρμοές γραμμένες για αυτό το λειτουργικό. Η μορφή Dalvik bytecode χρησιμοποιείται ακόμα για διανομή αλλά όχι στο περιβάλλον εκτέλεσης των νεότερων εκδόσεων του λειτουργικού Android.

[4] Επίπεδο βιβλιοθηκών (Libraries)

Figure 1.2:19-Οι βιβλιοθήκες του Android

Οι κύριες βιβλιοθήκες που διαθέτει το λειτουργικό σύστημα Android είναι :

1) **System C Library :**
   Είναι η ενσωματωμένη βιβλιοθήκη της C, διαμορφωμένη έτσι ώστε να είναι κατάλληλη για κινητές συσκευές που βασίζονται σε Linux. Η κύρια αρμοδιότητα της βιβλιοθήκης είναι οι λειτουργίες που αφορούν την μείωση της μνήμης.
2) **Surface Manager:**
   Είναι ο διαχειριστής των σχεδιαστικών επιφανειών. Η σχεδιαστική επιφάνεια είναι η οθόνη της συσκευής που έχει δεσμευθεί για να κρατήσει τα δεδομένα προς απεικόνιση. Ο Διαχειριστής
   των επιφανειών σε συνδυασμό με το διαχειριστή παραθύρων (Window Manager) καθορίζουν τη σειρά σχεδίασης των παραθύρων, έτσι ώστε να υπάρχει μια σωστή αλληλουχία.
3) **SQLITE:** αποτελεί μια πολύ ισχυρή βάση δεδομένων που χρησιμοποιείται από κινητές εφαρμογές. Εκτελεί τις λειτουργίες όπως η εγγραφή - ανάγνωση και αποθήκευση των δεδομένων.
4) **Βιβλιοθήκες 3D:** είναι οι βιβλιοθήκες αυτές που χρησιμοποιούνται για την τρισδιάστατη απεικόνιση**.**
5) **Βιβλιοθήκες πολυμέσων:** είναι η βιβλιοθήκη που επιτρέπει την αναπαραγωγή και εγγραφή πολλαπλών μέσων εικόνας και ήχου.
6) **SGL:** είναι μια μηχανή για που παρέχει τη δυνατότητα σχεδίασης των δισδιάστατων γραφικών**.**
7) **FreeType:** προσφέρει ευκρίνεια στα γραφικά που χρησιμοποιούνται στα bitmaps και στις γραμματοσειρές των εφαρμογών.
8) **LibWebCore:** παρέχει την πλοήγηση στο διαδίκτυο και χρησιμοποιείται από τον browser του Android και τις Web Views.
9) **WebKit:** είναι η μηχανή διάταξης των γραφικών που δίνει την δυνατότητα στον browser να απεικονίσει τις ιστοσελίδες.

[5] Επίπεδο Πυρήνα του LINUX

Figure 1.2:20-Linux kernel in Android Operating System

Το λογισμικό του Android βασίζεται στον πυρήνα Linux,έκδοση 4.9 που επιτρέπει να μπορεί να χρησιμοποιηθεί σε ένα μεγάλος εύρος από πλατφόρμες.

Ειδικότερα χρησιμοποιείται :

- Για τη διαχείριση μνήμης
- Για τη διαχείριση των διεργασιών
- Τις λειτουργίες του δικτύου
- Για την ασφάλεια του λειτουργικού συστήματος
- Οδηγίες για υλικό

1.2.2.4 Windows Phone

Το Windows Phone είναι μια οικογένεια λειτουργικών συστημάτων για κινητές έξυπνες συσκευές που αναπτύχθηκε από την Microsoft το 2010 για να αντικαταστήσει την προηγούμενη έκδοση λειτουργικού για κινητά της ίδιας εταιρείας που λεγόταν Microsoft Mobile.

Figure 1.2:21-XAMARIN UI

Το Windows Phone δεν είναι λειτουργικό σύστημα ανοικτού κώδικα και από το 2017 η Microsoft έχει ανακοινώσει την μη συνέχιση ανάπτυξής του. Το 2022 ήταν η τελευταία χρονιά που η εταιρεία παρείχε υποστήριξη για συσκευές που χρησιμοποιούσαν Windows phone λειτουργικό. Ένα από τα πλεονεκτήματά του ήταν ότι παρείχε εντελώς δωρεάν τη σουίτα εφαρμογών Office για κινητές συσκευές. Επίσης οι προγραμματιστές είχαν τη δυνατότητα να χρησιμοποιήσουν εφαρμογές που είχαν δημιουργήσει είτε σε

Android είτε σε C++ είτε σε Objective-C είτε σε Java. Το λειτουργικό σύστημα βασιζόταν σε μια ειδική έκδοση του .Net framework, ενώ σε αναλογία με το Android, είχε αρχιτεκτονική βασισμένη σε επίπεδα.

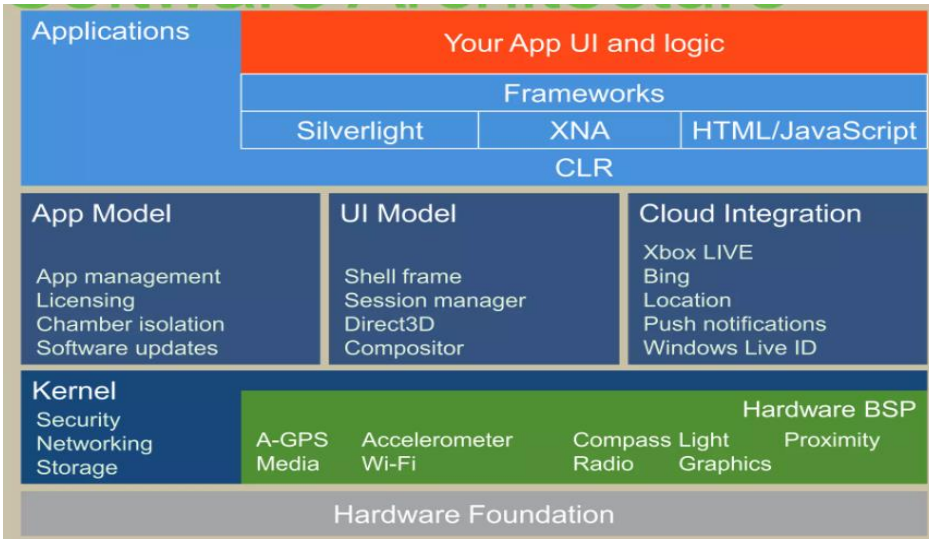


Figure 1.2:22-XAMARIN ARCHITECTURE

ΚΕΦΑΛΑΙΟ 2: ΕΦΑΡΜΟΓΕΣ ΔΙΑΧΕΙΡΙΣΗΣ ΚΤΗΝΙΑΤΡΕΙΟΥ

2.1 Εισαγωγή

Στην ενότητα αυτή θα παρουσιάσουμε τα βασικά εργαλεία που χρησιμοποιήσαμε κατά τη διάρκεια της ανάπτυξης της εφαρμογής μας. Θα αναφερθούμε σε τεχνολογίες τοπικών βάσεων δεδομένων που τρέχουν στην εφαρμογή μας όπως το SQLITE ή το RoomDB και για τις τεχνολογίες αποθήκευσης νέφους όπως το Firebase. Τελός θα μιλήσουμε για το περιβάλλον ανάπτυξης Android Studio και θα εξηγήσουμε τις δυνατότητες που μας παρέχει η γλώσσα JAVA.

2.2 Βάσεις δεδομένων

Οι βάσεις δεδομένων σε κινητές συσκευές χωρίζονται σε δύο μεγάλες κατηγορίες:

Τοπικές (Local Database):

- SQLITE (RDBMS-Relation Database Management System)
- RoomDB

Βάσεις δεδομένων Νέφος (Cloud Based Database)

- Firebase (NOSQL)

2.2.1 SQLITE

Το SQLITE είναι βασικά ένα Σύστημα Διαχείρισης Σχεσιακών βάσεων δεδομένων (RDBMS), το ίδιο με το SQL. Είναι μια βιβλιοθήκη ανοιχτού κώδικα σε διαδικασία που είναι αυτόνομη, χωρίς διακομιστή, έχει μηδενικές ρυθμίσεις παραμέτρων. Εδώ η μηδενική διαμόρφωση σημαίνει ότι σε αντίθεση με άλλα συστήματα διαχείρισης βάσεων δεδομένων, δεν χρειάζεται να διαμορφωθεί στις συσκευές.

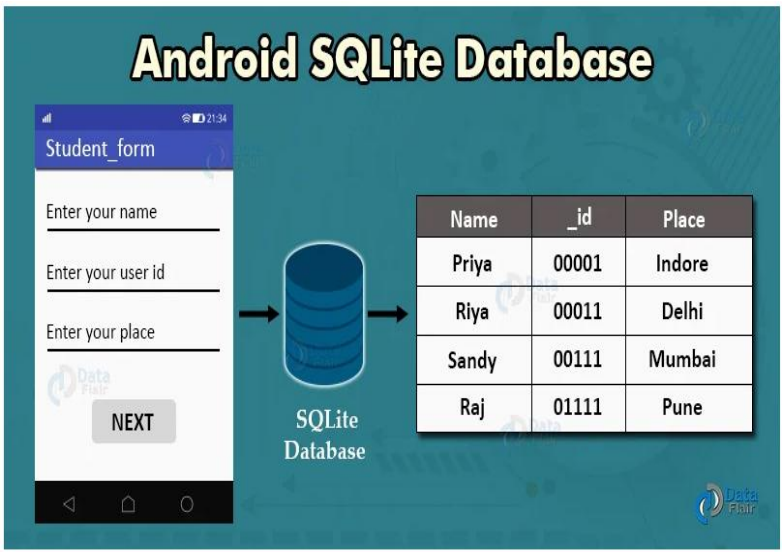


Figure 2.2:23-SQLITE

Έτσι το SQLITE χρησιμοποιείται για την εκτέλεση των λειτουργιών στις βάσεις δεδομένων που είναι αποθηκευμένες με τη μορφή γραμμφών και στηλών. Το SQLITE υποστηρίζεται ιδιαίτερα από το Android. Στην πραγματικότητα ,το Android έρχεται μαζί με την ενσωματωμένη εφαρμογή βάσης δεδομένων του SQLITE. Είναι διαθέσιμο σε κάθε Android εφαρμογή και δίνει έμφαση :

- στην επεκτασιμότητα
- στη συλλογή
- στην συγχρονικότητα
- στον έλεγχο

Προσπαθεί να παρέχει χώρο αποθήκευσης σε τοπικό επίπεδο και για εφαρμογές και για τις συσκευές.

Παρέχει τα παρακάτω θετικά στοιχεία:

- Είναι εξαιρετικά οικονομικό
- Είναι αποδοτικό
- Είναι αξιόπιστο
- Είναι ανεξάρτητο

Είναι πολύ απλό και δεν χρειάζεται να συγκριθεί με βάσεις δεδομένων πελάτη/διακομιστή.

Το Sqlite υποστηρίζει του ακόλουθους τρεις τύπους δεδομένων :

- **Text Type:** για αποθήκευση συμβολοσειρών ή δεδομένων τύπου χαρακτήρα.
- **Integer Type:** για αποθήκευση ακέραιου τύπου δεδομένων**.**
- **Real Type:** για αποθήκευση μεγάλων τιμών.

2.2.2 Shared Preferences

Ο πρωταρχικός σκοπός του Shared Preferences είναι να αποθηκεύει μικρού μεγέθους δεδομένα τα οποία δεν χρειάζονται να αποθηκευτούν στην βάση δεδομένων αλλά τα δεδομένα αυτά είναι χρήσιμα κατά τη διάρκεια της λειτουργίας της εφαρμογής. Μπορούμε να χρησιμοποιήσουμε το Shared Preferences σε περιπτώσεις όπου δεν χρειάζεται να αποθηκεύσουμε πολλά δεδομένα και δεν χρειαζόμαστε κάποια συγκεκριμένη δομή. Συνήθως χρησιμοποιούνται για μικρού μεγεθους δεδομένα.

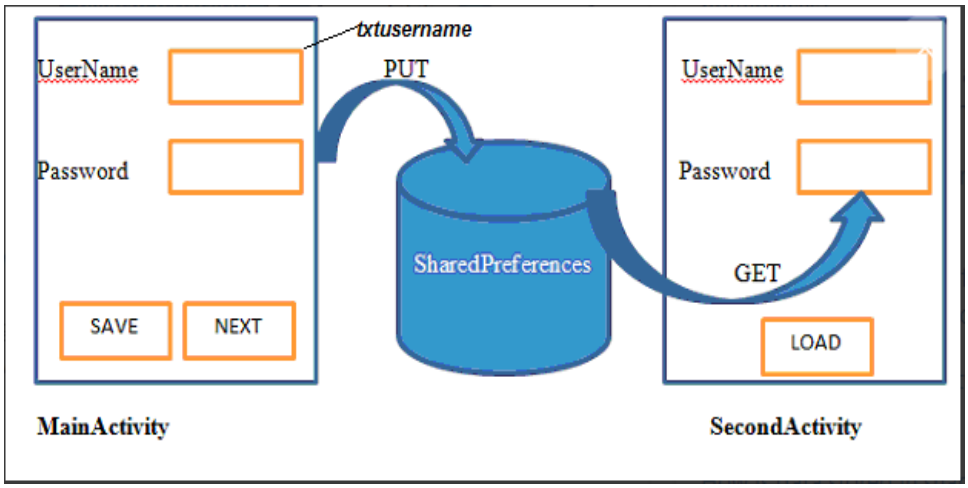


Figure 2.2:24-Shared preferences

2.2.3 Firebase

Figure 2.2:25-Firebase Icon

Είναι μια πλατφόρμα που παρέχει υπηρεσίες **backend** για τους προγραμματιστές λογισμικού που αναπτύσσουν εφαρμογές ή προγράμματα σε τομείς (java, flutter, Android). Προσφέρει τη δυνατότητα απλοποίησης πολλών διεργασιών. Επιπλέον, υπάρχει η σημαντική επιλογή της συνέπειας ανάλυσης δεδομένων στις εμφανίσεις των χρηστών.

Ας δούμε λίγο την ιστορική εξέλιξη της πλατφόρμας.

Η πλατφόρμα ιδρύθηκε το 2011 από δύο startups των Andrew Lee και James Tamplin. Το πρώτο όνομα εκείνη την εποχή ήτανε ENVOLVE. Ο κύριος σκοπός της πλατφόρμας ήταν να μπορούν οι προγραμματιστές λογισμικού να προσθέσουν μια εφαρμογή συνομιλίας (CHAT) στις εφαρμογές τους με απλοϊκό τρόπο. Το πρώτο προϊον Firebase που κυκλοφόρησε το 2012 ήταν η Βάση δεδομένων Firebase Realtime.

Παρείχε ένα API για συγχρονισμό δεδομένων μεταξύ των εφαρμογών του Android,web και iOS. Το Firebase Authentication και το Firebase Hosting κυκλοφόρησαν το 2014 καθιστώντας την εταιρεία ως κορυφαίο Backend Service.

Η λύση Baas μας παρέχει την δυνατότητα της απευθείας διαχείρισης της βάσης δεδομένων χωρίς τη διαμεσολάβηση κάποιας backend γλώσσας. Στην παραδοσιακή ανάπτυξη εφαρμογών χρειάζεται να υλοποιηθεί ο αντίστοιχος κώδικας για τη μεταφορά δεδομένων από την βάση. Έτσι, με τη βοήθεια της Firebase , το παραδοσιακό backend παρακάμπτεται.

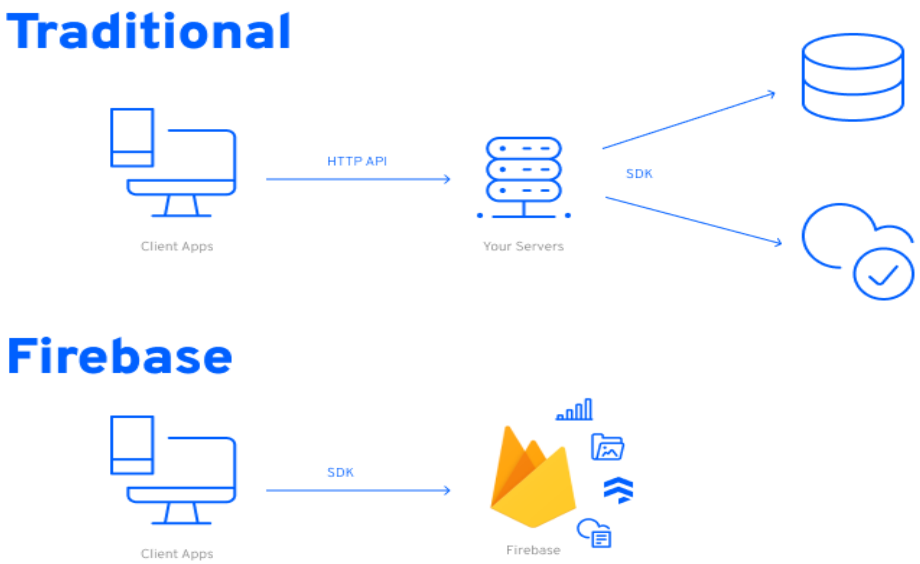


Figure 2.2:26-Traditional vs Firebase application

Πλέον, μέσω των αποκλειστικών API της Firebase μπορούμε να εκτελέσουμε τις υπηρεσίες ανάκτησης, ενημέρωσης και διαγραφής των δεδομένων κατευθείαν με χρήση της πλατφόρμας.

Το 2014 η Firebase εξαγοράστηκε από την Google και προστέθηκαν επιπλέον λειτουργίες όπως αυτές που φαίνονται στο παρακάτω σχήμα:

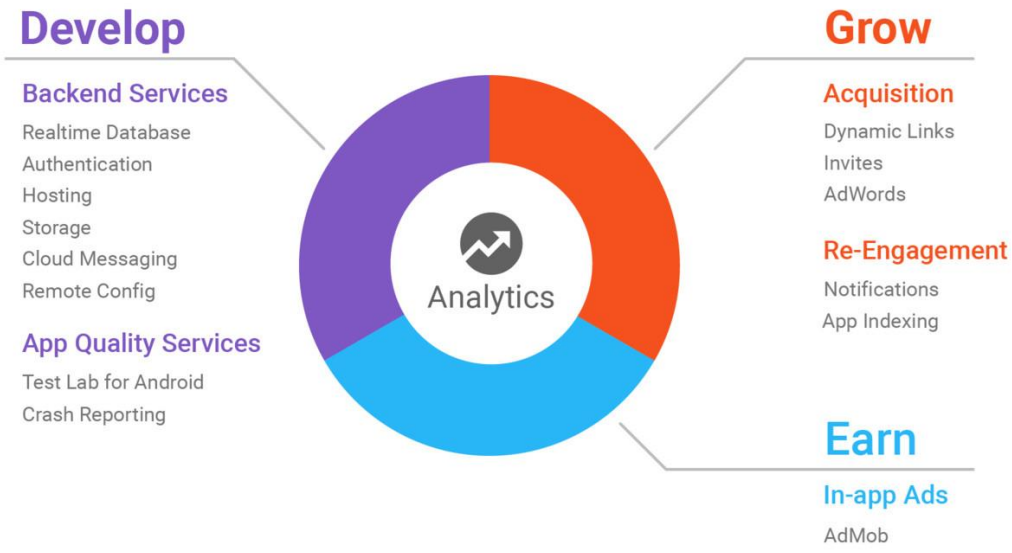


Figure 2.2:27-Νέες λειτουργίες της Firebase (Firebase Services)

Όπως φαίνεται στην παραπάνω εικόνα, υπάρχουν πολλές υπηρεσίες που παρέχει. Εκτός από την παροχή δωρεάν χρήσης, μπορούμε να έχουμε πρόσβαση στη βάση δεδομένων όπου καταχωρούνται τα δεδομένα χρήστη, εγγραφή, σύνδεση (login).

2.2.3.1 ΤΑ ΒΑΣΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΣ GOOGLE FIREBASE:

- Παράλληλη (σε πραγματικό χρόνο) αποθήκευση δεδομένων.
- Ταυτοποίηση χρηστών κατά την είσοδο
- Αποθήκευση δεδομένων (Υπηρεσίες Νέφους)
- Μηχανική μάθηση
- Ειδοποιήσεις (Notifications)
- Διαχειριστικό περιβάλλον (Admin panel)

2.2.3.2 Τι μπορούμε να κάνουμε μέσω Firebase:

- Μπορούμε να φτιάξουμε μια εφαρμογή συνομιλίας (Zoom, Telegram, Whatsapp)
- Μπορούμε να φτιάξουμε μια πλατφόρμα κοινωνικής δικτύωσης όπου οι χρήστες μπορούν να μοιράζονται δημοσιεύσεις / κοινοποιήσεις και οι άλλοι χρήστες να μπορούν να τα δουν αυτές τις κοινοποιήσεις άμεσα (Instagram, Facebook, Twitter).
- Μπορούμε να φτιάξουμε μια διαδικτυακή εφαρμογή διαγωνισμού όπου οι χρήστες ανταγωνίζονται μεταξύ τους.
- Μπορούμε να δημιουργήσουμε ένα σύστημα παρακολούθησης αποθέματος στο οποίο μπορούν να έχουν πρόσβαση ταυτόχρονα πολλά άτομα όσο από κινητές συσκευές αλλά όσο και από υπολογιστές.

2.2.3.3 Οι εφαρμογές που χρησιμοποιούν Firebase:

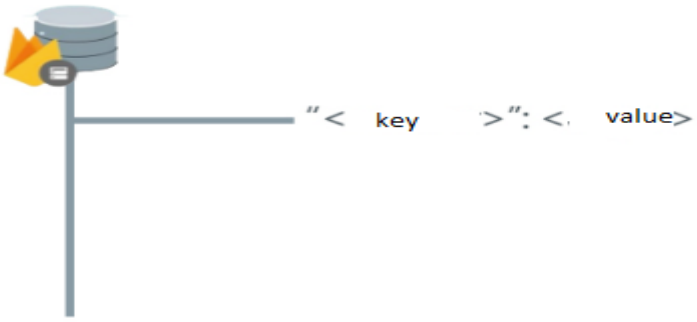
Figure 2.2:6 Κάποιες εφαρμογές που χρησιμοποιούν Firebase-

2.2.3.4 Οι υπηρεσίες της πλατφόρμας Firebase

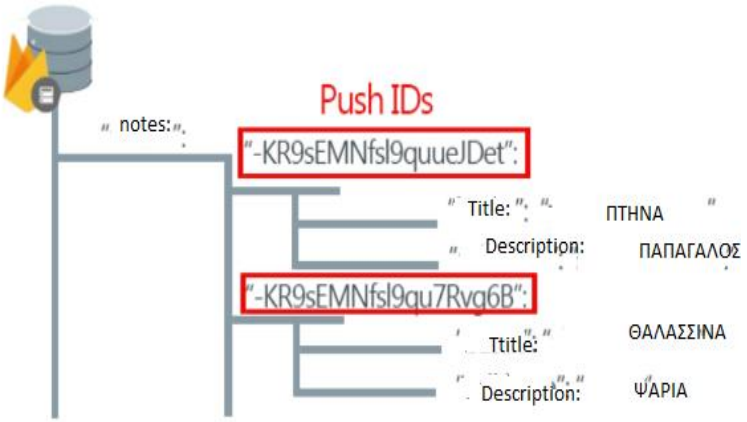
- **Authentication (Ταυτοποίηση χρήστη):** Μπορούμε να κάνουμε περιήγηση στις πληροφορίες των χρηστών που έχουν κάνει εγγραφή στην εφαρμογή μας. Επίσης μπορούμε να κάνουμε ταυτοποίηση των χρηστών. Μέσω της πλατφόρμας μπορούμε να δούμε τα στοιχεία των χρηστών όπως το email, την ημερομηνία εγγραφής. Με αυτή την υπηρεσία παρέχει λειτουργίες όπως επαλήθευση email, επαναφορά κωδικού πρόσβασης.
- **Database (Βάση δεδομένων):** Χάρη σε αυτή την υπηρεσία, το Firebase παρέχει στους χρήστες μια υπηρεσία βάσης δεδομένων NoSql ταυτόχρονης (σε πραγματικό χρόνο) πρόσβασης, που λειτουργεί σε ασύγχρονη (ξεχωριστή) δομή.
- **Storage (Μέσο αποθήκευσης):** Μέσω αυτής της υπηρεσίας μπορούμε να κάνουμε αποθήκευση ή αντιγραφή των δεδομένων (εικόνα η βίντεο) που έχουμε στον υπολογιστή μας στο νέφος (Cloud) και να έχουμε πρόσβαση από όπου θέλουμε.
- **Notification (Ειδοποιήσεις):** Εάν θέλουμε να επικοινωνήσουμε άμεσα με τους χρήστες της εφαρμογής μας, είτε στέλνοντας ειδοποιήσεις, είτε στέλνοντας μηνύματα, αυτή η υπηρεσία θα είναι η καλύτερη επιλογή.
- **Admob:** Μέσω αυτής της υπηρεσίας μπορούμε να προσθέσουμε διαφημίσεις της Google στην εφαρμογή μας και να κερδίσουμε χρήματα.
- **Firebase Analytics (Ανάλυση δεδομένων μέσω Firebase):** Με αυτή τη δομή, μπορούμε να έχουμε άμεση πρόσβαση σε πολλές πληροφορίες όπως ο αριθμός των ενεργών χρηστών, οι αλληλεπιδράσεις μεταξύ τους και πολλά άλλα στατιστικά που αφορούν τη λειτουργία της εφαρμογής μας

Database (Βάση δεδομένων): Η βάση δεδομένων Firebase είναι μια τεχνολογία αποθήκευσης δεδομένων πραγματικού χρόνου και ανήκει στην κατηγορία των βάσεων δεδομένων **NOSQL** (Μη Σχεσιακή βάσης δεδομένων) δηλαδή δεν υπάρχει κάποια συσχέτιση μεταξύ των πινάκων αλλά τα δεδομένα αποθηκεύονται σε ***JSON(JAVASCRIPT OBJECT NOTATION)*** Format σε σύννεφο(**CLOUD BASE DATABASE**).

Η διαχείριση τους είναι δυνατή με αντικείμενα JSON χωρίς να απαιτείται κανένα ερώτημα SQL. Εκτός από την αποθήκευση των δεδομένων παρέχει την δυνατότητα της άμεσης παρακολούθησης των αλλαγών των δεδομένων με τη σύγχρονη λειτουργία του. Η αποθήκευση δεδομένων γίνεται μέσω Key-Value μοντέλο. Για κάθε δεδομένα έχουμε κάποιο κλειδί μέσω του οποίου μπορούμε να προσπελάσουμε τα δεδομένα που αντιστοιχούν στο αντίστοιχο κλειδί.



Η αποθήκευση δεδομένων σε μορφή JSON έχουν μια δενδροειδή δομή.Όταν προσθέτουμε δεδομένα στο δέντρο JSON, τα δεδομένα γίνονται κόμβος(Node) σε μορφή JSON.



Για κάθε χρήστη της εφαρμογής που συνδέεται με τη βάση του αναθέτουμε κάποιο αναγνωριστικό κλειδί το οποίο είναι μοναδικό(Πρωτεύον)και μέσω του οποίου είτε

γίνεται η προσθήκη δεδομένων στον αντίστοιχο κόμβο(Node) είτε γίνεται η προσπέλαση των δεδομένων του αντίστοιχου χρήστη.

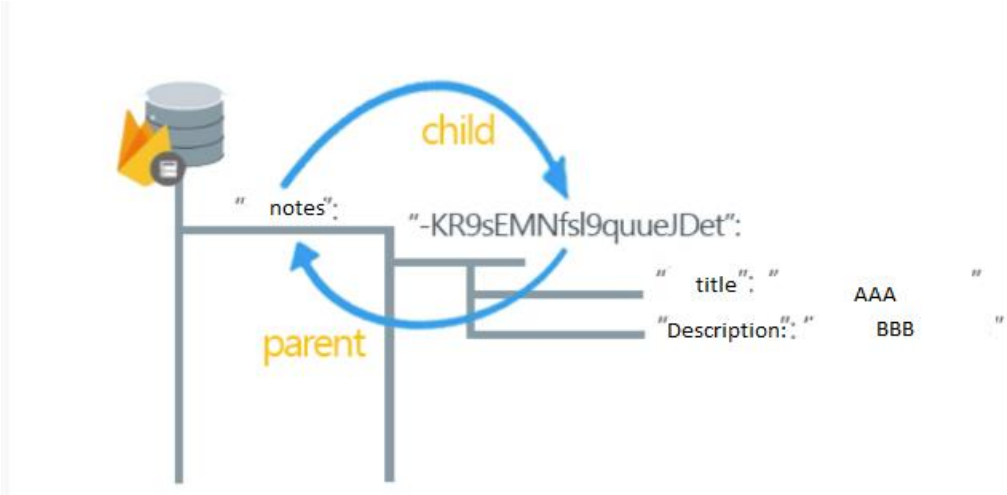
Οι έννοιες Parent &Child:

Υπάρχουν δύο βασικές έννοιες στη δομή του JSON:

Παιδί(Child):

Γονέας(Parent):

Στην παρακάτω δομή ο κόμβος με το ID του χρήστη αναφέρεται ώς θυγατρικός κόμβος(Child Node) του κόμβου “notes(Parent Node)”.



Η δήλωση θυγατρική σας επιτρέπει να εκτελείτε λειτουργίες όπως η ανάγνωση η εγγραφή μεταξύ των δεδομένων του γονικού κόμβου.

Στην παραπάνω εικόνα βλέπουμε την εφαρμογή των εννοιών **Parent & Child** στην εφαρμογή μας.Βλέπουμε για κάθε συνομιλία του χρήστη διατηρούμε μια σχέση

Γονέα παιδί μεταξύ των κόμβων για την αποθήκευση των δεδονμέων στην δενδροειδή βάση μας.

2.3 Android Studio

Η ανάπτυξη της εφαρμογής “Vet App” αναπτύχθηκε στο ολοκληρωμένο προγραμματιστικό περιβάλλον Android Studio. Το Android Studio είναι η επίσημο περιβάλλον ανάπτυξης για εφαρμογές (IDE) για το Android. Κυκλοφόρησε το 16 Μαίου 2003 σε συνέδριο της Google και αναπύχθηκε από την εταιρεία JETBRAINS-INTELIJ IDE.

Figure 2.3:28-Εκδοση Android Studio

Επίσης περιέχει το Android SDK (software development kit) το οποίο περιλαμβάνει ένα σύνολο από εργαλεία ανάπτυξης. Σε αυτά περιλαμβάνονται οι κατάλληλες βιβλιοθήκες που χρειάζονται για την ανάπτυξη των εφαρμογών, ένας αποσφαλματωτής (debugger), δείγμα κώδικα και ένας εξομοιωτής. Το SDK είναι ελεύθερα διαθέσιμο για όλα τα λειτουργικά συστήματα (όπως Windows, Linux, MACOS).

Μέχρι το 2014 το κύριο περιβάλλον ανάπτυξης (IDE) Android εφαρμογών ήταν το το Eclipse. Από το 2015 το Android Studio είναι το επίσημο IDE. Η γλώσσα προγραμματισμού που υποστηρίζει είναι είτε Java είτε Kotlin. Το Android Studio δίνει δυνατότητα συγγραφής, επεξεργασίας και αποθήκευσης των εφαρμογών και όλων των αρχείων που περιλαμβάνουν. Επίσης, το Android Studio επιτρέπει την εκτέλεση του συγγραφέντος κώδικα, είτε μέσω εξομοιωτή είτε μέσω hardware. Μέσω του Android Studio μπορεί να γίνει εντοπισμός των σφαλμάτων στον διαθέσιμο κώδικα.

Αρχικά παρουσιάστηκε σε ένα συνέδριο της Google I/O στις 16 Μαίου του 2013 αλλά ήταν ακόμα σε πρώιμο στάδιο και έτσι μετά από έναν χρόνο, το Δεκέμβριο του 2014 ήταν έτοιμη η έκδοση 1.0.

- Το Android Studio παρέχει διάφορες υπηρεσίες και ευκολίες στους προγραμματιστές των εφαρμογών όπως
- Είναι ευέλικτο (Cradle-based) σύστημα κατασκευής.
- Παρέχει ένα γρήγορο και πολλαπλών λειτουργιών εξομοιωτή (Emulator).
- Δυνατότητα ανάπτυξης εφαρμογών για όλες τις συσκευές Android (Smartphones, Smart-Tv)
- Όταν γίνονται τροποποιήσεις στον κώδικα το Android Studio επιτρέπει την άμεση εκτέλεση του χωρίς να δημιουργεί καινούριο εκτελέσιμο Apk (Τα αρχεία .apk είναι τα εκτελέσιμα αρχεία εγκατάστασης εφαρμογών Android).
- Περιέχει τα κατάλληλα εργαλεία για την άμεση επικοινωνία με το Github, προκειμένου να υλοποιηθεί ο κώδικας που βρίσκετε στο διαδίκτυο.
- Περιέχει εργαλεία ελέγχου της απόδοσης και έλεγχο συμβατότητας εφαρμογών.
- Υποστηρίζει τη Γλώσσα C++

Τέλος το τελευταίο πράγμα πρέπει να αναφέρουμε στο σημείο αυτό είναι το AVDManager. Το AVD Manager είναι το βοηθητικό εργαλείο όπου τρέχουμε τον εξομοιωτή και βλέπουμε τα αποτελέσματα του εκτελέσιμου κώδικα. Για την δημιουργία του εξομοιωτή χρησιμοποιούμε το AVDManager. Τα βήματα που πρέπει να ακολουθήσουμε για την δημιουργία του εξομοιωτή είναι τα εξής:

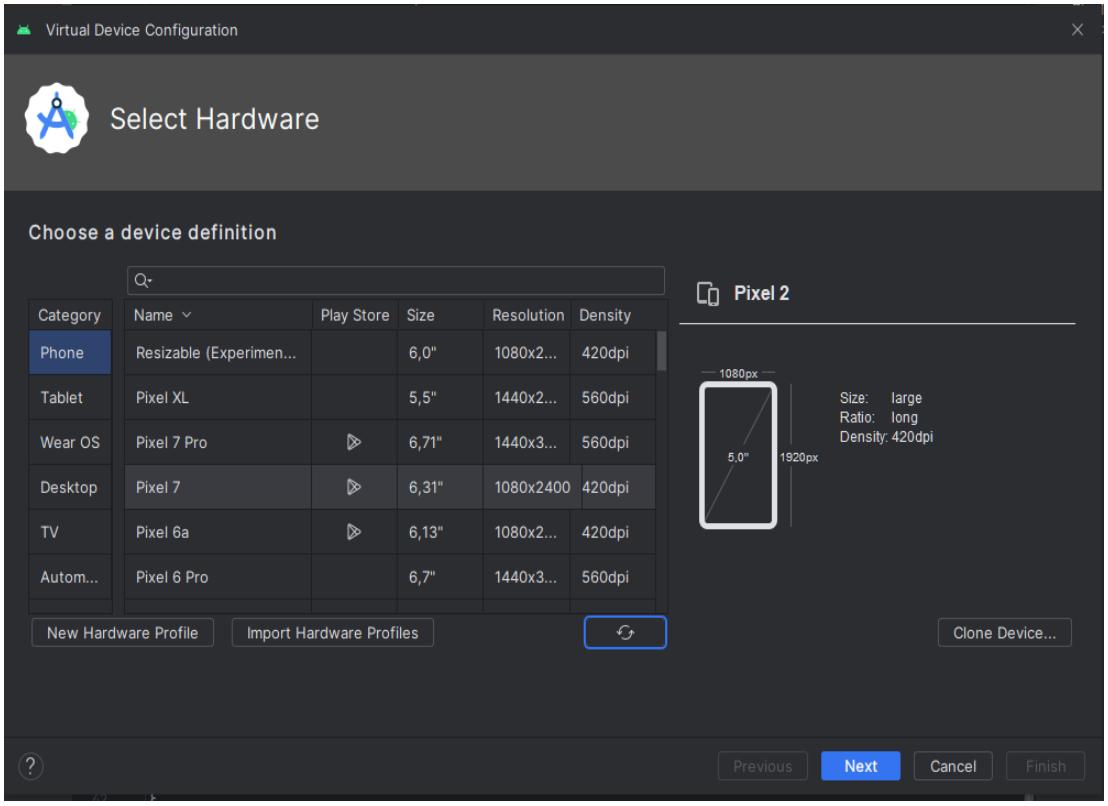


Figure 2.3:29-Android Emulator

Στο σημείο αυτό πατώντας στο κουμπί Next θα επιλέξουμε την εικονική συσκευή που θέλουμε να δημιουργηθεί. Δηλαδή καθορίζουμε την έκδοση API του Android που θέλουμε να τρέξουμε την εφαρμογή μας. Αν για κάποιο λόγο κάποια έκδοση του Android λειτουργικού δεν είναι διαθέσιμο τότε δίπλα εμφανίζεται ένα κουμπί Download όπου μας παρέχει την δυνατότητα της εγκατάστασης της συγκεκριμένης έκδοσης. Βλεπουμε στην παρακάτω εικόνα ότι η έκδοση Android 11 είναι διαθέσιμη στην διεπαφή μας αλλά αν θέλει να εγκαταστήσει μια καινούρια έκδοση πρέπει πατήσει το βελάκι που είναι διαθέσιμο δίπλα.

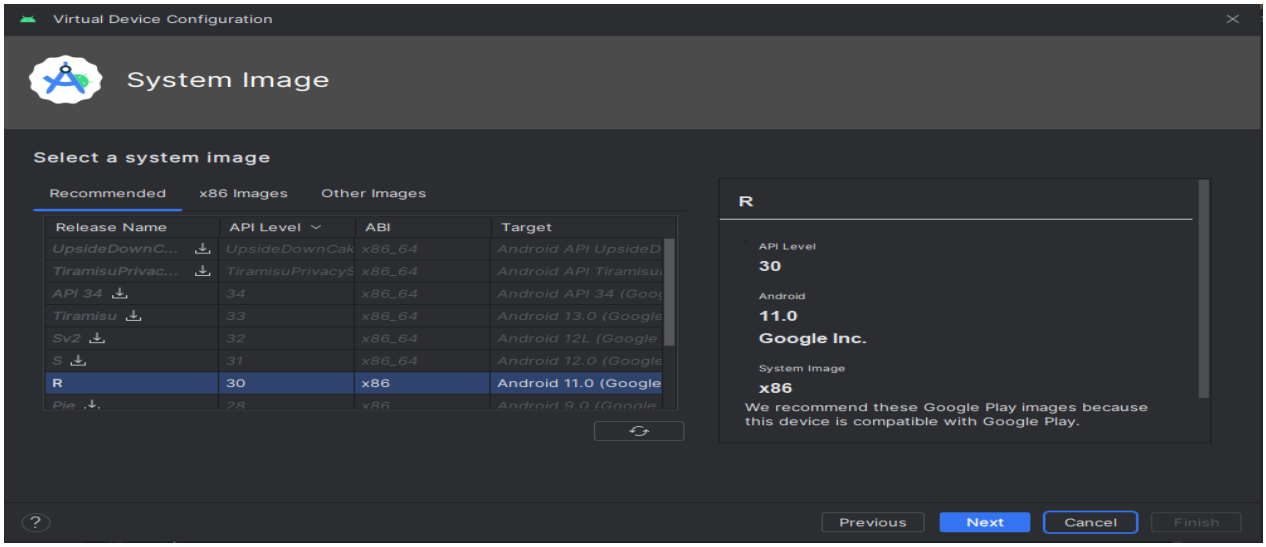


Figure 2.3:30-Android Emulator 2

Για την ολοκλήρωση της δημιουργίας της συσκευής θα πρέπει:

- να ελέγξουμε τις επιλογές μας
- να αναθέσουμε ένα όνομα
- να επιλέξουμε ένα προσανατολισμό

Τέλος πατώντας στο κουμπί “Finish” ολοκληρώνονται τα βήματα δημιουργίας της εικονικής συσκευής και ξεκινάει η διαδικασία της δημιουργίας.

Όπως φαίνεται και στην παρακάτω εικόνα η εικονική συσκευή έχει δημιουργηθεί και πλέον μπορούμε να εγκαταστήσουμε τις εφαρμογές μας σε αυτό.

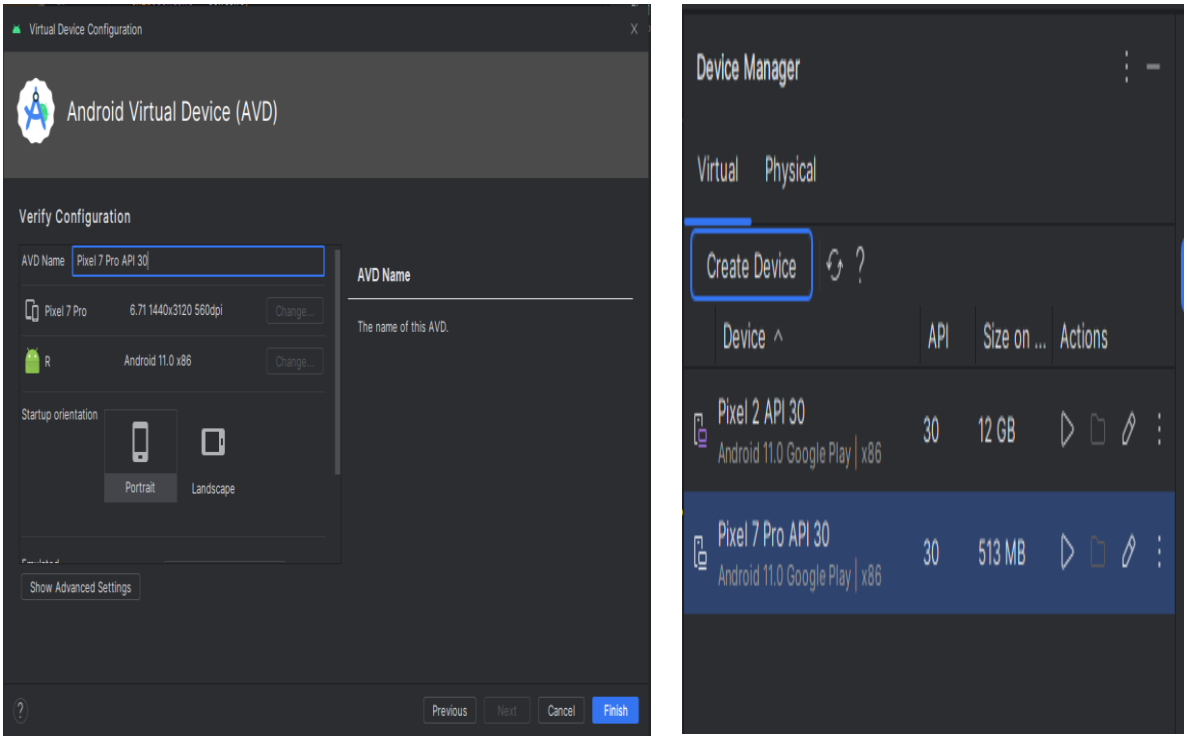


Figure 2.3:31-Android Emulator3

2.4 Java

Η Java είναι μια ευρέως χρησιμοποιούμενη γλώσσα προγραμματισμού για την υλοποίηση διαδικτυακών εφαρμογών. Είναι μια δημοφιλής επιλογή μεταξύ των προγραμματιστών για πάνω από δύο δεκαετίες, με εκατομμύρια εφαρμογές Java να έχουν δημιουργηθεί μέχρι σήμερα. Η Java είναι μια πολυ-πλατφορμική, αντικειμενοστρεφής και δικτυοκεντρική γλώσσα που μπορεί να χρησιμοποιηθεί ως πλατφόρμα από μόνη της. Είναι μια γρήγορη, ασφαλής, αξιόπιστη γλώσσα προγραμματισμού για την δημιουργία ενός μεγάλου εύρους εφαρμογών, από εφαρμογές για κινητά και εταιρικό λογισμικό μέχρι εφαρμογές για διαχείριση μεγάλου όγκο δεδομένων και τεχνολογίες από την πλευρά του εξυπηρετητή (server-side).

Η υποστήριξη αντικειμενοστραφούς προγραμματισμού δίνει το πλεονέκτημα στους προγραμματιστές να οργανώνουν τον κώδικα τους καλύτερα (επαναχρησιμοποιήσιμα objects) αποφεύγοντας την χρήση περιττού κώδικα, πράγμα που βοηθάει αρκετά στην συντήρηση μεγάλων και πολύπλοκων εφαρμογών. Η Java ανήκει στην κατηγορία μεταγλωττισμένων (compiled) γλωσσών. Στο παρακάτω σχήμα βλέπουμε τη διαδικασία μεταγλώττισης:

Figure 2.4:32-Java JVM

Ο κώδικας σε Java πρώτα μετατρέπεται σε bytecode (μια αναπαράσταση του κώδικα χαμηλού επιπέδου) ανεξαρτήτως της πλατφόρμας και έπειτα το bytecode διερμηνεύεται από την **Java Virtual Machine(JVM).** Η εικονική μηχανή αρχικά διαβάζει τις οδηγίες bytecode και στη συνέχεια τις εκτελεί. Καθώς το JVM είναι υπεύθυνο για την εκτέλεση του Bytecode, τα προγράμματα που είναι γραμμένα σε JAVA μπορούν να εκτελεστούν σε οποιαδήποτε που είναι εγκαταστημένο το JVM. Ανεξαρτήτως της πλατφόρμας που μεταγράφηκε το πρόγραμμα (Cross platform).

Ποια είναι τα πλεονεκτήματα της:

- **Απλότητα:** Εύκολη στην εκμάθηση και την κατανόηση. Το συντακτικό της είναι βασισμένο στη C++ ενώ γίνεται και χρήση αυτόματης συλλογής απορριμάτων (garbage collection), πράγμα που κάνει περιττή την ανάγκη για χειροκίνητη μετακίνηση μη αναφορούμενων αντικειμένων από τη μνήμη.
- **Φορητότητα :** Ο κώδικας Java μπορεί να τρέξει σε πολλαπλές πλατφόρμες απευθείας χωρίς να χρειάζεται να γίνει ξανά compile. Ο κώδικας bytecode είναι ανεξάρτητος πλατφόρμας.
- **Επεκτασιμότητα:** Η Java είναι επεκτάσιμη και ευέλικτη πράγμα που σημαίνει ότι μπορεί να χρησιμοποιηθεί για τη δημιουργία μεγάλου όγκου δεδομένων.
- **Ασφάλεια:** Η Java θεωρείται ασφαλής γλώσσα επειδή δε χρησιμοποιεί δείκτες (όπως η C). Επίσης τα προγράμματα της Java τρέχουν μέσα σε ένα εικονικό (virtual) περιβάλλον μηχανής (virtual machine sandbox).
- **Πολυνηματισμός:** Η Java χρησιμοποιεί ένα πολυνηματικό (multithreaded) περιβάλλον μέσα στο οποίο οι μεγαλύτερες διαδικασίες μετατρέπονται σε διάφορα νήματα που εκτελούνται ξεχωριστά. Το κύριο πλεονέκτημα του πολυνηματισμού είναι ότι δε χρειάζεται να παρέχεται μνήμη σε κάθε τρέχον νήμα.

Ποια είναι τα μειονεκτήματα της:

- **Απόδοση :** Παρότι η Java είναι μια σχετικά γρήγορη γλώσσα, σε γενικές γραμμές είναι πιο αργή από άλλες γλώσσες όπως η C και η C++ και αυτό γιατί πρέπει να διερμηνεύεται κατά τη διάρκεια της εκτέλεσης. Όπως αναφέρθηκε και προηγουμένων ο κώδικας μεταγλωτίζεται σε bytecode και έπειτα διερμηνεύεται σε εικονική μηχανή της JAVA (JVM).
- **Κατανάλωση μνήμης:** Γενικά ένα πρόγραμμα Java χρησιμοποιεί πολύ περισσότερη μνήμη αφού τρέχει πάνω στην JVM. Αυτό μπορεί να είναι μειονέκτημα για τις εφαρμογές που πρέπει να εκτελούνται σε συσκευές με μικρή μνήμη.
- **Κόστος:** Η γλώσσα μπορεί να είναι κοστοβόρα λόγω των μεγαλύτερων απαιτήσεων σε επεξεργαστική ίσχυ και μνήμη. Χρειαζόμαστε καλύτερο hardware γενικά για να τρέξουμε ένα πρόγραμμα Java.

Που χρησιμοποιείται η Java:

Η Java χρησιμοποιείται για ένα ευρύ φάσμα εφαρμογών.

Ας τις δούμε πιο αναλυτικά:

- **Κινητές εφαρμογές (Mobile Apps) :** Η Java χρησιμοποιείται σε μεγάλο βαθμό για τη δημιουργία εφαρμογών για κινητές συσκευές τύπου smartphone που έχουν λειτουργικό σύστημα Android.
- **Web Apps:** Ιδιαίτερα μεγάλη είναι η χρήση της γλώσσα Java για την δημιουργία διαδικτυακών εφαρμογών είτε πρόκειται για δημιουργία απλών ιστοσελίδων είτε πρόκειται για μεγάλες εφαρμογές που τρέχουν σε πολύπλοκα συστήματα εταιρειών.
- **Desktop Apps:** Η Java χρησιμοποιείται επιπλέον και για τη δημιουργία εφαρμογών όπως για παράδειγμα γραφικές διεπαφές (GUI). Σε αυτό το σκοπό βοηθούν ιδιαίτερα κάποιες συγκεκριμένες βιβλιοθήκες της Java όπως η Swing και η JavaFX
- **Εφαρμογές παιχνιδιών (Gaming Apps):** Παρότι δε θεωρείται ως η γλώσσα επιλογής για την ανάπτυξη εφαρμογών παιχνιδιών, η Java παρέχει αρκετές δυνατότητες μέσα από συγκεκριμένα εργαλεία και frameworks όπως είναι το LibGDX και το jMonkeyEngine.

Το λειτουργικό σύστημα Android βασίζεται σε γλώσσα προγραμματισμού JAVA. Η γλώσσα JAVA τρέχει στο JAVA VIRTUAL MACHINE στην οποία εκτελείται ο κώδικας byte code των εφαρμογών έτσι και στο Android λειτουργικό σύστημα υπάρχει η εικονική μηχανή DALVIK.

Η Dalvik είναι η μηχανή που είναι υπεύθυνη για την εκτέλεση των διάφορων εφαρμογών. Η μηχανή αυτή καθορίζει τις λειτουργίες που έχει να κάνει με τον πυρήνα του Linux , όπως η διαχείριση της μνήμης και πολυδιεργασίες.

Έτσι η εικονική μηχανή Dalvik παρέχει την δυνατότητα σε κάθε εφαρμογή να τρέχει τη δική της διεργασία ανεξάρτητα από το λειτουργικό σύστημα στο οποίο εκτελείται. Ειναι σχεδιασμένη με τέτοιο τρόπο, έτσι ώστε να είναι εφικτή η εκτέλεση πολλών εικονικών μηχανών ταυτόχρονα. Αυτό χάρη στο σχεδιασμό της με ελάχιστη χρήση της μνήμης μπορεί να εκτελείται με ευκολία.

Figure 2.4:33-JAVA

ΚΕΦΑΛΑΙΟ 3: ΕΦΑΡΜΟΓΕΣ ΔΙΑΧΕΙΡΙΣΗΣ ΚΤΗΝΙΑΤΡΕΙΟΥ

3.1 Εισαγώγή

Για να μπορούν να προσφέρουν μια ποιοτική υπηρεσία ,οι κτηνίατροι πρέπει συνεχώς να εξελίσσονται και να συμβαδίζουν με τα συνεχώς εξελισσόμενα ιατρικά πρότυπα. Αυτό σημαίνει :

- να ανακαλύψουν καινοτόμες τεχνικές
- να παρακολουθούν τις επιπτώσεις των φαρμάκων
- Συνεχής ενημέρωση για τις εξελίξεις στο κλάδο

Αυτό επιτυγχάνεται με την χρήση των νέων τεχνολογιών και ιδιαίτερα με τη χρήση των κινητών εφαρμογών. Οι διαθέσιμες κινητές εφαρμογές έχει φέρει πιο κοντά τους κτηνίατρους με τους ιδιοκτήτες των κατοικιδίων για την αντιμετώπιση των ασθενειών των κατοικιδιών τους ,αλλά και συνεχής ενημέρωση για τις διάφορες παθήσεις που έχουν τα κατοικίδια τους.

Με τη βοήθεια των συγκεκριμένων εφαρμογών μπορούμε να έχουμε μια καλύτερη εικόνα για :

- την εξέλιξη της υγείας των ζώων
- να παίρνουμε συμβουλές για την υγεία τους
- να ενημερωνόμαστε για τον τρόπο αντιμετώπισης των προβλημάτων

Σε γενικές γραμμές οι εφαρμογές αυτές παρέχουν τις ακόλουθες λειτουργίες:

- Υπολογισμός δοσολογίας φαρμάκων και παρακολούθηση της πορείας της θεραπείας
- Διάγνωση των ασθενειών και ενημέρωση του ιδιοκτήτη για τις διάφορε παθήσεις.
- Παροχή ενέσεων και ενημέρωση των ιδιοκτητών για την τήρηση των ημερομηνιών.
- Εκπαίδευση των πελατών σχετικά με τις πρώτες βοήθειες για τα κατοικίδια.

Στη συνέχεια θα παρουσιάσουμε τις ορισμένες εφαρμογές που είναι ήδη στην αγορά και θα δούμε περιληπτικά τις δυνατότητες που παρέχουν.

VETCALC

Η πρώτη εφαρμογή είναι το VetCalculator το οποίο παρέχει ένα γρήγορο και εύκολο τρόπο να γίνουν πολλοί υπολογισμοί.

Παρέχει την εκτέλεση έντεκα διαφορετικών υπολογισμών όπως:

- Υπολογισμός δόσεων φαρμάκων
- Υπολογισμός υγρών και ενεργειακών απαιτήσεων
- Μετατροπή των μονάδων και των θερμοκρασιών σύμφωνα με SI.

Η αναλυτική παρουσίαση των δυνατοτήτων και υπολογισμών που μπορεί να εκτελέσει η εφαρμογή υπάρχει στην διεύθυνση

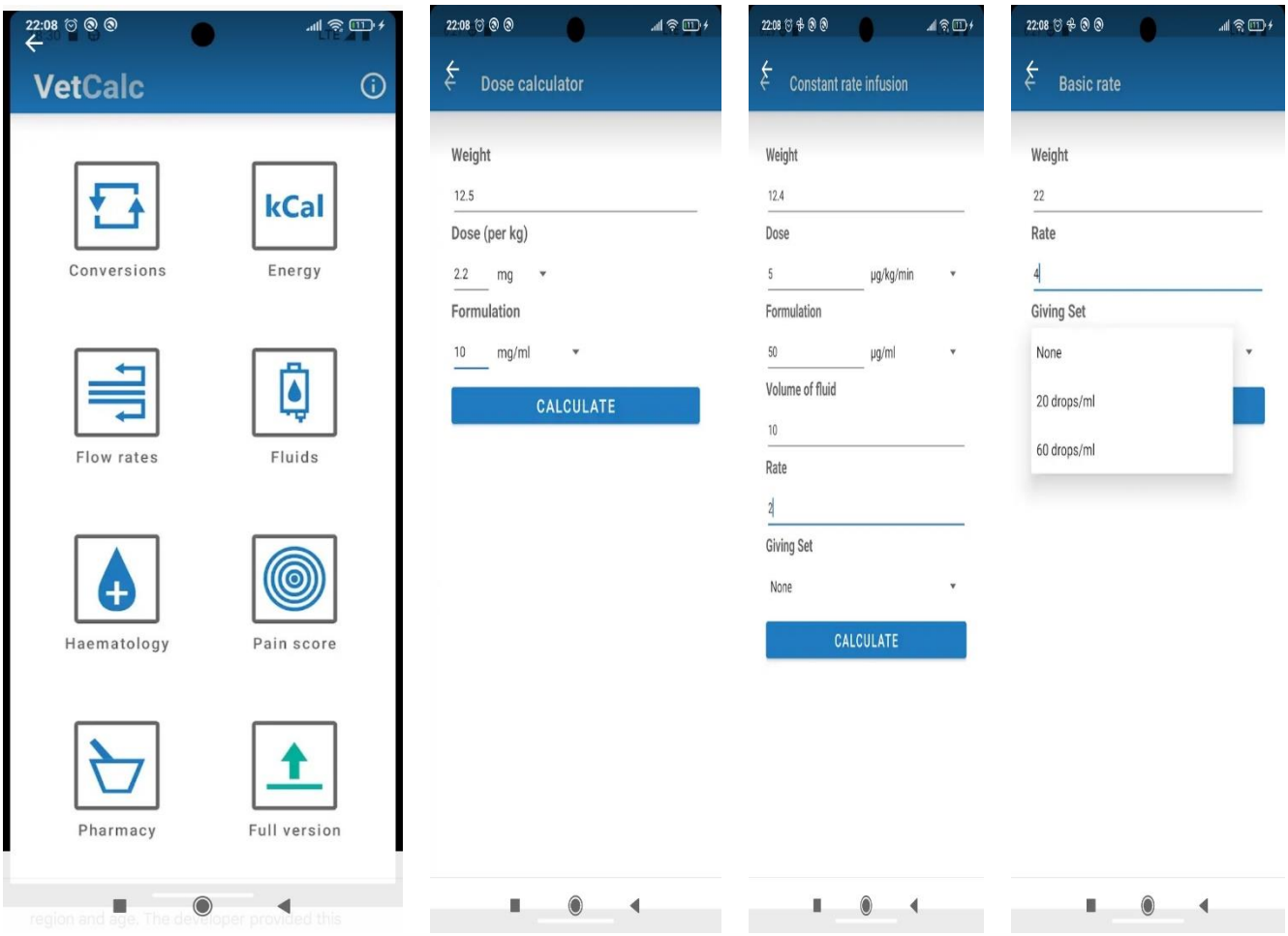


Figure 0:34-VETCALC

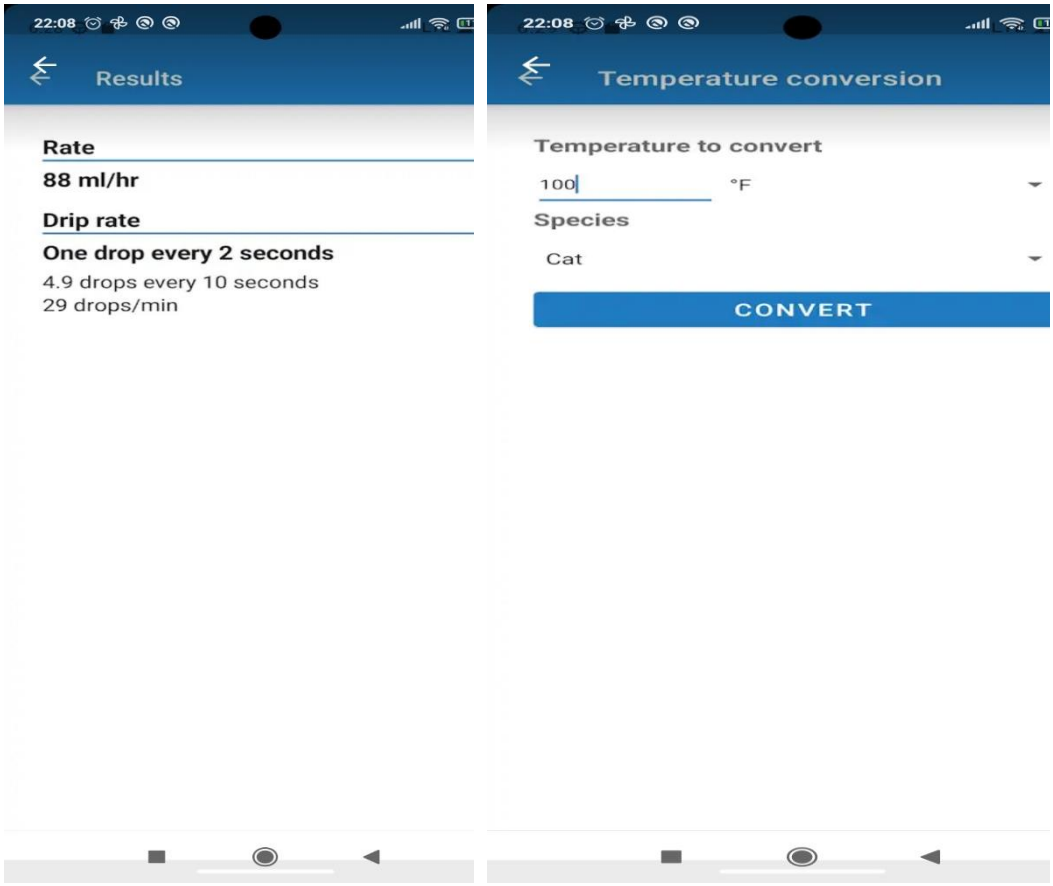


Figure 0:35-VETCALC 2

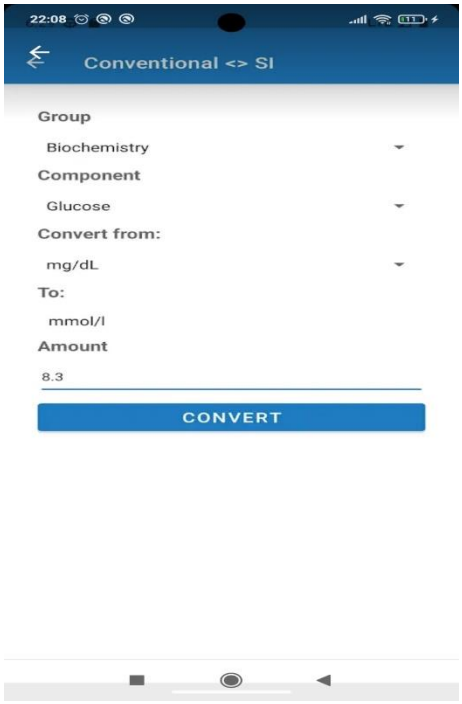


Figure 0:36-VETCALC

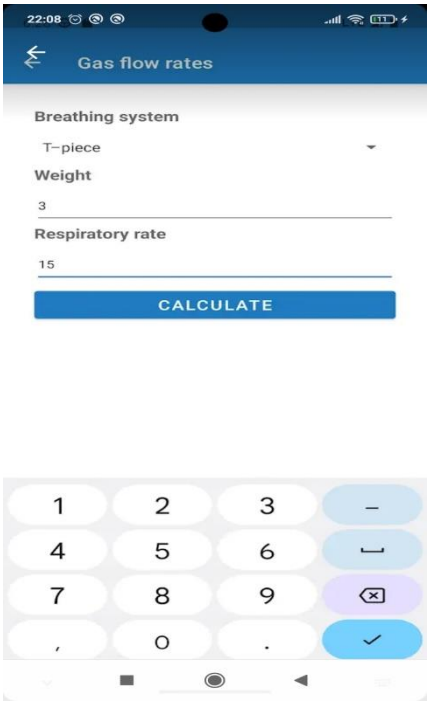


Figure 0:37-VET CALC

VETCALC+

To Vet Calculator Plus (VetCalc+) είναι ένα εύχρηστο κτηνιατρικό εργαλείο για όλους τους κτηνιάτρους και βοηθούς κτηνιάτρων αλλά και για φοιτητές. Με τη χρήση της εφαρμογής, οι υπολογισμοί για τη λήψη και δοσολογία των κτηνιατρικών φαρμάκων γίνεται πολύ εύκολη, ενώ παρέχεται δυνατίτητα αποθήκευσης και αποστολής των αποτελεσμάτων. Η εφαρμογή λειτουργεί οπουδήποτε χωρίς την ανάγκη σύνδεσης στο Διαδίκτυο. Περιέχει πάνω από 45 αριθμομηχανές (calculators) μεταξύ άλλων για: επείγοντα και αναισθητικά φάρμακα, αντιβιοτικά, κοινά κτηνιατρικά φάρμακα, ενδοφλέβια υγρά, CRI, θερμίδες, μεταγγίσεις αίματος, αρτηριακή πίεση, καμπύλες γλυκόζης, τοξικότητα σοκολάτας, υπολογισμό διαλογής Trama, μετατροπή μονάδων κ.α. Επιπλέον έχει αριθμομηχανές για:

- Αναισθησία
- μέτρηση αερίων αίματος
- παλμούς καρδιάς
- διάφορα στατιστικά
- υπολογισμό του χρόνου και της ημέρας.

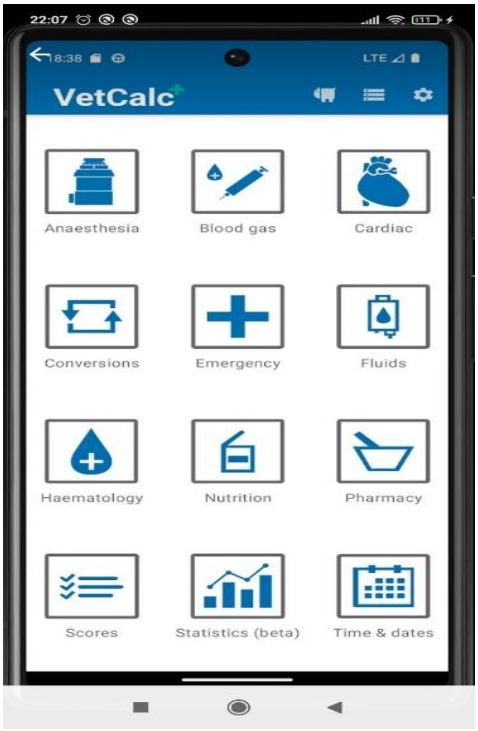


Figure 0:38-VetCalc

Figure 0:39-VetCalc

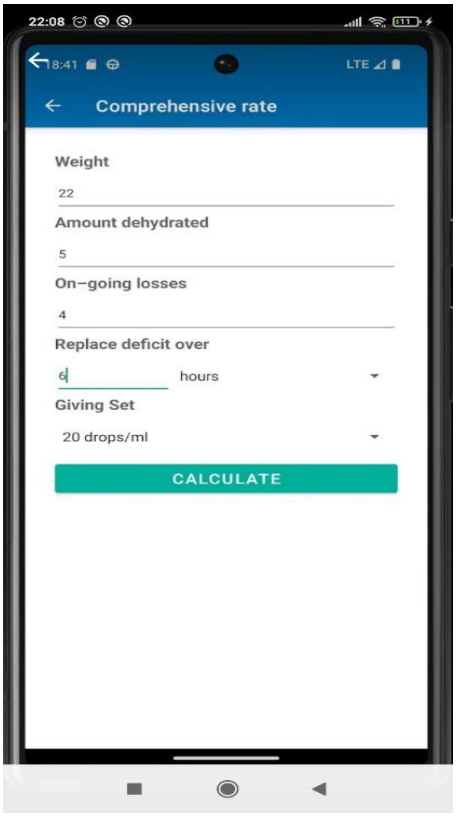


Figure 0:40-VETCALC

Figure 0:41-VETCALC

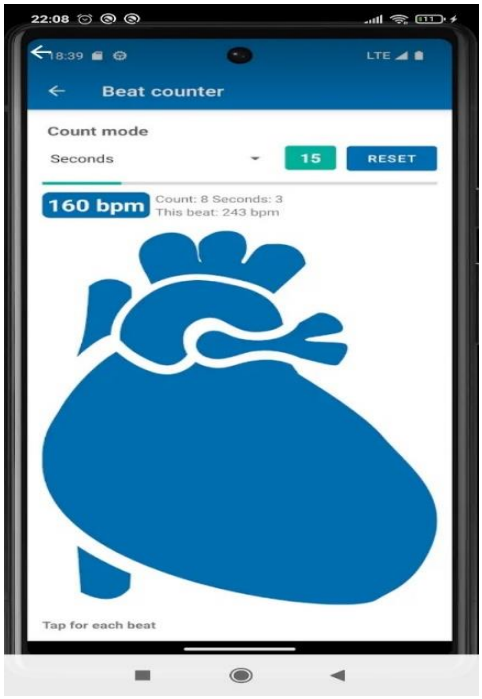


Figure 0:42-VETCALC

Figure 0:43-VETCALC

MSD Veterinary Manual

Η επόμενη εφαρμογή που θα αναφέρουμε εν συντομία, είναι το MSD Veterinary Manual. Πρόκειται για μια εφαρμογή η οποία περιέχει διάφορες πληροφορίες για την κατάσταση της υγείας των ζώων και καλύπτει μεγάλος μέρος των διαφόρων διαταραχών κτηνιατρικού ενδιαφέροντος. Η εφαρμογή παρέχει οδηγίες και πληροφορίες στους κτηνιάτρους και στους ιδιοκτήτες των αδέσποτων ζώων, σαφείς και πρακτικές εξηγήσεις για διάφορων καταστάσεων που μπορεί να προκύψουν στα σώματα των ζώων.

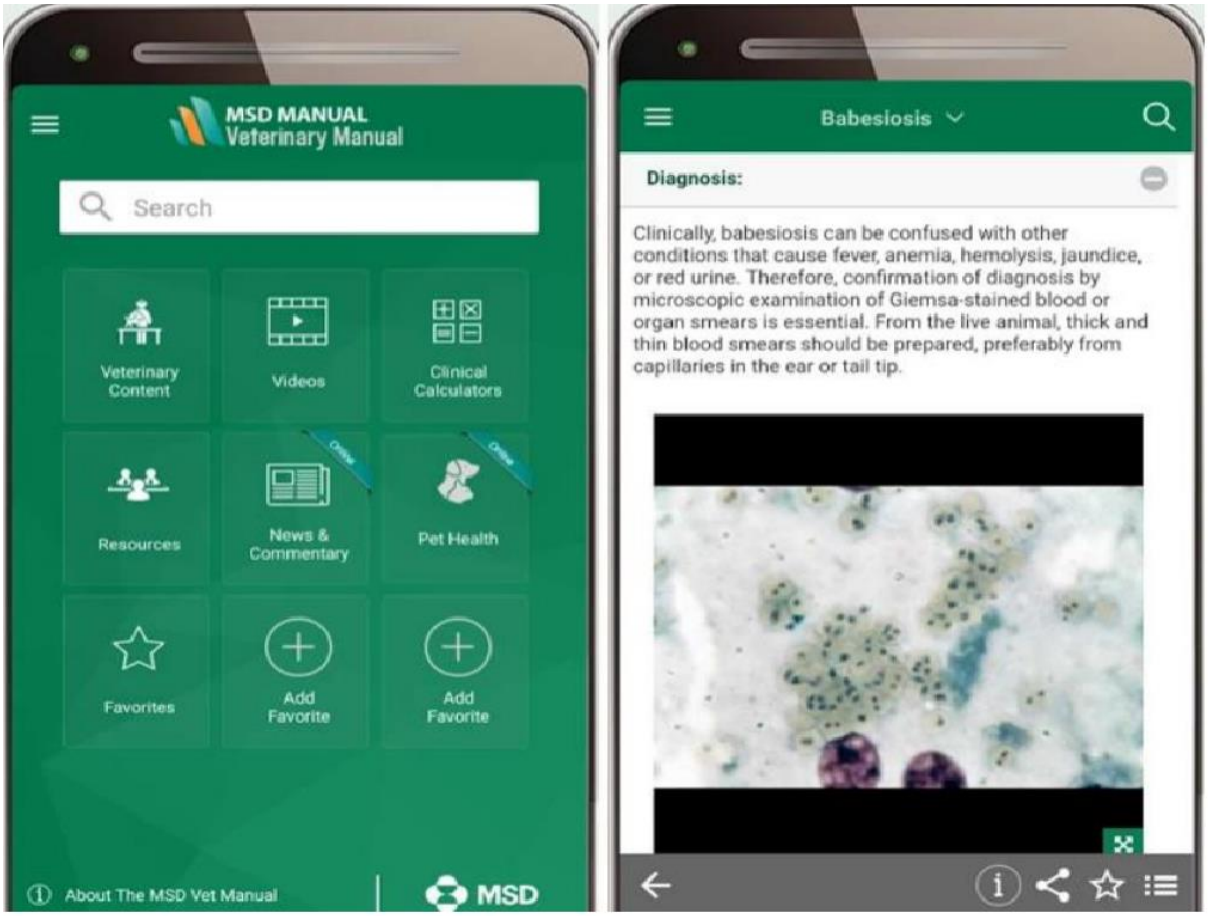


Figure 0:44-MSD APP

Οι υπηρεσίες που προσφέρει η εφαρμογή είναι:

- Χιλιάδες θέματα κτηνιατρικού ενδιαφέροντος. Όλα γράφονται και ενημερώνονται τακτικά από ειδικούς εμπειρογνώμονες (400 τον αριθμό) από πάρα πολλές χώρες.
- Οπτικό υλικό από χιλιάδες κτηνιατρικές διαταραχές
- Ερωτηματολόγιο για τον έλεγχο της γνώσης των κτηνιατρικών διαταραχών και θεραπειών.
- Παρέχει μεγάλο αριθμό οδηγών αναφοράς και εκατοντάδες από χρήσιμους πίνακες.

- Υλικό σχετικό με την υγεία των κατοικιδίων γραμμένο σε γλώσσα φιλική προς τον καταναλωτή.
- Συχνές ερωτήσεις και οδηγός χρήστη.

VitusVet

Τέλος θα μιλήσουμε για την εφαρμογή VitusVet η οποία απλοποιεί την παρακολούθηση της υγείας του κατοικίδιου σας συσσωρεύοντας όλες τις απαραίτητες πληροφορίες σε ένα μέρος. Είναι εξαιρετικό εργαλείο για όσους έχουν:

- Ένα η περισσότερα κατοικίδια
- Πολλούς παρόχους φροντίδας κατοικίδιων ζώων
- Για τους κτηνιάτρους.

Οι δυνατότητες που μας παρέχει η εφαρμογή μας είναι :

- Παρακολούθηση της κατάστασης υγείας του κατοικίδιου ζώου.
- Διαχείριση του βάρους του κατοικίδιου ζώου
- Παρακολούθηση τηε φαρμακευτικής αγωγής του κατοικίδιου ζώου.
- Άμεση πρόσβαση στις πληροφορίες του κατοικίδιου ζώου
- Άμεση πρόσβαση στο ιατρικό φάκελο του ζώου. π.χ. Ιστορικό εμβολιασμών
- Καθορισμός της λίστας των υποχρεώσεων για τη διαχείριση της υγείας των κατοικιδίων
- Καθορισμός των υπενθυμίσεων σε όλα τα απαραίτητα του ζώου (π.χ παροχή φαρμάκων η προγραμματισμός ραντεβού)
- Κλείσιμο ραντεβού με τον κτηνίατρο

Figure 0:45-VITUSVet

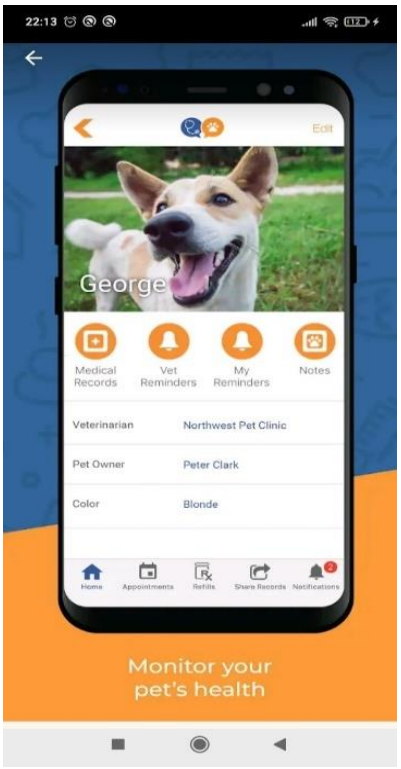


Figure 0:46-VitusVet

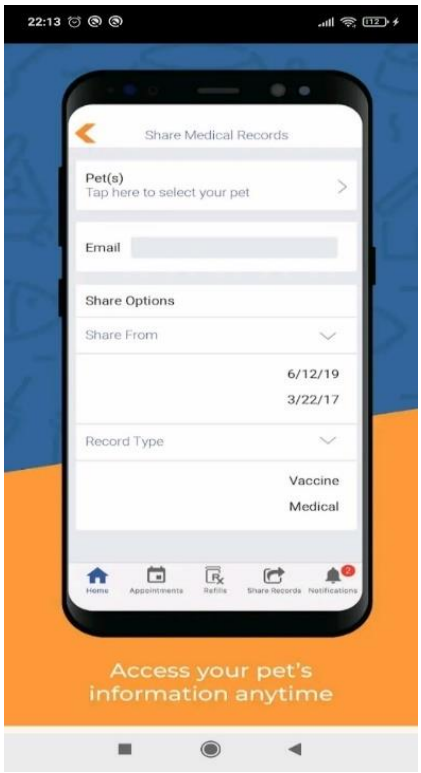


Figure 0:47-VitusVet

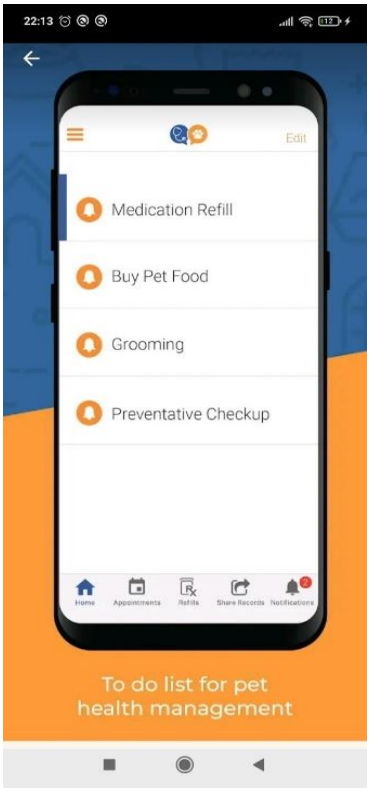


Figure 0:48-VitusVet

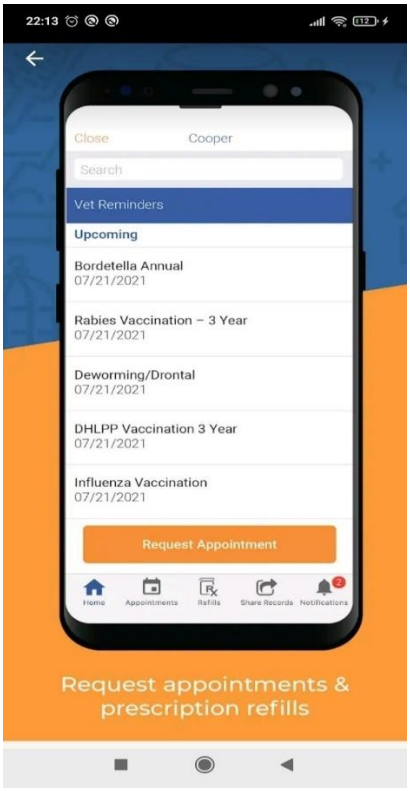


Figure 0:49-VitusVet

ΚΕΦΑΛΑΙΟ 4: ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΕΞΥΠΝΟΥ ΚΙΝΗΤΟΥ

Στο κεφάλαιο αυτό θα γίνει περαιτέρω ανάλυση και επεξήγηση των λειτουργιών της εφαρμογής. Θα γίνουν αναφορές στα κρίσιμα σημεία της υλοποίησης και με χρήση του κατάλληλου οπτικού υλικού σκοπεύουμε την καλύτερη κατανόηση από τη μεριά των χρηστών. Σκοπός της συγκεκριμένης εφαρμογής είναι η ανάλυση της χρήσης των κινητών εφαρμογών σε ένα κτηνιατρείο.

Πιο συγκεκριμένα τα ερευνητικά ερωτήματα που μπορούν να τεθούν είναι:

- Πώς αλληλεπιδρούν υπάλληλοι του κτηνιατρείου η οι ίδιοι η κτηνίατροι με τη χρήση των κινητών συσκευών και πώς οργανώνονται οι καθημερινές ανάγκες με χρήση των νέω τεχνολογιών όπως το ραντεβού με το κτηνίατρο η παρακολούθηση της θεραπείας των κατοικίδιων.
- Πώς ενσωματώνονται οι δυνατότητες των κινητών εφαρμογών σε επιμέρους γνωστικά αντικείμενα του κτηνιατρείου.
- Αν είναι αποδεκτή η χρήση των κινητών συσκευών στον τομέα της κτηνιατρικής είτε από τη μεριά των κτηνιάτρων είτε από τη μεριά των πελατών..

Στόχος μας είναι να αναδειχθούν οι αλληλεπιδράσεις ανάμεσα σε χρήστες (κτηνίατροι η πελάτες) και εργαλεία (κινητές συσκευές) όταν ασχολούνται για την υλοποίηση των θεραπειών. Επιπλέον, θα αναφερθούμε στη χρήση των οργανωτικών λειτουργιών μέσω των κινητών συσκευών με στόχο τη διατύπωση και διάγνωση των ασθενειών.
Όπως έχουμε περιγράψει και στο κεφάλαιο 2, η εφαρμογή μας διαθέτει μέσα πολλές λειτουργίες, όπως η κράτηση του ιστορικού για κάθε πελάτη και καθοδήγηση των πελατών ώστε να έχουν καλύτερα αποτελέσματα στη θεραπεία . Επίσης, διαθέτει και την δυνατότητα ανταλλαγής απόψεων και γνώσεων μεταξύ του ασθενή και εξειδικευμένου ιατρικού προσωπικού..

4.1 Σχεδιασμός

Μετά την εκκίνηση της εφαρμογής κάθε φορά εμφανίζεται η αρχική οθόνη στην οποία καλείται να εισάγει ο χρήστης της εφαρμογής εφαρμογής μας τα προσωπικά του στοιχεία όπως το e-mail και το password. Αν ο χρήστης είναι τύπου “CLIENT” τότε αφού έχουν επιβεβαιωθεί πρώτα τα στοιχεία που έχει εισάγει είναι σωστά μεταφέρεται σε μια οθόνη όπου περιέχει τις λειτουργίες όπως συνομιλία με τον εξειδικευμένο προσωπικό του κτηνιατρείου η κλείσιμο ραντεβού για μια συγκεκριμένη ημερομηνία. Από την άλλη μεριά αν ο χρήστης είναι τύπου “APP” δηλαδή εφαρμογή που απευθύνεται σε προσωπικό του κτηνιατρείου τότε πάλι ακολουθεί η διαδικασία της επαλήθευσης των στοιχείων αλλά τώρα τα περιεχόμενα της δεύτερης εφαρμογής διαφέρουν από τις λειτουργίες της πρώτης εφαρμογής. Εδώ εμφανίζονται η λίστα με τα διαθέσιμα ραντεβού της ημέρας καθώς και πάλι υπάρχει

δυνατότητα συνομιλίας με τους διαθέσιμους πελάτες τη εφαρμογής μας.

Μια γενική επεξήγηση των βασικών σημείων:

- **Model:**Τα μοντέλα που αλληλεπιδρούν με τους πίνακες της βάσης
- **Constant:** Περιέχει τα δεδομένα που καθορίζουν τον τύπο του χρήστη
- **Chat:** Περιέχει τις κατάλληλες κλάσεις και τα εργαλεία που μας παρέχει το λειτουργικό Android για την συνομιλία μεταξύ των δύο εφαρμογών
- **Appcompact:** Περιέχει τη κλάση όπου υπάρχουν οι ρυθμίσεις της εφαρμογής μας π.χ. Γλώσσα
- **Activity:** Είναι το αρχείο όπου περιέχει τις δραστηριότητες (Activities) της εφαρμογής μας.

4.1.1 User Stories

Τα Userstories είναι μια γενική περιγραφή, μια γενική εξήγηση των δυνατοτήτων του λογισμικού που γράφεται από την οπτική γωνία του τελικού χρήστη. Μέσα από τα UserStory προσπαθούμε να καθορίσουμε το πλαίσιο λειτουργίας της ομάδας που θα υλοποιηθεί.

Μια ιστορία χρήστη πρέπει να περιγράφει μια ενέργεια και μια δομή της είναι :

- **As a [persona]:** για ποιόν χρήστη θέλουμε να εκτελεστούν οι συγκεκριμένες ενέργειες.
- **I [want to]:**Οι στόχοι της εφαρμογής. Περιγράφουμε τις λειτουργίες που θέλουμε να εκτελεσθούν.
- **[So that]:** Τι θα κερδίσει ο χρήστης χρησιμοποιώντας την εφαρμογή μας.

Έχοντας υπόψη τα παραπάνω θα εξηγήσουμε τα UserStories:

- Οι χρήστες της εφαρμογής μπορούν μέσω e-mail και Password να μπορούν να συνδεθούν στην εφαρμογή.
- Ως ταυτοποιημένος χρήστης της εφαρμογής θα μπορώ να καταχωρήσω τα προσωπικά μου στοιχεία για καλύτερη ενημέρωση
- Ως ταυτοποιημένος χρήστης της εφαρμογής θα μπορώ να προσθέσω κατοικίδια στην εφαρμογή για να μπορώ να προσθέσω θεραπείες και φάρμακα .
- Ως ταυτοποιημένος χρήστης της εφαρμογής θα μπορώ να προσθέσω τις θεραπείες των ζώων και να έχω το ιστορικό τους για καλύτερη καθοδήγηση του κτηνιάτρου.
-

4.1.2 Wifrrames

Για το σχεδιασμό των Wireframes υπάρχει το ειδικό εργαλείο που λέγεται Figma. Εμείς τα wireframes τα σχεδιάσαμε σε περιβάλλον Android Studio και το υλοποιήσαμε. Παμέ τώρα μαζί με τις γραφικές διεπαφές να εξηγήσουμε και τη λειτουργικότητα που υποστηρίζουν.

Στην αρχική μας οθόνη και στις δύο εφαρμογές ζητάμε email και Password για ταυτοποίηση των χρηστών. Με το που θα πατήσει σύνδεση θα ελεγχθούν τα στοιχεία που έχει καταχωρήσει αν είναι σωστά και εφόσον ισχύουν θα μεταφερθούν στην κεντρική οθόνη ανάλογα με την εφαρμογή.

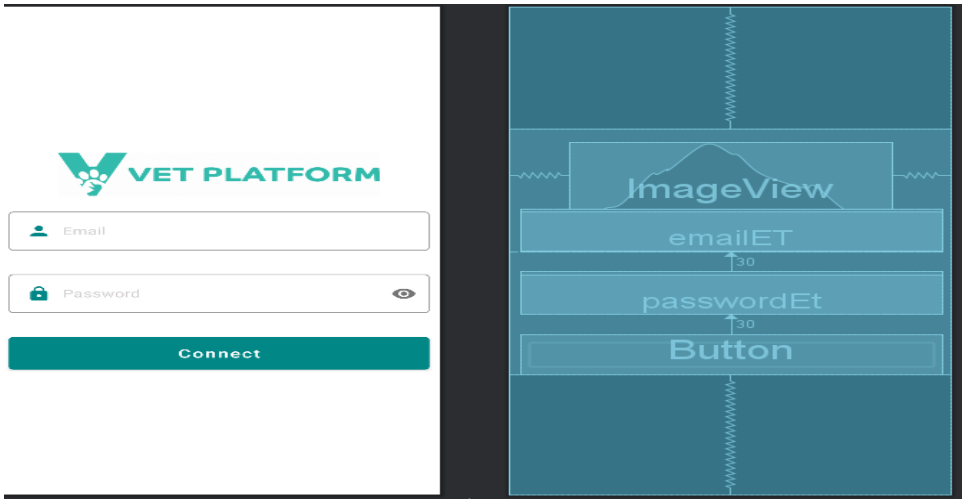


Figure 0:50-VetPlatform

Η Κύρια οθόνη της εφαρμογής "Client" είναι η επόμενη. Ο Χρήστης σε μια λίστα βλέπει τα διαθέσιμα ραντεβού που έχει κλείσει.Ο Πελάτης έχει περιορισμένα δικαιώματα στην εφαρμογή Client . Μπορεί να αλλάξει τη γλώσσα της συγκεκριμένης εφαρμογής ,να δει τα διαθέσιμα ραντεβού .

Figure 0:51-VetPlatform

Η Κύρια οθόνη της εφαρμογής "App" είναι η επόμενη. Ο Διαχειριστής(Admin) σε μια λίστα βλέπει τα διαθέσιμα ονόματα των πελατών .Υπο μορφή κάρτέλας εμφανίζονται τα προσωπικά στοιχεία του το όνομα και το email.

Ο Διαχειριστής μπορεί να :

- Ανταλλάξει μηνύματα με τον πελάτη
- Κλείσει ραντεβού στο συγκεκριμένο πελάτη
- Διαγράψει το προφίλ του συγκεκριμένου πελάτη.

- Ενημερώσει τα ήδη υπάρχοντα στοιχεία του.
- Κάνει προσθήκη κατοικιδίου στο όνομα του

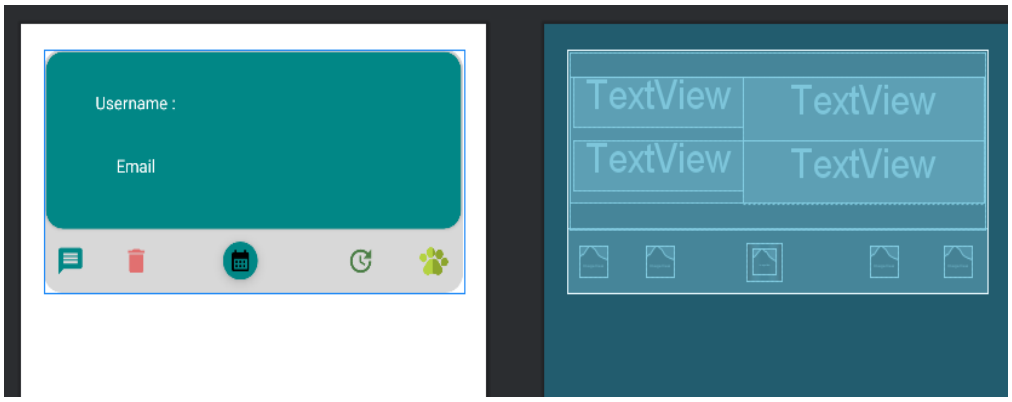


Figure 0:52-VetPlatform

Για την προσθήκη του προφίλ του πελάτη ,ο διαχειριστής θα πρέπει να εισάγει τα στοιχεία όπως:

- Ονομα
- Επίθετο
- E-mail
- Το συνθηματικό
- Το κινητό
- Το Τηλέφωνο του ατόμου.

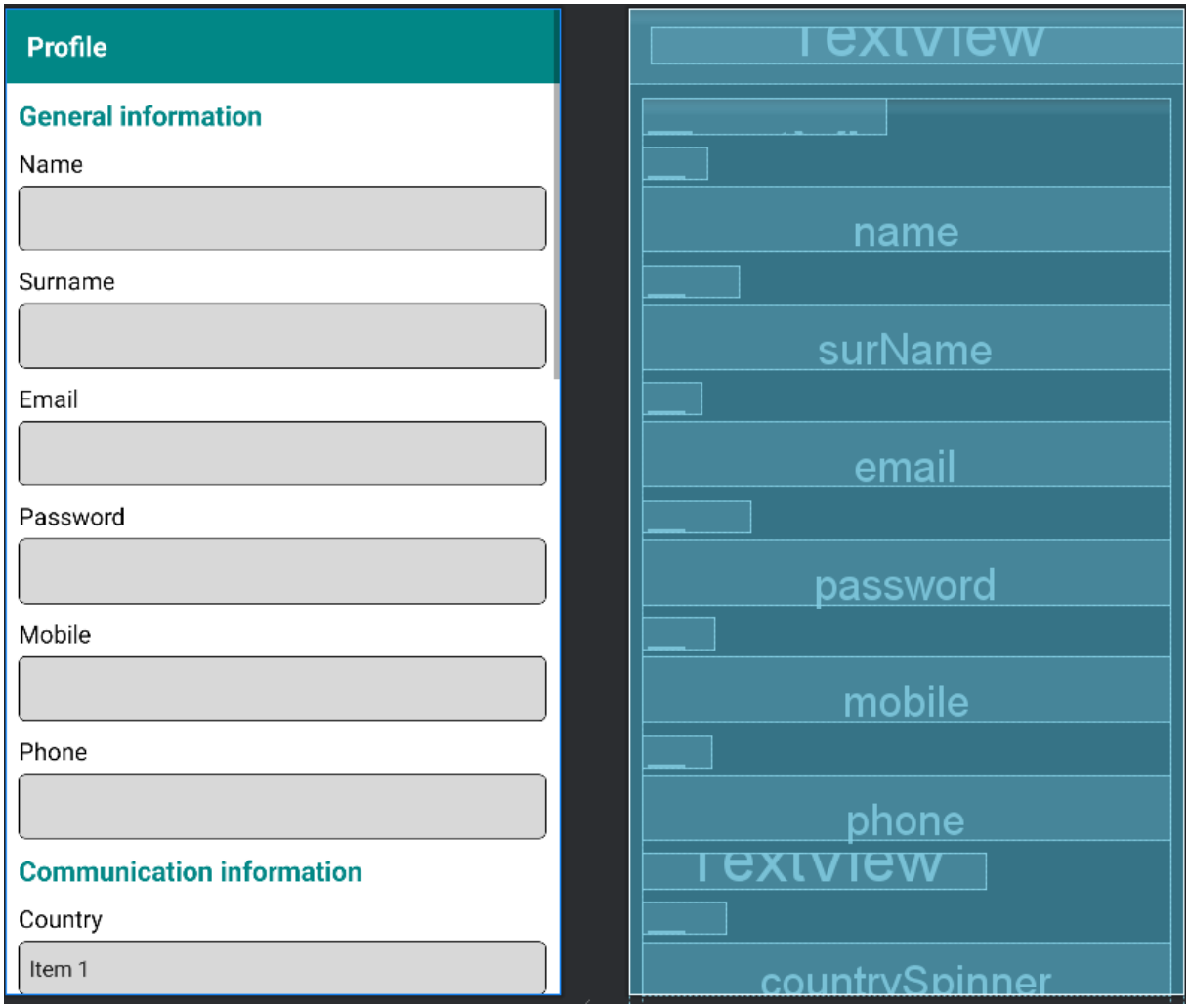


Figure 0:53Customer Profile

4.2 Η Υλοποίηση

Για την υλοποίηση της εφαρμογής θα πρέπει να ξεκινήσουμε τη διαμόρφωση Firebase. Αρχικά πρέπει να δημιουργήσουμε ένα καινούριο project όπως στην εικόνα που ακουλουθεί. Το Project το ονομάσαμε VETTAPP:

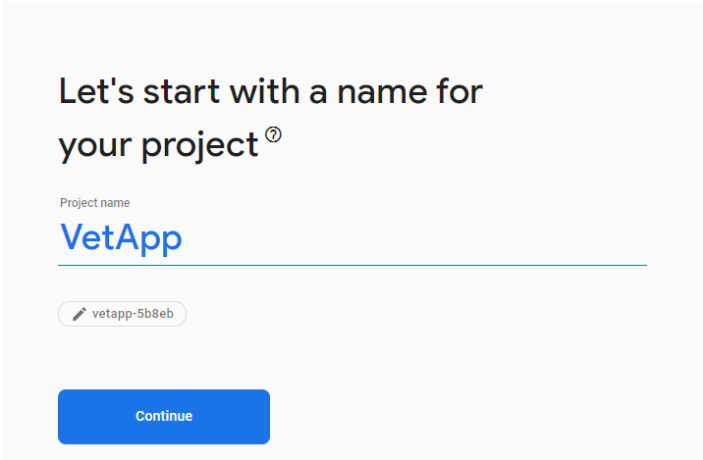


Figure 0:54-Firebase1

Το επόμενο βήμα είναι να ενεργοποιήσουμε τα Google Analytics για την εφαρμογή μας.

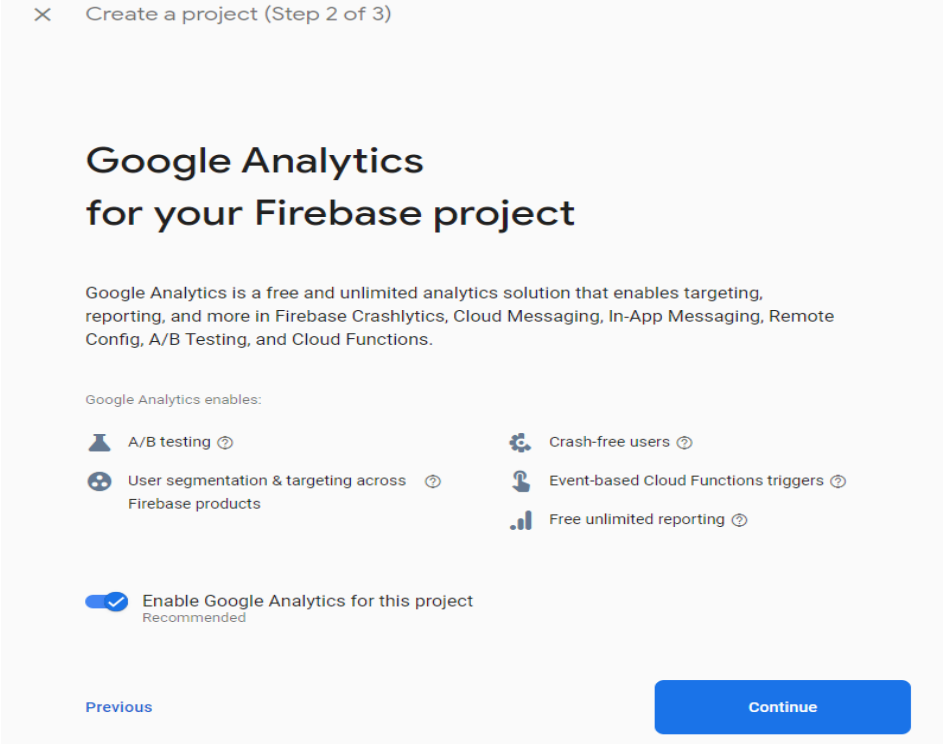


Figure 0:55-Firebase2

Το επόμενο παράθυρο μας ενημερώνει ότι έχει δημιουργηθεί το Project .

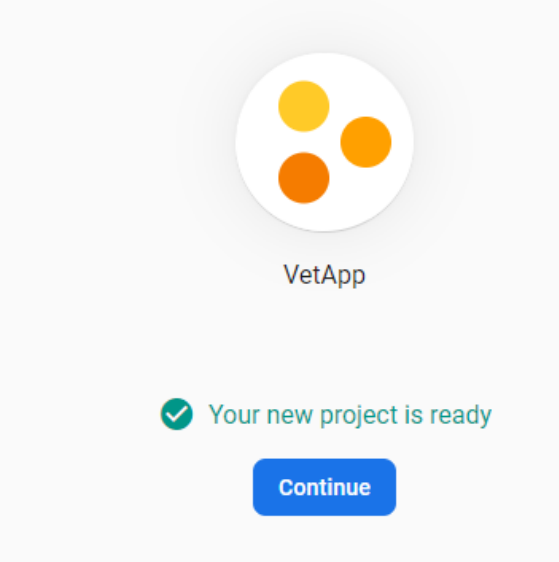


Figure 0:56-Firebase App Name

Τα παρακάτω βήματα που ακουλουθούμε είναι για να προστεθεί στο Android Studio. Κατεβάζουμε το Json αρχείο στον υπολογιστή μας και αργότερα θα προσθέσουμε στο κατάλληλο πεδίο του Android Studio.

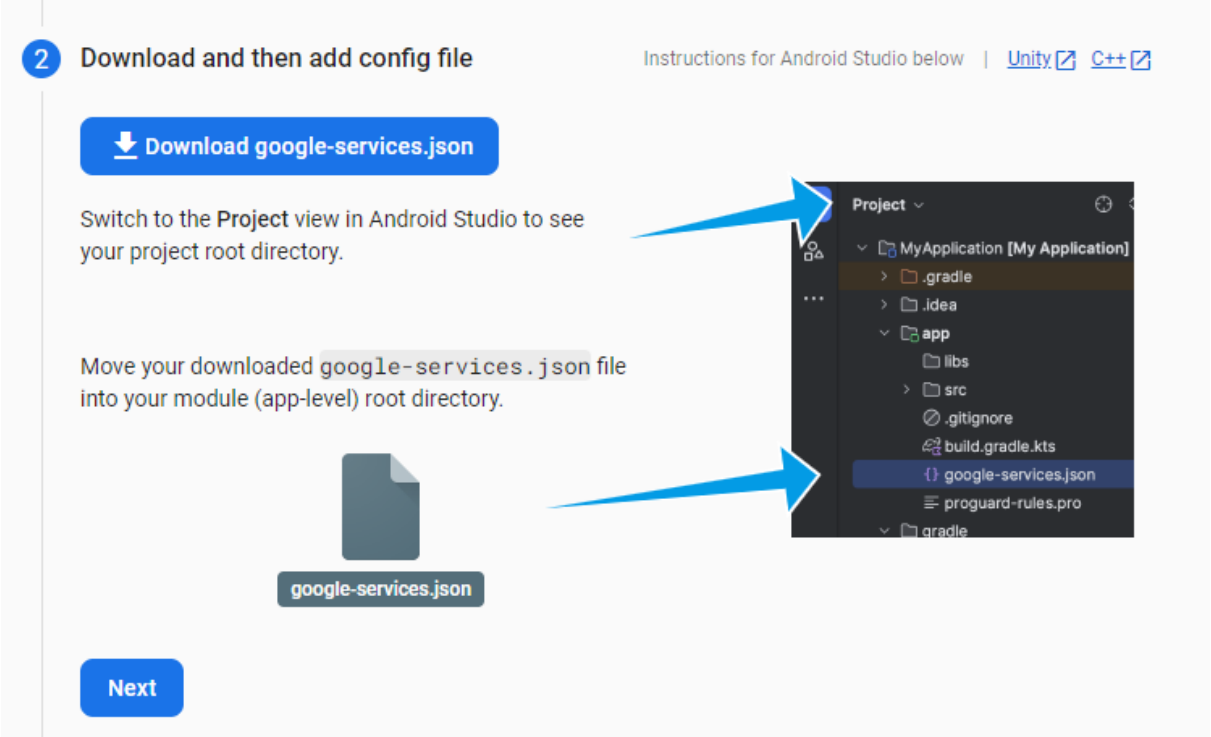


Figure 0:57-Firebase Json File

Στο σημείο αυτό εφόσον έχουμε δημιουργήσει το κατάλληλο πρότζεκ. πάμε να δημιουργήσουμε την εργασία στο περιβάλλον Android studio.

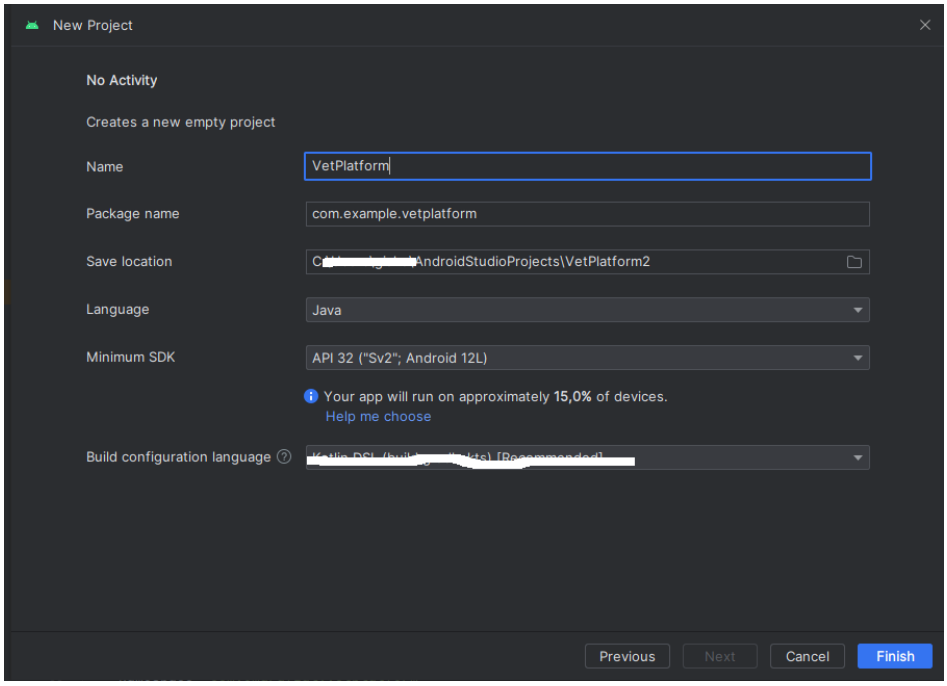


Figure 0:58-Android Emulator Name

Στο σημείο αυτό συμπληρώνουμε το όνομα VetPlatform. Επίσης επιλέγουμε τη γλώσσα προγραμματισμού να είναι JAVA και minimumSDK να είναι API32 δηλαδή Android 12. Πατώντας στο κουμπί Ολοκλήρωση(Finish) θα δημιουργηθεί το project μας.

Αφού έχουμε ολοκληρώσει την δημιουργία του project από τη μεριά του Android Studio πάμε τώρα να τροποποιήσουμε το Firebase προσθέτοντας λειτουργίες π.χ. όπως ταυτοποίηση του χρήστη.

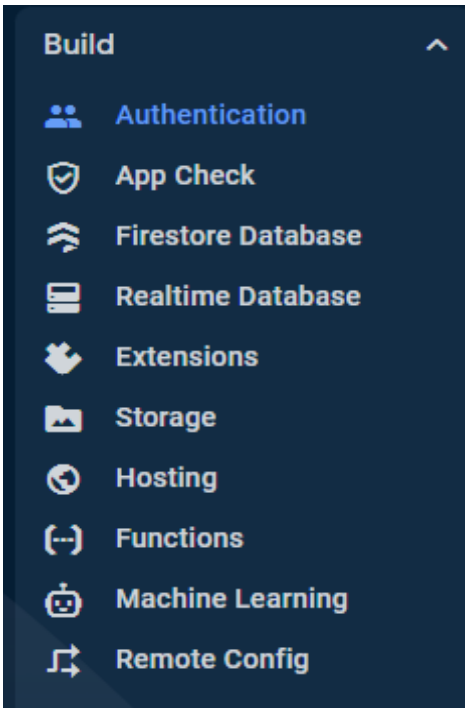


Figure 0:59-Firebase List

Επιλέγουμε Authentication-> Sign-in Method εμφανίζεται η εικόνα παρακάτω ,όπου μπορούμε να καθορίσουμε τον τρόπο ταυτοποίησης των χρηστών μέσω Firebase. Επιλέγουμε το κουτάκι Email/Password.

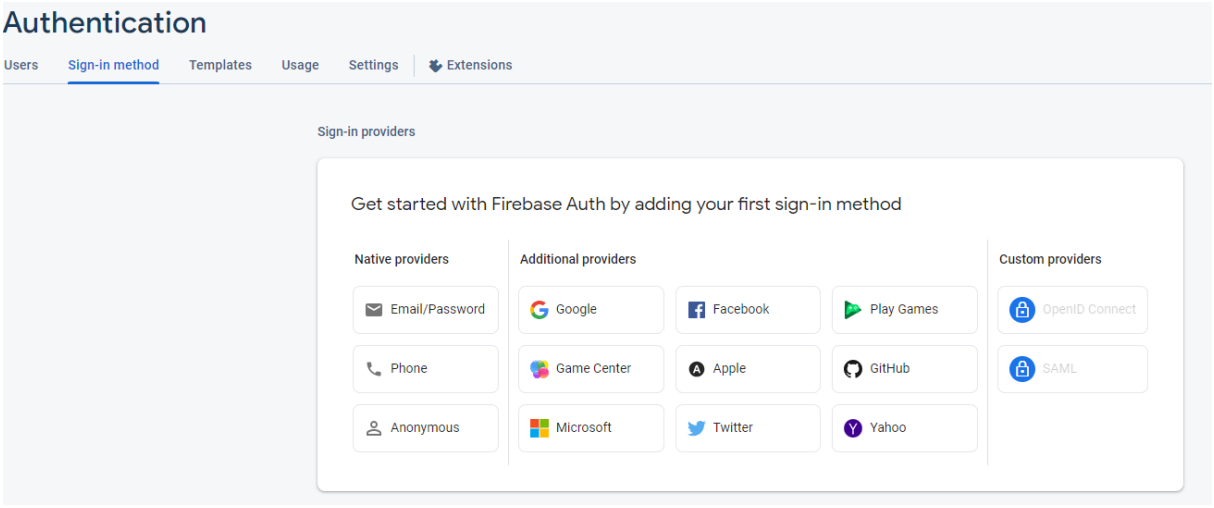


Figure 0:60-Firebase Authentication Method

Επιλέγουμε η ταυτοποίηση των χρηστών να γίνουν με E-mail και Password. Ενεργοποιούμε το πρώτο πεδίο και δεν ενεργοποιούμε το δεύτερο δηλαδή το EmailLink ,που σημάινει ότι οι χρήστες δε θα συμπληρώσουν κάποιο password Προχωράμε στη υλοποίηση της εφαρμογής μας και δημιουργούμε τα κατάλληλα Activities.

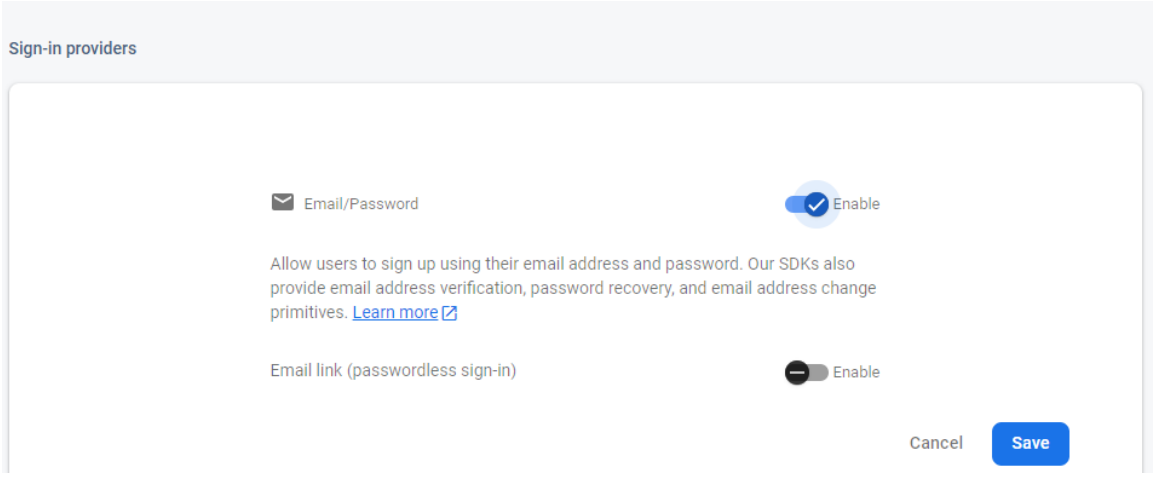


Figure 0:61-Firebase Email Authentication

4.3 Η εγκατάσταση

Για να εγκαταστήσουμε την εφαρμογή μας στο κινητό μας τηλέφωνο αρκεί να μεταφέρουμε το .apk αρχείο της εφαρμογής μας στη συσκευή μας που υπάρχει στο φάκελο ..\app]\build\outputs\apk\debug στον υπολογιστή μας. Βέβαια υπάρχουν και άλλοι τρόποι εγκατάστασης της εφαρμογής μας στη συσκευή μας. Αυτό γίνεται είτε με κατευθείαν εγκατάσταση της εφαρμογής συνδέοντας τη συσκευή μας στον υπολογιστή είτε ανεβάζοντας την εφαρμογή μας στο Google apps.

Τα βήματα για τη δημιουργία .apk αρχείο είναι τα εξής:

Βήμα1:Επιλέγουμε Menu->Build Bundle(s)/APK->Build APK

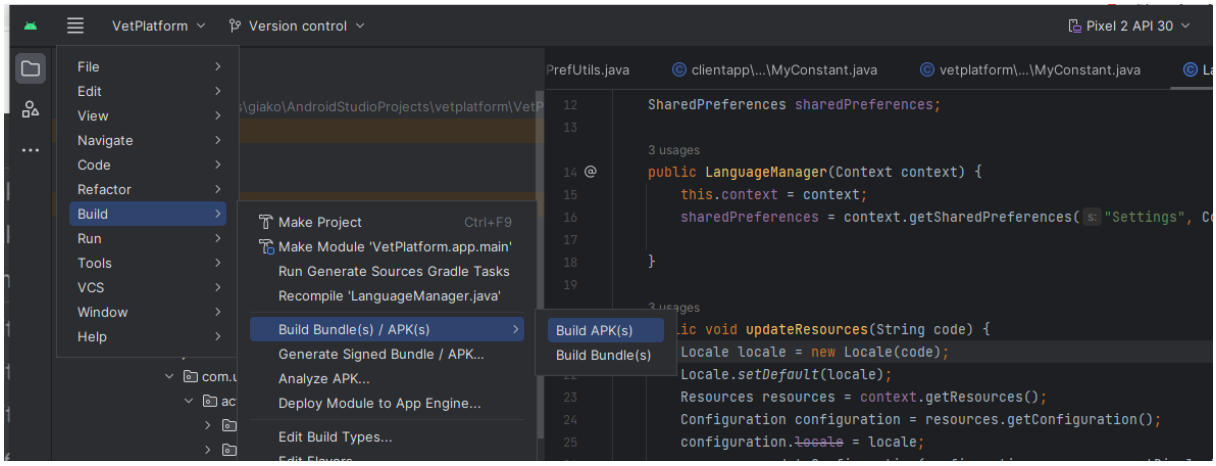


Figure 0:62-Build Apk File

Βήμα2: Εφόσον έχει ολοκληρωθεί η διαδικασία της δημιουργία του .apk αρχείου στο κάτω μέρος του γραφικού περιβάλλοντος εμφανίζεται το παράθυρο όπου δηλώνει την τοποθεσία του αρχείο στον υπολογιστή μας.

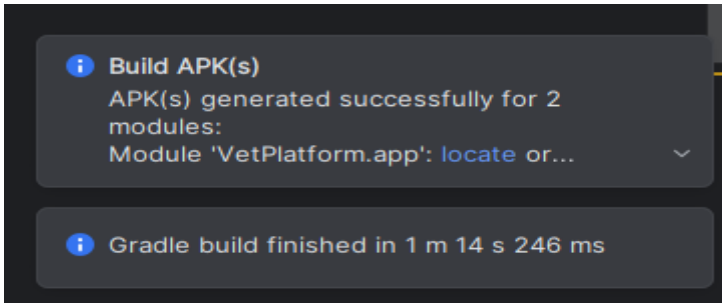


Figure 0:63-Apk Folder Path

Βήμα 3: Πατάμε στη λέξη locate και πηγαίνουμε στο φάκελο όπου βρίσκεται το .apk αρχείο. Ετσί η εφαρμογή μας είναι σε θέση να μπορεί να τρέξει σε οποιαδήποτε συσκευή που είναι συμβατή με την έκδοση της εφαρμογής μας.

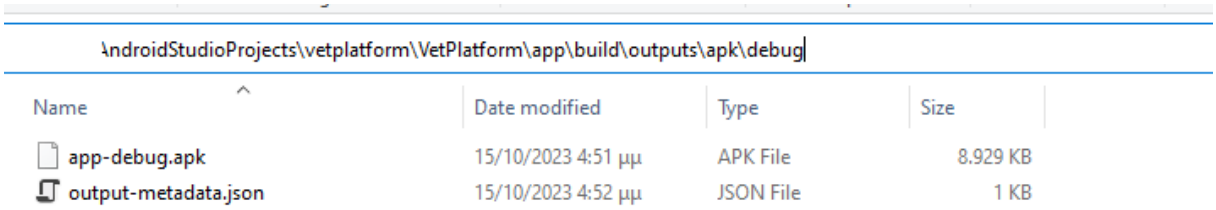


Figure 0:64-Folder data

4.4 Η Χρήση της εφαρμογής

Η εφαρμογή μας αποτελείται από δύο μέρη :

- Το πρώτο μέρος αφορά το κομμάτι του ιδιοκτήτη του κατοικιδίου.
- Το δεύτερο μέρος αφορά το κομμάτι του κτηνιατρείου.

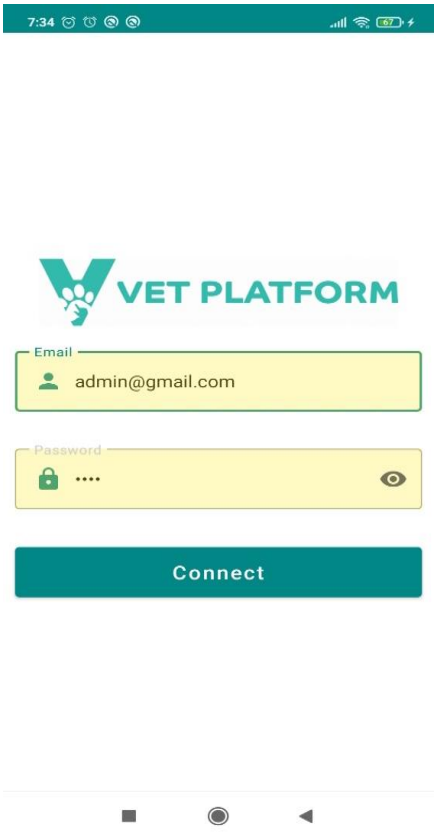


Figure 0:65-User Authentication

Figure 0:66-User Error Check

Όπως φαίνεται και στην παραπάνω εικόνα έχουμε τις εξής επιλογές:

- Σύνδεση (Connect) : Ο χρήστης εισάγοντας το κατάλληλο email και το Password γίνεται ταυτοποίηση των στοιχείων και αν όλα είναι εντάξει μεταβαίνει στην κεντρική οθόνη (Dashboard)

  Και στις δύο εφαρμογές η φόρμα ταυτοποίησης του χρήστη (είτε είναι Admin είτε Πελάτης) είναι τα ίδια. Απλά στη μία εφαρμογή βάζουμε το e-mail του κτηνίατρου ενώ στο άλλο τα στοιχεία του πελάτη.

Αν προκύψει κάποιο πρόβλημα (αν δεν είναι σωστό το email ή αν δεν έχει συμπληρώσει ο χρήστης το πεδίο email) τότε εμφανίζεται το κατάλληλο μήνυμα σφάλματος.

Ξεκινάμε με τη πρώτη εφαρμογή:

Μετά την ταυτοποίηση των προσωπικών στοιχείων ο Πελάτης (ο ιδιοκτήτης του κατοικιδίου) μεταφέρεται στην παρακάτω οθόνη και εχει τις εξής παρακάτω επιλογές.

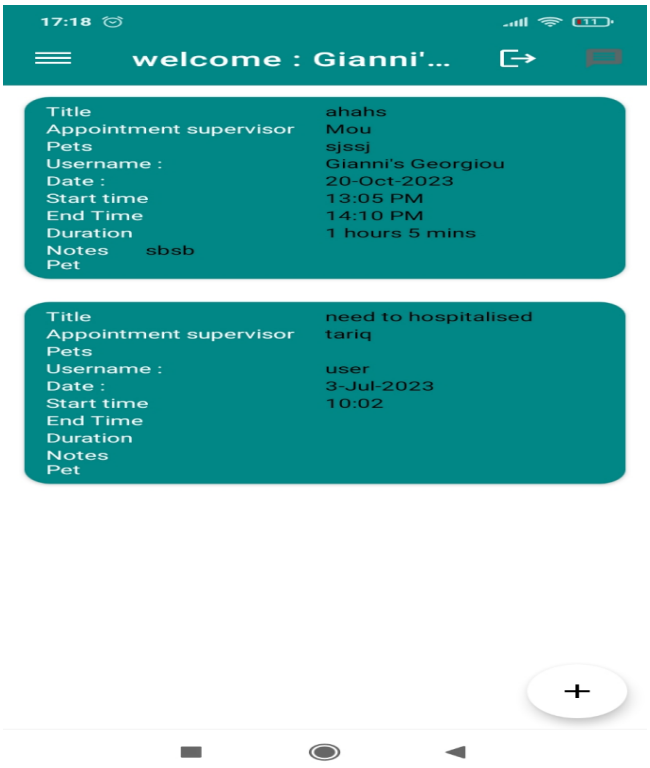


Figure :67-Client App Dashboard

1) Πατώντας στο εικονίδιο μενού στην αριστερή γωνία επάνω εμφανίζεται το συρόμενο Μενού.
2) Το αμέσως επόμενο στοιχείο εμφανίζει το όνομα του χρήστη που έχει εισέλθει στην εφαρμογή μας.
3) Η Τρίτη επιλογή με το βελάκι για να κάνει ο χρήστης έξοδο από την εφαρμογή.
4) Η τελευταία επιλογή στα ξεδία είναι η φόρμα συνομιλίας με τον κτηνίατρο.

Στην οθόνη αυτή εμφανίζονται τα διάφορα ραντεβού που έχει ο κλείσει ο πελάτης. Για τις διαθέσιμες μέρες και ώρες ο πελάτης υποχρεωτικά πρέπει να κανονίσει με τον γραμματέα του κτηνιατρείου. Τα διαθέσιμα ραντεβού του χρήστη εμφανίζονται σε μορφή λίστας στην κεντρική οθόνη της εφαρμογής μας. Ο πελάτης έχει δικαίωμα μόνο για ανάγνωση και δεν έχει δικαίωμα για διαγραφή η τροποποίηση.

Στον χρήστη της εφαρμογής Client App εμφανίζεται τα στοιχεία του ραντεβού που έχει κλείσει.

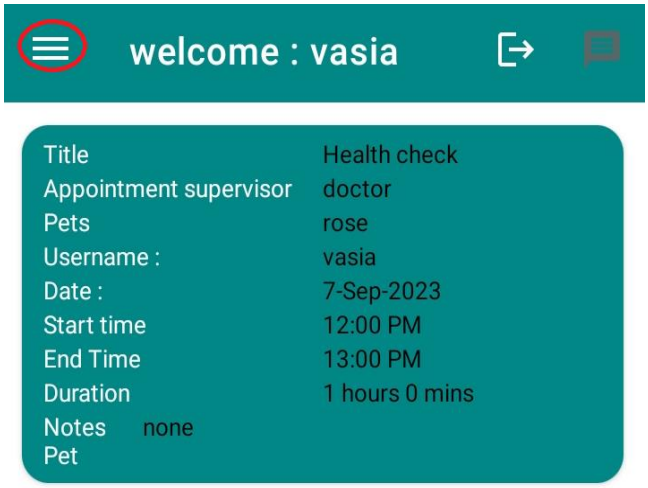


Figure 0:68-App Menu

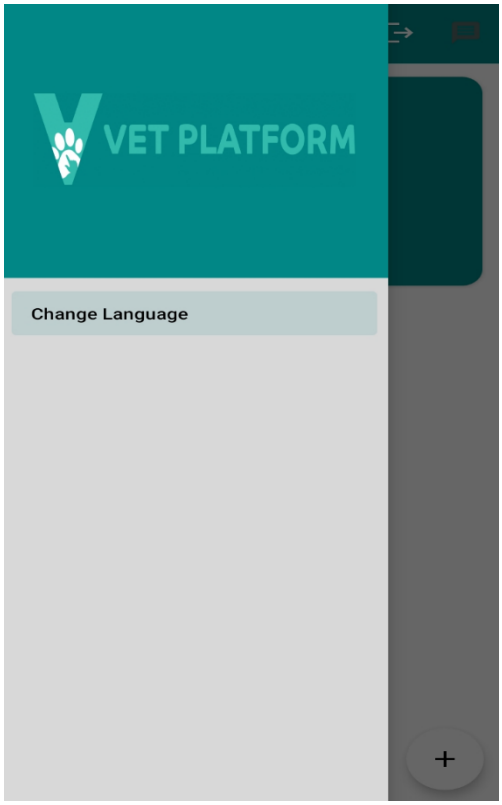


Figure 0:69-App Menu Choices

Από το μενού επιλογών μπορεί να αλλάξει την γλώσσα της εφαρμογής μας.Πατώντας την επιλογή αριστέρά εμφανίζεται η εικόνα που βλέπετε παρακάτω.Εχουμε δύο γλώσσες από επιλογές.Η πρώτη είναι τα Αγγλικά και η δεύτερη είναι τα Ελληνικά.

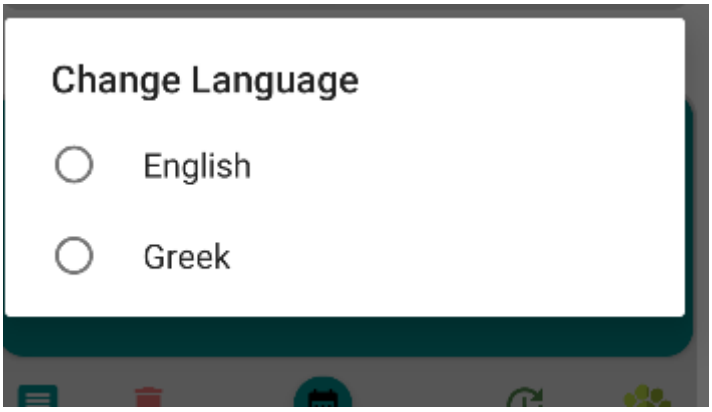


Figure 0:70-Select Language

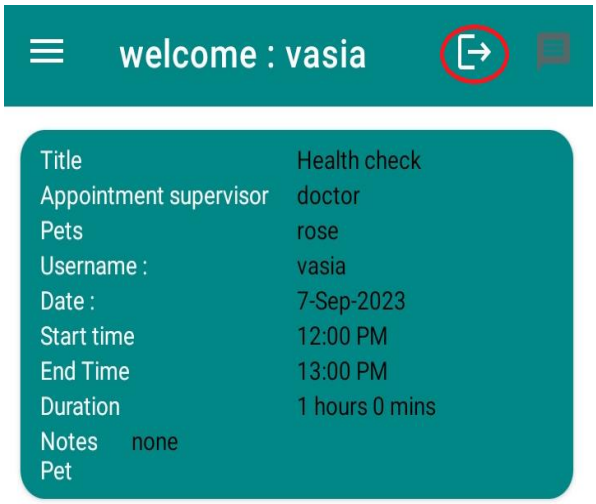


Figure 0:71-Client App logout Icon1

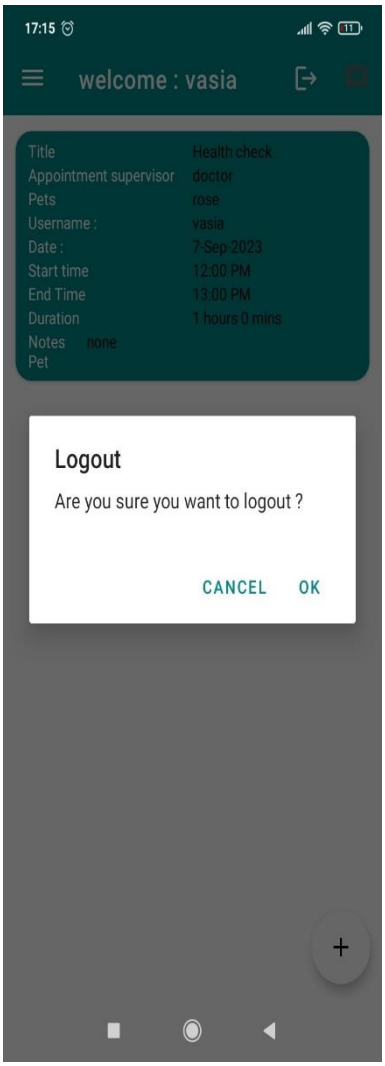


Figure 0:72-Client App logout Icon2

Ο Χρήστης της εφαρμογής μας πατώντας στο εικονίδιο που εμφανίζεται στην παραπάνω εικόνα μπορεί να κάνει έξοδο από την εφαρμογή. Εφόσον έχει πατήσει το

εικονίδιο για έξοδο θα εμφανιστεί το παρακάτω μήνυμα και πατώντας στην επιλογή ‘OK’ θα ολοκληρωθεί η διαδικασία εξόδου.

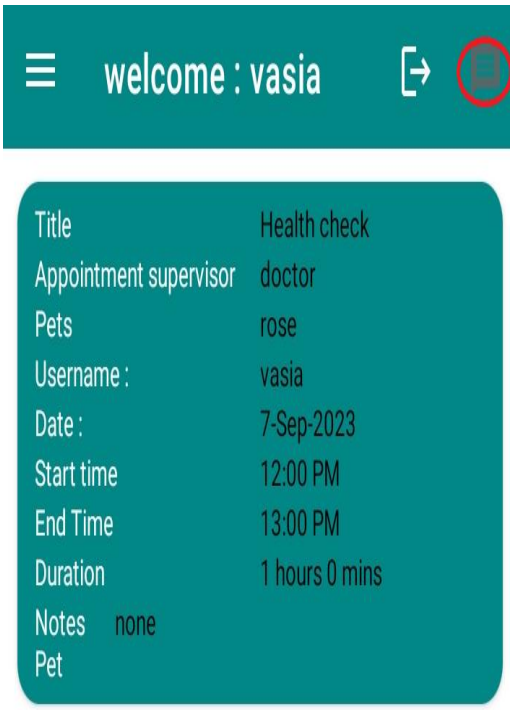


Figure 0:73- Chat Icon

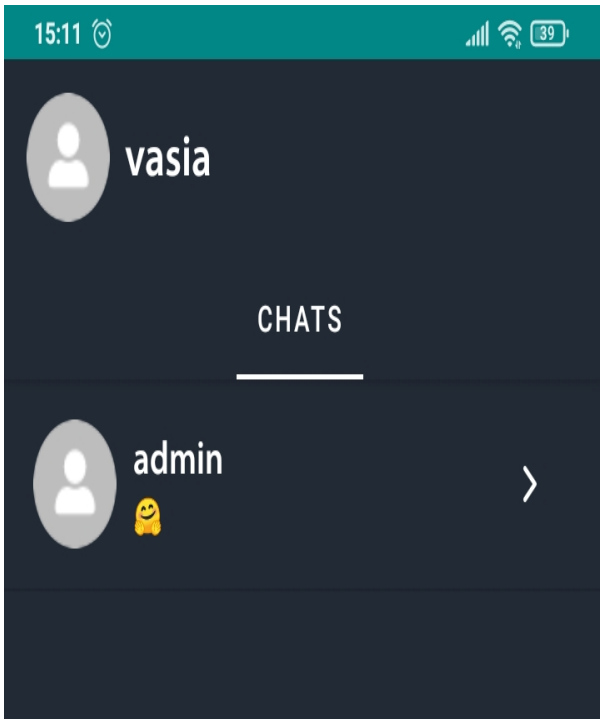


Figure 0:74-Chat User List

Εικόνα 4.8: Client App Chat Icon

Εδώ εμφανίζεται η λίστα των συνομιλιών με το εξειδικευμένο προσωπικό του κτηνιατρείου.Επιλέγοντας το εικονίδιο για ‘Chat’ ο χρήστης της εφαρμογής μας θα έχει την κατάλληλη ενημέρωση, καθοδήγηση από τον εξειδικευμένο προσωπικού του κτηνιατρείου για τις διάφορες απορίες που έχει είτε σχετικά με την πορεία της θεραπείας είτε για γενικές πληροφορίες.

Γενικά αν είναι να συνοψίσουμε την εφαρμογή Client,ο πελάτης έχει περιορισμένο εύρος λειτουργιών. Τα κύρια πράγματα που μπορεί να κάνει είναι να βλέπει τα δεδομένα που έχει προσθέσει ο Admin όπως και να ανταλλάζει μηνύματα με τον εξειδικευμένο προσωπικό του κτηνιατρείου.

Πάμε τώρα να δούμε την δεύτερη εφαρμογη (δεύτερο μέρος):

Μετά την ταυτοποίηση του χρήστη της εφαρμογής του κτηνιατρείου πρωτού να μεταβεί στην κεντρική οθόνη της εφαρμογής μας ζητάει τα δύο δικαιώματα από τον Διαχειριστή(Admin).

Ζητάμε την άδεια του χρήστη για να μπορεί να έχει πρόσβαση η εφαρμογή μας να μπορεί να κάνει read and write.

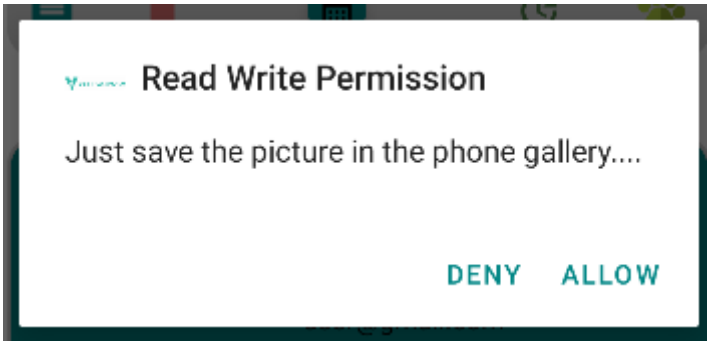


Figure 0:75-Android Permision

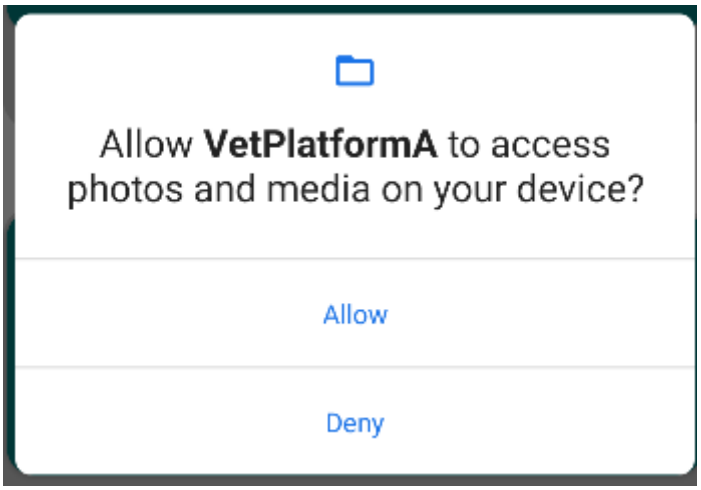


Figure 0:76-Android Permision 2

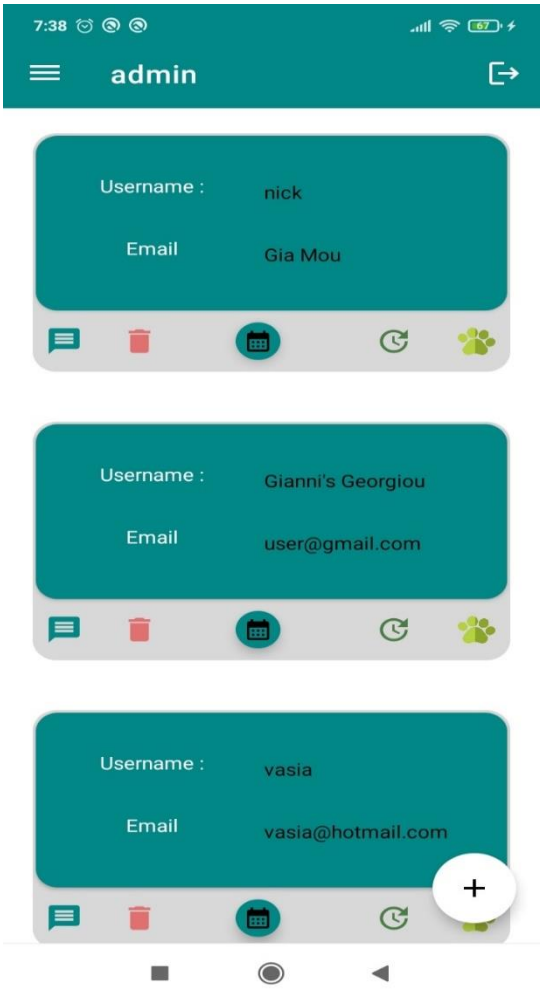


Figure 0:77-Android App Dashboard

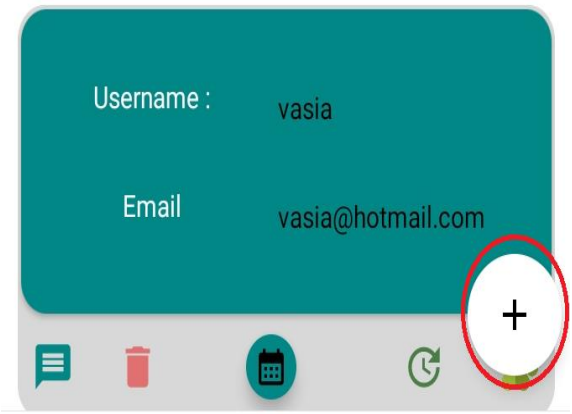


Figure 0:78-User Details

Admin App Dashboard

Από την κεντρική οθόνη της εφαρμογής App (Εφαρμογή που χρησιμοποιεί ο κτηνίατρος) πατώντας στο εικονίδιο άθροισμα μπορούμε να κάνουμε την προσθήκη του προφίλ του ιδιοκτήτη του αδέσποτου ζώου. Συμπληρώνοντας τα κατάλληλα πεδία της φόρμας που εμφανίζεται στις παρακάτω εικόνες κάνουμε την καταχώρηση του πελάτη.

Στην κεντρική οθόνη της εφαρμογής σε μορφή λίστας εμφανίζονται τα διαθέσιμα ονόματα των εγγεγραμμένων χρηστών της εφαρμογής μας. Επιλέγοντας την καρτέλα του χρήστη μπορούμε να εκτελέσουμε της εξής λειτουργίες:

- Ανταλλαγή μηνυμάτων με με τον χρήστη
- Διαγραφή του προφίλ του ατόμου.
- Κλείσιμο ραντεβού
- Ενημέρωση των προσωπικών στοιχείων που έχουν καταχωρηθεί στο σύστημα.

- Η τελευταία λειτουργία είναι η προσθήκη του κατοικίδιου

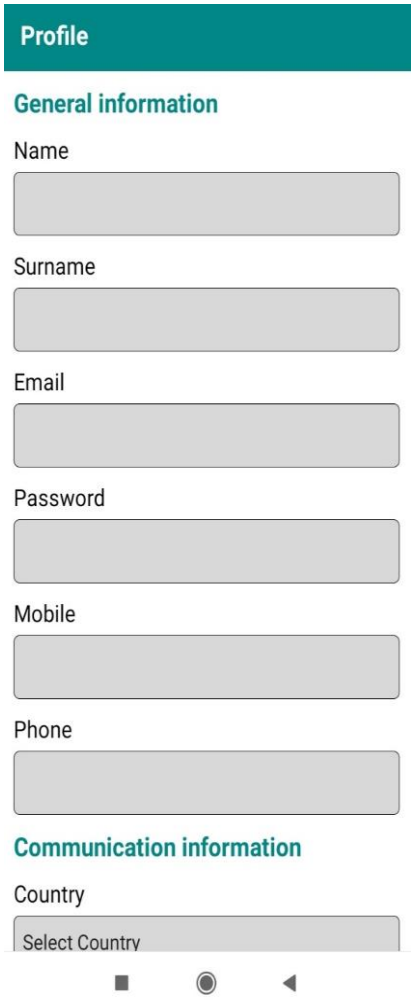


Figure 0:79-Profile 1

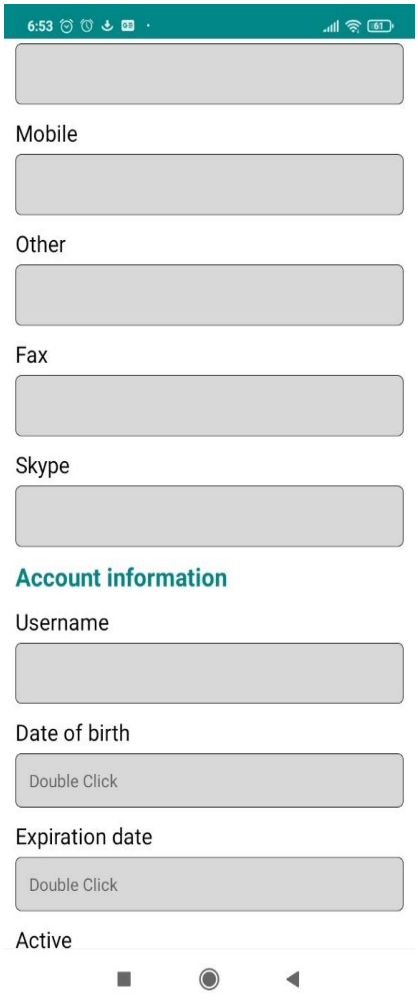


Figure 0:80-Profile 2

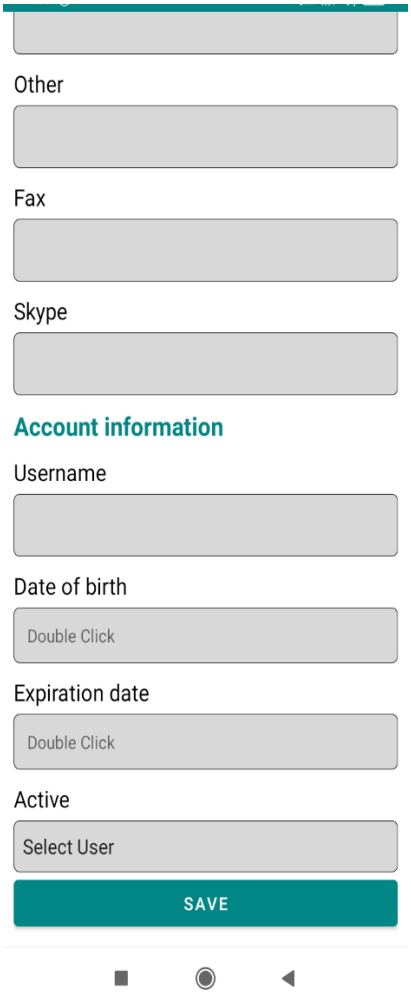


Figure 0:81-Profile 3

Add Profile Customer

Ο Γραμματέας συμπληρώνει τα απαραίτητα στοιχεία του χρήστη και αποθήκευει στη βάση. Με το πάτημα του κουμπιού αποθήκευση τα δεδομένα αποθηκεύονται στην πλατφόρμα Firebase.

Figure 0:82-Logout Icon

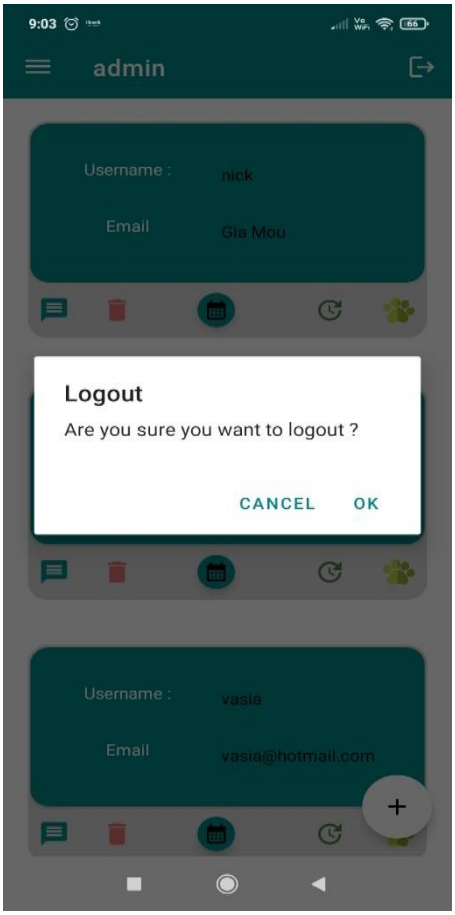


Figure 0:83-Logout Message

Η διαδικασία εξόδου από την εφαρμογή μας περιγράφεται στην παραπάνω εικόνα.

Figure 0:84-Menu Icon

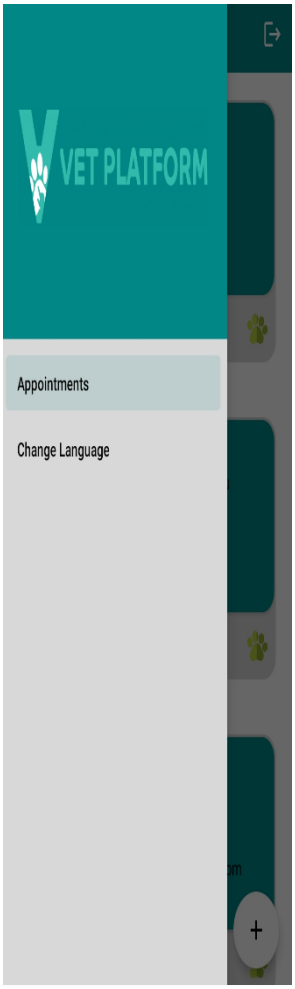


Figure 0:85-Menu inbox

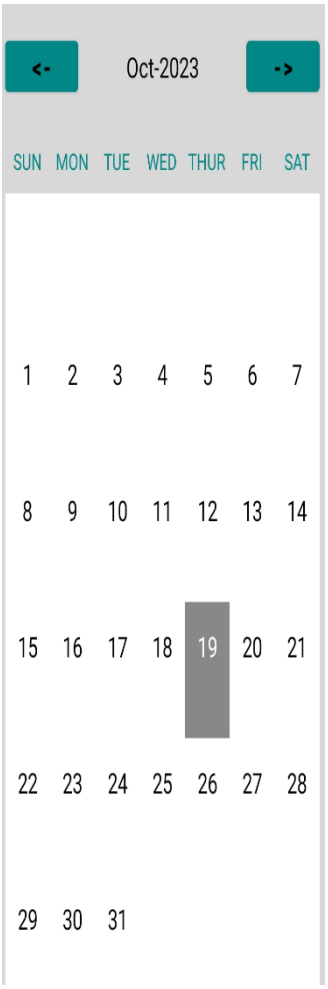


Figure 0:86-Appointments Calendar

Figure 0:87-Appointment Detail

Από το MENU επιλογών ακολουθώντας τα βήματα από τα αριστερά προς τα δεξιά εμφανίζονται τα διαθέσιμα ραντεβού που έχουν κλείσει η πελάτες μας. Οι ημερομηνίες που διαθέτουν ραντεβού έχουνε έντονο χρωματισμό. Πατώντας διπλό κλικ επάνω στην ημερομηνία εμφανίζεται η τελευταία οθόνη.

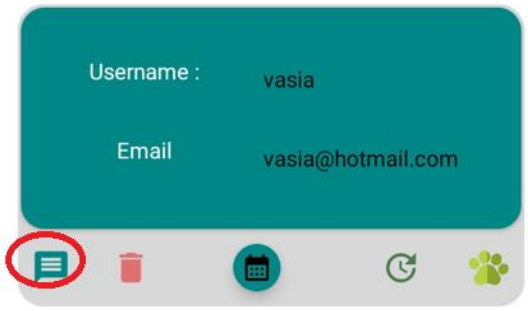


Figure 0:88-Customer Chat Icon

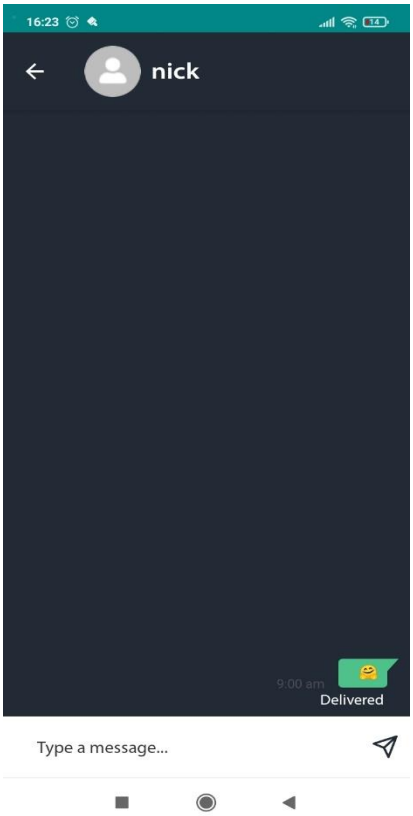


Figure 0:89-Chat Interface

Πατώντας το εικονίδιο αριστερά εμφανίζεται η φόρμα συνομιλιών που εμφανίζεται δίπλα.Μπορούμε να αντλαλάξουμε μηνύματα με τον πελάτη και να ενημερωθούμε για την πορεία της θεραπείας.

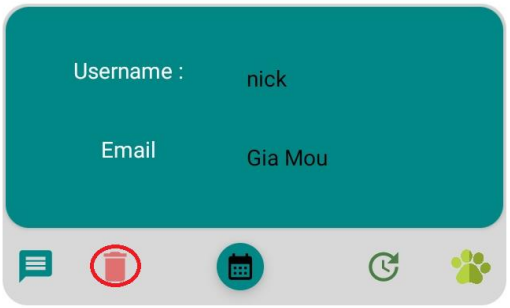


Figure 0:90-Delete Icon

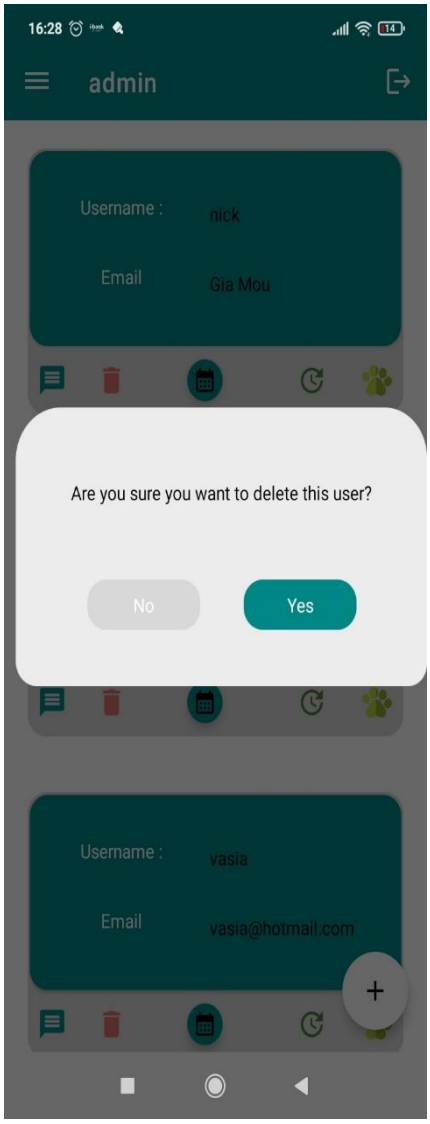


Figure 0:91-Delete Icon Message

Πατώντας το εικονίδιο διαγραφής εμφανίζεται το παράθυρο που βλέπετε στη δεξιά οθόνη.Ο γραμματέας εάν είναι σίγουρος για την ενέργεια που πάει να κάνει επιλέγει το 'Yes' και διαγράφοντα τα στοιχεία του πελάτη.

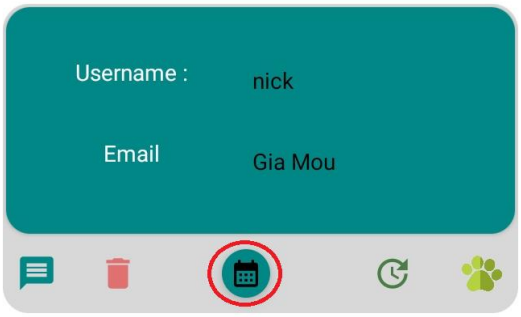


Figure 0:92-Appointment Icon

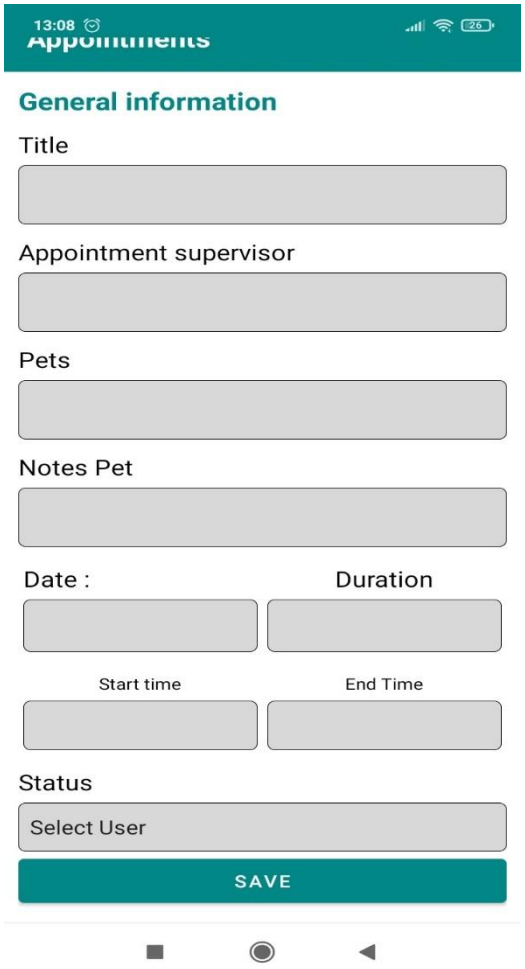


Figure 0:93-Appointment Interface

Πατώντας το εικονίδιο ημερολόγιο εμφανίζεται το παράθυρο που βλέπετε στη δεξιά οθόνη.Ο γραμματέας συμπληρώνει τα στοιχεία του της κράτησης με ποιόν ιατρό θα γίνει , το είδος του κατοικιδίου, την ώρα έναρξης του ραντεβού και τη διάρκεια.Εφόσος έχει ολοκληρωθεί με επιτυχία η διαδικασία υποβολής των στοιχείων για το ραντεβού στην κεντρική οθόνη του πελάτη εμφανίζεται το αντίστοιχό ραντεβού.

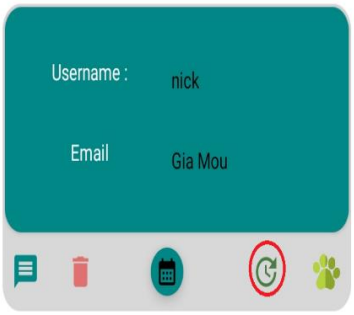


Figure 0:94-User Details Update Icon

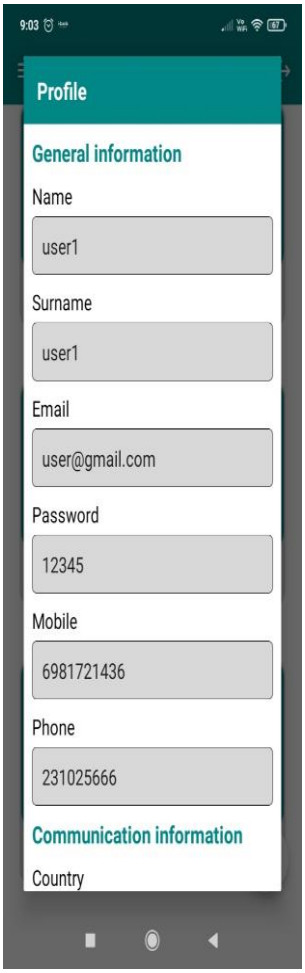


Figure 0:95-Update Interface 1

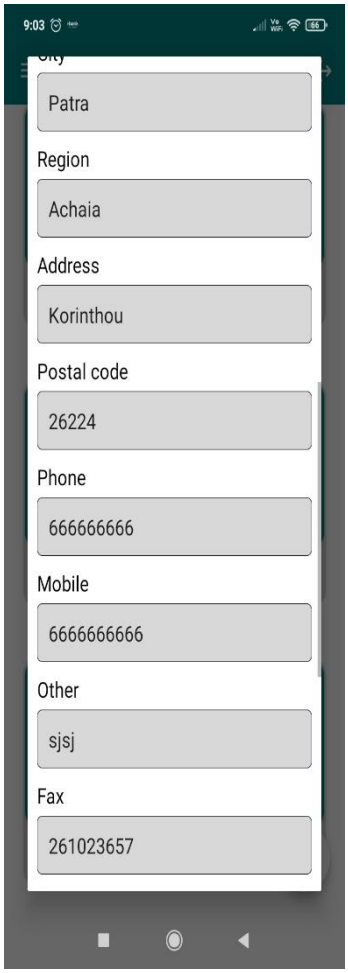


Figure 0:96-Update Interface 2

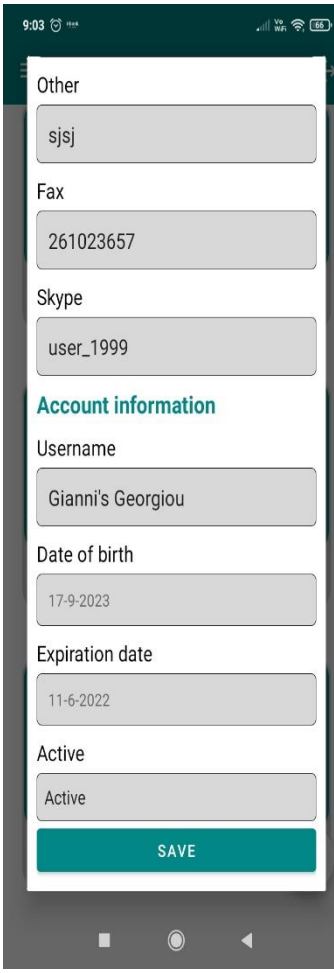


Figure 0:97-Update Interface 3

Επιλέγοντας το τέταρτο εικονίδιο μπορούμε να κάνουμε ενημέρωση στα στοιχεία του πελάτη και να αποθηκευτεί το αποτέλεσμα στην απομακρυσμένη βάση.

Figure 0:98-Pet Icon

Figure 0:99-Pet List

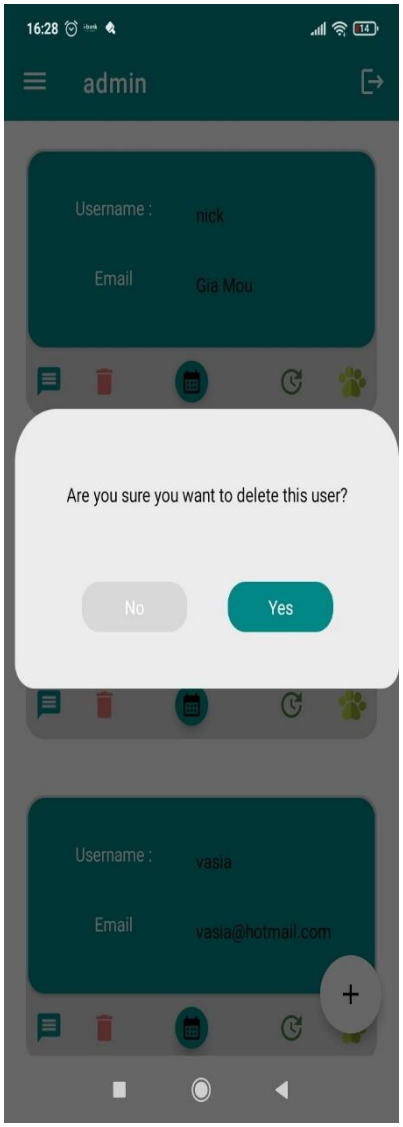


Figure 0:100-Delete Pet Profile

Επιλέγοντας το τελευταίο εικονίδιο μπορούμε να διαγράψουμε το κατοικίδιο που έχουμε αναθέσει ήδη σε κάποιον χρήστη η να κάνουμε ενημέρωση των στοιχείων του υπάρχοντος κατοικιδίου. Επιλέγοντας το δέυτερο εικονίδιο από την καρτέλα του κατικοιδίου εμφανίζεται η φόρμα που βλέπουμε στις παρακάτω φωτογραφίες και τροποποιώντας κάποια στοιχεία με καινούρια μπορεί να αλλάξουν τα υπάρχοντα δεδομένα.

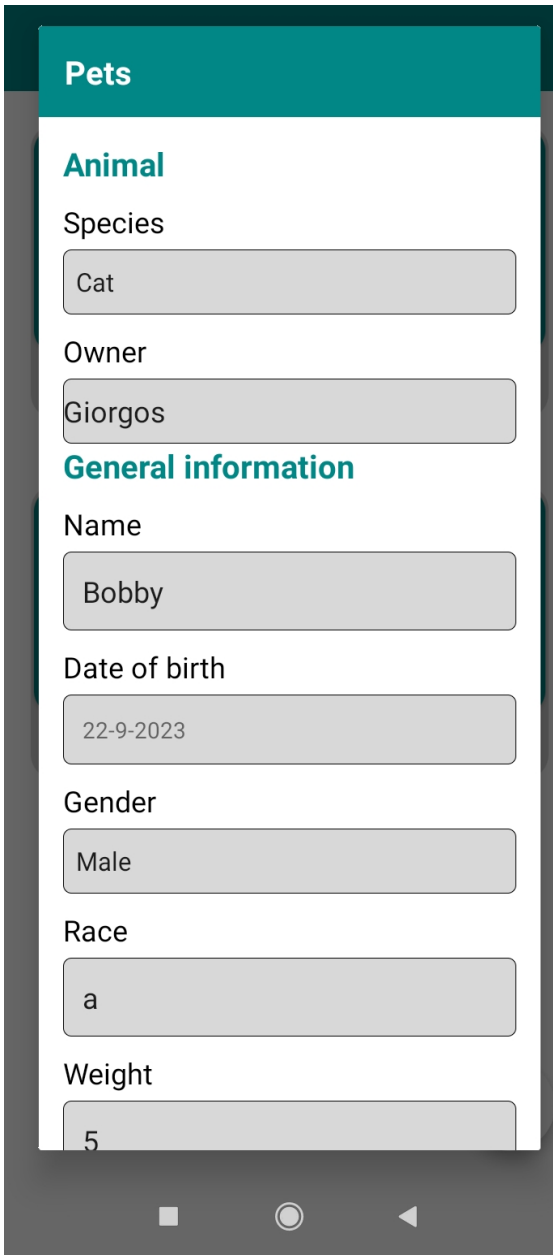


Figure 0:101-Pet Profile Data 1

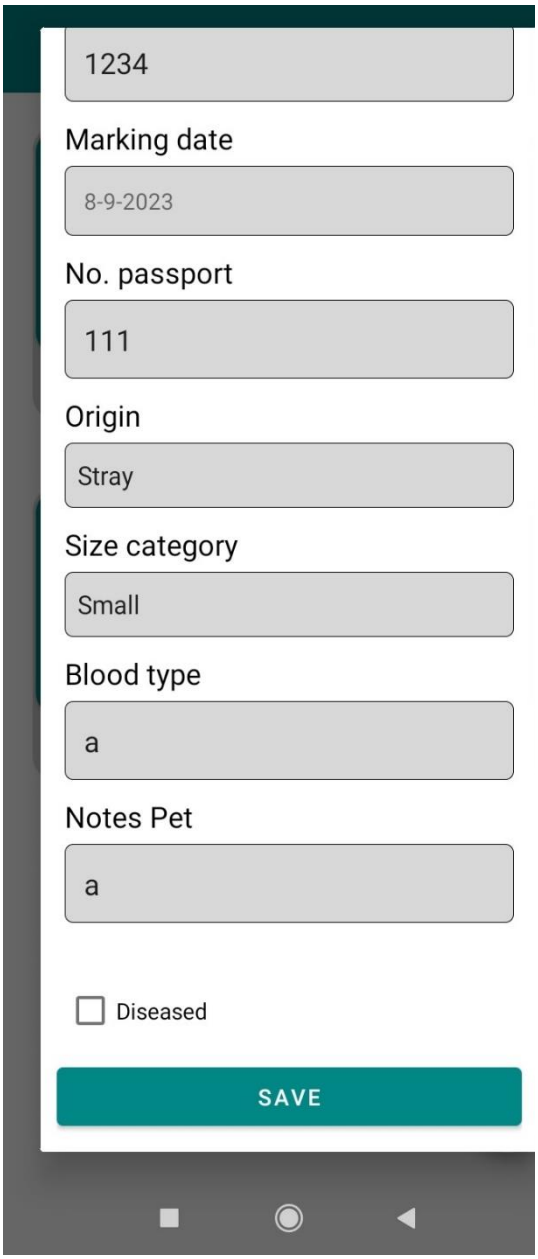
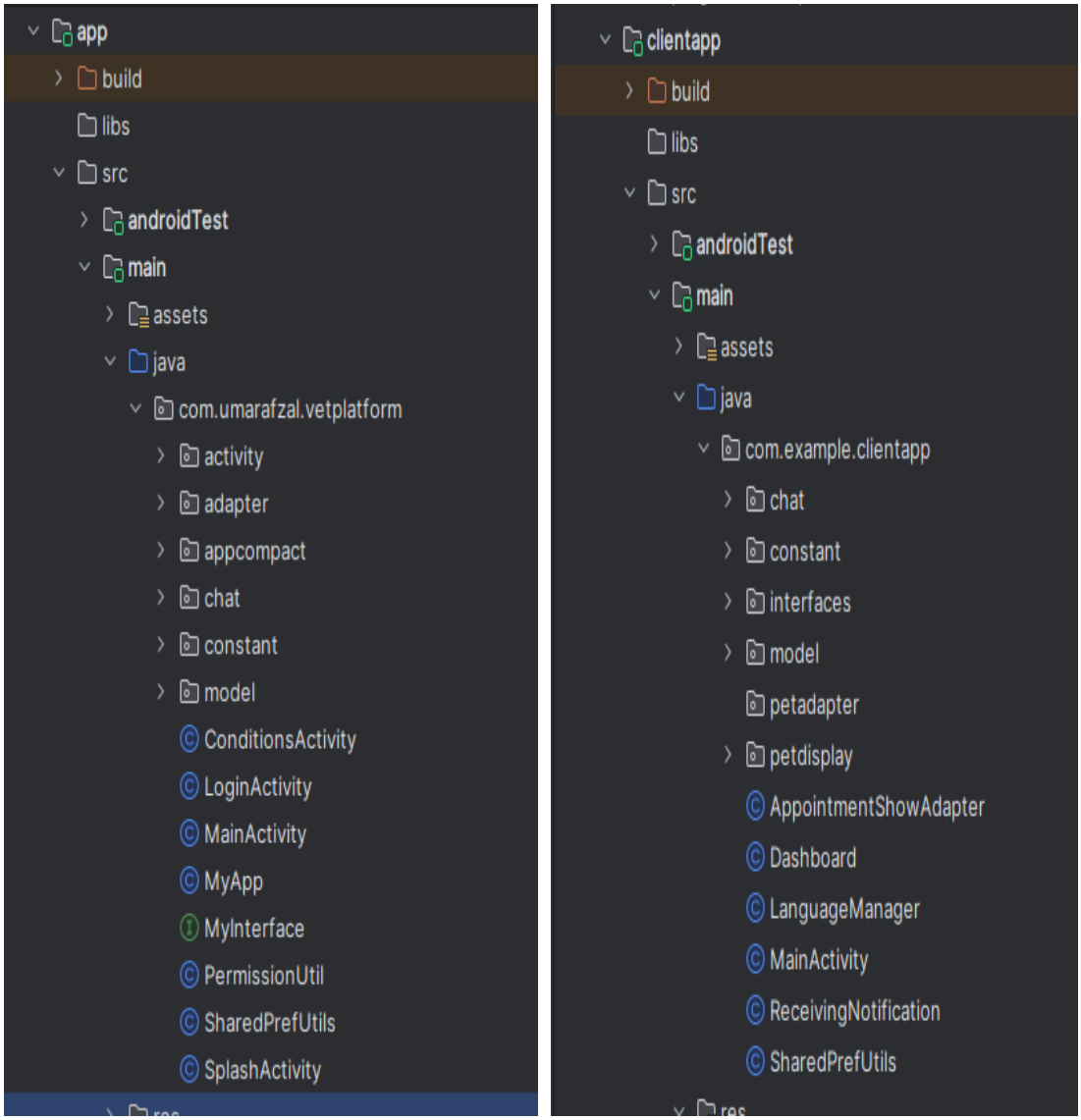


Figure 0:102-Pet Profile Data 2

Η προσθήκη του καινούριου κατοικιδίου γίνεται επιλέγοντας στο κάτω δεξιά μέρος της οθόνης το κουμπί πορσθήκης και εμφανίζεται η παρακάτω φόρμα συμπλήρωσης στοιχείων του ζώου.

ΚΕΦΑΛΑΙΟ 5: ΚΩΔΙΚΑΣ ΕΦΑΡΜΟΓΗΣ

5.1 Δομή



Στην παραπάνω εικόνα βλέπουμε γενικά τη δομή των δύο εφαρμογών.Στην αριστερή εικόνα γίνεται οι κατηγοριοποίηση των κλάσεων ανά λειτουργία σε ξεχωριστό φάκελο για την εφαρμογή(App) δηλαδή είναι η πρώτη εφαρμογή που χρησιμοποιεί ο Κτηνίατρος .

Η κατηγοριοποίηση των λειτουργιών για την εφαρμογή(APP) είναι :

- **Activity (Φάκελος):** Περιέχει τα Activities (Κύριες λειτουργίες) το κύριο συστατικό που εκτελείται στο στρώμα παρουσίασης(Presentation layer) για κάθε οθόνη της εφαρμογής.Τa activities της εφαρμογής μας είναι:
  - **CreateNewAppointment:** Είναι το Activity που διαχειρίζεται τις λειτουργίες που αφορούν για το κλείσιμο ραντεβού. Περιέχει τη φόρμα για τη συμπλήρωση των στοιχείων για το ραντεβού.***Activity_create_new_appointment.xml*** είναι η διεπαφή χρήστη όπου περιέχει τη κατάλληλη φόρμα για το ραντεβού.Ο κτηνίατρος συμπληρώνει το όνομα τοθ ιδιοκτήτη, το είδος του ζώου και άλλες λεπτομέρειες.
  - **PetDetailActivity:** Είναι το Activity που διαχειρίζεται τις λειτουργίες που αφορούν για τις ιδιότητες του κάθε κατοικιδίου. Διαχειρίζεται τα δεδομένα που εισάγει ο χρήστης της εφαρμογής από τη φόρμα ***pet_detail.xml***
  - **PetDisplayActivity:** Είναι το Activity που εμφανίζει τις λεπτομέρειες για κάθε κατοικίδιο που είναι αποθηκευμένο στη βάση της εφαρμογής μας. Η εμφάνιση των στοιχείων γίνεται σε μορφή λίστας..***Activity_pet_display.xml*** είναι η διεπαφή που έχει σχεδιαστεί για την εμφάνιση των στοιχείων της διεπαφής..
  - **AppointmentActivity**: Είναι το Activity που εμφανίζει το πρόγραμμα του ραντεβού ανα βδομάδα και επιτρέπει στον χρήστη της εφαρμογής να μπορεί να κάνει κράτηση σε μια συγκεκριμένη μέρα και σε μια συγκεκριμένη ώρα. Η εμφάνιση των στοιχείων γίνεται μέω της διεπαφής του χρήστη. ***activity_appointment.xml***
  - **AppointmentDisplayUser**: Είναι το Activity που εμφανίζει τα διαθέσιμα ραντεβού των χρηστών.Activity_appointment_display_user.xml είναι η διεπαφή του χρήστη.
  - **CreateProfileActivity:** Είναι το Activity που περιέχει τα κατάλληλα πεδία φόρμας για τη δημιουργία προφίλ πελάτη.***Activity_create_profile.xml*** είναι το αρχείο που περιέχει τα στοιχεία της φόρμας για τη δημιουργία του προφίλ του χρήστη.

- **Adapter(Φάκελος):** Οι προσαρμογείς(Adapters) είναι δομές που συνδέουν μια πηγή δεδομένων με το αντικείμενο που χρειάζεται τα δεδομένα.Ο προσαρμογέας(Adapter) καθιστά τα δεδομένα σε μια πηγή δεδομένων διαθέσιμα για χρήση σε οπτικά στοιχεία.Τα διαθέσιμα Adapters που έχουμε στην εφαρμογή μας είναι :
    - **AppointmentUserShowAdapter:** Είναι ο προσαρμογεάς που σημπληρώνει τα στοιχεία της λίστας (Recycle View) όπου εμφανίζονται τα δεδομένα των χρηστών της εφαρμογής.
    - **CalendarAdapter:** Είναι ο προσαρμογεάς που σημπληρώνει τα στοιχεία της λίστας (Recycle View) με τα διαθέσιμα ραντεβου της ημέρας.
    - **ClientShowAdapter:** Είναι ο προσαρμογεάς που σημπληρώνει τα στοιχεία της λίστας (Recycle View) με τα διαθέσιμα ονόματα των πελατών της εφαρμογής μας.

    - **PetDataAdapter:** Είναι ο προσαρμογεάς που σημπληρώνει τα στοιχεία της λίστας (Recycle View) με τα διαθέσιμα στοιχεία των κατοικιδίων ανά πελάτη της εφαρμογής μας.

- **Appcompact(Φάκελος):**Περιέχει τις δύο κλάσεις που είναι υπέυθυνα για τη γλώσσα της εφαρμογής μας.Περιέχει τις κατάλληλες μεθόδους που έχουν να κάνουν με τη γλώσσα της εφαρμογής μας. Οι διαθέσιμες κλάσεις μας είναι :
    - **LanguageManager:** Είναι η κλάση που περιέχει τις κατάλληλες μεθόδους (**methods**) και το σώμα δεδομένων (**data members**) για την αλλαγή της γλώσσας της εφαρμογής μας.
    - **MyApp:** Είναι το Activity που περιέχει τις κατάλληλη μεθόδο (Oncreate()) για την αλλάγή της κλάσης.
- **Constant (Φάκελος):** Είναι ο φάκελος που περιέχει την κλάση MyConstant.Η κλάση αυτή περιέχει τις σταθερές τιμές της εφαρμογής μας μέσα σε μια κλάση και μέσω αυτής της κλάσης η προσπέλαση των σταθερών τιμών γίνεται με πιο εύκολο και κατανοητό τρόπο.
- **Interfaces(Φάκελος):**Είναι ο φάκελος που περιέχει την διεπαφή MyInterface όπου υπάρχουν οι κατάλληλοι μέθοδοι για την προσπέλαση των δεδομένων από τη βάση δεδομένων η για τη συλπήρωση των δεδομένων της λίστας.
  Model (Φάκελος): Είναι ο φάκελος που περιέχει τις κλάσεις (Model) που έχει να κάνει με το τύπο δεδομένων που έχουμε στην εφαρμογή μας. Υπάρχουν οι μέθοδοι get και set για την προσπέλαση των data members της κλάσης και τονδημιουργό(Constructor) , για τη δημιουργία των αντικειμένενων από το συγκεκριμένο τύπου δεδομένων.
  Ο φάκελος περιέχει τις κλάσεις:
    - **AdminInfo:** Περιέχει τα στοιχεία του διαχειρστή της εφαρμογής και τις κατάλληλες μεθόδους για την διαχείρηση αυτών των δεδομένων.

- **AppointmentDateModel:** Περιέχει τα στοιχεία για την κράτηση της εφαρμογής και τις κατάλληλες μεθόδους για την διαχείρηση αυτών των δεδομένων. Περιέχει δύο μέλη δεδομένων, η μία για την ημερονηνία του ραντβεού και η δέυτερη είναι για το χρωματισμού της ημερομηνίας.
- **Chat:** Είναι η κλάση μοντέλο (Model) για τη συνομιλία μεταξύ των χρηστών. Περιέχει στοιχεία όπως ο αποστολεάς και ο παραλήπτης του μηνύματος κ.α και τις κατάλληλες μεθόδους για την συμπλήρωση των στοιχείων αυτών.
- **ChatList:** Είναι η κλάση μοντέλο (Model) για τη συνομιλία μεταξύ των χρηστών υπό μορφή λίστας. Περιέχει το ID για κάθε συνομιλία και τις κατάλληλες μεθόδους για την συμπλήρωση των στοιχείων αυτών.
- **PetData:** Περιέχει τα στοιχεία για το κατοικίδιο.
- **User:** Είναι η κλάση μοντέλο (Model) για τους χρήστες της εφαρμογής. Περιέχει στοιχεία όπως το όνομα του, η εικόνα του προφίλ, την κατάσταση του αν είναι ενεργός χρήστης η όχι .

- **MainActivity:**Είναι η κεντρική κλάση της εφαρμογής όπου περιέχει όλα τα στοιχεία σχετικά με την εφαρμογή.Αρχικά περιέχει τα δομικά στοιχεία όπως είναι το Menu η πεδίο της κεντρικής οθόνης όπου αλλάζουν τα δεδομένα μέσα στο στοιχείο αυτό και τις σταθερές που χρειάζονται για την επικοινωνία με την βάση .
- **LoginActivity:** Είναι η κλάση όπου γίνεται η ταυτοποίηση των στοιχείων των χρηστών της εφαρμογής.Γίνεται η σύνδεση με την απομακρυσμένη βάση και εφόσον υπάρχει ο συγκεκριμένος χρήστης στη βάση δεδομένων μεταφέρεται στην κεντρική οθόνη της εφαρμογής.

Η κατηγοριοποίηση των λειτουργιών για την εφαρμογή(ClientApp) είναι :

- **Constant(Φάκελος):** Είναι ο φάκελος που περιέχει την κλάση MyConstant.Η κλάση αυτή περιέχει τις σταθερές τιμές της εφαρμογής μας μέσα σε μια κλάση και μέσω αυτής της κλάσης η προσπέλαση των σταθερών τιμών γίνεται με πιο εύκολο και κατανοητό τρόπο.
- **Interfaces(Φάκελος):**Είναι ο φάκελος που περιέχει την διεπαφή MyInterface όπου υπάρχουν οι κατάλληλοι μέθοδοι για την προσπέλαση των δεδομένων από τη βάση δεδομένων η για τη συλπήρωση των δεδομένων της λίστας.
- **Model(Φάκελος):** Είναι ο φάκελος που περιέχει τις κλάσεις(Model) που έχει να κάνει με το τύπο δεδομένων που έχουμε στην εφαρμογή μας. Υπάρχουν οι μέθοδοι get και set για την προσπέλαση των data members της κλάσης και τονδημιουργό(Constructor) , για τη δημιουργία των αντικειμένενων από το συγκεκριμένο τύπου δεδομένων.
  Ο φάκελος περιέχει την κλάση PetData η οποία κρατάει τις πληροφορίες γι ατα κατοικίδια.
- **PetAdapter(Φάκελος):** Περιέχει την κλάση PetAdapter.Είναι ο προσαρμογεάς που σημπληρώνει τα στοιχεία της λίστας(Recycle View) με τα διαθέσιμα στοιχεία των κατοικιδίων ανά πελάτη της εφαρμογής μας.

PetDisplay(Φάκελος): Είναι το Activity που σημπληρώνει τα στοιχεία της λίστας(Recycle View) με τα διαθέσιμα ραντεβού που έχει ο πελάτης μας για κάθε κατοικίδιο του.

AppointmentShowAdapter: Είναι ο προσαρμογεάς που σημπληρώνει τα στοιχεία της λίστας (Recycle View) όπου εμφανίζει τα περιοχόμενα για κάθε ραντεβού που έχει κλήσει ο συγεκριμένος πελάτης.

Dashboard: Είναι η κεντρική οθόνη που εμφανίζονται τα στοιχεία των πελατών και στο επάνω μέρος της οθόνης υπάρχουν επιλογές Μενού από όπου μπορούμε να δούμε τα διαθέσιμα ραντβεού του συγκεκριμένου πελάτη η να αλλάξουμε τη γλώσσα της εφαρμογής μας.

LanguageManager: Είναι η κλάση που είναι υπέυθυνη για τη διαχείριση της γλώσσας της εφαρμογής μας.

MainActivity: Είναι η κεντρική κλάση της εφαρμογής όπου περιέχει όλα τα στοιχεία σχετικά με την εφαρμογή. Αρχικά περιέχει τα δομικά στοιχεία όπως είναι το Menu η πεδίο της κεντρικής οθόνης όπου αλλάζουν τα δεδομένα μέσα στο στοιχείο αυτό και τις σταθερές που χρειάζονται για την επικοινωνία με την βάση .

5.2 Συναρτήσεις

```
1 usage
130     private void chatList() {
131         mUsers = ne  com.example.clientapp.chat.Fragments.ChatsFragm
132         reference =  ent                                          h: "Users");
133         reference.a  private void chatList()
134             @Overri  VetPlatform.clientapp.main
135             public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
136                 mUsers.clear();
137                 for (DataSnapshot snapshot : dataSnapshot.getChildren()){
138                     User user = snapshot.getValue(User.class);
139                     for (Chatlist chatlist : usersList){
140                         if (user!= null && user.getId()!=null && chatlist!=null && chatlist.getId()!= null &&
141                                 user.getId().equals(chatlist.getId())){
142                             mUsers.add(user);
143                         }
144                     }
145                 }
146
147
148                 userAdapter = new UserAdapter(getContext(), onItemClick,mUsers,  ischat: true);
149                 recyclerView.setAdapter(userAdapter);
150             }
151
152             @Override
153             public void onCancelled(@NonNull DatabaseError databaseError) {
154
155             }
156         });
157     }
158
159 }
```

Η συνάρτηση αυτή επικοινωνεί με τη βάση και βρίσκει αρχικά τον κόμβο “ChatList” .Επειτα βρίσκεί τον θυγατρικό κόμβο που βρίσκεται κάτω από το αρχικό κόμβο και φέρνει όλα τα δεδομένα που υπάρχει κάτω από αυτό.Κάθε φορά που προστίθεται μια καινούρια πληροφορία καλείται αυτόματα αυτή η συνάρτηση και δίνεται σαν όρισμα στον προσαρμογέα της λίστας.Δηλαδή εμφανίζεται στην λίστα το καινούριο μήνυμα που έχει έρθει.

```
1 usage
private void updateToken(String token){
    DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Tokens");
    Token token1 = new Token(token);
    reference.child(fuser).setValue(token1);
}
```

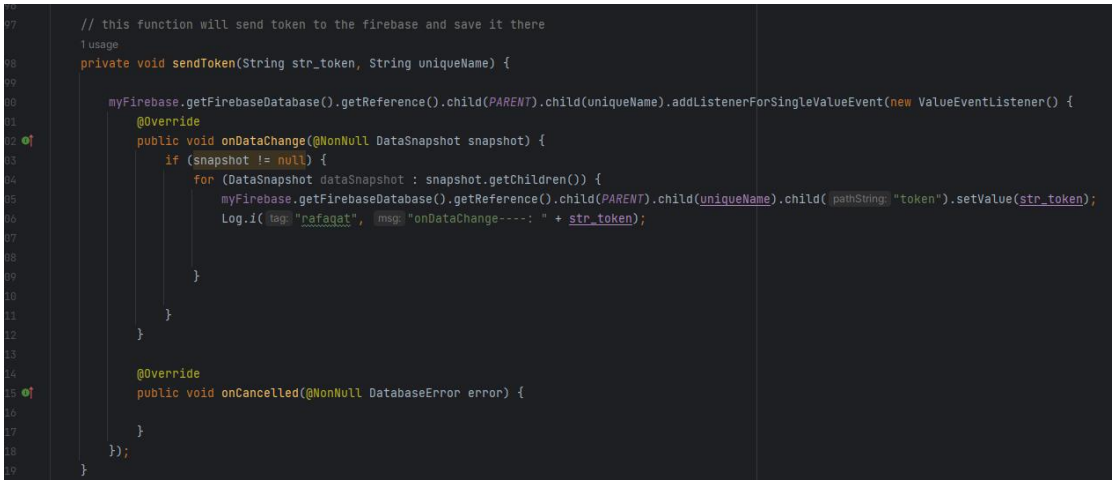
Η συνάρτηση αυτή φέρνει τον κλειδί του χρήστη που χρησιμοποιείται για την ταυτοποίηση του χρήστη.Είναι μοναδικό για κάθε χρήστη και κάθε φορά που γίνεται μια ανταλλαγή δεδομένων βοηθάει στην ασφάλεια του συστήματος.

```
98      @Override
99      public void detailPet(PetData model) {
100         AlertDialog.Builder alert = new AlertDialog.Builder( context: this);
101         LayoutInflater inflater = LayoutInflater.from( context: this);
102         View view = inflater.inflate(R.layout.activity_pet_detail_dialog_display,  root: null);
103         alert.setView(view);
104         AlertDialog alertDialog = alert.create();
105         alertDialog.setCanceledOnTouchOutside(true);
106         TextView aname = view.findViewById(R.id.name);
107         TextView owner = view.findViewById(R.id.owner);
108         MaterialTextView ap_dateOfBirth = view.findViewById(R.id.p_dateOfBirth);
109         TextView arace = view.findViewById(R.id.race);
110         TextView aweight = view.findViewById(R.id.weight);
111         TextView asterilization = view.findViewById(R.id.sterilization);
112         TextView aplacementID = view.findViewById(R.id.placementID);
113         TextView aelectronicID = view.findViewById(R.id.electronicID);
114         MaterialTextView amarkingDate = view.findViewById(R.id.markingDate);
115         MaterialTextView asterilizationDate = view.findViewById(R.id.sterilizationDate);
116         TextView apassportNo = view.findViewById(R.id.passportNo);
117         TextView abloodType = view.findViewById(R.id.bloodType);
118         TextView anotes_pet = view.findViewById(R.id.notes_pet);
119         TextView aspecies_spinner = view.findViewById(R.id.species_spinner);
120         TextView agender_spinner = view.findViewById(R.id.gender_spinner);
121         TextView acolorCode_spinner = view.findViewById(R.id.colorCode_spinner);
122         TextView aorigin_spinner = view.findViewById(R.id.origin_spinner);
123         TextView asize_spinner = view.findViewById(R.id.size_spinner);
124         CheckBox deceasedd = view.findViewById(R.id.diseased);
125
126         ////
127         aname.setText(model.getName());
128         asize_spinner.setText(model.getSize());
129         aplacementID.setText(model.getPid());
130         owner.setText(model.getOwner());
131         aspecies_spinner.setText(model.getSpecies());
132         agender_spinner.setText(model.getGender());
```

Είναι η μέθοδος που παίρνει όλα τα δεδομένα από την βάση και συμληρώνει τα κατάλληλα πεδία του αντικειμένου και αργότερα δηλώνουμε σαν όρισμα στο Interface που εμφανίζει όλα τα δεδομένα στην καρτέλα του κατικοιδίου.

```
    private void loginMethod(String email, String password) {
        mProgressDialogue.setTitle("Signing In");
        mProgressDialogue.setMessage("Please wait while we are get information.....");
        mProgressDialogue.setCanceledOnTouchOutside(false);
        mProgressDialogue.show();

        myFirebase.getFirebaseDatabase().getReference().child(PARENT).addListenerForSingleValueEvent(new ValueEventListener() {...});
    }

    // this is the build in method, whenever applications start, this method calls and check if user is already login or not
    @Override
    protected void onStart() {
        super.onStart();
        SharedPreferences sharedPreferences = getSharedPreferences( name: "db", Context.MODE_PRIVATE);

        try {
            String check = sharedPreferences.getString( s: "check",  s1: "null");
            if (check.equals("user")) {

                Intent intent = new Intent( packageContext: MainActivity.this, Dashboard.class);
                startActivity(intent);
                finish();

            } else if (check.equals("null")) {
                //  Intent intent = new Intent(MainActivity.this, LoginActivity.class);
                //  startActivity(intent);
                //  finish();

            }

        } catch (Exception e) {
        }
    }
```

Στη μέθοδο αυτή γίνεται η ταυτοποίηση των στοιχείων του χρήστη και αν ο συγκεριμένος χρήστης έχει καταχωρήσει τα στοιχεία του νωρίτερα στη βάση μας.Μετά την επαλήθευση των στοιχείων μέσω Intent ο χρήστης μεταφέρεται στην Κεντρική Οθόνη (Dashboard) της εφαρμογής μας.



```
    // this function will send token to the firebase and save it there
    1 usage
    private void sendToken(String str_token, String uniqueName) {

        myFirebase.getFirebaseDatabase().getReference().child(PARENT).child(uniqueName).addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if (snapshot != null) {
                    for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                        myFirebase.getFirebaseDatabase().getReference().child(PARENT).child(uniqueName).child( pathString: "token").setValue(str_token);
                        Log.i( tag: "rafaqat",  msg: "onDataChange----: " + str_token);


                    }

                }
            }

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

            }
        });
    }
```

Αυτή η μέθοδος στέλνει το Token στην απομακρυσμένη βάση και αποθηκύευει στον κόμβο εκεί που βρίσκεται το e-mail(uniqename) του χρήστη της εφαρμογής μας.Η συνάρτηση αυτή δέχεται δύο ορίσματα το πρώτο όρισμα είναι το αλφαριθμητικό(String) και το δέυτερο είναι το e-mail του χρήστη το οποίο είναι μοναδικό(Unique).Η η τιμή που έχει το κάθε token είναι διαφορετικό και μοναδικό για κάθε χρήστη της εφαρμογής μας.

```
@Override
public void delete(Chat model, int pos) {

    try {

        final Dialog dialog = new Dialog( context: MessageActivity.this);
        Objects.requireNonNull(dialog.getWindow()).setBackgroundDrawableResource(android.R.color.transparent);
        dialog.setContentView(R.layout.bmi_info_dialog);
        TextView yes = dialog.findViewById(R.id.yes);
        TextView show = dialog.findViewById(R.id.textshow);
        show.setText("Do you want to delete the message?");
        TextView no = dialog.findViewById(R.id.no);
        yes.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                reference = FirebaseDatabase.getInstance().getReference( path: "Chats");
                Query query = reference.orderByChild( path: "time").equalTo(model.getTime());
                query.addValueEventListener(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot snapshot) {
                        for(DataSnapshot snapshot1 : snapshot.getChildren()){
                            snapshot1.getRef().removeValue();
                            messageAdapter.notifyDataSetChanged();
                        }

                    }

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

                    }
                });
                dialog.dismiss();
            }
        });
        no.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) { dialog.dismiss(); }
        });
        dialog.show();
```

Με τη χρήση αυτής της συνάρτησης μπορούμε να διαγράψουμε τα μηνύματα μας από τη λίστα των συνομιλιών.Γινεται επικοινωνία με την απομακρυσμένη βάση(Firebase)της εφαρμογής μας και διαγράφονται τα δεδομένα από τον κόμβο που αντιστοιχεί.

```
1 usage
private void readMesagges(final String myid, final String userid, final String imageurl){
    mchat = new ArrayList<>();
    messageAdapter = new MessageAdapter( mContext: MessageActivity.this, mchat, imageurl, myInterface: this);

    reference = FirebaseDatabase.getInstance().getReference( path: "Chats");
    reference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            mchat.clear();
            for (DataSnapshot snapshot : dataSnapshot.getChildren()){
                Chat chat = snapshot.getValue(Chat.class);
                if (chat.getReceiver().equals(myid) && chat.getSender().equals(userid) ||
                        chat.getReceiver().equals(userid) && chat.getSender().equals(myid)){
                    mchat.add(chat);
                }

                recyclerView.setAdapter(messageAdapter);
            }
        }

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

        }
    });
}
```

Με τη χρήση αυτής της συνάρτησης διαβάζουμε από τη λίστα των συνομιλιών τα αντίστοιχα μηνύματα που αντιστοιχούν στο προφίλ δικό μας και φέρνει όλα τα δεδομένα και τα εμφανίζει μέσα σε μια λίστα.Δίνει σαν όρισμα στον προσαρμογέα της λίστας για την εμφάνιση των μυνημάτων.

```
1 usage
private void seenMessage(final String userid){
    reference = FirebaseDatabase.getInstance().getReference( path: "Chats");
    seenListener = reference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            for (DataSnapshot snapshot : dataSnapshot.getChildren()){
                Chat chat = snapshot.getValue(Chat.class);
                if (chat.getReceiver().equals(fuser) && chat.getSender().equals(userid)){
                    HashMap<String, Object> hashMap = new HashMap<>();
                    hashMap.put("isseen", true);
                    snapshot.getRef().updateChildren(hashMap);
                }
            }
        }

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

        }
    });
}
```

Με τη χρήση αυτής της συνάρτησης ενημερώνεται η βάση ότι η ανάγνωση του μηνύματος έγινε με επιτυχία από την μεριά του Παραλήπτη.Αυτό επιτυγχάνεται αλλάζοντας την μεταβλητή “issen ” από false σε true.

```
no usages
public class MyFirebaseIdService extends FirebaseInstanceIdService {

    2 usages
    @Override
    public void onTokenRefresh() {
        super.onTokenRefresh();
        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();

        String refreshToken = FirebaseInstanceId.getInstance().getToken();
        if (firebaseUser != null){
            updateToken(refreshToken);
        }
    }

    1 usage
    private void updateToken(String refreshToken) {
        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();

        DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Tokens");
        Token token = new Token(refreshToken);
        reference.child(firebaseUser.getEmail().replace( target: ".", replacement: "")).setValue(token);
    }
}
```

Η κλάση αυτή διαθέτει δύο μεθόδους.Η μία διαβάζει από τη βάση τα tokens του κάθε χρήστη και κάνει σύγκριση με το ήδη υπάρχων στη βάση και χρησιμοποιεί κάθε φορά

που κάνει μεταφορά δεδομένων για ασφαλή λειτουργία της εφαρμογής μας.

```
no usages
public class MyFirebaseMessaging extends FirebaseMessagingService {

    3 usages
    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        super.onMessageReceived(remoteMessage);

        String sented = remoteMessage.getData().get("sented");
        String user = remoteMessage.getData().get("user");

        SharedPreferences preferences = getSharedPreferences( name: "PREFS", MODE_PRIVATE);
        String currentUser = preferences.getString( s: "currentuser",  s1: "none");

        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();

        if (firebaseUser != null && sented.equals(firebaseUser.getEmail().replace( target: ".",  replacement: ""))) {
            if (!currentUser.equals(user)) {
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
                    sendOreoNotification(remoteMessage);
                } else {
                    sendNotification(remoteMessage);
                }
            }
        }
    }
```

Η συνάρτηση αυτή αναλόγως με το λειτουργικό σύστημα που χρησιμοποεί η συσκευωή μας καλεί την ανάλογη συνάρτηση Notification.

```
    1 usage
    private void lastMessage(final String userid, final TextView last_msg){
        theLastMessage = "default";
         String firebaseUser = sharedPrefUtils.get("email").replace( target: ".", replacement: "");
        DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Chats");

        reference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                for (DataSnapshot snapshot : dataSnapshot.getChildren()){
                    Chat chat = snapshot.getValue(Chat.class);
                   if (firebaseUser != null && chat != null) {
                        if (chat.getReceiver().equals(firebaseUser) && chat.getSender().equals(userid) ||
                                chat.getReceiver().equals(userid) && chat.getSender().equals(firebaseUser)) {
                            theLastMessage = chat.getMessage();
                        }
                    }
                }

                switch (theLastMessage){
                    case  "default":
                        last_msg.setText("No Message");
                        break;

                    default:
                        last_msg.setText(theLastMessage);
                        break;
                }

                theLastMessage = "default";
            }
```

Η συνάρτηση αυτή ελέγχει κάθε φορά στη λίστα συνομιλιών αν έχει γίνει κάποια αλλαγή.Αν έχει γίνει κάποια προσθήκη στη λίστα δεδομένων τότε καλεί τις κατάλληλες συναρτήσεις και ενημερώνει τους αντίστοιχους χρήστες για την ορθή ανάγνωση του μηνύματος.

```
1 usage
private void readUsers() {

    String firebaseUser = sharedPrefUtils.get("email");
    DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Users");

    reference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if (search_users.getText().toString().equals("")) {
                mUsers.clear();
                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    User user = snapshot.getValue(User.class);

                    if (user!= null && user.getId()!=null && firebaseUser!=null && !user.getId().equals(firebaseUser)) {
                        mUsers.add(user);
                    }
                }

                if(mUsers.size()==0){
                    frameLayout.setVisibility(View.VISIBLE);
                }
                else{
                    frameLayout.setVisibility(View.GONE);
                }

                userAdapter = new UserAdapter(getContext(), onItemClick,mUsers,  ischat: false);
                recyclerView.setAdapter(userAdapter);
            }
        }

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

        }
    });
}
}
```

Η συνάρτηση αυτή φέρνει όλους του χρήστες από την απομακρυσμένη βάση και κάνει αναζήτηση μέσα σε αυτά τα δεομένα τον συγκεριμένο χρήστη της εφαρμογής μας.

```
1 usage
80  private void loginMethod(String email, String password) {
81      mProgressDialogue.setTitle("Signing In");
82      mProgressDialogue.setMessage("Please wait while we are get information.....");
83      mProgressDialogue.setCanceledOnTouchOutside(false);
84      mProgressDialogue.show();
85      myFirebase.getFirebaseDatabase().getReference().child(PARENT).addListenerForSingleValueEvent(new ValueEventListener() {
86          @Override
87          public void onDataChange(@NonNull DataSnapshot snapshot) {
88              try {
89                  if (snapshot.exists()) {
90                      for (DataSnapshot dataSnapshot : snapshot.getChildren()) {...}
121                 }
122                 else {
123                     mProgressDialogue.dismiss();
124                     Toast.makeText( context: LoginActivity.this, "Contact Developer", Toast.LENGTH_SHORT).show();
125                 }
126             }
127             catch (Exception e)
128             {
129                 Toast.makeText( context: LoginActivity.this,  text: ""+e.getMessage(), Toast.LENGTH_SHORT).show();
130             }
131         }
132
133         @Override
134         public void onCancelled(@NonNull DatabaseError error) {
135
136         }
137     });
138 }
```

Η συνάρτηση αυτή εκτελείται κατά την διάρκει της εισόδου του χρήστη στην εφαρμογή μας.

```
1 usage
private void changeLanguageMethod() {
    final String[] language = {"English", "Greek"};
    AlertDialog.Builder alertDialog = new AlertDialog.Builder( context: this);
    alertDialog.setTitle("Change Language");
    alertDialog.setSingleChoiceItems(language,  checkedItem: -1, new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            if (i == 0) {
                languageManager.updateResources( code: "en");
                recreate();
            }
            if (i == 1) {
                languageManager.updateResources( code: "el");
                recreate();
            }
        }
    });

    alertDialog.create();
    alertDialog.show();
}
```

Η συνάρτηση αυτή καλείται για την αλλαγή της γλώσσας της εφαρμογής μας.Η εφαρμογή μας διαθέτει δύο επιλογές από γλώσσες, ελληνικά και αγγλικά.

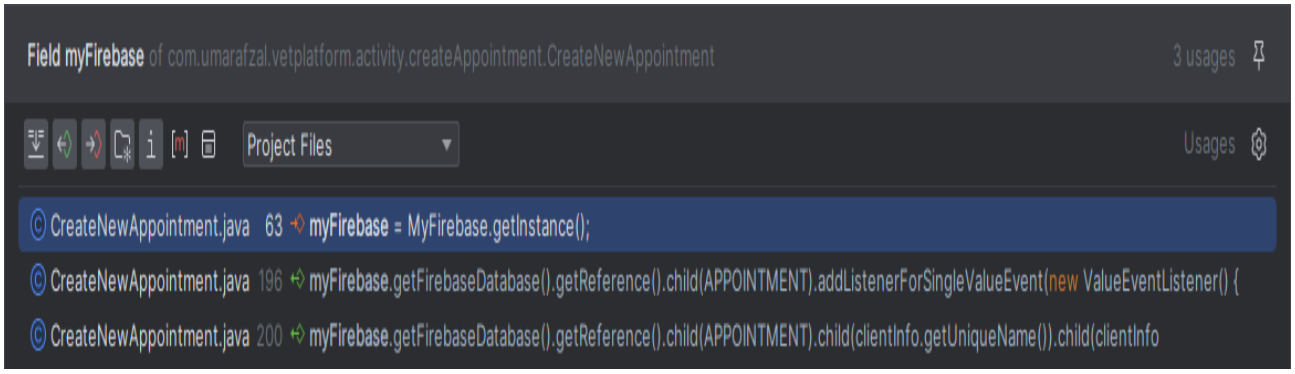
5.2 Κλήσεις στη βάση δεδομένων

Στην πρώτη εφαρμογή (ClientAApp)στα σημεία όπου υπάρχει επικοινωία με βάσεις δεδομένων είναι :	**Στην δεύτερη εφαρμογή (App)στα σημεία όπου υπάρχει επικοινωία με βάσεις δεδομένων είναι :**
UserAdapter: ○ ***lastMessage:*** ***ChatFragment:*** ○ ***UpdateToken:*** ○ ***ChatList:*** ***UserFragment:*** ○ ***SearchUsers:*** ○ ***ReadUsers:*** ***MyFirebaseIdService:*** ○ ***OnTokenRefresh:*** ○ ***UpdateToken:*** ***MyFirebaseMessaging:***	***CreateNewAppointment:*** ***PetDetailActivity:*** ***PetDisplayActivity:*** ○ ***deletePet:*** ***AppointmentActivity: :*** ○ ***getDateModel:*** ***AppointmentDisplayUser:*** ○ ***deleteAppointment:*** ***CreateProfileActivity:*** ○ ***sendData:*** ***UserAapter:*** ○ ***lastMessage:***

○ ***OnMessageReceived:*** ***MainActivityForChat:*** ○ ***Status:*** ***MessageActivity:*** ○ ***SeenMessage:*** ○ ***readMessage:*** ○ ***SendMessage:*** ○ ***sendNotification:*** ○ ***delete:*** ○ ***status:*** ***PetDisplay:*** ***Dashboard:*** ***MainActivity:*** ○ ***SendToken:*** ***loginMethod:***	***ChatsFragment:*** ○ ***UpdateToken:*** ○ ***ChatList:*** ***UsersFragment:*** ○ ***SearchUsers:*** ○ ***ReadUsers:*** ***MyFirebaseIdService:*** ○ ***onTokenRefresh:*** ○ ***updateToken:*** ***MyFirebaseMessaging:*** ○ ***onMessageReceived:*** ○ ***sendOreoNotification:*** ○ ***SendNotification:*** ***MainActivityForChat:*** ○ ***status:*** ***MessageActivity:*** ▪ ***seenMessage:*** ▪ ***sendMessage:*** ▪ ***readMessage:*** ▪ ***delete:*** ***LoginActivity:*** ○ ***LoginMethod:*** ***MainActivity:*** ▪ ***sconfigureToolBar:*** ▪ ***delete:***

App Κλάσεις όπου γίνεται επικοινωνία με τη βάση:

CreateNewAppointment:



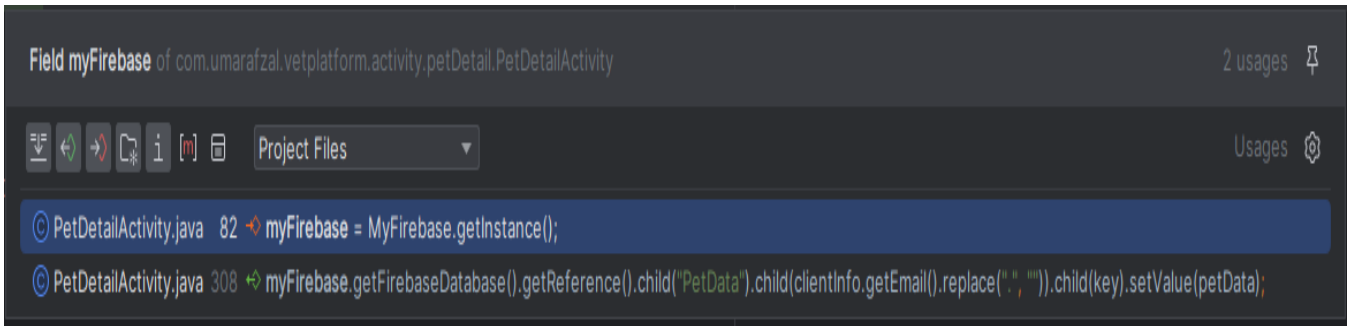
Στην παραπάνω εικόνα φαίνονται τα σημεία όπου γίνονται αναφορές στην βάση δεδομένων Firebase.Αρχικά μέσα στη μέθοδο Oncreate γίνεται η αρχικοποίηση του αντικειμένου.Επειτα γίνεται με τη μέθοδο getReference αναφορά στο θυγατρικό κόμβο Appointment.

Στην γραμμή 196 ενώ νωρίτερα έχουμε επιβεβαιώση ότι τα κατάλληλα πεδία της φόρμας έχουν συμπληρωθεί δημιουργείται το αντικείμενο ClientInfo και τα κατάλληλα πεδία του αντικειμένου και μετά καταχωρείται στην δενδροειδή δομή δημιουργείται ένα καινούριος θυγατρικός κόμβος.

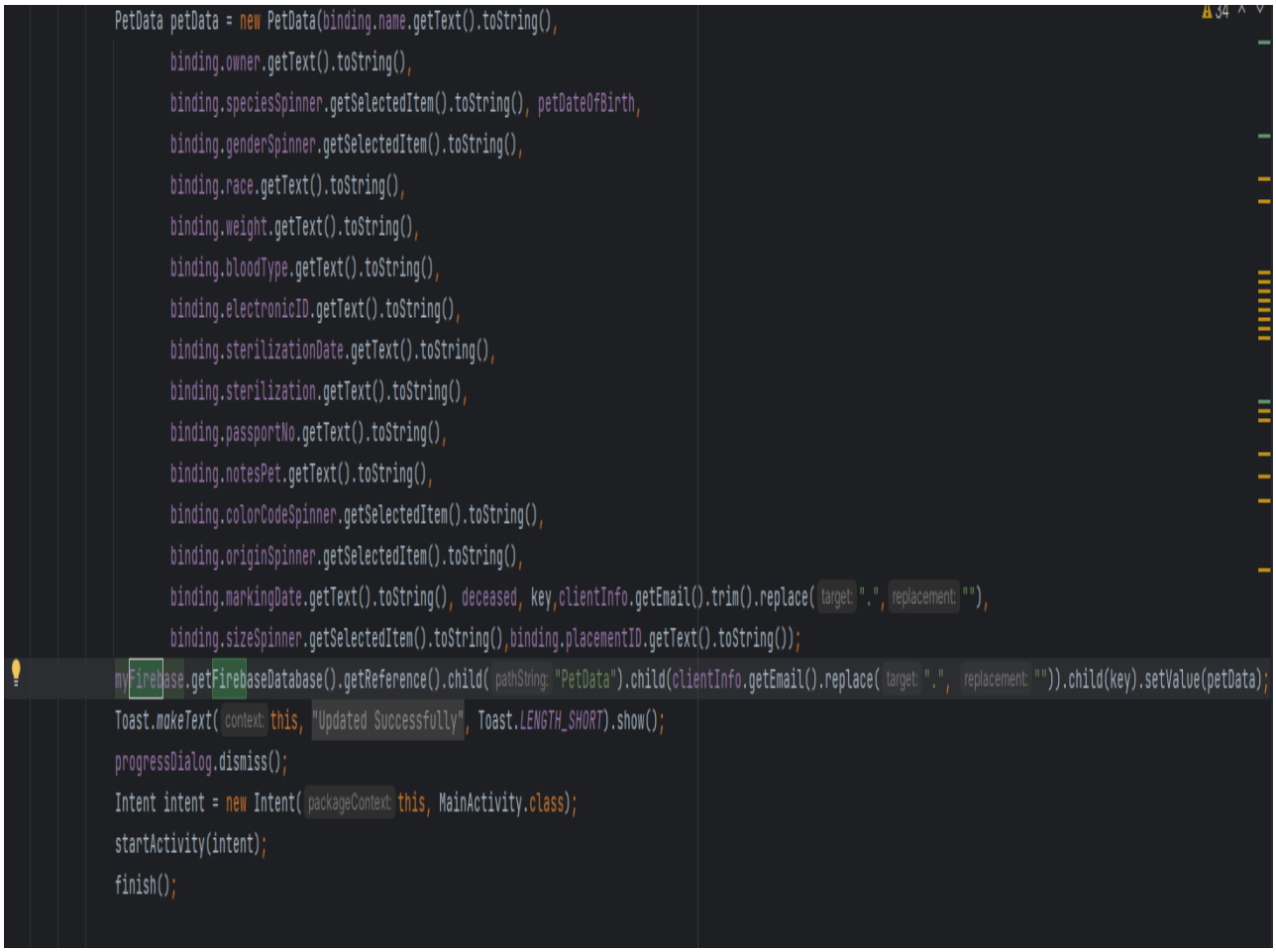
```
binding.save.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {


        if (!binding.name.getText().toString().isEmpty()) {
            if (!binding.surName.getText().toString().isEmpty()) {
                if (!binding.time.getText().toString().isEmpty()) {
                    if (!binding.date.getText().toString().isEmpty()) {
                        if (!binding.petname.getText().toString().isEmpty()) {
                            if (!binding.notes.getText().toString().isEmpty()) {

                                if (!binding.timeend.getText().toString().isEmpty())
                                {
                                    clientInfo.setDate(binding.date.getText().toString());
                                    clientInfo.setTime(binding.time.getText().toString());
                                    clientInfo.setTitle(binding.name.getText().toString());
                                    clientInfo.setStatus(binding.status.getSelectedItem().toString());
                                    clientInfo.setPetName(binding.petname.getText().toString());
                                    clientInfo.setNotes(binding.notes.getText().toString());
                                    clientInfo.setDuration(binding.displayTime.getText().toString());
                                    clientInfo.setEndtime(binding.timeend.getText().toString());
                                    clientInfo.setAppointmentSupervisor(binding.surName.getText().toString());
                                    myFirebase.getFirebaseDatabase().getReference().child(APPOINTMENT).addListenerForSingleValueEvent(new ValueEventListener() {
                                        @Override
                                        public void onDataChange(@NonNull DataSnapshot snapshot) {

                                            myFirebase.getFirebaseDatabase().getReference().child(APPOINTMENT).child(clientInfo.getUniqueName()).child(clientInfo.getKey()).setValue(clientInfo);
                                            senPushdNotification( body: "tar",  title: "Your Appointment Fix " + appointmentDate, clientInfo.getToken(), myKey);
                                            Log.i( tag: "tariq",  msg: "onDataChange: admin token" + clientInfo.getToken());

                                            Toast.makeText( context: CreateNewAppointment.this, "Updated Successfully", Toast.LENGTH_SHORT).show();
                                            finish();

                                        }

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

                                        }
```

Ο χρήστης μόλις έχει πατήσει στο κουμπί αποθήκευση , τότε ελέγχουμε τα πεδία της φόρμας στην διεπαφή του χρήστη αν έχουν αλλάξει τα δεδομένα και αν έχουν συμπληρωθεί κατάλληλε τότε τα καινούρια δεδομένα αποθηκέυονται στην απομακρύσμένη βάση.

PetDetailActivity:



Στην παραπάνω εικόνα φαινόνται τα σημεία μέσα στην κλάση όπου δημιουργείται το αντικείμενο myfirebase(γραμμή 82).Στην τελευταία γραμμή αποθηκέυονται τα δεδομένα του αντικειμένου “PetData” στην απομακρυσμένη βάση κάτω από την ετικέτα « PetData»

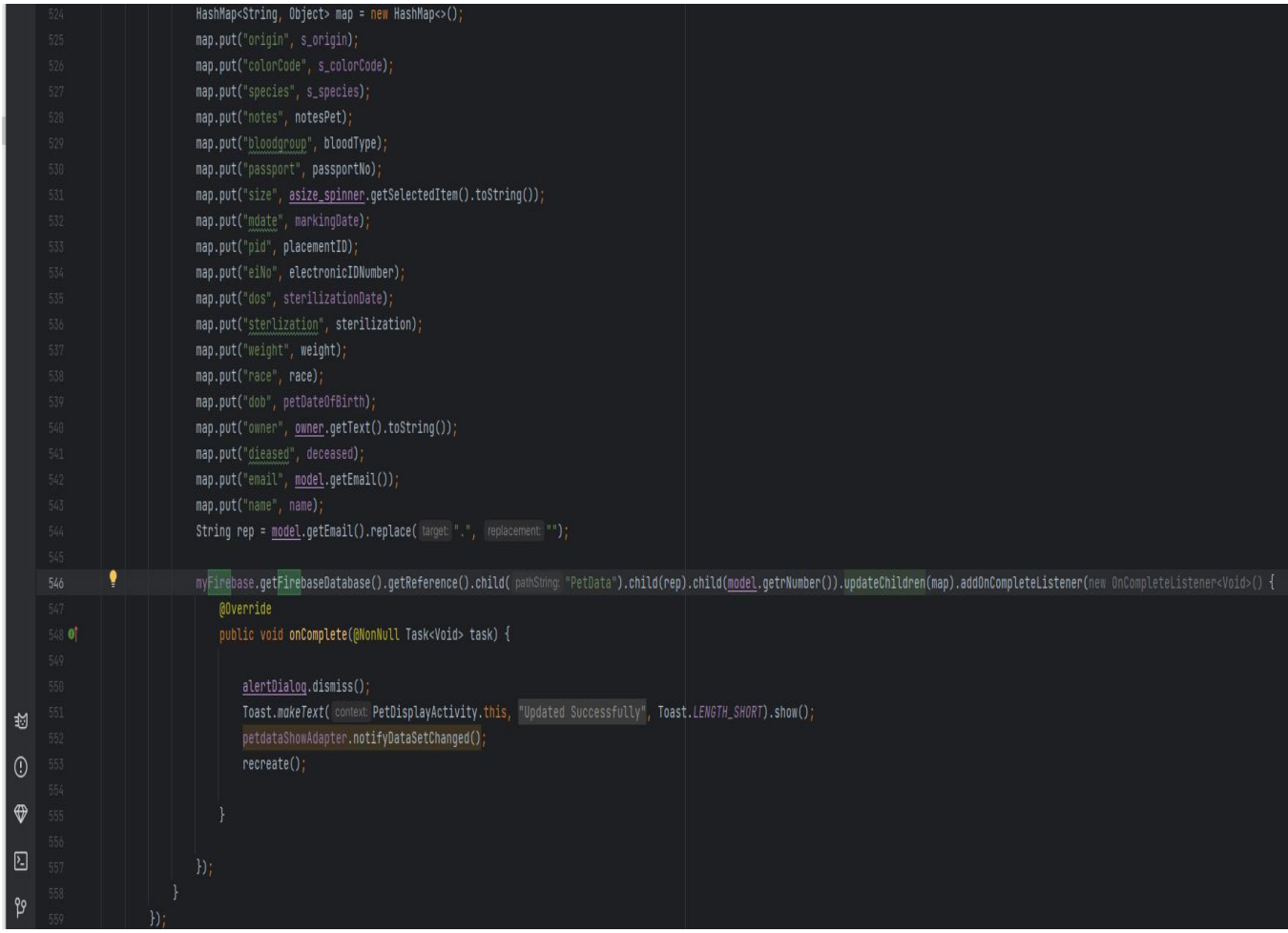


Συμπληρώνονται τα κατάλληλα πεδία του δημιουργού(Constructor) με τις κατάλληλες τιμές που έρχονται από την διεπαφή του χρήστη.Στην γραμμή 308 γίνεται η αναφορά στην απομακρυσμένη βάση και οι καινούριες τιμές προστείθονται κάτω

από την ετικέτα “ PetData”.Βρίσκει το Child κόμβο «PetData» εντοπίζει το email του συγεκριμένου χρήστη και με βάση το ID του συγκεκριμένου χρήστη και αποθηκέυει το αντικείμενο petData με όλα τα δεδομένα που έχει.

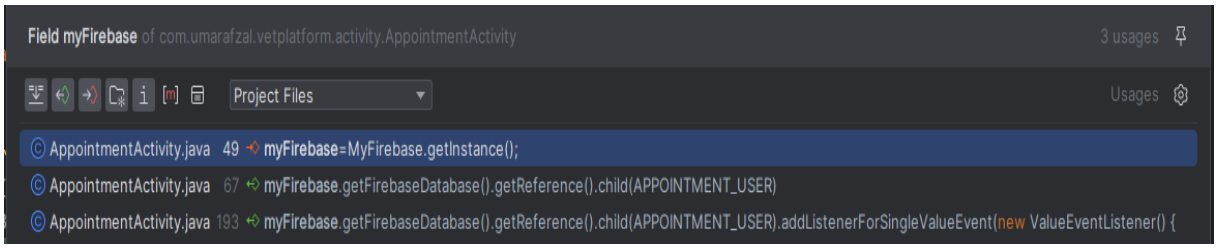
PetDisplayActivity:

Στην γραμμή 99 δημιουργείται μέσα στο Oncreate μέθοδο η αναφορά στο αντικείμενο τύπου Myfirebase.Εντοπίζει τους χρήστες που βρίσκονται κάτω από το κόμβο(Node)”PetData” εντοπίζει το email του συγκεκριμένου χρήστη και κάτω από αυτό δημιουργεί έναν καινούριο κόμβο με τα ήδη υπάρχοντα δεδομένα του αντικειμένου.

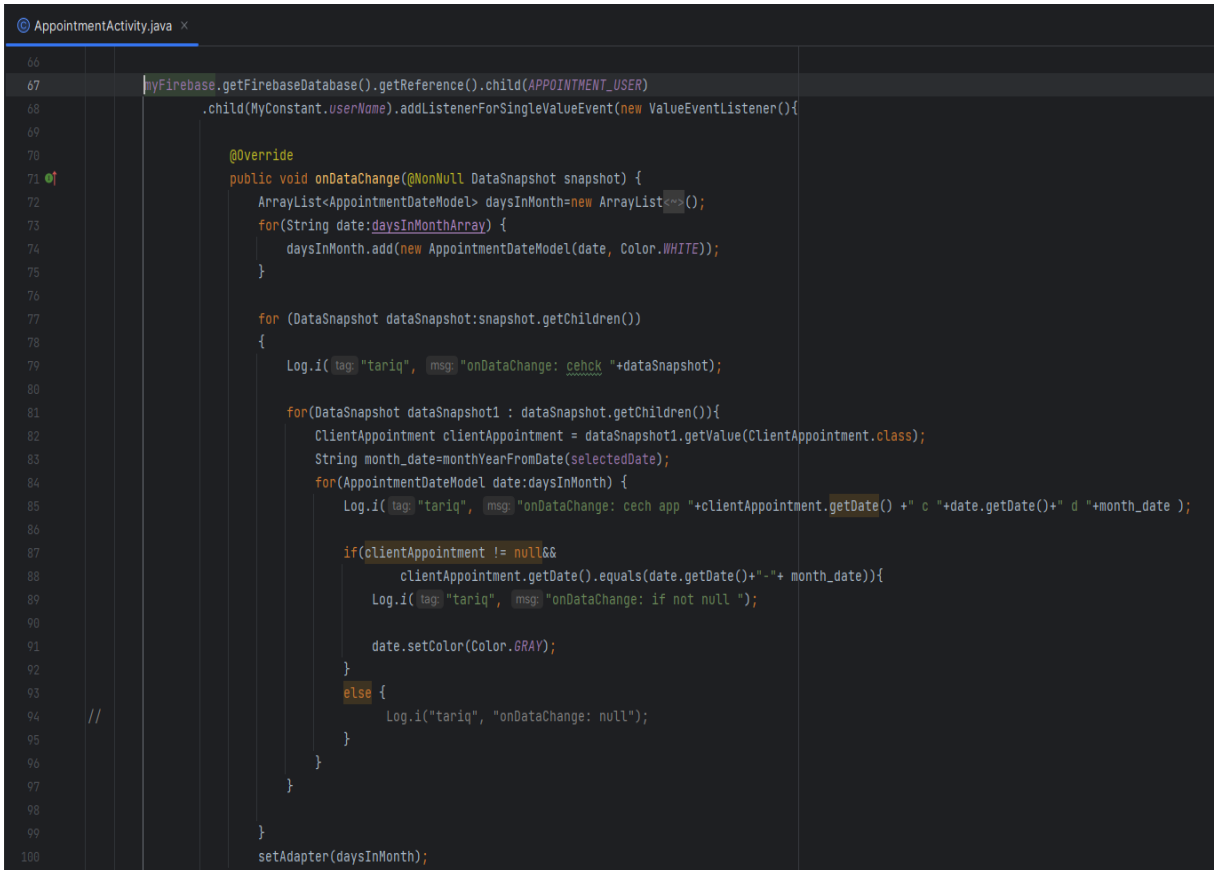


Στην γραμμή 546 πηγαίνει και εντοπίζει τον κόμβο με το e-mail του χρήστη και ότι έχει στο HashMap και έχει αλλάξει τιμή πηγαίνει και ενημερώνει τον αντίστοιχο κόμβο.

AppointmentActivity:



Στην γραμμή 67 δεσμέυεται το αντικείμενο που έχει δημιουργηθεί νωρίτερα με τον αντίστοιχο κόμβο που υπάρχει στη βάση με την ετικέτα ***APPOINTMENT_USER.***Η μεταβλητή ***APPOINTMENT_USER*** αντιστοιχεί στο email του χρήστη που θέλει να κλείσει ραντεβού.Φέρνει όλο το πρόγραμμα για τα ραντεβού και όπου υπάρχει διαθεσιμότητα θέτει σε άσπρο χρώμα και όπου δεν υπάρχει θέτει σε γκρί χρώμα.

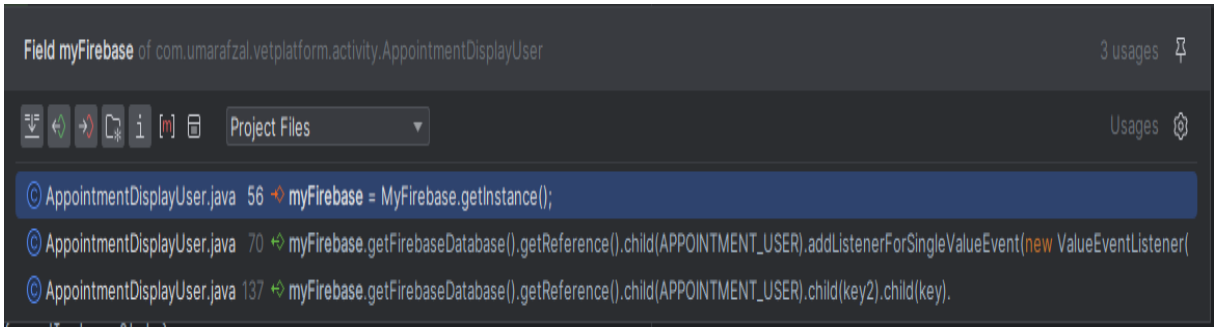


AppointmentActivity.java

```
193            myFirebase.getFirebaseDatabase().getReference().child(APPOINTMENT_USER).addListenerForSingleValueEvent(new ValueEventListener() {
194                @Override
195                public void onDataChange(@NonNull DataSnapshot snapshot) {
196                    if (snapshot.exists())
197                    {
198                        Intent intent=new Intent( packageContext: AppointmentActivity.this,AppointmentDisplayUser.class);
199
200                        for (DataSnapshot dataSnapshot:snapshot.getChildren()) {
201
202                            for (DataSnapshot dataSnapshot1:dataSnapshot.getChildren())
203                            {
204                                ClientAppointment clientAppointment = dataSnapshot1.getValue(ClientAppointment.class);
205                                if (clientAppointment.getDate().equals(dayText+"-"+monthYearFromDate(selectedDate)))
206                                {
207                                    goNext = true;
208                                    Log.i( tag: "rafagat",  msg: "onDataChange: "+clientAppointment.getDate());
209                                    String dateee =clientAppointment.getDate();
210    //                                Toast.makeText(AppointmentActivity.this, "checkkkkkkkkkk", Toast.LENGTH_SHORT).show();
211                                    intent.putExtra( name: "datee",dateee);
212                                    finish();
213
214
215                                }
216    //                            else
217    //                            {
218    //                                Toast.makeText(AppointmentActivity.this, "No Appointment this Date", Toast.LENGTH_SHORT).show();
219    //                            }
220                            }
221
222                        }
```

Στο σημείο αυτό ο χρήστης με το συμβάν πάτημα κουμπιού επάνω στο ραντεβού μέσω Intent μεταφέρρεται σε μια ξεχωριστή οθόνη(AppointmentDisplayUser).Οπου εμφανίζονται σε μορφή λίστας οι λεπτομέριες από τις διάφορες κρατήσεις που έχει κάνει ο χρήστης.

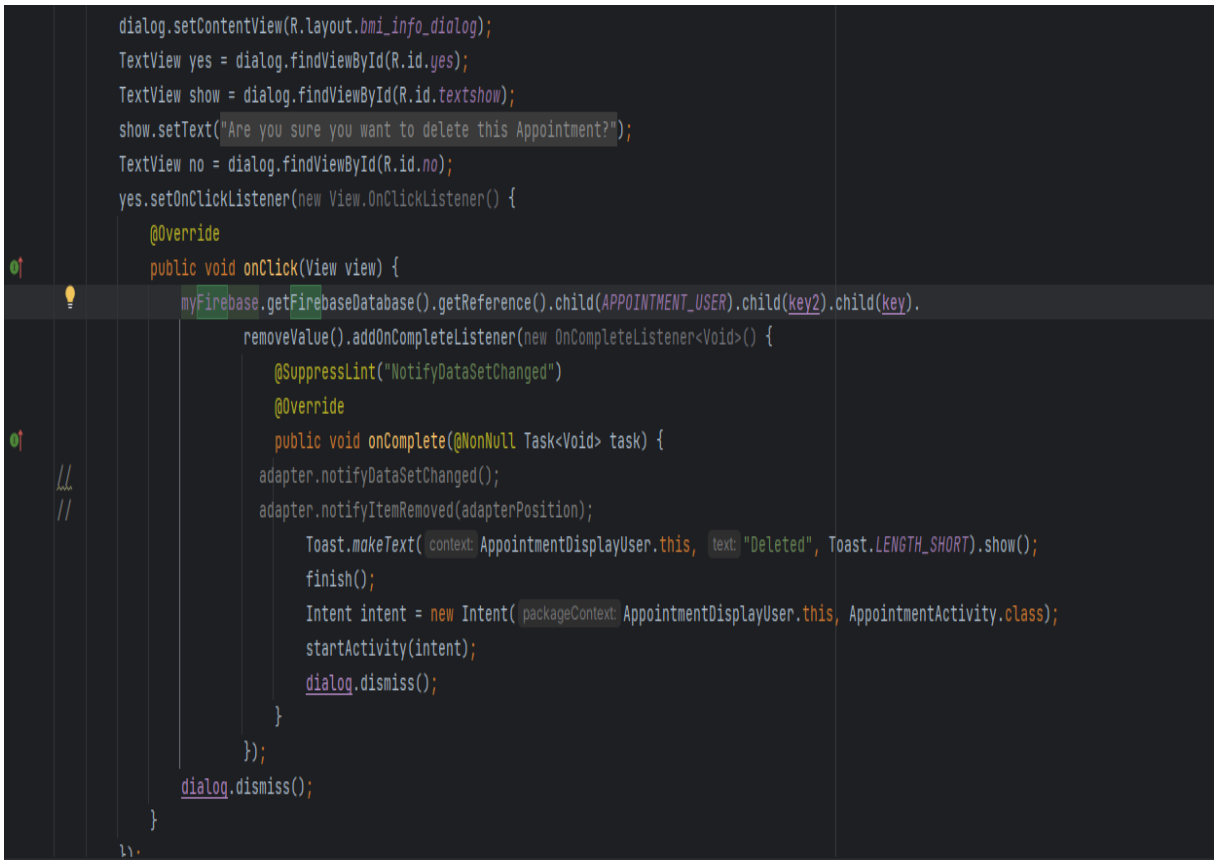
AppointmentDisplayUser:



Στην παραπάνω εικόνα εμφανίζονται τα σημεία όπου γίνεται η επικοινωία με τη βάση.Στην πρώτη γραμμή μέσα στη μέθοδο "**OnCreate"** δημιουργείται το αντικείμενο δηλαδή μια αναφορά στη βάση δεδομένων όπου υπάρχουν τα δεδομένα μας.

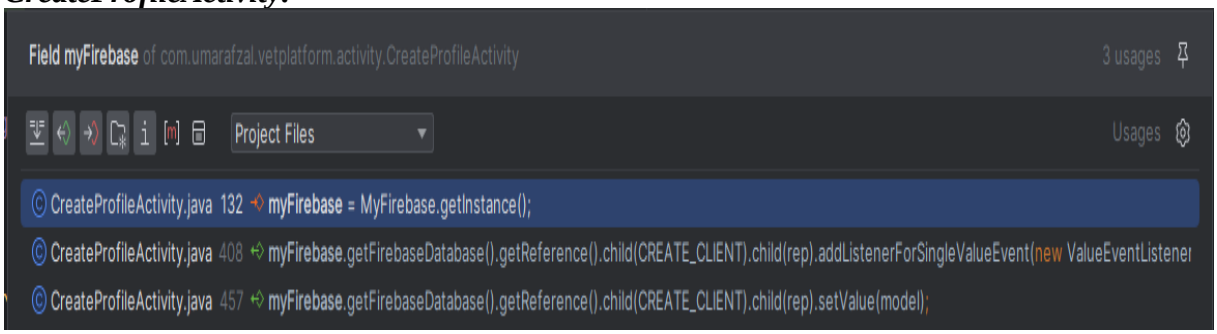
```
63
64          list = new ArrayList<>();
65          adapter = new AppointmentUserShowAdapter( context: this, list,  myInterface: this);
66          LinearLayoutManager linearLayoutManager = new LinearLayoutManager( context: this);
67          linearLayoutManager.setOrientation(RecyclerView.VERTICAL);
68          binding.recycleView.setLayoutManager(linearLayoutManager);
69
70          myFirebase.getFirebaseDatabase().getReference().child(APPOINTMENT_USER).addListenerForSingleValueEvent(new ValueEventListener() {
71              @Override
72              public void onDataChange(@NonNull DataSnapshot snapshot) {
73                  list.clear();
74                  if (snapshot.exists()) {
75                      for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
76                          {
77                              for (DataSnapshot dataSnapshot1 : dataSnapshot.getChildren()) {
78
79                                  ClientAppointment clientInfo = dataSnapshot1.getValue(ClientAppointment.class);
80                                  if (clientInfo.getDate().equals(b.get("datee"))) {
81                                      list.add(clientInfo);
82                                      binding.recycleView.setAdapter(adapter);
83                                      adapter.notifyDataSetChanged();
84                                      progressDialog.dismiss();
85                                  }
86
87                              }
88                          }
89
90                      }
91                  } else {
92                      progressDialog.setTitle("VetPlatformA");
93                      progressDialog.setMessage("No Appointment Found in this Date");
94                      progressDialog.dismiss();
95                  }
96
97
98          }
```

Στη σημείο αυτό τα διαθέσιμα ραντεβού του χρήστη για την συγεκεριμένη ημερομηνία μεταφέρεται από τη βάση και εμφαζίνονται μεσα σε **RecycleView(Λίστα).**Στο σημείο αυτό ο χρήστης μπορεί να δει τις λεπτομέρειες για τα διάφορα ραντεβού που έχει κλείσει.



Ο χρήστης έαν θέλει να διαγράψει κάποιο από τα διαθέσιμα ραντεβού που εμφανίζονται στη λίστα με το id του συγκεκριμένου ραντεβού γίνεται αναφορά στα δεδομένα που υπάρχουν στην απομακρυσμένη βάση και εκτελείτε η διαδικασία της εγγραφής.

CreateProfileActivity:

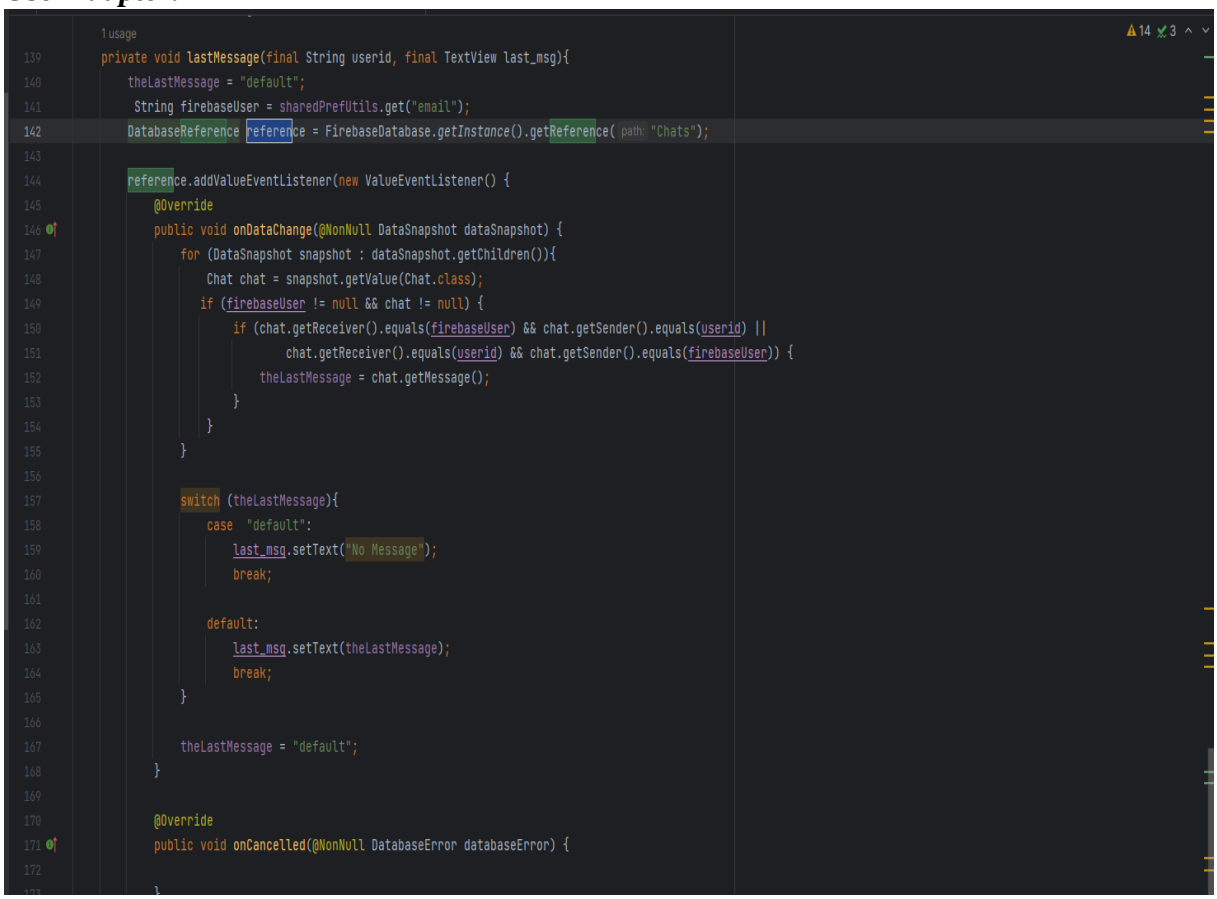


Στην κλάση CreateProfile σε τρία σημεία γίνεται αναφορά στο αντικείμενο(Instance) τύπου myFirebase .Στην γραμμή 132 γίνεται η αρχικοποίηση του αντικειμένου.

```
                //Send Data in Firebase
                myFirebase.getFirebaseDatabase().getReference().child(CREATE_CLIENT).child(rep).setValue(model);
                reference = FirebaseDatabase.getInstance().getReference( path: "Users").child(email.replace( target: ".", replacement: ""));
                HashMap<String, String> hashMap = new HashMap<>();
                hashMap.put("id", email.replace( target: ".", replacement: ""));
                hashMap.put("username", userName);
                hashMap.put("imageURL", "default");
                hashMap.put("status", "offline");
                hashMap.put("bio", "");
                hashMap.put("search", userName.toLowerCase());
                reference.setValue(hashMap).addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        if (task.isSuccessful()) {
                            Intent intent = new Intent( packageContext: CreateProfileActivity.this, MainActivity.class);
                            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
                            startActivity(intent);
                            finish();
                        }
                    }
                });

                progressDialog.dismiss();
                Toast.makeText( context: CreateProfileActivity.this, "Record Save in DB and QR Save in Gallery", Toast.LENGTH_SHORT).show();

            } catch (WriterException ex) {
                ex.printStackTrace();
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            }
        } else {
            progressDialog.dismiss();
//            Toast.makeText(CreateProfileActivity.this, R.string.contact_developer, Toast.LENGTH_SHORT).show();
        }
    }
```

Στο σημείο αυτό δημιουργούμε το προφίλ του Χρήστη και γίνεται η αποθήκευση των στοιχείων του καινούριο χρήστη κάτω από το κόμβο User.

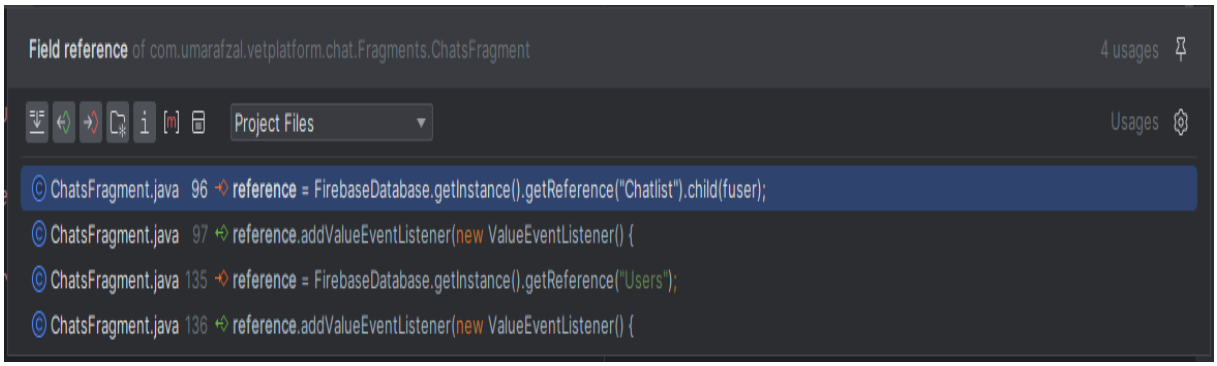
UserAdapter:



```
1 usage
private void lastMessage(final String userid, final TextView last_msg){
    theLastMessage = "default";
     String firebaseUser = sharedPrefUtils.get("email");
    DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Chats");

    reference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            for (DataSnapshot snapshot : dataSnapshot.getChildren()){
                Chat chat = snapshot.getValue(Chat.class);
                if (firebaseUser != null && chat != null) {
                    if (chat.getReceiver().equals(firebaseUser) && chat.getSender().equals(userid) ||
                            chat.getReceiver().equals(userid) && chat.getSender().equals(firebaseUser)) {
                        theLastMessage = chat.getMessage();
                    }
                }
            }

            switch (theLastMessage){
                case "default":
                    last_msg.setText("No Message");
                    break;

                default:
                    last_msg.setText(theLastMessage);
                    break;
            }

            theLastMessage = "default";
        }

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

        }
```

Εμφανίζεται η λίστα συνομιλιών των διάφορων χρηστών.

ChatsFragment:



Εμφανίζεται η οθόνη των συνομιλίων για το συγκεκριμένο χρήστη.

UsersFragment:

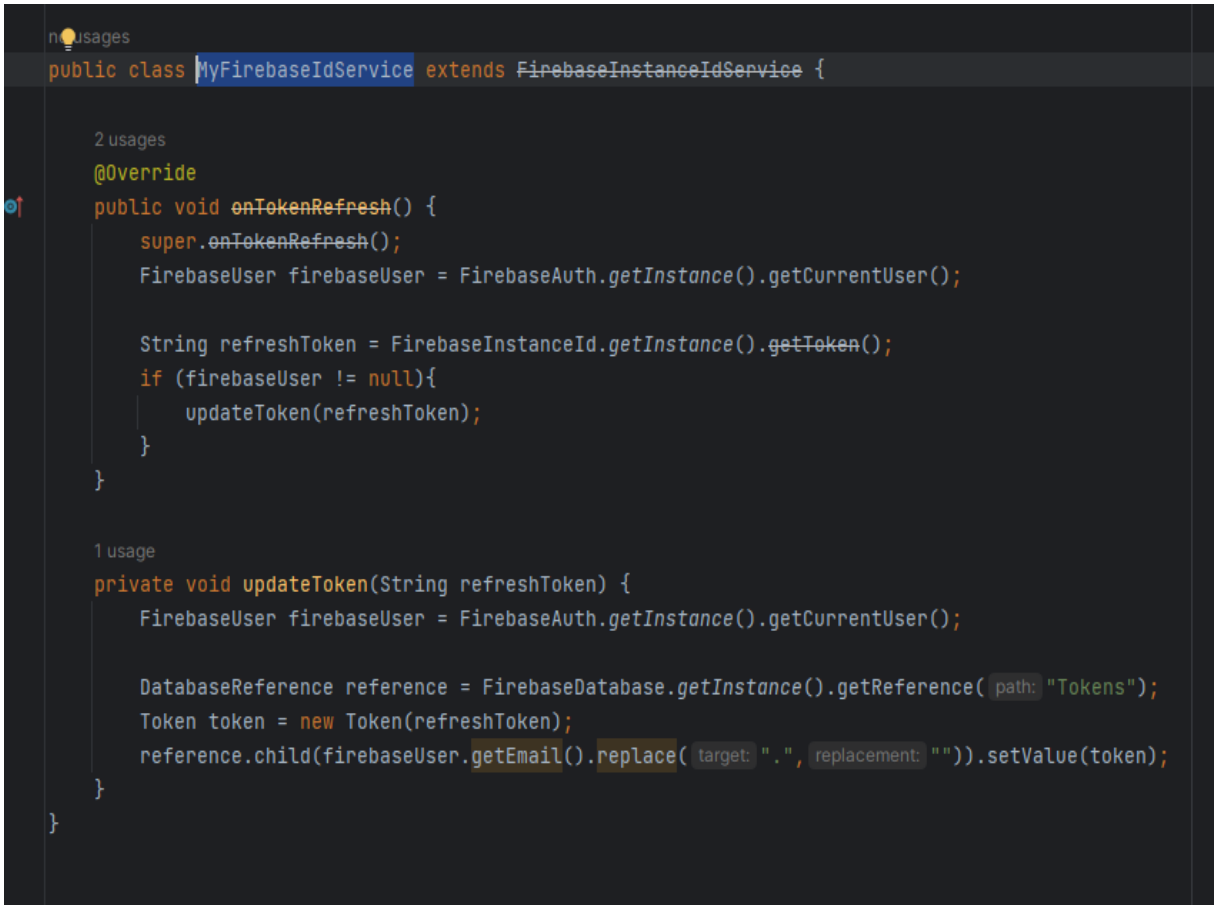
```
1 usage
private void readUsers() {

    String firebaseUser = sharedPrefUtils.get("email");
    DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Users");

    reference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if (search_users.getText().toString().equals("")) {
                mUsers.clear();
                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    User user = snapshot.getValue(User.class);

                    if (user!= null && user.getId()!=null && firebaseUser!=null && !user.getId().equals(firebaseUser)) {
                        mUsers.add(user);
                    }
                }

                if(mUsers.size()==0){
                    frameLayout.setVisibility(View.VISIBLE);
                }
                else{
                    frameLayout.setVisibility(View.GONE);
                }

                userAdapter = new UserAdapter(getContext(), onItemClick,mUsers,  ischat: false);
                recyclerView.setAdapter(userAdapter);
            }
        }
```

Με βάση το email του χρήστη μεταφέρονται όλα τα δεδομένα που αφορούν τον συγκεκριμένο χρήστη.

```
111         1 usage
            private void searchUsers(String s) {
112
113                 String fuser = sharedPrefUtils.get("email").replace( target: ".", replacement: "");
114                 Query query = FirebaseDatabase.getInstance().getReference( path: "Users").orderByChild( path: "search")
115                         .startAt(s)
116                         .endAt( value: s+"\uf8ff");
117
118                 query.addValueEventListener(new ValueEventListener() {
119                     @Override
120                     public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
121                         mUsers.clear();
122                         for (DataSnapshot snapshot : dataSnapshot.getChildren()){
123                             User user = snapshot.getValue(User.class);
124
125                             assert user != null;
126                             assert fuser != null;
127                             if (!user.getId().equals(fuser)){
128                                 mUsers.add(user);
129                             }
130                         }
131
132                         userAdapter = new UserAdapter(getContext(),onItemClick, mUsers,  ischat: false);
133                         recyclerView.setAdapter(userAdapter);
134                     }
135
136                     @Override
137                     public void onCancelled(@NonNull DatabaseError databaseError) {
138
139                     }
140                 });
141
142             }
```

Γίνεται αναζήτη όλων των στοιχείων του συγκεριμένου χρήστη .

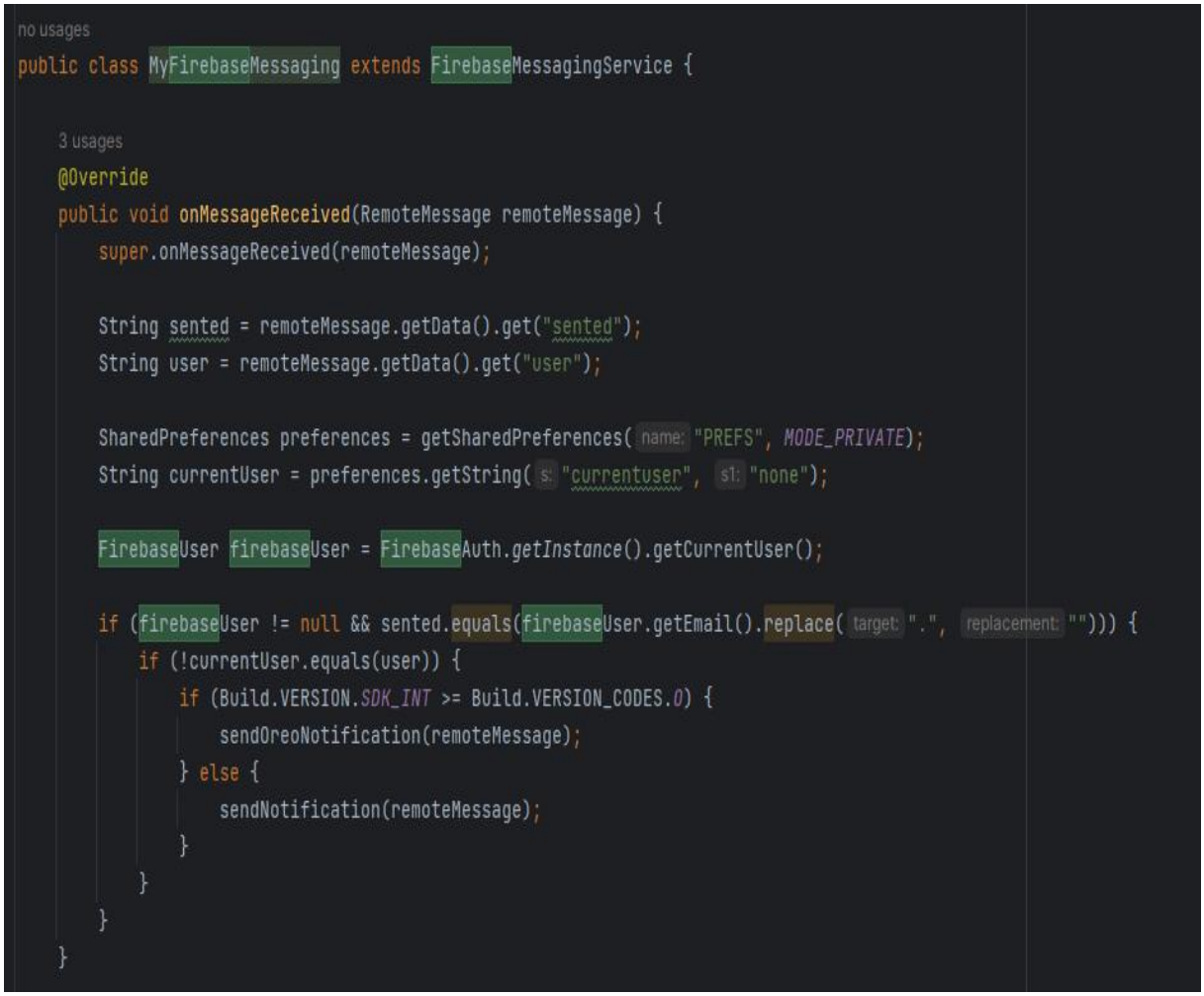
MyFirebaseIdService:



```
n usages
public class MyFirebaseIdService extends FirebaseInstanceIdService {

    2 usages
    @Override
    public void onTokenRefresh() {
        super.onTokenRefresh();
        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();

        String refreshToken = FirebaseInstanceId.getInstance().getToken();
        if (firebaseUser != null){
            updateToken(refreshToken);
        }
    }

    1 usage
    private void updateToken(String refreshToken) {
        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();

        DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Tokens");
        Token token = new Token(refreshToken);
        reference.child(firebaseUser.getEmail().replace( target: ".", replacement: "")).setValue(token);
    }
}
```

Η κλάση αυτή διαθέτει δύο μεθόδους.Η μία διαβάζει από τη βάση τα tokens του κάθε χρήστη και κάνει σύγκριση με το ήδη υπάρχον στη βάση και χρησιμοποιεί κάθε φορά που κάνει μεταφορά δεδομένων για ασφλαή λειτουργία της εφαρμογής μας.

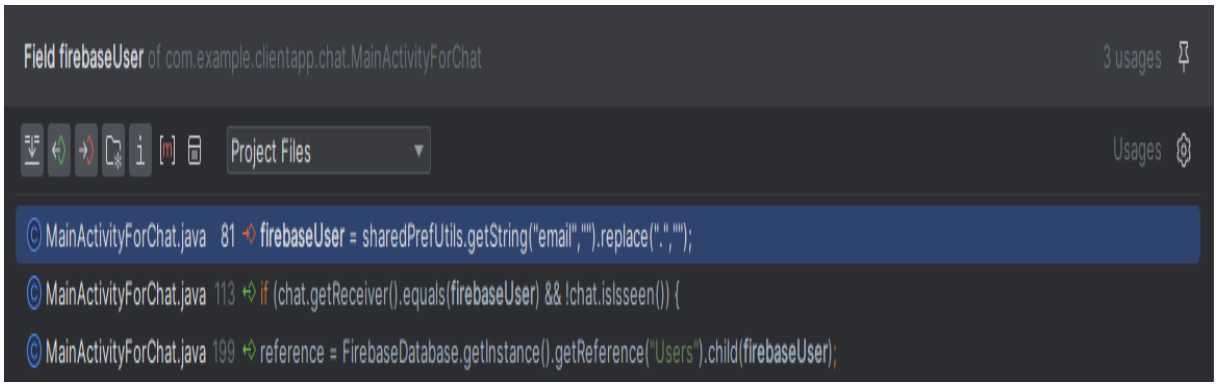
MyFirebaseMessaging:

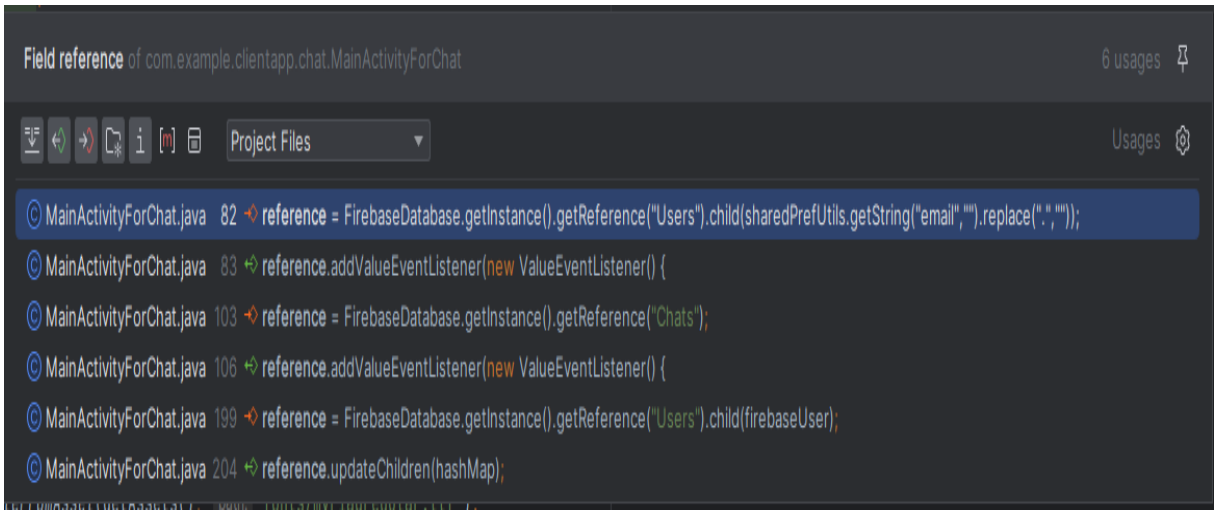


```
no usages
public class MyFirebaseMessaging extends FirebaseMessagingService {

    3 usages
    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        super.onMessageReceived(remoteMessage);

        String sented = remoteMessage.getData().get("sented");
        String user = remoteMessage.getData().get("user");

        SharedPreferences preferences = getSharedPreferences( name: "PREFS", MODE_PRIVATE);
        String currentUser = preferences.getString( s: "currentuser",  s1: "none");

        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();

        if (firebaseUser != null && sented.equals(firebaseUser.getEmail().replace( target: ".",  replacement: ""))) {
            if (!currentUser.equals(user)) {
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
                    sendOreoNotification(remoteMessage);
                } else {
                    sendNotification(remoteMessage);
                }
            }
        }
    }
```

Στο σημείο αυτό κάθε φορά που φτάνει ένα καινούριο μηνυμα από καποιο πελάτη εμφανίζεται μια ειδοποίηση (Notification)στην εφαρμογή μας.

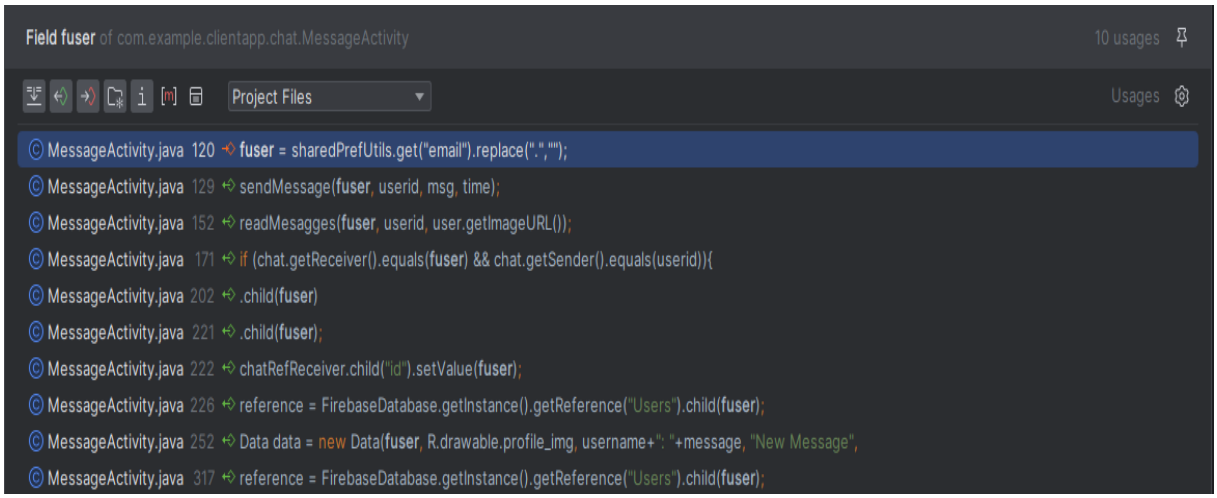
MainActivityForChat:

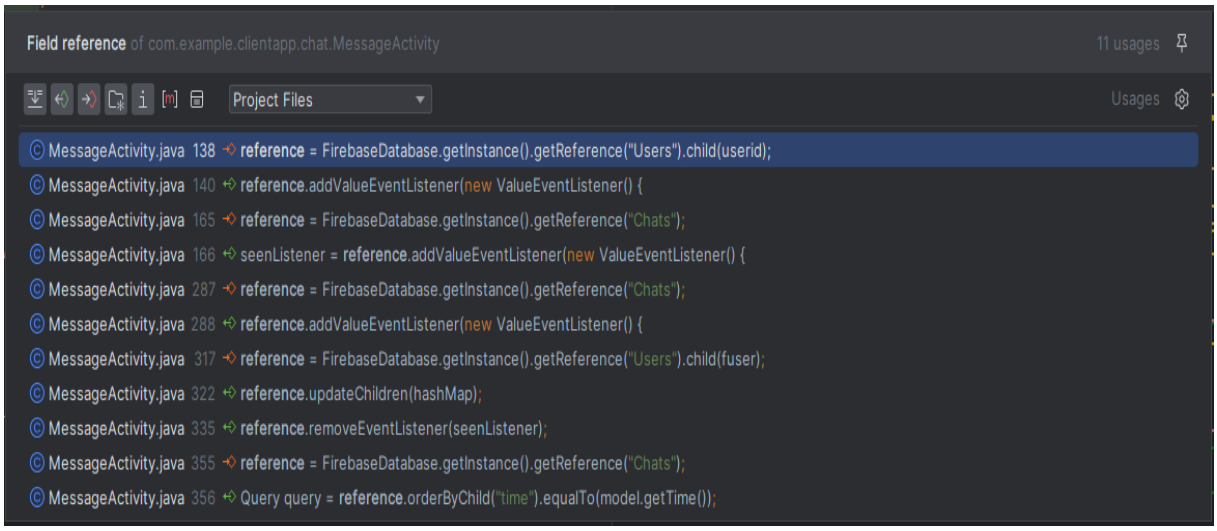




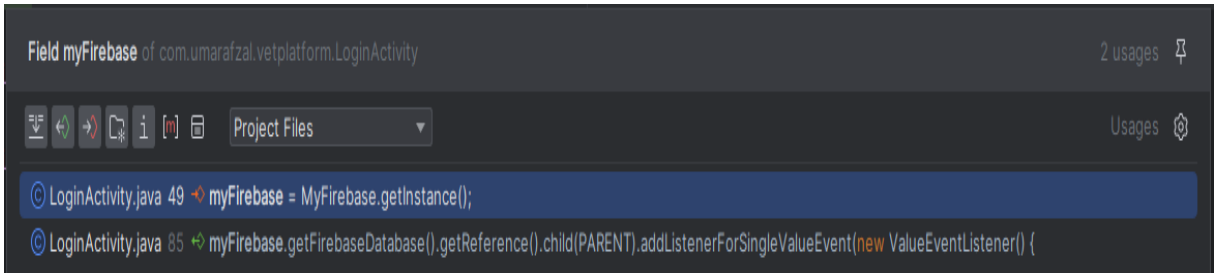
Στην παραπάνω εικόνα βλέπουμε όπου γίνεται οι επικοινωνία με τη βάση και μεταφέρονται όλα τα δεδομένα που ανήκουν στην υποκατηγορία του τίτλου μεταφέρονται και συμπληρώνονται στα κατάλληλα πεδία.

MessageActivity:





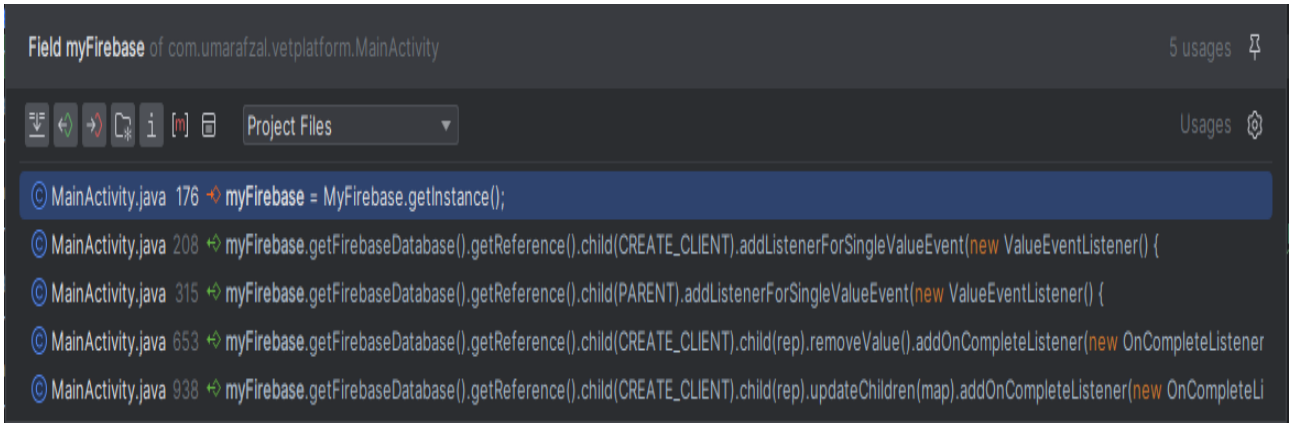
LoginActivity:



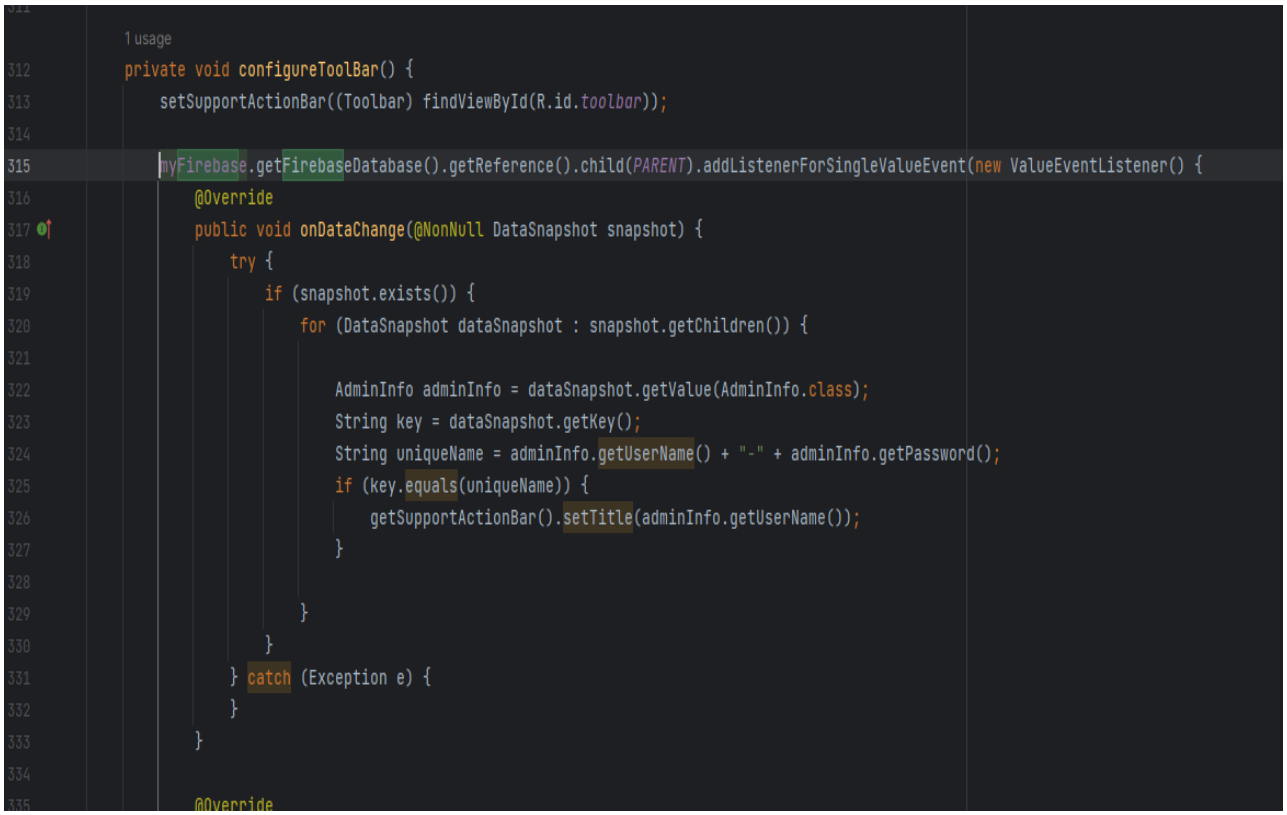

```
private void loginMethod(String email, String password) {
    mProgressDialogue.setTitle("Signing In");
    mProgressDialogue.setMessage("Please wait while we are get information.....");
    mProgressDialogue.setCanceledOnTouchOutside(false);
    mProgressDialogue.show();
    myFirebase.getFirebaseDatabase().getReference().child(PARENT).addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            try {
                if (snapshot.exists()) {
                    for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                        AdminInfo adminInfo=dataSnapshot.getValue(AdminInfo.class);
                        String key=dataSnapshot.getKey();
                        String uniqueName=adminInfo.getUserName()+"-"+adminInfo.getPassword();
                        Log.i( tag: "tariq",  msg: "onDataChange: "+email + " user "+adminInfo.getUserName());
//                          || adminInfo.getUserName().toLowerCase().equals(email.toLowerCase())
                        if (adminInfo.getEmail().trim().toLowerCase().equals(email.trim().toLowerCase())) {
                            if (adminInfo.getPassword().equals(password)) {

                                Intent intent = new Intent( packageContext: LoginActivity.this, MainActivity.class);
//                                  intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                                //intent.putExtra("email", str_email);
                                startActivity(intent);
                                sharedPrefUtils.setString("email",email.replace( target: ".",  replacement: "").trim());
                                finish();
                                mProgressDialogue.dismiss();
                            }
                            else
                            {
                                mProgressDialogue.dismiss();
                                Toast.makeText( context: LoginActivity.this, "Wrong Password", Toast.LENGTH_SHORT).show();
                            }


                        } else {
                            mProgressDialogue.dismiss();
                            binding.emailET.setError("Email not registered !");
                            binding.emailET.requestFocus();
                        }

                    }
                }
```

Είναι η συνάρτηση που εκτελεί τη λειτουργία της ταυτοποίησης του χρήστη.Συγκρίνει με τα αποθηκευμένα στοιχεία στη βάση και αν τα στοιχεία αυτά ταυτοίζονται μεταξύ τους μέσω ***Intent*** μεταφέρεται στην κεντιρκή οθόνη της εφαρμογής μας.Διαφορετικά εμφανίζεται το μήνυμα στον χρήστη ότι έχει καταχωρήση λάθος στοιχεία και να επαναλάβει την διαδικασία της ταυτοποίησης.

MainActivity:



Στην παραπάνω εικόνα βλέπουμε αναλυτικά τα σημεία στα οποία γίνονται οι αναφορές στη βάση και διαθέτει τα κατάλληλα συμβάντα(Events) για την προσπέλαση των δεδομένων.



Στο σημείο αυτό γίνεται η προσπέλαση των στοιχείων του διαχειριστή από την απομακρυσμένη βάση.Συγκρίνει το id του διαχειριστή και αν αυτά ταιριάζουν θέτει το όνομα του διαχειρσιτή σε όλα τα σημεία.

```
final Dialog dialog = new Dialog( context: MainActivity.this);
Objects.requireNonNull(dialog.getWindow()).setBackgroundDrawableResource(android.R.color.transparent);
dialog.setContentView(R.layout.bmi_info_dialog);
TextView yes = dialog.findViewById(R.id.yes);
TextView show = dialog.findViewById(R.id.textshow);
show.setText("Are you sure you want to delete this user?");
TextView no = dialog.findViewById(R.id.no);
yes.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String rep = model.getEmail().replace( target: ".",  replacement: "");

        String key = model.getUserName() + "-" + model.getPassword();
        myFirebase.getFirebaseDatabase().getReference().child(CREATE_CLIENT).child(rep).removeValue().addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                clientShowAdapter.notifyDataSetChanged();
                clientShowAdapter.notifyItemChanged(pos);
                startActivity(getIntent());
                finish();
            }
        });
        dialog.dismiss();
    }
});
no.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) { dialog.dismiss(); }
});
dialog.show();

} catch (Exception e) {
    Log.i( tag: "tariq",  msg: "delete: " + e);

}
}
```

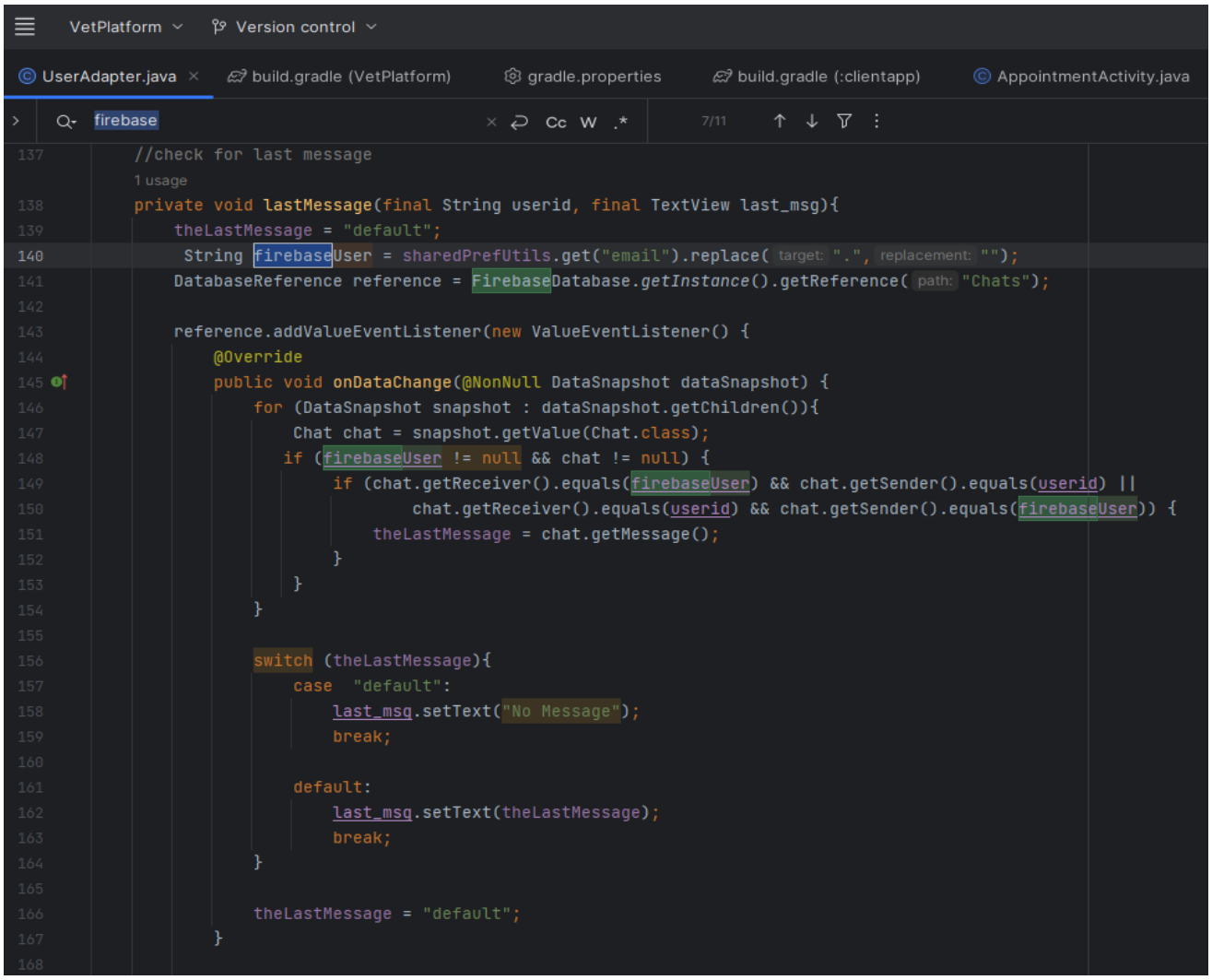
Μόλις ο διαχειριστής της εφαρμογής θα πατήσει για διαγραφή του χρήστη ενεργοποίειται το συμβάν που βλέπετε πάνω.Εμφανίζεται ο διάλογος που ζητάει από τον χρήστη επιβεβαίωση της πράξης που θέλει να εκτελέσει.Μόλις πατήσει ο χρήστης “Yes” εντοπίζει το όνομα του χρήστη και διαγράφει όλα τα στοιχεία που βρίσκονται κάτω από αυτό τον κόμβο.

```
937
938             myFirebase.getFirebaseDatabase().getReference().child(CREATE_CLIENT).child(rep).updateChildren(map).addOnCompleteListener(new OnCompleteListener<Void>() {
939                 @Override
940                 public void onComplete(@NonNull Task<Void> task) {
941                   /*          if(!userName.equals(model.getUserName()) ||! password.equals(model.getPassword())){
942                                 String key = model.getUserName() + "-" + model.getPassword();
943                                 myFirebase.getFirebaseDatabase().getReference().child(CREATE_CLIENT).child(key).
944                                         removeValue().addOnCompleteListener(new OnCompleteListener<Void>() {
945                                             @Override
946                                             public void onComplete(@NonNull Task<Void> task) {
947                                                 clientShowAdapter.notifyDataSetChanged();
948                                             }
949                                         });
950                                 }*/
951
952                     alertDialog.dismiss();
953                     Toast.makeText( context: MainActivity.this, "Updated Successfully", Toast.LENGTH_SHORT).show();
954                     clientShowAdapter.notifyDataSetChanged();
955                     recreate();
956
957                 }
958
959             });
960         }
961     });
```

Στην παραπάων εικόνα έαν έχει αλλάξει κάποιο στοιχείο στην διεπαφή του χρήστη αποθηκέυεται στην απομακρυσμένη βάση.

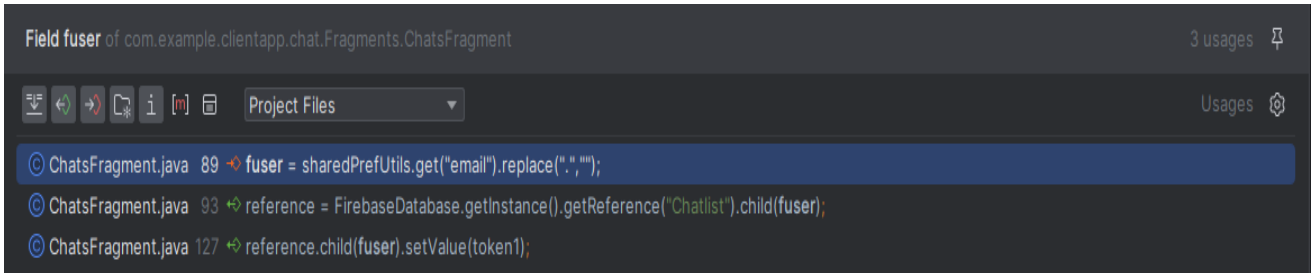
ClientApp Κλάσεις όπου γίνεται επικοινωνία με τη βάση:

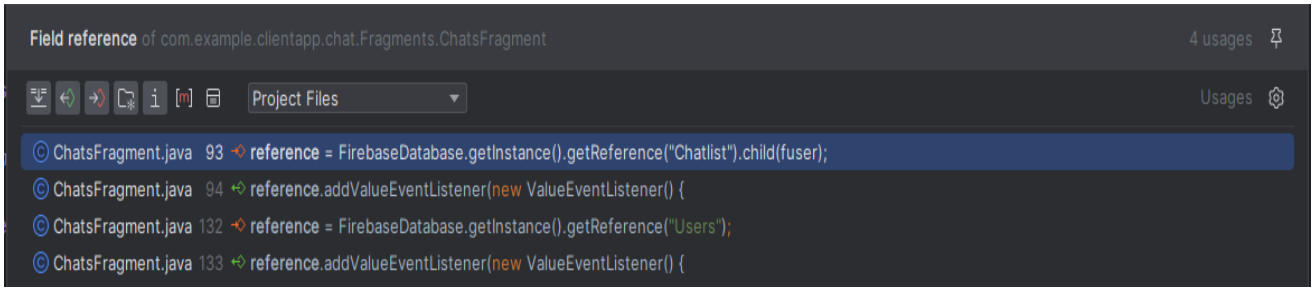
UserAdapter:

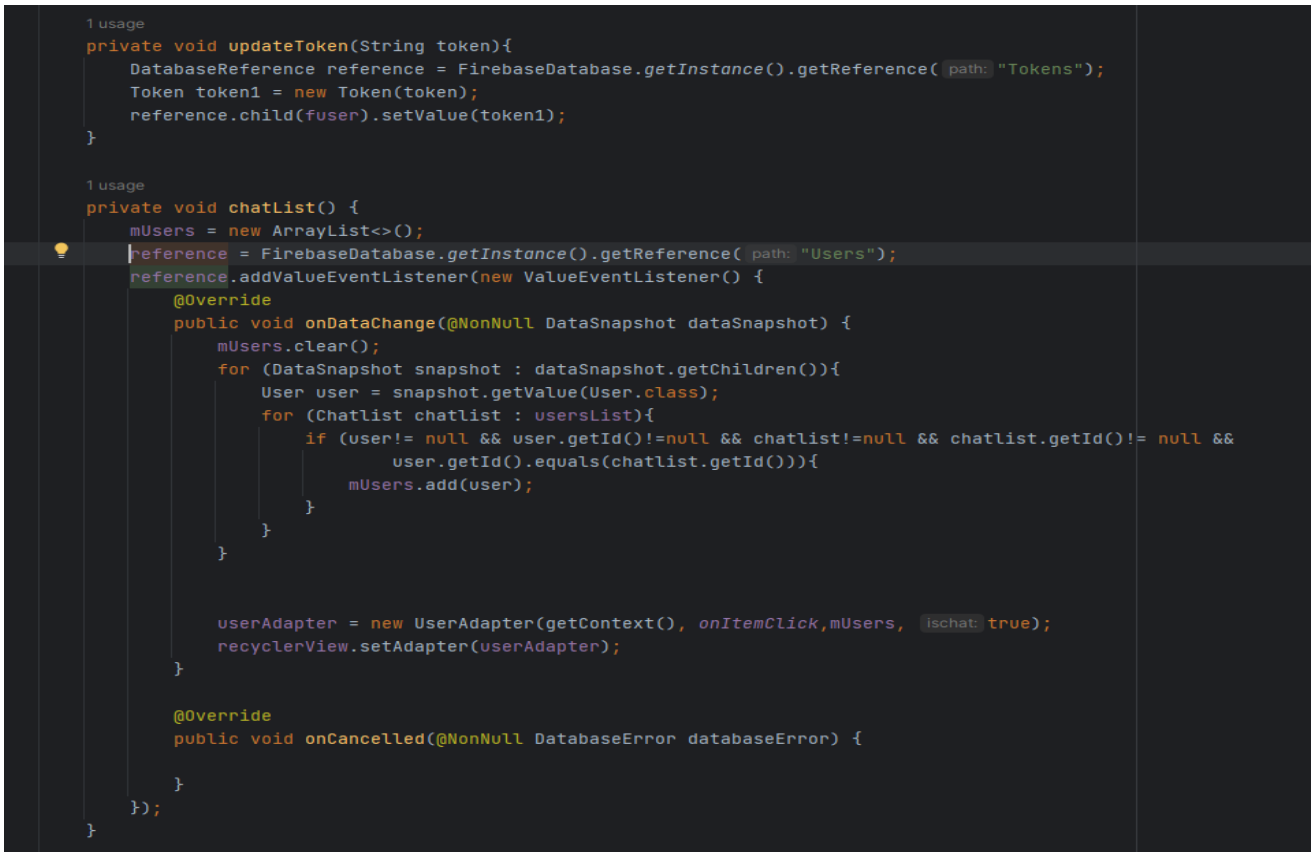


Μέσα στη μέθοδο LastMessage γίνεται οι επικινωνία με τη βάση δεδομένων.Στη βάση όσα δεδομένα που συσχετίζονται με τον κόμβο Chats μεταφέρονται μέσα στη μέθοδο αυτή και αυτά που αφορούν τον συγκεκριμένο χρήστη (με βάση το Id) εμφανίζεται στην αντίστοιχή εφαρμογή του χρήστη ότι έχει κάποιο μήνυμα .

ChatFragment:







```
1 usage
private void updateToken(String token){
    DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Tokens");
    Token token1 = new Token(token);
    reference.child(fuser).setValue(token1);
}

1 usage
private void chatList() {
    mUsers = new ArrayList<>();
    reference = FirebaseDatabase.getInstance().getReference( path: "Users");
    reference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            mUsers.clear();
            for (DataSnapshot snapshot : dataSnapshot.getChildren()){
                User user = snapshot.getValue(User.class);
                for (Chatlist chatlist : usersList){
                    if (user!= null && user.getId()!=null && chatlist!=null && chatlist.getId()!= null &&
                            user.getId().equals(chatlist.getId())){
                        mUsers.add(user);
                    }
                }
            }


            userAdapter = new UserAdapter(getContext(), onItemClick,mUsers,  ischat: true);
            recyclerView.setAdapter(userAdapter);
        }

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

        }
    });
}
```

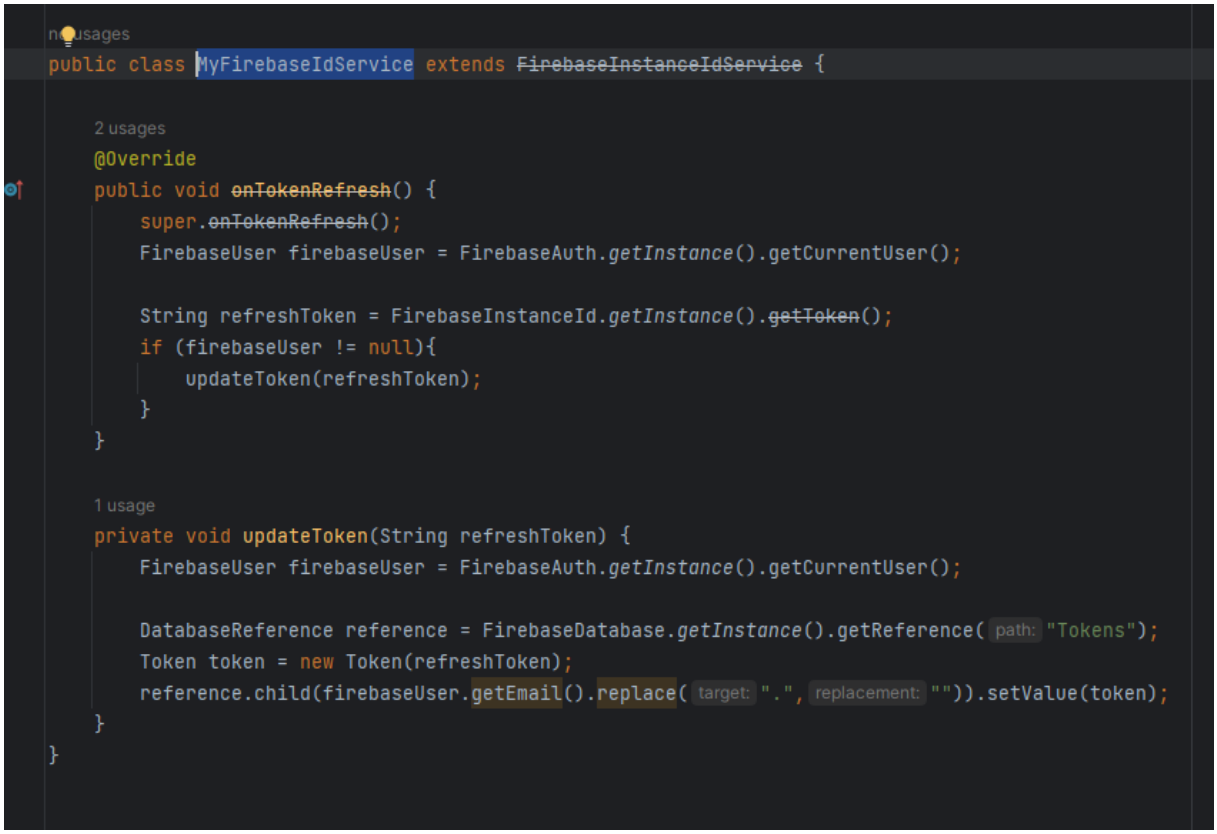
Μεταφέρονται όλα τα δεδομένα κάτω από το κόμβο Users και αποθηκέυονται σε μια λίστα.Διατρέχοντας τη λίστα των χρηστών επιλέγουμε τους χρήστες που τα απαραίτητα πεδία τη συνομιλία δεν είναι κενό τότε αυτό΄ς τους χρήστες αποθηκέυουμε σε μια ξεχωριστή λίστα.

UserFragment:

```
1 usage
private void updateToken(String token){
    DatabaseReference reference = FirebaseDatabase.getInstance().getReference( path: "Tokens");
    Token token1 = new Token(token);
    reference.child(fuser).setValue(token1);
}

1 usage
private void chatList() {
    mUsers = new ArrayList<>();
    reference = FirebaseDatabase.getInstance().getReference( path: "Users");
    reference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            mUsers.clear();
            for (DataSnapshot snapshot : dataSnapshot.getChildren()){
                User user = snapshot.getValue(User.class);
                for (Chatlist chatlist : usersList){
                    if (user!= null && user.getId()!=null && chatlist!=null && chatlist.getId()!= null &&
                            user.getId().equals(chatlist.getId())){
                        mUsers.add(user);
                    }
                }
            }


            userAdapter = new UserAdapter(getContext(), onItemClick,mUsers,  ischat: true);
            recyclerView.setAdapter(userAdapter);
        }
```

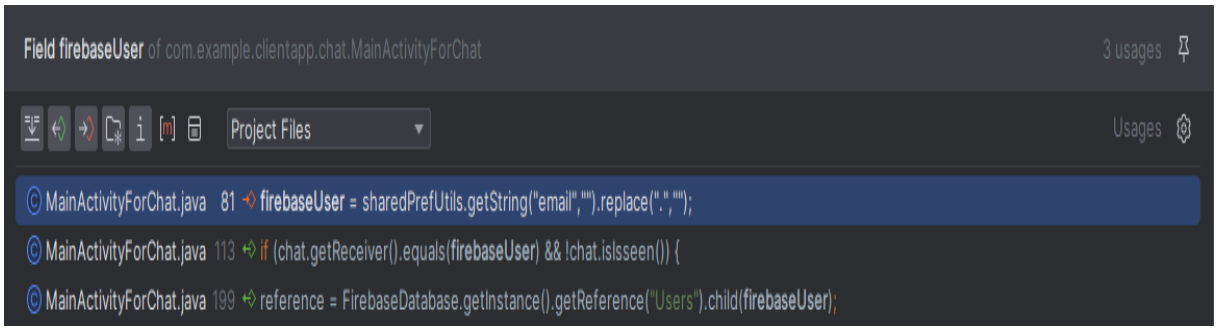
```
1 usage
private void readMesagges(final String myid, final String userid, final String imageurl){
    mchat = new ArrayList<>();
    messageAdapter = new MessageAdapter( mContext: MessageActivity.this, mchat, imageurl, myInterface: this);

    reference = FirebaseDatabase.getInstance().getReference( path: "Chats");
    reference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            mchat.clear();
            for (DataSnapshot snapshot : dataSnapshot.getChildren()){
                Chat chat = snapshot.getValue(Chat.class);
                if (chat.getReceiver().equals(myid) && chat.getSender().equals(userid) ||
                        chat.getReceiver().equals(userid) && chat.getSender().equals(myid)){
                    mchat.add(chat);
                }

                recyclerView.setAdapter(messageAdapter);
            }
        }

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

        }
    });
}
```

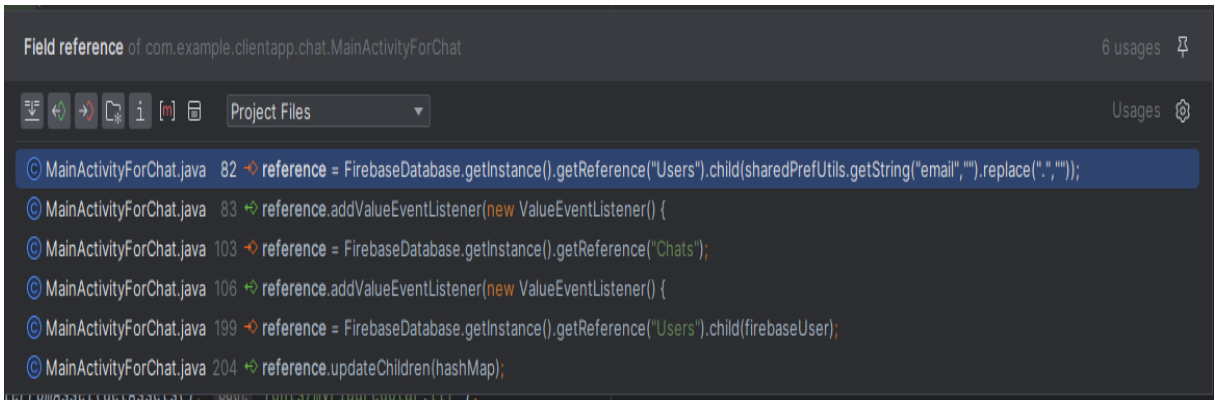
MyFirebaseIdService:



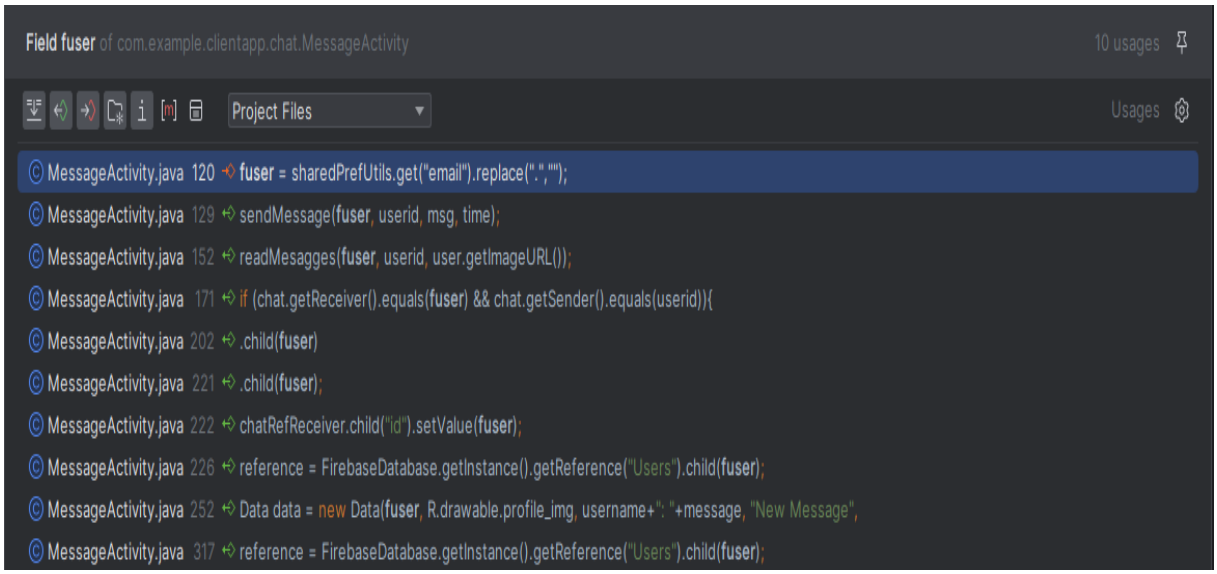
Η κλάση αυτή διαθέτει δύο μεθόδους.Η μία διαβάζει από τη βάση τα tokens του κάθε χρήστη και κάνει σύγκριση με το ήδη υπάρχον στη βάση και χρησιμοποιεί κάθε φορά που κάνει μεταφορά δεδομένων για ασφλαή λειτουργία της εφαρμογής μας.

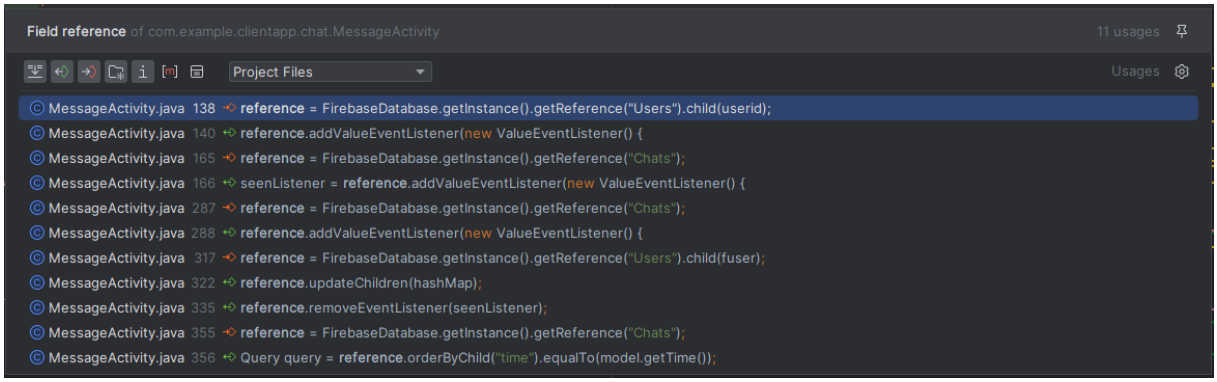
MainActivityForChat:





MessageActivity:





Η κλάση αυτή διαθέτει όλους τους μεθόδους για την ανταλλαγή των μηνυμάτων μεταξύ των χρηστών. Οι μέθοδοι αυτοί αφορού τις λειτουργίες όπως αποστολή και λήψη των μηνυμάτων αλλά αν ο χρήστης έκανε ανάγνωση η όχι.

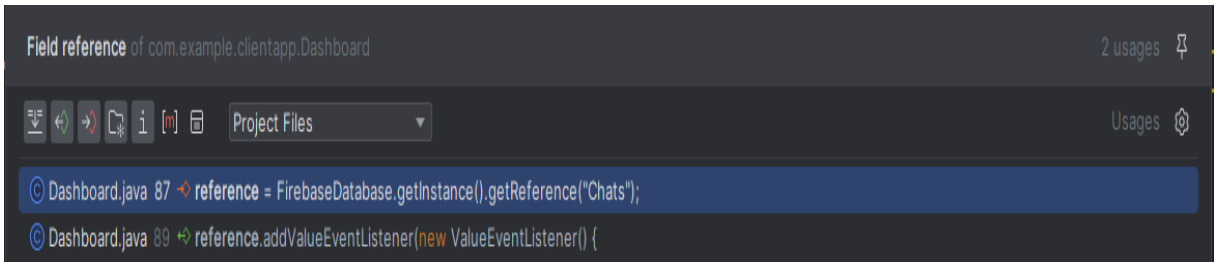
PetDisplay:

Στην παραπάνω εικόνα εμφανίζονται τα σημεία όπου υπάρχει επικοινωνία με τη βάση.Δημιουργούμε το αντικείμενο μέσα στη μέθοδο OnCreate.

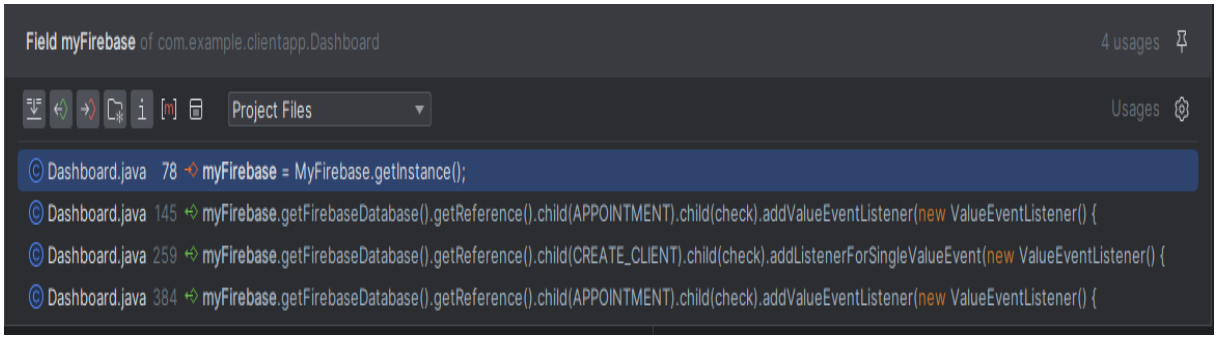
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding = DataBindingUtil.setContentView( activity: this, R.layout.activity_pet_display);
    languageManager = new LanguageManager( context: PetDisplay.this);

    sharedPrefUtils = new SharedPrefUtils( activity: this);
    myFirebase = MyFirebase.getInstance();
    progressDialog = new ProgressDialog( context: this);
    list = new ArrayList<>();

    LinearLayoutManager linearLayoutManager = new LinearLayoutManager( context: this);
    linearLayoutManager.setOrientation(RecyclerView.VERTICAL);
    binding.recycleView.setLayoutManager(linearLayoutManager);
    petdataShowAdapter = new PetDataAdapter( context: this, list, myInterface: this);

    myFirebase.getFirebaseDatabase().getReference().child( pathString: "PetData").child(sharedPrefUtils.get("email").replace( target: ".",  replacement: "")
            .trim()).addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            list.clear();
            if (snapshot.exists()) {
                for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                    Log.i( tag: "tariq",  msg: "dataSnapshot: " + dataSnapshot);

                    PetData clientInfo = dataSnapshot.getValue(PetData.class);
                    list.add(clientInfo);
                    binding.recycleView.setAdapter(petdataShowAdapter);
                    petdataShowAdapter.notifyDataSetChanged();
                    progressDialog.dismiss();


                }
```

Στην γραμμή 61 πηγαίνει στο κόμβο “PetData” και με βάση το email φέρνει όλα τα δεδομένα που αντιστοιχούν στο e-mail του συγκεκριμένου χρήστη και φορτώνει μέσα σε μια λίστα.Επειτα όλα τα δεδομένα που υπάρχουν στη λίστα δηλώνει σαν όρισμα στον προσαρμογέα της λίστας(RecycleView) για την εμφάνιση στον τελικό Χρήστη.

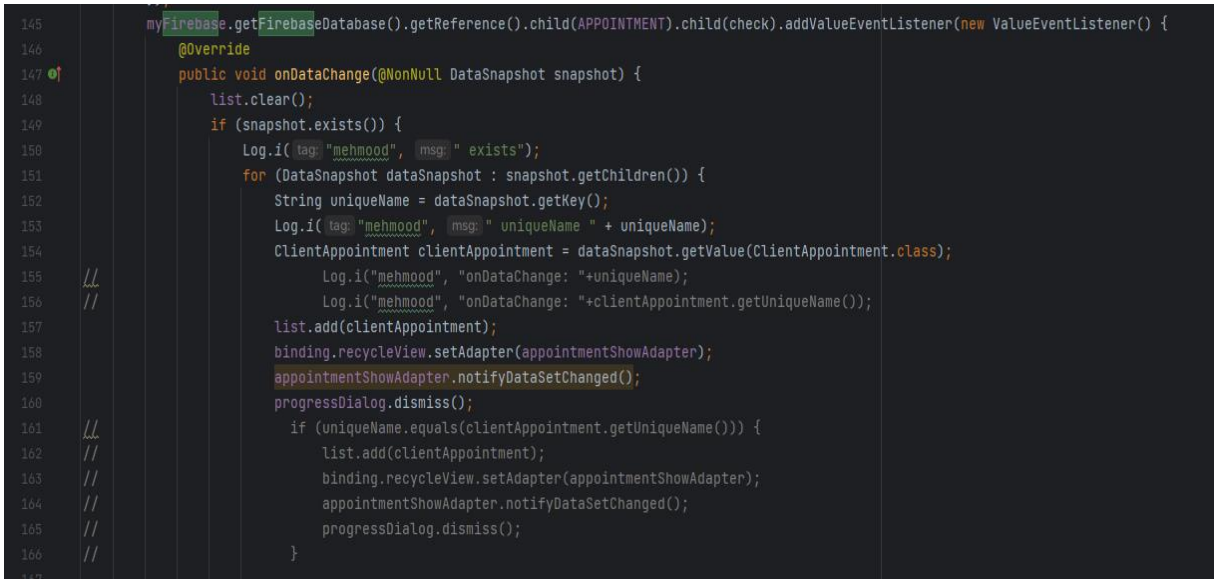
Dashboard:



Στην γραμμή 87 γίνεται η αναφορά και φέρνει όλα τα δεδομένα που βρίσκονται κάτω από το κόμβο “Chats”.

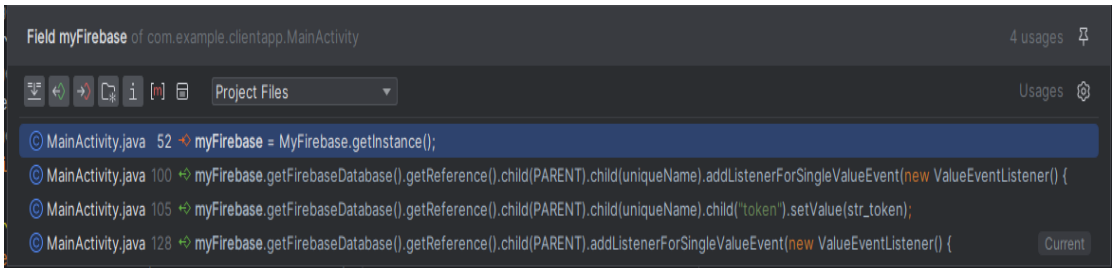


Η κλάση αυτή περιέχει όλες τις λειτουργίες της κεντρικής οθόνης.Οπως εμφάνιση των διαθέσιμων κρατήσεων του συγκεκριμένου χρήστη η προσθήκη νέου κατοικιδίου.



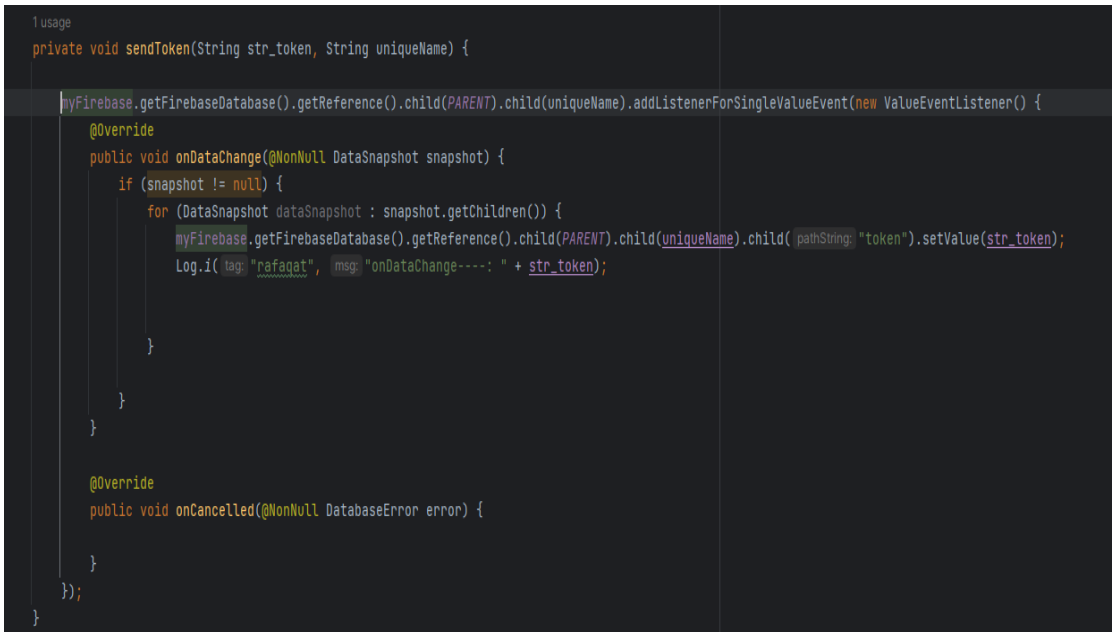
Στην παραπάνω εικόνα φέρνει όλα τα διαθέσιμα ραντεβού που έχει ο χρήστης και εμφανίζει σε μορφή της λίστας στην κεντρική οθόνη της εφαρμογής.Στην γραμμή 259 υπάρχει η μέθοδος που προσθέτει τα στοιχεία στην απομακρυσμένη βάση. Αυτή η μέθοδος εκτελείται κάθε φορά που ο χρήστης ανοίγει την εφαρμογή, για τη λήψη των πρόσφατων δεδομένων.

MainActivity:



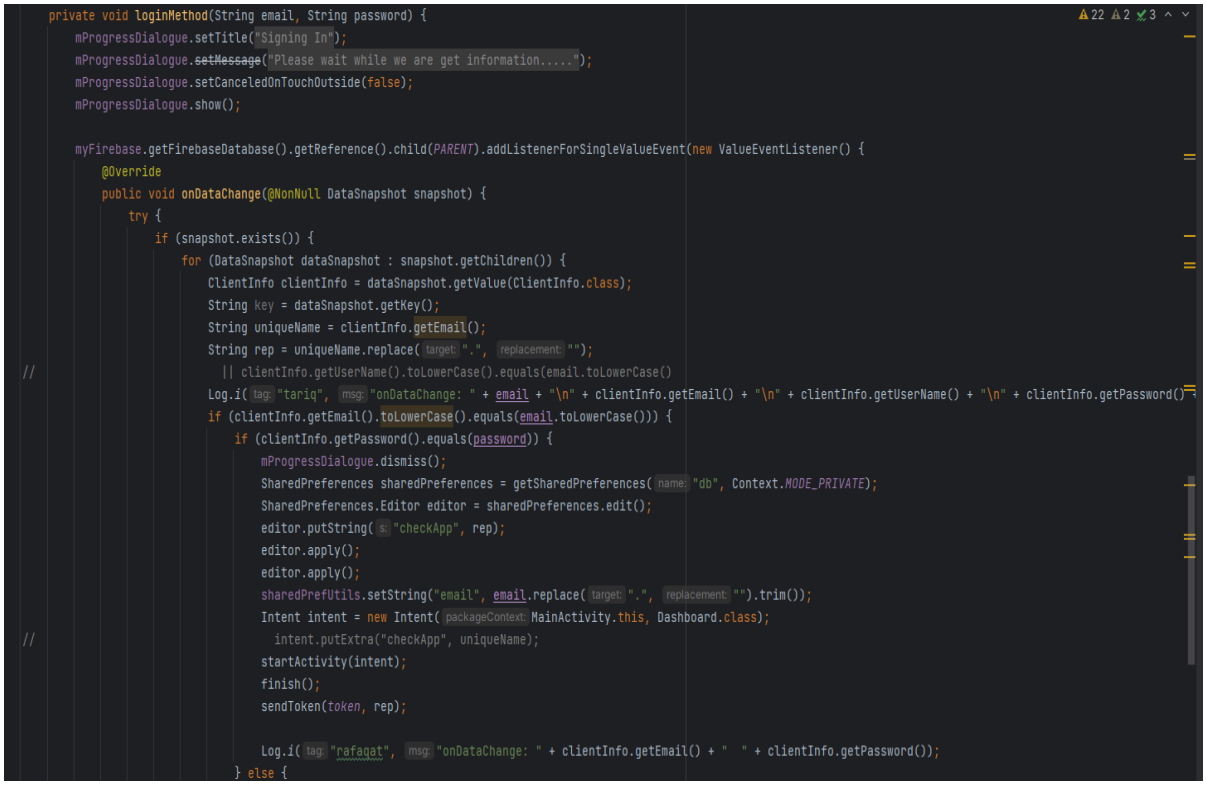
Στο σημείο αυτό στην κλάση MainActivty βλέπουμε τις αναφορές που έχουν γίνει στην απομακρυσμένη βάση .Στην γραμμή 52 δημιουργείται το αντικείμενο(Instanste) που περιέχει τις πληροφορίες για την απομακρυσμένη βάση που θέλουμε να συνδεθούμε.

Στην γραμμή 100 μεταφέροουν τα δεδομένα του συγκεκριμένου χρήστη.Στην γραμμή 105 μέσα στη μέθοδο sendToken γίνεται η αποστολή του Token(το κλειδί ασφαλείας του συγεκριμένου χρήστη)και προστίθεται κάτω από το κόμβο “token ”.



Στην παραπάνω εικόνα μέσα στο συμβλαν εγηξούμε τα βήματα επιβαιώσης των στοιχείων του χρήστη και αν τα δεδομένα στα κατάλληλα πεδία έχουν δηλωθεί σωστά αποθηκέυεται στην τελική βάση.

Η μέθοδος LoginMethod() περιέχει όλα τα κατάλληλα βήματα για την ταυτοποίηση των στοιχειών του χρήστη .Φέρνει από τη βάση όλα τα απαραίτητα δεδομένα και αν ο συγκεκριμένος χρήστης έχει καταχωρήσει τα στοιχεία του νωρίτερα στη βάση μας του επιτρέπει την πρόσβαση την κεντρική σελίδα της εφαρμογής μας.



```
private void loginMethod(String email, String password) {
    mProgressDialogue.setTitle("Signing In");
    mProgressDialogue.setMessage("Please wait while we are get information.....");
    mProgressDialogue.setCanceledOnTouchOutside(false);
    mProgressDialogue.show();

    myFirebase.getFirebaseDatabase().getReference().child(PARENT).addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            try {
                if (snapshot.exists()) {
                    for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                        ClientInfo clientInfo = dataSnapshot.getValue(ClientInfo.class);
                        String key = dataSnapshot.getKey();
                        String uniqueName = clientInfo.getEmail();
                        String rep = uniqueName.replace( target: ".",  replacement: "");
//                          || clientInfo.getUserName().toLowerCase().equals(email.toLowerCase()
                        Log.i( tag: "tariq",  msg: "onDataChange: " + email + "\n" + clientInfo.getEmail() + "\n" + clientInfo.getUserName() + "\n" + clientInfo.getPassword());
                        if (clientInfo.getEmail().toLowerCase().equals(email.toLowerCase())) {
                            if (clientInfo.getPassword().equals(password)) {
                                mProgressDialogue.dismiss();
                                SharedPreferences sharedPreferences = getSharedPreferences( name: "db", Context.MODE_PRIVATE);
                                SharedPreferences.Editor editor = sharedPreferences.edit();
                                editor.putString( s: "checkApp", rep);
                                editor.apply();
                                editor.apply();
                                sharedPrefUtils.setString("email", email.replace( target: ".",  replacement: "").trim());
                                Intent intent = new Intent( packageContext: MainActivity.this, Dashboard.class);
//                                  intent.putExtra("checkApp", uniqueName);
                                startActivity(intent);
                                finish();
                                sendToken(token, rep);

                                Log.i( tag: "rafaqat",  msg: "onDataChange: " + clientInfo.getEmail() + "   " + clientInfo.getPassword());
                            } else {
```

5.4 Σημεία ενδιαφέροντος

5.4.1 ΤΟ ΣΧΕΔΙΟ ΣΧΕΔΙΑΣΗΣ (DESIGN PATERNS) MVC

Το MVC είναι ένας όρος ο οποίος σημαίνει Model View Controller και αποτελεί την αρχιτεκτονική που χρησιμοποιεί το Android λειτουργικό σύστημα. Το MVC μοντέλο δεν θεωρείται μια καινούρια έννοια στον προγραμματισμό, αλλά ειδικά τα τελευταία χρόνια με την εξέλιξη του αντικειμενοστραφή προγραμματισμού έχει έρθει στην επικαιρότητα των προγραμματιστών και άρχισαν να καταλαβαίνουν τα πλεονεκτήματα που προσφέρει στους προγραμματιστές κατά την διάρκεια της συγγραφής κώδικα.

Ας ξεκινήσουμε αρχικά από το κομμάτι του μοντέλου (MODEL) της αρχιτεκτονικής MVC. Το μοντέλο έχει δημιουργηθεί ώστε να διευκολύνει την επικοινωνία της εφαρμογής μας με τη βάση δεδομένων και διευκολύνει τους προγραμματιστές να προσαρμόσουν τη δομή της σχεσιακής βάσης να προσαρμόσουν στην εφαρμογή τους. Είναι το κύριο μέρος του μοντέλου να διαχειρίζεται την ανάκτηση και την αποθήκευση δεδομένων στο σύστημα. Το Model στην ουσία αναπαριστά τα δεδομένα μας. Πολλές φορές οι κανόνες που ισχύουν στις σχεσιακές βάσεις δεδομένων είναι δύσκολο να εφαρμοστούν σε αντικειμενοστραφή προγραμματισμό. Όποτε στο σημείο αυτό ακολουθώντας το μοντέλο MVC αναπτύχθηκαν διάφοροι μέθοδοι και στρατηγικές για να διευκολύνουν την επικοινωνία της εφαρμογής με τη βάση χωρίς ιδιαίτερη προσπάθεια.

Το View είναι το οπτικό μέρος της εφαρμογής, έρχεται σε αλλεπίδραση με τον χρήστη μέσω των διάφορων διεπαφών που παρέχουν τα frameworks. Για την δημιουργία των διεπαφών για να αλληλεπιδράσουν με τους τελικούς χρήστες συνήθως χρησιμοποιείται η γλώσσα XML. Προβάλλουν τα δεδομένα που έρχονται από τα μοντέλα της αρχιτεκτονικής MVC όταν είναι για αναπαράσταση των δεδομένων ενώ όταν είναι να καταχωρήσουν δεδομένα στην αρχιτεκτονική μας προωθούν τα δεδομένα στα κατάλληλα Controller για περαιτέρω επεξεργασία και τα Controlllers με βάση των κατάλληλων κανόνων που έχουν οριστεί εκτελούν ή όχι την συγκεκριμένη επιθυμητή ενέργεια.

Τέλος, ο Controller εκτελεί τον ρόλο του διαχειριστή μεταξύ του Μοντέλου (Model) και της Οπτικής διεπαφής (View). Οι λειτουργικές ενέργειες στην εφαρμογή μας

εκτελούνται μέσω των Controllers. Οι Controllers καθορίζουν την επικοινωνία με τα μοντέλα, προωθούν τα αιτήματα για τη λήψη των δεδομένων και μετά από την κατάλληλη επεξεργασία τα στέλνουν για απεικόνιση στην οπτική διεπαφή χρηστών (View).

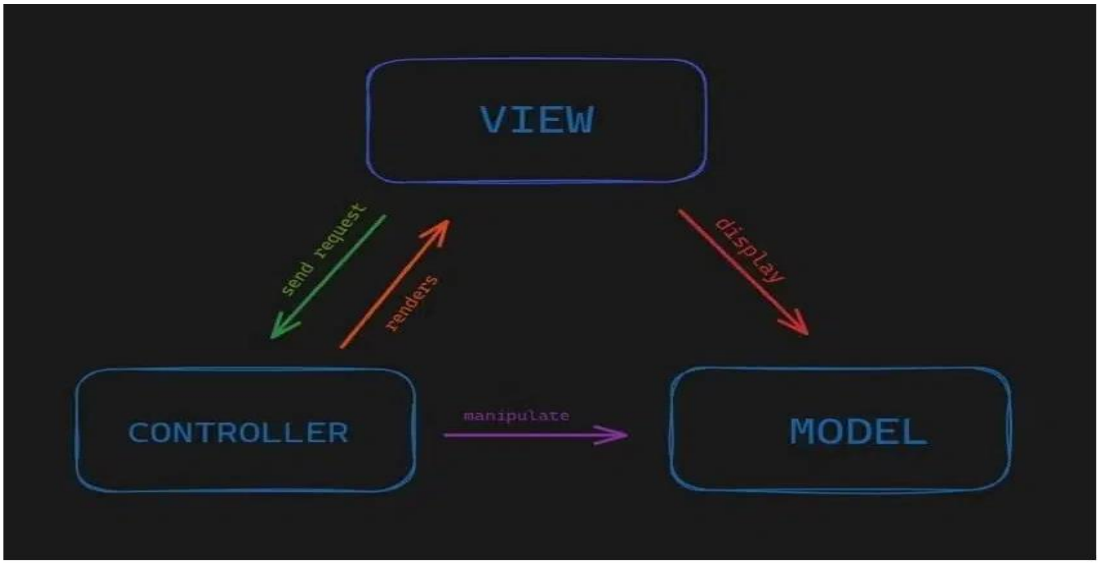


Figure 5.4:103-: Android Operating System Runtime Layer

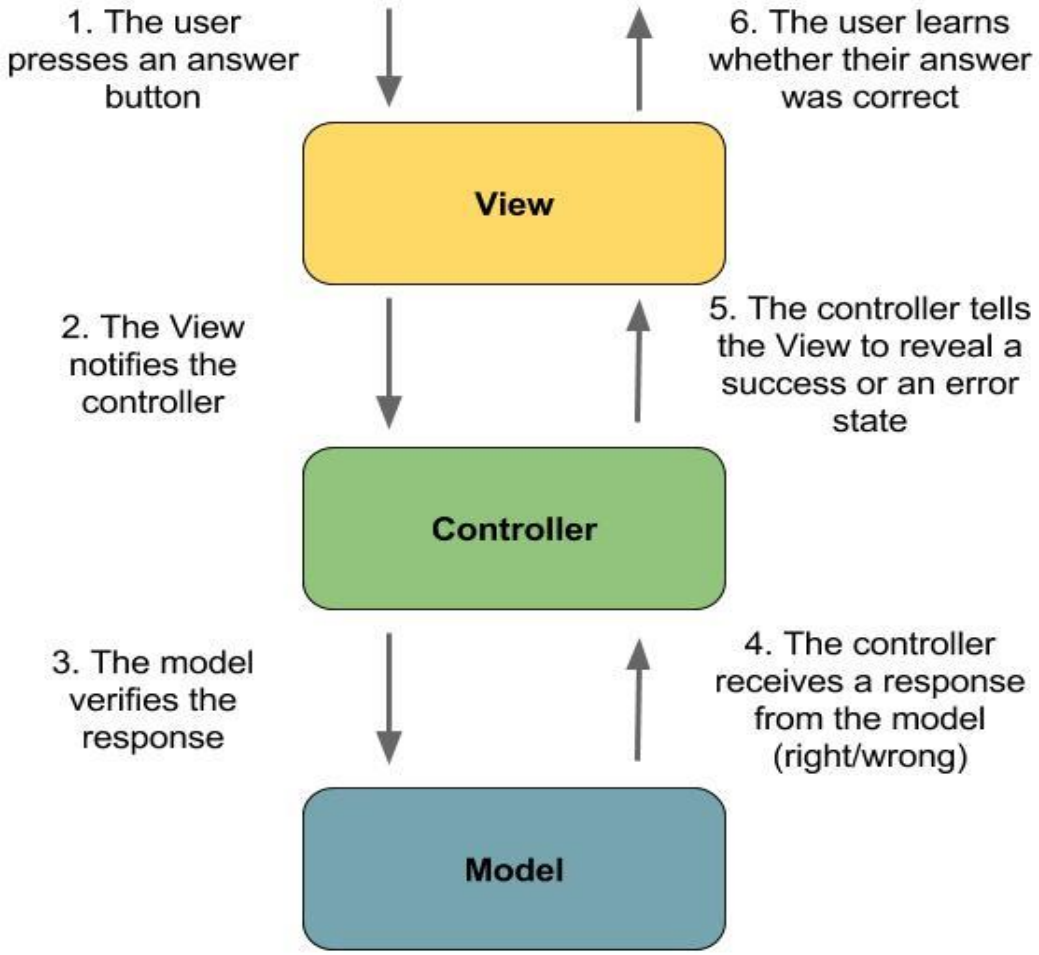


Figure 5.4:104-MVC αρχιτεκτονική και η σειρά εκτέλεσης λειτουργιών.

Ας δούμε με ποια σειρά εκτελούνται τα αντίστοιχα πεδία της αρχιτεκτονικής MVC. Θα μελετήσουμε την περίπτωση ο χρήστης της εφαρμογής να κάνει την ενέργεια να δει το μενού του εστιατορίου.

- Αρχικά ο Χρήστης της εφαρμογής στέλνει ένα αίτημα στη διεπαφή μας μέσω του View για την αναπαράσταση των κατάλληλων δεδομένων και ενεργοποιεί την αντίστοιχη μέθοδο του Controller για τη λήψη των δεδομένων από τη βάση.
- Controller από τη στιγμή που εκτελέσει την κατάλληλη επεξεργασία του αιτήματος (π.χ. αν έχει δικαίωμα να το κάνει αυτό ο συγκεκριμένος χρήστης ή όχι), επικοινωνεί με το κατάλληλο Μοντέλο για την λήψη των δεδομένων (χρήση μεταβλητής).

- Παίρνει τα δεδομένα από το Μοντέλο προσαρμόζοντας αυτά που ζητάει ο Χρήστης.
- Ο Controller προωθεί τα ζητούμενα δεδομένα στα κατάλληλα πεδία γραφικής διεπαφής ώστε ο χρήστης να έχει πρόσβαση στα ζητούμενα δεδομένα.

5.4.2 ΤΟ ΣΧΕΔΙΟ ΣΧΕΔΙΑΣΗΣ (DESIGN PATERN) MVVM

Το MVVM είναι η εξέλιξη του προτύπου ανάπτυξης του MVC. Ο βασικός σκοπός του MVVM είναι να παρέχει έναν σαφή διαχωρισμό μεταξύ του τομέα λογικού και του επιπέδου προβολής. Το MVVM παρέχει αμφίδρομη σύνδεση δεδομένων μεταξύ της προβολής αυτής και του μοντέλου προβολής.

Το μοτίβο MVVM μας επιτρέπει να διαχωρίζουμε την προβολή κώδικα και το μοντέλο μας. Κάτι το οποίο σημαίνει ότι όταν αλλάζει το μοντέλο, η προβολή δεν χρειάζεται. Με το μοντέλο προβολής, μπορούμε να εκτελούμε δοκιμές και να ελέγχουμε τη λογική συμπεριφορά.

- Τα βασικά στοιχεία του MVVM μοντέλου είναι :
- **Το Μοντέλο:**
  Διατηρεί τα δεδομένα
- **Το μοντέλο προβολής:**
  Λειτουργεί ως σύνδεση μεταξύ του μοντέλου και της προβολής
- **Προβολή:**
  Διατηρεί ις διεπαφές του Χρήστη

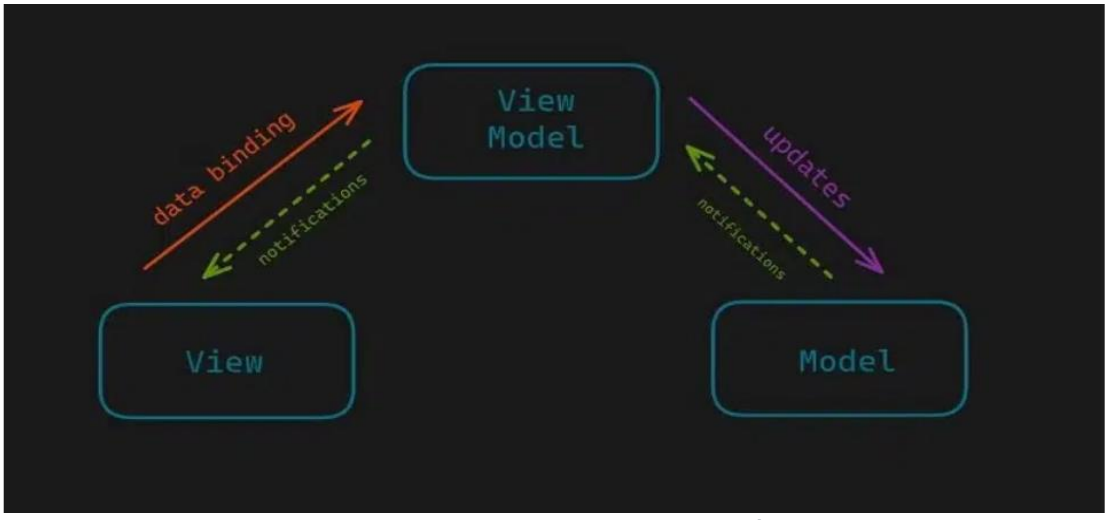


Figure 5.4.2:105-MVVM αρχιτεκτονική

- **Διαχωρισμός προβλημάτων:**
  Το μοντέλο MVC αποτελείται από τρία επίπεδα. Έχει ανατεθεί σε κάθε επίπεδο μια συγκεκριμένη λειτουργία αλλά και το κάθε επίπεδο είναι σε

στενή επαφή με τα υπόλοιπα επίπεδα της αρχιτεκτονικής. Διευκολύνει τους προγραμματιστές για τον εντοπισμό των προβλημάτων.

- **Επεκτασιμότητα:**
  Επίσης ένα από τα βασικά πλεονεκτήματα της αρχιτεκτονικής MVC είναι και η επεκτασιμότητα. Πρέπει κάποιες λειτουργίες που διαθέτει η εφαρμογή μας να έχουν σχεδιαστεί με τέτοιο αφηρημένο τρόπο, δηλαδή όχι να περιοριστούμε σε κάτι συγκεκριμένο, αλλά να το έχουμε σχεδιάσει με πολύ γενικό τρόπο.
- **Ελεγξιμότητα:**
  Η πιο σημαντική ιδιότητα της αρχιτεκτονικής MVC είναι η ελεγξιμότητα. Οι εφαρμογές που δημιουργήθηκαν με βάση τους κανόνες της συγκεκριμένης αρχιτεκτονικής παρέχουν μεγάλη ευκολία στους προγραμματιστές στον εντοπισμό των προβλημάτων και στη συντήρηση τους .

ΚΕΦΑΛΑΙΟ 6: ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Συμπεράσματα και Επεκτάσεις

Στο κεφάλαιο αυτό, πέραν από το επίλογο, θα αναφερθούμε στα συμπεράσματα και στις μελλοντικές αλλαγές που μπορούν να εφαρμοστούν πάνω στην εφαρμογή μας

6.1 Συμπεράσματα

Στην παρούσα διπλωματική εργασία μελετήσαμε τα έξυπνα κινητά τηλέφωνα και τις εφαρμογές. Επικεντρώσαμε στις εφαρμογές που αφορούν τα κτηνιατρεία και προσφέρουν στους χρήστες υπηρεσίες ώστε να έχουνε μια άριστη αντιμετώπιση για τις δύσκολες καταστάσεις που αντιμετωπίζουν με τα κατοικίδια τους.

Στη συνέχεια μελετήσαμε τα ήδη διαθέσιμα εργαλεία που κυκλοφορούν στην αγορά και καταλήξαμε στη χρήση της Firebase (Cloud Base) βάσης δεδομένων για αποθήκευση και για ταυτοποίηση των χρηστών της εφαρμογής μας.

Τέλος αναπτύξαμε την εφαρμογή μας με βάση UserStories και Wireframes. Βασίζοντας σε αυτά τα σχεδιαστικά μοντέλα καταφέραμε να υλοποιήσουμε την εφαρμογή μας σε ολοκληρωμένο περιβάλλον Android Studio.

Ολοκληρώνοντας μελετώντας και τις υπόλοιπες εφαρμογές που κυκλοφορούν στην αγορά προσπαθήσαμε με καινοτόμες ιδέες να καλύψουμε και να προσθέσουμε καινούριες λειτουργίες για την ορθή λειτουργία και άριστη κάλυψη των αναγκών του ενός κτηνιατρείου.

6.2 Επεκτάσεις

Οι προτάσεις για τις μελλοντικές επεκτάσεις της εφαρμογής μας είναι :

- Προσθήκη δυνατότητας εγγραφής με Gmail, Faceboook και άλλα…

- Προσθήκη νέων λειτουργιών όπως ειδοποιήσεις (Notifications) για υπενθύμιση εμβολιασμών.
- Προσθήκη δυνατότητας QR code σε κάθε κατοικίδιο και αναγνώριση μέσω κάμερας το ιστορικό του ζώου.
- Ανάπτυξη των κατάλληλων μεθόδων για σύνδεση με εξωτερικές συσκευές για μέτρηση θερμοκρασίας και παλμών και ταυτόχρονη ενημέρωση του κτηνιάτρου για την πο

ΒΙΒΛΙΟΓΡΑΦΙΑ:

- [1] GOOGLE:
  - www.google.com
- [2] ANDROID:
  - Developer.android.com
- [3] https://firebase.google.com/
- [4] Vet Calculator - Apps on Google Play
- [5] https://www.android.com/android-11/
- [6] VitusVet: Pet Health Care App - Apps on Google Play
- [7] History of mobile phones (uswitch.com)
- [8] https://developer.android.com/studio