ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΕΦΑΡΜΟΓΗ ΚΙΝΗΤΟΥ ΓΙΑ ΔΙΑΧΕΙΡΙΣΗ ΚΤΗΝΙΑΤΡΕΙΟΥ

VETERINARY PRACTICE MANAGEMENT MOBILE APPLICATION

ΟΝΟΜΑ:ΑΔΑΜΟΠΟΥΛΟΣ ΠΑΝΑΓΙΩΤΗΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:ΑΓΓΕΛΙΔΗΣ ΠΑΝΤΕΛΗΣ

ΚΟΖΑΝΗ,ΟΚΤΩΜΒΡΙΟΣ 2021

ΠΕΡΙΛΗΨΗ

Η εξέλιξη των Τεχνολογιών Πληροφορικής και Επικοινωνιών έχει φέρει τα έξυπνα κινητά στη ζωή των ανθρώπων. Καθημερινά τα χρησιμοποιούμε, όλο και περισσότερο, για αναζητήσεις πληροφοριών, ανταλλαγή μηνυμάτων, εικόνων και δεδομένων με άλλους χρήστες. Για τη βελτίωση της εμπειρίας των χρηστών, έχουν υλοποιηθεί συγκεκριμένες εφαρμογές σε αυτά μέσω των οποίων παρέχονται εξειδικευμένες λειτουργίες στους χρήστες τους.

Οι εφαρμογές των έξυπνων κινητών καλύπτουν ένα ευρύ φάσμα από εφαρμογές για καιρό, αθλήματα, ενημέρωση, παιχνίδια κ.λπ. Τον τελευταίο καιρό, έχουν κάνει την εμφάνισή τους και ιατρικές εφαρμογές που είτε δίνουν πληροφορίες είτε μπορούν να χρησιμοποιηθούν για καταχώριση στοιχείων και παρακολούθηση των ασθενών. Στο πλαίσιο αυτό των εφαρμογών, έχουν παρουσιαστεί και εφαρμογές για κτηνιατρεία.

Λέξεις-κλειδιά: Έξυπνακινητά, Android, Firebase, AndroidStudio, Κτηνιατρείο

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΕΙΣΑΓΩΓΗ

Στο κεφάλαιο 1 θα αναφερθούμε στα έξυπνα κινητά τηλέφωνα και τις εφαρμογές που έχουν αναπτυχθεί γι' αυτά. Θα ξεκινήσουμε με μια σύντομη ιστορική αναδρομή των έξυπνων κινητών και μια παρουσίαση των λειτουργικών συστημάτων που έχουν χρησιμοποιηθεί. Στη συνέχεια θα μελετήσουμε τις εφαρμογές έξυπνων κινητών που αφορούν όμως διαχείριση κτηνιατρείου.

Στο κεφάλαιο 2 θα παρουσιάσουμε τις τεχνολογίες και τα εργαλεία που χρησιμοποιήσαμε για την υλοποίηση της εφαρμογής για τα έξυπνα κινητά. Θα ξεκινήσουμε με τη βάση δεδομένων Firebaseπου χρησιμοποιήσαμε στην εφαρμογή μας. Το κεφάλαιο θα ολοκληρωθεί με μια αναφορά στο AndroidStudioτο εργαλείο που θα μας υποστηρίξει για την υλοποίηση της εφαρμογής.

Στο κεφάλαιο 3 θα δούμε τα βήματα που ακολουθήσαμε για την υλοποίηση της εφαρμογής για τα έξυπνα κινητά τηλέφωνα. Θα ξεκινήσουμε με το σχεδιασμό της και συγκεκριμένα θα αναφερθούμε στα UserStories και στα Wireframes. Κατόπιν θα δούμε, περιληπτικά, τα στάδια υλοποίησης και τα απαραίτητα βήματα για την εγκατάσταση. Τέλος θα υπάρχουν και ενδεικτικές οθόνες λειτουργίας της εφαρμογής.

Στο κεφάλαιο 4 θα παρουσιάσουμε τα συμπεράσματα που αποκομίσαμε μετά από την ανάπτυξη της εφαρμογής και θα αναφέρουμε προτάσεις για μελλοντικές επεκτάσεις της.

ΚΕΦΑΛΑΙΟ 1. ΕΞΥΠΝΑ ΚΙΝΗΤΑ ΤΗΛΕΦΩΝΑ ΚΑΙ ΕΦΑΡΜΟΓΕΣ

1.1 Εισαγωγή

Στο κεφάλαιο αυτό θα παρουσιάσουμε τα έξυπνα κινητά και τις εφαρμογές τους. Θα ξεκινήσουμε με την ιστορική αναδρομή των έξυπνων κινητών, αρχίζοντας με τα πρώτα βήματα που έγιναν στις τηλεπικοινωνίες και στη συνέχεια οδήγησαν στην εξέλιξη που έχουμε σήμερα. Στη συνέχεια θα δούμε τα λειτουργικά συστήματα που υποστηρίζουν τη λειτουργία των έξυπνων κινητών για να παρέχουν τις υπηρεσίες αυτές στους χρήστες. Τέλος θα αναφερθούμε στις εφαρμογές των έξυπνων κινητών για τα κτηνιατρεία.

1.2 Έξυπνα κινητά

1.2.1 Ιστορική Αναδρομή

Η ιστορία των έξυπνων κινητών αρχίζει περίπου στη δεκαετία του 1920 όπου έγιναν μια σειρά από πειράματα επικοινωνίας από και σε κινούμενα οχήματα. Έτσι το 1926 έγινε η πρώτη προσπάθεια όπου προσφέρθηκε η πρώτη επιτυχημένη υπηρεσία κινητής τηλεφωνίας σε επιβάτες πρώτης κατηγορίας στο Deutsche Reichsbahn στη διαδρομή μεταξύ Βερολίνου και Αμβούργου.
Αργότερα, στην άλλη πλευρά του Ατλαντικού, το 1946 έγιναν οι πρώτες κλήσεις με ραδιοτηλέφωνο αυτοκινήτου, στο Σικάγο. Δυστυχώς όμως οι διαθέσιμες ραδιοσυχνότητες ήταν πολύ λίγες και η υπηρεσία έφτασε πολύ γρήγορα στα όρια της χωρητικότητας που μπορούσε να υποστηρίξει. Το 1956, στη Σουηδία, κυκλοφόρησε το πρώτο αυτοματοποιημένο σύστημα κινητών τηλεφώνων για ιδιωτικά οχήματα. Βέβαια η συσκευή αυτή ζύγιζε 40 κιλά και απαιτούσε ειδική εγκατάσταση στο αυτοκίνητο.

Εικόνα 1 Το πρώτο σύστημα κινητών τηλεφώνων για αυτοκίνητα (USwitch, 2021)

Αρκετά χρόνια αργότερα, το 1969 ξεκίνησαν οι προσπάθειες από τις Σκανδιναβικές χώρες, οι μηχανικοί των οποίων ίδρυσαν το Nordic Mobile Telephone (NMT) Group σε μια προσπάθεια να αναπτυχθεί ένα δικό τους σύστημα επικοινωνία και όχι να εισαχθεί κάποιο έτοιμο από την Αμερική. Το 1973, ο γενικός διευθυντής της Motorola communications, ο Dr Martin Cooper, πραγματοποίησε την πρώτη δημόσια τηλεφωνική κλήση σε μια συσκευή που ζύγιζε 1,1 κιλά.

Το 1982, μηχανικοί και διαχειριστές από έντεκα ευρωπαϊκές χώρες συγκεντρώθηκαν στη Στοκχόλμη για να εξετάσουν εάν ένα ευρωπαϊκό ψηφιακό κινητό τηλέφωνο ήταν πολιτικά και τεχνικά. Η ομάδα υιοθέτησε το σκανδιναβικό μοντέλο συνεργασίας και έθεσε τα θεμέλια ενός διεθνούς προτύπου(USwitch, 2021).

Το πρώτο κινητό τηλέφωνο παρουσιάστηκε από τη Motorolato Μάρτιο του 1984 και ήταν το Motorola DynaTAC 8000x στην τιμή πώλησης των 3.995 δολαρίων. Βέβαια το 8000x δεν μπορούσε να χαρακτηριστεί κινητό - ζύγιζε σχεδόν δύο κιλά και χρειάστηκε δέκα ώρες για να φορτιστεί για 30 λεπτά χρόνου ομιλίας.

Το 1987 εγκρίθηκαν οι τεχνικές προδιαγραφές για το πρότυπο GSM εγκρίνονται. Επικεντρώθηκε στη διαλειτουργικότητα πέρα από τα εθνικά σύνορα με αποτέλεσμα να ασχολείται με διαφορετικές ζώνες συχνοτήτων, ποιότητα κλήσεων και χαμηλό κόστος.

Στις 25 Απριλίου 1989, η Motorolaεπανέρχεται με το νεότερο μοντέλο της το Motorola MicroTAC 9800x. Το μοντέλο αυτόπλησίαζε να είναι πιο πολύ κινητό με το (σχετικά) μικρό μέγεθος και το αναδιπλούμενο επιστόμιο. Φυσικά, και αυτό, όπως και ο προκάτοχός του το 8000xείχαν κεραίες και μπορούσαν να χρησιμοποιηθούν μόνο για την πραγματοποίηση κλήσεων.

Εικόνα2 Motorola MicroTAC 9800x(USwitch, 2021)

Το 1992 στάλθηκε το πρώτο μήνυμα SMS στον κόσμο στο Ηνωμένο Βασίλειο. Ο Neil Papworth, προγραμματιστής, 22 ετών, στα πλαίσια του έργου που είχε αναλάβει να αναπτύξει μιας υπηρεσία ανταλλαγής μηνυμάτων για τη Vodafone, έστειλε το μήνυμα κειμένου που έγραφε «Καλά Χριστούγεννα». Το μήνυμα και στάλθηκε στον Richard Jarvis, διευθυντή της Vodafone (USwitch, 2021).

Το πρώτο smartphone, δημιουργήθηκε από την IBM το 1992 και κυκλοφόρησε στην αγορά το 1994. Ονομάστηκε Simon Personal Communicator (SPC). Παρότι δεν ήταν πολύ συμπαγές και κομψό, είχε αρκετά στοιχεία τα οποία τα συναντάμε και σε έξυπνα κινητά που κυκλοφορούν σήμερα.Για παράδειγμα, το SPC ήταν εξοπλισμένο με οθόνη αφής, καθώς και τη δυνατότητα αποστολής και λήψης email και fax. Είχε ημερολόγιο, βιβλίο διευθύνσεων και εγγενή προγραμματιστή ραντεβού. Επίσης διέθετε τυπικά και προβλέψιμα πληκτρολόγια οθόνης με είσοδο γραφίδας. Χάρις τις λειτουργίες αυτές, οι οποίες ήταν διαφορετικές και αρκετά εξελιγμένες θεωρείται «Το πρώτο smartphone του κόσμου». Για λειτουργικό σύστημα το SPC χρησιμοποίησε το ROM-DOS, μια τροποποιημένη έκδοση του MS-DOS που έχει σχεδιαστεί ειδικά για

ενσωματωμένα συστήματα. Οι πωλήσεις του SPCήταν 50.000 και παρέμεινε στην αγορά για μόλις έξι μήνες(Tocci, n.d.).

Εικόνα 3SPC (Tocci, n.d.)

Το 1996, η Nokia παρουσίασε το Nokia 9000 Communicator. Το λειτουργικό του σύστημα ήταν το GEOS 3.0 και είχε κάποιες πρωτοποριακές εφαρμογές, όπως για παράδειγμα ένα γραφικό πρόγραμμα περιήγησης στο Web. Ο σχεδιασμός clamshell που θα κυριαρχούσε στην αγορά για τα επόμενα χρόνια έκρυβε ένα πλήρες πληκτρολόγιο QWERTY.

Εικόνα 4Nokia 9000 Communicator (Pothitos, 2016)

Πολλές προσπάθειες έγιναν και από την Ericsson με αρκετά μοντέλα που είτε απορρίφθηκαν πριν παρουσιαστούν στην αγορά είτε εγκρίθηκαν αλλά τελικά δεν παρουσιάστηκαν στην αγορά όπως το GS88. Όμως το Ericsson R380 κυκλοφόρησε στα τέλη του 1999 και ήταν η πρώτη φορητή συσκευή που χρησιμοποίησε το Symbian OS. Αυτό το λειτουργικό σύστημα θα συνεχίσει να κυριαρχεί στην αγορά μέχρι το τελευταίο τρίμηνο του 2010.

Εικόνα 5EricsonR380 (Pothitos, 2016)

Υπήρχαν και άλλες προσπάθειες όπως για παράδειγμα το Qualcomm PDQ 800, το οποίο δυστυχώς είχε πολλά προβλήματα και δεν κατάφερε να διατηρηθεί στην αγορά. Επίσης η Microsoftασχολήθηκε με τα έξυπνα κινητά, με λάθος όμως όραμα, να προσπαθεί να μεταφέρει τα Windowsσε αυτά (Pothitos, 2016).

Σιγά σιγά, άρχισε να δημιουργείται ένα πρότυπο κινητής επικοινωνίας για να επιτρέπει ασύρματη πρόσβαση σε φορητές ηλεκτρονικές συσκευές. Το 2000 οι έξυπνες συσκευές ήταν συνδεδεμένες σε ένα δίκτυο 3G. Αυτό βοήθησε στην ευρύτερη αποδοχή των έξυπνων συσκευών γιατί με τη βοήθειά του μπορούσε να πραγματοποιηθεί τηλεδιάσκεψη, να γίνει αποστολή μεγάλων συνημμένων email κ.λπ. Βέβαια τα δεδομένα και η πρόσβαση στο διαδίκτυο είχαν ένα αρκετά μεγάλο κόστος (Tocci, n.d.).

Στις Η.Π.Α τα κινητά τηλέφωνα χρησιμοποιήθηκαν κυρίως στην εργασία και γι' αυτό το λόγο είχαν μεγάλα φυσικά πληκτρολόγια και είσοδο γραφίδας. Έτσι το 2002 παρουσιάστηκε η πρώτη συσκευή BlackBerry, το 5810. Όταν το δει κάποιος, σίγουρα θα καταλάβει ότι διαθέτει υποστήριξη e-mail και ανταλλαγή μηνυμάτων κειμένου χάρις στο πληκτρολόγιο που διέθετε. Επίσης παρείχε πρόγραμμα περιήγησης WAP. Δυστυχώς όμως δε διέθετε ενσωματωμένο μικρόφωνο ή ηχείο, έπρεπε να συνδεθούν ακουστικά για να πραγματοποιηθούν κλήσεις.

Εικόνα 6BlackBerry 5810 (German, 2013)

Η μεγάλη ανάπτυξη των έξυπνων κινητών ήρθε το 2007 όταν ο ο Steve Jobs και η ομάδα του Macworld αποκάλυψαν το πρώτο iPhone. Η συγκεκριμένη συσκευή διέθετε την πιο κομψή συσκευή οθόνης αφής, αλλά επίσης ήταν η πρώτη συσκευή που προσέφερε μια πλήρη, πρόσβαση στο διαδίκτυο, σαν να ήταν στον υπολογιστή τους. Οι προηγούμενες εκδόσεις χρησιμοποιούσαν φυλλομετρητές ειδικά κατασκευασμένους για τη συσκευή τους και με περιορισμένες δυνατότητες. Διέθετε πολλά χαρακτηριστικά, όπως μνήμη 4Gή 8G, μπαταρία που διαρκούσε για 8 ώρες ομιλία και ένα πιο φιλικό προς το χρήστη πληκτρολόγιο σε σχέση με τον ανταγωνισμό (Tocci, n.d.).

Εικόνα 7i-phone (Pothitos, 2016)

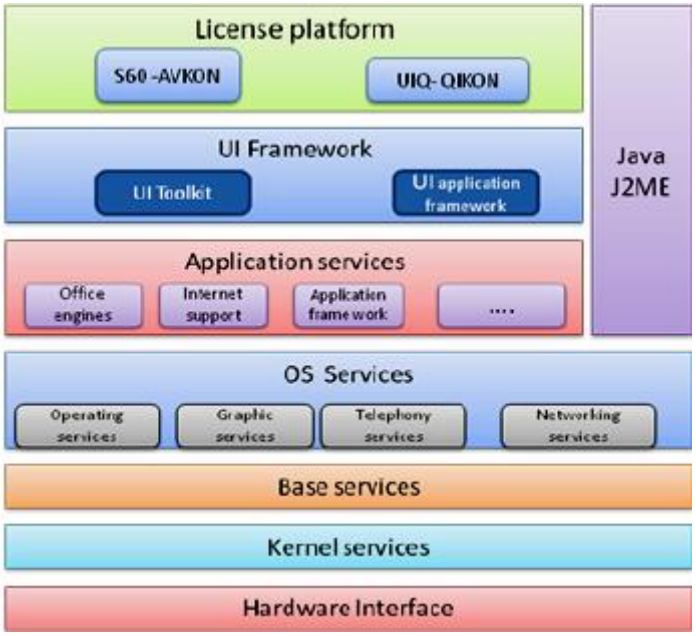
Παράλληλα αναπτυσσόταν ένα νέο λειτουργικό σύστημα το οποίο θα ήταν το αντίπαλο δέος για τα I-Phone. Ο Andy Rubin δημιούργησε τη δική του έκδοση ενός φορητού λειτουργικού συστήματος που ονομάζεται Android. Το πρώτο τηλέφωνο που βασίστηκε στο Android κυκλοφόρησε το 2008 και ήταν από την HTC, το HTC Dream ή το G1 όπως ήταν γνωστό στις ΗΠΑ.

Εικόνα 8HTCDream (Pothitos, 2016)

Το 2010 το λειτουργικό σύστημα Symbianπαρέδωσε την ηγετική θέση στην αγορά κινητών στο Android. Από τότε ξεκίνησε μια κούρσα μεταξύ κυρίως των λειτουργικών συστημάτων για τα έξυπνα κινητά και των εταιρειών παρουσιάζοντας νέα μοντέλα και δυνατότητες στο ευρύ κοινό (Pothitos, 2016). Στην ενότητα που ακολουθεί θα δούμε αναλυτικά τα λειτουργικά συστήματα για τα έξυπνα κινητά.

1.2.2 Λειτουργικά Συστήματα

Στην ενότητα αυτή θα παρουσιάσουμε, συνοπτικά, τα Λειτουργικά Συστήματα των έξυπνων κινητών. Το πρώτο λειτουργικό σύστημα ήταν το Το Symbian OS είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα, γραμμένο σε γλώσσα προγραμματισμού C ++ που αναπτύχθηκε από τη Symbian Ltd. το 1977. Το Symbian OS αποτελείται από πολλά επίπεδα, όπως βιβλιοθήκες λειτουργιών, μηχανές εφαρμογών, MKV, διακομιστές, το βασικό πυρήνα και το επίπεδο διεπαφής υλικού. Το Symbian ήταν το πιο διαδεδομένο λειτουργικό σύστημα κινητής συσκευής μέχρι το 2010, όπου και αντικαταστάθηκε από το Android.

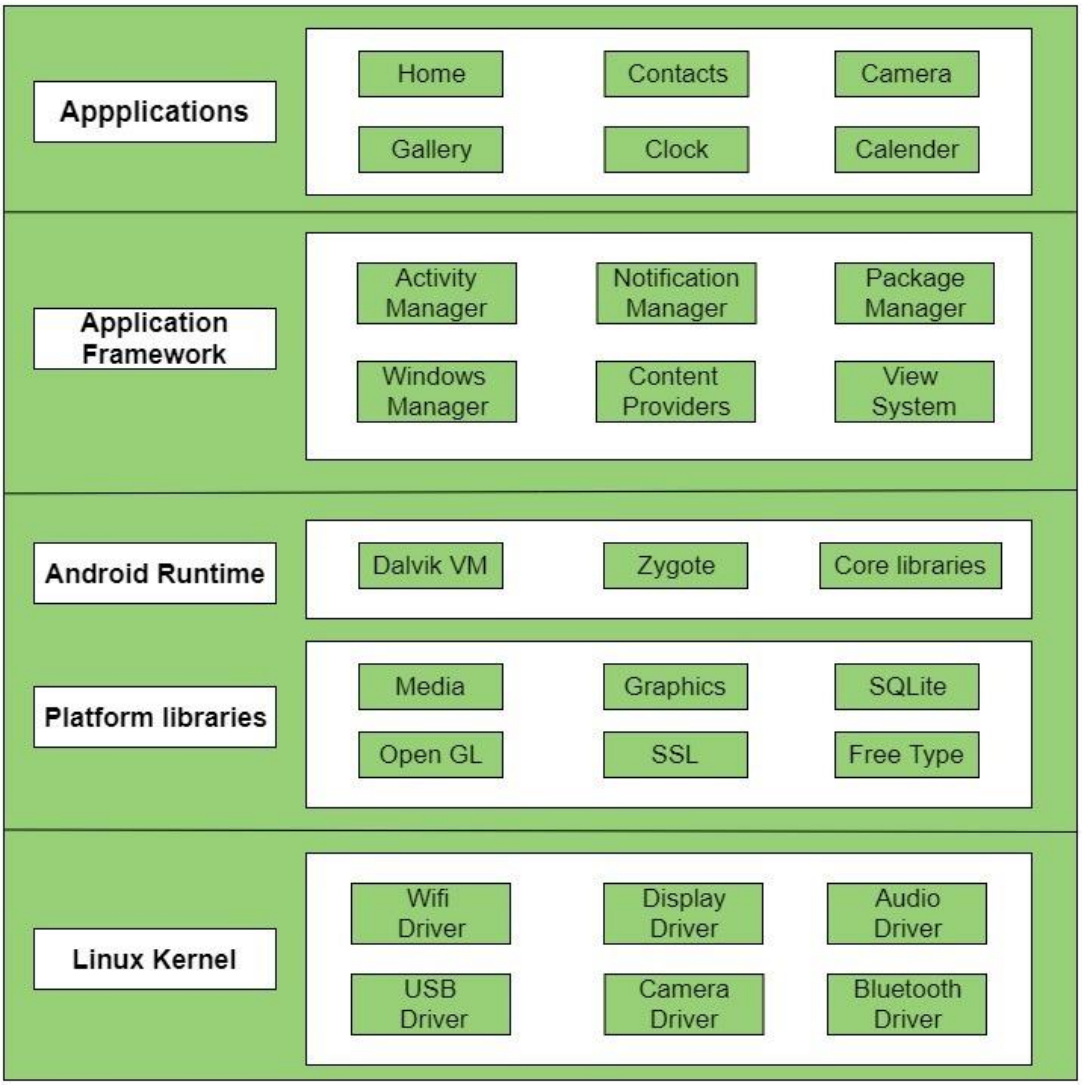


Εικόνα 9Αρχιτεκτονική SymbianOS (Sen, 2015)

Το Android είναι ένα λειτουργικό λογισμικό ανοιχτού κώδικα που βασίζεται σε Linux και χρησιμοποιεί και παρέχει ένα σύνολο υπηρεσιών όπως όπως ασφάλεια, διαχείριση μνήμης, διαχείριση διεργασιών, στοίβα δικτύου και ένα μοντέλο

προγράμματος οδήγησης χρησιμοποιώντας τις λειτουργίες του λειτουργικού συστήματος Linux. Προσφέρει ένα ευρύ φάσμα βιβλιοθηκών που επιτρέπουν στους προγραμματιστές εφαρμογών να δημιουργούν διαφορετικές εφαρμογές. Οι εφαρμογές Android γράφονται συνήθως σε γλώσσα προγραμματισμού Java (Hamed, Dara&Kremer, 2017).

Η αρχιτεκτονική Android περιέχει διαφορετικό αριθμό στοιχείων για την υποστήριξη οποιωνδήποτε αναγκών συσκευής Android. Το λογισμικό Android περιέχει έναν πυρήνα Linux ανοιχτού κώδικα με ένα μεγάλο αριθμό βιβλιοθηκών C / C ++ οι οποίες είναι διαθέσιμες μέσω υπηρεσιών πλαισίου εφαρμογών.



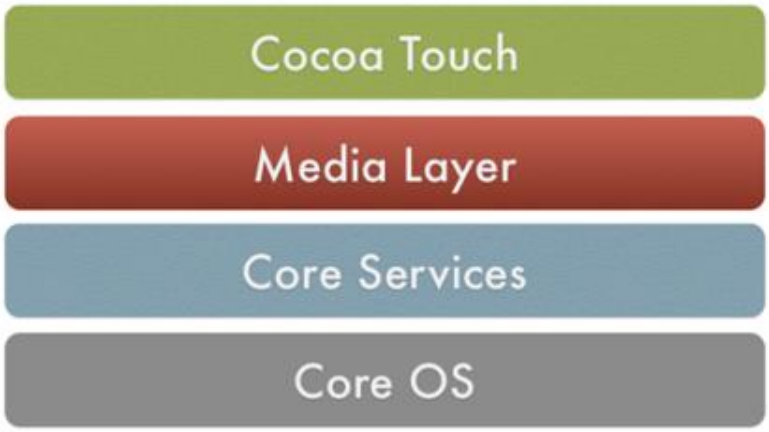
Εικόνα 10Αρχιτεκτονική Android (Praveenruhil, 2021)

Το Linux Kernel παρέχει κύρια λειτουργικότητα των διαθέσιμων λειτουργιών του λειτουργικού συστήματος σε έξυπνα κινητά και η Dalvik Virtual Machine (DVM) παρέχει την πλατφόρμα για την εκτέλεση μιας εφαρμογής Android. Ας δούμε όμως λίγο πιο αναλυτικά τα κύρια στοιχεία της αρχιτεκτονικής του Android:

- Εφαρμογές: οι εφαρμογές είναι το ανώτερο επίπεδο αρχιτεκτονικής Android. Όλες οι εφαρμογές που κατεβάζουμε θα εγκατασταθούν στο επίπεδο αυτό. Εκτελείται με τη βοήθεια των κλάσεων και των υπηρεσιών που παρέχονται από το κατώτερο επίπεδο το πλαίσιο εφαρμογής.
- Πλαίσιο εφαρμογής: παρέχει πολλές σημαντικές κλάσεις και υπηρεσίες που χρησιμοποιούνται για τη δημιουργία μιας εφαρμογής Android. Παρέχει μια γενική αφαίρεση για την πρόσβαση στο υλικό και βοηθά επίσης στη διαχείριση των πόρων που χρησιμοποιούνται από το περιβάλλον εργασίας χρήστη.
- Android runtime: είναι ένα από τα πιο σημαντικά μέρη του Android. Περιέχει στοιχεία όπως βασικές βιβλιοθήκες και την εικονική μηχανή Dalvik (DVM). Κυρίως, παρέχει τη βάση για το ανώτερο επίπεδο - πλαίσιο εφαρμογών και εξουσιοδοτεί την εφαρμογή μας με τη βοήθεια των βασικών βιβλιοθηκών.
- Βιβλιοθήκες πλατφόρμας: οι Βιβλιοθήκες πλατφόρμας περιλαμβάνουν διάφορες βιβλιοθήκες πυρήνα C / C ++ και βιβλιοθήκες βασισμένες σε Java, όπως Media, Graphics, Surface Manager, OpenGL κ.λπ. για να παρέχουν υποστήριξη για τα ανώτερα επίπεδα και τις εφαρμογές.
- Πυρήνας Linux: είναι η καρδιά της αρχιτεκτονικής του Android. Διαχειρίζεται όλα τα διαθέσιμα προγράμματα οδήγησης, όπως προγράμματα οδήγησης οθόνης, προγράμματα οδήγησης κάμερας, προγράμματα οδήγησης Bluetooth, προγράμματα οδήγησης ήχου, προγράμματα οδήγησης μνήμης κ.λπ. που απαιτούνται κατά τη διάρκεια του χρόνου εκτέλεσης. Ο πυρήνας Linux παρέχει ένα επίπεδο αφαίρεσης μεταξύ του υλικού της συσκευής και των άλλων στοιχείων της αρχιτεκτονικής του Android. Είναι υπεύθυνος για τη διαχείριση όλων των πόρων του υλικού όπως της μνήμης, της ισχύος, των συσκευών κ.λπ. (Praveenruhil, 2021)

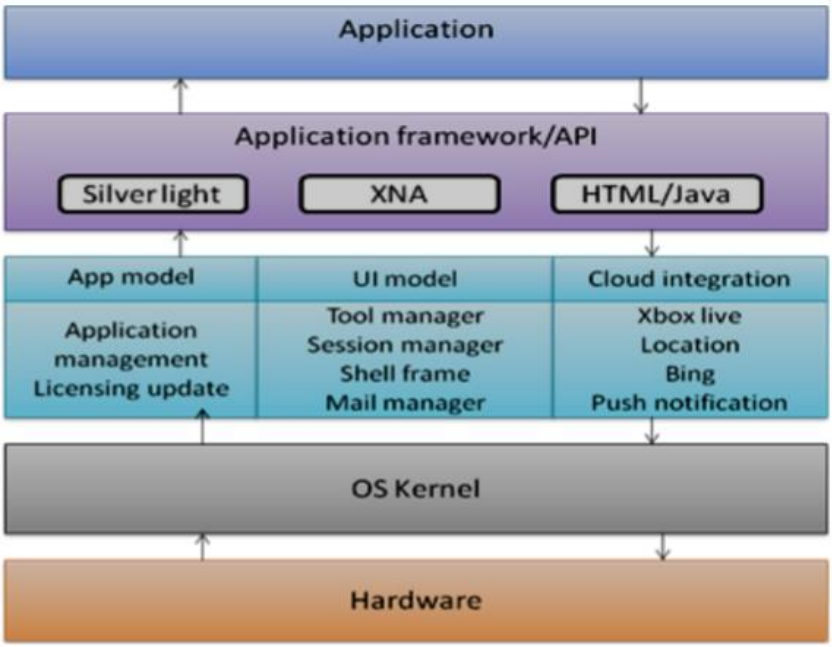
Το Apple iOS είναι ένα λειτουργικό σύστημα κλειστού κώδικα που αναπτύχθηκε από την Apple το 2007. Χρησιμοποιείται από προϊόντα μόνο για Apple (iPhone, iPod και iPad). Η αρχιτεκτονική του iOS βασίζεται σε πολυεπίπεδη αρχιτεκτονική όπου στο

ανώτερο επίπεδο, το iOS λειτουργεί ως ενδιάμεσος μεταξύ του υποκείμενου υλικού και των εφαρμογών. Οι εφαρμογές μιλούν με το υλικό μέσω μιας συλλογής καλά καθορισμένων διεπαφών συστήματος. Αυτές οι διεπαφές καθιστούν απλή τη σύνταξη εφαρμογών που λειτουργούν συνεχώς σε συσκευές με διάφορες δυνατότητες υλικού.
Τα κατώτερα επίπεδα παρέχουν τις βασικές υπηρεσίες στις οποίες βασίζεται όλη η εφαρμογή και το ανώτερο επίπεδο παρέχει εξελιγμένες υπηρεσίες γραφικών και διεπαφών. Η Apple παρέχει τις περισσότερες διεπαφές συστήματος σε ειδικά πακέτα που ονομάζονται πλαίσια(framework) (Naveen, 2021).



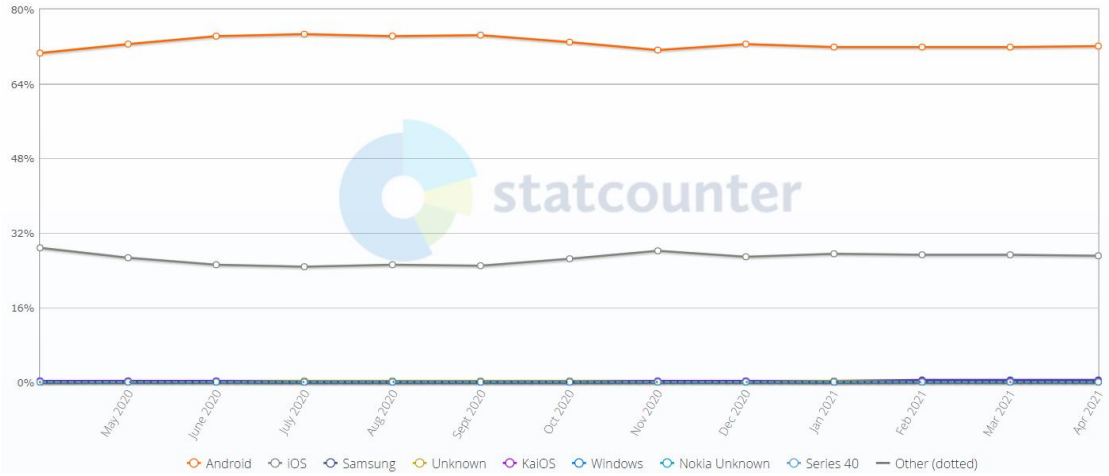
Εικόνα 11Αρχιτεκτονική iOS (Naveen, 2021)

Τέλος το λειτουργικό σύστημα Windows Phone είναι ένα κλειστό λειτουργικό σύστημα για έξυπνα κινητά που αναπτύχθηκε από τη Microsoft Corporation και χρησιμοποιείται από πολλές έξυπνες συσκευές (προσωπικοί ψηφιακοί βοηθοί, smartphone και συσκευές αφής). Το λειτουργικό σύστημα Windows Phone βασίζεται σε μια ειδική έκδοση του .Net framework.Και αυτό το λειτουργικό σύστημα στηρίζει την αρχιτεκτονική του σε επίπεδα, επιτυγχάνοντας την ανεξαρτησία της φυσικής συσκευής από τις εφαρμογές, την ευκολότερη ανάπτυξη εφαρμογών, την προσθήκη – ενημέρωση βιβλιοθηκών και λειτουργιών χωρίς προβλήματα κ.λπ. (Hamed, Dara&Kremer, 2017).



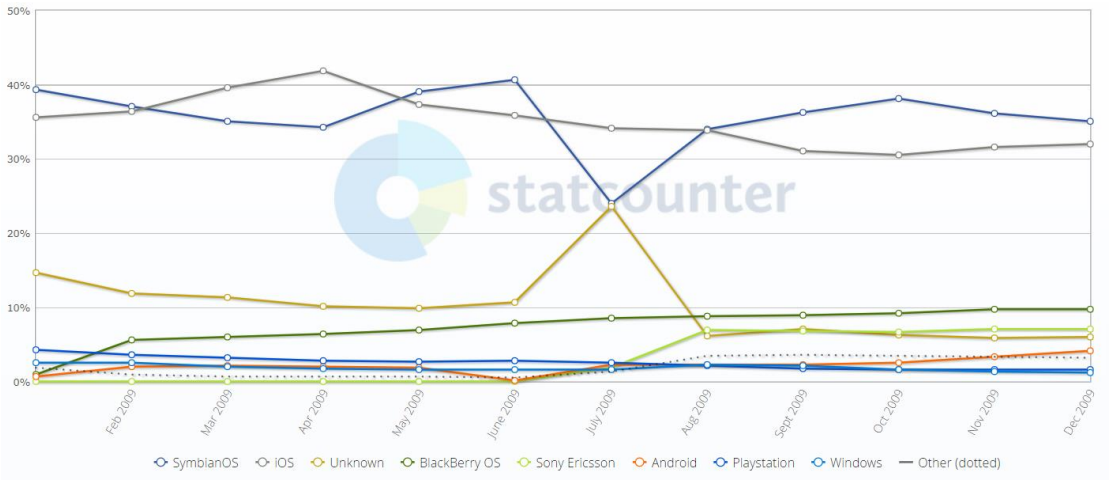
Εικόνα 12 Αρχιτεκτονική WindowsOS (Sen, 2015)

Σύμφωνα με το StatCounter (2021) το λειτουργικό σύστημα Androidκυριαρχεί στην αγορά, όπως μπορούμε να δούμε στην εικόνα που ακολουθεί:



Εικόνα 13 Μερίδιο αγοράς Λειτουργικών συστημάτων το 2021(statcounter, 2021)

Στην εικόνα που ακολουθεί μπορούμε να δούμε την αγορά το 2009, λίγο μετά την κυκλοφορία του Android, όπου παρατηρούμε την πτώση του λειτουργικού συστήματος Symbiankαι την αργή – σταδιακή αύξηση του Android.



Εικόνα 14 Μερίδιο αγοράς Λειτουργικών συστημάτων το 2009 (statcounter, 2021)

1.3 Εφαρμογές διαχείρισης κτηνιατρείου

Για να παραμείνουν στην κορυφή του επαγγέλματός τους, οι κτηνίατροι πρέπει να συμβαδίζουν με τα συνεχώς εξελισσόμενα ιατρικά πρότυπα. Αυτό σημαίνει ότι θα πρέπει να υιοθετούν καινοτόμες τεχνικές, να κατανοούν τα νέα φάρμακα και τις παρενέργειες που μπορεί να έχουν και να είναι πάντα ενήμεροι για τις νέες εξελίξεις στον κλάδο τους.Αυτό μπορεί να γίνει εύκολα σήμερα με τη βοήθεια των Τεχνολογιών Πληροφορικής και Επικοινωνιών και της εξέλιξης των έξυπνων κινητών. Πλέον είναι διαθέσιμές κτηνιατρικές εφαρμογές που βοηθούν τους κτηνιάτρους και τους ιδιοκτήτες κατοικίδιων να αποκτήσουν γρήγορη πρόσβαση στις γνώσεις των ειδικών και στις τελευταίες τάσεις.

Με τη βοήθεια των συγκεκριμένων εφαρμογών μπορούμε να παρακολουθούμε την εξέλιξη της υγείας των ζώων μας, να αντλούμε συμβουλές για την υγεία τους, να ενημερωνόμαστε για τα τελευταία δεδομένα και τον τρόπο αντιμετώπισης προβλημάτων. Ενδεικτικά οι εφαρμογές αυτές παρέχουν τις ακόλουθες λειτουργίες:

- Υπολογισμός δοσολογίας
- Διάγνωση ασθενειών
- Παροχή ενέσεων
- Εκπαίδευση πελατών σχετικά με τις πρώτες βοήθειες για κατοικίδια

Στον πίνακα που ακολουθεί μπορούμε να δούμε ορισμένες εφαρμογές για κτηνιατρείο, το λειτουργικό σύστημα έξυπνων κινητών για το οποίο είναι διαθέσιμες και το αντίστοιχο κόστος αγοράς (αν υπάρχει).

Πίνακας 1 Εφαρμογές Κτηνιατρείου

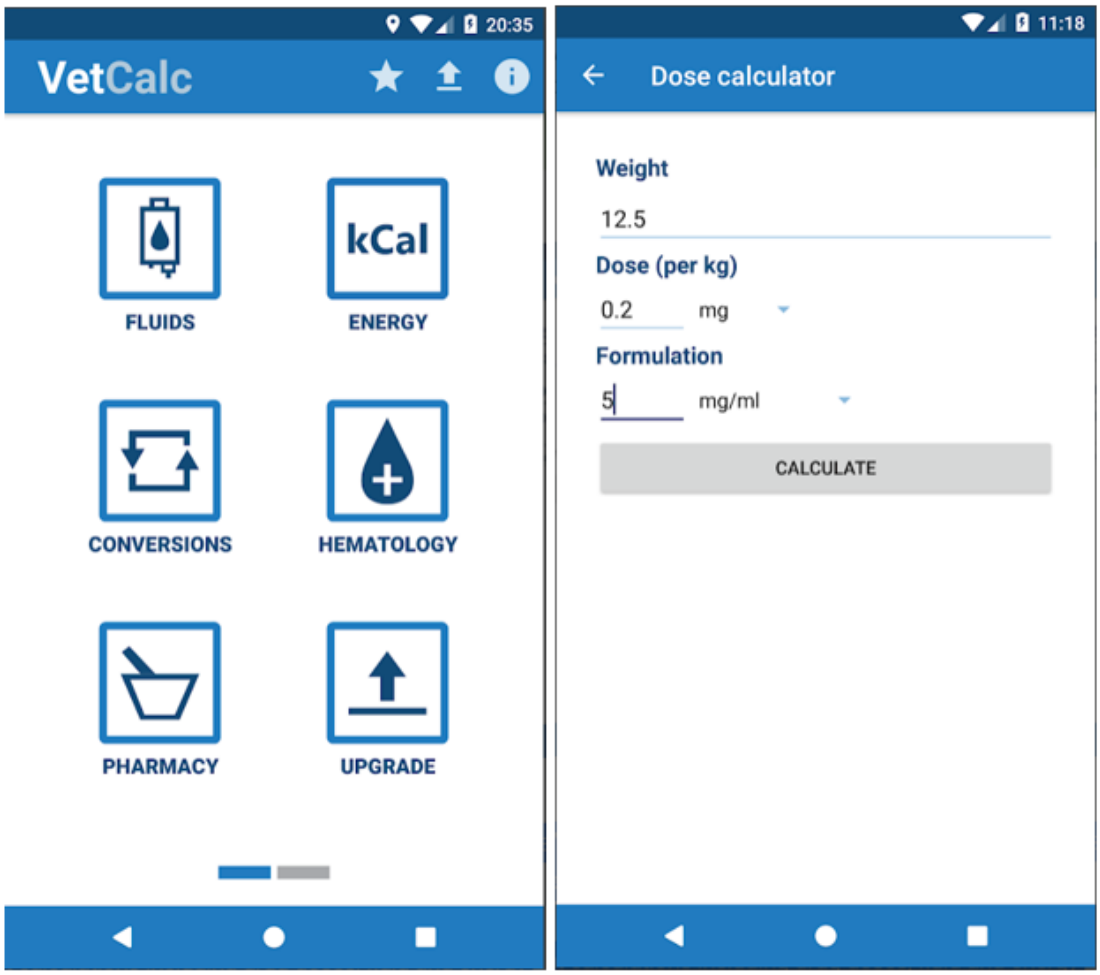
Εφαρμογή Κτηνιατρείου	Android		iOS		Τιμή
	Βαθμολογία	Ενημέρωση	Βαθμολογία	Ενημέρωση	
MSD Vet Manual	4.7	Jun 26, 2020	–	–	Δωρεάν
VetPDA Calcs	–	–	4.8	Apr 26, 2018	$4.99
Compendium of Veterinary Products	3.2	Jun 12, 2015	4.2	Sep 22, 2017	Δωρεάν
Vet Blood Tests Guide	n/a	Aug 21, 2012	–	–	$10.00
Animal & Veterinary Drugs	–	–	2.9	Nov 14, 2017	$0.99

Εφαρμογή Κτηνιατρείου	Android		iOS		Τιμή
	Βαθμολογία	Ενημέρωση	Βαθμολογία	Ενημέρωση	
Timeless Vet Drug Index	3.6	Feb 2, 2015	3	Oct 10, 2014	Δωρεάν
Plumb’s Veterinary Drugs	3.4	Aug 5, 2020	2.6	Jun 15, 2020	$95.84/χρόνο
Vetivex Flow Rate Calculator	4.3	Jul 30, 2018	4.8	2018	Δωρεάν
ImproView	n/a	May 9, 2019	2	May 9, 2019	Δωρεάν
Vet Nurse Quick Reference	4.5	Nov. 9, 2020	–	–	$2.99

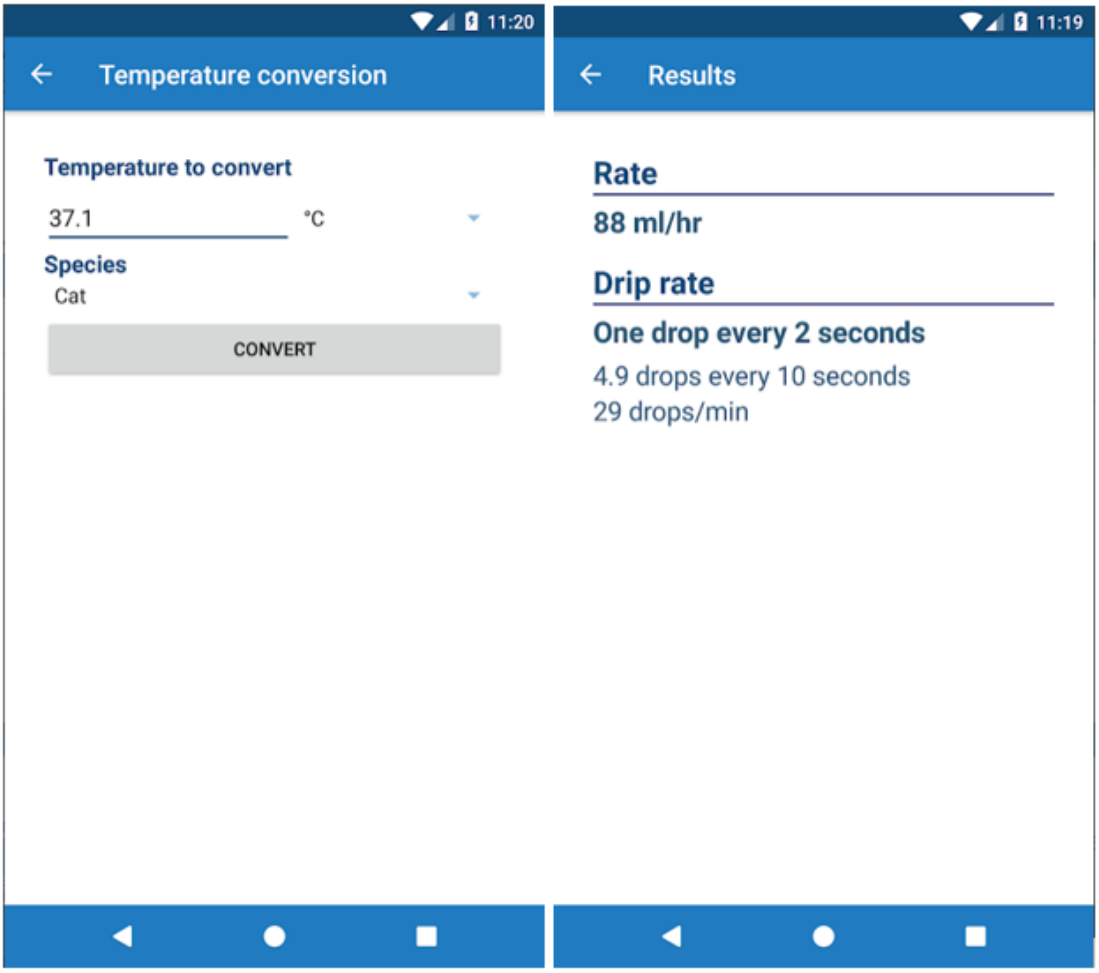
Στη συνέχεια θα παρουσιάσουμε ορισμένες εφαρμογές για να δούμε τις δυνατότητες που προσφέρουν.

Πρώτη εφαρμογή είναι το Vet Calculator το οποίο παρέχει έναν γρήγορο και εύκολο τρόπο για να γίνουν πολλοί υπολογισμοί που συνήθως εκτελούνται στην κτηνιατρική πρακτική. Επιτρέπει την εκτέλεση έντεκα διαφορετικών υπολογισμών, συμπεριλαμβανομένων δόσεων φαρμάκων, υγρών και ενεργειακών απαιτήσεων και εγχύσεων. Εκτελεί επίσης μετατροπές πολλών μονάδων, συμπεριλαμβανομένων θερμοκρασιών και μεταξύ μονάδων SI και συμβατικών μονάδων. Αναλυτική παρουσίαση των δυνατοτήτων και των υπολογισμών που μπορεί να εκτελέσει η εφαρμογή υπάρχει στη διεύθυνση: https://vetapps.co.uk/Version_Comparison.

Όπου ισχύει, τα χαρακτηριστικά του Vet Calculator αναφέρονται στην κατάλληλη βιβλιογραφία (VetApps, 2019).



Εικόνα 15 Αρχική σελίδα και υπολογισμός δόσης (VetApps, 2019)



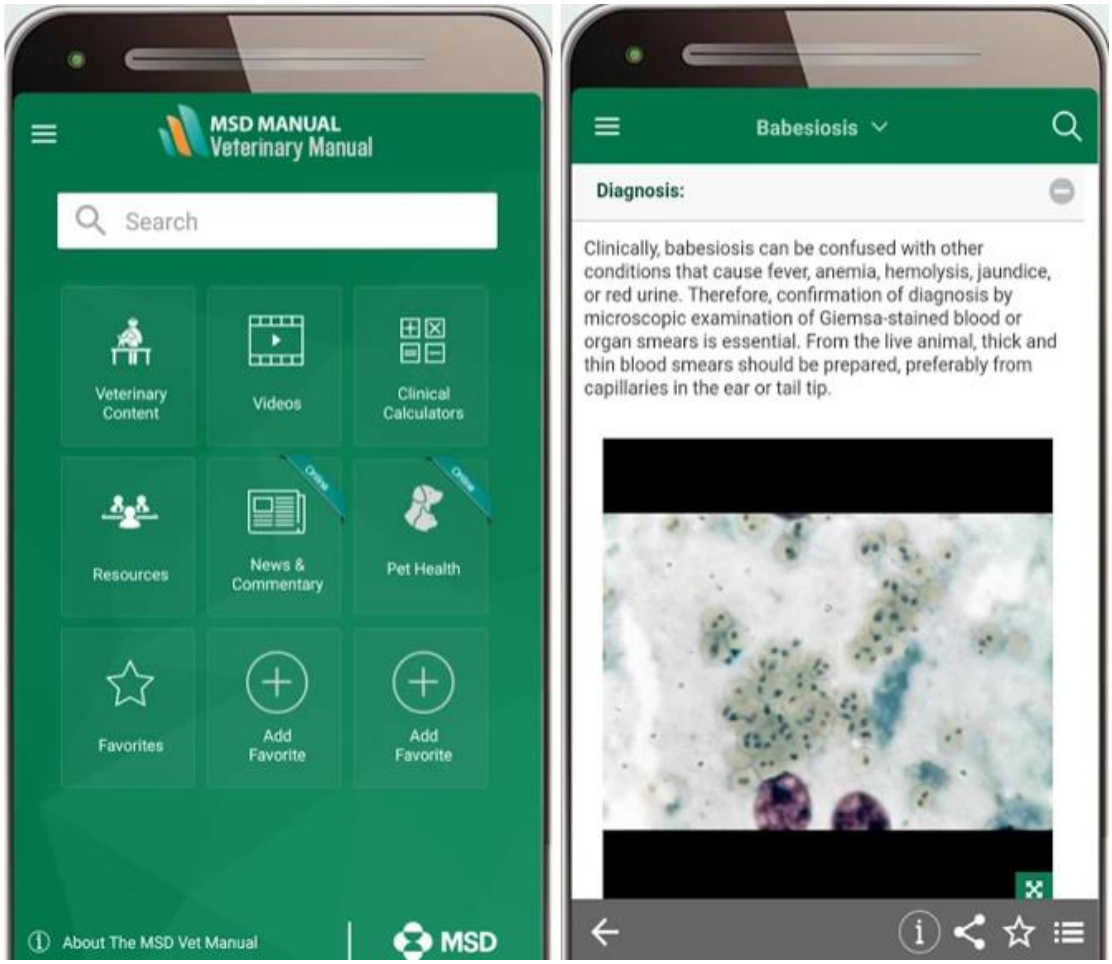
Εικόνα 16 Μετατροπέας θερμοκρασίας και υπολογισμός ρυθμού σταγόνων (VetApps, 2019)

Επόμενη εφαρμογή είναι το MSD Veterinary Manual, μια εφαρμογή για την υγεία των ζώωνκαι καλύπτει όλα τα είδη και τις διαταραχές κτηνιατρικού ενδιαφέροντος παγκοσμίως. Η εφαρμογή αυτή παρέχει στους κτηνιάτρους, τους μαθητές και άλλους επαγγελματίες της υγείας των ζώων σαφείς, πρακτικές εξηγήσεις για χιλιάδες καταστάσεις σε όλα τα συστήματα του σώματος των ζώων. Καλύπτει την αιτιολογία, την παθοφυσιολογία και τις επιλογές διάγνωσης και θεραπείας.

Η υπηρεσίες που προσφέρει η εφαρμογή MSD Veterinary είναι ενδεικτικά:

- Χιλιάδες θέματα που γράφονται και ενημερώνονται τακτικά από περισσότερους από 400 εμπειρογνώμονες κτηνιάτρουσε περισσότερες από 20 χώρες.
- Φωτογραφίες, εικόνες και βίντεο χιλιάδων διαταραχών και ασθενειών.
- Κουίζ για τον έλεγχο της γνώσης κτηνιατρικών διαταραχών, σημείων και θεραπειών.

- Διαδραστικές προσομοιώσεις περιπτώσεων - δοκιμή ικανότητας θεραπείας ενός συγκεκριμένου ασθενούς κατά τη διάρκεια της νόσου.
- Κλινικές αριθμομηχανές.
- Πολλοί οδηγοί αναφοράς και εκατοντάδες χρήσιμοι πίνακες.
- Περιεχόμενο υγείας κατοικίδιων ζώων - γραμμένο σε γλώσσα φιλική προς τον καταναλωτή, για τους πελάτες του κτηνιατρείου.
- Συχνές ερωτήσεις και οδηγός χρήστη (Merck Sharp & Dohme Corp, 2020).

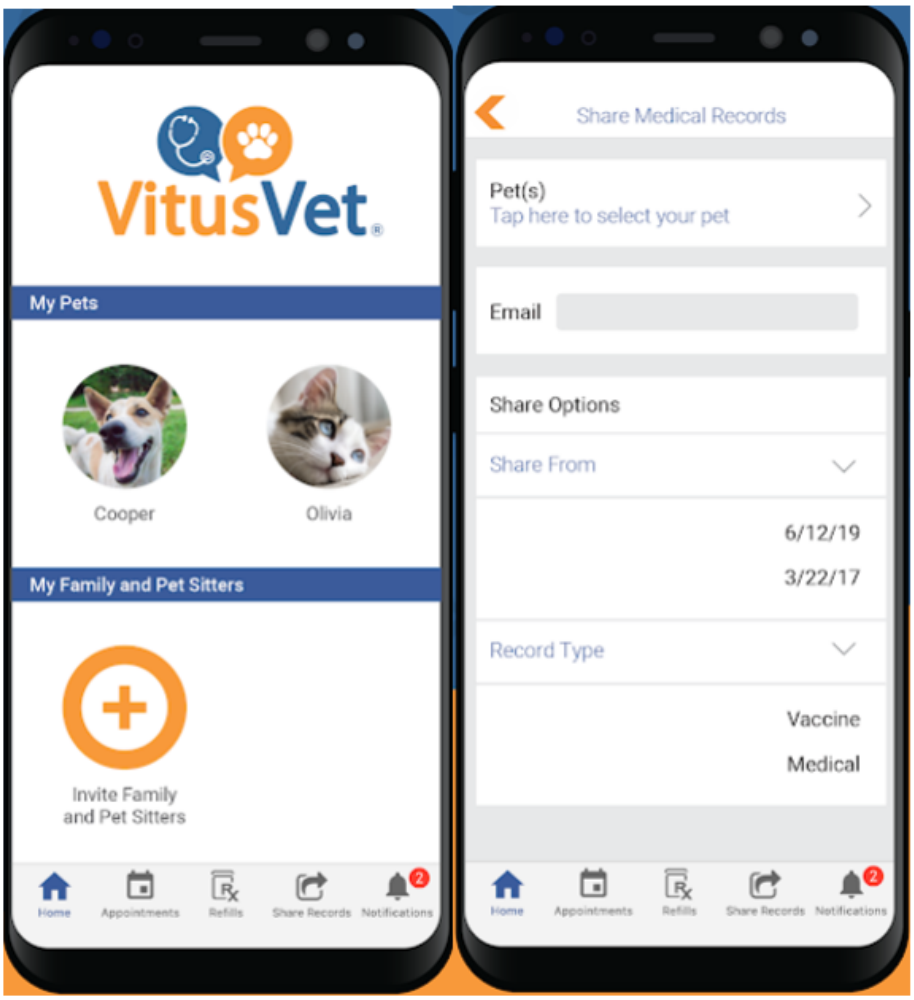


Εικόνα 17 Αρχική σελίδα και διάγνωση (Merck Sharp & Dohme Corp, 2020)

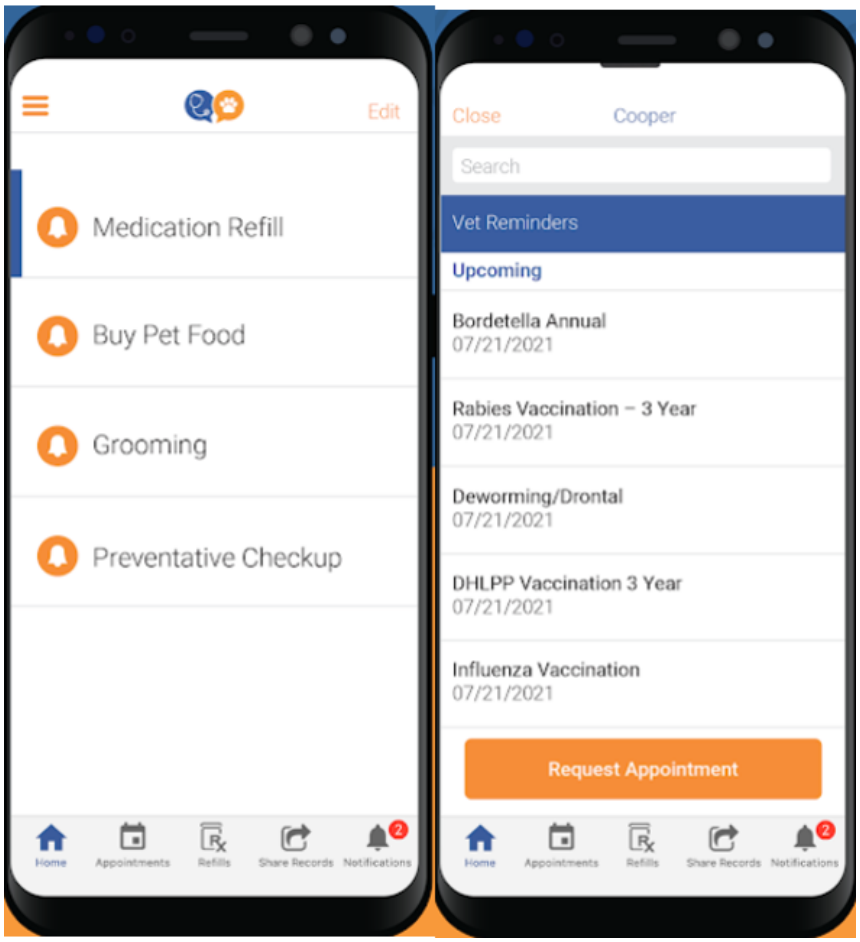
Τέλος θα δούμε την εφαρμογή VitusVet η οποία απλοποιεί την παρακολούθηση της υγείας του κατοικίδιου σας τοποθετώντας όλες τις απαραίτητες πληροφορίες σε ένα μέρος. Είναι ένα εξαιρετικό εργαλείο για όσους έχουν ένα ή περισσότερα κατοικίδια, όσους ταξιδεύουν με κατοικίδια και όσους έχουν πολλούς παρόχους φροντίδας κατοικίδιων ζώων, συμπεριλαμβανομένων κτηνιάτρων, groomers, κ.λπ..

Τα χαρακτηριστικά που προσφέρει η εφαρμογή είναι:

- Παρακολούθηση της υγείας του κατοικίδιου ζώου.
- Διαχείριση του βάρους του κατοικίδιου ζώου, παρακολούθηση των φαρμάκων, καταγραφή λεπτομερειών μικροτσίπ και ασφάλιση κατοικίδιων ζώων, καθώς και σημείωση αλλεργιών ή ιατρικών ειδοποιήσεων για γρήγορη αναφορά.
- Απόκτηση πρόσβασης στις πληροφορίες του κατοικίδιου ανά πάσα στιγμή.
- Απόκτηση πρόσβασης στα ιατρικά αρχεία του κατοικίδιου στο τηλέφωνό όπου κι αν βρισκόμαστε.Μπορούμε να έχουμε πρόσβαση στα αρχεία εμβολίων, φαρμάκων κ.λπ.
- Ορισμός λίστας υποχρεώσεων για τη διαχείριση της υγείας των κατοικίδιων.
- Ορισμός υπενθυμίσεων σε όλα όσα χρειάζεται το κατοικίδιο ζώο - από την παροχή φαρμάκων στο κατοικίδιό έως τον προγραμματισμό ραντεβού, την αγορά τροφίμων και άλλα.
- Αίτημα ραντεβού (VitusVet, 2020).



Εικόνα 18 Αρχική σελίδα και πληροφορίες ζώου (VitusVet, 2020)



Εικόνα 19 Λίστα υποχρεώσεων και αίτημα ραντεβού (VitusVet, 2020)

ΚΕΦΑΛΑΙΟ 2. ΤΕΧΝΟΛΟΓΙΕΣ ΚΑΙ ΕΡΓΑΛΕΙΑ ΔΗΜΙΟΥΡΓΙΑΣ ΕΦΑΡΜΟΓΗΣ ΚΙΝΗΤΟΥ

2.1 Εισαγωγή

Στην ενότητα αυτή θα παρουσιάσουμε τα δύο βασικά εργαλεία που χρησιμοποιήσαμε για την εφαρμογή μας. Τα εργαλεία αυτά είναι η βάση δεδομένων Firebasekαι το AndroidStudio. Τη Firebasetη χρησιμοποιήσαμε για την αποθήκευση των δεδομένων μας, ενώ με τη βοήθεια του AndroidStudioυλοποιήσαμε την εφαρμογή.

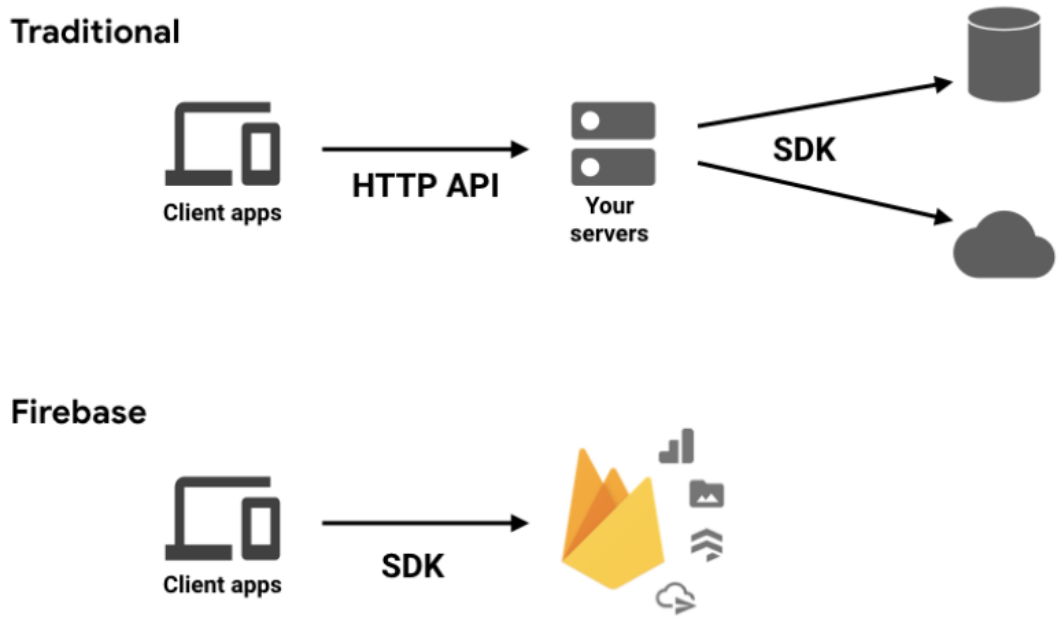
2.3.ΒάσηΔεδομένων

Η Firebase είναι μια πλατφόρμα ανάπτυξης λογισμικού που κυκλοφόρησε το 2011 από την Firebase inc και είχε υλοποιηθεί από τους Andrew Lee και James Tamplin. Η εταιρεία προσέφερε αρχικά ένα API για προγραμματιστές για τη διευκόλυνση της ενσωμάτωσης διαδικτυακών συνομιλιών για ιστότοπους.Οι προγραμματιστές βασίζονταν στην πλατφόρμα για συγχρονισμό δεδομένων εφαρμογών σε πραγματικό χρόνο. Οι Lee και Tamplin βλέποντας τη χρήση της εφαρμογής τους, αποφάσισαν να διαφοροποιήσουν την αρχιτεκτονική σε πραγματικό χρόνο από το σύστημα συνομιλίας, μια κίνηση που οδήγησε στην ίδρυση της Firebase το 2011. Η πλατφόρμα κυκλοφόρησε δημόσια τον Απρίλιο του 2012. Το πρώτο προϊόν Firebase που κυκλοφόρησε ήταν η Βάση δεδομένων Firebase Realtime. Είναι ένα API για συγχρονισμό δεδομένων εφαρμογών σε συσκευές Android, web και iOS.Η Firebase συγκέντρωσε χρηματοδότηση νεοφυών επιχειρήσεων άνω των 1 εκατομμυρίων δολαρίων το 2012 από συνεργάτες. Το Firebase Authentication και το Firebase Hosting κυκλοφόρησαν το 2014 από την Firebase, καθιστώντας την εταιρεία ως κορυφαίο mobile backend as a service (MbaaS) (Batschinski, n.d.).

HFirebaseαποκτήθηκε από την Google το 2014. Αυτή τη στιγμή διαθέτει έναν αριθμό από υπηρεσίες και ειδικά API. Ολόκληρη η πλατφόρμα είναι μια λύση Backend-as-a-Service τόσο για εφαρμογές για κινητά όσο και για Web που περιλαμβάνει υπηρεσίες για τη δημιουργία, τον έλεγχο και τη διαχείριση εφαρμογών.

Οι λύσεις BaaS μας επιτρέπουν να εξαλείψουμε την ανάγκη διαχείρισης βάσεων δεδομένων backend και απόκτησης αντίστοιχου υλικού. Στην παραδοσιακή ανάπτυξη

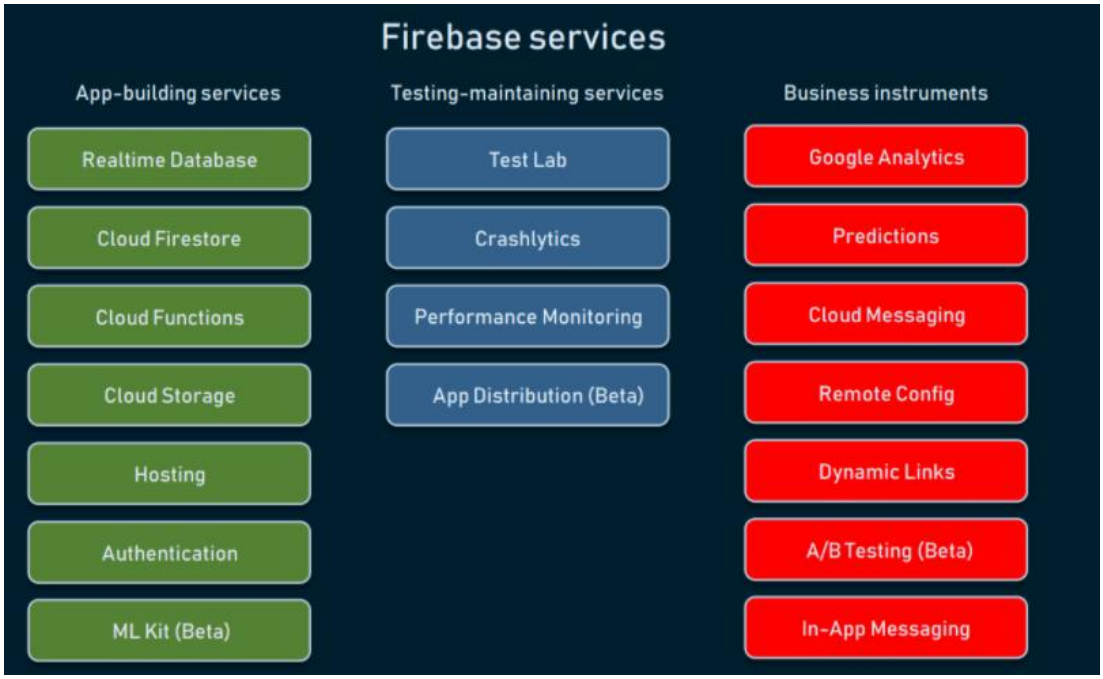
εφαρμογών περιλαμβάνεται η σύνταξη λογισμικού frontend και backend. Ο κωδικός frontend απλώς επικαλείται API τελικών σημείων που εκτίθενται από το backend και ο κώδικας backend είναι αυτός που πραγματικά εκτελεί τις λειτουργίες προς και από τη βάση δεδομένων. Ωστόσο, με τη βοήθεια της Firebase, το παραδοσιακό backend παρακάμπτεται, μεταφέρονταςτην εργασία στο λογισμικό του πελάτη.



Εικόνα20Traditional vs Firebase application (Stevenson, 2018)

Αντ'αυτού, μπορούμε να συνδέσουμε την εφαρμογή μας μέσω αποκλειστικών API για κάθε ξεχωριστή υπηρεσία. Στην περίπτωση της Firebase, υπάρχουν πολλά από αυτά που καλύπτουν όλο το φάσμα των τεχνολογιών υποστήριξης για μια εφαρμογή. Η λίστα των πλατφορμών που ενσωματώνει το Firebase περιλαμβάνει Android, iOS, Web και Unity.

Ποιες υπηρεσίες διαθέτει όμως η Firebase; Αν επισκεφθούμε τη σελίδα της θα δούμε ότι διαθέτει τρεις κατηγορίες υπηρεσιών. Αυτές χωρίζονται σε υπηρεσίες για δημιουργία εφαρμογών, διασφάλιση ποιότητας και μέσα για ανάπτυξη επιχειρήσεων.



Εικόνα21Firebase services (AltexSoft, 2019)

Στιςβάσεις δεδομένων συναντάμε τα:

- Firebase Realtime Database: το οποίο ήταν το πρώτο προϊόν που εμφανίστηκε με τη Firebase. Θεωρείτε η πιο καθιερωμένη και σταθερή υπηρεσία σε ολόκληρη την πλατφόρμα. Η βάση δεδομένων Realtime είναι ουσιαστικά μια αποθήκευση cloud NoSQL που μπορεί να συνδεθεί με την εφαρμογή για να παρέχει πρόσβαση σε πραγματικό χρόνο στα δεδομένα σε διαφορετικές πλατφόρμες. Ένα από τα πλεονεκτήματα είναι ότι η βάση δεδομένων μπορεί να λειτουργήσει εκτός σύνδεσης, προσωρινά αποθηκεύοντας τα δεδομένα στη μνήμη της συσκευής και μετά την επανασύνδεση στο Διαδίκτυο, να τα συγχρονίσει με τη βάση δεδομένων. Τα δεδομένα αποθηκεύονται σε μορφή JSON και οι χρήστες μπορούν να τα αναζητήσουν με ειδικές εντολές ερωτημάτων. Όσον αφορά την ασφάλεια, η βάση δεδομένων Realtime παρέχει πρόσβαση σε δεδομένα βάσει άδειας. Αυτό μπορεί να γίνει με τη βοήθεια του Firebase Authentication και με την παροχή δικαιωμάτων με βάση κανόνων ταυτότητας χρήστη ή ασφάλειας.
- Το Cloud Firestore είναι μια άλλη βάση δεδομένων NoSQL σε πραγματικό χρόνο που φιλοξενείται από το cloud. Σε αντίθεση με το Firebase Realtime Database, το Cloud Firestore έχει σχεδιαστεί για εταιρική χρήση, η οποία

συνεπάγεται επεκτασιμότητα, σύνθετα μοντέλα δεδομένων και προηγμένες επιλογές ερωτήματος.

Στο σημείο αυτό θα πρέπει να αναφέρουμε ότι η κονσόλα της Firebase μπορεί να χρησιμοποιηθεί για την προβολή δεδομένων και στις δύο βάσεις δεδομένων. Ένα άλλο αμοιβαίο σημείο είναι ότι υπάρχουν SDK για εργασία με κώδικα διακομιστή και των δύο βάσεων δεδομένων. Αυτά είναι διαθέσιμα για Python, Node.js, Golang, Ruby, PHP, Java, .NET και C #.

Η τελευταία βάση δεδομένων που συναντάμε είναι το Cloud Storage. Για την ακρίβεια δεν είναι ακριβώς βάση δεδομένων αλλά κάτι σαν το Google Cloud, όπου οι χρήστες μπορούν να ανεβάσουν περιεχόμενο, όπως αρχεία, φωτογραφίες, βίντεο κ.λπ.

Θα πρέπει επίσης να αναφέρουμε και το Authentication. Η firebaseυποστηρίζει ένα πλήθος από δυνατότητες για authenticationχρηστών στη βάση δεδομένων. Με τη βοήθεια της συγκεκριμένης υπηρεσίας μπορούμε να επιτύχουμε ασφαλείς συνδέσεις σε άλλες υπηρεσίες π.χ. στη βάση δεδομένων (AlexSoft, 2019).

2.4 AndroidStudio

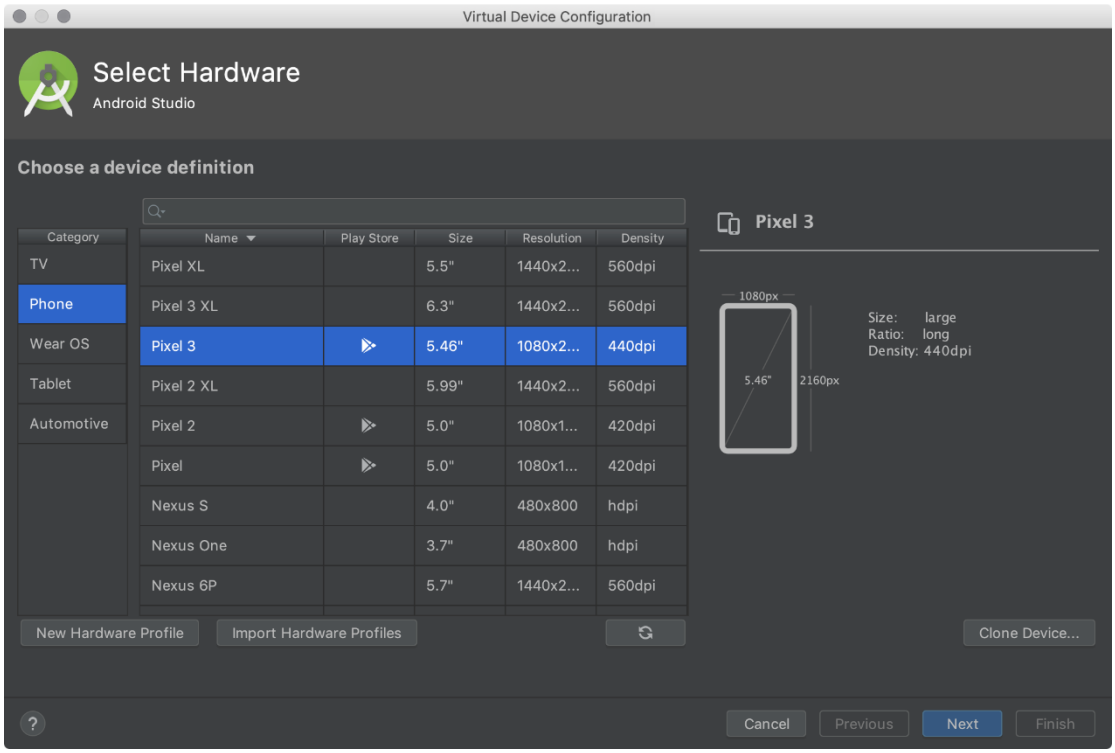
Το Android Studio ανακοινώθηκε για πρώτη φορά σε μια διάσκεψη της Google το 2013 και κυκλοφόρησε στο ευρύ κοινό το 2014 μετά από διάφορες εκδόσεις beta. Πριν από την κυκλοφορία του, η ανάπτυξη εφαρμογών Android γινόταν κυρίως μέσω του Eclipse IDE, το οποίο είναι ένα πιο γενικό Java IDE που υποστηρίζει επίσης πολλές άλλες γλώσσες προγραμματισμού.

Εικόνα 22Android Studio

Το Android Studio υποστηρίζει τους προγραμματιστές, απλοποιώντας πολλά τμήματα σε σύγκριση με το μη εξειδικευμένο λογισμικό, αλλά έχει ακόμα λίγο δρόμο, μέχρι να προσφέρει μια πλήρως διαισθητική και ομαλή εμπειρία.

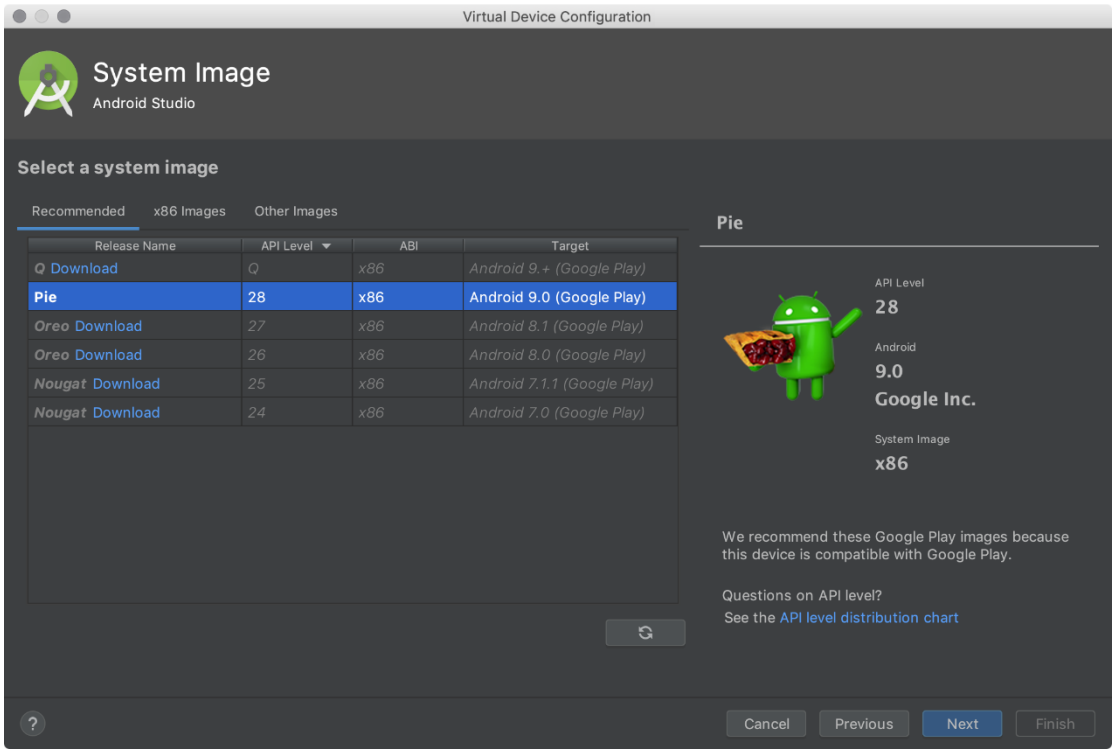
Ως IDE τότε, η δουλειά του Android Studio είναι να παρέχει τη διεπαφή στους προγραμματιστές, ώστε να μπορούν να δημιουργήσουν τις εφαρμογές και να χειρίζονται μεγάλο μέρος της περίπλοκης διαχείρισης αρχείων με απλοποιημένο τρόπο. Η γλώσσα προγραμματισμού που υποστηρίζει είναι είτε Java είτε Kotlin. Το Android Studio παρέχει υποστήριξη στους προγραμματιστές για τη συγγραφή, επεξεργασία και αποθήκευση των έργων αλλά και των αρχείων που αυτά περιλαμβάνουν. Ταυτόχρονα, το Android Studio επιτρέπει επίσης την εκτέλεση του κώδικα που έχουμε γράψει, είτε μέσω εξομοιωτή είτε μέσω ενός υλικού που είναι συνδεδεμένο στο μηχάνημά σας. Οποιαδήποτε από τις δύο επιλογές διαλέξουμε θα μπορούμε να εκτελέσουμε εντοπισμό σφαλμάτων του προγράμματος καθώς εκτελείται και να λαμβάνουμε σχόλια που να εξηγούν τα σφάλματα, ώστε να μπορούμε να λύσουμε πιο γρήγορα το πρόβλημα.

Πριν ολοκληρώσουμε την παρουσίαση του AndroidStudioθα πρέπει να αναφέρουμε το AVDManager. Αναφέραμε παραπάνω ότι μπορούμε να εκτελούμε τον κώδικα που έχουμε γράψει σε έναν εξομοιωτή. Για τη δημιουργία του εξομοιωτή χρησιμοποιούμε το AVDManager. Όταν το καλέσουμε επιλέγουμε τη συσκευή που θέλουμε να δημιουργήσουμε:



Εικόνα 23AVDManagerδημιουργία συσκευής (Mullis, 2017)

Στη συνέχεια επιλέγοντας Nextθα πρέπει να επιλέξουμε το SystemImageπου θέλουμε να έχει η συσκευή μας. Δηλαδή ποια έκδοση API, θέλουμε να εγκαταστήσουμε. Αν, για παράδειγμα, δεν υπάρχει κάποια, τότε βλέπουμε την ένδειξη Download, όπου πατώντας τη, μπορούμε να την κατεβάσουμε και να την εγκαταστήσουμε.

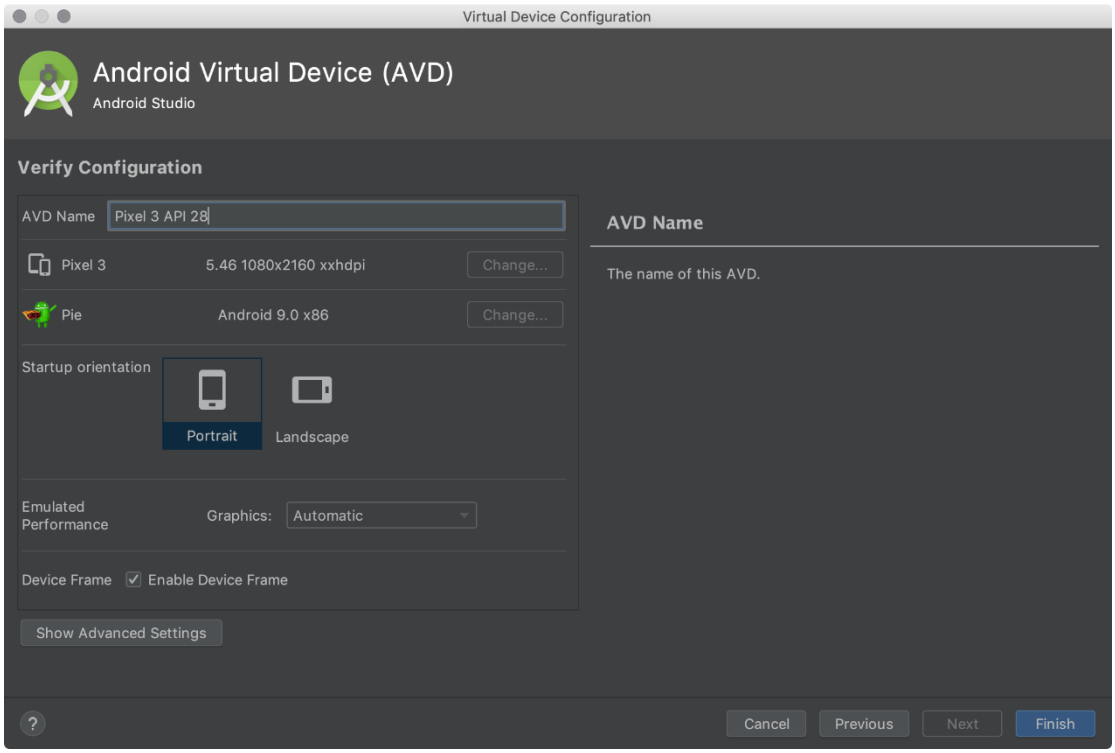


Εικόνα 24 Επιλογή System Image

Για να ολοκληρώσουμε τη δημιουργία της συσκευής θα πρέπει:

- να αναθεωρήσουμε τις επιλογές μας,
- να της δώσουμε ένα όνομα
- να επιλέξουμε προσανατολισμό

και να πατήσουμε το πλήκτρο Finishγια τη δημιουργία της εικονικής συσκευής.



Εικόνα 25 Ολοκλήρωση δημιουργίας εικονικής συσκευής

Πλέον στον AVDManagerθα έχει προστεθεί η εικονική συσκευή μας και θα μπορούμε να την ξεκινήσουμε, να την τροποποιήσουμε ή να τη διαγράψουμε (Mullis, 2017).

Εικόνα 26AVD Manager

ΚΕΦΑΛΑΙΟ 3. ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ ΕΞΥΠΝΟΥ ΚΙΝΗΤΟΥ

3.1 Σχεδιασμός

3.1.1 Εισαγωγή

Στην ενότητα αυτή θα παρουσιάσουμε τα βήματα που ακολουθήσαμε για το σχεδιασμό της εφαρμογής μας. Θα ξεκινήσουμε παρουσιάζοντας τα UserStories, δηλαδή τις δυνατότητες που θέλουμε να έχουν οι χρήστες της εφαρμογής μας. Στη συνέχεια θα δούμε τα Wireframes, τους πρόχειρους σχεδιασμούς των οθονών της εφαρμογής.

3.1.2 UserStories

Ένα UserStory είναι μια περιγράφη, μια γενική εξήγηση μιας δυνατότητας λογισμικού που γράφεται από την οπτική γωνία του τελικού χρήστη. Μέσα από τα UserStoryπροσπαθούμε με τη βοήθεια μιας μη τεχνικής γλώσσας να καθορίσουμε το πλαίσιο λειτουργίας της ομάδας που θα υλοποιήσει την εφαρμογή, δηλαδή να γνωρίζει η ομάδα τί λειτουργίες – δυνατότητες θα πρέπει να έχει η εφαρμογή που θα υλοποιήσει.

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

As a [persona], I [want to], [so that].

Όπου θα πρέπει να συμπληρώσουμε:

- As a [persona]: ποιος είναι ο χρήστης για τον οποίο θέλουμε να εκτελεστούν οι συγκεκριμένες ενέργειες.
- Wantto: τί θέλουμε να πετύχουμε; Περιγράφουμε την πρόθεσή και τις λειτουργίεςπου θέλουμε εκτελεσθούν.
- So that: τί θέλουμε να κερδίσει ο χρήστης (Rehkopf, n.d.).

Λαμβάνοντας τα παραπάνω υπόψη μας προχωρήσαμε στην καταγραφή των ακόλουθων Userstories:

- Σαν χρήστης της εφαρμογήςθα μπορώ να συμπληρώσω το emailκαι το passwordγια να εγγραφώ.

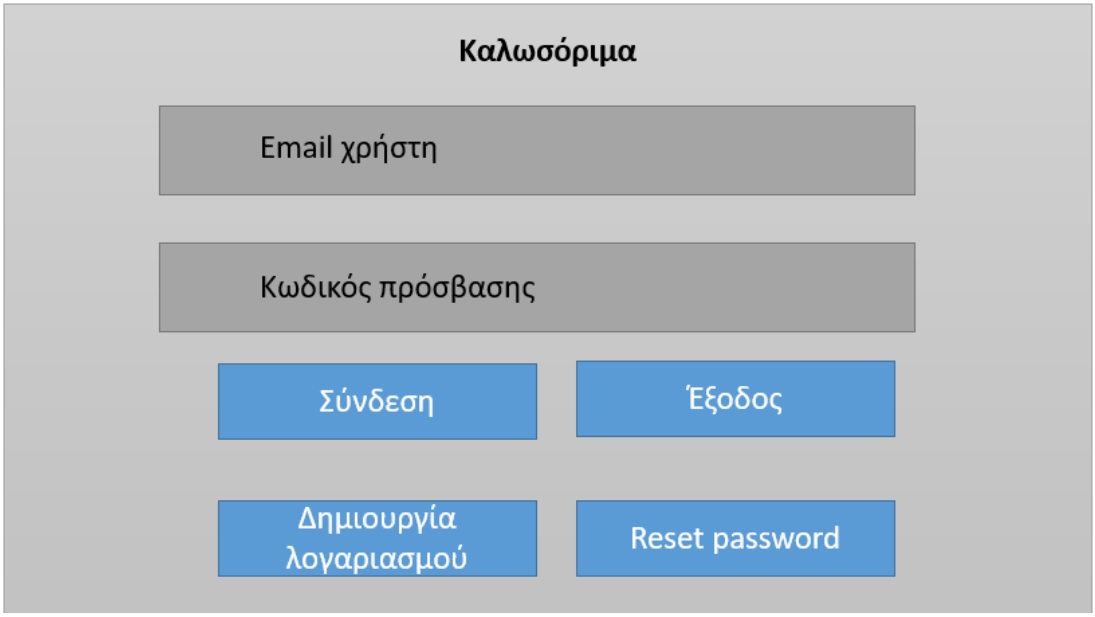
- Σαν χρήστης της εφαρμογής θα μπορώ να εισάγω το emailκαι το passwordγια να συνδεθώ στην εφαρμογή.
- Σαν χρήστης της εφαρμογής θα μπορώ να επιλέξω την επαναδημιουργία του passwordμου για να το δημιουργήσω ξανά αν το έχω ξεχάσει.
- Σαν συνδεδεμένος χρήστης της εφαρμογής θα μπορώ να εισάγω τα προσωπικά μου στοιχεία για καλύτερη ενημέρωσή μου.
- Σαν συνδεδεμένος χρήστης της εφαρμογής θα μπορώ να επικαιροποιώ τα προσωπικά μου στοιχεία για καλύτερη ενημέρωσή μου.
- Σαν συνδεδεμένος χρήστης της εφαρμογής θα μπορώ να προσθέσω ένα ζώο που έχω για να μπορώ να προσθέτω εμβόλια και θεραπείες.
- Σαν συνδεδεμένος χρήστης της εφαρμογής θα μπορώ να καταγράφω τα εμβόλια στα ζώα που έχω προσθέσει για να έχω ιστορικό τους.
- Σαν συνδεδεμένος χρήστης της εφαρμογής θα μπορώ να καταγράφω τις θεραπείες των ζώων που έχω προσθέσει για να έχω το ιστορικό τους.

3.1.3 Wireframes

Για το σχεδιασμό των Wireframesχρησιμοποιήσαμε το PowerPoint. Με τη βοήθειά του σχεδιάσαμε τις ακόλουθες οθόνες τις οποίες παρουσιάζουμε εξηγώντας παράλληλα και τη λειτουργικότητα που υποστηρίζουν.

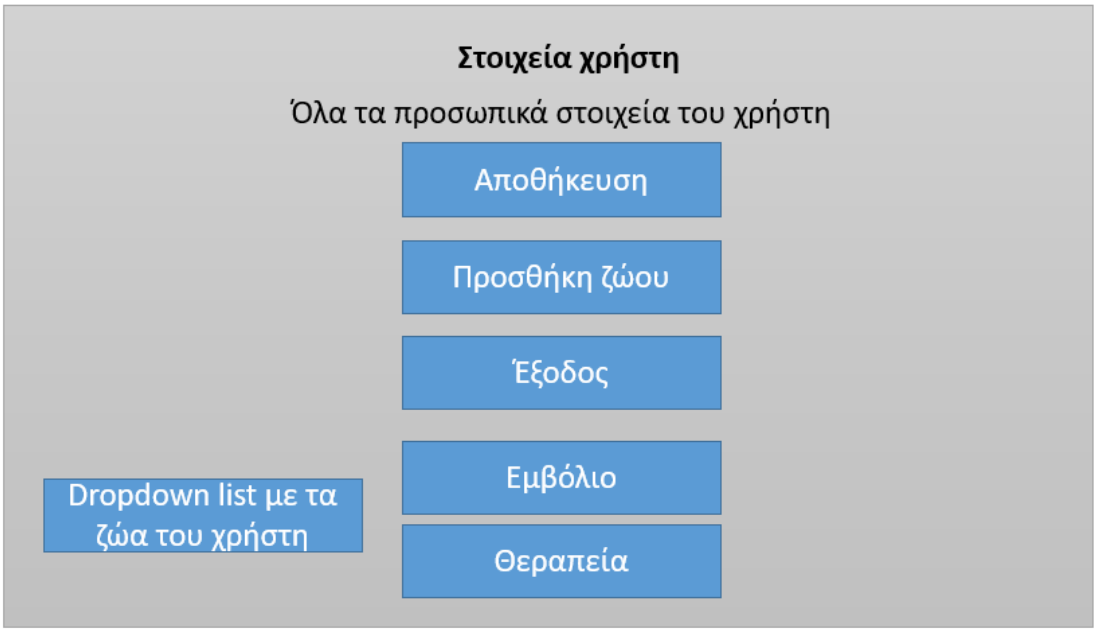
Στην αρχική οθόνη της εφαρμογής μας θα υπάρχει ένα καλωσόρισμαστην κορυφή και στη συνέχεια θα ζητάμε το emailκαι το passwordτου χρήστη. Με αυτά τα στοιχεία μόνο ο χρήστης θα μπορεί να εγγραφεί στην εφαρμογή, πατώντας το Έγγραφή. Αν πατήσει το Σύνδεση θα πρέπει να ελεγχθούν θα στοιχεία του και αν είναι σωστά να μεταφερθεί στην κεντρική οθόνη της εφαρμογής μας.

Αν πατήσει ResetPasswordσημαίνει ότι δε θυμάται το passwordπου έχει δώσει και πρέπει η εφαρμογή να τον υποστηρίξει για να εισάγει νέο. Τέλος θα πρέπει να μπορεί να φύγει από την εφαρμογή με το Έξοδος.



Η κύρια οθόνη της εφαρμογής μας είναι η επόμενη. Ο χρήστης θα βλέπει τα στοιχεία του, την πρώτη φορά δε θα υπάρχουν, οπότε θα μπορεί να τα εισάγει και πατώντας το Αποθήκευση να τα αποθηκεύει. Τις επόμενες φορές θα μπορεί να τα τροποποιήσει και με το αποθήκευση να ενημερώσει τις αλλαγές.

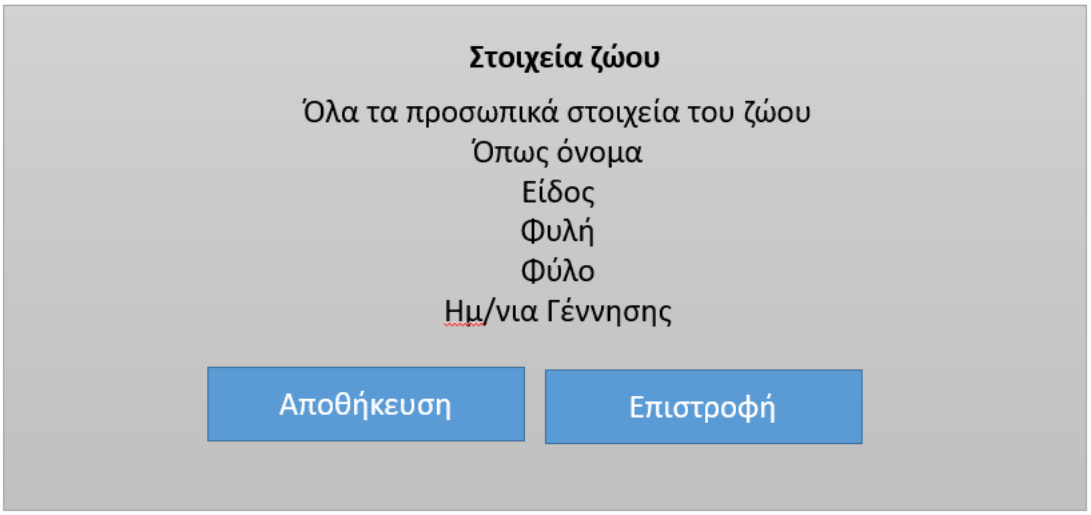
Ο χρήστης θα μπορεί να προσθέσει ένα ζώο που του ανήκει και θα εμφανιστεί στο Dropdownlistμε τα ζώα του χρήστη. Με το έξοδος θα βγαίνει από την εφαρμογή. Με το Εμβόλιο ή το Θεραπεία θα καλεί την αντίστοιχη οθόνη για εισαγωγή εμβολίου ή θεραπείας.



Όταν ο χρήστης επιλέξει να εισάγει ζώο, θα πρέπει να εισάγει τα στοιχεία του όπως:

- Όνομα
- Είδος
- Φυλή
- Φύλο
- Ημερομηνία Γέννησης

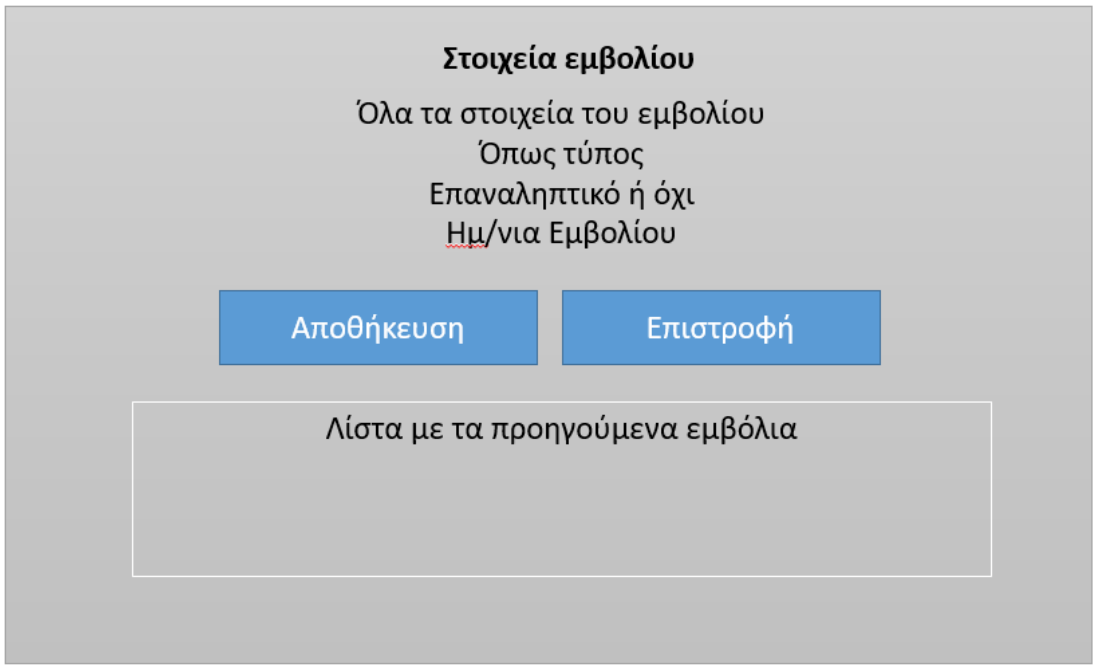
Και πατώντας το αποθήκευση θα αποθηκεύεται και θα γίνεται η αντιστοίχιση χρήστη – ζώου.



Όταν ο χρήστης επιλέξει να εισάγει εμβόλιο για ένα ζώο από τη λίστα θα πρέπει να εισάγει τα ακόλουθα στοιχεία του εμβολίου όπως:

- Τύπος
- Αν είναι επαναληπτικό ή όχι
- Ημερομηνία εμβολίου.

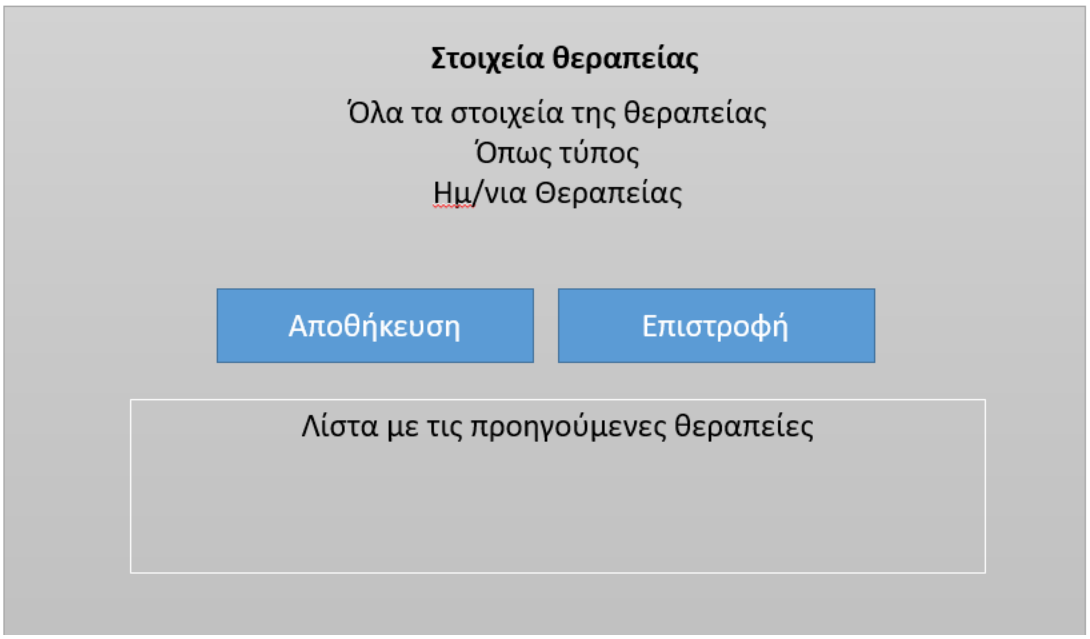
Πατώντας το αποθήκευση θα αποθηκεύεται το εμβόλιο στο ιστορικό εμβολιασμών του ζώου. Κάτω από τα πλήκτρα θα υπάρχει μια λίστα με το ιστορικό των εμβολίων που έχουν καταχωριστεί για το ζώο, για να έχουμε μια γενική εικόνα των εμβολιασμών.



Τέλος στην οθόνη της θεραπείας θα μπορούμε να εισάγουμε θεραπεία για το ζώο που έχουμε επιλέξει. Τα στοιχεία που πρέπει να εισάγουμε είναι:

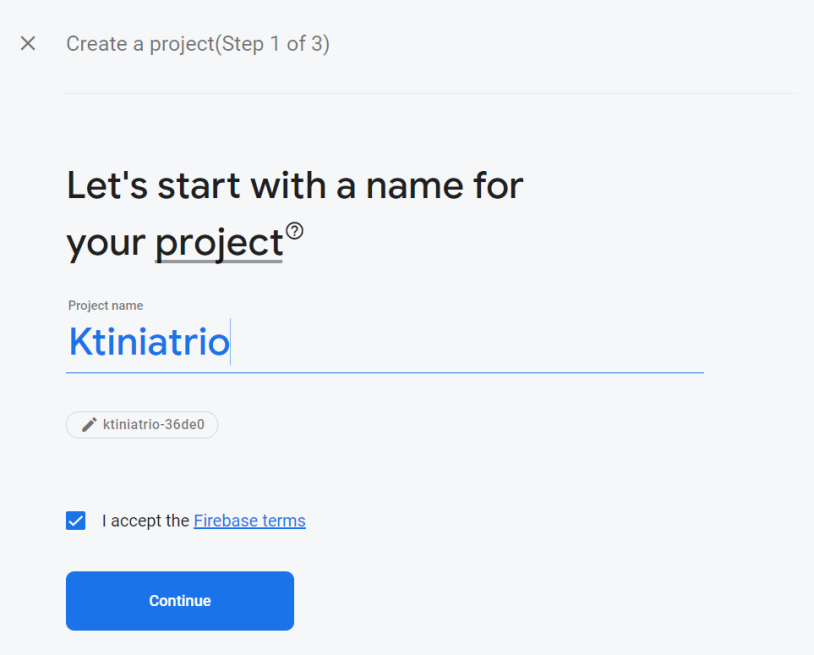
- Τύπος θεραπείας
- Ημερομηνία Θεραπείας.

Αν πατήσουμε το αποθήκευση θα αποθηκευτεί η θεραπεία στις θεραπείες του ζώου. Κάτω από τα buttonθα υπάρχει ένα ιστορικό με όλες τις θεραπείες που έχουν καταχωριστεί για το ζώο.

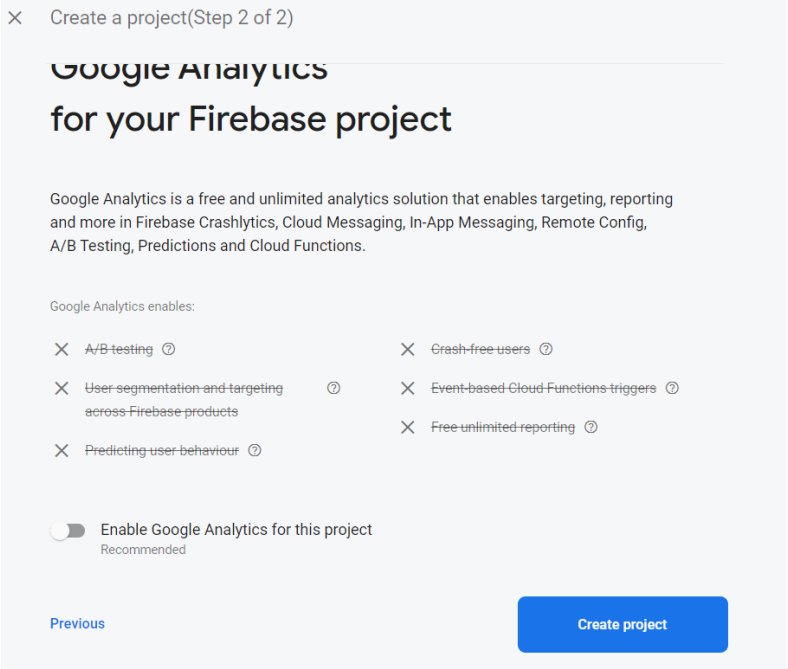


3.2 Υλοποίηση

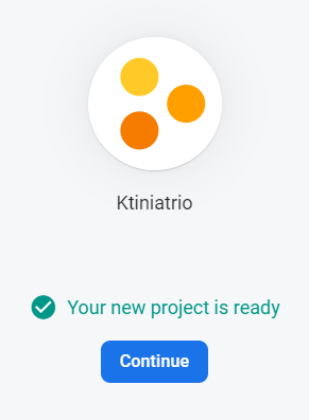
Για την υλοποίηση της εφαρμογής θα πρέπει να ξεκινήσουμε από τη διαμόρφωση της Firebase. Αρχικά θα πρέπει να δημιουργήσουμε ένα νέο project, όπως στην εικόνα που ακολουθεί. Το project το ονομάσαμε Ktiniatrio.



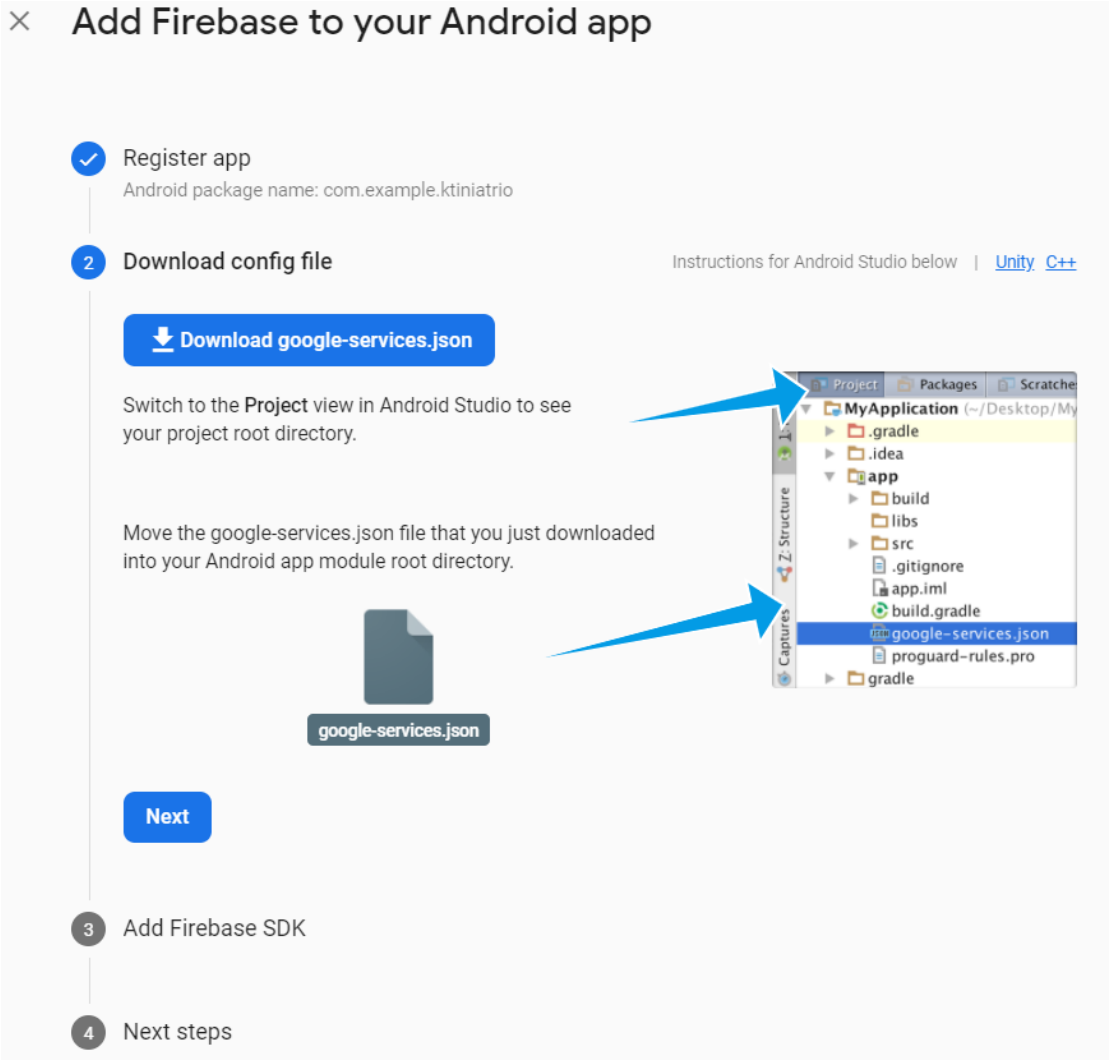
Στη συνέχεια επιλέγουμε αν θέλουμε να ενεργοποιήσουμε τα Google Analytics για το project μας.



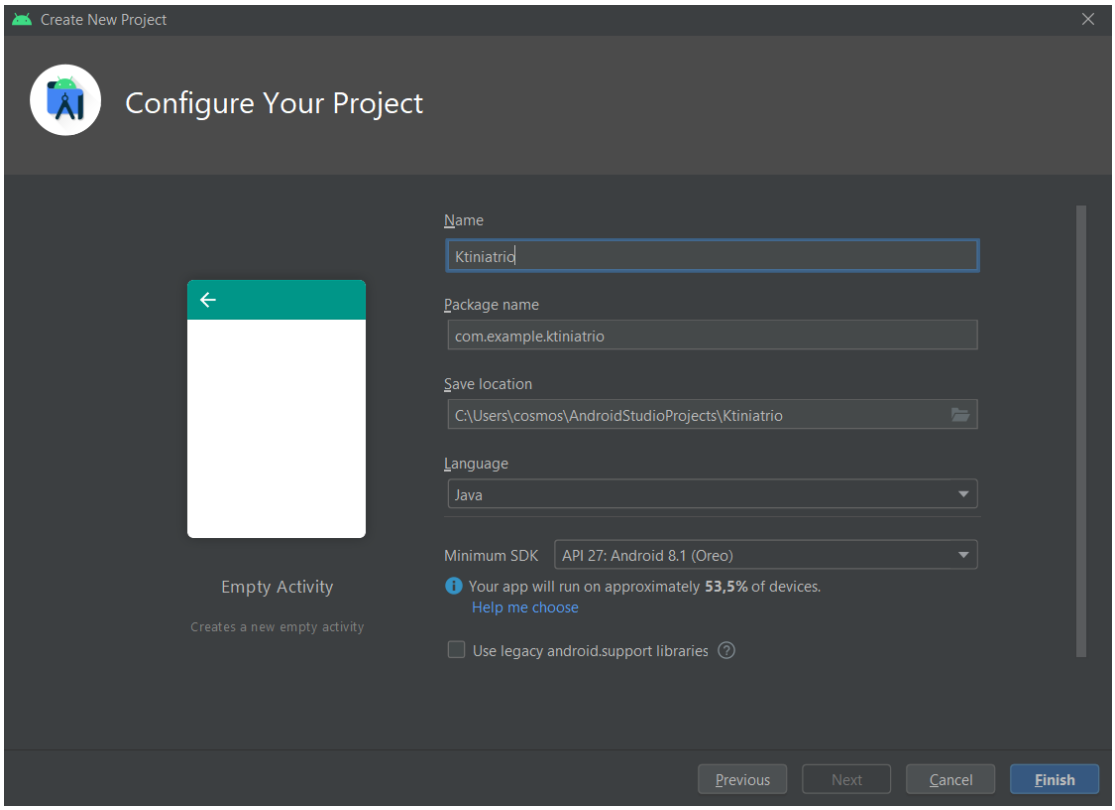
Το επόμενο βήμα είναι ενημερωτικό όπου μπορούμε να δούμε ότι έχει δημιουργηθεί το projectκαι πρέπει να πατήσουμε το Continue.



Κατόπιν ακολουθούν τα βήματα για να προσθέσουμε τη Firebase στο Android application που έχουμε δημιουργήσει.



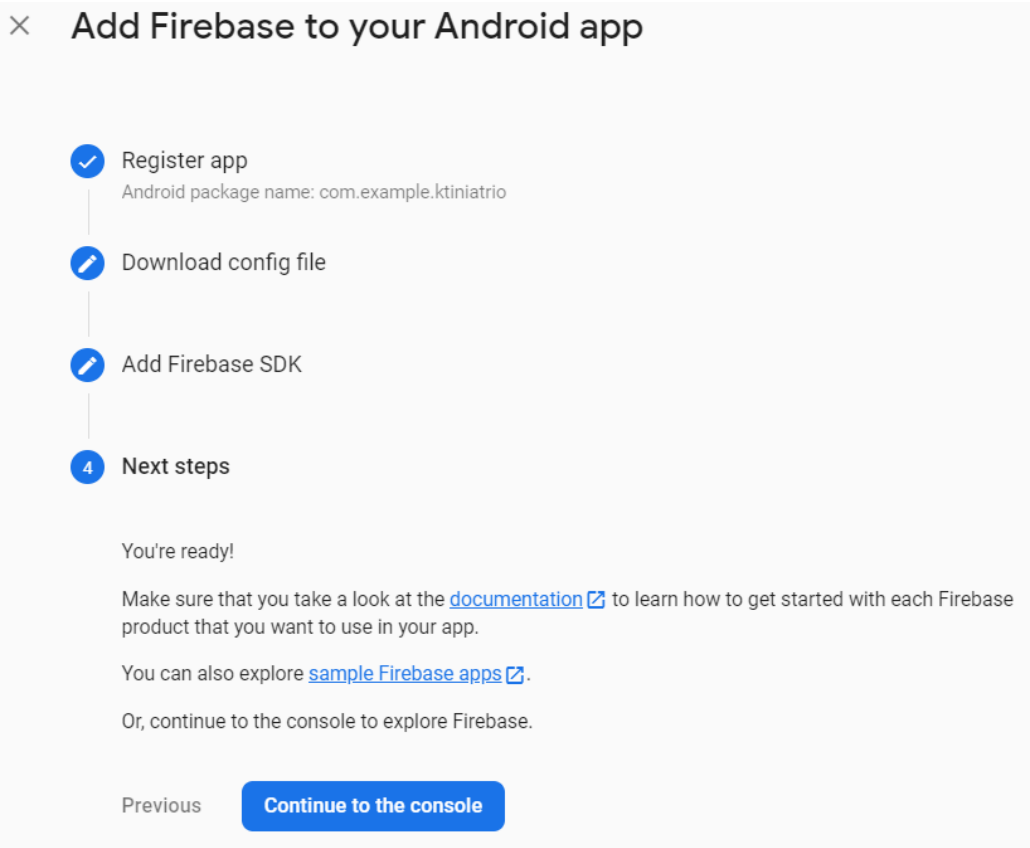
Στο σημείο αυτό θα μπορούσαμε να ξεκινήσουμε και το AndroidStudioκαι να δημιουργήσουμε ένα Androidapplicationτο οποίο θα παραμετροποιήσουμε κατάλληλα. Ζητώντας από το AndroidStudiova δημιουργήσουμε ένα νέο applicationθα μας εμφανίσει μια οθόνη όπου μας ζητά να συμπληρώσουμε τα στοιχεία του application:



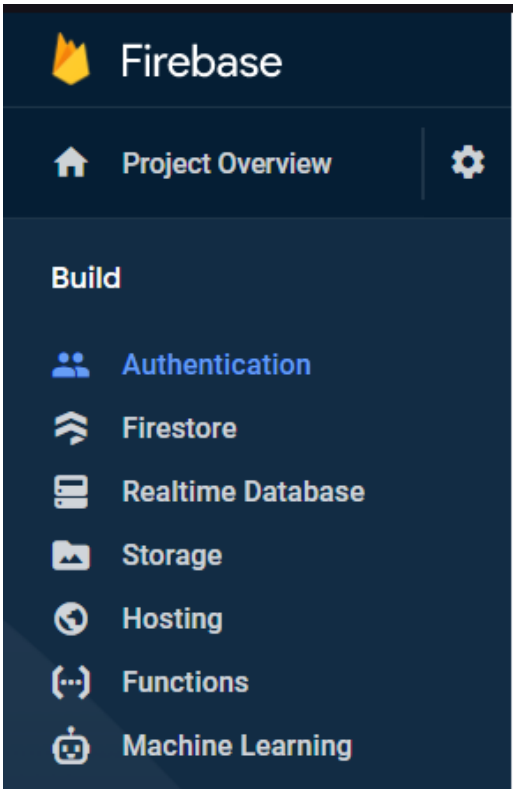
Συμπληρώνουμε το όνομα (Ktiniatrio) και το που θα είναι αποθηκευμένο. Επίσης επιλέγουμε τη γλώσσα προγραμματισμού να είναι η Javaκαι το minimumSDKνα είναι το API 27 δηλαδή το Android 8.1. Όπως μας ενημερώνει το AndroidStudion εφαρμογή μας θα εκτελείται στο 53,5% των συσκευών.

Πατώντας το Finishκαι περιμένοντας λίγο (ανάλογα με την ταχύτητα του υπολογιστή μας) θα έχει δημιουργηθεί το projectεμφανίζοντάς μας την MainActivityπου θα είναι η κεντρική Activityγια την εφαρμογή μας.

Ακολουθώντας τα βήματα αυτά και ενημερώνοντας κατάλληλα το Androidapplication, θα εμφανιστεί η ακόλουθη οθόνη όπου θα μας ενημερώνει ότι έχουμε ολοκληρώσει όλα τα βήματα και μπορούμε να συνεχίσουμε παραμετροποιώντας την βάση δεδομένων Firebase, προσθέτοντας λειτουργίες κ.λπ.

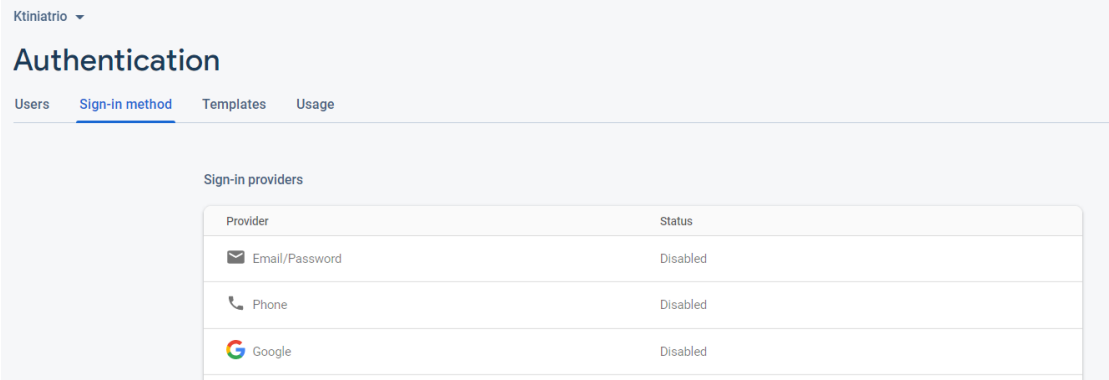


ΠατώνταςτοContinuetotheconsoleθαμεταφερθούμε στην κονσόλα λειτουργίας της Firebase.

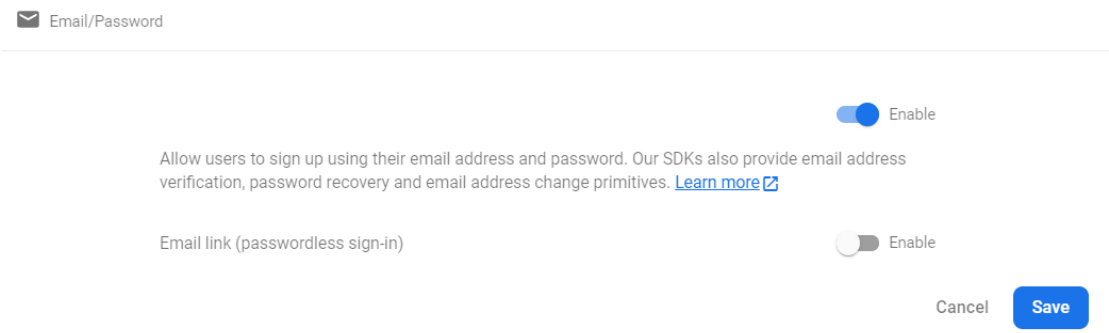


Στην κονσόλα αυτή μπορούμε να δούμε τις δυνατότητες της Firebase. Εμείς θα ενεργοποιήσουμε το Authenticationγια να μπορούν να εγγράφονται οι χρήστες με το emailκαι το passwordπου θέλουν και το RealtimeDatabaseγια να αποθηκεύουμε τα δεδομένα μας.

ΕπιλέγουμεAuthentication→Sign-inmethodκαιεμφανίζεται η ακόλουθη εικόνα όπου μπορούμε να καθορίσουμε τον τρόπο που θα γίνεται το Signin. Επιλέγουμε το Email/Password.



Αμέσως μετά θα πρέπει να καθορίσουμε τις επιλογές που θέλουμε για το AuthenticationEmail/Password. Επιλέγουμε ότι οι χρήστες θα συνδέονται με το emailκαι το passwordπου δίνουν. Όπως μας ενημερώνει η Firebaseπαρέχει emailverificationκαι passwordrecovery. Δεν ενεργοποιούμε το Emaillink, που σημαίνει ότι οι χρήστες δε θα συμπληρώνουν κάποιο password.



Προχωράμε στην υλοποίηση του application και δημιουργούμε τα κατάλληλα Activityγια τις λειτουργίες της εφαρμογής μας.
Τα Activityπου υπάρχουν στο applicationείναι:

- MainActivity: το κεντρικό Activityτης εφαρμογής μας η οποία ελέγχει τη δυνατότητα σύνδεσης με τη βάση δεδομένων, δέχεται τα στοιχεία του χρήστη, ελέγχει αν ο χρήστης έχει πατήσει Σύνδεση ή Εγγραφή ή Resetpasswordκαι ανάλογα εκτελεί τις κατάλληλες ενέργειες.
- FirstPageActivity: η κύρια Activityτης εφαρμογής μας. Εμφανίζει τα στοιχεία του χρήστη, τα οποία ο χρήστης μπορεί να τα προσθέσει κατά τη σύνδεσή του ή να τα τροποποιήσει και τις λειτουργίες που θέλει να εκτελέσει, όπως προσθήκη Ζώου, Εμβολιασμός, Θεραπείες ή έξοδος.
- EmvolioActivity
- AnimalActivity: είναιτοActivityγιαπροσθήκη ζώου στη βάση δεδομένων και αντιστοίχισή του με το χρήστη.
- TherapeiaActivity

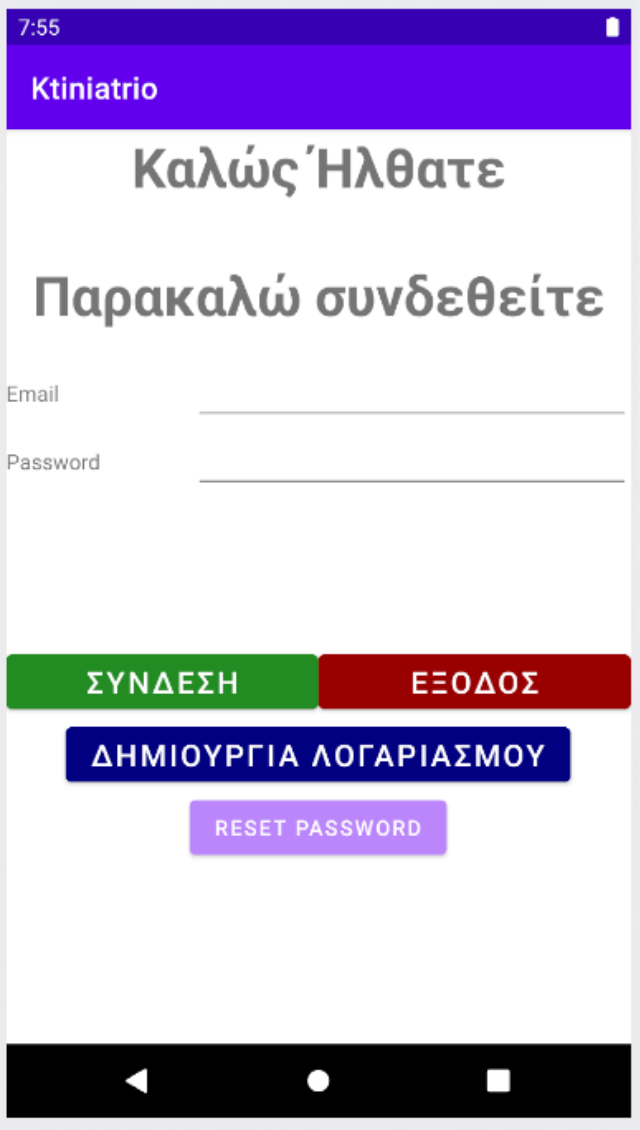
Επίσης υπάρχει και η κλάση Ownerη οποία περιέχει τα στοιχεία των ιδιοκτητών.

3.3. Εγκατάσταση

Για να εγκαταστήσουμε την εφαρμογή στο κινητό μας τηλέφωνο αρκεί να αντιγράψουμε το apkπου υπάρχει στον φάκελο app\build\outputs\apk\debug στο κινητό μας τηλέφωνο. Βέβαια η λύση αυτή είναι έμμεση. Η ενδεδειγμένη λύση είναι να ανεβάσουμε την εφαρμογή στο applicationτης Google.

3.4 Χρήση της εφαρμογής

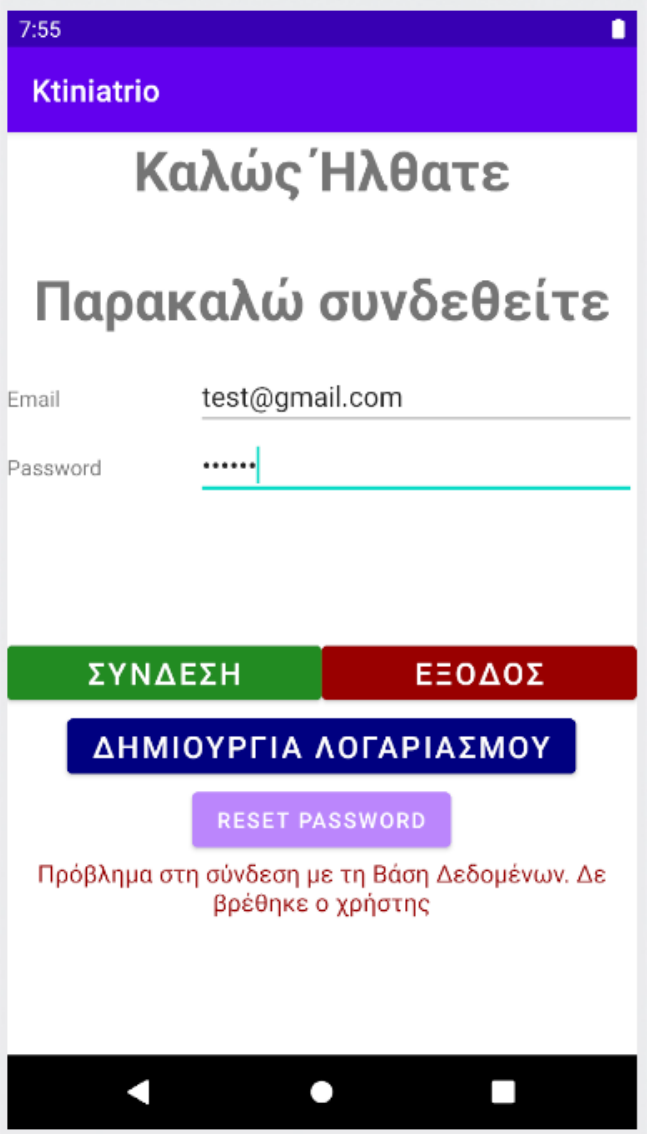
Η αρχική οθόνη της εφαρμογής είναι η ακόλουθη όπου θα πρέπει να εισάγουμε το email χρήστη και τον κωδικό πρόσβασης ενός εγγεγραμμένου χρήστη.



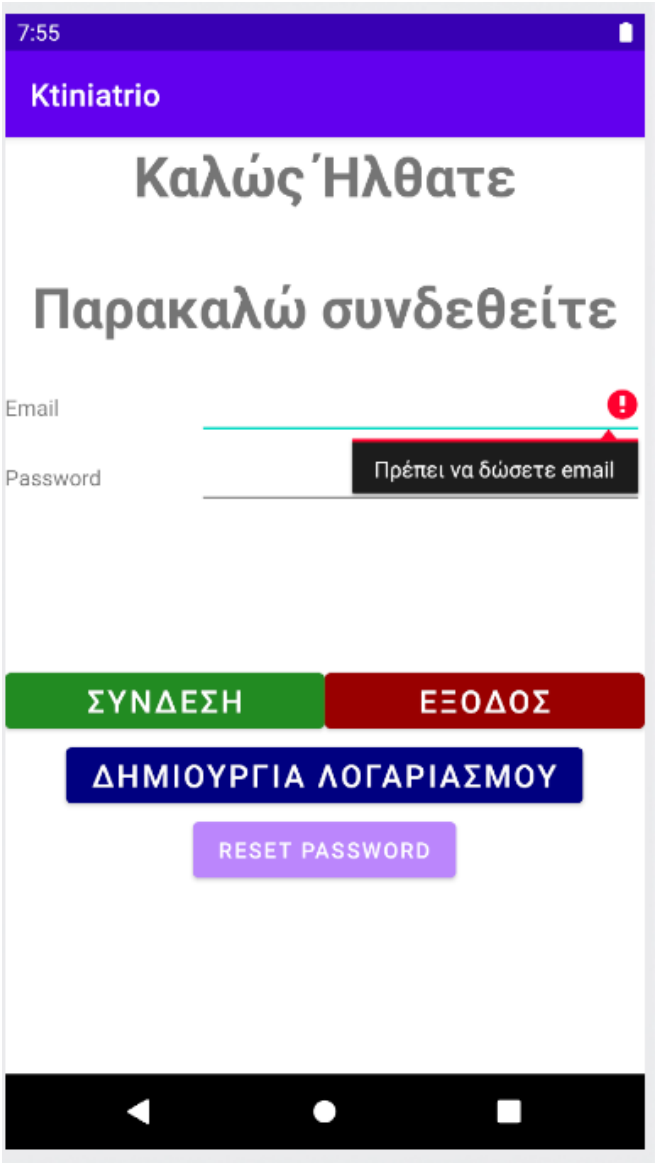
Όπως μπορούμε να δούμε έχουμε τέσσερις επιλογές:

- Σύνδεση
- Έξοδος: για τερματισμό της εφαρμογής
- Δημιουργία λογαριασμού: για εγγραφή της εφαρμογής δίνοντας το emailκαι το passwordκαι
- Resetpassword: για αποστολή ενός emailγια να γίνει resetto password.

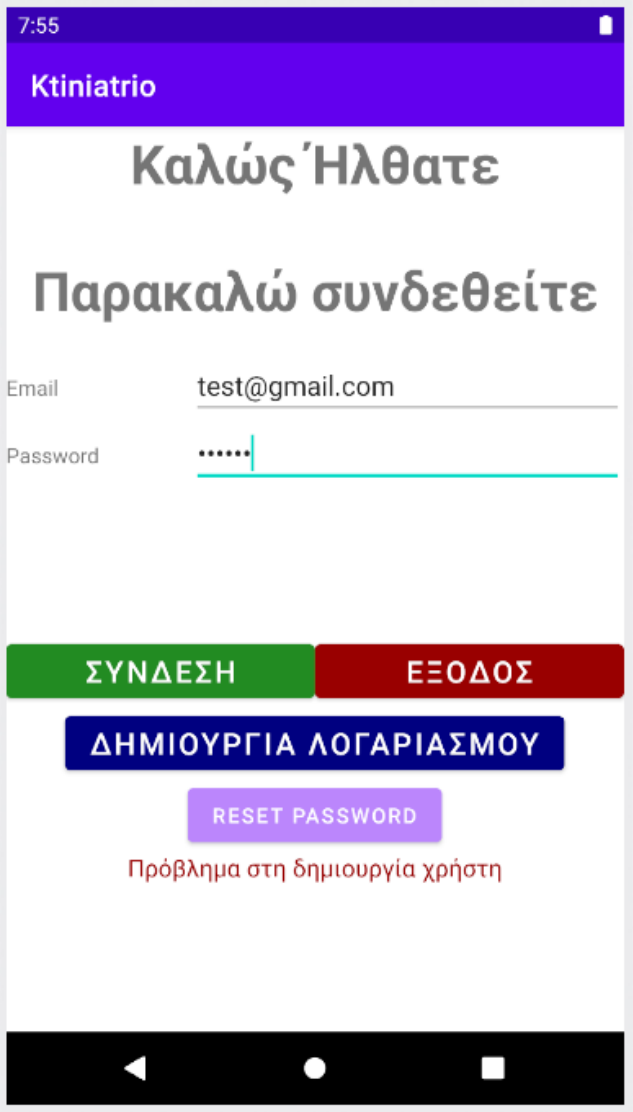
Αν εισάγουμε emailκαι passwordτα οποία δεν υπάρχουν στη βάση δεδομένων και προσπαθήσουμε να συνδεθούμε, τότε θα εμφανιστεί μήνυμα:



Αν προσπαθήσουμε να εγγραφούμε στο σύστημα χωρίς να δώσουμε emailή password τότε θα εμφανιστεί κατάλληλο μήνυμα στο αντίστοιχο πεδίο κειμένου που θα μας ενημερώνει για το λάθος που έχουμε κάνει:



Αν υπάρχει ήδη ο χρήστης ή υπάρξει κάποιο πρόβλημα με την επικοινωνία με τη βάση δεδομένων, παρότι έχουμε συμπληρώσει σωστά τα στοιχεία (email, password) και έχουμε πατήσει το Δημιουργία λογαριασμού θα δούμε κατάλληλο μήνυμα λάθους:



Αν πατήσουμε το ResetPasswordγιατί έχουμε ξεχάσει το passwordτότε η Firebaseθα στείλει ένα emailγια οδηγίες για το πώς να κάνουμε resetto password. Το passwordμπορεί να το κάνει resetκαι ο διαχειριστής της βάσης δεδομένων ή ο διαχειριστής μπορεί να σβήσει και τον χρήστη.

Αντίστοιχο μήνυμα λάθους θα εμφανιστεί αν προσπαθήσουμε να συνδεθούμε με λανθασμένα στοιχεία.

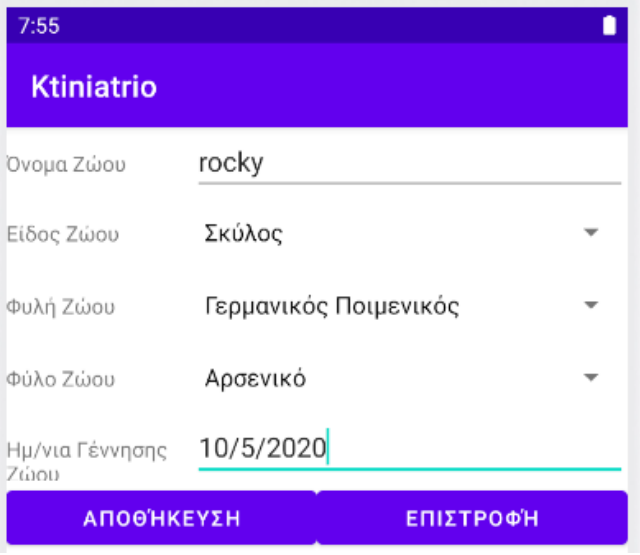
Αν τα στοιχεία που εισάγουμε κατά τη σύνδεσή μας είναι σωστά ή εισάγουμε σωστά στοιχεία κατά τη Δημιουργία λογαριασμού θα μεταφερθούμε στην κεντρική οθόνη της εφαρμογής μας όπου υπάρχει όλη η λειτουργικότητα.

Όπως μπορούμε να δούμε μπορούμε να εισάγουμε τα προσωπικά μας στοιχεία και να τα αποθηκεύσουμε. Μπορούμε να προσθέσουμε κάποιο ζώο που μας ανήκει και τότε θα εμφανιστεί στο dropdownlistδίπλα από τα buttonεμβόλια και θεραπεία. Επίσης μπορούμε να βγούμε από την εφαρμογή.

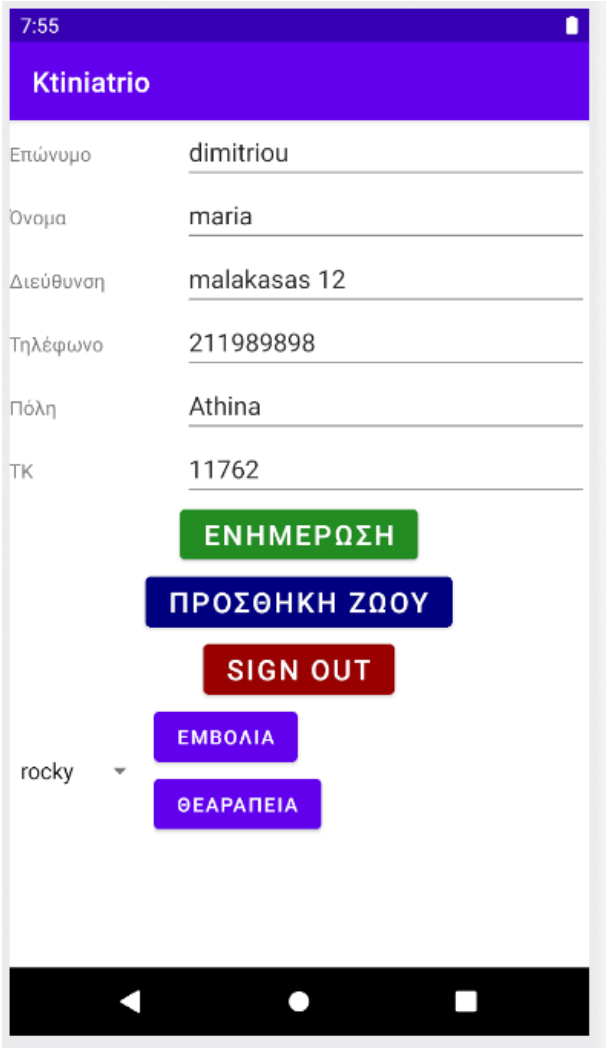
Ας δοκιμάσουμε αρχικά να πατήσουμε το Αποθήκευση χωρίς να έχουμε εισάγει στοιχεία. Τότε με βάση τους ελέγχους που κάνουμε θα εμφανιστεί μήνυμα λάθους πρώτα για το όνομα, μετά για το επώνυμο κ.λπ.

Αν εισάγουμε τα προσωπικά μας στοιχεία και αποσυνδεθούμε, το σύστημα θα τα έχει αποθηκεύσει στη βάση δεδομένων. Έτσι την επόμενη φορά που θα συνδεθούμε στην εφαρμογή θα δούμε την ακόλουθη εικόνα:

Αν πατήσουμε το προσθήκη ζώου θα εμφανιστεί η ακόλουθη οθόνη όπου πρέπει να εισάγουμε τα στοιχεία του ζώου:

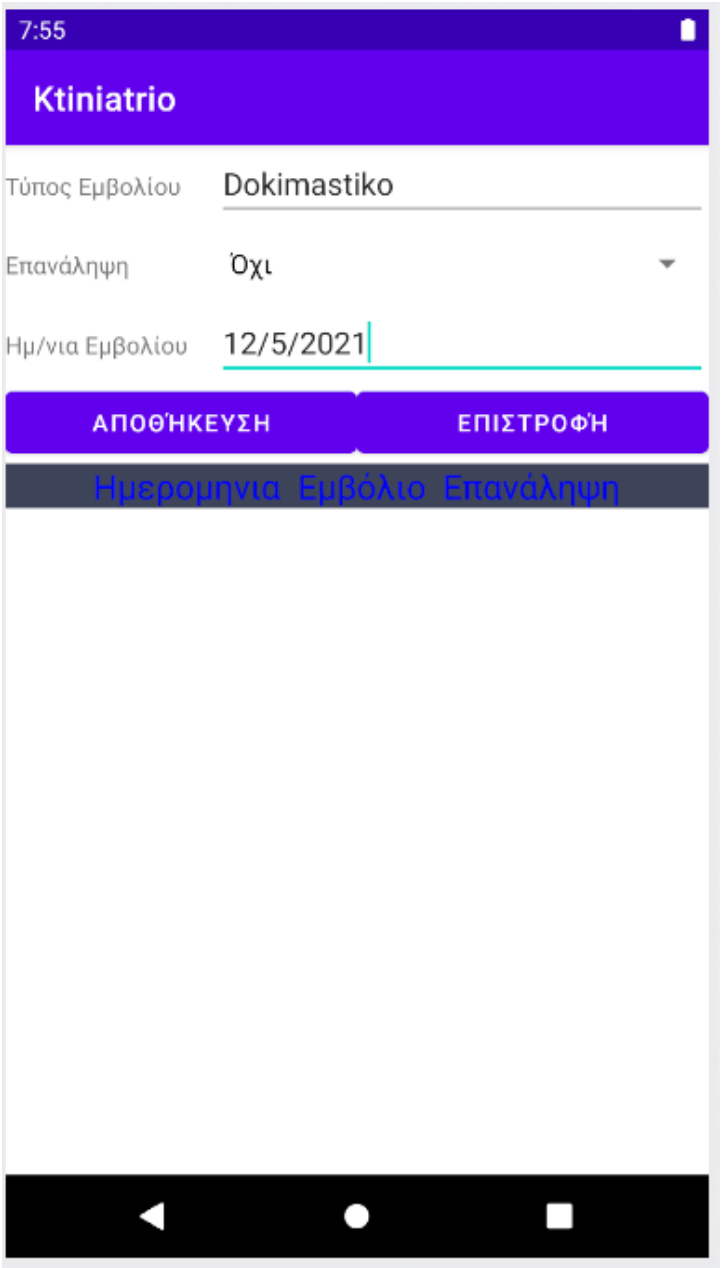


Αν πατήσουμε το Αποθήκευση θα αποθηκευτούν τα στοιχεία του ζώου στη βάση δεδομένων, διαφορετικά δε θα αποθηκευτούν και θα επιστρέψουμε στην κεντρική οθόνη. Έχοντας αποθηκεύσει το σκύλο στη βάση δεδομένων, επιστρέφουμε άμεσα στην κεντρική οθόνη της εφαρμογής μας όπου βλέπουμε την ακόλουθη εικόνα:

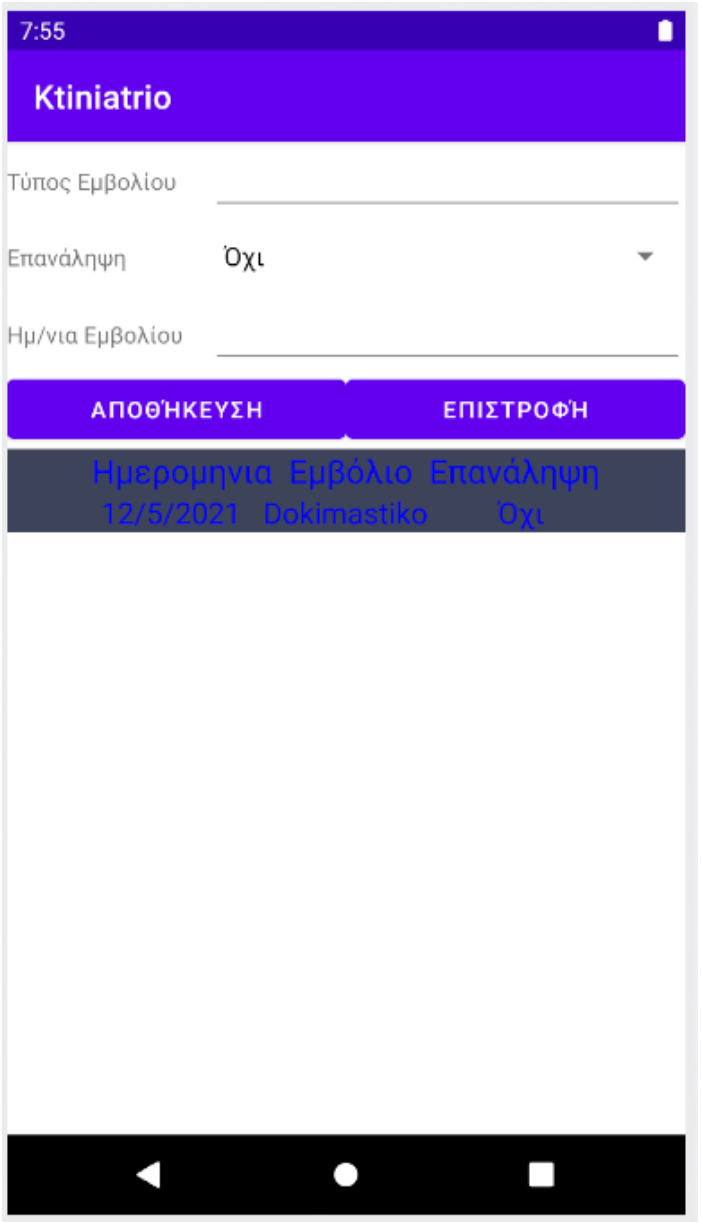


Επιλέγοντας το ζώο που θέλουμε από το dropdownlistκαι πατώντας το Εμβόλια θα μεταφερθούμε στην οθόνη όπου μπορούμε να δούμε τα εμβόλια που έχει ήδη κάνει το ζώο και να προσθέσουμε νέα.

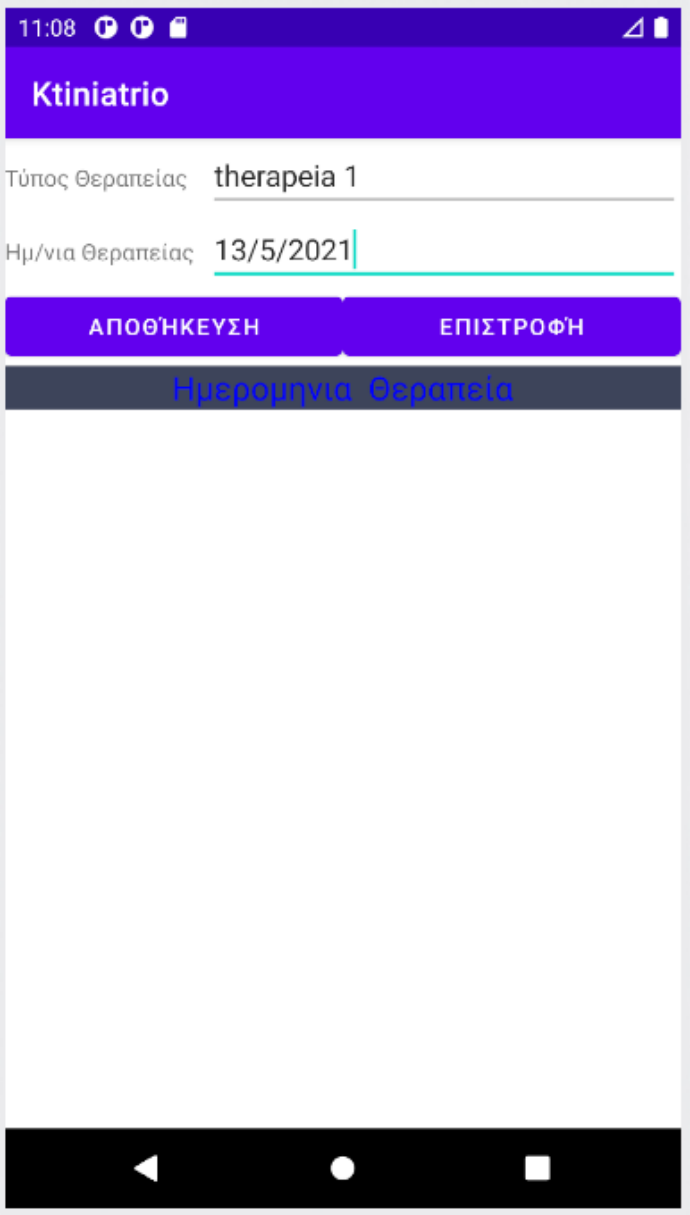
Όπως μπορούμε να δούμε στην εικόνα που ακολουθεί δεν υπάρχει κανένα εμβόλιο στο ιστορικό του, επομένως προσθέτουμε εμείς το πρώτο.



Αν πατήσουμε το αποθήκευση θα κλείσει η συγκεκριμένη οθόνη εισαγωγής εμβολίων και θα επιστρέψουμε στην κεντρική οθόνη. Αν πατήσουμε ξανά εμβόλια θα δούμε να υπάρχει ήδη το εμβόλιο που έχουμε εισάγει στο ιστορικό του ζώου που έχουμε επιλέξει:

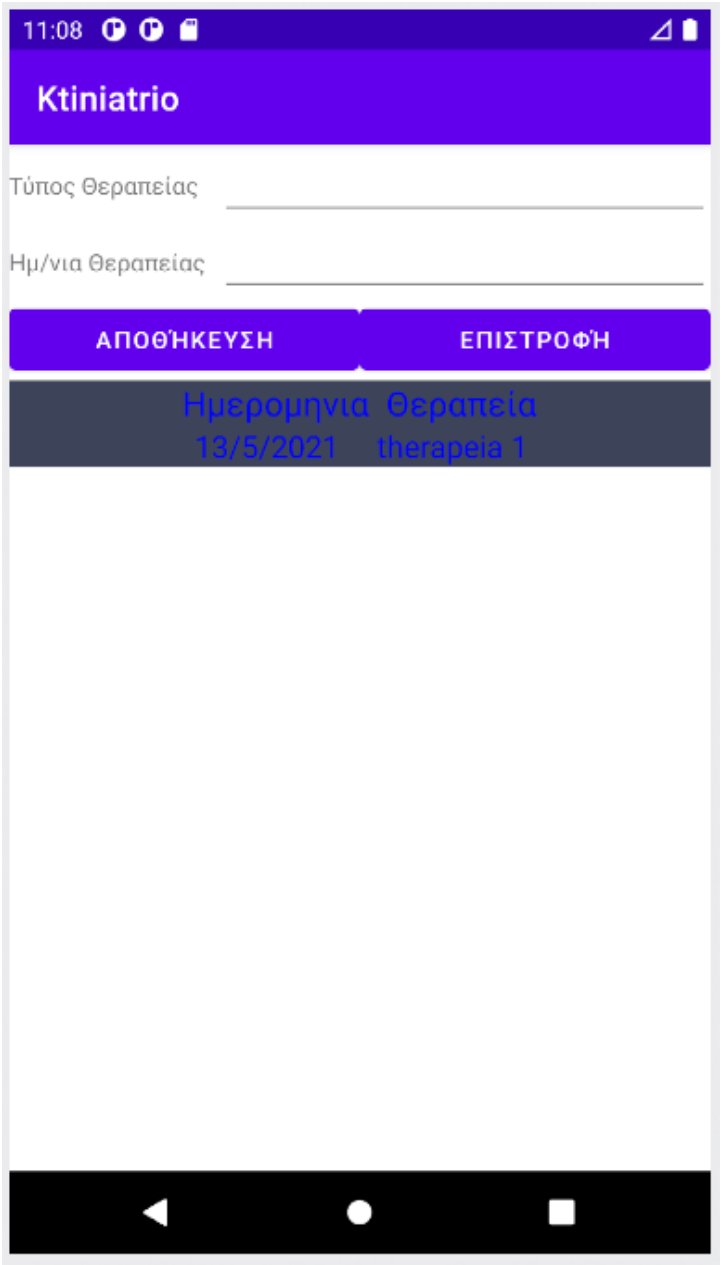


Η τελευταία μας επιλογή είναι οι Θεραπείες. Επιλέγουμε πάλι το ζώο για το οποίο θέλουμε να εισάγουμε τη Θεραπεία και πατάμε το Θεραπεία. Θα εμφανιστεί η ακόλουθη οθόνη στο κινητό μας όπου μπορούμε να εισάγουμε το όνομα της Θεραπείας και την ημερομηνία της. Στο κάτω μέρος της οθόνης βλέπουμε το ιστορικό των θεραπειών για το ζώο που υπάρχει στη βάση δεδομένων. Όπως είναι λογικό, δεν υπάρχει καμία θεραπεία και η λίστα είναι κενή.



Αν εισάγουμε τα παραπάνω στοιχεία και πατήσουμε το Αποθήκευση, τότε θα αποθηκευτεί η θεραπεία στη βάση δεδομένων και θα επιστρέψουμε στην κεντρική οθόνη της εφαρμογής μας. Αν πατήσουμε το Επιστροφή τότε θα επιστρέψουμε στην κεντρική οθόνη χωρίς να γίνει καμία αποθήκευση των δεδομένων.

Αν επιλέξουμε πάλι Θεραπεία για το ίδιο ζώο θα μεταφερθούμε στην οθόνη των Θεραπειών όπου θα δούμε την θεραπεία που έχουμε εισάγει να υπάρχει στο ιστορικό του ζώου.


11:08
Ktiniatrio
Τύπος Θεραπείας
Ημ/νια Θεραπείας
ΑΠΟΘΗΚΕΥΣΗ
ΕΠΙΣΤΡΟΦΗ
Ημερομηνια Θεραπεία
13/5/2021 therapeia 1

ΚΕΦΑΛΑΙΟ 4. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΤΑΣΕΙΣ

4.1 Συμπεράσματα

Στην πτυχιακή αυτή μελετήσαμε τα έξυπνα κινητά τηλέφωνα και τις εφαρμογές τους. Εστιάσαμε στις εφαρμογές που αφορούν κτηνιατρεία και προσφέρουν στους χρήστες υπηρεσίες για τα ζώα που έχουν.

Στη συνέχεια μελετήσαμε τα διαθέσιμα εργαλεία για την υλοποίηση της εφαρμογής και καταλήξαμε στη χρήση της Firebase σα βάσης δεδομένων γιατί παρέχει πρόσβαση μέσω του διαδικτύου για όλους τους χρήστες. Επίσης χρησιμοποιήσαμε το AndroidStudioγια την υλοποίηση της εφαρμογής.

Τέλος σχεδιάσαμε την εφαρμογή μας με βάση UserStorieskαι Wireframes. Προχωρήσαμε στην υλοποίηση, μια αρκετά πολύπλοκη και χρονοβόρα διαδικασία.

Ολοκληρώνοντας την εφαρμογή διαπιστώνουμε ότι η ανάπτυξη εφαρμογών για έξυπνα κινητά είναι μια αρκετά πολύπλοκη διαδικασία η οποία όμως μπορεί να απλοποιηθεί αν σχεδιαστεί σωστά.

4.2 Προτάσεις

Οι προτάσεις για μελλοντικές επεκτάσεις της εφαρμογής είναι:

- Προσθήκη γραφικών – εικόνων για να γίνει πιο ελκυστική.
- Προσθήκη δυνατότητας εγγραφής με Gmail.
- Προσθήκη νέων λειτουργιών όπως καταχώρηση ειδοποιήσεων για υπενθυμίσεις εμβολιασμών.
- Διαχωρισμός χρηστών σε κατηγορίες, όπως κτηνίατροι και ιδιοκτήτες ζώων. Με τον τρόπο αυτό θα μπορούν οι ιδιοκτήτες να αναζητήσουν κτηνίατρο, να τον βαθμολογήσουν κ.λπ.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Ξενόγλωσση

AltexSoft (2019), The Good and the Bad of Firebase Backend Services, Διαθέσιμοαπόhttps://www.altexsoft.com/blog/firebase-review-pros-cons-alternatives/

Batschinski (n.d.), What is Firebase? All secrets unlocked, Διαθέσιμοαπόhttps://blog.back4app.com/firebase/

German K. (2013), A decade (or so) of BlackBerry smartphones, Διαθέσιμοαπόhttps://www.cnet.com/pictures/a-decade-or-so-of-blackberry-smartphones-pictures/

Hamed T., DaraR. &Kremer S.C. (2017)Intrusion Detection in Contemporary Environments, Διαθέσιμοαπόhttps://www.sciencedirect.com/topics/computer-science/mobile-operating-system

Jackson K. (2018), A brief history of the smartphone, Διαθέσιμοαπόhttps://sciencenode.org/feature/How%20did%20smartphones%20evolve.php

Merck Sharp & Dohme Corp (2020), MSD Vet Manual, Διαθέσιμοαπόhttps://play.google.com/store/apps/details?id=com.msd.veterinary&hl=el&gl=US

Mullis A. (2017), Android Studio tutorial for beginners, Διαθέσιμοαπόhttps://www.androidauthority.com/android-studio-tutorial-beginners-637572/

Naveen (2021), iOS Architecture, Διαθέσιμοαπόhttps://intellipaat.com/blog/tutorial/ios-tutorial/ios-architecture/

Praveenruhil (2021), Android Architecture, Διαθέσιμοαπόhttps://www.geeksforgeeks.org/android-architecture/

Rehkopf M. (n.d.) User Stories with Examples and Template, Διαθέσιμοαπόhttps://www.atlassian.com/agile/project-management/user-stories

Sen A. (2015), Investigation on Trends of Mobile Operating Systems, Διαθέσιμοαπόhttps://www.researchgate.net/publication/280310649_Investigation_on_Trends_of_Mobile_Operating_Systems

Statcounter (2021), Mobile Operating System Market Share Worldwide, Διαθέσιμοαπόhttps://gs.statcounter.com/os-market-share/mobile/worldwide

Stevenson D. (2018), What is Firebase? The complete story, abridged., Διαθέσιμοαπόhttps://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0

Tocci M. (n.d.), History and Evolution of Smartphones, Διαθέσιμοαπόhttps://simpletexting.com/where-have-we-come-since-the-first-smartphone/

USwitch, (2021), History of mobile phones and the first mobile phone, Διαθέσιμοαπόhttps://www.uswitch.com/mobiles/guides/history-of-mobile-phones/

VetApps(2019)Vet Calculator, Διαθέσιμοαπόhttps://play.google.com/store/apps/details?id=com.VetApps.VetCalc&hl=en_US&gl=US

VitusVet(2020), VitusVet: Pet Health Care App, Διαθέσιμοαπόhttps://play.google.com/store/apps/details?id=com.vitusvet.android&hl=en&gl=US

ΠΑΡΑΡΤΗΜΑ – ΚΩΔΙΚΑΣ ΕΦΑΡΜΟΓΗΣ

MainActivity

```
package com.example.ktiniatrio;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.example.ktiniatrio.R.color;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import org.w3c.dom.Text;

import java.util.HashMap;
import java.util.Map;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    private FirebaseAuth mAuth;
    EditText emailText, passwordText;
    TextView resultText;
    Button btnReset;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        findViewById(R.id.sign_in_button).setOnClickListener(this);
        findViewById(R.id.signOutButton).setOnClickListener(this);
        findViewById(R.id.createButton).setOnClickListener(this);
        btnReset = (Button) findViewById(R.id.resetButton);
        emailText=findViewById(R.id.emailText);
        resultText=(TextView)findViewById((R.id.resultText));
        passwordText=findViewById(R.id.passwordText);
```

```
        mAuth = FirebaseAuth.getInstance();
        FirebaseDatabase database = FirebaseDatabase.getInstance("https://ktiniatrio-36de0-
default-rtdb.europe-west1.firebasedatabase.app/");
        DatabaseReference ref = database.getReference();
        DatabaseReference myRef , usersRef;

        database = FirebaseDatabase.getInstance("https://ktiniatrio-36de0-default-rtdb.europe-
west1.firebasedatabase.app/");
        myRef = database.getReference();
        usersRef = myRef.child("Cats");
        usersRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot snapshot) {
                Log.e("Count " ,""+snapshot.getChildrenCount());
                for (DataSnapshot postSnapshot: snapshot.getChildren()) {

                    Log.e("Get Data", postSnapshot.toString());
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Log.e("The read failed: " ,"ERROR!!!!!");
            }

        });

        btnReset.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mAuth.sendPasswordResetEmail(emailText.getText().toString())
.addOnCompleteListener(new OnCompleteListener<Void>() {
                            @Override
                            public void onComplete(@NonNull Task<Void> task) {
                                if (task.isSuccessful()) {
                                    Toast.makeText(MainActivity.this, "We have sent you instructions to
reset your password!", Toast.LENGTH_SHORT).show();
                                } else {
                                    Toast.makeText(MainActivity.this, "Failed to send reset email!",
Toast.LENGTH_SHORT).show();
                                }

                            }

                        });
            }
        });
    }

    public void signIn(){
        String strUserName = emailText.getText().toString();
        if(TextUtils.isEmpty(strUserName)) {
            emailText.setError("Πρέπει να δώσετε email");
            resultText.setText("Πρέπει να δώσετε email");
            return;
```

```
        }
        strUserName = passwordText.getText().toString();
        if (TextUtils.isEmpty(strUserName)) {
            passwordText.setError(("Δε δώσατε password"));
            return;
        }

        mAuth.signInWithEmailAndPassword(emailText.getText().toString(),
passwordText.getText().toString())
.addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {


                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()) {
                        // Sign in success, update UI with the signed-in user's information
                        Toast.makeText(MainActivity.this, "Authentication succeeded.",
                                Toast.LENGTH_SHORT).show();
                        FirebaseUser user = mAuth.getCurrentUser();
                        Intent myIntent = new Intent(MainActivity.this, FirstPageActivity.class);
                        myIntent.putExtra("key", emailText.getText().toString()); //Optional
parameters
                        FirebaseAuth.getInstance().signOut();
MainActivity.this.startActivity(myIntent);
finish();
                    } else {
                        // If sign in fails, display a message to the user.
                        Log.w("-------->", "signInWithEmail:failure", task.getException());
                        resultText.setText("Πρόβλημα στη σύνδεση με τη Βάση Δεδομένων. Δε
βρέθηκε ο χρήστης");
                        Toast.makeText(MainActivity.this, "Authentication failed.",
                                Toast.LENGTH_SHORT).show();
                    }
                }
            });
    }

    public void signOut(){
        //FirebaseAuth.getInstance().signOut();
finish();
    }

    public void createUser(){
        String username=emailText.getText().toString();
        String strUserName = emailText.getText().toString();
        if(TextUtils.isEmpty(strUserName)) {
            emailText.setError("Πρέπει να δώσετε email");
            resultText.setText("");
            return;
        }
        strUserName = passwordText.getText().toString();
        if (TextUtils.isEmpty(strUserName)) {
            passwordText.setError(("Δε δώσατε password"));
            resultText.setText("");
            return;
```

```
        }
        mAuth.createUserWithEmailAndPassword(emailText.getText().toString(),
passwordText.getText().toString())
.addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()) {
                        // Sign in success, update UI with the signed-in user's information
                        Log.d("-------->", "createUserWithEmail:success");
                        FirebaseUser user = mAuth.getCurrentUser();
                        Toast.makeText(MainActivity.this, "User
Created."+user.getDisplayName(),
                                Toast.LENGTH_SHORT).show();
                        Intent myIntent = new Intent(MainActivity.this, FirstPageActivity.class);
                        myIntent.putExtra("key", username); //Optional parameters
                        FirebaseAuth.getInstance().signOut();
MainActivity.this.startActivity(myIntent);
                    } else {
                        // If sign in fails, display a message to the user.
                        Log.w("-------->", "createUserWithEmail:failure", task.getException());
                        resultText.setText("Πρόβλημα στη δημιουργία χρήστη");
                        Toast.makeText(MainActivity.this, "Authentication failed.",
                                Toast.LENGTH_SHORT).show();
                    }
                }
            });
    }

    @Override
    public void onClick(View v) {
        int i = v.getId();
        if (i == R.id.sign_in_button) {
signIn();
        }
        else if (i == R.id.signOutButton) {
signOut();
        }
        else if (i == R.id.createButton) {
createUser();
        }
    }
}
```

FirstPageActivity

```
package com.example.ktiniatrio;

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

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
```

```
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;
import com.google.firebase.database.ValueEventListener;

import org.w3c.dom.Text;

import java.util.HashMap;
import java.util.Map;

public class FirstPageActivity extends AppCompatActivity {

    EditText surnameText, nameText, addressText, phoneText, cityText,tkText;
    private FirebaseAuth auth;
    private Button btnSignOut, saveUpdateButton,animalButton, embolioButton,
therapeiaButton;
    FirebaseDatabase database;
    DatabaseReference myRef, usersRef;
    String value;
    Spinner AnimalSpinner;
String[] animalsItems;
    ArrayAdapter<String> animalAdapter;

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

        btnSignOut = (Button) findViewById((R.id.signOutButton));
        animalButton = (Button) findViewById((R.id.animalButton));
        saveUpdateButton=(Button)findViewById(R.id.saveUpdateButton);
        embolioButton=(Button)findViewById(R.id.embolioButton);
        therapeiaButton=(Button)findViewById(R.id.therapeiaButton);
        surnameText=(EditText)findViewById((R.id.surnameText));
        nameText=(EditText)findViewById((R.id.nameText));
        phoneText=(EditText)findViewById((R.id.phoneText));
        addressText=(EditText)findViewById((R.id.addressText));
        cityText=(EditText)findViewById((R.id.cityText));
        tkText=(EditText)findViewById((R.id.tkText));
        AnimalSpinner=(Spinner)findViewById(R.id.AnimalSpinner);

        Intent intent = getIntent();
        value = intent.getStringExtra("key");
```

```
        FirebaseAuth.getInstance();
        database = FirebaseDatabase.getInstance("https://ktiniatrio-36de0-default-rtdb.europe-
west1.firebasedatabase.app/");
        myRef = database.getReference();
        usersRef = myRef.child("users");


        usersRef.child(value.replace(".","_")).get().addOnCompleteListener(new
OnCompleteListener<DataSnapshot>() {
            @Override
            public void onComplete(@NonNull Task<DataSnapshot> task) {
                if (!task.isSuccessful()) {
                    Log.d("++++firebase++++++", "Error getting data", task.getException());

                }
                else {
                    Map<String,String> td=(HashMap<String, String>)task.getResult().getValue();
                    if (td==null){
                        saveUpdateButton.setText("ΑΠΟΘΗΚΕΥΣΗ");
                    }
                    else {
                        saveUpdateButton.setText("ΕΝΗΜΕΡΩΣΗ");
                        surnameText.setText(td.get("surname"));
                        nameText.setText(td.get("name"));
                        addressText.setText(td.get("address"));
                        phoneText.setText(td.get("phone"));
                        cityText.setText(td.get("city"));
                        tkText.setText(td.get("tk"));
                    }
                }
            }
        });

        usersRef = myRef.child("Animals").child(value.replace(".","_"));
        usersRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot snapshot) {

                animalsItems = new String[(int)snapshot.getChildrenCount()];
                int i=0;
                for (DataSnapshot postSnapshot: snapshot.getChildren()) {
                    Map<String,String> td=(HashMap<String, String>)postSnapshot.getValue();
                    Log.d("------------------------>",postSnapshot.getValue().toString());
                    animalsItems[i]=td.get("name");
                    i++;
                }
                ok1();
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Log.e("The read failed: " ,"ERROR!!!!!");
            }

        });
```

```
        btnSignOut.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                FirebaseAuth.getInstance().signOut();
finish();
            }
        });

        animalButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent myIntent = new Intent(FirstPageActivity.this, AnimalActivity.class);
                myIntent.putExtra("key", value); //Optional parameters
                FirebaseAuth.getInstance().signOut();
FirstPageActivity.this.startActivity(myIntent);
            }
        });

        embolioButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent myIntent = new Intent(FirstPageActivity.this, EmvolioActivity.class);
                myIntent.putExtra("key", value); //Optional parameters
                myIntent.putExtra("animal", AnimalSpinner.getSelectedItem().toString());
//Optional parameters
                FirebaseAuth.getInstance().signOut();
FirstPageActivity.this.startActivity(myIntent);
            }
        });

        therapeiaButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent myIntent = new Intent(FirstPageActivity.this, TherapeiaActivity.class);
                myIntent.putExtra("key", value); //Optional parameters
                myIntent.putExtra("animal", AnimalSpinner.getSelectedItem().toString());
//Optional parameters
                FirebaseAuth.getInstance().signOut();
FirstPageActivity.this.startActivity(myIntent);
            }
        });

        saveUpdateButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String strUserName=nameText.getText().toString();
                if(TextUtils.isEmpty(strUserName)) {
                    nameText.setError("Πρέπει να δώσετε όνομα");
                    return;
                }
                strUserName=surnameText.getText().toString();
                if(TextUtils.isEmpty(strUserName)) {
                    surnameText.setError("Πρέπει να δώσετε επώνυμο");
                    return;
```

```
            }
            strUserName=addressText.getText().toString();
            if(TextUtils.isEmpty(strUserName)) {
               addressText.setError("Πρέπει να δώσετε διεύθυνση");
               return;
            }
            strUserName=phoneText.getText().toString();
            if(TextUtils.isEmpty(strUserName)) {
               phoneText.setError("Πρέπει να δώσετε τηλέφωνο");
               return;
            }
            strUserName=cityText.getText().toString();
            if(TextUtils.isEmpty(strUserName)) {
               cityText.setError("Πρέπει να δώσετε πόλη");
               return;
            }
            strUserName=tkText.getText().toString();
            if(TextUtils.isEmpty(strUserName)) {
               tkText.setError("Πρέπει να δώσετε ΤΚ");
               return;
            }
            if (saveUpdateButton.getText().equals("ΑΠΟΘΗΚΕΥΣΗ")){
               Owner owner=new
Owner(nameText.getText().toString(),surnameText.getText().toString(),addressText.getText(
).toString(),
phoneText.getText().toString(),cityText.getText().toString(),tkText.getText().toString());
               Map<String, Object> users = new HashMap<>();
               usersRef = myRef.child("users");
               users.put(value.replace(".","_"),owner);
               usersRef.updateChildren(users);
               usersRef.push();
            }
else{
               DatabaseReference hopperRef = usersRef.child(value.replace(".","_"));
               Map<String, Object> hopperUpdates = new HashMap<>();
hopperUpdates.put("name", nameText.getText().toString());
hopperUpdates.put("surname", surnameText.getText().toString());
hopperUpdates.put("address", addressText.getText().toString());
hopperUpdates.put("city", phoneText.getText().toString());
hopperUpdates.put("tk", tkText.getText().toString());
hopperUpdates.put("phone", phoneText.getText().toString());
               hopperRef.updateChildren(hopperUpdates);
            }
         }
      });
   }
   private void ok1(){
      animalAdapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_dropdown_item, animalsItems);
      AnimalSpinner.setAdapter(animalAdapter);
   }
}
```

AnimalActivity

```
package com.example.ktiniatrio;

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

import android.app.DatePickerDialog;
import android.content.Intent;
import android.icu.util.Calendar;
import android.icu.util.TimeZone;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.Spinner;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.FirebaseError;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.HashMap;
import java.util.Map;

public class AnimalActivity extends AppCompatActivity {
    String value;
    EditText animalName,editTextDate;
    Spinner SpeciesSpinner, FiliSpinner, FiloSpinner;
    FirebaseDatabase database;
    DatabaseReference myRef, usersRef;
    ArrayAdapter<String> caTsAdapter, dogsAdapter;
String[] catsItems,dogsItems;
    Button saveButton, returnButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_animal);
        Intent intent = getIntent();
        value = intent.getStringExtra("key");
        animalName=(EditText)findViewById((R.id.animalName));
        editTextDate=(EditText)findViewById((R.id.editTextDate));
        SpeciesSpinner=(Spinner)findViewById((R.id.SpeciesSpinner));
        FiliSpinner=(Spinner)findViewById((R.id.FiliSpinner));
        FiloSpinner=(Spinner)findViewById((R.id.FiloSpinner));
        saveButton=(Button)findViewById((R.id.saveButton));
```

```
        returnButton=(Button)findViewById((R.id.returnButton));

        editTextDate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
callDate();
            }
        });
        saveButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
saveAnimal();
            }
        });
        returnButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
finish();
            }
        });
String[] filoItems = new String[]{"Αρσενικό", "Θηλυκό"};
        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_dropdown_item, filoItems);
        FiloSpinner.setAdapter(adapter);
String[] AnimalItems = new String[]{"Σκύλος", "Γάτα"};
        adapter = new ArrayAdapter<>(this, android.R.layout.simple_spinner_dropdown_item,
AnimalItems);
        SpeciesSpinner.setAdapter(adapter);
        SpeciesSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener()
{

            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id)
{
                if (position==0) {
                    Log.d("----------------", "Σκύλος");
                    FiliSpinner.setAdapter(dogsAdapter);
                }
                else {
                    Log.d("----------------", "Γάτα");
                    FiliSpinner.setAdapter(caTsAdapter);
                }
            }

            public void onNothingSelected(AdapterView<?> adapterView) {
                return;
            }
        });
        FirebaseAuth.getInstance();
        database = FirebaseDatabase.getInstance("https://ktiniatrio-36de0-default-rtdb.europe-
west1.firebasedatabase.app/");
        myRef = database.getReference();
        usersRef = myRef.child("Cats");
        usersRef.addValueEventListener(new ValueEventListener() {
            @Override
```

```
            public void onDataChange(DataSnapshot snapshot) {
                catsItems = new String[(int)snapshot.getChildrenCount()];
                int i=0;
                for (DataSnapshot postSnapshot: snapshot.getChildren()) {

catsItems[i]=postSnapshot.getValue().toString().substring(postSnapshot.getValue().toString()
.indexOf("=")+1,postSnapshot.getValue().toString().length()-1);
                    i++;
                }
                ok1();
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Log.e("The read failed: " ,"ERROR!!!!!");
            }

        });
        usersRef = myRef.child("Dogs");
        usersRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot snapshot) {
                dogsItems = new String[(int)snapshot.getChildrenCount()];
                int i=0;
                for (DataSnapshot postSnapshot: snapshot.getChildren()) {

dogsItems[i]=postSnapshot.getValue().toString().substring(postSnapshot.getValue().toString(
).indexOf("=")+1,postSnapshot.getValue().toString().length()-1);
                    i++;
                }
                ok2();
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                Log.e("The read failed: " ,"ERROR!!!!!");
            }

        });


    }

    private void callDate(){
        DatePickerDialog.OnDateSetListener listener=new
DatePickerDialog.OnDateSetListener() {
            @Override
            public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth)
            {
                editTextDate.setText(dayOfMonth + "/" + monthOfYear + "/" + year);
            }};
        DatePickerDialog dpDialog=new DatePickerDialog(this, listener, 2021, 5, 14);
        dpDialog.show();
    }
    private void ok1(){
```

```
        caTsAdapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_dropdown_item, catsItems);
        FiliSpinner.setAdapter(caTsAdapter);
    }
    private void ok2(){
        dogsAdapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_dropdown_item, dogsItems);
        FiliSpinner.setAdapter(dogsAdapter);
    }

    private void saveAnimal(){
        usersRef = myRef.child("Animals");
        DatabaseReference hopperRef =
usersRef.child(value.replace(".","_")).child(animalName.getText().toString());
        Map<String, Object> hopperUpdates = new HashMap<>();
hopperUpdates.put("name", animalName.getText().toString());
hopperUpdates.put("dob", editTextDate.getText().toString());
hopperUpdates.put("species", SpeciesSpinner.getSelectedItem().toString());
hopperUpdates.put("fili", FiliSpinner.getSelectedItem().toString());
hopperUpdates.put("filo", FiloSpinner.getSelectedItem().toString());
        hopperRef.updateChildren(hopperUpdates);
finish();
    }
}
```

EmvolioActivity

```
package com.example.ktiniatrio;

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

import android.app.DatePickerDialog;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.view.Gravity;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.Spinner;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
```

```
import java.util.HashMap;
import java.util.Map;

public class EmvolioActivity extends AppCompatActivity {
   Spinner EpanalipsiSpinner;
   EditText embolioName,editTextDate;
   Button saveButton, returnButton;
   FirebaseDatabase database;
   DatabaseReference myRef, usersRef;
   String value, animal;
   TableLayout stk;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_emvolio);
      Intent intent = getIntent();
      value = intent.getStringExtra("key");
      animal = intent.getStringExtra("animal");
      EpanalipsiSpinner=(Spinner)findViewById(R.id.EpanalipsiSpinner);
      embolioName=(EditText)findViewById(R.id.embolioName);
      editTextDate=(EditText)findViewById(R.id.editTextDate);
      saveButton=(Button)findViewById((R.id.saveButton));
      returnButton=(Button)findViewById((R.id.returnButton));
String[] epanalipsiItems = new String[]{"Όχι", "Ναι"};
      ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_dropdown_item, epanalipsiItems);
      EpanalipsiSpinner.setAdapter(adapter);
      FirebaseAuth.getInstance();
      database = FirebaseDatabase.getInstance("https://ktiniatrio-36de0-default-rtdb.europe-
west1.firebasedatabase.app/");
      myRef = database.getReference();
      usersRef = myRef.child("Emvolio");
      //-----------------------------------------------
      stk = (TableLayout) findViewById(R.id.table_main);
      TableRow tbrow0 = new TableRow(this);
      TextView tv0 = new TextView(this);
      tv0.setText(" Ημερομηνια ");
      tv0.setTextColor(Color.BLUE);
      tv0.setTextSize(20);
      tbrow0.addView(tv0);
      TextView tv1 = new TextView(this);
      tv1.setText(" Εμβόλιο ");
      tv1.setTextColor(Color.BLUE);
      tv1.setTextSize(20);
      tbrow0.addView(tv1);
      TextView tv2 = new TextView(this);
      tv2.setText(" Επανάληψη ");
      tv2.setTextColor(Color.BLUE);
      tv2.setTextSize(20);
      tbrow0.addView(tv2);
stk.addView(tbrow0);
      usersRef = myRef.child("Emvolio").child(value.replace(".","_")).child(animal);
      usersRef.addValueEventListener(new ValueEventListener() {
         @Override
```

```
            public void onDataChange(DataSnapshot snapshot) {

                //animalsItems = new String[(int)snapshot.getChildrenCount()];
                int i=0;
                for (DataSnapshot postSnapshot: snapshot.getChildren()) {
                    Map<String,String> td=(HashMap<String, String>)postSnapshot.getValue();
                    Log.d("------------------------>",postSnapshot.getValue().toString());
                    ok1(td.get("dob"), td.get("embolio"), td.get("epanalipsi"));
                    //animalsItems[i]=td.get("name");
                    i++;
                }
            }

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

            }
});
        //---------------------------------------------
        editTextDate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
callDate();
            }
        });
        saveButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
saveEmvolio();
            }
        });
        returnButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
finish();
            }
        });
    }
    private void callDate(){
        DatePickerDialog.OnDateSetListener listener=new
DatePickerDialog.OnDateSetListener() {
            @Override
            public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth)
            {
                editTextDate.setText(dayOfMonth + "/" + monthOfYear + "/" + year);
            }};
        DatePickerDialog dpDialog=new DatePickerDialog(this, listener, 2021, 5, 14);
        dpDialog.show();
    }
    private void saveEmvolio(){
        usersRef = myRef.child("Emvolio");
        Log.d("---------->",value.replace(".","_"));
        Log.d("---------->",animal);
        Log.d("---------->",editTextDate.getText().toString().replace("/","_"));
```

```
        DatabaseReference hopperRef =
usersRef.child(value.replace(".","_")).child(animal).child(editTextDate.getText().toString().re
place("/","_"));
        Map<String, Object> hopperUpdates = new HashMap<>();
hopperUpdates.put("name", animal);
hopperUpdates.put("dob", editTextDate.getText().toString());
hopperUpdates.put("epanalipsi", EpanalipsiSpinner.getSelectedItem().toString());
hopperUpdates.put("embolio", embolioName.getText().toString());
        hopperRef.updateChildren(hopperUpdates);
finish();
    }

    private void ok1(String a, String b, String c){
        TableRow tbrow = new TableRow(this);
tbrow.setWeightSum(3);
        TextView t1v = new TextView(this);
        t1v.setText(a);
        t1v.setTextColor(Color.BLUE);
        t1v.setGravity(Gravity.CENTER);
        t1v.setTextSize(18);
        LinearLayout.LayoutParams params = new TableRow.LayoutParams(0,
TableLayout.LayoutParams.WRAP_CONTENT, 1f);
        t1v.setLayoutParams(params);
tbrow.addView(t1v);
        TextView t2v = new TextView(this);
        t2v.setText(b);
        t2v.setTextColor(Color.BLUE);
        t2v.setGravity(Gravity.CENTER);
        t2v.setTextSize(18);
        t2v.setLayoutParams(params);
tbrow.addView(t2v);
        TextView t3v = new TextView(this);
        t3v.setText(c);
        t3v.setTextColor(Color.BLUE);
        t3v.setGravity(Gravity.CENTER);
        t3v.setTextSize(18);
        t3v.setLayoutParams(params);
tbrow.addView(t3v);
stk.addView(tbrow);
    }
}
```

TherapeiaActivity

```
package com.example.ktiniatrio;

import android.app.DatePickerDialog;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.view.Gravity;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
```

```
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;

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

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.HashMap;
import java.util.Map;

public class TherapeiaActivity extends AppCompatActivity {

    EditText therapeiaName,editTextDate;
    Button saveButton, returnButton;
    FirebaseDatabase database;
    DatabaseReference myRef, usersRef;
    String value, animal;
    TableLayout stk;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_therapeia);
        Intent intent = getIntent();
        value = intent.getStringExtra("key");
        animal = intent.getStringExtra("animal");
        therapeiaName=(EditText)findViewById(R.id.therapeiaName);
        editTextDate=(EditText)findViewById(R.id.editTextDate);
        saveButton=(Button)findViewById((R.id.saveButton));
        returnButton=(Button)findViewById((R.id.returnButton));
        FirebaseAuth.getInstance();
        database = FirebaseDatabase.getInstance("https://ktiniatrio-36de0-default-rtdb.europe-
west1.firebasedatabase.app/");
        myRef = database.getReference();
        usersRef = myRef.child("Therapeia");
        //------------------------------------------------
        stk = (TableLayout) findViewById(R.id.table_main);
        TableRow tbrow0 = new TableRow(this);
        TextView tv0 = new TextView(this);
        tv0.setText(" Ημερομηνια ");
        tv0.setTextColor(Color.BLUE);
        tv0.setTextSize(20);
        tbrow0.addView(tv0);
        TextView tv1 = new TextView(this);
        tv1.setText(" Θεραπεία ");
        tv1.setTextColor(Color.BLUE);
```

```
        tv1.setTextSize(20);
        tbrow0.addView(tv1);
stk.addView(tbrow0);
        usersRef = myRef.child("Therapeia").child(value.replace(".","_")).child(animal);
        usersRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot snapshot) {

                //animalsItems = new String[(int)snapshot.getChildrenCount()];
                int i=0;
                for (DataSnapshot postSnapshot: snapshot.getChildren()) {
                    Map<String,String> td=(HashMap<String, String>)postSnapshot.getValue();
                    Log.d("----------------------->",postSnapshot.getValue().toString());
                    ok1(td.get("dob"), td.get("therapeia"));
                    i++;
                }
            }

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

            }
        });
        editTextDate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
callDate();
            }
        });
        saveButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
saveTherapeia();
            }
        });
        returnButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
finish();
            }
        });
    }
    private void callDate(){
        DatePickerDialog.OnDateSetListener listener=new
DatePickerDialog.OnDateSetListener() {
            @Override
            public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth)
            {
                editTextDate.setText(dayOfMonth + "/" + monthOfYear + "/" + year);
            }};
        DatePickerDialog dpDialog=new DatePickerDialog(this, listener, 2021, 5, 14);
        dpDialog.show();
    }
    private void saveTherapeia(){
        usersRef = myRef.child("Therapeia");
```

```
        Log.d("---------->",value.replace(".","_"));
        Log.d("---------->",animal);
        Log.d("---------->",editTextDate.getText().toString().replace("/","_"));
        DatabaseReference hopperRef =
usersRef.child(value.replace(".","_")).child(animal).child(editTextDate.getText().toString().re
place("/","_"));
        Map<String, Object> hopperUpdates = new HashMap<>();
hopperUpdates.put("name", animal);
hopperUpdates.put("dob", editTextDate.getText().toString());
hopperUpdates.put("therapeia", therapeiaName.getText().toString());
        hopperRef.updateChildren(hopperUpdates);
finish();
    }

    private void ok1(String a, String b){
        TableRow tbrow = new TableRow(this);
tbrow.setWeightSum(2);
        TextView t1v = new TextView(this);
        t1v.setText(a);
        t1v.setTextColor(Color.BLUE);
        t1v.setGravity(Gravity.CENTER);
        t1v.setTextSize(18);
        LinearLayout.LayoutParams params = new TableRow.LayoutParams(0,
TableLayout.LayoutParams.WRAP_CONTENT, 1f);
        t1v.setLayoutParams(params);
tbrow.addView(t1v);
        TextView t2v = new TextView(this);
        t2v.setText(b);
        t2v.setTextColor(Color.BLUE);
        t2v.setGravity(Gravity.CENTER);
        t2v.setTextSize(18);
        t2v.setLayoutParams(params);
tbrow.addView(t2v);
stk.addView(tbrow);
    }
}
```

Owner

```
package com.example.ktiniatrio;


public class Owner {
    public String name;
    public String surname;
    public String address;
    public String phone;
    public String city;
    public String tk;

    public Owner(){

    }
```

```
    public Owner(String name, String surname, String address, String phone, String city, String
tk){
        this.name=name;
this.surname=surname;
this.address=address;
this.phone=phone;
this.city=city;
        this.tk=tk;
    }
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
this.surname = surname;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
this.address = address;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
this.phone = phone;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
this.city = city;
    }

    public String getTk() {
        return tk;
    }

    public void setTk(String tk) {
        this.tk = tk;
```

```
    }
}
```