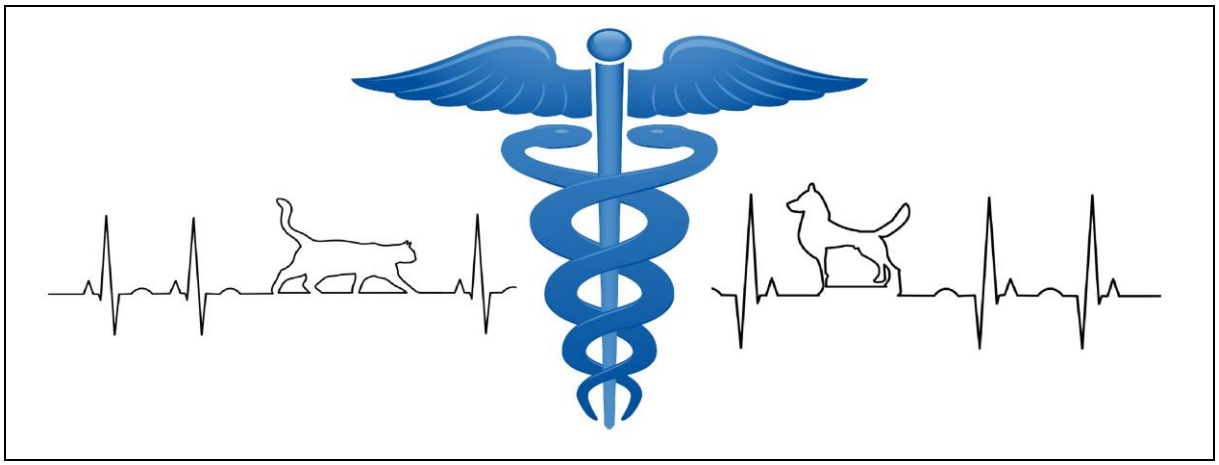
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Παρακολούθηση της υγείας των κατοικίδιων και επικοινωνία της εξαφάνισης τους. Εφαρμογή σε συστήματα Android



Του φοιτητή
Μπάρκου Γεωργίου
Αρ. Μητρώου: 154503

Επιβλέπων
Αδαμίδης Παναγιώτης
Βαθμίδα: Καθηγητής

Ημερομηνία 20-01-2023

Παρακολούθηση της υγείας των κατοικίδιων και επικοινωνία της εξαφάνισης τους. Εφαρμογή σε συστήματα Android

Κωδικός Δ.Ε. 21340

Ονοματεπώνυμο φοιτητή/τών Μπάρκος Γεώργιος

Ονοματεπώνυμο εισηγητή Αδαμίδης Παναγιώτης

Ημερομηνία ανάληψης Δ.Ε. 12-10-2021

Ημερομηνία περάτωσης Δ.Ε. 20-01-2023

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

«Another happy landing»

Πρόλογος

Η παρούσα διπλωματική εργασία επιλέχθηκε με σκοπό να προσφέρει λύση σε δύο υπαρκτά προβλήματα, που πολλοί ιδιοκτήτες κατοικίδιων αντιμετωπίζουν ή είναι δυνατόν να αντιμετωπίσουν. Το πρώτο αφορά την παρακολούθηση της υγείας του κατοικίδιου και τη καταγραφή του ιατρικού ιστορικού του. Η ψηφιοποίηση του βιβλιαρίου υγείας και η μεταφορά του σε κινητές συσκευές, διασφαλίζει την πρόσβαση του από οποιοδήποτε σημείο και οποιαδήποτε χρονική στιγμή, ενώ παράλληλα ενισχύει την προστασία του από φυσικές φθορές και καταστροφές. Το όφελος αυτό μεταφέρεται και στον κτηνίατρο, αφού η φύση των δεδομένων επιτρέπει τον εύκολο διαμοιρασμό και πρόσβαση σε αυτές τις πληροφορίες. Το δεύτερο πρόβλημα αφορά την μαζική κοινοποίηση της εξαφάνισης ενός κατοικίδιου. Η εξαφάνιση, ειδικά σε αστικά περιβάλλοντα, εγκυμονεί πληθώρα κινδύνων για το κατοικίδιο και η άμεση ενημέρωση είναι εξαιρετικά σημαντική στην ταχεία ανεύρεση του. Οι λειτουργίες αυτές περιλαμβάνονται σε μια εφαρμογή για κινητές συσκευές Android και καλούνται να διευκολύνουν, με έναν εύληπτο τρόπο, την καθημερινότητα όλων των ιδιοκτητών κατοικίδιων.

Περίληψη

Η εργασία αυτή αποτελεί μια προσέγγιση στη διευκόλυνση πρόσβασης στις σχετικές με την υγεία πληροφορίες των κατοικίδιων και στην έγκαιρη και μαζική ενημέρωση για περιστατικά εξαφάνισης κατοικίδιων. Το βιβλιάριο υγείας του κατοικίδιου, διατίθεται σε φυσική μορφή, γεγονός που το καθιστά ευάλωτο σε φθορές, καταστροφές και απώλειες. Παράλληλα, οι εξαφανίσεις κατοικίδιων αποτελούν περιστατικά που χρήζουν άμεσης επέμβασης και ενημέρωσης. Για την αντιμετώπιση αυτών των προβλημάτων, κρίθηκε σκόπιμο να αναπτυχθεί μια εφαρμογή για κινητές συσκευές Android, η οποία στηρίζεται σε μια στιβαρή υποδομή για την αξιόπιστη παροχή των υπηρεσιών αυτών, ταυτόχρονα, σε πλήθος ατόμων, ανεξάρτητα από περιοχή. Η εφαρμογή περιλαμβάνει λειτουργίες τόσο για τον κοινό χρήστη, κάτοχο κατοικίδιου, όσο και για τον κτηνίατρο, αν και για τον δεύτερο, οι λειτουργίες περιορίζονται προσωρινά στην προβολή και επεξεργασία των στοιχείων του κατοικίδιου του ασθενή, μέσω ανάγνωσης ενός QR code. Ο χρήστης αφού δημιουργήσει έναν λογαριασμό και συνδεθεί, μπορεί να προσθέσει στοιχεία για νέα κατοικίδια και να καταχωρήσει κατηγοριοποιημένες εγγραφές στον ιατρικό τους φάκελο. Υποστηρίζεται, επίσης, και η ανανέωση των στοιχείων του κατοικίδιου, με δυνατότητα δήλωσης του ως εξαφανισμένο. Μέσω αυτής της λειτουργίας, επιτρέπεται στον χρήστη να επιλέξει μια τοποθεσία πιθανής εξαφάνισης του κατοικίδιου στον χάρτη και να εισάγει στοιχεία επικοινωνίας. Η διαδικασία αυτή ολοκληρώνεται με την αποστολή ενημερωτικών email σε όλους τους χρήστες που έχουν δηλώσει ενδιαφέρον, και περιλαμβάνει τα στοιχεία του κατοικίδιου, καθώς και μια γραφική απεικόνιση της περιοχής εξαφάνισης. Λοιπές λειτουργίες περιλαμβάνουν την προβολή των πλησιέστερων κτηνιατρείων, τον ορισμό υπενθυμίσεων για θέματα υγείας του κατοικίδιου και την επεξεργασία των προτιμήσεων σχετικά με την ενημέρωση εξαφανίσεων μέσω email. Υπάρχει πληθώρα λειτουργιών που μπορούν να προστεθούν στην εφαρμογή, με την ανάπτυξη της να συνεχίζεται και μετά το πέρας της διπλωματικής εργασίας.

Monitoring the health of pets and reporting their disappearance.
Application on Android systems

Barkos Georgios

Abstract

This thesis offers an approach to facilitate the access to the pet's health data, and inform the public early and massively about pet disappearances. The pet's healthbook is available in physical form which makes it susceptible to damage, destruction, loss or misplacement. At the same time, pet disappearances are incidents that require immediate intervention and communication. Due to the nature of these issues, it was deemed appropriate to develop an Android mobile application based on a robust backend infrastructure to reliably provide its services to a large number of people, simultaneously, with no geographical restrictions. The application includes functions for both the general user and the veterinarian. However, the vet is temporarily limited to viewing and editing the pet's health data by scanning a QR code. The user, after creating an account and logging in, can add new pets and records to their pet's medical record for all supported medical categories, as well as edit a pet's personal information. Marking a pet as missing is also supported. Through this feature, the user can select an approximate location of the incident's occurrence on the map and enter their contact information. The process is completed by sending mass emails to all users of the application who have opted in to this feature, containing details about the pet as well as a graphical representation of the pet's likely location. Other features include viewing the nearest veterinary clinics, setting reminders for a pet's medical events and changing account preferences regarding receiving emails about lost pets. There are many features that can be added to the application, as its development will continue after the completion of this thesis.

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω την οικογένεια μου. Ήταν δίπλα μου σε όλη τη διάρκεια της εκπόνησης της εργασίας και μου έδιναν θάρρος και κουράγιο να συνεχίσω. Ευχαριστώ του φίλους μου, για την αλληλοσυμπαράσταση, καθώς, λίγο-πολύ, όλοι περνάμε τα ίδια. Ένα μεγάλο ευχαριστώ πηγαίνει στον συμφοιτητή, συνάδελφο και προπάντων, φίλο μου, Χαράλαμπο Σωτηρίου, για τις συμβουλές του και την πολύτιμη καθοδήγηση στην ανάπτυξη της Android εφαρμογής. Τέλος, θέλω να ευχαριστήσω τον καθηγητή μου, κ. Αδαμίδη Παναγιώτη, για την υπομονή του, την κατανόηση που έδειξε, καθώς και την καθοδήγηση που παρείχε στη διάρκεια της εκπόνησης της εργασίας.

Περιεχόμενα

Κατάλογος Σχημάτων

Συντομογραφίες

I/O	Input/ Output
FIFO	First In First Out
NPM	Node Package Manager
CRUD	Create-Read-Update-Delete
AOSP	Android Open Source Project
HAL	Hardware Abstraction Layer
UI	User Interface
MVVM	Model-View-ViewModel
APK	Android Application Package
DEX	Dalvik Executable
JVM	Java Virtual Machine
IDE	Integrated Development Environment
CMS	Content Management System
HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
API	Application Programming Interface
ART	Android Runtime
JSON	JavaScript Object Notation
JWT	JSON Web Token
SDK	Software Development Kit
ΕΜΖΣ	Εθνικό Μητρώο Ζώων Συντροφιάς

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εξημέρωση των ζώων και οι απαρχές της

Η εξημέρωση των ζώων είναι μια αμοιβαία σχέση μεταξύ των ζώων και των ανθρώπων που παρεμβαίνουν στην φροντίδα και την αναπαραγωγή τους [36]. Αποτελεί τη διαδικασία της προσαρμογής της άγριας ζωής, προς εκμετάλλευση από τον άνθρωπο. Η διαδικασία αυτή περατώνεται μέσω της επιλεκτικής αναπαραγωγής των ατόμων. Ζώα τα οποία φέρουν επωφελή προς τον άνθρωπο χαρακτηριστικά, επιλέγονται προς ελεγχόμενη και στοχευμένη αναπαραγωγή, με σκοπό την απομόνωση και την ανάδειξη των χαρακτηριστικών αυτών στις μελλοντικές γενιές.

Οι κλιματικές και περιβαλλοντικές αλλαγές που επέφερε η κορύφωση της τελευταίας εποχής των παγετώνων, περίπου 21.000 χρόνια πριν, καθώς και άλλα παγκόσμια κλιματικά γεγονότα, όπως η περίοδος του Younger Dryas [37], 12.900 χρόνια πριν, κατέστησαν την αναζήτηση για τροφή εξαιρετικά δύσκολη και ανάγκασαν τους ανθρώπους να εντείνουν τις προσπάθειες εύρεσης της. Από την αρχή της Ολόκαινου εποχής, 11.700 χρόνια πριν, οι ευνοϊκές κλιματικές συνθήκες και η αύξηση του ανθρώπινου πληθυσμού, οδήγησε σε μικρής κλίμακας εξημερώσεις ζώων αλλά και φυτών, το οποίο επέτρεψε στους ανθρώπινους πληθυσμούς να αυξήσουν την τροφή που προσλαμβάνουν μέσω του κυνηγιού και της συλλογής καρπών [38].

Ο άνθρωπος εξημέρωσε ζώα για πληθώρα λόγων. Κάποια ζώα πρόσφεραν φαγητό, έμμεσα και άμεσα, άλλα βοηθούσαν σε εργασίες, άλλα προσέφεραν συντροφιά ή ακόμα και τα τρία παράλληλα. Το πρώτο ζώο που εξημερώθηκε ήταν ο λύκος, περίπου 33.000 και 11.000 χρόνια πριν [39]. Έπειτα, ακολούθησαν ζώα όπως τα πρόβατα, οι αγελάδες και οι χοίροι, την ίδια περίοδο όπου η ανθρώπινη κοινωνία άρχισε να μετασχηματίζεται σε αγροτοκεντρική από τροφοσυλλεκτική [40]. Η διαδικασία και η απαρχή της εξημέρωσης παραμένουν άγνωστες, ωστόσο είναι ξεκάθαρο πως, οι πρόγονοι των πλέον εξημερωμένων ζώων, εκδήλωσαν χαρακτηριστικά τα οποία επωφελούσαν τους ανθρώπους - από γευστικό κρέας και ζεστό τρίχωμα σε μια φυσική συμπάθεια και προτίμηση προς αυτούς [41]. Μελέτες παρέχουν ενδείξεις πως οι πρώτοι σκύλοι ήταν γενετικά προδιατεθειμένοι να είναι φιλικοί [42], χαρακτηριστικό το οποίο ήταν καταλυτικό για την δημιουργία στενών, αμοιβαίων σχέσεων μεταξύ αυτών και των ανθρώπων, κατά τις οποίες, ο άνθρωπος παρέχει τροφή και καταφύγιο και ο σκύλος βοήθεια στο κυνήγι και φύλαξη της περιοχής [41]. Πιο συγκεκριμένα, υπάρχουν κάποια συμπεριφορικά χαρακτηριστικά τα οποία καθιστούν συγκεκριμένα είδη ζώων, καταλληλότερους υποψήφιους προς εξημέρωση. Αυτά είναι:

- Το μέγεθος και η οργάνωση των κοινωνικών δομών τους [43]. Ζώα με ενισχυμένη κοινωνική συμπεριφορά και ισχυρούς κοινωνικούς δεσμούς, τείνουν να εξημερώνονται πιο εύκολα.
- Η διαθεσιμότητα συντρόφων για αναπαραγωγή και η επιλεκτικότητα στην εύρεση τους [43]. Άτομα με πληθώρα επιλογών και λίγα κριτήρια επιλογής, διασφαλίζουν την συνεχή αναπαραγωγή.
- Η ευκολία και η ταχύτητα με την οποία οι γονείς δένονται με τους απογόνους τους και η ωριμότητα και η κινητικότητα των νεογνών μετά τη γέννα [43]. Τα ζώα τα οποία φροντίζουν και προστατεύουν τα μικρά τους, σε συνδυασμό με την υψηλή ετοιμότητα των δεύτερων μετά τη γέννα, διασφαλίζουν χαμηλότερα ποσοστά θνησιμότητας και κατ' επέκταση, έναν υγιή, αριθμητικά, πληθυσμό.
- Ο βαθμός ευελιξίας στη διατροφή τους και η ανοχή στα περιβάλλοντα διαβίωσης [43]. Τα ζώα με ποικιλία στη διατροφή τους και ελάχιστες απαιτήσεις όσον αφορά τις γεωγραφικές και

κλιματικές συνθήκες της περιοχής διαβίωσης τους, έχουν μεγαλύτερες ποσοστό ανοχής σε ραγδαίες αλλαγές του οικοσυστήματος.

- Η αντιδράσεις τους απέναντι στον άνθρωπο και σε νέα περιβάλλοντα, συμπεριλαμβανομένων των αντιδράσεων πάλης ή φυγής [43]. Η επιφυλακτικότητα στην παρουσία του ανθρώπου και η αντιδραστικότητα τους σε εξωτερικά ερεθίσματα, είναι τα πιο σημαντικά κριτήρια για την ενημέρωση ή μη ενός είδους.

Η εξημέρωση αποτελεί μια διαδικασία που, σε βάθος χρόνου, επιφέρει ριζικές αλλαγές στο είδος. Τα εξημερωμένα ζώα διαφέρουν γενετικά από τους άγριους προγόνους τους, ενώ η φιλικότητα προς τους ανθρώπους αποτελεί γενετικό χαρακτηριστικό τους [43]. Επιπλέον, εμφανίζουν σωματικά χαρακτηριστικά που υπάγονται στο «σύνδρομο της εξημέρωσης», όπως μικρότερα κρανία, κρεμαστά και εύκαμπτα αυτιά ή παραλλαγές στο χρώμα του τριχώματος τους [40]. Πολλά είδη ζώων που εξημερώθηκαν για κάποιον συγκεκριμένο σκοπό, σήμερα δεν τον εξυπηρετούν. Παράδειγμα αποτελούν οι σκύλοι, οι οποίο αρχικά εξημερώθηκαν για να προσφέρουν βοήθεια στο κυνήγι και στην φύλαξη οικισμών, καθώς και οι γάτες, οι οποίες έλεγχαν τους πληθυσμούς των παρασίτων. Τα δύο αυτά είδη, αν και ακόμα επιτελούν τους πρωταρχικούς ρόλους τους, υπό περιπτώσεις, κατά κύριο λόγο, αποτελούν ζώα συντροφιάς.

1.2 Άνθρωπος και κατοικίδια

Τα κατοικίδια ζώα έχουν αποδειχθεί καταλύτες στην ανάπτυξη του σύγχρονου ανθρώπινου πολιτισμού. Η συνεισφορά τους στην εύρεση και παροχή αξιόπιστης πηγής τροφής, αλλά και στην αντικατάσταση του ανθρώπου σε πολλές χειρωνακτικές εργασίες, επιτάχυνε την πρόοδο του ανθρώπου, μειώνοντας τον χρόνο που έπρεπε να διατεθεί καθημερινά, αποκλειστικά, για την επιβίωση, επιτρέποντας τον να εξερευνήσει διάφορες πτυχές του κόσμου. Ωστόσο, η προσφορά των κατοικίδιων δεν σταματάει εκεί, αλλά εκτείνεται και στις ευεργετικές επιρροές στην ψυχοσωματική υγεία του ανθρώπου. Οι ιδιοκτήτες κατοικίδιων είναι λιγότερο πιθανό να υποφέρουν από κατάθλιψη [44], ενώ η ενασχόληση με αυτά μπορεί να ανεβάσει τα επίπεδα της σεροτονίνης και της ντοπαμίνης, τα οποία επιφέρουν ηρεμία και χαλάρωση [44]. Επίσης, η συντροφικότητα που μπορεί να προσφέρει ένα κατοικίδιο, μπορεί να καταπολεμήσει την μοναξιά, ικανοποιώντας την ανθρώπινη ανάγκη του να είναι επιθυμητός. Επιπλέον, η ιδιοκτησία ενός κατοικίδιου, είναι κρίσιμη στην εκμάθηση, τόσο των παιδιών, όσο και τον ενηλίκων, της αρετής της υπευθυνότητας. Ένα κατοικίδιο αποτελεί, συνήθως, μια μακροχρόνια δέσμευση, την οποία ο άνθρωπος οφείλει να τηρεί.

Η φροντίδα των κατοικίδιων από τον άνθρωπο, πέρα από το ότι κρίνεται ηθικά σωστό να λαμβάνει χώρα, κατοχυρώνεται και από την πολιτεία. Με βάση τις παραγράφους δ και ε του άρθρου 5 του νόμου ν. 4830/2021 «Νέο πλαίσιο για την ευζωία των ζώων συντροφιάς - Πρόγραμμα "ΑΡΓΟΣ"» [45], το άτομο υποχρεούται να μεριμνά για τη ιατροφαρμακευτική περίθαλψη του κατοικίδιου του, συμπεριλαμβανομένων των ετήσιων εμβολιασμών, καθώς και να φροντίζει για την ασφαλή και άνετη διαβίωση του. Η καταγραφή των κτηνιατρικών επισκέψεων, των ασθενειών και των προγραμματισμένων εμβολιασμών, γίνεται στο βιβλιάριο του κατοικίδιου, το οποίο αποτελεί επίσημο έγγραφο. Πέρα από τις διατάξεις περί ατομικής ευθύνης προς τα κατοικίδια, το άρθρο 7 του ίδιου νόμου ορίζει τις συλλογικές αρμοδιότητες και υποχρεώσεις σε επίπεδο δήμων. Οι δήμοι οφείλουν να παρέχουν προγράμματα διαχείρισης αδέσποτων ζώων το οποίο περιλαμβάνει κατ' ελάχιστον (α) την περισυλλογή, (β) την παροχή κτηνιατρικής περίθαλψης, (γ) την ηλεκτρονική σήμανση και την καταγραφή στο Εθνικό Μητρώο Ζώων Συντροφιάς (δ) τη στείρωση και (ε) την υιοθεσία του [45]. Ωστόσο, η υγεία του κατοικίδιου δεν είναι κάτι που αφορά αποκλειστικά το κατοικίδιο αυτό καθαυτό, αλλά και τους

ανθρώπους γύρω του. Ασθένειες όπως η λύσσα, η τοξοπλάσμωση αλλά και διαφόρων ειδών παράσιτα, μπορούν να μεταδοθούν και να μεταφερθούν στον άνθρωπο. Είναι, λοιπόν, προς ατομικό και συλλογικό όφελος του ανθρώπου, τα κατοικίδια να παραμείνουν υγιή, όχι μόνο επειδή είναι ηθικά σωστό ή επιβάλλεται από τον νόμο, αλλά καθώς τίθεται θέμα προσωπικής και δημόσιας υγείας. Η έννοια της φροντίδας επεκτείνεται και στην προστασία της σωματικής ακεραιότητας από εξωτερικού κινδύνους. Τα ζώα συντροφιάς, βάση ορισμού, προορίζονται να συντηρηθούν από τον άνθρωπο κυρίως μέσα στην οικία του [47]. Τα ζώα αυτά, κατά κύριο λόγο, δεν είναι συνηθισμένα στην ζωή εκτός, πόσο μάλλον σε αστικά περιβάλλοντα τα οποία μπορούν να φανούν χαώδη για τις αισθήσεις τους. Η εξαφάνιση ενός τέτοιου κατοικίδιου πρέπει να χρήζει άμεσης επέμβασης, καθώς η ζωή του διατρέχει αυξημένο κίνδυνο.

1.3 Στόχοι της εργασίας

Η εργασία, σαν σύνολο, έχει ως σκοπό να ενημερώσει και να εφιστήσει την προσοχή των αναγνωστών στο εξαιρετικά σημαντικό θέμα της φροντίδας και διαχείρισης ενός κατοικίδιου, παρέχοντας έναν πρακτικό τρόπο επιτέλεσης αυτών των λειτουργιών, τόσο από την πλευρά του ιδιοκτήτη, όσο και τρίτων προσώπων, όπως του κτηνιάτρου και λοιπών φιλόζωων μελών της εκάστοτε τοπικής κοινωνίας. Οι στόχοι της παρούσας διπλωματικής εργασίας μπορούν να διακριθούν σε δύο κατηγορίες. Η πρώτη αφορά την ψηφιοποίηση των στοιχείων και του ιατρικού φακέλου του κατοικίδιου, με γνώμονα την φορητότητα και την διευκόλυνση της καταχώρησης νέων ιατρικών στοιχείων. Σε αυτήν ενσωματώνεται και ο ρόλος του κτηνιάτρου, αφού στόχο έχει να διευκολύνει την πρόσβασή τους στα στοιχεία των προς εξέταση κατοικίδιων, σε ηλεκτρονική μορφή, αλλά και την καταχώρηση νέων ιατρικών στοιχείων κατά την επίσκεψη τους. Ο δεύτερος επικεντρώνεται σε μια πιο κοινωνική διάσταση της ιδιοκτησίας ενός κατοικίδιου. Πρόκειται για ένα μέσο κοινοποίησης πιθανών εξαφανίσεων σε άτομα που εκδηλώνουν σχετικό ενδιαφέρον. Στόχος είναι η έγκαιρη, μαζική ενημέρωση του πλήθους, με σκοπό την άμεση κινητοποίηση του για την ανεύρεση των εν λόγω κατοικίδιων. Για τον λόγο αυτό, έχει αναπτυχθεί μια εφαρμογή για κινητές συσκευές Android, η οποία επιτρέπει την προσθήκη και διαχείριση των κατοικίδιων του χρήστη, με πλήρη σεβασμό στο φυσικό και επίσημο βιβλιάριο υγείας, καθώς και με λειτουργίες μαζικής ενημέρωσης, με αποστολή email με λεπτομερή στοιχεία σχετικά με την εξαφάνιση των κατοικίδιων.

1.4 Διάρθρωση της εργασίας

Η εργασία αποτελείται, συνολικά, από πέντε κεφάλαια. Το πρώτο αποτελεί την εισαγωγή, όπου γίνονται αναφορές σε ορισμούς και έννοιες, χρήσιμες για την κατανόηση του παρόντος θέματος, με σκοπό την προετοιμασία του αναγνώστη για τις διάφορες πτυχές της εργασίας. Στο δεύτερο κεφάλαιο αναλύονται με λεπτομέρεια όλες οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του πρακτικού μέρους της εργασίας. Αυτό χωρίζεται σε δύο διακριτές ενότητες, τις τεχνολογίες του backend και του frontend μέρους της εφαρμογής. Το τρίτο κεφάλαιο περιλαμβάνει μια σε βάθος ανάλυση του προβλήματος προς επίλυση, με παρουσίαση σχετικών προσεγγίσεων από τρίτους και σύγκριση αυτών με την παρούσα υλοποίηση. Έπειτα, στο τέταρτο κεφάλαιο, παρουσιάζεται η εφαρμογή αυτή καθαυτή, με περιγραφή όλων των λειτουργιών και των πιθανών ενεργειών που μπορεί να εκτελέσει ο χρήστης, ενώ παρατίθενται και σχετικές εικόνες για μια ολοκληρωμένη όψη των εν λόγω χαρακτηριστικών. Τέλος, στο πέμπτο κεφάλαιο γίνεται αυτοκριτική με παράθεση συμπερασμάτων για την όλη διαδικασία, τόσο της σχεδίασης όσο και της ανάπτυξης της εφαρμογής, με επίκεντρο στα προβλήματα που αντιμετωπίστηκαν αλλά και στις βελτιώσεις και προσθήκες που εν δυνάμει μπορούν να πραγματοποιηθούν.

1.5 Επίλογος

Στο κεφάλαιο αυτό πραγματοποιείται η εισαγωγή του αναγνώστη στο θέμα που πραγματεύεται η εργασία. Αρχικά, ορίστηκε η έννοια της εξημέρωσης των ζώων, πότε ξεκίνησε και για ποιους λόγους επιδιώχθηκε σε τέτοια κλίμακα. Στη συνέχεια αναφέρθηκαν οι λόγοι επιλογής των ζώων που εξημερώθηκαν ανά τα χρόνια, καθώς και η επίδραση της διαδικασίας αυτής στη φυσιολογία τους. Έπειτα, αναφέρθηκε η συνεισφορά των κατοικίδιων στην ανθρώπινη εξέλιξη και ευζωία. Επιπροσθέτως, έγινε εκτενής αναφορά στην ευθύνη και στις υποχρεώσεις που έχει ο άνθρωπος απέναντι στα κατοικίδια, τόσο σε ατομικό όσο και σε συλλογικό επίπεδο. Τέλος, αναλύθηκαν οι στόχοι της εργασίας και παρατέθηκε η διάρθρωση των κεφαλαίων που την απαρτίζουν.

Κεφάλαιο 2ο: Τεχνολογίες Ανάπτυξης

2.1 Εισαγωγή

Το πιο καθοριστικό μέρος στην απόφαση της υλοποίησης της προτεινόμενης λύσης, ήταν η διατήρηση της φορητότητας του βιβλιαρίου και η αμεσότητα δράσης, όσον αφορά το μέρος των εξαφανίσεων των κατοικίδιων. Οι δύο αυτοί παράγοντες δεν αφήνουν πολλά περιθώρια επιλογής της πλατφόρμας για την οποία μπορεί να χτιστεί η εφαρμογή. Ο προγραμματισμός της εφαρμογής για κινητές συσκευές αποτελεί την πιο ταιριαστή λύση, καθώς επιτρέπει την εύκολη πρόσβαση των εν λόγω πληροφοριών από οποιοδήποτε σημείο, καθώς είναι εξαιρετικά πιθανό πως θα διατηρείται στην κατοχή του ατόμου στα περισσότερα από τα πιθανά σενάρια χρήσης που υποστηρίζει η εφαρμογή. Ωστόσο, για τη σωστή λειτουργία της, απαιτείται και μια υποδομή που θα υποστηρίζει τη χρήση της από πολλαπλούς χρήστες, αλλά και θα καθιστά λειτουργικά τα προτεινόμενα χαρακτηριστικά της. Για τον λόγο αυτό, κρίθηκε απαραίτητη η ύπαρξη ενός server που θα εξυπηρετεί την εφαρμογή στις κινητές συσκευές. Προφανώς, αυτό συνεπάγεται ότι, για τη χρήση της εφαρμογής, απαιτείται σύνδεση στο διαδίκτυο. Στο κεφάλαιο αυτό θα γίνει ανάλυση όλων των επιμέρους τεχνολογιών που επιλέχθηκαν για την υλοποίηση των διακριτών αυτών στοιχείων της εφαρμογής, τόσο για την υλοποίηση αυτή καθαυτή όσο και για τον έλεγχο αυτής και τη συγγραφή της.

2.2 Τεχνολογίες που Χρησιμοποιήθηκαν

Για την εύρυθμη λειτουργία της εφαρμογής, χρησιμοποιήθηκαν ποικίλες τεχνολογίες. Ο διαχωρισμός τους γίνεται με βάση τον ευρύτερο ορισμό της αρχής του «separation of concerns» [64], κατά την οποία μια εφαρμογή χωρίζεται σε διακριτά τμήματα, με το καθένα να έχει έναν και μόνο έναν συγκεκριμένο ρόλο. Από τον διαχωρισμό αυτό, προκύπτουν δύο γενικές κατηγορίες, το backend και το frontend.

2.2.1 Backend

Ο όρος backend χρησιμοποιείται για να περιγράψει τα τμήματα μιας εφαρμογής που δεν είναι προσβάσιμα από τον χρήστη. Γνωστό και ως data access layer, περιλαμβάνει όλες τις λειτουργίες που σχετίζονται με την ανάπτυξη της εφαρμογής από τη μεριά του εξυπηρετητή (server-side-development). Παραδείγματα διεργασιών που διαχειρίζεται το backend είναι ο χειρισμός και η επεξεργασία ενός εισερχόμενου αιτήματος από κάποια πηγή, η πρόσβαση και η χειραγώγηση των δεδομένων μιας βάσης δεδομένων, η κρυπτογράφηση και αποκρυπτογράφηση δεδομένων, καθώς και ο χειρισμός αρχείων προς μεταφόρτωση (upload) και λήψη (download) [1].

2.2.1.1 JavaScript

Η JavaScript είναι μια scripting γλώσσα που επιτρέπει την υλοποίηση σύνθετων λειτουργειών σε ιστοσελίδες [2]. Κάνει δυνατή τη δημιουργία δυναμικά ανανεώσιμου περιεχομένου και αποτελεί το μέσο επεξεργασίας του περιεχομένου και της δομής μια σελίδας. Η JavasScript προσθέτει την έννοια της «συμπεριφοράς» σε μια σελίδα και σε συνδυασμό με την HTML, που καθορίζει τη δομή της, και των CSS, που επιδρούν στην εμφάνιση και το στυλ της, αποτελούν τον βασικό κορμό των τεχνολογιών ιστού.

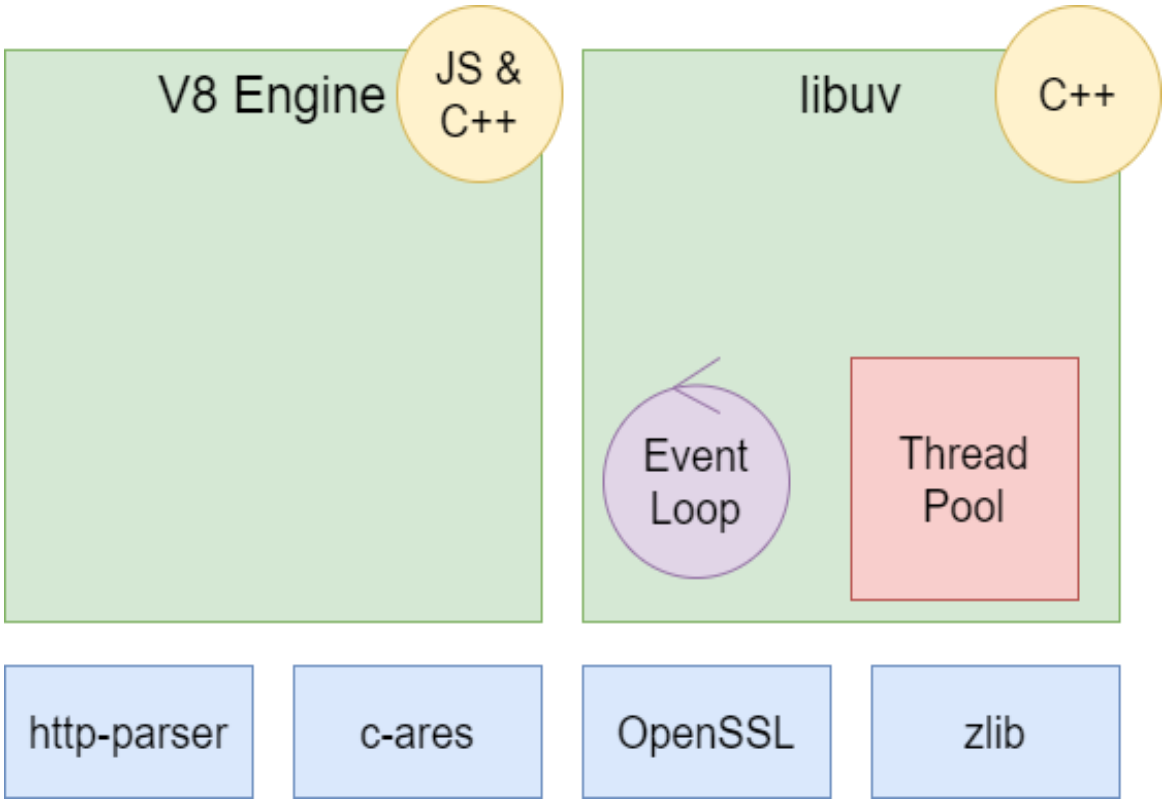
Η JavaScript ακολουθεί το ECMAScript πρότυπο [59], το οποίο παρέχεται από τον ECMA International [60], έναν οργανισμό προτυποποίησης για συστήματα επικοινωνίας και πληροφοριών. Σκοπός του προτύπου είναι η διασφάλιση της διαλειτουργικότητας των ιστοσελίδων, ανεξάρτητα από τον περιηγητή

ιστού που χρησιμοποιείται. Με βάση τη συμμόρφωση σε αυτούς τους κανόνες, προκύπτουν συγκεκριμένα χαρακτηριστικά που προσδιορίζουν την JavaScript. Αρχικά, είναι μια loosley typed γλώσσα. Αυτό σημαίνει πως διαθέτει type inference ανάλογα με το context που χρησιμοποιούνται οι μεταβλητές. Μέσω αυτού προκύπτουν αντιφατικές μετατροπές μεταξύ τύπων όπως η μετατροπή ενός αριθμού σε string, αν αυτός προστεθεί στο δεύτερο, αλλά την μετατροπή του string σε αριθμό αν ο πρώτος αφαιρεθεί από το string. Επίσης, είναι δυναμική, καθώς μια μεταβλητή που ήταν αρχικά string, μπορεί να στεγάσει έναν αριθμό. Επιπλέον, υποστηρίζει χαρακτηριστικά της αντικειμενοστρέφειας, όπως την κληρονομικότητα, μέσω των prototypes και όχι με κλάσεις και interfaces. Πιο συγκεκριμένα, κάθε αντικείμενο διαθέτει ένα prototype, το οποίο είναι ένα αντικείμενο, το οποίο με τη σειρά του διαθέτει ένα protoype. Στο τέλος της αλυσίδας, βρίσκεται το Object.prototype, το οποίο αποτελεί το πιο βασικό prototype που έχουν όλα τα αντικείμενα στην JavaScript, εξορισμού. Όταν εκτελείται μια μέθοδος ενός αντικειμένου, αν αυτή δεν υπάρχει, καλείται η μέθοδος του prototype του. Η διαδικασία αυτή επαναλαμβάνεται αναδρομικά μέχρι είτε να βρεθεί η μέθοδος ή κλήσεις να φτάσουν στο τέλος της αλυσίδας. Επιπροσθέτως, μπορούν να προστεθούν και να αφαιρεθούν πεδία σε αντικείμενα κατά το runtime, μέσω της ανάθεσης κάποιας τιμής στο εν λόγω επιθυμητό πεδίο του αντικειμένου. Για την προσθήκη ενός νέου πεδίου, υποστηρίζεται η σύνταξη obj.x = 1, αλλά και obj[‘x] = 1. Αυτό είναι εφικτό, καθώς ένα αντικείμενο στην JavaScript είναι πρακτικά ένα associative array. Επίσης υποστηρίζονται και χαρακτηριστικά του συναρτησιακού προγραμματισμού, καθώς μια συνάρτηση θεωρείται αντικείμενο. Λοιπά χαρακτηριστικά αποτελούν τα promises και τα async/await keywords για τον χειρισμό ασύγχρονων λειτουργιών, αλλά και οι ανώνυμες συναρτήσεις, οι οποίες μπορούν να αποτελέσουν και παραμέτρους σε άλλες συναρτήσεις.

Η JavaScript, πέρα από την χρήση της από τους web browsers, χρησιμοποιείται και στην πλευρά των servers, με το πιο δημοφιλές runtime της να είναι το Node.js, το οποίο και χρησιμοποιείται στην υλοποίηση του backend της παρούσας διπλωματικής εργασίας.

2.2.1.2 Node.js

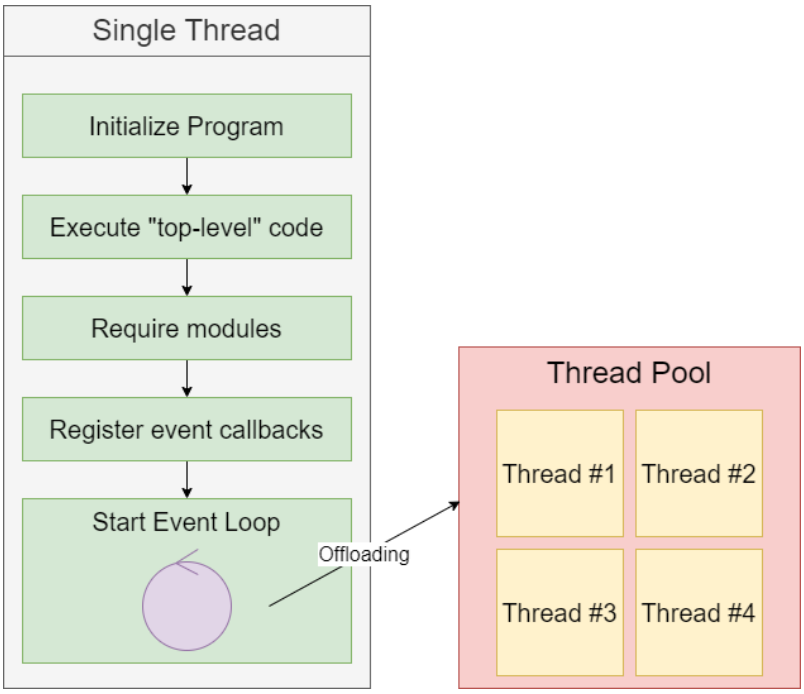
Το Node.js είναι ένα ασύγχρονο, event-driven runtime της JavaScript, το οποίο είναι σχεδιασμένο να υποστηρίζει την κατασκευή εύκολα επεκτάσιμων δικτυακών εφαρμογών [3]. Διαθέτει πληθώρα από dependencies, όπως το llhttp για το το HTTP parsing, το c-ares για τη διαχείριση των ασύγχρονων DNS αιτημάτων, το OpenSSL για κρυπτογράφηση και το zlib για συμπίεση και αποσυμπίεση. Ωστόσο, τα πιο σημαντικά είναι το V8 engine της google και το libuv. Η βιβλιοθήκη V8 παρέχει στο Node.js την JavaScript engine, η οποία ελέγχεται μέσω του V8 C++ API. Το libuv αποτελεί μια βιβλιοθήκη σε C, η οποία επικεντρώνεται στις ασύγχρονες I/O λειτουργίες και είναι αυτό που παρέχει στο Node.js πρόσβαση στο λειτουργικό σύστημα, στο σύστημα αρχείων και στο δίκτυο. Επιπλέον, το libuv υλοποιεί δύο εξαιρετικά σημαντικά features του Node.js, το event loop και το thread pool. Το event loop είναι υπεύθυνο για τη διαχείριση διεργασιών, όπως την εκτέλεση των callbacks και των network I/O λειτουργιών, ενώ το thread pool χρησιμοποιείται για πιο κοστοβόρες εργασίες, όπως την πρόσβαση στο σύστημα αρχείων, τη συμπίεση και την κρυπτογράφηση [3]. Γίνεται εύκολα αντιληπτό, πως, εφόσον μια Node.js εφαρμογή τρέχει σε ένα thread, το thread pool, που μπορεί να φτάσει τα 128, αποτελεί ένα ιδιαίτερα σημαντικό κομμάτι της, που μπορεί, εν δυνάμει, να αυξήσει την απόδοση της εκάστοτε εφαρμογής. Στο Σχήμα 2.1, φαίνεται η δομή του Node.js με τα κύρια συστατικά του, ενώ στο Σχήμα 2.2 μια ενδεικτική εκτέλεση εφαρμογής σε Node.js.



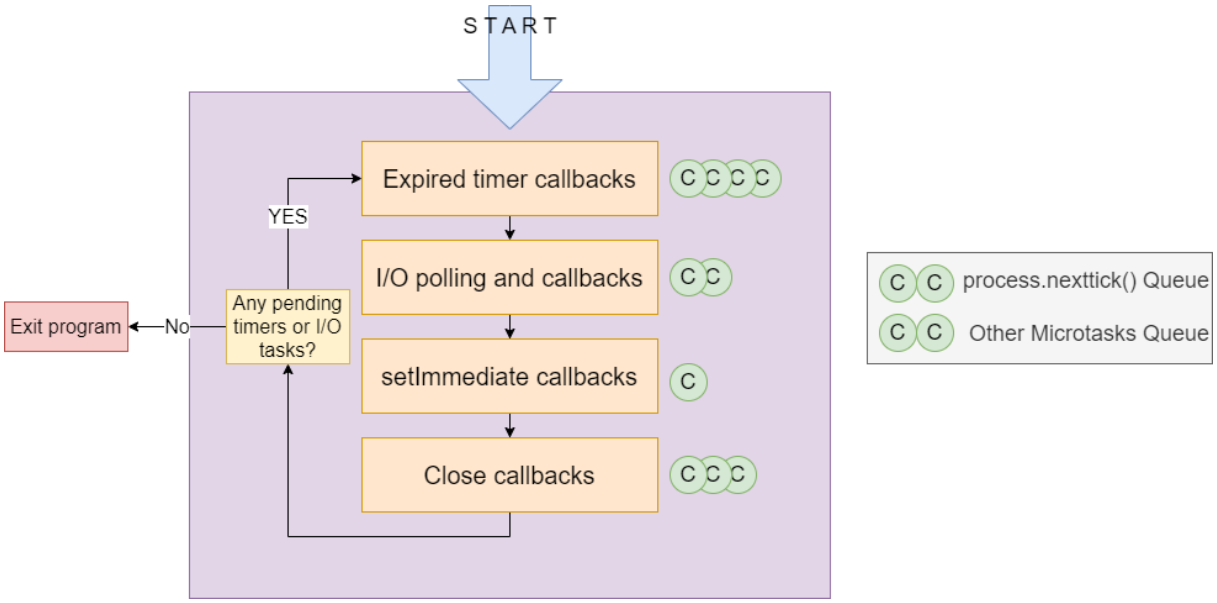
Σχήμα 2.1: Δομή του Node.js

Κατά την εκτέλεση μιας εφαρμογής, πρώτα γίνεται αρχικοποίηση του προγράμματος. Έπειτα εκτελείται ο top-level κώδικας, δηλαδή ο κώδικας που δεν βρίσκεται μέσα σε κάποιο callback function, και στη συνέχεια γίνεται εισαγωγή των απαιτούμενων modules. Τέλος, γίνεται η εγγραφή όλων των callbacks. Μόνο μετά από αυτήν τη διαδικασία θα ξεκινήσει να τρέχει το event loop. Το event loop αποτελείται από πολλά στάδια, και κάθε στάδιο έχει ένα FIFO callback queue, το οποίο περιλαμβάνει τα callbacks από τα events που λαμβάνονται από το loop. Τα τέσσερα πιο βασικά είναι τα Expired timer callbacks, I/O polling and callbacks, setImmediate callbacks και Close callbacks [4]. Για να μεταβεί το loop στο επόμενο στάδιο, πρέπει να ικανοποιηθούν ένα-ένα, όλα τα callbacks του queue του τρέχοντος σταδίου. Αν ένα callback εισέλθει στο queue κάποιου σταδίου που έχει παρέλθει, για να ικανοποιηθεί, θα πρέπει το loop να ξεκινήσει από την αρχή και να φτάσει στο αντίστοιχο στάδιο. Εκτός από αυτά τα στάδια, αξίζει να αναφερθούν ακόμα δύο, το process.nexttick() queue και το other microtasks queue. Το process.nexttick() είναι μια συνάρτηση που χρησιμοποιείται για την εκτέλεση κάποιου callback, απευθείας μετά το τρέχον στάδιο του event loop [4]. Το microtasks queue είναι υπεύθυνο για τον χειρισμό των promises που προκύπτουν από ασύγχρονες διεργασίες [4]. Αν εισέλθει κάποιο callback σε οποιοδήποτε από αυτά τα δύο queues, θα εκτελεστεί μετά την ολοκλήρωση του τρέχοντος queue. Στο σχήμα 2.3 φαίνεται, γραφικά, το event loop του Node.js.

Το Node.js επιλέχθηκε για την υλοποίηση του backend για πληθώρα λόγων. Αρχικά, είναι μια οικία τεχνολογία, τόσο στη δομή της και τη χρήση της, όσο και στη γλώσσα προγραμματισμού που αυτή χρησιμοποιεί. Επίσης, υπάρχει μια τεράστια κοινότητα προγραμματιστών που το χρησιμοποιούν, γεγονός που καθιστά την εύρεση εκπαιδευτικού και συμβουλευτικού υλικού, πολύ εύκολη. Ένα, υποπροϊόν της ευρείας διάδοση του node, είναι το npm, μέσω του οποίου οι προγραμματιστές έχουν πρόσβαση σε πάνω από 1 εκατομμύριο πακέτα [5] που μπορούν να συμπεριλάβουν στις εφαρμογές τους.



Σχήμα 2.2: Διάγραμμα ενδεικτικής εκτέλεσης εφαρμογής σε Node.js



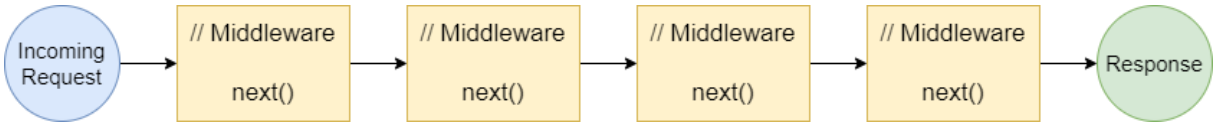
Σχήμα 2.3: Το Event Loop του Node.js

2.2.1.3 ExpressJS

Το express είναι ένα ευέλικτο web application framework του node.js που προσφέρει λειτουργίες για ανάπτυξη web και mobile εφαρμογών [6]. Είναι γραμμένο αποκλειστικά σε Node.js και προσφέρει ένα υψηλότερο επίπεδο αφαιρετικότητας, απλοποιώντας πολλές διαδικασίες. Μερικές από τις λειτουργίες που προσφέρει είναι ο ευκολότερος χειρισμός των requests και των responses, το complex routing και η προσθήκη των middleware, δηλαδή κώδικα που εκτελείται στο μέσο της διαδικασίας λήψης του request και αποστολής του response [4]. Για τον χειρισμό τους, έχει υλοποιηθεί ένα middleware stack, το οποίο εκτελεί τα middlewares με τη σειρά που δηλώνονται στον κώδικα. Κάθε φορά που

ολοκληρώνεται η εκτέλεση ενός middleware, καλείται το επόμενο στη σειρά, ενώ όλα τους επιδρούν στα ίδια, αρχικά response και request αντικείμενα. Στο σχήμα 2.4 απεικονίζεται γραφικά το middleware stack και ο κύκλος των αιτημάτων και απαντήσεων.

Το express.js αποτελεί το πιο δημοφιλές framework του node.js. Οι λειτουργίες που προφέρει διευκολύνουν και επιταχύνουν την ανάπτυξη node εφαρμογών, ενώ διαθέτει σταθερή υποστήριξη.



Σχήμα 2.4: Το middleware stack και ο κύκλο των αιτημάτων και των απαντήσεων στο Express.js

2.2.1.4 mongoDB

Η mongoDB αποτελεί ένα noSQL σύστημα βάσης δεδομένων. Κάθε βάση περιλαμβάνει δομές που καλούνται collections, ενώ κάθε collection περιλαμβάνει υποδομές, τα documents. Συγκρίνοντας την με μια σχεσιακή βάση δεδομένων, τα collections μπορούν να αντιστοιχιστούν στους πίνακες ενώ τα documents στις εγγραφές. Τα documents είναι σε BSON μορφή. Το BSON μοιάζει στη δομή με το JSON, καθώς αποτελείται από field/ value pairs, μόνο που κάθε πεδίο έχει αυστηρώς έναν τύπο δεδομένων. Η mongoDB διακρίνεται για την επεκτασιμότητα της, καθώς είναι πολύ εύκολο τα δεδομένα να κατανεμηθούν σε πολλές μηχανές, την ευελιξία της, αφού κάθε document του ίδιου collection μπορεί να έχει διαφορετικό πλήθος και τύπο πεδίων και, τέλος, την απόδοση της, καθώς παρέχει χαρακτηριστικά όπως τα embedded data models, που επιτρέπουν την λήψη σχετικών μεταξύ τους δεδομένων σε ένα round trip στη βάση, indexing για την επιτάχυνση της αναζήτησης, sharding για την κατανομή δεδομένων σε πολλές μηχανές και native data duplication για τη διασφάλιση της διαθεσιμότητας των πόρων.

2.2.1.5 Mongoose

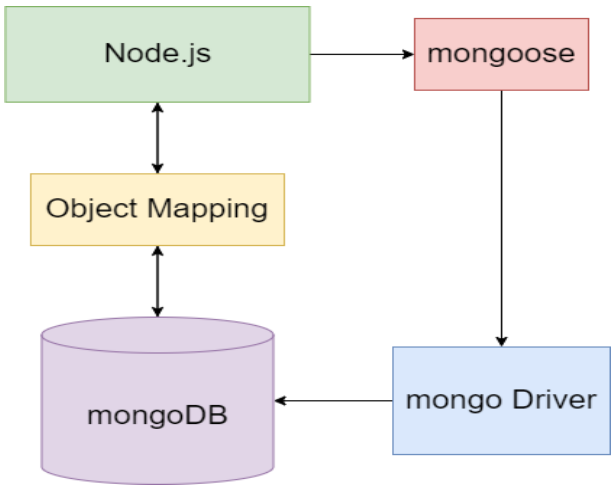
Η mongoose είναι μια object data modeling βιβλιοθήκη για την mongoDB και το Node.js, που προσφέρει ένα υψηλότερο επίπεδο αφαιρετικότητας στις λειτουργίες της βάσης. Διευκολύνει και επιταχύνει την ανάπτυξη εφαρμογών, προσφέροντας επιπλέον λειτουργίες όπως τα schemas, που χρησιμοποιούνται για την μοντελοποίηση των δεδομένων, data validation, query building και πληθώρα άλλων [4]. Τα schemas είναι το μέρος όπου περιγράφεται η δομή των δεδομένων, γίνονται οι έλεγχοι για την εγκυρότητα των δεδομένων και θέτονται προκαθορισμένες τιμές όπου και αν κριθεί απαραίτητο. Κατόπιν, από το schema, δημιουργείται το model, το οποίο είναι ένας wrapper, που προσφέρει τη διεπαφή με τη βάση, ώστε να γίνουν δυνατές οι CRUD εργασίες. Στο Σχήμα 2.5 απεικονίζεται η αναπαράσταση της σχέσης μεταξύ του Node.js, της mongoDB και της mongoose.

Η mongoose χρησιμοποιήθηκε για την απλότητα που προσφέρει στη διεπαφή με τη βάση και την αύξηση της παραγωγικότητας που τη συνοδεύει. Επίσης, καθιστά εύληπτη την αλληλεπίδραση του προγραμματιστή με τη βάση, μέσω των επιλογών της.

2.2.1.6 BULL και Redis

Το BULL αποτελεί ένα open source πακέτο για το Node.js το οποίο υλοποιεί μια ουρά για δρομολόγηση και εκτέλεση εργασιών στο υπόβαθρο. Οι ουρές, ως δομές δεδομένων, μπορούν να αποσυμφορήσουν σημεία υψηλής χρήσης των πόρων ενός server, αλλά και να μεταφέρουν εργασίες σε τρίτους servers, αυξάνοντας την απόδοση και τους χρόνους απόκρισης μιας εφαρμογής. Μια ουρά δημιουργείται μέσω

ενός στιγμιότυπου της Bull κλάσης. Από την ανάγκη διαχείρισης της ουράς, προκύπτουν τρεις διακριτές οντότητες. Η πρώτη αφορά την προσθήκη εργασιών σε αυτήν και είναι υπεύθυνη για την αποστολή μιας εργασίας στο υπόβαθρο. Κοινώς, αναφέρεται ως producer. Η επόμενη έχει να κάνει με τη διαχείριση των εργασιών που προστίθενται σε αυτήν. Πρόκειται για τον consumer, ο οποίος ορίζει πως θα εκτελεστούν οι εργασίες για την εκάστοτε ουρά, με βάση τα δεδομένα που λαμβάνουν. Η τελευταία είναι ο listener, ο οποίος ορίζει τις ενέργειες που θα εκτελεστούν για κάθε event που λαμβάνεται. Παράδειγμα αποτελεί το «completed» event, το οποίο σηματοδοτεί την ολοκλήρωση μιας εργασίας.



Σχήμα 2.5: Απεικόνιση της σχέσης μεταξύ του Node.js, της mongoose και της mongoDB

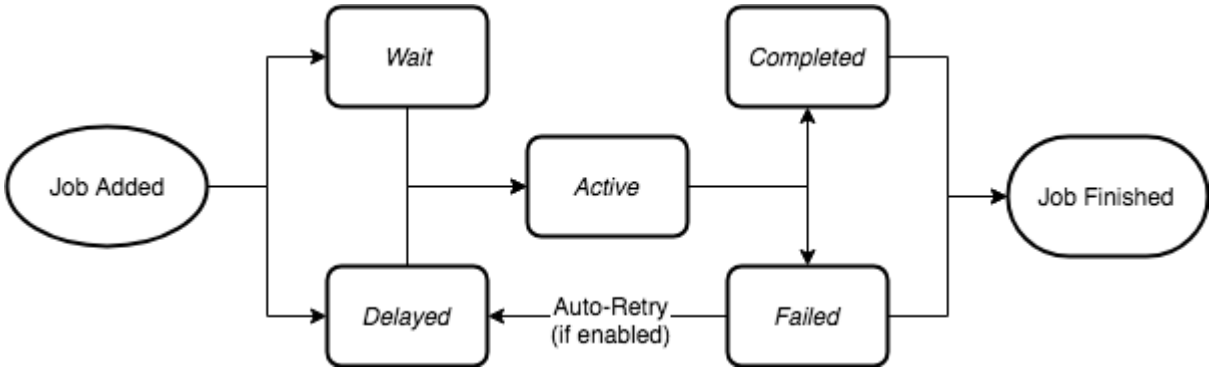
Ο κύκλος ζωής μιας εργασίας, από τη δημιουργία της μέχρι την ολοκλήρωση της, είναι αρκετά απλός και ευθύς. Αρχικά, μια εργασία, με την εισαγωγή της στην ουρά, μπορεί να έχει δύο πιθανές καταστάσεις, «wait» και «delayed». Η πρώτη υποδεικνύει ότι η εργασία έχει εισαχθεί στην ουρά και αναμένει εκτέλεση, ενώ η δεύτερη, ότι η εργασία αναμένει κάποιο timeout event ή απλά περιμένει να προαχθεί προς εκτέλεση. Ωστόσο, σε αυτήν την περίπτωση, δε θα εκτελεστεί άμεσα, αλλά θα τοποθετηθεί στην αρχή της ουράς, η οποία θα την παραδώσει προς εκτέλεση όταν βρεθεί διαθέσιμος worker [33]. Η επόμενη κατάσταση είναι η «active». Ως «active» ορίζονται οι εργασίες που εκτελούνται την τρέχουσα στιγμή. Μια «active» εργασία μπορεί στη συνέχεια να μπει σε κατάσταση «completed», αν ολοκληρωθεί επιτυχώς, ό,τι και να συνεπάγεται αυτό, και «failed», αν προκύψει κάποιο exception κατά την εκτέλεση της. Στο σχήμα 2.6 φαίνεται ο κύκλος ζωής μιας εργασίας στο BULL.

Υπάρχουν πέντε διακριτοί τύποι εργασιών, οι οποίοι έχουν να κάνουν με το πότε και με ποια σειρά θα εκτελεστούν από την ουρά:

- FIFO - First In First Out, ο οποίος είναι και ο προκαθορισμένος, με τις εργασίες να εκτελούνται με την σειρά που ελήφθησαν
- LIFO - Last In First Out, όπου οι εργασίες εκτελούνται με την αντίστροφη σειρά από αυτήν που ελήφθησαν
- Delayed, όπου η εργασία θα εκτελεστεί μετά από ένα καθορισμένο χρονικό διάστημα
- Prioritized, όπου οι εργασίες λαμβάνουν προτεραιότητα και αυτές με την μεγαλύτερη, εκτελούνται πριν από αυτές με την μικρότερη
- Repeatable, οι οποίες εκτελούνται επ' άπειρον ή μέχρι να επέλθει μια συγκεκριμένη ημερομηνία ή πλήθος εκτελέσεων, ανά ένα ορισμένο χρονικό διάστημα

Το BULL, για την λειτουργία του, βασίζεται στο Redis. Το Redis αποτελεί μια in-memory δομή αποθήκευσης δεδομένων. Μπορεί να χρησιμοποιηθεί ως βάση δεδομένων, cache, και message broker. Παρέχει πληθώρα τύπων δεδομένων, όπως strings, hash maps, lists, sets και sorted sets, bitmaps, hyperloglogs, geospatial indexes και streams [34].

Εργασίες οι οποίες είναι κοστοβόρες, τόσο σε πόρους όσο και σε χρόνο, αλλά και εργασίες που τείνουν να επαναλαμβάνονται ανά τακτικά χρονικά διαστήματα, είναι επιθυμητό να τρέχουν στο υπόβαθρο, ώστε να μην εισάγουν αδικαιολόγητες καθυστερήσεις στους χρήστες της εκάστοτε εφαρμογής. Στην παρούσα εφαρμογή, επιλέχθηκε το BULL ως ένας αξιόπιστος και ταχύς job scheduler, για τη δρομολόγηση και αποστολή e-mail και push notification.



Σχήμα 2.6: Διάγραμμα καταστάσεων εργασίας σε ουρά

2.2.1.7 Node-cron

Το node-cron αποτελεί ένα npm πακέτο για το node.js το οποίο επιτρέπει την εκτέλεση διεργασιών βάσει κάποιου προγράμματος. Χρησιμοποιείται για την αυτοματοποίησης επαναλαμβανόμενων διαδικασιών και βασίζεται στην σύνταξη του cron, ενός command-line εργαλείου της Unix οικογένειας των λειτουργικών συστημάτων. Μια cron εντολή αποτελείται από πέντε πεδία, όπου κάθε ένα αναφέρεται σε μια μονάδα χρόνου, με αυτές να είναι η μέρα, η ώρα, η μέρα του μήνα, ο μήνας και η μέρα της εβδομάδας. Κάθε πεδίο μπορεί να πάρει μια αριθμητική τιμή ή/και να συνδυαστεί με κάποιον τελεστή, όπως το « * » για όλες τις πιθανές τιμές του πεδίου, το « , » για την παράθεση λίστας τιμών, την « - » για τον ορισμό εύρους τιμών και το « / » για τον ορισμό βημάτων για κάποιο εύρος. Η σύνταξη μπορεί να είναι τόσο απλή όσο αυτή: “* * * * *”, η οποία ορίζει την εκτέλεση κάθε λεπτό, αλλά και σύνθετη όπως: “*/2,5 3-5 * 1,7,9 4/6,7”, οποία ορίζει την εκτέλεση στο 5^{ο} λεπτό της ώρας και κάθε 2 λεπτά από τις 3πμ μέχρι τις 5πμ του Ιανουαρίου, Ιουλίου και Σεπτεμβρίου, τις Κυριακές και Πέμπτες.

2.2.1.8 JSON Web Tokens

Το JSON Web Token αποτελεί έναν διακριτικό και URL-ασφαλή τρόπο αναπαράστασης ισχυρισμών προς μεταφορά μεταξύ δύο συμβαλλόμενων μερών [35]. Αποτελείται από τρία διακριτά μέρη: την κεφαλίδα, τα δεδομένα και την υπογραφή. Η κεφαλίδα περιλαμβάνει πληροφορίες σχετικά με το είδος του αλγορίθμου που θα χρησιμοποιηθεί για την κωδικοποίηση και τον τύπο του token. Τα δεδομένα προς μεταφορά βρίσκονται στο φορτίο του token σε μορφή JSON, ενώ η υπογραφή προκύπτει από την κεφαλίδα και το φορτίο σε συνδυασμό με ένα μυστικό. Το μυστικό δεν είναι υποχρεωτικό, όμως σε συνδυασμό με το γεγονός ότι τα περιεχόμενα του token είναι κωδικοποιημένα και όχι κρυπτογραφημένα, συνεπάγεται πως δεν προτείνεται να μεταφέρονται ευαίσθητα δεδομένα, αν το token, αυτό καθαυτό, δεν έχει κρυπτογραφηθεί. Το JWT, μετά την κωδικοποίηση, σχηματίζει μια

συμβολοσειρά, με το κάθε μέρος της να διαχωρίζεται με μια τελεία. Το JWT είναι stateless, καθώς δεν γνωρίζει, σε κανένα σημείο της ζωής του, ποιος το χρησιμοποιεί ούτε διατηρεί οποιαδήποτε σχετική πληροφορία. Συνεπώς, είναι ιδιαίτερα ταιριαστή λύση για χρήση σε RESTful APIs.

Η διαδικασία επαλήθευσης βασίζεται, κατά φύσιν, στη δομή του. Η κεφαλίδα και το φορτίο συνδυάζονται με το μυστικό, το οποίο είναι αποθηκευμένο στον server, για να δημιουργήσουν την υπογραφή. Έπειτα, σχηματίζεται το JWT string από τον συνδυασμό της κεφαλίδας, του φορτίου και της υπογραφής. Όταν ο server λαμβάνει ένα JWT, επαναλαμβάνει τη διαδικασία δημιουργίας της υπογραφής και τη συγκρίνει με αυτή που έλαβε στο JWT. Αν είναι ίδιες, τότε το token δεν έχει τροποποιηθεί από κάποιον κακόβουλο χρήστη και, συνεπώς, περιλαμβάνει αυτούσια την πληροφορία που προοριζόταν να λάβει ο server.

Ένα παράδειγμα χρήσης αποτελεί η πρόσβαση σε προστατευμένους πόρους κάποιας εφαρμογής, οι οποίοι μπορεί να εκτίθενται μέσω κάποιου endpoint. Πιο συγκεκριμένα, όταν ένας χρήστης συνδεθεί σε ένα σύστημα, αφού ο server επαληθεύσει τα διαπιστευτήρια του, του επιστρέφει ένα JWT, μοναδικό για αυτόν. Ο client, στη συνέχεια αποθηκεύει το JWT τοπικά και το χρησιμοποιεί σε κάθε αίτημα του προς τα προστατευμένα endpoints. Αν ο server επαληθεύσει την εγκυρότητα του token, του επιστρέφει τα δεδομένα.

2.2.1.9 Visual Studio Code

Το Visual Studio Code είναι ένας code editor της Microsoft. Υποστηρίζεται τόσο από Windows όσο και από Linux. Είναι εύκολα επεκτάσιμος, όσον αφορά τις γλώσσες προγραμματισμού που μπορεί να υποστηρίξει, αλλά και τα εργαλεία που προσφέρει, για την αύξηση της ευχρηστίας του με σκοπό την βελτίωση της εμπειρίας του χρήστη. Διαθέτει υποστήριξη για version control, με ενσωματωμένες εντολές του git καθώς και debugger για την αποσφαλμάτωση του κώδικα. Επίσης διαθέτει έξυπνη αυτόματη συμπλήρωση με βάση τον τύπο των μεταβλητών, τους ορισμούς των συναρτήσεων και των εισαγόμενων modules, με χρήση AI [7]. Το Visual Studio Code χρησιμοποιήθηκε για τη συγγραφή του κώδικα του backend.

2.2.1.10 Postman

Το Postman αποτελεί ένα εργαλείο για API testing. Είναι μια εύχρηστη εφαρμογή που διευκολύνει την ανάπτυξη των APIs. Παρέχει πληθώρα λειτουργιών, όπως τη δημιουργία και αποθήκευση request με συγκεκριμένα δεδομένα, τη διευκόλυνση ελέγχου των responses, συγγραφή documentation και διαμοιρασμού του API. Χρησιμοποιήθηκε σε όλα τα στάδια ανάπτυξης της παρούσας εφαρμογής καθώς αποτελεί έναν αξιόπιστο και γρήγορο τρόπο δοκιμής των endpoints.

2.2.2 Frontend

Ο όρος frontend αναφέρεται στα κομμάτια μιας εφαρμογής με τα οποία αλληλοεπιδρά άμεσα ο χρήστης. Είναι το σημείο επαφής του χρήστη με την εφαρμογή και μπορεί να αναφέρεται σε οποιαδήποτε συσκευή μέσω της οποίας είναι προσβάσιμη η εφαρμογή. Περιλαμβάνει όλες τις λειτουργίες που σχετίζονται με την ανάπτυξη της πλευράς του πελάτη. Παράδειγμα αποτελεί η κατασκευή της διεπαφής χρήστη μιας εφαρμογής και η μέριμνα σωστής λειτουργίας σε συσκευές, ανεξαρτήτως μεγέθους της οθόνης.

2.2.2.1 Android

Το Android αποτελεί ένα software stack ανοιχτού κώδικα, το οποίο δημιουργήθηκε για μια ευρεία ποικιλία συσκευών διαφόρων μεγεθών [8]. Αναπτύσσεται υπό την οικονομική αιγίδα της Google, μέσω

μιας κοινοπραξίας εταιρειών, εν ονόματι Open Handset Alliance. Όντας ανοιχτού κώδικα υπό την Apache άδεια, επιτρέπει σε οποιονδήποτε ενδιαφερόμενο να το προσαρμόσει και να το επεκτείνει. Ωστόσο, υπάρχουν συγκεκριμένες προϋποθέσεις και πρότυπα τα οποία πρέπει να πληροί μια υλοποίηση για να θεωρηθεί συμβατή. Για την αποτροπή ασυμβατοτήτων, το Android Open Source Project συντηρεί το Android Compatibility Program, το οποίο καθορίζει το τι σημαίνει να είναι μια συσκευή Android Compatible και τι απαιτείται από τους κατασκευαστές για να επιτευχθεί αυτό [8]. Αυτό συμβαίνει γιατί το Android, παρόλο που αποτελείται από πολλά μέρη, αντιμετωπίζεται ως ένα μεμονωμένο προϊόν και όχι μια προδιαγραφή ή συλλογή από κομμάτια [8].

Το Android ξεκίνησε το 2003 ως μια προσπάθεια ανάπτυξης ενός λειτουργικού συστήματος για ψηφιακές κάμερες από την Android Inc. Το 2004 όμως το ενδιαφέρον της εταιρείας άλλαξε και επικεντρώθηκε στις κινητές συσκευές. Το 2005 η εταιρεία εξαγοράστηκε από την Google και το 2007 ανακοινώθηκε η ίδρυση της κοινοπραξίας Open Handset Alliance για την προώθηση και ανάπτυξη του Android ως ένα δωρεάν, λειτουργικό σύστημα ανοιχτού κώδικα, με υποστήριξη για εφαρμογές τρίτων [9]. Το 2008 κυκλοφόρησε η πρώτη κινητή συσκευή που χρησιμοποιούσε το Android, ενώ μέχρι το 2012 είχε ξεπεράσει το iOS για να γίνει το πιο δημοφιλές λειτουργικό σύστημα για κινητές συσκευές. Εν έτη 2021, περίπου το 75% των κινητών συσκευών χρησιμοποιούν το Android ως λειτουργικό σύστημα [10]. Σήμερα, πέρα από κινητές συσκευές, μπορεί να βρεθεί σε tablet, smartwatches, ηχοσυστήματα, τηλεοράσεις και αυτοκίνητα.

2.2.2.1.1 Android Stack

Το software stack του Android αποτελείται από 5 κύρια συστατικά:

- **Applications**: Στην κορυφή βρίσκονται οι εφαρμογές. Το Android έρχεται με κάποιες προεγκατεστημένες εφαρμογές που παρέχουν λειτουργικότητα για βασικές ενέργειες, όπως το email, τα sms μηνύματα, το ημερολόγιο, τον browser, τις επαφές και άλλα [11]. Ο χρήστης έχει την ευχέρεια να αγνοήσει αυτές τις εφαρμογές και να εγκαταστήσει εφαρμογές τρίτων για να εξυπηρετήσει τις ανάγκες του. Επίσης, οι εφαρμογές μπορούν να προσφέρουν τη λειτουργικότητά τους και σε τρίτες εφαρμογές, αν αυτές απαιτούν την πραγματοποίηση κάποιας σχετικής ενέργειας, προσφέροντας ευελιξία και γλιτώνοντας χρόνο από τους developers.
- **Application Framework**: Το Application Framework παρέχει πρόσβαση σε πληθώρα services μέσω APIs γραμμένα σε Java. Τα APIs σχηματίζουν τα δομικά στοιχεία για να χτίσει κανείς μια Android εφαρμογή [11]. Τα σημαντικότερα services είναι τα:
- Activity Manager, το οποίο διαχειρίζεται το lifecycle της εφαρμογής και προσφέρει ένα κοινό navigation back stack για τα Activities [11]
- Content Providers, οι οποίοι επιτρέπουν σε εφαρμογές να αποκτήσουν πρόσβαση σε δεδομένα από άλλες εφαρμογές ή να μοιράσουν τα δικά τους δεδομένα [11]
- Resource Manager, ο οποίος παρέχει πρόσβαση σε πόρους όπως strings, ρυθμίσεις μορφοποίησης και layouts για τις διεπαφές χρήστη
- Notifications Manager, ο οποίος επιτρέπει στις εφαρμογές να προβάλλουν ειδοποιήσεις στον χρήστη [11]
- View System, το οποίο παρέχει εργαλεία για το χτίσιμο σύνθετων διεπαφών χρήστη
- **Native Libraries και Android Runtime**: Πολλά δομικά στοιχεία και services του Android, όπως το Android Runtime και το Hardware Abstraction Layer, απαιτούν βιβλιοθήκες γραμμένες σε C και C++ [11]. Για την πρόσβαση και την εκμετάλλευση τους, παρέχονται APIs. Για παράδειγμα, το OpenGl ES εκτίθεται μέσω του Java OpenGL API και προσφέρει υποστήριξη

για σχεδίαση 2D και 3D γραφικών [11]. Το Android Runtime είναι σχεδιασμένο να τρέχει πολλές εικονικές μηχανές σε συσκευές με λίγη μνήμη, εκτελώντας DEX αρχεία, έναν bytecode τύπο σχεδιασμένο ειδικά για το Android και βελτιστοποιημένο ώστε να έχει μηδαμινό αποτύπωμα στην μνήμη [11]. Από την έκδοση 5.0 και άνω, κάθε εφαρμογή τρέχει σε ξεχωριστή διεργασία και με το δικό της στιγμιότυπο του Android Runtime [11].

- **Hardware Abstraction Layer**: Το HAL προσφέρει διεπαφές που εκθέτουν τις δυνατότητες του υλικού της εκάστοτε συσκευής στο Application Framework [11]. Αποτελείται από πληθώρα από modules, όπου το καθένα υλοποιεί μια διακριτή διεπαφή για ένα συγκεκριμένο τύπο υλικού, όπως για παράδειγμα την κάμερα ή το Bluetooth [11].
- **Linux Kernel**: Το Kernel παρέχει ένα επίπεδο αφαιρετικότητας μεταξύ του υλικού και του υπόλοιπου stack. Το Android χρησιμοποιεί μια έκδοση του Linux kernel με μερικές επιπλέον προσθήκες, όπως το Low Memory Killer, για την πιο αποδοτική διαχείριση μνήμης, το wake locks, για τη διαχείριση της ενέργειας και άλλα σημαντικά στοιχεία για την εύρυθμη λειτουργία μιας mobile embedded πλατφόρμας [12].

2.2.2.1.2 Activity

Το Activity αποτελεί ένα από τα πιο σημαντικά συστατικά μιας Android εφαρμογής. Είναι μια, ως επί των πλείστων, ανεξάρτητη οντότητα και αναπαριστά μια μεμονωμένη οθόνη, αφού παρέχει το παράθυρο στο οποίο η εφαρμογή θα σχεδιάσει το UI [20]. Είναι παράλληλα το σημείο εισαγωγής μιας Android εφαρμογής, αφού δεν υπάρχει main μέθοδος. Μια εφαρμογή μπορεί να περιέχει από ένα ως πολλά Activities, τα οποία είναι ανεξάρτητα μεταξύ τους. Η ανεξαρτησία αυτή βασίζεται στο γεγονός ότι η πλοήγηση σε μια Android εφαρμογή δεν έχει απαραίτητα κάποια προκαθορισμένη ροή. Όταν μια εφαρμογή καλεί μια άλλη, δεν καλεί την εφαρμογή σαν σύνολο, αλλά ένα Activity της εφαρμογής εκείνης [20]. Η ελευθερία αυτή στην πλοήγηση προϋποθέτει πως υπάρχουν κατάλληλοι μηχανισμοί διαχείρισης των Activities, αφού αυτά μπορούν να εναλλάσσονται ανά πάσα στιγμή. Πιο συγκεκριμένα, ένα Activity έχει τέσσερις διακριτές καταστάσεις:

- Το Activity βρίσκεται στο προσκήνιο
- Το Activity βρίσκεται στο προσκήνιο, αλλά είναι εκτός focus
- Το Activity βρίσκεται στο υπόβαθρο, αλλά δεν έχει καταστραφεί
- Το Activity έχει καταστραφεί και για να ξαναπροβληθεί στον χρήστη, πρέπει να επαναδημιουργηθεί

Για τον χειρισμό αυτών των καταστάσεων, κάθε Activity διαθέτει έναν κύκλο ζωής – lifecycle – οποίος εκτίθεται στον προγραμματιστή μέσω κάποιων callback μεθόδων. Αναλυτικά, οι μέθοδοι αυτοί είναι οι εξής:

- onCreate, η οποία καλείται όταν δημιουργείται το Activity. Παρέχει επίσης και την προηγούμενη κατάσταση του, αν υπάρχει [21]. Ακολουθείται πάντα από την onStart.
- onRestart, η οποία καλείται αφού το Activity έχει σταματήσει και πριν ξεκινήσει ξανά [21]. Ακολουθείται πάντα από την onStart
- onStart, η οποία καλείται όταν το Activity γίνεται εμφανές στον χρήστη. Ακολουθείται από την onResume, αν το Actvity πηγαίνει στο προσκήνιο ή την onStop, αν μπαίνει σε κατάσταση hidden [21]
- onResume, η οποία καλείται όταν το Activity θα ξεκινήσει να αλληλοεπιδρά με τον χρήστη [21]. Ακολουθείται πάντα από την onPause.

- onPause, η οποία καλείται όταν το Activity χάσει τη θέση του στο προσκήνιο, δεν είναι πλέον focusable ή πριν την μετάβαση σε κατάσταση stopped/hidden ή destroyed [21]. Ακολουθείται από την onResume, αν το Activity επιστρέψει στο προσκήνιο ή την onStop, αν δεν είναι πλέον εμφανές στον χρήστη [21].
- onStop, η οποία καλείται όταν το Activity δεν είναι πλέον εμφανές στον χρήστη. Ακολουθείται από την onRestart, αν το Activity θα επιστρέψει στο προσκήνιο για να αλληλοεπιδράσει με τον χρήστη ή την onDestroy, αν καταστραφεί [21].
- onDestroy, η οποία αποτελεί την τελευταία κλήση πριν καταστραφεί το Activity. Αυτό μπορεί να συμβεί είτε επειδή καταστρέφεται χειροκίνητα από τον προγραμματιστή ή επειδή το σύστημα την καταστρέφει προσωρινά για να εξοικονομήσει χώρο στη μνήμη.

2.2.2.1.3 Fragment

Το Fragment είναι ένα κομμάτι του UI ή της συμπεριφοράς μιας εφαρμογής, το οποίο μπορεί να τοποθετηθεί μέσα σε ένα Activity [22]. Ένα Fragment είναι άρρηκτα και πλήρως συνδεδεμένο με ένα Activity καθώς δεν μπορεί να ορισθεί και να σταθεί μόνο του. Ένα Activity μπορεί να περιλαμβάνει πολλά Fragments, το καθένα από τα οποία μπορούν να εξυπηρετούν μια συγκεκριμένη λειτουργικότητα, που ανήκει εννοιολογικά και θεματικά στο Activity. Τα Fragments βοηθούν στην τμηματοποίηση και στον διαχωρισμό του UI, και κατ' επέκταση, στην επαναχρησιμοποίηση του, προάγοντας την οργάνωση των UI οντοτήτων, ενώ διευκολύνει και την μεταφορά δεδομένων μεταξύ διαφόρων οθονών καθώς τα Fragments μπορούν να έχουν ένα κοινό Activity, τα δεδομένα του οποίοι μπορούν να μοιραστούν. Τα Fragments έχουν και αυτά το δικό τους lifecycle, το οποίο όμως εξαρτάται από το Activity στο οποίο ανήκουν. Αν το σχετικό Activity καταστραφεί, τότε καταστρέφονται και όλα τα σχετιζόμενα Fragments. Η διαχείριση των Fragments, όσον αφορά τον κύκλο ζωής τους, γίνεται, όπως και στα Activities, μέσω callback μεθόδων. Πιο συγκεκριμένα, οι μέθοδοι αυτοί είναι οι εξής:

- onAttach, η οποία καλείται μόλις το Fragment συσχετιστεί με κάποιο Activity
- onCreate, η οποία καλείται για να πραγματοποιήσει την αρχικοποίηση του Fragment [22]
- onCreateView, η οποία δημιουργεί και επιστρέφει την ιεραρχία των view (στοιχεία του UI) που είναι συσχετισμένα με το Fragment [22]
- onActivityCreated, η οποία ενημερώνει το Fragment πως το Activity έχει ολοκληρώσει την onCreate μέθοδο [22]
- onViewStateRestored, η οποία ενημερώνει το Fragment πως η αποθηκευμένη κατάσταση της ιεραρχίας των view, έχει επαναφερθεί [22]
- onStart, η οποία κάνει το Fragment εμφανές στον χρήστη [22]
- onResume, η οποία καθιστά το Fragment αλληλοεπιδράσιμο από τον χρήστη [22]

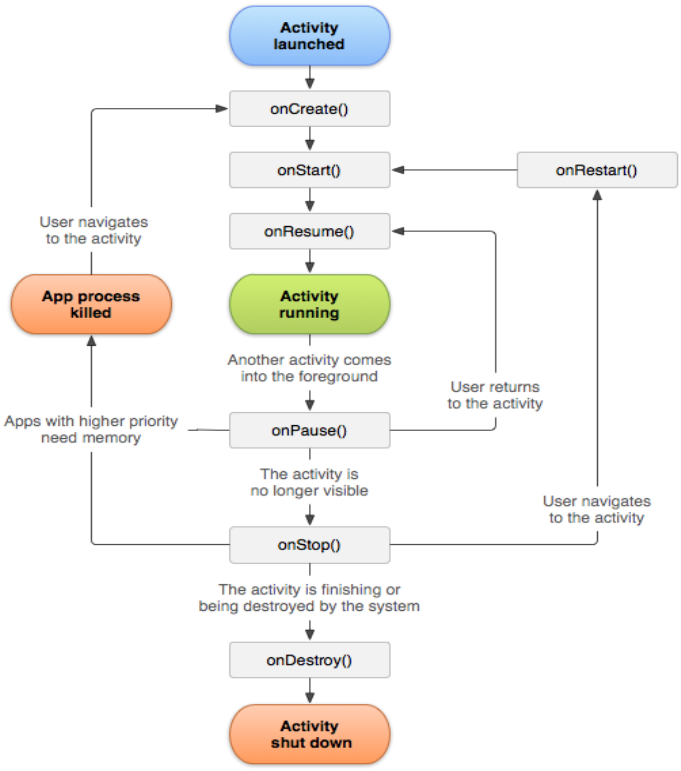
Αντίστοιχα, όταν ένα Fragment σταματήσει να χρησιμοποιείται, περνάει από μια αντίστροφη σειρά από callbacks:

- onPause, κατά την οποία το Fragmen δεν αλληλοεπιδρά πλέον με τον χρήστη
- onStop, όπου το Fragment δεν είναι πλέον εμφανές στον χρήστη
- onDestroyView, η οποία επιτρέπει στο Fragment να καθαρίσει τους πόρους που σχετίζονται με τα view του
- onDestroy, που καλείται για το τελικό καθάρισμα του Fragment
- onDetach, η οποία καλείται αμέσως πριν το Fragment σταματήσει να συσχετίζεται με το Activity του [22]

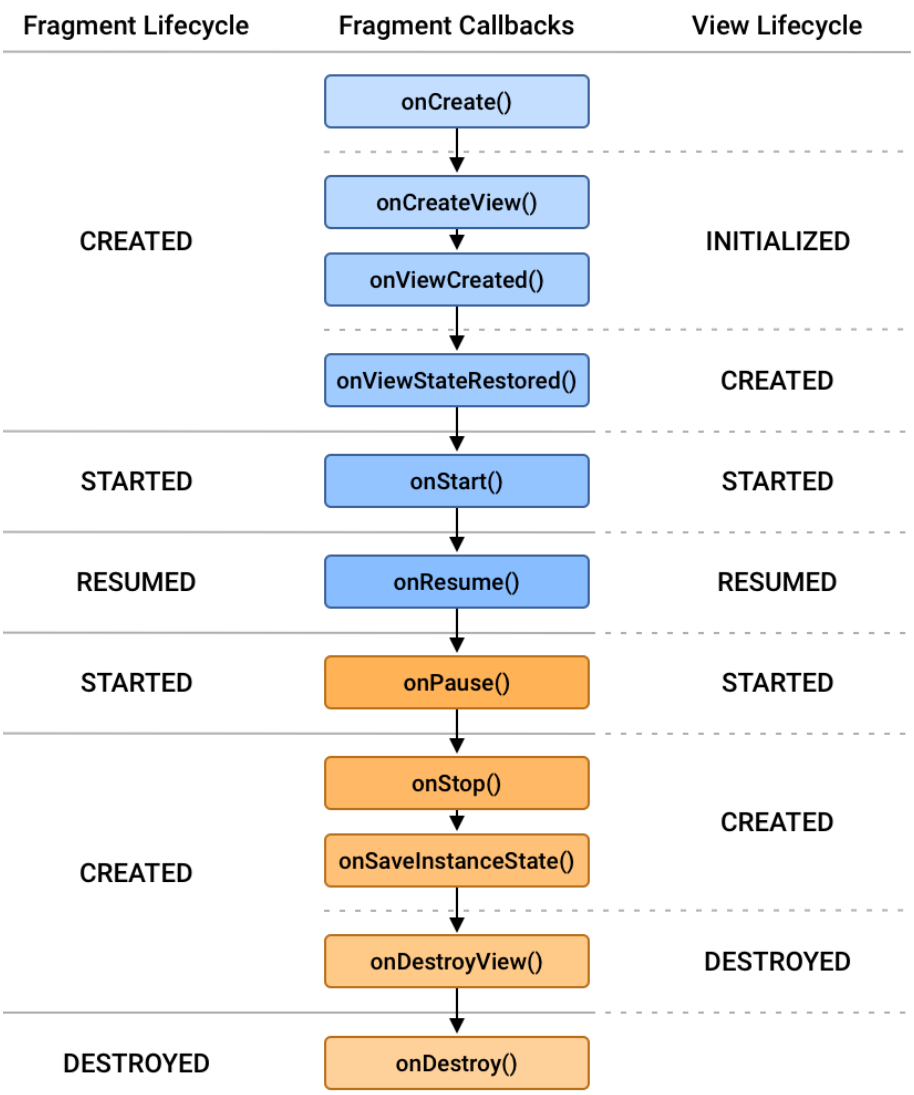
2.2.2.1.4 Observer Pattern και LiveData

Το Observer Pattern είναι ένα σχεδιαστικό πρότυπο το οποίο εγκαθιδρύει μια ένα-προς-πολλά εξάρτηση μεταξύ αντικειμένων [23]. Επιτρέπει τον ορισμό ενός μηχανισμού ενημέρωσης πολλών αντικειμένων, όταν συμβαίνουν αλλαγές σε κάποιο αντικείμενο το οποίο παρακολουθούν. Το αντικείμενο, η κατάσταση του οποίου παρακολουθείται, καλείται observable, ενώ τα αντικείμενα που τον παρακολουθούν, observers.

Τα LiveData είναι μια lifecycle-aware κλάση η οποία μπορεί να επιδεχτεί παρακολούθησης από άλλα στοιχεία, όπως Activities και Fragments. Όντας lifecycle-aware, γνωρίζουν την κατάσταση των observer τους, διασφαλίζοντας πως θα ενημερώσουν μόνο αυτούς που βρίσκονται σε μια ενεργή κατάσταση στο lifecycle τους [24], όπου ενεργή θεωρείται η Started και η Resumed. Η χρήση των LiveData έχει πληθώρα πλεονεκτημάτων. Αρχικά, ακολουθώντας το observer pattern, ενημερώνουν τους observers για τις επικείμενες αλλαγές στα δεδομένα και μπορούν να διασφαλίσουν πως τα στοιχεία του UI απεικονίζουν την τρέχουσα κατάσταση τους. Επίσης, είναι λιγότερο πιθανό να προκύψουν προβλήματα με την μνήμη, αφού οι observers δεσμεύονται από lifecycle αντικείμενα, τα οποία αποδεσμεύονται όταν το αντίστοιχο lifecycle καταστρέφεται [24]. Επιπλέον, δεν προκύπτουν αναπάντεχες διακοπές λειτουργίας της εφαρμογής εξαιτίας σταματημένων Activities, καθώς, αν το lifecycle ενός observer είναι ανενεργό, τότε δε λαμβάνει LiveData events [24]. Επίσης, εξαλείφονται και αντίστοιχα προβλήματα σχετιζόμενα με την αλλαγή στην παραμετροποίηση της συσκευής, όπως την περιστροφή της οθόνης, αφού το Activity ή το Fragment, με την επαναδημιουργία του, θα λάβει τα πιο πρόσφατα δεδομένα. Τέλος, τα LiveData αυτοματοποιούν τον χειρισμό των lifecycle event, αφού τα στοιχεία του UI απλά παρατηρούν τα σχετιζόμενα δεδομένα και δεν σταματούν την παρατήρηση. Τα LiveData διαχειρίζονται αυτόματα αυτές τις ενέργειες, καθώς αντιλαμβάνονται τις αλλαγές στην κατάσταση του lifecycle όσο παρατηρούν [24].



Σχήμα 2.7: Κύκλος ζωής του Activity



Σχήμα 2.8: Κύκλος ζωής του Fragment

2.2.2.1.5 MVVM Αρχιτεκτονική

Η αρχική προσέγγιση στη δημιουργία Android εφαρμογών ήταν η τοποθέτηση, τόσο του business logic όσο και των UI στοιχείων, καθώς και οποιουδήποτε άλλου κομματιού κώδικα ήταν απαραίτητος, στην κλάση του Activity [26]. Ως αποτέλεσμα, τα Activities ήταν εξαιρετικά περίπλοκα και δύσκολο να κατανοηθούν και να τροποποιηθούν, ενώ περιλάμβαναν αλληλεξαρτήσεις μεταξύ στοιχείων του κώδικα τους που αύξανε τον κίνδυνο δημιουργίας απρόβλεπτων προβλημάτων. Η αυξημένη πολυπλοκότητα δυσχέραινε τον χειρισμό των lifecycle events ενώ δυσκόλευε και το unit testing τους. Για τους λόγους αυτούς, η google εισήγαγε την Android Architecture Components βιβλιοθήκη, η οποία περιλαμβάνει τα LiveData και τα ViewModels [26], στα οποία βασίζεται η MVVM αρχιτεκτονική για την εφαρμογή και υλοποίησή της.

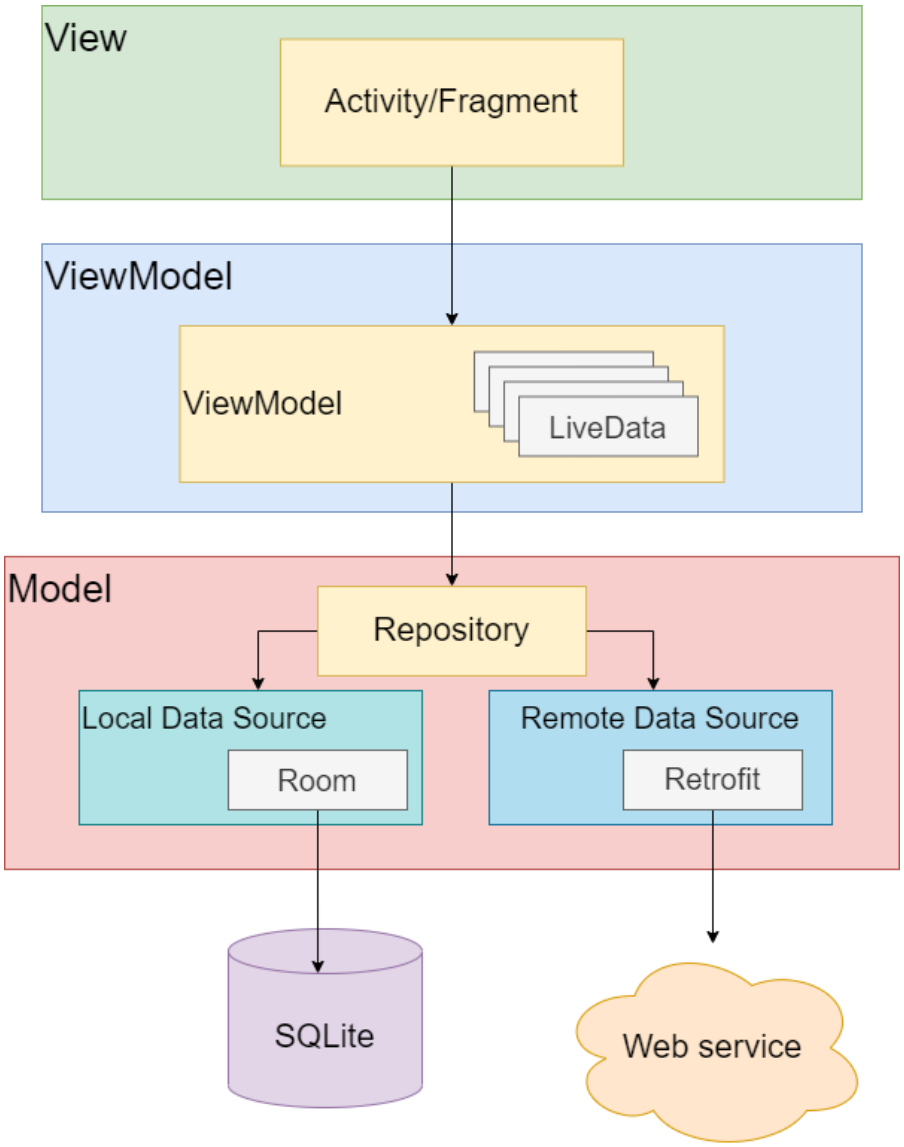
Η Model-View-Viewmodel αρχιτεκτονική ή MVVM για συντομία, αποτελεί ένα σχεδιαστικό πρότυπο το οποίο αποσκοπεί στον διαχωρισμό των λειτουργιών κάθε συστατικού που την απαρτίζει καθώς και την μεταξύ τους απεξαρτητοποίηση. Η εν λόγω τμηματοποίηση αποτυπώνεται στην αποσυσχέτιση των UI στοιχείων από το business logic μιας εφαρμογής καθώς και στην αποσυσχέτιση του τελευταίου από τα database operations.

Όπως προδίδει και η ονομασία του, το MVVM αποτελείται από τρία διακριτά επίπεδα, το View, το ViewModel και το Model. Το View βρίσκεται στην κορυφή. Αναπαριστά το UI της εφαρμογής και σκοπός του είναι να ενημερώνει το ViewModel για τις ενέργειες του χρήστη [27]. Δεν περιέχει καθόλου business logic και παρακολουθεί το ViewModel. Το ViewModel με τη σειρά του αποτελεί τον ενδιάμεσο μεταξύ του View και του Model. Είναι View agnostic, που σημαίνει πως δεν γνωρίζει και δεν χρειάζεται να γνωρίζει ποιο View το χρησιμοποιεί. Σκοπός του είναι να λαμβάνει δεδομένα από το Model και να τα κάνει διαθέσιμα στα Views μέσω των obsrevables. Το μόνο που αφορά ένα ViewModel είναι να διατηρεί ακριβείς αναφορές στα δεδομένα που χρειάζεται το View. Έτσι, ακόμα και αν ο χρήστης κλείσει την εφαρμογή και την ανοίξει μετά από αρκετές ώρες, θα δει ακριβώς τα ίδια δεδομένα με αυτά από όταν την έκλεισε [28]. Πιο συγκεκριμένα, η ViewModel κλάση, που είναι διαθέσιμη μέσω της Android Architecture Components βιβλιοθήκης, είναι σχεδιασμένη για να αποθηκεύει και να διαχειρίζεται δεδομένα σχετικά με το UI, με τέτοιο τρόπο ώστε να αντιλαμβάνεται τις καταστάσεις του lifecycle των UI controllers, όπως τα Activities και τα Fragments [29]. Οι αλλαγές αυτές μπορούν να επέλθουν είτε κατόπιν κάποιας ενέργειας του χρήστη ή λόγω άλλων ανεξάρτητων γεγονότων, μέσω του Android Framework, το οποίο διαχειρίζεται τις εν λόγω διεργασίες. Αν κάποιος UI controller καταστραφεί ή επαναδημιουργηθεί, όλα τα δεδομένα που διατηρούσε θα χαθούν, με αποτέλεσμα, ενδεχομένως, να χρειαστεί να επαναληφθούν αιτήματα σε πηγές δεδομένων, τα οποία δε θα έπρεπε εξαρχής να διαχειρίζονται από αυτόν τον controller. Γίνεται ξεκάθαρο πως ο ρόλος του ViewModel ως ενδιάμεσος, αποσυμφορεί τους UI controllers από περιττές ευθύνες, ενώ παράλληλα προάγει την αρχή του separation of concerns, τόσο στον χειρισμό του UI όσο και στον χειρισμό των database operations.

Το Model διαχειρίζεται όλες τις ενέργειες που σχετίζονται με την πρόσβαση στα δεδομένα και το business logic της εφαρμογής. Δεν είναι μια διακριτή οντότητα. Αποτελείται από πολλά στοιχεία όπως τα μοντέλα των αντικειμένων για την διαχείριση των απαντήσεων από πηγές δεδομένων, τα APIs για την επικοινωνία με τις πηγές αυτές, services που ενεργούν και επεξεργάζονται τα ληφθέντα δεδομένα και μεθόδους για τα αιτήματα αυτά καθαυτά. Το Model παρέχει τα δεδομένα προς παρουσίαση στο ViewModel, το οποίο με τη σειρά του τα προσφέρει στο View. Συχνά, υλοποιείται και το Repository Pattern. Ο ρόλος του Repository είναι να προσφέρει έναν τρόπο πρόσβασης σε δεδομένα που συγκεντρώνονται από διάφορες πηγές, όπως από εξωτερικά APIs, από την τοπική βάση ή από την cache, με σκοπό την απεξαρτητοποίηση τους από την υπόλοιπη εφαρμογή.

2.2.2.1.6 View Binding

Το View Binding αποτελεί ένα σχετικά νέο feature του Android Studio, το οποίο αντικαθιστά τη μέθοδο findViewById, η οποία χρησιμοποιείται για τη χειραγώγηση των views, και δίνει λύση σε προβλήματα που μπορούν να προκύψουν με τη χρήση της. Το View Binding καθιστά τη γραφή κώδικα, με σκοπό την αλληλεπίδραση με τα views, ευκολότερη, παράγοντας μια binding κλάση για κάθε XML Layout αρχείο που βρίσκεται στο συγκεκριμένο module του project [30]. Η εκάστοτε κλάση περιλαμβάνει αναφορές σε όλα τα views για τα οποία έχει δηλωθεί ID στο αντίστοιχο layout. Το View Binding, σε αντίθεση με το findViewById, προσφέρει null safety και type safety - null safety καθώς δεν υπάρχει κίνδυνος παραγωγής null pointer exception, αφού δημιουργεί άμεσες αναφορές στα υπάρχοντα views, ενώ σημαίνει ως nullable τα πεδία που ενδέχεται να μην έχουν τιμή, και type safety, αφού κάθε πεδίο που παράγεται έχει τον ίδιο τύπο με το view που αναφέρει, εξαλείφοντας τον κίνδυνο παραγωγής class cast exception [31].



Σχήμα 2.9: Διάγραμμα αρχιτεκτονικής MVVM

2.2.2.1.7 Data Binding

Το Data Binding είναι μια υποστηρικτική βιβλιοθήκη, η οποία επιτρέπει τη σύνδεση UI στοιχείων των layouts με πηγές δεδομένων της εφαρμογής, μέσω δηλωτικής μορφής, με τη χρήση annotation, αντί προγραμματιστικά [31]. Αποτελεί και αυτό μια προτεινόμενη εναλλακτική του findViewById, καθώς απλοποιεί και καθιστά πιο συντηρήσιμο τον κώδικα των Activities και των Fragments, αφού ελαχιστοποιεί τις σχετικές με το UI κλήσεις εντός αυτών. Επιπλέον, με τη χρήση του two-way Data Binding, γίνεται δυνατή η παρακολούθηση και η μεταβολή της τιμής ενός layout attribute, κατόπιν αλλαγής της κατάστασης του.

Όπως και το View Binding, το Data Binding παράγει binding κλάσεις για την αναφορά των views, ωστόσο το πρώτο προσφέρει γρηγορότερο compilation, αφού δεν περιλαμβάνει επεξεργασία των annotations. Αντίθετα, το View Binding υστερεί σε πιο προχωρημένα features, καθώς δεν υποστηρίζει two-way Binding, ούτε layout μεταβλητές και εκφράσεις, με αποτέλεσμα να μην μπορεί να χρησιμοποιηθεί για τη δήλωση δυναμικού UI απευθείας από ένα XML αρχείο [31].

2.2.2.1.8 Notification Channels

Κάθε εισερχόμενη ειδοποίηση, πρέπει να ανατίθεται σε ένα κανάλι. Η παραμετροποίηση του καναλιού αφορά την οπτική και ακουστική συμπεριφορά όλων των ειδοποιήσεων που το χρησιμοποιούν. Η συμπεριφορά αυτή ορίζεται κατά τη δημιουργία του, μέσω της παραμέτρου «importance» και μπορεί να λάβει τέσσερις διακριτές τιμές:

- IMPORTANCE_HIGH, όπου η ειδοποίηση παράγει ήχο και εμφανίζεται ως heads-up, με την ειδοποίηση να καλύπτει ένα μέρος της οθόνης
- IMPORTANCE_DEFAULT, όπου η ειδοποίηση παράγει μόνο ήχο και εμφανίζεται στο status bar
- IMPORTANCE_LOW, όπου η ειδοποίηση δεν παράγει ήχο, αλλά εμφανίζεται στο status bar
- IMPORTANCE_MIN, όπου η ειδοποίηση δεν παράγει ήχο, ούτε εμφανίζεται στο status bar

Ο χρήστης, μπορεί να πλοηγηθεί στις ρυθμίσεις της εφαρμογής και να διαχειριστεί τα κανάλια μεμονωμένα, ορίζοντας αν επιθυμεί να λαμβάνει ειδοποιήσεις από το κάθε κανάλι, καθώς και αν επιθυμεί αυτές να έχουν δόνηση ή να εμφανίζονται όσο η συσκευή είναι κλειδωμένη.

2.2.2.1.9 RecyclerView

Το RecyclerView κάνει δυνατή την προβολή συνόλων δεδομένων στην οθόνη, δημιουργώντας δυναμικά τα UI στοιχεία, με βάση τα εν λόγω δεδομένα, ενώ για την εμφάνιση τους ορίζεται ένα layout που κατασκευάζεται από τον χρήστη. Όταν ένα στοιχείο του σταματήσει να εμφανίζεται στην οθόνη, λόγω scroll του χρήστη, δεν καταστρέφεται, αλλά ανακυκλώνεται προς χρήση από το επόμενο στοιχείο που θα εισέλθει στην οθόνη. Για την λειτουργία του, απαιτείται η δήλωση του στο εκάστοτε layout αρχείο, όπως κάθε UI στοιχείο. Κάθε στοιχείο του RecyclerView είναι τύπου ViewHolder, και συνδέεται με τα εκάστοτε δεδομένα με την κλήση της onBindViewHolder() μεθόδου, ενώ για τη δημιουργία και τη διαχείριση των ViewHolders, χρησιμοποιείται η κλάση RecyclerView.Adapter. Τέλος, η διάταξη των στοιχείων της λίστας καθορίζεται από τον LayoutManager που ορίζεται, με τις επιλογές να περιλαμβάνουν τη χρήση ενός

- LinearLayoutManager, για την διάταξη σε μια μονοδιάστατη λίστα
- GridLayoutManager, για τη διάταξη σε ένα δισδιάστατο πλέγμα
- StaggeredGridLayoutManager, για τη διάταξη σε ένα δισδιάστατο πλέγμα, χωρίς τα στοιχεία να έχουν απαραίτητα τα ίδια μήκη και πλάτη

2.2.2.1.10 Android Studio

Το Android Studio είναι το κατεξοχήν IDE για την ανάπτυξη Android εφαρμογών. Είναι βασισμένο στο IntelliJ IDEA της Jet Brains, και περιλαμβάνει πληθώρα στοιχείων που αυξάνουν την αποδοτικότητα και βελτιώνουν, με ουσιώδη τρόπο, την εμπειρία προγραμματισμού εφαρμογών. Πιο συγκεκριμένα, περιλαμβάνει ενσωματωμένο emulator για την εκτέλεση της Android εφαρμογής, καθώς ο χρήστης μπορεί να επιλέξει από μια πληθώρα συσκευών με ποικιλία προδιαγραφών για να προσομοιώσει, ενώ δίνεται η δυνατότητα ανάπτυξης και δοκιμής εφαρμογών για όλες τις υποστηριζόμενες συσκευές. Επίσης, περιλαμβάνει έναν Layout Editor, ένα γραφικό περιβάλλον σχεδιασμού του UI, όπου ο χρήστης μπορεί εύκολα να χτίσει το layout της εφαρμογής. Μέσω drag ‘n drop λειτουργικότητας, μπορεί να σύρει στοιχεία του UI από τη διαθέσιμη παλέτα στοιχείων, και να τα τοποθετήσει στον editor, δηλώνοντας παράλληλα τις μεταξύ τους σχέσεις, χωρίς να γράψει XML κώδικα με το χέρι, αφού αυτός παράγεται αυτόματα με κάθε ενέργεια του. Επιπλέον, προσφέρει εργαλεία για την παρακολούθηση των

πόρων του συστήματος, δίνοντας χρήσιμες πληροφορίες για το πόση μνήμη χρησιμοποιεί η εφαρμογή, πόση μπαταρία καταναλώνει, πως χρησιμοποιεί τους δικτυακούς πόρους αλλά και τον επεξεργαστή. Επιπροσθέτως, το Android Build System κάνει compile τους πόρους της εφαρμογής και τον πηγαίο κώδικα και μέσω αυτών, χτίζει APKs ή Android App Bundles, τα οποία μπορεί κανείς να τα τεστάρει, να τα υπογράψει, να τα κάνει deploy και να τα κατανείμει [32]. Για τη διαχείριση και την αυτοματοποίηση αυτής της διαδικασίας, χρησιμοποιείται το Gradle. Επίσης, μέσω του APK Analyzer, παρέχει πληροφορίες σχετικά με την σύνθεση του APK. Η χρήση του APK Analyzer μπορεί να μειώσει τον χρόνο που ξοδεύεται στην αποσφαλμάτωση προβλημάτων σχετικά με τα DEX αρχεία ή τους πόρους της εφαρμογής, ενώ βοηθά στην μείωση του μεγέθους του APK [32].

2.2.2.2 Kotlin

Η Kotlin είναι μια δωρεάν, ανοιχτού κώδικα, statically typed γλώσσα προγραμματισμού γενικού σκοπού, αρχικά σχεδιασμένη για το Java Virtual Machine και το Android, που συνδυάζει χαρακτηριστικά του αντικειμενοστραφούς και του συναρτησιακού προγραμματισμού [13]. Η ανάπτυξη της ξεκίνησε από την JetBrains το 2011, ενώ το 2019, η Google ανακοίνωσε πως Kotlin θα είναι η προτεινόμενη γλώσσα για την ανάπτυξη Android εφαρμογών.

Η ανάπτυξη της ξεκίνησε με γνώμονα την παραγωγή μιας καλύτερης έκδοσης της Java, ωστόσο παραμένει πλήρως διαλειτουργική με αυτήν. Έχει τόσες ομοιότητες με την Java όσες και διαφορές, αλλά ταυτόχρονα λύνει και αρκετά προβλήματα της. Πιο συγκεκριμένα, διαθέτει type inference για τις μεταβλητές και δεν υποχρεώνει τις μεθόδους – πλέον συναρτήσεις – να είναι μέλη μιας κλάσης. Στην Kotlin, οι συναρτήσεις μπορούν να δηλωθούν στην κορυφή κάποιο αρχείου (top-level functions), τοπικά, εντός άλλων συναρτήσεων, σαν συνάρτηση-μέλος σε μια κλάση ή αντικείμενο και σαν extension συνάρτηση [13], η οποία δίνει τη δυνατότητα επέκτασης μια κλάσης με νέα λειτουργικότητα χωρίς να χρειάζεται να την κληρονομήσει. Πέρα από τα top-level functions, υποστηρίζει higher-order functions, anonymous functions, lambdas, inline functions, closures, tail recursion, και generics, δηλαδή, όλα τα πλεονεκτήματα μιας συναρτησιακής γλώσσας προγραμματισμού [13]. Επίσης, ίσως η σημαντικότερη βελτίωση είναι η αντιμετώπιση των NullPointerExceptions. Αυτό το πετυχαίνει απαγορεύοντας την null τιμή στους βασικούς τύπους και εισάγοντας nullable τύπους. Αν ένα String λάβει null τιμή, προκύπτει compilation error. Η null τιμή μπορεί να επιτραπεί δηλώνοντας τον τύπο, ακολουθούμενο με ένα λατινικό ερωτηματικό (? – String?). Επίσης, απλοποιεί τους ελέγχους για null τιμές. Πιο συγκεκριμένα, η έκφραση if (b != null) b.length, μπορεί να αντικατασταθεί με το b?.length [13]. Επιπλέον, εισάγει τον ασύγχρονο προγραμματισμό με τη χρήση των coroutines. Τα coroutines μοιάζουν εννοιολογικά με τα threads, καθώς τρέχουν ένα κομμάτι κώδικα παράλληλα με τον υπόλοιπο. Ωστόσο, τα coroutines δεν δεσμεύουν ένα συγκεκριμένο thread, αλλά μπορούν να αναστείλουν την εκτέλεση τους και να την συνεχίσουν σε ένα άλλο [14]. Τέλος, χάρις στα χαρακτηριστικά της, όπως τον χειρισμό των null και το type inference, ο κώδικας σε Kotlin είναι κατά κύριο λόγο πιο μικρός, χωρίς να χάνει από άποψη λειτουργικότητας. Η τυπική, προσεγγιστική εκτίμηση υποδεικνύει πως μια εφαρμογή γραμμένη σε Kotlin έχει κατά 40% λιγότερες γραμμές κώδικα από την ίδια σε Java [13].

2.2.2.3 Retrofit

Το Retrofit είναι ένας HTTP client για το Android. Χρησιμοποιεί την OkHttp βιβλιοθήκη, η οποία είναι υπεύθυνη για οποιεσδήποτε, χαμηλού επιπέδου δικτυακές εργασίες, για caching και για τη διαχείριση των αιτημάτων και των απαντήσεων [15]. Το Retorfit δίνει την ελευθερία επιλογής του converter για το serialization/deserialization σε/από, αντίστοιχα, JSON μορφή, ωστόσο η GSON βιβλιοθήκη είναι από τις πιο δημοφιλείς. Το Retrofit επιτρέπει τη δημιουργία interface σε Java και Kotlin, για τη διαχείριση

των αιτημάτων στα επιθυμητά endpoints, με τη χρήση annotations, που υποδηλώνουν το είδος του αιτήματος, τη διαδρομή, τις παραμέτρους, τις κεφαλίδες και το σώμα του. Οι απαντήσεις μπορούν εύκολα να αντιστοιχιστούν σε Java/Kotlin αντικείμενα και να χρησιμοποιηθούν στην εκάστοτε εφαρμογή. Επιπλέον, παρέχει μεθόδους για την ασύγχρονη, αλλά και σύγχρονη, αποστολή και λήψη των αιτημάτων.

2.2.2.4 Google Maps Platform

Το Google Maps Platform παρέχει δεδομένα στους προγραμματιστές, από τις υπηρεσίες Maps, Routes και Places της Google, μέσω APIs και SDKs. Κάνει εφικτή τη δυνατότητα προβολής χαρτών και αλληλεπίδρασης με αυτούς, τη λήψη δεδομένων τοποθεσίας και διαδρομών μεταξύ σημείων, ενώ καθιστά δυνατή την κατασκευή location-aware εφαρμογών, που ανταποκρίνονται με ουσιώδη για τον χρήστη τρόπο, στις τοπικές επιχειρήσεις και άλλες τοποθεσίες κοντά στη συσκευή του χρήστη [16]. Όλες οι υπηρεσίες της Google παρέχονται επί πληρωμή μέσω ενός μηνιαίου συνδρομητικού πλάνου κλιμακωτής χρέωσης. Ωστόσο, παρέχει υπηρεσίες αξίας 200€ μηνιαίως, για χρήση σε οποιαδήποτε υπηρεσία. Για την χρήση των SDKs και των APIs παρέχεται ένα API key, το οποίο φέρει και τον ρόλο παρακολούθησης των αιτημάτων για λόγους χρέωσης, καθώς αυτό είναι το μέσο ελέγχου χρήσης των υπηρεσιών. Παράδειγμα αποτελούν οι Dynamic Maps, όπου η χρέωση καταγράφεται κάθε φορά που φορτώνεται ένας χάρτης, όχι όμως όταν ο χρήστης αλληλοεπιδρά με αυτόν μέσω των στοιχείων ελέγχου του, όπως το zoom ή την αλλαγή layer. Το ποσό που διατίθεται κάθε μήνα στον προγραμματιστή, αντιστοιχεί σε 28.500 φορτώματα χάρτη, τα οποία είναι αρκετά για εφαρμογές με λίγη κίνηση ή/και μικρό πλήθος χρηστών.

Στην Android εφαρμογή της παρούσας εργασίας, χρησιμοποιήθηκαν τα Maps και Places SDKs για την προβολή επιχειρήσεων στον χάρτη, καθώς και το Static Maps API, για την λήψη στατικών εικόνων συγκεκριμένων τμημάτων του χάρτη.

2.2.3 GitHub

Το github προσφέρει cloud-based υπηρεσίες για repository hosting με τη χρήση του git [19], ένα open source version control λογισμικό, που επιτρέπει την παράλληλη και ταυτόχρονη εργασία πολλών χρηστών στο ίδιο project. Μέσω του github, οι χρήστες μπορούν να διαχωρίσουν τον κώδικα τους, να καταγράφουν τις αλλαγές στο project και να διαχειριστούν προβλήματα που προκύπτουν, ταυτόχρονα και ανεξάρτητα μεταξύ τους. Έτσι, αυξάνεται η παραγωγικότητα και ενθαρρύνεται η συνεργατικότητα.

2.3 Επίλογος

Στο κεφάλαιο αυτό έγινε εκτενής αναφορά στις τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της προτεινόμενης εφαρμογής. Αρχικά, έγινε διαχωρισμός των τεχνολογιών σε δύο διακριτές κατηγορίες, το frontend και το backend. Στη συνέχεια, για κάθε κατηγορία, έγινε λεπτομερής ανάλυση των βασικότερων χαρακτηριστικών των επιμέρους τεχνολογιών που χρησιμοποιήθηκα για την υλοποίηση της Android εφαρμογής και του server. Το backend της εφαρμογής χτίστηκε με το node.js, με χρήση του express.js ως framework, ενώ για τη βάση χρησιμοποιήθηκε η mongoDB και η mongoose, για τη διευκόλυνση της μοντελοποίησης των οντοτήτων. Το frontend αποτελεί μια εφαρμογή για Android συσκευές, γραμμένη σε Kotlin. Για το κομμάτι αυτό έγινε εκτενής αναφορά στην αρχιτεκτονική της εφαρμογής, στα design patterns που χρησιμοποιήθηκαν, αλλά και σε λοιπές τεχνολογίες και εργαλεία που αξιοποιήθηκαν, μέσω της πλατφόρμας του Android. Τέλος, αναφέρθηκαν και τα υποστηρικτικά εργαλεία, όπως τα IDEs και οι text editors για τη συγγραφή του κώδικα, καθώς και εφαρμογές για το testing των endpoints και τον έλεγχο και διαμοιρασμό του κώδικα.

Κεφάλαιο 3ο: Περιγραφή Προβλήματος

3.1 Εισαγωγή

Η παρούσα διπλωματική καλείται να προσφέρει λύση σε δύο προβλήματα. Αρχικά, το ιστορικό υγείας και ο ιατρικός φάκελος του κατοικίδιου, στεγάζονται στο βιβλιάριο υγείας του. Η φυσική μορφή του είναι ένας μεγάλος περιοριστικός παράγοντας, όσον αφορά την προστασία του από φθορές, καταστροφές, αλλά και τοποθετήσεις σε λανθασμένη θέση με αποτέλεσμα την εξαφάνιση του. Η πλήρης μεταφορά του σε ηλεκτρονική μορφή, πέρα από το ότι συμβαδίζει με την σύγχρονη εποχή και την τάση για ψηφιοποίηση, διασφαλίζει και την προστασία του περιεχομένου του βιβλιαρίου, την ακεραιότητα των πληροφοριών και την πρόσβαση σε αυτές, οποιαδήποτε στιγμή και από οποιοδήποτε σημείο, χωρίς την ύπαρξη φυσικών αντικρισμάτων. Το δεύτερο πρόβλημα αφορά την εξαφάνιση ενός κατοικίδιου. Για την ακρίβεια, η εφαρμογή προσφέρει ακόμα μια επιλογή για την έγκαιρη επικοινωνία της εξαφάνισης. Χρησιμοποιώντας τεχνικές υπό-συνθήκης μαζικής επικοινωνίας ειδοποιήσεων σε κινητές συσκευές, καθιστά δυνατή την κοινοποίηση κάποιας εξαφάνισης, εύκολα και γρήγορα. Η προσέγγιση αυτή δεν μπορεί από μόνη της να λύσει το θέμα της επικοινωνίας του προβλήματος. Σκοπός της είναι να προσφέρει ακόμα ένα μέσο άμεσης διάδοσης της πληροφορίας, καθώς ο χρόνος απόκρισης και δράσης μετά από κάποια δηλωμένη εξαφάνιση, είναι εξαιρετικά κρίσιμος για την αποτελεσματική και επιτυχή ανεύρεση του κατοικίδιού. Στο κεφάλαιο αυτό θα παρουσιαστούν υπάρχουσες εφαρμογές του χώρου της φροντίδας των κατοικίδιων και θα συγκριθούν με την εφαρμογή που αναπτύχθηκε στα πλαίσια της παρούσας εργασίας.

3.2 Υπάρχουσες Εφαρμογές

Υπάρχει πληθώρα εφαρμογών που προσεγγίζουν το θέμα της φροντίδας του κατοικίδιου από ποικίλες σκοπιές. Πιο συγκεκριμένα, υπάρχει η εφαρμογή “Pet First Aid” [49] του Αμερικάνικου Ερυθρού Σταυρού, η οποία, εκτός από βασικές λειτουργίες διαχείρισης του κατοικίδιου, παρέχει υπηρεσίες πρώτων βοηθειών και ιατρικής φύσης. Επίσης, υπάρχουν εφαρμογές όπως η “Chewy” [48] και η “AlmaPet” [50] οι οποίες επικεντρώνονται στο κομμάτι της αγοράς προμηθειών διατροφής και ψυχαγωγίας. Επιπλέον, εφαρμογές όπως η “Rover” [51] και η “Wag!” [52] ειδικεύονται στο κομμάτι της φροντίδας των κατοικίδιων από τρίτους (dog/cat-sitting) και στην αναζήτηση κατοικίδιου-συντρόφου, τόσο για ψυχαγωγικούς, όσο και για αναπαραγωγικούς σκοπούς. Ωστόσο, υπάρχουν και εφαρμογές που στον πυρήνα τους ενσωματώνουν λειτουργίες, που περιλαμβάνονται και στην παρούσα εργασία, με αποτέλεσμα να μπορούν να συγκριθούν άμεσα σε πολλούς τομείς. Οι εφαρμογές αυτές είναι οι 11pets: Pet Care [53], VitusVet [54], Petable [55], Great Pet Care [56], PetDesk [].

Σχήμα 3.1: Από αριστερά προς δεξιά: λογότυπα των εφαρμογών Chewy [48], Almapet [50], Rover [51], Wag! [52], Pet First Aid [49]

3.2.1 11pets: Pet Care

Η εφαρμογή 11pets: Pet care [53] αποτελεί μια ψηφιακή πλατφόρμα φροντίδας κατοικίδιων. Καλύπτει ένα ευρύ φάσμα αναγκών, σχετιζόμενων με την διαχείριση κατοικίδιων. Περιλαμβάνει λειτουργίες όπως διαχείριση του ιατρικού φακέλου του κατοικίδιου (εμβολιασμοί, αντιπαρασιτικές θεραπείες, φαρμακευτικές αγωγές), κοινοποίηση του φακέλου σε κτηνίατρο, διαχείριση της υγιεινής και της διατροφής, παρακολούθηση και καταγραφή των επισκέψεων σε κτηνιατρικές κλινικές, καθώς και παραγωγή υπενθυμίσεων για δραστηριότητες και ιατρικές υποχρεώσεις του κατοικίδιου. Η εφαρμογή περιλαμβάνει και ένα συνδρομητικό πλάνο το οποίο προσφέρει επιπλέον λειτουργείες, όπως συγχρονισμός του προγράμματος του κατοικίδιου με το ημερολόγιο της συσκευής, καταγραφή εξόδων αλλά και άλλων quality-of-life λειτουργειών.

3.2.2 VitusVet

Η VitusVet [54] παρέχει βασικές λειτουργείες διαχείρισης του ιατρικού φακέλου ενός κατοικίδιου, όπως τη δημιουργία και την κοινοποίηση του με συμβεβλημένο με την εφαρμογή κτηνίατρο. Επιπλέον, δίνεται η δυνατότητα προγραμματισμού ραντεβού σε κτηνιατρική κλινική καθώς και η παραγωγή υπενθυμίσεων για ιατρικά και μη γεγονότα. Ωστόσο, ο κτηνίατρος πρέπει να είναι συμβεβλημένος με την εφαρμογή και για το πλήρες πακέτο υπηρεσιών θα πρέπει να καταβάλει το μηνιαίο ποσό των 399$ δολαρίων τον μήνα. Τέλος, δεν διευκρινίζεται αν οι υπηρεσίες είναι διαθέσιμες εκτός των Ηνωμένων Πολιτειών της Αμερικής.

3.2.3 Petable

Η Petable [55], όπως και η VitusVet, προσφέρει λειτουργείες διαχείρισης του κατοικίδιου και του ιατρικού φακέλου του, καθώς και δυνατότητα δρομολόγησης ραντεβού με κτηνιατρικές κλινικές. Και αυτή προσφέρει επιπλέον υπηρεσίες διαχείρισης για τον κτηνίατρο, μέσω ενός CMS, με μηνιαία, κλιμακωτή συνδρομή.

3.2.4 Great Pet Care

Η Great Pet Care [56] αποτελεί μια εφαρμογή διατήρησης και κοινοποίησης στον κτηνίατρο, του ιατρικού φακέλου του κατοικίδιου. Οι άνθρωποι πίσω από την εφαρμογή, λαμβάνουν τον ρόλο του μεσάζοντα, σε περίπτωση που ο χρήστης θελήσει να λάβει σε ηλεκτρονική μορφή, εντός της εφαρμογής, τα επίσημα ιατρικά έγγραφα από μια επίσκεψη στον κτηνίατρο. Επιπλέον, προσφέρει τη δυνατότητα ρύθμισης ειδοποιήσεων για γεγονότα σχετιζόμενα με την υγεία και την καθημερινότητα του κατοικίδιου.

3.2.5 PetDesk

Η PetDesk [57] είναι μια εφαρμογή παρακολούθησης και καταγραφής της υγείας του κατοικίδιου. Παρέχει λειτουργείες προγραμματισμού ραντεβού με κτηνίατρο, ρύθμισης ειδοποιήσεων, ενώ προσφέρει και loyalty programs με συνεργαζόμενα καταστήματα. Όπως και σε προηγούμενες εφαρμογές, προσφέρεται και CMS για το διαχειριστικό κομμάτι, από την πλευρά του κτηνιάτρου, επί πληρωμή.

3.3 My Pet – mypet.gov.gr

Η ελληνική κυβέρνηση, στα πλαίσια της ψηφιοποίησης των προσωπικών στοιχείων και υπηρεσιών, και έπειτα των νέων διατάξεων περί φροντίδας των ζώων συντροφιάς [45], θα προχωρήσει στην ενοποίηση

υπαρχόντων υπηρεσιών σχετικών με τα κατοικίδια και τα αδέσποτα ζώα, υπό μια νέα πλατφόρμα, την My Pet. Πιο συγκεκριμένα, με βάση την κοινή υπουργική απόφαση υπ.αριθμ. 1609 ΕΞ 2022 - ΦΕΚ 203/Β/27-1-2022 [46], καθιερώνεται το Εθνικό Μητρώο Ζώων Συντροφιάς και καθορίζονται οι λεπτομέρειες λειτουργίας του. Το ΕΜΖΣ αποτελεί την βάση για την λειτουργία πληθώρας υπηρεσιών και εφαρμογών και περιλαμβάνει τα υπομητρώα και υπηρεσίες:

- Καταγραφής και Παρακολούθησης Ζώων Συντροφιάς
- Φιλοζωικών Σωματείων και Οργανώσεων
- Καταφυγίων Ζώων Συντροφιάς
- Επαγγελματιών και Ερασιτεχνών Εκτροφέων
- Πανελλήνια Πλατφόρμα Υιοθεσίας Αδέσποτων Ζώων Συντροφιάς
- Εθελοντικής Αιμοδοσίας Ζώων Συντροφιάς
- Αποθετήριο Διαβατηρίων

Οι υπηρεσίες που θα παρέχονται στους ιδιοκτήτες ζώων είναι η ταυτοποίηση QR ζώου, η ψηφιακή υπηρεσία δήλωσης μεταβίβασης δεσποζόμενου ζώου μεταξύ ιδιωτών, η υπηρεσία ανακοίνωσης θανάτου δεσποζόμενου ζώου συντροφιάς, η υπηρεσία απώλειας, δεσποζόμενου ζώου συντροφιάς, η υπηρεσία εύρεσης, δεσποζόμενου ζώου συντροφιάς, η υπηρεσία ανακοίνωσης πιθανής εύρεσης και η πύλη απώλειας και εύρεσης ζώων συντροφιάς [47]. Κάθε κατοικίδιο θα διαθέτει έναν ψηφιακό ιατρικό φάκελο, ο οποίος θα στεγάζει τα δεδομένα του βιβλιαρίου υγείας καθώς και πρόσθετες πληροφορίες που κρίνει σημασίας ό κτηνίατρος, για ιατρικά θέματα, αλλά και ο ιδιοκτήτης, για μη ιατρικά. Η ιατρική καρτέλα θα είναι προσβάσιμη τόσο μέσω της ψηφιακής υπηρεσίας mypet.gov.gr [47], όσο και αυτόνομα, μέσω iOS και Android εφαρμογών. Αποτελεί συνολικά ένα τεράστιο εγχείρημα, το οποίο θα ολοκληρωθεί εντός των επόμενων δύο ετών (2023 - 2024).

Σχήμα 3.2: Από αριστερά προς δεξιά: λογότυπα των εφαρμογών 11pets: Pet Care [53], VitusVet [54], Petable [55], Great Pet Care [56], PetDesk [57]

3.4 Σύγκριση

Σχήμα 3.3: Λογότυπο του προϊόντος Tractive [58]

Γίνεται ευκολά αντιληπτό πως, οι κατεξοχήν δωρεάν εφαρμογές ή οι εφαρμογές που μπορούν να χρησιμοποιηθούν και να αξιοποιηθούν πλήρως στη χώρα μας, είναι ελάχιστες. Η παρούσα εφαρμογή, αν και μοιράζεται πολλά από τα χαρακτηριστικά των παραπάνω, διαφοροποιείται από αυτές, εισάγοντας λειτουργίες διευκόλυνσης ανεύρεσης των κατοικίδιων σε περίπτωση εξαφάνισης. Υπάρχουν πακέτα υπηρεσιών που προσφέρουν επί πληρωμή εξοπλισμό και συνδρομητικές υπηρεσίες για την ζωντανή παρακολούθηση της θέσης του κατοικίδιου, όπως η Tractive [58], ωστόσο, μια τέτοια εφαρμογή δε θα μπορούσε να συγκριθεί σε ίσους όρους με την τρέχουσα εργασία. Η παρούσα διπλωματική προσφέρει δωρεάν τις υπηρεσίες της ενώ βασίζεται στην εκάστοτε τοπική κοινότητα για την αποτελεσματική λειτουργία της. Ουσιαστικά, ενσωματώνει και συνδυάζει

κάποιες σύγχρονες τακτικές κλήσης σε βοήθεια, από άτομα που αντιμετωπίζουν το σχετικό πρόβλημα, ενώ τις οργανώνει με έναν ουσιαστικό τρόπο και τις κάνει πιο προσβάσιμες. Έτσι, ταυτόχρονα, ενθαρρύνει τη συνεργασία των φιλόζωων πολιτών και προάγει το πνεύμα της αλληλεγγύης.

Όσον αφορά τις ενέργειες της ελληνικής κυβέρνησης, δύσκολα η παρούσα εφαρμογή μπορεί να συγκριθεί με την κλίμακα των υπηρεσιών που μπορεί εν δυνάμει να προσφέρει. Η νέα πλατφόρμα της κυβέρνησης, πέρα από το ότι θα αποτελεί επίσημη εφαρμογή και θα φέρει όλα τα επικαιροποιημένα στοιχεία των κατοικίδιων και των ιδιοκτητών τους, προσφέρει όλους τους πόρους που είναι διαθέσιμοι. Αποτελεί ένα τεράστιο βήμα προς τη σωστή κατεύθυνση, ωστόσο θα χρειαστεί χρόνος μέχρι να παρατηρηθούν αποτελέσματα καθώς απαιτείται μεταβολή στη συλλογική συνείδηση της κοινωνίας σχετικά με το θέμα της φροντίδας των κατοικίδιων, και ό,τι αυτό περιλαμβάνει, κάτι το οποίο χρειάζεται χρόνο και προσέγγιση από πολλές διαφορετικές γωνίες.

3.5 Επίλογος

Στο κεφάλαιο αυτό παρουσιάστηκαν τα προβλήματα που καλείται να αντιμετωπίσει η εφαρμογή. Αυτά αφορούν την ψηφιοποίηση του ιατρικού βιβλιαρίου του κατοικίδιου και την διευκόλυνση μαζικής ενημέρωσης του κοινού σε περίπτωση εξαφάνισης κατοικίδιου. Επιπλέον, έγινε εκτενής αναφορά σε πληθώρα παρόμοιων εφαρμογών, με στοχευμένη επισήμανση, τόσο των συγκρίσιμων με την παρούσα εφαρμογή χαρακτηριστικών τους, όσο και των στοιχείων που καθορίζουν την ταυτότητα τους. Επίσης, έγινε μια συγκεντρωτική σύγκριση τους με την τρέχουσα υλοποίηση της εφαρμογής που παρουσιάζεται στη διπλωματική εργασία, με έμφαση στα στοιχεία που αυτές υστερούν. Τέλος, αναφέρθηκαν οι μελλοντικές ενέργειες που προγραμματίζει η ελληνική πολιτεία, με την ενοποίηση σχετικών υπηρεσιών και την ανάπτυξη επίσημης πλατφόρμας για τη διαχείριση των κατοικίδιων

Κεφάλαιο 4ο: Επίλυση Προβλήματος

4.1 Εισαγωγή

Ο σχεδιασμός της εφαρμογής βασίστηκε, ως επί των πλείστων, σε ένα πραγματικό βιβλιάριο υγείας κατοικίδιου. Τα πεδία τα οποία καλείται να συμπληρώσει ο χρήστης αντιστοιχούν πλήρως σε αυτά του βιβλιαρίου. Οι κανόνες υποχρεωτικότητας των πεδίων αυτών, αποφασίστηκαν ημι-αυθαίρετα. Βασικό κριτήριο στην επιλογή τους ήταν η ελευθερία του χρήστη να συμπληρώσει όσα στοιχεία επιθυμεί, με συγκεκριμένους περιορισμούς ανά περίπτωση. Μπορεί δηλαδή να διατηρήσει ένα πλήρες αρχείο και πιστό αντίγραφο του βιβλιαρίου ή να συμπληρώσει μόνο τα υποχρεωτικά πεδία, κρατώντας τα απολύτως απαραίτητα. Είναι στην κρίση του το πως θα χρησιμοποιήσει την εφαρμογή. Επιπλέον, δεν υπάρχουν αυστηροί περιορισμοί στην εισαγωγή ημερομηνιών, καθώς η εισαγωγή εγγραφών στο ιατρικό ιστορικό του κατοικίδιου, μπορούν κάλλιστα να αφορούν παλαιότερες ημερομηνίες, αφού η μεταφορά των στοιχείων του φυσικού βιβλιαρίου στην εφαρμογή, είναι μια απολύτως έγκυρη ενέργεια.

Η εφαρμογή αποτελείται από 32 διακριτές οθόνες. Όλες τους ακολουθούν το ίδιο σχεδιαστικό πρότυπο για τη διασφάλιση της ομοιομορφίας και της συνοχής μεταξύ τους, αλλά και για την προβλεψιμότητα της ακολουθίας των ενεργειών, από την πλευρά του χρήστη. Όσον αφορά την υλοποίηση της εφαρμογής καθολικά, αυτή διακρίνεται σε δύο επιμέρους projects. Το πρώτο αποτελεί την Android εφαρμογή, με την οποία αλληλοεπιδρά ο χρήστης. Αυτή αποτελείται από τέσσερα βασικά Activities και πληθώρα Fragments συσχετισμένα με αυτά. Τα Activities αυτά ορίζονται με βάση την κύρια λειτουργικότητά τους και αφορούν την σύνδεση και εγγραφή του χρήστη και του κτηνιάτρου, την προβολή και προσθήκη κατοικίδιων, την προβολή και επεξεργασία των στοιχείων και του φακέλου των κατοικίδιων και την προβολή των χαρτών. Το δεύτερο μέρος της εφαρμογής είναι ο server, ο οποίος προσφέρει την υποδομή που καθιστά λειτουργικά όλα τα χαρακτηριστικά της εφαρμογής. Ο server εξυπηρετεί όλα τα αιτήματα της εφαρμογής, κάνοντας εφικτή την επικοινωνία με την βάση δεδομένων, ενώ είναι υπεύθυνος και για τη διαχείριση όλων των εργασιών που πραγματοποιούνται στο υπόβαθρο. Τέλος, η δομή τόσο του frontend όσο και του backend, σε επίπεδο οργάνωσης αρχείων, ακολουθεί την ίδια λογική. Τα αρχεία χωρίζονται σε φακέλους και πακέτα με βάση την λειτουργία τους και την ιδιότητα τους και δεν ομαδοποιούνται ανάλογα με τη σχετικότητα τους με κάποια λειτουργικότητα της εφαρμογής. Εξαίρεση αποτελεί το πακέτο googlemaps της Android εφαρμογής, το οποίο περιλαμβάνει όλα τα παρεμφερή αρχεία για την λειτουργία των χαρτών. Παρακάτω, ακολουθεί αναλυτική περιγραφή της εφαρμογής, ως σύνολο, με βάση τις κύριες λειτουργικότητες της, καθώς και των οθονών της Android εφαρμογής με τις πληροφορίες που προβάλουν.

4.2 Επισκόπηση ροής των αιτημάτων και των απαντήσεων

Η διαχείριση ενός αιτήματος από άκρο σε άκρο περιλαμβάνει τη συμμετοχή πλήθους ενδιάμεσων μερών. Στην κορυφή βρίσκεται κάποιο View, το οποίο είναι συσχετισμένο με ένα Fragment ή κάποιο Activity. Το Fragment ή Activity, καλεί το ViewModel που έχει δηλωμένο. Στην περίπτωση της παρούσας εφαρμογής, τα ViewModels εξυπηρετούν πολλαπλά Views, τα οποία υπάγονται στην ίδια θεματική κατηγορία. Για παράδειγμα, το PetViewModel, εξυπηρετεί όλα τα Views που σχετίζονται με λειτουργίες του κατοικίδιου, όπως την προβολή, προσθήκη και επεξεργασία. Το ViewModel, στη συνέχεια, καλεί το Repository, που με τη σειρά του, μέσω του Retrofit, θα στείλει το αίτημα στον server. Να σημειωθεί πως ο τρόπος διαχωρισμού των Repositories, ακολουθεί την ίδια λογική με τα ViewModels. Το Retrofit ορίζεται στην εφαρμογή μέσω μιας singleton κλάσης. Αυτό σημαίνει πως

υπάρχει μόνο ένα instance του ανά πάσα στιγμή. Η κλάση ServiceGenerator, είναι υπεύθυνη για την κατασκευή και παραμετροποίηση του. Αυτή περιλαμβάνει:

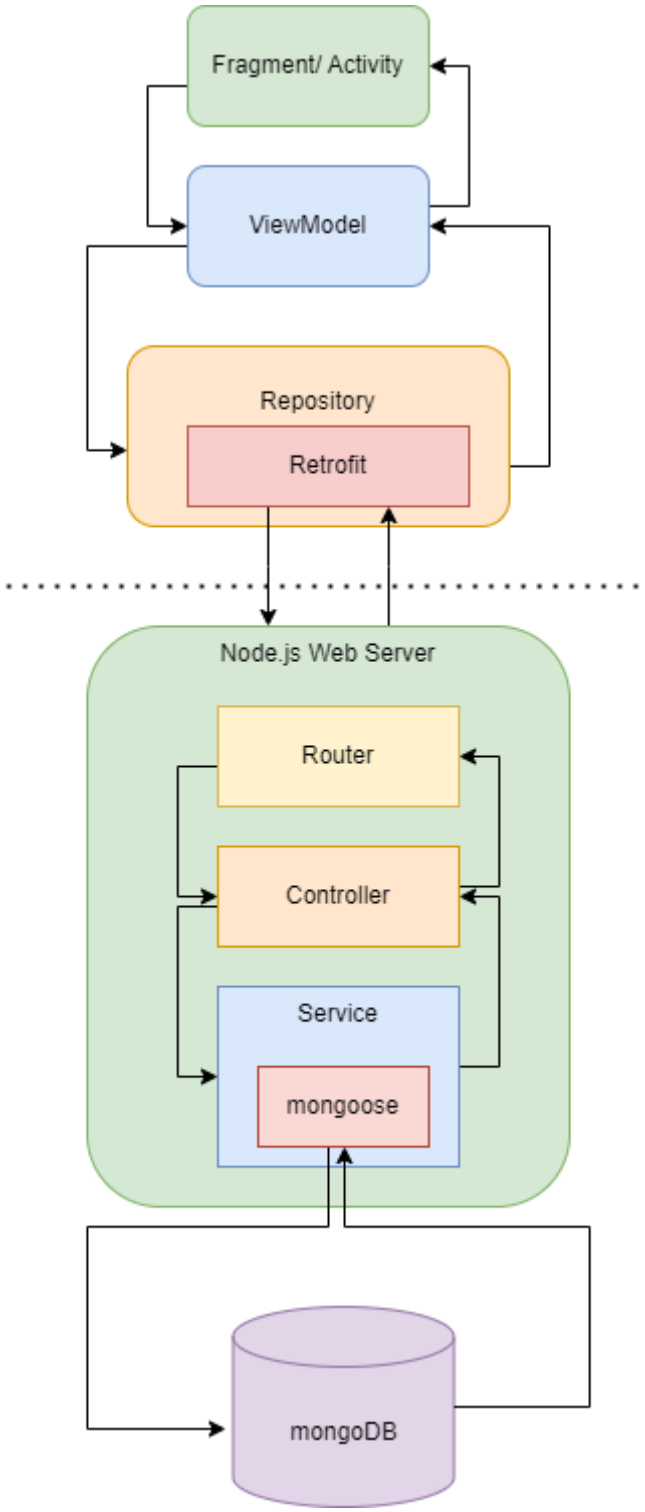
- τη δήλωση ενός header interceptor, για την ανάγνωση των headers των αιτημάτων, στο authorization μέρος των οποίων βρίσκεται ένα JWT ως bearer Token,
- τη δήλωση του OkHttpClient,
- τη δήλωση του base url του server
- τη δήλωση του json serializer/deserializer για το σώμα του αιτήματος
- την ανάθεση του API, ενός interface, το οποίο αντιστοιχεί τα endpoints προς κλήση με συναρτήσεις της Kotlin. Οι συναρτήσεις αυτές διαθέτουν annotations που καθορίζουν την http μέθοδο, τις query παραμέτρους, καθώς και το body του request

Η κλάση αυτή, εκθέτει το αντικείμενο του Retrofit μέσω μιας get μεθόδου. Το Retrofit υποστηρίζει την ασύγχρονη αποστολή αιτημάτων και προσφέρει callback μεθόδους για τη διαχείριση του response.

Όταν το αίτημα, εν τέλει, φτάσει στον server, το υποδέχεται ο αντίστοιχος Router, που είναι παραμετροποιημένος για να το εξυπηρετήσει. Η παραμετροποίηση αφορά το τύπο του http method που θα δεχτεί και το path με τις query παραμέτρους. Στον Router ορίζονται και τα middlewares που θα εκτελεστούν, με βάση τη σειρά που έχουν δηλωθεί. Η εφαρμογή υλοποιεί token based authorization, που σημαίνει πως κάθε αίτημα προς τον server φέρει ένα JWT στα headers του. Το JWT αυτό, έχει παραχθεί προηγουμένως από τον server και αποθηκεύεται στα SharedPreferences της Android εφαρμογής, ένα key/value store του Android Framework. Σε κάθε αίτημα, το πρώτο middleware που εκτελείται, αφορά την επιβεβαίωση της ορθότητας του JWT που ελήφθη. Εξαίρεση αποτελούν τα endpoints για την σύνδεση του χρήστη και του κτηνιάτρου, τα οποία είναι αυτά που παράγουν το JWT εξαρχής. Έπειτα, καλείται ο Controller που διαχειρίζεται το εκάστοτε σύνολο των endpoints. Ο Controller καλεί κάποιο Service που επικοινωνεί με τη βάση μέσω του Mongoose framework της mongoDB και έπειτα, αποστέλλει το response αντικείμενο.

Για τη διαχείριση της αναμονής της απάντησης, έχει κατασκευαστεί ένα Interface, το ResponseFunctions, το οποίο προσφέρει τρεις μεθόδους προς υλοποίηση, την onStart, την onSuccess και την onFail. Το Interface αυτό υλοποιείται από το Fragment ή το Activity που ξεκίνησε το request. Η onStart εκτελείται από το ViewModel, πριν αυτό καλέσει το Repository. Αυτή η μέθοδος χρησιμοποιείται για εκτέλεση ενεργειών εν αναμονή της απάντησης. Στην προκειμένη περίπτωση, για την εμφάνιση κάποιας ένδειξης αναμονής ολοκλήρωσης της ενέργειας. Όταν το Retrofit λάβει την απάντηση στο Repository, αποθηκεύει το response object σε ένα MutableLiveData αντικείμενο και το εκθέτει μέσω μιας get μεθόδου. Στη συνέχεια το ViewModel, ανάλογα με το status του response, εκτελεί την αντίστοιχη μέθοδο του ResponseFunctions. Τέλος, το Fragment ή το Activity, το οποίο έκανε εξαρχής observe το αντικείμενο του ViewModel, που έχει λάβει τιμή μέσω του Repository, μπορεί να διαχειριστεί τα ληφθέντα δεδομένα και να τα προβάλει στο View του.

Η διαδικασία αυτή αποτελεί τον κορμό της εφαρμογής, σαν σύνολο, όσον αφορά τη διαχείριση των αιτημάτων. Κάθε αίτημα προς τον server ξεκινάει είτε μέσω του πατήματος κάποιου κουμπιού ή κατά τη δημιουργία κάποιου Fragment ή Activity. Στο Σχήμα 4.1 φαίνεται ένα διάγραμμα της προαναφερθείσας διαδικασίας.



Σχήμα 4.1: Διάγραμμα διαχείρισης των αιτημάτων και των απαντήσεων

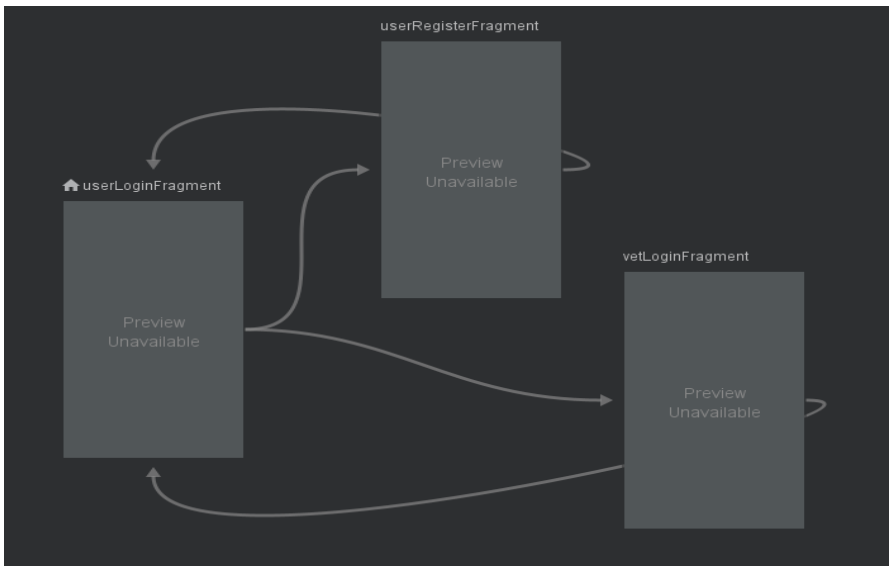
4.3 Σύνδεση στην εφαρμογή

Η οθόνη σύνδεσης χρήστη είναι η πρώτη οθόνη που αντικρίζει ο χρήστης όταν ανοίγει την εφαρμογή και αποτελεί το main Activity της. Στην πραγματικότητα, πρόκειται για ένα Fragment που φορτώνεται στο NavHost container view του Activity, το οποίο κάνει δυνατή τη πλοήγηση μεταξύ των Fragments μέσω του NavController αντικειμένου. Η σχέση των διαδρομών μεταξύ των Fragments δηλώνεται σε ένα Navigation Graph, το οποίο αποτελεί έναν διαδραστικό τρόπο δήλωσης της σχέσης αυτής, με εν

δυνάμει αμφίδρομα βέλη να δηλώνουν τις πιθανές κατευθύνσεις, από και προς τα εν λόγω Fragments. Μέσω της οθόνης σύνδεσης, ο χρήστης μπορεί να περατώσει τρεις διακριτές ενέργειες. Η πρώτη αφορά τη σύνδεση του στην εφαρμογή. Στο μέσο της οθόνης βρίσκονται δύο textInput στοιχεία, για την εισαγωγή του username και password. Κάτω από αυτά, βρίσκεται το κουμπί της σύνδεσης, το πάτημα του οποίου, πυροδοτεί το onClick event του, το οποίο ξεκινά τη διαδικασία της σύνδεσης. Πιο συγκεκριμένα, το onClick event είναι συνδεδεμένο μέσω data binding με μια μέθοδο του ViewModel, όπως είναι και τα textInput στοιχεία του username και password με δύο μεταβλητές. Η σχέση αυτή δηλώνεται στο xml αρχείο που ορίζει τη σχεδίαση του layout του Fragment. Με αυτόν τον τρόπο, το ViewModel έχει απευθείας πρόσβαση στις τιμές των στοιχείων αυτών. Η μέθοδος αυτή είναι υπεύθυνη για το validation των πεδίων - στην συγκεκριμένη περίπτωση, με βάση την υποχρεωτικότητα τους. Το αποτέλεσμα του validation των πεδίων, αποθηκεύεται σε μια MutableLiveData μεταβλητή, την οποία κάνει observe το Fragment. Όταν αυτή πάρει τιμή, σημαίνει πως το validation έχει αποτύχει και το Fragment εμφανίζει το μήνυμα σφάλματος στο σχετικό textView. Αν δεν υπάρχει κάποιο σφάλμα, καλείται η μέθοδος του UserViewModel που είναι υπεύθυνη για το login. Αυτή με τη σειρά της καλεί το UserRepository, το οποίο, μέσω του Retrofit στέλνει το αίτημα στον server. Ο server, αφού επιβεβαιώσει τα στοιχεία του χρήστη, παράγει ένα JWT και το τοποθετεί στα headers του response με βάση τη συνθήκη bearer Token. Η εφαρμογή, όταν λάβει το token, θα το αποθηκεύσει στα SharedPreferences για χρήση στα υπόλοιπα requests. Στο σχήμα 4.3 παρουσιάζεται η οθόνη σύνδεσης του χρήστη.

Κάτω από το κουμπί της σύνδεσης, υπάρχουν δύο λεκτικά. Το πρώτο αναφέρεται στη σύνδεση στην εφαρμογή ως κτηνίατρος και το δεύτερο παραπέμπει τον χρήστη στη δημιουργία νέου λογαριασμού. Η οθόνη σύνδεσης κτηνιάτρου διαφέρει μόνο στον χρωματισμό της. Η διαδικασία σύνδεσης είναι ακριβώς ίδια με αυτή του χρήστη. Η πλοήγηση πραγματοποιείται μέσω του NavController που προαναφέρθηκε. Το Σχήμα 4.2 δείχνει το NavGraph του MainActivity.

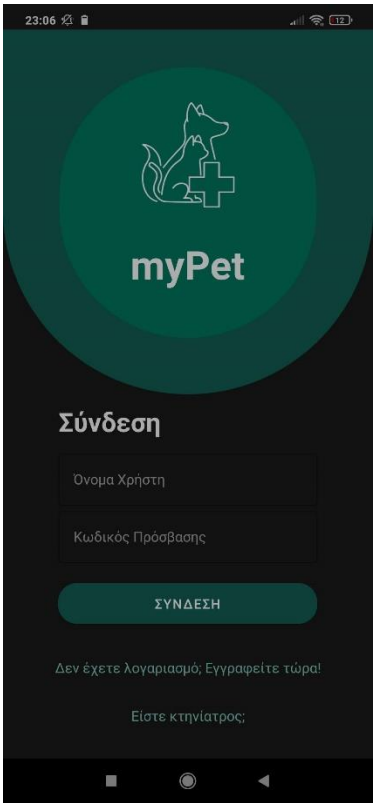
Η επιτυχής σύνδεση στην εφαρμογή ξεκινάει ακόμα μία διαδικασία - αυτήν της λήψης και αποθήκευσης του FCM token στη βάση. Στο σημείο αυτό καλείται η getToken() μέθοδος της FirebaseMessaging κλάσης που προσφέρεται από το Firebase SDK. Με την επιτυχή λήψη του token, καλείται η μέθοδος saveFCMToken του UserViewModel, η οποία είναι υπεύθυνη για την αποθήκευση του στη βάση. Η αποστολή των ειδοποιήσεων στην εφαρμογή, θα γίνεται με βάση αυτό το token.



Σχήμα 4.2: Navigation Graph του MainActivity

4.4 Εγγραφή στην εφαρμογή

Η εγγραφή στην εφαρμογή ακολουθεί μια διαφορετική προσέγγιση από αυτή της σύνδεσης. Η βασική διαφορά μεταξύ τους, είναι η μη χρήση data binding για την σύνδεση των textInput πεδίων των στοιχείων του χρήστη, με μεταβλητές στο ViewModel. Η φόρμα της εγγραφής περιλαμβάνει τέσσερα υποχρεωτικά πεδία, το όνομα χρήστη, το email, τον κωδικό πρόσβασης και την επιβεβαίωση κωδικού πρόσβασης. Κάθε πεδίο, πέρα από το ότι είναι υποχρεωτικό, διαθέτει και συγκεκριμένα validations, όπως για παράδειγμα, το email πρέπει να έχει συγκεκριμένη δομή για να θεωρείται έγκυρο ή οι κωδικοί πρέπει να ταιριάζουν. Για την υλοποίηση αυτή και τη μεταφορά της λογικής του validation εκτός του View, χρησιμοποιήθηκε ένα αντικείμενο-μοντέλο της φόρμας που διατηρεί την κατάσταση εγκυρότητας του κάθε στοιχείου. Κάθε φορά που χάνει focus ένα από αυτά τα πεδία, δηλαδή, κάθε φορά που ο χρήστης σταματά να τα επεξεργάζεται και τα αποεπιλέγει ή επιλέγει κάποιο άλλο στοιχείο, πυροδοτείται το onFocusChanged event του στοιχείου αυτού. Το event αυτό καλεί μια μέθοδο του ViewModel, η οποία είναι υπεύθυνη για το validation του πεδίου που δέχεται ως παράμετρο. Η μέθοδος αυτή, στη συνέχεια, αποθηκεύει την κατάσταση της εγκυρότητας του πεδίου σε μια MutableLiveData μεταβλητή, την οποία κάνει observe το Fragment. Με αυτόν τον τρόπο, το Fragment γνωρίζει πότε και ποιο στοιχείο της φόρμας δεν είναι έγκυρο και κατ' επέκταση, εμφανίζει το σχετικό μήνυμα κάτω από το αντίστοιχο textInput. Το κουμπί της εγγραφής πυροδοτεί μια παρόμοια διαδικασία με αυτή του login, κατά την οποία, αφού γίνουν validate όλα τα πεδία, καλείται η μέθοδος που είναι υπεύθυνη για την εγγραφή, η οποία με τη σειρά της θα καλέσει το Repository για να στείλει τα δεδομένα προς αποθήκευση στη βάση. Με την επιτυχή εγγραφή, ο χρήστης ανακατευθύνεται στην οθόνη σύνδεσης, όπου πλέον μπορεί να εισάγει τα στοιχεία του για να εισέλθει στην εφαρμογή. Στο Σχήμα 4.4, φαίνεται η μέθοδος διαχείρισης του onClick του κουμπιού εγγραφής.



Σχήμα 4.3: Οθόνη σύνδεσης χρήστη

```
fun onRegisterButtonClick(username : String, email : String, password: String, confirmPassword: String) {
    validateRegisterFormField(RegisterFormFields.Username, username)
    validateRegisterFormField(RegisterFormFields.Email, email)
    validateRegisterFormField(RegisterFormFields.Password, password)
    validateRegisterFormField(RegisterFormFields.ConfirmPassword, confirmPassword)

    if(registerFormValidation.isFormValidated()){
        responseListener?.OnStarted()
        registerUser(username, password, confirmPassword, email)
    }
}
```

Σχήμα 4.4: Μέθοδος διαχείρισης του onClick event του κουμπιού εγγραφής χρήστη

4.5 Λίστα κατοικίδιων

Με την επιτυχή σύνδεση του, ο χρήστης φτάνει στην οθόνη της λίστας των κατοικίδιων του, όπου μπορεί να δει, συνοπτικά, όλα τα κατοικίδια που έχει προσθέσει στην εφαρμογή. Η οθόνη αυτή αποτελεί ένα νέο Activity με πληθώρα από Fragments. Η λίστα υλοποιείται με ένα RecyclerView χρησιμοποιώντας έναν LinearLayoutManager. Κάθε στοιχείο τις λίστας αποτελείται από ένα CardView, με την εικόνα του κατοικίδιου στο αριστερό μέρος και το όνομα, την ράτσα, το φύλο και την ηλικία του στο δεξί. Η λήψη των δεδομένων των κατοικίδιων γίνεται στην onViewCreated() μέθοδο του Fragment, όπου και γίνεται η κλήση στην ανάλογη μέθοδο του PetsViewModel. Στο κάτω δεξί μέρος της οθόνης, βρίσκεται ένα κουμπί το οποίο ανοίγει ένα νέο Fragment, υπεύθυνο για την προσθήκη νέου κατοικίδιου. Η οθόνη αυτή περιλαμβάνει μια φόρμα με πληθώρα στοιχείων προς συμπλήρωση, βασισμένα στα στοιχεία που ο ιδιοκτήτης καλείται να συμπληρώσει σε ένα πραγματικό βιβλιάριο υγείας ενός κατοικίδιου. Η διαχείριση της εγκυρότητας και της υποχρεωτικότητας των πεδίων της φόρμας, γίνεται με τον τρόπο που περιεγράφηκε στην ενότητα της εγγραφής του χρήστη. Όταν όλα τα υποχρεωτικά πεδία συμπληρωθούν και πατηθεί το κουμπί της υποβολής, πραγματοποιείται ένα αίτημα, μέσω του PetViewModel και PetRepository, για την αποθήκευση του κατοικίδιου στη βάση. Στην πλευρά του backend, αφού ελεγχθεί το JWT ταυτοποίησης για την ορθότητα του, δημιουργείται ένα νέο αντικείμενο με τα ληφθέντα στοιχεία του κατοικίδιου και εισάγεται ως document στο collection “Pets” της βάσης. Στο Σχήμα 4.5 φαίνεται η δομή ενός document κατοικίδιου. Η ηλικία του κατοικίδιου, αποτελεί ένα virtual πεδίο του document, που σημαίνει ότι δεν αποθηκεύεται, αλλά υπολογίζεται όταν ζητηθεί το document. Η επιτυχής αποθήκευση ανακατευθύνει τον χρήστη στην οθόνη της λίστας των κατοικίδιων.

```
_id: ObjectId("6395f96b12114b31df4e148b")
id: ""
name: "Μόμπυ"
birthdate: 2022-12-11T00:00:00.000+00:00
colour: "Μαύρο"
distinguishingMarks: ""
species: "Σκύλος"
breed: "Ημίαιμο"
sex: "Αρσενικό"
photo: "1ED13URaG2QYZpH9djt6x03zmx7Z-gXVO"
weight: 10
height: 50
medicalRecord: ObjectId("6395f96b12114b31df4e1489")
owner: ObjectId("61c9956caccedf38e562be2e")
isMissing: false
__v: 0
```

Σχήμα 4.5: Ενδεικτικό document ενός κατοικίδιου

Στην άνω δεξιά γωνία της οθόνης των κατοικίδιων, υπάρχει ένα three-dot-menu, το πάτημα του οποίου εμφανίζει ένα μενού με πέντε επιλογές. Οι δύο πρώτες αφορούν τις υπενθυμίσεις και τα πλησιέστερα κτηνιατρεία και θα αναλυθούν στα επόμενα κεφάλαια. Μέσω της τρίτης επιλογής, ο χρήστης μπορεί να αλλάξει κάποιες ρυθμίσεις του λογαριασμού του. Για την ακρίβεια, υποστηρίζεται μονάχα μία στην τρέχουσα υλοποίηση της εφαρμογής. Πιο συγκεκριμένα, η επιλογή του τρίτου στοιχείου του μενού, ανοίγει σε ένα νέο Fragment. Το View αυτό περιλαμβάνει ένα check box, με σχετικό κείμενο στο πλάι. Σε αυτό το σημείο, ο χρήστης μπορεί να δηλώσει αν επιθυμεί να λαμβάνει email σχετικά με εξαφανίσεις κατοικίδιων. Η τιμή του check box αντικατοπτρίζει την τιμή που είναι αποθηκευμένη στο document του αντίστοιχου χρήστη στη βάση. Η αλλαγή της οριστικοποιείται με το πάτημα του κουμπιού αποθήκευσης των αλλαγών. Στην πράξη, η αλλαγή αυτή εξαιρεί τον χρήστη από την λίστα των χρηστών που κατασκευάζεται κατά την διαδικασία αποστολής email στο backend. Η επόμενη επιλογή του μενού προσφέρει πληροφορίες σχετικά με την εφαρμογή καθώς και στοιχεία επικοινωνίας. Η τελευταία επιλογή είναι η αποσύνδεση. Πατώντας σε αυτή, ο χρήστης αποσυνδέεται από τον λογαριασμό του και οδηγείται στην αρχική οθόνη της εφαρμογής, ενώ παράλληλα διαγράφεται το JWT που ήταν αποθηκευμένο στα SharedPreferences της εφαρμογής.

4.6 Υπενθυμίσεις

Ο χρήστης έχει τη δυνατότητα να ορίσει υπενθυμίσεις για ιατρικά συμβάντα του κατοικίδιου του. Η οθόνη των υπενθυμίσεων αποτελεί ένα Fragment, το οποίο διαθέτει ένα RecyclerView για την προβολή όλων τον ενεργών υπενθυμίσεων - τον υπενθυμίσεων δηλαδή, η ημερομηνία εκτέλεσης των οποίων δεν έχει παρέλθει. Η διαδικασία λήψης των δεδομένων είναι η ίδια με αυτή της οθόνης των κατοικίδιων. Κάθε στοιχείο υπενθύμισης του RecyclerView, αποτελείται από τον τύπο της υπενθύμισης, το όνομα του κατοικίδιου, την ημερομηνία αποστολής της υπενθύμισης και ένα κουμπί για τη διαγραφή της. Η διαγραφή μιας υπενθύμισης οδηγεί στην ανανέωση της τιμής της MutableLiveData μεταβλητής που παρακολουθεί το Fragment, γεγονός που οδηγεί στην άμεση αφαίρεση της, σε πραγματικό χρόνο, από την οθόνη. Στην κάτω δεξιά γωνία της οθόνης, βρίσκεται ένα κουμπί για την προσθήκη νέας υπενθύμισης. Το layout του Fragment της νέας υπενθύμισης διαθέτει τέσσερα υποχρεωτικά πεδία. Το πρώτο αποτελεί ένα drop-down list, που περιλαμβάνει τα ονόματα από όλα τα κατοικίδια του χρήστη. Το δεύτερο αποτελεί και αυτό ένα drop-down list, μέσω του οποίου ο χρήστης μπορεί να επιλέξει τον τύπο της υπενθύμισης. Αυτός μπορεί να είναι «Εμβολιασμός», «Αποπαρασίτωση» και «Θεραπεία». Το επόμενο πεδίο αποτελεί ένα date picker, για την επιλογή της επιθυμητής ημερομηνίας υπενθύμισης. Ο χρήστης μπορεί να επιλέξει μόνο μελλοντικές ημερομηνίες, συμπεριλαμβανομένης της τρέχουσας. Το τελευταίο πεδίο είναι ένα time picker, για την επιλογή της ακριβής ώρας της υπενθύμισης. Γίνεται αντιληπτό ότι ο χρήστης μπορεί να ορίσει κάποια μελλοντική ημερομηνία σε ακρίβεια λεπτού.

Για την αποστολή των ειδοποιήσεων από το backend, απαιτούνται τρία συστατικά, το node-cron, το BULL και το AdminSDK του Firebase Cloud Messaging. Αρχικά, ορίζεται ένα cron job, μια συνάρτηση δηλαδή που εκτελείται κάθε λεπτό, χάρις στο node-cron. Η συνάρτηση αυτή επιστρέφει από τη βάση όλες τις υπενθυμίσεις, η ημερομηνία και ώρα εκτέλεσης των οποίων, είτε έχει παρέλθει ή είναι ίση με την τρέχουσα. Αν δεν υπάρχουν υπενθυμίσεις, η συνάρτηση τυπώνει ένα σχετικό μήνυμα και επιστρέφει. Στην αντίθετη περίπτωση, η λίστα με τις υπενθυμίσεις προστίθεται στο queue του BULL για επεξεργασία στο υπόβαθρο. Όταν έρθει η ώρα της εκτέλεσης, εκτελείται η process μέθοδος της ουράς, στην οποία ετοιμάζεται το περιεχόμενο των ειδοποιήσεων και έπειτα καλείται η συνάρτηση sendPushNotification για κάθε ειδοποίηση. Λόγω της προσωποποιημένης φύσης της ειδοποίησης, δεν είναι δυνατή η αποστολή των ειδοποιήσεων σε batches, καθώς αυτή απαιτεί κοινό περιεχόμενο για όλες τις ειδοποιήσεις. Η sendPushNotifications χτίζει το message αντικείμενο, το οποίο περνάει ως

παράμετρο στην FCM.send μέθοδο, που παρέχεται από το AdminSDK του Firebase Cloud Messaging, για να αποσταλεί στην εφαρμογή. Όταν ολοκληρωθεί η αποστολή όλων των ειδοποιήσεων, πυροδοτείται το «complete» event της ουράς του BULL, στο οποίο οι υπενθυμίσεις που εκτελέστηκαν επιτυχώς, θέτονται σε status «sent». Αυτό θα τις αποτρέψει από το να εκτελεστούν ξανά στο μέλλον. Στο Σχήμα 4.6 παρουσιάζεται η μέθοδος αποστολής των push notifications, ενώ στο σχήμα 4.7 φαίνονται οι ειδοποιήσεις κατά την λήψη τους.

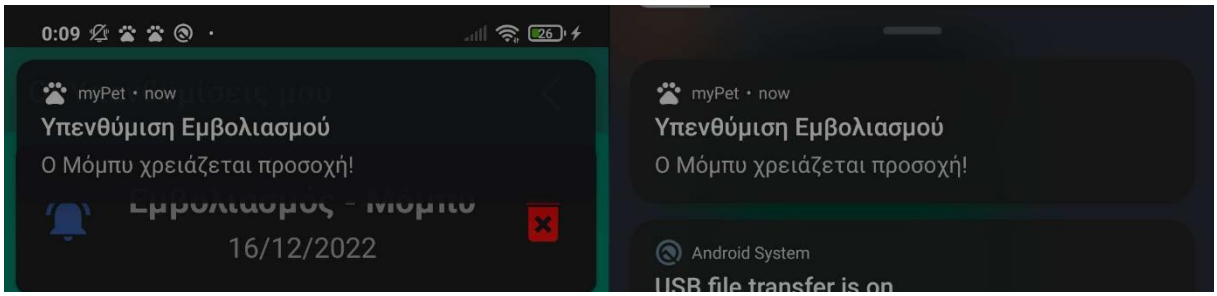
```
sendPushNotification= (fcm_token, title, body) => {
    try{
        let message = {
            android: {
                notification: {
                    title: title,
                    body: body,
                },
            },
            token: fcm_token
        };

        FCM.send(message, function(err, resp) {
            if(err){
                console.log(err.message)
                throw err;
            }else{
                console.log('Successfully sent notification');
            }
        });

        return;

    }catch(err){
        console.log(err.message)
        throw err;
    }
}
```

Σχήμα 4.6: Συνάρτηση αποστολής push notification

4.7 Πλησιέστερα κτηνιατρεία

Επιλέγοντας την προβολή των πλησιέστερων κτηνιατρείων, ανοίγει ένα νέο Activity με έναν χάρτη του google maps. Ο χρήστης, αφού επιτρέψει τη χρήση της τοποθεσίας του από την εφαρμογή, μεταφέρεται στην τρέχουσα τοποθεσία του στον χάρτη. Στο σημείο αυτό, μπορεί να δει όλα τα κτηνιατρεία που είναι καταγεγραμμένα ως επιχειρήσεις στο Google Places, με μαύρα markers – «πινέζες» - στον χάρτη. Κάνοντας zoom out, και εφόσον υπάρχουν περισσότερα από ένα markers σε κοντινή απόσταση, τα markers ομαδοποιούνται σε clusters, τα οποία συμβολίζονται με έναν κύκλο, με το πλήθος των κτηνιατρείων που περιλαμβάνουν, ως τιμή στο κέντρο τους. Πατώντας σε ένα από τα markers, εμφανίζεται ένα πλαίσιο, στο κάτω μέρος της οθόνης, το οποίο περιλαμβάνει πληροφορίες σχετικά με το επιλεγμένο κτηνιατρείο. Οι πληροφορίες αυτές περιλαμβάνουν το όνομα της επιχείρησης, τη διεύθυνση της, το τηλέφωνο επικοινωνίας και την βαθμολογία με βάση τις κριτικές.

Η λειτουργία του χάρτη βασίζεται στο MapsSDK και PlacesSDK. Με την κλήση της OnViewCreated του Fragment, λαμβάνεται μέσω viewbinding το MapView, το view που φιλοξενεί τον χάρτη, και καλείται η getMapAsync μέθοδος του, η οποία είναι υπεύθυνη για τη δημιουργία του. Όταν ο χάρτης είναι έτοιμος, καλείται η onMapReady μέθοδος. Εντός της, καλείται μια μέθοδος για τον ορισμό του ClusterManager και τη διαχείριση των markers, η οποία παραμετροποιείται για να διαχειρίζεται τα onClick events των markers, ενώ ορίζεται και ο renderer, ο οποίος είναι υπεύθυνος για τη σχεδίαση του χάρτη στην οθόνη. Η onClick μέθοδος του κάθε marker, καλεί τη μέθοδο getClusterItemDetails, η οποία μέσω του ViewModel και του Repository, πραγματοποιούν μια API κλήση για τη λήψη των στοιχείων της επιχείρησης. Στο Σχήμα 4.8 φαίνεται η υλοποίηση της μεθόδου setUpClusterManager. Επιπλέον, εφόσον ο χρήστης έχει παραχωρήσει άδεια χρήσης της τοποθεσίας του, με την ίδια λογική της κλήσης για λήψη των στοιχείων της εκάστοτε επιχείρησης, αποστέλλεται ένα αίτημα στο API των Google Places, για την λήψη όλων των επιχειρήσεων σε ακτίνα 10 χιλιομέτρων από την τρέχουσα θέση της συσκευής, με τη λέξη κλειδί “Κτηνιατρείο”. Από κατασκευή του, το συγκεκριμένο API θα προσπαθήσει να βρει τις επιχειρήσεις εντός αυτής της ακτίνας, άλλα υπάρχει περίπτωση να την επεκτείνει και να επιστρέψει περισσότερες, από την ευρύτερη περιοχή. Στο Σχήμα 4.9 παρουσιάζεται η προβολή του χάρτη.



Σχήμα 4.7: Αριστερά η ειδοποίηση ως heads-up και δεξιά η ειδοποίηση στο expanded status bar

4.8 Στοιχεία και ιατρικός φάκελος κατοικίδιου

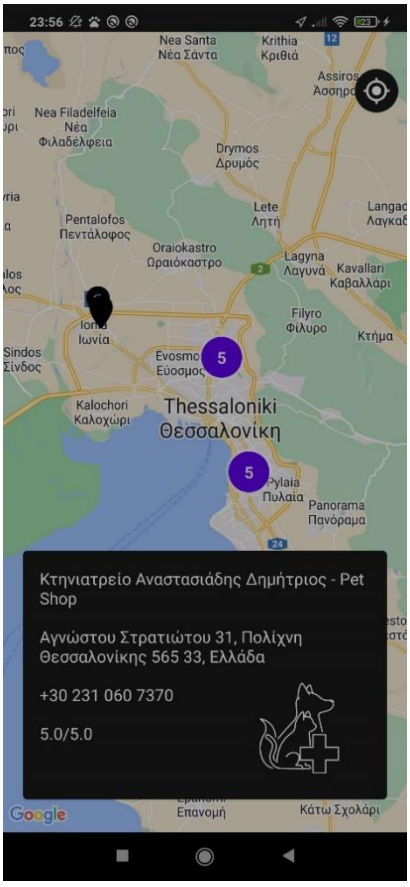
Με το click σε κάθε στοιχείο της λίστας του RecyclerView των κατοικίδιων, ανοίγει ένα νέο Activity, το PetDetaislActivity. Το PetDetailsActivity διαθέτει ένα BottomNavigationView και ένα FragmentContainerView. Το BottomNavigationView προσφέρει έναν εύκολο τρόπο πλοήγησης μεταξύ views, τα οποία φέρουν κύρια σημασία στην εφαρμογή και πρέπει να είναι προσβάσιμα από διάφορα σημεία της. Αποτελείται από μια οριζόντια μπάρα, τα στοιχεία της οποίας αντλούνται από ένα menu. Το click σε κάθε στοιχείο της, ανοίγει ένα Fragment και το τοποθετεί στο FragmentContainerView. Το αποτέλεσμα είναι η προβολή του περιεχομένου κάθε Fragment, απευθείας στην οθόνη, με την αντικατάσταση του προηγούμενου, μέσω γρήγορης εναλλαγής τους από το μενού.

Το BottomNavigationMenu διαθέτει τέσσερις επιλογές. Η πρώτη, και προκαθορισμένη, αφορά τα στοιχεία του κατοικίδιου και αντιστοιχεί στο PetInfoFragment. Το Layout του αποτελείται από ένα CardView, το οποίο με τη σειρά του χωρίζεται σε δύο επιμέρους CardView. Το πρώτο στεγάζει την εικόνα του κατοικίδιου, ενώ το δεύτερο ένα ScrollView, καθώς τα στοιχεία του κατοικίδιου δεν χωρούν εντός του. Στην κάτω δεξιά γωνία, υπάρχουν δύο κουμπιά. Το πρώτο οδηγεί σε ένα νέο Fragment για την επεξεργασία των στοιχείων του κατοικίδιου, συμπεριλαμβανομένης της αλλαγής εικόνας προφίλ. Το δεύτερο κουμπί, ανοίγει ένα DialogFragment, το οποίο περιέχει το qr code του κατοικίδιου. Το qr code παράγεται από το _id του κατοικίδιου, όπως είναι αποθηκευμένο στη βάση. Αυτό σημαίνει πως τα στοιχεία του κατοικίδιου μπορούν να προβληθούν μόνο μέσω της εφαρμογής και μόνο σε άτομα που έχουν συνδεθεί επιτυχώς. Στο Σχήμα 4.10 φαίνεται ο xml κώδικας του PetDetailsActivity layout.

```
private fun setupClusterManager() {
    clusterManager = ClusterManager<MapClusterItem>(requireContext(), googleMap).apply {
        setOnClusterItemClickListener(ClusterManager.OnClusterItemClickListener { item ->
            getClusterItemDetails(item)
            return@OnClusterItemClickListener true
        })
        renderer = MapSearcherClusterRenderer(
            context = requireContext(),
            clusterItemWidth = (resources.screenWidth * CLUSTER_ITEM_WIDTH_PERCENT).toInt(),
            map = googleMap,
            clusterManager = this
        )
    }
}
```

Σχήμα 4.8: Η onMapReady μέθοδος του MapSearcherFragment

Η επιλογή της εικόνας προς ανέβασμα, πραγματοποιείται με το άνοιγμα ενός Activity της εφαρμογής διαχείρισης αρχείων, στο οποίο ορίζεται ο τύπος των αρχείων προς αναζήτηση. Στην προκειμένη περίπτωση, πρόκειται για εικόνες οποιοδήποτε τύπου (image/*). Παράλληλα, έχει τεθεί ένας lifecycle observer, ο οποίος παρακολουθεί το αποτέλεσμα που θα επιστρέψει το Activity, δηλαδή, την εικόνα που θα επιλέξει ο χρήστης. Όταν αυτό λάβει τιμή, επιλέγεται το URI της εικόνας και μέσω αυτού χτίζεται ένα αρχείο με τη χρήση Input και Output stream. Τέλος, από αυτό το αρχείο λαμβάνεται το absolutePath του, το οποίο και περνάει παραμετρικά, μέσω του ViewModel και του Repository, στο αίτημα προς τον server.



Σχήμα 4.9: Προβολή του χάρτη με πληροφορίες κτηνιατρείου και clusters στο υπόβαθρο

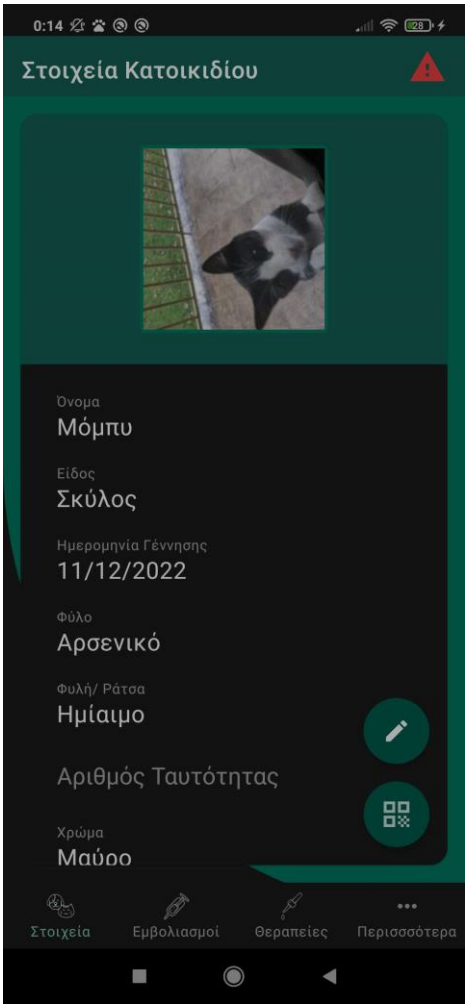
Οι υπόλοιπες επιλογές του BottomNavigationMenu, αφορούν την προβολή και προσθήκη στοιχείων στον ιατρικό φάκελο του κατοικίδιου. Εξαίρεση αποτελεί η τελευταία επιλογή, η οποία, οδηγεί σε ένα Fragment με επιπλέον κατηγορίες του ιατρικού φακέλου. Αυτό συμβαίνει καθώς, προτείνεται, το BottomNavigationMenu να φιλοξενεί από τρεις έως πέντε επιλογές. Οι εμβολιασμοί και οι αντιπαρασιτικές θεραπείες, μαζί με τα στοιχεία του κατοικίδιου, επιλέχθηκαν ως βασικές επιλογές για πληθώρα λόγων. Αρχικά, είναι στοιχεία που ζητούνται πιο συχνά και ο συνδυασμός τους προσφέρει το μεγαλύτερο μέρος της πληροφορίας του κατοικίδιου. Επίσης, οι περισσότερες επισκέψεις στον κτηνίατρο γίνονται για αυτούς τους λόγους. Επομένως, είναι λογικό, να είναι πιο εύκολα και γρήγορα προσβάσιμες τόσο από τον χρήστη όσο και από τον κτηνίατρο. Εννοείται πως δεν υποβαθμίζεται η σημασία των υπόλοιπων στοιχείων, απλά μεταβάλλονται πιο σπάνια και είναι προτιμότερο να «κρύβονται» πίσω από ένα επιπλέον βήμα.

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/bottomNavigationView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        app:labelVisibilityMode="labeled"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:menu="@menu/bottom_navigation_menu" />

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/navigationFragmentContainer"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/bottomNavigationView"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Σχήμα 4.10: XML κώδικας του layout του PetDetailsActivity

Όλες οι κατηγορίες του ιατρικού φακέλου - εμβολιασμοί, αντιπαρασιτικές θεραπείες, γενικές θεραπείες, εγχειρήσεις και διαγνωστικοί έλεγχοι - ακολουθούν την ίδια δομή, τόσο στην εμφάνιση τους και στην αλληλεπίδραση τους με τον χρήστη, όσο και στην λειτουργία των αιτημάτων προς τον server, για την προβολή και προσθήκη τους. Αρχικά, πραγματοποιείται το εκάστοτε αίτημα λήψης τους στον server. Έπειτα, εντός του Fragment, παρακολουθείται η μεταβλητή του PetsViewModel, η οποία θα λάβει τιμή όταν ληφθεί η απάντηση του server μέσω του Retorfit, εντός του PetsRepository. Όταν συμβεί αυτό, λαμβάνουν τιμές τα στοιχεία του RecyclerView, που έχει οριστεί για την προβολή τους, και ορίζεται το onClick event, το οποίο παρέχει μια λεπτομερή προβολή, σε νέο Fragment, των στοιχείων της εκάστοτε επιλογής. Αντίστοιχα με τις υπόλοιπες περιπτώσεις της εφαρμογής, η προσθήκη νέου στοιχείου, σε οποιαδήποτε κατηγορία, ακολουθεί την ίδια λογική, με την επιτυχή προσθήκη να ανακατευθύνει τον χρήστη στην οθόνη της λίστας της εν λόγω κατηγορίας του φακέλου, ενώ, για την αντίθετη περίπτωση,

εμφανίζεται ένα ενημερωτικό Toast στο κάτω μέρος της οθόνης, που φέρει μήνυμα σχετικό με το σφάλμα που προέκυψε. Στο Σχήμα 4.11 φαίνεται η προβολή των λεπτομερειών του κατοικίδιου.



Σχήμα 4.11: Προβολή των λεπτομερειών του κατοικίδιου, με εμφανές το BottomNavigationMenu στο κάτω μέρος της οθόνης

4.9 Δήλωση εξαφάνισης κατοικίδιου

Στην άνω δεξιά γωνία του PetDetailsActivity, βρίσκεται ένα κουμπί σε σχήμα κόκκινου τριγώνου. Πατώντας, εμφανίζεται ένα DialogFragment που προειδοποιεί τον χρήστη ότι πρόκειται να δηλώσει το κατοικίδιο του ως εξαφανισμένο. Με την αποδοχή, μεταφέρεται στο MissingPetFragment. Εδώ, έχει τη δυνατότητα να δηλώσει την περιοχή πιθανής εξαφάνισης του κατοικίδιου του καθώς και στοιχεία επικοινωνίας. Πιο συγκεκριμένα, πατώντας το κουμπί «Άνοιγμα Χάρτη"», ανοίγει ένα νέο Activity με ένα Fragment, το PointFinderFragment. Το Fragment αυτό φιλοξενεί τον χάρτη και η δημιουργία και εμφάνιση του είναι κοινή με την περίπτωση του χάρτη των πλησιέστερων κτηνιατρείων. Ωστόσο, στην προκειμένη περίπτωση, δεν υπάρχουν κλήσεις σε APIs της google, ούτε προβάλλονται markers. Ο χάρτης αυτός διαθέτει ένα onClick event το οποίο δημιουργεί έναν marker στο σημείο επαφής. Επιπλέον, αποθηκεύονται σε δύο μεταβλητές οι συντεταγμένες του σημείου που επιλέχθηκε. Όταν ο χρήστης αποφασίσει το σημείο που επιθυμεί να επιλέξει, μπορεί να πατήσει στο κουμπί «Επιλογή», το οποίο θα κλείσει τον χάρτη και θα επιστρέψει τις τιμές των συντεταγμένων. Έπειτα, ο χρήστης καλείται να συμπληρώσει τουλάχιστον ένα στοιχείο επικοινωνίας πριν μπορέσει να προχωρήσει. Με την επιτυχή εισαγωγή του και το πάτημα του κουμπιού της υποβολής, εμφανίζεται ένα DialogFragment που τον

ενημερώνει πως τα στοιχεία επικοινωνίας του θα κοινοποιηθούν σε τρίτους και παράλληλα, ζητά τη συναίνεση του. Σε αυτήν την περίπτωση, αποστέλλεται ένα αίτημα στον server και η εφαρμογή ανακατευθύνει τον χρήστη στην οθόνη στοιχείων του κατοικίδιου του, όπου μπορεί να παρατηρήσει πως, πλέον, υπάρχει ένα κόκκινο περίγραμμα γύρω από το κατοικίδιο του, υποδεικνύοντας ότι έχει δηλωθεί ως εξαφανισμένο. Όταν, και αν, ο χρήστης επιθυμήσει να δηλώσει την ανεύρεση του κατοικίδιου του, επαναλαμβάνει τη διαδικασία, μόνο που σε αυτήν την περίπτωση, το κουμπί δήλωσης εξαφάνισης, του εμφανίζει ένα αντίστοιχο μήνυμα, η αποδοχή του οποίου θα μεταβάλλει την κατάσταση του κατοικίδιου στην αρχική της μορφή.

Στην πλευρά του server, μόλις ληφθεί το αίτημα, υποβάλλεται προς επεξεργασία στο υπόβαθρο η εργασία αποστολής email. Αρχικά, βρίσκονται τα email των χρηστών που επιθυμούν να ενημερώνονται για σχετικά γεγονότα. Έπειτα, με τη χρήση ενός npm πακέτου, του node-mailer, δημιουργείται ένα Transport αντικείμενο, το οποίο διαθέτει παραμέτρους για τον ορισμό του host που θα εξυπηρετήσει την αποστολή, του port στο οποίο δέχεται τα αιτήματα, καθώς και στοιχεία αυθεντικοποίησης του χρήστη από τον οποίο θα γίνει η αποστολή email, όπως το username και το password. Κατόπιν, ορίζεται ένα αντικείμενο το οποίο φέρει πληροφορίες σχετικά με τον αποστολέα, το θέμα και το περιεχόμενο του email. Για το περιεχόμενο του email υποστηρίζεται η χρήση html. Η μέθοδος buildHtmlContent, δέχεται τα στοιχεία του κατοικίδιου, τις συντεταγμένες και τα στοιχεία επικοινωνίας που δηλώθηκαν από τον χρήστη, και χτίζει το περιεχόμενο του email. Η συντεταγμένες χρησιμοποιούνται ως query παράμετροι στο StaticMapsAPI της google, το οποίο επιστρέφει μια στατική εικόνα του χάρτη, ανάλογα με την περιοχή που ζητήθηκε. Άλλες παράμετροι αφορούν το επίπεδο zoom του χάρτη και τις διαστάσεις του. Επιπλέον, στον χάρτη σχηματίζεται ένας κύκλος, ακτίνας 1.5 χιλιομέτρου από το σημείο εξαφάνισης, που υποδεικνύει την πιθανή θέση του κατοικίδιου. Αφού ολοκληρωθεί η παραμετροποίηση του περιεχομένου, τα email αποστέλλονται σε ομάδες των 100 παραληπτών τη φορά. Στο Σχήμα 4.12 φαίνεται ο κώδικας για την παραμετροποίηση του node-mailer και στο Σχήμα 4.13 ένα παράδειγμα λήψης email.

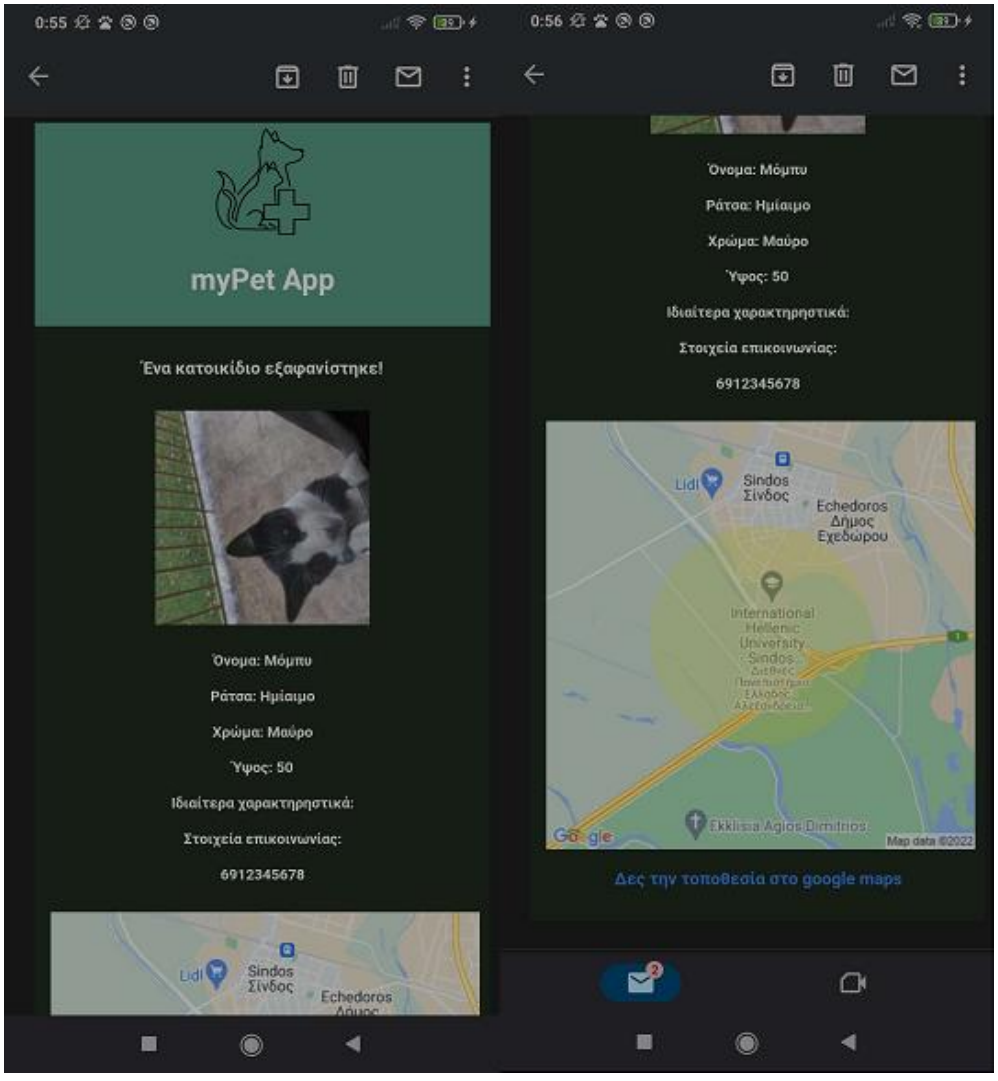
```
var transporter = nodemailer.createTransport({
    service: 'gmail',
    auth: {
        user: process.env.GOOGLE_USERNAME,
        pass: process.env.GOOGLE_APP_PASSWORD
    },
});

var mailOptions = {
    from : process.env.GOOGLE_USERNAME,
    subject : 'MyPet - Εξαφάνιση κατοικιδίου',
    html : BuildHtmlContent(job.data.pet, job.data.coordinates, job.data.contactInfo)
};
```

Σχήμα 4.12: Παραμετροποίηση του node-mailer

4.10 Έλεγχος QR Code

Με την επιτυχή σύνδεση του, ο κτηνίατρος μεταφέρεται στην κεντρική οθόνη του κτηνιατρικού μενού. Εκεί, διαθέτει μονάχα μια επιλογή - να ελέγξει το qr code ενός κατοικίδιου. Πατώντας το κουμπί «Σκανάρετε qr code», ανοίγει ένα Fragment, στο layout του οποίου υπάρχει ένα CodeScannerView. Το view αυτό παρέχεται από την code-scanner βιβλιοθήκη και βασίζεται στο zxing project. Στο Fragment, αφού ληφθεί η αναφορά του view μέσο view binding και δημιουργηθεί ένα CodeScanner αντικείμενο

μέσω αυτής, υλοποιείται η decodeCallback μέθοδος του αντικειμένου, στην οποία ορίζονται οι ενέργειες της επιτυχούς αναγνώρισης του qr code. Σε αυτήν την περίπτωση, καλείται το PetDetailsActivity, το οποία περιεγράφηκε σε προηγούμενο κεφάλαιο. Για λόγους ευκολίας, επαναχρησιμοποιήθηκαν τα Fragments και Activites του κατοικίδιου, καθώς οι διαθέσιμες ενέργειες του κτηνιάτρου αποτελούν ένα υποσύνολο των διαθέσιμων ενεργειών του χρήστη. Ένας κτηνίατρος μπορεί να πλοηγηθεί μέσω του BottomNavigation μενού, στον ιατρικό φάκελο του κατοικίδιου και να προσθέσει νέες στοιχεία σε κάθε κατηγορία. Σε κάθε νέα εγγραφή που προσθέτει, προστίθεται και το όνομά του ως απόδειξη προσθήκης από αυτόν. Δεν μπορεί, όμως, να δηλώσει το κατοικίδιο ως εξαφανισμένο ούτε να επεξεργαστεί τα προσωπικά του στοιχεία. Όλες αυτές οι επιλογές έχουν απενεργοποιηθεί. Επιπλέον, οι χρωματισμοί των μενού είναι διαφορετικοί, καθώς χρησιμοποιούν μπλε αποχρώσεις. Ο έλεγχος για την επιβολή όλων αυτών των αλλαγών στο layout βασίζεται στην αναγνώριση σύνδεσης κτηνιάτρου στην εφαρμογή. Πιο συγκεκριμένα, όταν ένας κτηνίατρος συνδέεται, αποθηκεύονται στο SharedPreferences το όνομα και το επίθετο του κτηνιάτρου. Όταν ελεγχθεί επιτυχώς το qr code ενός κατοικίδιου, και για τα υπόλοιπα layouts του κατοικίδιου, ελέγχεται η ύπαρξη αυτών των στοιχείων. Αν υπάρχουν, τότε στην εφαρμογή έχει συνδεθεί ένας κτηνίατρος και εφαρμόζονται οι αντίστοιχοι περιορισμοί, με την απόκρυψη και απενεργοποίηση συγκεκριμένων view και την αλλαγή της μορφοποίησης τους.



Σχήμα 4.13: Παράδειγμα λήψης email

4.11 Επίλογος

Στο κεφάλαιο αυτό έγινε παρουσίαση όλων των μερών της εφαρμογής που αναπτύχθηκε. Η εφαρμογή, τόσο η εφαρμογή Android, όσο και ο server, χωρίστηκε σε επιμέρους κατηγορίες, με την ομαδοποίηση των χαρακτηριστικών τους υπό κοινές, θεματικά και λειτουργικά, ενότητες. Αρχικά, έγινε επισκόπηση της ροής των αιτημάτων και των απαντήσεων, κάνοντας ξεκάθαρο τον τρόπο που επικοινωνεί η Android εφαρμογή με τον server. Στη συνέχεια, αναλύθηκε η δημιουργία λογαριασμού στην εφαρμογή και η σύνδεση, του χρήστη, αλλά και του κτηνιάτρου. Παρουσιάστηκαν οι επιλογές που έχει ο χρήστης, σε επίπεδο λογαριασμού. Αυτές αφορούν τη ρύθμιση υπενθυμίσεων για ιατρικά γεγονότα του κατοικίδιου, την προβολή των πλησιέστερων κτηνιατρείων στον χάρτη αλλά και την αλλαγή των ρυθμίσεων του λογαριασμού. Επίσης, περιεγράφηκε ο ιατρικός φάκελος του κατοικίδιου και ο τρόπος πλοήγησης και επεξεργασίας αυτού, σε επίπεδο σχεδιαστικό αλλά και λειτουργικό. Αναλύθηκε εκτενώς το μέρος της δήλωσης εξαφάνισης ενός κατοικίδιου, το οποίο απαιτεί τη συνεργασία πολλών εργαλείων και τεχνολογιών για να επιτευχθεί. Τέλος, αναφέρθηκε ο ρόλος του κτηνιάτρου και περιεγράφηκε ο τρόπος χρήσης του QR code, το οποίο αποτελεί το μέσο με το οποίο ο κτηνίατρος προβάλει τα στοιχεία του κατοικίδιου-ασθενή.

Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης

Η παρούσα εφαρμογή στο σύνολο της, πέρα από μια διπλωματική εργασία, αποτελεί και ένα προσωπικό όραμα, που πηγάζει από την επιθυμία μου να προσφέρω τη δική μου προσέγγιση για τη λύση υπαρκτών προβλημάτων που απασχολούν, όχι μόνο εμένα, αλλά και πολλά μέλη της κοινωνίας. Οι κύριοι στόχοι της εργασίας – η ψηφιοποίηση του βιβλιαρίου και η επικοινωνία της εξαφάνισης ενός κατοικίδιου – πραγματοποιήθηκαν πλήρως. Ωστόσο, υπάρχει πληθώρα βελτιώσεων και προσθηκών που μπορούν να πραγματοποιηθούν, οι οποίες αφορούν το όραμα μου για την μελλοντική ανάπτυξη της εφαρμογής. Αυτό δεν ακυρώνει ούτε υποτιμά τον χρόνο και την προσπάθεια που έχει αφιερωθεί στην ανάπτυξη της. Είναι, ωστόσο, ορθό και δίκαιο να αναγνωριστούν τα σημεία στα οποία υστερεί η τρέχουσα έκδοσή της και να αναφερθούν προτάσεις και λύσεις προς μελλοντική υλοποίηση.

Αρχικά, η σύνδεση αλλά και η εγγραφή στην εφαρμογή θα μπορούσε να γίνεται μέσω κάποιας τρίτης, έμπιστης εφαρμογής, όπως για παράδειγμα, σύνδεση με google ή facebook. Η επιλογή αυτή είναι ιδιαίτερα βολική για τον χρήστη, καθώς θα έχει ακόμα έναν λιγότερο λογαριασμό για να απομνημονεύσει. Επιπλέον, η ενσωμάτωση με το facebook ή λοιπά κοινωνικά δίκτυα, μπορεί να δώσει τη δυνατότητα στον χρήστη για τον διαμοιρασμό κάποιας πιθανής εξαφάνισης κατοικίδιου και σε αυτές τις πλατφόρμες, διευρύνοντας σημαντικά το κοινό στο οποίο κοινοποιείται η ειδοποίηση. Είναι λογικό πως, σε τέτοιες περιπτώσεις, όσο μεγαλύτερο το πλήθος των ατόμων που ενημερώνονται για την εξαφάνιση, τόσο πιο πιθανή είναι η έγκαιρη ανεύρεση του εν λόγω κατοικίδιου.

Ένα ακόμα χαρακτηριστικό αφορά τη χρήση QR code. Ο χρήστης θα μπορεί να παράγει ένα προσαρμοσμένο QR code, το οποίο θα περιέχει στοιχεία του κατοικίδιου που θα επιλέξει ο ίδιος. Το QR code αυτό θα μπορεί να εξαχθεί σε διάφορα μεγέθη και να εκτυπωθεί, ώστε να μπορεί να τοποθετηθεί σε μορφή κρεμαστού «μπρελόκ» στο κολάρο του κατοικίδιου. Όταν κάποιος ελέγχει το QR code μέσω της εφαρμογής, θα μπορεί να δει τα στοιχεία αυτά. Στην περίπτωση όμως που το κατοικίδιο έχει δηλωθεί ως εξαφανισμένο, θα του προβάλλεται ένα διαφορετικό σύνολο πληροφοριών, που ο ίδιος θα έχει επίσης δηλώσει, πιο στοχευμένων και προσωπικών, όπως τα στοιχεία επικοινωνίας του ιδιοκτήτη. Η προβολή των προσωπικών πληροφοριών θα προϋποθέτει τη συγκατάθεση του χρήστη. Εναλλακτικά, ο χρήστης θα μπορεί να επιλέξει ένα σύνολο στατικών πληροφοριών που θα θέλει να προβάλλονται μέσω του QR code, οι οποίες θα είναι αναγνώσιμες μέσω οποιουδήποτε QR reader, και όχι δεσμευμένες για προβολή αποκλειστικά μέσω της εφαρμογής.

Επιπλέον, η δημιουργία υπενθυμίσεων για γεγονότα υγείας του κατοικίδιου θα μπορούσε να αυτοματοποιηθεί περαιτέρω, καθώς ενέργειες όπως ο εμβολιασμός, έχουν σταθερά διαστήματα μεταξύ διαδοχικών ενεργειών. Οι ειδοποιήσεις θα μπορούσαν να αποστέλλονται και μέσω email, αν το επιθυμεί ο χρήστης.

Επίσης, είναι συνετό να κατασκευαστεί ένας μηχανισμός απαγόρευσης spam email από χρήστες που, ενδεχομένως να επιθυμούν να εκμεταλλευτούν την εφαρμογή για δικούς τους λόγους. Ο μηχανισμός αυτός θα πρέπει να θέτει όρια στην ημερήσια δήλωση εξαφανίσεων σε επίπεδο λογαριασμού, για να αποφευχθούν τέτοιου είδους περιστατικά.

Άλλες προσθήκες αφορούν τις λειτουργίες του κτηνιάτρου και συγκεκριμένα, την επέκταση τους. Στην τωρινή έκδοση της εφαρμογής, ο ρόλος του κτηνιάτρου περιορίζεται στην προβολή των στοιχείων του κατοικίδιου μέσω ανάγνωσης του QR code και στην προσθήκη εμβολιασμών, θεραπειών και λοιπών στοιχείων στον ιατρικό φάκελο του κατοικίδιου. Ενδεικτικά, ο κτηνίατρος θα μπορεί να διατηρεί ιστορικό των επισκέψεων με δυνατότητα γρήγορης αναζήτησης για ευκολότερη πρόσβαση σε

περίπτωση επόμενης επίσκεψης εξέτασης του κατοικίδιου. Παράλληλα, ο χρήστης θα μπορεί να αποθηκεύει καρτέλες κτηνιάτρων, που έχει επισκεφθεί, με τα στοιχεία επικοινωνίας τους και τη διεύθυνση του ιατρείου τους.

Μια ακόμη λειτουργία η οποία δεν υποστηρίζεται προς το παρόν και η οποία θα βελτίωνε την εφαρμογή, είναι η δυνατότητα τροποποίησης των προσωπικών πληροφοριών του χρήστη και του κτηνιάτρου. Η λειτουργία αυτή θα αφορά κυρίως την επεξεργασία των στοιχείων επικοινωνίας και της διεύθυνσης κατοικίας των χρηστών και του ιατρείου του κτηνιάτρου, ενώ επιπλέον τροποποιήσεις θα αφορούν την αλλαγή του κωδικού πρόσβασης.

Επιπροσθέτως, μπορεί να εισαχθεί ένας νέος τρόπος κοινοποίησης και προβολής ειδοποιήσεων σχετικά με εξαφανίσεις κατοικίδιων και πιο στοχευμένη πληροφόρηση με τον ορισμό ενός εικονικού ορίου γύρω από μια καθορισμένη τοποθεσία σε λογισμικό με δυνατότητα GPS ή RFID (geofence). Το geofence αποτελεί μια τεχνολογία η οποία συνδυάζει τη γνώση για την τρέχουσα θέση του χρήστη με την εγγύτητα του σε τοποθεσίες που μπορεί να τον ενδιαφέρουν [17]. Αρχικά ορίζεται η τοποθεσία που φέρει ενδιαφέρον μέσω των συντεταγμένων της και στη συνέχεια δημιουργείται μια περιοχή με βάση την ακτίνα-απόσταση από το επίκεντρο του σημείου. Η είσοδος, η έξοδος και η παραμονή στην περιοχή αυτή από τρίτους, πυροδοτεί συγκεκριμένα events τα οποία διαχειρίζονται την εν λόγω κατάσταση. Στην πράξη, όταν ένα κατοικίδιο δηλώνεται ως εξαφανισμένο, θα δημιουργείται ένα geofence με κέντρο τη συσκευή του χρήστη που δήλωσε την εξαφάνιση. Οποιαδήποτε συσκευή φέρει την εφαρμογή και εισέλθει στην περιοχή του geofence, θα λαμβάνει σε πραγματικό χρόνο μια σχετική ειδοποίηση. Προβάλλοντας την, θα μπορεί να δει, με λεπτομέρεια, τα στοιχεία της εξαφάνισης, όπως την περιοχή που συνέβη στον χάρτη και τα στοιχεία του κατοικίδιου. Επιπλέον, για την διατήρηση αυτής της πληροφορίας, καθώς οι ειδοποιήσεις από τη φύση τους είναι εφήμερες, θα δημιουργηθεί ένα inbox για κάθε λογαριασμό και το οποίο θα στεγάζει όλες τις ειδοποιήσεις που έχουν ληφθεί.

Μία ακόμη λειτουργία που θα μπορούσε να προστεθεί αφορά την ύπαρξη ενός πίνακα χαμένων κατοικίδιων. Ο χρήστης, μέσω αυτής της οθόνης, θα μπορεί να προβάλει όλα τα κατοικίδια που έχουν δηλωθεί ως εξαφανισμένα. Επίσης, θα υπάρχει και η δυνατότητα προβολής των εξαφανίσεων στον χάρτη, ενώ ο χρήστης θα μπορεί να προβάλει τις πληροφορίες τους με την επιλογή της αντίστοιχης εξαφάνισης.

Είναι πολύ σημαντικό η εφαρμογή να γίνει ακόμη πιο φιλική προς το χρήστη. Ο χρήστης θα πρέπει να είναι σε θέση να καταλάβει με τι δεδομένα θα πρέπει να τροφοδοτήσει τις φόρμες που συμπληρώνει και τι περιορισμούς έχει. Παρόλο που υπάρχει επικύρωση των δεδομένων (validations) κατά την υποβολή οποιασδήποτε φόρμας, με εμφάνιση μηνυμάτων λάθους και αποτροπή μη επιθυμητών πράξεων από πλευράς του χρήστη, θα ήταν ακόμα πιο αποτελεσματική η ύπαρξη κάποιου ιδιαίτερου μαθήματος (tutorial) ή υλικού για την εκπαίδευση του χρήστη της εφαρμογής ή έστω η δυνατότητα προβολής βοήθειας και πληροφοριών σχετικά με τις επιλογές που του παρουσιάζονται, οδηγώντας τον στην επιτυχή ολοκλήρωση της εκάστοτε ενέργειας.

Μια ακόμη βελτίωση αφορά την πολυγλωσσικότητα της εφαρμογής. Η εφαρμογή αυτή τη στιγμή υποστηρίζει μόνο τα Ελληνικά ως γλώσσα. Μελλοντικά, θα υποστηρίζονται και τα Αγγλικά, ενώ η εναλλαγή τους θα γίνεται μέσω ενός κουμπιού σε σχετικό σημείο της εφαρμογής.

Τέλος, μπορούν να γίνουν και διάφορες βελτιστοποιήσεις αλλά και ενσωματώσεις νέων χαρακτηριστικών των πλατφορμών, τόσο στο backend όσο και στο frontend. Ενδεικτικά, το backend θα μπορούσε να χρησιμοποιεί caching για να επιταχύνει τον χρόνο απόκρισης των αιτημάτων και να αποσυμφορήσει την βάση δεδομένων σε περιπτώσεις υψηλού φόρτου. Όσον αφορά την Android

εφαρμογή, η χρήση Dependency Injection Containers [61] με την ενσωμάτωση βιβλιοθηκών όπως το Hilt [62], για την τμηματοποίηση και ευκολότερη επαναχρησιμοποίηση τμημάτων κώδικα, αποτελεί μια προσθήκη που διερευνάται ενεργά. Επιπλέον, η χρήση της βιβλιοθήκης Jetpack Compose [63] για την προσαρμογή ελέγχων (controls) και συστατικών (components) στο λειτουργικό σύστημα του κάθε χρήστη (native UI), είναι ακόμα μια προσθήκη που έχει δρομολογηθεί για το κοντινό μέλλον.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Christensson. (2020, April 11). *Backend Definition*. [Online]. Available: https://techterms.com

[2] Mozilla and individual contributors (2005-2022). *What is JavaScript?*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

[3] Joyent, Inc. *About Node.js.* [Online]. Available: https://nodejs.org/en/about/#about-node-js

[4] Jonas Schmedtmann, Node.js, Express, mongoDB & More - The Complete Bootcamp 2021, *July 2021*. Accessed on: Oct 16, 2021. [Streaming video]. Available: udemy.com

[5] Ahmad Nassri, (2020, April 14). *So long, and thanks for all the packages!*. [Online]. Available: https://blog.npmjs.org/post/615388323067854848/so-long-and-thanks-for-all-the-packages.html

[6] OpenJS Foundation (2017). *Express Fast, unopinionated, minimalist web framework for Node.js*. [Online]. Available: https://expressjs.com/

[7] Microsoft (2022). *Visual Studio Code.* [Online]. Available: https://code.visualstudio.com/

[8] Android Open Source Project. (2021, Sep. 27). *Set up for Android Development*. [Online]. Available: https://source.Android.com/setup/intro\

[9] Encyclopedia Britanica. (2020, Aug. 27). *Android operating system* [Online]. Available: https://www.britannica.com/technology/Android-operating-system

[10] David Curry. (2022, Jan. 11). *Android Statistics (2022)*. [Online]. Available: https://www.businessofapps.com/data/Android-statistics/

[11] Android Open Source Project. (2021, Aug. 30). *Platform Architecture*. [Online]. Available: https://developer.Android.com/guide/platform

[12] Android Open Source Project. (2022, Jan. 10). *Android Architecture.* [Online]. Available: https://source.Android.com/devices/architecture

[13] Martin Heller. (2020, Mar. 23). *What is Kotlin? The Java alternative explained.* [Online]. Available: https://www.infoworld.com/article/3224868/what-is-kotlin-the-java-alternative-explained.html

[14] Kotlin Foundation. (2021, Sep. 12). *Coroutines basics.* [Online]. Available: https://kotlinlang.org/docs/coroutines-basics.html

[15] Andrius Baruckis. (2019, May 11). *How to handle RESTful web Services using Retrofit, OkHttp, Gson, Glide and Coroutines.* [Online]. Available: https://www.freecodecamp.org/news/kriptofolio-app-series-part-5/

[16] Google Developers. (2022, Feb. 02). *Google Maps Platform*. [Online]. Available: https://developers.google.com/maps/documentation/places/Android-sdk/overview

[17] Android Open Source Project. (2022. Jan. 27). *Create and monitor geofences.* [Online]. Available: https://developer.Android.com/training/location/geofencing

[18] Google Developers. (2022, Feb. 01). *FCM Architectural Overview.* [Online]. Available: https://firebase.google.com/docs/cloud-messaging/fcm-architecture

[19] Korbin Brown. (2019, Nov. 13). *What Is GitHub, and What Is It Used For?* [Online]. Available: https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/

[20] Google Developers, (2022, Feb. 01). *Introduction to Activities* [Online]. Available: https://developer.Android.com/guide/components/Activities/intro-Activities

[21] Google Developers, (2022, Feb. 01). *Activity* [Online]. Available: https://developer.Android.com/reference/Android/app/Activity#Activity-lifecycle

[22] Google Developers, (2022, Feb. 02). *Fragment* [Online]. Available: https://developer.Android.com/reference/Android/app/Fragment

[23] Chike Mgbemena, (2017, Jun. 21). *Android Design Patterns: The Observer Pattern.* [Online]. Available: https://code.tutsplus.com/tutorials/Android-design-patterns-the-observer-pattern--cms-28963

[24] Google Developers, (2022, Feb. 04). L*iveData Overview* [Online]. Available: https://developer.Android.com/topic/libraries/architecture/livedata

[25] Abhishek Srivastava (2020, Aug. 08). *Android SingleLiveEvent (of LiveData) for UI Event*. [Online]. Available: https://abhiappmobiledeveloper.medium.com/Android-singleliveevent-of-livedata-for-ui-event-35d0c58512da

[26] Mitch Tabian, REST API with MVVM and Retrofit2, *August 2021.* Accessed on: Oct 22, 2021. [Streaming video]. Available: https://codingwithmitch.com/courses/rest-api-mvvm-retrofit2/

[27] Risu Mishra (2022, Jun. 09). *MVVM (Model View ViewModel) Architecture Pattern in Android* [Online]. Available: https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-Android/

[28] Mitch Tabian (2018). *Model View View-Model (MVVM): Getting Started.* [Online]. Available: https://codingwithmitch.com/blog/getting-started-with-mvvm-Android/

[29] Google Developers, (2022, Feb. 04). *ViewModel Overview* [Online]. Available: https://developer.Android.com/topic/libraries/architecture/viewmodel

[30] Google Developers, (2022, Feb. 05). *View Binding* [Online]. Available: https://developer.Android.com/topic/libraries/view-binding

[31] Google Developers, (2022, Feb. 05). *Data Binding Library* [Online]. Available: https://developer.Android.com/topic/libraries/data-binding

[32] Google Developers, (2022, Feb. 01). *Analyze your build with APK Analyzer* [Online]. Available: https://developer.Android.com/studio/debug/apk-analyzer

[33] OptimalBits, (2022). *BULL* [Online]. Available: https://optimalbits.github.io/bull/

[34] redis, (2022). *Introduction to Redis* [Online]. Available: https://redis.io/docs/about/

[35] Internet Engineering Task Force (IETF), (2015, May). *JSON Web Token (JWT)* [Online]. Available: https://www.rfc-editor.org/rfc/rfc7519

[36] Zeder, Melinda A. "Core questions in domestication research". *Proceedings of the National Academy of Sciences*, vol. 112, issue 11, pp.3191-3198, Mar. 2015

[37] Hai Cheng et al. "Timing and structure of the Younger Dryas event and its underlying climate dynamics". *Proceedings of the National Academy of Sciences*, vol. 117, issue 38, Sep. 2020

[38] McHugo, G.P., Dover, M.J. & MacHugh, D.E. “Unlocking the origins and biology of domestic animals using ancient DNA and paleogenomics”. *BMC Biology*, vol. 17, issue 98, 2019

[39] Callaway, E. “Dog's dinner was key to domestication”. *Nature*, 2013

[40] Omri Wallach, (2022, Jul. 22). *Timeline: The Domestication of Animals* [Online]. Available: https://www.visualcapitalist.com/the-domestication-of-animals/

[41] Natasha Daly, (2019, Jul. 04). *Domesticated animals, explained* [Online]. Available: https://www.nationalgeographic.com/animals/article/domesticated-animals

[42] Bridgett M. vonHoldth et al., “Structural variants in genes associated with human Williams-Beuren syndrome underlie stereotypical hypersociability in domestic dogs”. *Science Advances*, vol. 3, issue 17, Jul. 2017

[43] Zeder, Melinda A. “The domestication of animals”. *Journal of Anthropological Research*, vol. 68, no. 2, pp.161-190, 2012

[44] Lawrence Robinson, Kai Lundgren, and Robert Segal, M.A., (2022, Nov. 17). *The Health and Mood-Boosting Benefits of Pets*, [Online]. Available: https://www.helpguide.org/articles/mental-health/mood-boosting-power-of-dogs.htm

[45] Υπουργείο Εσωτερικών, “Νέο πλαίσιο για την ευζωία των ζώων συντροφιάς - Πρόγραμμα ‘‘ΑΡΓΟΣ’’”, Εφημερίδα της Κυβερνήσεως της Ελληνικής Δημοκρατίας, vol. Α’ 169, pp.9429-9476, Feb. 2021

[46] Υπουργείο Εσωτερικών, “Καθορισμός όρων λειτουργίας του Εθνικού Μητρώου Ζώων Συντροφιάς και των Υπομητρώων του.”, Εφημερίδα της Κυβερνήσεως της Ελληνικής Δημοκρατίας, vol. Β’ 203, pp.2103-2112, Jan. 2022

[47] Μαρία Λιλιοπούλου, (2022, Aug. 07). *Αλλάζουν όλα με το pet.gov.gr και το... Mypet–pass: Τι είναι και τι θα περιλαμβάνει το Εθνικό Μητρώο Ζώων Συντροφιάς,* [Online]. Available: https://www.ethnos.gr/greece/article/218892/allazoynolametopetgovgrkaitomypetpasstieinaikaitithaperilambaneitoethnikomhtroozoonsyntrofias

[48] Chewy, Inc. (2016). *Chewy* (Version 22.23.7) [Mobile app]. Available: https://play.google.com/store/apps/details?id=com.chewy.Android&hl=en&gl=US&pli=1

[49] American Red Cross (2013). *Pet First Aid: American Red Cross* (Version 2.7.0) [Mobile app]. Available: https://play.google.com/store/apps/details?id=com.cube.arc.pfa&hl=en&gl=US

[50] ARX.NET S.A. (2021). *AlmaPet* (Version 1.0.2) [Mobile app]. Available: https://play.google.com/store/apps/details?id=net.arx.almapet

[51] Rover.com (2013). *Rover – Dog Boarding & Walking* (Version 22.1207.01) [Mobile app]. Available: https://play.google.com/store/apps/details?id=com.rover.Android&hl=en&gl=US

[52] Wag Labs, Inc. (2016). *Wag! – Dog Walkers & Sitters* (Version 3.38.0) [Mobile app]. Available: https://play.google.com/store/apps/details?id=com.ionicframework.wagAndroid554504&hl=en&gl=US

[53] 11 Pets Ltd (2015). *11pets: Pet care* (Version 5.005.010) [Mobile app]. Available: https://play.google.com/store/apps/details?id=com.m11pets.elevenpets&hl=en&gl=US

[54] VitusVet (2015). *VitusVet: Pet health Care App* (Version 3.42) [Mobile app]. Available: https://play.google.com/store/apps/details?id=com.vitusvet.Android&hl=en&gl=US

[55] Petable (2014). *Petable* (Version 3.1.2) [Mobile app]. Available: https://play.google.com/store/apps/details?id=pt.eurom.vetapp&hl=en&gl=US

[56] Metamorphosis, Inc. (2014). *Great Pet Care* (Version 5.0.0) [Mobile app]. Available: https://play.google.com/store/apps/details?id=com.pawprint.mobile&hl=en&gl=US

[57] PetDesk (2013). PetDesk – Pet Health Reminders (Version 8.12) [Mobile app]. Available: https://play.google.com/store/apps/details?id=com.locai.petpartner&hl=en&gl=US

[58] Tractive (2012-2022). *Tractive* [Commercial product]. Available: https://tractive.com/?gclid=CjwKCAiAy_CcBhBeEiwAcoMRHE9OBSx_7TPXm0Hc7Oks0tkuYs8Bd-XTn1u71JLNoYf7y8U9u9y7mRoCuBQQAvD_BwE

[59] *ECMAScript*, ECMA-262, ECMA (European Association for Standardizing Information and Communication Systems) International, Geneva, Switzerland, June. 2022

[60] ECMA International (2023). *Industry association for standardizing information and communication systems* [Online]. Available: https://www.ecma-international.org/

[61] Google Developers, (2023, Jan. 14). *Dependency Injection in Android* [Online]. Available: https://developer.android.com/training/dependency-injection

[62] Google Developers, (2023, Jan. 14). *Dependency Injection with Hilt* [Online]. Available: https://developer.android.com/training/dependency-injection/hilt-android

[63] Google Developers, (2023, Jan. 14). *Build better apps faster with Jetpack Compose* [Online]. Available: https://developer.android.com/jetpack/compose?gclid=CjwKCAiAwomeBhBWEiwAM43YIG6_1uIiA055fjJpXQ5Pl_5Hqws_zXSQFec3Ds1vL-xRLxGRtcfScBoC_YUQAvD_BwE&gclsrc=aw.ds

[64] DevIQ, (2023, Jan 14). *Separation of Concerns* [Online]. Available: https://deviq.com/principles/separation-of-concerns