ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής - Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη Εφαρμογής Υπηρεσιών Κατοικίδιων σε Web και Mobile Περιβάλλον** **Development of Pet Services Application for Web and Mobile**
Ονοματεπώνυμο Φοιτητή	**Γεώργιος Μικέλης**
Πατρώνυμο	**Διονύσιος**
Αριθμός Μητρώου	**ΜΠΣΠ20029**
Επιβλέπων	**Ευθύμιος Αλέπης, Αναπληρωτής Καθηγητής**

ΙΑΝΟΥΑΡΙΟΣ 2023

Τριμελής Εξεταστική Επιτροπή

Ευθύμιος Αλέπης Αναπληρωτής Καθηγητής	Μαρία Βίρβου Καθηγήτρια	Ευάγγελος Σακκόπουλος Αναπληρωτής Καθηγητής

Περιεχόμενα

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή και δάσκαλό μου κύριο Ευθύμιο Αλέπη για τη στήριξή του κατά τη διάρκεια της συγγραφής της διπλωματικής μου εργασίας, τον φίλο μου Κώστα Χρηστάκη που μου χάρισε την αρχική ιδέα για την εφαρμογή όντας και ο ίδιος κάτοχος κατοικίδιου, τους φίλους μου Σπύρο Αυγουστάτο και Παναγιώτη Βαραμέντη για τη συμβολή τους και τις ιδέες τους στον σχεδιασμό της εφαρμογής και τέλος τη γυναίκα μου Φωτεινή Νάκου για την καθημερινή της στήριξη.

Αφιερώσεις

Θα ήθελα να αφιερώσω τη διπλωματική μου εργασία στους γιούς μου Ηλία και Διονυσάκη (που έρχεται).

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μιας web και mobile εφαρμογής για την εύρεση επαγγελματιών που υποστηρίζουν υπηρεσίες φύλαξης, εκπαίδευσης, περιπάτου και γενικής περιποίησης/καλλωπισμού κατοικίδιων.

Η φύση της εφαρμογής είναι διττή καθώς αποτελείται από δύο δίδυμες εφαρμογές για τους κατόχους κατοικίδιων και τους επαγγελματίες του είδους.

Μεταξύ άλλων οι βασικές τεχνολογίες που χρησιμοποιούνται είναι το Ionic/Angular Framework και το .NET Framework. Η χρήση του Ionic Framework σε συνδυασμό με το Capacitor περιβάλλον εκτέλεσης έδωσε τη δυνατότητα για την ανάπτυξη μίας μοναδικής βάσης κώδικα (code base) για την εφαρμογή με την επιλογή διαφορετικών εκδόσεών της όπως web, Android, iOS.

Η κάθε εφαρμογή περιλαμβάνει σελίδα σύνδεσης και αποσύνδεσης, μενού πλοήγησης, περιοχή προφίλ και περιοχή συζητήσεων. Επίσης η εφαρμογή για τους κατόχους περιλαμβάνει επιπροσθέτως περιοχή προφίλ κατοικίδιων και περιοχή μενού προσφερόμενων υπηρεσιών.

Μία από τις βασικές λειτουργίες της εφαρμογής είναι ο εντοπισμός και η επαφή με επαγγελματές στην περιοχή του κατόχου βάσει μιας ακτίνας κάλυψης που μπορεί ο εκάστοτε επαγγελματίας να εξυπηρετήσει.

Λέξεις Κλειδιά: Ionic, Angular, .NET, εφαρμογές διαδικτύου, εφαρμογές κινητών, υβριδικές εφαρμογές, κατοικίδια, εύρεση υπηρεσιών

Abstract

The purpose of this thesis is to develop a web and mobile application for finding professionals who provide services such as pet sitting, training, walking, and general care/grooming of pets.

The nature of the application is dual, as it consists of two separate applications for pet owners and professionals in this field.

Among other technologies, the Ionic/Angular Framework and the .NET Framework are used. The use of the Ionic Framework in combination with the Capacitor runtime gave the ability to develop a single common code base for the application with the option of the deployment and publication of different versions such as web, Android, and iOS.

Each application includes a login and logout page, navigation menu, profile area, and discussion area. Additionally, the application for pet owners includes an additional profile area for their pets and a menu of services offered.

A major advantage of the application is that it allows pet owners to easily find and connect with local professionals.

Key words: Ionic, Angular, .NET, web development, mobile development, hybrid mobile apps, pets, find services

Κεφάλαιο 1: Εισαγωγή

Η παρούσα διπλωματική εργασία πραγματεύεται την ανάπτυξη μίας web και mobile εφαρμογής η οποία βοηθάει ανθρώπους που διαθέτουν κατοικίδια, να εντοπίσουν με εύκολο τρόπο επαγγελματίες που υποστηρίζουν υπηρεσίες φύλαξης, εκπαίδευσης, περιπάτου και γενικής περιποίησης/καλλωπισμού κατοικίδιων στην περιοχή τους. Παράλληλα, η εφαρμογή αυτή αποτελεί και μια πλατφόρμα που δύναται να φιλοξενήσει επαγγελματίες του είδους ώστε να διευκολύνει στην δημιουργία πελατολογίου και δικτύου συνεργασίας.

Η ανάγκη για τέτοιου είδους εφαρμογές, η οποία αναδείχθηκε εντονότερα κατά την περίοδο της πανδημίας του covid-19 εξαιτίας του περιορισμού των μετακινήσεων, παραμένει υψηλή. Κάτι τέτοιο είναι εμφανές στην αγορά, στην οποία καθημερινά εμφανίζονται αντίστοιχου τύπου εφαρμογές παροχής υπηρεσιών.

Η τεχνολογία που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής επιτρέπει την προβολή της τόσο σε περιβάλλον web (ιστοσελίδας) όσο και σε περιβάλλον mobile (android/ios) χωρίς να απαιτείται διαφορετικός κώδικας (κοινό code base), δίνοντας έτσι ταχύτητα και ευκολία κατά την ανάπτυξη, την συντήρηση και την επιπλέον εξέλιξή της, ελαχιστοποιώντας τα λάθη και μεγιστοποιώντας την ομοιογένεια μεταξύ των δύο περιβαλλόντων.

Παράλληλα, ισχυρό πλεονέκτημα αποτελεί η προσέγγιση ‘code first’ στην ανάπτυξη του back-end, η οποία εξασφαλίζει αφενός την προσαρμοστικότητα της εφαρμογής σε διαφορετικές RDBMS τεχνολογίες και database solutions και αφετέρου την εύκολη παραμετροποίηση του κώδικα όσον αφορά τις σχέσεις μεταξύ των κλάσεων.

Συμπερασματικά, η εφαρμογή αυτή θα μπορούσε να αποτελέσει την βάση για ένα production ready προϊόν στην αγορά.

Κεφάλαιο 2: Ανασκόπηση Πεδίου

Τα τελευταία χρόνια έχει αναπτυχθεί πληθώρα από frameworks γύρω από τεχνολογίες που αφορούν στην ανάπτυξη web εφαρμογών, όπως για παράδειγμα front end frameworks (Angular, React.Js, Vue.Js κλπ.), τα οποία διευκολύνουν την οργάνωση και την καθαρότητα του κώδικα. Ακόμα μεγαλύτερη είναι η ανάπτυξη των τεχνολογιών που αφορούν στη δημιουργία mobile εφαρμογών, δεδομένου ότι οι ανάγκες της αγοράς και οι δυνατότητες που προσφέρει το συνεχώς εξελισσόμενο hardware το επιτάσσουν. Πιο συγκεκριμένα, η τάση για mobile first προσέγγιση στην ανάπτυξη των εφαρμογών ανταποκρίνεται στην τάση των χρηστών να χρησιμοποιούν ολοένα και περισσότερο το κινητό τους αντί ενός υπολογιστή.

Αυτή η διαδικασία αντιστοιχεί σε μία διαρκώς αυξανόμενη ανάγκη για την ταχεία δημιουργία εφαρμογών που θα προσφέρουν την κάλυψη διάφορων υπηρεσιών. Αυτή τη στιγμή μία από τις μεγαλύτερες βιομηχανίες παγκοσμίως αφορά την παροχή πληροφοριών και υπηρεσιών μέσω διαδικτυακών εφαρμογών.

Η συγκεκριμένη παγκόσμια τάση επηρεάζει φυσικά και την ελληνική αγορά, στην οποία κυριαρχούν αντίστοιχες εταιρίες υποστήριξης εφαρμογών παροχής υπηρεσιών παράδοσης τροφίμων και έτοιμου φαγητού (για παράδειγμα e-food, wolt, box, instashop), επαγγελματικών υπηρεσιών (για παράδειγμα douleftaras.gr), τραπεζικών υπηρεσιών (για παράδειγμα winbank mobile app) κλπ.

Σε αυτή τη λογική και με αυτή την αφορμή αναπτύχθηκε και η εφαρμογή που παρουσιάζεται στην παρούσα διπλωματική εργασία.

Κεφάλαιο 3: Σκοπός της Εφαρμογής και Σύντομη Περιγραφή

Η εφαρμογή που παρουσιάζεται στην παρούσα διπλωματική εργασία ονομάζεται WalkieDoggie και όπως προαναφέρθηκε αφορά σε υπηρεσίες που σχετίζονται με την φροντίδα των κατοικίδιων. Η ιδέα για την εφαρμογή αυτή προήλθε από την ανάγκη εύρεσης κάποιου ατόμου

που θα παρείχε υπηρεσίες περιπάτου όταν ο ιδιοκτήτης του κατοικίδιου δεν είναι διαθέσιμος. Από αυτήν την ιδέα και από την σκοπιά των αναγκών του χρήστη μιας τέτοιας εφαρμογής προέκυψαν σταδιακά όλες οι απαραίτητες δυνατότητες και λειτουργίες που θα ήταν σκόπιμο να παρέχει. Στην συνέχεια επιλέχθηκαν σταδιακά οι αρμόζουσες τεχνολογίες για την ανάπτυξή της αλλά και η αρχιτεκτονική της με βάσει το business logic.

Στη βάση αυτή πάρθηκε η απόφαση η client-side εφαρμογή να χωριστεί σε δύο εφαρμογές ανάλογα με τον τύπο του χρήστη. Η μία εφαρμογή θα αφορά στους χρήστες με την ιδιότητα του επαγγελματία υπηρεσιών κατοικίδιων -στο εξής θα αναφέρονται ως “Pet Workers”- και η άλλη εφαρμογή θα αφορά στους χρήστες με την ιδιότητα του ιδιοκτήτη κατοικίδιων -στο εξής θα αναφέρονται ως “Pet Owners”. Η πρώτη θα αναφέρεται στο εξής και ως “client-business” εφαρμογή και η δεύτερη ως “client” εφαρμογή.

Παρακάτω ακολουθούν εν συντομία οι τεχνικές προδιαγραφές της client-business εφαρμογής :

- Ο χρήστης θα πρέπει να μπορεί να εγγραφεί στην εφαρμογή ή να συνδεθεί εφόσον είναι ήδη εγγεγραμμένος χρήστης με email και password. Θα μπορεί επίσης και να αποσυνδεθεί.
- Ο χρήστης θα πρέπει να μπορεί να δημιουργήσει το προφίλ του και να μπορεί να το παραμετροποιήσει. Κατά την επεξεργασία του προφίλ του θα έχει την δυνατότητα να δηλώσει κάποια προσωπικά στοιχεία, όπως ονοματεπώνυμο, διεύθυνση κατοικίας, υπηρεσίες -στο εξής θα αναφέρονται ως “pet-services”- που εξυπηρετεί και απόσταση από την κατοικία του που υποστηρίζει τις υπηρεσίες αυτές.
- Ο χρήστης θα πρέπει να μπορεί να δέχεται μηνύματα από επίδοξους πελάτες, να απαντάει σε αυτά και να ανατρέχει σε παρελθοντικές συνομιλίες.

Στην συνέχεια ακολουθούν εν συντομία οι τεχνικές προδιαγραφές της client εφαρμογής:

- Ο χρήστης θα πρέπει να μπορεί να εγγραφεί στην εφαρμογή ή να συνδεθεί εφόσον είναι ήδη εγγεγραμμένος χρήστης με email και password. Θα μπορεί επίσης και να αποσυνδεθεί.
- Ο χρήστης θα πρέπει να μπορεί να δημιουργήσει το προφίλ του και να μπορεί να το παραμετροποιήσει. Κατά την επεξεργασία του προφίλ του θα έχει την δυνατότητα να δηλώσει κάποια προσωπικά στοιχεία, όπως ονοματεπώνυμο και διεύθυνση κατοικίας.
- Ο χρήστης θα πρέπει να μπορεί να καταχωρεί σε μια ξεχωριστή οθόνη το/τα κατοικίδιο/α του μαζί με τα στοιχεία τους, μεταξύ άλλων είδος, ράτσα, γένος, ηλικία, ιατρικές ανάγκες κλπ.
- Ο χρήστης θα πρέπει να μπορεί να διαλέξει την υπηρεσία που αναζητά (περίπατο, φύλαξη, φροντίδα και εκπαίδευση) και στην συνέχεια η εφαρμογή θα του εμφανίζει τους επαγγελματίες που πληρούν τα κριτήρια υποστηριζόμενων υπηρεσιών και απόστασης από την κατοικία τους.
- Ο χρήστης θα πρέπει να μπορεί να στείλει μηνύματα στους επαγγελματίες που η εφαρμογή του εμφάνισε κατά την αναζήτηση αυτή και έπειτα να μπορεί να δέχεται μηνύματα από τους επαγγελματίες αυτούς, καθώς και να μπορεί να ανατρέχει σε παρελθοντικές συνομιλίες μαζί τους.

Υπάρχουν περιθώρια επέκτασης των δυνατοτήτων και λειτουργιών της εφαρμογής που θα αναφερθούν σε επόμενο κεφάλαιο.

Κεφάλαιο 4: Τεχνολογίες και Τεχνικές που Χρησιμοποιήθηκαν

Ο συνδυασμός τεχνολογιών (tech stack) που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής είναι ο εξής:

Front-end/Mobile: Ionic Framework 6 with Angular 14

Back-end/Web API: .NET 6 with Entity Framework

Database: PostgreSQL

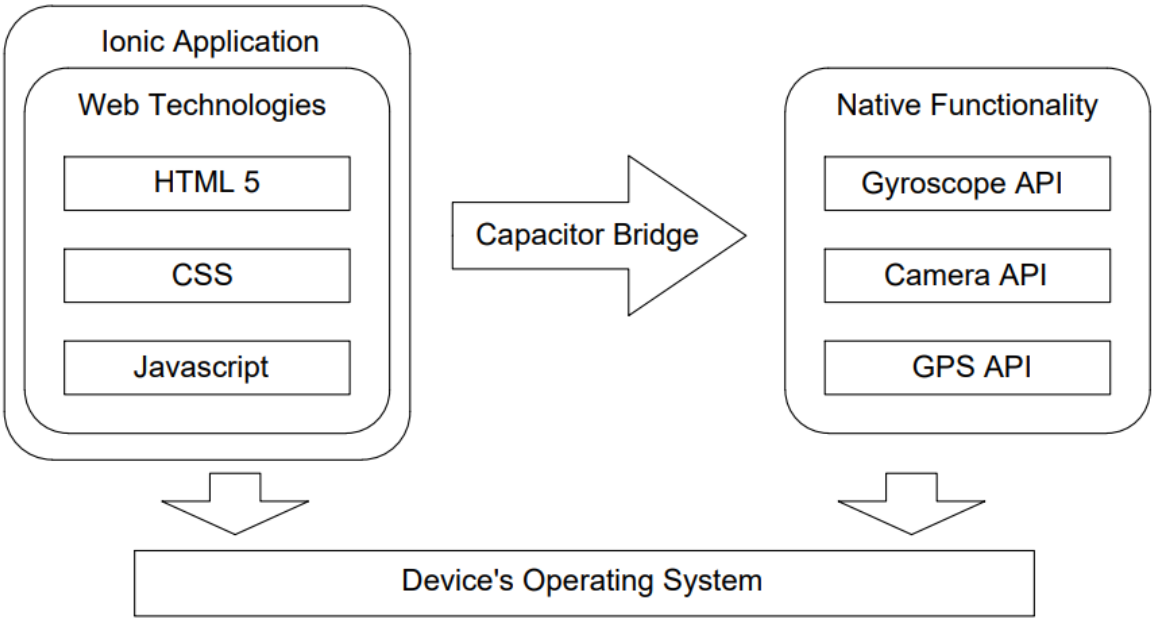
Κεφαλαιο 4.1 Ionic Framework

Η τεχνολογία που επιλέχθηκες για την ανάπτυξη των δύο client-side εφαρμογών είναι το Ionic Framework 6 σε συνδυασμό με την Angular.

Το Ionic Framework είναι ένα ανοιχτού κώδικα πλαίσιο ανάπτυξης (framework) εφαρμογών που βασίζεται σε HTML, CSS και JavaScript για την δημιουργία υβριδικών (hybrid) mobile εφαρμογών. Χρησιμοποιεί κατά κύριο λόγο την Angular, χωρίς να περιορίζεται σε αυτήν, καθώς μπορεί να χρησιμοποιηθεί σε συνδυασμό και με άλλα frameworks όπως React.Js, Vue.Js ή και χωρίς την χρήση κάποιου front-end framework (vanilla JavaScript).

Υβριδικές mobile εφαρμογές είναι εκείνες που έχουν αναπτυχθεί με την χρήση web τεχνολογιών (HTML, CSS και JavaScript) και είναι ενσωματωμένες σε ένα native περίβλημα (native wrapper) όπως το Cordova ή το Capacitor. Τέτοιες εφαρμογές μπορούν να διανεμηθούν κανονικά μέσω των app stores, αφού στην ουσία είναι native εφαρμογές που διαθέτουν απλά ένα webview μέσα στο οποίο τρέχει μια εφαρμογή ιστοσελίδας η οποία μπορεί να χρησιμοποιήσει τις native δυνατότητες της συσκευής (τοποθεσία, γυροσκόπιο, επιταχυνσιόμετρο, κάμερα κτλ) με την βοήθεια JavaScript APIs που προσφέρουν το Cordova ή το Capacitor.

Εικόνα 1: Διάγραμμα Ionic Application



Κάποια από τα πλεονεκτήματα που προσφέρει η τεχνολογία αυτή είναι τα εξής:

- Το Ionic καθιστά εύκολη την ανάπτυξη mobile εφαρμογών στους προγραμματιστές με εξοικείωση στις web τεχνολογίες, χωρίς την ανάγκη εκμάθησης native platform γλωσσών (πχ. Java ή Kotlin για Android, Swift ή Objective-C για iOS)
- Διαθέτει επίσης μια μεγάλη και ενεργή κοινότητα, που συνεπάγεται πως υπάρχει μια πληθώρα πληροφοριών και πηγών στο διαδίκτυο καθώς και ένα εκτενές οικοσύστημα με plugins και βιβλιοθήκες τρίτων μερών.
- Οι Ionic εφαρμογές είναι cross-platform εφαρμογές με μία κοινή βάση κώδικα. Αυτό σημαίνει ότι μπορούν να τρέξουν σε Android, σε iOS και φυσικά να φιλοξενηθούν σε

κάποιον server ως κανονικές ιστοσελίδες. Κάτι τέτοιο μπορεί να ελαχιστοποιήσει κατά πολύ τον χρόνο και τους πόρους κατά την ανάπτυξη, συντήρηση ή επέκταση, συγκριτικά με την ανάπτυξη, συντήρηση και επέκταση ξεχωριστών εφαρμογών για κάθε πλατφόρμα.
- Διαθέτει τέλος μια μεγάλη γκάμα από έτοιμα UI components και εργαλεία που βοηθούν στην γρήγορη ανάπτυξη εφαρμογών υψηλής ποιότητας, τόσο αισθητικά όσο και λειτουργικά.

Παρόλο που η επίδοση μιας hybrid εφαρμογής θα υστερεί πάντα σε σχέση με μια native εφαρμογή, καθώς ο κώδικας δεν μεταφράζεται σε native γλώσσα όπως συμβαίνει σε άλλες τεχνολογίες (πχ react native), τα προτερήματα που αναφέρθηκαν και κυρίως το γεγονός ότι προσφέρεται η δυνατότητα η εφαρμογή να είναι ταυτόχρονα προσβάσιμη από mobile και web περιβάλλοντα, οδήγησε στην επιλογή της συγκεκριμένης τεχνολογίας.

Κεφάλαιο 4.2 Angular Framework

Η Angular είναι ένα front-end JavaScript framework που χρησιμοποιείται ευρέως για την ανάπτυξη single page applications (SPAs). Σε μια SPA web εφαρμογή, η όλη ιστοσελίδα φορτώνεται στον browser και οι μεταγενέστερες επικοινωνίες (requests) με τον διακομιστή χειρίζονται από τον JavaScript κώδικα που τρέχει στον browser, αντί να απαιτείται η επαναφόρτωση πλήρους σελίδας ή άλλων σελιδών από τον διακομιστή. Αυτό οδηγεί σε μια πιο ενσωματωμένη και αποδοτική εμπειρία του χρήστη, καθώς η εφαρμογή δεν χρειάζεται να επικοινωνεί συνεχώς με τον διακομιστή και μπορεί αντίθετα να ενημερώνει δυναμικά την σελίδα με την βοήθεια του JavaScript κώδικα.

Ένα από τα κύρια πλεονεκτήματα της Angular είναι ότι παρέχει μια δομημένη, τμηματοποιημένη αρχιτεκτονική στον κώδικα. Η αρχιτεκτονική αυτή είναι βασισμένη στα components, η οποία επιτρέπει στους προγραμματιστές να χωρίσουν της εφαρμογή τους σε μικρότερα, χρηιμοποιήσιμα κομμάτια που συντηρούνται και παραμετροποιούνται εύκολα. Η Angular επίσης διαθέτει μια πληθώρα από ενσωματωμένα εργαλεία, όπως dependency injection, observables, και μια ισχυρή σύνταξη στην html ώστε να φορτώνεται περιεχόμενο με δυναμικό τρόπο από τον κώδικα.

Η SPA τεχνολογία που χρησιμοποιεί η Angular είναι ιδιαίτερα κατάλληλη για χρήση σε μια εφαρμογή ανεπτυγμένη με το Ionic Framework, καθώς μια ιστοσελίδα που θα έπρεπε να φορτώσει κάθε φορά μια νέα σελίδα από τον διακομιστή θα δυσκόλευε πολύ την ανάπτυξη της εφαρμογής.

Εκτός από τη δομή και τα ενσωματωμένα χαρακτηριστικά της, η Angular έχει μια ισχυρή κοινότητα και μεγάλο οικοσύστημα τρίτων βιβλιοθηκών και εργαλείων. Αυτό κάνει ευκολότερο την εύρεση λύσεων για κοινά προβλήματα και την ενσωμάτωση με άλλες τεχνολογίες.

Εν κατακλείδι, η Angular είναι μια δημοφιλής επιλογή για την ανάπτυξη SPAs, ιδιαίτερα όταν υπάρχει ανάγκη για μια δομημένη, κλιμακούμενη και εμπλουτισμένη λύση. Είναι μια καλή εναλλακτική λύση σε σύγκριση με τη χρήση vanilla JavaScript ή άλλων frameworks, καθώς παρέχει μια πιο πλήρη και γνωμική προσέγγιση στην ανάπτυξη ιστοσελίδων.

Κεφάλαιο 4.3 .NET 6 και Entity Framework

Το .NET 6 ήταν η τελευταία έκδοση του .NET Framework κατά την έναρξη της ανάπτυξης της παρούσας εφαρμογής και είναι σχεδιασμένο ώστε να είναι ένα γενικού σκοπού, cross-platform framework για την ανάπτυξη ενός ευρύ πεδίου εφαρμογών, όπως web, desktop, mobile, gaming, IoT και AI εφαρμογές. Ένα από τα κύρια χαρακτηριστικά του .NET 6 είναι η δυνατότητα δημιουργίας web APIs, τα οποία χρησιμοποιούνται για την κατασκευή RESTful (representational state transfer) υπηρεσιών οι οποίες είναι προσβάσιμες μέσω του http/https πρωτοκόλλου (internet protocol) από μια σειρά από εφαρμογές “πελάτες” (client aps) όπως web browsers, mobile συσκευές ή άλλους servers.

Κάποια από τα πλεονεκτήματα του .NET 6 είναι τα εξής:

- Cross-platform συμβατότητα: Το .NET 6 είναι σχεδιασμένο να τρέχει σε μια πληθώρα από πλατφόρμες/λογισμικά όπως Windows, Linux, macOS.
- Υψηλές αποδόσεις: Το .NET 6 είναι χτισμένο πάνω στο .NET runtime περιβάλλον το οποίο είναι βελτιστοποιημένο για γρήγορες ταχύτητες και επεκτασιμότητα, ώστε να μπορούν να αναπτυχθούν γρήγορα και αξιόπιστα web APIs τα οποία θα μπορούν να επεξεργαστούν μεγάλο αριθμό από requests.
- Διαθέτει μια μεγάλη και ενεργή κοινότητα, που συνεπάγεται πως υπάρχει μια πληθώρα πληροφοριών, πηγών και υψηλής ποιότητας documentation στο διαδίκτυο.
- Τέλος ένα από τα σημαντικότερα πλεονεκτήματα είναι ότι έχει ενσωματωμένο το Entity Framework για το οποίο γίνεται λόγος παρακάτω.

Το Entity Framework (EF) είναι ένα σύστημα «Αντιστοίχισης Αντικειμένων» ή αλλιώς Object-Relational-Mapper (ORM) το οποίο είναι ενσωματωμένο στο .NET και είναι σχεδιασμένο ώστε να απλοποιήσει τις διαδικασίες που σχετίζονται με την βάση δεδομένων αλλά και την επικοινωνία αυτής με το web API. Πιο συγκεκριμένα, το EF δίνει την δυνατότητα στον προγραμματιστή να σχεδιάσει το μοντέλο της βάσης (database schema) χρησιμοποιώντας τις σχέσεις των .NET κλάσεων (όπου σε αυτό το πλαίσιο καλούνται και entities) που έχει δημιουργήσει. Η προσέγγιση αυτή, κατά την οποία σχεδιάζονται πρώτα οι κλάσεις και βάση των σχέσεων των κλάσεων αυτών δημιουργείται όλο το μοντέλο της βάσης, ονομάζεται code-first προσέγγιση. Το EF επίσης αναλαμβάνει την μετάφραση του .NET/C# κώδικα σε SQL εντολές που εκτελούνται πάνω στην βάση. Οι δυνατότητες αυτές προσδίδουν τα εξής πλεονεκτήματα:

- Απλοποιημένη πρόσβαση στα αποθηκευμένα δεδομένα: Με το EF υπάρχει η δυνατότητα επικοινωνίας με την βάση χρησιμοποιώντας έναν οικείο στους περισσότερους προγραμματιστές «αντικειμενοστραφή» τρόπο, χωρίς την ανάγκη σύνταξης «ωμών» SQL queries. Αυτό κάνει την σύνταξη και την συντήρηση του κώδικα πρόσβασης δεδομένων (data access code) ευκολότερη.
- Ευελιξία: Το EF υποστηρίζει ένα σύνολο από database λύσεις, όπως MsSQL, PostgreSQL, OracleDb, MySQL και άλλων. Αυτό σημαίνει ότι μπορεί να γίνει αλλαγή της database τεχνολογίας οποιαδήποτε στιγμή χωρίς να χρειαστεί να συνταχθεί από την αρχή ο data access κώδικας, αφού το EF θα μεταφράσει τον υπάρχον .NET/C# κώδικα στις κατάλληλες κάθε φορά SQL εντολές ανάλογα με την επιλεγμένη βάση δεδομένων.

Όλοι οι παραπάνω λόγοι οδήγησαν στην επιλογή της συγκεκριμένης τεχνολογίας για την ανάπτυξη του web API μέρους της εφαρμογής.

Κεφάλαιο 4.4 PostgreSQL

Η PostgreSQL είναι ένα ανοιχτού κώδικα Relational Database Management System (RDBMS) και είναι δημοφιλές για την αξιοπιστία του και τις επιδόσεις του. Για τον λόγο αυτό επιλέχθηκε για να καλύψει τις ανάγκες της βάσης δεδομένων της εφαρμογής.

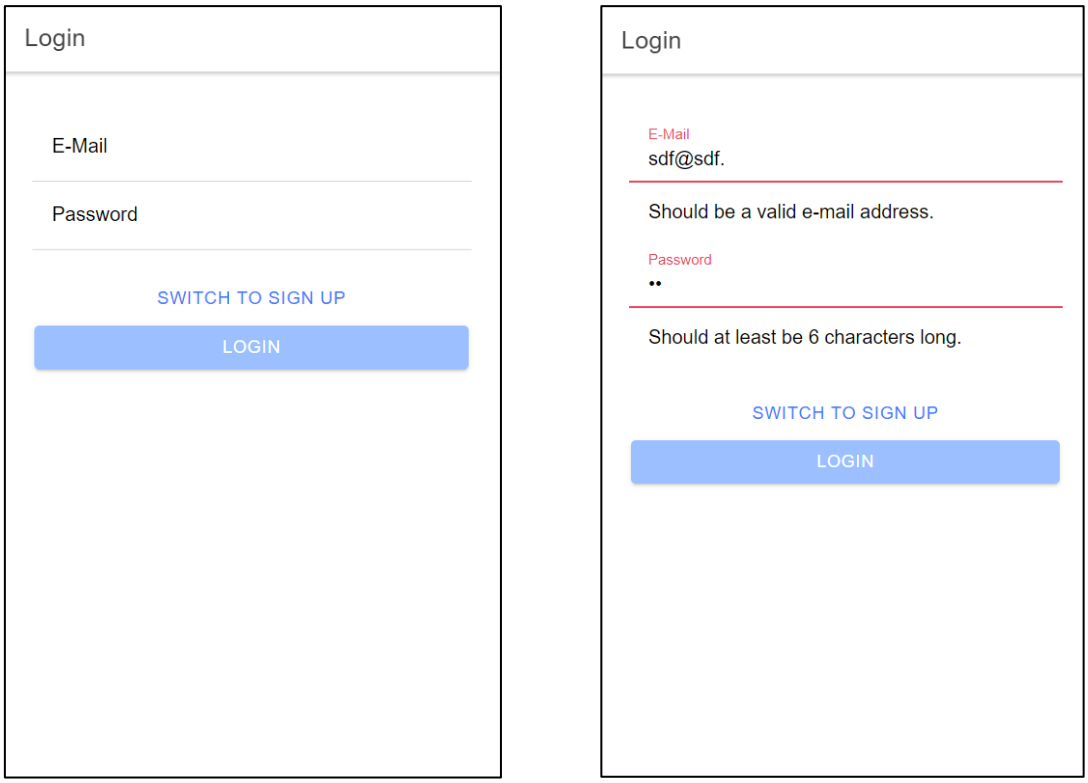
Κεφάλαιο 5: Παρουσίαση και Χρήση της Εφαρμογής

Στα επόμενα υποκεφάλαια θα περιγραφούν αναλυτικά οι λειτουργίες της εφαρμογής με τη βοήθεια εικόνων και επεξηγήσεων όπου χρειάζεται. Για κάθε σελίδα θα σημειώνεται το αν αφορά την εφαρμογή PetWorker, την εφαρμογή PetOwner ή και τις δύο.

Κεφάλαιο 5.1: Σελίδα Σύνδεσης

Για να συνδεθεί ένας χρήστης στην εφαρμογή χρειάζεται να εισάγει τα στοιχεία του. Αυτό πραγματοποιείται στην σελίδα σύνδεσης, η οποία είναι κοινή και στις δύο εφαρμογές. Παρακάτω ακολουθεί μία εικόνα της σελίδας αυτής.

Εικόνα 2: Σελίδα Σύνδεσης και σελίδα σύνδεσης μετά από μη έγκυρη εισαγωγή στοιχείων

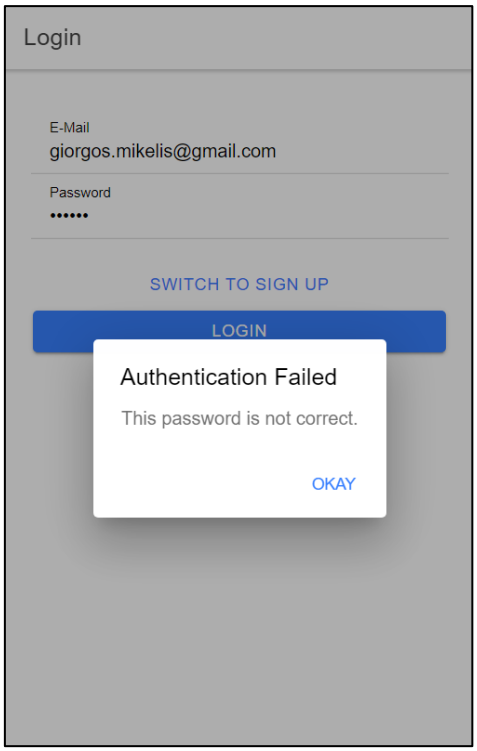


Όπως φαίνεται παραπάνω, η σελίδα περιλαμβάνει τον τίτλο login που κατατοπίζει τον χρήστη, το πεδίο E-Mail και το πεδίο Password για την εισαγωγή των αντίστοιχων στοιχείων και τέλος την επιλογή για απευθείας σύνδεση, εφόσον ο χρήστης έχει εγγραφεί ήδη στην εφαρμογή ή εναλλακτικά την επιλογή για την πρώτη εγγραφή.

Τα πεδία E-Mail και Password περιλαμβάνουν περιορισμούς σε σχέση με την εισαγωγή εύλογων στοιχείων (front-end data validation). Για παράδειγμα στο πεδίο E-Mail το περιεχόμενο πρέπει να έχει τη μορφή μιας ηλεκτρονικής διεύθυνσης (XXX@XXX.XXX) ειδάλλως το κουμπί login είναι ανενεργό. Αντίστοιχα, ένας κωδικός πρέπει να περιλαμβάνει τουλάχιστον 6 χαρακτήρες ώστε να ενεργοποιηθεί το κουμπί login. Όταν ισχύουν και οι δύο παραπάνω προϋποθέσεις τότε το login ενεργοποιείται. Αν οι προϋποθέσεις δεν πληρούνται, η πληροφορία αυτή γνωστοποιείται και στον χρήστη με σχετικά μηνύματα που εμφανίζονται κάτω από το εκάστοτε πεδίο.

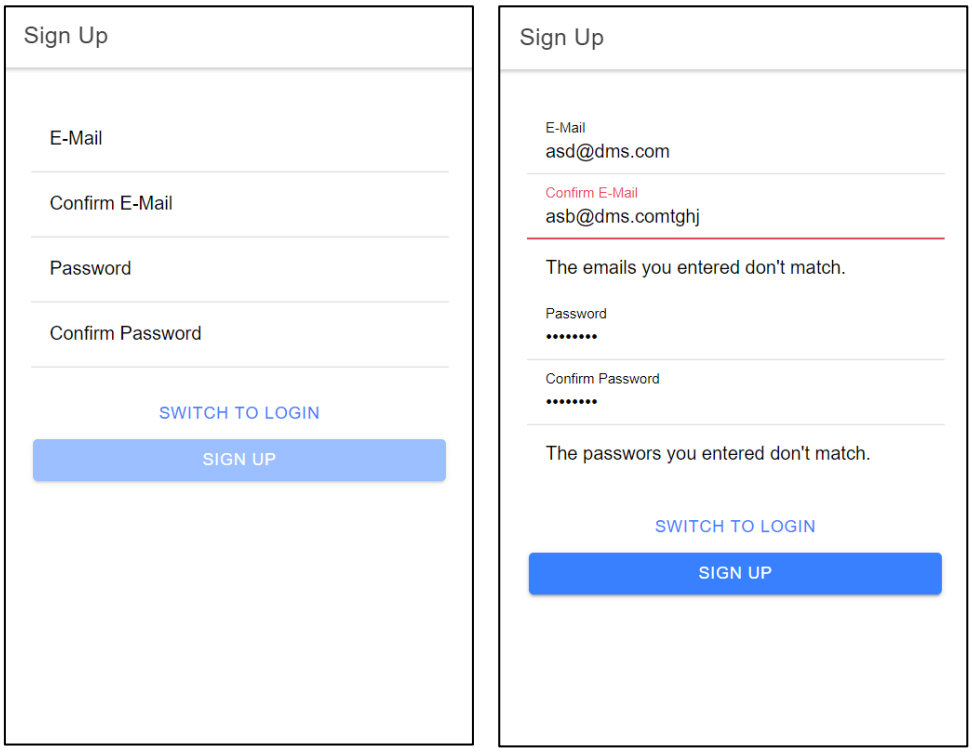
Στην περίπτωση που εισαχθούν αποδεκτά στοιχεία και πατηθεί το κουμπί login, στη συνέχεια θα πραγματοποιηθεί αυθεντικοποίηση του χρήστη με τα στοιχεία αυτά στον server. Εφόσον δεν υπάρχει χρήστης με το συγκεκριμένο e-mail στη βάση ή εφόσον υπάρχει χρήστης με το e-mail αυτό αλλά έχει εισαχθεί λανθασμένος κωδικός, θα εμφανιστεί μήνυμα λάθους το οποία θα τον πληροφορεί περί τίνος πρόκειται. Αν τα στοιχεία που εισήχθησαν υπάρχουν στη βάση, τότε επιτρέπεται η πρόσβαση στην εφαρμογή και ο χρήστης κατευθύνεται στην αρχική σελίδα της εκάστοτε εφαρμογής.

Εικόνα 3: Οθόνη με μήνυμα σφάλματος μετά την επικοινωνία με τον server



Εφόσον ο χρήστης επιλέξει το κουμπί Switch to Sign Up μεταφέρεται στη σελίδα εγγραφής, η οποία φαίνεται παρακάτω:

Εικόνα 4: Σελίδα Εγγραφής και σελίδα εγγραφής μετά από μη έγκυρη εισαγωγή στοιχείων



Η λειτουργία σε αυτή την περίπτωση είναι παρόμοια με αυτή του login, όσον αφορά τους περιορισμούς ορθότητας των εισαγόμενων στοιχείων, με την προσθήκη ενός ακόμη περιορισμού σχετικά με την ταύτιση του πεδίου E-Mail με το Confirm E-Mail και αντιστοίχως του πεδίου Password με το πεδίο Confirm Password.

Εφόσον εισαχθούν έγκυρα στοιχεία, τότε ελέγχεται η περίπτωση να υπάρχει ήδη κάποιος χρήστης με αυτό το e-mail στη βάση δεδομένων. Στην περίπτωση που κάτι τέτοιο ισχύει εμφανίζεται το σχετικό μήνυμα λάθους. Διαφορετικά ο χρήστης εγγράφεται επιτυχώς στη βάση και ταυτόχρονα συνδέεται στην εφαρμογή και μεταφέρεται στην Αρχική Σελίδα της εκάστοτε εφαρμογής, όπως και στην περίπτωση του login.

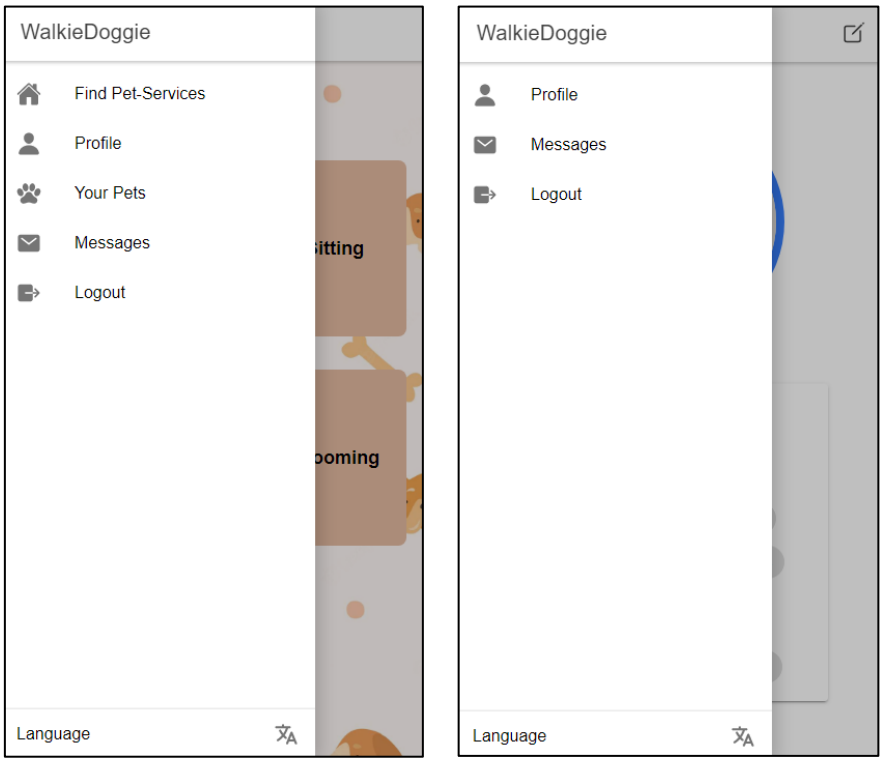
Επιπροσθέτως, στη σελίδα εγγραφής υπάρχει η δυνατότητα μεταφοράς στη σελίδα σύνδεσης.

Τέλος, οι σελίδες σύνδεσης και εγγραφής είναι οι μοναδικές που δεν δίνουν πρόσβαση στο μενού. Όλες οι υπόλοιπες βασικές σελίδες κορμού δίνουν πρόσβαση στο μενού με τη βοήθεια του κουμπιού hamburger menu.

Κεφάλαιο 5.2: Μενού

Οι δύο εφαρμογές PetOwner και PetWorker περιλαμβάνουν από ένα side μενού το οποίο εμφανίζεται είτε πατώντας στο κουμπί hamburger menu, είτε σέρνοντας το δάχτυλο ή το ποντίκι από τα αριστερά της οθόνης προς τα δεξιά. Οι βασικές εικόνες των μενού φαίνονται παρακάτω.

Εικόνα 5: Side Menu της εφαρμογής PetOwner (αριστερά) και Side Menu της εφαρμογής PetWorker (δεξιά)



Και τα δύο μενού περιλαμβάνουν τον τίτλο της εφαρμογής στο header τους, τη δυνατότητα αλλαγής της γλώσσας των λεκτικών της εφαρμογής στο footer τους και ενδιάμεσα όλες τις επιλογές περιήγησης που προσφέρουν. Ακόμη, μετά τις επιλογές αυτές υπάρχει η επιλογή logout η οποία οδηγεί τον χρήστη στη σελίδα εγγραφής/σύνδεσης αποσυνδέοντάς τον.

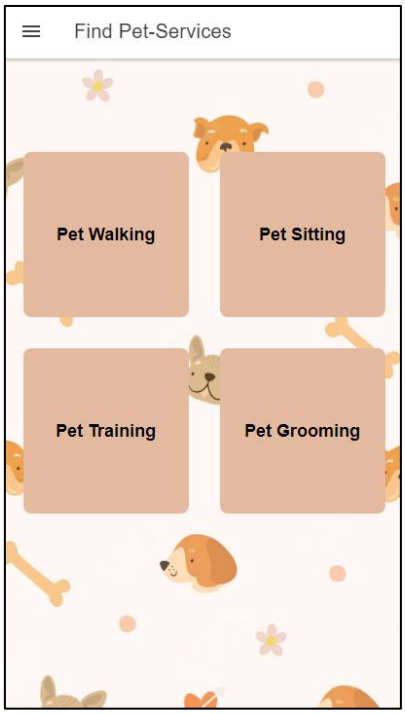
Κεφάλαιο 5.3: Περιοχή Find Pet-Services

Η περιοχή αυτή προσφέρεται μόνο στην εφαρμογή του PetOwner, καθώς περιλαμβάνει τις διαθέσιμες υπηρεσίες που μπορεί ένας ιδιοκτήτης κατοικίδιου να επιλέξει και όλα τα σχετιζόμενα με αυτές. Σημειώνεται επίσης ότι η αρχική σελίδα της περιοχής αυτής είναι η Αρχική Σελίδα (landing page) της εφαρμογής PetOwner στην οποία οδηγείται ο χρήστης μετά την επιτυχή σύνδεσή του. Η σύνδεση αυτή παραμένει ενεργή ώσπου ο χρήστης να αποσυνδεθεί χειροκίνητα (να πατήσει logout) ή να λήξει η χρονική διάρκεια του token (JWR) του, ακόμα κι αν επιλέξει να κλείσει την εφαρμογή.

Κεφάλαιο 5.3.1: Αρχική Σελίδα Find Pet-Services

Η αρχική σελίδα Find Pet-Services περιλαμβάνει τον τίτλο της περιοχής, όπως επίσης και τέσσερα κουμπιά σε παράταξη (tiles) τα οποία αναγράφουν την προσφερόμενη υπηρεσία, όπως φαίνεται και στην παρακάτω εικόνα.

Εικόνα 6: Αρχική σελίδα περιοχής Find Pet-Services

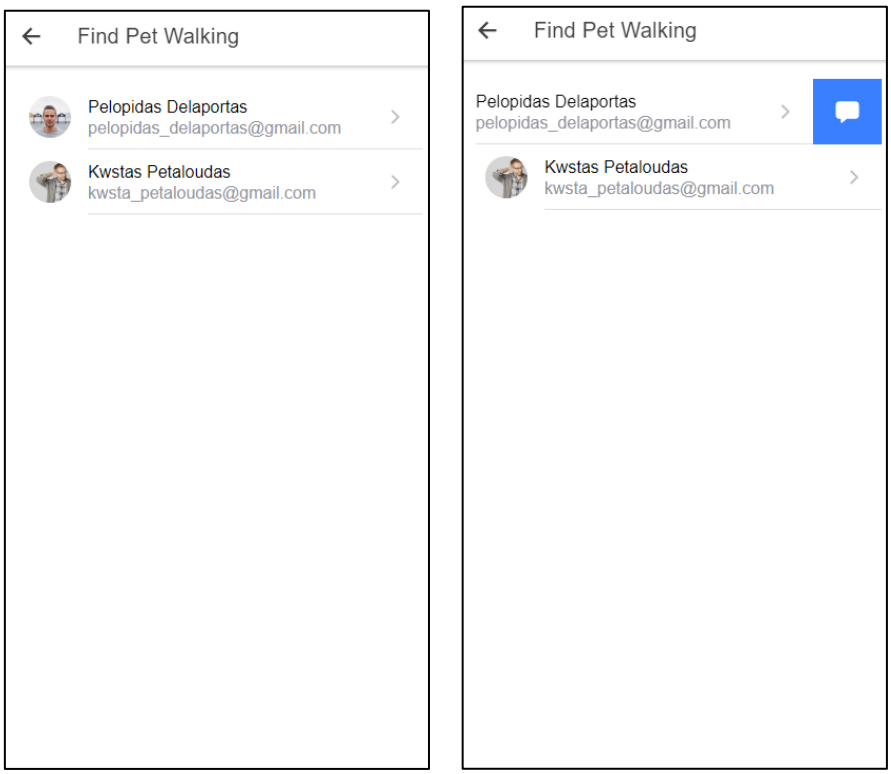


Το κάθε κουμπί υπηρεσίας ανακατευθύνει τον χρήστη (με forward navigation γραφικό) στην αντίστοιχη σελίδα της σχετικής υπηρεσίας.

Κεφάλαιο 5.3.2: Σελίδα Αναζήτησης Pet-Service

Μόλις ο χρήστης επιλέξει κάποιο tile της αρχικής σελίδας Find Pet-Services θα οδηγηθεί στην σχετική σελίδα Pet Service. Η σελίδα αυτή περιλαμβάνει έναν δυναμικό τίτλο με τη μορφή Find …. (εκάστοτε υπηρεσία που επιλέχθηκε), όπως επίσης και μία λίστα με τους διαθέσιμους επαγγελματίες της υπηρεσίας που δύνανται να προσφέρουν τις υπηρεσίες αυτές στην τοποθεσία που έχει δηλώσει ο χρήστης ως κατοικία. Ακόμα, το βέλος πλησίον του τίτλου προσφέρει τη δυνατότητα backward navigation προς την αρχική σελίδα Find Pet-Services.

Εικόνα 7: Σελίδα Υπηρεσίας Pet Walking (αριστερά) και Σελίδα Υπηρεσίας Pet Walking όπου έχει ανοίξει η επιλογή για συνομιλία με κάποιον επαγγελματία (δεξιά)



Η λίστα που προαναφέρθηκε αποτελείται από UI αντικείμενα, καθένα εκ των οποίων περιλαμβάνει δύο λειτουργίες.

Η πρώτη λειτουργία είναι αυτή της επιλεξιμότητας, δηλαδή ο χρήστης μπορεί να πατήσει το αντικείμενο ως απλό κουμπί. Αυτή θα οδηγήσει τον χρήστη στη σελίδα προφίλ του επαγγελματία, η οποία θα περιγραφεί σε επόμενο κεφάλαιο. Η σελίδα αυτή είναι ίδια με την σελίδα προφίλ του επαγγελματία, με τη διαφορά ότι δεν εμφανίζεται το κουμπί επεξεργασίας της. Η ανακατεύθυνση αυτή θα πραγματοποιηθεί με forward navigation (με τα αντίστοιχα γραφικά) και επομένως θα προσφέρεται η δυνατότητα πλοήγησης προς τα πίσω στην ίδια σελίδα.

Με τη δεύτερη λειτουργία ο χρήστης μπορεί να σύρει το αντικείμενο προς τα αριστερά, ώστε να εμφανιστεί η επιλογή για συνομιλία με τον συγκεκριμένο επαγγελματία. Αν ο χρήστης πατήσει στο μπλε κουμπί, τότε θα ανακατευθυνθεί προς την περιοχή μηνυμάτων και συγκεκριμένα στη σελίδα συνομιλίας με τον συγκεκριμένο επαγγελματία. Η περιοχή αυτή θα περιγραφεί σε επόμενο κεφάλαιο. Τέλος, υπάρχει και εδώ η λειτουργία του forward navigation.

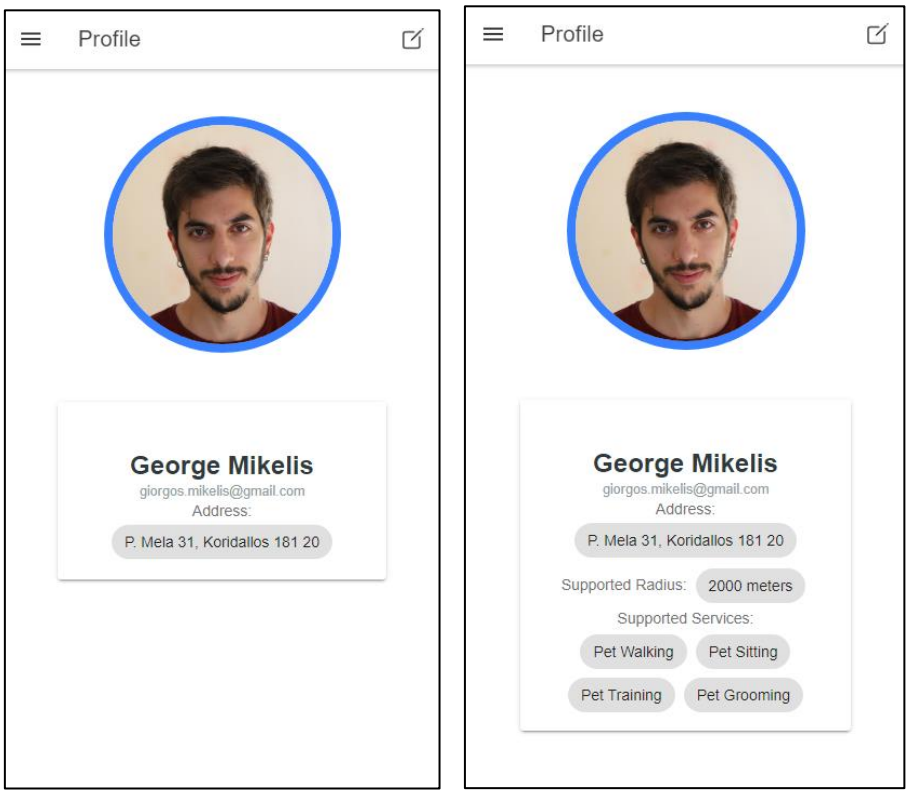
Κεφάλαιο 5.4: Περιοχή Προφίλ

Στην περιοχή αυτή ο χρήστης μπορεί να εισάγει τα προσωπικά του στοιχεία (φωτογραφία, διεύθυνση κλπ.). Έχει επίσης τη δυνατότητα επεξεργασίας των στοιχείων αυτών. Σημειώνεται πως η πληροφορία της διεύθυνσης είναι απαραίτητη ώστε να εμφανίζονται οι επαγγελματίες μιας υπηρεσίας που μπορούν να εξυπηρετήσουν την εκάστοτε περιοχή. Οι σελίδες προφίλ για τους PetOwners και PetWorkers διαφέρουν μεταξύ τους και θα σχολιαστούν ξεχωριστά.

Κεφάλαιο 5.4.1: Σελίδα Προφίλ

Ο εκάστοτε χρήστης PetOwner ή PetWorker οδηγείται στη σελίδα προφίλ από το side menu, όπως έχει προαναφερθεί. Οι δύο σελίδες έχουν αρκετές ομοιότητες μεταξύ τους και φαίνονται παρακάτω.

Εικόνα 8: Σελίδα προφίλ PetOwner (αριστερά) και Σελίδα προφίλ PetWorker (δεξιά)



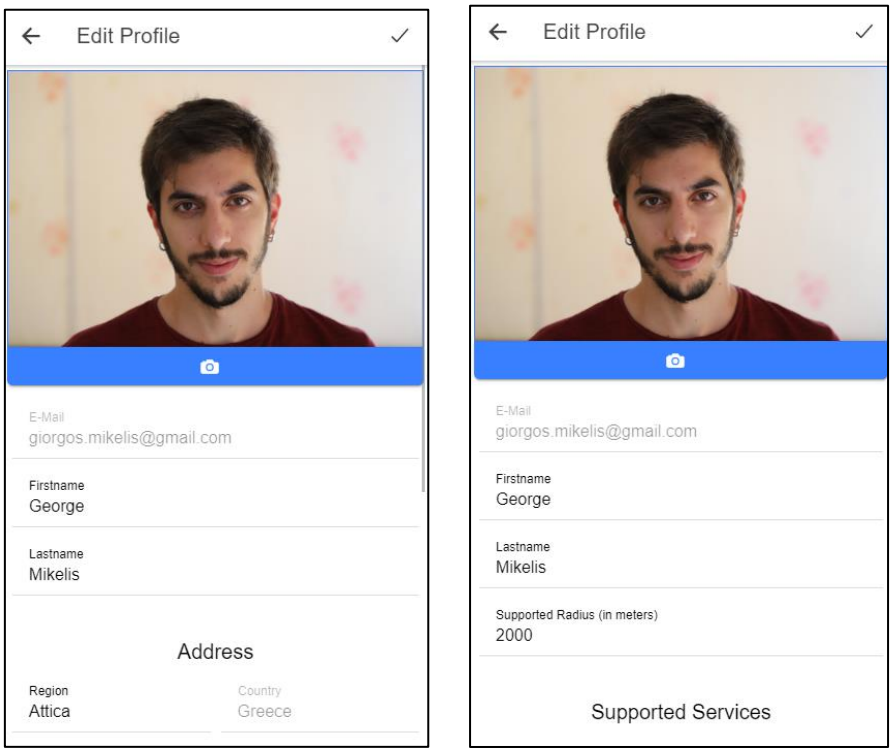
Και οι δύο σελίδες περιλαμβάνουν τον τίτλο Profile, την επιλογή επεξεργασίας του προφίλ πλησίον του τίτλου, τη φωτογραφία που έχει ανεβάσει ο χρήστης (αν δεν έχει ανεβάσει φωτογραφία φαίνεται μία default εικόνα), το ονοματεπώνυμο του χρήστη, τη διεύθυνση και το e-mail του.

Στη σελίδα του PetWorker εμφανίζονται επιπροσθέτως τα στοιχεία της υποστηριζόμενης ακτίνας υπηρεσιών σε μέτρα και οι υποστηριζόμενες εργασίες.

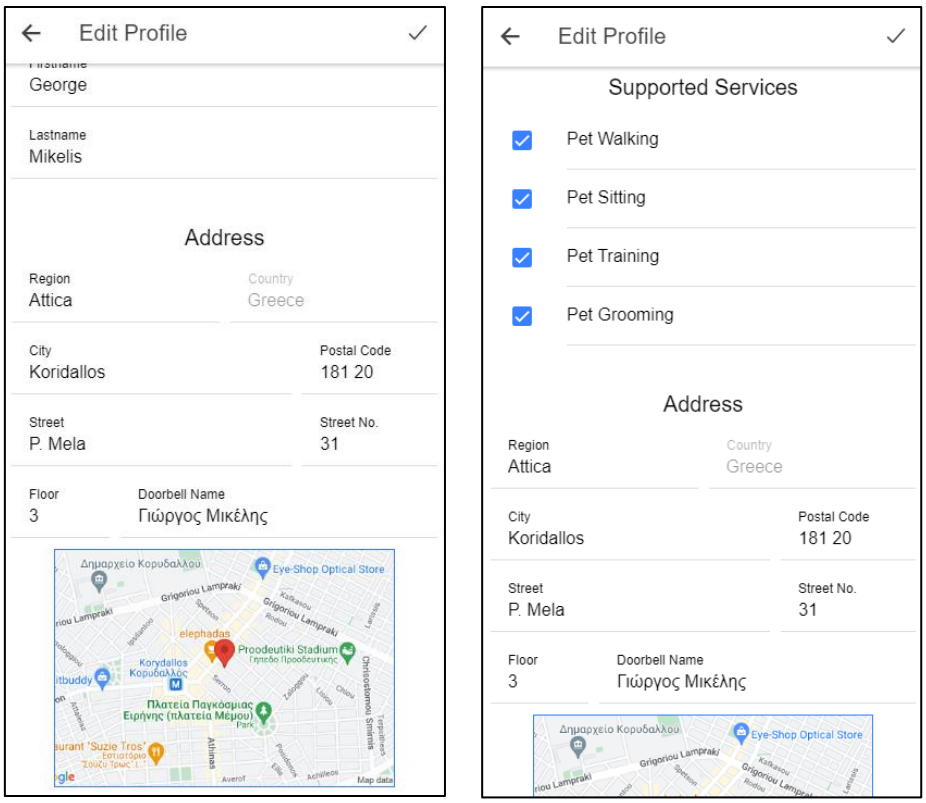
Κεφάλαιο 5.4.2: Σελίδα Επεξεργασίας Προφίλ

Όπως προαναφέρθηκε, ο χρήστης οδηγείται στις σελίδες επεξεργασίας προφίλ από το αντίστοιχο σύμβολο στη σελίδα προφίλ. Οι σελίδες αυτές περιλαμβάνουν backward navigation, το σύμβολο του οποίου παριστάνεται πλησίον του τίτλου. Και σε αυτή την περίπτωση οι σελίδες επεξεργασίας για τον PetOwner και τον PetWorker εμφανίζουν αρκετές ομοιότητες. Δύο παραδείγματα απεικονίζονται παρακάτω.

Εικόνα 9: Σελίδα Επεξεργασίας προφίλ PetOwner (αριστερά) και Σελίδα Επεξεργασίας προφίλ PetWorker (δεξιά) μέρος Α'



Εικόνα 10: Σελίδα Επεξεργασίας προφίλ PetOwner (αριστερά) και Σελίδα Επεξεργασίας προφίλ PetWorker (δεξιά) μέρος Β'



Και στις δύο σελίδες φαίνεται ο τίτλος Edit Profile, ένα σύμβολο (√) για την αποθήκευση των στοιχείων και η φωτογραφία με την επιλογή της εύρεσης και αποθήκευσης. Αν ο χρήστης επιλέξει να αποθηκεύσει ή να αλλάξει τη φωτογραφία προφίλ, η εφαρμογή ενεργοποιεί την κάμερα (είτε του κινητού είτε του υπολογιστή) και δίνει την επιλογή λήψης νέας φωτογραφίας ή εναλλακτικά τη λειτουργία file upload με τη δυνατότητα πρόσβασης στο photo gallery (αν πρόκειται για κινητό) ή στο file explorer (αν πρόκειται για υπολογιστή).

Εικόνα 11: Οθόνη με ανοιχτή κάμερα

Ακόμη, και οι δύο εφαρμογές εμφανίζουν το e-mail του χρήστη χωρίς τη δυνατότητα αλλαγής του, καθώς αποτελεί στοιχείο ταυτοποίησης του χρήστη. Στη συνέχεια, υπάρχει δυνατότητα συμπλήρωσης ονόματος και επωνύμου.

Στην εφαρμογή του PetWorker υπάρχουν σε αυτό το σημείο κάποια επιπρόσθετα πεδία που αφορούν την ακτίνα που μπορεί να προσφέρει τις υπηρεσίες του ο επαγγελματίας, όπως επίσης και τη λίστα με τις υπηρεσίες που εξυπηρετεί. Η ακτίνα δηλώνεται σε μέτρα και έχει ως κέντρο τη διεύθυνση που θα δηλώσει παρακάτω ο χρήστης. Οι υπηρεσίες που προσφέρει δηλώνονται με τη μορφή tick boxes και για να ενεργοποιηθεί το κουμπί αποθήκευσης, ο χρήστης θα πρέπει να έχει επιλέξει τουλάχιστον μία υπηρεσία.

Στη συνέχεια, οι δύο εφαρμογές έχουν πεδία για την χειροκίνητη εισαγωγή της διεύθυνσης. Εναλλακτικά, ο χρήστης μπορεί να επιλέξει τη διεύθυνσή του με τη βοήθεια του χάρτη, πατώντας επάνω του, ενέργεια η οποία θα του εμφανίσει επιλογές για αυτόματη εύρεση τοποθεσίας της συσκευής (είτε στο mobile είτε στο web, εφόσον έχουν δοθεί τα απαραίτητα δικαιώματα για αυτό) και για εύρεση της διεύθυνσης στον χάρτη (γίνεται άνοιγμα των google maps σε νέα σελίδα).

Εικόνα 12: Σελίδα Google Maps με λειτουργία επιλογής σημείου

Στην περίπτωση που ο χρήστης επιλέξει τη διεύθυνσή του μέσω του χάρτη ή της αυτόματης εύρεσης τοποθεσίας, εξακολουθεί να είναι απαραίτητη η χειροκίνητη εισαγωγή των στοιχείων ορόφου και ονόματος στο κουδούνι.

Τέλος, όταν ο χρήστης επιλέξει το κουμπί της αποθήκευσης, τα στοιχεία αποθηκεύονται στη βάση κι εκείνος οδηγείται στη σελίδα προφίλ με τα επικαιροποιημένα στοιχεία.

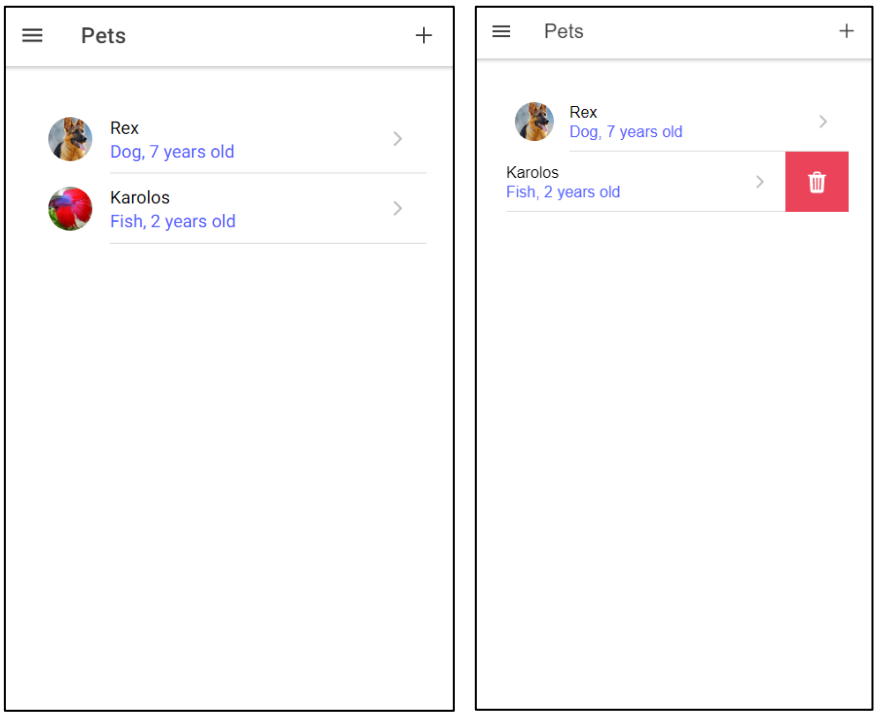
Κεφάλαιο 5.5: Περιοχή Κατοικίδιων

Η περιοχή κατοικίδιων είναι διαθέσιμη μόνο στην εφαρμογή PetOwner καθώς σε αυτοί οι χρήστες έχουν τη δυνατότητα να εισάγουν στοιχεία για τα κατοικίδιά τους, να τα επεξεργαστούν, να τα σβήσουν, καθώς επίσης και να ανασκοπήσουν τη λίστα με όλες τις εισαγωγές.

Κεφάλαιο 5.5.1: Σελίδα Λίστας Κατοικίδιων

Ο χρήστης οδηγείται στη Σελίδα Λίστας Κατοικίδιων από το side menu, με την επιλογή Your Pets. Η σελίδα αυτή περιλαμβάνει το σύμβολο του hamburger menu, τον τίτλο Pets και πλησίον του το σύμβολο (+) για την προσθήκη νέου κατοικίδιου. Ακολουθεί στη συνέχεια λίστα με όλα τα κατοικίδια που έχει εισάγει ο χρήστης. Για κάθε κατοικίδιο φαίνεται ενδεικτικά η φωτογραφία που έχει εισαχθεί, το όνομά του, το είδος και η ηλικία του.

Εικόνα 13: Σελίδα Λίστας Κατοικίδιων (αριστερά) και Σελίδα Λίστας Κατοικίδιων όπου έχει ανοίξει η επιλογή για διαγραφή (δεξιά)



Η λίστα που προαναφέρθηκε αποτελείται από UI αντικείμενα, καθένα εκ των οποίων περιλαμβάνει δύο λειτουργίες.

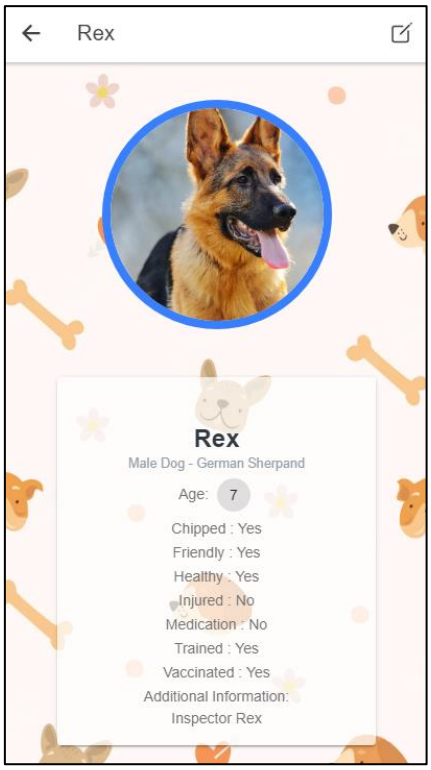
Η πρώτη λειτουργία είναι αυτή της επιλεξιμότητας, δηλαδή ο χρήστης μπορεί να πατήσει το αντικείμενο ως απλό κουμπί. Αυτή θα οδηγήσει τον χρήστη στη σελίδα προφίλ του κατοικίδιου, η οποία θα περιγραφεί σε επόμενο υποκεφάλαιο. Η ανακατεύθυνση αυτή θα πραγματοποιηθεί με forward navigation (με τα αντίστοιχα γραφικά) και επομένως θα προσφέρεται η δυνατότητα πλοήγησης προς τα πίσω στην ίδια σελίδα.

Με τη δεύτερη λειτουργία ο χρήστης μπορεί να σύρει το αντικείμενο προς τα αριστερά, ώστε να εμφανιστεί η επιλογή διαγραφής του αντικειμένου. Αν ο χρήστης πατήσει στο κόκκινο κουμπί, τότε το αντικείμενο θα διαγραφεί.

Κεφάλαιο 5.5.2: Σελίδα Προφίλ Κατοικίδιου

Στη σελίδα προφίλ κατοικίδιου ο χρήστης μπορεί να ανασκοπήσει τα χαρακτηριστικά που αφορούν το κατοικίδιο που έχει επιλέξει. Η σελίδα περιλαμβάνει το βέλος πλοήγησης, έναν δυναμικό τίτλο με το όνομα του κατοικίδιου, το κουμπί επεξεργασίας των στοιχείων, τη φωτογραφία του κατοικίδιου και τα στοιχεία του, όπως φαίνεται παρακάτω. Αυτή η σελίδα σκοπό έχει την προβολή των χαρακτηριστικών του κατοικίδιου.

Εικόνα 14: Σελίδα Προφίλ Κατοικίδιου

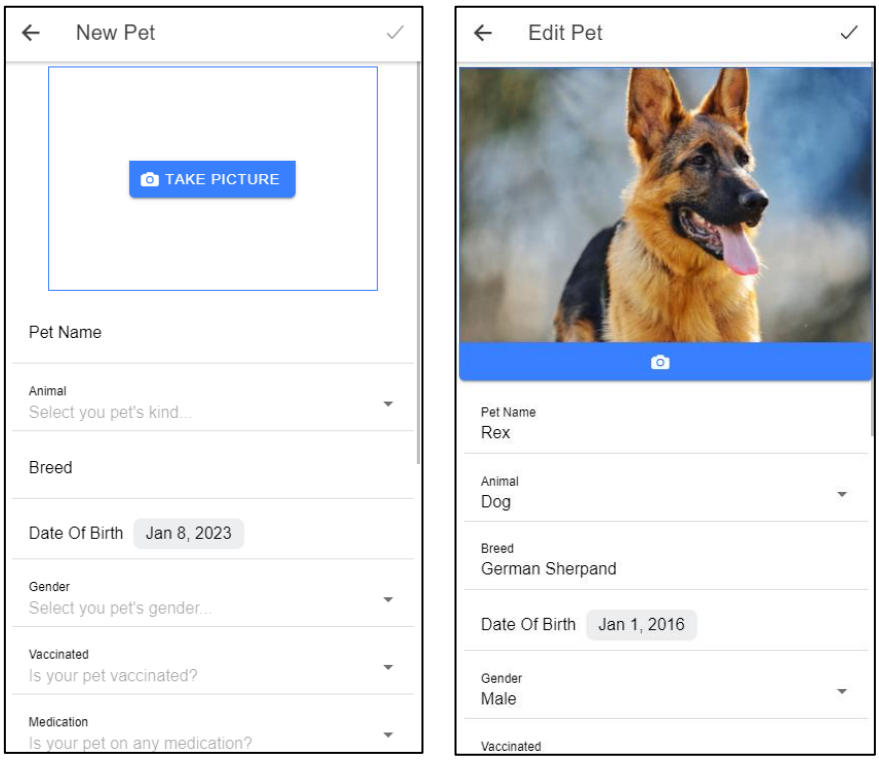


Κεφάλαιο 5.5.3: Σελίδα Επεξεργασίας Κατοικίδιου και Σελίδα Εισαγωγής Νέου Κατοικίδιου

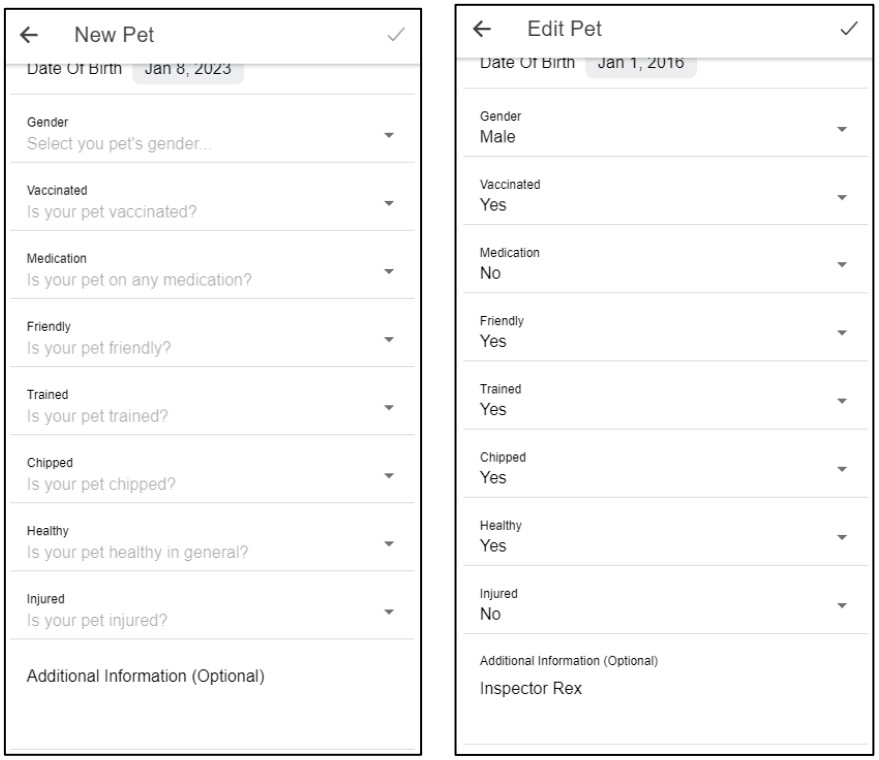
Οι σελίδες αυτές περιλαμβάνουν τη Φόρμα των στοιχείων του κατοικίδιου. Στην Σελίδα Επεξεργασίας ο χρήστης οδηγείται από το κουμπί επεξεργασίας της Σελίδας Προφίλ Κατοικίδιου, ενώ στη σελίδα Εισαγωγής Νέου Κατοικίδιου ο χρήστης οδηγείται από το κουμπί (+) προσθήκης νέου κατοικίδιου της Σελίδας Λίστας Κατοικίδιων.

Οι δύο αυτές σελίδες περιλαμβάνουν την ίδια Φόρμα Κατοικίδιου. Στην περίπτωση της νέας εισαγωγής η φόρμα αυτή είναι άδεια, ενώ στην περίπτωση της επεξεργασίας ενός προϋπάρχοντος προφίλ κατοικίδιου η Φόρμα είναι προσυμπληρωμένη με τα στοιχεία του. Οι δύο σελίδες φαίνονται παρακάτω.

Εικόνα 15: Σελίδα Εισαγωγής Νέου Κατοικίδιου (αριστερά) και Σελίδα Επεξεργασίας Κατοικίδιου (δεξιά) μέρος Α'



Εικόνα 16: Σελίδα Εισαγωγής Νέου Κατοικίδιου (αριστερά) και Σελίδα Επεξεργασίας Κατοικίδιου (δεξιά) μέρος Β'



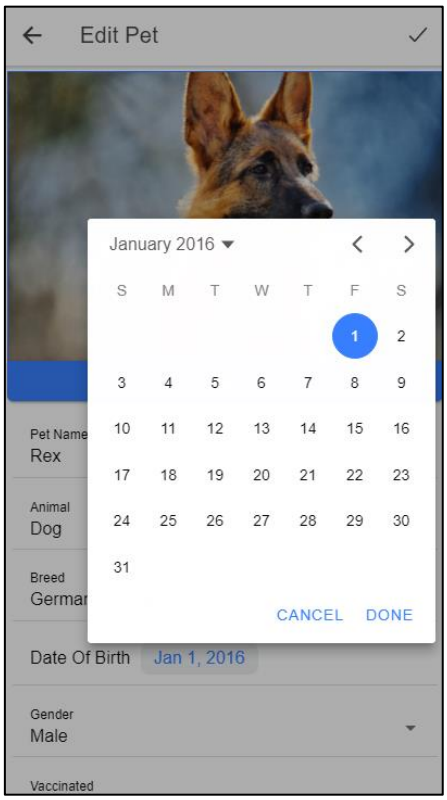
Και στις δύο σελίδες φαίνεται ο τίτλος New Pet και Edit Pet αντίστοιχα, ένα σύμβολο (√) για την αποθήκευση των στοιχείων και η φωτογραφία με την επιλογή της εύρεσης και αποθήκευσης. Αν ο χρήστης επιλέξει να αποθηκεύσει ή να αλλάξει τη φωτογραφία προφίλ, η εφαρμογή ενεργοποιεί την κάμερα (είτε του κινητού είτε του υπολογιστή) και δίνει την επιλογή λήψης νέας φωτογραφίας ή εναλλακτικά τη λειτουργία file upload με τη δυνατότητα πρόσβασης στο photo gallery (αν πρόκειται για κινητό) ή στο file explorer (αν πρόκειται για υπολογιστή).

Στη συνέχεια οι δύο εφαρμογές εμφανίζουν τα πεδία της φόρμας που ο χρήστης μπορεί να συμπληρώσει για το κατοικίδιό του, μεταξύ άλλων το όνομα, το είδος, τη ράτσα κλπ. Ακόμη, έχουν προστεθεί κάποια πεδία που θεωρούνται χρήσιμα για τον συνεργαζόμενο επαγγελματία, όπως για παράδειγμα αν το κατοικίδιο έχει εμβολιαστεί, αν είναι φιλικό, αν λαμβάνει κάποια αγωγή κλπ.

Ορισμένα από τα πεδία έχουν τη δυνατότητα συμπλήρωσης μέσα από μία αναδυόμενη λίστα (τα πεδία με το βέλος στα δεξιά). Η λίστα αυτή προσφέρει την επιλογή ενός μόνο χαρακτηριστικού.

Παράλληλα, στο πεδίο date of birth ο χρήστης επιλέγει την ημερομηνία από ένα αναδυόμενο ημερολόγιο, όπως φαίνεται παρακάτω. Σε αυτό το σημείο αξίζει να σημειωθεί ότι η ηλικία του κατοικίδιου που εμφανίζεται στη Σελίδα Προφίλ Κατοικίδιου υπολογίζεται αυτόματα από την εφαρμογή με βάση την τωρινή ημερομηνία.

Εικόνα 17: Αναδυόμενο ημερολόγιο στη Φόρμα Κατοικίδιου



Τέλος, όλα τα πεδία πέραν του τελευταίου είναι απαραίτητο να συμπληρωθούν ώστε να ενεργοποιηθεί το κουμπί αποθήκευσης της φόρμας.

Κεφάλαιο 5.6: Περιοχή Μηνυμάτων

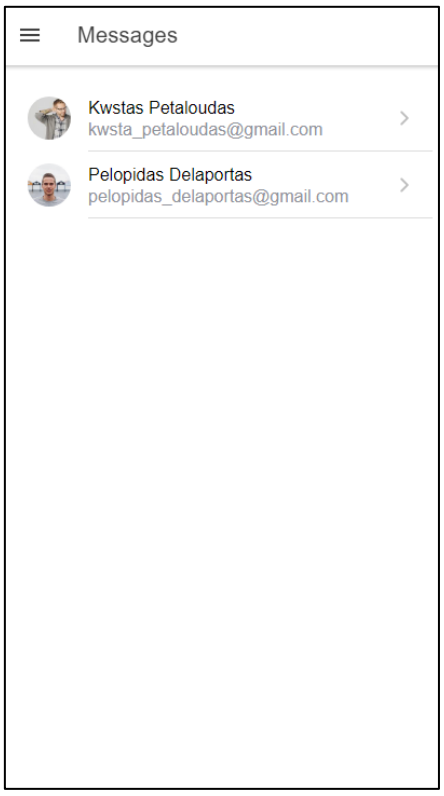
Η περιοχή μηνυμάτων περιλαμβάνει τις συνομιλίες που έχουν πραγματοποιηθεί μεταξύ PetOwners και PetWorkers. Οι συνομιλίες αυτές αποθηκεύονται και υπάρχει δυνατότητα συνέχισής τους.

Κεφάλαιο 5.6.1: Σελίδα Λίστας Συνομιλιών

Η σελίδα αυτή περιλαμβάνει μία λίστα με τα άτομα με τα οποία ο χρήστης έχει συνομιλήσει στο παρελθόν. Αν ο χρήστης είναι PetOwner τα άτομα της λίστας θα είναι PetWorkers και τούμπαλιν.

Ο εκάστοτε χρήστης οδηγείται στη σελίδα αυτή από το side menu της εφαρμογής, πατώντας στην επιλογή Messages. Η σελίδα που εμφανίζεται φαίνεται παρακάτω.

Εικόνα 18: Σελίδα Λίστας Συνομιλιών



Περιλαμβάνει το hamburger menu, τον τίτλο Messages και στη συνέχεια τη λίστα με τα άτομα με τα οποία ο χρήστης έχει συνομιλήσει. Για να μεταβεί ο χρήστης στις υφιστάμενες αποθηκευμένες συνομιλίες μπορεί να το κάνει είτε πατώντας στον χρήστη της επιλογής του, είτε σύροντας το αντικείμενο προς τα αριστερά. Στη δεύτερη περίπτωση εμφανίζεται ένα μπλε κουμπί που αν πατηθεί ο χρήστης μεταφέρεται στη συνομιλία.

Σημειώνεται ότι ο PetOwner μπορεί επίσης να εκκινήσει με πρωτοβουλία του μία νέα συνομιλία με κάποιον PetWorker με τον οποίο δεν έχει συνομιλήσει στο παρελθόν. Αυτό πραγματοποιείται στην περιοχή Find Pet-Services, όπως αναφέρθηκε σε προηγούμενο κεφάλαιο. Μόλις ξεκινήσει η συνομιλία με το νέο άτομο, αυτό θα προστεθεί στη λίστα συνομιλιών.

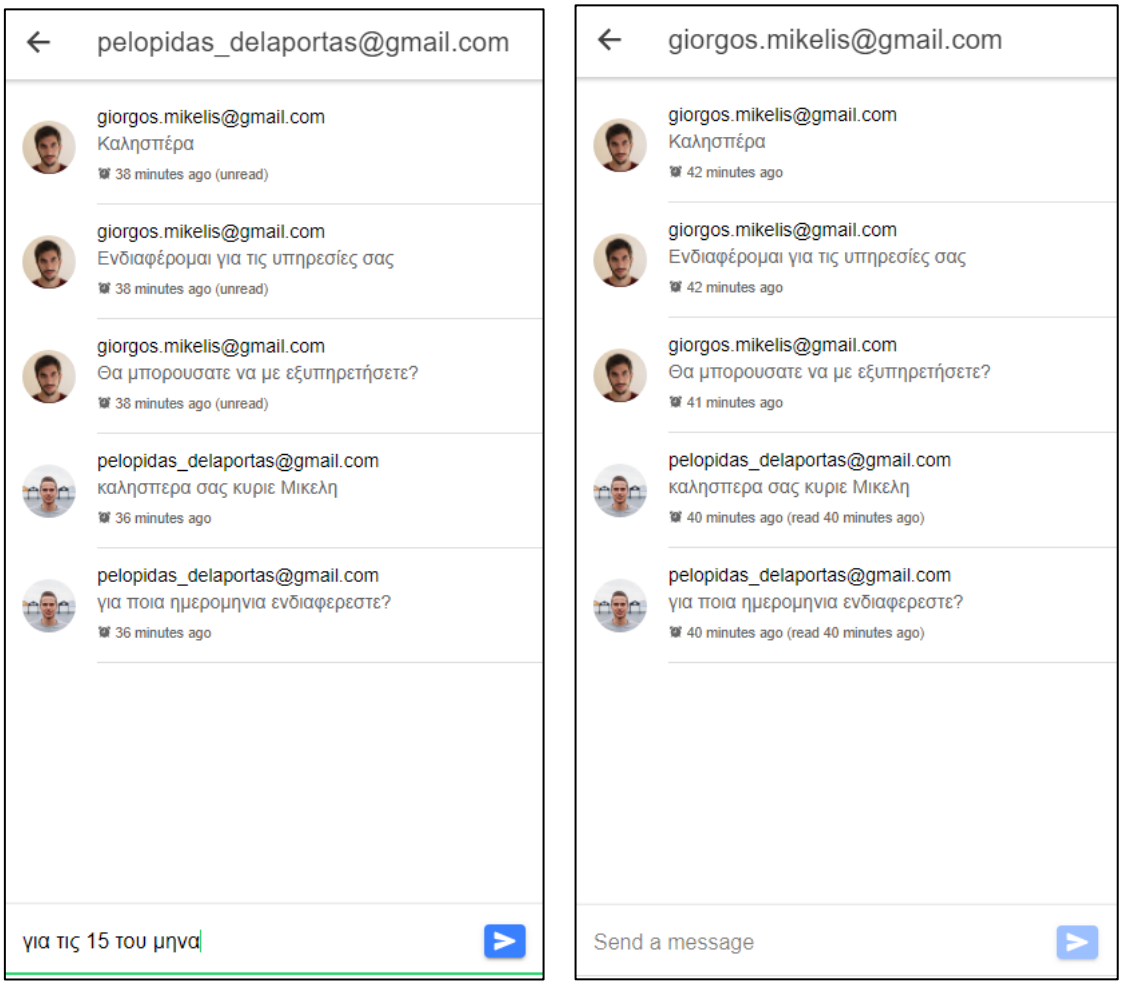
Αξίζει να σημειωθεί ότι ο PetWorker δεν έχει αυτή τη δυνατότητα της εκκίνησης νέας συνομιλίας με δική του πρωτοβουλία.

Τέλος, σημειώνεται ότι στη λίστα συνομιλιών εμφανίζεται σε κάθε αντικείμενο το όνομα του εκάστοτε χρήστη, η φωτογραφία του και το e-mail του. Ακόμη, η ταξινόμηση της λίστας πραγματοποιείται με κριτήριο τον χρόνο καταγραφής της πιο πρόσφατης συνομιλίας, που κατατάσσεται πρώτη.

Κεφάλαιο 5.6.2: Σελίδα Συνομιλίας

Στη σελίδα αυτή ο χρήστης μπορεί να ανασκοπήσει την καταγεγραμμένη συνομιλία του με τον συγκεκριμένο χρήστη που έχει επιλέξει. Μπορεί επίσης να συνεχίσει τη συνομιλία αυτή, στέλνοντας νέο μήνυμα. Παρακάτω παρουσιάζεται μία ενδεικτική συνομιλία.

Εικόνα 19: Παράδειγμα συνομιλίας με χρήστη στη Σελίδα Συνομιλίας



Στα αριστερά βλέπουμε την οθόνη της εφαρμογής PetOwner και στα δεξιά την αντίστοιχη οθόνη της εφαρμογής PetWorker.

Οι οθόνες περιλαμβάνουν πλήκτρο backward navigation που επιστρέφει τον χρήστη στη Σελίδα Μηνυμάτων. Ακόμη, ως τίτλο περιλαμβάνουν το e-mail του χρήστη με τον οποίο πραγματοποιείται η συνομιλία. Στη συνέχεια ακολουθεί η λίστα με τα μηνύματα στη χρονική σειρά που στάλθηκαν. Κάθε αντικείμενο μηνύματος περιλαμβάνει τη φωτογραφία και το e-mail του αποστολέα, όπως επίσης και τον χρόνο αποστολής.

Στο κάτω μέρος υπάρχει πεδίο εγγραφής νέου μηνύματος και πλήκτρο αποστολής, το οποίο ενεργοποιείται όταν ο χρήστης προσθέσει χαρακτήρες στο πεδίο.

Κεφάλαιο 6: Αρχιτεκτονική της Εφαρμογής

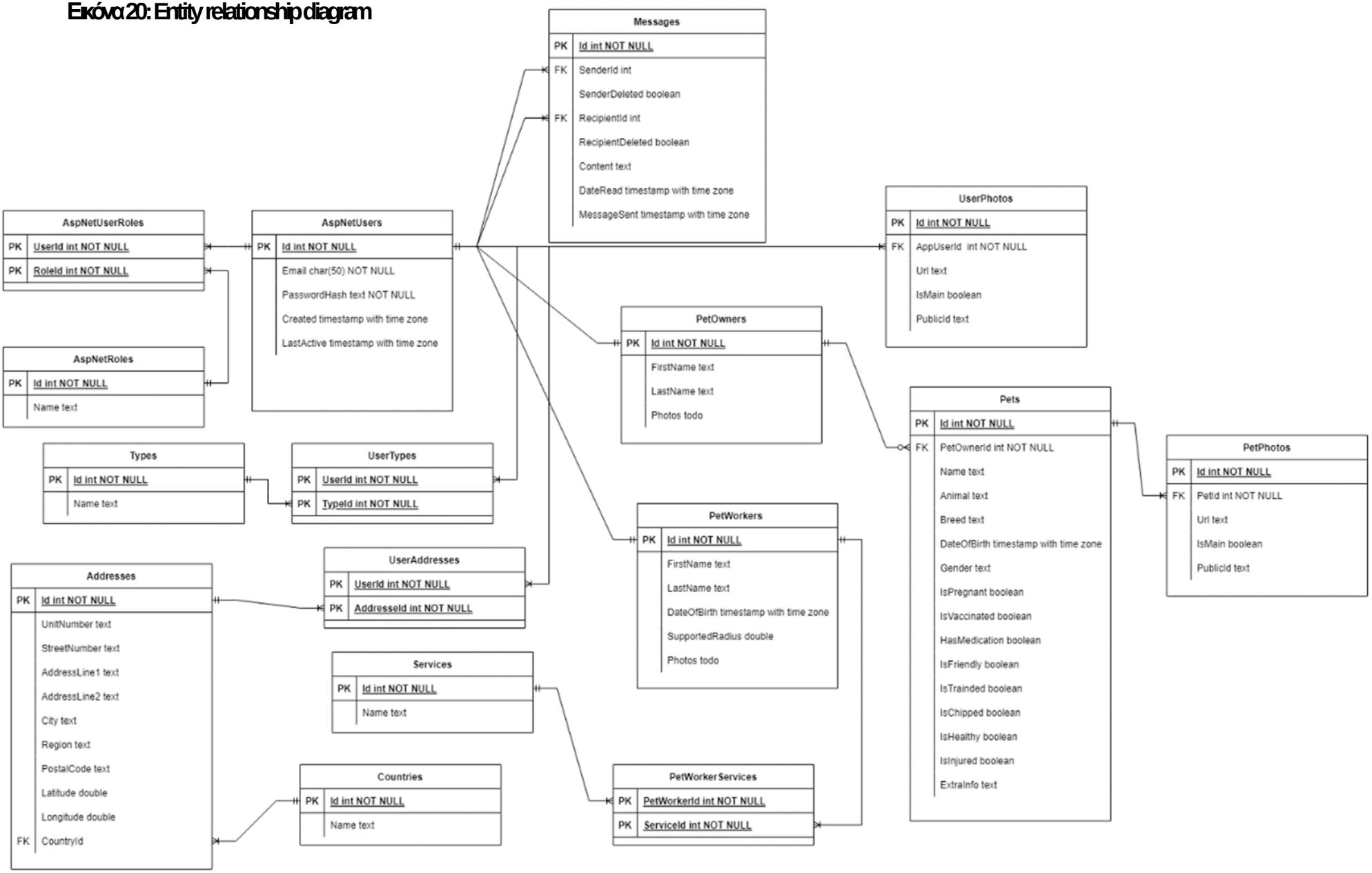
Στο κεφάλαιο αυτό θα γίνει μια περιγραφή της αρχιτεκτονικής της εφαρμογής.

Κεφάλαιο 6.1: Server-Side App (Back-End)

Το πρώτο βήμα ήταν ο σχεδιασμός της βάσης δεδομένων καθώς η ανάπτυξη της εφαρμογής βασίστηκε στον τρόπο με τον οποίο θα αποθηκεύονται τα δεδομένα στην βάση.

Η τελική εκδοχή του διαγράμματος σχέσεων των οντοτήτων (entity-relationship-diagram) φαίνεται στο παρακάτω σχήμα.

Εικόνα 20: Entity relationship diagram



Βάσει αυτού του σχήματος γράφτηκαν οι κλάσεις “Entity” καθώς και η κλάση DataContext που χρησιμοποιείται στο .NET ώστε να περιγραφεί το σχήμα της βάσης δεδομένων (database schema).

Όπως βλέπουμε στο παρακάτω απόσπασμα, στην κλάση DataContext ορίζονται τα DbSet (δηλαδή τα tables που θα δημιουργήσει στην βάση το EF όταν τρέξει ο κώδικας) και κάθε ένα από αυτά λαμβάνει την δομή του από το αντίστοιχο entity που δηλώνεται ανάμεσα στα angle brackets.

Σε αυτό το σημείο θα πρέπει να σημειωθεί ότι η κλάση DataContext επεκτείνει την κλάση IdentityDbContext η οποία είναι μια έτοιμη κλάση της βιβλιοθήκης του EntityFramework και η οποία αναλαμβάνει να κάνει ένα μεγάλο μέρος των εργασιών που σε άλλη περίπτωση θα έπρεπε να πραγματοποιηθούν από τον προγραμματιστή. Ορισμένες από αυτές τις εργασίες είναι η αυθεντικοποίηση του χρήστη, η κρυπτογράφηση των κωδικών, η αποκρυπτογράφηση των JWT που η κλάση θα συλλέξει αυτόματα από κάποιο http request και θα αναγνωρίσει σε ποιον χρήστη ανήκει και τον τύπο του χρήστη, αλλά και ο έλεγχος για το αν ο συγκεκριμένος χρήστης έχει την δικαιοδοσία να συλλέξει ή να τροποποιήσει τα συγκεκριμένα δεδομένα που ζητάει στο request.

```
namespace API.Data
{
    public class DataContext : IdentityDbContext<AppUser, AppRole, int,
        IdentityUserClaim<int>, AppUserRole, IdentityUserLogin<int>,
        IdentityRoleClaim<int>, IdentityUserToken<int>>
    {
        public DataContext(DbContextOptions options) : base(options)
        {
        }
        public DbSet<PetOwner> PetOwners { get; set; }
        public DbSet<PetWorker> PetWorkers { get; set; }
        public DbSet<Pet> Pets { get; set; }
        public DbSet<Country> Countries { get; set; }
        public DbSet<Address> Addresses { get; set; }
        public DbSet<UserAddress> UserAddresses { get; set; }
        public DbSet<PetService> PetServices { get; set; }
        public DbSet<PetWorkerService> PetWorkerServices { get; set; }
        public DbSet<Type> Types { get; set; }
        public DbSet<UserType> UserTypes { get; set; }
        public DbSet<Message> Messages { get; set; }
        public DbSet<Group> Groups { get; set; }
        public DbSet<Connection> Connections { get; set; }
        public DbSet<Booking> Bookings { get; set; }
        public DbSet<Slot> Slots { get; set; }
```

```
        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            optionsBuilder.EnableSensitiveDataLogging();
        }

        protected override void OnModelCreating(ModelBuilder builder)
        {
            base.OnModelCreating(builder);

            builder.Entity<AppUser>()
                .HasMany(ur => ur.UserRoles)
                .WithOne(u => u.User)
                .HasForeignKey(ur => ur.UserId)
                .IsRequired();

            builder.Entity<AppRole>()
                .HasMany(ur => ur.UserRoles)
                .WithOne(u => u.Role)
                .HasForeignKey(ur => ur.RoleId)
                .IsRequired();

            builder.Entity<UserType>()
                .HasKey(k => new { k.UserId, k.TypeId });

            builder.Entity<AppUser>()
                .HasMany(ut => ut.UserTypes)
                .WithOne(u => u.User)
                .HasForeignKey(ut => ut.UserId)
                .IsRequired();

            builder.Entity<Type>()
                .HasMany(ut => ut.UserTypes)
                .WithOne(u => u.Type)
                .HasForeignKey(ut => ut.TypeId)
```

```
            .IsRequired();

        builder.Entity<UserAddress>()
            .HasKey(k => new { k.UserId, k.AddressId });

        builder.Entity<AppUser>()
            .HasMany(ua => ua.UserAddresses)
            .WithOne(u => u.User)
            .HasForeignKey(ua => ua.UserId)
            .IsRequired();

        builder.Entity<Address>()
            .HasMany(ua => ua.UserAddresses)
            .WithOne(u => u.Address)
            .HasForeignKey(ut => ut.AddressId)
            .IsRequired();

        builder.Entity<PetWorkerService>()
            .HasKey(k => new { k.PetWorkerId, k.PetServiceId });

        builder.Entity<PetWorker>()
            .HasMany(ws => ws.PetWorkerServices)
            .WithOne(w => w.PetWorker)
            .HasForeignKey(ws => ws.PetWorkerId)
            .IsRequired();

        builder.Entity<PetService>()
            .HasMany(ws => ws.PetWorkerServices)
            .WithOne(w => w.PetService)
            .HasForeignKey(ws => ws.PetServiceId)
            .IsRequired();

        builder.Entity<Message>()
            .HasOne(u => u.Recipient)
            .WithMany(m => m.MessagesReceived)
            .OnDelete(DeleteBehavior.Restrict);
```

```
            builder.Entity<Message>()
                .HasOne(u => u.Sender)
                .WithMany(m => m.MessagesSent)
                .OnDelete(DeleteBehavior.Restrict);

            builder.ApplyUtcDateTimeConverter();
        }
    }
}
```

Κεφάλαιο 6.1.1: Repository Pattern και Unit of Work Pattern

Το Repository pattern είναι ένα design pattern (προγραμματιστικό μοτίβο σχεδίασης) που προσφέρει έναν αφαιρετικό τρόπο στην αποθήκευση και στην ανάκτηση δεδομένων από τη βάση δεδομένων. Αυτό το pattern προσδίδει έναν τρόπο για την δημιουργία ενός ξεχωριστού «επιπέδου» (layer) για τον κώδικα (ή αλλιώς την λογική) της πρόσβασης των δεδομένων (data access logic) η οποία μπορεί να χρησιμοποιηθεί από διάφορα μέρη ή στοιχεία της εφαρμογής (π.χ. controllers). Η βασική ιδέα πίσω από το Repository Pattern είναι η δημιουργία ενός “signle point of access” των δεδομένων, το οποίο δύναται να χρησιμοποιηθεί από διάφορα μέρη της εφαρμογής χωρίς να χρειάζεται να επαναλαμβάνεται κώδικας. Αυτή η δυνατότητα πληροί μια βασική προϋπόθεση (principle) ενός καθαρού κώδικα που λέγεται DRY (Don’t Repeat Yourself). Πιο αναλυτικά, δημιουργούνται κλάσεις -οι οποίες σε αυτό το πλαίσιο ονομάζονται repositories- στις οποίες δηλώνονται μέθοδοι που εμπεριέχουν τον κώδικα τον οποίο το Entity Framework θα μετατρέψει σε SQL queries για να επικοινωνήσει με την βάση δεδομένων.

Το Unit of Work pattern είναι κι αυτό ένα design pattern το οποίο βοηθάει στην διαχείριση πολλαπλών διεργασιών αποθήκευσης δεδομένων στην βάση (data storage operations) καθώς τις διαχειρίζεται ως «μία μονάδα εργασίας» (unit of work). Αυτό το pattern πολύ συχνά χρησιμοποιείται συνδυαστικά με το repository pattern, όπου το Unit of Work pattern βοηθά στη διαχείριση της πολλαπλών διεργασιών από ένα ή περισσότερα repositories. Με απλά λόγια, το Unit of Work «παρακολουθεί» τις αλλαγές που κάνουν στην βάση οι μέθοδοι των repositories και κάνει εφικτό ένα τελικό “commit” στην βάση εφόσον έχουν πάει όλα καλά στις ενέργειες αυτές, ή διαφορετικά ένα πισωγύρισμα (rollback) αν έχουν παρουσιαστεί λάθη σε μία ή περισσότερες.

Παρακάτω παρουσιάζεται η κλάση UnitOfWork που γράφτηκε ώστε να υλοποιήσει το Unit Of Work pattern :

```
namespace API.Data
{
    public class UnitOfWork : IUnitOfWork
    {
        private readonly DataContext _context;
        private readonly IMapper _mapper;

        public UnitOfWork(DataContext context, IMapper mapper)
```

```
        {
            _context = context;
            _mapper = mapper;
        }

        public IUserRepository UserRepository => new
UserRepository(_context, _mapper);

        public IPetRepository PetRepository => new PetRepository(_context,
_mapper);
        public IPetServicesRepository PetServicesRepository => new
PetServicesRepository(_context, _mapper);

        public IMessageRepository MessageRepository => new
MessageRepository(_context, _mapper);

        public async Task<bool> Complete()
        {
            return await _context.SaveChangesAsync() > 0;
        }

        public bool HasChanges()
        {
            return _context.ChangeTracker.HasChanges();
        }
    }
}
```

Παρατηρούμε την μέθοδο Complete η οποία θα επιστρέψει ένα ασύγχρονο Task τύπου boolean, αφού επιχειρήσει να σώσει τις αλλαγές στην βάση. Θα επιστρέψει true αν δεν προκύψουν σφάλματα και false εάν προκύψουν.

Παρακάτω παρουσιάζεται ένα παράδειγμα από τον κώδικα της εφαρμογής και συγκεκριμένα απ΄τον UsersController.cs όπου γίνεται χρήση του Unit Of Work.

```
[HttpPut("update-pet-owner")]
        public async Task<ActionResult>
UpdatePetOwnerProfile(PetOwnerProfileUpdateDto petOwnerProfileUpdateDto)
        {
            var user = await
_unitOfWork.UserRepository.GetUserByUsernameAsync(User.GetUsername());
```

```
            _mapper.Map(petOwnerProfileUpdateDto, user);
            _unitOfWork.UserRepository.UpdateUser(user);

            var addressDto = petOwnerProfileUpdateDto.Address;
            await
_unitOfWork.UserRepository.AddAddressToUserAndSetToMain(addressDto,
user.Id);

            var petOwner = await
_unitOfWork.UserRepository.GetPetOwnerAsync(user.Id);

            _mapper.Map(petOwnerProfileUpdateDto, petOwner);
            _unitOfWork.UserRepository.UpdatePetOwner(petOwner);

            if (await _unitOfWork.Complete()) return NoContent();

            return BadRequest("Failed to update user");
        }
```

Όπως βλέπουμε στον παραπάνω κώδικα, έχουμε μια μέθοδο του UsersController που λέγεται “UpdatePetOwnerProfile”. Όταν ο controller αυτός λάβει στο endpoint …/users/update-pet-owner ένα http request από κάποιον client, αφού τρέξει όλες τις μεθόδους του UserRepository που φαίνονται στον κώδικα (οι οποίες θα κάνουν τις ανάλογες ενέργειες στην βάση), στο τέλος θα καλέσει την μέθοδο Complete του Unit Of Work και θα επιστρέψει 204 Success No Content code εφόσον δεν προκύψουν σφάλματα και 400 BadRequest error code εφόσον προκύψουν.

Κεφάλαιο 6.1.2: LINQ, DTOs, και AutoMapper

Όπως προαναφέρθηκε, σε μια εφαρμογή που χρησιμοποιείται κάποιο ORM σύστημα (για παράδειγμα Entity Framework στο .NET ή Hibernate στην Java) δεν θα γραφτούν τα συνήθη SQL queries για την επικοινωνία με την βάση δεδομένων, αλλά αντίθετα θα συνταχθεί ο κατάλληλος κώδικας στην back-end εφαρμογή. Εν συνεχεία, ο κώδικας αυτός θα μετατραπεί στα κατάλληλα SQL queries από το ORM.

Σε μια .NET εφαρμογή η οποία χρησιμοποιεί το Entity Framework, ο κώδικας της γράφεται συνήθως με την LINQ (Language Integrated Query).

Το LINQ είναι ένα σύνολο λειτουργιών στο .NET το οποίο παρέχει έναν συγκεκριμένο τρόπο σύνταξης στην C# (αλλά και στην VB.NET) με τον οποίο τα δεδομένα μπορούν να φιλτραριστούν, να ταξινομηθούν, να μετασχηματιστούν και να αποθηκευτούν με ευκολία είτε στη μνήμη είτε σε μια βάση δεδομένων.

Μια καλή πρακτική, κατά την δημιουργία των controllers σε ένα web API, είναι ο διαχωρισμός του business logic επιπέδου από το data access επίπεδο. Της από της βασικούς τρόπους που επιτυγχάνεται της ο διαχωρισμός είναι η χρήση των DTOs (Data Transfer Object) για την περιγραφή και μορφοποίηση της πληροφορίας στην είσοδο και στην έξοδο του

controller. Για παράδειγμα, έστω ότι υπάρχει της controller που όταν δεχτεί ένα http request από κάποιον client πρέπει να συλλέξει την πληροφορία για έναν συγκεκριμένο χρήστη από την βάση. Το Entity Framework θα συλλέξει αυτήν την πληροφορία και θα επιστρέψει το “Entity” User που αντιστοιχεί σε αυτόν τον χρήστη, όπου κάθε στήλη του πίνακα της βάσης αντιστοιχεί σε ένα property αυτού του αντικειμένου. Της φορές της, για διάφορους λόγους, μπορεί να μην χρειάζεται να επιστραφεί όλη αυτή η πληροφορία στον client. Συνεπώς, αντί να επιστραφεί το Entity αντικείμενο, μπορεί να οριστεί ένα άλλο «αντικείμενο μεταφοράς», ή αλλιώς DTO, το οποίο έχει οριστεί ώστε να διαθέτει μόνο τα properties που χρειάζονται από αυτό το Entity και αυτό να επιστρέφεται ως έξοδος του controller στον client. Η ίδια τεχνική μπορεί να χρησιμοποιηθεί και για την είσοδο του controller. Συγκεκριμένα, με τα DTOs μπορεί να οριστεί η δομή που θα πρέπει να έχει η πληροφορία στο body του request.

Στο .NET μπορεί να γίνει χρήση της AutoMapper βιβλιοθήκης για αυτήν την αντιστοίχιση των αντικειμένων με εύχρηστο και γρήγορο τρόπο. Στο συγκεκριμένο παράδειγμα, μια τέτοια αντιστοίχιση θα ήταν αυτή του του Entity User στο DTO User και αντίθετα.

Παρακάτω ακολουθεί μία μέθοδος από το αρχείο UserRepository.cs η οποία εμπεριέχει ένα “query” γραμμένο με την LINQ σύνταξη. Η παρακάτω εντολή θα συλλέξει πληροφορία από την βάση δεδομένων και συγκεκριμένα από τον πίνακα PetOwners. Συγκεκριμένα, θα επιστρέψει την πληροφορία του PetOwner του οποίου το id ισούται με την παράμετρο id της μεθόδου, δηλαδή ένα αντικείμενο Entity τύπου PetOwner με την πληροφορία του συγκεκριμένου PetOwner. Στη συνέχεια, με την βοήθεια του AutoMapper library, θα μετατρέψει/αντιστοιχίσει αυτό το Entity αντικείμενο στο PetOwnerDto αντικείμενο που διαθέτει μόνο την πληροφορία που χρειάζεται να επιστραφεί στον client.

```
public async Task<PetOwnerDto> GetPetOwnerDtoAsync(int id)
{
       return await _context.PetOwners
            .Where(petOwner => petOwner.Id == id)
            .ProjectTo<PetOwnerDto>(_mapper.ConfigurationProvider)
            .SingleOrDefaultAsync();
}
```

Εφαρμόζοντας όλες παραπάνω πρακτικές ο κώδικας στους Controllers παραμένει μικρός, καθαρός και ευανάγνωστος, αποκρύπτοντας τις λεπτομέρειες του τρόπου με τον οποίο φέρνει την πληροφορία, με αποτέλεσμα κατά την ανάγνωση του κάθε endpoint να είναι ξεκάθαρο το «τι» είναι αυτό που κάνει -δηλαδή ποια είναι η μορφή της πληροφορίας που περιμένει στην είσοδο καθώς και ποια είναι τι μορφή έχει η πληροφορία επιστρέφει- και όχι το «πως» το κάνει.

Κεφάλαιο 6.2: Client-Side App (Front-End)

Για την ανάπτυξη του front-end μιας τέτοιας εφαρμογής όπου υπάρχει αυτή η καθαρή διάκριση των δυνατοτήτων και λειτουργιών που θα προσφέρει στον χρήστη ανάλογα με τον τύπο του (Pet Owner ή Pet Worker), θα μπορούσαν να ακολουθηθούν δύο προσεγγίσεις. Η μία προσέγγιση θα μπορούσε να ήταν η ανάπτυξη δύο «δίδυμων» εφαρμογών, μία εφαρμογή για τον χρήστη τύπου PetOwner και η άλλη για τον χρήστη τύπου PetWorker. Η άλλη προσέγγιση θα μπορούσε να ήταν η ανάπτυξη μιας μοναδικής εφαρμογής, η οποία κατά την διαδικασία εγγραφής του χρήστη θα υπήρχε η δυνατότητα επιλογής της εγγραφής του είτε ως PetOwner είτε ως PetWorker. Στην συνέχεια, η εφαρμογή θα εμφάνιζε στον χρήστη τις κατάλληλες σελίδες και λειτουργίες σύμφωνα με τον τύπο του.

Για την ανάπτυξη της εφαρμογή της παρούσας εργασίας επιλέχθηκε η προσέγγιση των δύο ξεχωριστών εφαρμογών, καθώς κάθε μία από αυτές θα είναι πιο μικρή, γρήγορη και

«ελαφριά» σε σχέση με μία πιο μεγάλη εφαρμογή με δύο μέρη, της οποίας ο κώδικας που αφορά στον αντιθέτου τύπου χρήστη από αυτόν του χρήστη ο οποίος την χρησιμοποιεί κάθε φορά, καθίσταται αχρείαστος και την κάνει πιο ογκώδη και άρα πιο «αργή» χωρίς ουσιαστικό λόγο.

Υπάρχει φυσικά και η περίπτωση όπου κάποιος χρήστης θα θέλει να χρησιμοποιήσει την εφαρμογή και με τους δύο τρόπους, δηλαδή και ως PetOwner και ως PetWorker. Στην περίπτωση αυτή ο χρήστης θα μπορεί να κατεβάσει και τις δύο εφαρμογές από το App Store ή το Google Play ή να επισκέπτεται την web εφαρμογή που καλύπτει κάθε φορά την ανάγκη του.

Στην περίπτωση αυτή που ο χρήστης χρησιμοποιεί την εφαρμογή και με τους δύο τρόπους, η εγγραφή του στην εφαρμογή γίνεται μόνο μία φορά. Πιο συγκεκριμένα, κατά την εγγραφή ενός χρήστη με το email και τον κωδικό του, τα στοιχεία αυτά αποθηκεύονται στον πίνακα AspNetUsers στην βάση δεδομένων. Ταυτόχρονα, γίνεται και η εγγραφή ενός νέου record στον πίνακα PetOwners ή στον πίνακα PetWorkers (ανάλογα με τον τύπο της εφαρμογής με την οποία έκανε εγγραφή), με το id αυτού του record να ισούται με το id της εγγραφής στον πίνακα AspNetUsers, ώστε να υπάρχει η αντιστοίχιση του κάθε χρήστη του AspNetUsers πίνακα, με ένα PetWorker ή ένα PetOwner προφίλ. Στην συνέχεια, όταν ο χρήστης αυτός θέλει να χρησιμοποιήσει και την εφαρμογή του αντίθετου τύπου από αυτόν που έκανε αρχικά την εγγραφή, δεν χρειάζεται να ξανακάνει εγγραφή. Για την ακρίβεια αν επιλέξει “εγγραφή” αντί για “σύνδεση” και δώσει ίδιο email με πριν, η εφαρμογή θα του εμφανίσει ένα μήνυμα λάθους το οποίο θα τον πληροφορεί ότι υπάρχει ήδη χρήστης με το συγκεκριμένο email και θα τον παροτρύνει να κάνει απλώς σύνδεση με αυτό το email. Έτσι, όταν για παράδειγμα ένας χρήστης έχει ήδη «εγγραφεί» με το email του στην εφαρμογή ως PetWorker και στην συνέχεια «συνδεθεί» για πρώτη φορά με το ίδιο email στην PetOwner εφαρμογή, τότε θα δημιουργηθεί ένα νέο record στον πίνακα PetOwner, με το id του νέου αυτού record να ισούται με το id που έχει ο χρήστης αυτός στον πίνακα AspNetUsers και PetWorkers. Με τον τρόπο αυτό ένας τέτοιος χρήστης μπορεί να έχει ένα PetOwner και ένα PetWorker προφίλ μόνο με ένα email και password. Γενικά, πέρα από το email και password, στον πίνακα AspNetUsers αποθηκεύονται όλα εκείνα τα στοιχεία που είναι κοινά (όπως Όνομα, Επώνυμο, Διεύθυνση κ.α), ώστε να μην υπάρχουν διπλά δεδομένα στην βάση, ενώ στους πίνακες PetOwner και PetWorkers υπάρχουν μόνο τα στοιχεία που αφορούν τον καθένα.

Κεφάλαιο 6.2.1: State Management

Σε μία web εφαρμογή, και κατ’ επέκταση σε μία hybrid mobile εφαρμογή, ο τρόπος διαχείρισης των δεδομένων της (state), ονομάζεται “state management”. Το state μπορεί να περιλαμβάνει οποιαδήποτε πληροφορία που απαιτείται για να λειτουργήσει μία εφαρμογή, όπως τα δεδομένα των χρηστών, την κατάσταση των διαδικασιών κλπ. Η διαχείριση του state μπορεί να γίνει με διάφορους τρόπους, αλλά ο πιο κοινός τρόπος είναι μέσω της χρήσης μιας βιβλιοθήκης ή framework που παρέχει ένα μηχανισμό για τη διαχείριση του state. Στην React αυτό συχνά γίνεται με την Redux βιβλιοθήκη που υλοποιεί το “redux pattern”. Στην Angular το state management μπορεί να γίνει με δύο τρόπους, είτε με την NgRx βιβλιοθήκη, η οποία είναι η υλοποίηση του redux pattern που παρέχει η Angular, είτε με την RxJS βιβλιοθήκη για την δημιουργία ενός δικτύου από “observables” και “subscribers” για την ενημέρωση των δεδομένων της εφαρμογής. Θεωρείται “best practice” (καλύτερη πρακτική) η χρήση της RxJS βιβλιοθήκης να γίνεται σε συνδυασμό με τα “Angular Services”. Στην παρούσα εφαρμογή επιλέχθηκε ο δεύτερος τρόπος για την διαχείριση του state.

Το RxJs είναι μια βιβλιοθήκη για την δημιουργία και επεξεργασία «ροών δεδομένων» (streams) στην JavaScript. Τα αντικείμενα με τα οποία γίνονται διαχειρίσιμα αυτά τα streams πληροφορίας ονομάζονται Observables. Υπάρχει η δυνατότητα δημιουργίας observable από οποιαδήποτε πηγή δεδομένων, από ένα απλό array, μέχρι το response ενός http request. Επίσης υπάρχει η δυνατότητα δημιουργίας ενός ή περισσότερων «subscribers» ενός observable, που θα ενημερώνονται κάθε φορά που θα εκδίδεται μια νέα τιμή από το observable αυτό. Έτσι, μπορεί να γίνεται η ενημέρωση των δεδομένων της εφαρμογής με

προγραμματισμένο τρόπο, ώστε να υπάρχει η βεβαιότητα ότι τα δεδομένα μεταξύ των διάφορων σημείων της εφαρμογής θα είναι συγχρονισμένα σε όλες τις περιπτώσεις.

Όπως σημειώθηκε παραπάνω, είναι μια καλή πρακτική η RxJs βιβλιοθήκη να χρησιμοποιείται συνδυαστικά με τα Angular Services. Τα Angular services είναι κλάσεις που παρέχουν μία λειτουργία ή μία συλλογή από λειτουργίες σε ένα σημείο της εφαρμογής. Τα services συνήθως δημιουργούνται ως singletons, δηλαδή δημιουργούνται μόνο μία φορά (ένα "single instance" της κλάσης) για όλη τη διάρκεια που μένει ενεργή η εφαρμογή, και παρέχουν τη λειτουργία τους σε οποιοδήποτε component ή service που το ζητήσει. Τα services μπορούν να χρησιμοποιηθούν για να αποθηκεύσουν δεδομένα, να καλούν διαδικασίες ή να επικοινωνούν με εξωτερικά συστήματα, όπως ένα web API. Ο καλύτερος τρόπος για την επικοινωνία με webAPIs είναι με την βοήθεια του HTTP Client της Angular. Ο HTTP client διαθέτει μεθόδους ονομασμένους σαν http verbs (get, put, delete, post, κτλ), οι οποίες επιστρέφουν αντικείμενα τύπου Observables της RxJs βιβλιοθήκης. Συνήθως, τα services είναι τοποθετημένα σε ξεχωριστά αρχεία και είναι διαθέσιμα σε όλα τα components της εφαρμογής μέσω της διεπαφής Dependency Injection (DI) της Angular. Έτσι, ένα component της εφαρμογής που είναι υπεύθυνο για την προβολή μιας πληροφορίας στον χρήστη, μέσω του Dependency Injection στον constructor της κλάσης του, μπορεί να έχει πρόσβαση στις μεθόδους του service του οποίου το instance κάνει inject ο constructor. Έστω ότι μια από τις μεθόδους αυτές κάνει ένα http request στο web API για την συλλογή αυτής της πληροφορίας. Η μέθοδος αυτή θα επιστρέφει ένα Observable αντικείμενο αυτής της πληροφορίας, πάνω στο οποίο το component αυτό, εφόσον το ίδιο καλεί αυτήν την μέθοδο και άρα έχει πρόσβαση στο observable αντικείμενο που επιστρέφει, μπορεί να κάνει "subscribe" στο observable αυτό ώστε να πάρει αυτήν την πληροφορία και να την προβάλει στον χρήστη.

Παρακάτω ακολουθεί ένα παράδειγμα από το αρχείο auth.service.ts της εφαρμογής. Στο αρχείο αυτό ορίζεται η κλάση AuthService ή οποία εμπεριέχει μεθόδους που αφορούν στην αυθεντικοποίηση του χρήστη από το webAPI. Συγκεκριμένα, στο απόσπασμα που ακολουθεί φαίνεται η μέθοδος signup για την εγγραφή του χρήστη στον server.

```
  signup(email: string, password: string): Observable<AuthResponseData> {
    const body = { email: email, password: password, userTypeId: 1 };

    return this.http.post<AuthResponseData>(this.apiUrl +
'account/register', body).pipe(
      tap(this.setUserData.bind(this))
    );
  }
```

Όπως βλέπουμε στην μέθοδο αυτή καλείται η μέθοδος post του HttpClient. Η μέθοδος αυτή θα «χτυπήσει» το endpoint …/account/register του webAPI ενώ το σώμα του request ορίζεται από το const "body". Η post μέθοδος θα επιστρέψει το Observable τύπου AuthResponseData (ένα custom TypeScript Interface που δημιουργήθηκε για την καθαρή περιγραφή της δομής του response αυτής της κλήσης) το οποίο Observable η ίδια η signup μέθοδος θα το επιστρέψει με την σειρά της όταν αυτή κληθεί. Η κλήση αυτή γίνεται μέσα στο αρχείο auth.page.ts, όπως βλέπουμε παρακάτω.

```
  authenticate(email: string, password: string) {
    const loadingMessage = this.isLogin ? this.strLoggingIn :
this.strCreatingAccount;
    const genericErrorMessage = this.isLogin ? this.strLoginFailedMsg :
this.strSignupFailedMsg;
    this.isLoading = true;
```

```
    this.loadingCtrl
      .create({ keyboardClose: true, message: loadingMessage })
      .then(loadingEl => {
        loadingEl.present();
        let authObs: Observable<AuthResponseData>;
        if (this.isLogin) {
          authObs = this.authService.login(email, password);
        } else {
          authObs = this.authService.signup(email, password);
        }
        authObs.subscribe(
          resData => {
            this.isLoading = false;
            loadingEl.dismiss();
            this.router.navigateByUrl('/pet-services');
            this.form.reset();
          },
          errRes => {
            loadingEl.dismiss();
            const code = errRes.error;
            let message = genericErrorMessage;
            if (code === 'An account with this email already exists') {
              message = this.strEmailExists;
            } else if (code === 'Invalid email') {
              message = this.strEmailNotFound;
            } else if (code.title && code.title === 'Unauthorized') {
              message = this.strIncorrectPsw;
            }
            this.showAlert(message);
          }
        );
      });
  }
```

Όπως φαίνεται, η μεταβλητή authObs ορίζεται ως το αντικείμενο επιστροφής της κλήσης της signup μεθόδου (στην περίπτωση που το isLogin boolean property έχει τιμή false), το οποίο είναι τύπου Observable<AuthResponseData>. Λίγες γραμμές παρακάτω γίνεται subscribe πάνω σε αυτό το αντικείμενο, δηλαδή καλείται η μέθοδος subscribe πάνω στο authObs η οποία επιστρέφει το AuthResponseData (resData) καθ' αυτό κατά την επιτυχία του http call, ή επιστρέφει error (errRes) κατά την αποτυχία του.

Κεφάλαιο 6.2.2: Η δομή της εφαρμογής στο Ionic/Angular περιβάλλον

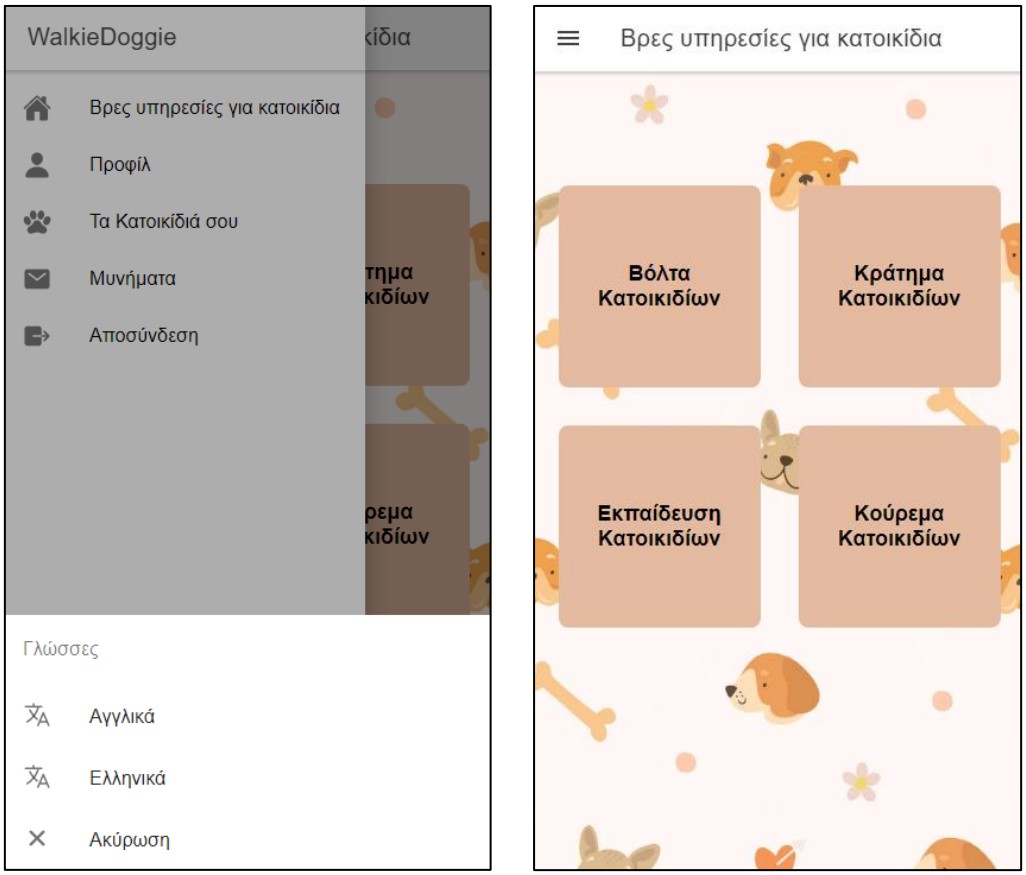
Μια Angular εφαρμογή χωρίζεται σε πρώτο επίπεδο με τα modules, όπου συνήθως κάθε module αφορά σε ένα συγκεκριμένο "feature" ή "aspect" της εφαρμογής. Στην συνέχεια, το περιεχόμενο του κάθε module χωρίζεται σε components. Σε μια Angular/Ionic εφαρμογή ο διαχωρισμός γίνεται σε πρώτο επίπεδο με τα pages, όπου κάθε page αντιστοιχεί σε μία σελίδα

ή οθόνη της εφαρμογής και συνήθως κάθε page είναι ταυτόχρονα και ένα Angular module. Έπειτα το page αυτό μπορεί να χωρίζεται και σε επιμέρους Angular components που το αποτελούν. Το Ionic Framework διαχειρίζεται το κάθε page με τέτοιον τρόπο ώστε να δίνεται η αίσθηση στον χρήστη ότι πρόκειται για μια «κανονική» native mobile εφαρμογή. Για παράδειγμα, μπορεί να οριστεί η πλοήγηση στην εφαρμογή να έχει την μορφή stack, οπότε όσο ο χρήστης προχωράει από ένα page σε ένα επόμενο, το instance των παλαιότερων με όλα τα δεδομένα τους (state) να παραμένει στο background και κάθε νέα οθόνη να φορτώνεται πάνω στην προηγούμενη, ενώ και τα γραφικά αυτής της εναλλαγής των οθονών να αντιστοιχούν στα native γραφικά μιας τέτοιας πλοήγησης. Μπορεί επίσης η πλοήγηση να έχει την μορφή menu, είτε με bottom bars, είτε με το γνωστό hamburger menu στο πάνω αριστερά σημείο της οθόνης, το οποίο ανοίγει ένα μικρό menu page με όλες τις διαθέσιμες οθόνες. Μπορεί ακόμα να γίνει και ένας συνδυασμός όλων των παραπάνω για το τελικό navigation της εφαρμογής. Σε κάθε περίπτωση το Ionic Framework προσφέρει έτοιμα UI components (hamburger menu, page headers κ.α) ώστε η εφαρμογή να μην διαφέρει καθόλου από μια android ή iOS εφαρμογή. Ωστόσο αυτά τα UI components είναι διαθέσιμα και προς παραμετροποίηση, τόσο αισθητικά όσο και λειτουργικά.

Κεφάλαιο 6.2.3: Δυνατότητα επιλογής γλώσσας της εφαρμογής

Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, η εφαρμογή δίνει την δυνατότητα στον χρήστη να επιλέξει την γλώσσα των λεκτικών της εφαρμογής μεταξύ της Αγγλικής και της Ελληνικής, όπως φαίνεται στις παρακάτω εικόνες.

Εικόνα 21: Λειτουργία μετάφρασης στα Ελληνικά



Η λειτουργία αυτή υλοποιήθηκε με την βοήθεια της βιβλιοθήκης “ngx-translate”. Για τους σκοπούς αυτής της δυνατότητας, δημιουργήθηκαν δύο .json αρχεία, el.json και en.json, τα οποία τοποθετήθηκαν στο path …/app/assest/i18n. Τα αρχεία αυτά διαθέτουν το καθένα ένα

JSON object με key-value-pairs (KVPs), που αποτελούν όλους τους λεκτικούς πόρους της εφαρμογής, δηλαδή τα κείμενα ή τις λέξεις που θα εμφανιστούν στον τελικό χρήστη. Τα κλειδιά (keys) του el.json πρέπει να ταυτίζονται με αυτά του en.json, ενώ οι τιμές (values) που έχουν είναι στην ελληνική γλώσσα για το el.json και στην αγγλική για το en.json αντίστοιχα. Έτσι σε όλη την εφαρμογή δεν γράφεται πουθενά “hardcoded” κείμενο, αντιθέτως χρησιμοποιούνται μόνο τα «κλειδιά» αυτών των JSON αρχείων και στην συνέχεια η βιβλιοθήκη ngx-translate τα «μεταφράζει» αυτόματα στην ελληνική ή την αγγλική τιμή, ανάλογα με την επιλογή της γλώσσας που έχει κάνει ο χρήστης.

Κεφάλαιο 6.2.4: Πρόσβαση στις Native λειτουργίες της συσκευής

Το Ionic framework και συγκεκριμένα το Capacitor runtime περιβάλλον εμπεριέχει πολλές λειτουργίες σε μορφή plugins, για την πρόσβαση στις native λειτουργίες της mobile συσκευής. Παρακάτω ακολουθούν κάποια αποσπάσματα του κώδικα που κάνουν χρήση αυτών των plugins:

```
  private locateUser() {
    if (!Capacitor.isPluginAvailable('Geolocation')) {
      this.showErrorAlert();
      return;
    }
    this.isLoading = true;
    Geolocation.getCurrentPosition()
      .then(geoPosition => {
        const coordinates: Coordinates = {
          lat: geoPosition.coords.latitude,
          lng: geoPosition.coords.longitude
        };
        this.createAddress(coordinates.lat, coordinates.lng);
        this.isLoading = false;
      })
      .catch(err => {
        this.isLoading = false;
        this.showErrorAlert();
      });
  }
```

Στο παραπάνω απόσπασμα φαίνεται η μέθοδος ‘locateUser’ που βρίσκεται στο αρχείο lacation-picker.component.ts. Στην μέθοδο αυτή γίνεται αρχικά ένας έλεγχος για το αν είναι διαθέσιμη η λειτουργία Geolocation στην συσκευή. Η μέθοδος αναζητά την τωρινή τοποθεσία του χρήστη με το Capacitor Geolocation plugin. Ο χρήστης θα ερωτηθεί αν δίνει δικαίωμα στην εφαρμογή να έχει πρόσβαση στην τοποθεσία του. Αν ο χρήστης δώσει δικαίωμα το plugin θα μπορέσει να προσπελάσει τα στοιχεία της τοποθεσίας και στην συνέχεια θα τα περάσει στο “geoPosition” αντικείμενο. Αν ο χρήστης αρνηθεί να δώσει το δικαίωμα στην εφαρμογή, ή αν το δώσει αλλά προκύψει κάποιο σφάλμα, θα εκτελεστεί το catch block του κώδικα.

```
  onTakePhoto() {
    if (!Capacitor.isPluginAvailable('Camera')) {
      this.filePickerRef.nativeElement.click();
      return;
```

```
    }
    Camera.getPhoto({
      quality: 90,
      source: this.platform.is('hybrid') ? CameraSource.Prompt :
CameraSource.Camera,
      correctOrientation: true,
      // height: 320,
      width: 300,
      resultType: CameraResultType.Uri
    })
    .then(image => {
      let imageBase64String;
      if (this.platform.is('hybrid')) {
        this.selectedImage = image.path;
        Filesystem.readFile({
          path: image.path
        }).then(value => {
          imageBase64String = value.data;
          this.imagePick.emit(imageBase64String);
          }
        );
      }
      else {
        this.selectedImage = image.webPath;
        fetch(image.webPath).then(response => {
          response.blob().then(blob => {
            this.convertBlobToBase64(blob).then(base64Str => {
              imageBase64String = base64Str;
              this.imagePick.emit(imageBase64String);
            });
          });
        });
      }
    })
    .catch(error => {
        console.log(error);
        if (this.usePicker) {
          this.filePickerRef.nativeElement.click();
        }
        return false;
      });
  }
```

Το παραπάνω απόσπασμα κάνει χρήση του Capacitor Camera plugin για να τραβήξει φωτογραφία από την κάμερα της συσκευής. Με την βοήθεια του Capacitor έχουμε γνώση της πλατφόρμας στην οποία τρέχει ο κώδικας κάθε φορά. Αν η πλατφόρμα έχει την τιμή “hybrid”

(δηλαδή η Ionic εφαρμογή τρέχει σε mobile συσκευή) τότε ο χρήστης θα ερωτηθεί αν θέλει να βγάλει νέα φωτογραφία ή να επιλέξει κάποια φωτογραφία από την συσκευή του. Αν η πλατφόρμα δεν είναι “hybrid” σημαίνει ότι η εφαρμογή τρέχει σε web περιβάλλον με την βοήθεια κάποιου browser. Στην περίπτωση αυτή θα προσπαθήσει να ανοίξει την κάμερα του υπολογιστή και να τραβήξει φωτογραφία. Αν δεν βρει κάμερα στον υπολογιστή ή ο χρήστης δεν δώσει την απαραίτητη πρόσβαση στον browser του για το άνοιγμα της κάμερας τότε θα εκτελεστεί ο κώδικας στο catch block όπου θα χρησιμοποιηθεί ο file picker για ανέβασμα φωτογραφίας από την συσκευή.

Κεφάλαιο 6.3: Λογική και τεχνικές για την εύρεση των διαθέσιμων PetWorkers

Στον πυρήνα της εφαρμογής αυτής βρίσκεται ο σχεδιασμός της λογικής σύμφωνα με την οποία ο χρήστης PetOwner θα μπορεί να βρει τους κατάλληλους PetWorkers για την κάλυψη των αναγκών της ζητούμενης υπηρεσίας.

Όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο, η αναζήτηση για τους κατάλληλους PetWorkers, ξεκινά όταν ο χρήστης επιλέξει τον τύπο της υπηρεσίας που τον ενδιαφέρει, πατώντας στο αντίστοιχο “tile” στην Σελίδα “Find Pet-Services”. Ο τρόπος με τον οποίο γίνεται αυτή η αναζήτηση στην εφαρμογή είναι ο εξής:

- Με το πάτημα ενός κουμπιού (tile) υπηρεσίας, στέλνεται ένα http request προς το webAPI, το οποίο εμπεριέχει την πληροφορία του τύπου της υπηρεσίας προς αναζήτηση.
- Ο αντίστοιχος controller λαμβάνει το request αυτό και κάνει μια αναζήτηση στην βάση δεδομένων, επιστρέφοντας τους PetWorkers των οποίων η περιοχή (Region) της διεύθυνσής τους ταυτίζεται με την περιοχή της διεύθυνσης του PetOwner που έκανε την αναζήτηση και η λίστα των υποστηριζόμενων υπηρεσιών που εξυπηρετούν εμπεριέχει τον τύπο της υπηρεσίας που αναζητήθηκε, ενώ ταυτόχρονα αποκλείει τον ίδιο τον χρήστη που έκανε την αναζήτηση σε περίπτωση που ο τελευταίος διαθέτει και προφίλ τύπου PetWorker . Αυτό γίνεται ώστε να μειωθεί ο όγκος της πληροφορίας που θα φορτωθεί στον μνήμη για περεταίρω έλεγχο και επεξεργασία.
- Στη συνέχεια γίνεται ένα φιλτράρισμα σε όλους τους PetWorkers που επιστράφηκαν από την βάση, ώστε να επιστραφούν στον χρήστη μόνο αυτοί των οποίων η απόσταση της διεύθυνσης τους από την διεύθυνση του PetOwner είναι μικρότερη ή ίση από την υποστηριζόμενη απόσταση (SupportedRadius) την οποία έχει δηλώσει ο κάθε PetWorker. Ο υπολογισμός αυτός της γεωγραφικής απόστασης γίνεται με την βοήθεια της βιβλιοθήκης GeoCoordinatePortable, η οποία για τον υπολογισμό της χρησιμοποιεί τις συντεταγμένες των δύο σημείων (latitude και longitude) ενώ λαμβάνει υπόψιν και την καμπυλότητα της γης.

Με τον τρόπο αυτό οι τελικοί PetWorkers που εμφανίζονται στον χρήστη είναι αυτοί των οποίων η διεύθυνση του PetOwner βρίσκεται εντός του νοητού κύκλου τους στον χάρτη, με κέντρο την διεύθυνσή τους και ακτίνα την υποστηριζόμενη απόσταση από αυτήν που έχουν δηλώσει ότι εξυπηρετούν.

Την παραπάνω λογική υλοποιεί η μέθοδος “GetEligiblePetServiceWorkers” η οποία βρίσκεται στο αρχείο PetServicesRepository.cs και την οποία καλεί ο PetServicesController. Ο κωδικάς της φαίνεται στο παρακάτω απόσπασμα:

```
public async Task<ICollection<PetServiceWorkerDto>>
GetEligiblePetServiceWorkers(int petOwnerId, Address petOwnerAddress, int
serviceCode)
        {
            var query = from addr in _context.Addresses
                        join ua in _context.UserAddresses on addr.Id equals
ua.AddressId
```

```
                        join pw in _context.PetWorkers on ua.UserId equals
pw.Id
                        where addr.Region == petOwnerAddress.Region && pw.Id
!= petOwnerId
                        select new
                        {
                            petWorker = pw,
                            address = addr
                        };

            var petWorkersWithAddress = await query.ToListAsync();

            var petServiceWorkersDto = new List<PetServiceWorkerDto>();
            foreach (var pwa in petWorkersWithAddress)
            {

                var supportedServices = new List<int>();
                var petWorkerServices = await
_context.PetWorkerServices.Where(pws => pws.PetWorkerId ==
pwa.petWorker.Id).ToListAsync();
                petWorkerServices.ForEach(pws =>
                {
                    supportedServices.Add(pws.PetServiceId);
                });

                if (!supportedServices.Contains(serviceCode))
                    continue;

                GeoCoordinate pin1 = new
GeoCoordinate(petOwnerAddress.Latitude, petOwnerAddress.Longitude);
                GeoCoordinate pin2 = new GeoCoordinate(pwa.address.Latitude,
pwa.address.Longitude);

                double distanceBetween = pin1.GetDistanceTo(pin2);
                if (distanceBetween <= pwa.petWorker.SupportedRadius)
                {
                    var petServiceWorkerDto = await _context.Users
                        .Where(user => user.Id == pwa.petWorker.Id)

.ProjectTo<PetServiceWorkerDto>(_mapper.ConfigurationProvider)
                        .SingleOrDefaultAsync();

                    petServiceWorkerDto.SupportedServices =
supportedServices;
                    petServiceWorkerDto.Distance = distanceBetween;

                    petServiceWorkersDto.Add(petServiceWorkerDto);
                }

            }

            petServiceWorkersDto = petServiceWorkersDto.OrderBy(p =>
p.Distance).ToList();

            return petServiceWorkersDto;
        }
```

Κεφάλαιο 6.4: Υλοποίηση της λειτουργίας ανταλλαγής μηνυμάτων

Ένα άλλο σημείο της εφαρμογής που αξίζει να αναφερθεί εδώ, είναι η λειτουργία ανταλλαγής μηνυμάτων. Η λειτουργικότητα αυτή αναπτύχθηκε με την βοήθεια της βιβλιοθήκης SignalR, η οποία μεταξύ άλλων υλοποιεί το WebSocket πρωτόκολλο.

Το WebSocket είναι ένα πρωτόκολλο το οποίο επιτρέπει την επικοινωνία μεταξύ ενός client και ενός server μέσω μίας μοναδικής συνεχούς σύνδεσης. Αυτό το καθιστά ιδανικό για εφαρμογές που απαιτούν συνεχή, real-time επικοινωνία, όπως εφαρμογές chat. Ένα από τα βασικά πλεονεκτήματα της χρήσης των WebSockets για την λειτουργία chat είναι ότι επιτρέπουν την λήψη μηνυμάτων με χαμηλή καθυστέρηση. Με τα παραδοσιακά http requests, πρέπει να οριστεί μία νέα σύνδεση για κάθε κύκλο αίτησης-απάντησης ο οποίος μπορεί να οδηγήσει σε μεγάλες καθυστερήσεις. Τα WebSockets από την άλλη, επιτρέπουν μια αμφίδρομη επικοινωνία δύο χρηστών μέσω μιας κοινής σύνδεσής τους δια μέσω του server η οποία τους επιτρέπει να «ακούνε» σε events, και κάτι τέτοιο προσφέρει πολύ χαμηλές καθυστερήσεις και ένα ποιοτικό User Experience (UX) του στον χρήστη.

Για την χρήση της SignalR βιβλιοθήκης χρειάζεται η δημιουργία των λεγόμενων “hubs” στον server, στο οποίο hub μπορούν να συνδεθούν οι client εφαρμογές. Οι εφαρμογές αυτές μπορούν να καλέσουν μεθόδους στο hub του server ή να “ακούσουν” για μηνύματα από αυτόν χρησιμοποιώντας τις αντίστοιχες client βιβλιοθήκες του SignalR. Η υλοποίηση ενός τέτοιου hub μπορεί να βρεθεί στο αρχείο MessageHub.cs της εφαρμογής.

Κεφάλαιο 7: Προβληματικές και Αντιμετώπιση

Κατά την ανάπτυξη της εφαρμογής της παρούσας εργασίας αρχικά είχε επιλεχθεί η χρήση της React Native τεχνολογίας, ως framework για τις δύο client εφαρμογές. Η React Native είναι κι αυτή μια cross-platform τεχνολογία, όπου με ένα κοινό code base γραμμένο σε JavaScript γλώσσα, μπορούν να παραχθούν δύο εκδόσεις της εφαρμογής, για ios και για android. Στην περίπτωση όμως αυτή η client εφαρμογή για το web θα έπρεπε να γραφτεί από την αρχή με έναν τελείως διαφορετικό τρόπο.

Για τον λόγο αυτό, και παρότι ένα μεγάλο μέρος των δύο client εφαρμογών είχε αναπτυχθεί, εν τέλει επιλέχθηκε το Ionic/Angular framework, στο οποίο και ξαναγράφτηκαν οι εφαρμογές, και στο οποίο όπως έχει προαναφερθεί είναι εφικτή η λειτουργία και σε web περιβάλλον.

Κεφάλαιο 8: Συμπεράσματα και Μελλοντικές Επεκτάσεις

Η εφαρμογή WalkieDoggie που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας είναι μια μακροσκελής εφαρμογή που απαντά στις σημερινές ανάγκες της αγοράς. Παρότι δεν πρόκειται για ένα production-ready προϊόν ακόμη, θα μπορούσε να αποτελέσει τη βάση για κάτι τέτοιο, προσφέροντας μία πολύ καλή πρώτη ύλη.

Υπάρχει ασφαλώς περιθώριο για μελλοντικές βελτιώσεις και επεκτάσεις που θα έκαναν το προϊόν ακόμη πιο ελκυστικό, λειτουργικό και οικονομικά βιώσιμο και ανταγωνιστικό. Παρακάτω δίνονται ενδεικτικά μερικές από τις πιθανές επεκτάσεις που θα μπορούσαν να αναπτυχθούν στο μέλλον:

- Θα μπορούσε να προστεθεί ένα ημερολόγιο ραντεβού των PetWorkers. Το ημερολόγιο αυτό θα είναι προσβάσιμο από τους PetOwners, οι οποίοι θα μπορούν να το ανασκοπήσουν ώστε να αναζητήσουν ελεύθερα slots, και να κάνουν αίτημα ώστε να κλείσουν κάποιο μοναδικό ή επαναλαμβανόμενο ραντεβού. Σε αυτό το αίτημα θα συμπεριλαμβάνεται η πληροφορία της φόρμας προφίλ του κατοικίδιου.
- Ο PetWorker θα μπορεί ναι ανασκοπήσει τις αιτήσεις για ραντεβού, να ανασκοπεί το προφίλ των κατοικίδιων και να έχει τη δυνατότητα να αποδεχθεί ή να απορρίψει κάποιο

αίτημα. Σε αυτή τη διαδικασία, η λειτουργία των συνομιλιών θα επιδρά συνεπικουρικά. Σημειώνεται ότι η βάση δεδομένων έχει την κατάλληλη υποδομή για αυτή τη λειτουργία.

- Θα μπορούσαν να προστεθούν μηνύματα ειδοποίησης (push notifications) για τα αιτήματα ραντεβού, τις αποδοχές ή τις απορρίψεις των ραντεβού, τις συνομιλίες κλπ.
- Θα μπορούσε να προστεθεί ένδειξη που θα δείχνει αν κάποιος χρήστης είναι ενεργός ή ανενεργός.
- Θα μπορούσε να προστεθεί χρέωση για τους επαγγελματίες χρήστες της εφαρμογής PetWorker. Το γεγονός ότι οι δύο εφαρμογές έχουν αναπτυχθεί ανεξάρτητα, βοηθά σε αυτή την προοπτική.

Παράρτημα 1: Βιντεοπαρουσίαση

Στον παρακάτω σύνδεσμο μπορείτε να βρείτε μία βιντεοπαρουσίαση της εφαρμογής:

https://clipchamp.com/watch/GaRohTRNtUP

Βιβλιογραφία

Alicea, A. (2022). *JavaScript: Understanding the Weird Parts.* Ανάκτηση από Udemy: https://www.udemy.com/course/understand-javascript/

Allen, S. (2013). *C# Generics.* Ανάκτηση από Pluralsite: https://www.pluralsight.com/courses/csharp-generics

Allen, S. (2016). *LINQ Fundamentals.* Ανάκτηση από Pluralsite: https://www.pluralsight.com/courses/linq-fundamentals-csharp-6

Allen, S. (2019). *ASP.NET Core Fundamentals.* Ανάκτηση από Pluralsite: https://www.pluralsight.com/courses/aspnet-core-fundamentals

Allen, S. (2019). *C# Fundamentals.* Ανάκτηση από Pluralsite: https://www.pluralsight.com/courses/csharp-fundamentals-dev

Angular Team. (2022). *Angular documentation.* Ανάκτηση από Angular: https://angular.io/docs

Cummings, N. (2022). *Build an app with ASPNET Core and Angular from scratch.* Ανάκτηση από Udemy: https://www.udemy.com/course/build-an-app-with-aspnet-core-and-angular-from-scratch/

Gamma, E., Helm, R., Johnson, R., Vlissides, J., & Booch, G. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software.* Reading, Mass.: Addison-Wesley.

Ionic Framework. (2022). *Ionic Framework documentation.* Ανάκτηση από Ionic DOCS: https://ionicframework.com/docs/

Ionic Team. (2022). *Capacitor documentation.* Ανάκτηση από Capacitor Docs: https://capacitorjs.com/docs

Martin, R. C. (2009). *Clean Code: A Handbook of Agile Software Craftsmanship.* Upper Saddle River: Prentice Hall.

Martin, R. C. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design.* Boston: Prentice Hall.

Microsoft. (2022). *.NET documentation.* Ανάκτηση από Microsoft: https://learn.microsoft.com/en-us/dotnet/fundamentals/

Microsoft. (2022). *Entity Framework documentation.* Ανάκτηση από Microsoft: https://learn.microsoft.com/en-us/ef/

Microsoft. (2022). *SignalR documentation.* Ανάκτηση από Microsoft: https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction?WT.mc_id=dotnet-35129-website&view=aspnetcore-7.0

PostgreSQL Global Development Group. (2022). *PostgreSQL documentation.* Ανάκτηση από PostgreSQL: https://www.postgresql.org/docs/

Schwarzmuller, M. (2022). *Ionic - Build iOS, Android & Web Apps with Ionic & Angular.* Ανάκτηση από Udemy: https://www.udemy.com/course/ionic-2-the-practical-guide-to-building-ios-android-apps/

Schwarzmuller, M. (2023). *Angular - The Complete Guide.* Ανάκτηση από Udemy: https://www.udemy.com/course/the-complete-guide-to-angular-2/

Schwarzmuller, M. (2023). *React Native - The Practical Guide.* Ανάκτηση από Udemy: https://www.udemy.com/course/react-native-the-practical-guide/

Steele, C. (2022). *The Ultimate MySQL Bootcamp: Go from SQL Beginner to Expert.* Ανάκτηση από Udemy: https://www.udemy.com/course/the-ultimate-mysql-bootcamp-go-from-sql-beginner-to-expert/

Stern, D. (2022). *Introduction to TypeScript.* Ανάκτηση από Udemy: https://www.udemy.com/course/typescript/

Turton, L. (2022). *Javascript Essentials.* Ανάκτηση από Udemy: https://www.udemy.com/course/javascript-essentials/