Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη Web Εφαρμογής με Χρήση Angular και ASP.NET Core** **Web Application Development using Angular and ASP.NET Core**
Ονοματεπώνυμο Φοιτητή	**Γεράσιμος Κρικέτος**
Πατρώνυμο	**Γεώργιος**
Αριθμός Μητρώου	**ΜΠΠΛ/ 17025**
Επιβλέπων	**Ευθύμιος Αλέπης, Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης **Ιανουάριος 2021**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ευθύμιος Αλέπης Αναπληρωτής Καθηγητής	Μαρία Βίρβου Καθηγήτρια	Κωνσταντίνος Πατσάκης Αναπληρωτής Καθηγητής

Περίληψη

Η εν λόγω εφαρμογή δημιουργήθηκε με σκοπό να δώσει μια λύση στο πρόβλημα που αντιμετωπίζουμε σχετικά με τα αδέσποτα ζώα και κατά κύριο λόγο τα σκυλιά, τα οποία αφορούν το μεγαλύτερο ποσοστό αυτών. Μέσα από μία σχετικά εύκολη διαδικασία, ο χρήστης της εφαρμογής μπορεί να συνεισφέρει στο να ελαττωθεί αυτή η δυσάρεστη κατάσταση. Μπορεί να εκτελέσει ενέργειες όπως: να κάνει upload ένα κατοικίδιο προς υιοθέτηση, να κοινοποιήσει ένα χαμένο σκυλί, να ζητήσει να υιοθετήσει κάποιο σκυλί της επιλογής του, πάντα αντικρίζοντας ένα ευχάριστο και εύχρηστο διαδικτυακό περιβάλλον, κύρια συστατικά του οποίου είναι η καλή διεπαφή και η εμπειρία του χρήστη (UI/UX). Πρόκειται για μία δυναμική, διαδικτυακή, Full-Stack εφαρμογή η οποία αναπτύχθηκε χρησιμοποιώντας κάποιες από τις τελευταίες τεχνολογίες ανάπτυξης Web εφαρμογών. Όσον αφορά την front-end λειτουργικότητα, επιλέχθηκε η Angular, ενώ ως Server-Side τεχνολογία προτιμήθηκε το ASP.NET Core Framework της Microsoft με χρήση της γλώσσας προγραμματισμού C#. Τέλος, για την επικοινωνία με τη βάση δεδομένων και την αποθήκευση διάφορων τύπων δεδομένων σε αυτήν, επιλέχθηκε ως RDBMS(Relational Database Management System) ο SQL Server της Microsoft.

Abstract

This application was created with the sole purpose of solving the stray animal problem, specifically with dogs, because they occupy the largest amount of the overall percentage. Through a relatively easy process, the user of the application can contribute to making this unpleasant situation better. The user can perform multiple actions, including post a pet for adoption, post a missing dog notice, request to adopt a dog, and will always experience a pleasant, user-friendly web environment, that values high-quality user interface and user experience (UI/UX). This is a dynamic, full-stack web application which was developed using some of the latest technologies in web development. The tool used for front-end functionality was Angular while for server-side development, the selected tool was Microsoft's ASP.NET Core Framework with C# programming language utilization. Finally, for database communications and for proper data storage, Microsoft SQL Server was used as the RDBMS.

Εισαγωγή

Τα αδέσποτα ζώα αποτελούν αναμφίβολα ένα ευαίσθητο θέμα της εποχής μας. Πολλοί από εμάς, μην έχοντας αρχικά κατά νου την δέσμευση, την υποχρέωση και την αναγκαιότητα της εναπόθεσης προσωπικού χρόνου που φέρνει μαζί του ένα κατοικίδιο, παίρνουμε την απόφαση είτε να αγοράσουμε είτε να υιοθετήσουμε ένα από αυτά.

Όταν όμως διαπιστώνουμε πως δεν μπορούμε να ανταπεξέλθουμε στις προαναφερθείσες προσδοκίες, καλούμαστε να πάρουμε άσχημα μέτρα, που επηρεάζουν αρνητικά τόσο το κατοικίδιό μας, όσο και εμάς τους ίδιους. Αποτέλεσμα των μέτρων αυτών είναι πολλά κατοικίδια να καταλήγουν χωρίς μόνιμη στέγη, είτε υπό τη φροντίδα φιλοζωικών οργανώσεων, είτε απευθείας στον δρόμο. Πώς όμως μπορούμε να αποφύγουμε αυτά τα αποτελέσματα;

Αποτελεί κρυφή αλήθεια το γεγονός ότι, οι περισσότεροι από εμάς, αν και εφόσον καταλήξουμε στην απόφαση να πάρουμε ένα σκυλί για κατοικίδιο θα πάρουμε λίγο χρόνο να καθίσουμε ήρεμα και να αποφανθούμε ποια ράτσα σκύλου μας αρέσει ώστε αμέσως μετά να απευθυνθούμε στον κατάλληλο άνθρωπο και τελικά να προβούμε σε μία αγορά μην έχοντας καν στο μυαλό μας ότι πολλά σκυλιά είναι ήδη εκεί.

Λίγοι είναι εκείνοι οι οποίοι θα δουν ένα αδέσποτο σκυλί και θα σκεφτούν να το πάρουν σπίτι τους, να το φροντίσουν, να το περιθάλψουν και να είναι πλέον για εκείνους το κατοικίδιό τους. Αυτό έχει σαν αποτέλεσμα να μην μειώνεται ο αριθμός των αδέσποτων σκυλιών στους δρόμους, και σε ακραίες περιπτώσεις να προστίθενται και άλλα.

Όσο μεγαλώνει ο αριθμός ένα μεγαλύτερο πρόβλημα αναδύεται. Μην έχοντας τον έλεγχο των ζώων αυτών, καθώς και τους πόρους ώστε όλα να λάβουν την απαραίτητη υγειονομική περίθαλψη, οδηγούνται αναπόφευκτα σε καταστάσεις όπως: έκθεση σε κακοποίηση από ασυνείδητους καθώς και προβλήματα υγείας τα οποία έχουν άμεσο αντίκρισμα στη δημόσια υγεία. Σύμφωνα με τα παραπάνω μερικές φορές η αποτρόπαια λύση της ευθανασίας αποτελεί μονόδρομο. Αυτό είναι αναγκαίο να αλλάξει. Πώς μπορούμε όμως να ωθήσουμε τους ανθρώπους ώστε να τοποθετήσουν πιο ψηλά στη λίστα των επιλογών τους την υιοθέτηση ενός αδέσποτου ζώου;

Σκοπός της εφαρμογής είναι να προσφέρει στον χρήστη όλα τα διαθέσιμα προς υιοθεσία ζώα είτε αυτά είναι αδέσποτα είτε προέρχονται από μία οικογένεια που αδυνατεί να τα φροντίσει. Να έρθει σε μια πρώτη επαφή μαζί τους, να μάθει παραπάνω πληροφορίες για αυτά και, αν και εφόσον το θελήσει, να δώσει σε ένα την ευκαιρία να γίνει μέλος της συντροφιάς του. Επιπλέον, όπως προαναφέρθηκε, δίνεται η δυνατότητα σε κάποιον χρήστη να ανεβάσει όλες τις πληροφορίες σχετικά με το κατοικίδιό του το οποίο προορίζει για υιοθεσία ώστε να γίνει και αυτό διαθέσιμο στο κοινό.

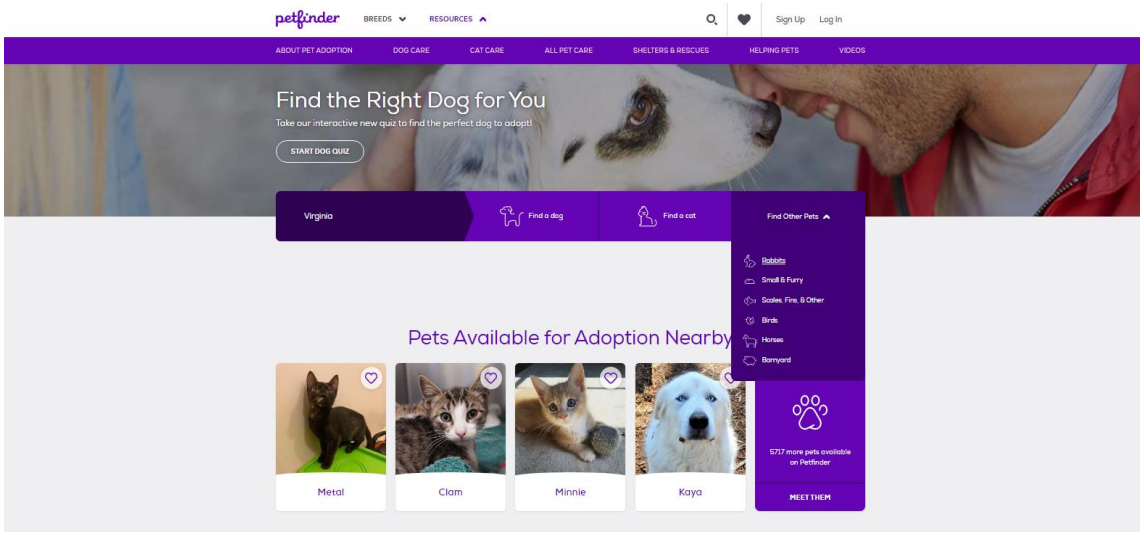
Επιπροσθέτως, πολλές είναι οι φορές που κάτοχοι κατοικίδιων έρχονται στη δυσάρεστη θέση όπου έχουν χάσει προσωρινά το κατοικίδιό τους, μπαίνοντας αμέσως μετά στην επίπονη διαδικασία να ενημερώνουν όσους περισσότερους μπορούν σχετικά με το συμβάν, να προβαίνουν στην διανομή και ανάρτηση, σε όσο το δυνατόν μεγαλύτερη εμβέλεια, φυλλαδίων με περιεχόμενο τη φωτογραφία και τα στοιχεία του σκύλου. Προκειμένου να έχουμε το καλύτερο δυνατό αποτέλεσμα πρέπει τόσο το κοινό στο οποίο θα απευθυνθούμε να είναι αρκετά μεγάλο, όσο και να φτάσει σε αυτό η πληροφορία όσο γίνεται πιο άμεσα.

Χρησιμοποιώντας τη δύναμη του διαδικτύου η διαδικασία αυτή μπορεί να γίνει πιο εύκολη και ταυτόχρονα πιο αποτελεσματική. Μέσω της εφαρμογής μπορεί να ικανοποιηθεί η συγκεκριμένη ανάγκη. Πιο συγκεκριμένα μπορεί πολύ εύκολα ο οποιοσδήποτε χρήστης να αναρτήσει το χαμένο κατοικίδιό του, μαζί με τα απαραίτητα στοιχεία, βάζοντας επιπρόσθετα ένα στίγμα στον χάρτη ώστε να γίνει γνωστή η τοποθεσία στην οποία χάθηκε. Με τον τρόπο αυτόν οι υπόλοιποι χρήστες μπορούν ανά πάσα στιγμή να λάβουν γνώση του γεγονότος ότι στην περιοχή τους έχει χαθεί ένα κατοικίδιο. Επιπλέον, όλοι έχουν πρόσβαση σε έναν ενιαίο χάρτη ο οποίος απεικονίζει τις τοποθεσίες των χαμένων κατοικίδιων.

Ανασκόπηση Πεδίου

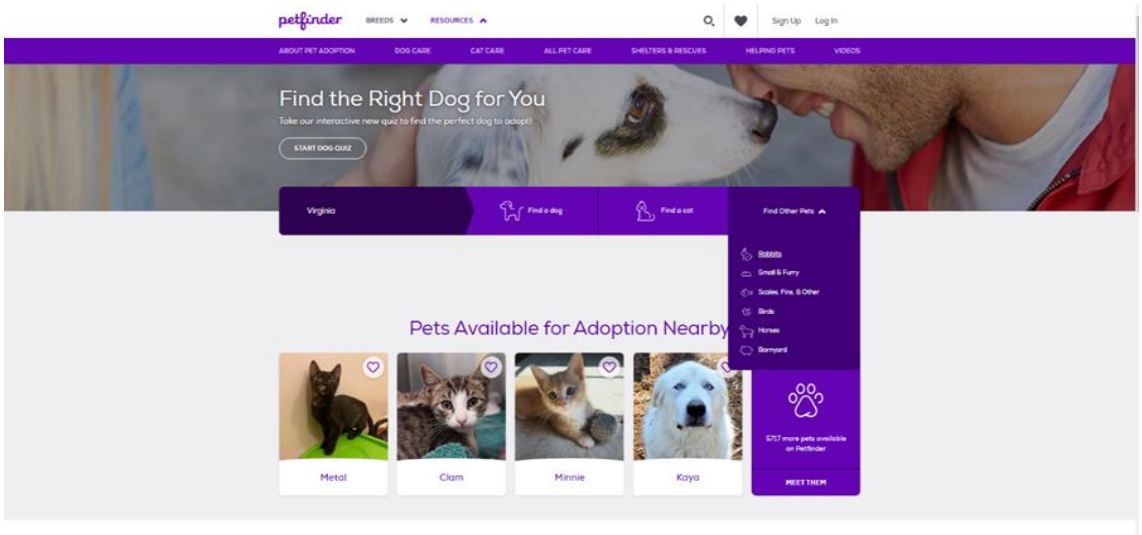
Εδώ θα γίνει μία παρουσίαση αντίστοιχης εφαρμογής η οποία αναπτύχθηκε με σκοπό να χρησιμοποιήσει την τεχνολογία του διαδικτύου και τις δυνατότητες που αυτό προσφέρει ώστε:

- Να κινήσει το ενδιαφέρον του κοινού προς την κατεύθυνση της υιοθεσίας κατοικίδιων.
- Να αυξήσει την αποτελεσματικότητα της υιοθεσίας και της διαδικασίας από την οποία αυτή διέπεται με σκοπό την εξάλειψη της ευθανασίας των αδέσποτων ζώων.
- Να δώσει αξία στην έννοια «κατοικίδιο» και να την ανάγει σε ένα ακόμα μέλος της οικογένειας του καθενός από εμάς.

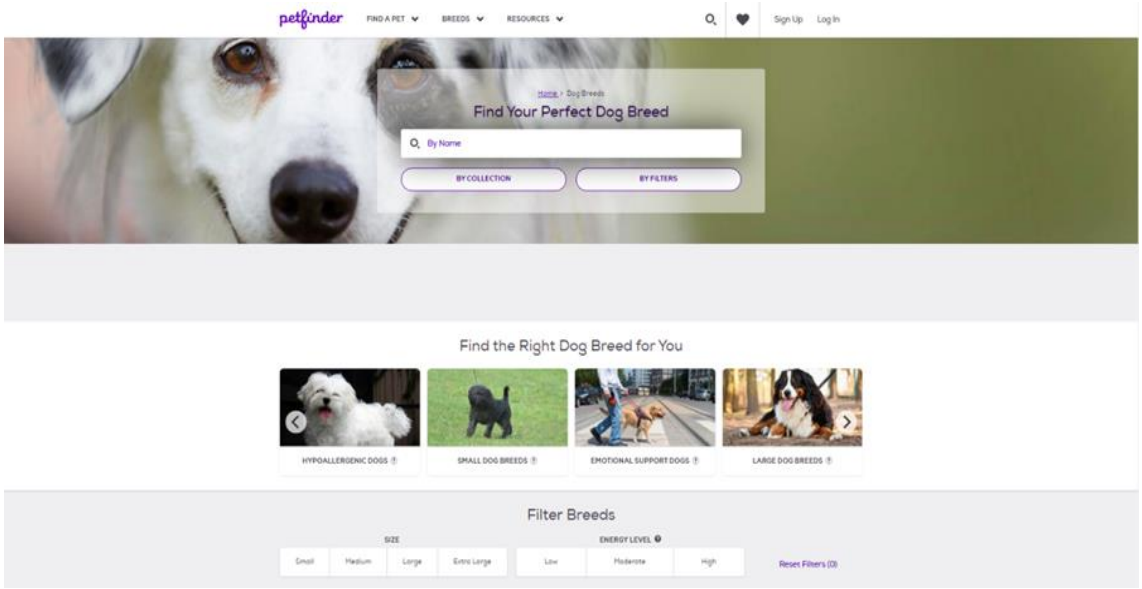


Η συγκεκριμένη εφαρμογή προσφέρει αρκετές δυνατότητες, μερικές από τις οποίες θα αναφερθούν παρακάτω.

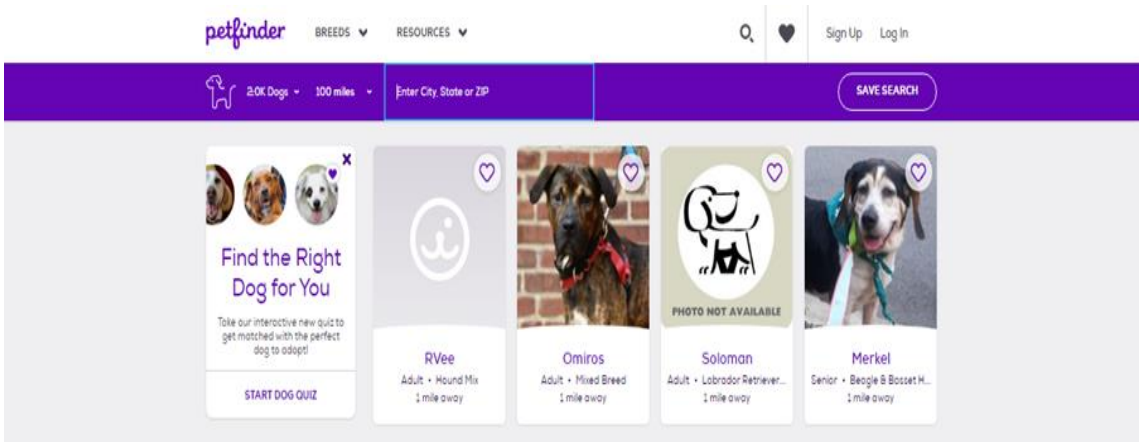
- Ο χρήστης δύναται να αποφασίσει τι είδους κατοικίδιο θέλει να υιοθετήσει μέσα από μια πληθώρα επιλογών η οποία περιέχει, όχι μόνο σκυλιά, αλλά και γάτες, κουνέλια, ψάρια, πουλιά και άλογα μεταξύ άλλων.



- Διαθέτει μία ποικιλία από ράτσες για όλα τα διαθέσιμα κατοικίδια για να διαλέξεις εκείνη που σου ταιριάζει καλύτερα.

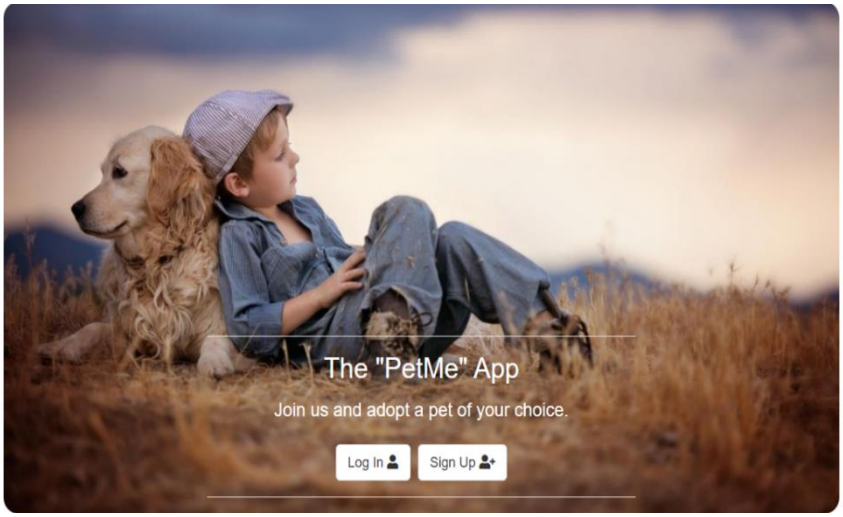


- Επίσης μέσω διάφορων φίλτρων μπορείς να βρεις διαθέσιμα κατοικίδια κοντά σου, δίνοντας πληροφορίες όπως είναι η πόλη, η πολιτεία καθώς και ο ταχυδρομικός κώδικας.

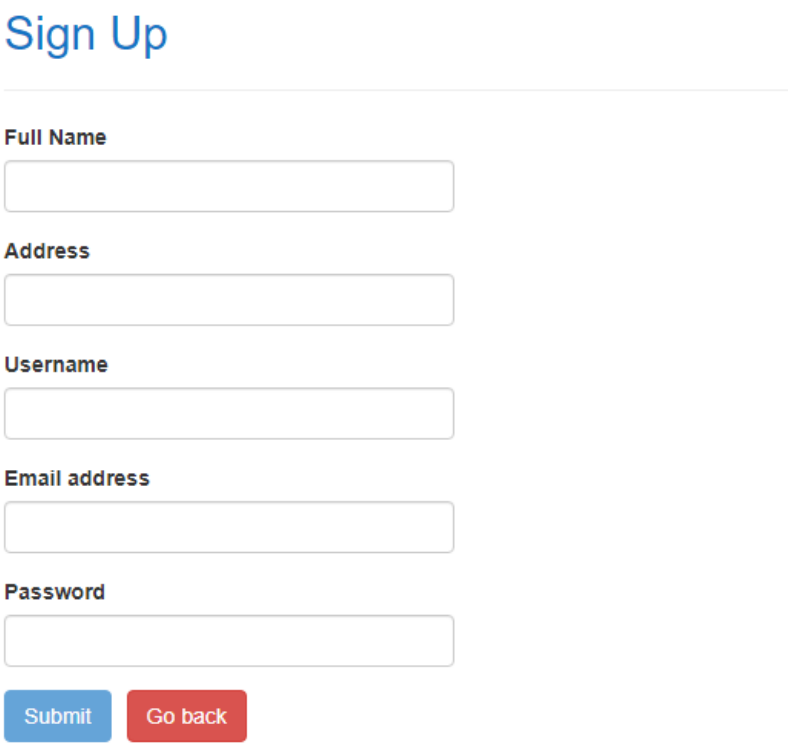


Παρουσίαση και Χρήση της Εφαρμογής

Προκειμένου κάποιος να κάνει χρήση της εφαρμογής θα πρέπει πρώτα να δημιουργήσει λογαριασμό σε αυτήν, μέσω μια διαδικασίας εγγραφής. Στην οθόνη υποδοχής μπορούμε να επιλέξουμε είτε να κάνουμε εγγραφή ως νέος χρήστης είτε να μεταβούμε στην φόρμα εισαγωγής των αναγνωριστικών μας, αν και εφόσον διαθέτουμε ήδη λογαριασμό.



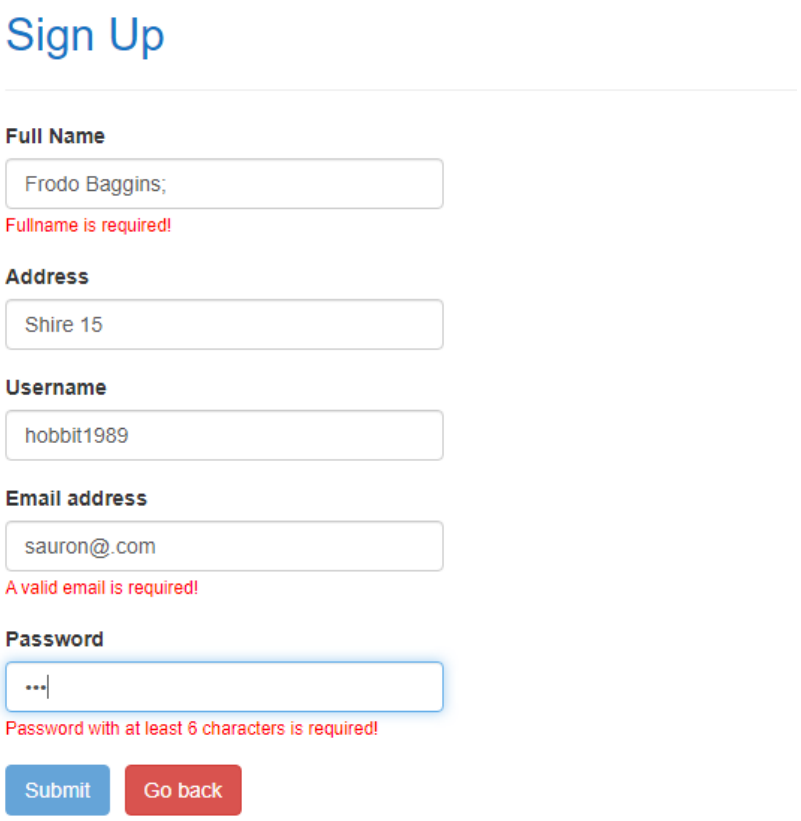
Στην πρώτη περίπτωση ο χρήστης καλείται να εισάγει μερικά στοιχεία για εκείνον όπως το ονοματεπώνυμο, την διεύθυνση, ένα αναγνωριστικό όνομα χρήστη, ένα email καθώς και έναν κωδικό πρόσβασης.



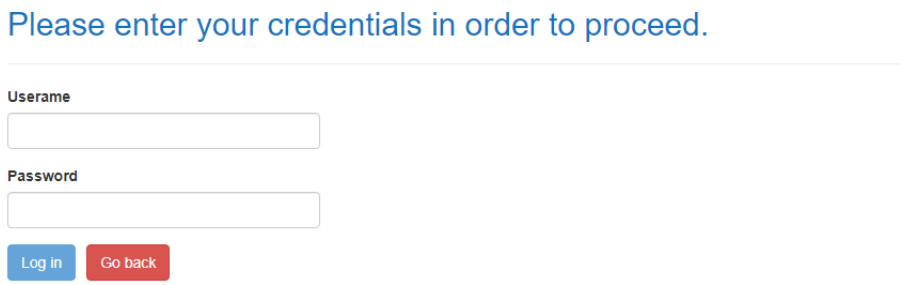
Η συγκεκριμένη φόρμα είναι εφοδιασμένη με ελέγχους ορθότητας όσον αφορά την εισαγωγή των δεδομένων σε αυτήν. Οι έλεγχοι αυτοί καλύπτουν τα παρακάτω σενάρια:

- Ο χρήστης δεν μπορεί να εισάγει ειδικούς χαρακτήρες (π.χ. * & ; #) στο ονοματεπώνυμο καθώς και στη διεύθυνση. Παράλληλα επιτρέπει εισαγωγή αριθμού στη διεύθυνση αλλά όχι στο ονοματεπώνυμο.
- Στην περίπτωση του email πρέπει ο χρήστης να εισάγει μια επιτρεπτή μορφή ηλεκτρονικού ταχυδρομείου, αλλιώς η φόρμα δεν θα είναι έγκυρη.
- Ο κωδικός πρέπει να αποτελείται από τουλάχιστον 6 χαρακτήρες, με ελεύθερη επιλογή αυτών από τον χρήστη.

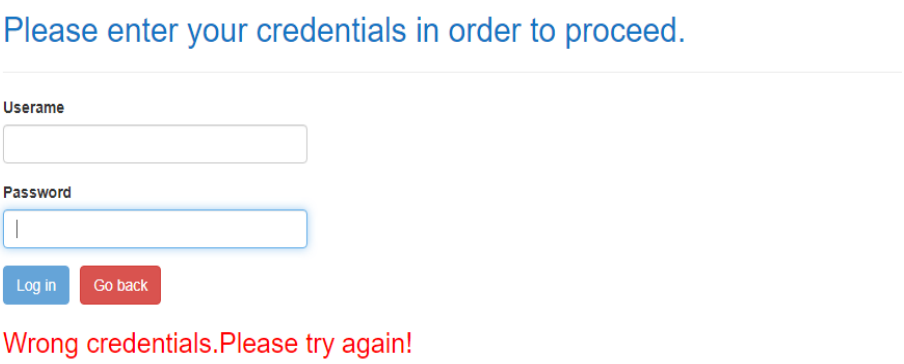
Αξίζει να σημειωθεί πως κάθε φορά που ο χρήστης εισάγει λάθος δεδομένα στα διαθέσιμα πεδία, ένα ή περισσότερα μηνύματα εμφανίζονται στην οθόνη με σκοπό να τον οδηγήσουν στην σωστή καταχώρηση προκειμένου να γίνει επιτυχώς η εγγραφή του. Μόνο όταν η φόρμα αναγνωρίσει ως σωστή την τιμή του κάθε πεδίου δίνει τη δυνατότητα στον χρήστη να την υποβάλλει πατώντας το σχετικό κουμπί το οποίο πλέον έχει ενεργοποιηθεί.



Από τη στιγμή που ο χρήστης ολοκληρώσει με επιτυχία την εγγραφή μπορεί πλέον να μεταβεί στην οθόνη εισαγωγής των αναγνωριστικών. Εκεί θα χρειαστεί να εισάγει το username και το password του, ώστε αν αυτά είναι σε συμφωνία με εκείνα που καταχώρησε κατά την εγγραφή του, να εισαχθεί επιτυχώς στην εφαρμογή.



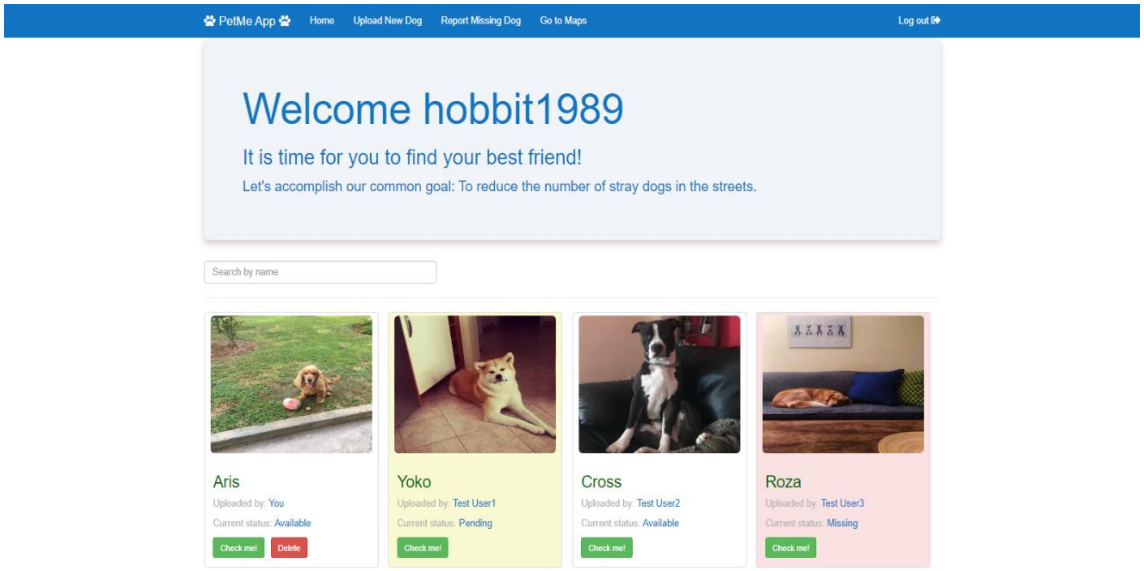
Ομοίως και εδώ τα δεδομένα που εισάγονται στα πεδία ελέγχονται για την ορθότητά τους εμφανίζοντας αντίστοιχα μηνύματα αν οι τιμές δεν συμφωνούν με τους κανόνες που έχουν οριστεί. Τέλος, σε περίπτωση που ο χρήστης έχει εισάγει στα πεδία τιμές οι οποίες συμφωνούν στους κανόνες, αλλά δεν συμφωνούν με τις τιμές που δήλωσε κατά την εγγραφή, υποδεικνύεται σε αυτόν ο λόγος αποτυχίας με αντίστοιχο μήνυμα.



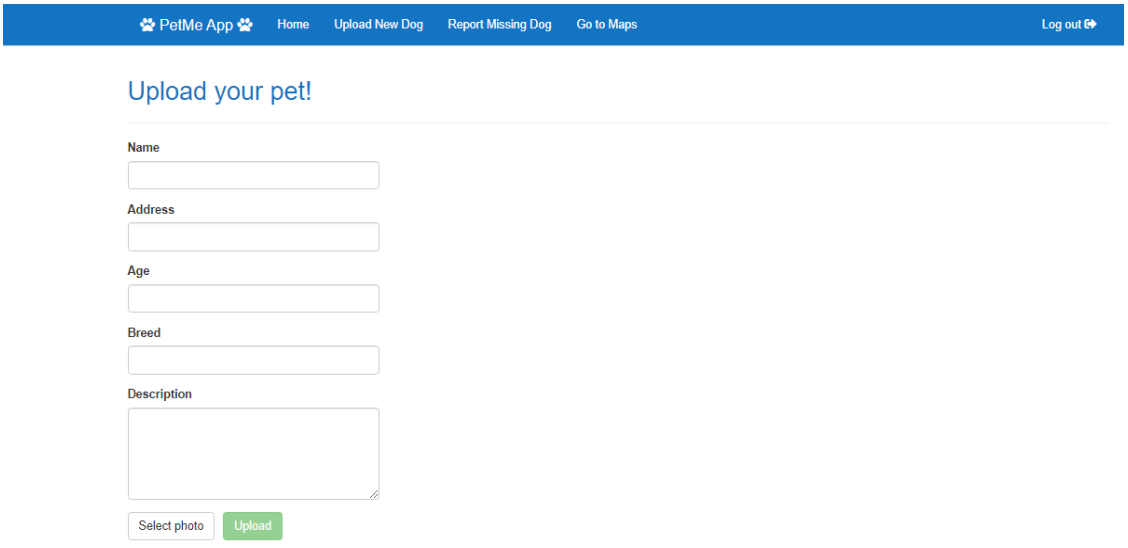
Αφού ο χρήστης έχει εισέλθει επιτυχώς στην εφαρμογή συναντά την αρχική σελίδα όπου και μπορεί να κάνει χρήση της λειτουργικότητας που προσφέρει.

Η λειτουργικότητα αυτή αφορά τα παρακάτω σενάρια:

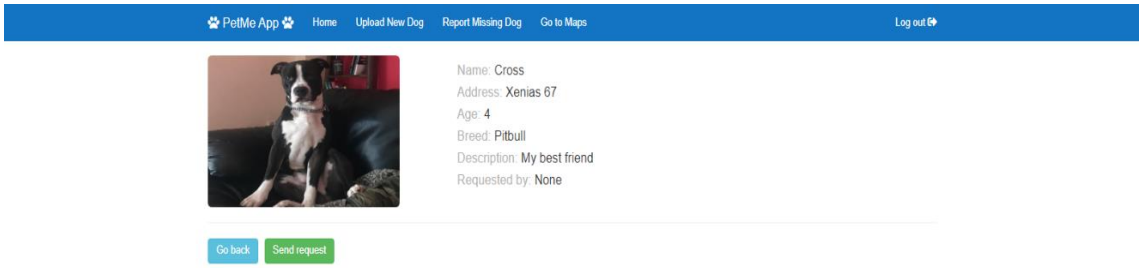
- Μπορεί ο χρήστης να δει μία λίστα από τα κατοικίδια που έχουν ήδη αναρτηθεί στην εφαρμογή.



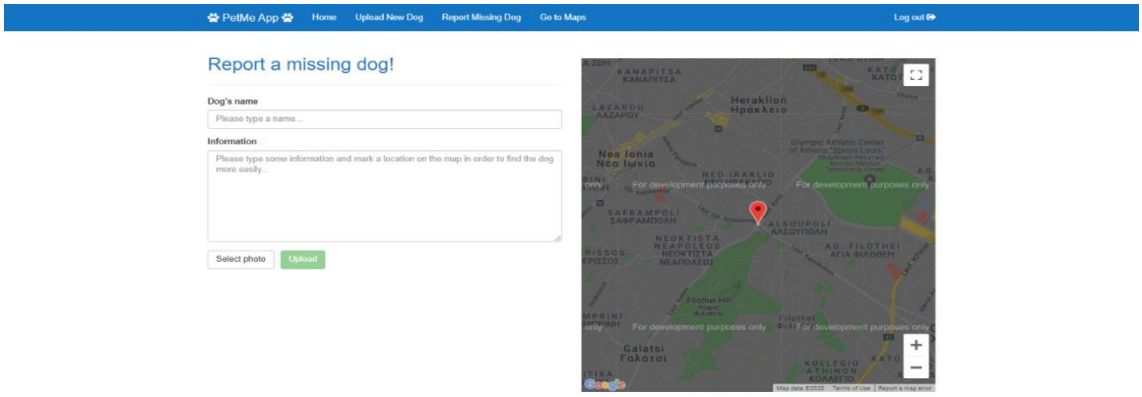
- Μπορεί επίσης να αναρτήσει και ο ίδιος ένα σκυλί, είτε αυτό είναι αδέσποτο είτε όχι, μέσω της επιλογής «Upload New Dog», προκειμένου να γίνει διαθέσιμο για υιοθεσία.



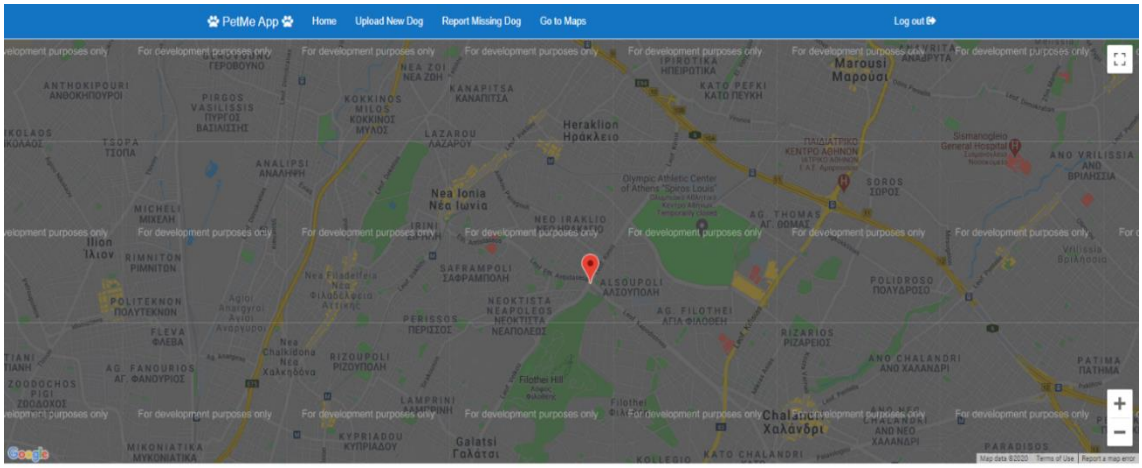
- Δίνεται σε όλους η δυνατότητα να μάθουν περισσότερες πληροφορίες για κάποιο από τα κατοικίδια της λίστας, πατώντας το κουμπί «Check Me!», ενώ αν κάποιος θέλει να αφαιρέσει μία δικιά του ανάρτηση μπορεί με το πάτημα του κουμπιού «Delete» να το καταφέρει. Στην πρώτη περίπτωση ο χρήστης δύναται να εκφράσει και το ενδιαφέρον του στέλνοντας αίτημα στον κάτοχο.



- Σε περίπτωση που θέλει να αναφέρει ότι το δικό του σκυλί έχει χαθεί, μέσω της επιλογής «Report Missing Dog» μπορεί να το επιτύχει εύκολα και γρήγορα.



- Με την επιλογή «Go to Maps» μπορεί να μεταβεί σε έναν χάρτη όπου απεικονίζονται όλα τα σημεία στα οποία διάφοροι χρήστες έχασαν το δικό τους σκυλί.



Αρχιτεκτονική Συστήματος

Εδώ θα γίνει μια προσπάθεια να αναλύσουμε τα κύρια δομικά στοιχεία της εφαρμογής και πώς αυτά λειτουργούν συμπληρωματικά το ένα στο άλλο με στόχο να επιτευχθεί η συνολική λειτουργικότητα.

Αρχιτεκτονική Πελάτη-Εξυπηρετητή

Όπως κάθε full-stack εφαρμογή έτσι και εδώ υιοθετήθηκε η αρχιτεκτονική πελάτη-εξυπηρετητή. (client-server architecture).

Κατά την αρχιτεκτονική αυτή υπάρχει ο πελάτης, ή οι πελάτες, οι οποίοι επικοινωνούν με τον εξυπηρετητή μέσω του διαδικτύου. Η επικοινωνία αυτή γίνεται προκειμένου ο πελάτης να ζητήσει τα δεδομένα που χρειάζεται και τα οποία φιλοξενεί ο εξυπηρετητής.

Ως πελάτης μπορεί να θεωρηθεί οποιοδήποτε λογισμικό είναι σε θέση να ζητήσει (request) υπηρεσίες ή δεδομένα μέσω του διαδικτύου. Ένα πρόγραμμα περιήγησης (web browser) μπορεί να χαρακτηριστεί ως πελάτης, διότι μέσω του διαδικτύου μπορεί να στείλει αιτήματα σε διάφορους εξυπηρετητές. Αντίστοιχα ο εξυπηρετητής αφορά το λογισμικό εκείνο που είναι σε θέση να λάβει τα αιτήματα των πελατών και, μετά από διεργασία που εκτελείται εσωτερικά, να αποστείλει πίσω στον πελάτη αυτό που του ζητήθηκε (response).

Προκειμένου όμως να επιτευχθεί η σωστή επικοινωνία μεταξύ τους θα πρέπει οι δύο πλευρές να τηρήσουν ένα σύνολο κανόνων σύμφωνα με το οποίο θα γίνεται η ανταλλαγή των μηνυμάτων μέσω του διαδικτύου. Το σύνολο κανόνων αυτό ονομάζεται HTTP (Hypertext Transfer Protocol) και είναι το πρωτόκολλο πάνω στα θεμέλια του οποίου γίνεται οποιαδήποτε ανταλλαγή δεδομένων μέσω διαδικτύου.

Πρωτόκολλο HTTP

Το Συγκεκριμένο πρωτόκολλο ανήκει στο επίπεδο εφαρμογής του OSI Model και είναι υπεύθυνο για την ανταλλαγή μηνυμάτων μεταξύ ενός client και ενός server. Στις περισσότερες των περιπτώσεων είναι το πρωτόκολλο το οποίο χρησιμοποιούν οι web-browsers όταν θέλουν να επικοινωνήσουν με ένα website. Υπάρχουν δύο είδη HTTP μηνυμάτων, τα αιτήματα (requests) και οι απαντήσεις (responses), με το κάθε ένα να έχει τη δικιά του δομή και μορφοποίηση.

Για να καταλάβουμε καλύτερα πως χρησιμοποιείται το εν λόγω πρωτόκολλο, ας δούμε πρώτα τι συμβαίνει όταν εμείς οι ίδιοι θέλουμε να επισκεφτούμε ένα website. Πληκτρολογώντας την διεύθυνση του και πατώντας «enter» ο web-browser στέλνει ένα HTTP request στον server ο οποίος φιλοξενεί το website αυτό. Ο server αφού λάβει το request, θα το αναλύσει και έπειτα θα στείλει πίσω στον browser ένα HTTP response. Τα HTTP request/response μηνύματα όμως έχουν μια συγκεκριμένη δομή καθώς επίσης περιέχουν πληροφορία σημαντική για τον δέκτη του μηνύματος.

Μία σημαντική πληροφορία είναι η HTTP μέθοδος (method), που χρησιμοποιείται κατά το request θέλοντας να δηλώσει τον σκοπό του και τι αυτό θέλει να πετύχει. Παρακάτω αναφέρονται μερικές από αυτές:

- GET – σκοπό έχει την ανάκτηση δεδομένων
- POST – δημιουργεί νέα δεδομένα
- PUT – ενημερώνει δεδομένα που ήδη υπάρχουν
- DELETE – διαγράφει δεδομένα

Επιπλέον, το request body είναι απαραίτητη παράμετρος στις περιπτώσεις όπου ένα request χρησιμοποιεί τις μεθόδους POST ή PUT. Όπως αναφέρθηκε παραπάνω, οι δύο αυτές μέθοδοι υποδεικνύον είτε ότι ένας νέος πόρος (resource) επρόκειτο να δημιουργηθεί, είτε ένας υπάρχον πόρος επρόκειτο να αλλάξει. Και στις δύο περιπτώσεις θα πρέπει, κατά το request, να σταλθούν τα νέα δεδομένα, έτσι ώστε ο server να γνωρίζει ποιος είναι ο νέος πόρος που καλείται να δημιουργήσει και επίσης, ποιος είναι ο νέος πόρος με τον οποίο θα αντικαταστήσει τον παλιό.

Μια άλλη σημαντική πληροφορία είναι οι επικεφαλίδες (headers) ενός HTTP μηνύματος. Οι επικεφαλίδες αυτές επιτρέπουν στον client και τον server να ενσωματώσουν επιπλέον πληροφορία σε ένα request ή ένα response.

Μαζί με τα παραπάνω, το status code που περιλαμβάνει ένα HTTP response αποτελεί σημαντική λεπτομέρεια, καθώς μας ενημερώνει εάν το request ολοκληρώθηκε επιτυχώς. Ένα status code απεικονίζεται με έναν τριψήφιο ακέραιο και μερικά από αυτά είναι:

- 200 OK – Το request ήταν επιτυχημένο.
- 201 Created – Το request ήταν επιτυχημένο και τα δεδομένα δημιουργήθηκαν. Ένα τέτοιο status code θα το συναντήσουμε σε ένα HTTP response το οποίο στάλθηκε από τον server μετά από ένα HTTP POST/PUT request.
- 400 Bad Request – Ο server δεν μπόρεσε να αναλύσει/αναγνωρίσει το request λόγω εσφαλμένης διατύπωσής του.
- 401 Unauthorized – Ο server δεν μπορεί να ολοκληρώσει επιτυχημένα το request, διότι η ταυτότητα του client δεν αναγνωρίστηκε επιτυχώς.
- 403 Forbidden - Ο server δεν μπορεί να ολοκληρώσει επιτυχημένα το request, διότι ο client , αν και έχει ταυτοποιηθεί επιτυχώς, δεν διαθέτει την κατάλληλη πρόσβαση σε αυτόν τον πόρο που ζητάει.
- 404 Not Found – Με αυτόν τον τρόπο ο server δηλώνει ότι, παρόλο η σύνταξη του request είναι σωστή και μπόρεσε αυτό να αναλυθεί, ο πόρος που ζητείται από τον client δεν μπόρεσε να βρεθεί.
- 500 Internal Server Error – Στην περίπτωση αυτή ο server απέτυχε εσωτερικά να επεξεργαστεί το αίτημα που έφτασε σε αυτόν. Το λάθος σε αυτήν την περίπτωση εντοπίζεται κατά κύριο λόγο στο λογισμικό που έχει γραφτεί για αυτόν.

Το HTTP πρωτόκολλο είναι εύχρηστο, και χρησιμοποιείται πλήρως από την client-server αρχιτεκτονική. Αποτελεί κύριο συστατικό όλων των web εφαρμογών, εφόσον αυτές απαιτείται να επικοινωνήσουν με έναν server, μέσω διαδικτύου, προκειμένου να ανακτήσουν, και όχι μόνο, σημαντικά για την λειτουργία τους δεδομένα.

Web API

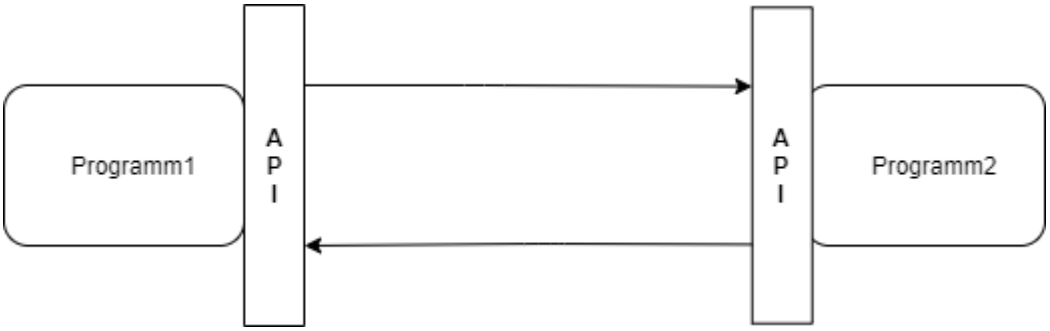
Όταν ερχόμαστε σε επαφή με έναν υπολογιστή, με ένα κινητό ή ακόμα και με μια κονσόλα παιχνιδιών το πρώτο πράγμα που αντικρίζουμε είναι μία διεπαφή. Η διεπαφή αυτή μας γνωστοποιεί στην ουσία το πώς πρέπει να δράσουμε ώστε να πετύχουμε αυτό που θέλουμε. Χάριν παραδείγματος, σε ένα smartphone πρέπει να πατήσουμε το εικονίδιο με το ακουστικό τηλεφώνου, προκειμένου να οδηγηθούμε σε μια άλλη οθόνη οπού θα έχουμε διαφορετική διεπαφή και η οποία στη συνέχεια θα μας κατευθύνει να καλέσουμε το πρόσωπο που επιθυμούμε.

Η διεπαφή μιας εφαρμογής που τρέχει σε έναν υπολογιστή, ή σε ένα smartphone, είναι φτιαγμένη έτσι ώστε να χρησιμοποιείται από εμάς τους ανθρώπους, οι οποίοι παίζουμε τον ρόλο του χρήστη (user). Για τον λόγο αυτό ονομάζεται διεπαφή χρήστη (user interface). Είναι μία «πύλη» μέσω της οποίας μπορούμε να εκμεταλλευτούμε την λειτουργικότητα που προσφέρει ένα λογισμικό.

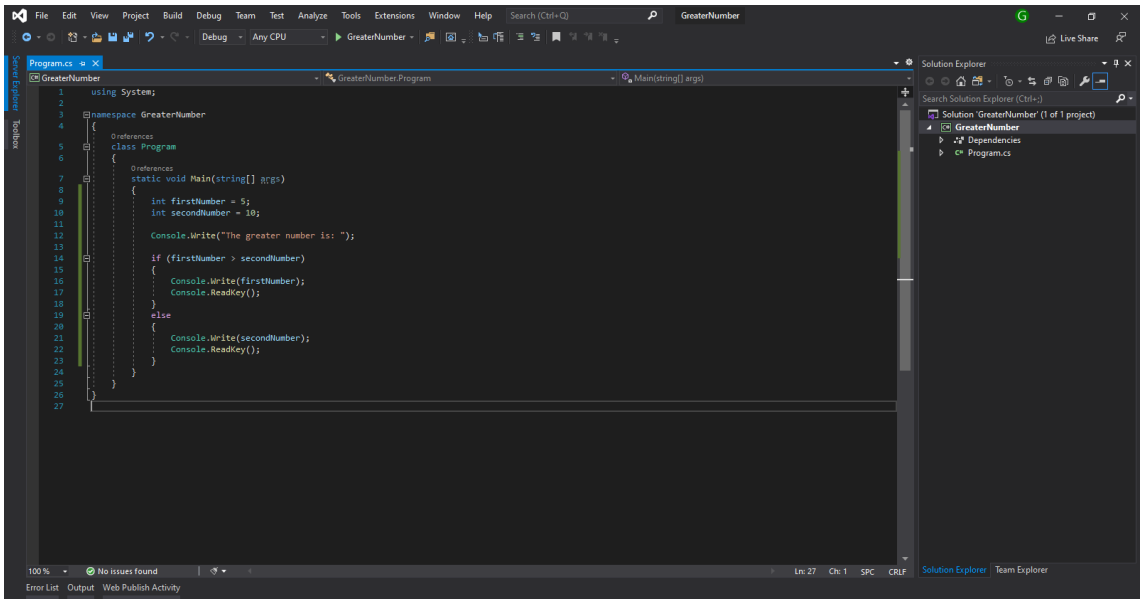
Πέρα όμως από την δυνατότητα επικοινωνίας μεταξύ ενός ανθρώπου και ενός λογισμικού μέσω ενός user interface υπάρχει και η επικοινωνία μεταξύ ενός λογισμικού και ενός άλλου λογισμικού μέσω ενός διαφορετικού είδους διεπαφής.

Ο όρος API αποτελεί το ακρωνύμιο της φράσης Application Programming Interface. Αν το μεταφράσουμε από το τέλος προς την αρχή τότε έχουμε την φράση Διεπαφή Προγράμματος Εφαρμογής. Αυτό σημαίνει ότι ένα λογισμικό μπορεί να εκθέσει τη λειτουργικότητά του με σκοπό να χρησιμοποιηθεί από κάποιο άλλο λογισμικό, πράγμα το οποίο στον προγραμματισμό το συναντάμε πάρα πολύ συχνά.

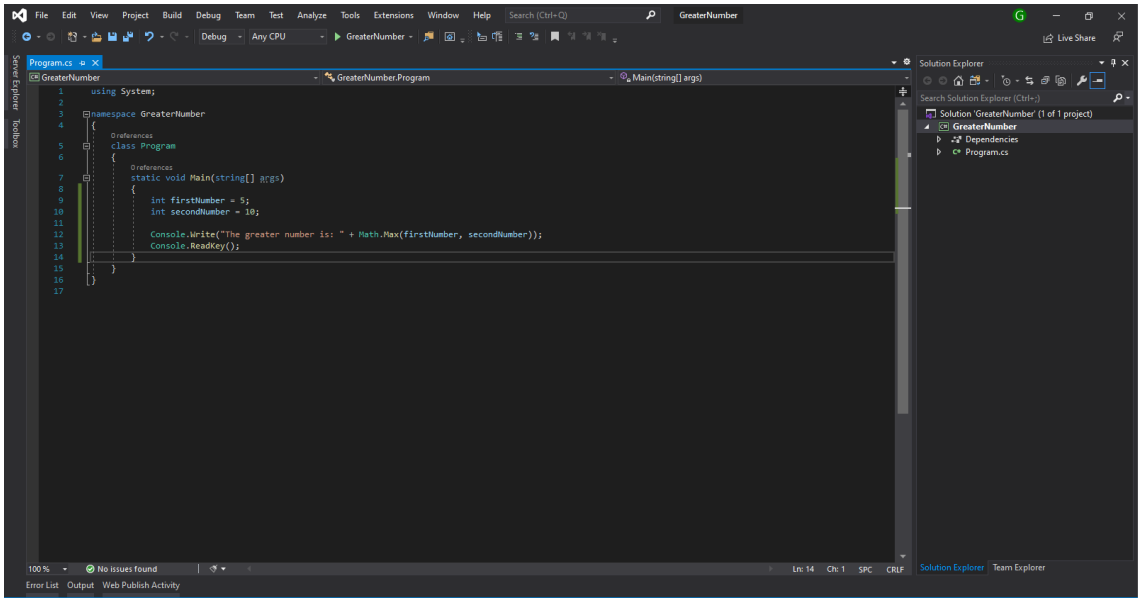
Ας αναφέρουμε ένα απλό παράδειγμα. Έστω ότι χτίζουμε ένα απλό λογισμικό, και πιο συγκεκριμένα ένα console application σε γλώσσα προγραμματισμού C#. Το σενάριο που θέλουμε να καλύψουμε απαιτεί να δείξουμε στην οθόνη τον μεγαλύτερο από δύο ακεραίους. Ο 1ος τρόπος που θα σκεφτόταν κάποιος, θα ήταν να συγκρίνει τους δύο αριθμούς, και αν ο πρώτος είναι μεγαλύτερος από τον δεύτερο δείξε στην οθόνη τον πρώτο, αλλιώς δείξε τον δεύτερο.



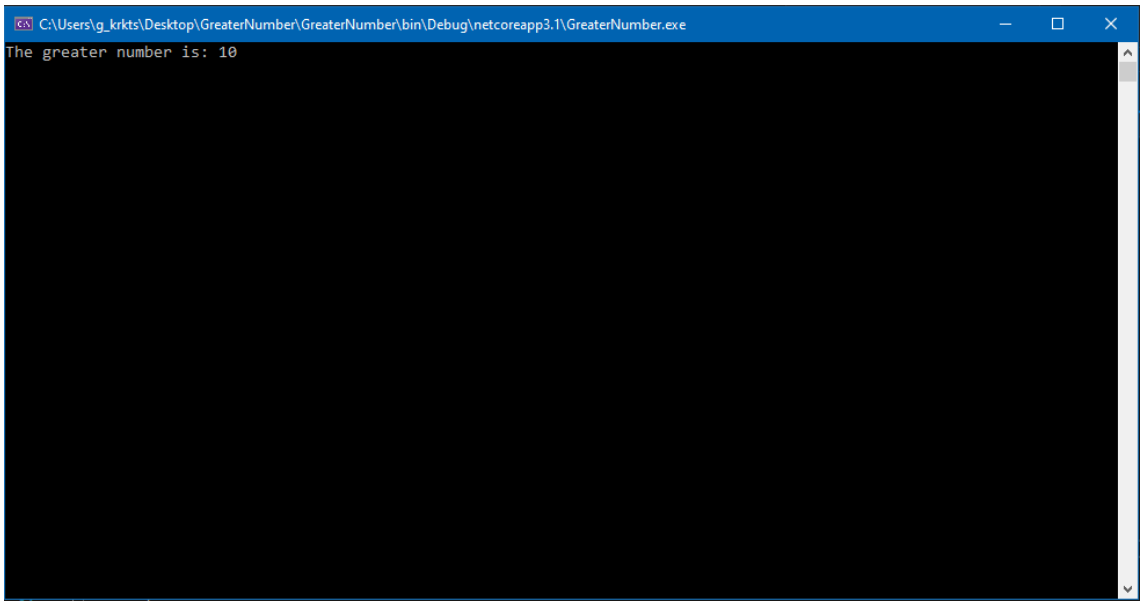
Παρακάτω φαίνεται ο κώδικας που υλοποιεί το σενάριο που αναφέρθηκε σύμφωνα με τον 1ο τρόπο.



Ένας 2ος τρόπος θα ήταν να χρησιμοποιήσουμε ένα API. Να εκμεταλλευτούμε δηλαδή κώδικα που υπάρχει ήδη, ώστε να εξυπηρετήσουμε τον σκοπό του σεναρίου. Στη συγκεκριμένη περίπτωση θα κάνουμε χρήση της κλάσης Math και των μεθόδων που εκθέτει η κλάση αυτή.



Ενώ εδώ φαίνεται το αποτέλεσμα στην κονσόλα μετά από εκτέλεση του προγράμματος και στις δύο περιπτώσεις.

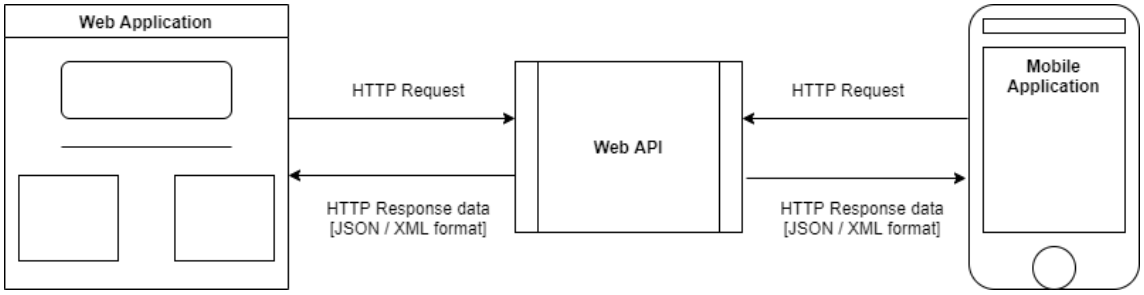


Από τα παραπάνω μπορούμε να δούμε πώς το πρόγραμμά μας επικοινώνησε με ένα άλλο πρόγραμμα, με σκοπό να εκμεταλλευτεί την λειτουργικότητα που προσφέρει και τελικά να πετύχουμε τον σκοπό μας. Η κλάση Math, μαζί με ότι αυτή προσφέρει, αποτελεί ένα API το οποίο οποιοδήποτε λογισμικό, γραμμένο σε C#, μπορεί να χρησιμοποιήσει. Είναι σημαντικό εδώ να σημειώσουμε ότι δεν χρειάστηκε απομακρυσμένη επικοινωνία προκειμένου να επωφεληθούμε των δυνατοτήτων της κλάσης Math.

Τι γίνεται όμως στην περίπτωση όπου έχουμε μία web εφαρμογή και θέλουμε να χρησιμοποιήσουμε ένα API το οποίο βρίσκεται σε κάποιον απομακρυσμένο server; Στην περίπτωση αυτή μιλάμε για ένα Web API, και είναι ένα κομμάτι το οποίο παίζει πολύ σημαντικό ρόλο στην ανάπτυξη μιας full-stack web εφαρμογής.

Μια web εφαρμογή πρέπει να είναι σε θέση να αποθηκεύει, να ανακτά, να τροποποιεί καθώς και να διαγράφει τα δεδομένα των χρηστών της. Λειτουργίες που προφανώς εκτελούνται κατόπιν αιτήματος του εκάστοτε χρήστη. Από τη στιγμή όμως που απαραίτητη προϋπόθεση για να επιτευχθεί αυτό είναι η επικοινωνία μέσω του διαδικτύου, θα πρέπει να υπάρχει μια απομακρυσμένη διεπαφή προγράμματος εφαρμογής (Web API), η οποία θα χρησιμοποιείται από την εφαρμογή μας προκειμένου να εκμεταλλευτεί την λειτουργικότητα του απομακρυσμένου λογισμικού.

Ένα Web API είναι ένα λογισμικό που έχει αναπτυχθεί με μία γλώσσα προγραμματισμού και φιλοξενείται σε κάποιόν server. Σκοπός του είναι να δέχεται HTTP requests από τους clients οι οποίοι ζητούν συγκεκριμένα data προκειμένου να τα διαχειριστούν ανάλογα, και έπειτα να στέλνει τα data αυτά στους clients μέσω HTTP responses. Είναι σημαντικό να αναφέρουμε εδώ ότι το λογισμικό που τρέχει ο εκάστοτε client δεν είναι απαραίτητο να έχει γραφτεί με την ίδια γλώσσα προγραμματισμού με αυτήν που γράφτηκε το Web API.



Στη συνέχεια θα αναλυθεί πώς η αρχιτεκτονική πελάτη – εξυπηρετητή, τα HTTP requests/responses καθώς και η οντότητα Web API, συνδυάστηκαν με τα απαραίτητα εργαλεία όπως οι γλώσσες προγραμματισμού και η βάση δεδομένων, και τελικά οδήγησαν στην ανάπτυξη της web εφαρμογής μας.

Ανάπτυξη Λογισμικού - Front End Development

Για την υλοποίηση μιας web εφαρμογής χρειάζεται να αναπτυχθεί λογισμικό τόσο στη μεριά του πελάτη (client-side), όσο και στη μεριά του εξυπηρετητή (server-side). Παρακάτω θα αναλύσουμε εργαλεία και τεχνικές που χρησιμοποιήθηκαν για την client – side ανάπτυξη λογισμικού.

Το λογισμικό αυτό έχει να κάνει με την λογική που κρύβεται πίσω από τον τρόπο με τον οποίο έχει δομηθεί η παρουσίαση της λειτουργικότητας της εφαρμογής, και πώς αυτή φτάνει στα μάτια του χρήστη. Αυτό που πρέπει κάθε web εφαρμογή να πετύχει, είναι να προσφέρει μια ευχάριστη εμπειρία στους χρήστες της ενώ σημαντικό ρόλο παίζει το να είναι απλή και εύχρηστη.

Για την υλοποίηση του client-side λογισμικού χρησιμοποιήθηκε ένα framework της JavaScript που ακούει στο όνομα Angular. Το Angular framework είναι μια πλατφόρμα ανάπτυξης λογισμικού με την οποία μπορείς να χτίσεις Single - Page Applications χρησιμοποιώντας την γλώσσα σήμανσης HTML σε συνδυασμό με CSS κανόνες, καθώς και τη γλώσσα προγραμματισμού Typescript.

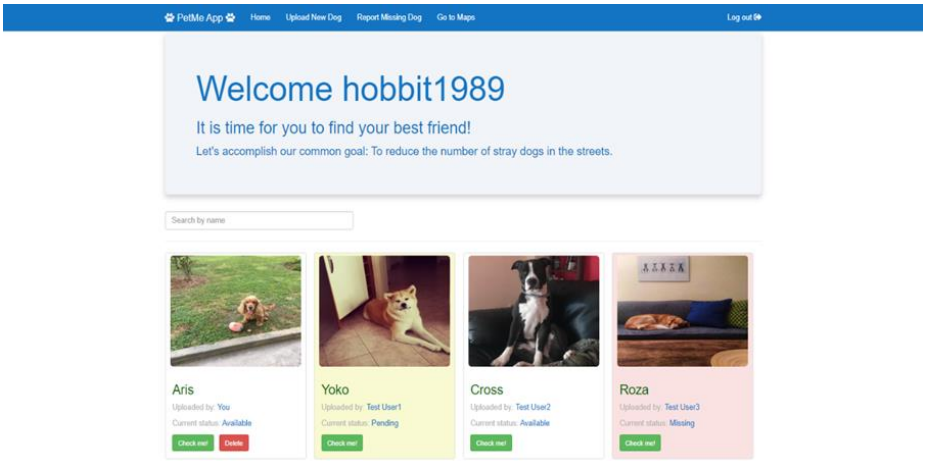
Η τεχνική που κρύβεται πίσω από μία Single - Page εφαρμογή, έγκειται στο γεγονός ότι όσο ο χρήστης περιηγείται στην εφαρμογή και ζητάει να δει τις σελίδες της, δεν αποστέλλεται νέο HTTP request κάθε φορά στον server προκειμένου η εκάστοτε σελίδα να φτάσει στον web browser και τελικά να εμφανιστεί σε αυτόν. Αυτό που πραγματικά συμβαίνει είναι ότι, όταν ο χρήστης καλέσει για πρώτη φορά την εφαρμογή, τότε στον web browser θα κατέβει μία και μόνο σελίδα, το περιεχόμενο καθώς και η δομή της οποίας θα αλλάζει δυναμικά κ ανάλογα με τις επιλογές του εκάστοτε χρήστη.

Αυτό έχει πολλά θετικά στοιχεία σχετικά με την επίδοση της εφαρμογής, δεδομένου ότι όλα τα αρχεία που είναι απαραίτητα για την παρουσίαση αυτής, είναι διαθέσιμα με ένα και μόνο HTTP request στον server. Έπειτα, μένει η διαχείριση των δεδομένων, όπου στην περίπτωση αυτή γίνονται τα κατάλληλα requests, προκειμένου τα data να είναι διαθέσιμα προς παρουσίαση.

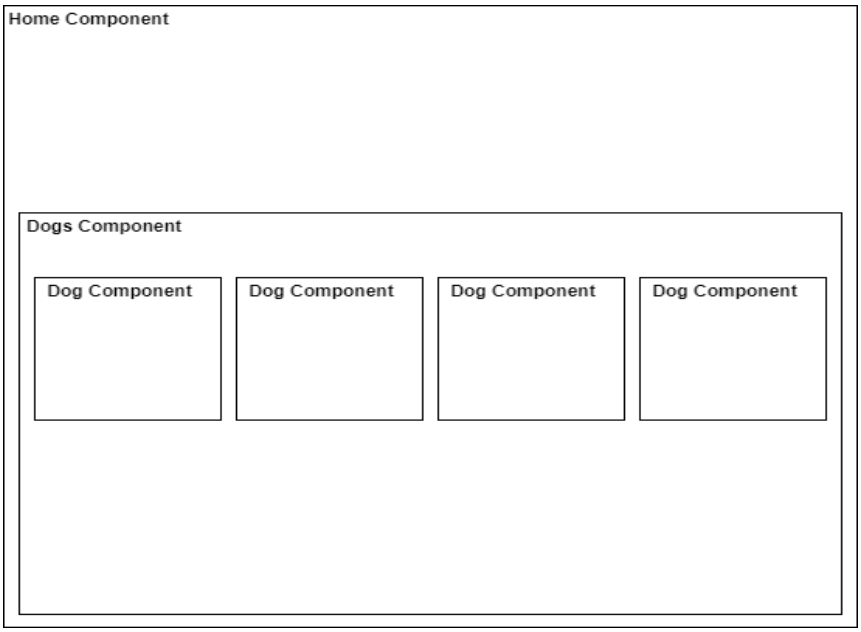
Η Angular διαθέτει αρκετά δομικά στοιχεία που συμβάλλουν στην ανάπτυξη μιας web εφαρμογής. Τα πιο σημαντικά από αυτά είναι τα Components και τα Services. Ας τα εξετάσουμε λίγο πιο λεπτομερώς παρακάτω, και ας δούμε πώς αυτά βοήθησαν στο χτίσιμο της εφαρμογής που υλοποιήθηκε στα πλαίσια αυτής της διπλωματικής εργασίας.

Angular Components

Ένα component στην πραγματικότητα είναι μία αυτόνομη δομή η οποία μπορεί και να επαναχρησιμοποιηθεί μέσα στη σελίδα ανάλογα με τις ανάγκες της εφαρμογής. Είναι μία οντότητα η οποία απαρτίζεται από τον δικό του HTML κώδικα, τους δικούς του CSS κανόνες καθώς και τα δικά του δεδομένα. Η έννοια του component θα γίνει ευκολότερα κατανοητή μέσα από τα παρακάτω διαγράμματα. Ως σημείο αναφοράς θα χρησιμοποιήσουμε την αρχική σελίδα.



Η συγκεκριμένη σελίδα και η ανάλυσή της σε components φαίνεται στο παρακάτω διάγραμμα.



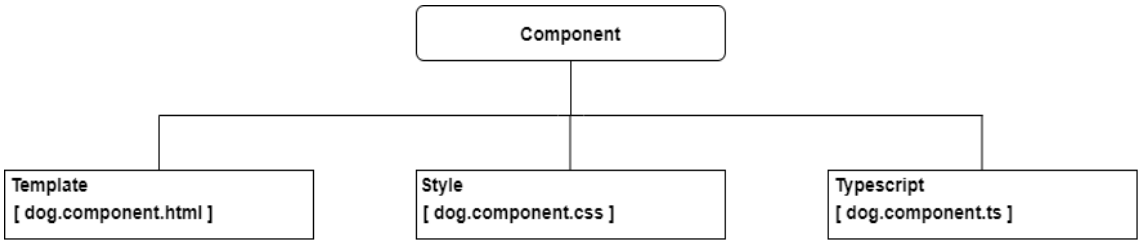
Από το παραπάνω διάγραμμα κάνει την εμφάνισή της η εσωτερική αρχιτεκτονική της σελίδας μέσω της Angular και ειδικότερα μέσω των components. Για να φτάσουμε στην τελική μορφή όμως πρέπει να δώσουμε στο κάθε component μία δομή και μία εμφάνιση. Αυτό επιτυγχάνεται γράφοντας HTML κώδικα σε συνδυασμό με CSS κανόνες. Πέρα όμως από την εμφάνιση, το component χρειάζεται και μία λογική με την βοήθεια της οποίας θα μπορεί να ανακτά τα δεδομένα που χρειάζεται από έναν απομακρυσμένο server καθώς και να τα επεξεργάζεται. Επιπλέον, με την λογική αυτή ένα component μπορεί να διαχειριστεί το περιεχόμενο της σελίδας, να επέμβει δηλαδή στην εμφάνισή της, προσθέτοντας ή αφαιρώντας CSS κανόνες και όχι μόνο.

Η υλοποίηση μέσω της οποίας κάνει την εμφάνισή της η λογική αυτή γίνεται με χρήση μεταβλητών διαφόρων τύπων δεδομένων, μεθόδων αλλά και κλάσεων, τα αντικείμενα των οποίων χρησιμοποιούνται για να αναπαραστήσουν οντότητες χρήσιμες για την εφαρμογή. Ένα παράδειγμα φαίνεται παρακάτω με την αναπαράσταση της κλάσης που αντιπροσωπεύει ένα σκυλί.

```
import { DogStatus } from './dog-status.enum';

export class Dog {
  public id: number;
  public name: string;
  public address: string;
  public imageUrl: string;
  public age: number;
  public breed: string;
  public description: string;
  public status: DogStatus;
  public uploadedBy: string;
  public requestedBy: string;
}
```

Γίνεται πλέον αντιληπτό ότι το οποιοδήποτε component απαρτίζεται από τρία κομμάτια: την δομή, την εμφάνιση και την λογική.



Ένα component γίνεται διαθέσιμο μέσω μιας ιδιότητας του που ονομάζεται **selector**. Ένας selector δεν είναι τίποτα άλλο από ένα HTML tag με όνομα που χαρακτηρίζει το component. Ένα παράδειγμα που θα μας βοηθήσει να καταλάβουμε καλύτερα είναι να αναφέρουμε ότι στην περίπτωση του Dog component, ο selector έχει τη μορφή **<app-dog></app-dog>** και το όνομά του ορίζεται στο Typescript αρχείο του component όπως φαίνεται παρακάτω.

```
@Component({
  selector: 'app-dog',
  templateUrl: './dog.component.html',
  styleUrls: ['./dog.component.css']
})
```

Όταν η Angular συναντήσει τον συγκεκριμένο selector μέσα στον HTML κώδικα, θα πάει και θα δημιουργήσει το αντίστοιχο component. Για να το δημιουργήσει θα εκτελέσει τις παρακάτω ενέργειες:

- Θα χρησιμοποιήσει το Template με βάση το οποίο θα φτιάξει την HTML δομή.
- Θα συμβουλευτεί το αρχείο με τους CSS κανόνες προκειμένου να δώσει την προβλεπόμενη εμφάνιση στην HTML δομή.
- Θα τρέξει τον κώδικα ο οποίος είναι γραμμένος σε γλώσσα Typescript, και ο οποίος θα κάνει ότι χρειάζεται για να εφαρμοστεί σωστά η λογική του component.

Τέλος είναι σημαντικό να αναφέρουμε μια άλλη δυνατότητα που μας παρέχει η Angular η οποία είναι να συμπεριλάβουμε ένα component μέσα σε ένα άλλο. Το μόνο που έχουμε να κάνουμε είναι να εντάξουμε τον selector του ενός μέσα στο Template του άλλου. Παρακάτω θα δούμε πώς αυτό γίνεται εφικτό με την τεχνική του parent-child component. Μία τεχνική η οποία είναι πολύ χρήσιμη για τον λόγο του ότι καθιστά εύκολη την επικοινωνία μεταξύ των components. Η επικοινωνία αυτή γίνεται κατά κύριο λόγο με σκοπό την ανταλλαγή δεδομένων.

Στην ανάλυση των components της αρχικής σελίδας παρατηρούμε ότι το Dog component βρίσκεται μέσα στο Dogs component δίνοντάς στο πρώτο την ιδιότητα του child component, ενώ στο δεύτερο την ιδιότητα του parent component.

dogs.component.html

```
src > app > dogs > dogs.component.html > ...
1   <app-dog *ngFor="let dog of this.dogs | dogPipe: searchTerm" [dog]="dog"></app-dog>
2
```

Το Template του Dogs component είναι σχετικά απλό. Κάνει τόσες επαναλήψεις, όσα είναι και τα αντικείμενα της λίστας this.dogs. Με τον τρόπο αυτό δημιουργεί κάθε φορά και έναν καινούριο selector του Dog component, με αποτέλεσμα να χτίζεται ένα νέο Dog component σε κάθε επανάληψη.

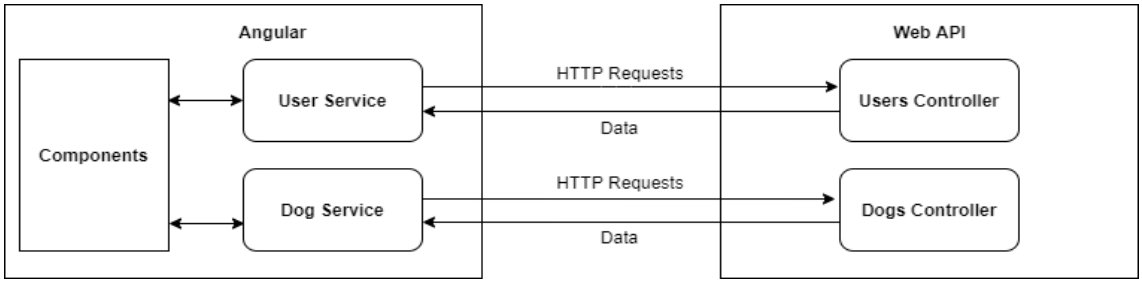
Ένα άλλο σημαντικό στοιχείο της συγκεκριμένης υλοποίησης είναι ότι το Dogs component, ως parent component, μπορεί σε κάθε επανάληψη να περάσει στο child component ένα αντικείμενο της κλάσης Dog προκειμένου να χρησιμοποιήσει τα data που αυτό διαθέτει και τελικά να τα εμφανίσει στο δικό του Template. Έτσι γίνεται δυνατή η επικοινωνία μεταξύ των components.

Angular Services

Έως τώρα έχουμε μιλήσει για τα components τα οποία, εκτός από το να δομούν την εφαρμογή, είναι υπεύθυνα να προβάλλουν και να επεξεργάζονται τα δεδομένα τα οποία διαθέτουν. Ήρθε η ώρα να αναλύσουμε τον τρόπο με τον οποίο τα δεδομένα αυτά γίνονται διαθέσιμα προκειμένου να επεξεργαστούν. Για τον σκοπό αυτό η Angular μας προσφέρει ένα δυνατό εργαλείο προκειμένου η εφαρμογή μας να έχει ανά πάσα ώρα και στιγμή τα δεδομένα που χρειάζεται. Τα services, ουσιαστικά, αποτελούν την πύλη εξόδου της εφαρμογής μας στον έξω κόσμο και παρακάτω θα περιγράψουμε πως γίνεται η χρήση αυτών με κώδικα.

Ένα service δεν είναι τίποτα παραπάνω από μια κλάση η οποία διαθέτει μεθόδους, και όχι μόνο. Κάθε μία από αυτές τις μεθόδους είναι σε θέση να αποστείλει ένα HTTP request σε κάποιο Web API, και να ζητήσει τα δεδομένα που επιθυμεί, να δημιουργήσει καινούρια ή ακόμα και να τροποποιήσει υπάρχοντα δεδομένα. Όταν, για παράδειγμα, θέλουμε να ζητήσουμε όλα τα σκυλιά από τη βάση δεδομένων μας, η εφαρμογή μας θα στείλει ένα HTTP – GET request προκειμένου να ζητήσει τα δεδομένα. Ενώ αν θελήσουμε να κάνουμε εγγραφή στην εφαρμογή, τότε μέσω του service θα σταλθεί ένα HTTP – POST request μαζί με τα δεδομένα τα οποία τελικά θα αποθηκευτούν στη βάση δεδομένων.

Προκειμένου να γίνει πιο κατανοητή η χρησιμότητα των services στην ανάπτυξη της εφαρμογής θα περιγράψουμε παρακάτω δύο από αυτά. Ένα service φτιάχνεται με σκοπό να εξυπηρετήσει ένα συγκεκριμένο περιεχόμενο. Αυτό γίνεται με σκοπό να υπάρχει περισσότερη αυτονομία, με την έννοια ότι κάθε service οφείλει να εξυπηρετεί μία συγκεκριμένη οντότητα της εφαρμογής. Για τον λόγο αυτόν στην εν λόγω εφαρμογή χρειάστηκαν δύο services.



Το user-service διαθέτει μεθόδους οι οποίες μας δίνουν την δυνατότητα να επικοινωνήσουμε με το Web API με σκοπό:

- Να δημιουργήσουμε έναν νέο χρήστη. Η λειτουργικότητα αυτή καλείται κάθε φορά που κάποιος θέλει να κάνει εγγραφή στην εφαρμογή μας. Έτσι, η συγκεκριμένη μέθοδος θα αποστείλει ένα HTTP POST request στο Web API το οποίο θα διαθέτει τα στοιχεία που έχει υποβάλλει ο χρήστης με σκοπό να γίνει η καταγραφή τους στην βάση.

```
public postNewUser(user: User): Observable<User> {
  return this.http.post<User>(this.apiUrl + 'api/users/CreateUser', user);
}
```

- Να αναγνωρίσουμε αν ένας χρήστης διαθέτει λογαριασμό στην εφαρμογή. Η συγκεκριμένη μέθοδος θα αποστείλει ένα HTTP POST request στο Web API το οποίο θα διαθέτει τα credentials του χρήστη, και αφού αυτά επαληθευτούν επιτυχώς, ο χρήστης θα μπορέσει να μεταβεί στην αρχική του σελίδα.

```
public login(username: string, password: string): Observable<User> {
  return this.http.get<User>(this.apiUrl + 'api/users/Login/' + username + '/' + password);
}
```

Το dog-service διαθέτει μεθόδους οι οποίες μας δίνουν την δυνατότητα να επικοινωνήσουμε με το Web API με σκοπό:

- Να ζητήσει από το Web API όλα τα διαθέσιμα σκυλιά προς υιοθεσία, προκειμένου να έχει πρόσβαση ο χρήστης. Η συγκεκριμένη μέθοδος θα αποστείλει ένα HTTP GET request στο Web API το οποίο θα επιστρέψει όλες τις αντίστοιχες εγγραφές που είναι καταχωρημένες στη βάση δεδομένων.

```
public getDogs() {
  return this.http.get<Dog[]>(this.getBaseUrl() + 'api/dogs/GetAll').subscribe(
    (dogs: Dog[]) => {
      this.dogs = dogs;
      this.missingDogs = dogs.filter(d => d.status === DogStatus.Missing) as MissingDog[];
      this.dogSource.next(this.dogs.slice());
    }
  );
}
```

- Να δημιουργήσουμε μία νέα εγγραφή στη βάση δεδομένων που θα αφορά ένα νέο σκυλί. Στην περίπτωση αυτή θα αποστείλουμε ένα HTTP POST request στο Web API καθώς επίσης και τα στοιχεία του προς ανάρτηση σκυλιού.

```
public deleteDog(dog: Dog): Observable<Dog> {
  console.log('DOG-SERVICE: DELETE DOG');
  return this.http.post<Dog>(this.getBaseUrl() + 'api/dogs/Delete', dog).pipe(
    tap((data) => {
      this.removeDogFromList(data);
    })
  );
}
```

- Να ενημερώσουμε τα δεδομένα ενός σκυλιού που υπάρχει ήδη καταχωρημένο στη βάση δεδομένων, αλλάζοντας κάποιες από τις ιδιότητές του, όπως το status του σκυλιού.

```
public updateDogStatus(dog: Dog, status: number): Observable<Dog> {
  console.log('DOG-SERVICE: UPDATE DOG STATUS');
  return this.http.put<Dog>(this.getBaseUrl() + `api/dogs/Update/${status}`, dog);
}
```

Ανάπτυξη Λογισμικού - Back End Development

Παραπάνω περιγράψαμε τις διεργασίες εκείνες που ήταν απαραίτητες προκειμένου να καθοριστεί η εμφάνιση της εφαρμογής, να γίνουν διαθέσιμα όσα δεδομένα χρειάζεται να προβληθούν, ενώ συμπεριλάβαμε και τη λογική μέσω της οποίας τα δεδομένα αυτά επεξεργάζονται κατάλληλα ώστε στο τέλος να ικανοποιήσουν τα αιτήματα του χρήστη, όση ώρα εκείνος περιηγείται στην εφαρμογή.

Η Angular χρησιμοποιεί τα services ώστε να αποστείλει HTTP requests σε έναν απομακρυσμένο server με σκοπό να ανακτήσει, τροποποιήσει, διαγράψει ή και να δημιουργήσει νέα δεδομένα. Έως τώρα δεν έχουμε περιγράψει καθόλου τι ακριβώς συμβαίνει από τη στιγμή που ένα service της Angular θα αποστείλει ένα HTTP request. Ο server σε αυτήν την περίπτωση θα χρειαστεί να διαχειριστεί αυτό το request, όπως και όλα τα υπόλοιπα που θα καταφθάνουν σε αυτόν για όση ώρα ο χρήστης είναι στην εφαρμογή.

Η server-side λειτουργικότητα υλοποιήθηκε χρησιμοποιώντας το ASP.NET Core Framework της Microsoft και την γλώσσα προγραμματισμού C#. Με τα παραπάνω εργαλεία φτιάχτηκε ένα Web API το οποίο δέχεται HTTP requests από τον πελάτη (client). Με τη λογική που διαθέτει, αναλύει το request, εκτελεί ότι υπαγορεύεται από τη δομή του, και τέλος αποστέλλει πίσω στον πελάτη ένα HTTP response το οποίο περιέχει τα κατάλληλα data που σε πρώτο χρόνο ζητήθηκαν.

Παρακάτω θα περιγράψουμε την server-side ανάπτυξη λογισμικού που τελικά εφοδιάζει τον server με την λογική αυτή, δίνοντάς του την ικανότητα να διαχειρίζεται τα αιτήματα που δέχεται και να επιστρέφει στους πελάτες τις απαντήσεις που περιμένουν. Επιπλέον θα γίνει ανάλυση της αρχιτεκτονικής του λογισμικού καθώς και της σχεδίασης αυτού.

Models

Τα δεδομένα παίζουν πολύ σημαντικό ρόλο στην ανάπτυξη μίας εφαρμογής, καθώς δίχως αυτά δεν μπορεί να υπάρξει σωστή και επαρκής λειτουργικότητα. Σκοπός της client-server αρχιτεκτονικής είναι η ανταλλαγή δεδομένων. Για να γίνει δυνατή αυτή η ανταλλαγή θα πρέπει με κάποιον τρόπο να αναπαραστήσουμε τα δεδομένα προκειμένου να είμαστε σε θέση αργότερα να τα επεξεργαστούμε, και όχι μόνο. Η αναπαράσταση των δεδομένων στον server γίνεται με την δημιουργία C# κλάσεων. Οι κλάσεις αυτές γεννιούνται μετά από μία ανάλυση που γίνεται σχετικά με το θέμα της εφαρμογής, τις ανάγκες του οποίου καλείται να καλύψει το λογισμικό.

Μία από τις κύριες οντότητες (entities) στην εν λόγω εφαρμογή είναι το σκυλί. Η λειτουργικότητα που προσφέρει δεν μπορεί να σταθεί χωρίς την αναπαράσταση της οντότητας αυτής. Επομένως είναι επιτακτική ανάγκη να έχουμε μία κλάση η οποία θα περιέχει τις ιδιότητες που αντιπροσωπεύουν ένα σκυλί.

```
public class Dog : BaseEntity
{
    3 references | 0 exceptions
    public string Name { get; set; }
    2 references | 0 exceptions
    public string Address { get; set; }
    3 references | 0 exceptions
    public string ImageUrl { get; set; }
    0 references | 0 exceptions
    public int Age { get; set; }
    0 references | 0 exceptions
    public string Breed { get; set; }
    1 reference | 0 exceptions
    public string Description { get; set; }
    0 references | 0 exceptions
    public string UploadedBy { get; set; }
    0 references | 0 exceptions
    public string RequestedBy { get; set; }
    5 references | 0 exceptions
    public DogStatus Status { get; set; }
    0 references | 0 exceptions
    public double Latitude { get; set; }
    0 references | 0 exceptions
    public double Longitude { get; set; }
    2 references | 0 exceptions
    public User User { get; set; }
}
```

Μπορούμε να παρατηρήσουμε ότι μέσα στην κλάση Dog εμπεριέχεται όλη η πληροφορία που εμείς θεωρούμε ότι θα καλύψει με τον καλύτερο τρόπο τις ανάγκες μας. Ωστόσο το περιεχόμενο είναι δυναμικό. Αυτό σημαίνει ότι στην πορεία ανάπτυξης του λογισμικού μπορούν είτε να προστεθούν, είτε να αφαιρεθούν ιδιότητες από την κλάση, ανάλογα πάντα με το τι απαιτείται από το business logic της εφαρμογής.

Μία άλλη απαραίτητη οντότητα της εφαρμογής είναι ο χρήστης. Είναι αναγκαίο να τον αναπαραστήσουμε προκειμένου να μπορέσουμε να υποστηρίξουμε λειτουργικότητα όπως είναι η εγγραφή και η σύνδεση στην εφαρμογή. Για τον λόγο αυτό φτιάχτηκε μια νέα κλάση η οποία κρατάει όλη την απαραίτητη πληροφορία που περιγράφει έναν χρήστη.

```
public class User : BaseEntity
{
    1 reference | 0 exceptions
    public string FullName { get; set; }
    0 references | 0 exceptions
    public string Address { get; set; }
    2 references | 0 exceptions
    public string Username { get; set; }
    1 reference | 0 exceptions
    public string Email { get; set; }
    2 references | 0 exceptions
    public string Password { get; set; }
    2 references | 0 exceptions
    public ICollection<Dog> MyDogs { get; set; }
}
```

Το σύνολο των κλάσεων οι οποίες χρειάζονται ώστε να αναπαραστήσουν όλες τις απαραίτητες οντότητες, αποτελούν το Μοντέλο της εφαρμογής (Model). Όπως και στην περίπτωση των ιδιοτήτων των κλάσεων, έτσι και εδώ, το business logic της εφαρμογής ενδέχεται να οδηγήσει στη δημιουργία νέων κλάσεων, καθώς και στη διαγραφή άλλων. Είναι πολύ σημαντικό να τονίσουμε εδώ, πως η διαδικασία επιλογής των κλάσεων, πρέπει να γίνεται μετά από εκτεταμένη έρευνα και ανάλυση των αναγκών της εφαρμογής, ώστε να προληφθούν τυχόν μελλοντικές μεγάλες αλλαγές που ενδέχεται να επηρεάσουν την λειτουργικότητά της.

Controllers

Η server-side ανάπτυξη λογισμικού έχει να κάνει με την λογική που πρέπει να διαθέτει ο server για να διαχειριστεί τα διαφορετικά requests που καταφτάνουν σε αυτόν. Είδαμε σε προηγούμενο κεφάλαιο ότι η Angular μέσω των services μπορεί να αποστείλει τέτοια requests στον server. Για να το πετύχει αυτό χρησιμοποιώντας το HTTP πρωτόκολλο θα πρέπει ένα service να ξέρει το URL το οποίο πρέπει να καλέσει ώστε να εκτελέσει αυτό ακριβώς που επιθυμεί. Για περισσότερη κατανόηση της όλης διαδικασίας θα περιγράψουμε το σενάριο κατά το οποίο η εφαρμογή μας προβάλλει τα διαθέσιμα προς υιοθεσία σκυλιά στον χρήστη, ο οποίος μόλις συνδέθηκε στην εφαρμογή.

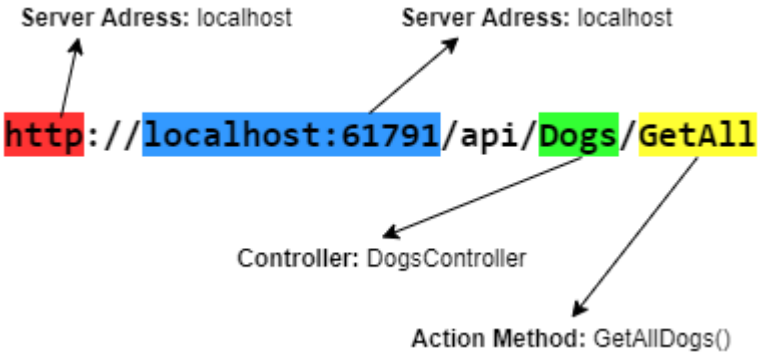
```
public getDogs() {
  return this.http.get<Dog[]>(this.getBaseUrl() + 'api/dogs/GetAll').subscribe(
    (dogs: Dog[]) => {
      this.dogs = dogs;
      this.missingDogs = dogs.filter(d => d.status === DogStatus.Missing) as MissingDog[];
      this.dogSource.next(this.dogs.slice());
    }
  );
}
```

Η δομή του URL είναι εκείνη που θα καθορίσει ποιο κομμάτι κώδικα θα εκτελεστεί στον server, όταν ένα request φτάσει σε αυτόν. Ο server διαβάζει το URL, και κατευθύνεται από αυτό προκειμένου να καταλάβει τι ακριβώς πρέπει να εκτελέσει καθώς και που να το βρει. Για την περαιτέρω καλύτερη εξήγηση, θα εισάγουμε εδώ την έννοια του controller που παίζει πολύ σημαντικό ρόλο στην ανάπτυξη του server-side λογισμικού.

Ο controller είναι μία κλάση η οποία, μέσω της λειτουργικότητας που διαθέτει, θα δεχτεί, θα αναλύσει και τελικά θα απαντήσει στο HTTP request που θα καταφτάσει σε αυτόν. Κάθε request που καταφθάνει στον server έχει σκοπό να ανακτήσει έναν ή πολλούς από τους πόρους του, να δημιουργήσει καινούριους, ή ακόμα και να τροποποιήσει κάποιους που ήδη υπάρχουν. Όπως μας υποδεικνύει το Μοντέλο που περιγράψαμε νωρίτερα, οι έως τώρα διαθέσιμοι πόροι είναι τα αντικείμενα που έχουν να κάνουν με την οντότητα «Σκυλί», καθώς και τα αντικείμενα που έχουν να κάνουν με την οντότητα «Χρήστης». Έτσι, σκοπός των requests που δέχεται ο server είναι να επενεργήσει πάνω σε αυτά τα αντικείμενα, με τρόπο που υποδεικνύεται από τη δομή του HTTP μηνύματος.

Μία σωστή τακτική είναι να φτιάχνουμε έναν ξεχωριστό controller για κάθε οντότητα της εφαρμογής, και ο οποίος θα διαχειρίζεται όλα εκείνα τα requests που αφορούν την οντότητα αυτή. Με τον τρόπο αυτόν διαχωρίζουμε την λειτουργικότητα, προσδίδοντας στον κάθε controller ένα σαφές περιεχόμενο. Το περιεχόμενο αυτό αφορά μεταβλητές και μεθόδους.

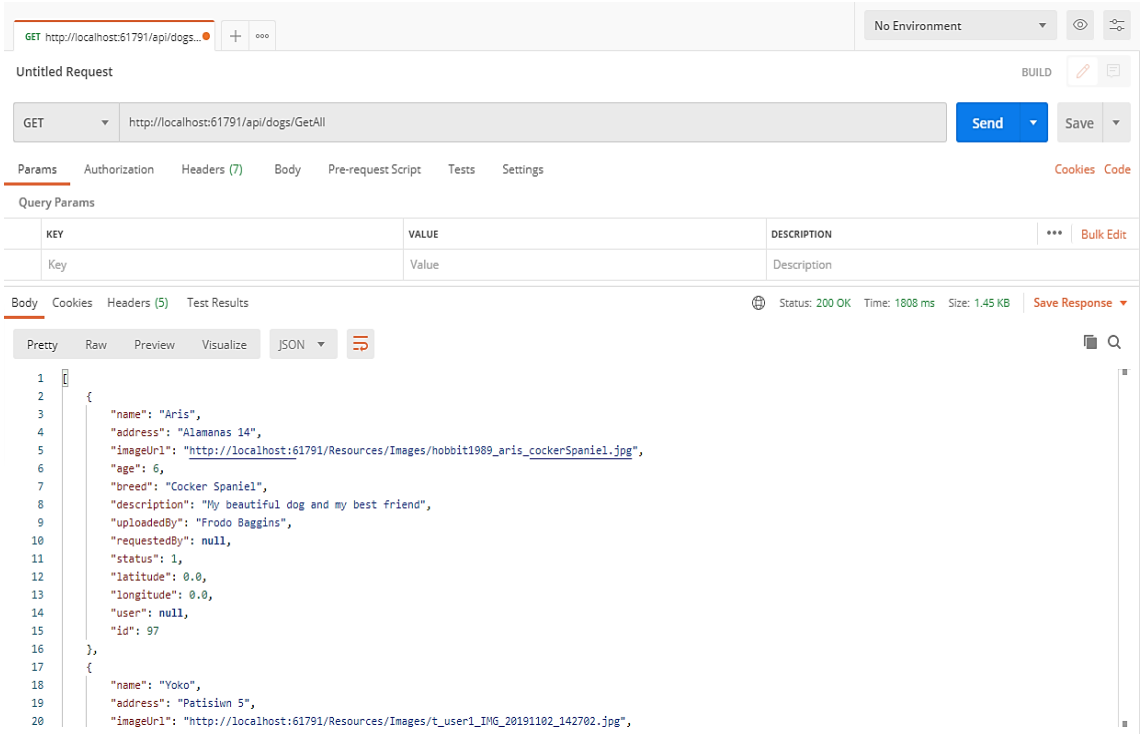
Οι μέθοδοι αυτές ονομάζονται action methods και αποτελούν τα endpoints του Web API. Σκοπός τους είναι να εξυπηρετήσουν το εισερχόμενο request. Η επιλογή της σωστής μεθόδου από τον controller υποδεικνύεται από τη δομή του request. Σε αυτό το σημείο θα χρειαστεί να περιγράψουμε την ανατομία ενός URL, όπως αυτό που χρησιμοποιεί το service της Angular για να ζητήσει τα διαθέσιμα σκυλιά από τον server.



Όταν το service καλέσει το συγκεκριμένο URL, θα χρησιμοποιήσει το HTTP πρωτόκολλο προκειμένου να αποστείλει ένα request στον server ο οποίος βρίσκεται στην διεύθυνση που δηλώνεται. Επιπλέον, δύο σημαντικές πληροφορίες είναι το όνομα του controller για τον οποίο προορίζεται το request, καθώς και η action method που θα εκτελεστεί και τελικά θα φέρει εις πέρας το request αυτό. Παρακάτω απεικονίζεται η μέθοδος που τελικά θα εκτελεστεί καθώς και ο κώδικας που διαθέτει.

```
// GET: api/dogs/GetAll
[HttpGet("GetAll")]
0 references
public IActionResult GetAllDogs()
{
    var dogs = _repository.GetAll();
    return this.Ok(dogs);
}
```

Η μέθοδος αυτή θα ζητήσει από τη βάση δεδομένων όλα τα διαθέσιμα σκυλιά, και αμέσως μετά θα τα επιστρέψει. Για να δούμε αναλυτικά όλα τα δεδομένα που γυρνάει σαν απάντηση η συγκεκριμένη μέθοδος θα κάνουμε χρήση του Postman.



Το Postman είναι ένα εργαλείο με τη βοήθεια του οποίου μπορούμε να στέλνουμε requests στον server και άμεσα να βλέπουμε τι μας επιστρέφει. Αυτό είναι πολύ βοηθητικό γιατί μπορούμε να συμπεράνουμε αν τα endpoints της εφαρμογής μας δουλεύουν σωστά και, αν όχι, να μπορέσουμε από το status code του HTTP response να διακρίνουμε τι πήγε λάθος. Βλέπουμε ότι τα δεδομένα που επιστρέφονται είναι αντικείμενα της κλάσης Dog σε JSON format. Με αυτόν τον τρόπο το service της Angular λαμβάνει τα δεδομένα αυτά και είναι πλέον σε θέση να τα διαχειριστεί αναλόγως.

Entity Framework Core

Το ASP.NET Core Framework μας δίνει τη δυνατότητα να εκμεταλλευτούμε ένα πολύ δυνατό εργαλείο το οποίο είναι το Entity Framework Core. Το εργαλείο αυτό λειτουργεί σαν Object Relational Mapper (ORM).

Το EF Core μας δίνει τη δυνατότητα είτε να δημιουργήσουμε μία βάση δεδομένων κάνοντας χρήση του Μοντέλου, είτε να δημιουργήσουμε το Μοντέλο από μία βάση δεδομένων που χτίσαμε από την αρχή. Στην πρώτη προσέγγιση, η οποία ονομάζεται “code-first approach ”, το EF Core διαβάζει το Μοντέλο, και τις κλάσεις από τις οποίες αυτό αποτελείται, και φτιάχνει μία καινούρια βάση δεδομένων. Η δομή του κάθε πίνακα στη βάση θα αντιστοιχεί στη δομή της κάθε οντότητας της εφαρμογής. Αυτό σημαίνει ότι κάθε ιδιότητα της οντότητας «Σκυλί» θα αντιστοιχεί σε μια κολώνα του αντίστοιχου πίνακα, καθώς επίσης και ότι κάθε εγγραφή στον πίνακα αυτό θα αντιπροσωπεύει ένα αντικείμενο της κλάσης Dog. Στη δεύτερη προσέγγιση, η οποία ονομάζεται “database-first approach ”, διαβάζει τους πίνακες της βάσης και φτιάχνει τις κλάσεις που θα αποτελέσουν το Μοντέλο. Η προσέγγιση που επιλέχτηκε για την ανάπτυξη της παρούσας εφαρμογής είναι η code-first.

Έτσι, είμαστε σε θέση να κάνουμε χρήση του Μοντέλου της server-side εφαρμογής προκειμένου να δημιουργήσουμε τους πίνακες στη βάση δεδομένων μας. Αυτό μπορεί να επιτευχθεί κάνοντας χρήση των κλάσεων του EF Core. Μία από αυτές είναι και η κλάση που κληρονομεί από την κλάση DbContext, και αποτελεί τη γέφυρα της εφαρμογής με την βάση δεδομένων.

```
31 references
public class AppDbContext : DbContext
{
    0 references | 0 exceptions
    public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)
    {
        Database.Migrate();
    }

    0 references | 0 exceptions
    public DbSet<Dog> Dogs { get; set; }
    4 references | 0 exceptions
    public DbSet<User> Users { get; set; }

    0 references | 0 exceptions
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.ApplyConfiguration(new DogEntityConfiguration());

        modelBuilder.ApplyConfiguration(new UserEntityConfiguration());
    }

}
```

Η κλάση αυτή διαθέτει δύο αντικείμενα της κλάσης DbSet<T>, το ένα για την οντότητα «Σκυλί» και το άλλο για την οντότητα «Χρήστης». Αυτές οι δύο ιδιότητες της κλάσης αντιπροσωπεύουν τους πίνακες που θα φτιαχτούν στη βάση δεδομένων. Αφού έχει γραφτεί ο απαραίτητος κώδικας και ο οποίος θα χρησιμοποιηθεί από το EF Core για να φτιάξει τελικά τη βάση δεδομένων και τους πίνακές της, θα χρειαστεί να τρέξουμε μια σειρά εντολών οι οποίες θα φέρουν εις πέρας την όλη διαδικασία.

Οι εντολές αυτές είναι οι εξής:

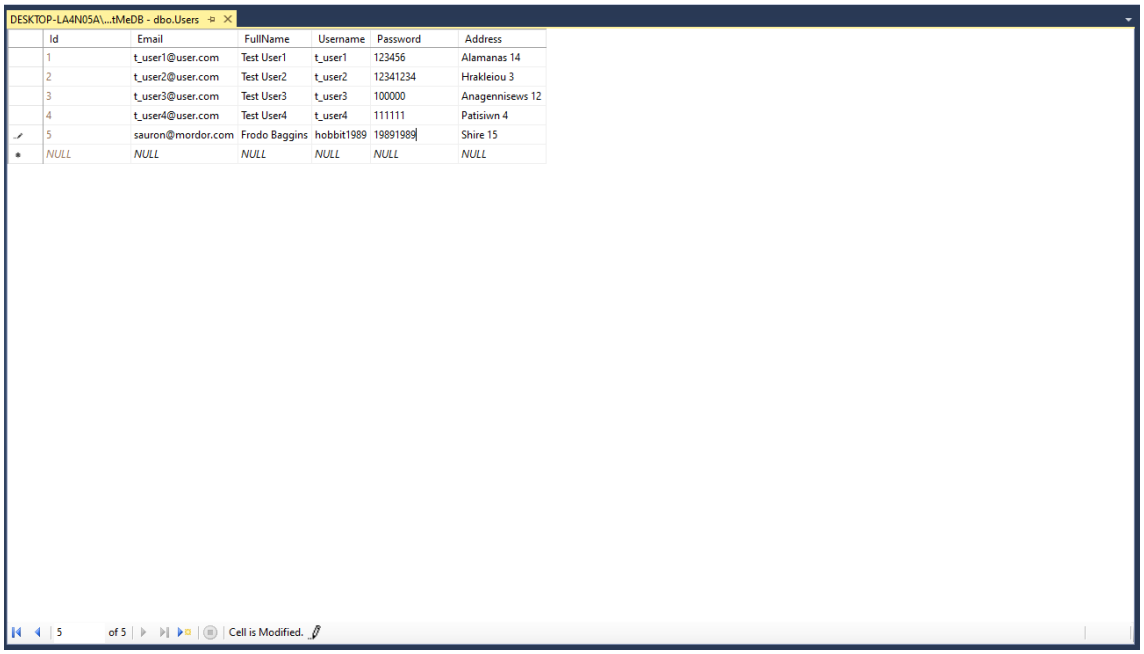
- Add-Migration: Η εντολή αυτή προσθέτει ένα καινούριο αρχείο στην εφαρμογή μας το οποίο περιέχει τις όλες τις αλλαγές που έχει βρει το EF Core διαβάζοντας το Μοντέλο και οι οποίες θα εκτελεστούν στη βάση με την εκτέλεση της επόμενης εντολής.
- Update-Database: Η εντολή αυτή θα πάει και θα εφαρμόσει στην βάση δεδομένων τις αλλαγές που έχει βρει το EF Core από την πρώτη εντολή.

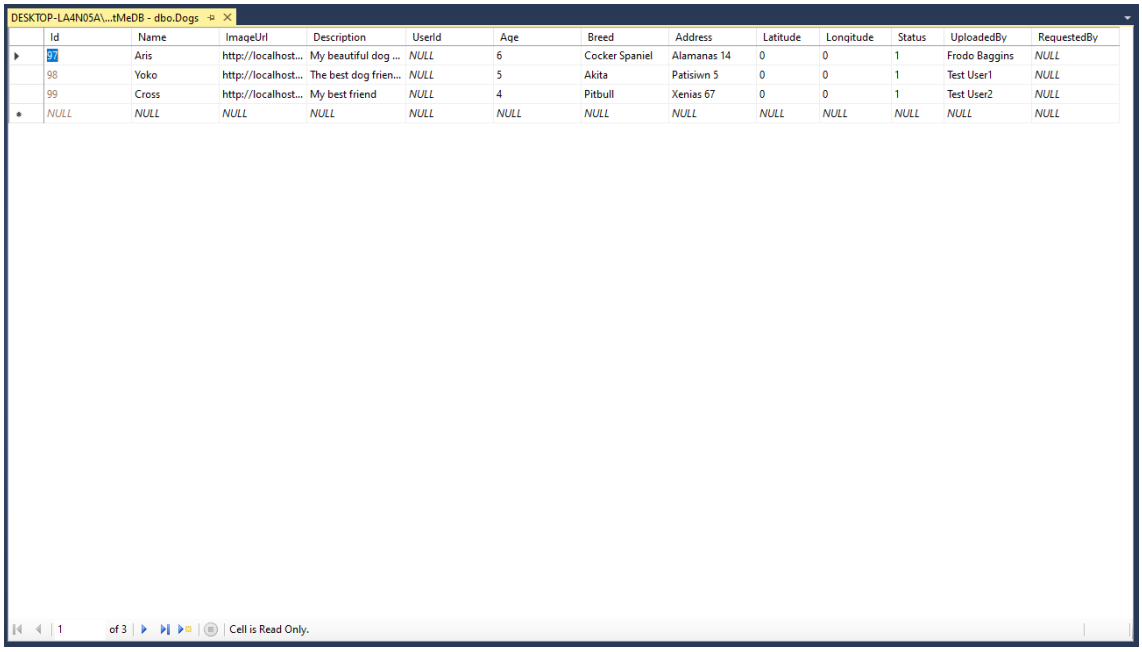
Με την code-first προσέγγιση καταφέραμε να αποτυπώσουμε την δομή του Μοντέλου σε μία βάση δεδομένων χωρίς να χρειαστεί να ασχοληθούμε απευθείας με αυτήν. Αυτό το πετύχαμε γράφοντας τις κλάσεις που εκπροσωπούν τις οντότητες της εφαρμογής, καθώς και μερικές ακόμα λεπτομέρειες που το EF Core χρειάζεται για να κάνει σωστά την αντιστοίχιση.

Ανάπτυξη Λογισμικού – Βάση Δεδομένων

Οποιαδήποτε εφαρμογή λογισμικού δεν μπορεί να λειτουργήσει σωστά αν δεν εφοδιαστεί με δεδομένα. Τα δεδομένα είναι εκείνα που δίνουν νόημα στην εκάστοτε εφαρμογή και την κάνουν χρήσιμη. Σημαντικό σημείο στην ανάπτυξη ενός λογισμικού αποτελεί το ότι κάποια από αυτά τα δεδομένα είναι τόσο απαραίτητα που η ανάγκη αποθήκευσής τους, με σκοπό την μελλοντική ανάκτηση ή επεξεργασία, είναι επιτακτική.

Αυτό επιτυγχάνεται δημιουργώντας μία βάση δεδομένων η οποία θα αποθηκεύει όλη την απαραίτητη πληροφορία που χρειάζεται η εφαρμογή για να μπορεί να υποστηρίξει την λειτουργικότητά της. Τα δεδομένα της συγκεκριμένης εφαρμογής αφορούν την οντότητα «Σκυλί» και την οντότητα «Χρήστης». Προκειμένου οι πληροφορίες για τις δύο αυτές οντότητες να αποθηκεύονται και να είναι διαθέσιμες, χρησιμοποιήθηκε μία σχεσιακή βάση δεδομένων με 2 πίνακες κατά αντιστοιχία με τις δύο οντότητες της εφαρμογής.





Ως βάση δεδομένων χρησιμοποιήθηκε ο SQL Server της Microsoft. Όπως είδαμε παραπάνω, όταν ένα HTTP request φτάνει στον server, αυτό προωθείται στον controller ο οποίος εκτελεί στην συνέχεια καλεί την σωστή action method. Ο κώδικας που τρέχει στη μέθοδο αυτή θα χρειαστεί πολλές φορές να επικοινωνήσει με τη βάση δεδομένων προκειμένου να ενεργήσει επί των δεδομένων που κατοικούν σε αυτήν.

Για να γίνει η επικοινωνία αυτή εφικτή θα πρέπει ο server να ξέρει σε ποια βάση να συνδεθεί, ώστε μετά να ζητήσει τα δεδομένα που χρειάζεται. Η ανάγκη αυτή καλύπτεται ορίζοντας στον server ένα connection string, το οποίο και περιέχει όλη την πληροφορία που απαιτείται ώστε να γίνει επιτυχώς η σύνδεση αυτή.

```
"ConnectionString": {
  "PetMeAppDB": "Server=DESKTOP-LA4N05A\\SQLEXPRESS;Database=PetMeDB;Trusted_Connection=True;"
},
```

Κατά την πορεία ανάπτυξης του λογισμικού, θα χρειαστεί πολλές φορές να επέμβουμε στη βάση, να δούμε τις εγγραφές που έχουν γραφτεί στους πίνακές της, να αλλάξουμε ή και να σβήσουμε μερικές από αυτές για διάφορους λόγους. Η δυνατότητα αυτή μας δίνεται μέσα από ένα λογισμικό που γίνεται διαθέσιμο από την Microsoft και λέγεται SQL Server Management Studio (SSMS) και το οποίο ξέρει πως να διαχειριστεί τη βάση καθώς και πως να μιλήσει σε αυτήν.

Μέσω αυτού του εργαλείου, το server-side λογισμικό μπορεί, έμμεσα, να επικοινωνήσει με τη βάση δεδομένων. Ο κώδικας που εκτελείται όταν καλείται μια action method, πολλές φορές θέλει να απευθυνθεί στη βάση αποστέλλοντας ένα query. Στην πραγματικότητα όμως το query αυτό φτάνει πρώτα στο SSMS, το οποίο το εκτελεί, λαμβάνοντας τελικά από τη βάση τα απαραίτητα δεδομένα.

Επιπλέον, έχουμε και εμείς οι ίδιοι την δυνατότητα να χρησιμοποιήσουμε έναν script editor, ο οποίος έρχεται μαζί με το SSMS, ώστε να αποστείλουμε κάποιο query στη βάση δεδομένων προκειμένου να πάρουμε πίσω δεδομένα. Τέλος, μπορούμε να έχουμε μια γραφική απεικόνιση της σχεσιακής βάσης η οποία θα μας δείχνει τους πίνακες από τους οποίους αυτή απαρτίζεται, καθώς και τις σχέσεις που συνδέουν τους πίνακές της.

Συμπεράσματα και Μελλοντικές επεκτάσεις

Μία από τις μελλοντικές επεκτάσεις της εφαρμογής θα είναι η εισαγωγή ενός μηχανισμού κατά τον οποίο ο χρήστης θα ειδοποιείται με email για την πορεία του αιτήματος υιοθεσίας που έχει καταβάλλει. Επιπλέον η εισαγωγή διαχειριστικού ρόλου αποτελεί ενισχυτικό παράγοντα στην λειτουργικότητα της εφαρμογής. Με τον τρόπο αυτό θα υπάρχει καλύτερος έλεγχος του περιεχομένου της εφαρμογής

Τέλος η ασφάλεια θα πρέπει συνεχώς να επαναξιολογείται προκειμένου η ίδια η εφαρμογή να μην είναι επιρρεπής σε εξωτερικές κακόβουλες επιθέσεις. Η εισαγωγή ενός πιο εξειδικευμένου τρόπου authentication και authorization είναι επιτακτική ανάγκη σε μια εφαρμογή που καλείται να υποστηρίξει αρκετούς χρήστες καθώς και διάφορους ρόλους.

Βιβλιογραφία

GeeksforGeeks 2019, ανακτήθηκε Σεπτέμβριο 2020, <https://www.geeksforgeeks.org/client-server-model/>

MDN Web Docs 2019, Mozilla, ανακτήθηκε Σεπτέμβριο 2020, <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>

Divine, P 2019, Medium, ανακτήθηκε Σεπτέμβριο 2020, <https://medium.com/better-programming/the-anatomy-of-an-http-request-728a469ecba9>

Anderson, R 2020, Microsoft, ανακτήθηκε Σεπτέμβριο 2020, <https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-5.0&tabs=visual-studio>

PetFinder Website, ανακτήθηκε Σεπτέμβριο 2020, < https://www.petfinder.com/>