ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη WEB εφαρμογής αναφοράς προβλημάτων στην Περιφερειακή Ενότητα Θεσσαλονίκης»

Της φοιτήτριας
Κανέλογλου Χριστίνας
Αρ. Μητρώου: 123836

Επιβλέπων Καθηγητής
Ευστάθιος Αντωνίου

Θεσσαλονίκη 2020

0

Τίτλος Δ.Ε.: Ανάπτυξη WEB εφαρμογής αναφοράς προβλημάτων στην Περιφερειακή Ενότητα Θεσσαλονίκης

Κωδικός Δ.Ε.: 19047

Ονοματεπώνυμο φοιτητή: Κανέλογλου Χριστίνα

Ονοματεπώνυμο εισηγητή: Αντωνίου Ευστάθιος

Ημερομηνία ανάληψης Δ.Ε: 19-11-2019

Ημερομηνία περάτωσης Δ.Ε.: 18-09-2020

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

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Πρόλογος

Το αντικείμενο της παρούσας πτυχιακής εργασίας είναι η αναφορά των προβλημάτων που προκύπτουν στους δημότες καθημερινά, καθώς και η αντίστοιχη επίλυση τους από τους αρμόδιους διαχειριστές του εκάστοτε Δήμου. Μείζον κομμάτι της πτυχιακής εργασίας αποτελεί και ο στατιστικός καταμερισμός των προβλημάτων σε σχέση με το σύνολο τους και τον χρόνο.

Η συγκεκριμένη web εφαρμογή που χωρίζεται σε δύο μέρη, στο διαχειριστικό κομμάτι και στη σελίδα αναφοράς, αφενός δίνει τη δυνατότητα στους δημότες να θίγουν τα προβλήματα με τα οποία έρχονται αντιμέτωποι καθημερινά, και αφετέρου διευκολύνει τους αρμόδιους κάθε Δήμου ώστε να προβούν στην ανασκόπηση και εν συνεχεία στην επίλυση τους, καθώς πλέον τα προβλήματα παρουσιάζονται πιο συνοπτικά και κατηγοριοποιημένα.

Βασιζόμενη στα παραπάνω, η web εφαρμογή πληροί προϋποθέσεις ευχρηστίας και κοινωνικής αλληλεγγύης που αποτελούν θεμέλια μιας ισορροπημένης κοινωνίας. Η πρακτική εφαρμογή αυτής της εργασίας δύναται να λειτουργήσει επικουρικά με τις κοινωνικές αρετές, που οφείλουν να διέπουν κάθε κοινωνία και κρίνεται σαφές ότι η πιθανή υλοποίηση της θα αποτελούσε μια ελπιδοφορα και προοδευτική κίνηση του κράτους.

0

Περίληψη

Η παρούσα πτυχιακή εργασία αφορά την κατασκευή μιας web εφαρμογής, για την αναφορά προβλημάτων, από τους δημότες της Περιφερειακής Ενότητας της Θεσσαλονίκης και την επεξεργασία αυτών των προβλημάτων από τους αρμόδιους, του κάθε Δήμου. Πρωταρχικός στόχος της πλατφόρμας, θα είναι η γρήγορη και λεπτομερή ενημέρωση των αρμόδιων ανθρώπων ενός Δήμου, για τα τρέχοντα προβλήματα που υπάρχουν σε αυτόν, καθώς και η επίλυσή τους.

Τα βασικά στοιχεία αυτής της πτυχιακής εργασίας αφορούν κυρίως δύο μέρη. Το πρώτο, είναι το διαχειριστικό κομμάτι, στο οποίο οι αρμόδιοι (Επιβλέποντες Διαχειριστές ή Διαχειριστές του εκάστοτε Δήμου) θα έχουν την δυνατότητα να παρακολουθούν τα προβλήματα που καταχωρούνται από τους δημότες, καθώς και να επεξεργάζονται την κατάσταση τους. Επιπλέον, θα μπορούν να βλέπουν στατιστικά, που θα περιέχουν πληροφορίες όπως για παράδειγμα, τα επιλυμένα προβλήματα σε σχέση με το σύνολο και το χρόνο. Το δεύτερο, είναι η σελίδα αναφοράς προβλημάτων, στην οποία οι δημότες θα έχουν την δυνατότητα να αναφέρουν κάποιο πρόβλημα που έχουν παρατηρήσει.

Συμπερασματικά, αυτή η εφαρμογή στοχεύει στην δημιουργία ενός Δήμου με λιγότερα προβλήματα. Αυτό θα επιτευχθεί, με την συνεργασία των ανθρώπων του Δήμου καθώς και των ίδιων των δημοτών. Μέσα από αυτήν την συνεργασία, θα υπάρξει βελτίωση στην καθημερινότητα όλων μας.

«Development of a WEB application for reporting problems in the Regional Unit of Thessaloniki»

«Christina Kaneloglou»

Abstract

The current thesis concerns the construction of a web application, for the reporting of problems, by the citizens of the Regional Unit of Thessaloniki and the elaboration of these problems by the competent, of each Municipality. The primary goal of the platform will be the fast and detailed information of the competent people of a Municipality, about the current problems that exist in it, as well as their solution.

This dissertation focuses on two main parts. The first one is the management part, in which the managerial staff (Supervising Administrators or Administrators of each Municipality) will have the opportunity to monitor the problems registered by the citizens, as well as to process their situation. In addition, they will be able to view analytics, which will contain information such as, for example, the solved problems in relation to the total and time. The second is the problem reporting page, where citizens will be able to report any problems they have noticed.

In conclusion, this application aims to create a Municipality with fewer problems. This will be achieved, with the cooperation of the people of the Municipality as well as the citizens themselves. Through this cooperation, there will be an improvement in our own daily life.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Τεχνολογίες που χρησιμοποιήθηκαν

1.1 Εισαγωγή

Στο παρόν κεφάλαιο θα γίνει μία αναλυτική περιγραφή των τεχνολογιών και προγραμμάτων που χρησιμοποιήθηκαν κατά την ανάπτυξη του προγράμματος, προκειμένου να γίνει κατανοητό στον αναγνώστη τόσο το θεωρητικό, όσο και το πρακτικό μέρος.

1.2 Τεχνολογίες

1.2.1 HTML

Η HTML (Hypertext Markup Language) είναι μία γλώσσα σήμανσης υπερκειμένου που χρησιμοποιείται, για έγγραφα που έχουν σχεδιαστεί για προβολή σε πρόγραμμα περιήγησης στο Web.

Η HTML γράφεται υπό μορφή στοιχείων HTML τα οποία αποτελούνται από ετικέτες (tags), οι οποίες περικλείονται μέσα σε σύμβολα «μεγαλύτερο από» και «μικρότερο από» (για παράδειγμα <html>), μέσα στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες HTML συνήθως λειτουργούν ανά ζεύγη (για παράδειγμα <h1> και </h1>), με την πρώτη να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης (ή σε άλλες περιπτώσεις ετικέτα ανοίγματος και ετικέτα κλεισίματος αντίστοιχα). Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ.

Το παρακάτω (κλασικό) παράδειγμα “Hello World!” είναι γραμμένο σε κώδικα HTML:

```
<!DOCTYPE html>
<html>
  <head>
      <title>This is a title</title>
  </head>
  <body>
      <div>
      <p>Hello world!</p>
      </div>
  </body>
</html>
```

Το κείμενο μεταξύ <html> και </html> περιγράφει την ιστοσελίδα και το κείμενο μεταξύ <body> και </body> είναι το ορατό περιεχόμενο της σελίδας. Το κείμενο σήμανσης <title>This is a title</title> ορίζει τον τίτλο της σελίδας του προγράμματος περιήγησης και η ετικέτα ορίζει ένα τμήμα της σελίδας που χρησιμοποιείται για εύκολο στυλ.

Τα στοιχεία της HTML χρησιμοποιούνται για να χτίσουν όλους τους ιστότοπους. Η HTML επιτρέπει την ενσωμάτωση εικόνων και άλλων αντικειμένων μέσα στη σελίδα, και μπορεί να χρησιμοποιηθεί για να εμφανίσει διαδραστικές φόρμες. Παρέχει τις μεθόδους δημιουργίας δομημένων εγγράφων (δηλαδή εγγράφων που αποτελούνται από το περιεχόμενο που μεταφέρουν και από τον κώδικα μορφοποίησης του περιεχομένου) καθορίζοντας δομικά σημαντικά στοιχεία για το κείμενο, όπως κεφαλίδες, παραγράφους,

λίστες, συνδέσμους, παραθέσεις και άλλα. Μπορούν επίσης να ενσωματώνονται σενάρια εντολών σε γλώσσες όπως η JavaScript, τα οποία επηρεάζουν τη συμπεριφορά των ιστοσελίδων HTML.

Ο σκοπός ενός web browser είναι να διαβάζει τα έγγραφα HTML και να τα συνθέσει σε σελίδες που μπορεί κανείς μα διαβάσει. Επιπλεόν, οι web browsers μπορούν να αναφέρονται σε στυλ μορφοποίησης CSS για να ορίζουν την εμφάνιση και τη διάταξη του κειμένου και του υπόλοιπου υλικού. Ο οργανισμός W3C, ο οποίος δημιουργεί και συντηρεί τα πρότυπα για την HTML και τα CSS, ενθαρρύνει τη χρήση των CSS αντί διαφόρων στοιχείων της HTML για σκοπούς παρουσίασης του περιεχομένου.

1.2.2 CSS

Η CSS (Cascading Style Sheets-Διαδοχικά Φύλλα Στυλ) ή (αλληλουχία φύλλων στυλ) είναι μια γλώσσα υπολογιστή που ανήκει στην κατηγορία των γλωσσών φύλλων στυλ, που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου, που έχει γραφτεί με μια γλώσσα σήμανσης.

Χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε στις γλώσσες HTML και XHTML, δηλαδή για τον έλεγχο της εμφάνισης μιας ιστοσελίδας και γενικότερα ενός ιστοτόπου. Η CSS είναι μια γλώσσα υπολογιστή προορισμένη να αναπτύσσει στυλιστικά μια ιστοσελίδα δηλαδή να διαμορφώνει περισσότερα χαρακτηριστικά, χρώματα, στοίχιση και να δίνει περισσότερες δυνατότητες, σε σχέση με την html. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται ως απαραίτητη.

Το CSS έχει σχεδιαστεί για να καθορίζει τη μορφοποίηση των εγγράφων υπερκειμένου. Αυτό περιλαμβάνει τα χρώματα, τις γραμματοσειρές, τα στυλ και τις διατάξεις των πληροφοριών που εμφανίζονται σε έναν ιστότοπο. Υπάρχουν πολλοί επιπλέον λόγοι για να χρησιμοποιήσει κανείς τον κώδικα CSS για τη μορφοποίηση της εμφάνισης της ιστοσελίδας του. Θα μπορέσει να βελτιώσει την προσβασιμότητα του περιεχομένου των σελίδων του. Ένα άλλο χαρακτηριστικό είναι ότι η χρήση CSS θα του επιτρέψει να ενεργοποιήσει την ίδια μορφοποίηση πολλαπλών σελίδων του ιστότοπού του. Αυτό είναι χρήσιμο επειδή αν ένας ιστότοπος αποτελείται από περισσότερες από μία σελίδες , τότε δεν θα χρειαστεί να επαναληφθεί ολόκληρος ο κώδικας για κάθε μία από τις σελίδες αυτές. Η κάθε σελίδα μπορεί να έχει την ίδια γραμματοσειρά και μέγεθος γραμματοσειράς, και αυτό επιτυγχάνεται με τη χρήση του κώδικα CSS. Αυτό θα βοηθήσει, επίσης, να παραμείνουν τα αρχείων των σελίδων μικρά και καθαρά, έτσι ώστε να μπορούν να παραμετροποιηθούν πιο εύκολα και αποτελεσματικά.

Ένα αρχείο HTML κώδικα για έναν ιστότοπο είναι ένα πολύ περίπλοκο έγγραφο, το οποίο σε έναν άνθρωπο μπορεί να μοιάζει με εκατοντάδες γραμμές άχρηστων, τυχαίων και ανοργάνωτων γραμμάτων, αριθμών και συμβόλων. Με την χρήση του κώδικα CSS, αυτό το έγγραφο θα είναι πολύ μικρότερο, επειδή δεν θα χρειαστεί να πληκτρολογηθούν διάφοροι κώδικες για πολλές σελίδες. Αυτό θα το κάνει πολύ πιο απλό και πιο εύκολο στην υλοποίηση, και ιδιαίτερα στις περιπτώσεις που πρέπει να αναζητήσουμε και να διορθώσουμε κάποιο πρόβλημα ή να προσθέσουμε κάτι καινούριο.

Οι προδιαγραφές του CSS τηρούνται και ενημερώνονται από το W3C. Αυτό σημαίνει ότι το W3C είναι υπεύθυνο για τη δημιουργία των προτύπων για τους κανόνες που πρέπει να ακολουθούνται για την συγγραφή κώδικα. Το W3C είναι διεθνές, ώστε οι ιστότοποι από οποιαδήποτε χώρα, θα ακολουθήσουν το ίδιο σύνολο κανόνων. Αυτός είναι και ο λόγος για τον οποίο το Διαδίκτυο ονομάζεται World Wide Web. Το W3C αποτελείται από έναν αριθμό οργανισμών-μελών που είναι υπεύθυνοι για τη δημιουργία των προτύπων για τις διαδικασίες κωδικοποίησης του World Wide Web και αποτελείται από περίπου 400 μέλη.

Ο κώδικας CSS είναι πολύ πιο κατανοητός όταν το διαβάζετε ως άνθρωπος, επειδή χρησιμοποιεί κανονικές αγγλικές λέξεις για την επισήμανση διαφορετικών ετικετών και τμημάτων του κώδικα. Κάθε φύλλο στυλ, έχει το δικό του σύνολο κανόνων. Κάθε σύνολο κανόνων έχει τους δικούς του επιλογείς και μπλοκ δηλώσεων. Ένα μπλοκ δήλωσης, περιέχει μία λίστα με τους κανόνες που θα εφαρμοστούν στο στοιχείο, για το οποίο δημιουργήθηκε αυτό. Στο CSS, οι επιλογείς προορίζονται για να δηλώσουν ποια στοιχεία ισχύουν για κάθε στυλ.

Πριν από τη χρήση του CSS, σχεδόν όλα τα οπτικά χαρακτηριστικά των αρχείων HTML διατηρήθηκαν μέσα στον κώδικα HTML. Αυτό περιλαμβάνει τις ετικέτες για χρώματα γραμματοσειρών, χρώματα φόντου και μοτίβα, ευθυγραμμίσεις, αλλά και σύνορα και πράγματα αυτού του είδους. Πλέον με την χρήση του CSS αυτό δεν είναι απαραίτητο να πραγματοποιηθεί και θα βοηθήσει να κρατηθεί το αρχείο του κώδικα της σελίδας καθαρότερο ώστε ο κώδικας να μπορεί να διαβαστεί πιο εύκολα.

1.2.3 JavaScript

Η JavaScript (JS) είναι μία από τις πιο δημοφιλείς γλώσσες προγραμματισμού ηλεκτρονικών υπολογιστών στον Παγκόσμιο Ιστό (Web). Αν και η JavaScript είναι κυρίως γνωστή ως γλώσσα ιστοσελίδων, χρησιμοποιείται και σε άλλες εφαρμογές εκτός αυτών, όπως PDF, Node.js, Apache CouchDB και Adobe Acrobat. Επιπλέον, η JavaScript έχει γίνει δημοφιλής και στην ανάπτυξη εφαρμογών ιστού από την πλευρά του διακομιστή (server-side). Ένα τέτοιο παράδειγμα είναι το Node.js, το οποίο θα αναλυθεί παρακάτω.

Η σύνταξή της JavaScript είναι επηρεασμένη από τη C, όπως και η Java, και είναι κι οι δύο αντικειμενοστραφείς γλώσσες προγραμματισμού. Επιπλέον, η JavaScript αντιγράφει πολλά ονόματα και συμβάσεις ονοματοδοσίας από τη Java. Παρ' όλα αυτά δεν θα πρέπει σε καμία περίπτωση να συγχέεται με την Java. Τονίζεται πως ο σωστός τρόπος γραφής είναι "Javascript" και όχι 'Java script' σαν δύο λέξεις, όπως λανθασμένα γράφεται ορισμένες φορές.

Λίγα λόγια για την ιστορία της JavaScript. Η συγκεκριμένη γλώσσα προγραμματισμού δημιουργήθηκε αρχικά από τον Brendan Eich της εταιρείας Netscape με την επωνυμία Mocha. Στη συνέχεια, μετονομάστηκε σε LiveScript, και τελικά πήρε το όνομα JavaScript, κυρίως επειδή η ανάπτυξή της επηρεάστηκε περισσότερο από τη γλώσσα προγραμματισμού Java. Αρχικά χρησιμοποιήθηκε για προγραμματισμό από την πλευρά του πελάτη (client) και χαρακτηρίστηκε σαν client-side γλώσσα προγραμματισμού. Στη συνέχεια όμως χρησιμοποιήθηκε και για συγγραφή κώδικα από την πλευρά του διακομιστή (server).

Το παρακάτω (κλασικό) παράδειγμα "Hello World!" είναι γραμμένο σε κώδικα JavaScript:

```
<script type="text/javascript">
alert('Γεια σου, κόσμε!');
</script>
```

1.2.3.1 jQuery

Η jQuery είναι μία μικρή, γρήγορη και πλούσια σε χαρακτηριστικά βιβλιοθήκη της JavaScript. Έχει σχεδιαστεί για να απλοποιήσει πολλά πράγματα στην πλευρά του πελάτη (client-side), όπως

διασταύρωση και χειραγώγηση εγγράφων HTML, διαχείριση συμβάντων, κινούμενα σχέδια και Ajax, και είναι συμβατή σε πολλούς φυλλομετρητές ιστού. Κυκλοφόρησε τον Ιανουάριο του 2006 από τον John Resig και με ένα συνδυασμό ευελιξίας και επεκτασιμότητας, άλλαξε τον τρόπο με τον οποίο εκατομμύρια άνθρωποι γράφουν JavaScript.

Για να φορτωθεί η συγκεκριμένη βιβλιοθήκη σε μια ιστοσελίδα μπορεί να γίνει είτε παρέχοντας τοπικά το αρχείο είτε μέσω ενός συνδέσμου από τους πολλούς διακομιστές που την φιλοξενούν.

Ακολουθεί ένα παράδειγμα σύγκρισης κώδικα JavaScript και jQuery, το οποίο έχει ως αποτέλεσμα την αλλαγή του κειμένου ενός κουμπιού.

```
/*jQuery*/
$( "button.continue" ).html( "Continue" );
/*javaScript*/
document.querySelector('button.continue').innerHTML = Continue;
```

1.2.3.2 Backbone.js

Το Backbone.js είναι μία βιβλιοθήκη της JavaScript, η οποία βασίζεται στο πρότυπο σχεδιασμού εφαρμογής Model-view-presenter (MVP). Δίνει δομή σε εφαρμογές ιστού παρέχοντας μοντέλα (models), συμβάντα (events), συλλογές (collections) με ένα API πλούσιο σε λειτουργίες, views με χειρισμό συμβάντων, και όλα αυτά τα συνδέει με το υπάρχον API της εφαρμογής μέσω διεπαφής RESTful JSON.

Το πρότζεκτ βρίσκεται στο GitHub κι είναι διαθέσιμο για χρήση με την άδεια λογισμικού MIT. Η μόνη προϋπόθεση για να ολοκληρωθεί η εγκατάσταση του είναι η ύπαρξη της βιβλιοθήκης Underscore.js (>= 1.8.3).

Κατά την διάρκεια δημιουργίας μιας web εφαρμογής, η οποία περιλαμβάνει πολλά Javascript αρχεία, είναι πολύ εύκολο να δημιουργηθεί σύγχυση μεταξύ των διαφόρων κλήσεων σε μεθόδους. Με αυτόν τον τρόπο, δεν θα έχουμε το επιθυμητό αποτέλεσμα, καθώς πολλά από τα δεδομένα που θα έχουν παραχθεί δεν θα είναι συγχρονισμένα με την διεπαφή του χρήστη HTML και την βάση δεδομένων. Για αυτό τον λόγο, όταν έχουμε να αντιμετωπίσουμε την δημιουργία μιας πιο πλούσιας εφαρμογής είναι καλό να χρησιμοποιούμε πιο δομημένες προσεγγίσεις.

Μια τέτοια προσέγγιση, είναι το Backbone.js, το οποίο αναπαριστά τα δεδομένα ως Μοντέλα, τα οποία μπορούν να δημιουργηθούν, να επικυρωθούν, να καταστραφούν και να αποθηκευτούν στο διακομιστή. Με αυτόν τον τρόπο, κάθε φορά που κάποιος χρήστης προβεί σε μία ενέργεια, η οποία με την σειρά της θα προκαλέσει κάποια αλλαγή στο μοντέλο, αυτόματα ειδοποιείται το αντίστοιχο view που περιλαμβάνει την διαχείριση αυτής. Αυτό έχει ως αποτέλεσμα, το view να προβεί στις κατάλληλες ενέργειες, έτσι ώστε να παραχθούν τα νέα δεδομένα, να ενημερωθεί το μοντέλο και ταυτόχρονα να ενημερωθεί αυτόματα και το αντίστοιχο HTML αρχείο.

Ουσιαστικά, αυτό που θέλει να καταφέρει το Backbone.js είναι να διατηρήσει την λογική της επιχείρησης ξεχωριστή από την διεπαφή του χρήστη και με αυτόν τον τρόπο να διευκολύνει τις αλλαγές στο πρότζεκτ.

1.2.3.3 Node.js

Το Node.js είναι μία πλατφόρμα ανάπτυξης λογισμικού κυρίως στην πλευρά του διακομιστή, χρησιμοποιώντας την γλώσσα προγραμματισμού JavaScript. Κύριο χαρακτηριστικό του είναι η ασύγχρονη επικοινωνία μεταξύ των υπολογιστικών πόρων, η οποία επιτυγχάνεται με την χρήση events που προσφέρει η JavaScript.

Το Node.js δημιουργήθηκε το 2009 από τον Ryan Dahl. Η ιδέα για την ανάπτυξη του *node* προήλθε από την ανάγκη του ίδιου να βρει τον πιο αποδοτικό τρόπο να ενημερώνει τον χρήστη σε πραγματικό χρόνο για την κατάσταση ενός αρχείου που ανέβαζε στο διαδίκτυο. Στην αρχή χρησιμοποίησε αρκετές γλώσσες προγραμματισμού όπως C, Lua, Haskell, αλλά το αποτέλεσμα δεν ήταν το επιθυμητό. Στο τέλος, κατέληξε να ασχοληθεί με την JavaScript, εξαιτίας της κυκλοφορίας της μηχανής V8 (V8 JavaScript Engine) της Google.

Ακολουθεί ένα απλό παράδειγμα node για έναν απλό HTTP εξυπηρετή, το οποίο έχει ως αποτέλεσμα την εμφάνιση μιας σελίδας, με κείμενο το «Hello World» , στην διεύθυνση "http://127.0.0.1:1337.

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
console.log('Server running at http://127.0.0.1:1337/');
```

1.2.4 Firebase

Το Firebase είναι μία πλατφόρμα, η οποία έχει αναπτυχθεί από την Google, και έχει ως σκοπό την δημιουργία εφαρμογών για κινητά (Android, iOS) και ιστούς (Web). Ιδρύθηκε το 2011 ως ανεξάρτητη εταιρεία. Από το 2014 όμως, ανήκει στην Google και πλέον είναι κορυφαία στην ανάπτυξη εφαρμογών.

Το Firebase παρέχει προϊόντα για να αναπτυχθούν εφαρμογές υψηλής ποιότητας. Σαν σύνολο διαθέτει 20 προιόντα τα οποία είναι χωρισμένα σε τρεις κατηγορίες: Develop, Quality, and Grow. Μερικά από αυτά, που ανήκουν στην κατηγορία Develop κι έχουν χρησιμοποιηθεί στην παρούσα πτυχιακή εργασία, είναι τα εξής: Cloud Firestore (είναι μια ευέλικτη, επεκτάσιμη βάση δεδομένων), Authentication (παρέχει backend υπηρεσίες για τον έλεγχο της ταυτότητας των χρηστών σε μία εφαρμογή), Cloud Storage (είναι μία υπηρεσία που χρησιμοποιείται για την αποθήκευση περιεχομένου, όπως φωτογραφίες και βίντεο).

1.3 Προγραμματιστικά Περιβάλλοντα

1.3.1 IntelliJ IDEA

Το IntelliJ IDEA είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού γραμμένο σε Java και η πρώτη έκδοση του κυκλοφόρησε τον Ιανουάριο του 2001. Περιλαμβάνει υποστήριξη για εντοπισμό σφαλμάτων, ενσωματωμένο έλεγχο για Git, διάφορες επισημάνσεις και χρώματα ανάλογα με τη γλώσσα προγραμματισμού, βοηθό για τυχόν λάθη και επεξηγήσεις του κώδικα και διάφορα πρόσθετα (plugins). Είναι επίσης πολύ προσαρμόσιμο στις ανάγκες του εκάστοτε χρήστη, έχοντας τη δυνατότητα να

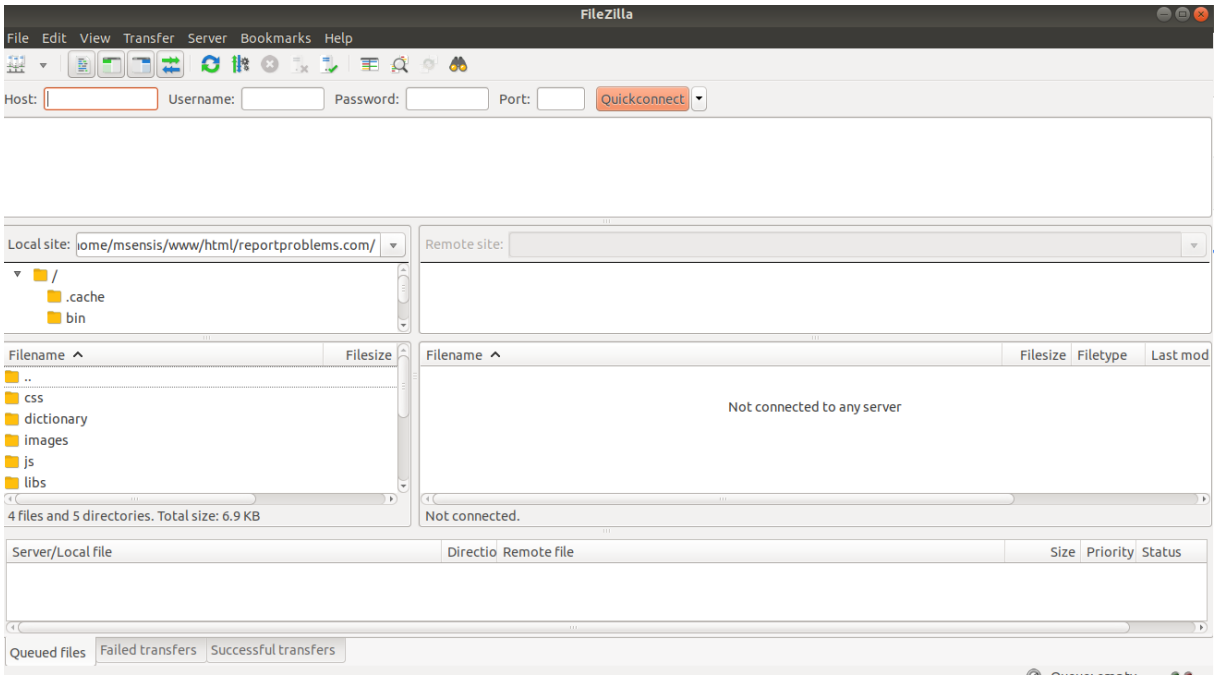
αλλάξουν τα χρώματα και το θέμα του προγράμματος, τις συντομεύσεις πληκτρολογίου, αλλά και πολλές άλλες ρυθμίσεις και προτιμήσεις.

Ακολουθεί μία εικόνα που παρουσιάζει την κύρια οθόνη του.

Εικόνα 1: IntelliJ IDEA περιβάλλον

1.3.2 FileZilla

Πρόκειται για έναν FTP πελάτη τον οποίο μπορούμε να χρησιμοποιήσουμε για να ανεβάσουμε αρχεία σε κάποιον εξυπηρετητή. Υπάρχουν διαθέσιμες εκδόσεις για Linux, Windows και MacOS. Ο πελάτης υποστηρίζει FTP, SFTP και FTPS (FTP μέσω SSL / TLS). Είναι ένα ελεύθερο λογισμικό [4] και το περιεχόμενο της αρχικής οθόνης του φαίνεται στη παρακάτω εικόνα.



Εικόνα 2: FileZilla περιβάλλον

1.3.3 BitBucket

Για να διευκολύνουμε την διαδικασία κατασκευής του προγράμματος χρησιμοποιήθηκε το BitBucket ένα version control system που ανήκει στην Atlassian από το 2014. Σύμφωνα με πρόσφατα στατιστικά στο BitBucket υπάρχουν πάνω από 10 εκατομμύρια χρήστες και πάνω από 28 εκατομμύρια repositories καθιστώντας το ένα από τους μεγαλύτερους παρόχους Open Source λογισμικού στον κόσμο.

Το BitBucket προσφέρει δωρεάν λογαριασμούς με απεριόριστο αριθμό private repositories. Με αυτόν τον τρόπο παρέχεται η δυνατότητα αποθήκευσης ιδιωτικά του κώδικα κι η καλύτερη οργάνωση του, η παρακολούθηση των αλλαγών που γίνονται σε αυτον κι η εύκολη πρόσβαση του.

1.3.4 NGINX

Ο nginx είναι ένας υψηλής απόδοσης web server, ο οποίος μπορεί επίσης να χρησιμοποιηθεί ως αντίστροφος διακομιστής μεσολάβησης, καθώς και διακομιστής μεσολάβησης IMAP / POP3. Δημιουργήθηκε το 2004 από τον Igor Sysoev και παρέχεται δωρεάν.

Το NGINX είναι γνωστό για την υψηλή απόδοση, τη σταθερότητα, το πλούσιο σύνολο χαρακτηριστικών, την απλή διαμόρφωση και τη χαμηλή κατανάλωση πόρων.

1.4 Επίλογος

Στο παρόν κεφάλαιο έγινε μια προσπάθεια να περιγραφούν συνοπτικά όλες οι τεχνολογίες που χρησιμοποιήθηκαν, καθώς και τα προγραμματιστικά περιβάλλοντα, για την ανάπτυξη της εφαρμογής. Ο αναγνώστης σε αυτό το σημείο θα πρέπει να είναι έτοιμος για να συνεχίσουμε στην λεπτομερή εξήγηση της εφαρμογής.

Κεφάλαιο 2ο: Περιγραφή Συστήματος

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει η περιγραφή της εφαρμογή που αναπτύχθηκε στο πλαίσιο της παρούσας πτυχιακής εργασίας. Η εφαρμογή χωρίζεται ουσιαστικά σε δύο κομμάτια, στο κομμάτι του Front-End και στο κομμάτι του Back-End. Παρακάτω, θα περιγραφούν λεπτομερώς τα κύρια σημεία από τα οποία αποτελείται το κάθε κομμάτι.

2.2 FRONT-END

Το Front-End αναφέρεται ουσιαστικά στο κομμάτι του ιστού που μπορεί να δει και να αλληλεπιδράσει ένας χρήστης. Η αναπτυξη της παρούσας εφαρμογής στηρίζεται στο Backbone.js, που είναι μία βιβλιοθήκη της JavaScript όπως έχεις περιγραφεί πιο πάνω. Πιο συγκεκριμένα, από τη συγκεκριμένη βιβλιοθήκη έχουν χρησιμοποιηθεί τα views, routers και templates, τα οποία θα αναλυθούν λεπτομερώς στη συνέχεια.

2.2.1 Λειτουργία

Για την έναρξη της εφαρμογής υπάρχει ένα βασικό αρχείο, το οποίο στην συγκεκριμένη περίπτωση έχουμε ονομάσει main.js και φορτώνεται στο αρχείο index.html. Το index.html ως γνωστόν είναι η σελίδα που εμφανίζεται όταν ένας επισκέπτης ζητά πρόσβαση στον ιστότοπο. Σε περίπτωση που αυτό το αρχείο δεν φορτωθεί, η εφαρμογή δεν μπορεί να ξεκινήσει. Από αυτό και μόνο, μπορεί να καταλάβει κάποιος της σημαντικότητα του.

Παρακάτω, εμφανίζεται το <body> του αρχείου index.html το οποίο χρησιμοποιεί η εφαρμογή.

```
<body id="body">
     <div id="main-wrapper" class="show">
     <div class="nav-header custom-gradient-1"></div>

     <header id="header" class="header"></header>

     <aside id="navbar-side" class="nk-sidebar"></aside>

     <main id="container" class="content-body"></main>

     <footer class="footer"></footer>

     </div>

     <footer></footer>

     <script src="libs/jquery/jquery-3.4.1.min.js"></script>
     <script src="https://cdn.jsdelivr.net/npm/apexcharts"></script>
     <script src="js/main.js"></script>
</body>
```

Στο <header> του παραπάνω κώδικα με id="header", φορτώνεται το header από το διαχειριστικό κομμάτι. Στο <aside> με id="navbar-side", φορτώνεται το μενού πλοήγησης που παρατηρειται στο αριστερό κομμάτι της εφαρμογής. Το <main> με id="container" είναι το σημείο, στο οποίο φορτώνονται τα υπόλοιπα templates της εφαρμογής. Ουσιαστικά είναι το κομμάτι της εφαρμογής, το οποίο αλλάζει διαρκώς καθώς ο χρήστης επιλέγει να αλλάξει οθόνη. Τέλος, στο κάτω μέρος του body, παρατηρούμε ότι φορτώνεται το αρχείο main.js.

Στο αρχείο main.js φορτώνονται κάποιες βιβλιοθήκες που χρησιμοποιεί η εφαρμογή. Επιπλέον, δημιουργείται το configuration του Firebase, στο οποίο ουσιαστικά βασίζεται το Back-End της εφαρμογής, και γίνεται η αρχικοποίηση του. Το πιο σημαντικό όμως είναι ότι το main.js φορτώνει τον router, ο οποίος διαχειρίζεται όλα τα request.

2.2.2 Router

Ο Router του Backbone.js μας παρέχει έναν μηχανισμό, με τον οποίο μας δίνεται η δυνατότητα να αντιγράψουμε τις διευθύνσεις URL και να τις χρησιμοποιήσουμε για να φτάσουμε στην δημιουργία του View, που θέλουμε να χρησιμοποιήσουμε τη δεδομένη στιγμή.

Παρακάτω παρουσιάζεται ένα απλό παράδειγμα ενός router:

```
var TheRouter = Backbone.Router.extend({
    routes: {
       "": "defaultRoute",
       "Login": "loginRoute",
      .......
    },
    loggedIn: false,
    userData: null,
    moreUserData: null,
    initialize: function (options) {
       self = this;

       this.loggedIn = options.loggedIn;
       if(options.data)
      {
            this.userData = options.data;
      }

      if(options.userDetails)
      {
            this.moreUserData = options.userDetails;
      }

    },
    defaultRoute: function () {
      ...
    },
```

```
    loginRoute: function () {
      ...
      this.view = new LoginView();
      },
});
module.exports = TheRouter;
```

Στον παραπάνω κώδικα, βλέπουμε ένα block με όνομα routes, το οποίο περιέχει όλα τα routes της εφαρμογής, δηλαδή όλες τις σελίδες που θα έχει πρόσβαση ο χρήστης. Η δομή ενός στοιχείου του routes είναι η εξής: "routeName": "routeFunction". Πιο συγκεκριμένα, στα αριστερά αναφέρεται το όνομα του συγκεκριμένου route και στα δεξιά η function που θα εκτελέσει το συγκεκριμένο route, στην οποία θα δημιουργηθεί και το αντίστοιχο view. Τέλος, το μόνο που πρέπει να κάνουμε για να ενεργοποιήσουμε το συγκεκριμένο route είναι να αντικαταστήσουμε το URL στον browser με το “https://examplepage.com/#routeName”.

Αφού, έγινε περιγραφή ενός router, μπορούμε να προχωρήσουμε στην ανάλυση των router της παρούσας εφαρμογής. Όλοι οι routers βρίσκονται μέσα στον φάκελο /src/main/web/js/routers.

2.2.2.1 main.router.ts

Ο συγκεκριμένος Router διαχειρίζεται τις σελίδες που έχουν σχέση με το Διαχειριστικό κομμάτι της εφαρμογής. Παρακάτω αναφέρονται όλες οι πιθανές διαδρομές:

- "": "defaultRoute"
- "Login": "loginRoute"
- "ForgotPassword": "forgotPasswordRoute"
- "Home": "homeRoute"
- "Problems": "problemsRoute"
- "Problem(/:id)": "problemRoute"
- "Analytics": "analyticsRoute"
- "Users": "usersRoute"
- "User(/:id)": "userRoute"
- "Profile": "profileRoute"

2.2.2.2 reportProblem.router.ts

Ο συγκεκριμένος Router διαχειρίζεται τις σελίδες που έχουν σχέση με το κομμάτι Αναφοράς Προβλήματος της εφαρμογής. Παρακάτω αναφέρονται όλες οι πιθανές διαδρομές:

- "": "defaultRoute"
- "ReportProblemMainPage": "reportProblemMainPageRoute"
- "ReportProblem": "reportProblemRoute"

2.2.3 Views

Στο Backbone.js τα views μας δίνουν την δυνατότητα να διαχειριζόμαστε πιο εύκολα εφαρμογές μεγάλης κλίμακας. Η γενική ιδέα είναι, να οργανώσουμε την εφαρμογή μας σε λογικά views, που θα διαχειρίζονται κάποια συγκεκριμένα HTML αρχεία. Κατά κάποιο τρόπο, μπορούμε να πούμε ότι τα views είναι υπεύθυνα για την διαχείριση των συμβάντων, που λαμβάνουν χώρα όταν ο χρήστης κάνει κάποια ενέργεια στο UI της εφαρμογής. Δηλαδή, είναι αυτά που θα λάβουν τις κατάλληλες ενέργειες στην πυροδότηση ενός τέτοιου συμβάντος, π.χ. όταν πατηθεί κάποιο κουμπί, το view θα εκτελέσει ένα σύνολο εντολών, έτσι ώστε να υπάρξει το κατάλληλο αποτέλεσμα.

Παρακάτω παρουσιάζεται ένα απλό παράδειγμα ενός view:

```
require("../../libs/backbone/backbone.ajaxq.adapter.ts");
var Handlebars = require('handlebars');
var Template = require('text-
loader!../../js/templates/template.template.html');

var sampleView = Backbone.View.extend({
    el: $("#body"),
    tagName: '',
    template: Handlebars.compile(Template),
    initialize: function (options) {
        …
        this.render();

    },
    superRender: Backbone.View.prototype.render,
    render: function () {
        this.superRender();
        this.$el.html(this.template());
    },
    events: {
        "click .item": "itemClicked",
        ...
      },
      itemClicked: function () {
          alert('add code here!');
      }
});
```

Στον παραπάνω κώδικα, παρατηρούμε ότι στην αρχή κάνουμε require κάποιες βιβλιοθήκες που θα χρησιμοποιηθούν στο συγκεκριμένο view, όπως τους handlebars, καθώς επίσης και το αρχείο html που συνδέεται με το συγκεκριμένο view.

Στη συνέχεια, δημιουργούμε ένα view με όνομα sampleView, το οποίο είναι επέκταση της υπάρχουσας κατηγορίας View του Backbone.js . Μέσα στο view διακρίνουμε ένα el property, το οποίο ουσιαστικά είναι το DOM element, στο οποίο θα συμβαίνουν όλα τα events. Στο tagName property, που υπάρχει ακριβώς από κάτω, μπορούμε να ορίσουμε το είδος του tag που είναι το el, π.χ. div .

Στο template αποθηκεύουμε το Template που έχουμε κάνει πιο πάνω require, αφού πρώτα το κάνουμε compile με την handlebars. Η compile παίρνει το template μαζί με τα ειδικά tags που χρησιμοποιούμε και το μετατρέπει σε JS, έτσι ώστε να μπορούν να περαστούν μετά μεταβλητές και δεδομένα αλλά και να τρέξουν συνθήκες, επαναλήψεις μέσα σε αυτό.

Ακολουθεί, όπως βλέπουμε, η initialize. Ουσιαστικά κάνουμε override την initialize συνάρτηση του view, κι εκεί μέσα εκτελούμε την render. Μέσα στην render, γίνεται η ενημέρωση του el property με το template, στο οποίο έγινε αναφορά πιο πάνω. Στο template μπορούμε να περάσουμε παραμετρικά, τα δεδομένα που θέλουμε να εμφανίσουμε μέσα στο template, με τον εξής τρόπο this.$el.html(this.template({“data”: myData}));

Τέλος, ένα σημαντικό κομμάτι ενός view, είναι αυτό της διαχείρισης των events. Όπως παρατηρούμε, υπάρχει ένα block με όνομα events. Το backbone διαβάζει αυτό το block και κάνει bind τα αντίστοιχα events που αναφέρονται. Ένα event μπορεί να δηλωθεί με τον αντίστοιχο τρόπο: {"event selector": "callback"}. Στα αριστερά, διακρίνουμε το event σε κάποιο συγκεκριμένο element, ενώ στα δεξιά το όνομα της function που θα κληθεί.

Αφού, έγινε περιγραφή των βασικών λειτουργιών ενός view, μπορούμε να προχωρήσουμε στην ανάλυση των views της παρούσας εφαρμογής. Όλα τα views βρίσκονται μέσα στον φάκελο /src/main/web/js/views.

2.2.3.1 login.view.ts

Το login.view.ts φορτώνει το login.template.html στο body και είναι υπεύθυνο για την είσοδο του χρήστη στην εφαρμογή.

Κύρια function του συγκεκριμένου view είναι η **login** και εκτελείται όταν ο χρήστης πατήσει το κουμπί Σύνδεση. Στη login αποθηκεύουμε σε δύο μεταβλητές το email και τον κωδικό που έχει εισάγει ο χρήστης στα αντίστοιχα πεδία. Στη συνέχεια, περνάμε αυτές τις μεταβλητές σαν παράμετρο στην firebase.auth().signInWithEmailAndPassword(), η οποία είναι η κατάλληλη μέθοδος του Firebase για να διαχειριστεί την είσοδο του χρήστη στην εφαρμογή. Σε περίπτωση επιτυχίας, θα εκτελεστούν οι εντολές που βρίσκονται μέσα στο then κι επομένως θα φορτωθεί η αρχική σελίδα της εφαρμογής. Σε αντίθετη περίπτωση, θα εκτελεστούν οι εντολές που βρίσκονται μέσα στην catch και έτσι θα εμφανιστεί κατάλληλο μήνυμα σφάλματος στο χρήστη.

Η function **loginWithEnter**, ουσιαστικά εκτελείται κάθε φορά που ο χρήστης πληκτρολογεί μέσα στα δύο πεδία που υπάρχουν στην φόρμα Σύνδεσης. Αυτό που κάνει η συγκεκριμένη μέθοδος, είναι να ελέγχει αν το κουμπί του πληκτρολογίου, το οποίο πατήθηκε από τον χρήστη, είναι το ENTER. Σε περίπτωση που είναι, εκτελείται η login function, η οποία αναλύθηκε παραπάνω.

2.2.3.2 forgotPassword.view.ts

Το forgotPassword.view.ts φορτώνει το forgotPassword.template.html στο body και είναι υπεύθυνο για την ανάκτηση κωδικού εισόδου στην εφαρμογή, σε περίπτωση που ο χρήστης έχει ξεχάσει τον κωδικό πρόσβασης του.

Κύρια function του συγκεκριμένου view είναι η **forgot** και εκτελείται όταν ο χρήστης πατήσει το κουμπί Υποβολή της φόρμας. Στη login αποθηκεύουμε σε μία μεταβλητή το email που έχει εισάγει ο χρήστης στο αντίστοιχο πεδίο. Στη συνέχεια, περνάμε αυτή την μεταβλητή σαν παράμετρο στην firebase.auth().sendPasswordResetEmail(), η οποία είναι μία μέθοδος του Firebase. Η συγκεκριμένη μέθοδος ελέγχει αν το email υπάρχει στο Authentication του Firebase. Σε περίπτωση που υπάρχει, θα σταλεί ένα email ανάκτησης κωδικού στο email του χρήστη και θα εμφανιστεί στην οθόνη κατάλληλο μήνυμα επιτυχίας. Σε αντίθετη περίπτωση, θα εμφανιστεί στην οθόνη κατάλληλο μήνυμα σφάλματος.

Η function **forgotWithEnter**, ουσιαστικά εκτελείται κάθε φορά που ο χρήστης πληκτρολογεί μέσα στο πεδίο που υπάρχει στην φόρμα. Αυτό που κάνει η συγκεκριμένη μέθοδος, είναι να ελέγχει αν το κουμπί του πληκτρολογίου, το οποίο πατήθηκε από τον χρήστη, είναι το ENTER. Σε περίπτωση που είναι, εκτελείται η forgot function, η οποία αναλύθηκε παραπάνω.

2.2.3.3 navbar_side.view.ts

Το navbar_side.view.ts φορτώνει το navbar_side.template.html στο element με id="navbar-side", έτσι ώστε να εμφανιστεί στην οθόνη το μενού πλοήγησης της εφαρμογής.

Στο συγκεκριμένο view υπάρχει μία function με όνομα **addActive**, η οποία εκτελείται κάθε φορά που χρήστης πατάει κάποιο από τα στοιχεία του μενού. Ουσιαστικά αυτό που κάνει είναι να προσθέτει καποιους CSS κανόνες στο ενεργό στοιχείο του μενού, έτσι ώστε αυτό να ξεχωρίζει από τα υπόλοιπα.

2.2.3.4 header.view.ts

Το header.view.ts φορτώνει το header.template.html στο element με id="header", έτσι ώστε να εμφανιστεί στην οθόνη πάνω μέρος της εφαρμογής, στο οποίο εμφανίζεται και το Μενού Χρήστη.

Μοναδική function του συγκεκριμένου view είναι η **logout** και εκτελείται όταν ο χρήστης πατήσει το κουμπί Αποσύνδεση που υπάρχει στο Μενού Χρήστη. Η logout εκτελεί την firebase.auth().signOut(), η οποία είναι η κατάλληλη μέθοδος του Firebase για την έξοδο του χρήστη από την εφαρμογή. Σε περίπτωση επιτυχίας, θα εκτελεστούν οι εντολές που βρίσκονται μέσα στο then κι επομένως ο χρήστης θα πλοηγηθεί στην οθόνη Σύνδεσης της εφαρμογής. Σε αντίθετη περίπτωση, θα εκτελεστούν οι εντολές που βρίσκονται μέσα στην catch και έτσι θα εμφανιστεί κατάλληλο μήνυμα σφάλματος στο χρήστη.

2.2.3.5 navbar_header.view.ts

Το navbar_header.view.ts φορτώνει το navbar_header.template.html στο element με class="nav-header". Το συγκεκριμένο view είναι υπεύθυνο για την εμφάνιση του λογότυπου της εφαρμογής στο πάνω αριστερά κομμάτι της οθόνης.

Στο συγκεκριμένο view, υπάρχει μόνο μία function με όνομα **toggleMenu**, η οποία εκτελείται κάθε φορά που ο χρήστης πατάει το βελάκι που υπάρχει δίπλα από το λογότυπο και είναι υπεύθυνη για το κλείσιμο και άνοιγμα του μενού.

2.2.3.6 home.view.ts

Το home.view.ts φορτώνει το home.template.html στο element με id="container" και είναι υπεύθυνο για την εμφάνιση της αρχικής σελίδας.

Αρχικά στην **initialize**, γίνεται έλεγχος για τον ρόλο του χρήστη, έτσι ώστε στη συνέχεια να γίνουν οι κατάλληλες ενέργειες στην βάση και να αποθηκευτούν σε συγκεκριμένους πίνακες, μόνο τα προβλήματα

στα οποία έχει αρμοδιότητα ο συγκεκριμένος χρήστης. Πιο συγκεκριμένα, οι πίνακες αυτοί είναι οι παρακάτω:

- **pendingProblems**: σε αυτόν τον πίνακα αποθηκεύονται τα προβλήματα με κατάσταση "Σε Αναμονή"
- **editingProblems**: σε αυτόν τον πίνακα αποθηκεύονται τα προβλήματα με κατάσταση "Σε Επεξεργασία"
- **completedProblems**: σε αυτόν τον πίνακα αποθηκεύονται τα προβλήματα με κατάσταση "Ολοκληρώθηκε"
- **totalProblems**: σε αυτόν τον πίνακα αποθηκεύονται όλα τα προβλήματα.

Οι παραπάνω πίνακες χρησιμοποιούνται ως δεδομένα στα διαγράμματα και στον χάρτη που εμφανίζονται στην Αρχική οθόνη.

Για την εμφάνιση του χάρτη είναι υπεύθυνη η function **showMap.** Η συγκεκριμένη function καλεί την renderMap, που βρίσκεται στο view map.view.ts και ουσιαστικά είναι η μέθοδος, η οποία εμφανίζει τον χάρτη. Αφού εμφανιστεί ο χάρτης εκτελείται η **placeMarkersOnMap** function, η οποία καλεί την placeMarkerOnMap του map.view.ts, για κάθε πρόβλημα που υπάρχει στον πίνακα totalProblems. Με αυτόν τον τρόπο εμφανίζονται οι πινέζες στον χάρτη.

Για την εμφάνιση των διαγραμμάτων στο πάνω μέρος της Αρχικής οθόνης είναι υπεύθυνη η function **createDashboard**, η οποία με τη σειρά της καλεί τις **createColumnChart** και **createBarCharts**. Η πρώτη δημιουργεί το ραβδόγραμμα, που εμφανίζει τα προβλήματα των τελευταίων έξι μηνών. Η δεύτερη καλεί με τη σειρά της την function **createBarChart** τρεις φορές, κάθε φορά με διαφορετικά δεδομένα. Έτσι δημιουργούνται τα τρία ραβδογράμματα του δεξιού τμήματος της οθόνης.

Για την δημιουργία των παραπάνω διαγραμμάτων και της εμφάνισης των κατάλληλων δεδομένων ήταν απαραίτητη η δημιουργία κάποιων μεθόδων, οι οποίες φιλτράρουν τους πίνακες που έχουν αναφερθεί παραπάνω και επιστρέφουν έναν καινούργιο πίνακα, μόνο με τα δεδομένα που χρειαζόμαστε σε κάθε περίπτωση.

Η function **createLastMonbthsArray** παίρνει ως παράμετρο έναν ακέραιο αριθμό x, ο οποίος αντιστοιχεί σε μήνες. Ουσιαστικά, η συγκεκριμένη μέθοδος επιστρέφει έναν πίνακα με τους τελευταίους x μήνες.

Η function **createChartLabels** παίρνει ως παράμετρο έναν πίνακα με μήνες και μας επιστρέφει έναν νέο πίνακα, ο οποίος περιλαμβάνει τα λεκτικά των μηνών που εμφανίζονται στον οριζόντιο άξονα του αριστερά διαγράμματος.

Η function **data** παίρνει ως παράμετρο έναν από τους πίνακες που έχουν αναφερθεί πιο πάνω (π.χ. pendingProblems). Στη συνέχεια σε αυτόν τον πίνακα γίνονται κάποιες ενέργειες έτσι ώστε να συλλεχθούν και να αποθηκευτούν σε έναν νέο πίνακα, μόνο τα προβλήματα των τελευταίων 6 μηνών και μάλιστα χωρισμένα ανά μήνες. Ουσιαστικά, η συγκεκριμένη μέθοδος επιστρέφει έναν πίνακα αυτής της μορφής: [6, 8, 10, 12, 13, 14], όπου το 6 είναι ο αριθμός των προβλημάτων του μήνα x1, το 8 ο αριθμός των προβλημάτων του μήνα x2 κι ουτο καθεξής. Τέλός, αυτός ο πίνακας θα χρησιμοποιήθει ως data στα options του αριστερά ραβδογράμματος.

2.2.3.7 users.view.ts

Το users.view.ts φορτώνει το users.template.html στο element με id="container" και είναι υπεύθυνο για την εμφάνιση της σελίδας των χρηστών. Αρχικά στην **initialize**, αποθηκεύονται στον πίνακα **users** όλοι οι χρήστες που υπάρχουν στην βάση. Στη συνέχεια καλείται η **render** και μετά η **loadData**.

Η function loadData, είναι υπεύθυνη για την εμφάνιση των χρηστών στον πίνακα. Πιο συγκεκριμένα, αν ο πίνακας users έχει στοιχεία τότε αυτά θα εμφανιστούν στον πίνακα, σε αντίθετη περίπτωση θα εμφανιστεί ένα κατάλληλο μήνυμα.

Η function **showHideActionButton,** καλεί την αντίστοιχη μέθοδο που υπάρχει στο αρχείο generic.view.ts και είναι υπεύθυνη για την εμφάνιση και εξαφάνιση του κουμπιού διαγραφής.

Η function **editUser,** καλείται όταν ο χρήστης πατήσει το εικονίδιο επεξεργασίας, που υπάρχει σε κάθε χρήστη που εμφανίζεται στον πίνακα. Είναι υπεύθυνη για την μετάβαση του χρήστη στην αντίστοιχη σελίδα επεξεργασίας.

Η function **deleteUsers,** καλείται όταν ο χρήστης πατήσει είτε το εικονίδιο διαγραφής, που υπάρχει σε κάθε χρήστη που εμφανίζεται στον πίνακα, είτε το κουμπί διαγραφής που εμφανίζεται πάνω από τον πίνακα, όταν ο χρήστης επιλέξει κάποιον από τους χρήστες. Η συγκεκριμένη μέθοδος είναι υπεύθυνη για την διαγραφή ενός ή παραπάνω χρηστών.

Η function **showFilters** εκτελείται όταν πατηθεί το κουμπί με id="filterBtn" και είναι υπεύθυνη για την εμφάνιση κάποιων φίλτρων αναζήτησης.

Η function **refreshTable** εκτελείται όταν πατηθεί το κουμπί με id="refreshBtn" και είναι υπεύθυνη για την ανανέωση του πίνακα των χρηστών. Ουσιαστικά, ξαναγεμίζει τον πίνακα users με όλους τους χρήστες που υπάρχουν στην βάση.

Η function **showFilteringUsers** εκτελείται όταν πατηθεί το κουμπί με id="goBtn" και είναι υπεύθυνη για την εμφάνιση μόνο των χρηστών ενός συγκεκριμένου Δήμου, μετά από κατάλληλη αναζήτηση στην βάση δεδομένων και ενημέρωση του πίνακα users.

2.2.3.8 user.view.ts

Το user.view.ts φορτώνει το user.template.html στο element με id="container" και είναι υπεύθυνο για την εμφάνιση της σελίδας επεξεργασίας ενός χρήστη και δημιουργίας καινούργιου χρήστη.

Αρχικά στην **initialize**, στην μεταβλητή **userID**. αποθηκεύεται το id του χρήστη εφόσον υπάρχει. Σε περίπτωση που υπάρχει, θέτουμε την μεταβλητη **editable** σε true και στη συνέχεια αποθηκεύουμε στο **userData** τα στοιχεία του χρήστη μετά από αντίστοιχο κάλεσμα στη βάση δεδομένων. Αυτό σημαίνει ότι θα εμφανιστεί στον χρήστη η σελίδα επεξεργασίας του χρήστη, στην οποία θα εμφανίζονται όλα τα στοιχεία του. Σε αντίθετη περίπτωση, θέτουμε την μεταβλητη **editable** σε false. Άρα θα εμφανιστεί η σελίδα δημιουργίας καινούργιου χρήστη.

Η κύρια function του συγκεκριμένου view είναι η **clickOnSubmit**, η οποία εκτελείται όταν πατηθεί το κουμπί με id="submit". Η συγκεκριμένη μέθοδος είναι υπεύθυνη για την δημιουργία καινουργιου χρήστη ή την επεξεργασία ενός υπάρχων. Η διαφοροποίηση γίνεται με τον έλεγχο της μεταβλητής editable, με αντίστοιχο τρόπο όπως έχει περιγραφεί παραπάνω.

Η function **municipalitySelectShowHide**, εκτελείται κάθε φορά που αλλάζει η τιμή στο πεδίο ρόλος. Αν η τιμή είναι “Διαχειριστής Δήμου” εμφανίζεται ένα ακόμα πεδίο με τίτλο “Δήμος”, αλλιώς το πεδίο αυτό εξαφανίζεται.

2.2.3.9 problems.view.ts

Το problems.view.ts φορτώνει το problems.template.html στο element με id="container" και είναι υπεύθυνο για την εμφάνιση της σελίδας των προβλημάτων. Αρχικά στην **initialize**, αποθηκεύονται στον πίνακα **problems** όλα τα προβλήματα που υπάρχουν στην βάση και σχετίζονται με το είδος του χρήστη που είναι συνδεδεμένος. Στη συνέχεια καλείται η **render** και μετά η **loadData**.

Η function **loadData**, είναι υπεύθυνη για την εμφάνιση των προβλημάτων στον πίνακα. Πιο συγκεκριμένα, αν ο πίνακας problems έχει στοιχεία τότε αυτά θα εμφανιστούν στον πίνακα, σε αντίθετη περίπτωση θα εμφανιστεί ένα κατάλληλο μήνυμα.

Η function **showHideActionButton,** καλεί την αντίστοιχη μέθοδο που υπάρχει στο αρχείο generic.view.ts και είναι υπεύθυνη για την εμφάνιση και εξαφάνιση του κουμπιού διαγραφής.

Η function **editProblem,** καλείται όταν ο χρήστης πατήσει το εικονίδιο επεξεργασίας, που υπάρχει σε κάθε πρόβλημα που εμφανίζεται στον πίνακα. Είναι υπεύθυνη για την μετάβαση του χρήστη στην αντίστοιχη σελίδα επεξεργασίας.

Η function **deleteProblems,** καλείται όταν ο χρήστης πατήσει είτε το εικονίδιο διαγραφής, που υπάρχει σε κάθε πρόβλημα που εμφανίζεται στον πίνακα, είτε το κουμπί διαγραφής που εμφανίζεται πάνω από τον πίνακα, όταν ο χρήστης επιλέξει κάποιο από τα προβλήματα. Η συγκεκριμένη μέθοδος είναι υπεύθυνη για την διαγραφή ενός ή παραπάνω προβλημάτων.

Η function **showFilters** εκτελείται όταν πατηθεί το κουμπί με id="filterBtn" και είναι υπεύθυνη για την εμφάνιση κάποιων φίλτρων αναζήτησης.

Η function **refreshTable** εκτελείται όταν πατηθεί το κουμπί με id="refreshBtn" και είναι υπεύθυνη για την ανανέωση του πίνακα των προβλημάτων. Ουσιαστικά, ξαναγεμίζει τον πίνακα problems με όλα τα προβλήματα που υπάρχουν στην βάση.

Η function **showFilteringProblems** εκτελείται όταν πατηθεί το κουμπί με id="goBtn" και είναι υπεύθυνη για την εμφάνιση μόνο των προβλημάτων ενός συγκεκριμένου Δήμου ή με συγκεκριμένη κατάσταση ή με συνδυασμό των δύο παραπάνω, μετά από κατάλληλη αναζήτηση στην βάση δεδομένων και ενημέρωση του πίνακα problems.

2.2.3.10 problem.view.ts

Το problem.view.ts φορτώνει το problem.template.html στο element με id="container" και είναι υπεύθυνο για την εμφάνιση της σελίδας επεξεργασίας ενός προβλήματος.

Αρχικά στην **initialize**, στην μεταβλητή **problemId**. αποθηκεύεται το id του προβλήματος. Στη συνέχεια αποθηκεύουμε στο **problemData** τα στοιχεία του προβλήματος μετά από αντίστοιχο κάλεσμα στη βάση δεδομένων. Αυτό σημαίνει ότι θα εμφανιστεί στον χρήστη η σελίδα επεξεργασίας του προβλήματος, στην οποία θα εμφανίζονται όλα τα στοιχεία του.

Η κύρια function του συγκεκριμένου view είναι η **clickOnSubmit**, η οποία εκτελείται όταν πατηθεί το κουμπί με id="submit". Η συγκεκριμένη μέθοδος είναι υπεύθυνη για την ανανέωση της βάσης δεδομένων με τα καινούργια στοιχεία του προβλήματος. Σε περίπτωση επιτυχίας εμφανίζεται κατάλληλο μήνυμα επιτυχίας, και σε περίπτωση που αλλάξει η κατάσταση του προβλήματος καλείται η function sendEmail. Ενώ σε αντίθετη περίπτωση κατάλληλο μήνυμα σφάλματος.

Η function **sendEmail** είναι υπεύθυνη για την αποστολή ενός email στον χρήστη που έχει αναφέρει το συγκεκριμένο πρόβλημα.

Η function **showPhotoPreview**, εκτελείται κάθε φορά που ο χρήστης πατήσει πάνω στην φωτογραφία και είναι υπεύθυνη για την εμφάνιση της προεπισκόπησης της φωτογραφίας.

Η function **hidePhotoPreview**, εκτελείται κάθε φορά που ο χρήστης πατήσει το κουμπί με class="closeP" και είναι υπεύθυνη για την εξαφάνιση της προεπισκόπησης της φωτογραφίας.

2.2.3.11 analytics.view.ts

Το analytics.view.ts φορτώνει το analytics.template.html στο element με id="container" και είναι υπεύθυνο για την εμφάνιση της σελίδας των Αναλύσεων.

Αρχικά στην **initialize**, γίνεται έλεγχος για τον ρόλο του χρήστη, έτσι ώστε στη συνέχεια να γίνουν οι κατάλληλες ενέργειες στην βάση και να αποθηκευτούν σε συγκεκριμένους πίνακες, μόνο τα προβλήματα στα οποία έχει αρμοδιότητα ο συγκεκριμένος χρήστης. Πιο συγκεκριμένα, οι πίνακες αυτοί είναι οι παρακάτω:

- **pendingProblems**: σε αυτόν τον πίνακα αποθηκεύονται τα προβλήματα με κατάσταση “Σε Αναμονή”
- **completedProblems**: σε αυτόν τον πίνακα αποθηκεύονται τα προβλήματα με κατάσταση “Ολοκληρώθηκε”

Οι παραπάνω πίνακες χρησιμοποιούνται ως δεδομένα στα διαγράμματα που εμφανίζονται στην συγκεκριμένη οθόνη. Επιπλέον, δημιουργείται κ ο πίνακας **municipalities,** ο οποίος περιλαμβάνει όλους τους δήμους σε περίπτωση που ο χρήστης είναι Επιβλέπων Διαχειριστής, ενώ σε αντίθετη περίπτωση περιλαμβάνει μόνο τον δήμο για τον οποίο είναι αρμόδιος ο Διαχειριστής.

Για την εμφάνιση του διαγράμματος με τίτλο “Περιγραφή Προβλημάτων Τελευταίου Εξαμήνου” είναι υπεύθυνη η function **createLast6MonthsLineChart.** Για την εμφάνιση του διαγράμματος με τίτλο “Σύνολο Προβλημάτων” είναι υπεύθυνη η function **createTotalProblemsPieChart.** Ενώ για την εμφάνιση του διαγράμματος με τίτλο “Προβλήματα Τελευταίων 30 Ημερών” είναι υπεύθυνη η function **createLastMonthPieChartsForMunicipalities.**

Για την δημιουργία των παραπάνω διαγραμμάτων και της εμφάνισης των κατάλληλων δεδομένων ήταν απαραίτητη η δημιουργία κάποιων μεθόδων, οι οποίες φιλτράρουν τους πίνακες που έχουν αναφερθεί παραπάνω και επιστρέφουν έναν καινούργιο πίνακα, μόνο με τα δεδομένα που χρειαζόμαστε σε κάθε περίπτωση.

Η function **createLastMonbthsArray** παίρνει ως παράμετρο έναν ακέραιο αριθμό x, ο οποίος αντιστοιχεί σε μήνες. Ουσιαστικά, η συγκεκριμένη μέθοδος επιστρέφει έναν πίνακα με τους τελευταίους x μήνες.

Η function **createChartLabels** παίρνει ως παράμετρο έναν πίνακα με μήνες και μας επιστρέφει έναν νέο πίνακα, ο οποίος περιλαμβάνει τα λεκτικά των μηνών που εμφανίζονται στον οριζόντιο άξονα του αριστερά διαγράμματος.

Η function **last30DaysProblems** παίρνει ως παράμετρο έναν πίνακα με προβλήματα και επιστρέφει έναν καινούργιο πίνακα, ο οποίος περιλαμβάνει μόνο τα προβλήματα των τελευταίων 30 ημερών

Η function **getProblemsByMunicipality** παίρνει ως παράμετρο έναν πίνακα με προβλήματα και το όνομα ενός δήμου. Στη συνέχεια επιστρέφει έναν καινούργιο πίνακα, ο οποίος περιλαμβάνει μόνο τα προβλήματα που σχετίζονται με τον συγκεκριμένο δήμο.

Η function **data** παίρνει ως παράμετρο έναν από τους πίνακες που έχουν αναφερθεί πιο πάνω (π.χ. pendingProblems). Στη συνέχεια σε αυτόν τον πίνακα γίνονται κάποιες ενέργειες έτσι ώστε να συλλεχθούν και να αποθηκευτούν σε έναν νέο πίνακα, μόνο τα προβλήματα των τελευταίων 6 μηνών και μάλιστα χωρισμένα ανά μήνες. Ουσιαστικά, η συγκεκριμένη μέθοδος επιστρέφει έναν πίνακα αυτής της μορφής: [6, 8, 10, 12, 13, 14], όπου το 6 είναι ο αριθμός των προβλημάτων του μήνα x1, το 8 ο αριθμός των προβλημάτων του μήνα x2 κι ουτο καθεξής. Τέλός, αυτός ο πίνακας θα χρησιμοποιήθει ως data στα options του αριστερά ραβδογράμματος.

2.2.3.12 profile.view.ts

Το profile.view.ts φορτώνει το profile.template.html στο element με id="container" και είναι υπεύθυνο για την εμφάνιση της σελίδας επεξεργασίας του προφίλ του χρήστη.

Αρχικά στην **initialize**, στην μεταβλητή **userID** αποθηκεύεται το id του χρήστη και στην μεταβλητή **email** το email του χρήστη. Στη συνέχεια καλείται η render.

Η κύρια function του συγκεκριμένου view είναι η **clickOnSubmit**, η οποία εκτελείται όταν πατηθεί το κουμπί με id="submit". Η συγκεκριμένη μέθοδος είναι υπεύθυνη για την ολοκλήρωση της επεξεργασίας του χρήστη. Σε περίπτωση επιτυχίας, ανανεώνεται η βάση και εμφανίζεται κατάλληλο μήνυμα στην οθόνη. Σε αντίθετη περίπτωση, θα εμφανιστεί μήνυμα σφάλματος.

Η function **changePassword**, εκτελείται όταν πατηθεί το κουμπί με id="changePass" και είναι υπεύθυνη για την αλλαγή κωδικού πρόσβασης του χρήστη..

2.2.3.13 report_problem_main_page.view.ts

Το report_problem_main_page.view.ts φορτώνει το report_problem_main_page.template.html στο body και είναι υπεύθυνο για την εμφάνιση της Αρχικής Σελίδας Αναφοράς Προβλήματος.

2.2.3.14 report_problem.view.

Το report_problem.view.ts φορτώνει το report_problem.template.html στο body και είναι υπεύθυνο για την εμφάνιση της Σελίδας Αναφοράς Προβλήματος.

Η κύρια function του συγκεκριμένου view είναι η **clickOnSubmit**, η οποία εκτελείται όταν πατηθεί το κουμπί με id="submit". Στην συγκεμένη μέθοδο, σε περίπτωση που ο χρήστης έχει επιλέξει φωτογραφία, εκτελείται ο αντίστοιχος κώδικας έτσι ώστε αυτή η φωτογραφία να αποθηκευτεί στο Firebase Storage. Στη συνέχεια καλείται η function **submitData**, η οποία είναι υπεύθυνη για την δημιουργία του καινούργιου προβλήματος στην βάση δεδομένων. Σε περίπτωση είτε επιτυχίας είτε αποτυχίας, εμφανίζονται τα κατάλληλα μηνύματα.

Η function **uploadPhoto** εκτελείται όταν πατηθεί το κουμπί με id="uploadPhoto" και είναι υπεύθυνη για την επιλογή και εμφάνιση μιας φωτογραφίας.

Η function **removePhoto** εκτελείται όταν πατηθεί το κουμπί με id="**removePhoto** " και είναι υπεύθυνη για την διαγραφή της φωτογραφίας που είχε επιλέξει νωρίτερα ο χρήστης.

2.2.4 Templates

Στην παρούσα εφαρμογή, επιλέξαμε να παρουσιάσουμε την κάθε οθόνη με ένα η περισσότερα templates. Με αυτόν τρόπο, απλοποιείται η λειτουργικότητα της εφαρμογής και ταυτόχρονα γίνεται πιο εύκολη σε τυχόν αλλαγές.

Τα templates που έχουν χρησιμοποιηθεί, δεν είναι τα κλασικά templates που θα μπορούσε κάποιος να συναντήσει, δηλαδή δεν περιέχουν απλά html κώδικα. Όπως θα δούμε, τα συγκεκριμένα templates χρησιμοποιούν handlebars, για αυτό το λόγο γίνονται compile πριν χρησιμοποιηθούν σε κάποιο view.

Για παράδειγμα, μέσα στο template περνάμε διάφορα δεδομένα με την χρήση handlebars, π.χ. {{ data }} . Επιπλέον, μπορούν να χρησιμοποιηθούν διάφορες συνθήκες ή επαναλήψεις, π.χ. {{#each collection}} … {{/each}}, {{#if editable}}...{{/if}}.

Τέλος, είναι σημαντικό να αναφερθεί, πως το τεμάχισμα του html κώδικα σε πολλά templates αλλά και η προσθήκη handlebars σε αυτά, έχουν διευκολύνει σε μεγάλο βαθμό την διαχείριση των αλλαγών κι έχουν μειώσει το μέγεθος του κώδικα.

Αφού, έγινε περιγραφή των βασικών λειτουργιών ενός template, μπορούμε να προχωρήσουμε στην αναφορά των templates της παρούσας εφαρμογής. Όλα τα templates βρίσκονται μέσα στον φάκελο /src/main/web/js/templates και είναι τα παρακάτω:

- **login.template.html:** απεικονίζει την οθόνη Σύνδεσης
- **forgotPassword.template.html:** απεικονίζει την οθόνη Ανάκτησης Κωδικού
- **home.template.html:** απεικονίζει την Αρχική Οθόνη
- **users.template.html:** απεικονίζει την οθόνη εμφάνισης των Χρηστών
- **users_table.template.html:** απεικονίζει μόνο τον πίνακα που εμφανίζεται στην σελίδα των χρηστών
- **user.template.html:** απεικονίζει την οθόνη Δημιουργίας/Επεξεργασίας Χρήστη
- **problems.template.html:** απεικονίζει την οθόνη των Προβλημάτων
- **problems_table.template.html:** απεικονίζει μόνο τον πίνακα που εμφανίζεται στην σελίδα των προβλημάτων
- **problem.template.html:** απεικονίζει την οθόνη Επεξεργασίας Προβλήματος
- **analytics.template.html:** απεικονίζει την οθόνη των Αναλύσεων
- **profile.template.html:** απεικονίζει την οθόνη του Προφίλ του Χρήστη
- **navbar_header.template.html:** απεικονίζει το πάνω μέρος της οθόνης, το οποίο περιλαμβάνει και το Μενού Χρήστη
- **navbar_side.template.html:** απεικονίζει το Μενού Πλοήγησης
- **header.template.html:** απεικονίζει το λογότυπο της εφαρμογής
- **report_problem_main_page.template.html:** απεικονίζει την Αρχική Σελίδα Αναφοράς Προβλήματος
- **report_problem.template.html:** απεικονίζει τη Σελίδα Αναφοράς Προβλήματος

2.2.5 Βιβλιοθήκες

Στην παρούσα ενότητα θα γίνει αναφορά σε κάποιες από τις κύριες βιβλιοθηκες, που έχουν χρησιμοποιηθει στο πρότζεκτ, για να έχουμε το επιθυμητό αποτέλεσμα.

2.2.5.1 Bootstrap

Το Bootstrap είναι μια συλλογή εργαλείων ανοιχτού κώδικα για τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών. Το Bootstrap αποτελείται ουσιαστικά από μια σειρά στυλ(stylsheets) που εφαρμόζουν τα διάφορα συστατικά του πακέτου εργαλείων. Περιέχει HTML και CSS για τις μορφές τυπογραφίας, κουμπιά πλοήγησης και άλλα στοιχεία του περιβάλλοντος. Επιπλεόν, είναι συμβατό με

όλους τους φυλλομετρητές. Έχει χρησιμοποιηθεί στην παρούσα πτυχιακή εργασία για να δημιουργηθεί μια responsive εφαρμογή. Αυτό σημαίνει ότι οι διάφορες οθόνες θα προσαρμόζονται δυναμικά, λαμβάνοντας υπόψη τα χαρακτηριστικά της κάθε συσκευής.

2.2.5.2 FontAwesome

Η FontAwesome είναι μία εργαλειοθήκη με ευρεία γκάμα δωρεάν αλλά και επί πληρωμή εικονιδίων, βασισμένη σε CSS. Στην παρούσα εφαρμογή, έχουν χρησιμοποιηθεί αρκετά από τα δωρεάν εικονίδια που προσφέρει, καθώς η ενσωμάτωση τους στον κώδικα είναι πολύ εύκολη.

2.2.5.3 Handlebars

Τα Handlebars είναι μια απλή γλώσσα προτύπων, η οποία χρησιμοποιεί κάποια handlebars expressions. Ενα handlebar expression αποτελείται από: {{ περιεχόμενο }}, το οποίο όταν εκτελεστεί το template θα αντικατασταθεί από κάποιες τιμές.

Στην παρούσα πτυχιακή εργασία, τα handlebars έχουν χρησιμοποιηθεί ευρέως για την φόρτωση δεδομένων στα templates. Επιπλέον, εσωτερικά των templates, μας δίνεται η δυνατότητα να χρησιμοποιήσουμε συνθήκες και βρόχους, με την χρήση των handlebars.

Επιπλέον, τα handlebars μας δίνουν την δυνατότητα να δημιουργήσουμε δικούς μας helpers, τους οποίους θα εισάγουμε στα templates και θα μας δώσουν το επιθυμητό αποτέλεσμα.

Παρακάτω, βλέπουμε τον κώδικα για την δημιουργία ενός helper με όνομα sum, ο οποίος επιστρέφει το άθροισμα δύο αριθμών.

```
Handlebars.registerHelper('sum', function (a,b) {
    var sum = a+b;
    return sum;
});
```

Η χρήση του helper μέσα στο template γίνεται με τον παρακάτω τρόπο:

```
<div>{{sum 4 5}}</div>
```

2.2.5.4 DataTables

Το DataTables είναι ένα plugin για την βιβλιοθήκη jQuery της JavaScript. Είναι ένα εξαιρετικά ευέλικτο εργαλείο, το οποίο έχει χρησιμοποιηθεί στην παρούσα εφαρμογή για την προσθήκη σελιδοποίησης και αναζήτησης στους πίνακες που εμφανίζονται.

Η χρήση της είναι πολύ εύκολη και γίνεται απλά με την προσθήκη του παρακάτω κώδικα:

```
$("#dataTables").dataTable({
    "ordering": false,
    "info":       false
```

```
});
```

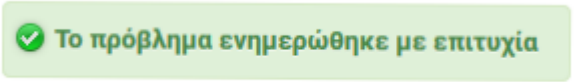
Στους πίνακες της παρούσας εφαρμογής δεν χρησιμοποιήθηκε ordering και info. Αυτό διαπιστώνεται από το παραπάνω παράδειγμα, αφού θέτουμε τις τιμές των ordering και info σε false.

2.2.5.5 notify.js

Το notify.js είναι ένα plugin για την βιβλιοθήκη jQuery της JavaScript, το οποίο μας παρέχει απλές αλλά πλήρως προσαρμόσιμες ειδοποιήσεις, Έχει χρησιμοποιηθεί στην παρούσα εφαρμογή για την εμφάνιση σφαλμάτων και επιτυχημένων ειδοποιήσεων, έτσι ώστε να ενημερώνεται ο χρήστης καθ' όλη τη διάρκεια της περιήγησης του στην εφαρμογή.

Παρακάτω, υπάρχει ο κώδικας για την εμφάνιση μιας ειδοποίησης επιτυχίας καθώς και το αντίστοιχο αποτέλεσμα:

```
$.notify("Success message", "success");
```



Εικόνα 3: Στιγμιότυπο ειδοποίησης

2.2.5.6 apexCharts

Τα apexCharts είναι μια βιβλιοθήκη για την εμφάνιση διαγραμμάτων, συνεπώς για την απεικόνιση διαφόρων δεδομένων. Είναι responsive, δηλαδή τα διαγράμματα προσαρμόζονται δυναμικά ανάλογα με το μέγεθος της οθόνης. Επιπλέον, είναι interactive, δηλαδή σου δίνει την δυνατότητα για zoom in/out, εξαγωγή svg, κατάλληλα μηνύματα όταν ο χρήστης κάνει hover στα δεδομένα. Τέλος, είναι δυναμικά, κάνοντας τα δεδομένα πραγματικά διαδραστικά.

Παρακάτω, υπάρχει ο κώδικας για την δημιουργία ενός line Chart:

```
var options = {
      series: [
      {
        name: "High",
        data: [28, 29, 33, 36, 32, 32, 33]
      },
      {
        name: "Low",
        data: [12, 11, 14, 18, 17, 13, 13]
```

```
      }
      ],
      chart: {
         height: 350,
         type: 'line',
         dropShadow: {
             enabled: true,
             color: '#000',
             top: 18,
             left: 7,
             blur: 10,
             opacity: 0.2
         },
         toolbar: {
            show: false
         }
      },
      colors: ['#77B6EA', '#545454'],
      dataLabels: {
            enabled: true,
      },
      stroke: {
            curve: 'smooth'
      },
      title: {
            text: 'Average High & Low Temperature',
            align: 'left'
      },
      grid: {
            borderColor: '#e7e7e7',
            row: {
                colors: ['#f3f3f3', 'transparent'],
                opacity: 0.5
            },
      },
      xaxis: {
            categories: ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
'Jul'],
            title: {
               text: 'Month'
            }
      },
      yaxis: {
            title: {
             text: 'Temperature'
            },
```

```
            min: 5,
            max: 40
      },
      legend: {
            position: 'top',
            horizontalAlign: 'right',
            floating: true,
            offsetY: -25,
            offsetX: -5
      }
      };

      var chart = new ApexCharts(document.querySelector("#chart"),
options);
      chart.render();
```

2.3 BACK-END

Το Back-End αναφέρεται ουσιαστικά στο πίσω μέρος μιας εφαρμογής που χρησιμοποιείται από κάποιον χρήστη. Με λίγα λόγια, ασχολείται με το κομμάτι του server και της βάσης δεδομένων. Η αναπτυξη της παρούσας εφαρμογής στηρίζεται στο Firebase, που είναι μία πλατφόρμα, η οποία έχει αναπτυχθεί από την Google, όπως έχεις περιγραφεί πιο πάνω. Τέλος, για την χρήση κάποιων λειτουργιών του Firebase, έχει χρησιμοποιηθεί και η node.js

2.3.1.1 Firebase SDK

Όπως αναφέρθηκε παραπάνω, το back-end της παρούσας εφαρμογής βασίζεται στο Firebase. Για να μπορέσουμε όμως να το χρησιμοποιήσουμε, ήταν πρώτα αναγκαίο να δημιουργήσουμε ένα Firebase project. Η δημιουργία ενός Firebase project είναι πολύ απλή. Αυτό που πρέπει να κάνουμε είναι να επισκεφτούμε σε έναν φυλλομετρητή την Firebase console, να πατήσουμε Add Project, να δώσουμε όνομα στο καινούργιο μας πρότζεκτ και να ακολουθήσουμε τα απαραίτητα βήματα, έως ότου φτάσουμε στην επιτυχημένη δημιουργία.

Αφού τελειώσουμε με την δημιουργία του Firebase project, θα πρέπει να εγκαταστήσουμε το Firebase SDK στην εφαρμογή μας. Αυτό έγινε μέσω του npm. Στη συνέχεια, μπορούμε να χρησιμοποιήσουμε το Firebase κάνοντας το require, στα αρχεία που θέλουμε να το χρησιμοποιήσουμε.

Στη συνέχεια, για να αρχικοποιήσουμε το Firebase, πρέπει να εισάγουμε το configuration του. Αυτό γίνεται στο αρχείο του project, με όνομα main.js. Παρακάτω υπάρχει ο κώδικας του configuration.

```
// Your web app's Firebase configuration
var firebaseConfig = {
      apiKey: "AIzaSyCE_mRefrHJvmaQgsGdK-6UC2FlcDUnl9A",
      authDomain: "municipality-problems.firebaseapp.com",
      databaseURL: "https://municipality-problems.firebaseio.com",
      projectId: "municipality-problems",
      storageBucket: "municipality-problems.appspot.com",
```

```
      messagingSenderId: "594959954044",
      appId: "1:594959954044:web:0639dd8118b65680c019f9",
      measurementId: "G-8HY9LB74CQ"
};

// Initialize Firebase
firebase.initializeApp( firebaseConfig );
```

Αφού έχει ολοκληρωθεί και αυτό το κομμάτι, πλέον είμαστε σε θέση να χρησιμοποιήσουμε το firebase, έτσι ώστε να εκτελέσουμε τις απαραίτητες λειτουργίες της εφαρμογής. .

2.3.1.2 Firebase Admin

Το Firebase Admin μας δίνει την δυνατότητα να έχουμε πρόσβαση σε προνομιακές λειτουργίες, τις οποίες δεν θα μπορούσαμε να έχουμε μόνο με την χρήση του Firebase SDK. Στην παρούσα εφαρμογή η χρήση του, ήταν απαραίτητη για την διαχείριση των χρηστών. Πιο συγκεκριμένα, χρησιμοποιήθηκε για την δημιουργία χρήστη, την διαγραφή χρήστη/χρηστών και την αλλαγή κωδικού.

Παρακάτω παρατίθεται ο κώδικας για αυτές τις λειτουργίες:

```
//create user
admin.auth().createUser(userData)
    .then(function( userRecord ) {
            //See the UserRecord reference doc for the contents of
userRecord.
            console.log('Successfully created new user:',
userRecord.uid);
    })
    .catch(function(error) {
            console.log('Error creating new user:', error);
    });

//delete user
admin.auth().deleteUsers(ids)
    .then(function(deleteUsersResult) {
            console.log('Successfully deleted ' +
deleteUsersResult.successCount + ' users');
            console.log('Failed to delete ' +
deleteUsersResult.failureCount + ' users');
    })
    .catch(function(error) {
            console.log('Error deleting users:', error);
    });

//update user's password
```

```
admin.auth().updateUser(uid, {
    password: newPassword
})
.then(function(userRecord) {
    // See the UserRecord reference doc for the contents of userRecord.
    console.log('Successfully updated user', userRecord.toJSON());
})
.catch(function(error) {
    console.log('Error updating user:', error);
});
```

Για να μπορέσουμε να το χρησιμοποιήσουμε, θα πρέπει πρώτα να το εισάγουμε στο πρότζεκτ. Η εισαγωγή του λοιπόν στο πρότζεκτ, έγινε με την χρήση του Node.js, στο οποίο θα αναφερθούμε πιο κάτω.

2.3.1.3 Cloud Firestore

Το Cloud Firestore είναι μία NoSQL βάση δεδομένων από το Firebase και το Google Cloud Platform, η οποία χρησιμοποιείται για την ανάπτυξη εφαρμογών κινητών, ιστού και διακομιστών. Το διαχειριστικό κομμάτι του Firebase, μας δίνει την δυνατότητα να διαχειριστούμε τη βάση δεδομένων μας.

Ας περιγράψουμε τώρα πως λειτουργεί το Cloud Firestore. Το Cloud Firestore αποθηκεύει τα δεδομένα μας σε documents, τα οποία περιέχουν πεδία που αντιστοιχίζονται σε τιμές. Αυτά τα έγγραφα με τη σειρά τους, αποθηκεύονται σε collections. Με αυτόν τον τρόπο μπορούμε να οργανώσουμε τα δεδομένα μας και να δημιουργησουμε τα κατάλληλα ερωτήματα.

Στην παρούσα εφαρμογή έχουμε τα εξής collections:

- users: Στο collection “users” αποθηκεύουμε όλους του χρήστες. Ο κάθε χρήστης δημιουργείται αφού πρώτα έχει δημιουργηθεί χρήστης στο Authentication (θα αναλυθεί παρακάτω). Με αυτόν τον τρόπο ο κάθε χρήστης από την συλλογή users θα έχει ένα μοναδικό doc id, το οποίο θα είναι ίδιο με το user uid του χρήστη στο Authentication. Αυτό γίνεται γιατί το Firebase Authentication μας δίνει την δυνατότητα να αποθηκεύσουμε κάποια συγκεκριμένα στοιχεία στον χρήστη κατά την δημιουργία του. Οπότε με αυτόν τον τρόπο στην συλλογή users θα αποθηκεύουμε όλα τα στοιχεία που θέλουμε να έχει ένας χρήστης και ταυτόχρονα θα υπάρχει αντιστοίχιση και με τον χρήστη του Authentication. Τα πεδία της συγκεκριμένης συλλογής λοιπόν είναι τα εξής: email, firstName (όνομα), lastName (επίθετο), municipality (δήμος), role (ρόλος), phone (τηλέφωνο).
- problems: Στο collection “problems” αποθηκεύονται όλα τα προβλήματα που αναφέρονται από τους πολίτες και στην συνέχεια διαχειρίζονται οι υπεύθυνοι των Δήμων. Τα πεδία της συγκεκριμένης συλλογής λοιπόν είναι τα εξής: description (μια μικρή περιγραφή του προβλήματος), firstName (όνομα), lastName (επίθετο), email, municipality (δήμος), submissionDate (ημερομηνία υποβολής προβλήματος), status (κατάσταση προβλήματος), lat (γεωγραφικό πλάτος), lng (γεωγραφικό μήκος), image (φωτογραφία προβλήματος).

Για να χρησιμοποιήσουμε το Firestore στην εφαρμογή μας και να μπορέσουμε να κάνουμε τα απαραίτητα ερωτήματα, πρέπει πρώτα να αρχικοποιήσουμε ένα instance από το Firestore. Αυτό γίνεται με τον παρακάτω τρόπο:

```
var db = firebase.firestore();
```

Πλέον είμαστε σε θέση να χρησιμοποιήσουμε το Firestore. Παρακάτω παρατίθεται ο κώδικας για κάποιες από τις βασικές λειτουργίες του Firestore στην παρούσα εφαρμογή, όπως δημιουργία ενός document, διαγραφή ενός document, λήψη όλων των εγγραφων:

```
// Add a new document in collection "users"
db.collection("users").doc(user.uid).set({
      email: "c.kaneloglou@gmail.com",
      firstName : "Χριστίνα",
      lastName : "Κανέλογλου",
      municipality: "-",
      role: "SupervisingAdmin",
      phone : "6988936435"
})
.then(function() {
      console.log("Document successfully written!");
})
.catch(function(error) {
      console.error("Error writing document: ", error);
});

//Get all documents of collection "users" and save them in users table
except from the logged in user
this.db.collection('users').get().then((snapshot) => {
            self.users = [];
            snapshot.docs.forEach(doc => {
                        if(doc.id != self.userId) {
                        var user = {
                                    "data": doc.data(),
                                    "id": doc.id
                  }
                        self.users.push(user);
                        }
            })
    });

//delete document
db.collection(Collection).doc(id).delete()
                .then(function() {
                        //success delete
                })
                .catch(function(error) {
                        //handle errors
```

```
                });
```

2.3.1.4 Cloud Storage

To Cloud Storage To Cloud Storage for Firebase είναι μια ισχυρή, απλή και οικονομικά αποδοτική υπηρεσία αποθήκευσης αντικειμένων. Έχει δημιουργηθεί για να παρέχει στους προγραμματιστές εφαρμογών έναν χώρο, στον οποίο θα μπορούν να αποθηκεύουν περιεχόμενο, όπως φωτογραφίες, βίντεο, ήχο.

Στην παρούσα εφαρμογή έχει χρησιμοποιηθεί για την αποθήκευση των εικόνων των προβλημάτων που υποβάλουν οι χρήστες καθώς και για την εμφάνιση αυτών των προβλημάτων στο διαχειριστικό κομμάτι. Κάθε αρχείο όταν ανεβαίνει στο Cloud Storage, εκτός από το όνομα του, έχει και ένα συγκεκριμένο file location, το οποίο χρησιμοποιείται από τον χρήστη για την λήψη του.

Παρακάτω παρατίθεται ο κώδικας για το ανέβασμα της φωτογραφίας στο Cloud Storage:

```
var storageRef = this.storage.ref(this.uploadFile.name);
var uploadTask =  storageRef.put(this.uploadFile);

uploadTask.on('state_changed', function(snapshot){
// Observe state change events such as progress, pause, and resume
}, function(error) {
        console.log("Unsuccessful Upload");
}, function() {
       storageRef.getDownloadURL().then(function(url) {
       self.downloadURL = url;
       self.submitData();
       }).catch(function(error) {
            console.error(error);
       });
});
```

2.3.1.5 Authentication

Ένα τμήμα της παρούσας εφαρμογής, όπως έχουμε αναφέρει, είναι η δημιουργία ενός διαχειριστικού περιβάλλοντος. Αυτόματα δηλαδή αναφερόμαστε σε χρήστες. Οι περισσότερες εφαρμογές πρέπει να γνωρίζουν την ταυτότητα ενός χρήστη και να αποθηκεύουν με ασφάλεια τα δεδομένα του. Για αυτόν ακριβώς τον λόγο, υπάρχει το Firebase Authentication.

Το Firebase Authentication λοιπόν, παρέχει υπηρεσίες backend για έλεγχο ταυτότητας χρηστών στις εφαρμογές. Υποστηρίζει έλεγχο ταυτότητας με χρήση κωδικών πρόσβασης, αριθμών τηλεφώνου, Google, Facebook, Twitter και άλλα. Στην παρούσα εφαρμογή, έχει χρησιμοποιηθεί η χρήση email/κωδικού.

Το Firebase, μας προσφέρει ένα διαχειριστικό περιβάλλον. Μία από τις κατηγορίες που μας προσφέρει είναι το Authentication, στο οποίο έχουμε πρόσβαση από το μενού αριστερά. Εκεί μπορούμε να παρατηρήσουμε όλους τους χρήστες που έχουν δημιουργηθεί. Πιο συγκεκριμένα, κάθε φορά που δημιουργούμε καινούργιο χρήστη, ο πίνακας θα ανανεώνεται. Κάθε χρήστης που δημιουργείται, θα έχει ένα μοναδικό User UID, το οποίο αναγράφεται στον πίνακα. Επιπλέον, το Authentication μας προσφέρει μερικές ακόμα λειτουργίες, όπως απενεργοποίηση χρήστη, διαγραφή χρήστη και επαναφορά κωδικού.

Τέλος, στην παρούσα εφαρμογή και όσον αφορά το διαχειριστικό κομμάτι, το Firebase Auth έχει χρησιμοποιηθεί και για τις λειτουργίες του login και logout χρήστη. Παρακάτω υπάρχει ο κώδικας που χρησιμοποιήθηκε για την υλοποίηση των δύο αυτών λειτουργιών.

```
//Login
firebase.auth().signInWithEmailAndPassword(email, password)
            .then(function(user) {
            //success login
            })
            .catch(function(error) {
                  // Handle Errors here.
                  var errorCode = error.code;
                 var errorMessage = error.message;
                  // ...
            });

//Logout
firebase.auth().signOut().then(function() {
            window.open('./','_self')
      }).catch(function(error) {
            console.error('Error while logging out', error)
      });
```

2.3.2 node.js

Για την χρήση του Firebase Admin και την υλοποίηση των κατάλληλων λειτουργιών, που αυτό μας προσφέρει, ήταν απαραίτητη η εγκατάσταση του Node.js στην παρούσα εφαρμογή. Όπως έχουμε αναφέρει, το Node.js είναι μία πλατφόρμα ανάπτυξης λογισμικού κυρίως στην πλευρά του διακομιστή, χρησιμοποιώντας την γλώσσα προγραμματισμού JavaScript.

Η εγκατάσταση του Firebase Admin λοιπόν, έγινε με την χρήση του npm. Στη συνέχεια μέσα στον φάκελο /src/backend/ δημιουργήσαμε ένα αρχείο με όνομα app.js, το οποίο θα είναι και το εκτελέσιμο αρχείο μας.

Μέσα σε αυτό το αρχείο γίνεται αρχικοποίηση του Firebase Admin με τον εξής τρόπο:

```
var admin = require('firebase-admin');
var serviceAccount = require("./serviceAccount/key.json");
admin.initializeApp({
  credential: admin.credential.cert(serviceAccount),
  databaseURL: "https://municipality-problems.firebaseio.com"
});
```

Η μεταβλητή serviceAccount κάνει require ένα json αρχείο, με το private key του Firebase Admin SDK. Το private key δημιουργήθηκε από την κονσόλα του Firebase και πιο συγκεκριμένα στο Settings > Service Accounts. Το json αυτό αρχείο χρησιμοποιείται σαν παράμετρος στο πεδίο credential. Επιπλέον, στο databaseURL θα βάλουμε το URL που αντιστοιχεί στην βάση της εφαρμογής μας.

Στη συνέχεια, κι αφού εγκατασταθούν μέσω npm, κάνουμε require το Express.js και το body-parser.

Το Express.js είναι ένα framework για το Node.js, το οποίο έχει σχεδιαστεί για την δημιουργία εφαρμογών ιστού και API. Πιο συγκεκριμένα, είναι το τυπικό server framework για το Node.js . Για να μπορέσουμε να διαχειριστούμε τα HTTP POST ερωτήματα στο Express.js, ήταν απαραίτητη και η προσθήκη του module body-parser. Ουσιαστικά, το body-parser εξάγει ολόκληρο το σώμα μιας εισερχόμενης ροής και το αποθηκεύει στο req.body .

Παρακάτω, παρατίθεται ο κώδικας για ένα απλό παράδειγμα διαχείρισης ενός http post request από το back-end και πως αυτό καλείται από την πλευρά του front-end με την χρήση ajax.

```
//create new user using Express.js (back-end)
app.post('/rest/createUser', function (req, res) {
      //το req.body ουσιαστικά περιέχει τα στοιχεία του χρήστη, τα οποία
τα στέλνουμε όταν
            //καλούμε το request με την  χρήση ajax
var userData = req.body;

      //call Firebase Admin function for creating new user
      admin.auth().createUser(userData)
            .then(function(userRecord) {
              //send back to the ajax call the UserRecord reference doc
    res.send(userRecord);
      })
      .catch(function(error) {
                console.log('Error creating new user:', error);
      });
})


//ajax call (front-end)
$.ajaxDefaultQ({
      url: "/rest/createUser",
              crossDomain: true,
```

```
            dataType: 'json',
            type: "post",
            data: JSON.stringify(userData),
            contentType: "application/json; charset=UTF-8"
}).done(function(user, status, xhr){
      //success
}).fail(function(response) {
      //handle errors
});
```

2.4 Επίλογος

Σε αυτό το κεφάλαιο έγινε μία προσπάθεια περιγραφής του συστήματος, τόσο από την πλευρά του Front-End, όσο κι από την πλευρά του Back-End. Εφόσον βρισκόμαστε στο τέλος αυτού το κεφαλαίου, είμαστε πλέον στη θέση να προχωρήσουμε στην αναλυτική περιγραφή της εφαρμογής.

Κεφάλαιο 3ο: Αναλυτική Περιγραφή και Χρήση της Εφαρμογής

3.1 Εισαγωγή

Στο τελευταίο αυτό κεφάλαιο της παρούσας πτυχιακής εργασίας θα περιγραφεί η αναλυτική λειτουργία της εφαρμογής με τη χρήση στιγμιοτύπων οθόνης (screenshots), ώστε να γίνει όσο το δυνατόν πιο κατανοητή στον αναγνώστη.

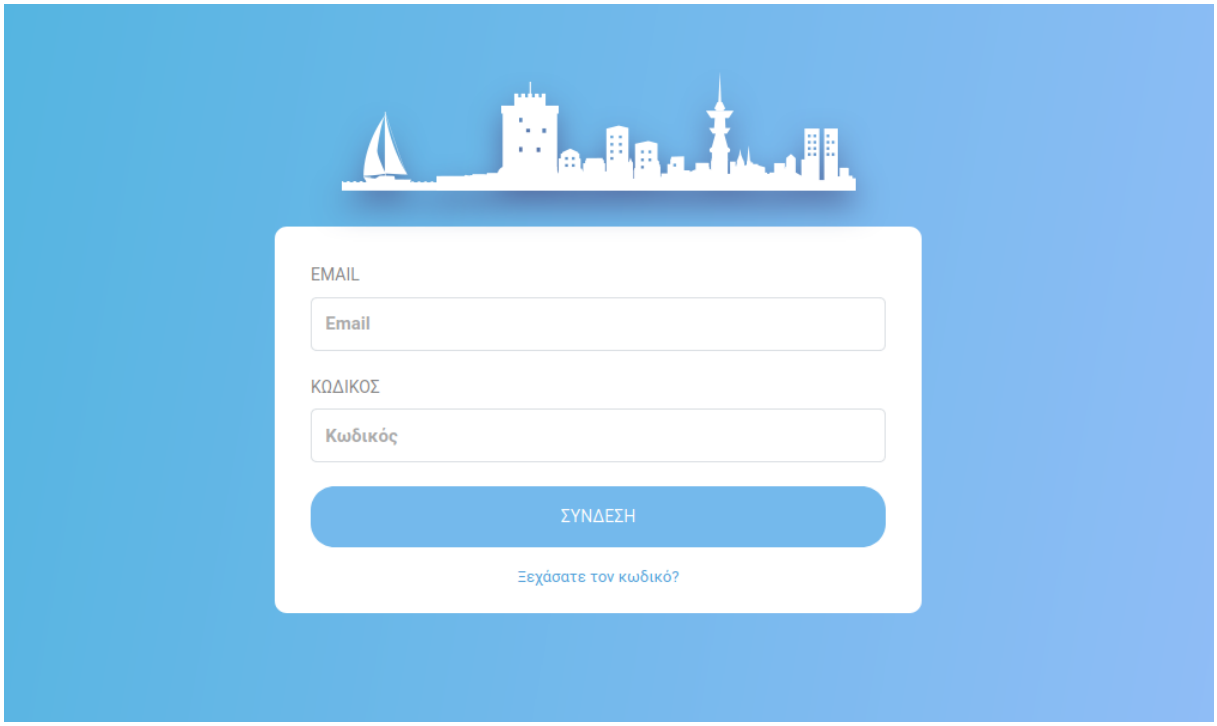
3.2 Διαχειριστικό Περιβάλλον

Σε αυτό το υποκεφάλαιο θα περιγράψουμε το διαχειριστικό κομμάτι της εφαρμογής. Το διαχειριστικό κομμάτι αναφέρεται στο περιβάλλον, στο οποίο θα έχουν πρόσβαση μόνο οι αρμόδιοι των Δήμων της περιφερειακής ενότητας της Θεσσαλονίκης. Μέσα από αυτό θα έχουν την δυνατότητα να διαχειρίζονται τα προβλήματα, που υποβάλλονται από τους πολίτες.

3.2.1 Σύνδεση

Μόλις ο χρήστης ανοίξει την εφαρμογή, εμφανίζεται η οθόνη της Σύνδεσης. Στην οθόνη παρατηρούμε το logo της εφαρμογής και ακριβώς από κάτω μία φόρμα. Στην φόρμα, ο χρήστης πρέπει να εισάγει το email και τον κωδικό του και στη συνέχεια να πατήσει το κουμπί Σύνδεση. Αν το ζεύγος στοιχείων που έχει πληκτρολογήσει αντιστοιχεί σε κάποιον χρήστη, τότε η είσοδος του στη εφαρμογή θα είναι επιτυχής. Σε αντίθετη περίπτωση, θα εμφανιστεί στην οθόνη ένα μήνυμα σφάλματος με το κατάλληλο μήνυμα.

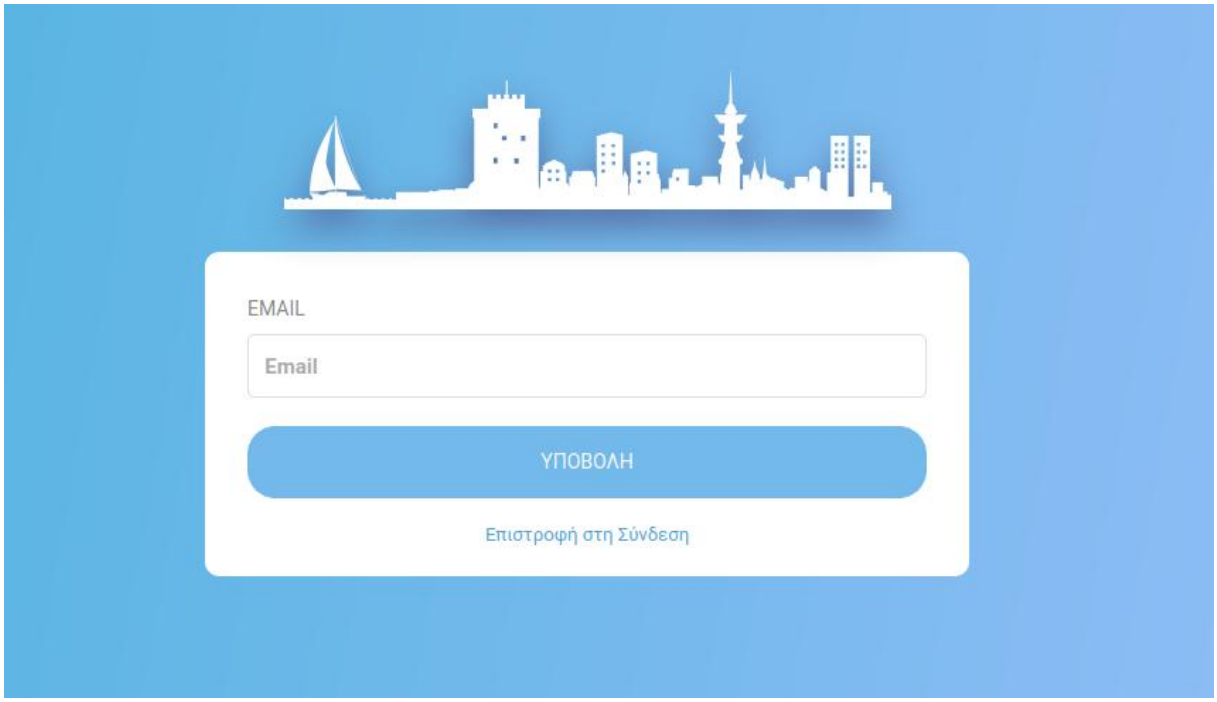
Επιπλέον στην συγκεκριμένη οθόνη, ακριβώς κάτω από το κουμπί Σύνδεση, παρατηρούμε έναν σύνδεσμο με το μήνυμα “Ξεχάσατε τον κωδικό?”, ο οποίος οδηγεί τον χρήστη στη σελίδα ανάκτησης κωδικού. Η συγκεκριμένη σελίδα θα περιγραφεί στην συνέχεια.



Εικόνα 3: Στιγμιότυπο οθόνης Σύνδεσης 1

3.2.2 Ανάκτηση Κωδικού

Σε περίπτωση που κάποιος χρήστης ξεχάσει τον κωδικό του για την είσοδο στην εφαρμογή, του δίνεται η δυνατότητα ανάκτησης κωδικού. Στην σελίδα Ανάκτησης Κωδικού το μόνο που έχει να κάνει ο χρήστης είναι να συμπληρώσει το email του και στη συνέχεια να πατήσει υποβολή. Αν το email που έχει δώσει είναι σωστό και υπάρχει στους εγγεγραμμένους χρήστες της εφαρμογής, τότε θα του λάβει ένα email ανάκτησης κωδικού. Στο συγκεκριμένο email θα υπάρχει ένας σύνδεσμος, ο οποίος θα του ανοίγει μία σελίδα για να συμπληρώσει τον καινούργιο του κωδικό. Σε περίπτωση που το email είναι λανθασμένο, θα εμφανιστεί στην οθόνη ένα μήνυμα σφάλματος με το κατάλληλο μήνυμα.

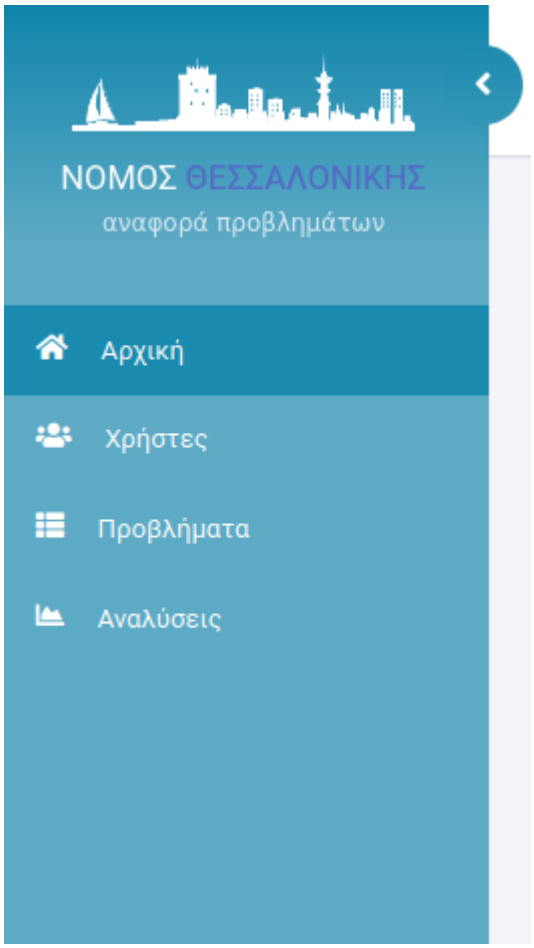


Εικόνα 4: Στιγμιότυπο οθόνης Ανάκτησης Κωδικού

3.2.3 Μενού Πλοήγησης

Μετά την είσοδο του χρήστη στην εφαρμογή, στο αριστερό μέρος όλων των οθονών, εμφανίζεται ένα μενού πλοήγησης. Στο πάνω μέρος εμφανίζεται το λογότυπο της εφαρμογής και ακριβώς από κάτω μία λίστα με τους συνδέσμους, που οδηγούν στις αντίστοιχες σελίδες.

Επιπλέον, πάνω δεξιά στο μενού υπάρχει ένα βελάκι με το οποίο μπορείς να κλείσεις και να ανοίξεις το μενού. Σε περίπτωση, που το μενού είναι κλειστό, εμφανίζονται πλέον μόνο τα εικονίδια και τα λεκτικά εμφανίζονται σε περίπτωση που κάνεις hover πάνω σε αυτά.

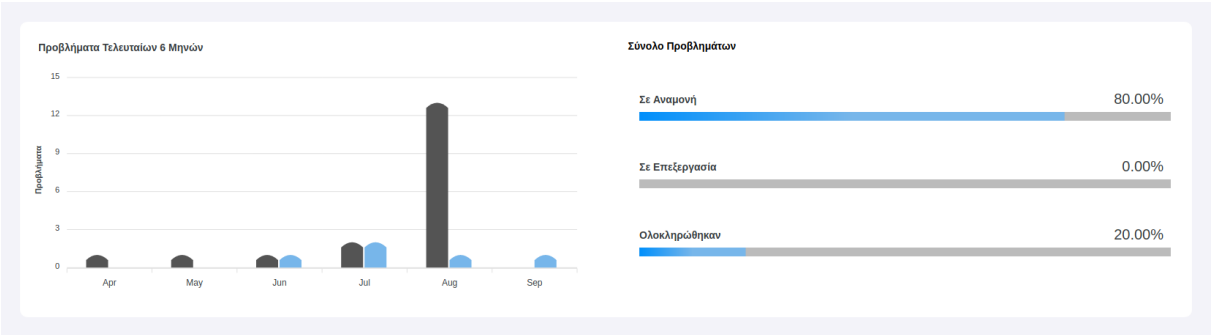


Εικόνα 5: Μενού Πλοήγησης 1

3.2.4 Αρχική Σελίδα

Μετά την είσοδο του χρήστη στην εφαρμογή, εμφανίζεται η Αρχική Σελίδα. Η σελίδα αυτή χωρίζεται σε δύο μέρη, σε αυτό των διαγραμμάτων και σε αυτό του χάρτη.

Στο πάνω μέρος, κάνουν την εμφάνιση τους 2 διαγράμματα που αφορούν τα προβλήματα των τελευταίων 6 μηνών.

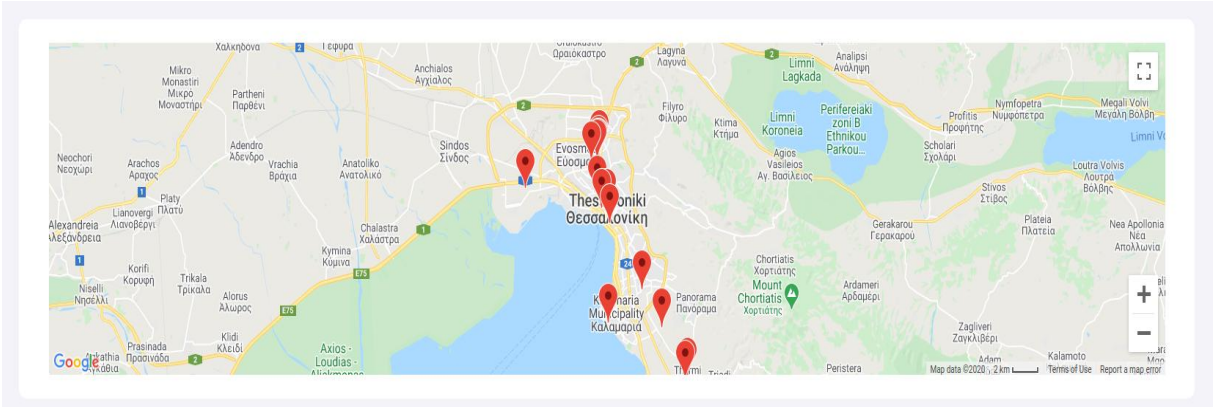


Εικόνα 6: Γραφήματα Αρχικής Σελίδας 1

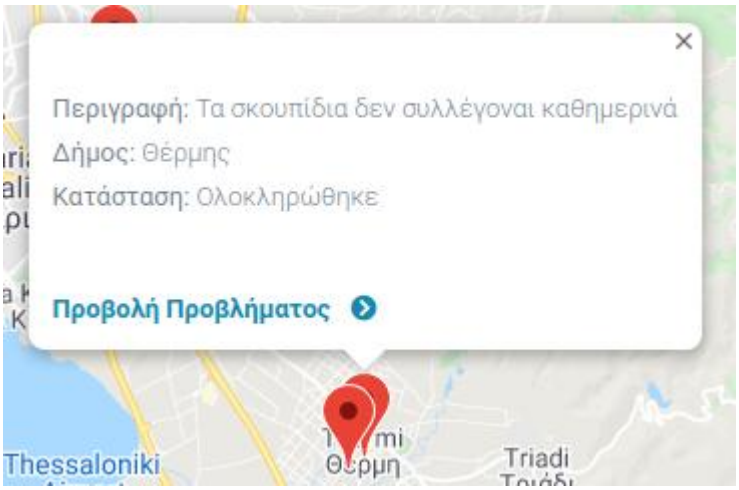
Το αριστερά διάγραμμα, πρόκειται για ένα ραβδόγραμμα, στο οποίο εμφανίζονται τα προβλήματα όλων των δήμων που έχουν υποβληθεί τους τελευταίους 6 μήνες. Πιο συγκεκριμένα, στον οριζόντιο άξονα του διαγράμματος, παρατηρούμε ότι τα προβλήματα είναι χωρισμένα σε μήνες. Με σκούρο γκρι χρώμα, εμφανίζονται τα προβλήματα που είναι Σε Αναμονή, ενώ με ανοιχτό γαλάζιο αυτά που έχουν Ολοκληρωθεί. Τέλος, αν ο χρήστης οδηγήσει το ποντίκι πάνω σε μία μπάρα, θα εμφανιστεί ένα tooltip, στο οποίο θα αναγράφεται ο αριθμός των προβλημάτων.

Στα δεξιά παρατηρούμε τρία ραβδογράμματα, το καθένα από αυτά αναφέρεται σε μία από τις τρεις καταστάσεις που μπορεί να έχει ένα πρόβλημα. Πιο συγκεκριμένα, στο πάνω ραβδόγραμμα, εμφανίζονται τα προβλήματα των τελευταίων 6 μηνών όλων των Δήμων, με κατάσταση «Σε Αναμονή». Στο μεσαίο αυτά με κατάσταση «Σε Επεξεργασία» και στο τελευταίο αυτά με κατάσταση «Ολοκληρώθηκαν». Όπως θα παρατηρήσετε, πάνω δεξιά από κάθε ραβδόγραμμα εμφανίζεται ένα ποσοστό. Αυτό αναφέρεται στο ποσοστό των προβλημάτων της κάθε κατάστασης στο σύνολο των προβλημάτων. Τέλος, αν ο χρήστης οδηγήσει το ποντίκι πάνω σε μία μπάρα, θα εμφανιστεί ένα tooltip, στο οποίο θα αναγράφεται ο αριθμός των προβλημάτων.

Στο κάτω μέρος της οθόνης εμφανίζεται ένας χάρτης με πολλές πινέζες σε διάφορα σημεία της πόλης. Η κάθε πινέζα αντιστοιχεί σε ένα συγκεκριμένο πρόβλημα. Αν ο χρήστης πατήσει πάνω σε κάποια πινέζα, θα εμφανιστεί ένα popup παράθυρο, το οποίο θα περιλαμβάνει μερικές πληροφορίες για το συγκεκριμένο πρόβλημα, καθώς κι ένα hyperlink με τίτλο «Προβολή Προβλήματος». Σε περίπτωση που ο χρήστης το πατήσει, θα μεταβεί στην αντίστοιχη σελίδα του προβλήματος.



Εικόνα 7: Χάρτης Εμφάνισης Προβλημάτων Αρχικής Σελίδας

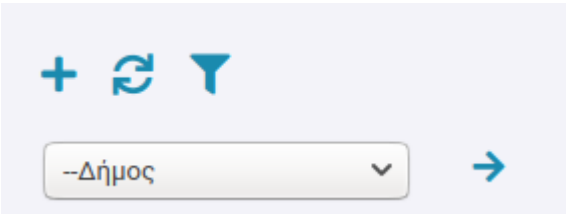


Εικόνα 8: Προβολή Προβλήματος Χάρτη

3.2.5 Χρήστες

Στη συγκεκριμένη οθόνη υπάρχει ένα πίνακας, στον οποίο εμφανίζονται όλοι οι διαχειριστές. Η οθόνη αυτή είναι διαθέσιμη μόνο για τους Επιβλέποντες Διαχειριστές. Πιο συγκεκριμένα, αν ο χρήστης που έχει εισέλθει στην εφαρμογή έχει τον ρόλο του Επιβλέποντα Διαχειριστή, τότε σε αυτόν εμφανίζονται οι Διαχειριστές όλων των δήμων, καθώς και οι υπόλοιποι Επιβλέποντες Διαχειριστές, εφόσον υπάρχουν. Τα στοιχεία που εμφανίζονται στον πίνακα είναι τα εξής: Ονοματεπώνυμο, Δήμος, email.

Ακριβώς πάνω από τον πίνακα, διακρίνουμε κάποια εικονίδια. Στα αριστερά, υπάρχει ένα κουμπί με το σύμβολο +, το οποίο οδηγεί στην σελίδα δημιουργίας καινούργιου χρήστη. Στην μέση, υπάρχει ένα κουμπί, με το οποίο γίνεται ανανέωση ο πίνακας. Και στα δεξιά, υπάρχει ένα κουμπί, με το οποίο ο χρήστης μπορεί να κάνει φιλτράρισμα τον πίνακα. Πιο συγκεκριμένα, κι όπως θα δείτε στην εικόνα από κάτω, με το που πατήσει ο χρήστης το κουμπί, θα εμφανιστεί ακριβώς από κάτω ένα στοιχείο ελέγχου επιλογής με όλους τους Δήμους. Ο χρήστης μπορεί να επιλέξει έναν Δήμο και στη συνέχεια ο πίνακας θα ανανεωθεί και θα εμφανίσει μόνο τους διαχειριστές του συγκεκριμένου Δήμου.

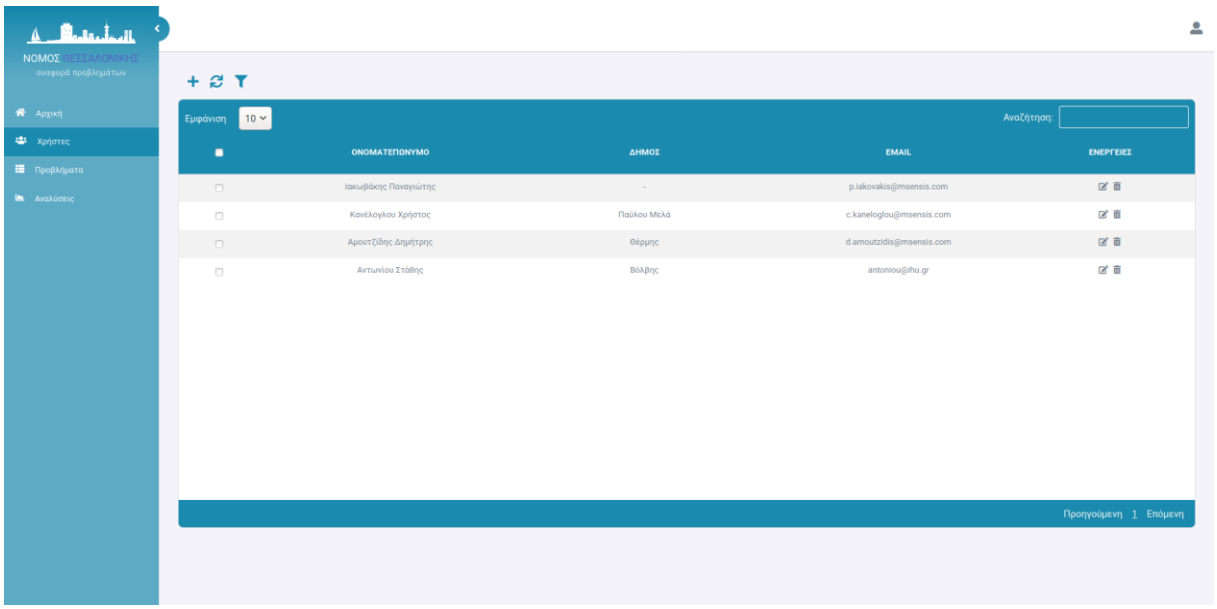


Εικόνα 9: Πρόσθετες λειτουργίες οθόνης Χρηστών

Με τη σειρά του ο πίνακας μας προσφέρει τις εξής δυνατότητες. Αρχικά, όπως παρατηρούμε κάτω δεξιά στον πίνακα υπάρχει σελιδοποίηση. Από προεπιλογή, ο αριθμός των στοιχείων που εμφανίζονται σε κάθε σελίδα του πίνακα είναι δέκα. Σε περίπτωση που ο χρήστης θέλει να αλλάξει αυτόν τον αριθμό, μπορεί κάνοντας τις κατάλληλες ενέργειες. Πάνω αριστερά στον πίνακα, υπάρχει ένα λεκτικό Εμφάνιση κι ακριβώς δίπλα ένα select με κάποιες επιλογές (10, 25, 50, 100). Ο χρήστης με αυτόν τον τρόπο έχει την δυνατότητα να αλλάξει των αριθμό των στοιχείων που θα εμφανίζεται σε κάθε σελίδα του πίνακα.

Στη συνέχεια, μία πολύ σημαντική λειτουργία που προσφέρει ο πίνακας, είναι αυτή της αναζήτησης. Πάνω δεξιά στον πίνακα, υπάρχει ένα input, στο οποίο ο χρήστης μπορεί να γράψει το όνομα ή επώνυμο του διαχειριστή, τον Δήμο στον οποίο ανήκει ή το email του και αυτόματα ο πίνακας να επιστρέψει του χρήστες που αντιστοιχούν στην αναζήτηση.

Επιπλέον, στον διαχειριστή δίνονται και κάποιες ενέργειες για κάθε χρήστη που εμφανίζεται στον πίνακα. Πατώντας το κουμπί , ο χρήστης θα μεταβεί στην σελίδα επεξεργασίας του συγκεκριμένου διαχειριστη. Ενώ πατώντας το κουμπί , θα εμφανιστεί στον χρήστη ένα popup μήνυμα με τίτλο "Επιβεβαίωση Διαγραφής". Σε περίπτωση που ο χρήστης πατήσει το κουμπί Delete, τότε ο συγκεκριμένος Διαχειριστής θα διαγραφεί οριστικά από την εφαρμογή. Τέλος, στον χρήστη δίνεται κι η δυνατότητα της πολλαπλής διαγραφής. Το μόνο που έχει να κάνει, είναι να επιλέξει τους χρήστες που θέλει και στη συνέχεια να πατήσει το κουμπί της διαγραφής, που θα εμφανιστεί ακριβώς πάνω από τον πίνακα.



Εικόνα 10: Στιγμιότυπο οθόνης Σελίδας Χρηστών

3.2.5.1 Δημιουργία Χρήστη

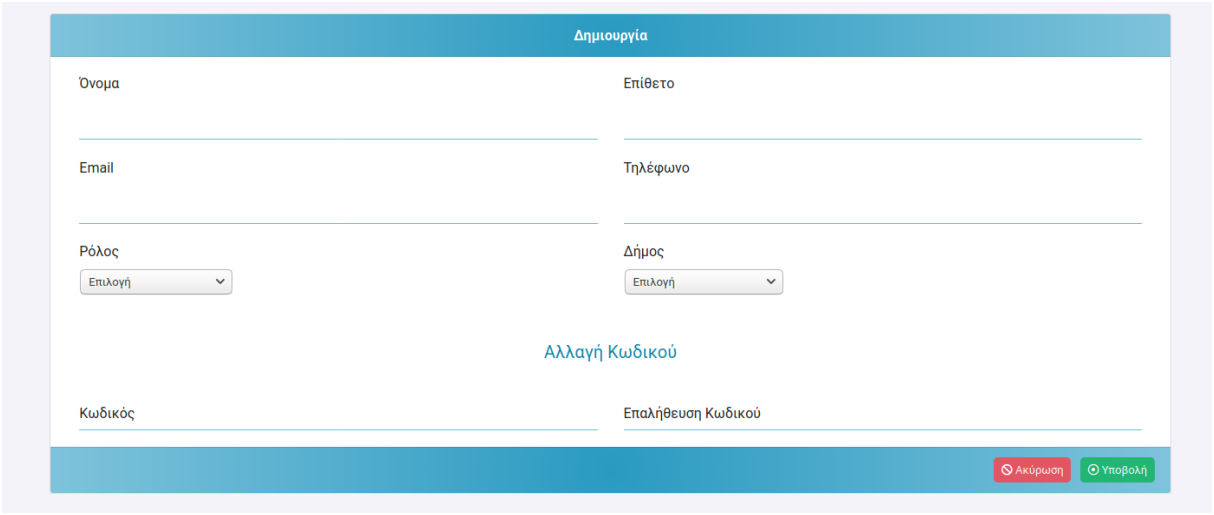
Στην συγκεκριμένη οθόνη, ένας Επιβλέπων Διαχειριστής, έχει την δυνατότητα να δημιουργήσει έναν καινούργιο χρήστη, συμπληρώνοντας κάποια πεδία. Όλα τα πεδία, που εμφανίζονται στην φόρμα, είναι υποχρεωτικά. Σε περίπτωση που δεν συμπληρωθούν, θα εμφανιστεί κατάλληλο μήνυμα σφάλματος.

Τα πεδία λοιπόν είναι τα εξής:

- Όνομα: το όνομα του χρήστη
- Επίθετο: το επίθετο του χρήστη
- Email: το email του χρήστη

- Τηλέφωνο: το τηλέφωνο του χρήστη
- Ρόλος: εδώ υπάρχει ένα select με της εξής επιλογές, Επιβλέπων Διαχειριστής ή Διαχειριστής Δήμου
- Δήμος: το συγκεκριμένο select θα εμφανιστεί εφόσον ο Ρόλος, που έχει επιλεχθεί, είναι Διαχειριστής Δήμου. Είναι ένα select με όλους τους Δήμους της περιφερειακής ενότητας Θεσσαλονίκης
- Κωδικός: κωδικός για τη εισαγωγή του χρήστη στην εφαρμογή. Μπορεί να αλλαχθεί στη συνέχεια
- Επαλήθευση Κωδικού: επαλήθευση του κωδικού που πληκτρολογήθηκε νωρίτερα

Όταν ο χρήστης συμπληρώσει όλα τα πεδία, μπορεί να πατήσει το κουμπί υποβολή, κι έτσι να δημιουργήσει επιτυχώς έναν καινούργιο χρήστη.

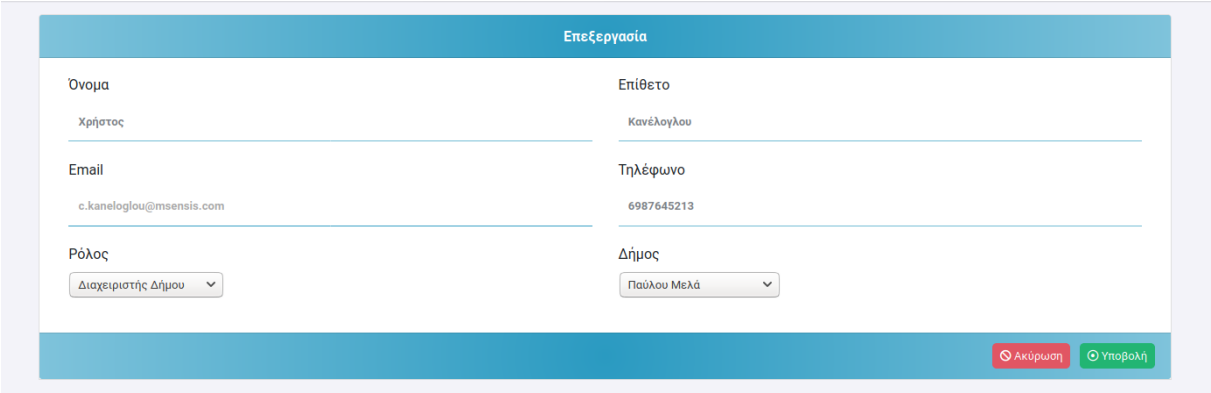


Εικόνα 11: Φόρμα Δημιουργίας Χρήστη

3.2.5.2 Επεξεργασία Χρήστη

Στην συγκεκριμένη οθόνη, ένας Επιβλέπων Διαχειριστής, έχει την δυνατότητα να επεξεργαστεί έναν ήδη υπάρχων χρήστη. Πιο συγκεκριμένα, μπορεί να επεξεργαστεί το όνομα, το επίθετο, το τηλέφωνο και τον ρόλο του χρήστη. Αφού ολοκληρώσει τις αλλαγές που θέλει, μπορεί να πατήσει το κουμπί Υποβολή, και με αυτόν τρόπο να αποθηκεύσει τα καινούργια στοιχεία του χρήστη.

Τέλος, πρέπει να αναφερθεί, ότι στον χρήστη δεν δίνεται η δυνατότητα να αλλάξει το email του χρήστη καθώς και τον κωδικό πρόσβασης του. Ο λόγος είναι διότι μετά ο χρήστης δεν θα μπορεί να έχει πρόσβαση στην εφαρμογή, αν δεν του δοθούν τα καινούργια στοιχεία σύνδεσης.

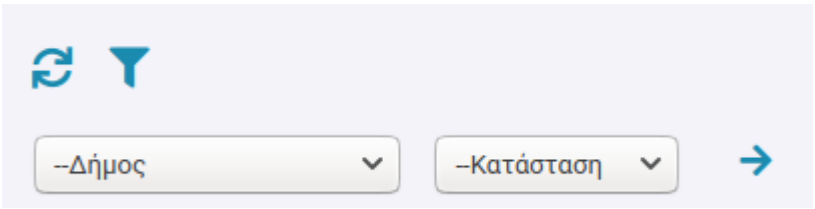


Εικόνα 12: Φόρμα Επεξεργασίας Χρήστη

3.2.6 Προβλήματα

Στη συγκεκριμένη οθόνη μπορεί να έχει πρόσβαση κάποιος χρήστης, είτε είναι Επιβλέπων Διαχειριστής, είτε είναι Διαχειριστής Δήμου. Η μόνη διαφοροποίηση είναι ότι, στους πρώτους θα εμφανίζονται τα προβλήματα όλων των Δήμων, ενώ στους δεύτερους μόνο τα προβλήματα του συγκεκριμένου Δήμου. Για κάθε πρόβλημα εμφανίζονται κάποια συγκεκριμένα στοιχεία στον πίνακα. Τα στοιχεία αυτά είναι τα εξής: περιγραφη, δήμος, ημερομηνία υποβολής, κατάσταση.

Ακριβώς πάνω από τον πίνακα, διακρίνουμε κάποια εικονίδια. Στα αριστερά, υπάρχει ένα κουμπί, με το οποίο γίνεται ανανέωση ο πίνακας. Και στα δεξιά, υπάρχει ένα κουμπί, με το οποίο ο χρήστης μπορεί να κάνει φιλτράρισμα τον πίνακα. Πιο συγκεκριμένα, κι όπως θα δείτε στην εικόνα από κάτω, με το που πατήσει ο χρήστης το κουμπί, θα εμφανιστεί ακριβώς από κάτω ένα select με όλους τους Δήμους και ένα select με τις διαθέσιμες καταστάσεις που μπορεί να έχει ένα πρόβλημα. Ο χρήστης μπορεί να χρησιμοποιήσει τα select, είτε συνδυαστικά π.χ. Δήμος=Θεσσαλονίκης και Κατάσταση=Ολοκληρώθηκε, το οποίο θα πρέπει να εμφανίσει στον πίνακα όλα τα προβλήματα που έχουν αναφερθεί στον Δήμο Θεσσαλονίκης και έχουν ολοκληρωθεί, είτε μεμονωμένα, δηλαδή να κάνει φιλτράρισμα ή με βάση τον δήμο ή με βάση την κατάσταση του προβλήματος.

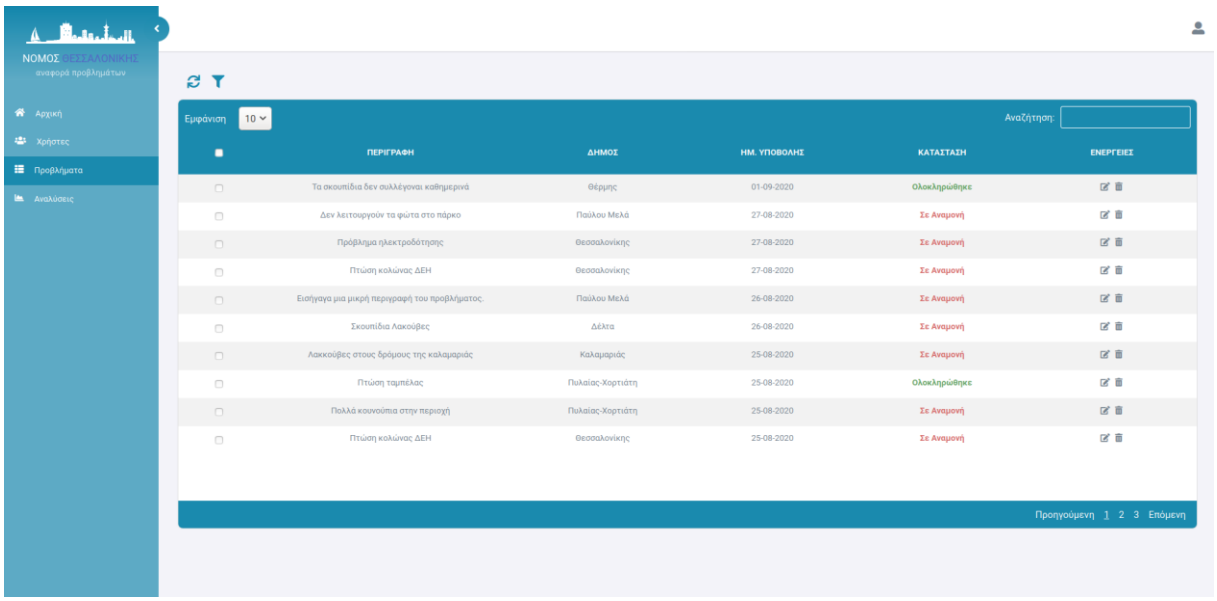


Εικόνα 13: Πρόσθετες λειτουργίες οθόνης Προβλημάτων

Με τη σειρά του ο πίνακας μας προσφέρει τις εξής δυνατότητες. Αρχικά, όπως παρατηρούμε κάτω δεξιά στον πίνακα υπάρχει σελιδοποίηση. Από προεπιλογή, ο αριθμός των στοιχείων που εμφανίζονται σε κάθε σελίδα του πίνακα είναι δέκα. Σε περίπτωση που ο χρήστης θέλει να αλλάξει αυτόν τον αριθμό, μπορεί κάνοντας τις κατάλληλες ενέργειες. Πάνω αριστερά στον πίνακα, υπάρχει ένα λεκτικό Εμφάνιση κι ακριβώς δίπλα ένα select με κάποιες επιλογές (10, 25, 50, 100). Ο χρήστης με αυτόν τον τρόπο έχει την δυνατότητα να αλλάξει των αριθμό των στοιχείων που θα εμφανίζεται σε κάθε σελίδα του πίνακα.

Στη συνέχεια, μία πολύ σημαντική λειτουργία που προσφέρει ο πίνακας, είναι αυτή της αναζήτησης. Πάνω δεξιά στον πίνακα, υπάρχει ένα input, στο οποίο ο χρήστης μπορεί να γράψει την περιγραφή του προβλήματος ή τον δήμο στον οποίο ανήκει ή την ημερομηνία που καταχωρήθηκε ή την κατάσταση του και αυτόματα ο πίνακας να επιστρέψει τα προβλήματα που αντιστοιχούν στην αναζήτηση.

Επιπλέον, στον διαχειριστή δίνονται και κάποιες ενέργειες για κάθε πρόβλημα που εμφανίζεται στον πίνακα. Πατώντας το κουμπί  , ο χρήστης θα μεταβεί στην σελίδα επεξεργασίας του συγκεκριμένου προβλήματος. Ενώ πατώντας το κουμπί  , θα εμφανιστεί στον χρήστη ένα popup μήνυμα με τίτλο "Επιβεβαίωση Διαγραφής". Σε περίπτωση που ο χρήστης πατήσει το κουμπί Delete, τότε το συγκεκριμένο πρόβλημα θα διαγραφεί οριστικά από την εφαρμογή. Τέλος, στον χρήστη δίνεται κι η δυνατότητα της πολλαπλής διαγραφής. Το μόνο που έχει να κάνει, είναι να επιλέξει τα προβλήματα που θέλει και στη συνέχεια να πατήσει το κουμπί της διαγραφής, που θα εμφανιστεί ακριβώς πάνω από τον πίνακα.



Εικόνα 14: Στιγμιότυπο οθόνης Σελίδας Προβλημάτων

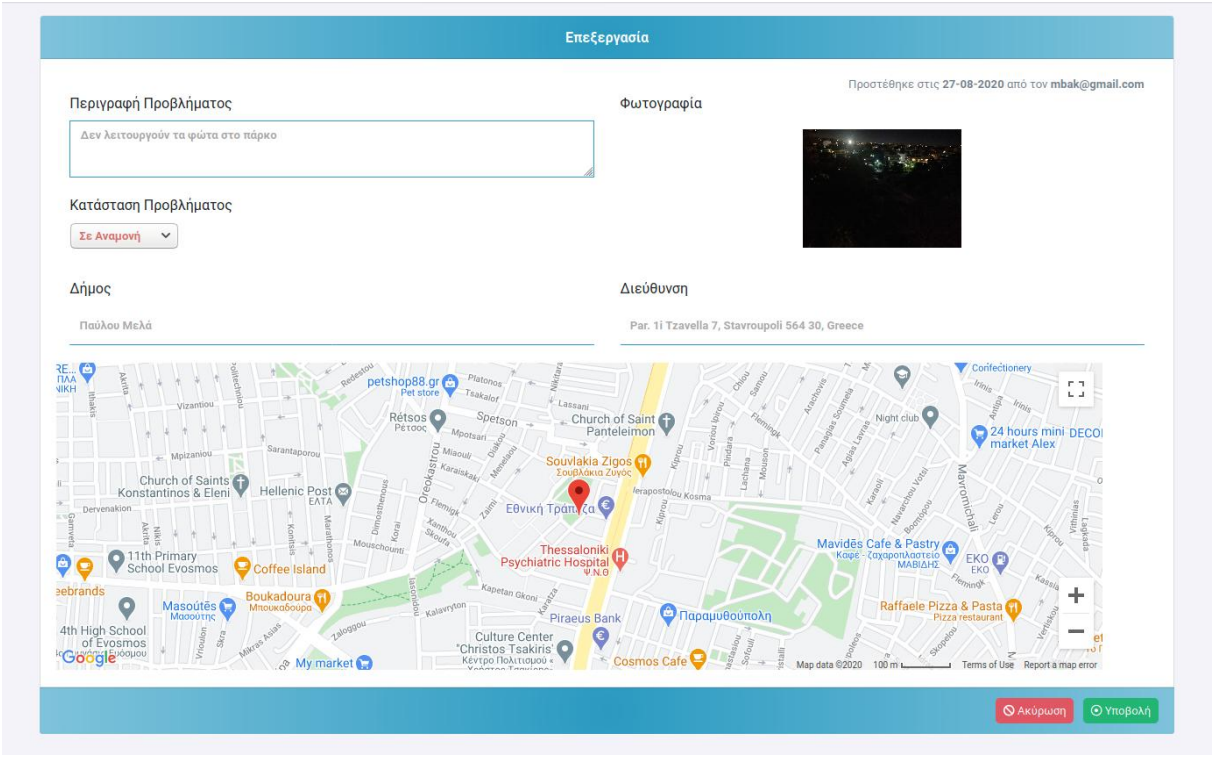
3.2.6.1 Επεξεργασία Προβλήματος

Στην συγκεκριμένη οθόνη, ένας Διαχειριστής έχει την δυνατότητα να δει κάποια στοιχεία του προβλήματος που έχει αναφερθεί, έτσι ώστε να προχωρήσει στην επίλυση του. Τα στοιχεία που εμφανίζονται στην οθόνη είναι τα εξής:

- Περιγραφή προβλήματος: εδώ εμφανίζεται μια μικρή περιγραφή του προβλήματος
- Κατάσταση: η κατάσταση του προβλήματος, η οποία μπορεί να είναι «Σε Αναμονή», «Σε Επεξεργασία» ή «Ολοκληρώθηκε»
- Δήμος: ο δήμος στον οποίο ανήκει το πρόβλημα
- Διεύθυνση: η διεύθυνση που παρατηρήθηκε το πρόβλημα
- Φωτογραφία: εδώ εμφανίζεται μία φωτογραφία του προβλήματος σε περίπτωση που υπάρχει. Επιπλέον υπάρχει η δυνατότητα μεγέθυνσης της φωτογραφίας, απλά κάνοντας κλικ πάνω της

- Χάρτης: στο κάτω μέρος της φόρμας παρατηρείται ένας χάρτης, στον οποίο με πινέζα εμφανίζεται η τοποθεσία του προβλήματος

Ο Διαχειριστής μπορεί να επεξεργαστεί μόνο την κατάσταση του προβλήματος. Πιο συγκεκριμένα, ένα πρόβλημα με το που δηλώνεται, έχει ως προεπιλογή την κατάσταση του Σε Αναμονή. Όταν ο Διαχειριστής δώσει εντολή σε κάποιον αρμόδιο, να ξεκινήσει η επίλυση του προβλήματος, τότε η κατάσταση του θα πρέπει να αλλαχθεί Σε Επεξεργασία. Τέλος, όταν η επίλυση του προβλήματος ολοκληρωθεί, θα πρέπει να αλλάξει η κατάσταση του σε Ολοκληρώθηκε. Κάθε φορά που αλλάζει η κατάσταση, ο Διαχειριστής θα πρέπει να πατάει το κουμπί Υποβολή, έτσι ώστε να αποθηκευτεί η καινούργια κατάσταση.



Εικόνα 15: Φόρμα Επεξεργασίας Προβλήματος

3.2.7 Αναλύσεις

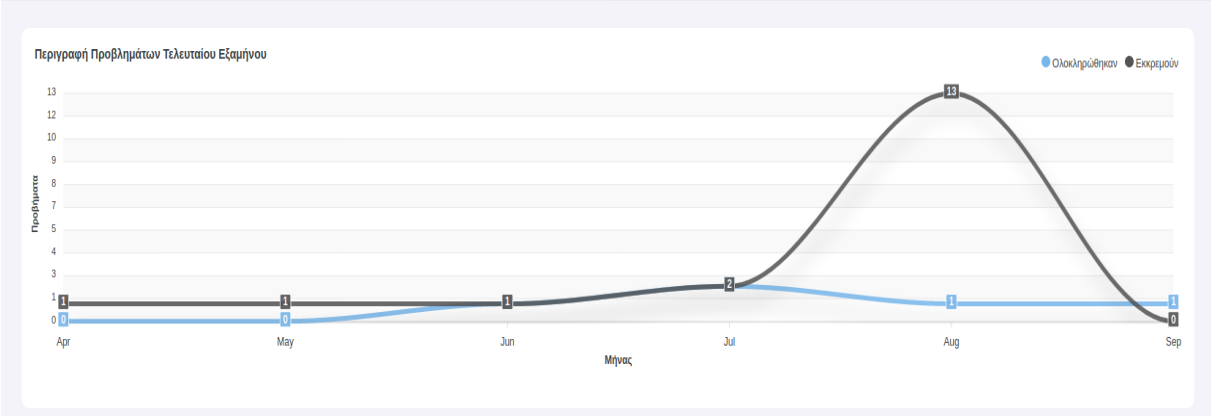
Στη συγκεκριμένη οθόνη εμφανίζονται κάποια στατιστικά σχετικά με τα προβλήματα. Ένας χρήστης, είτε αυτός είναι Επιβλέπων Διαχειριστής, είτε είναι Διαχειριστής Δήμου, έχει πρόσβαση στην συγκεκριμένη οθόνη. Η μόνη διαφοροποίηση είναι ότι ένας Επιβλέπων Διαχειριστής, έχει στη διάθεση του στατιστικά για τα προβλήματα όλων των δήμων, ενώ ένας Διαχειριστής Δήμου μόνο για τα προβλήματα του δήμου του.

Η οθόνη των Αναλύσεων χωρίζεται σε τρία κομμάτια. Κάθε κομμάτι αναπαριστά και μια διαφορετική συλλογή δεδομένων. Παρακάτω θα αναλύσουμε ξεχωριστά το κάθε κομμάτι.

Στο πάνω μέρος των Αναλύσεων, υπάρχει ένα διάγραμμα γραμμής, το οποίο αναπαριστά τα προβλήματα του τελευταίου Εξαμήνου. Στην περίπτωση του Επιβλέπων Διαχειριστή, εμφανίζονται τα προβλήματα

όλων των δήμων, ενώ στην περίπτωση ενός Διαχειριστή Δήμου εμφανίζονται τα προβλήματα μόνο του συγκεκριμένου δήμου.

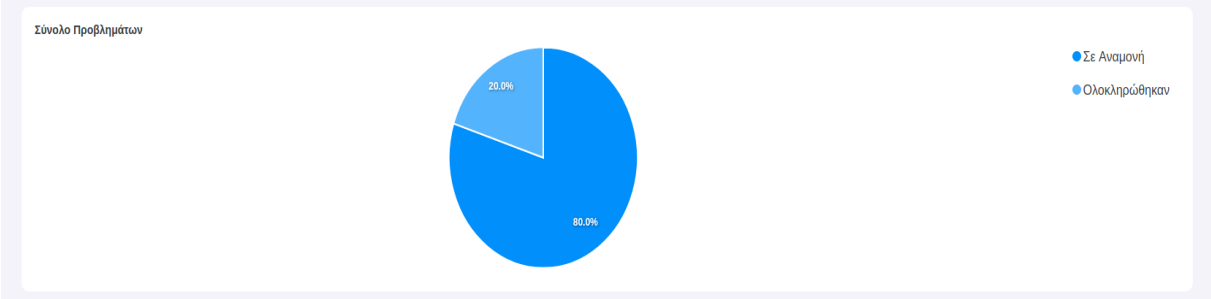
Πιο συγκεκριμένα, παρατηρούμε ότι ο οριζόντιος άξονας του διαγράμματος είναι χωρισμένος σε μήνες, ενώ στον άξονα y εμφανίζεται ο αριθμός των προβλημάτων. Με σκούρο γκρι χρώμα, εμφανίζονται τα προβλήματα που εκκρεμούν, ενώ με ανοιχτό γαλάζιο αυτά που έχουν ολοκληρωθεί. Τέλος, αν ο χρήστης οδηγήσει το ποντίκι πάνω σε μία γραμμή του διαγράμματος, θα εμφανιστεί ένα tooltip, στο οποίο θα αναγράφεται ο αριθμός των προβλημάτων που εκκρεμούν και ολοκληρώθηκαν τον συγκεκριμένο μήνα.



Εικόνα 16: Γράφημα Προβολής Προβλημάτων Τελευταίου Εξαμήνου

Στο μεσαίο μέρος των Αναλύσεων, υπάρχει ένα διάγραμμα πίτας, το οποίο αναπαριστά το σύνολο των προβλημάτων που έχουν αναφερθεί ως τώρα. Στην περίπτωση του Επιβλέπων Διαχειριστή, εμφανίζονται τα προβλήματα όλων των δήμων, ενώ στην περίπτωση ενός Διαχειριστή Δήμου εμφανίζονται τα προβλήματα μόνο του συγκεκριμένου δήμου.

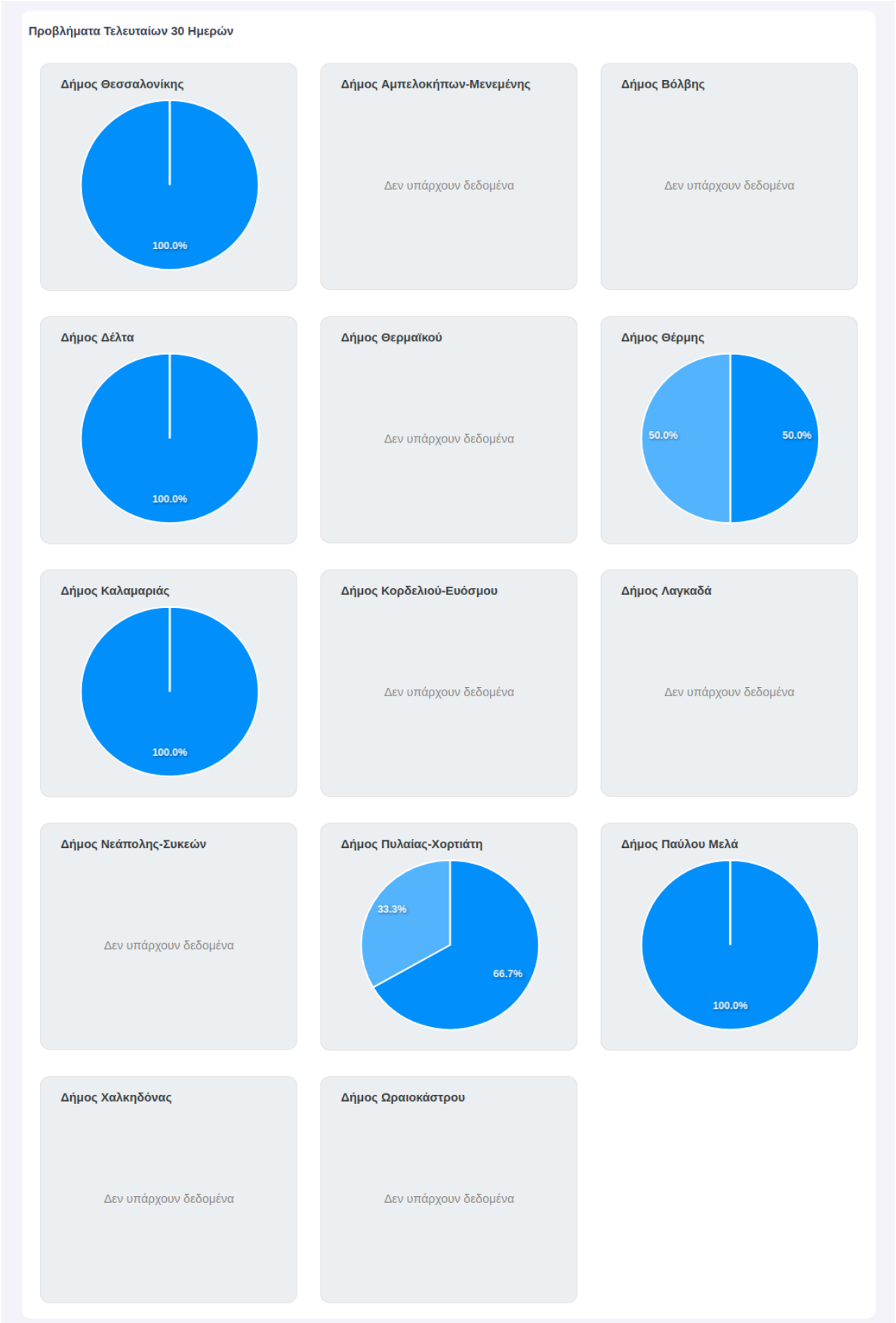
Πιο συγκεκριμένα, με σκούρο μπλε χρώμα, εμφανίζονται τα προβλήματα που είναι Σε Αναμονή, ενώ με ανοιχτό γαλάζιο αυτά που έχουν Ολοκληρωθεί. Πάνω στην πίτα εμφανίζεται το ποσοστό της κάθε κατηγορίας με το σύνολο των προβλημάτων. Τέλος, αν ο χρήστης οδηγήσει το ποντίκι πάνω στην πίτα, θα εμφανιστεί ένα tooltip, στο οποίο θα αναγράφεται ο αριθμός των προβλημάτων που αντιστοιχούν στην συγκεκριμένη κατηγορία.



Εικόνα 17: Γράφημα Προβολής του Συνόλου των Προβλημάτων

Στο κάτω μέρος των Αναλύσεων, εμφανίζονται πολλά διαγράμματα πίτας. Το κάθε διάγραμμα αναφέρεται σε έναν συγκεκριμένο δήμο, κι αναπαριστά τα προβλήματα των τελευταίων 30 ημερών. Στην περίπτωση του Επιβλέπων Διαχειριστή, εμφανίζονται διαγράμματα πίτας για όλους τους δήμους, ενώ στην περίπτωση ενός Διαχειριστή Δήμου εμφανίζεται ένα μόνο διάγραμμα πίτας, που αφορά μόνο τον συγκεκριμένο δήμο.

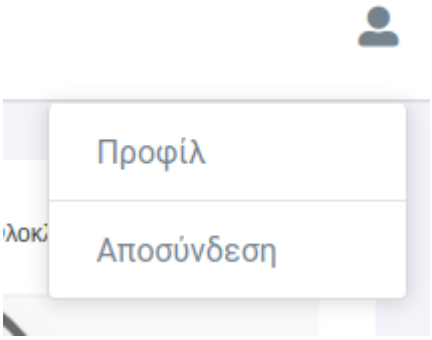
Πιο συγκεκριμένα, με σκούρο μπλε χρώμα, εμφανίζονται τα προβλήματα που είναι Σε Αναμονή, ενώ με ανοιχτό γαλάζιο αυτά που έχουν Ολοκληρωθεί. Πάνω στην πίτα εμφανίζεται το ποσοστό της κάθε κατηγορίας με το σύνολο των προβλημάτων. Τέλος, αν ο χρήστης οδηγήσει το ποντίκι πάνω στην πίτα, θα εμφανιστεί ένα tooltip, στο οποίο θα αναγράφεται ο αριθμός των προβλημάτων που αντιστοιχούν στην συγκεκριμένη κατηγορία.



Εικόνα 18: Γραφήματα Προβολής Προβλημάτων των Τελευταίων 30 Ημερών

3.2.8 Μενού Χρήστη

Στο πάνω δεξιά μέρος της εφαρμογής υπάρχει το εξής εικονίδιο , το οποίο αν πατήσει ο χρήστης, θα εμφανιστεί ένα μενού. Στο μενού αυτό υπάρχουν δύο στοιχεία. Το πρώτο ονομάζεται Προφίλ και το δεύτερο ονομάζεται Αποσύνδεση.



Εικόνα 19: Μενού Χρήστη

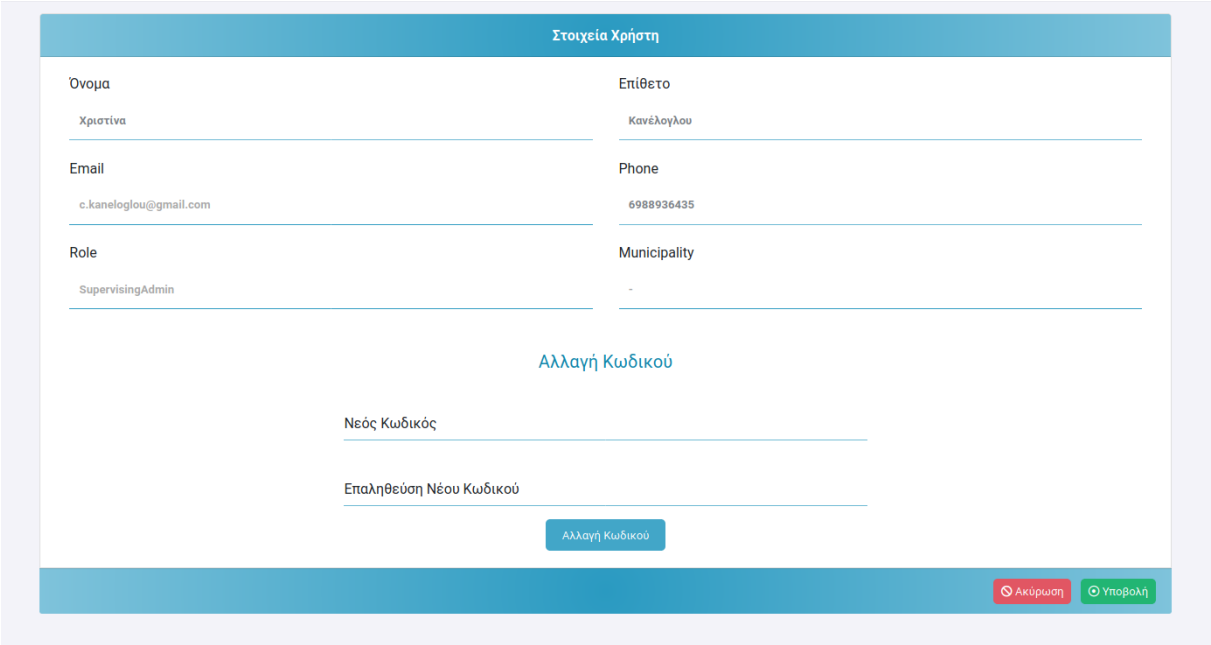
3.2.8.1 Προφίλ

Από το μενού που αναφέρθηκε προηγουμένως, αν ο χρήστης πατήσει το Προφίλ, θα πλοηγηθεί στο προφίλ του. Στην συγκεκριμένη οθόνη εμφανίζονται λεπτομερώς τα στοιχεία του χρήστη και του δίνεται η δυνατότητα να τα επεξεργαστεί.

Πιο συγκεκριμένα, ένας χρήστης έχει την δυνατότητα να αλλάξει μόνο το όνομα, το επίθετο και το τηλέφωνο του. Τον ρόλο και τον δήμο, μπορεί να τα επεξεργαστεί μόνο κάποιος Επιβλέπων Διαχειριστής για έναν χρήστη. Επιπλέον, η αλλαγή του email δεν επιτρέπεται.

Αφού κάποιος χρήστη κάνει τις αλλαγές που θέλει, πρέπει να πατήσει το κουμπί Υποβολή για να ολοκληρώσει την αποθήκευση των νέων στοιχείων. Σε περίπτωση επιτυχίας, θα εμφανιστεί κατάλληλο μήνυμα. Σε αντίθετη περίπτωση, θα εμφανιστεί μήνυμα σφάλματος.

Τέλος, στην συγκεκριμένη οθόνη, παρέχεται επίσης η δυνατότητα στον χρήστη να αλλάξει τον κωδικό πρόσβασης του στο σύστημα. Για να γίνει αυτό πρέπει να συμπληρώσει τα πεδία Νέος Κωδικός και Επαλήθευση Νέου Κωδικού, και στην συνέχεια να πατήσει το κουμπί Αλλαγή Κωδικού. Ο χρήστης θα ενημερωθεί για την επιτυχία αλλαγής κωδικού με κατάλληλο μήνυμα. Σε περίπτωση σφάλματος, θα εμφανιστεί αντίστοιχο μήνυμα.



Εικόνα 20: Προφίλ Χρήστη

3.2.8.2 Αποσύνδεση

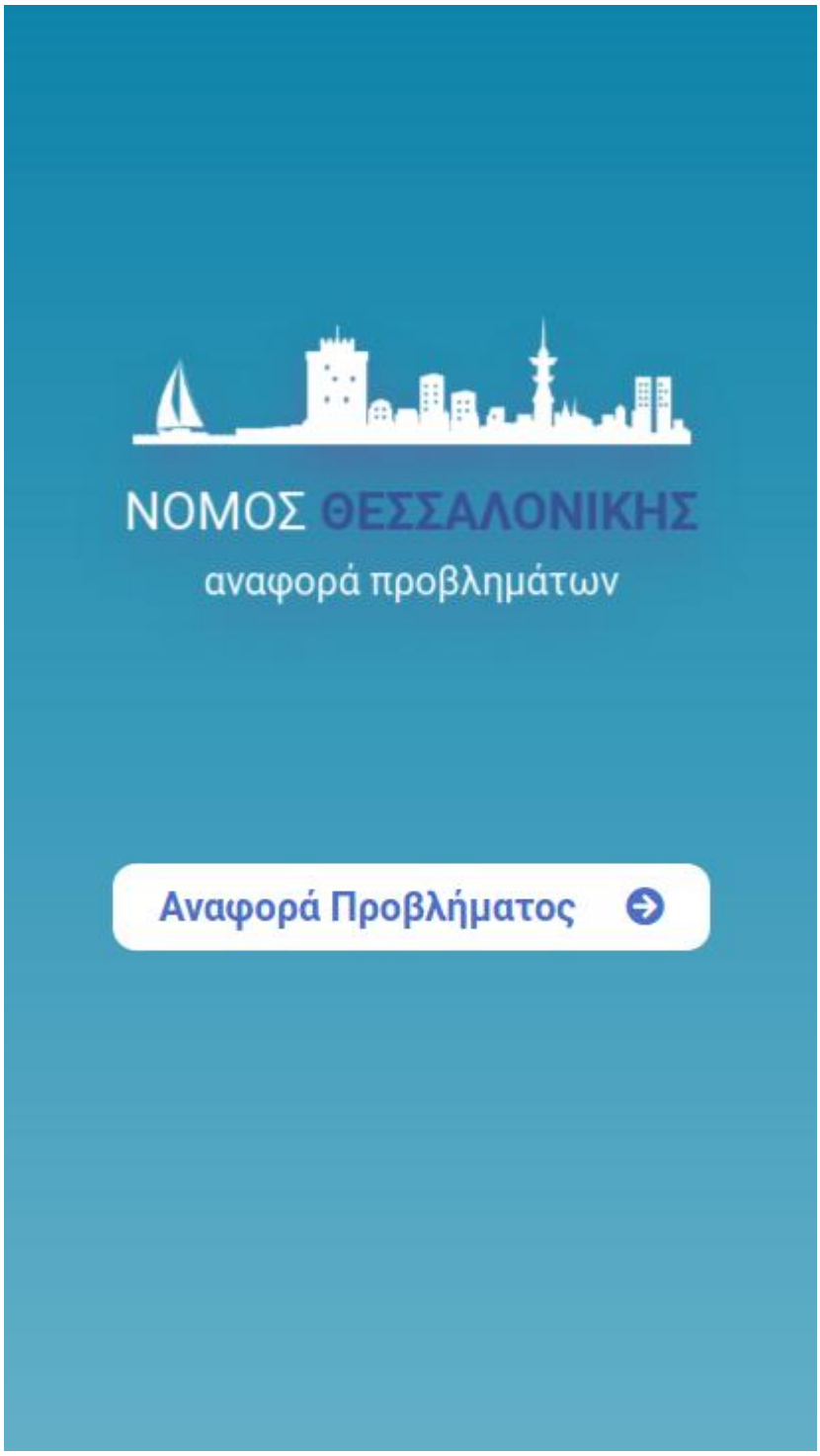
Για να κάνει αποσύνδεση ο χρήστης από την εφαρμογή, θα πρέπει να πατήσει παω στην Αποσύνδεση στο μενού που αναφέρθηκε προηγουμένως.

3.3 Αναφορά Προβλήματος

Σε αυτό το υποκεφάλαιο θα περιγράψουμε το κομμάτι της εφαρμογής, που αναφέρεται στην αναφορά κάποιου προβλήματος. Ο κάθε πολίτης θα μπορεί να έχει πρόσβαση σε μία συγκεκριμένη σελίδα, στην οποία θα αναφέρει κάποιο πρόβλημα που έχει παρατηρήσει σε κάποιο Δήμο.

3.3.1 Αρχική σελίδα αναφοράς προβλήματος

Με την είσοδο του χρήστη στην σελίδα αναφοράς προβλημάτων, εμφανίζεται σε πρώτη φάση μία αρχική σελίδα. Σε αυτήν διακρίνουμε στο πάνω μέρος το λογότυπο της εφαρμογής κι ακριβώς από κάτω ένα κουμπί, με το κείμενο: Αναφορά Προβλήματος. Όταν ο χρήστης πατήσει το συγκεκριμένο κουμπί, θα μεταφερθεί στην σελίδα όπου θα μπορεί να αναφέρει κάποιο καινούργιο πρόβλημα.



Εικόνα 21: Αρχική σελίδα αναφοράς προβλήματος

3.3.2 Σελίδα αναφοράς προβλήματος

Στην συγκεκριμένη οθόνη, ένας πολίτης έχει την δυνατότητα να υποβάλει ένα καινούργιο πρόβλημα, το οποίο στη συνέχεια θα εμφανιστεί στο διαχειριστικό περιβάλλον της εφαρμογής. Τα στοιχεία που εμφανίζονται στην οθόνη, και τα οποία πρέπει να συμπληρώσει κάποιος για ολοκληρώσει την υποβολή ενός προβλήματος, είναι τα εξής:

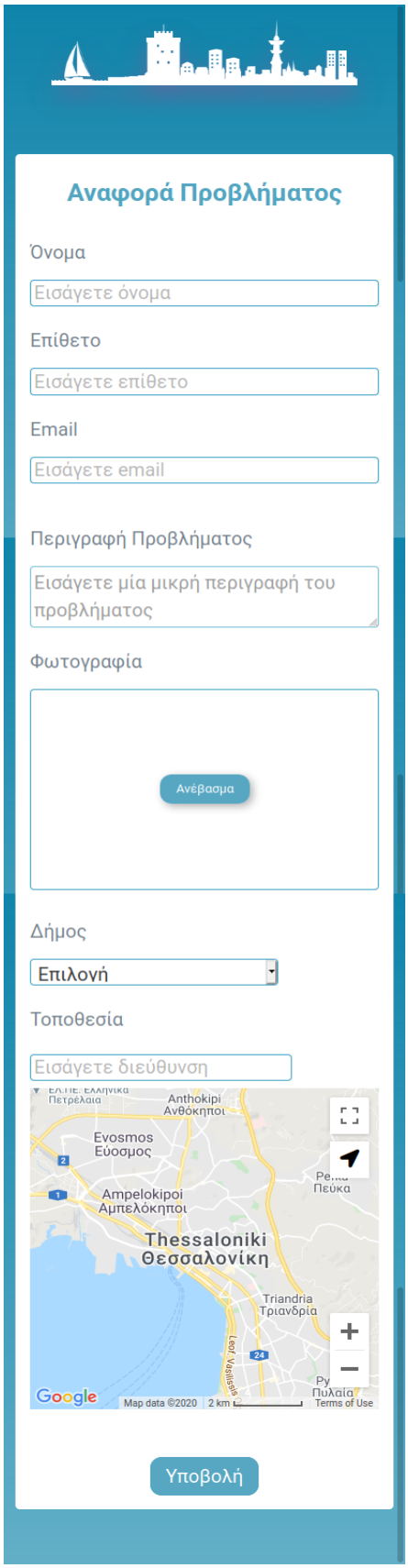
- Όνομα: το όνομα του χρήστη
- Επίθετο: το επίθετο του χρήστη
- Email: το email του χρήστη
- Περιγραφή προβλήματος: εδώ ο χρήστης πρέπει να εισάγει μία μικρή περιγραφή του προβλήματος
- Φωτογραφία: ο χρήστης αν θέλει μπορεί να ανεβάσει μια φωτογραφία, για να γίνει πιο κατανοητό το πρόβλημα (δεν είναι υποχρεωτικό)
- Δήμος: ο χρήστης πρέπει να επιλέξει τον δήμο, στον οποίο ανήκει το πρόβλημα, από μία λίστα
- Τοποθεσία: ο χρήστης πρέπει να συμπληρώσει την τοποθεσία που παρατηρείται το πρόβλημα

Σχετικά με την τοποθεσία, στον χρήστη δίνονται οι εξής οι δυνατότητες:

Η πρώτη δυνατότητα είναι πως, ο χρήστης μπορεί να συμπληρώσει την τοποθεσία γράφοντας στο αντίστοιχο πεδίο την διεύθυνση. Την ώρα που πληκτρολογεί, θα παρατηρήσει πως ακριβώς από κάτω, θα εμφανιστεί ένα παραθυράκι, στο οποίο θα εμφανίζονται κάποιες δυνατές επιλογές. Όταν επιλέξει αυτή που θέλει, στον χάρτη θα εμφανιστεί μία πινέζα στην επιθυμητή τοποθεσία.

Η δεύτερη δυνατότητα, είναι η λήψη της τοποθεσίας που βρίσκεται τώρα ο χρήστης. Αυτό γίνεται πατώντας το κουμπί , το οποίο βρίσκεται στο πάνω δεξιά μέρος του χάρτη. Με το που ο χρήστης το πατήσει, θα εμφανιστεί ένα μήνυμα αποδοχής της τοποθεσίας. Με το που το αποδεχθεί, θα εμφανιστεί στον χάρτη μία πινέζα, στην τοποθεσία που βρίσκεται τη δεδομένη στιγμη.

Η τρίτη και τελευταία δυνατότητα, είναι ο χρήστης να πατήσει μέσα στον χάρτη, στην τοποθεσία που θέλει. Με το που το κάνει αυτό, θα εμφανιστεί μία πινέζα στο συγκεκριμένο σημείο και ταυτόχρονα θα συμπληρωθεί το πεδίο με την διεύθυνση.



Εικόνα 22: Σελίδα αναφοράς προβλήματος

3.4 Επίλογος

Σε αυτό το κεφάλαιο έγινε μια αναλυτική παρουσίαση της λειτουργίας και χρήσης της εφαρμογής με χρήση στιγμιοτύπων οθόνης (screenshots) και αναλυτική περιγραφή του καθενός. Ακολουθούν τα συμπεράσματα και οι μελλοντικές επεκτάσεις της εφαρμογής.

Κεφάλαιο 4ο: Συμπεράσματα και προτάσεις βελτίωσης

4.1 Συμπεράσματα

Στην παρούσα πτυχιακή εργασία, αναλύθηκε λεπτομερώς, μια web εφαρμογή αναφοράς και διαχείρισης προβλημάτων στην περιφερειακή ενότητα της Θεσσαλονίκης. Για την υλοποίηση της χρησιμοποιήθηκαν βασικές web τεχνολογίες, όπως HTML, CSS, JS, καθώς και βιβλιοθήκες όπως η Backbone.js, και πλατφόρμες όπως το Firebase. Αποκτήθηκαν σημαντικές γνώσεις, τόσο στο κομμάτι της οπτικής σχεδίασης, όσο και στο κομμάτι της διασύνδεσης του, με τον εξυπηρετητή και την βάση δεδομένων.

Το web είναι αναμφισβήτητα κυρίαρχο στην καθημερινότητα όλων μας. Αυτό κάνει την δημιουργία web εφαρμογών, η οποία με τη σειρά τους θα επιλύσει μια σειρά από καθημερινά προβλήματα των πολιτών, να κερδίζει δημοτικότητα. Αξίζει λοιπόν, να εμβαθύνουμε και να αφιερώσουμε χρόνο για τη υλοποίηση τέτοιων εφαρμογών, καθώς θα έχουν άμεσο αντίκτυπο στην βελτίωση της καθημερινής μας ζωής.

4.2 Προτάσεις Βελτίωσης

Ο τρόπος με τον οποίο έχει υλοποιηθεί η εν λόγω εφαρμογή, επιτρέπει την προσθήκη νέων χαρακτηριστικών:

- Στις κατηγορίες των χρηστών του διαχειριστικού κομματιού της εφαρμογής, θα μπορούσε να προστεθεί μία ακόμα κατηγορία. Πρόκειται για την κατηγορία του Επιβλέποντος Διαχειριστή Δήμου. Ουσιαστικά, οι χρήστες αυτού του τύπου, θα μπορούν να δημιουργούν καινούργιους διαχειριστές για τον συγκεκριμένο δήμο και να διαχειρίζονται αυτούς του διαχειριστές.
- Όσον αφορά τα προβλήματα θα μπορούσε να προστεθεί μία ακόμα κατηγορία στην κατάσταση τους. Αυτή η κατηγορία θα αναφέρεται στα προβλήματα, που δηλώνονται αλλά δεν ισχύουν. Ως τώρα στους διαχειριστές, δίνεται η δυνατότητα να διαγράφουν κάποιο πρόβλημα που δεν υπάρχει. Με την προσθήκη της νέας κατηγορίας, θα μπορέσει να γίνει καταμέτρηση των ψευδών προβλημάτων.
- Το Firebase θα μπορούσε να αντικατασταθεί από ένα πιο ολοκληρωμένο back-end σύστημα, το οποίο θα προσδίδει περισσότερες λειτουργίες. Για παράδειγμα, το back-end μπορεί να υλοποιηθεί με τη χρήση του Spring Framework.
- Η εφαρμογή θα μπορούσε να επεκταθεί, έτσι ώστε να είναι εφικτή η αναφορά και διαχείριση προβλημάτων σε όλη την χώρα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία

[1] Jon Duckett: HTML & CSS: Design and Build Web Sites. 2011

[2] Andy Budd, Emil Bjorklund: CSS Mastery.

[3] Vadim Mirgorod: Backbone.js Cookbook. 2013

[4] Jon Duckett: JavaScript and JQuery: Interactive Front-End Web Development. 2014

[5] David Sawyer McFarland: JavaScript and jQuery: The Missing Manual. 2014

[6] Robin Wieruch: The Road to React with Firebase: Your journey to master advanced React for business web applications. 2019

[7] David Herron: Node.js Web Development: Server-side web development made easy with Node 14 using practical examples. 2020

Internet Site

[8] w3scools.com, “HTML Tutorial”. [Online]. Available: https://www.w3schools.com/html/

[9] Wikipedia, “CSS”. [Online]. Available: https://en.wikipedia.org/wiki/CSS

[10] w3scools.com, “CSS Tutorial”. [Online]. Available: https://www.w3schools.com/css/

[11] JavaScript.com, “JavaScript” ”. [Online]. Available: https://www.javascript.com/

[12] Wikipedia, “JavaScript”. [Online]. Available: https://en.wikipedia.org/wiki/JavaScript

[13] w3scools.com, “JavaScript Tutorial”. [Online]. Available: https://www.w3schools.com/js/

[14] jquery.com, “jQuery, write less, do more.” ”. [Online]. Available: https://www.jquery.com/

[15] w3scools.com, “jQuery Tutorial”. [Online]. Available: https://www.w3schools.com/jquery/

[16] backbone.js, “Backbone.js”. [Online]. Available: https://www.backbonejs.org

[17] Wikipedia, “Backbone.js”. [Online]. Available: https://en.wikipedia.org/wiki/Backbone.js

[18] node.js, “Node.js”. [Online]. Available: https://www.nodejs.org

[19] Wikipedia, “Node.js”. [Online]. Available: https://en.wikipedia.org/wiki/Node.js

[20] Firebase, “Firebase helps mobile and web app teams succeed”. [Online]. Available: https://firebase.google.com/

[21] Wikipedia, “Firebase”. [Online]. Available: https://en.wikipedia.org/wiki/Firebase

[22] Wikipedia, “IntelliJ IDEA”. [Online]. Available: https://en.wikipedia.org/wiki/IntelliJ_IDEA

[23] JET BRAINS, “IntelliJ IDEA”. [Online]. Available: https://www.jetbrains.com/idea/

[24] FileZilla, “FileZilla The free FTP Solution”. [Online]. Available: https://filezilla-project.org/

[25] Wikipedia, “BitBucket”. [Online]. Available: https://en.wikipedia.org/wiki/Bitbucket

[26] Atlassian Bitbucket, “Bitbucket”. [Online]. Available: https://bitbucket.org

[27] NGINX, “NGINX wiki”. [Online]. Available: https://www.nginx.com/resources/wiki/

[28] Bootstrap, “Build fast, responsive sites with Bootstrap”. [Online]. Available: https://getbootstrap.com/

[29] Font Awesome, “Font Awesome Icons”. [Online]. Available: https://fontawesome.com/

[30] Handlebars, “handlebars”. [Online]. Available: https://handlebarsjs.com/

[31] APEXCHARTS, “Apexcharts.js, Modern & Interactive Open-source Charts”. [Online]. Available: https://apexcharts.com/

[32] Notify, “Notify.js”. [Online]. Available: https://notifyjs.jpillora.com/

[33] DataTables, “DataTables: Add advanced interaction controls to your HTML tables the free & easy way”. [Online]. Available: https://datatables.net/