



Ι Ο Ν Ι Ο Π Α Ν Ε Π Ι Σ Τ Η Μ Ι Ο Κ Ε Ρ Κ Υ Ρ Α Σ
Σ Χ Ο Λ Η Θ Ε Τ Ι Κ Ω Ν Ε Π Ι Σ Τ Η Μ Ω Ν
Τ Μ Η Μ Α Π Λ Η Ρ Ο Φ Ο Ρ Ι Κ Η Σ

Π Τ Υ Χ Ι Α Κ Η
Ε Ρ Γ Α Σ Ι Α

ΕΦΑΡΜΟΓΗ WEB ΓΙΑ ΤΗΝ ΑΜΦΙΔΡΟΜΗ ΕΠΙΚΟΙΝΩΝΙΑ
ΠΟΛΙΤΩΝ-ΔΗΜΟΥ ΣΕ ΘΕΜΑΤΑ ΥΠΗΡΕΣΙΩΝ ΚΑΘΑΡΙΟΤΗΤΑΣ

ΓΕΩΡΓΙΟΥ ΑΝΔΡΕΑΣ ΑΜ: Π2016167

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:
ΣΤΕΦΑΝΙΔΑΚΗΣ ΜΙΧΑΗΛ, ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ

ΚΕΡΚΥΡΑ 2020

ΕΥΧΑΡΙΣΤΙΕΣ

Σε αυτό το σημείο της πτυχιακής εργασίας μου έχω την ανάγκη να ευχαριστήσω ιδιαίτερα τον επιβλέπων καθηγητή μου κ. Στεφανιδάκη Μιχαήλ για την εμπιστοσύνη και την υποστήριξη που μου έδειξε σε ένα καινοτόμο θέμα πτυχιακής εργασίας με αρκετές προκλήσεις. Οφείλω επίσης ένα μεγάλο ευχαριστώ στην οικογένεια μου για την υπομονή και την κατανόηση που μου έδειξαν κατά τη διάρκεια των σπουδών μου.

Π Ε Ρ Ι Λ Η Ψ Η

Σκοπός της παρούσας πτυχιακής εργασίας είναι ο σχεδιασμός και η υλοποίηση μιας διαδικτυακής εφαρμογής για την αμφίδρομη επικοινωνία μεταξύ πολιτών και δήμου σε θέματα υπηρεσιών καθαριότητας. Προκειται για μια εφαρμογή web που συνδυάζει προηγμένα χαρακτηριστικά όπως google maps. Υπάρχουν αρκετές παρόμοιες εφαρμογές στο διαδίκτυο που είναι πολύ διαδεδομένες στο ευρύ κοινό. Η συγκεκριμένη εφαρμογή διαφέρει από τις υπάρχοντες εφαρμογές γιατί συνδυάζει τρεις καινούργιες επιλογές. Οι επιλογές αυτές είναι: η αναφορά κάδου, η αναφορά των παραμελημένων σκουπιδιών και το αίτημα προσθήκης κάδου πάνω σε ειδικά διαμορφωμένο χάρτη google. Οι πολίτες θα συνεργάζονται με το αρμόδιο τμήμα καθαριότητας για να πετύχουν τα καλύτερα αποτελέσματα στην πόλη τους. Εφόσον επιτευχθεί σε μεγάλο βαθμό αυτό, το αποτέλεσμα θα είναι πολλές πόλεις καθαρότερες.

Λέξεις κλειδιά: διαδικτυακή εφαρμογή, εφαρμογή καθαριότητας, χάρτης google, αναφορά κάδου, αναφορά σκουπιδιών.

ABSTRACT

The purpose of this dissertation is to design and implement an online application for two-way communication between citizens and the municipality on cleaning services. It is a web application that combines advanced features such as google maps. There are several similar applications on the internet that are very popular with the general public. This application differs from existing applications because it combines three new options. The options: the bin report, the report of neglected garbage and the request to add a bin on a specially designed google map. Citizens will work with the cleaning department to achieve the best results in their city. If this is achieved to a large extent, the result will be many cities cleaner.

Keywords: web application, cleaning application, google maps, bucket report, garbage report.

Π Ε Ρ Ι Ε Χ Ο Μ Ε Ν Α

Ευχαριστίες.....	1
Περίληψη και λέξεις κλειδιά.....	2
Abstract.....	3
1. Εισαγωγή.....	5
1.1 Γενικά.....	5
1.2 Σκοπός και Στόχοι.....	6
1.3 Δομή.....	8
2. Λογισμικά και εργαλεία που χρησιμοποιήθηκαν.....	9
2.1 Linux.....	9
2.2 Django.....	9
2.3 Google Maps.....	10
2.4 Html, CSS, Javascript.....	11
3. Σχεδιασμός διαδικτυακής εφαρμογής.....	13
3.1 Πρότυπα Σχέδια.....	13
3.2 Σχέδιο Βάσης Δεδομένων.....	23
3.3 Σύνδεση Backend - Frontend.....	24
4. Υλοποίηση διαδικτυακής εφαρμογής.....	27
4.1 Υλοποίηση Server & API (Backend).....	27
4.2 Υλοποίηση web application interface (Frontend).....	40
5. Η διαδικτυακή εφαρμογή online.....	53
5.1 Heroku.....	53
5.2 Διαδικασία μεταφόρτωσης.....	54
6. Συμπεράσματα.....	58
6.1 Μελλοντική Επέκταση.....	59
Παράρτημα.....	60
Βιβλιογραφία - Δικτυογραφία.....	61

1.1 Γενικά

Σε μία εποχή όπου ο κάθε πολίτης χρησιμοποιεί τις διαδικτυακές εφαρμογές (web applications) για να καλύψει τις ανάγκες του, όποιες και αν είναι αυτές (ψυχαγωγία, ενημέρωση, εργασία, επικοινωνία), ο κλάδος της πληροφορικής καλείται να αναπτύσσεται συνεχώς και να δημιουργεί καινοτόμες εφαρμογές.

Τα βασικότερα πλεονεκτήματα των διαδικτυακών εφαρμογών είναι:

- **Η προσβασιμότητα από οποιαδήποτε συσκευή** δηλαδή οι χρήστες των διαδικτυακών εφαρμογών έχουν άμεση πρόσβαση στις εφαρμογές που θέλουν να χρησιμοποιήσουν από οποιονδήποτε υπολογιστή ή άλλη συσκευή, χωρίς την εγκατάσταση κάποιου επιπρόσθετου λογισμικού. [1]
- **Δεν καταναλώνουν πόρους και χώρο του χρήστη**, εφόσον οι διαδικτυακές εφαρμογές δεν εκτελούνται στον υπολογιστή του χρήστη και δεν καταναλώνουν πόρους από το σύστημά του. Για τον λόγο αυτό οι εφαρμογές διαδικτύου είναι ιδιαίτερα ελαφριές για την τοπική υπολογιστική μονάδα.[1]
- **Γρήγορη αναβάθμιση**, σε μια κλασική τοπική εφαρμογή η αναβάθμιση του συστήματος θα πρέπει να γίνει σε κάθε ένα υπολογιστή ξεχωριστά πράγμα που απαιτεί χρόνο και χρήμα. Αντίθετα σε μια διαδικτυακή εφαρμογή η αναβάθμιση πραγματοποιείται μόνο στον εξυπηρετητή που φιλοξενεί την εφαρμογή και ταυτόχρονα το αναβαθμισμένο πρόγραμμα είναι διαθέσιμο σε όλους τους χρήστες. Με τον τρόπο αυτόν εξοικονομείται χρόνος ο οποίος είναι ιδιαίτερα πολύτιμος κυρίως για τις μεγάλες επιχειρήσεις ενώ ως συνέπεια του παραπάνω σημαντικά μειωμένο είναι και το κόστος της αναβάθμισης μιας και απαιτείται λιγότερο εργατικό δυναμικό για την διεκπεραίωση της αναβάθμισης.[1]

Φυσικά δεν λείπουν και τα μειονεκτήματα των διαδικτυακών εφαρμογών τα οποία είναι:

- **Αδυναμία χρήσης χωρίς σύνδεση στο διαδίκτυο**, οι διαδικτυακές εφαρμογές δεν είναι δυνατόν να χρησιμοποιηθούν αν δεν υπάρχει σύνδεση με το διαδίκτυο.[1]
- **Μη πλήρης συμβατότητα των περιηγητών**, ένα ακόμα μειονέκτημα που αφορά την τελευταία έκδοση της HTML είναι η μη πλήρης συμβατότητα των περιηγητών με την έκδοση αυτή. Αν και τα πλεονεκτήματα και οι δυνατότητες της HTML5 είναι πολλά, αρκετοί από τους περιηγητές δεν είναι ακόμα πλήρως συμβατοί με αυτά. Έτσι δεν γίνεται πλήρης χρήση των δυνατοτήτων αυτών, πράγμα που περιορίζει τους προγραμματιστές που έχουν αναλάβει ένα έργο.[1]

Οι βασικότεροι παράγοντες για την αποδοτική λειτουργία μιας εφαρμογής, είναι η ταχεία ανταπόκριση στις απαιτήσεις του χρήστη, η φιλικότητα και ευχρηστία προς αυτόν. Η διαδικτυακή εφαρμογή που αναπτύχθηκε έχει στόχο να είναι όσο πιο φιλική και εύκολη για χρήση γίνεται, διότι θα χρησιμοποιείται από όλες τις ηλικίες.

1.2 Σκοπός και Στόχοι

Η πόλη της Κέρκυρας έχει ένα σοβαρό πρόβλημα με τα σκουπίδια. Πολλές φορές οι κάδοι είναι γεμάτοι για αρκετές μέρες και υπάρχουν σκουπίδια παραμελημένα. Με αφορμή αυτό αναπτύχθηκε μια εφαρμογή που καλείται να δίνει λύσεις σε αυτά τα προβλήματα.

Ο κύριος στόχος της εφαρμογής είναι να δίνει την δυνατότητα στον χρήστη να επιλέξει και να αναφέρει αν κάποιος κάδος είναι γεμάτος, σπασμένος ή αγνοείται, ούτως ώστε το αρμόδιο τμήμα να αντιδράσει άμεσα και στοχευμένα. Ο πολίτης θα έχει την δυνατότητα να κάνει αναφορές για σκουπίδια παραμελημένα τα οποία βρίσκονται σε περιοχές που δεν υπάρχουν κάδοι. Επίσης θα έχει την δυνατότητα να αιτηθεί προσθήκη νέου κάδου σε μια τοποθεσία που νιώθει ότι χρειάζεται.

Ο κάθε πολίτης υποχρεούται να δημιουργήσει προσωπικό λογαριασμό με την ηλεκτρονική του διεύθυνση εφόσον θέλει να κάνει αιτήσεις και αναφορές για κάδους και σκουπίδια. Με αυτόν τον τρόπο ελαχιστοποιούμε τις ψευδής αναφορές και απαιτούμε σοβαρότητα στην χρήση της εφαρμογής. Αυτή η εφαρμογή βοηθάει το αρμόδιο τμήμα που καθαρίζει την πόλη να εντοπίζει με ακρίβεια την τοποθεσία των γεμάτων κάδων και των ξεχασμένων σκουπιδιών. Επίσης με την χρήση της

εφαρμογής οι πολίτες θα διατηρούν την γειτονία και γενικότερα την πόλη τους καθαρή.

Η παρούσα εφαρμογή έχει δύο όψεις, μία για τους χρήστες (πολίτες) και η άλλη όψη για τους αρμόδιους του δήμου (διαχειριστές). Με τη συνεργασία των δύο πλευρών, ο αρχικός στόχος θα μπορεί να πραγματοποιηθεί σε γρήγορο χρόνο και στοχευμένα θα καθαρίζονται όλα τα σκουπίδια από την πόλη.

Υπάρχουν αρκετές παρόμοιες εφαρμογές με αυτήν που παρουσιάζεται στην παρούσα πτυχιακή εργασία, ιδιαίτερα στο εξωτερικό όπου είναι πολύ συνδεδεμένες με το ευρύ κοινό. Για παράδειγμα, η εφαρμογή **FixMyStreet** [2] όπου οι χρήστες ανεβάζουν φωτογραφίες και σχετικές πληροφορίες με τους χαλασμένους δρόμους ούτως ώστε το αρμόδιο τμήμα να αναλάβει την επισκευή τους.

Μια άλλη παρόμοια εφαρμογή είναι η **TrashOut** [3] όπου οι πολίτες έχουν την επιλογή να ανεβάσουν στην εφαρμογή εικόνα και σχετικές πληροφορίες από σκουπίδια, έτσι ώστε το αρμόδιο τμήμα να αναλάβει την εκκαθάριση τους.

Οι δύο παραπάνω διαδικτυακές εφαρμογές έχουν ως κεντρική ιδέα την αναφορά των προβλημάτων μέσω εικόνων και κάποιον μικρών περιγραφών. Στην δική μας περίπτωση στην διαδικτυακή εφαρμογή που αναπτύχθηκε υπάρχει αυτό το κομμάτι στην αναφορά σκουπιδιών όπου ο χρήστης ανεβάζει εικόνα στην αναφορά του με τα παραμελημένα σκουπίδια. Επιπλέον, υπάρχουν ακόμα δύο επιλογές που μπορεί να κάνει ο χρήστης, θα έχει την δυνατότητα να αναφέρει τους γεμάτους κάδους που αυτό συμβαίνει τακτικά και να αιτηθεί προσθήκη νέου κάδου σε μία τοποθεσία που πιστεύει ότι χρειάζεται. Μπορεί να παρακολουθεί ποιος κάδος είναι άδειος και ποιος άδειος ούτως ώστε να γνωρίζει που μπορεί να πετάξει τα σκουπίδια του.

1.3 Δομή

Η εν λόγω πτυχιακή εργασία είναι δομημένη σε έξι (6) κεφάλαια τα οποία έχουν ως στόχο τη ανάλυση των βασικών σκελών της, τόσο δηλαδή το σχεδιαστικό όσο και το κομμάτι της ανάπτυξης. Κατά την πραγματοποίηση της εργασίας θα υπάρξουν παραπομπές σε κείμενα και έρευνες τρίτων, όπου θα αριθμούνται (IEEE [1]) για να γίνεται πιο εύκολος ο εντοπισμός τους στο παράρτημα βιβλιογραφία και δικτυογραφία.

Συγκεκριμένα, το **Κεφάλαιο 1** αποτελεί μια πρώτη επαφή με την εφαρμογή καθώς στοχεύει στην εισαγωγή της ιδέας και στην επίδειξη υπαρχόντων παρόμοιων εφαρμογών. Αυτό το κεφάλαιο προσπαθεί να κατατοπίσει τον αναγνώστη και να του δώσει μια ιδέα για το τι θα ακολουθήσει.

Στο αμέσως επόμενο **Κεφάλαιο 2**, γίνεται αναφορά στα λογισμικά και στα εργαλεία που χρησιμοποιήθηκαν για τη σχεδίαση και ανάπτυξη της διαδικτυακής εφαρμογής. Όλη η υλοποίηση της εφαρμογής έγινε σε περιβάλλον Unix. Η εφαρμογή αναπτύχθηκε με τη βοήθεια του λογισμικού Django σε γλώσσα προγραμματισμού python. Πολύ σημαντικό εργαλείο που συμπεριλαμβάνεται στην εφαρμογή είναι η χρήση χάρτη από την Google (Google Maps).

Στο **Κεφάλαιο 3**, παρουσιάζονται τα αρχικά σχέδια, τα οποία σχεδιαστηκαν σε χαρτί για να δημιουργηθεί μια πρώτη ιδέα για το πώς θα δείχνει η εφαρμογή. Στη συνέχεια σχεδιάστηκε το σχήμα της βάσης δεδομένων που χρειαζόταν για την εφαρμογή. Το τελευταίο σχεδιάγραμμα είναι η σύνδεση backend (server side) με το frontend.

Στο **Κεφάλαιο 4** περιγράφεται η υλοποίηση της διαδικτυακής εφαρμογής. Αυτό το κεφάλαιο χωρίζεται σε δύο βασικά τμήματα: το πρώτο είναι το Backend όπου υλοποιούμε το Server και το API, δηλαδή τον σκελετό της εφαρμογής και στο δεύτερο μέρος υλοποιούμε το Frontend (web app interface), δηλαδή το πώς θα δείχνει η εφαρμογή μας.

Τέλος, η πτυχιακή εργασία ολοκληρώνεται με το **Κεφάλαιο 5**, συνοψίζοντας τα ωφέλιμα μέρη της πτυχιακής και παραθέτοντας προτάσεις για τη μελλοντική επέκταση της εφαρμογής.

ΛΟΓΙΣΜΙΚΑ ΚΑΙ ΕΡΓΑΛΕΙΑ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ

2.1 Linux

Για τις ανάγκες υλοποίησης της διαδικτυακής εφαρμογής μου ασχολήθηκα σε περιβάλλον Unix συγκεκριμένα σε Ubuntu. Το Ubuntu είναι ένα ανοικτού κώδικα, ελεύθερο και δωρεάν λειτουργικό σύστημα βασισμένο στον πυρήνα Linux. Το Ubuntu ξεκίνησε το 2004, βασισμένο στη διανομή Debian. Ο στόχος του Ubuntu είναι η παροχή ενός διαρκώς ενημερωμένου, σταθερού λειτουργικού συστήματος για τον μέσο χρήστη, με ενισχυμένη έμφαση στην ευκολία χρήσης και εγκατάστασης. [4]

Χρηματοδοτείται από την Canonical Ltd., μία ιδιωτική επιχείρηση που ιδρύθηκε από τον Νοτιοαφρικανό επιχειρηματία Μαρκ Σάτλγουορθ (Mark Shuttleworth). Αντί να πωλεί το Ubuntu καθεαυτό, η Canonical καταγράφει έσοδα από την επί πληρωμή τεχνική υποστήριξη που παρέχει για το προϊόν της. Διατηρώντας το Ubuntu ελεύθερο και ανοικτό η Canonical δέχεται και την βοήθεια τρίτων προγραμματιστών για την ανάπτυξή του. Υπάρχουν αρκετές επίσημες παραλλαγές του Ubuntu όπως το Xubuntu, Kubuntu, Lubuntu, Ubuntu Gnome και το Ubuntu Mate. Όλα είναι βασισμένα στον πυρήνα Ubuntu απλά αλλάζει η διεπαφή με τον χρήστη στο κάθε ένα. [4]

2.2 Django

Το Django είναι ένα πλαίσιο ιστού (Web Framework) που έχει σχεδιαστεί για να υποστηρίζει την ανάπτυξη διαδικτυακών εφαρμογών, με απλά λόγια σε αυτό το λογισμικό βασίζεται ο σκελετός κάθε εφαρμογής ιστού. Είναι ένα υψηλού επιπέδου Python Web πλαίσιο με προτερήματα του την ταχεία ανάπτυξη και τον καθαρό, ρεαλιστικό σχεδιασμό. Είναι χτισμένο από έμπειρους προγραμματιστές, με τη χρήση αυτού αποφεύγεται η μεγάλη ταλαιπωρία της ανάπτυξης Ιστού, με αποτέλεσμα να μπορεί ο προγραμματιστής να εστιάσει καλύτερα στη σύνταξη της εφαρμογής του χωρίς να χρειάζεται να ανακαλύψει εκ νέου τον τροχό. Είναι δωρεάν και ανοικτού κώδικα.

Πρωταρχικός στόχος του Django είναι να διευκολύνει τη δημιουργία σύνθετων - δυναμικών ιστοτόπων που βασίζονται σε βάση δεδομένων.

Το πλαίσιο αυτό δίνει έμφαση στην επαναχρησιμοποίηση και την «δυνατότητα μεταφοράς» των στοιχείων, τον λιγότερο κώδικα, την ταχεία ανάπτυξη και την αρχή της μη επαναλαμβανόμενης συμπεριφοράς .

Η γλώσσα προγραμματισμού Python χρησιμοποιείται παντού στο λογισμικό Django, ακόμη και στα αρχεία ρυθμίσεων, στα μοντέλα δεδομένων (πίνακες βάσης δεδομένων), και γενικά σε όλες τις βιβλιοθήκες.

Το λογισμικό Django παρέχει την ευκολία δημιουργίας, ανάγνωσης, ενημέρωσης και διαγραφής από την βάση δεδομένων σε οποιαδήποτε στιγμή. Στο λογισμικό αυτό υπάρχει συνδεδεμένη βάση δεδομένων από την πρώτη εγκατάστασή του η οποία είναι η SQLite. [5]



2.3 Google Maps

Η Google προσφέρει μερικές υπηρεσίες δωρεάν για αρκετό διάστημα, μια από αυτές είναι οι χάρτες google maps platform όπου μπορούν να αξιοποιηθούν στις διαδικτυακές εφαρμογές ή ιστότοπους. Στη δική μας περίπτωση ο χάρτης δεν είναι στατικός, είναι δυναμικός, δηλαδή αλληλεπιδρά με τον χρήστη. Με τη βοήθεια της γλώσσας προγραμματισμού Javascript προσθέσαμε στο χάρτη κάδους σκουπιδιών και κάδους ανακύκλωσης, όπου ο χρήστης μπορεί να κάνει αναφορά βάσει αυτών.

Οι χάρτες google προσφέρουν δορυφορικές εικόνες, αεροφωτογραφίες, χάρτες δρόμων, διαδραστική θέα στους δρόμους 360 °, συνθήκες κυκλοφορίας σε πραγματικό χρόνο και σχεδιάζουν διαδρομές για ταξίδια με τα πόδια, αυτοκίνητο, ποδήλατο ή δημόσια μέσα μεταφοράς. Το 2020 η google ανακοίνωσε ότι οι χάρτες google χρησιμοποιήθηκαν από περισσότερα 1 δισεκατομμύριο άτομα κάθε μήνα.

Με την ενσωμάτωση της γλώσσας προγραμματισμού Javascript μπορούν να γίνουν πάρα πολλές αλλαγές πάνω στον χάρτη, για παράδειγμα στην δική μας εφαρμογή προσθέσαμε εικονίδια με κάδους ακριβώς στην τοποθεσία που βρίσκονται για να μπορέσουμε να είμαστε όσο πιο ακριβής γίνεται. Επίσης όταν πατάμε στο

εικονίδιο του κάδου ανοίγει ένα παραθυράκι στο χάρτη με κάποιες επιλογές για τον χρήστη. [6]



2.3 Html, Css, Javascript

Η **HTML** (**H**yper**T**ext **M**arkup **L**anguage) είναι μια γλώσσα μορφοποίησης υπερκειμένου και η κύρια γλώσσα δόμησης ιστοσελίδων του WWW (World Wide Web), τα στοιχεία της είναι βασικά δομικά στοιχεία των ιστοσελίδων. Χρησιμοποιείται για να απεικονίζει ένα τμήμα κειμένου έτσι ώστε να εμφανίζεται καλύτερα. Επιτρέπει την ενσωμάτωση ήχου και εικόνων στις web σελίδες.

Αρχικά είχε κατασκευαστεί με σκοπό μόνο την μορφοποίηση κειμένου, στη συνέχεια εξελίχθηκε και ενσωμάτωσε και άλλες τεχνικές. Η γλώσσα HTML αποτελείται από ετικέτες (Tags), οι οποίες περικλείονται μέσα σε σύμβολα `< >` (για παράδειγμα `<html>`). Οι ετικέτες συνήθως λειτουργούν ανά ζεύγη (παράδειγμα αρχή `<h1>` και τέλος `</h1>`), η πρώτη ονομάζεται ετικέτα έναρξης και η δεύτερη ετικέτα λήξης.

Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ.

Ο σκοπός ενός web browser είναι να διαβάσει τα έγγραφα HTML και να τα συνθέτει σε σελίδες που μπορεί κανείς να διαβάσει ή να ακούσει. Ο browser δεν εμφανίζει τις ετικέτες HTML, αλλά τις χρησιμοποιεί για να παρουσιάσει το περιεχόμενο της σελίδας. Μπορούν επίσης να ενσωματώνονται σενάρια εντολών σε γλώσσες όπως η JavaScript, τα οποία επηρεάζουν τη συμπεριφορά των ιστοσελίδων HTML και από στατικές τις κάνουν διαδραστικές. [7]

Η **CSS** γλώσσα προγραμματισμού ανήκει στην κατηγορία των γλωσσών φύλλων ύφους που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης (πχ HTML).

Χρησιμοποιείται δηλαδή 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 δίνει περισσότερες δυνατότητες σε σχέση με την html. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται ως απαραίτητη. [7]

Η **Javascript** είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές. Υποστηρίζει εμφωλευμένες συναρτήσεις όπως επίσης και ανώνυμες συναρτήσεις. Η πιο συνηθισμένη χρήση της JavaScript είναι να προσθέτει συμπεριφορά στην HTML στην πλευρά του χρήστη. Τα script είναι ενσωματωμένα στις HTML σελίδες και αλληλεπιδρούν με το Μοντέλο Αντικειμένου Εγγράφου (DOM – Document Object Model). Με την προσθήκη της Javascript σε μια ιστοσελίδα ιστού ο προγραμματιστής επιτυγχάνει να δημιουργήσει μια δυναμική ιστοσελίδα όπου ο χρήστης αλληλεπιδρά με αυτήν. [7]



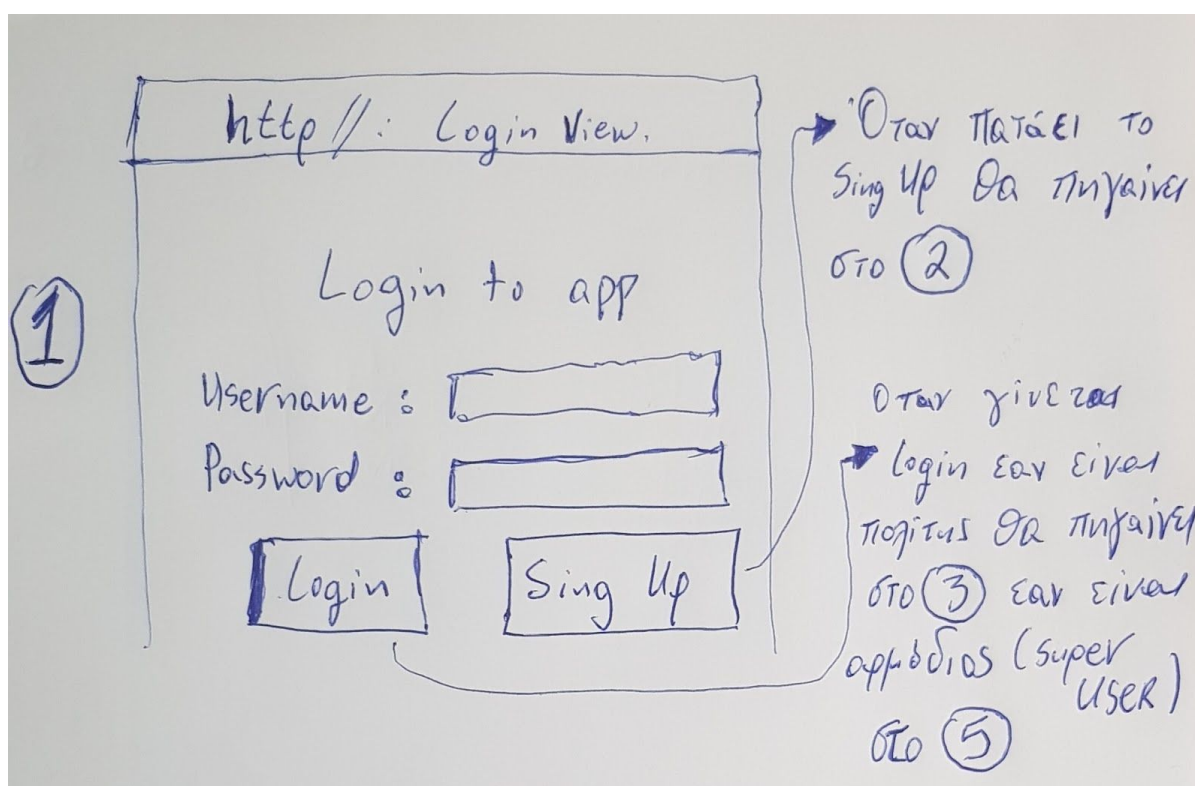
ΣΧΕΔΙΑΣΜΟΣ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ

Στην συγκεκριμένη ενότητα παρουσιάζονται τα προσχέδια της διαδικτυακής εφαρμογής, που δημιουργήθηκαν με σκοπό να μας βοηθήσουν να καταλάβουμε καλύτερα την κεντρική ιδέα και να αποκτήσουμε μια περαιτέρω γνώση που θα μας οδηγήσει στην υλοποίηση της.

3.1 Πρότυπα σχέδια

Τα σχέδια των προτύπων (templates) δείχνουν το πώς φανταζόμαστε την διαδικτυακή εφαρμογή μας προτού την υλοποιήσουμε. Με τη δημιουργία αυτών των προτύπων σχηματίζουμε μια καθαρή εικόνα για τα δεδομένα μας άρα θα μπορέσουμε να ανακαλύψουμε και το σχέδιο της βάσης δεδομένων μας για αργότερα.

3.1.1 Πρότυπο σχέδιο σύνδεσης χρήστη στην εφαρμογή



Εικόνα 1. Login page

Στην πρώτη επαφή του χρήστη με την εφαρμογή καθαριότητας θα βλέπει τη σελίδα σύνδεσης. Εάν ο χρήστης δεν έχει λογαριασμό θα πατάει στο SignUp όπου θα τον παραπέμπει στην **Εικόνα 2** για να δημιουργήσει ένα. Εάν ο απλός χρήστης (πολίτης) έχει λογαριασμό θα γίνεται σύνδεση και θα ανακατευθύνεται στη **Εικόνα 3**. Στην περίπτωση που συνδεθεί κάποιος από το αρμόδιο τμήμα καθαριότητας (Super User) η εφαρμογή τον ανακατευθύνει στη **Εικόνα 6**. Στην περίπτωση που κάποιος χρήστης δεν θέλει να δημιουργήσει λογαριασμό, θα έχει την δυνατότητα να διαβάζει τις καταστάσεις των κάδων πάνω στον χάρτη αλλά δεν θα μπορεί να κάνει αναφορές, δηλαδή θα μπορεί να παρακολουθεί στην γειτονιά του εάν κάποιος κάδος γέμισε ή έσπασε ή αγνοείται.

3.1.2 Πρότυπο σχέδιο δημιουργίας λογαριασμού χρήστη

http:// Sing Up View

Sing Up

② Username :

Password :

Password Again:

Email :

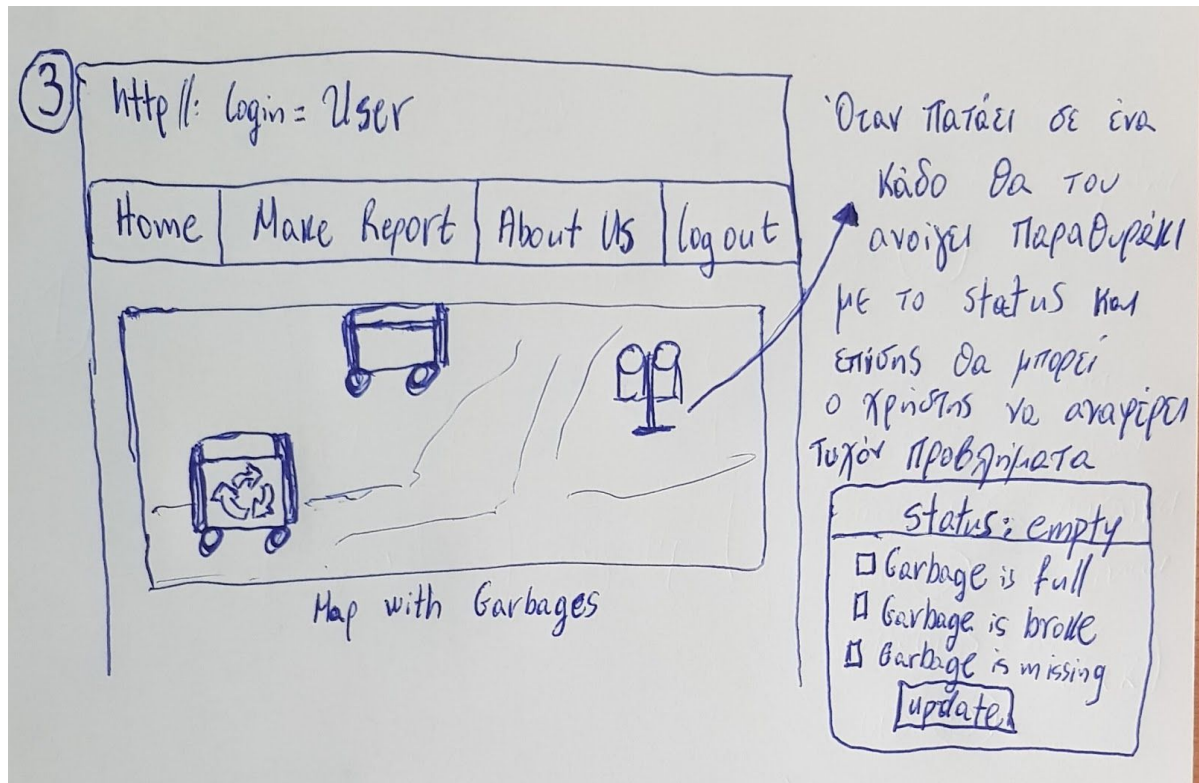
Μόλις δημιουργηθεί λογαριασμός θα επιστρέψει στο login view ①

Εικόνα 2. Sign Up page

Στη **Εικόνα 2** ο χρήστης θα δημιουργεί λογαριασμό. Αρχικά θα γεμίζει την φόρμα με ορθές πληροφορίες. Για τη δημιουργία λογαριασμού θα χρειαστεί να ορίσει ένα όνομα χρήστη που δεν χρησιμοποιείται από κάποιον άλλο χρήστη, ένα ισχυρό κωδικό ασφαλείας και την ηλεκτρονική διεύθυνση του. Εφόσον δημιουργηθεί ο λογαριασμός, αυτόματα θα στέλνεται ένα μήνυμα επαλήθευσης στον ηλεκτρονικό λογαριασμό του χρήστη, όπου για να γίνει ενεργός χρήστης θα πρέπει να επαληθεύσει

τον λογαριασμό του. Εάν δεν επαληθεύσει τον λογαριασμό του ο χρήστης θα παραμείνει ανενεργός και δεν θα μπορεί να κάνει αναφορές παρά μόνο να βλέπει τις καταστάσεις των κάδων.

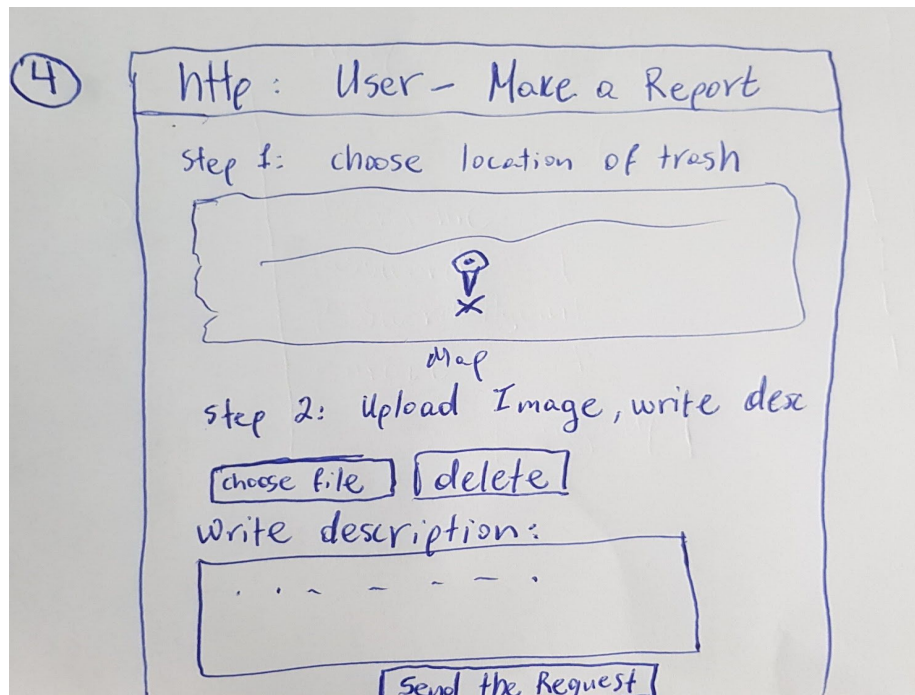
3.1.3 Πρότυπο σχέδιο αρχικής σελίδας του χρήστη μετά την σύνδεση



Εικόνα 3. User homepage

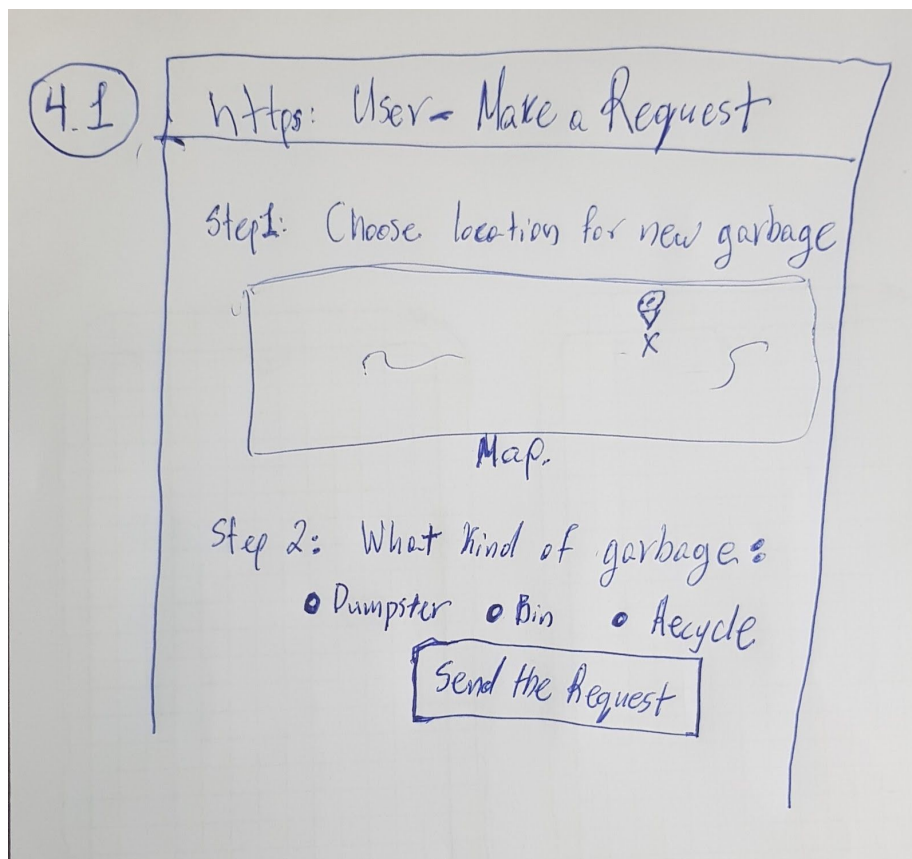
Στην αρχική σελίδα του χρήστη θα υπάρχει μια μπάρα με διάφορες επιλογές, όπως **Make a Report** δηλαδή να αναφέρει κάποια παραμελημένα σκουπίδια που παρατήρησε, **About us** να διαβάσει σχετικές πληροφορίες για την εφαρμογή ή να αποσυνδεθεί, **Make a Request** δηλαδή να αιτείται μια προσθήκη νέου κάδου. Κάτω από την μπάρα θα είναι ο χάρτης με τους κάδους όπου όταν πατήσει σε κάποιον κάδο ο χρήστης θα του εμφανίζεται η κατάσταση του κάδου και οι τρεις επιλογές που μπορεί ο χρήστης να αναφέρει: **1)** ο κάδος είναι γεμάτος, **2)** ο κάδος είναι σπασμένος, **3)** ο κάδος αγνοείται. Εφόσον ο χρήστης αναφέρει κάτι από τις τρεις επιλογές ο κάδος αυτόματα από πράσινο χρώμα θα αλλάζει και θα γίνεται κόκκινο: αυτό θα σημαίνει ότι κάτι δεν πάει καλά.

3.1.4 Πρότυπα σχέδια - δημιουργίας αναφοράς από τον χρήστη για παραμελημένα σκουπίδια και δημιουργίας αιτήματος για προσθήκη νέου κάδου από τον χρήστη.



Εικόνα 4. User Make a Report

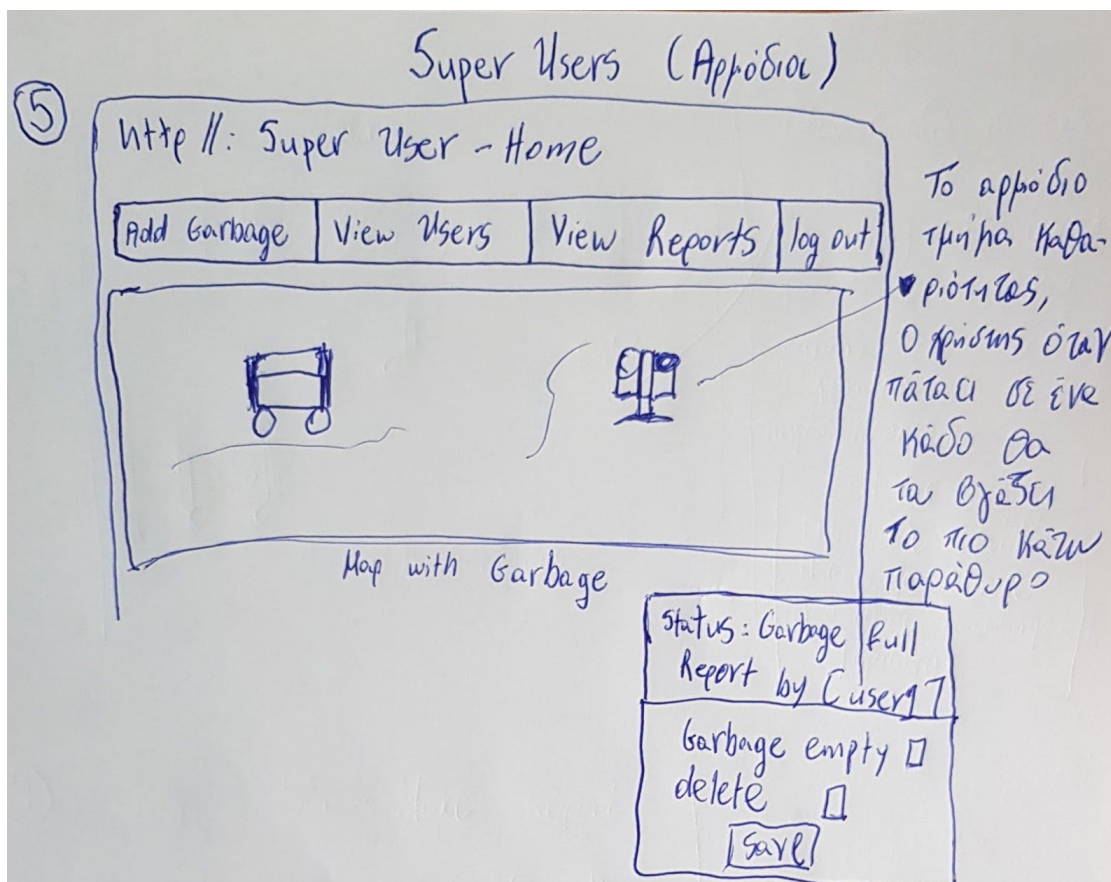
Ο χρήστης αν επιλέξει να κάνει μια αναφορά για σκουπίδια τα οποία είναι παραμελημένα και μακριά από κάδους θα κατευθύνεται στην παραπάνω σελίδα. Αρχικά θα πρέπει να επιλέξει την τοποθεσία που είδε τα σκουπίδια, στην συνέχεια θα ανεβάζει μια σχετική φωτογραφία από αυτά και τέλος θα πληκτρολογεί μια σχετική περιγραφή για να βοηθήσει την υπηρεσία να καταλάβει καλύτερα την κατάσταση. Το αίτημα αυτό θα στέλνεται στους αρμόδιους για εξέταση.



Εικόνα 5. User Make a Request.

Σε περίπτωση όπου ο χρήστης επιθυμεί να αιτηθεί προσθήκη νέου κάδου σε μια τοποθεσία θα μεταβαίνει στην πιο πάνω φόρμα για να υλοποιήσει την αίτηση του. Εκεί θα του ζητηθεί να κουνήσει το σημάδι στην τοποθεσία που πιστεύει ότι χρειάζεται κάδος. Στο επόμενο βήμα θα επιλέξει τη είδος κάδου πιστεύει ότι χρειάζεται εκεί και τέλος θα στέλνει το αιτημα του για να το επεξεργαστεί η αρμόδια ομάδα.

3.1.5 Πρότυπο σχέδιο αρχικής σελίδας του αρμόδιου χρήστη μετά την σύνδεση



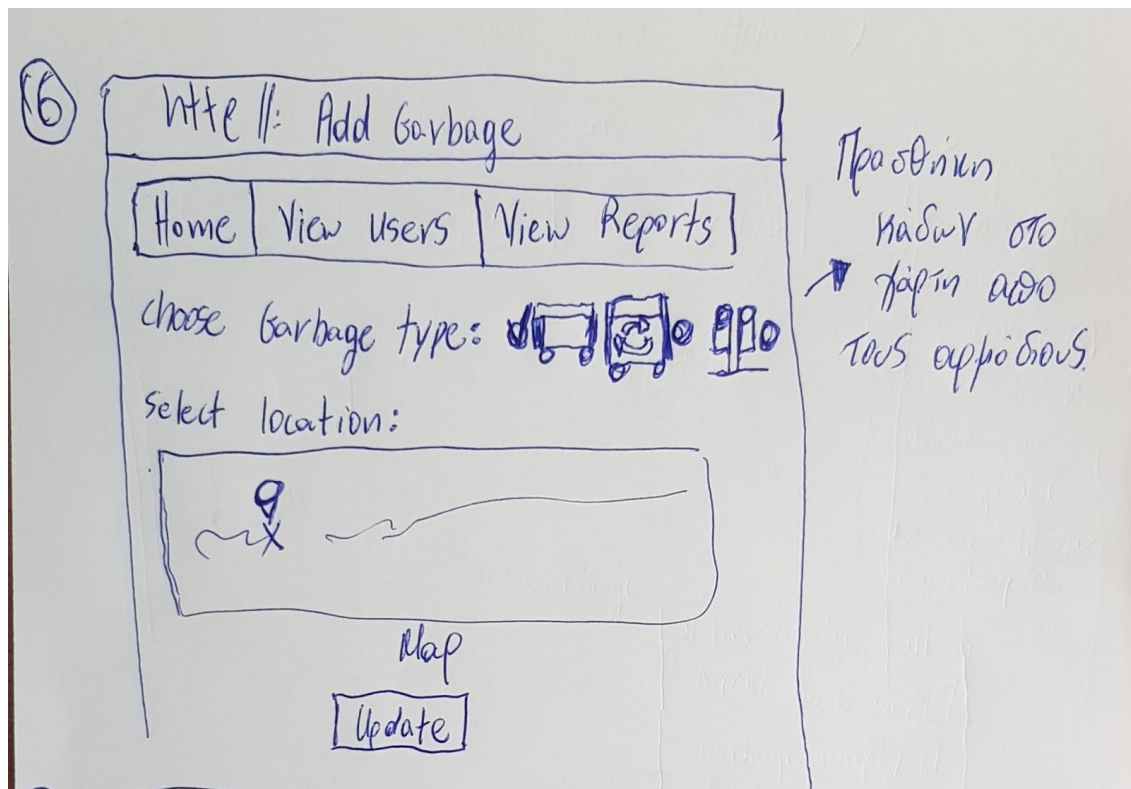
Εικόνα 6. SuperUser homepage

Στην αρχική σελίδα του SuperUser (η αρμόδια υπηρεσία) θα υπάρχει μια μπάρα επιλογών όπως προσθήκη κάδων στον χάρτη, δείτε τους χρήστες, δείτε τις αναφορές των χρηστών, δείτε τα αιτήματα των χρηστών.

Πιο κάτω από την μπάρα θα μπορούν να βρουν τους κάδους στον χάρτη, όπου αν πατήσουν σε αυτούς θα ανοίγει ένα παραθυράκι με την κατάσταση του κάθε ενός. Αν για παράδειγμα είναι γεμάτος ο κάδος θα είναι με κόκκινο χρώμα πάνω στον χάρτη και όταν πατήσουμε πάνω σε αυτόν θα περιγράφει την κατάσταση του, όπως επίσης από ποιόν χρήστη ενημερώθηκε.

Ο Super User θα έχει την επιλογή να αλλάξει την αναφορά δηλαδή να αδειάσει τον κάδο ή να τον διαγράψει τελείως από τον χάρτη.

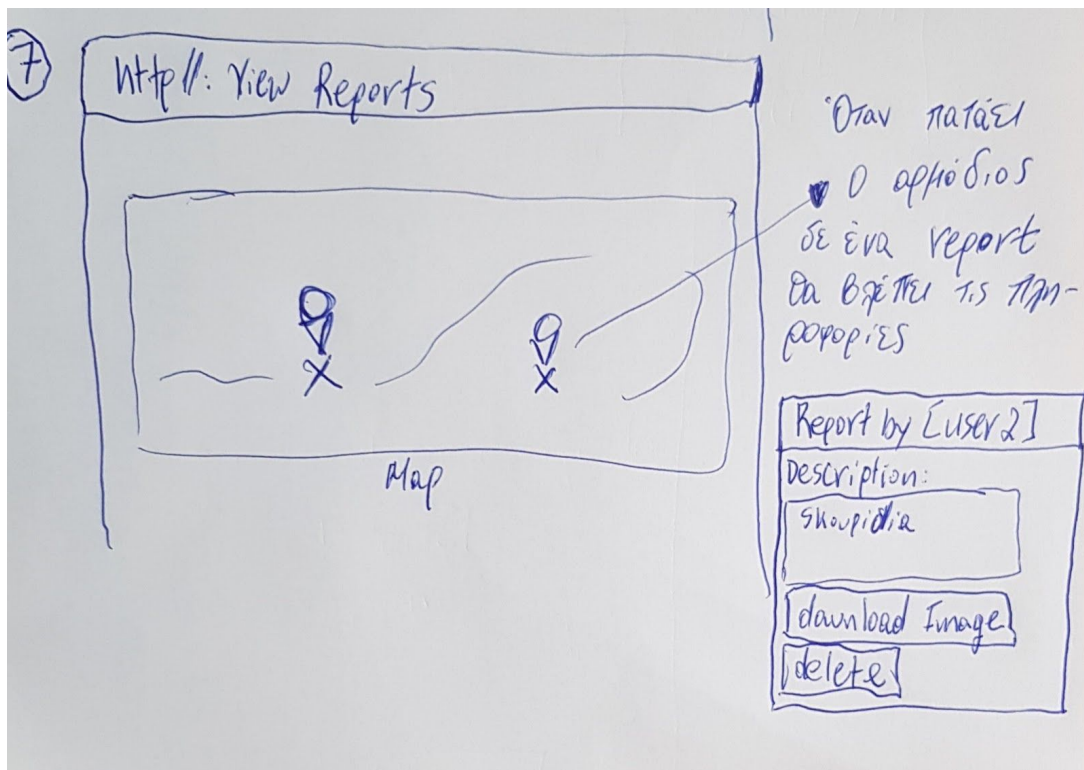
3.1.6 Πρότυπο σχέδιο προσθήκης κάδων από τους αρμόδιους της υπηρεσίας



Εικόνα 7. SuperUser Add Garbages

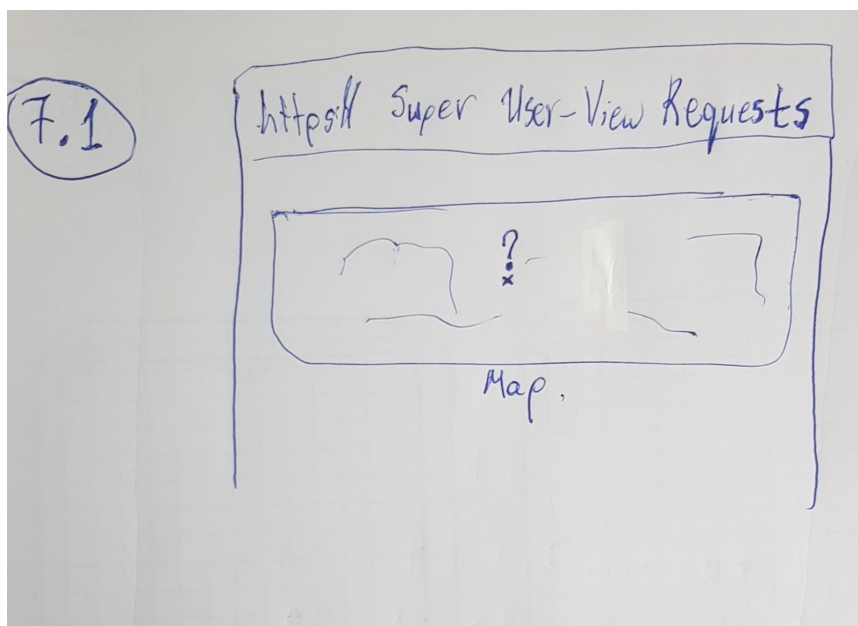
Στη σελίδα αυτή θα έχουν πρόσβαση μόνο οι αρμόδιοι της υπηρεσίας καθαρισμού όπου θα έχουν την δυνατότητα να προσθέτουν καινούργιους κάδους πάνω στον χάρτη. Αρχικά θα επιλέγουν το είδος του κάδου δηλαδή κάδος ανακύκλωσης, κανονικός κάδος ή καλάθι πόλης. Στην συνέχεια θα επιλέγουν την τοποθεσία που θέλουν να προσθέσουν τον κάδο. Τέλος θα αποθηκεύουν τους καινούργιους κάδους.

3.1.7 Πρότυπα σχέδια ανάγνωσης αναφορών και ανάγνωσης αιτημάτων από τους αρμόδιους



Εικόνα 8. SuperUser view reports

Στην σελίδα **view reports** οι αρμόδιοι θα παρακολουθούν τις αναφορές των χρηστών πάνω στον χάρτη. Όταν πατήσουν πάνω σε μία αναφορά θα ανοίγει ένα μικρο παραθυράκι με πληροφορίες της αναφοράς για παράδειγμα ποιος χρήστης έκανε την αναφορά, την περιγραφή της αναφοράς και μια σχετική φωτογραφία που θα μπορεί να κατεβάσει ο αρμόδιος. Εάν το πρόβλημα επιλυθεί ο superuser θα έχει την δυνατότητα να διαγράψει την αναφορά.



Εικόνα 9. Super User view request

Στην παραπάνω σελίδα οι αρμόδιοι (super users) θα έχουν την δυνατότητα να παρακολουθούν τα αιτήματα για προσθήκη κάδων των χρηστών. Οι αρμόδια υπηρεσία αφού εξετάσει το αίτημα και συμφωνήσει ότι υπάρχει η ανάγκη για προσθήκη νέου κάδου, θα τοποθετεί ένα νέο κάδο στην τοποθεσία εκεί, ψηφιακά και αναλογικά.

3.1.8 Πρότυπο σχέδιο διαχείρισης χρηστών από τους αρμόδιους

8

http:// View Users

User Name	Active	Email	Block.
User 1	☒	@hotmail	☐
User 2.	☐	@gmail	☐

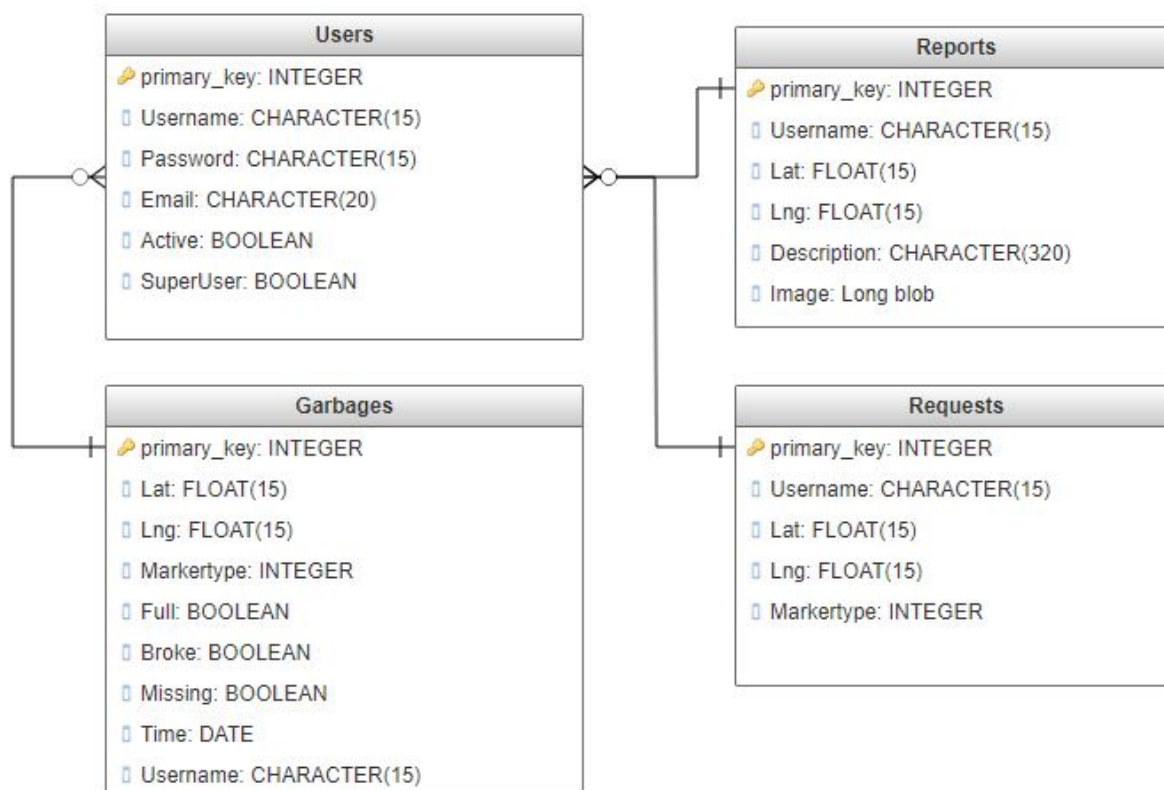
Εικόνα 10. SuperUser view the users

Οι superuser θα έχουν τη δυνατότητα να παρακολουθούν τους χρήστες της εφαρμογής και επίσης από ενεργούς χρήστες να τους αλλάζουν σε ανενεργούς εφόσον το κρίνουν απαραίτητο.

3.2 Σχέδιο Βάσης Δεδομένων

Κάθε δυναμική εφαρμογή για να ανταπεξέλθει στις ανάγκες της χρησιμοποιεί μία ή περισσότερες βάσεις δεδομένων. Η βάση δεδομένων κάθε εφαρμογής πρέπει να είναι σωστά διαμορφωμένη ούτως ώστε σε κάθε απαίτηση της εφαρμογής να υπάρχει πρόσβαση στα δεδομένα που χρειάζονται. Ο σκοπός μιας βάσης δεδομένων είναι η οργανωμένη αποθήκευση πληροφορίας και η δυνατότητα εξαγωγής της πληροφορίας αυτής, σύμφωνα με ερωτήματα που τίθενται στη σχεσιακή βάση δεδομένων.

Στην δική μας περίπτωση χρησιμοποιούμε σχεσιακή βάση δεδομένων (SQLite) δηλαδή μία συλλογή δεδομένων οργανωμένη σε συσχετιζόμενους πίνακες που παρέχει ταυτόχρονα ένα μηχανισμό για ανάγνωση, εγγραφή, τροποποίηση ή και πιο πολύπλοκες διαδικασίες πάνω στα δεδομένα.



Εικόνα 11. Σχήμα Βάσης Δεδομένων

Στη συγκεκριμένη διαδικτυακή εφαρμογή χρησιμοποιήσαμε μια βάση δεδομένων με τρεις πίνακες. Ο πίνακας Users προσδιορίζει τους χρήστες της εφαρμογής όπου θα είναι οι απλοί χρήστες οι πολίτες και οι super users οι αρμόδιοι.

Σε αυτό το πίνακα θα αποθηκεύονται το όνομα χρήστη, ο κωδικός ασφαλείας, η ηλεκτρονική διεύθυνση του χρήστη, ενεργός χρήστης και υπερ χρήστης.

Στον πίνακα με τις αναφορές (Reports) θα αποθηκεύονται όλα τα δεδομένα που θα προσδιορίζουν τα παραμελημένα σκουπίδια στον χάρτη δηλαδή το όνομα του χρήστη που έκανε την αναφορά, τις συντεταγμένες των σκουπιδιών στον χάρτη, η περιγραφή του χρήστη που έκανε την αναφορά και μια εικόνα που θα ανεβάζει αυτός που κάνει την αναφορά.

Ο πίνακας Garbages της βάσης δεδομένων της εφαρμογής θα κρατάει δεδομένα για τους κάδους που θα εμφανίζονται στον χάρτη. Συγκεκριμένα το μοναδικό αριθμό κλειδί κάθε κάδου, τις συντεταγμένες του κάθε κάδου, τι τύπος κάδου είναι, εάν είναι γεμάτος ή σπασμένος ή αγνοείται, την ώρα που έγινε η ενημέρωση και το όνομα του χρήστη που έκανε την ενημέρωση.

Τέλος, ο πίνακας Requests θα δέχεται τα δεδομένα με τα αιτήματα των χρηστών για προσθήκη νέου κάδου σε μία τοποθεσία. Αυτά τα δεδομένα θα είναι το όνομα του χρήστη που έκανε την αίτηση, οι συντεταγμένες που θα ήθελα να προστεθεί ο κάδος και ο τύπος του κάδου.

3.3 Σύνδεση Backend με Frontend

3.3.1 Το μπροστινό (frontend) και πίσω μέρος (backend)

Στην αρχιτεκτονική μιας διαδικτυακής εφαρμογής, ενδέχεται να υπάρχουν πολλά επίπεδα μεταξύ του υλικού και του τελικού χρήστη. Το μπροστινό μέρος (frontend) είναι μια αφαίρεση, απλοποιώντας το υποκείμενο στοιχείο παρέχοντας μια φιλική προς τον χρήστη διεπαφή, ενώ το πίσω μέρος χειρίζεται συνήθως την αποθήκευση δεδομένων και την επιχειρησιακή λογική.

Στις τηλεπικοινωνίες, το μπροστινό μέρος μπορεί να θεωρηθεί συσκευή ή υπηρεσία, ενώ το πίσω μέρος είναι η υποδομή που υποστηρίζει την παροχή υπηρεσιών.

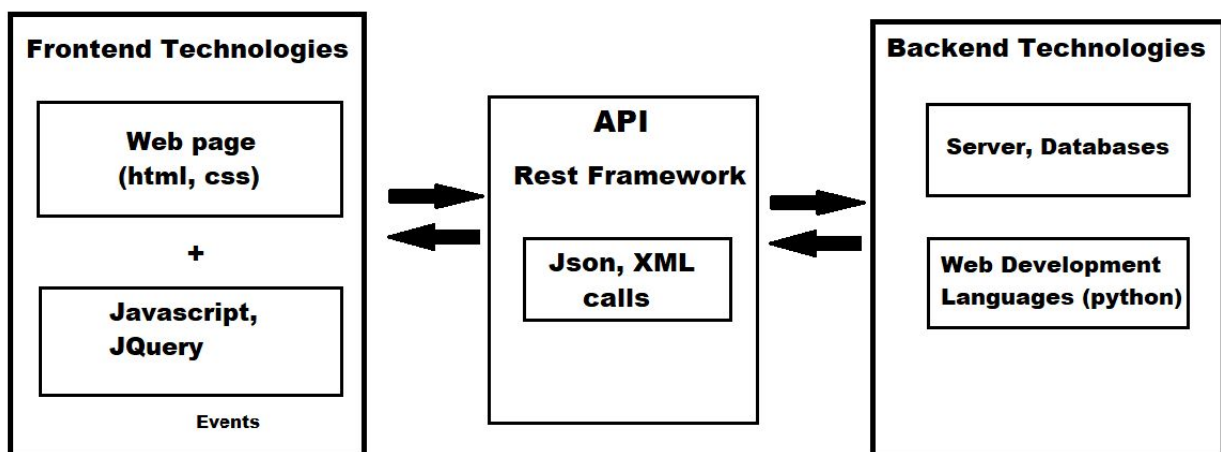
Ένας βασικός κανόνας είναι ότι η πλευρά πελάτη (ή "front end") είναι οποιοδήποτε στοιχείο που χειρίζεται ο χρήστης. Ο κωδικός side-server (ή "back end") βρίσκεται συνήθως στο διακομιστή.

Με τον όρο **Backend** δηλώνουμε όλες τις διεργασίες που εκτελούνται στο «παρασκήνιο». Ως συνήθως αφορούν τον διακομιστή που εκτελεί υπηρεσίες βάσης δεδομένων και όχι μόνο. Αυτές οι πράξεις δεν είναι ορατές ούτε προσβάσιμες προς τους χρήστες. Για παράδειγμα η διαδικασία συμπλήρωσης της ηλεκτρονικής φόρμας, ο χρήστης βλέπει μεν την φόρμα αλλά από τη στιγμή που θα κάνει το πάτημα του κουμπιού δεν βλέπει τα δεδομένα που και πώς αποθηκεύονται. Ή ακόμα ένα άλλο

παράδειγμα: το πάτημα σε ένα σύνδεσμο που σε μεταφέρει σε άλλη σελίδα του website. Για να γίνεται αυτή η διαδικασία ορθά υπάρχει κάποιος κώδικας στην πλευρά του server.

Το **Frontend** κομμάτι αποτελείται από εργαλεία και κώδικες που απεικονίζουν κείμενα, εικόνες, βίντεο, ήχους έτσι ώστε να εμφανίζονται καλύτερα στον χρήστη. Είναι η πρακτική της μετατροπής των δεδομένων σε γραφική διεπαφή, μέσω της χρήσης HTML, CSS και JavaScript, έτσι ώστε οι χρήστες να μπορούν να προβάλλουν και να αλληλεπιδρούν με αυτά τα δεδομένα. Υπάρχουν πολλά εργαλεία και πλατφόρμες, όπως το WordPress, το Joomla και το Drupal, διαθέσιμα που μπορούν να χρησιμοποιηθούν για την ανάπτυξη της διεπαφής ενός ιστότοπου και η κατανόηση των εργαλείων που ταιριάζουν καλύτερα σε συγκεκριμένες εργασίες σηματοδοτεί τη διαφορά μεταξύ της ανάπτυξης ενός ιστότοπου που έχει παραβιαστεί και ενός καλά σχεδιασμένου, επεκτάσιμου ιστότοπου.

3.3.2 Σύνδεση μπροστινού (frontend) και πίσω μέρους (backend)



Εικόνα 12. Σύνδεση backend με frontend

Όπως αναφέραμε πιο πριν, το μπροστινό μέρος είναι αυτό που βλέπει και αλληλεπιδρά με τον χρήστη, ενώ το πίσω μέρος εκτελεί κώδικα για τη σωστή λειτουργία της εφαρμογής και αποθηκεύει τα δεδομένα. Αυτές οι δύο τεχνολογίες μπροστινό μέρος και πίσω μέρος (frontend and backend) για να είναι αποτελεσματικές πρέπει να επικοινωνούν μεταξύ τους. Σε αυτό το σημείο επεμβαίνει το API (Application Programming Interface). Το API είναι ένας διαμεσολαβητής

λογισμικού που επιτρέπει σε δύο εφαρμογές να μιλούν μεταξύ τους. Με άλλα λόγια, ένα API είναι το messenger που παραδίδει το αίτημά του χρήστη στον πάροχο και, στη συνέχεια, δίνει την απάντηση. Όταν οι προγραμματιστές γράφουν κώδικα, δεν ξεκινούν συχνά από το μηδέν. Τα API επιτρέπουν στους προγραμματιστές να κάνουν επαναλαμβανόμενες αλλά πολύπλοκες διαδικασίες επαναχρησιμοποιήσιμες με λίγο κώδικα. Η ταχύτητα με την οποία τα API επιτρέπουν στους προγραμματιστές να δημιουργούν εφαρμογές είναι καθοριστικής σημασίας για τον τρέχοντα ρυθμό ανάπτυξης εφαρμογών. Για παράδειγμα φανταστείτε έναν σερβιτόρο (API) σε ένα εστιατόριο. Εσείς, ο πελάτης (frontend), κάθεστε στο τραπέζι με ένα μενού επιλογών για παραγγελία και η κουζίνα (backend) είναι ο πάροχος που θα εκπληρώσει την παραγγελία σας. Χρειάζεστε έναν σύνδεσμο για να κοινοποιήσετε την παραγγελία σας στην κουζίνα και στη συνέχεια να παραλάβετε το φαγητό σας πίσω στο τραπέζι σας. Δεν μπορεί να είναι ο σεφ γιατί μαγειρεύει στην κουζίνα. Χρειάζεστε κάτι για να συνδέσετε τον πελάτη (frontend) που παραγγέλλει φαγητό και τον σεφ (backend) που το ετοιμάζει. Εκεί ο σερβιτόρος - το API - μπαίνει στο παιχνίδι.

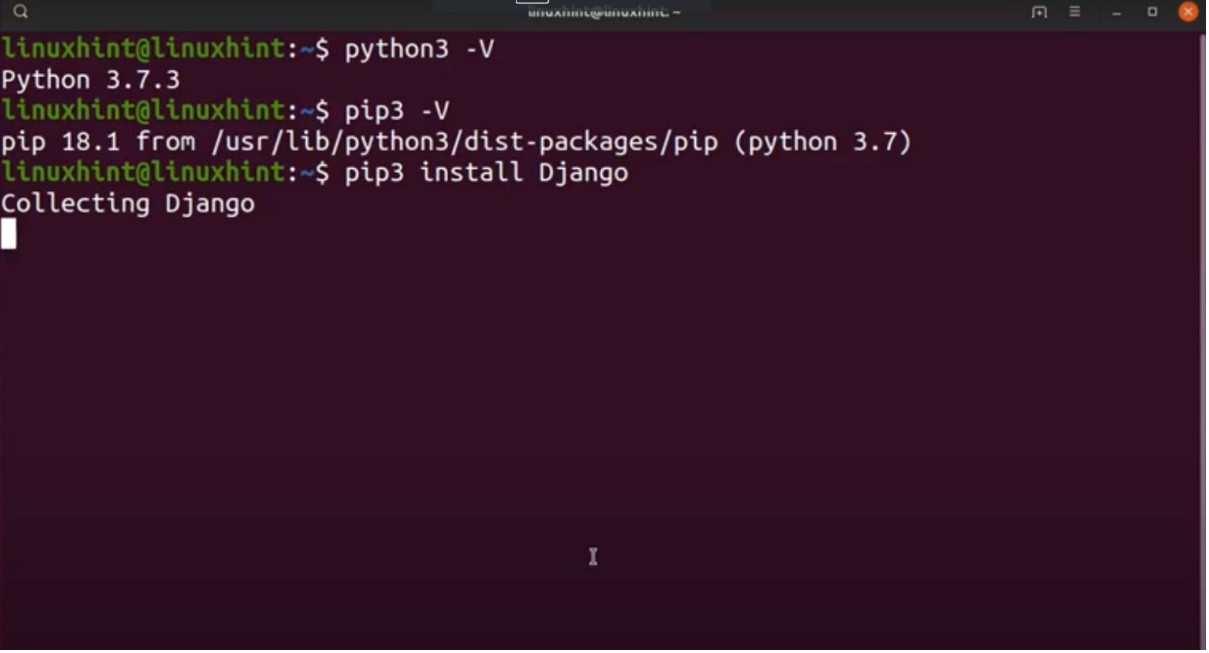
Ο σερβιτόρος παίρνει την παραγγελία σας, την παραδίδει στην κουζίνα, λέγοντας στην κουζίνα τι να κάνει. Στη συνέχεια, σας δίνει την απάντηση, σε αυτήν την περίπτωση, το φαγητό.

ΥΛΟΠΟΙΗΣΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ

4.1 Υλοποίηση Server & API (Backend)

Για την ολική υλοποίηση της διαδικτυακής εφαρμογής χρειάζεται να εγκαταστήσουμε το λογισμικό Django, το οποίο όπως προαναφέρθηκε είναι ένα πλαίσιο ιστού σε γλώσσα προγραμματισμού python όπου εκεί θα χτίσουμε βήμα βήμα την εφαρμογή μας. Με τη βοήθεια των πολλών βιβλιοθηκών του λογισμικού εξασφαλίζουμε ασφάλεια και γλιτώνουμε χρόνο.

4.1.1 Εγκατάσταση Django framework



```
linuxhint@linuxhint:~$ python3 -V
Python 3.7.3
linuxhint@linuxhint:~$ pip3 -V
pip 18.1 from /usr/lib/python3/dist-packages/pip (python 3.7)
linuxhint@linuxhint:~$ pip3 install Django
Collecting Django
█
```

Εικόνα 13. Εγκατάσταση Django [5]

Πρώτα απ' όλα ελέγχουμε εάν η έκδοση της python και του pip είναι αναβαθμισμένη στον υπολογιστή μας, έτσι ώστε να μπορέσουμε να εγκαταστήσουμε σωστά την Django. Εφόσον η εγκατάσταση τελειώσει εκτελούμε την εντολή **`sudo django-admin startproject (project Name)`** για να ξεκινήσουμε την κωδικοποίηση της εφαρμογής μας.

4.1.2 Υλοποίηση Βάσης Δεδομένων (Models)

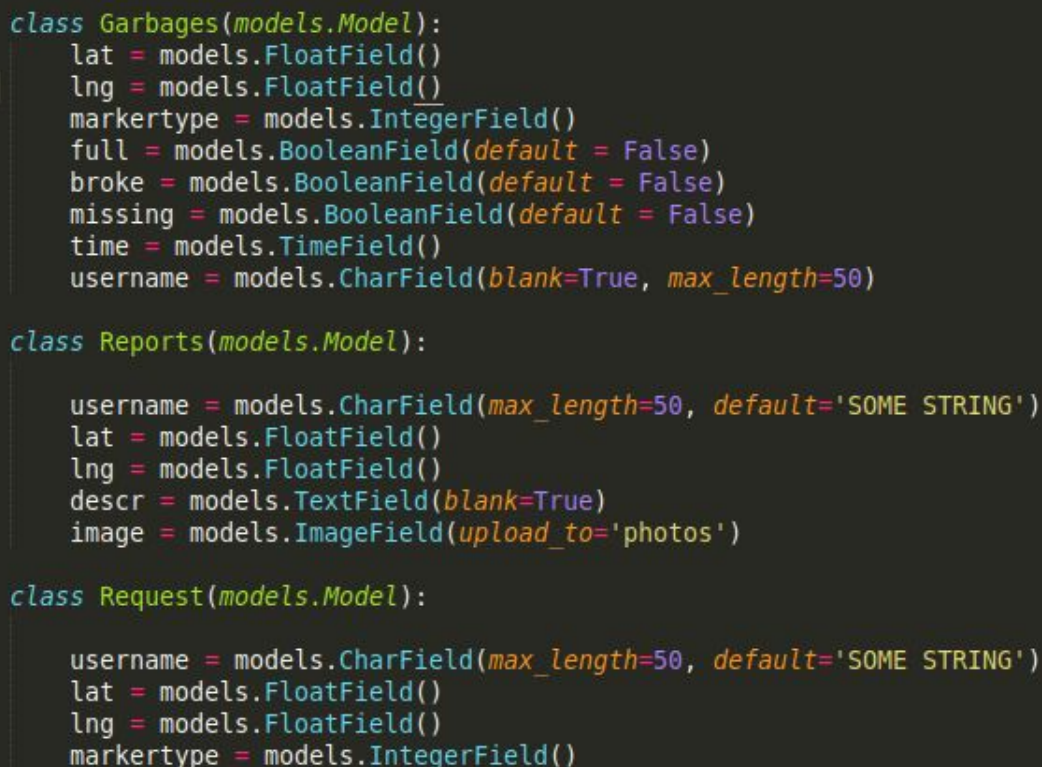
Στην django οι πίνακες της βάσης δεδομένων ορίζονται ως μοντέλα. Για παράδειγμα εάν στην SQL γλώσσα θα γράφαμε:

```
CREATE TABLE myapp_person (  
    "id" serial NOT NULL PRIMARY KEY,  
    "first_name" varchar(30) NOT NULL,  
    "last_name" varchar(30) NOT NULL  
);
```

στην django το γράφουμε σε γλώσσα python έτσι:

```
from django.db import models  
  
class Person(models.Model):  
    first_name = models.CharField(max_length=30)  
    last_name = models.CharField(max_length=30)
```

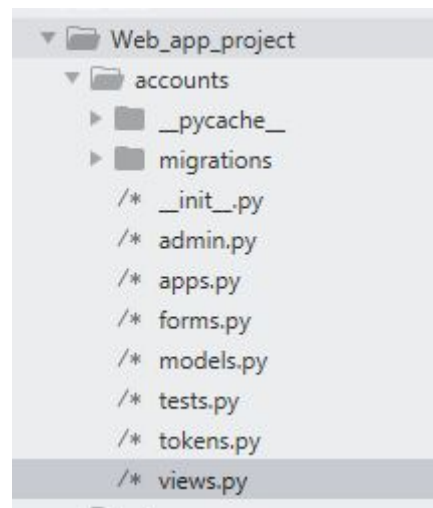
Στην δική μας περίπτωση τα μοντέλα μας θα είναι τέσσερα όσοι και οι πίνακες που σχεδιάσαμε πάρα πάνω στην **Εικόνα 11**. Σχήμα Βάσης Δεδομένων.



```
class Garbages(models.Model):  
    lat = models.FloatField()  
    lng = models.FloatField()  
    markertype = models.IntegerField()  
    full = models.BooleanField(default = False)  
    broke = models.BooleanField(default = False)  
    missing = models.BooleanField(default = False)  
    time = models.TimeField()  
    username = models.CharField(blank=True, max_length=50)  
  
class Reports(models.Model):  
    username = models.CharField(max_length=50, default='SOME STRING')  
    lat = models.FloatField()  
    lng = models.FloatField()  
    descr = models.TextField(blank=True)  
    image = models.ImageField(upload_to='photos')  
  
class Request(models.Model):  
    username = models.CharField(max_length=50, default='SOME STRING')  
    lat = models.FloatField()  
    lng = models.FloatField()  
    markertype = models.IntegerField()
```

Εικόνα 14. Τα τρία απο τα τέσσερα μοντέλα της διαδικτυακής εφαρμογής.

Το τέταρτο μοντέλο που αφορά τους λογαριασμούς χρηστών υλοποιήθηκε από έτοιμη βιβλιοθήκη του Django (Django authentication system).



Εικόνα 15. Λογαριασμοί χρηστών

Δημιουργείται ένα αυτοματοποιημένο μοντέλο με όνομα `accounts` όπου είναι ο πυρήνας του συστήματος ελέγχου ταυτότητας. Αυτοί οι λογαριασμοί είναι άτομα που αλληλεπιδρούν με τη διαδικτυακή εφαρμογή. Με τη χρήση αυτού του μοντέλου ελέγχουμε την εξουσιοδότηση των χρηστών που χρησιμοποιούν την εφαρμογή.

Υπάρχει μόνο μία κατηγορία χρήστη που ελέγχει όλες τις άλλες η κατηγορία `admin`.

Το αυτοματοποιημένο μοντέλο χρηστών της Django έχει πέντε κύρια χαρακτηριστικά όπου στην δική μας περίπτωση αυτά χρειαζόμαστε. (`Username`, `Password`, `Email`, `SuperUser`, `Active`)

4.1.3 Υλοποίηση διευθύνσεων (`urls`) και προβολών (`views`)

Ένα καθαρό και όμορφο σχήμα URL είναι μια σημαντική λεπτομέρεια σε μια εφαρμογή ιστού υψηλής ποιότητας. Το Django μας επιτρέπει να σχεδιάζουμε διευθύνσεις URL όπως θέλουμε, χωρίς περιορισμούς στο πλαίσιο. Για να σχεδιάσουμε τις διευθύνσεις URL για μια εφαρμογή δημιουργούμε ένα αρχείο **`urls.py`** στο οποίο είναι όλες οι διευθύνσεις. Αυτή η ενότητα είναι καθαρός κώδικας Python και παρέχει αντιστοίχιση μεταξύ των εκφράσεων διαδρομής URL στις προβολές μας (`views`).

```
urlpatterns = [
    path('', homepage),
    path('admin/', admin.site.urls),
    path('home/', home),
    path('request/', UserRequest_view),
    path('request/add_report/', add_report, name="add_report"),
    path('accounts/login/', login_view),
    path('accounts/register/', register_view),
    path('activate/<uidb64>/<token>/', activate, name='activate'),
    path('accounts/logout/', logout_view),
    path('full/<str:pk>/', update_full, name="update_full"),
    path('broke/<str:pk>/', update_broke, name="update_broke"),
    path('missing/<str:pk>/', update_missing, name="update_missing"),
    path('empty/<str:pk>/', empty, name="empty"),
    path('fixed/<str:pk>/', fixed, name="fixed"),
    path('found/<str:pk>/', found, name="found"),
    path('delete/<str:pk>/', delete, name="delete"),
    path('deleteReport/<str:pk>/', deleteReport, name="deleteReport"),
    path('add/', add_view),
    path('add/garbage/', add_garbage, name="add_garbage"),
    path('viewReports/', reports_view),
    path('api/', include('clean.urls')),
    path('success/', success_view),
    path('requestadd/', requestadd_view),
    path('requestadd/garbage/', requestadd_garbage, name="requestadd_garbage"),
    path('viewRequest/', request_view),
    path('deleteRequest/<str:pk>/', deleteRequest, name="deleteRequest"),
    path('verify/', verify_view),
    path('thanks/', thanks_view),
    path('thanks2/', thanks2_view),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Εικόνα 16. Κώδικας με τις διευθύνσεις URLs

Στην παραπάνω εικόνα παρατηρούμε όλες τις διευθύνσεις που χρησιμοποιεί η διαδικτυακή εφαρμογή. Κάθε διεύθυνση αντιστοιχεί σε μία προβολή.

Το μονοπάτι `path('accounts/login/', login_view)`, αντιστοιχεί στην `login_view` προβολή.

```
def login_view(request):
    next = request.GET.get('next')
    form = UserLoginForm(request.POST or None)
    if form.is_valid():
        username = form.cleaned_data.get('username')
        password = form.cleaned_data.get('password')
        user = authenticate(username=username, password=password)
        login(request, user)
        if next:
            return redirect(next)
        return redirect('/home')
    context = {
        'form': form,
    }
    return render(request, "login.html", context)
```

Εικόνα 17. Κώδικας προβολής `login_view`

Ο χρήστης μέσω της σελίδας στέλνει μια φόρμα σε αίτημα με το όνομα χρήστη και τον κωδικό του. Δεν μπορούμε να εμπιστευόμαστε όλα δεδομένα που εισέρχονται με αιτήματα, καθώς είναι ανεπεξέργαστα δεδομένα που δεν έχουν επικυρωθεί. Δεν πρέπει ποτέ να χρησιμοποιούμε δεδομένα από το αίτημα και να τα αποθηκεύουμε στη βάση δεδομένων μας. Οι φόρμες Django **cleaned_data** μας εξασφαλίζουν ότι οι κανόνες επικύρωσης ορίζουν για αυτήν τη φόρμα είναι ικανοποιημένοι ή όχι. Με την εντολή **form.is_valid()** ελέγχουμε όλες τις επικυρώσεις μας κατά το `request.POST` και δημιουργούμε λεξικό που περιέχει έγκυρα δεδομένα. Έτσι, τα δεδομένα που ανακτώνται από το χαρακτηριστικό `forms.cleaned_data` περνούν από την επικύρωση φόρμας. Εάν είναι ορθά τα στοιχεία παραπέμπει τον χρήστη στην προβολή `home` εάν είναι λάθος του εμφανίζει μήνυμα λάθους.

Το μονοπάτι `path('accounts/register/', register_view)`, αντιστοιχεί στην `register_view` προβολή.

```
def register_view(request):
    next = request.GET.get('next')
    form = UserRegisterForm(request.POST or None)
    if form.is_valid():
        user = form.save(commit=False)
        password = form.cleaned_data.get('password')
        user.set_password(password)
        user.is_active = False
        user.save()
        current_site = get_current_site(request)
        mail_subject = 'Activate your account.'
        message = render_to_string('acc_active_email.html', {
            'user': user,
            'domain': current_site.domain,
            'uid': urlsafe_base64_encode(force_bytes(user.id)),
            'token': account_activation_token.make_token(user),
        })
        to_email = form.cleaned_data.get('email')
        email = EmailMessage(
            mail_subject, message, to=[to_email]
        )
        email.send()
        return HttpResponseRedirect('Please confirm your email address to complete the registration.<a href="http://127.0.0.1')
    context = {
        'form': form,
    }
    return render(request, "signup.html", context)
```

Εικόνα 18. Κώδικας προβολής `register_view`

Ο χρήστης στην περίπτωση που καλέσει την διεύθυνση εγγραφής νέου χρήστη (`http://localhost/accounts/register/`) θα πάρει ως απάντηση μια φόρμα εγγραφής χρήστη. Εφόσον γεμίσει τη φόρμα την στέλνει για να περαστεί στην βάση δεδομένων. Ο παραπάνω κώδικας ελέγχει εάν τα δεδομένα που καταχώρησε ο χρήστης είναι έγκυρα και στην περίπτωση που είναι, ο χρήστης δημιουργείται αλλά δεν είναι ενεργός. Ο χρήστης μπορεί να ενεργοποιήσει τον λογαριασμό του μέσω της ηλεκτρονικής διεύθυνσης του.

Το μονοπάτι `path('full/<str:pk>/',update_full, name="update_full")`, αντιστοιχεί στην `update_full` προβολή.

```
@login_required
def update_full(request, pk):

    garbage=Garbages.objects.get(id=pk)
    garbage.full = "True"
    garbage.username = request.user.username
    garbage.save()

    return redirect('/home')
```

Εικόνα 19. Συνάρτηση ενημέρωσης πληρότητας κάδου

Στην αρχή του κώδικα βλέπουμε την δήλωση `@login_required`. Είναι από την έτοιμη βιβλιοθήκη της Django authentication όπου για να εκτελεστεί η συνάρτηση κάτω από αυτή την δήλωση χρειάζεται ο χρήστης που κάνει την ενέργεια να είναι συνδεδεμένος.

Η συνάρτηση δέχεται ένα αίτημα με ένα μοναδικό κλειδί κάποιου συγκεκριμένου κάδου. Αποθηκεύει τη νέα κατάσταση του συγκεκριμένου κάδου και το όνομα του χρήστη που έκανε την αναφορά. Στην συνέχεια ανακατευθύνει τον χρήστη στην αρχική σελίδα.

Το μονοπάτι `path('broke/<str:pk>/',update_broke, name="update_broke")`, αντιστοιχεί στην προβολή `update_broke`.

```
@login_required
def update_broke(request, pk):

    garbage=Garbages.objects.get(id=pk)
    garbage.broke = "True"
    garbage.username = request.user.username
    garbage.save()

    return redirect('/home')
```

Εικόνα 20. Συνάρτηση ενημέρωσης καταστρεμένου κάδου

Το ίδιο και σε αυτή την περίπτωση χρειάζεται ο χρήστης να είναι συνδεδεμένος για να εκτελεστεί η παραπάνω συνάρτηση. Η συνάρτηση δέχεται ένα

αίτημα με ένα μοναδικό κλειδί κάποιου συγκεκριμένου κάδου. Αποθηκεύει τη νέα κατάσταση του συγκεκριμένου κάδου και το όνομα του χρήστη που έκανε την αναφορά. Στην συνέχεια ανακατευθύνει τον χρήστη στην αρχική σελίδα.

Το μονοπάτι `path('missing/<str:pk>/', update_missing, name="update_missing")`, αντιστοιχεί στην προβολή `update_missing`.

```
@login_required
def update_missing(request, pk):

    garbage=Garbages.objects.get(id=pk)
    garbage.missing = "True"
    garbage.username = request.user.username
    garbage.save()

    return redirect('/home')
```

Εικόνα 21. Συνάρτηση ενημέρωσης αγνοούμενου κάδου

Πάλι στην συγκεκριμένη περίπτωση χρειάζεται ο χρήστης να είναι συνδεδεμένος για να εκτελεστεί η παραπάνω συνάρτηση. Η συνάρτηση δέχεται ένα αίτημα με ένα μοναδικό κλειδί κάποιου συγκεκριμένου κάδου. Αποθηκεύει τη νέα κατάσταση του συγκεκριμένου κάδου και το όνομα του χρήστη που έκανε την αναφορά. Στην συνέχεια ανακατευθύνει τον χρήστη στην αρχική σελίδα.

Το μονοπάτι `path('empty/<str:pk>/',empty, name="empty")`, αντιστοιχεί στην προβολή `empty`.

```
#Admin's and superusers's options
@login_required
def empty(request, pk):
    if request.user.is_superuser:
        garbage=Garbages.objects.get(id=pk)
        garbage.full = "False"
        garbage.save()
        return redirect('/home')
    else:
        return HttpResponse('You are not a SuperUser!')
```

Εικόνα 22. Συνάρτηση ενημέρωσης άδειου κάδου

Όπως αναφέραμε παραπάνω στα σχέδια της διαδικτυακής εφαρμογής, για το άδειασμα των κάδων είναι υπεύθυνοι μόνο οι αρμόδιοι για αυτό στην παραπάνω συνάρτηση πρέπει να είναι συνδεδεμένος ο χρήστης και να είναι **super user** για να μπορεί να ενημερώσει την κατάσταση του κάδου σε «άδειος».

Το μονοπάτι `path('fixed/<str:pk>/',fixed, name="fixed")`, αντιστοιχεί στην προβολή `fixed`.

```
@login_required
def fixed(request, pk):
    if request.user.is_superuser:
        garbage=Garbages.objects.get(id=pk)
        garbage.broke = "False"
        garbage.save()
        return redirect('/home')
    else:
        return HttpResponse('You are not a SuperUser!')
```

Εικόνα 23. Συνάρτηση ενημέρωσης διόρθωσης κάδου

Για την συγκεκριμένη εργασία διορθώσεις των κάδων είναι υπεύθυνοι μόνο οι αρμόδιοι για αυτό στην παραπάνω συνάρτηση πρέπει να είναι συνδεδεμένος ο χρήστης και να είναι **super user** για να μπορεί να ενημερώσει την κατάσταση του κάδου σε «διορθωμένος».

Το μονοπάτι `path('found/<str:pk>/', found, name="found")`, αντιστοιχεί στην προβολή `found`.

```
@login_required
def found(request, pk):
    if request.user.is_superuser:
        garbage=Garbages.objects.get(id=pk)
        garbage.missing = "False"
        garbage.save()
        return redirect('/home')
    else:
        return HttpResponse('You are not a SuperUser!')
```

Εικόνα 24. Συνάρτηση ενημέρωσης ότι βρέθηκε ο κάδος

Για την συγκεκριμένη εργασία αναζήτησης των κάδων είναι υπεύθυνοι μόνο οι αρμόδιοι για αυτό στην παραπάνω συνάρτηση πρέπει να είναι συνδεδεμένος ο χρήστης και να είναι **super user** για να μπορεί να ενημερώσει την κατάσταση του κάδου σε κανονική.

Το μονοπάτι `path('request/add_report/', add_report, name="add_report")`, αντιστοιχεί στην προβολή `add_report`.

```
@login_required
def add_report(request):
    if request.method == 'POST':
        name = request.user.username
        newlat = request.POST["lat"]
        newlng = request.POST["lng"]
        newdescription = request.POST["comment"]
        newimg = request.FILES["img"]
        newReport= Reports(username=name,lat=newlat,lng=newlng,descr=newdescription,image=newimg)
        newReport.save()
        return redirect('/success')
    return HttpResponseForbidden('allowed only via POST')
```

Εικόνα 25. Συνάρτηση αίτημα καθαρισμού παραμελημένων σκουπιδιών

Στην παραπάνω συνάρτηση χρειάζεται να είσαι συνδεδεμένος χρήστης όπου έχεις επιλέξει να κάνεις αναφορά με παραμελημένα σκουπίδια. Με την μέθοδο POST request στέλνονται σε αυτή την συνάρτηση οι συντεταγμένες, η περιγραφή και η εικόνα που ανέβασε ο χρήστης, τα συγκεκριμένα στοιχεία αποθηκεύονται στην βάση για να εμφανιστούν στο αρμόδιο τμήμα όπου θα εξετάσει την κατάσταση.

Το μονοπάτι `path('viewReports/', reports_view)`, αντιστοιχεί στην προβολή `reports_view`.

```
@never_cache
@login_required
def reports_view(request):
    if request.user.is_superuser:
        return render(request, "viewReports.html", {})
```

Εικόνα 26. Συνάρτηση προβολής αιτημάτων χρηστών

Στην παραπάνω συνάρτηση για να αποκτήσεις πρόσβαση πρέπει να είσαι συνδεδεμένος **super user** όπου θα σε κατατοπίσει σε ένα χάρτη με τα αιτήματα των χρηστών.

Ακόμη οι **super users** έχουν την δυνατότητα να προσθέτουν και να αφαιρούν κάδους από τον χάρτη.

Το μονοπάτι `path('add/garbage/', add_garbage, name="add_garbage")`, αντιστοιχεί στην προβολή `add_garbage` όπου οι super users προσθέτουν κάδους και το μονοπάτι `path('delete/<str:pk>/', delete, name="delete")`, αντιστοιχεί στην προβολή `delete` όπου εκεί διαγράφουν τους κάδους.

```
@login_required
def add_garbage(request):
    if request.user.is_superuser:
        if request.method == 'POST':
            newlat = request.POST["lat"]
            newlng = request.POST["lng"]
            markertype = request.POST["markertype"]
            newGarbage = Garbage(lat=newlat, lng=newlng, markertype=markertype, full="False",
                                broke="False", missing="False", time="12:00:00")
            newGarbage.save()
            return redirect('/home')
        else:
            return HttpResponse('You are not a SuperUser')
```

Εικόνα 27. Συνάρτηση προσθήκης κάδου

Στην παραπάνω συνάρτηση έχουν πρόσβαση μόνο οι **super users** όπου εκεί με την μέθοδο `request.POST` εισάγουν τις νέες συντεταγμένες του νέου κάδου και τον τύπο του κάδου όπου και αποθηκεύεται στην βάση δεδομένων.

```

@login_required
def deleteReport(request, pk):
    if request.user.is_superuser:
        report=Reports.objects.get(id=pk)
        report.delete()
        return redirect('/home')
    else:
        return HttpResponse('You are not a SuperUser!')

```

Εικόνα 28. Συνάρτηση διαγραφής κάδου

Ο **super user** στο αίτημα για διαγραφή του συγκεκριμένου κάδου στέλνει το μοναδικό κλειδί αυτού.

4.1.4 Υλοποίηση API rest framework

Όπως αναφέρεται πιο πάνω στο κεφάλαιο 3.3 (*Σύνδεση Backend - Frontend*) χρειαζόμαστε ένα API όπου συνδέει το backend με το frontend. Δημιουργήθηκε ένα api rest framework όπου σειριοποιεί τα δεδομένα από τα μοντέλα της εφαρμογής, ούτως ώστε να μπορούν να αποθηκεύονται και να μεταφέρονται. Στα δεδομένα αυτά θα έχει πρόσβαση και το μπροστινό μέρος της εφαρμογής με μεθόδους json / xml call, δηλαδή θα αιτείται τα δεδομένα ο χρήστης (μέσω κώδικα javascript) και εφόσον είναι εξουσιοδοτημένος θα του επιστρέφονται σε μορφή json έτσι ώστε να μορφοποιηθούν κατάλληλα για να εμφανιστούν στον περιηγητή του χρήστη.

Σειριοποίηση μοντέλων στο rest framework

```

from rest_framework import serializers
from .models import Garbages, Reports

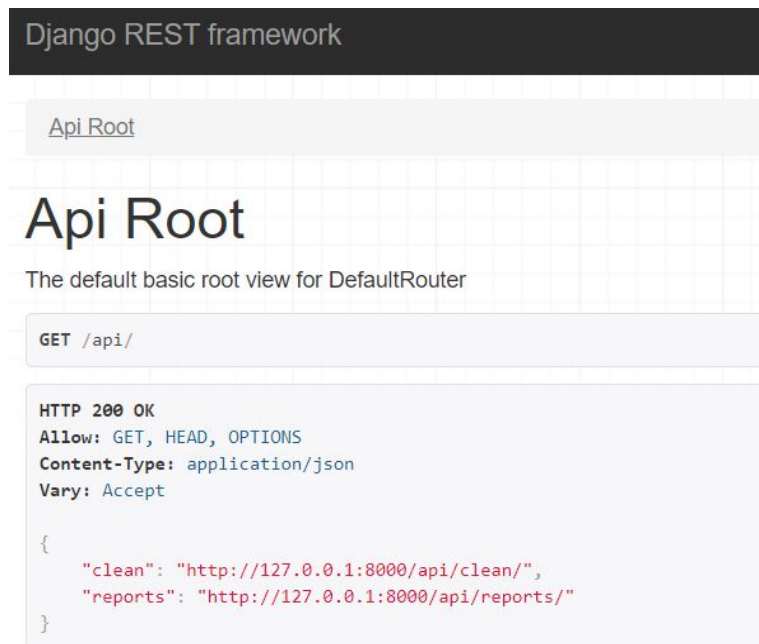
class GarbagesSerializer(serializers.ModelSerializer):
    class Meta:
        model = Garbages
        read_only_fields = ('id','lat','lng','markertype','full','broke','missing','time','username')
        fields = ('id','lat','lng','markertype','full','broke','missing','time','username')

class ReportsSerializer(serializers.ModelSerializer):
    class Meta:
        model = Reports
        fields = ('id','username','lat','lng','descr','image')

```

Εικόνα 29. Κλάσεις σειριοποίησης μοντέλων

Στις παραπάνω δύο κλάσεις παρατηρούμε την διαδικασία σειριοποίησης των μοντέλων σε δεδομένα του rest framework. Στην πρώτη κλάση έχουν πρόσβαση όλοι οι χρήστες άλλα μπορούν μόνο κάνουν ανάγνωση τα δεδομένα δεν μπορούν να τα τροποποιήσουν ή να τα διαγράψουν. Στην δεύτερη κλάση έχουν πρόσβαση μόνο οι Super Users όπου έχουν την πλήρη πρόσβαση. Όπως παρατηρούμε και στις δύο περιπτώσεις επιλέγουμε εμείς ποιά πεδία θελουμε να φαίνονται με την δηλωση του μέσα στις παρενθέσεις της εντολής **fields**.



Εικόνα 30. Django Rest framework

Στην παραπάνω εικόνα βλέπουμε την διεπαφή του χρήστη με το API όπου περιέχει δύο συνδέσμους ο clean που περιέχει τα δεδομένα του μοντέλου Garbages σε μορφή json και ο σύνδεσμος reports που περιέχει τα δεδομένα του μοντέλου Reports.

Garbages List

GET /api/clean/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

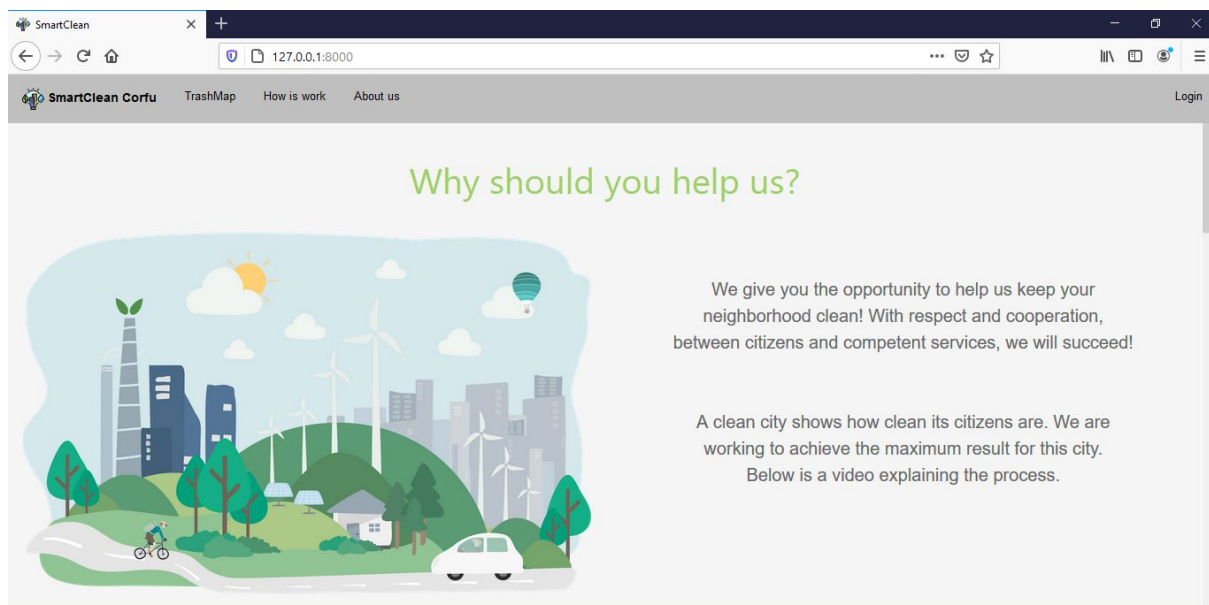
```
[  
  {  
    "id": 1,  
    "lat": 39.625213,  
    "lng": 19.916018,  
    "markertype": 1,  
    "full": false,  
    "broke": false,  
    "missing": false,  
    "time": "12:00:00",  
    "username": ""  
  },  
  ]
```

Εικόνα 31. Δεδομένα μοντέλου Garbages σε json μορφή

4.2 Υλοποίηση web application interface (Frontend)

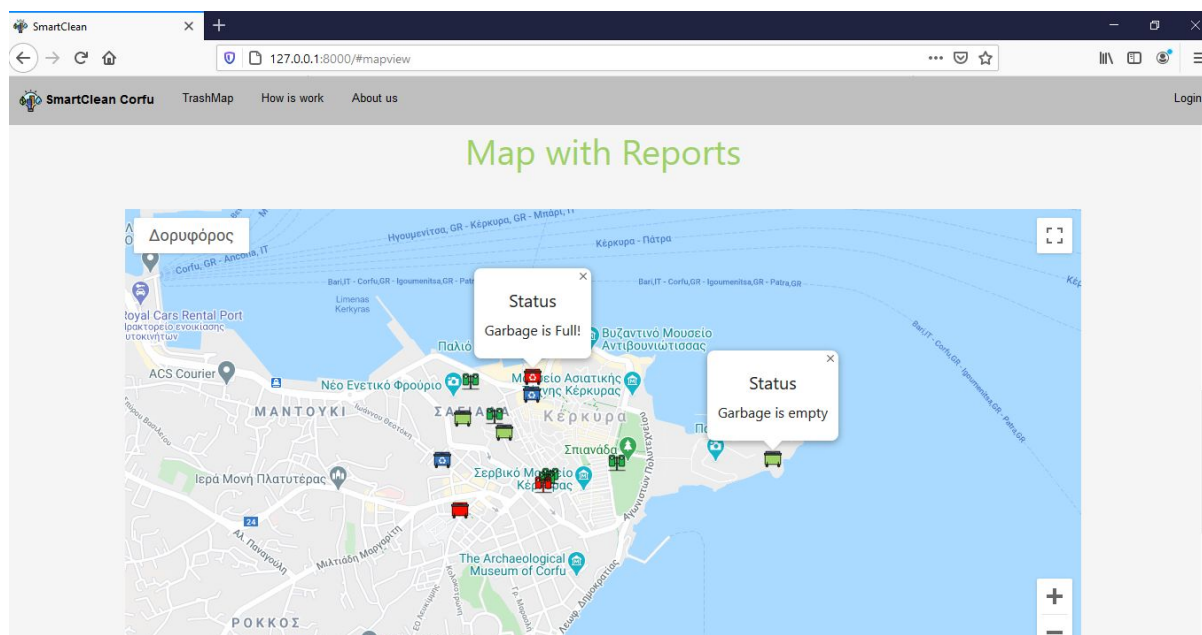
Στο μπροστινό μέρος της εφαρμογής όπως αναφέρθηκε παραπάνω αλληλεπιδρά με τους χρήστες, στην περίπτωσή μας οι χρήστες είναι δύο οι απλοί χρήστες (πολίτες) και οι super user (αρμόδιοι υπηρεσίας καθαριότητας). Σε αυτή την ενότητα θα ασχοληθούμε με το σενάριο χρήσης κάθε πλευράς όπου και θα αναλυθεί σε βάθος η κάθε σελίδα.

4.2.1 Σενάριο χρήσης απλού χρήστη (πολίτη) - Αρχική σελίδα



Εικόνα 32. Πρώτη επαφή με την εφαρμογή.

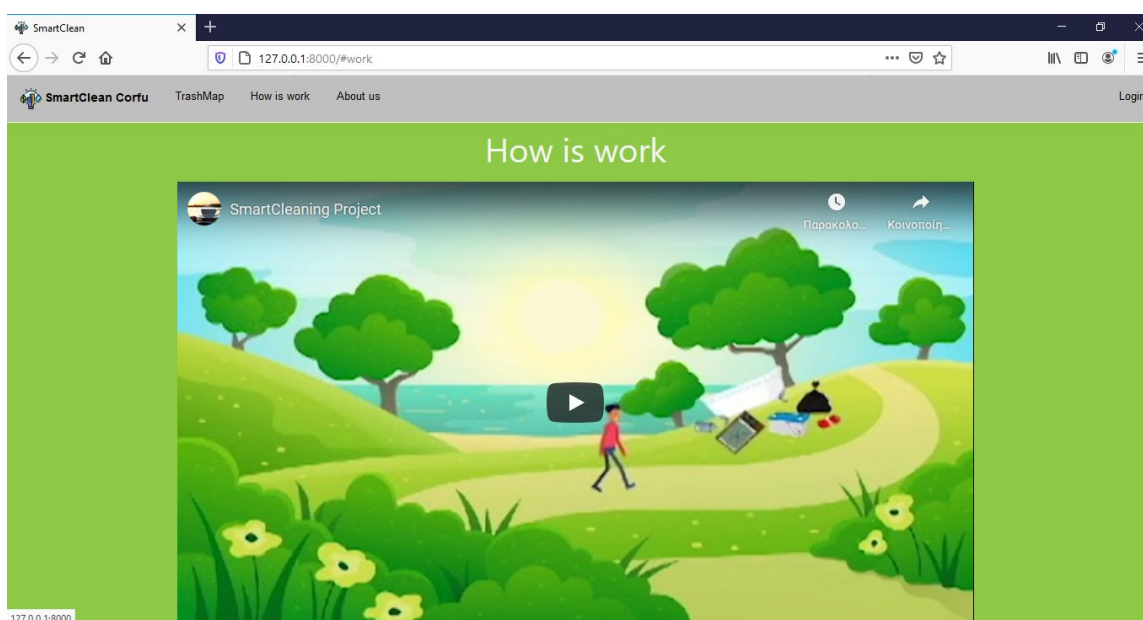
Στο πρώτο κάλεσμα της εφαρμογής ο ανώνυμος χρήστης βλέπει μια πρώτη εικόνα με την εφαρμογή, μια μπάρα από επιλογές (Trash Map, How is work, About us, Login) και μία σύντομη περιγραφή της εφαρμογής.



Εικόνα 33. Χάρτης με καταστάσεις των κάδων.

Λίγο παρακάτω στην ίδια σελίδα χωρίς ακόμα να κάνει συνδεση ο χρήστης μπορεί να παρατηρήσει τις καταστάσεις των κάδων. Για παράδειγμα, ο χρήστης έχει πατήσει στον κόκκινο κάδο και παρατηρεί ότι η κατάσταση του κάδου είναι «γεμάτος». Ενώ δίπλα με κανονικό χρώμα (πράσινο) ο κάδος έχει κατάσταση «άδειος».

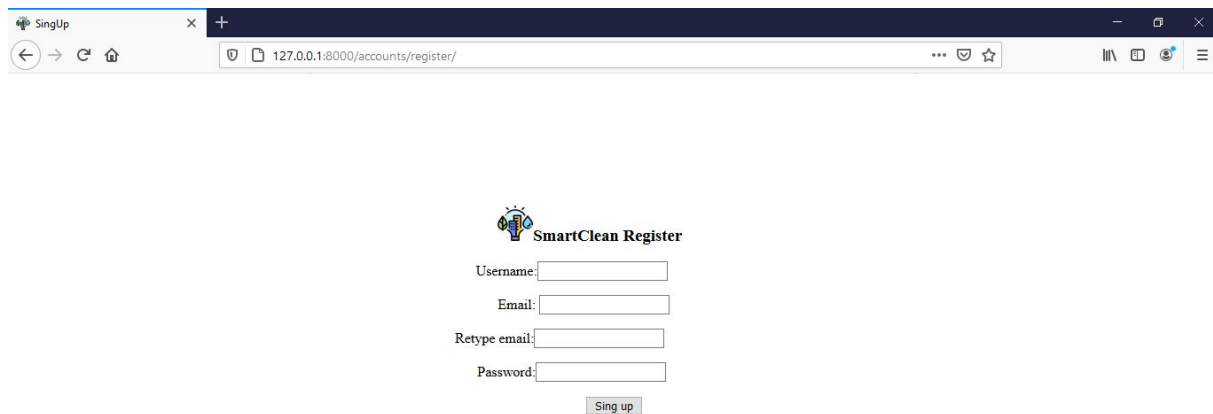
Στην ίδια σελίδα παρακάτω ο χρήστης συναντά ένα βίντεο που τον καθοδηγεί για το πώς θα κάνει λογαριασμό και πως θα χρησιμοποιεί την εφαρμογή.



Εικόνα 34. Βίντεο καθοδήγησης.

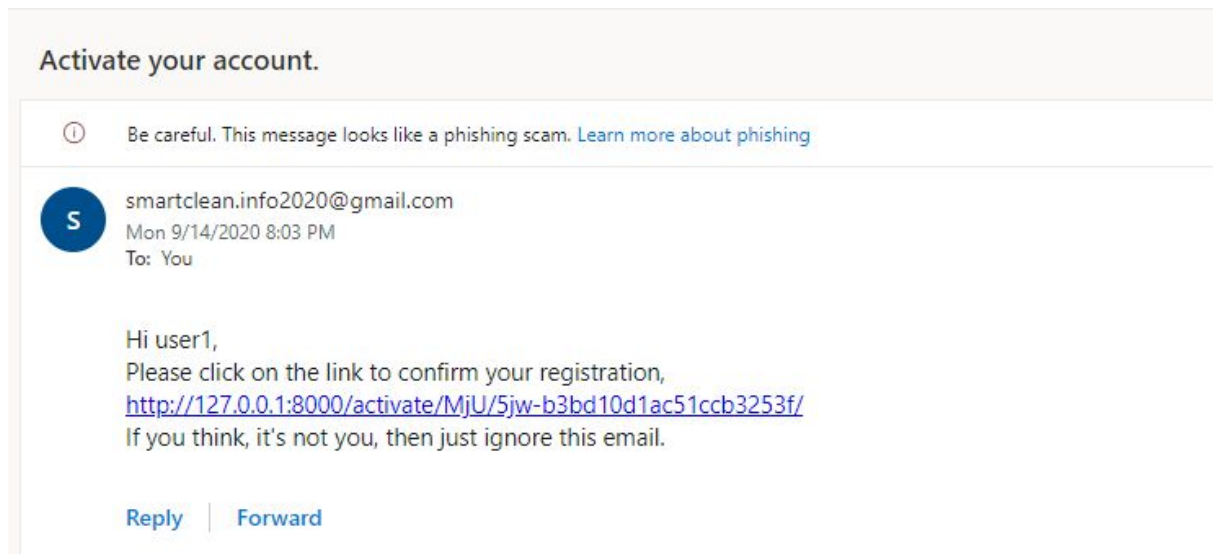
4.2.2 Σενάριο χρήσης απλού χρήστη (πολίτη) - Δημιουργία λογαριασμού

Στην συνέχεια ο πολίτης μπορεί να επιλέξει την δημιουργία λογαριασμού, η εφαρμογή θα τον παραπέμπει σε μια φόρμα εγγραφής χρήστη, όπου ο χρήστης θα γεμίζει τις φόρμες. Εφόσον είναι ορθά τα στοιχεία του ένα ηλεκτρονικό μήνυμα επιβεβαίωσης λογαριασμου θα του στέλνεται στην ηλεκτρονική διεύθυνση του. Για να ενεργοποιήσει το λογαριασμό του πρέπει να κάνει κλικ στο σύνδεσμο επιβεβαίωσης.



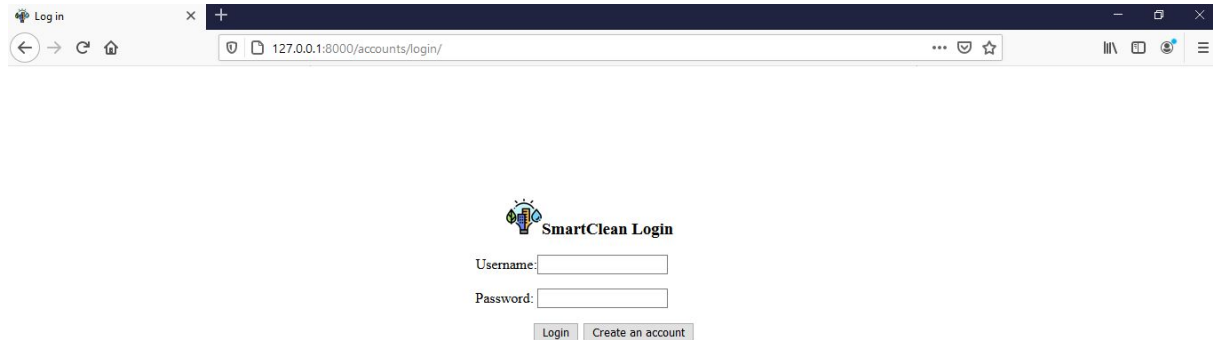
The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/accounts/register/". The page title is "SingUp". The main content area features a logo for "SmartClean Register" and a registration form with the following fields: "Username:", "Email:", "Retype email:", and "Password:". Below these fields is a "Sing up" button.

Εικόνα 35. Φόρμα δημιουργίας λογαριασμού.



Εικόνα 36. Ηλεκτρονικό μήνυμα επιβεβαίωσης χρήστη.

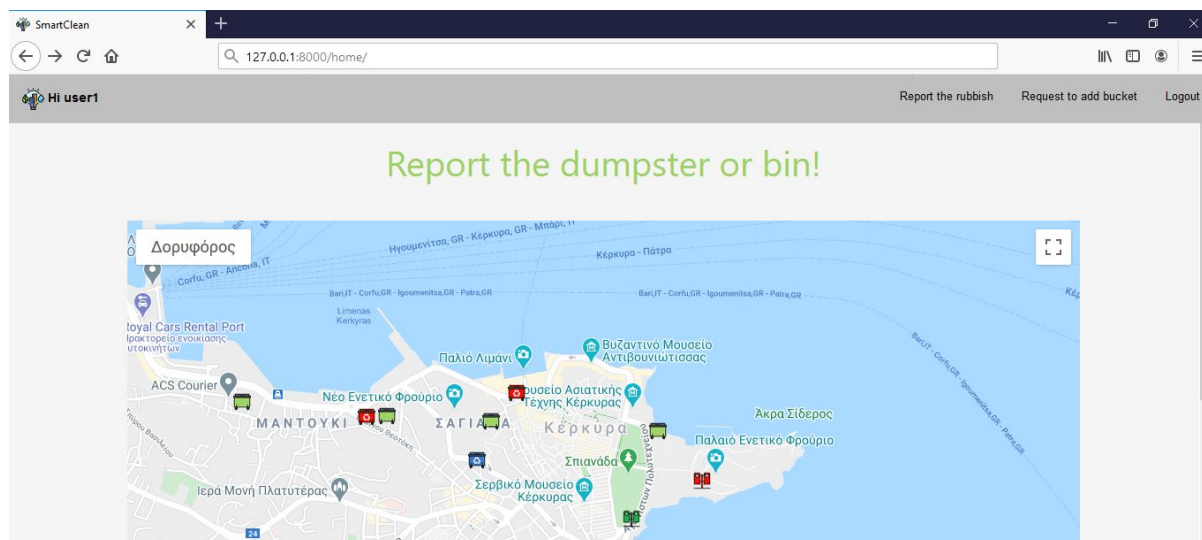
4.2.3 Σενάριο χρήσης απλού χρήστη (πολίτη) - Σύνδεση



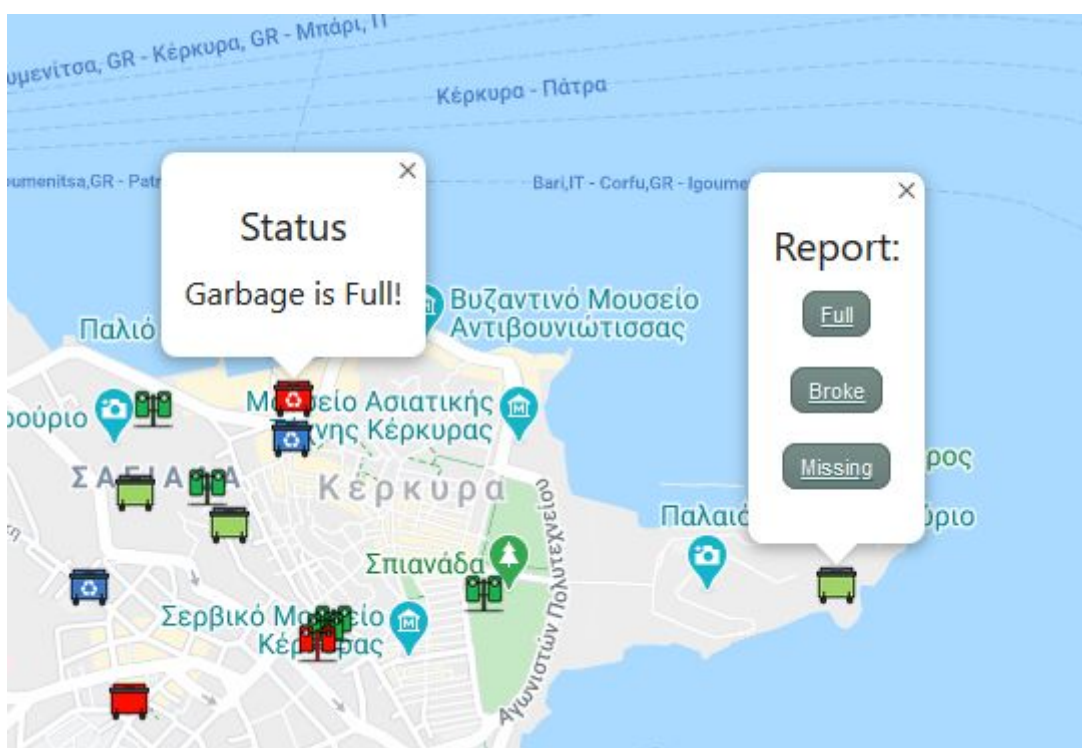
Εικόνα 37. Σελίδα σύνδεσης χρήστη.

Ο χρήστης αφού επιβεβαιώσει το λογαριασμό του μεταβαίνει στην σελίδα σύνδεσης. Συνδέεται στην εφαρμογή και μπορεί να κάνει κάποιες αναφορές αφού είναι ενεργός χρήστης.

4.2.4 Σενάριο χρήσης απλού χρήστη (πολίτη) - Αρχική σελίδα



Εικόνα 38. Αρχική σελίδα χρήστη.

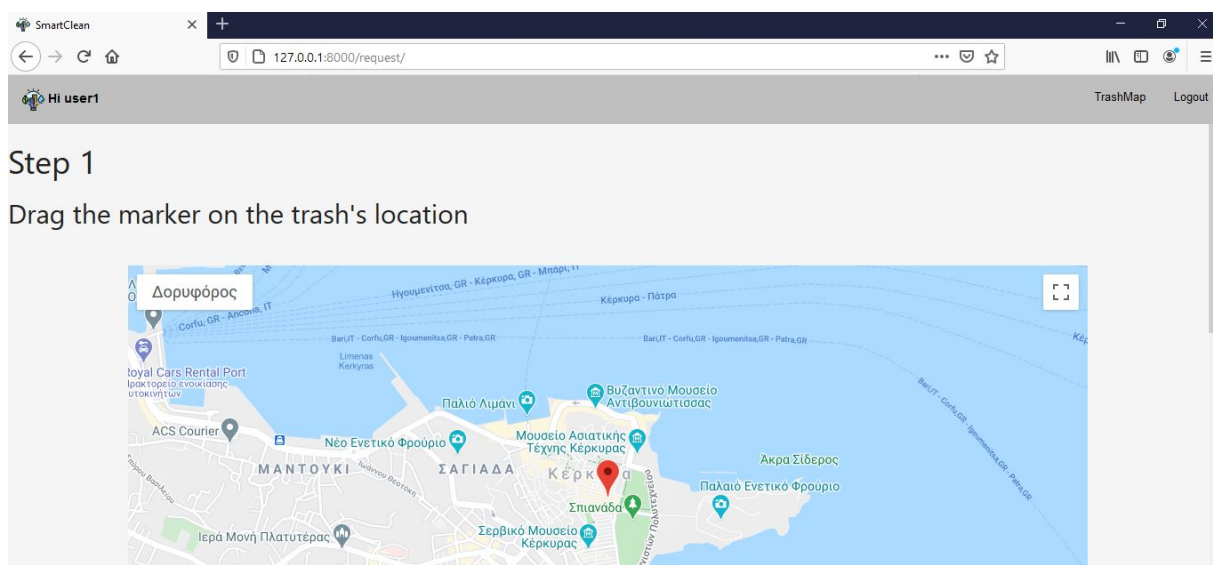


Εικόνα 39. Αρχική σελίδα χρήστη - εστίαση στο χάρτη.

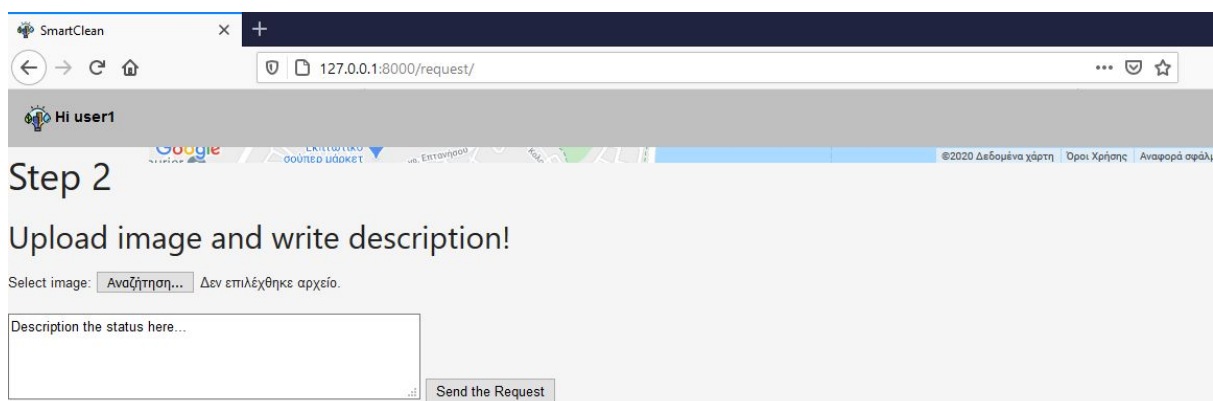
Ο χρήστης έχει την δυνατότητα να αναφέρει τους γεμάτους κάδους ή τους σπασμένους ή τους αγνοούμενους μόνο με δύο κλικ. Πρώτα επιλέγει τον κάδο, ανοίγει ένα παραθυράκι και ο χρήστης αναφέρει την νέα κατάσταση του κάδου.

4.2.5 Σενάριο χρήσης απλού χρήστη (πολίτη) - Αναφορά σκουπιδιών

Ο χρήστης έχει την δυνατότητα να αναφέρει παραμελημένα σκουπίδια πατώντας στην επιλογή Make Report απο την μπάρα επιλογών, όπου εκεί του δίνονται οι οδηγίες πως θα πραγματοποιήσει την αναφορά παραμελημένων σκουπιδιών.



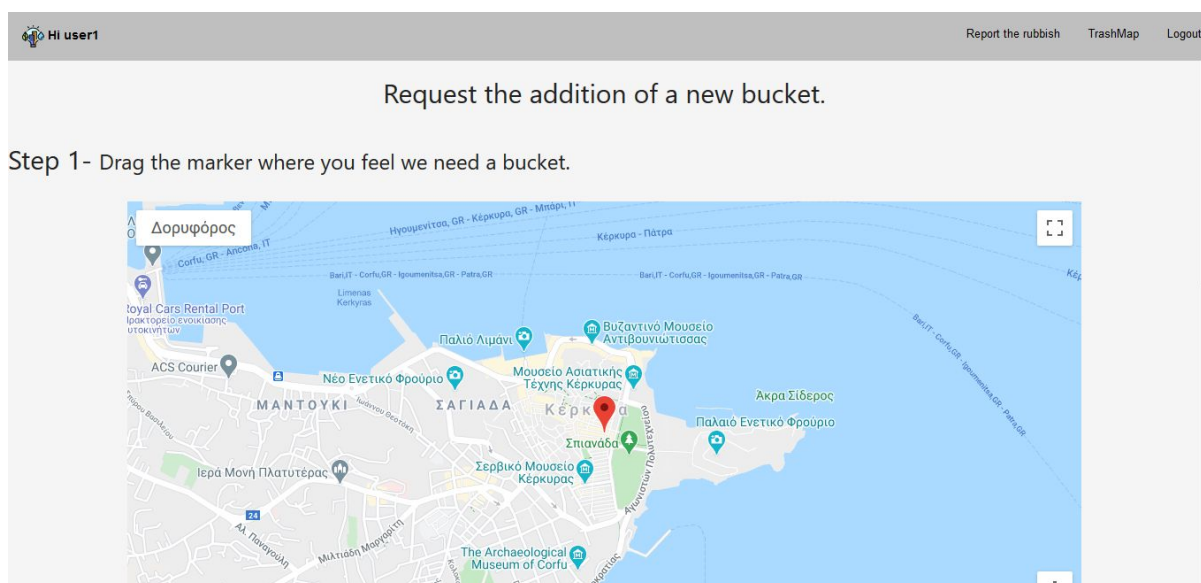
Εικόνα 40. Αναφορά παραμελημένων σκουπιδιών.



Εικόνα 41. Αναφορά παραμελημένων σκουπιδιών. (Συνέχεια)

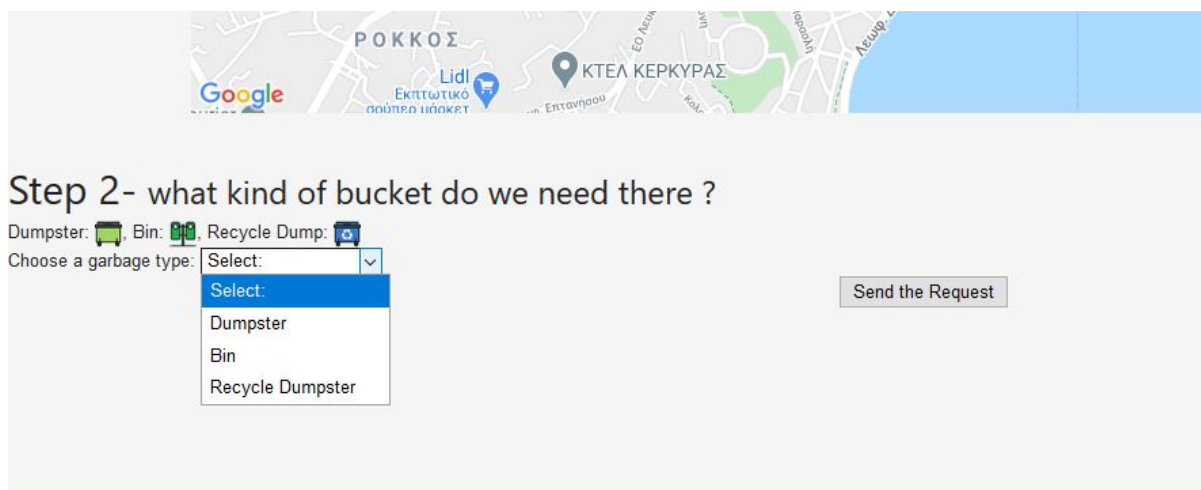
Στην **εικόνα 40** και **41** παρατηρούμε ότι ο χρήστης επέλεξε να κάνει αναφορά στα σκουπίδια που παρατήρησε. Αρχικά του ζητάει να τραβήξει το σημάδι στην τοποθεσία που είναι τα σκουπίδια. Στην συνέχεια του ζητάει να ανεβάσει μια εικόνα με τα παραμελημένα σκουπίδια και να γράψει μια σχετική περιγραφή για αυτά. Αφού τα κάνει όλα αυτά μπορεί να στείλει την αναφορά του όπου και θα εξεταστεί από το αρμόδιο τμήμα καθαριότητας.

4.2.6 Σενάριο χρήσης απλού χρήστη (πολίτη) - Δημιουργία αιτήματος για προσθήκη νέου κάδου



Εικόνα 42. Δημιουργία αιτήματος προσθήκης νέου κάδου. (1ο στάδιο)

Η παραπάνω εικόνα είναι ένα απόσπασμα από την σελίδα όπου ο χρήστης επέλεξε να δημιουργήσει ένα αίτημα για προσθήκη νέου κάδου. Το αρμόδιο τμήμα θα το πάρει στην συνέχεια και θα το επεξεργαστεί το αίτημα. Στο 1ο στάδιο ο χρήστης πρέπει να τραβήξει το σημάδι στην τοποθεσία που επιθυμεί να προστεθεί ο νέος κάδος.



Step 2- what kind of bucket do we need ?

Dumpster: , Bin: , Recycle Dump: 

Choose a garbage type:

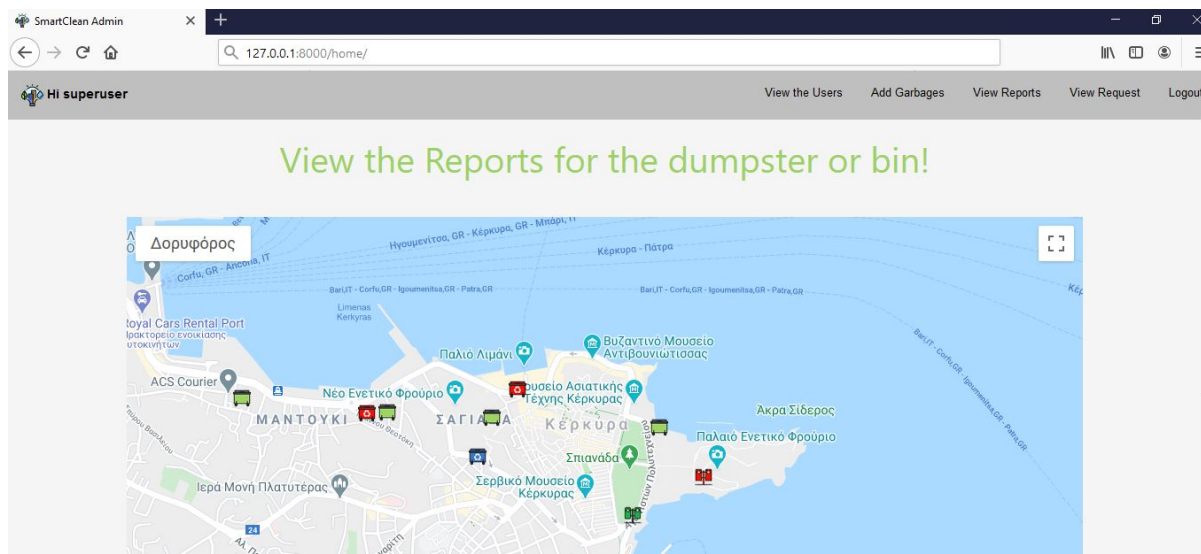
- Select:
- Dumpster
- Bin
- Recycle Dumpster

Εικόνα 43. Δημιουργία αιτήματος προσθήκης νέου κάδου.(2ο στάδιο)

Στην συνέχεια στην ίδια σελίδα παρακάτω, για να ολοκληρωθεί το αίτημα του χρήστη πρέπει να επιλέξει το είδος του κάδου που πιστεύει ότι χρειάζεται. Εφόσον επιλέξει και το είδος του κάδου, στέλνει το αίτημα του στο αρμόδιο τμήμα.

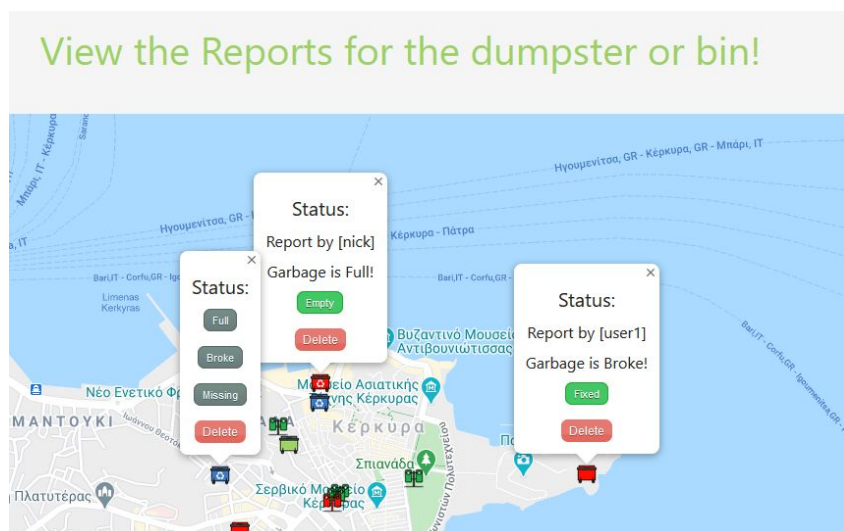
4.2.7 Σενάριο χρήσης super user (αρμόδια υπηρεσία) - Αρχική σελίδα

Σε περίπτωση που πραγματοποιηθεί η σύνδεση και ο χρήστης είναι super user θα ανακατευθύνεται στην αρχική σελίδα του υπερ χρήστη, όπου εκεί υπάρχουν επιλογές προσαρμοσμένες στις ανάγκες του.



Εικόνα 44. Αρχική σελίδα του super user.

Στην αρχική σελίδα του super user συναντάμε μια μπάρα επιλογών (View the Users, Add Garbages, View Reports, View Request, Logout) και στο κέντρο της σελίδας τον χάρτη με τις καταστάσεις των κάδων.

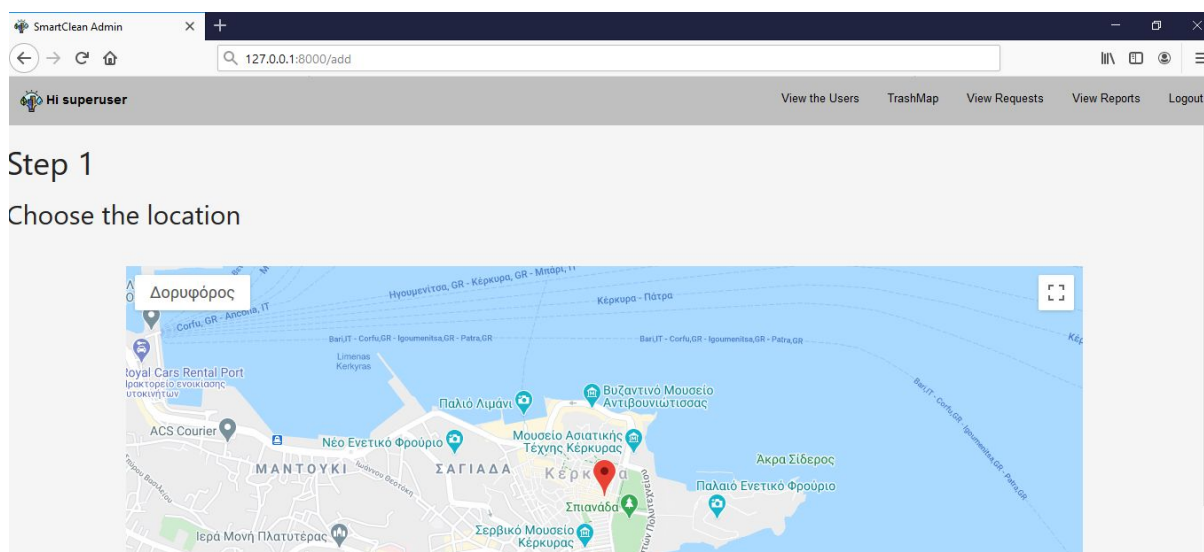


Εικόνα 45. Χάρτης με τους κάδους και τις επιλογές του super user.

Στην παραπάνω εικόνα παρατηρούμε τις επιλογές που μπορεί να εκτελέσει ο super user. Στην περίπτωση που οι κάδοι είναι γεμάτοι ο super user μπορεί να αλλάξει την κατάσταση τους σε άδειους. Επίσης εάν είναι σπασμένος ή αγνοείται ο κάδος μπορεί να τους επαναφέρει σε κανονική κατάσταση. Ο super user έχει την δυνατότητα να βλέπει απο ποιον χρήστη έγινε η αναφορά του κάδου. Μπορεί να διαγράψει όποια στιγμή θέλει όποιον κάδο θέλει.

4.2.8 Σενάριο χρήσης super user (αρμόδια υπηρεσία) - Προσθήκη κάδου

Ο super user έχει το δικαίωμα να προσθέτει και να διαγράφει τους κάδους πάνω στο χάρτη.



Εικόνα 46. Προσθήκη κάδου 1ο στάδιο.

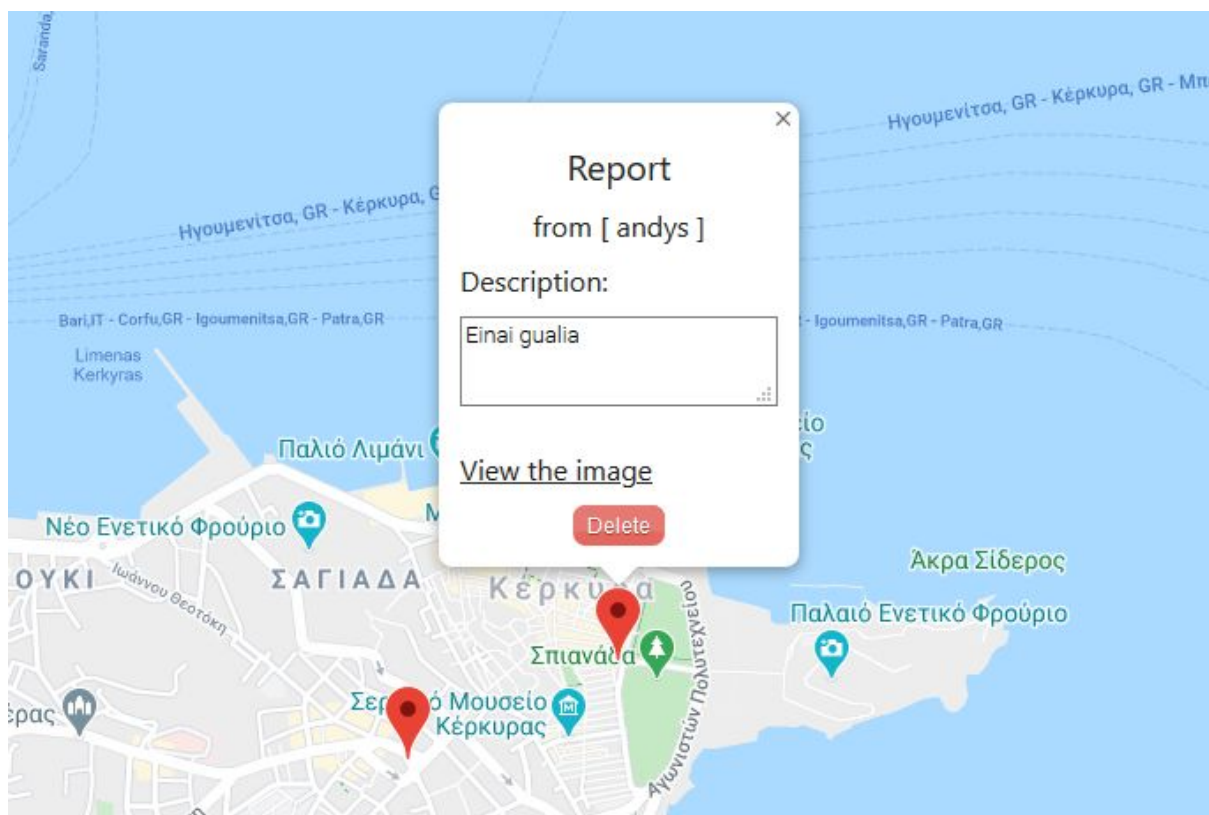


Εικόνα 47. Προσθήκη κάδου 2ο στάδιο.

Για να γίνει η προσθήκη του νέου κάδου ο super user πρέπει να εκτελέσει δύο βήματα: πρώτα πρέπει να επιλέξει την τοποθεσία του νέο κάδο και στην συνέχεια να επιλέξει το είδος του κάδου, εφόσον πραγματοποιήσει αυτά τα δύο βήματα αποθηκεύει το νέο κάδο.

4.2.9 Σενάριο χρήσης super user (αρμόδια υπηρεσία) - Προβολή αναφορών

Σε αυτή τη κατηγορία ο super user μπορεί να διαβάσει τις αναφορές που έκαναν οι χρήστες για τα παραμελημένα σκουπίδια.

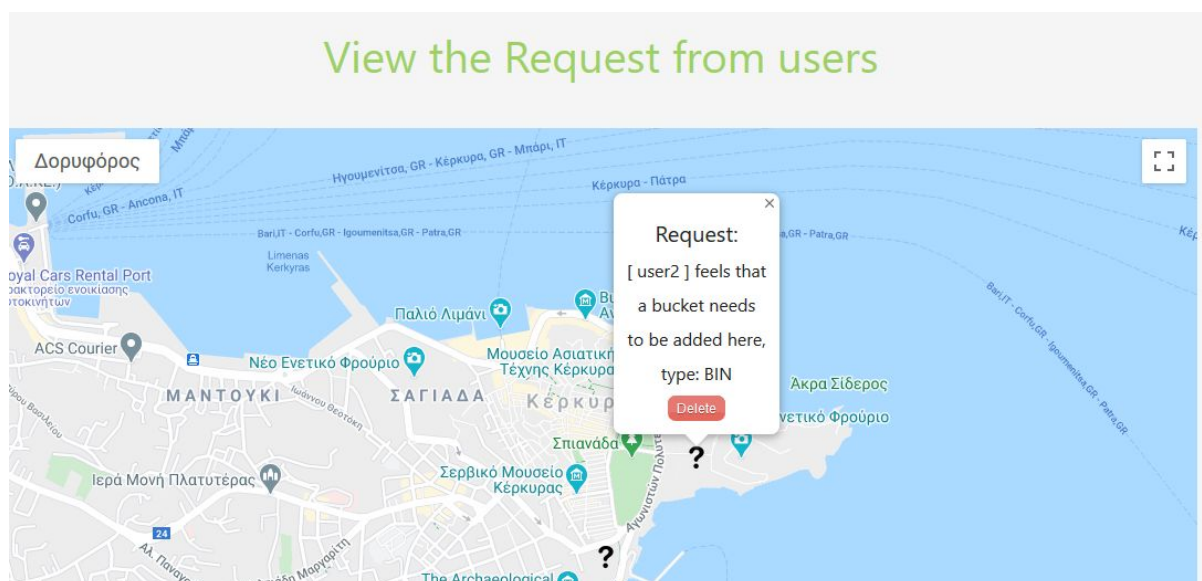


Εικόνα 48. Χάρτης με αναφορές παραμελημένων σκουπιδιών.

Εφόσον πατήσει στο σημάδι ο αρμόδιος ανοίγει ένα παραθυράκι με πληροφορίες σχετικές με τα παραμελημένα σκουπίδια. Για παράδειγμα, γράφει από ποιόν χρήστη έγινε η αναφορά, σύντομη περιγραφή σχετικά με τα σκουπίδια και ένας σύνδεσμος με την εικόνα των σκουπιδιών. Οι αρμόδια υπηρεσία εφόσον το τακτοποιείσει έχει την δυνατότητα να διαγράψει την αναφορά.

4.2.10 Σενάριο χρήσης super user (αρμόδια υπηρεσία) - Προβολή αιτημάτων

Σε αυτή την υποενότητα θα δούμε τη σελίδα με τα αιτήματα προσθήκης κάδων των χρηστών.



Εικόνα 49. Χάρτης με τα αιτήματα των χρηστών.

Στο πιο πάνω παράδειγμα εμφανίζονται δύο ερωτηματικά στο χάρτη των super users, αυτά είναι δύο αιτήματα των πολιτών. Το ένα από τα δύο είναι ανοιχτό και περιγράφει ότι ο χρήστης με όνομα user2 αισθάνεται ότι χρειάζεται να προστεθεί ένας κάδος στην τοποθεσία του ερωτηματικού με τύπο κάδου BIN (ο μικρός κάδος). Εφόσον το αρμόδιο τμήμα εξετάσει το αίτημα και συμφωνήσει με τον χρήστη θα τοποθετήσει τον κάδο ψηφιακά πάνω στον χάρτη όπως το σενάριο **4.2.6 Προσθήκη κάδου**.

4.2.11 Σενάριο χρήσης super user (αρμόδια υπηρεσία) - Προβολή χρηστών

Select user to change

Q Search

Action: Go 0 of 3 selected

☐	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
☐	admin	[REDACTED]			✓
☐	superuser	[REDACTED]@hotmail.com			✓
☐	user1	[REDACTED]hotmail.com			✗

3 users

Εικόνα 50. Προβολή χρηστών

Permissions

☒ Active
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

☐ Staff status
Designates whether the user can log into this admin site.

☐ Superuser status
Designates that this user has all permissions without explicitly assigning them.

Εικόνα 51. Επιλογές εξουσιοδότησης χρήστη.

Ο super user είναι εξουσιοδοτημένος να ενεργοποιεί και να απενεργοποιεί τυχόν κακόβουλους χρήστες, να τους διαγραφεί τελείως εφόσον δεν ενεργούν με τους κανόνες της εφαρμογής.

4.1 Heroku

Το Heroku είναι μια πλατφόρμα που προσφέρει υπηρεσία (Platform as a Service – PaaS) δηλαδή είναι ένας τρόπος ενοικίασης φιλοξενίας, λειτουργικών συστημάτων, αποθηκευτικού χώρου και δικτυακής υποδομής μέσω του διαδικτύου. Αυτό το είδος υπηρεσιών επιτρέπει στο προγραμματιστή να ενοικιάσει εικονικοποιημένους (Virtualized) διακομιστές και συνεργαζόμενες υπηρεσίες με σκοπό να φιλοξενήσουν υπάρχουσες εφαρμογές ή να αναπτύσσουν και να δοκιμάσουν νέες.

Η συγκεκριμένη πλατφόρμα είναι κομψή, ευέλικτη και εύκολη στη χρήση, προσφέρει στους προγραμματιστές τον απλούστερο δρόμο για την διαδικτυακή φιλοξενία των εφαρμογών τους.

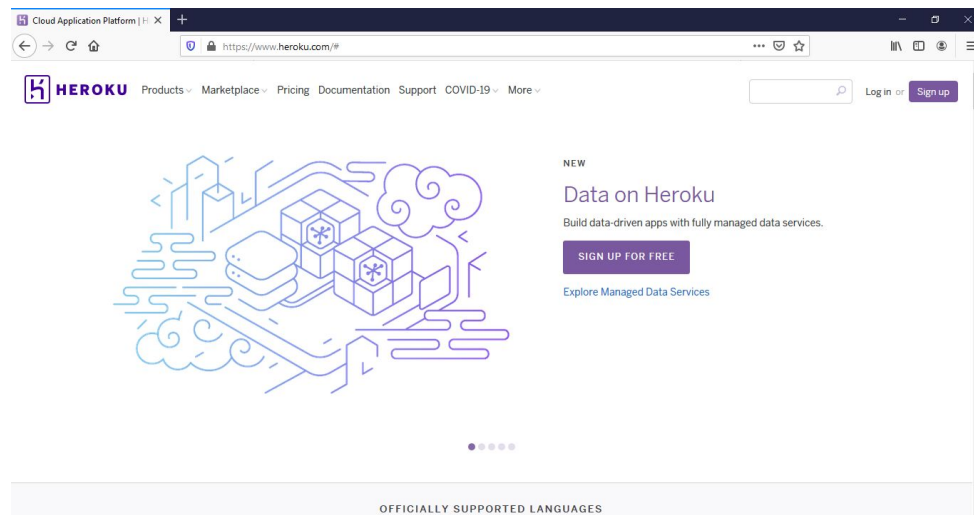
Το Heroku διαχειρίζεται πλήρως το κομμάτι του server side, δίνοντας στους προγραμματιστές την ελευθερία να επικεντρωθούν στο βασικό τους στόχο χωρίς να αποσπά την προσοχή τους για τη συντήρηση διακομιστών, υλικού ή υποδομής. Η πλατφόρμα του Heroku παρέχει υπηρεσίες, εργαλεία, ροές εργασίας και υποστήριξη polyglot - όλα σχεδιασμένα για να βελτιώσουν την παραγωγικότητα του προγραμματιστή.

Τα πλεονεκτήματα μιας πλατφόρμας που προσφέρει υπηρεσία (PaaS):

- Τα συστατικά του λειτουργικού συστήματος μπορούν να παραμετροποιηθούν και να αλλάξουν συχνά.
- Οι ομάδες ανάπτυξης μπορούν να εργάζονται από διαφορετική γεωγραφική τοποθεσία σε projects ανάπτυξης λογισμικού χρησιμοποιώντας υπηρεσίες που εκτείνονται πέρα από τα εθνικά σύνορα.
- Όσον αφορά το κόστος, ελαχιστοποιείται καθώς γίνεται καλύτερη αξιοποίηση των υλικών υποδομών.

4.2 Διαδικασία μεταφόρτωσης

Σε αυτή την υποενότητα περιγράφεται η διαδικασία η οποία ακολουθήθηκε για να τρέξει η διαδικτυακή εφαρμογή στο διαδίκτυο (online). Αρχικά επισκεπτόμαστε την ιστοσελίδα του Heroku για να δημιουργήσουμε δωρεάν λογαριασμό και στην συνέχεια το εγκαθιστούμε στον υπολογιστή μας.



Εικόνα 52. Επίσκεψη ιστοσελίδας Heroku.

```
Terminal - andreas@andreas-Lenovo-ideapad-300-15ISK: ~/Desktop/project online
File Edit View Terminal Tabs Help
andreas@andreas-Lenovo-ideapad-300-15ISK:~/Desktop/project online$ heroku
CLI to interact with Heroku

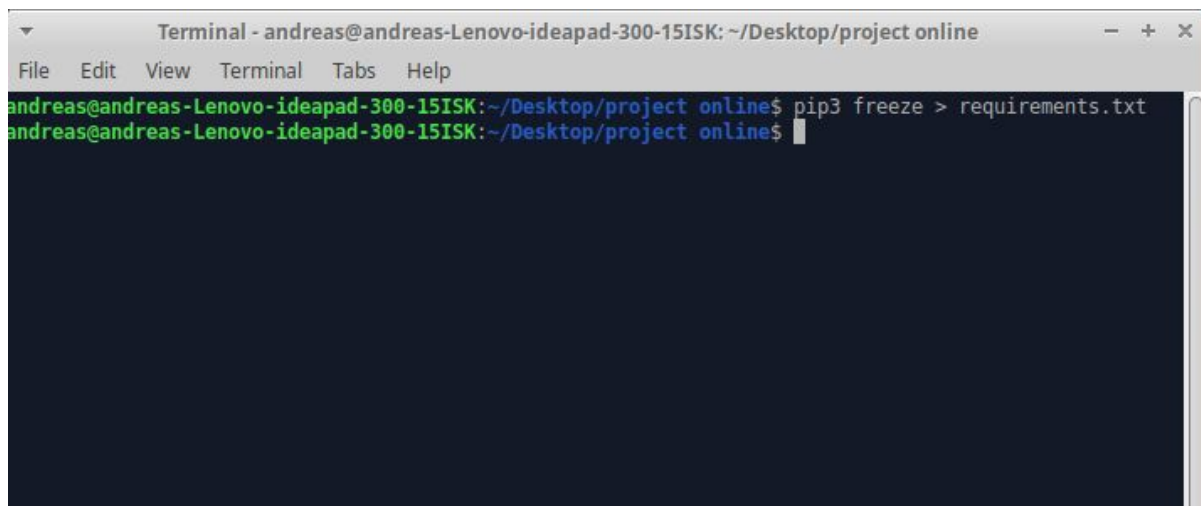
VERSION
  heroku/7.43.1 linux-x64 node-v12.16.2

USAGE
  $ heroku [COMMAND]

COMMANDS
  access          manage user access to apps
  addons          tools and services for developing, extending, and operating your app
  apps            manage apps on Heroku
  auth           check 2fa status
  authorizations  OAuth authorizations
  autocomplete    display autocomplete installation instructions
  buildpacks      scripts used to compile apps
  certs           a topic for the ssl plugin
  ci              run an application test suite on Heroku
  clients         OAuth clients on the platform
```

Εικόνα 53. Εγκατάσταση του Heroku.

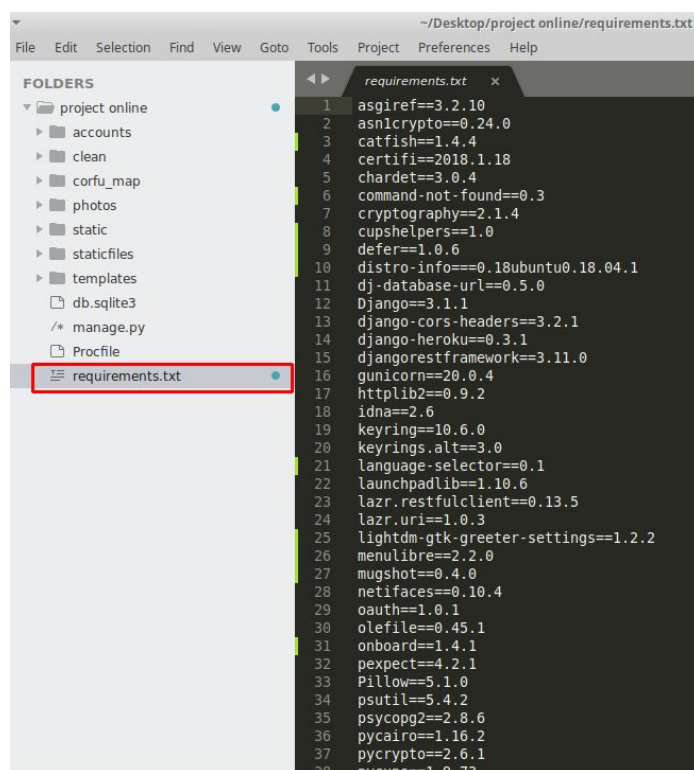
Εκτελούμε την εντολή **pip3 freeze > requirements.txt** για να δημιουργήσουμε ένα αρχείο κειμένου με τις απαιτήσεις της διαδικτυακής εφαρμογής μας. Το συγκεκριμένο αρχείο θα αποθηκευτεί στο φάκελο του έργου μας.



```
Terminal - andreas@andreas-Lenovo-ideapad-300-15ISK: ~/Desktop/project online
File Edit View Terminal Tabs Help
andreas@andreas-Lenovo-ideapad-300-15ISK:~/Desktop/project online$ pip3 freeze > requirements.txt
andreas@andreas-Lenovo-ideapad-300-15ISK:~/Desktop/project online$
```

Εικόνα 54. Δημιουργία αρχείου απαιτήσεων εφαρμογής.

Το αρχείο απαιτήσεων δημιουργήθηκε με επιτυχία όπως παρατηρούμε παρακάτω.



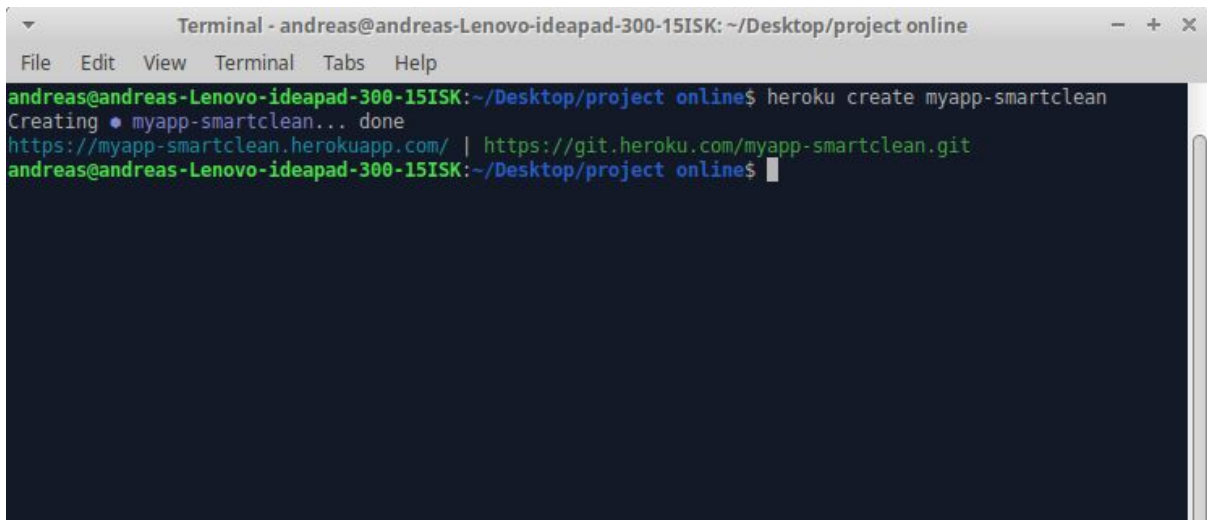
```
~/Desktop/project online/requirements.txt
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  project online
    accounts
    clean
    corfu_map
    photos
    static
    staticfiles
    templates
    db.sqlite3
    /* manage.py
    Procfile
    requirements.txt
requirements.txt
1 asgiref==3.2.10
2 asn1crypto==0.24.0
3 catfish==1.4.4
4 certifi==2018.1.18
5 chardet==3.0.4
6 command-not-found==0.3
7 cryptography==2.1.4
8 cupshelpers==1.0
9 defer==1.0.6
10 distro-info==0.18ubuntu0.18.04.1
11 dj-database-url==0.5.0
12 Django==3.1.1
13 django-cors-headers==3.2.1
14 django-heroku==0.3.1
15 django-rest-framework==3.11.0
16 gunicorn==20.0.4
17 httplib2==0.9.2
18 idna==2.6
19 keyring==10.6.0
20 keyrings.alt==3.0
21 language-selector==0.1
22 launchpadlib==1.10.6
23 lazr.restfulclient==0.13.5
24 lazr.uri==1.0.3
25 lightdm-gtk-greeter-settings==1.2.2
26 menulibre==2.2.0
27 mugshot==0.4.0
28 netifaces==0.10.4
29 oauth==1.0.1
30 olefile==0.45.1
31 onboard==1.4.1
32 pexpect==4.2.1
33 Pillow==5.1.0
34 psutil==5.4.2
35 pycopg2==2.8.6
36 pycairo==1.16.2
37 pycrypto==2.6.1
38 pycups==1.9.73
```

Εικόνα 55. Αρχείο απαιτήσεων εφαρμογής.

Στο αρχείο απαιτήσεων βλέπουμε όλα τα εργαλεία με την συγκεκριμένη έκδοση που χρησιμοποιεί η εφαρμογή μας. Με αυτό τον τρόπο δείχνουμε στην

πλατφόρμα heroku τι πρέπει να εγκαταστήσει όταν ανεβάσουμε τη εφαρμογής μας στην υπηρεσία του.

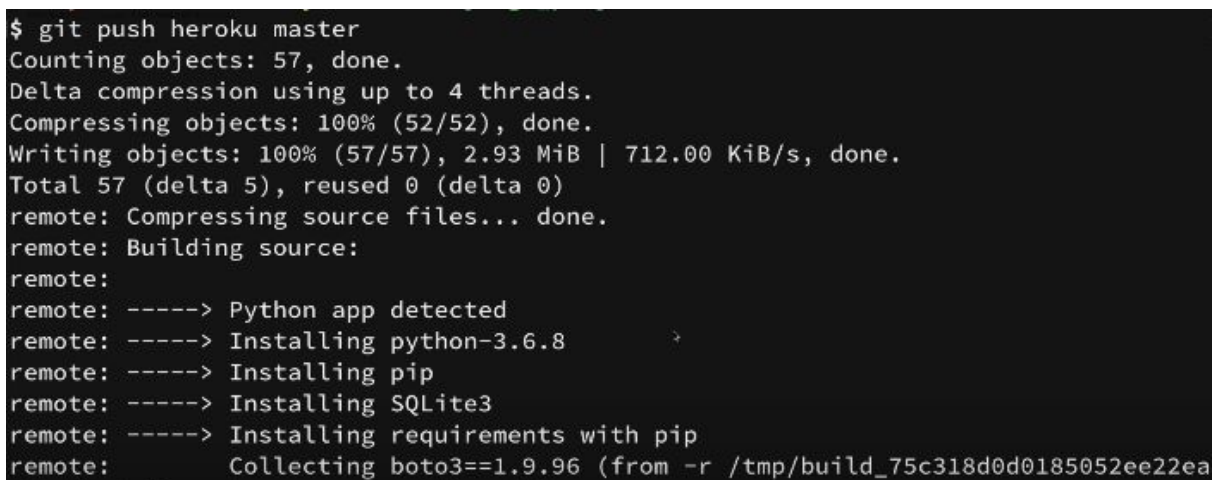
Στην συνέχεια με την εντολή **heroku create myapp-smartclean** δημιουργεί μια νέα εφαρμογή με όνομα myapp-smartclean.



```
Terminal - andreas@andreas-Lenovo-ideapad-300-15ISK: ~/Desktop/project online
File Edit View Terminal Tabs Help
andreas@andreas-Lenovo-ideapad-300-15ISK:~/Desktop/project online$ heroku create myapp-smartclean
Creating • myapp-smartclean... done
https://myapp-smartclean.herokuapp.com/ | https://git.heroku.com/myapp-smartclean.git
andreas@andreas-Lenovo-ideapad-300-15ISK:~/Desktop/project online$
```

Εικόνα 56. Δημιουργία νέας εφαρμογής στο heroku.

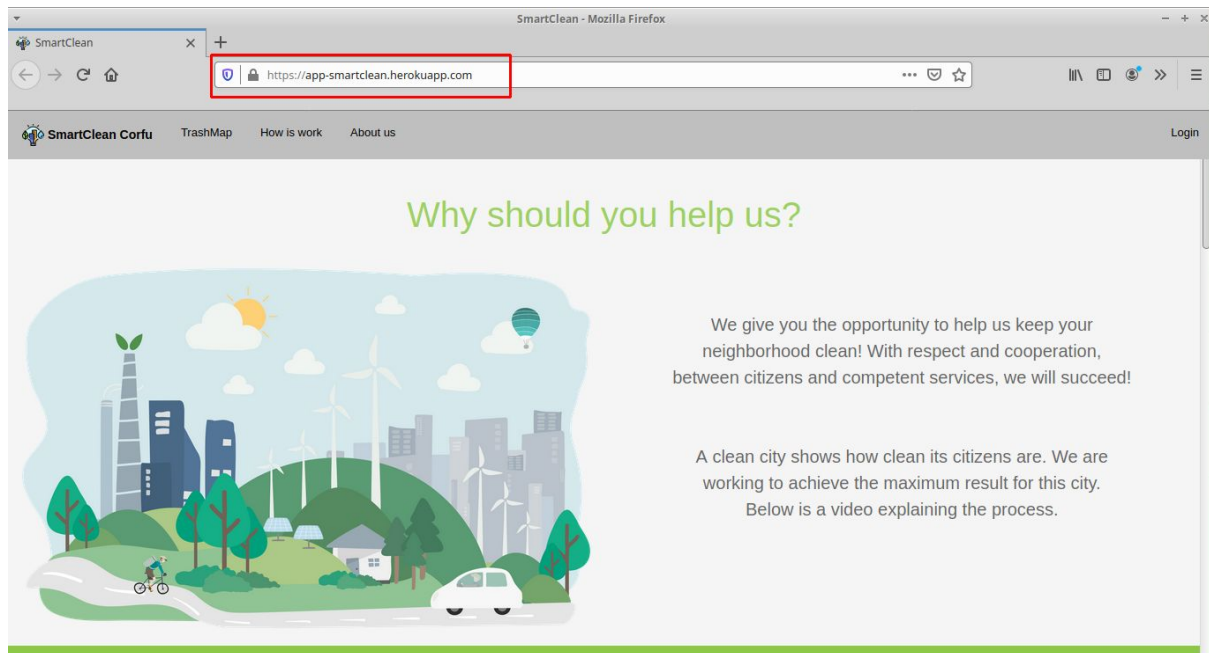
Ανεβάζουμε τους φακέλους του έργου με την εντολή **git push heroku master** ούτως ώστε η εφαρμογή να τρέχει online.



```
$ git push heroku master
Counting objects: 57, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (52/52), done.
Writing objects: 100% (57/57), 2.93 MiB | 712.00 KiB/s, done.
Total 57 (delta 5), reused 0 (delta 0)
remote: Compressing source files... done.
remote: Building source:
remote:
remote: ----> Python app detected
remote: ----> Installing python-3.6.8
remote: ----> Installing pip
remote: ----> Installing SQLite3
remote: ----> Installing requirements with pip
remote: Collecting boto3==1.9.96 (from -r /tmp/build_75c318d0d0185052ee22ea
```

Εικόνα 57. Ανέβασμα εφαρμογής στο heroku.

Τέλος παρατηρούμε ότι η διαδικτυακή εφαρμογή τρέχει online όπου ο κάθε ένας μπορεί να την καλέσει με όνομα <https://myapp-smartclean.herokuapp.com>.



Εικόνα 58. Η διαδικτυακή εφαρμογή online.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Η πτυχιακή αυτή εκτός από το γεγονός ότι είναι ενδιαφέρουσα προγραμματιστικά και αρκετά μοντέρνα, λύνει ένα καθημερινό πρόβλημα που απασχολεί χιλιάδες πολίτες στο κόσμο. Εάν όχι όλοι, οι περισσότεροι πολίτες επιθυμούν να ζουν σε μια καθαρή πόλη χωρίς σκουπίδια και ακαθαρσίες. Η παρούσα διαδικτυακή εφαρμογή καλεί τους πολίτες να συνεργαστούν με τις αρμόδιες υπηρεσίες καθαριότητας για να πετύχουν το κύριο στόχο τους δηλαδή να γίνουν καθαρές όλες τις τοποθεσίες με σκουπίδια.

Κατα την διάρκεια της υλοποίησης της διαδικτυακής εφαρμογής, μου δόθηκε η ευκαιρία να ασχοληθώ με γλώσσες προγραμματισμού και τεχνολογίες που δεν είχα ασχοληθεί στο παρελθόν. Ανέπτυξα σημαντική εμπειρία και τεχνικές που θα με βοηθήσουν στο μέλλον. Με την καθοδήγηση του επιβλέπον καθηγητή μου, έμαθα καινούργιες στρατηγικές για το πώς να λύνω τα προβλήματα που αφορούν τις προγραμματιστικές λειτουργίες.

Όσον αφορά την εφαρμογή, μπορούμε να πούμε ότι ο στόχος επιτεύχθηκε διότι καλύπτει όλες τις ανάγκες και της απαιτήσεις του συγκεκριμένου προβλήματος καθαριότητας. Το γεγονός ότι όλες οι λειτουργίες της εφαρμογής είναι απλές και εύχρηστες έχει ως αποτέλεσμα να χρησιμοποιείται από διάφορες ηλικίες. Είναι μία εφαρμογή που έχει την δυνατότητα να πετύχει σε παγκόσμιο επίπεδο γιατί όλοι οι άνθρωποι στον πλανήτη μπορούν να την χρησιμοποιήσουν για να επωφεληθούν οι ίδιοι αλλά και οι συνάνθρωποι τους. Δεν είναι κρυφό ότι υπάρχουν περιθώρια βελτίωσης στο σχεδιαστικό και λειτουργικό κομμάτι.

Σχετικά με τα προβλήματα και τα σφάλματα που παρουσιάστηκαν κατά την ανάπτυξη της εφαρμογής ήταν αρκετά, αλλά με υπομονή και πίστη επιλύθηκαν όλα.

6.1 Μελλοντική Επέκταση

Παρακάτω αναφέρονται κάποιες βελτιώσεις που μπορούν να γίνουν στην εφαρμογή σε μελλοντική εργασία.

Στο κομμάτι όπου ο χρήστης κάνει αναφορά για κάποιο κάδο ότι είναι γεμάτος ή σπασμένος ή αγνοείται μπορούμε να προσθέσουμε βαθμούς. Για κάθε ορθή αναφορά που κάνει θα παίρνει ένα ποσό από βαθμούς όπου αυτοί οι βαθμοί θα εξαργυρώνονται με δώρα ή εκπτωτικά κουπόνια. Επίσης στο κομμάτι που αναφέρει για παραμελημένα σκουπίδια να προσθέσουμε περισσότερους βαθμούς. Με αυτό το τρόπο θα προκαλέσουμε το ενδιαφέρον των χρηστών.

Στο κομμάτι όπου η αρμόδια υπηρεσία βλέπει στον χάρτη τις αναφορές των χρηστών και πηγαίνει στο κάθε κάδο για να επιλύσει το πρόβλημα, μπορούμε να προσθέσουμε ένα αλγόριθμο ο οποίος να βρίσκει την πιο σύντομη διαδρομή στο super user απο το σημείο που βρίσκεται με όλους τους κάδους που έχουν πρόβλημα.

Παράρτημα

Τα αρχεία της διαδικτυακής εφαρμογής βρίσκονται σε αυτό το σύνδεσμο :
<https://github.com/andreasgeorgiou/smartclean>

Βιβλιογραφία - Δικτυογραφία

- [1] «Ανάπτυξη Διαδικτυακών εφαρμογών θεωρία και Πράξη» - Παναγιώτης Δ. Κεντερλής, 2009
- [2] Εφαρμογή FixMyStreet, <https://www.fixmystreet.com/>
- [3] Εφαρμογή TrashOut, <https://www.trashout.ngo/>
- [4] Ubuntu description, <https://ubuntu.com/>
- [5] Django web framework, <https://www.djangoproject.com/>
- [6] Google maps, <https://developers.google.com/maps/documentation>
- [7] «Beginning HTMLS and CSS3», Christopher Murphy, Richard Clark, Oli Studholme και Divya Manian, 2012
- [8] «Web Application Architecture», Leon Shklar και Richard Rosen, 2003
- [9] Βοηθήματα για τα σφαλματα στον κώδικα, <http://stackoverflow.com/>
- [10] Βοηθήματα για Html, CSS, Javascript <https://www.w3schools.com/>