ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΑΠΟΜΑΚΡΥΣΜΕΝΗ ΔΙΑΧΕΙΡΙΣΗ ΑΥΤΟΜΑΤΟΥ ΠΟΤΙΣΜΑΤΟΣ ΓΙΑ ΓΕΩΡΓΙΚΗ ΧΡΗΣΗ

Μαρίνος Παπαδάκης

Επιβλέπων: Φώτιος Βαρτζιώτης,

Λέκτορας

Ηράκλειο, Ιανουάριος, 2019

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΑΠΟΜΑΚΡΥΣΜΕΝΗ ΔΙΑΧΕΙΡΙΣΗ ΑΥΤΟΜΑΤΟΥ ΠΟΤΙΣΜΑΤΟΣ ΓΙΑ ΓΕΩΡΓΙΚΗ ΧΡΗΣΗ

Μαρίνος Παπαδάκης

Επιβλέπων: Φώτιος Βαρτζιώτης,

Λέκτορας

Ηράκλειο, Ιανουάριος, 2019

REMOTE AUTOMATIC WATERING FOR AGRICULTURAL USE

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Άρτα, 2019

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής

   Όνομα: Φώτιος

   Επίθετο: Βαρτζιώτης

   Βαθμίδα: Λέκτορας

2. Μέλος επιτροπής

   Όνομα:

   Επίθετο:

   Βαθμίδα:

3. Μέλος επιτροπής

   Όνομα:

   Επίθετο:

   Βαθμίδα:

Ο/Η Προϊστάμενος/η του Τμήματος

Όνομα:

Επίθετο:

Βαθμίδα:

Υπογραφή

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα πτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Παπαδάκης, Μαρίνος

Υπογραφή

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω την οικογένεια μου που μου έδωσαν την δυνατότητα να σπουδάσω όλα αυτά τα χρόνια. Επίσης τον κ. Βαρτζιώτη Φώτιο για την καθοδήγηση και την στήριξη των ιδεών μου κατά την διάρκεια εκπόνησης της παρακάτω πτυχιακής εργασίας. Τέλος τον φίλο Σμυρλή Μιχαήλ για την βοήθεια στην μεθοδικότητα λειτουργίας σε διάφορα μέρη του προγράμματος.

ΠΕΡΙΛΗΨΗ

Στην παρακάτω πτυχιακή εργασία θα αναλυθεί ο τρόπος δημιουργίας και χρήσης μιας ιστοσελίδας ελέγχου απομακρυσμένου αυτόματου ποτίσματος για γεωργική χρήση. Για την υλοποίηση της ιστοσελίδας έχουν χρησιμοποιηθεί οι τεχνολογίες Java Servlets, MySQL, JavaScript, και Html. Όλα τα παραπάνω φιλοξενούνται σε έναν τοπικό Server, τον Tomcat 9 ο οποίος είναι εγκατεστημένος σε έναν πολύ μικρού μεγέθους ηλεκτρονικό υπολογιστή που ονομάζεται Raspberry Pi 3. Με την συνεργασία όλων των παραπάνω, ο χρήστης θα μπορεί να επιλέγει ποτίσματα που αφορούν την παρούσα χρονική στιγμή και μελλοντικά. Επίσης θα έχει την δυνατότητα να καταχωρίσει το email του μέσω της ιστοσελίδας στην βάση δεδομένων για να λαμβάνει ενημερώσεις σχετικά με την πορεία των ποτισμάτων. Ιδιαίτερη έμφαση έχει δοθεί όσον αναφορά το λογισμικό περιβάλλον με το οποίο θα έρχεται σε επαφή ο χρήστης, έτσι ώστε να μην δημιουργούνται τυχόν δυσλειτουργίες κατά την επιλογή ενός προγραμματισμένου ή παρόντος ποτίσματος. Ο απομακρυσμένος έλεγχος του αυτοματισμού θα γίνεται με την χρήση της διαδικτυακής υπηρεσίας No-Ip, εφόσον ο μικροϋπολογιστής θα βρίσκεται στην πηγή νερού όπου είναι τοποθετημένο το μοτέρ ποτίσματος.

Λέξεις-κλειδιά: Ιστοσελίδα, Απομακρυσμένο, Πότισμα, Μικροϋπολογιστής

ABSTRACT

In the following thesis, the design and development of a web site with remote automatic irrigation control for agricultural use will be analyzed. The technologies of Java Servlets, MySQL, JavaScript, and Html have been used to implement the site. All of the above are hosted on a local Tomcat 9 server, which is installed on a small single-board computer named Raspberry Pi 3. With the collaboration of all the above, the user will be able to choose watering actions that are relevant at this time and in the future. He will also be able to register his email via the website interface in the database to get updates on the watering progress. Particular emphasis has been placed on the software environment, in order to provide a user-friendly platform to avoid any malfunctions and possible errors when choosing a schedule or present watering. Remote control of the device will be reached by using the No-Ip internet service, because the watering system is installed near at the water source.

Keywords: Raspberry Pi 3, Remote, Watering, Site, Database

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΔΙΑΓΡΑΜΜΑΤΩΝ/ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΕΙΣΑΓΩΓΗ

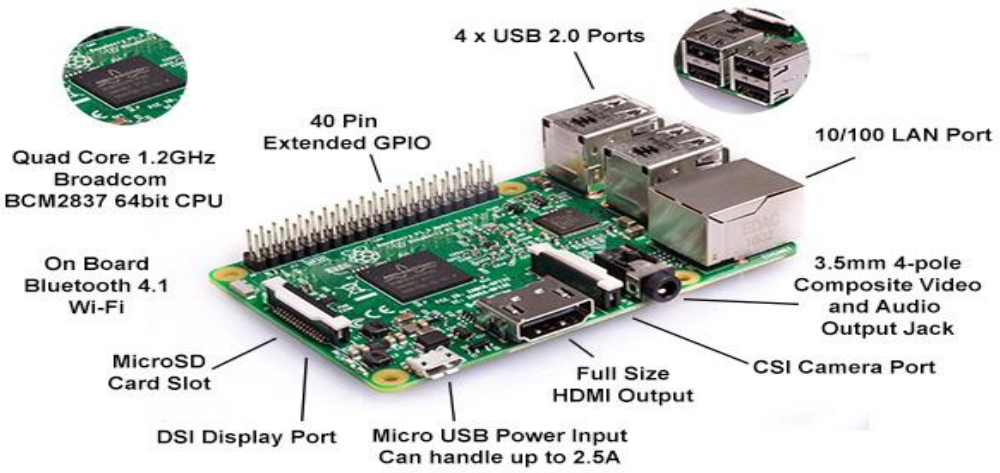
Στις μέρες μας η τεχνολογία έχει γίνει κομμάτι της ζωής του ανθρώπου. Συνεχώς προκύπτουν νέες ανάγκες οι οποίες ωθούν στην ολοένα και ταχύτερη εξέλιξη της. Πιο συγκεκριμένα το IoT (διαδίκτυο των πραγμάτων), το οποίο επιτρέπει τον έλεγχο και την διαχείριση διάφορων ηλεκτρονικών συσκευών, έχει απλοποιήσει την διεπαφή του χρήστη με αυτές. Για να δημιουργηθούν και να υλοποιηθούν τα παραπάνω υπάρχουν πολλοί συνδυασμοί λογισμικού και συσκευών που μπορούν να μας δώσουν ένα επιθυμητό αποτέλεσμα. Η ανάγκη που με οδήγησε στην δημιουργία ενός αυτοματισμού απομακρυσμένης διαχείρισης ποτίσματος δημιουργήθηκε από τον πατέρα μου, καθώς κάθε φορά που ήθελε να βάλει σε λειτουργία το μοτέρ ποτίσματος έπρεπε να πάει στον χώρο τον οποίο βρισκόταν. Έτσι, έχοντας διδαχτεί σε παλαιοτέρα εξάμηνα της σχολής μου Java Servlets, MySql και Html, χρησιμοποιώντας τον μικροϋπολογιστή Raspberry Pi 3 δημιούργησα μια εφαρμογή διαδικτύου η οποία διαχειρίζεται απομακρυσμένα και απλούστερα την διεξαγωγή ενός ποτίσματος. Το περιβάλλον συγγραφής κώδικα που χρησιμοποίησα είναι το Eclipse διότι με αυτό είχα την μεγαλύτερη οικειότητα κατά την πορεία μου ως φοιτητής. Με λίγα λόγια η διαδικτυακή εφαρμογή που υλοποιήθηκε για αυτήν την πτυχιακή εργασία θα είναι σε θέση να λαμβάνει μια επιλογή χρονικής διάρκειας ενός ποτίσματός από τον χρήστη, που αφορά το παρών ή την προγραμματισμένη διεξαγωγή του κατά την διάρκεια μιας εβδομάδας. Επίσης ο χρήστης θα έχει την δυνατότητα να λαμβάνει ειδοποιήσεις σχετικά με την πορεία του ποτίσματός στο email του, εφόσον έχει κάνει εγγραφή στην υπηρεσία που εμπεριέχεται στην εφαρμογή. Στα παρακάτω κεφάλαια θα αναλυθούν εν μέρη οι τεχνολογίες και οι συσκευές οι οποίες χρησιμοποιήθηκαν για να γίνει όσο το δυνατόν πιο κατανοητό στον αναγνώστη ο τρόπος λειτουργίας του παραπάνω αυτοματισμού. Στο κεφάλαιο 1 παρακάτω θα δούμε τι είναι το Raspberry Pi 3, τα τεχνικά χαρακτηρίστηκα του και μερικές από τις χρήσεις που μπορεί να έχει. Στο κεφάλαιο 2 θα αναφέρουμε τι είναι οι τεχνολογίες εφαρμογών διαδικτύου, Java Servlets, Html, JavaScript, Tomcat καθώς επίσης και η MySQL. Στο κεφάλαιο 3 θα απεικονισθούν γραφήματα με μερική ανάλυση, που προσδιορίζουν τον τρόπο λειτουργίας της εφαρμογής. Τέλος στο κεφάλαιο 4 παρατίθεται ένα εγχειρίδιο χρήσης της εφαρμογής Watering.

1. ΤΙ ΕΙΝΑΙ ΤΟ RASPBERRY PI.

Το Raspberry Pi θα το χαρακτηρίζαμε σαν ένα υπολογιστή σε μέγεθος τηλεκάρτας. Αναπτύχθηκε στο Ηνωμένο Βασίλειο με σκοπό την παρότρυνση των μαθητών στα σχολεία για εκμάθηση της πληροφορικής. Με την πάροδο του χρόνου έγινε πολύ δημοφιλές λόγο του μεγέθους και της χρηστικότητας του σε τομείς όπως η ρομποτική, ο προγραμματισμός, οι αυτοματισμοί κ.α. Οι πωλήσεις του έως τον Μάρτιο του 2018 σύμφωνα με τον οργανισμό Raspberry Pi Foundation είχαν φτάσει τα 19 εκατομμύρια παγκοσμίως. Εάν συνδέσουμε ποντίκι, πληκτρολόγιο, οθόνη και μια κάρτα μνήμης με λειτουργικό Noobs ή Raspian (linux) θα μπορούσαμε να πούμε ότι έχουμε ένα ολοκληρωμένο υπολογιστικό σύστημα για γενική χρήση με τιμή που δεν ξεπερνάει τα 40 ευρώ. Στόχος της εταιρείας είναι το μικρότερο δυνατό μέγεθος με όσο γίνεται μεγαλύτερη επεξεργαστική ισχύ. Οι διαδικτυακές κοινότητες που έχουν δημιουργηθεί για την επεκτασιμότητα και την αντιμετώπιση προβλημάτων για τον παραπάνω πρωτότυπο μικροϋπολογιστή ολοένα και πληθαίνουν με τους χρήστες του να είναι σε θέση να επιλύσουν οποιοδήποτε πρόβλημα. [1]

1.1 ΤΕΧΝΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΗΚΑ RASPBERRY PI 3

Το τελευταίο μοντέλο Raspberry Pi 3 που κυκλοφόρησε τον Φεβρουάριο του 2016 με διαστάσεις 8,5 χ 5,6 εκ, μέγεθος στο οποίο δεν μας είχαν συνηθισμένους στην αρχή, θα χρησιμοποιηθεί στην παρούσα πτυχιακή εργασία. Όσον αναφορά τα μηχανήματα που το απαρτίζουν, περιλαμβάνει on-Board επεξεργαστή quad core στα 1,2GHz, μνήμη ram χωρητικότητας 1 Gb, προσαρμογέα δικτύου ethernet και Wi-Fi, έξοδο full size hdmi, bluetooth, θέση για κάρτα μνήμης τύπου MicroSD, έξοδο audio jack 3.5mm, έξοδο οθόνης τύπου dsi και τέλος είσοδο για camera τύπου csi. Επίσης ένα από τα βασικότερα χαρακτηρίστηκα του είναι τα 40 προγραμματιζόμενα pin που διαθέτει όπου ο χρήστης μπορεί να χρησιμοποιήσει σαν είσοδο ή σαν έξοδο. Η τροφοδοσία του μπορεί να γίνει με έναν φορτιστή κινητού 2.5A και καλωδίου τύπου B-Micro Usb. [1]



Εικόνα 1.1 Raspberry Pi 3

1.2 ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΚΑΙ ΜΕΙΟΝΕΚΤΗΜΑΤΑ RASPBERRY PI 3

ΠΛΕΟΝΤΕΚΤΗΜΑΤΑ

Συνοψίζοντας όλα τα στοιχειά πρωτοτυπίας του Raspberry Pi 3 θα μπορούσαμε να πούμε πως τα πλεονεκτήματα του είναι τα εξής.

- Χαμηλό κόστος.
- Μικρό μέγεθος.
- Εύκολη σύνδεση στο διαδίκτυο χάρις των on-board προσαρμογέων δικτύου.
- Συμβατό με τις επικρατέστερες γλώσσες προγραμματισμού.
- Χρησιμοποιεί Linux και πιο συγκεκριμένα ειδικά διαμορφωμένες εκδόσεις ανάλογα την χρήση του.

ΜΕΙΟΝΕΚΤΗΜΑΤΑ

Το κύριο χαρακτηριστικό του όμως, το μικρό μέγεθος δημιουργεί τα παρακάτω αρνητικά στοιχεία.

- Χαμηλή επεξεργαστική ισχύ για μεγάλο φόρτο εργασίας.
- Δεν διαθέτει έξοδο αναλογικού σήματος
- Δεν διαθέτει real time clock (Για να κρατάει την ώρα πρέπει να είναι σε λειτουργία).

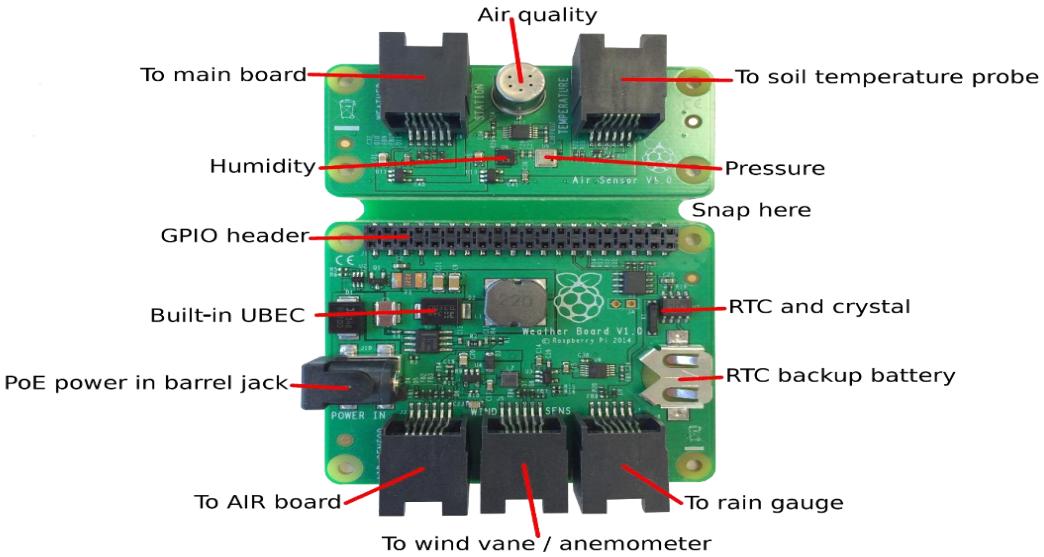
Ο ρυθμός ανάπτυξης της τεχνολογίας είναι πολύ γρήγορος με αποτέλεσμα να μην μπορούν όλοι οι χρήστες να ακολουθήσουν σε κάθε βήμα του. Αυτό δημιουργεί σαφώς όλα τα παραπάνω πλεονεκτήματα και μειονεκτήματα.

1.3 ΓΕΝΙΚΕΣ ΧΡΗΣΕΙΣ ΤΟΥ RASPBERRY PI 3

Το Raspberry Pi 3 έχει σαν συσκευή πολλές εφαρμογές. Μερικές από αυτές είναι δημιουργία του κατασκευαστή με την υποστήριξη λογισμικού, και άλλες διαφορών χρηστών που το χρησιμοποιούν. Παρακάτω θα δούμε μερικές από αυτές και θα αναλύσουμε συνοπτικά τον τρόπο λειτουργίας τους.

1.3.1 ΜΕΤΕΟΡΟΛΟΓΙΚΟΣ ΣΤΑΘΜΟΣ

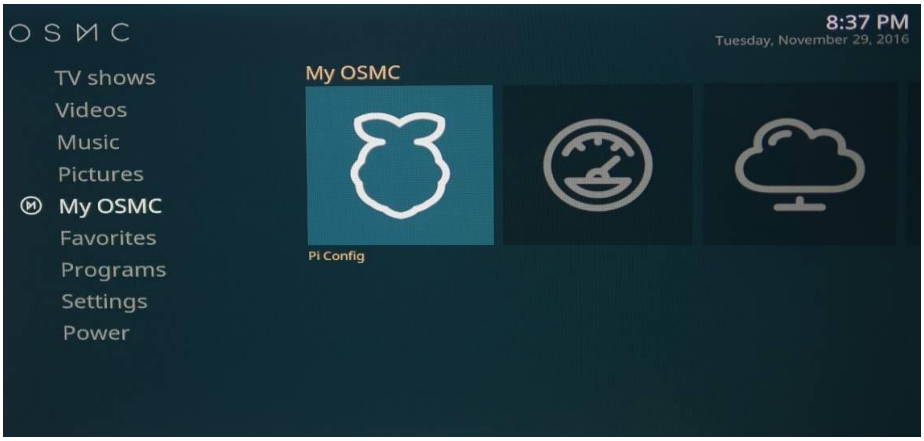
Χρησιμοποιώντας αισθητήρες μέτρησης του αέρα, της υγρασίας, της πίεσης, της βροχής και της θερμοκρασίας ο χρήστης μπορεί να σαν έχει είσοδο στους ακροδέκτες του Raspberry Pi 3 διάφορες τιμές που σε συνεργασία με το καταλληλά διαμορφωμένο λογισμικό που παρέχεται δωρεάν από τον κατασκευαστή του, μπορεί να τις προβάλει ή να τις επεξεργαστεί. Λαμβάνοντας υπόψιν λοιπόν μια τιμή από τις παραπάνω θα μπορούσαμε να τερματίσουμε η να ξεκινήσουμε μια άλλη διεργασία. Σαν παράδειγμα θα πάρουμε το παρών θέμα της πτυχιακής εργασίας, έτσι λοιπόν εάν ο αισθητήρας βροχής καταγράψει μια τιμή που υπερβαίνει το 1 χιλιοστό τότε η τρέχων διεργασία ποτίσματος θα διακόπτεται. [2]



Εικόνα 1.2 Weather Station

1.3.2 OSMC (OPEN SOURCE MEDIA CENTER)

Το OSMC είναι ένα λογισμικό ανοιχτού κώδικα που δημιουργήθηκε το 2014 στο οποίο ο χρήστης μπορεί να προβάλει αρχεία βίντεο ή να ακούσει μουσική που είναι αποθηκευμένη σε έναν σκληρό δίσκο, στο τοπικό δίκτυο ή στο Internet. Βάση της εύκολης συνδεσιμότητας που έχει το Raspberry Pi 3 με συσκευές όπως τηλεόραση και προτζέκτορα οι δυνατότητες των συσκευών αυτών μπορούν να διευρυνθούν ακόμα περισσότερο. Για να κάνουμε τα παραπάνω θα πρέπει να έχουμε κατεβάσει από την επίσημη ιστοσελίδα το λογισμικό και να έχουμε στην κατοχή μας ένα καλώδιο hdmi για την σύνδεση με την συσκευή προβολής. Το λογισμικό για την εγκατάσταση του χρειάζεται μερικά λεπτά, οι τύποι αρχείων που μπορεί να προβάλει είναι πάρα πολλοί και η διεπαφή του με τον χρήστη πολύ εύκολη.[3]

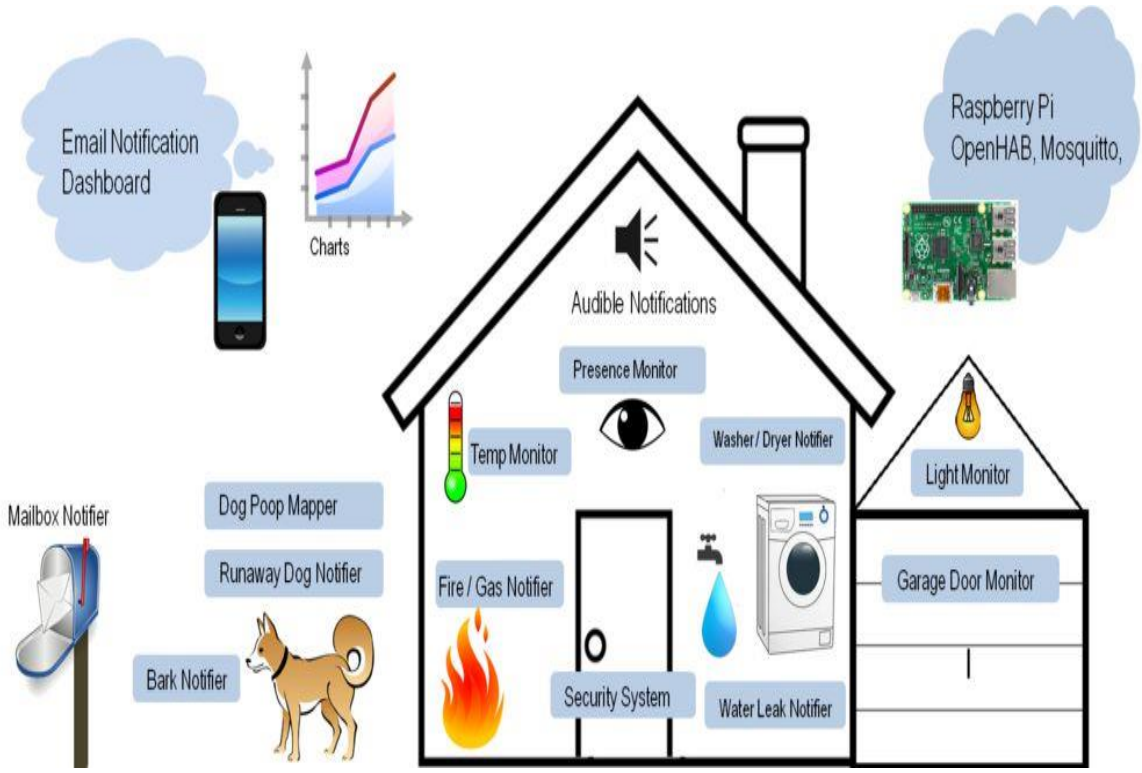


Εικόνα 1.3 OSMC

1.3.3 ΕΞΥΠΝΟ ΣΠΙΤΙ

Μια από τις κορυφαίες καθημερινές χρήσεις του Raspberry είναι ο πλήρης έλεγχος μιας κατοικίας. Μπορούμε να ανοιγοκλείνουμε τα φωτά και ότι άλλο απαιτεί ηλεκτρική ενέργεια, με ηλεκτρονόμους (ρελέ) που οι επαφές του διαχειρίζονται από τους ακροδέκτες του μικροϋπολογιστή μας. Υπάρχει δυνατότητα λήψης ειδοποιήσεων μέσω email και sms σε περίπτωση εσωτερικού κίνδυνου όπως μια πυρκαγιά μέσω των αισθητήρων πυρανίχνευσης ή σε περίπτωσή εξωτερικού κίνδυνου όπως διάρρηξης μέσω των αισθητήρων ανίχνευσης κίνησης. Έχοντας τα παραπάνω σαν είσοδο θα μπορούσαμε να τα αντιμετωπίσουμε μέσω αυτομάτου συστήματος πυρόσβεσης και αντικλεπτικής σειρήνας. Είδαμε ένα πολύ μικρό ποσοστό αυτοματισμών που μπορούμε να δημιουργήσουμε για να έχουμε ένα έξυπνο σπίτι.

Παρόλα αυτά κάθε άνθρωπος, αναλόγως τις ανάγκες του, μπορεί να διαμορφώσει τους αυτοματισμούς της οικίας του, βάση δικών του ιδεών ή άλλων χρηστών που βρίσκονται διαθέσιμες σε ανοιχτές κοινότητες του Raspberry στο διαδίκτυο. [4]



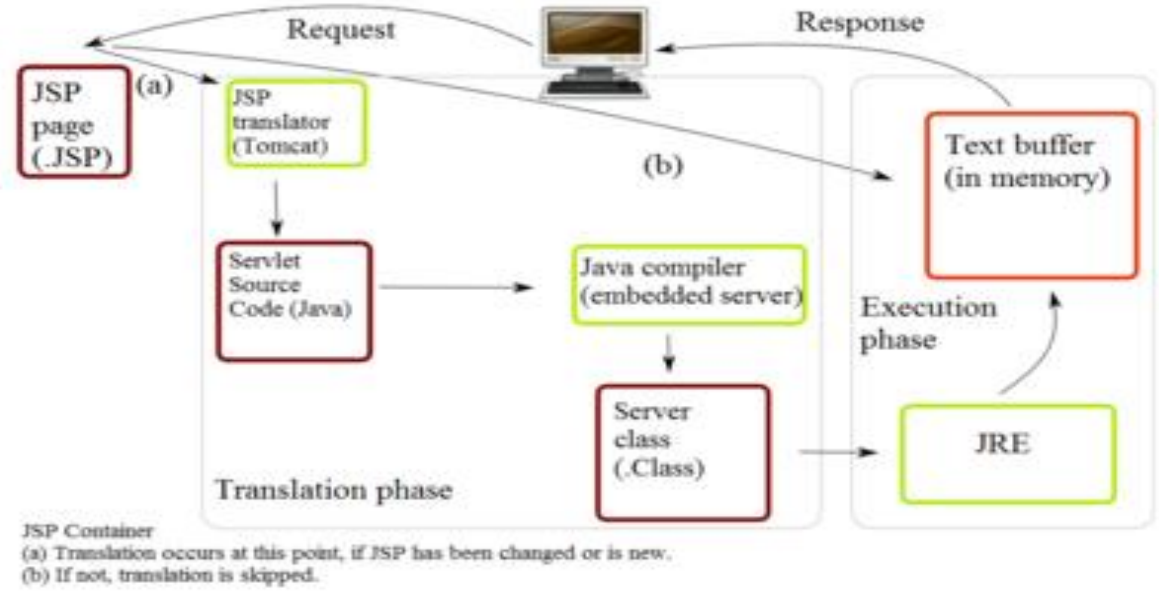
Εικόνα 1.4 (Smart House)

2 ΧΡΗΣΗ ΤΕΧΝΟΛΟΓΙΩΝ ΔΙΑΔΥΚΤΙΟΥ

Υπάρχουν αρκετοί συνδυασμοί του Raspberry Pi 3 με τεχνολογίες προγραμματισμού που μπορούν να χρησιμοποιηθούν για να έχουμε την δημιουργία ενός αυτοματισμού. Στην παρούσα εργασία θα χρησιμοποιήσουμε τις τεχνολογίες των Java Servlets, JavaScript, Html και MySQL τις οποίες διδάχτηκα σαν μαθήματα κατά την φοίτηση μου στο Τ.Ε.Ι Μηχανικών Πληροφορικής Άρτας.

2.1 ΤΙ ΕΙΝΑΙ Η ΤΕΧΝΟΛΟΓΙΑ ΕΦΑΡΜΟΓΩΝ ΔΙΑΔΥΚΤΙΟΥ JAVA SERVLETS

Servlet ονομάζεται μια κλάση της Java που δέχεται αιτήματα τύπου HTTP (Hyper Text Transfer Protocol) και επιστρέφει δυναμικό περιεχόμενο χρησιμοποιώντας όλες τις δυνατότητες που παρέχει η Java.



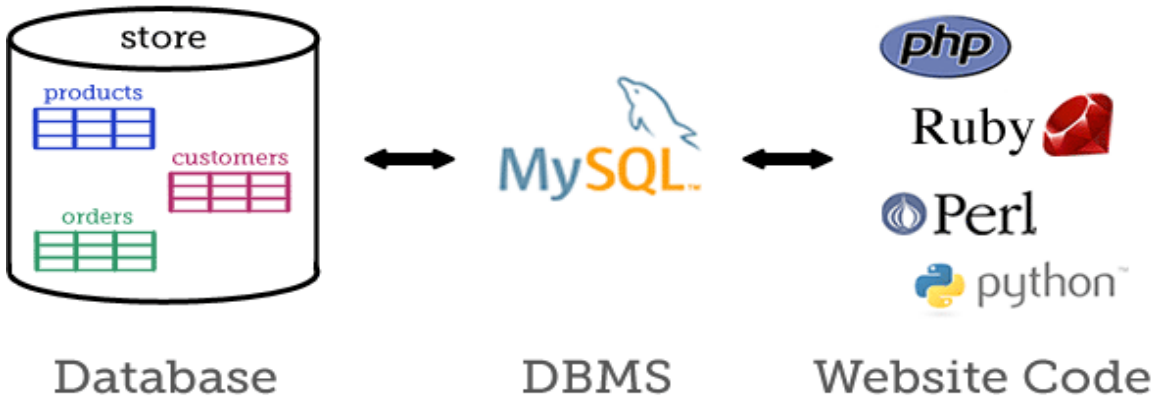
Εικόνα 2.1 Java Servlets

Το πρώτο Servlet δημιουργήθηκε από την εταιρεία Sun Microsystems το 1997, δύο χρόνια μετά την δημοσίευση της Java ως γλώσσα προγραμματισμού ανοικτού κώδικα από την ίδια. Τα Servlets άρχισαν γρήγορα να αντικαθιστούν άλλες τεχνολογίες εφαρμογών διαδικτύου όπως η Common Gateway Interface (CGI), αφού έκαναν μια μόνο σύνδεση με τον εξυπηρετητή μειώνοντας σημαντικά τις συντριβές πακέτων στο διαδίκτυο. Για να προβληθεί

σε έναν web browser είναι προ απαιτούμενη η συνύπαρξη της γλώσσας σήμανσης υπερκείμενου Html μέσα στον κώδικα. Από την μεριά του εξυπηρετητή (server) υπάρχει ένας servlet container ο οποίος επιστρέφει το δυναμικό περιεχόμενο του servlet και εκτελεί τοπικά στον server γραμμές εντολών της Java. Το σημαντικότερο πακέτο που χρησιμοποιείται για την δημιουργία ενός servlet είναι το javax.servlet.http διότι περιέχει τα request και τα response. Δυο είναι οι βασικές μέθοδοι που απαρτίζουν ένα servlet, η doGet και η doPost. Στην doGet μέθοδο τα δεδομένα εισόδου ενός χρήστη ενσωματώνονται στο url που αποστέλλεται στο servlet ενώ στην μέθοδο doPost τα δεδομένα αποστέλλονται ξεχωριστά. Έτσι όπως είναι λογικό από τα παραπάνω στην μέθοδο doGet τα δεδομένα είναι εμφανή με μικρό όγκο ενώ στην doPost το αντίθετο. [5]

2.2 ΤΙ ΕΙΝΑΙ Η MYSQL

MySQL ονομάζεται το πρόγραμμα διαχείρισης καταχωρήσεων που έχουν εισαχθεί σε μια βάση δεδομένων.



Εικόνα 2.2 MySQL

Απαριθμεί περίπου 11 εκατομμύρια χρήστες έως σήμερα και αυτό το κάνει να είναι από τα δημοφιλέστερα. Ανήκει στην εταιρία MySQL AB η οποία είναι θυγατρική της Oracle. Βάση δεδομένων μπορούμε να πούμε πως είναι ένα σύνολο αποθηκευμένων καταχωρήσεων. Η MySQL έχει την δυνατότητα εισαγωγής, διαγράφης, προβολής και ενημέρωσης των καταγράφων που εμπεριέχονται στην παραπάνω. Στην παρούσα εργασία θα αποθηκεύουμε προγραμματισμένα ποτίσματα κ άθολη την διάρκεια της εβδομάδας εισάγοντας σε πίνακες της βάσης δεδομένων την ημέρα, την ώρα έναρξης και την διάρκεια του ποτίσματός.

Πριν την εισαγωγή μιας καταχώρισης στην βάση θα γίνεται προσπέλαση αυτής και θα ελέγχεται αν αυτή υπάρχει ήδη ή εάν συμπίπτει με κάποια ήδη υπάρχουσα. [8][9]

2.3 ΤΙ ΕΙΝΑΙ Η JAVASCRIPT

Η JavaScript είναι μια γλώσσα προγραμματισμού η οποία δεν έχει καμία σχέση με την Java. Βοηθάει στην συμπληρωματική υλοποίηση μιας ιστοσελίδας στον web browser χωρίς όμως να μπορεί να την στηρίξει εξολοκλήρου. Τρέχει τοπικά σε κάθε client χωρίς να έχει επικοινωνία με τον server και συνεργάζεται άμεσα με την Html για την δημιουργία στοιχείων. Μπορεί να κάνει ελέγχους σχετικά με τις εισόδους του χρήστη και να αποστέλλει στον server τις σωστές τιμές.[6] Για να γίνει κατανοητή η λειτουργία της θα αναφέρουμε τον τρόπο λειτουργίας του προγραμματισμένου ποτίσματος στην παρών εργασία. Ο χρήστης μπορεί να επιλέξει την μέρα και τις ώρες ενός προγραμματισμένου ποτίσματος. Για να μην δημιουργήσουμε όλες τις ώρες τις ημέρας από το 0 έως το 23 με στατικό τρόπο χρησιμοποιούμε μια λειτουργία η οποία το κάνει δυναμικά.

```
(function() {
    var elm = document.getElementById('hourTo'),
    df = document.createDocumentFragment();
        for (var i = 0; i <= 23; i++) {
            var option = document.createElement('option');
            option.value = i;
            if(i % 10 == i){
                option.appendChild(document.createTextNode("0"+i+":00")); }
            else {
                option.appendChild(document.createTextNode(i+":00")); }
             df.appendChild(option); }
        lm.appendChild(df); }());
```

Εικόνα 2.3 JavaScript

Ουσιαστικά στο παραπάνω παράδειγμα η λειτουργία μας παίρνει από την φόρμα επιλογής της Html την μεταβλητή hourTo και δημιουργεί αντικείμενα επιλογής από το 0 έως και το 23 που προβάλλονται τοπικά στον χρήστη. Η δυναμική δημιουργία μεταβλητών-αντικειμένων σαφώς και δεν είναι η μοναδική χρήση της καθώς μπορεί να κάνει και έλεγχο για διάφορους παραμέτρους της τιμής, όπως για παράδειγμα το μήκος μιας λέξης.

2.4 ΤΙ ΕΙΝΑΙ Η HTML

HTML (Hyper Text Markup Language) είναι μια γλώσσα δημιουργίας ιστοσελίδων που αφορά την απεικόνιση κειμένου και δομικών στοιχείων σε έναν browser. Η κυριά δομή της οπού αναγνωρίζεται από όλους τους φυλλομετρητές αποτελείται από ζεύγη ετικετών. Τα πιο απαραίτητα για να ξεκινήσει κάποιος την συγγραφή μιας ιστοσελίδας είναι τα παρακάτω.:

- **<html> </html>:** Περιγράφει την έναρξη και λήξη της γλώσσας σήμανσης που εμπεριέχεται σε αυτά τα δυο tags.
- **<head> </head>**: Περιγράφει πληροφορίες σχετικά με το έγγραφο που ακολουθεί όπως λέξεις- κλειδιά, τον συγγραφέα, τον τίτλο κ.α.
- **<title> </title>:** Περιέχει τον τίτλο της ιστοσελίδας.
- **<body> </body>:** Ο κύριος κορμός της ιστοσελίδας βρίσκεται αναμεσά σε αυτά τα δυο tags.

Η Html μονή της δημιουργεί αντικείμενα για προβολή με στατικό τρόπο και αυτό όπως καταλαβαίνουμε δεν είναι λειτουργικό. Έτσι για να επεκτείνουμε τις δυνατότητες της, την χρησιμοποιούμε σε συνεργασία με την JavaScript που αναφέραμε προηγουμένως. [7]

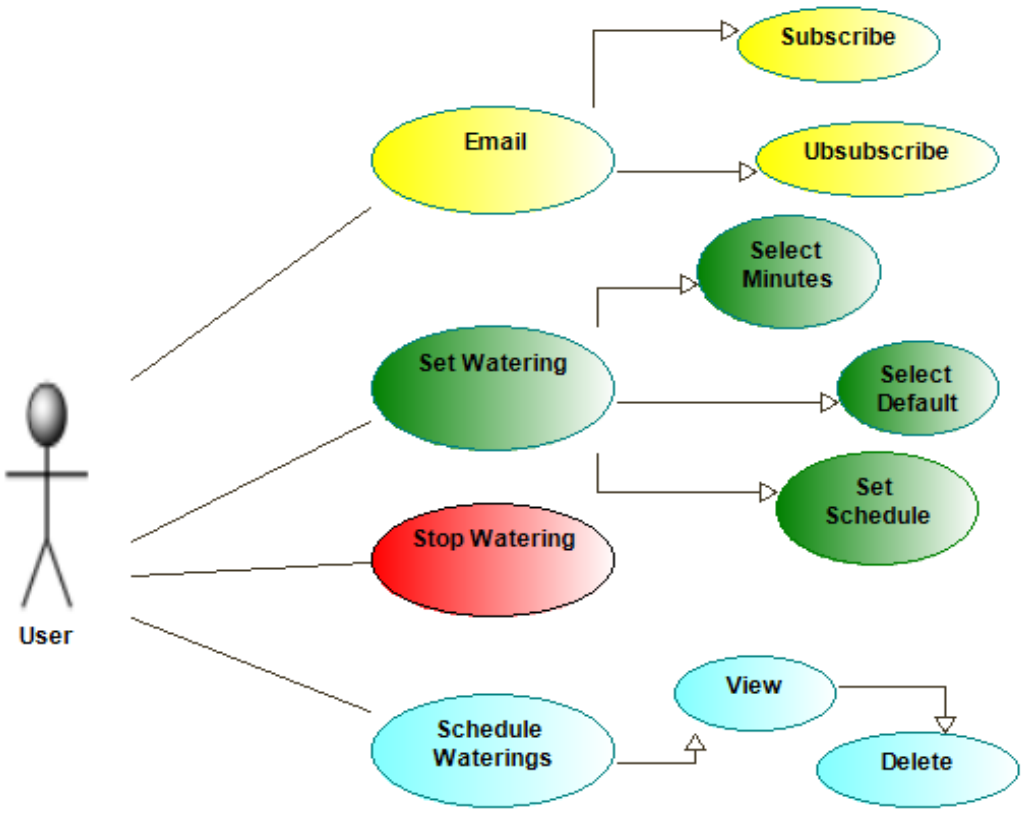
2.5 ΤΙ ΕΙΝΑΙ Ο TOMCAT

Ο Tomcat είναι ένα λογισμικό ανοικτού κώδικα που αναπτύχθηκε από την εταιρία Apache Software Foundation και μας δίνει την δυνατότητα διαχείρισης των Java Servlet, JavaServer Pages και στατικών σελίδων τύπου html. Ουσιαστικά θα μπορούσαμε να πούμε πως είναι ένα πρόγραμμα διαχείρισης ιστοσελίδων. Έπειτα από την εγκατάσταση του μπορούμε να έχουμε πρόσβαση σε αυτόν πληκτρολογώντας την διεύθυνση του μηχανήματος που τον φιλοξενεί στην πόρτα 8080. Επιλέγοντας την σελίδα manager μπορούμε να κάνουμε εγκατάσταση αρχείων war (Web Archive) που χρησιμοποιούνται στην παρούσα πτυχιακή εργασία. Ο τύπος των παραπάνω αρχείων αποθηκεύεται στον υποκατάλογο webapps που εμπεριέχεται στα αρχεία δομής του Tomcat. [11],[12]

3 ΔΙΑΓΡΑΜΜΑΤΑ UML

Η UML (Unified Modeling Language) είναι μια γλώσσα σχεδίασης γραφικής αναπαράστασης κλάσεων και τρόπου λειτουργίας ενός λογισμικού προγράμματος. Απαρτίζεται από όρους, διαγράμματα και σύμβολα τα οποία είναι κοινά για όλες τις περιπτώσεις ανάλυσης. Παρακάτω θα απεικονίσουμε μερικά από τα διαγράμματα βασισμένα στο λογισμικό που έχει αναπτυχθεί για την παρών πτυχιακή εργασία με όνομα Watering. [10]

3.1 ΔΙΑΓΡΑΜΜΑ ΠΕΡΙΠΤΩΣΕΩΝ (USE CASE DIAGRAM)



Εικόνα 3.1 Use Case Diagram

Στο παραπάνω διάγραμμα μπορούμε να δούμε τις 4 βασικές επιλογές χρήσης της διαδικτυακής εφαρμογής που μπορεί να κάνει ο χρήστης.

1. **<u>Λήψη ειδοποιήσεων με email.</u>**

Σε αυτήν την περίπτωση ο χρήστης μπορεί να εισάγει το email του εφόσον αυτό δεν υπάρχει ήδη ή να το διαγράψει από την λίστα της βάσης δεδομένων.

2. <u>Επιλογή ποτίσματος.</u>

Για την εκκίνηση του ποτίσματός ο χρήστης έχει 3 επιλογές. Οι 2 από αυτές αφορούν το παρών. Μπορεί να εισάγει τα λεπτά που θέλει να διαρκέσει το πότισμα είτε να πατήσει ένα από τα κουμπιά που περιέχουν προκαθορισμένο χρόνο ποτίσματός. Στην 3η επιλογή μπορεί να επιλέξει την ημέρα, την ώρα έναρξης και την ώρα λήξης ενός προγραμματισμένου ποτίσματός.

3. <u>Διακοπή ποτίσματος.</u>

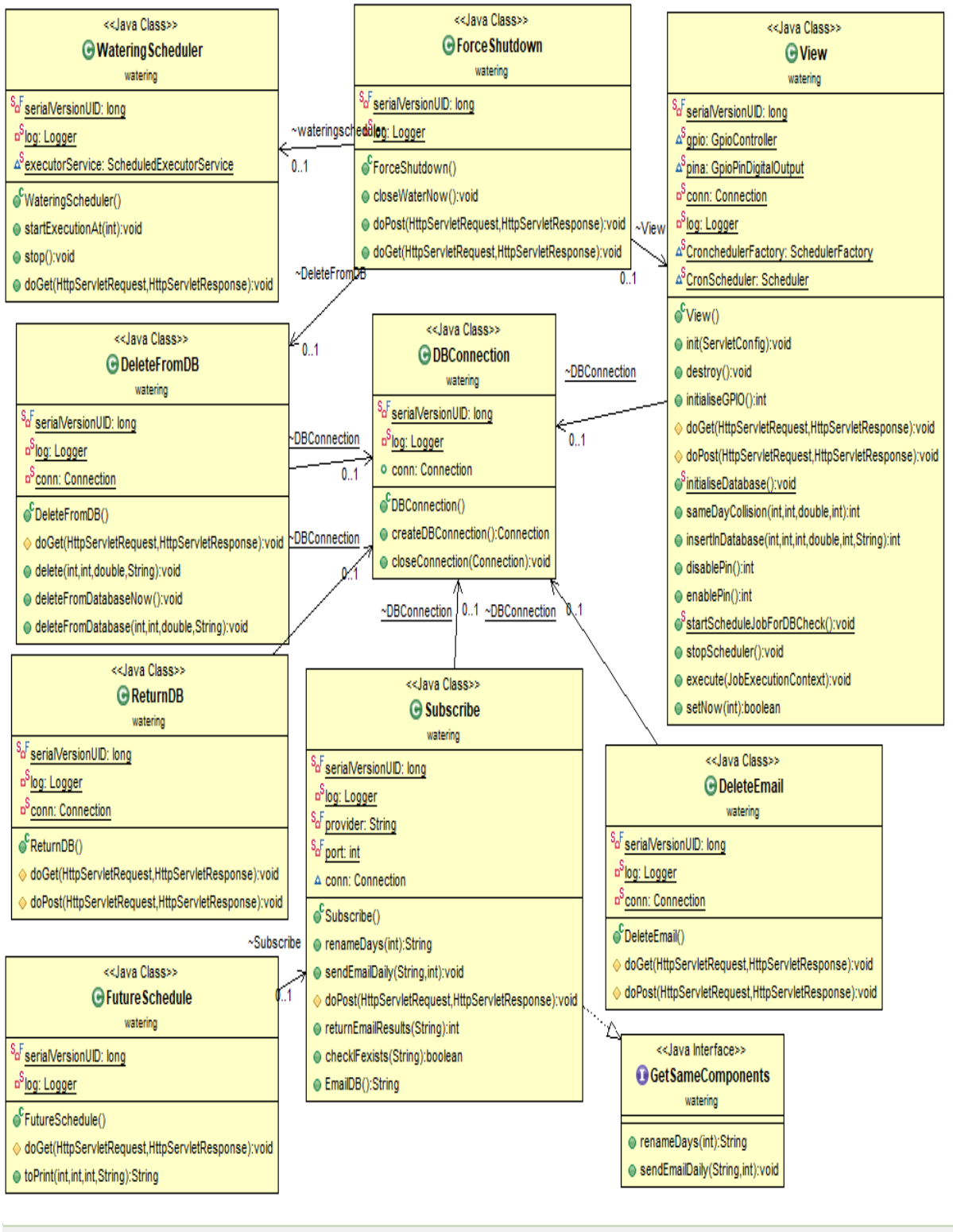
Με την επιλογή της διακοπής ποτίσματος σταματάει αμέσως το πότισμα που βρίσκεται σε εξέλιξη ενώ παράλληλα διαγράφεται από τις καταχωρίσεις του πίνακα της βάσεως δεδομένων.

4. <u>Προβολή προγραμματισμένων ποτισμάτων.</u>

Πατώντας το κουμπί των προγραμματισμένων ποτισμάτων έχει την δυνατότητα προβολής των ποτισμάτων που έχουν εισαχθεί στην βάση δεδομένων (εάν υπάρχουν) και να διαγράψει όποια επιθυμεί.

Είναι σαφές πως ο χρήστης μπορεί να κάνει μια επιλογή κάθε φορά και να λαμβάνει μια απάντηση από τον server. Για παράδειγμα στην περίπτωση που θα γίνει επιλογή ήδη υπάρχοντος ποτίσματος θα λάβει την απάντηση πως υπάρχει σύγκρουση με ενέργεια που έχει καταχωρηθεί ήδη.

3.2 ΔΙΑΓΡΑΜΜΑ ΚΛΑΣΕΩΝ (CLASS DIAGRAM)



Εικόνα 3.2 Class Diagram

Στην παραπάνω εικόνα μπορούμε να δούμε τις κλάσεις που απαρτίζουν την διαδικτυακή εφαρμογή, τις σχέσεις μεταξύ τους και τις μεθόδους από τις οποίες αποτελούνται. Όλες

συνδέονται με την βάση δεδομένων (έμμεσα η άμεσα) προκειμένου να γίνει έλεγχος σύγκρουσης, διαγράφης η εισαγωγής των ποτισμάτων. Η κεντρική κλάση που καλείται όταν ξεκινάμε την διαδικτυακή εφαρμογή είναι η View.class. Στην μέθοδο doGet που περιχέεται στην παραπάνω κλάση, «τραβάμε» την αρχική σελίδα index.html οπού προβάλλεται στον χρήστη. Έπειτα αναλόγως την επιλογή χρήστη, καλείται το αντίστοιχο servlet ή κλάση που συσχετίζονται με αυτήν. Στα παρακάτω υποκεφάλαια θα αναλυθεί ο τρόπος λειτουργίας των μεθόδων που απαρτίζουν τις κλάσεις.

3.2.1 DBCONNECTION.JAVA

```
1. package watering;
2.
3. import java.sql.Connection;
4. import java.sql.SQLException;
5. import javax.servlet.http.HttpServlet;
6. import org.apache.log4j.Logger;
7. import org.apache.tomcat.jdbc.pool.DataSource;
8. import org.apache.tomcat.jdbc.pool.PoolProperties;
9.
10. public class DBConnection extends HttpServlet {
11.     private static final long serialVersionUID = 1L;
12.     private static Logger log = Logger.getLogger(DBConnection.class);
13.     public Connection conn = null;
14.
15.     public Connection createDBConnection() throws SQLException {
16.           String url = "jdbc:mariadb://localhost:3306/";
17.           String dbName = "wateringdb";
18.           String driver = "org.mariadb.jdbc.Driver";
19.           String userName = "root";
20.           String password = "hondaglx125";
21.           PoolProperties p = new PoolProperties();
22.           p.setUrl(url + dbName);
23.           p.setDriverClassName(driver);
24.           p.setPassword(password);
25.           p.setUsername(userName);
26.           p.setMaxActive(100);
27.           DataSource datasource = new DataSource();
28.           datasource.setPoolProperties(p);
29.           conn = datasource.getConnection();
30.           log.info("Connected to the database");
31.           return conn;
32.     }
33.
34.     public void closeConnection(Connection conn) throws SQLException {
35.           conn.close();
36.           log.info("Disconnected from the database");
37.     }
38. }
```

Στον παραπάνω κώδικα της κλάσης DBConnection.java μπορούμε να δούμε πως απαρτίζεται από 2 μεθόδους.

- createDBConnection: Περιέχει εντολές και πληροφορίες σχετικά με την σύνδεση στην τοπική βάση δεδομένων.
- closeConnection: Διακόπτει τη σύνδεση με την τοπική βάση δεδομένων.

Η παραπάνω κλάση έχει πολύ σημαντικό ρολό διότι καλείται σχεδόν από όλες τις υπόλοιπες κλάσεις του προγράμματος και μας επιτρέπει την πρόσβαση σε όλες τις καταχωρίσεις των ποτισμάτων που είναι αποθηκευμένες. Σημαντικό επίσης θα ήταν να πούμε πως η λειτουργία της δεν είναι άμεσα εμφανή καθώς ο χρήστης δεν λαμβάνει κάτι σαν απάντηση στο γραφικό περιβάλλον της διαδικτυακής εφαρμογής.

3.2.2 DELETEEMAIL.JAVA

```
1. package watering;

2. import java.io.IOException;
3. import java.io.PrintWriter;
4. import java.sql.Connection;
5. import java.sql.PreparedStatement;
6. import java.sql.ResultSet;
7. import java.sql.SQLException;
8. import java.sql.Statement;
9. import javax.servlet.ServletException;
10. import javax.servlet.annotation.WebServlet;
11. import javax.servlet.http.HttpServlet;
12. import javax.servlet.http.HttpServletRequest;
13. import javax.servlet.http.HttpServletResponse;
14. import org.apache.log4j.Logger;

15. @WebServlet("/DeleteEmail")
16. public class DeleteEmail extends HttpServlet {
17. private static final long serialVersionUID = 1L;
18. private static Logger log = Logger.getLogger(DeleteEmail.class);
19. private static Connection conn;
20. static DBConnection DBConnection = new DBConnection();

21. protected void doGet(HttpServletRequest request, HttpServletResponse response)
22. throws ServletException, IOException {
23. response.setContentType("text/html");
24. response.setCharacterEncoding("UTF-8");
25. PrintWriter out = response.getWriter();
26. out.println("<html><head>");
27. out.println("<meta name=\"viewport\" content=\"width=device-width, initial-
    scale=1.0\">");
28. out.println("<meta http-equiv=\"Content-Type\" content=\"text/html;charset=UTF-
    8\">");
```

```
29. out.println("<link rel=\"stylesheet\" href=\"graphic.css\"
    type=\"text/css\">");
30. out.println("</head><body>");
31. out.println("<div class=\"responsed\" align=\"center\">");

32. try {
33. conn = DBConnection.createDBConnection();
34. String query = "SELECT * FROM emailT";
35. Statement st = conn.createStatement();
36. ResultSet rs = st.executeQuery(query);
37. if (rs.next() == false) {
38. out.println("<h3>Δεν υπάρχουν εγγεραμμένα email.</h3>");
39. log.info("No emails exist");
40. response.setHeader("Refresh", "4;url=./");
41. } else {
42. out.println("<center><h3>Εγγεγραμμένα Email</h3>");
43. out.println("<table id=\"waterings\" border=\"2\">");
44. out.println("<tr class=\"scheduleTH\">");
45. out.println("<td>Email</td>");
46. out.println("</tr><br><br>");
47. log.info("Listing emails");
48. do {
49. out.println("<tr>");
50. out.println("<td>" + rs.getString("email") + "</td>");
51. out.println("<form  method =\"post\" action=\"DeleteEmail\">");
52. out.println("<input type=\"hidden\" name=\"email\" value=" +
    rs.getString("email") + " />");
53. out.println("<td><button class=\"rmbutton\"
    id=\"stop\"><b>X</b></button></form></td>");
54. out.println("</tr>");
55. } while (rs.next());
56. out.println("</table>");
57. }
58. DBConnection.closeConnection(conn);
59. out.println("<a href=\"./\"><button>Επιστροφή</button></a>");
60. out.println("</div></body></html>");
61. } catch (SQLException e) {
62. e.printStackTrace();
63. }
64. }

65. protected void doPost(HttpServletRequest request, HttpServletResponse response)
66. throws ServletException, IOException {
67. response.setContentType("text/html");
68. response.setCharacterEncoding("UTF-8");
69. PrintWriter out = response.getWriter();
70. String email = request.getParameter("email");
71. out.println("<html><head>");
72. out.println("<meta name=\"viewport\" content=\"width=device-width, initial-
    scale=1.0\">");
73. out.println("<meta http-equiv=\"Content-Type\" content=\"text/html;charset=UTF-
    8\">");
74. out.println("<link rel=\"stylesheet\" href=\"graphic.css\"
    type=\"text/css\">");
75. out.println("</head><body>");
76. out.println("<div class=\"responsed\" align=\"center\">");
77. out.println("</div></body></html>");
78. response.setHeader("Refresh", "0;url=./DeleteEmail?");
79. try {
```

```
80. conn = DBConnection.createDBConnection();
81. log.info("Deleting: "+ email);
82. String query = " DELETE FROM emailT WHERE email = ?";
83. PreparedStatement preparedStmt = conn.prepareStatement(query);
84. preparedStmt.setString(1, email);
85. preparedStmt.execute();
86. DBConnection.closeConnection(conn);
87. } catch (Exception e) {
88. e.printStackTrace();
89. }
90. }
91. }
```

Η παραπάνω κλάση καλείται όταν ο χρήστης θέλει να προβάλει τα email που είναι καταχωρημένα στην βάση δεδομένων με σκοπό να διαγράψει κάποιο από αυτά. Οι 2 μέθοδοι που απαρτίζουν αυτήν την κλάση είναι οι παρακάτω.

- doGet: Προβάλει έπειτα από σύνδεση με την βάση δεδομένων τις καταχωρίσεις του πίνακα emailT (εφόσον υπάρχουν) και μας δίνει την δυνατότητα διαγράφης τους.
- doPost: Εφόσον επιλεγεί κάτι προς διαγραφή η παρούσα μέθοδος διαγραφεί την επιλεγμένη καταχώρηση από τον πίνακα emaiT.

Οι κύριες γραμμές κώδικα στην παραπάνω κλάση είναι αυτές της προβολής και διαγράφης καταχωρίσεων του πίνακα emailT.

3.2.3 DELETEFROMDB.JAVA

```
1. package watering;
2.
3. import java.io.IOException;
4. import java.io.PrintWriter;
5. import java.sql.Connection;
6. import java.sql.PreparedStatement;
7. import java.sql.SQLException;
8. import javax.mail.internet.AddressException;
9. import javax.servlet.ServletException;
10. import javax.servlet.annotation.WebServlet;
11. import javax.servlet.http.HttpServlet;
12. import javax.servlet.http.HttpServletRequest;
13. import javax.servlet.http.HttpServletResponse;
14. import org.apache.log4j.Logger;
15.
16. @WebServlet("/DeleteFromDB")
17. public class DeleteFromDB extends HttpServlet {
18.     private static final long serialVersionUID = 1L;
19.     private static Logger log = Logger.getLogger(DeleteFromDB.class);
20.     private static Connection conn;
```

```
21.    static DBConnection DBConnection = new DBConnection();
22.
23.    protected void doGet(HttpServletRequest request, HttpServletResponse
   response)
24.                  throws ServletException, IOException {
25.           response.setContentType("text/html");
26.           response.setCharacterEncoding("UTF-8");
27.           PrintWriter out = response.getWriter();
28.           String day = request.getParameter("day");
29.           String hourFrom = request.getParameter("hourFrom");
30.           String hourTo = request.getParameter("hourTo");
31.           String status = request.getParameter("status");
32.           out.println("<html><head>");
33.           out.println("<link rel=\"stylesheet\" href=\"graphic.css\"
   type=\"text/css\">");
34.           out.println("<meta name=\"viewport\" content=\"width=device-width,
   initial-scale=1.0\">");
35.           out.println("<meta http-equiv=\"Content-Type\"
   content=\"text/html;charset=UTF-8\">");
36.           out.println("</head><body>");
37.           out.println("<div class=\"responsed\" align=\"center\">");
38.           out.println("</div></body></html>");
39.           response.setHeader("Refresh", "0;url=./ReturnDB?");
40.
41.           try {
42.                  delete(Integer.parseInt(day), Integer.parseInt(hourFrom),
   Double.parseDouble(hourTo), status);
43.           } catch (NumberFormatException | SQLException | AddressException e) {
44.                  e.printStackTrace();
45.           }
46.    }
47.    public void delete(int day, int hourFrom, double hourTo, String status)
48.                  throws SQLException, AddressException {
49.           if (status.equals("on")) {
50.                  ForceShutdown ForceShutdown = new ForceShutdown();
51.                  ForceShutdown.closeWaterNow();
52.           } else if (status.equals("off")) {
53.                  deleteFromDatabase(day, hourFrom, hourTo, status);
54.           }
55.    }
56.
57.    public void deleteFromDatabaseNow() throws SQLException {
58.           conn = DBConnection.createDBConnection();
59.           Log.info("Watering With Status ON deleted");
60.           String status = "on";
61.           PreparedStatement st = conn.prepareStatement("DELETE FROM
   wateringtable WHERE Status = ?");
62.           st.setString(1, status);
63.           st.executeUpdate();
64.           DBConnection.closeConnection(conn);
65.    }
66.
67.    public void deleteFromDatabase(int day, int hourFrom, double hourTo, String
   status) throws SQLException {
68.           conn = DBConnection.createDBConnection();
69.           Log.info("Deleting Insertion: Day: " + day + " hourFrom: " + hourFrom
   + " hourTo: " + hourTo + " Status: " + status);
70.           PreparedStatement st = conn
```

```
71.                         .prepareStatement("DELETE FROM wateringtable WHERE Day =
   ? AND hourFrom = ? AND hourTo = ?  AND Status = ?");
72.           st.setInt(1, day);
73.           st.setInt(2, hourFrom);
74.           st.setDouble(3, hourTo);
75.           st.setString(4, status);
76.           st.executeUpdate();
77.           DBConnection.closeConnection(conn);
78.    }
79. }
```

Οι μέθοδοι που βρίσκονται στην παραπάνω κλάση αφορούν την ενέργεια διαγράφης μιας καταχώρησης που βρίσκεται στον πίνακα προγραμματισμένων ποτισμάτων (wateringtable). Γίνεται έλεγχος εάν η καταχώριση που θέλουμε να διαγράψουμε είναι ένα ενεργό ή ένα ανενεργό πότισμα. Παρακάτω αναφέρεται συνοπτικά η λειτουργία των μεθόδων της κλάσης αυτής.

- doGet: Δέχεται τα δεδομένα της καταχώρισης που επιλέγουμε να διαγράψουμε και τα αποστέλλει στην μέθοδο delete.
- delete: Γίνεται έλεγχος της κατάστασης στην οποία βρίσκεται το πότισμα. Εάν αυτό είναι ενεργό θα διαγραφεί από τις καταχωρίσεις ενώ παράλληλα θα διακοπεί. Εάν είναι ανενεργό καλεί την μέθοδο deleteFromDatabase.
- deleteFromDatabaseNow: Διαγραφεί από τον πίνακα τις καταχωρίσεις με status “on”
- deleteFromDatabase: Διαγραφεί από τον πίνακα την καταχώρηση με status “off” και τα σχετιζόμενα δεδομένά που λαμβάνει από την μέθοδο delete.

3.2.4 FORCESHUTDOWN.JAVA

```
1. package watering;

2. import java.io.IOException;
3. import java.io.PrintWriter;
4. import java.sql.SQLException;
5. import javax.servlet.annotation.WebServlet;
6. import javax.servlet.http.HttpServlet;
7. import javax.servlet.http.HttpServletRequest;
8. import javax.servlet.http.HttpServletResponse;
9. import org.apache.log4j.Logger;

10. @WebServlet("/ForceShutdown")
11. public class ForceShutdown extends HttpServlet {

12. private static final long serialVersionUID = 1L;
13. private static Logger log = Logger.getLogger(ForceShutdown.class);
14. View View = new View();
```

```
15. WateringScheduler wateringscheduler = new WateringScheduler();
16. DeleteFromDB  DeleteFromDB= new DeleteFromDB ();

17. public void closeWaterNow() throws SQLException {
18. wateringscheduler.stop();
19. View.disablePin();
20. log.info("Pin Closed");
21. DeleteFromDB.deleteFromDatabaseNow();
22. }

23. public void doPost(HttpServletRequest request, HttpServletResponse
    response) throws IOException {

24. try {
25. closeWaterNow();
26. } catch (SQLException e) {
27. e.printStackTrace();
28. }
29. response.setContentType("text/html");
30. response.setCharacterEncoding("UTF-8");
31. PrintWriter out = response.getWriter();
32. out.println("<html><head>");
33. out.println("<meta name=\"viewport\" content=\"width=device-width, initial-
    scale=1.0\">");
34. out.println("<meta http-equiv=\"Content-Type\"
    content=\"text/html;charset=UTF-8\">");
35. out.println("<link rel=\"stylesheet\" href=\"graphic.css\"
    type=\"text/css\">");
36. out.println("</head><body>");
37. out.println("<div class=\"responsed\" align=\"center\">");
38. out.println("<h3>Το πότισμα σταμάτησε.</h3>");
39. out.println("<h3>Θα μεταβείτε στην αρχική σελίδα αυτόματα.</h3>");
40. out.println("</div></body></html>");
41. response.setHeader("Refresh", "3;url=./");
42. }

43. public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException {
44. doPost(request, response);
45. }
46. }
```

Η παραπάνω κλάση καλείται για να διακόψει την διεξαγωγή του τρέχων ποτίσματος (εάν αυτό υπάρχει) ενώ παράλληλα διαγράφει την καταχώρηση με status “on” από τον πίνακα της βάσης δεδομένων wateringtable. Παρακάτω βλέπουμε τις 2 βασικές μεθόδους αυτής της κλάσης.

- closeWaterNow: Τερματίζει την λειτουργία του νήματος, καλεί την μέθοδο View.disablePin και διαγραφεί την καταχώρηση με status “on”.

- doPost: Καλεί την παραπάνω μέθοδο και μας εμφανίζει μήνυμα επιβεβαίωσης τερματισμού. Έπειτα από 3 δευτερόλεπτα μας μεταφέρει αυτόματα στην προηγουμένη σελίδα (αρχική σελίδα).

3.2.5 FUTURESCHEDULE.JAVA

```
1. package watering;

2. import java.io.IOException;
3. import java.io.PrintWriter;
4. import java.util.Calendar;

5. import javax.servlet.ServletException;
6. import javax.servlet.annotation.WebServlet;
7. import javax.servlet.http.HttpServlet;
8. import javax.servlet.http.HttpServletRequest;
9. import javax.servlet.http.HttpServletResponse;

10. //import org.apache.log4j.Logger;

11. @WebServlet("/FutureSchedule")
12. public class FutureSchedule extends HttpServlet {
13. private static final long serialVersionUID = 1L;
14. // private static Logger log = Logger.getLogger(FutureSchedule.class);
15. Subscribe Subscribe = new Subscribe();

16. protected void doGet(HttpServletRequest request, HttpServletResponse response)
17. throws ServletException, IOException {
18. response.setContentType("text/html");
19. response.setCharacterEncoding("UTF-8");
20. PrintWriter out = response.getWriter();
21. String hourFrom = request.getParameter("hourFrom");
22. String day = request.getParameter("day");
23. String duration = request.getParameter("duration");
24. String hourTo = request.getParameter("hourTo");
25. String status = request.getParameter("status");

26. out.println("<html><head>");
27. out.println("<link rel=\"stylesheet\" href=\"graphic.css\"
    type=\"text/css\">");
28. out.println("<meta name=\"viewport\" content=\"width=device-width, initial-
    scale=1.0\">");
29. out.println("<meta http-equiv=\"Content-Type\" content=\"text/html;charset=UTF-
    8\">");
30. out.println("</head>");
31. out.println("<body>");
32. out.println("<div class=\"responsed\" align=\"center\">");
33. out.println("<p align=\"CENTER\">"
34. + toPrint(Integer.parseInt(hourFrom), Integer.parseInt(duration),
    Integer.parseInt(day), status));
35. out.println("<a href=\"./\"><button>Επιστροφή</button></a><br>");
36. out.println("<a href=\"DeleteFromDB?hourFrom=" + hourFrom + "&day=" + day +
    "&hourTo=" + hourTo + "&status="
37. + status + "\"><button id=\"stop\">Ακύρωση</button></a><br><br><br>");
```

```
38. out.println("<a href=\"ReturnDB\"\"><button>Προγγραματισμένα
    ποτίσματα</button></a> ");
39. out.println("</div></body></html>");
40. response.setHeader("Refresh", "5;url=./");
41. }

42. public String toPrint(int hourFrom, int duration, int day, String status) {
43. Calendar c = Calendar.getInstance();

44. if (c.get(Calendar.DAY_OF_WEEK) == day && c.get(Calendar.HOUR_OF_DAY) <=
    hourFrom) {
45. if (hourFrom % 10 == hourFrom) {
46. if (duration == 1) {
47. return ("<h2>Σήμερα!\n" + " Ωρα έναρξης: " + hourFrom + ":00. Διάρκεια
    ποτίσματος : " + duration
48. + " λεπτό.</h2>");
49. } else {
50. return ("<h2>Σήμερα!\n" + " Ωρα έναρξης: " + hourFrom + ":00. Διάρκεια
    ποτίσματος : " + duration
51. + " λεπτά.</h2>");
52. }
53. } else {
54. if (duration == 1) {
55. return ("<h2>Σήμερα!\n" + " Ωρα έναρξης: " + hourFrom + ":00. Διάρκεια
    ποτίσματος : " + duration
56. + " λεπτό.</h2>");
57. } else {
58. return ("<h2>Σήμερα!\n" + " Ωρα έναρξης: " + hourFrom + ":00. Διάρκεια
    ποτίσματος : " + duration
59. + " λεπτά.</h2>");
60. }
61. }
62. } else {
63. if ((hourFrom % 10) == hourFrom) {
64. if (duration == 1) {
65. return ("<h2>Το πότισμα θα ξεκινήσει τη μέρα " + Subscribe.renameDays(day) + "
    και ώρα 0" + hourFrom
66. + ":00. Διάρκεια ποτίσματος : " + duration + " λεπτό.</h2>");
67. } else {
68. return ("<h2>Το πότισμα θα ξεκινήσει τη μέρα " + Subscribe.renameDays(day) + "
    και ώρα 0" + hourFrom
69. + ":00. Διάρκεια ποτίσματος : " + duration + " λεπτά. </h2>");
70. }
71. } else {
72. if (duration == 1) {
73. return ("<h2>Το πότισμα θα ξεκινήσει τη μέρα " + Subscribe.renameDays(day) + "
    και ώρα " + hourFrom
74. + ":00. Διάρκεια ποτίσματος : " + duration + " λεπτό.</h2>");
75. } else {
76. return ("<h2>Το πότισμα θα ξεκινήσει τη μέρα " + Subscribe.renameDays(day) + "
    και ώρα " + hourFrom
77. + ":00. Διάρκεια ποτίσματος : " + duration + " λεπτά.</h2>");
78. }
79. }
80. }
81. }
82. }
```

Η παραπάνω κλάση μας βοηθάει στην επιβεβαίωση ενός προγραμματισμένου ποτίσματος. Ουσιαστικά μας διαμορφώνει τα δεδομένα που εισάγει ο χρήστης έτσι ώστε να είναι ευανάγνωστα και κατανοητά. Αποτελείται από τις 2 παρακάτω μεθόδους.

- doGet: Απορροφάει από την διεύθυνση τα δεδομένα εισαγωγής του χρήστη και με την χρήση της μεθόδου toPrint τα προβάλει.
- toPrint: Προβάλει τα περιεχόμενα των μεταβλητών hourFrom και duration, ενώ με την χρήση της μεθόδου renameDays(day) μεταφράζει τον αριθμό της ημέρας σε λογική έξοδο για τον χρήστη.

3.2.6 GETSAMECOMPONENTS.JAVA

```
1. package watering;
2. import javax.mail.internet.AddressException;
3. public interface GetSameComponents {
4. public String renameDays(int i);
5. public void sendEmailDaily(String text, int flag) throws AddressException;
6. }
```

Κλάση που είναι υπεύθυνη για την αποστολή των email. Υλοποιείται κατά την λειτουργία της κλάσης Subscribe και προσφέρει διασύνδεση των μεθόδων renameDays και sendEmailDaily που θα αναφέρουμε παρακάτω.

3.2.7 RETURNDB.JAVA

```
1. package watering;
2. import java.io.IOException;
3. import java.io.PrintWriter;
4. import java.sql.Connection;
5. import java.sql.ResultSet;
6. import java.sql.SQLException;
7. import java.sql.Statement;
8. import java.text.SimpleDateFormat;
9. import java.util.Calendar;
10. import java.util.Date;
11. import javax.servlet.ServletException;
12. import javax.servlet.annotation.WebServlet;
13. import javax.servlet.http.HttpServlet;
14. import javax.servlet.http.HttpServletRequest;
```

```
15. import javax.servlet.http.HttpServletResponse;
16. import org.apache.log4j.Logger;

17. @WebServlet("/ReturnDB")
18. public class ReturnDB extends HttpServlet {
19. private static final long serialVersionUID = 1L;
20. private static Logger log = Logger.getLogger(ReturnDB.class);
21. private static Connection conn = null;
22. static DBConnection DBConnection = new DBConnection();

23. protected void doGet(HttpServletRequest request, HttpServletResponse
    response)
24. throws ServletException, IOException {
25. response.setContentType("text/html");
26. response.setCharacterEncoding("UTF-8");
27. PrintWriter out = response.getWriter();
28. out.println("<html><head>");
29. out.println("<meta name=\"viewport\" content=\"width=device-width, initial-
    scale=1.0\">");
30. out.println("<meta http-equiv=\"Content-Type\"
    content=\"text/html;charset=UTF-8\">");
31. out.println("<link rel=\"stylesheet\" href=\"graphic.css\"
    type=\"text/css\">");
32. out.println("</head><body>");
33. out.println("<div class=\"responsed\" align=\"center\">");

34. try {
35. conn = DBConnection.createDBConnection();
36. log.info("Get watering insesrtions");
37. String query = "SELECT * FROM wateringtable";
38. Statement st = conn.createStatement();
39. ResultSet rs = st.executeQuery(query);
40. Subscribe sb = new Subscribe();
41. if (rs.next() == false) {
42. out.println("<center><h3>Δεν υπάρχουν προγραμματισμένα ποτίσματα.</h3>");
43. response.setHeader("Refresh", "3;url=./");
44. } else {
45. out.println("<center><h3>Προγραμματισμένα Ποτίσματα</h3>");
46. out.println(
47. "<iframe
    src=\"http://free.timeanddate.com/clock/i6igvaxw/n26/tlgr17/fn6/fs16/fcfff/
    tct/pct/ftb/tt0/tm1/td1/th1/tb4\" frameborder=\"0\" width=\"199\"
    height=\"42\" ></iframe>");
48. out.println("<table id=\"waterings\" border=\"2\">");
49. out.println("<tr class=\"scheduleTH\">");
50. out.println("<td>Ημέρα</td>");
51. out.println("<td>Έναρξη</td>");
52. out.println("<td>Διάρκεια (Λεπτά)</td>");
53. out.println("<td>Κατάσταση</td>");
54. out.println("</tr><br><br>");
55. do {
56. Calendar c = Calendar.getInstance();
57. Calendar cal = Calendar.getInstance();
58. String pattern = "HH:mm";
59. c.set(Calendar.HOUR_OF_DAY, rs.getInt("hourFrom"));
60. c.set(Calendar.MINUTE, rs.getInt("hourFromMin"));
61. Date d = c.getTime();
62. SimpleDateFormat simpleDateFormat = new SimpleDateFormat(pattern);
63. String output = simpleDateFormat.format(d);
```

```
64. String color = null;
65. if (rs.getString("status").equals("on")) {
66. color = "#DAA520";// Color gold
67. }
68. out.println("<tr style=\"color: " + color + ";\">");
69. out.println("<td>");
70. if ((rs.getInt("day") == cal.get(Calendar.DAY_OF_WEEK)
71. && cal.get(Calendar.HOUR_OF_DAY) <= rs.getInt("hourFrom"))
72. || rs.getString("status").equals("on")) {
73. out.println("Σήμερα");
74. } else {
75. out.println(sb.renameDays(rs.getInt("day")));
76. out.println("</td>");
77. }
78. out.println("<td>" + output + "</td>");
79. out.println("<td>" + rs.getInt("Duration") + "</td>");
80. out.println("<td>");
81. if (rs.getString("status").equals("on")) {
82. out.println("Ενεργό");
83. } else {
84. out.println("Ανενεργό");
85. }
86. out.println("</td>");
87. out.println("<form  method =\"GET\" action=\"DeleteFromDB\">");
88. out.println("<input type=\"hidden\" name=\"hourFrom\" value=" +
    rs.getInt("hourFrom") + " />");
89. out.println("<input type=\"hidden\" name=\"day\" value=" + rs.getInt("day")
    + " />");
90. out.println("<input type=\"hidden\" name=\"hourTo\" value=" +
    rs.getDouble("hourTo") + " />");
91. out.println("<input type=\"hidden\" name=\"status\" value=" +
    rs.getString("status") + " />");
92. out.println("<td><button class=\"rmbutton\"
    id=\"stop\"><b>X</b></button></form></td>");
93. out.println("</tr>");
94. } while (rs.next());
95. out.println("</table>");
96. }
97. DBConnection.closeConnection(conn);
98. out.println("<a href=\"./\"><button>Επιστροφή</button></a>");
99. out.println("</div></body></html>");
100.      } catch (SQLException e) {
101.      e.printStackTrace();
102.      }
103.      }

104.      protected void doPost(HttpServletRequest request, HttpServletResponse
    response)
105.      throws ServletException, IOException {
106.      doGet(request, response);
107.      }
108.      }
```

Αυτή η κλάση καλείται όταν ο χρήστης θέλει να προβάλει τα ποτίσματα που είναι καταχωρημένα στην βάση δεδομένων. Αποτελείται από 2 μεθόδους.

- doGet: Προβάλει σε μορφή πίνακα όλα τα ποτίσματα που είναι καταχωρημένα στον πίνακα wateringtable της βάσης δεδομένων (wateringdb) ενώ παράλληλα μας δίνει την δυνατότητα διαγράφης τους.
- doPost: Καλεί την μέθοδο doGet.

3.2.8 SUBSCRIBE.JAVA

```
1. package watering;

2. import java.io.IOException;
3. import java.io.PrintWriter;
4. import java.sql.Connection;
5. import java.sql.PreparedStatement;
6. import java.sql.ResultSet;
7. import java.sql.SQLException;
8. import java.sql.Statement;
9. import java.util.Calendar;
10. import java.util.Properties;
11. import javax.mail.Authenticator;
12. import javax.mail.MessagingException;
13. import javax.mail.PasswordAuthentication;
14. import javax.mail.Session;
15. import javax.mail.Transport;
16. import javax.mail.internet.AddressException;
17. import javax.mail.internet.InternetAddress;
18. import javax.mail.internet.MimeMessage;
19. import javax.servlet.ServletException;
20. import javax.servlet.annotation.WebServlet;
21. import javax.servlet.http.HttpServlet;
22. import javax.servlet.http.HttpServletRequest;
23. import javax.servlet.http.HttpServletResponse;
24. import org.apache.commons.validator.routines.EmailValidator;
25. import org.apache.log4j.Logger;

26. @WebServlet("/Subscribe")
27. public class Subscribe extends HttpServlet implements GetSameComponents {
28. private static final long serialVersionUID = 1L;
29. private static Logger log = Logger.getLogger(Subscribe.class);
30. private static final String provider = "smtp.office365.com";
31. private static final int port = 587;
32. Connection conn;
33. static DBConnection DBConnection = new DBConnection();

34. public String renameDays(int i) {
35. String dayName = null;
36. switch (i) {
37. case 1:
38. dayName = "Κυριακή";
39. break;
40. case 2:
41. dayName = "Δευτέρα";
42. break;
43. case 3:
44. dayName = "Τρίτη";
```

```
45. break;
46. case 4:
47. dayName = "Τετάρτη";
48. break;
49. case 5:
50. dayName = "Πέμπτη";
51. break;
52. case 6:
53. dayName = "Παρασκευή";
54. break;
55. case 7:
56. dayName = "Σάββατο";
57. break;
58. }
59. return dayName;
60. }

61. public void sendEmailDaily(String text, int flag) throws AddressException {
62. String to;
63. String fin = "\n\n\nΑυτό είναι ένα αυτόματο μήνυμα. Παρακαλώ μην απαντήσετε σε
    αυτό το μήνυμα ηλεκτρονικού ταχυδρομείου.";

64. try {
65. to = EmailDB();
66. log.info("Email clients : " + to + to.length());
67. if (to.length() > 0) {
68. InternetAddress[] parse = InternetAddress.parse(to, true);
69. String password = "******";
70. String from = "wateringbot@outlook.com";
71. Properties props = System.getProperties();
72. props.put("mail.smtp.starttls.enable", "true");
73. props.put("mail.smtp.auth", "true");
74. props.put("mail.smtp.host", provider);
75. props.put("mail.smtp.port", port);

76. Session session = Session.getDefaultInstance(props, new Authenticator() {
77. protected PasswordAuthentication getPasswordAuthentication() {
78. return new PasswordAuthentication(from, password);
79. }
80. });
81. MimeMessage message = new MimeMessage(session);
82. message.setFrom(new InternetAddress(from));

83. switch (flag) {
84. case 1:
85. message.setRecipients(javax.mail.Message.RecipientType.TO, parse);
86. message.setSubject("[Watering Bot] Ένα πότισμα είναι έτοιμο να ξεκινήσει.",
    "UTF-8");
87. message.setText(text + fin, "UTF-8");
88. break;
89. case 2:
90. message.setRecipients(javax.mail.Message.RecipientType.TO, parse);
91. message.setSubject("[Watering Bot] Ένα πότισμα ολοκληρώθηκε με επιτυχία.",
    "UTF-8");
92. message.setText(text + fin, "UTF-8");
93. break;
94. case 3:
95. message.setRecipients(javax.mail.Message.RecipientType.TO, parse);
96. message.setSubject("[Watering Bot] Ένα πότισμα διακόπηκε.", "UTF-8");
```

```
97. message.setText(text + fin, "UTF-8");
98. break;
99. case 4:
100.   message.setSubject("[Watering Bot] Επιτυχής Εγγραφή", "UTF-8");
101.   message.setText(
102.   "Εγγραφήκατε με επιτυχία στη λίστα μας.\n Μπορείτε να καταργήσετε την
   εγγραφή σας πατώντας το κουμπί κατάργηση εγγραφής στο site."
103.   + fin,
104.   "UTF-8");
105.   message.setRecipients(javax.mail.Message.RecipientType.TO, text);
106.   break;
107.   }
108.   Transport.send(message);
109.   log.info("Sent message successfully....");
110.   } else {
111.   log.info("No recipients");
112.   }
113.   } catch (InstantiationException | IllegalAccessException |
   ClassNotFoundException | SQLException
114.   | MessagingException e) {
115.   e.printStackTrace();
116.   }
117.   }

118.   protected void doPost(HttpServletRequest request, HttpServletResponse
   response)
119.   throws ServletException, IOException {
120.   String email = request.getParameter("email");
121.   response.setContentType("text/html");
122.   response.setCharacterEncoding("UTF-8");
123.   PrintWriter out = response.getWriter();
124.   int checkEx = returnEmailResults(email);
125.   out.println("<meta name=\"viewport\" content=\"width=device-width, initial-
   scale=1.0\">");
126.   out.println("<meta http-equiv=\"Content-Type\"
   content=\"text/html;charset=UTF-8\">");
127.   out.println("<link rel=\"stylesheet\" href=\"graphic.css\"
   type=\"text/css\">");
128.   out.println("</head><body>");
129.   out.println("<div class=\"responsed\" align=\"center\">");

130.   if (checkEx == 1) {
131.   out.println("<h3>Το email υπάρχει ήδη.</h3>");
132.   } else if (checkEx == 2) {
133.   try {
134.   sendEmailDaily(email, 4);
135.   out.println("<h3>Η εγγραφή σας δημιουργήθηκε με επιτυχία. Θα λαμβάνεται
   ενημερώσεις.</h3>");
136.   } catch (AddressException e) {
137.   e.printStackTrace();
138.   }
139.   } else {
140.   out.println("<h3>Μη έγκυρη διεύθυνση ηλεκτρονικού ταχυδρομείου.Παρακαλώ
   προσπαθήστε ξανά.</h3>");
141.   }
142.   out.println("<h3>Θα μεταβείτε στην αρχική σελίδα αυτόματα.</h3>");
143.   out.println("<a href=\"./\"><button>Επιστροφή</button></a>");
144.   out.println("</div></body></html>");
145.   response.setHeader("Refresh", "5;url=./");
```

```
146.   }

147.   public int returnEmailResults(String email) {

148.   try {
149.   conn = DBConnection.createDBConnection();
150.   boolean check = checkIFexists(email);
151.   boolean valid = EmailValidator.getInstance().isValid(email);

152.   Calendar calendar = Calendar.getInstance();
153.   java.sql.Date startDate = new java.sql.Date(calendar.getTime().getTime());

154.   if (valid) {
155.   if (!check) {
156.   String query = " insert into emailT (email,Date)" + " values (?, ?)";
157.   PreparedStatement preparedStmt = conn.prepareStatement(query);
158.   preparedStmt.setString(1, email);
159.   preparedStmt.setDate(2, startDate);
160.   preparedStmt.execute();
161.   log.error("Email inserted in database");
162.   DBConnection.closeConnection(conn);
163.   return 2;
164.   } else {

165.   return 1;
166.   }
167.   } else {
168.   log.error("Invalid email address");
169.   DBConnection.closeConnection(conn);
170.   return 3;
171.   }
172.   } catch (Exception e) {
173.   e.printStackTrace();
174.   }
175.   return 0;
176.   }

177.   public boolean checkIFexists(String email)
178.   throws InstantiationException, IllegalAccessException,
   ClassNotFoundException, SQLException {

179.   String query = "SELECT * FROM emailT";
180.   Statement st = conn.createStatement();
181.   ResultSet rs = st.executeQuery(query);

182.   while (rs.next()) {
183.   if (email.equals(rs.getString("email"))) {
184.   log.info("Email already exists");
185.   DBConnection.closeConnection(conn);
186.   return true;
187.   }
188.   }

189.   return false;
190.   }

191.   public String EmailDB()
192.   throws InstantiationException, IllegalAccessException,
   ClassNotFoundException, SQLException {
```

```
193.    conn = DBConnection.createDBConnection();
194.    log.info("Email list entered");
195.    String query = "SELECT * FROM emailT";
196.    Statement st = conn.createStatement();
197.    String result = "";
198.    ResultSet rs = st.executeQuery(query);

199.    while (rs.next()) {
200.    String email = rs.getString("email");
201.    result += email + ", ";
202.    }
203.    DBConnection.closeConnection(conn);
204.    return result;
205.    }
206.    }
```

Η παραπάνω κλάση περιέχει μεθόδους για την μετάφραση της ημέρας που εισάγει ο χρήστης, τον έλεγχο και την ορθότητα του email καθώς επίσης και τις πληροφορίες του αποστολέα και του παραλήπτη των μηνυμάτων ηλεκτρονικού ταχυδρομείου. Οι 6 μέθοδοι της κλάσης αυτής είναι οι εξής.

- renameDays: Ελέγχει τις περιπτώσεις εισαγωγής ημέρας από την αρχική σελίδα της εφαρμογής και μας επιστρέφει το ανάλογο αποτέλεσμα σε μορφή χαρακτήρων που αποθηκεύεται στην μεταβλητή dayName.
- sendEmailDaily: Περιέχει πληροφορίες του αποστολέα των μηνυμάτων, κάνει προσπέλαση του πίνακα που περιέχονται τα εγγεγραμμένα email και αποστέλλει στους παραλήπτες που είναι καταχωρημένοι στον πίνακα emailT το μήνυμα που θα ορίσουμε κατά την υλοποίηση της.
- doPost: Παρέχει πληροφορίες σχετικά με την εισαγωγή email που εμφανίζονται σαν απάντηση στον χρήστη, κάνοντας ελέγχους σε συνεργασία με τις παρακάτω μεθόδους.
- returnEmailResults: Γίνεται έλεγχος της ορθότητας του email και αποθήκευση στον πίνακα καταχωρήσεων emaiT εφόσον δεν υπάρχει ήδη.
- checkIFexists: Κάνει προσπέλαση του πίνακα των email για να ελεγχθεί εάν υπάρχει ήδη.
- EmailDB: Κάνει προσπέλαση όλων των καταχωρημένων email του πίνακα και τα αποθηκεύει στην μεταβλητή result. Χρησιμοποιείται κυρίως από την παραπάνω μέθοδο sendEmailDaily για την αποστολή μηνυμάτων σε όλους τους παραλήπτες.

3.2.9 VIEW.JAVA

```
1.  package watering;
2.
3.  import java.io.IOException;
4.  import java.io.PrintWriter;
5.  import java.sql.Connection;
6.  import java.sql.PreparedStatement;
7.  import java.sql.ResultSet;
8.  import java.sql.SQLException;
9.  import java.sql.Statement;
10. import java.util.Calendar;
11. import java.util.concurrent.TimeUnit;
12. import javax.mail.internet.AddressException;
13. import javax.servlet.RequestDispatcher;
14. import javax.servlet.ServletConfig;
15. import javax.servlet.ServletException;
16. import javax.servlet.annotation.WebServlet;
17. import javax.servlet.http.HttpServlet;
18. import javax.servlet.http.HttpServletRequest;
19. import javax.servlet.http.HttpServletResponse;
20. import org.apache.log4j.Logger;
21. import org.quartz.CronScheduleBuilder;
22. import org.quartz.Job;
23. import org.quartz.JobBuilder;
24. import org.quartz.JobDetail;
25. import org.quartz.JobExecutionContext;
26. import org.quartz.JobExecutionException;
27. import org.quartz.ScheduleBuilder;
28. import org.quartz.Scheduler;
29. import org.quartz.SchedulerException;
30. import org.quartz.SchedulerFactory;
31. import org.quartz.Trigger;
32. import org.quartz.TriggerBuilder;
33. import org.quartz.impl.StdSchedulerFactory;
34. import com.pi4j.io.gpio.GpioController;
35. import com.pi4j.io.gpio.GpioFactory;
36. import com.pi4j.io.gpio.GpioPinDigitalOutput;
37. import com.pi4j.io.gpio.PinState;
38. import com.pi4j.io.gpio.RaspiPin;
39.
40. @WebServlet("/timer")
41.
42. public class View extends HttpServlet implements Job {
43.    private static final long serialVersionUID = 1L;
44.    static GpioController gpio;
45.    static GpioPinDigitalOutput pina;
46.    private static Connection conn = null;
47.    static DBConnection DBConnection = new DBConnection();
48.    private static Logger log = Logger.getLogger(View.class);
49.    static SchedulerFactory CronchedulerFactory = new StdSchedulerFactory();
50.    static Scheduler CronScheduler;
51.
52.    public void init(ServletConfig config) throws ServletException {
53.          log.info("Server Initialized");
54.          try {
55.                initialiseGPIO();
56.                startScheduleJobForDBCheck();
57.          } catch (SchedulerException e1) {
```

```
58.                 e1.printStackTrace();
59.           }
60.     }
61.
62.     public void destroy() {
63.           DeleteFromDB DeleteFromDB = new DeleteFromDB  ();
64.           try {
65.                 gpio.unprovisionPin(pina);
66.                 stopScheduler();
67.                 WateringScheduler watS = new WateringScheduler();
68.                 watS.stop();
69.                 pina.low();
70.                 DeleteFromDB.deleteFromDatabaseNow();
71.           } catch (SchedulerException e) {
72.                 log.error(e.getMessage());
73.           } catch (SQLException e) {
74.                 e.printStackTrace();
75.           }
76.     }
77.
78.     public int initialiseGPIO() {
79.           if (pina == null) {
80.                 gpio = GpioFactory.getInstance();
81.                 pina = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_07,
    "MyLED", PinState.LOW);
82.                 return 1;
83.           }
84.           return 0;
85.     }
86.
87.     protected void doGet(HttpServletRequest request, HttpServletResponse
    response)
88.                 throws ServletException, IOException {
89.           if (pina.isHigh()) {
90.                 RequestDispatcher rd = request.getRequestDispatcher("/WEB-
    INF/html/indexON.html");
91.                 rd.include(request, response);
92.           } else {
93.                 RequestDispatcher rd = request.getRequestDispatcher("/WEB-
    INF/html/indexOFF.html");
94.                 rd.include(request, response);
95.           }
96.     }
97.
98.     protected void doPost(HttpServletRequest request, HttpServletResponse
    response)
99.                 throws ServletException, IOException {
100.                response.setContentType("text/html");
101.                response.setCharacterEncoding("UTF-8");
102.                PrintWriter out = response.getWriter();
103.                String defaultbt = request.getParameter("defaultbt");
104.                String button = request.getParameter("button");
105.                String info = ("<h3><b>Προσοχή!</b>"
106.                            + "Υπάρχει <a  style=\"color:red;\"
    href=\"ReturnDB\">προγραμματισμένο</a> πότισμα "
107.                            + "που συμπίπτει με το παρών.</h3/<br>");
108.                String stopbt = ("<a href=\"ForceShutdown\"><button
    id=\"stop\">Διακοπή Ποτίσματος </button></a><br>");
```

```
109.                String returnbt = ("<a
    href=\"./\"><button>Επιστροφή</button></a>");
110.                out.println("<html><head>");
111.                out.println("<link rel=\"stylesheet\" href=\"graphic.css\"
    type=\"text/css\">");
112.                out.println("<meta name=\"viewport\" content=\"width=device-
    width, initial-scale=1.0\">");
113.                out.println("<meta http-equiv=\"Content-Type\"
    content=\"text/html;charset=UTF-8\">");
114.                out.println("</head><body>");
115.                out.println("<div class=\"responsed\" align=\"center\">");
116.
117.                try {
118.                      if (defaultbt != null) {
119.                            int value = Integer.parseInt(defaultbt);
120.                            if (setNow(value)) {
121.                                  out.println("<h3>Ένα πότισμα ξεκίνησε με
    διάρκεια: " + value + " λεπτά.</h3>");
122.                                  out.println(stopbt);
123.                                  response.setHeader("Refresh", "5;url=./");
124.                            } else {
125.                                  out.println(info);
126.                            }
127.                            out.println(returnbt);
128.                      } else if (button.equals("manualbt") && button != null)
    {
129.                            int minutes =
    Integer.parseInt(request.getParameter("minutes"));
130.                            if (minutes <= 0 || minutes > 300) {
131.                                  out.println("<h3><b>Προσοχή!</b> Η
    διάρκεια δεν μπορεί να είναι μηδενική,"
132.                                                + " αρνητική ή μεγαλύτερη από
    300 λεπτά.</h3>");
133.                            } else {
134.                                  if (setNow(minutes)) {
135.                                        out.println("<h3>Ένα πότισμα
    ξεκίνησε με διάρκεια: " + minutes + " λεπτά.</h3>");
136.                                        out.println(stopbt);
137.                                        response.setHeader("Refresh",
    "5;url=./");
138.                                  } else {
139.                                        out.println(info);
140.                                  }
141.                            }
142.                            out.println(returnbt);
143.                      } else if (button.equals("schedulebt") && button !=
    null) {
144.                            Calendar c = Calendar.getInstance();
145.                            int day =
    Integer.parseInt((request.getParameter("day")));
146.                            int hourFrom =
    Integer.parseInt((request.getParameter("hourFrom")));
147.                            int hourTo =
    Integer.parseInt((request.getParameter("hourTo")));
148.                            int duration = 0;
149.                            int limit = 0;
150.
151.                            if ((hourFrom >= 19) && (hourTo <= 5)) {
```

```
152.                                     duration = ((hourTo - hourFrom) + 24) *
    60;
153.                                     limit = (hourTo - hourFrom) + 24;
154.                               } else if (hourFrom >= 0 && hourTo <= 23) {
155.                                     if (hourFrom >= hourTo) {
156.                                           out.println(
157.                                                       "<h3><b>Προσοχή!</b>Η
    ώρα έναρξης πρέπει να ειναι μικρότερη απο την ώρα λήξης.</h3>");
158.                                           out.println(returnbt);
159.                                           response.setHeader("Refresh",
    "5;url=./");
160.                                           return;
161.                                     } else {
162.                                           duration = (hourTo - hourFrom) * 60;
163.                                           limit = hourTo - hourFrom;
164.                                     }
165.                               }
166.                               if (limit > 5) {
167.                                     out.println("<h3><b>Προσοχή!</b>Η διάρκεια
    δεν μπορεί να είναι μεγαλύτερη από 5 ώρες.</h3>");
168.                                     out.println(returnbt);
169.                                     response.setHeader("Refresh", "5;url=./");
170.                                     return;
171.                               } else if (c.get(Calendar.DAY_OF_WEEK) == day &&
    c.get(Calendar.HOUR_OF_DAY) == hourFrom) {
172.                                     if (setNow(duration)) {
173.                                           out.println("<h3>Ένα πότισμα
    ξεκίνησε με διάρκεια: " + duration + " Λεπτά.</h3>");
174.                                           out.println(stopbt);
175.                                     } else {
176.                                           out.println(info);
177.                                     }
178.                                     out.println(returnbt);
179.                               } else {
180.                                     String status = "off";
181.                                     if (insertInDatabase(day, hourFrom, 00,
    hourTo, duration, status) == 1) {
182.
       response.sendRedirect("./FutureSchedule?duration=" + duration +
    "&hourFrom=" + hourFrom
183.                                                       + "&hourTo=" + hourTo +
    "&day=" + day + "&status=" + status);
184.                                           response.setHeader("Refresh",
    "8;url=./");
185.                                     } else {
186.                                           out.println(info);
187.                                           out.println(returnbt);
188.                                     }
189.                               }
190.                         }
191.                   } catch (NumberFormatException e) {
192.                         out.println("<h3><b>Προσοχή!</b> Όλες οι παράμετροι θα
    πρέπει να είναι ακέραιοι.</h3>");
193.                         response.setHeader("Refresh", "8;url=./");
194.                         out.println(returnbt);
195.                   } catch (SQLException e) {
196.                         e.printStackTrace();
197.                   } catch (AddressException e) {
198.                         e.printStackTrace();
```

```
199.                } catch (InstantiationException e) {
200.                      e.printStackTrace();
201.                } catch (IllegalAccessException e) {
202.                      e.printStackTrace();
203.                } catch (ClassNotFoundException e) {
204.                      e.printStackTrace();
205.                }
206.                out.println("</div></body></html>");
207.          }
208.
209.          public static void initialiseDatabase()
210.                      throws SchedulerException, SQLException,
    InterruptedException, AddressException {
211.                Log.info("Initialize Database");
212.                String results = null;
213.                Calendar c = Calendar.getInstance();
214.                conn = DBConnection.createDBConnection();
215.                String query = "SELECT * FROM wateringtable;";
216.                Statement st = conn.createStatement();
217.                ResultSet rs = st.executeQuery(query);
218.                WateringScheduler watS = new WateringScheduler();
219.                Subscribe manageToday = new Subscribe();
220.                while (rs.next()) {
221.                      int day = rs.getInt("Day");
222.                      int hourFrom = rs.getInt("hourFrom");
223.                      int hourTo = rs.getInt("hourTo");
224.
225.                      if (day == c.get(Calendar.DAY_OF_WEEK) && hourFrom ==
    c.get(Calendar.HOUR_OF_DAY)) {
226.                            Log.info("Event for watering found!");
227.                            int duration = rs.getInt("Duration");
228.                            String update = "UPDATE wateringtable SET
    Status='on' WHERE Day=" + day + " and hourFrom=" + hourFrom
229.                                        + " and hourTo=" + hourTo + ";";
230.
231.                            st.executeQuery(update);
232.                            results = "Ώρα ποτίσματος : " +
    rs.getInt("hourFrom") + " Διάρκεια Ποτίσματος : "
233.                                        + rs.getInt("Duration")+"\n";
234.                            Log.info(results);
235.                            DBConnection.closeConnection(conn);
236.                            watS.startExecutionAt(duration);
237.                            TimeUnit.SECONDS.sleep(30);
238.                            manageToday.sendEmailDaily(results, 1);
239.                      }
240.                }
241.          }
242.
243.          public int sameDayCollision(int day, int hour1, double hour2, int
    duration) throws SQLException {
244.                conn = DBConnection.createDBConnection();
245.                Log.info("Checking For collision");
246.                String query = "SELECT * FROM wateringtable";
247.                Statement st = conn.createStatement();
248.                ResultSet rs = st.executeQuery(query);
249.                int hourFDB = 0;
250.                double hourTDB = 0;
251.                int dayDB = 0;
252.
```

```
253.                while (rs.next()) {
254.                       dayDB = rs.getInt("Day");
255.                       hourFDB = rs.getInt("hourFrom");
256.                       hourTDB = rs.getDouble("hourTo");
257.
258.                       if (day == dayDB && hour1 >= 19 && hour2 <= 4 && hourFDB
    >= 19 && hourTDB <= 4) {
259.                              if (((hour1 <= hourFDB) && (hour2 >= hourTDB)) ||
    ((hour1 >= hourFDB) && (hour2 <= hourTDB))) {
260.                                     log.info("Day and duration collision----
    1");
261.                                     return 0;
262.                              }
263.                              if (((hour1 >= hourFDB) && (hour2 >= hourTDB)) ||
    ((hour1 <= hourFDB) && (hour2 <= hourTDB))) {
264.                                     log.info("Day and duration collision----
    2");
265.                                     return 0;
266.                              }
267.                       }
268.                       if ((day == dayDB && ((hour1 < 19 && hourFDB <= 19) ||
    (hour1 >= 18 && hour2 >= 19)))) {
269.                              if (((hour1 <= hourFDB) && (hour2 > hourFDB)) ||
    ((hour1 >= hourFDB) && (hour2 <= hourTDB))) {
270.                                     log.info("Day and duration collision----
    3");
271.                                     return 0;
272.                              }
273.                              if ((hour1 < hourTDB) && (hour2 >= hourTDB)) {
274.                                     log.info("Day and duration collision----
    4");
275.                                     return 0;
276.                              }
277.                       }
278.                       if (day == dayDB && hour1 >= 19 && hour2 <= 4 && hourFDB
    >= 14 && hourTDB > hour1) {
279.                              log.info("Finishing Time collision's to same days
    starting time----1");
280.                              return 0;
281.                       }
282.                       if (day == dayDB && (hour1 >= 19 || hour1 < 19) && hour2
    >= 19 && hourFDB >= 19 && hourTDB <= 4
283.                                     && hour2 > hourFDB) {
284.                              log.info("Finishing Time collision's to same days
    starting time----2");
285.                              return 0;
286.                       }
287.                       if (day + 1 == 8 && 1 == dayDB && hour1 >= 19 && hour2
    <= 5 && hourFDB <= 5 && hourTDB <= 9
288.                                     && hour2 > hourFDB) {
289.                              log.info("Finishing Time collision's to nexts
    days starting time");
290.                              return 0;
291.                       }
292.                       if (day + 1 == dayDB && hour1 >= 19 && hour2 <= 5 &&
    hourFDB <= 5 && hourTDB <= 9 && hourFDB < hour2) {
293.                              log.info("Finishing Time collision's to nexts
    days starting time");
294.                              return 0;
```

```
295.                         }
296.                         if (day - 1 == 0 && 7 == dayDB && hour1 <= 4 && hourFDB
    >= 19 && hourTDB <= 4 && hourTDB > hour1) {
297.                               log.info("Starting Time collision's to before's
    days ending time");
298.                               return 0;
299.                         }
300.                         if (day - 1 == dayDB && hour1 <= 4 && hourFDB >= 19 &&
    hourTDB <= 5 && hourTDB > hour1) {
301.                               log.info("Starting Time collision's to before's
    days ending time");
302.                               return 0;
303.                         }
304.                   }
305.                   return 1;
306.             }
307.
308.             public int insertInDatabase(int day, int hourFrom, int hourFromMin,
    double hourTo, int duration, String status)
309.                         throws SQLException, AddressException,
    InstantiationException, IllegalAccessException,
310.                         ClassNotFoundException {
311.                   int check = sameDayCollision(day, hourFrom, hourTo, duration);
312.                   if (check == 1) {
313.                         String query = " insert into wateringtable (Day,
    hourFrom, hourFromMin, hourTo, duration, Status, Date)"
314.                                     + " values (?, ?, ?, ?, ?, ?, ?)";
315.                         Calendar c = Calendar.getInstance();
316.                         java.sql.Date startDate = new
    java.sql.Date(c.getTime().getTime());
317.                         PreparedStatement preparedStmt =
    conn.prepareStatement(query);
318.                         preparedStmt.setInt(1, day);
319.                         preparedStmt.setInt(2, hourFrom);
320.                         preparedStmt.setInt(3, hourFromMin);
321.                         preparedStmt.setDouble(4, hourTo);
322.                         preparedStmt.setInt(5, duration);
323.                         preparedStmt.setString(6, status);
324.                         preparedStmt.setDate(7, startDate);
325.                         preparedStmt.execute();
326.                         log.info("Insertion Completed");
327.                         DBConnection.closeConnection(conn);
328.                         return 1;
329.                   } else {
330.                         log.info("Υπαρχει ηδη προγραμματισμενο πότισμα");
331.                         DBConnection.closeConnection(conn);
332.                         return 0;
333.                   }
334.             }
335.
336.             public int disablePin() {
337.                   pina.low();
338.                   return 1;
339.             }
340.
341.             public int enablePin() {
342.                   pina.high();
343.                   return 1;
344.             }
```

```
345.
346.         public static void startScheduleJobForDBCheck() throws
    SchedulerException {
347.                 log.info("Schedule will start for checking database at the
    start of each day");
348.                 JobDetail job =
    JobBuilder.newJob(View.class).withIdentity("view").build();
349.
350.                 @SuppressWarnings("rawtypes")
351.                 ScheduleBuilder scheduleBuilder =
    CronScheduleBuilder.cronSchedule("0 0 * * * ?");
352.
353.                 @SuppressWarnings("unchecked")
354.                 Trigger trigger =
    TriggerBuilder.newTrigger().withSchedule(scheduleBuilder).build();
355.
356.                 CronScheduler = CronchedulerFactory.getScheduler();
357.                 CronScheduler.scheduleJob(job, trigger);
358.                 CronScheduler.start();
359.         }
360.
361.         public void stopScheduler() throws SchedulerException {
362.                 CronScheduler.shutdown(true);
363.         }
364.
365.         public void execute(JobExecutionContext jExeCtx) throws
    JobExecutionException {
366.                 try {
367.                         log.info("Check database insertions!");
368.                         initialiseDatabase();
369.                 } catch (SchedulerException e) {
370.                         e.printStackTrace();
371.                 } catch (SQLException e) {
372.                         e.printStackTrace();
373.                 } catch (InterruptedException e) {
374.                         e.printStackTrace();
375.                 } catch (AddressException e) {
376.                         e.printStackTrace();
377.                 }
378.         }
379.
380.         public boolean setNow(int duration) throws AddressException,
    InstantiationException, IllegalAccessException,
381.                         ClassNotFoundException, SQLException {
382.                 Calendar c = Calendar.getInstance();
383.                 int today = c.get(Calendar.DAY_OF_WEEK);
384.                 int hourFrom = c.get(Calendar.HOUR_OF_DAY);
385.                 int hourMinutes = c.get(Calendar.MINUTE);
386.                 double hourSeconds = (double) c.get(Calendar.SECOND) / 3600;
387.                 log.info(hourSeconds);
388.                 double hourTo = 0;
389.                 double computehourMinute = (double) hourMinutes / 60;
390.                 double computeDuration = (double) duration / 60;
391.                 double computeDelay = computehourMinute + computeDuration +
    hourSeconds;
392.
393.                 if (computeDelay >= 1) {
394.                         hourTo = (double) (hourFrom + computeDelay);
395.                         if (hourTo >= 24) {
```

```
396.                                int hourDif = hourFrom - 24;
397.                                hourTo = 0;
398.                                hourTo = (double) (hourDif + computeDelay);
399.                                Log.info(hourTo);
400.                          }
401.                    } else if (computeDelay < 1) {
402.                          hourTo = (double) hourFrom + computeDelay;
403.                    }
404.                    String status = "on";
405.                    if (insertInDatabase(today, hourFrom, hourMinutes, hourTo,
    duration, status) == 1 && pina.isLow()) {
406.                          WateringScheduler watS = new WateringScheduler();
407.                          watS.startExecutionAt(duration);
408.                          return true;
409.                    } else
410.                          return false;
411.              }
412.    }
```

Η παραπάνω κλάση είναι η βασική που καλείται κάθε φορά που ξεκινάμε την διαδικτυακή εφαρμογή. Περιέχει την μέθοδο doGet η οποία προβάλει την αρχική σελίδα της εφαρμογής. Αποτελείται από 14 μεθόδους συνολικά που κάνουν τα εξής.

- init: Είναι η μέθοδος που καλείται όταν τρέχει για πρώτη φορά η εφαρμογή. Καλεί την μέθοδο που αρχικοποιεί και ορίζει την κατάσταση του ακροδέκτη GPIO_07. Επίσης ξεκινάει την λειτουργία του χρονομετρητή (`startScheduleJobForDBCheck`) που ελέγχει τις καταχωρίσεις στην βάση δεδομένων.
- destroy: Μέθοδος η οποία καλείται όταν διαγράφουμε την εφαρμογή από τον server. Περιέχει μεθόδους αποδέσμευσης του ακροδέκτη, διακοπής του χρονομετρητή και διαγράφη της τρέχουσας διεργασίας αν υπάρχει.
- initialiseGPIO: Αρχικοποίηση της κατάστασης του ακροδέκτη σε low και δήλωση χρήσης του GPIO_07 με την ονομασία pina.
- doGet: Ελέγχει την κατάσταση του ακροδέκτη και καλεί την ανάλογη αρχική σελίδα από την τοποθεσία /WEB-INF/html/.
- doPost: Ελέγχει την επιλογή εισαγωγής ποτίσματος του χρήστη όλων των πεδίων που αφορούν το πότισμα. Διαχωρίζει τον τύπο ποτίσματος και ορίζει την χρονική διάρκεια που αυτό θα εκτελεσθεί στο παρών ή μελλοντικά.
- initialiseDatabase: Ελέγχει εάν κάποιο από τα προγραμματισμένα ποτίσματα τηρεί τις προϋποθέσεις για να ξεκινήσει και θέτει τον χρονομετρητή-νήμα (WateringScheduler) σε λειτουργία.

- sameDayCollision: Κάνει έλεγχο με τις ήδη υπάρχουσες καταχωρήσεις για περιπτώσεις σύγκρουσης μεταξύ τους.
- insertInDatabase: Εισάγει στον πίνακα προγραμματισμένων ποτισμάτων τα δεδομένα εισαγωγής από το χρήστη με την βοήθεια της παραπάνω μεθόδου.
- disablePin: Διακόπτει την λειτουργία του ακροδέκτη GPIO_07.
- enablePin: Ενεργοποιεί την λειτουργία του ακροδέκτη GPIO_07.
- startScheduleJobForDBCheck: Προγραμματισμένη διεργασία που εκτελείται στην αρχή κάθε ώρας.
- stopScheduler: Σταματάει την παραπάνω διεργασία.
- execute: Το περιεχόμενο αυτής της μεθόδου ξεκινάει τον έλεγχο εάν κάποιο προγραμματισμένο πότισμα τηρεί της προϋποθέσεις της μεθόδου initialiseDatabase.
- setNow: Αυτή η μέθοδος υλοποιείτε κατά την επιλογή ενός παρών ποτίσματος. «Τραβάει» στοιχεία της ώρας και της ημερομηνίας και τα αποθηκεύει στον πίνακα προγραμματισμένων ποτισμάτων για αποφυγή συγκρούσεων.

3.2.10 WATERINGSCHEDULER.JAVA

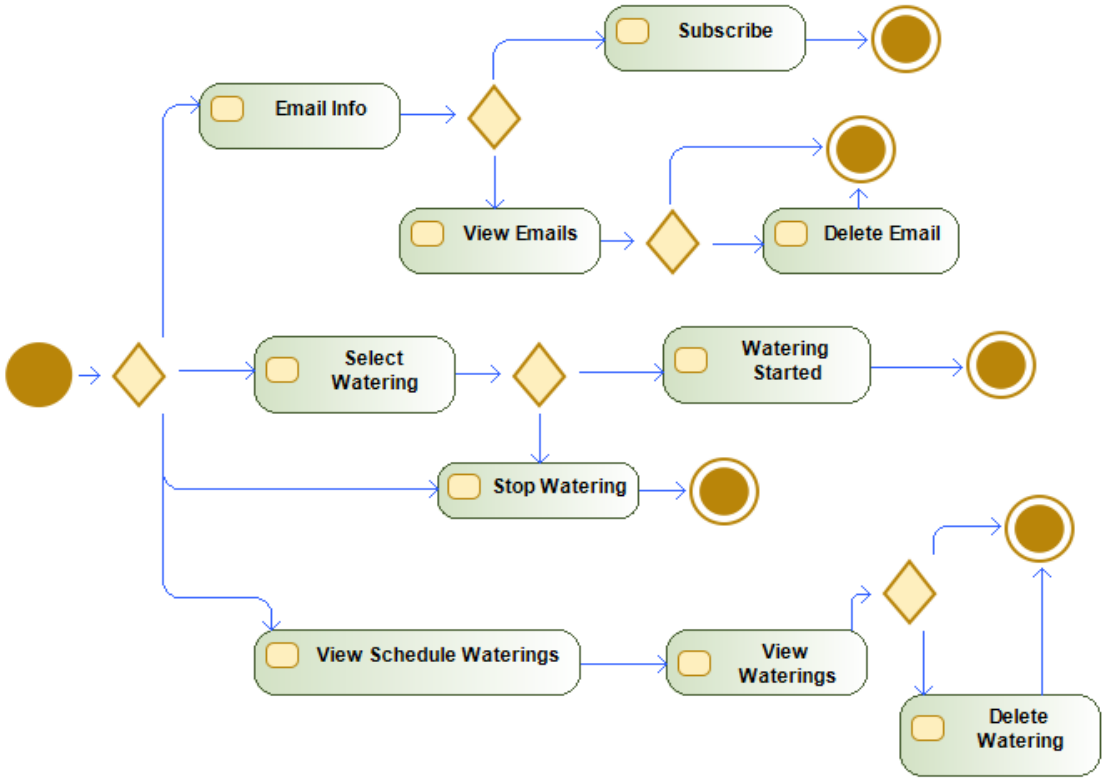
```
1. package watering;

2. import java.io.IOException;
3. import java.util.concurrent.Executors;
4. import java.util.concurrent.ScheduledExecutorService;
5. import java.util.concurrent.TimeUnit;

6. import javax.mail.internet.AddressException;
7. import javax.servlet.ServletException;
8. import javax.servlet.annotation.WebServlet;
9. import javax.servlet.http.HttpServlet;
10. import javax.servlet.http.HttpServletRequest;
11. import javax.servlet.http.HttpServletResponse;
12. import org.apache.log4j.Logger;

13. @WebServlet("/scheduler")
14. public class WateringScheduler extends HttpServlet {

15. private static final long serialVersionUID = 1L;
16. private static Logger log = Logger.getLogger(WateringScheduler.class);
17. static ScheduledExecutorService executorService;

18. public void startExecutionAt(int duration) {
19. executorService = Executors.newScheduledThreadPool(1);
20. final Runnable thread = new Runnable() {
21. public void run() {

22. Subscribe manageToday = new Subscribe();
23. View View = new View();
```

```
24. ForceShutdown ForceShutdown = new ForceShutdown();
25. try {
26. log.info("Execution started! Duration: " + duration + "!");
27. View.enablePin();
28. TimeUnit.SECONDS.sleep(duration * 60 - 2);
29. ForceShutdown.closeWaterNow();
30. manageToday.sendEmailDaily(
31. "Το πότισμα με διάρκεια " + duration + " λεπτά ολοκληρώθηκε με επιτυχία!",
    2);
32. log.info("Execution finished normally");
33. } catch (Exception ex) {
34. log.error("Force Shutdown Thread " + ex.getMessage());
35. try {
36. manageToday.sendEmailDaily(
37. "Το πότισμα με διάρκεια " + duration + " διακόπηκε από εσάς ή από κάποιο
    σφάλμα", 3);
38. } catch (AddressException e) {
39. e.printStackTrace();
40. }
41. }
42. }
43. };
44. executorService.schedule(thread, 0, TimeUnit.MILLISECONDS);
45. }

46. public void stop() {
47. executorService.shutdownNow();
48. }

49. public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException {
50. doPost(request, response);
51. }
52. }
```

Αυτή η κλάση-νήμα είναι ο βασικός χρονομετρητής της εφαρμογής. Δημιουργεί ένα νήμα που διατηρεί την κατάσταση του ακροδέκτη ενεργή για τον χρόνο που θα του ορίσουμε. Απαρτίζεται από τις 3 παρακάτω μεθόδους.

- startExecutionAt: Νήμα το οποίο θέτει σε λειτουργία τον ακροδέκτη για τον χρόνο που του έχουμε ορίσει (duration). Κατά την ολοκλήρωση αυτής της διεργασίας αποστέλλεται με την χρήση της μεθόδου sendEmailDaily email επιβεβαίωσης στα εγγεγραμμένα email. Σε περίπτωση απρόσμενης διακοπής του ποτίσματος ο χρήστης λαμβάνει το μήνυμα ταχυδρομείου που περιλαμβάνεται στην λειτουργία catch (33-38).
- stop: Διακόπτει την λειτουργία του νήματος.
- doGet: Καλεί την μέθοδο doPost.

3.3 ΔΙΑΓΡΑΜΜΑ ΕΝΕΡΓΕΙΩΝ (ACTIVITY DIAGRAM)

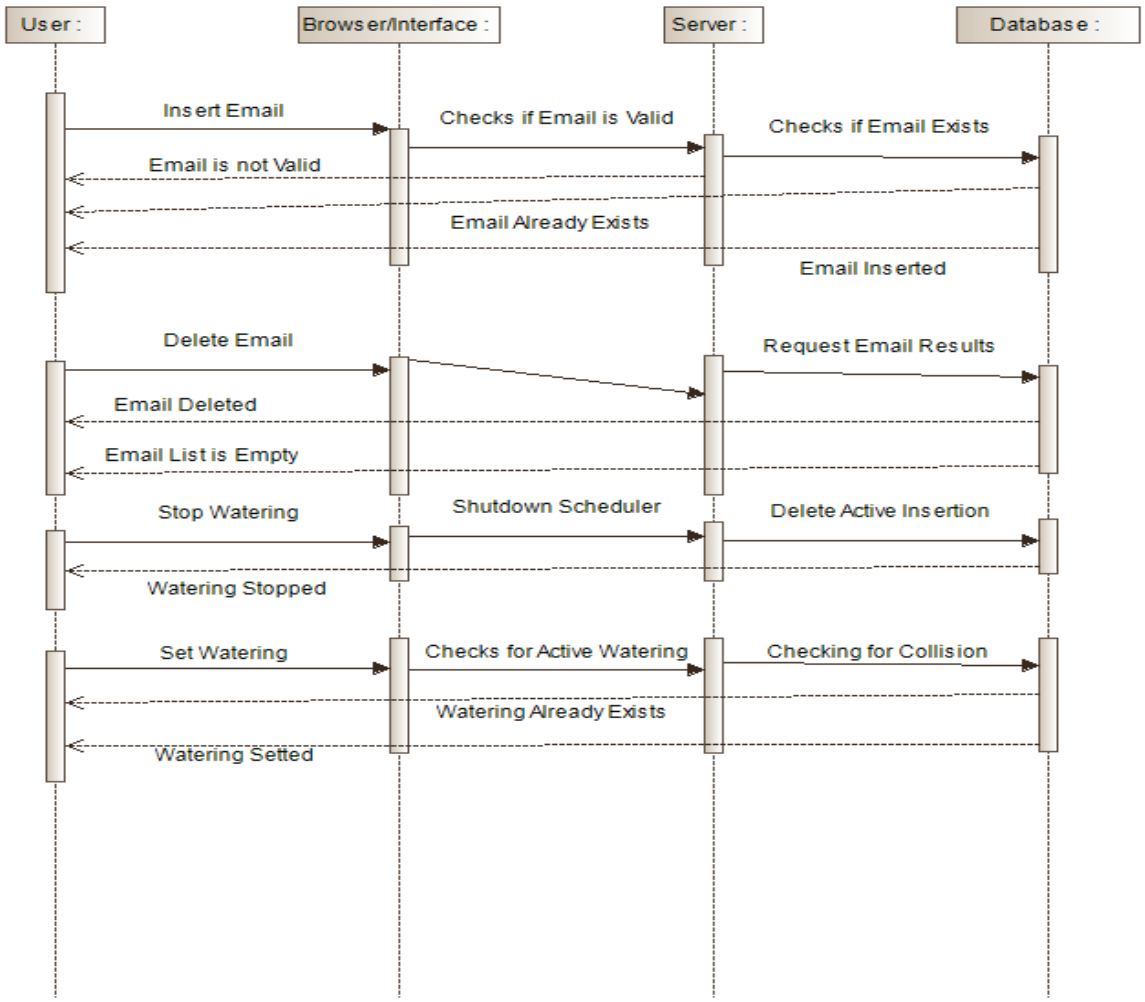


Εικόνα 3.3 Activity Diagram

Στο διάγραμμα παραπάνω απεικονίζονται οι ενέργειες που μπορεί να κάνει ο χρήστης χρησιμοποιώντας την εφαρμογή καθώς και οι αποφάσεις που μπορεί να πάρει έπειτα από κάθε ενέργεια. Τέσσερις είναι οι βασικές ενέργειες που μπορεί να κάνει ο χρήστης:

1. Με την επιλογή του πεδίου Email Info μπορεί να εισάγει η να διαγράψει το email που επιθυμεί.
2. Στο πεδίο Select Watering μπορεί να επιλέξει ανάμεσα σε ένα παρών η προγραμματισμένο πότισμα. Έπειτα από την επιλογή του μπορεί να το διακόψει η να το ακυρώσει.
3. Μπορεί να διακόψει άμεσα την διεξαγωγή του τρέχων ποτίσματος.
4. Μπορεί να προβάλει τον πίνακα των ποτισμάτων και να διαγράψει κάποιο από αυτά.

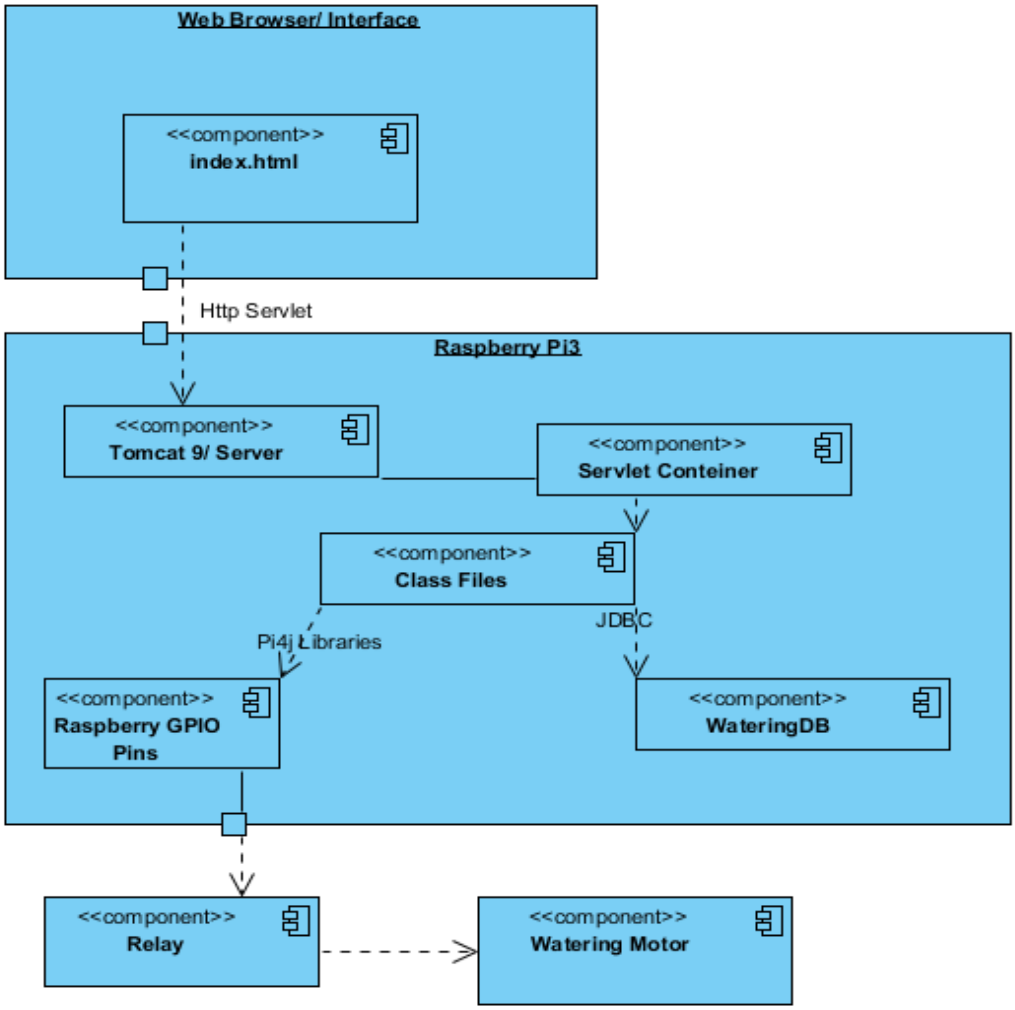
3.4 ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ (SEQUENCE DIAGRAM)



Εικόνα 3.4 Sequence Diagram

Στο παραπάνω διάγραμμα μπορούμε να δούμε όλες τις πιθανές επιλογές που μπορεί να κάνει ο χρήστης, τον τρόπο που η επιλογή επεξεργάζεται από την διαδικτυακή εφαρμογή και τις πιθανές απαντήσεις που μπορεί να λάβει πίσω στον browser σαν απάντηση. Ουσιαστικά βλέπουμε το «ταξίδι» της πληροφορίας και τον τρόπο που επεξεργάζεται από το σύστημα.

3.5 ΔΙΑΓΡΑΜΜΑ ΕΞΑΡΤΗΜΑΤΩΝ (COMPONENT DIAGRAM)



Εικόνα 3.5 Component Diagram

Στο διάγραμμα παραπάνω μπορούμε να δούμε γραφικά πως επικοινωνούν μεταξύ τους τα εξαρτήματα που απαρτίζουν τον αυτοματισμό καθώς και τις σχέσεις μεταξύ τους. Πατώντας την διεύθυνση της διαδικτυακής εφαρμογής στον φυλλομετρητή απεικονίζεται η αρχική σελίδα που βρίσκεται στο αρχείο index.html που είναι αποθηκευμένη στον server ο οποίος είναι εγκατεστημένος στο Raspeberry Pi 3. Έπειτα δημιουργείτε επικοινωνία με τις κλάσεις και την βάση δεδομένων του συστήματος. Τέλος οι ακροδέκτες ενεργοποιούν, με την χρήση των βιβλιοθηκών Pi4J, την επαφή που βρίσκεται στο Relay και αυτό με την σειρά του αφήνει το ρεύμα να περάσει και να ενεργοποιήσει το μοτέρ ποτίσματος.

4 ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ ΔΙΑΔΥΚΤΙΑΚΗΣ ΕΦΑΡΜΟΓΗΣ WATERING

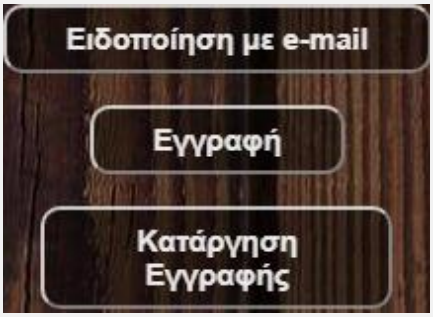
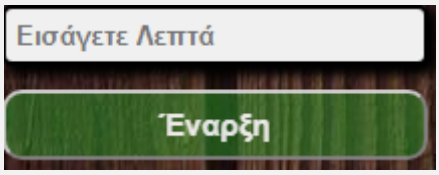
Στον παρών κεφάλαιο θα αναλύσουμε παραδείγματα χρήσης της διαδικτυακής εφαρμογής Watering. Η αρχική σελίδα της εφαρμογής θα χρησιμοποιείται σαν αρχή κάθε παραδείγματος και για αυτόν τον λόγο θα κάνουμε την γενική αναφορά της σε αυτό το σημείο.

Εικόνα 4.1 Αρχική Σελίδα Εφαρμογής

Όπως βλέπουμε στην παραπάνω εικόνα η αρχική σελίδα της διαδικτυακής εφαρμογής απαρτίζεται από 6 βασικά στοιχεία τα οποία είναι στην μορφή κυρίως κουμπιών. Για να έχουμε πρόσβαση σε αυτή πληκτρολογούμε σε οποιοδήποτε φυλλομετρητή την διεύθυνση: http://papadakis92.ddns.net/Watering.

4.1 ΓΕΝΙΚΑ ΓΙΑ ΤΗΝ ΕΦΑΡΜΟΓΗ

Στον παρακάτω πίνακα μπορούμε να δούμε τις βασικές λειτουργίες της διαδικτυακής εφαρμογής που εμπεριέχονται στην αρχική σελίδα.

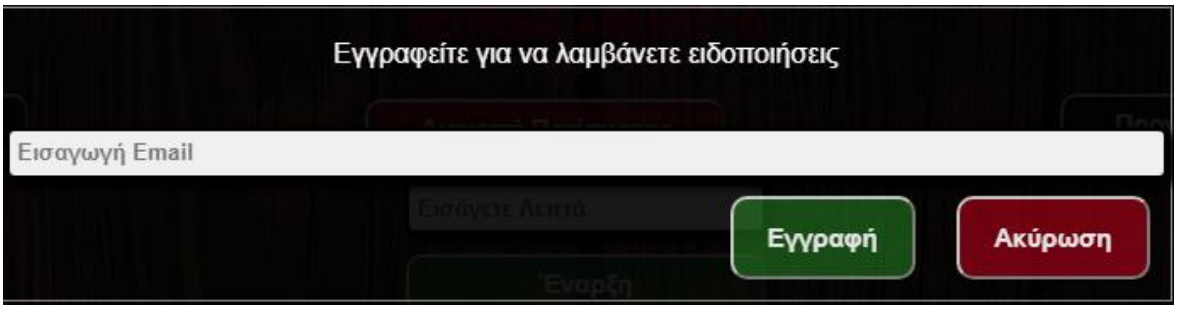
Πεδίο	Λειτουργία
	Πατώντας πάνω σε αυτό το πεδίο ο χρήστης μπορεί να εισάγει το e-mail του έτσι ώστε να λαμβάνει ενημερώσεις σχετικά με την πορεία των ποτισμάτων όπως επίσης και να το διαγράψει εάν αυτό υπάρχει ήδη.
	Διακόπτει το τρέχων πότισμα εάν αυτό υπάρχει.
	Προβάλει όλα τα προγραμματισμένα ποτίσματα με την δυνατότητα διαγράφης τους.
	Λαμβάνει τον χρόνο ποτίσματος σε λεπτά εφόσον δεν υπάρχει τρέχων ή σύγκρουση με κάποιο ήδη προγραμματισμένο.
	Προκαθορισμένα κουμπιά χρόνου για την εκκίνηση ενός ποτίσματος εφόσον δεν υπάρχει ήδη τρέχων ή σύγκρουση με κάποιο ήδη προγραμματισμένο. Ο μέγιστος χρόνος στην υλοποίηση του προγράμματος αυτού είναι 5 ώρες. *
	Λαμβάνει την μέρα, την ώρα έναρξης και την ώρα λήξης που ο χρήστης επιθυμεί να πραγματοποιηθεί το πότισμα.

Πίνακας 4-1 Γενικές Λειτουργίες Εφαρμογής

*Ο μέγιστος χρόνος μπορεί να αλλάξει σε οποιαδήποτε τιμή εάν αυτό είναι επιθυμητό.

4.2 ΕΓΓΡΑΦΗ ΜΕ EMAIL

Πατώντας το κουμπί «Εγγραφή» το οποίο εμφανίζεται στο αναπτυσσόμενο υπο-μενού «Ειδοποίηση με e-mail» μας εμφανίζεται η παρακάτω εικόνα.



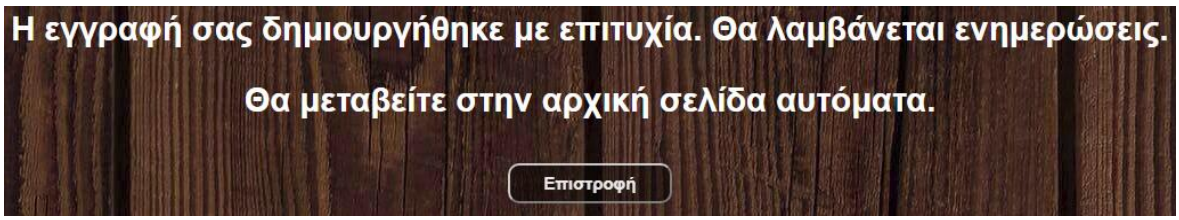
Εικόνα 4.2 Εγγραφή Email

1. Βάζουμε το email μας στο πεδίο «Εισαγωγή Email».
2. Πατάμε «Εγγραφή».

<u>Περιπτώσεις:</u>

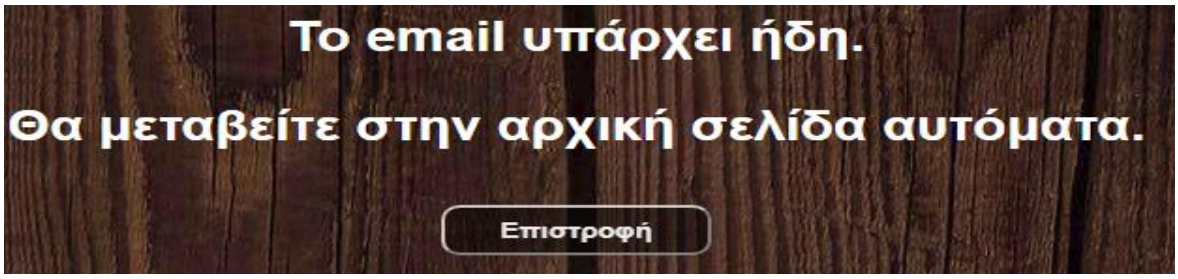
1. Εάν η εγγραφή μας ολοκληρωθεί με επιτυχία θα μετάβουμε στην σελίδα που απεικονίζεται παρακάτω και θα λάβουμε μήνυμα επιβεβαίωσης στο email που καταχωρίσαμε. Έπειτα θα λάβουμε μετά από μερικά δευτερόλεπτα μήνυμα επιβεβαίωσης στην διεύθυνση την οποία δηλώσαμε και θα λαμβάνουμε τις παρακάτω ενημερώσεις αναλόγως την διεργασία που εκτελείται.

- Εκκίνηση προγραμματισμένου ποτίσματος.
- Λήξη ποτίσματος.
- Απρόσμενη διακοπή ποτίσματος.



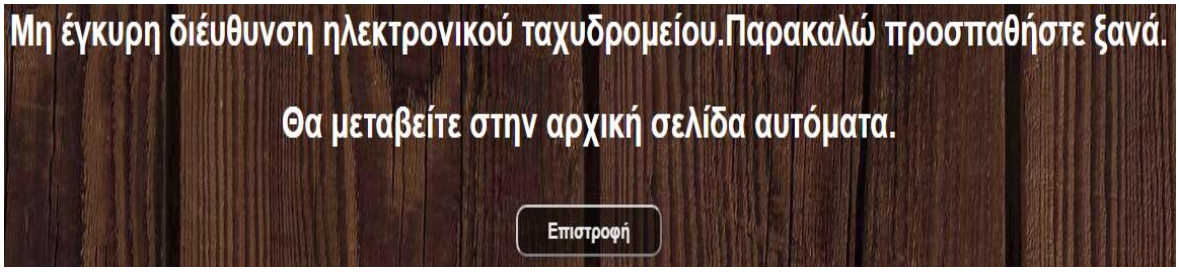
Εικόνα 4.3 Επιτυχής Εγγραφή

2. Στην περίπτωση που το email που θέλουμε να εισάγουμε έχει καταχωρηθεί ήδη θα πάρουμε την παρακάτω εικόνα σαν απάντηση.



Εικόνα 4.4 Ήδη Υπάρχων Email

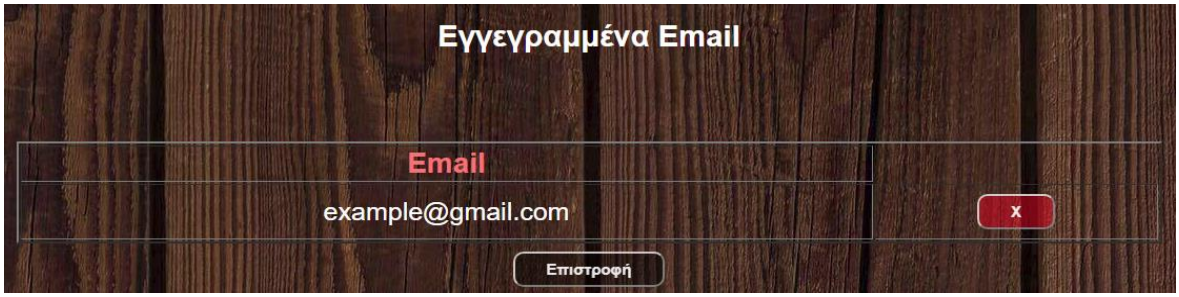
3. Τέλος εάν η δομή του email είναι λανθασμένη (π.χ papadakis.com) θα εμφανιστεί το παρακάτω μήνυμα.



Εικόνα 4.5 Μη Έγκυρο Email

4.3 ΔΙΑΓΡΑΦΗ EMAIL

Πατώντας το κουμπί «Κατάργηση Εγγραφής» το οποίο εμφανίζεται στο αναπτυσσόμενο υπομενού «Ειδοποίηση με e-mail» μας εμφανίζεται η παρακάτω σελίδα οπού μας δίνεται η δυνατότητα να προβάλουμε τα εγγεγραμμένα email (εάν υπάρχουν) και να διαγράψουμε οποία θέλουμε.



Εικόνα 4.6 Εγγεγραμμένα email

Για να διαγράφουμε το email που επιθυμούμε πατάμε δεξιά στο εικονίδιο x.

4.4 ΠΡΟΒΟΛΗ ΚΑΙ ΔΙΑΓΡΑΦΗ ΠΡΟΓΡΑΜΜΑΤΙΣΜΕΝΟΥ ΚΑΙ ΤΡΕΧΩΝ ΠΟΤΙΣΜΑΤΟΣ

Ο χρήστης της εφαρμογής έχει την δυνατότητα να προβάλει και να διαγράψει ποτίσματα που αφορούν τόσο το παρών όσο και το μέλλον. Πατώντας πάνω στο κουμπί «Προγραμματισμένο Πότισμα» θα μεταβεί στην παρακάτω σελίδα, εφόσον υπάρχουν προγραμματισμένα ή τρεχων ποτίσματα.

Εικόνα 4.7 Προγραμματισμένα Ποτίσματα

Στον παραπάνω πίνακα προγραμματισμένων ποτισμάτων μπορούμε να διακρίνουμε 4 βασικά χαρακτηρίστηκα.

- Την ημέρα για την οποία το πότισμα είναι προγραμματισμένο.
- Την ώρα που θα ξεκινήσει.
- Την χρονική διάρκεια που θα εκτελεσθεί σε λεπτά.
- Την κατάσταση οπού βρίσκεται την παρούσα χρονική στιγμή.

Σε περίπτωση που θέλουμε να διαγράψουμε κάποιο από αυτά πατάμε το κουμπί 

που βρίσκεται στα αριστερά. Εάν η κατάσταση του ποτίσματος του οποίου θέλουμε να διαγράψουμε είναι «Ενεργό» τότε παράλληλα με την διαγραφή του γίνεται και ο τερματισμός του. Να σημειωθεί επίσης πως κάθε πότισμα που έχει ορισθεί, θα διεκπεραιωθεί μια μόνο φόρα και έπειτα θα διαγραφεί αυτόματα από την λίστα προγραμματισμένων ποτισμάτων.

4.5 ΕΠΙΛΟΓΗ ΠΟΤΙΣΜΑΤΟΣ

Η επιλογή ποτίσματος αποτελεί το κύριο συστατικό της εφαρμογής και χωρίζεται σε 2 βασικές κατηγορίες :

1. Παρών πότισμα.
2. Προγραμματισμένο πότισμα.

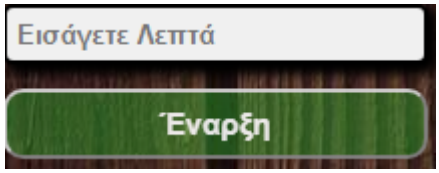
Και στις 2 περιπτώσεις τα ποτίσματα εγγράφονται στην λίστα ποτισμάτων για να μπορεί ο χρήστης να κάνει καλύτερη διαχείριση.

4.5.1 ΠΑΡΩΝ ΠΟΤΙΣΜΑ

Χωρίζεται σε 2 υποκατηγορίες για την όσο δυνατόν καλύτερη κάλυψη των αναγκών του χρήστη. Έτσι λοιπόν δίνεται η δυνατότητα είτε να επιλέξουμε ποσά λεπτά θέλουμε είτε να πατήσουμε ένα κουμπί προκαθορισμένου χρόνου για την διεκπεραίωση ενός ποτίσματος.

1. Προσαρμοσμένο πότισμα.

Εάν θέλουμε να ορίσουμε ένα διαμορφούμενο πότισμα θα επικεντρωθούμε στην λειτουργία της αρχικής οθόνης που απεικονίζεται στην παρακάτω εικόνα.



Εικόνα 4.8 Προσαρμοσμένο Πότισμα

Τα βήματα είναι τα εξής για την εισαγωγή ενός διαμορφούμενου ποτίσματος.

1. Στο πεδίο «Εισάγετε Λεπτά» μπορούμε να βάλουμε χρονική διάρκεια από 1 έως 300 λεπτά.
2. Πατάμε το κουμπί «Έναρξη».

Εάν το πότισμα που εισάγουμε δεν συμπίπτει με κάποιο ήδη υπάρχων ή κάποιο προγραμματισμένο τότε θα λάβουμε το εξής μήνυμα επιβεβαίωσης.

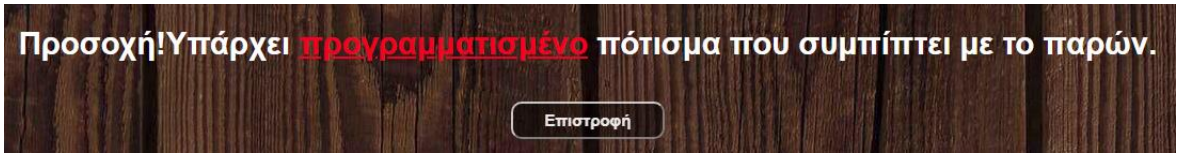
Εικόνα 4.9 Επιβεβαίωση Έναρξης

Σε περίπτωση που θέλουμε να τερματίσουμε την εκτέλεσή της ενέργειας που μόλις υποβάλαμε, πατάμε το κουμπί 

.

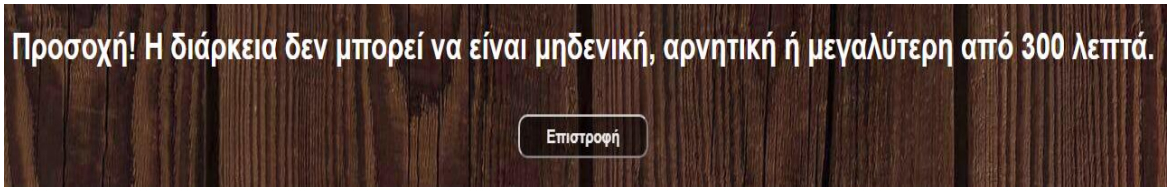
Σε διαφορετικές περιπτώσεις που το πότισμα συμπίπτει με κάποιο άλλο ή τα δεδομένα που εισάγουμε δεν είναι σωστά ή η χρονική διάρκεια υπερβαίνει τα 300 λεπτά, θα λάβουμε σαν απάντηση ένα από τα παρακάτω:

1. Σε περίπτωση σύμπτυξης με τρέχων ή προγραμματισμένο πότισμα.



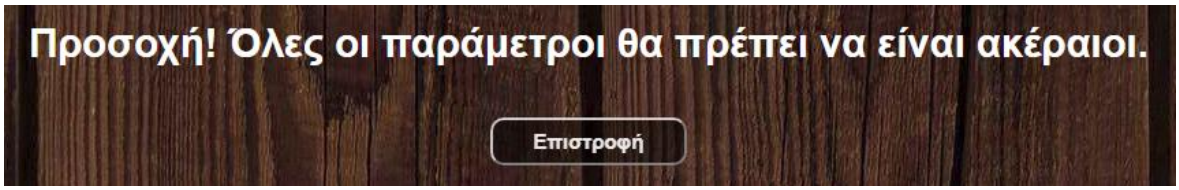
Εικόνα 4.10 Σύγκρουση Ποτίσματος

2. Σε περίπτωση διάρκειας μεγαλύτερης των 300 λεπτών, μικρότερης του 0 και αρνητικής.



Εικόνα 4.11 Μήνυμα Λανθασμένης Εισόδου

3. Σε περίπτωση που αυτό που εισάγει ο χρήστης δεν είναι ακέραιοι αριθμοί.



Εικόνα 4.12 Μήνυμα Λανθασμένης Εισόδου.

2. Προκαθορισμένο πότισμα.

Το πεδίο που αφορά το προκαθορισμένο πότισμα απεικονίζεται στην εικόνα παρακάτω και αφορά περιπτώσεις που ο χρήστης επιθυμεί κάποιον από τους χρόνους αυτούς και χρησιμοποιείται για αμεσότερη χρήση. Η παραπάνω περιορισμοί δεν λαμβάνονται υπόψιν, εκτός την σύγκρουση με κάποιο άλλο ήδη υπάρχων ή προγραμματισμένο πότισμα.

Εικόνα 4.13 Επιλογή Προκαθορισμένου Ποτίσματος

Για την επιλογή προκαθορισμένου ποτίσματος πατάμε απλώς πάνω στο κουμπί με την χρονική διάρκεια που επιθυμούμε.

4.5.2 ΠΡΟΓΡΑΜΜΑΤΙΣΜΕΝΟ ΠΟΤΙΣΜΑ

Η δεύτερη βασική επιλογή ενός ποτίσματος είναι αυτή που δεν αφορά το παρών αλλά το μέλλον. Με αυτήν ο χρήστης έχει την δυνατότητα να αποθήκευσει ποτίσματα που θέλει να γίνουν κατά την χρονική διάρκεια μιας εβδομάδας για μια μόνο φόρα. Για αυτού του τύπου την επιλογή θα μετάβουμε στο πεδίο της εφαρμογής που απεικονίζεται στην παρακάτω εικόνα.

Εικόνα 4.14 Επιλογή Προγραμματισμένου Ποτίσματος

Βήματα:

1. Επιλέγουμε την ημέρα.
2. Επιλέγουμε την ώρα έναρξης.
3. Επιλέγουμε την ώρα λήξης.
4. Πατάμε το κουμπί «Έναρξη».

Υποθέτουμε πως ο χρήστης εισάγει ένα πότισμα για την μέρα Πέμπτη με ώρα έναρξης 15:00 και ώρα λήξης 18:00.

Στην περίπτωση που η παραπάνω καταχώρηση υπάρχει ήδη ή οι ώρες συμπίπτουν με άλλες ήδη υπάρχουσας, ο χρήστης θα λάβει την παρακάτω απάντηση στον φυλλομετρητή.

Εικόνα 4.15 Σύγκρουση Ποτίσματος

Στην περίπτωση που η καταχώρηση που εισάγουμε δεν υπάρχει θα λάβουμε το παρακάτω μήνυμα επιβεβαίωσης.

Εικόνα 4.16 Επιβεβαίωση Προγραμματισμένου Ποτίσματος

Για την ακύρωση της καταχώρησης πατάμε πάνω στο κουμπί 

. Εάν δεν έχουμε πρόσβαση για κάποιο λόγο στην παραπάνω σελίδα ή εάν θέλουμε να δούμε πως η καταχώρηση έγινε επιτυχώς, πατάμε στο κουμπί «Προγραμματισμένο Πότισμα» της αρχικής σελίδας.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Από την διαδικασία εκπόνησης της παραπάνω πτυχιακής εργασίας, καθώς και την συγγραφή του κώδικα οπού χρησιμοποίησα για την υλοποίηση του αυτοματισμού, καταλήγω στα εξής συμπεράσματα:

- Γενικά

1. Το **διαδίκτυο** στις μέρες είναι ένα πολύ χρήσιμο εργαλείο το οποίο μπορεί να μας επιλύσει το μεγαλύτερο ποσοστό προβλημάτων ειδικά στον τομέα της πληροφορικής και της τεχνολογίας. Σχεδόν σε όλα τα ερωτήματα που μου δημιουργήθηκαν απευθύνθηκα σε αυτό.
2. Φτάνοντας στο τελικό στάδιο των σπουδών μου κατάλαβα ποσό σημαντικό είναι να παρακολουθεί κάνεις με αφοσίωση το **πρόγραμμα των μαθήματων** καθ' όλη την διάρκεια φοίτησης. Όλα τα εργαλεία που χρησιμοποίησα για την υλοποίηση του κώδικα μου είχαν διδαχθεί σαν μαθήματα, στα οποία εάν είχα δώσει μεγαλύτερη βάση θα μου ήταν πολύ πιο εύκολο και γρήγορο να δημιουργήσω αυτήν την πτυχιακή.
3. Η **μεθοδικότητα** από άνθρωπο σε άνθρωπο διαφέρει κατά πολύ χωρίς να σημαίνει πως κάποιος σκέφτεται λάθος.
4. Πολύ σημαντικό ρολό παίζει το **κοινωνικό δίκτυο** με το οποίο ερχόμαστε σε επαφή. Όταν ο κοινωνικός σου κύκλος απαρτίζεται από άτομα των ιδίων ενδιαφερόντων με τα δικά σου τότε υπάρχει κάτι σαν ανταγωνισμός που ο ένας πολεμάει να γίνει καλύτερος από τον άλλο.
Αυτό πιστεύω έχει σαν αποτέλεσμα να αναζητάμε εν συνεχεία το κάτι παραπάνω και να θέτουμε ποιοτικότερους και μεγαλύτερους στόχους.
5. Τέλος, θεωρώ πως τον σημαντικότερο ρολό για να γίνουν σχεδόν όλα από τα παραπάνω τον έχει **ο επιβλέπων καθηγητής**. Συγκεκριμένα είχα την τιμή να συνεργαστώ με τον κ. Βαρτζιώτη Φώτιο που συνεχώς με παρότρυνε και μου έδειχνε πως έχω την δυνατότητά να αντιμετωπίσω οποιοδήποτε πρόβλημα αντιμετώπιζα.

- Για την πτυχιακή

1. Υπάρχουν πολύ πιο απλούστεροι τρόποι να δημιουργηθεί ο παραπάνω αυτοματισμός με την χρήση διαφορετικών τεχνολογιών διαδικτύου και μικροεπεξεργαστών.

2. Υπάρχουν έτοιμες βιβλιοθήκες που θα μπορούσα να χρησιμοποιήσω για να αποφύγω μεγάλα κομμάτια κώδικα, όπως αυτό της σύγκρουσης των ποτισμάτων που έγραψα μονός

μου. Παρόλα αυτά λόγο μικρής τριβής μου με την Java επέλεξα να κάνω την συγγραφή του κώδικα βάση της δικιάς μου λογικής οπού κατανοούσα καλυτερα.

3. Όσο αναφορά το υλικό κομμάτι που χρησιμοποίησα (Raspberry Pi 3) σαν συσκευή για να ελέγχω το μοτέρ ποτίσματος, μπορώ να πω πως ήταν αρκετά εύχρηστο και παρέχει πολλές δυνατότητες που θα μπορούσα να αξιοποιήσω μελλοντικά.

ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΙΜΟΤΗΤΕΣ

Όπως είπαμε στην αρχή αυτής της εργασίας η διαδικτυακή εφαρμογή δημιουργήθηκε βάση συγκεκριμένο προσωπικών αναγκών. Έτσι οι λειτουργίες που την απαρτίζουν είναι συγκεκριμένες και τροποποιημένες για ιδιαίτερη χρήση. Για παράδειγμα ο μέγιστος χρόνος ποτίσματος είναι οι 5 ώρες διότι έπειτα από αυτήν την χρονική περίοδο το πηγάδι που τροφοδοτεί το δίκτυο παύει να έχει νερό. Μελλοντικός στόχος για την επεκτασιμότητα του συστήματος είναι η ευρύτερη χρηστική δυνατότητα με την πρόσθεση των παρακάτω λειτουργιών.

- **Αισθητήρας υγρασίας:** Σε περίπτωση που ξεκινάει βροχόπτωση να σταματάει η διαδικασία ποτίσματος.
- **Αισθητήρας πίεσης:** Επιβεβαίωση προς τον χρήστη πως το δίκτυο δεν έχει κάποια διαρροή και το νερό έχει φτάσει όντως στην επιθυμητή καλλιέργεια.
- **Τροποποίηση λογισμικού εφαρμογής της εφαρμογής:** Τροποποίηση του μενού προγραμματισμένου ποτίσματος για να μπορεί ο χρήστης να εισάγει την επιθυμητή ώρα έναρξης και διάρκεια σε όλο το φάσμα των λεπτών της ώρας.
- **Δημιουργίας εφαρμογής apk:** Δημιουργία εφαρμογής για την χρήση από φορητές συσκευές (smartphone,tablet).
- **Απεικόνιση υπολειπομένου χρόνου ποτίσματος:** Με αυτόν το τρόπο ο χρήστης θα γνωρίζει άμεσα πόσος χρόνος υπολείπετε για να ολοκληρωθεί το πότισμα.
- **Ειδοποίηση κατά την διακοπή ρεύματος:** Όταν θα διακόπτεται το ρεύμα του μοτέρ ο χρήστης θα ενημερώνεται με σχετικό μήνυμα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1]https://translate.google.gr/translate?hl=el&sl=en&u=https://en.wikipedia.org/wiki/Raspberry_Pi&prev=search

[2] https://projects.raspberrypi.org/en/projects/build-your-own-weather-station

[3] https://osmc.tv/about/

[4] https://randomnerdtutorials.com/11-clever-uses-for-your-raspberry-pi/

[5] https://repository.kallipos.gr/bitstream/11419/3976/1/01_chapter_7.pdf

[6] https://ti-einai.gr/javascript/

[7] https://ti-einai.gr/html/

[8] https://el.wikipedia.org/wiki/MySQL

[9]https://ti-einai.gr/mysql/

[10]https://www2.dmst.aueb.gr/dds/ism/oo/indexw.htm

[11]https://en.wikipedia.org/wiki/Apache_Tomcat

[12]https://www.youtube.com/watch?v=kkQOm02kep0&t=29s