ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Ανάπτυξη εφαρμογών χρησιμοποιώντας ανοιχτές προγραμματιστικές διεπαφές συστημάτων κοινωνικής δικτύωσης.

Application development based on open APIs of social networking systems.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Λαμπρινίδη Αργυρώ

Επιβλέποντες Καθηγητές: Λάλης Σπυρίδων-Γεράσιμος
Αναπληρωτής Καθηγητής Π.Θ.

Δασκαλοπούλου Ασπασία
Επίκουρος Καθηγήτρια Π.Θ.

Βόλος, Σεπτέμβριος 2012

Ευχαριστίες

Με την περάτωση της παρούσας διπλωματικής εργασίας θα ήθελα να ευχαριστήσω τον κ. Λάλη για την πολύτιμη βοήθεια και συνεχή καθοδήγηση που μου πρόσφερε όλους αυτούς τους μήνες, όπως και για την εμπιστοσύνη που μου έδειξε. Επίσης θα ήθελα να ευχαριστήσω τους φίλους μου και τον Γιώργο για την στήριξή τους όλα αυτά τα χρόνια των προπτυχιακών σπουδών μου και την βοήθεια τους στην ολοκλήρωση της διπλωματικής μου εργασίας. Τέλος, οφείλω ένα μεγάλο ευχαριστώ την οικογένειά μου που είναι πάντα δίπλα μου σε κάθε προσπάθειά μου.

Περίληψη

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η διαχείριση και διασύνδεση αισθητήρων και ελεγκτών. Ως πλατφόρμα επικοινωνίας των πόρων ορίστηκε το social media Twitter και αναπτύχθηκε ένα πρωτόκολλο επικοινωνίας με αυτούς. Οι πόροι εμφανίζονται ως Twitter endpoints και η αλληλεπίδραση μαζί τους γίνεται με μηνύματα σε μορφή κειμένου φυσικής γλώσσας ώστε η επικοινωνία αυτή να φαίνεται οικεία στο χρήστη. Αναπτύχθηκε επίσης ένα απλό και βολικό cloud-based Περιβάλλον Προγραμματισμού Αυτοματισμών και διαχείρισης εφαρμογών αυτοματισμού , το οποίο δίνει την δυνατότητα στο χρήστη να δημιουργεί και να διαχειρίζεται τις δικές του εφαρμογές αυτοματισμού, συνδυάζοντας τους πόρους που έχει στη διάθεσή του, χωρίς ωστόσο να απαιτείται από εκείνον η εγκατάσταση ειδικού λογισμικού για την πρόσβαση σε αυτό. Για την διευκόλυνση της δημιουργίας των εφαρμογών αυτοματισμών του χρήστη αναπτύχθηκε μια Γλώσσα Προγραμματισμού Αυτοματισμών με περιορισμένες και πιο απλές δομές σε σχέση με τις γλώσσες προγραμματισμού γενικής χρήσης καθώς και μια γραφική διεπαφή χρήστη.

Abstract

The subject of the present thesis is the management and the interconnection of sensors and controllers. Social media Twitter has been defined as a resources communication platform, while a protocol of interaction with them has been developed as well. The resources appear as Twitter endpoints and the interaction with them takes place through messages with the form of a natural language text, so that this interaction seems more familiar to the user. Moreover, a simple and convenient cloud-based Automation Programming Environment and Management of Automation Applications have been created, which makes the user able to create and deal with his own automation applications, combining the available resources, without being asked to install a specific software in order to access it. Furthermore, an Automation Programming Language has been built for the facilitation of the user's construction of automation applications, which includes limited and less complicated structures than the general use programming languages, as well as a user graphical interface.

Κατάλογος Πινάκων

Πίνακας 1: Κοινές Υποστηριζόμενες Εντολές 22
Πίνακας 2: Υποστηριζόμενες Εντολές Κάμερας 25
Πίνακας 3: Υποστηριζόμενες Εντολές Αισθητήρα Κίνησης 28
Πίνακας 4: Υποστηριζόμενες Εντολές Αισθητήρα Θερμοκρασίας 32
Πίνακας 5: Τύποι Δεδομένων 35
Πίνακας 6: Αριθμητικοί Τελεστές 38
Πίνακας 7: Σχεσιακοί Τελεστές 38
Πίνακας 8: Λογικοί Τελεστές 39

Κατάλογος Εικόνων

Εικόνα 1: Επικοινωνία Χρήστη-Πόρων 15
Εικόνα 2: Επικοινωνία μεταξύ των Twitter Accounts. 16
Εικόνα 3:Αρχιτεκτονική Συστήματος Διαχείρισης Εφαρμογών 17
Εικόνα 4: Επικοινωνία Εφαρμογών Χρήστη-Πόρων 18
Εικόνα 5: Επικοινωνία του Runtime με το υπόλοιπο Σύστημα 19
Εικόνα 6: Αρχιτεκτονική Υλοποίησης Πόρων 20
Εικόνα 7: Ακολουθιακό Διάγραμμα Πόρου για Upload Αρχείου στο Google Docs. 23
Εικόνα 8: Ακολουθιακό Διάγραμμα για την Εντολή “capture image”. 26
Εικόνα 9: Εμβέλειες Μεταβλητών 36
Εικόνα 10: Στάδια της Διαδικασίας Μεταγλώττισης 49
Εικόνα 11: Διαδικασία Δημιουργίας Λεκτικού και Συντακτικού Αναλυτή 50
Εικόνα 12: Ακολουθιακό Διάγραμμα για τη διαδικασία ανταλλαγής του xml αρχείου 59
Εικόνα 13: UML Διάγραμμα Κλάσεων που υλοποιούν την Κάμερα 66
Εικόνα 14: UML Διάγραμμα Κλάσεων που υλοποιούν τον Αισθητήρα Κίνησης 67
Εικόνα 15: UML Διάγραμμα των Κλάσεων που υλοποιούν τον Αισθητήρα Θερμοκρασίας .. 68
Εικόνα 16: UML Διάγραμμα Κλάσεων που υλοποιούν τον Μεταγλωττιστή 69
Εικόνα 17: UML Διάγραμμα Κλάσεων που υλοποιούν το Runtime 70

Πίνακας Περιεχομένων

1 Εισαγωγή 10
2 Σχετικές Εργασίες 12
2.1 Το Twitter ως Πλατφόρμα 12
2.2 Τεχνολογίες Υποστήριξης Εφαρμογών Αυτοματισμού και End-User Programming 13
3 Πλατφόρμα Επικοινωνίας 14
4 Runtime και Servlets 16
5 Πόροι και Πρωτόκολλο Επικοινωνίας 20
5.1 Κάμερα 24
5.2 Αισθητήρας Κίνησης 27
5.3 Αισθητήρας Θερμοκρασίας 31
6 Γλώσσα Προγραμματισμού 35
6.1 Τύποι Δεδομένων και Τελεστές 35
6.2 Εντολές 39
6.3 Δομές Ελέγχου Προγράμματος 40
6.4 Παραδείγματα Εφαρμογών 42
6.4.1 Εφαρμογή I 42
6.4.2 Εφαρμογή II 43
6.4.3 Εφαρμογή III 44
6.4.4 Εφαρμογή IV 45
6.4.5 Εφαρμογή V 46
6.4.6 Εφαρμογή VI 47
7 Μεταγλωττιστής 49
7.1 Λεκτικός και Συντακτικός Αναλυτής 50
7.2 Σημασιολογικός Αναλυτής 51
7.3 Παραγωγή τελικού κώδικα 52
8 Διεπαφή Χρήστη 54
8.1 Αρχική σελίδα 55
8.2 Εισαγωγή/ Διαγραφή Πόρου-Twitter endpoint 56
8.3 Δημιουργία νέας εφαρμογής 57
8.4 Τροποποίηση εφαρμογής 60

8.5 Διαγραφή εφαρμογής 61
8.6 Εκκίνηση/Τερματισμός εφαρμογής 62
8.7 Configuration 63
9 Επίλογος 64
9.1 Συμπεράσματα 64
9.2 Βελτιώσεις και Επεκτάσεις 64
10 Appendix 65
11 Αναφορές 71

1 Εισαγωγή

Ο αριθμός των αισθητήρων και των ελεγκτών που υπάρχουν γύρω μας ολοένα και αυξάνεται. Λόγω της αύξησης αυτής υπάρχει προοπτική για πολλές χρήσιμες εφαρμογές που πιθανώς μπορεί να σκεφτεί ο ίδιος ο χρήστης. Οι εφαρμογές αυτές στόχο έχουν την εξυπηρέτηση των αναγκών των χρηστών και την βελτίωση της ποιότητας ζωής τους. Οι ανάγκες κάθε χρήστη είναι μοναδικές και δεδομένου αυτού θα ήταν ενδιαφέρον να του δοθεί η δυνατότητα δημιουργίας εφαρμογών αυτοματισμών συνδυάζοντας τους πόρους που έχει στη διάθεσή του.

Η ευέλικτη διαχείριση και διασύνδεση των πόρων που ο χρήστης έχει στη διάθεσή του είναι τα βασικά στοιχεία με τα οποία θα επιτευχθεί πρόσθετη λειτουργικότητα αλλά ταυτόχρονα αποτελούν πρόβλημα. Σημαντικό χαρακτηριστικό των πόρων είναι η ετερογένεια. Πόροι διαφορετικής φύσης, με διαφορετικές λειτουργίες καλούνται να συνδυαστούν για τη δημιουργία μιας εφαρμογής αυτοματισμού.

Η διαχείριση των πόρων και η δημιουργία των εφαρμογών αυτοματισμών προϋποθέτει την επικοινωνία με τους πόρους. Ωστόσο για την πρόσβαση στους πόρους δεν μπορεί να απαιτείται η εγκατάσταση ειδικού λογισμικού στις συσκευές του χρήστη. Επίσης, οι συμβατικές γλώσσες προγραμματισμού δεν ενδείκνυνται για την δημιουργία των εφαρμογών αυτοματισμών από τους ίδιους τους χρήστες. Οι συμβατικές γλώσσες προγραμματισμού χρησιμοποιούνται συνήθως από προγραμματιστές ή από άτομα με μεγάλη εξοικείωση στον προγραμματισμό. Έτσι, θα ήταν αρκετά δύσκολο για τους απλούς χρήστες να δημιουργήσουν τις εφαρμογές που επιθυμούν χρησιμοποιώντας τις πολύπλοκες για εκείνους γλώσσες προγραμματισμού.

Στόχος αυτής της διπλωματικής εργασίας είναι να κάνει ένα βήμα προς αυτή την κατεύθυνση. Πιο συγκεκριμένα αναπτύχθηκε ένα σύστημα βάση του οποίου αποτελεί ο απομακρυσμένος έλεγχος και η διαχείριση των πόρων καθώς και η δημιουργία εφαρμογών αυτοματισμών από τους χρήστες. Δίνεται στο χρήστη η δυνατότητα υλοποίησης των δικών του αυτοματισμών με συνδυασμό των πόρων που έχει στη διάθεσή του. Για να επιτευχθεί κάτι τέτοιο αναπτύχθηκε ένα πρωτόκολλο επικοινωνίας σε μορφή κειμένου, που να φαίνεται φυσικό στον χρήστη. Σημαντικό είναι επίσης ότι η πλατφόρμα επικοινωνίας που χρησιμοποιήθηκε δεν απαιτεί την εγκατάσταση ειδικού λογισμικού αφού είναι η πλατφόρμα του Twitter. Επίσης δημιουργήθηκαν ένα Περιβάλλον Προγραμματισμού Αυτοματισμών και μια Γλώσσα Προγραμματισμού τα οποία δίνουν την δυνατότητα στο χρήστη να “προγραμματίσει” τις εφαρμογές του και σε συνδυασμό με το πρωτόκολλο επικοινωνίας ξεπερνούν το πρόβλημα της ετερογένειας των πόρων. Η Γλώσσα Προγραμματισμού είναι απλή και φιλική στο χρήστη αφού το σύστημα που δημιουργήθηκε δεν απευθύνεται αποκλειστικά σε προγραμματιστές αλλά σε ανθρώπους με βασικές γνώσεις στον προγραμματισμό. Τέλος, παρέχεται μια γραφική διεπαφή για την ανάπτυξη και την διαχείριση των εφαρμογών.

Στα κεφάλαια που ακολουθούν θα γίνει περιγραφή του συστήματος που δημιουργήθηκε. Πιο συγκεκριμένα, στο Κεφάλαιο 2 θα γίνει αναφορά εργασιών σχετικών με το αντικείμενο της παρούσας διπλωματικής εργασίας. Στο Κεφάλαιο 3 γίνεται περιγραφή και ανάλυση της πλατφόρμας επικοινωνίας που καθορίστηκε. Στο Κεφάλαιο 4 περιγράφεται η αρχιτεκτονική του συστήματος που δημιουργήθηκε και ειδικότερα του βασικότερου συστατικού μέρους (component) αυτού , του Runtime . Στο Κεφάλαιο 5 αναφέρεται η αρχιτεκτονική του

συστήματος των πόρων και το πρωτόκολλο επικοινωνίας με αυτούς. Στο Κεφάλαιο 6 περιγράφεται η Γλώσσα Προγραμματισμού Αυτοματισμών που δημιουργήθηκε ενώ στο Κεφάλαιο 7 γίνεται αναφορά στον Μεταγλωττιστή που χρησιμοποιείται για την μετατροπή της Γλώσσας Προγραμματισμού Αυτοματισμών σε εκτελέσιμο πρόγραμμα (Java κλάση) . Στο Κεφάλαιο 8 περιγράφεται η γραφική διεπαφή χρήστη και οι δυνατότητές της. Τέλος, στο Κεφάλαιο 9 είναι ο επίλογος ενώ στα Κεφάλαια 10 και 11 το Appendix και οι Αναφορές αντίστοιχα.

2 Σχετικές Εργασίες

Το σύστημα διαχείρισης πόρων που δημιουργήθηκε βασίστηκε σε δύο κύρια χαρακτηριστικά πάνω στα οποία έγινε και η αναζήτηση σχετικών εργασιών. Το ένα χαρακτηριστικό είναι η χρήση του Twitter ως πλατφόρμα επικοινωνίας, ενώ το δεύτερο είναι ο προγραμματισμός εφαρμογών αυτοματισμών από τον χρήστη.

2.1 Το Twitter ως Πλατφόρμα

Τα social media ενθαρρύνουν την ανάπτυξη client εφαρμογών και αυτό φαίνεται από τα εκατοντάδες API που παρέχονται στους προγραμματιστές. Ως αποτέλεσμα είναι η ανάπτυξη αναρίθμητων εφαρμογών στηριζόμενων σε ένα ή περισσότερα social media κάθε φορά, που επιτελούν διαφορετικές λειτουργίες και ικανοποιούν ανάγκες του σύγχρονου χρήστη. Το Twitter έχει χρησιμοποιηθεί ευρέως ως πλατφόρμα επικοινωνίας ανθρώπων και ως πλατφόρμα άντλησης πληροφοριών, αλλά δεν είναι τόσο διαδεδομένη η χρήση του ως πλατφόρμα επικοινωνίας με μηχανήματα/συσκευές.

Υπάρχουν μεμονωμένες προσπάθειες απομακρυσμένου ελέγχου συσκευών μέσω Twitter. Για να γίνει όμως κάτι τέτοιο εφικτό είναι απαραίτητος ο προγραμματισμός της συσκευής ώστε να δέχεται μηνύματα μέσα από το Twitter και να τα "αποκωδικοποιεί" για να αντιληφθεί τη λειτουργία που πρέπει να επιτελέσει. Το γεγονός αυτό είναι και ο μεγαλύτερος ανασταλτικός παράγοντας ανάπτυξης εφαρμογών για επικοινωνία με συσκευές. Παρ' όλα αυτά υπάρχουν στο Internet αρκετά παραδείγματα για το πώς μια συσκευή, όπως για παράδειγμα ένα πλυντήριο ρούχων, θα μπορούσε να ρυθμιστεί ώστε να αλληλεπιδρά με το χρήστη μέσω μηνυμάτων του Twitter.

Μια εφαρμογή που υλοποιεί την επικοινωνία χρήστη-συσκευής μέσω social media είναι το **Twingz [1]** . Ο χρήστης κάνει log in στην εφαρμογή μέσω του Account που έχει στο Twitter ή στο Facebook και στη συνέχεια μπορεί να επικοινωνήσει με τις συσκευές που έχει στην κατοχή του, εφόσον αυτές υποστηρίζουν την εφαρμογή Twingz. Το Twingz ωστόσο δεν παρέχει δυνατότητα για δημιουργία αυτοματισμών από τον χρήστη.

2.2 Τεχνολογίες Υποστήριξης Εφαρμογών Αυτοματισμού και End-User Programming

Ένα σύστημα που βρίσκεται ήδη στην αγορά είναι το **Qbus [2]**. Το συγκεκριμένο σύστημα υποστηρίζει έξυπνα περιβάλλοντα, κυρίως σε επίπεδο σπιτιού. Όπως αναφέρεται, ο χρήστης μπορεί να διαχειρίζεται τους πόρους που διαθέτει, αλλά και να δημιουργεί μικρούς αυτοματισμούς χρησιμοποιώντας λογικές πράξεις, καθώς και λογικές αναλογικές λειτουργίες μέσα από ένα γραφικό περιβάλλον.

Επίσης η **Nokia [3]** έχει κάνει εκτενή έρευνα πάνω σε τέτοιες τεχνολογίες και έχει αναπτύξει ένα σύστημα διαχείρισης έξυπνων περιβαλλόντων με χρήση κινητών τηλεφώνων. Έχει αναπτυχθεί μια πλατφόρμα ανάπτυξης εφαρμογών σε δύο επίπεδα (middleware και εργαλεία χρήστη) που επιτρέπει στους χρήστες να δημιουργούν και να εκτελούν στο κινητό τους τηλέφωνο τις δικές τους εφαρμογές.

Ένα ακόμα σύστημα που υποστηρίζει την δημιουργία αυτοματισμών από το χρήστη σε έξυπνα περιβάλλοντα είναι το **PlayingWithTheBits [4]**. Το σύστημα αυτό παρέχει ένα σχετικά απλό γραφικό περιβάλλον ανάπτυξης αυτοματισμών το οποίο παρά την ετερογένεια των πόρων μπορεί επιτύχει τον συνδυασμό τους. Οι λειτουργίες κάθε πόρου αναπαρίστανται με κομμάτια παζλ και έτσι η δημιουργία σύνθετων αυτοματισμών γίνεται εύκολα κατανοητή στο χρήστη.

Το **Patch Panel [5]** επιτρέπει control-flow διαδραστικότητα σε έξυπνα περιβάλλοντα. Στο σύστημα υπάρχει ένας κεντρικός διαχειριστής (Event Heap) ο οποίος υποστηρίζει λειτουργίες publisher /subscriber. Έτσι δέχεται νέα events από τους εγγεγραμμένους clients και τα παραλαμβάνουν οι clients τους οποίους αναφέρονται. Η διαδικασία αίτησης-παραλαβής νέου event γίνεται στην λογική των tuplespaces. Το πρόβλημα της ετερογένειας μεταξύ των συσκευών που επιθυμεί ο χρήστης να επικοινωνήσουν επιλύεται με μηχανές καταστάσεων και δυνατότητες αξιολόγησης εξίσωσης.

Το **TeC [6]** είναι μια ακόμα τεχνολογία ανάπτυξης εφαρμογών σε έξυπνα περιβάλλοντα από τον χρήστη. Τα συστήματα στο TeC βασίζονται στη αυτονομία των αντικειμένων που με μια πιο αφαιρετική ματιά δρουν ως ομάδες χωρίς κεντρικό διαχειριστή. Το TeC προσφέρει μια σχεδιαστική γλώσσα προγραμματισμού στο χρήστη ώστε να ελέγχει και να διαχειρίζεται έξυπνα περιβάλλοντα.

3 Πλατφόρμα Επικοινωνίας

Η εφαρμογή που αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας συνδυάζει τις δυνατότητες που παρέχουν τα social media, και πιο συγκεκριμένα το **Twitter [13]** , το **Flickr [14]** και το **Google Docs [15]**, και τις πληροφορίες που μπορούν να αντληθούν από διάφορους αισθητήρες και ελεγκτές. Ο χρήστης μπορεί να χειρίζεται ένα σύνολο πόρων που έχει στην διάθεσή του και να δημιουργεί αυτοματισμούς για να καλύψει τις ανάγκες του. Η επικοινωνία με τους διαθέσιμους πόρους γίνεται μέσω της πλατφόρμας του Twitter.

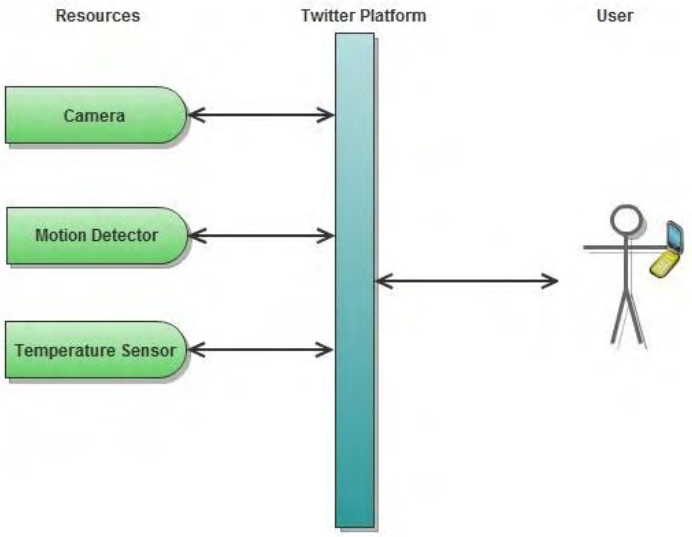
Ο λόγος που επιλέχθηκε το Twitter ως μέσω επικοινωνίας είναι αφενός γιατί αποτελεί μια σταθερή και αξιόπιστη πλατφόρμα και αφετέρου γιατί με αυτό τον τρόπο δίνεται η δυνατότητα επικοινωνίας και διαχείρισης χωρίς να απαιτείται η εγκατάσταση ειδικού λογισμικού. Ο χρήστης μπορεί συνεπώς να συνδέεται στο Twitter από τον υπολογιστή, το κινητό τηλέφωνο ή οποιοδήποτε άλλο τερματικό και να επικοινωνεί με τους πόρους του.

Κάθε πόρος έχει ένα Twitter Account και έχει προγραμματιστεί ώστε να λαμβάνει τα μηνύματα που του αποστέλλονται. Οι πόροι, όπως και το Περιβάλλον Προγραμματισμού Αυτοματισμών, "εμφανίζονται" δηλαδή ως Twitter endpoints. Έτσι, χρησιμοποιείται το Twitter και όλες οι δυνατότητες που παρέχει στους χρήστες του για να επιτευχθεί η επιθυμητή επικοινωνία. Οι πόροι επικοινωνούν με τους χρήστες και με το Περιβάλλον Προγραμματισμού Αυτοματισμών όπως ακριβώς επικοινωνούν όλοι οι υπόλοιποι Twitter Users.

Δε θα ήταν ωστόσο σωστό οι πόροι να λαμβάνουν μηνύματα από οποιοδήποτε Twitter User. Μόνο συγκεκριμένα Twitter Accounts θα πρέπει να μπορούν να επικοινωνούν με τους πόρους για να διασφαλιστεί η ασφάλεια και η σωστή λειτουργία τους. Για το λόγο αυτό, η ανταλλαγή μηνυμάτων γίνεται με Direct Messages (και όχι με tweets) όπου είναι απαραίτητο το αμοιβαίο "follow" μεταξύ των δύο Twitter Accounts. Με τον τρόπο αυτό μόνο εξουσιοδοτημένοι χρήστες μπορούν να επικοινωνούν με τον εκάστοτε πόρο.

Πολύ σημαντική ιδιαιτερότητα του συστήματος που αναπτύχθηκε είναι ότι τα μηνύματα που ανταλλάσσονται μεταξύ των χρηστών και των πόρων είναι σε φυσική γλώσσα. Ο χρήστης αλληλεπιδρά με τους πόρους σχεδόν όπως μιλάει σε ανθρώπους. Ωστόσο τα μηνύματα που ανταλλάσσονται μέσω του Twitter μπορούν να χαρακτηριστούν ως εντολές αφού υποδηλώνουν την επιθυμία του χρήστη για την εκτέλεση κάποιας ενέργειας. Κάθε πόρος υποστηρίζει μια σειρά εντολών ανάλογα με τη φύση του και τις δυνατότητές του. Κατά την άφιξη λοιπόν ενός νέου μηνύματος γίνεται αποκωδικοποίηση ώστε να αντιληφθεί ο πόρος αν πρόκειται για έγκυρη εντολή ή όχι.

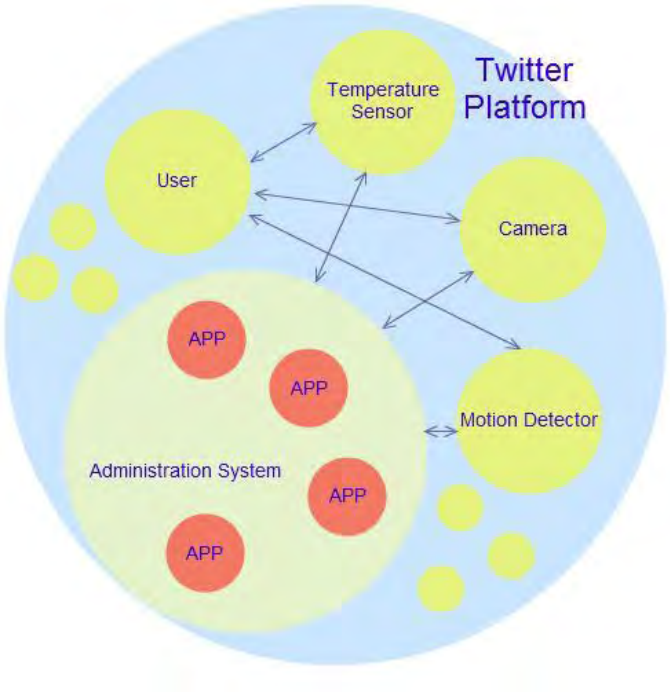
Στέλνοντας μια εντολή με Direct Message ο πόρος τη λαμβάνει, την επεξεργάζεται και στέλνει και εκείνος την δική του απάντηση. Στην Εικόνα 1 φαίνεται σχηματικά η επικοινωνία ενός χρήστη με τους πόρους.



Εικόνα 1: Επικοινωνία Χρήστη-Πόρων

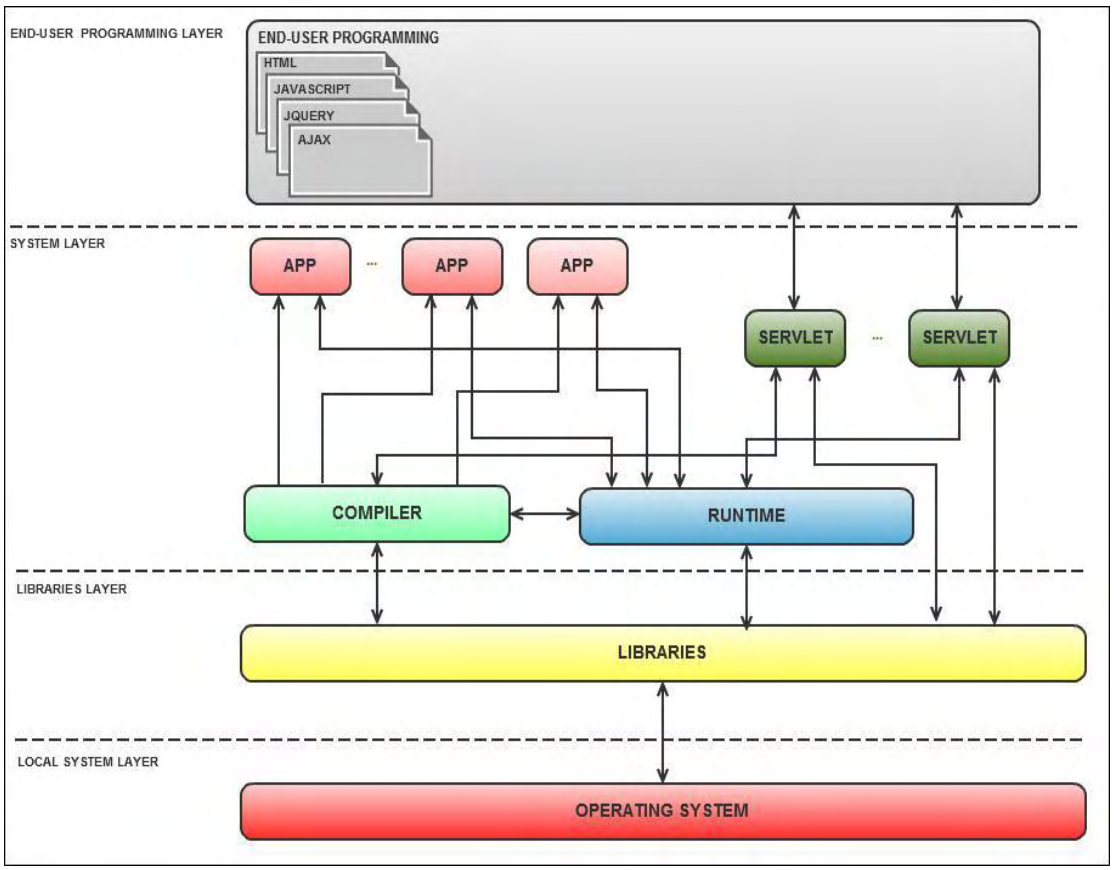
4 Runtime και Servlets

Κάθε χρήστης μπορεί να επικοινωνήσει άμεσα με κάποιο πόρο, στέλνοντάς του μια εντολή στο Twitter αλλά όπως έχει ήδη αναφερθεί δίνεται η δυνατότητα να προγραμματίζει και τους δικούς του αυτοματισμούς εφαρμογών. Για να επιτευχθεί η επικοινωνία των αυτοματισμών με τους πόρους το Περιβάλλον Προγραμματισμού Αυτοματισμών αποτελεί επίσης ένα Twitter endpoint. Οι αυτοματισμοί επικοινωνούν, δηλαδή, με τους πόρους μέσω του Twitter Account του Περιβάλλοντος Προγραμματισμού Αυτοματισμών, το screen name του οποίου είναι “main_process”. Όλες οι εφαρμογές του χρήστη εκτελούν τις λειτουργίες τους και “εμφανίζονται” διαμέσου του Twiiter Account του Περιβάλλοντος Προγραμματισμού Αυτοματισμών .



Εικόνα 2: Επικοινωνία μεταξύ των Twitter Accounts. Με μπλέ χρώμα αναπαρίσταται η πλατφόρμα του Twiiter ενώ με κίτρινο τα Twitter Accounts. Κάθε πόρος έχει ένα Twitter Account όπως επίσης και το Περιβάλλον Προγραμματισμού Αυτοματισμών. Με τα βέλη συμβολίζεται η εφικτή ανταλλαγή Direct Message. Με πορτοκαλί χρώμα συμβολίζονται οι εφαρμογές του χρήστη οι οποίες επικοινωνούν με τον υπόλοιπο κόσμο του Twitter διαμέσου του Twitter Account του Περιβάλλοντος Προγραμματισμού Αυτοματισμών.

Η αρχιτεκτονική του συστήματος που αναπτύχθηκε φαίνεται στην Εικόνα 3. Βάση του συστήματος που αναπτύχθηκε αποτελεί το λειτουργικό σύστημα του υπολογιστή και οι απαραίτητες βιβλιοθήκες (**JavaCC [11]**, Java βιβλιοθήκες των **Twitter API [7]** και **Google Docs API [9]** κτλ). Πάνω σε αυτά στηρίχθηκε η δημιουργία όλων των υπόλοιπων εξαρτημάτων (components). Στο αμέσως επόμενο επίπεδο βρίσκονται ο Μεταγλωττιστής, μέσω του οποίου Γλώσσα Προγραμματισμού Αυτοματισμών μεταφράζεται σε γλώσσα Java, το Runtime, και τα Java Servlets που χρησιμοποιούνται στην επικοινωνία της HTML σελίδας διεπαφής χρήστη. Τέλος στο ανώτατο επίπεδο βρίσκεται η γραφική διεπαφή χρήστη (HTML σελίδα διαχείρισης).

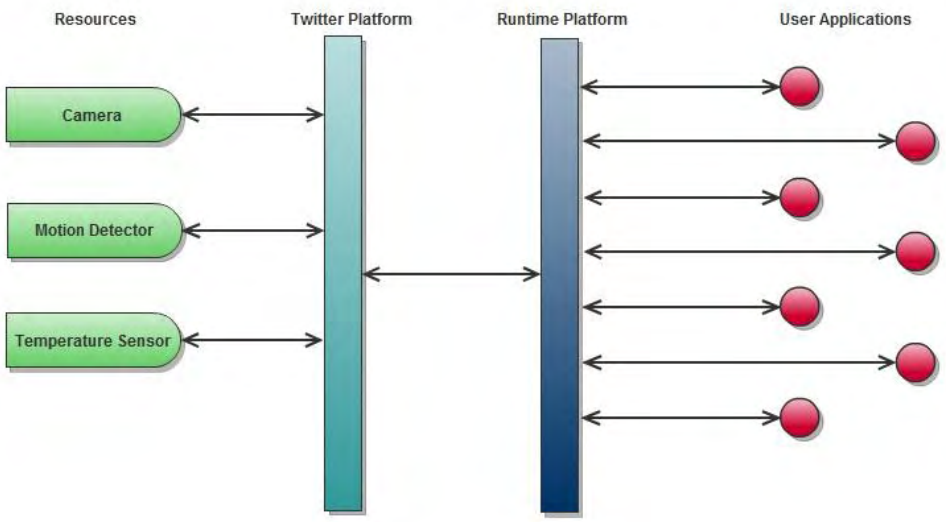


Εικόνα 3:Αρχιτεκτονική Συστήματος Διαχείρισης Εφαρμογών

RUNTIME

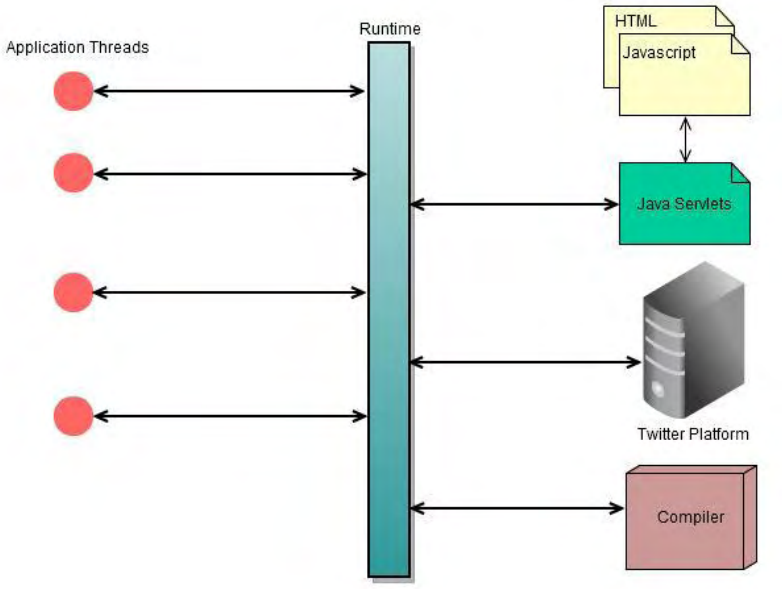
Ιδιαίτερης σημασίας συστατικό μέρος του Περιβάλλοντος Προγραμματισμού Αυτοματισμών που υλοποιήθηκε αποτελεί η Runtime βιβλιοθήκη που σχεδιάστηκε για τη ρύθμιση ης λειτουργίας του υπόλοιπου συστήματος. Η κλάση MyRuntime.java είναι ο κυρίως διαχειριστής του Περιβάλλοντος Προγραμματισμού Αυτοματισμών που αναπτύχθηκε. Όλα τα υπόλοιπα μέρη του συστήματος, δηλαδή οι εφαρμογές αυτοματισμών του χρήστη, τα Java Servlets που συνδέουν τη γραφική διεπαφή χρήστη με το υπόλοιπο σύστημα, και ο Μεταγλωττιστής που μετατρέπει τους αυτοματισμούς του χρήστη σε εκτελέσιμα προγράμματα, είναι απαραίτητο να επικοινωνούν με το Runtime ώστε να επιτελέσουν τη λειτουργικότητά τους.

Η ζωτικής σημασίας για το σύστημα που αναπτύχθηκε, επικοινωνία με το Twitter είναι αρμοδιότητα του Runtime. Ουσιαστικά το Runtime αποτελεί το Twitter endpoint του Περιβάλλοντος Προγραμματισμού Αυτοματισμών. Στην Εικόνα 4 φαίνεται σχηματικά η διαδρομή της πληροφορίας από τις εφαρμογές στους πόρους και αντίστροφα.



Εικόνα 4: Επικοινωνία Εφαρμογών Χρήστη-Πόρων

Το Runtime όντας κύριος διαχειριστής του Περιβάλλοντος Προγραμματισμού εφαρμογών συνδέει όλα τα συστατικά μέρη του συστήματος ώστε να επιτευχθεί η επιθυμητή λειτουργικότητα. Στην Εικόνα 5 αναπαριστάται το Runtime και τα μέρη του συστήματος με τα οποία επικοινωνεί.



Εικόνα 5: Επικοινωνία του Runtime με το υπόλοιπο Σύστημα

Τα Servlets και ο Μεταγλωττιστής επιτυγχάνουν την επικοινωνία με το Runtime μέσω sockets ενώ οι εφαρμογές συμπεριλαμβάνουν την κλάση στις βιβλιοθήκες και αποκτούν πρόσβαση στις μεθόδους της χρησιμοποιώντας την έκφραση Runtime.<μέθοδος> (Η κλάση MyRuntime.java έχει οριστεί ως public κλάση όπως και οι εσωτερικές της μέθοδοι οπότε η πρόσβαση με τον τρόπο αυτό είναι εφικτή).

Η έναρξη και ο τερματισμός μιας εφαρμογής καθώς και η επικοινωνία με το Twitter είναι αρμοδιότητες της κλάσης MyRuntime.java. Πιο συγκεκριμένα, όταν μια εφαρμογή θέλει να στείλει ένα Direct Message καλεί την μέθοδο sengMessage() της κλάσης Runtime.java όπου το μήνυμα/εντολή παίρνει την τελική του μορφή (<appname> <command> <ramdon_int>) και αποστέλλεται στον Twitter User που έχει οριστεί από την εφαρμογή. Κατά τη λήψη ενός μηνύματος γίνεται αποκωδικοποίηση για να "αντιληφθεί" το Runtime σε ποια εφαρμογή αναφέρεται το μήνυμα ώστε να το προωθήσει σε αυτή. Για την προώθηση του μηνύματος το Runtime καλεί τη μέθοδο receiveNewMessage() της εκάστοτε εφαρμογής αυτοματισμού.

Επίσης το Runtime εξυπηρετεί όλες τις ανάγκες του συστήματος για σύνδεση στο Twitter, δηλαδή εκτός από την αποστολή και λήψη των Direct Message, υπάρχουν μέθοδοι για εύρεση των followers του Twitter Account του συστήματος διαχείρισης αλλά και για την εύρεση όσων από αυτούς ακολουθεί, για εύρεση ύπαρξης αμοιβαίου follow και τέλος, για follow ή unfollow ενός Twitter Account.

Οι εφαρμογές αυτοματισμών του χρήστη επικοινωνούν με το Runtime είτε για αποστολή ή λήψη Direct Message, είτε για έναρξη ή τερματισμό της εφαρμογής αυτής. Ωστόσο οι ενέργειες για έναρξη ή τερματισμό κάποιας εφαρμογής, καθώς επίσης και follow ή unfollow

κάποιου Twitter User πυροδοτούνται από το χρήστη μέσω της HTML σελίδας διεπαφής. Έτσι η ροή των ενεργειών είναι HTML σελίδα -> Servlet -> Runtime.

Το Runtime περιβάλλον υλοποιείται από τις κλάσεις SocketMessage.java, MyRuntime.java και από τη διεπαφή AppInterface.java. Η κλάση SocketMessage.java χρησιμοποιείται για την επικοινωνία των Servlets με το Runtime. Τα Servlets στέλνουν διαμέσου των sockets αντικείμενα της κλάσης SocketMessage.java ώστε να αναφέρουν την μέθοδο που επιθυμούν να κληθεί και τα ορίσματα που αυτή δέχεται. Η κλάση MyRuntime.java επιτελεί την κυρίως λειτουργία του Runtime και δημιουργεί αντικείμενα της διεπαφής AppInterface.java κατά τη σύνθεση και δημιουργία των java κλάσεων που αντιπροσωπεύουν τους αυτοματισμούς του χρήστη.

SERVLETS

Το Servlet **ActiveInactiveApps.java** ουσιαστικά “ζητάει” από το Runtime να του αναφέρει τις ενεργές και μη ενεργές εφαρμογές του χρήστη. Έτσι καλείται η μέθοδος getActiveApps() της κλάσης MyRuntime.java, διαβάζεται ένα αρχείο όπου καταγράφονται όλες οι εφαρμογές του χρήστη και επιστρέφεται στο Servlet μια λίστα με τις ενεργές εφαρμογές και μια λίστα με τις μη ενεργές.

Μέσω του Servlet **AddResource.java** η HTML σελίδα ενημερώνεται από το Runtime (καλείται η μέθοδος findResToFollow()) για τα Twitter Accounts που μπορεί να επιλέξει να κάνει follow ο χρήστης ενώ με το **RmvResource.java** (καλείται η μέθοδος fingResToUnfollow() της MyRuntime.java) ενημερώνεται για τα Twitter Accounts που μπορεί ο χρήστης να επιλέξει για να κάνει unfollow.

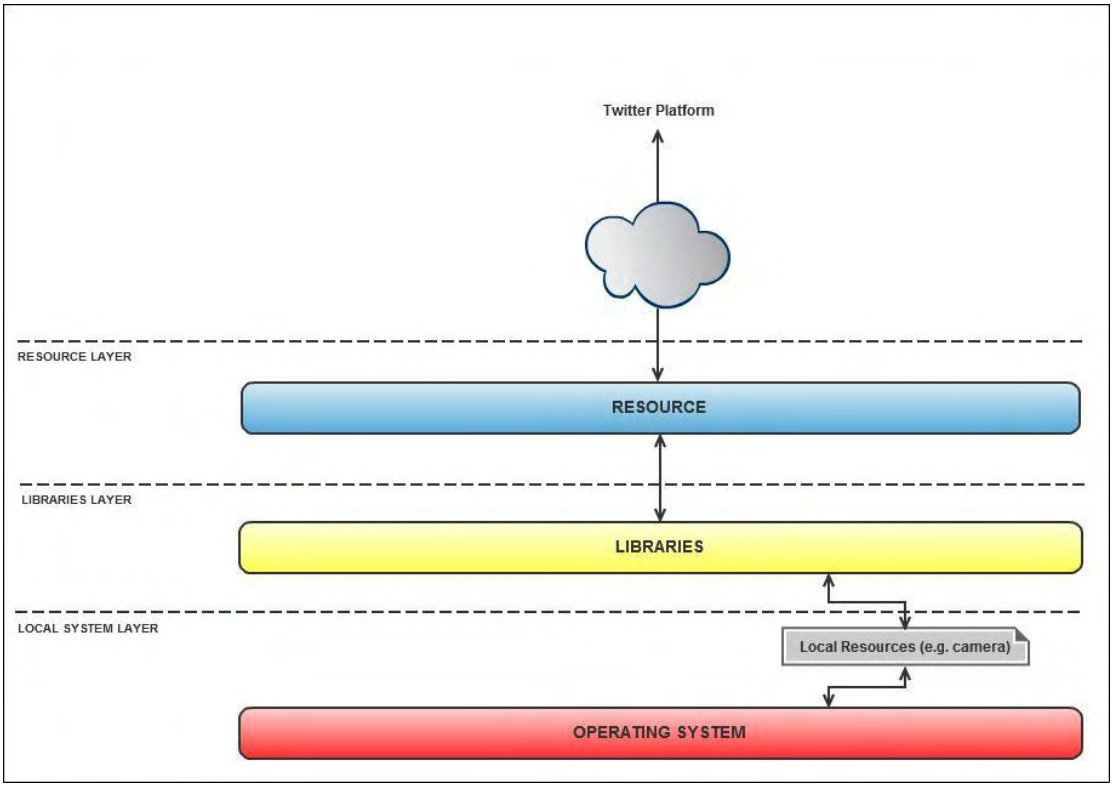
Η λειτουργία του follow ή του unfollow κάποιου Twitter Account γίνεται με συνδιασμό των Servlets **Follow.java** και **Unfollow.java** και του Runtime (μέθοδοι follow() και unfollow() αντίστοιχα).

Για την υποβολή κάποιας νέας εφαρμογής καλείται το Servlet **NewApp.java** ενώ τα Servlets **Edit.java** και **EditiedApp.java** καλούνται στην τροποποίηση κάποιας ήδη υπάρχουσας εφαρμογής.

Όταν ο χρήστης επιλέξει μέσω την HTML σελίδας την αρχή της εκτέλεσης μιας εφαρμογής καλείται το Servlet **StartApp.java** ενώ για τον τερματισμό μιας εφαρμογής το Servlet **StopApp.java** και η έναρξη ή ο τερματισμός γίνονται αντίστοιχα από τις μεθόδους startApp() και stopApp() της κλάσης MyRuntime.java**.**

5 Πόροι και Πρωτόκολλο Επικοινωνίας

Όπως έχει ήδη αναφερθεί, η επικοινωνία με τους πόρους γίνεται μέσω των Direct Messages του Twitter . Η αρχιτεκτονική υλοποίησης των πόρων φαίνεται στην Εικόνα 6. Βάση του συστήματος αποτελεί το λειτουργικό σύστημα. Στο αμέσως επόμενο επίπεδο βρίσκονται οι βιβλιοθήκες που χρησιμοποιήθηκαν (**JavaCV [10]**, βιβλιοθήκες Java για **Twiiter API [7]**, **Flickr API [8]**, **Google Docs API [9]** κτλ.). Στη συνέχεια, πάνω σε αυτές τις βιβλιοθήκες δημιουργήθηκε το πρόγραμμα που υλοποιεί τον πόρο και το οποίο επικοινωνεί με την πλατφόρμα του Twitter για λήψη και αποστολή των Direct Messages.



Εικόνα 6: Αρχιτεκτονική Υλοποίησης Πόρων

Μπορεί λοιπόν ο χρήστης ή κάποια εφαρμογή του χρήστη να στείλει ένα μήνυμα στον πόρο για να του πει ποια ενέργεια επιθυμεί να εκτελεστεί. Τα μηνύματα αυτά αποτελούν εντολές τις οποίες ο εκάστοτε πόρος είναι ικανός να αντιληφθεί μετά από αποκωδικοποίηση. Η αποδεκτή μορφή των μηνυμάτων είναι η παρακάτω:

< STRING> <COMMAND> < STRING>

Το API του Twitter δεν επιτρέπει την αποστολή συνεχόμενων ίδιων tweets ή Direct Messages από ένα Twitter Account σε μικρό χρονικό διάστημα. Ωστόσο με μεγάλη πιθανότητα ένας χρήστης ή μια εφαρμογή θα θέλει να στείλει την ίδια εντολή σε κάποιο πόρο εντός των ορίων του χρονικού περιορισμού. Για να ξεπεραστεί αυτό το πρόβλημα η εντολή πρέπει να ακολουθείται από μια τυχαία συμβολοσειρά <STRING>

(συμπεριλαμβανομένης της κενής συμβολοσειράς) ώστε να διαφοροποιείται το αποστελλόμενο μήνυμα και να μην αρνείται το API του Twitter την προώθηση του μηνύματος στον παραλήπτη.

Το <STRING> που προηγείται της εντολής χρησιμοποιείται για να δηλώνονται διαφορετικές εφαρμογές μέσα από το ίδιο Twitter Account. Χρησιμοποιείται δηλαδή κυρίως από το Περιβάλλον Προγραμματισμού Αυτοματισμών ώστε να δηλώνει από ποια εφαρμογή στάλθηκε η συγκεκριμένη εντολή. Μπορεί και ο χρήστης να προσθέσει μια συμβολοσειρά η οποία να προηγείται της εντολής αλλά για δική του ευκολία, προτείνεται να επικοινωνεί με τους πόρους με την παρακάτω μορφή μηνυμάτων:

<COMMAND> < STRING>

Συνεπώς ο πόρος είναι ικανός να αντιληφθεί την εντολή, εφόσον αυτή υπάρχει, ακόμα και αν προηγείται ή έπεται οποιαδήποτε συμβολοσειρά και η απάντηση που θα λάβει ο χρήστης θα είναι της μορφής:

< STRING> <REPLY> <RANDOM_INT>

όπου < STRING> η συμβολοσειρά που προηγούταν της εντολής στο μήνυμα που είχε στείλει ο χρήστης.

Οι πόροι εκτός από τις εντολές που είναι σχετικές με τη φύση και τις δυνατότητές τους υποστηρίζουν και δύο εντολές με κοινές λειτουργίες.

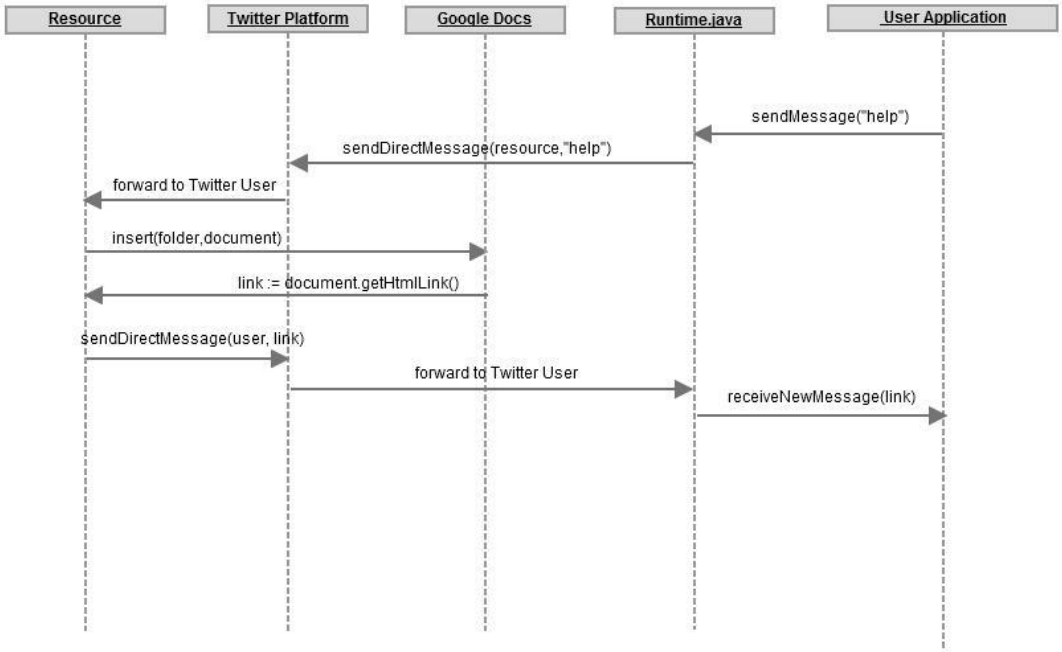
Στον Πίνακα περιγράφονται οι κοινές εντολές σε όλους τους πόρους.

Πίνακας 1: Κοινές Υποστηριζόμενες Εντολές

Όνομα Εντολής	Σύνταξη Εντολής	Περιγραφή Λειτουργίας	Σύνταξη Απάντησης	Παραδείγματα
help	***help***	**Στέλνεται στο χρήστη ως απάντηση link σε αρχείο με λίστα των εντολών που υποστηρίζει ο πόρος**	***link: - <link> - <random_int>***	Request: ***help*** Response: ***link: - http:// - 642***
info[1]	***info***	**Στέλνεται στο χρήστη ως απάντηση link σε xml αρχείο με πληροφορίες για τις εντολές που υποστηρίζει ο πόρος και τις απαντήσεις σε αυτές**	***info: - <link> - <random_int>***	Request: ***info*** Response: ***info: - http:// - 258***

[1] Η εντολή αυτή χρησιμοποιείται από το σύστημα διαχείρισης χωρίς ωστόσο αυτό να σημαίνει ότι δε μπορεί να δοθεί και από απλούς χρήστες.

Όταν στέλνεται η εντολή "help" σε κάποιο πόρο χρησιμοποιείται το API του Google Docs και ο πόρος ανεβάζει στο Google Docs ένα αρχείο που περιέχει τις υποστηριζόμενες εντολές. Στη συνέχεια καλείται η μέθοδος responseToHelp() ώστε να σταλεί ως απάντηση το link του αρχείου. Τέλος αν η εντολή είναι "info" χρησιμοποιείται το Google Docs API ώστε ο πόρος να ανεβάσει ένα αρχείο xml με πληροφορίες για τις εντολές που υποστηρίζει και τα μηνύματα που ο ίδιος στέλνει και καλείται η μέθοδος responseToInfo() για να σταλεί ως απάντηση το link του αρχείου. Οι εντολές "help" και "info" ακολουθούν παρόμοια διαδικασία κατά την εκτέλεσή τους και στην Εικόνα 7 φαίνεται το ακολουθιακό διάγραμμα.



Εικόνα 7: Ακολουθιακό Διάγραμμα Πόρου για Upload Αρχείου στο Google Docs.

Στη συνέχεια του κεφαλαίου θα περιγραφούν κάποιοι ενδεικτικοί πόροι που υποστηρίζει το σύστημα καθώς και η υλοποίησή τους.

5.1 Κάμερα

Στην υλοποίηση του πόρου αυτού χρησιμοποιήθηκε η κάμερα που είναι συνδεδεμένη στον υπολογιστή.

Αρχικά δημιουργήθηκε ένα Twitter Account για τον πόρο με screen name “cam_device” και με αυτό τον τρόπο μπήκαν τα θεμέλια του καναλιού επικοινωνίας.

Απαραίτητος ήταν στη συνέχεια ο καθορισμός των εντολών/μηνυμάτων που θα υποστηρίζει ο πόρος. Οι εντολές αυτές φαίνονται στον Πίνακα 2.

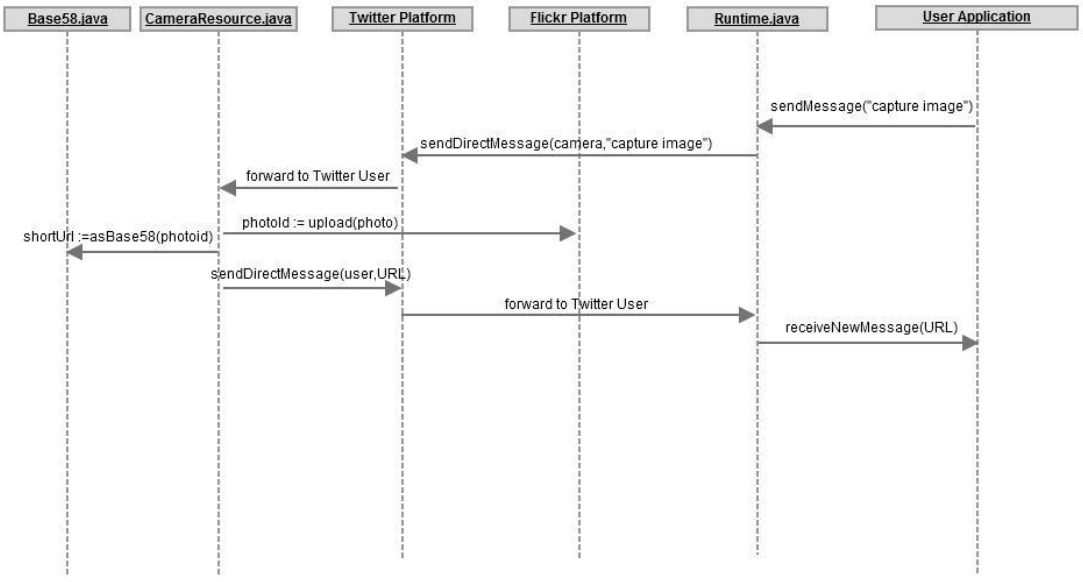
Πίνακας 2: Υποστηριζόμενες Εντολές Κάμερας

Όνομα Εντολής	Σύνταξη Εντολής	Περιγραφή Λειτουργίας	Σύνταξη Απάντησης	Παραδείγματα
capture image	***capture image***	**Δίνεται εντολή στην κάμερα να βγάλει φωτογραφία. Η φωτογραφία ανεβαίνει στο flickr και το link της στέλνεται ως απάντηση στο χρήστη.**	***photo: - <photo_link> - <random_int>***	Request: ***capture image*** Response: ***photo: - http:// - 642***
help capture image	***help capture image***	**Στέλνεται στο χρήστη ως απάντηση η περιγραφή της λειτουργίας της εντολής "capture image".**	***webcam takes picture <random_int>***	Request: ***help capture image*** Response: ***webcam takes picture 421***

Αναλυτικότερα, ο πόρος υλοποιείται μέσω της κλάσης CameraResource.java η οποία περιέχει main() μέθοδο. Κατά τη κλήση της main() μεθόδου διαβάζονται οι εντολές που υποστηρίζει ο πόρος, από το αρχείο όπου είναι καταγεγραμμένες (καλείται η μέθοδος readCommands()) και στη συνέχεια μέσω των API του Twitter και του Flickr γίνεται η σύνδεση στα δύο αυτά social media (καλείται η μέθοδος connectTwitterFlickr()). Εσωτερικά της μεθόδου connectTwitterFlickr() υπάρχει listener για τα εισερχόμενα Direct Messages.

Όταν ο πόρος λαμβάνει κάποιο Direct Message γίνεται αποκωδικοποίησή και ανιχνεύεται αν εμπεριέχει κάποια από τις εντολές που υποστηρίζει ο πόρος. Σε περίπτωση που δεν εμπεριέχεται καμία εντολή στέλνεται η απάντηση “invalid command”, καλώντας τη μέθοδο responseToInvalidCommand(). Διαφορετικά, αν εμπεριέχεται κάποια από τις υποστηριζόμενες εντολές, καλείται η μέθοδος executeCommand() που εκτελεί τις απαραίτητες ενέργειες κάθε φορά.

Σε περίπτωση που η εντολή είναι “capture image”, τότε καλείται η μέθοδος captureImage() μέσω της οποίας, και με τη χρήση της βιβλιοθήκης OpenCV, η κάμερα τραβάει φωτογραφία. Στη συνέχεια καλείται η μέθοδος uploadImage() που ανεβάζει τη φωτογραφία στο Flickr. Επίσης η μέθοδος uploadImage() καλεί την μέθοδο asBase58() της υπερκλάσης Base58 ώστε να μετατρέψει, το τύπου long, id της φωτογραφίας που μόλις ανέβασε στο Flickr σε αριθμό 58-ης βάσης. Το Flickr υποστηρίζει μια υπηρεσία URL shortening η χρήση της οποίας προϋποθέτει τη μετατροπή του id της φωτογραφίας σε αριθμό 58-ης βάσης. Ο λόγος που γίνεται κάτι τέτοιο είναι για διασφαλιστεί ότι το μήνυμα που θα αποσταλεί μέσω Twitter θα έχει αριθμό χαρακτήρων εντός των επιτρεπόμενων ορίων (<140 χαρακτήρες). Τέλος καλείται η μέθοδος responseToCaptureImage() ώστε να σταλεί ως απάντηση το link της φωτογραφίας. Η παραπάνω διαδικασία περιγράφεται σχηματικά στο ακολουθιακό διάγραμμα της Εικόνας 8.



Εικόνα 8: Ακολουθιακό Διάγραμμα για την Εντολή “capture image”.

Αν η εντολή είναι “help capture image” καλείται η μέθοδος responseToHelpCaptureImage() ώστε να σταλεί απάντηση με την περιγραφή της λειτουργίας της εντολής “capture image”.

5.2 Αισθητήρας Κίνησης

Για να υλοποιηθεί ο αισθητήρας κίνησης χρησιμοποιήθηκε η κάμερα που είναι συνδεδεμένη στον υπολογιστή.

Η κίνηση εντοπίζεται με την παρακάτω διαδικασία:

Η κάμερα λαμβάνει διαδοχικές φωτογραφίες οι οποίες στη συνέχεια συγκρίνονται. Αν εντοπιστούν διαφορές έχει ανιχνευθεί κίνηση.

Το screen name στο Twitter του αισθητήρα κίνησης είναι “MotionDetector_” και όπως είναι γνωστό, η επικοινωνία γίνεται μέσα από αυτό το διαδικτυακό προφίλ.

Οι εντολές που υποστηρίζει ο πόρος φαίνονται στον Πίνακα 3.

Πίνακας 3: Υποστηριζόμενες Εντολές Αισθητήρα Κίνησης

Όνομα Εντολής	Σύνταξη Εντολής	Περιγραφή Λειτουργίας	Σύνταξη Απάντησης	Παραδείγματα
motion status request	***motion status request***	**Αποστέλλεται στο χρήστη απάντηση σχετικά με το αν ανιχνεύθηκε ή όχι κίνηση την τρέχουσα χρονική στιγμή**	***motion detected*** ***ή*** ***no motion detected***	Request: ***motion status request*** Response: ***motion detected 24***
turn on notification	***turn on notification***	**Ενεργοποιείται μια υπηρεσία ειδοποίησης που ενημερώνει τον χρήστη σε περίπτωση που εντοπιστεί κίνηση**	***notification is on…***	Request: ***turn on notification*** Response: ***notification is on… 596***
turn off notification	***turn off notification***	**Απενεργοποιείται η υπηρεσία ειδοποίησης κίνησης**	***notification is off…***	Request: ***turn off notification*** Response: ***notification is off… 4***
notification status	***notification status***	**Ο χρήστης ενημερώνεται για το αν έχει ενεργοποιημένη την υπηρεσία ειδοποίησης ή όχι**	***notification is on…*** ***ή*** ***notification is off…***	Request: ***notification status*** Response: ***notification is off… 93***
help motion status request	***help motion status request***	**Στέλνεται στο χρήστη ως απάντηση η περιγραφή της λειτουργίας της εντολής “motion status request”.**	***return the current detection(motion or no)***	Request: ***help motion status request*** Response: ***return the current detection(motion or no) 8***

help turn on notification	***help turn on notification***	**Στέλνεται στο χρήστη ως απάντηση η περιγραφή της λειτουργίας της εντολής "turn on notification ".**	***turn on notification process***	Request: ***help turn on notification*** Response: ***turn on notification process 12***
help turn off notification	***help turn off notification***	**Στέλνεται στο χρήστη ως απάντηση η περιγραφή της λειτουργίας της εντολής "turn off notification".**	***turn off notification process***	Request: ***help turn off notification*** Response: ***turn off notification process 41***
help notification status	***help notification status***	**Στέλνεται στο χρήστη ως απάντηση η περιγραφή της λειτουργίας της εντολής "notification status".**	***return notification process status (on or off)***	Request: ***help notification status*** Response: ***return notification process status (on or off) 581***

Ο αισθητήρας κίνησης υλοποιείται από τις κλάσεις MotionDetector.java, MotionDetection.java, ImageCompare.java και Subscriber.java. Η κλάση MotionDetector.java περιέχει main μέθοδο η οποία αρχικά καλεί τη μέθοδο readCommands() για να "διαβάζει" τις υποστηριζόμενες εντολές που υπάρχουν καταγεγραμμένες σε ένα αρχείο και στη συνέχεια καλεί τη μέθοδο connectTwitter() για να γίνει η σύνδεση με το Twitter. Η μέθοδος connectTwitter() περιέχει ένα listener ώστε να λαμβάνονται τα direct messages που αποστέλλονται σε αυτόν. Κατά την άφιξη ενός μηνύματος γίνεται "αποκωδικοποίηση" για να διαπιστωθεί αν περιέχει κάποια έγκυρη εντολή. Εάν δεν υπάρχει έγκυρη εντολή καλείται η μέθοδος responseToInvalidCommand() και στέλνεται η απάντηση "command invalid". Εάν εμπεριέχεται έγκυρη εντολή τότε καλείται η μέθοδος executeCommand() και η διαδικασία συνεχίζεται.

Σε περίπτωση που η εντολή είναι "motion status request" καλείται η μέθοδος responseToMotionStatusRequest() όπου γίνεται λήψη δύο συνεχόμενων φωτογραφιών που συγκρίνονται με τη μέθοδος compare() της κλάσης ImageCompare.java. Η απάντηση που στέλνεται στο χρήστη είναι ανάλογη του αποτελέσματος της σύγκρισης.

Αν η εντολή είναι "turn on notification", αρχικά ο χρήστης εγγράφεται σε λίστα μέσω της μεθόδου addSubcriber() (Στη λίστα προστίθενται αντικείμενα της κλάσης Subscriber.java). Στη συνέχεια, καλείται είτε η μέθοδος startMotionDetection() είτε η resumeMotionDetection() της κλάσης MotionDetection.java ανάλογα με την κατάσταση του νήματος της κλάσης αυτής. Ουσιαστικά δηλαδή η ανίχνευση κίνησης γίνεται με την εκτέλεση του νήματος της κλάσης MotionDetection.java. Η startMotionDetection() καλείται όταν η κατάσταση του νήματος είναι RUNNABLE ενώ η resumeMotionDetection() όταν η κατάσταση του νήματος είναι WAIT. Σε περίπτωση που εντοπιστεί κίνηση ο πόρος ενημερώνει όλους τους εγγεγραμμένους χρήστες.

Στην εντολή "turn off notification" αρχικά καλείται η μέθοδος removeSubscriber() όπου αφαιρείται ο χρήστης από τη λίστα με τους εγγεγραμμένους για την υπηρεσία ειδοποίησης. Στη συνέχεια ελέγχεται η λίστα και αν είναι άδεια τότε το νήμα της κλάσης MotionDetection.java μπαίνει στην κατάσταση WAIT.

Στην άφιξη της εντολής "notification status" ελέγχεται αν ο χρήστης είναι γραμμένος στη λίστα της υπηρεσίας ειδοποίησης και αποστέλλεται σε αυτόν η ανάλογη απάντηση.

Η υλοποίηση των εντολών "help" και "info" έγινε με τον τρόπο που περιγράφηκε στον προηγούμενο πόρο. Επίσης αν η εντολή είναι κάποια από τις "help motion status request", "help turn on notification", "help turn off notification" ή "help notification status" καλούνται αντίστοιχα οι μέθοδοι responseToHelpMotionStatusRequest(), responseToHelpTurnOnNotification(), responseToHelpTurnOffNotification(), responseToHelpNotificationStatus() και στέλνεται απάντηση στο χρήστη.

5.3 Αισθητήρας Θερμοκρασίας

Το πρόγραμμα που υλοποιήθηκε προσομοιώνει τη λειτουργία ενός αισθητήρα θερμοκρασίας. Παράγονται τυχαίες τιμές ανά τακτά χρονικά διαστήματα που χρησιμοποιούνται ως τιμές θερμοκρασίας. Οι τιμές που παράγονται δεν έχουν καμία σχέση με την πραγματική θερμοκρασία του περιβάλλοντος χώρου και ως εκ τούτου η υλοποίηση έγινε αποκλειστικά για την εξυπηρέτηση των αναγκών του demo που συνοδεύει την διπλωματική εργασία.

Ο πόρος έχει Twitter Account με screen name “SensorTemp”.

Στον Πίνακα 4 παρουσιάζεται η λίστα των εντολών που υποστηρίζει ο πόρος.

Πίνακας 4: Υποστηριζόμενες Εντολές Αισθητήρα Θερμοκρασίας

Όνομα Εντολής	Σύνταξη Εντολής	Περιγραφή Λειτουργίας	Σύνταξη Απάντησης	Παραδείγματα
temperature status request	***temperature status request***	**Στέλνεται ως απάντηση την τρέχουσα θερμοκρασία περιβάλλοντος**	***the temperature is - <int> -***	Request: ***temperature status request*** Response: ***the temperature is -25-89***
turn on notification	***turn on notification - <OPERATOR> <TEMPERATURE> -*** [2]	**Ενεργοποιείται η υπηρεσία ειδοποίησης. Σε περίπτωση που ξεπεραστεί το προκαθορισμένο από το χρήστη στέλνεται μήνυμα ειδοποίησης.**	**notification is on…**	Request: ***turn on notification*** Response: ***notification is on… 56***
turn off notification	***turn off notification***	**Απενεργοποιείται η υπηρεσία ειδοποίησης**	***notification is off…***	Request: ***turn off notification*** Response: ***notification is off… 854***
notification status	***notification status***	**Ο χρήστης ενημερώνεται για το αν έχει ενεργοποιημένη την υπηρεσία ειδοποίησης ή όχι**	***notification is on…*** ***ή*** ***notification is off…***	Request: ***notification status*** Response: ***notification is off… 625***
help temperature status request	***help temperature status request***	**Στέλνεται στο χρήστη ως απάντηση η περιγραφή της λειτουργίας της εντολής "temperature status request".**	***return the current temperature***	Request: ***help temperature status request*** Response: ***return the current temperature 954***

[2] Ανάμεσα στις παύλες μπαίνει η παράμετρος που θέτει ο χρήστης. Όπου <OPERATOR> ο χρήστης πρέπει να εισάγει είτε το σύμβολο "<" είτε το ">" και όπου <TEMPERATURE> την θερμοκρασία που θα αποτελεί το όριο.

help turn on notification	***help turn on notification***	**Στέλνεται στο χρήστη ως απάντηση η περιγραφή της λειτουργίας της εντολής "turn on notification ".**	***turn on notification process***	Request: ***help turn on notification 74*** Response: ***turn on notification process***
help turn off notification	***help turn off notification***	**Στέλνεται στο χρήστη ως απάντηση η περιγραφή της λειτουργίας της εντολής "turn off notification ".**	***turn off notification process***	Request: ***help turn off notification*** Response: ***turn off notification process***
help notification status	***help notification status***	**Στέλνεται στο χρήστη ως απάντηση η περιγραφή της λειτουργίας της εντολής "notification status".**	***return notification process status (on or off)***	Request: ***help notification status*** Response: ***return notification process status (on or off)***

Οι κλάσεις TemperatureSensor.java, Subscriber.java και Notification.java υλοποιούν την προσομοίωση του αισθητήρα θερμοκρασίας. Η κλάση TemperatureSensor.java περιέχει μια main μέθοδο μέσα από την οποία ανακτώνται οι εντολές που υποστηρίζει ο πόρος (καλείται η μέθοδος readCommands()) και γίνεται η σύνδεση με το Twitter (μέθοδος connectTwitter()) ώστε να λαμβάνονται τα Direct Messages που έχουν παραλήπτη τον αισθητήρα θερμοκρασίας. Όταν ο listener που υπάρχει στην μέθοδο connectTwitter() λάβει νέο μήνυμα, γίνεται αποκωδικοποίηση και ελέγχεται αν εμπεριέχει έγκυρη εντολή. Αν δεν περιέχει κάποια εντολή τότε καλείται η μέθοδος responseToInvalidCommand() και στέλνεται στο χρήστη η απάντηση “command invalid ”. Αν η εντολή είναι έγκυρη καλείται η μέθοδος executeCommand() και η διαδικασία συνεχίζεται.

Σε περίπτωση που ο χρήστης έχει στείλει την εντολή “temperature status request” λαμβάνεται η τρέχουσα θερμοκρασία (στην ουσία παράγεται ένας τυχαίος αριθμός) και στέλνεται απάντηση στο χρήστη μέσω της μεθόδου responseToTemperatureStatusRequest().

Με την εντολή “turn on notification - <OPERATOR><TEMPERATURE> -” , δημιουργείται ένα αντικείμενο της κλάσης Subscriber.java και εισάγεται σε μια λίστα μέσω της μεθόδου addSubscriber(). Επίσης καλείται η μέθοδος startNotification() ή η μέθοδος resumeNotification() ανάλογα με την κατάσταση του νήματος της κλάσης Notification.java ώστε να αρχίζει η εκτέλεση του νήματος της κλάση αυτής. Η κλάση Notification.java είναι υπεύθυνη για την υπηρεσία ειδοποίησης.

Με την εντολή “turn off notification” ο χρήστης διαγράφεται από τη λίστα της υπηρεσίας ειδοποίησης με κλήση της μεθόδου removeSubscriber(). Επίσης αν η λίστα είναι άδεια καλείται η μέθοδος suspendNotification() και το νήμα της κλάσης Notification.java μπαίνει στην κατάσταση WAIT.

Αν ο χρήστης στείλει την εντολή “notification status” ελέγχεται αν είναι εγγεγραμμένος στην υπηρεσία ειδοποίησης και στέλνεται ανάλογη απάντηση μέσω της μεθόδου responseToNotificationStatusRequest().

Η υλοποίηση των εντολών “help” και “info” έγινε με τον τρόπο που περιγράφηκε στους προηγούμενους πόρους.

Αν η εντολή είναι κάποια από τις “help temperatue status request”, “help turn on notification - <OPERATOR> <TEMPERATURE> -”, “help turn off notification” ή “help notification status” καλούνται αντίστοιχα οι μέθοδοι responseToHelpNotificationStatusRequest(), responseToHelpTurnOnNotification(), responseToHelpTurnOffNotification(), responseToHelpNotificationStatus() και στέλνεται απάντηση στο χρήστη.

6 Γλώσσα Προγραμματισμού

Τα εργαλεία που δίνονται στο χρήστη για τη δημιουργία των αυτοματισμών είναι αφενός η γραφική διεπαφή (HTML σελίδα διαχείρισης) και αφετέρου μια Γλώσσα Προγραμματισμού που δημιουργήθηκε ειδικά για αυτό το σκοπό. Η Γλώσσα Προγραμματισμού είναι αρκετά εύκολη στην κατανόηση, φιλική στο χρήστη και σε γενικές γραμμές ακολουθεί συνήθης δομές και μοντέλα προγραμματισμού γεγονός που κάνει το χρήστη να νιώθει αρκετά οικεία. Σε αυτό το κεφάλαιο θα γίνει αναφορά στη Γλώσσα Προγραμματισμού που αναπτύχθηκε. Στις υποενότητες που ακολουθούν θα περιγραφεί αναλυτικά η Γλώσσα αυτή και πως ο χρήστης μπορεί να την χρησιμοποιεί ώστε να δημιουργήσει την εφαρμογή που επιθυμεί.

Η ανάπτυξη του κώδικα της εφαρμογής αυτοματισμού στην HTML σελίδα διαχείρισης γίνεται σε τρία πεδία, το πεδίο init, το πεδίο body και το πεδίο clean up. Στο πεδίο init ο χρήστης καθορίζει τις ενέργειες που επιθυμεί να γίνουν πριν την αρχή της εκτέλεσης του κυρίως μέρους της εφαρμογής. Στο πεδίο body γράφεται το κυρίως μέρος της εφαρμογής και στο πεδίο clean up ορίζονται οι ενέργειες που ο χρήστης επιθυμεί να εκτελεστούν ακριβώς πριν τον τερματισμό της εφαρμογής.

Στις επόμενες υποενότητες καθορίζονται οι επιτρεπτές εντολές και δομές κώδικα για κάθε πεδίο της εφαρμογής αυτοματισμών και παρατίθενται παραδείγματα ενδεικτικών εφαρμογών.

6.1 Τύποι Δεδομένων και Τελεστές

Τη βάση της Γλώσσας Προγραμματισμού αποτελούν τέσσερις απλοί τύποι δεδομένων που παρατίθενται στο Πίνακα 5.

Πίνακας 5

Τύπος	Σημασία	Αντιστοιχία στη Java
var	Αναπαριστά αριθμητικές τιμές	double
message	Αναπαριστά συμβολοσειρές	string
boolean	Αναπαριστά τιμές αληθείς/ψευδείς	boolean
time	Αναπαριστά την τρέχουσα ώρα	System.nanoTime()

ΔΗΛΩΣΗ ΜΕΤΑΒΛΗΤΩΝ

Στην Γλώσσα Προγραμματισμού είναι απαραίτητη η δήλωση του τύπου των μεταβλητών και η αρχικοποίηση πριν τη χρήση τους και αυτό γίνεται με τον εξής τρόπο:

τύπος όνομα_μεταβλητής = αρχική_τιμή ;

Παραδείγματος χάριν μια μεταβλητή τύπου var με όνομα “i” και αρχική τιμή 0 δηλώνεται :
var i=0;

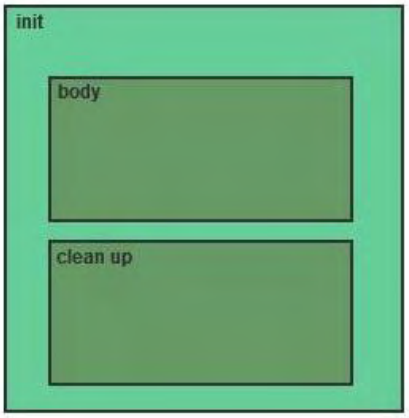
Επίσης για να δηλώσουμε μια μεταβλητή τύπου message με όνομα “str” και αρχική τιμή “I am a string” γράφουμε: message str=”I am a string”;

Τέλος αν η μεταβλητή ήταν τύπου boolean με όνομα “isboolean” και αρχική τιμή true η δήλωσή της θα ήταν : boolean isboolean= true;

Η αρχικοποίηση μιας μεταβλητής είναι απαραίτητη κατά τη δήλωση. Αν παραληφθεί η δήλωση ή η αρχικοποίηση μια μεταβλητής ο μεταγλωττιστής θα εντοπίσει λάθος. Ωστόσο δεν ισχύει το ίδιο για τις μεταβλητές τύπου time. Η δήλωση μιας μεταβλητής τύπου time γίνεται ως εξής: time όνομα_μεταβλητής; (πχ time currentTime;). Σε περίπτωση που ο χρήστης επιχειρήσει να αρχικοποιήσει την μεταβλητή τύπου time ο μεταγλωττιστής θα εντοπίσει λάθος και ο λόγος που συμβαίνει αυτό είναι γιατί η μεταβλητή αυτή αντιπροσωπεύει την ώρα σε κάθε χρονική στιγμή, οπότε η τιμή της καθορίζεται από τον μεταγλωττιστής.

ΕΜΒΕΛΕΙΑ ΜΕΤΑΒΛΗΤΩΝ

Κατά τη δημιουργία μιας μεταβλητής στην πραγματικότητα δημιουργείται ένα στιγμιότυπο του αντίστοιχου τύπου. Έτσι οι δυνατότητές της περιορίζονται στις δυνατότητες του τύπου που η μεταβλητή είναι δηλωμένη, δηλαδή μια μεταβλητή τύπου message δε θα μπορούσε να χρησιμοποιηθεί για αποθήκευση αριθμητικών τιμών. Επίσης είναι σημαντικό να αναφερθεί ότι μια μεταβλητή δε μπορεί να αλλάξει τον τύπο της καθ’ όλη τη διάρκεια του χρόνου ζωής του προγράμματος.



Εικόνα 9: Εμβέλειες Μεταβλητών

Η Γλώσσα Προγραμματισμού που αναπτύχθηκε επιτρέπει τη δήλωση μεταβλητών εξωτερικά αλλά και εσωτερικά σε οποιοδήποτε block κώδικα (ένα block ξεκινά με “{” και κλείνει με “}”) και δημιουργείται νέα εμβέλεια για τις μεταβλητές αυτού του block. Οι μεταβλητές που δηλώνονται στο πεδίο init εξωτερικά οποιουδήποτε block αποτελούν global μεταβλητές. Ωστόσο δεν ισχύει το ίδιο και για τα πεδία body και clean up. Όπως φαίνεται στην Εικόνα 9 μια μεταβλητή που είναι δηλωμένη στο πεδίο init (και δεν περιέχεται σε κάποιο block) είναι ορατή και στα πεδία body και clean up. Μια μεταβλητή δηλωμένη στο body είναι ορατή μόνο εντός του συγκεκριμένου πεδίου και το ίδιο ισχύει για μεταβλητές που έχουν δηλωθεί στο clean up. Έτσι όταν η μεταβλητή δηλωθεί στο πεδίο init μπορούμε να πούμε ότι θεωρείται global ενώ όταν δηλωθεί στα πεδία body ή clean up έχει εμβέλεια το αντίστοιχο πεδίο.

Ακολουθούν δύο παραδείγματα σχετικά τη χρήση των μεταβλητών και την εμβέλειά τους. Το Παράδειγμα Ι θα περνούσε επιτυχώς τη διαδικασία της μεταγλώττισης ενώ το Παράδειγμα ΙΙ δε θα περνούσε επιτυχώς τη διαδικασία αυτή.

Παράδειγμα Ι

init

```
var myvar=0;
```

body

```
```

clean up

```
var i=0;
myvar++;
```

Στο πεδίο init ορίζεται η μεταβλητή myvar και αρχικοποιείται στην τιμή μηδέν (0). Η μεταβλητή αυτή είναι ορατή από το πεδίο clean up, λόγω του ότι δηλώθηκε στο πεδίο init εξωτερικά οποιουδήποτε block, και έτσι μπορεί να γίνει επεξεργασία της από τα πεδία clean up και body. Στο πεδίο clean up δηλώνεται η μεταβλητή i και είναι ορατή μόνο στο πεδίο αυτό.

Παράδειγμα ΙΙ

init

```
var i=0;
if(i==0){
  var j=1;
}
```

body

```
```

clean up

```
i++;
j++;
```

Στο πεδίο init δηλώνεται η μεταβλητή i, η οποία θεωρείται global και είναι ορατή και στα πεδία body και clean up. Ωστόσο η μεταβλητή j δηλώνεται μεν στο πεδίο init αλλά εσωτερικά ενός block κώδικα και συνεπώς δεν είναι ορατή στα άλλα δύο πεδία αλλά μόνο εσωτερικά του συγκεκριμένου block κώδικα. Έτσι ο μεταγλωττιστής θα εντοπίσει λάθος αφού στο πεδίο clean up γίνεται ενέργεια αλλαγής τιμής και για τις δύο μεταβλητές που έχουν δηλωθεί στο πεδίο init.

ΑΡΙΘΜΗΤΙΚΟΙ ΤΕΛΕΣΤΕΣ

Ο ψευδοκώδικας υποστηρίζει τους αριθμητικούς τελεστές που φαίνονται στον Πίνακα 6.

Πίνακας 6

Τελεστής	Σημασία
+	πρόσθεση
-	αφαίρεση
++	προσαύξηση
--	μείωση
/	διαίρεση
*	πολλαπλασιασμός
%	Υπόλοιπο

Οι παραπάνω τελεστές λειτουργούν όπως και στη γλώσσα Java και εφαρμόζονται **μόνο** σε μεταβλητές αριθμητικού τύπου (var). Ο τελεστής πρόσθεσης (+) στη Java χρησιμοποιείται και ως τελεστής συνένωσης String αλλά στην Γλώσσα Προγραμματισμού που αναπτύχθηκε χρησιμοποιείται μόνο ως αριθμητικός τελεστής.

Όταν ένας τελεστής προσαύξησης ή μείωσης χρησιμοποιείται ως μέρος μιας μεγαλύτερης έκφρασης έχει σημασία η θέση που θα τοποθετηθεί. Για τη θέση των τελεστών προσαύξησης και μείωσης οι κανόνες είναι ίδιοι με αυτούς της Java.

ΣΧΕΣΙΑΚΟΙ ΚΑΙ ΛΟΓΙΚΟΙ ΤΕΛΕΣΤΕΣ

Οι τελεστές που υποστηρίζει η Γλώσσα Προγραμματισμού για να καθοριστούν οι σχέσεις μεταξύ των τιμών φαίνονται στον Πίνακα 7.

Πίνακας 7

Τελεστής	Σημασία
==	ίσο με
!=	άνισο
>	μεγαλύτερο από
<	μικρότερο από
>=	μεγαλύτερο από ή ίσο με
<=	μικρότερο από ή ίσο με

Όλοι οι παραπάνω σχεσιακοί τελεστές μπορούν να χρησιμοποιηθούν για σύγκριση αριθμητικών τιμών και μεταβλητών τύπου var. Για σύγκριση μεταβλητών τύπου message και boolean μπορούν να χρησιμοποιηθούν μόνο οι τελεστές == και !=. Τέλος, για σύγκριση τιμών ώρας (αναφορά στις τιμές ώρας θα γίνει στις επόμενες υποενότητες) χρησιμοποιούνται μόνο οι τελεστές > και < .

Στον Πίνακα 8 φαίνονται οι λογικοί τελεστές.

Πίνακας 8

Τελεστής	Σημασία
&	ΚΑΙ
|	Ή
^	ΑΠΟΚΛΕΙΣΤΙΚΟ Ή
||	ΒΡΑΧΥΚΥΚΛΩΜΕΝΟ Ή
&&	ΒΡΑΧΥΚΥΚΛΩΜΕΝΟ ΚΑΙ
!	ΟΧΙ

Για του λογικούς τελεστές, οι τελεστέοι πρέπει να είναι τύπου boolean, ενώ το αποτέλεσμα από μια λογική λειτουργία είναι τύπου boolean.

ΤΕΛΕΣΤΗΣ ΕΚΧΩΡΗΣΗΣ

Ο τελεστής εκχώρησης έχει χρησιμοποιηθεί ήδη στην αρχικοποίηση των μεταβλητών, συμβολίζεται με το σύμβολο της ισότητας (=) και η γενική του σύνταξη είναι η ακόλουθη:

μεταβλητή =έκφραση;

Για να μην εντοπίσει λάθος ο μεταγλωττιστής ο τύπος της μεταβλητής πρέπει να είναι συμβατός με τον τύπο της έκφρασης.

6.2 Εντολές

ΠΡΟΤΑΣΗ BREAK

Η πρόταση break στην Γλώσσα Προγραμματισμού που αναπτύχθηκε χρησιμοποιείται για έξοδο από ένα βρόχο ή από τη δομή αναμονής μηνυμάτων. Όταν το πρόγραμμα συναντήσει μια πρόταση break μέσα σε ένα βρόχο, τότε έχουμε τον τερματισμό του βρόχου, ενώ ο έλεγχος του προγράμματος συνεχίζει αμέσως από την επόμενη πρόταση που ακολουθεί μετά το βρόχο. Ομοίως αν το πρόγραμμα συναντήσει μια πρόταση break μέσα σε μια δομή αναμονής μηνυμάτων τότε τερματίζεται η αναμονή. Αν η πρόταση break δεν βρίσκεται εντός του βρόχου ή της δομής αναμονής μηνυμάτων (while και case αντίστοιχα) τότε ο μεταγλωττιστής εντοπίζει λάθος.

ΠΡΟΤΑΣΗ SENDMSG

Με την εντολή sendmsg η εφαρμογή στέλνει ένα Direct Message. Η ακριβής μορφή της είναι : sendmsg(<RECEIVER_STRING>, <MESSAGE_STRING>);

Όπου, όπως δηλώνουν και τα ονόματα των παραμέτρων, <RECEIVER_STRING> είναι ο παραλήπτης του Direct Message και <MESSAGE_STRING > το μήνυμα που θα σταλεί.

Οι παράμετροι <RECEIVER_STRING> και < MESSAGE_STRING > πρέπει να είναι συμβολοσειρές διαφορετικά θα εντοπιστεί λάθος από τον μεταγλωττιστή. Αν για παράδειγμα κάποια εφαρμογή θέλει να στείλει την εντολή “capture image” στον πόρο με screen name “cam_device” η εντολή θα διαμορφωνόταν ως εξής: sendmsg(“cam_device”,”capture image”);

Κατά τη διαδικασία μεταγλώττισης ελέγχεται αν υπάρχει αμοιβαίο “follow” μεταξύ των Twitter Accounts του συστήματος διαχείρισης και του αποδέκτη του μηνύματος και αναφέρεται λάθος σε περίπτωση που δεν υπάρχει.

Η εντολή sendmsg μπορεί να γραφεί και στα τρία πεδία (init, body, clean up) καθώς επίσης και εσωτερικά όλων των block κώδικα.

6.3 Δομές Ελέγχου Προγράμματος

ΔΟΜΗ IF

Η γενικότερη μορφή της πρότασης if είναι η ακόλουθη:

```
If(συνθήκη){
   σειρά προτάσεων
}
else{
   σειρά προτάσεων
}
```

Ακριβώς όπως στις περισσότερες γλώσσες προγραμματισμού αν είναι αληθής η συνθήκη τότε εκτελούνται οι προτάσεις εσωτερικά του if διαφορετικά εκτελούνται οι προτάσεις του else (αν αυτό υπάρχει λόγω του ότι είναι προαιρετικό). Εδώ πρέπει να σημειωθεί ότι είναι απαραίτητη η ύπαρξη τουλάχιστον μιας πρότασης εσωτερικά των block του if και του else. Όταν ο μεταγλωττιστής εντοπίσει άδεια block θα αναφέρει λάθος.

ΔΟΜΗ WHILE

Η γενικότερη μορφή της πρότασης while είναι η ακόλουθη:

```
while(συνθήκη){
   σειρά προτάσεων
}
```

Όπως και στη δομή if, το όρισμα της συνθήκης καθορίζει τη συνθήκη που ελέγχει το βρόχο και μπορεί να είναι οποιαδήποτε έγκυρη έκφραση boolean. Ο βρόχος επαναλαμβάνεται όσο η συνθήκη είναι αληθής. Όταν η συνθήκη γίνει ψευδής συνεχίζει η εκτέλεση της εφαρμογής από την πρόταση που ακολουθεί αμέσως μετά τον βρόχο. Όπως και στη δομή if είναι απαραίτητη η ύπαρξη μιας τουλάχιστον πρότασης εσωτερικά του block, διαφορετικά θα αναφερθεί πρόβλημα από τον μεταγλωττιστή.

ΔΟΜΗ CASE

Η δομή case δημιουργήθηκε για την ανάγκη της Γλώσσας Προγραμματισμού για αναμονή μηνυμάτων. Η γενική μορφή της είναι:

I.
```
case (  ( συνθήκη )  <ΛΟΓΙΚΟΣ_ΤΕΛΕΣΤΗΣ>  recvmsg ( <SENDER _STRING> ,
<MESSAGE_STRING >,  μεταβλητή  )  )  {
   σειρά προτάσεων
}
```

ή

II.
```
case( (συνθήκη) <ΛΟΓΙΚΟΣ_ΤΕΛΕΣΤΗΣ>   recvarmsg (<SENDER _STRING>,
<MESSAGE_STRING >,   μεταβλητή1,  μεταβλητή2 ) ){
   σειρά προτάσεων
}
```

Στην πρώτη Περίπτωση I για να εκτελεστούν οι προτάσεις που βρίσκονται εσωτερικά του block πρέπει να αληθεύει η έκφραση:
(συνθήκη) <λογικός τελεστής> recvmsg (<SENDER_STRING> , <MESSAGE_STRING> , μεταβλητή)

Ουσιαστικά η έκφραση αυτή αποτελείται από δύο μέρη. Το πρώτο είναι μια συνήθης λογική συνθήκη ενώ το δεύτερο είναι η πρόταση recvmsg. Η λειτουργία της πρότασης recvmsg είναι να ανιχνεύει αν κάποιο από τα μηνύματα που λαμβάνει η εφαρμογή είναι αυτό που ορίζεται στην ίδια. Για παράδειγμα αν ο χρήστης έχει ορίσει recvmsg(“arlabrin”, “hello”,msg) αυτό σημαίνει ότι η συγκεκριμένη πρόταση recvmsg περιμένει το μήνυμα “hello” από τον Twitter User “arlabrin” και θα το αποθηκεύσει στη μεταβλητή msg. Η μεταβλητή msg πρέπει να είναι τύπου message και να έχει προηγηθεί δήλωση της. Συνεπώς το αποτέλεσμα της πρότασης recvmsg είναι τύπου boolean (true αν λήφθηκε το αναμενόμενο μήνυμα και false αν όχι) και σε συνδυασμό με το πρώτο μέρος της έκφρασης

έχουμε ως αποτέλεσμα μια boolean τιμή. Όταν η boolean τιμή είναι αληθής τότε εκτελούνται οι προτάσεις εσωτερικά του block.

Στην περίπτωση ΙΙ για να εκτελεστούν οι προτάσεις που βρίσκονται εσωτερικά του block πρέπει να αληθεύει η έκφραση:

(συνθήκη) <λογικός τελεστής> recvarmsg (<SENDER_STRING> , <MESSAGE_STRING> , μεταβλητή1, μεταβλητή2)

Όπως προηγουμένως η έκφραση αυτή αποτελείται από δύο μέρη. Η διαφορά με την Περίπτωση Ι είναι ότι η πρόταση recvarmsg χρησιμοποιείται για να αναμονή μηνυμάτων με μεταβλητό μέρος. Στη μεταβλητή1 θα αποθηκευτεί ολόκληρο το λαμβανόμενο μήνυμα ενώ στην μεταβλητή2 το μεταβλητό μέρος του μηνύματος. Συνεπώς η μεταβλητή1 πρέπει να είναι τύπου message ενώ ο τύπος της μεταβλητής2 πρέπει να είναι συμβατός με τον τύπο του μεταβλητού μέρους του μηνύματος που λαμβάνεται.

Η δομή case γράφεται αποκλειστικά στο πεδίο body και θεωρείται η κυρίως διαδικασία της εφαρμογής. Στο πεδίο body μπορούν να υπάρχουν περισσότερες από μια δομές case αλλά εξωτερικά αυτών δε μπορεί να γραφεί καμία άλλη δομή ούτε δήλωση μεταβλητής.

6.4 Παραδείγματα Εφαρμογών

6.4.1 Εφαρμογή Ι

Μια από τις πιο απλές εφαρμογές που θα μπορούσε να δημιουργήσει ο χρήστης είναι η παρακάτω:

```
init
sendmsg("cam_device", "capture image");
message l= "";

body
case((true)&&recvmsg("cam_device","photo:",l)){
    sendmsg("arlabrin",l);
}

clean up
```

Αρχικά, από το πεδίο init, στέλνεται στην κάμερα εντολή για λήψη φωτογραφίας και δηλώνεται η μεταβλητή `l` που είναι τύπου message για να χρησιμοποιηθεί στο πεδίο

body. Σο πεδίο body έχουν καθοριστεί οι ενέργειες που θα γίνουν μόλις ληφθεί μήνυμα από την κάμερα με το link για τη φωτογραφία που τράβηξε. Το μήνυμα που λαμβάνεται και περιέχει το link της φωτογραφίας αποθηκεύεται στη μεταβλητή `l` και στη συνέχεια αποστέλλεται στον Twitter User με screen name “arlabrin”.

6.4.2 Εφαρμογή ΙΙ

Μια επίσης απλή εφαρμογή θα μπορούσε να είναι η παρακάτω:

```
init
sendmsg(“MotionDetector_”, “turn on notification”);
message l= “”;
var i=0;

body
case((i==3)&&recvmsg("MotionDetector_","motion detected",l)){
    i=0;
    sendmsg("arlabrin",l);
}
case((i<3)&& recvmsg("MotionDetector_","motion detected",l)){
     i++;
}

clean up
sendmsg(“MotionDetector_”, “turn off notification”);
```

Στο πεδίο init ενεργοποιείται η υπηρεσία ειδοποίησης κίνησης και δηλώνονται οι μεταβλητές `l` και `i` οι οποίες θα χρησιμοποιηθούν στο πεδίο body. Στο πεδίο body έχουν οριστεί δύο δομές case που αναμένουν το ίδιο μήνυμα αλλά με διαφορετική συνθήκη. Στην πρώτη δομή case αναμένεται μήνυμα “motion detected” από τον αισθητήρα κίνησης αλλά θα εκτελεστούν οι εντολές εσωτερικά του block μόνο εάν η μεταβλητή `i` έχει την τιμή 3. Σε περίπτωση που πληρούνται οι συνθήκες η μεταβλητή `i` μηδενίζεται και στέλνεται στον Twitter User με screen name“arlabrin” το μήνυμα που λήφθηκε από τον αισθητήρα κίνησης.

Εάν ληφθεί μήνυμα “motion detected” αλλά η τιμή της μεταβλητής `i` είναι μικρότερη του 3 τότε απλά αυξάνεται η τιμή της. Συμπερασματικά, η εφαρμογή που μόλις περιγράφηκε ειδοποιεί το χρήστη σε κάθε τρίτη κίνηση που ανιχνεύει ο αισθητήρας κίνησης.

Στο πεδίο clean up απενεργοποιείται η υπηρεσία ειδοποίησης κίνησης.

6.4.3 Εφαρμογή ΙΙΙ

Ακόμα μια εφαρμογή θα μπορούσε να είναι η εξής:

```
init
    message l= "";
    sendmsg("MotionDetector_", "turn on notification");

body
    case((true)&&recvmsg("MotionDetector_","motion detected",l)){
        sendmsg("cam_device", "capture image");
    }
    case((true)&& recvmsg("cam_device", "photo:")){
          sendmsg("arlabrin", l);
    }

clean up
    sendmsg("MotionDetector_", "turn off notification");
```

Στο πεδίο init ενεργοποιείται η υπηρεσία ειδοποίησης κίνησης και δηλώνεται η μεταβλητή `l` που θα χρησιμοποιηθεί στο πεδίο body. Στο πεδίο body, όπου εκτελούνται ενέργειες ανάλογα με το μήνυμα που λήφθηκε, έχουμε δύο δομές case. Η πρώτη δομή καθορίζει τις ενέργειες στην περίπτωση που ληφθεί μήνυμα "motion detected" από τον αισθητήρα κίνησης και αποτελούνται από την αποστολή μηνύματος στην κάμερα για λήψη φωτογραφίας. Η δεύτερη δομή καθορίζει τις ενέργειες όταν ληφθεί μήνυμα από την κάμερα με το link της φωτογραφίας που ζητήθηκε από την πρώτη δομή case και αποτελούνται από την αποστολή του μηνύματος αυτού στον Twitter User με screen name "arlabrin". Τέλος στο πεδίο clean up απενεργοποιείται η υπηρεσία ειδοποίησης κίνησης.

6.4.4 Εφαρμογή IV

Η παρακάτω εφαρμογή αποτελεί παράδειγμα χρήσης του αισθητήρα θερμοκρασίας:

```
init
message l= "";
var temp=0;
sendmsg("SensorTemp", "temperature status request");

body
case((true)&&recvarmsg("SensorTemp ","the temperature is", l,
temp)){
    if(temp>30){
      sendmsg("arlabrin", "the temperature is high");
    }

}

clean up
```

Στο πεδίο init δηλώνονται οι μεταβλητές `l` και `temp` που θα χρησιμοποιηθούν στο πεδίο body και στέλνεται μήνυμα στον αισθητήρα θερμοκρασίας για ενημέρωση σχετικά με την τρέχουσα θερμοκρασία. Στο πεδίο body υπάρχει δομή case όπου όταν ληφθεί μήνυμα με την θερμοκρασία αποθηκεύεται στην μεταβλητή `temp` η τιμή της θερμοκρασίας που αναφέρθηκε . Στη συνέχεια ελέγχεται η τιμή αυτή και αν ξεπερνά τους 30 βαθμούς στέλνεται το μήνυμα "the temperature is high" στον Twitter User με screen name "arlabrin".

6.4.5 Εφαρμογή V

Η επόμενη εφαρμογή συνδυάζει τον αισθητήρα κίνησης με την μεταβλητή time.

```
init
    time t;
    message l= "";
    sendmsg("MotionDetector_", "turn on notification");

body
    case(((t>00:00)&&(t<06:00))&&recvmsg("MotionDetector_","motio
    ndetected",l)){
        sendmsg("cam_device", "capture image");
      }
    case((true)&&recvmsg("cam_device","photo",l)){
        sendmsg("arlabrin", l);
      }

clean up
    sendmsg("MotionDetector_", "turn off notification");
```

Αρχικά δηλώνονται οι μεταβλητές `time` και `l` στο πεδίο init και ενεργοποιείται η υπηρεσία ειδοποίησης σε περίπτωση ανίχνευσης κίνησης του αισθητήρα κίνησης. Αν η ώρα είναι μεταξύ 00:00 και 06:00 το πρωί (η ώρα γράφεται σε 24 σύστημα) και ανιχνευτεί κίνηση τότε στέλνεται μήνυμα στην κάμερα για λήψη φωτογραφίας. Μόλις ληφθεί το μήνυμα με το link της φωτογραφίας προωθείται στον Twitter User με screen name "arlabrin". Στο πεδίο clean up απενεργοποιείται η υπηρεσία ειδοποίησης κίνησης.

6.4.6 Εφαρμογή VI

Η τελευταία εφαρμογή που παρατίθεται ως παράδειγμα χρησιμοποιεί μόνο τον αισθητήρα θερμοκρασίας και θεωρητικά θα μπορούσε να αποτελέσει αυτοματισμό για τον κλιματισμό ενός σπιτιού.

```
init
sendmsg("SensorTemp", "turn on notification -<15-");
message l= "";
var temp=0;
body
case((true)&&recvarmsg("SensorTemp","the temperature
is",l,temp)){
   if(temp<15){
     sendmsg("SensorTemp", "turn off notification");
     sendmsg("arlabrin", "temperature is low" );
     sendmsg("SensorTemp", "turn on notification ->20-");
   }
   else{
     sendmsg("SensorTemp", "turn off notification");
     sendmsg("arlabrin", "temperature is normal" );
     sendmsg("SensorTemp", "turn on notification -<15-");
   }
}

clean up
sendmsg("SensorTemp", "turn off notification");
```

Αρχικά ενεργοποιείται η υπηρεσία ειδοποίησης θερμοκρασίας του αισθητήρα θερμοκρασίας και δηλώνονται οι μεταβλητές `l` και `temp` που θα χρησιμοποιηθούν στο πεδίο body. Εσωτερικά του πεδίου body υπάρχει δομή case που αναμένει μηνύματα ειδοποίησης του αισθητήρα θερμοκρασίας. Το μεταβλητό μέρος των μηνυμάτων αποθηκεύεται στην μεταβλητή `temp` και στη συνέχεια γίνεται χρήση της δομής `if` για να ελεγχθεί η τιμή του. Αν η θερμοκρασία είναι μικρότερη των 15 βαθμών τότε ενημερώνεται ο Twitter User με screen name "arlabrin", τερματίζεται η συγκεκριμένη υπηρεσία ειδοποίησης και ορίζεται μια νέα με κατώτατο όριο τη θερμοκρασία των 20 βαθμών, διαφορετικά ενημερώνεται ο χρήστης "arlabrin", τερματίζει η υπηρεσία ειδοποίησης και ορίζεται μια νέα με κατώτατο όριο τους 15 βαθμούς.

Η παραπάνω εφαρμογή ειδοποιεί τον χρήστη πότε η θερμοκρασία πέφτει κάτω του επιτρεπτού ορίου που έχει οριστεί. Όταν ενημερωθεί ο χρήστης ότι η θερμοκρασία δεν είναι μέσα στα επιτρεπτά όρια τότε θα κάνει τις απαραίτητες ενέργειες (π.χ. θα ενεργοποιήσει το σύστημα κλιματισμού) και όταν η θερμοκρασία επανέλθει στα επιθυμητά

για το χρήστη επίπεδα θα σταλεί και πάλι μήνυμα ενημέρωσης (ώστε να απενεργοποιήσει το σύστημα κλιματισμού).

7 Μεταγλωττιστής

Ο χρήστης, όπως έχουμε ήδη πει, αναπτύσσει την εφαρμογή του στα πεδία init, body και clean up. Η εφαρμογή του χρήστη για να μπορέσει να εκτελεστεί μετατρέπεται σε πρόγραμμα Java και η μετατροπή αυτή γίνεται μέσω του μεταγλωττιστής που έχει δημιουργηθεί. Όπως σε όλους του μεταγλωττιστές τα βασικά στάδια από τα οποία θα περάσει ο κώδικας που έχει γραφεί στην Γλώσσα Προγραμματισμού που αναπτύχθηκε μέχρι να πάρει την τελική του μορφή ως Java κλάση είναι η Λεκτική Ανάλυση, η Συντακτική Ανάλυση, η Σημασιολογική Ανάλυση και το στάδιο Παραγωγής Τελικού Κώδικα.

Εικόνα 10: Στάδια της Διαδικασίας Μεταγλώττισης

Όπως έχει ήδη αναφερθεί το Runtime είναι ο κεντρικός διαχειριστής του συστήματος που αναπτύχθηκε και οι εφαρμογές για να επιτελούν τις λειτουργίες τους πρέπει να επικοινωνούν με αυτό. Για να διασφαλιστεί η επιτυχής επικοινωνία μεταξύ του Runtime και των εφαρμογών υπάρχουν συγκεκριμένες συμβάσεις σύμφωνα με τις οποίες πρέπει να δημιουργείται ο τελικός εκτελέσιμος κώδικας κάθε εφαρμογής αυτοματισμού. Κάθε εφαρμογή περιέχει τις μεθόδους receiveNewMessage(), start(), stop(), init(), cleanup() και run() που υλοποιούν τα Interfaces AppInterface και Runnable. Ο κώδικας εσωτερικά των μεθόδων receiveNewMessage(), start() και stop() είναι κοινός σε όλες τις εφαρμογές. Η μέθοδος receiveNewMessage() καλείται από το Runtime όταν εκείνο εντοπίσει ότι υπάρχει Direct Message για την συγκεκριμένη εφαρμογή αυτοματισμού και το μήνυμα αποθηκεύεται σε buffer τύπου BlockingQueue. Η μέθοδος start() καλείται από το Runtime για να αρχίσει η εκτέλεση της εφαρμογής ενώ η stop() για να τερματιστεί η εκτέλεσή της. Τέλος η μέθοδος run() καλείται από την init() ώστε να ξεκινήσει η εκτέλεση του νήματος της εφαρμογής και ο τερματισμός του επιτελείται μέσω της μεθόδου stop().

Όταν η εφαρμογή αυτοματισμού πρέπει να στείλει μήνυμα σε κάποιο Twitter User ή πόρο θα καλείται η μέθοδος sendMessage() του Runtime. Επίσης, κατά τη σημασιολογική

ανάλυση, η κλάση SemanticAnalysis.java καλεί τη μέθοδο existFriendship() του Runtime ώστε να διαπιστωθεί αν υπάρχει αμοιβαίο follow με τον Twitter User στον οποίο πρόκειται να σταλεί ή από τον οποίο αναμένεται κάποιο μήνυμα.

Στις επόμενες υποενότητες θα γίνει αναφορά σε καθένα από τα στάδια που συνθέτουν τον μεταγλωττιστής.

7.1 Λεκτικός και Συντακτικός Αναλυτής

Τα στάδια Λεκτικής και Συντακτικής Ανάλυσης υλοποιήθηκαν με το εργαλείο **JavaCC [11]**. Η βιβλιοθήκη JavaCC, όπως έχει αναφερθεί σε προηγούμενο κεφάλαιο, παράγει Λεκτικούς και Συντακτικούς Αναλυτές σε γλώσσα Java, ενώ το εργαλείο jjTree που τη συνοδεύει δημιουργεί το δέντρο συντακτικής ανάλυσης . Η JavaCC υποστηρίζει ανοδική ανάλυση (Top-Down) χωρίς να επιτρέπει την αριστερή αναδρομή και δημιουργεί αναλυτές για γραμματικές LL(k) αφού υποστηρίζει και πρόβλεψη (με χρήση της λέξης LOOKAHEAD(αριθός_συμβόλων_προς_εξέταση)). Επίσης δεν επιτρέπει την αριστερή αναδρομή αλλά παρέχει τη δυνατότητα περάσματος ορισμάτων κατά τη Συντακτική Ανάλυση.

Η διαδικασία που ακολουθείται μέχρι την τελική δημιουργία του Λεκτικού και Συντακτικού Αναλυτή φαίνεται στην Εικόνα 11. Το εργαλείο jjTree είναι ένας προ-επεξεργαστής της JavaCC. Ουσιαστικά η λειτουργία που επιτελείται από το jjTree είναι η πρόσθεση εντολών για τη δημιουργία του δέντρου συντακτικής ανάλυσης στο αρχικό αρχείο που έχει δημιουργήσει ο χρήστης. Έτσι ο χρήστης δημιουργεί ένα αρχείο με κατάληξη .jjt το οποίο λαμβάνεται ως είσοδο από το εργαλείο jjTree που με τη σειρά του δημιουργεί ένα αρχείο με .jj κατάληξη. Το νέο αρχείο παραλαμβάνεται από τη JavaCC και δημιουργείται ο τελικός αναλυτής..

Εικόνα 11: Διαδικασία Δημιουργίας Λεκτικού και Συντακτικού Αναλυτή

Η μορφή του αρχείου .jjt είναι η παρακάτω:
(Ίδια μορφή έχει και το αρχείο .jj αλλά με επιπλέον εντολές για τη δημιουργία του δέντρου συντακτικής ανάλυσης.)

options{
// Ρύθμιση επιλογών του αναλυτή που θα δημιουργηθεί
}

PARSER_BEGIN (όνομα_κύριας_κλάσης)
// Κώδικας Java (Δήλωση της κύριας κλάσης και του κώδικα που θα περιέχεται σε αυτή)
PARSER_END (όνομα_κύριας_κλάσης)

SKIP:
{
//Αναφέρονται οι χαρακτήρες που θα αγνοούνται από τον αναλυτή
}

TOKEN:{
//Δήλωση δεσμευμένων λέξεων του μεταγλωττιστή
}

SPECIAL_TOKEN{
// Δήλωση ειδικών δεσμευμένων λέξεων που θα αγνοούνται από τον μεταγλωττιστή
}

Δήλωση μη-τερματικών. (Η μορφή των μη-τερματικών είναι αυτή των δηλώσεων μεθόδων στη Java (τύπος_επιστροφής όνομα_τερματικού(παράμετροι)). Εσωτερικά των μη-τερματικών μπορούν να οριστούν μεταβλητές και να κληθούν άλλα μη-τερματικά. Επίσης μπορεί να συμπεριληφθεί και κώδικας Java γραμμένος μέσα σε άγκιστρα ({ })Τέλος, για την αναπαράσταση αναδρομικών εκφράσεων χρησιμοποιούνται χαρακτηριστικά του συμβολισμού BNF (,+,?) κτλ.)*

7.2 Σημασιολογικός Αναλυτής

Η JavaCC σε συνδυασμό με το εργαλείο jjTree δημιουργεί εκτός από τον Λεκτικό Αναλυτή και τον Συντακτικό Αναλυτή, και ένα Δέντρο Συντακτικής Ανάλυσης, μια κλάση για κάθε ένα μη-τερματικό που δηλώθηκε στο αρχείο εισόδου και ένα Visitor Pattern για τη προσπέλαση του Συντακτικού Δέντρου. Έτσι η σημασιολογική ανάλυση επιτελείται μέσω της κλάσης SemanticAnalysis.java και του Visitor Pattern.

Ο σημασιολογικός αναλυτής που δημιουργήθηκε είναι στατικός, δηλαδή περιορίζεται μόνο στον έλεγχο και για τον εντοπισμό σημασιολογικών λαθών ενός προγράμματος ενώ δεν αποδίδει ερμηνεία στο πρόγραμμα (ενέργεια που κάνουν οι δυναμικοί σημασιολογικοί αναλυτές). Κατά την σημασιολογική ανάλυση δημιουργείται επίσης ένας πίνακας συμβόλων με τη χρήση του οποίου γίνονται οι απαραίτητοι έλεγχοι.

Ένας από τους ελέγχους που επιτελούνται είναι ο έλεγχος τύπων, διαδικασία κατά την οποία ελέγχονται οι τύποι των μεταβλητών και αν είναι συμβατοί με τις τιμές που ανατίθενται σε αυτές. Επίσης γίνεται έλεγχος ύπαρξης ονομάτων, δηλαδή εντοπίζεται αν έχει δηλωθεί μια μεταβλητή πριν τη χρήση της. Ένας ακόμα έλεγχος είναι ο έλεγχος μοναδικότητας αφού δύο μεταβλητές δεν επιτρέπεται να έχουν το ίδιο όνομα. Τέλος γίνεται έλεγχος ροής ώστε να διαπιστωθεί η ορθότητα χρήσης της πρότασης break (πρέπει να βρίσκεται μόνο εσωτερικά των δομών while και case).

7.3 Παραγωγή τελικού κώδικα

Το στάδιο Παραγωγής Τελικού Κώδικα αποτελείται από δύο επιμέρους στάδια, την Παραγωγή Κώδικα για καθολικές μεταβλητές και την Παραγωγή Κώδικα για τις υπόλοιπες λειτουργίες της εφαρμογής. Αρχικά εκτελείται η Παραγωγή Κώδικα για τις καθολικές μεταβλητές και ακολουθεί η Παραγωγή Κώδικα για το υπόλοιπο πρόγραμμα.

Για την Παραγωγή Τελικού Κώδικα χρησιμοποιείται ο πίνακας συμβόλων, το Visitor Pattern και οι κλάσεις CodeGeneratorForGlobelVariables.java και CodeGenerator.java.

Η νέα εφαρμογή που δημιουργείται αποθηκεύεται στο πακέτο apps και συμπεριλαμβάνει τις κλάσεις DirMessage.java και MyRuntime.java του πακέτου runtime.

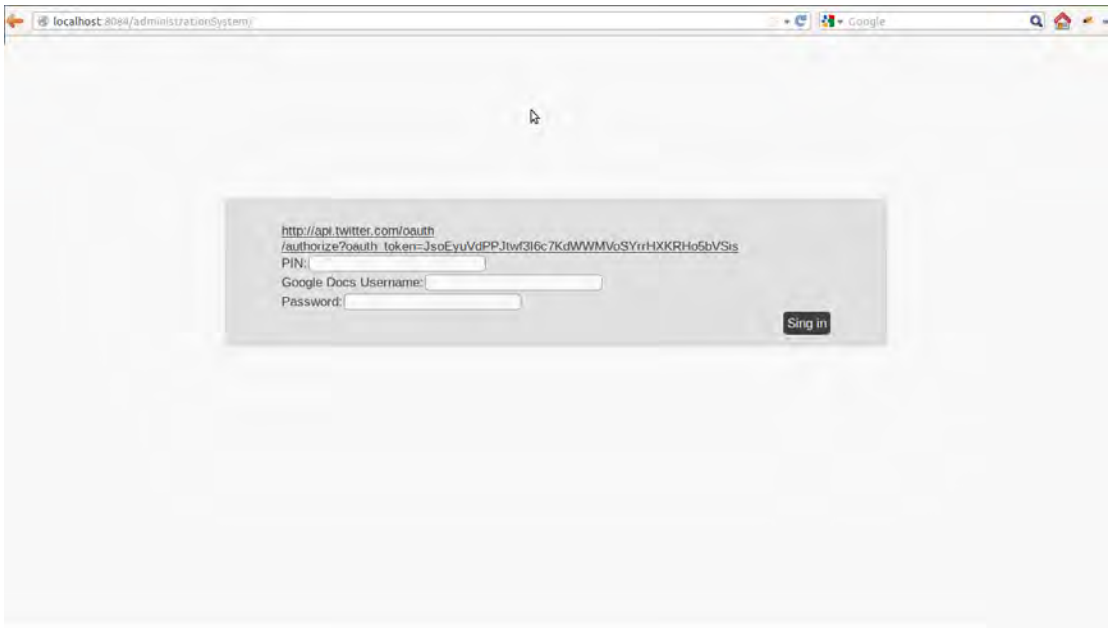
Η μορφή του τελικού αρχείου εξόδου είναι η παρακάτω:

```
/* ----------------------------------------------------------------------------------------------------*/

package apps;

import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.BlockingQueue;
import java.util.logging.Level;
import java.util.logging.Logger;

import runtime.DirMessage;
import runtime.MyRuntime;

/**
*
*@author iro
*/

public  class app_name  implements AppInterface, Runnable{
      public app_name (){ }
      private Thread nth;
      DirMessage dirMessage;
      static BlockingQueue<DirMessage> buf=new
      ArrayBlockingQueue<DirMessage>(100);

      @Override
      public void receiveNewMessage(DirMessage msg){
            try {
                  buf.put(msg);
            } catch (InterruptedException ex) {
                  Logger.getLogger(first.class.getName()).log(Le
            vel.SEVERE, null, ex);
            }
      }

      @Override
      public void start() {
            app_name myapp=new app_name();
            nth=new Thread(myapp);
            init();
            nth.start();
```

```
            return;
      }

      @Override
      public void stop() {
            nth.stop();
            cleanup();
            return;
      }

      @Override
      public void init(){
      //Κώδικας που έχει ορίσει ο χρήστης στο πεδίο init
      }

      @ Override
      public void cleanup(){
      //Κώδικας που έχει ορίσει ο χρήστης στο πεδίο cleanup
      }

      @ Override
      public void run(){
      //Κώδικας που έχει ορίσει ο χρήστης στο πεδίο body
      }
}

/* ----------------------------------------------------------------------------------------------------*
```

8 Διεπαφή Χρήστη

Η αρχική εικόνα της HTML σελίδας με την οποία έρχεται σε επαφή ο χρήστης κατά την είσοδο για πρώτη φορά στο σύστημα είναι η παρακάτω:

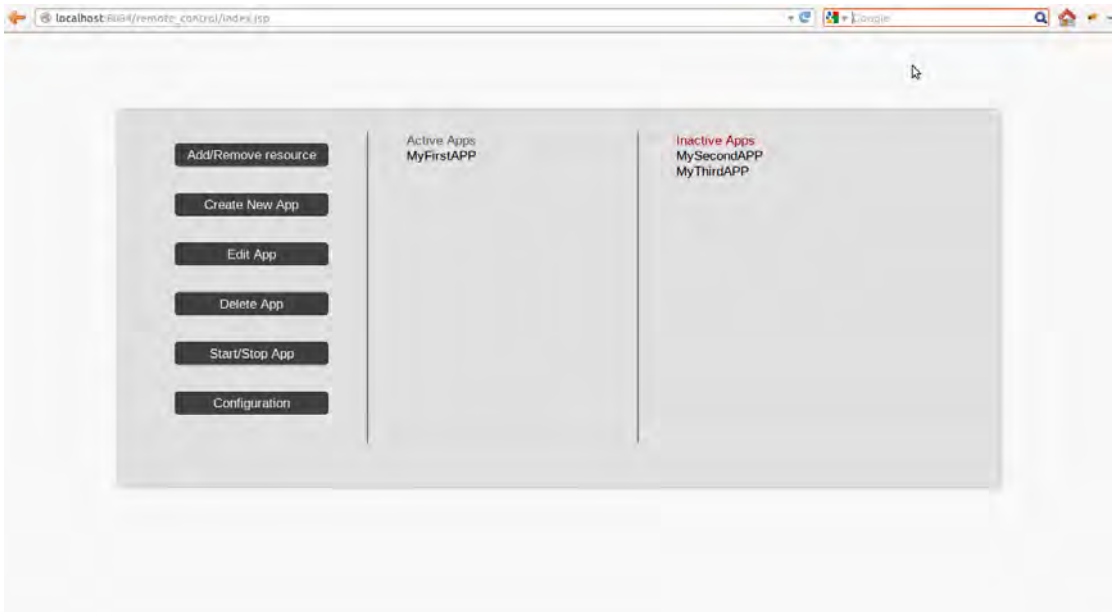


Ο χρήστης πρέπει να συμπληρώσει το Goggle Docs Username και το Google Docs password ώστε να μπορεί το σύστημα να έχει πρόσβαση στον Google Docs Account του χρήστη. Είναι σημαντικό να αναφερθεί ότι πρέπει να δοθεί πρόσβαση στο ίδιο Google Docs Account με αυτό που έχει δοθεί στους πόρους ώστε να μπορεί να γίνει η επιθυμητή ανταλλαγή αρχείων. Επίσης, ο χρήστης πρέπει να μεταβεί στο link που υποδεικνύεται από το παράθυρο διαλόγου ώστε να συνδεθεί στο Twitter Account του συστήματος διαχείρισης και να εγκρίνει την πρόσβαση σε αυτό της client εφαρμογής που δημιουργήθηκε. Μόλις ο χρήστης δώσει την έγκριση εμφανίζεται ένας κωδικός που πρέπει να γραφεί στο πεδίο pin της εικόνας.

Στη συνέχεια αφού ολοκληρωθεί επιτυχώς η παραπάνω διαδικασία ο χρήστης αποκτά πρόσβαση σε όλες τις δυνατότητες του συστήματος διαχείρισης των οποίων η περιγραφή ακολουθεί στις επόμενες υποενότητες.

8.1 Αρχική σελίδα

Η κεντρική σελίδα του συστήματος είναι η παρακάτω:

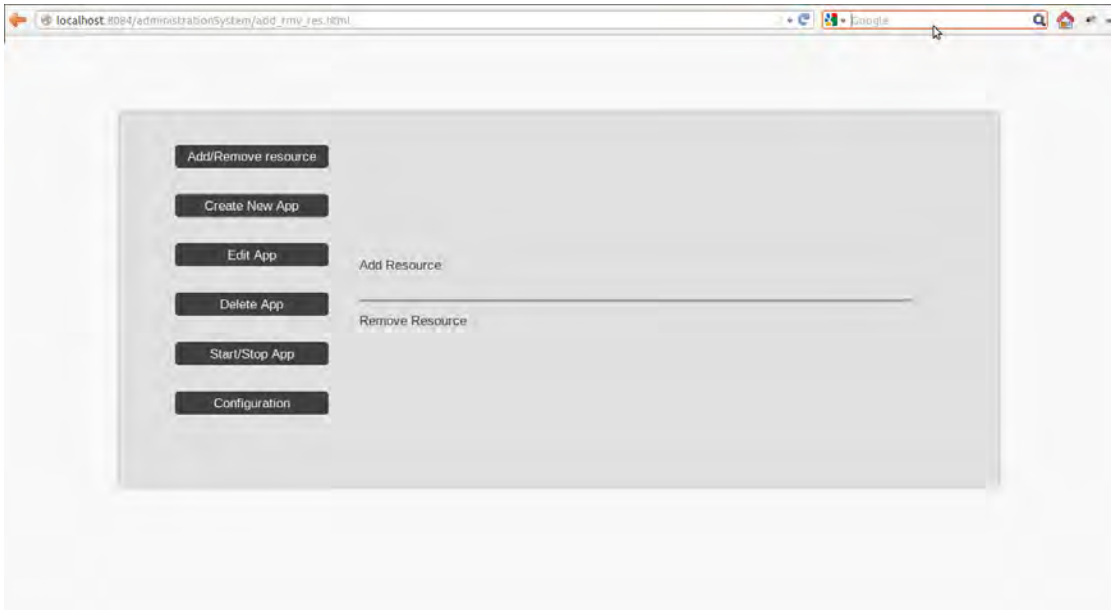


Αριστερά υπάρχει το μενού όπου ο χρήστης μπορεί να επιλέξει ανάμεσα στην πρόσθεση/ αφαίρεση κάποιου πόρου/Twitter endpoint (ουσιαστικά follow/unfollow), στη δημιουργία νέας εφαρμογής, στην τροποποίηση κάποιας ήδη υπάρχουσας εφαρμογής , στη διαγραφή μιας εφαρμογής, στην ενεργοποίηση ή απενεργοποίηση κάποιας εφαρμογής και τέλος, στη ρύθμιση των παραμέτρων του συστήματος.

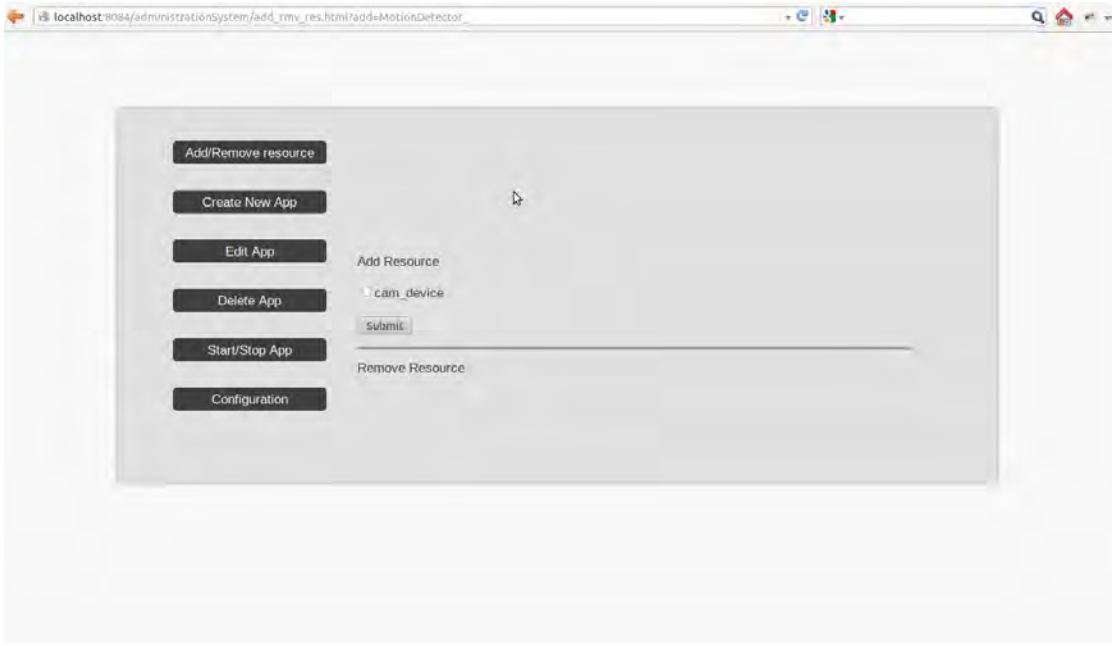
Στο κέντρο και στη δεξιά πλευρά της οθόνης φαίνονται δύο λίστες. Η μια περιέχει τις ενεργές εφαρμογές και η άλλη τις μη ενεργές.

8.2 Εισαγωγή/ Διαγραφή Πόρου-Twitter endpoint

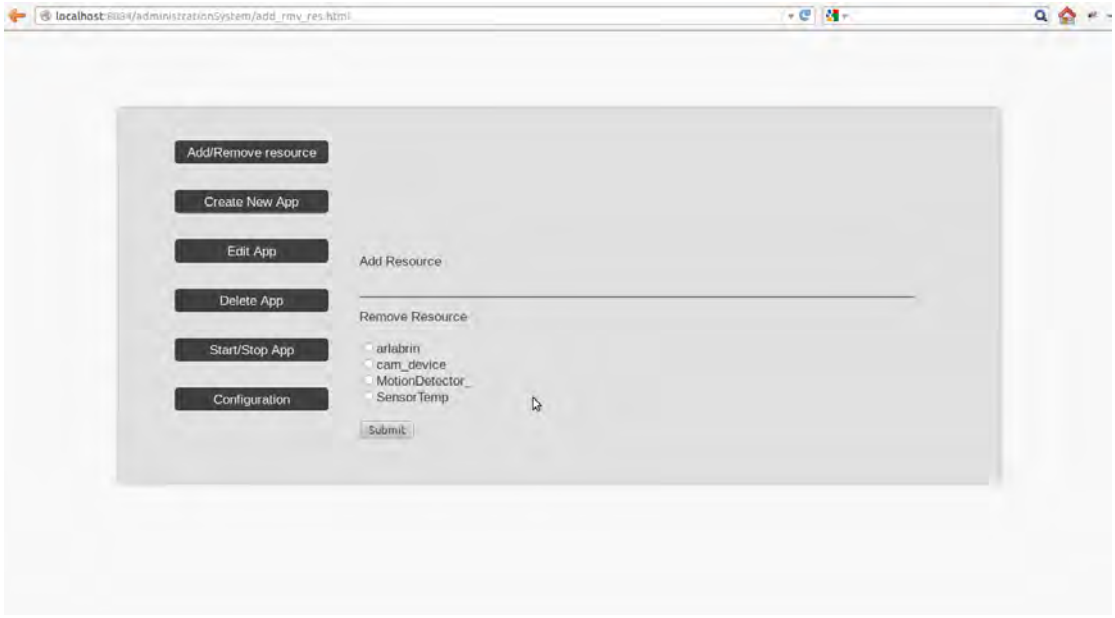
Αν ο χρήστης επιλέξει από το μενού “Add/Remove Resource” θα εμφανιστεί η παρακάτω σελίδα:



Επιλέγοντας “Add Resource” εμφανίζεται ένα παράθυρο όπου ο χρήστης πρέπει να επιλέξει κάποιο από τους προτεινόμενους πόρους-Twitter endpoints για να κάνει follow. Πατώντας “Submit” το Twitter Account του συστήματος διαχείρισης κάνει follow το συγκεκριμένο χρήστη και πλέον υπάρχει η δυνατότητα ανταλλαγής Direct Message, άρα η δυνατότητα επικοινωνίας.

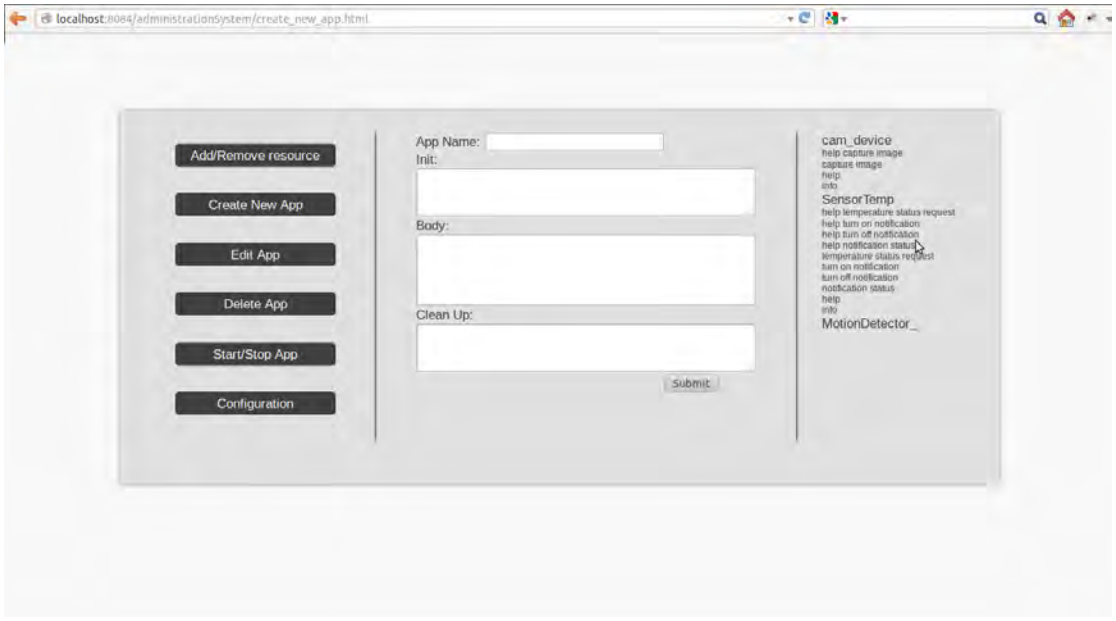


Αν ο χρήστης επιλέξει να αφαιρέσει κάποιο Twitter User , επιλέγοντας “Remove Resource” εμφανίζονται οι υποψήφιοι προς unfollow. Ο χρήστης επιλέγει κάποιο από αυτούς και πατάει “Submit” για να ολοκληρωθεί η διαδικασία.



8.3 Δημιουργία νέας εφαρμογής

Επιλέγοντας ο χρήστης από το μενού “Create New App” εμφανίζονται τα πεδία init, body και clean up εσωτερικά των οποίων θα γραφεί ο κώδικας στην Γλώσσα Προγραμματισμού που αναπτύχθηκε ειδικά για την δημιουργία των αυτοματισμών.

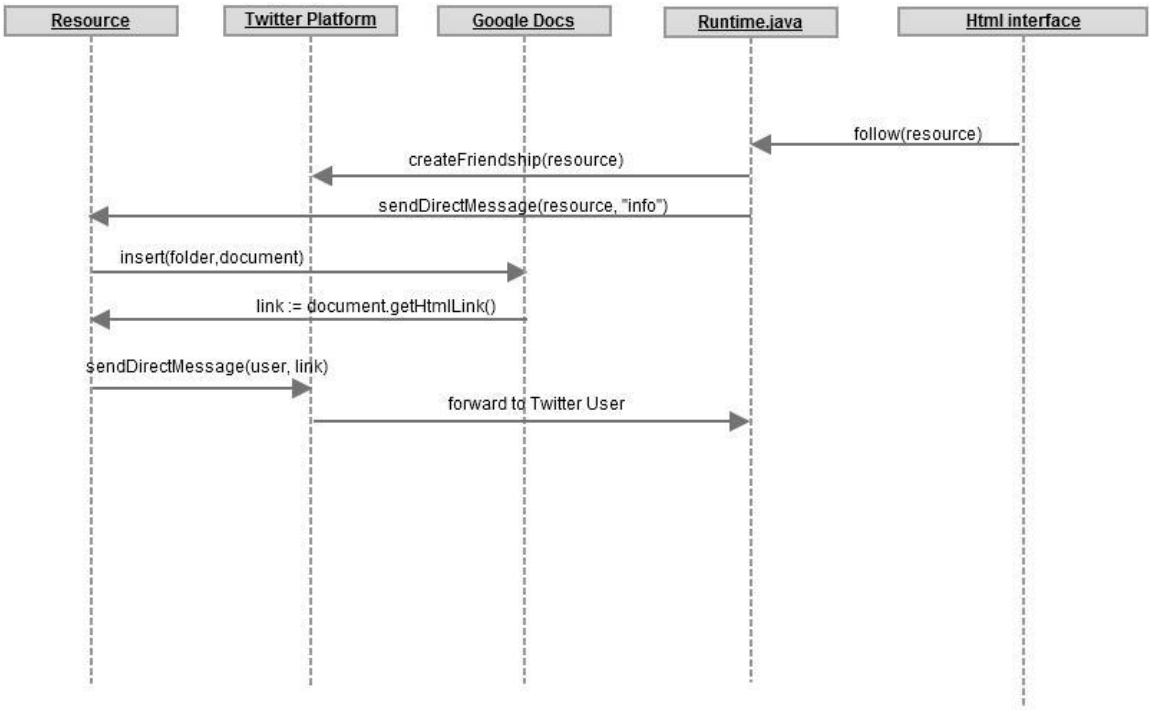


Στην δεξιά πλευρά εμφανίζονται οι διαθέσιμοι πόροι και οι εντολές που δέχονται

ώστε να διευκολυνθεί ο χρήστης κατά τον προγραμματισμό του αυτοματισμού.

Το Περιβάλλον Προγραμματισμού Αυτοματισμών μόλις κάνει follow έναν νέο πόρο στέλνει σε αυτόν την εντολή “info”. Ο πόρος δέχεται την εντολή και στέλνει ως απάντηση link σε αρχείο xml όπου υπάρχουν πληροφορίες σχετικά με τις εντολές που υποστηρίζει και τις απαντήσεις σε αυτές. Το Περιβάλλον Προγραμματισμού Εφαρμογών “κατεβάζει” και αποθηκεύει το xml αρχείο για μελλοντική χρήση.

Στην Εικόνα 12 φαίνεται το ακολουθιακό διάγραμμα για τη διαδικασία ανταλλαγής του xml αρχείου.

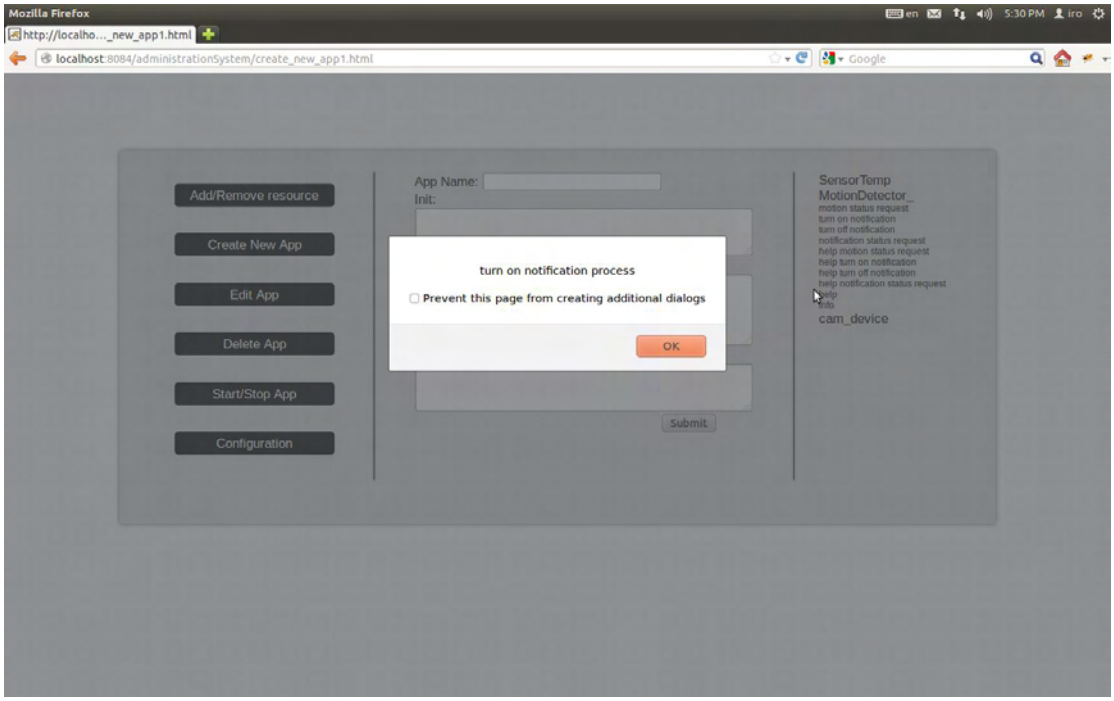


Εικόνα 12: Ακολουθιακό Διάγραμμα για τη διαδικασία ανταλλαγής του xml αρχείου

Όταν ο χρήστης επιλέξει από το μενού “Create New App” ή “Edit App” ανακτώνται από το xml αρχείο κάθε πόρου οι υποστηριζόμενες εντολές και εμφανίζονται υπό τη μορφή λίστας στην δεξιά πλευρά της οθόνης.

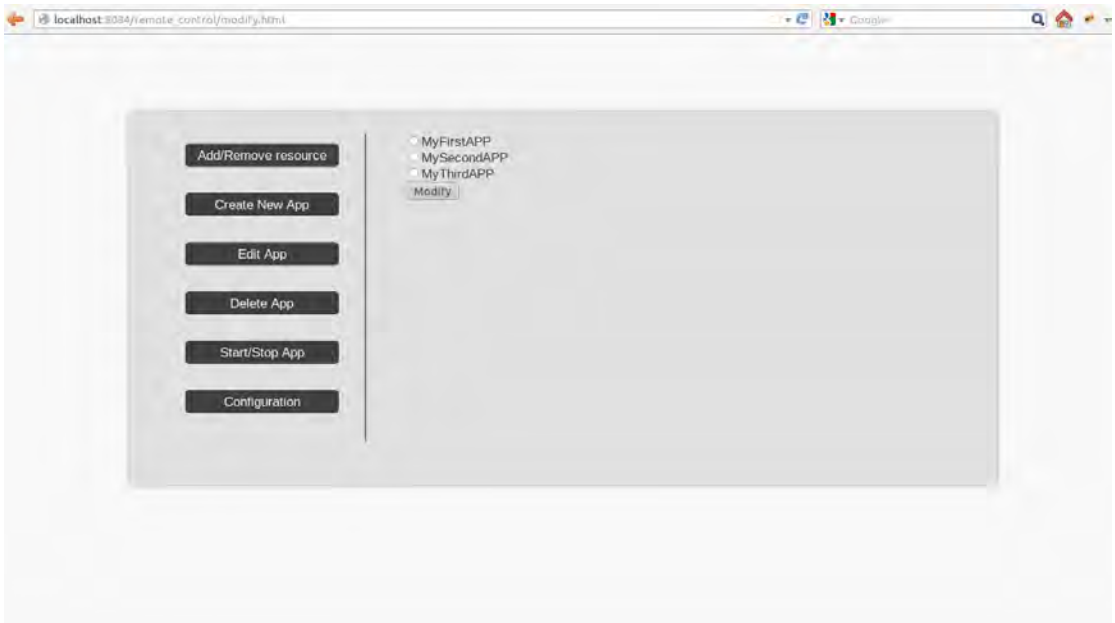
Το xml αρχείο χρησιμοποιείται και κατά τη διαδικασία μεταγλώττισης. Αν εσωτερικά της δομής case αναμένεται μήνυμα με μεταβλητό μέρος ελέγχεται αν ο τύπος του μεταβλητού μέρους συμπίπτει με τον τύπο της μεταβλητής στην οποία θα γίνει η αποθήκευσή του.

Τέλος, επιλέγοντας ο χρήστης κάποια από τις υποστηριζόμενες εντολές του πόρου που εμφανίζονται στην δεξιά πλευρά εμφανίζεται μήνυμα σχετικά με την απάντηση που στέλνει ο πόρος δεχόμενος την συγκεκριμένη εντολή.



8.4 Τροποποίηση εφαρμογής

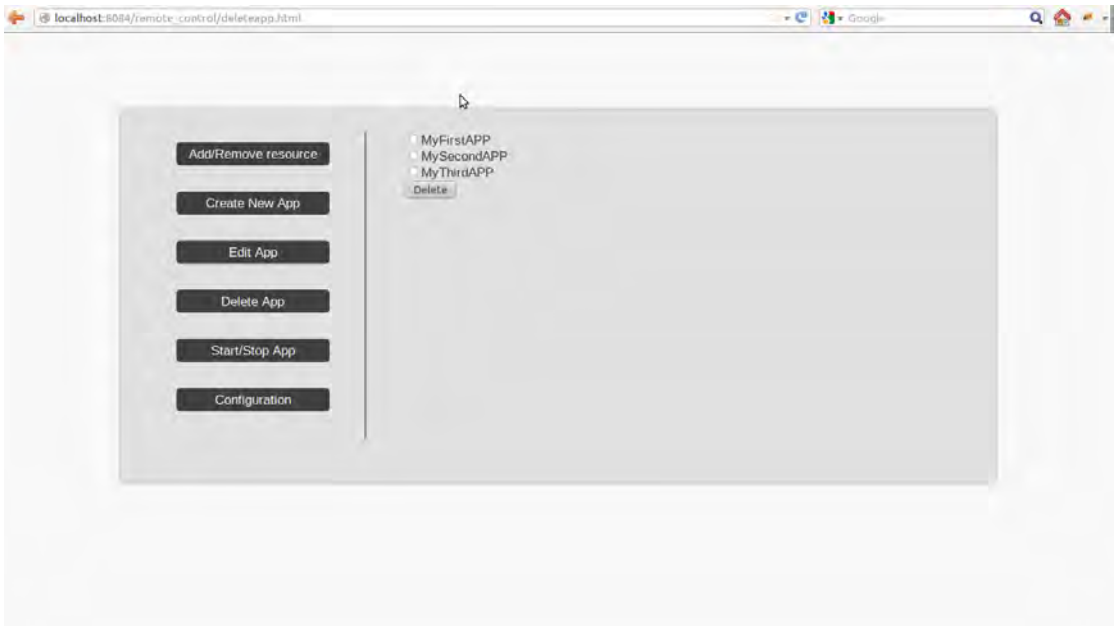
Σε περίπτωση που υπάρχει ανάγκη τροποποίησης ή διόρθωσης κάποιας υπάρχουσας εφαρμογής αυτοματισμού, ο χρήστης μπορεί να επιλέξει από το μενού “Edit App” και θα εμφανιστεί λίστα με τις εφαρμογές ώστε να καθοριστεί ποια από αυτές θα τροποποιηθεί.



Στη συνέχεια, αφού γίνει η επιλογή εμφανίζεται στην οθόνη ο κώδικας που ήταν καταχωρημένος στη συγκεκριμένη εφαρμογή ώστε να τροποποιηθεί από τον χρήστη.

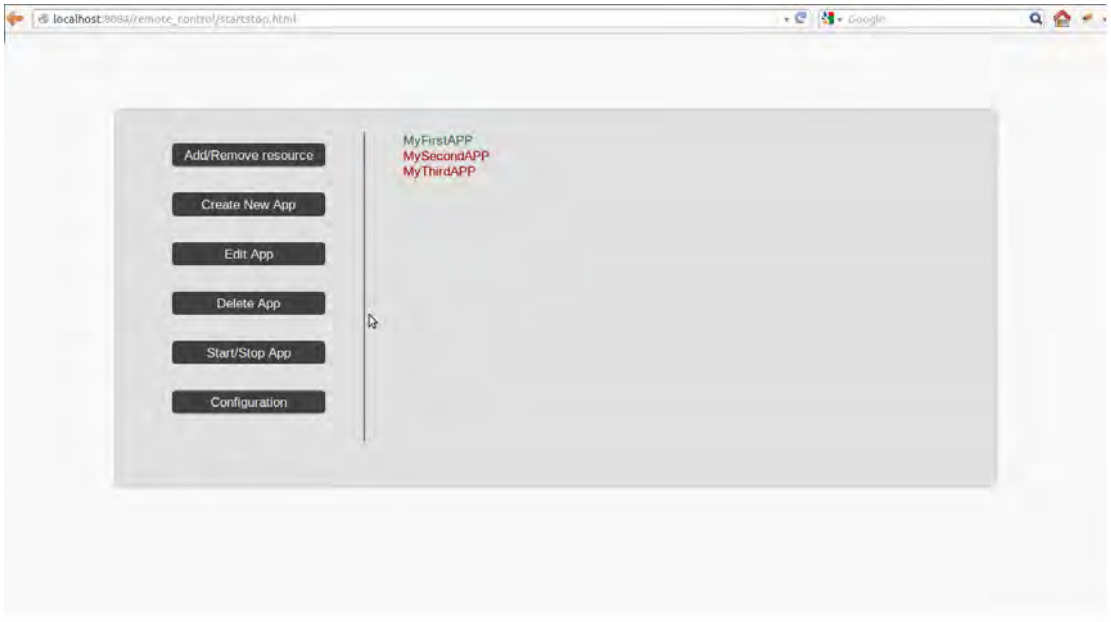
8.5 Διαγραφή εφαρμογής

Επιλέγοντας από το μενού “Delete App” εμφανίζεται μια λίστα με τις υπάρχουσες εφαρμογές από την οποία ο χρήστης θα επιλέξει την εφαρμογή που επιθυμεί να διαγράψει. Αν η εφαρμογή προς διαγραφή είναι ενεργή η διαγραφή δε μπορεί να πραγματοποιηθεί. Ο χρήστης πρέπει πρώτα να απενεργοποιήσει την εφαρμογή και στη συνέχεια να προχωρήσει στη διαγραφή.



8.6 Εκκίνηση/Τερματισμός εφαρμογής

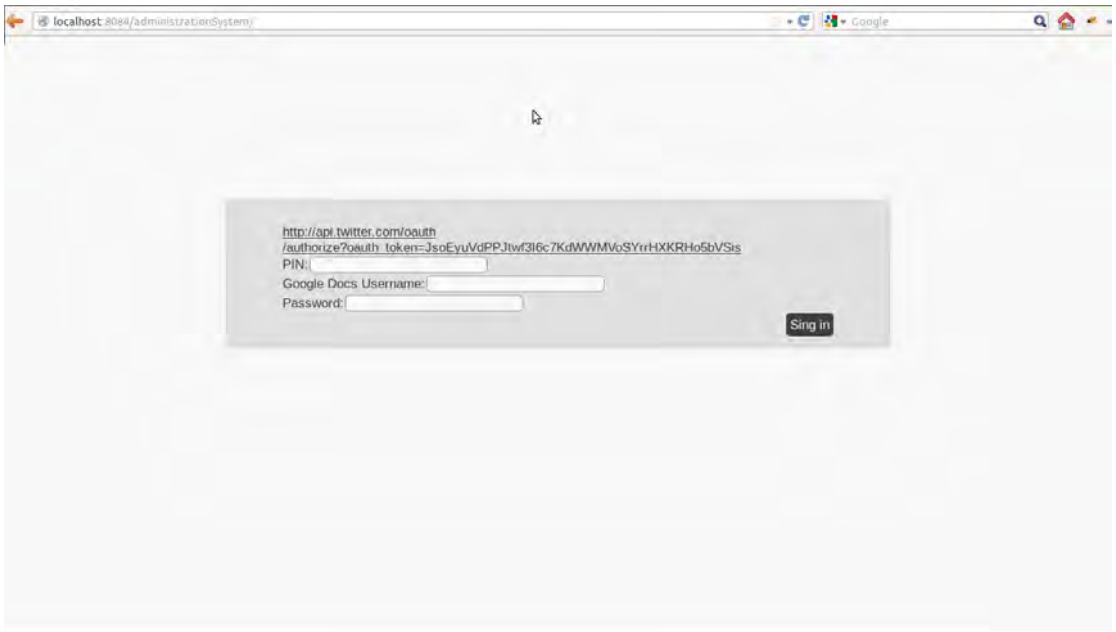
Επιλέγοντας “Start/Stop App” ο χρήστης μπορεί να ενεργοποιεί ή να απενεργοποιεί της εφαρμογές που έχει δημιουργήσει.



Με πράσινο χρώμα είναι οι ενεργές εφαρμογές ενώ με κόκκινο οι μη ενεργές. Κάνοντας κλικ πάνω σε μια εφαρμογή αυτομάτως αλλάζει η κατάστασή της από ενεργή σε μη ενεργή και το αντίθετο.

8.7 Configuration

Τέλος, δίνεται η δυνατότητα στο χρήστη να αλλάξει το Twitter Account και το Google Account του συστήματος διαχείρισης. Έτσι επιλέγοντας “configuration” εμφανίζεται το αρχικό παράθυρο και ο χρήστης καλείται να εισάγει τα νέα στοιχεία.



9 Επίλογος

9.1 Συμπεράσματα

Η παρούσα διπλωματική εργασία είχε ως κύριο στόχο να δώσει την δυνατότητα στο χρήστη να αλληλεπιδρά με απλό και φυσικό τρόπο με τους πόρους που έχει στη διάθεσή του και να τους συνδυάζει εύκολα ώστε να υλοποιήσει λειτουργίες αυτοματισμού. Δημιουργήθηκαν λοιπόν εικονικοί πόροι και υλοποιήθηκε ένα περιβάλλον προγραμματισμού εφαρμογών για τους πόρους αυτούς. Σημαντικό ρόλο στην εργασία αυτή αποτέλεσαν τα social media. Πιο συγκεκριμένα, χρησιμοποιήθηκε το social media Twitter ως δίαυλος επικοινωνίας των πόρων με το χρήστη και το περιβάλλον προγραμματισμού αυτοματισμών που αναπτύχθηκε.

9.2 Βελτιώσεις και Επεκτάσεις

Στο μέλλον θα μπορούσαν να γίνουν ορισμένες βελτιστοποιήσεις στο σύστημα που αναπτύχθηκε.

Αρχικά θα μπορούσαν να δημιουργηθούν πραγματικοί πόροι που θα αποτελούν συστατικά μέρη ενός περιβάλλοντος χώρου. Μέχρι στιγμής οι πόροι είναι εικονικοί και έχουν δημιουργηθεί απλώς για τις ανάγκες του demo της διπλωματικής εργασίας. Ωστόσο, θα ήταν ιδανική η δημιουργία ενός πραγματικού χώρου αποτελούμενο από συσκευές ικανές να αλληλεπιδράσουν με τον χρήστη.

Επίσης, σχετικά με την διεπαφή του χρήστη, θα μπορούσε να δημιουργηθεί ένα γραφικό περιβάλλον προγραμματισμού για διευκόλυνση του χρήστη. Έτσι, η Γλώσσα Προγραμματισμού που δημιουργήθηκε θα μπορούσε να επιτελέσει το ρόλο ενός ενδιάμεσου κώδικα μεταξύ της γραφικής αναπαράστασης του αυτοματισμού και του Java προγράμματος. Ένα γραφικό περιβάλλον προγραμματισμού θα βοηθούσε τους χρήστες που δεν είναι εξοικειωμένοι με τον προγραμματισμό να δημιουργήσουν εύκολα τις εφαρμογές που επιθυμούν.

Σχετικά με την Γλώσσα Προγραμματισμού Αυτοματισμών, θα μπορούσε να προστεθεί και μια δομή “timer” ώστε να δίνεται η δυνατότητα στον αυτοματισμό του χρήστη να εκτελεί περιοδικά κάποιες ενέργειες.

Τέλος, θα μπορούσαν να γίνουν τροποποιήσεις στο υπάρχον σύστημα ώστε να γίνει η επικοινωνία μέσω της πλατφόρμας του Twitter όσο το δυνατόν πιο αξιόπιστη. Έχει παρατηρηθεί ότι υπάρχουν στιγμές που η σύνδεση με το Twitter “πέφτει” (Το γεγονός αυτό δεν προκαλείται από κάποια αστοχία του συστήματος αλλά από την ίδια την πλατφόρμα του Twitter). Αν και η διαδικασία ανάκαμψης της σύνδεσης γίνεται άμεσα, αν σταλεί κάποιο μήνυμα μέσα στο συγκεκριμένο χρονικό διάστημα, στον πόρο του οποίου η σύνδεση δεν είναι καλή τότε δε θα το παραλάβει. Έτσι θα μπορούσε να γίνει κατάλληλος χειρισμός αυτής της περίπτωσης.

10 Appendix

ΤΕΧΝΟΛΟΓΙΕΣ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ

Η διεκπεραίωση της διπλωματικής εργασίας στηρίχθηκε στη χρήση αρκετών τεχνολογιών. Το σύστημα διαχείρισης πόρων που αναπτύχθηκε, καθώς και η προσομοίωση των πόρων δημιουργήθηκαν πάνω στο **NetBeans 7.0 [12]** (πλατφόρμα ανάπτυξης εφαρμογών) και στο μεγαλύτερο μέρος με τη γλώσσα προγραμματισμού **Java**. Επίσης χρησιμοποιήθηκαν οι γλώσσες **HTML** και **Javascript** για την υλοποίηση του client side του συστήματος.

Η βιβλιοθήκη **JavaCV [10]** που χρησιμοποιήθηκε αποτελεί ένα wrapper της βιβλιοθήκης OpenCV, που είναι γραμμένη στις γλώσσες C και C++, για τη γλώσσα Java. Η OpenCV αναπτύχθηκε από την Intel και είναι βιβλιοθήκη ελεύθερου λογισμικού για την επεξεργασία εικόνας. Για την προσομοίωση των πόρων της κάμερας και του αισθητήρα κίνησης χρησιμοποιήθηκε μια κάμερα συνδεδεμένη στον υπολογιστή και η πρόσβαση σε αυτή έγινε με τη χρήση της JavaCV.

Το **Flickr API [7]** χρησιμοποιήθηκε από την κάμερα ώστε να ανεβάζει τις φωτογραφίες που βγάζει όταν της ζητείται, το **Google Docs API [8]** χρησιμοποιήθηκε ώστε να δημοσιεύουν οι πόροι έγγραφα κατ' απαίτηση και το **Twitter API [6]** για την πρόσβαση στη πλατφόρμα του Twitter.

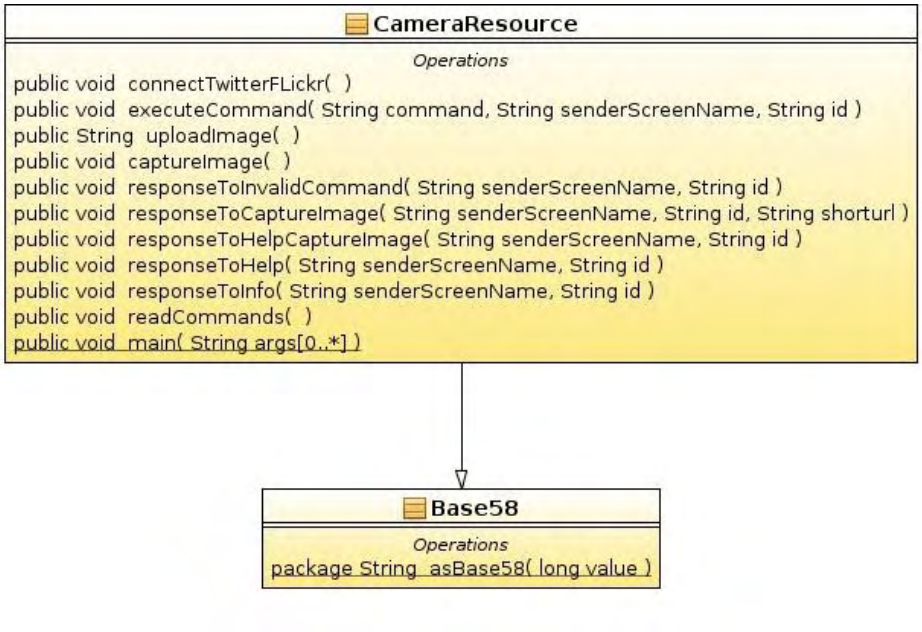
Για την υλοποίηση του μεταγλωττιστή (compiler) που μεταφράζει τους αυτοματισμούς του χρήστη από ψευδοκώδικα σε Java προγράμματα χρησιμοποιήθηκε η βιβλιοθήκη **JavaCC [11]**. Η JavaCC (Java Compiler Compiler) είναι βιβλιοθήκη ελεύθερου λογισμικού και χρησιμοποιείται για δημιουργία λεκτικών και συντακτικών αναλυτών. Μαζί με την JavaCC παρέχεται και το εργαλείο JJTree, το οποίο δημιουργεί το δέντρο συντακτικής ανάλυσης. Τα εργαλεία αυτά χρησιμοποιήθηκαν στην δημιουργία του μεταγλωττιστή.

Στο client side της σελίδας διαχείρισης χρησιμοποιήθηκε η τεχνολογία **jquery**. Το jquery είναι μια βιβλιοθήκη javascript που ολοένα και περισσότερα sites επιλέγουν να τη χρησιμοποιήσουν. Είναι συμβατή με όλους τους browsers ενώ στην απλή javascript χρειαζόταν ειδικός χειρισμός για να γίνει αυτό εφικτό. Με τη βιβλιοθήκη jquery ο προγραμματιστής έχει πολύ περισσότερες δυνατότητες και η σελίδα ή η εφαρμογή που δημιουργεί είναι σημαντικά πιο διαδραστική με τον χρήστη.

Τέλος οι κλήσεις στον server γίνονται μέσω της τεχνολογίας **AJAX** (**A**synchronous **J**avaScript **A**nd **X**ML). Το AJAX είναι αποτέλεσμα του συνδυασμού των τεχνολογιών javascript και xml και χρησιμοποιείται στην επικοινωνία της ιστοσελίδας με τον server.

ΚΑΜΕΡΑ

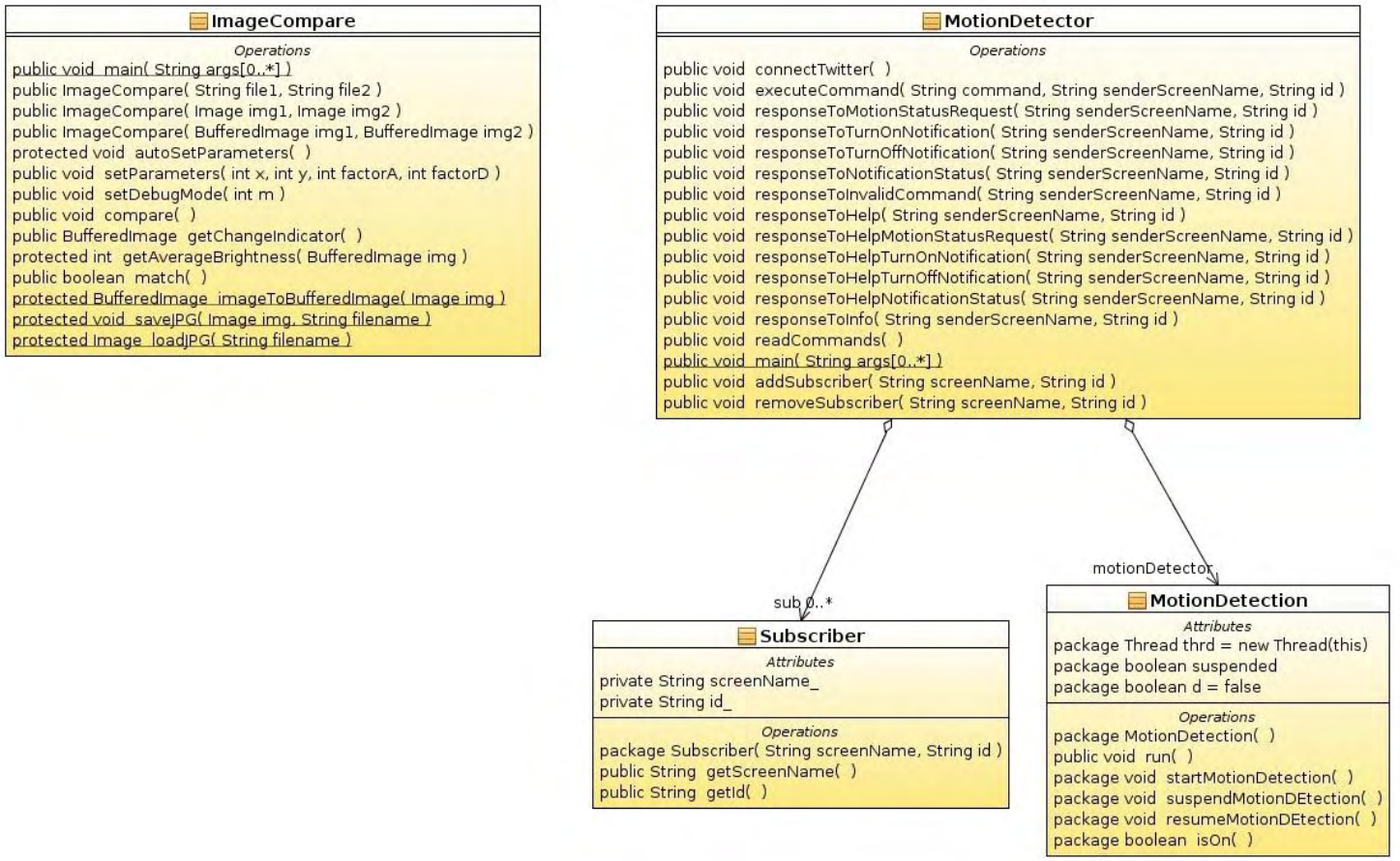
Στην Εικόνα 13 φαίνεται το UML διάγραμμα των κλάσεων που υλοποιούν τον πόρο της κάμερας.



Εικόνα 13: UML Διάγραμμα Κλάσεων που υλοποιούν την Κάμερα

ΑΙΣΘΗΤΗΡΑΣ ΚΙΝΗΣΗΣ

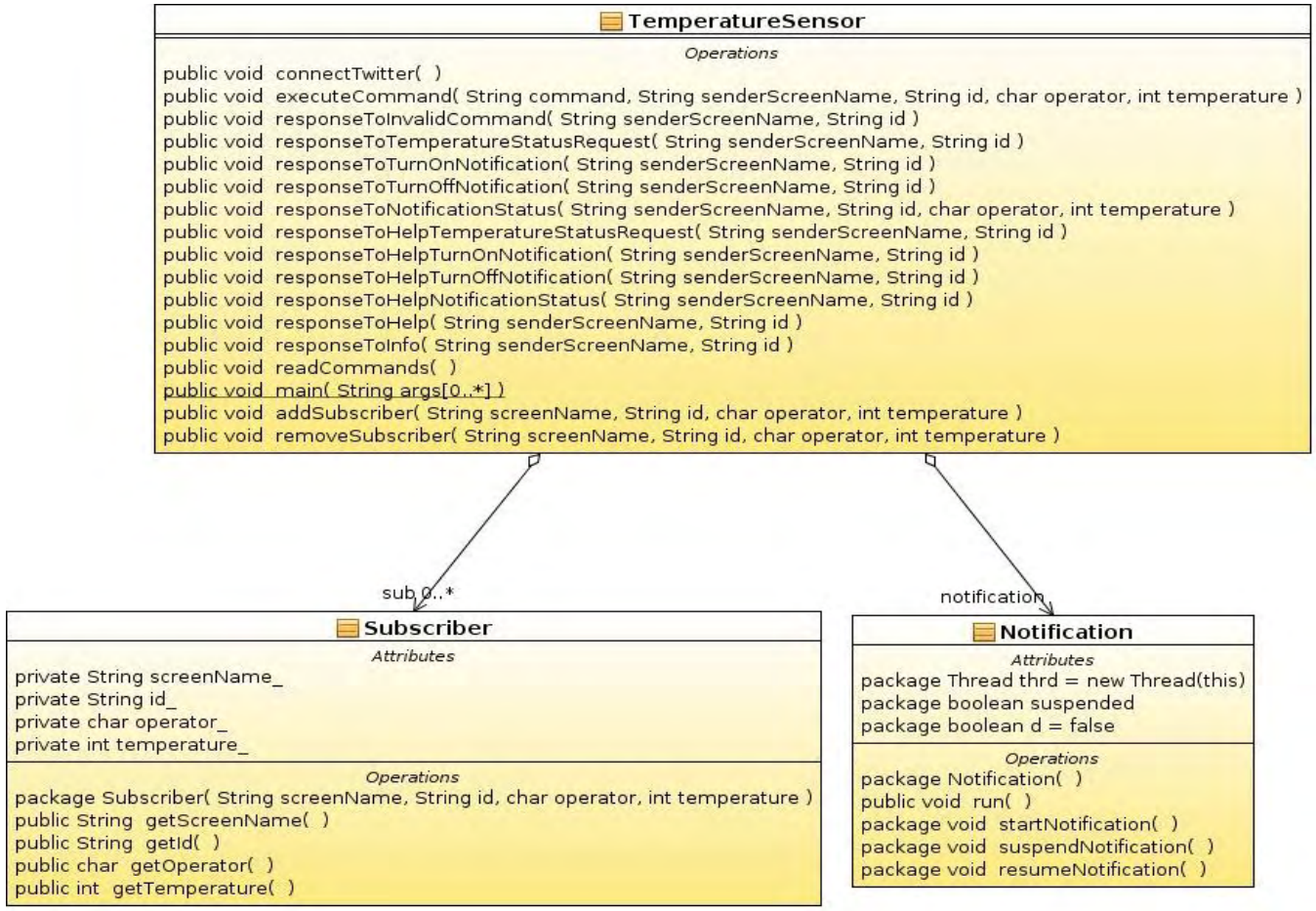
Στην Εικόνα 14 φαίνεται το UML διάγραμμα των κλάσεων που υλοποιούν τον πόρο του αισθητήρα κίνησης.



Εικόνα 14: UML Διάγραμμα Κλάσεων που υλοποιούν τον Αισθητήρα Κίνησης

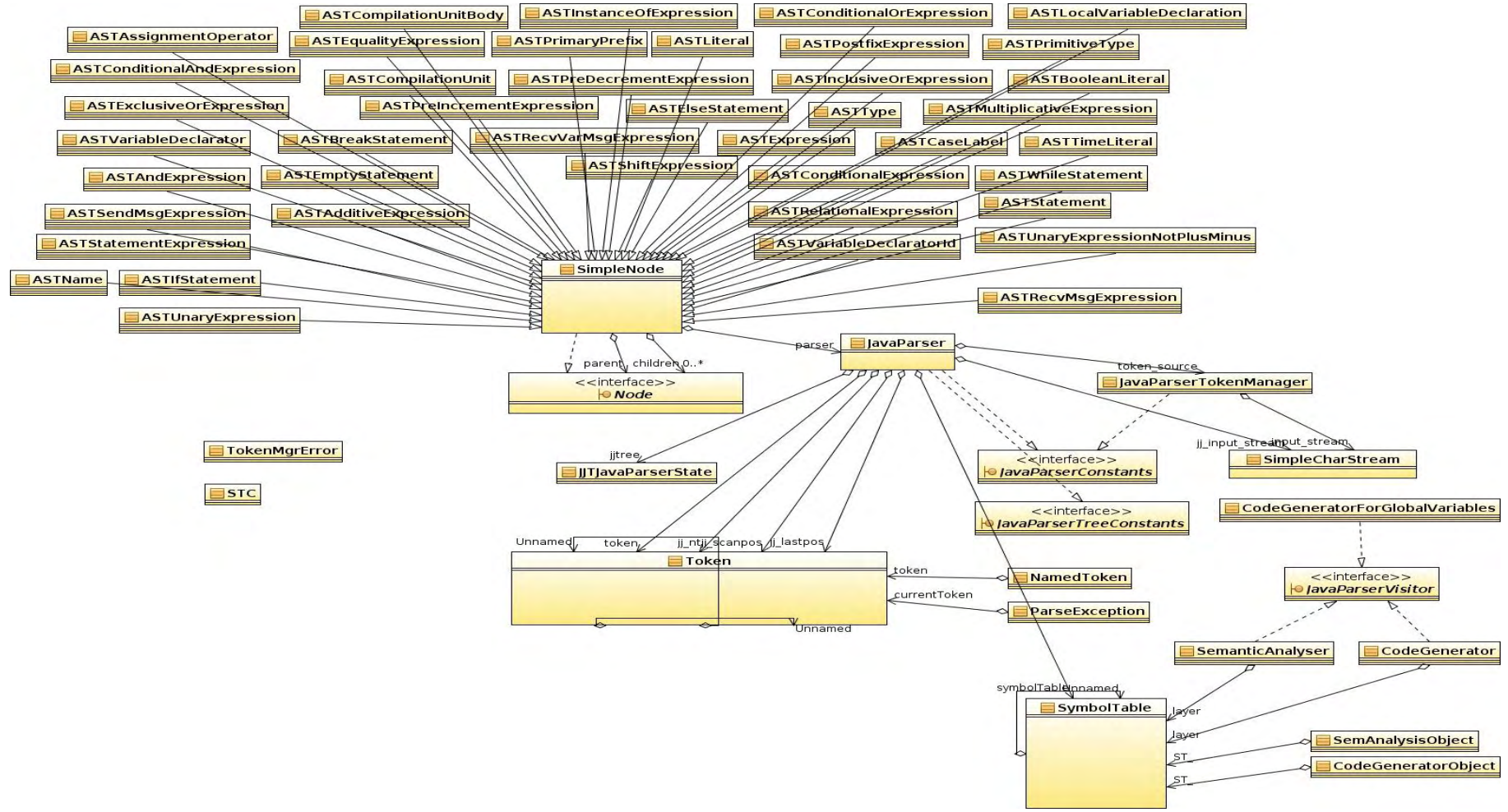
ΑΙΣΘΗΤΗΡΑΣ ΘΕΡΜΟΚΡΑΣΙΑΣ

Στην Εικόνα 15 φαίνεται το UML διάγραμμα των κλάσεων που υλοποιούν τον πόρο του αισθητήρα θερμοκρασίας.



Εικόνα 15: UML Διάγραμμα των Κλάσεων που υλοποιούν τον Αισθητήρα Θερμοκρασίας

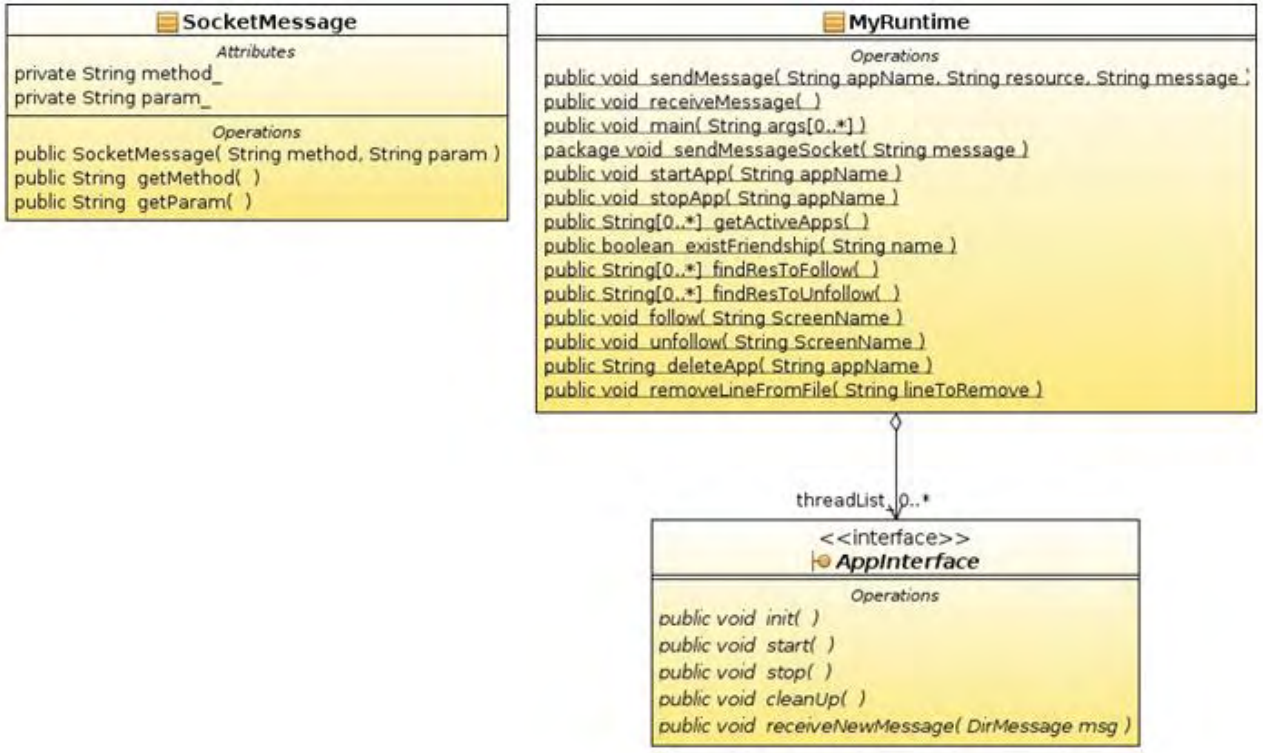
ΜΕΤΑΓΛΩΤΤΙΣΤΗΣ



Εικόνα 16: UML Διάγραμμα Κλάσεων που υλοποιούν τον Μεταγλωττιστή

RUNTIME

Στην Εικόνα 17 παρουσιάζεται το UML διάγραμμα των κλάσεων που υλοποιούν το Runtime.



Εικόνα 17: UML Διάγραμμα Κλάσεων που υλοποιούν το Runtime

11 Αναφορές

[1] Twingz- http://www.twingz.com/index.html

[2] QBUS - http://www.qbus.be/en/about_us/why_qbus/flexibility

[3] Dimitris N. Kalofonos, Franklin D. ReynoldsTask. "Task-Driven End-User Programming of Smart Spaces Using Mobile Devices". Nokia Research Center Cambridge, January 2006. http://research.nokia.com/files/tr/NRC-TR-2006-001.pdf

[4] Jan Humble, Andy Crabtree, Terry Hemmings , Karl-Petter Åkesson*,Boriana Koleva, Tom Rodden, Pär Hansson. "Playing with the Bits User-configuration of Ubiquitous Domestic Environments " . http://pages.cpsc.ucalgary.ca/~saul/wiki/uploads/CPSC7018108/humble_crabtree_rodden__playing-with-the-bits_user-configuration-of-ubiquitous-domestic-environments__ubicomp-2003.pdf

[5] Rafael Ballagas, Andy Szybalski, Armando Fox. "Patch Panel: Enabling Control-Flow Interoperability in Ubicomp Environments". http://andy.bigwhitebox.org/papers/patchpanel.pdf

[6] Joao Pedro Sousa, Vasilios Tzeremes, Ala'a EI Masri. George Mason University, Computer Science Department. "Space-Aware *TeC*: End-User Development of Safety and Control Systems for Smart Spaces". http://cs.gmu.edu/~jpsousa/research/papers/TeC-IEEE%20SMC.pdf

[7] Twitter API -https://dev.twitter.com/

[8] Flickr API - http://www.flickr.com/services/api/

[9] Google Docs API - https://developers.google.com/google-apps/documents-list/

[10]JavaCV - http://code.google.com/p/javacv/

[11] JavaCC - http://javacc.java.net/

[12] NetBeans - http://netbeans.org/

[13] Twitter – https://twitter.com/

[14] Flickr – http://www.flickr.com/

[15] Google Docs - http://www.google.com/google-d-s/documents/