ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

"Ανάπτυξη συστήματος για την παρακολούθηση και έλεγχο ασύρματων δικτύων αισθητήρων μέσω του παγκόσμιου ιστού."

Διπλωματική Εργασία

Δημητρός Κωνσταντίνος

Επιβλέπων: Σπυράκης Παύλος

Πολυτεχνική Σχολή
Τμήμα Μηχανικών Η/Υ και Πληροφορικής

Σεπτέμβριος 2008

Πρόλογος

Θα ήταν κοινός τόπος να επισημάνουμε την εξέλιξη του διαδικτύου τα τελευταία χρόνια, το πλήθος της πληροφορίας και το πόσο εύκολα μπορεί να γίνει διαθέσιμη αυτή μέσω του παγκόσμιου ιστού. Εκατοντάδες υπηρεσίες έχουν σχεδιαστεί κατάλληλα έτσι ώστε μέσω του διαδικτύου η πρόσβαση σε πληροφορία και δεδομένα να είναι η πλέον γρήγορη και έυκολη διαδικασία. Η χρήση των υπολογιστών και της τεχνολογίας δικτύων είναι μονόδρομος για τη υποστήριξη του διαδικτύου.

Μια, σχετικά νέα, κατηγορία στην ευρύτερη επιστήμη των υπολογιστών και δικτύων είναι τα Ασύρματα Δίκτυα Αισθητήρων (Wireless Sensor Networks). Αυτά τα δίκτυα αποτελούνται από δεκάδες, εκατοντάδες(ακόμα και χιλιάδες στα προσεχή χρόνια) κόμβους οι οποίοι διαθέτουν πλήθος μονάδων ελέγχου και μικροσκοπικών αισθητήρων. Σκοπός αυτών των δικτύων είναι η εκμετάλλευση της φύσης τους για τη συλλογή πληροφορίας, την μεταξύ τους συνεργασία και τελικώς την αξιοποίηση αυτού του πακέτου για την εξυπηρέτηση διαφόρων ειδών αναγκών. Όπως αναφέρθηκε, τα WSN αποτελούν μια νέα σχετικά τεχνολογία και παρόλα αυτά, λόγω των πολλών εφαρμογών τέτοιου είδους δικτύων, πλήθος επιστημόνων και ερευνητών ασχολούνται με αυτά και την εξέλιξή τους.

Σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μέρους ενός πλήρως κατανεμημένου συστήματος το οποίο θα μπορεί να δώσει στους χρήστες του τη δυνατότητα να διαχειρίζονται, να ελέγχουν, να παρακολουθούν, να σχεδιάζουν εύκολα εφαρμογές για ασύρματα δίκτυα αισθητήρων μέσω του παγκόσμιου ιστού. Το σύστημα αυτό το καλούμε Webdust2. Το σύστημα αυτό βρίσκεται υπό συνεχή εξέλιξη και άτομα από διάφορους τομείς(υλικού, λογισμικού, δικτύων, κτλ.) εργάζονται έτσι ώστε το σύστημα αυτό να είναι όσο πιο πλήρες και πιο εύκολα προσβάσιμο και ενημερώσιμο γίνεται. Ειδικότερα, στην παρούσα εργασία, εφόσον περιγράψουμε συνοπτικά την αρχιτεκτονική του θα αναλύσουμε τα μέρη εκείνα τα οποία έχουν σχέση με την αποθήκευση των δεδομένων, την διαχείρισή τους και τελικά την παρουσίασή τους προς τον τελικό χρήστη μέσω του διαδικτίου. Η ανάπτυξη της εφαρμογής γίνεται με χρήση της τεχνολογίας Hibernate και είναι γραμμένη σε γλώσσα Java.

Ευχαριστίες

Θα ήθελα πρώτα από όλα να ευχαριστήσω τους γονείς μου οι οποίοι έχουν σταθεί στο πλευρό μου και με έχουν στηρίξει με κάθε δυνατό τρόπο όλα αυτά τα χρόνια. Η υπομονή τους και η πίστη τους σε μένα υπήρξε και υπάρχει σημαντική όχι μόνο για το διάστημα που εκπονούσα την παρούσα διπλωματική, αλλά και καθόλη τη διάρκεια των σπουδών μου και όλης μου της ζωής.

Έπειτα, θα ήθελα να ευχαριστήσω θερμά τον Δρ. Ιωάννη Χατζηγιαννάκη ο οποίος εμπιστεύθηκε στο πρόσωπο μου μέρος της ανάπτυξης της εφαρμογής αλλά και για τον απαράμιλλο ζήλο και διάθεση που επέδειξε κατά την όλη διαδικασία, την πολύτιμη βοήθεια και καθοδήγηση του σε κάθε σημείο της εργασίας με κόστος του δικού του πολυτίμου του χρόνου. Του εύχομαι τα καλύτερα για τη συνέχεια στη προσωπική και επαγγελματική του διαδρομή.

Θερμές ευχαριστίες προς τον καθηγητή του τμήματος μας κ. Παύλο Σπυράκη για την επίβλεψη της παρούσας εργασίας αλλά και για την εμπιστοσύνη που μου έδειξε για την εκπόνησή της. Ο υποδειγματικός τρόπος διδασκαλίας του, ο ζήλος και η αγάπη προς του φοιτητές του καθώς και οι αμέτρητες γνώσεις που μας παρείχε κατά τα χρονιά των σπουδών μας , μας εμφύσησαν την αγάπη το πάθος που χρειάζεται κάθε νέος άνθρωπος στο ξεκίνημα του καθώς και τη τεχνογνωσία και τη νοοτροπία που απαιτείται από κάθε μηχανικό.

Τέλος, θα ήθελα να ευχαριστήσω όλους τους συναδέλφους και φίλους μου που έμειναν στο πλευρό μου καθόλη τη διάρκεια της εκπόνησης αυτής της διπλωματικής.

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Στους Γονείς μου…

Κεφάλαιο 1

Εισαγωγή

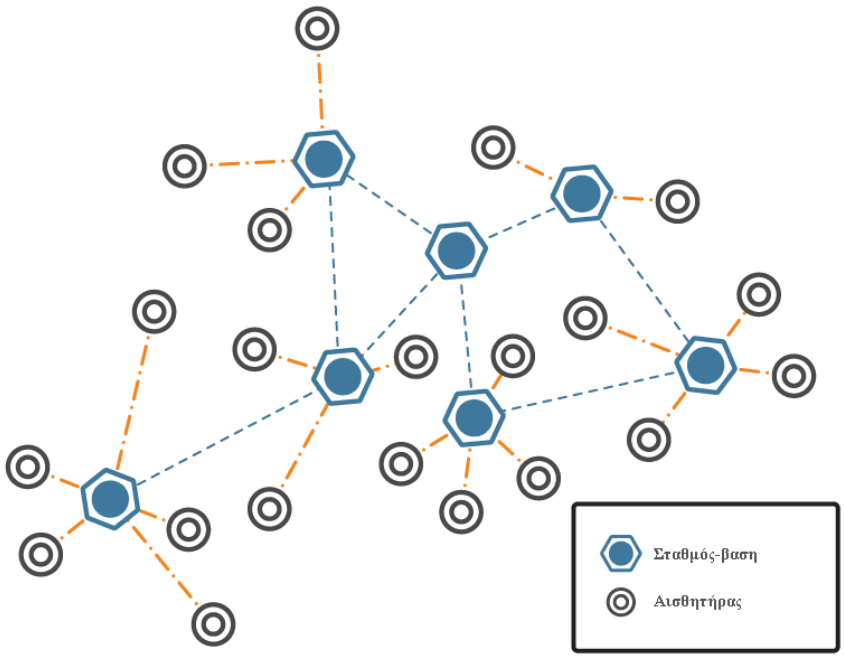
1.1 Ασύρματα Δίκτυα Αισθητήρων (Wireless Sensor Networks)

Τα τελευταία χρόνια η εξέλιξη που αφορά στην ηλεκτρονική τεχνολογία είναι ραγδαία. Πιο συγκεκριμένα, σήμερα, ο βαθμός ολοκλήρωσης των ψηφιακών συστημάτων είναι τέτοιος ώστε το μέγεθος των σύγχρονων μικροελεγκτών ή επεξεργαστών να είναι εντυπωσιακά μικρό. Παράλληλα, μια σχετικά νέα τάση που παρατηρείται είναι αυτή της χρήσης πλήθους υπολογιστικών συστημάτων για την επίτευξη ενός κοινού στόχου. Τα Ασύρματα Δίκτυα Αισθητήρων(WSN) είναι αποτέλεσμα της συνισταμένης των δυο τελευταίων τάσεων και αποτελούν μια σχετικά νέα εξέλιξη στην επιστήμη των υπολογιστών, των τηλεπικοινωνιών και της τεχνολογίας. Το έδαδος είναι εξαιρετικά πρόσφορο για αυτη την εξέλιξη καθώς επίσης και πλήθος επιστημόνων ασχολούνται με τα δίκτυα αυτά λόγω του μεγάλου εύρους εφαρμογών που έχουν σε διάφορους τομείς(Βλέπε Παράγραφος 1.3).

Με τον όρο Ασύρματα Δίκτυα Αισθητήρων (WSN) εννοούμε συλλογές από κόμβους μικρο-αισθητήρων οι οποίοι είναι οργανωμένοι μέσα σε ένα ασύρματο δίκτυο. Κάθε κόμβος έχει επεξεργαστική δυνατότητα (π.χ. Μικροελεγκτή, CPU, κτλ.), μνήμη (μνήμη προγράμματος, μνήμη δεδομένων, κτλ.), ένα πομποδέκτη ραδιοσυχνοτήτων η/και οποιοδήποτε άλλο μέσο ασύρματης επικοινωνίας, μια πηγή ενέργειας και διάφορους αισθητήρες (θερμόμετρο, κάμερα, μικρόφωνο, μετρητή υγρασίας, κ.α.) για καταγραφή, αποθήκευση και αποστολή δεδομένων.

Ένα ασύρματο δίκτυο αισθητήρων (Βλέπε Σχήμα 1.1) αποτελείται από εκατοντάδες ή

χιλιάδες χαμηλού κόστους κόμβους(*motes*) οι οποίοι είτε έχουν μια συγκεκριμένη τοποθεσία ή είναι τυχαία διασκορπισμένοι έτσι ώστε να παρακολουθούν το γύρω τους περιβάλλον. Αισθητήρες συνήθως επικοινωνούν μεταξύ τους χρησιμοποιώντας μια προσέγγιση που καλείται *multi-hop*. Σύμφωνα με αυτή την προσέγγιση η ροή των δεδομένων περνά από κόμβο σε κόμβο και καταλήγει σε ειδικούς κόμβους που καλούνται σταθμοί ή κόμβοι βάσης ή καταβόθρες(*sink*). Ο σταθμός βάσης συνδέει το δίκτυο αισθητήρων σε άλλο δίκτυο (όπως μια πύλη) για τη διάδοση των δεδομένων που ανιχνεύονται για περαιτέρω επεξεργασία. Οι σταθμοί βάσης έχουν πιο ενισχυμένες δυνατότητες από ένα απλό κόμβο αισθητήρων δεδομένου ότι πρέπει να κάνουν πολύπλοκες επεξεργασίες δεδομένων. Αυτό δικαιολογεί το γεγονός ότι οι σταθμοί βάσης έχουν επεξεργαστές κλάσης τερματικού ή φορητού Η/Υ, και φυσικά αρκετή μνήμη, ενέργεια, αποθηκευτικό χώρο και υπολογιστική ισχύ για να εκτελέσουν το έργο τους. Συνήθως, η επικοινωνία μεταξύ των σταθμών βάσης πραγματοποιείται με συνδέσεις υψηλού εύρους ζώνης.

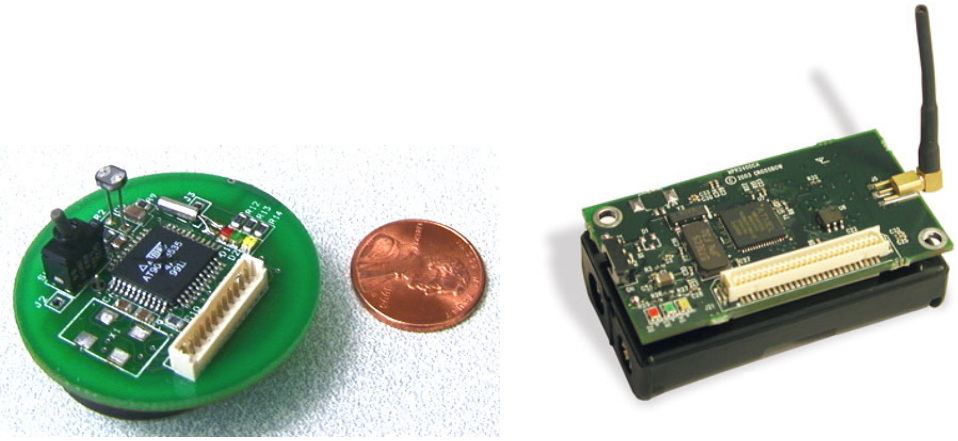


Σχήμα 1.1: Παράδειγμα ενός Wireless Sensor Network.

Τα δίκτυα αυτά ονομάζονται επίσης δίκτυα *έξυπνης σκόνης* (Smart Dust). Ο προσδιορισμός αυτός πρωτοκαθιερώθηκε το 2001 από τον *Kristofer S. J. Pister* (University of California, Berkeley) ο οποίος συμμετείχε στην ανάπτυξη εφαρμογής που θα έκανε χρήση της τεχνολογίας WSN μεγάλης κλίμακας κάτω από τη χρηματοδότηση του DAPRA[1]. Στη βιβλιογραφία συναντώνται και οι δυο όροι για τον χαρακτηρισμό των δικτύων αυτών. Στην παρούσα εργασία θα αναφερόμαστε σε αυτά με τον όρο WSN.

[1]Defense Advanced Research Projects Agency's Microsystems Technology Office

Ένώ αρχικά θα μπορούσε να πει κανείς πως η αρχική ιδέα για την ανάπτυξη WSN είναι απλή, η υλοποίηση των δικτύων αυτών δεν είναι και η πλέον εύκολη υπόθεση. Υπάρχει πλήθος παραγόντων(Βλέπε Παράγραφος 1.2) που επιρεάζουν συνολικά το σχεδιασμό ενός WSN. Παρόλα αυτά, η ενασχόληση πλήθους ερευνητών από δίαφορα επιστημονικά πεδία αλλά και οι χρηματοδοτήσεις, κάνουν τα WSN πιο αποδοτικά και πιο ελκυστικά ακόμη και στην ανοιχτή αγορά.



(α) a mote (β) mica-Z mote

Σχήμα 1.2: Εικόνες ασύρματων αισθητήρων

1.2 Παράγοντες για το σχεδιασμό WSN

Τοπολογία δικτύου: Η τοπολογία ενός ασύρματου δικτύου αισθητήρων μπορεί να είναι εντελώς τυχαία ή να ακολουθεί κάποιο πρότυπο. Μπορεί να γίνει ρίψη συσκευών από αεροπλάνο (η συγκεκριμένη μέθοδος έχει ήδη δοκιμαστεί), από καταπέλτη, με τα χέρια, κτλ. Μετά την τοποθέτηση των κόμβων η τοπολογία μπορεί να αλλάξει λόγω θέσης, συνδεσιμότητας, έλλειψης ενέργειας. Επιπλέον, νέοι κόμβοι μπορεί να τοποθετηθούν για να αντικαταστήσουν παλιούς που τέθηκαν εκτός λειτουργίας ή για να επεκτείνουν το σύστημα. Επομένως, το σύστημα πρέπει να αντιδρά δυναμικά απέναντι σε όλους αυτούς τους παράγοντες.

Προσαρμοστικότητα του δικτύου: Ένα ασύρματο δίκτυο αισθητήρων έχει να αντιμετωπίσει δυσκολίες όπως συχνές αστοχίες υλικού, συχνά προβλήματα στην επικοινωνία μεταξύ κόμβων και μεγάλο μέγεθος και πυκνότητα δικτύου. Οι αλγόριθμοι, οι οποίοι ρυθμίζουν την επικοινωνία μεταξύ των κόμβων πρέπει να έχουν μεγάλες ανοχές σε λάθη, να αντιλαμβάνονται γρήγορα πότε μια σύνδεση δε λειτουργεί σωστά και να δημιουργούν καινούριες. Επιπλέον, πρέπει να αντιμετωπίσουν τη μεγάλη πυκνότητα του

δικτύου και να μπορούν να εφαρμόζονται σε περιπτώσεις στις οποίες έχουμε αρκετές χιλιάδες κόμβων.

Μέσο μετάδοσης: Η επικοινωνία μεταξύ των κόμβων του δικτύου γίνεται μέσω ραδιοσυχνοτήτων, αν και έχουν παρουσιαστεί σενάρια για επικοινωνία μέσω laser. Για τις ραδιοσυχνότητες πρέπει να επιλεγεί ζώνη μετάδοσης, η οποία θα είναι διαθέσιμη για εμπορική χρήση, π.χ οι συχνότητες που χρησιμοποιούνται από τα wireless LAN. Επιπλέον, πρέπει να υπάρχουν μεγάλα περιθώρια ανοχής σε θόρυβο και παρεμβολές, δεδομένων των συνθηκών που επικρατούν στους χώρους που ενδέχεται να χρησιμοποιηθούν τα ασύρματα δίκτυα αισθητήρων. Υπάρχει η επιλογή μεταξύ narrowband και wideband ραδιοσυχνοτήτων. Στην πρώτη περίπτωση έχουμε το πλεονέκτημα της χαμηλής κατανάλωσης ενέργειας και της απλής υλοποίησης σε hardware, ενώ στη δεύτερη περίπτωση έχουμε το πλεονέκτημα της μεγαλύτερης ανοχής σε θόρυβο και του μεγαλύτερου data rate.

Περιορισμοί υλικού και κόστους παραγωγής: Ένας κόμβος ασύρματου δικτύου αισθητήρων αποτελείται από τέσσερα βασικά τμήματα, έναν αισθητήρα(ή και περισσότερους), μία CPU, ένα πομποδέκτη και μια μονάδα ισχύος. Επίσης, είναι δυνατόν να υπάρχουν κι άλλα τμήματα, όπως μονάδα GPS ή κινητήρας. Όλες αυτές οι μονάδες πρέπει να χωρούν σε μια συσκευασία μεγέθους σπιρτόκουτου ή, ιδανικά, ακόμα μικρότερη. Ακόμα, πρέπει να καταναλώνουν ελάχιστη ενέργεια, να είναι αυτόνομες και να λειτουργούν ανεξάρτητες, να προσαρμόζονται στο περιβάλλον τους, να αντέχουν σε αντίξοες συνθήκες. Επομένως, υπάρχουν πολύ μεγάλες απαιτήσεις στον τομέα του υλικού, και πρέπει η έρευνα στον τομέα της κατασκευής ολοκληρωμένων κυκλωμάτων να υποστηρίξει αυτές τις ανάγκες. Στα παραπάνω έρχεται να προστεθεί η απαίτηση για πολύ μικρό κόστος παραγωγής ενός κόμβου. Ένα ασύρματο δίκτυο αισθητήρων προφανώς είναι συμφέρον, αν κάθε κόμβος στοιχίζει τόσο λίγο ώστε το συνολικό κόστος να μην είναι απαγορευτικό.

Κατανάλωση ενέργειας: Η λειτουργία ενός κόμβου-αισθητήρα συνοψίζεται ως εξής: ανίχνευση γεγονότος, επεξεργασία πληροφορίας, μετάδοση ή λήψη μηνύματος. Επομένως, η ενέργεια ενός κόμβου ξοδεύεται με τρεις τρόπους.

- Επικοινωνία: Η περισσότερη ενέργεια καταναλώνεται εδώ, στην οποία περιλαμβάνεται τόσο η μετάδοση όσο και η λήψη δεδομένων. Γενικά, για μικρές αποστάσεις και για τις δύο περιπτώσεις καταναλώνεται περίπου η ίδια ενέργεια. Επίσης, λαμβάνουμε υπόψη μας και την ενέργεια για την αρχικοποίηση του πομποδέκτη. Επομένως, πρέπει να βρούμε έναν έξυπνο τρόπο να διαχειριστούμε τον πομποδέκτη και να ελαχιστοποιήσουμε μεταδόσεις και λήψεις μηνυμάτων. Τυπικές τιμές για αυτήν την περιοχή είναι 50 nJ/bit.

- Επεξεργασία πληροφορίας: Εδώ, η κατανάλωση ενέργειας είναι πολύ μικρότερη σε σχέση με την ενέργεια για επικοινωνία. Μπορούμε να εκμεταλλευτούμε την ισχύ του επεξεργαστή και να κάνουμε συμπίεση δεδομένων ή μπορούμε ακόμα και να επεξεργαστούμε τα μηνύματα που λαμβάνουμε και να αφαιρέσουμε περιττή πληροφορία (π.χ το ίδιο μήνυμα έχει φθάσει από δύο κόμβους), προκειμένου να γλιτώσουμε bit μεταδιδόμενης πληροφορίας. Τυπική τιμή για τους σύγχρονους επεξεργαστές είναι 1 pJ/εντολή.

- Μέτρηση πεδίου: Η ενέργεια που καταναλώνεται εδώ είναι σταθερή και περιλαμβάνει την ενέργεια για λήψη δείγματος από τους αισθητήρες ενός κόμβου. Προφανώς, μπορούμε να ελαττώσουμε την ενέργεια που καταναλώνουμε, όσο πιο αραιά κάνουμε μετρήσεις της ποσότητας που μας ενδιαφέρει (π.χ θερμοκρασία). Τυπικές τιμές σε αυτό το πεδίο είναι 4 nJ ανά δείγμα (ακρίβεια 10 bit/δειγμα).

Πηγές ενέργειας: Υπάρχουν αρκετές τεχνολογικές προτάσεις ως υποψήφιες πηγές ενέργειας για χρήση σε ασύρματα δίκτυα αισθητήρων, καθεμιά με τα πλεονεκτήματα και τα μειονεκτήματα της. Προφανώς αυτό που ενδιαφέρει περισσότερο είναι η διάρκεια ζωής και η αυτονομία που προσφέρουν στους κόμβους του δικτύου, αλλά υπάρχουν και άλλες σημαντικές παράμετροι όπως το μέγεθος, η ευκολία στη σύνδεση με τους κόμβους, κ.α. Προς το παρόν, οι υπάρχουσες λύσεις προσφέρουν περιορισμένη αυτονομία. Η τεχνολογία η οποία χρησιμοποιείται ως επί το πλείστον είναι οι μπαταρίες λιθίου. Οι τελευταίες γενιές των μπαταριών αυτών είναι επαναφορτιζόμενες, γεγονός το οποίο είναι πολύ χρήσιμο. Μια άλλη πρόταση από το χώρο των μπαταριών, είναι οι νέες μπαταρίες λεπτού φιλμ οξειδίου βαναδίου και οξειδίου μολυβδαινίου. Οι χωρητικότητες τους είναι ανάλογες των μπαταριών λιθίου και επιπλέον έχουν το πλεονέκτημα ότι μπορούν να ενσωματωθούν στους κόμβους πιο εύκολα. Μια άλλη πρόταση είναι η χρήση φωτοηλεκτρικών κυττάρων, τα οποία παράγουν ενέργεια από το φως. Το φως αυτό μπορεί να προέρχεται είτε από τον Ήλιο, είτε από κάποια άλλη φωτεινή πηγή, όπως ο εσωτερικός φωτισμός. Βέβαια, στη δεύτερη περίπτωση η παραγόμενη ενέργεια είναι κλάσμα της πρώτης. Η λύση αυτή μπορεί να χρησιμοποιηθεί περισσότερο ως ένας τρόπος να αναπληρώνεται η ενέργεια μιας μπαταρίας, η οποία θα είναι η κύρια πηγή ενέργειας. Αν αυτό μπορεί να γίνεται με έναν ικανοποιητικό ρυθμό, τότε μπορεί να αυξηθεί αρκετά ο χρόνος ζωής ενός ασύρματου δικτύου αισθητήρων.

1.3 Εφαρμογές WSN

Το φάσμα των εφαρμογών των WSN είναι αρκετά ευρύ και είναι ακόμα ανοικτό. Ολοένα γίνονται και νέες προτάσεις για εφαρμογές αφού τα WSN μας δίνουν νέες δυνατότητες και αντικαθιστούν ακόμη και παλαιότερες μεθόδους. Η παράμετρος που επηρεάζει περισσότερο από όλες το είδος των εφαρμογών που μπορούν να αναπτυχθούν με τη χρήση των WSN είναι τα είδη των αισθητήρων που μπορούν να ενσωματωθούν στους κόμβους. Οι κλάδοι που μπορούν να επωφεληθούν από την ιδέα της λήψης και διακίνησης δεδομένων σε πραγματικό χρόνο είναι αμέτρητες. Το θέμα είναι τι είδους δεδομένα μπορεί να είναι αυτά. Οι δημοφιλέστερες φυσικές παράμετροι για τις οποίες υπάρχουν αισθητήρες που μπορούν να ενσωματωθούν στα ολοκληρωμένα των κόμβων είναι η θερμοκρασία, η ορατή ακτινοβολία (εικόνα), η υπέρυθρη ακτινοβολία, ο ήχος, η ταχύτητα, η επιτάχυνση, η υγρασία, η δόνηση, η πίεση, η ποσότητα χημικών ουσιών, ο μαγνητισμός κ.α. Με βάση αυτά, υπάρχει πλήθος τομέων στους οποίους τα δεδομένα αυτά ή συνδιασμός τους θα μπορούσαν να φανούν χρήσιμα και να έχουν πρακτική εφαρμογή. Έτσι, ενδεικτικά, έχουμε εφαρμογές WSN σε τομείς όπως:

Περιβαλλοντική Παρατήρηση

Δίκτυα αισθητήρων μπορούν να χρησιμοποιηθούν για την παρακολούθηση των περιβαλλοντικών αλλαγών. Ένα παράδειγμα θα μπορούσε να είναι η ανίχνευση της ρύπανσης των υδάτων σε μια λίμνη που βρίσκεται κοντά σε ένα εργοστάσιο που χρησιμοποιεί χημικές ουσίες. Οι κόμβοι αισθητήρων θα μπορούσαν να είναι τυχαία τοποθετημένοι σε άγνωστες και εχθρικές περιοχές και να ανέφεραν την ακριβή προέλευση του ρύπου σε μια κεντρική αρχή ώστε να λάβει τα κατάλληλα μέτρα για να περιοριστεί η διασπορά της ρύπανσης. Άλλα παραδείγματα περιλαμβάνουν πυρανίχνευση δασών, ατμοσφαιρική ρύπανση και παρατήρηση των βροχοπτώσεων στη γεωργία.

Στρατιωτική Παρακολούθηση

Ο στρατός χρησιμοποιεί δίκτυα αισθητήρων για επιτήρηση του πεδίου μάχης. Αισθητήρες θα μπορούσαν να παρακολουθούν την κυκλοφορία οχημάτων, τη θέση του εχθρού ή ακόμη και τη διαφύλαξη των εξοπλισμών της πλευράς που χρησιμοποιεί αισθητήρες.

Παρακολούθηση Κτιρίων

Αισθητήρες μπορεί επίσης να χρησιμοποιηθούν σε μεγάλα κτίρια ή εργοστάσια για παρακολούθηση των κλιματικών αλλαγών. Θερμοστάτες και κόμβοι με αισθητήρες θερμοκρασίας τοποθετούνται παντού σε όλες τις περιοχές του κτίριου. Επιπλέον, οι αισθητήρες θα μπορούσαν να χρησιμοποιηθούν για την παρακολούθηση κραδασμών που θα μπορούσαν να προκαλέσουν βλάβη στη δομή του κτιρίου.

Υγεία και Ιατρική

Αισθητήρες μπορούν να χρησιμοποιηθούν σε βιοϊατρικές εφαρμογές για να βελτιώσουν την ποιότητα της περίθαλψης που παρέχεται. Αισθητήρες εμφυτεύονται στο ανθρώπινο σώμα για να παρακολουθούν ιατρικά προβλήματα, όπως του καρκίνου, και να βοηθήσει τους ασθενείς να παραμένουν υγειείς. Επίσης, ένας άλλος τρόπος χρήσης των WSN στην ιατρική είναι αυτός της διευκόλυνσης των σωστών επιλογών σε καταστάσεις ανάγκης. Δηλαδή, τα ασθενοφόρα αποφασίζουν με βάση τη διαθεσιμότητα δωματίων και γιατρών σε ποιο νοσοκομείο να προσκομίσουν τους ασθενείς, οι γιατροί καλούνται αυτόματα με τις αφίξεις ασθενών ή επιδείνωσης της κατάστασής τους κτλ.

Έλεγχος Κίνησης και Κυκλοφοριακού Προβλήματος

Ο τομέας αυτός που έχουν προσελκύει έντονο ενδιαφέρον από ερευνητικές εφαρμογές. Αισθητήρες τοποθετημένοι σε σταθερά σημεία του οδικού δικτύου ή/και σε αυτοκίνητα μπορούν να συλλέγουν πληροφορίες σχετικά με την κυκλοφορία και να βοηθούν τους ενδιαφερόμενους χρήστες (οδηγούς) στις αποφάσεις τους. Αυτή η προοπτική θα μπορούσε να βοηθήσει στην βελτίωση του κυκλοφοριακού προβλήματος.

Τα WSN μπορούν σίγουρα να προσφέρουν λύσεις και νέα δυναμική στους παραπάνω τομείς. Οι υπάρχουσες εφαρμογές τους αλλά και οι εφαρμογές που είναι ακόμη υπό μελέτη είναι σίγουρο οτι έχουν να προσφέρουν πολλά στο μέλλον της τεχνολογίας και του περιβάλλοντος.

1.4 Απομακρυσμένος Έλεγχος και WEB Εφαρμογές

Ένα βασικό κομμάτι που έχει σχέση με τις εφαρμογές των WSN είναι ο Απομακρυσμένος Έλεγχος. Σε πάρα πολλές περιπτώσεις, ο έλεγχος από απόσταση είναι ίσως και ο βασικότερος λόγος ανάπτυξης των εφαρμογών αυτών. Από το πιο απλό σενάριο όπου κόμβοι αισθητήρων είναι απλά τοποθετημένοι σε μια περιοχή με σκοπό την άντληση πληροφοριών έως και σενάρια στα οποία οι κόμβοι είναι μεταβλητής θέσης ή η περιοχή του δικτύου είναι δυσπρόσβατη, υπάρχει έντονη ανάγκη ελέγχου των αισθητήρων από απόσταση. Σε κάθε εφαρμογή WSN, ο χρήστης μπορεί να παρακολουθεί τη ροή των δεδομένων, τις μετρήσεις των αισθητήρων και επίσης μπορεί να επεμβαίνει στο σύστημα του WSN για να επιλέγει το ποιές μετρήσεις τον ενδιαφέρουν κάθε φορά ή να τροποποιήσει τις ρυθμίσεις κάποιου αισθητήρα ή ακόμα και με εντολές να αλλάξει την τοπολογία του δικτίου(σε περίπτωση που υπάρχει κατάλληλος εξοπλισμός). Είναι φανερό, πως για όλες αυτές τις περιπτώσεις χρειάζεται ένα από απόσταση σύστημα το οποίο να επιτρέπει εύκολα στο χρήστη να μπορεί να διαχειριστεί πλήρως το WSN. Αν, τώρα, θέλουμε να γενικεύσουμε ακόμη περισσότερο, έτσι ώστε ο χρήστης να έχει τη δυνατότητα από

οπουδήποτε να ελέγξει και να διαχειριστεί ένα τέτοιο σύστημα, φτάνουμε στην ιδέα του WEB.

Το WEB(Παγκόσμιος Ιστός/Διαδίκτυο) είναι το μεγαλύτερο μέσο, παγκόσμια, για τη διάδοση και διάθεση πληροφορίας. Οποιαδήποτε πληροφορία μπορεί να φιλοξενηθεί στον παγκόσμιο ιστό και οποιοσδήποτε, από οπουδήποτε, να μπορεί να έχει πρόσβαση σε αυτό. Είναι κοινός τόπος να αναφέρουμε την εξέλιξη, το εύρος και τη χρήση του WEB σήμερα. Οργανισμοί με ποικίλες αρμοδιότητες και επιχειρηματικές λειτουργίες αντιμετωπίζουν την ανάγκη να παρέχουν υπηρεσίες βασισμένες στο διαδίκτυο για τους χρήστες της ίδιας της επιχείρησης, άλλα κέντρα δεδομένων, καθώς και τον έξω κόσμο. Για την παροχή αυτών των υπηρεσιών, χρειάζεται να αναπτύξουν Web αρχιτεκτονικές βασιζόμενες στο μοντέλο του εξυπηρετητή-πελάτη(*server-client*) που θα είναι διαθέσιμες σε μεγάλο βαθμό, ασφαλής, και εύκολα εξελίξιμες. Έτσι ερχόμαστε στην έννοια των Web-υπηρεσιών. Οι εφαρμογές που σχετίζονται με Web-υπηρεσίες, είναι οργανωμένες με τέτοιους μηχανισμούς, έτσι, ώστε να προσφέρουν μια διεπαφή(*interface*) στο WEB. Αυτό το *interface* μπορεί να είναι για εξωτερικούς πελάτες, όπως είναι η περίπτωση των οργανισμών με παροχή υπηρεσιών μέσω Ίντερνετ, ή για εσωτερικές Web-εφαρμογές που να μοιράζονται πληροφορίες και τη ροή των εργασιών μεταξύ των διαφόρων ομάδων ή με συνεργάτες. Παρακάτω παρατίθενται τα πιο τυπικά είδη εξυπηρέτησης που πολλοί από τους σημερινούς οργανισμούς πρέπει να παρέχουν. Η λίστα, αν και όχι πλήρης, παρέχει μια εικόνα σύνδεσης των Web-τεχνολογιών με τον τρόπο και το είδος των υπηρεσιών που προσφέρονται από έναν οργανισμό:

- Εφαρμογές ηλεκτρονικού εμπορίου(Internet Business)
- Παρουσία οργανισμών στο διαδίκτυο
- Υποστήριξη offline πωλήσεων
- Εταιρικά interface
- Εφαρμογές ειδικού σκοπού

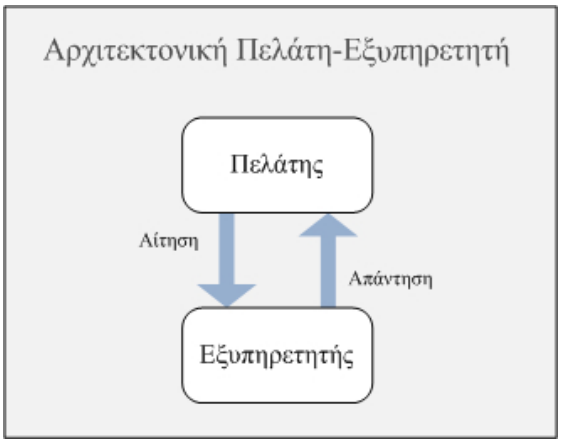
Όλα αυτά τα σενάρια απαιτούν Web-κεντρικές λύσεις που προσφέρουν υπηρεσίες με ποικίλα επίπεδα σε διαθεσιμότητα, ασφάλεια, επεκτασιμότητα και απόδοση. Η ανάπτυξη μιας τέτοιας Web-εφαρμογής ειδικά για WSN είναι ο στόχος της παρούσας εργασίας. Η ιδέα αυτή δεν είναι νέα. Υπάρχει πλήθος εφαρμογών που χρησιμοποιούν το διαδύκτιο για παρουσίαση, έλεγχο και διαχείριση τέτοιων δικτύων. Το διαδίκτυο, όμως, αποτελεί απλά το *interface* της όλης εφαρμογής που συνολικά σχετίζεται με το WSN δίκτυο. Απαιτείται, λοιπόν, η περιγραφή για την αρχιτεκτονική της εφαρμογής η οποία θα εξασφαλίζει σε αυτήν ευελιξία, σταθερότητα, ασφάλεια και επεκτασιμότητα.

1.5 Αρχιτεκτονική στις Web Εφαρμογές

Η αρχιτεκτονική ενός υπολογιστικού συστήματος επηρεάζει τη λειτουργικότητα, απόδοση, αποτελεσματικότητα, συντήρηση και επέκταση του καθόλη τη διάρκεια ζωής του και για αυτό χρειάζεται ιδιαίτερη προσοχή στο αρχικό στάδιο σχεδίασης ώστε το σύστημα να είναι σωστά δομημένο. Το ίδιο ισχύει και για την ανάπτυξη εφαρμογών οι οποίες πρέπει να είναι σωστά σχεδιασμένες και δομημένες. Στις εφαρμογές κυρίαρχο είδος αρχιτεκτονικής είναι αυτή του πελάτη-εξυπηρετητή(*client-server*) ή αλλιώς αρχιτεκτονική 2-επιπέδων. Σχέδια εφαρμογών που περιέχουν περισσότερα από δύο επίπεδα αναφέρονται ως πολυ-επίπεδα ή ν-επιπέδων.

1.5.1 Αρχιτεκτονική 2-Επιπέδων

Η πιο γνωστή αρχιτεκτονική τα τελευταία είκοσι χρόνια – κυρίως σε εφαρμογές βάσεων δεδομένων – είναι αυτή του πελάτη-εξυπηρετητή (client-server). Σε αυτή την αρχιτεκτονική(Βλέπε Σχήμα 1.3), ο πελάτης στέλνει ένα αίτημα(*request*) για δεδομένα στον εξυπηρετητή και αυτός επιστρέφει την απάντηση(*response*), την οποία επεξεργάζεται ο πελάτης και εμφανίζει στο χρήστη τα αποτελέσματα, όπως φαίνεται στο διάγραμμα. Το πρόβλημα με αυτή την προσέγγιση είναι ότι η εμφάνιση των δεδομένων και η επεξεργασία τους γίνεται από το ίδιο πρόγραμμα, τον πελάτη. Αν υπάρχουν πολλαπλά κανάλια διάχυσης της πληροφορίας ή συχνή αλλαγή στη μορφή παρουσίασης, τοτε θα πρέπει να αλλάζει κάθε φορά η client εφαρμογή.



Σχήμα 1.3: Μοντέλο Πελάτη-Εξυπηρετητή

1.5.2 Αρχιτεκτονική 3-Επιπέδων

Μερικά σχέδια εφαρμογών είναι πιο εξελιγμένα και αποτελούνται από τρία διαφορετικά μέρη: τους πελάτες (*clients*), τους εξυπηρετητές εφαρμογών (*application servers*) που

επεξεργάζονται δεδομένα για τους πελάτες και τους εξυπηρετητές βάσεων δεδομένων (*database servers*) οι οποίοι αποθηκεύουν δεδομένα για τους δεύτερους. Αυτή η θεώρηση της αρχιτεκτονικής μιας εφαρμογής ονομάζεται αρχιτεκτονική 3-επιπέδων, και είναι το πιο συχνά χρησιμοποιούμενο είδος της *client-server* αρχιτεκτονικής. Η αρχιτεκτονική 3-επιπέδων, θεωρούμε οτι αυτό το μοντέλο ικανοποιεί τις περισσότερες απαιτήσεις μας για το σχεδιασμό μιας εφαρμογής WSN. Επιθυμούμε να διαχωρίσουμε σε 3 επίπεδα τον όλο σχεδιασμό για να υπάρχει καλύτερη οργάνωση καθόλη τη διάρκεια της ανάπτυξης της εφαρμογής αλλά και επειδή η κατανομή αυτή δημιουργεί στην ουσία 3 διαφορετικά σημεία ενδιαφέροντος. Τα επίπεδα (Βλέπε Σχήμα 1.4) αυτά είναι:

Data Layer(Επίπεδο Δεδομένων) Το βασικό συστατικό στις περισσότερες εφαρμογές είναι τα δεδομένα. Τα δεδομένα πρέπει με κάποιο τρόπο γίνουν διαθέσιμα στο επίπεδο παρουσίασης. Το επίπεδο των δεδομένων είναι ένα ξεχωριστό στοιχείο, του οποίου μοναδικός σκοπός είναι η διάθεση των δεδομένων που βρίσκονται στη βάση δεδομένων και η αποστολή τους σε αυτόν που έκανε την αίτηση. Με την προσέγγιση αυτή, τα δεδομένα μπορούν να επαναχρησιμοποιηθούν, λογικά, με την έννοια ότι ένα τμήμα της εφαρμογής της επαναχρησιμοποιώντας το ίδιο ερώτημα (*query*) μπορεί να κάνει μια κλήση προς το επίπεδο δεδομένων, αντί να εκτελείται το ερώτημα πολλές φορές. Αυτό είναι γενικά πιο συντηρήσιμο (*maintainable*).

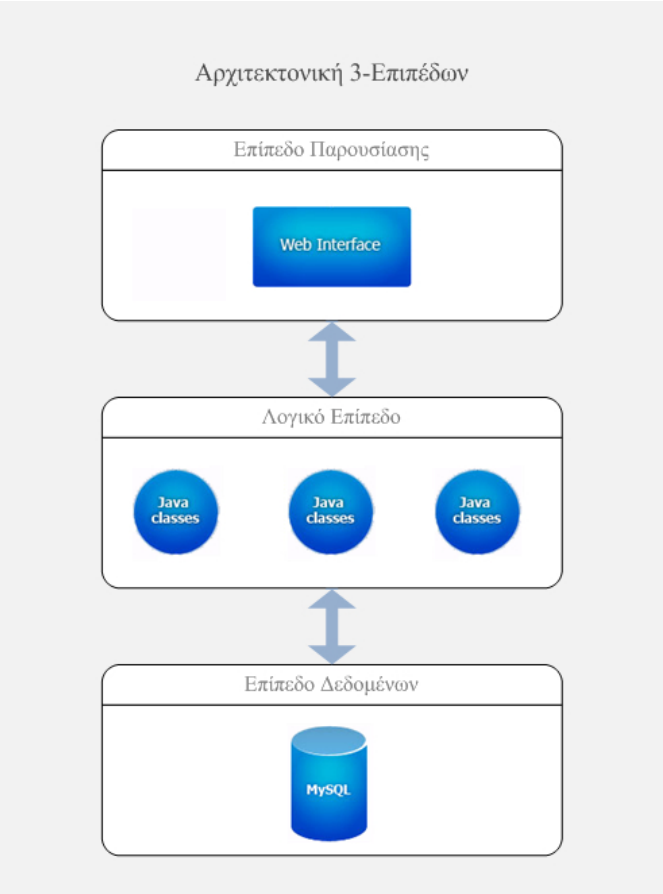
Logic Layer(Λογικό Επίπεδο) Αν και μια ιστοσελίδα θα μπορούσε να έχει πρόσβαση στα δεδομένα, επικοινωνώντας άμεσα με το επίπεδο δεδομένων, συνήθως επικοινωνεί μέσω ενός άλλου επιπέδου το οποίο ονομάζεται λογικό επίπεδο. Το επίπεδο αυτό ειναι ζωτικής σημασίας αφού ελέγχει τα δεδομένα εισόδου πριν καλέσει κάποια μέθοδο από το επίπεδο δεδομένων. Αυτό διασφαλίζει την εισαγωγή δεδομένων, πριν από τη διαδικασία, και μπορεί συχνά να εξασφαλίσει πως και τα αποτελέσματα είναι ακριβή επίσης.

Ωστόσο, οι κανόνες του λογικού επιπέδου δεν ισχύουν μόνο για την επικύρωση των δεδομένων. Αυτοί οι κανόνες εφαρμόζονται σε υπολογισμούς ή σε οποιαδήποτε άλλη ενέργεια που λαμβάνει χώρα στο συγκεκριμένο επίπεδο. Όσο περισσότερη λογική τοποθετήσουμε σε αυτό το επίπεδο τόσο το καλύτερο αφού η λογική αυτή μπορεί να επαναχρησιμοποιηθεί μεταξύ εφαρμογών.

Ένας από τους καλύτερους λόγους για την επαναχρησιμοποίηση της λογικής μιας εφαργμογής είναι ότι οι εφαρμογές που ξεκινούν από μικρές συνήθως αυξάνονται σε λειτουργικότητα. Για παράδειγμα, μια εταιρεία αρχίζει να αναπτύσει μια ιστοσελίδα, και καθώς αντιλαμβάνεται τις επιχειρηματικές της ανάγκες, αποφασίζει αργότερα να προσθέσει μια έξυπνη εφαρμογή για τους πελάτες της και μια νέα υπηρεσία για να συμπληρώσει την ιστοσελίδα. Το λογικό επίπεδο βοηθάει ώστε η

λογική να είναι τέτοια ώστε να υπάρχει μέγιστη δυνατότητα επαναχρησιμοποίησης.

Presentation Layer(Επίπεδο Παρουσίασης) Το επίπεδο παρουσίασης είναι το πιο σημαντικό στρώμα απλά γιατί είναι εκείνο που ο κάθε χρήστης βλέπει και χρησιμοποιεί. Ακόμη και με καλά δομημένα τα επίπεδα της λογικής και των δεδομένων, αν το επίπεδο παρουσίασης είναι άσχημα σχεδιασμένο, οι χρήστες αποκτούν μια φτωχή άποψη του συστήματος. Είναι καλύτερα να αφαιρεθεί οσο περισσότερο γίνεται η λογική της εφαρμογής έξω από το UI (*User Interface*) και αναπτυχθεί στο λογικό επίπεδο. Αυτό συνεπάγεται συνήθως περισσότερο κώδικα, αλλά, η υπέρβαση χρόνου (η οποία κυμαίνεται από ελάχιστη έως μέτρια, ανάλογα με το μέγεθος της εφαρμογής) κάνει την εφαρμογή πιο αποδοτική τελικώς.



Σχήμα 1.4: Παράδειγμα Αρχιτεκτονικής 3-Επιπέδων

1.5.3 Αρχιτεκτονική v-Επιπέδων

Γενικεύοντας την αρχιτεκτονική 3-επιπέδων φτάνουμε στην αρχιτεκτονική v-επιπέδων η οποία βασίζεται σε μια πιο γενική ιδέα σύμφωνα με την οποία μπορούμε να προσθέσουμε επίπεδα ανάλογα με τις ανάγκες της εφαρμογής. Τα πλεονεκτήματα των αρχιτεκτονικών v-επιπέδων είναι ότι είναι πολύ πιο επεκτάσιμες, αφού ισορροπούν και να διανέμουν το φορτίο της επεξεργασίας μεταξύ πολλών εξειδικευμένων κόμβων εξυπηρετητών. Αυτό με τη σειρά του βελτιώνει τη συνολική απόδοση του συστήματος και της αξιοπιστίας, αφού περισσότερο από το φορτίο επεξεργασίας μπορεί να εξυπηρετηθεί ταυτόχρονα.

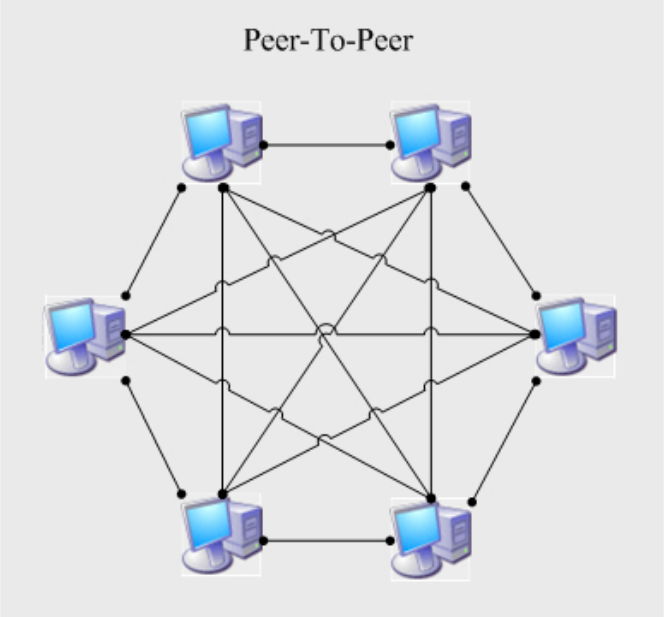
Τα μειονεκτήματα της αρχιτεκτονικής v-επιπέδων περιλαμβάνουν:

1. Μεγαλύτερο φορτίο στο ίδιο το δίκτυο, λόγω της μεγαλύτερης κίνησης του δικτύου.
2. Μεγαλύτερη δυσκολία στον προγραμματισμό και τη δοκιμή τους από αρχιτεκτονικές 2-επιπέδων, επειδή περισσότερες συσκευές πρέπει να επικοινωνούν προκειμένου να ολοκληρωθεί μια αίτηση ενός πελάτη.

1.5.4 Αρχιτεκτονική Peer-To-Peer

Ένα άλλο είδος αρχιτεκτονικής δικτύου που είναι γνωστό ως *peer-to-peer* (Βλέπε Σχήμα 1.5), γιατί κάθε κόμβος ή στιγμιότυπο του προγράμματος μπορεί ταυτόχρονα να λειτουργήσει ως πελάτης ή εξυπηρετητής, διότι ο καθένας έχει ισοδύναμα καθήκοντα και ιδιότητες. Το ακρωνύμιο P2P αποτελεί συχνά συντομογραφία των *peer-to-peer* αρχιτεκτονικών. Οι κόμβοι προσαρμόζονται και αυτοδιοργανώνονται καθώς εισέρχονται ή αποχωρούν από το σύστημα, ικανοποιώντας την ιδιότητα της κλιμάκωσης και της ανοχής στις αποτυχίες. Η απλή δομή, το μηδαμινό κόστος και η άναρχη ροή πληροφορίας είναι τα στοιχεία που καθιστούν τη λειτουργία των P2P δικτύων μοναδική.

Η φιλοσοφία αυτών των δικτύων δίνει τη δυνατότητα στους συμμετέχοντες, της δημιουργίας δυναμικά αναπτυσσόμενων χώρων, το περιεχόμενο των οποίων καθορίζεται από τους ίδιους τους χρήστες. Από την άλλη τα δίκτυα P2P καθιστούν δυνατή την αντιγραφή και διανομή αρχείων μεταξύ χρηστών, τα οποία προστατεύονται από πνευματικά δικαιώματα, όπως τραγούδια, ταινίες και λογισμικό, χωρίς τη συναίνεση του κατόχου των πνευματικών δικαιωμάτων. Η ευρεία χρήση των δικτύων P2P για αυτόν τον σκοπό συντέλεσε ώστε τα δίκτυα να ταυτιστούν με έννοιες όπως «παρανομία» και να υποστούν πόλεμο τόσο τα ίδια και οι δημιουργοί τους, όσο και οι χρήστες τους. Παράλληλα έγιναν όμως και ενέργειες χρηστών αλλά και ολόκληρων κοινοτήτων για να



Σχήμα 1.5: Παράδειγμα Αρχιτεκτονικής Peer-To-Peer

καταδείξουν ότι τα P2P χρησιμοποιούνται και για καλό σκοπό. Ενδεικτική είναι πρόσφατη προσπάθεια πανεπιστημίων και ερευνητικών κέντρων για τη δημιουργία μιας εφαρμογής για P2P δίκτυα. Το εγχείρημα αυτό γνωστό και ως LionShare βασίζεται στα δεύτερης γενιάς P2P δίκτυα και φτιάχνεται για το διαμοιρασμό στους χρήστες του ακαδημαϊκού υλικού. Ένα άλλο ίσως πιο γνωστό τέτοιο δίκτυο είναι το SETI@home[2].

1.5.5 Χαρακτηριστικά

Κάποια από τα χαρακτηριστικά τέτοιων αρχιτεκτονικών είναι τα εξής:

- Στις περισσότερες περιπτώσεις, μια *client-server* αρχιτεκτονική επιτρέπει στα υπολογιστικά συστήματα να κατανείμουν τους ρόλους και τις ευθύνες τους μεταξύ πολλών ανεξάρτητων υπολογιστών οι οποίοι είναι γνωστοί μόνο μέσω ενός δικτύου. Αυτό δημιουργεί ένα πρόσθετο πλεονέκτημα σε αυτήν την αρχιτεκτονική: μεγαλύτερη ευκολία συντήρησης. Για παράδειγμα, είναι δυνατή η αντικατάσταση, επισκευή, αναβάθμιση, ή ακόμα και ο επανεντοπισμός ενός εξυπηρετητή ενώ οι πελάτες του παραμένουν απληροφόρητοι και συνάμα ανεπηρέαστοι από την αλλαγή αυτή. Η ανεξαρτησία από τέτοιου είδους αλλαγές αναφέρεται επίσης ως ενθυλάκωση(*encapsulation*).

[2]Το SETI@home (”SETI at home”) αποτελεί ένα project κατανεμημένων υπολογισμών (grid computing) το οποίο χρησημοποιεί συνδεδεμένους στο διαδίκτυο υπολογιστές, φιλοξενείται από το Εργαστήριο Επιστημών του Διαστήματος(Space Sciences Laboratory), στο Πανεπιστήμιο της Καλιφόρνια, στο Μπέρκλεϋ των ΗΠΑ. Ο όρος SETI είναι συντομογραφία του Search for Extra-Terrestrial Intelligence. Το SETI@home ανακοινώθηκε στις 17 Μαΐου του 1999.

- Όλα τα δεδομένα είναι αποθηκευμένα στους εξυπηρετητές, οι οποίοι γενικά έχουν πολύ καλύτερους ελέγχους ασφάλειας από ό, τι οι περισσότεροι πελάτες τους. Οι εξυπηρετητές μπορούν να ελέγχουν καλύτερα την πρόσβαση και τους πόρους, ώστε να εξασφαλίζεται ότι μόνον οι πελάτες με τα κατάλληλα δικαιώματα μπορούν να έχουν πρόσβαση και να αλλάξουν δεδομένα.

- Εφόσον η αποθήκευση δεδομένων είναι συγκεντρωμένη, οι ενημερώσεις για τα δεδομένα αυτά είναι πολύ ευκολότερο να διαχειριστούν από ό, τι θα ήταν δυνατό στο πλαίσιο ενός P2P παραδείγματος. Στο πλαίσιο μιας P2P αρχιτεκτονικής, τα δεδομένα των ενημερωμένων εκδόσεων χρειάζεται να διανεμηθούν και να εφαρμοστούν σε κάθε ”peer” στο δίκτυο, διαδικασία η οποία είναι τόσο χρονοβόρα και επιρρεπής σε λάθη μιας και είναι πιθανόν το πλήθος των peers να είναι χιλιάδες ή ακόμα και εκατομμύρια.

- Πολλές ώριμες *client-server* τεχνολογίες είναι ήδη διαθέσιμες, οι οποίες είχαν σχεδιαστεί ώστε να διασφαλίζεται η ασφάλεια, η «φιλικότητα» της διεπαφής προς χρήστη, καθώς και ευκολία χρήσης.

Ο σχεδιασμός με διαχωρισμό σε επίπεδα μπορεί να βοηθήσει στην ανάπτυξη των κατανεμημένων εφαρμογών. Επειδή ο κώδικας είναι κατανεμημένος στα επίπεδα, μπορεί να προστεθεί στο σχεδιασμό της εφαρμογής ένα επίπεδο που να διευκολύνει τη χρήση των απομακρυσμένων και Web υπηρεσιών, με ελάχιστη εργασία. Οι εφαρμογές των συστημάτων WSN είναι κατανεμημένες και επομένως η αρχιτεκτονική επιπέδων είναι μονόδρομος για τη σχεδίαση τους. Όπως θα δούμε παρακάτω(Βλέπε 1.6, 2), οι κατεμεμημένες εφαρμογές WSN χρησιμοποιούν αρχιτεκτονική επιπέδων.

1.6 Συστήματα WSN

1.6.1 Εισαγωγικά Θέματα

Όπως ειπώθηκε και νωρίτερα (Βλέπε Παράγραφος 1.1), τα ασύρματα δίκτυα αισθητήρων γίνονται όλο και πιο δημοφιλή από την ερευνητική κοινότητα. Το κόστος τους έχει μια σταθερά πτωτική τάση τα τελευταία χρόνια, επιτρέποντας περισσότερους ερευνητές να δημιουργήσουν το δικό τους σύστημα δοκιμών δικτύου αισθητήρων. Ο αριθμός των διαθέσιμων ειδικών περιβάλλοντων λογισμικών για τα εν λόγω δίκτυα επίσης αυξάνεται διαρκώς με αμείωτο ρυθμό.

Κατά τη διάρκεια της εξέλιξης αυτών των τεχνολογιών τα συστήματα και οι εφαρμογές έχουν προταθεί και αναπτυχθεί σε ”κύματα”. Αρχικά, οι πρώτες εφαρμογές που

εμφανίστηκαν (παρακολούθηση στόχου, αίθουσα επιτήρησης, περιβαλλοντικής παρακολούθησης, κλπ) αφορούσαν έναν αριθμό κόμβων αισθητήρων που ανήκαν σε απλό δίκτυο αισθητήρων. Η έμφαση δόθηκε κυρίως στην παροχή αλγορίθμων και πρωτοκόλλων για την αποτελεσματική διαχείριση ενέργειας, τη δρομολόγηση, τον εντοπισμό, κλπ. Στη συνέχεια, ήρθε το επόμενο κύμα των εφαρμογών που παρέχει πρόσθετες δυνατότητες και υπηρεσίες, και σε κάποιο βαθμό χρησιμοποιεί έναν αριθμό αισθητήρων πυλών (*gateways*), θέτοντας έτσι νέα πρότυπα στους όρους του μεγέθους και της ετερογένειας, επιτρέποντας, επίσης, τη διαχείριση πολλών δικτύων αισθητήρων σαν ένα μοναδικό. Το τρίτο κύμα, που σήμερα έχει αρχίσει να κερδίζει έδαφος, συνδέεται με εφαρμογές που ανήκουν στη σφαίρα του διαδικτύου. Έτσι, στις μέρες μας, σε αντίθεση με προηγούμενες εφαρμογές WSN, η διεπαφή (*interface*) μεταξύ διαφορετικών δικτύων και εφαρμογων αποκτά μεγαλύτερη σημασία.

Αυτή η εξέλιξη των WSN εφαρμογών, σε συνάρτηση με άλλους σημαντικούς παράγοντες, όπως η έλλειψη τυποποίησης σε εξοπλισμό και λογισμικό, οδήγησε σε:

i *Πληθώρα παραδειγμάτων προγραμματισμού έτσι ώστε να συμπληρωθεί το κενό:* Ένας μεγάλος αριθμός διαφορετικών προσεγγίσεων που αφορούν στο *middleware* και σε περιβάλλοντα για τη διαχείριση του δικτύου αισθητήρων έχει προταθεί τα τελευταία χρόνια. Η επιλογή μιας από όλες τις διαφορετικές προσεγγίσεις μπορεί να γίνει ανάμεσα στην ευκολία χρήσης, την εκφραστικότητα (όσον αφορά τη διαθέσιμη λειτουργικότητα του δικτύου) και τη χρήση των πόρων των κόμβων (την ενέργεια, τη μνήμη, κ.λπ.).

ii *Πλήθος λειτουργιών και υπηρεσιών:* Σταδιακά ο αριθμός των υπηρεσιών που παρέχει το λογισμικό WSN έχει αυξηθεί, λόγω των επιπρόσθετων απαιτήσεων των εφαρμογών. Σήμερα, ολοένα και περισσότερο, γίνεται λόγος για την ετερογένεια, την ποιότητα των υπηρεσιών, την ασφάλεια, την εμπιστοσύνη, την τυποποίηση των διεπαφών (*interfaces*), πέρα από την αποτελεσματική διαχείριση πόρων, τα ποσοστά επιτυχίας της παράδοσης των δεδομένων, κλπ. Επίσης, κόμβοι που υποστηρίζουν κίνηση (*actuators*) και ενσύρματα δίκτυα αισθητήρων αποκτούν ολοένα και μεγαλύτερη σημασία.

iii Λειτουργία των περιβάλλοντων λογισμικού σε πολλά επίπεδα: Οι πιο πρόσφατες εφαρμογές WSN μπορεί να απαιτούν διαφορετικές αρχιτεκτονικές από αυτές που χρησιμοποιούνταν στις πρώτες υλοποιήσεις τέτοιων εφαρμογών. Για παράδειγμα, σε ορισμένες περιπτώσεις, προτείνεται να χρησιμοποιούνται κλιμακωτά δίκτυα, που χρησιμοποιούν διαφορετικά είδη κόμβων αισθητήρων, δημιουργώντας επιπλέον ιεραρχικά επίπεδα (Βλέπε Κεφάλαιο 1.5). Επίσης, όπως είναι προφανές για να είναι μια εφαρμογή κλίμακας διαδικτύου το λογισμικό στην κορυφή των κόμβων και των πυλών (*gateways*) ενός WSN είναι απαραίτητο.

1.6.2 Θέματα Σχεδιασμού και Απαιτήσεων

Όσον αφορά τις απαιτήσεις και άλλα θέματα που διέπουν το σχεδιασμό και την υλοποίηση των συστημάτων WSN, θα πρέπει να σημειωθεί ότι η κατάσταση εξακολουθεί να είναι ρευστή. Εξακολουθεί να μην υπάρχει ευρεία συναίνεση πάνω σε πράγματα όπως είναι οι πλατφόρμες υλικού και ο σχεδιασμός λειτουργικών συστημάτων, και υπάρχει και το συνεχώς διευρυνόμενο πεδίο WSN εφαρμογών. Επίσης, το όραμα του ίδιου WSN αλλάζει, όπως η εφαρμοσμένη έρευνα στον τομέα προχωρεί. Από το αρχικό όραμα της "εκατοντάδες ή δεκάδες χιλιάδες των κόμβων αισθητήρων σε κάθε δίκτυο αισθητήρων", κινούμαστε προς ένα πιο ρεαλιστικό αριθμό κόμβων (στην κλίμακα των χιλιάδων σήμερα), ή σε μια σειρά από δίκτυα αισθητήρων με ένα μάλλον μικρό αριθμό αισθητήρων σε κάθε τέτοιο δίκτυο. Αυτό σχετίζεται άμεσα με τις πρακτικές δυσκολίες που αντιμετωπίζουν οι προγραμματιστές στην ανάπτυξη WSN εφαρμογών.

Έχοντας υπόψη αυτές τις παραμέτρους, υπάρχουν ορισμένα θέματα και απαιτήσεις που είναι κοινά στις περισσότερες από τις πλατφόρμες WSN που προτείνονται μέχρι στιγμής, και τα πράγματα που αλλάζουν είναι οι αφαιρέσεις που γίνονται για να είναι πιο εύκολο το έργο των προγραμματιστών, ως αποτέλεσμα των διαφορετικών προσεγγίσεων που έχουν ληφθεί. Η αποδοτική διαχείριση πόρων είναι οριστικά το θέμα στο σχεδιασμό λογισμικού για WSN - περιορισμοί στις δυνατότητες επεξεργασίας, την ενέργεια και το εύρος ζώνης επικοινωνίας, απλά λόγια, δεν μπορούν εύκολα να αγνοηθούν στο WSN. Στη συνέχεια, ανοχή σε σφάλματα και η προσαρμοστικότητα είναι οι άλλοι παράγοντες που παραδοσιακά χρησιμοποιούνται ως θέματα σχεδιασμού για WSN λογισμικό. Οι εργασίες που αφορούν δεδομένα είναι άλλος ένας σημαντικός παράγοντας. Η συνάθροιση δεδομένων αξιολογήθηκε ως ένας σημαντικός παράγοντας, αλλά τα τελευταία χρόνια υπήρξαν προτάσεις για μεταφορά της πολυπλοκότητας σε ανώτερες βαθμίδες του δικτύου (δηλαδή, τους κόμβους με περισσότερες επεξεργαστικές ικανότητες). Ασφάλεια, εμπιστοσύνη, ετερογένεια και επεκτασιμότητα είναι οι τέσσερις πολύ σημαντικοί παράγοντες στην ανάπτυξη μελλοντικών WSN εφαρμογών, για να γίνει ακόμη πιο κοντινό το όραμα της ανάπτυξης εφαρμογών για ασύρματα δίκτυα τεράστιας κλίμακας.

1.6.3 Κατηγοριοποίηση Συστημάτων

Τελευταία, παρατηρείται μια κλίση για μετάβαση από τις περιοχές δικτύων αισθητήρων όπου φιλοξενείται ένα μοναδικό δίκτυο, σε εφαρμογές που αφορούν σε πολλαπλά δίκτυα αισθητήρων. Σύμφωνα με αυτή την πραγματικότητα, τα συστήματα εφαρμογών WSN κατηγοριοποιούνται ως εξής:

1.6.3.1 Σκοπός του συστήματος

Σύμφωνα με αυτό το κριτήριο, το ερώτημα που γεννιέται είναι αν τα συστήματα ασυρμάτων δικτύων αισθητήρων θα πρέπει να παρέχουν έναν αριθμό από λειτουργίες και χαρακτηριστικά μέσω μιας εφαρμογής που έχει υλοποιηθεί (περιορισμένη επεκτασιμότητα) ή αν θα παρέχουν ένα σύνολο από εργαλεία και API για την ανάπτυξη τέτοιων εφαρμογών που θα λειτουργούν ως διεπαφές προς άλλα περιβάλλοντα. Στις περισσότερες των περιπτώσεων ακολουθείται ένα μεικτό μοντέλο.

Πλατφόρμες όπως το MoteView [1] και το ScatterViewer [2] ανοίκουν στην κατηγορία των εργαλείων με περιορισμένη επεκτασιμότητα παρέχοντας κάποιες λειτουργίες που είναι δύσκολα επεκτάσιμες και η αρχιτεκτονική τους είναι πολύ στενά δεμένη ενώ πλατφόρμες όπως το SenseWeb [3], το P2PBridge [4], το Hourglass [5], το GSN [6], ανοίκουν στην κατηγορία της ανάπτυξης εργαλείων εφαρμογών, παρέχοντας εργαλεία αρκετά γενικά για την ανάπτυξη WSN μεγάλης κλίμακας, καθένα με το δικό του τρόπο. Τα SNACK [7] and DSN [8] παρέχουν στους χρήστες γλώσσες σε αφαιρετικό επίπεδο για την δημιουργία νέων συστατικών εφαρμογών σαν αντικατάστατα της ευρείας χρήσης γλώσσας προγραμματισμού nesC. Οι περισσότερες από τις υπόλοιπες εφαρμογές πέφτουν κάπου στο ενδιάμεσο. Υπάρχουν επίσης κάποιες λύσεις όπως οι Abstract Regions [9], Hood [10], και Neidas [11] οι οποίες παρέχουν λογικές αφαίρεσης των γειτονικών κόμβων δικτύου για να διευκολύνουν την αλληλεπίδραση κόμβων και το διαμειρασμό δεδομένων. Το Kairos [12] παρέχει λογικές αφαίρεσης για τον προγραμματισμό τέτοιων συστημάτων αλλά στη σφαίρα ενός δικτύου καθολικού χαρακτήρα υλοποιείται σαν πρόσθετο της Python. Το Tenet [13] ορίζει μια γλώσσα για την οργάνωση των εργασιών των αισθητήρων, παρέχοντας μια βιβλιοθήκη με ενα σύνολο από καθήκοντα για τέτοιους αισθητήρες και αρχιτεκτονική επιπέδων, όπου οι απαιτητικές εργασίες εκτελούνται στους κύριους κόμβους, οι οποίοι είναι πιο ικανοί από τους υπόλοιπους.

1.6.3.2 Προσαρμοστικότητα

Το κριτήριο της προσαρμοστικότητας έχει να κάνει με το αν το λογισμικό παρέχει αυτοοργανωτικές λειτουργίες ή αν οι παράμετροι των δικτύων μπορούν ή πρέπει να οριστούν από του διαχειριστές. Οι περισσότερες εφαρμογές δικτύων αισθητήρων παρέχουν κάποια χαρακτηριστικά αυτο-οργάνωσης και διαχείρισης δικτύου. Τέτοια χαρακτηριστικά επιλύουν προβλήματατα όπως ο χρονικός συγχρονισμός (π.χ. TinyDB [14]), δρομολόγηση και πρωτόκολλα (π.χ. η στοίβα ZigBee), αλλά γενικά οι διαχειριστές έχουν συνήθως ένα μικρό έλεγχο. Από την άλλη πλευρά κάποιες εφαρμογές (όπως το MoteView [1]) απαιτούν την παρέμβαση του διαχειριστή του δικτύου για να φέρουν σε πέρας

ακόμη και απλές εργασίες. Υπάρχει επίσης το θέμα του αν πολιτικές που παρέχουν δυναμικό χαρακτήρα μπορούν να ενημερωθούν δυναμικά και οι ίδιες. Το RUNES [15] είναι υπεύθυνο για τέτοια θέματα που επιτρέπουν τα συστατικά λογισμικού να ενημερώνονται "*on-the-fly*". Το Impala [16] είναι επίσης ικανό να ενημερώνεται κατά το χρόνο εκτέλεσής του και να προσαρμόζεται σε συνθήκες αλλαγής των συνθηκών του δικτύου και αστοχίες υλικού.

1.6.3.3 Ανοχή στα λάθη

Η ανοχή στα λάθη στην στα WSN είναι μια ιδέα της οποίας η σημασιολογία ποικίλει σημαντικά σε σχέση με το επίπεδο του δικτύου για το οποίο γίνεται λόγος. Στο μέσο επίπεδο, το επίπεδο του συστήματος και των εφάρμογών των WSN εφαρμόχεται με πολλαπλούς τρόπους. Για παράδειγμα, σε περίπτωση που ένας κόμβος αντιμετωπίσει πρόβλημα με το υλικό ή το λογισμικό του, θα πρέπει να υπάρχει ένας τρόπος εξέτασης του αν σε αυτόν τον κόμβο είχε ανατεθεί κάποια εργασία ή αν υπάρχει κάποια αντίστροφη διαδικασία ανακάλυψής του. Σαν παράδειγμα, το TASK [17] ασχολείται με τέτοιες περιπτώσεις χρησιμοποιώντας ένα άγρυπνο χρονομετρητή για να διαπιστώνει σφάλματα και να επαναφέρει κόμβους από τρυγύρω κόμβους που λειτουργούν σωστά χρησιμοποιώντας ένα σχέδιο με διαμοιραζόμενα *queries*. Δυο άλλα θέματα είναι το θέμα της αστοχίας υλικού και το θέμα της αξιόπιστης μετάδοσης δεδομένων και δυαδικού κώδικα διαμέσου του δικτύου αισθητήρων.

1.6.3.4 Εκτέλεση λογισμικού

Το κριτήριο αυτό αναφέρεται στο αν το λογισμικό τρέχει σε κάποια υπάρχουσα εφαρμογή δικτύων αισθητήρων ή αν τρέχει επισης σε επίπεδο κόμβων αισθητήρων. Οι πιο συνηθισμένες υλοποιήσεις έχουν χαρακτηριστικά τα οποία συνδέονται με την εκτέλεση οτυ λογισμικού σε επίπεδο κόμβων αισθητήρων. Από την άλλη πλευρά, το SenseWeb και το GSN χρησιμοποιούν κάποιες αφαιρετικές προοπτικές για να επικοινωνούν με το λογισμικό που τρέχει στις πύλες των δικτύων αιθητήρων και επίσης δεν εξαρτώνται απί λογισμικο που έχει υλοποιηθεί από τα ίδια (προφανώς οι οδηγοί για κάθε εφαρμογή θα πρέπει να έχουν υλοποιηθεί).

1.6.3.5 Αριθμός WSN, Ετερογένεια

Ένα άλλο σημαντικό χαρακτηριστικό είναι ο αριθμός των διαφορετικών δικτύων αισθητήρων στο σύστημα και κατά πόσο αυτά τα δίκτυα αποτελούνται από διαφορετικά

υποσυστατικά σε υλικό και λογισμικό. Η πλειοψηφία των πλατφορμών που εμφανίστηκαν τα τελευταία χρόνια έδωσαν πολύ προσοχή σε τέτοιες υποθέσεις. Πολλαπλές πηγές πληροφόρησης, οι οποίες ενδεχομένως να προέρχονται από εντελώς διαφορετικά δίκτυα αισθητήρων, θα είναι ένα βασικό χαρακτηριστικό στις εφαρμογές WSN μεγάλης κλίμακας. Όσον αφορά τα απλά δίκτυα αισθητήρων, κάποιες αντιπροσωπευτικές προσεγγίσεις είναι τα TinyDB [14], Cougar [18], MoteView [1], Mat'e [19], ScatterViewer [2] και Agilla [20]. Οι προσεγγίσεις αυτές είναι, επίσης, περιορισμένες όσον αφορά την ετερογένεια, εν μέρει επειδή στηρίζονται στη χρήση ειδικών πλατφορμών λειτουργικών συστημάτων και υλικού, αν και κάποιες παρέχουν μηχανισμούς για τον προσδιορισμό π.χ., νέων αισθητήρων. Το Impala, από την άλλη πλευρά, αναλαμβάνει σχεδόν εξ ολοκλήρου ομογενή συστήματα.

Η διαχείριση των πολλαπλών δικτύων αισθητήρων είναι μια έννοια που συζητήθηκαν στο jWebdust [21]. Το jWebdust προτείνει την έννοια του εικονικού δικτύου αισθητήρνω που αποτελείται από έναν αριθμό διακριτών δικτύων αισθητήρων που μπορούν να διαχειρίζονται ένα μοναδικό. Η ιδέα της επέκτασης της διαχείρισης πολλαπλών δικτύων αισθητήρων, και η θέσπιση μιας peer-to-peer έννοιας, έρχεται να στηριχθεί στα Hourglass [5] και Global Sensor Networks [22] τα οποία αποτελούν περιβάλλοντα για την υποστήριξη γεωγραφικά διαχωριζόμενων δικτύων και συνδέονται με εφαρμογές παρέχοντας μια σειρά από υπηρεσίες. Η βασική ιδέα είναι η διεπαφή των δικτύων αισθητήρων και των εφαρμογών για το χρήστη με σκοπό τη δημοσίευση τοπικά δημιουργομένων ρευμάτων δεδομένων ή ρευμάτων αιτήσεων ενδιαφέροντος χωρίς ανησυχία για τις υποκείμενες υποδομές αισθητήρων. Οι SenseWeb, GSN, MetroSense [23], CarTel [24], επιπλέον, φαίνεται να υποστηρίζουν την ετερογένεια σε μεγαλύτερο βαθμό, τουλάχιστον θεωρητικά. Ένα άλλο ενδιαφέρον πρόσφατο περιβάλλον που επιτρέπει τη χρήση πολλαπλών δικτύων αισθητήρων είναι το Sensor Web Enablement [25].

Ένας στόχος των μελλοντικών WSN εφαρμογών θα πρέπει να είναι η υποστήριξη των πολλαπλών WSN με παρόμοιο τρόπο με τον τρόπο που συνδέονται οι πελάτες σε ένα δίκτυο P2P για κοινή χρήση αρχείων, έτσι ώστε για την απλοποίηση της όλης διαδικασίας, να επεκτείνουν τις λειτουργίες παρέχοντραι στις υπάρχουσες πλατφόρμες, και η πραγματική υποστήριξη WSN εφαρμογών κλίμακας διαδικτύου.

1.6.3.6 Κινητικότητα κομβων ή πυλών

Ένα άλλο σημαντικό χαρακτηριστικό είναι αν το σύστημα υποστηρίζει κινητικότητα των κόμβων και των πυλών. Σήμερα, φαίνεται να υπάρχουν μόνο λίγα τα συστήματα που να υποστηρίζουν τέτοια χαρακτηριστικά. Όσο το όραμα του μέλλοντος των WSN

μεγαλώνει ώστε συμπεριληφθούν νέες εφαρμογές, η κινητικότητα αναμένεται να διαδραματίσει κρίσιμο ρόλο στο μέλλον. Η αρχιτεκτονική και η υλοποίηση μιας πλατφόρμας WSN θα επηρεαστούν σε μεγάλο βαθμό στην περίπτωση υιοθέτησης αυτών των ιδεών. Η ιδέα της χρησιμοποίησης κινητών συσκευών, όπως κινητά τηλέφωνα, για να ενεργούν ως αισθητήριες συσκευες ή ενδιάμεσα προϊόντα, είναι ένα παράδειγμα μιας τέτοιας ιδέας. Τα CarTel, Skylark [**?**] και MetroSense κάνουν χρήση αυτής της ιδέας (της χρήσης κινητών τηλεφώνων) σε κάποιο βαθμό, ενώ το SenseWeb μπορεί να τη χρησιμοποιήσει πιθανώς και αυτό. Το TinyLIME [26] έχει σχεδιαστεί ειδικά για να ικανοποιεί τις ανάγκες των κινητών πυλών (*mobile gateways*), παρότι η αρχιτεκτονική του με κάποιο τρόπο το κάνει λιγότερο ελκυστικό, δεδομένου ότι χρησιμοποιεί ένα *single-hop* σύστημα επικοινωνίας. Το Impala, επίσης, είναι σχεδιασμένο να λειτουργεί σε σενάρια υψηλής κινητικότητας , στην πραγματικότητα, σε κάποιο βαθμό, η αρχιτεκτονική του βασίζεται στην κινητικότητα.

1.6.3.7 Στιγμιότυπα συστήματος

Το παρόν κριτήριο ασχολείται με το άν υπάρχουν ξεχωριστά υποδίκτυα σε ένα σύστημα και πως αυτά επικοινωνούν και αλληλεπιδρούν μεταξύ τους. Στις περισσότερες περιπτώσεις, ακόμη και αν υπάρχει υποστήριξη πολλαπλών WSN, δεν έχουν άμεση ή έμμεση επικοινωνία μεταξύ τους, καθώς και όλα τα αποτελέσματα αποθηκεύονται σε μια κεντρική βάση δεδομένων (όπως προβλέπει το TinyDB). Το IrisNet [27], ήταν μια μετακίνηση από αυτή την προσέγγιση, χρησιμοποιώντας μια κατανεμημένη βάση δεδομένων και επιτρέποντας πολλαπλά στιγμιότυπα. Τα Hourglass, GSN, Skylark και P2PBridge επιτρέπουν μεταξύ τους WSN-λειτουργικότητα μέσω P2P συστημάτων επικοινωνίας. Επίσης, το Agimone [28] είναι ένα παράδειγμα των κινητών πρακτόρων συνδυασμένων με κάποιο λογισμικό γεφύρωσης για να επιτρέψουν τη μετανάστευση των κινητών υπαλλήλων, από το ένα δίκτυο αισθητήρων δίκτυο στο άλλο (πράγμα που θα επιτρέψει τόσο στα δεδομένα και στη λειτουργικότητα να περνά από δίκτυο σε δίκτυο).

1.6.3.8 Διεπαφή

Οι πρώτες πλατφόρμες για WSNs παρείχαν διεπαφές στον τελικό χρήστη και τον υπόλοιπο κόσμο μέσω προσαρμοσμένων εφαρμογών διεπαφών (όπως το TinyDB) ή μέσω μιας web σελίδας. Αυτή η τάση φαίνεται να μετατοπίζεται, επίσης, προς την κατεύθυνση παροχής διασυνδέσεων μέσω XML, SOAP, κλπ., προκειμένου να ενταχθούν με άλλα περιβάλλοντα, όπως είναι η περίπτωση του ArchRock [29] και του Octavex [30]. Ανάλογες προσεγγίσεις ακολουθούνται στο Sensilink [31], το SynapSense OneClick [32] και

το XServe [33] (μέρος της theMoteWorks πλατφόρμας). Φυσικά, εν γένει, όταν θα έχετε μια βάση δεδομένων στο σύστημά σας μπορείτε πάντα να ορίσετε διεπαφές ώστε να την ενσωματώσετε, με κάποιο τρόπο, με άλλες εφαρμογές. Μια προσέγγιση για την παροχή διασύνδεσης προς τον υπόλοιπο κόσμο, είναι μέσω της χρήσης των πρωτοκόλλων P2P, όπως και στιςν HourGlass, P2PBridge, Skylark. Συνήθως αυτό γίνεται με χρήση ενός υποστρώματος λογισμικού P2P , όπως το JXTA [34]. Επίσης, υπάρχει η προσέγγιση του δικτύου πυλών αισθητήρων που λειτουργούν ως γέφυρες, όπως στο ArchRock και το TinyREST [35], αντιστοιχίζοντας τα αναγνωριστικά IDs των εσωτερικών WSN κόμβων σε IPv4 και IPv6 διευθύνσεις.

1.6.3.9 Ασφάλεια και θέματα εμπιστοσύνης

Με τους όρους ασφάλεια και εμπιστοσύνη, θα αναφερόμαστε στην κρυπτογραφημένη επικοινωνία και την άδεια πρόσβασης, αντίστοιχα. Εκτός από τη χρήση, π.χ., του TinySec [36], σχετικά με το επίπεδο του κόμβο αισθητήρων, δεν έχει γίνει πραγματικά πολύ έργο σε αυτούς τους τομείς, όταν αναφερόμαστε τόσο στην εμπιστοσύνη μέσα στο δίκτυο αισθητήρων όσο και στην επικοινωνία εφαρμογών μεταξύ WSN. Για παράδειγμα, το πρόβλημα αποδοχής ενός αισθητήρα που προστέθηκε πρόσφατα κόμβο ως ένα αξιόπιστο μέλος του δικτύου έχει μεγάλη σημασία. Επιπλέον, δεν είναι σαφές πώς ένας αισθητήρας ή ένας χρήστης από το ένα WSN ή μια εφαρμογή-πελάτη θα μπορούσε ή θα πρέπει να έχει πρόσβαση σε τις λειτουργίες και τους πόρους του άλλου WSN. Ορισμένα από τα θέματα αυτά αντιμετωπίζονται με Sensor Web Enablement, όπου οι τελικοί χρήστες μπορούν να έχουν διαφορετικούς ρόλους και προνόμια στην πρόσβαση δεδομένων και την διαχείριση διαφορετικών δικτύων αισθητήρων μέσω του συστήματος.

Κεφάλαιο 2

Αρχιτεκτονική

Στο κεφάλαιο αυτό θα παρουσιάσουμε την αρχιτεκτονική από δύο υπάρχοντα περιβάλλοντα πάνω σε ασύρματα δίκτυα αισθητήρων και θα καταλήξουμε στην αρχιτεκτονική του Webdust2συστήματος το οποίο αποτελεί τελικά την εξέλιξή τους και βρίσκεται υπο συνεχή ανάπτυξη και επέκταση. Το πρώτο είναι το *jWebDust* το οποίο αποτελεί ένα ευέλικτο σύστημα για την ανάπτυξη εφαρμογών για ασύρματα δίκτυα μικροαισθητήρων, υλοποιημένο σε Java και TinyOS. Το δεύτερο είναι το *ShareSense* το οποίο είναι ένα περιβάλλον παρακολούθησης πολλαπλών δικτύων μικροαισθητήρων που βασίζεται σε μία αρχιτεκτονική ομοτίμων (peer-to-peer) για την επικοινωνία ανάμεσα σε ασύνδετα δίκτυα και χρήστες. Ας κοιτάξουμε παρακάτω αναλυτικότερα τα χαρακτηριστικά αυτών των περιβαλλόντων και τελικώς θα αναλύσουμε την αρχιτεκτονική της εφαρμογής μας(Webdust2).

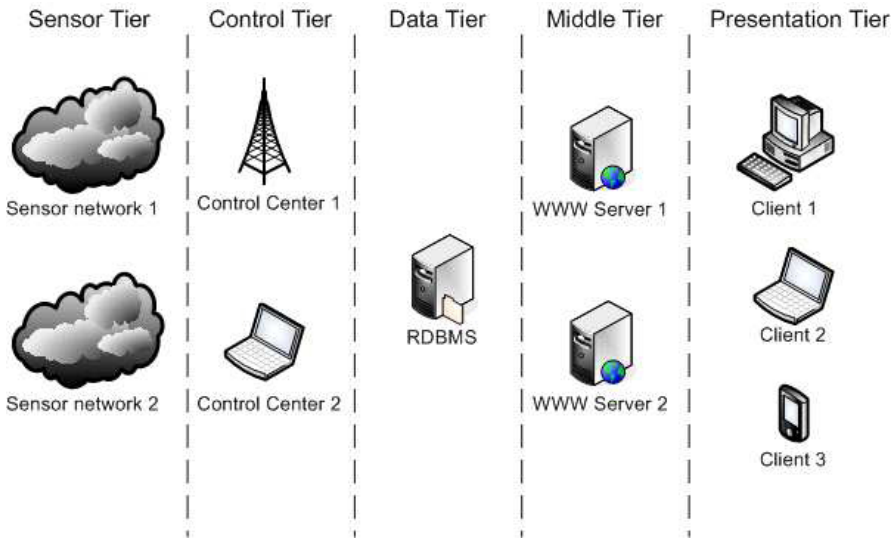
2.1 jWebDust

Το jWebDust είναι ένα γενικό και ευέλικτο περιβάλλον ανάπτυξης εφαρμογών και απομακρυσμένης διαχείρισης για ασύρματα δίκτυα μικροαισθητήρων υλοποιημένο σε Java και TinyOS. Παρέχει τα κατάλληλα μέσα στον προγραμματιστή μιας εφαρμογής ώστε να δημιουργήσει ένα παραμετροποιημένο περιβάλλον που θα παρέχει ένα ευρύ φάσμα υπηρεσιών για ασύρματα δίκτυα μικροαισθητήρων. Πιο συγκεκριμένα, το jWebDust αποθηκεύει την πληροφορία που καταγράφεται από τους αισθητήρες σε μία βάση δεδομένων, δίνει τη δυνατότητα καταγραφής πραγματοποιήσιμων στατιστικών και παρέχει ένα σύνολο διεπαφών βασισμένων στο web, που επιτρέπουν στο σχεδιαστή να παρουσιάσει τα καταγεγραμμένα δεδομένα στο χρήστη ανάλογα με τις ανάγκες της

εφαρμογής. Επιπλέον, το jWebDust βασίζεται σε μία ευέλικτη αρχιτεκτονική πολλαπλών επιπέδων (multitier) που αφαιρετικοποιεί τις λεπτομέρειες του πραγματικού δικτύου, της αποθήκευσης και ανάκτησης των μετρήσεων και της παρουσίασης. Χάρη σε αυτή την ευελιξία το jWebDust μπορεί να χειρίζεται αυτόματα την προσθήκη νέων συσκευών με διαφορετικά χαρακτηριστικά σε ένα δίκτυο μικροαισθητήρων, υποστηρίζει την ασύνδετη λειτουργία του συστήματος και επιτρέπει την ταυτόχρονη χρήση από μεγάλο αριθμό χρηστών. Ας δούμε αναλυτικότερα παρακάτω την αρχιτεκτονική και τα κύρια χαρακτηριστικλα του jWebDust συστήματος.

2.1.1 Αρχιτεκτονική

Το jWebDust, όπως αναφέρθηκε και παραπάνω, έχει σχεδιαστεί σε μια αρχιτεκτονική βασισμένη σε components με πολλούς και ποικίλους στόχους σχεδιασμού όπως η εστίαση στην αυτονομία, την αξιοπιστία και τη διαθεσιμότητα. Σε υψηλό επίπεδο, τα στοιχεία του jWebDust είναι οργανωμένα, χρησιμοποιώντας το N-tier μοντέλο εφαρμογής, ως εξής: (i) Το Sensor επίπεδο, που αποτελείται από ένα ή περισσότερα ασύρματα δίκτυα αισθητήρων ανεπτυγμένα σε περιοχές ενδιαφέροντος, (ii) το Control επίπεδο που αντιστοιχεί στη βαθμίδα των κέντρων ελέγχου όπου τα ασύρματα δίκτυα αισθητήρων κάνουν αναφορά στην πραγματοποίηση των γεγονότων, (iii) το Data επίπεδο υπεύθυνο για την αποθήκευση των πληροφοριών που προέρχονται από το ασύρματο δίκτυο(-υα) αισθητήρων, (iv) το Middle επίπεδο που είναι υπεύθυνο για την επεξεργασία των δεδομένων για τη διμιουργία στατιστικών και άλλων χρήσιμων πληροφοριών και (v) το Presentation επίπεδο το οποίο διασυνδέει τις πληροφορίες με τον εξωτερικό χρήστη με έναν εύκολο τρόπο με βάση τις δυνατότητες του συστήματός του. Οι πέντε βαθμίδες που απαρτίζουν το jWebDust σύστημα παρουσιάζονται εδώ (Βλέπε Σχήμα 2.1).



Σχήμα 2.1: Αρχιτεκτονική 5-επιπέδων

Το Sensor επίπεδο. Συνήθως, το επίπεδο αυτό αποτελεί το θεμέλιο της κάθε εφαρμογής για ασύρματα δίκτυα αισθητήρων. Οι κόμβοι αισθητήρων είναι συνήθως διάσπαρτοι στην περιοχή ενδιαφέροντος και αποτελούν ένα ή περισσότερα δίκτυα αισθητήρων. Κάθε ένας από αυτούς τους διάσπαρτους κόμβους έχει την ικανότητα να συλλέγει στοιχεία και να δρομολογεί τα δεδομένα πίσω στο Κέντρο Ελέγχου και τους τελικούς χρήστες. Τα δεδομένα διοχετεύονται στο Κέντρο Ελέγχου από μια multi-hop αρχιτεκτονική και στη συνέχεια το Κέντρο Ελέγχου επικοινωνεί με τις άλλες βαθμίδες του συστήματος ενδεχομένως μέσω Internet. Κάθε κόμβος θα εκτελέσει λογισμικό που βασίζεται στο λειτουργικό σύστημα TinyOS, χρησιμοποιεί ένα πρωτόκολλο ανακάλυψης κόμβου για την αναφορά των αρακτηριστικών των αισθητήρων και της επεξεργασίας στο Κέντρο Ελέγχου και ένα πρωτόκολλο διάδοσης ερωτημάτων(queries) για τη διανομή ερωτημάτων στο δίκτυο αισθητήρων και τη διάδοση των στοιχείων που ταιριάζουν σε αυτά τα ερωτήματα πίσω στο Κέντρο Ελέγχου.

Το Control επίπεδο. Το Control επίπεδο αποτελείται από τα κέντρα ελέγχου κάθε δικτύου αισθητήρων. Τα κέντρα ελέγχου είναι υπεύθυνα για τη συγκέντρωση όλων των μετρήσεων που προέρχονται από τα δίκτυα αισθητήρων καθώς και τη διαβίβαση των ερωτημάτων από το data επίπεδο στους κόμβους αισθητήρων. Με άλλα λόγια, λειτουργούν ως πύλες μεταξύ των επιπέδων Sensor και Data. Σε επίπεδο hardware, ένα τυπικό κέντρο ελέγχου αποτελείται από ένα μικρό κόμβο που είναι συνδεδεμένος με ένα επιτραπέζιο PC ή laptop έχει σύνδεση δικτύου με ένα server βάσης δεδομένων. Ο κόμβος ο οποίος αντιστοιχεί στο Control επίπεδο είναι απαραίτητο να επικοινωνεί με το δίκτυο αισθητήρων. Εναλλακτικά, μπορεί να χρησιμοποιηθεί μια ενσωματωμένη πλατφόρμα. Ένα σημαντικό χαρακτηριστικό αυτού του επιπέδου είναι *η ικανότητα να λειτουργούν ακόμα και όταν δεν υπάρχει σύνδεση με το Data επίπεδο για συνεχή χρονική περίοδο*. Η σημασία αυτού του χαρακτηριστικού περιγράφεται από το γεγονός ότι τα κέντρα ελέγχου, από μόνα τους, μπορούν να έχουν ασύρματη σύνδεση με το Data επίπεδο κατά την οποία, η αδιάλειπτη επικοινωνία δεν είναι εγγυημένη. Κατά τη διάρκεια μιας τέτοιας περιόδου, οποιαδήποτε νέα ερωτήματα(*queries*) που διατυπώνονται δεν θα διαβιβάζονται στο δίκτυο αισθητήρων (από τη στιγμή που το κέντρο ελέγχου δεν μπορεί να ενημερώνεται για αυτά τα ερωτήματα), ενώ οι μετρήσεις που λαμβάνονται από αισθητήρες δεν θα σταλούν στο Data επίπεδο, αλλά θα αποθηκεύονται σε τοπικό επίπεδο. Μόλις το κέντρο ελέγχου δημιουργεί μια σύνδεση με το Data επίπεδο, όλες οι τοπικά αποθηκευμένες μετρήσεις προοθούνται στο Data επίπεδο και νέα ερωτήματα διαβιβάζονται στο δίκτυο.

Το Data επίπεδο.Τα συστατικά στοιχεία σε αυτό το επίπεδο βασίζονται στο σύστημα της σχεσιακής βάσης δεδομένων και στη λειτουργικότητα και τις μεθόδους που σχετίζονται με τις υπηρεσίες που απαιτούνται από τα Control και Middle επίπεδα. Η βάση

δεδομένων του συστήματος αποτελείται από πίνακες που μπορούν να οργανωθούν σε τρεις κατηγορίες:

i *Mote related* πίνακες. Οι πληροφορίες που σχετίζονται με τα hardware χαρακτηριστικά των κόμβων που συνθέτουν τα ασύρματα δίκτυα αισθητήρων είναι οργανωμένες σε αυτή την κατηγορία. Η βάση δεδομένων συστήματος υποστηρίζει ετερογενή δίκτυα αισθητήρων, δηλαδή δίκτυα που αποτελούνται από διαφορετικά είδη motes (π.χ. Telos, MicaZ motes), και που μπορεί να έχουν διαφορετικά είδη αισθητήρων που συνδέονται με αυτά (π.χ. φως, υγρασία κ.λπ.).

ii *Query related* πίνακες. Αυτή η ομάδα κατέχει πίνακες με πληροφορίες που αφορούν στα ερωτήματα που γίνονται στο δίκτυο που σχετίζονται με πολλούς αισθητήρες ή /και πολλαπλών τύπων αισθητήρες.

iii *Sensor related* πίνακαw. Όλες οι πληροφορίες που ελήφθησαν από τα δίκτυα αισθητήρων που ταιριάζουν σε ένα συγκεκριμένο ερώτημα(*query*) είναι αποθηκευμένα σε αυτόν τον πίνακα. Κάθε εγγραφή αντιστοιχεί σε μέτρηση που προέρχονται από ένα συγκεκριμένο κόμβο και ενός απλού αισθητήρα.

Το Middle επίπεδο.Το Middle επίπεδο αποτελείται από όλα τα συστατικά στοιχεία που συνθέτουν την jWebDust λογική και είναι υπεύθυνα για την μεταφορά δομών και δεδομένων στο Presentation επίπεδο. Τα συστατικά στοιχεία αυτού του επιπέδου μπορεί να θεωρηθούν ως εφαρμογές που τρέχουν σε ένα server χωρίς πρόσωπο (επίσης αναφερόμενα και ως servlets). Αυτά τα στοιχεία έχουν αυτοτελή λειτουργικότητα που εκτελούνται ανεξάρτητα, προκειμένου να επεξεργαστούν συγκεκριμένες δομές δεδομένων και να παράγουν κάποια έξοδο. Σε αυτό το βαθμό, τα στοιχεία αυτά, συνεισφέρου σε μια απλούστερη διανομή του φόρτου εργασίας και επιτρέπουν την εύκολη ανάπτυξη των υπηρεσιών που σχετίζονται με το Presentation επίπεδο. Στο jWebDust τα στοιχεία των Presentation και Middle επιπέδων μπορούν να λειτουργούν ταυτόχρονα και ανεξάρτητα. Τα στοιχεία έχουν διατυπωθεί με βάση τις διαθέσιμες δομές δεδομένων και τις λειτουργίες που απαιτούνται από κάθε φορέα. Για παράδειγμα, η διαχείριση των ερωτημάτων(*queries*) χειρίζεται από ένα μόνο συστατικό στοιχείο και οι μέθοδοι που έχουν αναπτυχθεί εκτελούν λειτουργίες, όπως createQuery, deleteQuery, updateQuery, κλπ. Όλες αυτές οι μέθοδοι δέχονται παραμέτρους που αποστέλονται απο το Presentation επίπεδο. Η λογική της δημιουργίας ενός νέου ερωτήματος (δηλαδή τον έλεγχο του αν ένα παρόμοιο ερώτημα υπάρχει ήδη ή αν ένασ κόμβος παρέχει τους αισθητήρες που ορίζονται στο ερώτημα) είναι τυποποιημένη σε ένα μόνο συστατικό στοιχείο.

Το Presentation επίπεδο.Το Presentation επίπεδο είναι, βασικά, η διεπαφή με το χρήστη. Είναι το πλέον διαθέσιμο επίπεδο στους τελικούς χρήστες της εφαρμογής και είναι

υπεύθυνο για την συλλογή των δεδομένων εισόδου και την παρουσίαση των πληροφοριών που συλλέγονται από το δίκτυο αισθητήρων. Οι όροι thin client και rich client χρησιμοποιούνται για την περιγραφή των δυνατοτήτων του Presentation επιπέδου που εξαρτάται από τους κάθε φορά διαθέσιμους πόρους. Χαρακτηρίζονται ως rich client τα στοιχεία του επιπέδου που είναι πλούσια σε λειτουργικότητα. Η υποστήριξη των thin client διασφαλίζει τη διαλειτουργικότητα του συστήματος, μέσω των διαφόρων διαθέσιμων αρχιτεκτονικών συστημάτων. Οι τελικοί χρήστες έχουν τη δυνατότητα να επιλέξουν μεταξύ των διαθέσιμων λύσεων προκειμένου να μεγιστοποιήσουν τους διαθέσιμους πόρους. Η ανοιχτή αρχιτεκτονική του jWebDust σύστηματος επιτρέπει σε νέα στοιχεία που θα εισαχθούν σε μεταγενέστερα στάδια να αντιμετωπίσουν τέτοιου είδους θέματα.

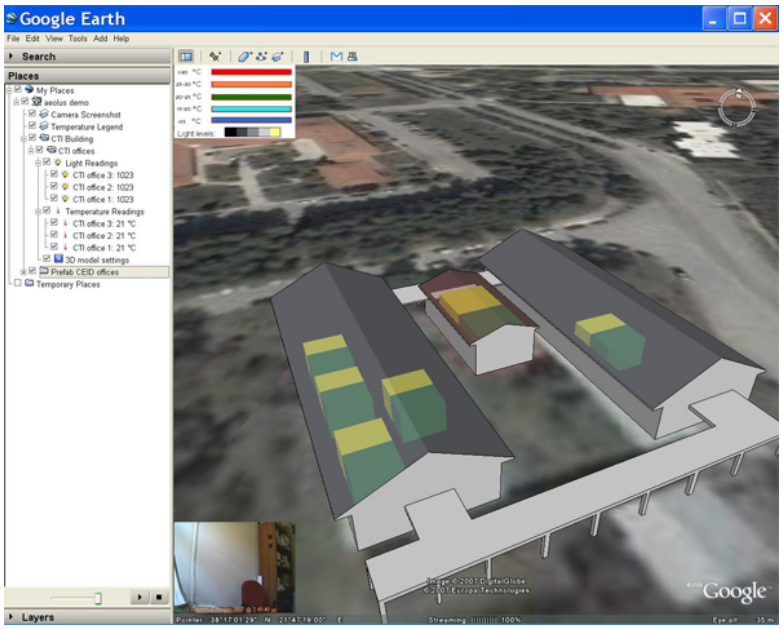
2.1.2 Χαρακτηριστικά

Παραπάνω παρουσιάσαμε μία αναλυτική περιγραφή του jWebDust συστήματος. Ας δούμε εδώ ποια είναι τα διάφορα χαρακτηριστικά και οι λειτουργίες που προσφέρει. Οι κύριες δυνατότητες του jWebDust είναι:

1. Παρέχει ένα περιβάλλον για τυποποίηση και διαχείριση πλήθους πρωτοκόλλων χαμηλού επιπέδου με *ελάχιστη αναπτυξιακή και διαχειριστική προσπάθεια*.
2. Επιτρέπει την ανάπτυξη νέων λειτουργιών που μπορούν να εύκολα να ενσωματωθούν στην υπόλοιπη αρχιτεκτονική και τα επίπεδα της ιεραρχίας έτσι ώστε να ταιριάζει καλύτερα στις ανάγκες των εφαρμογών.
3. Υποστηρίζει πολλαπλά δίκτυα αισθητήρων(π.χ. φυσικά διαχωρισμένα) με πολλαπλά/ξεχωριστά κέντρα ελέγχου και επιτρέπει τον έλεγχο τους σαν απλό ιδεατό δίκτυο αισθητήρων.
4. Χειρίζεται δίκτυα αισθητήρων που αποτελούνται από συσκευές με *ετερογενή* χαρακτηριστικά. Τα δίκτυα αισθητήρων στιν πραγματικό κόσμο είναι σπάνια ομογενή.
5. Υποστηρίζει Αποσυνδεδεμένα/Κινητά Κέντρα Ελέγχου με το να χειρίζεται μεγάλες χρονικά αποσυνδέσεις των κέντρων ελέγχου, και των δικτύων που συνδέονται σε αυτά, και έτσι δεν έχουμε απόλειες πληροφορίας
6. Πολλοί χρήστες μπορούν παράλληλα να παρακολουθήσουν, θέσουν ερωτήματα, οπτικοποιήσουν την εκτέλεση ενός WSN μέσω ενός Web interface.

2.2 ShareSense

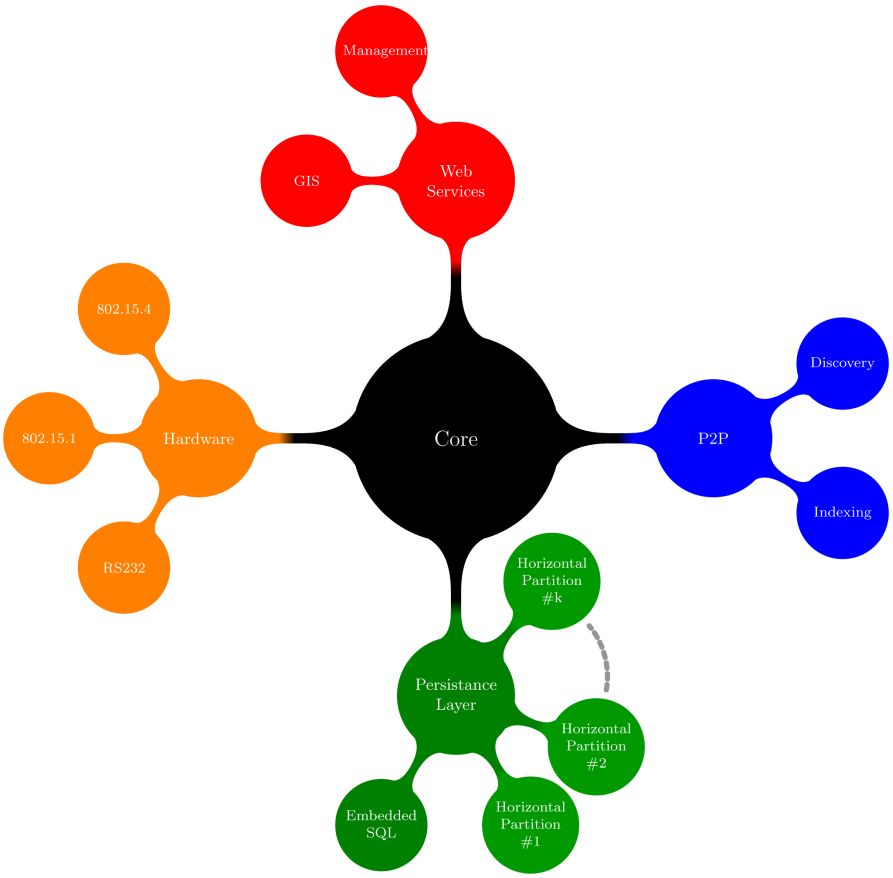
Το ShareSense είναι ένα περιβάλλον παρακολούθησης πολλαπλών δικτύων μικροαισθητήρων που βασίζεται σε μία αρχιτεκτονική ομοτίμων (peer-to-peer) για την επικοινωνία ανάμεσα σε ασύνδετα δίκτυα και χρήστες. Είναι υλοποιημένο σε Java και χρησιμοποιεί την πλατφόρμα JXTA σαν υπόβαθρο για την υλοποίηση της αρχιτεκτονικής, απαλλάσσοντας έτσι το χρήστη από προβλήματα συνδεσιμότητας και διαλειτουργικότητας. Κάθε δίκτυο μικροαισθητήρων διαχειρίζεται από μία εγκατάσταση του jWebDust και αντιπροσωπεύεται στο δίκτυο από ένα gateway ομότιμο κόμβο. Οι χρήστες συμμετέχουν στο peer-to-peer δίκτυο σαν ξεχωριστοί ομότιμοι (user peers) υποβάλλοντας ερωτήσεις και λαμβάνοντας απαντήσεις. Ειδικοί ομότιμοι κόμβοι (routing peers) χρησιμοποιούνται για τη δρομολόγηση δεδομένων και την ανεύρεση ομοτίμων. Κύριος στόχος είναι τα συμμετέχοντα δίκτυα μικροαισθητήρων να φαίνονται σαν ένας ενιαίο δίκτυο στο οποίο οι χρήστες υποβάλλουν ερωτήσεις χωρίς να χρειάζεται γνώση για την τοπολογία και αρχιτεκτονική του. Αυτό επιτυγχάνεται με τη χρήση ενός επεξεργαστή ερωτήσεων που εκτελείται σε κάθε ομότιμο κόμβο χρήστη που καταμερίζει και υποβάλλει την ερώτηση στους κατάλληλους gateway peers. Επιπλέον, το ShareSense διαθέτει και ένα ελκυστικό γραφικό περιβάλλον βασισμένο στο GoogleEarth (Βλέπε Σχήμα 2.2), που οπτικοποιεί σε πραγματικό χρόνο τις μετρήσεις που λαμβάνονται από τα συμμετέχοντα δίκτυα μικροαισθητήρων, καθώς και τον χώρο στον οποίο αναπτύσσονται. Η διαχείριση του συνολικού δικτύου καθως και η εξαγωγή πιο λεπτομερών στατιστικών γίνεται μέσω μίας web εφαρμογής.



Σχήμα 2.2: Παράδειγμα χρήσης του ShareSense με GoogleEarth

2.3 WebDust2

Το σύστημα Webdust2αποτελεί την εξέλιξη του jWebDust σε μια γενικότερη εφαρμογή, πλήρως κατανεμημένη, και ανεπτυγμένη με βάση την *component* λογική. Σκοπός της σχεδίασής του είναι ο συνδιασμός των περισσότερων συνιστωσών που αφορούν στα WSN, την αρχιτεκτονική τους, τις δυνατότητες τους, έτσι ώστε να αποτελέσει ένα πλήρες σύστημα διαχείρισης, ελέγχου και επεξεργασίας ασυρμάτων δικτύων αισθητήρων. Το σύστημα είναι υπό συνεχή ανάπτυξη και εξέλιξη μιας και οι περιοχές που καλύπτει είναι πολλές και διάφορες(θέματα δρομολόγησης σε WSN, ασφάλεια, μετάδοση δεδομένων, διαχείρηση, κλπ.). Ένα γενικό πλάνο του συστήματος φαίνεται στο παρακάτω σχήμα(Βλέπε Σχήμα 2.3).



Σχήμα 2.3: Γενικό πλάνο του WebDust2 συστήματος

Εδώ θα κάνουμε μια γενική περιγραφή του Webdust2. Το σύστημα αποτελείται απο ένα βασικό πυρήνα (*Core*) ο οποίος είναι είναι υπεύθυνος για την υποστήριξη όλων των προσαρτημένων σε αυτόν τμημάτων. Όλα τα τμήματα-κομμάτια του συστήματος επικοινωνούν και συνδέονται με αυτόν. Το τμήμα του υλικού (*Hardware*) αφορά στο υλικό αυτό καθεαυτό με ότι αυτό συνεπάγεται. Για παράδειγμα, το υλικό μπορεί να περιλαμβάνει αισθητήρες, κόμβους αισθητήρων, πύλες (*gateways*), υλικό ασύρματης μετάδοσης, κοκ. Το *Persistense* τμήμα-στρώμα είναι η διασύνδεση των δεδομένων με τον πυρήνα και

η παροχή κλάσεων και μεθόδων που βοηθούν στον πιο ασφαλή χειρισμό των δεδομένων αλλά και γενικότερα σε αφαίρεση ως προς τον χειρισμό ερωτημάτων που αφορούν τη βάση δεδομένων. Το όλο σύστημα μπορεί να είναι συνδεδεμένο σε *P2P* δίκτυο στο, και απο το, οποίο μπορεί να στέλνει και να λαμβάνει δεδομένα που έχουν σχέση ενδεχομένως με άλλα παρόμια δίκτυα ή απλά με ερωτήματα που αφορούν στο ίδιο το σύστημα επιτρέποντας του, έτσι, να είναι πλήρως κατανεμημένο. Τέλος, υπάρχει και το τμήμα των *Web Services* το οποίο αποτελεί έδαφος για ανάπτυξη πλήθους εφαρμογών που έχουν σχέση με τις υπηρεσίες τις οποίες μπορεί να προσφέρει το σύστημα στους χρήστες WSN. Το Webdust2ενσωματώνει όλες τις δυνατότητες και τα πλεονεκτήματα των jWebDust και ShareSense συστημάτων και λόγω της *component*-κεντρικής φιλοσοφίας του είναι πλήρως επεκτάσιμο και επιτυγχάνει μεγάλο βαθμό ετερογένειας.

Η περιγραφή του όλου συστήματος υπήρξε σύντομη διότι η παρούσα εργασία ασχολείται με την ανάπτυξη τμημάτων-μερών του και όχι με την κάλυψη όλων των εκδοχών του. Έτσι, σε αυτή την εργασία, θα παρουσιάσουμε το σύστημα μας με την παραδοχή-θεώρηση πως το σύστημα ακολουθεί την αρχιτεκτονική 3-επιπέδων όπως αυτή παρουσιάστηκε σε παραπάνω εδάφιο(Βλέπε Παράγραφος 1.5.2). Αναλυτική περιγραφή ακολουθεί στα επόμενα κεφάλαια.

Κεφάλαιο 3

Επίπεδο Δεδομένων

Ξεκινώντας την πιο αναλυτική περιγραφή του συστήματος, σε αυτό το κεφάλαιο θα παρουσιάσουμε όλα τα τμήματα εκείνα τα οποία σχετίζονται με το βασικό επίπεδο, το επίπεδο των δεδομένων. Οπως είναι γνωστό, τα δεδομένα, η αποθήκευση τους, η ασφάλεια και η ευστάθεια παίζουν μείζον ρόλο στην ανάπτυξη ενός συστήματος WSN. Ένας καλός σχεδιασμός είναι απαραίτητος καθώς με το επίπεδο αυτό σχετίζονται τα δεδομένα, η βάση, καθώς και η περισσότερη ροή πληροφορίας του συστήματος. Αρχικά θα περιγράψουμε τη βάση δεδομένων η οποία χρησιμοποιήθηκε για την ανάπτυξη του συστήματος, και αργότερα την πιο αναλυτική περιγραφή των πινάκων και των σχέσεων που το διέπουν.

3.1 HSQLDB

Η HSQLDB είναι μια SQL σχεσιακή βάση δεδομένων με πυρήνα γραμμένο σε Java. Έχει ένα οδηγό JDBC και υποστηρίζει ένα πλούσιο υποσύνολο της ANSI SQL-92 (φορμάτ BNF δέντρου) και επιπλέον τις αναβαθμίσεις της SQL 99 και 2003. Διαθέτει ένα μικρό (λιγότερο από 100k σε μία έκδοση για applets), γρήγορο μηχανισμό βάσης δεδομένων ο οποίος προσφέρει παράλληλα πίνακες τόσο σε μνήμη όσο και σε δίσκο και υποστηρίζει λειτουργίες και ως server και ως ενσωματωμένη σε κάποιο σύστημα. Επιπλέον, περιλαμβάνει εργαλεία, όπως ένα ελάχιστο web server και εργαλεία ερωτημάτων(*queries*) και διαχείρισης που τρέχουν στη μνήμη (μπορεί να τρέχουν σαν applets).

Η HSQLDB χρησιμοποιείται σαν βάση δεδομένων και σαν persistence(Βλέπε Κεφάλαιο 4) μηχανή σε πολλές εφαρμογές λογισμικού ανοικτού κώδικα και ακόμη και σε πολλές εμπορικές εφαρμογές και προϊόντα. Είναι πολύ σταθερή και αξιόπιστη. Είναι γνωστή για το μικρό της μέγεθος, την ικανότητα να εκτελείται εξ' ολοκλήρου στη μνήμη,

την ευελιξία της και την ταχύτητά της. Όλα τα προηγούμενα χαρακτηριστικά, καθώς και το γεγονός οτι το πακέτο λογισμικού της είναι εντελώς ελεύθερο για χρήση και διανομή οδήγησαν στην επιλογή της συγκεκριμένης βάσης δεδομένων ως κατάλληλη για το σύστημά μας.

3.2 Σχεδιασμός Βάσης Δεδομένων

Η ραχοκοκκαλιά της εφαρμογής είναι η βάση δεδομένων εφόσον αποτελεί και τη βάση για να αναπτυχθούν όλες οι υπυρεσίες των παραπάνω επιπέδων. Το μοντέλο που χρησιμοποιούμε, δεν θα μπορούσε να ήταν άλλο από το μοντέλο Σχέσεων-Οντοτήτων. Στη συνέχεια ακολουθεί η περιγραφή αναλυτικά.

Οι πίνακες είναι κατηγοριοποιημένοι με βάση τις υπηρεσίες που προσφέρουν στα υψηλότερα επίπεδα. Έτσι έχουμε:

General – Πίνακες που κρατούν γενικές πληροφορίες.

Hardware – Πίνακες που κρατούν πληροφορίες που αφορούν τα τεχνικά χαρακτηριστικά των αισθητήρων.

Readings – Πίνακας που κρατά πληροφορίες για τα δεδομένα που λήφθηκαν από το WSN.

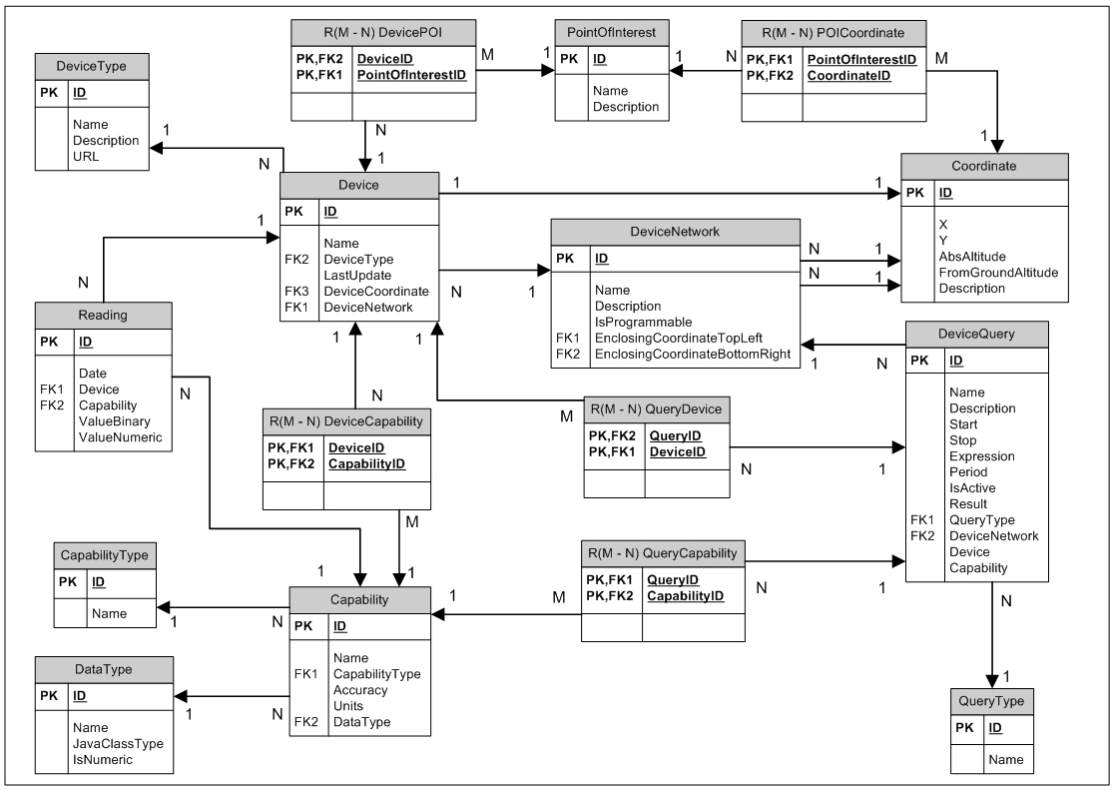
Queries – Πίνακες που κρατούν πληροφορίες για ενεργά query ή για query που έχουν γίνει στο παρελθόν.

Το Διάγραμμα Σχέσεων-Οντοτήτων(Βλέπε Σχήμα 3.1) του συστήματος αποτελείται από 11 πίνακες και 5 πίνακες σχέσεων N-M. Είναι μια πρώτη προσέγγιση για το σύστημα Webdust2αλλά είναι σχεδιασμένο έτσι ώστε να επιδέχεται μελλοντικές επεκτάσεις και προσθήκες.

Σημείωση: *Για λόγους απλότητας, η στήλη ID σε κάθε πίνακα στη βάση δεδομένων αντιστοιχεί στο όνομα του πίνακα με τη λέξη "ID" επιπροσθέτως. Για παράδειγμα για τον πίνακα Device η στήλη ID αντιστοιχεί στην Device_ID.*

Σύμφωνα με τους πίνακες που ακολουθούν, οι οποίοι περιγράφουν τα περιεχόμενα των δομών που ανήκουν στο επίπεδο των δεδομένων, έχουμε τις εξής συντομογραφίες:

Field – Το όνομα του πεδίου



Σχήμα 3.1: Διάγραμμα Σχέσεων-Οντοτήτων του Webdust2συστήματος

Type – Ο τύπος δεδομένων του πεδίου. Αποδεκτές τιμές: *I – integer, D – decimal, T – text, B - 1-bit, D - date/time, C – character*. Αν ο τύπος είναι *character*, το μέγιστο επιτρεπτό μήκος δίνεται επίσης(e.g. *C, 50*).

R – Το πεδίο είναι υποχρεωτικό. Πιθανές τιμές: *Y - ναι, N - όχι*.

U – Το πεδίο πρέπει να έχει μοναδική τιμή. Πιθανές τιμές: *Y - ναι, N - όχι*.

Id – Η τιμή του πεδίου είναι αυτόματα αυξανόμενη. Πιθανές τιμές: *Y - ναι, N - όχι*.

K – Το πεδίο είναι κλειδί. Πιθανές τιμές: *P - primary key, S - secondary key, F - foreign key*.

Description – Σύντομη περιγραφή του πεδίου.

3.2.1 General Πίνακες

Εδώ περιγράφουμε όλους τους πίνακες που κρατούν γενικές πληροφορίες του Webdust2συστήματος. Ονομαστικά οι πίνακες αντιπροσωπεύουν τα εξής:

DataType – Τύπος δεδομένων της μέτρησης.

Coordinate – Συντεταγμένες των αισθητήρων.

DataType Πινακας. Ο πίνακας αυτός αντιπροσωπεύει το είδος των δεδομένων που προκύπτουν από μια συγκεκριμένη ικανότητα μέτρησης κάποιου αισθητήρα. Για παράδειγμα, οι αισθητήρες θερμοκρασίας δημιουργούν τύπους δεδομένων(double), οι αισθητήρες εικόνας παράγουν τύπους ροής δεδομένων(stream).

Field	Type	R	U	Id	K	Description
ID	I	Y	Y	Y	P	Ταυτότητα του DataType
Name	C, 40	Y	N	N		Όνομα του DataType
JavaClassType	C, 40	N	N	N		Ο *Java* τύπος δεδομένων για το DataType
isNumeric	B	N	N	N		Το συγκεκριμένο DataType είναι αριθμήσιμο ή όχι

Πίνακας 3.1: Περιγραφή του πίνακα DataType

Αυτός ο πίνακας έχει σχέση με τους παρακάτω πίνακες:

1. Capability(1-N)

Coordinate Πινακας. Ο πίνακας αυτός αντιπροσωπεύει όλες τις συντεταγμένες και τις ιδιότητες που βοηθούν στον εντοπισμό των συσκευών. Η ακριβής τοποθεσία μπορεί εύκολα να βρεθεί λόγω των ιδιοτήτων όπως μήκος, πλάτος, υψόμετρο, κ.λ.π.

Field	Type	R	U	Id	K	Description
ID	I	Y	Y	Y	P	Ταυτότητα του Coordinate
X	D	Y	N	N		Γεωγραφικό μήκος (X θέση)
Y	D	Y	N	N		Γεωγραφικό πλάτος (Y θέση)
AbsAltitude	D	N	N	N		Απόλυτο ύψος (από την επιφάνεια της θάλασσας)
FromGroundAltitude	D	N	N	N		Ύψος (από το έδαφος)
Description	T	N	N	N		Μια περιγραφή

Πίνακας 3.2: Περιγραφή του πίνακα Coordinate

Αυτός ο πίνακας έχει σχέση με τους παρακάτω πίνακες:

1. Device(1-1)
2. PointOfInterest(M-N - μέσω του πίνακα-σχέσης POICoordinate)

3.2.2 Hardware Πίνακες

Εδώ περιγράφουμε πίνακες που κρατούν πληροφορίες σχετικά με το υλικό του Webdust2συστήματος. Ονομαστικά οι πίνακες αντιπροσωπεύουν τα εξής:

Device – Η συσκευή-κόμβος.

DeviceType – Τύπος της συσκευής.

Capability – Ο αισθητήρας.

CapabilityType – Τύπος του αισθητήρα.

Device Πίνακας. Ο πίνακας αυτός αντιπροσωπεύει τη συσκευή (κόμβο). Ο πίνακας Device περιέχει πληροφορίες για τη συσκευή όπως κάποιες ιδιότητές του, τη θέση του κ.α. Για παράδειγμα, πότε η συσκευή ενημερώθηκε τελευταία, σε ποιο δίκτυο ανήκει ή ποιες είναι οι συντεταγμένες που βρίσκεται η συσκευή.

Field	Type	R	U	Id	K	Description
ID	I	Y	Y	Y	P	Ταυτότητα του Device
DeviceType	I	Y	N	N	F	Τύπος του Device
DeviceNetwork	I	Y	N	N	F	Δίκτυο που ανήκει το Device
Name	C, 40	Y	N	N		Όνομα του Device
LastUpdate	D	N	Y	N		Τελευταία ενημέρωση του Device
DeviceCoordinate	I	Y	Y	N	F	Συντεταγμένες του Device

Πίνακας 3.3: Περιγραφή του πίνακα Device

Αυτός ο πίνακας έχει σχέση με τους παρακάτω πίνακες:

1. Capability(N-M - μέσω του πίνακα-σχέσης DeviceCapability)
2. PointOfInterest(N-M - μέσω του πίνακα-σχέσης DevicePOI)
3. Query(M-N - μέσω του πίνακα-σχέσης QueryDevice)
4. DeviceType(N-1)
5. DeviceNetwork(N-1)
6. Coordinate(1-1)

DeviceType Πίνακας. Ο πίνακας αυτός αντιπροσωπεύει τον τύπο της συσκευής. Ο DeviceType πίνακας περιέχει ενδείξεις σχετικά με τους τύπους των κόμβων-αισθητήρων του δικτύου (Μίκα, Mica2, κλπ.). Είναι σημαντικό να κάνουμε τέτοιες διακρίσεις, δεδομένου ότι κάθε είδος συσκευής(κομβος) έχει διαφορετικά χαρακτηριστικά και δυνατότητές και θέλουμε να έχουμε μια σαφή εικόνα του τι υπάρχει στο δικτύο. Για παράδειγμα, μια απλή συσκευή μπορεί να έχει Name = ”MBR410CB”, Description = ”This is a mica2 mote”, κλπ.

Αυτός ο πίνακας έχει σχέση με τους παρακάτω πίνακες:

Field	Type	R	U	Id	K	Description
ID	I	Y	Y	Y	P	Ταυτότητα του DeviceType
Name	C, 40	Y	N	N		Όνομα του DeviceType
Description	T	N	Y	N		Περιγραφή του DeviceType
URL	T	N	Y	Y		URL περισσότερες πληροφορίες για αυτό το DeviceType

Πίνακας 3.4: Περιγραφή του πίνακα DeviceType

1. Device(1-N)

Capability Πίνακας. Ο πίνακας αυτός αντιπροσωπεύει τις ικανότητες που έχουν οι αισθητήρες. Οι δυνατότητες αυτές ποικίλουν στο είδος, την ακρίβεια ή τις μονάδες. Για παράδειγμα, ορισμένα είδη μπορούν να είναι θερμοκρασία, υγρασία, κ.λπ.

Field	Type	R	U	Id	K	Description
ID	I	Y	Y	Y	P	Ταυτότητα του Capability
CapabilityType	I	Y	N	N	F	Type of the Capability
Name	C, 40	Y	N	N		Όνομα του Capability
Accuracy	D	N	N	N		Ακρίβεια του Capability(η μέγιστη απόκλιση)
Units	I	N	N	N		Αριθμός μονάδων του Capability
DataType	I	Y	N	N	F	Τύπος δεδομένων του Capability

Πίνακας 3.5: Περιγραφή του πίνακα Capability

Αυτός ο πίνακας έχει σχέση με τους παρακάτω πίνακες:

1. Device(M-N - μέσω του πίνακα-σχέσης DeviceCapability)
2. Query(M-N - μέσω του πίνακα-σχέσης QueryCapability)
3. CapabilityType(N-1)
4. DataType(N-1)

CapabilityType Πίνακας. Ο πίνακας αυτός αντιπροσωπεύει τον τύπο των δυνατοτήτων ενός αισθητήρα. Εδώ παρατίθενται όλα τα είδη των δυνατοτήτων. Για παράδειγμα θερμοκρασία, υγρασία, κ.λπ.

Αυτός ο πίνακας έχει σχέση με τους παρακάτω πίνακες:

1. Capability(1-N)

Field	Type	R	U	Id	K	Description
ID	I	Y	Y	Y	P	Ταυτότητα του CapabilityType
Name	C, 40	Y	N	N		Όνομα του CapabilityType

Πίνακας 3.6: Περιγραφή του πίνακα CapabilityType

3.2.3 Network Πίνακες

Εδώ περιγράφουμε πίνακες που κρατούν πληροφορίες σχετικά με το δίκτυο του συστήματος Webdust2. Ονομαστικά οι πίνακες αντιπροσωπεύουν τα εξής:

DeviceNetwork – Το δίκτυο του συστήματος.

PointOfInterest – Περιοχή ενδιαφέροντος.

DeviceNetwork Πίνακας. Ο πίνακας αυτός αντιπροσωπεύει το δίκτυο των συσκευών. Ο DeviceNetwork πίνακας περιέχει εγγραφές για όλα τα δίκτυα αισθητήρων του Webdust2. Ο πίνακας αυτός χρησιμοποιείται για να γίνεται ευκολότερα η διάκριση μεταξύ των κόμβων που ανήκουν σε διαφορετικά δίκτυα αισθητήρων. Εδώ αποθηκεύονται πληροφορίες του δικτύου όπως, το όνομα του δικτύου ή η περιοχή όπου βρίσκεται το δίκτυο.

Field	Type	R	U	Id	K	Description
ID	I	Y	Y	Y	P	Ταυτότητα του DeviceNetwork
Name	C, 40	Y	N	N		Όνομα του DeviceNetwork
Description	T	N	N	N		Περιγραφή του DeviceNetwork
EnclosingCoordinateTopLeft	D	Y	N	N		Πάνω αριστερά συντεταγμένη του DeviceNetwork
EnclosingCoordinateBottomRight	D	Y	N	N		Κάτω δεξιά συντεταγμένη του DeviceNetwork

Πίνακας 3.7: Περιγραφή του πίνακα DeviceNetwork

Αυτός ο πίνακας έχει σχέση με τους παρακάτω πίνακες:

1. Device(1-N)
2. Query(1-N)

PointOfInterest Πίνακας. Ο πίνακας αυτός αντιπροσωπεύει μια περιοχή ενδιαφέροντος. Ο χρήστης του Webdust2συστήματος μπορεί να επιλέξει μια περιοχή στην οποία

Field	Type	R	U	Id	K	Description
ID	I	Y	Y	Y	P	Ταυτότητα του PointOfInterest
Name	C, 40	Y	N	N		Όνομα του PointOfInterest
Description	T	N	N	N		Περιγραφή του PointOfInterest

Πίνακας 3.8: Περιγραφή του πίνακα PointOfInterest

θέλει να ελέγξει μετρήσεις από τους αισθητήρες ή γενικότερα να παρατηρήσει την τοπολογία του δικτύου.

Αυτός ο πίνακας έχει σχέση με τους παρακάτω πίνακες:

1. POICoordinate(1-M)
2. DevicePOI(1-N)

3.2.4 Reading Πίνακες

Εδώ περιγράφουμε πίνακες που κρατούν πληροφορίες σχετικά με τις μετρήσεις που καταγράφονται. Ονομαστικά οι πίνακες αντιπροσωπεύουν τα εξής:

Reading – Μέτρηση αισθητήρα.

Reading Πίνακας. Ο πίνακας αυτός αντιπροσωπεύει τις μετρήσεις. Όλες οι πληροφορίες που λαμβάνονται από τα δίκτυα αισθητήρων που απαντούν σε ένα συγκεκριμένο ερώτημα (*query*) είναι αποθηκευμένα στον πίνακα Reading. Κάθε εγγραφή αντιπροσωπεύει μια μέτρηση που προέρχεται από μια συγκεκριμένη συσκευή στο δίκτυο και ένα μόνο αισθητήρα. Στον πίνακα αυτό πρέπει να παρουσιάζονται όλες οι πληροφορίες για τη συγκεκριμένη μέτρηση. Για παράδειγμα, πότε λήφθηκε η μέτρηση, από ποια συσκευή, ποιο είναι το είδος της μέτρησης ή τι τύπου είναι η τιμή της ανάγνωσης.

Αυτός ο πίνακας έχει σχέση με τους παρακάτω πίνακες:

1. Device(N-1)
2. Capability(N-1)

Field	Type	R	U	Id	K	Description
ID	I	Y	Y	Y	P	Ταυτότητα του Reading
Device	I	Y	N	N	F	Συσκευή απο την οποία λήφθηκε το Reading
Capability	I	Y	N	N	F	Ειδος μέτρησης του Reading
Date	D	Y	Y	N		Ημ/νια που λήφθηκε το Reading
ValueBinary	D	N	N	N		Τιμή του Reading - Εδώ σαν ροή πληροφορίας (binary)
ValueNumeric	D	N	N	N		Τιμή του Reading - Εδώ σαν αριθμός (numeric)

Πίνακας 3.9: Περιγραφή του πίνακα Reading

3.2.5 Query Πίνακες

Εδώ περιγράφουμε πίνακες που κρατούν πληροφορίες σχετικά με τα ερωτήματα (*queries*) που γίνονται. Ονομαστικά οι πίνακες αντιπροσωπεύουν τα εξής:

Query – Ερώτημα (*query*).

QueryType – Τύπος ερωτήματος.

Query Πίνακας. Ο πίνακας αυτός αντιπροσωπεύει τα ερωτήματα (*queries*) που γίνονται. Περιγραφή του ερωτήματος, χρόνος που έγινε η αίτηση για το ερώτημα και χρόνος που δοθηκε η απάντηση, είναι μερικά από τα στοιχεία που αποθηκεύονται σε αυτόν τον πίνακα.

Αυτός ο πίνακας έχει σχέση με τους παρακάτω πίνακες:

1. Capability(N-M - μέσω του πίνακα-σχέσης QueryCapability)
2. Device(N-M - μέσω του πίνακα-σχέσης QueryDevice)
3. QueryType(N-1)
4. DeviceNetwork(N-1)

QueryType Πίνακας. Ο πίνακας αυτός αντιπροσωπεύει τα ερωτήματα (*queries*) που γίνονται. Περιγραφή του ερωτήματος, χρόνος που έγινε η αίτηση για το ερώτημα και χρόνος που δοθηκε η απάντηση, είναι μερικά από τα στοιχεία που αποθηκεύονται σε αυτόν τον πίνακα.

Αυτός ο πίνακας έχει σχέση με τους παρακάτω πίνακες:

1. Query(1-N)

Field	Type	R	U	Id	K	Description
ID	I	Y	Y	Y	P	Ταυτότητα του Query
Name	C, 40	Y	N	N		Όνομα του Query
Description	T	N	N	N		Περιγραφή του Query
Start	D	N	N	N		Στιγμή εκκίνησης του Query
Stop	D	N	N	N		Στιγμή λήξης του Query
Expression	I	N	N	N		Η πρόταση του Query
Period	I	N	N	N		Περίοδος του Query(msec)
IsActive	B	Y	N	N		Κατάσταση του Query
Result	T	N	N	N		Αποτέλεσμα που επιστρέφει το Query
QueryType	I	Y	N	N	FK	Τύπος του Query
DeviceNetwork	I	Y	N	N	FK	Δίκτυο που έγινε το Query
Device	I	Y	N	N	FK	Συσκευή που έγινε το Query
Capability	I	Y	N	N	FK	Δυνατότητα αισθητήρα για την οποία έγινε το Query

Πίνακας 3.10: Περιγραφή του πίνακα Query

Field	Type	R	U	Id	K	Description
ID	I	Y	Y	Y	P	Ταυτότητα του QueryType
Name	C, 40	Y	N	N		Όνομα του QueryType

Πίνακας 3.11: Περιγραφή του πίνακα QueryType

3.2.6 Πίνακες-Σχέσεις

Εδώ περιγράφουμε όλους τους πίνακες-σχέσεις του συστήματος Webdust2.

DeviceCapability Πινακας-Σχέση. Ο πίνακας αυτός αντιπροσωπεύει μια σχέση N-M και είναι στην πραγματικότητα η σχέση που συνδέει τη συσκευή και τις δυνατότητές της. Η σχέση αυτή είναι αναγκαία επειδή μια συσκευή μπορούν να έχχει περισσότερες από μία δυνατότητες και αντίστροφα μια δυνατότητα μπορεί να εμφανίζεται σε περισσότερες από μία συσκευές. Για παράδειγμα, η συσκευή με ID = 56 έχει 3 δυνατότητες, μία είναι μέτρηση της θερμοκρασίας, μια άλλη η μέτρηση της υγρασίας και η μέτρηση της φωτεινότητας. Αντίστοιχα, για παράδειγμα, η δυνατότητα που αντιπροσωπεύει τη θερμοκρασία μπορεί να εμφανιστεί σε μια συσκευή με ID = 3, ID = 4, ή και σε πολλές άλλες συσκευές.

Αυτός ο πίνακας-σχέση σχετίζεται με τους παρακάτω πίνακες:

1. Device(N-1)
2. Capability(M-1)

Field	Type	R	U	Id	K	Description
ID	I	Y	Y	Y	P	Ταυτότητα του DeviceCapability
DeviceID	I	Y	N	N	FK	Στήλη που αντιστοιχεί στο DeviceID του πίνακα Device
CapabilityID	I	Y	N	N	FK	Στήλη που αντιστοιχεί στο CapabilityID του πίνακα Capability

Πίνακας 3.12: Περιγραφή του πίνακα-σχέσης DeviceCapability

DevicePOI Πίνακας-Σχέση. Ο πίνακας αυτός αντιπροσωπεύει μια σχέση N-M και είναι στην πραγματικότητα η σχέση που συνδέει τη συσκευή και την περιοχή ενδιαφέροντος. Η σχέση αυτή είναι αναγκαία επειδή πολλές συσκευές μπορούν να υπάρχουν μέσα σε μια περιοχή ενδιαφέροντος και, αντίστοιχα, πολλές περιοχές ενδιαφέροντος μπορούν να περιέχουν μία συγκεκριμένη συσκευή.

Field	Type	R	U	Id	K	Description
ID	I	Y	Y	Y	P	Ταυτότητα του DevicePOI
DeviceID	I	Y	N	N	FK	Στήλη που αντιστοιχεί στο DeviceID του πίνακα Device
PointOfInterestID	I	Y	N	N	FK	Στήλη που αντιστοιχεί στο PointOfInterestID του πίνακα PointOfInterest

Πίνακας 3.13: Περιγραφή του πίνακα-σχέσης DevicePOI

Αυτός ο πίνακας-σχέση σχετίζεται με τους παρακάτω πίνακες:

1. Device(N-1)
2. PointOfInterest(M-1)

POICoordinate Πίνακας-Σχέση. Ο πίνακας αυτός αντιπροσωπεύει μια σχέση N-M και είναι στην πραγματικότητα η σχέση που συνδέει τις συντεταγμένες και την περιοχή ενδιαφέροντος. Η σχέση αυτή είναι αναγκαία επειδή πολλές συντεταγμένες που είναι αποθηκευμένες στη βάση δεδομένων μπορούν να υπάρχουν μέσα σε μια περιοχή ενδιαφέροντος που ορίζει ο χρήστης και, αντίστοιχα, πολλές περιοχές ενδιαφέροντος μπορούν να περιέχουν μία συγκεκριμένη συντεταγμένη.

Αυτός ο πίνακας-σχέση σχετίζεται με τους παρακάτω πίνακες:

1. Coordinate(M-1)

Field	Type	R	U	Id	K	Description
ID	I	Y	Y	Y	P	Ταυτότητα του POICoordinate
PointOfInterestID	I	Y	N	N	FK	Στήλη που αντιστοιχεί στο PointOfInterestID του πίνακα PointOfInterest
CoordinateID	I	Y	N	N	FK	Στήλη που αντιστοιχεί στο CoordinateID του πίνακα Coordinate

Πίνακας 3.14: Περιγραφή του πίνακα-σχέσης POICoordinate

2. PointOfInterest(N-1)

QueryDevice Πίνακας-Σχέση. Ο πίνακας αυτός αντιπροσωπεύει μια σχέση N-M και είναι στην πραγματικότητα η σχέση που συνδέει τα ερωτήματα και την συσκευή. Η σχέση αυτή είναι αναγκαία επειδή πολλά ερωτήματα μπορούν να γίνουν για μια συσκευή και, αντίστοιχα, πολλές συσκευές να υπάρχουν σε ένα ερώτημα.

Field	Type	R	U	Id	K	Description
ID	I	Y	Y	Y	P	Ταυτότητα του QueryDevice
QueryID	I	Y	N	N	FK	Στήλη που αντιστοιχεί στο QueryID του πίνακα Query
DeviceID	I	Y	N	N	FK	Στήλη που αντιστοιχεί στο DeviceID του πίνακα Device

Πίνακας 3.15: Περιγραφή του πίνακα-σχέσης QueryDevice

Αυτός ο πίνακας-σχέση σχετίζεται με τους παρακάτω πίνακες:

1. Query(N-1)
2. Device(M-1)

QueryCapability Πίνακας-Σχέση. Ο πίνακας αυτός αντιπροσωπεύει μια σχέση N-M και είναι στην πραγματικότητα η σχέση που συνδέει τα ερωτήματα και τις δυνατότητες των αισθητήρων. Η σχέση αυτή είναι αναγκαία επειδή πολλά ερωτήματα μπορούν να γίνουν για την άντληση μιας συγκεκριμένης μέτρησης κάποιου αισθητήρα και, αντίστοιχα, πολλοί αισθητήρες μπορούν να έχουν γίνει αντικείμενα ενός μόνο ερωτήματος.

Αυτός ο πίνακας-σχέση σχετίζεται με τους παρακάτω πίνακες:

1. Query(N-1)
2. Capability(M-1)

Field	Type	R	U	Id	K	Description
ID	I	Y	Y	Y	P	Ταυτότητα του QueryCapability
QueryID	I	Y	N	N	FK	Στήλη που αντιστοιχεί στο QueryID του πίνακα Query
CapabilityID	I	Y	N	N	FK	Στήλη που αντιστοιχεί στο CapabilityID του πίνακα Capability

Πίνακας 3.16: Περιγραφή του πίνακα-σχέσης QueryCapability

Κεφάλαιο 4

Λογικό Επίπεδο

Στο κεφάλαιο αυτό, θα αναλύσουμε το ενδιάμεσο επίπεδο της αρχιτεκτονικής 3-επιπέδων και θα δούμε αναλυτικά πως έχει γίνει ο σχεδίασμος, ποια εργαλεία χρησιμοποιήθηκαν και ποια είναι όλα τα υπο-συστατικά του λογικού επιπέδου. Εφόσον το επίπεδο αυτό, αποτελεί το συνδετικό κρίκο ανάμεσα στα δεδομένα και την εξωτερική εικόνα της εφαρμογής προς τον τελικό χρήστη, είναι απαραίτητη η ανάπτυξη του κατάλληλου λογισμικού το οποίο θα επιτρέπει αυτή την επικοινωνία και τη ροή πληροφορίας μεταξύ των επιπέδων. Σκοπός μας, εξ' αρχής, είναι ο σχεδιασμός να είναι αρκετά ευέλικτος έτσι, ώστε, να υπάρχει δυνατότητα επέκτασης και ενήμέρωσης της όλης εφαρμογής με σχετικά εύκολο και αποδοτικό χρονικά τρόπο. Για τη εκπλήρωση αυτού του σκοπού, σημαντικό ρόλο έπαιξε το εργαλείο Hibernate (Βλέπε Παράγραφος 4.2) το οποίο αποτελεί την υλοποίηση του Java Persistence API [1] και παρέχει ένα δυνατό πλαίσιο που βοηθάει του προγραμματιστές λογισμικού να διαχειρίζονται εύκολα τα σχεσιακά δεδομένα του υποκείμενου επιπέδου δεδομένων.

4.1 Η τεχνική ORM (Object Relational Mapping)

Το ORM είναι μια τεχνική προγραμματισμού για τη μετατροπή δεδομένων μεταξύ συστημάτων ασύμβατων τύπων στις σχεσιακές βάσεις δεδομένων και στις αντικειμενοστρεφείς γλώσσες προγραμματισμού. Αυτό δημιουργεί, πράγματι, μια "μια εικονική βάση δεδομένων ως αντικείμενο", το οποίο μπορεί να χρησιμοποιηθεί μέσα από τη γλώσσα προγραμματισμού.

[1] Το Java Persistence API, μερικές φορές συναντάται και ως JPA, είναι ένα πλαίσιο ανάπτυξης προγραμματισμού σε γλώσσα Java το οποίο επιτρέπει στους προγραμματιστές να διαχειρίζονται σχεσιακά δεδομένα σε εφαρμογές που αναπτύσσονται στις πλατφόρμες Java, Standard και Enterprise Edition.

4.1.1 Γιατί ORM?

Οι εργασίες που έχουν σχέση με τη διαχείριση δεδομένων στον αντικειμενοστραφή προγραμματισμό τυπικά υλοποιούνται με το χειρισμό αντικειμένων, τα οποία σχεδόν πάντα δεν έχουν μονοδιάστατες τιμές. Ο προγραμματιστής πρέπει είτε να μετατρέψει τις τιμές των αντικειμένων σε σύνολα από απλούστερες τιμές για αποθήκευση στη βάση δεδομένων, ή να χρησιμοποιήσει απλές, μονοδιάστατες τιμές μέσα στο πρόγραμμα. Το ORM βοηθά στην υλοποίηση της πρώτης προσέγγισης. Τα ίδια στοιχεία που αποθηκεύονται σε μια απλή τιμή αντικειμένου πιθανόν θα πρέπει να αποθηκεύονται σε διάφορους πίνακες της βάσης δεδομένων. Μια ORM υλοποίηση θα πρέπει συστηματικά να επιλέγει ποιους πίνακες να χρησιμοποιήσει και να δημιουργήσει την απαραίτητη SQL. Η τεχνική αυτή δίνει λύση σε θέματα όπως :

- Επιδόσεις
- Γραμμική επεκτασιμότητα
- Διαχειρισιμότητα των πράξεων CRUD (Δημιουργία, Ανάγνωση, Ενημέρωση και Διαγραφή) για τις πολύπλοκες σχέσεις
- Απλούστευση και συνέπεια της εγγραφής κώδικα για ταχεία ανάπτυξη εφαρμογών
- Συντήρηση και ευελιξία των εφαρμογών

Το πραγματικό κέρδος στη χρήση του ORM είναι η εξοικονόμηση χρόνου, η απλοποίηση της ανάπτυξης, η αύξηση των επιδόσεων ή της επεκτασιμότητας, και βοηθά στο να ελαχιστοποιηθούν οι αρχιτεκτονικές προκλήσεις που σχετίζονται με την αδυναμία των ORM εργαλείου ή την εμπειρία του προγραμματιστή.

4.1.2 Θέματα ORM

Πολλά πακέτα έχουν δημιουργηθεί για να υλοποιήσουν αυτή την τεχνική και να αυτοματοποιήσουν τη διαδικασίας της OR αντιστοίχισης. Έτσι, λαμβάνοντας υπόψη τον κατάλογο των πινάκων στη βάση δεδομένων, και τα αντικείμενα στο πρόγραμμα, θα αντιστοιχιθούν αυτόματα το ένα με το άλλο. Από την προοπτική του προγραμματιστή, το σύστημα που θα πρέπει να έχει μια μόνιμη συλλογή από αντικείμενα. Κάποιος, μπορεί να δημιουργήσει αντικείμενα και να δουλέψει με αυτά, και, τελικώς, αυτόματα θα τα αντικείμενα αυτά να καταλήγουν στη βάση δεδομένων.

Στην πράξη, όμως, τα πράγματα δεν είναι ποτέ αρκετά τόσο απλά. Όλα ORM συστήματα τείνουν να γίνονται ορατά με διάφορους τρόπους, μειώνοντας, έτσι, σε κάποιο βαθμό την ικανότητα κάποιου να αγνοήσει τη βάση δεδομένων. Ακόμη χειρότερα, το στρώμα μετάφρασης μπορεί να είναι χρονοβόρο και αναποτελεσματικό (ιδίως από το πως γράφει την SQL), με αποτέλεσμα τα προγράμματα που είναι πιο αργά και χρησιμοποιούν περισσότερη μνήμη από όταν ο κώδικας είναι γραμμένος "με το χέρι".

4.1.3 Είδη ORM

Η τεχνική ORM μπορεί να υλοποιηθεί με διάφορους τρόπους, όπως:

- *Pure relational*: Ολόκληρη η εφαρμογή, περιλαμβάνοντας και το UI, είναι σχεδιασμένη γύρω από το σχεσιακό μοντέλο και τις λειτουργίες που είναι βασισμένες στην SQL.
- *Light object mapping*: Οι οντότητες παρουσιάζονται σαν κλάσεις οι οποίες είναι χειροκίνητα αντιστοιχισμένες με τους σχεσιακούς πίνακες.
- *Medium object mapping*: Η εφαρμογή είναι σχεδιασμένη αντικειμενοστρεφώς. Η SQL παράγεται κατά τη διάρκεια της δημιουργίας της εφαρμογής μέσω ενός εργαλείου που δημιουργεί κώδικα ή κατά τη διάρκεια της εκτέλεσης από τον κώδικα του framework.
- *Full object mapping*: Αυτός ο τρόπος υποστηρίζει εξελιγμένη μοντελοποίηση των αντικειμένων όπως: σύνθεση, κληρονομικότητα, πολυμορφισμό, και μονιμότητα.

Στο Webdust2χρησιμοποιείται η τελευταία κατηγορία τεχνικής ORM. Αυτό γιατί χρησιμοποιούμε το Hibernate (Βλέπε Παράγραφος 4.2) το οποίο αποτελεί ένα ολοκληρωμένο εργαλείο αυτής της τεχνικής και αναλαμβάνει αυτοματοποιήσει διαδικασίες αντιστοίχισης και να φέρει σε πέρας πολλές λειτουργίες οι οποίες θα απαιτούσαν χειροκίνητο προγραμματισμό. Ο τρόπος με τον οποίο γίνεται η αντιστοίχιση παρουσιάζεται παρακάτω (Βλέπε Παράγραφος 4.6). Επιλέξαμε αυτή την τεχνική γιατί, όπως θα δούμε, χάρην αυτού του εργαλείου (hibernate), ο σχεδιασμός είναι πιο διαφανής και επιτρέπει σε κάποιον που θέλει να αναβαθμίσει ή να επεκτείνει την εφαρμογή να επέμβει άμεσα στον κώδικα και να κάνει τις κατάλληλες αλλαγές.

4.2 Hibernate

Το Hibernate Framework είναι λογισμικό ανοιχτού κώδικα (ελεύθερο λογισμικό) που σκοπό έχει να συνδέσει τα αντικείμενα που δημιουργούνται σε μία αντικειμενοστρεφή

γλώσσα προγραμματισμού (Java) με τους πίνακες μιας σχεσιακής βάσης δεδομένων. Η σύνδεση αυτή επιτυγχάνεται με την χρήση επιπρόσθετης πληροφορίας (metadata) που τοποθετείται(Βλέπε Παράγραφος 4.6) κατάλληλα (μαζί με τον κώδικα Java ή σε ξεχωριστά xml αρχεία) και περιγράφει την αντιστοιχία μεταξύ των αντικειμένων και της βάσης δεδομένων. Γενικά το Hibernate προσφέρει την αυτόματη μετατροπή της μίας μορφής (αντικείμενα) στην άλλη (σχεσιακή βάση δεδομένων).

4.2.1 Χρήση

Το Hibernate συγκαταλέγεται στην κατηγορία του ORM λογισμικού. Το ORM λογισμικό, όπως είδαμε παραπάνω (Βλέπε Παράγραφος 4.1), στοχεύει στην δημιουρία μιας διεπαφής (interface) μεταξύ των διαδεδομένων σχεσιακών βάσεων δεδομένων και του αντικειμενοστρεφούς προγραμματισμού. Με απλά λόγια, προσφέρει την χρησιμοποίηση μιας σχεσιακής βάσης δεδομένων ως αντικειμενοστεφή. Για να το επιτύχει αυτό δημιουργεί αντιστοιχίες μεταξύ των εννοιών του αντικειμενοστεφούς προγραμματισμού (συσχετίσεις, κληρονομικότητα, πολυμορφισμός) - που δεν υπάρχουν σε μία σχεσιακή βάση δεδομένων - και των πινάκων και σχέσεων μεταξύ των πινάκων μιας σχεσιακής βάσης. Με αυτό τον τρόπο ο προγραμματιστής βλέπει τελικά μία αντικειμενοστρεφή βάση δεδομένων, παρ'όλο που στην ουσία χρησιμοποιεί μια σχεσιακή. Έτσι ο προγραμματιστής χρησιμοποιεί τα αντικείμενα της συγκεκριμένης εφαρμογής, τα τροποποιεί σχετικά με τη λογική της εφαρμογής που αναπτύσσει και τα αποθηκεύει (τροποποιεί, διαγράφει και αναζητά) στην βάση ως αντικείμενα, σκεπτόμενος δηλαδή με αντικειμενοστραφείς έννοιες και όχι με βάση το σχήμα της σχεσιακής βάσης δεδομένων. Σε αυτό το σημείο είναι το Hibernate που, γνωρίζοντας την αντιστοιχία μεταξύ βάσης και λογικής της εφαρμογής, αναλαμβάνει να κατασκευάσει την κατάληλη εντολή της SQL η οποία και στέλνεται τελικά στην βάση δεδομένων. Έπειτα, τα αποτελέσματα που επιστρέφει η βάση το Hibernate τα επιστρέφει στον προγραμματιστή ως αντικείμενα της εφαρμογής. Το Hibernate αποτελεί μέρος του ενδιαμέσου επιπέδου μεταξύ εφαρμογής και βάσης δεδομένων.

4.2.2 Χαρακτηριστικά

Το Hibernate προσφέρει τα παρακάτω στον προγραμματιστή:

- *Παραγωγικότητα*: Στην ανάπτυξη λογισμικού ένα μεγάλο μέρος της προγραμματιστικής προσπάθειας αφιερώνεται στην διεπαφή της εφαρμογής με τη βάση δεδομένων. Το Hibernate αυτοματοποιώντας τις βασικές λειτουργίες (CRUD – Create Read Update Delete) επιτρέπει αρχικά στον προγραμματιστή να επικεντρώνει την

προσπάθειά του στη λογική της εφαρμογής (business logic). Επίσης, υπάρχει η δυνατότητα να ακολουθηθούν δύο στρατηγικές αναπτυξης λογισμικού: είτε αρχίζοντας από το μοντέλο δεδομένων είτε από τη βάση δεδομένων. Αυτό μειώνει σε μεγάλο βαθμό το χρόνο ανάπτυξης.

- *Συντηρησιμότητα*: Με τη χρήση του Hibernate γράφονται σημαντικά λιγότερες γραμμές κώδικα και ο κώδικας είναι πιο κατανοητός και καλογραμμένος. Αυτό κάνει την συντήρηση της εφαρμογής ευκολότερη.

- *Ανεξαρτησία από τη βάση δεδομένων*: Με τη συμβατότητα του Hibernate με διαφορετικές βάσεις δεδομένων και τη δυνατότητα σύνδεσής του με τη βάση μέσω δηλώσεων οριζομένων σε ειδικό αρχείο η αναπτυσσόμενη εφαρμογή μπορεί με ελάχιστες τροποποιήσεις να χρησιμοποιηθεί με βάσεις δεδομένων διαφορετικών κατασκευαστών. Το γεγονός αυτό στερεί μεν από το Hibernate την εκμετάλλευση των ιδιαίτερων χαρακτηριστικών της χρησιμοποιούμενης βάσης, όμως, και σε αυτή την περίπτωση, δίνεται η δυνατότητα χρήσης πηγαίας SQL μέσα στο Hibernate που εκμεταλλεύεται τα ιδιαίτερα αυτά χαρακτηριστικά. Αυτό βέβαια μειώνει την ανεξαρτησία του Hibernate.

4.2.3 Αντικείμενα στο Hibernate

Είναι επιτακτικό, σε αυτό το σημείο, να πούμε μερικά πράγματα για τον κύκλο ζωής των αντικειμένων, μιας και το Hibernate χειρίζεται και δουλεύει πάνω σε αντικείμενα. Το πώς γίνεται ένα αντικείμενο persistent, το πως σταματά να θεωρείται persistent καθώς και το πως μέθοδοι και άλλες λειτουργείες ενεργοποιούν αυτές τις μεταβάσεις, είναι μερικά θέματα τα οποία επιδέχονται αναφορά. Το `Session` είναι υπεύθυνο για για τη διαχείριση των καταστάσεων των αντικειμένων. Παρακάτω υπάρχει μια σύντομη περιγραφή των καταστάσεων των αντικειμένων, του persistent περιεχομένουΒλέπε Παράγραφος 4.2.3.2, της ταυτότητα των αντικειμένων και των αντικειμένων σε detached κατασταση. Για περαιτέρω πληροφορίες και ανάλυση του θέματος μπορείται να ανατρέξετε: *Java Persistence with Hibernate, Chapter 9 – Working with objects*.

4.2.3.1 Καταστάσεις αντικειμένων

Κατά τη διάρκεια της ζωής του, ένα αντικείμενο μπορεί να μεταβεί από διάφορες καταστάσεις. Αυτές μπορεί να είναι:

- *Transient* - αντικείμενα που αρχικοποιούνται με τον τελεστή `new`, δεν έχουν καμία σχέση με τη βάση δεδομένων και η κατάστασή τους έχει χαθεί εφόσον δεν αναφέρεται σε αυτό κάποιο άλλο αντικείμενο.
- *Persistent* - αντικείμενο που είναι στιγμιότυπο μιας οντότητας χάρη στην ταυτότητα που υπάρχει λόγω της βάσης δεδομένων. Τα persistent αντικείμενα μπορεί να είναι αντικείμενα που έχουν αρχικοποιηθεί από την εφαρμογή, και στη συνέχεια να έχουν γίνει persistent από μια κλήση μεθόδου. Τα persistent στιγμιότυπα σχετίζονται πάντα με το *persistent περιεχόμενο*Βλέπε Παράγραφος 4.2.3.2.
- *Detached* - αντικείμενα που ήταν persistent σε ένα persistent περιεχόμενο και μετά τη λήξη του θα χρησιμοποιούνται ακόμη από την εφαρμογή, αλλά χωρίς κανένα συγχρονισμό με τη βάση δεδομένων.
- *Removed* - αντικείμενα που έχουν προγραμματιστεί για διαγραφή στο τέλος ενός συνόλου εργασιών, αλλά εξακολουθούν να διαχειρίζονται από το persistent περιεχόμενο εως ότου να ολοκληρωθεί αυτό το σύνολο εργασιών.

4.2.3.2 Persistent Περιεχόμενο

Το persistent περιεχόμενο μπορεί να θεωρηθεί ότι είναι μια προσωρινή μνήμη που διαχειρίζεται στιγμιότυπα των αντικειμένων-οντοτήτων. Σε μια Hibernate εφαρμογή, ένα `Session` έχει ένα persistent περιεχόμενο. Το persistent περιεχόμενο είναι χρήσιμο για διάφορους λόγους:

- Το Hibernate μπορεί να κάνει αυτόματο *dirty checking* και *transactional write-behind*.
- Το Hibernate να χρησιμοποιήσει το persistent περιεχόμενο ως ένα πρώτο επίπεδο προσωρινής μνήμης.
- Το Hibernate μπορεί να εγγυηθεί εμβέλεια της ταυτότητας ενός java αντικειμένου.
- Το Hibernate μπορεί να επεκτείνει το persistent περιεχόμενο και να καλύψει μια ολόκληρη *συζήτηση*[2].

Το Hibernate `Session` υλοποιεί *write-behind*. Αλλαγές στα persistent αντικείμενα γίνονται μέσα στην εμβέλεια ενός persistent περιεχομένου και δεν μεταδίδονται στη βάση δεδομένων. Αυτό επιτρέπει στο Hibernate να συγχωνεύσει πολλές αλλαγές σε ένα ελάχιστο αριθμό των αιτημάτων προς τη βάση δεδομένων, βοηθώντας την ελαχιστοποίηση

[2]Η έννοια της *συζήτησης* (conversation) εδώ έχει την έννοια της επικοινωνίας για τη μεταφορά δεδομένων. Για παράδειγμα, μια σειρά από αιτήσεις θα μπορουσε να αποτελεί μια *συζήτηση*

των επιπτώσεων λόγω της καθυστερημένης απόκρισης του δικτίου. Ο συγχρονισμός του persistent περιεχομένου με τη βάση δεδομένων ονομάζεται *flushing*. Η διαδικασία αυτή συμβαίνει όταν:

- Όταν μια `Transaction`(Βλέπε Παράγραφος 4.2.4) στο Hibernate API έχει γίνει commit.
- Πριν να εκτελεστεί ένα ερώτημα(*query*).
- Όταν μια εφαρμογή καλέσει εξωτερικά `session.flush()`.

4.2.3.3 Detached Αντικείμενα

Εάν ένα αντικείμενο βγει από την εμβέλεια της σίγουρης, εγγυημένης ταυτότητας(αναγνωριστικού), θα το ονομάσουμε: αναφορά σε ένα detached αντικείμενο. Για παράδειγμα, εάν πρόκειται να ελέγξουμε την ταυτότητα(αναγνωριστικο id) των δύο αντικειμένων αφού έχουν εγκαταλείψει το persistent περιεχόμενο τότε, ίσως πάρουμε λάθος αποτελέσματα σε σχέση με την ταυτότητες(αναγνωριστικα id) που είναι αποθηκευμένα στη βάση δεδομένων. Σε περίπτωση που έχουμε έναν detached αντικείμενο, και μετά τη δημιουργία ενός νέου `Session`, μπορούμε εκ νέου το επισυνάψουμε σε ένα νέο αντικείμενο, χρησιμοποιώντας τη μέθοδο session.merge ().

4.2.4 Transactions

Οι βάσεις δεδομένων υλοποιούν την ιδέα μιας μερίδας εργασιών ως ένα database transaction. Ένα database transaction ομαδοποιεί τις λειτουργίες που απαιτούν πρόσβαση στα δεδομένα, οι οποίες είναι SQL λειτουργίες. Όλες οι SQL προτάσεις εκτελούνται μέσα σε ένα και μοναδικό transaction. Δεν υπάρχει τρόπος να σταλεί μια SQL πρόταση εκτός ενός database transaction. Ένα database transaction εγγυάται ότι θα τελειώνει με έναν από τους δυο παρακάτω τρόπους: Είτε θα εντελώς `commited` ή εντελώς *rolled back*. Επίσης λέμε ότι τα transactions είναι *atomic*. Το Hibernate χρησιμοποιεί το `org.hibernate.Transaction` σαν ένα *interface*. Αυτό το *interface* χρησιμοποιούμε σε όλη μας την εφαρμογή.

4.3 Σχεδιασμός Οντοτήτων-Κλάσεων

Στα παραπάνω έγινε λόγος για την αντιστοίχιση των αντικειμένων της εφαρμογής με τους σχεσιακούς πίνακες της βάσης δεδομένων. Εδώ περιγράφουμε αναλυτικά ποια είναι τα αντικείμενα-οντότητες-κλασεις οι οποίες ορίστικαν έτσι ώστε να αντιστοιχίζονται στους πίνακες της HSQLDB βάσης δεδομένων της εφαρμογής. Όλες αυτές οι οντότητες κλάσεις αποτελούν μέρος του λογικού επιπέδου και είναι αυτές που υλοποιούν την παραπάνω τεχνική του ORM. Είναι διαχωρισμένες σε 3 κατηγορίες ανάλογα με τη σχέση τους με τα φυσικά χαρακτηριστικά της εφαρμογής και φαίνονται παρακάτω:

Logic – Οντότητες που έχουν να κάνουν καθαρά με τη λογική της εφαρμογής και τον τρόπο που είναι οργανωμένη.

Geo – Οντότητες που σχετίζονται με τα φυσικά γεωγραφικά στοιχεία της εφαρμογής.

Query – Οντότητες που σχετίζονται με τα ερωτήματα (*queries*) που αφορούν στην εφαρμογή και την ανάκτηση πληροφορίας.

Σημείωση: *Λαμβάνοντας υπόψιν μας ότι η εφαρμογή είναι ανεπτυγμένη σε java, οι τύποι δεδομένων που αναφέρονται αντιστοιχούν σε αυτούς που χρησιμοποιουνται στη java.*

4.3.1 Logic Οντότητες

4.3.1.1 DataType οντότητα

Η οντότητα-κλάση αυτή σχετίζεται με τον πίνακα DataType και τα μέλη της έχουν ως εξής:

Field	Data type	Description
ID	int	Ταυτότητα του data type
Name	String	Όνομα του data type
JavaClassType	class	Ο τύπος δεδομένων της Java κλάσης του data type
isNumeric	boolean	Είναι αριθμήσιμη ή όχι η τιμή του DataType
capabilities	Set	Σύνολο από Capabilities

Πίνακας 4.1: Περιγραφή της οντότητας DataType

Αυτή η οντότητα έχει σχέση με τις παρακάτω:

1. Capability (1-N)

4.3.1.2 Device οντότητα

Η οντότητα-κλάση αυτή σχετίζεται με τον πίνακα Device και τα μέλη της έχουν ως εξής:

Field	Data type	Description
ID	int	Ταυτότητα του Device
deviceType	DeviceType	Τύπος Device
deviceNetwork	DeviceNetwork	Δίκτυο που ανήκει το Device
Name	String	Όνομα του Device
LastUpdate	Timestamp	Τελευταία ενημέρωση του Device
deviceCoordinate	Coordinate	Συντεταγμένες του Device
capabilities	Set	Σύνολο από Capabilities
readings	Set	Σύνολο από Readings

Πίνακας 4.2: Περιγραφή της οντότητας Device

Αυτή η οντότητα έχει σχέση με τις παρακάτω:

1. Capability (N-M)
2. DeviceType (N-1)
3. DeviceNetwork (N-1)
4. Coordinate (1-1)
5. PointOfInterest (N-M)
6. DeviceQuery (M-N)
7. Reading (1-N)

4.3.1.3 DeviceType οντότητα

Η οντότητα-κλάση αυτή σχετίζεται με τον πίνακα DeviceType και τα μέλη της έχουν ως εξής:

Αυτή η οντότητα έχει σχέση με τις παρακάτω:

1. Device (1-N)

Field	Data type	Description
ID	int	Ταυτότητα του Device type
Name	String	Όνομα του Device type
Description	String	Περιγραφή του Device type
URL	String	URL για πληροφορίες σχετικά με το Device type
devices	Device	Σύνολο από devices

Πίνακας 4.3: Περιγραφή της οντότητας DeviceType

Field	Data type	Description
ID	integer	Ταυτότητα του Capability
capabilityType	CapabilityType	Τύπος του Capability
Name	String	Όνομα του Capability
Accuracy	String	Ακρίβεια του Capability
Units	String	Αριθμός μονάδων του Capability
dataType	DataType	Τύπος δεδομένων του Capability
devices	Set	Σύνολο από Devices
readings	Set	Σύνολο από Readings

Πίνακας 4.4: Περιγραφή της οντότητας Capability

4.3.1.4 Capability οντότητα

Η οντότητα-κλάση αυτή σχετίζεται με τον πίνακα Capability και τα μέλη της έχουν ως εξής:

Αυτή η οντότητα έχει σχέση με τις παρακάτω:

1. Device (M-N)
2. CapabilityType (N-1)
3. DataType (N-1)
4. Reading (1-N)
5. DeviceQuery (M-N)

4.3.1.5 CapabilityType οντότητα

Η οντότητα-κλάση αυτή σχετίζεται με τον πίνακα CapabilityType και τα μέλη της έχουν ως εξής:

Αυτή η οντότητα έχει σχέση με τις παρακάτω:

Field	Data type	Description
ID	integer	Ταυτότητα του CapabilityType
Name	String	Όνομα του CapabilityType
capabilities	Set	Σύνολο από Capabilities

Πίνακας 4.5: Περιγραφή της οντότητας CapabilityType

1. Capability (1-N)

4.3.1.6 DeviceNetwork οντότητα

Η οντότητα-κλάση αυτή σχετίζεται με τον πίνακα DeviceNetwork και τα μέλη της έχουν ως εξής:

Field	Data type	Description
ID	int	Ταυτότητα του Device Network
Name	String	Όνομα του Device Network
Description	String	Περιγραφή του Device Network
EnclosingCoordinateTopLeft	Coordinate	Πάνω αριστερά συντεταγμένη του Device Network
EnclosingCoordinateBottomRight	Coordinate	Κάτω δεξιά συντεταγμένη του Device Network
devices	Set	Σύνολο από Devices που περιλαμβάνονται στο Device Network

Πίνακας 4.6: Περιγραφή της οντότητας DeviceNetwork

Αυτή η οντότητα έχει σχέση με τις παρακάτω:

1. Device (1-N)
2. DeviceQuery (1-N)
3. Coordinate (δύο N-1 σχέσεις - πεδία EnclosingCoordinateTopLeft και EnclosingCoordinateBotton

4.3.2 Geo Οντότητες

4.3.2.1 Coordinate οντότητα

Η οντότητα-κλάση αυτή σχετίζεται με τον πίνακα Coordinate και τα μέλη της έχουν ως εξής:

Field	Data type	Description
ID	int	Ταυτότητα του Coordinate
X	double	X γεωγραφικό μήκος (X θέση)
Y	double	Y γεωγραφικό πλάτος (Y θέση)
AbsAltitude	double	Απόλυτο ύψος (από το ύψος της θάλασσας)
FromGroundAltitude	double	Ύψος (από το έδαφος)
Description	String	Περιγραφή για τις συντεταγμένες

Πίνακας 4.7: Περιγραφή της οντότητας Coordinate

Αυτή η οντότητα έχει σχέση με τις παρακάτω:

1. Device (1-1)
2. PointOfInterest (M-N)
3. DeviceNetwork (δύο 1-N σχέσεις - πεδίο EnclosingCoordinateTopLeft και EnclosingCoordinateB

4.3.2.2 PointOfInterest οντότητα

Η οντότητα-κλάση αυτή σχετίζεται με τον πίνακα PointOfInterest και τα μέλη της έχουν ως εξής:

Field	Data type	Description
ID	int	Ταυτότητα του PointOfInterest
Name	double	Όνομα του PointOfInterest
Description	String	Περιγραφή για το PointOfInterest

Πίνακας 4.8: Περιγραφή της οντότητας PointOfInterest

Αυτή η οντότητα έχει σχέση με τις παρακάτω:

1. Device (M-N)
2. Coordinate (N-M)

4.3.3 Queries Οντότητες

4.3.3.1 Reading οντότητα

Η οντότητα-κλάση αυτή σχετίζεται με τον πίνακα Reading και τα μέλη της έχουν ως εξής:

Field	Data type	Description
ID	int	Ταυτότητα του Reading
readingDevice	Device	η συσκευή από όπου λήφθηκε το Reading
readingCapability	Capability	η μέτρηση έλαβε το Reading
Date	Date	Ημερομηνία του Reading
ValueBinary	double	Η τιμή του Reading - Ως binary
ValueNumeric	double	Η τιμή του Reading - Ως numeric

Πίνακας 4.9: Περιγραφή της οντότητας Reading

Αυτή η οντότητα έχει σχέση με τις παρακάτω:

1. Device (N-1)
2. Capability (N-1)

4.3.3.2 DeviceQuery οντότητα

Η οντότητα-κλάση αυτή σχετίζεται με τον πίνακα DeviceQuery και τα μέλη της έχουν ως εξής:

Field	Data type	Description
ID	int	Ταυτότητα του DeviceQuery
Name	Double	Όνομα του DeviceQuery
Description	String	Περιγραφή για το DeviceQuery
Start	Timestamp	Στιγμή εκκίνησης του DeviceQuery
Stop	Timestamp	Στιγμή λήξης του DeviceQuery
Expression	String	Η πρόταση του DeviceQuery
Period	Double	Περίοδος του DeviceQuery
IsActive	Boolean	Κατάσταση του DeviceQuery
Result	String	Αποτέλεσμα που επιστρέφει το DeviceQuery
queryType	QueryType	Τύπος του DeviceQuery
deviceNetwork	DeviceNetwork	Δίκτυο που έγινε το DeviceQuery
devices	Set	Σύνολο από devices
capabilities	Set	Σύνολο από capabilities

Πίνακας 4.10: Περιγραφή της οντότητας DeviceQuery

Αυτή η οντότητα έχει σχέση με τις παρακάτω:

1. DeviceNetwork (N-1)

2. QueryType (N-1)
3. Device (N-M)
4. Capability (N-M)

4.3.3.3 QueryType οντότητα

Η οντότητα-κλάση αυτή σχετίζεται με τον πίνακα QueryType και τα μέλη της έχουν ως εξής:

Field	Data type	Description
ID	int	Ταυτότητα του DeviceQuery
Name	Double	Όνομα του DeviceQuery

Πίνακας 4.11: Περιγραφή της οντότητας QueryType

Αυτή η οντότητα έχει σχέση με τις παρακάτω:

1. DeviceQuery (1-N)

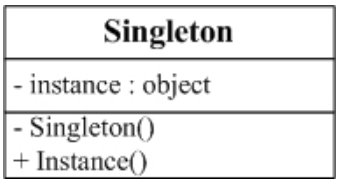
4.4 Design Patterns

Τα Design Patterns είναι, βασικά, εργαλεία σχεδιασμού τα οποία χρησιμοποιούνται για να αναβαθμίσουν τον υπάρχον κώδικα. Επιτρέπουν στον προγραμματιστή, ο κώδικας που γράφει να είναι πιο εύκολος στο να υλοποιηθεί, να γίνει *build* και να συντηρηθεί. Είναι εργαλεία τα οποία αυξάνουν την αποδοτικότητα, και το πιο σημαντικό είναι οτι επιτρέπουν στον προγραμματιστή να αναβαθμίσει τις επιδεξιότητές του πάνω στο σχεδιαστικό κομμάτι των εφαρμογών, να βελτιώσει την ποιότητα των έργων του και να του δώσει ένα πιο ευρύ φάσμα όσον αναφορά το σύνολο των δεξιοτήτων του. Δίνουν τη δινατότητα στον προγραμματιστή να βρει νέες απαντήσεις σε τυπικά και πιο εξειδικευμένα προβλήματα που αντιμετωπίζει καθόλη τη διάρκεια ανάπτυξης κώδικα. Ορίζουν ένα κοινό προγραμματιστικό μοντέλο το οποίο μπορεί να κατανοηθεί και να μεταφραστεί μεταξύ όλων εκείνων των προγραμματιστών που είναι εξοικοιωμένοι με τέτοια πρότυπα. Τα Design Patterns τυποποιούν κοινά προγραμματιστικά έργα σε αναγνωρίσιμες φόρμες δίνοντας στις εφαρμογές καλύτερη συνοχή και γενικά βοηθούν στο να γίνει κάποιος προγραματιστής καλύτερος στον σχεδιασμό. Τέλος, πρέπει να δηλώσουμε εδώ, οτι αυτά τα εργαλεία σχεδιασμού είναι βασισμένα και εξαρτώνται από τις πλευρές των αντικειμενοστρεφών γλωσσών προγραμματισμου. Στην εφαρμογή Webdust2,

χρησιμοποιείται η γλώσσα προγραμματισμού JAVA -μακράν αντικειμενοστρεφής- και επομένως τέτοια εργαλεία είναι αρκετά χρήσιμα για την ανάπτυξη και σχεδιασμό του κώδικα της παρούσας εφαρμογής.

Singleton Pattern

Το *Singleton* pattern είναι ένας τρόπος να παρέχεις καθολική προσπέλαση στο στιγμιότυπο μιας κλάσης χωρίς να είναι διαθέσιμος ο δημιουργός εκτός αυτής της κλάσης. Η κλάση singleton δημιουργεί από μόνη της ένα στιγμιότυπο και το διατηρεί μεταξύ πολλαπλών νημάτων διεργασιών. Το *Singleton* pattern έχει ένα κύριο συστατικό: το *Singleton*. Η singleton κλάση λειτουργεί σαν μια καθολική 'αποθήκη' όπου φυλάσσεται το ίδιο της το στιγμιότυπο και ο δημιουργός της είναι private. Έτσι, δεν δημιουργείται κανένα στιγμιότυπο εκτός κλάσης και μόνο ένα στιγμιότυπο κατοικοεδρεύει μέσα στο singleton.



Σχήμα 4.1: UML Διάγραμμα για το Singleton pattern

Όπως θα δούμε πιο κάτω, στην εφαρμογή μας χρησιμοποιούμε κάποιους χειριστές (Βλέπε Παράγραφος 4.5) για τα δεδομένα και τη διαχείρισή τους. Επειδή θέλουμε κατα τη διάρκεια του runtime της εφαρμογής να έχουμε μόνο ένα στιγμιότυπο τέτοιων χειριστών αλλά να μπορούν να υπάρχουν πολλά διαφορετικά νήματα διεργασιών να έχουν πρόσβαση και να διατηρούν αυτό το στιγμιότυπο καταφύγαμε στη χρήση αυτής της τεχνικής σχεδιασμού.

4.5 Χειριστές Δεδομένων

Η αρχιτεκτονική που χρησιμοποιεί η εφαρμογή Webdust2είναι μια αρχιτεκτονική επιπέδων και στρωμάτων. Με τον τρόπο αυτό επιτυγχάνουμε αφαίρεση και καλύτερη οργάνωση αλλά και ροή δεδομένων από επίπεδο σε επίπεδο. Το λογικό επίπεδο στο οποίο βρισκόμαστε είναι το επίπεδο το οποίο αποτελεί το συνδετικό μέσο μεταξύ των δεδομένων και της παρουσίασης της εφαρμογής στο χρήστη. Λόγω αυτής της αφαίρεσης, θα πρέπει να ορίσουμε κάποια αντικείμενα(παροχή ενός υποτυπώδους interface) με τα οποία, εύκολα κάποιος, από το επίπεδο παρουσίασης θα μπορεί να χρησιμοποιήσει για να χειριστεί τα δεδομένα. Έτσι, φτάνουμε στην ανάγκη του ορισμού των χειριστών δεδομένων. Οι χειριστές δεδομένων είναι στην ουσία κάποιες java κλάσεις οι οποίες είναι

έτσι γραμμένες για να προσφέρουν ένα interface μέσω του οποίου θα μπορεί να γίνει ευκολότερος ο χειρισμός του συστήματος, όπως π.χ. προσθήκη ενός καινούργιου κόμβου αισθητήρων, αναζήτηση κάποιας μέτρησης με βάση το όνομα ή επιστροφή μιας λίστας με όλα δίκτυα που βρίσκονται σε μια συγκεκριμένη περιοχή.

Για σκοπούς καλύτερης οργάνωσης της λογικής της εφαμογής και λειγότερων γραμμών κώδικα ακολουθήσαμε κάποιες από τις τεχνικές που προσφέρει ο προγραμματισμός σε JAVA αλλά και τα design patterns(Βλέπε Παράγραφος 4.4). Ορίσαμε μια γενική κλάση-χειριστή(AbstractManager) η οποία είναι βασική κλάση που χειρίζεται όλες τις CRUD λειτουργίες του συστήματός μας. Οι υπόλοιπες υπό-κλάσεις κληρονομούν από αυτήν. Η κάθε υπο-κλάση είναι υλοποιημένη σύμφωνα με το Singleton pattern.

4.5.1 AbstractManager

Αυτή η κλάση υλοποιεί έναν abstract χειριστή ο οποίος είναι υπεύθυνος για όλες τις βασικές CRUD λειτουργίες τις βάσης δεδομένων. Οι υπόλοιπες υπο-κλάσεις κληρονομούν από αυτήν. Λειτουργίες όπως εισαγωγή, ενημέρωση, διαγραφή, επιστροφή λίστας παρέχονται από αυτή την κλάση και όλες οι υπο-κλάσεις τις κληρονομούν. Οι μέθοδοι τις κλάσης αναλυτικά είναι οι εξής:

`add` – Προσθέτει μία νέα εγγραφή στη βάση δεδομένων, σύμφωνα με το αντικείμενο εισόδου.
Είσοδος: Ένα αντικείμενο οποιοδήποτε τύπου.
Έξοδος: όχι.
Έλεγχοι: όχι.

`update` – Ενημερώνει μία εγγραφή στη βάση δεδομένων, σύμφωνα με το αντικείμενο εισόδου.
Είσοδος: Ένα αντικείμενο οποιοδήποτε τύπου.
Έξοδος: όχι.
Έλεγχοι: όχι.

`delete` – Διαγράφει μία εγγραφή από τη βάση δεδομένων, σύμφωνα με το αντικείμενο εισόδου.
Είσοδος: Ένα αντικείμενο οποιοδήποτε τύπου και το id του.
Έξοδος: όχι.
Έλεγχοι: Έλεγχος για το αν το διεγραμένο αντικείμενο έχει σχέση με άλλα.

`list` – Επιστρέφει μια λίστα με όλες τις εγγραφές από τη βάση δεδομένων, σύμφωνα με το αντικείμενο εισόδου.

Είσοδος: Ένα αντικείμενο οποιοδήποτε τύπου.
Έξοδος: Μια λίστα με όλες τις εγγραφές(αντικείμενα) που υπάρχουν στον πίνακα που σχετίζεται με το αντικείμενο που δίνεται στην είσοδο.
Έλεγχοι: όχι.

`getByID` – Επιστρέφει την εγγραφή από τη βάση δεδομένων που αντιστοιχεί στο id του αντικειμένου της εισόδου.
Είσοδος: Ένα αντικείμενο οποιοδήποτε τύπου και το id του.
Έξοδος: Ένα αντικείμενο τύπου οντότητας.
Έλεγχοι: όχι.

4.5.2 CapabilityManager

Αυτή η κλάση υλοποιεί τον χειριστή για όλες τις λειτουργίες που αφορούν την οντότητα Capability. Κληρονομεί όλες τις μεθόδους της κλάσης AbstractManager και δεν έχει καμία επιπλέον μέθοδο.

4.5.3 CapabilityTypeManager

Αυτή η κλάση υλοποιεί τον χειριστή για όλες τις λειτουργίες που αφορούν την οντότητα CapabilityType. Κληρονομεί την κλάση AbstractManager και δεν έχει καμία επιπλέον μέθοδο.

4.5.4 CoordinateManager

Αυτή η κλάση υλοποιεί τον χειριστή για όλες τις λειτουργίες που αφορούν την οντότητα Coordinate. Κληρονομεί την κλάση AbstractManager και παρέχει επιπλέον κλάσεις που αφορούν στο χειρισμό των συντεταγμένων της εφαρμογής.

Οι επιπλέον μέθοδοι που παρέχονται από αυτόν το χειριστή είναι οι εξής:

`getAllIncluded(final Coordinate topLeft, final Coordinate bottomRight)` – Εκτελεί ερώτημα (*query*) στη βάση δεδομένων και επιστρέφει μια λίστα από τις συντεταγμένες που βρίσκονται μέσα στα όρια τα οποία δόθηκαν στην είσοδο.
Input: Η πάνω αριστερά και η κάτω δεξιά συντεταγμένη (ορίζοντας την περιοχή ορθογωνίου) όπως αυτές είναι ορισμένες στην οντότητα-κλάση Coordinate.
Output: Μία λίστα με τις συντεταγμένες που βρίσκονται μέσα στα όρια.
Checks: όχι.

4.5.5 DataTypeManager

Αυτή η κλάση υλοποιεί τον χειριστή για όλες τις λειτουργίες που αφορούν την οντότητα DataType. Κληρονομεί την κλάση AbstractManager και δεν έχει καμία επιπλέον μέθοδο.

4.5.6 DeviceManager

Αυτή η κλάση υλοποιεί τον χειριστή για όλες τις λειτουργίες που αφορούν την οντότητα Device. Κληρονομεί την κλάση AbstractManager και παρέχει επιπλέον κλάσεις που αφορούν στο χειρισμό των συσκευών της εφαρμογής.

Οι επιπλέον μέθοδοι που παρέχονται μέσω αυτού του χειριστή είναι οι εξής:

`getByNodeIDNetworkID(final int nodeID, final DeviceNetwork net)` Αναζητά την συσκευή η οποία έχει το id που δίνει ο χρήστης και βρίσκεται στο συγκεκριμένο δίκτυο αισθητήρων.
Input: Ένα id συσκευής και ένα δίκτυο στο οποίο η συσκευή βρίσκεται.
Output: Η συσκευή η οποία πληρεί τα παραπάνω κριτήρια.
Checks: όχι.

4.5.7 DeviceTypeManager

Αυτή η κλάση υλοποιεί τον χειριστή για όλες τις λειτουργίες που αφορούν την οντότητα DeviceType. Κληρονομεί την κλάση AbstractManager και παρέχει επιπλέον κλάσεις σχετικά με το χειρισμό του τύπου των συσκευών.

Οι επιπλέον μέθοδοι που παρέχονται μέσω αυτού του χειριστή είναι οι εξής:

`searchByName(String name)` Εκτελεί ερώτημα (*query*) στη βάση δεδομένων και αναζητά όλες τις συσκευές που έχουν τον τύπο ο οποίος δίνεται στην είσοδο.
Input: το όνομα του τύπου της συσκευής που δίνεται από το χρήστη (case-sensitive).
Output: Λίστα με τις συσκευές που έχουν το δοσμένο από το χρηστη τύπο.
Checks: όχι.

`searchByDescription(String description)` Εκτελεί ερώτημα (*query*) στη βάση δεδομένων και αναζητά όλες τις συσκευές που έχουν την περιγραφή η οποία δίνεται στην είσοδο.

Input: Μια περιγραφή (μια λέξη ή περισσότερες) του τύπου της συσκευής.
Output: Λίστα με τους τύπους συσκευών που έχουν τη δοσμένη περιγραφή.
Checks: όχι.

4.5.8 DeviceNetworkManager

Αυτή η κλάση υλοποιεί τον χειριστή για όλες τις λειτουργίες που αφορούν την οντότητα device networks. Κληρονομεί την κλάση AbstractManager και παρέχει επιπλέον κλάσεις σχετικά με το χειρισμό του τύπου των δικτύων αισθητήρων.

Οι επιπλέον μέθοδοι που παρέχονται από αυτόν το χειριστή είναι οι εξής:

`getAllIncluded(final Coordinate topLeft, final Coordinate bottomRight)` Ο δημιουργός που αρχικοποιεί τον χειριστή.
Input: Μία σύνδεση με τη βάση δεδομένων - θα πρέπει να είναι ήδη αρχικοποιημένη.
Output: όχι.
Checks: όχι.

4.5.9 ReadingManager

Αυτή η κλάση υλοποιεί τον χειριστή για όλες τις λειτουργίες που αφορούν την οντότητα readings. Κληρονομεί την κλάση AbstractManager και παρέχει επιπλέον κλάσεις σχετικά με το χειρισμό του τύπου των μετρήσεων.

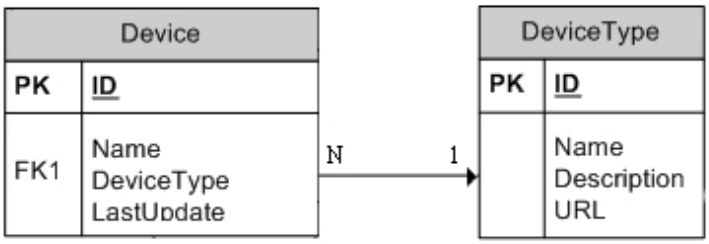
Οι επιπλέον μέθοδοι που παρέχονται μέσω αυτού του χειριστή είναι οι εξής:

`latestByDevice(final Device dev)` Λήψη της τελευταίας μέτρησης κάποιας συσκευής.
Input: Μια συσκευή.
Output: Η τελευταία μέτρηση.
Checks: όχι.

`latestByDeviceCapability(final Device dev, final Capability cap)` Ο δημιουργός που αρχικοποιεί τον χειριστή.
Input: Μια συσκευή και μια δυνατότητα ενός αισθητήρα.
Output: Η τελευταία μέτρηση.
Checks: όχι.

4.6 Παράδειγμα με χρήση Hibernate

Για να γίνει στον αναγνώστη πιο κατανοητή η λειτουργία του συστήματος με τη βοήθεια του εργαλείου Hibernate σε αυτό το σημείο θα κάνουμε την περιγραφή μιας τυπικής περίπτωσης. Έστω ότι έχουμε τον πίνακα των συσκευών και τον πίνακα του τύπου των συσκευών. Η σχέση που συνδέει αυτούς τους δύο πίνακες είναι N-1 (Βλέπε Σχήμα 4.2).



Σχήμα 4.2: Σχέση μεταξύ Device και DeviceType.

Προσέξτε ότι το πεδίο DeviceType του πίνακα DEVICE είναι εξωτερικό κλειδί (foreign key) και συνδέει το Device με τους τύπους συσκευών DEVIVE_TYPE. Στην εφαρμογή μας πρέπει να κατασκευάσουμε δύο αντικείμενα τα Device και DeviceType:

Αρχείο *Device.java*:

```
package webdust.core.logic;

import webdust.core.geo.Coordinate;
import java.sql.Timestamp;

public class Device {
private int ID;
private String Name;
private Timestamp LastUpdate;
private DeviceType deviceType;

public DeviceType getDeviceType() { // Getter
return deviceType;
}
public void setDeviceType(final DeviceType deviceType) { // Setter
this.deviceType = deviceType;
}
// The rest of Getters and Setters
}
```

Αρχείο *DeviceType.java*:

```
package webdust.core.logic;

public class DeviceType {
private int ID;
private String Name;
private String Description;
private String URL;

public String getName() { //Getter
return Name;
}
public void setName(final String name) { // Setter
Name = name;
// The rest of Getters and Setters
}
```

Χρησιμοποιόντας το Hibernate πρέπει να κατασκευάσουμε τα XML αρχεία που περιγράφουν την αντιστοιχία μεταξύ των αντικειμένων και των πινάκων της βάσης δεδομένων:

Αρχείο *Device.hbm.xml*:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="webdust.core.logic.Device" table="DEVICE">
<id name="ID" column="DEVICE_ID">
<generator class="native"/>
</id>
<property name="Name" not-null="true"/>
<property name="LastUpdate" type="date"/>

<many-to-one name="deviceType"
column="DEVICE_TYPE"
class="webdust.core.logic.DeviceType"
lazy="false"
cascade="save-update,delete"
not-null="true"/>
</class>

</hibernate-mapping>
```

Αρχείο *DeviceType.hbm.xml*:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="webdust.core.logic.DeviceType" table="DEVICE_TYPE">

<id name="ID" column="DEVICE_TYPE_ID">
<generator class="native"/>
</id>
<property name="Name" not-null="true"/>
<property name="Description"/>
<property name="URL"/>

</class>

</hibernate-mapping>
```

Επίσης πρέπει να ορίσουμε ένα αρχείο με τα χαρακτηριστικά της βάσης δεδομένων που χρησιμοποιούμε. Στην περίπτωσή μας χρησιμοποιούμε τη HSQLDB ως βάση δεδομένων και το αρχείο που περιέχει όλα τα χαρακτηριστικά είναι το *hibernate.cfg.xml*.

Γράφοντας τα παραπάνω έχουμε κάνει τα εξής:

- Έχουμε αντιστοιχίσει τα αντικείμενα Device και DeviceType με τους πίνακες DEVICE και DEVICE_TYPE της βάσης δεδομένων (με το class των XML αρχείων).
- Έχουμε ορίσει τα πρωτεύοντα κλειδιά (με το id των XML αρχείων).
- Έχουμε αντιστοιχίσει τα χαρακτηριστικά των αντικειμένων (Name, Description, URL) με τις αντίστοιχες στήλες των σχεσιακών πινάκων (NAME, DESCRIPTION, URL) (με το property των XML αρχείων).
- Έχουμε ορίσει τη συσχέτιση (association) μεταξύ των δύο πινάκων (με τα many-to-one, one-to-many και set των XML αρχείων).
- Έχουμε ορίσει τη βάση δεδομένων που θα χρησιμοποιήσει το Hibernate (το αρχείο αυτό τοποθετείται στο classpath της εφαρμογής μας και το Hibernate το βρίσκει αυτόματα).

Έχοντας ορίσει όλα αυτά μπορούμε να αρχίσουμε το Hibernate και να αναφερθούμε στα αντικείμενά μας όπως περιγράφεται στη συνέχεια:

Παράδειγμα χρήσης:

```
Session session = HibernateUtil.getSession(); // Όρισε ένα νέο Session
Transaction tx = session.beginTransaction(); // Όρισε μια καινούργια Transaction

Device x = new Device(); // Όρισε μια νέα συσκευή
x.Name = "Sensor1"; // Δώσε όνομα στη συσκευή
x.Description = "This is the first device"; // Δώσε περιγραφή
x.URL = "http://www.firstdevice.org"; // Δώσε ένα URL

session.save(x); // Αποθήκευσε τη νέα συσκευή

tx.commit(); // Κάνε commit την αλλαγή
if (session.isOpen()) session.close(); // Τερμάτισε το Session
```

Στο παράδειγμα αυτό αποθηκεύσαμε στη βάση δεδομένων ένα Device αντικείμενο που ορίσαμε. Παρατηρήστε ότι σε όλη την έκταση του παραδείγματος δε χρησιμοποιήσαμε καθόλου SQL κώδικα, πράγμα που επιτρέπει το διαχωρισμό του logic της εφαρμογής απο τη βάση δεδομένων. Ο προγραμματιστής μπορεί ετσι να αφοσιωθεί στην κωδικοποίηση της εφαρμογής χωρίς να συνδέει στενά τον κώδικα της εφαρμογής με τον κώδικα που χρειάζεται για την τροποποίηση των δεδομένων στη βάση. Επίσης, οι αλλαγές στη βάση δεδομένων είναι ευκολότερο να αντιμετωπιστούν στα XML αρχεία των αντιστοιχίσεων παρα οπουδήποτε μέσα στον κώδικα. Ακόμα, χρειάζονται πολύ λιγότερες γραμμές κώδικα για τις βασικές τροποποιήσεις των δεδομένων (CRUD – Create Read Update Delete) στη βάση δεδομένων. Αυτά, σε μεγάλης έκτασης εφαρμογές, αποδυκνύονται πολύτιμα τόσο όσον αφορά το χρόνο ανάπτυξης της εφαρμογής όσο και στην αποσφαλμάτωση του κώδικα κατα την ανάπτυξη της εφαρμογής.

Το παράδειγμα που περιγράψαμε είναι πολύ απλό και ενδεικτικό της χρήσης του Hibernate. Το Hibernate υποστηρίζει πολύπλοκες συσχετίσεις μεταξύ αντικειμένων όπως συσχετίσεις (associations), πολυμορφισμό (polymorphism) και κληρονομικότητα (inheritance) και τις κυριότερες βάσεις δεδομένων (Oracle, MySQL, PostgreSQL, Microsoft SQL Server, DB2, Sybase κτλ.). Επίσης μπορεί να χρησιμοποιηθεί τόσο σε unmanaged (π.χ. το παράδειγμα που αναφέραμε) όσο και σε managed application περιβάλλοντα (π.χ. JBoss).

Κεφάλαιο 5

Επίπεδο Παρουσίασης

Το επίπεδο της Παρουσίασης είναι το τελευταίο από τα υπόλοιπα επίπεδα που αναλύσαμε. Αυτό σε καμία περίπτωση δε σημαίνει οτι είναι το λιγότερο σημαντικό. Κάθε άλλο, ο σωστός σχεδιασμός αυτού του επιπέδου είναι πολύ σημαντικός γιατί στην πραγματικότητα το interface της εφαρμογής είναι αυτό που θα αντικρύσει ο χρήστης και πρέπει να είναι τόσο ελκυστικό όσο και λειτουργικό, ουσιαστικό και να καλύπτει όλες τις δυνατότητες του συστήματος Webdust2. Ακόμα και αν ο σχεδιασμός ενός συστήματος έχει γίνει με τον καλυτερο τρόπο, αν ο σχεδιασμός του επιπέδου παρουσίασης δεν είναι ο κατάλληλος, τότε η εφαρμογή όχι απλά δεν είναι ελκυστική και λειτουργική για το χρήστη, αλλά και, ενδεχομένως, πολλες από τις δυνατότητες του συστήματος μπορούν να μείνουν αναξιοποίητες και να καταλλήξουν άχρηστες.

Ένα άλλο θέμα, που είναι σχετικό, είναι αυτό της αφαίρεσης. Στην παρούσα εφαρμογή δεν έχει δοθεί τόση σημασία στην άμεση εικόνα του interface προς το χρήστη. Με αυτό τι εννοούμε? Στην εφαρμογή Webdust2, με τον τρόπο που είναι σχεδιασμένη, δεν παρέχουμε έτοιμο interface αλλά έχουμε δημιουργήσει ένα πολύ δυνατό και σταθερό API που παρέχει όλα εκείνα τα αντικείμενα-κλασεις που χρειάζεται ένας σχεδιαστής Web ιστοσελίδων για να προχωρήσει στο σχεδιασμό της εξωτερικής εικόνας της εφαρμογής. Έδώ ακριβώς επαφύεται το θέμα της αφαίρεσης. Η εφαρμογή, έτσι, γίνεται πιο επεκτάσιμη, πιο αναβαθμίσιμη και επιτρέπει σε κάποιον ο οποίος είναι πιο εξειδικευμένος στον τομέα του design να αφοσιωθεί στο έργο του χωρίς να έχει πλήρη γνώση του υποκείμενου συστήματος(πραγμα που δεν τον ενδιαφέρει - πέρα από μια γενική εικόνα της λειτουργίας του).

Στο κεφάλαιο αυτό γίνεται μια αναφορά σε Java τεχνολογίες που χρησιμοποιήθηκαν και ακολουθεί η αναλυτική παρουσίαση και περιγραφή των οθονών που αποτελούν την βασική μορφή του UI της εφαρμογής Webdust2.

5.1 Java Τεχνολογίες

Η ανάπτυξη της εφαρμογής μας έγινε με βάση τη γλώσσα προγραμματισμού JAVA. Είναι λογικό λοιπόν να πούμε, οτι στραφήκαμε στην αναζήτηση Java τεχνολογιών και λύσεων για να χτίσουμε την εφαρμογή και να εμπλουτίσουμε το περιεχόμενό της. Ας δούμε λοιπόν παρακάτω ποιες είναι αυτές οι τεχνολογίες, ποια είναι τα χαρακτηριστικά τους και ποια η χρησιμότητά τους.

5.1.1 Java Servlets

Τα Servlets είναι η απάντηση στον CGI προγραμματισμό. Πρόκειται για προγράμματα που εκτελούνται στο διακομιστή(server) και δημιουργούν δυναμικό περιεχόμενο. Τα Servlets προτιμούνται από τον παραδοσιακό CGI προγραμματισμό για τους εξής λόγους:

- *Αποδοτικότητα* - Χρησιμοποιώντας τον CGI προγραμματισμό, κάθε φορά που λαμβάνεται μια HTTP αίτηση μια νέα διαδικασία έχει ξεκινήσει, η οποία μπορεί να οδηγήσει σε κακή απόδοση και σε θέματα επεκτασιμότητας. Χρησιμοποιώντας Servlets, η Java VM είναι πάντα σε λειτουργία, επομένως, ξεκινώντας ένα Servlet δημιουργεί ένα νήμα Java Servlet , σε αντίθεση με ένα σύστημα διαδικασία.
- *Ισχύς* - Τα Servlets προσφέρουν ισχύ που δεν μπορούν να προσφέρουν τα παραδοσιακά CGI. Τα Servlets σου επιτρέπουν να κάνεις πράγματα τα οποία είτε θα ήταν πολύ δύσκολα ή αδύνατα επειδή έχουν πρόσβαση σε όλη την οικογένεια των Java APIs. Πολύ εύκολα μοιράζονται δεδομένα και διατηρούν πληροφορίες, κάνουν εντοπισμό session και να δημιουργούν pool συνδέσεις με βάσεις δεδομένων ευκολότερα.
- *Ασφάλεια* - Τα Servlets μπορούν να τρέχουν από τη Servlet engine ένα περιοριστικό sandbox, παρόμια με το sandbox ενός φυλλομετρητή ιστού για applets. Αυτό βοηθά στην προστασία από κακόβουλα Servlets.
- *Κόστος* - Υπάρχουν πολλοί "δωρεάν" ή χαμηλού κόστους web servers που διατίθενται είτε για προσωπική χρήση ή για κυκλοφορία χαμηλού όγκου δεδομένων. Αν υπάρχει ήδη ένας web server, μπορεί να προστεθεί η Servlet τεχνολογία γρήγορα, εύκολα και το καλύτερο από όλα, φτηνά.
- *Δυνατότητα μεταφοράς* - Το Servlet API εκμεταλλεύεται την πλατφόρμα Java. Είναι ένα πολύ απλό API που υποστηρίζεται από το σύνολο σχεδόν όλων των web servers, έτσι ώστε τα Servlets να μπορούν να μετακινηθούν από πλατφόρμα σε πλατφόρμα, συνήθως χωρίς καμία απολύτως τροποποίηση.

Ένα Servlet είναι μια Java κλάση και, επομένως, πρέπει να εκτελεστεί από τη Java VM, η οποία ονομάζεται και Servlet engine. Ένα Servlet φορτώνεται από την μηχανή όταν καλείται και παραμένει εκεί μέχρι το να εκφορτωθεί εξωτερικά ή η μηχανή να κλείσει.

5.1.2 Java Server Pages

Η Java Server Pages ή JSP είναι η λύση της Sun για την ανάπτυξη δυναμικών web sites. Η JSP παρέχει εξαιρετική υποστήριξη για server-side scripting για τη δημιουργία Web εφαρμογών. Η JSP επιτρέπει στους προγραμματιστές την απευθείας εισαγωγή java κώδικα σε JSP αρχείο, γεγονός που καθιστά την διαδικασία ανάπτυξης πολύ απλή και τη συντήρηση πολύ εύκολη. Οι JSP σελίδες είναι αποδοτική, φορτώνει στη μνήμη των web servers κατά τη διάρκεια της παραλαβής της πρώτης αίτησης και οι επόμενες κλήσεις εξυπηρετούνται μέσα σε πολύ σύντομο χρονικό διάστημα.

Στο σημερινό περιβάλλον στους περισσότερους servers δικτυακών τόπων οι δυναμικές σελίδες βασίζονται σε αίτηση του χρήστη. Η βάση δεδομένων είναι ένας πολύ βολικός τρόπος για την αποθήκευση των δεδομένων των χρηστών και των άλλων πληροφοριών. Το JDBC παρέχει εξαιρετική δυνατότητα σύνδεσης σε βάση δεδομένων σε ετερογενές περιβάλλον βάσης δεδομένων. Χρησιμοποιώντας JDBC και JSP πολύ εύκολο να αναπτύξει web εφαρμογές που είναι σχετισμένες με βάση δεδομένων.

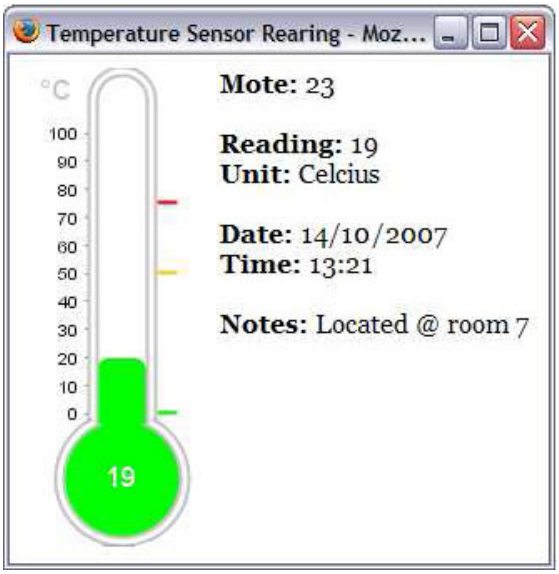
Η Java είναι γνωστή για το χαρακτηριστικό της ”write once, run anywhere”. Οι JSP σελίδες είναι ανεξάρτητες πλατφόρμας. Τα JSP αρχεία μπορούν να μεταφερθούν σε οποιαδήποτε πλατφόρμα.

5.1.3 JFreeChart

Το JFreeChart είναι μια ελεύθερη 100% Java βιβλιοθήκη διαγραμμάτων που καθιστά εύκολη για τους προγραμματιστές την απεικόνιση χαρτών των εφαρμογών τους σε επαγγελματική ποιότητα. Τα χαρακτηριστικά του JFreeChart περιλαμβάνουν:

- Ένα συμπαγές και καλά τεκμηριωμένο API, που υποστηρίζει ένα ευρύ φάσμα τύπων γραφημάτων.
- Ευέλικτο σχεδιασμό, ο οποίος μπορεί εύκολα να επεκταθεί, και που στοχεύει τόσο server-side όσο και client-side εφαρμογές.
- Υποστήριξη για πολλούς τύπους αρχείων εξόδου, συμπεριλαμβανομένων των Swing components, αρχείων εικόνας (συμπεριλαμβανομένων των PNG και JPEG), και μορφές αρχείων vector γραφικών (συμπεριλαμβανομένων των PDF, SVG και EPS).

- JFreeChart είναι "ανοικτού κώδικα" ή, πιο συγκεκριμένα, ελεύθερου λογισμικού. Διανέμεται σύμφωνα με τους όρους της GNU Lesser General Public License (LGPL), το οποίο επιτρέπει την χρήση σε ιδιόκτητες εφαρμογές.



Σχήμα 5.1: Παράδειγμα τιμής θερμοκρασίας ενός αισθητήρα με το JFreeChart

5.2 Σχεδιασμός UI

Για να γίνουμε πιο σαφής, στην εφαρμογή μας όπως έχουμε δει έως τώρα, έχουμε ορίσει μια σειρά από Java κλάσεις οντοτήτων(Βλέπε Παράγραφος 4.3) και χειριστών(Βλέπε Παράγραφος 4.5). Έχοντας έτσι δομημένο τον κώδικα της εφαρμογής μας μπορούμε και επιτυγχάνουμε ένα είδος αφαίρεσης. Έτσι, εδώ, στο τμήμα του σχεδιασμού, επικεντρώνουμε την προσοχή μας στο κομμάτι της σχεδίασης και όχι τόσο στον τρόπο με τον οποίο θα αντλήσουμε τα δεδομένα μέσα από τη βάση. Οι τεχνολογίες που αναφέρθηκαν ανωτέρω (Servlets, JSP, κλπ.) είναι αυτές οι οποίες χρησιμοποιήθηκαν για το σχεδιασμό του UI. Στα servlet, καθώς και στα JSP αρχεία, αρχικοποιούμε κλάσεις και δημιουργούμε στιγμιότυπα των ήδη ορισμένων στοιχείων(οντότητες, χειριστές, κλπ.)από το λογικό επίπεδο. Παρακάτω θα παρουσιάσουμε ένα σύνολο από οθόνες της εφαρμογής στις οποίες φαίνεται η λειτουγικότητα του συστήματός μας και παράλληλα παρέχεται ένα πολύ στοιχειώδες UI.

Ο σχεδιασμός του UI του Webdust2είναι στοιχειώδης. Είναι σημαντικό εδώ να δηλώσουμε οτι το σενάριο μιας νέας σχεδίασης είναι αρκούντως εφικτό μιας και η εφαρμογή μας είναι ανοικτού λογισμικού και πλήρως επεκτάσιμη. Η εξωτερική εμφάνιση της εφαρμογής δεν είναι ο κύριος στόχος αυτής της διπλωματικής. Από την άλλη, η

πλήρης παρουσίαση των δυνάτοτήτων του συστήματος είναι επιτακτική και αυτό γίνεται στο παρόν κεφάλαιο.

5.2.1 Χειρισμός Coordinate

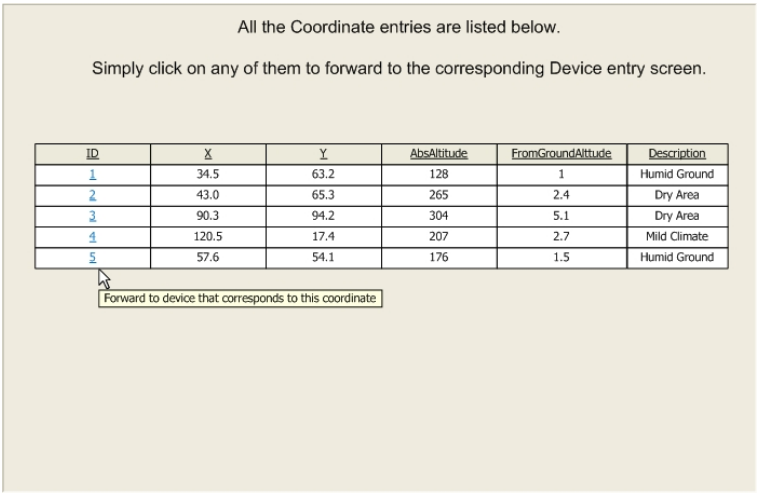
Εδώ παρουσιάζουμε τις οθόνες που σχετίζονται με το χειρισμό της Coordinate οντότητας.

5.2.1.1 Κατάταξη όλων των Coordinate εγγραφών

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.2) απαριθμούνται όλες οι Coordinate εγγραφές. Στην οθόνη αυτή γίνεται χρήση της Coordinate οντότητας, η κλάση CoordinateManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- Επιλογή Coordinate: Hyperlink. Ο χρήστης απλά πατάει ”κλικ” στο ID της επιλογής του για να προωθηθεί στην οθόνη με τις λεπτομέρειες της Coordinate εγγραφής.
- Ταξινόμηση κατά: Hyperlink. Ο χρήστης μπορεί να κάνει ένα απλό ”κλικ” σε μία από τις στήλες του πίνακα για να κάνει ταξινόμηση(η ταξινόμηση μπορεί να γίνει κατά ID, X, Y, κτλ..).



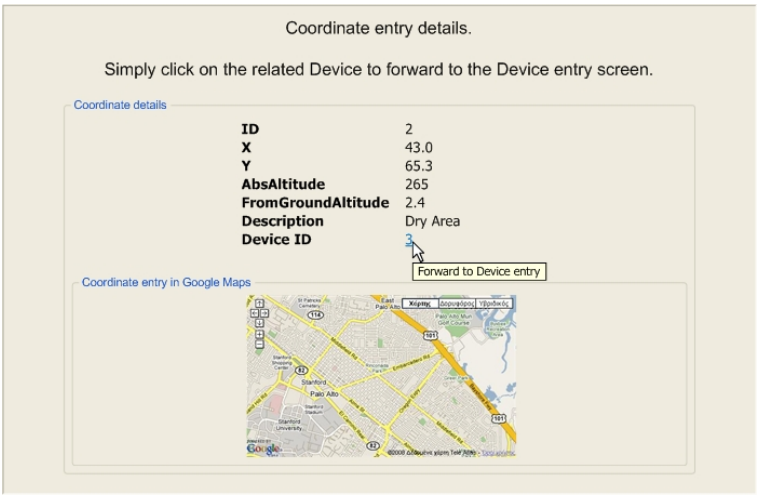
Σχήμα 5.2: Λίστα όλων των Coordinate εγγραφών

5.2.1.2 Λεπτομέρειες της εγγραφής Coordinate

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.3) παρουσιάζονται όλες οι λεπτομέρειες της Coordinate εγγραφής. Στην οθόνη αυτή γίνεται χρήση της Coordinate οντότητας, η κλάση CoordinateManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- Επιλογή Device: Hyperlink. Ο χρήστης απλά πατάει ″κλικ″ στο ID της επιλογής του για να προωθηθεί στην οθόνη με τις λεπτομέρειες της Coordinate εγγραφής.
- Παρουσίαση στο Google Maps: Google Maps. Ο χρήστης μπορεί να παρακολουθήσει την περιοχή γύρω απο τις συγκεκριμένες συντεταγμένες μέσω του Google Maps interface. Μπορεί να κάνει ζοομ και να διαλέξει πως θέλει να βλέπει το terrain(προεπισκόπηση χάρτη, δορυφόρου ή υβριδική).



Σχήμα 5.3: Λεπτομέρειες της εγγραφής Coordinate

5.2.1.3 Αναζήτηση εγγραφών Coordinate σε περιοχή ορθογωνίου

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.4) ο χρήστης μπορεί να αναζητήσει για όλες τις Coordinate εγγραφές που είνα μέσα στη δοσμένη περιοχή ορθογωνίου. Στην οθόνη αυτή γίνεται χρήση της Coordinate οντότητας, η κλάση CoordinateManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- Top Left X bound: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια τιμή(Double) η οποία αναπαριστά την X συντεταγμένη της πάνω αριστερά γωνίας της ορθογώνιας περιοχής.

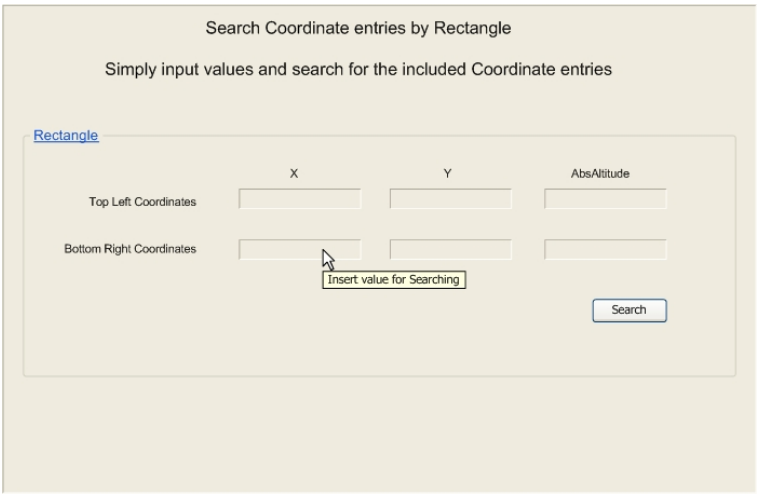
- Top Left Y bound: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια τιμή(Double) η οποία αναπαριστά την Y συντεταγμένη της πάνω αριστερά γωνίας της ορθογώνιας περιοχής.
- Top Left AbsAltitude bound: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια τιμή(Double) η οποία αναπαριστά το απόλυτο ύψος του σημείου της πάνω αριστερά γωνίας της ορθογώνιας περιοχής.
- Bottom Right X bound: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια τιμή(Double) η οποία αναπαριστά την X συντεταγμένη της κάτω δεξιά γωνίας της ορθογώνιας περιοχής.
- Bottom Right Y bound: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια τιμή(Double) η οποία αναπαριστά την Y συντεταγμένη της κάτω δεξιά γωνίας της ορθογώνιας περιοχής.
- Bottom Right AbsAltitude bound: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια τιμή(Double) η οποία αναπαριστά το απόλυτο ύψος του σημείου της κάτω δεξιά γωνίας της ορθογώνιας περιοχής.
- Search: Submit button. Το κουμπί αυτό υποβάλλει τα εισαγόμενα στη φόρμα δεδομένα

Έλεγχοι

- Έλεγχος πληρότητας των πεδίων: Ενεργοποιείται με το πάτημα του κουμπιού "Search". Ελέγχει κάθε ένα πεδίο. Άν, έστω και ένα είναι κενό, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένης εισαγωγής: Ενεργοποιείται με το πάτημα του κουμπιού "Search". Ελέγχει αν τα δεδομένα υποβλήθηκαν επιτυχώς. Αν ναι, επιστρέφεται στο χρήστη μια οθόνη με μια λίστα όλων των Coordinate εγγραφών που είναι μέσα στα δοσμένα όρια, διαφορετικα, ο χρήστης προωθείται σε μία οθόνη με το μήνυμα: 'An error occured, please try again!'.

5.2.2 Χειρισμός Device

Εδώ παρουσιάζουμε τις οθόνες που σχετίζονται με το χειρισμό της Device οντότητας.



Σχήμα 5.4: Coordinate εγγραφές που περιλαμβάνονται στην ορθογώνια περιοχή

5.2.2.1 Κατάταξη όλων Device(διαγραφή επίσης)

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.5) απαριθμούνται όλες οι Device εγγραφές. Υπάρχει δυνατότητα διαγραφής εδώ. Στην οθόνη αυτή γίνεται χρήση της Device οντότητας, η κλάση DeviceManager έχει τα εξής χαρακτηριστικά:

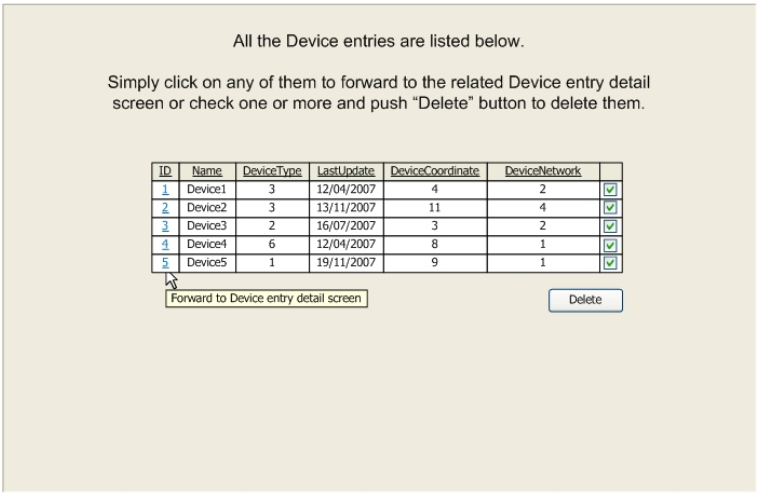
Λειτουργίες

- Επιλογή Device: Hyperlink. Ο χρήστης απλά πατάει ”κλικ” στο ID της επιλογής του για να προωθηθεί στην οθόνη με τις λεπτομέρειες της Device εγγραφής.
- Διαγραφή Device: CheckBox. Ο χρήστης ”τσεκάρει” απλά ένα ή περισσότερα check-box για να διαγράψει εγγραφές τύπου Device.
- Delete: Submit Button. Το κουμπί αυτό στέλνει το αίτημα για διαγραφη
- Ταξινόμηση: Hyperlink. Ο χρήστης μπορεί να κάνει ένα απλό ”κλικ” σε μία από τις στήλες του πίνακα για να κάνει ταξινόμηση(η ταξινόμηση μπορεί να γίνει κατά ID, Name, LastUpdate, κτλ..).

Έλεγχοι

- Έλεγχος των ”check-boxes”: Ενεργοποιείται με το πάτημα του κουμπιού ”Delete”. Ελέγχει όλα τα ”check-boxes”. Εάν κάθε ένα από αυτά είναι αμαρκάριστο, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένου αιτήματος: Ενεργοποιείται με το πάτημα του κουμπιού ”Delete”. Ελέγχει εάν το αίτημα ήταν επιτυχές. Αν ναι, ο χρήστης οδηγείται σε

μια οθόνη με το μήνυμα: ”The deletion was successful”, διαφορετικά ο χρήστης προωθείται σε μια οθόνη με το μήνυμα: ’An error occured, please try again!’.



Σχήμα 5.5: Λίστα όλων των Device εγγραφών

5.2.2.2 Λεπτομέρειες της εγγραφής Device

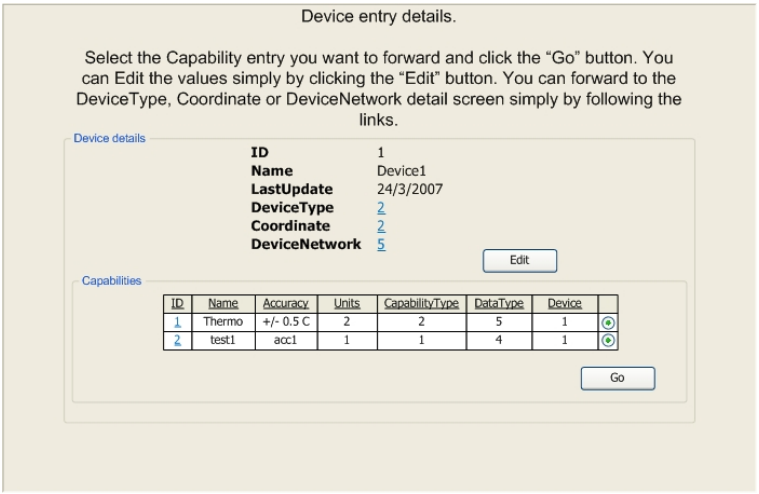
Σε αυτή την οθόνη(Βλέπε Σχήμα 5.6) παρουσιάζονται όλες οι λεπτομέρειες της Device εγγραφής. Στην οθόνη αυτή γίνεται χρήση της Device οντότητας, η κλάση DeviceManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- Επιλογή Capability: Radio button. Ο χρήστης μπορεί να διαλέξει κάποια Capability εγγραφή για να προωθηθεί.
- Επιλογή DeviceType: HyperLink. Ο χρήστης μπορεί να προωθηθεί στην οθόνη της DeviceType εγγραφής.
- Επιλογή Coordinate: HyperLink. Ο χρήστης μπορεί να προωθηθεί στην οθόνη της Coordinate εγγραφής.
- Επιλογή DeviceNetwork: HyperLink. Ο χρήστης μπορεί να προωθηθεί στην οθόνη της DeviceNetwork εγγραφής.
- Go: Submit button. Αυτό το κουμπί προωθεί το χρήστη στην οθόνη της Capability εγγραφής.
- Edit: Submit button. Αυτό το κουμπί προωθεί το χρήστη στην οθόνη επεξεργασίας της Device εγγραφής.

Έλεγχοι

- Έλεγχος επιλογής: Ενεργοποιείται με το πάτημα του κουμπιού "Go". Ελέγχει εάν έχει επιλεγεί κάποιο "radio button". Αν όχι, ειδοποιεί το χρήστη.



Σχήμα 5.6: Λεπτομέρειες της εγγραφής Device

5.2.2.3 Προσθήκη εγγραφής Device

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.7) μπορεί να προσθέση μια νέα Device εγγραφή. Στην οθόνη αυτή γίνεται χρήση της Device οντότητας, η κλάση DeviceManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- Name: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει ένα όνομα για τη νέα Device εγγραφή που θέλει να προσθέσει. Το πεδίο είναι υποχρεωτικό.
- LastUpdate: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια τιμή ημερομηνίας για τη νέα Device εγγραφή. Το πεδίο είναι προαιρετικό.
- DeviceType: Drop-down Menu. Ο χρήστης επιλέγει μια DeviceType τιμή από τις τιμές που έχουν επιστραφεί στο "drop-down" μενού. Η επιλογή είναι υποχρεωτική .
- DeviceNetwork: Drop-down Menu. Ο χρήστης επιλέγει μια DeviceNetwork τιμή από τις τιμές που έχουν επιστραφεί στο "drop-down" μενού. This option is arbitral.
- Capability: Drop-down Menu. Ο χρήστης επιλέγει μια Capability τιμή από τις τιμές που έχουν επιστραφεί στο "drop-down" μενού. Η επιλογή είναι υποχρεωτική .

- X: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει την τιμή X(double) για τη νέα Device εγγραφή που θέλει να προσθέσει. Το πεδίο είναι υποχρεωτικό.
- Y: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει την τιμή Y(double) για τη νέα Device εγγραφή που θέλει να προσθέσει. Το πεδίο είναι υποχρεωτικό.
- AbsAltitude: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει την τιμή AbsAltitude(double) για τη νέα Device εγγραφή που θέλει να προσθέσει. Το πεδίο είναι προαιρετικό.
- FromGroundAltitude: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει την τιμή FromGroundAltitude(double) για τη νέα Device εγγραφή που θέλει να προσθέσει. Το πεδίο είναι προαιρετικό.
- Description: Text field. Σε αυτό το πεδίο ο χρήστης εισάγει την τιμή Description(text) για τη νέα Device εγγραφή που θέλει να προσθέσει. Το πεδίο είναι προαιρετικό.
- Submit: Submit button. Το κουμπί αυτό υποβάλλει τα εισαγόμενα στη φόρμα δεδομένα
- Clear: Clear button. Το κουμπί αυτό καθαρίζει όλα τα πεδία ώστε ο χρήστης να μπορεί από την αρχή να τα συμπληρώσει.

Έλεγχοι

- Έλεγχος πληρότητας των πεδίων: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει κάθε υποχρεωτικό πεδίο. Αν υπάρχει τουλάχιστον ένα κενό, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένης εισαγωγής: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει αν τα δεδομένα υποβλήθηκαν επιτυχώς. Αν ναι, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'Entry succesfully added.', διαφορετικά, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'An error occured, please try again!'.

5.2.2.4 Επεξεργασία εγγραφής Device

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.8) ο χρήστης μπορεί να επεξεργαστεί και να αλλάξει την τιμή της Device εγγραφής. Στην οθόνη αυτή γίνεται χρήση της Device οντότητας, η κλάση DeviceManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

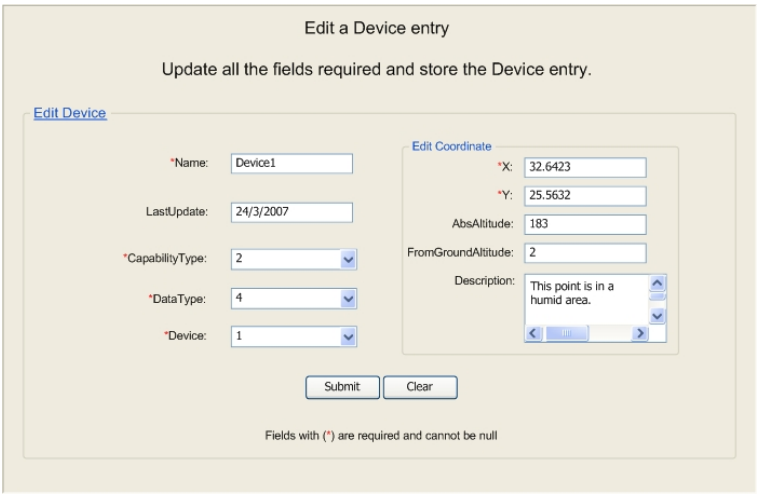
Σχήμα 5.7: Προσθήκη εγγραφής Device

- Name: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει ένα όνομα για την DeviceType εγγραφή που θέλει να ενημερώσει. Το πεδίο είναι υποχρεωτικό.
- LastUpdate: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια τιμή ημερομηνίας για την Device εγγραφή που θέλει να ενημερώσει. Το πεδίο είναι προαιρετικό.
- DeviceType: Drop-down Menu. Ο χρήστης επιλέγει μια DeviceType τιμή από τις τιμές που έχουν επιστραφεί στο ”drop-down” μενού. Η επιλογή είναι υποχρεωτική .
- DeviceNetwork: Drop-down Menu. Ο χρήστης επιλέγει μια DeviceNetwork τιμή από τις τιμές που έχουν επιστραφεί στο ”drop-down” μενού. This option is arbitral.
- Capability: Drop-down Menu. Ο χρήστης επιλέγει μια Device από τις τιμές που έχουν επιστραφεί στο ”drop-down” μενού. Η επιλογή είναι υποχρεωτική .
- X: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει την τιμή X(double) για την Device εγγραφή που θέλει να ενημερώσει. Το πεδίο είναι υποχρεωτικό.
- Y: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει την τιμή Y(double) για την Device εγγραφή που θέλει να ενημερώσει. Το πεδίο είναι υποχρεωτικό.
- AbsAltitude: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει την τιμή AbsAltitude(double) για την Device εγγραφή που θέλει να ενημερώσει. Το πεδίο είναι προαιρετικό.
- FromGroundAltitude: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει την τιμή FromGroundAltitude(double) για την Device εγγραφή που θέλει να ενημερώσει. Το πεδίο είναι προαιρετικό.

- Description: Text field. Σε αυτό το πεδίο ο χρήστης εισάγει την τιμή Description(text) για την Device εγγραφή που θέλει να ενημερώσει. Το πεδίο είναι προαιρετικό.
- Submit: Submit button. Το κουμπί αυτό υποβάλλει τα εισαγόμενα στη φόρμα δεδομένα
- Clear: Clear button. Το κουμπί αυτό καθαρίζει όλα τα πεδία ώστε ο χρήστης να μπορεί από την αρχή να τα συμπληρώσει.

Έλεγχοι

- Έλεγχος πληρότητας των πεδίων: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει κάθε υποχρεωτικό πεδίο. Αν υπάρχει τουλάχιστον ένα κενό, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένης εισαγωγής: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει αν τα δεδομένα υποβλήθηκαν επιτυχώς. Αν ναι, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'Entry successfully updated.', διαφορετικά, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'An error occured, please try again!'.



Σχήμα 5.8: Επεξεργασία εγγραφής Device

5.2.2.5 Αναζήτηση εγγραφών Device κατά όνομα

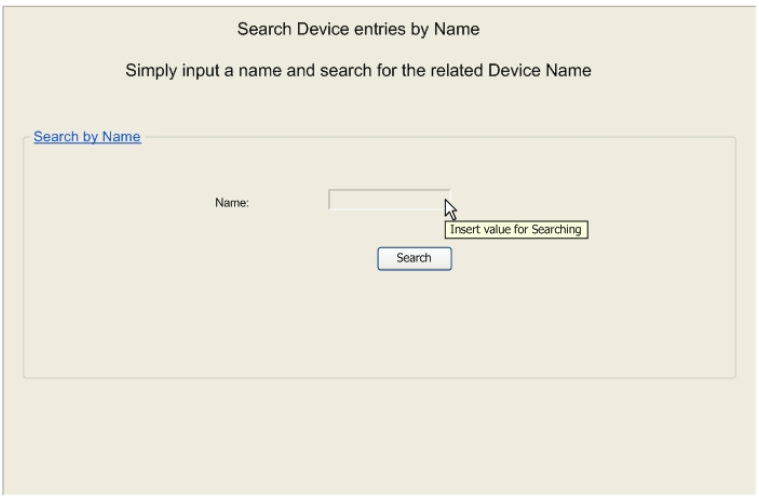
Σε αυτή την οθόνη(Βλέπε Σχήμα 5.9) ο χρήστης μπορεί να αναζητήσει για όλες τις Device εγγραφές των οποίων το όνομα(Name) περιέχει τη δοσμένη από το χρήστη συμβολοσειρά. Στην οθόνη αυτή γίνεται χρήση της Device οντότητας, η κλάση DeviceManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- Name: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει ένα όνομα(String).
- Search: Submit button. Το κουμπί αυτό υποβάλλει τα εισαγόμενα στη φόρμα δεδομένα.

Έλεγχοι

- Έλεγχος πληρότητας των πεδίων: Ενεργοποιείται με το πάτημα του κουμπιού "Search". Ελέγχει το πεδίο. Αν είναι κενό, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένης εισαγωγής: Ενεργοποιείται με το πάτημα του κουμπιού "Search". Ελέγχει αν τα δεδομένα υποβλήθηκαν επιτυχώς. Αν ναι, ο χρηστης οδηγείται σε μια οθόνη με όλες τις Device εγγραφές που περιέχουν το συγκεκριμένο όνομα, διαφορετικά, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'An error occured, please try again!'.



Σχήμα 5.9: Αναζήτηση εγγραφών Device κατά όνομα

5.2.3 Χειρισμός DeviceType

Εδώ παρουσιάζουμε τις οθόνες που σχετίζονται με το χειρισμό της DeviceType οντότητας.

5.2.3.1 Κατάταξη όλων DeviceType(διαγραφή επίσης)

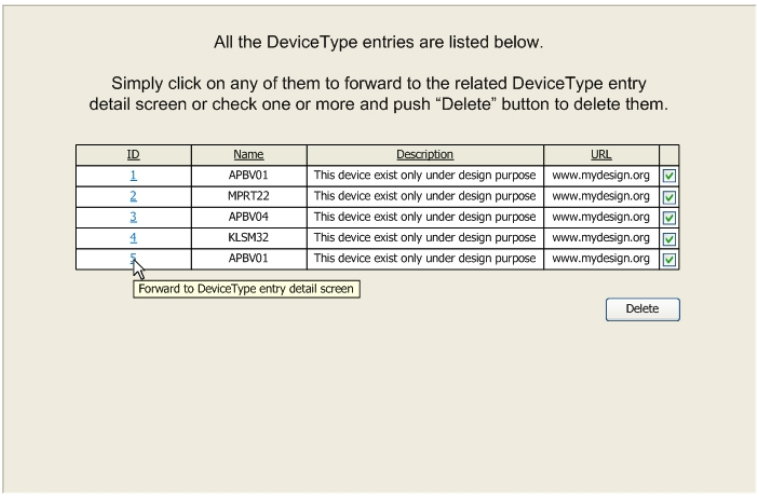
Σε αυτή την οθόνη(Βλέπε Σχήμα 5.10) απαριθμούνται όλες οι DeviceType εγγραφές. Υπάρχει δυνατότητα διαγραφής εδώ. Στην οθόνη αυτή γίνεται χρήση της DeviceType οντότητας, η κλάση DeviceTypeManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- DeviceType Επιλογή: Hyperlink. Ο χρήστης απλά πατάει ”κλικ” στο ID της επιλογής του για να προωθηθεί στην οθόνη με τις λεπτομέρειες της DeviceType εγγραφής.
- DeviceType Deletion: CheckBox. Ο χρήστης ”τσεκάρει” απλά ένα ή περισσότερα check-box για να διαγράψει εγγραφές τύπου DeviceType.
- Delete: Submit Button. Το κουμπί αυτό στέλνει το αίτημα για διαγραφη
- Ταξινόμηση: Hyperlink. Ο χρήστης μπορεί να κάνει ένα απλό ”κλικ” σε μία από τις στήλες του πίνακα για να κάνει ταξινόμηση(η ταξινόμηση μπορεί να γίνει κατά ID, Name, Description, κτλ..).

Έλεγχοι

- Έλεγχος των ”check-boxes”: Ενεργοποιείται με το πάτημα του κουμπιού ”Delete”. Ελέγχει όλα τα ”check-boxes”. Εάν κάθε ένα από αυτά είναι αμαρκάριστο, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένου αιτήματος: Ενεργοποιείται με το πάτημα του κουμπιού ”Delete”. Ελέγχει εάν το αίτημα ήταν επιτυχές. Αν ναι, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα: ”The deletion was successful”, διαφορετικά ο χρήστης προωθείται σε μια οθόνη με το μήνυμα: ’An error occured, please try again!’.



Σχήμα 5.10: Λίστα όλων των DeviceType εγγραφών

5.2.3.2 Λεπτομέρειες της εγγραφής DeviceType

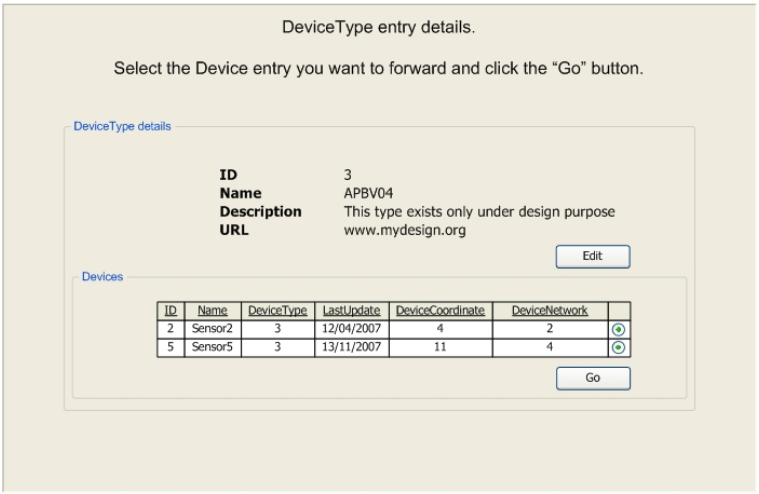
Σε αυτή την οθόνη(Βλέπε Σχήμα 5.11) παρουσιάζονται όλες οι λεπτομέρειες της DeviceType εγγραφής. Στην οθόνη αυτή γίνεται χρήση της DeviceType οντότητας, η κλάση DeviceTypeManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- Device Επιλογή: Radio button. Ο χρήστης μπορεί να διαλέξει κάποια Device εγγραφή για να προωθηθεί.
- Go: Submit button. Αυτό το κουμπί προωθεί το χρήστη στην οθόνη της Device εγγραφής.
- Edit: Submit button. Αυτό το κουμπί προωθεί το χρήστη στην οθόνη επεξεργασίας της DeviceType εγγραφής.

Έλεγχοι

- Έλεγχος επιλογής: Ενεργοποιείται με το πάτημα του κουμπιού "Go". Ελέγχει εάν έχει επιλεγεί κάποιο "radio button". Αν όχι, ειδοποιεί το χρήστη.



Σχήμα 5.11: Λεπτομέρειες της εγγραφής DeviceType

5.2.3.3 Αναζήτηση εγγραφών DeviceType κατά όνομα

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.12) ο χρήστης μπορεί να αναζητήσει για όλες τις DeviceType εγγραφές των οποίων το όνομα(Name) περιέχει τη δοσμένη από το χρήστη

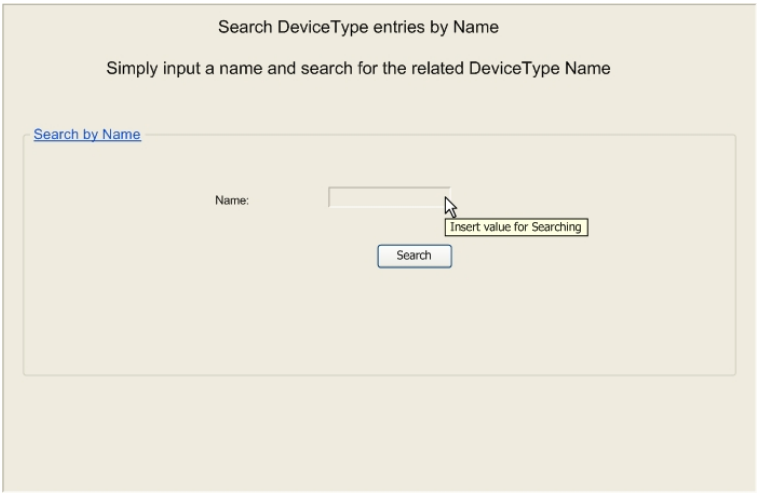
συμβολοσειρά. Στην οθόνη αυτή γίνεται χρήση της DeviceType οντότητας, η κλάση DeviceTypeManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- Name: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει ένα όνομα(String).
- Search: Submit button. Το κουμπί αυτό υποβάλλει τα εισαγόμενα στη φόρμα δεδομένα.

Έλεγχοι

- Έλεγχος πληρότητας των πεδίων: Ενεργοποιείται με το πάτημα του κουμπιού "Search". Ελέγχει το πεδίο. Αν είναι κενό, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένης εισαγωγής: Ενεργοποιείται με το πάτημα του κουμπιού "Search". Ελέγχει αν τα δεδομένα υποβλήθηκαν επιτυχώς. Αν ναι, ο χρηστης οδηγείται σε μια οθόνη με όλες τις DeviceType εγγραφές που περιέχουν το δοσμένο όνομα, διαφορετικά, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'An error occured, please try again!'.



Σχήμα 5.12: Αναζήτηση εγγραφών DeviceType κατά όνομα

5.2.3.4 Αναζήτηση εγγραφών DeviceType ως προς περιγραφή

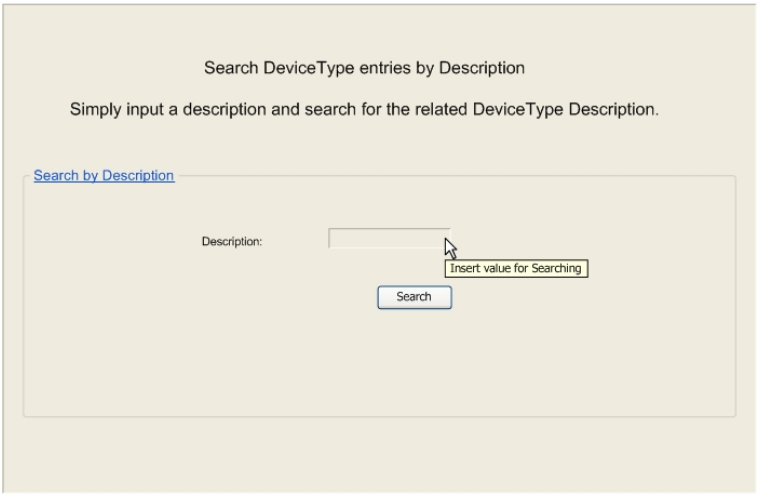
Σε αυτή την οθόνη(Βλέπε Σχήμα 5.13) ο χρήστης μπορεί να αναζητήσει για όλες τις Device εγγραφές των οποίων η περιγραφή(Description) περιέχει τη δοσμένη από το χρήστη συμβολοσειρά. Στην οθόνη αυτή γίνεται χρήση της DeviceType οντότητας, η κλάση DeviceTypeManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- Description: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια περιγραφή(String).
- Search: Submit button. Το κουμπί αυτό υποβάλλει τα εισαγόμενα στη φόρμα δεδομένα.

Έλεγχοι

- Έλεγχος πληρότητας των πεδίων: Ενεργοποιείται με το πάτημα του κουμπιού "Search". Ελέγχει το πεδίο. Αν είναι κενό, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένης εισαγωγής: Ενεργοποιείται με το πάτημα του κουμπιού "Search". Ελέγχει αν τα δεδομένα υποβλήθηκαν επιτυχώς. Αν ναι, ο χρηστης οδηγείται σε μια οθόνη με όλες τις DeviceType εγγραφές που περιέχουν τη δοσμένη περιγραφή, διαφορετικά, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'An error occured, please try again!'.



Σχήμα 5.13: Αναζήτηση εγγραφών DeviceType ως προς περιγραφή

5.2.3.5 Προσθήκη εγγραφής DeviceType

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.14) ο χρήστης μπορεί να προσθέση μια νέα DeviceType εγγραφή. Στην οθόνη αυτή γίνεται χρήση της DeviceType οντότητας, η κλάση DeviceTypeManager έχει τα εξής χαρακτηριστικά:

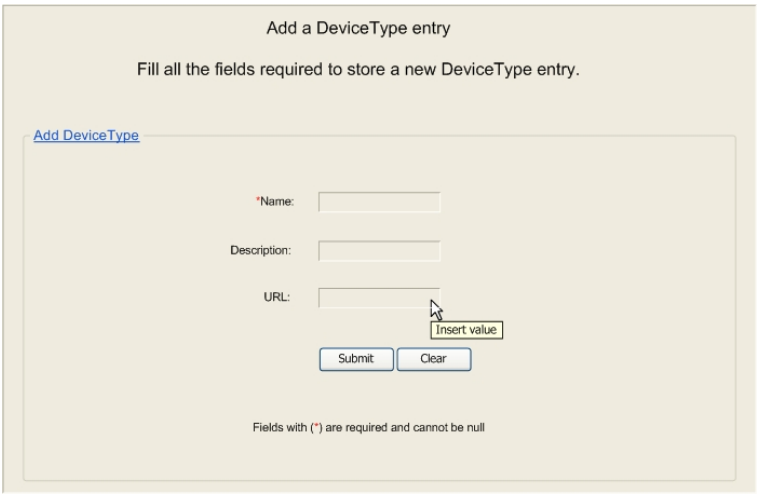
Λειτουργίες

- Name: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει ένα όνομα για τη νέα DeviceType εγγραφή που θέλει να προσθέσει. Το πεδίο είναι υποχρεωτικό.

- Description: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια περιγραφή για τη νέα DeviceType εγγραφή. Το πεδίο είναι προαιρετικό.
- URL: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει a url για τη νέα DeviceType εγγραφή. Αυτό το πεδίο είναι προαιρετικό.
- Submit: Submit button. Το κουμπί αυτό υποβάλλει τα εισαγόμενα στη φόρμα δεδομένα.
- Clear: Clear button. Το κουμπί αυτό καθαρίζει όλα τα πεδία ώστε ο χρήστης να μπορεί από την αρχή να τα συμπληρώσει.

Έλεγχοι

- Έλεγχος πληρότητας των πεδίων: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει κάθε υποχρεωτικό πεδίο. Αν υπάρχει τουλάχιστον ένα κενό, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένης εισαγωγής: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει αν τα δεδομένα υποβλήθηκαν επιτυχώς. Αν ναι, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'Entry succesfully added.', διαφορετικά, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'An error occured, please try again!'.



Σχήμα 5.14: Προσθήκη εγγραφής DeviceType

5.2.3.6 Επεξεργασία εγγραφής DeviceType

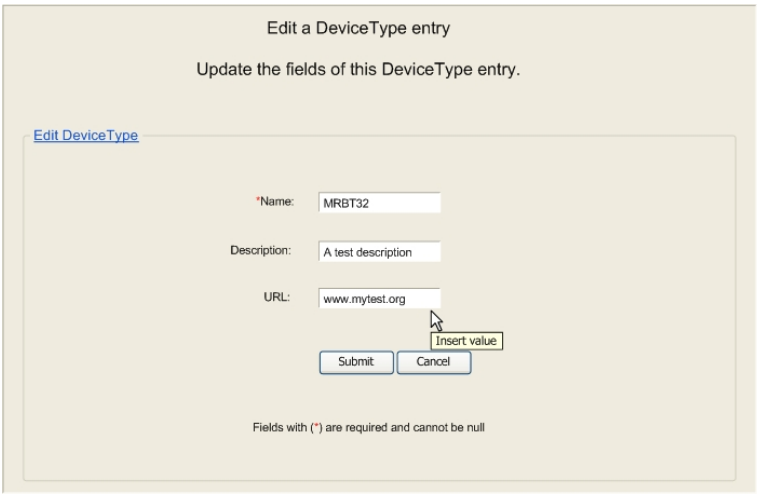
Σε αυτή την οθόνη(Βλέπε Σχήμα 5.15) ο χρήστης μπορεί να επεξεργαστεί και να αλλάξει την τιμή της DeviceType εγγραφής. Στην οθόνη αυτή γίνεται χρήση της DeviceType οντότητας, η κλάση DeviceTypeManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- Name: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει ένα όνομα για την DeviceType εγγραφή που θέλει να ενημερώσει. Το πεδίο είναι υποχρεωτικό.
- Description: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια περιγραφή για την DeviceType εγγραφή που θέλει να ενημερώσει. Το πεδίο είναι προαιρετικό.
- URL: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει ένα url για την DeviceType εγγραφή που θέλει να ενημερώσει. Το πεδίο είναι προαιρετικό.
- Submit: Submit button. Το κουμπί αυτό υποβάλλει τα εισαγόμενα στη φόρμα δεδομένα.
- Clear: Clear button. Το κουμπί αυτό καθαρίζει όλα τα πεδία ώστε ο χρήστης να μπορεί από την αρχή να τα συμπληρώσει.

Έλεγχοι

- Έλεγχος πληρότητας των πεδίων: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει κάθε υποχρεωτικό πεδίο. Άν, έστω και ένα είναι κενό, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένης εισαγωγής: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει αν τα δεδομένα υποβλήθηκαν επιτυχώς. Αν ναι, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'Entry succesfully updated.', διαφορετικά, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'An error occured, please try again!'.



Σχήμα 5.15: Επεξεργασία εγγραφής DeviceType

5.2.4 Χειρισμός Capability

Εδώ παρουσιάζουμε τις οθόνες που σχετίζονται με το χειρισμό της Capability οντότητας.

5.2.4.1 Κατάταξη όλων των Capability εγγραφών(διαγραφή επίσης)

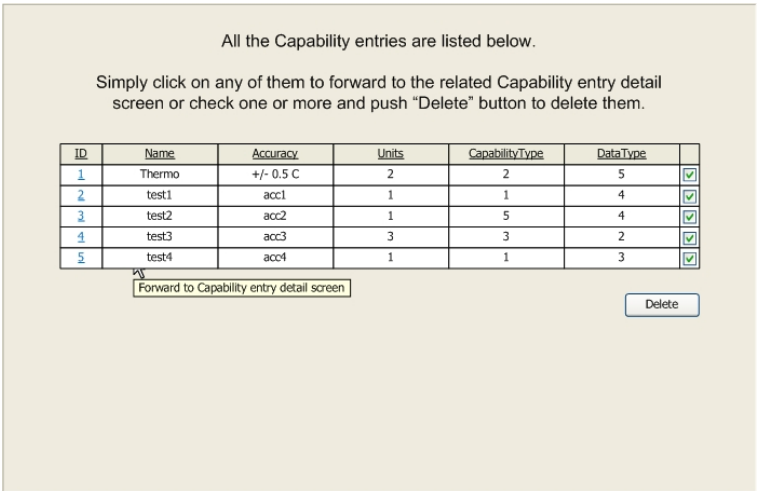
Σε αυτή την οθόνη(Βλέπε Σχήμα 5.16) απαριθμούνται όλες οι Capability εγγραφές. Υπάρχει δυνατότητα διαγραφής εδώ. Στην οθόνη αυτή γίνεται χρήση της Capability οντότητας, η κλάση CapabilityManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- Capability Επιλογή: Hyperlink. Ο χρήστης απλά πατάει ”κλικ” στο ID της επιλογής του για να προωθηθεί στην οθόνη με τις λεπτομέρειες της Capability εγγραφής.
- Capability Deletion: CheckBox. Ο χρήστης ”τσεκάρει” απλά ένα ή περισσότερα check-box για να διαγράψει εγγραφές τύπου Capability.
- Delete: Submit Button. Το κουμπί αυτό στέλνει το αίτημα για διαγραφή.
- Ταξινόμηση: Hyperlink. Ο χρήστης μπορεί να κάνει ένα απλό ”κλικ” σε μία από τις στήλες του πίνακα για να κάνει ταξινόμηση(η ταξινόμηση μπορεί να γίνει κατά ID, Name, Accuracy, κτλ..).

Έλεγχοι

- Έλεγχος των ”check-boxes”: Ενεργοποιείται με το πάτημα του κουμπιού ”Delete”. Ελέγχει όλα τα ”check-boxes”. Εάν κάθε ένα από αυτά είναι αμαρκάριστο, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένου αιτήματος: Ενεργοποιείται με το πάτημα του κουμπιού ”Delete”. Ελέγχει εάν το αίτημα ήταν επιτυχές. Αν ναι, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα: ”The deletion was successful”, διαφορετικά ο χρήστης προωθείται σε μια οθόνη με το μήνυμα: ’An error occured, please try again!’.



Σχήμα 5.16: Λίστα όλων των Capability εγγραφών

5.2.4.2 Λεπτομέρειες της εγγραφής Capability

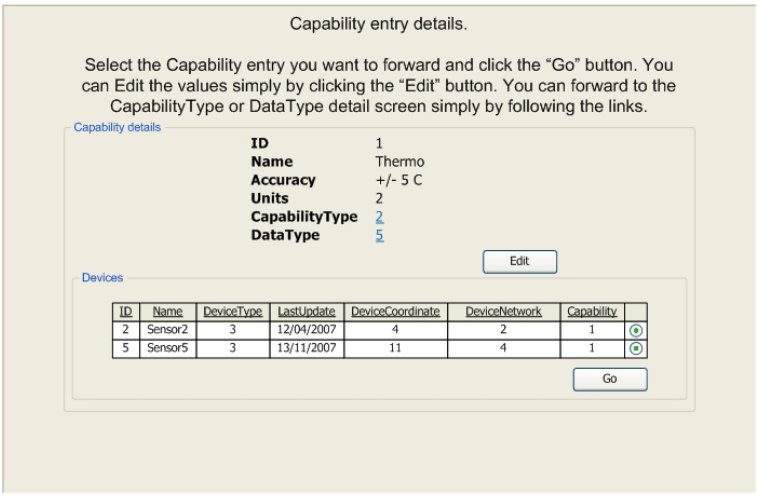
Σε αυτή την οθόνη(Βλέπε Σχήμα 5.17) παρουσιάζονται όλες οι λεπτομέρειες της Capability εγγραφής. Στην οθόνη αυτή γίνεται χρήση της Capability οντότητας, η κλάση CapabilityManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- Device Επιλογή: Radio button. Ο χρήστης μπορεί να διαλέξει κάποια Device εγγραφή για να προωθηθεί.
- CapabilityType Επιλογή: HyperLink. Ο χρήστης μπορεί να προωθηθεί στην οθόνη της CapabilityType εγγραφής.
- DataType Επιλογή: HyperLink. Ο χρήστης μπορεί να προωθηθεί στην οθόνη της DataType εγγραφής.
- Go: Submit button. Αυτό το κουμπί προωθεί το χρήστη στην οθόνη της Device εγγραφής.
- Edit: Submit button. Αυτό το κουμπί προωθεί το χρήστη στην οθόνη επεξεργασίας της Capability εγγραφής.

Έλεγχοι

- Έλεγχος επιλογής: Ενεργοποιείται με το πάτημα του κουμπιού ”Go”. Ελέγχει εάν έχει επιλεγεί κάποιο ”radio button”. Αν όχι, ειδοποιεί το χρήστη.



Σχήμα 5.17: Λεπτομέρειες της εγγραφής Capability

5.2.4.3 Προσθήκη εγγραφής Capability

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.18) ο χρήστης μπορεί να προσθέση μια νέα Capability εγγραφή. Στην οθόνη αυτή γίνεται χρήση της Capability οντότητας, η κλάση CapabilityManager έχει τα εξής χαρακτηριστικά:

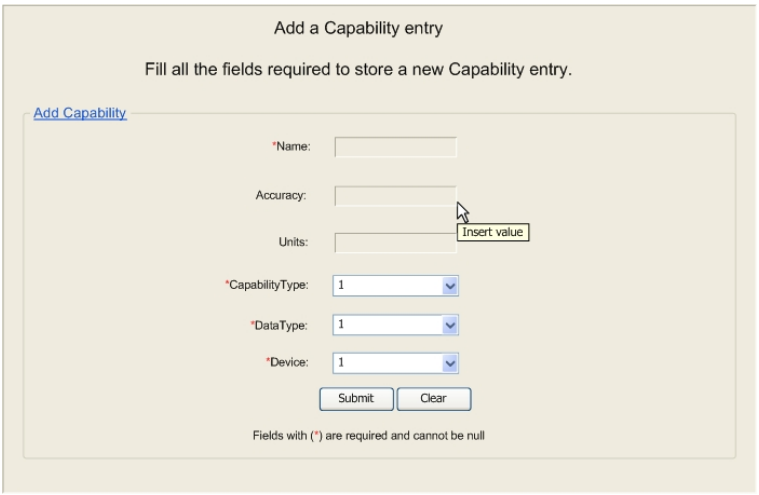
Λειτουργίες

- Name: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει ένα όνομα για τη νέα Capability εγγραφή he wants to add. Το πεδίο είναι υποχρεωτικό.
- Accuracy: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια τιμή ακρίβειας για τη νέα Capability εγγραφή. Το πεδίο είναι προαιρετικό.
- Units: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει τον αριθμό των μονάδων για αυτή τη νέα Capability εγγραφή. Το πεδίο είναι προαιρετικό.
- CapabilityType: Drop-down Menu. Ο χρήστης επιλέγει μια CapabilityType τιμή από τις τιμές που έχουν επιστραφεί στο "drop-down" μενού. Η επιλογή είναι υποχρεωτική.
- DataType: Drop-down Menu. Ο χρήστης επιλέγει μια DataType τιμή από τις τιμές που έχουν επιστραφεί στο "drop-down" μενού. Η επιλογή είναι υποχρεωτική .
- Device: Drop-down Menu. Ο χρήστης επιλέγει μια Device από τις τιμές που έχουν επιστραφεί στο "drop-down" μενού. Η επιλογή είναι υποχρεωτική.
- Submit: Submit button. Το κουμπί αυτό υποβάλλει τα εισαγόμενα στη φόρμα δεδομένα.

- Clear: Clear button. Το κουμπί αυτό καθαρίζει όλα τα πεδία ώστε ο χρήστης να μπορεί από την αρχή να τα συμπληρώσει.

Έλεγχοι

- Έλεγχος πληρότητας των πεδίων: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει κάθε υποχρεωτικό πεδίο. Αν υπάρχει τουλάχιστον ένα κενό, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένης εισαγωγής: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει αν τα δεδομένα υποβλήθηκαν επιτυχώς. Αν ναι, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'Entry succesfully added.', διαφορετικά, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'An error occured, please try again!'.



Σχήμα 5.18: Προσθήκη εγγραφής Capability

5.2.4.4 Επεξεργασία εγγραφής Capability

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.19) ο χρήστης μπορεί να επεξεργαστεί και να αλλάξει την τιμή της Capability εγγραφής. Στην οθόνη αυτή γίνεται χρήση της Capability οντότητας, η κλάση CapabilityManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- Name: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει ένα όνομα για την DeviceType εγγραφή he wants to update. Το πεδίο είναι υποχρεωτικό.
- Accuracy: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια τιμή ακρίβειας για την Capability εγγραφή που θέλει να ενημερώσει. Το πεδίο είναι προαιρετικό.

- Units: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει τον αριθμό των μονάδων για την Capability εγγραφή που θέλει να ενημερώσει. Το πεδίο είναι προαιρετικό.
- CapabilityType: Drop-down Menu. Ο χρήστης επιλέγει μια CapabilityType τιμή από τις τιμές που έχουν επιστραφεί στο ”drop-down” μενού. Η επιλογή είναι υποχρεωτική .
- DataType: Drop-down Menu. Ο χρήστης επιλέγει μια DataType τιμή από τις τιμές που έχουν επιστραφεί στο ”drop-down” μενού. Η επιλογή είναι υποχρεωτική .
- Device: Drop-down Menu. Ο χρήστης επιλέγει μια Device από τις τιμές που έχουν επιστραφεί στο ”drop-down” μενού. This option is required.
- Submit: Submit button. Το κουμπί αυτό υποβάλλει τα εισαγόμενα στη φόρμα δεδομένα.
- Clear: Clear button. Το κουμπί αυτό καθαρίζει όλα τα πεδία ώστε ο χρήστης να μπορεί από την αρχή να τα συμπληρώσει.

Έλεγχοι

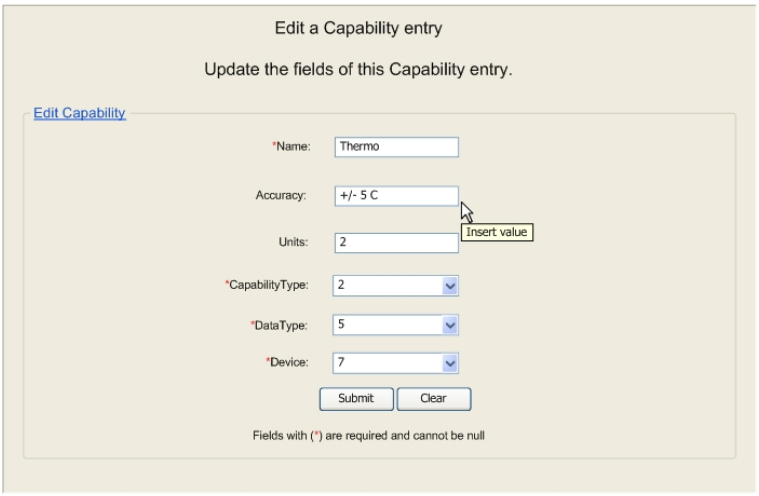
- Έλεγχος πληρότητας των πεδίων: Ενεργοποιείται με το πάτημα του κουμπιού ”Submit”. Ελέγχει κάθε υποχρεωτικό πεδίο. Αν υπάρχει τουλάχιστον ένα κενό, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένης εισαγωγής: Ενεργοποιείται με το πάτημα του κουμπιού ”Submit”. Ελέγχει αν τα δεδομένα υποβλήθηκαν επιτυχώς. Αν ναι, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα ’Entry successfully updated.’, διαφορετικά, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα ’An error occured, please try again!’.

5.2.5 Χειρισμός CapabilityType

Εδώ παρουσιάζουμε τις οθόνες που σχετίζονται με το χειρισμό της CapabilityType οντότητας.

5.2.5.1 Κατάταξη όλων των CapabilityType εγγραφών(διαγραφή επίσης)

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.20) απαριθμούνται όλες οι CapabilityType εγγραφές. Υπάρχει δυνατότητα διαγραφής εδώ. Στην οθόνη αυτή γίνεται χρήση της CapabilityType οντότητας, η κλάση CapabilityTypeManager έχει τα εξής χαρακτηριστικά:



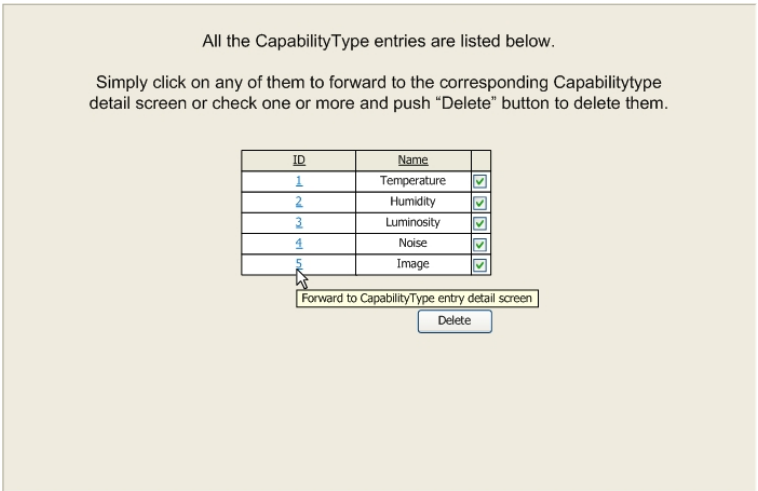
Σχήμα 5.19: Επεξεργασία εγγραφής Capability

Λειτουργίες

- CapabilityType Επιλογή: Hyperlink. Ο χρήστης απλά πατάει ”κλικ” στο ID της επιλογής του για να προωθηθεί στην οθόνη με τις λεπτομέρειες της CapabilityType εγγραφής.
- CapabilityType Deletion: CheckBox. Ο χρήστης ”τσεκάρει” απλά ένα ή περισσότερα check-box για να διαγράψει εγγραφές τύπου CapabilityType.
- Delete: Submit Button. Το κουμπί αυτό στέλνει το αίτημα για διαγραφή.
- Ταξινόμηση: Hyperlink. Ο χρήστης μπορεί να κάνει ένα απλό ”κλικ” σε μία από τις στήλες του πίνακα για να κάνει ταξινόμηση(η ταξινόμηση μπορεί να γίνει κατά ID or Name).

Έλεγχοι

- Έλεγχος των ”check-boxes”: Ενεργοποιείται με το πάτημα του κουμπιού ”Delete”. Ελέγχει όλα τα ”check-boxes”. Εάν κάθε ένα από αυτά είναι αμαρκάριστο, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένου αιτήματος: Ενεργοποιείται με το πάτημα του κουμπιού ”Delete”. Ελέγχει εάν το αίτημα ήταν επιτυχές. Αν ναι, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα: ”The deletion was successful”, διαφορετικά ο χρήστης προωθείται σε μια οθόνη με το μήνυμα: ’An error occured, please try again!’.



Σχήμα 5.20: Λίστα όλων των CapabilityType εγγραφών

5.2.5.2 Λεπτομέρειες της εγγραφής CapabilityType

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.21) παρουσιάζονται όλες οι λεπτομέρειες της CapabilityType εγγραφής. Στην οθόνη αυτή γίνεται χρήση της CapabilityType οντότητας, η κλάση CapabilityTypeMan έχει τα εξής χαρακτηριστικά:

Λειτουργίες

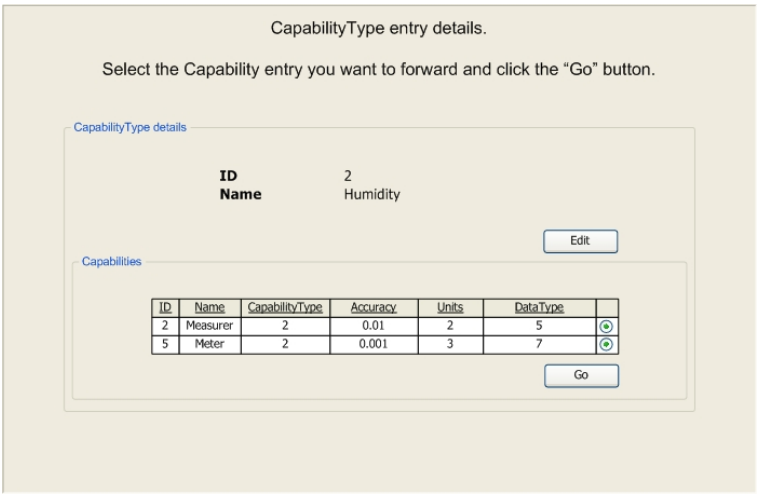
- Capability Επιλογή: Radio button. Ο χρήστης μπορεί να διαλέξει κάποια Capability εγγραφή για να προωθηθεί.
- Go: Submit button. Αυτό το κουμπί προωθεί το χρήστη στην οθόνη της Capability εγγραφής.
- Edit: Submit button. Αυτό το κουμπί προωθεί το χρήστη στην οθόνη επεξεργασίας της CapabilityType εγγραφής.

Έλεγχοι

- Έλεγχος επιλογής: Ενεργοποιείται με το πάτημα του κουμπιού "Go". Ελέγχει εάν έχει επιλεγεί κάποιο "radio button". Αν όχι, ειδοποιεί το χρήστη.

5.2.5.3 Προσθήκη εγγραφής CapabilityType

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.22) ο χρήστης μπορεί να προσθέση μια νέα CapabilityType εγγραφή. Στην οθόνη αυτή γίνεται χρήση της CapabilityType οντότητας, η κλάση CapabilityTypeMana έχει τα εξής χαρακτηριστικά:



Σχήμα 5.21: Λεπτομέρειες της εγγραφής CapabilityType

Λειτουργίες

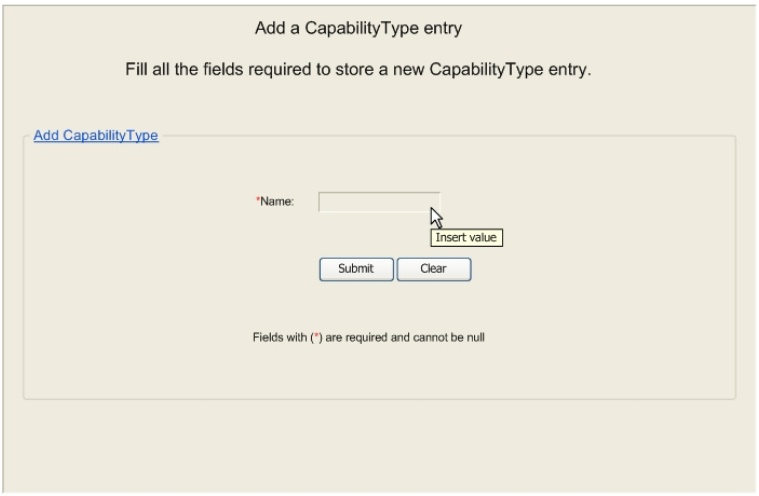
- Name: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει ένα όνομα για τη νέα CapabilityType εγγραφή που θέλει να προσθέσει. Το πεδίο είναι υποχρεωτικό.
- Submit: Submit button. Το κουμπί αυτό υποβάλλει τα εισαγόμενα στη φόρμα δεδομένα.
- Clear: Clear button. Το κουμπί αυτό καθαρίζει όλα τα πεδία ώστε ο χρήστης να μπορεί από την αρχή να τα συμπληρώσει.

Έλεγχοι

- Έλεγχος πληρότητας των πεδίων: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει το υποχρεωτικό πεδίο. Αν είναι κενό, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένης εισαγωγής: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει αν τα δεδομένα υποβλήθηκαν επιτυχώς. Αν ναι, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'Entry succesfully added.', διαφορετικά, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'An error occured, please try again!'.

5.2.5.4 Επεξεργασία εγγραφής CapabilityType

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.23) ο χρήστης μπορεί να επεξεργαστεί και να αλλάξει την τιμή της CapabilityType εγγραφής. Στην οθόνη αυτή γίνεται χρήση της CapabilityType οντότητας, η κλάση CapabilityTypeManager έχει τα εξής χαρακτηριστικά:



Σχήμα 5.22: Προσθήκη εγγραφής CapabilityType

Λειτουργίες

- Name: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει ένα όνομα για την CapabilityType εγγραφή που θέλει να ενημερώσει. Το πεδίο είναι υποχρεωτικό.
- Submit: Submit button. Το κουμπί αυτό υποβάλλει τα εισαγόμενα στη φόρμα δεδομένα.
- Clear: Clear button. Το κουμπί αυτό καθαρίζει όλα τα πεδία ώστε ο χρήστης να μπορεί από την αρχή να τα συμπληρώσει.

Έλεγχοι

- Έλεγχος πληρότητας των πεδίων: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει κάθε υποχρεωτικό πεδίο. Αν υπάρχει τουλάχιστον ένα κενό, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένης εισαγωγής: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει αν τα δεδομένα υποβλήθηκαν επιτυχώς. Αν ναι, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'Entry succesfully updated.', διαφορετικά, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'An error occured, please try again!'.

5.2.6 Χειρισμός DataType

Εδώ παρουσιάζουμε τις οθόνες που σχετίζονται με το χειρισμό της DataType οντότητας.

Σχήμα 5.23: Επεξεργασία εγγραφής CapabilityType

5.2.6.1 Κατάταξη όλων των DataType εγγραφών(διαγραφή επίσης)

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.24) απαριθμούνται όλες οι DataType εγγραφές. Υπάρχει δυνατότητα διαγραφής εδώ. Στην οθόνη αυτή γίνεται χρήση της DataType οντότητας, η κλάση DataTypeManager έχει τα εξής χαρακτηριστικά:

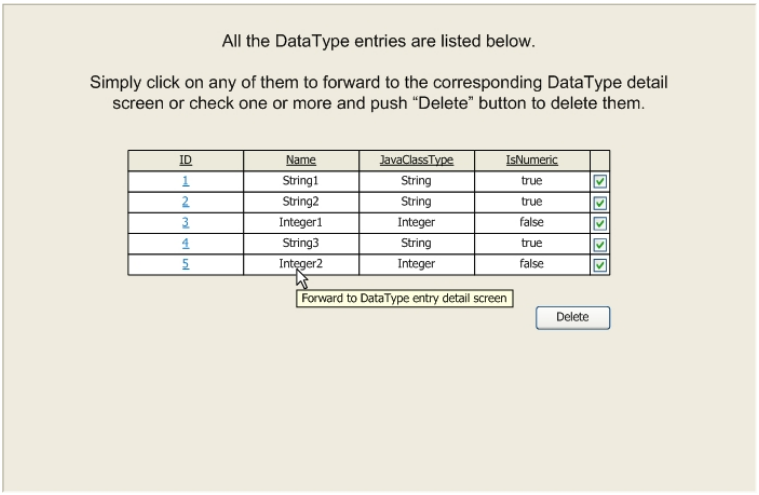
Λειτουργίες

- DataType Επιλογή: Hyperlink. Ο χρήστης απλά πατάει ”κλικ” στο ID της επιλογής του για να προωθηθεί στην οθόνη με τις λεπτομέρειες της DataType εγγραφής.
- DataType Deletion: CheckBox. Ο χρήστης ”τσεκάρει” απλά ένα ή περισσότερα check-box για να διαγράψει εγγραφές τύπου DataType.
- Delete: Submit Button. Το κουμπί αυτό στέλνει το αίτημα για διαγραφη
- Ταξινόμηση: Hyperlink. Ο χρήστης μπορεί να κάνει ένα απλό ”κλικ” σε μία από τις στήλες του πίνακα για να κάνει ταξινόμηση(η ταξινόμηση μπορεί να γίνει κατά ID, Name, κτλ...).

Έλεγχοι

- Έλεγχος των ”check-boxes”: Ενεργοποιείται με το πάτημα του κουμπιού ”Delete”. Ελέγχει όλα τα ”check-boxes”. Εάν κάθε ένα από αυτά είναι αμαρκάριστο, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένου αιτήματος: Ενεργοποιείται με το πάτημα του κουμπιού ”Delete”. Ελέγχει εάν το αίτημα ήταν επιτυχές. Αν ναι, ο χρήστης οδηγείται σε

μια οθόνη με το μήνυμα: ”The deletion was successful”, διαφορετικά ο χρήστης προωθείται σε μια οθόνη με το μήνυμα: ’An error occured, please try again!’.



Σχήμα 5.24: Λίστα όλων των DataType εγγραφών

5.2.6.2 Λεπτομέρειες της εγγραφής DataType

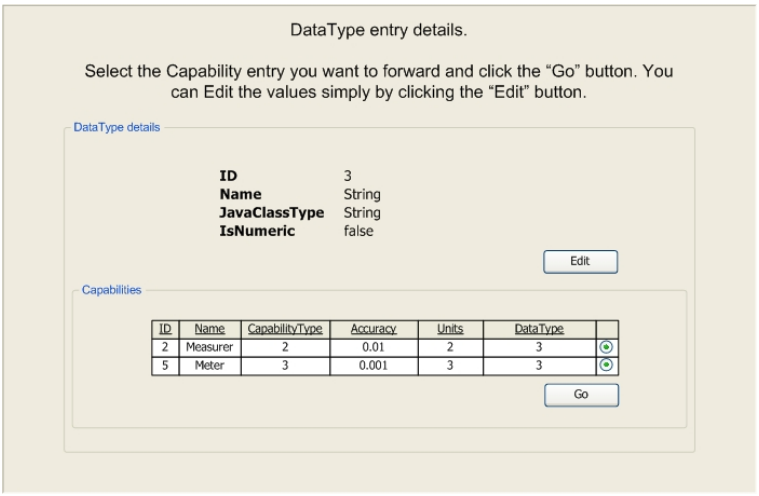
Σε αυτή την οθόνη(Βλέπε Σχήμα 5.25) παρουσιάζονται όλες οι λεπτομέρειες της DataType εγγραφής. Στην οθόνη αυτή γίνεται χρήση της DataType οντότητας, η κλάση DataTypeManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- Capability Επιλογή: Radio button. Ο χρήστης μπορεί να διαλέξει κάποια Capability εγγραφή για να προωθηθεί.
- Go: Submit button. Αυτό το κουμπί προωθεί το χρήστη στην οθόνη της Capability εγγραφής.
- Edit: Submit button. Αυτό το κουμπί προωθεί το χρήστη στην οθόνη επεξεργασίας της DataType εγγραφής.

Έλεγχοι

- Έλεγχος επιλογής: Ενεργοποιείται με το πάτημα του κουμπιού ”Go”. Ελέγχει εάν έχει επιλεγεί κάποιο ”radio button”. Αν όχι, ειδοποιεί το χρήστη.



Σχήμα 5.25: Λεπτομέρειες της εγγραφής DataType

5.2.6.3 Προσθήκη εγγραφής DataType

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.26) ο χρήστης μπορεί να προσθέση μια νέα DataType εγγραφή. Στην οθόνη αυτή γίνεται χρήση της DataType οντότητας, η κλάση DataTypeManager έχει τα εξής χαρακτηριστικά:

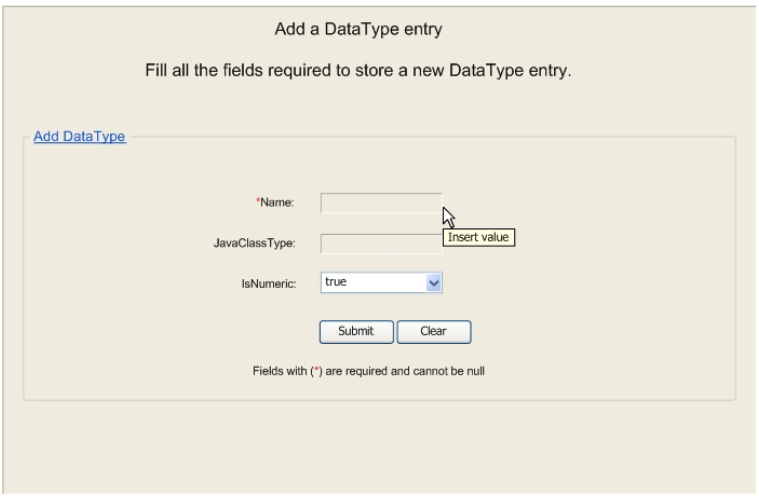
Λειτουργίες

- Name: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει ένα όνομα για τη νέα DataType εγγραφή που θέλει να προσθέσει. Το πεδίο είναι υποχρεωτικό.
- JavaClassType: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει τον τύπο της *java* κλάσης για τη νέα DataType εγγραφή που θέλει να προσθέσει. Το πεδίο αυτό είναι προαιρετικό.
- IsNumeric: Drop-down Menu(true-false). Ο χρηστης επιλέγει αν η DataType εγγραφή που θέλει να προσθέσει είναι αριθμήσιμη ή όχι. Το πεδίο αυτό είναι προαιρετικό.
- Submit: Submit button. Το κουμπί αυτό υποβάλλει τα εισαγόμενα στη φόρμα δεδομένα.
- Clear: Clear button. Το κουμπί αυτό καθαρίζει όλα τα πεδία ώστε ο χρήστης να μπορεί από την αρχή να τα συμπληρώσει.

Έλεγχοι

- Έλεγχος πληρότητας των πεδίων: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει το υποχρεωτικό πεδίο. Αν είναι κενό, ειδοποιεί το χρήστη.

- Έλεγχος επιτυχημένης εισαγωγής: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει αν τα δεδομένα υποβλήθηκαν επιτυχώς. Αν ναι, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'Entry successfully added.', διαφορετικά, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'An error occured, please try again!'.



Σχήμα 5.26: Προσθήκη εγγραφής DataType

5.2.6.4 Επεξεργασία εγγραφής DataType

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.27) ο χρήστης μπορεί να επεξεργαστεί και να αλλάξει την τιμή της DataType εγγραφής. Στην οθόνη αυτή γίνεται χρήση της DataType οντότητας, η κλάση DataTypeManager έχει τα εξής χαρακτηριστικά:

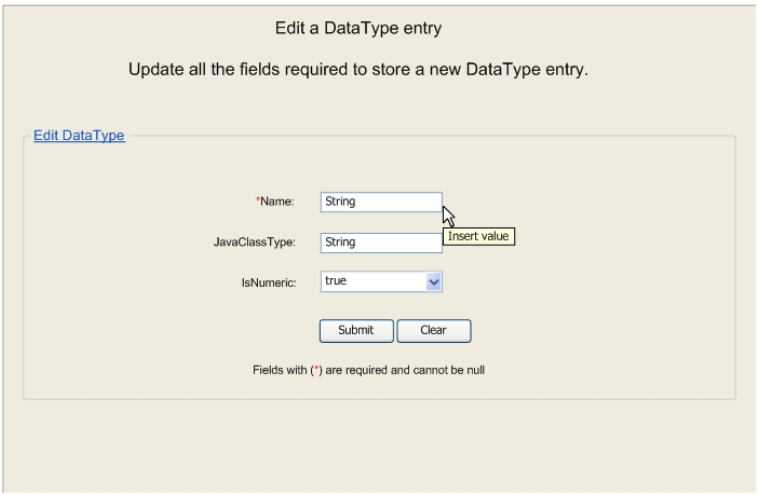
Λειτουργίες

- Name: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει ένα όνομα για την DataType εγγραφή που θέλει να ενημερώσει. Το πεδίο είναι υποχρεωτικό.
- JavaClassType: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει τον τύπο της *java* κλάσης για τη νέα DataType εγγραφή που θέλει να ενημερώσει. Το πεδίο αυτό είναι προαιρετικό.
- IsNumeric: Drop-down Menu(true-false). Ο χρηστης επιλέγει αν η DataType εγγραφή που θέλει να ενημερώσει είναι αριθμήσιμη ή όχι. Το πεδίο αυτό είναι προαιρετικό.
- Submit: Submit button. Το κουμπί αυτό υποβάλλει τα εισαγόμενα στη φόρμα δεδομένα.

- Clear: Clear button. Το κουμπί αυτό καθαρίζει όλα τα πεδία ώστε ο χρήστης να μπορεί από την αρχή να τα συμπληρώσει.

Έλεγχοι

- Έλεγχος πληρότητας των πεδίων: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει κάθε υποχρεωτικό πεδίο. Αν υπάρχει τουλάχιστον ένα κενό, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένης εισαγωγής: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει αν τα δεδομένα υποβλήθηκαν επιτυχώς. Αν ναι, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'Entry successfully updated.', διαφορετικά, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'An error occured, please try again!'.



Σχήμα 5.27: Επεξεργασία εγγραφής DataType

5.2.7 Χειρισμός DeviceNetwork

Εδώ παρουσιάζουμε τις οθόνες που σχετίζονται με το χειρισμό της DeviceNetwork οντότητας.

5.2.7.1 Κατάταξη όλων των DeviceNetwork εγγραφών(διαγραφή επίσης)

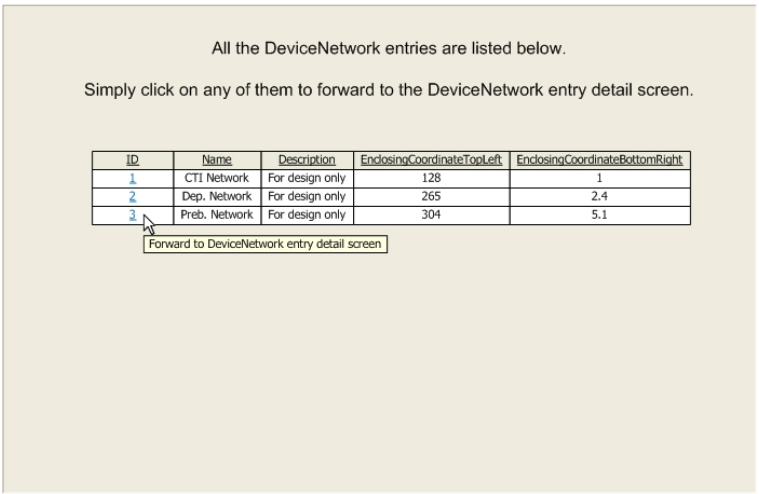
Σε αυτή την οθόνη(Βλέπε Σχήμα 5.28) απαριθμούνται όλες οι DeviceNetwork εγγραφές. Υπάρχει δυνατότητα διαγραφής εδώ. Στην οθόνη αυτή γίνεται χρήση της DeviceNetwork οντότητας, η κλάση DeviceNetworkTypeManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- DeviceNetwork Επιλογή: Hyperlink. Ο χρήστης απλά πατάει ”κλικ” στο ID της επιλογής του για να προωθηθεί στην οθόνη με τις λεπτομέρειες της DeviceNetwork εγγραφής.
- DeviceNetwork Deletion: CheckBox. Ο χρήστης ”τσεκάρει” απλά ένα ή περισσότερα check-box για να διαγράψει εγγραφές τύπου DeviceNetwork.
- Delete: Submit Button. Το κουμπί αυτό στέλνει το αίτημα για διαγραφή.
- Ταξινόμηση: Hyperlink. Ο χρήστης μπορεί να κάνει ένα απλό ”κλικ” σε μία από τις στήλες του πίνακα για να κάνει ταξινόμηση(η ταξινόμηση μπορεί να γίνει κατά ID, Name, κτλ...).

Έλεγχοι

- Έλεγχος των ”check-boxes”: Ενεργοποιείται με το πάτημα του κουμπιού ”Delete”. Ελέγχει όλα τα ”check-boxes”. Εάν κάθε ένα από αυτά είναι αμαρκάριστο, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένου αιτήματος: Ενεργοποιείται με το πάτημα του κουμπιού ”Delete”. Ελέγχει εάν το αίτημα ήταν επιτυχές. Αν ναι, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα: ”The deletion was successful”, διαφορετικά ο χρήστης προωθείται σε μια οθόνη με το μήνυμα: ’An error occured, please try again!’.



Σχήμα 5.28: Λίστα όλων των DeviceNetwork εγγραφών

5.2.7.2 Λεπτομέρειες της εγγραφής DeviceNetwork

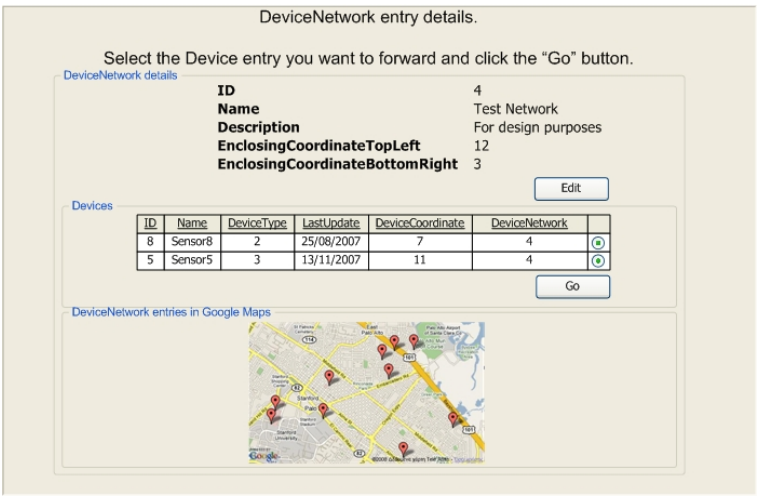
Σε αυτή την οθόνη(Βλέπε Σχήμα 5.29) παρουσιάζονται όλες οι λεπτομέρειες της DeviceNetwork εγγραφής. Στην οθόνη αυτή γίνεται χρήση της DeviceNetwork οντότητας, η κλάση DeviceNetworkManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- Device Επιλογή: Radio button. Ο χρήστης μπορεί να διαλέξει κάποια Device εγγραφή για να προωθηθεί.
- Go: Submit button. Αυτό το κουμπί προωθεί το χρήστη στην οθόνη της Device εγγραφής.
- Edit: Submit button. Αυτό το κουμπί προωθεί το χρήστη στην οθόνη επεξεργασίας της DeviceNetwork edit screen.
- Παρουσίαση Google Maps: Google Maps Markers. Ο χρήστης μπορεί να παρακολουθεί την περιοχή με την οποία οι DeviceNetwork εγγραφές σχετίζονται. Κάθε κόμβος σε αυτή την περιοχή παρουσιάζεται ως *Marker*.

Έλεγχοι

- Έλεγχος επιλογής: Ενεργοποιείται με το πάτημα του κουμπιού "Go". Ελέγχει εάν έχει επιλεγεί κάποιο "radio button". Αν όχι, ειδοποιεί το χρήστη.



Σχήμα 5.29: Λεπτομέρειες της εγγραφής DeviceNetwork

5.2.7.3 Προσθήκη εγγραφής DeviceNetwork

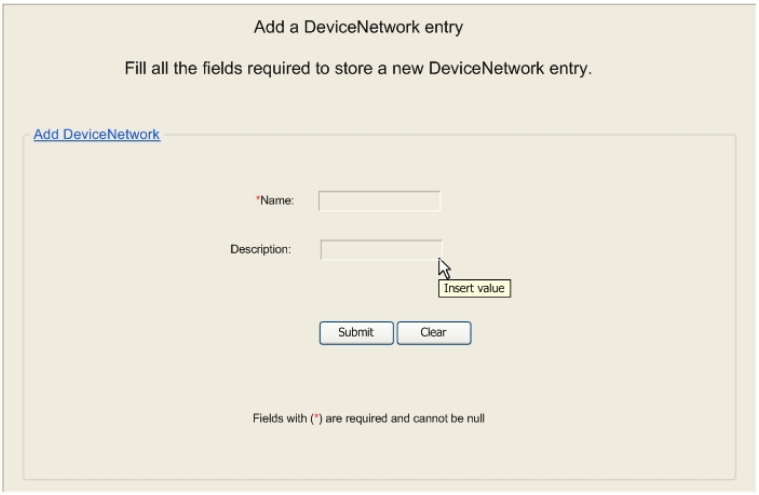
Σε αυτή την οθόνη(Βλέπε Σχήμα 5.30) ο χρήστης μπορεί να προσθέση μια νέα DeviceNetwork εγγραφή. Στην οθόνη αυτή γίνεται χρήση της DeviceNetwork οντότητας, η κλάση DeviceNetworkMan έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- Name: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει ένα όνομα για τη νέα DeviceNetwork εγγραφή που θέλει να προσθέσει. Το πεδίο είναι υποχρεωτικό.
- Description: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια περιγραφή για τη νέα DeviceNetwork εγγραφή. Το πεδίο είναι προαιρετικό.
- Submit: Submit button. Το κουμπί αυτό υποβάλλει τα εισαγόμενα στη φόρμα δεδομένα.
- Clear: Clear button. Το κουμπί αυτό καθαρίζει όλα τα πεδία ώστε ο χρήστης να μπορεί από την αρχή να τα συμπληρώσει.

Έλεγχοι

- Έλεγχος πληρότητας των πεδίων: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει το υποχρεωτικό πεδίο. Αν είναι κενό, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένης εισαγωγής: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει αν τα δεδομένα υποβλήθηκαν επιτυχώς. Αν ναι, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'Entry succesfully added.', διαφορετικά, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'An error occured, please try again!'.



Σχήμα 5.30: Προσθήκη εγγραφής DeviceNetwork

5.2.7.4 Επεξεργασία εγγραφής DeviceNetwork

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.31) ο χρήστης μπορεί να επεξεργαστεί και να αλλάξει την τιμή της DeviceNetwork εγγραφής. Στην οθόνη αυτή γίνεται χρήση της DeviceNetwork οντότητας, η κλάση DeviceNetworkManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- Name: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει ένα όνομα για την DeviceNetwork εγγραφή που θέλει να ενημερώσει. Το πεδίο είναι υποχρεωτικό.
- Submit: Submit button. Το κουμπί αυτό υποβάλλει τα εισαγόμενα στη φόρμα δεδομένα.
- Clear: Clear button. Το κουμπί αυτό καθαρίζει όλα τα πεδία ώστε ο χρήστης να μπορεί από την αρχή να τα συμπληρώσει.

Έλεγχοι

- Έλεγχος πληρότητας των πεδίων: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει κάθε υποχρεωτικό πεδίο. Αν υπάρχει τουλάχιστον ένα κενό, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένης εισαγωγής: Ενεργοποιείται με το πάτημα του κουμπιού "Submit". Ελέγχει αν τα δεδομένα υποβλήθηκαν επιτυχώς. Αν ναι, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'Entry succesfully updated.', διαφορετικά, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'An error occured, please try again!'.

Σχήμα 5.31: Επεξεργασία εγγραφής DeviceNetwork

5.2.7.5 Αναζήτηση εγγραφών DeviceNetwork σε περιοχή ορθογωνίου

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.32) ο χρήστης μπορεί να αναζητήσει για όλες τις DeviceNetwork εγγραφές οι οποίες βρίσκονται μέσα στη δοσμένη περιοχή ορθογωνίου.

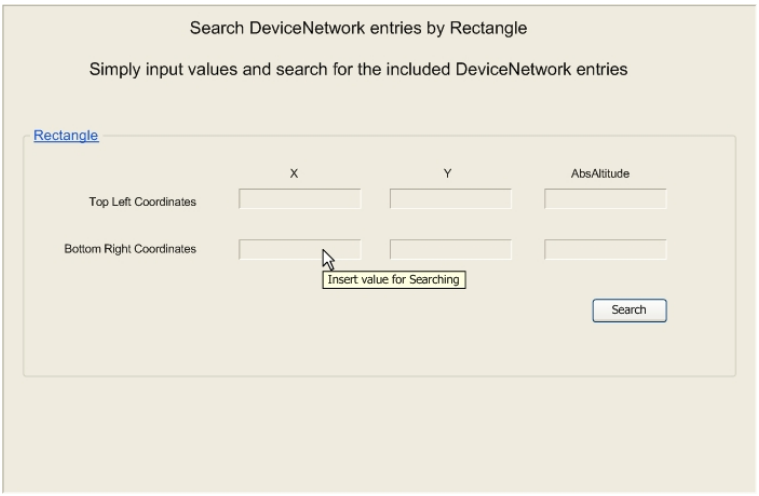
Στην οθόνη αυτή γίνεται χρήση της DeviceNetwork οντότητας, η κλάση DeviceNetworkManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

- Top Left X bound: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια τιμή(Double) η οποία αναπαριστά την X συντεταγμένη της πάνω αριστερά γωνίας της ορθογώνιας περιοχής..
- Top Left Y bound: Character field.Σε αυτό το πεδίο ο χρήστης εισάγει μια τιμή(Double) η οποία αναπαριστά την Y συντεταγμένη της πάνω αριστερά γωνίας της ορθογώνιας περιοχής.
- Top Left AbsAltitude bound: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια τιμή(Double) η οποία αναπαριστά το απόλυτο ύψος του σημείου της πάνω αριστερά γωνίας της ορθογώνιας περιοχής.
- Bottom Right X bound: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια τιμή(Double) η οποία αναπαριστά την X συντεταγμένη της κάτω δεξιά γωνίας της ορθογώνιας περιοχής.
- Bottom Right Y bound: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια τιμή(Double) η οποία αναπαριστά την Y συντεταγμένη της κάτω δεξιά γωνίας της ορθογώνιας περιοχής.
- Bottom Right AbsAltitude bound: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια τιμή(Double) η οποία αναπαριστά το απόλυτο ύψος του σημείου της κάτω δεξιά γωνίας της ορθογώνιας περιοχής.
- Search: Submit button. Το κουμπί αυτό υποβάλλει τα εισαγόμενα στη φόρμα δεδομένα.

Έλεγχοι

- Έλεγχος πληρότητας των πεδίων: Ενεργοποιείται με το πάτημα του κουμπιού "Search". Ελέγχει κάθε ένα πεδίο. Αν υπάρχει τουλάχιστον ένα κενό, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένης εισαγωγής: Ενεργοποιείται με το πάτημα του κουμπιού "Search". Ελέγχει αν τα δεδομένα υποβλήθηκαν επιτυχώς. Αν ναι, ο χρηστης οδηγείται σε μια οθόνη με όλες τις DeviceNetwork εγγραφές που βρίσκονται μέσα στα δοσμένα όρια, διαφορετικά, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα 'An error occured, please try again!'.



Σχήμα 5.32: Αναζήτηση εγγραφών DeviceNetwork σε περιοχή ορθογωνίου

5.2.8 Χειρισμός Reading

Εδώ παρουσιάζουμε τις οθόνες που σχετίζονται με το χειρισμό της Reading οντότητας.

5.2.8.1 Κατάταξη όλων των Reading εγγραφών

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.33) απαριθμούνται όλες οι Reading εγγραφές. Στην οθόνη αυτή γίνεται χρήση της Reading οντότητας, η κλάση ReadingManager έχει τα εξής χαρακτηριστικά:

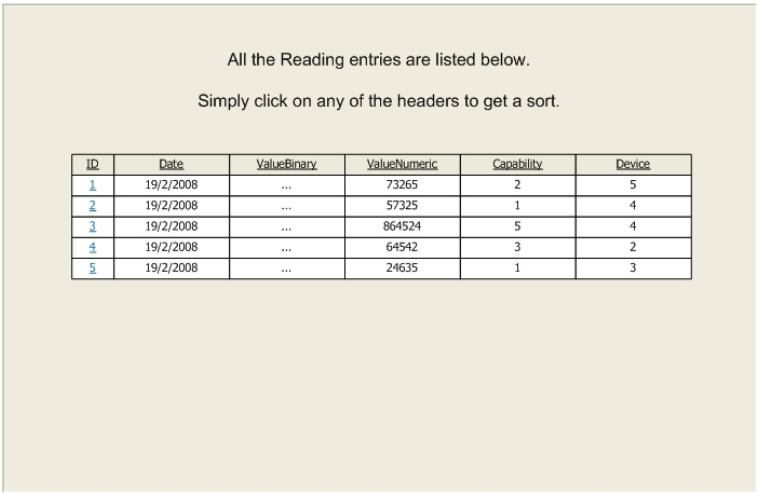
Λειτουργίες

- Ταξινόμηση: Hyperlink. Ο χρήστης μπορεί να κάνει ένα απλό ”κλικ” σε μία από τις στήλες του πίνακα για να κάνει ταξινόμηση(η ταξινόμηση μπορεί να γίνει κατά ID, Date, κτλ..).

5.2.8.2 Αναζήτηση εγγραφών Reading κατά ημερομηνία

Σε αυτή την οθόνη(Βλέπε Σχήμα 5.34) ο χρήστης μπορεί να αναζητήσει για όλες τις Reading εγγραφές των οποίων η ημερομηνία περιέχεται μεταξύ των δύο δοσμένων ημερομηνιών που εισάγει ο χρήστης. Στην οθόνη αυτή γίνεται χρήση της Reading οντότητας, η κλάση ReadingManager έχει τα εξής χαρακτηριστικά:

Λειτουργίες

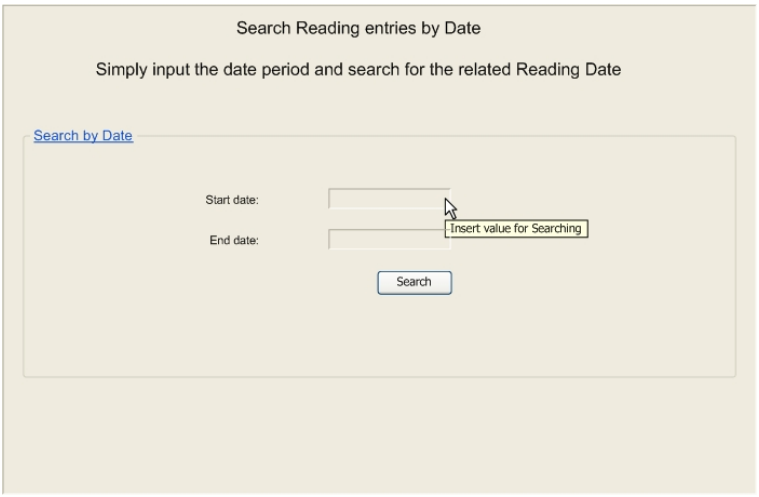


Σχήμα 5.33: Λίστα όλων των Reading εγγραφών

- Start Date: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια αρχική ημερομηνία.
- End Date: Character field. Σε αυτό το πεδίο ο χρήστης εισάγει μια τελική ημερομηνία.
- Search: Submit button. Το κουμπί αυτό υποβάλλει τα εισαγόμενα στη φόρμα δεδομένα.

Έλεγχοι

- Έλεγχος πληρότητας των πεδίων: Ενεργοποιείται με το πάτημα του κουμπιού ”Search”. Ελέγχει το πεδίο. Αν είναι κενό, ειδοποιεί το χρήστη.
- Έλεγχος επιτυχημένης εισαγωγής: Ενεργοποιείται με το πάτημα του κουμπιού ”Search”. Ελέγχει αν τα δεδομένα υποβλήθηκαν επιτυχώς. Αν ναι, ο χρηστης οδηγείται σε μια οθόνη με όλες τις Reading εγγραφές που είναι μέσα στο δοσμένο διάστημα ημερομηνιών, διαφορετικά, ο χρήστης οδηγείται σε μια οθόνη με το μήνυμα ’An error occured, please try again!’.



Σχήμα 5.34: Αναζήτηση εγγραφών Reading κατά ημερομηνία

Κεφάλαιο 6

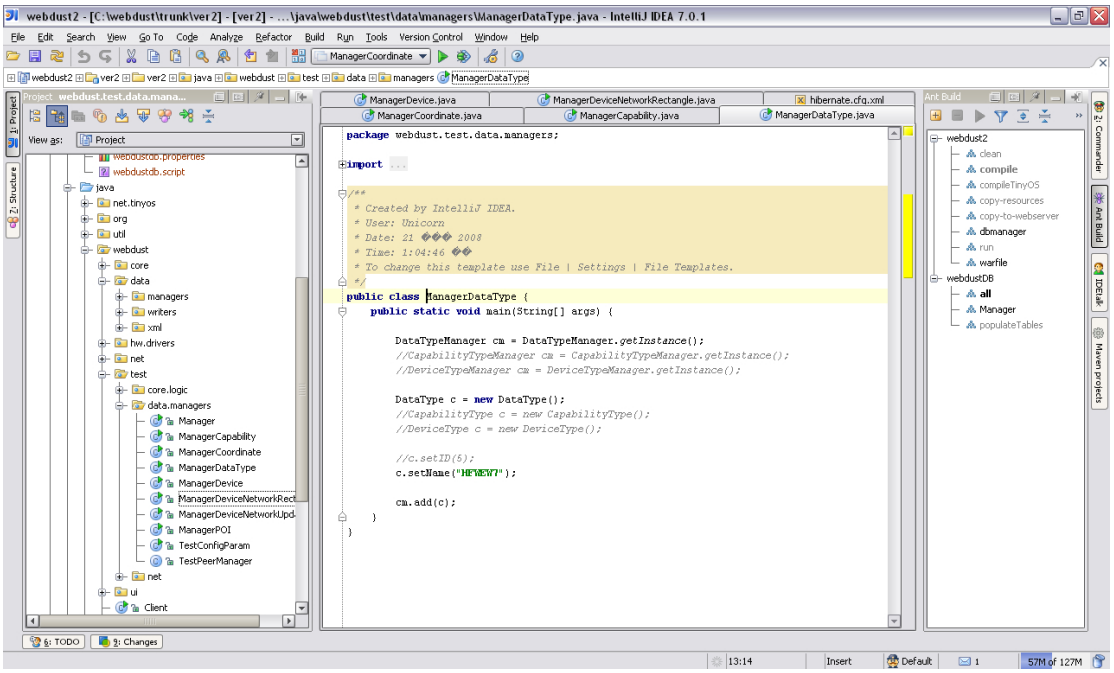
Εργαλεία Ανάπτυξης Κώδικα

Στο κεφάλαιο αυτό θα κάνουμε μια μικρή αναφορά-σύνοψη στα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση και την υποστήριξη της διπλωματικής αυτής εργασίας.

6.1 IntelliJ IDEA

Το IntelliJ IDEA (Βλέπε Σχήμα 6.1)αποτελεί ένα εμπορικό ολοκληρωμένο περιβάλλον ανάπτυξης κώδικα και εφαρμογών σε Java από την εταιρεία JetBrains. Το περιβάλλον αυτό είναι γραμμένο σε γλώσσα Java και υποστηρίζεται από τις περισσότερες πλατφόρμες λογισμικού. Η πρώτη έκδοση του IntelliJ IDEA εμφανίστηκε τον Ιανουάριο, του 2001, και γρήγορα έγινε πολύ δημοφιλής, όπως το πρώτο Java IDE (Integrated Development Environment) με μια σειρά ολοκληρωμένων εργαλείων για *refactoring* που επιτρέπουν στους προγραμματιστές το γρήγορο επανασχεδιασμό του κώδικά τους. Η σχεδίαση του IntelliJ IDEA επικεντρώνει στην παραγωγικότητα των προγραμματιστών. Τα χαρακτηριστικά του έχουν σκοπό να επιταχύνουν την ανάπτυξη και να επιτρέπουν στους προγραμματιστές να επικεντρωθούν στη λειτουργικότητα ενώ το IntelliJ IDEA θα χειρίζεται τα πιο κοινότοπα καθήκοντα που έχουν σχέση με τον κώδικα.

Το IntelliJ IDEA είναι αρκετά απλό στη χρήση του και ο προγραμματιστής μπορεί πολύ γρήγορα να εξοικοιωθεί με αυτό ακόμη και αν είναι συνηθισμένος σε άλλα IDE. Μέσα στα πολύ βασικά χαρακτηριστικά του IntelliJ IDEA είναι η ενσωμάτωση πολλών λογισμικών ανοιχτού κώδικα όπως τα SVN και Ant(Βλέπε στα υποκεφάλαια που ακολουθούν).



Σχήμα 6.1: Περιβάλλον IntelliJ IDEA

6.2 Subversion

Το Subversion είναι ένα δωρεάν/open-source σύστημα ελέγχου των εκδόσεων λογισμικού. Το Subversion διαχειρίζεται αρχεία και καταλόγους στην πάροδο του χρόνου. Ένα δέντρο που περιέχει το σύνολο των φακέλων μιας εφαρμογής λογισμικού τοποθετείται σε ένα κεντρικό αρχείο(*repository*). Το *repository* είναι λίγο πολύ όπως ένας κανονικός server αρχείων, με την ιδιότητα όμως να θυμάται κάθε αλλαγή που έγινε, και ποτέ, στα αρχεία και τους καταλόγους. Αυτό επιτρέπει την ανάκτηση παλαιότερων εκδόσεων των δεδομένων, και εξετάζει την ιστορία του τρόπου με τον οποίο άλλαξαν τα δεδομένα.

Το Subversion μπορεί να έχει πρόσβαση στο repository μέσω δικτύου και έτσι, αυτό το χαρακτηριστικό, του επιτρέπει να χρησιμοποιείται από ανθρώπους σε διαφορετικούς υπολογιστές. Κατα κάποιο τρόπο, η δυνατότητα διαφόρων ανθρώπων να τροποποιούν και να διαχειρίζονται το ίδιο σετ δεδομένων από τις αντίστοιχες τοποθεσίες τους ενισχύει τη συνεργασία. Η πρόοδος μπορεί να σημειωθεί πιο γρήγορα, αν δεν είναι μόνο ένας αυτός που θα πρέπει να κάνει όλες τις τροποποιήσεις που πρέπει να συμβούν. Και επειδή το έργο είναι versioned, δεν υπάρχει φόβος για τυχόν εσφαλμένες αλλαγές στον κώδικα - γιατί, απλά, αν υπάρξουν κάποιες, υπάρχει δυνατότητα αναίρεσης της αλλαγής. Η χρήση Subversion είναι απαραίτητη για ανάπτυξη εφαρμογών μεγάλου εύρους και αυτό γιατί πολλοί προγραμματιστές δουλεύουν για την περάτωση του έργου.

6.3 Ant

Ένα άλλο εργαλείο που χρησιμοποιήθηκε είναι το Ant. Το Ant είναι ένα δωρεάν εργαλείο ανοικτού κώδικα που χρησιμοποιείται ευρέως από τους προγραμματιστές της Java σε όλο τον κόσμο. Μερικές από τις εργασίες που μπορούν να εκτελεστούν από το εργαλείο αυτό είναι ενδεικτικά οι παρακάτω :

1. Compile των κλάσεων.
2. Δημιουργία, αντιγραφή, μετακίνηση και διαγραφή αρχείων και φακέλων του file system.
3. Κλήση του JUnit[1] για αυτόματη εκτέλεση των unit tests.
4. Συμπίεση και αποσυμπίεση αρχείων.
5. Πακετάρισμα κλάσεων και αρχείων σε jar , war , ear αρχεία.
6. Αποστολή και λήψη αρχείων μέσω ftp.
7. Deployment εφαρμογών σε web servers.

Να σημειωθεί ότι οι παραπάνω εργασίες αποτελούν μόνο ένα πολύ μικρό μέρος των εργασιών που μπορούν να εκτελεστούν μέσα από το Ant.

Ενώ το Ant έρχεται έτοιμο με αρκετές λειτουργίες το δυνατό του σημείο είναι ότι σχεδιάστηκε με γνώμονα την επεκτασιμότητα. Έτσι αν κάποια εργασία δεν υποστηρίζεται εν γένει μπορεί κάποιος προγραμματιστής να γράψει κώδικα και να ενσωματώσει τη δυνατότητα αυτή στο Ant ώστε να μπορεί να χρησιμοποιήσει τη λειτουργία αυτή μέσα από το Ant απολαμβάνοντας όλες τις ευκολίες που αυτό του δίνει.

Η δύναμη του Ant βρίσκεται στην επανάληψη. Αν εκτελώ κάποια ή κάποιες εργασίες επανηλλημένα με το χέρι όπως για παράδειγμα : compile, καθάρισμα φακέλων όπου θα τοποθετηθούν τα .class αρχεία, δημιουργία jar κ.τ.λ. τότε μπορώ αυτές τις εργασίες να τις γράψω σαν targets σε ένα build file (μια μόνο φορά) και να χρησιμοποιώ το Ant όποτε θέλω να εκτελέσω μια από αυτές. Επίσης το Ant μου δίνει τη δυνατότητα να δημιουργήσω «αλυσίδες» εργασιών. Για παράδειγμα ένα «καθαρό» compile προϋποθέτει διαγραφή όλων των .class αρχείων και compile όλων των .java αρχείων. Με τη χρήση του ant κάτι τέτοιο είναι πολύ απλό να γίνει και απαλλάσσει τον προγραμματιστή από πιθανά λάθη. Όλος ο «προγραμματισμός» στο Ant γίνεται μέσα από ένα build file το οποίο είναι γραμμένο σε απλή xml οπότε μπορούν να γίνουν αλλαγές γρήγορα και εύκολα.

[1] Το JUnit είναι ένα πλαίσιο δοκιμών μεμονομένων μονάδων για την γλώσσα προγραμματισμού Java. Δημιουργήθηκε από Kent Beck και Erich Gamma.

6.4 Trac

Το Trac είναι ένα ενισχυμένο σύστημα εντοπισμού θεμάτων και wiki για την ανάπτυξη έργων λογισμικού γραμμένο σε γλώσσα Python. Το Trac χρησιμοποιεί μια μινιμαλιστική προσέγγιση σε web-based λογισμικό διαχείρισης έργων. Η αποστολή του είναι να βοηθήσει τους υπεύθυνους ανάπτυξης να γράψουν το καλύτερο λογισμικό ενώ διαμένουν μακριά. Το Trac θα πρέπει να επιβληθεί όσο το δυνατόν λιγότερο στις διεργασίες και τις πολιτικές ανάπτυξης που έχουν παγιωθεί απο μια ομάδα ανάπτυξης λογισμικού.

Παρέχει μια διεπαφή για Subversion(Βλέπε Παράγραφος 6.2), ένα ολοκληρωμένο Wiki και δυνατότητες εύκολης αναφοράς.

Το Trac επιτρέπει διατύπωση wiki σε περιγραφές θεμάτων και παράδοση μηνυμάτων, δημιουργία συνδέσμων και μονοκόμματες αναφορές μεταξύ σφαλμάτων, καθηκόντων, αρχείων και wiki σελίδων. Ένα χρονοδιάγραμμα δείχνει όλα τα τρέχοντα και τα τελευταία γεγονότα στη σειρά, κάνοντας την επισκόπηση του έργου και την καταγραφή της προόδου πολύ εύκολη.

6.5 CruiseControl

Το CruiseControl είναι ένα πλαίσιο(framework) το οποίο φροντίζει για μια συνεχή build διαδικασία. Περιλαμβάνει plugins για την ειδοποίηση μέσω ηλεκτρονικού ταχυδρομείου, για το εργαλείο Ant (Βλέπε Παράγραφος 6.3), και διάφορα εργαλεία ελέγχου του κώδικα. Ένα web interface παρέχεται για την παρουσίαση των στοιχείων των τρεχόντων και των προηγούμενων build διαδικασιών.

Το CruiseControl διανέμεται στο πλαίσιο άδειας τύπου BSD και είναι ελεύθερο για χρήση. Το CruiseControl εμμένει στο open-source μοντέλο και ως εκ τούτου καθιστά τον πηγαίο κώδικα ελεύθερο.

Βιβλιογραφία

[1] Crossbow Technology Inc. Mote-view monitoring software. URL http://www.xbow.com/.

[2] The scatterweb wireless sensor network platform. URL http://www.scatterweb.de.

[3] J. Liu B. Priyantha A. Santanche, S. Nath and F. Zhao. Senseweb: Browsing the physical world in real time. Demo Abstract, ACM/IEEE IPSN06, Nashville, TN, 2006.

[4] C. Decker M. Beigl M. Isomura, T. Riedel and H. Horiuchi. Sharing sensor networks, sixth international workshop on smart appliances and wearable computing (iwsawc). Lisbon, Portugal, Proceedings of the ICDCS 2006, 2006.

[5] J. Ledlie M. Roussopoulos M. Seltzer J. Shneidman, P. Pietzuch and M. Welsh. Hourglass: An infrastructure for connecting sensor networks and applications. Tech. report, Harvard TR-21-04, 2004.

[6] M. Hauswirth K. Aberer and A. Salehi. The global sensor networks middleware for efficient and flexible deployment and interconnection of sensor networks. Tech. report, Ecole Polytechnique Federale de Lausanne (EPFL), 2006.

[7] E. Kohler B. Greenstein and D. Estrin. A sensor network application construction kit (snack), in the proc. of the 2nd international conference on embedded networked sensor systems (sensys 2004). Tech. report, Ecole Polytechnique Federale de Lausanne (EPFL):69––80.

[8] A. Tavakoli J. Hellerstein P. Levis S. Shenker D. Chu, L. Popa and I. Stoica. The design and implementation of a declarative sensor network system. Tech. report, University of California, 2006.

[9] M. Welsh and G. Mainland. Programming sensor networks using abstract regions. In the Proc. of 1st Usenix/ACM Symposium on Networked Systems Design and Implementation (NSDI'04):29–42.

[10] E. Brewer C. Sharp and D. Culler. Hood: A neighborhood abstraction for sensor networks. In the Proc. of MobiSYS'04, 2004.

[11] A. Lachenmann et al. Versatile support for efficient neighborhood data sharing. In the Proc. of the European Workshop on Sensor Networks (EWSN 2007), 2007.

[12] O. Gnawali R. Gummadi and R. Govindan. Macro-programming wireless sensor networks using kairos. In the Proc. of the International Conference on Distributed Computing in Sensor Systems (DCOSS'05), Springer:126––140.

[13] O. Gnawali et al. The tenet architecture for tiered sensor networks. In the Proc. of the 4th international conference on Embedded networked sensor systems (SenSys'06):153––166.

[14] J. Hellerstein S. Madden, M. Franklin and W. Hong. Tinydb: An acquisitional query processing system for sensor networks. Journal of ACM TODS 30:122––173.

[15] R. Gold M. Lad C. Mascolo L. Mottola G.P. Picco T. Sivaharan N. Weerasinghe P. Costa, G. Coulson and Σ. Ζαχαριάδης. The runes middleware for networked embedded systems and its application in a disaster management scenario. Proceedings of the 5th IEEE International Conference on Pervasive Computing and Communications (PERCOM07) (New York, USA), IEEE Press, Μάρτιος .

[16] T. Liu and M. Martonosi. Impala: A middleware system for managing autonomic, parallel sensor systems. In the proc. of the 9th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming (PPoPP'03):107––118.

[17] J. Hellerstein W. Hong P. Buonadonna, D. Gay and S. Madden. Task: Sensor network in a box. In the Proceedings of the 2nd European Workshop on Sensor Networks:133––144.

[18] The cougar sensor database project homepage. URL http://www.cs.cornell.edu/database/cougar.

[19] Application specific virtual machines for tinyos. URL http://www.cs.berkeley.edu/pal/mate-web/.

[20] G. Roman C. Fok and C. Lu. Rapid development and flexible deployment of adaptive wireless sensor network applications. In Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'05): 653––662.

[21] και Σ. Νικολετσέας Ι. Χατζηγιαννάκης, Γ. Μυλωνάς. A java-based generic application environment for wireless sensor networks. In the proceedings of

the first International Conference on Distributed Computing in Sensor Systems (DCOSS '05):376––386.

[22] M. Hauswirth K. Aberer and A. Salehi. The global sensor networks middleware for efficient and flexible deployment and interconnection of sensor networks. Tech. report, Ecole Polytechnique Federale de Lausanne (EPFL), 2006.

[23] E. Miluzzo R. Peterson G. Ahn S. Eisenman, N. Lane and A. Campbell. Metrosense project: People-centric sensing at scale. In Workshop on World-Sensor-Web (WSW 2006), Boulder, Οκτώβριος 2006.

[24] K. Chen M. Goraczko A. Miu E. Shih Y. Zhang H. Balakrishnan B. Hull, V. Bychkovsky and S. Madden. Cartel: A distributed mobile sensor computing system. In the Proceedings of SenSys '06, 2006.

[25] Ogc sensor web enablement, overview and highlevel architecture. OpenGIS white paper, OGC 06 - 050r2.

[26] M. Giorgetta A. Giusti A. Murphy C. Curino, M. Giani and G. Picco. Tinylime: Bridging mobile and sensor networks through middleware. Third IEEE International Conference on Pervasive Computing and Communications, PerCom 2005:61––72.

[27] B. Karp P. Gibbons and S. Nath S. Seshan Y. Ke. Irisnet: An architecture for a world-wide sensor web. IEEE Pervasive Computing 2, no. 4.:29––42.

[28] G. Roman G. Hackmann, C. Fok and Agimone C. Lu. Middleware support for seamless integration of sensor and ip networks. In the Proceedings of 2006 International Conference on Distributed Computing in Sensor Systems (DCOSS '06), 2006.

[29] Arch rock, a new embedded web services experience for wireless sensor networks. In the Proc. of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06), 2006.

[30] The octavex platform, octave technology inc. URL http://www.octavetech.com/solutions/octavex.html.

[31] Sensilink wsn middleware platform, meshnetics. URL http://www.meshnetics.com.

[32] Synapsense oneclick wsn software architecture. URL http://www.synapsense.com.

[33] Project jxta. URL http://www.xbow.com.

[34] Moteworks software platform. URL http://www.jxta.org/.

[35] T. Luckenbach et al. Tinyrest - a protocol for integrating sensor networks into the internet. In the Proceedings of the First REALWSN 2005Workshop on Real-World Wireless Sensor Networks, 2005.

[36] N. Sastry C. Karlof and D. Wagner. Tinysec: A link layer security architecture for wireless sensor networks. In the Proc. of the 2nd ACM Conference on Embedded Networked Sensor Systems (SensSys 2004), 2004.