ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΑΤΡΩΝ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΙΑΣ

ΑΡΙΘΜΟΣ: 1321

Σχεδίαση και ανάπτυξη πλατφόρμας απομακρυσμένης εποπτείας (βίντεο) μέσω του διαδικτύου. Design and development platform for remote surveillance (video) over the Internet.

ΣΠΟΥΔΑΣΤΗΣ:

ΝΤΟΡΛΗΣ ΘΕΟΔΩΡΟΣ

ΕΙΣΗΓΗΤΗΣ(ΕΣ}:

ΚΑΨΑΛΗΣ ΒΑΣΙΛΕΙΟΣ

ΤΟΠΑΛΗΣ ΕΥΑΓΓΕΛΟΣ

ΠΑΤΡΑ 2013

ΠΡΟΛΟΓΟΣ

Η πτυχιακή αυτή εργασία αποτελεί την κορύφωση των σπουδών μου στο Τ.Ε.Ι. Πατρών στη σχολή τεχνολογικών εφαρμογών του τμήματος Ηλεκτρολογίας και αποτελεί εργασία σχεδιασμού και ανάπτυξης, μίας πλατφόρμας απομακρυσμένης εποπτείας μέσω του διαδικτύου. Θα ήθελα να ευχαριστήσω του γονείς μου και τον αδερφό μου για όλα όσα μου έχουν προσφέρει στην διάρκεια των μαθητικών και φοιτητικών μου χρόνων και την αμέριστη υποστήριξη τους σε κάθε μου επιλογή. Επίσης θέλω να ευχαριστήσω και τους επιβλέποντες καθηγητές μου κ. Καψάλη Βασίλειο και Τοπάλη Ευάγγελο για την εμπιστοσύνη τους.

Στο πρώτο κεφάλαιο γίνεται μία εισαγωγή στην γλώσσα προγραμματισμού της Java η οποία χρησιμοποιήθηκε για την ανάπτυξη του προγράμματος, και στις βάσεις δεδομένων. Στο δεύτερο κεφάλαιο περιγράφεται η δομή και η λειτουργία του Server, επίσης στο τρίτο κεφάλαιο περιγράφεται η δομή και η λειτουργία του Applet, στο τέταρτο κεφάλαιο περιγράφεται η αρχιτεκτονική των ιστοσελίδων. Τέλος στο πέμπτο κεφάλαιο εξηγώ την λειτουργία των δυο εφαρμογών και τις απαιτήσεις.

ΠΕΡΙΛΗΨΗ

Στόχος της πτυχιακής είναι η σχεδίαση και ανάπτυξη μιας πλατφόρμας απομακρυσμένης εποπτείας εσωτερικού ή εξωτερικού χώρου που θα παρέχει την εξής λειτουργικότητα:

Προβολή streaming video σε πραγματικό χρόνο, δυνατότητα εγγραφής βίντεο σε πραγματικό χρόνο στον server ή στον client, δυνατότητα χρονοπρογραμματισμού, έλεγχος κίνησης της κάμερας, παρακολούθηση της θερμοκρασίας και της φωτεινότητας και τέλος η δημιουργία κατάλληλων αναφορών. Για την ανάπτυξη της πλατφόρμας θα αναπτυχθούν δύο εφαρμογές με τη χρήση της γλώσσας προγραμματισμού Java. Η μια εφαρμογή θα είναι ο server, που θα συνδέεται με την κάμερα και θα λαμβάνει τις τιμές των μετρούμενων μεγεθών από το δίκτυο LonWorks, ο οποίος θα στέλνει video και δεδομένα σε απομακρυσμένους clients. Η δεύτερη εφαρμογή θα είναι ο client ο οποίος θα αποτελεί ένα Java Applet το οποίο θα είναι ενσωματωμένο σε μία ιστοσελίδα και θα παρέχει τη διεπαφή ελέγχου του χρήστη, για τον έλεγχο των παρεχόμενων λειτουργιών και την απεικόνιση του video. Η εφαρμογή θα επιτρέπει διαβαθμισμένη πρόσβαση στις λειτουργίες της με προστασία κωδικού, για δύο κατηγορίες χρηστών: διαχειριστής και απλός χρήστης. Η επικοινωνία ανάμεσα στον server και τον client θα βασίζεται στα πρωτόκολλα HTTP, TCP και UDP.

Περιεχόμενα

ΕΙΣΑΓΩΓΗ

Επιτήρηση είναι η παρακολούθηση της συμπεριφοράς, των δραστηριοτήτων, ή άλλων πληροφοριών που αλλάζουν, συνήθως από ανθρώπους με σκοπό τη διαχείριση, την καθοδήγηση, ή την προστασία τους. Αυτό μπορεί να περιλαμβάνει παρατήρηση από απόσταση με τη βοήθεια του ηλεκτρονικού εξοπλισμού (όπως κάμερες CCTV), ή την υποκλοπή των πληροφοριών που διαβιβάζονται ηλεκτρονικά (όπως η κίνηση στο Internet ή τηλεφώνου κλήσεις). Η παρακολούθηση μπορεί να αναφέρεται σε σχετικά χαμηλής τεχνολογίας μεθόδους όπως οι πράκτορες μυστικών υπηρεσιών και οι ταχυδρομικές παρακολουθήσεις. Η επιτήρηση είναι πολύ χρήσιμη για τις κυβερνήσεις και την επιβολή του νόμου για τη διατήρηση του κοινωνικού ελέγχου, να αναγνωρίζουν και να παρακολουθούν τις απειλές, και την πρόληψη / διερεύνηση της εγκληματικής δραστηριότητας. Κάμερες παρακολούθησης είναι οι κάμερες που χρησιμοποιούνται για το σκοπό της παρατήρησης μιας περιοχής. Συχνά είναι συνδεδεμένες με μια συσκευή εγγραφής ή στο δίκτυο, και μπορεί να παρακολουθούνται από φύλακα ή υπάλληλο επιβολής του νόμου. Οι κάμερες και ο εξοπλισμός καταγραφής που χρησιμοποιούνται είναι σχετικά ακριβά και απαιτούν ανθρώπινο προσωπικό για την παρακολούθηση τους, αλλά η ανάλυση του βίντεο έχει γίνει ευκολότερη με αυτοματοποιημένο λογισμικό που οργανώνει ψηφιακά τα βίντεο σε μια βάση δεδομένων, καθώς και από λογισμικό ανάλυσης βίντεο (όπως VIRAT και HumanID). Το ποσό του βίντεο έχει επίσης μειωθεί δραστικά από αισθητήρες κίνησης που καταγράφουν μόνο όταν ανιχνεύεται κίνηση. Με φθηνότερες τεχνικές παραγωγής, οι κάμερες παρακολούθησης είναι απλές και ανέξοδες ώστε να μπορούν να χρησιμοποιηθούν στο σπίτι στα συστήματα ασφαλείας, καθώς και για την καθημερινή εποπτεία. Στις Ηνωμένες Πολιτείες, το Υπουργείο Εσωτερικής Ασφάλειας δίνει σε βραβεία δισεκατομμύρια δολάρια ετησίως σε Εσωτερικής Ασφάλειας επιχορηγήσεις για τις τοπικές, πολιτειακές και ομοσπονδιακές υπηρεσίες για την εγκατάσταση σύγχρονου εξοπλισμού παρακολούθησης βίντεο. Η ανάπτυξη των κεντρικών δικτύων καμερών CCTV η οποία επιβλέπει τους κοινόχρηστους χώρους, συνδέεται με βάσεις δεδομένων που περιέχουν εικόνες και την ταυτότητα των ανθρώπων (βιομετρικά δεδομένα) και είναι σε θέση να παρακολουθούν τις κινήσεις των ανθρώπων σε όλη την πόλη, και να τους εντοπίζουν. Trapwire είναι ένα παράδειγμα ενός τέτοιου δικτύου.

ΚΕΦΑΛΑΙΟ 1

1.1 Εισαγωγή στην Java

1.1.1 Ιστορία

Στις αρχές του 1991, η *Sun* αναζητούσε το κατάλληλο εργαλείο για να αποτελέσει την πλατφόρμα ανάπτυξης λογισμικού σε μικρό-συσκευές (έξυπνες οικιακές συσκευές έως πολύπλοκα συστήματα παραγωγής γραφικών). Τα εργαλεία της εποχής ήταν γλώσσες όπως η C++ και η C. Μετά από διάφορους πειραματισμούς προέκυψε το συμπέρασμα ότι οι υπάρχουσες γλώσσες δεν μπορούσαν να καλύψουν τις ανάγκες τους. Ο "πατέρας" της Java, James Gosling, που εργαζόταν εκείνη την εποχή για την Sun, έκανε ήδη πειραματισμούς πάνω στη C++ και είχε παρουσιάσει κατά καιρούς κάποιες πειραματικές γλώσσες (C++ ++) ως πρότυπα για το νέο εργαλείο που αναζητούσαν στην *Sun*. Τελικά μετά από λίγο καιρό κατέληξαν με μια πρόταση για το επιτελείο της εταιρίας, η οποία ήταν η γλωσσά *Oak*. Το όνομά της το πήρε από το ομώνυμο δένδρο (βελανιδιά) το οποίο ο Gosling είχε έξω από το γραφείο του και έβλεπε κάθε μέρα.

1.1.2 Από την Oak στη Java

Η Oak ήταν μία γλώσσα που διατηρούσε μεγάλη συγγένεια με την C++. Παρόλα αυτά είχε πολύ πιο έντονο αντικειμενοστραφή (*object oriented*) χαρακτήρα σε σχέση με την C++ και χαρακτηριζόταν για την απλότητα της. Σύντομα οι υπεύθυνοι ανάπτυξης της νέας γλώσσας ανακάλυψαν ότι το όνομα *Oak* ήταν ήδη κατοχυρωμένο οπότε κατά την διάρκεια μιας εκ των πολλών συναντήσεων σε κάποιο τοπικό καφέ αποφάσισαν να μετονομάσουν το νέο τους δημιούργημα σε Java που εκτός των άλλων ήταν το όνομα της αγαπημένης ποικιλίας καφέ για τους δημιουργούς της. Η επίσημη εμφάνιση της *Java* αλλά και του *HotJava* (πλοηγός με υποστήριξη *Java*) στη βιομηχανία της πληροφορικής έγινε το Μάρτιο του 1995 όταν η *Sun* την ανακοίνωσε στο συνέδριο *Sun World* 1995. Ο πρώτος μεταγλωττιστής (*compiler*) της ήταν γραμμένος στη γλώσσα C από τον James Gosling. Το1994, ο A.Van Hoff ξαναγράφει τον μεταγλωττιστή της γλώσσας σε *Java*, ενώ το Δεκέμβριο του 1995 πρώτες οι IBM, Borland, Mitsubishi Electronics, Sybase και Symantec ανακοινώνουν σχέδια να χρησιμοποιήσουν τη *Java* για την δημιουργία λογισμικού. Από εκεί και πέρα η Java ακολουθεί μία ανοδική πορεία και είναι πλέον μία από τις πιο δημοφιλείς γλώσσες στον χώρο της πληροφορικής. Στις 13 Νοεμβρίου του 2006 η Java έγινε πλέον μια γλώσσα ανοικτού κώδικα (GPL) όσον αφορά το μεταγλωττιστή (javac) και το πακέτο ανάπτυξης (JDK, Java Development Kit).

1.1.3 Η εξαγορά από την Oracle και το μέλλον της Java

Στις 27 Απριλίου 2010 η εταιρία λογισμικού Oracle Corporation ανακοίνωσε ότι μετά από πολύμηνες συζητήσεις ήρθε σε συμφωνία για την εξαγορά της Sun Microsystems και των τεχνολογιών (πνευματικά δικαιώματα/ πατέντες) που η δεύτερη είχε στην κατοχή της ή δημιουργήσει. Η συγκεκριμένη συμφωνία θεωρείται σημαντική για το μέλλον της Java και του γενικότερου οικοσυστήματος τεχνολογιών γύρω από αυτή μιας και ο έμμεσος έλεγχος της τεχνολογίας και η εξέλιξη της περνάει σε άλλα χέρια.

1.1.4 Τα χαρακτηριστικά της Java

Ένα από τα βασικά πλεονεκτήματα της Java έναντι των περισσότερων άλλων γλωσσών είναι η ανεξαρτησία του λειτουργικού συστήματος και πλατφόρμας. Τα προγράμματα που είναι γραμμένα σε *Java* τρέχουνε ακριβώς το ίδιο σε Windows, Linux, Unix και Macintosh (σύντομα θα τρέχουν και σε Playstation καθώς και σε άλλες κονσόλες παιχνιδιών) χωρίς να χρειαστεί να ξαναγίνει μεταγλώττιση (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα. Για να επιτευχθεί όμως αυτό χρειαζόταν κάποιος τρόπος έτσι ώστε τα προγράμματα γραμμένα σε Java να μπορούν να είναι «κατανοητά» από κάθε υπολογιστή ανεξάρτητα του είδους επεξεργαστή (Intel x86, IBM, Sun SPARC, Motorola) αλλά και λειτουργικού συστήματος (Windows, Unix, Linux, BSD, MacOS). Ο λόγος είναι ότι κάθε κεντρική μονάδα επεξεργασίας κατανοεί διαφορετικό κώδικα μηχανής. Ο συμβολικός κώδικας (*assembly*) που μεταφράζεται και εκτελείται σε Windows είναι διαφορετικός από αυτόν που μεταφράζεται και εκτελείται σε έναν υπολογιστή Macintosh. Η λύση δόθηκε με την ανάπτυξη της *Εικονικής Μηχανής* (*Virtual Machine* ή VM ή ΕΜ στα ελληνικά).

1.1.5 Η εικονική μηχανή της Java

Αφού γραφεί κάποιο πρόγραμμα σε Java, στη συνέχεια μεταγλωττίζεται μέσω του μεταγλωττιστή javac, ο οποίος παράγει έναν αριθμό από αρχεία .class (κώδικας byte ή bytecode). Ο κώδικας byte είναι η μορφή που παίρνει ο πηγαίος κώδικας της Java όταν μεταγλωττιστεί. Όταν πρόκειται να εκτελεστεί η εφαρμογή σε ένα μηχάνημα, το Java Virtual Machine που πρέπει να είναι εγκατεστημένο σε αυτό θα αναλάβει να διαβάσει τα αρχεία .class. Στη συνέχεια τα μεταφράζει σε γλώσσα μηχανής που να υποστηρίζεται από το λειτουργικό σύστημα και τον επεξεργαστή, έτσι ώστε να εκτελεστεί (να σημειωθεί εδώ ότι αυτό συμβαίνει με την παραδοσιακή Εικονική Μηχανή (Virtual Machine). Πιο σύγχρονες εφαρμογές της εικονικής Μηχανής μπορούν και μεταγλωττίζουν εκ των προτέρων τμήματα bytecode απευθείας σε κώδικα μηχανής (εγγενή κώδικα ή native code) με αποτέλεσμα να βελτιώνεται η ταχύτητα). Χωρίς αυτό δε θα ήταν δυνατή η εκτέλεση λογισμικού γραμμένου σε Java. Πρέπει να σημειωθεί ότι η JVM είναι λογισμικό που εξαρτάται από την πλατφόρμα, δηλαδή για κάθε είδος λειτουργικού συστήματος και αρχιτεκτονικής επεξεργαστή υπάρχει διαφορετική έκδοση του. Έτσι υπάρχουν διαφορετικές JVM για Windows, Linux, Unix, Macintosh, κινητά τηλέφωνα, παιχνιδομηχανές κλπ. Οτιδήποτε θέλει να κάνει ο προγραμματιστής (ή ο χρήστης) γίνεται μέσω της εικονικής μηχανής. Αυτό βοηθάει στο να υπάρχει μεγαλύτερη ασφάλεια στο σύστημα γιατί η εικονική μηχανή είναι υπεύθυνη για την επικοινωνία χρήστη - υπολογιστή. Ο προγραμματιστής δεν μπορεί να γράψει κώδικα ο οποίος θα έχει καταστροφικά αποτελέσματα για τον υπολογιστή γιατί η εικονική μηχανή θα τον ανιχνεύσει και δε θα επιτρέψει να εκτελεστεί. Από την άλλη μεριά ούτε ο χρήστης μπορεί να κατεβάσει «κακό» κώδικα από το δίκτυο και να τον εκτελέσει. Αυτό είναι ιδιαίτερα χρήσιμο για μεγάλα κατανεμημένα συστήματα όπου πολλοί χρήστες χρησιμοποιούν το ίδιο πρόγραμμα συγχρόνως.

1.1.6 Ο συλλέκτης απορριμμάτων (Garbage Collector)

Ακόμα μία ιδέα που βρίσκεται πίσω από τη *Java* είναι η ύπαρξη του συλλέκτη απορριμμάτων (*Garbage Collector*). Συλλογή απορριμμάτων είναι μία κοινή ονομασία που χρησιμοποιείται στον τομέα της πληροφορικής για να δηλώσει την ελευθέρωση τμημάτων μνήμης από δεδομένα που δε χρειάζονται και δε χρησιμοποιούνται άλλο. Αυτή η απελευθέρωση μνήμης στη Java είναι αυτόματη και γίνεται μέσω του συλλέκτη απορριμμάτων. Υπεύθυνη για αυτό είναι και πάλι η εικονική μηχανή η οποία μόλις «καταλάβει» ότι ο σωρός (heap) της μνήμης (στη Java η συντριπτική πλειοψηφία των αντικειμένων αποθηκεύονται στο σωρό σε αντίθεση με τη C++ όπου αποθηκεύονται κυρίως στη στοίβα) κοντεύει να γεμίσει ενεργοποιεί το συλλέκτη απορριμμάτων. Έτσι ο προγραμματιστής δε χρειάζεται να ανησυχεί για το πότε και αν θα ελευθερώσει ένα συγκεκριμένο τμήμα της μνήμης, ούτε και για σφάλματα δεικτών. Αυτό είναι ιδιαίτερα σημαντικό γιατί είναι κοινά τα σφάλματα προγραμμάτων που οφείλονται σε λανθασμένο χειρισμό της μνήμης.

1.1.7 Επιδόσεις

Παρόλο που η εικονική μηχανή προσφέρει όλα αυτά (και όχι μόνο) τα πλεονεκτήματα, η *Java* αρχικά ήταν πιο αργή σε σχέση με άλλες προγραμματιστικές γλώσσες υψηλού επιπέδου (high-level) όπως η C και Η C++. Εμπειρικές μετρήσεις στο παρελθόν είχαν δείξει ότι η C++ μπορούσε να είναι αρκετές φορές γρηγορότερη από την Java. Ωστόσο γίνονται προσπάθειες από τη Sun για τη βελτιστοποίηση της εικονικής μηχανής, ενώ υπάρχουν και άλλες υλοποιήσεις της εικονικής μηχανής από διάφορες εταιρίες (όπως της IBM), οι οποίες μπορεί σε κάποια σημεία να προσφέρουν καλύτερα και σε κάποια άλλα χειρότερα αποτελέσματα. Επιπλέον με την καθιέρωση των μεταγλωττιστών JIT (Just In Time), οι οποίοι μετατρέπουν τον κώδικα byte απευθείας σε γλώσσα μηχανής, η διαφορά ταχύτητας από τη C++ έχει μικρύνει κατά πολύ.Οι τελευταίες εκδόσεις του javac με τη χρήση της τεχνολογίας Hot Spot έχουν καταφέρει αξιόλογες επιδόσεις που πλησιάζουν ή και ξεπερνούν σε μερικές περιπτώσεις τον εγγενή κώδικα.

1.1.8 Εργαλεία ανάπτυξης

Όλα τα εργαλεία που χρειάζεται κάποιος για να γράψει Java προγράμματα έρχονται δωρεάν, από το περιβάλλον ανάπτυξης μέχρι εργαλεία *build* όπως το Apache Ant και βιβλιοθήκες, ενώ υπάρχουν πολλές διαφορετικές υλοποιήσεις της *Εικονικής Μηχανής* και του *μεταγλωττιστή* (πχ the GNU Compiler for Java) της *Java*. Πολλά εργαλεία και τεχνολογίες σε Java μπορούν να βρεθούν στο Apache Software Foundation αλλά και στο Jakarta Project.

1.1.9 Ολοκληρωμένο περιβάλλον ανάπτυξης (IDE)

Για να γράψει κάποιος κώδικα Java δε χρειάζεται τίποτα άλλο παρά έναν επεξεργαστή κειμένου, όπως το Σημειωματάριο (Notepad) των Windows ή ο vi (γνωστός στο χώρο του Unix). Παρ'όλ'αυτά, ένα ολοκληρωμένο περιβάλλον ανάπτυξης (*IDE*) βοηθάει πολύ, ιδιαίτερα στον εντοπισμό σφαλμάτων (debugging). Υπάρχουν αρκετά διαθέσιμα, ενώ πολλά από αυτά έρχονται δωρεάν όπως το Eclipse που χρησιμοποίησα για την δημιουργία της πτυχιακής.

1.2 Εισαγωγή στις βάσεις Δεδομένων

Με τον όρο **βάση δεδομένων** εννοείται μία συλλογή από *συστηματικά οργανωμένα*

(formatted) σχετιζόμενα δεδομένα. Ένας τηλεφωνικός κατάλογος, για παράδειγμα, θεωρείται βάση δεδομένων, καθώς αποθηκεύει και οργανώνει σχετιζόμενα τμήματα πληροφορίας, όπως είναι το όνομα και ο αριθμός τηλεφώνου. Ωστόσο, στον κόσμο των υπολογιστών, με τον όρο βάση δεδομένων αναφερόμαστε σε μια συλλογή σχετιζόμενων δεδομένων τμημάτων πληροφορίας ηλεκτρονικά αποθηκευμένων. Αφού δημιουργήσουμε μια βάση δεδομένων με πολλές εγγραφές, τότε μπορούμε σε αυτήν να ψάξουμε και να βρούμε εύκολα και γρήγορα το/τα στοιχείο/α που επιθυμούμε. Η ηλεκτρονική βάση δεδομένων χρησιμοποιεί ιδιαίτερου τύπου λογισμικό προκειμένου να οργανώσει την αποθήκευση των δεδομένων της. Ένα πρόγραμμα που διαχειρίζεται βάσεις δεδομένων αποκαλείται *Σύστημα Διαχείρισης Βάσεων Δεδομένων (**DBMS**, DataBase Management System)*το οποίο είναι ένα σύνολο από προγράμματα που επιτρέπουν τον χειρισμό των δεδομένων μιας ή περισσοτέρων βάσεων δεδομένων που ανήκουν στο ίδιο σύστημα, και με την βοήθειά του μπορούμε να υποθηκεύσουμε, προσθέσουμε, τροποποιήσουμε, εμφανίσουμε ή και διαγράψουμε τα αποθηκευμένα δεδομένα.
Τα δεδομένα που υπάρχουν στις βάσεις δεδομένων πρέπει να είναι :
• ***Ολοκληρωμένα** (Integrated)*, δηλ. τα δεδομένα πρέπει να είναι αποθηκευμένα σε ομοιόμορφα οργανωμένα σύνολα αρχείων όπου δεν πρέπει να υπάρχει επανάληψη ή πλεονασμός (redundancy) των ίδιων στοιχείων.
• ***Καταμεριζόμενα** (Shared)*, δηλ. να μπορούν περισσότεροι του ενός χρήστες να βλέπουν και να μοιράζονται τα ίδια δεδομένα την ίδια χρονική στιγμή.
Οι στόχοι μιας βάσης δεδομένων είναι οι εξής :
• **Ο περιορισμός της πολλαπλής αποθήκευσης των ίδιων στοιχείων (redundancy).** Εάν τα ίδια δεδομένα καταχωρηθούν στη βάση δυο φορές, τότε ανακύπτουν δυο σοβαρά προβλήματα. Το πρώτο πρόβλημα είναι ότι σπαταλούμε άσκοπα αποθηκευτικό χώρο στο σκληρό δίσκο, αφού την ίδια πληροφορία την αποθηκεύουμε δυο φορές. Το δεύτερο και σοβαρότερο πρόβλημα, είναι ότι υπάρχει ο κίνδυνος δημιουργίας ασυνεπών δεδομένων (inconsistent data). Εάν κρατάμε δύο φορές την ίδια πληροφορία και η πληροφορία αυτή σε κάποια χρονική στιγμή υποστεί κάποιο είδος επεξεργασίας τότε η επεξεργασία αυτή θα πρέπει να εφαρμοσθεί και στις δύο καταχωρήσεις που αφορούν το ίδιο δεδομένο, διότι διαφορετικά, η βάση θα περιέχει δεδομένα που δεν είναι συνεπή. Για το λόγω αυτό, ένας από τους πρώτους ελέγχους που πραγματοποιούμε στη βάση αμέσως μετά το σχεδιασμό της, είναι ο έλεγχος παρουσίας επαναλαμβανόμενων πεδίων, και η απομάκρυνσή τους, εφ' όσον υπάρχουν**.**
• **Ο καταμερισμός (sharing) των ίδιων στοιχείων σ' όλους τους χρήστες.**
• **Η ομοιομορφία (uniformity) στον χειρισμό και την αναπαράσταση των δεδομένων**.
• **Η επιβολή κανόνων ασφαλείας (security).** Όπως να απαγορεύει την πρόσβαση στα δεδομένα μη εξουσιοδοτημένων ατόμων. Αυτό ισχύει κυρίως σε μεγάλες βάσεις δεδομένων με πολλούς χρήστες, και η τεχνική που συνίσταται είναι ο καθορισμός ομάδων χρηστών (user groups) με διαφορετικά δικαιώματα πρόσβασης στον καθένα από αυτούς. Ο κάθε χρήστης λαμβάνει ένα κωδικό πρόσβασης (password) και τα καθήκοντα που μπορεί να επιτελέσει είναι εντελώς συγκεκριμένα και καθορισμένα εκ των προτέρων.
• **Η διατήρηση της ακεραιότητας (integrity) και της αξιοπιστίας (reliability) των δεδομένων.**
Η βάση θα πρέπει να διαθέτει σύστημα δημιουργίας αντιγράφων ασφαλείας των δεδομένων που είναι καταχωρημένα σε αυτή (backups). Η ταυτόχρονη αποθήκευση των δεδομένων σε περισσότερους από ένα δίσκους, είναι μια εργασία επιβεβλημένη, προκειμένου να είναι δυνατή η ανάκτησή τους σε περιπτώσεις κατάρρευσης της βάσης για οποιοδήποτε λόγω**.**
• **Η ανεξαρτησία των δεδομένων (data independence) και των προγραμμάτων από τον φυσικό τρόπο αποθήκευσης των δεδομένων.**

Τα δεδομένα μιας βάσης δεδομένων αποθηκεύονται (οργανώνονται) στις εξής στοιχειώδεις μορφές :

• ***Πεδίο*** *(Field)*, είναι το μικρότερο κομμάτι δεδομένων στο οποίο μπορούμε να αναφερθούμε και περιέχει ένα μόνο χαρακτηριστικό ή ιδιότητα ενός στοιχείου της βάσης δεδομένων. Ένα πεδίο χαρακτηρίζεται ακόμη και από το είδος των δεδομένων που μπορεί να περιέχει, όπως :

Αλφαριθμητικό (alphanumeric), μπορεί να περιέχει γράμματα, ψηφία ή και ειδικούς χαρακτήρες.

Αριθμητικό (numeric), μπορεί να περιέχει μόνο αριθμούς.

Αλφαβητικό (alphabetic), μπορεί να περιέχει μόνο γράμματα (αλφαβητικούς χαρακτήρες).

Ημερομηνίας (date), μπορεί να περιέχει μόνο ημερομηνίες.

Δυαδικό (binary), μπορεί να περιέχει ειδικού τύπου δεδομένα, όπως εικόνες, ήχους κ.ά.

Λογικό (logical), μπορεί να περιέχει μόνο μία από δύο τιμές, οι οποίες αντιστοιχούν σε δύο διακριτές καταστάσεις και μπορούν να χαρακτηρισθούν σαν 0 και 1 ή σαν αληθές (true) και ψευδές (false).

Σημειώσεων (memo), μπορεί να περιέχει κείμενο με μεταβλητό μήκος, το οποίο μπορεί να είναι και αρκετά μεγάλο και συνήθως αποθηκεύεται σαν ξεχωριστό αρχείο από το κύριο αρχείο.

• ***Εγγραφή*** *(Record)*, είναι ένα σύνολο από διαφορετικά πεδία που περιέχει όλες τις πληροφορίες για ένα στοιχείο της βάσης δεδομένων.

Όσον αφορά τις εγγραφές, χρήσιμοι ορισμοί είναι οι εξής :

Μήκος εγγραφής (record length) καλείται το άθροισμα που προκύπτει από τα μήκη των πεδίων που την αποτελούν.

Δομή εγγραφής (record layout) ή *γραμμογράφηση* καλείται ο τρόπος με τον οποίο οργανώνουμε τα πεδία μιας εγγραφής.

Διάβασμα (read) από αρχείο σημαίνει τη μεταφορά των δεδομένων του αρχείου, που γίνεται συνήθως ανά μία εγγραφή, από το μέσο αποθήκευσης (σκληρό δίσκο ή δισκέτα) στην κεντρική μνήμη του υπολογιστή για επεξεργασία.

Γράψιμο (write) σε αρχείο σημαίνει μεταφορά των δεδομένων του αρχείου, που γίνεται συνήθως ανά μία εγγραφή, από την κεντρική μνήμη του υπολογιστή στο μέσο αποθήκευσης (σκληρό δίσκο ή δισκέτα).

• ***Αρχείο*** *(File)*, είναι ένα σύνολο από πολλά παρόμοια στοιχεία (εγγραφές) της βάσης δεδομένων.

• ***Πρωτεύον Κλειδί*** *(Primary Key)*, είναι ένα πεδίο ή συνδυασμός πεδίων που χαρακτηρίζει μοναδικά μια εγγραφή.

• ***Κλειδί*** *(Key)*, είναι ένα πεδίο που δεν έχει κατ' ανάγκη μοναδική τιμή και που μπορούμε να το χρησιμοποιήσουμε για να κάνουμε αναζήτηση σ' ένα αρχείο.

• ***Ξένο Κλειδί*** *(Foreign Key)*, είναι μια ιδιότητα (πεδίο) που είναι πρωτεύον κλειδί σε μια οντότητα (πίνακας) αλλά που υπάρχει και σε μια άλλη οντότητα (πίνακας) σαν απλή ιδιότητα. Τα ξένα κλειδιά είναι απαραίτητα για να μπορέσουμε να κάνουμε τις συσχετίσεις (συνδέσεις, επικοινωνίες) ανάμεσα στις οντότητες (πίνακες).

Οι βάσεις δεδομένων είναι σημαντικές διότι δίνουν στη σελίδα μας την έννοια της “εφαρμογής”. Δηλαδή η σελίδα μας δεν είναι απλά μία στατική σελίδα που απλά μας πληροφορεί για κάτι αλλά μπορούμε να αποθηκεύουμε δεδομένα, να τα επεξεργαζόμαστε να τα χρησιμοποιούμε γενικότερα. Έτσι, η σελίδα μας γίνεται μία δυναμική “σελίδα”. Εκατομμύρια sites χρησιμοποιούν βάσεις δεδομένων. Για παράδειγμα amazon, ebay, facebook κλπ. Ολόκληρες πλατφόρμες βασίζονται σε βάσεις δεδομένων όπως η πλατφόρμα blogging “blogspot” ,”wordpress” κλπ. Όπως αναφέραμε και παραπάνω το ΣΔΒΔ είναι ένα σύνολο από προγράμματα και υπορουτίνες που έχουν να κάνουν με τον χειρισμό της βάσης δεδομένων, όσον αφορά τη δημιουργία, τροποποίηση, διαγραφή στοιχείων, με ελέγχους

ασφαλείας κ.ά. Οι χρήστες των εφαρμογών αντλούν τα στοιχεία που τους ενδιαφέρουν από τη βάση δεδομένων χωρίς να είναι σε θέση να γνωρίζουν με ποιο τρόπο είναι οργανωμένα τα δεδομένα σ'αυτήν. Το ΣΔΒΔ παίζει τον ρόλο του μεσάζοντα ανάμεσα στον χρήστη και τη βάση δεδομένων και μόνο μέσω του ΣΔΒΔ μπορεί ο χρήστης να αντλήσει πληροφορίες από τη βάση δεδομένων. Ένα ΣΔΒΔ μπορεί να είναι εγκατεστημένο σ' έναν μόνο υπολογιστή ή και σ' ένα δίκτυο υπολογιστών και μπορεί να χρησιμοποιείται από έναν χρήστη ή και από πολλούς χρήστες. Ένα *Σύστημα Βάσης Δεδομένων (ΣΒΔ)* ή *DBS (Data Base System)* αποτελείται από το υλικό, το λογισμικό, τη βάση δεδομένων και τους χρήστες. Είναι δηλαδή ένα σύστημα με το οποίο μπορούμε να αποθηκεύσουμε και να αξιοποιήσουμε δεδομένα με τη βοήθεια ηλεκτρονικού υπολογιστή.
Αναλυτικά :
• Το *υλικό (hardware)* αποτελείται όπως είναι γνωστό από τους ηλεκτρονικούς υπολογιστές, τα περιφερειακά, τους σκληρούς δίσκους, τις μαγνητικές ταινίες κ.ά., όπου είναι αποθηκευμένα τα αρχεία της βάσης δεδομένων αλλά και τα προγράμματα που χρησιμοποιούνται για την επεξεργασία τους.
• Το *λογισμικό (software)* είναι τα προγράμματα που χρησιμοποιούνται για την επεξεργασία των δεδομένων (στοιχείων) της βάσης δεδομένων.
• Η *βάση δεδομένων (data base)* αποτελείται από το σύνολο των αρχείων όπου είναι αποθηκευμένα τα δεδομένα του συστήματος. Τα στοιχεία αυτά μπορεί να βρίσκονται αποθηκευμένα σ' έναν φυσικό υπολογιστή αλλά και σε περισσότερους. Όμως, στον χρήστη δίνεται η εντύπωση ότι βρίσκονται συγκεντρωμένα στον ίδιο υπολογιστή. Τα δεδομένα των αρχείων αυτών είναι *ενοποιημένα (data integration)*, δηλ. δεν υπάρχει πλεονασμός (άσκοπη επανάληψη) δεδομένων και *μερισμένα (data sharing)*, δηλ. υπάρχει δυνατότητα ταυτόχρονης προσπέλασης των δεδομένων από πολλούς χρήστες. Ο κάθε χρήστης έχει διαφορετικά δικαιώματα και βλέπει διαφορετικό κομμάτι της βάσης δεδομένων, ανάλογα με τον σκοπό για τον οποίο συνδέεται.
• Οι *χρήστες (users)* μιας βάσης δεδομένων χωρίζονται στις εξής κατηγορίες :
• *Τελικοί χρήστες (end users)*. Χρησιμοποιούν κάποια εφαρμογή για να παίρνουν στοιχεία από μια βάση δεδομένων, έχουν τις λιγότερες δυνατότητες επέμβασης στα στοιχεία της βάσης δεδομένων, χρησιμοποιούν ειδικούς κωδικούς πρόσβασης και το σύστημα τούς επιτρέπει ανάλογα πρόσβαση σε συγκεκριμένο κομμάτι της βάσης δεδομένων.
• *Προγραμματιστές εφαρμογών (application programmers)*. Αναπτύσσουν τις εφαρμογές του ΣΒΔ σε κάποια από τις γνωστές γλώσσες προγραμματισμού.
• *Διαχειριστής δεδομένων (data administrator – DA)*. Έχει τη διοικητική αρμοδιότητα και ευθύνη για την οργάνωση της βάσης δεδομένων και την απόδοση δικαιωμάτων πρόσβασης στους χρήστες.
• *Διαχειριστής βάσης δεδομένων (database administrator – DBA)*. Λαμβάνει οδηγίες από τον διαχειριστή δεδομένων και είναι αυτός που διαθέτει τις τεχνικές γνώσεις και αρμοδιότητες για τη σωστή και αποδοτική λειτουργία του ΣΔΒΔ.

1.3 Η αρχιτεκτονική των ΣΔΒΔ

Όπως είδαμε νωρίτερα, ένα ΣΔΒΔ (Σύστημα Διαχείρισης Βάσης Δεδομένων) έχει σαν αποστολή τη διαχείριση των δεδομένων των αρχείων της βάσης, δηλ. την προσθήκη, διαγραφή, τροποποίηση εγγραφών, την αναζήτηση μέσα στις εγγραφές κ.ά.). Το ΣΔΒΔ δέχεται αιτήσεις από τους χρήστες των εφαρμογών και επικοινωνεί με τα αρχεία της βάσης δεδομένων για να τις διεκπεραιώσει. Αυτή η κοινή διεπαφή (interface) των εφαρμογών με τα αρχεία αποκαλείται *λογική διεπαφή*. Οι εφαρμογές που δημιουργούμε δεν απασχολούνται με

τον τρόπο που είναι αποθηκευμένα τα δεδομένα, πόσο χώρο καταλαμβάνουν κοκ και αυτή η ιδιότητα είναι γνωστή ως *ανεξαρτησία δεδομένων*. Αυτό σημαίνει πρακτικά ότι οποιαδήποτε αλλαγή στον τρόπο οργάνωσης των αρχείων της βάσης δεδομένων δεν θα συνεπάγεται και αλλαγή στις εφαρμογές· ένα πρόβλημα που ταλαιπωρούσε πολύ τους προγραμματιστές παλαιοτέρων εποχών. Ακόμη, η προσθήκη, η κατάργηση ή και η τροποποίηση κάποιων εφαρμογών δεν θα έχει καμία επίπτωση στον τρόπο οργάνωσης των αρχείων της βάσης δεδομένων. Στα ΣΔΒΔ έχει επικρατήσει η λεγόμενη αρχιτεκτονική των τριών επιπέδων (βαθμίδων), όπου τα τρία επίπεδα είναι τα εξής :

• *Εσωτερικό επίπεδο (internal level),* έχει να κάνει με την αποθήκευση των αρχείων στον σκληρό δίσκο, δηλ. την πραγματική ή φυσική κατάστασή τους.

• *Εξωτερικό επίπεδο (external level)*, έχει να κάνει με τους χρήστες είτε αυτοί είναι απλοί χειριστές, είτε προγραμματιστές ή και οι διαχειριστές της βάσης δεδομένων.

• *Εννοιολογικό επίπεδο (conceptual level)*, είναι ένα ενδιάμεσο επίπεδο που διασυνδέει τα δύο άλλα επίπεδα και έχει να κάνει με τη λογική σχεδίαση των αρχείων της βάσης δεδομένων.

1.4 Συσχετίσεις

Με τον όρο *συσχέτιση (relationship)* αναφερόμαστε στον τρόπο σύνδεσης (επικοινωνίας) δύο ξεχωριστών οντοτήτων, ώστε να μπορούμε να αντλούμε στοιχεία (πληροφορίες) από τον συνδυασμό τους.

1.4 Τα τρία βασικά μοντέλα

Υπάρχουν τρία βασικά μοντέλα που έχουν επικρατήσει στις βάσεις δεδομένων, το ιεραρχικό, το δικτυωτό και το σχεσιακό, και τα οποία αναπτύχθηκαν με βάση αντίστοιχες δομές.

Το Ιεραρχικό Μοντέλο Βάσεων Δεδομένων

Το ιεραρχικό μοντέλο (hierarchical) έχει μια ιεραρχική δομή που θυμίζει δένδρο. Οι οντότητες μοιάζουν με απολήξεις από κλαδιά δένδρων και τοποθετούνται σε επίπεδα ιεραρχίας. Τα κλαδιά παριστάνουν τις συσχετίσεις ανάμεσα στις οντότητες.
Από μια οντότητα που βρίσκεται σ' ένα ανώτερο επίπεδο εκκινούν πολλά κλαδιά, καθένα από τα οποία καταλήγει σε μια οντότητα που βρίσκεται σ' ένα χαμηλότερο επίπεδο. Αλλά, σε κάθε οντότητα που βρίσκεται σ' ένα χαμηλότερο επίπεδο αντιστοιχεί μία και μόνο μία οντότητα που βρίσκεται σ' ένα ανώτερο επίπεδο. Το μοντέλο αυτό ήταν το πρώτο που εμφανίσθηκε αλλά σήμερα θεωρείται δύσχρηστο και ξεπερασμένο.

Το Δικτυωτό Μοντέλο Βάσεων Δεδομένων

Και στο δικτυωτό (network) μοντέλο, τα στοιχεία τοποθετούνται σ' ένα επίπεδο ιεραρχίας, αλλά κάθε στοιχείο μπορεί να συσχετισθεί με πολλά στοιχεία είτε σ' ένα κατώτερο ή σ' ένα ανώτερο επίπεδο.

Το Σχεσιακό Μοντέλο Βάσεων Δεδομένων

Το σχεσιακό (relational) μοντέλο έχει επικρατήσει σήμερα στην αναπαράσταση των δεδομένων καθώς διαθέτει σημαντικά πλεονεκτήματα ως προς τα άλλα δύο και οι βάσεις δεδομένων που σχεδιάζονται σύμφωνα μ' αυτό αποκαλούνται σχεσιακές (relational databases). Με τις σχεσιακές βάσεις δεδομένων διαθέτουμε έναν σαφή, απλό και εύκολα

κατανοητό τρόπο για να μπορέσουμε να αναπαραστήσουμε και να διαχειριστούμε τα δεδομένα μας. Υστερούν μόνο σε ταχύτητα υπολογισμών και σε χώρο αποθήκευσης, αλλά μόνο όταν έχουμε να κάνουμε πολύ μεγάλες βάσεις δεδομένων. Στο μοντέλο αυτό οι βάσεις δεδομένων περιγράφονται με αυστηρές μαθηματικές έννοιες και ο χρήστης βλέπει τις οντότητες και τις συσχετίσεις με τη μορφή πινάκων (tables) και σχέσεων (relations) αντίστοιχα.
Στις ***Σχεσιακές*** *(Relational)* βάσεις δεδομένων, τα δεδομένα συνδέονται μεταξύ τους με ***σχέσεις*** *(relations)*, οι οποίες προκύπτουν από τα κοινά πεδία που υπάρχουν σε διαφορετικά αρχεία. Τα αρχεία αποκαλούνται *πίνακες (tables)*, οι εγγραφές *γραμμές (rows)* και τα πεδία *στήλες (columns)*. Η ύπαρξη μιας κοινής τιμής στα πεδία δύο αρχείων καθορίζει και μια σχέση μεταξύ των γραμμών διαφορετικών πινάκων. Κάθε πεδίο του πίνακα μπορεί να πάρει ορισμένες μόνο τιμές, οι οποίες μπορεί να καθορίζονται από τον τύπο δεδομένων της ιδιότητας, όπως ονόματα ή αριθμοί για παράδειγμα, ή και από αυτό που εκφράζει, όπως το ότι δεν μπορούμε να έχουμε αρνητικό βάρος ή αρνητικό ΑΦΜ, για παράδειγμα. Το σύνολο των αποδεκτών τιμών μιας οντότητας αποκαλείται *πεδίο ορισμού (domain)*. Οι σχεσιακές βάσεις δεδομένων έχουν το πλεονέκτημα ότι είναι λογικά κατανοητές και πολύ ευέλικτες και δεκτικές σε αλλαγές. Όπως είναι εύκολα κατανοητό, η βασικότερη εργασία που έχουμε να κάνουμε κατά τον σχεδιασμό μιας σχεσιακής βάσης δεδομένων είναι να ορίσουμε τους πίνακες που θα χρησιμοποιήσουμε καθώς και τα πεδία που θα περιέχει ο καθένας απ' αυτούς. Η διαδικασία αυτή αποκαλείται κατασκευή του *σχήματος (schema)* μιας βάσης δεδομένων. Οι κανόνες που πρέπει να ακολουθούμε πιστά κατά τον σχεδιασμό μιας σχεσιακής βάσης δεδομένων είναι οι εξής :
• Η κάθε οντότητα πρέπει να παριστάνεται ως ένας ξεχωριστός πίνακας.
• Η κάθε στήλη του πίνακα αντιστοιχεί σε μια ιδιότητα της οντότητας.
• Η κάθε γραμμή του πίνακα αντιστοιχεί σε μια εμφάνιση της οντότητας.
• Η κάθε γραμμή πρέπει να είναι μοναδική, δηλ. αποκλείεται να υπάρχουν δύο ή και περισσότερες γραμμές που να περιέχουν τα ίδια ακριβώς στοιχεία.
• Η σειρά εμφάνισης των γραμμών δεν έχει καμία σημασία.

1.5 Τα σχεσιακά ΣΔΒΔ (RDBMS)

Τα Σχεσιακά Συστήματα Διαχείρισης Βάσεων Δεδομένων (ΣΣΔΒΔ) ή RBMS (Relational DataBase Management Systems) αναπτύχθηκαν με βάση το σχεσιακό μοντέλο και έχουν επικρατήσει πλήρως στον χώρο. Κατά τον σχεδιασμό και τη δημιουργία μιας σχεσιακής βάσης δεδομένων, οι πίνακες αποτελούν το μοναδικό δομικό και απαραίτητο στοιχείο για μπορέσουν να αναπαρασταθούν οι πληροφορίες που περιέχονται στη βάση δεδομένων.
Για να μπορέσουμε να προσθέσουμε, διαγράψουμε ή τροποποιήσουμε τα στοιχεία που περιέχονται σε μια βάση δεδομένων, χρησιμοποιούμε ειδικές γλώσσες προγραμματισμού. Η γλώσσα που αποτελεί σήμερα ένα διεθνές πρότυπο για την επικοινωνία των χρηστών με τα Σχεσιακά ΣΔΒΔ είναι η *MYSQL*. Τα Σχεσιακά ΣΔΒΔ τα διακρίνουμε στα *μεγάλα*, τα οποία αφορούν κυρίως μεγάλους οργανισμούς και επιχειρήσεις, έχουν τεράστιο όγκο δεδομένων και πολλούς χρήστες ταυτόχρονα, και τέτοια συστήματα είναι τα Oracle, Ingres, Informix, SQL Server κ.ά. και τα *μικρά*, τα οποία αφορούν κυρίως απλούς χρήστες, όπως είναι η Microsoft Access, η Paradox, η FoxPro κ.ά.

1.6 Το Μοντέλο Οντοτήτων – Συσχετίσεων

Το μοντέλο που έχει επικρατήσει σήμερα για να παραστήσει τις έννοιες ή τη δομή μιας βάσης δεδομένων είναι το *Μοντέλο Οντοτήτων –Συσχετίσεων (ΟΣ)*. Οι βασικές (θεμελιώδεις) έννοιες του μοντέλου αυτού είναι οι εξής :

• Οντότητες
• Ιδιότητες ή Χαρακτηριστικά
• Συσχετίσεις

Για να αναπαραστήσουμε ένα Μοντέλο Οντοτήτων – Συσχετίσεων χρησιμοποιούμε ειδικά διαγράμματα, όπου τα ορθογώνια συμβολίζουν τις οντότητες, οι ρόμβοι τις συσχετίσεις και οι ελλείψεις τις ιδιότητες. Με ευθείες γραμμές συνδέουμε τις οντότητες που συσχετίζονται με κάποιο τρόπο μεταξύ τους. Όλα τα παραπάνω αποτελούν τη λογική δομή μιας βάσης δεδομένων, μια εργασία που είναι απαραίτητο να γίνει πριν από την καταχώριση και την επεξεργασία των στοιχείων(πληροφοριών) της βάσης δεδομένων. Το μοντέλο οντοτήτων – συσχετίσεων αποτελεί μια γενική περιγραφή των γενικών στοιχείων που απαρτίζουν μια βάση δεδομένων και απεικονίζει την αντίληψη που έχουμε για τα δεδομένα (εννοιολογικό), χωρίς να υπεισέρχεται σε λεπτομέρειες υλοποίησης.

1.7 Γιατί MySQL

Παρέχει απλοποίηση στην πρόσβαση, βελτίωση στις επιδόσεις, την επεκτασιμότητα και την αξιοπιστία και ικανοποίηση στην εκθετική αύξηση των απαιτήσεων σε αποθήκευση με δραματικά χαμηλότερο κόστος. Επωφελούμαστε από την εξειδίκευση, τις βέλτιστες πρακτικές, την εξυπηρέτηση, την υποστήριξη και τη διαχείριση της MySQL για σύνθετα περιβάλλοντα. Η MySQL μειώνει το συνολικό κόστος κτήσης του λογισμικού βάσης δεδομένων. Επίσης υπάρχει μείωση του κόστους αδειοδότησης της βάσης δεδομένων κατά 90 τοις εκατό, μείωση του χρόνου εκτός λειτουργίας των συστημάτων κατά 60 τοις εκατό, μείωση των δαπανών για υλικό εξοπλισμού κατά 70 τοις εκατό και μείωση του κόστους διαχείρισης, τεχνικού σχεδιασμού και υποστήριξης έως και 50 τοις εκατό.
Παρακάτω αναφέρονται κάποια χαρακτηριστικά και κάποια πλεονεκτήματα της MySQL **:**

Επεκτασιμότητα και ευελιξία

Τα συστήματα της Sun για MySQL μπορούν να επεκταθούν, για να ικανοποιήσουν την εκθετική αύξηση του όγκου δεδομένων που χαρακτηρίζει τα εμπλουτισμένα πολυμέσα, τις Διαδικτυακές κοινότητες και άλλες υπηρεσίες Web. Εκτελείται οτιδήποτε από βαθιά ενσωματωμένες εφαρμογές με καταλαμβανόμενο χώρο μόλις 1MB, ή μαζικές "αποθήκες" δεδομένων - με terabyte πληροφοριών. Επίσης τα μοναδικά στο είδος τους συστήματα της Sun μπορούν να "επεκταθούν μέσα στο διακομιστή, ενώ η τεχνολογία εικονικοποίησης της Sun μπορεί να μειώσει τις απαιτήσεις σε κατανάλωση ρεύματος και χώρο και να επιτύχει εξοικονόμηση κόστους και μεγαλύτερο σεβασμό προς το περιβάλλον.

Υψηλή διαθεσιμότητα.

Εκτέλεση διαμορφώσεων υψηλής ταχύτητας κύριας /εξαρτώμενης αναπαραγωγής βασισμένη σε γραμμές και υβριδική αναπαραγωγή. Οι εξειδικευμένοι διακομιστές συμπλεγμάτων προσφέρουν άμεση μεταγωγή μετά από αστοχία.

Πλεονεκτήματα αποθήκευσης δεδομένων και web.

Υψηλής απόδοσης μηχανισμός ερωτημάτων. Εξαιρετικά γρήγορη δυνατότητα εισαγωγής δεδομένων. Ισχυρή υποστήριξη για εξειδικευμένες λειτουργίες web - συμπεριλαμβανομένων των γρήγορων αναζητήσεων πλήρους κειμένου.

Ισχυρή προστασία δεδομένων.

Πανίσχυροι μηχανισμοί διασφάλισης της προσπέλασης μόνο από εξουσιοδοτημένους

χρήστες. Υποστήριξη SSH και SSL για ασφαλείς και σίγουρες συνδέσεις. Ισχυρή κρυπτογράφηση δεδομένων και λειτουργίες αποκρυπτογράφησης.

Ολοκληρωμένη ανάπτυξη εφαρμογών.

Υποστήριξη για υποθηκευμένες διαδικασίες, κανόνες ενεργοποίησης, λειτουργίες, προβολές, δρομείς, γλώσσες SQL προτύπου ANSI, και πολλά περισσότερα. Βιβλιοθήκες προσθέτων για την ενσωμάτωση της υποστήριξης βάσεων δεδομένων MySQL σχεδόν σε κάθε εφαρμογή.

Ευκολία διαχείρισης.

Υπάρχει χρήση του προγραμματισμού συμβάντων - αυτόματος προγραμματισμός συνήθων επαναλαμβανόμενων εργασιών βασισμένων σε SQL για εκτέλεση σε διακομιστή βάσης δεδομένων. Επίσης ο μέσος χρόνος από τη λήψη λογισμικού μέχρι την ολοκλήρωση της εγκατάστασης είναι λιγότερος από δεκαπέντε λεπτά.

Ελευθερία ανοικτού πηγαίου κώδικα και υποστήριξη 24 ώρες το εικοσιτετράωρο, 7 ημέρες την εβδομάδα.

Μπορεί να προσφέρει εικοσιτετράωρη υποστήριξη και διαθέσιμη επανόρθωση μέσω του δικτύου MySQL. Εταιρική ποιότητα και έτοιμο για εταιρική χρήση από την εγκατάσταση ως την υποστήριξη.

Το χαμηλότερο συνολικό κόστος κτήσης.

Εξοικονόμηση κόστους αδειοδότησης βάσεων δεδομένων και εξόδων για υλικό εξοπλισμού με ταυτόχρονη μείωση του χρόνου εκτός λειτουργίας.

Με γνώμονα την ταχύτητα.

Χάρη στο συνδυασμό πρωτοποριακών τεχνολογιών, της εξειδίκευσης σε συστήματα και των προηγμένων μεθόδων ρύθμισης της Sun, τα συστήματα της Sun για MySQL μας προσφέρουν μια αποτελεσματική και ολοκληρωμένη λύση MySQL. Έτσι, είναι πλέον ευκολότερο για μας να υλοποιήσουμε MySQL και να ικανοποιήσουμε απαιτήσεις για νέες υπηρεσίες Web – με αύξηση των επιδόσεων μέχρι και 3 φορές.

Απλότητα πάνω απ' όλα.

Παρέχει από τα εισαγωγικά συστήματα μέχρι τους πανίσχυρους διακομιστές, με επεξεργαστές Intel, AMD ή SPARC, μεγάλη μνήμη, ταχύτατες λειτουργίες I/O και τις πλέον πρόσφατες τεχνολογίες. Ακόμη όλα τα συστήματα της Sun για MySQL βελτιστοποιούνται και προσαρμόζονται σύμφωνα με τα εκάστοτε περιβάλλοντα MySQL. Επίσης μπορούμε να εκτελέσουμε το λειτουργικό σύστημα της επιλογής μας: Linux, Windows, Solaris 10 ή OpenSolaris.

Μείωση στους κινδύνους με το MySQL Experts.

Οι βέλτιστες πρακτικές, οι αρχιτεκτονικές αναφορές και τα blueprint της Sun, που διατίθενται προς λήψη δωρεάν, σε συνδυασμό με τις δεξιότητες ρύθμισης και υλοποίησης MySQL της Sun και τον οργανισμό επαγγελματικών υπηρεσιών της Sun, μπορούν να μας βοηθήσουν να συνθέσουμε γρήγορα και με ασφάλεια τη δική μας υποδομή Web.

Κεφάλαιο 2

2.1 Η αρχιτεκτονική του Server

Σε αυτό το κεφάλαιο θα γίνει η περιγραφή της αρχιτεκτονικής και της λειτουργίας του Server. Η βασική λειτουργία του Server είναι να παίρνει την εικόνα από την κάμερα και να την στέλνει στο Applet. Άλλες λειτουργίες του Server είναι ο έλεγχος των χρηστών που έχουν πρόσβαση στην εφαρμογή, η καταγραφή των τιμών από τους αισθητήρες του iLon στην βάση δεδομένων, η καταγραφή βίντεο και εικόνας, ο χρονοπρογραμματισμός για την καταγραφή βίντεο και η κίνηση της κάμερας.

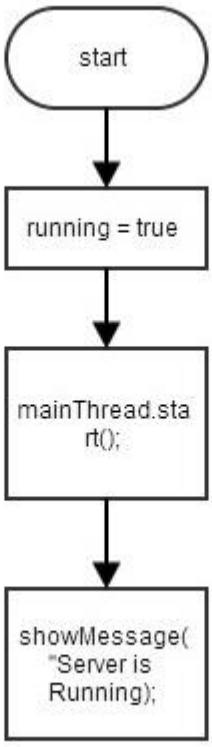
2.2 Η κλάση Main

Στην Main γίνονται όλες οι αρχικοποιήσεις των κλάσεων και ένα μεγάλο μέρος των μεταβλητών που χρειάζονται για να λειτουργήσει το πρόγραμμα. Η βασική μέθοδος της Main αλλά και όλου του προγράμματος είναι η run. Η run εκτός από μία μέθοδος είναι και ένα επιπλέων νήμα επεξεργασίας το οποίο αρχικοποιείτε στον δημιουργό, και τίθεται σε λειτουργία όταν το επιλέγει ο χρήστης μέσω ενός κουμπιού μέσα από την εφαρμογή για να αρχίσει να δέχεται request ο Server. Άλλες δύο μέθοδοι είναι η start και η stop. Η start είναι υπεύθυνη για να θέσει σε λειτουργία την run, ενώ η stop για να τερματίσει την λειτουργία της run και για να αποδεσμεύσει τους πόρους του συστήματος που καταναλώνει το πρόγραμμα και να τερματίσει την λειτουργία του. Παρακάτω περιγράφονται οι λειτουργίες κάθε μεθόδου και παρουσιάζονται τα διαγράμματα ροής της κάθε μεθόδου. Η μέθοδος start είναι η ακόλουθη:

```
public void start(){
        running = true;
        mainThread.start();
        showMessage("Server is running");
       }
```

Την στιγμή που ο χρήστης πατήσει το κουμπί Start Server στο GUI του προγράμματος καλείτε η μέθοδος start. Η start θα αλλάξει την τιμή της μεταβλητής running από false που την είχαμε δηλώσει αρχικά σε true. Η running έχει δηλωθεί σαν global variable πιο πάνω ώστε να είναι ορατή σε όλες τις μεθόδους και με τους εξής τροποποιητές protected volatile boolean. Protected σημαίνει ότι η κλάση Main, το πακέτο με την ονομασία default που περιέχει τις υπόλοιπες κλάσεις του προγράμματος και η υποκλάση κάθε κλάσης εάν υπάρχει μπορούν να δουν αυτή την μεταβλητή. Όταν πολλαπλά νήματα χρησιμοποιούν την ίδια μεταβλητή, κάθε νήμα θα έχει το δικό του αντίγραφο της τοπικής cache για αυτή τη μεταβλητή. Έτσι, όταν πρόκειται για την ενημέρωση της τιμής της μεταβλητής, στην πραγματικότητα ενημερώνεται στην τοπική κρυφή μνήμη και όχι στην κύρια μεταβλητή που βρίσκεται στην μνήμη. Το νήμα που χρησιμοποιεί την ίδια μεταβλητή δεν ξέρει τίποτα για την τιμή που άλλαξε από το άλλο νήμα, στην συγκεκριμένη περίπτωση η κλάση Main. Για να αποφύγουμε αυτό το πρόβλημα, θα πρέπει να δηλώσουμε την μεταβλητή ως volatile, τότε δεν θα αποθηκεύεται στην τοπική μνήμη cache. Κάθε φορά που ένα νήμα θα ενημερώνει την μεταβλητή θα ενημερώνει την μεταβλητή στην κύρια μνήμη και όχι στην τοπική. Έτσι, άλλα νήματα μπορούν να έχουν πρόσβαση στην ενημερωμένη μεταβλητή. Ο τροποποιητής boolean σημαίνει ότι η μεταβλητή είναι λογική δηλαδή μπορεί να πάρει τις τιμές true ή false. Αφού η μεταβλητή running γίνει true καλούμε την μέθοδο start από την κλάση Thread. Στην αρχή της παραγράφου ανέφερα ότι έχουμε αρχικοποιήσει το νέο νήμα επεξεργασίας στον δημιουργό,

για να γίνει αυτό θα πρέπει πρώτα να ορίσουμε μία μεταβλητή τύπου Thread σαν global variable και την οποία την ονομάσαμε mainThread. Στην συνέχεια φτιάχνουμε το αντικείμενο της κλάσης Thread με τον εξής τρόπο mainThread = new Thread(this); Η κλάση Thread είναι μία έτοιμη κλάση που μας παρέχει η Java και περιέχει κάποιες μεθόδους όπως η start (η μέθοδος start της κλάσης Thread δεν πρέπει να συγχέεται με την μέθοδο start που έχω φτιάξει εγώ και την έχω ονομάσει έτσι για να καταλαβαίνει την λειτουργία της κάποιος που διαβάζει τον κώδικα). Αφού καλέσουμε την start με την εντολή mainThread.start(); το νέο νήμα επεξεργασίας θα τρέχει μέσα στην μέθοδο run. Με την τελευταία εντολή καλούμε την μέθοδο showMessage και της περνάμε μία συμβολοσειρά η οποία είναι Server is running για να την εμφανίσει στο GUI της εφαρμογής με σκοπό να ενημερώσει ότι ο Server είναι σε λειτουργία και μπορεί να δεχτεί request. Το διάγραμμα ροής τη μεθόδου φαίνεται παρακάτω:



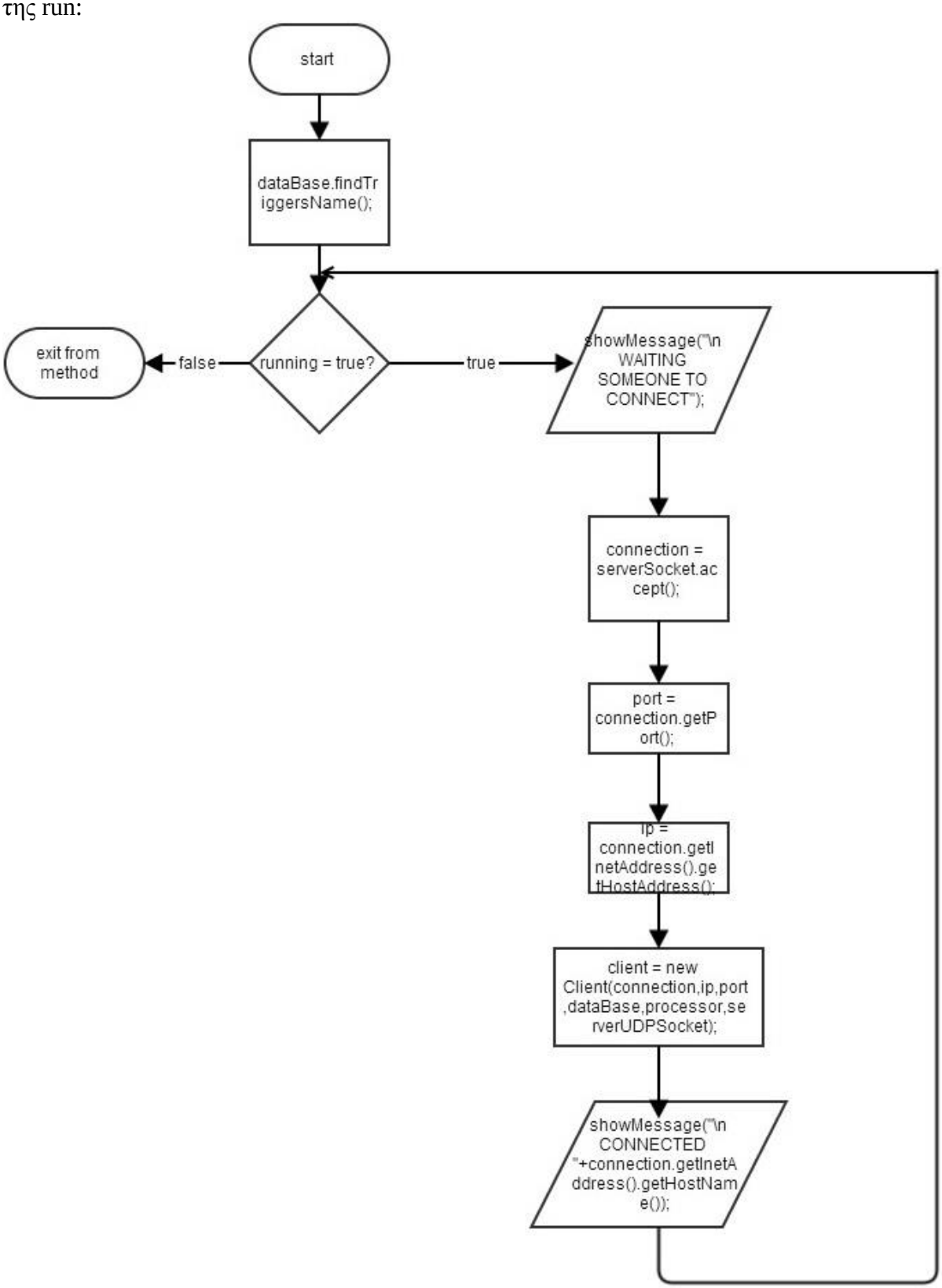
Η μέθοδος run είναι η ακόλουθη:

```
public void run() {
		dataBase.findTriggersName();
		while(running){
			showMessage("\n WAITING SOMEONE TO CONNECT");
			try {
				connection = serverSocket.accept();
			} catch (IOException e) {
				e.printStackTrace();
			}
			port = connection.getPort();
			ip = connection.getInetAddress().getHostAddress();
```

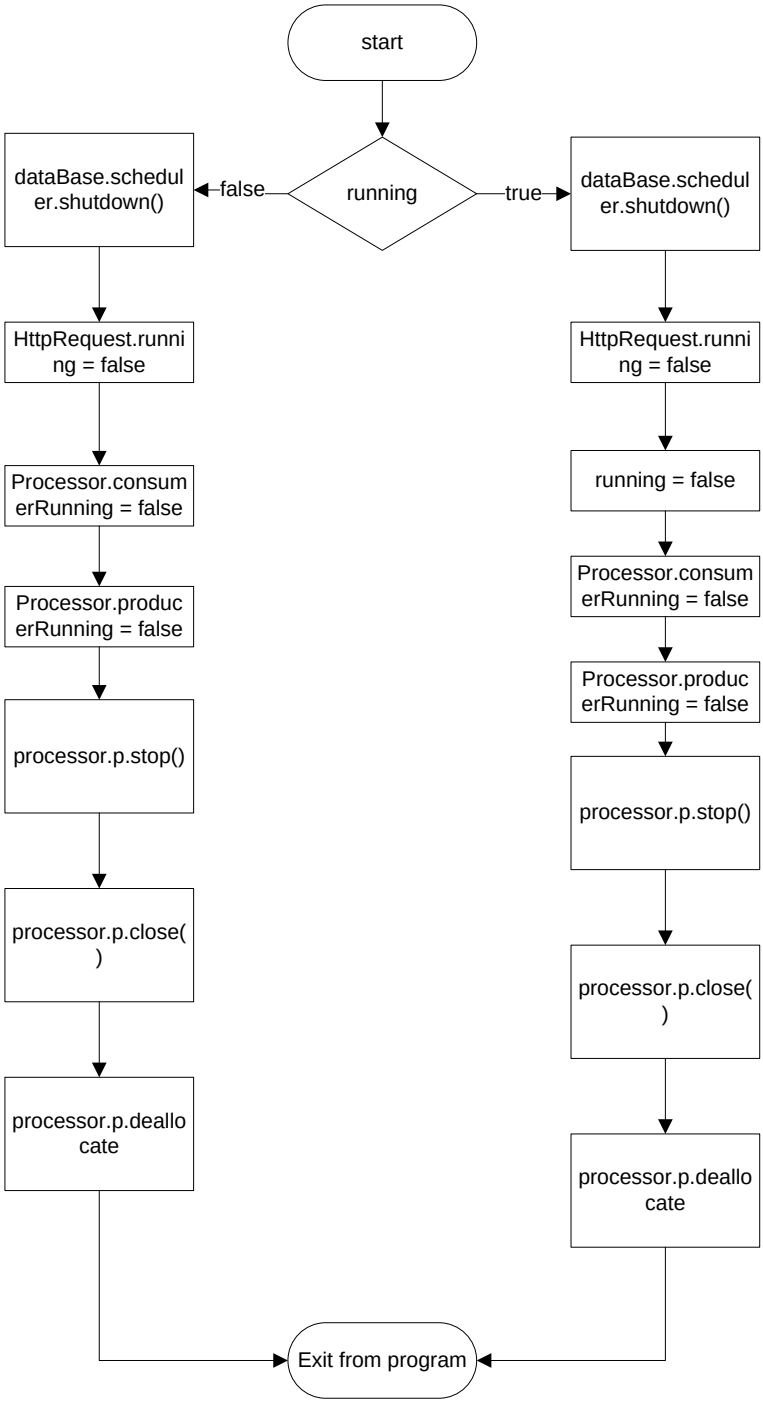
```
                    client = new
Client(connection,ip,port,dataBase,processor,serverUDPSocket);
                    showMessage("\n CONNECTED
"+connection.getInetAddress().getHostName());
                }
}
```

Μόλις αρχίσει να τρέχει η μέθοδος run το πρώτο που κάνει είναι να καλέσει την μέθοδο findTriggersName από την κλάση DataBase της οποίας το αντικείμενο το έχουμε ονομάσει dataBase. Η λειτουργία και ο λόγος που καλείται η συγκεκριμένη μέθοδος θα αναλυθεί στην παράγραφο 2.6 που αφορά τον χρονοπρογραμματισμό εγγραφής βίντεο. Στην συνέχεια μπαίνει στον βρόχο while ο οποίος θα εκτελείτε όσο η μεταβλητή running θα είναι true, αυτός είναι και ο λόγος για τον οποίο χρησιμοποιούμε ένα επιπλέων νήμα επεξεργασίας ώστε το πρόγραμμα να μην «κολλήσει» στον βρόχο και δεν μπορεί να εκτελέσει άλλες διαδικασίες. Αρχικά με την μέθοδο showMessage μας εμφανίζει το μήνυμα στο GUI του προγράμματος την φράση WAITING SOMEONE TO CONNECT το οποίο μας ενημερώνει ότι περιμένει κάποιων χρήστη να συνδεθεί με την εφαρμογή. Έπειτα μπαίνει σε ένα try μπλοκ το οποίο σημάνει ότι το πρόγραμμα θα προσπαθήσει να εκτελέσει τον κώδικα που βρίσκεται μέσα σε αυτό. Εάν για κάποιο λόγο υπάρξει πρόβλημα κατά την εκτέλεση του κώδικα τότε θα «πετάξει» μία εξαίρεση και θα μπει μέσα στο catch μπλοκ ώστε να «πιάσει» την εξαίρεση και να μην σταματήσει η λειτουργία του προγράμματος. Το αντικείμενο της εσωτερικής κλάσης ServerSocket ονομάζεται serverSocket με το οποίο καλούμε την μέθοδο accept η οποία περιμένει μέχρι να δεχτεί κάποιο request και έπειτα το αποθηκεύει στην μεταβλητή connection η ποία είναι τύπου Socket. Το αντικείμενο της κλάσης ServerSocket το έχουμε αρχικοποίηση στον δημιουργό με τον εξής τρόπο serverSocket = new ServerSocket(8080,50);. Η παράμετρος 8080 σημαίνει ότι τα requests θα τα δέχεται στο port 8080 και η παράμετρος 50 σημαίνει ότι μπορούν να συνδεθούν μόνο 50 χρήστες ταυτόχρονα. Στην συνέχεια αποθηκεύει σε ξεχωριστές μεταβλητές το port και την ip του εκάστοτε χρήστη που έχει συνδεθεί και φτιάχνει ένα καινούργιο στιγμιότυπο της κλάσης Client περνώντας αυτές τις μεταβλητές και κάποιες άλλες στην κλάση η οποίες θα εξηγηθούν στη παράγραφο που αναφέρεται σε αυτήν την κλάση. Τέλος η μέθοδος showMessage μας εμφανίζει την ip του χρήστη που έχει συνδεθεί την συγκεκριμένη στιγμή. Το διάγραμμα ροής

της run:


start
dataBase.findTriggersName();
running = true?
false
exit from method
true
showMessage("\n WAITING SOMEONE TO CONNECT");
connection = serverSocket.accept();
port = connection.getPort();
ip = connection.getInetAddress().getHostAddress();
client = new Client(connection,ip,port,dataBase,processor,serverUDPSocket);
showMessage("\n CONNECTED "+connection.getInetAddress().getHostName());

Εάν ο χρήστης θέλει να τερματίσει την λειτουργία του server θα πρέπει να πατήσει το κουμπί Close Server στο GUI της εφαρμογής η απλά να κλείσει το παράθυρο, και στις δύο περιπτώσεις καλείται η μέθοδος stop. Εάν προηγουμένως δεν έχουμε πατήσει το κουμπί Start Server π.χ. για να μην μπορεί κάποιος χρήστης να συνδεθεί αλλά για να καταγράφει τις τιμές από τους αισθητήρες και να τους αποθηκεύει στην βάση δεδομένων, τότε είναι έγκυρη η πρώτη συνθήκη. Στην αρχή του μπλοκ καλούμε την μέθοδο shutdown από την εσωτερική κλάση Scheduler της οποίας το αντικείμενο το έχουμε ονομάσει scheduler και το οποίο ανήκει στην κλάση DataBase. Με αυτόν τον τρόπο τερματίζουμε την λειτουργία που είναι υπεύθυνη για τον χρονοπρογραμματισμό εγγραφής βίντεο. Στην συνέχεια αλλάζουμε τις τιμές στις λογικές μεταβλητές running που ανήκει στην κλάση HttpRequest, στις consumerRunning και producerRunning που ανήκουν στην κλάση Processor σε false. Αυτό το κάνουμε για να τερματίσουμε τα νήματα επεξεργασίας στα οποία ανήκουν και τα οποία θα εξηγηθούν σε επόμενα κεφάλαια πιο αναλυτικά. Στην συγκεκριμένη περίπτωση καλέσαμε της μεταβλητές χωρίς να έχουμε φτιάξει τα αντικείμενα των κλάσεων στις οποίες ανήκουν αλλά γράφοντας μόνο τα ονόματα των κλάσεων. Αυτό γίνεται λόγο ότι η μεταβλητές αυτές είναι στατικές δηλαδή έχουν τον τροποποιητή static. Παρακάτω καλούμε τις μεθόδους stop, close και deallocate από την εσωτερική κλάση Player με αντικείμενο p που ανήκει στην κλάση Processor με αντικείμενο processor. Η πρώτη μέθοδος διακόπτη την καταγραφή της εικόνας από τον player, η δεύτερη διακόπτει οποιαδήποτε δραστηριότητα που καταναλώνει πόρους του συστήματος. και η τρίτη απελευθερώνει όλους τους πόρους του συστήματος. Τέλος τερματίζει η λειτουργία του προγράμματος. Εάν η μεταβλητή running είναι true τότε έγκυρη είναι η δεύτερη συνθήκη και το μπλοκ κώδικα είναι τι ίδιο με το παραπάνω με την μόνη διαφορά ότι αλλάζουμε την τιμή της μεταβλητής running σε false για να τερματίσουμε το νήμα επεξεργασίας. Το διάγραμμα ροής είναι το παρακάτω:



2.3 Η κλάση Processor

Η κλάση Processor όπως δηλώνει και το όνομα της είναι υπεύθυνη για να τραβάει τα καρέ από την κάμερα και να τα μετατρέπει σε εικόνες για να μπορούμε να τις επεξεργαστούμε και να τις χρησιμοποιήσουμε όπως θέλουμε. Η Processor αποτελείται από δύο νήματα επεξεργασίας το ένα ονομάζεται t1 και τρέχει μέσα στην μέθοδο consumerThread() και το δεύτερο ονομάζεται t2 και τρέχει μέσα στην μέθοδο producerThread(). Χρησιμοποιούμε δύο νήματα επεξεργασίας ώστε να τρέχουν ταυτόχρονα και να είναι ανεξάρτητα μεταξύ τους. Σε αυτή την κλάση βλέπουμε ότι οι αρχικοποιήσεις αλλά και η λειτουργία των νημάτων επεξεργασίας είναι διαφορετικές σε σχέση με την κλάση Main λόγω ότι έχουμε δύο επιπλέον νήματα επεξεργασίας. Μέσα στις μεθόδους producerThread() και consumerThread()

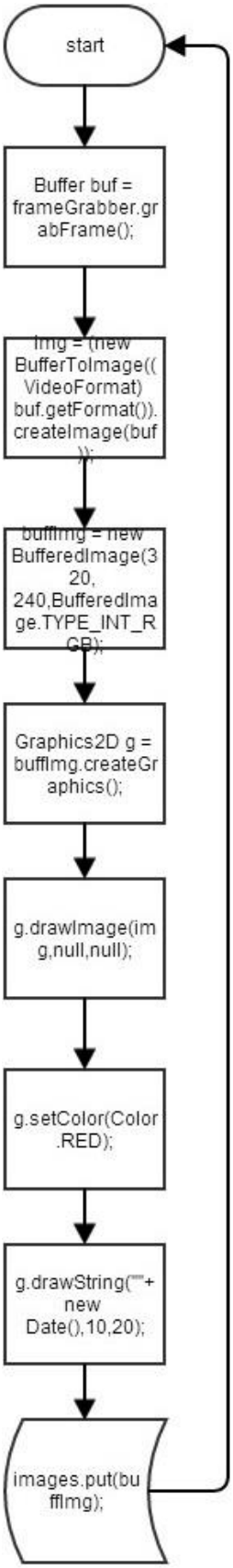
αρχικοποιούμε τα νήματα επεξεργασίας και τα θέτουμε σε λειτουργία. Για αρχή θα πρέπει να αρχικοποιήσουμε τον Player και να ορίσουμε την πηγή από την οποία θα παίρνει την εικόνα. Αυτό θα γίνει μέσα στον δημιουργό με τον εξής τρόπο:

```
try {
                    p = Manager.createRealizedPlayer(new MediaLocator("vfw://1"));
            } catch (NoPlayerException e) {
                    e.printStackTrace();
            } catch (CannotRealizeException e) {
                    e.printStackTrace();
            } catch (IOException e) {
                    e.printStackTrace();
            }
```

Αφού έχουμε δηλώσει την μεταβλητή p σαν global και είναι τύπου Player μέσα στον δημιουργό χρησιμοποιούμε την κλάση Manager η οποία είναι το σημείο πρόσβασης για την απόκτηση των απαραιτήτων πόρων του συστήματος όπως Player, και DataSources και θέτουμε τον Player σε κατάσταση realized με την μέθοδο createRealizedPlayer. Αυτό σημαίνει ότι ο Player ξέρει τους πόρους που μπορεί να χρησιμοποιήσει. Ο αναπαραγωγός πολυμέσων JMF συνήθως χρησιμοποίει DataSources για την διαχείριση της μεταφοράς των πολυμέσων και στην συγκεκριμένη περίπτωση την εικόνα από την κάμερα. Μία DataSource ενσωματώνει την θέση των πολυμέσων, το πρωτόκολλο και το λογισμικό που χρησιμοποιείτε για την μεταφορά των πολυμέσων. Μια DataSource προσδιορίζεται είτε από τον JMF MediaLocator ή μια διεύθυνση URL (Universal Resource Locator). Στην μέθοδο createRealizedPlayer έχουμε περάσει σαν παράμετρο τον MediaLocator και στον MediaLocator την θέση στην οποία βρίσκεται το πολυμέσο και στην συγκεκριμένη περίπτωση η κάμερα αλλά και το πρωτόκολλο το οποίο είναι vfw. Στην συνέχεια με την μέθοδο start λέμε στον Player να εκκινήσει το πρόγραμμα αναπαραγωγής το συντομότερο δυνατό και να παρουσιάσει τα δεδομένα . Για να γίνουν όλα τα παραπάνω θα πρέπει να φτιάξουμε το στιγμιότυπο της κλάσης Processor και στην συγκεκριμένη περίπτωση αυτό γίνεται μέσα από τον δημιουργό της κλάσης Main. Στην συνέχεια στον δημιουργό της Main καλούμε τις μεθόδους Processor.producerThread και Processor.consumerThread για να αρχίσει η μετατροπή των καρέ και οι οποίες είναι στατικές. Μέσα στο νήμα producerThread τρέχει η μέθοδος producer η οποία είναι η παρακάτω:

```
public static void producer(){
                Buffer buf = frameGrabber.grabFrame();
                // Convert frame to an buffered image so it can be processed and saved
           img = (new BufferToImage((VideoFormat) buf.getFormat()).createImage(buf));
                buffImg = new BufferedImage(320, 240,BufferedImage.TYPE_INT_RGB);
                Graphics2D g = buffImg.createGraphics();
                g.drawImage(img,null,null);
                g.setColor(Color.RED);
                g.drawString(""+new Date(),10,20);
                try {
                        images.put(buffImg);
                } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }
```

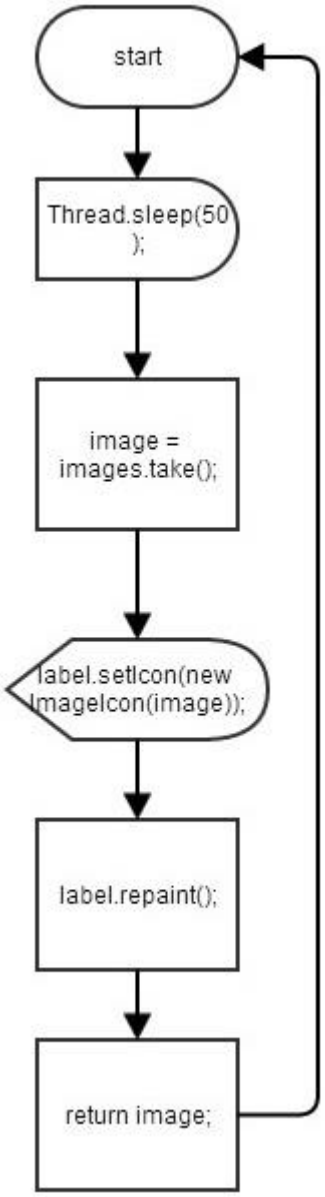
Στην αρχή της μεθόδου τραβάμε ένα καρέ από τον Player και το αποθηκεύουμε σε ένα buffer. Το αντικείμενο buffer μεταφέρει τα δεδομένα από το ένα στάδιο επεξεργασίας σε ένα άλλο και περιέχει πληροφορίες όπως τον χρόνο, το μήκος και την μορφή των δεδομένων που φέρει, καθώς και οποιαδήποτε πληροφορία header που ενδέχεται να απαιτηθεί για την επεξεργασία των δεδομένων. Στην συνέχεια μετατρέπουμε το καρέ σε image δηλαδή σε μία εικόνα ώστε να μπορούμε να την επεξεργαστούμε. Έπειτα δημιουργούμε ένα buffered image δηλαδή μία νέα εικόνα προσωρινά αποθηκευμένη στην μνήμη με διαστάσεις 320,240 και το οποίο είναι τύπου TYPE_INT_RGB. Ο τύπος TYPE_INT_RGB σημαίνει ότι το χρώμα στην εικόνα αντιπροσωπεύετε σαν 4 μπιτ ακέραιο. Ζωγραφίζουμε το καρέ που έχουμε φτιάξει επάνω στο buffered image με την μέθοδο drawImage(img,null,null), θέτουμε σαν χρώμα το κόκκινο με την μέθοδο setColor και εμφανίζουμε την ημερομηνία στην θέση 10,20 με την μέθοδο drawString. Τέλος το buffered image το αποθηκεύουμε σε ένα πίνακα με το όνομα images, ο πίνακας αυτός είναι τύπου FIFO με χωρητικότητα 10 εικόνων. Με την μέθοδο put εισάγουμε την εικόνα στον πίνακα μέχρι ο αριθμός των εικόνων φτάσει στον αριθμό 10. Εάν συμπληρωθεί ο μέγιστος αριθμός εικόνων τότε το νήμα επεξεργασίας περιμένει έως αυτός ο αριθμός μειωθεί. Παρακάτω φαίνεται το διάγραμμα ροής:


start
Buffer buf = frameGrabber.grabFrame();
Img = (new BufferToImage((VideoFormat) buf.getFormat()).createImage(buf));
buffImg = new BufferedImage(320, 240,BufferedImage.TYPE_INT_RGB);
Graphics2D g = buffImg.createGraphics();
g.drawImage(img,null,null);
g.setColor(Color.RED);
g.drawString(""+new Date(),10,20);
images.put(buffImg);

Μέσα στο νήμα consumerThread τρέχει η μέθοδος consumer η οποία είναι η παρακάτω:

```
public static BufferedImage consumer(){
        try {
                Thread.sleep(50);
        } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
                                try {
                                   image = images.take();
                                        label.setIcon(new ImageIcon(image));
                                } catch (InterruptedException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                        label.repaint();
                        return image;
        }
```

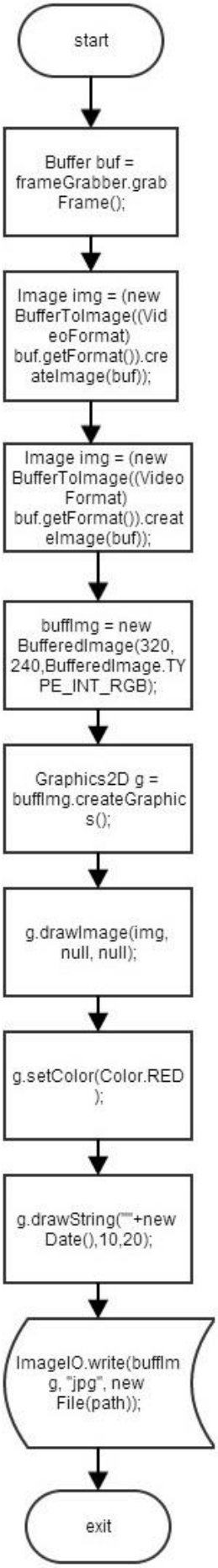
Όπως δηλώνει και το όνομα της η λειτουργία της είναι να παίρνει τα καρέ που είναι αποθηκευμένα στον πίνακα και να τα εμφανίζει στο GUI της εφαρμογής ή να τα δίνει στην κλάση Client για να τα στείλει μέσω του διαδικτύου στο applet. Η διαφορά στην δήλωση αυτής της μεθόδου είναι ότι αντί για void την έχουμε δηλώσει σαν BufferedImage. Εάν δηλώσουμε κάποια μέθοδο σαν BufferedImage τότε αυτή η μέθοδος επιστρέφει ένα buffered image στο σημείο που την καλέσαμε, αυτό θα εξηγηθεί καλύτερα στην κλάση Client. Το πρώτο που γίνεται όταν σε αυτήν την μέθοδο είναι μία καθυστέρηση των 50 ms. Ο λόγος αυτής της καθυστέρησης είναι για να επιτύχουμε τα 20 καρέ το δευτερόλεπτο και να μην χρησιμοποιεί η μέθοδος όλη την υπολογιστική ισχύ του συστήματος. Έπειτα με την image = images.take() παίρνουμε το πρώτο στοιχείο του πίνακα και το αποθηκεύουμε σε μία μεταβλητή με το όνομα image. Ακολούθως εμφανίζουμε την εικόνα στην εφαρμογή με την εντολή label.setIcon(new ImageIcon(image)) και μετά σβήνουμε την εικόνα με την label.repaint() για να εμφανιστή η επόμενη. Τέλος με την εντολή return image λέμε στο πρόγραμμα να επιστρέψει αυτήν την μεταβλητή. Παρακάτω φαίνεται το διάγραμμα ροής της consumer:



Τέλος η μέθοδος grabframe καλείτε από την Client όταν ο χρήστης που είναι συνδεδεμένος θέλει να τραβήξει μία φωτογραφία. Αυτή η μέθοδος είναι ίδιας λογικής με την producer στις πρώτες 7 εντολές, όμως αντί να αποθηκεύσουμε την εικόνα στον πίνακα την αποθηκεύουμε στον σκληρό δίσκο του συστήματος μας με την μέθοδο που έχουμε καλέσει από την εσωτερική κλάση της Java που είναι η ImageIO. Η πρώτη παράμετρος που παίρνει η μέθοδος write είναι η εικόνα που θέλουμε να αποθηκεύσουμε η δεύτερη παράμετρος είναι η μορφή της εικόνας και συγκεκριμένη περίπτωση είναι JPG και τέλος τον φάκελο στον οποίο θέλουμε να αποθηκεύσουμε την εικόνα. Παρακάτω φαίνονται η μέθοδος grabframe και το διάγραμμα ροής της μεθόδου:

```
public void grabFrame(String path){
                Buffer buf = frameGrabber.grabFrame();
                // Convert frame to an buffered image so it can be processed and saved
                Image img = (new BufferToImage((VideoFormat)
buf.getFormat()).createImage(buf));
                buffImg = new BufferedImage(320, 240,BufferedImage.TYPE_INT_RGB);
                Graphics2D g = buffImg.createGraphics();
```

```
                g.drawImage(img, null, null);
                g.setColor(Color.RED);
                g.drawString(""+new Date(),10,20);
                try {
                        ImageIO.write(buffImg, "jpg", new File(path));
                } catch (IOException e) {
                        e.printStackTrace();
                }
}
```


start
Buffer buf = frameGrabber.grabFrame();
Image img = (new BufferToImage((VideoFormat) buf.getFormat()).createImage(buf));
Image img = (new BufferToImage((VideoFormat) buf.getFormat()).createImage(buf));
buffImg = new BufferedImage(320, 240,BufferedImage.TYPE_INT_RGB);
Graphics2D g = buffImg.createGraphics();
g.drawImage(img, null, null);
g.setColor(Color.RED);
g.drawString(""+new Date(),10,20);
ImageIO.write(buffImg, "jpg", new File(path));
exit

2.4 Η κλάση Client

Όπως είχαμε αναφέρει και στην παράγραφο που αφορούσε την κλάση Main όταν ένας χρήστης συνδέεται στον Server δημιουργείτε ένα νέο στιγμιότυπο της κλάσης Client το οποίο περιέχει το socket του χρήστη. Η κλάση έχει και αυτή ένα επιπλέον νήμα επεξεργασίας το οποίο είναι υπεύθυνο για την επικοινωνία του Server και του applet. Μέσα σε αυτό το νήμα τρέχουν δύο μέθοδοι οι οποίες είναι η setupStreams και η whileConnected. Η μέθοδος setupStreams φαίνεται παρακάτω:

```
public void setupStreams(){
		try {
			input = new ObjectInputStream(connection.getInputStream());
			output = new ObjectOutputStream(connection.getOutputStream());
		} catch (IOException e) {
			e.printStackTrace();
		}
	}
```

Σε αυτήν την μέθοδο αρχικοποιούμε την σύνδεση μεταξύ του Server και του applet ώστε να μπορούμε να επικοινωνούμε με τον Server. Η επικοινωνία θα γίνεται μέσω του πρωτοκόλλου TCP για να εξασφαλίσουμε ότι τα μηνύματα θα φτάνουν στον προορισμό τους χωρίς να χαθούν. Αφού επιδεχθεί η σύνδεση μεταξύ των δύο εφαρμογών θα αρχίσει η εκτέλεση της επόμενης μεθόδου η οποία είναι η whileConnected. Όπως δηλώνω και το όνομα της αυτή η μέθοδος θα εκτελείτε όσο οι δύο εφαρμογές είναι συνδεδεμένες. Η whileConnected φαίνεται παρακάτω:

```
public void whileConnected(){
		do{
			try {
				message = (String) input.readObject();
			} catch (ClassNotFoundException e){
				e.printStackTrace();
			} catch (IOException e) {
				e.printStackTrace();
			}
			if(message.equals("PLAY")){
				streaming = true;
				streaming();
			}
			if(message.equals("PAUSE")){
				streaming = false;
			}
			if(message.equals("CAPTURE_VIDEO_START")){
				videoId = dataBase.findVideoId(id)+1;
				videoWriter =
ToolFactory.makeWriter("c://"+Username+videoId+".mp4");
				  videoWriter.addVideoStream(0, 0,
ICodec.ID.CODEC_ID_MPEG4,320, 240);
				record = true;
				startTime = System.currentTimeMillis();
```

```
                recordingVideo();
                threadRec.start();
                dataBase.storeVideoDBStart(id, Username+videoId);
                message = "nothing";
        }
        if(message.equals("CAPTURE_VIDEO_STOP")){
                record = false;
                stopRecording = true;
                stopTime = System.currentTimeMillis();
                duration = stopTime - startTime;
                dataBase.storeVideoDBStop(duration/1000, id);
message = "nothing";
        }
        if(message.equals("initStreaming")){
                streaming = true;
                streaming();
                message = "nothing";
        }
        if(message.equals("GRABFRAME")){
                int imageId = dataBase.findImageId(id)+1;
                processor.grabFrame("c://"+Username+imageId+".jpg");
                dataBase.storeImageDB(id,Username+imageId);
                message = "nothing";
        }
        if(message.contains("LOGIN")){
                beginIndex = 5;
                Character Char = new Character(message.charAt(beginIndex));
           do{
                Username = Username+Char;
                beginIndex++;
                Char = new Character(message.charAt(beginIndex));
           }while(!Char.equals(','));
           beginIndex = beginIndex+1;
           Char = new Character(message.charAt(beginIndex));
           do{
                Password = Password+Char;
                beginIndex++;
                Char = new Character(message.charAt(beginIndex));
           }while(!Char.equals('.'));
           dataBase.findUser(Username,Password);
           if(dataBase.findUser(Username, Password)){
                id = dataBase.id;
                        try {
                                serverUDPSocket = new DatagramSocket();
                        } catch (SocketException e1) {
                                e1.printStackTrace();
                        }
                        if(dataBase.findIfAdmin(Username, Password)){
                try {
```

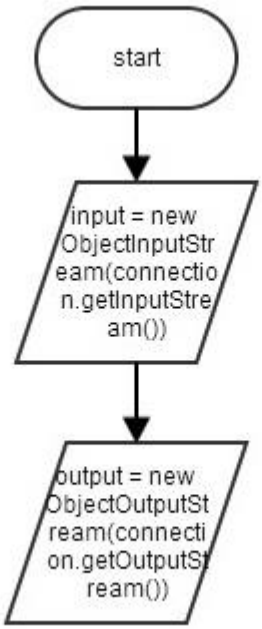
```
output.writeObject("CONFIRMED"+serverUDPSocket.getLocalPort());
                                        output.flush();
                                    } catch (IOException e) {
                                        e.printStackTrace();
                                    }
                                System.out.println("admin");
                                    }else{
                                        try {

output.writeObject("CONFIRMED"+serverUDPSocket.getLocalPort()+"noAdmin");
                                            output.flush();
                                        } catch (IOException e) {
                                            e.printStackTrace();
                                        }
                                        System.out.println("noAdmin");
                                    }

dataBase.userLogIn(id,connection.getInetAddress().getHostAddress());
                            }else{
                                try {
                                        output.writeObject("DENIED");
                                        output.flush();
                                        Username = "";
                                        Password = "";
                                    } catch (IOException e) {
                                        e.printStackTrace();
                                    }
                            }
                            message = "nothing";
                        }
                        if(message.contains("schedule")){
                                videoId = dataBase.findVideoId(id)+1;
                                String startDateTime = "";
                                String endDateTime = "";
                                String comma = ",";
                                message = message.replace("schedule", "");
                                String[] temp;
                                temp = message.split(comma);
                                startDateTime = temp[0];
                                endDateTime = temp[1];

dataBase.ScheduleRec(startDateTime,endDateTime,Username,Username+videoId,
id);
                                message = "nothing";
                        }
                    }while(!message.equals("END"));
                    if(message.equals("END") && streaming){
                            stopStreaming();
                            try {
```

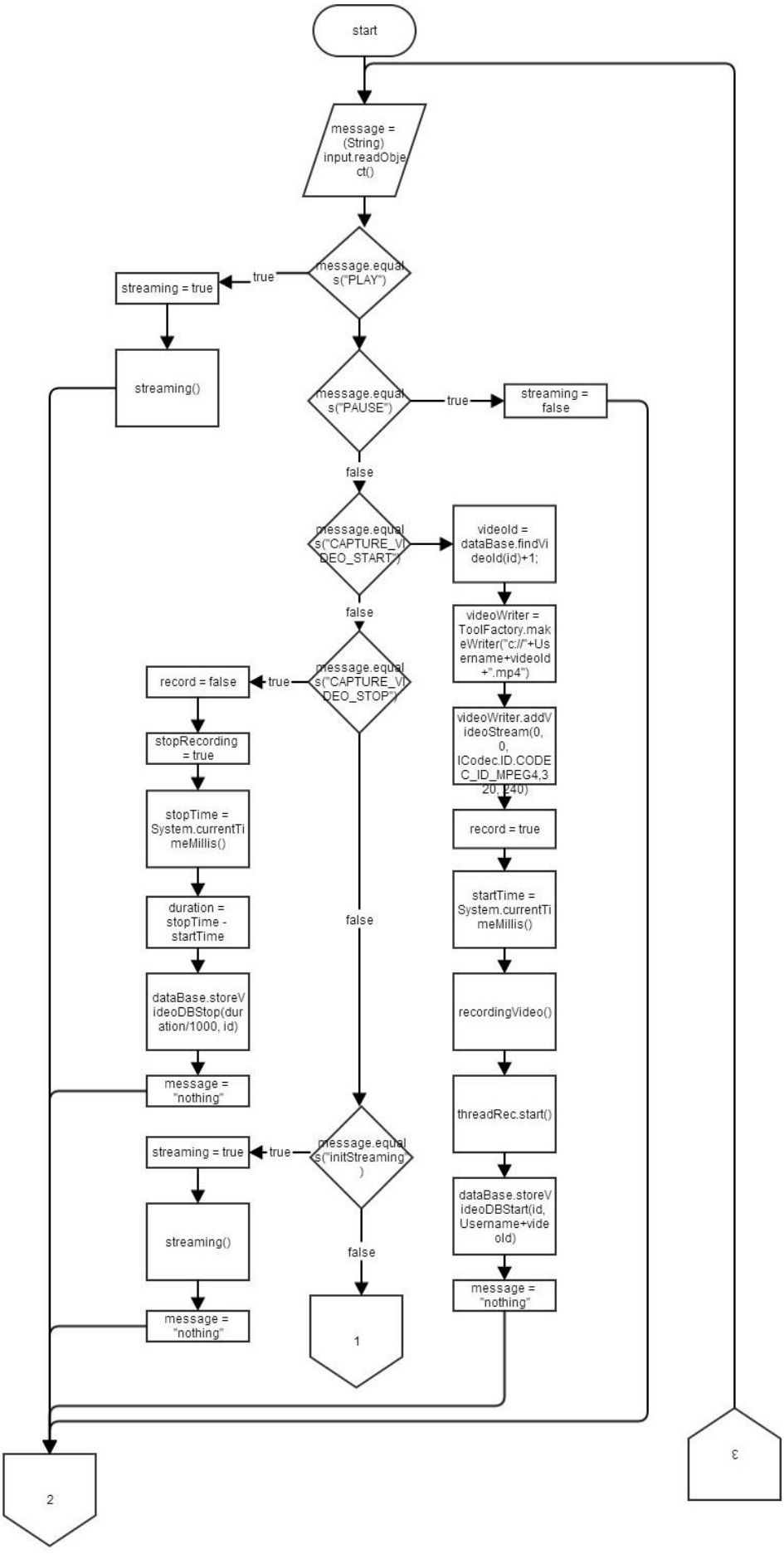
```
                        streaming = false;
                        running  = false;
                        input.close();
                        output.close();
                        connection.close();
                } catch (IOException e) {
                        e.printStackTrace();
                }
                dataBase.userLogOut(id);
        }else if(message.equals("END") && !streaming){
                try {
                        input.close();
                        output.close();
                        running = false;
                        connection.close();
                        serverUDPSocket.close();
                        dataBase.userLogOut(id);
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }
}
```

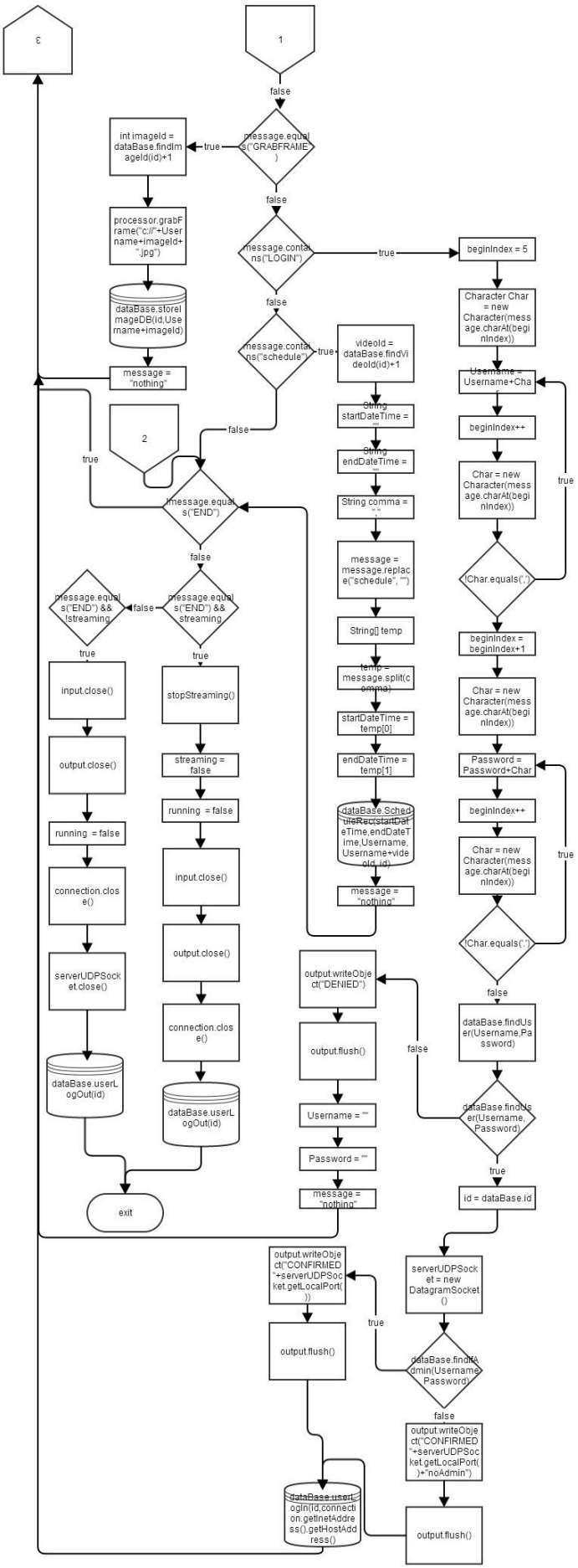
Για να επικοινωνήσει το applet με τον Server στέλνει κάποια προκαθορισμένα μηνύματα. Το πρώτο πράγμα που κάνει η μέθοδος είναι να περιμένει να διαβάσει ένα από αυτά τα μηνύματα με την εντολή message = (String) input.readObject(). Με την input.readObject() διαβάζει το μήνυμα την μετατρέπει σε συμβολοσειρά με την (String) και την αποθηκεύει σε μία μεταβλητή με το όνομα message. Η μεταβλητή message συγκρίνεται με κάποιες συμβολοσειρές και εκτελεί την ανάλογη λειτουργία. Οι συμβολοσειρές αυτές είναι: PLAY, PAUSE, CAPTURE_VIDEO_START, CAPTURE_VIDEO_STOP, initStreaming, GRABFRAME, LOGIN, schedule, END. Με την συμβολοσειρά PLAY αλλάζουμε την τιμή της λογικής μεταβλητής streaming true και καλούμε την μέθοδο streaming (οι λειτουργίες των μεθόδων θα εξηγηθούν παρακάτω) ώστε να στέλνει ο Server εικόνα στο applet. Με την PAUSE το μόνο που κάνουμε είναι να αλλάζουμε την τιμή της μεταβλητής streaming σε false ώστε να σταματήσει να στέλνει εικόνα στο applet. Με την CAPTURE_VIDEO_START ψάχνουμε το πλήθος των βίντεο που έχει τραβήξει ο χρήστης με την μέθοδο dataBase.findVideoId(id)+1 και το αυξάνουμε κατά ένα ώστε να βρούμε τον αριθμό του βίντεο που τραβάει εκείνη την στιγμή. Έπειτα ορίζουμε την θέση που θέλουμε να αποθηκεύσουμε το βίντεο στον σκληρό δίσκο με την μέθοδο videoWriter = ToolFactory.makeWriter("c://"+Username+videoId+".mp4") η οποία είναι εσωτερική της java. Η αποθήκευση των βίντεο γίνεται στον c με όνομα το username του χρηστή μαζί με τον αριθμό και τον τύπο του βίντεο και το αποθηκεύουμε σε μία μεταβλητή η οποία είναι τύπου IMediaWriter δηλαδή υπεύθυνη για την εγγραφή βίντεο. Στην συνέχεια με την μέθοδο videoWriter.addVideoStream(0, 0, ICodec.ID.CODEC_ID_MPEG4,320, 240) αρχικοποιούμε μία ροή βίντεο με id 0 στην θέση 0, κωδικοποίηση MPEG4 και μέγεθος 320, 240. Αλλάζουμε την τιμή της μεταβλητής record σε true και καταγράφουμε την τρέχουσα ώρα σε millsecond για να υπολογίσουμε την χρονική διάρκεια του βίντεο. Μετά καλούμε την μέθοδο recordingVideo για να αρχικοποιήσουμε το νήμα επεξεργασίας που υπάρχει σε αυτή και το θέτουμε σε λειτουργία με την threadRec.start() για να αρχίσει η καταγραφή του βίντεο. Τέλος καλούμε την μέθοδο storeVideoDBStart(id, Username+videoId) περνώντας σαν παραμέτρους

το id του χρήστη και το όνομα του βίντεο. Με την CAPTURE_VIDEO_STOP σταματάμε την καταγραφή του βίντεο αλλάζοντας την τιμή της μεταβλητής record σε false ώστε να τερματίσει το νήμα επεξεργασίας που είναι υπεύθυνο για την καταγραφή. Για να αποδεσμεύσουμε το βίντεο από το σύστημα αλλάζουμε την μεταβλητή stopRecording σε true. Καταγράφουμε την τρέχουσα ώρα και αφαιρούμε την ώρα που άρχισε το βίντεο από την ώρα που τελείωσε για να βρούμε την χρονική διάρκεια του και μέσω της μεθόδου storeVideoDBStop(duration/1000, id) που ανήκει στην κλάση DataBase αποθηκεύουμε την διάρκεια του σε δευτερόλεπτα. Με την initStreaming αρχικοποιούμε την αποστολή βίντεο και επίσης ο Server αρχίζει να στέλνει βίντεο στο applet. Με την GRABFRAME ο χρήστης μπορεί να τραβήξει φωτογραφία και να το αποθήκευση στον Server. Στέλνοντας τη συμβολοσειρά GRABFRAME ο Server με την μέθοδο dataBase.findImageId(id)+1 ψάχνει το πλήθος των φωτογραφιών που έχει τραβήξει ο εκάστοτε χρήστης και το αυξάνει κατά ένα για να βρει τον αριθμό της τρέχουσας φωτογραφίας. Στην συνέχεια με την μέθοδο grabFrame("c://"+Username+imageId+".jpg") τραβάει την φωτογραφία με όνομα το όνομα του χρήστη μαζί με το id της φωτογραφίας. Τέλος με την μέθοδο dataBase.storeImageDB(id,Username+imageId) αποθηκεύει το όνομα της φωτογραφίας στην βάση δεδομένων. Με την εντολή LOGIN γίνονται δύο βασικές διαδικασίες. Πρώτων γίνεται η ταυτοποίηση του χρήστη που θέλει να συνδεθεί στη εφαρμογή μέσω του ονόματος χρήστη και του κωδικού πρόσβασης όπως επίσης και αν ο χρήστης έχει δικαιώματα διαχειριστή ή όχι. Δεύτερον καθορίζεται το UDP port στο οποίο θα στέλνει ο Server το βίντεο στο applet, αυτό γίνεται στέλνοντας ο Server το port που έχει ανοίξει. Η μορφή του μηνύματος που φτάνει στον Server είναι η εξής: LOGIN USERNAME,PASSWORD.. Εάν το μήνυμα περιέχει την λέξη LOGIN τότε ορίζουμε ένα δείκτη με το όνομα beginIndex με τιμή 5 και αρχικοποιούμε μία μεταβλητή Char τύπου Character. Σε αυτήν την μεταβλητή αποθηκεύουμε με την μέθοδο message.charAt(beginIndex) τον χαρακτήρα στην θέση 5 της συμβολοσειράς. Δηλαδή εάν μετά την λέξη LOGIN έχουμε την λέξη username τότε η τιμή της Char θα είναι ο χαρακτήρας u. Έπειτα στην συνέχεια μπαίνει σε έναν βρόχο και αποθηκεύουμε αυτόν τον χαρακτήρα στην μεταβλητή Username, αυξάνουμε τον δείκτη κατά ένα και αποθηκεύουμε τον επόμενο χαρακτήρα στην μεταβλητή Char και τον προσθέτουμε στην μεταβλητή Username. Αυτό γίνεται μέχρι να συναντήσουμε τον χαρακτήρα κόμμα. Μόλις βγούμε από τον βρόχο αυξάνουμε τον δείκτη κατά ένα και επαναλαμβάνουμε την ίδια διαδικασία για τον κωδικό πρόσβασης μέχρι ο χαρακτήρας να είναι η τελεία και τον αποθηκεύουμε στην μεταβλητή Password. Για να βρούμε εάν υπάρχει αυτός ο χρήστης καλούμε την μέθοδο findUser(Username,Password) από την κλάση DataBase και περνάμε σαν παραμέτρους το όνομα χρήστη και τον κωδικό πρόσβασης. Εάν υπάρχει ο χρήστης στην βάση δεδομένων τότε η μέθοδος findUser(Username, Password) επιστρέφει την λογική τιμή true. Στην συνέχεια αποθηκεύουμε το id του χρήστη για να το χρησιμοποιήσουμε στην συνέχεια όπως για την καταγραφή βίντεο. Δημιουργούμε ένα UDP port με την εντολή serverUDPSocket = new DatagramSocket(). Στην συνέχεια εξετάζουμε εάν ο χρήστης έχει δικαιώματα διαχειριστή ή όχι μέσω της μεθόδου dataBase.findIfAdmin(Username, Password). Εάν έχει δικαίωμα διαχειριστή τότε η μέθοδος επιστρέφει την τιμή true και στέλνει το εξής μήνυμα output.writeObject("CONFIRMED"+serverUDPSocket.getLocalPort()) στο applet. Στην περίπτωση όμως που δεν έχει δικαιώματα διαχειριστή τότε στέλνει το εξής μήνυμα output.writeObject("CONFIRMED"+serverUDPSocket.getLocalPort()+"noAdmin"). Και στις δύο περιπτώσεις μόλις στείλει το μήνυμα στο applet ο Server καταγράφει στη βάση δεδομένων την ημερομηνία και την ώρα που ο χρήστης συνδέθηκε στην εφαρμογή μέσω της μεθόδου dataBase.userLogIn(id,connection.getInetAddress().getHostAddress()) περνώντας σαν παραμέτρου το id του χρήστη και την ip του. Τέλος εάν τα στοιχεία που έχει εισάγει ο χρήστης στο applet είναι λανθασμένα τότε ο Server στέλνει το μήνυμα output.writeObject("DENIED") και περιμένει να εισάγει ο χρήστης τα σωστά στοιχεία κα να

επαναλάβει την διαδικασία. Εάν το μήνυμα περιέχει την λέξει schedule τότε ο Server θα χρονοπρογραμματίση την καταγραφή βίντεο σε μία χρονική περίοδο που θα την έχει ορίσει ο χρήστης από το applet. Αυτό γίνεται στέλνοντας μαζί με την λέξη schedule τις ημερομηνίες που θα αρχίσει η καταγραφή βίντεο και που θα τελειώσει. Το πλήρες μήνυμα που στέλνει το applet είναι scheduledateTimeStart,dateTimeEnd και θα εξηγηθεί πλήρως στο κεφάλαιο που αφορά το applet. Στην συνέχεια βρίσκει το id του βίντεο με τον τρόπο που αναφέραμε παραπάνω και αρχικοποιεί τρείς μεταβλητές οι οποίες είναι οι startDateTime, endDateTime και comma. Με την message = message.replace("schedule", "") διαγράφουμε την λέξη schedule από την συμβολοσειρά ώστε να μπορούμε να βρούμε τις δύο ημερομηνίες. Για να διαχωρίσουμε τις ημερομηνίες αρχικοποιούμε ένα πίνακα τύπου String, με την μέθοδο temp = message.split(comma) αποθηκεύουμε την ημερομηνία που αφορά το ξεκίνημα της καταγραφής στην θέση 0 του πίνακα και στην θέση 1 την ημερομηνία για το τέλος της καταγραφής. Τέλος αποθηκεύουμε αυτές τις ημερομηνίες στις αντίστοιχες μεταβλητές και τις περνάμε σαν παραμέτρους στην μέθοδο dataBase.ScheduleRec(startDateTime,endDateTime,Username,Username+videoId, id) μαζί με το όνομα χρήστη και το όνομα του βίντεο για να αρχίσει ο χρονοπρογραμματισμός. Εάν ο χρήστης αποσυνδεθεί τότε το applet στέλνει την λέξη END και έτσι βγαίνει από τον βρόχο ώστε να αποδέσμευση του πόρους του συστήματος όπως τα ports που έχει δεσμεύσει. Μόλις βγει από τον βρόχο υπάρχουν δύο περιπτώσεις. Η μία περίπτωση είναι να γίνεται streaming την στιγμή που αποσυνδέεται και η άλλη είναι να μην γίνεται streaming. Στην πρώτη περίπτωση αλλάζουμε την τιμή της μεταβλητής streaming σε false για να σταματήσει η αποστολή βίντεο επίσης αλλάζουμε και την running σε false για να τερματίσει το συγκεκριμένο νήμα επεξεργασίας. Τέλος κλείνουμε όλες τις συνδέσεις που αφορούν τον συγκεκριμένο χρήστη με τις input.close() output.close() connection.close() και με την μέθοδο dataBase.userLogOut(id) καταγράφουμε στην βάση δεδομένων την ημερομηνία και την ώρα που αποσυνδέθηκε ο χρήστης. Στην δεύτερη περίπτωση το διαφορετικό που κάνουμε είναι να μην αλλάζουμε την τιμή της streaming και να κλείνουμε το UDP port. Τα διάγραμμα ροής των setupStreams και whileConnected αντίστοιχα φαίνονται παρακάτω:



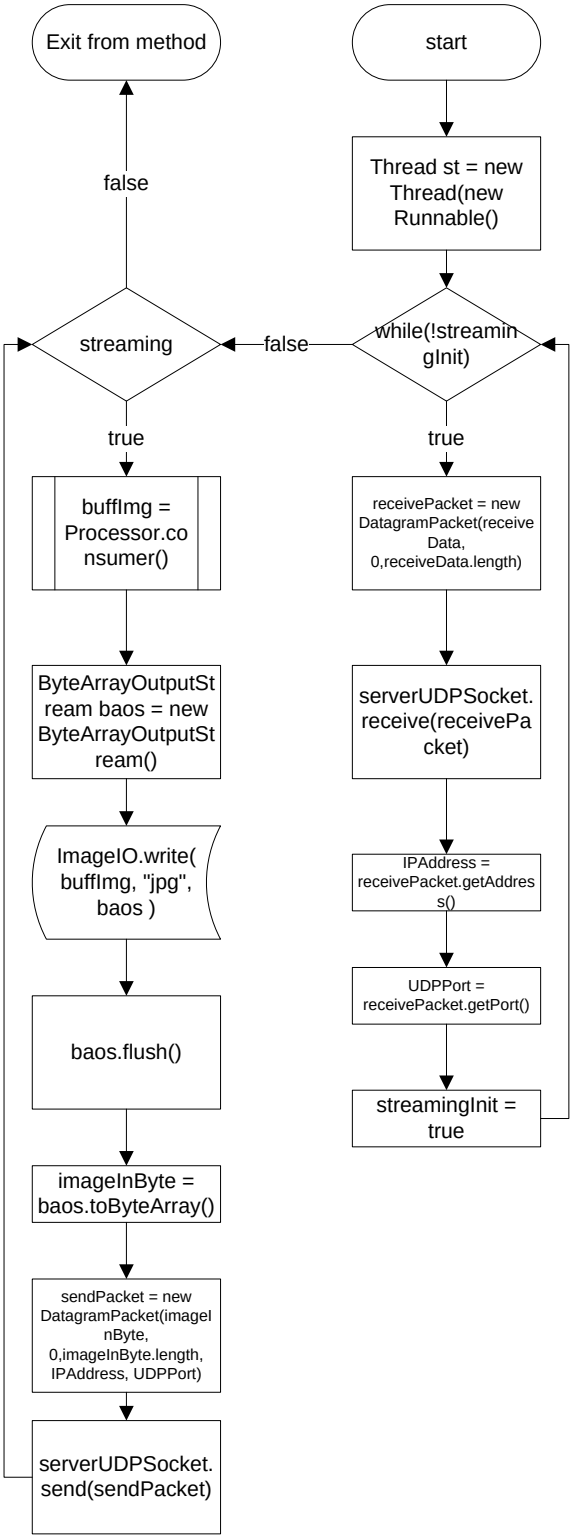

start
message = (String) input.readObject()
message.equals("PLAY")
true
streaming = true
streaming()
message.equals("PAUSE")
true
streaming = false
false
message.equals("CAPTURE_VIDEO_START")
videoId = dataBase.findVideoId(id)+1;
videoWriter = ToolFactory.makeWriter("c://"+Username+videoId+".mp4")
videoWriter.addVideoStream(0, 0, ICodec.ID.CODEC_ID_MPEG4,320, 240)
record = true
startTime = System.currentTimeMillis()
recordingVideo()
threadRec.start()
dataBase.storeVideoDBStart(id, Username+videoId)
message = "nothing"
false
message.equals("CAPTURE_VIDEO_STOP")
true
record = false
stopRecording = true
stopTime = System.currentTimeMillis()
duration = stopTime - startTime
dataBase.storeVideoDBStop(duration/1000, id)
message = "nothing"
false
message.equals("initStreaming")
true
streaming = true
streaming()
message = "nothing"
false
1
2
3


3
1
false
message.equals("GRABFRAME")
true
int imageId = dataBase.findImageId(id)+1
processor.grabFrame("c://"+Username+imageId+".jpg")
dataBase.storeImageDB(id,Username+imageId)
message = "nothing"
false
message.contains("LOGIN")
true
beginIndex = 5
Character Char = new Character(message.charAt(beginIndex))
Username = Username+Char
beginIndex++
Char = new Character(message.charAt(beginIndex))
!Char.equals(";")
true
beginIndex = beginIndex+1
Char = new Character(message.charAt(beginIndex))
Password = Password+Char
beginIndex++
Char = new Character(message.charAt(beginIndex))
!Char.equals(";")
true
false
dataBase.findUser(Username,Password)
dataBase.findUser(Username, Password)
false
true
id = dataBase.id
serverUDPSocket = new DatagramSocket()
dataBase.findAdmin(Username, Password)
true
false
output.writeObject("CONFIRMED"+serverUDPSocket.getLocalPort())
output.flush()
output.writeObject("CONFIRMED"+serverUDPSocket.getLocalPort()+"noAdmin")
output.flush()
dataBase.userLogIn(id,connection.getInetAddress().getHostAddress()
false
output.writeObject("DENIED")
output.flush()
Username = ""
Password = ""
message = "nothing"
false
videoId = dataBase.findVideoId(id)+1
String startDateTime = ""
String endDateTime = ""
String comma = ","
message = message.replace("schedule", "")
String[] temp
temp = message.split(comma)
startDateTime = temp[0]
endDateTime = temp[1]
dataBase.SchedRect(startDateTime,endDateTime,Username, Username+videoId, id)
message = "nothing"
false
2
true
message.equals("END")
false
message.equals("END") && !streaming
message.equals("END") && streaming
true
input.close()
output.close()
running = false
connection.close()
serverUDPSocket.close()
dataBase.userLogOut(id)
true
stopStreaming()
streaming = false
running = false
input.close()
output.close()
connection.close()
dataBase.userLogOut(id)
exit

Η μέθοδος streaming στέλνει την εικόνα στο applet. Η μέθοδος αυτή περιέχει ένα νήμα επεξεργασίας το οποίο έχουμε ονομάσει st. Μέσα στην run έχουμε δύο βρόχους while. Ο πρώτο βρόχος εκτελείται όσο η μεταβλητή streamingInit έχει την τιμή false. Τον συγκεκριμένο βρόχο τον χρησιμοποιούμε για να ανοίξουμε ένα UDP port στο δρομολογητή. Όπως αναφέραμε παραπάνω ο Server έχει δεσμεύσει ένα UDP port για την αποστολή της εικόνας στο applet, άρα στέλνοντας από το applet ένα UDP πακέτο στο port που έχει δεσμεύσει ο Server ο δρομολογητής γνωρίζει σε ποιόν υπολογιστή θα δρομολογήσει τα UDP πακέτα που θα στείλει ο Server και θα περιέχουν την εικόνα. Συνεπώς στον πρώτο βρόχο φτιάχνουμε ένα DatagramPacket με όνομα receivePacket για να δεχτούμε τα δεδομένα που μας στέλνει το applet. Τα δεδομένα αυτά τα διαβάζουμε με την εντολή serverUDPSocket.receive(receivePacket). Στην μεταβλητή IPAddress αποθηκεύουμε την ip του applet και στην μεταβλητή UDPPort το port του applet που είναι ίδιο με το port του Server. Τέλος αλλάζουμε την τιμή της μεταβλητής streamingInit σε true για να μην ξαναμπεί το πρόγραμμα στον βρόχο. Ο δεύτερος βρόχος χρησιμοποιείτε από το πρόγραμμα για την αποστολή των εικόνων στο applet. Αρχικά αποθηκεύουμε ένα καρέ στη μεταβλητή buffImg και με την παρακάτω διαδικασία μετατρέπουμε το καρέ σε ένα πίνακα από byte με όνομα imageInByte για να τα στείλουμε στο applet. Με την sendPacket = new DatagramPacket(imageInByte, 0,imageInByte.length, IPAddress, UDPPort) δημιουργούμε ένα νέο udp πακέτο το οποίο περιέχει την ip το udp port και τον πίνακα imageInByte. Τέλος με την serverUDPSocket.send(sendPacket) στέλνουμε αυτό το udp πακέτο σε μικρότερα πακέτα των 64kb και μόλις ολοκληρωθεί η αποστολή των πακέτων περιμένουμε με την μέθοδο sleep() 100ms για να στείλουμε τα επόμενα δεδομένα. Παρακάτω φαίνονται η μέθοδος και το διάγραμμα ροής:

```
public void streaming(){
        Thread st = new Thread(new Runnable(){
            @Override
            public void run() {
                while(!streamingInit){
    receivePacket = new DatagramPacket(receiveData, 0,receiveData.length);
                try {
                    serverUDPSocket.receive(receivePacket);
                } catch (IOException e) {
                    e.printStackTrace();
                }
                IPAddress = receivePacket.getAddress();
                UDPPort = receivePacket.getPort();
                streamingInit = true;
                }
                while(streaming){
                buffImg = Processor.consumer();
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
                try {
                    ImageIO.write(buffImg, "jpg", baos );
                    baos.flush();
                } catch (IOException e1) {
                    e1.printStackTrace();
                }
```

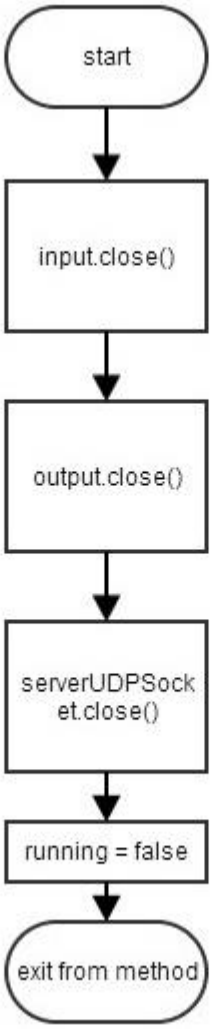
```
                                        imageInByte = baos.toByteArray();
sendPacket = new DatagramPacket(imageInByte, 0,imageInByte.length, IPAddress,
UDPPort);
                                        try {
                                                serverUDPSocket.send(sendPacket);
                                        } catch (IOException e) {
                                        }
                                        sleep();
                                    }
                                }
                        });
                        st.start();
                }
```



Την μέθοδο stopStreaming την χρησιμοποιούμε για να κλείσουμε τα ανοικτά ports. Με τα input.close() και output.close() κλείνουμε τα TCP port ενώ με serverUDPSocket.close() του UDP port του Server. Και τέλος αλλάζουμε την μεταβλητή της τιμής running σε false για να τερματιστεί το νήμα επεξεργασίας που είναι υπεύθυνο για την επικοινωνία μεταξύ Server και applet. Παρακάτω φαίνεται το διάγραμμα η stopStreaming και το διάγραμμα ροής της:

public void stopStreaming(){

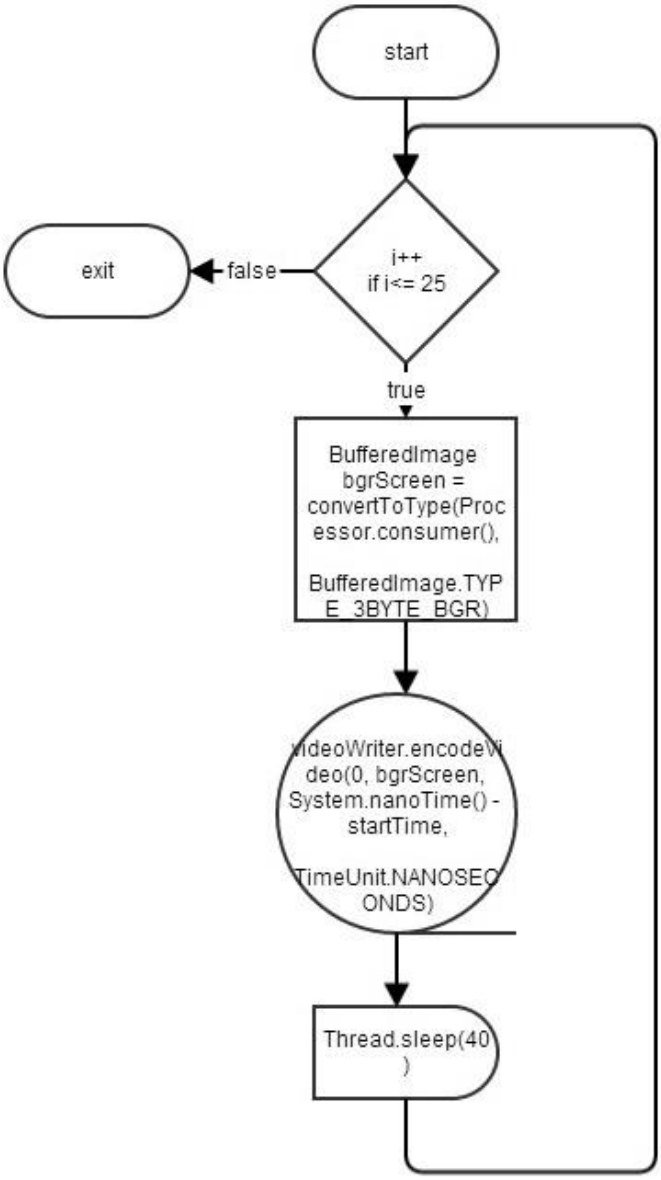
```
            try {
                input.close();
                output.close();
                serverUDPSocket.close();
                running = false;
            } catch (IOException e) {
                e.printStackTrace();
            }
}
```



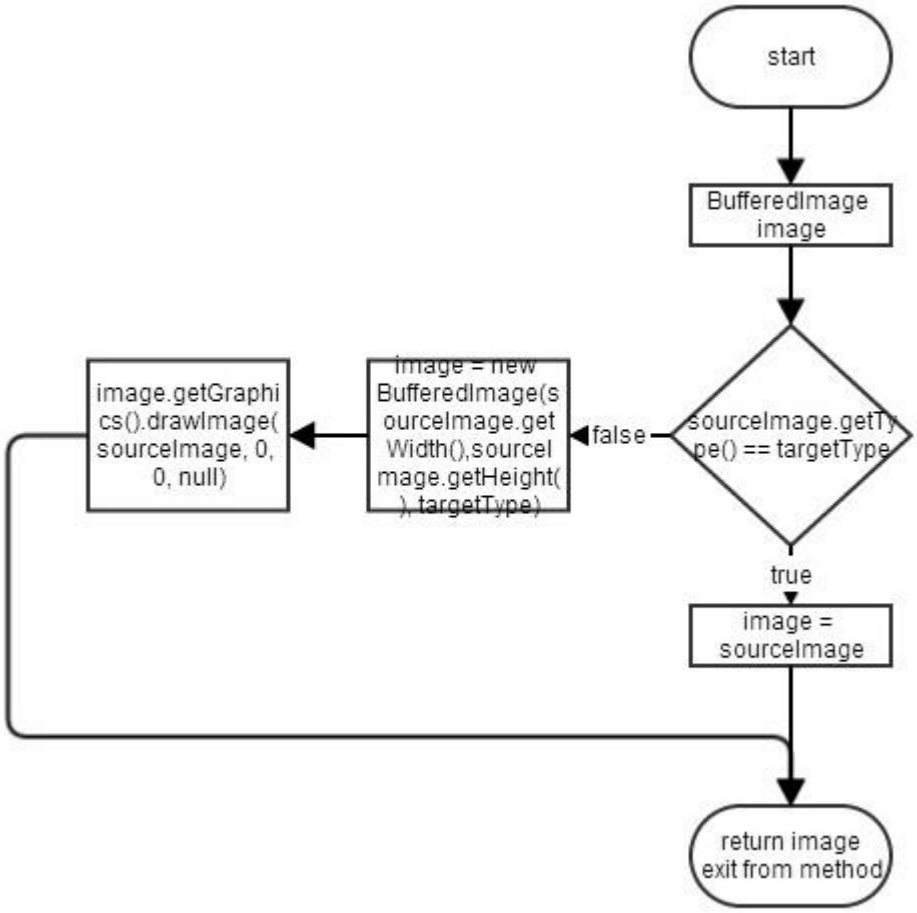
Η μέθοδος recordingVideo() χρησιμοποιείτε από το πρόγραμμα για την εγγραφή βίντεο και περιέχει ένα νήμα επεξεργασίας το οποίο το έχουμε ονομάσει threadRec. Μέσα στην run υπάρχει ένας βρόχος while. Ο βρόχος καλεί την μέθοδο recVideo() η οποία παίρνει τα καρέ και με την βοήθεια της convertToType() τα μετατρέπει σε κατάλληλη μορφή για την δημιουργία του βίντεο. Μέσα στην recordingVideo() υπάρχει ένα βρόχος for ο οποίος επαναλαμβάνεται 25 φορές το δευτερόλεπτο λόγω ότι θέλουμε το βίντεο να έχει 25 καρέ. Μέσα στην for φτιάχνουμε ένα BufferedImage με την ονομασία bgrScreen και αποθηκεύουμε το καρέ το οποίο το έχουμε μετατρέψει με την βοήθεια της convertToType(Processor.consumer(),BufferedImage.TYPE_3BYTE_BGR). Μέσα στην convertToType περνάμε σαν παραμέτρους το καρέ από την Processor.consumer() και τον

τύπο που θέλουμε να το μετατρέψουμε. Έπειτα γράφουμε το καρέ στο βίντεο με την μέθοδο videoWriter.encodeVideo(0, bgrScreen, System.nanoTime() startTime,TimeUnit.NANOSECONDS) και περιμένουμε 40 ms για να επιτευχθεί ο στόχος των 25 καρέ το δευτερόλεπτο. Μέσα στην convertToType() αρχικοποιούμε ένα BufferedImage με το όνομα image. Εάν το καρέ που έχουμε περάσει σαν παράμετρο είναι ίδιος τύπος με τύπο εικόνας που θέλουμε τότε η μέθοδος απλά επιστρέφει το καρέ. Στην περίπτωση που δεν είναι ίδιος τύπος φτιάχνουμε ένα νέο BufferedImage με την μέθοδο new BufferedImage(sourceImage.getWidth(),sourceImage.getHeight(), targetType) το οποίο έχει τις ίδιες διαστάσει με το πηγαίο και το τύπο εικόνας που θέλουμε. Στην συνέχεια με την μέθοδο image.getGraphics().drawImage(sourceImage, 0, 0, null) σχεδιάζουμε το πηγαίο καρέ στο νέο BufferedImage που φτιάξαμε παραπάνω με το όνομα image και τέλος το επιστρέφουμε με την εντολή return image. Όπως αναφέραμε παραπάνω όταν ο χρήστης θελήσει να σταματήσει την εγγραφή βίντεο αλλάζουμε την μεταβλητή record σε false και την τιμή της stopRecording σε true. Αλλάζοντας την stopRecording σε true με την μέθοδο videoWriter.close() σταματάμε την εγγραφή βίντεο και τι αποδεσμεύουμε από το πρόγραμμα. Τέλος αλλάζουμε την stopRecording σε false για να έχει την δυνατότητα ο χρήστης να καταγράψει και πάλι βίντεο. Παρακάτω φαίνονται οι recVideo(),convertToType(BufferedImage sourceImage, int targetType) και τα διαγράμματα ροής τους αντίστοιχα.

```
public void recVideo(){
            // convert to the right image type
                    for(int i=0; i<25; i++){
                BufferedImage bgrScreen = convertToType(Processor.consumer(),
                    BufferedImage.TYPE_3BYTE_BGR);
          // encode the image to stream #0
                videoWriter.encodeVideo(0, bgrScreen, System.nanoTime() - startTime,
                    TimeUnit.NANOSECONDS);
                // sleep for frame rate milliseconds
                try {
                   Thread.sleep(40);
                }
                catch (InterruptedException e) {
                   // ignore
                }
                        }
                }
```




```
public static BufferedImage convertToType(BufferedImage sourceImage, int targetType) {
                        BufferedImage image;
                    // if the source image is already the target type, return the source image
                        if (sourceImage.getType() == targetType) {
                            image = sourceImage;
                        }
                        // otherwise create a new image of the target type and draw the
new image
                    else {
                       image = new
BufferedImage(sourceImage.getWidth(),sourceImage.getHeight(), targetType);
                        image.getGraphics().drawImage(sourceImage, 0, 0, null);
                        }
                        return image;
                }
```



2.5 Η κλάση DataBase

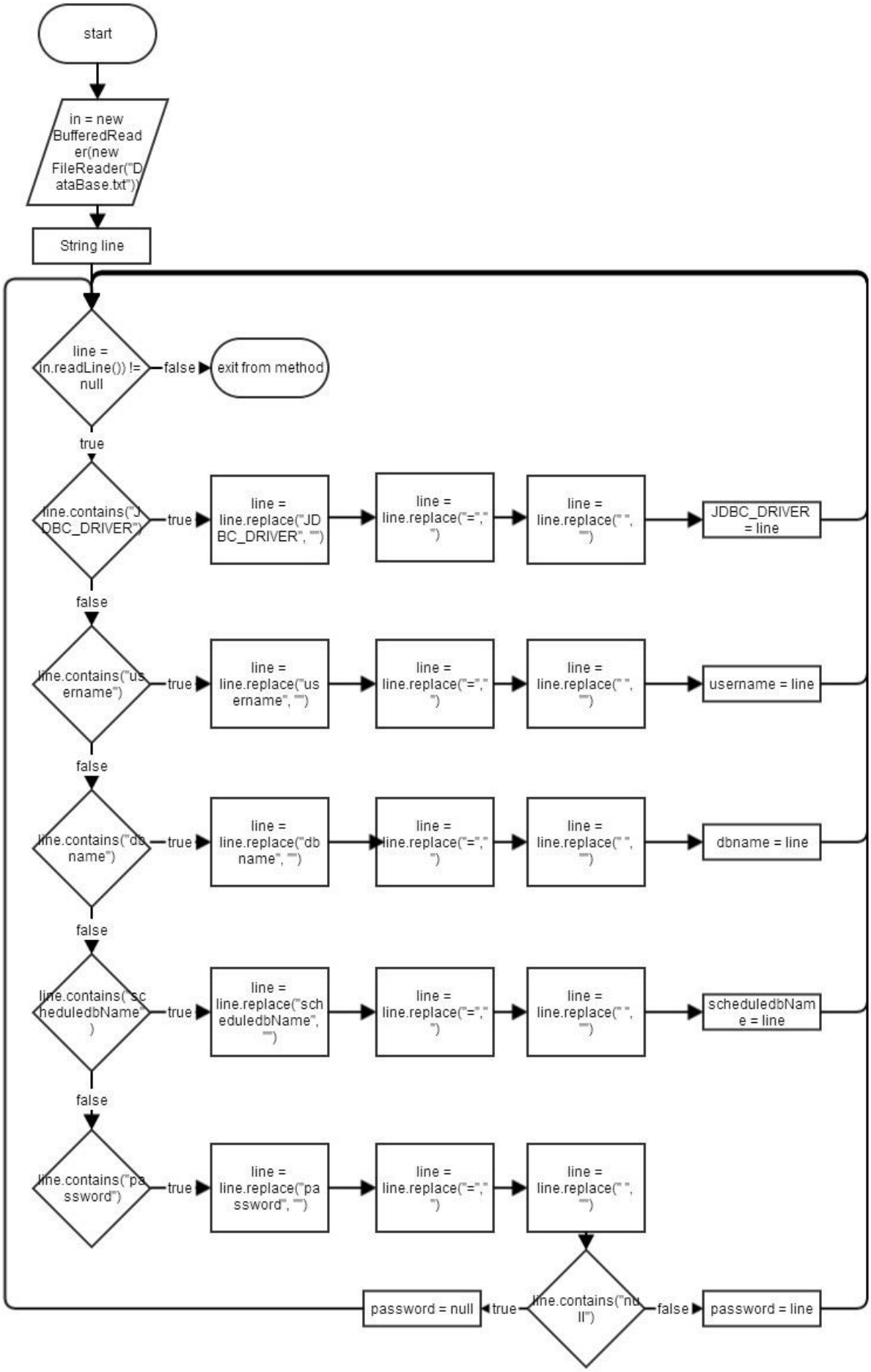
Η κλάση DataBase όπως δηλώνει και το όνομα της χρησιμοποιείτε από το πρόγραμμα για την εισαγωγή, επεξεργασία και την αναζήτηση στοιχείων στην βάση δεδομένων. Τα στοιχεία αυτά είναι: όνομα χρήστη, κωδικός πρόσβασης, δικαιώματα χρήστη, καταγραφή σύνδεσης και αποσύνδεσης χρήστη στην εφαρμογή, καταγραφή αριθμού και ονόματος εικόνας ή βίντεο που έχει τραβήξει ο εκάστοτε χρήστης και καταγραφή φωτεινότητας και θερμοκρασίας από τον iLON κάθε 1 λεπτό. Επίσης στην βάση δεδομένων αποθηκεύονται οι εγγραφές των βίντεο που είναι προγραμματισμένες από του χρήστες. Για τον χρονοπρογραμματισμό εγγραφής βίντεο χρησιμοποιούμε μια εξωτερική βιβλιοθήκη της Java με το όνομα quartz. Επίσης για να συνδεθεί η εφαρμογή με την βάση δεδομένων χρησιμοποιούμε μια άλλη εξωτερική βιβλιοθήκη με το όνομα JDBC. Όπως είχαμε αναφέρει στην παράγραφο που αφορούσε την κλάση Main καλούσαμε δύο μεθόδους της DataBase οι οποίες είναι οι εξής: dataBaseInfo() και Connect(). Με την dataBaseInfo() διαβάζουμε ένα αρχείο τύπου txt το οποίο θα πρέπει να βρίσκεται στον ίδιο φάκελο με το εκτελέσιμο αρχείο του Server και περιέχει πληροφορίες την βάσης δεδομένων. Το αρχείο DataBase έχει την παρακάτω δομή:

JDBC_DRIVER = com.mysql.jdbc.Driver
username = root
password = null
dbname = jdbc:mysql://localhost/ptixiaki
scheduledbName = jdbc:mysql://localhost/scheduler.
Η βάση δεδομένων που χρησιμοποιούμε είναι η mysql άρα ο οδηγός που θα χρησιμοποιήσουμε για να συνδεθεί η εφαρμογή με την βάση έχει το όνομα com.mysql.jdbc.Driver. Παρακάτω δηλώνουμε το username και το password της βάσης δεδομένων. Η dbname και η scheduledbName είναι οι βάσεις δεδομένων που χρησιμοποίει η εφαρμογή και δηλώνουμε την διεύθυνση και το όνομα κάθε βάσης. Παρακάτω φαίνεται και εξηγείτε η λειτουργία της dataBaseInfo().

```
public void dataBaseInfo(){
	try {
			in = new BufferedReader(new FileReader("DataBase.txt"));
			String line;
			while((line = in.readLine()) != null)
			{
				if(line.contains("JDBC_DRIVER")){
					line = line.replace("JDBC_DRIVER", "");
					line = line.replace("=","");
					line = line.replace(" ", "");
					JDBC_DRIVER = line;
				}
				if(line.contains("username")){
					line = line.replace("username", "");
					line = line.replace("=","");
					line = line.replace(" ", "");
					username = line;
				}
				if(line.contains("dbname")){
					line = line.replace("dbname", "");
					line = line.replace("=","");
					line = line.replace(" ", "");
					dbname = line;
				}
				if(line.contains("scheduledbName")){
					line = line.replace("scheduledbName", "");
					line = line.replace("=","");
					line = line.replace(" ", "");
					scheduledbName = line;
				}
				if(line.contains("password")){
					line = line.replace("password", "");
					line = line.replace("=","");
					line = line.replace(" ", "");
					if(line.contains("null")){
						password = null;
					}else{
					password = line;
```

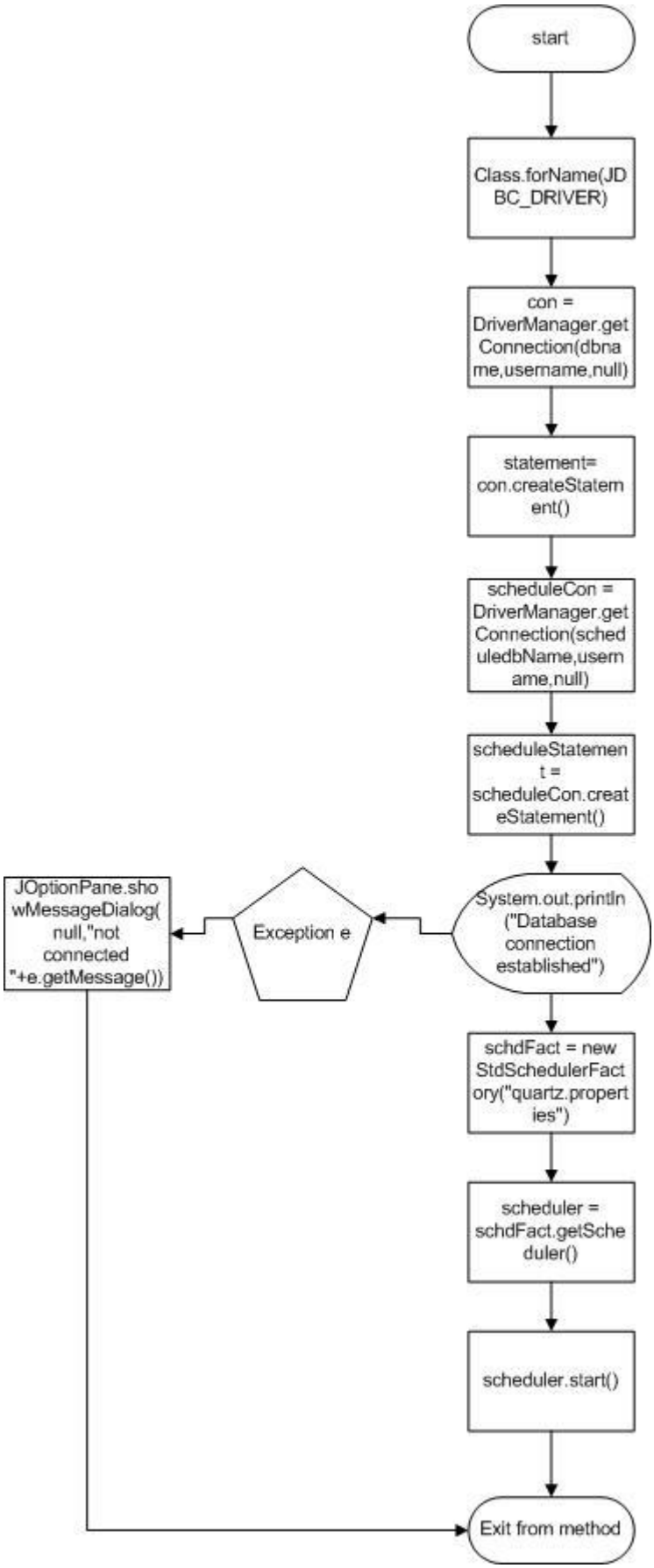
```
                                        }
                                    }
                                }
                            } catch (IOException e) {
                                e.printStackTrace();
                            }
    }
```

Με την εντολή in = new BufferedReader(new FileReader("DataBase.txt")) το πρόγραμμα διαβάζει μια γραμμή κειμένου. Μια γραμμή θεωρείται ότι πρέπει να τερματίζεται με μια αλλαγή γραμμής ('\ n'), μια μεταφορά μετ 'επιστροφής (' \ r '), ή επαναφοράς ακολουθούμενη αμέσως από μια αλλαγή γραμμής. Έπειτα μπαίνει σε έναν βρόχο while ο οποίος επαναλαμβάνεται όσο υπάρχουν σειρές στο κείμενο. Στην συνέχεια γίνονται διαδοχικές συγκρίσεις με την μεταβλητή line που είναι αποθηκευμένη η σειρά του κειμένου για να βρεθεί σε ποιά σειρά βρίσκεται την συγκεκριμένη στιγμή. Η πρώτη σύγκριση που γίνεται είναι με την συμβολοσειρά JDBC_DRIVER, εάν η συνθήκη είναι έγκυρη τότε το πρόγραμμα σβήνει την συμβολοσειρά JDBC_DRIVER με την line = line.replace("JDBC_DRIVER", "") όπως και τους χαρακτήρες ‘=’ και ‘κενό’ με την line = line.replace("=", "") και line = line.replace(" ", "") αντίστοιχα. Τέλος το όνομα του οδηγού αποθηκεύετε στην μεταβλητή JDBC_DRIVER για να την χρησιμοποιήσουμε στην συνέχεια του προγράμματος. Αυτή η διαδικασία επαναλαμβάνεται για όλε τις παραμέτρους που έχουμε περάσει στο αρχείο DataBase. Το διάγραμμα ροής της dataBaseInfo():


start
in = new BufferedReader(new FileReader("DataBase.txt"))
String line
line = in.readLine()) != null
false
exit from method
true
line.contains("JDBC_DRIVER")
true
line = line.replace("JDBC_DRIVER", "")
line = line.replace("=","")
line = line.replace(" ","")
JDBC_DRIVER = line
false
line.contains("username")
true
line = line.replace("username", "")
line = line.replace("=","")
line = line.replace(" ","")
username = line
false
line.contains("dbname")
true
line = line.replace("dbname", "")
line = line.replace("=","")
line = line.replace(" ","")
dbname = line
false
line.contains("scheduledbName")
true
line = line.replace("scheduledbName", "")
line = line.replace("=","")
line = line.replace(" ","")
scheduledbName = line
false
line.contains("password")
true
line = line.replace("password", "")
line = line.replace("=","")
line = line.replace(" ","")
line.contains("null")
false
password = line
true
password = null

Για να συνδεθεί το πρόγραμμα με την βάση δεδομένων χρησιμοποιεί την μέθοδο Connect. Αρχικά μέσα στην μέθοδο Connect λέμε στο πρόγραμμα να φορτώσει τον mysql οδηγό με την εντολή Class.forName(JDBC_DRIVER) και έπειτα φτιάχνουμε μία σύνδεση μεταξύ της εφαρμογής και της βάσης δεδομένων με όνομα con και περνώντας σαν παραμέτρους το όνομα, το όνομα χρήστη και το κωδικό πρόσβασης της βάσης δεδομένων. Στην συνέχεια δημιουργούμε μία δήλωση που μας επιτρέπει να κάνουμε ερωτήματα στη βάση δεδομένων SQL με την statement= con.createStatement(). Η σύνδεση που έχουμε φτιάξει με το όνομα con αφορά την βάση δεδομένων στην οποία έχουμε αποθηκεύσει στοιχεία όπως τα ονόματα των χρηστών τους κωδικούς πρόσβασης, το πλήθος των εικόνων και των βίντεο που έχει καταγράψει ο εκάστοτε χρήστης και την διάρκεια των βίντεο. Παρακάτω φτιάχνουμε μία σύνδεση με την βάση δεδομένων με όνομα scheduleCon στην οποία αποθηκεύουμε τις εγγραφές βίντεο που έχει προγραμματίσει ο κάθε χρήστης. Με την System.out.println ("Database connection established") εμφανίζουμε στην κονσόλα το μήνυμα ότι η συνδέσεις επιτευχθήκαν με επιτυχία, εάν για κάποιο λόγο η εφαρμογή δεν μπορέσει να συνδεθεί με τις βάσεις δεδομένων τότε δημιουργείτε μια εξαίρεση και εμφανίζεται ένα παράθυρο με την εντολή JOptionPane.showMessageDialog(null,"not connected "+e.getMessage()) το οποίο περιέχει το σφάλμα. Στην συνέχεια για να φτιάξουμε έναν χρονοπρογραμματιστή θα πρέπει να ορίσουμε τις ιδιότητες του μέσω της κλάσης οι οποίες βρίσκονται σε ένα αρχείο με το όνομα quartz.properties. Για να διαβάσουμε αυτό το αρχείο ορίζουμε μία μεταβλητή με το όνομα schdFact και τύπου StdSchedulerFactory και την αρχικοποιούμε με το εξής τρόπο schdFact = new StdSchedulerFactory("quartz.properties") περνώντας το όνομα του αρχείου σαν παράμετρο. Έπειτα αρχικοποιούμε τον χρονοπρογραμματιστή με το όνομα scheduler με τον εξής τρόπο scheduler = schdFact.getScheduler(). Τέλος τον θέτουμε σε λειτουργία με την scheduler.start(). Παρακάτω φαίνονται η μέθοδος Connect και το διάγραμμα ροή της:

```
public void Connect() {
      try {
         Class.forName(JDBC_DRIVER);
         con = DriverManager.getConnection(dbname,username,null);
         statement= con.createStatement();
         scheduleCon = DriverManager.getConnection(scheduledbName,username,null);
         scheduleStatement = scheduleCon.createStatement();
         System.out.println ("Database connection established");
      }
      catch (Exception e) {
         JOptionPane.showMessageDialog(null,"not connected "+e.getMessage());
      }
      try {
schdFact = new StdSchedulerFactory("quartz.properties");
                        scheduler = schdFact.getScheduler();
                scheduler.start();
                 } catch (SchedulerException e2) {
                        e2.printStackTrace();
                 }
   }
```


start
Class.forName(JDBC_DRIVER)
con = DriverManager.getConnection(dbname,username,null)
statement= con.createStatement()
scheduleCon = DriverManager.getConnection(scheduledbName,username,null)
scheduleStatement = scheduleCon.createStatement()
System.out.println("Database connection established")
Exception e
JOptionPane.showMessageDialog(null,"not connected "+e.getMessage())
schdFact = new StdSchedulerFactory("quartz.properties")
scheduler = schdFact.getScheduler()
scheduler.start()
Exit from method

Η μέθοδος που χρησιμοποιούμε για να καταχωρίσουμε ένα νέο χρήστη στην βάση δεδομένων ονομάζεται registerUser(username, password, rights) και παίρνει σαν παραμέτρους το όνομα χρήστη, τον κωδικό πρόσβασης και τα δικαιώματα χρήστη. Αυτά τα εισάγουμε μέσα από το GUI του Server. Αρχικά μέσα στον δημιουργό της κλάσης DataBase έχουμε δημιουργήσει τρία πεδία κειμένου και ένα κουμπί με το όνομα register. Εάν ο χρήστης εισάγει τα στοιχεία στα αντίστοιχα πεδία και πατήσει το κουμπί register τότε τα στοιχεία αυτά θα αποθηκευτούν στις αντίστοιχες μεταβλητές με τον τρόπο που φαίνεται παρακάτω: register.addActionListener(new ActionListener(){

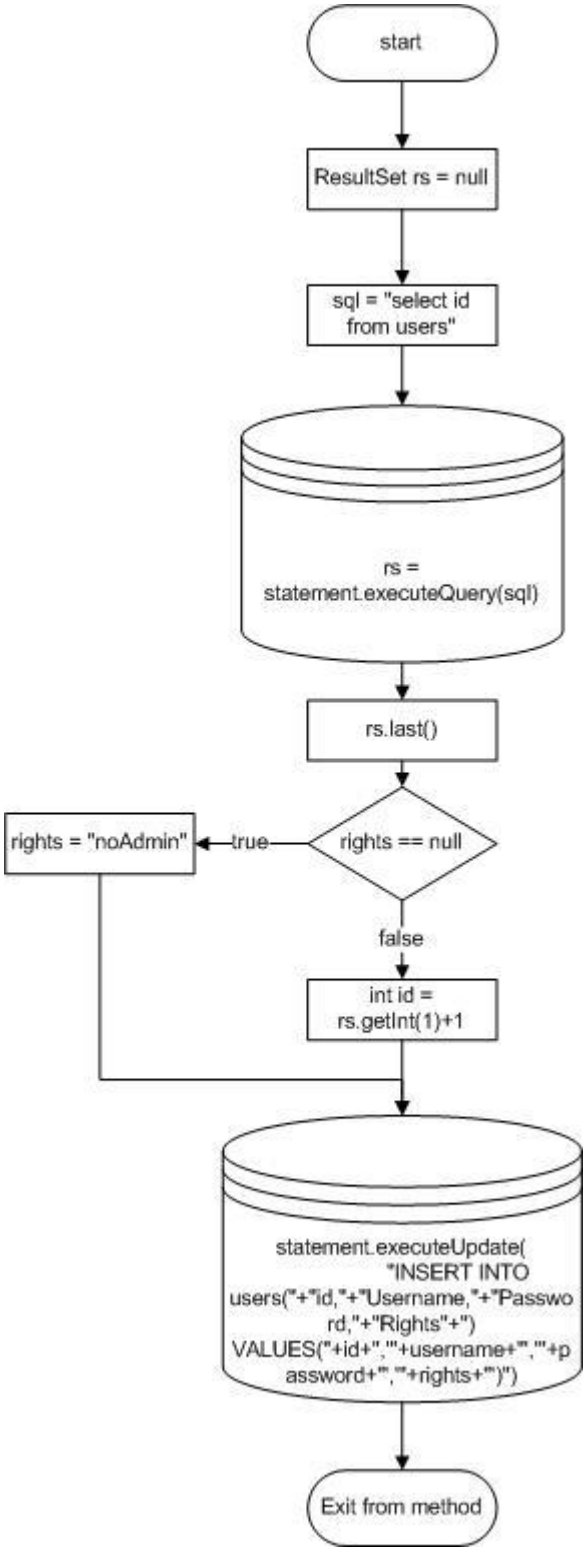
```
                public void actionPerformed(ActionEvent e) {
                        String username = UsernameTF.getText();
                        String password = PasswordTF.getText();
                        String rights = rightsTF.getText();
                        registerUser(username,password,rights);
                }
        });.
```

Πατώντας το κουμπί register αυτόματα καλείτε η παραπάνω μέθοδος και με την getText() παίρνουμε το κείμενο των πεδίων και το αποθηκεύουμε στις αντίστοιχες μεταβλητές. Τέλος καλούμε την μέθοδο registerUser(username,password,rights). Η μέθοδος registerUser(username,password,rights) φαίνεται παρακάτω:

```
        public void registerUser(String username, String password, String rights){
                ResultSet rs = null;
                sql = "select id from users";
                try {
                        rs = statement.executeQuery(sql);
                        rs.last();
                } catch (SQLException e1) {
                        e1.printStackTrace();
                }
                if(rights == null){
                        rights = "noAdmin";
                }
                try {
                        int id = rs.getInt(1)+1;
                        statement.executeUpdate(
        "INSERT INTO users("+"id,"+"Username,"+"Password,"+"Rights"+")
VALUES("+id+",'"+username+"','"+password+"','"+rights+"')");
                } catch (SQLException e) {
                        e.printStackTrace();
                }
        }
```

Μέσα στην μέθοδο registerUser ορίζουμε μία μεταβλητή τύπου ResultSet και με όνομα rs. Οι μεταβλητές αυτές αποθηκεύουν δεδομένα από ερωτήματα που έχουμε κάνει στην βάση δεδομένων. Έπειτα ορίζουμε το ερώτημα προς την βάση δεδομένων που στην συγκεκριμένη περίπτωση είναι να διαλέξει την στήλη id από τον πίνακα users. Έπειτα θέτουμε αυτό το ερώτημα προς την βάση δεδομένων με την rs = statement.executeQuery(sql) και αποθηκεύουμε το αποτέλεσμα στην μεταβλητή rs. Για να βρούμε το πλήθος των χρηστών που

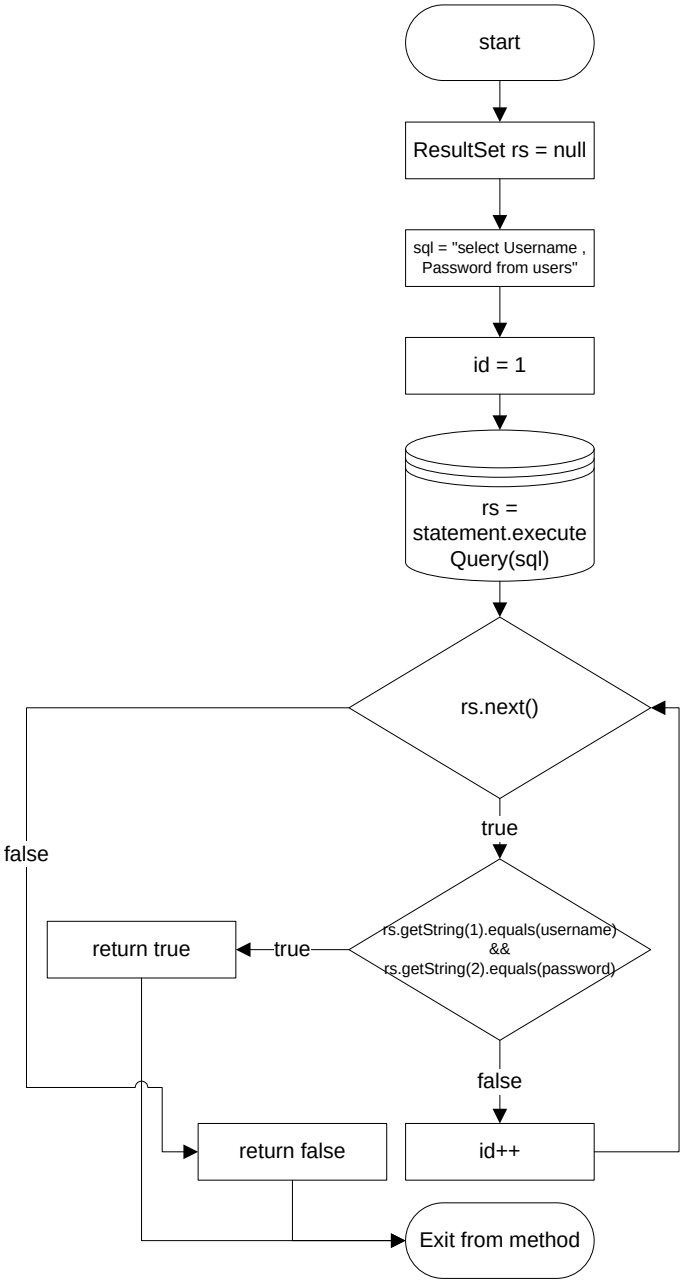
είναι καταχωρημένοι στην βάση θέτουμε τον δείκτη με την μέθοδο rs.last() στην τελευταία σειρά. Εάν ο χρήστης δεν έχει δικαιώματα διαχειριστή τότε μεταβλητή rights παίρνει την τιμή noAdmin. Τέλος αυξάνουμε το id του χρήστη κατά ένα και εκτελούμε το ερώτημα προς την βάση δεδομένων με την statement.executeUpdate("INSERT INTO users("+"id,"+"Username,"+"Password,"+"Rights"+") VALUES("+id+",'"+username+"','"+password+"','"+rights+"')") εισάγοντας τα τρία στοιχεία στα αντίστοιχα πεδία του πίνακα. Παρακάτω φαίνεται το διάγραμμα ροής της μεθόδου:



Όπως είχαμε αναφέρει στην παράγραφο που αφορούσε την κλάση Client όταν ένα χρήστης συνδεθεί στον Server ο Server ψάχνει μέσω της μεθόδου findUser(username, password) εάν υπάρχει ο χρήστης αυτός. Η μέθοδος findUser(username, password) φαίνεται παρακάτω:

```
public boolean findUser(String username, String password){
                ResultSet rs = null;
                sql = "select Username , Password from users";
                id = 1;
                try {
                        rs = statement.executeQuery(sql);
                   while(rs.next()){
                                if(rs.getString(1).equals(username) &&
rs.getString(2).equals(password)){
                                        return true;
                                }
                                id++;
                   }
                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return false;
        }
```

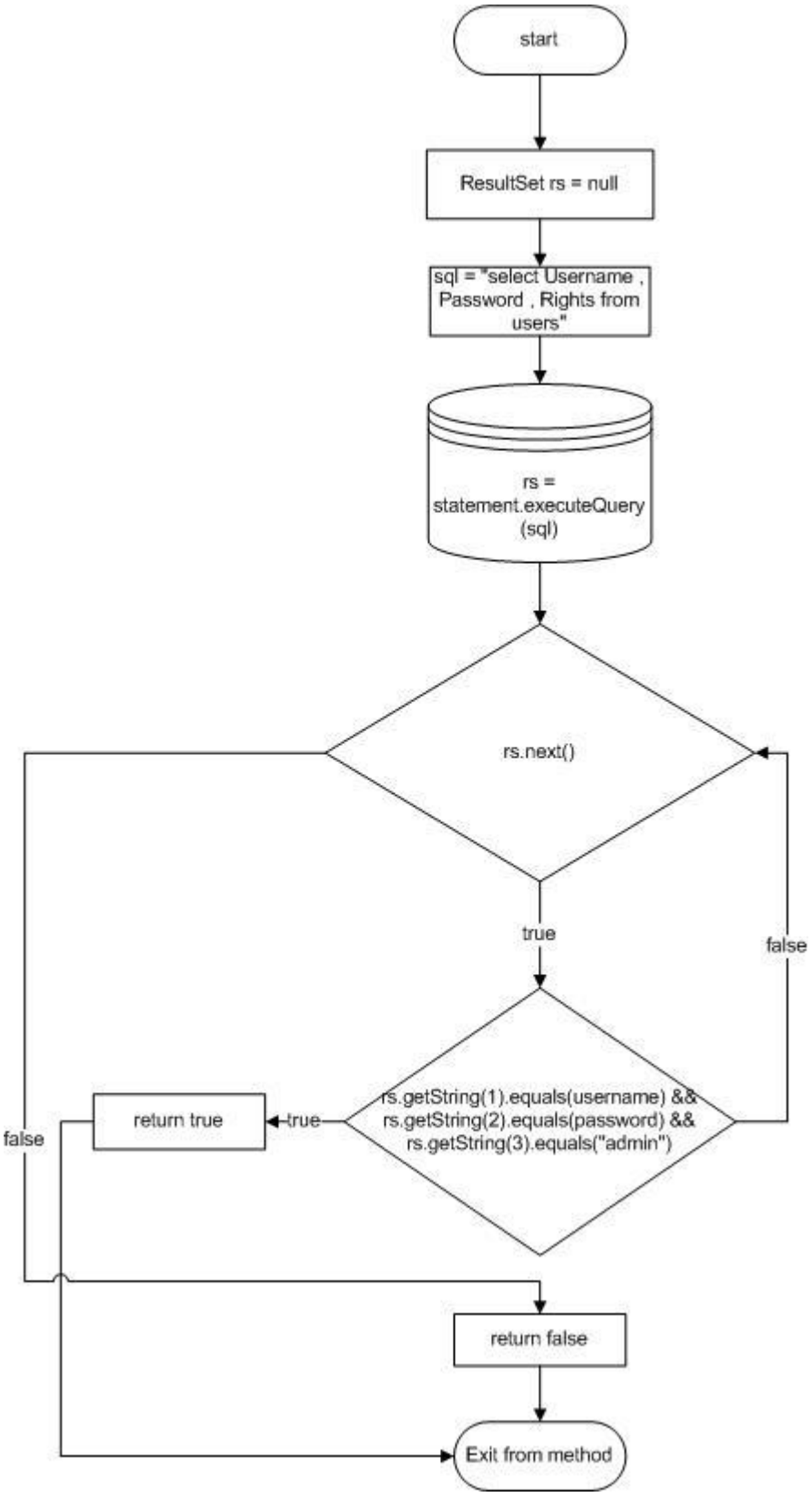
Ορίζουμε πάλι μία μεταβλητή τύπου ResultSet με όνομα rs όπως πριν και θέτουμε το ερώτημα στην βάση δεδομένων όμως αυτή την φορά επιλέγοντας τις στήλες username και password από τον πίνακα users. Θέτουμε το id ίσο με ένα και εκτελούμε το ερώτημα στην βάση. Έπειτα μπαίνει σε ένα βρόχο while ο οποίος επαναλαμβάνεται όσο έχουμε σειρές να διαβάσουμε στην βάση. Αυτό που κάνει η rs.next() είναι να διαβάζει κάθε φορά μία σειρά από τις στήλες που έχουμε επιλέξει με το ερώτημα που κάναμε και να αποθηκεύει τα στοιχεία της κάθε σειράς στην rs. Επίσης όσο υπάρχουν σειρές για να διαβάσει επιστρέφει την τιμή true. Εάν κάποιο από τα αποτελέσματα ταιριάζει με τις παραμέτρους που έχουμε περάσει στην μέθοδο τότε διακόπτετε η λειτουργία της μεθόδου και επιστρέφει την τιμή true και κάθε φορά που κάνει μία σύγκριση αυξάνεται το id για να βρούμε το id του χρήστη που έχει συνδεθεί. Στην περίπτωση που ο χρήστης δεν υπαρχή στην βάση δεδομένων τότε επιστρέφει την τιμή false. Παρακάτω φαίνεται το διάγραμμα ροής της μεθόδου:



Μία άλλη μέθοδος που χρησιμοποιούμε στην κλάση Client για να βρούμε αν ο χρήστης έχει δικαιώματα διαχειριστή είναι η findIfAdmin(String username, String password). Αυτή η κλάση μοιάζει πολύ με την findUser() με την μόνη διαφορά να είναι στο ότι επιλέγουμε και την στήλη Rights από τον πίνακα users μαζί με τις άλλες δύο. Στην βάση δεδομένων οι χρήστες που έχουν δικαιώματα διαχειριστή το αντίστοιχο πεδίο στην στήλη Rights έχει την τιμή admin ενώ οι υπόλοιποι έχουν την τιμή noAdmin. Έτσι όταν γίνεται η σύγκριση μπορούμε να βρούμε τα δικαιώματα του χρήστη. Η μέθοδος και το διάγραμμα ροής φαίνονται παρακάτω:

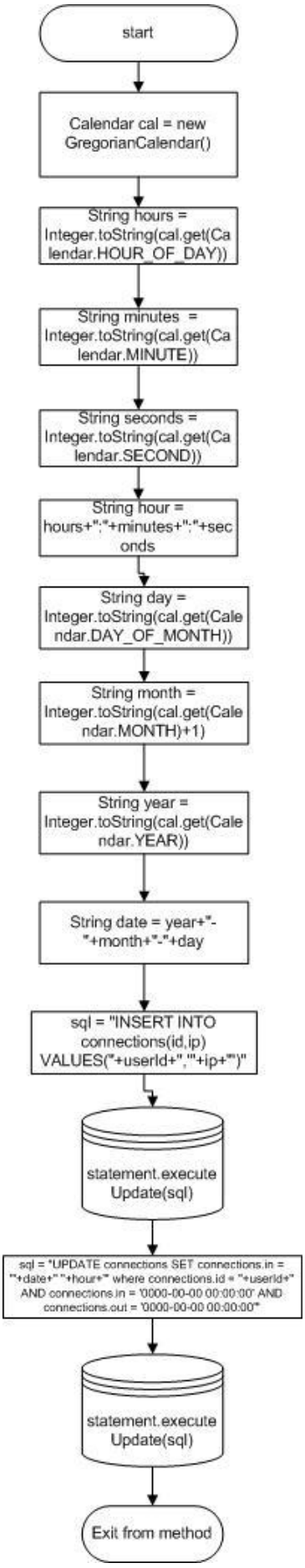
```
public boolean findIfAdmin(String username, String password){
        ResultSet rs = null;
                sql = "select Username , Password , Rights from users";
```

```
                try {
                        rs = statement.executeQuery(sql);
                   while(rs.next()){
                                if(rs.getString(1).equals(username) &&
rs.getString(2).equals(password) && rs.getString(3).equals("admin")){
                                        return true;
                                }
                   }
                } catch (SQLException e) {
                        e.printStackTrace();
                }
                return false;
```


start
ResultSet rs = null
sql = "select Username , Password , Rights from users"
rs = statement.executeQuery (sql)
rs.next()
true
false
rs.getString(1).equals(username) && rs.getString(2).equals(password) && rs.getString(3).equals("admin")
true
return true
false
return false
Exit from method

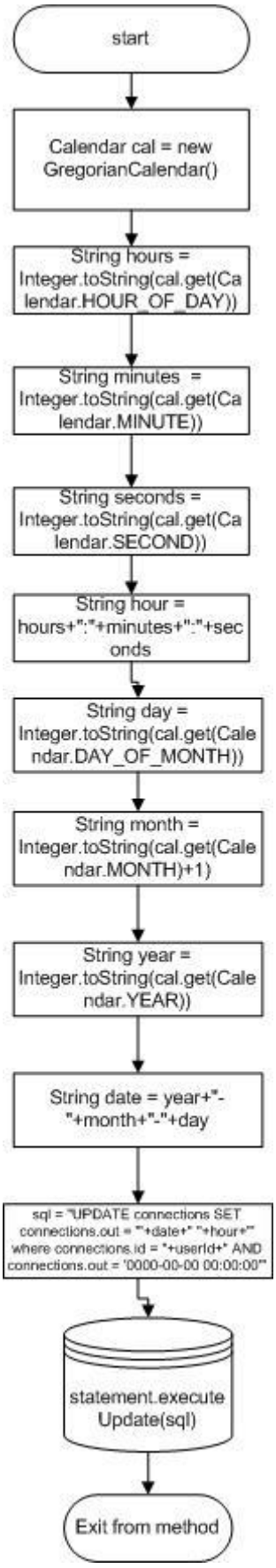
Την μέθοδο userLogIn(int userId, String ip) την χρησιμοποιούμε για να καταγράψουμε την ώρα, την ημερομηνία και την ip που συνδέθηκε ο κάθε χρήστης. Σαν παραμέτρους περνάμε το id του χρήστη και την ip του χρήστη. Αρχικά φτιάχνουμε ένα ημερολόγιο με όνομα cal. Έπειτα φτιάχνουμε την ώρα και την ημερομηνία στο πρότυπο timestamp της mysql το οποίο είναι yy-mm-dd hour:minutes:seconds για να το αποθηκεύσουμε στην βάση δεδομένων. Στην συνέχεια θέτουμε ένα ερώτημα στην βάση δεδομένων το οποίο είναι να εισάγουμε τιμές στα πεδία id και ip. Τέλος καταγράφουμε το timestamp στον πίνακα connections στην στήλη in. Παρακάτω φαίνεται η μέθοδος και το διάγραμμα ροής της μεθόδου:

```
	public void userLogIn(int userId, String ip){
		Calendar cal = new GregorianCalendar();
		String hours = Integer.toString(cal.get(Calendar.HOUR_OF_DAY));
		 String minutes  = Integer.toString(cal.get(Calendar.MINUTE));
		 String seconds = Integer.toString(cal.get(Calendar.SECOND));
		 String hour = hours+":"+minutes+":"+seconds;
		 String day = Integer.toString(cal.get(Calendar.DAY_OF_MONTH));
		 String month = Integer.toString(cal.get(Calendar.MONTH)+1);
		 String year = Integer.toString(cal.get(Calendar.YEAR));
		 String date = year+"-"+month+"-"+day;
		 sql = "INSERT INTO connections(id,ip) VALUES("+userId+",'"+ip+"')";
		try {
			statement.executeUpdate(sql);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		 sql = "UPDATE connections SET connections.in = '"+date+" "+hour+"' where
connections.id = "+userId+" AND connections.in = '0000-00-00 00:00:00' AND
connections.out = '0000-00-00 00:00:00'";
		 try {
				statement.executeUpdate(sql);
			} catch (SQLException e) {
				e.printStackTrace();
			}
	}
```


start
Calendar cal = new GregorianCalendar()
String hours = Integer.toString(cal.get(Calendar.HOUR_OF_DAY))
String minutes = Integer.toString(cal.get(Calendar.MINUTE))
String seconds = Integer.toString(cal.get(Calendar.SECOND))
String hour = hours+":"+minutes+":"+seconds
String day = Integer.toString(cal.get(Calendar.DAY_OF_MONTH))
String month = Integer.toString(cal.get(Calendar.MONTH)+1)
String year = Integer.toString(cal.get(Calendar.YEAR))
String date = year+"-"+month+"-"+day
sql = "INSERT INTO connections(id,ip) VALUES("+userId+",'"+ip+"')"
statement.executeUpdate(sql)
sql = "UPDATE connections SET connections.in = '"+date+" "+hour+"' where connections.id = "+userId+" AND connections.in = '0000-00-00 00:00:00' AND connections.out = '0000-00-00 00:00:00'"
statement.executeUpdate(sql)
Exit from method

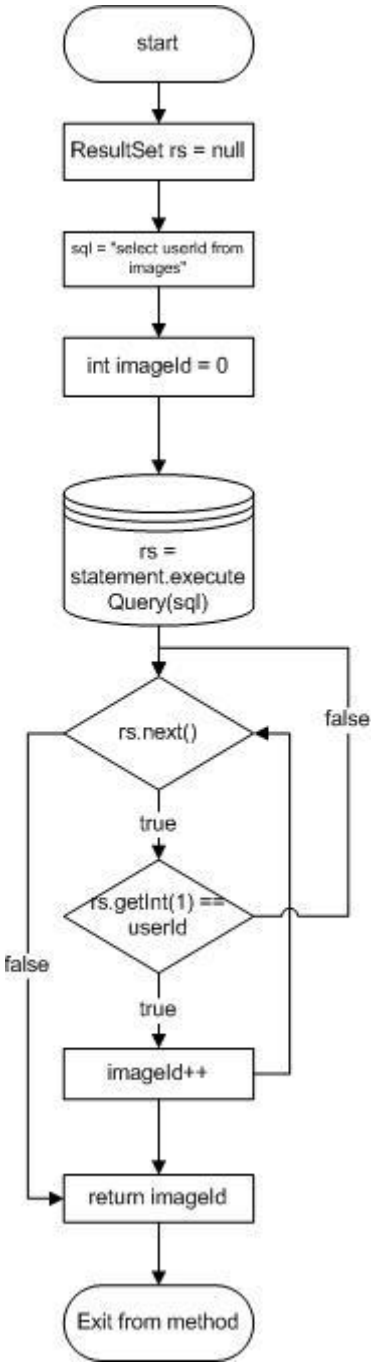
Αντίστοιχα την userLogOut(int userId) την χρησιμοποιούμε για να καταγράψουμε την ώρα και την ημερομηνία που αποσυνδέεται από τον Server. Η μέθοδος είναι ίδια με την userLogIn(int userId, String ip) με δύο διαφορές. Η πρώτη είναι ότι χρησιμοποιούμε σαν παράμετρο μόνο το id του χρήστη, και η δεύτερη εισάγουμε με την sql = "INSERT INTO sql = "UPDATE connections SET connections.out = '"+date+" "+hour+"' where connections.id = "+userId+" AND connections.out = '0000-00-00 00:00:00'" το timestamp στον πίνακα connections στην στήλη out που αντίστοιχή στο user id που έχουμε περάσει σαν παράμετρο στην μέθοδο. Παρακάτω φαίνονται η μέθοδος και το διάγραμμα ροής:

```
public void userLogOut(int userId){
                Calendar cal = new GregorianCalendar();
                String hours = Integer.toString(cal.get(Calendar.HOUR_OF_DAY));
                 String minutes  = Integer.toString(cal.get(Calendar.MINUTE));
                 String seconds = Integer.toString(cal.get(Calendar.SECOND));
                 String hour = hours+":"+minutes+":"+seconds;
                 String day = Integer.toString(cal.get(Calendar.DAY_OF_MONTH));
                 String month = Integer.toString(cal.get(Calendar.MONTH)+1);
                 String year = Integer.toString(cal.get(Calendar.YEAR));
                 String date = year+"-"+month+"-"+day;
                 sql = "UPDATE connections SET connections.out = '"+date+" "+hour+"'
where connections.id = "+userId+" AND connections.out = '0000-00-00 00:00:00'";
                try {
                        statement.executeUpdate(sql);
                } catch (SQLException e) {
                        e.printStackTrace();
                }
        }
```


start
Calendar cal = new GregorianCalendar()
String hours = Integer.toString(cal.get(Calendar.HOUR_OF_DAY))
String minutes = Integer.toString(cal.get(Calendar.MINUTE))
String seconds = Integer.toString(cal.get(Calendar.SECOND))
String hour = hours+":"+minutes+":"+seconds
String day = Integer.toString(cal.get(Calendar.DAY_OF_MONTH))
String month = Integer.toString(cal.get(Calendar.MONTH)+1)
String year = Integer.toString(cal.get(Calendar.YEAR))
String date = year+"-"+month+"-"+day
sql = "UPDATE connections SET connections.out = "'+date+' '+hour+'" where connections.id = "+userId+" AND connections.out = '0000-00-00 00:00:00'"
statement.executeUpdate(sql)
Exit from method

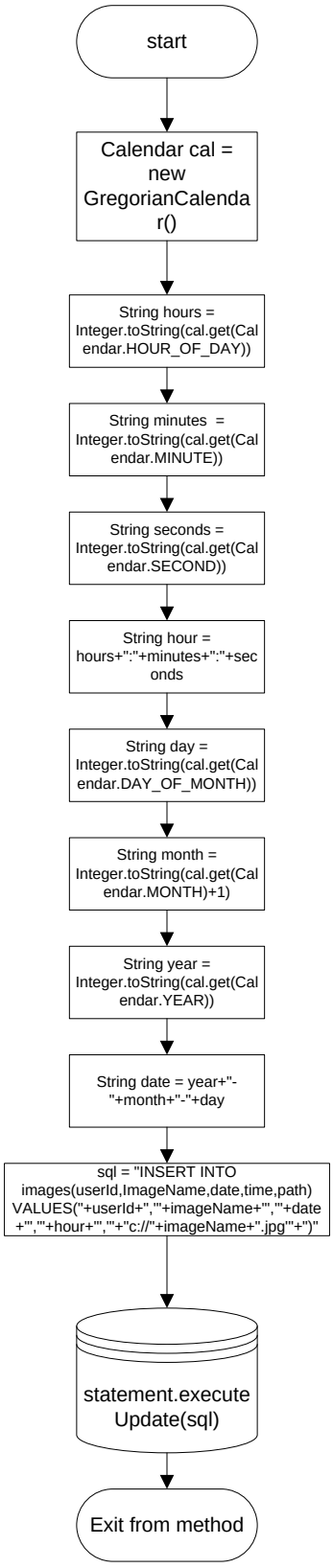
Την μέθοδο findImageId(int userId) την χρησιμοποιούμε για να βρούμε το πλήθος των εικόνων που έχει καταγράψει ο χρήστης που έχουμε περάσει σαν παράμετρο στην μέθοδο. Αρχικά ορίζουμε μία μεταβλητή με όνομα rs η οποία είναι τύπου ResultSet. Στην συνέχεια με την sql = "select userId from images" διαλέγουμε την στήλη userId από τον πίνακα images και δηλώνουμε μια μεταβλητή imageId = 0. Έπειτα εκτελούμε την ερώτηση στην βάση δεδομένων με την rs = statement.executeQuery(sql). Στην συνέχεια μπαίνουμε σε ένα βρόχο while. Μέσα στον βρόχο εάν είναι έγκυρη η συνθήκη rs.getInt(1) == userId αυξάνουμε την μεταβλητή κατά ένα. Όταν δεν θα υπάρχουν άλλα στοιχεία για να διαβάσει το πρόγραμμα τότε επιστρέφει την τιμή της μεταβλητής imageId όπου περιέχει το πλήθος των εικόνων. Η μέθοδος και το διάγραμμα ροής φαίνονται παρακάτω:

```
public int findImageId(int userId){
		ResultSet rs = null;
		sql = "select userId from images";
		int imageId = 0;
		try {
			rs = statement.executeQuery(sql);
			while(rs.next()){
				if(rs.getInt(1) == userId){
					imageId++;
				}
			}
		} catch (SQLException e) {
			e.printStackTrace();
		}
		return imageId;
	}
```


start
ResultSet rs = null
sql = "select userId from images"
int imageId = 0
rs = statement.executeQuery(sql)
rs.next()
false
true
rs.getInt(1) == userId
false
true
imageId++
return imageId
Exit from method

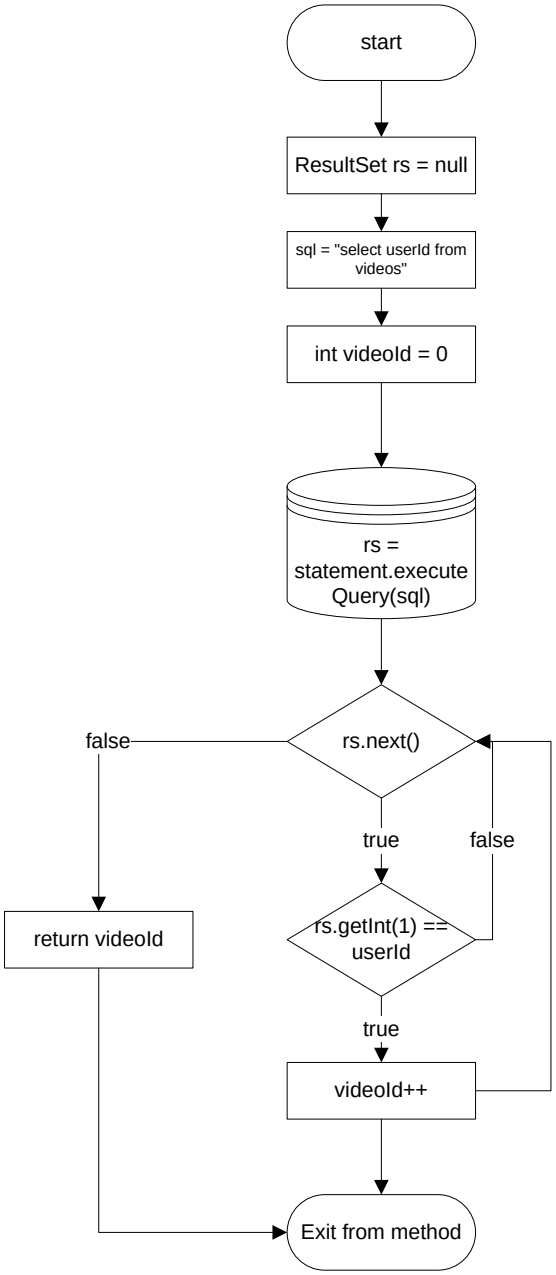
Εάν κάποιος χρήστης έχει καταγράψει μία εικόνα τότε για να αποθηκεύσουμε ποιος την έχει καταγράψει, ποιά χρονική στιγμή, με ποιο όνομα και σε ποιόν φάκελο στον σκληρό δίσκο τότε χρησιμοποιούμαι την μέθοδο storeImageDB(int userId, String imageName). Σαν παράμετρο περνάμε το userId και το όνομα της εικόνας. Στην αρχή της μεθόδου φτιάχνουμε το timestamp όπως και στις προηγούμενες μεθόδους, έπειτα με την sql = "INSERT INTO images(userId,ImageName,date,time,path) VALUES("+userId+",'"+imageName+"','"+date+"','"+hour+"','"+"c://"+imageName+".jpg'"+")" εισάγουμε τις τιμές: όνομα χρήστη, όνομα εικόνας, ημερομηνία και ώρα, διαδρομή στον δίσκο που είναι αποθηκευμένη η εικόνα στις αντίστοιχα πεδία στην βάση δεδομένων. Τέλος με την statement.executeUpdate(sql) εκτελούμε την εντολή sql. Η μέθοδος storeImageDB(int userId, String imageName) και το διάγραμμα ροής φαίνονται παρακάτω:

```
public void storeImageDB(int userId, String imageName){
                Calendar cal = new GregorianCalendar();
                String hours = Integer.toString(cal.get(Calendar.HOUR_OF_DAY));
                 String minutes  = Integer.toString(cal.get(Calendar.MINUTE));
                 String seconds = Integer.toString(cal.get(Calendar.SECOND));
                 String hour = hours+":"+minutes+":"+seconds;
                 String day = Integer.toString(cal.get(Calendar.DAY_OF_MONTH));
                 String month = Integer.toString(cal.get(Calendar.MONTH)+1);
                 String year = Integer.toString(cal.get(Calendar.YEAR));
                 String date = year+"-"+month+"-"+day;
                sql = "INSERT INTO images(userId,ImageName,date,time,path) 
VALUES("+userId+",'"+imageName+"','"+date+"','"+hour+"','"+"c://"+imageName+".jpg'"+"
)";
                try {
                        statement.executeUpdate(sql);
                } catch (SQLException e) {
                        e.printStackTrace();
                }
        }
```


start
Calendar cal = new GregorianCalendar()
String hours = Integer.toString(cal.get(Calendar.HOUR_OF_DAY))
String minutes = Integer.toString(cal.get(Calendar.MINUTE))
String seconds = Integer.toString(cal.get(Calendar.SECOND))
String hour = hours+":"+minutes+":"+seconds
String day = Integer.toString(cal.get(Calendar.DAY_OF_MONTH))
String month = Integer.toString(cal.get(Calendar.MONTH)+1)
String year = Integer.toString(cal.get(Calendar.YEAR))
String date = year+"-"+month+"-"+day
sql = "INSERT INTO images(userId,ImageName,date,time,path) VALUES("+userId+",'"+imageName+"','"+date+"','"+hour+"','"+"c://"+imageName+".jpg'"+")"
statement.executeUpdate(sql)
Exit from method

Για να βρούμε το id του βίντεο που καταγράφει ένας χρήστης μία συγκεκριμένη στιγμή χρησιμοποιούμε την μέθοδο findVideoId(int userId). Αρχικά ορίζουμε μία μεταβλητή με όνομα rs και τύπου ResultSet και δηλώνουμε την εντολή sql="select userId from videos" την οποία θα την εκτελέσουμε με την rs = statement.executeQuery(sql) και τα αποτελέσματα αποθηκεύονται στην rs. Με την εντολή sql που δηλώσαμε παραπάνω επιλέγουμε την στήλη userId από τον πίνακα videos. Έπειτα μπαίνει σε ένα βρόχο while ο οποίος έχει ακριβώς την ίδια λειτουργία με τον αντίστοιχο βρόχο στην μέθοδο findVideoId(int userId) με την μόνη διαφορά να είναι ότι τώρα αυξάνουμε την μεταβλητή videoId που ορίσαμε πριν τον βρόχο. Τέλος η μέθοδος επιστρέφει το πλήθος των βίντεο που έχει καταγράψει ο χρήστης έτσι ώστε στην κλάση Client να το αυξήσουμε κατά ένα για να βρούμε το Id του νέου βίντεο. Η μέθοδος μαζί με το διάγραμμα ροής φαίνονται παρακάτω:

```
public int findVideoId(int userId){
        ResultSet rs = null;
        sql = "select userId from videos";
        int videoId = 0;
        try {
                rs = statement.executeQuery(sql);
                while(rs.next()){
                        if(rs.getInt(1) == userId){
                                videoId++;
                        }
                }
        } catch (SQLException e) {
                e.printStackTrace();
        }
        return videoId;
}
```

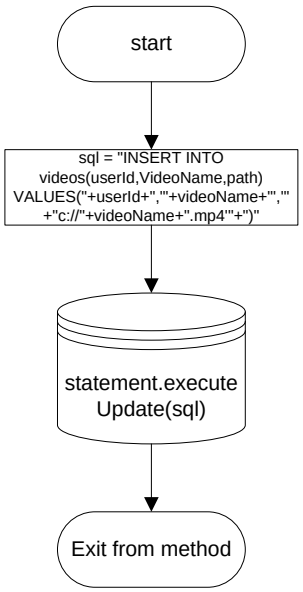


Για να καταγράψουμε στη βάση δεδομένων το όνομα, την διάρκεια και τον φάκελο στον οποίο αποθηκεύτηκε ένα βίντεο όταν εγγραφή του από έναν χρήστη γίνεται σε πραγματικό χρόνο χρησιμοποιούμε τις έξης δύο μεθόδους: storeVideoDBStart(int userId, String videoName) και storeVideoDBStop(long duration, int id). Όπως είδαμε στην κλάση Client την πρώτη μέθοδο την καλούμε όταν αρχίζει η εγγραφή του βίντεο στην οποία περνάμε σαν παραμέτρους το id του χρήστη που κάνει την εγγραφή και το όνομα του βίντεο. Η μέθοδος storeVideoDBStart(int userId, String videoName) φαίνεται παρακάτω:

```
public void storeVideoDBStart(int userId, String videoName){
		sql = "INSERT INTO videos(userId,VideoName,path)
VALUES("+userId+",'"+videoName+"','"+"c://"+videoName+".mp4'"+")";
		try {
			statement.executeUpdate(sql);
		} catch (SQLException e) {
			e.printStackTrace();
		}
```

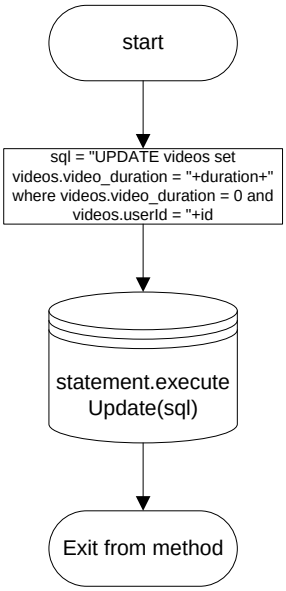
```
	}
```

Με την εντολή sql = "INSERT INTO videos(userId,VideoName,path) VALUES("+userId+",'"+videoName+"','"+"c://"+videoName+".mp4'"+")" λέμε στο πρόγραμμα να εισάγει στον πίνακα videos και στις στήλες userId,VideoName,path τις αντίστοιχες τιμές. Τέλος με την statement.executeUpdate(sql) εκτελούμε αυτή την εντολή. Παρακάτω φαίνεται το διάγραμμα ροής της μεθόδου:



Την μέθοδο storeVideoDBStop(long duration, int id) την χρησιμοποιούμε όταν ο χρήστης διακόψει την εγγραφή του βίντεο. Στην μέθοδο αυτή περνάμε την διάρκεια του βίντεο και το id του χρήστη όπως είδαμε στην κλάση Client. Με την εντολή sql = "UPDATE videos set videos.video_duration = "+duration+" where videos.video_duration = 0 and videos.userId = "+id λέμε στο πρόγραμμα να ενημερώσει με την διάρκεια του βίντεο το πεδίο video_duration με τιμή 0 που αντιστοιχεί στο id του χρήστη. Η μέθοδος με το διάγραμμα ροής φαίνονται παρακάτω:

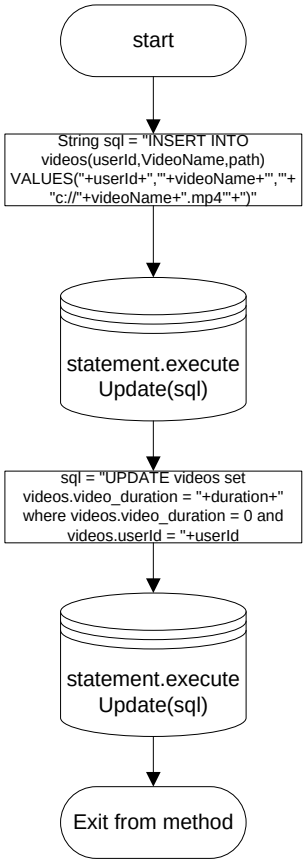
```
public void storeVideoDBStop(long duration, int id){
		sql = "UPDATE videos set videos.video_duration = "+duration+" where videos.video_duration = 0 and videos.userId = "+id;
		try {
			statement.executeUpdate(sql);
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}
```



Η μέθοδος storeVideoDB(String videoName, int userId, long duration) φαίνεται παρακάτω:

```
public static void storeVideoDB(String videoName, int userId, long duration){

		String sql = "INSERT INTO videos(userId,VideoName,path) 
VALUES("+userId+",'"+videoName+"','"+"c://"+videoName+".mp4'"+")";
		try {
			statement.executeUpdate(sql);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		sql = "UPDATE videos set videos.video_duration = "+duration+" where 
videos.video_duration = 0 and videos.userId = "+userId;
		try {
			statement.executeUpdate(sql);
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}
```

Εάν μία εγγραφή βίντεο είναι χρονοπρογραμματισμένη τότε για να καταγράψουμε το id του χρήστη, το όνομα και την διάρκεια του βίντεο στη βάση δεδομένων χρησιμοποιούμε την παραπάνω μέθοδο. Η λειτουργία της συγκεκριμένης μεθόδου είναι ίδια με τις λειτουργίες των δύο προηγούμενων μαζί. Η μόνη διαφορά είναι ότι η storeVideoDB(String videoName, int userId, long duration) είναι τύπου static δηλαδή δεν χρειάζεται να την καλέσουμε χρησιμοποιώντας το αντικείμενο της κλάσης DataBase. Όπως είχαμε αναφέρει έχουμε φτιάξει στη κλάση Client το αντικείμενο της κλάσης DataBase για να μπορούμε να καλέσουμε τις μεθόδους της. Η παραπάνω μέθοδος χρησιμοποιείτε μόνο από τη κλάση Schedule, άρα φτιάχνοντας άλλο ένα αντικείμενο της DataBase στην Schedule θα είχαμε συνολικά δύο στιγμιότυπα της DataBase στο πρόγραμμα με αποτέλεσμα τη σύγχυση του προγράμματος. Το διάγραμμα ροής φαίνεται παρακάτω:



Την μέθοδο ScheduleRec(String startDateTime, String endDateTime, String username, String videoName, int userId) την χρησιμοποιούμε για να καταγράψουμε στη βάση δεδομένων μία χρονοπρογραμματισμένη εγγραφή βίντεο, δηλαδή τις ημερομηνίες και ώρες έναρξης και διακοπής της εγγραφής. Αρχικά για την καλύτερη κατανόηση της μεθόδου θα πρέπει να εξηγήσουμε ορισμένες κλάσης της βιβλιοθήκης Quartz όπως η Scheduler, Job, JobDetail, Trigger, JobBuilder και TriggerBuilder. Η Scheduler είναι το κύριο API για την αλληλεπίδραση με το χρονοπρογραμματιστή. Ο κύκλος ζωής ενός χρονοπρογραμματιστή είναι συνδεδεμένος από την δημιουργία του με τον SchedulerFactory και μέχρι να καλέσουμε την μέθοδο shutdown(). Με την Scheduler μπορούμε να προσθέσουμε και να αφαιρέσουμε Jobs και Triggers. Job είναι η εργασία που θέλουμε να εκτελέσουμε από τον χρονοπρογραμματιστή δηλαδή η εγγραφή βίντεο. Η JobDetail χρησιμοποιείτε για να καθορίσουμε τις λεπτομέρειες κάθε εργασίας όπως το ποιά κλάση θα εκτέλεση την εργασία, την ταυτότητα της εργασίας που στη περίπτωσή μας είναι ο αριθμός της εργασίας ή αλλιώς το jobkey και το όνομα του χρήστη που θέλει να κάνει την εγγραφή και κάποια άλλα στοιχεία όπως το όνομα του βίντεο, ο αριθμός του βίντεο και η ημερομηνία και η ώρα λήξεις της εγγραφής. Η Trigger είναι μία κλάση η οποία καθορίζει στον χρονοπρογραμματιστή πότε θα εκτελεστεί μία εργασία. Η JobBuilder χρησιμοποιείτε για να δημιουργήσουμε στιγμιότυπα της JobDetail που με την σειρά τους δημιουργούν στιγμιότυπα των Jobs. Η TriggerBuilder χρησιμοποιείτε για την δημιουργία στιγμιότυπων της Trigger. Στην βάση δεδομένων που καταγράφουμε τις εργασίες δεν θα πρέπει οι εργασίες να έχουν το ίδιο jobkey και τα triggers το ίδιο triggerkey. Για να βρούμε τον αριθμό των triggerkey καλούμε την μέθοδο findTriggersName() όταν θέτουμε τον Server σε λειτουργία όπως είδαμε στην κλάση Main. Ο αριθμός των triggerkey αποθηκεύετε στην μεταβλητή triggerKey. Η μέθοδος ScheduleRec(String startDateTime, String endDateTime, String username, String videoName, int userId) φαίνεται παρακάτω:

```
public void ScheduleRec(String startDateTime, String endDateTime, String username, String
videoName, int userId){
		findJobKey(username);
		key++;
		jobkey = "jobKey"+String.valueOf(key);
		triggerKey++;
		triggerName = "trigger" + triggerKey;
		JobDetail job = JobBuilder.newJob(Schedule.class)
				.withIdentity(jobkey,username)
				.usingJobData("Name", videoName)
				.usingJobData("userId", userId)
			.usingJobData("Date", endDateTime)// 2013/03/03 18:31:00
			.storeDurably()
				.build();
		 trigger = newTrigger()
		   .withIdentity(triggerName, username)
		   .withSchedule(cronSchedule(startDateTime)
		   .withMisfireHandlingInstructionDoNothing())
		   .build();
		try {
			scheduler.scheduleJob(job,trigger);
		} catch (SchedulerException e1) {
			e1.printStackTrace();
		}
	}
```

Στην αρχή της μεθόδου με την findJobKey(username) ψάχνουμε τον αριθμό των εγγραφών που έχει χρονοπρογραμματίση ο χρήστης που έχουμε πέραση σαν παράμετρο (οι μέθοδοι findJobKey(username) και findTriggersName() θα εξηγηθούν παρακάτω). Στην συνέχεια αυξάνουμε το key με την key++ κατά ένα για να βρούμε τον αριθμό της νέας εργασίας και με την jobkey = "jobKey"+String.valueOf(key) δημιουργούμε το jobkey. Έπειτα αυξάνουμε την μεταβλητή triggerKey++ και με την triggerName = "trigger" + triggerKey ορίζουμε το triggerName το οποίο αποτελείται από την λέξη trigger και το triggerKey. Για να δημιουργήσουμε μία νέα εργασία χρησιμοποιούμε την: JobDetail job = JobBuilder.newJob(Schedule.class)

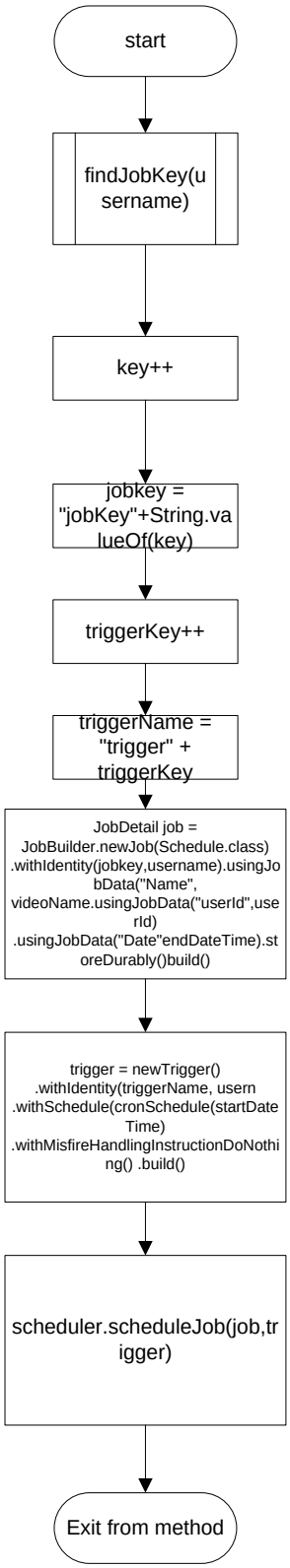
```
				.withIdentity(jobkey,username)
				.usingJobData("Name", videoName)
				.usingJobData("userId", userId)
			.usingJobData("Date", endDateTime)// 2013/03/03 18:31:00
			.storeDurably()
				.build();
```

Κάθε φορά που θα ορίζουμε μία νέα εργασία θα δημιουργείτε ένα νέο στιγμιότυπο της κλάσης Schedule. Με την .withIdentity(jobkey,username) ορίζουμε την ταυτότητα της νέας εργασίας η οποία θα ανήκει στο group με το όνομα του χρήστη που την δημιούργησε και με αριθμό το jobkey. Όπως αναφέραμε για να περάσουμε κάποια στοιχεία στο αντίστοιχο στιγμιότυπο που δημιουργείτε κάθε φορά χρησιμοποιούμε την μέθοδο .usingJobData() περνώντας τα στοιχεία σαν παραμέτρους στην μέθοδο. Με την μέθοδο .storeDurably() το πρόγραμμα κρατάει αποθηκευμένη στη βάση δεδομένων την εργασία και μετά την εκτέλεση της και με την .build() δημιουργούμε την νέα εργασία. Με την trigger = newTrigger()

```
		.withIdentity(triggerName, username)
```

```
.withSchedule(cronSchedule(startDateTime)
.withMisfireHandlingInstructionDoNothing())
.build();
```

ορίζουμε ένα νέο trigger με ταυτότητα η οποία ανήκει στο group με το όνομα του χρήστη και αριθμό όνομα το triggerName. Με την μέθοδο .withSchedule(cronSchedule(startDateTime) ορίζουμε την χρονική στιγμή που θα εκτελεστεί η εργασία. Η χρονική στηγμή είναι στα των cron expressions τα οποία θα αναλύσουμε στο κεφάλαιο που αφορά το applet λόγο ότι εκεί δημιουργείται η συγκεκριμένη έκφραση. Εάν για κάποιο λόγο δεν ήταν δυνατή η εκτέλεση της εργασίας που είχε προγραμματιστεί τότε με την μέθοδο .withMisfireHandlingInstructionDoNothing() το πρόγραμμα την παραβλέπει και εκτελεί την επόμενη χρονικά μέθοδο. Η .build() εκτελεί την ίδια ακριβώς λειτουργία με την προηγούμενη. Τέλος με την scheduler.scheduleJob(job,trigger) προγραμματίζουμε την εργασία. Το διάγραμμα ροής της μεθόδου φαίνεται παρακάτω:


start
findJobKey(username)
key++
jobkey = "jobKey"+String.valueOf(key)
triggerKey++
triggerName = "trigger" + triggerKey
JobDetail job = JobBuilder.newJob(Schedule.class) .withIdentity(jobkey,username).usingJobData("Name", videoName.usingJobData("userId",userId) .usingJobData("Date"endDateTime).storeDurably()build()
trigger = newTrigger() .withIdentity(triggerName, usern .withSchedule(cronSchedule(startDateTime) .withMisfireHandlingInstructionDoNothing() .build()
scheduler.scheduleJob(job,trigger)
Exit from method

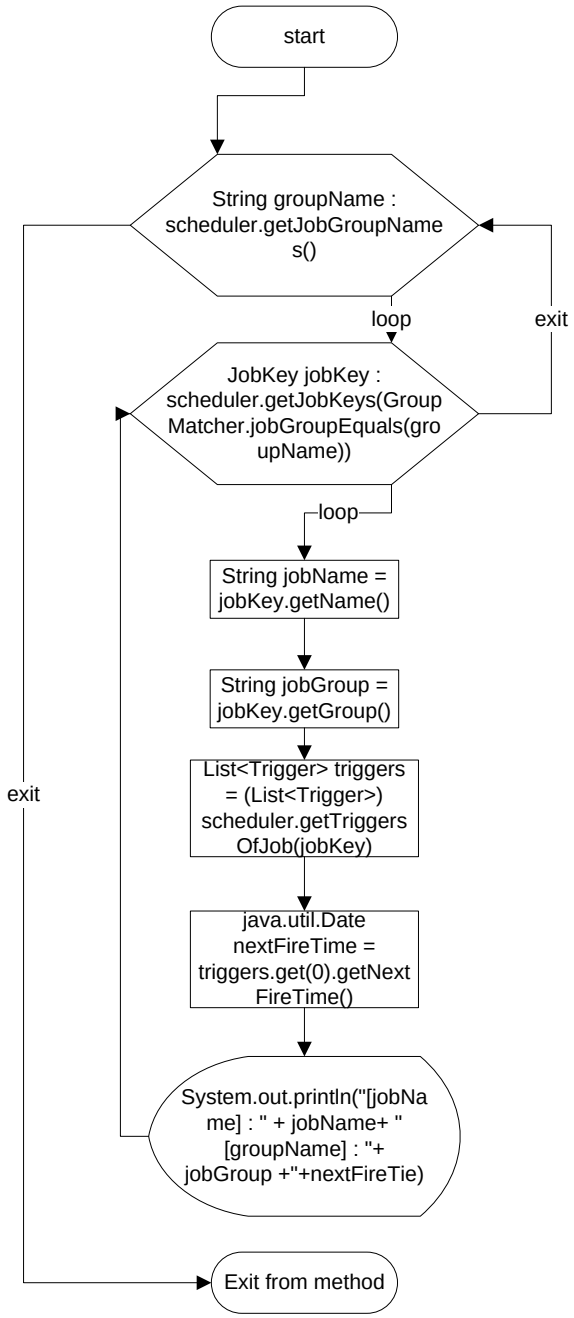
Εάν θέλουμε να διαγράψουμε όλες τις προγραμματισμένες εργασίες τότε πατάμε το κουμπί Clear Scheduler στο GUI του Server και έτσι καλούμε την μέθοδο clearScheduler(). Μέσα στην μέθοδο καλούμε την clear() από το στιγμιότυπο scheduler η μεθοδος φαίνεται παρακάτω:

```
public void clearScheduler(){
                try {
                        scheduler.clear();
                } catch (SchedulerException e1) {
                        e1.printStackTrace();
                }
        }
```

Εάν ο χρήστης θέλει να εμφανίσει όλες τις προγραμματισμένες εγγραφές βίντεο τότε θα πρέπει να πατήσει το κουμπί Display Jobs στο GUI του Server. Πατώντας το συγκεκριμένο κουμπί καλούμε την μέθοδο displayJobs(). Η μέθοδος φαίνεται παρακάτω:

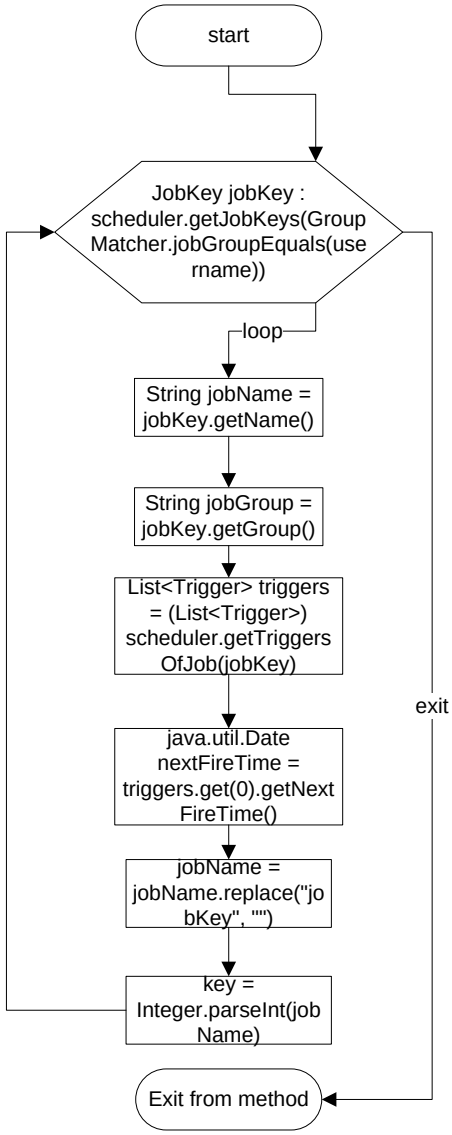
```
public void displayJobs(){
                 try {
for (String groupName : scheduler.getJobGroupNames()) {
for (JobKey jobKey : scheduler.getJobKeys(GroupMatcher.jobGroupEquals(groupName))) {
String jobName = jobKey.getName();
String jobGroup = jobKey.getGroup();
 //get job's trigger
List<Trigger> triggers = (List<Trigger>) scheduler.getTriggersOfJob(jobKey);
java.util.Date nextFireTime = triggers.get(0).getNextFireTime();
System.out.println("[jobName] : " + jobName + " [groupName] : "+ jobGroup + " - " +
nextFireTime); }
                                }
                        } catch (SchedulerException e) {
                                e.printStackTrace();}
}
```

Η συγκεκριμένη μέθοδος περιέχει δύο βρόχους for. Με τον πρώτο βρόχο for βρίσκουμε κάθε φορά το όνομα ενός group και με τον δεύτερο βρόχο for ψάχνουμε να βρούμε πόσες εργασίες περιέχει το συγκεκριμένο group. Έπειτα αποθηκεύουμε το όνομα της εργασίας και το group στο οποίο ανήκει η εργασία στις μεταβλητές jobName και jobGroup αντίστοιχα. Με την List<Trigger> triggers = (List<Trigger>) scheduler.getTriggersOfJob(jobKey) αποθηκεύουμε σε μία λίστα με το όνομα triggers το trigger κάθε εργασίας. Τέλος αποθηκεύουμε την χρονική στιγμή που θα εκτελεστεί η κάθε εργασία με την java.util.Date nextFireTime = triggers.get(0).getNextFireTime() και εκτυπώνουμε με την System.out.println("[jobName] : " + jobName + " [groupName] : "+ jobGroup + " - " + nextFireTime) το όνομα της εργασίας, το group στο οποίο ανήκει και την χρονική στιγμή που θα εκτελεστεί. Το διάγραμμα ροής φαίνεται παρακάτω:


start
String groupName : scheduler.getJobGroupNames()
loop
exit
JobKey jobKey : scheduler.getJobKeys(GroupMatcher.jobGroupEquals(groupName))
loop
String jobName = jobKey.getName()
String jobGroup = jobKey.getGroup()
List<Trigger> triggers = (List<Trigger>) scheduler.getTriggersOfJob(jobKey)
java.util.Date nextFireTime = triggers.get(0).getNextFireTime()
System.out.println("[jobName] : " + jobName+ " [groupName] : "+ jobGroup +"+nextFireTie)
exit
Exit from method

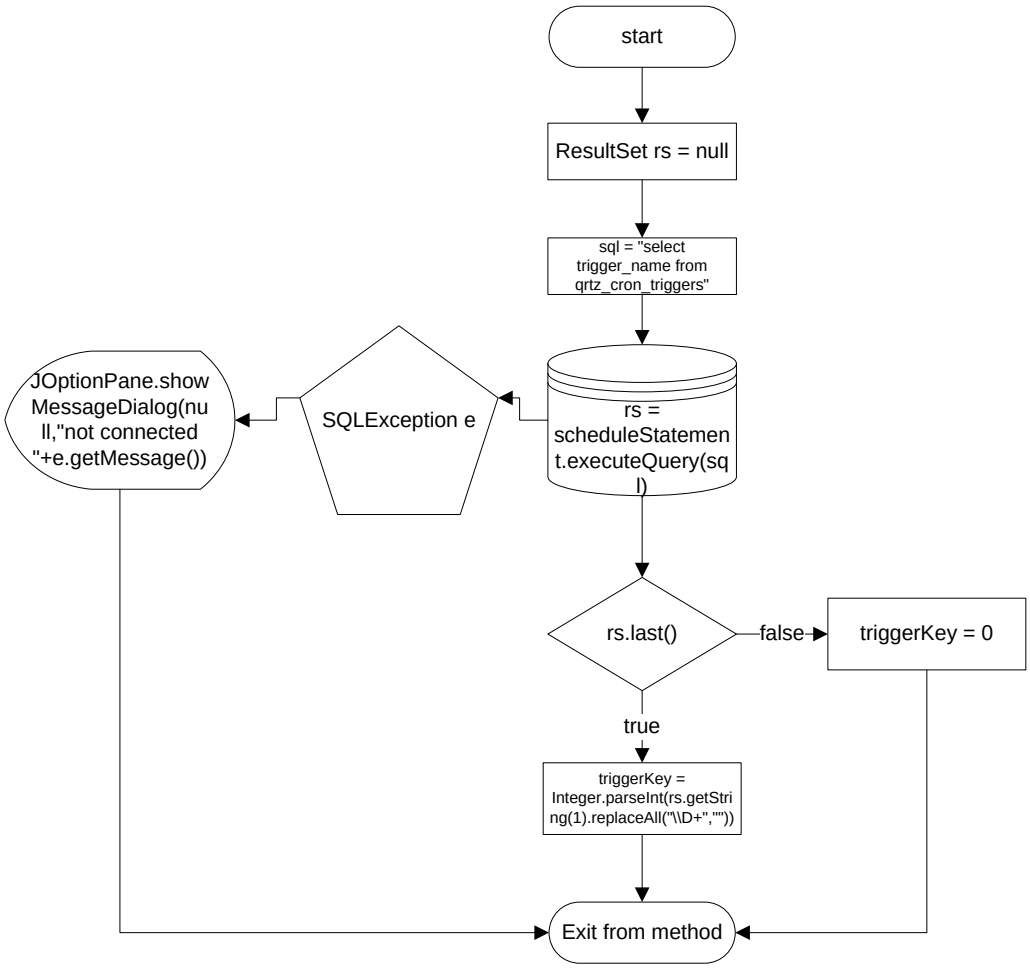
Με την findJobKey(String username) βρίσκουμε πόσες εργασίες είναι αποθηκευμένες σε ένα group. Με τον βρόχο for βρίσκουμε κάθε φορά την εργασία η οποία ανήκει στο group εργασιών που έχουμε περάσει σαν παράμετρο στην μέθοδο και έχει το όνομα του χρήστη. Αποθηκεύουμε το όνομα της εργασίας και το group στο οποίο ανήκει η εργασία στις μεταβλητές jobName και jobGroup αντίστοιχα. Με την List<Trigger> triggers = (List<Trigger>) scheduler.getTriggersOfJob(jobKey) αποθηκεύουμε σε μία λίστα με το όνομα triggers το trigger κάθε εργασίας. Τέλος αποθηκεύουμε την χρονική στιγμή που θα εκτελεστεί η κάθε εργασία με την java.util.Date nextFireTime = triggers.get(0).getNextFireTime(). Τέλος δημιουργούμε τον αριθμό key της νέας εργασίας με τις jobName = jobName.replace("jobKey", "") και key = Integer.parseInt(jobName). Η μέθοδος και το διάγραμμα ροής φαίνονται παρακάτω:

```
        public void findJobKey(String username){
                try {
for (JobKey jobKey : scheduler.getJobKeys(GroupMatcher.jobGroupEquals(username))) {
String jobName = jobKey.getName();
String jobGroup = jobKey.getGroup();
//get job's trigger
 List<Trigger> triggers = (List<Trigger>) scheduler.getTriggersOfJob(jobKey); java.util.Date
nextFireTime = triggers.get(0).getNextFireTime();
 jobName = jobName.replace("jobKey", "");
 key = Integer.parseInt(jobName);
 return; }
                        } catch (SchedulerException e) {
                                e.printStackTrace();
                        }
}
```



Με την findTriggersName() βρίσκουμε τον αριθμό των triggers που είναι αποθηκευμένα στη βάση δεδομένων. Ορίζουμε μια μεταβλητή με όνομα rs τύπου ResultSet και την εντολή sql = "select trigger_name from qrtz_cron_triggers". Με την συγκεκριμένη εντολή διαλέγουμε την στήλη trigger_name από τον πίνακα qrtz_cron_triggers. Έπειτα κάνουμε την ερώτηση στη βάση δεδομένων με την rs = scheduleStatement.executeQuery(sql) και τα αποτελέσματα αποθηκεύονται στην μεταβλητή rs. Εάν δεν είναι αποθηκευμένο κάποιο trigger στην βάση τότε η rs.last() επιστρέφει την τιμή false και η μεταβλητή triggerKey ισούται με μηδέν σε αντίθετη περίπτωση σβήνουμε το όνομα του trigger για να μας μείνει μόνο ο αριθμός και τον αποθηκεύουμε στη μεταβλητή triggerKey η παραπάνω διαδικασία γίνεται με την συγκεκριμένη εντολή triggerKey = Integer.parseInt(rs.getString(1).replaceAll("\\D+","")). Εάν για κάποιο λόγο η εφαρμογή δεν μπορέσει να συνδεθεί με την βάση δεδομένων εμφανίζεται ένα παράθυρο που περιέχει το πρόβλημα. Τέλος κλείνουμε την σύνδεση με την συγκεκριμένη βάση δεδομένων με την scheduleCon.close(). Η καταγραφή των προγραμματισμένων εγγραφών δεν χρειάζεται την συγκεκριμένη σύνδεση στην βάση δεδομένων αλλά γίνεται με την scheduler.scheduleJob(job,trigger). Η μέθοδος μαζί με το διάγραμμα ροής φαίνονται παρακάτω:

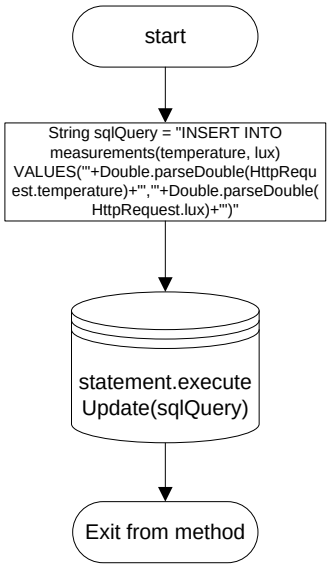
```
public void findTriggersName(){
                try {
                ResultSet rs = null;
                        sql = "select trigger_name from qrtz_cron_triggers";
                              rs = scheduleStatement.executeQuery(sql);
                              if(rs.last()){
                 triggerKey = Integer.parseInt(rs.getString(1).replaceAll("\\D+",""));
                              }else{
                                      triggerKey = 0;
                                      }
            }
            catch (SQLException e) {
               JOptionPane.showMessageDialog(null,"not connected "+e.getMessage());
            }
                try {
                        scheduleCon.close();
               } catch (SQLException e) {
                        e.printStackTrace();
               }
        }
```



Η μέθοδος storeMeasuresinDB() χρησιμοποιείτε από το πρόγραμμα για την καταγραφή της θερμοκρασίας και της φωτεινότητας από τον iLon. Με την String sqlQuery = "INSERT INTO measurements(temperature, lux)

VALUES('"+Double.parseDouble(HttpRequest.temperature)+"','"+Double.parseDouble(Http Request.lux)+"')" λέμε στο πρόγραμμα να αποθηκεύσει την θερμοκρασία και τα lux στον πίνακα measurements στα αντίστοιχα πεδία. Τις τιμές αυτές τις παίρνουμε από την κλάση HttpRequest και τις μετατρέπουμε από τύπου string σε double. Τέλος με την statement.executeUpdate(sqlQuery) κάνουμε το ερώτημα στην βάση δεδομένων. Η μέθοδος και το διάγραμμα ροής φαίνονται παρακάτω:

```
public static void storeMeasuresinDB(){
String sqlQuery = "INSERT INTO measurements(temperature, lux)
VALUES('"+Double.parseDouble(HttpRequest.temperature)+"','"+Double.parseDouble(Http
Request.lux)+"')";
                try {
                        statement.executeUpdate(sqlQuery);
                } catch (SQLException e) {e.printStackTrace();}
}
```



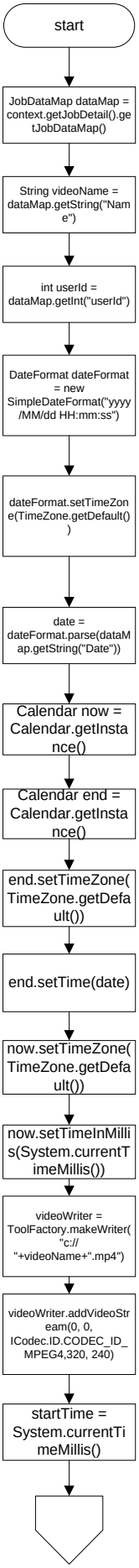
2.6 Η κλάση Schedule

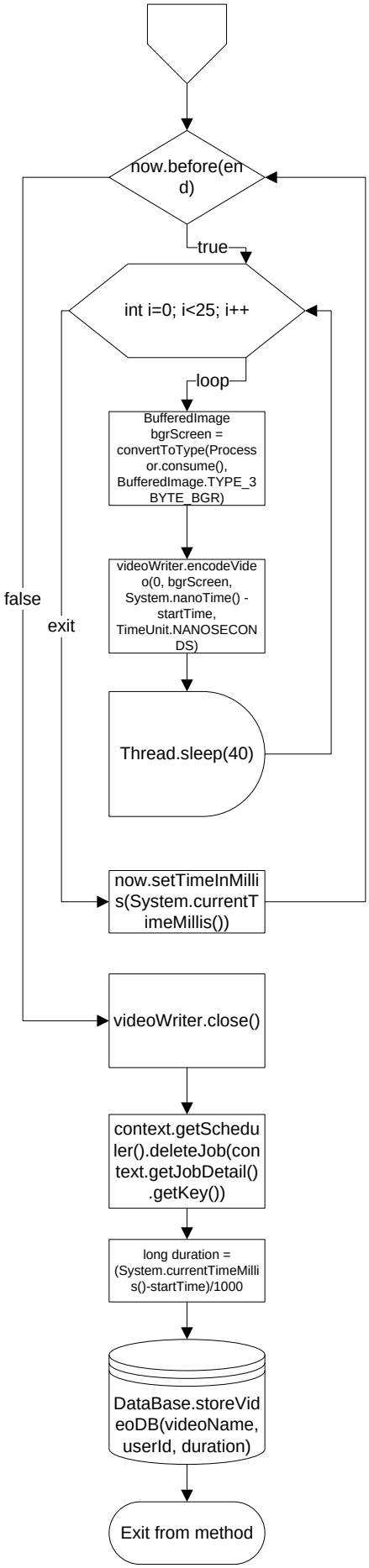
Όπως αναφέραμε στην προηγούμενη παράγραφο οι εργασίες εκτελούνται στην κλάση Schedule. Αρχικά ορίζουμε μία μεταβλητή τύπου JobDataMap με όνομα dataMap και αποθηκεύουμε σε αυτή τα JobDetails που έχουμε περάσει με τον τρόπο που είδαμε στην προηγούμενη πράγραφο. Έπειτα με την String videoName = dataMap.getString("Name") αποθηκεύουμε το όνομα του βίντεο στην μεταβλητή videoName. Στην συνέχεια αποθηκεύουμε το id του χρήστη με την int userId = dataMap.getInt("userId"). Γιαν να βρούμε την χρονική στιγμή που θα σταματήσει η εγγραφή βίντεο θα πρέπει να φτιάξουμε την μορφή της μερομήνια και της ώρας με το εξής τρόπο DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss"). Θέτουμε σαν χρονική ζώνη την χρονική ζώνη του υπολογιστή με την dateFormat.setTimeZone(TimeZone.getDefault()). Έπειτα αποθηκεύουμε με την εντολή date = dateFormat.parse(dataMap.getString("Date")) στην μεταβλητή date την χρονική στιγμή στην μορφή που δημιουργήσαμε προηγουμένως. Στην συνέχεια δημιουργούμε δύο ημερολόγια με ονόματα now και end. Με την end.setTimeZone(TimeZone.getDefault()) και now.setTimeZone(TimeZone.getDefault()) θέτουμε σαν χρονική ζώνη και στα δύο την χρονική ζώνη του υπολογιστή και στο ημερολόγιο με το όνομα end την χρονική στιγμή που θα σταματήσει η εγγραφή του βίντεο με την μέθοδο end.setTime(date). Με την now.setTimeInMillis(System.currentTimeMillis())

θέτουμε σαν χρονική στιγμή στο ημερολόγιο με το όνομα now την τρέχουσα χρονική στιγμή. Με τις videoWriter = ToolFactory.makeWriter("c://"+videoName+".mp4"), videoWriter.addVideoStream(0, 0, ICodec.ID.CODEC_ID_MPEG4,320, 240) και startTime = System.currentTimeMillis() αρχικοποιούμε την εγγραφή βίντεο όπως είδαμε στην κλάση Client. Στην συνέχεια μπαίνουμε σε ένα βρόχο while ο οποίος επαναλαμβάνεται όσο η τρέχουσα χρονική στιγμή είναι χρονικά ποιο πίσω από την χρονική στιγμή που έχουμε θέσει για να σταματήσει η εγγραφή του βίντεο. Μέσα στον βρόχο while έχουμε ένα βρόχο for ο οποίος είναι ίδιος με τον βρόχο της μεθόδου recVideo() της κλάσης Client για την δημιουργία των καρέ. Έπειτα με την now.setTimeInMillis(System.currentTimeMillis()) θέτουμε ξανά την χρονική στιγμή στο ημερολόγιο με το όνομα now την τρέχουσα χρονική στιγμή. Εάν η τρέχουσα χρονική στιγμή είναι ίση ή ποιό μπροστά από την χρονική στιγμή στο ημερολόγιο end τότε σταματάει να επαναλαμβάνεται ο βρόχος while και με την videoWriter.close() σταματάμε την εγγραφή βίντεο. Για να σβήσουμε την συγκεκριμένη εργασία από την βάση δεδομένων χρησιμοποιούμε την context.getScheduler().deleteJob(context.getJobDetail().getKey()). Τέλος αποθηκεύουμε την διάρκεια του βίντεο σε δευτερόλεπτα στην μεταβλητή duration και καλούμε την μέθοδο storeVideoDB(videoName, userId, duration) από την κλάση DataBase για να καταγράψουμε την εγγραφή στην βάση δεδομένων. Η μέθοδος convertToType(BufferedImage sourceImage, int targetType) είναι ίδια με την αντίστοιχη μέθοδο που υπάρχει στην κλάση Client και έτσι δεν χρειάζεται εξήγηση. Η μέθοδος execute (JobExecutionContext context) και το διάγραμμα ροής φαίνονται παρακάτω:

```
public void execute throws JobExecutionException {

        JobDataMap dataMap = context.getJobDetail().getJobDataMap();
    String videoName = dataMap.getString("Name");
    int userId = dataMap.getInt("userId");
    DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
    dateFormat.setTimeZone(TimeZone.getDefault());
    try {
            date = dateFormat.parse(dataMap.getString("Date"));
        } catch (ParseException e) {
            e.printStackTrace();
        }
    Calendar now = Calendar.getInstance();
    Calendar end = Calendar.getInstance();
    end.setTimeZone(TimeZone.getDefault());
    end.setTime(date);
    now.setTimeZone(TimeZone.getDefault());
    now.setTimeInMillis(System.currentTimeMillis());
    videoWriter = ToolFactory.makeWriter("c://"+videoName+".mp4");
    videoWriter.addVideoStream(0, 0, ICodec.ID.CODEC_ID_MPEG4,320, 240);
    startTime = System.currentTimeMillis();
    while(now.before(end)){
        for(int i=0; i<25; i++){
        BufferedImage bgrScreen = convertToType(Processor.consumer(),
            BufferedImage.TYPE_3BYTE_BGR);
    // encode the image to stream #0
        videoWriter.encodeVideo(0, bgrScreen, System.nanoTime() - startTime,
            TimeUnit.NANOSECONDS);
       // sleep for frame rate milliseconds
```

```
            try {
               Thread.sleep(40);
            }
            catch (InterruptedException e) {
               // ignore
            }
                        }
     now.setTimeInMillis(System.currentTimeMillis());
    }
            videoWriter.close();
    try {
                        context.getScheduler().deleteJob(context.getJobDetail().getKey());
            } catch (SchedulerException e) {
                        e.printStackTrace();
            }
    long duration = (System.currentTimeMillis()-startTime)/1000;
    DataBase.storeVideoDB(videoName, userId, duration);
}
```

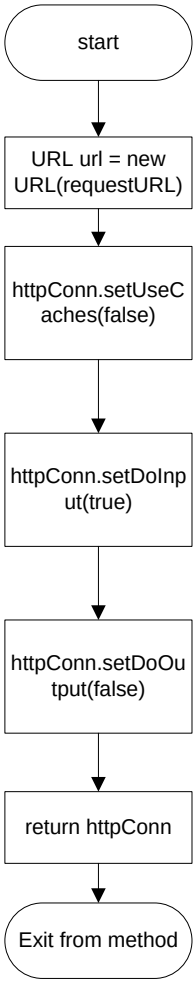

start
JobDataMap dataMap = context.getJobDetail().getJobDataMap()
String videoName = dataMap.getString("Name")
int userId = dataMap.getInt("userId")
DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss")
dateFormat.setTimeZone(TimeZone.getDefault())
date = dateFormat.parse(dataMap.getString("Date"))
Calendar now = Calendar.getInstance()
Calendar end = Calendar.getInstance()
end.setTimeZone(TimeZone.getDefault())
end.setTime(date)
now.setTimeZone(TimeZone.getDefault())
now.setTimeInMillis(System.currentTimeMillis())
videoWriter = ToolFactory.makeWriter("c:// "+videoName+".mp4")
videoWriter.addVideoStream(0, 0, ICodec.ID.CODEC_ID_MPEG4,320, 240)
startTime = System.currentTimeMillis()


now.before(end)
true
int i=0; i<25; i++
loop
BufferedImage bgrScreen = convertToType(Processor.consume(), BufferedImage.TYPE_3BYTE_BGR)
videoWriter.encodeVideo(0, bgrScreen, System.nanoTime() - startTime, TimeUnit.NANOSECONDS)
Thread.sleep(40)
false
exit
now.setTimeInMillis(System.currentTimeMillis())
videoWriter.close()
context.getScheduler().deleteJob(context.getJobDetail().getKey())
long duration = (System.currentTimeMillis()-startTime)/1000
DataBase.storeVideoDB(videoName, userId, duration)
Exit from method

2.7 Η κλάση HttpUtility

Η κλάση HttpUtility περιέχει μεθόδους για να μπορούμε να κάνουμε request σε ένα Server μέσω http get, επίσης παρέχει μεθόδους για να παίρνουμε απαντήσεις από ένα Server. Στην συγκεκριμένη περίπτωση Server θα εννοούμε τον iLon από τον οποίο θέλουμε να πάρουμε την θερμοκρασία και τα lux. Όλες οι μεταβλητές και οι μέθοδοι σε αυτή την κλάση είναι τύπου static. Αρχικά ορίζουμε μία μεταβλητή τύπου HttpURLConnection με όνομα httpConn η ποία αναπαριστά μια http σύνδεση. Με την μέθοδο sendGetRequest(String requestURL) στέλνουμε στον Server που έχουμε περάσει σαν παράμετρο στην μέθοδο ένα request τύπου get. Αρχικά ορίζουμε ένα νέο url με όνομα url με τον εξής τρόπο URL url = new URL(requestURL). Έπειτα ανοίγουμε την σύνδεση μεταξύ εφαρμογής και Server με την httpConn = (HttpURLConnection) url.openConnection(). Για την συγκεκριμένη σύνδεση δεν χρησιμοποιούμε cache μνήμη περνώντας την παράμετρο false στην μέθοδο httpConn.setUseCaches(false). Εάν θέλουμε να διαβάσουμε την απάντηση του Server περνάμε την παράμετρο true στην μέθοδο httpConn.setDoInput(true) και τέλος για να δείξουμε ότι κάνουμε ένα get request περνάμε σαν παράμετρο την false στην μέθοδο httpConn.setDoOutput(false) και με την return httpConn επιστρέφουμε την μεταβλητή httpConn λόγο ότι θα την χρειαστούμε στην συνέχεια. Η μέθοδος μαζί με το διάγραμμα ροής φαίνονται παρακάτω:

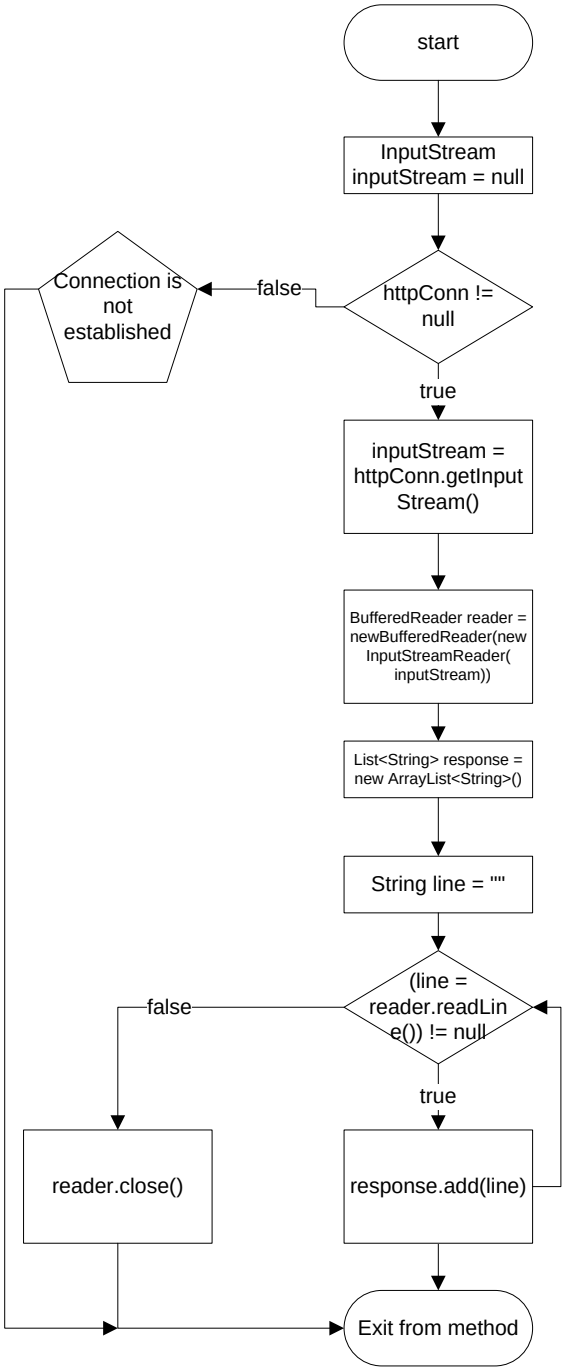
```
public static HttpURLConnection sendGetRequest(String requestURL)
        throws IOException {
    URL url = new URL(requestURL);
    httpConn = (HttpURLConnection) url.openConnection();
    httpConn.setUseCaches(false);
    httpConn.setDoInput(true); // true if we want to read server's response
    httpConn.setDoOutput(false); // false indicates this is a GET request
    return httpConn;
}
```



Με την μέθοδο readMultipleLinesRespone() διαβάζουμε την απάντηση του Server. Αρχικά ορίζουμε μία μεταβλητή τύπου InputStream με όνομα inputStream με την οποία θα διαβάσουμε την απάντηση του Server. Στην συνέχεια εάν έχει επιτευχθεί η σύνδεση μεταξύ της εφαρμογής και του Server διαβάζουμε την απάντηση με την inputStream = httpConn.getInputStream() στην αντίθετη περίπτωση δημιουργείτε μία εξαίρεση και εμφανίζεται στην κονσόλα το μήνυμα Connection is not established. Στην συνέχεια ορίζουμε μία μεταβλητή τύπου BufferedReader με όνομα reader και περνάμε σαν παράμετρο το inputStream για να το διαβάσουμε. Ορίζουμε την μεταβλητή line που είναι τύπου string στην οποία θα αποθηκεύουμε προσωρινά την κάθε γραμμή της απάντησης του Server. Έπειτα μπαίνουμε σε ένα βρόχο while ο οποίος επαναλαμβάνεται όσο υπάρχουν γραμμές για να διαβάσει το πρόγραμμα. Κάθε φορά που διαβάζει το πρόγραμμα μία γραμμή την αποθηκεύει στην λίστα που έχουμε ορίσει προηγουμένως με την response.add(line). Τέλος κλείνουμε τον reader με την reader.close() και επιστρέφουμε τον πίνακα που έχουμε αποθήκευση την απάντηση του Server. Η μέθοδος μαζί με το διάγραμμα ροής φαίνονται παρακάτω:

```
public static String[] readMultipleLinesRespone() throws IOException {
    InputStream inputStream = null;
    if (httpConn != null) {
        inputStream = httpConn.getInputStream();
    } else {
        throw new IOException("Connection is not established.");
    }
    BufferedReader reader = new BufferedReader(new InputStreamReader(
```

```
            inputStream));
    List<String> response = new ArrayList<String>();
    String line = "";
    while ((line = reader.readLine()) != null) {
       response.add(line);
    }
    reader.close();
    return (String[]) response.toArray(new String[0]);
}
```

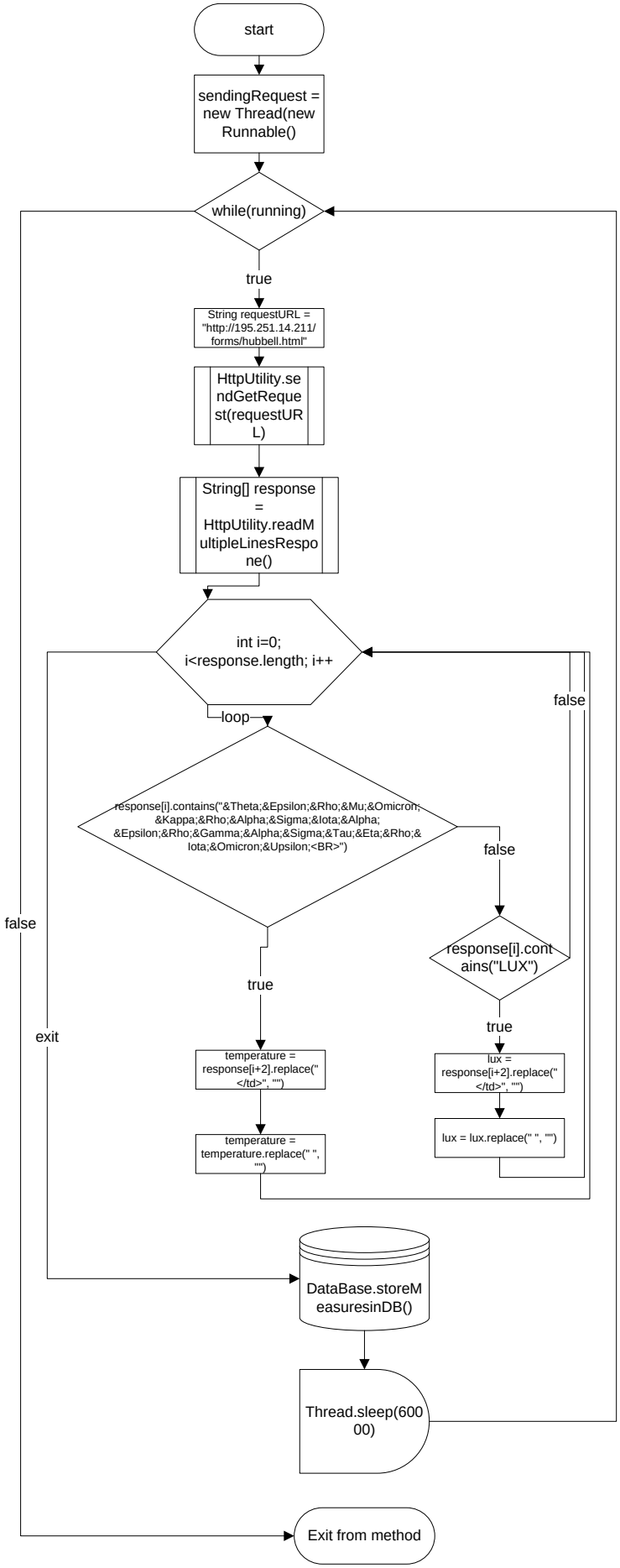


2.8 Η κλάση HttpRequest

Η κλάση HttpRequest χρησιμοποίει τις μεθόδους της κλάσης HttpUtility για να στείλει get request στο Server που στην συγκεκριμένη περίπτωση είναι ο iLon. Η HttpRequest περιέχει μόνο μία μέθοδο την sendGetRequest() μέσα στην οποία έχουμε δημιουργήσει ένα νέο νήμα επεξεργασίας με το όνομα sendingRequest. Μέσα στην run έχουμε ένα βρόχο while ο οποίος επαναλαμβάνεται όσο η τιμή της μεταβλητής running είναι true. Στην αρχή του βρόχου ορίζουμε μία μεταβλητή με όνομα requestURL και τύπου string η οποία περιέχει το url της ιστοσελίδας του iLon από την οποία θα πάρουμε τα δεδομένα. Το url αυτό είναι το http://195.251.14.211/forms/hubbell.html. Έπειτα μέσω της μεθόδου sendGetRequest(requestURL) που ανήκει στην κλάση HttpUtility στέλνουμε το request στον Server αφού έχουμε περάσει το url σαν παράμετρο στην μέθοδο. Μέσω της HttpUtility.readMultipleLinesRespone() αποθηκεύουμε την απάντηση του Server στον πίνακα με το όνομα response με τον εξής τρόπο String[] response = HttpUtility.readMultipleLinesRespone(). Στην συνέχεια μπαίνουμε σε ένα βρόχο for ο οποίος ξεκινάει από το μηδέν μέχρι το μήκος του πίνακα response. Με τον συγκεκριμένο βρόχο ψάχνουμε να βρούμε ποια από τις σειρές της απάντησης του Server περιέχει την συγκεκριμένη συμβολοσειρά ΘΕΡΜΟΚΡΑΣΙΑ ΕΡΓΑΣΤΗΡΙΟΥ
. Όταν βρούμε την συγκεκριμένη συμβολοσειρά αυξάνουμε τον δείκτη του πίνακα κατά δύο η οποία έχει την εξής μορφή: </td> 25 όπου 25 είναι μία τυχαία τιμή της θερμοκρασίας. Με τους εξής τρόπους temperature = response[i+2].replace("</td>", "") και temperature = temperature.replace(" ", "") απομονώνουμε την θερμοκρασία και την αποθηκεύουμε στην μεταβλητή temperature. Η ίδια διαδικασία ακολουθητέε για να βρούμε την τιμή των lux με την μόνη διαφορά ότι τώρα ψάχνουμε να βρούμε την συμβολοσειρά Lux. Αφού βρούμε την συγκεκριμένη συμβολοσειρά αυξάνουμε πάλι τον δείκτη κατά δύο και με την ίδια διαδικασία απομονώνουμε την τιμή των lux και την αποθηκεύουμε στην μεταβλητή με όνομα lux. Έπειτα καλούμε την DataBase.storeMeasuresinDB() και αποθηκεύουμε τις τιμές στην βάση δεδομένων όπως είδαμε προηγουμένως και τέλος με την Thread.sleep(60000) περιμένουμε 1 λεπτό για να πάρουμε τις επόμενες τιμές. Η μέθοδος μαζί με το διάγραμμα ροής φαίνονται παρακάτω:

```
  public static void sendGetRequest() {
         sendingRequest = new Thread(new Runnable(){
@Override
public void run() {
while(running){
 // sending GET request
String requestURL = "http://195.251.14.211/forms/hubbell.html";
try {
HttpUtility.sendGetRequest(requestURL);
String[] response = HttpUtility.readMultipleLinesRespone();
for(int i=0; i<response.length; i++){
if(response[i].contains("&Theta;&Epsilon;&Rho;&Mu;&Omicron;&Kappa;&Rho;&Alpha;&
Sigma;&Iota;&Alpha;&Epsilon;&Rho;&Gamma;&Alpha;&Sigma;&Tau;&Eta;&Rho;&Iota;
&Omicron;&Upsilon;<BR>")){
temperature = response[i+2].replace("</td>", "");
temperature = temperature.replace(" ", "");
                                                }
```

```
if(response[i].contains("LUX")){
lux = response[i+2].replace("</td>", "");
lux = lux.replace(" ", "");
                                        }
                                    }
                                } catch (IOException ex) {
                                    ex.printStackTrace();
                                }
DataBase.storeMeasuresinDB();
                                try {
                                        Thread.sleep(60000);
                                    } catch (InterruptedException e) {
                                        e.printStackTrace();
                                    }
                                }
                            }
        });
        sendingRequest.start();
    }
```


start
sendingRequest = new Thread(new Runnable()
while(running)
true
String requestURL = "http://195.251.14.211/forms/hubbell.html"
HttpUtility.sendGetRequest(requestURL)
String[] response = HttpUtility.readMultipleLinesRespone()
int i=0; i<response.length; i++
loop
false
response[i].contains("ΘΕΡΜΟΚΡΑΣΙΑΕΡΓΑΣΤΗΡΙΟΥ
")
false
response[i].contains("LUX")
true
temperature = response[i+2].replace("</td>", "")
temperature = temperature.replace(" ", "")
true
lux = response[i+2].replace("</td>", "")
lux = lux.replace(" ", "")
false
exit
DataBase.storeMeasuresinDB()
Thread.sleep(60000)
Exit from method

ΚΕΦΑΛΑΙΟ 3

3.1 Η αρχιτεκτονική του Applet

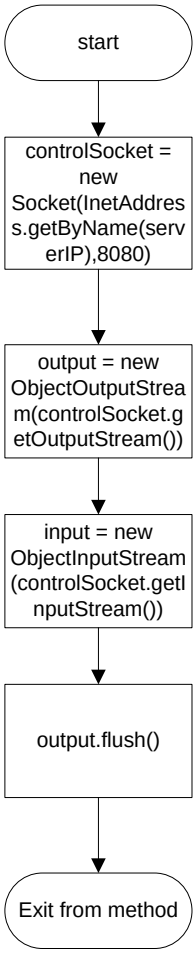
Το Java applet είναι μία εφαρμογή η οποία είναι ενσωματωμένη σε μία ιστοσελίδα και εκτελείτε από την JVM. Η μόνη διαφορά ως προς την σύνταξη που έχει ένα java applet με ένα java application είναι ότι στην περίπτωση του java applet δεν έχουμε την μέθοδο public static void main(String[] args) αλλά 4 διαφορετικές μεθόδους. Η μέθοδος init χρησιμοποιείτε για την αρχικοποίηση του και καλείτε από τον browser. Η μέθοδος start καλείτε από τον browser αμέσως μετά την μέθοδο init ή όταν ο χρήστης επιστρέφει στην ιστοσελίδα που περιέχει το applet. Η μέθοδος stop καλείτε αυτόματα όταν ο χρήστης φεύγει από την σελίδα που περιέχει το applet και η μέθοδος destroy καλείτε μόνο όταν η λειτουργία του browser τερματίζεται φυσιολογικά. Στην δική μας περίπτωση το applet χρησιμοποίει αυτές τις τέσσερις μεθόδους όπως επίσης και κάποιες επιπλέον οι οποίες εκτελούν κάποιες διαδικασίες. Οι λειτουργίες του applet είναι να στέλνει το όνομα χρήστη και τον κωδικό πρόσβασης στον Server, να λαμβάνει εικόνα από τον Server, να μπορεί ο χρήστης να καταγράφει βίντεο και εικόνα στον cilent ή στον Server, να λαμβάνει τιμές από τους αισθητήρες του iLon και να τις παρουσιάζει σε γραφήματα, να προγραμματίζει εγγραφές βίντεο στον Server και να ελέγχει την κάμερα στους τέσσερις άξονες.

3.2 Η κλάση Receiver

Η κλάση Receiver περιέχει τις τέσσερις βασικές μεθόδους που αναφέραμε ποιο πάνω και κάποιες επιπλέον. Η init περιέχει την αρχικοποίηση του GUI όπως είναι τα κουμπιά, τα πεδία κειμένου για την εισαγωγή τιμών, και τα γραφικά. Επειδή η συγκεκριμένη μέθοδος είναι πολύ μεγάλη δεν θα την αναλύσουμε διεξοδικά όπως κάναμε με τις προηγούμενες μεθόδους αλλά θα αναλύσουμε κάποια σημαντικά κομμάτια στην συνέχεια του κεφαλαίου. Η πρώτη μέθοδος που θα αναλύσουμε είναι η start η οποία καλείτε αμέσως μετά την init. Η λειτουργία της start είναι η αρχικοποίηση της σύνδεσης με τον server με το TCP πρωτόκολλο επικοινωνίας. Αρχικά δημιουργούμε ένα socket το οποίο το ονομάζουμε controlSocket με τον εξής τρόπο controlSocket = new Socket(InetAddress.getByName(serverIP),8080). Όπως βλέπουμε το συγκεκριμένο socket έχει ip διεύθυνση την ip του Server και port το 8080. Η ip του Server είναι αποθηκευμένη στην μεταβλητή serverIP. Για να βρούμε την ip του Server στην μέθοδο init με την url = getDocumentBase() αποθηκεύουμε το url από το οποίο προέρχεται η ιστοσελίδα που έχει ενσωματωμένο το applet. Έπειτα με την serverIP = url.getHost() αποθηκεύουμε την ip του Server στην μεταβλητή serverIP. Με αυτό τον τρόπο δεν υπάρχει δέσμευση ποιά θα είναι η ip του Server. Με τις output = new ObjectOutputStream(controlSocket.getOutputStream()), input = new ObjectInputStream(controlSocket.getInputStream()) και output.flush() δημιουργούμε την σύνδεση μεταξύ applet και Server. Παρακάτω φαίνονται η μέθοδος και το διάγραμμα ροής:

```
public void start(){
        try {
                controlSocket = new Socket(InetAddress.getByName(serverIP),8080);
                output = new ObjectOutputStream(controlSocket.getOutputStream());
                input = new ObjectInputStream(controlSocket.getInputStream());
                output.flush();
        } catch (UnknownHostException e) {
                e.printStackTrace();
        } catch (IOException e) {
```

```
                e.printStackTrace();
            }
        }
```

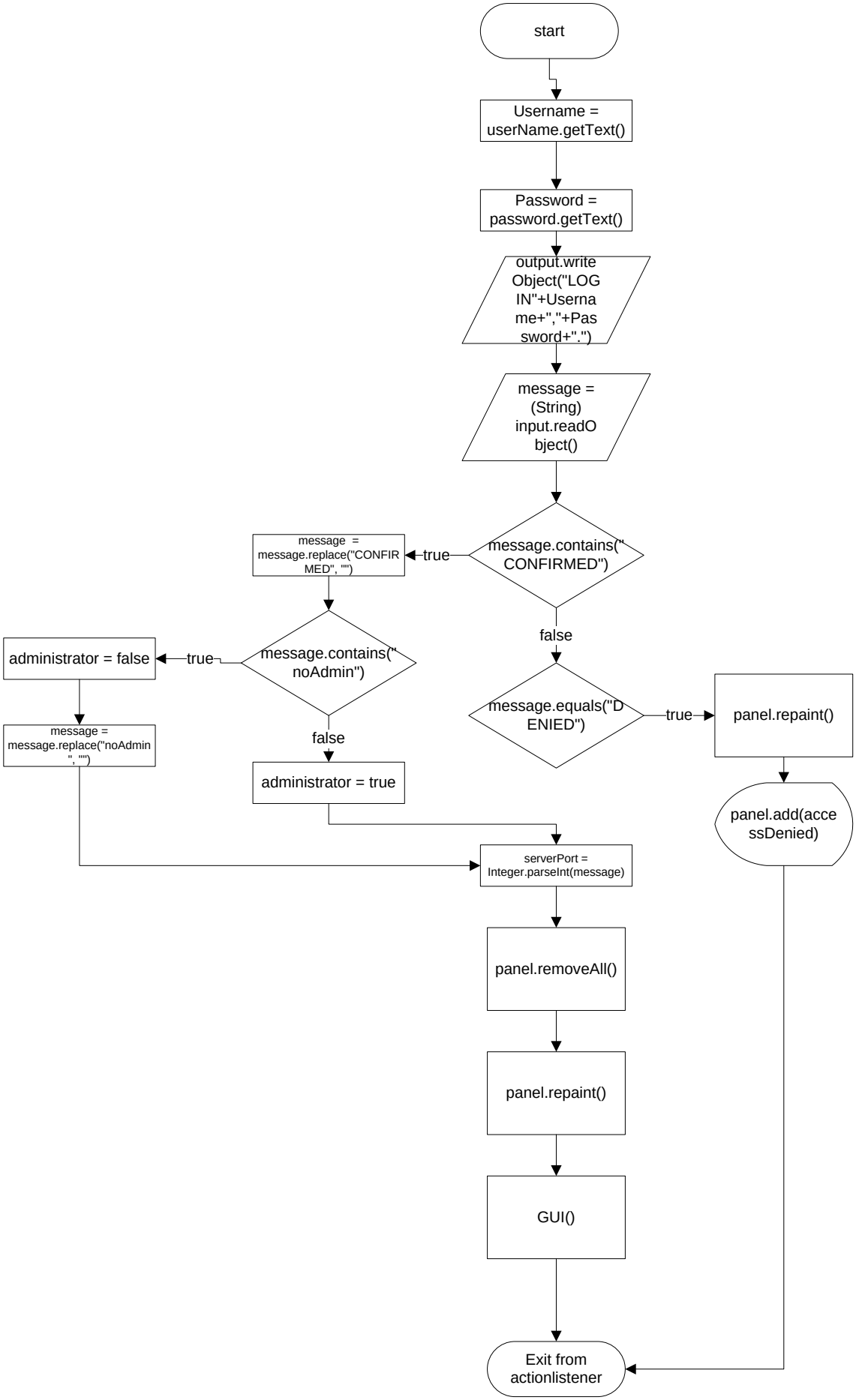


Αυτή τη στιγμή το applet περιμένει να γράψουμε το όνομα χρήστη και τον κωδικό πρόσβασης ώστε να τα στείλει στον Server. Αφού ο χρήστης εισάγει τα στοιχεία του και πατήσει το κουμπί Log In τότε εκτελείτε ο κώδικας μέσα στον ακροατή ενεργειών που ανήκει στο συγκεκριμένο κουμπί. Με τον όρο ακροατή ενεργειών ονομάζουμε την διαδικασία που έχουμε ορίσει στην μέθοδο init και εκτελείτε όταν πατήσουμε το κουμπί στο οποίο ανήκει. Παρακάτω φαίνεται ο ακροατής ενεργειών με όνομα login:

```
logIn.addActionListener(new ActionListener(){
                @Override
                public void actionPerformed(ActionEvent e) {
                        Username = userName.getText();
                        Password = password.getText();
                        try {
                        output.writeObject("LOGIN"+Username+","+Password+".");
                        } catch (IOException e1) {
                                e1.printStackTrace();
                        }
                        try {
                                message = (String) input.readObject();
                        } catch (ClassNotFoundException e1) {
                                e1.printStackTrace();
```

```
} catch (IOException e1) {
        e1.printStackTrace();
}
if(message.contains("CONFIRMED")){
        message  = message.replace("CONFIRMED", "");
        if(message.contains("noAdmin")){
                administrator = false;
                message = message.replace("noAdmin", "");
        }
        else{
                administrator = true;
        }
        serverPort = Integer.parseInt(message);
        panel.removeAll();
        panel.repaint();
        GUI();
}
if(message.equals("DENIED")){
        panel.repaint();
        panel.add(accessDenied);
} } });
```

Σε κάθε ακροατή ενεργειών υπάρχει η μέθοδος actionPerformed(ActionEvent e) μέσα στην οποία βρίσκεται ο κώδικας που θέλουμε να εκτελέσουμε. Στην αρχή της μεθόδου διαβάζουμε τα στοιχεία του χρήστη από τα δύο πεδία κειμένου και τα αποθηκεύουμε στις μεταβλητές Username και Password. Στην συνέχεια με την output.writeObject("LOGIN"+Username+","+Password+".") στέλνουμε το μήνυμα στον Server με την μορφή LOGIN"+Username+","+Password+".". Αφού σταλθεί το μήνυμα το applet με την message = (String) input.readObject() περιμένει να διαβάσει την απάντηση του Server και την αποθηκεύει στην μεταβλητή message. Εάν η απάντηση του Server περιέχει την συμβολοσειρά CONFIRMED και την noAdmin τότε θέτουμε την τιμή της μεταβλητής administrator σε false. Στην περίπτωση που δεν περιέχει την συμβολοσειρά noAdmin θέτουμε την τιμή της μεταβλητής administrator σε true. Και στις δύο περιπτώσεις με την serverPort = Integer.parseInt(message) αποθηκεύουμε το αριθμό του UDP port στην μεταβλητή serverPort. Στην συνέχεια με τις panel.removeAll() και panel.repaint() καθαρίζουμε το applet από τα γραφικά και με την μέθοδο GUI() σχεδιάζουμε GUI του applet αφού και στις δύο παραπάνω περιπτώσεις έχουμε γίνει δεκτοί στην εφαρμογή. Τέλος αν τα στοιχεία που εισήγαγε ο χρήστης δεν είναι σωστά τότε το μήνυμα περιέχει την συμβολοσειρά DENIED και το μόνο που εμφανίζεται στην οθόνη είναι η ένδειξη INVALID USERNAME OR PASSWORD και το applet περιμένει ο χρήστης να εισάγει τα σωστά στοιχεία του. Το διάγραμμα ροής του ακροατή ενεργειών login φαίνεται παρακάτω:

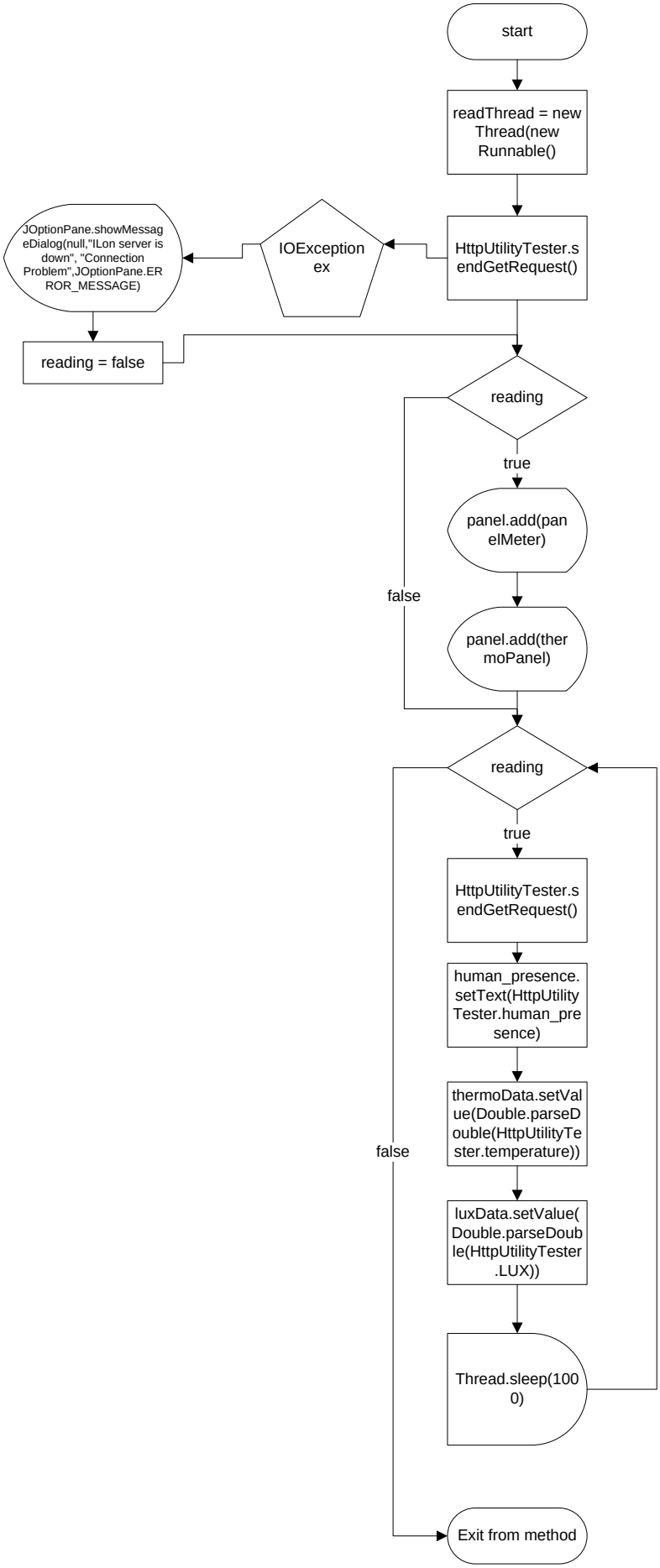

start
Username = userName.getText()
Password = password.getText()
output.write Object("LOG IN"+Userna me+","+Pas sword+".")
message = (String) input.readO bject()
message.contains(" CONFIRMED")
true
message = message.replace("CONFIR MED", "")
false
message.contains(" noAdmin")
true
administrator = false
message = message.replace("noAdmin ", "")
false
administrator = true
message.equals("D ENIED")
true
panel.repaint()
panel.add(acce ssDenied)
serverPort = Integer.parseInt(message)
panel.removeAll()
panel.repaint()
GUI()
Exit from actionlistener

Όπως αναφέραμε ποιο πάνω η μέθοδος GUI() δημιουργεί το gui του applet δηλαδή τα κουμπιά, τα πεδία κειμένου, την ένδειξη θερμοκρασίας και φωτεινότητας των οποίων η λειτουργία θα εξηγηθεί αναλυτικότερα στο κεφάλαιο 5. Εάν ο χρήστης έχει δικαιώματα διαχειριστή τότε εμφανίζει και τα κουμπιά με τα οποία μπορεί να ελέγχει την κάμερα στους τέσσερις άξονες. Μία άλλη λειτουργία της GUI() είναι να καλεί την μέθοδο readValues() η οποία διαβάζει τις τιμές από τον iLON και τις εμφανίζει στο applet. Η μέθοδος readValues() φαίνεται παρακάτω:

```
public void readValues(){
                readThread = new Thread(new Runnable(){
                        @Override
                        public void run() {
                                try{
                                        HttpUtilityTester.sendGetRequest();
                                }catch(IOException ex){
JOptionPane.showMessageDialog(null,"ILon server is down", "Connection
Problem",JOptionPane.ERROR_MESSAGE);
                                        reading = false;
                                }
                                if(reading){
                                        panel.add(panelMeter);
                                        panel.add(thermoPanel);
                                }
                                while(reading){
                                        try {
                                                HttpUtilityTester.sendGetRequest();
                                        } catch (IOException e1) {
                                                e1.printStackTrace();
                                        }

        human_presence.setText(HttpUtilityTester.human_presence);

        thermoData.setValue(Double.parseDouble(HttpUtilityTester.temperature));

        luxData.setValue(Double.parseDouble(HttpUtilityTester.LUX));
                                        try {
                                                Thread.sleep(1000);
                                        } catch (InterruptedException e) {
                                                e.printStackTrace();
                                        } } }
                                    }); }
```

Η μέθοδος readValues() περιέχει ένα επιπλέον νήμα επεξεργασίας το οποίο έχουμε ονομάσει readThread και έχει την ίδια αρχιτεκτονική με τα νήματα επεξεργασίας του Server. Αρχικά στην μέθοδο run κάνουμε ένα http request στον iLon με την μέθοδο HttpUtilityTester.sendGetRequest() που ανήκει στην κλάση HttpUtilityTester. Εάν για κάποιο λόγο ο iLon δεν ανταποκρίνεται δημιουργείτε μία εξαιρέσει και με την JOptionPane.showMessageDialog(null,"ILon server is down", "Connection Problem",JOptionPane.ERROR_MESSAGE) εμφανίζεται ένα νέο παράθυρο το οποίο μας πληροφορεί ότι ο υπάρχει ένα πρόβλημα στην σύνδεση και ο iLon είναι εκτός λειτουργίας. Επίσης στην συγκεκριμένη περίπτωση η τιμή της μεταβλητής reading γίνεται false. Στην

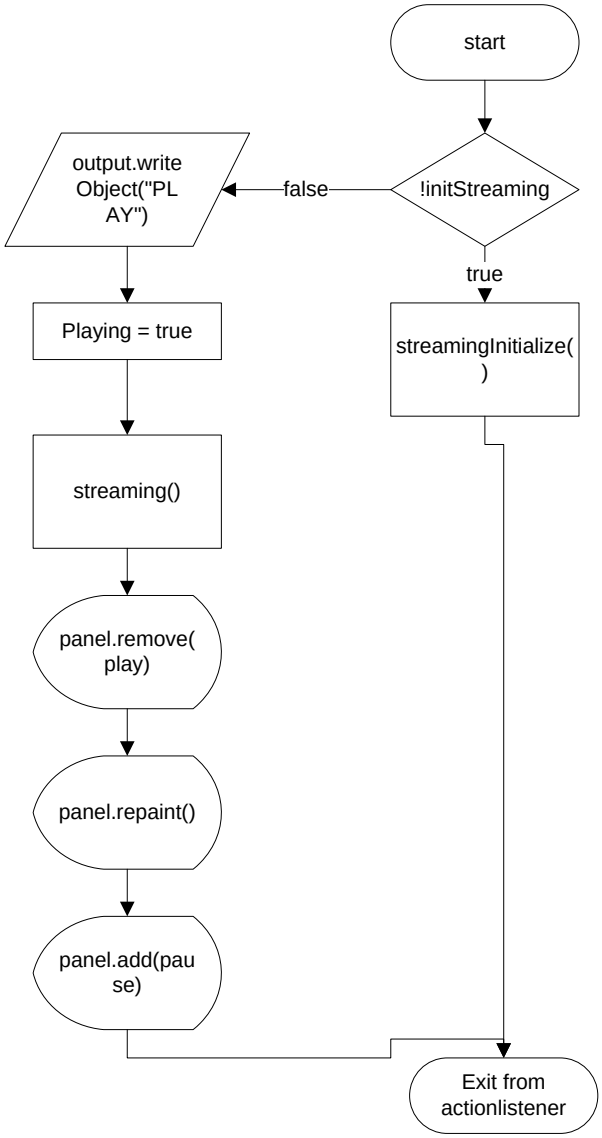
συνέχεια εάν η τιμή της reading είναι true δηλαδή ο iLon είναι εντός λειτουργίας με τις panel.add(panelMeter) και panel.add(thermoPanel) εμφανίζουμε στο applet το θερμόμετρο και τον δείκτη ενδείξεως των lux. Έπειτα μπαίνουμε σε ένα βρόχο while ο οποίο επαναλαμβάνεται όσο η τιμή της reading είναι true. Αρχικά στο βρόχο κάνουμε ένα http request με τον τρόπο που είδαμε προηγουμένως. Με την human_presence.setText(HttpUtilityTester.human_presence) εμφανίζουμε την λέξη OC_OCCUPIED στο πεδίο κειμένου που μας πληροφορεί ότι υπάρχει ανθρώπινη παρουσία στον χώρο, σε αντίθετη περίπτωση εμφανίζεται η λέξη OC_UNOCCUPIED. Για να εμφανίσουμε την τιμή της θερμοκρασίας στο θερμόμετρο χρησιμοποιούμε την μέθοδο thermoData.setValue(Double.parseDouble(HttpUtilityTester.temperature)). Η μέγιστη τιμή που μπορεί να πάρει το θερμόμετρο είναι 45 °C και η ελάχιστη είναι -10 °C. Με την luxData.setValue(Double.parseDouble(HttpUtilityTester.LUX)) εμφανίζουμε την τιμή των lux σε ένα κυκλικό μετρητή ο οποίος έχει μέγιστα όρια από 0 έως 400. Τέλος με την Thread.sleep(1000) περιμένουμε ένα δευτερόλεπτο μέχρι να πάρουμε την επόμενη μέτρηση. Το διάγραμμα ροής φαίνεται παρακάτω:


start
readThread = new Thread(new Runnable()
HttpUtilityTester.sendGetRequest()
IOException ex
JOptionPane.showMessageDialog(null,"ILon server is down", "Connection Problem",JOptionPane.ERROR_MESSAGE)
reading = false
reading
true
panel.add(panelMeter)
panel.add(thermoPanel)
false
reading
true
HttpUtilityTester.sendGetRequest()
human_presence.setText(HttpUtilityTester.human_presence)
thermoData.setValue(Double.parseDouble(HttpUtilityTester.temperature))
luxData.setValue(Double.parseDouble(HttpUtilityTester.LUX))
Thread.sleep(1000)
false
Exit from method

Πατώντας το κουμπί play στο GUI του applet καλείτε ο αντίστοιχος ακροατής ενεργειών και αρχίζει η αναπαραγωγή βίντεο σε πραγματικό χρόνο από την κάμερα. Ο ακροατής ενεργειών που αντιστοιχεί στο κουμπί play φαίνεται παρακάτω:

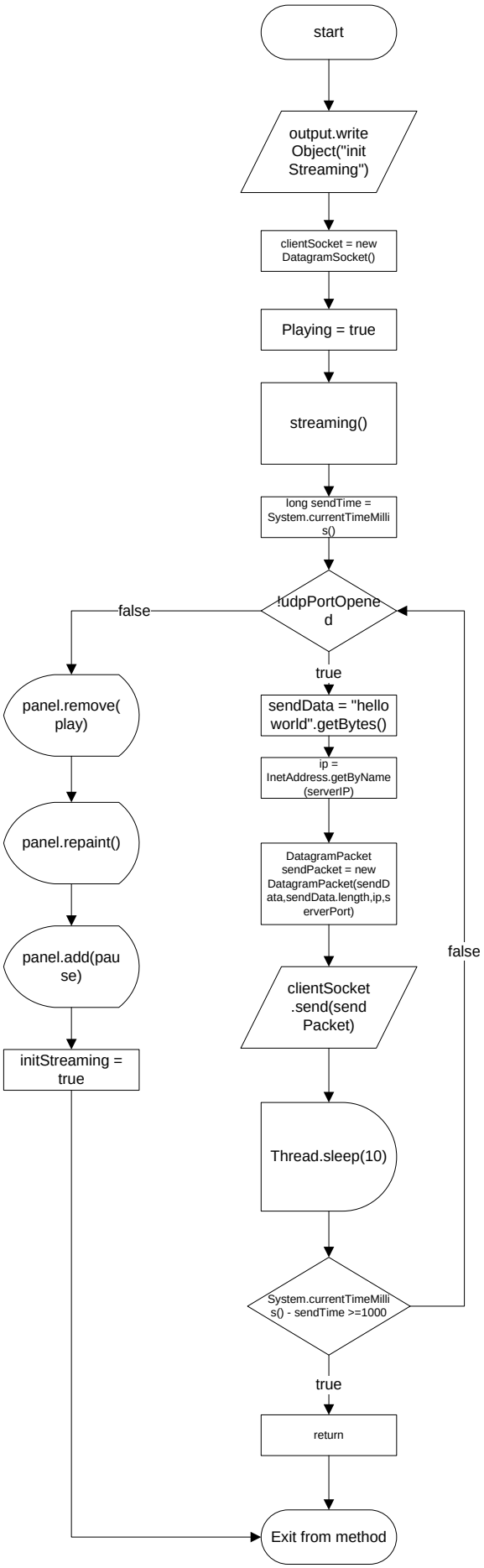
```
play.addActionListener(new ActionListener(){
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                if(!initStreaming){
                                        streamingInitialize();
                                }else{
                                        try {
                                                output.writeObject("PLAY");
                                                Playing = true;
                                                streaming();
                                        } catch (IOException e1) {
                                                e1.printStackTrace();
                                        }
                                        panel.remove(play);
                                        panel.repaint();
                                        panel.add(pause);
                                } } });
```

Η αρχικοποίηση της μετάδοσης βίντεο γίνεται την πρώτη φορά που ο χρήστης θα πατήσει το κουμπί play. Η αρχική τιμή της μεταβλητή initStreaming είναι false άρα μπαίνοντας για πρώτη φορά στον ακροατή ενεργειών ισχύει η συνθήκη !initStreaming και έτσι καλείται η μέθοδος streamingInitialize() της οποία η λειτουργία θα εξηγηθεί στην συνέχεια. Στην περίπτωση που έχει αρχικοποιηθεί η μετάδοση βίντεο στέλνουμε το μήνυμα PLAY στον Server με την εντολή output.writeObject("PLAY"), αλλάζουμε την τιμή της μεταβλητής Playing σε true και καλούμε την μέθοδο streaming(), διαγράφουμε το κουμπί play από το applet με την panel.remove(play), ανανεώνουμε το GUI του applet και εμφανίζουμε το κουμπί pause με την panel.add(pause) στην θέση του κουμπιού play. Το διάγραμμα ροής του ακροατή ενεργειών φαίνεται παρακάτω:



Όπως αναφέραμε πιο πάνω την μέθοδο streamingInitialize() την χρησιμοποιούμε για την αρχικοποίηση της μετάδοσης βίντεο από τον Server στο applet. Αρχικά στην μέθοδο στέλνουμε το μήνυμα initStreaming με την εντολή output.writeObject("initStreaming") και έπειτα δημιουργούμε ένα udp socket με όνομα clientSocket. Αλλάζουμε την τιμή της μεταβλητής Playing σε true και καλούμε την μέθοδο streaming() η οποία περιμένει να δεχτεί δεδομένα από τον Server. Στην συνέχεια στην μεταβλητή sendTime καταγράφουμε την ώρα του συστήματος με την long sendTime = System.currentTimeMillis(). Μπαίνουμε σε ένα βρόχο while ο οποίος επαναλαμβάνεται όσο η τιμή της μεταβλητής udpPortOpened είναι false. Ο βρόχος while έχει σκοπό να ανοίξουμε ένα udp port στο δρομολογητή όπως είχαμε αναφέρει στην παράγραφο 2.4. Αρχικά μέσα στο βρόχο μετατρέπουμε την λέξη hello world σε byte και τα αποθηκεύουμε στην μεταβλητή sendData με την εξής εντολή sendData = "hello world".getBytes(). Έπειτα με την ip = InetAddress.getByName(serverIP) αποθηκευουμε την ip του Server στην μεταβλητή ip και με την DatagramPacket sendPacket = new DatagramPacket(sendData,sendData.length,ip,serverPort) δημιουργούμε ένα πακέτο udp. Το συγκεκριμένο udp πακέτο περιέχει την ip του Server, την συμβολοσειρά hello world και το udp port που μας έχει στείλει ο Server και το στέλνουμε με την εντολή clientSocket.send(sendPacket). Τέλος περιμένουμε 10 ms με την Thread.sleep(10) και εάν έχει περάσει 1 δευτερόλεπτο από την αποστολή του πρώτου πακέτου τότε με το keyword

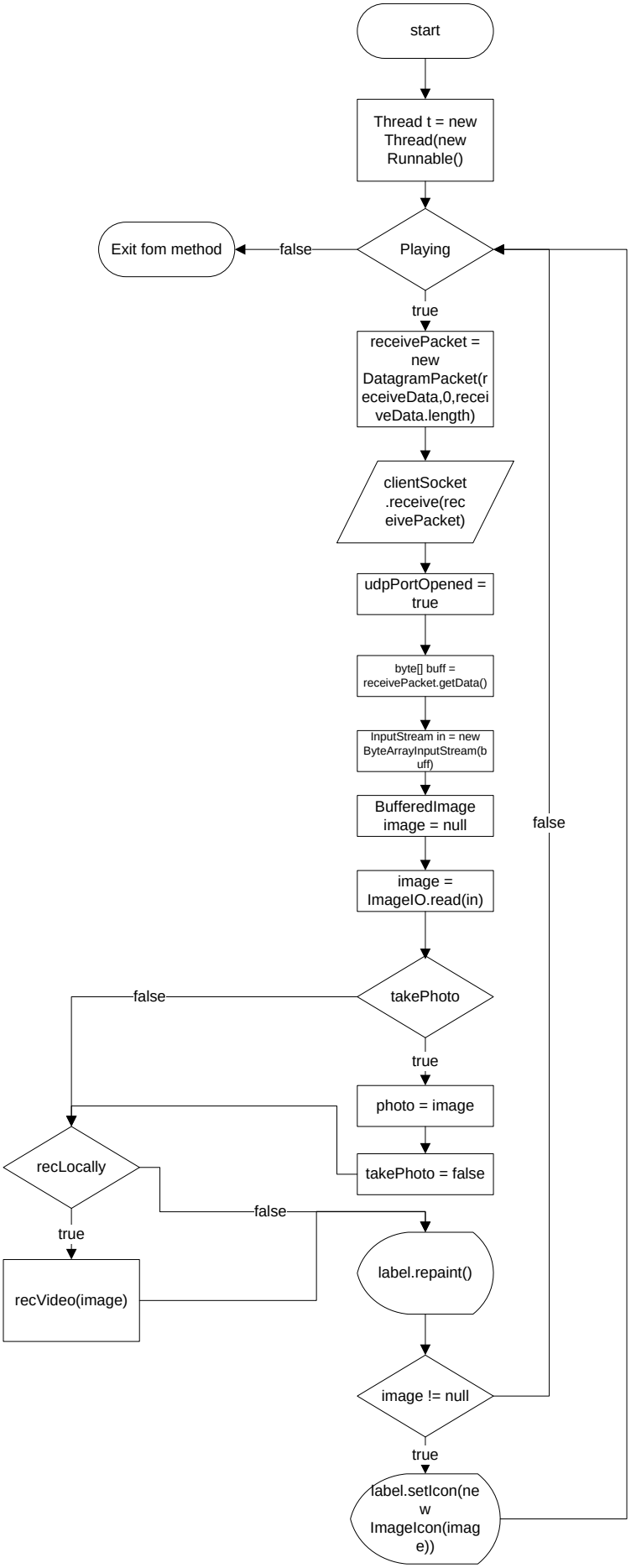
return διακόπτεται η λειτουργία της μεθόδου και ο χρήστης θα πρέπει να πατήσει πάλι το κουμπί play. Όλη η παραπάνω διαδικασία γίνεται γιατί ένα από τα κύρια χαρακτηριστικά του UDP είναι ότι δεν εγγυάται αξιόπιστη επικοινωνία. Τα πακέτα UDP που αποστέλλονται από έναν υπολογιστή μπορεί να φτάσουν στον παραλήπτη με λάθος σειρά, διπλά ή να μην φτάσουν καθόλου εάν το δίκτυο έχει μεγάλο φόρτο. Εάν μέσα στο χρονικό διάστημα του ενός δευτερολέπτου ο Server έχει διαβάσει το udp πακέτο που του στείλαμε και έχει αρχίσει την μετάδοση βίντεο, τότε η μεταβλητή udpPortOpened στην μέθοδο streaming() θα γίνει true και θα σταματήσει να επαναλαμβάνεται ο βρόχος while. Σε αυτή την περίπτωση αφαιρούμε το κουμπί play με την panel.remove(play), κάνουμε μία ανανέωση στο GUI του applet με την panel.repaint() και τοποθετούμε το κουμπί pause στην θέση του play. Τέλος αλλάζουμε την τιμή της μεταβλητής initStreaming σε true για να μην εκτελεστή η ίδια διαδικασία μόλις ο χρήστης πατήσει πάλι το κουμπί play. Το διάγραμμα ροής της μεθόδου streamingInitialize() φαίνεται παρακάτω:


start
output.write Object("init Streaming")
clientSocket = new DatagramSocket()
Playing = true
streaming()
long sendTime = System.currentTimeMillis()
!udpPortOpened
false
true
panel.remove(play)
panel.repaint()
panel.add(pause)
initStreaming = true
sendData = "hello world".getBytes()
ip = InetAddress.getByName(serverIP)
DatagramPacket sendPacket = new DatagramPacket(sendData,sendData.length,ip,serverPort)
clientSocket.send(sendPacket)
Thread.sleep(10)
System.currentTimeMillis() - sendTime >=1000
false
true
return
Exit from method

Η μέθοδος streaming() περιέχει ένα νέο νήμα επεξεργασίας με όνομα t ώστε η αναπαραγωγή του βίντεο να γίνεται παράλληλα με τις υπόλοιπες λειτουργίες του applet. Στην μέθοδο run έχουμε ένα βρόχο while ο οποίος επαναλαμβάνεται όσο η τιμή της μεταβλητή είναι true. Αρχικά στον βρόχο δημιουργούμε ένα νέο udp πακέτο με όνομα receivePacket με την εξής εντολή receivePacket = new DatagramPacket(receiveData,0,receiveData.length). Έπειτα με την clientSocket.receive(receivePacket) περιμένουμε μέχρι να δεχτούμε δεδομένα από τον Server. Όσο η μέθοδος streaming() περιμένει να δεχτεί δεδομένα η μέθοδος streamingInitialize() στέλνει τα udp πακέτα στο Server. Την στιγμή που η μέθοδος streaming() δεχτεί τα δεδομένα από τον Server θα τα αποθηκεύσει στον πίνακα receiveData αφού τον έχουμε περάσει σαν παράμετρο στο udp πακέτο που έχουμε φτιάξει παραπάνω και θα συνεχιστή η εκτέλεση του υπόλοιπου κώδικα μέσα στην μέθοδο streaming(). Η αμέσως επόμενη εντολή είναι η αλλαγή της τιμής της μεταβλητής udpPortOpened από false σε true. Με αυτό τον τρόπο σταματάει να επαναλαμβάνεται ο βρόχος while στην μέθοδο streamingInitialize(). Έπειτα αποθηκεύουμε τα δεδομένα σε ένα νέο πίνακα με το όνομα buff κα τα τοποθετούμε σε μία ροή δεδομένων με όνομα in. Ορίζουμε ένα νέο bufferedImage με όνομα image και με την image = ImageIO.read(in) δημιουργούμε ξανά το καρέ από την ροή δεδομένων και το αποθηκεύουμε στην μεταβλητή image. Στην συνέχεια εάν ο χρήστης θέλει να καταγράψει μία εικόνα στον υπολογιστή του από το βίντεο πατάει το κουμπί με όνομα TAKE PICTURE και η τιμή της μεταβλητής takePhoto γίνεται true. Με αυτό τον τρόπο ισχύει η συνθήκη μέσα στην if αποθηκεύεται το καρέ στην μεταβλητή photo που είναι τύπου bufferedImage και η τιμή της takePhoto ξαναγίνεται false. Εάν ο χρήστης θέλει να καταγράψει ένα βίντεο από το applet στον υπολογιστή του θα πρέπει να πατήσει το κουμπί START REC στο applet και έτσι η τιμή της μεταβλητής recLocally γίνεται true και καλείται η μέθοδος recVideo(image). Η καταγραφή βίντεο και εικόνας θα εξηγηθεί λεπτομερώς στην συνέχεια του κεφαλαίου. Τέλος με την label.repaint() σβήνουμε το περιεχόμενο του label και εάν η μεταβλητή image δεν είναι κενή τότε σχεδιάζουμε το νέο καρέ στο label με την label.setIcon(new ImageIcon(image)) για να το εμφανίσουμε στην οθόνη. Label στην java είναι μία περιοχή εμφάνισης μικρού κειμένου, εικόνα η και τα δύο μαζι. Η μέθοδος με το διάγραμμα ροής φαίνονται παρακάτω:

```
public void streaming(){
                Thread t = new Thread(new Runnable(){
                        @Override
                        public void run() {
                                while(Playing){
                                 receivePacket = new
DatagramPacket(receiveData,0,receiveData.length);
                                        try {
                                                clientSocket.receive(receivePacket);
                                        } catch (IOException e) {
                                                e.printStackTrace();
                                        }
                                        udpPortOpened = true;
                                        byte[] buff = receivePacket.getData();
                                        InputStream in = new ByteArrayInputStream(buff);
                                        BufferedImage image = null;
                                        try {
                                                image = ImageIO.read(in);
                                        } catch (IOException e) {
                                                e.printStackTrace();
                                        }
```

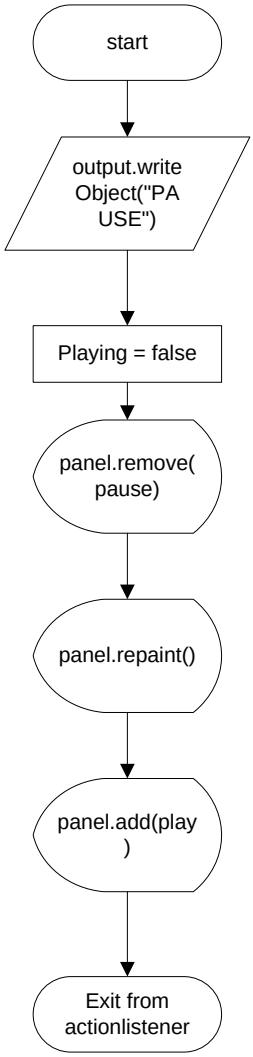
```
                                        if(takePhoto){
                                                photo = image;
                                                takePhoto = false;
                                        }
                                        if(recLocally){
                                                recVideo(image);
                                        }
                                        label.repaint();
                                        if(image != null)
                                        label.setIcon(new ImageIcon(image));
                                }
                        } });
t.start();  }
```


start
Thread t = new Thread(new Runnable()
Playing
false
Exit fom method
true
receivePacket = new DatagramPacket(receiveData,0,receiveData.length)
clientSocket .receive(receivePacket)
udpPortOpened = true
byte[] buff = receivePacket.getData()
InputStream in = new ByteArrayInputStream(buff)
BufferedImage image = null
image = ImageIO.read(in)
takePhoto
false
true
photo = image
takePhoto = false
recLocally
false
true
recVideo(image)
label.repaint()
image != null
false
true
label.setIcon(new ImageIcon(image))

Εάν ο χρήστης θέλει να παγώσει την μετάδοση της εικόνας από τον Server θα πρέπει να πατήσει το κουμπί pause στο applet έτσι ώστε να καλέσει τον αντίστοιχο ακροατή ενεργειών. Ο ακροατής ενεργειών με όνομα pause φαίνεται παρακάτω:

```
pause.addActionListener(new ActionListener(){
                @Override
                public void actionPerformed(ActionEvent e) {
                        try {
                                output.writeObject("PAUSE");
                                Playing = false;
                        } catch (IOException e1) {
                                e1.printStackTrace();
                        }
                        panel.remove(pause);
                        panel.repaint();
                        panel.add(play);
                }
        });
```

Αρχικά στέλνουμε το μήνυμα PAUSE στον Server με την output.writeObject("PAUSE") για να σταματήσει την αποστολή εικόνας στο applet. Έπειτα αλλάζουμε την τιμή της μεταβλητής Playing σε false για να σταματήσει η εκτέλεση του νήματος επεξεργασίας που βρίσκεται στην μέθοδο steaming(). Τέλος διαγράφουμε το κουμπί pause από το GUI του applet με την panel.remove(pause), κάνουμε μία ανανέωση στο GUI και προσθέτουμε το κουμπί play με την panel.add(play). Παρακάτω φαίνεται το διάγραμμα ροής του ακροατή ενεργειών:



Ο χρήστης έχει δύο επιλογές για να αποθηκεύσει τα βίντεο ή τις εικόνες που έχει καταγράψει. Η πρώτη είναι να τα αποθηκεύσει στον Server και η δεύτερη στον client δηλαδή στο υπολογιστή που τρέχει το applet. Αυτό μπορεί να το καθορίσει ο χρήστης επιλέγοντας στο ραδιόπληκτρο του applet την επιλογή Server ή Client. Επιλέγοντας τον Server καλείτε ο αντίστοιχος ακροατής ο οποίος φαίνεται παρακάτω:

```
Server.addActionListener(new ActionListener(){
                @Override
                public void actionPerformed(ActionEvent e) {
                        server = true;
                        client = false;

                }
});
```

Μέσα στον συγκεκριμένο ακροατή ενεργειών αλλάζουμε τις τιμές των μεταβλητών server σε true και client σε false. Το αντίθετο γίνεται όταν επιλέγουμε τον Client όπως φαίνεται παρακάτω:

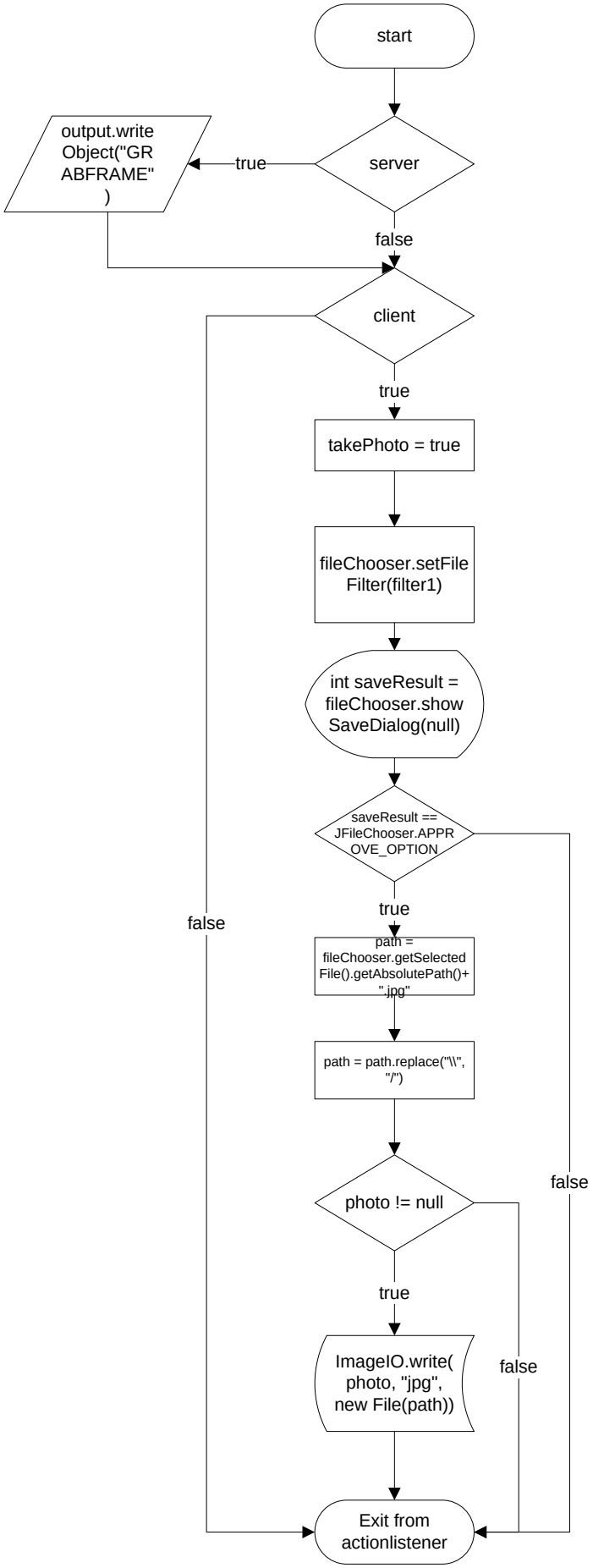
```
Client.addActionListener(new ActionListener(){
                @Override
```

```
                public void actionPerformed(ActionEvent e) {
                        client = true;
                        server = false;
                }
        });
```

Με το κουμπί TAKE PICTURE καλούμε τον αντίστοιχο ακροατή ενεργειών ο οποίος περιέχει τον κώδικα για την καταγραφή εικόνας στον client ή στον Server. Ο ακροατής ενεργειών με όνομα grabFrame φαίνεται παρακάτω:

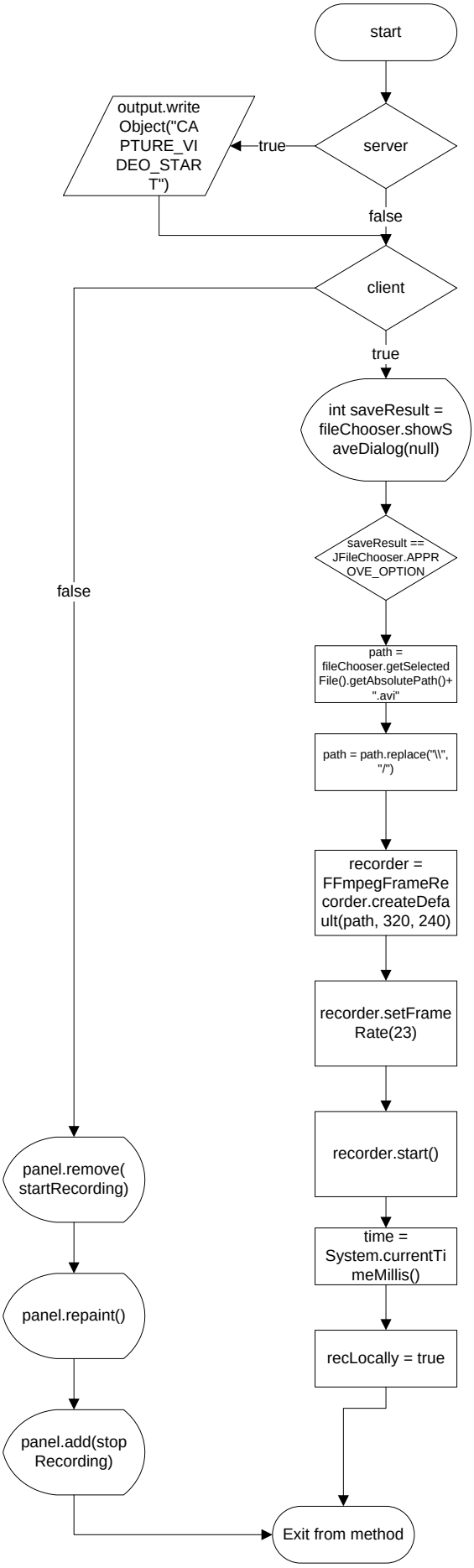
```
grabFrame.addActionListener(new ActionListener(){
                @Override
                public void actionPerformed(ActionEvent e) {
                        if(server){
                        try {
                                output.writeObject("GRABFRAME");
                        } catch (IOException e1) {
                                e1.printStackTrace();
                        }
                  }
                        if(client){
                                takePhoto = true;
                                fileChooser.setFileFilter(filter1);
                                int saveResult = fileChooser.showSaveDialog(null);
                                if(saveResult == JFileChooser.APPROVE_OPTION){
                        path = fileChooser.getSelectedFile().getAbsolutePath()+".jpg";
                                        path = path.replace("\\", "/");
                                        if(photo != null)
                                        try {
                        ImageIO.write(photo, "jpg", new File(path));
                                        } catch (IOException e1) {
                                                e1.printStackTrace();
                                        }
                                } } } });
```

Εάν η μεταβλητή server είναι true τότε στέλνουμε το μήνυμα GRABFRAME στον Server με την output.writeObject("GRABFRAME") και τερματίζεται η εκτέλεση του ακροατή ενεργειών grabFrame. Στην αντίθετη περίπτωση δηλαδή αν έχουμε επιλέξει τον client ακολουθείτε διαφορετική διαδικασία. Αρχικά αλλάζουμε την τιμή της μεταβλητής takePhoto σε true και θέτουμε ένα φίλτρο το όποιο μόλις ανοίξουμε το παράθυρο για να αποθηκεύσουμε την εικόνα μας εμφανίζει μόνο τα αρχεία με επέκταση jpg και jpeg. Το φίλτρο επεκτάσεων το έχουμε ορίσει στην μέθοδο init με τον εξής τρόπο filter1 = new FileNameExtensionFilter("JPEG Images","jpg","jpeg"). Έπειτα εμφανίζουμε με την int saveResult = fileChooser.showSaveDialog(null) το παράθυρο με το οποίο θα αποθηκεύσουμε την εικόνα. Εάν πατήσουμε οκ στο παράθυρο τότε αποθηκεύουμε την διαδρομή που έχουμε επιλέξει στην μεταβλητή path και αντικαθιστούμε τα σύμβολα \\ με / λόγο ότι με αυτόν τον τρόπο γίνεται η σύνταξη μιας διαδρομής δίσκου στην java. Τέλος αν η μεταβλητή photo δεν είναι κενή με την ImageIO.write(photo, "jpg", new File(path)) αποθηκεύουμε την εικόνα με επέκταση jpg και στη διαδρομή που έχουμε επιλέξει. Παρακάτω φαίνεται το διάγραμμα ροής του ακροατή ενεργειών:


start
server
true
output.write Object("GR ABFRAME")
false
client
true
takePhoto = true
fileChooser.setFile Filter(filter1)
int saveResult = fileChooser.show SaveDialog(null)
saveResult == JFileChooser.APPR OVE_OPTION
true
false
path = fileChooser.getSelected File().getAbsolutePath()+ ".jpg"
path = path.replace("\\", "/")
photo != null
false
true
ImageIO.write(photo, "jpg", new File(path))
false
Exit from actionlistener

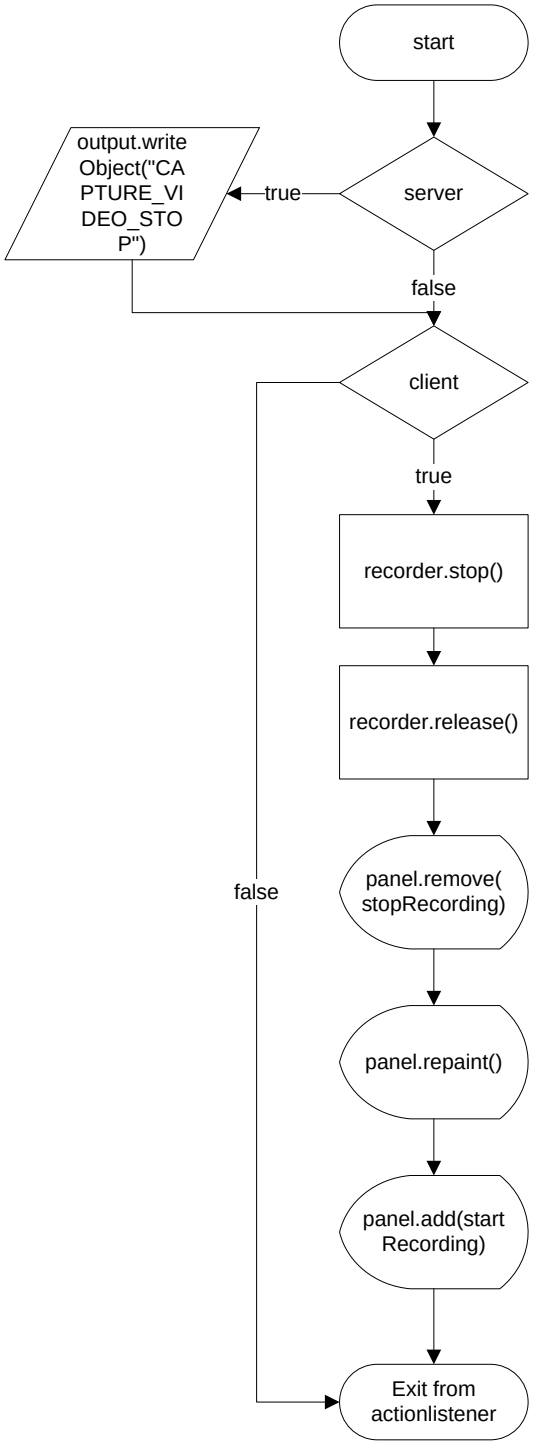
Εάν ο χρήστης θέλει να καταγράψει βίντεο τότε θα πρέπει να πατήσει το κουμπί με όνομα START REC. Αν έχει επιλέξει server τότε στέλνουμε το μήνυμα CAPTURE_VIDEO_START στον Server με την output.writeObject("CAPTURE_VIDEO_START") και εμφανίζουμε το κουμπί STOP REC με τον ίδιο τρόπο όπως κάναμε για τα προηγούμενα. Στην περίπτωση που έχει επιλέξει τον client εμφανίζουμε το παράθυρο διαλόγου με το οποίο θα επιλέξουμε την διαδρομή για την αποθήκευση του βίντεο με τον τρόπο που είδαμε στην περίπτωση της καταγραφής εικόνας στον client. Έπειτα αποθηκεύουμε την διαδρομή με τον ίδιο τρόπο. Στην συνέχεια ορίζουμε έναν εγγραφέα βίντεο με τον εξής τρόπο recorder = FFmpegFrameRecorder.createDefault(path, 320, 240). Περνάμε σαν παραμέτρους την διαδρομή που θέλουμε να αποθηκεύσουμε το βίντεο και τις διαστάσεις του βίντεο οι οποίες είναι 320 μήκος και 240 πλάτος. Έπειτα θέτουμε τον ρυθμό των καρέ σε 23 ανά δευτερόλεπτο με την recorder.setFrameRate(23). Εκκινούμε τον εγγραφέα βίντεο με την recorder.start(), καταγράφουμε την ώρα του συστήματος στην μεταβλητή time και αλλάζουμε την τιμή της μεταβλητής recLocally σε true. Τέλος εμφανίζουμε το κουμπί pause. Ο ακροατής ενεργειών με το διάγραμμα ροής φαίνονται παρακάτω:

```
startRecording.addActionListener(new ActionListener(){
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                if(server){
                                        try {
                                        output.writeObject("CAPTURE_VIDEO_START");
                                        } catch (IOException e1) {
                                                e1.printStackTrace();
                                        }
                                }
                                if(client){
                                int saveResult = fileChooser.showSaveDialog(null);
                                if(saveResult == JFileChooser.APPROVE_OPTION){
                                path = fileChooser.getSelectedFile().getAbsolutePath()+".avi";
                                                path = path.replace("\\", "/");
                                                try {
                        recorder = FFmpegFrameRecorder.createDefault(path, 320, 240);
                                                recorder.setFrameRate(23);
                                                   recorder.start();
                                                   time = System.currentTimeMillis();
                                                        recLocally = true;
                                                        } catch (Exception e1) {
                                                                e1.printStackTrace();
                                                        }
                                                }       }
                                panel.remove(startRecording);
                                panel.repaint();
                                panel.add(stopRecording);
                        } });
```


start
server
true
output.write Object("CAPTURE_VIDEO_START")
false
client
true
int saveResult = fileChooser.showSaveDialog(null)
saveResult == JFileChooser.APPROVE_OPTION
path = fileChooser.getSelectedFile().getAbsolutePath()+".avi"
path = path.replace("\\", "/")
recorder = FFmpegFrameRecorder.createDefault(path, 320, 240)
recorder.setFrameRate(23)
recorder.start()
time = System.currentTimeMillis()
recLocally = true
false
panel.remove(startRecording)
panel.repaint()
panel.add(stopRecording)
Exit from method

Με το κουμπί STOP REC καλείτε ο ακροατής ενεργειών stopRecording ο οποίος περιέχει τον κώδικα για την διακοπή της εγγραφής βίντεο. Αν έχει επιλεγεί ο Server για την αποθήκευση του βίντεο τότε στέλνουμε το μήνυμα CAPTURE_VIDEO_STOP με την output.writeObject("CAPTURE_VIDEO_STOP") και εμφανίζουμε το κουμπί START REC για την καταγραφή νέου βίντεο. Στην περίπτωση που έχει επιλεγεί ο client για την αποθήκευση του βίντεο αλλάζουμε την τιμή της recLocally σε false, σταματάμε την εγγραφή με την recorder.stop() και απελευθερώνουμε τους πόρους του συστήματος από τον εγγραφέα βίντεο. Τέλος εμφανίζουμε το κουμπί START REC στην θέση του STOP REC. Ο ακροατής ενεργειών και το διάγραμμα ροής φαίνονται παρακάτω:

```
stopRecording.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent e) {
            if(server){
                try {
                output.writeObject("CAPTURE_VIDEO_STOP");
                } catch (IOException e1) {
                    e1.printStackTrace();
                }
            }
            if(client){
                recLocally = false;
                try {
                    recorder.stop();
                    recorder.release();
                } catch (Exception e1) {
                    e1.printStackTrace();
                }
            }
            panel.remove(stopRecording);
            panel.repaint();
            panel.add(startRecording);
        } });
```



Η μέθοδος που είναι υπεύθυνη για την εγγραφή βίντεο είναι η παρακάτω:

```
public void recVideo(BufferedImage buffImg){
        try {
                        recorder.record(IplImage.createFrom(buffImg));
                } catch (Exception e) {
                        e.printStackTrace();
                } } }
```

Περνάμε σαν παράμετρο στην μέθοδο το καρέ κάθε φορά που την καλούμε και με την recorder.record(IplImage.createFrom(buffImg)) γράφουμε στο βίντεο το καρέ. Για να στρίψει ο χρήστης την κάμερα αριστερά θα πρέπει να πατήσει το κουμπί left και με αυτόν τον τρόπο καλείτε ο αντίστοιχος ακροατής ενεργειών ο οποίος φαίνεται παρακάτω:

```
left.addActionListener(new ActionListener(){
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                if(!turning)
HttpUtilityTester.turnCamera("http://195.251.14.211/forms/ILON%201000/ilon%201000%20IMAGE/Web/forms/camera.html?ILONWEB_URL=%2Fforms%2Fpantilt11.html&NVL_panL=1.0+1");
                                turning = true;
                                stopLeft = true;
                        } });
```

Εάν η κάμερα δεν στρίβει εκείνη την στιγμή δηλαδή η τιμή της μεταβλητής είναι false τότε με τη μέθοδο turnCamera() που ανήκει στη κλάση HttpUtilityTester κάνουμε ένα http request στον Server στο αντίστοιχο url που έχουμε περάσει σαν παράμετρο. Τέλος αλλάζουμε της τιμές των μεταβλητών turning σε true και stopLeft σε true. Η ίδια διαδικασία γίνεται και με τις υπόλοιπες κατευθύνσεις. Για να σταματήσουμε την κάμερα την στιγμή που στρίβει τότε πατάμε το κουμπί stop. Ο αντίστοιχος ακροατής ενεργειών του stop είναι ο παρακάτω:

```
stop.addActionListener(new ActionListener(){
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                if(stopLeft){
HttpUtilityTester.turnCamera("http://195.251.14.211/forms/ILON%201000/ilon%201000%20IMAGE/Web/forms/camera.html?ILONWEB_URL=%2Fforms%2Fpantilt11.html&NVL_panL=0.0+0");
                                        turning = false;
                                        stopLeft = false;
                                }else if(stopRight){
HttpUtilityTester.turnCamera("http://195.251.14.211/forms/ILON%201000/ilon%201000%20IMAGE/Web/forms/camera.html?ILONWEB_URL=%2Fforms%2Fpantilt11.html&NVL_panR=0.0+0");
                                        turning = false;
                                        stopRight = false;
                                }else if(stopUp){
HttpUtilityTester.turnCamera("http://195.251.14.211/forms/ILON%201000/ilon%201000%20IMAGE/Web/forms/camera.html?ILONWEB_URL=%2Fforms%2Fpantilt11.html&NVL_tiltUp=0.0+0");
                                        turning = false;
                                        stopUp = false;
                                }else if(stopDown){
HttpUtilityTester.turnCamera("http://195.251.14.211/forms/ILON%201000/ilon%201000%20IMAGE/Web/forms/camera.html?ILONWEB_URL=%2Fforms%2Fpantilt11.html&NVL_tiltD=0.0+0");
                                        turning = false;
                                        stopDown = false;
                                } } });
```

Εάν πατήσουμε το κουμπί stop και εκείνη την στιγμή η κάμερα στρίβει αριστερά τότε με την HttpUtilityTester.turnCamera("http://195.251.14.211/forms/ILON%201000/ilon%201000%20IMAGE/Web/forms/camera.html?ILONWEB_URL=%2Fforms%2Fpantilt11.html&NVL_panL=0.0+0") κάνουμε ένα http request στον iLon και σταματάει η αλλαγή κατεύθυνσης. Τέλος αλλάζουμε τις τιμές των μεταβλητών turning σε false και stopLeft σε false. Το ίδιο γίνεται και στις υπόλοιπες περιπτώσεις αλλάζοντας πάντα την τιμή της turning σε false και των αντίστοιχων μεταβλητών σε false. Για να επιλέξουμε την ημερομηνία έναρξης μίας χρονοπρογραμματισμένης εγγραφής βίντεο κάνουμε κλικ στο κενό πεδίο το οποίο βρίσκεται κάτω από το label με όνομα Start Date. Κάνοντας κλικ εμφανίζεται ένα αναδυόμενο ημερολόγιο από το οποίο επιλέγουμε την ημερομηνία έναρξης και με αυτόν τον τρόπο καλείτε ο αντίστοιχος ακροατής αλλαγών με όνομα dateChooserStart ο οποίος φαίνεται παρακάτω:

```
dateChooserStart.addPropertyChangeListener(new java.beans.PropertyChangeListener() {
        public void propertyChange(java.beans.PropertyChangeEvent evt) {
           dateStartOnlyPopupChanged(evt);  } });
```

Ο συγκεκριμένος ακροατής αλλαγών καλεί την μέθοδο dateStartOnlyPopupChanged και περνάει σαν παράμετρο στην μέθοδο την αλλαγή που έγινε. Η μέθοδος dateStartOnlyPopupChanged φαίνεται παρακάτω:

```
private void dateStartOnlyPopupChanged(java.beans.PropertyChangeEvent evt) {
   if (evt.getNewValue() instanceof Date)
      setDateStart((Date)evt.getNewValue());
}
```

Εάν η αλλαγή που έγινε είναι στιγμιότυπο της κλάσης Date τότε καλούμε την μέθοδο setDateStart((Date)evt.getNewValue()) περνώντας σαν παράμετρο την αλλαγή της ημερομηνίας. Η μέθοδος setDateStart((Date)evt.getNewValue()) φαίνεται παρακάτω:

```
public void setDateStart(Date date)
{
   dayofMonthString = "";
   monthString = "";
   yearString = "";
   if (date != null){
     dayofMonthString = dayofMonth.format(date);
     monthString = month.format(date);
     yearString = year.format(date);
   }
}
```

Αρχικά στην μέθοδο δηλώνουμε τρείς μεταβλητές τύπου string οι οποίες είναι οι εξής: dayofMonthString, monthString και yearString. Έπειτα αν η μεταβλητή date δεν είναι κενή αποθηκεύουμε την ημέρα στην μεταβλητή dayofMonthString με την dayofMonthString = dayofMonth.format(date), το μήνα στην μεταβλητή monthString με την monthString = month.format(date) και τον χρόνο στη μεταβλητή yearString με την yearString = year.format(date). Για να επιλέξουμε την ημερομηνία λήξης της εγγραφής βίντεο κάνουμε κλικ στο κενό πεδίο κάτω από το label End Date και καλείτε ο ακροατής αλλαγών dateChooserEnd. Σε αυτό τον ακροατή αλλαγών καλούμε τη μέθοδο dateEndtOnlyPopupChanged(evt) περνώντας σαν παράμετρο την αλλαγή που έγινε. Ο ακροατής αλλαγών με όνομα dateChooserEnd φαίνεται παρακάτω:

```
dateChooserEnd.addPropertyChangeListener(new java.beans.PropertyChangeListener() {
        public void propertyChange(java.beans.PropertyChangeEvent evt) {
           dateEndtOnlyPopupChanged(evt);
        }
     });
```

Η μέθοδος dateEndtOnlyPopupChanged(evt) φαίνεται παρακάτω:

```
private void dateEndtOnlyPopupChanged(java.beans.PropertyChangeEvent evt) {
   if (evt.getNewValue() instanceof Date)
      setDateEnd((Date)evt.getNewValue());
}
```

Εάν η αλλαγή που έγινε είναι στιγμιότυπο της κλάσης Date τότε καλούμε την μέθοδο setDateEnd((Date)evt.getNewValue()) και περνάμε σαν παράμετρο την ημερομηνία. Η μέθοδος setDateEnd((Date)evt.getNewValue()) φαίνεται παρακάτω:

```
public void setDateEnd(Date date)
{
   dateStringEnd = "";
   if (date != null)
                dateStringEnd = dateFormat.format(date);
}
```

Αρχικά δηλώνουμε μια μεταβλητή με όνομα dateStringEnd και τύπου string. Έπειτα αν η μεταβλητή date που περάσαμε σαν παράμετρο στην μέθοδο δεν είναι κενή τότε αποθηκεύουμε την ημερομηνία στην μεταβλητή dateStringEnd. Για να ορίσουμε ώρα έναρξης και διακοπής της εγγραφής συμπληρώνουμε την ώρα και τα λεπτά τα πεδία που περιέχουν μηδέν κάτω από τις αντίστοιχες ημερομηνίες έναρξής και λήξης. Κάνοντας κλικ στο κουμπί Schedule Job καλούμε τον αντίστοιχο ακροατή ενεργειών με όνομα scheduleJob ο οποίος φαίνεται παρακάτω:

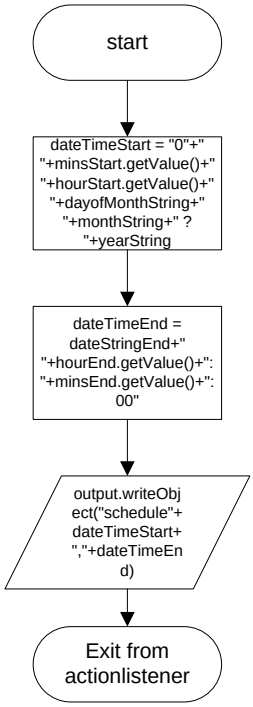
```
scheduleJob.addActionListener(new ActionListener(){
                        @Override
                        public void actionPerformed(ActionEvent e) {
dateTimeStart = "0"+" "+minsStart.getValue()+" "+hourStart.getValue()+"
"+dayofMonthString+" "+monthString+" ? "+yearString;
 dateTimeEnd = dateStringEnd+" "+hourEnd.getValue()+":"+minsEnd.getValue()+":00";
                    try {
                        output.writeObject("schedule"+dateTimeStart+","+dateTimeEnd);
                        } catch (IOException e1) {
                                e1.printStackTrace();
                        } } });
```

Αρχικά στην μεταβλητή dateTimeStart συνθέτουμε την ημερομηνία και ώρα έναρξης της εγγραφής. Η συμβολοσειρά από τα αριστερά προς τα δεξιά είναι η ακόλουθη: δευτερόλεπτα, κενό, λεπτά, κενό, ώρες, κενό, ημέρα του μήνα, κενό, μήνας, κενό, ερωτηματικό, κενό, έτος. Οι τιμές που μπορούν να πάρουν οι μεταβλητές φαίνονται στον παρακάτω πίνακα:

Όνομα	**Επιτρεπόμενες τιμές**	**Επιτρεπόμενοι ειδικοί χαρακτήρες**
Δευτερόλεπτα	0-59	
Λεπτά	0-59	

Ώρες	0-23	
Ημέρα του μήνα	1-31	
Μήνας	0-11 ή Ιανουάριος- Φεβρουάριος	
Ημέρα της εβδομάδας	1-7 ή Δευτέρα – Παρασκευή	?
Έτος	Κενό ή 1970-2099	

Το ερωτηματικό σημαίνει ότι αφήνουμε το πεδίο ημέρα της εβδομάδας κενό επειδή το προσδιορίσαμε στο πεδίο ημέρα του μήνα. Στην μεταβλητή dateTimeEnd συνθέτουμε την ημερομηνία και ώρα λήξης της προγραμματισμένης εγγραφής. . Η συμβολοσειρά από τα αριστερά προς τα δεξιά είναι η ακόλουθη: ημερομηνία, κενό, ώρα, κενό, λεπτά, κενό δευτερόλεπτα. Η μορφή της συμβολοσειράς που στέλνουμε στον Server είναι η εξής: schedule"+dateTimeStart+","+dateTimeEnd. Η αποστολή της συμβολοσειράς γινεται με την εντολή output.writeObject("schedule"+dateTimeStart+","+dateTimeEnd). Το διάγραμμα ροής του ακροατή ενεργειών φαίνεται παρακάτω:



Η μέθοδος stop φαίνεται παρακάτω:

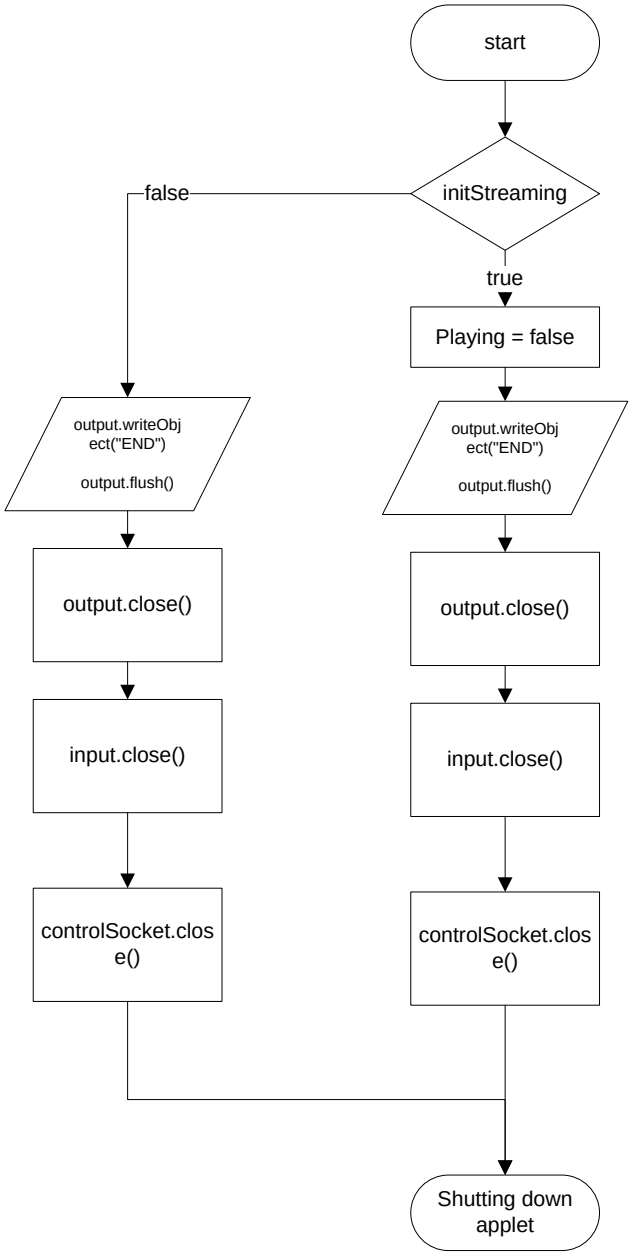
```
public void stop(){
                reading = false;
                HttpUtility.disconnect();
        }
```

Αρχικά στην μέθοδο αλλάζουμε την τιμή της μεταβλητή reading σε false και στην συνέχεια αποσυνδεόμαστε απο την σύνδεση που έχουμε δημιουργήση μεταξή του applet και του iLon για να μπορούμε να διαβάζουμε τις τιμές από του αισθητήρες. Η μέθοδος destroy φαίνεται παρακάτω:

```
public void destroy(){
                if(initStreaming){
```

```
            Playing = false;
        try {
            output.writeObject("END");
            output.flush();
        } catch (IOException e1) {
            e1.printStackTrace();
        }
        try {
            output.close();
            input.close();
            controlSocket.close();
        } catch (IOException e1) {
            e1.printStackTrace();
        }
        }else{
            try {
                output.writeObject("END");
                output.flush();
            } catch (IOException e1) {
                e1.printStackTrace();
            }
            try {
                output.close();
                input.close();
                controlSocket.close();
            } catch (IOException e1) {
                e1.printStackTrace();
            } } }
```

Αν έχει αρχικοποιηθεί η μετάδοση βίντεο τότε η τιμή της μεταβλητής initStreaming θα είναι true και κατά πάσα πιθανότητα και η τιμή της μεταβλητής Playing θα είναι true. Έτσι αλλάζουμε την Playing από true σε false για να τερματιστεί η λειτουργία του νήματος επεξεργασίας που αφορά την λήψη βίντεο από τον Server. Έπειτα στέλνουμε το μήνυμα END στον Server και με τις εντολές output.close() input.close() controlSocket.close() κλείνουμε την συνδέσει που αφορά την επικοινωνία με το TCP πρωτόκολλο. Στην αντίθετη περίπτωση επαναλαμβάνουμε την προηγούμενη διαδικασία με την διαφορά να είναι ότι τώρα δεν θα αλλάξουμε την τιμή της Playing σε false. Το διάγραμμα ροής της μεθόδου destroy φαίνεται παρακάτω:



3.3 Η κλάση HttpUtility Tester

Η κλάση HttpUtilityTester είναι ίδια με την HttpUtility του Server εκτός από δύο διαφορές. Η πρώτη διαφορά είναι στην μέθοδο sendGetRequest και συγκεκριμένα στο βρόχο for ο οποίος περιέχει μία επιπλέον σύγκριση η οποία είναι η παρακάτω:

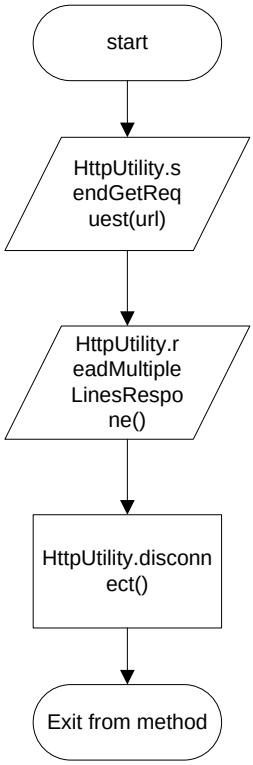
```
if(response[i].contains("OC_OCCUPIED")){
                human_presence = "OC_OCCUPIED";
        }else if(response[i].contains("OC_UNOCCUPIED")){
                human_presence = "OC_UNOCCUPIED";}
```

Εάν η απάντηση του Server περιέχει την συμβολοσειρά OC_OCCUPIED τότε την αποθηκεύουμε στην μεταβλητή human_presence. Στην περίπτωση που η απάντηση του Server περιέχει την συμβολοσειρά OC_UNOCCUPIED αποθηκεύουμε αυτήν στη μεταβλητή human_presence. Η σύγκριση αυτή έχει σαν σκοπό να βρούμε αν ο αισθητήρας του iLon

αντιλαμβάνεται ανθρώπινη παρουσία στον χώρο. Η δεύτερη διαφορά στην κλάση είναι ότι έχουμε άλλη μια μέθοδο με όνομα turnCamera της οποίας η λειτουργία είναι ο έλεγχος της κάμερας. Η μέθοδος turnCamera(String url) φαίνεται παρακάτω:

```
public static void turnCamera(String url) {
      // sending GET request
      try {
         HttpUtility.sendGetRequest(url);
         HttpUtility.readMultipleLinesRespone();
      } catch (IOException ex) {
      }
    HttpUtility.disconnect();
  }
```

Αρχικά στην μέθοδο κάνουμε ένα http get request στο url που έχουμε περάσει σαν παράμετρο και έπειτα διαβάζουμε την απάντηση του Server. Τέλος διακόπτουμε την σύνδεση μεταξύ iLon και applet. Το διάγραμμα ροής φαίνεται παρακάτω:



3.4 Η κλάση HttpUtility

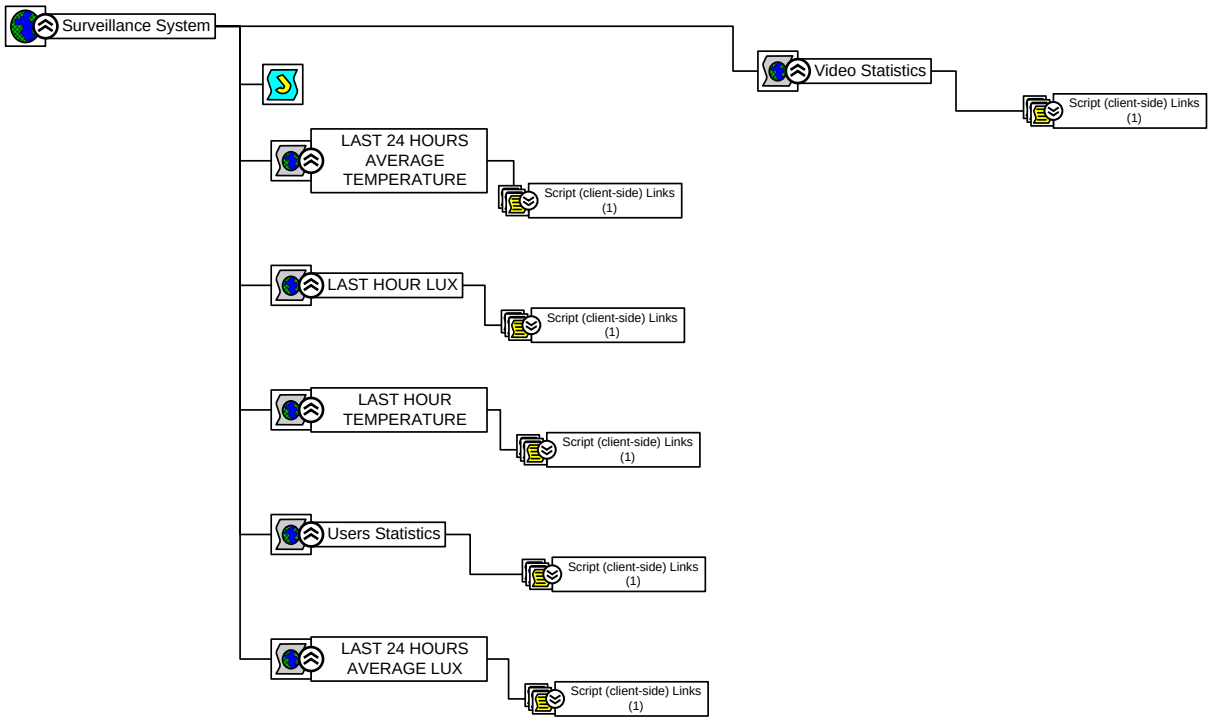
Η κλάση HttpUtility είναι ίδια με την κλάση HttpUtility στον Server και δεν χρειάζεται περεταίρω ανάλυση.

Κεφάλαιο 4

4.1 Εισαγωγή

Στο κεφάλαιο αυτό γίνεται περιγραφή σχεδίασης των βάσεων δεδομένων και της ιστοσελίδας και εξήγηση της κάθε λειτουργίας της.

4.2 Η ιστοσελίδα σε μορφή διαγράμματος



4.3 Η ιστοσελίδα Surveillance System

Η ιστοσελίδα Surveillance System περιέχει το Java applet καθώς και ιστοσελίδες που μπορούμε να εξάγουμε στατιστικά διαγράμματα στοιχείων από την βάση δεδομένων. Με την ετικέτα <applet> ενσωματώνουμε ένα java applet στην ιστοσελίδα. Με την ιδιότητα code μέσα στην ετικέτα <applet> αναφερόμαστε στην κλάση η οποία περιέχει τις μεθόδους init, start, stop και destroy. Οι ιδιότητες width και height αναφέρονται αντίστοιχα στο μήκος και στο ύψος του applet. Η ιδιότητα archive αναφέρεται στην τοποθεσία του αρχείου jar το οποίο περιέχει τις κλάσεις του applet. Με την ετικέτα ορίζουμε την τοποθεσία ενός αντικειμένου στην ιστοσελίδα και με την ιδιότητα align="left" τοποθετούμε το αντικείμενο στο αριστερό μέρος της σελίδας. Η ετικέτα <a> ορίζει μια υπερ-σύνδεση, η οποία χρησιμοποιείται για τη σύνδεση από τη μία σελίδα στην άλλη. Με την ιδιότητα href καθορίζουμε τη διεύθυνση της σελίδας στην οποία θέλουμε να μεταφερθούμε. Με την ιδιότητα target καθορίζουμε που θέλουμε να ανοίξουμε την νέα σελίδα. Στην περίπτωση μας η νέα σελίδα ανοίγει σε μία νέα καρτέλα του browser επειδή η ιδιότητα target έχει την τιμή blank. Τέλος κάνοντας αριστερό κλικ στο κείμενο Video Statistics μεταφερόμαστε στην σελίδα από την οποία μπορούμε να δούμε τον αριθμό των βίντεο που έχουν καταγράψει οι χρήστες. Η παραπάνω διαδικασία είναι η ίδια για τις υπόλοιπες ιστοσελίδες. Ο κώδικας της σελίδας:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Surveillance System</title>
</head>
<body>
<div align="left"><a href="Video Statistics.php" target="_blank">Video Statistics</a></div>
<div align="left"><a href="UsersStatistics.php" target="_blank">Users Statistics</a></div>
<div align="left"><a href="lastHourTemp.php" target="_blank">Last Hour Temp</a></div>
<div align="left"><a href="last24hours.php" target="_blank">Last 24 hours
Temperature</a></div>
<div align="left"><a href="last24hoursLux.php" target="_blank">Last 24 Hours
Lux</a></div>
<a href="lastHourLux.php" target="_blank">Last Hour Lux</a>
<center>
<applet code="Receiver.class" width="650" height="650" archive="files.jar">
</applet>
</center>
</body>
</html>
```

4.4 Η ιστοσελίδα Users Statistics

Η ιστοσελίδα Users Statistics ενσωματώνει ένα php script με το οποίο εμφανίζουμε ένα στατιστικό διάγραμμα που μας δείχνει την συχνότητα σύνδεσης των χρηστών στην εφαρμογή. Αρχικά δηλώνουμε τρεις μεταβλητές με τα στοιχεία της βάσης δεδομένων. Με την συνάρτηση mysql_connect($dbhost, $dbuser, $dbpass) συνδεόμαστε με την βάση δεδομένων. Με την συνάρτηση mysql_select_db('ptixiaki') συνδεόμαστε με τον πίνακα με το όνομα ptixiaki. Έπειτα με την συνάρτηση $FC = new FusionCharts("Pie3D","500","500") ορίζουμε το διάγραμμα το οποίο είναι τύπου πίτας και έχει μήκος και ύψος 500 pixel. Με την συνάρτηση $strParam="Caption=Incidence of users connections; subCaption=; showBorder=1" ορίζουμε τον τίτλο του διαγράμματος και με την $FC->setChartParams($strParam) θέτουμε στο διάγραμμα τον τίτλο. Στην συνέχεια στην μεταβλητή $sql ορίζουμε το query που θα κάνουμε στην βάση δεδομένων. Διαλέγουμε τα πεδία id και Username από τον πίνακα users και ενώνουμε το πεδίο id του πίνακα users με το πεδίο userId του πίνακα videos και τα ταξινομούμε με βάση το πεδίοUsername του πίνακα users. Με την συνάρτηση $result = mysql_query($sql, $conn) κάνουμε το query στην βάση δεδομένων και αποθηκεύουμε τα αποτελέσματα στην μεταβλητή $result. Ορίζουμε ένα πίνακα με όνομα $arrayConnections και ένα δείκτη $index ίσο με μηδέν. Έπειτα με τον βρόχο while αποθηκεύουμε τα αποτελέσματα από την μεταβλητή $result στον πίνακα $arrayConnections. Στην μεταβλητή $sql ορίζουμε το νέο query με το οποίο διαλέγουμε το πεδίο Username από τον πίνακα users και με την συνάρτηση mysql_query($sql,$conn) κάνουμε το query στην βάση δεδομένων. Στην συνέχεια ορίζουμε δύο νέους πίνακες με όνομα $arrayUsername, $arrData και θέτουμε τη τιμή του δείκτη ίση με το μηδέν. Με τον βρόχο while αποθηκεύουμε τα αποτελέσματα από το query που κάναμε προηγουμένως στον πίνακα arrayUsernames. Έπειτα ορίζουμε τον μετρητή $times που μας δείχνει πόσες φορές συνδέθηκε ένας χρήστης ίσο με το μηδέν. Για να βρούμε πόσες φορές έχει συνδεθεί κάθε χρήστης χρησιμοποιούμε δύο βρόχους for. Στον πρώτο βρόχο for έχουμε πρόσβαση κάθε

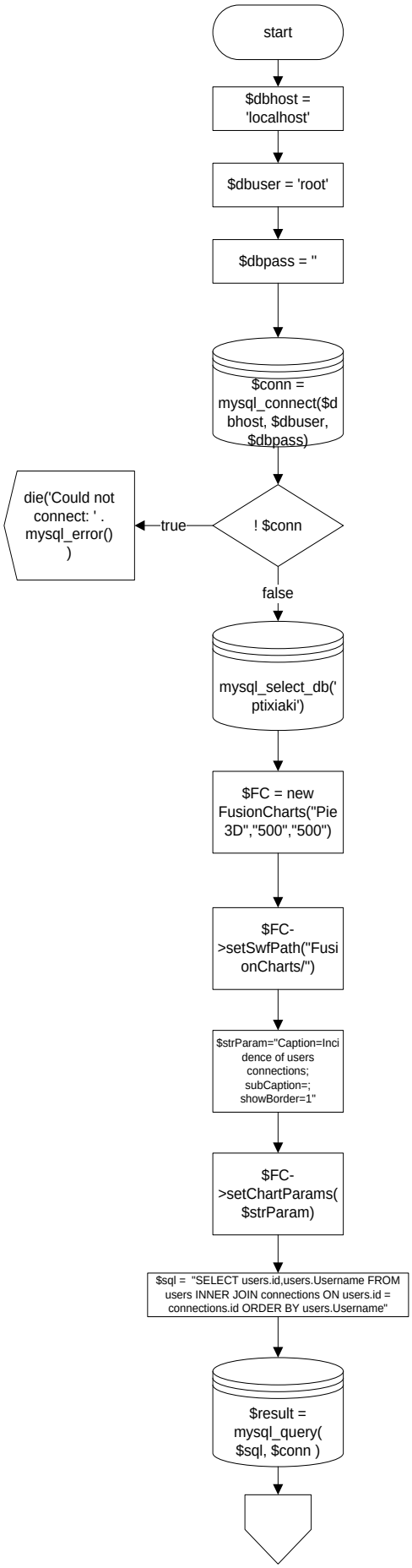
φορά στο όνομα κάθε χρήστη που υπάρχει στον πίνακα users, ενώ στο δεύτερο βρόχο έχουμε πρόσβαση στα ονόματα χρηστών που έχουν συνδεθεί στην εφαρμογή. Εάν ένας χρήστης με όνομα π.χ. Πέτρος έχει συνδεθεί τρείς φορές στην εφαρμογή τότε το όνομά του θα είναι καταχωρημένο στον πίνακα $arrayConnections τρείς φορές. Άρα όσες φορές υπάρχει το όνομα κάθε χρήστη στον πίνακα $arrayConnections[$j] τόσες φορές αυξάνουμε τον μετρητή $times++. Κάθε φορά που έχουμε συγκρίνει όλα τα περιεχόμενα του πίνακα $arrayConnections αποθηκεύουμε το όνομα χρήστη που συγκρίνουμε εκείνη την στιγμή στον πίνακα $arrData[$index][0] και τον μετρητή $times στην θέση $arrData[$index][1]. Στην συνέχεια μηδενίζουμε τον μετρητή $times, αυξάνουμε τον δείκτη $index++ κατά ένα και επαναλαμβάνουμε την διαδικασία για όλα τα στοιχεία του πίνακα $arrayUsernames. Τέλος κλείνουμε την σύνδεση μεταξύ ιστοσελίδας και βάσης δεδομένων με την συνάρτηση mysql_close($conn), προσθέτουμε τα δεδομένα στο διάγραμμα με την συνάρτηση $FC->addChartDataFromArray($arrData) και το εμφανίζουμε στην οθόνη με την $FC->renderChart(). Ο κώδικας της σελίδας:

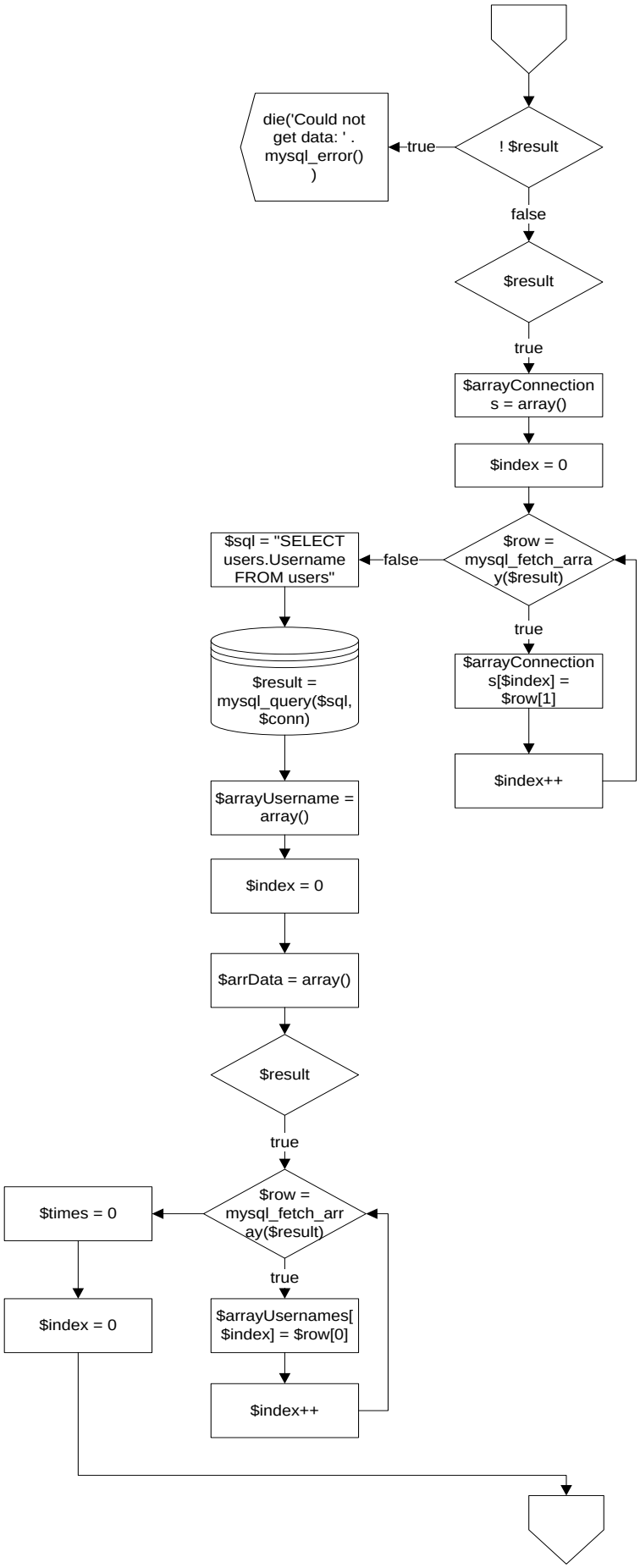
```
<?php
	//We have included ../Includes/FusionCharts_Gen.php, which contains
	//FusionCharts PHP Class to help us easily embed charts
	//We have also used ../Includes/DBConn.php to easily connect to a database.
	include("Includes/FusionCharts_Gen.php");
	include("Includes/DBConn.php");
?>
<HTML>
	<HEAD>
		<TITLE>
			Users Statistics
		</TITLE>
		<?php
 //You need to include the following JS file, if you intend to embed the chart using JavaScript.
//Embedding using JavaScript avoids the "Click to Activate..." issue in Internet Explorer
//When you make your own charts, make sure that the path to this JS file is correct.
//Else, you will JavaScript errors.
		?>
		<SCRIPT LANGUAGE="Javascript" SRC="FusionCharts/FusionCharts.js">
		</SCRIPT>
	</HEAD>
	<BODY>
		<CENTER>
		<?php
		    //In this example, we show how to connect FusionCharts to a database.
		    //For the sake of ease, we have used an MySQL databases containing two tables.
		    // Connect to the Database
		      $dbhost = 'localhost';
		      $dbuser = 'root';
		      $dbpass = '';
		      $conn = mysql_connect($dbhost, $dbuser, $dbpass);
		      if(! $conn )
		      {
		      die('Could not connect: ' . mysql_error());
```

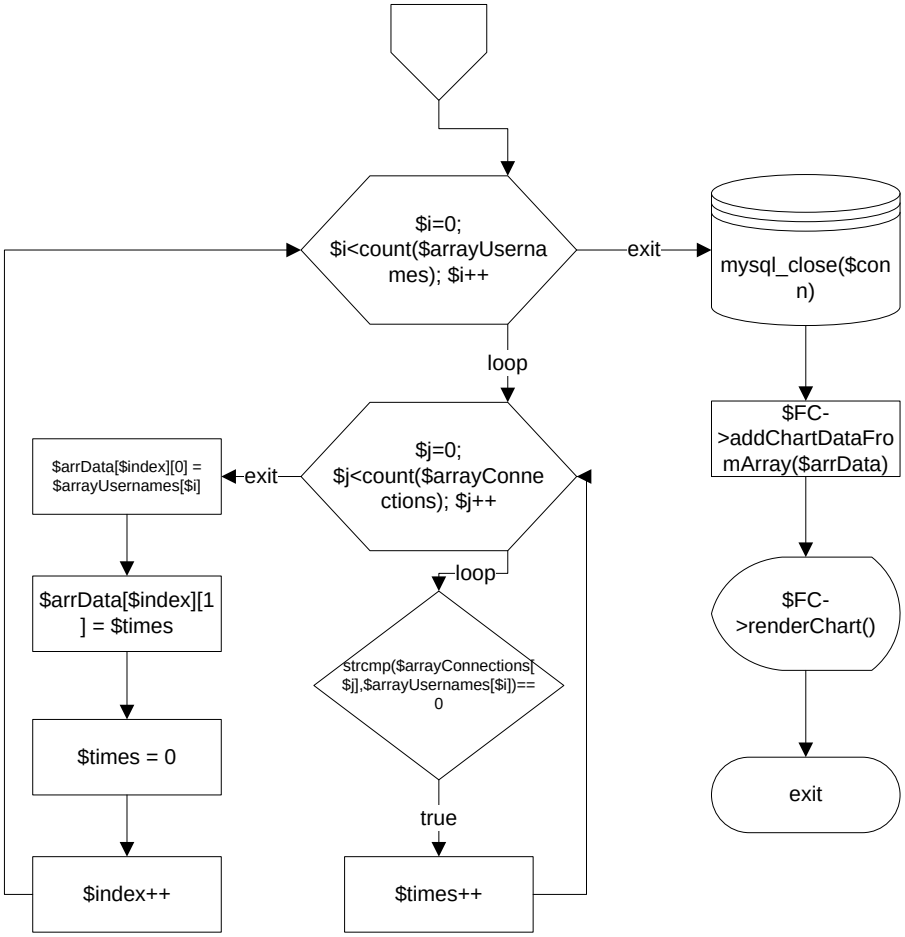
```
                }
                mysql_select_db('ptixiaki');
                //  Create pie 3d chart object using FusionCharts PHP Class
                $FC = new FusionCharts("Pie3D","500","500");
                // Set Relative Path of chart SWF file.
                $FC->setSwfPath("FusionCharts/");
                // Define chart attributes
           $strParam="Caption=Incidence of users connections; subCaption=; showBorder=1"
                //Set chart attributes
                $FC->setChartParams($strParam);
$sql =  "SELECT users.id,users.Username FROM users INNER JOIN connections ON
users.id = connections.id ORDER BY users.Username";
                $result = mysql_query( $sql, $conn );
                if(! $result )
                {
                die('Could not get data: ' . mysql_error());
                }
//Pass the SQL Query result to the FusionCharts PHP Class function
//along with field/column names that are storing chart values and corresponding category
names
                //to set chart data from database
                if ($result)
                {       $arrayConnections = array();
               $index = 0;
                while($row =  mysql_fetch_array($result)){
                $arrayConnections[$index] = $row[1];
                $index++;
                                                                }
                }
                $sql = "SELECT users.Username FROM users";
                $result = mysql_query($sql,$conn);
                $arrayUsername = array();
                $index = 0;
                $arrData = array();
                if($result){
                while($row =  mysql_fetch_array($result)){
                $arrayUsernames[$index] = $row[0];
                $index++;
                                        }
                $times = 0;
                $index = 0;
                for($i=0; $i<count($arrayUsernames); $i++){
                for($j=0; $j<count($arrayConnections); $j++){
                if(strcmp($arrayConnections[$j],$arrayUsernames[$i])==0){
                $times++;
                                        }
                                        }
                $arrData[$index][0] = $arrayUsernames[$i];
                $arrData[$index][1] = $times;
                $times = 0;
```

```
                $index++;
                                        }
                                        }
                mysql_close($conn);
                $FC->addChartDataFromArray($arrData);
                # Render the chart
                $FC->renderChart();
            ?>
            </CENTER>
        </BODY>
</HTML>
```

Το διάγραμμα ροής της σελίδας:


start
$dbhost = 'localhost'
$dbuser = 'root'
$dbpass = ''
$conn = mysql_connect($dbhost, $dbuser, $dbpass)
! $conn
true
die('Could not connect: ' . mysql_error())
false
mysql_select_db('ptixiaki')
$FC = new FusionCharts("Pie3D","500","500")
$FC->setSwfPath("FusionCharts/")
$strParam="Caption=Incidence of users connections; subCaption=; showBorder=1"
$FC->setChartParams($strParam)
$sql = "SELECT users.id,users.Username FROM users INNER JOIN connections ON users.id = connections.id ORDER BY users.Username"
$result = mysql_query($sql, $conn)


! $result
true
die('Could not get data: ' . mysql_error())
false
$result
true
$arrayConnections = array()
$index = 0
$row = mysql_fetch_array($result)
false
$sql = "SELECT users.Username FROM users"
true
$arrayConnections[$index] = $row[1]
$index++
$result = mysql_query($sql, $conn)
$arrayUsername = array()
$index = 0
$arrData = array()
$result
true
$row = mysql_fetch_array($result)
$times = 0
true
$arrayUsernames[$index] = $row[0]
$index++
$index = 0



4.5 Η ιστοσελίδα Video Statistics

Μέσω της ιστοσελίδας Video Statistics μπορούμε να δούμε σε διάγραμμα τον αριθμό των βίντεο που έχει καταγράψει ο κάθε χρήστης. Αρχικά συνδεόμαστε στην βάση δεδομένων με όνομα ptixiaki με τον τρόπο που είδαμε στην προηγούμενη ιστοσελίδα. Έπειτα με την συνάρτηση $FC = new FusionCharts("Pie3D","500","500") ορίζουμε ένα νέο διάγραμμα τύπου πίτας με μήκος και ύψος 500 pixel. Ορίζουμε τον τίτλο του διαγράμματος στην μεταβλητή $strParam και με την συνάρτηση $FC->setChartParams($strParam) θέτουμε τον τίτλο στο διάγραμμα. Στην μεταβλητή $sql ορίζουμε το query που θα κάνουμε στην βάση με το οποίο διαλέγουμε τα πεδία id και Username απο τον πίνακα users, ενώνουμε τα πεδίο id του πίνακα users με το πεδίο userId του πίνακα videos και τα ταξινομούμε βάσει του πεδίου Username που βρίσκεται στον πίνακα users. Με την συνάρτηση $result = mysql_query($sql, $conn) κάνουμε το query στην βάση δεδομένων και αποθηκεύουμε τα αποτελέσματα στη μεταβλητή $result. Εάν δεν για κάποιο λόγο δεν γίνει το query εμφανίζεται το μήνυμα Could not get data. Στην συνέχεια ορίζουμε ένα πίνακα με όνομα $ arrayVideos και ένα δείκτη $index ο οποίος ισούται με το μηδέν. Χρησιμοποιούμε ένα βρόχο while για να αποθηκεύσουμε τα ονόματα των χρηστών που έχουν συνδεθεί στο πίνακα $ arrayVideos []. Έπειτα στην μεταβλητή $sql ορίζουμε ένα νέο query με το οποίο διαλέγουμε το πεδίο Username από τον πίνακα users. Με την συνάρτηση mysql_query($sql,$conn) αποθηκεύουμε τα αποτελέσματα του query στη μεταβλητή $result. Ορίζουμε δύο νέους πίνακες $arrayUsername, $arrData και ένα δείκτη $index ο οποίος ισούται με το μηδέν. Χρησιμοποιούμε τον βρόχο while για να αποθηκεύσουμε τα αποτελέσματα του query που κάναμε προηγουμένως στον πίνακα $arrayUsernames [] και περιέχουν τα ονόματα όλων των χρηστών που υπάρχου στον πίνακα users στη βάση δεδομένων. Ορίζουμε ένα μετρητή με

όνομα $times ο οποίος ισούται με μηδέν και μηδενίζουμε τον δείκτη $index. Έπειτα μπαίνουμε σε ένα διπλό βρόχο for ο οποίος εκτελεί την ίδια λειτουργία με αυτόν στην ιστοσελίδα Users Statistics, δηλαδή όσες φορές ένας χρήστης έχει καταγράψει ένα βίντεο τόσες φορές θα υπάρχει το όνομα του στον πίνακα $arrayVideos. Αποθηκεύουμε κάθε φορά το όνομα του χρήστη στο πίνακα $arrData[$index][0] και τον αριθμό των βίντεο που έχει καταγράψει στον πίνακα $arrData[$index][1]. Μηδενίζουμε το μετρητή, αυξάνουμε τον δείκτη κατά ένα και επαναλαμβάνουμε την διαδικασία για όλους τους χρήστες. Τέλος αποσυνδεόμαστε από την βάση δεδομένων με την συνάρτηση mysql_close($conn), προσθέτουμε τον πίνακα $arrData στο διάγραμμα με την συνάρτηση $FC->addChartDataFromArray($arrData) και το εμφανίζουμε με την $FC->renderChart().

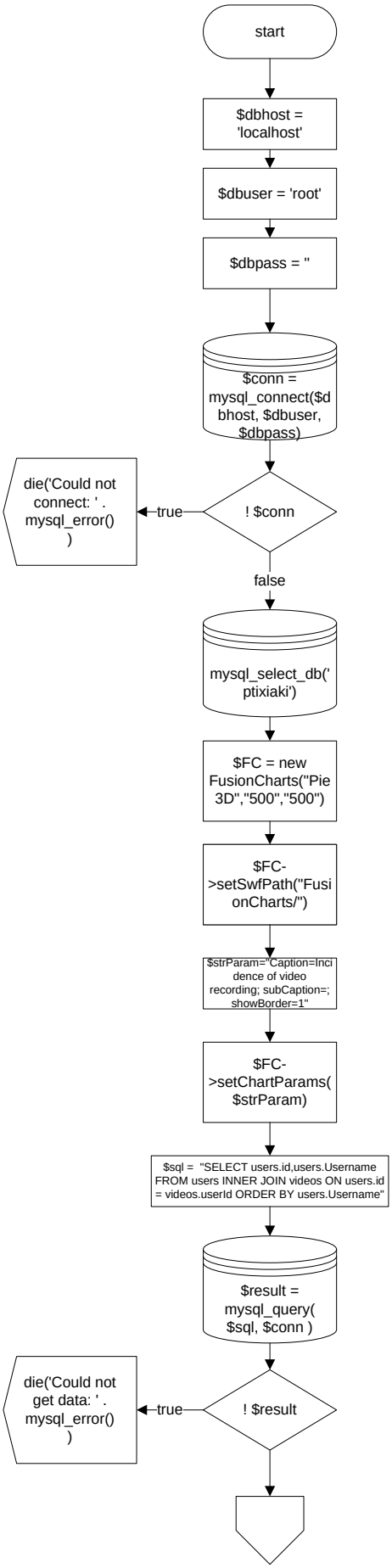
Ο κώδικας της σελίδας:

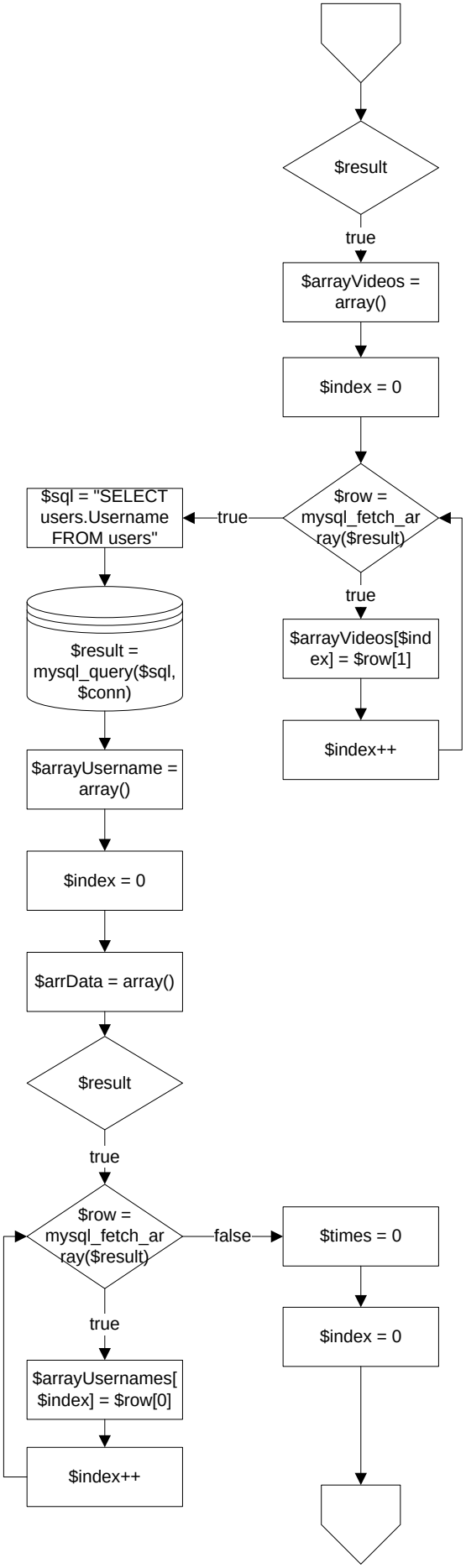
```
<?php
	//We have included ../Includes/FusionCharts_Gen.php, which contains
	//FusionCharts PHP Class to help us easily embed charts
	//We have also used ../Includes/DBConn.php to easily connect to a database.
	include("Includes/FusionCharts_Gen.php");
	include("Includes/DBConn.php");
?>
<HTML>
	<HEAD>
		<TITLE>
			Video Statistics
		</TITLE>
		<?php
 //You need to include the following JS file, if you intend to embed the chart using JavaScript.
//Embedding using JavaScript avoids the "Click to Activate..." issue in Internet Explorer
//When you make your own charts, make sure that the path to this JS file is correct.
 //Else, you will JavaScript errors.
		?>
		<SCRIPT LANGUAGE="Javascript" SRC="FusionCharts/FusionCharts.js">
		</SCRIPT>
	</HEAD>
	<BODY>
		<CENTER>
		<?php
//In this example, we show how to connect FusionCharts to a database.
//For the sake of ease, we have used an MySQL databases containing two tables.
// Connect to the Database
$dbhost = 'localhost';
$dbuser = 'root';
$dbpass = '';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
die('Could not connect: ' . mysql_error());
		}
mysql_select_db('ptixiaki');
```

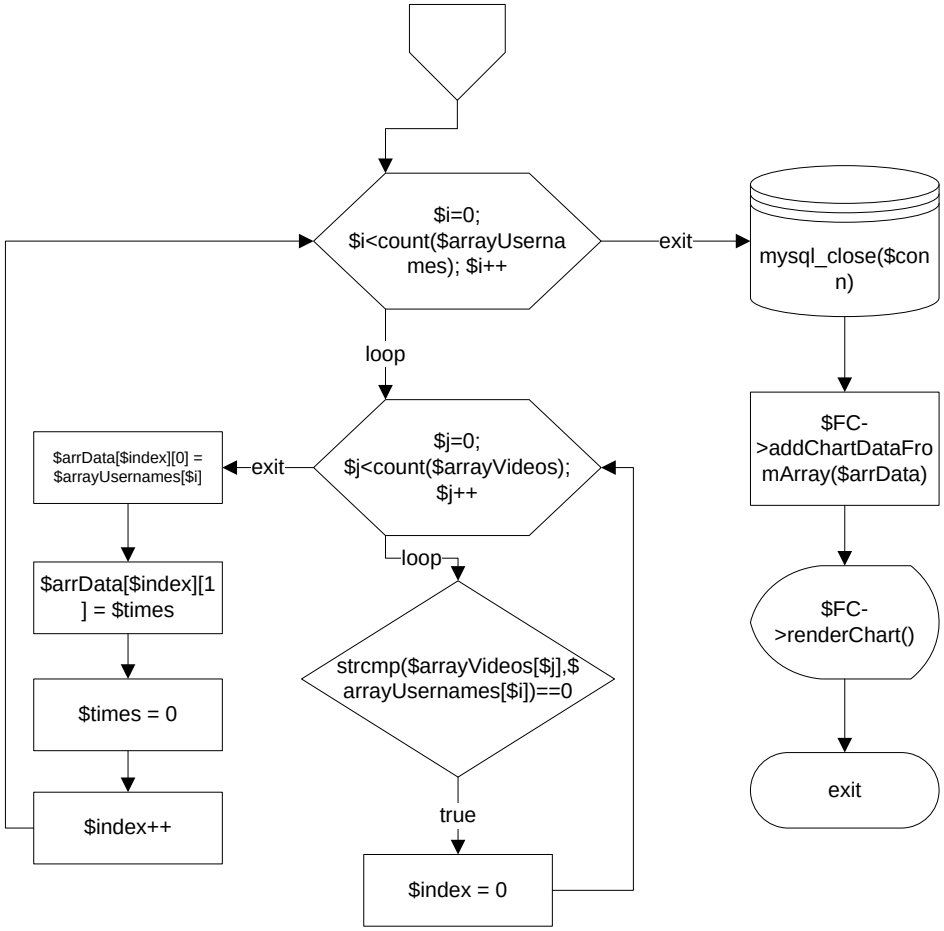
```
// Create pie 3d chart object using FusionCharts PHP Class
$FC = new FusionCharts("Pie3D","500","500");
# Set Relative Path of chart SWF file.
$FC->setSwfPath("FusionCharts/");
# Define chart attributes
$strParam="Caption=Incidence of video recording; subCaption=; showBorder=1";
// Set chart attributes
$FC->setChartParams($strParam);
$sql =  "SELECT users.id,users.Username FROM users INNER JOIN videos ON users.id =
videos.userId ORDER BY users.Username";
$result = mysql_query( $sql, $conn );
if(! $result )
{
die('Could not get data: ' . mysql_error());
                    }
 //Pass the SQL Query result to the FusionCharts PHP Class function
 //along with field/column names that are storing chart values and corresponding category
names
//to set chart data from database
  if ($result)
 {       $arrayVideos = array();
$index = 0;
while($row =  mysql_fetch_array($result)){
$arrayVideos[$index] = $row[1];
$index++;
                                                                        }
                    }
$sql = "SELECT users.Username FROM users";
$result = mysql_query($sql,$conn);
$arrayUsername = array();
$index = 0;
$arrData = array();
if($result){
while($row =  mysql_fetch_array($result)){
$arrayUsernames[$index] = $row[0];
$index++;
                                                    }
$times = 0;
$index = 0;
for($i=0; $i<count($arrayUsernames); $i++){
for($j=0; $j<count($arrayVideos); $j++){
if(strcmp($arrayVideos[$j],$arrayUsernames[$i])==0){
$times++;
                                                    }
                                                    }
$arrData[$index][0] = $arrayUsernames[$i];
$arrData[$index][1] = $times;
$times = 0;
$index++; }
                                                    }
```

```
mysql_close($conn);
$FC->addChartDataFromArray($arrData);
//Render the chart
$FC->renderChart();
            ?>
            </CENTER>
      </BODY>
</HTML>
```

Το διάγραμμα ροής της σελίδας:


start
$dbhost = 'localhost'
$dbuser = 'root'
$dbpass = ''
$conn = mysql_connect($dbhost, $dbuser, $dbpass)
! $conn
true
die('Could not connect: ' . mysql_error())
false
mysql_select_db('ptixiaki')
$FC = new FusionCharts("Pie 3D","500","500")
$FC->setSwfPath("FusionCharts/")
$strParam="Caption=Incidence of video recording; subCaption=; showBorder=1"
$FC->setChartParams($strParam)
$sql = "SELECT users.id,users.Username FROM users INNER JOIN videos ON users.id = videos.userId ORDER BY users.Username"
$result = mysql_query($sql, $conn)
! $result
true
die('Could not get data: ' . mysql_error())


$result
true
$arrayVideos = array()
$index = 0
$row = mysql_fetch_array($result)
true
$sql = "SELECT users.Username FROM users"
true
$arrayVideos[$index] = $row[1]
$index++
$result = mysql_query($sql, $conn)
$arrayUsername = array()
$index = 0
$arrData = array()
$result
true
$row = mysql_fetch_array($result)
false
$times = 0
$index = 0
true
$arrayUsernames[$index] = $row[0]
$index++



4.6 Η ιστοσελίδα LAST HOUR LUX

Η ιστοσελίδα LAST HOUR LUX περιέχει ένα php script με το οποίο μπορούμε να δούμε σε γράφημα τις τιμές της φωτεινότητας την τελευταία ώρα. Αρχικά συνδεόμαστε πάλι στη βάση δεδομένων με την ίδια διαδικασία που είδαμε στις προηγούμενες σελίδες. Στην συνέχεια ορίζουμε με την συνάρτηση $FC = new FusionCharts("Line2D","1100","500") ένα διάγραμμα τύπου Line2D με μήκος 1100 και ύψος 500. Ορίζουμε τον τίτλο και τις ονομασίες των αξόνων X και Y στην μεταβλητή $strParam και θέτουμε στο διάγραμμα τις ιδιότητες αυτές με την συνάρτηση $FC->setChartParams($strParam). Έπειτα ορίζουμε στην μεταβλητή $sql το query που θα κάνουμε στη βάση δεδομένων με το οποίο διαλέγουμε το πεδίο lux από τον πίνακα measurements. Με την συνάρτηση $result = mysql_query($sql, $conn) κάνουμε το query στη βάση δεδομένων και αποθηκεύουμε τα αποτελέσματα στη μεταβλητή $result. Έπειτα με την συνάρτηση $up_limit = mysql_num_rows($result) βρίσκουμε πόσες σειρές έχει ο πίνακας measurements και αφαιρούμε τον αριθμό 60 από τον αριθμό των γραμμών. Με αυτό τον τρόπο βρίσκουμε τις 60 τελευταίες μετρήσεις αφού καταγράφουμε τις τιμές των lux κάθε ένα λεπτό. Έπειτα ορίζουμε ένα νέο query στη μεταβλητή $sql με το οποίο διαλέγουμε το πεδίο lux από τον πίνακα measurements αλλά αυτή την φορά ανάμεσα στα δύο όρια που βρήκαμε παραπάνω τα οποία είναι $up_limit και $down_limit. Με την συνάρτηση $result = mysql_query($sql, $conn) κάνουμε το query στη βάση δεδομένων και αποθηκευουμε τα αποτελέσματα στη μεταβλητή $result. Ορίζουμε ένα πίνακα με όνομα $arrData και ένα δείκτη $i που ισούται με το μηδέν. Στον βρόχο while αποθηκεύουμε τα λεπτά στο πίνακα $arrData στη θέση $arrData[$i][0] = $i και τα αποτελέσματα από το query στη θέση $arrData[$i][1] = $lux. Τέλος θέτουμε στο διάγραμμα τα δεδομένα από το πίνακα, κλείνουμε την σύνδεση μεταξύ της σελίδας και της βάσης δεδομένων με την συνάρτηση

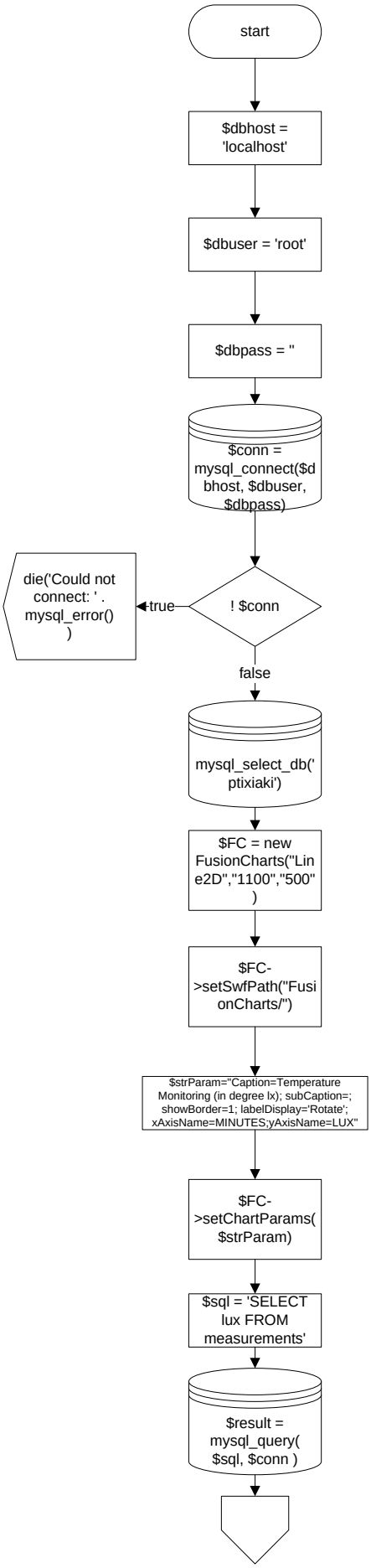
mysql_close($conn) και εμφανίζουμε τι=ο διάγραμμα στην σελίδα με τη συνάρτηση $FC->renderChart().

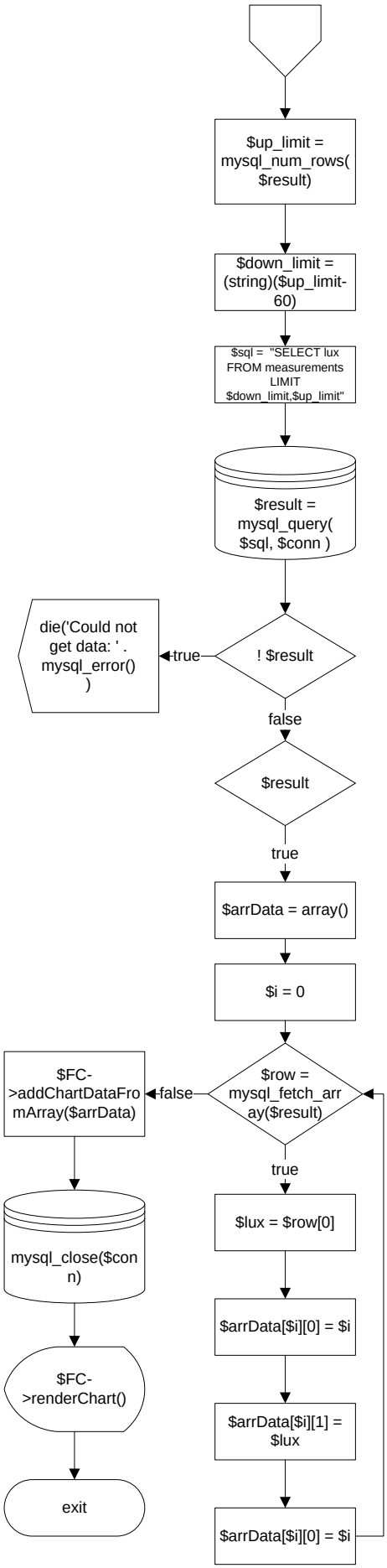
Ο κώδικας της σελίδας:

```
<?php
	//We have included ../Includes/FusionCharts_Gen.php, which contains
	//FusionCharts PHP Class to help us easily embed charts
	//We have also used ../Includes/DBConn.php to easily connect to a database.
	include("Includes/FusionCharts_Gen.php");
	include("Includes/DBConn.php");
?>
<HTML>
	<HEAD>
		<TITLE>
			LAST HOUR LUX
		</TITLE>
		<?php   //You need to include the following JS file, if you intend to embed the chart using JavaScript.
	//Embedding using JavaScript avoids the "Click to Activate..." issue in Internet Explorer
	 //When you make your own charts, make sure that the path to this JS file is correct.
	  //Else, you will JavaScript errors.
		?>
		<SCRIPT LANGUAGE="Javascript" SRC="FusionCharts/FusionCharts.js">
		</SCRIPT>
	</HEAD>
	<BODY>
		<CENTER>
		<?php
			// Connect to the Database
			$dbhost = 'localhost';
			$dbuser = 'root';
			$dbpass = '';
			$conn = mysql_connect($dbhost, $dbuser, $dbpass);
			if(! $conn )
			{
			die('Could not connect: ' . mysql_error());
			}
			mysql_select_db('ptixiaki');
			$FC = new FusionCharts("Line2D","1100","500");
			//Set Relative Path of chart SWF file.
			$FC->setSwfPath("FusionCharts/");
			// Define chart attributes
  $strParam="Caption=Temperature Monitoring (in degree lx); subCaption=; showBorder=1; labelDisplay='Rotate'; xAxisName=MINUTES;yAxisName=LUX";
			#  Set chart attributes
			$FC->setChartParams($strParam);
			$sql = 'SELECT lux FROM measurements';
			$result = mysql_query( $sql, $conn );
```

```
            $up_limit = mysql_num_rows($result);
            $down_limit = (string)($up_limit-60);
            $sql =  "SELECT lux FROM measurements LIMIT $down_limit,$up_limit";
            $result = mysql_query( $sql, $conn );
            if(! $result )
            {
            die('Could not get data: ' . mysql_error());
            }
            //Pass the SQL Query result to the FusionCharts PHP Class function
            //to set chart data from database
            if ($result)
            {
            $arrData = array();
            $i = 0;
                  while($row = mysql_fetch_array($result)){
                  $lux = $row[0];
                 $arrData[$i][0] = $i;
                  $arrData[$i][1] = $lux;
                  $i++;
                                                              }
                  $FC->addChartDataFromArray($arrData);
            }
            mysql_close($conn);
            # Render the chart
            $FC->renderChart();
      ?>
      </CENTER>
   </BODY>
</HTML>
```

Το διάγραμμα ροής της σελίδας:


start
$dbhost = 'localhost'
$dbuser = 'root'
$dbpass = ''
$conn = mysql_connect($dbhost, $dbuser, $dbpass)
! $conn
true
die('Could not connect: ' . mysql_error())
false
mysql_select_db('ptixiaki')
$FC = new FusionCharts("Line2D","1100","500")
$FC->setSwfPath("FusionCharts/")
$strParam="Caption=Temperature Monitoring (in degree lx); subCaption=; showBorder=1; labelDisplay='Rotate'; xAxisName=MINUTES;yAxisName=LUX"
$FC->setChartParams($strParam)
$sql = 'SELECT lux FROM measurements'
$result = mysql_query($sql, $conn)


$up_limit = mysql_num_rows($result)
$down_limit = (string)($up_limit-60)
$sql = "SELECT lux FROM measurements LIMIT $down_limit,$up_limit"
$result = mysql_query($sql, $conn)
! $result
true
die('Could not get data: ' . mysql_error())
false
$result
true
$arrData = array()
$i = 0
$row = mysql_fetch_array($result)
false
$FC->addChartDataFromArray($arrData)
true
$lux = $row[0]
$arrData[$i][0] = $i
$arrData[$i][1] = $lux
$arrData[$i][0] = $i
mysql_close($conn)
$FC->renderChart()
exit

4.7 Η ιστοσελίδα LAST HOUR TEMPERATURE

Με την ιστοσελίδα LAST HOUR TEMPERATURE μπορούμε να δούμε τις τιμές της θερμοκρασίας την τελευταία ώρα. Αρχικά συνδεόμαστε στη βάση δεδομένων με τη γνωστή διαδικασία. Έπειτα ορίζουμε ένα διάγραμμα το οποίο είναι Line2D με μήκος 1100 και ύψος 500. Στη μεταβλητή $strParam ορίζουμε το τίτλο του διαγράμματος καθώς και τα ονόματα των αξόνων και με τη συνάρτηση $FC->setChartParams($strParam) θέτουμε αυτές τις ιδιότητες στο διάγραμμα. Στη μεταβλητή $sql ορίζουμε το query που θα κάνουμε στη βάση δεδομένων με το οποίο διαλέγουμε το πεδίο temperature από τον πίνακα measurements. Με τη συνάρτηση $result = mysql_query($sql, $conn) κάνουμε το query στη βάση δεδομένων και αποθηκεύουμε τα αποτελέσματα στη μεταβλητή $result. Με τη συνάρτηση $up_limit = mysql_num_rows($result) βρίσκουμε τον αριθμό των σειρών στον πίνακα και τον αποθηκεύουμε στη μεταβλητή $up_limit. Αφαιρούμε τον αριθμό 60 από την μεταβλητή $up_limit για να βρούμε το κάτω όριο. Ορίζουμε ένα νέο query στη μεταβλητή $sql με το οποίο επιλέγουμε τις τιμές του πεδίου temperature που βρίσκονται ανάμεσα στα όρια $down_limit και $up_limit από τον πίνακα measurements. Με τη συνάρτηση $result = mysql_query($sql, $conn) κάνουμε το query στη βάση δεδομένων και αποθηκεύουμε τα αποτελέσματα στη μεταβλητή $result. Στη συνέχεια ορίζουμε ένα πίνακα με όνομα $arrData και ένα δείκτη $i ο οποίος ισούται με μηδέν. Με το βρόχο while αποθηκεύουμε τα λεπτά στη θέση $arrData[$i][0] = $i και τη θερμοκρασία στη θέση $arrData[$i][1] = $temp. Τέλος θέτουμε τα δεδομένα του πίνακα $arrData στο διάγραμμα με τη συνάρτηση $FC->addChartDataFromArray($arrData), κλείνουμε τη σύνδεση μεταξύ της σελίδας και της βάσης δεδομένων με τη συνάρτηση mysql_close($conn) και εμφανίζουμε το διάγραμμα στη σελίδα.

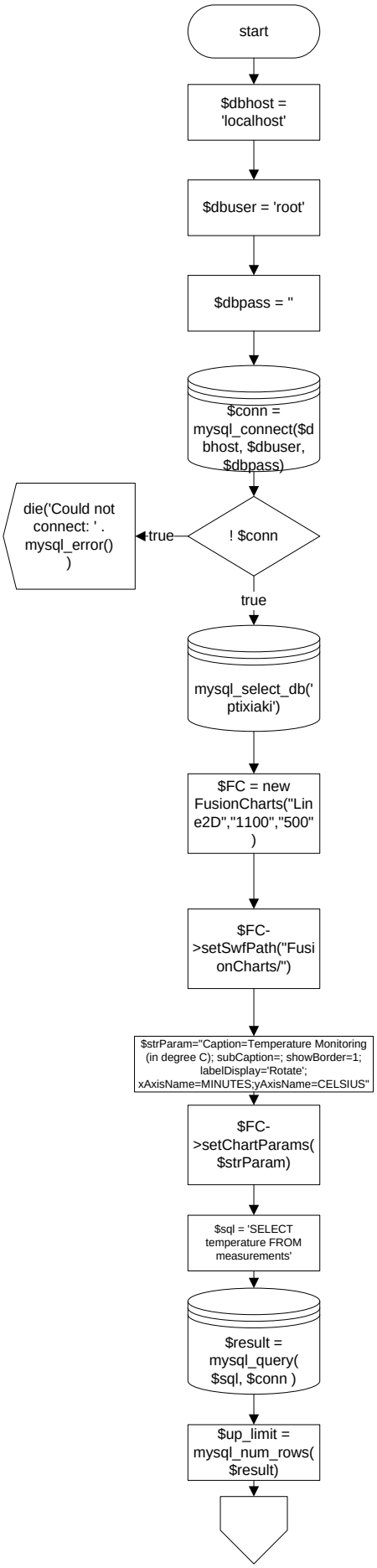
Ο κώδικας της σελίδας:

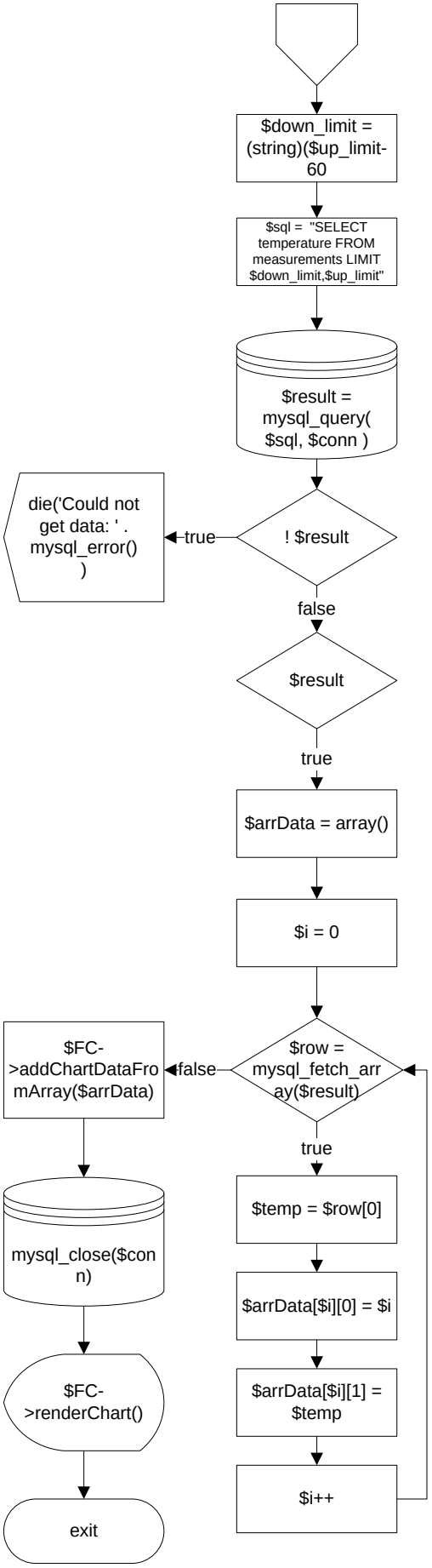
```
<?php
	//We have included ../Includes/FusionCharts_Gen.php, which contains
	//FusionCharts PHP Class to help us easily embed charts
	//We have also used ../Includes/DBConn.php to easily connect to a database.
	include("Includes/FusionCharts_Gen.php");
	include("Includes/DBConn.php");
?>
<HTML>
	<HEAD>
		<TITLE>
			LAST HOUR TEMPERATURE
		</TITLE>
		<?php
 //You need to include the following JS file, if you intend to embed the chart using JavaScript.
//Embedding using JavaScript avoids the "Click to Activate..." issue in Internet Explorer
//When you make your own charts, make sure that the path to this JS file is correct.
//Else, you will JavaScript errors.
		?>
		<SCRIPT LANGUAGE="Javascript" SRC="FusionCharts/FusionCharts.js">
		</SCRIPT>
	</HEAD>
	<BODY>
```

```
<CENTER>
<?php
    // Connect to the Database
    $dbhost = 'localhost';
    $dbuser = 'root';
    $dbpass = '';
    $conn = mysql_connect($dbhost, $dbuser, $dbpass);
    if(! $conn )
    {
    die('Could not connect: ' . mysql_error());
    }
    mysql_select_db('ptixiaki');
   $FC = new FusionCharts("Line2D","1100","500");
//Set Relative Path of chart SWF file.
$FC->setSwfPath("FusionCharts/");
//Define chart attributes
$strParam="Caption=Temperature Monitoring (in degree C); subCaption=; showBorder=1;
labelDisplay='Rotate'; xAxisName=MINUTES;yAxisName=CELSIUS";
     // Set chart attributes
     $FC->setChartParams($strParam);
    $sql = 'SELECT temperature FROM measurements';
    $result = mysql_query( $sql, $conn );
   $up_limit = mysql_num_rows($result);
   $down_limit = (string)($up_limit-60);
   $sql =  "SELECT temperature FROM measurements LIMIT $down_limit,$up_limit";
        $result = mysql_query( $sql, $conn );
        if(! $result )
        {
        die('Could not get data: ' . mysql_error());
        }
 //Pass the SQL Query result to the FusionCharts PHP Class function
        //to set chart data from database
        if ($result)
        {
            $arrData = array();
             $i = 0;
             while($row = mysql_fetch_array($result)){
            $temp = $row[0];
             $arrData[$i][0] = $i;
             $arrData[$i][1] = $temp;
             $i++;
                                                    }
        $FC->addChartDataFromArray($arrData);
        }
        mysql_close($conn);
        // Render the chart
        $FC->renderChart();
?>
</CENTER>
</BODY>
```

</HTML>

Το διάγραμμα ροής της σελίδας:


start
$dbhost = 'localhost'
$dbuser = 'root'
$dbpass = ''
$conn = mysql_connect($dbhost, $dbuser, $dbpass)
! $conn
true
die('Could not connect: ' . mysql_error())
true
mysql_select_db('ptixiaki')
$FC = new FusionCharts("Line2D","1100","500")
$FC->setSwfPath("FusionCharts/")
$strParam="Caption=Temperature Monitoring (in degree C); subCaption=; showBorder=1; labelDisplay='Rotate'; xAxisName=MINUTES;yAxisName=CELSIUS"
$FC->setChartParams($strParam)
$sql = 'SELECT temperature FROM measurements'
$result = mysql_query($sql, $conn)
$up_limit = mysql_num_rows($result)


$down_limit = (string)($up_limit-60
$sql = "SELECT temperature FROM measurements LIMIT $down_limit,$up_limit"
$result = mysql_query($sql, $conn)
! $result
true
die('Could not get data: ' . mysql_error())
false
$result
true
$arrData = array()
$i = 0
$row = mysql_fetch_array($result)
false
$FC->addChartDataFromArray($arrData)
true
$temp = $row[0]
$arrData[$i][0] = $i
$arrData[$i][1] = $temp
$i++
mysql_close($conn)
$FC->renderChart()
exit

4.8 Η ιστοσελίδα LAST 24 HOURS AVERAGE LUX

Με την σελίδα LAST 24 HOURS AVERAGE LUX μπορούμε να δούμε τις μέσες τιμές της φωτεινότητας το τελευταίο εικοσιτετράωρο. Αρχικά συνδεόμαστε με τη βάση δεδομένων με τον τρόπο που έχουμε δει στις προηγούμενες σελίδες. Έπειτα ορίζουμε το διάγραμμα με το τρόπο που είδαμε στη προηγούμενη ιστοσελίδα αφού και αυτό είναι Line2D. Ορίζουμε στη μεταβλητή $strParam το τίτλο του διαγράμματος καθώς και τα ονόματα των αξόνων και θέτουμε αυτές τις ιδιότητες στο διάγραμμα. Στη μεταβλητή $sql ορίζουμε το query που θα κάνουμε στη βάση δεδομένων με το οποίο επιλέγουμε το πεδίο lux από το πίνακα measurements. Με τη συνάρτηση $result = mysql_query($sql, $conn) κάνουμε το query στη βάση δεδομένων και αποθηκεύουμε τα αποτελέσματα στη μεταβλητή $result. Με τη συνάρτηση $up_limit = mysql_num_rows($result) βρίσκουμε τον αριθμό των σειρών στο πίνακα measurements. Αφαιρούμε τον αριθμό 1440 από τον αριθμό των σειρών για να βρούμε όλες της τιμές των lux το τελευταίο εικοσιτετράωρο και την αποθηκεύουμε στη μεταβλητή $down_limit. Αυτό το κάνουμε επειδή το εικοσιτετράωρο έχει 1440 λεπτά και εμείς παίρνουμε τις μετρήσεις κάθε λεπτό. Στη συνέχεια ορίζουμε ένα νέο query στη μεταβλητή $sql με το οποίο επιλέγουμε το πεδίο lux από τη σειρά $down_limit έως τη $up_limit. Με τη συνάρτηση $result = mysql_query($sql, $conn) κάνουμε το query στη βάση δεδομένων και αποθηκεύουμε τα αποτελέσματα στη μεταβλητή $result. Έπειτα ορίζουμε τις εξής μεταβλητές $lux=0, $hour=1, $index=0, και τους πίνακες $arrData και $tempArray. Με το βρόχο while αποθηκεύουμε στο πίνακα $tempArray[] τα αποτελέσματα από το query που κάναμε στη βάση δεδομένων. Θέτουμε πάλι τη μεταβλητή $index = 0 ίση με το μηδέν και μπαίνουμε σε ένα διπλό βρόχο for. Ο πρώτος βρόχος επαναλαμβάνεται 24 φορές δηλαδή όσες και οι ώρες ενώ ο δεύτερος επαναλαμβάνεται 60 όσα τα λεπτά. Μέσα στο δεύτερο βρόχο προσθέτουμε τις τιμές των lux για κάθε ώρα και έπειτα αποθηκεύουμε στο πίνακα $arrData[$i][0] την ώρα και στο πίνακα $arrData[$i][1] τη μέση τιμή των lux κάθε ώρας. Με τη συνάρτηση number_format((float)$lux/60, 2, '.', '') διαιρούμε το άθροισμα των lux διά δύο και το αποτέλεσμα είναι δεκαδικός με δύο δεκαδικά ψηφία. Την παραπάνω διαδικασία την κάνουμε 24 φορές δηλαδή για κάθε ώρα του εικοσιτετραώρου. Με τη συνάρτηση $FC->addChartDataFromArray($arrData) θέτουμε τα δεδομένα στο διάγραμμα. Τέλος κλείνουμε τη σύνδεση με τη συνάρτηση mysql_close($conn) και εμφανίζουμε το διάγραμμα στη σελίδα με τη συνάρτηση $FC->renderChart().

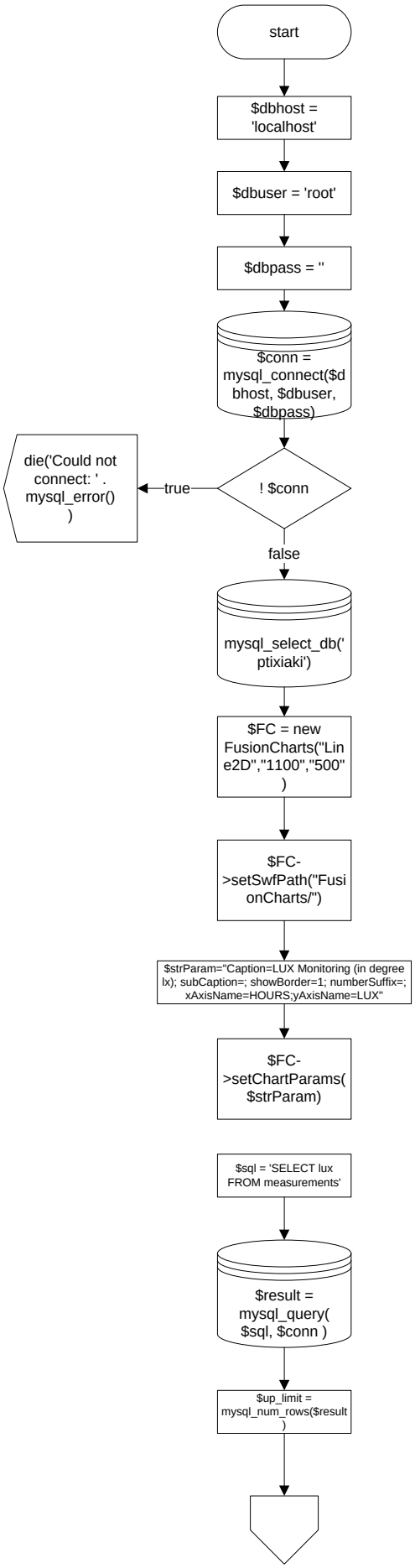
Ο κώδικας της σελίδας:

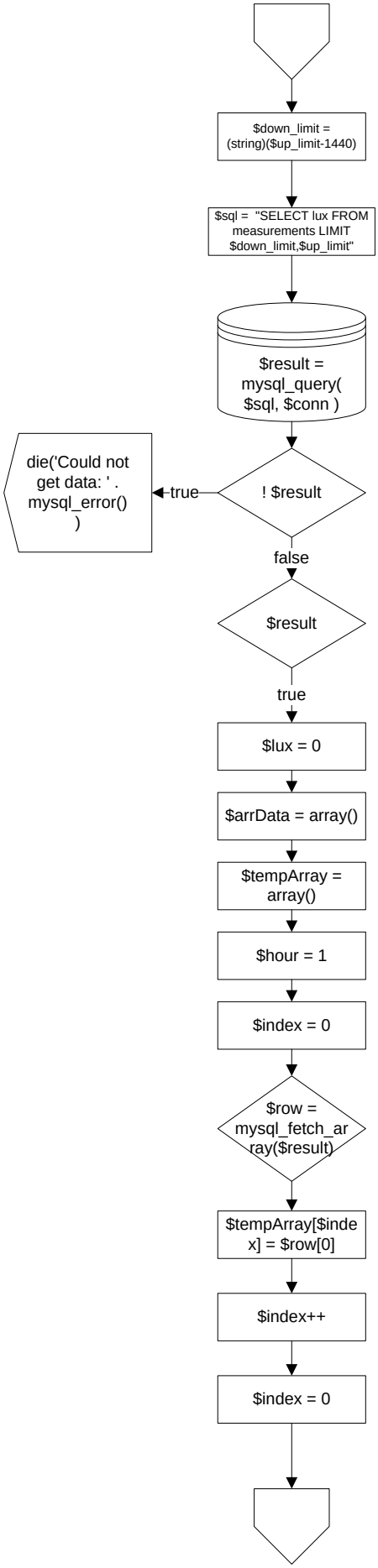
```
<?php
	//We have included ../Includes/FusionCharts_Gen.php, which contains
	//FusionCharts PHP Class to help us easily embed charts
	//We have also used ../Includes/DBConn.php to easily connect to a database.
	include("Includes/FusionCharts_Gen.php");
	include("Includes/DBConn.php");
?>
<HTML>
	<HEAD>
		<TITLE>
			LAST 24 HOURS AVERAGE LUX
		</TITLE>
		<?php
 //You need to include the following JS file, if you intend to embed the chart using JavaScript.
```

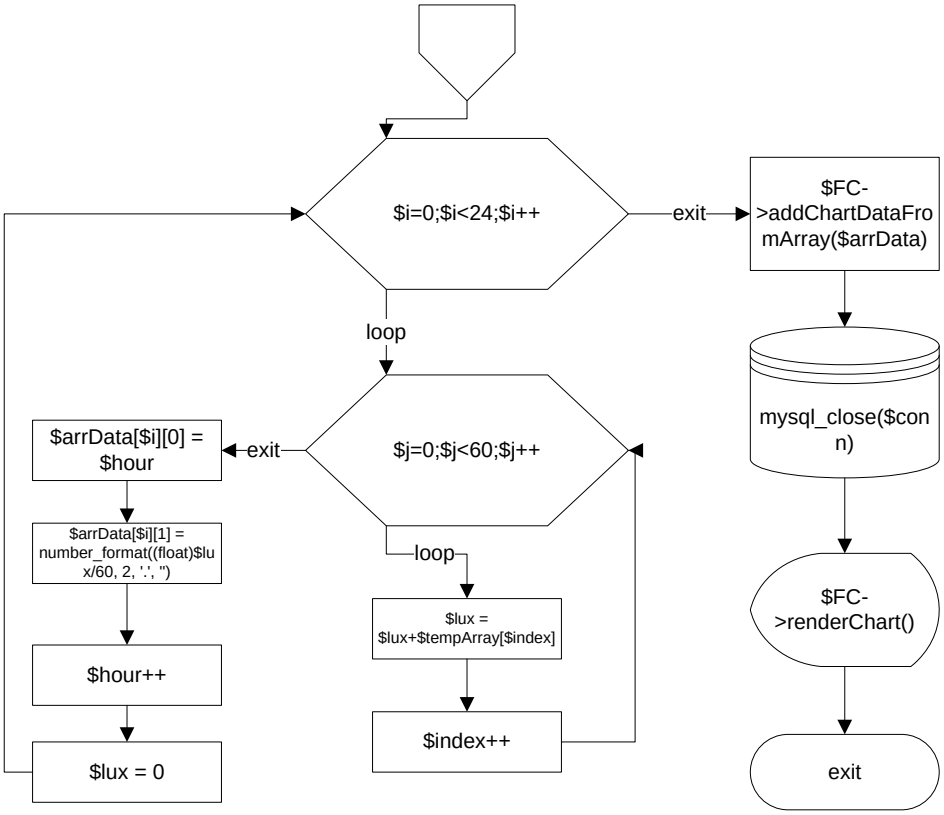
```
//Embedding using JavaScript avoids the "Click to Activate..." issue in Internet Explorer
//When you make your own charts, make sure that the path to this JS file is correct.
//Else, you will JavaScript errors.
            ?>
            <SCRIPT LANGUAGE="Javascript" SRC="FusionCharts/FusionCharts.js">
            </SCRIPT>
      </HEAD>
      <BODY>
            <CENTER>
            <?php
                 // Connect to the Database
                  $dbhost = 'localhost';
                  $dbuser = 'root';
                  $dbpass = '';
                  $conn = mysql_connect($dbhost, $dbuser, $dbpass);
                  if(! $conn )
                  {
                  die('Could not connect: ' . mysql_error());
                  }
                  mysql_select_db('ptixiaki');
                  $FC = new FusionCharts("Line2D","1100","500");
                  //Set Relative Path of chart SWF file.
                  $FC->setSwfPath("FusionCharts/");
                  //Define chart attributes
                  $strParam="Caption=LUX Monitoring (in degree lx); subCaption=;
showBorder=1; numberSuffix=; xAxisName=HOURS;yAxisName=LUX";
                  //Set chart attributes
                  $FC->setChartParams($strParam);
                  $sql = 'SELECT lux FROM measurements';
                  $result = mysql_query( $sql, $conn );
                  $up_limit = mysql_num_rows($result);
                  $down_limit = (string)($up_limit-1440);
                  $sql =  "SELECT lux FROM measurements LIMIT $down_limit,$up_limit";
                 $result = mysql_query( $sql, $conn );
                  if(! $result )
                  {
                  die('Could not get data: ' . mysql_error());
                  }
//Pass the SQL Query result to the FusionCharts PHP Class function
//to set chart data from database
                  if ($result)
                  {       $lux = 0;
                        $arrData = array();
                        $tempArray = array();
                         $hour = 1;
                         $index = 0;
                       while($row =  mysql_fetch_array($result)){
                       $tempArray[$index] = $row[0];
                       $index++;
                                                                  }
```

```
                $index = 0;
                 for($i=0;$i<24;$i++){
                 for($j=0;$j<60;$j++){
                $lux = $lux+$tempArray[$index];
                 $index++;
                                                        }
                 $arrData[$i][0] = $hour;
                 $arrData[$i][1] = number_format((float)$lux/60, 2, '.', '');
                 $hour++;
                 $lux = 0;
                                                        }
                 $FC->addChartDataFromArray($arrData);
            }
            mysql_close($conn);
            # Render the chart
            $FC->renderChart();
        ?>
        </CENTER>
    </BODY>
</HTML>
```

Το διάγραμμα ροής της σελίδας:


start
$dbhost = 'localhost'
$dbuser = 'root'
$dbpass = ''
$conn = mysql_connect($dbhost, $dbuser, $dbpass)
! $conn
true
die('Could not connect: ' . mysql_error())
false
mysql_select_db('ptixiaki')
$FC = new FusionCharts("Line2D","1100","500")
$FC->setSwfPath("FusionCharts/")
$strParam="Caption=LUX Monitoring (in degree lx); subCaption=; showBorder=1; numberSuffix=; xAxisName=HOURS;yAxisName=LUX"
$FC->setChartParams($strParam)
$sql = 'SELECT lux FROM measurements'
$result = mysql_query($sql, $conn)
$up_limit = mysql_num_rows($result)


$down_limit = (string)($up_limit-1440)
$sql = "SELECT lux FROM measurements LIMIT $down_limit,$up_limit"
$result = mysql_query($sql, $conn)
! $result
true
die('Could not get data: ' . mysql_error())
false
$result
true
$lux = 0
$arrData = array()
$tempArray = array()
$hour = 1
$index = 0
$row = mysql_fetch_array($result)
$tempArray[$index] = $row[0]
$index++
$index = 0



4.9 Η ιστοσελίδα LAST 24 HOURS AVERAGE TEMPERATURE

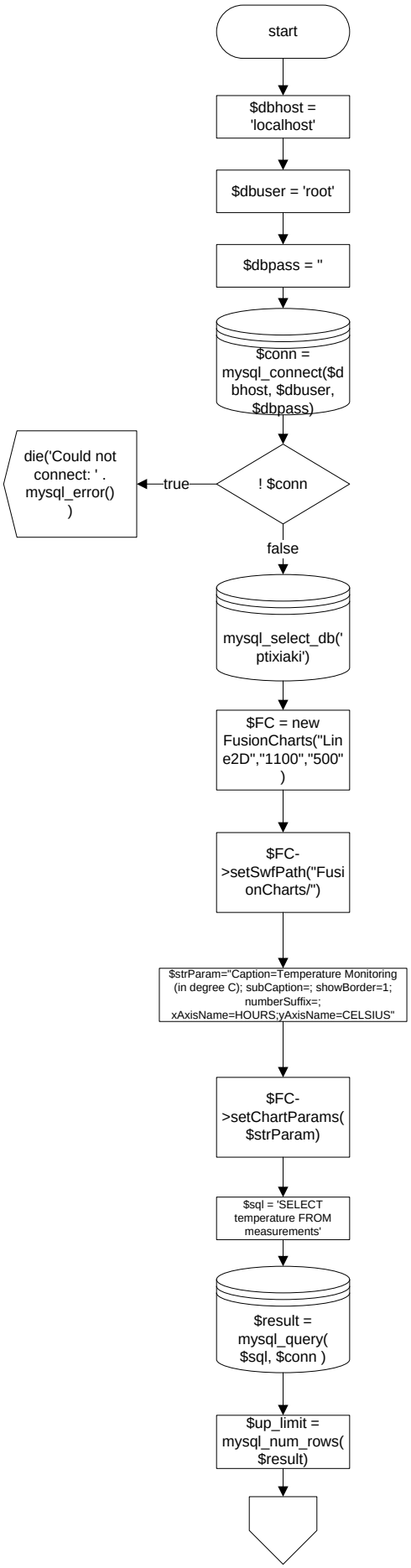
Η ιστοσελίδα last 24 hours average temperature περιέχει ένα php script με το οποίο μπορούμε να δούμε τις μέσες τιμές της θερμοκρασίας το τελευταίο εικοσιτετράωρο. Αρχικά συνδεόμαστε με τη βάση δεδομένων με τον τρόπο πού έχουμε δει ποιό πάνω. Έπειτα ορίζουμε ένα νέο διάγραμμα τύπου Line2D με μήκος 1100 και ύψος 500 pixel. Στη μεταβλητή $strParam ορίζουμε τον τίτλο του διαγράμματος και τα ονόματα των X και Y αξόνων. Με τη συνάρτηση $FC->setChartParams($strParam) θέτουμε τις ιδιότητες αυτές στο διάγραμμα. Στη μεταβλητή $sql ορίζουμε το query που θα κάνουμε στη βάση δεδομένων με το οποίο επιλέγουμε το πεδίο temperature από τον πίνακα measurements. Με τη συνάρτηση $result = mysql_query($sql, $conn) κάνουμε το query στη βάση δεδομένων και αποθηκεύουμε τα αποτελέσματα στη μεταβλητή $result. Με τη συνάρτηση $up_limit = mysql_num_rows($result) βρίσκουμε τον αριθμό των σειρών στον πίνακα measurements και τον αποθηκεύουμε στη $up_limit. Όπως και προηγουμένως αφαιρούμε από τον αριθμό των σειρών το 1440 για να βρούμε τις τιμές του τελευταίου εικοσιτετραώρου. Στην $sql ορίζουμε το νέο query που θα κάνουμε στη βάση δεδομένων με το οποίο επιλέγουμε από το πεδίο temperature μόνο τις σειρές που βρίσκονται ανάμεσα στα όρια $down_limit και $up_limit. Με τη συνάρτηση $result = mysql_query($sql, $conn) κάνουμε το query που ορίσαμε προηγουμένως και αποθηκεύουμε τα αποτελέσματα στη μεταβλητή $result. Έπειτα ορίζουμε τις μεταβλητές $temp, $hour, $index και τους πίνακες $arrData και $tempArray. Στο βρόχο while αποθηκεύουμε τα αποτελέσματα από τη μεταβλητή $result στο πίνακα $tempArray. Στη συνέχεια μηδενίζουμε το δείκτη $index και μπαίνουμε σε ένα διπλό βρόχο for. Ο βρόχος αυτός έχει την ίδια ακριβώς λειτουργία με τον αντίστοιχο που είδαμε στη σελίδα LAST 24 HOURS AVERAGE LUX. Η μόνη διαφορά είναι ότι τώρα προσθέτουμε τις τιμές της θερμοκρασίας κάθε ώρας. Τέλος θέτουμε τα στοιχεία του πίνακα $arrData στο διάγραμμα, κλείνουμε τη σύνδεση με τη βάση δεδομένων και εμφανίζουμε το διάγραμμα στη σελίδα.

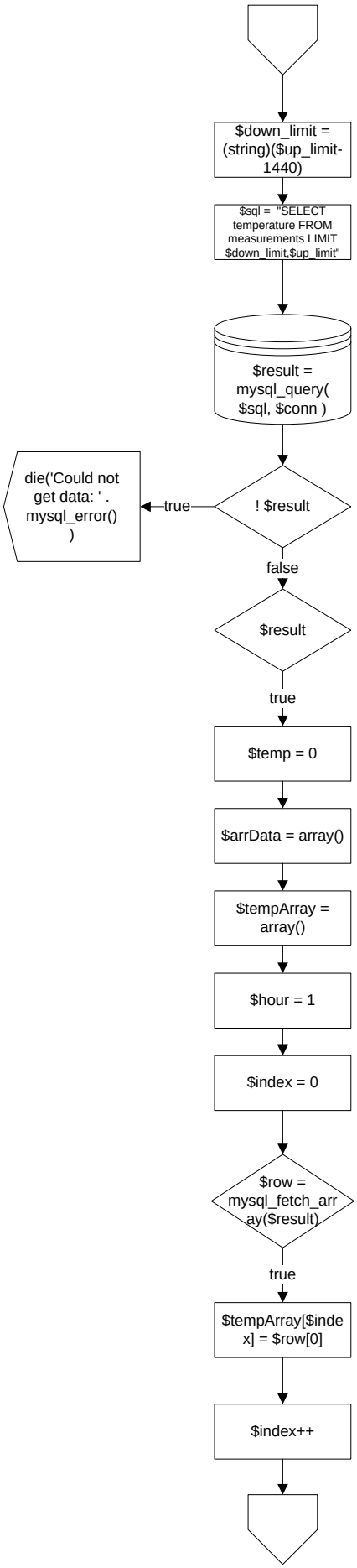
Ο κώδικας της σελίδας:

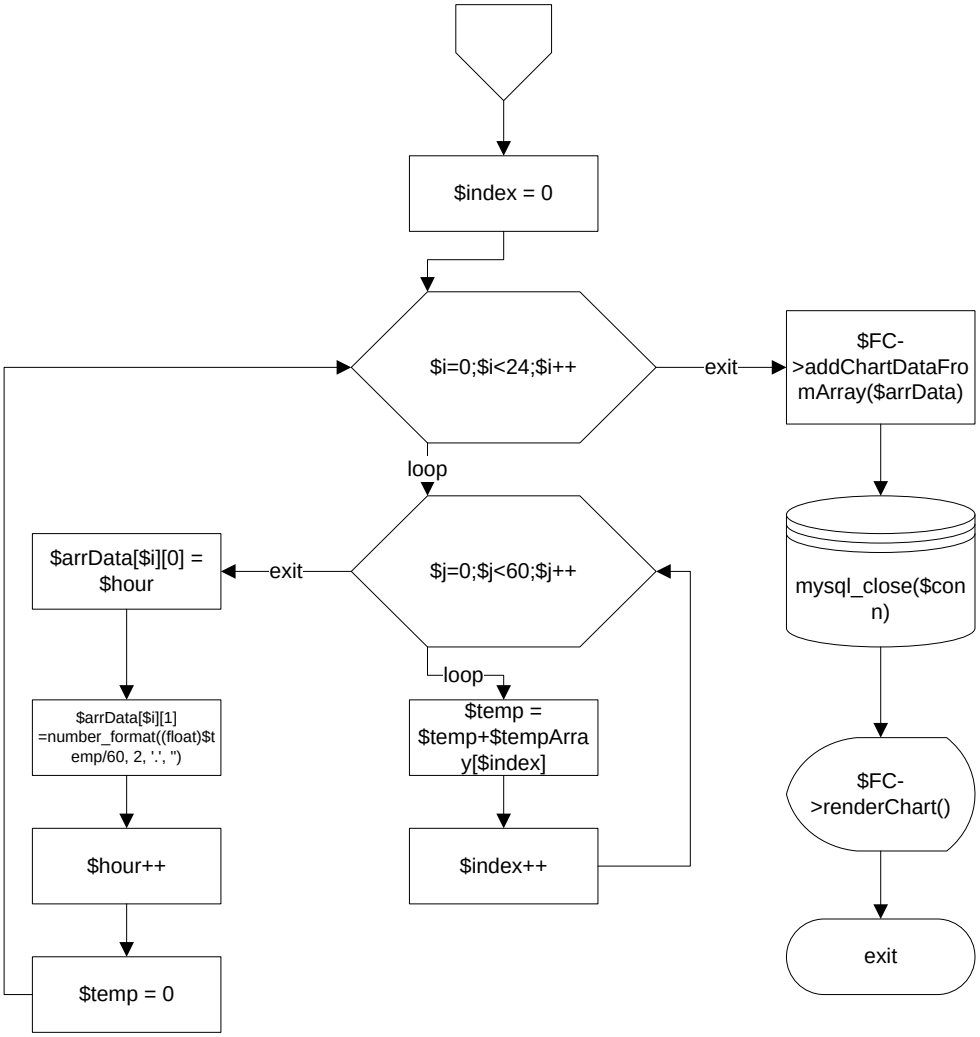
```
<?php
	//We have included ../Includes/FusionCharts_Gen.php, which contains
	//FusionCharts PHP Class to help us easily embed charts
	//We have also used ../Includes/DBConn.php to easily connect to a database.
	include("Includes/FusionCharts_Gen.php");
	include("Includes/DBConn.php");
?>
<HTML>
	<HEAD>
		<TITLE>
			LAST 24 HOURS AVERAGE TEMPERATURE
		</TITLE>
		<?php
//You need to include the following JS file, if you intend to embed the chart using JavaScript.
//Embedding using JavaScript avoids the "Click to Activate..." issue in Internet Explorer
//When you make your own charts, make sure that the path to this JS file is correct.
//Else, you will JavaScript errors.
		?>
		<SCRIPT LANGUAGE="Javascript" SRC="FusionCharts/FusionCharts.js">
		</SCRIPT>
	</HEAD>
	<BODY>
		<CENTER>
		<?php
		    // Connect to the Database
		    $dbhost = 'localhost';
		    $dbuser = 'root';
		    $dbpass = '';
		    $conn = mysql_connect($dbhost, $dbuser, $dbpass);
		    if(! $conn )
		    {
		    die('Could not connect: ' . mysql_error());
		    }
		    mysql_select_db('ptixiaki');
		    $FC = new FusionCharts("Line2D","1100","500");
		    //Set Relative Path of chart SWF file.
		    $FC->setSwfPath("FusionCharts/");
		   // Define chart attributes
$strParam="Caption=Temperature Monitoring (in degree C); subCaption=; showBorder=1;
numberSuffix=; xAxisName=HOURS;yAxisName=CELSIUS";
		    // Set chart attributes
		    $FC->setChartParams($strParam);
		   //Fetch all records using SQL Query
		   $sql = 'SELECT temperature FROM measurements';
		   $result = mysql_query( $sql, $conn );
		   $up_limit = mysql_num_rows($result);
		   $down_limit = (string)($up_limit-1440);
$sql =  "SELECT temperature FROM measurements LIMIT   $down_limit,$up_limit";
		    $result = mysql_query( $sql, $conn );
```

```
            if(! $result )
            {
            die('Could not get data: ' . mysql_error());
            }
            //Pass the SQL Query result to the FusionCharts PHP Class function
            //to set chart data from database
            if ($result)
            {       $temp = 0;
            $arrData = array();
            $tempArray = array();
            $hour = 1;
            $index = 0;
            while($row =  mysql_fetch_array($result)){
            $tempArray[$index] = $row[0];
            $index++;
                                                        }
            $index = 0;
            for($i=0;$i<24;$i++){
            for($j=0;$j<60;$j++){
            $temp = $temp+$tempArray[$index];
            $index++;
                                                        }
            $arrData[$i][0] = $hour;
            $arrData[$i][1] =number_format((float)$temp/60, 2, '.', '');
            $hour++;
            $temp = 0;
                                                        }
            $FC->addChartDataFromArray($arrData);
            }
            mysql_close($conn);
            // Render the chart
            $FC->renderChart();
        ?>
        </CENTER>
    </BODY>
</HTML>
```

Το διάγραμμα ροής της σελίδας:


start
$dbhost = 'localhost'
$dbuser = 'root'
$dbpass = ''
$conn = mysql_connect($dbhost, $dbuser, $dbpass)
! $conn
true
die('Could not connect: ' . mysql_error())
false
mysql_select_db('ptixiaki')
$FC = new FusionCharts("Line2D","1100","500")
$FC->setSwfPath("FusionCharts/")
$strParam="Caption=Temperature Monitoring (in degree C); subCaption=; showBorder=1; numberSuffix=; xAxisName=HOURS;yAxisName=CELSIUS"
$FC->setChartParams($strParam)
$sql = 'SELECT temperature FROM measurements'
$result = mysql_query($sql, $conn)
$up_limit = mysql_num_rows($result)


$down_limit = (string)($up_limit-1440)
$sql = "SELECT temperature FROM measurements LIMIT $down_limit,$up_limit"
$result = mysql_query($sql, $conn)
! $result
true
die('Could not get data: ' . mysql_error())
false
$result
true
$temp = 0
$arrData = array()
$tempArray = array()
$hour = 1
$index = 0
$row = mysql_fetch_array($result)
true
$tempArray[$index] = $row[0]
$index++

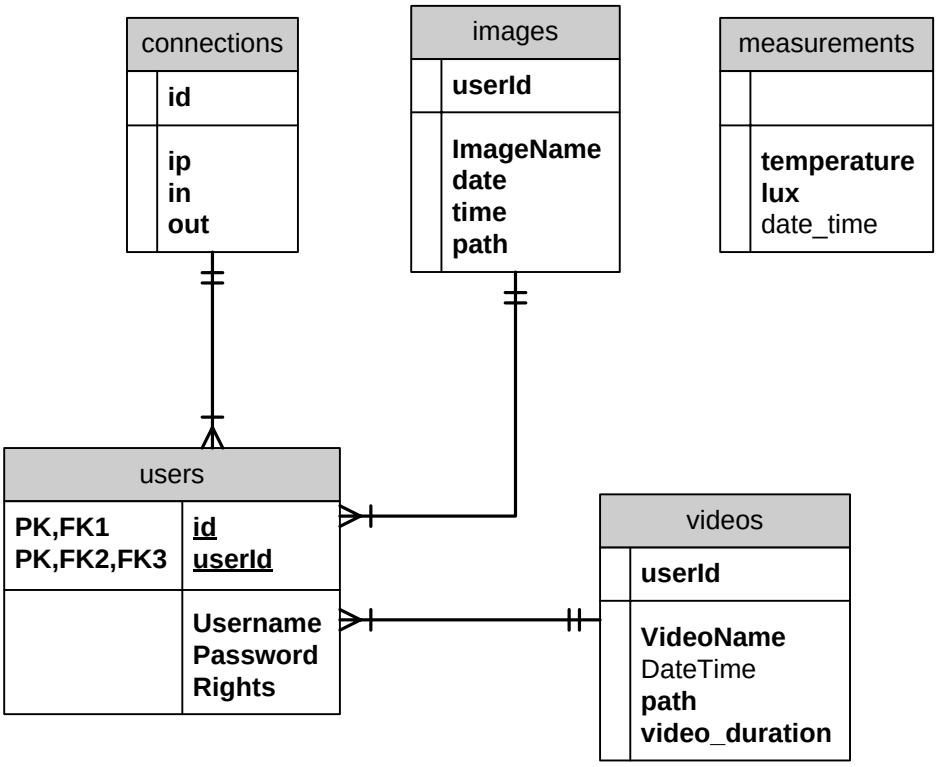


4.10 Η βάση δεδομένων ptixiaki

Η βάση δεδομένων ptixiaki περιέχει τους εξής πέντε πίνακες: users, connections, images, measurements και videos. Ο πίνακας users περιέχει τα πεδία id, Username, Password, Rights. Ο πίνακας connections περιέχει τα πεδία id, ip, in, out. Κάθε φορά που συνδέεται ο χρήστης στην εφαρμογή σε αυτό τον πίνακα αποθηκεύουμε το id του χρήστη το οποίο είναι ίδιο με το id στο πίνακα users, την ip που συνδέεται εκείνη την στιγμή, στο πεδίο in αποθηκεύουμε την ώρα και την ημερομηνία που συνδέθηκε ο χρήστης και στο πεδίο out αποθηκεύουμε την ώρα και την ημερομηνία που αποσυνδέθηκε ο χρήστης από την εφαρμογή. Ο πίνακας images περιέχει τα πεδία userId, ImageName, date, time και path. Στο πεδίο userId αποθηκεύουμε το id του χρήστη που κατέγραψε την εικόνα, στο πεδίο ImageName αποθηκεύουμε το όνομα της εικόνας που κατέγραψε ο χρήστης εκείνη τη στιγμή, στο πεδίο date αποθηκεύουμε την ημερομηνία καταγραφής της εικόνας, στο πεδίο time αποθηκεύουμε την ώρα που κατέγραψε ο χρήστης την εικόνα και στο πεδίο path αποθηκεύουμε την διαδρομή του δίσκου όπου είναι αποθηκευμένη η εικόνα. Ο πίνακας videos περιέχει τα πεδία userId, VideoName, DateTime, path, video_duration. Στο πεδίο userId αποθηκεύουμε το id του χρήστη που κατέγραψε το βίντεο, στο πεδίο VideoName αποθηκεύουμε το όνομα του βίντεο που καταγράφηκε, στο πεδίο DateTime αποθηκεύουμε την ώρα και την ημερομηνία που έγινε η καταγραφή του βίντεο καθώς επίσης αυτό το πεδίο έχει την προεπιλογή current_timestamp, στο πεδίο path αποθηκεύουμε την διαδρομή του δίσκου που είναι αποθηκευμένο το βίντεο και στο πεδίο video_duration αποθηκεύουμε την διάρκεια του βίντεο. Ο πίνακας measurements περιέχει τα

πεδία temperature, lux και date_time. Σε αυτό τον πίνακα αποθηκεύουμε τις μετρήσεις που παίρνουμε από τους αισθητήρες του iLon. Στο πεδίο temperature αποθηκεύουμε την θερμοκρασία, στο πεδίο lux αποθηκεύουμε την φωτεινότητα και στο πεδίο date_time αποθηκεύουμε την ημερομηνία και την ώρα που καταγράφηκαν οι παραπάνω μετρήσεις. Το πεδίο date_time έχει ως προεπιλογή την τιμή 0000-00-00 00:00:00.

Το διάγραμμα της βάσης δεδομένων φαίνεται παρακάτω:

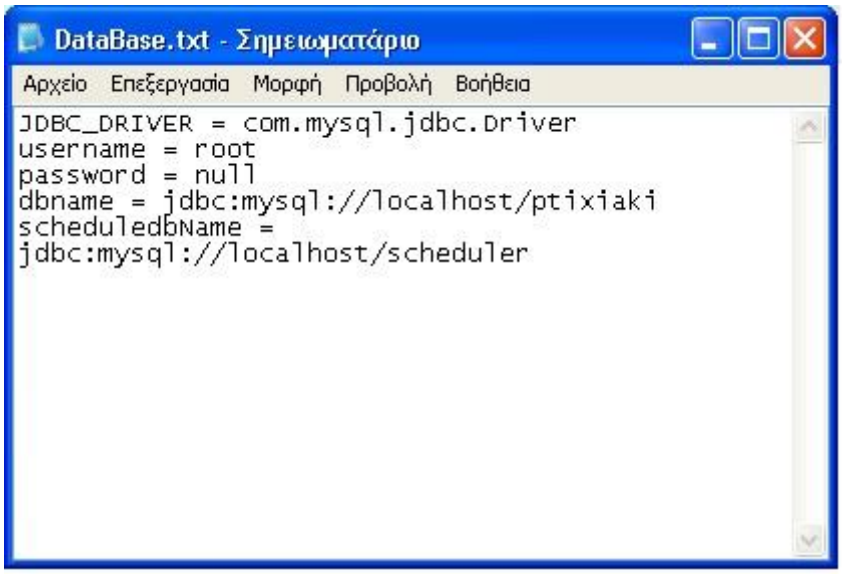


ΚΕΦΑΛΑΙΟ 5

5.1 Απαιτήσεις, εγκατάσταση και λειτουργία της εφαρμογής

Για να λειτουργήσει ο Server θα πρέπει να έχουμε εγκαταστημένη την Java 7 Update 7 ή νεώτερη λόγο ότι η συγκεκριμένη εφαρμογή έχει δημιουργηθεί με κώδικα της Java. Άλλη μία προϋπόθεση για να λειτουργήσει η εφαρμογή είναι να έχουμε μαζί με το εκτελέσιμο αρχείο με όνομα Server.jar στο φάκελο τα αρχεία DataBase.txt και quartz.properties. Το αρχείο DataBase.txt περιέχει τις πληροφορίες για την σύνδεση της εφαρμογής με τις βάσης δεδομένων.

Το αρχείο DataBase.txt:



Στο πεδίο JDBC_DRIVER ορίζουμε τον οδηγό με τον οποίο θα συνδεθούμε με τη βάση δεδομένων. Στην περίπτωσή μας επειδή η βάση δεδομένων είναι mysql χρησιμοποιούμε τον οδηγό com.mysql.jdbc.Driver. Στο πεδίο username συμπληρώνουμε το όνομα χρήστη της βάσης δεδομένων. Στο πεδίο password συμπληρώνουμε τον κωδικό πρόσβασης της βάσης δεδομένων και στη περίπτωση που δεν έχουμε κάποιο κωδικό τότε συμπληρώνουμε τη λέξει null. Στο πεδίο dbname συμπληρώνουμε την ip και το όνομα των βάσεων δεδομένων αφού πρόκειται να συνδεθούμε και με τις δύο.

Το αρχείο quartz.properties:

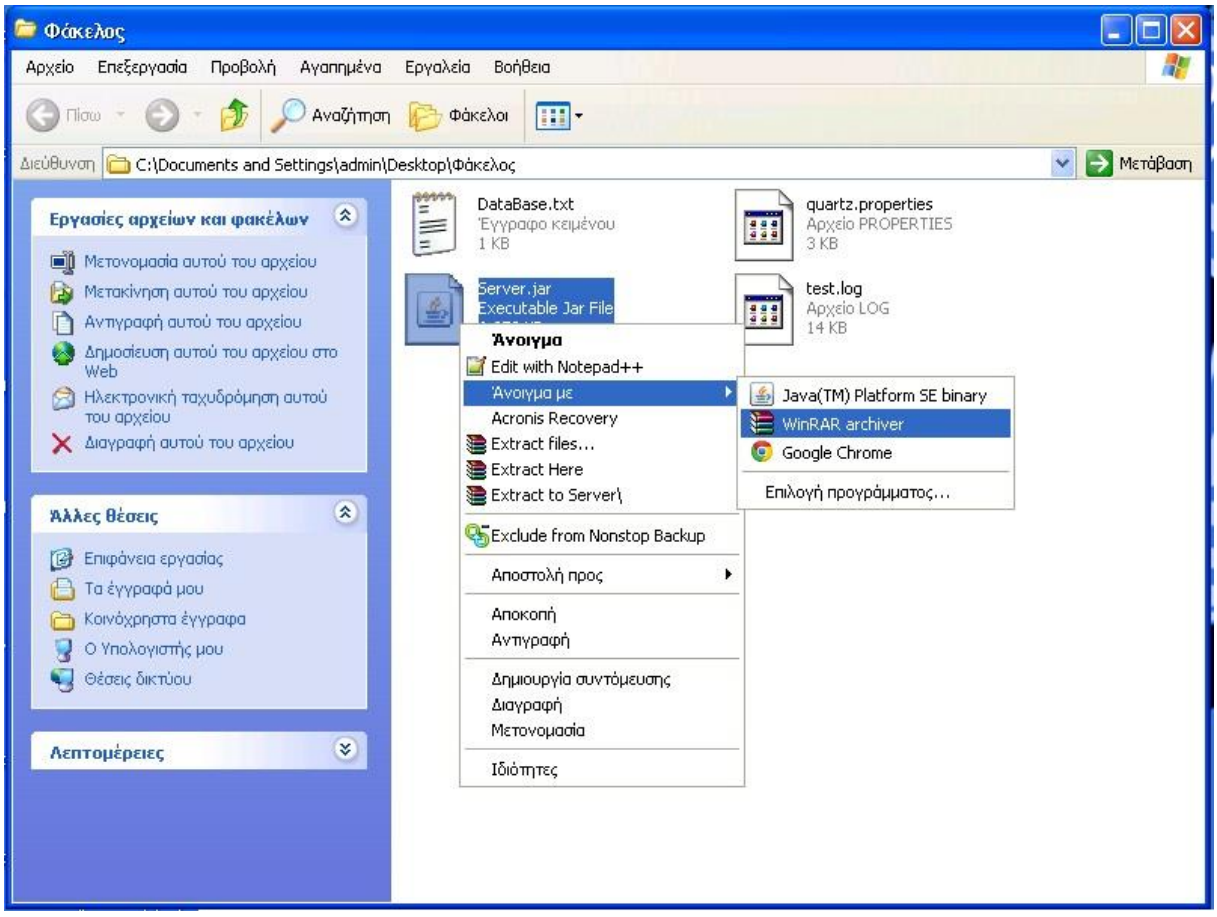
```
#Skip Update Check
#Quartz contains an "update check" feature that connects to a server to
#check if there is a new version of Quartz available
org.quartz.scheduler.skipUpdateCheck: true
org.quartz.scheduler.instanceName =OZS_SCHEDULAR
org.quartz.threadPool.class = org.quartz.simpl.SimpleThreadPool
```

```
org.quartz.threadPool.threadCount = 10
org.quartz.threadPool.threadsInheritContextClassLoaderOfInitializingThread = true
org.quartz.threadPool.threadPriority = 5
#specify the jobstore used
org.quartz.jobStore.misfireThreshold = 1
org.quartz.jobStore.class = org.quartz.impl.jdbcjobstore.JobStoreTX
org.quartz.jobStore.driverDelegateClass = org.quartz.impl.jdbcjobstore.StdJDBCDelegate
org.quartz.jobStore.useProperties = false
#specify the TerracottaJobStore
#org.quartz.jobStore.class = org.terracotta.quartz.TerracottaJobStore
#org.quartz.jobStore.tcConfigUrl = localhost:9510
#The datasource for the jobstore that is to be used
org.quartz.jobStore.dataSource = myDS
#quartz table prefixes in the database
org.quartz.jobStore.tablePrefix = qrtz_
org.quartz.jobStore.misfireThreshold = 1
org.quartz.jobStore.isClustered = false
#The details of the datasource specified previously
org.quartz.dataSource.myDS.driver =com.mysql.jdbc.Driver
org.quartz.dataSource.myDS.URL =jdbc:mysql://localhost:3306/scheduler
org.quartz.dataSource.myDS.user =root
org.quartz.dataSource.myDS.password =
org.quartz.dataSource.myDS.maxConnections = 20
#Clustering
#clustering currently only works with the JDBC-Jobstore (JobStoreTX
#or JobStoreCMT). Features include load-balancing and job fail-over
#(if the JobDetail's "request recovery" flag is set to true). It is important to note that
# When using clustering on separate machines, make sure that their clocks are synchronized
#using some form of time-sync service (clocks must be within a second of each other).
#See http://www.boulder.nist.gov/timefreq/service/its.htm.
# Never fire-up a non-clustered instance against the same set of tables that any
#other instance is running against.
# Each instance in the cluster should use the same copy of the quartz.properties file.
org.quartz.jobStore.isClustered = true
org.quartz.jobStore.clusterCheckinInterval = 20000
#Each server must have the same copy of the configuration file.
#auto-generate instance ids.
org.quartz.scheduler.instanceId = AUTO
#Note :
#If a job throws an exception, Quartz will typically immediately
#re-execute it (and it will likely throw the same exception again).
#It's better if the job catches all exception it may encounter, handle them,
#and reschedule itself, or other jobs. to work around the issue.
```

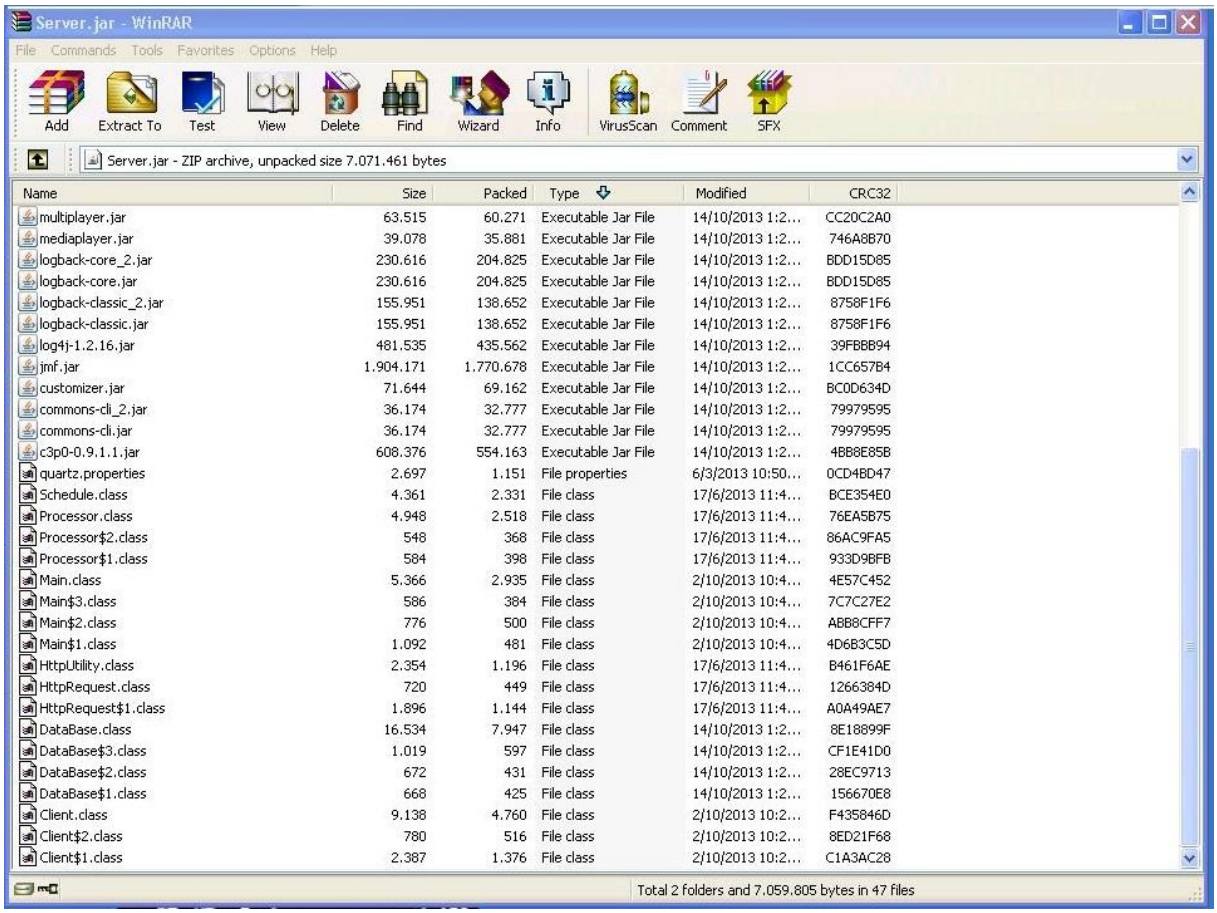
Στο αρχείο quartz.properties τα σημαντικά πεδία που πρέπει να συμπληρώσουμε είναι έξι. Τα υπόλοιπα πεδία τα αφήνουμε ως έχουν. Στο πεδίο org.quartz.threadPool.threadCount ορίζουμε πόσες εργασίες επιτρέπεται να εκτελεί ταυτόχρονα ο χρονοπρογραμματιστής. Τα υπόλοιπα πεδία αφορούν την σύνδεση με τη βάση δεδομένων. Στο πεδίο org.quartz.dataSource.myDS.driver ορίζουμε τον οδηγό με τον οποίο θα συνδεθούμε στη βάση δεδομένων, ο οποίος είναι ο com.mysql.jdbc.Driver. Στο πεδίο com.mysql.jdbc.Driver

ορίζουμε την ip, το port και το όνομα της βάσης δεδομένων τα οποία είναι jdbc:mysql://localhost:3306/scheduler. Στα πεδία org.quartz.dataSource.myDS.user και org.quartz.dataSource.myDS.password ορίζουμε το όνομα χρήστη και το κωδικό πρόσβασης της βάσης δεδομένων. Στη περίπτωση που η βάση δεδομένων δεν έχει κωδικό πρόσβασης αφήνουμε κενό το πεδίο org.quartz.dataSource.myDS.password. Τέλος στο πεδίο org.quartz.dataSource.myDS.maxConnections ορίζουμε πόσες φορές μπορεί να συνδεθεί ταυτόχρονα η εφαρμογή στη βάση δεδομένων. Ο αριθμός αυτός θα πρέπει να είναι μεγαλύτερος ή ίσος από τον αριθμό που έχουμε ορίσει το πεδίο org.quartz.threadPool.threadCount. Για να έχει πρόσβαση η εφαρμογή στο αρχείο quartz.properties θα πρέπει να βρίσκεται μέσα στο εκτελέσιμο αρχείο Server.jar. για να διαπιστώσουμε αν το αρχείο quartz.properties βρίσκεται μέσα στο Server.jar ακολουθούμε τη παρακάτω διαδικασία:

Ανοίγουμε το αρχείο Server.jar με το winrar



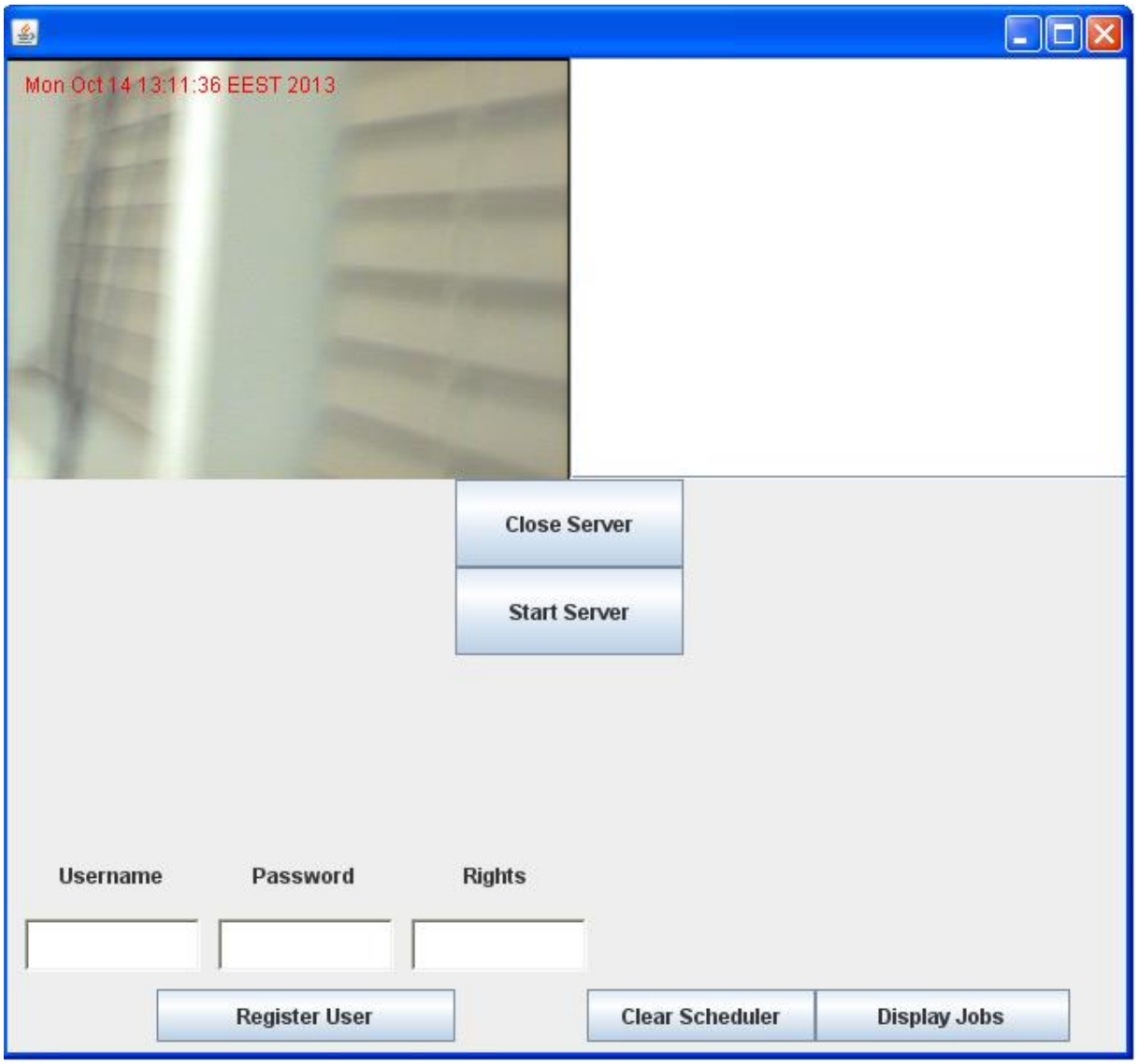
Ελέγχουμε αν υπάρχει το αρχείο στο παρακάτω παράθυρο:



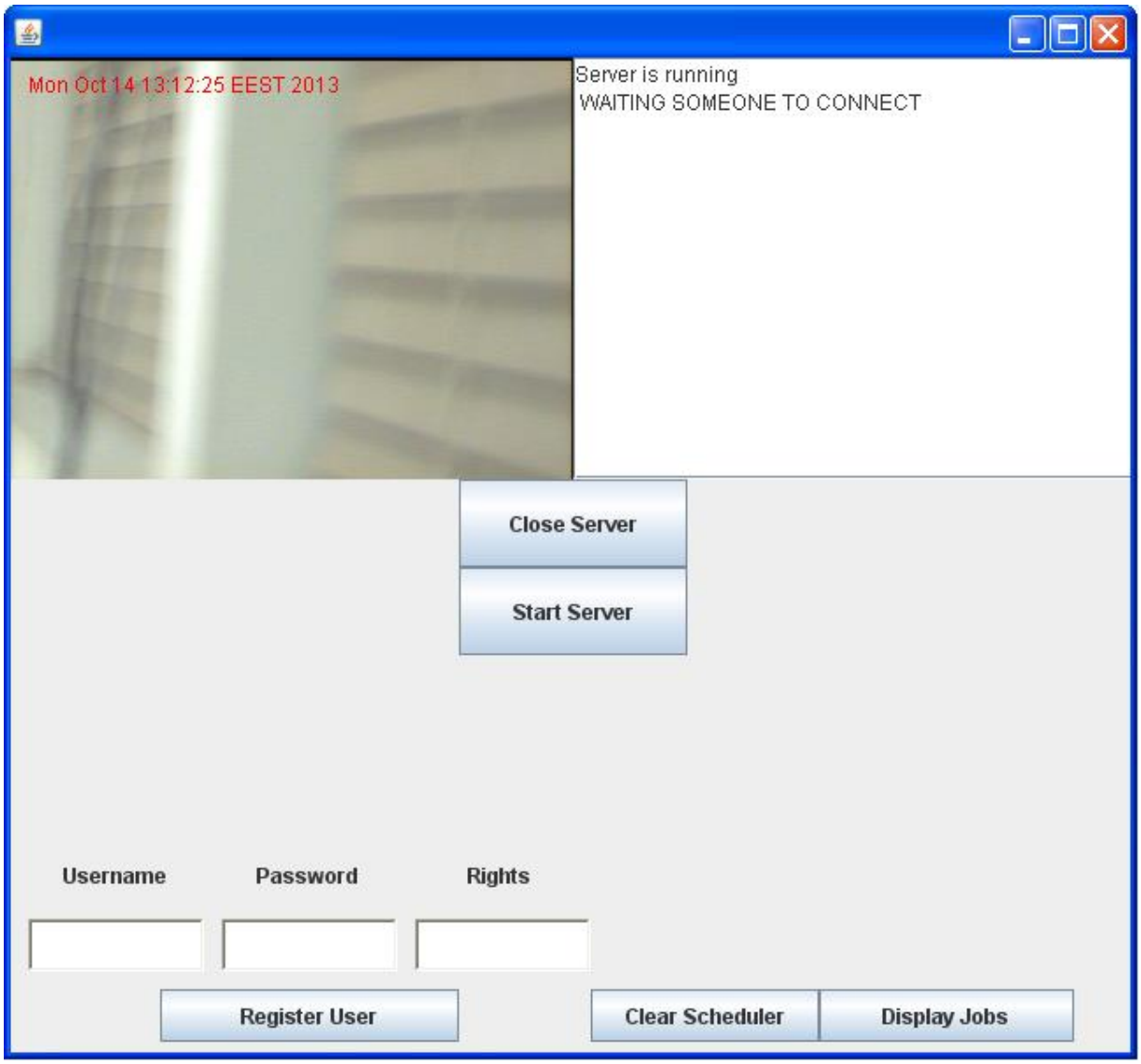
Αν το αρχείο quartz.properties δεν υπάρχει στο Server.jar ακολουθούμε την εξής διαδικασία:

1) Ανοίγουμε την γραμμή εντολών, γράφουμε cd και τη διαδρομή του φακέλου που περιέχει τα δύο αρχεία και πατάμε enter.
2) Στη συνέχεια γράφουμε jar uf Server.jar quartz.properties πατάμε enter και περιμένουμε να ολοκληρωθεί η διαδικασία.

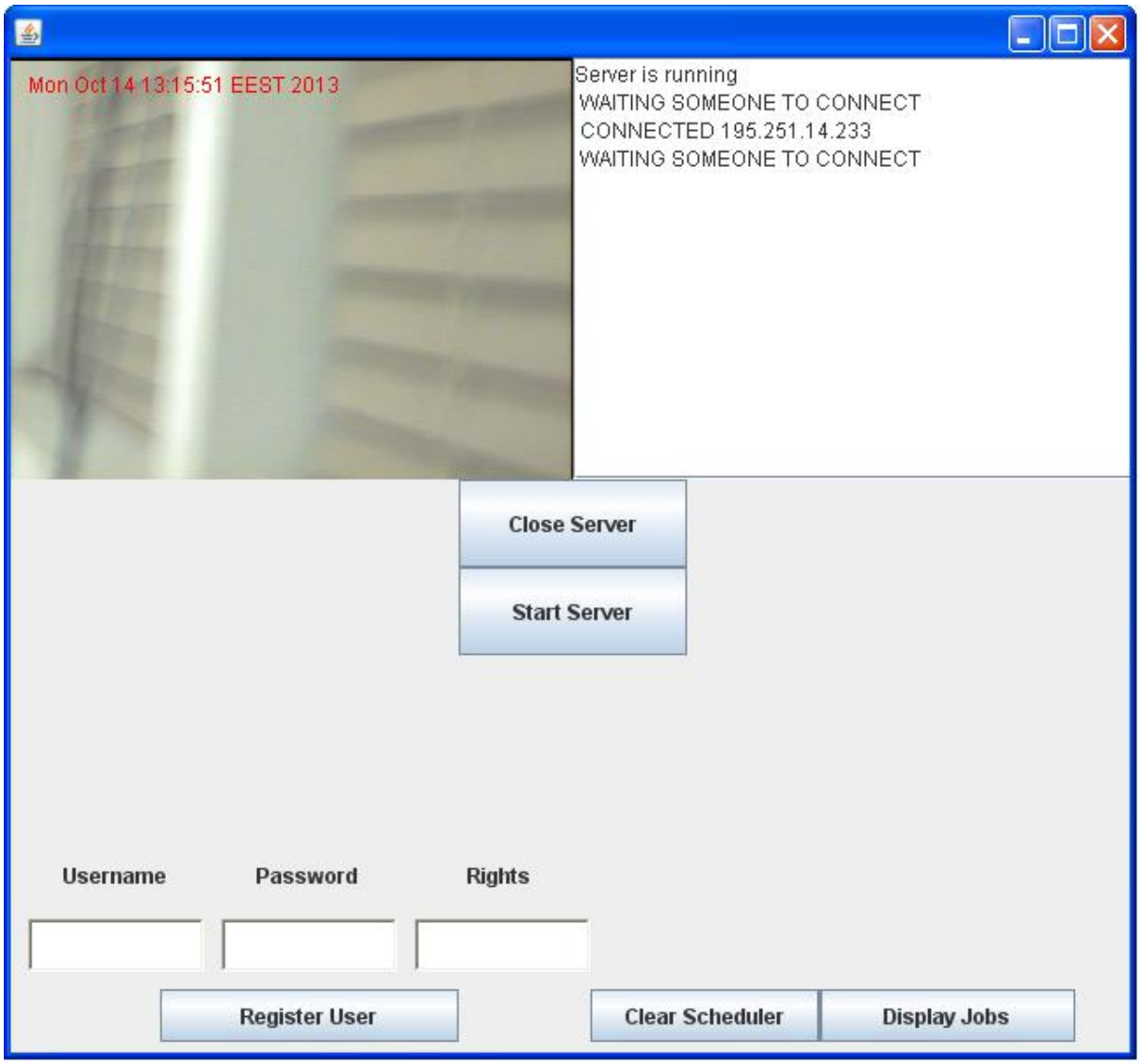
Με την παραπάνω διαδικασία προσθέτουμε το quartz.properties στο Server.jar. Για να ανοίξουμε την εφαρμογή πατάμε διπλό αριστερό κλικ στο Server.jar και εμφανίζεται η παρακάτω εικόνα:



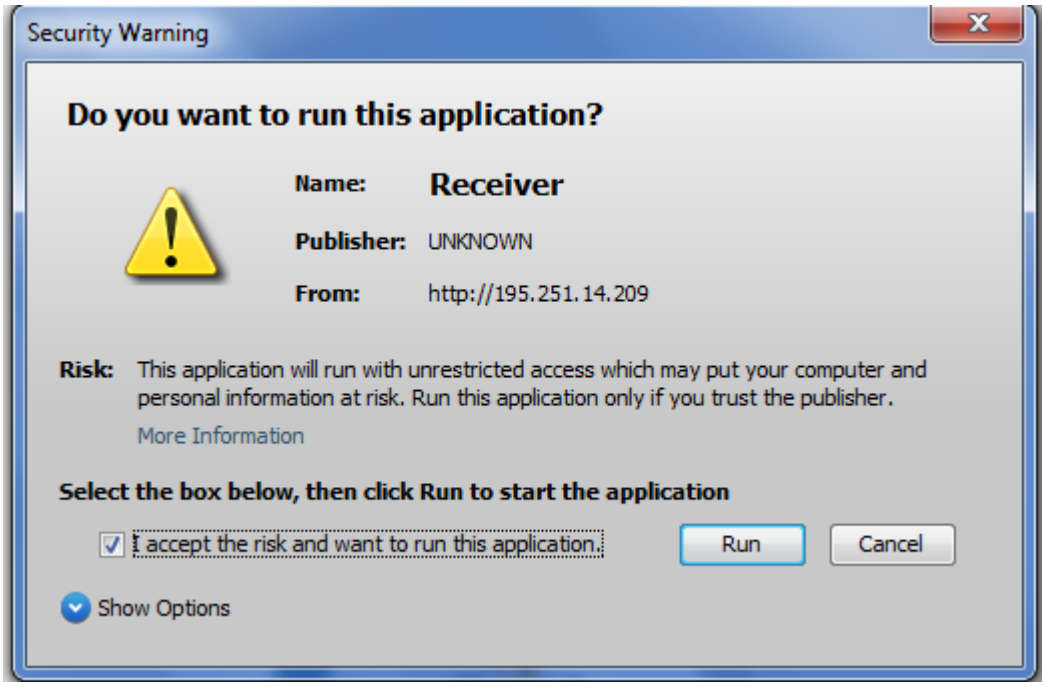
Για να μπορέσει ένας χρήστης να συνδεθεί στην εφαρμογή θα πρέπει να κάνουμε αριστερό κλικ στο κουμπί Start Server. Όταν πατήσουμε το κουμπί Start Server εμφανίζεται το μήνυμα: Server is Running WAITING SOMEONE TO CONNECT το οποίο μας πληροφορεί ότι από αυτή την στιγμή μπορούν να συνδεθούν χρήστες στην εφαρμογή.



Αν κάποιος χρήστης συνδεθεί στην εφαρμογή τότε μας εμφανίζει τη λέξει CONNECTED και δίπλα την ip με την οποία συνδέθηκε.

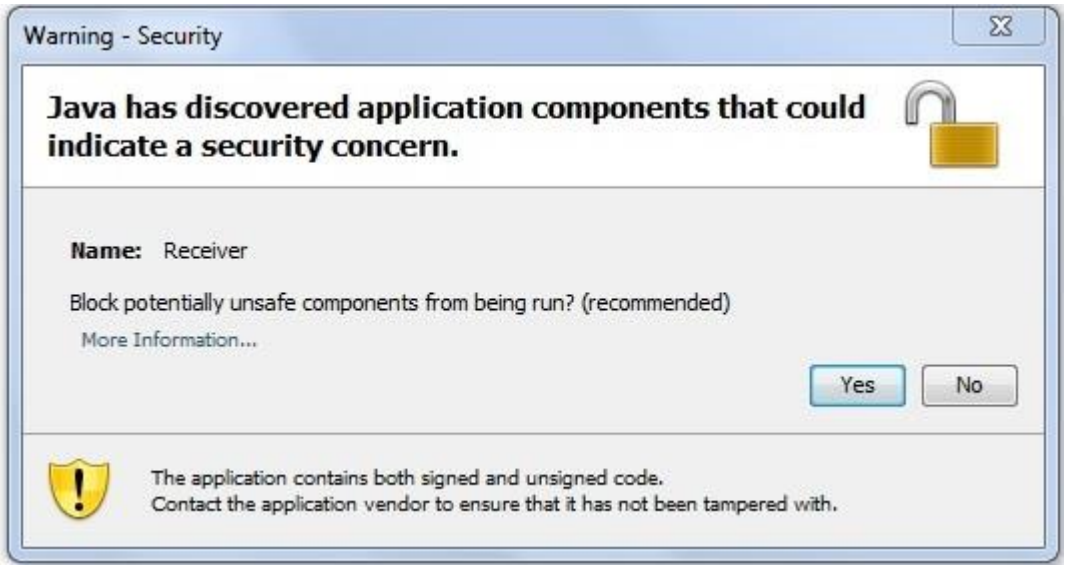


Για να τρέξουμε το applet στον browser θα πρέπει να έχουμε εγκαταστήσει στον υπολογιστή μας την Java 7 Update 7 ή νεότερη έκδοση. Αρχικά μπαίνουμε στη διεύθυνση 195.251.14.209/jar2/fusion και εμφανίζεται η παρακάτω προειδοποίηση:



Επιλέγουμε το κουτί και μετά κάνουμε αριστερό κλικ στο run. Έπειτα εμφανίζεται άλλη μία προειδοποίηση στην οποία κάνουμε αριστερό κλικ στην επιλογή no.

Η δεύτερη προειδοποίηση:

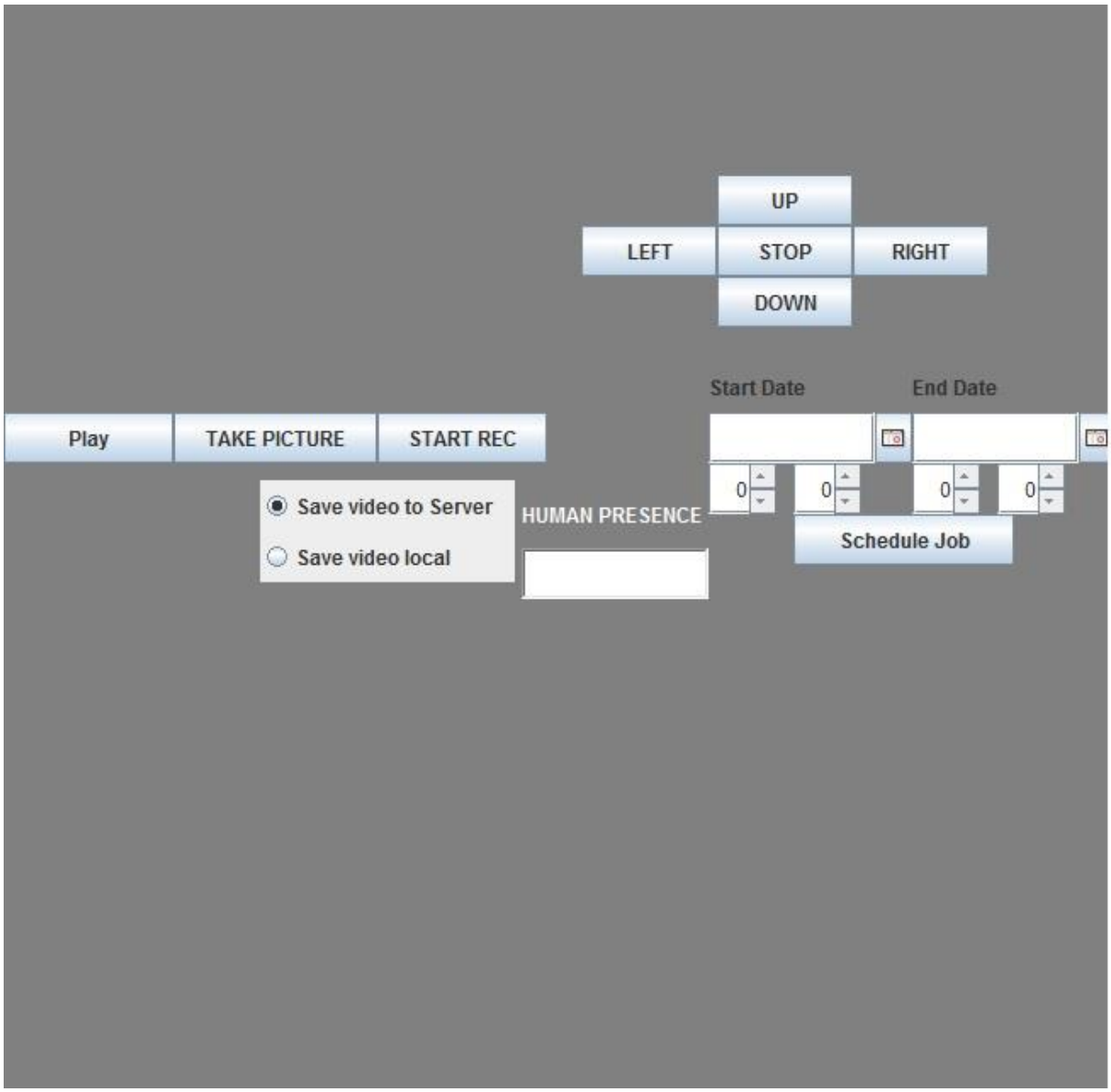


Τέλος περιμένουμε μέχρι να φορτώσει το applet και έπειτα συμπληρώνουμε στα αντίστοιχα πεδία το όνομα χρήστη και τον κωδικό πρόσβασης και κάνουμε αριστερό κλικ στο κουμπί Login.

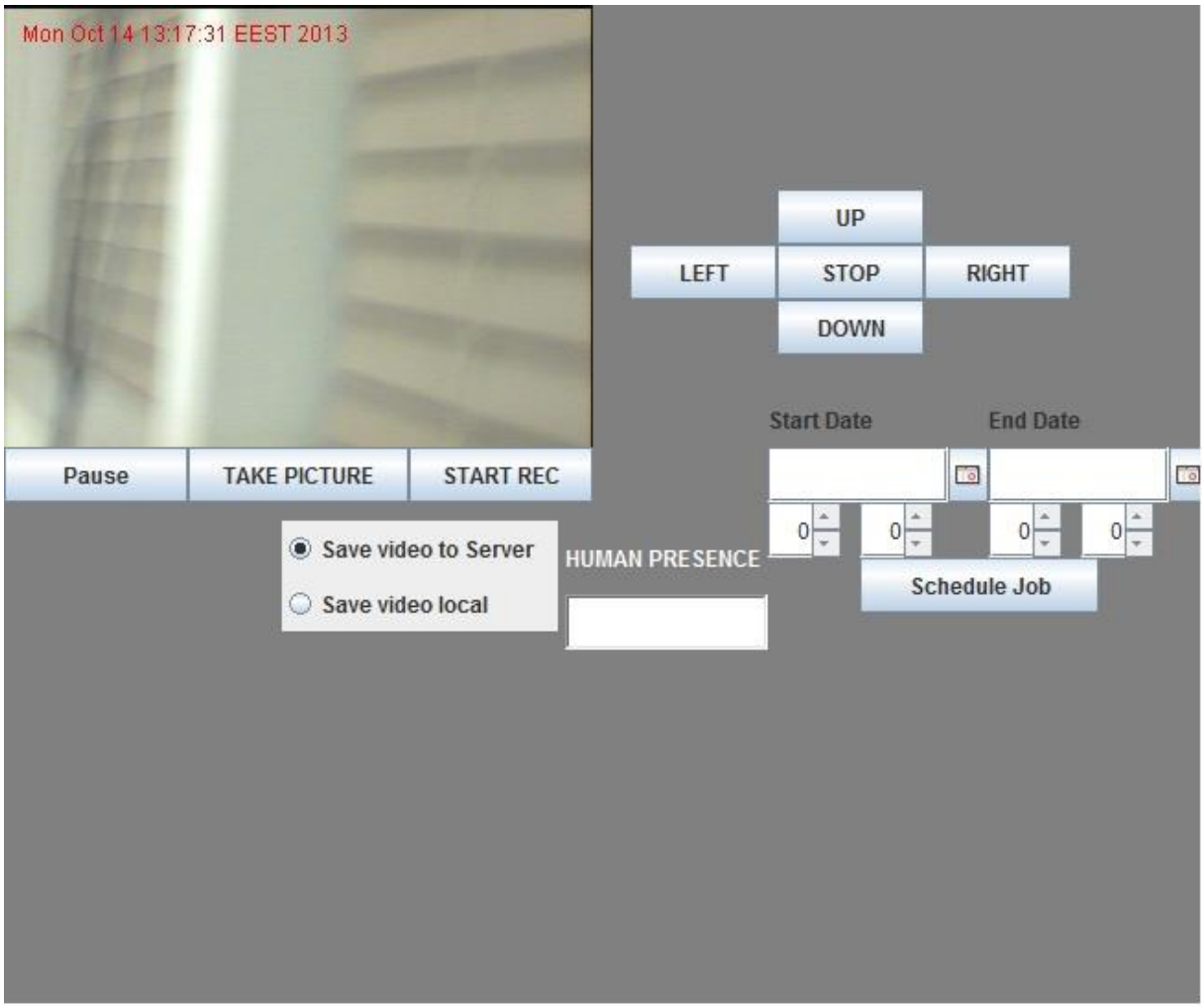
Η φόρμα εισόδου του applet:

Αν τα στοιχεία μας είναι τα σωστά τότε εμφανίζεται το GUI του applet μέσα από το οποίο μπορούμε να αλληλεπιδράσουμε με την εφαρμογή.

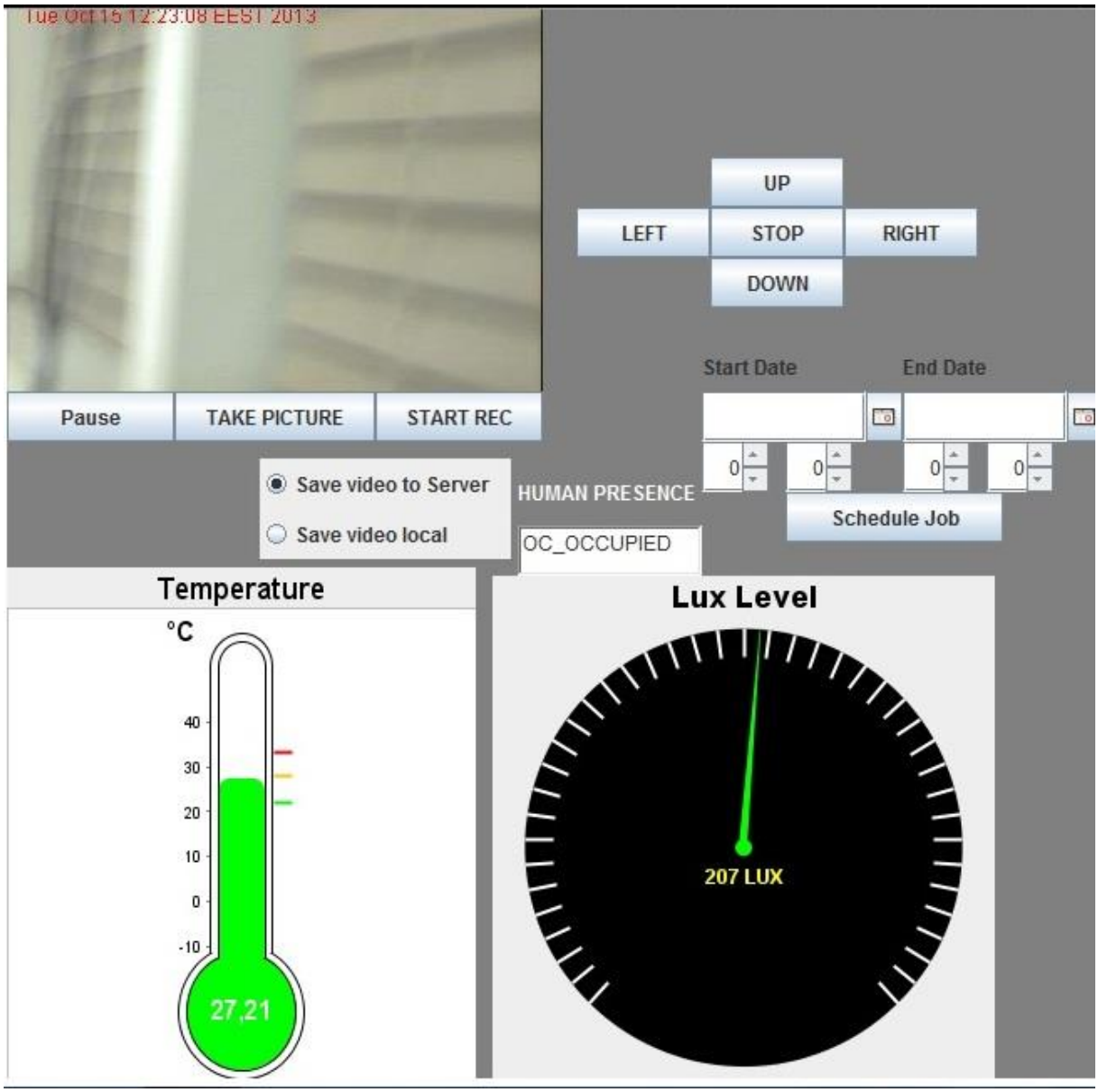
Το GUI του applet:



Αν ο χρήστης πατήσει το κουπί play τότε εμφανίζεται η εικόνα από τη μετάδοση στο πάνω αριστερό μέρος του applet όπως φαίνεται στη παρακάτω εικόνα:



Από όλες τις εικόνες που έχουμε δει μέχρι στιγμής από το GUI του applet απουσιάζει το θερμόμετρο, ο μετρητής των lux και το πεδίο human presence είναι κενό, αυτό συμβαίνει επειδή ο iLon είναι εκτός λειτουργίας. Στη παρακάτω εικόνα ο iLon είναι εντός λειτουργίας:



Για να εγκαταστήσουμε το applet στον http server μέσα στο φάκελο www δημιουργούμε ένα φάκελο με ότι όνομα θέλουμε, στην συγκεκριμένη περίπτωση τον έχουμε ονομάσει jar2. Μέσα στο φάκελο jar2 τοποθετούμε τον φάκελο fusion ο οποίος περιέχει τους φακέλους FusionCharts και Includes καθώς και τα αρχεία Video Statistics.php, UsersStatistics.php, lastHourTemp.php, lastHourLux.php, last24hoursLux.php, last24hours.php, index.html, files.jar και echelon.png. Το αρχείο files.jar περιέχει τις κλάσης του applet.

ΒΙΒΛΙΟΓΡΑΦΙΑ

http://el.wikipedia.org/wiki/Java

http://en.wikipedia.org/wiki/Surveillance

http://en.wikipedia.org/wiki/Database

http://www.javabeat.net/2009/02/volatile-keyword-in-java/

http://stackoverflow.com/questions/6001211/format-of-type-int-rgb-and-type-int-argb

http://www.youtube.com/watch?v=Vrt5LqpH2D0

http://www.xuggle.com/public/documentation/java/api/

http://www.javacodegeeks.com/2011/02/xuggler-tutorial-frames-capture-video.html

http://docs.oracle.com/javase/1.5.0/docs/api/java/io/BufferedReader.html

http://www.vogella.com/articles/MySQLJava/article.html#jdbc

http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Calendar.html

http://docs.oracle.com/javase/7/docs/api/java/util/Calendar.html

http://www.tutorialspoint.com/java/java_applet_basics.htm

http://docs.oracle.com/javase/tutorial/uiswing/components/label.html

http://docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html

http://docs.oracle.com/javase/7/docs/api/java/io/ByteArrayInputStream.html

http://docs.oracle.com/javase/7/docs/api/javax/swing/JRadioButton.html

http://docs.oracle.com/javase/6/docs/api/javax/swing/JFileChooser.html

http://docs.oracle.com/cd/E12058_01/doc/doc.1014/e12030/cron_expressions.htm

http://en.wikipedia.org/wiki/Cron

http://docs.fusioncharts.com/charts/

ΠΑΡΑΡΤΗΜΑ

Ο κώδικας του Server

Η κλάση Main:

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.IOException;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.Vector;
import javax.media.CaptureDeviceInfo;
import javax.media.CaptureDeviceManager;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import org.quartz.SchedulerException;
public class Main extends JFrame implements Runnable{
		/**
		 *
		 */
		private static final long serialVersionUID = 1L;
		String ip;
		int port;
		Thread mainThread;
		ArrayList<Client> clients;
		Client client;
		int index = 0;
		ServerSocket serverSocket;
		Socket connection;
		JPanel panel;
		Processor processor;
		Vector<CaptureDeviceInfo> list;
		CaptureDeviceInfo di = null;
		JScrollPane scrollPane;
		JTextArea textArea;
		JButton startServer,stopServer;
		InetAddress serverIP;
	       DataBase dataBase;
		DatagramSocket serverUDPSocket;
		protected volatile boolean running = false;
		public static void main(String[] args){
		new Main();
		}
		@SuppressWarnings("unchecked")
		public Main(){
			list = CaptureDeviceManager.getDeviceList ( null );
			try {
			serverIP = InetAddress.getLocalHost();
			} catch (UnknownHostException e1) {
			// TODO Auto-generated catch block
			}
			setSize(645,600);
			addWindowListener(new WindowListener(){
				@Override
```

```
		public void windowActivated(WindowEvent e) {
		// TODO Auto-generated method stub
		}
		@Override
		public void windowClosed(WindowEvent e) {
		// TODO Auto-generated method stub
		}
		@Override
		public void windowClosing(WindowEvent e) {
		stop();
		}
		@Override
		public void windowDeactivated(WindowEvent e) {
		// TODO Auto-generated method stub
		}
		@Override
		public void windowDeiconified(WindowEvent e) {
		// TODO Auto-generated method stub
		}
		@Override
		public void windowIconified(WindowEvent e) {
		// TODO Auto-generated method stub
		}
		@Override
		public void windowOpened(WindowEvent e) {
		// TODO Auto-generated method stub

		}
});
textArea = new JTextArea();
scrollPane = new JScrollPane(textArea);
startServer = new JButton("Start Server");
startServer.setSize(130, 50);
startServer.setLocation(255, 290);
startServer.setFocusable(false);
stopServer = new JButton("Close Server");
stopServer.setSize(130, 50);
stopServer.setLocation(255, 240);
stopServer.setFocusable(false);
textArea.setEditable(false);
scrollPane.setSize(320, 240);
scrollPane.setLocation(320, 0);
scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
scrollPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
panel = new JPanel();
panel.setLayout(null);
panel.add(scrollPane);
panel.add(startServer);
panel.add(stopServer);
add(panel);
dataBase  = new DataBase(clients, panel);
//dataBase.Connect();
clients = new ArrayList<Client>();
		try {
		serverSocket = new ServerSocket(8080,50);
		} catch (IOException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
		}
processor = new Processor(panel);
Processor.producerThread();
Processor.consumerThread();
mainThread = new Thread(this);
setVisible(true);
startServer.addActionListener(new ActionListener(){
		@Override
```

```
                        public void actionPerformed(ActionEvent e) {
                        start();
                        }
        });
                stopServer.addActionListener(new ActionListener(){
                        @Override
                        public void actionPerformed(ActionEvent e) {
                        stop();
                        }
                });
                dataBase.dataBaseInfo();
                dataBase.Connect();
                HttpRequest.running = true;
                HttpRequest.sendGetRequest();
        }
        @Override
        public void run() {
                dataBase.findTriggersName();
                while(running){
                        showMessage("\n WAITING SOMEONE TO CONNECT");
                        try {
                        connection = serverSocket.accept();
                        } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
                        port = connection.getPort();
                        ip = connection.getInetAddress().getHostAddress();
                        client = new Client(connection,ip,port,dataBase,processor,serverUDPSocket);
                        showMessage("\n CONNECTED "+connection.getInetAddress().getHostName());
                }
        }
        public void stop(){
                if(!running){
                        try {
                        dataBase.scheduler.shutdown();
                        } catch (SchedulerException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
        HttpRequest.running = false;
        Processor.consumerRunning = false;
        Processor.producerRunning = false;
        processor.p.stop();
        processor.p.close();
        processor.p.deallocate();
        System.exit(0);
              }
if(running){
         try {
                        dataBase.scheduler.shutdown();
                        } catch (SchedulerException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
          HttpRequest.running = false;
         running = false;
         Processor.consumerRunning = false;
                Processor.producerRunning = false;
                processor.p.stop();
                processor.p.close();
                processor.p.deallocate();
           System.exit(0);
}
        }
        public void showMessage(String mes){
```

```
                textArea.append(mes);
        }
        public void start(){
         running = true;
         mainThread.start();
         showMessage("Server is running");
        }
}
```

Η κλάση Processor

```
import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.Date;
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.BlockingQueue;
import javax.imageio.ImageIO;
import javax.media.Buffer;
import javax.media.CannotRealizeException;
import javax.media.Manager;
import javax.media.MediaLocator;
import javax.media.NoPlayerException;
import javax.media.Player;
import javax.media.control.FrameGrabbingControl;
import javax.media.format.VideoFormat;
import javax.media.util.BufferToImage;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;
public class Processor {
        Player p;
        JPanel panel;
        static JLabel label;
        protected static FrameGrabbingControl frameGrabber;
        static Image img;
        static BufferedImage buffImg;
        protected volatile static boolean consumerRunning = false;
        protected volatile static boolean producerRunning = false;
        static BlockingQueue<BufferedImage> images = new ArrayBlockingQueue<BufferedImage>(10);
        static BufferedImage image;
        public Processor(JPanel panel){
                this.panel = panel;
                label = new JLabel();
                label.setSize(320,240);
                label.setLocation(0, 0);
                panel.add(label);
                try {
                p = Manager.createRealizedPlayer(new MediaLocator("vfw://1"));
                } catch (NoPlayerException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                } catch (CannotRealizeException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                }
                p.start();
                frameGrabber = (FrameGrabbingControl)p.getControl("javax.media.control.FrameGrabbingControl");
                consumerRunning = true;
```

```
		producerRunning = true;
	}
	public static void consumerThread() {
	  Thread t1 = new Thread(new Runnable(){
		@Override
		public void run() {
			while(consumerRunning){
			consumer();
			  }
		}
	  });
	  t1.start();
}
	public static void producerThread() {
		 Thread t2 = new Thread(new Runnable(){
			@Override
			public void run() {
				while(producerRunning){
				producer();
					}
			}
		 });
		 t2.start();
	  }
	public static BufferedImage consumer(){
		try {
		Thread.sleep(50);
		} catch (InterruptedException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
		}
							try {
						image = images.take();
						  label.setIcon(new ImageIcon(image));
						  } catch (InterruptedException e) {
						  // TODO Auto-generated catch block
						  e.printStackTrace();
						  }
					label.repaint();
					return image;
	}
	public static void producer(){
		Buffer buf = frameGrabber.grabFrame();
		// Convert frame to an buffered image so it can be processed and saved
	   img = (new BufferToImage((VideoFormat) buf.getFormat()).createImage(buf));
		buffImg = new BufferedImage(320, 240,BufferedImage.TYPE_INT_RGB);
		Graphics2D g = buffImg.createGraphics();
		g.drawImage(img,null,null);
		g.setColor(Color.RED);
		g.drawString(""+new Date(),10,20);
		try {
		images.put(buffImg);
		} catch (InterruptedException e) {
		// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
	public void grabFrame(String path){
		Buffer buf = frameGrabber.grabFrame();
		// Convert frame to an buffered image so it can be processed and saved
		Image img = (new BufferToImage((VideoFormat) buf.getFormat()).createImage(buf));
		buffImg = new BufferedImage(320, 240,BufferedImage.TYPE_INT_RGB);
		Graphics2D g = buffImg.createGraphics();
		g.drawImage(img, null, null);
		g.setColor(Color.RED);
		g.drawString(""+new Date(),10,20);
```

```
                        try {
                        ImageIO.write(buffImg, "jpg", new File(path));
                        } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
    }
}
```

Η κλάση Client

```
import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.Socket;
import java.net.SocketException;
import java.util.concurrent.TimeUnit;
import javax.imageio.ImageIO;
import com.xuggle.mediatool.IMediaWriter;
import com.xuggle.mediatool.ToolFactory;
import com.xuggle.xuggler.ICodec;




public class Client implements Runnable{
    String ip;
    int TcpPort,UDPPort;
    DatagramSocket serverUDPSocket;
    DatagramPacket receivePacket;
    InetAddress IPAddress;
    Thread clientThread;
    byte[] receiveData = new byte[1024];
            byte[] sendData = new byte[20000];
            BufferedImage buffImg;
            byte[] imageInByte;
            DatagramPacket sendPacket;
            ObjectOutputStream output;
            ObjectInputStream input;
            Socket connection;
            String message = "";
            int videoId,id,beginIndex;
            IMediaWriter videoWriter;
            long stopTime,startTime,duration;
            DataBase dataBase;
            volatile boolean record = false,stopRecoeding = false;
            protected volatile boolean running = false;
            String Username = "",Password = "";
            boolean stopRecording = false;
            Thread threadRec;
            Processor processor;
            volatile boolean streaming =false,streamingInit = false;
            public Client(Socket connection, String ip, int TcpPort, DataBase dataBase, Processor processor, DatagramSocket
serverUDPSocket) {
                        this.ip = ip;
                        this.TcpPort = TcpPort;
                        this.connection = connection;
                        this.dataBase = dataBase;
                        this.processor = processor;
                        clientThread = new Thread(this);
```

```
            running = true;
            clientThread.start();
}
@Override
public void run() {
            while(running){
            setupStreams();
           whileConnected();
            }
}
public void setupStreams(){
            try {
            input = new ObjectInputStream(connection.getInputStream());
            output = new ObjectOutputStream(connection.getOutputStream());
            } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            }
}
public void whileConnected(){
            do{
                        try {
                        message = (String) input.readObject();
                        } catch (ClassNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
                        if(message.equals("PLAY")){
                        streaming = true;
                        streaming();
                        }
                        if(message.equals("PAUSE")){
                        streaming = false;
                        }
                        if(message.equals("CAPTURE_VIDEO_START")){
                        videoId = dataBase.findVideoId(id)+1;
                        videoWriter = ToolFactory.makeWriter("c://"+Username+videoId+".mp4");
                        videoWriter.addVideoStream(0, 0, ICodec.ID.CODEC_ID_MPEG4,320, 240);
                        record = true;
                        startTime = System.currentTimeMillis();
                        recordingVideo();
                        threadRec.start();
                        dataBase.storeVideoDBStart(id, Username+videoId);
                        message = "nothing";
                        }
                        if(message.equals("CAPTURE_VIDEO_STOP")){
                        record = false;
                        stopRecording = true;
                        stopTime = System.currentTimeMillis();
                        duration = stopTime - startTime;
                       dataBase.storeVideoDBStop(duration/1000, id);
                        message = "nothing";
                        }
                        if(message.equals("initStreaming")){
                        streaming = true;
                        streaming();
                        message = "nothing";
                        }
                        if(message.equals("GRABFRAME")){
                        int imageId = dataBase.findImageId(id)+1;
                        processor.grabFrame("c://"+Username+imageId+".jpg");
                        dataBase.storeImageDB(id,Username+imageId);
                        message = "nothing";
```

```
                    }
                    if(message.contains("LOGIN")){
                    beginIndex = 5;
                    Character Char = new Character(message.charAt(beginIndex));
                       do{
                       Username = Username+Char;
                       beginIndex++;
                       Char = new Character(message.charAt(beginIndex));
                       }while(!Char.equals(','));
                       beginIndex = beginIndex+1;
                       Char = new Character(message.charAt(beginIndex));
                       do{
                       Password = Password+Char;
                       beginIndex++;
                       Char = new Character(message.charAt(beginIndex));
                       }while(!Char.equals('.'));
                       dataBase.findUser(Username,Password);
                       if(dataBase.findUser(Username, Password)){
                       id = dataBase.id;
                                        try {
                                        serverUDPSocket = new DatagramSocket();
                                        } catch (SocketException e1) {
                                        // TODO Auto-generated catch block
                                        e1.printStackTrace();
                                        }
                                        if(dataBase.findIfAdmin(Username, Password)){
                                try {
                                        output.writeObject("CONFIRMED"+serverUDPSocket.getLocalPort());
                                        output.flush();
                                        } catch (IOException e) {
                                        e.printStackTrace();
                                        }
                                        }else{
                                                try {

output.writeObject("CONFIRMED"+serverUDPSocket.getLocalPort()+"noAdmin");
                                                output.flush();
                                                } catch (IOException e) {
                                                e.printStackTrace();
                                                }
                                        }
                                dataBase.userLogIn(id,connection.getInetAddress().getHostAddress());
                       }else{
                                try {
                                        output.writeObject("DENIED");
                                        output.flush();
                                        Username = "";
                                        Password = "";
                                        } catch (IOException e) {
                                        e.printStackTrace();
                                        }
                       }
                     message = "nothing";
                    }
                    if(message.contains("schedule")){
                    videoId = dataBase.findVideoId(id)+1;
                    String startDateTime = "";
                    String endDateTime = "";
                    String comma = ",";
                    message = message.replace("schedule", "");
                    String[] temp;
                    temp = message.split(comma);
                    startDateTime = temp[0];
                    endDateTime = temp[1];
                    dataBase.ScheduleRec(startDateTime,endDateTime,Username,Username+videoId, id);
                    message = "nothing";
```

```
                        }
                }while(!message.equals("END"));
                if(message.equals("END") && streaming){
                        stopStreaming();
                        try {
                        streaming = false;
                        running  = false;
                        input.close();
                        output.close();
                        connection.close();
System.out.println("TCP Socket is disconnected "+connection.isClosed()+" "+"UDP Socket is disconnected "+serverUDPSocket.isClosed());
                        } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
                        dataBase.userLogOut(id);
                }else if(message.equals("END") && !streaming){
                        try {
                        input.close();
                        output.close();
                        running = false;
                        connection.close();
                        serverUDPSocket.close();
                        dataBase.userLogOut(id);
System.out.println("TCP Socket is disconnected "+connection.isClosed()+" "+"UDP Socket is disconnected "+serverUDPSocket.isClosed());
                        } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
                }
        }
                public void sleep(){
                        try {
                        Thread.sleep(100);
                        } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
                }
                public void streaming(){
                        Thread st = new Thread(new Runnable(){
                                @Override
                                public void run() {
                                        while(!streamingInit){
                                        receivePacket = new DatagramPacket(receiveData, 0,receiveData.length);
                                        try {
                                        serverUDPSocket.receive(receivePacket);
                                        } catch (IOException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                        }
                                IPAddress = receivePacket.getAddress();
                                UDPPort = receivePacket.getPort();
                                System.out.println("connection opened");
                                streamingInit = true;
                                }
                                while(streaming){
                                buffImg = Processor.consumer();
                                ByteArrayOutputStream baos = new ByteArrayOutputStream();
                                try {
                                ImageIO.write(buffImg, "jpg", baos );
                                baos.flush();
                                } catch (IOException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                                }
```

```
                imageInByte = baos.toByteArray();
                sendPacket = new DatagramPacket(imageInByte, 0,imageInByte.length, IPAddress, UDPPort);
                                try {
                                serverUDPSocket.send(sendPacket);
                                } catch (IOException e) {
                                }
                                sleep();
                            }
                        }
                });
                st.start();
        }
        public void stopStreaming(){
                try {
                input.close();
                output.close();
                serverUDPSocket.close();
                running = false;
                } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                }
        }
        public void recordingVideo(){
                threadRec = new Thread(new Runnable(){
                        @Override
                        public void run() {
                                while(record){
                                recVideo();
                                if(stopRecording)
                                videoWriter.close();
                                stopRecording = false;
                                }
                        }
                });
        }
        public void recVideo(){
    // convert to the right image type
for(int i=0; i<25; i++){
   BufferedImage bgrScreen = convertToType(Processor.consumer(),
    BufferedImage.TYPE_3BYTE_BGR);
   // encode the image to stream #0
   videoWriter.encodeVideo(0, bgrScreen, System.nanoTime() - startTime,
       TimeUnit.NANOSECONDS);
      // sleep for frame rate milliseconds
         try {
            Thread.sleep(40);
         }
         catch (InterruptedException e) {
            // ignore
                        }
                }
         }
         public static BufferedImage convertToType(BufferedImage sourceImage, int targetType) {
                  BufferedImage image;
            // if the source image is already the target type, return the source image
                        if (sourceImage.getType() == targetType) {
                           image = sourceImage;
                        }
               // otherwise create a new image of the target type and draw the new image
                  else {
                        image = new BufferedImage(sourceImage.getWidth(),sourceImage.getHeight(), targetType);
                        image.getGraphics().drawImage(sourceImage, 0, 0, null);
                        }
                   return image;
                   }
```

```
}
```

Η κλάση DataBase

```
import static org.quartz.CronScheduleBuilder.cronSchedule;
import static org.quartz.TriggerBuilder.newTrigger;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Time;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.List;
import java.util.TimeZone;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import org.quartz.JobBuilder;
import org.quartz.JobDetail;
import org.quartz.JobKey;
import org.quartz.Scheduler;
import org.quartz.SchedulerException;
import org.quartz.SchedulerFactory;
import org.quartz.Trigger;
import org.quartz.impl.StdSchedulerFactory;
import org.quartz.impl.matchers.GroupMatcher;
public class DataBase {
public Connection con = null , scheduleCon = null;
  static  public Statement statement= null , scheduleStatement = null;
   public String JDBC_DRIVER ;
   public String username,password;
   static String dbname,scheduledbName;
   String sql="select * from users";
   BufferedReader in;
   String name,size,path,ip;
   Date date;
   Time time;
   String dateTime;
   int clientIndex = 0;
   int tableIndex = 0;
   Client c;
   boolean newUser = true;
            ArrayList<Client> clients;
            JLabel UsernameLabel,PasswordLabel,rightsLabel;
            TextField UsernameTF,PasswordTF,rightsTF;
            JButton register,clearSchedule,findJobs;
            int id;
            Thread recThread;
            Scheduler scheduler;
            SchedulerFactory schdFact;
            DateFormat dateFormat;
            Trigger trigger;
```

```
        String jobkey,triggerName = "trigger";
        int triggerKey = 0,key = 0;
public  DataBase(ArrayList<Client> clients, JPanel panel){
        this.clients = clients;
         dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
        dateFormat.setTimeZone(TimeZone.getDefault());
        panel.setLayout(null);
        rightsTF = new TextField(5);
        rightsTF.setSize(100, 30);
        rightsTF.setLocation(230, 490);
        rightsLabel = new JLabel("Rights");
        rightsLabel.setSize(100, 30);
        rightsLabel.setLocation(260, 450);
        UsernameLabel = new JLabel("Username");
        PasswordLabel = new JLabel("Password");
        UsernameTF = new TextField(10);
        PasswordTF = new TextField(10);
        register = new JButton("Register User");
        clearSchedule = new JButton("Clear Scheduler");
        findJobs = new JButton("Display Jobs");
        findJobs.setSize(145, 30);
        findJobs.setLocation(460, 530);
        clearSchedule.setSize(130, 30);
        clearSchedule.setLocation(330, 530);
        UsernameLabel.setSize(100, 30);
        UsernameLabel.setLocation(30, 450);
        PasswordLabel.setSize(100, 30);
        PasswordLabel.setLocation(140, 450);
        UsernameTF.setSize(100, 30);
        UsernameTF.setLocation(10, 490);
        PasswordTF.setSize(100, 30);
        PasswordTF.setLocation(120, 490);
        register.setSize(170, 30);
        register.setLocation(85, 530);
        register.setFocusable(false);
        panel.add(UsernameLabel);
        panel.add(PasswordLabel);
        panel.add(UsernameTF);
        panel.add(PasswordTF);
        panel.add(register);
        panel.add(clearSchedule);
        panel.add(findJobs);
        panel.add(rightsTF);
        panel.add(rightsLabel);
        findJobs.addActionListener(new ActionListener(){
                                @Override
                                public void actionPerformed(ActionEvent e) {
                                        displayJobs();
                                }
        });
        clearSchedule.addActionListener(new ActionListener(){
                                @Override
                                public void actionPerformed(ActionEvent e) {
                                    clearScheduler();
                                }
        });
        register.addActionListener(new ActionListener(){
                                @Override
                                public void actionPerformed(ActionEvent e) {
                                        String username = UsernameTF.getText();
                                        String password = PasswordTF.getText();
                                        String rights = rightsTF.getText();
                                        registerUser(username,password,rights);
                                }
        });
}
```

```
public void Connect() {
   try {
      Class.forName(JDBC_DRIVER);
      con = DriverManager.getConnection(dbname,username,null);
      statement= con.createStatement();
      scheduleCon = DriverManager.getConnection(scheduledbName,username,null);
      scheduleStatement = scheduleCon.createStatement();
      System.out.println ("Database connection established");
   }
   catch (Exception e) {
      JOptionPane.showMessageDialog(null,"not connected "+e.getMessage());
   }
   try {
                                    schdFact = new StdSchedulerFactory("quartz.properties");
                                    scheduler = schdFact.getScheduler();

                  // and start it off
                  scheduler.start();
                        } catch (SchedulerException e2) {
                        // TODO Auto-generated catch block
                        e2.printStackTrace();
                        }
}
public void dataBaseInfo(){
         try {
                                    in = new BufferedReader(new FileReader("DataBase.txt"));
                                    String line;
                                    while((line = in.readLine()) != null)
                                    {
                                             if(line.contains("JDBC_DRIVER")){
                                                      line = line.replace("JDBC_DRIVER", "");
                                                      line = line.replace("=","");
                                                      line = line.replace(" ", "");
                                                      JDBC_DRIVER = line;
                                             }
                                             if(line.contains("username")){
                                                      line = line.replace("username", "");
                                                      line = line.replace("=","");
                                                      line = line.replace(" ", "");
                                                      username = line;
                                             }
                                             if(line.contains("dbname")){
                                                      line = line.replace("dbname", "");
                                                      line = line.replace("=","");
                                                      line = line.replace(" ", "");
                                                      dbname = line;
                                             }
                                             if(line.contains("scheduledbName")){
                                                      line = line.replace("scheduledbName", "");
                                                      line = line.replace("=","");
                                                      line = line.replace(" ", "");
                                                      scheduledbName = line;
                                             }
                                             if(line.contains("password")){
                                                      line = line.replace("password", "");
                                                      line = line.replace("=","");
                                                      line = line.replace(" ", "");
                                                      if(line.contains("null")){
                                                      password = null;
                                             }else{
                                             password = line;
                                                      }
                                             }
                                    }
                        } catch (IOException e) {
```

```
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                    }
  }
        public void registerUser(String username, String password, String rights){
                    ResultSet rs = null;
                    sql = "select id from users";
                    try {
                    rs = statement.executeQuery(sql);
                    rs.last();
                    } catch (SQLException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                    }
                    if(rights == null){
                    rights = "noAdmin";
                    }
                    try {
                    int id = rs.getInt(1)+1;
                    statement.executeUpdate(
"INSERT INTO users("+"id,"+"Username,"+"Password,"+"Rights"+") VALUES("+id+",'"+username+"','"+password+"','"+rights+"')");
                    } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                    }
        }
        public boolean findUser(String username, String password){
                    ResultSet rs = null;
                    sql = "select Username , Password from users";
                    id = 1;
                    try {
                    rs = statement.executeQuery(sql);
                       while(rs.next()){
                                if(rs.getString(1).equals(username) && rs.getString(2).equals(password)){
                                          return true;
                                          }
                                id++;
                       }
                    } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                    }
                    return false;
        }
  public boolean findIfAdmin(String username, String password){
        ResultSet rs = null;
                    sql = "select Username , Password , Rights from users";
                    try {
                    rs = statement.executeQuery(sql);
                       while(rs.next()){
        if(rs.getString(1).equals(username) && rs.getString(2).equals(password) && rs.getString(3).equals("admin")){
                                return true;
                                          }
                       }
                    } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                    }
                    return false;
  }
        public void userLogIn(int userId, String ip){
                    Calendar cal = new GregorianCalendar();
                    String hours = Integer.toString(cal.get(Calendar.HOUR_OF_DAY));
                     String minutes  = Integer.toString(cal.get(Calendar.MINUTE));
                     String seconds = Integer.toString(cal.get(Calendar.SECOND));
                     String hour = hours+":"+minutes+":"+seconds;
```

```
                String day = Integer.toString(cal.get(Calendar.DAY_OF_MONTH));
                String month = Integer.toString(cal.get(Calendar.MONTH)+1);
                String year = Integer.toString(cal.get(Calendar.YEAR));
                String date = year+"-"+month+"-"+day;
                sql = "INSERT INTO connections(id,ip) VALUES("+userId+",'"+ip+"')";
                try {
                statement.executeUpdate(sql);
                } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                }
                sql = "UPDATE connections SET connections.in = '"+date+" "+hour+"' where connections.id = "+userId+" AND
connections.in = '0000-00-00 00:00:00' AND connections.out = '0000-00-00 00:00:00'";
                try {
                        statement.executeUpdate(sql);
                        } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
        }
        public void userLogOut(int userId){
                Calendar cal = new GregorianCalendar();
                String hours = Integer.toString(cal.get(Calendar.HOUR_OF_DAY));
                String minutes  = Integer.toString(cal.get(Calendar.MINUTE));
                String seconds = Integer.toString(cal.get(Calendar.SECOND));
                String hour = hours+":"+minutes+":"+seconds;
                String day = Integer.toString(cal.get(Calendar.DAY_OF_MONTH));
                String month = Integer.toString(cal.get(Calendar.MONTH)+1);
                String year = Integer.toString(cal.get(Calendar.YEAR));
                String date = year+"-"+month+"-"+day;
                sql = "UPDATE connections SET connections.out = '"+date+" "+hour+"' where connections.id = "+userId+" AND
connections.out = '0000-00-00 00:00:00'";
                try {
                statement.executeUpdate(sql);
                } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                }
        }
        public int findImageId(int userId){
                ResultSet rs = null;
                sql = "select userId from images";
                int imageId = 0;
                try {
                        rs = statement.executeQuery(sql);
                        while(rs.next()){
                        if(rs.getInt(1) == userId){
                        imageId++;
                                }
                        }
                } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                }
                return imageId;
        }
        public void storeImageDB(int userId, String imageName){
                Calendar cal = new GregorianCalendar();
                String hours = Integer.toString(cal.get(Calendar.HOUR_OF_DAY));
                String minutes  = Integer.toString(cal.get(Calendar.MINUTE));
                String seconds = Integer.toString(cal.get(Calendar.SECOND));
                String hour = hours+":"+minutes+":"+seconds;
                String day = Integer.toString(cal.get(Calendar.DAY_OF_MONTH));
                String month = Integer.toString(cal.get(Calendar.MONTH)+1);
                String year = Integer.toString(cal.get(Calendar.YEAR));
                String date = year+"-"+month+"-"+day;
```

```
sql = "INSERT INTO images(userId,ImageName,date,time,path)
VALUES("+userId+",'"+imageName+"','"+date+"','"+hour+"','"+"c://"+imageName+".jpg'"+")";
                        try {
                        statement.executeUpdate(sql);
                        } catch (SQLException e) {
                        e.printStackTrace();
                        }
        }
        public int findVideoId(int userId){
                        ResultSet rs = null;
                        sql = "select userId from videos";
                        int videoId = 0;
                        try {
                                rs = statement.executeQuery(sql);
                                while(rs.next()){
                                if(rs.getInt(1) == userId){
                                videoId++;
                                        }
                                }
                        } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
                        return videoId;
        }
        public void storeVideoDBStart(int userId, String videoName){
sql = "INSERT INTO videos(userId,VideoName,path) VALUES("+userId+",'"+videoName+"','"+"c://"+videoName+".mp4'"+")";
                        try {
                        statement.executeUpdate(sql);
                        } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
        }
        public void storeVideoDBStop(long duration, int id){
sql = "UPDATE videos set videos.video_duration = "+duration+" where videos.video_duration = 0 and videos.userId = "+id;
                        try {
                        statement.executeUpdate(sql);
                        } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
        }
        public static void storeVideoDB(String videoName, int userId, long duration){
String sql = "INSERT INTO videos(userId,VideoName,path) VALUES("+userId+",'"+videoName+"','"+"c://"+videoName+".mp4'"+")";
                        try {
                        statement.executeUpdate(sql);
                        } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
sql = "UPDATE videos set videos.video_duration = "+duration+" where videos.video_duration = 0 and videos.userId = "+userId;
                        try {
                        statement.executeUpdate(sql);
                        } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
        }
public void ScheduleRec(String startDateTime, String endDateTime, String username, String videoName, int userId){
                        findJobKey(username);
                        key++;
                        jobkey = "jobKey"+String.valueOf(key);
                        triggerKey++;
                        triggerName = "trigger" + triggerKey;
                        JobDetail job = JobBuilder.newJob(Schedule.class)
```

```
                                .withIdentity(jobkey,username)
                                .usingJobData("Name", videoName)
                                .usingJobData("userId", userId)
                                 .usingJobData("Date", endDateTime)// 2013/03/03 18:31:00
                                 .storeDurably()
                                 .build();
                    trigger = newTrigger()
                   .withIdentity(triggerName, username)
                   .withSchedule(cronSchedule(startDateTime)
                   .withMisfireHandlingInstructionDoNothing())// 0 30 18 3 3 ? 2013
                   .build();
                    try {
                    scheduler.scheduleJob(job,trigger);
                    } catch (SchedulerException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                    }
          }
          public void clearScheduler(){
                    try {
                    scheduler.clear();
                    } catch (SchedulerException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                    }
          }
          @SuppressWarnings("unchecked")
          public void displayJobs(){
                     try {
for (String groupName : scheduler.getJobGroupNames()) {
for (JobKey jobKey : scheduler.getJobKeys(GroupMatcher.jobGroupEquals(groupName))) {
String jobName = jobKey.getName();
String jobGroup = jobKey.getGroup();
//get job's trigger
List<Trigger> triggers = (List<Trigger>) scheduler.getTriggersOfJob(jobKey);
java.util.Date nextFireTime = triggers.get(0).getNextFireTime();
System.out.println("[jobName] : " + jobName + " [groupName] : "+ jobGroup + " - " + nextFireTime);
                                                  }
          }
                                } catch (SchedulerException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                                }
           }
          @SuppressWarnings("unchecked")
          public void findJobKey(String username){
                     try {
                    for (JobKey jobKey : scheduler.getJobKeys(GroupMatcher.jobGroupEquals(username))) {

                                          String jobName = jobKey.getName();
                                           String jobGroup = jobKey.getGroup();
                                          //get job's trigger
                                           List<Trigger> triggers = (List<Trigger>) scheduler.getTriggersOfJob(jobKey);
                                           java.util.Date nextFireTime = triggers.get(0).getNextFireTime();
                                           jobName = jobName.replace("jobKey", "");
                                           key = Integer.parseInt(jobName);
                                           return;
                                                  }
                                } catch (SchedulerException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                                }
           }
          public void findTriggersName(){
                     try {
                ResultSet rs = null;
```

```
                        sql = "select trigger_name from qrtz_cron_triggers";
                           rs = scheduleStatement.executeQuery(sql);
                           if(rs.last()){
                           triggerKey = Integer.parseInt(rs.getString(1).replaceAll("\\D+",""));
                           }else{
                           triggerKey = 0;
                                                   }
                  }
                  catch (SQLException e) {
                   JOptionPane.showMessageDialog(null,"not connected "+e.getMessage());
                  }
                        try {
                        scheduleCon.close();
                        } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
            }
        public static void storeMeasuresinDB(){
        String sqlQuery = "INSERT INTO measurements(temperature, lux)
VALUES('"+Double.parseDouble(HttpRequest.temperature)+"','"+Double.parseDouble(HttpRequest.lux)+"')";
                        try {
                        statement.executeUpdate(sqlQuery);
                        } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
                  }
        }
```

Η κλάση Schedule

```
import java.awt.image.BufferedImage;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.TimeZone;
import java.util.concurrent.TimeUnit;
import org.quartz.Job;
import org.quartz.JobDataMap;
import org.quartz.JobExecutionContext;
import org.quartz.JobExecutionException;
import org.quartz.SchedulerException;
import com.xuggle.mediatool.IMediaWriter;
import com.xuggle.mediatool.ToolFactory;
import com.xuggle.xuggler.ICodec;
public class Schedule implements Job{
        Date date;
        IMediaWriter videoWriter;
        long startTime;
        @Override
        public void execute(JobExecutionContext context) throws JobExecutionException {
        JobDataMap dataMap = context.getJobDetail().getJobDataMap();
           String videoName = dataMap.getString("Name");
           int userId = dataMap.getInt("userId");
           DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
           dateFormat.setTimeZone(TimeZone.getDefault());
           try {
                   date = dateFormat.parse(dataMap.getString("Date"));
                   } catch (ParseException e) {
                   // TODO Auto-generated catch block
                   e.printStackTrace();
                   }
```

```
            Calendar now = Calendar.getInstance();
            Calendar end = Calendar.getInstance();
            end.setTimeZone(TimeZone.getDefault());
            end.setTime(date);
            now.setTimeZone(TimeZone.getDefault());
            now.setTimeInMillis(System.currentTimeMillis());
            videoWriter = ToolFactory.makeWriter("c://"+videoName+".mp4");
            videoWriter.addVideoStream(0, 0, ICodec.ID.CODEC_ID_MPEG4,320, 240);
            startTime = System.currentTimeMillis();
            while(now.before(end)){
            for(int i=0; i<25; i++){
                  BufferedImage bgrScreen = convertToType(Processor.consumer(),
                  BufferedImage.TYPE_3BYTE_BGR);
                  // encode the image to stream #0
                   videoWriter.encodeVideo(0, bgrScreen, System.nanoTime() - startTime,
                   TimeUnit.NANOSECONDS);
                // sleep for frame rate milliseconds
                   try {
                      Thread.sleep(40);
                   }
                   catch (InterruptedException e) {
                     // ignore
                   }
                                    }
             now.setTimeInMillis(System.currentTimeMillis());
            }
                        videoWriter.close();
            try {
                        context.getScheduler().deleteJob(context.getJobDetail().getKey());
                        } catch (SchedulerException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
            long duration = (System.currentTimeMillis()-startTime)/1000;
            DataBase.storeVideoDB(videoName, userId, duration);
          }
          public static BufferedImage convertToType(BufferedImage sourceImage, int targetType) {
                        BufferedImage image;
                  // if the source image is already the target type, return the source image
                              if (sourceImage.getType() == targetType) {
                                 image = sourceImage;
                              }
                 // otherwise create a new image of the target type and draw the new image
                    else {
                              image = new BufferedImage(sourceImage.getWidth(),sourceImage.getHeight(), targetType);
                              image.getGraphics().drawImage(sourceImage, 0, 0, null);
                              }
                              return image;
                    }
}
```

Η κλάση HttpUtility

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
/**
 * This class encapsulates methods for requesting a server via HTTP GET and
 * provides methods for parsing response from the server.
 */
```

```
public class HttpUtility {
    /**
     * Represents an HTTP connection
     */
    private static HttpURLConnection httpConn;
    /**
     * Makes an HTTP request using GET method to the specified URL.
     *
     * @param requestURL
     *            the URL of the remote server
     * @return An HttpURLConnection object
     * @throws IOException
     *             thrown if any I/O error occurred
     */
    public static HttpURLConnection sendGetRequest(String requestURL)
            throws IOException {
        URL url = new URL(requestURL);
        httpConn = (HttpURLConnection) url.openConnection();
        httpConn.setUseCaches(false);
        httpConn.setDoInput(true); // true if we want to read server's response
        httpConn.setDoOutput(false); // false indicates this is a GET request
        return httpConn;
    }
    /**
     * Returns an array of lines from the server's response. This method should
     * be used if the server returns multiple lines of String.
     *
     * @return an array of Strings of the server's response
     * @throws IOException
     *             thrown if any I/O error occurred
     */
    public static String[] readMultipleLinesRespone() throws IOException {
        InputStream inputStream = null;
        if (httpConn != null) {
            inputStream = httpConn.getInputStream();
        } else {
            throw new IOException("Connection is not established.");
        }
 BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));
        List<String> response = new ArrayList<String>();
        String line = "";
        while ((line = reader.readLine()) != null) {
            response.add(line);
        }
        reader.close();
        return (String[]) response.toArray(new String[0]);
    }
    /**
     * Closes the connection if opened
     */
    public static void disconnect() {
        if (httpConn != null) {
            httpConn.disconnect();
        }
    }
}
```

Η κλάση HttpRequest

```
import java.io.IOException;
public class HttpRequest {
    /**
     * This program uses the HttpUtility class to send a GET request
     */
                static String lux,human_presence,temperature;
```

```
	static volatile boolean running = false;
	static Thread sendingRequest;
   public static void sendGetRequest() {
	sendingRequest = new Thread(new Runnable(){
				@Override
				public void run() {
					while(running){
				// sending GET request
				String requestURL = "http://195.251.14.211/forms/hubbell.html";
				try {
				   HttpUtility.sendGetRequest(requestURL);
				   String[] response = HttpUtility.readMultipleLinesRespone();
				   for(int i=0; i<response.length; i++){
if(response[i].contains("&Theta;&Epsilon;&Rho;&Mu;&Omicron;&Kappa;&Rho;&Alpha;&Sigma;&Iota;&Alpha;
&Epsilon;&Rho;&Gamma;&Alpha;&Sigma;&Tau;&Eta;&Rho;&Iota;&Omicron;&Upsilon;<BR>")){
					temperature = response[i+2].replace("</td>", "");
					temperature = temperature.replace(" ", "");
					}
					if(response[i].contains("LUX")){
					lux = response[i+2].replace("</td>", "");
					lux = lux.replace(" ", "");
					}
				   }
				} catch (IOException ex) {
				   ex.printStackTrace();
				}
				DataBase.storeMeasuresinDB();
				try {
						Thread.sleep(60000);
						} catch (InterruptedException e) {
						// TODO Auto-generated catch block
						e.printStackTrace();
						}
				}
			}
	});
	sendingRequest.start();
   }
}
```

Ο κώδικας του Applet

Η κλάση Receiver

```
import java.awt.Color;
import java.awt.Component;
import java.awt.Image;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.Socket;
import java.net.SocketException;
import java.net.URL;
import java.net.UnknownHostException;
import java.text.DateFormat;
```

```
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.imageio.ImageIO;
import javax.swing.ButtonGroup;
import javax.swing.ImageIcon;
import javax.swing.JApplet;
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.filechooser.FileNameExtensionFilter;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.MeterPlot;
import org.jfree.chart.plot.ThermometerPlot;
import org.jfree.data.Range;
import org.jfree.data.general.DefaultValueDataset;
import org.jfree.ui.RectangleInsets;
import com.googlecode.javacv.FFmpegFrameRecorder;
import com.googlecode.javacv.FrameRecorder;
import com.googlecode.javacv.FrameRecorder.Exception;
import com.googlecode.javacv.cpp.opencv_core.IplImage;
import com.toedter.calendar.JDateChooser;
import com.toedter.components.JSpinField;
public class Receiver extends JApplet implements Runnable{
        DatagramSocket clientSocket;
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[20000];
        InetAddress ip;
        InputStream stream;
        DatagramPacket receivePacket;
        JLabel label;
   String path = "";
   JFileChooser fileChooser;
   FileNameExtensionFilter filter1;
   FileNameExtensionFilter filter2;
   long time;
   ChartPanel thermoPanel,panelMeter;
        JFreeChart thermoChart,meterchart;
        ThermometerPlot thermoPlot;
        DefaultValueDataset thermoData,luxData;
        double value = 0;
        MeterPlot meterPlot;
        JDateChooser dateChooserStart,dateChooserEnd;
   DateFormat dateFormat,dayofMonth,month,year; // dayofMonth , month for start the schedule dateFormat to stop the schedule
   JSpinField hourStart,hourEnd,minsStart,minsEnd;
   String dayofMonthString,monthString,yearString;
String serverIP;
Socket controlSocket;
Component comp;
JPanel panel;
String port;
String message;
JLabel USERNAME , PASSWORD , accessDenied,tempLabel,luxLabel,humanpreLabel,endDate,startDate;
TextField userName,password,lux,temperature,human_presence;
JButton left,right,up,down,stop,play,pause,grabFrame,startRecording,stopRecording,logIn,scheduleJob;
boolean connected = false;
boolean initStreaming = false;
boolean Playing = false;
boolean reading = true;
boolean recording = false,notRecording = true, client = false, server = true;
volatile boolean recLocally = false,stopRecLocally = false,takePhoto = false,udpPortOpened = false;
boolean stopLeft=false,stopRight=false,stopUp=false,stopDown=false,turning=false,administrator;
String Username;
```

```
String Password;
ObjectOutputStream output;
ObjectInputStream input;
Thread thread,readThread,recThread;
JRadioButton Server,Client;
ButtonGroup bg;
FrameRecorder recorder;
int serverPort;
   BufferedImage photo;
   String dateStringStart,dateStringEnd,timeStringEnd,timeStringStart,dateTimeStart,dateTimeEnd;
   URL url;
   Image echelonImg;
   JLabel echelonLabel;
            private static final long serialVersionUID = 1L;
            @Override
            public void init(){
scheduleJob = new JButton("Schedule Job");
hourStart = new JSpinField(0,23);
hourStart.setAutoscrolls(true);
hourEnd = new JSpinField(0,23);
hourEnd.setAutoscrolls(true);
minsStart = new JSpinField(0,59);
minsStart.setAutoscrolls(true);
minsEnd = new JSpinField(0,59);
minsEnd.setAutoscrolls(true);
startDate = new JLabel("Start Date");
endDate = new JLabel("End Date");
dateFormat = new SimpleDateFormat("yyyy/MM/dd");
dayofMonth = new SimpleDateFormat("dd");
month = new SimpleDateFormat("MM");
year = new SimpleDateFormat("yyy");
dateChooserStart = new JDateChooser();
dateChooserEnd = new JDateChooser();
thermoData = new DefaultValueDataset(0.0);
thermoPlot = new ThermometerPlot(thermoData);
thermoPlot.setInsets(new RectangleInsets(0,0,0,0));
thermoPlot.setUnits(2);
thermoPlot.setSubrangeInfo(ThermometerPlot.NORMAL, 22.0, 28.0);
      thermoPlot.setSubrangeInfo(ThermometerPlot.WARNING, 28.0,33.0);
      thermoPlot.setSubrangeInfo(ThermometerPlot.CRITICAL, 33.0,45.0);
      thermoPlot.setRange(-10.0, 45.0);
      thermoPlot.setBulbRadius(40);
            thermoData.setValue(value);
            thermoChart = new JFreeChart("Temperature",JFreeChart.DEFAULT_TITLE_FONT,thermoPlot,false);
            thermoPanel = new ChartPanel(thermoChart);
            luxData = new DefaultValueDataset(400.0);
            meterPlot = new MeterPlot(luxData);
            meterPlot.setUnits("LUX");
            meterPlot.setRange(new Range(0.0,400.0));
            luxData.setValue(200.0);
            meterchart = new JFreeChart("Lux Level",meterPlot);
            meterchart.setBackgroundImageAlpha((float) 0.75);
            panelMeter = new ChartPanel(meterchart);
            filter1 = new FileNameExtensionFilter("JPEG Images","jpg","jpeg");
            filter2 = new FileNameExtensionFilter("AVI Videos","avi");
            fileChooser = new JFileChooser();
            label = new JLabel();
            label.setSize(320, 240);
            label.setLocation(0, 0);
            Server = new JRadioButton("Save video to Server",true);
            Server.setSize(150, 30);
            Server.setLocation(151, 280);
            Client  = new JRadioButton("Save video local",false);
            Client.setSize(150, 30);
            Client.setLocation(151, 310);
            bg = new ButtonGroup();
```

```
        bg.add(Client);
        bg.add(Server);
        left = new JButton("LEFT");
        left.setSize(80, 30);
        left.setLocation(340, 130);
        left.setFocusable(false);
        stop = new JButton("STOP");
        stop.setSize(80, 30);
        stop.setLocation(420, 130);
        stop.setFocusable(false);
        right = new JButton("RIGHT");
        right.setSize(80, 30);
        right.setLocation(500, 130);
        right.setFocusable(false);
        up =new JButton("UP");
        up.setSize(80, 30);
        up.setLocation(420, 100);
        up.setFocusable(false);
        down = new JButton("DOWN");
        down.setSize(80, 30);
        down.setLocation(420, 160);
        down.setFocusable(false);
        thread = new Thread();
        humanpreLabel = new JLabel("HUMAN PRESENCE");
        humanpreLabel.setForeground(Color.WHITE);
        human_presence = new TextField(13);
        play = new JButton("Play");
        play.setFocusable(false);
        pause = new JButton("Pause");
        pause.setFocusable(false);
        grabFrame = new JButton("TAKE PICTURE");
        startRecording = new JButton("START REC");
        stopRecording = new JButton("STOP REC");
        USERNAME = new JLabel("USERNAME");
        PASSWORD = new JLabel("PASSWORD");
        accessDenied = new JLabel("INVALID USERNAME OR PASSWORD");
        accessDenied.setSize(250, 30);
        accessDenied.setLocation(10, 200);
        accessDenied.setForeground(Color.RED);
        USERNAME.setSize(100, 30);
        USERNAME.setLocation(295, 178);
        USERNAME.setForeground(Color.WHITE);
        PASSWORD.setSize(100, 30);
        PASSWORD.setLocation(295, 228);
        PASSWORD.setForeground(Color.WHITE);
        logIn = new JButton("Log In");
        logIn.setSize(100, 30);
        logIn.setLocation(275, 298);
        logIn.setFocusable(false);
        userName = new TextField(10);
        userName.setSize(100,30);
        userName.setLocation(275, 203);
        userName.setEditable(true);
        password = new TextField(10);
        password.setSize(100, 30);
        password.setEchoChar('*');
        password.setLocation(275, 253);
        panelMeter.setSize(300, 300);
        panelMeter.setLocation(290, 350);
        thermoPanel.setSize(280, 320);
thermoPanel.setLocation(0, 345);
            humanpreLabel.setSize(130, 30);
            humanpreLabel.setLocation(305, 285);
            human_presence.setSize(110, 30);
            human_presence.setLocation(305, 320);
            dateChooserStart.setSize(120,30);
```

```
		dateChooserStart.setLocation(415, 240);
		hourStart.setSize(40, 30);
		hourStart.setLocation(415, 270);
		minsStart.setSize(40, 30);
		minsStart.setLocation(465, 270);
		startDate.setSize(100, 30);
		startDate.setLocation(415, 210);
		dateChooserEnd.setSize(120, 30);
		dateChooserEnd.setLocation(535, 240);
		hourEnd.setSize(40, 30);
		hourEnd.setLocation(535, 270);
		minsEnd.setSize(40, 30);
		minsEnd.setLocation(585, 270);
		endDate.setSize(100, 30);
		endDate.setLocation(535, 210);
		scheduleJob.setSize(130, 30);
		scheduleJob.setLocation(465, 300);
			play.setFocusable(false);
			play.setSize(100, 30);
			play.setLocation(0, 240);
			pause.setSize(100, 30);
			pause.setLocation(0, 240);
			grabFrame.setSize(120, 30);
			grabFrame.setLocation(100, 240);
			grabFrame.setFocusable(false);
			startRecording.setSize(100, 30);
			startRecording.setLocation(220, 240);
			stopRecording.setSize(100, 30);
stopRecording.setLocation(220, 240);
startRecording.setFocusable(false);
panel = new JPanel();
panel.setLayout(null);
url = getDocumentBase();
serverIP = url.getHost();
echelonImg = getImage(url,"echelon.png");
	echelonLabel = new JLabel(new ImageIcon(echelonImg));
	echelonLabel.setSize(624, 158);
	echelonLabel.setLocation(13, 10);
			panel.add(userName);
			panel.add(password);
			panel.add(logIn);
			panel.add(USERNAME);
			panel.add(PASSWORD);
			panel.add(echelonLabel);
			panel.setBackground(Color.gray);
			add(panel);
			play.addActionListener(new ActionListener(){
				@Override
				public void actionPerformed(ActionEvent e) {
					if(!initStreaming){
					streamingInitialize();
					}else{
						try {
						output.writeObject("PLAY");
						Playing = true;
						streaming();
						} catch (IOException e1) {
						// TODO Auto-generated catch block
						e1.printStackTrace();
						}
					panel.remove(play);
					panel.repaint();
					panel.add(pause);
					}
				}
			});
```

```
pause.addActionListener(new ActionListener(){
	@Override
	public void actionPerformed(ActionEvent e) {
		try {
		output.writeObject("PAUSE");
		Playing = false;
		} catch (IOException e1) {
		// TODO Auto-generated catch block
		e1.printStackTrace();
		}
		panel.remove(pause);
		panel.repaint();
		panel.add(play);
	}
});
grabFrame.addActionListener(new ActionListener(){
	@Override
	public void actionPerformed(ActionEvent e) {
		if(server){
		try {
		output.writeObject("GRABFRAME");
		} catch (IOException e1) {
		// TODO Auto-generated catch block
		e1.printStackTrace();
		}
	  }
		if(client){
			takePhoto = true;
			fileChooser.setFileFilter(filter1);
			int saveResult = fileChooser.showSaveDialog(null);
			if(saveResult == JFileChooser.APPROVE_OPTION){
				path = fileChooser.getSelectedFile().getAbsolutePath()+".jpg";
				path = path.replace("\\", "/");
				if(photo != null)
				try {
				ImageIO.write(photo, "jpg", new File(path));
				} catch (IOException e1) {
				// TODO Auto-generated catch block
				e1.printStackTrace();
				}
			}
		}
	}
});
startRecording.addActionListener(new ActionListener(){
	@Override
	public void actionPerformed(ActionEvent e) {
		if(server){
			try {
			output.writeObject("CAPTURE_VIDEO_START");
			} catch (IOException e1) {
			// TODO Auto-generated catch block
			e1.printStackTrace();
			}
		}
		if(client){
			int saveResult = fileChooser.showSaveDialog(null);
			if(saveResult == JFileChooser.APPROVE_OPTION){
			path = fileChooser.getSelectedFile().getAbsolutePath()+".avi";
			path = path.replace("\\", "/");
				try {
		recorder = FFmpegFrameRecorder.createDefault(path, 320, 240);
		recorder.setFrameRate(23);
		recorder.start();
		time = System.currentTimeMillis();
			recLocally = true;
```

```
                                        } catch (Exception e1) {
                                                // TODO Auto-generated catch block
                                                e1.printStackTrace();
                                                        }
                                                }
                                }
                                panel.remove(startRecording);
                                panel.repaint();
                                panel.add(stopRecording);
                        }
});
stopRecording.addActionListener(new ActionListener(){
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                if(server){
                                        try {
                                        output.writeObject("CAPTURE_VIDEO_STOP");
                                        } catch (IOException e1) {
                                        // TODO Auto-generated catch block
                                        e1.printStackTrace();
                                        }
                                }
                                if(client){
                                        recLocally = false;
                                        try {
                                        recorder.stop();
                                        recorder.release();
                                        } catch (Exception e1) {
                                        // TODO Auto-generated catch block
                                        e1.printStackTrace();
                                        }
                                }
                        panel.remove(stopRecording);
                        panel.repaint();
                        panel.add(startRecording);
                        }
});
logIn.addActionListener(new ActionListener(){
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                Username = userName.getText();
                                Password = password.getText();
                                try {
                                output.writeObject("LOGIN"+Username+","+Password+".");
                                } catch (IOException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                                }
                                try {
                                message = (String) input.readObject();
                                } catch (ClassNotFoundException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                                } catch (IOException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                                }
                                if(message.contains("CONFIRMED")){
                                        message  = message.replace("CONFIRMED", "");
                                        if(message.contains("noAdmin")){
                                        administrator = false;
                                        message = message.replace("noAdmin", "");
                                        }
                                        else{
                                        administrator = true;
                                        }
```

```
                                        serverPort = Integer.parseInt(message);
                                        System.out.println(message);
                                        panel.removeAll();
                                        panel.repaint();
                                        GUI();
                                }
                                if(message.equals("DENIED")){
                                panel.repaint();
                                panel.add(accessDenied);
                                }
                        }
                });
                left.addActionListener(new ActionListener(){
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                if(!turning)
HttpUtilityTester.turnCamera("http://195.251.14.211/forms/ILON%201000/ilon%201000%20IMAGE/Web/forms/camera.html?ILONWEB
_URL=%2Fforms%2Fpantilt11.html&NVL_panL=1.0+1");
                        turning = true;
                        stopLeft = true;
                        }
                });
                right.addActionListener(new ActionListener(){
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                if(!turning)
HttpUtilityTester.turnCamera("http://195.251.14.211/forms/ILON%201000/ilon%201000%20IMAGE/Web/forms/camera.html?ILONWEB
_URL=%2Fforms%2Fpantilt11.html&NVL_panR=1.0+1");
                        turning = true;
                        stopRight = true;
                        }
                });
                up.addActionListener(new ActionListener(){
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                if(!turning)
HttpUtilityTester.turnCamera("http://195.251.14.211/forms/ILON%201000/ilon%201000%20IMAGE/Web/forms/camera.html?ILONWEB
_URL=%2Fforms%2Fpantilt11.html&NVL_tiltUp=1.0+1");
                        turning = true;
                        stopUp = true;
                        }
                });
                down.addActionListener(new ActionListener(){
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                if(!turning)
HttpUtilityTester.turnCamera("http://195.251.14.211/forms/ILON%201000/ilon%201000%20IMAGE/Web/forms/camera.html?ILONWEB
_URL=%2Fforms%2Fpantilt11.html&NVL_tiltD=1.0+1");
                        turning = true;
                        stopDown = true;
                        }
                });
                stop.addActionListener(new ActionListener(){
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                if(stopLeft){
HttpUtilityTester.turnCamera("http://195.251.14.211/forms/ILON%201000/ilon%201000%20IMAGE/Web/forms/camera.html?ILONWEB
_URL=%2Fforms%2Fpantilt11.html&NVL_panL=0.0+0");
                                turning = false;
                                stopLeft = false;
                                }else if(stopRight){
HttpUtilityTester.turnCamera("http://195.251.14.211/forms/ILON%201000/ilon%201000%20IMAGE/Web/forms/camera.html?ILONWEB
_URL=%2Fforms%2Fpantilt11.html&NVL_panR=0.0+0");
                                turning = false;
                                stopRight = false;
```

```
                                        }else
if(stopUp){HttpUtilityTester.turnCamera("http://195.251.14.211/forms/ILON%201000/ilon%201000%20IMAGE/Web/forms/camera.html?I
LONWEB_URL=%2Fforms%2Fpantilt11.html&NVL_tiltUp=0.0+0");
                                        turning = false;
                                        stopUp = false;
                                        }else if(stopDown){
HttpUtilityTester.turnCamera("http://195.251.14.211/forms/ILON%201000/ilon%201000%20IMAGE/Web/forms/camera.html?ILONWEB
_URL=%2Fforms%2Fpantilt11.html&NVL_tiltD=0.0+0");
                                        turning = false;
                                        stopDown = false;
                                        }
                                }
                        });
                        Server.addActionListener(new ActionListener(){
                                @Override
                                public void actionPerformed(ActionEvent e) {
                                        server = true;
                                        client = false;
                                }
                        });
                        Client.addActionListener(new ActionListener(){
                                @Override
                                public void actionPerformed(ActionEvent e) {
                                        client = true;
                                        server = false;
                                }
                        });
            dateChooserStart.addPropertyChangeListener(new java.beans.PropertyChangeListener() {
          public void propertyChange(java.beans.PropertyChangeEvent evt) {
            dateStartOnlyPopupChanged(evt);
          }
        });
dateChooserEnd.addPropertyChangeListener(new java.beans.PropertyChangeListener() {
          public void propertyChange(java.beans.PropertyChangeEvent evt) {
            dateEndtOnlyPopupChanged(evt);
          }
        });
                        scheduleJob.addActionListener(new ActionListener(){
                                @Override
                                public void actionPerformed(ActionEvent e) {
dateTimeStart = "0"+" "+minsStart.getValue()+" "+hourStart.getValue()+" "+dayofMonthString+" "+monthString+" ? "+yearString;
                                dateTimeEnd = dateStringEnd+" "+hourEnd.getValue()+":"+minsEnd.getValue()+":00";
                                try {
                                        output.writeObject("schedule"+dateTimeStart+","+dateTimeEnd);
                                        } catch (IOException e1) {
                                        // TODO Auto-generated catch block
                                        e1.printStackTrace();
                                        }
                                        }
                        });
                }
private void dateStartOnlyPopupChanged(java.beans.PropertyChangeEvent evt) {
   if (evt.getNewValue() instanceof Date)
      setDateStart((Date)evt.getNewValue());
}
public void setDateStart(Date date)
{
   dayofMonthString = "";
   monthString = "";
   yearString = "";
   if (date != null){
     dayofMonthString = dayofMonth.format(date);
     monthString = month.format(date);
     yearString = year.format(date);
   }
}
```

```
private void dateEndtOnlyPopupChanged(java.beans.PropertyChangeEvent evt) {
   if (evt.getNewValue() instanceof Date)
      setDateEnd((Date)evt.getNewValue());
}
public void setDateEnd(Date date)
{
   dateStringEnd = "";
   if (date != null)
dateStringEnd = dateFormat.format(date);
   System.out.println(dateStringEnd);
}
            public void streamingInitialize(){
                        try {
                        output.writeObject("initStreaming");
                        } catch (IOException e3) {
                        // TODO Auto-generated catch block
                        e3.printStackTrace();
                        }
                        try {
                        clientSocket = new DatagramSocket();
                        } catch (SocketException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
                        Playing = true;
                        streaming();
                        try {
                        long sendTime = System.currentTimeMillis();
                           while(!udpPortOpened){
                                    sendData = "hello world".getBytes();
                                    ip = InetAddress.getByName(serverIP);
                                    DatagramPacket sendPacket = new DatagramPacket(sendData,sendData.length,ip,serverPort);
                                    clientSocket.send(sendPacket);
                                    try {
                                    Thread.sleep(10);
                                    } catch (InterruptedException e) {
                                    // TODO Auto-generated catch block
                                    e.printStackTrace();
                                    }
                                    if(System.currentTimeMillis() - sendTime >=1000){
                                    return;
                                    }
                           }
                        } catch (IOException e1) {
                        // TODO Auto-generated catch block
                        e1.printStackTrace();
                        }
                        panel.remove(play);
                        panel.repaint();
                        panel.add(pause);
                        initStreaming = true;
            }
            public void start(){
                        try {
                        controlSocket = new Socket(InetAddress.getByName(serverIP),8080);
                        output = new ObjectOutputStream(controlSocket.getOutputStream());
                        input = new ObjectInputStream(controlSocket.getInputStream());
                        output.flush();
                        } catch (UnknownHostException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        }
            }
```

```
public void stop(){
reading = false;
HttpUtility.disconnect();
}
public void destroy(){
		if(initStreaming){
		Playing = false;
		try {
		output.writeObject("END");
		output.flush();
		} catch (IOException e1) {
		// TODO Auto-generated catch block
		e1.printStackTrace();
		}
		try {
		output.close();
		input.close();
		controlSocket.close();
		} catch (IOException e1) {
		// TODO Auto-generated catch block
		e1.printStackTrace();
		}
		}else{
				try {
				output.writeObject("END");
				output.flush();
				} catch (IOException e1) {
				// TODO Auto-generated catch block
				e1.printStackTrace();
				}
				try {
				output.close();
				input.close();
				controlSocket.close();
				} catch (IOException e1) {
				// TODO Auto-generated catch block
				e1.printStackTrace();
				}
		}
}
public void streaming(){
		Thread t = new Thread(new Runnable(){
				@Override
				public void run() {
						while(Playing){
						 receivePacket = new DatagramPacket(receiveData,0,receiveData.length);
							try {
							clientSocket.receive(receivePacket);
							} catch (IOException e) {
							// TODO Auto-generated catch block
							e.printStackTrace();
							}
							udpPortOpened = true;
							byte[] buff = receivePacket.getData();
							InputStream in = new ByteArrayInputStream(buff);
							BufferedImage image = null;
							try {
							image = ImageIO.read(in);
							} catch (IOException e) {
							// TODO Auto-generated catch block
							e.printStackTrace();
							}
							if(takePhoto){
							photo = image;
							takePhoto = false;
							}
```

```
                                        if(recLocally){
                                        recVideo(image);
                                        }
                                label.repaint();
                                if(image != null)
                                label.setIcon(new ImageIcon(image));
                                }
                        }

                });
        t.start();
        }
        public void GUI(){
                panel.add(startRecording);
                panel.add(grabFrame);
                panel.add(play);
                if(administrator){
                panel.add(left);
                panel.add(right);
                panel.add(up);
                panel.add(down);
                panel.add(stop);
                }
                panel.add(Server);
                panel.add(Client);
                panel.add(label);
                panel.add(human_presence);
                panel.add(humanpreLabel);
                panel.add(dateChooserStart);
                panel.add(dateChooserEnd);
                panel.add(endDate);
                panel.add(startDate);
                panel.add(hourStart);
                panel.add(minsStart);
                panel.add(hourEnd);
                panel.add(minsEnd);
                panel.add(scheduleJob);
                panel.repaint();
                readValues();
                readThread.start();
        }
        @Override
        public void run() {
        }
        public void readValues(){
                readThread = new Thread(new Runnable(){
                        @Override
                        public void run() {
                                try{
                                HttpUtilityTester.sendGetRequest();
                                }catch(IOException ex){
JOptionPane.showMessageDialog(null,"ILon server is down", "Connection Problem",JOptionPane.ERROR_MESSAGE);
                                reading = false;
                                }
                                if(reading){
                                panel.add(panelMeter);
                                panel.add(thermoPanel);
                                }
                                while(reading){
                                        try {
                                        HttpUtilityTester.sendGetRequest();
                                        } catch (IOException e1) {
                                        // TODO Auto-generated catch block
                                        e1.printStackTrace();
                                        }
                                        human_presence.setText(HttpUtilityTester.human_presence);
```

```
                                                    thermoData.setValue(Double.parseDouble(HttpUtilityTester.temperature));
                                                    luxData.setValue(Double.parseDouble(HttpUtilityTester.LUX));
                                                    try {
                                                    Thread.sleep(1000);
                                                    } catch (InterruptedException e) {
                                                    // TODO Auto-generated catch block
                                                    e.printStackTrace();
                                                    }
                                            }
                                    }
                            });
                }
                public void recVideo(BufferedImage buffImg){
            try {
                                    recorder.record(IplImage.createFrom(buffImg));
                                    } catch (Exception e) {
                                    // TODO Auto-generated catch block
                                    e.printStackTrace();
                                    }
                            }
                }
```

Η κλάση HttpUtilityTester

```
import java.io.IOException;
public class HttpUtilityTester {
   /**
    * This program uses the HttpUtility class to send a GET request to
    */
              static String LUX,temp,human_presence,temperature,requestURL;
   public static void sendGetRequest() throws IOException {
      // test sending GET request
       requestURL = "http://195.251.14.211/forms/hubbell.html";
         HttpUtility.sendGetRequest(requestURL);
         String[] response = HttpUtility.readMultipleLinesRespone();
         for (int i=0; i<response.length; i++) {
            if(response[i].contains("(LUX)")){
              LUX = response[i+2].replace("</td>", "");
            }
            if(response[i].contains("OC_OCCUPIED")){
              human_presence = "OC_OCCUPIED";
            }else if(response[i].contains("OC_UNOCCUPIED")){
              human_presence = "OC_UNOCCUPIED";
            }
            if(response[i].contains("&Theta;&Epsilon;&Rho;&Mu;&Omicron;&Kappa;&Rho;&Alpha;&Sigma;&Iota;&Alpha;
&Epsilon;&Rho;&Gamma;&Alpha;&Sigma;&Tau;&Eta;&Rho;&Iota;&Omicron;&Upsilon;<BR>")){
              temperature = response[i+2].replace("</td>", "");
            }
         }
   }
   public static void turnCamera(String url) {
      // sending GET request
      try {
         HttpUtility.sendGetRequest(url);
         HttpUtility.readMultipleLinesRespone();
      } catch (IOException ex) {
      }
    HttpUtility.disconnect();
   }
}
```

Η κλάση HttpUtility

```
import java.io.BufferedReader;
import java.io.IOException;
```

```
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
/**
 * This class encapsulates methods for requesting a server via HTTP GET/POST and
 * provides methods for parsing response from the server.
 *
 * @author www.codejava.net
 *
 */
public class HttpUtility {
    /**
     * Represents an HTTP connection
     */
    private static HttpURLConnection httpConn;
    /**
     * Makes an HTTP request using GET method to the specified URL.
     *
     * @param requestURL
     *            the URL of the remote server
     * @return An HttpURLConnection object
     * @throws IOException
     *             thrown if any I/O error occurred
     */
    public static HttpURLConnection sendGetRequest(String requestURL)
            throws IOException {
        URL url = new URL(requestURL);
        httpConn = (HttpURLConnection) url.openConnection();
        httpConn.setUseCaches(false);
        httpConn.setDoInput(true); // true if we want to read server's response
        httpConn.setDoOutput(false); // false indicates this is a GET request
        return httpConn;
    }
    /**
     * Returns an array of lines from the server's response. This method should
     * be used if the server returns multiple lines of String.
     *
     * @return an array of Strings of the server's response
     * @throws IOException
     *             thrown if any I/O error occurred
     */
    public static String[] readMultipleLinesRespone() throws IOException {
        InputStream inputStream = null;
        if (httpConn != null) {
            inputStream = httpConn.getInputStream();
        } else {
            throw new IOException("Connection is not established.");
        }
BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));
        List<String> response = new ArrayList<String>();
        String line = "";
        while ((line = reader.readLine()) != null) {
         response.add(line);
        }
        reader.close();
        return (String[]) response.toArray(new String[0]);
    }
    /**
     * Closes the connection if opened
     */
    public static void disconnect() {
        if (httpConn != null) {
            httpConn.disconnect(); }}}
```