Λογισμικό Διαχείρισης Αποθήκης Καταστημάτων Πώλησης Υπολογιστών και Περιφερειακών

Όνομα Σπουδαστή: Βαγγέλης Τζαβλάκης
ΑΜ:3775

Επιβλέπων καθηγητής : Αϊβαλής Κωνσταντίνος

Πίνακας Περιεχομένων

1) Σύνοψη

Με την παρούσα πτυχιακή εργασία, έγινε προσπάθεια ανάπτυξης και υλοποίησης μιας εφαρμογής η οποία ασχολείται με την αποθήκευση , την καταγραφή της τοποθεσίας, τα χαρακτηριστικά αλλά και την ποσότητα των προϊόντων μέσα σε μια αποθήκη. Καταχωρεί τους προμηθευτές καθώς και τις παραγγελίες του καταστήματος, χαρακτηριστικά απαραίτητα για την σωστή και εύρυθμη λειτουργίας μιας αποθήκης.

Έχοντας ως κύριους άξονες την αποτελεσματικότητα του προγράμματος αλλά και το λειτουργικό και ευχάριστο interface, το πρόγραμμα αυτό αναπτύχθηκε με την χρήση της java, παίρνοντας και αποθηκεύοντας τις πληροφορίες του σε έναν MySQL server,λόγο της δυνατότητας του να αποθηκεύει μεγάλου όγκους δεδομένων , πράγμα απαραίτητο σε ένα τέτοιο πρόγραμμα.

Με τον παραπάνω τρόπο μπόρεσε να δημιουργηθεί ένα πρόγραμμα τόσο χρηστικό, όσο και εύκολα αλλά και γρήγορα διαχειρήσιμο από τον οποιονδήποτε κατέχει την θέση του διαχειριστή της αποθήκης, χωρίς απαραίτητα να χρειάζεται ειδικές γνώσεις πάνω στου υπολογιστές.

2)Περίληψη

Η βασική ιδέα της πτυχιακής είναι η διευκόλυνση των χρηστών της , για την ικανοποίηση των επιμέρους αναγκών τους. Όσο για την καταχώρηση νέων παραγγελιών, νέων προϊόντων αλλά και την ομαλή εξυπηρέτηση τους ως προς την δουλειά τους. Υπάρχουν δύο ομάδες ανθρώπων οι οποίοι εργάζονται το πρόγραμμα. Α)Ο διαχειριστής(Administrator) του προγράμματος αλλά και της αποθήκης και Β) Ο αποθηκάριος(User) ο οποίος έχει μια πιο άμεση επαφή με το πρόγραμμα καθημερινά. Οι δυνατότητες η οποίες δίνονται στης δύο αυτές ομάδες χρηστών είναι οι παρακάτω:

2.1)Δυνατότητες αποθηκάριου (User)

1. Μπορεί να δει τα στοιχεία των προϊόντων όπως:
    - Το συγκεκριμένο ID με βάση την σειρά όπου προστέθηκε το αντίστοιχο προϊόν.
    - Το Barcode του εκάστοτε προϊόντος το οποίο είναι μοναδικό για κάθε προϊόν.
    - Το Όνομα το προϊόντος.
    - Την ποσότητα που υπάρχει στην αντίστοιχη αποθήκη.
    - Τα κιλά.
    - Της διαστάσεις.
    - Και την τοποθεσία που βρίσκεται στην αποθήκη.
2. Μπορεί να καταχωρήσει ένα καινούριο προϊόν.
3. Να διαγράψει ένα προϊόν.
4. Να φτιάξει μία αναφορά με τα προϊόντα όπου επιπλέον του δίνεται η δυνατότητα να τα εκτυπώσει.
5. Μπορεί να τροποποιήσει κάποια πληροφορία όσον αφορά τα προϊόντα.
6. Να αναζητήσει της πληροφορίες των προμηθευτών.
7. Όπως επίσης να τροποποιήσει και να εισάγει έναν καινούριο προμηθευτή.
8. Μπορεί να δει όλες της παραγγελίες που έχουν γίνει και να δει στοιχεία όπως:
    - Το ID που επίσης είναι η σειρά με την οποία έγινε η συγκεκριμένη παραγγελία.
    - Το όνομα του προϊόντος το οποίο παρήγγειλε κάποιος συνάδελφος του.
    - Την ποσότητα που παρήγγειλε .
    - Τον προμηθευτή από τον οποίο θα έρθει.
    - Τον υπάλληλο όπου έγινε η παραγγελία.
    - Το πότε έγινε η παραγγελία.
    - Πότε θα αρχίσει να ταξιδεύει η παραγγελία για να έρθει στην αποθήκη.
    - Πότε ενδέχεται να έρθει.
    - Και με ποια μεταφορική έχει γίνει συμφωνία για την μεταφορά των προϊόντων.
9. Μπορεί να επικυρώσει μία παραγγελία με το που έρθει το προϊόν.
10. Να παραγγείλει ένα προϊόν.
11. Όπως και να δει τα προϊόντα με την μικρότερη ποσότητα έτσι ώστε να προνοήσει να προβεί σε μια καινούρια παραγγελία.

2.2)Δυνατότητες διαχειριστή(Admin)

Οι δυνατότητες του διαχειριστή είναι ακριβώς οι ίδιες με τον αποθηκάριο όπως:

1 Μπορεί να δει τα στοιχεία των προϊόντων όπως:

- Το συγκεκριμένο ID με βάση την σειρά όπου προστέθηκε το αντίστοιχο προϊόν.
- Το Barcode του εκάστοτε προϊόντος το οποίο είναι μοναδικό για κάθε προϊόν.
- Το Όνομα το προϊόντος.
- Την ποσότητα που υπάρχει στην αντίστοιχη αποθήκη.
- Τα κιλά.
- Της διαστάσεις.
- Και την τοποθεσία που βρίσκεται στην αποθήκη.

2 Μπορεί να καταχωρήσει ένα καινούριο προϊόν.

3 Να διαγράψει ένα προϊόν.

4 Να φτιάξει μία αναφορά με τα προϊόντα όπου επιπλέον του δίνεται η δυνατότητα να τα εκτυπώσει.
5 Μπορεί να τροποποιήσει κάποια πληροφορία όσον αφορά τα προϊόντα.
6 Να αναζητήσει της πληροφορίες των προμηθευτών.
7 Όπως επίσης να τροποποιήσει και να εισάγει έναν καινούριο προμηθευτή.
8 Μπορεί να δει όλες της παραγγελίες που έχουν γίνει και να δει στοιχεία όπως:
   - Το ID που επίσης είναι η σειρά με την οποία έγινε η συγκεκριμένη παραγγελία.
   - Το όνομα του προϊόντος το οποίο παρήγγειλε κάποιος συνάδελφος του.
   - Την ποσότητα που παρήγγειλε .
   - Τον προμηθευτή από τον οποίο θα έρθει.
   - Τον υπάλληλο όπου έγινε η παραγγελία.
   - Το πότε έγινε η παραγγελία.
   - Πότε θα αρχίσει να ταξιδεύει η παραγγελία για να έρθει στην αποθήκη.
   - Πότε ενδέχεται να έρθει.
   - Και με ποια μεταφορική έχει γίνει συμφωνία για την μεταφορά των προϊόντων.
9 Μπορεί να επικυρώσει μία παραγγελία με το που έρθει το προϊόν.
10 Να παραγγείλει ένα προϊόν.
11 Όπως και να δει τα προϊόντα με την μικρότερη ποσότητα έτσι ώστε να προνοήσει να προβεί σε μια καινούρια παραγγελία.

Απλά με την διαφορά ότι έχει ένα δικό του ξεχωριστό παράθυρο, του οποίου δίνεται η δυνατότητα να διαχειρίζεται τους χρήστες. Η ενέργειες που μπορεί να προβεί ο διαχειριστής στους χρήστες(αποθηκάριους) είναι οι εξής:

1. Μπορεί να δει όλα τα στοιχεία των χρηστών όπως:
    - Το μοναδικό ID του χρήστη.
    - Το Username(Όνομα Χρήστη) του.
    - Το Password(Κωδικό) του.
    - Τι θέση έχει, δηλαδή η 2 κατηγορίες που ανέφερα πριν(User, Admin)
    - Το Όνομα του και το Επίθετο.
    - Το τηλέφωνο του.
    - Και το Email του.
2. Όπως επίσης να τροποποιήσει κάποια δεδομένα από τους χρήστες. Όπως να διαγράψει, να εισάγει έναν καινούριο χρήστη και να τροποποιήσει κάποια από τα παραπάνω στοιχεία που ανέφερα πιο πάνω.
3. Να δει όλες της ενέργειες που έχουν κάνει οι χρήστες στο πρόγραμμα αυτό. Όπου αναγράφονται στοιχεία όπως:
    a. Το πότε μπήκαν.
    b. Πότε άλλαξαν κάτι(Ακριβής ώρα και ημερομηνία).
    c. Που το άλλαξαν.
    d. Τι τιμή άλλαξαν.
    e. Και πότε βγήκαν.

3) Τεχνολογίες που χρησιμοποιήθηκαν.

Οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της συγκεκριμένης πτυχιακής είναι:

a) Η γλώσσα προγραμματισμού υψηλού επιπέδου JAVA.

b) Μία βάση δεδομένων που μας βοήθησε να καταχωρούμε και να αποθηκεύουμε τα δεδομένα μας. Αυτό έγινε με την το βοήθεια του MySQL Workbench

c) Ένα πρόγραμμα οπού θα μας έκανε compile την γλώσσα java και αυτό ήταν το πρόγραμμα NetBeans.

d) Μίκρο-εργαλεια που χρειαστήκαμε να κατεβάσουμε και να τα προσθέσουμε στο NetBeans, για να εμφανίσουμε πιο καλά τα στατιστικά μας ήταν το Jaspersoft.

3.1) Αντικειμενοστραφής Προγραμματισμός.

Στην πληροφορική αντικειμενοστραφής προγραμματισμό (object-oriented programming) ονομάζουμε ένα προγραμματιστικό υπόδειγμα το οποίο εμφανίστηκε στα τέλη της δεκαετίας του 1960 και καθιερώθηκε κατά τη δεκαετία του 1990, αντικαθιστώντας σε μεγάλο βαθμό το παραδοσιακό υπόδειγμα του δομημένου προγραμματισμού. Πρόκειται για μία μεθοδολογία ανάπτυξης προγραμμάτων, υποστηριζόμενη από κατάλληλες γλώσσες προγραμματισμού, όπου ο χειρισμός σχετιζόμενων δεδομένων και των διαδικασιών που επενεργούν σε αυτά γίνεται από κοινού, μέσω μίας δομής δεδομένων που τα περιβάλλει ως αυτόνομη οντότητα με ταυτότητα και δικά της χαρακτηριστικά. Αυτή η δομή δεδομένων καλείται αντικείμενο και αποτελεί πραγματικό στιγμιότυπο στη μνήμη ενός σύνθετου, και πιθανώς οριζόμενου από τον χρήστη, τύπου δεδομένων ονόματι κλάση. Η κλάση προδιαγράφει τόσο δεδομένα όσο και τις διαδικασίες οι οποίες επιδρούν επάνω τους· αυτή υπήρξε η πρωταρχική καινοτομία του ΑΠ.

Έτσι μπορεί να οριστεί μία προδιαγραφή δομής αποθήκευσης (π.χ. μία κλάση «τηλεόραση») η οποία να περιέχει τόσο ιδιότητες (π.χ. μία μεταβλητή «τρέχον κανάλι») όσο και πράξεις ή χειρισμούς επί αυτών των ιδιοτήτων (π.χ. μία διαδικασία «άνοιγμα της τηλεόρασης»). Στο εν λόγω παράδειγμα κάθε υλική τηλεόραση (κάθε αντικείμενο αποθηκευμένο πραγματικά στη μνήμη) αναπαρίσταται ως ξεχωριστό, «φυσικό» στιγμιότυπο αυτής της πρότυπης, ιδεατής κλάσης. Επομένως μόνο τα αντικείμενα καταλαμβάνουν χώρο στη μνήμη του υπολογιστή ενώ οι κλάσεις αποτελούν απλώς «καλούπια». Οι αιτίες που ώθησαν στην ανάπτυξη του ΑΠ ήταν οι ίδιες με αυτές που οδήγησαν στην ανάπτυξη του δομημένου προγραμματισμού (ευκολία συντήρησης, οργάνωσης, χειρισμού και επαναχρησιμοποίησης κώδικα μεγάλων και πολύπλοκων εφαρμογών), όμως τελικώς η αντικειμενοστρέφεια επικράτησε καθώς μπορούσε να ανταπεξέλθει σε προγράμματα πολύ μεγαλύτερου όγκου και πολυπλοκότητας. [1]

Οι περισσότερες αντικειμενοστραφείς έννοιες εμφανίστηκαν αρχικά στη γλώσσα προγραμματισμού Simula 67, η οποία ήταν προσανατολισμένη στην εκτέλεση προσομοιώσεων του πραγματικού κόσμου. Οι ιδέες της Simula 67 επηρέασαν κατά τη δεκαετία του '70 την ανάπτυξη της Smalltalk, της γλώσσας που εισήγαγε τον όρο αντικειμενοστραφής προγραμματισμός. Η Smalltalk αναπτύχθηκε από τον Άλαν Κέι της εταιρείας Xerox στο πλαίσιο μίας εργασίας με στόχο τη δημιουργία ενός χρήσιμου, αλλά και εύχρηστου, προσωπικού υπολογιστή. Όταν η τελική έκδοση της Smalltalk έγινε διαθέσιμη το 1980 η έρευνα για την αντικατάσταση του δομημένου προγραμματισμού με ένα πιο σύγχρονο υπόδειγμα ήταν ήδη εν εξελίξει. Στη γλώσσα αυτή όλοι οι τύποι δεδομένων ήταν κλάσεις (δεν υπήρχαν δηλαδή πια παραδοσιακές δομές δεδομένων παρά μόνο αντικείμενα).

Την ίδια περίπου εποχή, και επίσης με επιρροές από τη Simula, ολοκληρωνόταν η ανάπτυξη της C++ ως μίας ισχυρής επέκτασης της δημοφιλούς γλώσσας προγραμματισμού C στην οποία είχαν "μεταμοσχευθεί" αντικειμενοστραφή χαρακτηριστικά. Η επιρροή της C++ καθ' όλη της δεκαετία του '80 ήταν καταλυτική με αποτέλεσμα τη σταδιακή κυκλοφορία αντικειμενοστραφών εκδόσεων πολλών γνωστών διαδικαστικών γλωσσών προγραμματισμού. Κατά το πρώτο ήμισυ της δεκαετίας του '90 η βαθμιαία καθιέρωση στους μικροϋπολογιστές των γραφικών διασυνδέσεων χρήστη (GUI), για την ανάπτυξη των οποίων ο ΑΠ φαινόταν ιδιαιτέρως κατάλληλος, και η επίδραση της C++ οδήγησαν στην επικράτηση της αντικειμενοστραφείς ως βασικού προγραμματιστικού υποδείγματος.

Το 1995 η εμφάνιση της Java, μίας ιδιαίτερα επιτυχημένης, πλήρως αντικειμενοστραφούς γλώσσας που έμοιαζε συντακτικώς με τη C/C++ και προσέφερε πρωτοποριακές για την εποχή δυνατότητες, έδωσε νέα ώθηση στον ΑΠ. Παράλληλα εμφανίστηκαν ποικίλες άτυπες βελτιώσεις στο βασικό προγραμματιστικό υπόδειγμα, όπως οι αντικειμενοστραφείς γλώσσες μοντελοποίησης λογισμικού, τα σχεδιαστικά πρότυπα κλπ. Το 2001 η Microsoft εστίασε την προσοχή της στην πλατφόρμα .NET, μία ανταγωνιστική της Java πλατφόρμα ανάπτυξης και εκτέλεσης λογισμικού η οποία ήταν εξολοκλήρου προσανατολισμένη στην αντικειμενοστρέφεια. Αυτή τη στιγμή η C++ και η Java είναι οι πιο δημοφιλείς αντικειμενοστραφείς γλώσσες. Η Java συγκεκριμένα έχει σχεδιαστεί ειδικά για χρήση σε κατανεμημένες εφαρμογές μέσω δικτύου και διαδικτύου. [1]

3.2)Γλώσσα προγραμματισμού JAVA

Η Java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που σχεδιάστηκε από την εταιρεία πληροφορικής Sun Microsystems.

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).

Άλλα ιδιαίτερα χαρακτηριστικά της Java είναι ο πολυνηματισμός, η ιδιαίτερη τεχνολογία δικτύου καθώς και το χαρακτηριστικό της ότι μπορεί να χρησιμοποιηθεί σε οποιουδήποτε μεγέθους εφαρμογή. Ένα άλλο δυνατό σημείο της Java είναι οι δυνατότητες που παρέχει για την δημιουργία ιστοσελίδων και μάλιστα δυναμικών, οι οποίες χρησιμοποιώντας το πρωτόκολλο μεταφοράς HTTP και την γλώσσα υπερκειμένου HTML προσφέρουν στους χρήστες του διαδικτύου πλείστες δυνατότητες. [1]

3.3)NetBeans

Το Java SE 8, το Java SE Embedded 8 και το Java ME Embedded 8.Το NetBeans είναι μια πλατφόρμα ανάπτυξης λογισμικού γραμμένη σε Java. Η πλατφόρμα NetBeans επιτρέπει την ανάπτυξη εφαρμογών από ένα σύνολο αρθρωτών στοιχείων λογισμικού που ονομάζονται ενότητες. Οι εφαρμογές που βασίζονται στην πλατφόρμα NetBeans, συμπεριλαμβανομένου του ολοκληρωμένου περιβάλλοντος ανάπτυξης NetBeans (IDE), μπορούν να επεκταθούν από τρίτους προγραμματιστές.

Το NetBeans IDE προορίζεται κυρίως για ανάπτυξη σε Java, αλλά υποστηρίζει και άλλες γλώσσες, συγκεκριμένα PHP, C / C ++ και HTML5. Επίσης είναι ένα πλαίσιο για την απλούστευση της ανάπτυξης εφαρμογών επιφάνειας εργασίας Java Swing. Η δέσμη ενεργειών NetBeans IDE για το Java SE περιέχει αυτό που απαιτείται για να ξεκινήσει την ανάπτυξη εφαρμογών και εφαρμογών που βασίζονται στην πλατφόρμα NetBeans. Επιπλέον και πιο σημαντικό είναι ότι δεν απαιτείται επιπλέον SDK. Η πλατφόρμα προσφέρει επαναχρησιμοποιήσιμες υπηρεσίες , επιτρέποντας στους προγραμματιστές να επικεντρωθούν στη λογική που αφορά στην εφαρμογή τους. Μεταξύ των χαρακτηριστικών της πλατφόρμας είναι:

a) Διαχείριση διεπαφής χρήστη (π.χ. μενού και γραμμές εργαλείων).
b) Διαχείριση ρυθμίσεων χρηστών.
c) Διαχείριση αποθήκευσης (αποθήκευση και φόρτωση οποιουδήποτε είδους δεδομένων).
d) Διαχείριση παραθύρων.
e) Οδηγός πλαισίου (υποστηρίζει διαλόγους βήμα προς βήμα).
f) NetBeans Visual Library.
g) Ολοκληρωμένα εργαλεία ανάπτυξης. [2]

3.4) Γλώσσα υπολογιστών στις βάσεις δεδομένων (SQL)

Η είναι μία γλώσσα υπολογιστών στις βάσεις δεδομένων, που σχεδιάστηκε για τη διαχείριση δεδομένων, σε ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (Relational Database Management System, RDBMS) και η οποία, αρχικά, βασίστηκε στη σχεσιακή άλγεβρα. Η γλώσσα περιλαμβάνει δυνατότητες ανάκτησης και ενημέρωσης δεδομένων, δημιουργίας και τροποποίησης σχημάτων και σχεσιακών πινάκων, αλλά και ελέγχου πρόσβασης στα δεδομένα. Η SQL ήταν μία από τις πρώτες γλώσσες για το σχεσιακό μοντέλο του Edgar F. Codd, στο σημαντικό άρθρο του το 1970, και έγινε η πιο ευρέως χρησιμοποιούμενη γλώσσα για τις σχεσιακές βάσεις δεδομένων.

Η γλώσσα SQL είναι ένα εργαλείο για την οργάνωση, διαχείριση και ανάκτηση δεδομένων που είναι αποθηκευμένα σε έναν υπολογιστή. Γεννήθηκε μαζί με την τεχνολογία των σχεσιακών βάσεων δεδομένων και έγινε στερεότυπη για την δια δραστικότητα αυτών. Χρησιμοποιείται κατά κόρων σε σύγχρονες εφαρμογές για να θέτει ερωτήματα σε τέτοιες βάσεις. Επιτρέπει σε άλλες συμβατές με αυτή γλώσσες προγραμματισμού, να εκτελούν ερωτήματα σε ένα μεγάλο εύρος δεδομένων και λειτουργεί αντιμετωπίζοντας τα δεδομένα με μια απλή και τυποποιημένη διάταξη δυσδιάστατων πινάκων δεδομένων, με γραμμές και στήλες. Το απλό αυτό μοντέλο όμως, που επιτρέπει τη σύνταξη κομψών ερωτημάτων σε μια βάση δεδομένων έχει και το τίμημα του. Η πολυπλοκότητα των σχέσεων μεταξύ των δεδομένων του πραγματικού κόσμου δεν μπορεί να αποτυπωθεί εύκολα σε γραμμές και στήλες, αναγκάζοντας έτσι τα δεδομένα να κατακερματιστούν σε πολλαπλούς πίνακες, για να εκτελεστούν έστω και απλές εργασίες. Το γεγονός αυτό γεννάει δυο προβλήματα α) τα ερωτήματα μπορεί να γίνουν ιδιαίτερα πολύπλοκα προκειμένου να συγκεντρώσουν δεδομένα από πολλούς πίνακες και β) το κόστος επεξεργασίας για την πρόσβαση των δεδομένων με τέτοια ερωτήματα σε σχεσιακές βάσεις, μπορεί να αποβεί κολοσσιαίο [3]

3.5) MySQL Workbench.

Το MySQL Workbench είναι ένα εργαλείο σχεδιασμού οπτικής βάσης δεδομένων που ενσωματώνει την ανάπτυξη SQL, τη διαχείριση, τον σχεδιασμό βάσεων δεδομένων, τη δημιουργία και τη συντήρηση σε ένα ενιαίο ολοκληρωμένο περιβάλλον ανάπτυξης για το σύστημα βάσεων δεδομένων MySQL. Είναι ο διάδοχος του DBDesigner 4 από το fabFORCE.net και αντικαθιστά το προηγούμενο πακέτο λογισμικού MySQL GUI Tools Bundle.

Αυτό το ‘εργαλείο’ μας βοήθησε για να αποθηκεύουμε δεδομένα που είναι χρήσιμα για εμάς. Όπως την καταγραφή των δεδομένων των προϊόντων μας , της παραγγελίες μας, τα δεδομένα των χρηστών ακόμα και της κινήσεις των χρηστών. [4]

3.6) Jaspersoft Reporting tool.

Το JasperReports είναι ένα εργαλείο αναφοράς, σε Java ανοιχτού κώδικα που μπορεί να εκτυπώσει σε διάφορες πηγές, όπως: οθόνη, εκτυπωτής, σε μορφή PDF, HTML, Microsoft Excel, RTF, ODT .
Μπορεί να χρησιμοποιηθεί σε Java εφαρμογές, συμπεριλαμβανομένου του Java EE ή εφαρμογών ιστού, για τη δημιουργία δυναμικού περιεχομένου. Όλα τα jasperReports επεξεργάζονται σε μορφή XML. [5]

4)Περιγραφή της Βάσης Δεδομένων.

Για να αποθηκεύσουμε όλα τα δεδομένα που θα χρειαζόμασταν, ώστε να κρατάμε σημαντικές πληροφορίες της κάθε αποθήκης και να της επεξεργαστούμε θα χρειαστούμε μία βάση δεδομένων . Η συγκεκριμένη βάση που δημιουργήσαμε έχει το όνομα ‘Company’. Με στοιχεία για να μπούμε σε αυτήν είναι:

- MySQL Ip: 127.0.0.1:3306
- Username: root
- Password: zoot

Μέσα στην βάση πρέπει όλα μας τα αρχεία να είναι ομαδοποιημένα έτσι ώστε να πετύχουμε την σωστή οργάνωση των στοιχείων μας, και να είναι πιο εύκολη η χρήση τους. Έτσι οι ομάδες που έχουμε δημιουργήσει είναι οι εξής:

Users, όπου εκεί αποθηκεύουμε όλες της πληροφορίες που θέλουμε για αυτούς που έχουν πρόσβαση στο πρόγραμμα μας. Δηλαδή τον αποθηκάριο και τον διαχειριστή. Γι’ αυτό θα χρειαστούμε δύο ιδιότητες χρηστών User και Admin.

Users:

Name	***Datatype***	***Ιδιότητες***
Id	***INT(11)***	***Public Key, Not Null, Auto Increment***
Username	***VARCHAR(45)***	***Not Null***
Password	***VARCHAR(45)***	***Not Null***
Status	***VARCHAR(45)***	***Not Null***
First_Last_Name	***VARCHAR(45)***	***Not Null***
ContactPhone	***VARCHAR(45)***	***Not Null***
Email	***VARCHAR(45)***	***Not Null***

- *Id*: Είναι ένας μοναδικός αριθμός για κάθε χρήση όπου δηλώνει και την σειρά που εισήχθη ο συγκεκριμένος χρήστης. Όπου είναι ξεχωριστό για κάθε αποθηκάριο.
- *Username*: Το username του υπαλλήλου που πρέπει να βάλει για να μπει μέσα στο πρόγραμμα.
- *Password*: Τον κωδικό που πρέπει να βάλει σε συνδυασμό με το κατάλληλο username για να μπει επιτυχώς στο πρόγραμμα.
- *Status*: Την ιδιότητα του συγκεκριμένου υπαλλήλου στην αποθήκη (User/Admin).
- *First_Last_Name: Το ονοματεπώνυμο του υπαλλήλου.*
- *ContactPhone*: Το τηλέφωνο του.
- *Email*: Το email του.

Products. Αποθηκεύουμε όλες της πληροφορίες για τα προϊόντα που έχουμε στην αποθήκης μας.

Products:

Name	***Datatype***	***Ιδιότητες***
Id	***INT(11)***	***Public Key, Not Null, Auto Increment***
Barcode	***VARCHAR(45)***	***Not Null, Unique index***
Name	***VARCHAR(45)***	***Not Null***
Quantity	***INT(11)***	***Not Null***
Weight	***VARCHAR(45)***	***Not Null***
Dimentions	***VARCHAR(45)***	***Not Null***
Employee	***VARCHAR(45)***	***Not Null***
Location	***VARCHAR(45)***	***Not Null***

- *Id*: Είναι ένας μοναδικός αριθμός για κάθε προϊόν όπου δηλώνει και την σειρά που εισήχθη. Όπου είναι ξεχωριστό για κάθε αποθηκάριο.
- *Barcode*: Άλλος ένας μοναδικός αριθμός που βγαίνει από την εταιρεία που έχει προέλθει το συγκεκριμένο προϊόν.
- *Name*: Το όνομα του προϊόντος.
- *Quantity*: Η ποσότητα του συγκεκριμένου προϊόντος που υπάρχει μέσα στην αποθήκη.
- *Weight*: Το βάρος του προϊόντος.
- *Dimentions*: Οι διαστάσεις του.
- *Employee*: Εδώ αναγράφεται ο υπάλληλος που το καταχώρησε μέσα στην βάση δεδομένων.
- *Location*: Η τοποθεσία του προϊόντος μέσα στην αποθήκη.\

Suppliers. Κάθε εταιρεία έχει μπλοκ που καταχωρούνται όλοι οι προμηθευτές για την άμεση παραγγελία των προϊόντων. Έτσι και μία αποθήκη χρειάζεται να έχει όλα της τα αποθέματα γεμάτα για την σωστή λειτουργία της.

Suppliers:

Name	***Datatype***	***Ιδιότητες***
Id	***INT(11)***	***Public Key, Not Null, Auto Incremental***
SupplierName	***VARCHAR(450)***	***Not Null***
Address	***VARCHAR(450)***	***Not Null***
City	***VARCHAR(450)***	***Not Null***
PostalCode	***VARCHAR(450)***	***Not Null***
Country	***VARCHAR(450)***	***Not Null***
Phone	***VARCHAR(450)***	***Not Null***
Email	***VARCHAR(450)***	***Not Null***
Employee	***VARCHAR(450)***	***Not Null***

- *Id: Μοναδικός αριθμός για κάθε προμηθευτή*, δηλώνει και την σειρά που εισήχθη.
- *SupplierName: Το όνομα του προμηθευτή.*
- *Address: Η διεύθυνση του προμηθευτή.*
- *City: Η πόλη από όπου βρίσκεται η εταιρεία.*
- *PostalCode: Ο Ταχυδρομικός κώδικας της περιοχής που βρίσκεται το αντίστοιχο κατάστημα.*
- *Phone: Το τηλέφωνο επικοινωνίας για την συνεννόηση της αποθήκης με τον προμηθευτή.*
- *Email: Το email της εταιρίας που είναι ένας διαφορετικός τρόπος συνεννόησης.*
- *Employee: Ο υπάλληλος που έβαλε αυτή την επαφή.*\

Orders. *Κάθε εταιρεία πρέπει να αποθηκεύει της αποδείξεις για της αγορές που έχει πραγματοποιήσει.*

Orders:

Name	***Datatype***	***Ιδιότητες***
OrderId	***INT(11)***	***Public Key, Not Null, Auto Incremental***
ProductName	***VARCHAR(405)***	***Not Null***
Quantity	***INT(11)***	***Not Null***
Supplier	***VARCHAR(405)***	***Not Null***
Employee	***VARCHAR(405)***	***Not Null***
OrderDate	***VARCHAR(405)***	***Not Null***
DateShipping	***VARCHAR(405)***	***Not Null***
DatePromised	***VARCHAR(405)***	***Not Null***
ShippingMethod	***VARCHAR(405)***	***Not Null***
Status	***VARCHAR(405)***	***Not Null***

- *OrderID: Μοναδικός αριθμός για κάθε παραγγελία που έχει γίνει,* δηλώνει και την σειρά που έγινε η κάθε μια.
- *ProductName: Το όνομα του προϊόντος που παράγγειλαν.*
- *Quantity: Η ποσότητα που παράγγειλαν.*
- *Supplier: Από ποιόν προμηθευτή αγόρασαν το προϊόν.*
- *Employee: Ποιος έκανε την παραγγελία.*
- *OrderDate: Η ημερομηνία που έγινε η συγκεκριμένη παραγγελία.*
- *DateShipping: Πότε θα φύγει από τον προμηθευτή.*
- *DatePromised: Ενδεχόμενη άφιξη του προϊόντος στην αποθήκη (αυτό συνήθως κανονίζεται μέσω τηλεφώνου με τον αγοραστή).*
- *ShippingMethod: Ποια θα είναι η μεταφορική η οποία θα είναι υπεύθυνη για την μεταφορά του προϊόντος.*
- *Status: Και η κατάσταση που βρίσκεται η παραγγελία. Έχουμε δύο τιμές που μπαίνουν εδώ. Μια είναι το Confirmed που μπαίνει όταν η παραγγελία έχει φτάσει στην αποθήκη επιτυχώς και η άλλη είναι το Pending.. που σημαίνει ότι η παραγγελία έχει σταλθεί , έχει αποδεχθεί από τον προμηθευτή και περιμένουν να έρθει το προϊόν στην αποθήκη. Μόλις φτάσει το προϊόν στην αποθήκη ο υπάλληλος πρέπει να επεξεργαστεί της παραγγελίες και να πατήσει Confirmed.*

Logfile: *Για να υπάρχει μία ομαλή εξυπηρέτηση των πελατών πρέπει να υπάρχει και ο αντίστοιχος επιβλέπων. Γι' αυτό η αποθήκη θα πρέπει να προσλάβει έναν διαχειριστή ο οποίος θα έχει ακριβώς της ίδιες ικανότητες με τους αποθηκάριους , η μόνη τους διαφορά θα είναι ότι η διαχειριστές θα ελέγχουν τους αποθηκάριους για την δουλεία τους. Κάθε διαχειριστής λοιπόν θα πρέπει να ελέγχει κάθε κίνηση του χρήστη, έτσι ώστε εάν πράξει λανθασμένα, να απευθυνθεί στον αποθηκάριο και να του κάνει την κατάλληλη επίπληξη για να τον διόρθωση.*

Logfile:

Name	***Datatype***	***Ιδιότητες***
Id	***INT(11)***	***Public Key, Not Null, Auto Incremental***
Date_Time	***VARCHAR(450)***	***-/-***
ChangedBy	***VARCHAR(450)***	***-/-***
Action	***VARCHAR(450)***	***-/-***
Value	***VARCHAR(450)***	***-/-***
InWhat	***VARCHAR(450)***	***-/-***

- *Id: Μοναδικός αριθμός για κάθε πράξη που έχει γίνει*, δηλώνει και την σειρά που έγινε η κάθε μια.
- *Date_Time: Ακριβής ώρα και ημερομηνία που έγινε.*
- *ChangedBy: Από ποιόν έγινε.*
- *Action: Τι ακριβώς έκανε.*
- *Value: Τι τιμή άλλαξε.*
- *InWhat: Σε τι το άλλαξε.*

5)Ανάλυση και επεξήγηση του προγράμματος.

5.0.1)Ανάλυση προγράμματος με χρήση UML

Για την καλύτερη κατανόηση το προγράμματος χρησιμοποιήθηκε ένα μοντέλο UML ,έτσι ώστε να γίνει μία περιγραφή των περιεχομένων του προγράμματος.

Ένα μοντέλο λοιπόν είναι μία αφηρημένη περιγραφή ενός φυσικού συστήματος και αποτελεί ένα σχέδιο για την κατασκευή ενός συστήματος. Γιαυτό χρησιμοποιήθηκε η UML, όπου είναι μια οπτική αντικειμενοστραφής γλώσσα μοντελοποίησης που χρησιμοποιείται για:

a) Απεικόνιση (visualization)

b) Προδιαγραφή (specification)

c) Τεκμηρίωση (documentation)

d) Κατασκευή (construction)

των δομικών συστατικών ενός συστήματος(λογισμικού η όχι).

Χαρακτηριστικά της UML

1. *Είναι ιδιαίτερα εκφραστική*
2. *Σημασιολογικά, είναι εκτενής. Υποστηρίζει τη σημασιολογία τύπων και μοντέλων για όλα τα μοντέλα ενός συστήματος.*
3. *Σε επίπεδο βασικών αρχήν , είναι μικρή και απλή.*
4. *Είναι επεκτάσιμη και υπάρχει η δυνατότητα εμπλουτισμού του μετά-μοντέλου με κλάσεις, ιδιότητες και σημασιολογία.*
5. *Είναι επακριβώς ορισμένη με βάση τα δομικά συστατικά ενός αντικειμενοστραφούς συστήματος.*
6. *Προήλθε από την ενοποίηση των συμβολισμών που χρησιμοποιούσαν οι μμεθοδολογίες Booch, OMT, OOSE κ.λπ.*
7. *Ενσωματώνει τις ιδέες "καλής πρακτικής" από τη βιομηχανία λογισμικού.*
    - *Είναι σήμερα βιομηχανικό πρότυπο και αναπτύχθηκε στο πλαίσιο του οργανισμού OMG.*
8. *Υλοποιεί την ανάγκη της βιομηχανίας λογισμικού για μια ενιαία γλώσσα μμοντελοποίησης.*
9. *Αντιμετώπιση σημερινών και βραχυπρόθεσμων προβλημάτων στην ανάπτυξη λογισμικού:*
    a. *Κλίμακα*
    b. *Γλώσσες: Java, C++, Smalltalk, Ada, Visual Basic*
    c. *Πολυεπεξεργασία και παραλληλία*
    d. *Πρότυπα Λογισμικού (Patterns)*
    e. *Ψηφίδες λογισμικού (Componentware)*
    f. *Μοντελοποίησης επιχειρησιακής πρακτικής*

Δομικά στοιχεία της UML

1) ***Κλάση (class).*** Ένα σύνολο αντικειμένων με κοινή δομή και συμπεριφορά.
2) ***Ενεργή κλάση (active class).*** Μια κλάση που περιγράφει μια διεργασία ή ένα νήμα εκτέλεσης και αλληλοεπιδρά με άλλες.
3) ***Περίπτωση χρήσης (use case).*** Μια λειτουργία που επιτελεί ένα σύστημα και είναι διαθέσιμη στο χρήστη. Είναι συμπεριφορά του συστήματος που συνεπάγεται τη συνεργασία ενός συνόλου αντικειμένων.
4) **Συνιστώσα (component).** Ένα φυσικό και επαναχρησιμοποιήσιμο τμήμα ενός συστήματος, με λογική και φυσική υπόσταση που συνήθως υλοποιεί κάποιες διεπιφάνειες (interfaces).
5) ***Συνεργασία (collaboration).*** Η περιγραφή μιας διάδρασης μμεταξύ ενός συνόλου αντικειμένων.
6) ***Κόμβος (node).*** Ένας υπολογιστικός πόρος που έχει κάποια μνήμη και υπολογιστική ικανότητα, οπότε εκεί αποθηκεύεται ή/και εκτελείται το λογισμικό.
7) ***Ενεργών (actor).*** Εξωτερική του συστήματος οντότητα που χρησιμοποιεί τη λειτουργικότητα και τις διεπιφάνειές του.

Έτσι για την δημιουργία των UML χρησιμοποιήσαμε το Visual Paradigm.

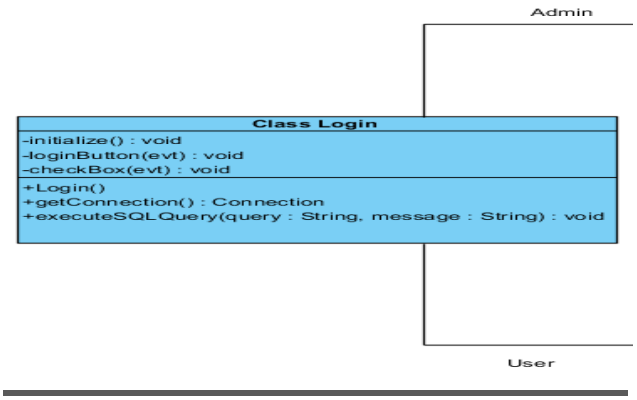
Σχήμα UML του προγράμματος.

Εικόνα 5.0.1-1: UML Προγράμματος

Παραπάνω δίνεται το σχήμα UML του προγράμματος που δείχνει τα δομικά στοιχεία των κλάσεων , όπου αναλύει όλες της κλάσεις που χρησιμοποιήθηκαν για να 'χτιστεί' το πρόγραμμα. Στην συνέχεια ακολουθεί μια περιγραφή όλων των κλάσεων και πώς αυτές είναι δομημένες έτσι ώστε να κάνουν μια λειτουργία.

Ανάλυση UML

1)Login



Εικόνα 5.0.1-2: UML Class Login

Αυτό το κουμπί είναι το πρώτο παράθυρο απ' όλα που ανοίγει. Εδώ γίνεται ο έλεγχος στον οποίο δίνει την δυνατότητα στων χρήστη ,πληκτρολογώντας τα κατάλληλα στοιχεία όπου έχουν δωθεί από τον διαχειριστή να εισέλθει στο πρόγραμμα. Ανάλογα με την ιδιότητα του χρήστη μεταφέρεται σε ένα ξεχωριστό Panel.Χωρίς αυτά τα στοιχεία καθιστάτε δύσκολο έως και αδύνατο στον χρήστη να μπει στο πρόγραμμα αν δεν πληκτρολογήσει σωστά τα δεδομένα.

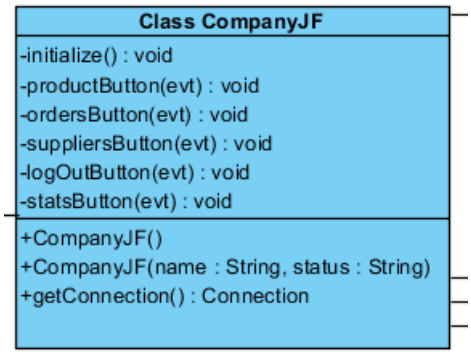
Σε αυτή την κλάση ως private μέθοδοι είναι:

- **Initialize() :** Σε αυτό το σημείο αλλάζω/ορίζω το εικονίδιο που θα υπάρχει όταν ανοίξει το πρόγραμμα.
- **loginButton(evt) :** Εδώ προσαρμόζω της ενέργειες που θα κάνει αυτό το κουμπί . Στην περιπτωσή μας ελέγχει εάν τα δεδομένα που εισήγαγε είναι καταχωρημένα στην βάση δεδομένων, εάν ναι ελέγχει επίσης εάν ο συγκεκριμένος χρήστης είναι User η Admin.
- **checkButton(evt) :** Κάθε φορά που πληκτρολογεί κάποιος των κωδικό του για να εισέλθει μέσα στο πρόγραμμα, για λόγους ασφάλειας , ο κάθε αριθμός που πληκτρολογεί να εμφανίζεται με '*'. Έτσι έχει προστεθεί ένα κουμπί όπου ο χρήστης με αυτό έχει την δυνατότητα να δει τι πληκτρολογεί.

Και ως public :

- **Login():** Εδώ βάζουμε της μεθόδους όπου θέλουμε να τρέξουμε κάθε φορά που πατάμε run.
- **getConnection()**: Με αυτή την μέθοδο 'συνδεόμαστε' στην βάση δεδομένων μας με το κατάλληλο username και password.
- **executeSQLQuery(String : query, String : message) :** Κάθε φορά που κάνει κάποιος ενέργεια με την βάση δεδομένων μας δίνεται δυνατότητα με αυτή την μέθοδο να μας πετάγεται ένα μήνυμα όπου μας πληροφορεί αν έγινε η ενέργεια σωστά ή αν δεν έγινε καθόλου.

2) Αρχικό Panel. Για χρήστη(User)



Εικόνα 5.0.1-3: UML Class CompanyJF

Με αυτό το κουμπί φορτώνεται ένα άλλο Panel που έχει διάφορες επιλογές να κάνει ο χρήστης. Όπως να κάνει καινούριες παραγγελίες , να δει της παραγγελίες κλπ.

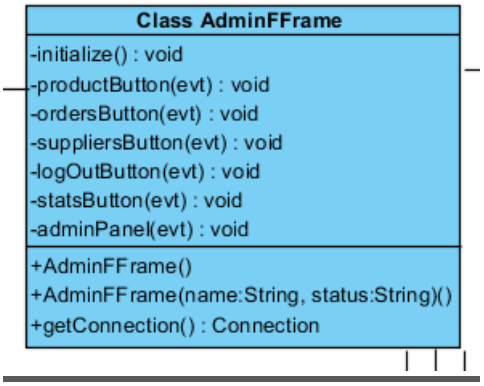
Σε αυτή την κλάση ως private μέθοδοι είναι:

- **Initialize():** Σε αυτό το σημείο αλλάζω/ορίζω το εικονίδιο που θα υπάρχει όταν ανοίξει το πρόγραμμα.
- **productButton(evt)**: Το κουμπί για να κατευθυνθεί στην κλάση Products.
- **ordersButton(evt):** Το κουμπί για να κατευθυνθεί στην κλάση Orders.
- **suppliersButton(evt):** Το κουμπί για να κατευθυνθεί στην κλάση Suppliers.
- **logOutButton(evt):** Το κουμπί για να κατευθυνθεί στην κλάση loggoutAnswer και να αποσυνδεθεί από το πρόγραμμα.
- **statsButton(evt):** Το κουμπί για να κατευθυνθεί στην κλάση stats.

Και ως public :

- **CompanyJF():**Εδώ βάζουμε της μεθόδους όπου θέλουμε να τρέξουμε κάθε φορά που πατάμε run. Αυτή η κλάση χρησιμοποιήθηκε μόνο στην δημιουργία του προγράμματος καθώς ήταν για δοκιμαστικό σκοπό και δεν χρησιμεύει τίποτα άλλο στο πρόγραμμα μας.
- **CompanyJF(name: String, status, String):** Αυτή η κλάση είναι που καλείται καθ' όλη την διάρκεια του προγράμματος. Καθώς παίρνει δύο ορίσματα, το όνομα του εργαζόμενου και την θέση του. Αυτές οι δύο μεταβλητές με βοήθησαν στο να εμφανίζω κάθε φορά ποιος είναι ο χρήστης που έχει συνδεθεί και να γνωρίζω ανά πάσα στιγμή της ενέργειες που κάνει στο πρόγραμμα .
- **getConnection():** Αυτή η κλάση δεν χρειάζεται πουθενά στην συγκεκριμένη κλάση , ισχύει το ίδιο πράγμα όπως και με την CompanyJF().

3) Αρχικό Panel. Για διαχειριστή(Admin)



Εικόνα 5.0.1-4: UML Class AdminFFrame

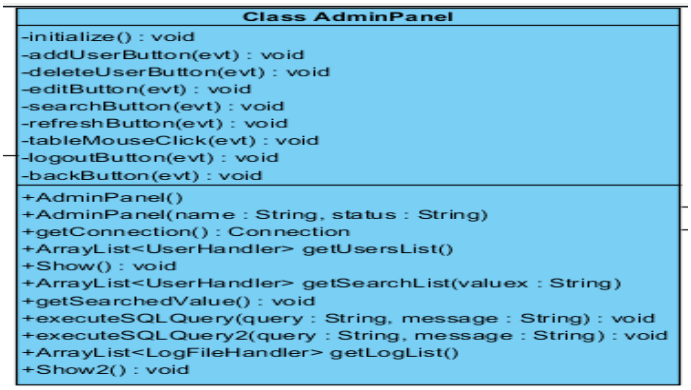
Σε αυτή την κλάση έχει άδεια να μπει μόνο οι χρήστες όπου έχουν ως Status το Admin. Έχει ακριβώς της ίδιες μεθόδους με ότι έχει ο χρήστης απλά με την διαφορά του κουμπιού για την διαχείριση των χρηστών.

Σε αυτή την κλάση ως private μέθοδοι είναι:

- **Initialize():** Σε αυτό το σημείο αλλάζω/ορίζω το εικονίδιο που θα υπάρχει όταν ανοίξει το πρόγραμμα.
- **productButton(evt**): Το κουμπί για να κατευθυνθεί στην κλάση Products.
- **ordersButton(evt):** Το κουμπί για να κατευθυνθεί στην κλάση Orders.
- **suppliersButton(evt):** Το κουμπί για να κατευθυνθεί στην κλάση Suppliers.
- **logOutButton(evt):** Το κουμπί για να κατευθυνθεί στην κλάση loggoutAnswer και να αποσυνδεθεί από το πρόγραμμα.
- **statsButton(evt):** Το κουμπί για να κατευθυνθεί στην κλάση stats.
- **adminPanel(evt):** Το κουμπί για να κατευθυνθεί στην κλάση AdminPanel.

Και ως public :

- **AdminFFrame():**Εδώ βάζουμε της μεθόδους όπου θέλουμε να τρέξουμε κάθε φορά που πατάμε run. Αυτή η κλάση χρησιμοποιήθηκε μόνο στην δημιουργία του προγράμματος καθώς ήταν για δοκιμαστικό σκοπό και δεν χρησιμεύει τίποτα άλλο στο πρόγραμμα μας.
- **AdminFFrame(name: String, status, String):** Αυτή η κλάση είναι που καλείται καθ' όλη την διάρκεια του προγράμματος. Καθώς παίρνει δύο ορίσματα, το όνομα του εργαζόμενου και την θέση του. Αυτές οι δύο μεταβλητές με βοήθησαν στο να εμφανίζω κάθε φορά ποιος είναι ο χρήστης που έχει συνδεθεί και να γνωρίζω ανά πάσα στιγμή της ενέργειες που κάνει στο πρόγραμμα .
- **getConnection():** Αυτή η κλάση δεν χρειάζεται πουθενά στην συγκεκριμένη κλάση , ισχύει το ίδιο πράγμα όπως και με την CompanyJF().



Εικόνα 5.0.1-5: UML Class AdminPanel

Σε αυτή την κλάση έχουμε της παραπάνω ιδιότητες που έχει ο διαχειριστής. Εδώ σε κάθε ενέργεια που κάνει ο χρήστης καταχωρείτε επίσης στην βάση δεδομένων με όνομα logfile την ενέργεια που έκανε.

Σε αυτή την κλάση ως private μέθοδοι είναι:

- **Initialize():** Σε αυτό το σημείο αλλάζω/ορίζω το εικονίδιο που θα υπάρχει όταν ανοίξει το πρόγραμμα.
- **addUserButton(evt):** Το κουμπί για να κατευθυνθεί στην κλάση AddUser. Όπου η κλάση είναι φτιαγμένη ακριβώς όπως την login απλα με διαφορετικό query.
- **deleteUserButton(evt):** Εδώ με την βοήθεια της sql διαγράφω μία τιμή από την βάση δεδομένων μου .
- **editButton(evt):** Ανανεώνω στοιχεία από την βάση δεδομένων μου.
- **searchButton(evt):** Ψάχνω συγκεκριμένα στοιχεία και ανανεώνω τον πίνακα.
- **refreshButton(evt):** Απλά ανανεώνω τον πίνακα με τα στοιχεία που βρίσκονται στην βάση δεδομένων.
- **tableMouseClicked(evt):** Εμφανίζω στοιχεία από τον πίνακα και τα τοποθετώ στα κατάλληλα jTextField ώστε να μπορώ να τα επεξεργαστώ πιο γρήγορα.
- **backButton(evt):** Παω στο Panel που βρισκόμουν πρίν.

Και ως public :

- **AdminPanel(name:String,Status:String):** Ισχύει ότι και οι προηγούμενες κλάσεις. Αυτές οι δύο μεταβλητές με βοήθησαν στο να εμφανίζω κάθε φορά ποιος είναι ο χρήστης που έχει συνδεθεί και να γνωρίζω ανά πάσα στιγμή της ενέργειες που κάνει στο πρόγραμμα .
- **Show():** Εδώ γίνεται η εμφάνιση των δεδομένων από την βάση μας στων πίνακα.
- **ArrayList<UserHandler> getSearchList(valuex:String):** Εδώ τραβάω όλα τα δεδομένα από την βάση δεδομένων.
- **getSearchedValue():** Με αυτή την κλάση παίρνω τα αποτελέσματα από την αναζήτηση που θέλω να κάνω.
- **executeSQLQuery(String : query, String : message):** Σε αυτή την κλάση βλέπουμε ότι χρησιμοποιώ 2 executeSQLQuery. Το ένα είναι για να μας εμφανίζει το μήνυμα λάθους/επιτυχίας. Και το άλλο είναι για να μην μας εμφανίσει τίποτα.
- **ArrayList<LogFileHandler> getLogList():** Εδώ φτιάχνουμε ένα ArayList για το lofile μας.
- **Show2():** Εδώ εκτυπώνω τα αποτελέσματα από το παραπάνω ArrayList σε ένα jTextPanel.

4)Products.

Εικόνα 5.0.1-6: UML Class Products

Στην κλάση products δίνεται η δυνατότητα στον χρήστη να επεξεργαστεί τα δεδομένα των προϊόντων του.

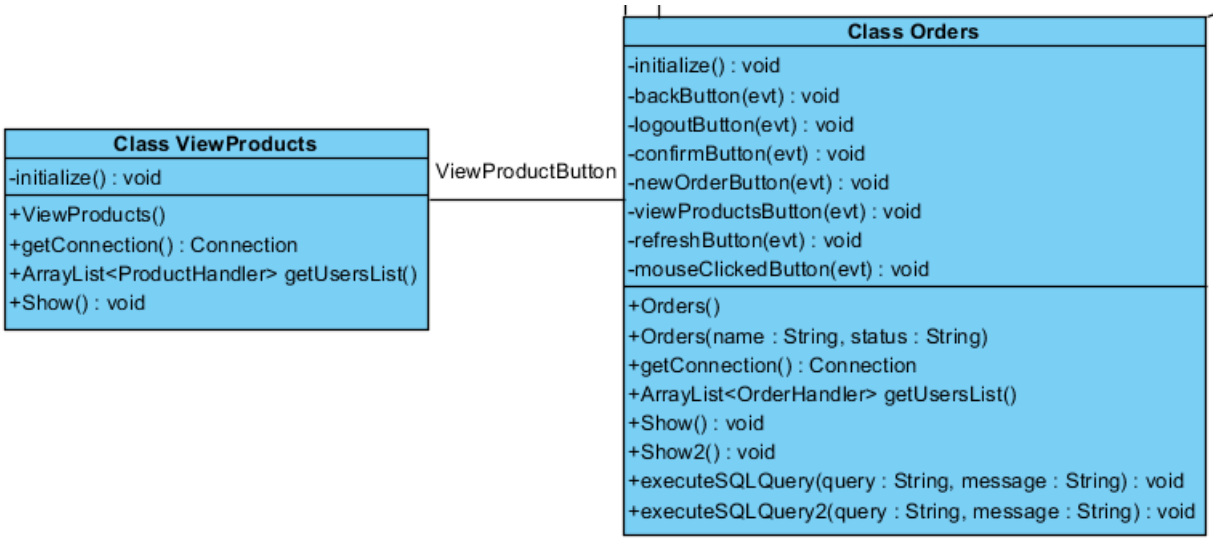
Σε αυτή την κλάση ως private μέθοδοι είναι:

- **Initialize():** Σε αυτό το σημείο αλλάζω/ορίζω το εικονίδιο που θα υπάρχει όταν ανοίξει το πρόγραμμα.
- **updateButton(evt):** Ανανεώνω στοιχεία από την βάση δεδομένων μου.
- **tableMouseClicked(evt):** Εμφανίζω στοιχεία από τον πίνακα και τα τοποθετώ στα κατάλληλα jTextField ώστε να μπορώ να τα επεξεργαστώ πιο γρήγορα.
- **insertButton(evt):** Το κουμπί για να κατευθυνθεί στην κλάση Insert. Όπου η κλάση είναι φτιαγμένη ακριβώς όπως την addUser απλά με διαφορετικό query.
- **deleteButton(evt):** Εδώ με την βοήθεια της sql διαγράφω μία τιμή από την βάση δεδομένων μου .
- **refreshButton(evt):** Απλά ανανεώνω τον πίνακα με τα στοιχεία που βρίσκονται στην βάση δεδομένων.
- **searchButton(evt):** Ψάχνω συγκεκριμένα στοιχεία και ανανεώνω τον πίνακα.
- **reportButton(evt):** Με την βοήθεια του JasperRepots εμφανίζω τα προϊόντα μου σε ένα pdf.
- **backButton(evt):** Παω στο Panel που βρισκόμουν πριν.
- **logOutButton(evt):** Το κουμπί για να κατευθυνθεί στην κλάση loggoutAnswer και να αποσυνδεθεί από το πρόγραμμα.

Και ως public έχω της μεθόδους που είναι ίδιοι με αυτούς που είχαν στο adminPanel.

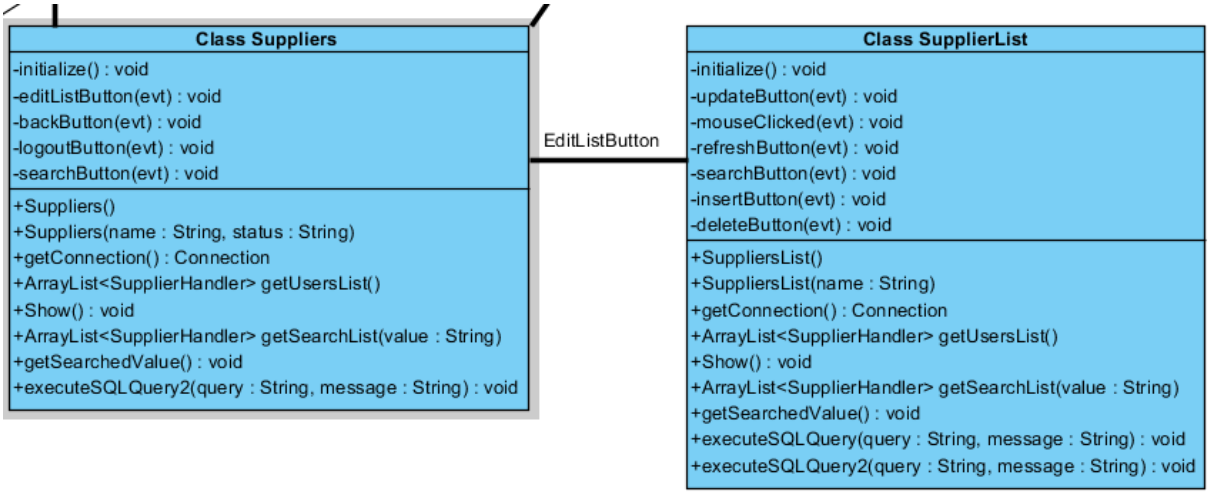
Γενικά όλες οι άλλες κλάσεις έχουν της ίδιες μεθόδους απλά απευθύνονται σε διαφορετικούς πίνακες, οπότε θα αναφερθούν παρακάτω μόνο σχηματικά.

5)Orders.



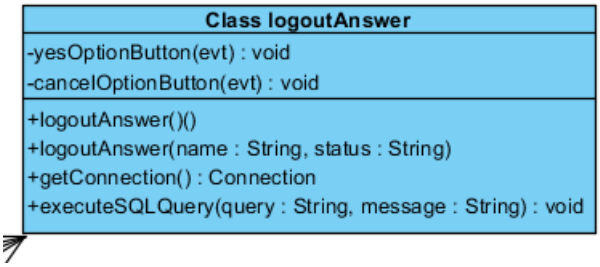
Εικόνα 5.0.1-7: UML Class Orders

6)Suppliers



Εικόνα 5.0.1-8: UML Class Suppliers+SuppliersList

7)logoutAnswer



Εικόνα 5.0.1-9: UML Class logoutAnswer

Όλες οι κλάσεις όμως συνδέονται με το logoutAnswer όπου τερματίζει την περιήγηση του χρήστη στο πρόγραμμα.

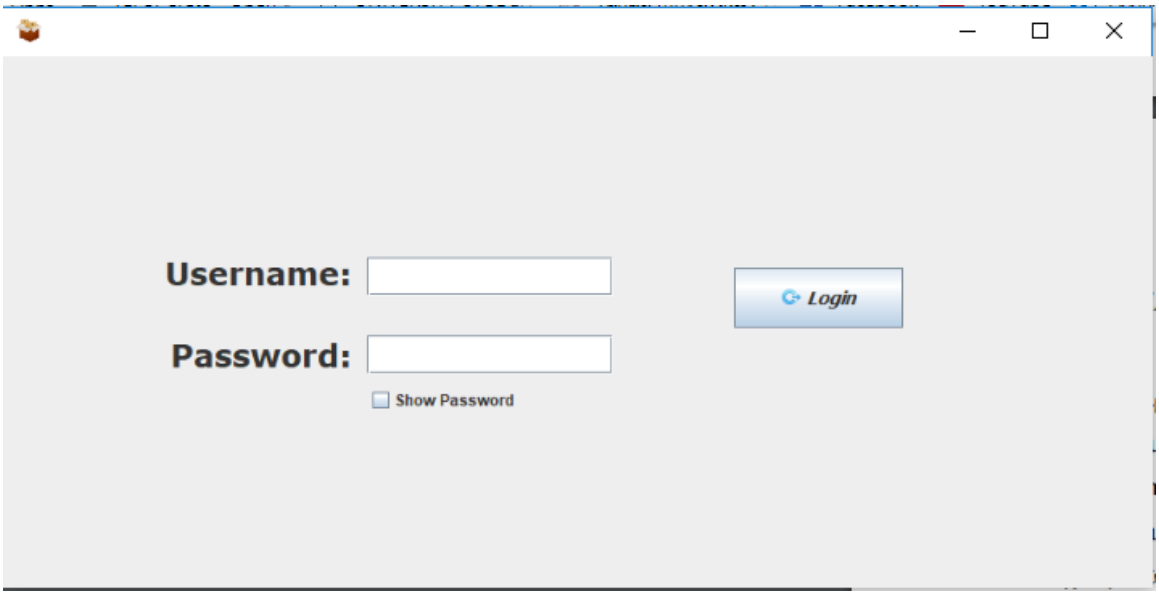
8) Handlers

Στο πρόγραμμα αυτό ,ο τομέας Handlers είναι ο πιο σημαντικός από όλους. Γιατί με αυτές της κλάσεις μπορούμε να τραβήξουμε τα δεδομένα μας, και να τα βάζουμε στα ArrayList μας. Γιαυτό χρησιμοποιήθηκαν getters για κάθε μία από αυτές.

5.0.2)Ανάλυση και επεξήγηση του προγράμματος.

Παρακάτω θα περιγράψουμε όλες της λειτουργίες του προγράμματος που αναπτύχθηκε. Έστω λοιπόν ένας εργαζόμενος (αποθηκάριος) στην αρχή της βάρδιας του . Το πρώτο πράγμα που θα πρέπει να κάνει είναι να μπει με τα στοιχεία που του έχει δώσει ο διαχειριστής, όπου είναι μοναδικά για κάθε χρήστη.

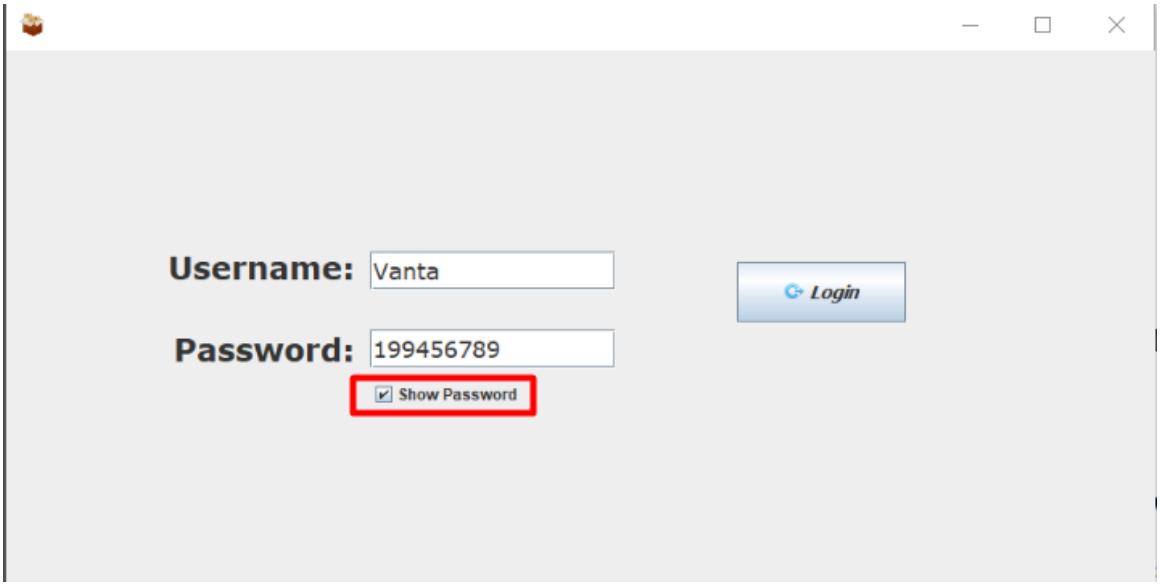
Το πρώτο 'παράθυρο' που ανοίγει πριν μπεις στο πρόγραμμα είναι το παρακάτω:



Εικόνα 5.0.2-1:Login User

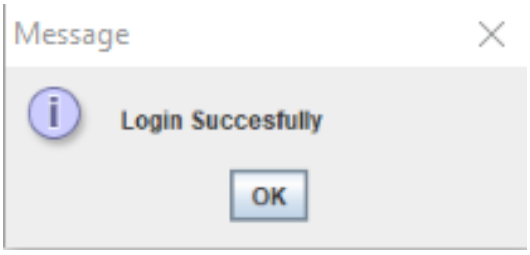
Αν δεν έχει τα στοιχεία που ζητούνται τότε δεν δικαιούται να εισέλθει στο πρόγραμμα. Κάθε λογαριασμός των αποθηκάριων δημιουργείται από τον διαχειριστή του προγράμματος. Και αμέσως δίδονται τα στοιχεία που πρέπει να εισάγουν για να μπουν στο πρόγραμμα.

Εάν πληκτρολογήσουν λάθος τον κωδικό που τους έχει δώσει ο διαχειριστής μπορούν να πατήσουν το κουμπί (Show Password) έτσι ώστε να βρουν το λάθος τους και να τον διορθώσουν ώστε να μπουν στο πρόγραμμα. Όπως φαίνεται στην παρακάτω φωτογραφία.



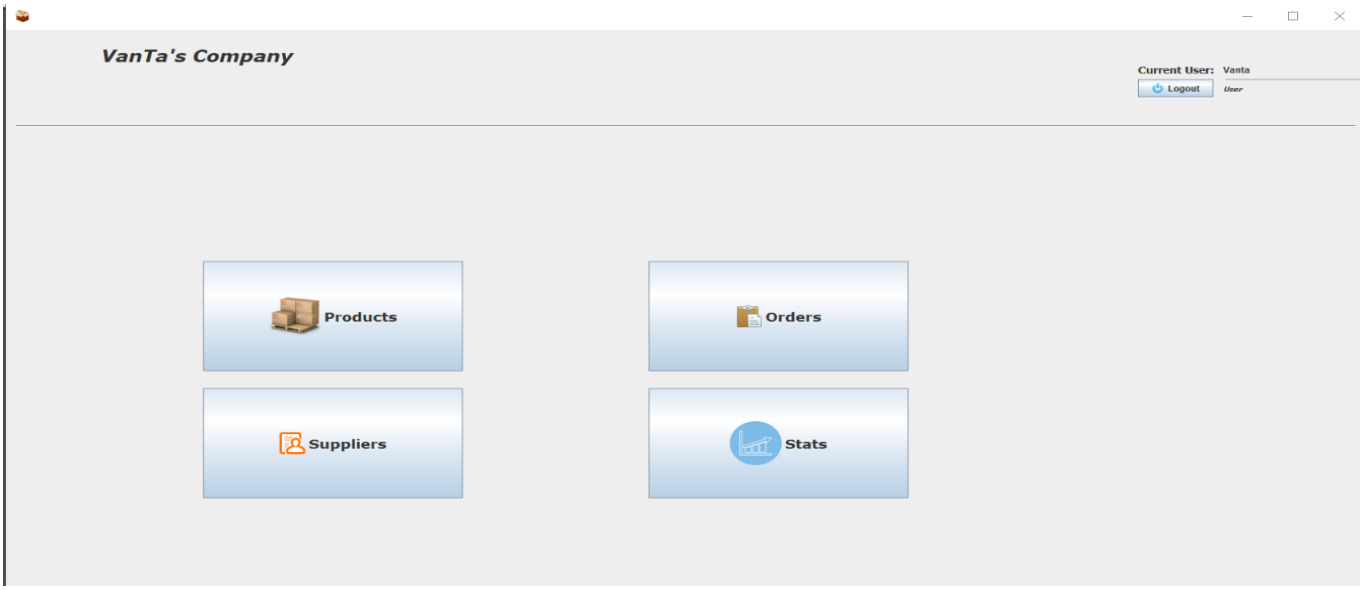
Εικόνα 5.0.2-2:Login User

Εάν τα στοιχεία που έδωσε ο αποθηκάριος είναι σωστά τότε πετάγεται ένα μήνυμα που επιβεβαιώνει τον χρήστη ότι εισήλθαν στο πρόγραμμα επιτυχώς.



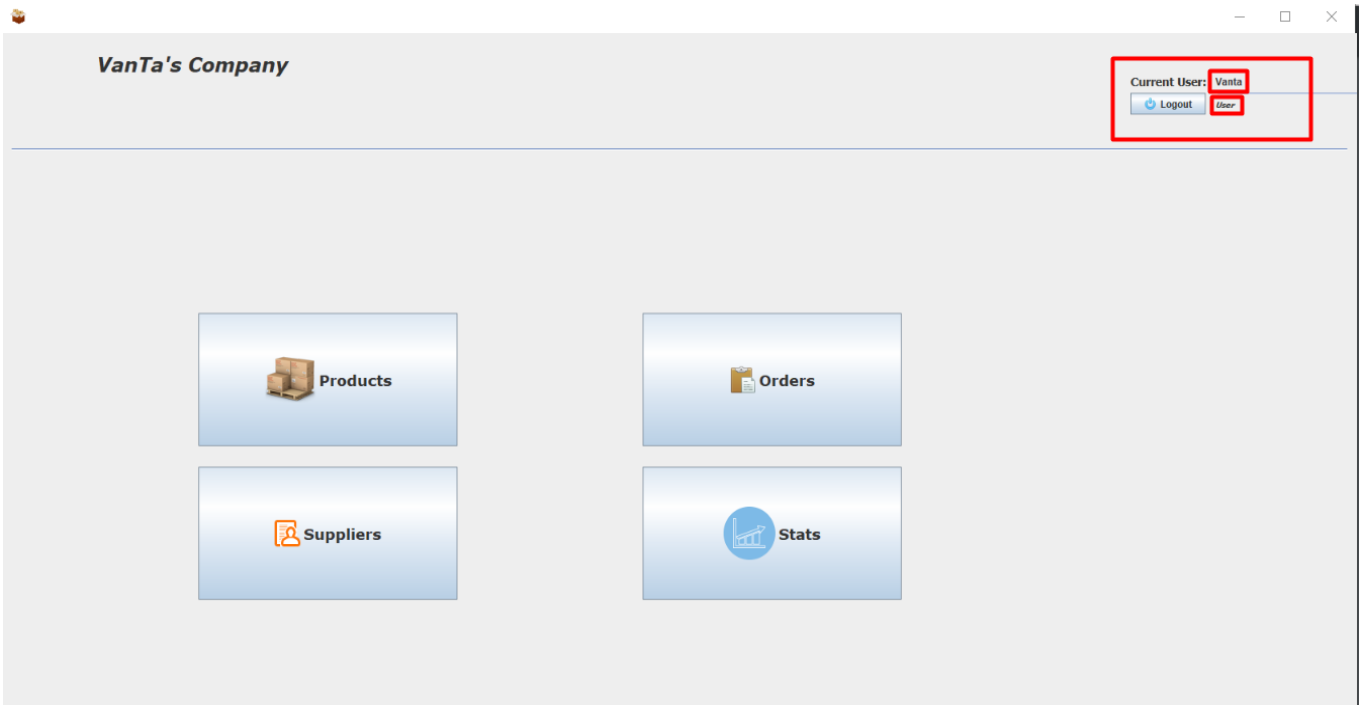
Εικόνα 5.0.2-3:Login Succesfully

Αφού εισέλθει στο πρόγραμμα του εμφανίζεται το εξής ‘παράθυρο’:



Εικόνα 5.0.2-4: CompanyJF Frame

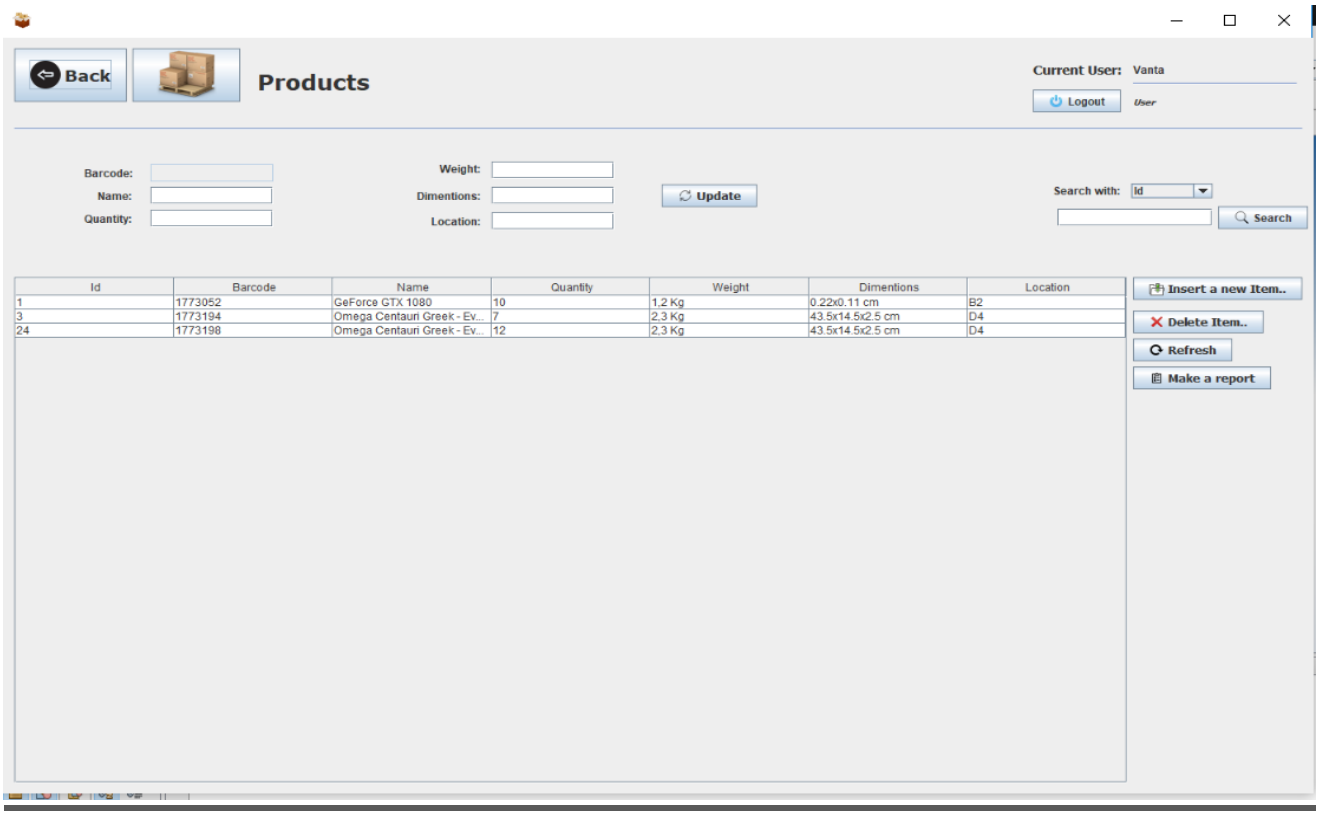
Πάνω δεξιά ,για διευκόλυνση του χρήστη, αναγράφονται το όνομα του αποθηκάριου που είναι συνδεδεμένος εκείνη την χρονική στιγμή αλλά και την ιδιότητα του(User η Admin). Σε όλη την διάρκεια της πλοήγησης του κάθε υπαλλήλου υπάρχει και η δυνατότητα αποσύνδεσης από τον λογαριασμό.



Εικόνα 5.0.2-5: CompanyJF Frame

Μόλις δώσει τα σωστά στοιχεία και εισέλθει μέσα στο πρόγραμμα εμφανίζονται τέσσερα διαφορετικά κουμπιά που το κάθε ένα έχει διαφορετικές ενέργειες. Τα κουμπιά είναι Products, Orders, Suppliers, Stats. Products για ενέργειες που αφορούν τα προϊόντα, Orders για ενέργειες που αφορούν της παραγγελίες, Suppliers που αφορούν τους προμηθευτές και Stats όπου δίνεται η δυνατότητα στον υπάλληλο να δει της ελλείψεις που έχουν τα προϊόντα. Δηλαδή αν η ποσότητα κάποιου προϊόν είναι λιγότερη από 15.

5.1)Ανάλυση Products.

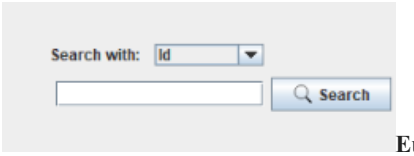


Εικόνα 5.1-1: Products

Όταν λοιπόν πατηθεί το κουμπί Products μας εμφανίζεται το παράθυρό που φαίνεται παραπάνω .

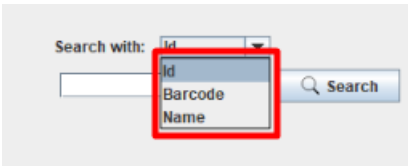
Στο παράθυρο των Products ο υπάλληλος μπορεί να δει μία λίστα από τα προϊόντα που υπάρχουν στην αποθήκη, καθώς και της λεπτομέρειες του κάθε προϊόν.

Για να μπορεί να προηγηθεί εύκολα και γρήγορα ο υπάλληλος έχει προστεθεί η δυνατότητα αναζήτησης , όπου μπορεί εύκολα να αναζητήσει το προϊόν που θέλει να μάθει διάφορες πληροφορίες. Διότι αν υπάρχουν πολλά προϊόντα καταχωρημένα είναι δύσκολο ως προς τον υπάλληλο να αναζητήσει αυτό που θέλει. Οπότε η αναζήτηση του προϊόν καθιστάτε δύσκολη.



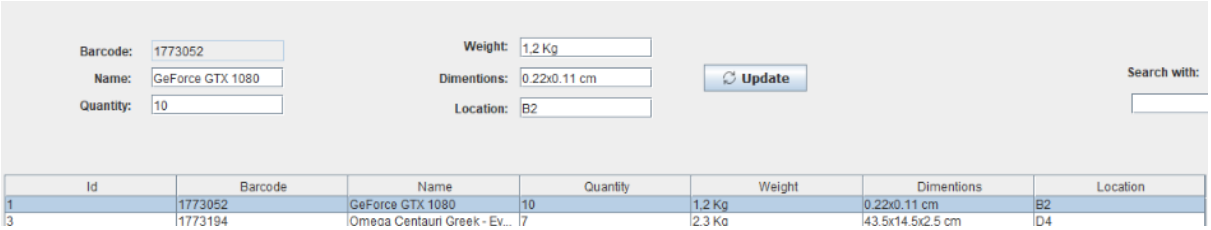
Εικόνα 5.1-2: Procut Search

Για να μπορεί να αναζητηθεί ένα προϊόν , μπορεί να βάλει στοιχεία όπου είναι μοναδικά σε κάθε προϊόν όπως Id, Barcode ή το Όνομα.



Εικόνα 5.1-3: Procut Search values

Υπάρχει και η δυνατότητα διόρθωση λάθους , καθώς μπορεί να αλλάξει κάποιες από της τιμές ενός προϊόντος. Το Barcode δεν μπορεί να τροποποιηθεί καθώς είναι μοναδικό.

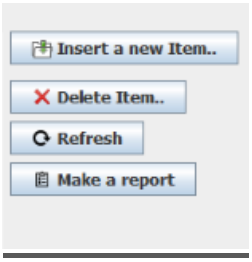


Εικόνα 5.1-4: Barcode

Οι ενέργειες που πρέπει να προβούν για να τροποποιηθούν κάποια στοιχεία από ένα προϊόν είναι να κάνει κλικ ,ο υπάλληλος, το προϊόν που θέλει να αλλάξει της πληροφορίες του. Αμέσως μετά τα στοιχεία του θα συμπληρωθούν πάνω στα κενά και θα είναι έτοιμα προς επεξεργασία. Όπως παρατηρείτε το Barcode δεν του δίνεται η δυνατότητα επεξεργασίας, για τον λόγο που γράφεται παραπάνω.

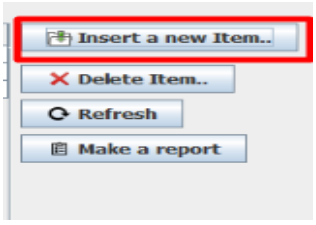
5.1.1)Επιπλέον δυνατότητες.

Εκτός από την τροποποίηση και την αναζήτηση των στοιχείων υπάρχουν και οι εξής δυνατότητες.



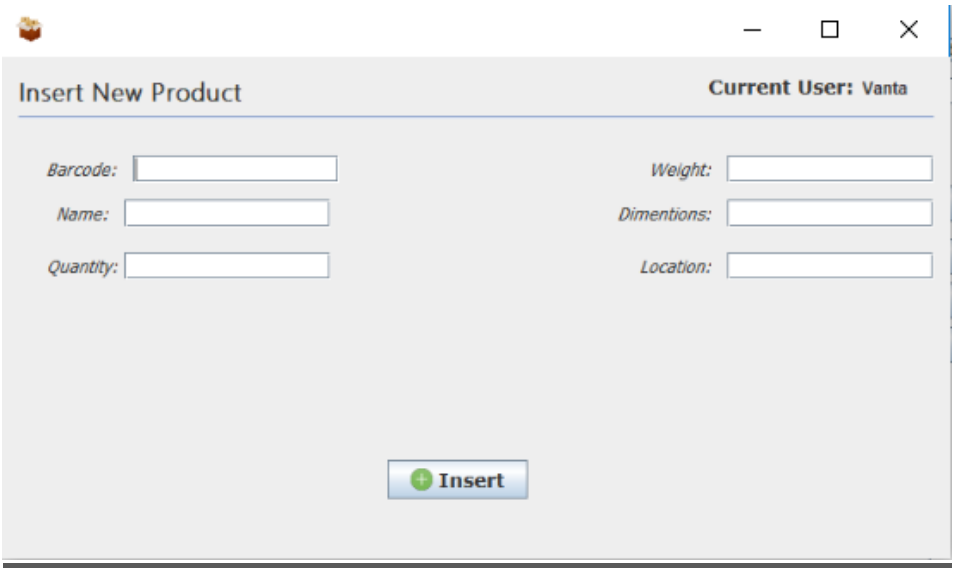
Εικόνα 5.1.1-1: Procut buttons

a) Insert a new items.



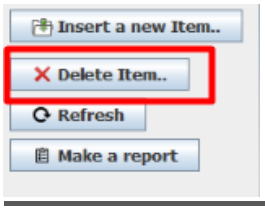
Εικόνα 5.1.1-2: Insert a Product

Πατώντας αυτό το κουμπί εμφανίζεται ένα παραθυράκι όπου σου δίνεται η δυνατότητα να εκχωρήσεις ένα καινούριο προϊόν στην αποθήκη. Εδώ προσθέτεις όλα τα χαρακτηριστικά του καινούριου προϊόντος έτσι ώστε να καταχωρηθεί στην βάση δεδομένων όπου εδώ ο υπάλληλος θα πρέπει να είναι προσεκτικός όταν καταχωρεί το καινούριο προϊόν μην κάνει λάθος το Barcode . Γιατί δεν υπάρχει τρόπος επεξεργασίας των στοιχείων που έδωσε και θα πρέπει μετά να διαγράψει το προϊόν και να το ξανά προσθέσει στην βάση δεδομένων.



Εικόνα 5.1.1-3: Insert a Product

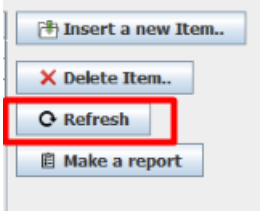
b) Delete.



Εικόνα 5.1.1-4: Delete a Product

Πατώντας το κουμπί αυτό μας δίνεται η δυνατότητα διαγραφής κάποιας εκχώρησης . Επιλέγοντας πρώτα το προϊόν που θέλουμε να διαγράψουμε και μετά πατάμε το παραπάνω κουμπί. Με αποτέλεσμα να διαγραφεί από την βάση δεδομένων.

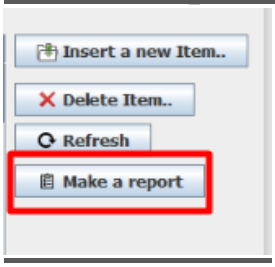
c) Refresh.



Εικόνα 5.1.1-5: Refresh Product

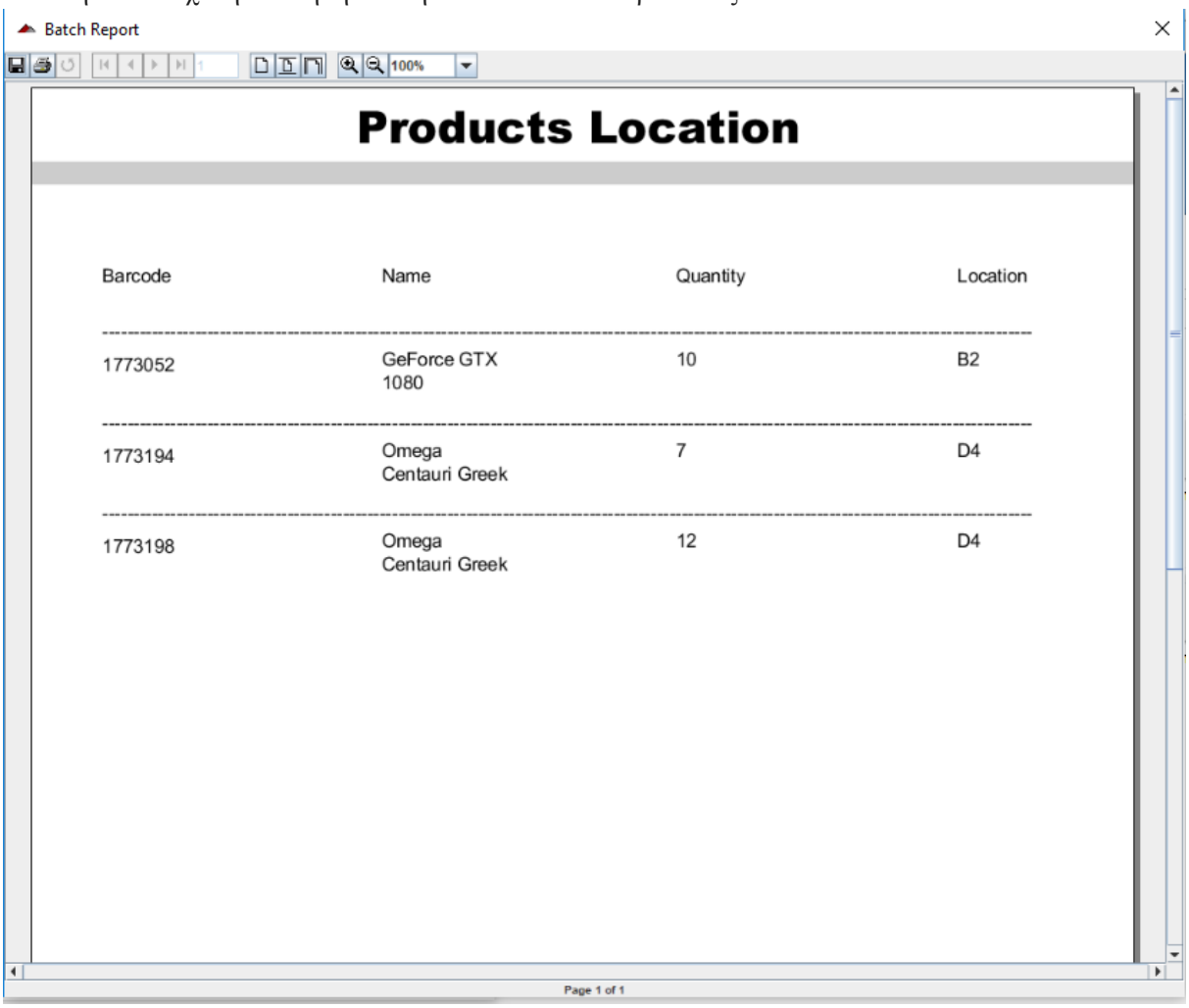
Κάθε φορά που κάνουμε μια αλλαγή στα προϊόντας μας ή μια εκχώρηση θα πρέπει να πατάμε το συγκεκριμένο κουμπί έτσι ώστε να ανανεώνεται ο πίνακας των προϊόντων με την ανανεωμένη βάση δεδομένων.

d) Make a report.



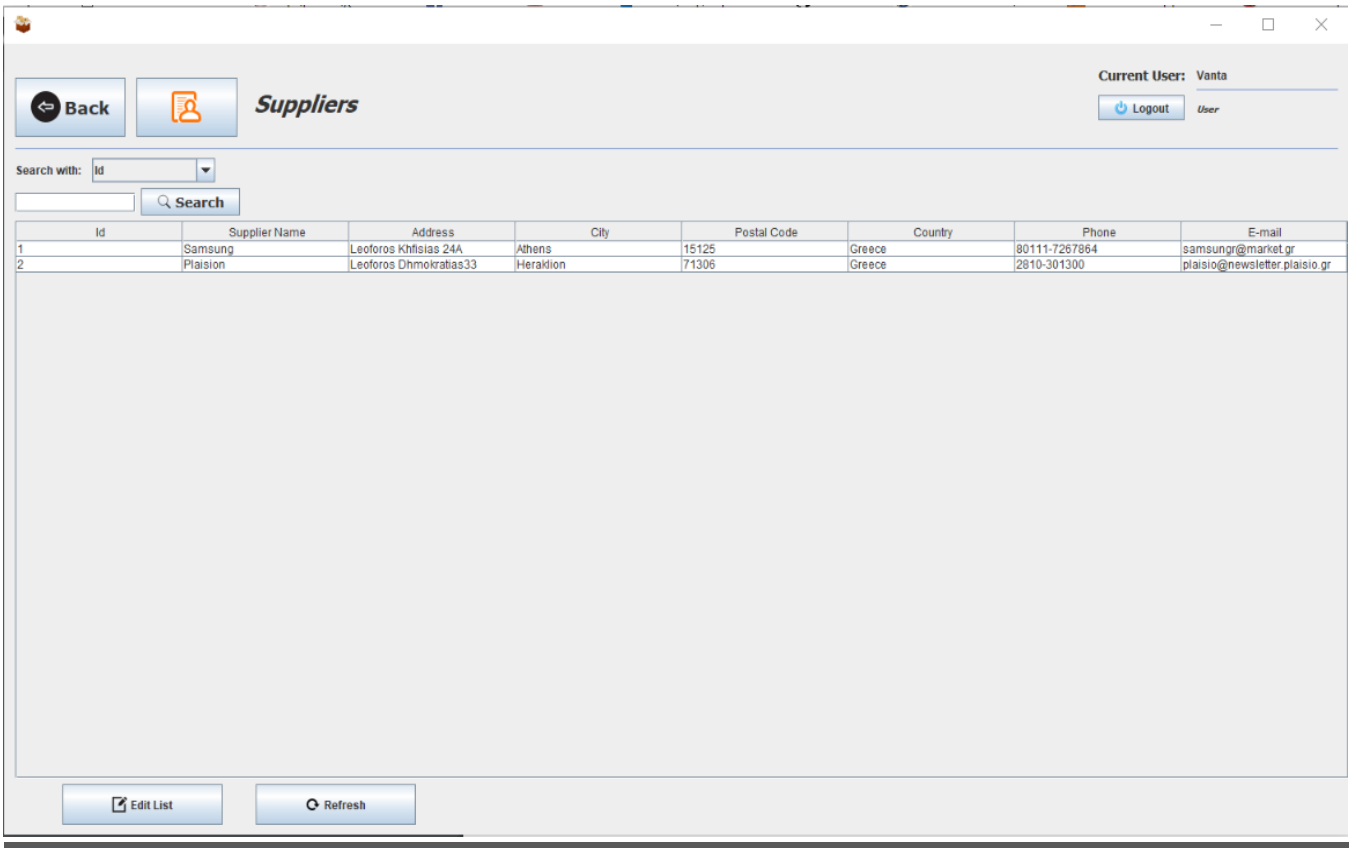
Εικόνα 5.1.1 -6: Make a report

Σκοπός του προγράμματος είναι η διευκόλυνση του υπαλλήλου. Είναι δύσκολο για έναν υπάλληλο να θυμάται τόσες πολλές λεπτομέρειες , για τόσο πολλά προϊόντα . Με αφορμή αυτό, θα ήταν πιο εύκολο να εκτυπωθούν σε μία κόλλα Α4 και από εκεί να κάνει της διάφορες λειτουργίες που του ζητούνται . Χάρις το JasperSoft Reports το καταφέραμε αυτό και μπορούμε να εκτυπώσουμε ότι θέλουμε από την βάση δεδομένων. Στην προκυμμένη περίπτωση θέλουμε να εκτυπώσουμε όλα τα προϊόντα που έχουμε. Αλλά κάνοντας την αναζήτηση πιο εύκολη θα χρειαστούμε μόνο το Barcode , το Όνομα του προϊόντος , την ποσότητα που έχει η αποθήκη και την τοποθεσία του προϊόντος .



Εικόνα 5.1.1-7: Report

5.2)Ανάλυση Suppliers.

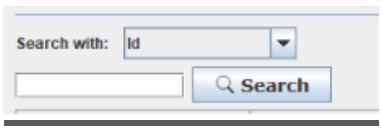


Εικόνα 5.2-1: Suppliers

Όταν λοιπόν πατηθεί το κουμπί Suppliers μας εμφανίζεται το παράθυρό που φαίνεται παραπάνω .

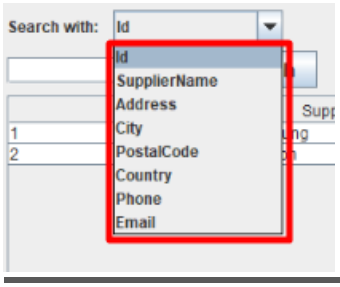
Στο παράθυρο των Suppliers ο υπάλληλος μπορεί να δει μία λίστα από τους προμηθευτές οπού γεμίζει προϊόντα την αποθήκη, καθώς και της λεπτομέρειες του κάθε προμηθευτή. Έτσι με αυτές της πληροφορίες διευκολύνει την δημιουργία μιας νέας παραγγελίας.

Για να μπορεί να προηγηθεί εύκολα και γρήγορα ο υπάλληλος έχει προστεθεί η δυνατότητα αναζήτησης όπως και με τα προϊόντα, όπου μπορεί εύκολα να αναζητήσει τον προμηθευτή που θέλει να μάθει διάφορες πληροφορίες. Διότι αν υπάρχουν πολλοί καταχωρημένοι προμηθευτές είναι δύσκολο ως προς τον υπάλληλο να αναζητήσει αυτόν που θέλει. Οπότε η αναζήτηση του κάθε προμηθευτή καθιστάτε δύσκολη.



Εικόνα 5.2-2: Suppliers Search

Όπως με τα προϊόντα, για να μπορεί να αναζητηθεί ένας προμηθευτής , θα μπορεί να βάλει στοιχεία όπου είναι μοναδικά σε κάθε προμηθευτή όπως Id, SupplierName(Το όνομα του προμηθευτή) ,Address(Διεύθυνση), City(Πόλη που βρίσκεται η επιχείρηση), PostalCode(Ο ταχυδρομικός κώδικας), Country(Την χώρα από όπου είναι), Phone(Το τηλέφωνο της) ακόμα και το Email.

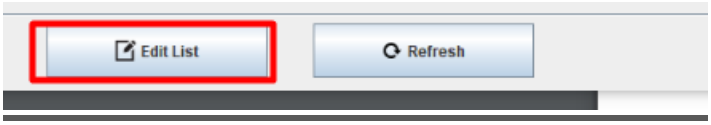


Εικόνα 5.2-3: Suppliers search

5.2.1)Επιπλέον δυνατότητες.

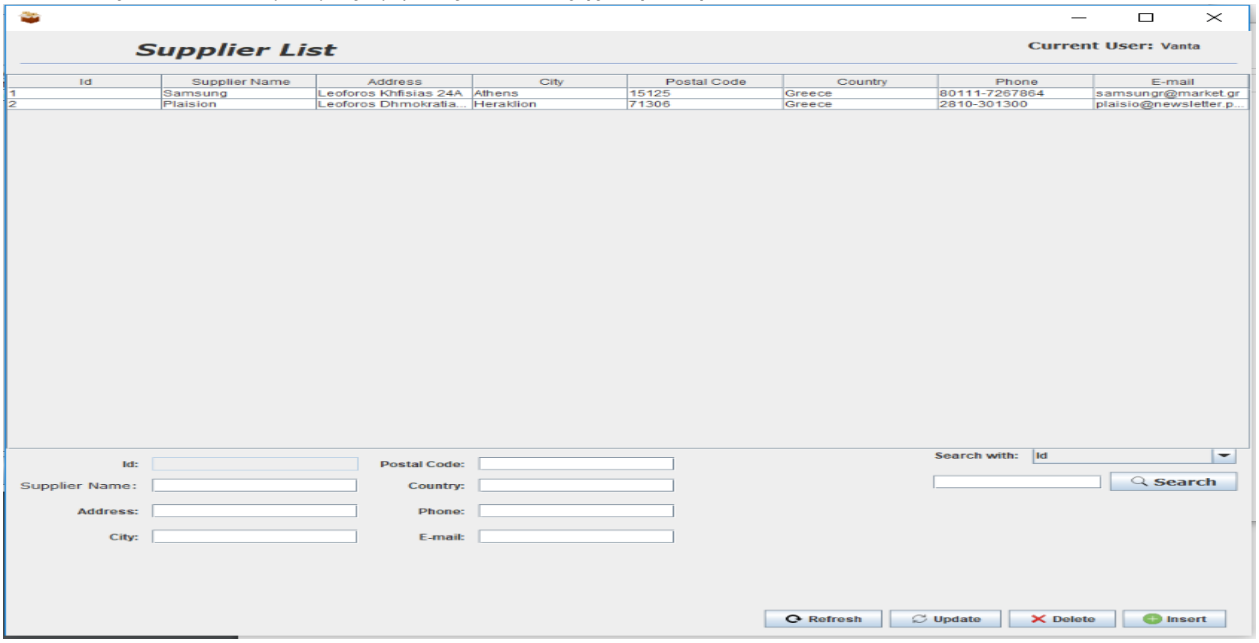
Σε αυτό το παράθυρο εκτός από την γρήγορη αναζήτηση των προμηθευτών , έχουν προστεθεί 2 παραπάνω ενέργειες.

a) Edit List.



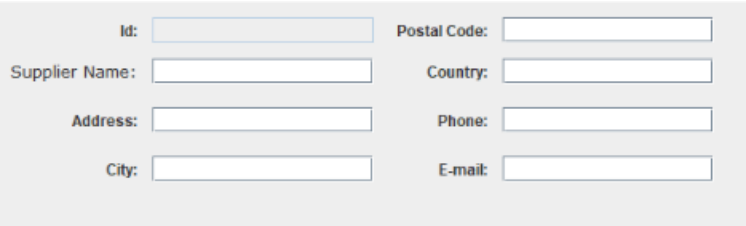
Εικόνα 5.2.1-1: Suppliers edit list

Πατώντας αυτό το κουμπί μας εμφανίζεται το εξής παράθυρο.



Εικόνα 5.2.1-2: Suppliers List

Όπου μας ανοίγει ξανά έναν πίνακα με της πληροφορίες των προμηθευτών, απλά εδώ μας δίνεται η δυνατότητα επεξεργασίας αυτών των δεδομένων.



Εικόνα 5.2.1-3: Suppliers edit

Πάλι εδώ έχουμε έναν μοναδικό αριθμό για κάθε προμηθευτή, οπότε το πρόγραμμα δεν αφήνει περιθώρια τροποποίησης αυτή της τιμής.

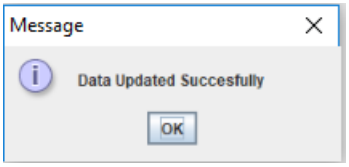
Εδώ έχουμε μία ανάλογη συμπεριφορά για την τροποποίηση των στοιχείων, όπως γινόταν δηλαδή και στα στοιχεία των προϊόντων . Επιλέγουμε των προμηθευτή που θα θέλαμε να τροποποιήσουμε , αμέσως μετά τα στοιχεία μας εμφανίζονται στα κενά πεδία στην εικόνα παραπάνω. Όπου έχουμε 4 διαφορετικές λειτουργίες.

Insert. - Το κουμπί αυτό εισάγει έναν καινούριο προμηθευτή.

Delete. – Το κουμπί αυτό διαγράφει μία τιμή.

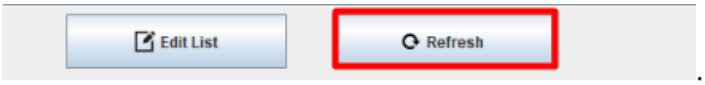
Refresh. – Ανανεώνει τον πίνακα των τιμών που εμφανίζουμε στο παράθυρο μας.

Update. – Ανανεώνει τις τιμές στην βάση δεδομένων μας. Όταν πατήσουμε το συγκεκριμένο κουμπί και γίνει επιτυχώς η ανανέωση μας εμφανίζεται το παρακάτω μήνυμα , δηλαδή ότι η ανανέωση έγινε επιτυχώς.



Εικόνα 5.2.1-4: Suppliers updated successfully

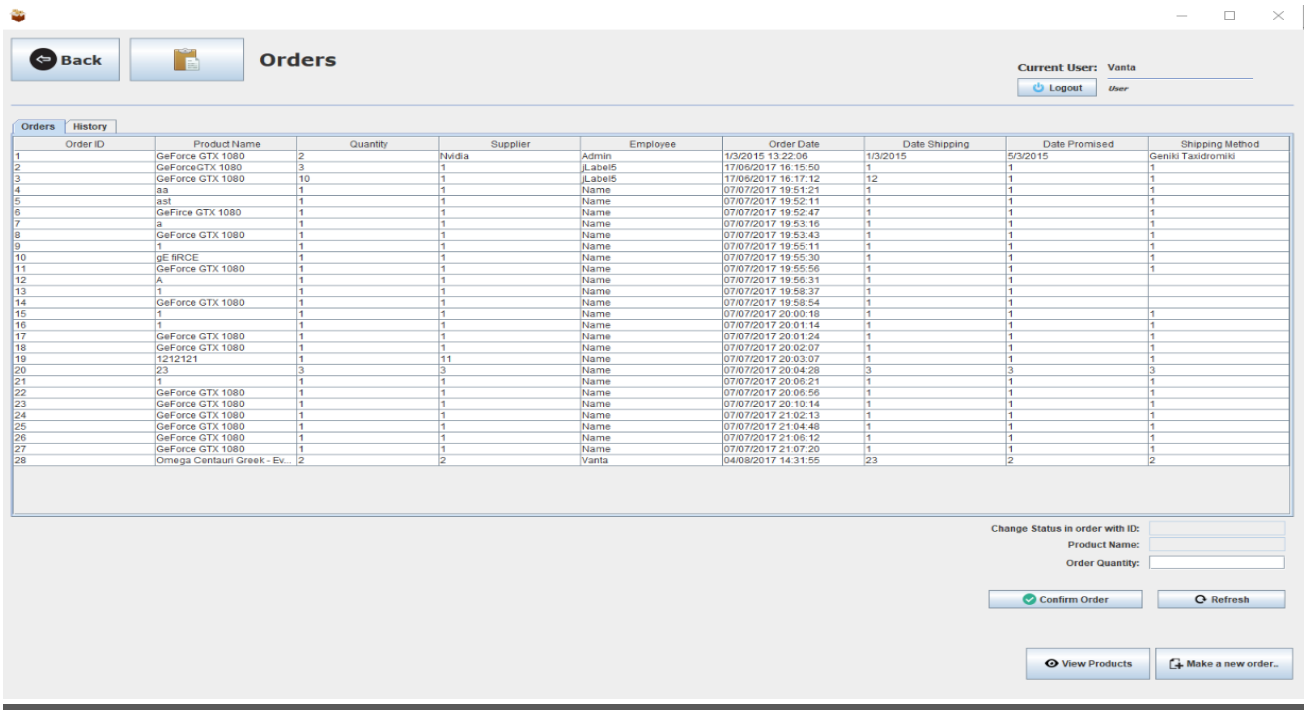
b) Refresh



Αυτό το κουμπί εμφανίζει ξανά τα στοιχεία από την αρχή. Το κουμπί αυτό προστέθηκε για να εμφανίζονται τα ανανεωμένα στοιχεία.

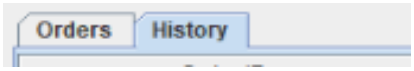
Εικόνα 5.2.1-5: Suppliers refresh

5.3)Ανάλυση Orders.

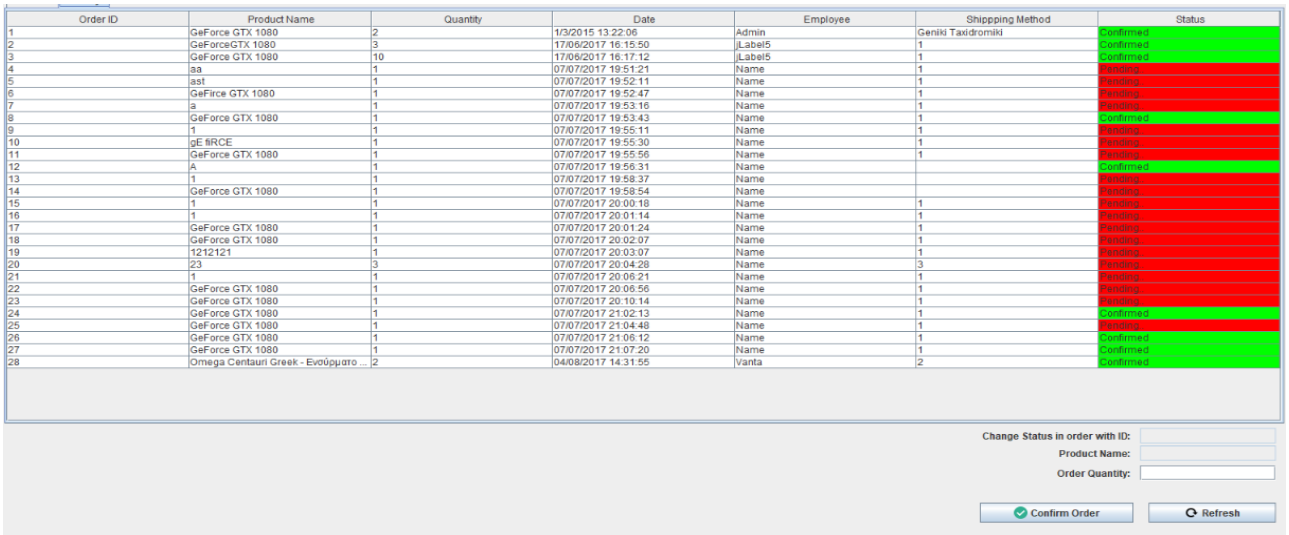


Εικόνα 5.3-1: Orders

Όταν λοιπόν πατηθεί το κουμπί Orders μας εμφανίζεται το παράθυρό που φαίνεται παραπάνω .



Στο παράθυρο των Orders ο υπάλληλος μπορεί να δει δύο λίστες, η μία όπου ονομάζεται Orders που δίνεται η δυνατότητα στον υπάλληλο να δει της πληροφορίες για της παραγγελίες όπου έχουν γίνει. Και το Ιστορικό των παραγγελιών, όπου μπορεί να επικυρώσει της καινούριες παραγγελίες. *Οι τιμές που έχουν εισαχθεί και φαίνονται στην φωτογραφία είναι ενδεικτικές.*

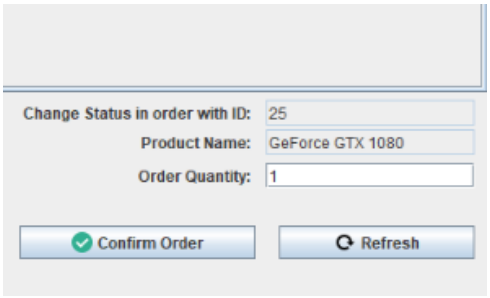


Εικόνα 5.3-2: Orders

Με πράσινο χρώμα εμφανίζονται οι παραγγελίες που έχουν έρθει στην αποθήκη , και με κόκκινο χρώμα οι παραγγελίες που ναι μεν έχει γίνει η παραγγελία αλλά δεν έχει φτάσει ακόμα στην αποθήκη. Με αυτόν τον τρόπο μπορεί ο υπάλληλος να ξέρει τι παραγγελίες περιμένει ακόμα.

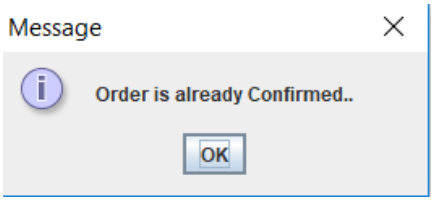
Εικόνα 5.3-3: Orders status

Για να αλλάξουμε την κατάσταση της παραγγελίας απλός πατάμε σε μία παραγγελία που θέλουμε να της αλλάξουμε κατάσταση.



Εικόνα 5.3-4: Confirm Order

Και μετά που θα πατήσουμε στην παραγγελία όπου θέλουμε να αλλάξουμε τότε πατάμε το Confirm Order.Έτσι το Pending.. που γράφεται στον πίνακα μας θα αλλάξει χρώμα και θα γίνει πράσινο και θα αλλάξει κατάσταση , δηλαδή από Pending .. θα γίνει Confirmed .Εάν όμως είναι ήδη Confirmed μας πετάγεται το παρακάτω μήνυμα λάθους.



Εικόνα 5.3-5: Orders/Message confirmation

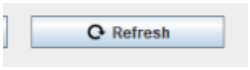
Με το που γίνει μία παραγγελία Confirmed , ανανεώνει την βάση δεδομένων ότι ήρθε η παραγγελία και επίσης αυξάνει το προϊόν κατά την ποσότητα που αναγράφεται στο Quantity.

5.3.1)Επιπλέον δυνατότητες

Σε αυτό το παράθυρο εκτός από την ανασκόπηση των παραγγελιών και από την επιβεβαίωση των παραγγελιών δίνονται στον υπάλληλο άλλες τρείς δυνατότητες.

a) Refresh.

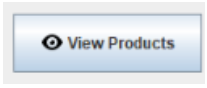
Όπου γίνεται μία ανανέωση των πινάκων(Orders/History).



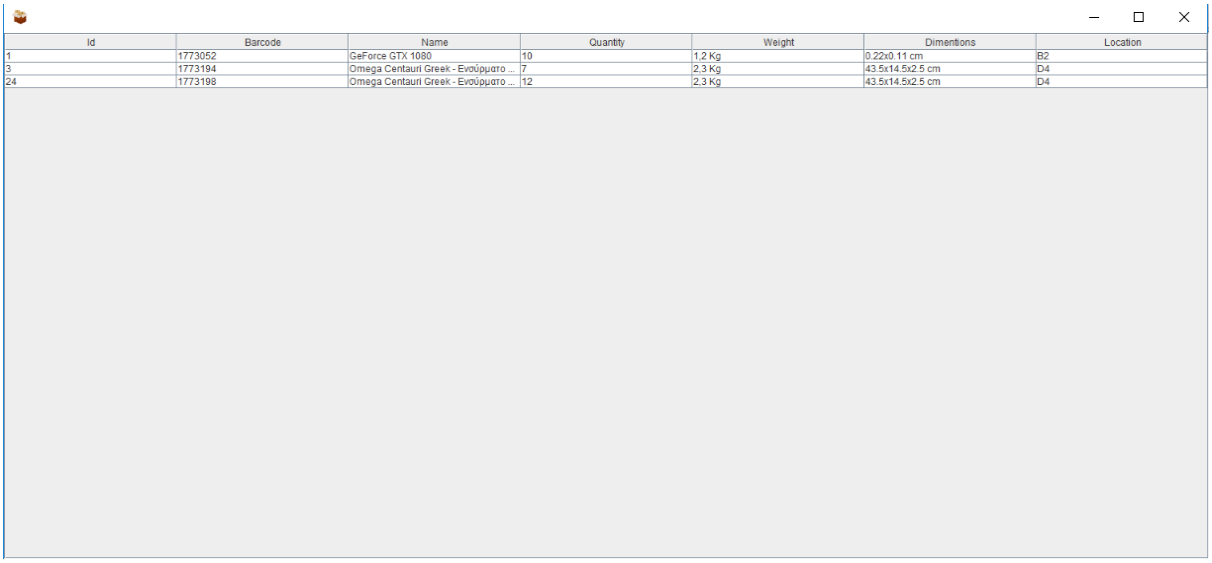
Εικόνα 5.3.1-1: Refresh

b) View Products.

Εδώ δίνεται η δυνατότητα στον υπάλληλο να δει τα προϊόντα και της ποσότητες τους ώστε να αποφασίσει τι πρέπει να παραγγείλει.



Εικόνα 5.3.1-2: View Products

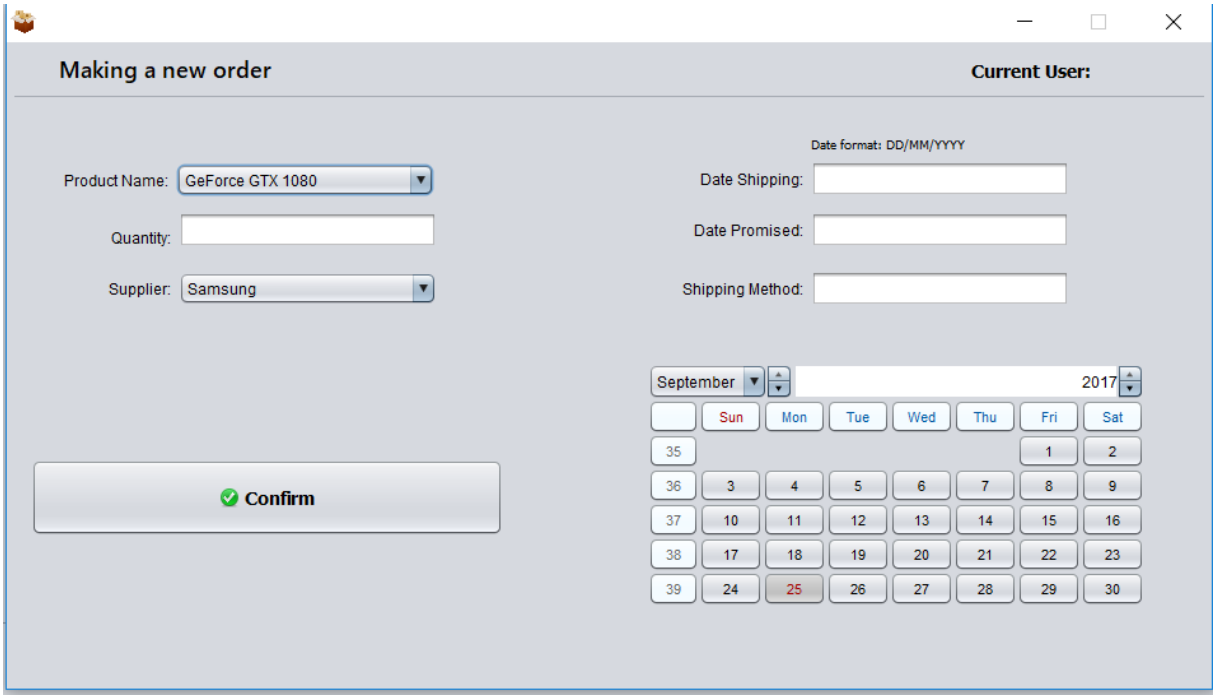


Id	Barcode	Name	Quantity	Weight	Dimentions	Location
1	1773052	GeForce GTX 1080	10	1,2 Kg	0.22x0.11 cm	B2
3	1773194	Omega Centauri Greek - Ενσύρματο ...	7	2,3 Kg	43.5x14.5x2.5 cm	D4
24	1773198	Omega Centauri Greek - Ενσύρματο ...	12	2,3 Kg	43.5x14.5x2.5 cm	D4

Εικόνα 5.3.1-3: Product table in orders

c) Make a new order.

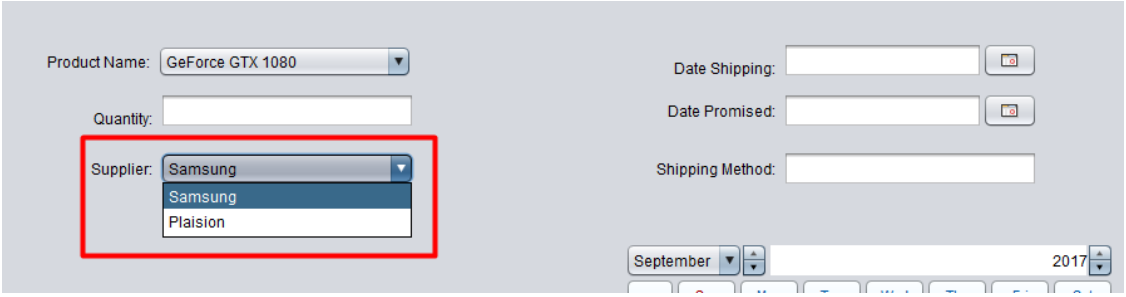
Εδώ ο υπάλληλος μπορεί να κάνει μία καινούρια παραγγελία. Καθώς όταν πατήσει το κουμπί του εμφανίζει το παρακάτω παράθυρο.



Εικόνα 5.3.1-4: Creating a new order

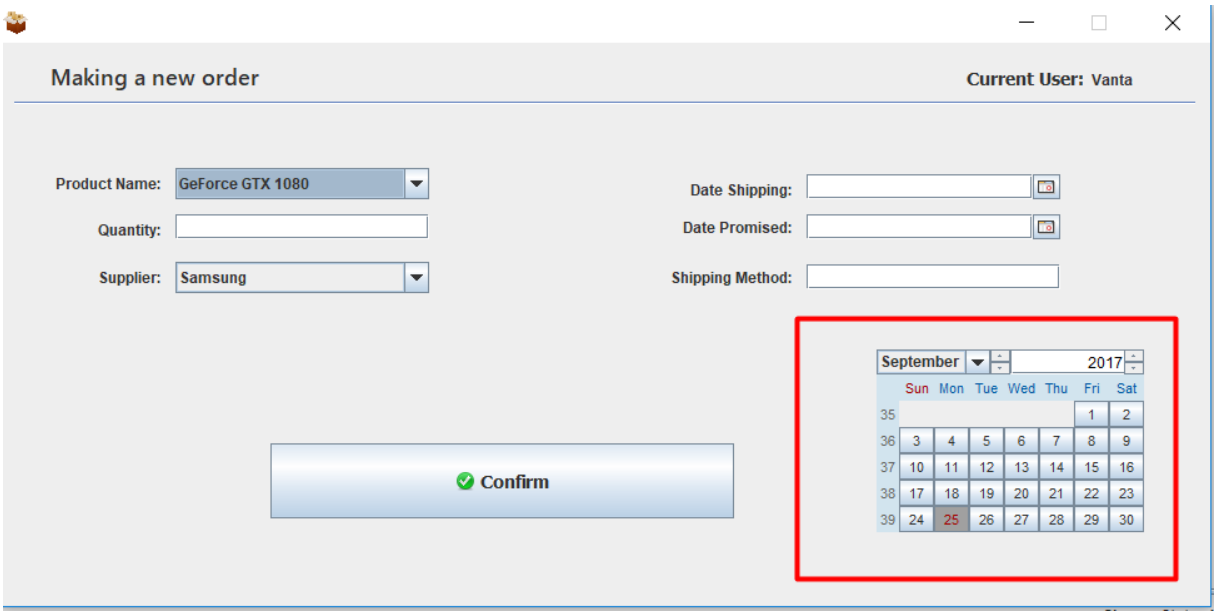
Όπου αν παρατηρήσετε στο Product Name υπάρχει μία μπάρα που επιτρέπει να παραγγείλει μόνο προϊόντα που υπάρχουν καταχωρημένα στην βάση δεδομένων. Με το που δοθούν οι κατάλληλες πληροφορίες πρέπει να πατήσει το κουμπί Confirm έτσι ώστε να προχωρήσει η διαδικασία παραγγελίας του προϊόντος. Εάν θέλει να παραγγείλει κάτι διαφορετικό αρχικά πρέπει να πάει στο κουμπί Products και να προσθέσει όλες της πληροφορίες του καινούριου προϊόντος. Μετά θα μπορεί να κάνει μία παραγγελία με το καινούριο προϊόν.

Όπως επίσης και τα ονόματα των προμηθευτών τραβιούνται από την βάση δεδομένων, δηλαδή ο αποθηκάριος πρέπει να έχει καταχωρημένο τα στοιχεία του προμηθευτή έτσι ώστε να γίνει η παραγγελία από τον συγκεκριμένο.



Εικόνα 5.3.1-5: Creating a new order/Supplier Names

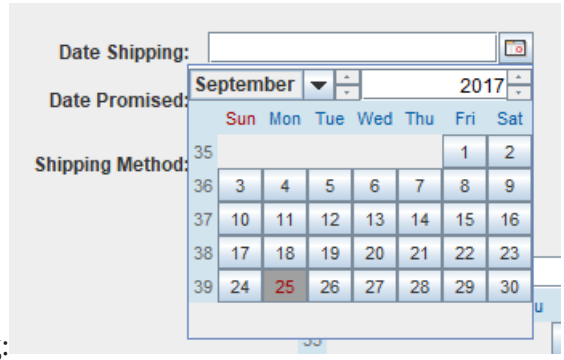
Έχει προστεθεί και ένα ημερολόγιο ώσπου ο αποθηκάριος να μπορεί να ελέγξει της μέρες της εβδομάδας έτσι ώστε να αποφύγει λάθος που μπορεί να έχει επιρροή στην αποθήκη.



Εικόνα 5.3.1-6: Creating a new order/Date

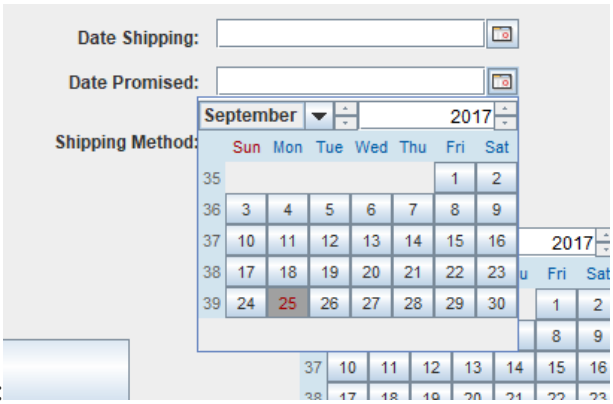
Το Date Shipping και το Date Promised επιλέγονται από ένα popup όπου αναγράφεται η ημερομηνία.

Date Shipping:



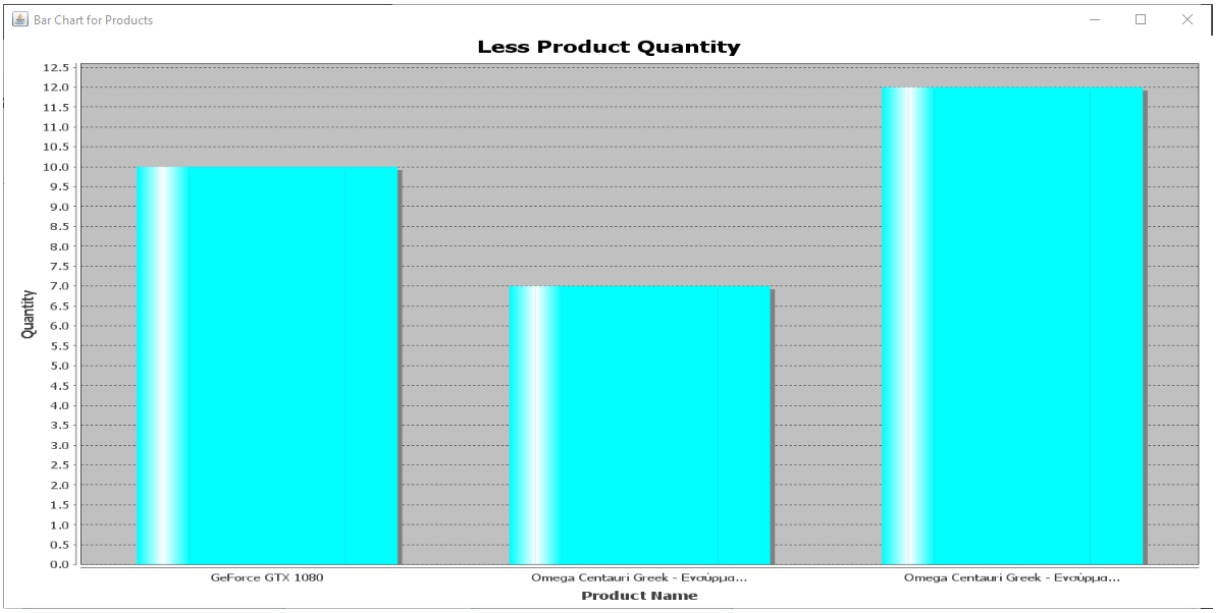
Εικόνα 5.3.1-7: Creating a new order/Date Shipping

Date Promised:



Εικόνα 5.3.1-8: Creating a new order/Date Promised

5.4)Ανάλυση Stats.



Εικόνα 5.4-1: Stats

Όταν λοιπόν πατηθεί το κουμπί Stats μας εμφανίζεται το παράθυρό που φαίνεται παραπάνω .

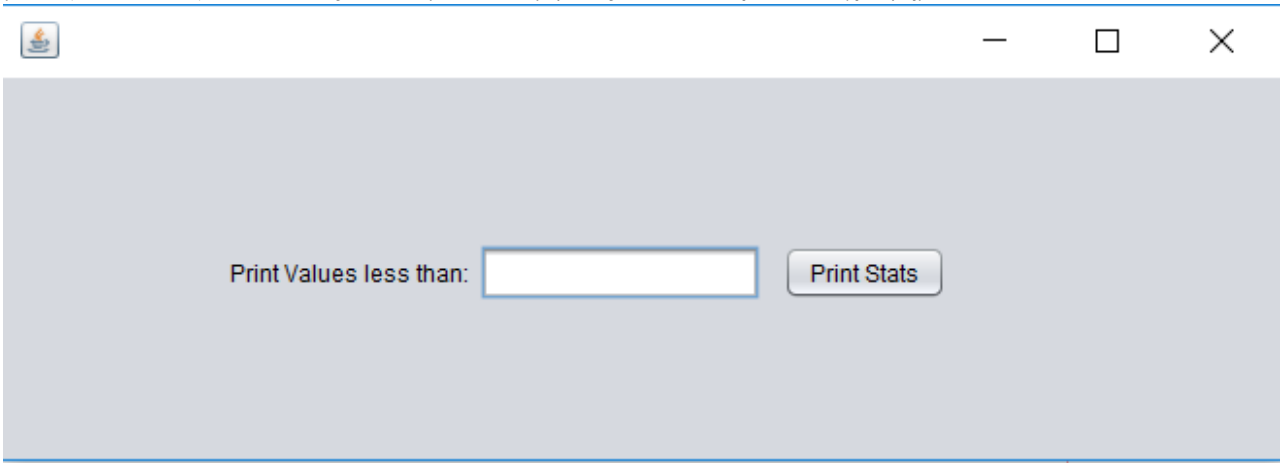
Το συγκεκριμένο παράδειγμα έχει φτιαχτεί έτσι ώστε να βγάζει τα προϊόντα που η ποσότητα τους μέσα στην αποθήκη είναι κάτω από 15 . Αυτό το παράθυρο δημιουργήθηκε γιατί κάθε υπάλληλος πρέπει να γνωρίζει ανά πάσα στιγμή πότε πρέπει να κάνει μια καινούρια παραγγελία και να γεμίσει την αποθήκη του, έτσι ώστε ποτέ να μην μένει εκτός προϊόντα και να γίνεται μία σωστή λειτουργίας μιας αποθήκης. Έτσι η αποθήκη πάντα θα έχει προϊόντα να δώσει και θα αποφεύγει τον κίνδυνο να μείνει χωρίς προϊόντα .

Τα αποτελέσματα που εμφανίζουν τα στατιστικά μπορούν να αλλάξουν ανάλογα με αυτά που επιθυμεί ο αγοραστείς του προγράμματος. Αλλάζοντας το query (δηλαδή την ερώτηση που κάνουμε στην sql βάση μας)

```
String query="SELECT * FROM products where Quantity<15";
```

Εικόνα 5.4-2: Stats SQL query

Άρα για την ευκολία του χρήστη προστέθηκε ένα παράθυρο όπου θα μπορεί ο εκάστοτε εργαζόμενος να μπορεί να επεξεργαστεί αυτό το παράθυρο και να εμφανίζει τα δεδομένα που θα θέλει να δει. Εδώ αφού βάλει την ποσότητα που δεν θέλει την πληκτρολογεί στο κουτί εκεί μέσα(JTextField). Πατώντας το κουμπί του εμφανίζεται το παραπάνω γράφημα.



Εικόνα 5.4-3: Updated stats

Έτσι ο υπάλληλος θα μπορεί να δει τα προϊόντα τα οποία έχει έλλειψη η αποθήκη. Εμείς αρχικά είχαμε βάλει μία ενδεικτική τιμή για το παράδειγμα μας, έτσι ώστε να βγάλουμε ένα γράφημα. Τώρα ο εργαζόμενος θα πρέπει να πληκτρολογήσει την μη επιθυμητή ποσότητα και να πατήσει το κουμπί.

Αυτό το καταφέραμε με το διάβασμα της τιμής που είχαμε πληκτρολογήσει , αλλά επειδή είναι συμβολοσειρά αυτό που διαβάζεται από ένα jTextField μετατρέπουμε την συμβολοσειρά σε Int. Μετά την μετατροπή καλούμε το ανάλογο query που μεταφέρεται στην βάση μας και από εκεί παίρνουμε τα δεδομένα που θέλουμε.

```
int result = Integer.parseInt(jTextField1.getText());

String query="SELECT * FROM products where Quantity<="+result;
```

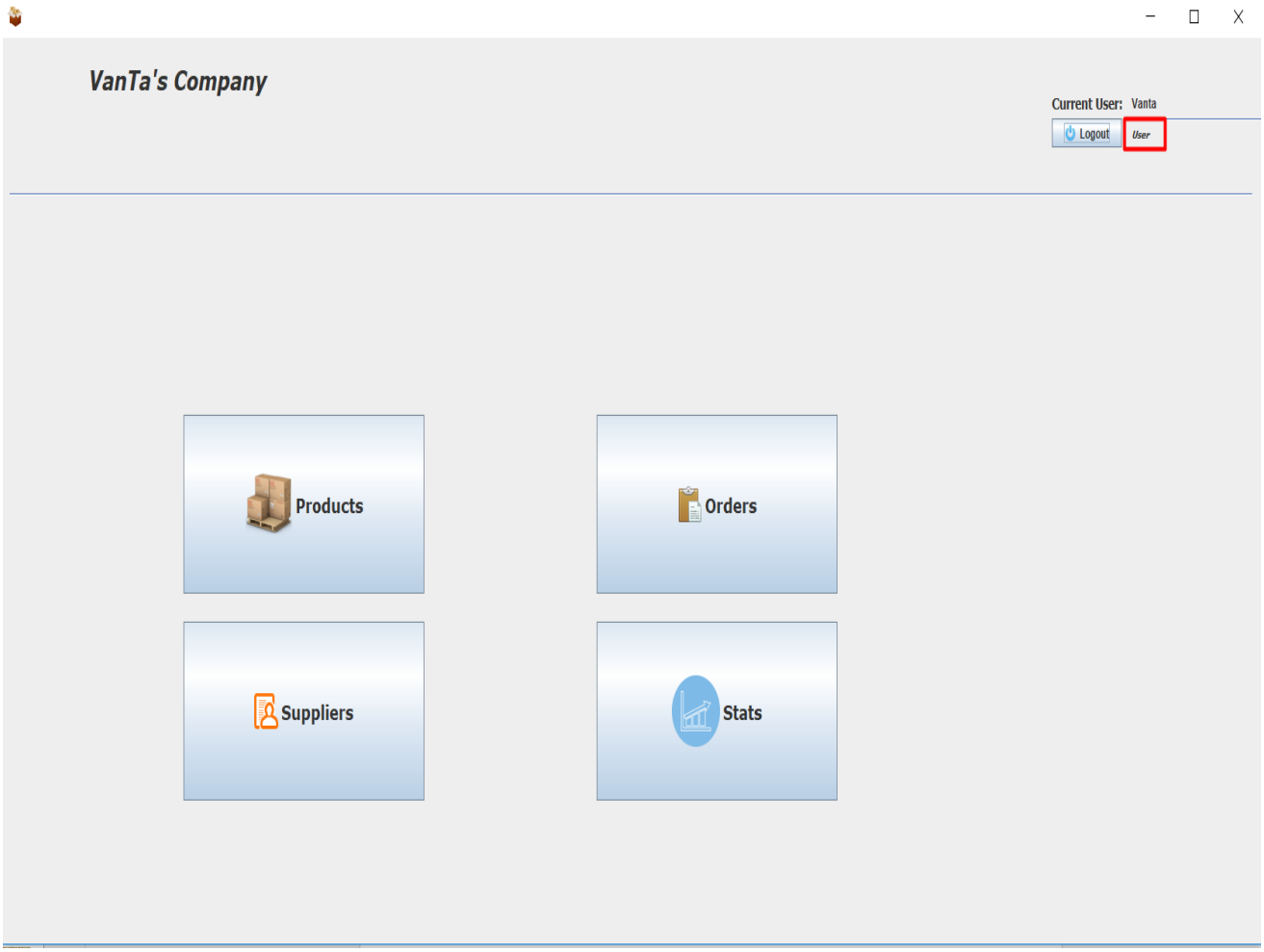
Εικόνα 5.4-4: Java code for stats

5.5)Ανάλυση Admin Panel.

Στην αρχή είχαμε μιλήσει για της δύο ομάδες υπαλλήλων που υπάρχουν στην αποθήκη έτσι ώστε να υπάρχει μία σωστή διαχείριση. Είναι ο αποθηκάριος και ο διαχειριστής. Ο αποθηκάριος ασχολείται κυρίως με τα προϊόντα . Τα προϊόντα θα πρέπει να είναι τοποθετημένα στην κατάλληλη θέση έτσι ώστε ο κάθε αποθηκάριος να πηγαίνει αυτόματα στο κατάλληλο τμήμα της αποθήκης και να το βρίσκει. Από την άλλη ο διαχειριστής «προσέχει» της ενέργειες που πράττουν οι αποθηκάριοι στο πρόγραμμα . Εάν ένα προϊόν καταχωρηθεί σε λάθος τοποθεσία , τότε ο διαχειριστής θα ξέρει ποιος έκανε το λάθος, και να προβεί στης κατάλληλες ενέργειες.

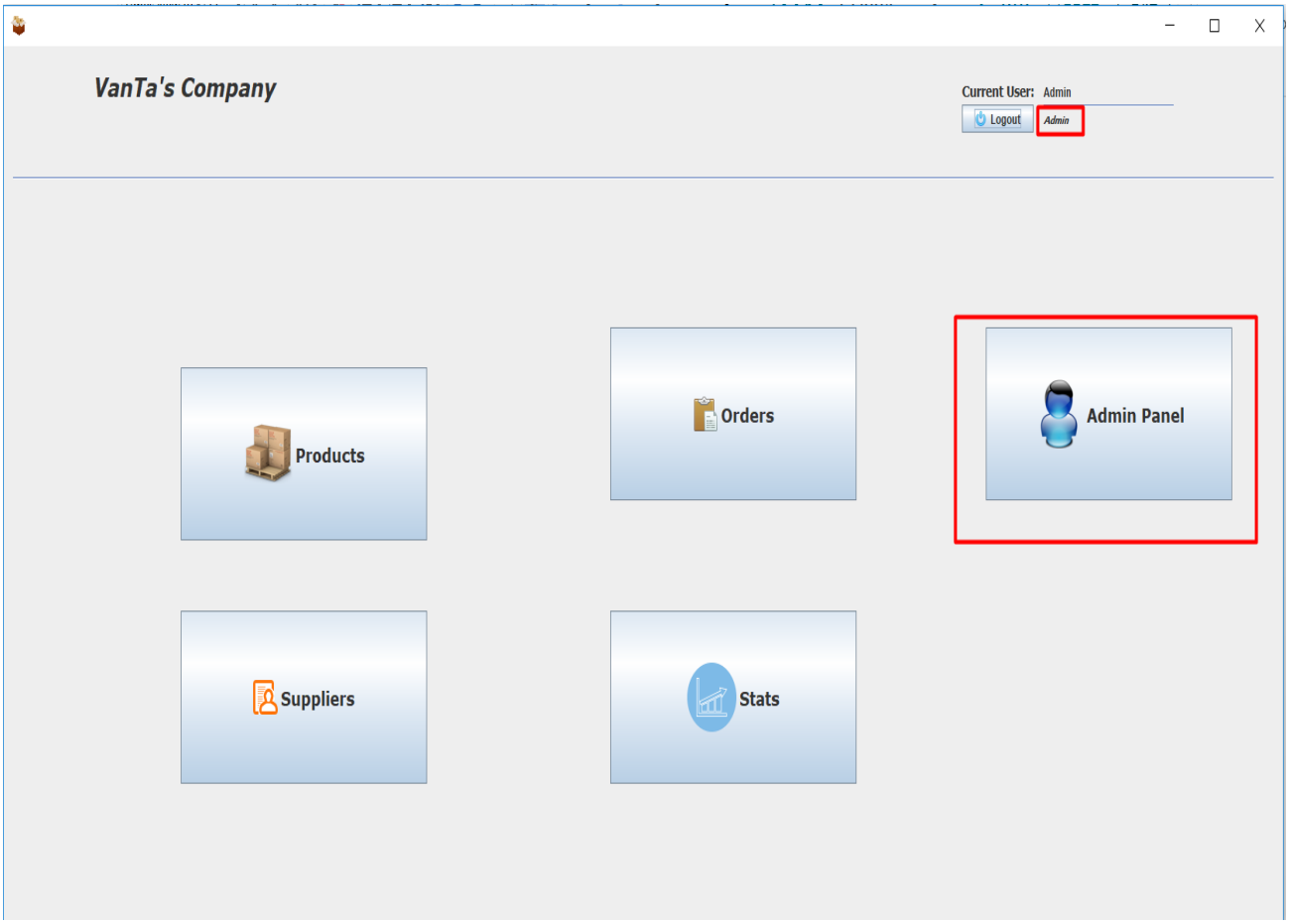
Έτσι και το πρόγραμμα ‘αναγνωρίζει’ την θέση που έχουν οι χρήστες και τους εμφανίζεται το κατάλληλο ‘Control Panel’. Αυτό ελέγχεται κάθε φορά που εισέρχεται κάποιος υπάλληλος στο πρόγραμμα.

Για απλούς χρήστες το παράθυρο που βγαίνει είναι:



Εικόνα 5.5-1: User panel

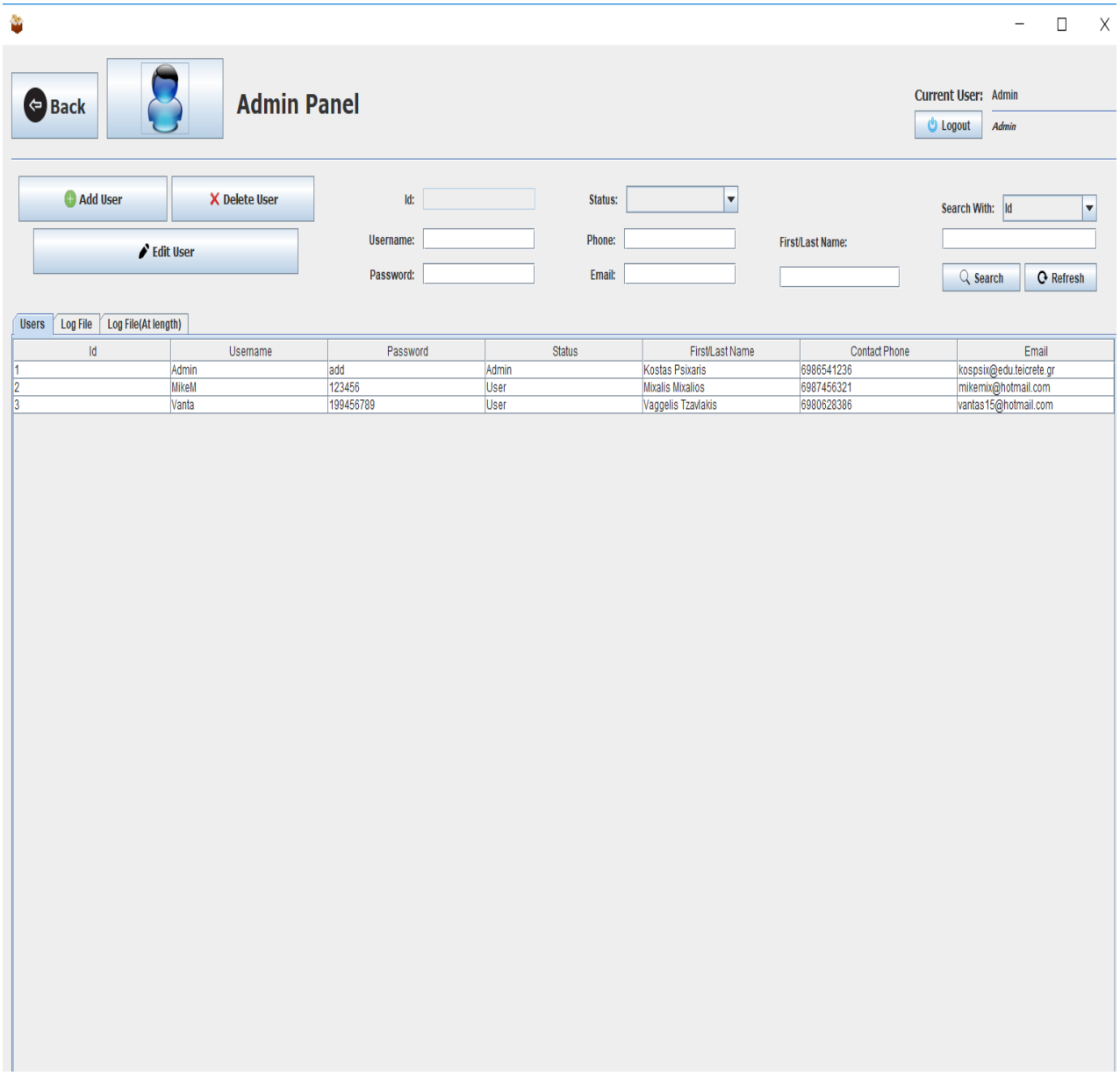
Για τους διαχειριστές το παράθυρο που βγαίνει είναι:



Εικόνα 5.5-2: Admin panel

Όπως παρατηρείτε στο κεντρικό παράθυρο όπου έχει εισέλθει ο διαχειριστής έχει προστεθεί άλλο ένα κουμπί που εμφανίζεται καθαρά μόνο στους διαχειριστές του προγράμματος.

Με το που πατηθεί αυτό το κουμπί εμφανίζεται το παρακάτω παράθυρο:



Εικόνα 5.5-3: Admin panel

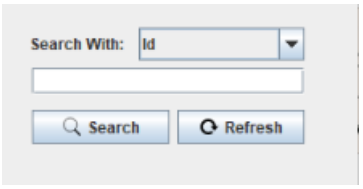
5.5.1)Επιπλέον δυνατότητες

Σε αυτό το παράθυρο , καθώς έχει πρόσβαση όποιος είναι διαχειριστής της αποθήκης, έχουμε της εξής δυνατότητες:

a) **Search.**
b) **Edit User.**
c) **Add User.**
d) **Delete User.**
e) **Refresh.**
f) **Logfile.**

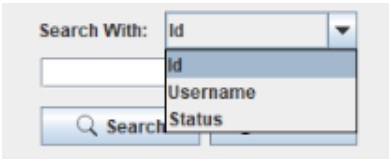
a) Search.

Για να μπορεί να προηγηθεί εύκολα και γρήγορα ο υπάλληλος έχει προστεθεί η δυνατότητα αναζήτησης , όπου μπορεί εύκολα να αναζητήσει τους υπαλλήλους που θέλει να μάθει διάφορες πληροφορίες



Εικόνα 5.5.1-1: Admin panel- Search

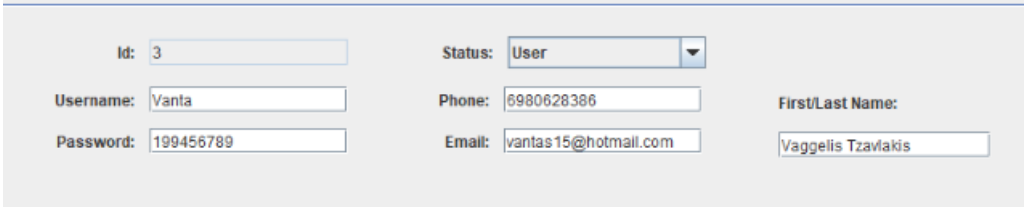
Για να μπορεί να αναζητηθεί ένας υπάλληλος , μπορεί να βάλει στοιχεία , όπως Id, Username ή ακόμα να εμφανίσει όλους τους χρήστες βάζοντας στο Search with, Users.



Εικόνα 5.5.1-2: Admin panel- Search values

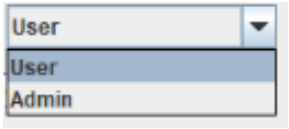
b) Edit User.

Υπάρχει και η δυνατότητα να αλλάξει κάποιες από της τιμές ενός υπαλλήλου. Το Id δεν μπορεί να τροποποιηθεί καθώς είναι μοναδικό σε κάθε έναν υπάλληλο.



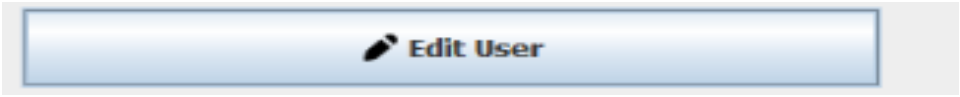
Εικόνα 5.5.1-3: Admin panel- Edit User

Επίσης για την διευκόλυνση του διαχειριστή έχει προστεθεί και μια μπάρα επιλογών . Με αποτέλεσμα να ελαχιστοποιήσουμε της πιθανότητες ορθογραφικού λάθος. Πράγμα που μας βοηθάει για την σωστή λειτουργία του προγράμματος. Έχει δύο τιμές, User, Admin.



Εικόνα 5.5.1-4: User status

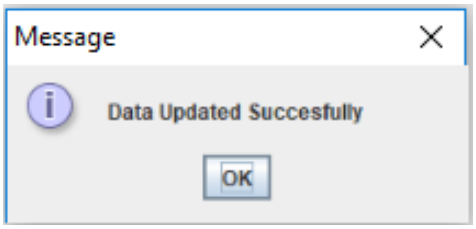
Όταν ο διαχειριστής τελειώσει την αλλαγή πρέπει να πατήσει το κουμπί edit user για να καταχωρηθεί επιτυχώς στην βάση δεδομένων.



Εικόνα 5.5.1-5: Admin panel-Edit user

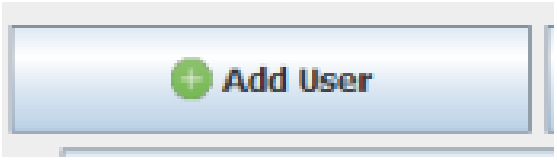
Οι ενέργειες που πρέπει να προβούν για να τροποποιηθούν κάποια στοιχεία από έναν υπάλληλο είναι να κάνει κλικ ,ο διαχειριστής , τον υπάλληλο που θέλει να αλλάξει της πληροφορίες του. Αμέσως μετά τα στοιχεία του θα συμπληρωθούν πάνω στα κενά και θα είναι έτοιμα προς επεξεργασία..

Για την επιτυχής αλλαγή των δεδομένων έχει προστεθεί ένα Popup που μας σιγουρεύει για το αν έγινε σωστά η αλλαγή στην βάση δεδομένων.



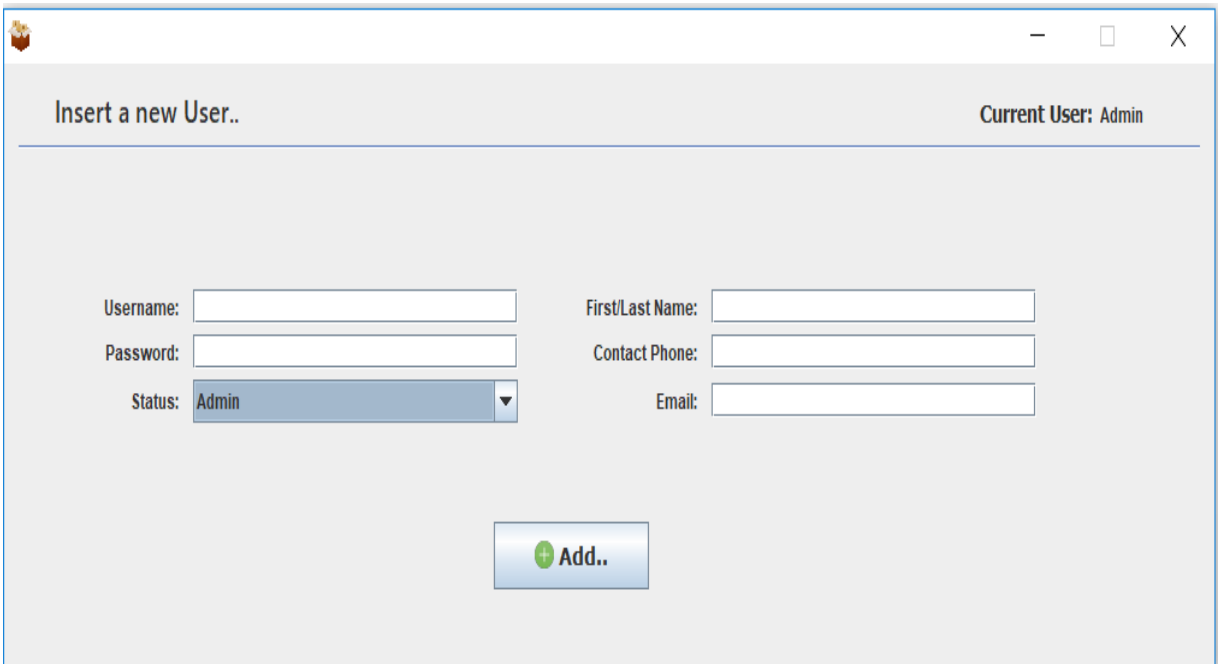
Εικόνα 5.5.1-6: Admin panel- Message Confirmation

c) Add User.



Εικόνα 5.5.1-7: Admin panel- Add User

Με το κουμπί αυτό δίνεται η δυνατότητα εισαγωγής νέου χρήστη. Πατώντας αυτό το κουμπί του ανοίγει ένα άλλο παράθυρο που έτσι μπορεί να καταχωρήσει τα στοιχεία του αποθηκάριου.



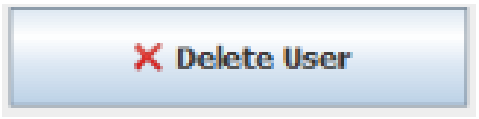
Εικόνα 5.5.1-8: Admin panel- Add User

Αφού έχουν καταχωρηθεί όλα τα στοιχεία του αποθηκάριου μπορεί να επιλέξει με την βοήθεια της μπάρας την ιδιότητα του εργαζομένου.

Εικόνα 5.5.1-9: Admin panel- Add User

Συμπληρώνοντας τα κατάλληλα στοιχεία και επιλέγοντας τον βαθμό που θα έχει στην αποθήκη (Admin/User) πατώντας το παραπάνω κουμπί (Add..) οι τιμές καταχωρούνται επιτυχώς στην βάση δεδομένων.

d) Delete User.

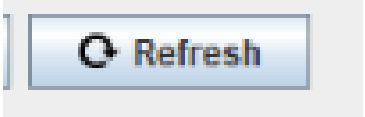


Εικόνα 5.5.1-10: Admin panel- delete User

Επιλέγοντας τον χρήστη που θέλουμε να αφαιρέσουμε από την βάση δεδομένων και π0ατώντας το παραπάνω κουμπί επιτυγχάνουμε την διαγραφή του επιλεγμένου υπαλλήλου από την βάση δεδομένων.

e) Refresh.

Πατώντας αυτό το κουμπί γίνεται μία ανανέωση του πίνακα Users για να μπορεί ο διαχειριστής να επιβλέψει αν έγινε σωστά η καταχώρηση του νέου υπαλλήλου.

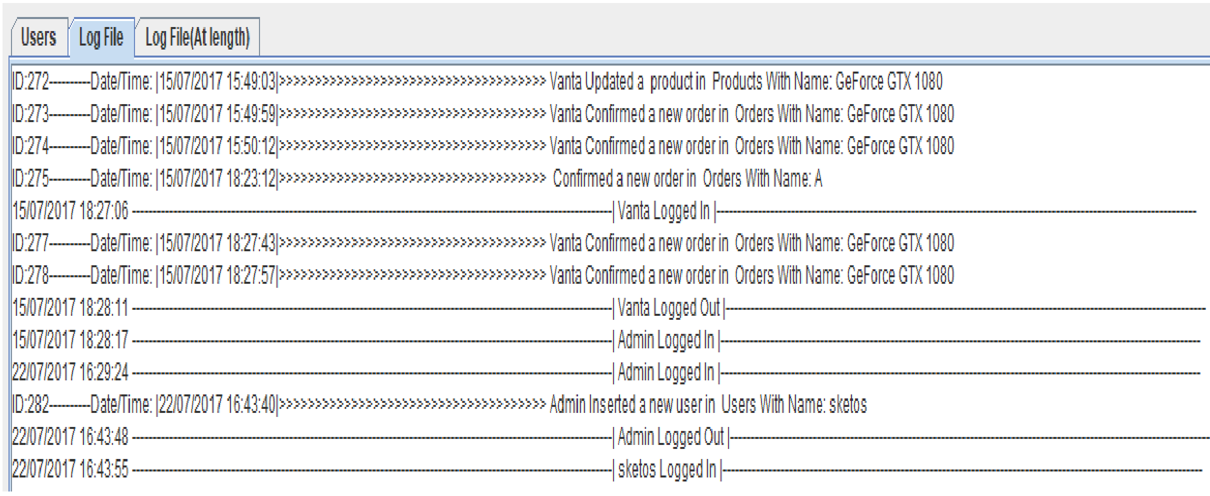


Εικόνα 5.5.1-11: Admin panel- Refresh Table

f) Logfile.

Όπως ανέφερα πιο πάνω ο διαχειριστής θα πρέπει να γνωρίζει ακριβώς της πράξεις που έχουν προβεί οι χρήστες. Έτσι ώστε να αναγνωρίζετε το λάθος και να γίνουν οι ανάλογες παρατηρήσεις στον υπάλληλο για να γίνει καλύτερος.

Στο πρόγραμμα προστέθηκε λοιπόν , που είναι αναγνώσιμες μόνο από τον διαχειριστή του προγράμματος, ένα logfile. Εκεί αναγράφονται τα στοιχεία των κινήσεων των αποθηκάριων. Όπως από το πότε μπήκε , ποιος άλλαξε μία τιμή και από πού και επίσης την τιμή που άλλαξε, πρόσθεσε ή διέγραψε.

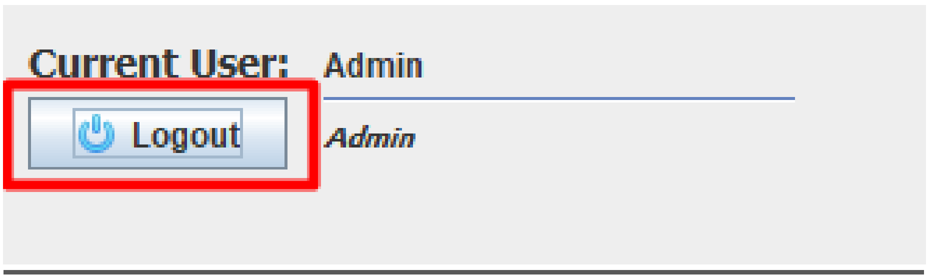


Εικόνα 5.5.1-12: Admin panel- Logfile

Επίσης έχει προστεθεί άλλη μία ιδιότητα όπου ο διαχειριστής μπορεί να δει αναλυτικά την βάση δεδομένων του logfile(Log File(At length)).

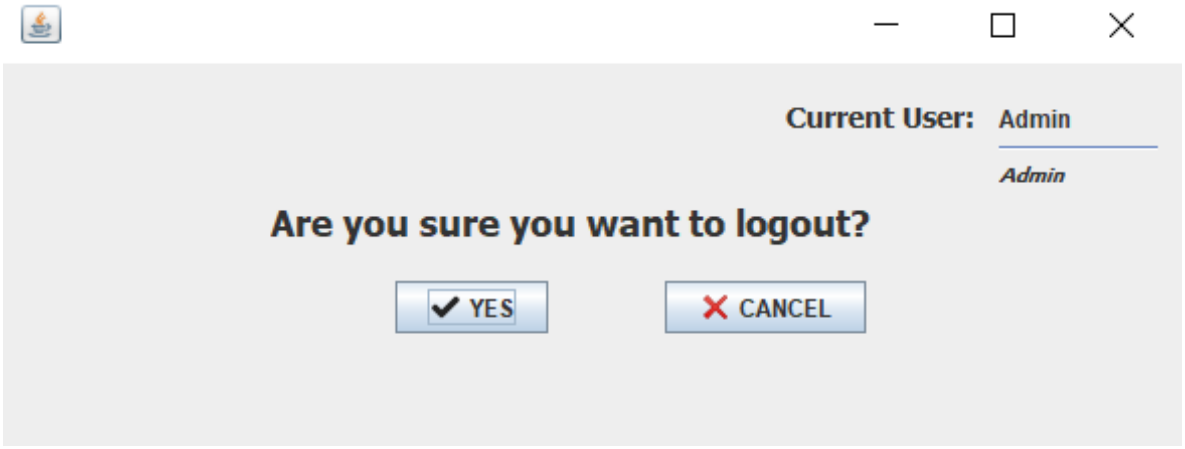
5.6)Logout

Το τελευταίο πράγμα που πρέπει να κάνει ο υπάλληλος μέχρι το τέλος της βάρδιας του είναι να αποσυνδεθεί από το πρόγραμμα. Έτσι προστέθηκε πάνω δεξιά ένα κουμπί όπου μπορεί να αποσυνδεθεί από το πρόγραμμα επιτυχώς.



Εικόνα 5.6-1: Admin panel- Logout

Έτσι μόλις το πατήσει θα του εμφανίσει ένα ερώτημα , αν είναι σίγουρος για αυτήν την επιλογή αν πατήσει ναι θα κατευθυνθεί ξανά στην αρχή του προγράμματος όπου θα πρέπει να ξαναδώσει τα στοιχεία του για να ξαναμπεί στο πρόγραμμα. Αν πατήσει άκυρο τότε συνεχίζει την περιήγηση στο πρόγραμμα.



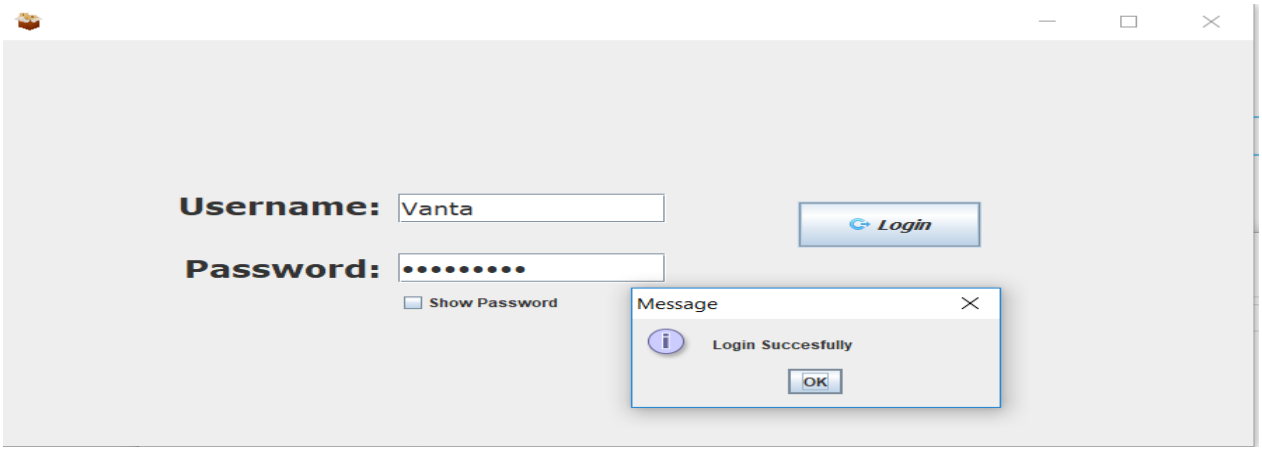
Εικόνα 5.6-2: Logout Answer

6) Αναλυτική Περιγραφή

Σε αυτό το κεφάλαιο θα αναφέρω παραδείγματα από της δυνατότητες που έχουν οι χρήστες στο πρόγραμμα. Σε κάθε κίνηση τους (Εισαγωγή, διαγραφή, τροποποίηση) εκτός από την εισαγωγή/τροποποίηση των πληροφοριών καταχωρείται και κάποιες πληροφορίες που είναι χρήσιμες για έναν διαχειριστή.

6.1)Εισαγωγής καινούριου προϊόντος.

Αρχικά θα πρέπει να εισέλθει μέσα στο πρόγραμμα με το δικό του λογαριασμό(δοσμένο από τον διαχειριστή).

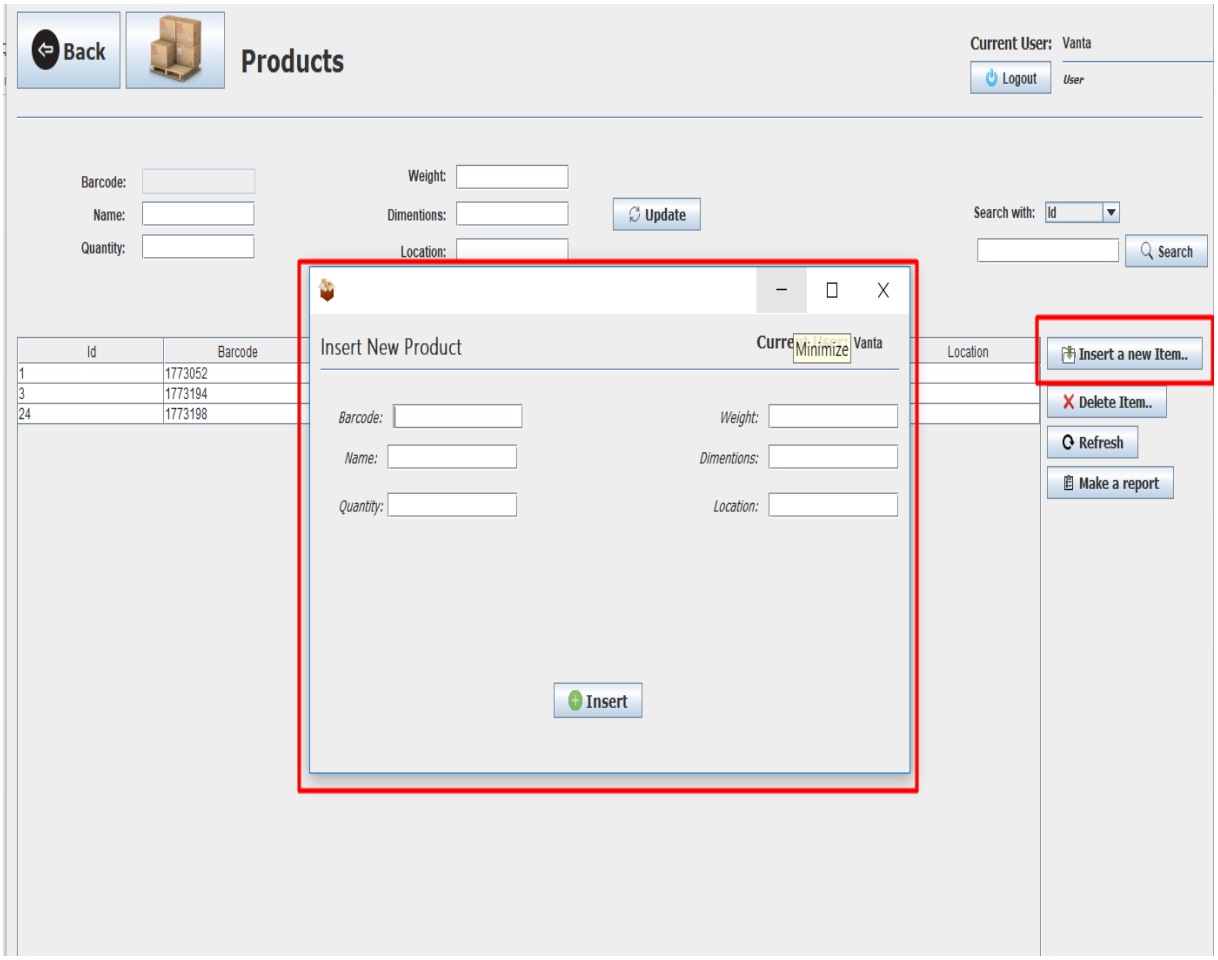


Εικόνα 6.1-1: Login panel- Logout Success

Αφού γίνει επιτυχώς η σύνδεση στο πρόγραμμα θα πρέπει να επιλέξει το Products αφού θα θέλει να εισάγει μία καινούρια τιμή στην βάση δεδομένων. Μπορούμε να παρατηρήσουμε πάνω δεξιά ότι έχει γίνει η επιτυχής σύνδεση στην βάση δεδομένων οπού αναγράφεται η ιδιότητα του υπαλλήλου και το username του. Έτσι μπορούμε να σιγουρευτούμε ότι συνδέθηκε στο δικό του λογαριασμό.

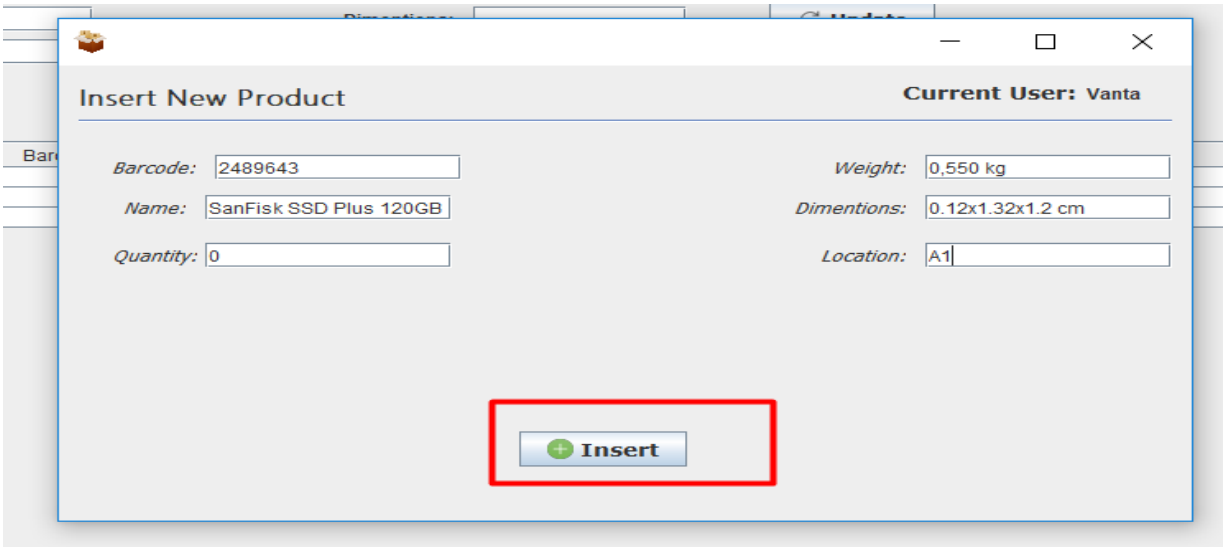
Εικόνα 6.1-2: CompanyJF panel

Αφού πατήσει το Products θα κατευθυνθεί στο παρακάτω παράθυρο, και πατώντας το Insert a new Item.. θα του εμφανίσει άλλο ένα παράθυρο όπου θα το ζητήσει τα στοιχεία του καινούριου προϊόντος.



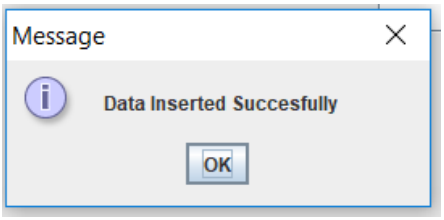
Εικόνα 6.1-3: Products- Insert Panel

Αφού πληκτρολογήσουμε ακριβώς τα στοιχεία του προϊόντος πατάμε Insert.



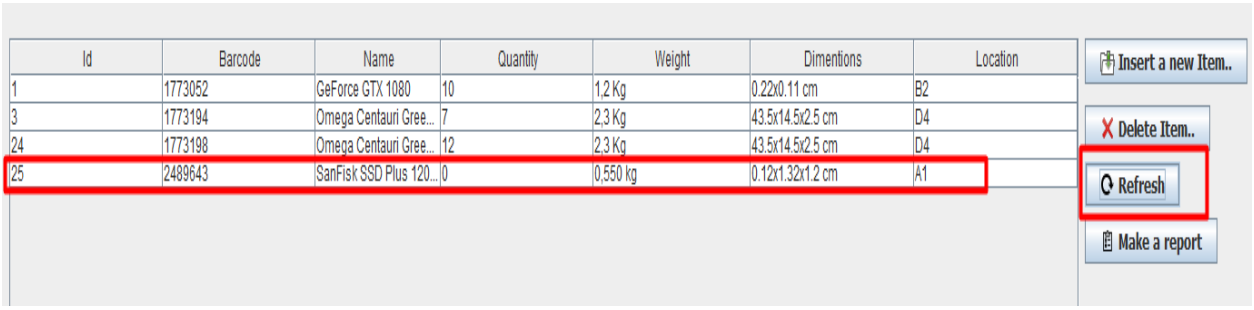
Εικόνα 6.1-4: Products- Insert

Αφού βεβαιωθούμε με το μήνυμα που μας ειδοποιεί ότι έγινε η επιτυχής καταχωρήσει των δεδομένων στην βάση.



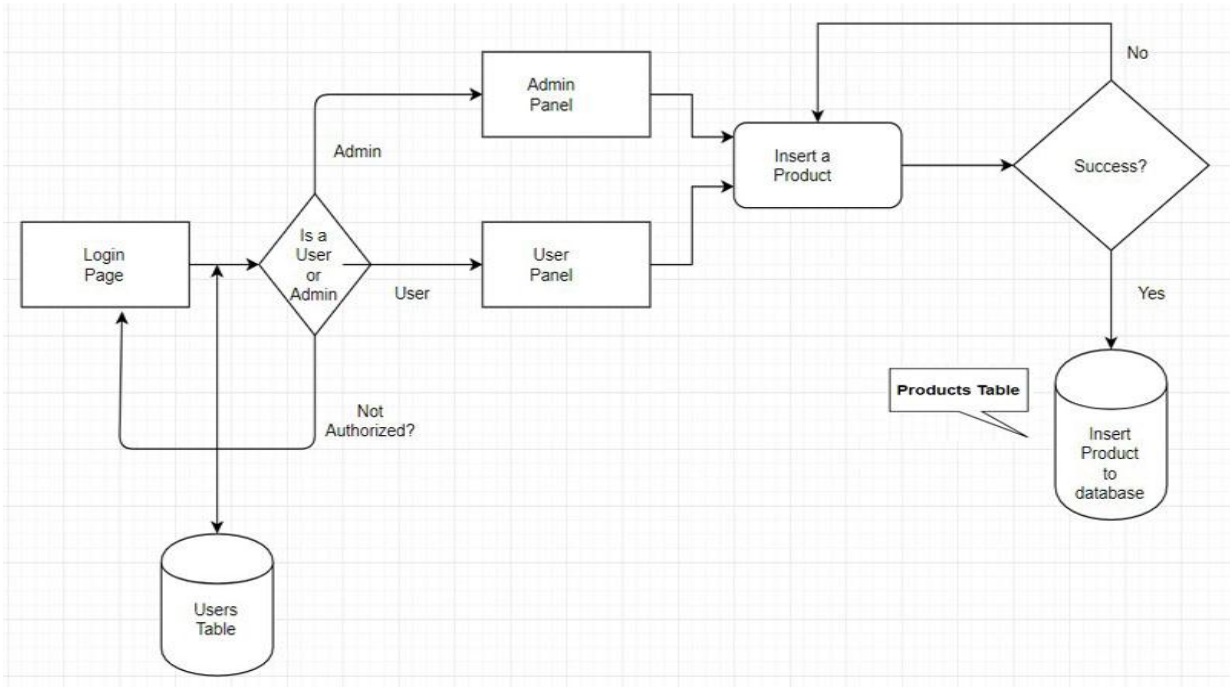
Εικόνα 6.1-5: Products-Data inserted Succesfully

Θα πρέπει να βεβαιωθούμε και από τον πίνακα ότι το προϊόν που προσθέσαμε έχει προστεθεί και στην βάση δεδομένων μας. Πατώντας ένα Refresh έτσι ώστε να ανανεωθεί ο πίνακας τον προϊόντων μας βλέπουμε ότι έχει εμφανίσει άλλη μία τιμή στον πίνακα μας.



Εικόνα 6.1-6: Products-Refresh

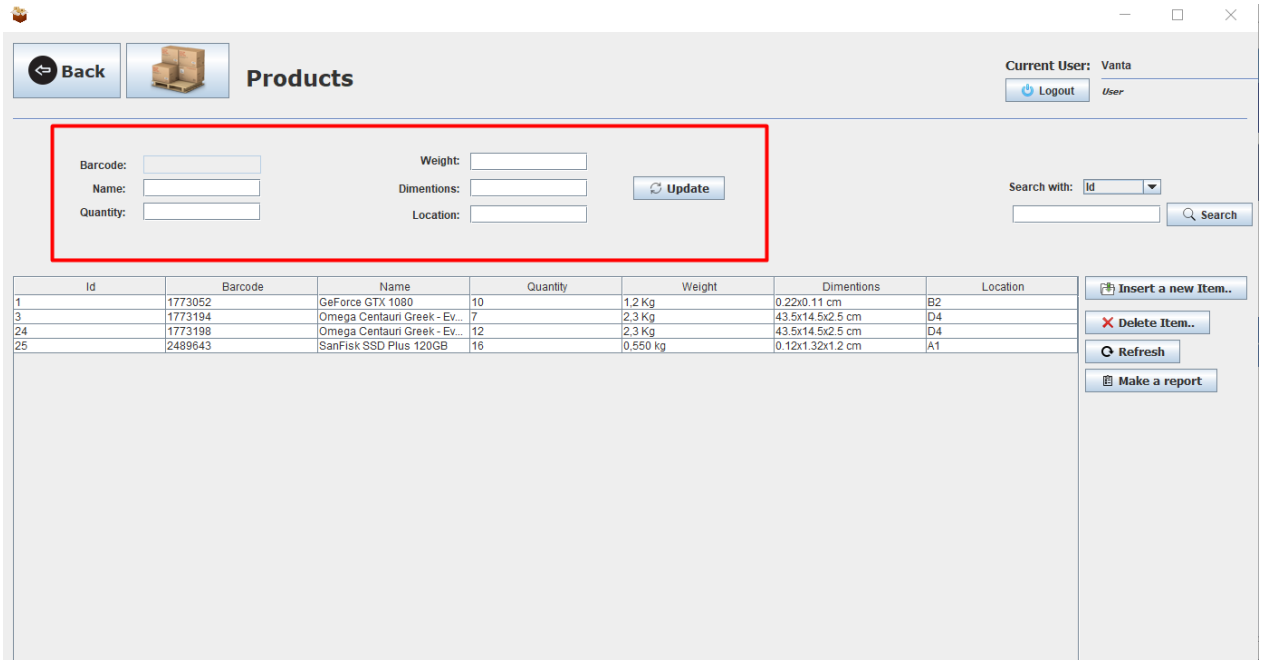
Παρακάτω παραθέτω ένα διάγραμμα ροής όπου επεξηγεί ακριβώς την διαδικασία εισαγωγής ενός προϊόντος στην βάση δεδομένων. Σχεδιασμένο από το draw.io [8].



Εικόνα 6.1-7: Products Scheme

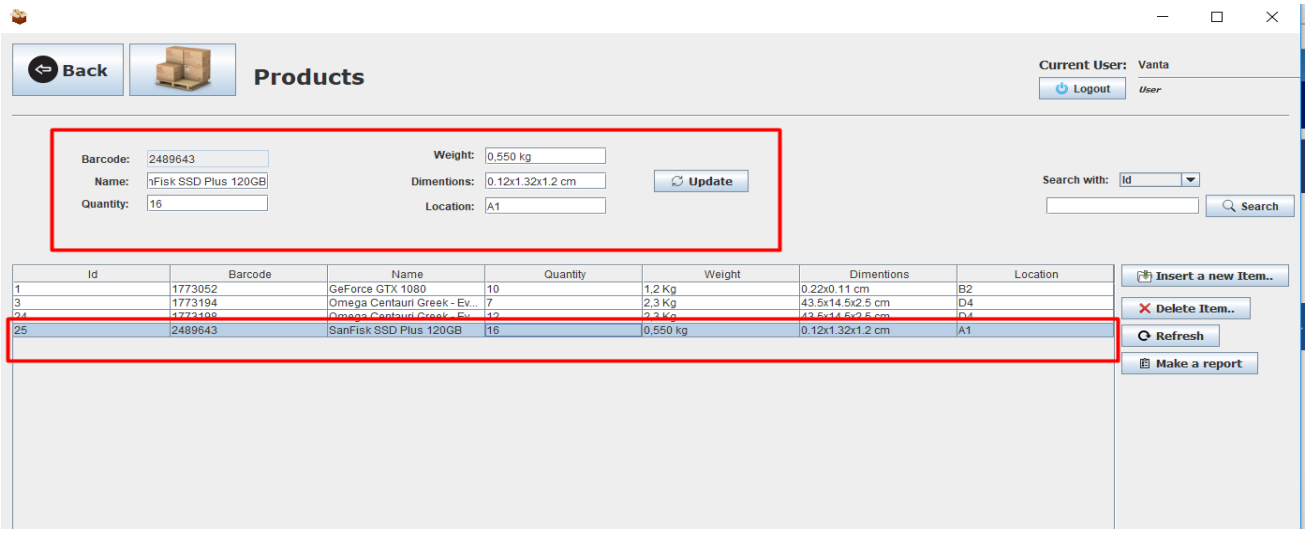
6.2)Ανανέωση πληροφοριών προϊόντων.

Όπως αναφέραμε στην ανάλυση του προγράμματος, ο αποθηκάριος έχει την δυνατότητα να τροποποιήσει κάποιες από της πληροφορίες των προϊόντων. Γιαυτό και έχουν προστεθεί πάνω από τον πίνακα, διάφορα jTextField όπου εκεί θα μπορούμε να βάλουμε τα καινούρια δεδομένα των προϊόντων και να τα καταχωρούμε στην βάση δεδομένων απευθείας.



Εικόνα 6.2-1: Product Update

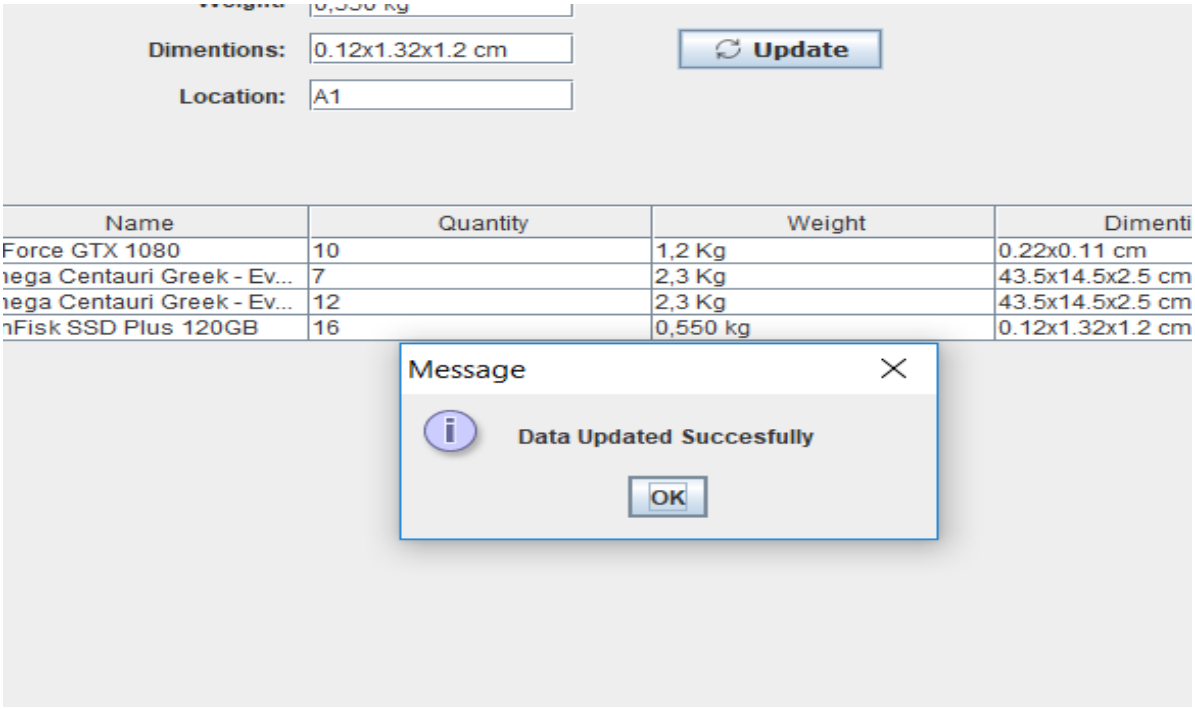
Όταν βρούμε το ανάλογο προϊόν που θέλουμε να τροποποιήσουμε πρέπει να πατήσουμε πάνω του στο πίνακα έτσι ώστε να περάσουν όλες η πληροφορίες στα jTextFeild.



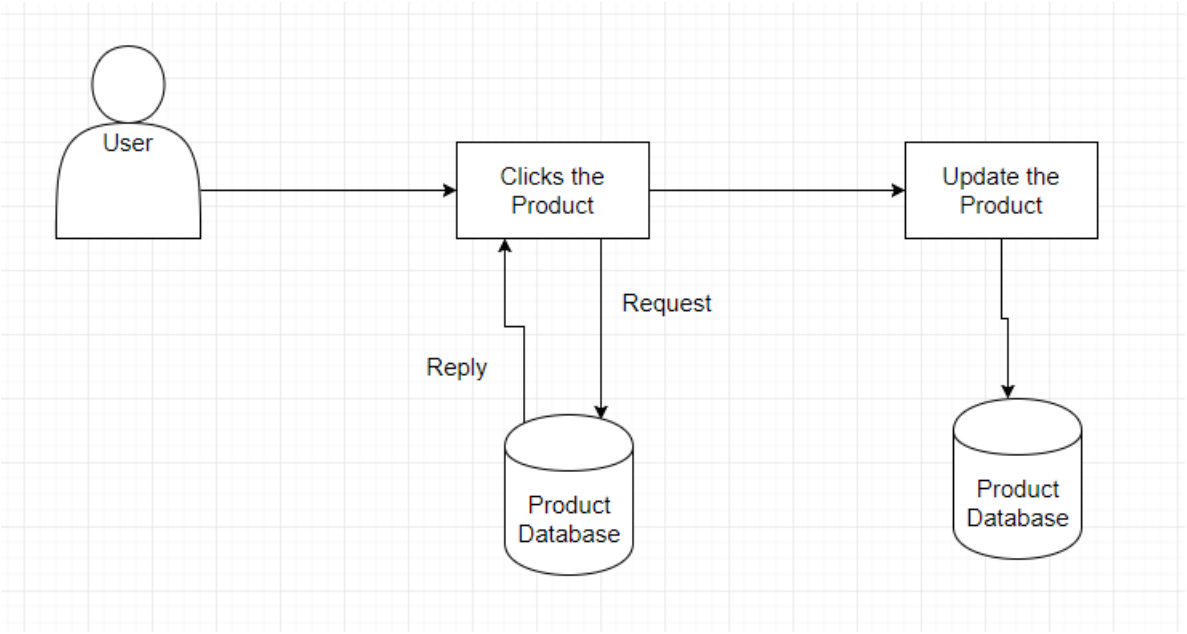
Εικόνα 6.2-2: Product Table

Έτσι αυτομάτως περνάνε όλα τα δεδομένα του συγκεκριμένου προϊόντος , δεν χρειάζεται να τα πληκτρολογούμε από την αρχή και με αυτό τον τρόπο καταφέρνουμε την αποφυγή λάθους που μπορεί να είναι μοιραία στην πορεία εξέλιξης της εταιρίας.

Όταν γίνει η ανανέωση των δεδομένων και πατηθεί το κουμπί Update το πρόγραμμα μας ενημερώνει ότι έγινε επιτυχώς η αλλαγή του προϊόντος και καταχωρήθηκε επιτυχώς στην βάση δεδομένων. Επίσης για διαγραφή ενός προϊόντος γίνονται ακριβώς οι ίδιες ενέργειες.



Εικόνα 6.2-3: Product- Table Update Success

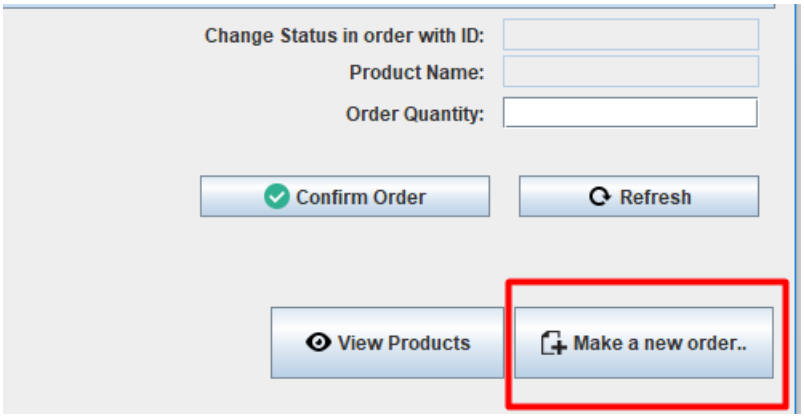


Εικόνα 6.2-4: Product Update Scheme

6.3)Παραγγελία προϊόντων.

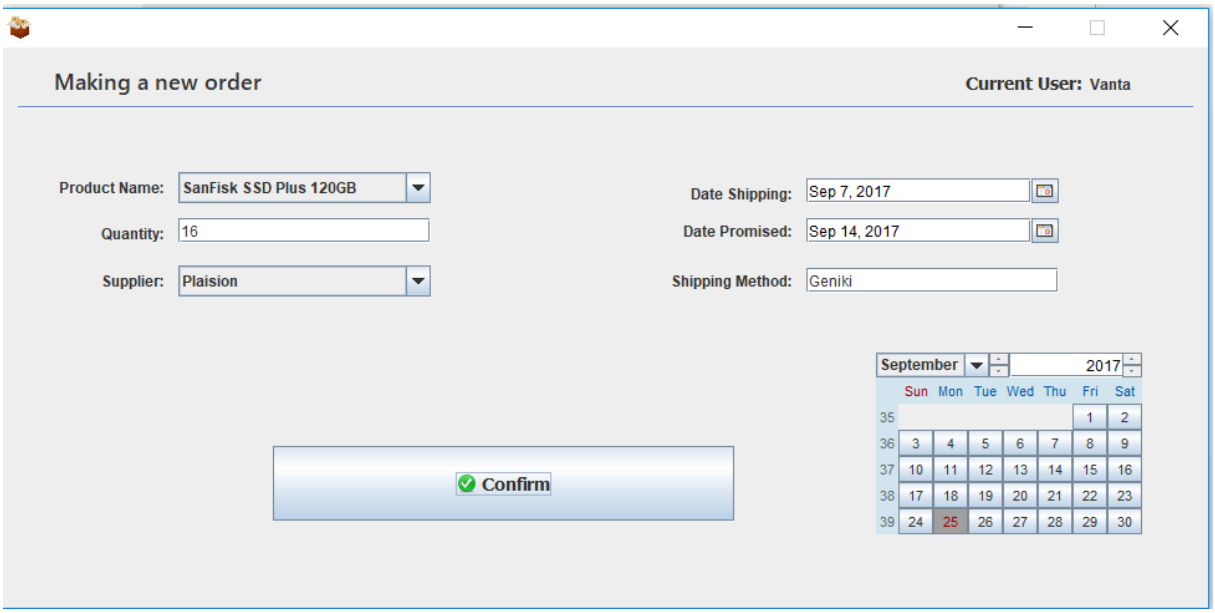
Αφού έχει γίνει η επιτυχής καταχώρηση του προϊόντος στην βάση δεδομένων θα πρέπει να παραγγείλει . Έτσι θα κατευθυνθεί στα Orders για να επιλέξει την επιθυμητή ποσότητα για να εξοπλίσει την αποθήκη με το καινούριο προϊόν.

Άρα θα πατήσει το Make new order για να κάνει μία καινούρια παραγγελία για την αποθήκη.



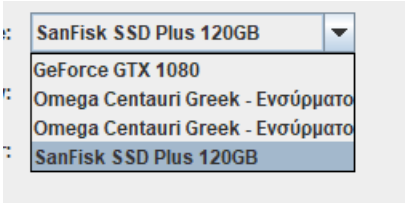
Εικόνα 6.3-1: New Order

Αφού πατηθεί αυτό το κουμπί θα ανοίξει ένα παράθυρο έτσι ώστε να εισάγει της πληροφορίες που χρειάζεται έτσι ώστε να γίνει επιτυχής η παραγγελία . Όπως η ποσότητα που θέλει να παραγγείλει , από ποιόν θα το πάρει κ.α. Πατώντας το Insert θα ειδοποιηθεί με ένα μήνυμα ,αν έγινε καταχώρηση στην βάση δεδομένων η όχι.

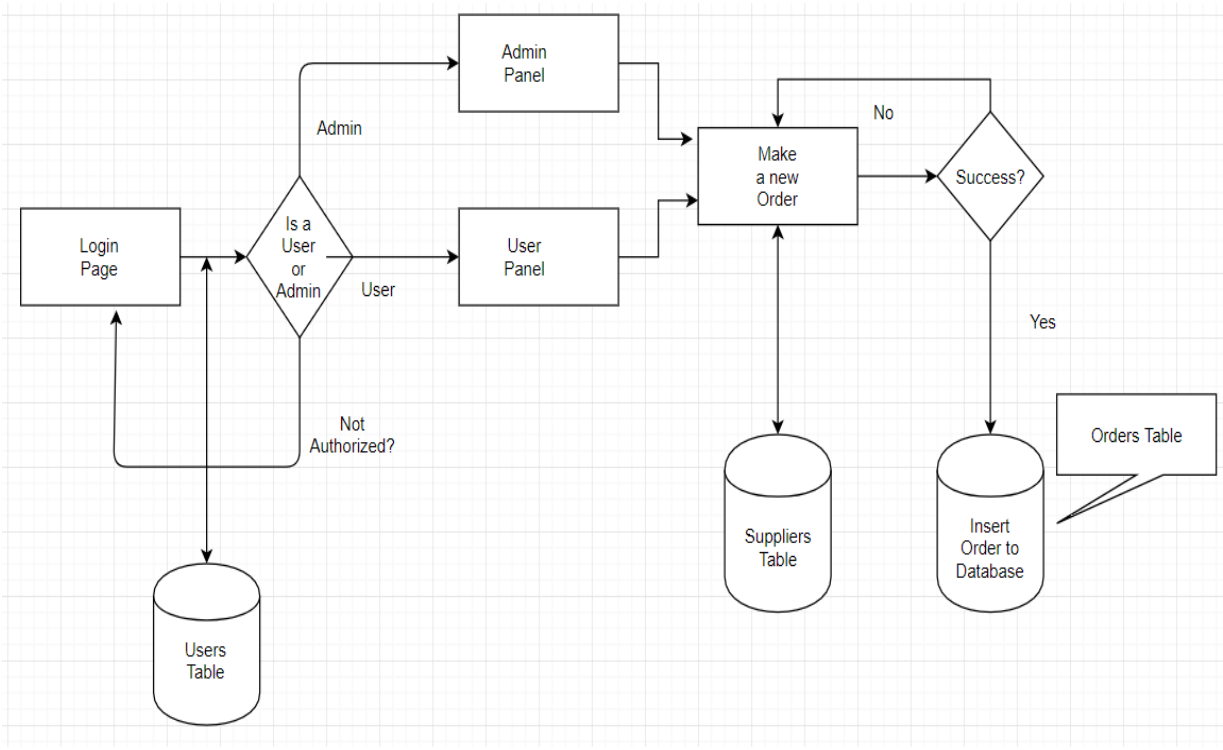


Εικόνα 6.3-2: New Order Panel

Αν παρατηρήσουμε την μπάρα με τα Product Name έχει εισαχθεί και η καινούρια τιμή που βάλαμε πιο πριν χωρίς να προβούμε σε άλλη ενέργεια.



Εικόνα 6.3-3: ComboBox



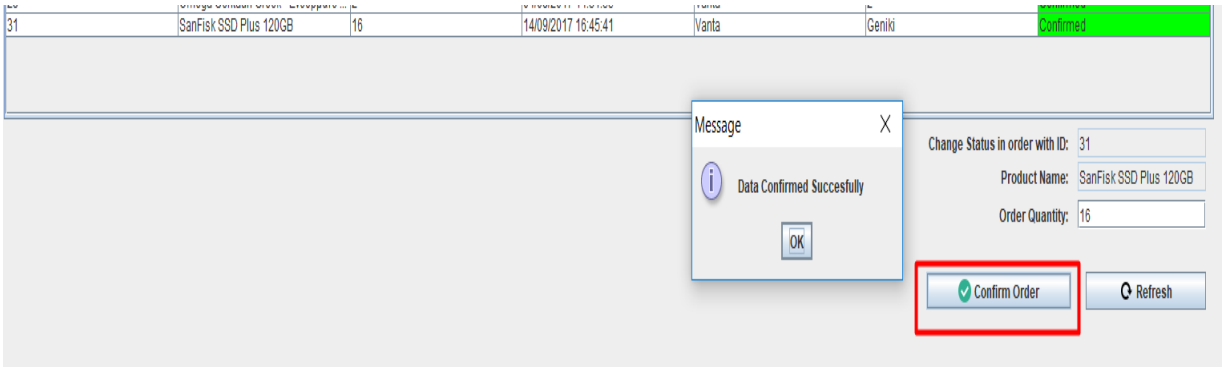
Εικόνα 6.3-4: New Order Scheme

Αν παρατηρήσουμε τώρα στο History θα δούμε ότι έχει προστεθεί μία καινούρια παραγγελία που είναι Pending.. (Αφού έχουμε πατήσει Refresh για να ανανεώσουμε των πίνακα μας με την βάση δεδομένων). Αυτό σημαίνει ότι έχει γίνει επιτυχώς η παραγγελία, μας έχουν ενημερώσει για την πορεία της εξέλιξης της, όταν έρθουν θα πάμε και θα την επικυρώσουμε .

28	Omega Centauri Greek - Ενσύρματο ...	2	04/08/2017 14:31:55	Vanta	2	Confirmed
31	SanFisk SSD Plus 120GB	16	14/09/2017 16:45:41	Vanta	Geniki	Pending..

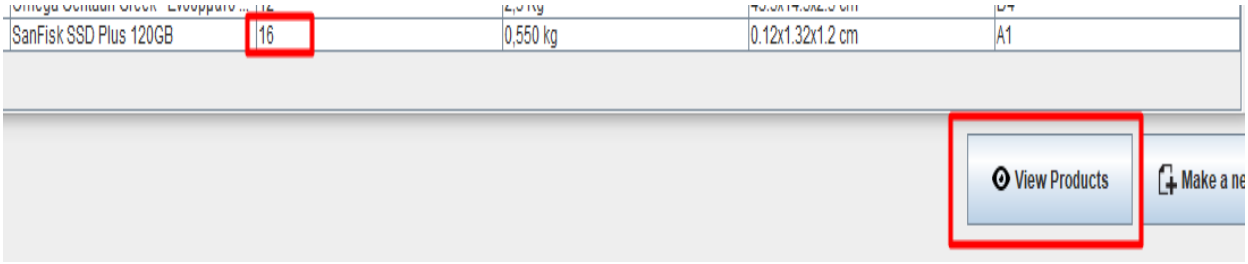
Εικόνα 6.3-5: Confirm Order

Όταν την επικυρώσουμε θα μας εμφανίσει ένα μήνυμα, ότι η παραγγελία μας έχει έρθει επιτυχώς.

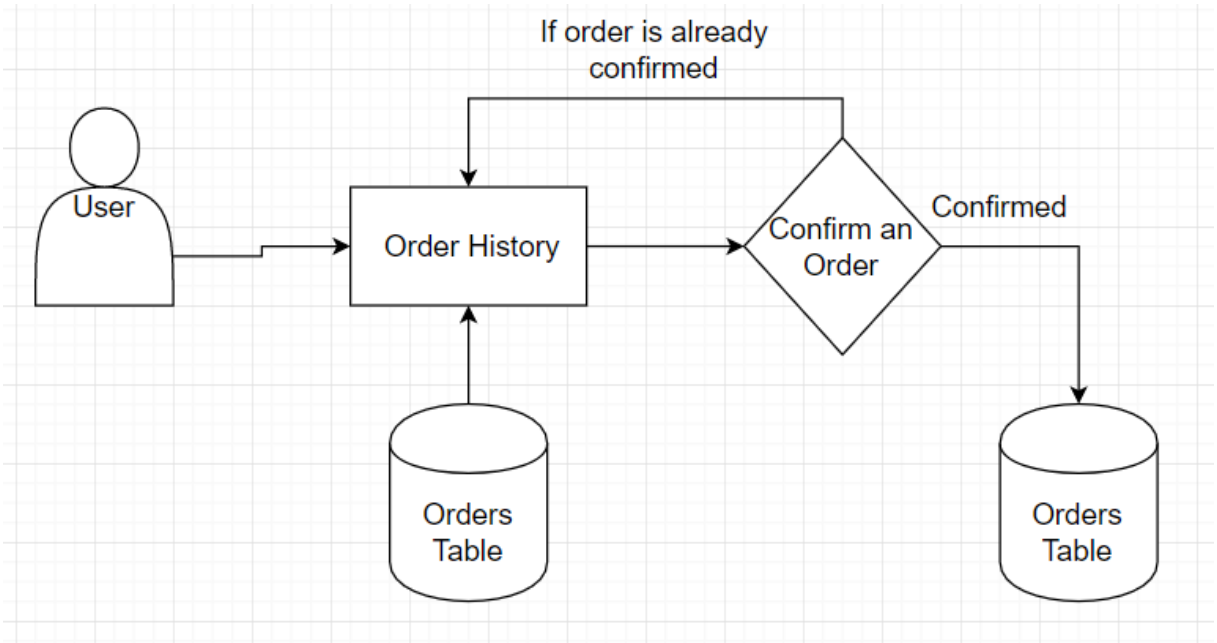


Εικόνα 6.3-6: Success Confirm

Αν πάμε και πατήσουμε το View Products θα παρατηρήσουμε ότι η ποσότητα του προϊόντος που παραγγείλαμε και ήρθε επιτυχώς στην αποθήκη έχει αυξηθεί κατά την ποσότητα που είχαμε παραγγείλει.



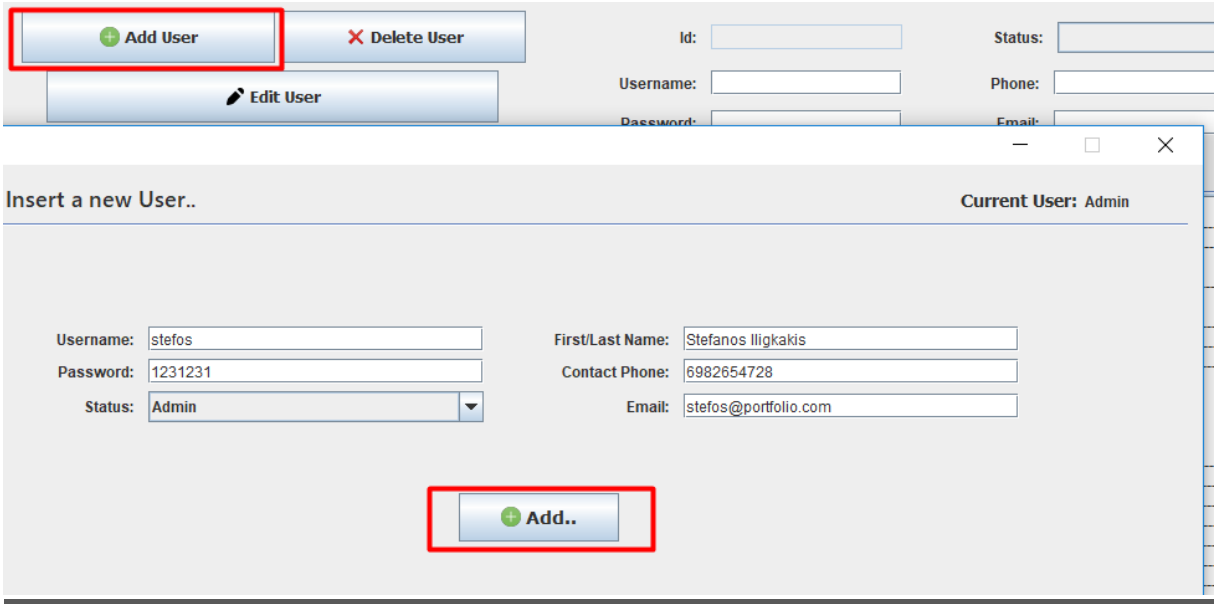
Εικόνα 6.3-7: View Products



Εικόνα 6.3-8: Making new Order Scheme

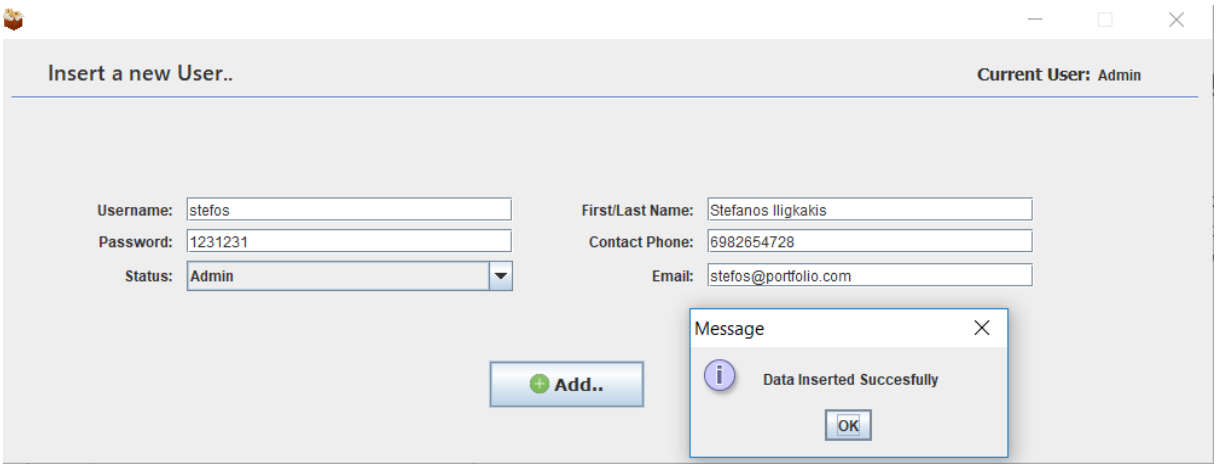
6.4)Αξιολόγηση διαχειριστή και εισαγωγή καινούριου χρήστη.

6.4.1)Εισαγωγή καινούριου χρήστη.



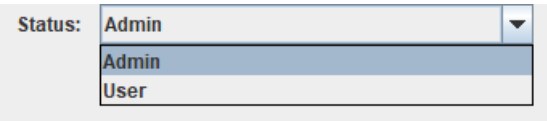
Εικόνα 6.4.1-1: Adding a User

Για εισαγωγή ενός χρήστη ο διαχειριστής πρέπει να πατήσει add user. Θα ανοίξει ένα παράθυρο όπου εκεί θα μπορεί ο διαχειριστής να εισάγει έναν καινούριο χρήστη. Πληκτρολογώντας τα κατάλληλα στοιχεία για τον συγκεκριμένο χρήστη και πατώντας add καταφέρνουμε την καταχώρηση του χρήστη στην βάση δεδομένων.



Εικόνα 6.4.1-2: Data Inserted Succesfully

Εδώ μπορεί να επιλέξει και την ιδιότητα του εργαζόμενου , για την διευκόλυνση μας έχω προσθέσει μία μπάρα με μόνο δύο τιμές (Admin/User).

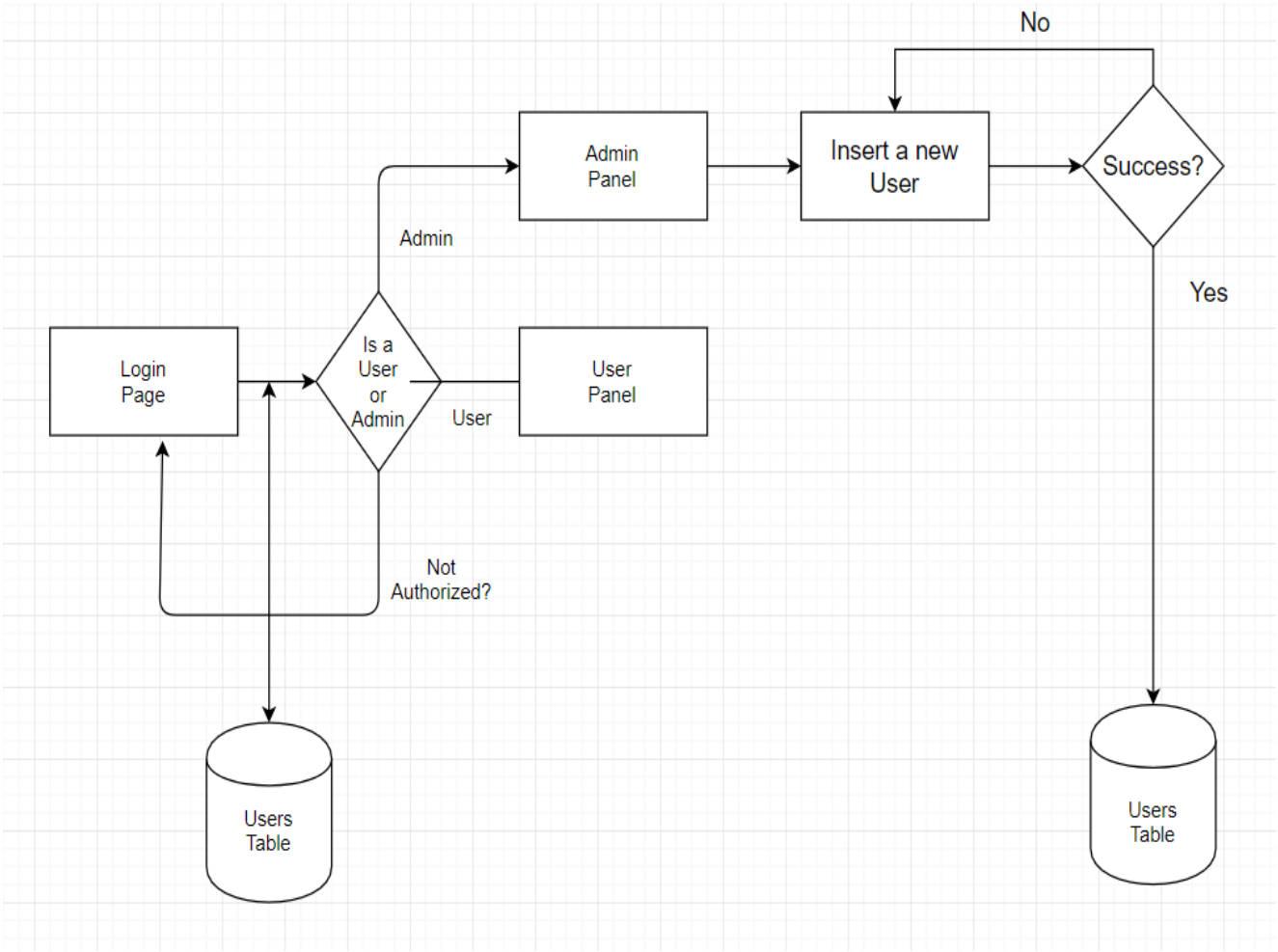


Εικόνα 6.4.1-3: Status Combobox

Έτσι αν πατήσουμε refresh θα ανανεώσουμε τον πίνακα μας με την βάση δεδομένων .

Id	Username	Password	Status	First/Last Name	Contact Phone	Email
1	Admin	add	Admin	Kostas Psixaris	6986541236	kospsix@edu.teicrete.gr
2	MikeM	123456	User	Mixalis Mixalios	6987456321	mikemix@hotmail.com
3	Vanta	199456789	User	Vaggelis Tzavlakis	6980628386	vantas15@hotmail.com
4	stefos	1231231	Admin	Stefanos Iligkakis	6982654728	stefos@portfolio.com

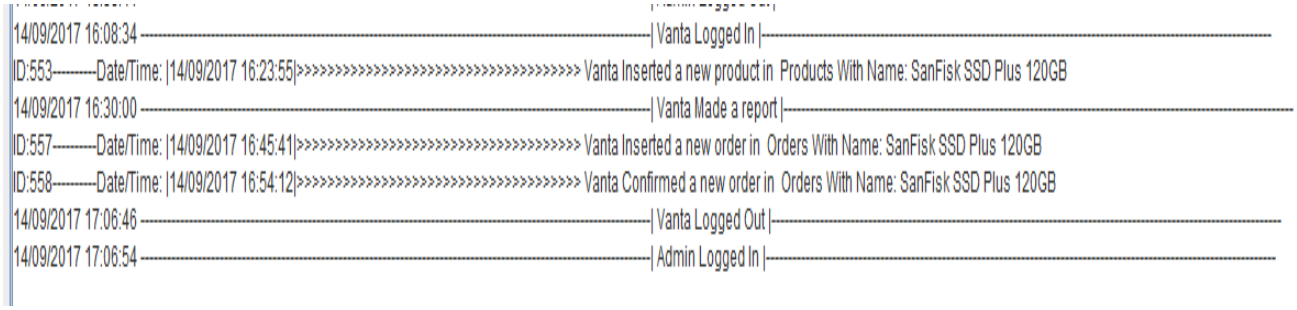
Εικόνα 6.4.1-4: User Table



Εικόνα 6.4.1-5: Adding a User Scheme

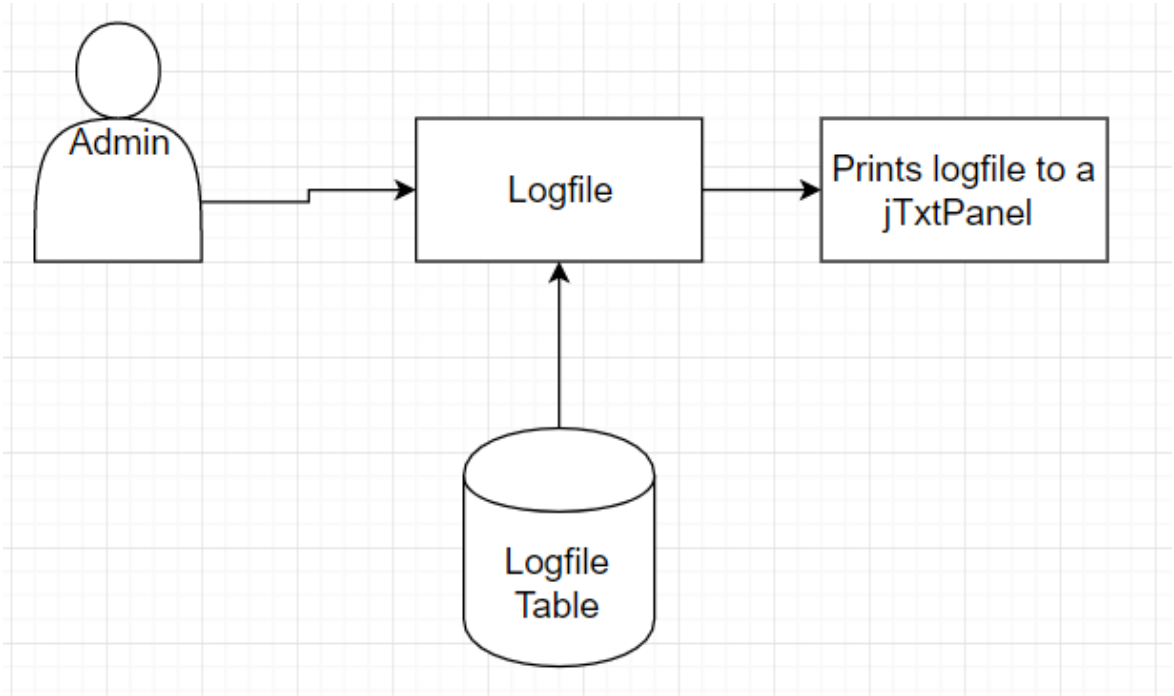
6.4.2)Αξιολόγηση διαχειριστή

Όλες αυτές οι αλλαγές που έγιναν στα παραδείγματα , ο διαχειριστής μπορεί να της δει όλες στο logfile έτσι ώστε να μπορεί να αξιολογήσει τους εργαζόμενους του.



Εικόνα 6.4.2-1: Adding a User

Για να είναι μια αποθήκη υγιείς θα πρέπει να μοιραστούν αρμοδιότητες σε κάθε εργαζόμενους. Ο αποθηκάριος είναι υπεύθυνος για την ταξινόμηση και την παραγγελία των προϊόντων και ο διαχειριστής είναι υπεύθυνος για να ελέγχει τους αποθηκάριους έτσι ώστε να εργάζονται σωστά και με υπευθυνότητα. Γι'αυτό έχει μπει και το logfile, έτσι ώστε να κάνει την 'ζωή' του διαχειριστή πιο εύκολή .



Εικόνα 6.4.2-2: Watching logfile Scheme

7)Παραδείγματα Κώδικα

7.1)Η σύνδεση με τη βάση δεδομένων

Παρακάτω παρουσιάζουμε ένα απλό μέρος κώδικα που κάνει τη σύνδεση με τη βάση δεδομένων και εκτελεί ένα ερώτημα SQL σε ένα πίνακα της βάσης.

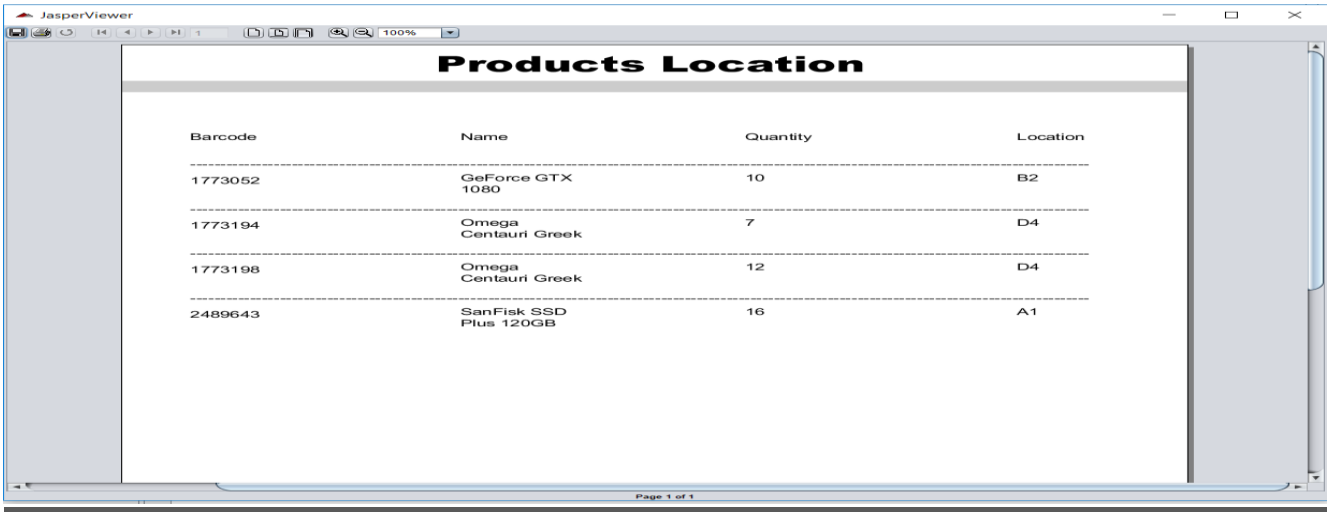
```
Connection con;
Statement st;
public Connection getConnection(){


    try{

         con = DriverManager.getConnection("jdbc:mysql://localhost:3306/company?autoReconnect=true&useSSL=false","root","zoot");
        return con;
    }catch (Exception e){
    e.printStackTrace();
    return null;
    }
}
```

Εικόνα 7.1-1: Logging to Database

Αυτό το κομμάτι του κώδικα καλείται κάθε φορά που θέλουμε να διαβάσουμε/τροποποιήσουμε κάτι από την βάση δεδομένων. Με αυτόν τον τρόπο, με το σωστό ερώτημα που θα κάνουμε στην βάση δεδομένων μας, εμφανίζουμε τα επιθυμητά αποτελέσματα. Εδώ μέσα σε μία try και με την βοήθεια του DriverManager.getConnection() συνδεόμαστε στην βάση δεδομένων με τα στοιχεία που θα δώσουμε . Όπου μέσα πρέπει να έχει την μορφή getConnection(database,username,password).

7.2)Στήσιμο JasperReports

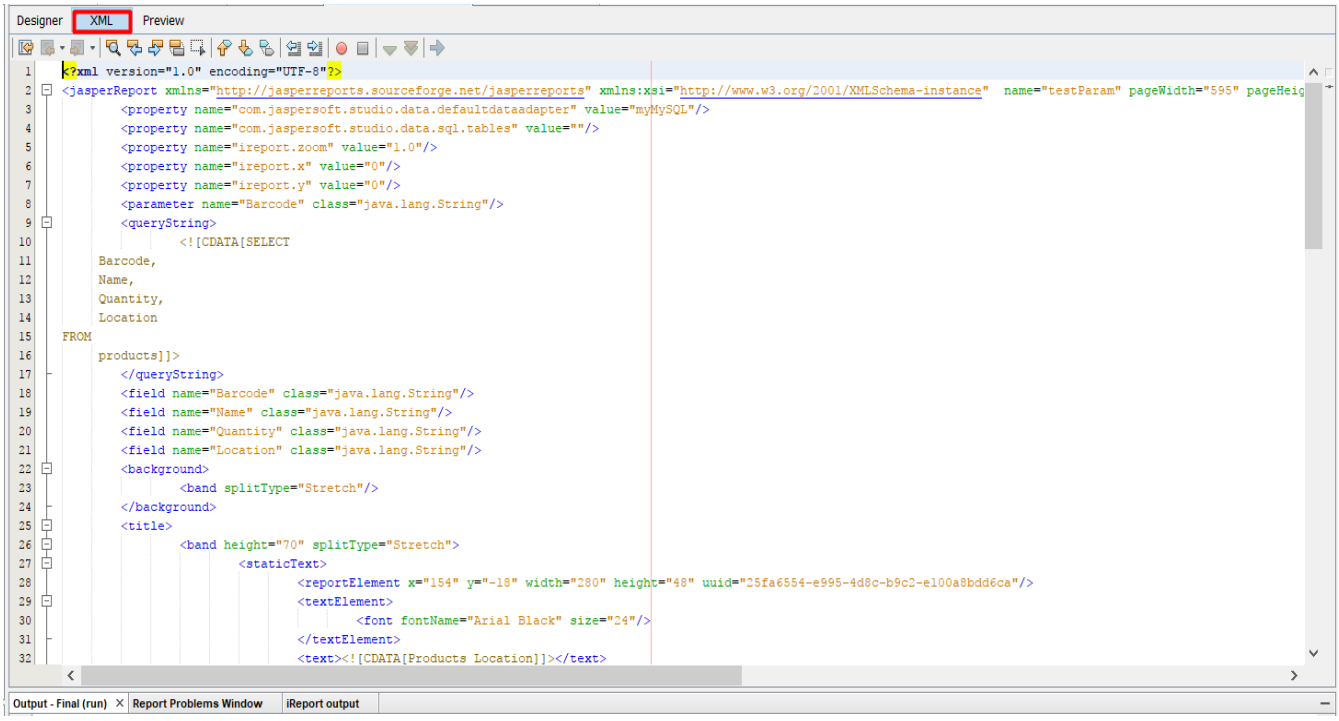
Είχαμε προσθέσει στον πίνακα των προϊόντων ένα κουμπί που μας δίνει την δυνατότητα να τα εκτυπώσουμε σε μία κόλλα Α4. Εδώ θα εξηγήσουμε πώς το καταφέραμε αυτό.


JasperViewer
100%
Products Location
Barcode
Name
Quantity
Location
1773052
GeForce GTX 1080
10
B2
1773194
Omega Centauri Greek
7
D4
1773198
Omega Centauri Greek
12
D4
2489643
SanFisk SSD Plus 120GB
16
A1
Page 1 of 1

Αρχικά πρέπει να φτιάξουμε το πώς θα είναι η προεπισκόπηση των προϊόντων. Αφού έχουμε κάνει import όλα τα απαραίτητα πράγματα για να μπορεί να τρέξει το JasperReports. Φτιάχνουμε το jrxml μας που στην προκειμένη περίπτωση είναι κάπως έτσι:

Εικόνα 7.2-3: Product Location

Μετά από αυτό το βήμα, αφού θέλουμε να τραβάμε πληροφορίες από μία βάση δεδομένων θα πάμε στο xml αρχείου που έχουμε δημιουργήσει παραπάνω.



Εικόνα 7.2-4: Product Location XML code

```
        <parameter name="Barcode" class="java.lang.String"/>
        <queryString>
                <![CDATA[SELECT
    Barcode,
    Name,
    Quantity,
    Location
FROM
    products]]>
        </queryString>
        <field name="Barcode" class="java.lang.String"/>
        <field name="Name" class="java.lang.String"/>
        <field name="Quantity" class="java.lang.String"/>
        <field name="Location" class="java.lang.String"/>
```

Εικόνα 7.2-5: Product Location XML code

Όπου δημιουργούμε ένα query το οποίο απευθύνεται στην βάση δεδομένων. Αμέσως μετά τα αποτελέσματα που μας δίνει η «ερώτηση» που κάναμε στην βάση δεδομένων τα τοποθετούμε στο jrxml που φτιάξαμε.

Και παρακάτω έχουμε το κουμπί (make a report) το οποίο βρίσκεται στα Products. Αρχικά δηλώνουμε την τοποθεσία του jrxml αρχείου και μετά όλες οι κατάλληλες εντολές ώστε να το εμφανίσουμε.

```
    try {
        InputStream url7 = getClass().getResourceAsStream("/myireports/ProductReport.jrxml");
        JasperDesign jasperDesign = JRXmlLoader.load(url7);
        JasperReport jasperReport = JasperCompileManager.compileReport(jasperDesign);
        JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, null, con);
        JasperViewer.viewReport(jasperPrint,false);


    } catch (JRException ex) {
        System.out.println("print: " + ex.getMessage());
        String connectMsg = "Could not create the report " + ex.getMessage() + " " + ex.getLocalizedMessage();
        System.out.println(connectMsg);
    }
```

Εικόνα 7.2-6: Product Location XML code

Βιβλιογραφία

1. C.J. Date, (2003) Εισαγωγή στα Συστήματα Βάσεων Δεδομένων, Τόμος Α΄ (Εκδόσεις Κλειδάριθμος) 6 η Αμερικάνικη έκδοση
2. **NetBeans:** https://en.wikipedia.org/wiki/NetBeans
3. Ιωάνης Μανωλόπουλος,Απόστος Ν.Παπαδόπουλος Συστήματα ΒάσεωνΔεδομένων Θεωρία και πρακτική εφαρμογή.
4. **MySqlWorkbench:** https://en.wikipedia.org/wiki/MySQL_Workbench
5. **Jaspersoft:** https://www.jaspersoft.com/
6. **Draw.io:** https://www.draw.io/