**Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης**

**Σχολή Τεχνολογικών Εφαρμογών**
**Τμήμα Εφαρμοσμένης Πληροφορικής και Πολυμέσων**

# Πτυχιακή εργασία

# Ανάπτυξη συστήματος διαχείρισης αποθήκης

**Παντελής Μπαλαδάκης (Α.Μ. 2659)**

**Επιβλέπων Καθηγητής: Μαλάμος Αθανάσιος, Καπετανάκης Κωνσταντίνος**

# Abstract

Inventory management is one of the most important business functions of a company. This thesis analyses the implementation of a decentralized warehouse management system using modern Internet technology. We will examine each technology and discuss the reason that it was selected and we will present in detail each stage of the development process.

# Σύνοψη

Η διαχείριση των αποθεμάτων των προϊόντων μίας επιχείρησης είναι μία από τις πιο σημαντικές λειτουργίες της. Σε αυτή της εργασία θα ασχοληθούμε με την ανάπτυξη ενός αποκεντρωμένου συστήματος διαχείρισης μίας αποθήκης, με την χρήση σύγχρονων τεχνολογιών του διαδικτύου. Θα αναλύσουμε τις τεχνολογίες και τους λόγους που αυτές επιλέχθηκαν και θα παρουσιάσουμε όλα τα στάδια της διαδικασίας ανάπτυξης της εφαρμογής.

# Περιεχόμενα

Abstract ..... 3
Σύνοψη ..... 4
Περιεχόμενα ..... 5
1 Εισαγωγή ..... 8
1.1 Περίληψη ..... 8
1.2 Κίνητρο για την διεξαγωγή της εργασίας ..... 8
1.3 Σκοπός και στόχοι εργασίας ..... 8
1.4 Δομή εργασίας ..... 8
2 Συστήματα διαχείρισης αποθήκης ..... 10
2.1 Γενικά ..... 10
2.2 Πλεονεκτήματα ..... 10
2.3 Ιστορικά ..... 10
2.4 Πρώτα προϊόντα λογισμικού ..... 11
2.5 Εξέλιξη τεχνολογιών ..... 11
2.5.1 Γραφικές διεπαφές ..... 11
2.5.2 Βάσεις δεδομένων ..... 12
2.6 Εξέλιξη συστημάτων διαχείρισης αποθήκης ..... 12
3 Ανάλυση εφαρμογής ..... 14
3.1 Τεχνολογίες που χρησιμοποιήθηκαν ..... 14
3.1.1 Γενικά ..... 14
3.1.2 MySQL ..... 14
3.1.3 HTML και CSS ..... 14
3.1.4 PHP ..... 15
4 Δομή εφαρμογής ..... 16
4.1 Απαιτήσεις συστήματος ..... 16
4.2 Οργάνωση αρχείων ..... 16
4.3 Βάση δεδομένων ..... 16
4.3.1 Πίνακας users ..... 16
4.3.2 Πίνακας category ..... 17
4.3.3 Πίνακας product ..... 17
4.3.4 Πίνακας transfer ..... 18
4.4 Βιβλιοθήκες ..... 19

4.4.1 jpgraph ... 19
4.4.2 mpdf ... 19
4.4.3 phpExcel ... 19
4.5 Επικοινωνία με βάση δεδομένων ... 19
4.5.1 Συνάρτηση connect ... 19
4.5.2 Συνάρτηση query ... 20
4.5.3 Συνάρτηση query_row ... 21
4.5.4 Συνάρτηση query_single ... 22
4.5.5 Συνάρτηση update ... 23
4.6 Βασική δομή σελίδων ... 23
4.6.1 Αρχείο header.php ... 23
4.6.2 Αρχείο footer.php ... 29
4.6.3 Μία βασική σελίδα ... 30
4.7 Ανάλυση σελίδων ... 30
4.7.1 Είσοδος στο σύστημα ... 31
4.7.2 Έξοδος από το σύστημα ... 32
4.7.3 Κεντρική σελίδα ... 33
4.7.4 Διαχείριση Χρηστών ... 33
4.7.5 Διαχείριση Κατηγοριών προϊόντων ... 36
4.7.6 Διαχείριση προϊόντων ... 39
4.7.7 Διαχείριση κινήσεων προϊόντων ... 44
4.7.8 Δημιουργία γραφημάτων ... 50
4.7.9 Μαζική εισαγωγή συναλλαγών ... 55
5 Οδηγίες χρήσης ... 59
5.1 Επισκέπτης ... 59
5.1.1 Είσοδος στο σύστημα ... 59
5.2 Απλός χρήστης ... 59
5.2.1 Προβολή κατηγοριών ... 60
5.2.2 Προβολή χρηστών ... 61
5.2.3 Προϊόντα ... 61
5.2.4 Αγορές και Πωλήσεις ... 62
5.2.5 Έξοδος ... 64
5.3 Διαχειριστής ... 64
5.3.1 Χρήστες ... 65
5.3.2 Κατηγορίες ... 66
5.3.3 Προϊόντα ... 68

5.3.4 Αγορές και πωλήσεις ..... 71
5.4 Μαζική εισαγωγή στοιχείων ..... 73
5.5 Γράφημα κίνησης προϊόντος ..... 73
6 Συμπεράσματα ..... 75
7 Bibliography ..... 76

# 1 Εισαγωγή

## 1.1 Περίληψη

Σε αυτή η εργασία θα ασχοληθούμε με το λογισμικό που χρησιμοποιείται για την διαχείριση των εμπορευμάτων μίας αποθήκης. Θα παρουσιαστεί η ιστορική εξέλιξη του και οι παράλληλη εμφάνιση και εξέλιξη των τεχνολογιών που χρησιμοποιούνται για αρκετές εφαρμογές στο διαδίκτυο. Μετά θα παρουσιάσουμε πως αυτές οι τεχνολογίες μπορούν να χρησιμοποιηθούν για την ανάπτυξη ενός τέτοιου συστήματος και πόσο εύκολο είναι να συνδυαστούν για να παράγουν το επιθυμητό αποτέλεσμα.

## 1.2 Κίνητρο για την διεξαγωγή της εργασίας

Μία από τις πρώτες χρήσεις των υπολογιστών σε εταιρικά περιβάλλοντα ήταν η διαχείριση των αποθεμάτων των προϊόντων. Τα προγράμματα που χρησιμοποιούνταν για αυτή τη δουλειά ήταν αρχικά εφαρμογές που έδιναν ιδιαίτερο βάρος στην εκτέλεση των απαιτούμενων λειτουργιών αγνοώντας τους τομείς της φιλικότητας προς τον χρήστη και των επιδόσεων. Εξαιτίας της καθημερινής χρήσης τους και του πλήθους των εξειδικευμένων δυνατοτήτων που περιέχουν πολλά από αυτά χρησιμοποιούνται ακόμη και σήμερα, αρκετά χρόνια ή μερικές φορές και δεκαετίες μετά την δημιουργία τους.

Η ευρεία ανάπτυξη του World Wide Web έχει οδηγήσει στην ανάπτυξη αρκετών τεχνολογιών που έχουν σαν σκοπό την φιλική σχεδίαση των διεπαφών χρήστη, την οπτική αναπαράσταση μεγάλου εύρους πληροφοριών, την αποθήκευση και τη διαχείριση μεγάλου όγκου δεδομένων. Όλα τα παραπάνω είναι τα απαραίτητα συστατικά για την ανάπτυξη ενός τέτοιου συστήματος. Θα θέλαμε να εξερευνήσουμε την προοπτική χρήσης τους για την δημιουργία μίας μεγάλης εφαρμογής τέτοιου είδους.

## 1.3 Σκοπός και στόχοι εργασίας

Στόχος της εργασίας είναι η ανάπτυξη μίας μοντέρνας εφαρμογής διαχείρισης αποθήκης χρησιμοποιώντας σύγχρονες τεχνολογίες ανάπτυξης διαδικτυακών εφαρμογών. Θα δημιουργήσουμε μία εφαρμογή της οποίας το interface θα είναι γραμμένο σε HTML5 και CSS3, με την αποθήκευση των δεδομένων να γίνεται σε μία βάση δεδομένων MySQL και την γλώσσα PHP για την δυναμική δημιουργία των σελίδων και την επικοινωνία με την βάση δεδομένων.

Μέσω αυτού θα εξετάσουμε την καταλληλόλητα των συγκεκριμένων τεχνολογιών για αυτό το είδος εφαρμογών και θα αναλύσουμε την διαδικασία ανάπτυξης με χρήση κάθε μίας από αυτές.

## 1.4 Δομή εργασίας

Το πρώτο κεφάλαιο της παρουσιάζει περιληπτικά το περιεχόμενο και τους στόχους.

Το δεύτερο κεφάλαιο δείχνει την ιστορική διαδρομή των συστημάτων διαχείρισης αποθήκης και των αντίστοιχων τεχνολογιών.

Στο τρίτο κεφάλαιο δείχνουμε τις δυνατότητες της εφαρμογής και τις τεχνολογίες με τις οποίες υλοποιήθηκε.

Το τέταρτο κεφάλαιο επικεντρώνεται στην ανάλυση της βάσης δεδομένων και του κώδικα που υλοποιήθηκε.

Το πέμπτο κεφάλαιο παρουσιάζει τα συμπεράσματα που προέκυψαν από την έρευνα και την διαδικασία ανάπτυξης.

# 2 Συστήματα διαχείρισης αποθήκης

## 2.1 Γενικά

Η διαχείριση των αποθεμάτων των προϊόντων μίας επιχείρησης είναι ένας πάρα πολύ σημαντικός τομέας για την σωστή και επικερδή λειτουργία της. Με αυτόν τον όρο αναφερόμαστε στην παρακολούθηση των μεταβολών της ποσότητας κάθε προϊόντος, είτε πρόκειται για αγορά είτε για πώληση και στην εξαγωγή στοιχείων μέσο της επεξεργασίας των αποτελεσμάτων.

## 2.2 Πλεονεκτήματα

Οι εταιρίες που χρησιμοποιούν εκτενώς τέτοιες λύσεις προσπαθούν να μειώσουν το κόστος λειτουργίας τους μέσω της αποτελεσματικότερης διαχείρισης των αποθεμάτων. Αν αυτή γίνεται αποτελεσματικά μπορεί να οδηγήσει σε πολλά οφέλη για την εταιρεία, τα οποία παρουσιάζονται παρακάτω.

- Αποφυγή ελλείψεων προϊόντων, καθώς είναι ευκολότερο να φανεί πότε η ποσότητα ενός αγαθού μειώνεται.
- Αποφυγή συσσώρευσης μεγάλης ποσότητας από ένα αγαθό. Αντί να παραγγέλνονται περιοδικά πάγιες ποσότητες από ένα προϊόν μπορούν να γίνουν νέες παραγγελίες μόνο όταν είναι απαραίτητο, αποφεύγοντας την συσσώρευση παλιότερων προϊόντων και την καταστροφή τυχών ευπαθών προϊόντων με το πέρασμα του χρόνου.
- Παρακολούθηση των θέσεων και των μετακινήσεων των προϊόντων καθώς αυτά μεταφέρονται ανάμεσα σε διάφορες τοποθεσίες της εταιρείας.
- Απομόνωση του ανθρώπινου παράγοντα, που πιθανών να δημιουργήσει λάθη καθώς πρόκειται για μία βαρετή και επαναλαμβανόμενη διαδικασία.
- Εξοικονόμηση προσωπικού με την αυτοματοποίηση της διαδικασίας.
- Άμεση εξαγωγή συγκεντρωτικών αναφορών και στοιχείων.
- Παρακολούθηση των τάσεων αγοράς και πώλησης που διευκολύνουν την λήψη επαγγελματικών αποφάσεων.

## 2.3 Ιστορικά

Η παρακολούθηση των αποθεμάτων είναι μία επίπονη και αρκετά πολύπλοκη διαδικασία η οποία παραδοσιακά γινότανε με την χρήση χειρόγραφων εγγράφων. Κάποιος υπάλληλος χρειάζεται να αναγνωρίσει τα προϊόντα και να γράψει τα στοιχεία τους και τα στοιχεία της συναλλαγής ή της μεταφοράς.

Είναι φανερό πως ειδικά για επιχειρήσεις που διακινούν μεγάλο όγκο προϊόντων η παραπάνω διαδικασία είναι χρονοβόρα και απαιτεί αρκετό προσωπικό για να γίνει, ενώ ταυτόχρονα υπάρχει αυξημένη πιθανότητα να γίνουν λάθη και παραλείψεις. Επίσης μετά την καταγραφή των συναλλαγών δημιουργείται ένας μεγάλος όγκος εγγράφων, των οποίων η επεξεργασία είναι εξίσου δύσκολη.

Η αυτοματοποίηση της διαδικασίας άρχισε να φαίνεται σαν μία λογική πρόοδος στον τομέα. Αν και οι πρώτες προσπάθειες έγιναν από το 1932 στο Harvard University Business Administration, βασίζονταν στη χρήση μηχανικών συστημάτων που ήταν αρκετά πολύπλοκα και ακριβά.

Μετέπειτα προσπάθειες διαχώρισαν το πρόβλημα σε δύο μέρη. Το πρώτο ασχολείται με την αυτοματοποίηση της αναγνώρισης των προϊόντων και το δεύτερο με την συλλογή και επεξεργασία των δεδομένων.

Η αυτοματοποίηση της διαδικασίας ξεκίνησε να αναπτύσσεται το 1952, αλλά οι συσκευές που παρήχθησαν ήταν αρκετά ογκώδεις, ακριβές και δύσκολες στη χρήση. Αυτή η προσπάθεια κατέληξε σε αποτελέσματα το 1970, όπου αναπτύχθηκε το πρότυπο UPC (Universal Product Code), που με χρήση της νέας τότε τεχνολογίας των γραμμωτών κωδικών (barcode), υιοθέτησε μία συγκεκριμένη μορφή.

Η επεξεργασία των δεδομένων μέχρι τότε συναντούσε το πρόβλημα πως οι υπολογιστές της εποχής δεν μπορούσαν να χειριστούν τα διαθέσιμα δεδομένα. Όπως και σε αυτόν τον τομέα έγινε πρόοδος, αρχικά με τη χρήση εξειδικευμένων μηχανημάτων και μετέπειτα με τους υπολογιστές γενικής χρήσης, οι οποίοι χρησιμοποιούσαν εξειδικευμένο λογισμικό.

Παρακάτω θα επικεντρωθούμε τον τομέα του λογισμικού, που είναι και το θέμα της εργασίας.

## 2.4 Πρώτα προϊόντα λογισμικού

Το λογισμικό που χρησιμοποιήθηκε αρχικά για την διαχείριση αποθήκης έπαιρνε την είσοδο με την χρήση punch cards και εκτύπωνε τα αποτελέσματα σε χαρτί, κάνοντας την χρήση του αρκετά δύσκολη. Όμως η εξέλιξη των υπολογιστών οδήγησε σύντομα την αντικατάστασή του με συστήματα που χρησιμοποιούσαν πληκτρολόγιο και ποντίκι.

Οι περιορισμένες δυνατότητες παραγωγής γραφικών που είχαν οι πρώτοι υπολογιστές σε συνδυασμό με την περιορισμένη μνήμη και τον αδύναμο επεξεργαστή καθόρισαν την μορφή και τον τρόπο λειτουργίας του. Αρχικά δεν επέτρεπαν την ανάπτυξη πολύπλοκων και εύχρηστων διεπαφών, με αποτέλεσμα να τρέχει στην γραμμή εντολών. Συνήθως για την επιτέλεση διαφόρων λειτουργιών χρειαζόταν η χρήση κωδικών εντολών.

Η αποθήκευση των δεδομένων από την άλλη έπρεπε να γίνει με τον πιο απλό τρόπο, καθώς το μικρό μέγεθος της μνήμης δεν επέτρεπε πολύπλοκες δομές δεδομένων. Κάθε πρόγραμμα χρησιμοποιούσε την δική του μορφή αρχείων, και τα δεδομένα πολλές φορές ήταν μοιρασμένα σε πολλά αρχεία συγκεκριμένου μεγέθους ώστε να χωράνε να φορτωθούν στη μνήμη.

## 2.5 Εξέλιξη τεχνολογιών

Αν και οι αρχικές επιλογές που παρουσιάσαμε υπαγορεύονταν από τους περιορισμούς της τεχνολογίας της εποχής, αυτή σε καμία περίπτωση δεν έμεινε στάσιμη. Παρακάτω θα εξετάσουμε τους τομείς της διεπαφής και της αποθήκευσης δεδομένων.

### 2.5.1 Γραφικές διεπαφές

Τα πρώτα συστήματα μπορούσαν να εμφανίσουν μόνο χαρακτήρες, όμως σύντομα υλοποιήθηκε η δυνατότητα εμφάνισης γραφικών. Αυτό αρχικά οδήγησε στη δημιουργία προγραμμάτων που μπορούσαν να εμφανίσουν συγκεκριμένα γραφικά όπως γραφήματα ή εικόνες αλλά σύντομα άρχισαν να αναπτύσσονται εφαρμογές και μετέπειτα λειτουργικά συστήματα που όλη η διεπαφή ήταν γραφική.

Σήμερα όλοι οι υπολογιστές και οι υπολογιστικές συσκευές γενικής χρήσης χρησιμοποιούν κάποια μορφή γραφικής διεπαφής, εκτός από πολύ εξειδικευμένες περιπτώσεις. Ακόμα πιο σημαντικό είναι πως η ανάπτυξη μίας γραφικής διεπαφής δεν αποτελεί μία επίπονη διαδικασία καθώς υπάρχει μία πληθώρα βιβλιοθηκών για κάθε πλατφόρμα που καθιστά την συγγραφή τέτοιων προγραμμάτων εξαιρετικά απλή.

Επιπλέον έχουν αναπτυχθεί αρκετές γλώσσες προγραμματισμού που έχουν σαν στόχο την περιγραφή της εμφάνισης μίας σελίδας. Η πιο γνωστή είναι η HTML η οποία χρησιμοποιείται για την συγγραφή των σελίδων στο διαδίκτυο.

### 2.5.2 Βάσεις δεδομένων

Η αποθήκευση δεδομένων αρχικά γινότανε με την χρήση ενός η περισσοτέρων αρχείων, το περιεχόμενο των οποίων διαχειρίζονταν η εφαρμογή που τα χρησιμοποιούσε. Αν και η έρευνα στις βάσεις δεδομένων είχε ξεκινήσει από το 1960, ενώ οι σχεσιακές βάσεις δεδομένων άρχισαν να αναπτύσσονταν το 1970, η χρήση ενός επιπλέον εξειδικευμένου προγράμματος μόνο για την αποθήκευση των δεδομένων δεν ήταν εφικτή, αφενός γιατί δεν υπήρχαν ώριμα τέτοια προγράμματα αλλά κυρίως γιατί οι πόροι των υπολογιστών της εποχής δεν επαρκούσαν ώστε να μπορούν να τρέχουν τέτοιες εφαρμογές.

Όμως αυτό άλλαξε σύντομα με την αναπόφευκτη αύξηση των δυνατοτήτων των υπολογιστών και την ανάγκη για συνεπή αποθήκευση και ανάκληση δεδομένων, που οδήγησε στην καθιέρωση των σχεσιακών βάσεων δεδομένων και της γλώσσας SQL. Η αποθήκευση δεδομένων έπαψε να είναι μία διαδικασία που απαιτεί την ανάπτυξη εξειδικευμένου λογισμικού αλλά πλέον κάθε πρόγραμμα μπορεί να επικοινωνήσει με μία βάση δεδομένων SQL και να αποθηκεύσει τα δεδομένα που χρειάζεται ή να ζητήσει αποτελέσματα από απλές ή πολύπλοκες επερωτήσεις.

## 2.6 Εξέλιξη συστημάτων διαχείρισης αποθήκης

Όπως δείξαμε παραπάνω οι τεχνολογίες που χρειάζεται να χρησιμοποιήσει ένα σύστημα διαχείρισης αποθήκης έχουν αναπτυχθεί ραγδαία και πολλές νέες εφαρμογές τις χρησιμοποιούν με εντυπωσιακά αποτελέσματα. Παρόλα αυτά υπάρχουν σε λειτουργία και αρκετά συστήματα που τρέχουν απαρχαιωμένες εκδόσεις του λογισμικού. Οι λόγοι για τους οποίους συμβαίνει αυτό είναι αρκετοί.

Ο σημαντικότερος είναι η πολυπλοκότητα ενός τέτοιου συστήματος. Το πλήθος των λειτουργιών που παρέχει σημαίνει ότι η συγγραφή ενός νέου προγράμματος θα είναι ταυτόχρονα χρονοβόρα και θα έχει μεγάλο κόστος. Επιπλέον υπάρχει μεγάλος κίνδυνος εμφάνισης προβλημάτων μετά την αντικατάσταση του παλιού συστήματος, όπως υπάρχει και με κάθε νέα εφαρμογή. Τα παλιότερα συστήματα έχουν δοκιμαστεί για χρόνια ή πολλές φορές δεκαετίες, ενώ ένα νέο σύστημα θα χρειαστεί αρκετό χρόνο μέχρι να φτάσει στο ίδιο επίπεδο ωριμότητας. Κάθε πρόβλημα μπορεί να συνεπάγεται κάποιο χρόνο κατά την οποία η επιχείρηση δεν θα λειτουργεί, κάτι που πολλές φορές είναι απαγορευτικό λόγω κόστους.

Η ίδια η διαδικασία της αλλαγής του υπάρχοντος συστήματος πολλές φορές απαιτεί πάρα πολύ χρόνο από μόνη της, ενώ αν τα παλιότερα συστήματα τρέχουν σε εξίσου παλιούς υπολογιστές τότε χρειάζεται και αγορά υλικού, το οποίο πρέπει να ελεγχτεί αν είναι συμβατό με τις υπάρχουσες υποδομές.

Πέρα από την εγκατάσταση του λογισμικού χρειάζεται και η μεταφορά των εγγραφών από το παλιό σύστημα στο νέο, με τα παλιότερα συστήματα σπάνια να υποστηρίζουν κάποια λειτουργία εξαγωγής δεδομένων. Αυτό σημαίνει πως τα δεδομένα της εφαρμογής πρέπει να διαβαστούν από ένα εξειδικευμένο πρόγραμμα, γεγονός που αυξάνει περισσότερο το κόστος υλοποίησης.

Το προσωπικό της επιχείρησης χρειάζεται και αυτό να εκπαιδευτεί για το νέο λογισμικό, κάτι που είναι πιο δύσκολο όσο περισσότερους υπαλλήλους έχει μία εταιρεία.

Ένας ακόμη βασικός παράγοντας είναι το γεγονός πως κάθε επιχείρηση μπορεί να έχει εξειδικευμένες ανάγκες τις οποίες μπορεί να ικανοποιεί το υπάρχον λογισμικό της αλλά να μην καλύπτονται από άλλες λύσεις που υπάρχουν στην αγορά.

# 3 Ανάλυση εφαρμογής

## 3.1 Τεχνολογίες που χρησιμοποιήθηκαν

### 3.1.1 Γενικά

Ο στόχος μας ήταν η δημιουργία μίας εφαρμογής βασισμένη στις σύγχρονες τεχνολογίες του διαδικτύου με αρχιτεκτονική client / server. Μεγαλύτερο πλεονέκτημα αυτής της αρχιτεκτονικής είναι πως ενώ πολλοί χρήστες μπορούν να είναι συνδεδεμένοι στο σύστημα και να την χρησιμοποιούν ταυτόχρονα, τα στοιχεία βρίσκονται σε έναν server και παραμένουν συγχρονισμένα. Ακόμη μπορούμε να χρησιμοποιήσουμε τις αρκετά ώριμες τεχνολογίες που χρησιμοποιούνται στο World Wide Web και την τεχνογνωσία που έχει αναπτυχθεί μαζί με αυτές.

Σε αυτή την ενότητα θα παρουσιάσουμε τις τεχνολογίες που χρησιμοποιήσαμε.

### 3.1.2 MySQL

Το σημαντικότερο συστατικό της εφαρμογής είναι οι πληροφορίες για τα προϊόντα και τις κινήσεις τους. Αυτά θα αποθηκεύονται σε μία σχεσιακή βάση δεδομένων ώστε να μπορούν να αναπαρασταθούν και οι μεταξύ τους σχέσεις. Επιλέξαμε να χρησιμοποιήσουμε την βάση MySQL.

Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακής βάσης ανοικτού κώδικα όπως λέγεται (relational database management system - RDBMS), που χρησιμοποιεί την Structured Query Language (SQL), την πιο γνωστή γλώσσα για την προσθήκη, την πρόσβαση και την επεξεργασία δεδομένων σε μία Βάση Δεδομένων. Η βάση δεδομένων MySQL έχει γίνει η πιο δημοφιλής βάση δεδομένων ανοικτού κώδικα, λόγω της αυξημένης απόδοσης, της υψηλής αξιοπιστίας και της ευελιξίας που παρέχει.

Επιπλέον, χρησιμοποιείται σε περισσότερες από 6 εκατομμύρια εγκαταστάσεις κλιμακούμενες από μεγάλες εταιρείες μέχρι εξειδικευμένες εφαρμογές με ενσωματωμένες βάσεις δεδομένων σε όλο τον κόσμο. Επειδή είναι ανοικτού κώδικα (open source), οποιοσδήποτε μπορεί να κατεβάσει την MySQL και να την διαμορφώσει σύμφωνα με τις ανάγκες του, σύμφωνα πάντα με την γενική άδεια που υπάρχει. Είναι κοινά αποδεκτό πως δουλεύει καλύτερα όταν διαχειρίζεται περιεχόμενο και όχι όταν εκτελεί συναλλαγές. Η MySQL αυτή τη στιγμή μπορεί να λειτουργήσει σε περιβάλλον Linux, Unix, και Windows.

### 3.1.3 HTML και CSS

Το interface της εφαρμογής είναι γραμμένο στη γλώσσα παρουσίασης περιεχομένου HTML σε συνδυασμό με την γλώσσα μορφοποίησης περιεχομένου CSS. Η HTML χρησιμοποιείται ευρέως για την δημιουργία όλων των σελίδων στο διαδίκτυο. Στόχος της γλώσσας είναι να περιγράψει την δομή μίας σελίδας μαζί με το περιεχόμενό της, και μπορεί να χρησιμοποιηθεί για σελίδας με όσο απλή ή πολύπλοκη δομή είναι επιθυμητό.

Η HTML χρησιμοποιείται σε συνδυασμό με την γλώσσα CSS. Μία σωστά δομημένη σελίδα HTML δίνει πληροφορίες μόνο για το περιεχόμενο και την δομή της, όμως η εμφάνισή της είναι βασική. Για την τροποποίηση της εμφάνισης χρησιμοποιείται η γλώσσα CSS. Ο στόχος είναι να παραμείνει η βασική σελίδα HTML όσο πιο «καθαρή» γίνεται, όπου με το καθαρή εννοείται να περιέχει μόνο τις απαραίτητες πληροφορίες και τίποτα παραπάνω. Έπειτα αυτή η σελίδα συνδυάζεται με ένα αρχείο CSS, το οποίο περιγράφει την εμφάνιση κάθε στοιχείου.

Για κάθε στοιχείο ορίζονται οι επιθυμητοί κανόνες. Κάθε κανόνας περιέχει την διαδρομή του στοιχείου στο οποίο αντιστοιχεί μέσα στο έγγραφο και πως αυτό θα εμφανίζεται. Οι κανόνες εμφάνισης μπορούν να αναφέρονται σε ένα πλήθος χαρακτηριστικών από τα χρώματα και το μέγεθος μέχρι το μέγεθος και τη στοίχιση των γραμμάτων που περιέχει. Πάλι ανάλογα με την περίπτωση μπορούν να χρησιμοποιηθούν λιγότεροι οι περισσότεροι κανόνες για να περιγράψουν μία σελίδα που θα είναι αισθητικά λιγότερο ή περισσότερο περίτεχνη.

Ο διαχωρισμός της HTML από το αρχείο CSS κάνει πολύ εύκολη την τροποποίηση του ενός από τα δύο χωρίς να χρειάζεται να τροποποιηθεί το άλλο. Με αυτόν τον τρόπο είναι εφικτό να χρησιμοποιηθεί ένα αρχείο CSS για όλες τις σελίδες ενός site, δίνοντας μία ομοιόμορφη εμφάνιση άσχετα με το περιεχόμενο κάθε σελίδας. Επιπλέον είναι δυνατόν να τροποποιηθεί μόνο το CSS ενός site, αλλάζοντας συνολικά τη μορφή όλων των σελίδων.

#### *3.1.3.1 HTML5*

Η γλώσσα HTML δημιουργήθηκε αρχικά για την προβολή απλών σελίδων κειμένου και την σύνδεση τους με άλλες. Σταδιακά άρχισε να υπάρχει η ανάγκη κατασκευής πιο εντυπωσιακών σελίδων, οι οποίες θα αντιδρούν στις ενέργειες του χρήστη, ενώ τα τελευταία χρόνια έχουν αρχίσει να εμφανίζονται σελίδες που είναι ολοκληρωμένες εφαρμογές. Όμως η τελευταία έκδοση του πρότυπου της HTML είναι η 4.01 που οριστικοποιήθηκε το 1999. Καθώς οι απαιτήσεις από την γλώσσα αυξανόταν άρχισε να γίνεται επιτακτική η ανάγκη μίας νέας έκδοσης που θα ανταποκρίνεται σε αυτές.

Η επόμενη έκδοση της HTML είναι η έκδοση 5, η οποία άρχισε να αναπτύσσεται το 2004 και αναμένεται να οριστικοποιηθεί μετά το 2015. Η νέα γλώσσα προσφέρει μία πλειάδα δυνατοτήτων με σκοπό να κάνει την δημιουργία ιστοσελίδων πολύ πιο εύκολη και να μην είναι απαραίτητη η χρήση ξένων εργαλείων. Η HTML5 είναι ένα πρότυπο που αναπτύσσεται εδώ και πολλά χρόνια με σκοπό να αποτελέσει την κύρια γλώσσα κατασκευής εφαρμογών στο διαδίκτυο. Τον τελευταίο καιρό έχουν παγιωθεί αρκετά από τα στοιχεία της, κάνοντας εφικτή την δοκιμή τους σε πρακτικό επίπεδο.

Στην εργασία χρησιμοποιήσαμε αρκετά στοιχεία από την έκδοση 5. Εξαιτίας της φύσης της γλώσσας και του τρόπου με τον οποίο ενσωματώνονται οι αλλαγές δεν υπάρχει κάποια ριζική απόκλιση από την προηγούμενη έκδοση, απλά εισάγονται στοιχεία που προσφέρουν επιπλέον λειτουργίες.

### 3.1.4 PHP

Η PHP είναι μια γλώσσα οντοκεντρική γλώσσα προγραμματισμού ξεκίνησε να αναπτύσσεται το 1991. Παρόλο που μπορεί να χρησιμοποιηθεί σαν γλώσσα προγραμματισμού γενικής χρήσης έχει καθιερωθεί να τρέχει σε συνδυασμό με έναν web server για τη δημιουργία σελίδων Web με δυναμικό περιεχόμενο. Εξαιτίας της φύσης των εργασιών που χρειάζεται να επιτελεστούν στον προγραμματισμό μίας διαδικτυακής εφαρμογής τις περισσότερες φορές χρησιμοποιείται σαν διαδικαστική γλώσσα, χωρίς να γίνεται χρήση των οντοκεντρικών της δυνατοτήτων.

Πρόκειται για την περισσότερο διαδεδομένη γλώσσα για την δημιουργία ιστοσελίδων, όπου χρησιμοποιείται σε περισσότερα από 250 εκατομμύρια σελίδες. Η ευρεία διάδοσή της οφείλεται στην ευκολία χρήσης της και στο πλήθος των δυνατοτήτων που προσφέρει. Έχει δυνατότητες επικοινωνίας με έναν web server ώστε να λαμβάνει αιτήσεις από αυτόν και να επιστρέφει δυναμικά δημιουργημένες σελίδες, ενώ μπορεί να επικοινωνήσει με όλες τις δημοφιλείς βάσεις δεδομένων.

# 4 Δομή εφαρμογής

## 4.1 Απαιτήσεις συστήματος

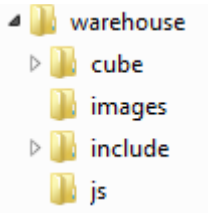
Η εφαρμογή είναι γραμμένη σε γλώσσα PHP, με τις σελίδες που παράγει να είναι γραμμένες σε HTML και τα δεδομένα αποθηκεύονται σε μία σχεσιακή βάση δεδομένων. Αυτό σημαίνει πως για να λειτουργήσει απαιτεί την ύπαρξη ενός web server που μπορεί να επεξεργαστεί αρχεία PHP.

Επιλέξαμε τον Apache server και τη βάση δεδομένων MySQL. Για την εγκατάσταση τους χρησιμοποιήσαμε το πρόγραμμα EasyPHP, το οποίο περιέχει τα παραπάνω προγράμματα με όλες τις ρυθμίσεις που χρειάζονται ώστε να συνεργάζονται μεταξύ τους. Το EasyPHP είναι διαθέσιμο από τη διεύθυνση http://sourceforge.net/projects/quickeasyphp/files/latest/download?source=dlp.

## 4.2 Οργάνωση αρχείων

Τα αρχεία της εφαρμογής βρίσκονται στο φάκελο warehouse. Αυτός ο φάκελος χρειάζεται να βρίσκεται στη διαδρομή C:\Program Files (x86)\EasyPHP-12.1\www, η οποία είναι η διαδρομή μέσα στην οποία ο Apache ψάχνει να βρει τα αρχεία που του ζητάνε. Τα αρχεία της εφαρμογής που θα ζητάμε από τον server θα βρίσκονται με τη σειρά τους στη διεύθυνση http://localhost/warehouse/.

Δημιουργήσαμε την παρακάτω δομή φακέλων:




Οι φάκελοι εξυπηρετούν στις εξής λειτουργίες:

- images: Σε αυτό τον φάκελο αποθηκεύονται οι εικόνες που χρησιμοποιούνται στην εργασία.
- include: Στον φάκελο include αποθηκεύονται όλα τα αρχεία PHP που χρησιμοποιούνται σαν βιβλιοθήκες, δηλαδή αυτά που χρησιμοποιούμε αυτούσια σε όλες τις σελίδες. Επιπλέον εκεί βρίσκονται όλες οι βιβλιοθήκες PHP που χρησιμοποιήθηκαν για τις επιπλέον λειτουργίες.
- js: Σε αυτόν τον φάκελο βρίσκονται όλες οι βιβλιοθήκες JavaScript που χρησιμοποιήθηκαν για την βελτίωση της λειτουργικότητας του site.
- warehouse: Ο φάκελος warehouse περιέχει τα αρχεία που δημιουργούν κάθε σελίδα καθώς και αυτά που εκτελούν τις λειτουργίες του προγράμματος.

## 4.3 Βάση δεδομένων

Καθώς τα δεδομένα αποθηκεύονται σε μία βάση δεδομένων ήταν απαραίτητη η δημιουργία των κατάλληλων πινάκων για τη σωστή αποθήκευση τους. Χρησιμοποιήσαμε 4 πίνακες.

### 4.3.1 Πίνακας users

Ο πίνακας users περιέχει τα στοιχεία των χρηστών. Έχει την παρακάτω μορφή:

| # | Name | Type | Collation | Attributes | Null | Default | Extra |
|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | No | None | AUTO_INCREMENT |
| 2 | name | varchar(15) | utf8_general_ci | | No | None | |
| 3 | password | varchar(15) | utf8_general_ci | | No | None | |
| 4 | is_admin | tinyint(1) | | | No | None | |

Το πεδίο **id** είναι το πρωτεύον κλειδί του πίνακα. Είναι ένας ακέραιος που η τιμή του αποδίδεται και αυξάνεται αυτόματα με κάθε εισαγωγή.

Το πεδίο **name** κρατάει το όνομα κάθε χρήστη. Είναι ένα αλφαριθμητικό (VARCHAR) 15 χαρακτήρων.

Το πεδίο **password** κρατάει τον κωδικό του χρήστη. Είναι ένα αλφαριθμητικό (VARCHAR) 15 χαρακτήρων.

Το πεδίο **is_admin** αποθηκεύει το αν ο χρήστης έχει δικαιώματα διαχειριστή ή όχι. Αν ένας χρήστης είναι διαχειριστής τότε έχει τιμή 1, αλλιώς 0. Είναι ένας BOOLEAN.

### 4.3.2 Πίνακας category

Ο πίνακας category κρατάει τις κατηγορίες των προϊόντων. Είναι ο πιο απλός πίνακας και περιέχει μόνο δύο πεδία.

| # | Name | Type | Collation | Attributes | Null | Default | Extra |
|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | No | None | AUTO_INCREMENT |
| 2 | name | varchar(50) | utf8_general_ci | | No | None | |

Το πεδίο **id** είναι το πρωτεύον κλειδί του πίνακα. Είναι ένας ακέραιος που η τιμή του αποδίδεται και αυξάνεται αυτόματα με κάθε εισαγωγή.

Το πεδίο **name** κρατάει το όνομα κάθε κατηγορίας. Είναι ένα αλφαριθμητικό (VARCHAR) 50 χαρακτήρων.

### 4.3.3 Πίνακας product

Ο πίνακας product περιέχει τα στοιχεία κάθε προϊόντος. Έχει 5 πεδία.

Αξίζει να σημειωθεί ότι η ποσότητα κάθε προϊόντος δεν αποθηκεύεται σε αυτόν τον πίνακα σαν ακέραιος. Καθώς δεν μας ενδιαφέρει μόνο η απόλυτη ποσότητα αλλά οι μεταβολές της σε συνάρτηση με τον χρόνο η ύπαρξη ενός ακεραίου στον πίνακα product δεν καλύπτει τις ανάγκες μας.

| # | Name | Type | Collation | Attributes | Null | Default | Extra |
|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | No | None | AUTO_INCREMENT |
| 2 | category_id | int(11) | | | No | None | |
| 3 | name | varchar(50) | utf8_general_ci | | No | None | |
| 4 | description | text | utf8_general_ci | | No | None | |
| 5 | price_buy | float | | | No | None | |
| 6 | price_sell | float | | | No | None | |

Το πεδίο **id** είναι το πρωτεύον κλειδί του πίνακα. Είναι ένας ακέραιος που η τιμή του αποδίδεται και αυξάνεται αυτόματα με κάθε εισαγωγή.

Το πεδίο **category_id** είναι ένα ξένο κλειδί του πίνακα. Σε αυτό κρατείται το id της κατηγορίας στην οποία ανήκει το προϊόν. Είναι ένας ακέραιος.

Το πεδίο **name** κρατάει το όνομα κάθε προϊόντος. Είναι ένα αλφαριθμητικό (VARCHAR) 50 χαρακτήρων.

Το πεδίο **description** είναι η περιγραφή ενός προϊόντος. Είναι τύπου text, δηλαδή μπορεί να περιέχει κείμενο απεριόριστου μήκους.

Το πεδίο **price_buy** περιέχει την τιμή αγοράς ενός προϊόντος. Είναι ένα πεδίο τύπου float.

Το πεδίο **price_sell** περιέχει την τιμή αγοράς ενός προϊόντος. Είναι ένα πεδίο τύπου float.

### 4.3.4 Πίνακας transfer

Καθώς θέλουμε να παρακολουθούμε τις μεταβολές στην ποσότητα κάθε προϊόντος χρειάστηκε να δημιουργήσουμε τον πίνακα transfer ώστε να αποθηκεύουμε την κάθε μία ξεχωριστά.

Ο πίνακας έχει τα παρακάτω πεδία:

| # | Name | Type | Collation | Attributes | Null | Default | Extra |
|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | No | None | AUTO_INCREMENT |
| 2 | product_id | int(11) | | | No | None | |
| 3 | stock | int(11) | | | No | None | |
| 4 | date | date | | | No | None | |

Το πεδίο **id** είναι το πρωτεύον κλειδί του πίνακα. Είναι ένας ακέραιος που η τιμή του αποδίδεται και αυξάνεται αυτόματα με κάθε εισαγωγή.

Το πεδίο **product_id** είναι ένα ξένο κλειδί του πίνακα. Σε αυτό κρατείται το id του προϊόντος για το οποίο δημιουργήθηκε η εγγραφή. Καθώς αποθηκεύει ένα id είναι ένας ακέραιος.

Το πεδίο **stock** κρατάει το πλήθος των προϊόντων που μεταφέρθηκαν. Αν αυτό είναι θετικός πρόκειται για αγορά, ενώ αν είναι αρνητικός πρόκειται για πώληση. Είναι ένας ακέραιος.

Το πεδίο **date** είναι η ημερομηνία στην οποία έγινε η μεταφορά. Είναι τύπου date, δηλαδή κρατάει ημερομηνία αλλά όχι ώρα.

## 4.4 Βιβλιοθήκες

Εδώ θα παρουσιάσουμε τις βιβλιοθήκες που χρησιμοποιήσαμε.

### 4.4.1 jpgraph

Το jpgraph είναι μία βιβλιοθήκη PHP που αναλαμβάνει την δημιουργία γραφικών παραστάσεων. Είναι μία βιβλιοθήκη ανοικτού κώδικα που είναι διαθέσιμη από τη διεύθυνση http://jpgraph.net/download/. Μπορεί να δημιουργήσει πολλών ειδών γραφήματα και παρέχει μεγάλες δυνατότητες παραμετροποίησης τους.

### 4.4.2 mpdf

Το mpdf είναι μία βιβλιοθήκη που αναλαμβάνει την μετατροπή μίας σελίδας HTML σε ένα έγγραφο PDF. Είναι διαθέσιμη από τη διεύθυνση http://www.mpdf1.com/mpdf/download.

### 4.4.3 phpExcel

Το phpExcel είναι μία βιβλιοθήκη που δίνει την δυνατότητα ανάγνωσης και δημιουργίας αρχείων Excel. Βρίσκεται στη διεύθυνση http://phpexcel.codeplex.com/. Η κλάση είναι συμβατή με αρχεία doc και docx που χρησιμοποιούνται από τις τελευταίες εκδόσεις του Microsoft Office, καθώς και με αρχεία CSV που παράγονται από άλλες εφαρμογές.

## 4.5 Επικοινωνία με βάση δεδομένων

Σχεδόν όλες οι λειτουργίες της εφαρμογής χρησιμοποιούν ή τροποποιούν περιεχόμενα της βάσης δεδομένων. Η PHP παρέχει αρκετές συναρτήσεις που δίνουν την δυνατότητα εκτέλεσης των λειτουργιών που χρειάζονται, όμως αυτές έχουν συγκεκριμένα ονόματα ανάλογα με τον τύπο της βάσης δεδομένων που χρησιμοποιείται και χρειάζονται να χρησιμοποιηθούν με έναν συγκεκριμένο τρόπο που καθιστά την εκ των υστέρων ανάγνωση του κώδικα δύσκολη.

Για τους παραπάνω λόγους δημιουργήθηκε μία σειρά συναρτήσεων που αναλαμβάνει την επικοινωνία με την βάση επιστρέφοντας τα αποτελέσματα με την μορφή βασικών τύπων δεδομένων και πινάκων PHP.

Πρόκειται για το αρχείο db.php το οποίο βρίσκεται στον φάκελο include. Κάθε φορά που χρειάζεται να χρησιμοποιηθεί η βάση γίνεται include το παραπάνω αρχείο και μετά καλούνται οι κατάλληλες συναρτήσεις.

### 4.5.1 Συνάρτηση connect

Η συνάρτηση connect αναλαμβάνει την σύνδεση με τη βάση. Αρχικά ορίζεται ένας πίνακας DB_CONF, μέσα στον οποίο ορίζεται το όνομα χρήστη και ο κωδικός για τη σύνδεση στη MySQL, ο υπολογιστής στον οποίο αυτή βρίσκεται και το όνομα της βάσης δεδομένων που χρησιμοποιείται.

```
    $DB_CONF['host'] = 'localhost';
    $DB_CONF['name'] = 'warehouse';
    $DB_CONF['user'] = 'root';
    $DB_CONF['password'] = '';
```

Φαίνεται πως χρησιμοποιούνται σαν όνομα χρήστη το root και κανένας κωδικός. Πρόκειται για τον χρήστη που δημιουργείται από το EasyPHP.

Έπειτα καλούνται οι συναρτήσεις για την σύνδεση στη βάση.

```
    mysql_connect($DB_CONF['host'], $DB_CONF['user'],
$DB_CONF['password']) or die('connect(): '.mysql_error());

    mysql_set_charset('utf8');

    mysql_select_db($DB_CONF['name']) or die('selectdb():
'.mysql_error());
```

Η εφαρμογή συνδέεται στη βάση, επιλέγει να παίρνει αποτελέσματα σε κωδικοποίηση UTF-8 ώστε να μην υπάρχει πρόβλημα με τα ελληνικά και επιλέγει την κατάλληλη βάση.

Η συνάρτηση connect δεν χρειάζεται να καλεστεί ανεξάρτητα αλλά καλείται από κάθε άλλη συνάρτηση που τη χρησιμοποιεί.

### 4.5.2 Συνάρτηση query

Η συνάρτηση query χρησιμοποιείται για την αποστολή ενός ερωτήματος SQL και την επιστροφή της απάντησης, αν υπάρχει, με τη μορφή δισδιάστατου πίνακα PHP.

```
    function query($q)
```

Αρχικά συνδέεται στη βάση δεδομένων με την connect και τρέχει το ερώτημα SQL που δίνεται σαν παράμετρο.

```
    connect();

    $result = mysql_query($q) or die('query:"$q" '.mysql_error());
```

Το αποτέλεσμα επιστρέφεται στη μεταβλητή result, όμως πρόκειται για ένα αντικείμενο PHP, το οποίο θέλουμε να το κάνουμε πίνακα. Γι' αυτό το λόγο δημιουργείται ένας κενός πίνακας και διατρέχεται ανά γραμμή.

```
    $ar = array();

    for($i=0; $row = mysql_fetch_assoc($result); $i++){
        $ar[$i] = array();
…
    }
```

Σε κάθε επανάληψη παίρνουμε τα δεδομένα από μία εγγραφή της απάντηση σαν ένα πίνακα PHP και εισάγουμε έναν νέο κενό πίνακα στον πίνακα που δημιουργήθηκε. Ένας μετρητής i που αυξάνεται σε κάθε επανάληψη καθορίζει τη θέση που θα μπει σε αυτόν.

```
        foreach($row as $key => $value){
            $ar[$i][$key] = $value;
        }
```

Εσωτερικά με μία δεύτερη for διατρέχουμε τη γραμμή αυτή και βάζουμε τα στοιχεία της στον πίνακα ar.

Όταν διατρέξουμε όλες τις γραμμές του αποτελέσματος επιστρέφουμε τον ar. Συνολικά ο κώδικας της συνάρτησης είναι αυτός:

```
function query($q){
    connect();

    $result = mysql_query($q) or die('query:"$q" '.mysql_error());

    $ar = array();

    for($i=0; $row = mysql_fetch_assoc($result); $i++){
        $ar[$i] = array();
        foreach($row as $key => $value){
            $ar[$i][$key] = $value;
        }
    }

    return $ar;
}
```

### 4.5.3 Συνάρτηση query_row

Η συνάρτηση query_row λειτουργεί όπως η query με τη διαφορά ότι επιστρέφει μόνο μία γραμμή του αποτελέσματος. Είναι χρήσιμη σε περιπτώσεις που τρέχουμε μία εντολή που ξέρουμε ότι επιστρέφει μόνο ένα αποτέλεσμα.

Και αυτή τρέχει το ερώτημα q, όμως ελέγχει ότι επιστρέφετε ακριβώς μία εγγραφή.

```
    if(mysql_num_rows($result) != 1){
        return null;
    }
```

Αν δεν ισχύει αυτό επιστρέφει null για την αποφυγή τυχών λαθών, αλλιώς συνεχίζει και παίρνει τα αποτελέσματα. Δεν χρειάζεται επανάληψη γιατί ξέρουμε πως έχουμε ένα αποτέλεσμα.

```
    $row = mysql_fetch_assoc($result);
```

Μετά διατρέχει αυτή τη γραμμή και εισάγει τα δεδομένα σε έναν νέο πίνακα τον οποίο επιστρέφει. Ο πλήρης κώδικας είναι ο εξής:

```
function query_row($q){
    connect();
    $result = mysql_query($q) or die('query_row:"'.$q.'"
'.mysql_error());

    if(mysql_num_rows($result) != 1){
        return null;
    }
    $row = mysql_fetch_assoc($result);

    $ar = array();
    foreach($row as $key => $value){
        $ar[$key] = $value;
    }
    return $ar;
}
```

### 4.5.4 Συνάρτηση query_single

Η query_single είναι μία ακόμη βοηθητική συνάρτηση που επιστρέφει μόνο μία τιμή. Είναι παρόμοια με τις προηγούμενες αλλά αφού γνωρίζουμε πως έχουμε μόνο μία τιμή για αποτέλεσμα δεν χρειάζεται να το διατρέξουμε.

```
function query_single($q){
    connect();

    $result = mysql_query($q) or die('query_single:"$q"
'.mysql_error());

    if(mysql_num_rows($result) != 1){
        return null;
    }

    $row = mysql_fetch_array($result);

    return $row[0];
}
```

### 4.5.5 Συνάρτηση update

Η συνάρτηση update εκτελεί ένα ερώτημα SQL αλλά δεν επιστρέφει καμία τιμή, χρήσιμο σε περιπτώσεις που εισάγουμε κάτι στη βάση και δεν χρειάζεται να κάνουμε κάτι παραπάνω.

```
function update($q){
    connect();

    $result = mysql_query($q) or die('update:"$q"
'.mysql_error());
}
```

## 4.6 Βασική δομή σελίδων

Το χαρακτηριστικό που έχουν όλες οι σελίδες είναι πως μοιράζονται ένα μεγάλο κομμάτι κοινού κώδικα HTML. Επιπλέον αυτός βρίσκεται συγκεντρωμένος στην αρχή και το τέλος της σελίδας. Πρόκειται για τον κώδικα που δημιουργεί τη σελίδα και το αριστερό μενού καθώς και για το κάτω μέρος της σελίδας.

Για να αποφύγουμε την επανάληψη όμοιου κώδικα σε όλες τις σελίδες και να διευκολύνουμε τυχών μελλοντικές αλλαγές δημιουργήσαμε δύο αρχεία header και footer τα οποία εισάγονται στην αρχή και στο τέλος της σελίδας.

### 4.6.1 Αρχείο header.php

Το αρχείο header.php περιέχει την αρχική HTML της σελίδας:

```
<?php session_start() ?>
<!doctype html>
<html>
<head>
```

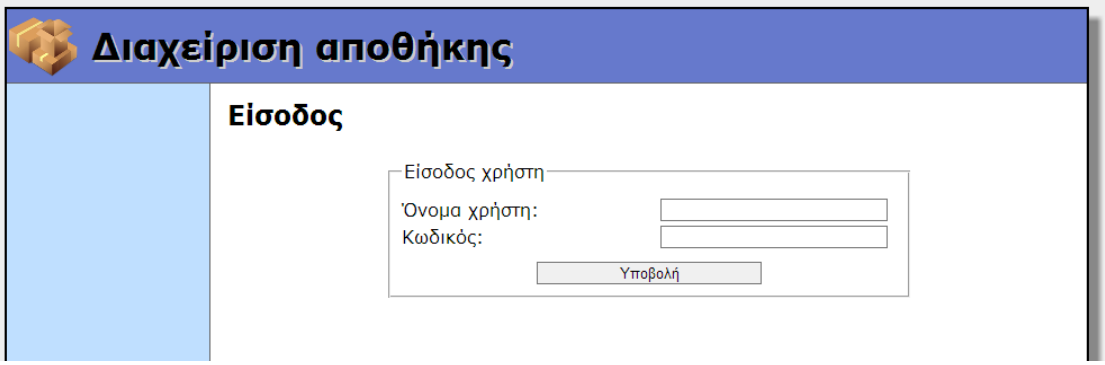
```
    <meta charset="utf-8">
    <title>Διαχείριση Αποθήκης</title>
    <link href="style.css" rel="stylesheet" type="text/css">

    <!--[if lt IE 9]><script
src="js/html5shiv.js"></script><![endif]-->
    <script src="js/jquery-1.8.2.js"></script>
    <script src="js/jquery.validate.js"></script>
</head>

<body>
<div class="container">
    <header>
        <h1>Διαχείριση αποθήκης</h1>
    </header>

    <nav>
        <ul>
<?php

?>


        </ul>
        <!--
        <aside>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing
elit. Praesent urna velit, bibendum nec lacinia vitae, vestibulum
vitae enim.</p>
        </aside>
        -->
    </nav><!-- END nav -->

    <article>
    <?php

    ?>
```

Όμως είναι υπεύθυνο και για την δυναμική δημιουργία περιεχομένου, όπως φαίνεται παρακάτω.

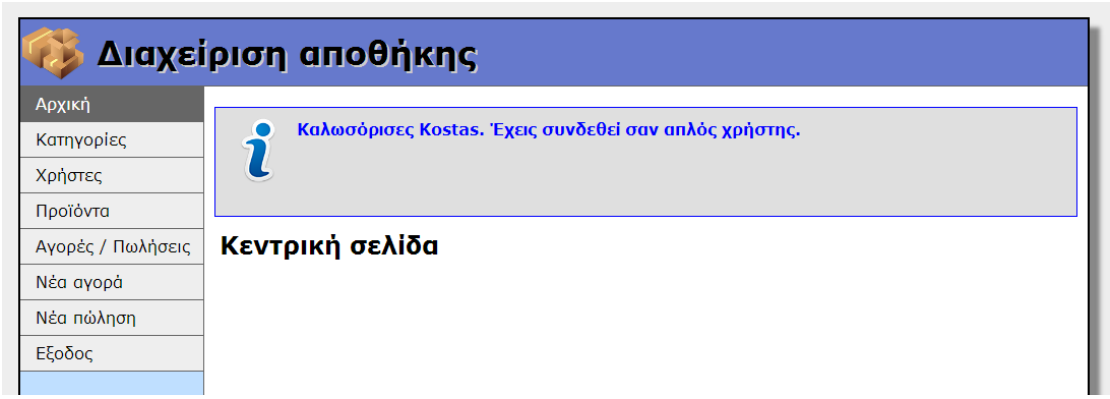
#### *4.6.1.1 Δημιουργία μενού*

Στα αριστερά της σελίδας υπάρχει ένα μενού με όλους τους διαθέσιμου συνδέσμους. Όμως αυτοί διαφέρουν ανάλογα με τον χρήστη που τους βλέπει.

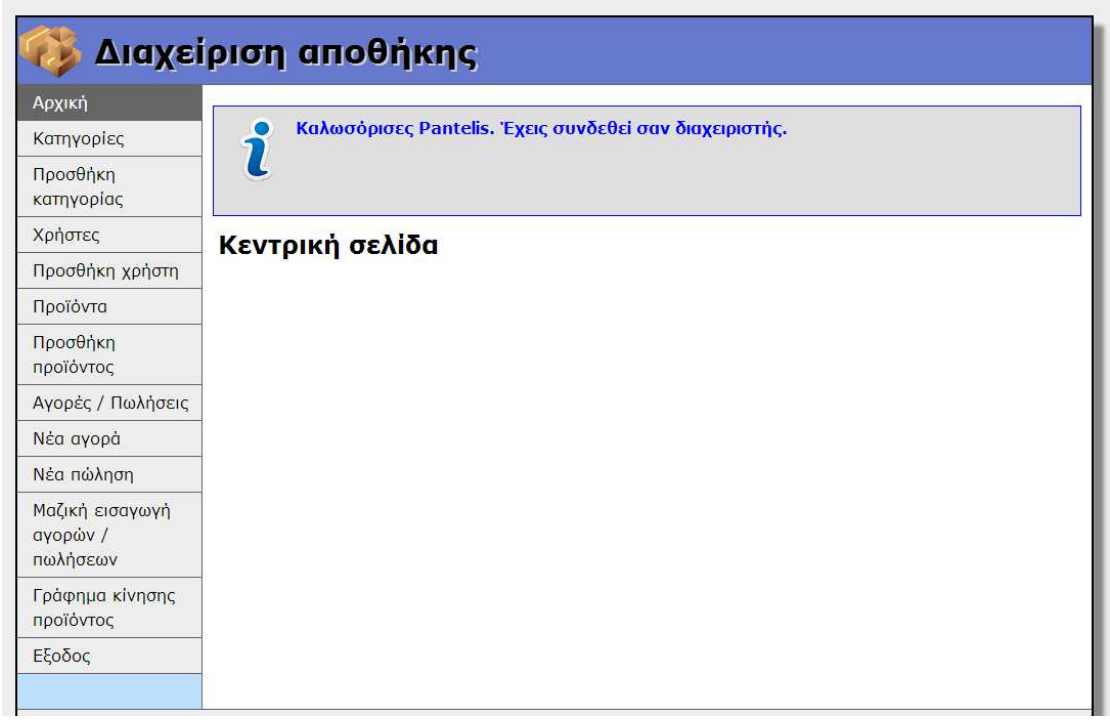
Η πιο απλή περίπτωση είναι όταν ο χρήστης δεν έχει κάνει login, οπότε δε βλέπει καμία επιλογή.




Σε περίπτωση που έχει μπει σαν απλός χρήστης βλέπει τις επιλογές που του επιτρέπονται.




Τέλος αν μπει σαν διαχειριστής έχει πρόσβαση σε όλες τις επιλογές του συστήματος.



Άρα φαίνεται πως το μενού χρειάζεται να δημιουργείται δυναμικά και να λαμβάνει υπόψη του τα δικαιώματα κάθε χρήστη.

Για τη δημιουργία του μενού δημιουργείται αρχικά ένας δισδιάστατος πίνακας PHP που περιέχει όλες τις επιλογές των μενού. Για κάθε μία από αυτές περιέχεται το κείμενο που φαίνεται στα μενού, το link στο οποίο παραπέμπει και το επίπεδο του χρήστη που μπορεί να τη δει ανάλογα με το αν είναι διαχειριστής ή όχι.

```
<?php
    $menu = array(
        array(  "url" => "main.php",
                "text" => "Αρχική",
                "admin" => 0
            ),
        array(  "url" => "category_list.php",
                "text" => "Κατηγορίες",
                "admin" => 0
            ),
        array(  "url" => "category_add.php",
                "text" => "Προσθήκη κατηγορίας",
                "admin" => 1
            ),
        ...

    );
```

Όταν ένας χρήστης έχει κάνει Login τότε αποθηκεύεται στο session μία ιδιότητα που καθορίζει αν είναι διαχειριστής ή όχι. Πρόκειται για το $_SESSION['user_is_admin'], βάση του οποίου δημιουργείται το μενού.

Συγκεκριμένα με μία for διατρέχεται ο παραπάνω πίνακας.

```
    foreach($menu as $item){
…
    }
```

Μέσα σε αυτή ελέγχεται αρχικά αν έχει οριστεί η παραπάνω μεταβλητή. Αν όχι το αντικείμενο αγνοείται, δηλαδή δεν εμφανίζεται τίποτα.

Αν το τρέχον αντικείμενο έχει δηλωθεί πως μπορεί να το δει μόνο διαχειριστής και ενώ η μεταβλητή στο session λέει πως δεν έχει μπει διαχειριστής τότε πάλι δεν εμφανίζεται τίποτα.

```
        if(!isset($_SESSION['user_is_admin']) ||
($_SESSION['user_is_admin'] == 0 && $item['admin'] == 1))

            continue;
```

Σε αντίθετη περίπτωση εμφανίζεται κανονικά.

```
        echo '<li><a href="'.$item['url'].'" ';

        if('/warehouse/'.$item['url'] == $current)
            echo 'class="selected" ';

        echo '>'.$item['text'].'</a></li>';
    }
```

Το στοιχείο $current είναι η διεύθυνση της σελίδας που προβάλλεται, η οποία παίρνεται δυναμικά.

```
    $current = $_SERVER['PHP_SELF'];
```

Αν αυτή συμπίπτει με το τρέχον στοιχείο τότε εισάγεται ένα επιπλέον CSS class σε αυτό, ώστε να είναι εμφανές ότι είναι αυτό που προβάλλεται.

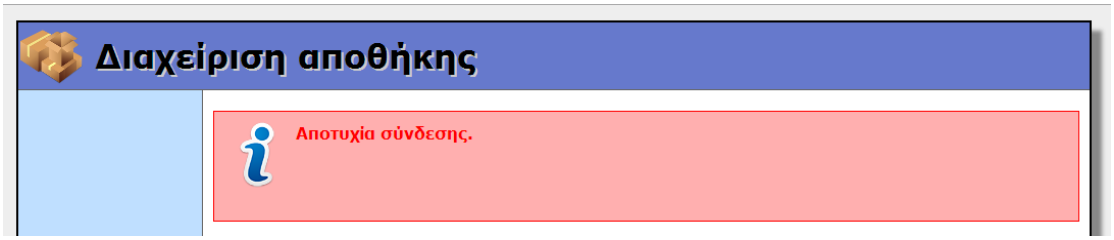
```
        if('/warehouse/'.$item['url'] == $current)

            echo 'class="selected" ';
```

#### *4.6.1.2 Εμφάνιση μηνυμάτων*

Πολλές φορές είναι επιθυμητό να ενημερώσουμε τον χρήστη πως κάποια ενέργεια που έκανε αποθηκεύτηκε ή πως παρουσιάστηκε κάποιο πρόβλημα. Εξαιτίας της διάρθρωσης των αρχείων δεν αρκεί απλά να τυπώσουμε το μήνυμα, γιατί το λάθος μπορεί να έγινε σε ένα αρχείο διαφορετικό από αυτό που προβάλλεται. Γι' αυτό τον λόγο χρησιμοποιείται ένας μηχανισμός μέσω του session.

Ένα μήνυμα επιτυχής ενέργειας φαίνεται παρακάτω:

Αντίθετα ένα μήνυμα λάθους εμφανίζεται ως εξής:




Για να δημιουργήσει ένα αρχείο ένα μήνυμα το οποίο θα το δει ο χρήστης γίνεται χρήση δύο μεταβλητών, των $_SESSION['errors'] και isset($_SESSION['info'].

Ένα μήνυμα λάθος δημιουργείται αποθηκεύοντας στο session το μήνυμα, για παράδειγμα:

```
$_SESSION['errors'] = 'Αποτυχία σύνδεσης.';
```

Ένα μήνυμα που περιέχει μία πληροφορία δημιουργείται με την χρήση του info.

```
$_SESSION['info'] = 'Καλωσόρισες '
```

Το header.php ελέγχει αν σε αυτές τις μεταβλητές είναι αποθηκευμένες κάποιες τιμές.

```
if(isset($_SESSION['errors'])){
```

Αν βρει μία τέτοια τιμή τότε εμφανίζει το περιεχόμενο της με κατάλληλο τρόπο.

```
    echo '<div class="error"> <p>' . $_SESSION['errors'] .
'</p></div>';
```

Όμως για να μην εμφανιστεί και την επόμενη φορά είναι αναγκαίο να σβήσει αυτή την τιμή.

```
            unset($_SESSION['errors']);
```

Με παρόμοιο τρόπο εμφανίζεται και το πληροφοριακό μήνυμα. Συνολικά ο κώδικας που αναλαμβάνει αυτή τη διαδικασία είναι:

```
    <?php
        if(isset($_SESSION['errors'])){

            echo '<div
class="error"><p>'.$_SESSION['errors'].'</p></div>';

            unset($_SESSION['errors']);
        }

        if(isset($_SESSION['info'])){

            echo '<div
class="info"><p>'.$_SESSION['info'].'</p></div>';

            unset($_SESSION['info']);
        }
    ?>
```

### 4.6.2 Αρχείο footer.php

Το footer.php εισάγεται στο τέλος των σελίδων και περιέχει τον κώδικα για να τερματίσουν τα στοιχεία HTML, μαζί με τα στοιχεία της σελίδας.

```
    </article><!-- END article -->

    <footer>
        <p>Τμήμα Εφαρμοσμένης Πληροφορικής και Πολυμέσων</p>
        <p> Δημιουργία πλατφόρμας διαχείρισης αποθήκης
καταστήματος και προβολής στατιστικών γραφημάτων</p>

    </footer>
</div><!-- END .container -->
</body>
</html>
```

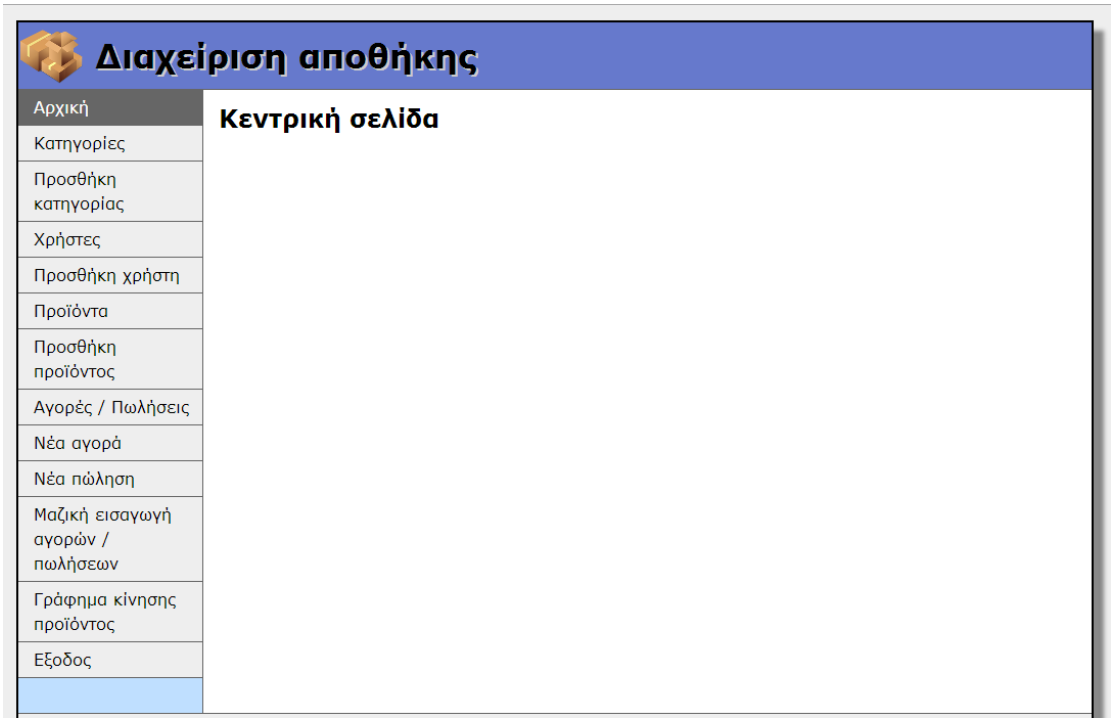
Παρόλο που θα μπορούσαν να είχαν μπει και σε αυτό λειτουργίες PHP, δεν υπήρχε λόγος με την υπάρχουσα δομή.

### 4.6.3 Μία βασική σελίδα

Με τα δύο παραπάνω αρχεία επιτυγχάνουμε να απλοποιήσουμε πάρα πολύ την δομή κάθε σελίδας. Συγκεκριμένα ένα νέο αρχείο αρκεί να τα περιέχει στην αρχή και στο τέλος:

```
<?php
    require_once('include/header.php');
?>


<?php
    require_once('include/footer.php');
?>
```

Οι δύο παραπάνω εντολές αρκούν για την δημιουργία μίας σελίδας:




Με αυτό τον τρόπο το αρχείο κάθε σελίδας είναι όσο πιο απλό γίνεται και δεν επαναλαμβάνεται ο κώδικας που είναι κοινός σε όλες τις σελίδες, κάνοντας τις αλλαγές σε αυτόν πολύ πιο εύκολες.

## 4.7 Ανάλυση σελίδων

Σε αυτή την ενότητα θα παρουσιάσουμε τις επιμέρους σελίδες που δημιουργήσαμε. Θα τις χωρίσουμε σε κατηγορίες ανάλογα με την οντότητα που ασχολούνται.

Οι περισσότερες ενέργειες χρειάζονται δύο αρχεία, ένα αρχείο όπου ο χρήστης συμπληρώνει μία φόρμα και ένα αρχείο που επεξεργάζεται τα στοιχεία της.

### 4.7.1 Είσοδος στο σύστημα

Η φόρμα για την είσοδο στο σύστημα δημιουργείται στο αρχείο index.php, το οποίο είναι το πρώτο αρχείο που φορτώνεται όταν ο χρήστης μπει στη διεύθυνση http://127.0.0.1/warehouse/.

#### *4.7.1.1 Αρχείο index.php*

Το αρχείο φορτώνει τα αρχεία header και footer στην αρχή και στο τέλος, όπως δείξαμε στη βασική δομή των σελίδων.

Ο κώδικας που δημιουργεί τη φόρμα είναι ο παρακάτω:

```
    <form method="post" action="login_action.php">
    <fieldset id="login"><legend>Είσοδος χρήστη</legend>
        <label>Όνομα χρήστη:</label><input type="text" name="name"
class="required" /><br/>
        <label>Κωδικός:</label><input type="password"
name="password" class="required" /><br/>
        <input type="submit" value="Υποβολή" />
    </fieldset>
    </form>
```

Τα πεδία class που έχουν τιμή required χρησιμοποιούνται ώστε ο χρήστης να μην μπορεί να υποβάλλει την φόρμα χωρίς να τα έχει συμπληρώσει. Ο έλεγχος γίνεται με javascript και ενεργοποιείται με την παρακάτω εντολή:

```
    <script>
    $(document).ready(function(){
        $("form").validate({
                errorClass: "invalid",
                errorElement: ""
            });
    });
    </script>
```

Η φόρμα παραπέμπει στο αρχείο login_action.php.

#### *4.7.1.2 Αρχείο login_action.php*

Στο αρχείο αυτό επεξεργάζονται τα στοιχεία της φόρμας εισόδου. Αφού πρόκειται για αρχείο που επεξεργάζεται μία φόρμα φορτώνει με την require_once το αρχείο db.php για να χρησιμοποιήσει τις συναρτήσεις επικοινωνίας με τη βάση δεδομένων. Το αρχείο δεν εμφανίζει κάτι στον χρήστη αλλά τον παραπέμπει σε άλλες σελίδες, οπότε δεν χρειάζεται να περιέχει τα header και footer.

Αρχικά παίρνουμε τις μεταβλητές για το όνομα και τον κωδικό που έχει στείλει η φόρμα και επιλέγουμε τους χρήστες που έχουν αυτά τα στοιχεία από τη βάση.

```
    $q = "SELECT * FROM `users` WHERE name='$name' AND
password='$password'";
    $result = query_row($q);
```

Μετά ελέγχουμε αν το αποτέλεσμα είναι false, οπότε δεν υπάρχει τέτοιος χρήστης. Σε αυτή την περίπτωση κάνουμε redirect τον χρήστη στην αρχική σελίδα εμφανίζοντας το κατάλληλο μήνυμα λάθους.

```
    $_SESSION['errors'] = 'Αποτυχία σύνδεσης.';
    header("Location: index.php");
```

Αν ο χρήστης έχει δώσει έγκυρα στοιχεία τότε αποθηκεύουμε στο session κάποια από αυτά ώστε να τα χρησιμοποιήσουμε στο μέλλον. Μετά τον ανακατευθύνουμε στην κεντρική σελίδα.

```
    $_SESSION['info'] = 'Καλωσόρισες '.$result['name'];
    $_SESSION['user_id'] = $result['id'];
    $_SESSION['user_is_admin'] = $result['is_admin'];

    header("Location: main.php");
```

### 4.7.2 Έξοδος από το σύστημα

Η επιλογή της εξόδου είναι διαθέσιμη από το κεντρικό μενού, αφού ο χρήστης κάνει login.

#### *4.7.2.1 Αρχείο logout_action.php*

Το αρχείο logout_action.php απλά σβήνει από το session τις μεταβλητές που αποθηκεύσαμε κατά την είσοδο σχετικά με τον χρήστη. Έπειτα διαγράφει το τρέχον session.

```
    unset($_SESSION['user_id']);
    unset($_SESSION['user_is_admin']);

    session_destroy();
    header("Location: index.php");
```

Τελικά ο χρήστης ανακατευθύνεται στη σελίδα εισόδου.

### 4.7.3 Κεντρική σελίδα

Μετά την επιτυχή είσοδο στο σύστημα ο χρήστης πηγαίνει στην αρχική σελίδα.

#### *4.7.3.1 Αρχείο main.php*

Το αρχείο main.php περιέχει μόνο τον header και τον footer και μία επικεφαλίδα. Δεν παρέχει καμία λειτουργικότητα, απλά δείχνει το κεντρικό μενού ώστε ο χρήστης να επιλέξει τι θέλει να κάνει.

### 4.7.4 Διαχείριση Χρηστών

Ο χρήστης μπορεί να δει όλους τους χρήστες της εφαρμογής, να προσθέσει νέους, να διαγράψει ή να τροποποιήσει τα στοιχεία σε κάποιους από τους υπάρχοντες.

Αυτές οι λειτουργίες γίνονται στα παρακάτω αρχεία.

#### *4.7.4.1 Αρχείο users_list.php*

Το αρχείο users_list.php εμφανίζει μία λίστα με τους χρήστες και τα στοιχεία τους. Καθώς το βλέπει ο χρήστης χρειάζεται να περιέχει τα header και footer για να φαίνεται η μορφή της σελίδας. Επίσης χρειάζεται το αρχείο της βάσης.

Έπειτα δημιουργείται ένας πίνακας HTML με τις κατάλληλες στήλες. Μετά φορτώνονται όλοι οι χρήστες.

```
    $q = "SELECT * FROM `users`";
    $theusers = query($q);
```

Μετά διατρέχουμε τον πίνακα των αποτελεσμάτων. Για κάθε εγγραφή εισάγουμε μία γραμμή στον πίνακα και εμφανίζονται τα κατάλληλα στοιχεία σαν κελιά.

Για το πεδίο is_admin ελέγχεται η τιμή του και αντί να εμφανίζεται 1 ή 0 εισάγεται μία εικόνα ή ένα κενό κελί ανάλογα με το αν πρόκειται για διαχειριστή ή όχι.

```
if($users['is_admin'] == 1){
    echo '<td class="actions"><img src="images/tick.png" /></td>';
}else{
    echo '<td class="actions"> </td>';
}
```

Στη στήλη των ενεργειών χρειάζεται να ελέγξουμε αν ο τρέχον χρήστης είναι διαχειριστής, από το πεδίο του session που αποθηκεύσαμε κατά το login. Αν είναι εμφανίζονται οι εικόνες για την διαγραφή και την τροποποίηση σαν σύνδεσμοι.

Οι σύνδεσμοι δημιουργούνται έχοντας σαν μία παράμετρο τύπου GET το id του χρήστη.

```
echo '<td class="actions"><a
href="users_delete_action.php?id='.$users['id'].'"><img
src="images/delete.png" /></a>
        <a href="users_add.php?id='.$users['id'].'"><img
src="images/edit.png" /></a></td>';
```

Σε αντίθετη περίπτωση εμφανίζονται πάλι οι δύο εικόνες αλλά σε μία ασπρόμαυρη παραλλαγή τους για να φαίνονται απενεργοποιημένες και δεν έχουν συνδέσμους.

```
    echo '<td class="actions">';
    echo '<img src="images/delete-disabled.png" />';
    echo '<img src="images/edit-disabled.png" />';
    echo '</td>';
```

#### *4.7.4.2 Αρχείο users_delete_action.php*

Το αρχείο users_delete_action.php αναλαμβάνει την διαγραφή ενός χρήστη. Παίρνει για παράμετρο το id του χρήστη και τον σβήνει. Δεν εμφανίζει κάτι στον χρήστη οπότε δεν περιέχει άλλα αρχεία για την αρχή και το τέλος πέρα από το αρχείο με τις λειτουργίες της βάσης δεδομένων.

Το αρχείο απλά συνδέεται στη βάση και μετά σβήνει την εγγραφή με την κατάλληλη εντολή SQL.

```
$q = "DELETE FROM `warehouse`.`users` WHERE `users`.`id` = $id";
$result = update($q);
```

Μετά αποθηκεύει ένα μήνυμα για να εμφανιστεί στον χρήστη και τον ξαναγυρνάει στη λίστα των χρηστών.

```
    $_SESSION['info'] = 'Ο χρήστης διαγράφηκε.';
    header("Location: users_list.php");
```

#### *4.7.4.3 Αρχείο users_add.php*

Το αρχείο users_add.php περιέχει τη φόρμα για την εισαγωγή νέου χρήστη. Καθώς το βλέπει ο χρήστης χρειάζεται να περιέχει τα header και footer για να φαίνεται η μορφή της σελίδας.

Το ίδιο αρχείο χρησιμοποιείται για την τροποποίηση ενός υπάρχοντος χρήστη. Ανάλογα με την περίπτωση λειτουργεί διαφορετικά.

Για να καταλάβει αν πρόκειται για εισαγωγή ή τροποποίηση ελέγχεται αν έχει δοθεί σαν παράμετρος κατά την αίτηση το πεδίο id ή όχι.

```
if( isset( $_REQUEST['id'] ) )
```

Το πεδίο αυτό δίνεται από τον σύνδεσμο για την τροποποίηση του χρήστη, ενώ αν γίνει κλικ από το κεντρικό μενού (οπότε πρόκειται για εισαγωγή) δεν δίνεται.

Αν πρόκειται για εισαγωγή νέου χρήστη ορίζεται μία μεταβλητή id με τιμή ίση με 0.

Στην περίπτωση της τροποποίησης φορτώνεται από την βάση ο χρήστης με το id που δόθηκε με τα στοιχεία του αποθηκεύονται στις κατάλληλες μεταβλητές.

```
$q = "SELECT * FROM `users` WHERE `id` = $id";
$users = query_row($q);

$name = $users['name'];
$password = $users['password'];
$admin = $users['is_admin'];
```

Έπειτα εισάγεται ο κώδικας για τον έλεγχο της φόρμας όπως έχουμε δει παραπάνω και δημιουργείται η φόρμα. Και στις δύο περιπτώσεις η φόρμα είναι ίδια. Η διαφορά της τροποποίησης με την εισαγωγή είναι πως χρειάζεται τα πεδία της φόρμας να συμπληρωθούν εκ των προτέρων.

Αυτό γίνεται ελέγχοντας αν το πεδίο id έχει τιμή ίση με 0 ή όχι. Αν δεν είναι 0 τότε προστίθεται στο πεδίο μία ιδιότητα value που περιέχει την κατάλληλη τιμή, όπως φαίνεται παρακάτω για το πεδίο του ονόματος.

```
<label>Όνομα χρήστη:</label>
<input type="text" name="name" class="required"
    <?php if($id != 0) echo "value=\"$name\""; ?>
/>
<br/>
```

Επίσης υπάρχει και ένα hidden πεδίο με όνομα id που περιέχει την τιμή της αντίστοιχης μεταβλητής για να σταλεί το είδος της ενέργειας στο επόμενο αρχείο.

```
<input type="hidden" name="id" value="<?php echo $id; ?>" />
```

Η φόρμα υποβάλλεται στο users_add_action.php.

#### *4.7.4.4 Αρχείο users_add_action.php*

Σε αυτό το αρχείο παίρνουμε τις πληροφορίες που έχουν σταλεί από την φόρμα εισαγωγής και τις αποθηκεύουμε στη βάση. Δεν εμφανίζεται τίποτα στον χρήστη, οπότε περιέχεται μόνο το αρχείο με τις λειτουργίες της βάσης.

Όπως και πριν οι ενέργειες διαφοροποιούνται ανάλογα με το αν πρόκειται για εισαγωγή ή τροποποίηση.

Αρχικά παίρνουμε τις τιμές της φόρμας. Επειδή το combobox λειτουργεί διαφορετικά σε σχέση με τις υπόλοιπες μεταβλητές μας ενδιαφέρει μόνο αν έχει σταλεί μία τιμή.

```
    if(isset($_REQUEST['admin']))
        $admin = 1;
    else
        $admin = 0;
```

Έπειτα ελέγχεται η τιμή του id που έχει σταλεί. Αν είναι 0 τότε είναι μία καινούργια εισαγωγή οπότε δημιουργούμε μία εντολή SQL insert και κάνουμε την εισαγωγή.

```
$q = "INSERT INTO `warehouse`.`users` (`id`, `name`, `password`,
`is_admin`) VALUES (NULL, '$name', '$password', $admin);";
$result = update($q);
```

Αν το id δεν ήταν 0 τότε πρόκειται για μία τροποποίηση. Γράφουμε μία εντολή SQL update όπου τροποποιούμε τα στοιχεία του χρήστη που έχει το συγκεκριμένο id.

```
$q = "UPDATE `warehouse`.`users` SET `name` = '$name' WHERE
`users`.`id` = $id;";
$result = update($q);
```

Και στις δύο περιπτώσεις αποθηκεύεται το κατάλληλο μήνυμα και μετά ο χρήστης ανακατευθύνεται στη σελίδα με τη λίστα των χρηστών.

### 4.7.5 Διαχείριση Κατηγοριών προϊόντων

Στις κατηγορίες ο χρήστης μπορεί να κάνει τις ίδιες ενέργειες που κάνει και στους χρήστες, δηλαδή προβολή, εισαγωγή, διαγραφή και τροποποίηση.

Παρακάτω παρουσιάζουμε τα αρχεία που εκτελούν αυτές τις λειτουργίες.

#### *4.7.5.1 Αρχείο category_list.php*

Το αρχείο category_list.php δημιουργεί έναν πίνακα με όλες τις κατηγορίες και τις ενέργειες που μπορεί να γίνουν σε αυτές. Καθώς είναι σελίδα που προβάλλεται στον χρήστη συμπεριλαμβάνει τα αρχεία header, footer και db .

Αρχικά δημιουργείται ένας πίνακας HTML με τα κατάλληλα κελιά για επικεφαλίδες και μετά φορτώνονται οι κατηγορίες από τη βάση.

Καθώς μας ενδιαφέρει να έχουμε και το πλήθος των προϊόντων που περιέχει κάθε κατηγορία η εντολή επιλογής είναι πιο πολύπλοκη από τους χρήστες.

```
SELECT c.`id`, c.`name`, count(p.`category_id`) AS count
FROM `category` c LEFT JOIN `product` p
ON p.`category_id` = c.`id`
GROUP BY c.`id`
ORDER BY c.id
```

Διαλέγουμε όλες τις κατηγορίες από τον πίνακα category και το πλήθος των id των product κάνοντάς τον left join με τον πίνακα product στο πεδίο id του product. Με αυτόν τον τρόπο εμφανίζεται μία εγγραφή για κάθε κατηγορία ακόμη και αν δεν υπάρχουν προϊόντα σε αυτή.

Το πλήθος δημιουργείται με την μέθοδο count της SQL, όπου μετρούνται τα id των προϊόντων. Με το group by category.id προσδιορίζουμε πως θέλουμε να μετρήσουμε τα προϊόντα ανά κατηγορία.

Έπειτα διατρέχουμε τα αποτελέσματα και εμφανίζουμε τα στοιχεία. Για να εμφανίσουμε ενεργοποιημένα ή όχι κουμπιά ενεργειών χρησιμοποιούμε τις έγχρωμες και τις ασπρόμαυρες εικόνες όπως και παραπάνω.

Αν ένας χρήστης δεν είναι διαχειριστής βλέπει μόνο απενεργοποιημένα κουμπιά. Αν κάποιος είναι διαχειριστής το κουμπί της τροποποίησης είναι πάντα ενεργοποιημένο όμως το κουμπί της διαγραφής είναι ενεργοποιημένο μόνο αν μία κατηγορία δεν περιέχει προϊόντα.

Με αυτόν τον τρόπο δεν επιτρέπεται η διαγραφή κατηγοριών που να έχουν προϊόντα. Αν γινόταν κάτι τέτοιο θα έπρεπε είτε να μένουν προϊόντα χωρίς κατηγορία, που δεν επιτρέπεται, είτε να σβήνουμε όλα τα προϊόντα κάθε κατηγορίας και όλες τις κινήσεις της, που θα συνεπάγονταν μεγάλη απώλεια δεδομένων.

#### *4.7.5.2 Αρχείο category_delete_action.php*

Το αρχείο category_delete_action.php παίρνει σαν παράμετρο το id μίας κατηγορίας και τη σβήνει.

```
DELETE FROM `warehouse`.`category`
WHERE `category`.`id` = $id
```

Έχουμε προηγουμένως εξασφαλίσει πως δεν θα υπάρχουν προϊόντα σε αυτή, αφού δεν επιτρέπεται να σβηστεί μία γεμάτη κατηγορία.

Έπειτα ο χρήστης ανακατευθύνεται στη λίστα με τις κατηγορίες.

#### *4.7.5.3 Αρχείο category_add.php*

Το αρχείο category_add.php παρουσιάζει την φόρμα τροποποίησης και προσθήκης μίας νέας κατηγορίας. Το αν θα γίνει προσθήκη ή τροποποίηση αποφασίζεται ανάλογα με το αν έχει δοθεί και ένα όνομα κατηγορίας ή όχι. Αν είναι τροποποίηση φορτώνονται τα στοιχεία της κατηγορίας με το συγκεκριμένο id ώστε να γεμίσει η φόρμα.

```
SELECT * FROM `category`
WHERE `id` = $id
```

Έπειτα εισάγουμε τον κώδικα ελέγχου της φόρμας και την ίδια τη φόρμα. Η φόρμα περιέχει και ένα κρυμμένο πεδίο id που έχει την τιμή του id της κατηγορίας που τροποποιείται ή 0 αν είναι εισαγωγή.

Η φόρμα υποβάλλεται στο αρχείο category_add_action.php

#### *4.7.5.4 Αρχείο category_add_action.php*

Το αρχείο category_add_action.php περιέχει τον κώδικα για την εισαγωγή ή την τροποποίηση της κατηγορίας. Το είδος της ενέργειας αποφασίζεται ανάλογα με το αν το id είναι 0 ή όχι.

Αν το id είναι 0 δημιουργείται μία νέα εντολή insert που περιέχει το όνομα της κατηγορίας.

```
INSERT INTO `warehouse`.`category` (`id` ,`name`)
VALUES (NULL , '$name')
```

Αν δεν είναι μηδενικό τότε τροποποιείται με νέα εντολή update το προϊόν που έχει το συγκεκριμένο id.

```
UPDATE `warehouse`.`category`
SET `name` = '$name'
WHERE `category`.`id` = $id
```

Και στις δύο περιπτώσεις αποθηκεύεται το κατάλληλο μήνυμα και μετά ο χρήστης ανακατευθύνεται στη σελίδα με τη λίστα των κατηγοριών.

### 4.7.6 Διαχείριση προϊόντων

Οι ενέργειες που μπορεί να κάνει ο χρήστης σχετικές με τα προϊόντα είναι η προβολή τους, η εισαγωγή τους, η διαγραφή και η τροποποίησή τους, καθώς και η εξαγωγή τους σε ένα αρχείο Excel.

Αυτές γίνονται στα αρχεία που θα δούμε παρακάτω.

#### *4.7.6.1 Αρχείο product_list.php*

Στο αρχείο product_list.php δημιουργείται μία λίστα με όλα τα διαθέσιμα προϊόντα. Καθώς πρόκειται για κανονική σελίδα περιλαμβάνει τον header και τον footer, καθώς και το αρχείο db ώστε να γίνουν οι ενέργειες στη βάση δεδομένων.

Αρχικά δημιουργούνται οι επικεφαλίδες του πίνακα HTML με τα κατάλληλα πεδία. Έπειτα επιλέγονται όλες οι τιμές του προϊόντος από τον πίνακα product καθώς και το όνομα της κατηγορίας στην οποία ανήκει, βάση του ξένου κλειδιού στο πεδίο category_id.

```
SELECT `product`.`id`, `product`.`name`,
`product`.`description`,
    `product`.`price_buy`, `product`.`price_sell`,
    `category`.`name` AS category_name
FROM `product`, `category`
WHERE `product`.`category_id` = `category`.`id`
```

Έπειτα διατρέχουμε τα αποτελέσματα και τυπώνουμε τα στοιχεία στα κατάλληλα κελιά, αντίστοιχα με τις επικεφαλίδες που δημιουργήσαμε για τον πίνακα HMTL παραπάνω. Για λόγους ευχρηστίας υπάρχει και ένα πεδίο stock το οποίο υπολογίζει το συνολικό stock από ένα προϊόν.

Καθώς το στοκ δεν αποθηκεύεται στατικά σε κάθε προϊόν αλλά προκύπτει από το άθροισμα των συναλλαγών που το αφορούν πρέπει να υπολογιστεί με επιπλέον. Αυτό είναι το άθροισμα των εγγραφών του stock που έχουν σαν ξένο κλειδί στο πεδίο product_id το id του τρέχοντος προϊόντος.

```
SELECT sum(stock)
FROM `transfer`
WHERE product_id = ".$product['id']
```

Όπως και στα προηγούμενα αντικείμενα, ανάλογα με το αν ο χρήστης είναι διαχειριστής ή όχι εμφανίζονται τα ενεργά ή όχι κουμπιά για τις ενέργειες.

Τέλος στο κάτω μέρος της σελίδας υπάρχει ένας σύνδεσμος για την εξαγωγή προϊόντος σαν έγγραφο xls.

```
<a class="excel" href="product_export_excel.php">
  Εξαγωγή σε Excel
</a>
```

#### *4.7.6.2 Αρχείο product_delete_action.php*

Το αρχείο product_delete_action.php είναι το αρχείο που καλείται όταν ο χρήστης επιλέξει να σβήσει ένα προϊόν. Δεν έχει γραφική απεικόνιση αλλά εκτελεί τις κατάλληλες ενέργειες και ξαναστέλνει τον χρήστη στη λίστα προϊόντων. Κατά συνέπεια περιέχει μόνο το αρχείο με τις συναρτήσεις της βάσης δεδομένων.

Η διαγραφή ενός προϊόντος γίνεται με την κατάλληλη εντολή SQL που το διαγράφει βάση του πρωτεύοντος κλειδιού του.

```
DELETE FROM `warehouse`.`product`
WHERE `product`.`id` = $id
```

Όμως στον πίνακα transfers αυτό το κλειδί χρησιμοποιείται σαν ξένο κλειδί ώστε να φαίνεται σε πιο προϊόν αναφέρεται μία συναλλαγή. Κατά συνέπεια με τη διαγραφή του προϊόντος παραμένουν εγγραφές που περιέχουν ορφανά ξένα κλειδιά, κάτι που δεν είναι σωστό, καθώς η βάση δεν είναι πια συνεπής.

Επιλέξαμε με την διαγραφή ενός προϊόντος να διαγράφουμε και όλες τις σχετικές συναλλαγές, ώστε να αποφύγουμε αυτό το πρόβλημα.

```
DELETE FROM `warehouse`.`transfer`
WHERE `transfer`.`product_id` = $id
```

Έπειτα αποθηκεύεται το κατάλληλο μήνυμα και ο χρήστης ανακατευθύνεται στην λίστα.

#### *4.7.6.3 Αρχείο product_add.php*

Το αρχείο product_add.php περιέχει τη φόρμα εισαγωγής προϊόντων. Όπως και στα προηγούμενα η φόρμα χρησιμοποιείται και για την εισαγωγή και για την τροποποίηση των προϊόντων ανάλογα με τις παραμέτρους που έχουν δοθεί κατά την κλήση της. Πρόκειται για κανονική σελίδα οπότε περιέχει header, footer και το αρχείο της βάση για να φορτώνει δεδομένα.

Συγκεκριμένα αν πρόκειται για τροποποίηση προϊόντος τότε ο σύνδεσμος στην λίστα προϊόντων έχει κατασκευαστεί ώστε να περνάει και μία παράμετρο id με το κλειδί του

προϊόντος. Αν έχει οριστεί φορτώνονται τα στοιχεία του. Έπειτα υπάρχει ο κώδικας για τον έλεγχο των πεδίων της φόρμας.

Η διαφορά που έχει αυτή η φόρμα σε σχέση με τις προηγούμενες είναι το πεδίο επιλογής κατηγορίας. Αυτό δεν έχει στατική τιμή αλλά είναι απαραίτητο να βρίσκει τις κατηγορίες από τη βάση δεδομένων.

Οι κατηγορίες παρουσιάζονται μέσα σε ένα drop down menu, δηλαδή ένα πεδίο select. Μέσα σε αυτό διατρέχουμε όλες τις κατηγορίες που υπάρχουν στη βάση και για κάθε μία δημιουργείται ένα πεδίο option, που έχει σαν value το κλειδί κάθε κατηγορίας, καθώς αυτό μας ενδιαφέρει να στέλνεται για την εισαγωγή, ενώ σαν περιεχόμενο έχει το όνομα της κατηγορίας.

```
<select name="category_id">
<?php

    $q = "SELECT * FROM `category`";
    $categories = query($q);

    foreach($categories as $category){
        if($id == 0 || $category_id != $category['id'])
            echo '<option
value="'.$category['id'].'">'.$category['name'].'</option>';
        else
            echo '<option value="'.$category['id'].'"
selected>'.$category['name'].'</option>';
    }
?>
</select>
```

Η συνθήκη ελέγχει αν υπάρχει κάποια προεπιλεγμένη κατηγορία, το οποίο συμβαίνει κατά την τροποποίηση. Αυτό χρειάζεται γιατί στην τροποποίηση το μενού πρέπει να έχει επιλεγμένη την τιμή της, που δεν γίνεται απλά βάζοντας την ιδιότητα value, όπως στα άλλα πεδία της HTML. Όταν διατρέχουμε το πεδίο option ελέγχουμε αν το id της τρέχουσας κατηγορίας είναι το id την κατηγορίας στην οποία ανήκει το προϊόν που τροποποιούμε και αν ναι βάζουμε την ιδιότητα selected σε αυτό.

Έπειτα η φόρμα κατασκευάζεται και γεμίζει όπως προηγουμένως. Υπάρχει ένα hidden πεδίο id που έχει την τιμή του κλειδιού του προϊόντος που τροποποιούμε ή 0 αν είναι νέο προϊόν.

#### *4.7.6.4 Αρχείο product_add_action.php*

Το αρχείο product_add_action.php αναλαμβάνει την επεξεργασία των στοιχείων που στέλνονται από την φόρμα εισαγωγής προϊόντων και την εισαγωγή τους στη βάση.

Αν η φόρμα έχει στείλει το πεδίο id με τιμή μηδέν τότε πρόκειται για νέα εισαγωγή, οπότε εκτελεί μία εντολή INSERT.

```
INSERT INTO `warehouse`.`product` (`id`, `category_id`,
`name`,
    `description`, `price_buy`, `price_sell`)
VALUES (NULL, '$category_id', '$name', '$description',
    '$price_buy', '$price_sell');
```

Αν αντίθετα το κλειδί δεν είναι μηδέν τότε πρόκειται για τροποποίηση και τρέχει μία εντολή UPDATE για το προϊόν που έχει το συγκεκριμένο κλειδί.

```
UPDATE `warehouse`.`product`
SET `category_id` = '$category_id', `name` = '$name',
    `description` = '$description', `price_buy` =
'$price_buy',
    `price_sell` = '$price_sell'
WHERE `product`.`id` = $id;
```

Τέλος αποθηκεύεται το κατάλληλο μήνυμα και ο χρήστης ανακατευθύνεται στην λίστα.

#### *4.7.6.5 Αρχείο product_export_excel.php*

Η εφαρμογή υποστηρίζει την δημιουργία ενός αρχείου Excel με όλα τα προϊόντα. Αυτό γίνεται από το αρχείο product_export_excel.php, στο οποίο δείχνει ένας σύνδεσμος στο τέλος της λίστας με τα προϊόντα.

Η δημιουργία του αρχείο γίνεται με την βιβλιοθήκη PHPExcel, τα αρχεία της οποίας έχουμε αντιγράψει στον φάκελο includes. Αρχικά περιλαμβάνουμε την κατάλληλη βιβλιοθήκη στον κώδικα, καθώς και το db.php για την επικοινωνία με την βάση.

```
require_once 'include/phpExcel/PHPExcel.php';
require_once('include/db.php');
```

Το αρχείο που θα σταλεί πίσω δεν είναι ένα αρχείο HTML που θα το δει στον browser του, αλλά ένα αρχείο xls, που θα το κατεβάσει. Για αυτό τον λόγο χρειάζεται να σταλθούν οι κατάλληλοι headers ώστε ο browser να το αντιμετωπίσει κατάλληλα.

```
header('Content-Type: application/vnd.ms-excel');
header('Content-Disposition:
attachment;filename="transfers.xls"');
header('Cache-Control: max-age=0');
```

Μετά φορτώνουμε όλα τα προϊόντα από τη βάση με τα ονόματα της κατηγορίας στην οποία ανήκει το καθένα.

```
SELECT `product`.`id`, `product`.`name`,
`product`.`description`,
    `product`.`price_buy`, `product`.`price_sell`,
    `category`.`name` AS category_name
FROM `product`, `category`
WHERE `product`.`category_id` = `category`.`id`
```

Η δημιουργία ενός εγγράφου γίνεται δημιουργώντας ένα αντικείμενο PHPExcel.

```
$objPHPExcel = new PHPExcel();
```

Σε αυτό βάζουμε τις κατάλληλες τιμές στον τίτλο και στο θέμα του.

```
$objPHPExcel->getProperties()
            ->setTitle("Στοιχεία κίνησης προϊόντων")
            ->setSubject("Στοιχεία κίνησης προϊόντων");

$objPHPExcel->getActiveSheet()
            ->setTitle('Κίνηση προϊόντων');
```

Έπειτα βάζουμε στα κατάλληλα κελιά τις επικεφαλίδες. Τα κελιά μετρούνται με τον συμβολισμό του Excel, όπου χρησιμοποιείται ένα λατινικό γράμμα A, B, C κτλ. για την στήλη και ένα νούμερο με την γραμμή, όπου η αρίθμηση των γραμμών ξεκινάει από 1.

```
$objPHPExcel->setActiveSheetIndex(0)
            ->setCellValue('A1', 'Όνομα')
            ->setCellValue('B1', 'Περιγραφή')
            ->setCellValue('C1', 'Κατηγορία')
            ->setCellValue('D1', 'Τιμή αγοράς')
            ->setCellValue('E1', 'Τιμή Πώλησης')
            ->setCellValue('F1', 'Stock');
```

Μετά διατρέχουμε τα προϊόντα που διαλέξαμε προηγουμένως. Πέρα από τη for χρειαζόμαστε και έναν βοηθητικό μετρητή i για να μετράμε τις γραμμές του Excel. Αφού οι

γραμμές ξεκινάνε από το 1 και στην πρώτη γραμμή έχουμε βάλει τις επικεφαλίδες το i έχει αρχική τιμή 2.

Για κάθε προϊόν βρίσκουμε και το stock του με την παρακάτω εντολή.

```
SELECT sum(stock)
FROM `transfer`
WHERE product_id = ".$product['id']
```

Έπειτα εισάγουμε τα στοιχεία του στις κατάλληλες στήλες του εγγράφου και στη γραμμή i και τέλος αυξάνουμε το i.

```
$i = 2;
foreach($products as $product){
    $stock = query_single("SELECT sum(stock) FROM `transfer` WHERE
product_id = ".$product['id']);

    $objPHPExcel->setActiveSheetIndex(0)
            ->setCellValue("A$i", $product['name'])
            ->setCellValue("B$i", $product['description'])
            ->setCellValue("C$i", $product['category_name'])
            ->setCellValue("D$i", $product['price_buy'])
            ->setCellValue("E$i", $product['price_sell'])
            ->setCellValue("F$i", $stock);
    $i++;
}
```

Τέλος χρειάζεται να γράψουμε το έγγραφο στην έξοδο, ώστε να αποσταλεί στον χρήστη.

```
$objWriter = PHPExcel_IOFactory::createWriter($objPHPExcel,
'Excel5');
$objWriter->save('php://output');
```

Προσδιορίζουμε την Έκδοση Excel5 για μεγαλύτερη συμβατότητα, αλλά θα μπορούσαμε να είχαμε δημιουργήσει ένα έγγραφο σε πιο σύγχρονη έκδοση. Καθώς όμως το έγγραφό μας δεν χρησιμοποιεί κάποια εξεζητημένη δυνατότητα του Excel δεν υπάρχει λόγος για κάτι τέτοιο.

### 4.7.7 Διαχείριση κινήσεων προϊόντων

Καθώς μας ενδιαφέρει να κρατάμε πληροφορίες για κάθε συναλλαγή που γίνεται για ένα προϊόν, κρατάμε μία διαφορετική οντότητα transfer. Ένα αντικείμενο transfer μπορεί να

είναι μία αγορά ή μία πώληση, ανάλογα με αν έχει αρνητική ή θετική τιμή στο πεδίο της ποσότητας.

Οι σχετικές ενέργειες γίνονται στα παρακάτω αρχεία.

#### *4.7.7.1 Αρχείο transfer_list.php*

Το αρχείο transfer_list.php δημιουργεί την λίστα με όλες τις κινήσεις των προϊόντων. Είναι ένα αρχείο που βλέπει ο χρήστης, οπότε συμπεριλαμβάνει το header, το footer και το αρχείο της βάσης.

Αρχικά δημιουργείται ένας πίνακας HTML και βάζουμε τις επικεφαλίδες που θέλουμε.

Έπειτα επιλέγουμε από τη βάση όλες τις εγγραφές του πίνακα transfer. Για κάθε εγγραφή επιλέγουμε και το όνομα του προϊόντος που έχει το ίδιο κλειδί με το ξένο κλειδί product_id.

```
SELECT `transfer`.`id`, `stock`, `date`,
    `name`, `price_buy`, `price_sell`
FROM `transfer`, `product`
WHERE `transfer`.`product_id` = `product`.`id`
```

Έπειτα διατρέχουμε τα αποτελέσματα και δημιουργούμε τα κελιά του πίνακα. Όπως και στα προηγούμενα αντικείμενα, ανάλογα με το αν ο χρήστης είναι διαχειριστής ή όχι εμφανίζονται τα ενεργά ή όχι κουμπιά για τις ενέργειες.

Στο κάτω μέρος της σελίδας υπάρχει ένας σύνδεσμος για την μαζική εξαγωγή σαν έγγραφο xls.

```
<a class="excel" href="transfer_export_excel.php">
  Εξαγωγή σε Excel
</a>
```

#### *4.7.7.2 Αρχείο transfer_delete_action.php*

Το αρχείο transfer_delete_action.php παίρνει σαν παράμετρο το id μίας συναλλαγής και τη σβήνει.

```
DELETE FROM `warehouse`.` transfer`
WHERE ` transfer`.`id` = $id
```

Έπειτα αποθηκεύεται το κατάλληλο μήνυμα και ο χρήστης ανακατευθύνεται στη λίστα με τις κατηγορίες.

#### *4.7.7.3 Αρχείο transfer_add.php*

Το αρχείο transfer_add.php περιέχει τον κώδικα προσθήκης μίας νέας αγοράς ή πώλησης. Είναι ένα αρχείο που βλέπει ο χρήστης, οπότε συμπεριλαμβάνει το header, το footer και το αρχείο της βάσης.

Όπως και στα προηγούμενα η φόρμα χρησιμοποιείται και για την εισαγωγή και για την τροποποίηση των αντικειμένων ανάλογα με τις παραμέτρους που έχουν δοθεί κατά την κλήση της. Αν ο χρήστης πατήσει στο κουμπί τροποποίησης από τη λίστα θα δοθεί και μία μεταβλητή GET id με το κλειδί του εν λόγου αντικειμένου.

Αν πρόκειται για τροποποίησης φορτώνεται η κατάλληλη μεταφορά που έχει αυτό το κλειδί και εισάγεται ο κώδικας ελέγχου της φόρμας. Έπειτα δημιουργείται η φόρμα.

Όπως και για τις κατηγορίες τα προϊόντα εμφανίζονται σε ένα στοιχείο select της HTML αφού φορτωθούν από την βάση. Αν πρόκειται για τροποποίησης προϊόντος τότε στο προϊόν που έχει το κλειδί στο οποίο αναφέρεται η μεταφορά προστίθεται η ιδιότητα selected για να είναι επιλεγμένο.

```
<select name="product_id">
<?php

    $q = "SELECT * FROM `product`";
    $products = query($q);

    foreach($products as $product){
        if($id == 0 || $product_id != $product['id'])
            echo '<option
value="'.$product['id'].'">'.$product['name'].'</option>';
        else
            echo '<option value="'.$product['id'].'"
selected>'.$product['name'].'</option>';
    }
?>
</select>
```

Το πεδίο της ποσότητας είναι τύπου number. Αυτός είναι ένας νέος τύπος πεδίων της HTML5, όπου παρουσιάζει δύο κουμπιά που ο χρήστης μπορεί να πατήσει για να αυξομειώσει την ποσότητα. Επιτρέπονται αρνητικές ποσότητες για αν υποδηλώσουν ότι πρόκειται για πώληση.

```
<label>Ποσότητα:</label><input type="number" name="stock"
class="required number" <?php if($id != 0) echo
"value=\"$stock\""; ?> />
```

Ομοίως το πεδίο της ημερομηνίας που έγινε η συναλλαγή είναι ένα πεδίο τύπου date, που είναι και αυτό ένα νέο πεδίο της HTML5. Όταν ο χρήστης το πατήσει εμφανίζεται ένα ημερολόγιο από το οποίο μπορεί να επιλέξει την ημερομηνία και την ώρα της συναλλαγής.

```
<label>Ημερομηνία:</label><input type="date" name="date"
class="required" <?php if($id != 0) echo "value=\"$date\""; ?> />
```

Έπειτα η φόρμα κατασκευάζεται και γεμίζει όπως προηγουμένως. Υπάρχει ένα hidden πεδίο id που έχει την τιμή του κλειδιού του προϊόντος που τροποποιούμε ή 0 αν είναι νέο προϊόν.

#### *4.7.7.4 Αρχείο transfer_add_action.php*

Το αρχείο transfer_add_action.php αναλαμβάνει την αποθήκευση των στοιχείων που στέλνονται από την φόρμα εισαγωγής προϊόντων και την εισαγωγή τους στη βάση.

Αρχικά χρειάζεται να αλλάξει η μορφή της ημερομηνίας που στέλνεται από το πεδίο date. Αυτό έχει πρώτα τον μήνα, μετά τη μέρα και τέλος το έτος χωρισμένα με παύλες. Δημιουργούμε τρια διαφορετικά string php με τη συνάρτηση explode.

```
list($month, $day, $year) = explode("-", $date);
```

Η μορφή που θέλουμε για να κάνουμε την εισαγωγή σαν μία ημερομηνία MySQL είναι πρώτα ο χρόνος, μετά ο μήνας και τέλος η μέρα, οπότε τα συνδυάζουμε σε ένα καινούργιο string.

```
$date = "$year-$month-$day";
```

Επιπλέον ελέγχεται η τιμή του πεδίου stock. Αν η ποσότητα της συναλλαγής είναι 0 τότε δεν έχει νόημα, οπότε αποθηκεύεται το κατάλληλο μήνυμα και ο χρήστης ανακατευθύνεται στη λίστα συναλλαγών.

```
if($stock == 0){
    $_SESSION['errors'] = 'Η ποσότητα δεν μπορεί να είναι 0.';
    header("Location: transfer_add.php");
    return;
}
```

Αν η φόρμα έχει στείλει το πεδίο id με τιμή μηδέν τότε πρόκειται για νέα εισαγωγή, οπότε εκτελεί μία εντολή INSERT.

```
INSERT INTO `warehouse`.`transfer` (`id` ,`product_id` ,`stock`
,`date`)
VALUES (NULL , '$product_id', '$stock', '$date');
```

Αν αντίθετα το κλειδί δεν είναι μηδέν τότε πρόκειται για τροποποίηση και τρέχει μία εντολή UPDATE για το προϊόν που έχει το συγκεκριμένο κλειδί.

```
UPDATE `warehouse`.`transfer`
SET `product_id` = '$product_id', `stock` = '$stock', `date` =
'$date'
WHERE `transfer`.`id` = $id;
```

Τέλος αποθηκεύεται το κατάλληλο μήνυμα και ο χρήστης ανακατευθύνεται στην λίστα.

#### *4.7.7.5 Αρχείο transfer_export_excel.php*

Το αρχείο transfer_export_excel.php αναλαμβάνει την εξαγωγή όλων των συναλλαγών σε ένα αρχείο Excel. Ο χρήστης πηγαίνει σε αυτό από τον κατάλληλο σύνδεσμο στο τέλος της λίστας συναλλαγών.

Η δημιουργία του αρχείο γίνεται πάλι με την βιβλιοθήκη PHPExcel. Αρχικά περιλαμβάνουμε την κατάλληλη βιβλιοθήκη στον κώδικα, καθώς και το db.php για την επικοινωνία με την βάση.

```
require_once 'include/phpExcel/PHPExcel.php';
require_once('include/db.php');
```

Ορίζουμε πως πρόκειται για ένα αρχείο που δεν είναι HTML για να εμφανιστεί στον browse, αλλά ένα αρχείο xls, που πρέπει να κατέβει στον υπολογιστή του χρήστη.

```
header('Content-Type: application/vnd.ms-excel');
header('Content-Disposition:
attachment;filename="transfers.xls"');
header('Cache-Control: max-age=0');
```

Μετά φορτώνουμε όλα τα transfer από τη βάση με τα ονόματα των προϊόντων που αναφέρονται.

```
SELECT `transfer`.`id`, `product`.`id` AS product_id,
    `stock`, `date`, `name`, `price_buy`, `price_sell`
FROM `transfer`, `product`
WHERE `transfer`.`product_id` = `product`.`id`
```

Η δημιουργία ενός εγγράφου γίνεται δημιουργώντας ένα αντικείμενο PHPExcel με τις κατάλληλες ιδιότητες.

```
$objPHPExcel = new PHPExcel();
$objPHPExcel->getActiveSheet()
            ->setTitle('Κίνηση προϊόντων');
$objPHPExcel->getProperties()
            ->setTitle("Στοιχεία κίνησης προϊόντων")
            ->setSubject("Στοιχεία κίνησης προϊόντων");
```

Σε αυτό βάζουμε τις κατάλληλες τιμές στον τίτλο και στο θέμα του.

```
$objPHPExcel->getProperties()
            ->setTitle("Στοιχεία κίνησης προϊόντων")
            ->setSubject("Στοιχεία κίνησης προϊόντων");

$objPHPExcel->getActiveSheet()
            ->setTitle('Κίνηση προϊόντων');
```

Έπειτα βάζουμε στα κατάλληλα κελιά τις επικεφαλίδες.

```
$objPHPExcel->setActiveSheetIndex(0)
             ->setCellValue('A1', 'Κωδικός')
             ->setCellValue('B1', 'Ποσότητα')
             ->setCellValue('C1', 'Ημερομηνία')
             ->setCellValue('D1', 'Κόστος')
             ->setCellValue('E1', 'Προϊόν');
```

Μετά διατρέχουμε τα προϊόντα που διαλέξαμε προηγουμένως. Έχουμε την for που διατρέχει τις συναλλαγές και έναν βοηθητικό μετρητή i για να μετράμε τις γραμμές του

Excel, το οποίο ξεκινάει από το δύο αφού οι γραμμές ξεκινάνε από το 1 και στην πρώτη γραμμή έχουμε βάλει τις επικεφαλίδες.

Εισάγουμε τα στοιχεία του στις κατάλληλες στήλες του εγγράφου και στη γραμμή i και τέλος αυξάνουμε το i.

```
$i = 2;
foreach($transfers as $transfer){
    $objPHPExcel->setActiveSheetIndex(0)
            ->setCellValue("A$i", $transfer['product_id'])
            ->setCellValue("B$i", $transfer['stock'])
            ->setCellValue("C$i", $transfer['date'])
            ->setCellValue("D$i", $cost)
            ->setCellValue("E$i", $transfer['name']);
    $i++;
}
```

Τέλος χρειάζεται να γράψουμε το έγγραφο στην έξοδο, ώστε να αποσταλεί στον χρήστη.

```
$objWriter = PHPExcel_IOFactory::createWriter($objPHPExcel,
'Excel5');
$objWriter->save('php://output');
```

Το αρχείο τελειώνει αυτόματα με την αποθήκευσή του στην έξοδο.

### 4.7.8 Δημιουργία γραφημάτων

Μία επιπλέον λειτουργία της εφαρμογής είναι η δημιουργία γραφημάτων με τις μεταβολές της ποσότητας κάθε προϊόντος. Για την κατασκευή των γραφημάτων χρησιμοποιείται η βιβλιοθήκη jpgraph.

Χρειάστηκαν τα αρχεία που παραθέτουμε παρακάτω.

#### *4.7.8.1 Αρχείο stock_graph.php*

Το αρχείο stock_graph.php είναι υπεύθυνο για την παρουσίαση των γραφημάτων. Ο χρήστης μπορεί να το ανοίξει από τον σύνδεσμο στο κυρίως μενού, και καθώς είναι αρχείο που εμφανίζεται στον browser περιλαμβάνει τα header και footer, καθώς και το db.

Το αρχείο παρουσιάζει μία φόρμα με ένα drop down menu με προϊόντα. Ανάλογα με το αν ο χρήστης έχει διαλέξει προϊόν ή όχι μπορεί να εμφανίζει μια εικόνα με το κατάλληλο γράφημα. Ο διαχωρισμός γίνεται ανάλογα με την τιμή της μεταβλητής id. Αν δεν έχει δηλωθεί έχει τιμή 0, αλλιώς παίρνει την τιμή που έχει περαστεί στις παραμέτρους GET.

Αν έχει περαστεί δημιουργούνται δύο σύνδεσμοι για την εμφάνιση της εικόνας και τη δημιουργία του εγγράφου PDF.

```
if($id != 0){
    echo '<div class="graph"><img
src="graph_product_stock.php?id='.$id.'"></div>';
    echo '<a download="Report.png" class="pdf"
href="stock_graph_export_pdf.php?product_id='.$id.'">Αποθήκευση</a>'
;

}
```

Έπειτα δημιουργείται η φόρμα με τα προϊόντα, δηλαδή φορτώνονται τα προϊόντα από τη βάση και δημιουργούνται τα κατάλληλα αντικείμενα τύπου option, όπως και στις προηγούμενες περιπτώσεις. Αν φαίνεται το γράφημα κάποιου προϊόντος τότε επιλέγεται και στη λίστα.

Η φόρμα έχει σαν action τον εαυτό της. Η διαφορά είναι πως όταν ξαναφορτωθεί θα έχει περάσει το πεδίο id και θα εμφανιστούν η εικόνα του γραφήματος και ο σύνδεσμος με το αρχείο pdf.

#### *4.7.8.2 Αρχείο graph_product_stock.php*

Το αρχείο graph_product_stock.php δημιουργεί το γράφημα με την κίνηση ενός προϊόντος. Σε αντίθεση με τα προηγούμενα αρχεία που έχουμε δημιουργήσει δεν επιστρέφει κώδικα HTML αλλά μία εικόνα.

Το αρχείο χρησιμοποιεί την βιβλιοθήκη jpgraph, οπότε χρειάζεται να συμπεριλάβουμε το κατάλληλο αρχείο.

```
require_once ('include/jpgraph/jpgraph.php');
```

Χρειάζονται δύο επιπλέον includes για το γράφημα που θέλουμε να κατασκευάσουμε. Το ένα είναι το jpgraph_line.php, που δημιουργεί γραφήματα τύπου γραμμής, σαν αυτό που θέλουμε, και το δεύτερο είναι το jpgraph_date.php που αναλαμβάνει την επεξεργασία δεδομένων τύπου date.

```
require_once ('include/jpgraph/jpgraph_line.php');
require_once( "include/jpgraph/jpgraph_date.php" );
```

Το αρχείο καλείται με μία παράμετρο id, που είναι το κλειδί του προϊόντος που θέλουμε να παρουσιάσουμε. Από τη βάση φορτώνουμε αρχικά το όνομά του.

Έπειτα διαλέγουμε όλες τις ημερομηνίες χρησιμοποιούμε την παρακάτω εντολή:

```
SELECT `date`
FROM `transfer`
WHERE product_id = $id
ORDER BY `date` ASC
```

Για να επιλέξουμε τα κατάλληλα στοιχεία που θα χρησιμοποιήσουμε στα γραφήματα χρειαζόμαστε δύο πίνακες, έναν με όλες τις ημερομηνίες ώστε να μπει στον οριζόντιο άξονα και έναν με τις ποσότητες που θα μπει στον κάθετο άξονα. Κάθε ημερομηνία πρέπει να είναι σε δευτερόλεπτα, δηλαδή ένας ακέραιος, και κάθε ποσότητα είναι το άθροισμα των συναλλαγών μέχρι αυτή την ημερομηνία.

Αρχικά διατρέχουμε δηλώνουμε δύο πίνακες που θα κρατάνε τις ποσότητες και τις ημερομηνίες σαν ακεραίους, τους stock και dates.

```
$stock = array();
$dates = array();
```

Τα περιεχόμενα τους θα βρίσκονται σε αντίστοιχες θέσεις, δηλαδή η πρώτη θέση του πίνακα με τις ποσότητες θα έχει την ποσότητα του προϊόντος στην ημερομηνία που βρίσκεται στην πρώτη θέση του πίνακα με τις ημερομηνίες.

Μετά διατρέχουμε τις ημερομηνίες που πήραμε από τη βάση, οι οποίες βρίσκονται στη μορφή που τις επιστρέφει η MySQL, σαν έτος, μήνας και ημέρα χωρισμένα με παύλες. Χρειαζόμαστε αυτή τη μορφή για τα παρακάτω υποερωτήματα.

```
foreach($dates_res as $ar){
    ...
}
```

Μέσα σε αυτό το loop επιλέγουμε το άθροισμα όλων των tranfers που έχουν γίνει για το συγκεκριμένο προϊόν μέχρι αυτή την ημερομηνία.

```
$q = "SELECT sum(stock) FROM `transfer` WHERE product_id = $id AND
     `date` <= '".$ar['date']."'";
$sum = query_single($q);
```

Έπειτα αποθηκεύουμε το άθροισμα στον πίνακα, στην επόμενη θέση του, stock και την ημερομηνία στον πίνακα dates, αφού την μετατρέψουμε σε ακέραιο με την συνάρτηση strtotime.

```
    array_push($stock, $sum);
    array_push($dates, strtotime($ar['date']));
```

Αφού έχουμε τα δεδομένα στους πίνακες stock και dates μπορούμε να κατασκευάσουμε το γράφημα. Κατασκευάζουμε ένα νέο αντικείμενο τύπου Graph στο οποίο βάζουμε το μέγεθος της εικόνας και του δίνουμε τον τίτλο. Επιπλέον δίνουμε και κάποια περιθώρια για να μην κολλάει με τον τίτλο και τις άκρες της εικόνας.

```
    $graph = new Graph(500,400);
    $graph->title->Set("Κίνηση προϊόντος - $name");
    $graph->SetMargin(40,40,30,130);
```

Για να εμφανίζονται οι ακέραιοι σαν ημερομηνίες πρέπει να ορίσουμε το κατάλληλο εργαλείο εμφάνισης δεδομένων του άξονα, που είναι το παρακάτω:

```
    $graph->SetScale('datlin');
```

Έπειτα ορίζουμε ότι στον οριζόντιο άξονα θέλουμε οι ημερομηνίες να περιστρέφουν κατά 90 μοίρες για να φαίνονται κάθετα και να μην πέφτουν η μία πάνω στην άλλη.

```
    $graph->xaxis->SetLabelAngle(90);
```

Τέλος ορίζουμε πως θέλουμε στον οριζόντιο άξονα να φαίνονται ετικέτες με την ημερομηνία κάθε 5 μέρες. Αφού τα δεδομένα είναι σε δευτερόλεπτα οι πέντε μέρες είναι το γινόμενο 5 επί 24 ώρες επί 60 λεπτά επί 60 δευτερόλεπτα.

```
    $graph->xaxis->scale->ticks->Set(24*60*60*5);
```

Μετά κατασκευάζουμε την γραμμή του διαστήματος από τα δεδομένα που έχουμε παραπάνω στους δύο πίνακες και την κάνουμε μπλε. Αυτή είναι ακόμη ανεξάρτητη από το γράφημα.

```
    $lineplot=new LinePlot($stock, $dates);

    $lineplot->SetColor('blue');
```

Τέλος προσθέτουμε αυτή τη γραμμή στο γράφημα και λέμε να ζωγραφιστεί, δηλαδή να σταλεί στην έξοδο.

```
$graph->Add($lineplot);
$graph->Stroke();
```

Αυτό αποστέλλεται σαν εικόνα, που είναι το πεδίο src του στοιχείου img που έχουμε ορίσει παραπάνω.

#### *4.7.8.3 Αρχείο stock_graph_export_pdf.php*

Το αρχείο stock_graph_export_pdf.php δημιουργεί ένα αρχείο PDF που μπορεί να κατεβάσει ο χρήστης με την εικόνα του γραφήματος. Ο σύνδεσμος σε αυτή τη σελίδα δημιουργείται από την φόρμα εμφάνισης του γραφήματος όταν ο χρήστης έχει επιλέξει κάποιο προϊόν και περιέχει και μία μεταβλητή GET με το κλειδί του προϊόντος.

Για την δημιουργία του χρησιμοποιήσαμε την βιβλιοθήκη mPDF, τα αρχεία της οποίας έχουμε αντιγράψει στον φάκελο include. Για να λειτουργήσει χρειάζεται να κατασκευάσουμε ένα απόσπασμα κώδικα HTML, τον οποίο θα μετατρέψει σε PDF.

Για την δημιουργία του περιεχομένου κατασκευάζουμε ένα div που κεντράρει το περιεχόμενό του μέσα στο οποίο βάζουμε ένα στοιχείο img που δείχνει την εικόνα του γραφήματος με το κλειδί του προϊόντος που δόθηκε.

```
$html .= ‘<div style=”text-align:center;">
            <img src="graph_product_stock.php?id='.$id.'">
         </div>';
```

Έπειτα δημιουργούμε ένα αντικείμενο τύπου mPDF. Σε αυτό δίνουμε σαν είσοδο το string με τον κώδικα HTML που δημιουργήσαμε προηγουμένως και γράφουμε το περιεχόμενο στην έξοδο.

```
$mpdf=new mPDF();

$mpdf->WriteHTML($html);
$mpdf->Output();
```

Η έξοδος είναι ένα αρχείο PDF το οποίο ο browser του χρήστη του ζητάει να κατεβάσει.

### 4.7.9 Μαζική εισαγωγή συναλλαγών

Η εφαρμογή δίνει την δυνατότητα εισαγωγής πολλών συναλλαγών ταυτόχρονα διαβάζοντας τα περιεχόμενα ενός αρχείου Excel.

Για την υλοποίηση χρησιμοποιούνται τα δύο αρχεία που αναλύονται παρακάτω.

#### *4.7.9.1 Αρχείο transfer_import.php*

Το αρχείο transfer_import.php είναι μια φόρμα μέσω της οποίας ο χρήστης μπορεί να ανεβάσει ένα αρχείο Excel στον server. Ο χρήστης μπορεί να πάει σε αυτή τη σελίδα επιλέγοντας την κατάλληλη επιλογή από το μενού. Περιέχει το header και το footer, αλλά αφού δεν φορτώνει δεδομένα από τη βάση δεν χρειάζεται το db.php.

Πρόκειται για μία στατική σελίδα που περιέχει απλά μία φόρμα. Όμως αφού η φόρμα χρησιμοποιείται για να γίνει upload ένα αρχείο χρειάζεται ένα πεδίο τύπου file και κάποιες διαφορετικές επιλογές.

```
<form method="post" action="transfer_import_action.php"
      enctype="multipart/form-data">

    <fieldset><legend>Μαζική εισαγωγή</legend>

    <label>Αρχείο Excel:</label>
    <input name="file" type="file" />

    <input type="submit" value="Υποβολή" />

    </fieldset>
</form>
```

Η ιδιότητα enctype πρέπει να είναι δηλωμένη και να περιέχει την τιμή multipart/form-data, αλλιώς δεν πρόκειται να σταλεί σωστά το αρχείο.

#### *4.7.9.2 Αρχείο transfer_import_action.php*

Το αρχείο transfer_import_action.php αναλαμβάνει να πάρει το αρχείο, να το ανοίξει και να αποθηκεύσει τα περιεχόμενα στην βάση. Και εδώ χρησιμοποιείται η βιβλιοθήκη PHPExcel. Αφού θα χρειαστεί να αποθηκεύσει δεδομένα στη βάση πρέπει να περιέχει το αρχείο db.php.

Για να λειτουργήσει σωστά το αρχείο Excel που θα ανεβάσει ο χρήστης πρέπει να έχει 3 στήλες που θα περιέχουν τον κωδικό του προϊόντος στο οποίο αναφέρεται η συναλλαγή, την ποσότητα και την ημερομηνία που έγινε.

Αρχικά ελέγχεται αν υπάρχει κάποιο πρόβλημα με το ανέβασμα του αρχείου, οπότε σώζεται το κατάλληλο μήνυμα και ο χρήστης ανακατευθύνεται στη λίστα συναλλαγών. Τα στοιχεία του αρχείου που ανεβάσαμε βρίσκονται στη μεταβλητή $_FILES["file"]. Πρόκειται για έναν πίνακα, που στο μέλος error περιέχει τυχών μηνύματα, ή έχει τιμή μηδέν αν όλα πήγαν καλά.

```
if ($_FILES["file"]["error"] > 0) {
    $_SESSION['errors'] = 'Παρουσιάστηκε πρόβλημα: ' .
       $_FILES["file"]["error"];
    header("Location: transfer_list.php");
    return;
}
```

Έπειτα χρειάζεται να εισάγουμε το κατάλληλο αρχείο από τη βιβλιοθήκη PHPExcel, το οποίο αναλαμβάνει το άνοιγμα αρχείων.

```
require_once 'include/phpExcel/PHPExcel/IOFactory.php';
```

Μετά παίρνουμε το όνομα του αρχείου. Τα αρχεία που ανεβάζει ο χρήστης γράφονται προσωρινά στον χώρο προσωρινής αποθήκευσης του server με ένα τυχαίο όνομα. Αυτό βρίσκεται στην μεταβλητή $_FILES["file"]["tmp_name"].

Αυτό το αρχείο το ανοίγουμε με την PHPExcel.

```
$inputFileName = $_FILES["file"]["tmp_name"];
$objPHPExcel = PHPExcel_IOFactory::load($inputFileName);
```

Έπειτα εξάγουμε τα περιεχόμενα του πρώτου φύλλου εργασίας σαν πίνακα σαν πίνακα για να μπορέσουμε να τα χειριστούμε με εντολές PHP.

```
$sheetData = $objPHPExcel->getActiveSheet()
                         ->toArray(null,true,true,true);
```

Δηλώνουμε μία μεταβλητή count για να μετρήσουμε τις εισαγωγές και διατρέχουμε τις γραμμές του πίνακα, αυξάνοντας κατάλληλα το count. Αν και δεν εξυπηρετεί προγραμματιστικά θα τη χρησιμοποιήσουμε παρακάτω για να ενημερώσουμε τον χρήστη για το πλήθος των εισαγωγών.

```
    $count = 0;
    foreach($sheetData as $row) {

        ...

        $count++;
    }
```

Για κάθε μία γραμμή παίρνουμε τα περιεχόμενα των στηλών που χρειαζόμαστε και τα εισάγουμε σε μεταβλητές PHP.

```
        $date = $row['C'];
        $stock = $row['B'];
        $product_id = $row['A'];
```

Η ημερομηνία δίνεται με τη μορφή του Excel, δηλαδή μήνας, ημέρα, χρόνος, χωρισμένες με παύλες. Την φέρνουμε στη μορφή που υποστηρίζεται από την MySQL κόβοντας και ξανακολλώντας το string.

```
    list($month, $day, $year) = explode("-", $date);
    $date = "20$year-$month-$day";
```

Έπειτα κάνουμε την εισαγωγή του προϊόντος, με τα στοιχεία του να είναι αυτά που πήραμε από την τρέχουσα γραμμή.

```
$q = "INSERT INTO `warehouse`.`transfer` (`id` ,`product_id`
,`stock` ,`date`) VALUES (NULL , '$product_id', '$stock',
'$date');";
$result = update($q);
```

Τέλος δημιουργούμε το μήνυμα που θα εμφανιστεί στον χρήστη. Εδώ χρησιμοποιούμε και την μεταβλητή count για να τον ενημερώσουμε για το πλήθος των νέων εγγραφών που εισάγαμε.

```
$_SESSION['info'] = "Η εισαγωγή ολοκληρώθηκε. Προστέθηκαν $count
                     νέες εγγραφές.";
```

Μετά ο χρήστης ανακατευθύνετε στη λίστα των συναλλαγών.

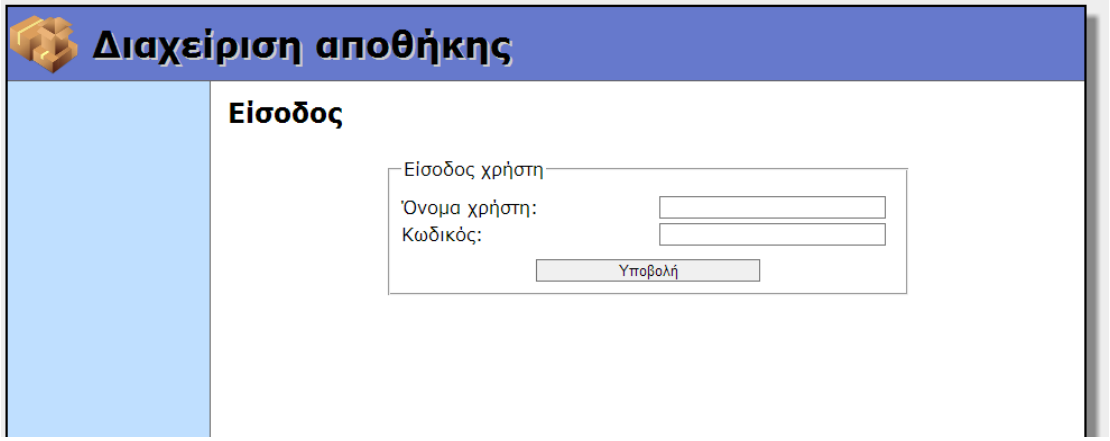
# 5 Οδηγίες χρήσης

## 5.1 Επισκέπτης

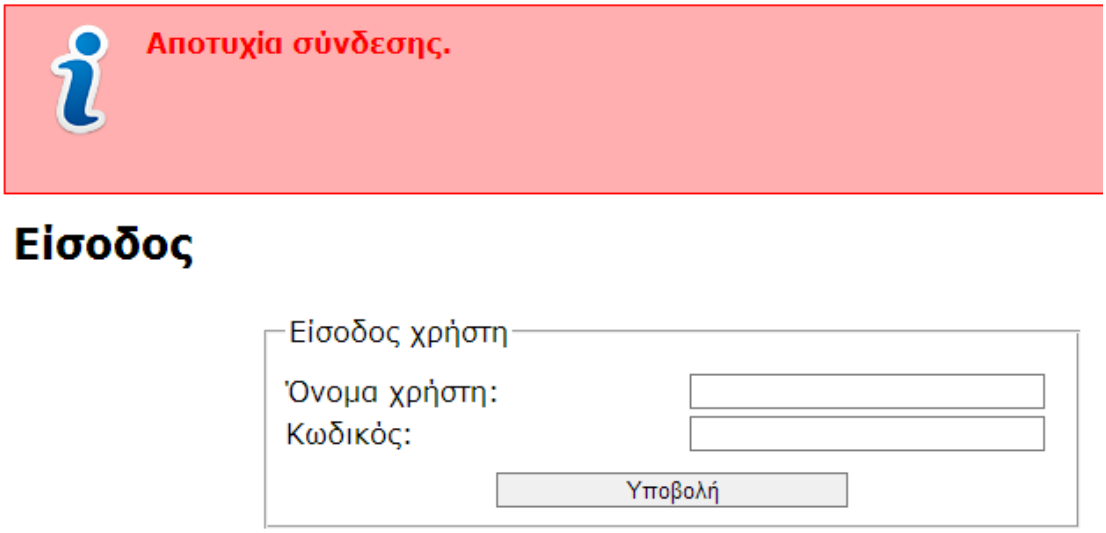
Καθώς το σύστημα απευθύνεται στους υπαλλήλους μίας εταιρείας οι απλοί χρήστες, δηλαδή οι επισκέπτες δεν μπορούν να κάνουν καμία ενέργεια πέρα από το να μπουν στο σύστημα.

### 5.1.1 Είσοδος στο σύστημα

Η αρχική σελίδα που βλέπει ένας απλός χρήστης πηγαίνοντας στη διεύθυνση http://127.0.0.1/warehouse/ είναι η φόρμα εισόδου.

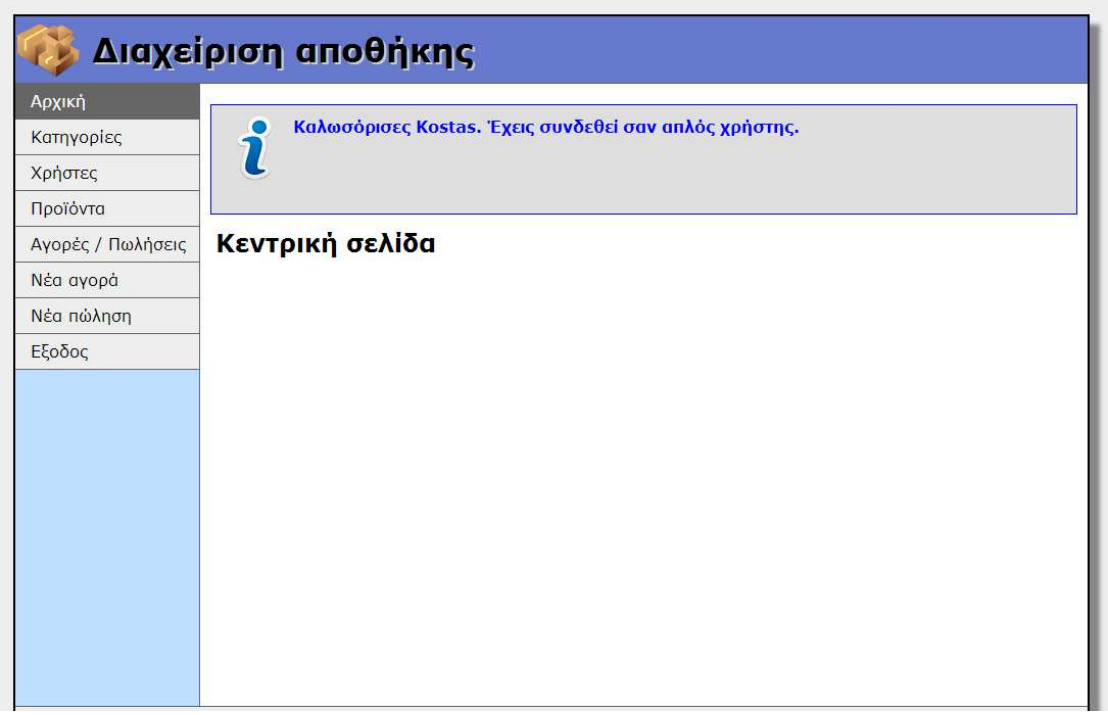



Εκεί συμπληρώνει τα στοιχεία του ώστε να μπει στο σύστημα. Σε περίπτωση λάθος στοιχείων τότε επιστρέφει σε αυτή τη φόρμα και εμφανίζεται το κατάλληλο μήνυμα, αλλιώς προωθείται στην αρχική σελίδα της εφαρμογής.




## 5.2 Απλός χρήστης

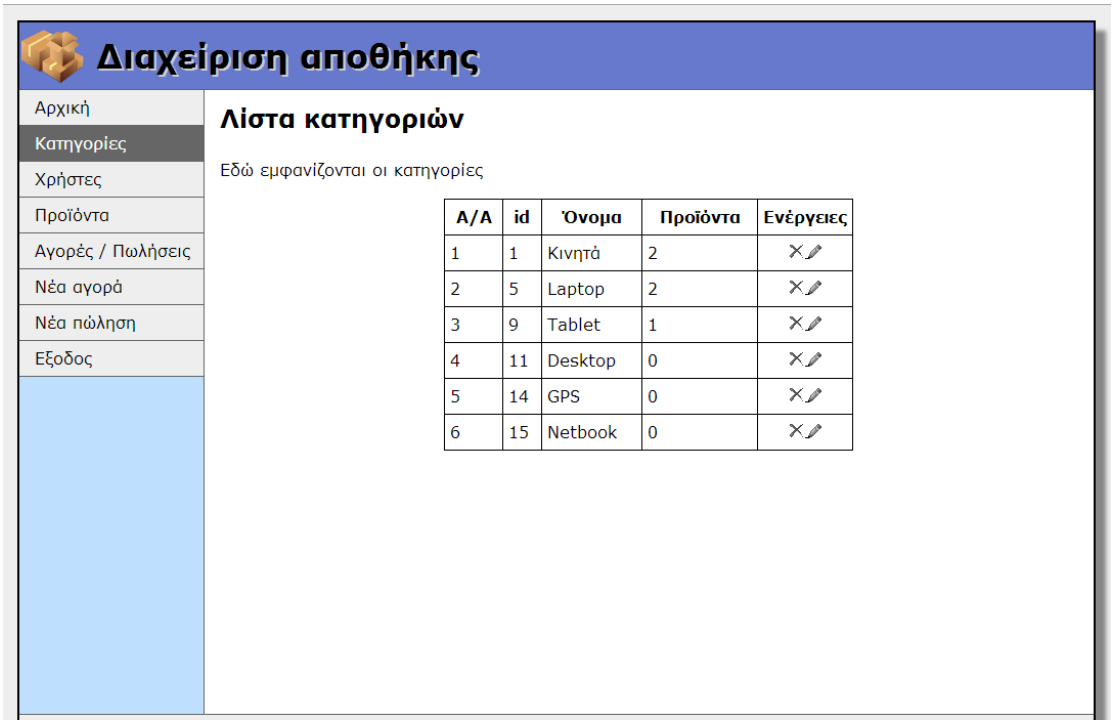
Αν ο χρήστης εισάγει έγκυρα στοιχεία τότε φαίνεται η αρχική σελίδα της εφαρμογής.




Αν δεν είναι διαχειριστής εμφανίζεται ένας περιορισμένος αριθμός επιλογών, οι οποίες έχουν νόημα ώστε να χρησιμοποιηθούν από έναν απλό υπάλληλο.

### 5.2.1 Προβολή κατηγοριών

Επιλέγοντας την επιλογή Κατηγορίες από το μενού εμφανίζεται μία λίστα με τις κατηγορίες προϊόντων που περιέχονται στη βάση.

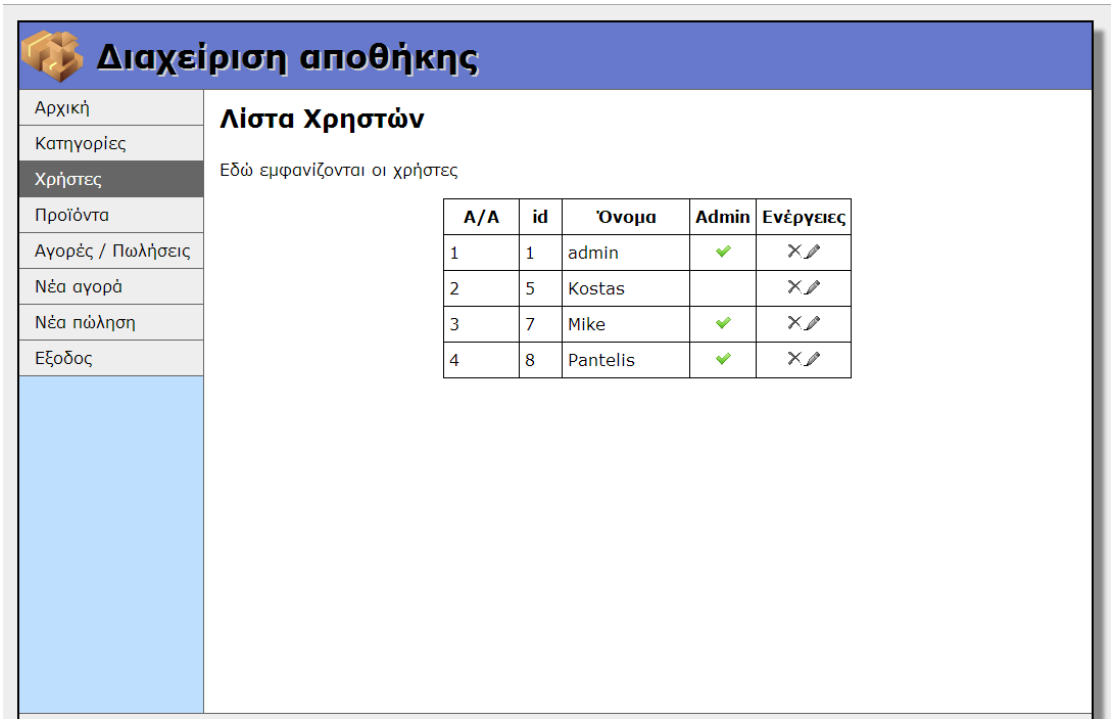



| Α/Α | id | Όνομα | Προϊόντα | Ενέργειες |
|---|---|---|---|---|
| 1 | 1 | Κινητά | 2 | |
| 2 | 5 | Laptop | 2 | |
| 3 | 9 | Tablet | 1 | |
| 4 | 11 | Desktop | 0 | |
| 5 | 14 | GPS | 0 | |
| 6 | 15 | Netbook | 0 | |

Αν και φαίνονται οι επιλογές για διαγραφή και τροποποίηση κάθε κατηγορίας είναι απενεργοποιημένες ώστε ένας απλός αποθηκάριος να μην έχει πρόσβαση σε αυτές.

### 5.2.2 Προβολή χρηστών

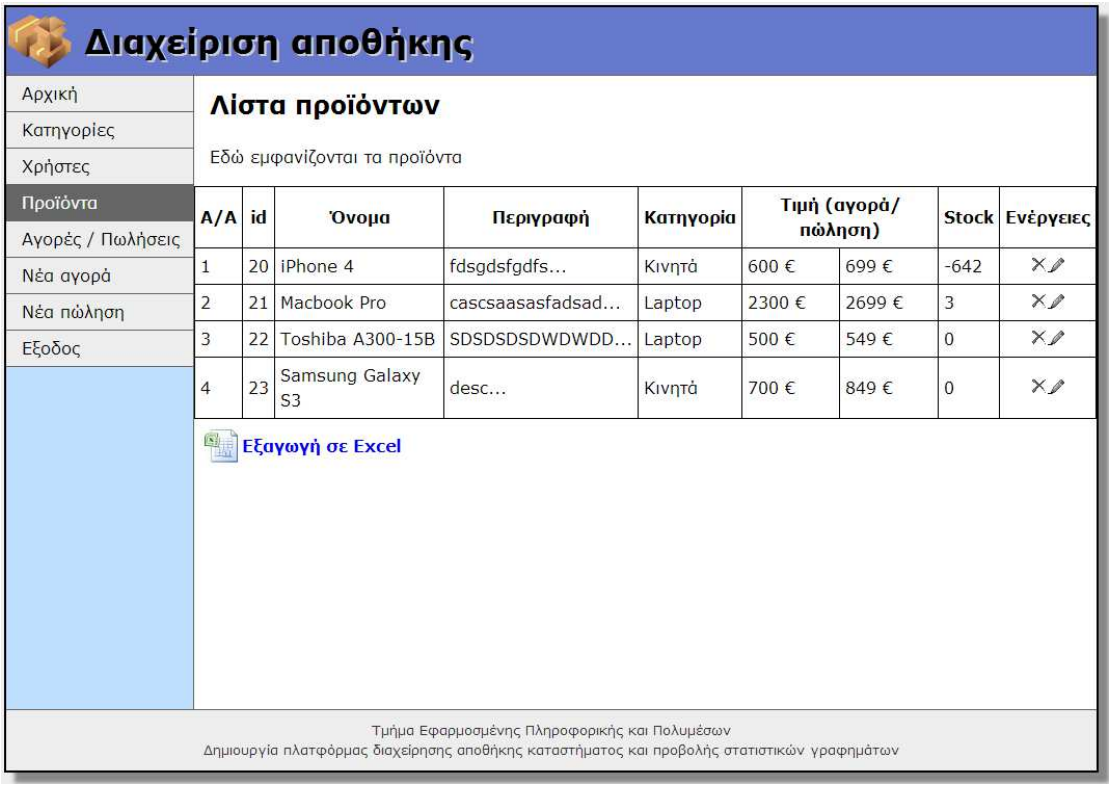
Επιλέγοντας την επιλογή Χρήστες από το μενού εμφανίζεται μία λίστα με τους χρήστες της εφαρμογής.




Αν και φαίνεται ποιοι χρήστες είναι διαχειριστές, δεν δίνεται δυνατότητα τροποποίησης τους.

### 5.2.3 Προϊόντα

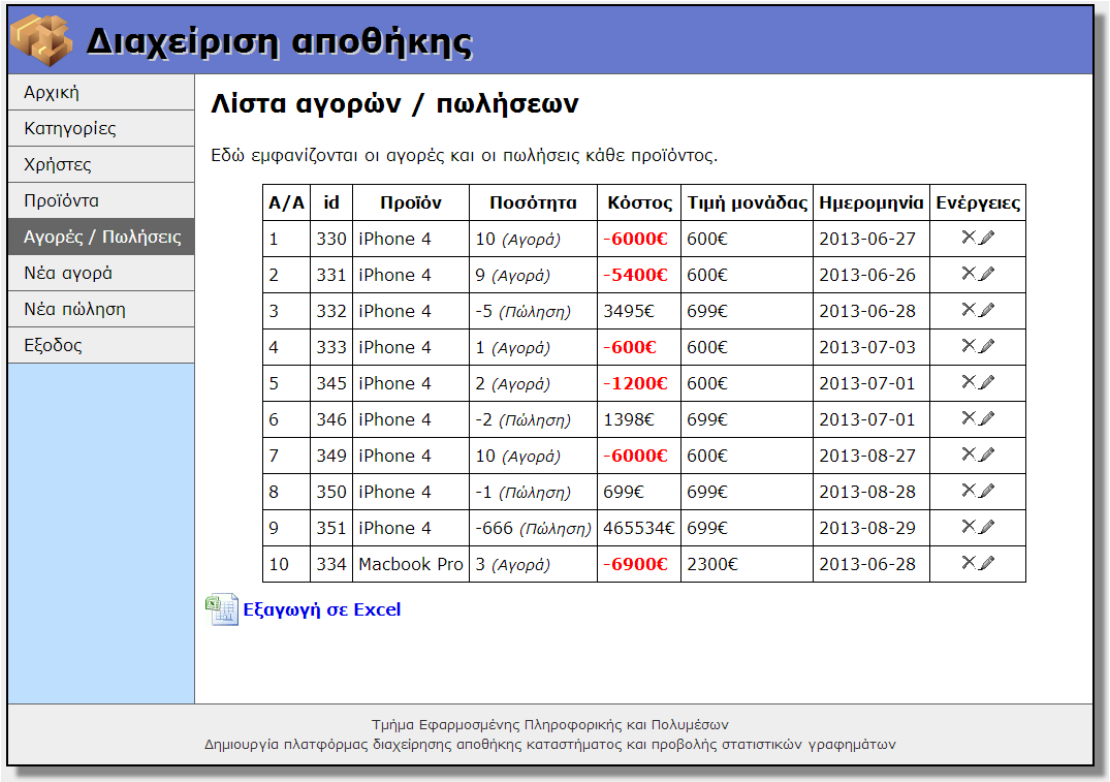
Η λίστα με τα προϊόντα είναι διαθέσιμη και φαίνονται τα στοιχεία του καθενός από την επιλογή Προϊόντα.




Όπως και πριν δεν είναι δυνατή η τροποποίησή του καθενός.

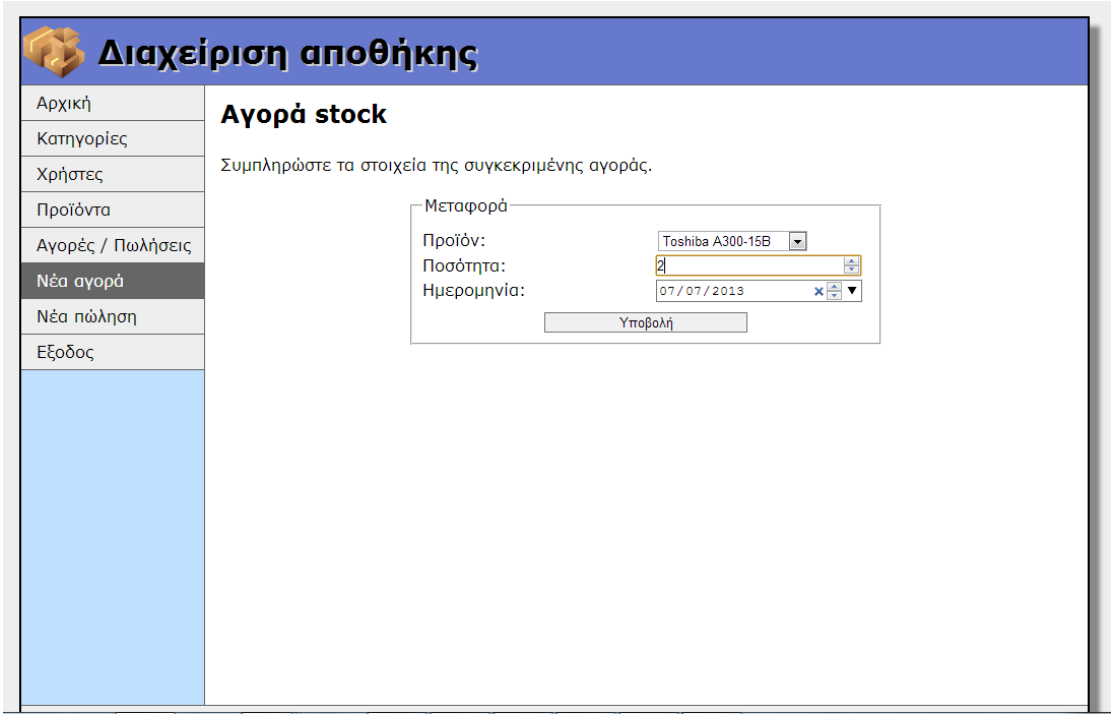
### 5.2.4 Αγορές και Πωλήσεις

Η κύρια ευθύνη ενός αποθηκάριου είναι η μεταβολή των κινήσεων των προϊόντων, το οποίο γίνεται από την επιλογή Αγορές / Πωλήσεις. Αρχικά μπορεί να δει την λίστα τους αλλά δεν μπορεί να τις τροποποιήσει.

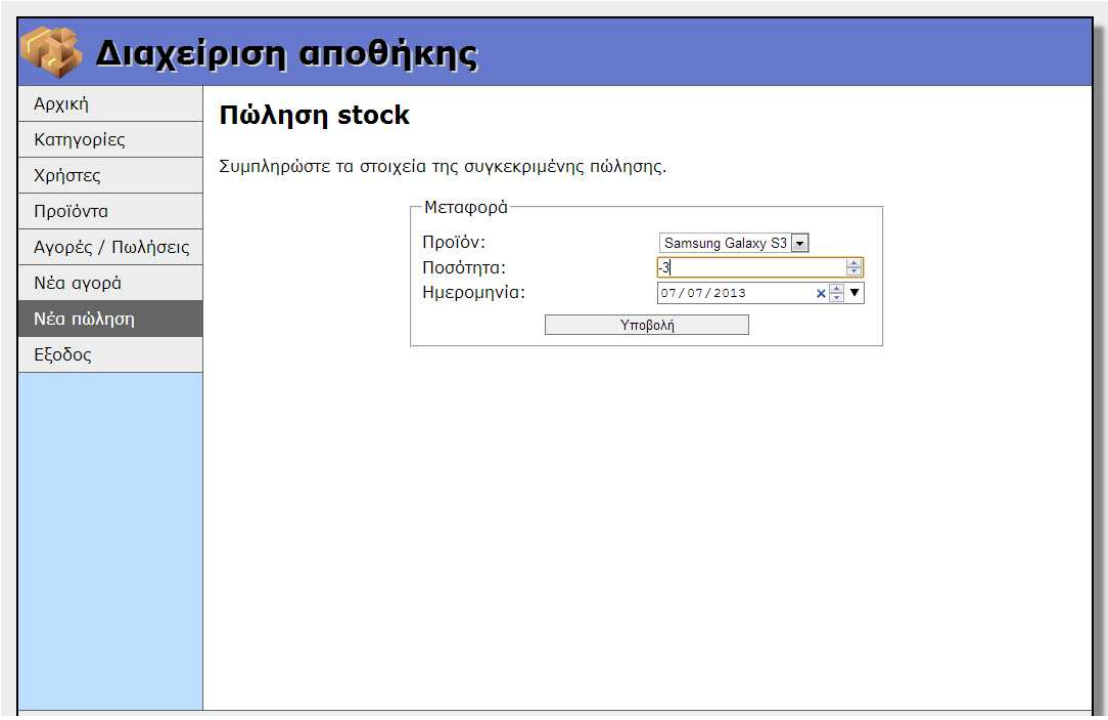


### *5.2.4.1 Προσθήκη αγοράς ή πώλησης*

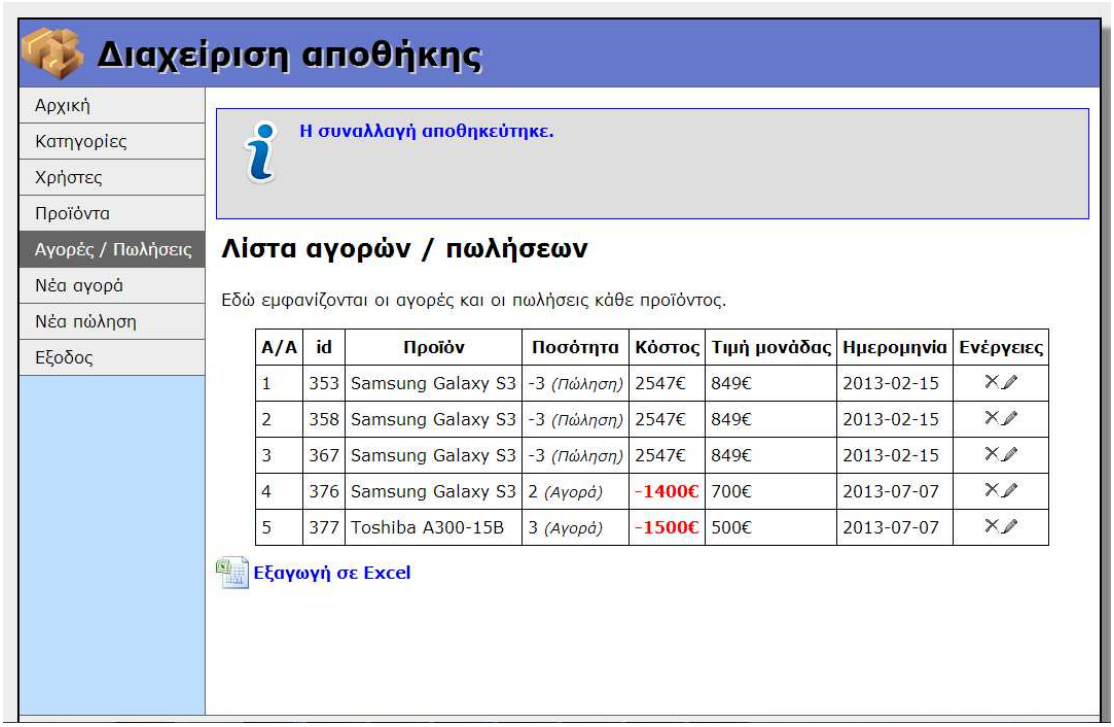
Με την επιλογή Νέα αγορά μπορεί να δημιουργηθεί ένα νέο τέτοιο στοιχείο. Ο χρήστης πρέπει να επιλέξει το προϊόν, την ποσότητα της συναλλαγής και την ημερομηνία.




Αν η ποσότητα είναι θετική πρόκειται για αγορά, ενώ αν είναι αρνητική είναι μία πώληση.



Με την αποθήκευση της συναλλαγής επιστρέφει στην λίστα με τις κινήσεις και εμφανίζεται το κατάλληλο μήνυμα.



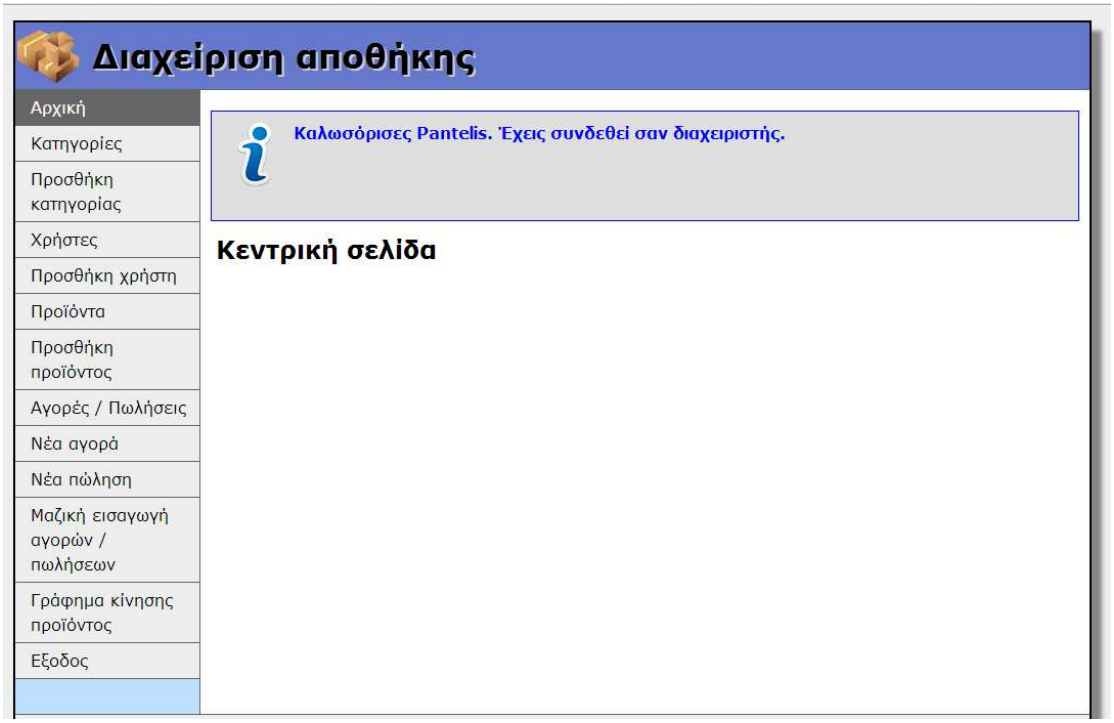

| Α/Α | id | Προϊόν | Ποσότητα | Κόστος | Τιμή μονάδας | Ημερομηνία | Ενέργειες |
|---|---|---|---|---|---|---|---|
| 1 | 353 | Samsung Galaxy S3 | -3 *(Πώληση)* | 2547€ | 849€ | 2013-02-15 | |
| 2 | 358 | Samsung Galaxy S3 | -3 *(Πώληση)* | 2547€ | 849€ | 2013-02-15 | |
| 3 | 367 | Samsung Galaxy S3 | -3 *(Πώληση)* | 2547€ | 849€ | 2013-02-15 | |
| 4 | 376 | Samsung Galaxy S3 | 2 *(Αγορά)* | **-1400€** | 700€ | 2013-07-07 | |
| 5 | 377 | Toshiba A300-15B | 3 *(Αγορά)* | **-1500€** | 500€ | 2013-07-07 | |

### 5.2.5 Έξοδος

Η τελευταία επιλογή του μενού επιτρέπει στον χρήστη να βγει από το σύστημα και επιστρέφει στην οθόνη εισόδου.

## 5.3 Διαχειριστής

Αν ο λογαριασμός ενός χρήστη έχει δικαιώματα διαχειριστή τότε έχει πρόσβαση σε όλες τις επιλογές του συστήματος.




Μπορεί να κάνει ότι και ο απλός χρήστης αλλά στις περισσότερες από τις προηγούμενες οθόνες είναι ενεργοποιημένες επιπλέον δυνατότητες.

### 5.3.1 Χρήστες

Με την επιλογή χρήστες εμφανίζεται πάλι η λίστα με τους χρήστες αλλά είναι ενεργοποιημένα τα κουμπιά για την τροποποίηση αι διαγραφή χρήστη.

## Λίστα Χρηστών

Εδώ εμφανίζονται οι χρήστες

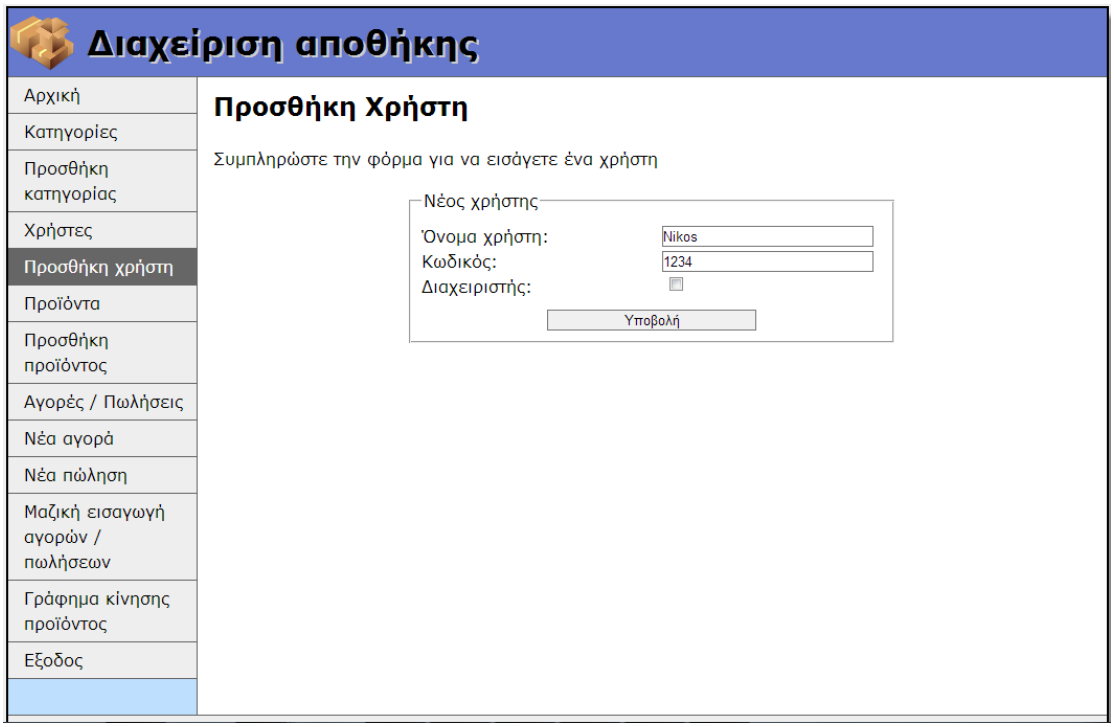
| Α/Α | id | Όνομα | Admin | Ενέργειες |
|---|---|---|---|---|
| 1 | 1 | admin | ✔ | ✕ |
| 2 | 4 | test1 | | ✕ |

#### *5.3.1.1 Διαγραφή χρήστη*

Η διαγραφή ενός χρήστη μπορεί να γίνει πατώντας το κουμπί με το κόκκινο Χ.

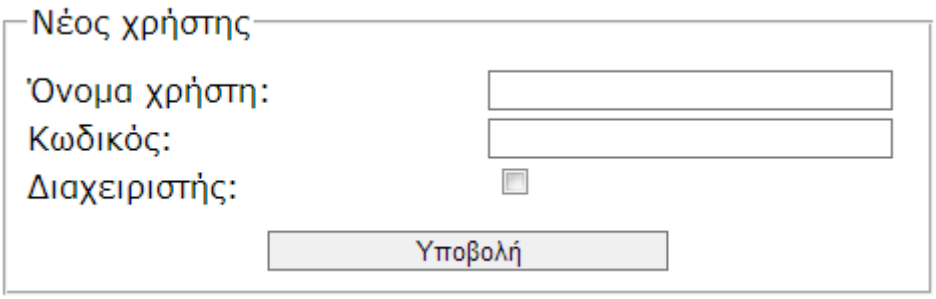
#### *5.3.1.2 Τροποποίηση στοιχείων χρήστη*

Τα στοιχεία ενός χρήστη μπορούν να τροποποιηθούν πατώντας το κουμπί με το μολύβι.




### *5.3.1.3 Εισαγωγή χρήστη*

Για την εισαγωγή ενός νέου χρήστη πρέπει να επιλεγεί το Προσθήκη χρήστη από το μενού στα αριστερά.




Αν επιλεγεί το checkbox Διαχειριστής τότε ο νέος χρήστης θα έχει πλήρη δικαιώματα πρόσβασης.

## 5.3.2 Κατηγορίες

Η λίστα με τις κατηγορίες έχει και αυτή ενεργοποιημένες τις επιλογές για διαγραφή και τροποποίηση.

| Α/Α | id | Όνομα | Προϊόντα | Ενέργειες |
|---|---|---|---|---|
| 1 | 1 | Κινητά | 3 | |
| 2 | 2 | Tablets | 0 | |
| 3 | 5 | Laptop | 0 | |
| 4 | 6 | PC | 0 | |

#### *5.3.2.1 Διαγραφή κατηγορίας*

Η διαγραφή μίας συγκεκριμένης κατηγορίας γίνεται πατώντας το κόκκινο Χ από τη στήλη Ενέργειες. Δεν γίνεται να διαγραφεί μία κατηγορία που περιέχει προϊόντα, όπως φαίνεται παραπάνω για την κατηγορία Κινητά, γι' αυτό και το κουμπί διαγραφής είναι απενεργοποιημένο.

#### *5.3.2.2 Τροποποίηση κατηγορίας*

Η τροποποίηση των στοιχείων μίας κατηγορίας γίνεται πατώντας το μολύβι.

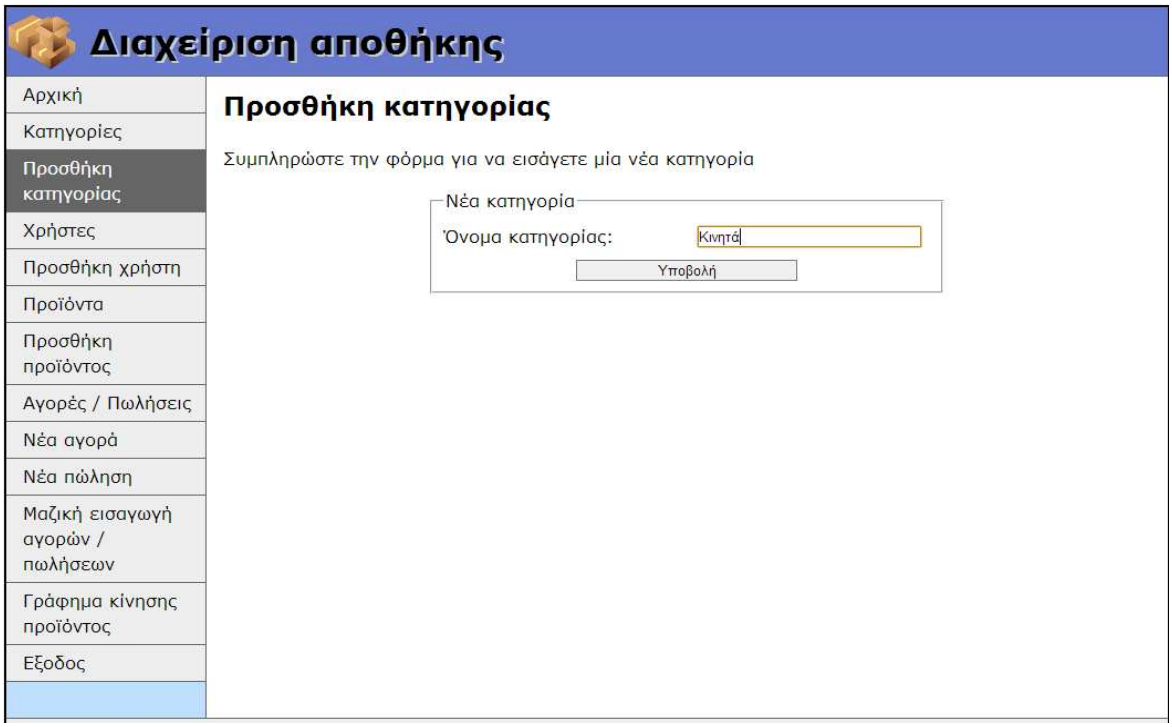
Νέα κατηγορία

Όνομα κατηγορίας: Κινητά

Υποβολή

#### *5.3.2.3 Εισαγωγή νέας κατηγορίας*

Μία νέα κατηγορία εισάγεται πατώντας το Προσθήκη κατηγορίας από το κεντρικό μενού.




## 5.3.3 Προϊόντα

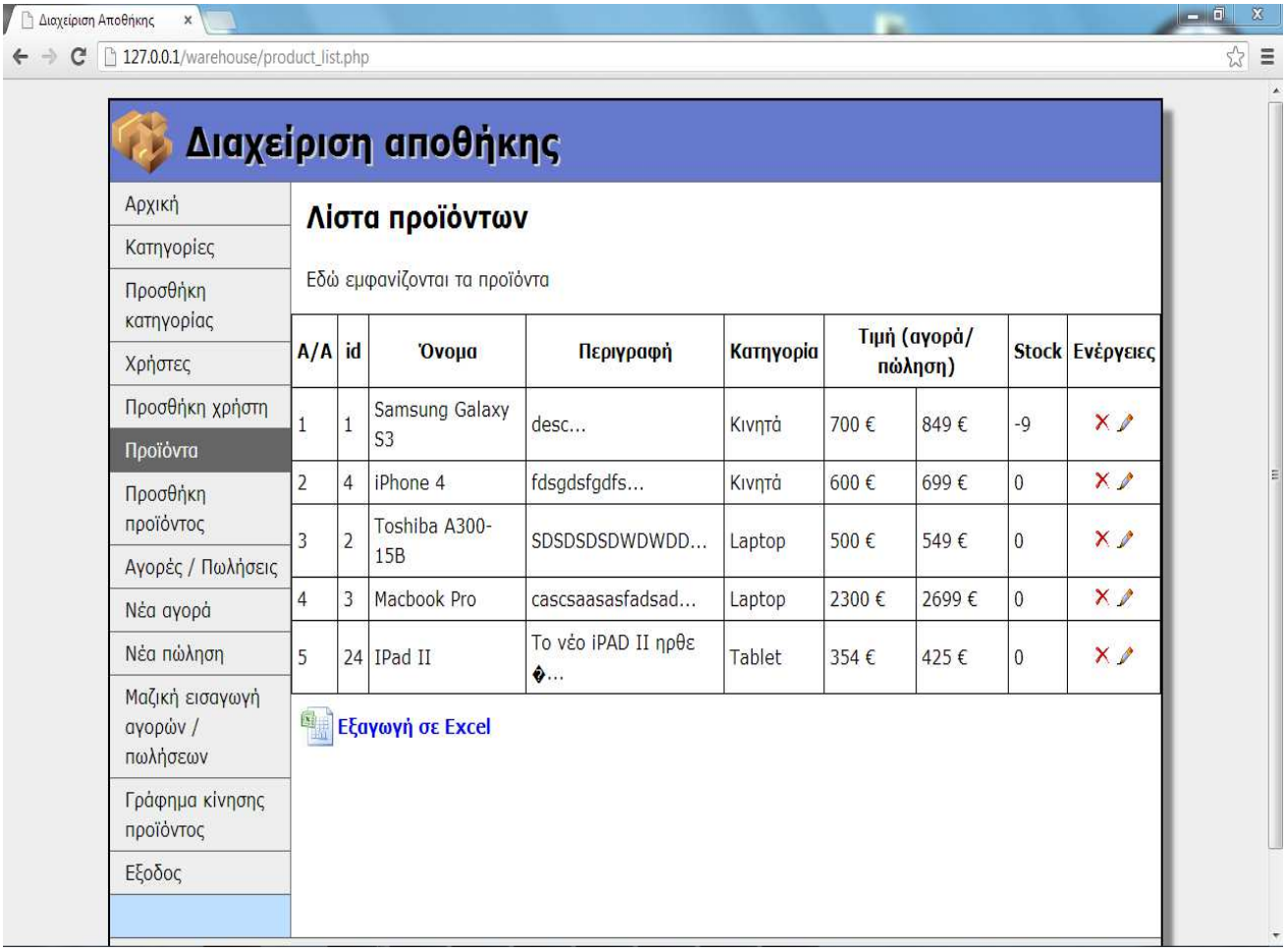
Η λίστα με τα προϊόντα έχει ενεργοποιημένες τις επιλογές διαγραφής και τροποποίησης.

| A/A | id | Όνομα | Περιγραφή | Κατηγορία | Τιμή | Stock | Ενέργειες |
|---|---|---|---|---|---|---|---|
| 1 | 1 | iPhone 4 | fdsfs... | Κινητά | 500 € | 70 | |
| 2 | 5 | Samsung Galaxy S | das... | Κινητά | 333 € | 42 | |
| 3 | 6 | Nokia Lumia 920 | fdgfdsgfgfdgfd... | Κινητά | 400 € | 16 | |

Εξαγωγή σε Excel

### *5.3.3.1 Εξαγωγή προϊόντων*

Κάτω από τη λίστα των προϊόντων υπάρχει η επιλογή Εξαγωγή σε Excel. Πατώντας την η εφαρμογή δημιουργεί ένα αρχείο Excel το οποίο ο χρήστης μπορεί να κατεβάσει.




| Α/Α | id | Όνομα | Περιγραφή | Κατηγορία | Τιμή (αγορά/ πώληση) | | Stock | Ενέργειες |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Samsung Galaxy S3 | desc... | Κινητά | 700 € | 849 € | -9 | |
| 2 | 4 | iPhone 4 | fdsgdsfgdfs... | Κινητά | 600 € | 699 € | 0 | |
| 3 | 2 | Toshiba A300-15B | SDSDSDSDWDWDD... | Laptop | 500 € | 549 € | 0 | |
| 4 | 3 | Macbook Pro | cascsaasasfadsad... | Laptop | 2300 € | 2699 € | 0 | |
| 5 | 24 | IPad II | Το νέο iPAD II ηρθε ...  | Tablet | 354 € | 425 € | 0 | |

Αυτό περιέχει ένα έγγραφο xls με όλα τα προϊόντα.

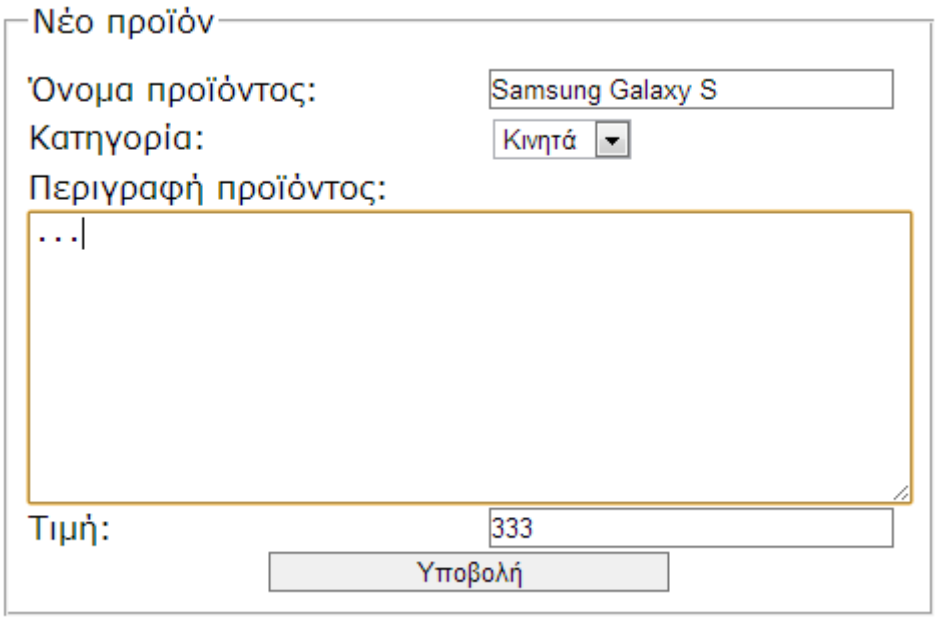
| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Όνομα | Περιγραφ | Κατηγορίε | Τιμή | Stock |
| 2 | iPhone 4 | fdsfs | Κινητά | 500 | 70 |
| 3 | Samsung ( | das | Κινητά | 333 | 42 |
| 4 | Nokia Lum | fdgfdsgfgf | Κινητά | 400 | 16 |
| 5 | | | | | |
| 6 | | | | | |

### *5.3.3.2 Διαγραφή προϊόντος*

Η διαγραφή ενός προϊόντος γίνεται με το κατάλληλο κουμπί από τη στήλη ενέργειες.

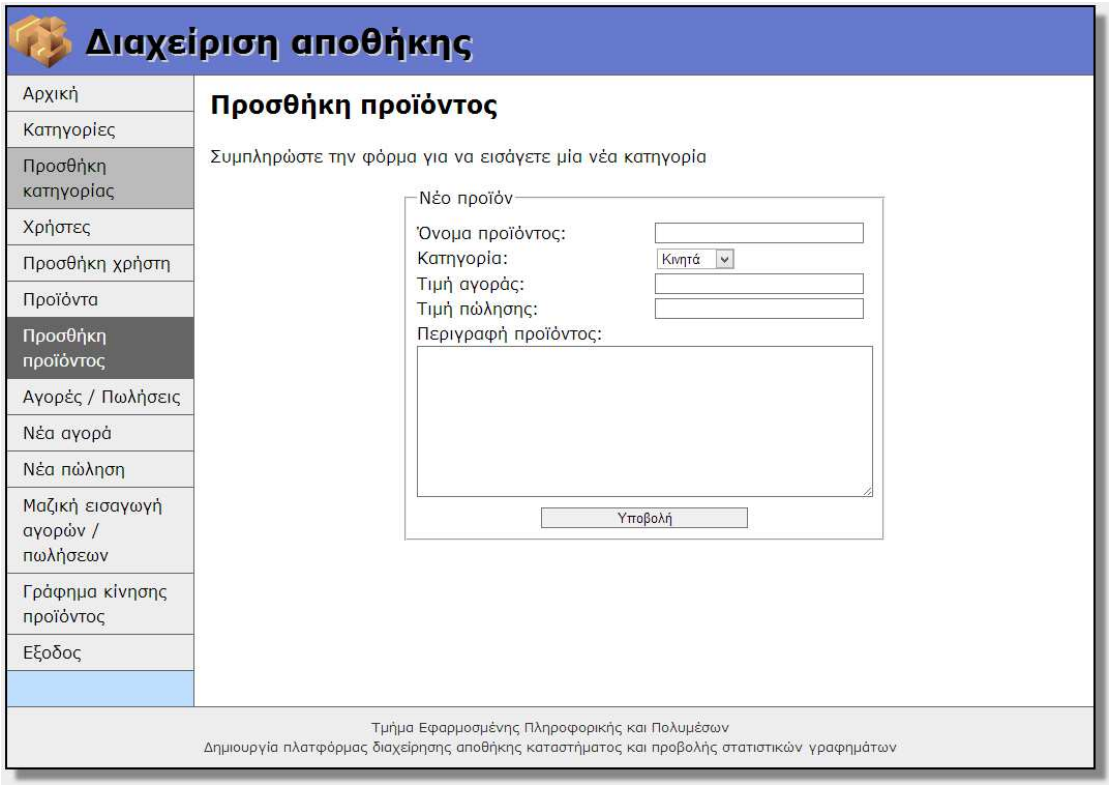
### *5.3.3.3 Τροποποίηση προϊόντος*

Η τροποποίηση στοιχείων ενός προϊόντος γίνεται με το κουμπί του μολυβιού από τη στήλη ενέργειες.




### *5.3.3.4 Εισαγωγή προϊόντος*

Ένα προϊόν εισάγεται από την επιλογή Προσθήκη προϊόντος από το αριστερό μενού.




Η κατηγορία επιλέγεται από το αντίστοιχο drop down menu, που εμφανίζει όλες τις κατηγορίες.

### 5.3.4 Αγορές και πωλήσεις

Όπως και παραπάνω οι αγορές και οι πωλήσεις φαίνονται με ενεργοποιημένα τα κουμπιά των ενεργειών.

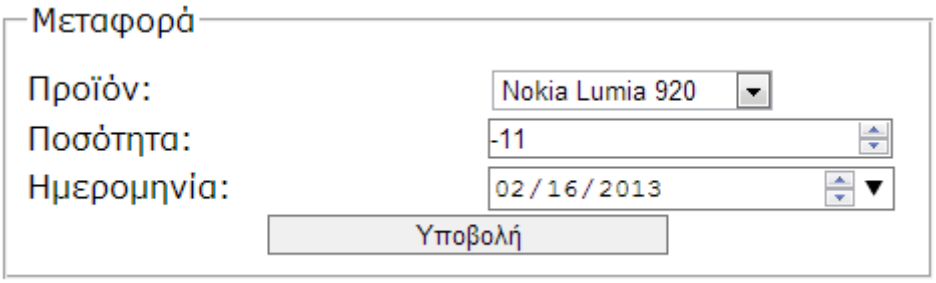
| Α/Α | id | Προϊόν | Ποσότητα | Κόστος | Τιμή μονάδας | Ημερομηνία | Ενέργειες |
|---|---|---|---|---|---|---|---|
| 1 | 330 | iPhone 4 | 10 *(Αγορά)* | **-6000€** | 600€ | 2013-06-27 | ✕ |
| 2 | 331 | iPhone 4 | 9 *(Αγορά)* | **-5400€** | 600€ | 2013-06-26 | ✕ |
| 3 | 332 | iPhone 4 | -5 *(Πώληση)* | 3495€ | 699€ | 2013-06-28 | ✕ |
| 4 | 333 | iPhone 4 | 1 *(Αγορά)* | **-600€** | 600€ | 2013-07-03 | ✕ |
| 5 | 345 | iPhone 4 | 2 *(Αγορά)* | **-1200€** | 600€ | 2013-07-01 | ✕ |
| 6 | 346 | iPhone 4 | -2 *(Πώληση)* | 1398€ | 699€ | 2013-07-01 | ✕ |
| 7 | 349 | iPhone 4 | 10 *(Αγορά)* | **-6000€** | 600€ | 2013-08-27 | ✕ |
| 8 | 350 | iPhone 4 | -1 *(Πώληση)* | 699€ | 699€ | 2013-08-28 | ✕ |
| 9 | 351 | iPhone 4 | -666 *(Πώληση)* | 465534€ | 699€ | 2013-08-29 | ✕ |
| 10 | 334 | Macbook Pro | 2 *(Αγορά)* | **-6000€** | 3000€ | 2013-06-28 | ✕ |

#### *5.3.4.1 Διαγραφή*

Η διαγραφή γίνεται πατώντας το κατάλληλο κουμπί για την επιθυμητή μεταφορά.

#### *5.3.4.2 Τροποποίηση*

Το κουμπί του μολυβιού παρουσιάζει τα στοιχεία σε μία φόρμα ώστε να μπορούν να τροποποιηθούν.

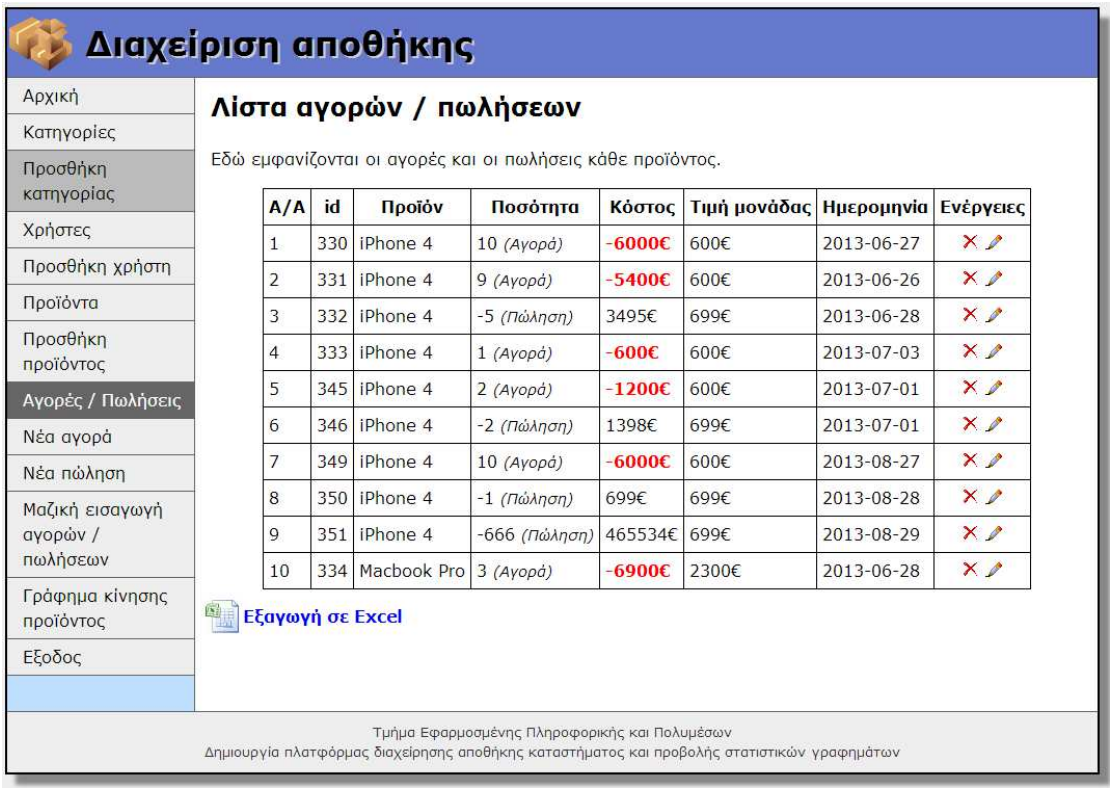



#### *5.3.4.3 Εισαγωγή*

Η εισαγωγή γίνεται με την επιλογή Νέα αγορά / Πώληση από το κεντρικό μενού, όπως και με τον απλό χρήστη.

#### *5.3.4.4 Εξαγωγή*

Στην σελίδα με τη λίστα των κινήσεων έχει προστεθεί μία νέα επιλογή για τον διαχειριστή, το Εξαγωγή σε Excel.



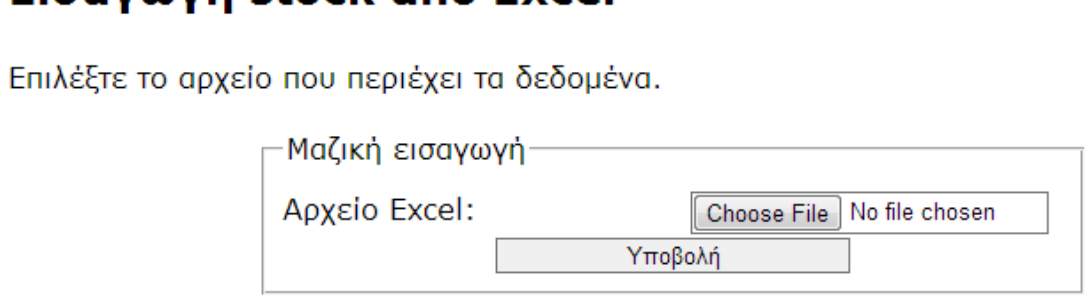

| Α/Α | id | Προϊόν | Ποσότητα | Κόστος | Τιμή μονάδας | Ημερομηνία | Ενέργειες |
|---|---|---|---|---|---|---|---|
| 1 | 330 | iPhone 4 | 10 *(Αγορά)* | -6000€ | 600€ | 2013-06-27 | |
| 2 | 331 | iPhone 4 | 9 *(Αγορά)* | -5400€ | 600€ | 2013-06-26 | |
| 3 | 332 | iPhone 4 | -5 *(Πώληση)* | 3495€ | 699€ | 2013-06-28 | |
| 4 | 333 | iPhone 4 | 1 *(Αγορά)* | -600€ | 600€ | 2013-07-03 | |
| 5 | 345 | iPhone 4 | 2 *(Αγορά)* | -1200€ | 600€ | 2013-07-01 | |
| 6 | 346 | iPhone 4 | -2 *(Πώληση)* | 1398€ | 699€ | 2013-07-01 | |
| 7 | 349 | iPhone 4 | 10 *(Αγορά)* | -6000€ | 600€ | 2013-08-27 | |
| 8 | 350 | iPhone 4 | -1 *(Πώληση)* | 699€ | 699€ | 2013-08-28 | |
| 9 | 351 | iPhone 4 | -666 *(Πώληση)* | 465534€ | 699€ | 2013-08-29 | |
| 10 | 334 | Macbook Pro | 3 *(Αγορά)* | -6900€ | 2300€ | 2013-06-28 | |

Πατώντας το ο διαχειριστής μπορεί να κατεβάσει ένα έγγραφο του Microsoft Excel που περιέχει όλες τις κινήσεις.

| | A | B | C |
|---|---|---|---|
| 1 | Προϊόν | Ποσότητα | Ημερομηνία |
| 2 | iPhone 4 | 10 | 2012-12-05 |
| 3 | iPhone 4 | -3 | 2012-12-06 |
| 4 | iPhone 4 | 5 | 2012-12-21 |
| 5 | iPhone 4 | -2 | 2012-12-21 |
| 6 | iPhone 4 | 4 | 2012-12-31 |
| 7 | Samsung ( | 10 | 2012-12-12 |
| 8 | Samsung ( | -1 | 2012-12-20 |
| 9 | iPhone 4 | 10 | 2012-12-23 |
| 10 | Nokia Lum | 10 | 2013-01-01 |
| 11 | Nokia Lum | -5 | 2013-01-09 |
| 12 | iPhone 4 | 1 | 2013-02-07 |
| 13 | Nokia Lum | 22 | 2013-02-07 |
| 14 | Nokia Lum | -11 | 2013-02-16 |
| 15 | Samsung ( | 33 | 2013-02-17 |
| 16 | iPhone 4 | 44 | 2013-02-15 |
| 17 | iPhone 4 | -1 | 2013-06-08 |
| 18 | iPhone 4 | 1 | 2013-06-08 |
| 19 | iPhone 4 | 1 | 2013-06-08 |

## 5.4 Μαζική εισαγωγή στοιχείων

Πατώντας την επιλογή Μαζική εισαγωγή αγορών / πωλήσεων ο χρήστης μπορεί να ανεβάσει ένα έγγραφο Excel που περιέχει έναν μεγάλο αριθμό κινήσεων.



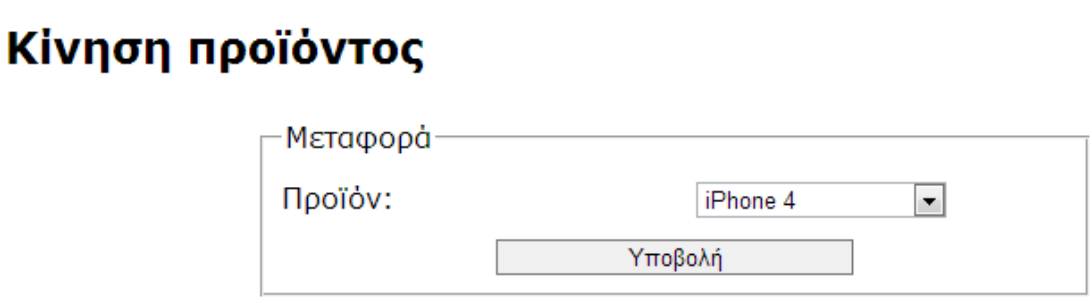

Η μορφή του αρχείου πρέπει να είναι τέτοια ώστε στην πρώτη στήλη να έχει το id του προϊόντος, στη δεύτερη το πλήθος και στην Τρίτη την ημερομηνία.

| | A | B | C |
|---|---|---|---|
| 1 | 1 | 55 | 29/12/2013 |
| 2 | 1 | -8 | 30/12/2013 |

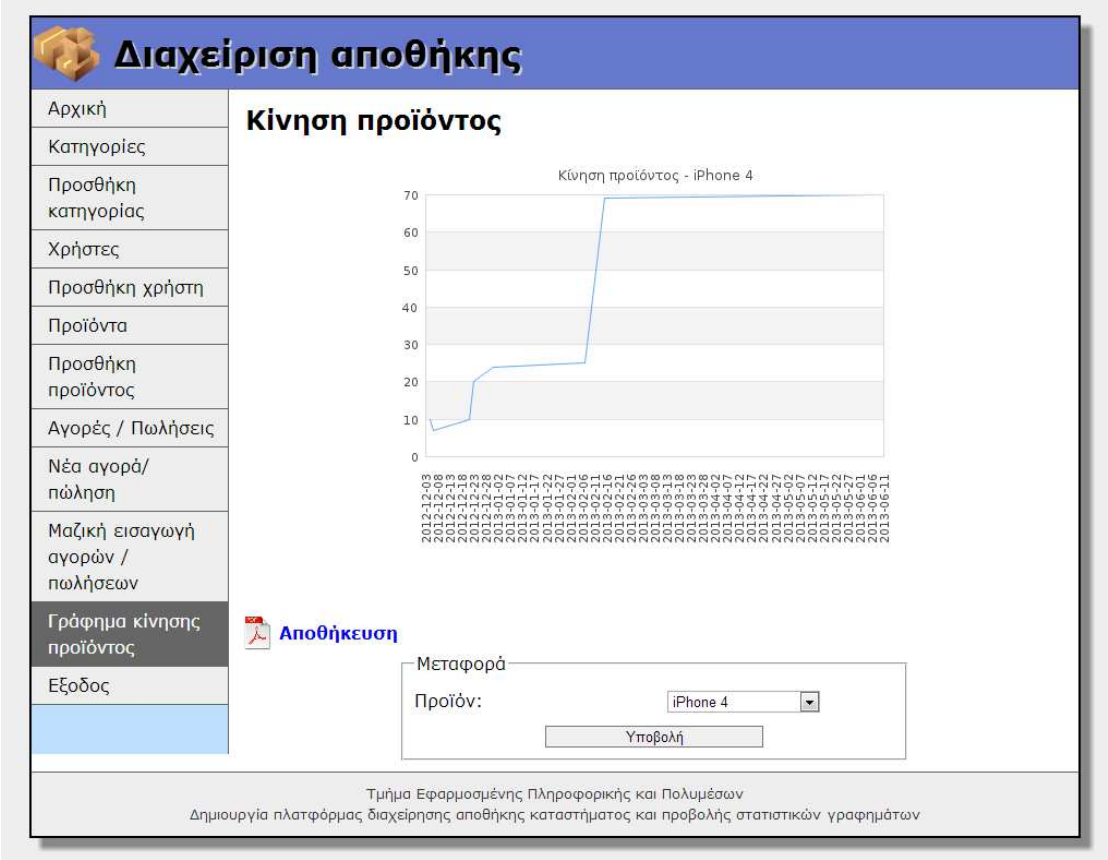
Τα στοιχεία εισάγονται κατευθείαν στη βάση χωρίς να χρειάζεται να μπουν ξεχωριστά από τη φόρμα.

## 5.5 Γράφημα κίνησης προϊόντος

Για κάθε προϊόν μπορεί να δημιουργηθεί ένα γράφημα με τα στοιχεία κίνησής του. Αυτό γίνεται με την επιλογή Γράφημα κίνησης προϊόντος. Ο χρήστης χρειάζεται να επιλέξει το επιθυμητό προϊόν και να πατήσει το Υποβολή.




Εμφανίζεται ένα γράφημα που δείχνει την μεταβολή του stock με το πέρασμα του χρόνου.




Πατώντας το πλήκτρο αποθήκευση ο χρήστης μπορεί να κατεβάσει το διάγραμμα σαν αρχείο PDF.

# 6 Συμπεράσματα

Όπως είδαμε τα προγράμματα διαχείρισης αποθήκης ήταν μία από τις πρώτες εμπορικές εφαρμογές που αναπτύχθηκαν για τους ηλεκτρονικούς υπολογιστές και πολλές εταιρίες έσπευσαν να τα υιοθετήσουν. Αν τα αρχικά προγράμματα του είδους σήμερα θεωρούνται πρωτόγονα, στην εποχή τους προσέφεραν ένα πλήθος πλεονεκτημάτων.

Με το πέρασμα του χρόνου αναπτύχθηκαν αρκετές τεχνολογίες με την χρήση των οποίων ένα πρόγραμμα τέτοιου είδους μπορεί να βελτιωθεί, και σταδιακά εμφανίζονται αρκετές τέτοιες εφαρμογές. Όμως μία σταθερή τάση που υπάρχει στον χώρο είναι η αποφυγή των αλλαγών και τις μετακίνησης σε πιο σύγχρονες τεχνολογίες τόσο από τις εταιρείες που τα χρησιμοποιούν όσο και από τις εταιρείες που τα αναπτύσσουν.

Φαίνεται πως κάτι τέτοιο γίνεται γιατί τελικά πρόκειται για μία διαδικασία που έχει μεγάλο κόστος και μπορεί να διαρκέσει πολύ ή και να διαταράξει την ομαλή λειτουργία της επιχείρησης. Εν τέλει η αναβάθμιση ή όχι είναι μία επιχειρηματική υπόθεση και δεν μπορεί να ζυγιστεί με γνώμονα μόνο τα τεχνολογικά πλεονεκτήματα.

Σχετικά με τις τεχνολογίες που αξιολογήσαμε η ωριμότητά τους οι δυνατότητες τους δεν τέθηκαν ποτέ υπό αμφισβήτηση. Πρόκειται για βασικές τεχνολογίες που αναπτύσσονται για πάνω από μία δεκαετία και είναι υπεύθυνες για την δημιουργία και την παρουσίαση του μεγαλύτερου μέρους του περιεχομένου στο διαδίκτυο.

Η HTML5 είναι μία σημαντική και ταυτόχρονα διακριτική αναβάθμιση της HTML, από την άποψη ότι τα νέα στοιχεία δεν παύουν να είναι συμβατά με τα παλιότερα, αλλά με μπορούν να προστεθούν όπου χρειάζονται ώστε να αντικαταστήσουν εξωτερικές βιβλιοθήκες παρέχοντας απαραίτητες ή αρκετά εξεζητημένες δυνατότητες. Η σταθερότητα και το πλήθος δυνατοτήτων είναι επίσης χαρακτηριστικά της PHP και της MySQL.

Ένα χαρακτηριστικό που προκάλεσε μεγάλη εντύπωση είναι η πληθώρα των εξωτερικών βιβλιοθηκών που παρέχονται για τη γλώσσα PHP. Για όλες τις λειτουργίες που θελήσαμε να υλοποιήσουμε το πρόβλημα δεν ήταν ποτέ η εύρεσή τους αλλά η επιλογή ανάμεσα στο πλήθος των προσφορών. Αυτές διαφέρουν στους τομείς των δυνατοτήτων και της ευχρηστίας, κάνοντας απαραίτητη την δοκιμή κάποιων πριν τη χρήση τους.

Βέβαια η μεγάλη διάδοση των παραπάνω τεχνολογιών δεν εγγυάται σε καμία περίπτωση πως η δημιουργία μίας εφαρμογής είναι μία απλή υπόθεση. Αντίθετα απαιτείται καλή θεωρητική κατάρτιση για την κάθε μία και κυρίως στο πως συνδυάζονται μεταξύ του. Όπως αν υπάρχουν οι γνώσεις, τότε σε συνδυασμό με όρεξη για δουλειά μπορούν σίγουρα να αποφέρουν καρπούς.

# 7 Bibliography

1. Enterprise resource planning. *Wikipedia.* [Ηλεκτρονικό] http://en.wikipedia.org/wiki/Enterprise_resource_planning.

2. **Logistics.** Wikipedia. [Ηλεκτρονικό] http://en.wikipedia.org/wiki/Logistics.

3. Guest Column: Knowing When a WMS or WCS Is Right for Your Company. *sdcexec.com.* [Ηλεκτρονικό] http://www.sdcexec.com/news/10327037/guest-column-knowing-when-a-wms-or-wcs-is-right-for-your-company.

4. ERP History. *Open Source ERP Guru.* [Ηλεκτρονικό] http://opensourceerpguru.com/2009/02/25/erp-history/.

5. MRP Versus ERP. *sheetmetalworld.com.* [Ηλεκτρονικό] http://sheetmetalworld.com/sheet-metal-news/17-it-for-manufacturing-management-and-production/11086-mrp-versus-erp.

6. MySQL ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ. *qualisys.gr.* [Ηλεκτρονικό] http://www.qualisys.gr/MySQL-basi-dedomenon.html.

7. Τι είναι μία MySQL βάση δεδομένων? *tophost.gr.* [Ηλεκτρονικό] http://support.tophost.gr/index.php?/Knowledgebase/Article/View/102/4/mysql.

8. Εισαγωγή στις Βάσεις Δεδομένων (DataBases). *ΚΕΝΤΡΟ ΠΛΗ.ΝΕ.Τ. Ν. ΦΛΩΡΙΝΑΣ.* [Ηλεκτρονικό] http://dide.flo.sch.gr/Plinet/Tutorials/Tutorials-DataBases.html.

9. PHP. *Wikipedia.* [Ηλεκτρονικό] http://en.wikipedia.org/wiki/PHP.

10. HTML. *Βικιπαίδεια.* [Ηλεκτρονικό] http://el.wikipedia.org/wiki/HTML.

11. HTML5. *Βικιπαίδεια.* [Ηλεκτρονικό] http://el.wikipedia.org/wiki/HTML5.

12. INVENTORY MANAGEMENT HISTORY PART FOUR. *Almyta Systems* . [Ηλεκτρονικό] http://www.almyta.com/Inventory_Management_History_4.asp.

13. Lean Manufacturing (minimizing waste with TPS and its descendants). *Almyta Systems* . [Ηλεκτρονικό] http://www.almyta.com/Inventory_Management_History_6.asp.

14. How Inventory Management Systems Work. *howstuffworks.com.* [Ηλεκτρονικό] http://money.howstuffworks.com/how-inventory-management-systems-work1.htm.

15. Graphical user interface. *Wikipedia.* [Ηλεκτρονικό] http://en.wikipedia.org/wiki/Graphical_user_interface.

16. The first GUIs. *catb.org.* [Ηλεκτρονικό] http://www.catb.org/~esr/writings/taouu/html/ch02s05.html.

17. Database. *Wikipedia.* [Ηλεκτρονικό] http://en.wikipedia.org/wiki/Database.

18. PHP and MySQL Tutorial – Lesson 1. *webmonkey.com.* [Ηλεκτρονικό] http://www.webmonkey.com/2010/02/php_and_mysql_tutorial_-_lesson_1/.