Τ.Ε.Ι. ΚΡΗΤΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Πληροφοριακό Σύστημα Διαχείρισης Δεδομένων Παραγωγής Ελαιόλαδου του Συνεταιρισμού Λέσβου

Παναγιώτης Πασινιός ΑΜ 2821
Βασίλης Ιωσηφέλλης ΑΜ 2969

Ηράκλειο Κρήτης, 2015

Περίληψη

Η διπλωματική αυτή εργασία έχει στόχο το σχεδιασμό, την ανάλυση, και την υλοποίηση ενός συστήματος διαχείρισης της παραγωγής ελαιόλαδου για την ένωση αγροτικών συνεταιρισμών Λέσβου. Το σύστημα αυτό θα προσφέρει δυνατότητες εισαγωγής και διαχείρισης των παραγωγών, των τιμών των διάφορων τύπων ελαιόλαδου, προεπισκόπησης της τιμής πώλησης μιας παραγωγής, καθώς και αναλυτικές αναφορές για τις πωλήσεις όσο αφορά τον συνεταιρισμό, τα εργοστάσια, τους χρήστες.

1. Εισαγωγή

1.1 Ένωση Αγροτικών Συνεταιρισμών Λέσβου

1.1.1 Προφίλ

Η Ένωση Αγροτικών Συνεταιρισμών Λέσβου είναι η δευτεροβάθμια συνεταιριστική οργάνωση αγροτών του νομού Λέσβου. Ιδρύθηκε το 1929 με έδρα την πρωτεύουσα του νομού τη Μυτιλήνη.
Τα κεντρικά της γραφεία στεγάζονται σήμερα στη Σκάλα Παμφίλων της Μυτιλήνης, όπου βρίσκεται και το βιομηχανικό συγκρότημα που διαθέτει όλους τους αναγκαίους χώρους και τον απαραίτητο εξοπλισμό για να διενεργηθούν όλες οι εργασίες από τον ποιοτικό έλεγχο και την αποθήκευση του ελαιόλαδου ως την τυποποίηση. Στην περιοχή ευθύνης της δραστηριοποιούνται 19.720 ελαιοπαραγωγοί που καλλιεργούν 485.434 στρέμματα στα οποία υπάρχουν 8.286.300 δέντρα.
Στόχος της είναι η διαφύλαξη των συμφερόντων των συνεταιρισμένων της και η διαφύλαξη της υψηλής ποιότητας των προϊόντων που παράγονται στο νομό.
Η ΕΑΣ δραστηριοποιείται στην παραγωγή, τυποποίηση και διάθεση ελαιόλαδου, πυρηνέλαιου, σαπουνιού και στην εμπορία ζωοτροφών, λιπασμάτων, γεωργικών μηχανημάτων, φαρμάκων και ειδών ελαιοσυλλογής.

1.1.2 Δραστηριότητες

- Παραγωγή, τυποποίηση, διάθεση ελαιόλαδου, πυρηνέλαιου και σαπουνιού.
- Εμπορία ζωοτροφών, λιπασμάτων, γεωργικών μηχανημάτων & φαρμάκων, ειδών ελαιοσυλλογής.
- Τουριστικό γραφείο, γραφείο ασφαλειών, γραφείο Επιδοτήσεων & κοινοτικών προγραμμάτων.

Για την διανομή των προϊόντων της στην Ελλάδα έχει αντιπροσώπους οι οποίοι ελέγχουν σημαντικές γεωγραφικές περιοχές.

1.2 Διαδίκτυο & Παγκόσμιος Ιστός

1.2.1 Τι είναι το Διαδίκτυο

Το Διαδίκτυο είναι ένα μεγάλο σύστημα διασυνδεδεμένων μικρότερων δικτύων υπολογιστών, οι οποίοι επικοινωνούν μεταξύ τους χρησιμοποιώντας καθορισμένα πρωτόκολλα. Οι υπολογιστές ανταλλάσσουν δεδομένα και πληροφορίες με τη μορφή πακέτων, η μορφή των οποίων είναι σαφώς ορισμένη από τα πρωτόκολλα που χρησιμοποιούνται.
Οποιοσδήποτε διασυνδεδεμένος υπολογιστής είναι σε θέση να επικοινωνήσει με τους υπόλοιπους, στέλνοντας ή λαμβάνοντας πληροφορίες.

1.2.2 Τι είναι ο Παγκόσμιος Ιστός

Ο Παγκόσμιος ιστός είναι ίσως το κυριότερο και δημοφιλέστερο κομμάτι του Διαδικτύου. Χρησιμοποιώντας το πρωτόκολλο επικοινωνίας «http», οι συνδεδεμένοι

υπολογιστές στέλνουν ή δέχονται δεδομένα, τα οποία στη συνέχεια μεταφράζονται - με τη βοήθεια συνήθως ενός φυλλομετρητή (αγγλ. web browser) – σε κείμενο, εικόνα, βίντεο και ήχο. Η τεχνολογία του ιστού δημιουργήθηκε το 1989 από τον Βρετανό Τιμ Μπέρνερς Λη (Sir Timothy John Berners-Lee), που εκείνη την εποχή εργαζόταν στον Ευρωπαϊκό Οργανισμό Πυρηνικών Ερευνών (CERN) στην Γενεύη της Ελβετίας.

1.2.3 Πελάτες και Εξυπηρετητές (αγγλ. clients and servers)

Ο παγκόσμιος ιστός λειτουργεί με την επικοινωνία Πελατών και Εξυπηρετητών. Ο εξυπηρετητής είναι ένας υπολογιστής τρέχει ειδικό λογισμικό ώστε να μπορεί να επεξεργάζεται αιτήματα, να «σερβίρει» δεδομένα και να αποθηκεύει πληροφορίες. Οι Πελάτες είναι ειδικά προγράμματα (π.χ. φυλλομετρητές) τα οποία μπορούν να επικοινωνούν με τους Εξυπηρετητές, να λαμβάνουν τις απαντήσεις τους, και να τις μετατρέπουν σε αντιληπτή από τον άνθρωπο μορφή (κείμενο, εικόνα, βίντεο, ήχος).

2. Στόχος της εργασίας

Στόχος της παρούσας εργασίας είναι η αξιοποίηση των δυνατοτήτων που παρέχει το Διαδίκτυο και ο Παγκόσμιος Ιστός, ώστε να επιτευχθούν οι στόχοι της Ένωσης Αγροτικών Συνεταιρισμών Λέσβου με αποδοτικότερο τρόπο.

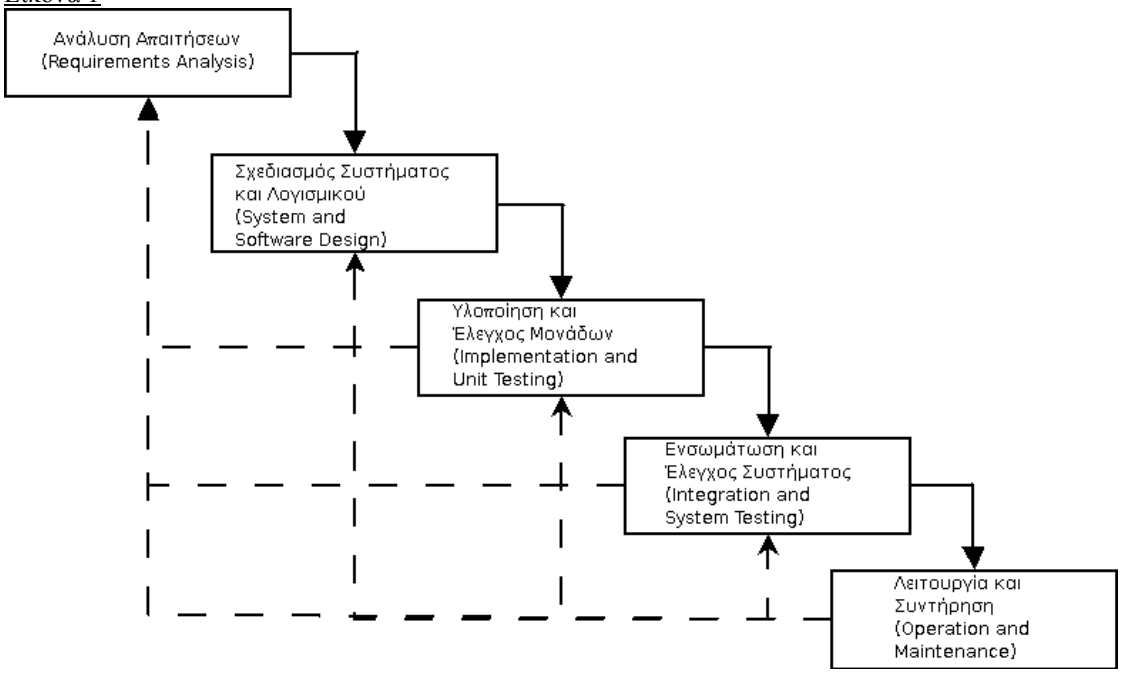
3. Ανάπτυξη εφαρμογής

3.1 Μεθοδολογία ανάπτυξης

Για την υλοποίηση της εφαρμογής, έγινε αρχικά ανάλυση των απαιτήσεων και σχεδίαση των τεχνικών προγραμματισμού που θα χρησιμοποιηθούν μετέπειτα στην υλοποίηση του συστήματος – όπως περιγράφει η μεθοδολογία που είναι γνωστή ως «μοντέλο του καταρράκτη»(εικόνα 1). Στη συνέχεια η εφαρμογή χωρίστηκε σε επιμέρους κομμάτια τόσο σχετικά με τα διάφορα επίπεδα τεχνολογίας (βάση δεδομένων, εφαρμογή εξυπηρετητή, διεπαφή χρήστη), όσο και με βάση τις διάφορες λειτουργίες. Κάθε ένα από αυτά τα κομμάτια υλοποιήθηκε αρχικά ξεχωριστά, και όταν έφτανε σε σχεδόν τελικό στάδιο, ενοποιούνταν με τα υπόλοιπα.

Χρησιμοποιήθηκαν αποκλειστικά αντικειμενοστραφείς τεχνικές, δόθηκε μεγάλη σημασία στις έννοιες του πολυμορφισμού και τις ενθυλάκωσης, ενώ δόθηκε μεγάλη έμφαση στην επαναχρησιμοποίηση κώδικα, την ομαδοποίηση δηλαδή λειτουργιών με τρόπο ώστε να μπορούν να χρησιμοποιούνται ξανά χωρίς κόπο.

Εικόνα 1



3.2 Γλώσσες προγραμματισμού που χρησιμοποιήθηκαν

3.2.1 HTML

Η HTML (Hyper Text Markup Language) είναι μία περιγραφική γλώσσα, ένας ειδικός τρόπος γραφής ιστοσελίδων. Ο φυλλομετρητής είναι το πρόγραμμα στον υπολογιστή του χρήστη που αναγνωρίζει αυτόν τον ειδικό τρόπο γραφής και εκτελεί τις εντολές που περιέχονται σε αυτό. Η HTML χρησιμοποιεί ειδικές ετικέτες σήμανσης (tags). Τα tags βρίσκονται μεταξύ των συμβόλων < και >. Οι HTML ιστοσελίδες είναι στατικές δηλαδή το περιεχόμενο τους είναι σταθερό, αυτό σημαίνει ότι δεν μπορούν να επικοινωνήσουν με άλλες εφαρμογές όπως π.χ με μία βάση δεδομένων.

3.2.2 Java

Η Java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού, η οποία είναι ανεξάρτητη λειτουργικού συστήματος και πλατφόρμας. Τα προγράμματα που είναι γραμμένα σε Java τρέχουνε με τον ίδιο τρόπο σε Windows, Linux/Unix και Macintosh χωρίς να χρειαστεί να ξαναγίνει μεταγλώττιση ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα.

Java Servlets

Ένα Servlet είναι μια κλάση της Java που χρησιμοποιείται για την επέκταση ενός εξυπηρετητή ο οποίος φιλοξενεί εφαρμογές που βασίζονται στο μοντέλο αιτήματος-ανταπόκρισης(request-response). Μερικά από τα πλεονεκτήματα που παρουσιάζουν τα Servlet είναι η ευχρηστία τους καθώς είναι γραμμένα σε Java , μια ευρέως διαδεδομένη και εύχρηστη γλώσσα προγραμματισμού. Επιπλέον η εκφραστικότητα τους, καθώς μπορούν να επικοινωνήσουν με τον web server τόσο και με τον εαυτό τους, και η φορητότητα τους καθώς μπορούν να εγκατασταθούν σε οποιονδήποτε εξυπηρετητή και πλατφόρμα.

JSP (Java Server Pages)

Το jsp είναι μία συμπληρωματική Java τεχνολογία που επιτρέπει συνδυασμό HTML με κώδικα Java. Οι jsp σελίδες στη ουσία είναι σελίδες δυναμικού περιεχομένου, σε αντίθεση με το στατικό περιεχόμενο της HTML. Είναι η ανάγκη του να γράφεις καθαρό HTML καθαρό HTML κώδικα, έχω όμως τη δυνατότητα να προσθέσεις δυναμικό περιεχόμενο, που οδήγησε στην δημιουργία του jsp, μιας που κάτι τέτοιο γινόταν με σχετικά άκομψο τρόπο μέσα από τα Java Servlets.

3.2.3 SQL

Η SQL (Structured Query Language) είναι μία γλώσσα προγραμματισμού βάσεων δεδομένων, που σχεδιάστηκε για τη διαχείριση δεδομένων, σε ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων και η οποία, αρχικά, βασίστηκε στη σχεσιακή άλγεβρα. Η γλώσσα περιλαμβάνει δυνατότητες ανάκτησης και ενημέρωσης δεδομένων, δημιουργίας και τροποποίησης σχημάτων και σχεσιακών πινάκων, αλλά και ελέγχου πρόσβασης στα δεδομένα.

3.2.4 CSS

Τα CSS (Cascading Style Sheets) είναι ηλεκτρονικά έγγραφα με ένα σύνολο κανόνων για την μορφοποίηση μιας ιστοσελίδας.
Ενδεικτικά, χρησιμοποιείται για να διαμορφώσει:

- Τα χρώματα και το μέγεθος διαφόρων στοιχείων της ιστοσελίδας
- Την συμπεριφορά τους σε διάφορες ενέργειες

Παλιότερα, συνηθίζονταν να διαμορφώνεται κάθε στοιχείο της ιστοσελίδας ξεχωριστά, σε κάθε μία ξεχωριστή σελίδα. Η μεγάλη ευκολία που προσφέρει η χρήση ενός CSS, είναι ότι οι ενδεχόμενες αλλαγές, γίνονται μόνο σε ένα έγγραφο και αυτόματα εφαρμόζονται σε όλες τις

σελίδας που το χρησιμοποιούν. Έτσι μία ιστοσελίδα που χρησιμοποιεί CSS, μπορεί πολύ ευκολότερα να αλλάξει όψη.

3.3 Τεχνολογίες / Εργαλεία

3.3.1 MySQL

Μία βάση δεδομένων είναι μία δομημένη συλλογή δεδομένων. Το MySQL είναι ένα σύστημα διαχείρισης βάσεων δεδομένων, μέσα από το οποίο μπορούμε να εκτελούμε ερωτήματα στη βάση δεδομένων, να ανακτούμε δεδομένα, να εισάγουμε ή να διαγράφουμε δεδομένα, να ενημερώνουμε τις εγγραφές μας, να δημιουργούμε ή να διαγράφουμε πίνακες ή ακόμα και ολόκληρες βάσεις δεδομένων.

3.3.2 Apache Tomcat

Ο Apache Tomcat είναι μία εφαρμογή - εξυπηρετητής που επιτρέπει την εκτέλεση κώδικα Java, καθώς επίσης υλοποιεί τις προδιαγραφές των τεχνολογιών JSP και Java Servlets.

3.3.3 NetBeans IDE

Το Netbeans είναι ένα εργαλείο – περιβάλλον ανάπτυξης εφαρμογών, σχεδιασμένο κυρίως για Java εφαρμογές, αλλά όχι αποκλειστικά. Μέσα από αυτό μπορούμε επίσης να τρέξουμε εικονικούς εξυπηρετητές (π.χ. Apache Tomcat) ώστε να προσομοιώνουμε τις αντίστοιχες λειτουργίες και να εκτελούμε τα προγράμματά μας.

3.4 Απαιτήσεις Συστήματος

Βάσει των γλωσσών προγραμματισμού και των τεχνολογιών που είδαμε παραπάνω ότι χρησιμοποιούμε για την εφαρμογή μας, συνοψίζουμε τις ελάχιστες απαιτήσεις συστήματος:

- MySQL Server
- Apache Tomcat
- NetBeans (προτεινόμενη έκδοση: 8.0.2 ή μεταγενέστερη)

Προτείνεται η εγκατάσταση του πακέτου NetBeans "Java EE" που έρχεται μαζί με τον Apache Tomcat Server ώστε να είναι ευκολότερη η διαδικασία.

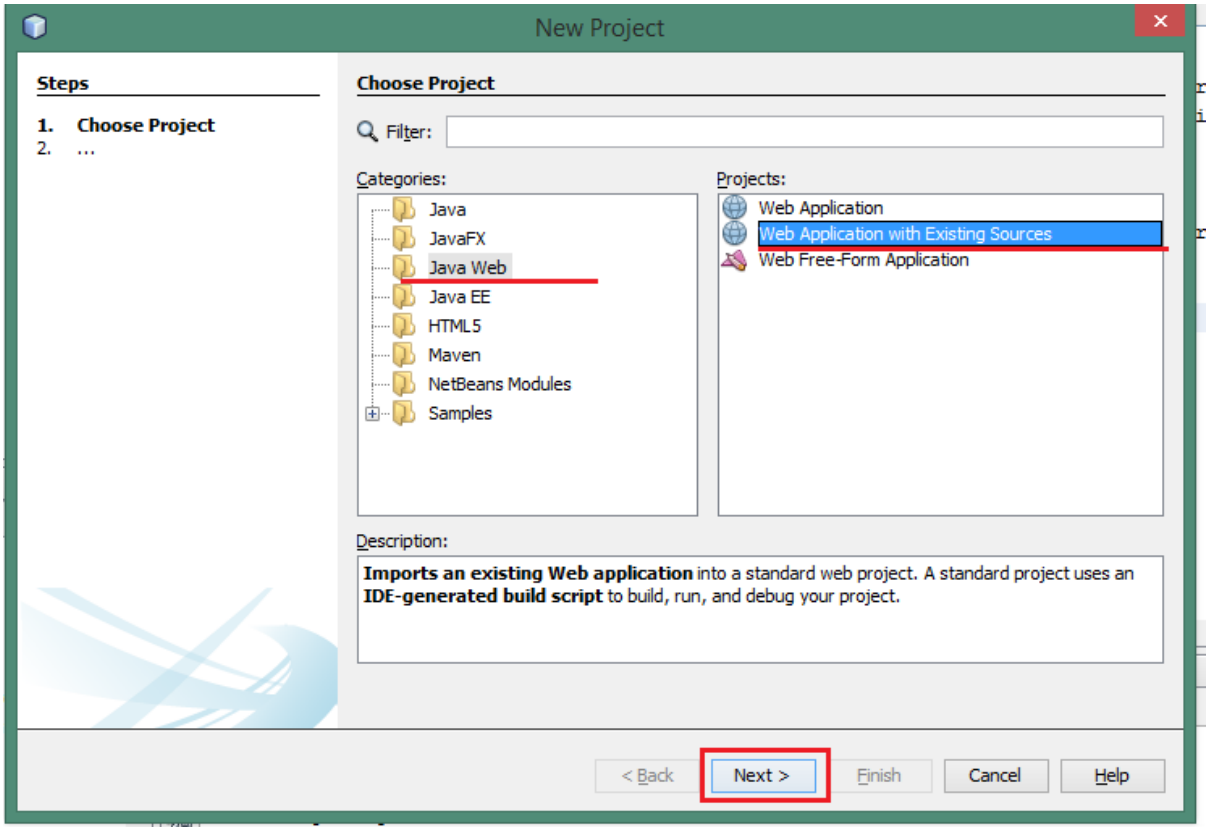
3.5 Εγκατάσταση Εφαρμογής

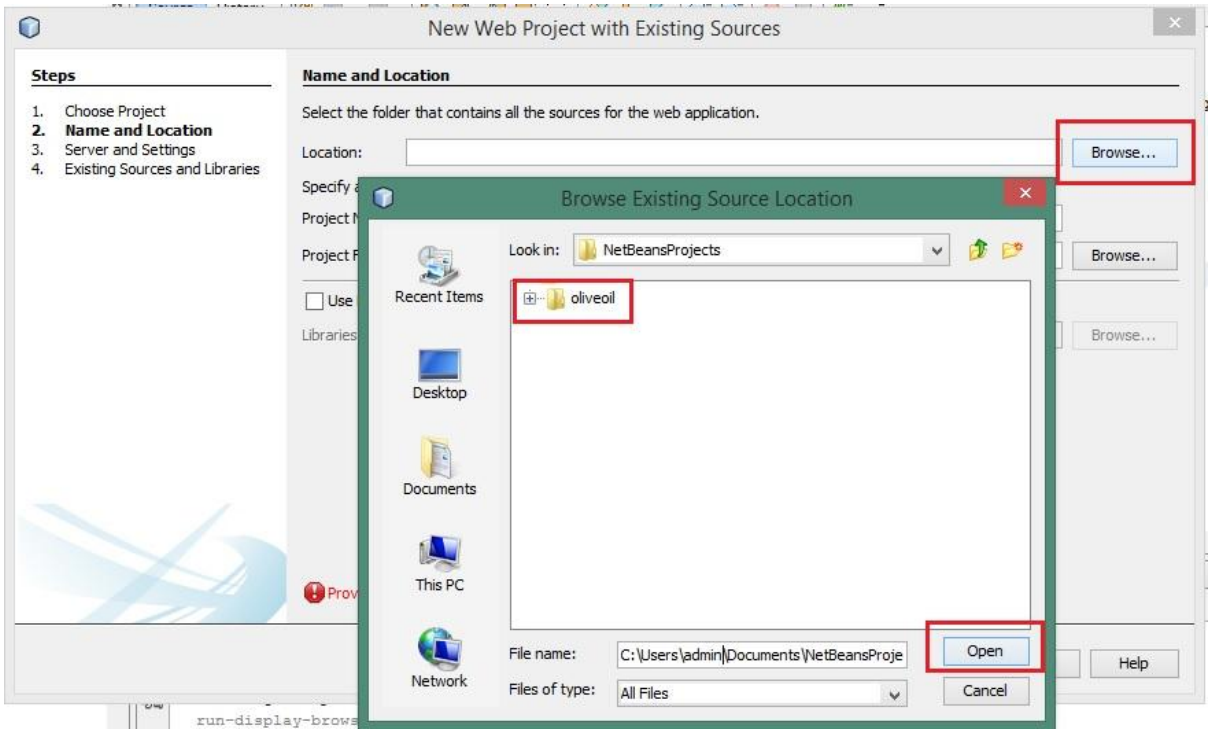
Αρχικά εκτελούμε το παρεχόμενο .sql αρχείο για τη δημιουργία της βάσης δεδομένων και των πινάκων που χρειάζονται.

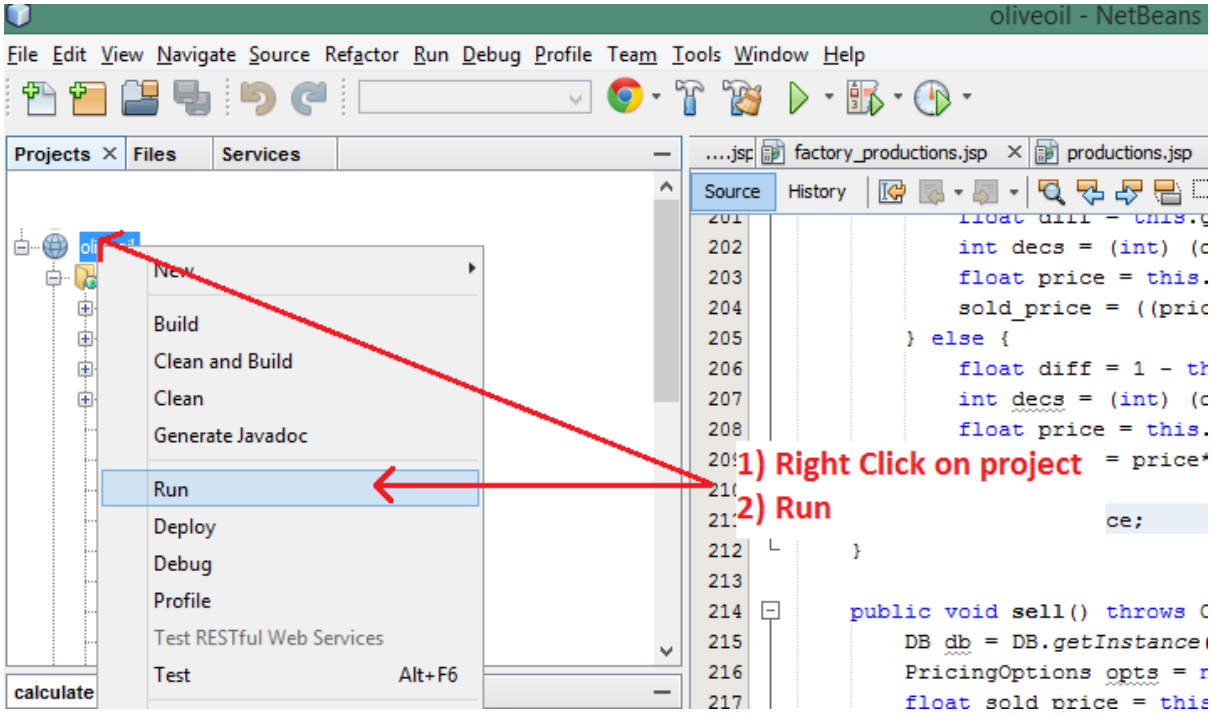
Στη συνέχεια, αφού κάνουμε extract το .zip με τον πηγαίο κώδικα της εφαρμογής, μεταφέρουμε όλο τον φάκελο στο σημείο που το NetBeans αποθηκεύει τα projects (π.χ. C:\Users\admin\Documents\NetBeansProjects).

Αφού έχουμε μεταφέρει το project στο σωστό φάκελο, ανοίγουμε το NetBeans, και επιλέγουμε File->New Project. Από το παράθυρο που θα ανοίξει επιλέγουμε “Java Web” και “Web Application with Existing Sources” και πατάμε next. Στο “location” πατάμε “Browse” και επιλέγουμε το φάκελο που μεταφέραμε. Τέλος πατάμε “Finish”.

Πριν τρέξουμε το project, βρίσκουμε στα source packages το αρχείο DB.java, και στις γραμμές 20 & 21 βάζουμε το username και το password της βάσης μας.
Ακολουθούν στιγμιότυπα της παραπάνω διαδικασίας σε εικόνες:




New Web Project with Existing Sources
Steps
1. Choose Project
2. Name and Location
3. Server and Settings
4. Existing Sources and Libraries
Name and Location
Select the folder that contains all the sources for the web application.
Location:
Browse...
Browse Existing Source Location
Look in: NetBeansProjects
Recent Items
Desktop
Documents
This PC
Network
oliveoil
File name: C:\Users\admin\Documents\NetBeansProje
Files of type: All Files
Open
Cancel
Help


oliveoil - NetBeans
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Files
Services
factory_productions.jsp
productions.jsp
Source
History
New
Build
Clean and Build
Clean
Generate Javadoc
Run
Deploy
Debug
Profile
Test RESTful Web Services
Test Alt+F6
calculate
1) Right Click on project
2) Run

4. Περιγραφή Συστήματος

4.1 Τεχνικές υλοποίησης

4.1.1 Γενικά

- Για την υλοποίηση του συστήματος ακολουθήθηκε η εξής μεθοδολογία:
Για κάθε λειτουργία έγινε διαχωρισμός του κομματιού του γραφικού περιβάλλοντος, από την εκτέλεση της λειτουργίας. Τα διάφορα μέρη του γραφικού περιβάλλοντος υλοποιήθηκαν από αρχεία jsp (ένα για κάθε λειτουργία), ενώ η επεξεργασία/εκτέλεση της κάθε λειτουργίας υλοποιήθηκε με java servlets.

- Κάθε βασικό στοιχείο της εφαρμογής (χρήστης, εργοστάσιο, παραγωγή, βάση δεδομένων) αντιστοιχίστηκε επίσης σε μία κλάση java, στην οποία περιέχονται οι ιδιότητες και οι βασικές λειτουργίες του στοιχείου.

- Για τη λειτουργία εισόδου, χρησιμοποιήθηκε τεχνική με χρήση http session, όπου κρατάει τα στοιχεία του χρήστη που συνδέεται. Μετά από την είσοδο του χρήστη, τα στοιχεία του (για την ακρίβεια ο μοναδικός κωδικός του, το όνομα χρήστη, και το αν είναι διαχειριστής ή όχι) παραμένουν στο session για όση ώρα αυτός παραμένει ενεργός. Μετά από μία ώρα αδράνειας, το session λήγει και ο χρήστης πρέπει να ξαναπεράσει από τη διαδικασία εισόδου για να χρησιμοποιήσει το σύστημα.Το http session υλοποιεί η κλάση java HttpSession η οποία και χρησιμοποιήθηκε.

- Οι χρήστες του συστήματος χωρίζονται σε 2 κατηγορίες: Διαχειριστής και Παραγωγός.
Ο Διαχειριστής μπορείς να δει όλες της παραγωγές, να προσθέσει/διαγράψει παραγωγές, χρήστες, εργοστάσια, να δει παραγωγές ανά χρήστη, ανά εργοστάσιο, καθώς και όλες τις αναφορές, όπως επίσης να αλλάξει τις τιμές ελαιόλαδου.
Ο Παραγωγός μπορεί να δει τις παραγωγές που αφορούν τον ίδιο, καθώς και να υπολογίσει την αξία των απούλητων παραγωγών.

4.1.2 Οδήγηση / Αντιστοίχηση σελίδων με λειτουργίες

Κάθε αίτηση σελίδας οδηγείται στο αρχείο index.jsp. Εκεί λαμβάνοντας υπόψιν την GET παράμετρο “action”, καθώς επίσης και το αν υπάρχει συνδεδεμένος χρήστης και αν αυτός είναι διαχειριστής ή όχι, επιλέγουμε το αντίστοιχο jsp αρχείο που πρέπει να φορτωθεί.
Έτσι για παράδειγμα το URL “*http://localhost:8084/oliveoil/?action=users*” σημαίνει ότι θα φορτωθεί το αρχείο “users.jsp”.
Κάθε φορά που κάποιος μη συνδεδεμένος χρήστης προσπαθεί να προσπελάσει κάποια εσωτερική σελίδα του συστήματος (οποιαδήποτε σελίδα δηλαδή εκτός της σελίδα εισόδου ή τη σελίδα εγγραφής)

4.1.3 Επικοινωνία με βάση δεδομένων

Για τη λειτουργία με τη βάση δεδομένων υλοποιήθηκε η κλάση DB, η οποία προσφέρει μεθόδους για την εκτέλεση SELECT, INSERT και DELETE sql εντολών.

4.2 Λειτουργίες Συστήματος

4.2.1 Λειτουργίες συνοπτικά

- Εγγραφή νέου χρήστη (register)
    - Έλεγχος μοναδικότητας ονόματος χρήστη
- Σύνδεση (login)
- Υπολογισμός αξίας παραγωγής με βάση τις τρέχουσες τιμές (παραγωγός)
- Διαγραφή Χρήστη (διαχειριστής)
- Προσθήκη Εργοστασίου (διαχειριστής)
- Διαγραφή Εργοστασίου (διαχειριστής)
- Αλλαγή τιμών ελαιόλαδου (διαχειριστής)
- Προσθήκη παραγωγής (διαχειριστής)
- Διαγραφή παραγωγής (διαχειριστής)
- Μαρκάρισμα παραγωγής ως πουλημένη(διαχειριστής)
- Λίστα παραγωγών χρήστη (διαχειριστής)
- Λίστα παραγωγών εργοστασίου (διαχειριστής)
- Αναζήτηση χρήστη (διαχειριστής)
- Αναφορές (διαχειριστής)
    - Συνολικά στατιστικά συνεταιρισμού
    - Στατιστικά ανα εργοστάσιο
    - Στατιστικά ανά χρήστη

4.2.2 Περιγραφή λειτουργιών

Εγγραφή νέου χρήστη

Η εγγραφή νέου χρήστη γίνεται με τη συμπλήρωση φόρμας με τα απαιτούμενα στοιχεία που βρίσκεται στη σχετική σελίδα (?action=register υλοποιημένη από το register.jsp). Τα ζητούμενα στοιχεία είναι τα εξής (όλα υποχρεωτικά):

- Όνομα (πραγματικό)
- Όνομα χρήστη
- Κωδικός
- Α.Φ.Μ.
- Τηλέφωνο
- Διεύθυνση
- Email
- Αριθμός τραπέζης

Με την υποβολή της φόρμας τα στοιχεία αποστέλλονται στο αντίστοιχο servlet (RegisterAction), το οποίο αφού συλλέξει όλα τα δεδομένα που έχουν αποσταλεί πραγματοποιεί τους εξής ελέγχους:

- Έλεγχος συμπλήρωσης όλων των πεδίων
- Έλεγχος μοναδικότητας ονόματος χρήστη

Αν όλοι οι έλεγχοι περάσουν επιτυχώς, τότε ο χρήστης καταχωρείται στη βάση δεδομένων και το servlet ανακατευθύνει τον χρήστη στη σελίδα login.
Σε αντίθετη περίπτωση (π.χ. ο χρήστης συμπλήρωσε όνομα χρήστη που χρησιμοποιείται ήδη) το servlet ανακατευθύνει τον χρήστη στη σελίδα του register δείχνοντας το σχετικό σφάλμα.
Κατά την καταχώρηση των πεδίων στη βάση δεδομένων, ο κωδικός κρυπτογραφείται με την συνάρτηση MD5 για λόγους ασφαλείας.

Σύνδεση

Ο χρήστης συμπληρώνει την αντίστοιχη φόρμα με το όνομα χρήστη και τον κωδικό, και τα στοιχεία στέλνονται στο servlet που υλοποιεί τη διαδικασία του login. Εκεί, αφού ο κωδικός κρυπτογραφηθεί με τη μέθοδο MD5 για να μπορέσει να συγκριθεί με τον αντίστοιχο κωδικό στη βάση δεδομένος, γίνεται έλεγχος για το αν υπάρχει εγγραφή στη βάση με το συγκεκριμένο όνομα χρήστη και κωδικό.
Αν ο χρήστης βρεθεί, τότε δημιουργείται session που αποθηκεύει τα στοιχεία του χρήστη που έχει συνδεθεί, αλλιώς το servlet ανακατευθύνει και πάλι στη φόρμα εισόδου δείχνοντας το σχετικό σφάλμα. Το session που δημιουργείται είναι διάρκειας μίας ώρας – πράγμα που σημαίνει μετά από μία ώρα αδράνειας του χρήστη, το session αυτό λήγει και διαγράφεται, και ο χρήστης θα πρέπει να κάνει εκ νέου τη διαδικασία εισόδου για να χρησιμοποιήσει το σύστημα.

Υπολογισμός αξίας παραγωγής με βάση τις τρέχουσες τιμές (παραγωγός)

Ο χρήστης – παραγωγός αφού κάνει login, μπορεί να δει τη λίστα παραγωγών που είναι καταχωρημένες στο σύστημα και αφορούν τον ίδιο. Τα στοιχεία που του παρουσιάζονται σχετικά με τις παραγωγές του είναι τα εξής:

- Ημερομηνία παραγωγής
- Βάρος ελιάς
- Βάρος παραγόμενου ελαιόλαδου
- Οξύτητα
- Βιολογικό ή όχι
- Αλεστικά έξοδα
- Τιμή (αν έχει ήδη πωληθεί)

Για όσες από τις παραγωγές δεν έχουν ήδη πουληθεί, μπορεί να επιλέξει να δει την αξία τους τη δεδομένη χρονική στιγμή (Αυτό θα πει πόσο θα πωλούνταν η παραγωγή με βάση τις τιμές που ισχύουν τη συγκεκριμένη χρονική στιγμή).
Ο υπολογισμός της αξίας γίνεται (στην κλάση Production που αφορά τις παραγωγές) λαμβάνοντας υπόψιν την οξύτητα της συγκεκριμένης παραγωγής, το αν αφορά βιολογικό ή μη ελαιόλαδο, καθώς και την τρέχουσα τιμή του συγκεκριμένου τύπου ελαιόλαδου.
Συγκεκριμένα ο τύπος που μας δίνει την αξία της παραγωγής είναι ο εξής:
Αξία = (T+A)*B-K
Όπου:
T: Τιμή κιλού (βιολογικού ή όχι)
A: Ποσό που προκύπτει από το έξτρα κόστος ελαιόλαδου ανά 0.1 οξύτητας (θετικό για οξύτητα > 1, αρνητικό για οξύτητα <1). Π.χ. αν κάθε 0.1 οξύτητας κοστίζει 0.1 ευρώ, τότε για οξύτητα 1.2 A= 2*0.1, για οξύτητα 0.8 A=-(2*0.1)
B: Βάρος παραγωγής σε κιλά
K: Κόστος παραγωγής

Λίστα Εργοστασίων (διαχειριστής)

Ο διαχειριστής μπορεί να δει τη λίστα των καταχωρημένων εργοστασίων μαζί με όλα τα στοιχεία τους (διεύθυνση, ιδιοκτήτης, τηλέφωνο). Από τη λίστα μπορούμε να επιλέξουμε να διαγράψουμε κάποιο εργοστάσιο ή να δούμε της παραγωγές κάποιου.
Στο σχετικό servlet (factories.jsp) με την μέθοδο getAllFactories() της κλάσης Factory ανακτούμε όλα τα εργοστάσια από τη βάση δεδομένων, και στη συνέχεια τα διατρέχουμε ώστε να πάρουμε όλα τα στοιχεία τους.
Στην ίδια σελίδα υπάρχει επιλογή για προσθήκη εργοστασίου.

Προσθήκη Εργοστασίου (διαχειριστής)

Ο διαχειριστής μπορεί να προσθέσει εργοστάσιο συμπληρώνοντας τη σχετική φόρμα στην αντίστοιχη σελίδα (?action=add_factory – υλοποιημένη από το add_factory.jsp). Με την υποβολή της φόρμας τα στοιχεία στέλνονται στον σχετικό servlet (AddFactoryAction), το οποίο ελέγχει αν όλα τα πεδία έχουν συμπληρωθεί, και προσθέτει τη σχετική εγγραφή στη βάση δεδομένων ή αν υπάρχει σφάλμα επιστρέφει στη φόρμα σημειώνοντας το σχετικό σφάλμα.

Διαγραφή Εργοστασίου (διαχειριστής)

Στη λίστα με τα εργοστάσια, ο διαχειριστής έχει την επιλογή να διαγράψει όποιο από αυτά επιθυμεί. Αυτό γίνεται με ανακατεύθυνση στο σχετικό servlet (DeleteFactoryAction), στέλνοντας σαν GET παράμετρο στο url το id του εργοστασίου που πρέπει να διαγραφεί.
Το servlet διαγράφει το εργοστάσιο και επιστρέφει στη σελίδα με τη λίστα εργοστασίων.

Αλλαγή τιμών ελαιόλαδου (διαχειριστής)

Πηγαίνοντας στις ρυθμίσεις συστήματος (?action=options – υλοποιημένη από το options.jsp), ο διαχειριστής μπορεί να αλλάξει την τιμή του ελαιόλαδου (βιολογικού ή μη) καθώς και το Ποσό μεταβολής τιμής / 0.1 οξύτητας.
Συμπληρώνει τα σχετικά πεδία της φόρμας, με την υποβολή της οποίας τα δεδομένα στέλνονται στο SaveOptionsAction servlet. Εκεί γίνονται οι σχετικοί έλεγχοι (συμπλήρωση όλων των πεδίων, αριθμητικός τύπος δεδομένων), και αν όλα είναι σωστά προχωράει στην ενημέρωση των πεδίων στη βάση δεδομένων και επιστρέφει στη σελίδα ρυθμίσεων.

Προσθήκη παραγωγής (διαχειριστής)

Στη σελίδα με τη λίστα των χρηστών/παραγωγών, ο διαχειριστής μπορεί να επιλέξει να προσθέσει παραγωγή σε κάποιον από αυτούς. Πατώντας την αντίστοιχη επιλογή, μεταφέρεται στη σελίδα με τη φόρμα Εισαγωγής Παραγωγής (?action=add_production, add_production.jsp).
Σε αυτή τη σελίδα υπάρχει ένα μενού επιλογής εργοστασίου, το οποίο συμπληρώνεται μετά από σχετικό ερώτημα στη βάση δεδομένων το οποίο μας δίνει όλα τα διαθέσιμα εργοστάσια.
Αφού επιλεχτεί εργοστάσιο και συμπληρωθούν και τα αντίστοιχα πεδία, η φόρμα αποστέλλεται στο servlet που είναι υπεύθυνο για την εισαγωγή παραγωγής (AddProductionAction). Εκεί αφού γίνει έλεγχος για την παρουσία και την εγκυρότητα των πεδίων, η παραγωγή γράφεται στη βάση δεδομένων και το Servlet μας ανακατευθύνει στη σελίδα με τη λίστα παραγωγών του συγκεκριμένου χρήστη, ενημερώνοντας για την προσθήκη με σχετικό μήνυμα.

Διαγραφή παραγωγής (διαχειριστής)

Στη λίστα παραγωγών (είτε ενός χρήστη είτε ενός εργοστασίου), ο διαχειριστής μπορεί να επιλέξει να διαγράψει κάποια από αυτές.
Πατώντας το σχετικό εικονίδιο μεταφέρεται στο servlet που είναι υπεύθυνο για τη διαγραφή παραγωγής (DeleteProductionAction), περνώντας ως όρισμα GET το id της παραγωγής που είναι προς διαγραφή. Εκεί αφού γίνει η διαγραφή, γίνεται ανακατεύθυνση στην προηγούμενη λίστα παραγωγών, ενημερώνοντας μας για τη διαγραφή με σχετικό μήνυμα.

Μαρκάρισμα παραγωγής ως πουλημένη(διαχειριστής)

Στη λίστα παραγωγών (είτε ενός χρήστη είτε ενός εργοστασίου), ο διαχειριστής μπορεί να επιλέξει να σημειώσει κάποια από αυτές σαν «πουλημένη».
Πατώντας το σχετικό εικονίδιο μεταφέρεται στο servlet που είναι υπεύθυνο για την πώληση παραγωγής (SoldProductionAction), περνώντας ως όρισμα GET το id της παραγωγής που είναι προς πώληση. Εκεί, αφού υπολογιστεί η τιμή πώλησης της παραγωγής με βάση τις τρέχουσες τιμές, η παραγωγή μαρκάρεται ως πουλημένη.

Λίστα παραγωγών χρήστη (διαχειριστής)

.Από τη λίστα χρηστών ο διαχειριστής μπορεί να επιλέξει να δει τις παραγωγές ενός συγκεκριμένου χρήστη. Από την κλάση διαχείρισης των παραγωγών (Class Production), χρησιμοποιώντας την μέθοδο *getAllProductionsForUser(userId)*, η οποία τραβάει από τη βάση δεδομένων όλες τις παραγωγές του συγκεκριμένου χρήστη.

Στη συνέχεια εμφανίζονται όλες οι παραγωγές σε μία λίστα, παρουσιάζοντας όλα τα στοιχεία τους, μαζί με τις δυνατές επιλογές για την κάθε μία (πώληση/διαγραφή).

Λίστα παραγωγών εργοστασίου (διαχειριστής)

Από τη λίστα εργοστασίων ο διαχειριστής μπορεί να επιλέξει να δει τις παραγωγές ενός συγκεκριμένου εργοστασίου. Από την κλάση διαχείρισης των εργοστασίων (Class Factory), χρησιμοποιώντας την μέθοδο *getAllProductions ()*, η οποία τραβάει από τη βάση δεδομένων όλες τις παραγωγές του συγκεκριμένου εργοστασίου.

Στη συνέχεια εμφανίζονται όλες οι παραγωγές σε μία λίστα, παρουσιάζοντας όλα τα στοιχεία τους, μαζί με τις δυνατές επιλογές για την κάθε μία (πώληση/διαγραφή).

Αναζήτηση χρήστη (διαχειριστής)

Στη σελίδα με τη λίστα των χρηστών, ο διαχειριστής μπορεί να αναζητήσει έναν χρήστη βάσει του ονόματος ή του ΑΦΜ του. Συμπληρώνοντας το σχετικό πεδίο και πατώντας υποβολή, στέλνεται ο όρος αναζήτησης με τη μέθοδο POST και χρησιμοποιώντας τη μέθοδο search(term) της κλάσης User, ανακτούμε όλους τους χρήστες που το όνομα ή το ΑΦΜ τους ταιριάζει με τον όρο αναζήτησης.

Αν η φόρμα υποβληθεί χωρίς να συμπληρωθεί κάποιος όρος, τότε τα αποτελέσματα είναι όλοι οι χρήστες.

Αναφορές - Συνολικά στατιστικά συνεταιρισμού

Επιλέγοντας «Συνολικές αναφορές» ο διαχειριστής μπορεί να δει συνολικά τον όγκο των παραγωγών καθώς και το σύνολο εισπράξεων για όλο το ελαιόλαδο του συνεταιρισμού, καθώς επίσης και ξεχωριστά για βιολογικό και μη βιολογικό ελαιόλαδο.

Τα στοιχεία αυτά παράγονται από την κλάση Production, η οποία προσφέρει τις μεθόδους getTotalOlivesWeight(), getTotalOilWeight(), getAverageAcidity(), getTotalIncome(), οι οποίες με χρήση των μεθόδων SUM και AVG της MySQL υπολογίζουν τα αθροίσματα και τους μέσους όρους.

Αναφορές - Στατιστικά ανά εργοστάσιο

Αντίστοιχα με τα συνολικά στατιστικά μπορεί να δει ο διαχειριστής και για όλα τα εργοστάσια ξεχωριστά. Επιλέγοντας «Αναφορές ανά εργοστάσιο» ο διαχειριστής βλέπει τα συνολικά στοιχεία παραγωγής για κάθε εργοστάσιο (Όγκος ελιάς, όγκος λαδιού, μέση οξύτητα, σύνολο εισπράξεων).

Τα στοιχεία αυτά παράγονται από την κλάση Production, η οποία προσφέρει τη μέθοδο getProductionsInfoByFactory(), η οποία με χρήση των μεθόδων SUM, AVG και GROUP BY της MySQL φέρνει τα συνολικά στοιχεία ομαδοποιημένα ανά εργοστάσιο.
Για κάθε εργοστάσιο στη λίστα ο χρήστης μπορεί να επιλέξει να δει αναλυτικότερη αναφορά για κάποιο εργοστάσιο, με ομαδοποιημένες παραγωγές ανά χρήστη, το οποίο γίνεται από την κλάση Factory μέθοδος getAllProductionsByUser(), με χρήση ξανά των μεθόδων SUM και GROUP BY της MySQL.

Αναφορές - Στατιστικά ανά χρήστη

Στην τελευταία αυτή κατηγορία αναφορών, ο διαχειριστής μπορεί να δει συνολικά στατιστικά ανά χρήστη (όγκος ελιάς, όγκος λαδιού, μέση οξύτητα, σύνολο εισπράξεων).
Κατ' αντιστοιχία με τις προηγούμενες κατηγορίες αναφορών, η κλάση Production προσφέρει τη μέθοδο getProductionsInfoByUser(), η οποία με χρήση των μεθόδων SUM και AVG της MySQL υπολογίζουν τα αθροίσματα και τους μέσους όρους ομαδοποιημένα ανά χρήστη.

4.3 Περιγραφή Βάσης Δεδομένων

4.3.1 Πίνακες

Οι πίνακες της βάσης δεδομένων που προέκυψαν μετά την ανάλυση των απαιτήσεων είναι οι εξής:

Table `user`

id	username	password	email	name	address	phone	afm	bank_account	Is_admin

Table `factory`

id	name	address	owner	phone

Table `production`

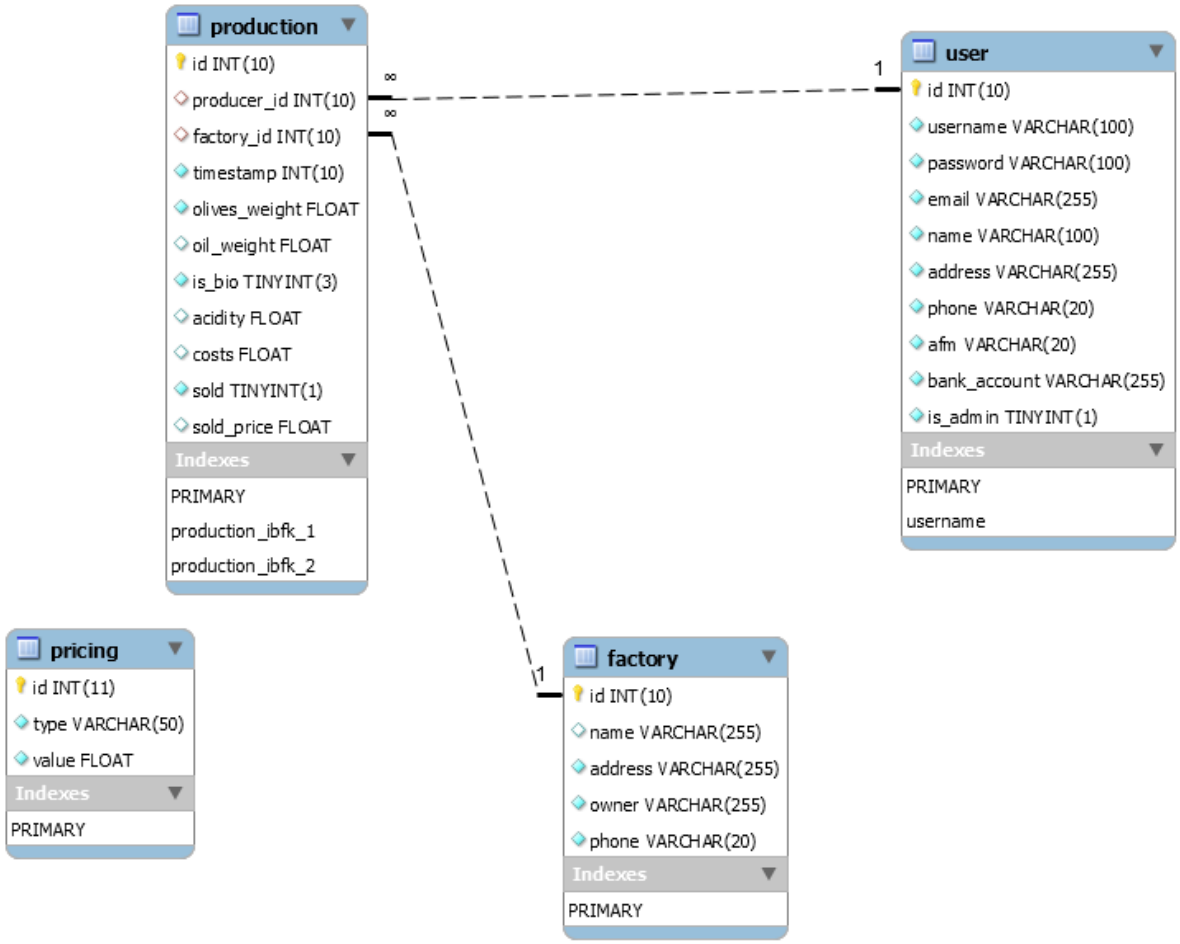
id	producer_id	factory_id	timestamp	olives_weight	oil_weight	is_bio	acidity	costs	sold_price

Table `pricing`

id	type	value

Σε όλους τους πίνακες υπάρχει ένας μοναδικός κωδικός για κάθε εγγραφή που εισάγεται, ο οποίος δημιουργείται αυτόματα από το σύστημα.

4.3.2 Σχεσιακό μοντέλο



<u>Σχόλιο</u>

Όπως φαίνεται και από το σχεσιακό μοντέλο, ο πίνακας production περιέχει δύο ξένα κλειδιά:

Producer_id – αναφέρεται στον πίνακα user

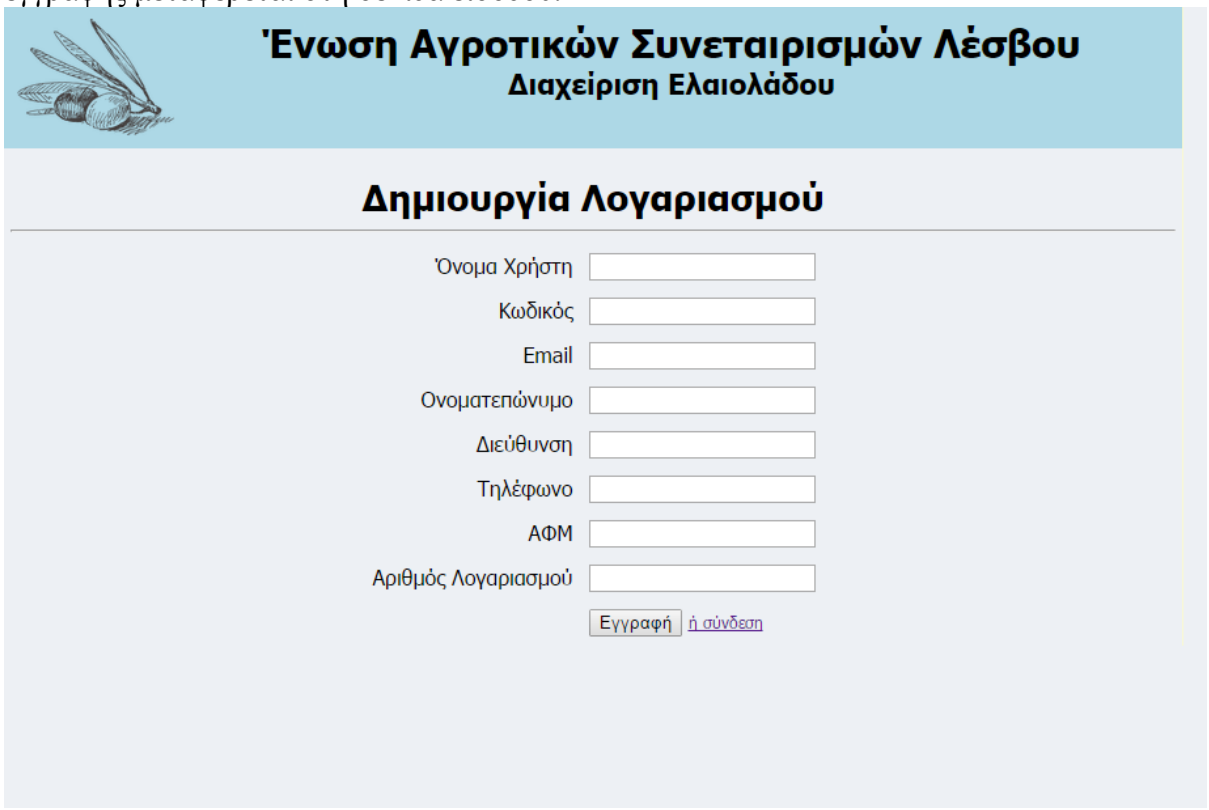
Factory_id – αναφέρεται στον πίνακα factory.

Και τα δύο κλειδιά έχουν προστεθεί με την επιλογή «ON DELETE SET NULL», ώστε κατά τη διαγραφή εγγραφών από των πίνακα user ή factory να μην υπάρχουν conflicts, ούτε να χρειάζεται να διαγράφονται οι συγκεκριμένες παραγωγές.

5. Παρουσίαση Συστήματος

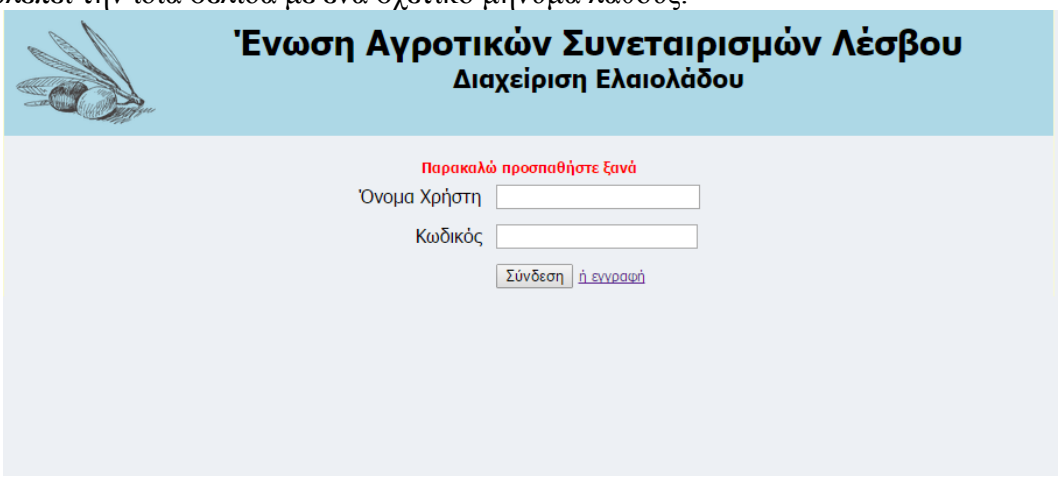
5.1 Εγγραφή χρήστη

Σελίδα εγγραφής Χρήστη (?action=register). Ο χρήστης συμπληρώνει τη φόρμα με τα στοιχεία του και αποστέλλει τη φόρμα για επεξεργασία. Σε περίπτωση λάθους ο χρήστης επιστρέφει στην ίδια σελίδα βλέποντας ένα μήνυμα λάθους – σε περίπτωση επιτυχούς εγγραφής μεταφέρεται στη σελίδα εισόδου.



5.2 Είσοδος στο σύστημα

Σελίδα εισόδου στο σύστημα (?action=login). Ο χρήστης συμπληρώνει το όνομα χρήστη και τον κωδικό του και αποστέλλει την φόρμα. Σε περίπτωση μη επιτυχούς σύνδεσης ο χρήστης ξαναβλέπει την ίδια σελίδα με ένα σχετικό μήνυμα λάθους.



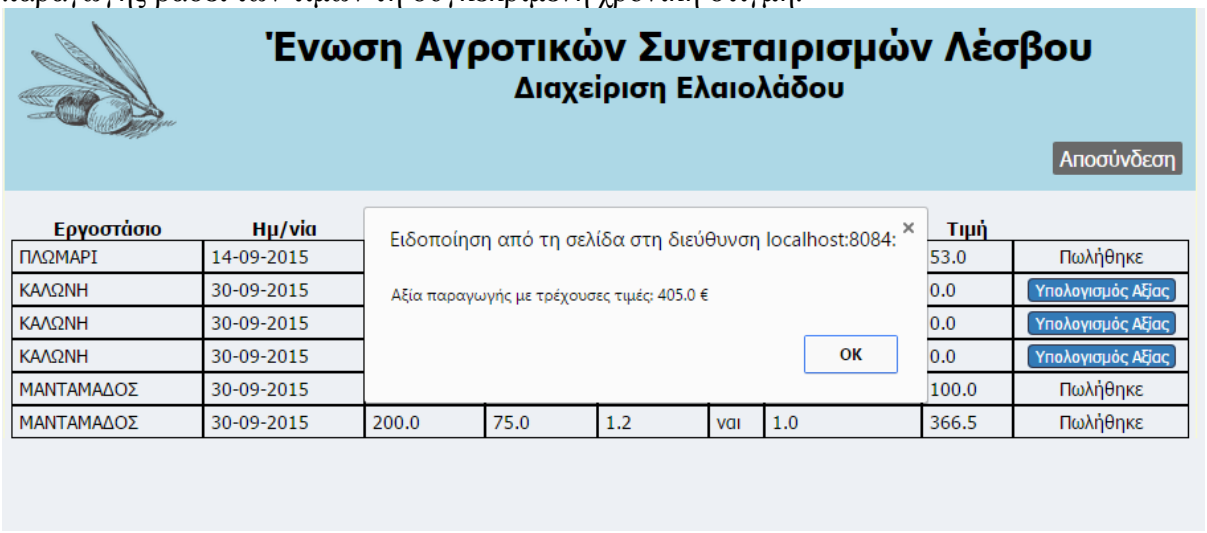
5.3 Σελίδα χρήστη (παραγωγές)

Η σελίδα που βλέπει ο απλός χρήστης μετά την είσοδό του στο σύστημα (?action=main). Του παρουσιάζονται όλες οι παραγωγές μαζί με όλα τα στοιχεία τους. Σε περίπτωση που έχουν πωληθεί βλέπει την τιμή πώλησης, σε αντίθετη περίπτωση έχει επιλογή υπολογισμού αξίας με βάσει τις τωρινές τιμές.

Εργοστάσιο	Ημ/νία	Ελιές(kg)	Λάδι(kg)	Οξύτητα	BIO	Αλεστικά(€)	Τιμή	
ΠΛΩΜΑΡΙ	14-09-2015	200.0	30.0	0.5	όχι	44.5	53.0	Πωλήθηκε
ΚΑΛΩΝΗ	30-09-2015	333.0	111.0	1.1	όχι	88.0	0.0	Υπολογισμός Αξίας
ΚΑΛΩΝΗ	30-09-2015	999.0	99.0	1.0	ναι	90.0	0.0	Υπολογισμός Αξίας
ΚΑΛΩΝΗ	30-09-2015	888.0	88.0	1.0	ναι	80.0	0.0	Υπολογισμός Αξίας
ΜΑΝΤΑΜΑΔΟΣ	30-09-2015	200.0	75.0	1.2	όχι	1.0	100.0	Πωλήθηκε
ΜΑΝΤΑΜΑΔΟΣ	30-09-2015	200.0	75.0	1.2	ναι	1.0	366.5	Πωλήθηκε

5.4 Υπολογισμός αξίας παραγωγής (με τρέχουσες τιμές)

Πατώντας την επιλογή υπολογισμού αξίας, ο χρήστης ενημερώνεται για την τιμή της παραγωγής βάσει των τιμών τη συγκεκριμένη χρονική στιγμή.



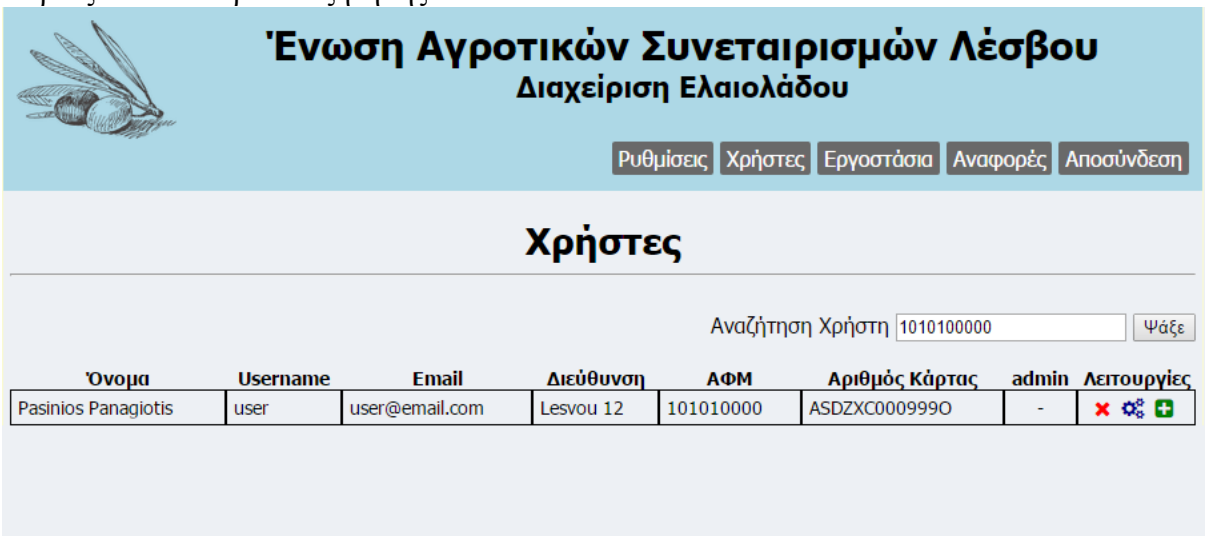
Εργοστάσιο	Ημ/νία						Τιμή	
ΠΛΩΜΑΡΙ	14-09-2015						53.0	Πωλήθηκε
ΚΑΛΩΝΗ	30-09-2015						0.0	Υπολογισμός Αξίας
ΚΑΛΩΝΗ	30-09-2015						0.0	Υπολογισμός Αξίας
ΚΑΛΩΝΗ	30-09-2015						0.0	Υπολογισμός Αξίας
ΜΑΝΤΑΜΑΔΟΣ	30-09-2015						100.0	Πωλήθηκε
ΜΑΝΤΑΜΑΔΟΣ	30-09-2015	200.0	75.0	1.2	ναι	1.0	366.5	Πωλήθηκε

5.5 Λίστα χρηστών

Η αρχική σελίδα του διαχειριστή είναι η σελίδα των χρηστών (?action=users). Σε αυτή βλέπει μία λίστα με όλους τους χρήστες και όλα τα στοιχεία τους, μαζί με επιλογές διαγραφής, προβολής παραγωγών ή προσθήκης παραγωγής για κάθε χρήστη.

5.6 Αναζήτηση Χρήστη

Στην ίδια σελίδα υπάρχει επιλογή αναζήτησης χρήστη βάσει ονόματος ή βάσει Α.Φ.Μ. Ο διαχειριστής συμπληρώνει το πεδίο της φόρμας αναζήτησης με τον όρο που αναζητεί, και με την υποβολή της φόρμας επανέρχεται στην ίδια σελίδα βλέποντας μόνο τους χρήστες που ταιριάζουν στον όρο αναζήτησης που έθεσε.



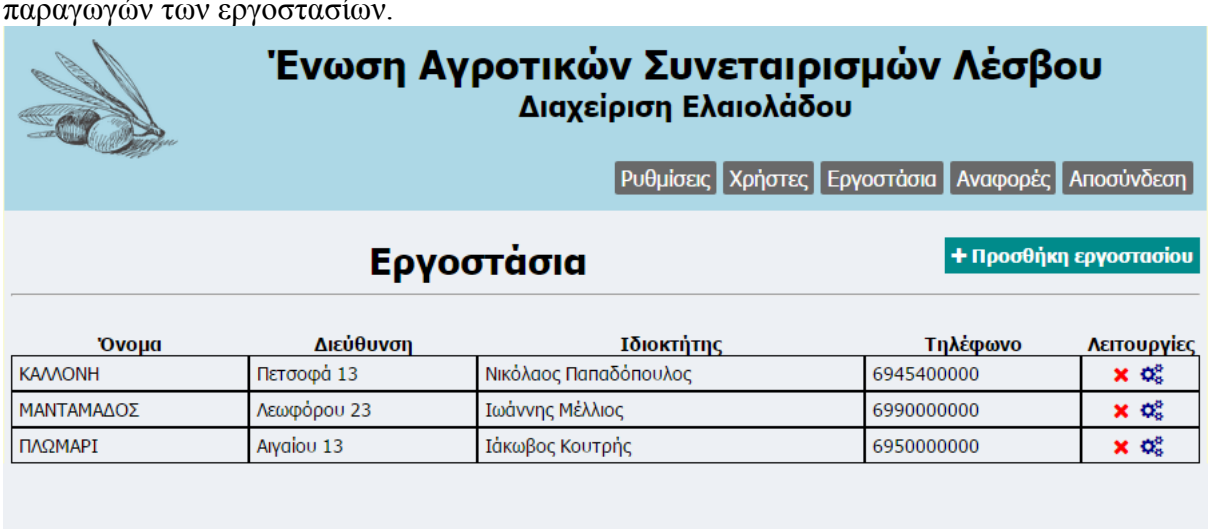
5.7 Παραγωγές χρήστη

Επιλέγοντας να δει τις παραγωγές ενός χρήστη, ο χρήστης μεταφέρεται (?action=productions) και βλέπει μία λίστα με όλα τα στοιχεία της παραγωγής του χρήστη. Για κάθε παραγωγή έχει επιλογή διαγραφής και πώλησης (ή αναίρεσης πώλησης).

Εργοστάσιο	Ημ/νία	Ελιές(kg)	Λάδι(kg)	Οξύτητα	BIO	Αλεστικά(€)	Τιμή	Λειτουργίες
ΜΑΝΤΑΜΑΔΟΣ	08-10-2015	111.0	11.0	1.0	όχι	1.0	32.0	
ΚΑΛΛΟΝΗ	09-10-2015	100.0	10.0	0.95	όχι	1.0	99.0	
ΚΑΛΛΟΝΗ	09-10-2015	122.0	10.0	1.0	όχι	1.0	99.0	
ΚΑΛΛΟΝΗ	09-10-2015	102.0	10.0	0.9	όχι	1.0	99.5	

5.8 Λίστα εργοστασίων

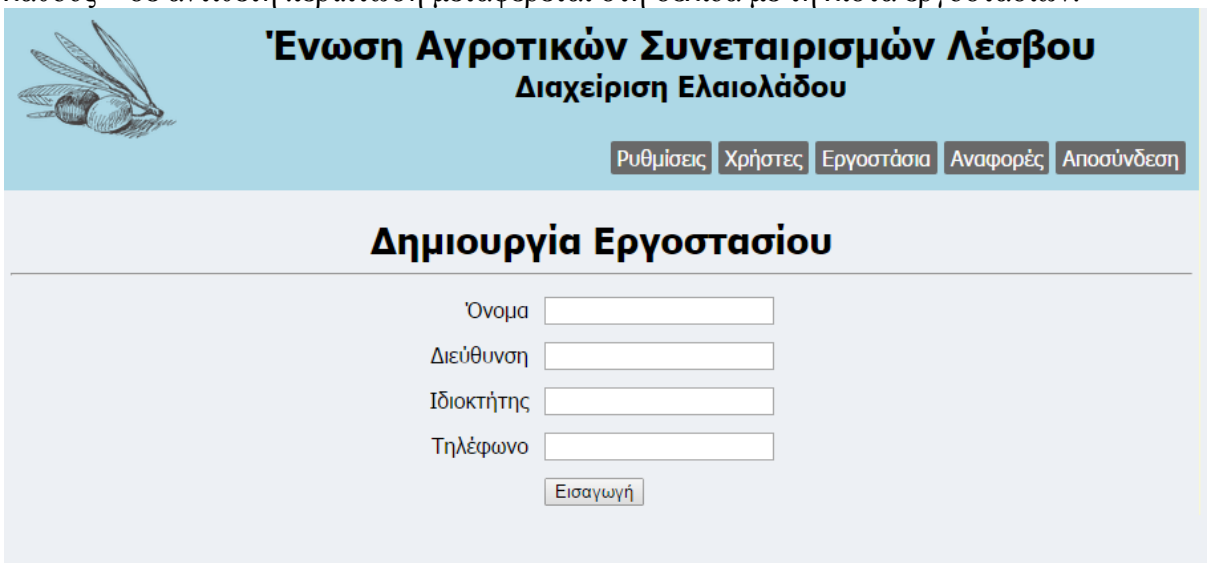
Στη σελίδα με τη λίστα εργοστασίων (?action=factories), ο διαχειριστής βλέπει τα στοιχεία των καταχωρημένων εργοστασίων, και έχει τις επιλογές διαγραφής ή προβολής των παραγωγών των εργοστασίων.



Όνομα	Διεύθυνση	Ιδιοκτήτης	Τηλέφωνο	Λειτουργίες
ΚΑΛΛΟΝΗ	Πετσοφά 13	Νικόλαος Παπαδόπουλος	6945400000	
ΜΑΝΤΑΜΑΔΟΣ	Λεωφόρου 23	Ιωάννης Μέλλιος	6990000000	
ΠΛΩΜΑΡΙ	Αιγαίου 13	Ιάκωβος Κουτρής	6950000000	

5.9 Προσθήκη εργοστασίου

Επιλέγοντας να προσθέσει εργοστάσιο (?action=add_factory), παρουσιάζεται στον χρήστη μία φόρμα συμπλήρωσης των στοιχείων του εργοστασίου. Σε περίπτωση μη έγκυρης συμπλήρωσης των πεδίων, ο χρήστης επανέρχεται στην ίδια σελίδα βλέποντας ένα μήνυμα λάθους – σε αντίθετη περίπτωση μεταφέρεται στη σελίδα με τη λίστα εργοστασίων.



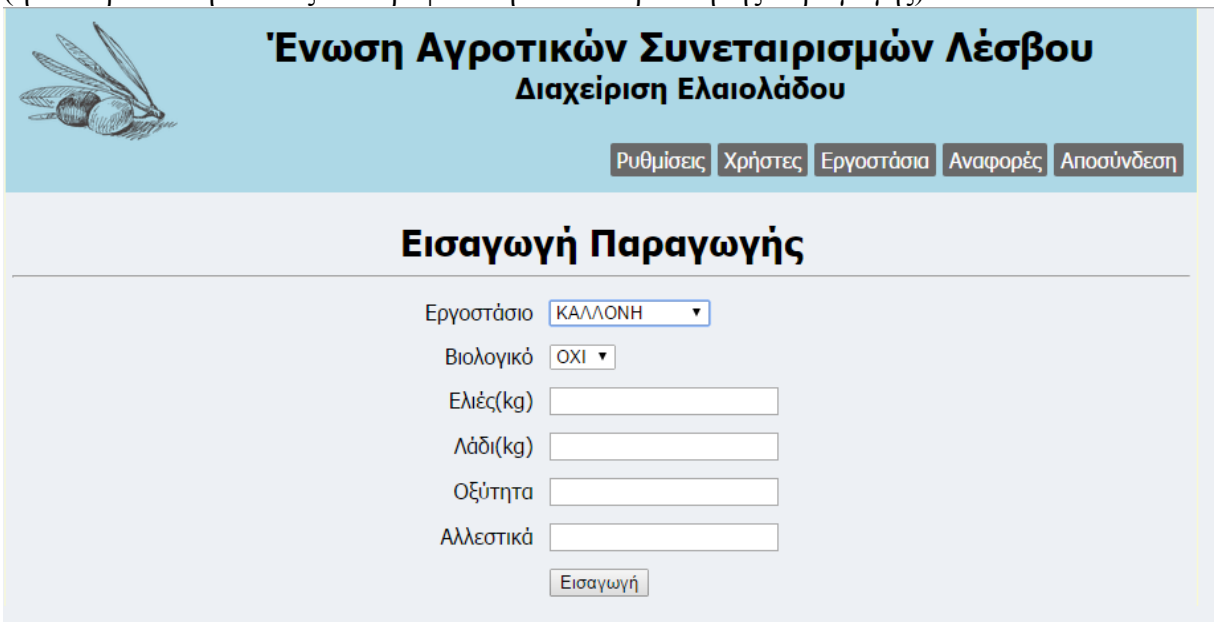
5.10 Παραγωγές Εργοστασίου

Επιλέγοντας να δει τις παραγωγές ενός εργοστασίου, ο χρήστης μεταφέρεται (?action= factory_productions) και βλέπει μία λίστα με όλα τα στοιχεία της παραγωγής του εργοστασίου. Για κάθε παραγωγή έχει επιλογή διαγραφής και πώλησης (ή αναίρεσης πώλησης).

Χρήστης	Ημ/νία	Ελιές(kg)	Λάδι(kg)	Οξύτητα	BIO	Αλεστικά(€)	Τιμή	Λειτουργίες
Pasinios Panagiotis	30-09-2015	333.0	111.0	1.1	όχι	88.0	0.0	
Pasinios Panagiotis	30-09-2015	999.0	99.0	1.0	ναι	90.0	0.0	
Pasinios Panagiotis	30-09-2015	888.0	88.0	1.0	ναι	80.0	0.0	
Vasilis Iosifellis	09-10-2015	100.0	10.0	0.95	όχι	1.0	99.0	
Vasilis Iosifellis	09-10-2015	122.0	10.0	1.0	όχι	1.0	99.0	
Vasilis Iosifellis	09-10-2015	102.0	10.0	0.9	όχι	1.0	99.5	

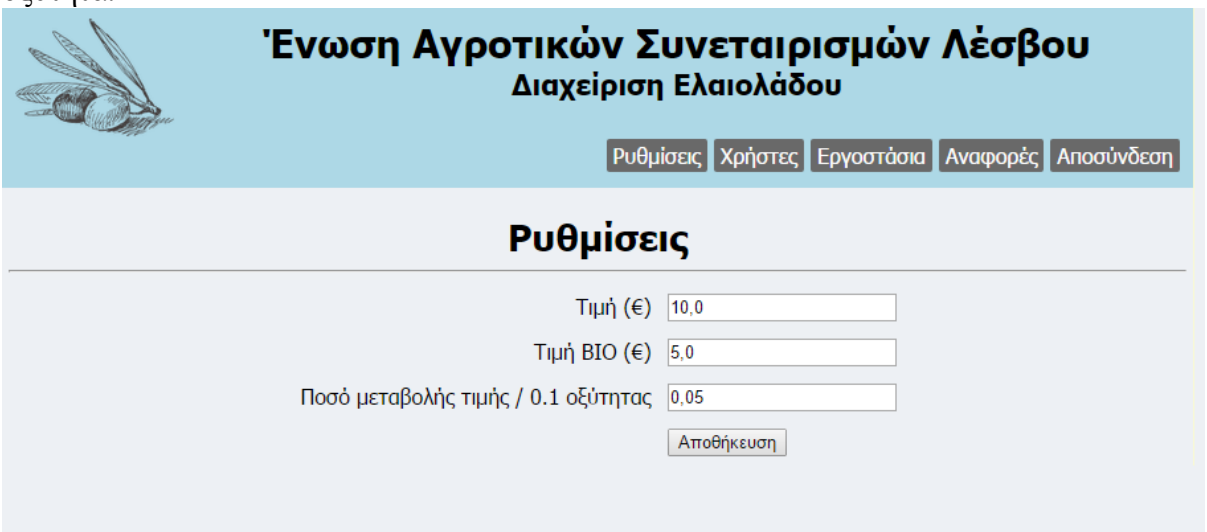
5.11 Προσθήκη παραγωγής

Από τη σελίδα με τη λίστα χρηστών, ο διαχειριστής μπορεί να επιλέξει να προσθέσει μία παραγωγή για κάποιον συγκεκριμένο χρήστη. Μεταβαίνοντας στη σχετική σελίδα (?action=productions) υπάρχει μία φόρμα με τα απαραίτητα πεδία συμπλήρωσης για την παραγωγή. Με την υποβολή της φόρμας, μεταβαίνει στη λίστα των παραγωγών του χρήστη (ή σε περίπτωση λάθους επιστρέφει στη σελίδα προσθήκης παραγωγής)



5.12 Επεξεργασία τιμών

Πηγαίνοντας στις Ρυθμίσεις (?action=options), ο διαχειριστής μπορεί να θέσει τις τιμές του ελαιόλαδου, του βιολογικού ελαιόλαδου, καθώς και του ποσού μεταβολής τιμής ανά οξύτητα.



5.13 Αναφορές

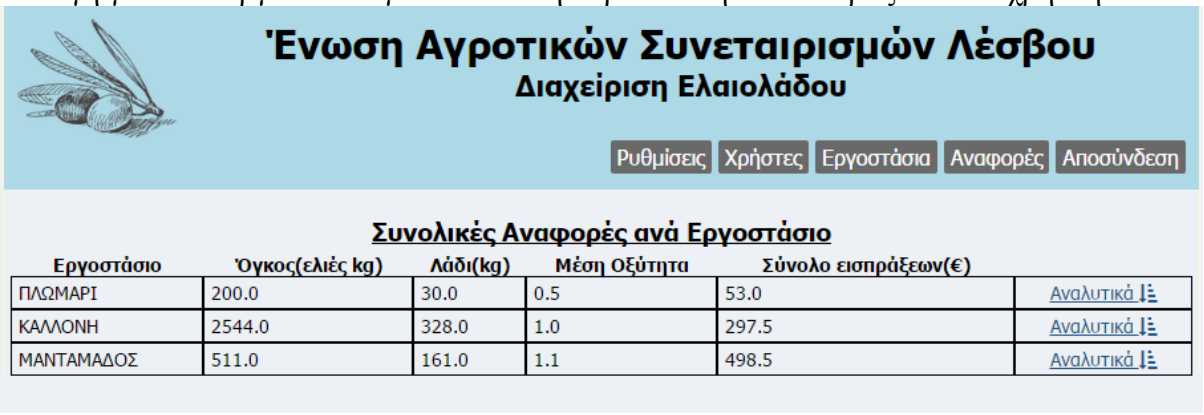
Στην κεντρική σελίδα των αναφορών (?action=reports) ο διαχειριστής μπορεί να επιλέξει μεταξύ των 3 βασικών κατηγοριών αναφορών (Συνολικές – ανά Εργοστάσιο – ανά Χρήστη)

5.14 Συγκεντρωτικά στοιχεία Συνεταιρισμού

Στη σελίδα συνολικών αναφορών (?action=total_reports) υπολογίζονται τα σύνολα του συνεταιρισμού όσο αφορά όλο το ελαιόλαδο, το βιολογικό, το μη βιολογικό.
Για κάθε ένα από αυτά εμφανίζονται ο όγκος της ελιάς, ο όγκος του λαδιού, η μέση οξύτητα, και το σύνολο των εισπράξεων.

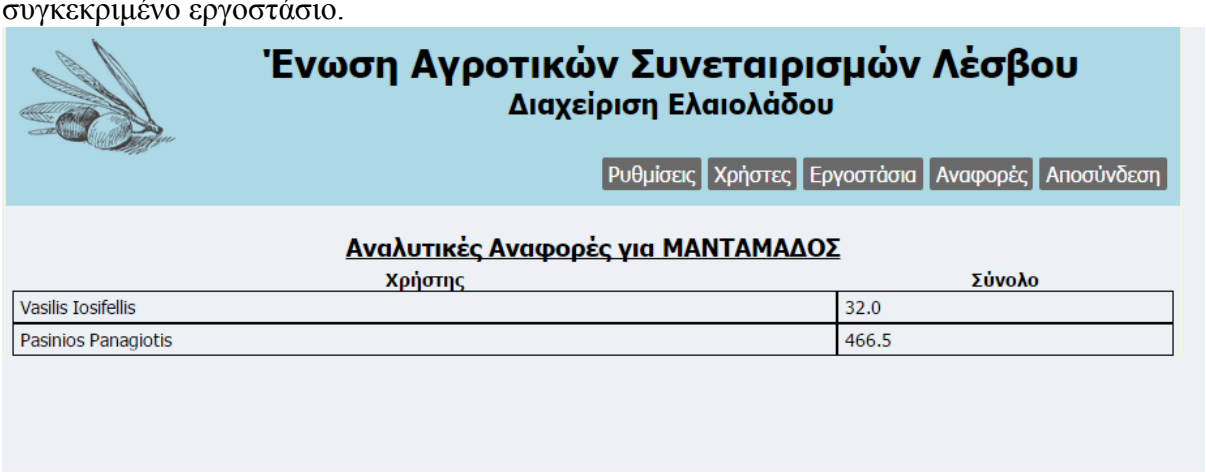
5.15 Συγκεντρωτικές αναφορές ανά εργοστάσιο

Στις αναφορές ανά εργοστάσιο (?action=factory_reports), βλέπουμε μία λίστα με τα υπάρχοντα εργοστάσια, μαζί με συνολικά ποσά που αφορούν τον όγκο ελιάς, τον όγκο λαδιού, τη μέση οξύτητα και το σύνολο εισπράξεων για κάθε ένα εργοστάσιο. Υπάρχει και η επιλογή για κάθε εργοστάσιο για αναλυτική παρουσίαση των εισπράξεων ανά χρήστη.



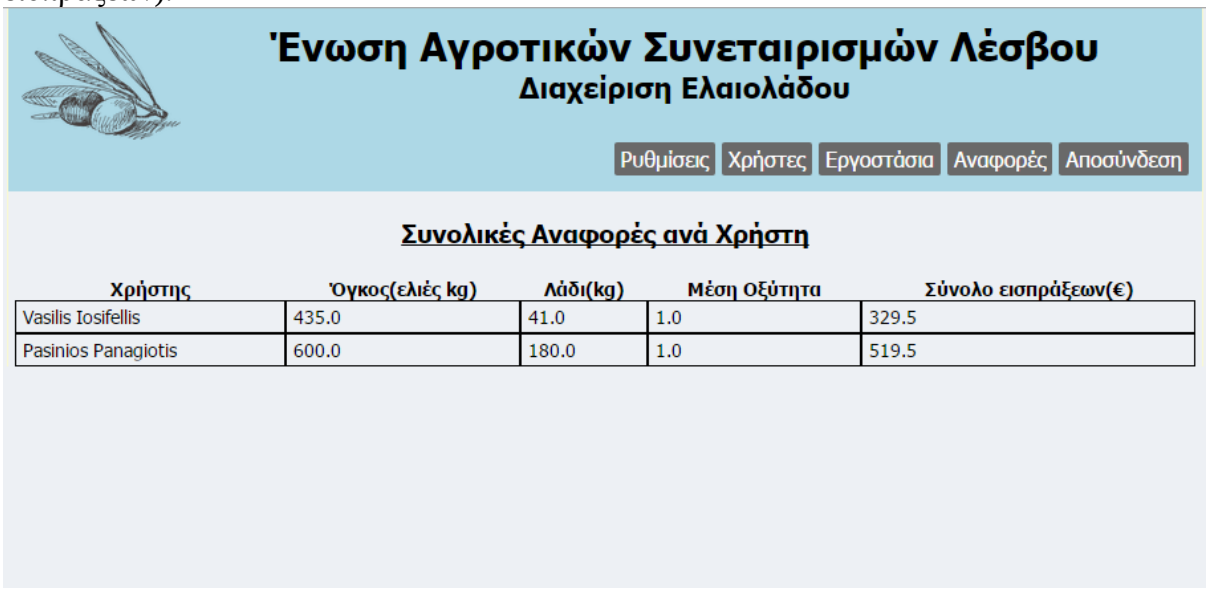
5.16 Αναλυτικές εισπράξεις εργοστασίου ανά χρήστη

Επιλέγοντας αναλυτική παρουσίαση των εισπράξεων ανά χρήστη (?action=factory_reports_per_user) βλέπουμε τις εισπράξεις κάθε χρήστη από το συγκεκριμένο εργοστάσιο.



5.17 Συγκεντρωτικές αναφορές ανά χρήστη

Στις αναφορές ανά χρήστη (?action=user_reports), βλέπουμε αναλυτικά τα σύνολα των παραγωγών για κάθε χρήστη του συστήματος (όγκος ελιάς, λάδι, μέση οξύτητα, σύνολο εισπράξεων).



6. Παράρτημα κώδικα

Παρακάτω υπάρχουν δείγματα κώδικα όπως χρησιμοποιήθηκαν κατά την υλοποίηση της εφαρμογής.

6.1. Java

Κλάση Factory

```
package mainapp;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashMap;

public class Factory {
    private int ID;
    private String name;
    private String address;
    private String owner;
    private String phone;

    public Factory(int ID, String name, String address, String owner, String phone) {
        this.ID = ID;
        this.name = name;
        this.address = address;
        this.owner = owner;
        this.phone = phone;
    }

    public Factory(int ID) throws ClassNotFoundException, SQLException {
        DB db = DB.getInstance();
        ResultSet rs = DB.selectRows("factory", "*", "`id` = "+ID, null, null);
        ArrayList res = new ArrayList<>();
        while(rs.next()){
            this.ID = ID;
            this.address = rs.getString("address");
            this.phone = rs.getString("phone");
            this.name = rs.getString("name");
            this.owner = rs.getString("owner");
        }
    }

    public ArrayList<Production> getAllProductions() throws ClassNotFoundException, SQLException {
        ResultSet rs = DB.selectRows("production", "*", "`factory_id` = "+this.getID(), null, null);
        ArrayList res = new ArrayList<>();
        while(rs.next()){
            res.add(new Production(rs.getInt("id"), rs.getInt("producer_id"),
                    rs.getInt("factory_id"), rs.getString("timestamp"),
                    rs.getFloat("olives_weight"), rs.getFloat("oil_weight"),
                    rs.getFloat("acidity"), (rs.getInt("is_bio") == 1), rs.getFloat("costs"),
                    (rs.getInt("sold") == 1), rs.getFloat("sold_price")));
        }
        return res;
    }
```

```
public HashMap getAllProductionsByUser() throws SQLException, ClassNotFoundException{
      String sql = "SELECT `producer_id`, SUM(`sold_price`) as `total` "+
            "FROM `production` WHERE `factory_id` = "+this.getID()+" AND `sold` = 1 GROUP BY
`producer_id`";
      ResultSet rs = DB.execSelectSql(sql);
      HashMap map = new HashMap();
      while(rs.next()){
         map.put(rs.getInt("producer_id"), rs.getFloat("total"));
      }
      return map;
   }

   public static ArrayList<Factory> getAllFactories() throws SQLException, ClassNotFoundException{
      DB db = DB.getInstance();
      ResultSet rs = DB.selectRows("factory", "*", null, "`name` ASC", null);
      ArrayList res = new ArrayList<>();
      while(rs.next()){
         res.add(new Factory(rs.getInt("id"), rs.getString("name"), rs.getString("address"),
               rs.getString("owner"), rs.getString("phone")));
      }
      return res;
   }
   public static void delete(int ID) throws ClassNotFoundException, SQLException {
      DB db = DB.getInstance();
      DB.deleteRow("factory", "`id` = "+ID);
   }
   public int getID() {return ID;}
   public void setID(int ID) {this.ID = ID;}
   public String getAddress() {return address;}
   public void setAddress(String address) {this.address = address;}
   public String getOwner() {return owner;}
   public void setOwner(String owner) {this.owner = owner;}
   public String getPhone() {return phone;}
   public void setPhone(String phone) {this.phone = phone;}
   public String getName() {return name;}
   public void setName(String name) {this.name = name;}
}
```

Κλάση PricingOptions

```
package mainapp;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

public class PricingOptions {
   private float normalPrice;
   private float bioPrice;
   private float acidityStep;

   public PricingOptions () throws ClassNotFoundException, SQLException {
      DB db = DB.getInstance();
      ResultSet rs = DB.selectRows("pricing", "*", null, null, null);
      this.normalPrice = 0;
      this.bioPrice = 0;
      this.acidityStep = 0;
      while(rs.next()){
         if (rs.getString("type").equals("normal_price")) {
```

```
            this.normalPrice = rs.getFloat("value");
        }
        else if (rs.getString("type").equals("bio_price")) {
            this.bioPrice = rs.getFloat("value");
        }
        else if (rs.getString("type").equals("acidity_step")) {
            this.acidityStep = rs.getFloat("value");
        }
    }
  }

  public PricingOptions (boolean loadPrices) throws ClassNotFoundException, SQLException {
    if (loadPrices == true) {
        DB db = DB.getInstance();
        ResultSet rs = DB.selectRows("pricing", "*", null, null, null);
        this.normalPrice = 0;
        this.bioPrice = 0;
        this.acidityStep = 0;
        while(rs.next()){
            if (rs.getString("type").equals("normal_price")) {
                this.normalPrice = rs.getFloat("value");
            }
            else if (rs.getString("type").equals("bio_price")) {
                this.bioPrice = rs.getFloat("value");
            }
            else if (rs.getString("type").equals("acidity_step")) {
                this.acidityStep = rs.getFloat("value");
            }
        }
    }
    else {
        this.normalPrice = 0;
        this.bioPrice = 0;
        this.acidityStep = 0;
    }
  }

  public boolean save() {
    try {
        DB.execUpdateSql("UPDATE `pricing` SET `value` = "+this.normalPrice+" WHERE `type` =
'normal_price'");
        DB.execUpdateSql("UPDATE `pricing` SET `value` = "+this.bioPrice+" WHERE `type` = 'bio_price'");
        DB.execUpdateSql("UPDATE `pricing` SET `value` = "+this.acidityStep+" WHERE `type` =
'acidity_step'");
    } catch (ClassNotFoundException | SQLException ex) {
        return false;
    }
    return true;
  }
  public float getNormalPrice() {return normalPrice;}
  public void setNormalPrice(float normalPrice) {this.normalPrice = normalPrice;}
  public float getBioPrice() {return bioPrice;}
  public void setBioPrice(float bioPrice) {this.bioPrice = bioPrice;}
  public float getAcidityStep() {return acidityStep;}
  public void setAcidityStep(float acidityStep) {this.acidityStep = acidityStep;}
}
```

Servlet LoginAction

```
package mainapp;

import java.io.IOException;
import java.io.PrintWriter;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class LoginAction extends HttpServlet {

   /**
    * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
    * methods.
    *
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    * @throws java.lang.ClassNotFoundException
    * @throws java.sql.SQLException
    * @throws java.security.NoSuchAlgorithmException
    */
   protected void processRequest(HttpServletRequest request, HttpServletResponse response)
           throws ServletException, IOException, ClassNotFoundException, SQLException,
NoSuchAlgorithmException {
       //response.setContentType("text/html;charset=UTF-8");
       boolean error = false;
       String username = request.getParameter("username");
       String password = request.getParameter("password");
       MessageDigest md;
       md = MessageDigest.getInstance("MD5");
       md.update(password.getBytes());
       //byte[] passBytes = password.getBytes("UTF-8");
       byte[] digest = md.digest();
       StringBuffer sb = new StringBuffer();
       for (byte b : digest) {
           sb.append(String.format("%02x", b & 0xff));
       }
       String md5pass = sb.toString();
       ResultSet rs = DB.selectRows("user", "*", "`username` = '"+username+"' AND `password` =
'"+md5pass+"'", null, null);
       if (!rs.last()) {
          error = true;
       }
       else {
          rs.beforeFirst();
          rs.next();
          HttpSession session = request.getSession();
          session.setAttribute("username", rs.getString("username"));
          session.setAttribute("userId", rs.getString("id"));
```

```
            session.setAttribute("isAdmin", rs.getInt("is_admin"));
            session.setMaxInactiveInterval(60*60); // 60 mins
        }
        if (error) {
            response.sendRedirect("?action=login&error=1");
        }
        else {
            response.sendRedirect("?action=main");
        }
    }
    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(LoginAction.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SQLException ex) {
            Logger.getLogger(LoginAction.class.getName()).log(Level.SEVERE, null, ex);
        } catch (NoSuchAlgorithmException ex) {
            Logger.getLogger(LoginAction.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(LoginAction.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SQLException ex) {
            Logger.getLogger(LoginAction.class.getName()).log(Level.SEVERE, null, ex);
        } catch (NoSuchAlgorithmException ex) {
            Logger.getLogger(LoginAction.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public String getServletInfo() {
        return "";
    }
```

6.2 JSP

index.jsp

```
<%
   String action = "main.jsp";
   if ( (request.getParameter("action") != null)) {
      action = request.getParameter("action").toLowerCase()+".jsp";
   }
   if (!action.toLowerCase().equals("login.jsp") &&
!action.toLowerCase().equals("register.jsp")) {
      if (session.getAttribute("username") == null)
         response.sendRedirect("?action=login");
   }
   else{
      if (session.getAttribute("username") != null)
         response.sendRedirect("?action=main");
   }
   if (session.getAttribute("isAdmin") != null && (Integer)session.getAttribute("isAdmin")
== 1) {
      if(action.toLowerCase().equals("main.jsp"))
         response.sendRedirect("?action=users");
   }
%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
   <head>
      <title>Ένωση Αγροτικών Συνεταιρισμών Λέσβου</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <link rel="stylesheet" href="style.css" type="text/css" />
      <link rel="stylesheet" href="font-awesome.css" type="text/css" />
   </head>
   <body>
      <div class="container">
         <%@include file="header.jsp" %>
         <div class="content">
            <jsp:include page="<%=action%>" flush="true" />
         </div>
      </div>
   </body>
</html>
```

login.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<center>
   <form method="POST" action="${pageContext.request.contextPath}/LoginAction">
      <h6 class="form_error">
         <%
            if (request.getParameter("error") != null) {
               out.print("Παρακαλώ προσπαθήστε ξανά");
            }
         %>
      </h6>
      <h6 class="success">
         <%
            if (request.getParameter("reg") != null) {
               out.print("Επιτυχής εγγραφή. Μπορείτε να συνδεθείτε.");
            }
         %>
      </h6>
      <table>
         <tr>
            <td><label for="username">Όνομα Χρήστη</label></td>
            <td>
               <input type="textfield" name="username" id="username" autocomplete="off" />
            </td>
         </tr>
         <tr>
            <td><label for="password">Κωδικός</label></td>
            <td>
               <input  type="password"  name="password"  id="password"  autocomplete="off"
/>
            </td>
         </tr>
         <tr>
            <td></td>
            <td>
               <input type="submit" name="submit" value="Σύνδεση" />
               <a href="?action=register" style="font-size: 12px">ή εγγραφή</a>
            </td>
         </tr>
      </table>
   </form>
</center>
```

6.3 CSS

Δείγμα CSS που χρησιμοποιήθηκε

```
.header {
    background: lightblue;
    text-align: center;
    padding: 10px
}
.container {
    width:100%;
    max-width: 900px;
    margin: 0 auto;
    border-left: 1px solid lightgoldenrodyellow;
    border-right: 1px solid lightgoldenrodyellow;
}

.content {
        padding: 20px 5px 0 5px;
}

.form_error {
    font-size: 12px;
    color: red;
}

.success {
    color: green;
}

form table{
    margin: auto;
}
form td:first-child {
    text-align: right;
}

form td {
    padding: 5px;
}

.menu {
}

.menu-items {
    list-style: none;
    padding: 0;
    margin: 0;
    text-align: right;
}

.menu-item {
    display: inline-block;
    padding: 5px 0;
}

.menu-item a {
    padding: 3px 5px;
```

```
    border-radius: 2px;
    background-color: dimgrey;
    color: white;
    text-decoration: none;
}

.text-center {
    text-align: center;
}

table.info-table {
    font-size: 14px;
    border-spacing: 0px;
    border-collapse: separate;
    width: 100%;
}

table.info-table td {
    border: 1px solid black;
    padding: 3px 5px;
}

table.info-table {
    font-size: 14px;
}

.link-button {
    font-size: 14px;
    color: white;
    padding: 5px;
    text-decoration: none;
    font-weight: bold;
    background-color: darkcyan;
    float: right;
    margin-bottom: 10px;
}
.actions a {
    margin-right: 4px;
    font-size: 16px
}

.box {
    background-color: cadetblue;
    width: 250px;
    height: 100px;
    margin: 10px;
    padding: 10px;
    display: inline-block;
    text-align: center;
    font-size: 21px;
    color: #333333;
}

.box h1{
    padding: 5px;
    margin-bottom: 10px
}
.box a {
    color: #333333;
}
```

```
.red {
    color: red;
}

.green {
    color: green;
}

.blue {
    color: darkblue;
}

.orange {
    color: burlywood;
}

.calc_button {
    text-decoration: none;
    padding: 1px 5px;
    font-size: 12px;
    border-radius: 3px;
    color: #fff;
    background-color: #337ab7;
    border-color: #2e6da4;
    text-align: center;
    border: 1px solid #334;
    border-radius: 4px;
}
```

7. Βιβλιογραφία – Πηγές

- Core Servlets and JavaServer Pages: Core Technologies (Hall Marty, Brown Larry)
- Java API – Oracle Documentation
- MITLibraries (http://libguides.mit.edu)
- W3 Schools. On-line Tutorials and Examples
- Stackoverflow.com