Πανεπιστήμιο Πελοποννήσου

Σχολή Θετικών Επιστημών και Τεχνολογίας

Τμήμα Επιστήμης και Τεχνολογίας Υπολογιστών

Μεταπτυχιακή Διπλωματική Εργασία

Σχεδιασμός και Ανάπτυξη Διαδραστικού, Διαδικτυακού Σχεδιαστικού Εργαλείου

Χατζηευστρατίου Γεώργιος

Επιβλέπων:

Λέπουρας Γεώργιος

Τρίπολη, Ιανουάριος 2013

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

Abstract:

World Wide Web (www) has achieved an important role in communication, information sharing and service delivery now-a-days. World Wide Web consists of millions of web sites and web based applications which are deployed and can be visited all over the world without the limitation of time and geographical boundaries. We discuss web drawing tool for creation of Logic gates ,UML diagrams simple draws. We discuss technical aspects like Mvc, Design Patterns, Demeter Law, Frameworks Persistence , ORM and Databases collecting the prerequisites for creating a web application and developing one. The application is made with Jsf Framework, Glassfish Server, PostgresSQL JavaScript and JPA. We use JPA as ORM tool for the mapping with the database tables and creating and storing Users. JavaScript is used for the stability and reabillty that has in web applications. Is a powerful programming language made for use in WEB applications. The combination of JavaScript makes the application response quick.

The Java Persistence API, sometimes referred to as JPA, is a Java programming language framework managing relational data in applications using Java Platform, Standard Edition and Java Platform, Enterprise Edition. PostgresSQL is a the most advanced open source object-relational database server . It has a strong reputation for reliability, data integrity, and correctness. It runs on all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows. JavaServer Faces (JSF) is a Java specification for building component-based user interfaces for web applications. It was formalized as a standard through the Java Community Process and is part of the Java Platform, Enterprise Edition. GlassFish is an open-source application server project started by Sun Microsystems for the Java EE platform and now sponsored by Oracle Corporation.

Κεφάλαιο 1.Εισαγωγή

Στόχος της Εργασίας

Στόχος της εργασίας είναι ο σχεδιασμός και η ανάπτυξη ενός διαδραστικού, διαδικτυακού σχεδιαστικού εργαλείου. Ένα τέτοιο εργαλείο θα επιτρέπει στο χρήστη να υλοποιήσει σχέδια διαφορετικών σκοπών (π.χ. διαγράμματα ροής, λογικά κυκλώματα, κατόψεις, κ.λπ.) χωρίς να είναι υποχρεωμένος να εγκαταστήσει τοπικά στον υπολογιστή του κάποιο εξειδικευμένο πρόγραμμα αλλά μόνο με τη χρήση ενός προγράμματος πλοήγησης και με την προϋπόθεση της σύνδεσης στο Internet. Στα πλαίσια της εργασίας θα εκτελεστούν τα ακόλουθα:

- Καταγραφή αντίστοιχων σχεδιαστικών συστημάτων
- Ανάλυση απαιτήσεων χρηστών
- Σχεδιασμός αρχιτεκτονικής συστήματος
- Σχεδιασμός οθονών συστήματος

Για τις ανάγκες υλοποίησης του συστήματος θα γίνει χρήση αντικειμενοστραφούς μεθοδολογίας ανάπτυξης λογισμικού και θα χρησιμοποιηθούν UML διαγράμματα ως εργαλεία μοντελοποίησης τις εφαρμογής. Παράλληλα θα μελετηθούν οι δυνατότητες των τεχνολογιών J2EE.

Το σύστημα που καλούμαστε να υλοποιήσουμε είναι ένα διαδικτυακό σχεδιαστικό εργαλείο που θα αναπτυχθεί στα πλαίσια μιας διπλωματικής. Στη γενική περίπτωση το σχεδιαστικό εργαλείο αποτελεί μια σύνθετη εφαρμογή αφού οφείλει να λάβει υπόψη μεγάλο αριθμό λειτουργιών και δυνατοτήτων που θα πρέπει να καλύπτει ένα ολοκληρωμένο εργαλείο.

Λόγω του περιορισμένου χρόνου ανάπτυξης παρουσιάζουμε μια απλοποιημένη μορφή διαδικτυακού σχεδιαστικού εργαλείου με περιορισμένο αριθμό λειτουργικών απαιτήσεων. Η παρούσα εργασία μπορεί όμως να είναι μια βάση για περαιτέρω εξέλιξη της εφαρμογής και διαφόρων πρόσθετων στοιχείων (plug ins) που θα καλυτερεύσουν την επίδοση της εφαρμογής.

Την συγκεκριμένη εφαρμογή θα μπορεί να δει και να χρησιμοποιήσει οποιοδήποτε χρήστης έχει σύνδεση στο διαδίκτυο. Παράλληλα θα υπάρχει η δυνατότητα κάποιος να κάνει εγγραφή στο σύστημα βάζοντας τα στοιχεία του και δημιουργώντας λογαριασμό που θα του δίνει την δυνατότητα να διαχειρίζεται τον λογαριασμό του, να δημιουργεί σχέδια, να τα αποθηκεύει και να τα εκτυπώνει.

Φυσικά θα πρέπει να υπάρχει ένας διαχειριστής για την τήρηση τις καλής λειτουργίας της εφαρμογής την υποστήριξη σε τυχόν προβλήματα που θα έχει ο χρήστης και την παραμετροποίηση των δυνατοτήτων του συστήματος.

Κεφάλαιο 2.Παρουσίαση αντίστοιχων εργαλείων

Στη συνέχεια θα παραθέτουμε μια σύντομη παρουσίαση των χαρακτηριστικών και των λειτουργιών υφιστάμενων διαδικτυακών σχεδιαστικών εργαλείων.

http://yuml.me/

Είναι ένα διαδικτυακό εργαλείο δημιουργίας διαγραμμάτων UML, που μας δίνει τη δυνατότητα να σχεδιάσουμε διαγράμματα UML με γραπτό τρόπο.
Ενώ στα περισσότερα σχεδιαστικά εργαλεία ο χρήστης με την βοήθεια του ποντικιού μπορεί να σχεδιάσει κάποιο διάγραμμα άμεσα στην οθόνη, το σχέδιο στο yuml έχουμε τη δυνατότητα να το κάνουμε γραπτά με γλώσσα παραγωγής διαγραμμάτων.

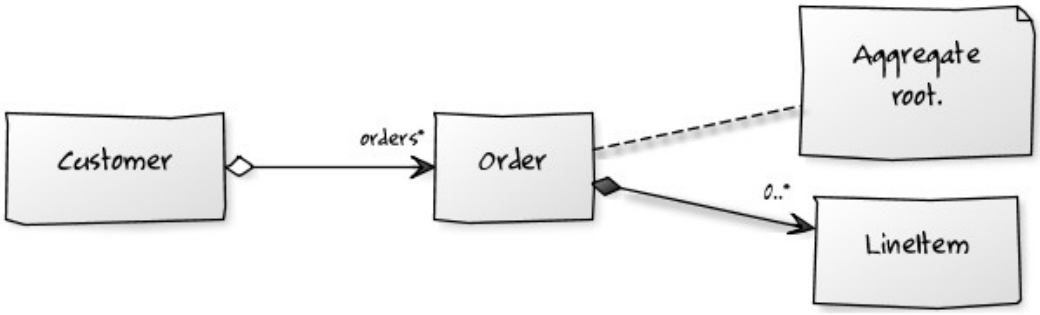
Έχει υλοποιηθεί σε Ruby on Rails και η χρήση του απαιτεί εγγραφή και πληρωμή.
Αρνητικό στοιχείο θεωρείται η απαίτηση μάθησης της γλώσσας για την δημιουργία σχεδίου μιας και άλλα εργαλεία δεν θέτουν τέτοια προϋπόθεση. Επίσης το yuml μπορεί να δημιουργήσει μόνο διαγράμματα uml και δεν έχει τη δυνατότητα υποστήριξης άλλων τύπων διαγραμμάτων. Το yuml δεν ενδείκνυται για παραγωγή μεγάλων και απαιτητικών σχεδίων. Από το site προτείνεται η έκδοση του UMLet για desktop χρήση καθώς είναι πιο σταθερή και αξιόπιστη.

Για την εκμάθηση της γλώσσας υπάρχουν αρκετά παραδείγματα που κάποιος μπορεί με αντιγραφή και επικόλληση να δημιουργήσει το σχέδιο που θέλει. Στα θετικά η ύπαρξη φόρουμ όπου μπορεί κάποιος να θέσει ερωτήσεις ως προς την χρήση των εντολών τις γλώσσας και να ζητήσει την βοήθεια άλλων μελών.

Παράδειγμα γλώσσας δημιουργίας διαγράμματος κλάσης

```
// Cool Class Diagram
[Customer]<>-orders*>[Order]
[Order]++-0..*>[LineItem]
[Order]-[note:Aggregate root.]
```

Και σαν αποτέλεσμα έχουμε το εξής UML διάγραμμα κλάσεων



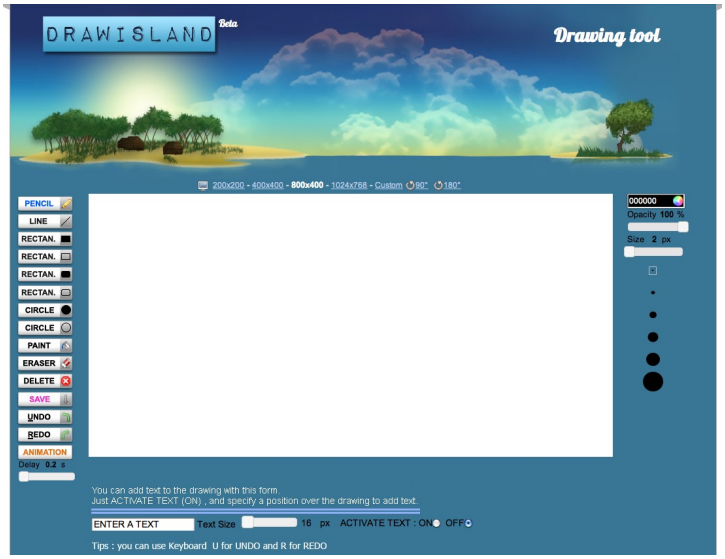
http://www.gliffy.com/

Πολύ εύχρηστο διαδικτυακό σχεδιαστικό εργαλείο που δίνει την δυνατότητα σχεδίασης ποιοτικών διαγραμμάτων επαγγελματικής απόδοσης. Το gliffy λειτουργεί κατευθείαν στο πρόγραμμα πλοήγησης χωρίς καμία άλλη προετοιμασία.

Με δεδομένο ότι σχεδιάζουμε άμεσα στην οθόνη έχει την δυνατότητα να κάνει την εργασία πιο παραγωγική και πιο γρήγορη. Επίσης έχει την δυνατότητα να δημιουργήσει πολλά διαγράμματα, μοντέλα όπως διαγράμματα venn, δίκτυα υπολογιστών, UML διαγράμματα κτλ, ενώ υπάρχει και έκδοση για δημιουργία Google Apps. Μειονέκτημα του είναι η χρήση επί πληρωμή.

http://drawisland.com/

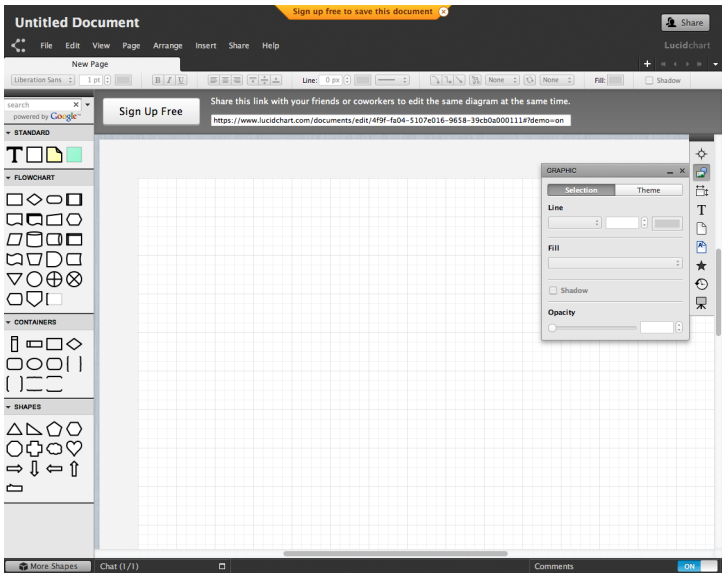
Αρκετά απλό σχεδιαστικό εργαλείο, εύκολο στη χρήση με περιορισμένες όμως δυνατότητες.



Στα θετικά η δυνατότητα δημιουργίας κινούμενων εικόνων, στα αρνητικά η δυνατότητα σχεδίασης μόνο απλών σχημάτων και η απουσία δυνατότητας επέκτασης.

https://www.lucidchart.com

Διαδικτυακό σχεδιαστικό εργαλείο με πολλές δυνατότητες επέκτασης, εργαλειοθήκες για διαφορετικά σχήματα και δυνατότητα επέκτασης. Μειονέκτημα του η απαίτηση μηνιαίας συνδρομής.



Κεφάλαιο 3.Ανάλυση απαιτήσεων

Γενικές Απαιτήσεις

Στόχος της παρούσας διπλωματικής είναι ο σχεδιασμός και η υλοποίηση ενός διαδικτυακού εργαλείου το οποίο θα επιτρέπει την εύκολη και γρήγορη δημιουργία ενός πλήθους διαφορετικών διαγραμμάτων και σχημάτων (π.χ. διαγράμματα ροής, ηλεκτρικά κυκλώματα, λογικά κυκλώματα, κλπ) με τη χρήση έτοιμων βασικών σχημάτων. Η ονομασία της εφαρμογής θα είναι gDesign.

Απαιτήσεις Χρηστικότητας

Η διαδικτυακή εφαρμογή gDesign θα είναι προσπελάσιμη από σύνολο διαφορετικών ομάδων χρηστών και ρόλων και επομένως κρίνεται σημαντική η παροχή ομοιόμορφων, χρηστικών και ευέλικτων διεπαφών χρήστη ώστε να καλύπτεται το μεγαλύτερο δυνατό εύρος χρηστών ανάλογα και με το βαθμό εξοικείωσης σε διαδικτυακές εφαρμογές. Οι λειτουργίες τις εφαρμογής σε ότι αφορούν θέματα χρηστικότητας παρατίθενται παρακάτω:

- Η εφαρμογή, όπου αυτό είναι δυνατό, θα πρέπει να παρέχει ένα ομοιογενές περιβάλλον εργασίας με τη χρήση κεντρικής οθόνης από την οποία ο χρήστης θα μπορεί να μεταβεί σε άλλες οθόνες της εφαρμογής με τη χρήση μενού επιλογών, εργαλειοθηκών (toolbars), κουμπιών πλοήγησης ή πλήκτρων συντομεύσεων (shortcuts).
- Όπου είναι δυνατό θα πρέπει να παρέχονται γραφικές ή δικτυακές διεπαφές για την προεπισκόπηση των δεδομένων των εφαρμογών.
- Όλες οι εφαρμογές θα πρέπει να παρέχουν εύχρηστα και ευδιάκριτα μενού πλοήγησης (navigation menus), μενού pop-up, εργαλειοθήκες (toolbars), κουμπιά, εικονίδια, συνδέσμους και πλήκτρα συντόμευσης (shortcuts) ώστε να είναι εύκολα αντιληπτά από το χρήστη. Επίσης, είναι επιθυμητή η ενιαία τυποποίηση όλων των παραπάνω για το σύνολο των εφαρμογών.
- Κάθε εφαρμογή θα πρέπει να παρέχει βοηθητικές πληροφορίες σε πραγματικό χρόνο (real time) κατά την εκτέλεση ενεργειών όπως καταχώρηση δεδομένων στο σύστημα, διαγραφή ή τροποποίηση στοιχείων μέσα από την προβολή βοηθητικών μηνυμάτων, την προεπιλογή δεδομένων (μέσα από τη χρήση combo boxes κ.α.) κατά την καταχώρηση, την παροχή pop-up μενού ή συντομεύσεων , όπου αυτό είναι δυνατό.

- Θα πρέπει να παρέχονται μηχανισμοί ελέγχου της εγκυρότητας και της ποιότητας των δεδομένων και ταυτόχρονης και άμεσης προειδοποίησης μέσω μηνυμάτων στην οθόνη εργασίας.

Απαιτήσεις για τη Διεπαφή Χρήστη

Διαχείριση Δεδομένων

Οι διαχειριζόμενες οντότητες περιλαμβάνουν τα σχήματα ομαδοποιημένα σε κατηγορίες ανάλογα με τον τύπο διαγράμματος που σχεδιάζει ο χρήστης (π.χ. διάγραμμα ροής: ορθογώνιο, ρόμβος, κλπ).
Η διαχείριση των δεδομένων που σχετίζονται με τις διαχειριζόμενες οντότητες αναφέρεται στην εισαγωγή νέας «εγγραφής» στη Βάση Δεδομένων, στη διαγραφή και στη μεταβολή υφιστάμενης εγγραφής.

- Η εισαγωγή, διαγραφή και μεταβολή δεδομένων που αντιστοιχούν σε διακριτές διαχειριζόμενες οντότητες, πραγματοποιούνται μέσω κοινής διεπαφής χρήσης, η οποία παρέχει και τις τρεις σχετικές λειτουργίες.
- Οι διεπαφές χρήσης διαχείρισης οντοτήτων, στις περιπτώσεις που απαιτείται για λόγους ταχύτητας καταχώρησης στοιχείων και σε περιπτώσεις μικρής κρισιμότητας, υποστηρίζουν τη μαζική εισαγωγή, διαγραφή και τροποποίηση δεδομένων χωρίς την υποχρεωτική «συναλλαγή» με το επόμενο επίπεδο για κάθε μία μεταβολή. Οι μεταβολές πραγματοποιούνται συνολικά βελτιώνοντας έτσι την απόκριση της σχετικής διεπαφής χρήσης.

Αναζήτηση

Η αναζήτηση οντοτήτων, αποτελεί σημαντικό στοιχείο όλων των εφαρμογών που πραγματεύονται τη διαχείρισή τους και την υλοποίηση σχετικής λειτουργικότητας. Ως λειτουργία αποτελεί προϋπόθεση για την υλοποίηση των λειτουργιών συντήρησης των αντίστοιχων δεδομένων. Αποτελεί εν γένει την πλέον συχνά χρησιμοποιούμενη λειτουργία, επηρεάζοντας σημαντικά την αποτελεσματικότητα των αντίστοιχων διεπαφών χρήσης.

Η αναζήτηση των οντοτήτων πρέπει να βασίζεται στη χρήση πολλαπλών και παραμετρικών κριτηρίων, δίνοντας τη δυνατότητα στο χρήστη να εντοπίζει την επιθυμητή πληροφορία με ελάχιστα βήματα (στις γενικές περιπτώσεις σε ένα βήμα). Πιο συγκεκριμένα:

- Τα κριτήρια αναζήτησης αντιστοιχούν σε περιορισμό των τιμών που μπορεί να περιλαμβάνει συγκεκριμένο χαρακτηριστικό της αναζητούμενης οντότητας, δίνοντας

τη δυνατότητα στο χρήστη να προσδιορίζει συγκεκριμένους κανόνες οι οποίοι θα πρέπει κατ' ελάχιστο να περιλαμβάνουν:

- Ισότητα με συγκεκριμένη τιμή, προσδιοριζόμενη από το χρήστη,
- Ισότητα με συγκεκριμένη τιμή, επιλεγόμενη από σύνολο επιτρεπόμενων τιμών,
- Συμφωνία τιμής με μάσκα που περιλαμβάνει χαρακτήρες «μπαλαντέρ» για συμβολοσειρές.

- Ως προσδιοριζόμενα κριτήρια αναζήτησης καθορίζονται χαρακτηριστικά της αναζητούμενης οντότητας, όπου μπορούν να χρησιμοποιούνται για τον προσδιορισμό των επιθυμητών τιμών σε οποιοδήποτε συνδυασμό μεταξύ τους κατ' ελάχιστο μέσω λογικής σύζευξης.

Υποθέσεις - Παραδοχές

Οι παρακάτω παραδοχές έγιναν με σκοπό να παρουσιαστεί μια απλοποιημένη έκδοση του συστήματος που να ικανοποιεί με καλύτερο τρόπο τις ανάγκες τις μελέτης .

- Ο κάθε χρήστης θα πρέπει να μπορεί να δει τι κάνει η εφαρμογή χωρίς να είναι εγγεγραμμένος ή να έχει κάποια ειδικά δικαιώματα .
- Ο κάθε χρήστης θα μπορεί να εγγραφεί και μετά από ορθή εισαγωγή των στοιχείων του μέσω φόρμας θα μπορεί να έχει την δυνατότητα σχεδίασης
- Θα υπάρχουν 3 ειδών χρήστες: Users, Admin, Viewers.
- Ανάλογα με το επίπεδο διαβάθμισης τους θα έχουν και τα αντίστοιχα δικαιώματα.
- Ο User θα μπορεί να κάνει κάποιο σχέδιο εφόσον είναι συνδεδεμένος.
- Ο Admin θα μπορεί να διαχειριστεί την εφαρμογή στο backend της.
- Ο Viewer θα μπορεί να δει το διαδικτυακό εργαλείο χωρίς να μπορεί να σχεδιάσει κάτι ή να χρησιμοποιήσει κάποια δυνατότητα του εργαλείου.
- Ο Admin θα μπορεί να ελέγχει τα στοιχεία των χρηστών αλλά όχι τα προσωπικά δεδομένα.
- Ο User θα έχει την δυνατότητα να αλλάξει τα στοιχεία του και να δηλώσει τηλέφωνο διεύθυνση και λοιπά προσωπικά στοιχεία.
- Ο Admin θα πρέπει να μπορεί να έχει οθόνη που θα ελέγχει την ορθότητα τις εφαρμογής και θα μπορεί μέσω της οθόνης αυτής να βγάζει αναφορές για τη λειτουργία και την τρέχουσα κατάσταση τις εφαρμογής καθώς και για την χρήση που γίνεται μαζί με άλλα στοιχεία που θα μας ζητούνται.

- Η εφαρμογή θα πρέπει να έχει τις εξής σελίδες Index/AboutUs/Login Register /Contact θα υπάρχει μια Login Register για την εγγραφή του χρήστη

- Κάθε Registered User μπορεί να κάνει χρήση του εργαλείου και να το επεκτείνει με διάφορα plug-ins που θα είναι διαθέσιμα για αγορά έτσι ώστε να έχει περισσότερες δυνατότητες για σχεδίαση.
- Το σύστημα προτείνει εναλλακτικά plug ins για αγορά από τον χρήστη. Δεν υπάρχει μέγιστος ενδεικτικός αριθμός προτεινόμενων plug ins.
- Δεν υπάρχει ένας ενδεικτικός χρονικός περιορισμός για να δημιουργηθεί ένα σχέδιο.
- Δεν υπάρχει σύνδεση των σχεδίων με κάποιο άλλο εργαλείο αλλά ούτε και κάποια άλλη χρήση αυτού.
- Η ακύρωση του σχεδίου προκαλείται μόνο από τον χρήστη (νέα επιθυμία χωρίς νέες προτάσεις) και δεν υπάρχει αυτόματη διαδικασία διαγραφής ακύρωσης σχεδίου.
- Για την αποθήκευση του σχεδίου δεν ισχύουν χρονικοί περιορισμοί.
- Για την διαγραφή του σχεδίου διαγράμματος δεν ισχύουν χρονικοί περιορισμοί.
- Υπάρχει η δυνατότητα εκτύπωσης του σχεδίου σε εκτυπωτή.
- Το Δεξί κλικ θα είναι ενεργοποιημένο και ελεύθερο για χρήση.

Λειτουργικές προδιαγραφές

Καταχώριση νέων χρηστών

Περιγραφή: Η εφαρμογή εμφανίζει φόρμα, μέσω της οποίας ένας νέος χρήστης εισάγει τα στοιχεία του στο σύστημα.
Είσοδος: Στοιχεία χρήστη.
Επεξεργασία: Καταχώριση στοιχείων και δημιουργία κωδικού χρήστη.
Έξοδοι: Κωδικός χρήστη, ενημερωμένο αρχείο χρηστών ή μήνυμα λάθους σε περίπτωση προβληματικής εισόδου δεδομένων.

Εισαγωγή υπάρχοντος χρήστη στο σύστημα

Περιγραφή: Η εφαρμογή εμφανίζει φόρμα, μέσω της οποίας ένας νέος χρήστης εισάγει τον κωδικό του για να προσπελάσει την εφαρμογή.
Είσοδος: Κωδικός χρήστη.
Επεξεργασία: Έλεγχος ύπαρξης του κωδικού χρήστη και αναζήτηση των προσωπικών στοιχείων του.

Έξοδοι: Στοιχεία χρήστη ή μήνυμα λάθους σε περίπτωση αποτυχημένης εισαγωγής κωδικού μήνυμα λάθους σε περίπτωση λάθος εισαγωγής .

Δημιουργία Σχεδίασης

Περιγραφή: Η εφαρμογή εμφανίζει οθόνη με την δυνατότητα επιλογής σχεδίασης.
Είσοδος: Ο χρήστης μετακινώντας το mouse του μπορεί να δημιουργήσει κάποιο σχέδιο.
Επεξεργασία: Δημιουργία εγγραφής σχεδίου και καταχώρισή της στο σύστημα για κάθε επιλογή του χρήστη.
Έξοδοι: Στοιχεία διαθέσιμων σχεδίων στην οθόνη του χρήστη και ενημερωμένο αρχείο σχεδίων ή μήνυμα λάθους.

Αποθήκευση

Περιγραφή: Η εφαρμογή εμφανίζει στην οθόνη την δυνατότητα επιλογής αποθήκευσης τις εκάστοτε σχεδίασης.
Είσοδος: Στην μπάρα επιλογών υπάρχει εικονίδιο με το save από την εφαρμογή gDesign, πατώντας το εικονίδιο εμφανίζεται μήνυμα αποθήκευσης του σχεδίου ζητώντας επιβεβαίωση και αποθηκεύει το σχέδιο μας.
Επεξεργασία: Δημιουργία εγγραφής σχεδίου και καταχώρισή της στο σύστημα για κάθε επιλογή του χρήστη.
Έξοδοι: Το μήνυμα για την ορθή αποθήκευση του σχεδίου ή μήνυμα λάθους αποθήκευσης η μήνυμα λάθους συμβάντος.

Εκτύπωση

Περιγραφή: Η εφαρμογή εμφανίζει στην οθόνη την δυνατότητα επιλογής εκτύπωσης τις εκαστοτε σχεδίασης.
Είσοδος: Στην μπάρα επιλογών υπάρχει εικονίδιο με το εικονίδιο εκτύπωσης από την εφαρμογή gDesign ,πατώντας το εικονίδιο εμφανίζεται μήνυμα εκτύπωσης του σχεδίου ζητώντας επιβεβαίωση και τις δυνατές επιλογές εκτύπωσης καθώς και εκτυπωτή.
Επεξεργασία: Δημιουργία εγγραφής σχεδίου και καταχώρισή της στο σύστημα για κάθε επιλογή του χρήστη.
Έξοδοι: Το μήνυμα για την ορθή αποθήκευση του σχεδίου ή μήνυμα λάθους αποθήκευσης η μήνυμα λάθους συμβάντος.

Αλλαγή στοιχείων

Περιγραφή: Η εφαρμογή εμφανίζει στην οθόνη την δυνατότητα αλλαγής στοιχείων χρήστη .

Είσοδος: Στην μπάρα επιλογών υπάρχει link που μας πηγαίνει στην φόρμα αλλαγής στοιχείων χρήστη.

Επεξεργασία: Αλλαγή στοιχείων χρήστη και αποθήκευση .

Έξοδοι: Το μήνυμα για την ορθή εισαγωγή και αποθήκευση των στοιχείων
ή μήνυμα λάθους αποθήκευσης η μήνυμα λάθους συμβάντος.

Έξοδος χρήστη από την εφαρμογή

Περιγραφή: Η εφαρμογή εμφανίζει στην οθόνη την δυνατότητα επιλογής εξόδου από την εφαρμογή Log out.

Είσοδος: Στην μπάρα επιλογών υπάρχει εικονίδιο με το εικονίδιο εξόδου από την εφαρμογή gDesign ,πατώντας το εικονίδιο εμφανίζεται μήνυμα επιβεβαίωσης για την έξοδο και την τυχόν αποθήκευση του σχεδίου στην βάση.

Επεξεργασία: Αποθήκευση και καταχώρισή του σχεδίου στο σύστημα πριν την έξοδο από αυτό.

Έξοδοι: Το μήνυμα για την ορθή αποθήκευση του σχεδίου ή μήνυμα λάθους αποθήκευσης η μήνυμα λάθους συμβάντος καθώς και μήνυμα για την επιβεβαίωση τις εξόδου από το σύστημα.

Μη λειτουργικές προδιαγραφές

Απαιτήσεις χρήσης

Το λογισμικό θα πρέπει να περιέχει φιλικό περιβάλλον προς τον χρήστη με υποστήριξη γραφικών και συνδυασμό χρήσης ποντικιού και πληκτρολογίου. Θα υπάρχουν φόρμες καταχώρησης στοιχείων με αυτοματοποίηση πεδίων όπου αυτό επιτρέπεται (π.χ. ΝΑΙ/ΟΧΙ της αναβάθμισης θα επιλέγεται από τον χρήστη με αυτόματο τρόπο).

Απαιτήσεις αξιοπιστίας

Η πρόσβαση από 3 χρήστες ταυτόχρονα προστατεύεται με δυνατότητα να χρησιμοποιούν την εφαρμογή με δικό τους κωδικό χρήστη και password. Το λογισμικό θα υποστηρίζει «κλείδωμα» των εγγραφών κάθε χρήστη ώστε να μην υπάρχουν προβλήματα λάθους κατά τη διάρκεια ενημέρωσης από άλλο χρήστη.

Απαιτήσεις επιδόσεων

Θα υπάρχει αξιοποίηση των πόρων του συστήματος, όπως για παράδειγμα η χρήση μαγνητικών ταινιών για διαδικασίες αποθήκευσης αντιγράφων των δεδομένων (backup) θα πραγματοποιείται τις βραδινές ώρες(κατά την διάρκεια συντήρησης). Επιπλέον, η απόκριση από το σύστημα βάσεων δεδομένων δεν θα ξεπερνά τα 2 δευτερόλεπτα.

Απαιτήσεις υποστήριξης

Το λογισμικό θα διατίθεται online και θα είναι διαθέσιμο σε οποιοδήποτε χρήστη έχει πρόσβαση στο internet.

Απαιτήσεις σχεδίασης

Η σχεδίαση θα γίνει σύμφωνα με το πρότυπο carner και την UML.

Απαιτήσεις υλοποίησης

Θα χρησιμοποιηθεί η γλώσσα Java λόγω της δυνατότητας ανάπτυξης δια-πλατφορμικών εφαρμογών με αντικειμενοστραφή προγραμματισμό σε συνδυασμό με χρήση SQL για υποβολή ερωτημάτων προς τη Βάση Δεδομένων, JavaScript για την διαδραστικότητα στο Internet, HTML και css για την μορφοποίηση των σελίδων.

Απαιτήσεις επικοινωνίας με άλλα συστήματα

Το σύστημα θα περιλαμβάνει σύνδεση Βάσεων Δεδομένων με την εφαρμογή μέσω του πρωτοκόλλου JDBC ώστε να υπάρχει συγχρονισμός της λειτουργίας τους.

Απαιτήσεις Βάσεων Δεδομένων

Θα χρησιμοποιηθεί σύστημα Βάσεων Δεδομένων postgresSQL.

Φυσικές απαιτήσεις

Το λογισμικό αρχικά θα εγκατασταθεί σε έναν Glassfish server και θα έχει εγκατεστημένα επίσης την postgresSQL για την Βάση δεδομένων.

Το μοντέλο του πεδίου προβλήματος

Θα πρέπει να υπάρχει σχεδιασμός για βασικές μεθόδους που θα είναι προσβάσιμες που θα χρησιμοποιηθούν αργότερα σε άλλα κομμάτια τις εφαρμογής.

Κεφάλαιο 4.Σχεδίαση Εφαρμογής

Το Μοντέλο UML

Ως ένα αρχικό βήμα πριν την επεξεργασία του προβλήματος θεωρούμε την απαρίθμηση των απαιτήσεων όπως αυτές προκύπτουν από την γενική περιγραφή της μελέτης περίπτωσης.

Η απαρίθμηση των απαιτήσεων θα μας βοηθήσει στην ιχνηλασιμότητα (traceability) αυτών κατά την διάρκεια της ανάπτυξης του συστήματος. Γενικότερα η διαχείριση απαιτήσεων (requirements management) και το μοντέλο που προκύπτει αποτελεί βάση για την σχεδίαση τον έλεγχο και την τεκμηρίωση του συστήματος .

Η ιχνηλάτηση των απαιτήσεων επιτρέπει

- Την καλύτερη αξιολόγηση των αλλαγών στις απαιτήσεις η οποία θα ήταν απαραίτητη στην ανάπτυξη ενός πραγματικού συστήματος .
- Την αξιολόγηση των αποτελεσμάτων του έλεγχου (testing)
- Την επαλήθευση της ικανοποίησης των απαιτήσεων
- Τη διαχείριση των αλλαγών

Επιπλέον εκτός από τις λειτουργικές απαιτήσεις θα πρέπει να λάβουμε υπόψη μας τις μη λειτουργικές απαιτήσεις (non-functional requirements) του συστήματος

Πρωταρχικός μας στόχος στη δημιουργία του μοντέλου του πεδίου προβλήματος είναι να προσδιορίσουμε τις αφαιρέσεις του πραγματικού κόσμου που απαιτούνται για την κατασκευή του συστήματος που θέλουμε να κατασκευάσουμε. Θα πρέπει επίσης να προσδιορίσουμε τις σχέσεις μεταξύ τους. Τρεις είναι οι πιο σημαντικοί τύποι σχέσεων:

- η σχέση γενίκευσης/ειδίκευσης (generalization/specialization) με την οποία περιγράφουμε την κληρονομικότητα,
- η σχέση συναρμολόγησης (aggregation) με την οποία περιγράφουμε τη σύνθεση και τέλος
- οι απλές συσχετίσεις που περιγράφουν την ανάγκη ύπαρξης άλλων αντικειμένων με σκοπό την υλοποίηση μιας συμπεριφοράς.

Θα δοθεί έμφαση στην κατασκευή των κλάσεων και των ιδιοτήτων τους και όχι στην κατασκευή και εύρεση των μεθόδων .

Κάποιοι βασικοί κανόνες για την εύρεση των κλάσεων είναι :

1.Τα ουσιαστικά και οι ονομαστικές φράσεις γίνονται κλάσεις ή πεδία κλάσεων.

2. Τα ρήματα και οι ρηματικές φράσεις γίνονται μέθοδοι και συνδέσεις (associations).

3. Οι κτητικές φράσεις δείχνουν ότι το ουσιαστικό αναπαριστά μια ιδιότητα παρά μια κλάση.

Μετά την εφαρμογή των κανόνων θα φτάσουμε σε ένα αριθμό υποψήφιων κλάσεων.

Όνομα υποψήφιας κλάσης	**Σκεπτικό**
AppUsers	Για να γίνει χρήση της εφαρμογής θα πρέπει να υπάρχει κάποιος χρήστης που είναι εγγεγραμμένος καθώς και κάποιος που θα μπορεί να την συντηρήσει και να τη δει.
Wuser	Βοηθητική κλάση με τις ίδιες λειτουργίες με την AppUsers
RollbackFailureException	Κλάση για το RollBack όταν γίνεται το Transaction.
Draw	Η κλάση αυτή θα είναι χρήσιμη για την αποθήκευση του σχεδίου με βάση το σχέδιο που έχει γίνει .Θα έχει στοιχεία για το Σχέδιο που θα γίνει, όπως η ώρα, η μέρα, ο χρήστης. .(Μελλοντική Επέκταση)Στο παρόν έχει γίνει με JavaScript.
Diagram	Η κλάση αυτή θα είναι χρήσιμη για την αποθήκευση του διαγράμματος με βάση το διάγραμμα που έχει δημιουργηθεί . Θα έχει στοιχεία για το διάγραμμα που θα έγινε όπως η ώρα, η μέρα, ο χρήστης.(Μελλοντική Επέκταση)
Colors	Η κλάση αυτή θα μπορεί να έχει τα χρώματα που θα μπορούμε να χρησιμοποιήσουμε.(Μελλοντική Επέκταση) Στο Demo τα χρώματα είναι με JavaScript.
CurveData	Για την λειτουργία του σχεδιασμού για να μπορεί να σχεδιαστεί κάτι. .(Μελλοντική Επέκταση)
Sound	Η κλάση αυτή θα έχει την δυνατότητα να μας ειδοποιεί όταν κάτι πάει λάθος την διάρκεια της σχεδίασης. .(Μελλοντική Επέκταση)
Cursor	Με την χρήση αυτής της κλάσης θα μπορούμε να αλλάξουμε την εμφάνιση του cursor.(Μελλοντική Επέκταση)
PaginationHelper	Η κλάση αυτή θα γίνει για να μας βοηθήσει στις σελίδες.
JsfUtil	Η κλάση αυτή είναι για την προειδοποίηση σε τυχόν λάθη

	που θα κάνουμε.
AuthenticationPhaseListener	Για την αυθεντικοποιηση χρήστη θα γίνεται σε συνεργασία με την Usermanager
AppUsersJpaController	Ο AppUserController που λειτουργεί με το JPA
WuserJpaController	Ο WuserController που λειτουργεί με το JPA
AppUsersController	Η κλάση αυτή θα χειρίζεται την λειτουργία του AppUsersController χρήστη και θα συνεργάζεται με την κλάση AppUsers στο πακέτο entity.
DrawingController	Η κλάση αυτή θα χειρίζεται την λειτουργία στην του Draw χρήστη
NavigationController	Η κλάση αυτή θα χειρίζεται την λειτουργία στην πορεία του χρήστη
WuserController	Η κλάση αυτή θα χειρίζεται την λειτουργία του Admin χρήστη και θα συνεργάζεται με την κλάση admin στο πακέτο entity.
IllegalOrphanException	Κλάση για τα Exception .
NonexistentEntityException	Κλάση για Entity Exception
PreexistingEntityException	Κλάση για τα Entity Exception
UserManager	Δημιουργεί τον χρήστη και το Session ελέγχει εάν ο χρήστης είναι εγγεγραμμένος
userdetails	Για την αποστολή Email χρήστη.
Μελλοντική κλάση	Λειτουργίες

Το μοντέλο των περιπτώσεων χρήσης (use case model)

Το μοντέλο περιπτώσεων χρήσης θεωρείται από τα ποιο βασικά για την ανάπτυξη του συστήματος γιατί μας περιγράφει τι θα κάνει η εφαρμογή που θα αναπτυχθεί περιληπτικά. Είναι ένα απλό σχέδιο που δείχνει την ροή των λειτουργιών που θα κάνει η εφαρμογή χωρίς να μπαίνει σε λεπτομέρειες και βαθιές έννοιες που θα δυσκολεύουν την ανάλυση .

Έχει κεντρικό ρόλο και θα χρησιμοποιηθεί για την ανάπτυξη τις εφαρμογής και την ιχνηλάτηση των απαιτήσεων τις εφαρμογής

Οι περιπτώσεις χρήσης μπορούν να χρησιμοποιηθούν με πολλούς διαφορετικούς τρόπους και σε διαφορετικές περιπτώσεις, όπως:

- Για την περιγραφή των προδιαγραφών του συστήματος
- Για τον προσδιορισμό των απαιτήσεων
- Για την τεκμηρίωση της λειτουργικότητας του συστήματος

Περιπτώσεις χρήσης

Εύρεση χειριστών

Το πρώτο βήμα για την ανάπτυξη του μοντέλου των περιπτώσεων χρήσης (ΠΧ) είναι η εύρεση των χειριστών (actors) του συστήματος. Ως χειριστή ορίζουμε ένα χρήστη του συστήματος με συγκεκριμένο ρόλο που αλληλεπιδρά με αυτό. Γα να βρούμε τους χειριστές θα πρέπει να δούμε τις απαιτήσεις όπως περιγράφονται στο πρόβλημα. Ένας χειριστής συμμετέχει σε μια περίπτωση χρήσης ενώ μια περίπτωση χρήσης μπορεί να θεωρηθεί ότι ορίζει μια ακολουθία από αλληλεπιδράσεις ανάμεσα σε ένα η πιο πολλούς χρήστες.
Όταν φτάσουμε στο σημείο να προσδιορίσουμε τους χειριστές πρέπει να είμαστε σίγουροι το πόσο πληρούν τις προϋποθέσεις ότι καλύπτουν όλες τις ανάγκες τις εφαρμογής μας.
Έτσι εκτός από τους χρήστες οι χειριστές μπορεί να είναι πχ κάποια άλλα συστήματα ή κάποια έξοδος η εκτύπωση η αποθήκευση σε κάποια βάση δεδομένων.

Η διαδικασία ανάπτυξης του μοντέλου των περιπτώσεων χρήσης συνίσταται στον προσδιορισμό των παρακάτω:

- Ποιος χρησιμοποιεί το σύστημα
- Πώς χρησιμοποιείται το σύστημα

Για την δημιουργία του gdesign μπορούμε να δημιουργήσουμε ένα αρχικό σύνολο περιπτώσεων χρήσης που είναι :

1. Δημιουργία Λογαριασμού χρήστη
2. Σύνδεση χρήστη με Σύστημα (Log In)
3. Διαχείριση λογαριασμού.
4. Δημιουργία σχεδίου.
5. Προβολή αποθηκευμένων σχεδίων.
6. Διαχείριση αποθηκευμένων σχεδίων.
7. Έξοδος από το σύστημα(Log Out).
8. Διαγραφή λογαριασμού χρηστή.

Αρχικό σύνολο περιπτώσεων χρήσης (scope map).
Η καταγραφή των περιπτώσεων χρήσης δεν μπορεί να θεωρηθεί ως τελική μιας και έχει στόχο να προσδιορίσει το μέγεθος του μοντέλου μας.

Προτείνεται το συγκεκριμένο μοντέλο για την προσέγγιση των περιπτώσεων χρήσης

1. Τίτλος περίπτωσης χρήσης :

1.1. Σύντομη περιγραφή:

….

1.2. Χειριστές:

2. Ροή γεγονότων.

2.1 Βασική ροή

….

2.2 Εναλλακτικές ροές

2.2.1 Εναλλακτική ροή 1

….

2.2.1 Εναλλακτική ροή 2

…

Περίπτωση χρήσης Δημιουργία Λογαριασμού χρήστη.

1. Τίτλος περίπτωσης χρήσης : Δημιουργία Λογαριασμού χρήστη

1.1. Σύντομη περιγραφή:

Ο χρήστης επιλέγει την εγγραφή στο σύστημα μέσω του link για την δημιουργία λογαριασμού**.** Η εφαρμογή εμφανίζει μια φόρμα διάλογου και εισαγωγής στοιχείων όπου ο χρήστης θα πρέπει να εισάγει τα στοιχεία του έτσι ώστε να εγγραφεί στο σύστημα και να δημιουργήσει λογαριασμό.

1.2. Χειριστές: χρήστης

2. Ροή γεγονότων.

2.1 Βασική ροή

1. Ο χρήστης πατά το link στην αρχική σελίδα για την εγγραφή στο σύστημα .
2. Η εφαρμογή επιστρέφει την φόρμα έγγραφης όπου ο χρήστης θα πρέπει να κάνει εισαγωγή στοιχείων.
3. Ο χρήστης κάνει εισαγωγή των στοιχείων του στην φόρμα που εμφανίζει το σύστημα και πατά το κουμπί οκ .
4. Η εφαρμογή ελέγχει την εγκυρότητα των στοιχείων.
5. Ο χρήστης επιβεβαιώνει την είσοδο στοιχείων και την εγγραφή του στην εφαρμογή.
6. Η εφαρμογή επιστρέφει μήνυμα για την εγγραφή του χρήστη και εμφανίζει κουμπί για επιστροφή στην αρχική σελίδα.
7. Ο χρήστης πατά το κουμπί επιστροφής στην αρχική σελίδα.
8. Η εφαρμογή επιστρέφει στην αρχική σελίδα.

2.2 Εναλλακτικές ροές

2.2.1 Εναλλακτική ροή 1

Σε οποιοδήποτε σημείο της περίπτωσης χρήσης ο χρήστης μπορεί να ακυρώσει τη

δημιουργία λογαριασμού. Το αποτέλεσμα είναι ότι δεν δημιουργείται λογαριασμός.

2.2.1 Εναλλακτική ροή 2

4 α . Η εφαρμογή εμφανίζει μήνυμα για ελλιπή είσοδο στοιχείων

4β . Η εφαρμογή εμφανίζει μήνυμα για την μη ορθή είσοδο στοιχείων

4 γ . Η εφαρμογή επιστρέφει στο βήμα 3.

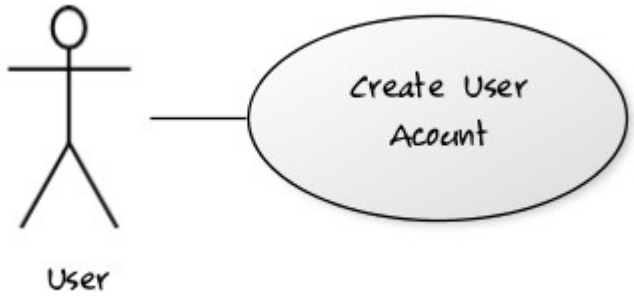
3. Ειδικές απαιτήσεις

Ο έλεγχος εισαγωγής θα γίνεται στην Server κομμάτι τις εφαρμογής .

4. Κατάσταση εισόδου (preconditions):Δεν υπάρχει.

5. Κατάσταση εξόδου (postconditions):

1. Ο λογαριασμός χρήστη δημιουργήθηκε επιτυχημένα ο λογαριασμός έχει ενεργοποιηθεί και ο χρήστης είναι πλέον εγγεγραμμένος χρήστης.
2. Ο λογαριασμός δε δημιουργήθηκε είτε γιατί ο χρήστης απέτυχε να εισάγει τη σωστή πληροφορία είτε γιατί ακύρωσε τη διαδικασία δημιουργίας λογαριασμού.



Εικόνα 1 Περίπτωση χρήσης Δημιουργία Λογαριασμού χρήστη.

Περίπτωση χρήσης Σύνδεση χρήστη με Σύστημα (Log In).

1. Τίτλος περίπτωσης χρήσης : Σύνδεση χρήστη με Σύστημα (Log In)

1.1. Σύντομη περιγραφή:

Η περίπτωση χρήσης log in επιτρέπει στον ανώνυμο χρήστη να ταυτοποιηθεί από την εφαρμογή και να αποκτήσει πρόσβαση στη λειτουργικότητα της ανάλογα με το ρόλο του. Αν ο ρόλος του αντιστοιχεί σε χρήστη τότε αποκτά το ρόλο του καταχωρημένου χρήστη. Σε αντίθετη περίπτωση αποκτά το ρόλο του Διαχειριστή του συστήματος. Τέλος εάν ο χρήστης δεν έχει λογαριασμό στο σύστημα του δίνεται η δυνατότητα να δημιουργήσει ένα νέο λογαριασμό πελάτη.

1.2. Χειριστές: χρήστης

2. Ροή γεγονότων.

2.1 Βασική ροή

Αυτή η περίπτωση χρήσης αρχίζει όταν ο πελάτης επιλέξει τη λειτουργία «log-in".

1. Η εφαρμογή παρουσιάζει τη φόρμα εισαγωγής στο σύστημα όπου ζητά από το χρήστη να δώσει το username και το password.
2. Ο χρήστης εισάγει το username και το password.
3. Η εφαρμογή ελέγχει ότι το ζευγάρι των τιμών username/password αντιστοιχεί σε έναν χρήστη του συστήματος.
4. Ο χρήστης εισέρχεται στο σύστημα αποκτώντας πρόσβαση στη λειτουργικότητα του ρόλου που έχει ο χρήστης (καταχωρημένος χρήστης ή διαχειριστής του συστήματος).
5. Το σύστημα παρουσιάζει μήνυμα καλωσορίσματος του χρήστη.

Η περίπτωση χρήσης τελειώνει.

2.2 Εναλλακτικές ροές

2.2.1 Εναλλακτική ροή 1

Ο χρήστης δεν έχει λογαριασμό. Το σύστημα επιτρέπει την «Δημιουργία Λογαριασμού Πελάτη» σε όσους χρήστες δεν έχουν λογαριασμό.

2.2.1 Εναλλακτική ροή 2

Το σύστημα δε μπόρεσε να αναγνωρίσει το συνδυασμό username/password.

3α. Το σύστημα παρουσιάζει στο χρήστη σχετικό μήνυμα.

3β. Η ροή μεταφέρεται στο βήμα 1 και εμφανίζει μήνυμα προειδοποιήσεις μην ορθής εισαγωγής στοιχείων .

3. Ειδικές απαιτήσεις

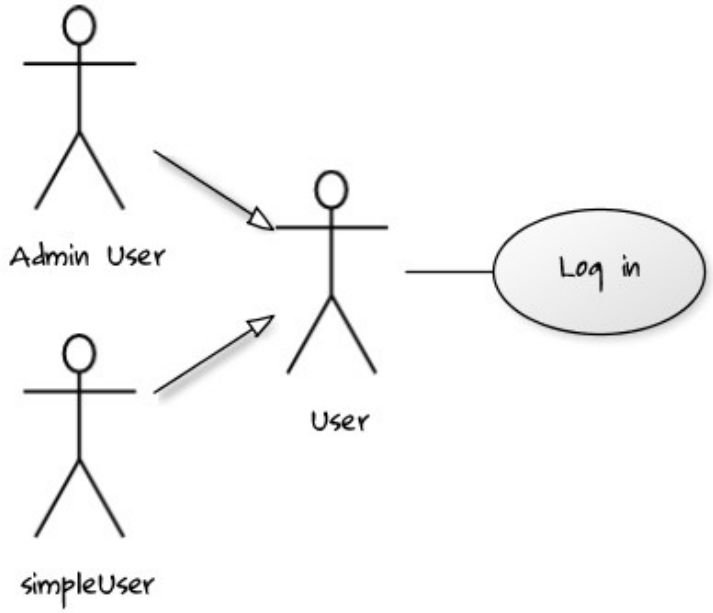
Η πληροφορία username/password πρέπει να μεταφέρεται κρυπτογραφημένη από τον internet browser στον web server έτσι ώστε να αποφύγουμε τον κίνδυνο υποκλοπής δεδομένων.

5. Κατάσταση εξόδου (postconditions):

1. Ο πελάτη ταυτοποιήθηκε επιτυχημένα από το σύστημα και ξεκίνησε να χρησιμοποιεί

το σύστημα

2. Το σύστημα απέτυχε να ταυτοποιήσει τον πελάτη και παραμένει με το ρόλο του ανώνυμου πελάτη.



Εικόνα 2 Περίπτωση χρήσης Σύνδεση χρήστη με Σύστημα (Log In).

Περίπτωση χρήσης Διαχείριση λογαριασμού.

1. Τίτλος περίπτωσης χρήσης : Διαχείριση λογαριασμού.

1.1. Σύντομη περιγραφή:

Η περίπτωση χρήσης Διαχείριση λογαριασμού επιτρέπει στον χρήστη να διαχειριστεί τον λογαριασμό του και να τροποποίηση τα στοιχεία του.

1.2. Χειριστές: χρήστης

2. Ροή γεγονότων.

2.1 Βασική ροή

Αυτή η περίπτωση χρήσης αρχίζει όταν ο πελάτης επιλέξει τη λειτουργία Διαχείριση λογαριασμού.

1. Η εφαρμογή παρουσιάζει τη φόρμα τροποποιήσεις
2. Ο χρήστης εισάγει τις τροποποιήσεις που θέλει να κάνει.
3. Η εφαρμογή ελέγχει για την ορθότητα των στοιχείων και επιστρέφει μήνυμα επιβεβαίωσης
4. Ο χρήστης επιβεβαιώνει την τροποποιήσει στοιχείων πατώντας το οκ
5. Το σύστημα παρουσιάζει μήνυμα ορθής τροποποίησης στοιχείων.

Η περίπτωση χρήσης τελειώνει.

2.2 Εναλλακτικές ροές

2.2.1 Εναλλακτική ροή 1

Ο χρήστης ακυρώνει την τροποποίηση. Το σύστημα επιτρέπει την «Διαχείριση λογαριασμού» σε όσους χρήστες έχουν λογαριασμό.

2.2.1 Εναλλακτική ροή 2

Το σύστημα δε μπόρεσε να αναγνωρίσει ορθά στοιχειά στην εισαγωγή (λάθος ημερομηνία).

2α. Το σύστημα παρουσιάζει στο χρήστη σχετικό μήνυμα.

2β. Η ροή μεταφέρεται στο βήμα 2.

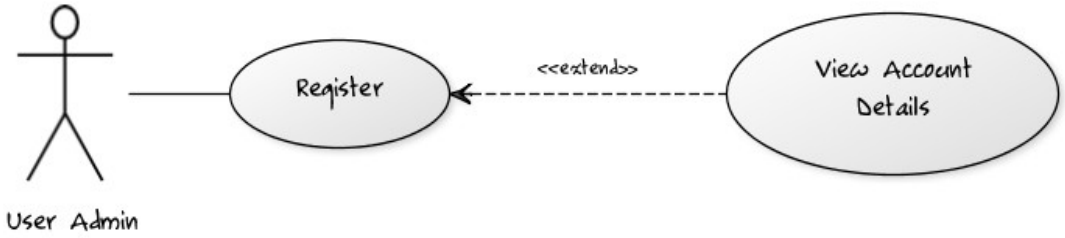
3. Ειδικές απαιτήσεις

Η εισαγωγή στοιχείων πρέπει να είναι στα αγγλικά.

4. Κατάσταση εισόδου (preconditions):Δεν υπάρχει.

5. Κατάσταση εξόδου (postconditions):

6. Η εισαγωγή έγινε επιτυχημένα από το σύστημα.
7. Το σύστημα απέτυχε να τροποποίηση τα στοιχεία που άλλαξαν λόγο μη συμβατότητας ορθότητας τις εισαγωγής.



Εικόνα 3 Περίπτωση χρήσης Διαχείριση λογαριασμού.

Περίπτωση χρήσης Δημιουργία σχεδίου.

1. Τίτλος περίπτωσης χρήσης : Δημιουργία σχεδίου.

1.1. Σύντομη περιγραφή:

Η περίπτωση χρήσης Δημιουργία σχεδίου. Επιτρέπει στον χρήστη να δημιουργήσει σχέδιο η διάγραμμα.

1.2. Χειριστές: χρήστης

2. Ροή γεγονότων.

2.1 Βασική ροή

Αυτή η περίπτωση χρήσης αρχίζει όταν ο πελάτης επιλέξει τη λειτουργία Διαχείριση λογαριασμού.

1. Η εφαρμογή παρουσιάζει τη φόρμα σχεδίασης.
2. Ο χρήστης σχεδιάζει το σχέδιο η το διάγραμμα που θέλει να κάνει και όταν τελείωση πατά την αποθήκευση.
3. Η εφαρμογή επιστρέφει μήνυμα επιβεβαίωσης αποθήκευσης.
4. Ο χρήστης επιβεβαιώνει την αποθήκευση σχεδίου πατώντας το οκ.
5. Το σύστημα παρουσιάζει μήνυμα ορθής αποθήκευσης σχεδίου.

Η περίπτωση χρήσης τελειώνει.

2.2 Εναλλακτικές ροές

2.2.1 Εναλλακτική ροή 1

Ο χρήστης ακυρώνει την σχεδίαση. Το σύστημα επιτρέπει την «Δημιουργία σχεδίου» σε όσους χρήστες έχουν λογαριασμό.

2.2.1 Εναλλακτική ροή 2

Το σύστημα δε μπόρεσε να αποθηκεύσει το σχέδιο.

2α. Το σύστημα παρουσιάζει στο χρήστη σχετικό μήνυμα.

2β. Η ροή μεταφέρεται στο βήμα 2.

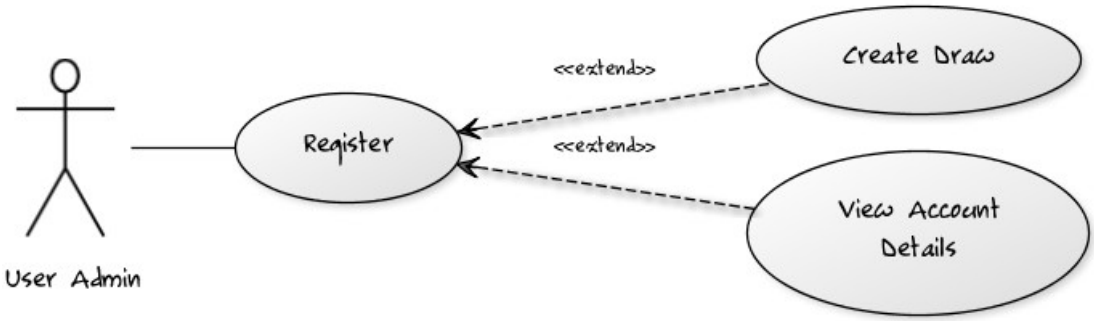
3. Ειδικές απαιτήσεις

Δεν υπάρχει.

4. Κατάσταση εισόδου (preconditions):Δεν υπάρχει.

5. Κατάσταση εξόδου (postconditions):

1. Η αποθήκευση έγινε επιτυχημένα από την εφαρμογή.
2. Η εφαρμογή απέτυχε να αποθηκεύσει το σχέδιο λόγο τεχνικών δυσκολιών.



Εικόνα 4 Περίπτωση χρήσης Δημιουργία σχεδίου.

Περίπτωση χρήσης Προβολή αποθηκευμένων σχεδίων.

1. Τίτλος περίπτωσης χρήσης : Προβολή αποθηκευμένων σχεδίων.

1.1. Σύντομη περιγραφή:

Η περίπτωση χρήσης Προβολή αποθηκευμένων σχεδίων επιτρέπει στον χρήστη να δει τα αποθηκευμένα σχέδια.

1.2. Χειριστές: χρήστης

2. Ροή γεγονότων.

2.1 Βασική ροή

Αυτή η περίπτωση χρήσης αρχίζει όταν ο πελάτης επιλέξει τη λειτουργία Προβολή αποθηκευμένων σχεδίων.

1. Η εφαρμογή παρουσιάζει τη φόρμα Προβολή υποθηκευμένων σχεδίων.
2. Ο χρήστης διαλέγει από την λίστα την αποθήκευση που θέλει.
3. Η εφαρμογή επιστρέφει την επιλεγμένη αποθήκευση με τα στοιχεία τις

Η περίπτωση χρήσης τελειώνει.

2.2 Εναλλακτικές ροές

2.2.1 Εναλλακτική ροή 1

Ο χρήστης ακυρώνει την Προβολή Αποθηκευμένων στοιχείων. Το σύστημα επιτρέπει την «Προβολή αποθηκευμένων σχεδίων» στον χρήστη που έχει λογαριασμό.

3. Ειδικές απαιτήσεις

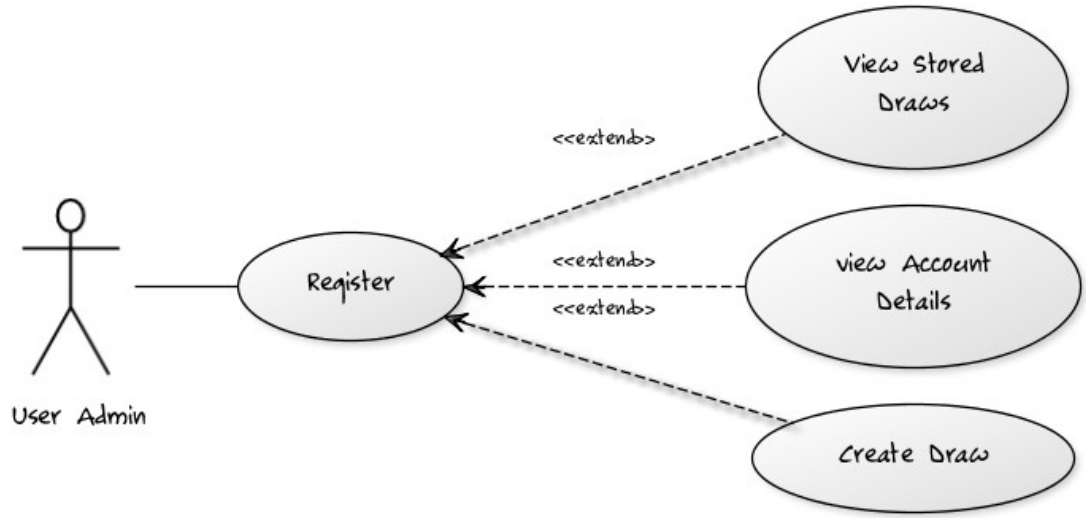
Πρέπει να υπάρχει κάποιο αποθηκευμένο σχέδιο για να προβληθεί κάτι.

4. Κατάσταση εισόδου (preconditions):Δεν υπάρχει.

5. Κατάσταση εξόδου (postconditions):

Η Προβολή έγινε επιτυχημένα από την εφαρμογή.

Το σύστημα απέτυχε να προβάλει τα αποθηκευμένα σχέδια επειδή δεν υπάρχει κάτι αποθηκευμένο.



Εικόνα 5 Περίπτωση χρήσης Προβολή αποθηκευμένων σχεδίων.

Περίπτωση χρήσης Έξοδος από το σύστημα(Log Out).

1. Τίτλος περίπτωσης χρήσης : Έξοδος από το σύστημα(Log Out).

1.1. Σύντομη περιγραφή:

Η περίπτωση χρήσης Έξοδος από το σύστημα(Log Out) επιτρέπει στον χρήστη να βγει από την εφαρμογή.

1.2. Χειριστές: χρήστης

2. Ροή γεγονότων.

2.1 Βασική ροή

Αυτή η περίπτωση χρήσης αρχίζει όταν ο πελάτης επιλέξει τη λειτουργία Έξοδος από το σύστημα(Log Out).

1. Η εφαρμογή έχει στις επιλογές τη δυνατότητα σε link για έξοδο από την εφαρμογή

2. Ο χρήστης πατά έξοδος από την εφαρμογή.
3. Η εφαρμογή επιστρέφει μήνυμα επιβεβαίωσης εξόδου.
4. Ο χρήστης επιβεβαιώνει το μήνυμα πατώντας το οκ
5. Το σύστημα παρουσιάζει μήνυμα εξόδου από τον λογαριασμό του χρήστη και επιστρέφει στην αρχική οθόνη.

Η περίπτωση χρήσης τελειώνει.

2.2 Εναλλακτικές ροές

2.2.1 Εναλλακτική ροή 1

Ο χρήστης ακυρώνει την έξοδο. Το σύστημα επιτρέπει την Έξοδος από το σύστημα(Log Out) σε όσους χρήστες είναι ήδη στον λογαριασμό τους.

2.2.1 Εναλλακτική ροή 2

Το σύστημα δε μπόρεσε να εξέλθει σωστά από την εφαρμογή λόγο διακοπής τις σύνδεσης.

2α. Το σύστημα παρουσιάζει στο χρήστη σχετικό μήνυμα.

2β. Η ροή μεταφέρεται στην περίπτωση χρήσης Σύνδεση χρήστη με Σύστημα (Log In).

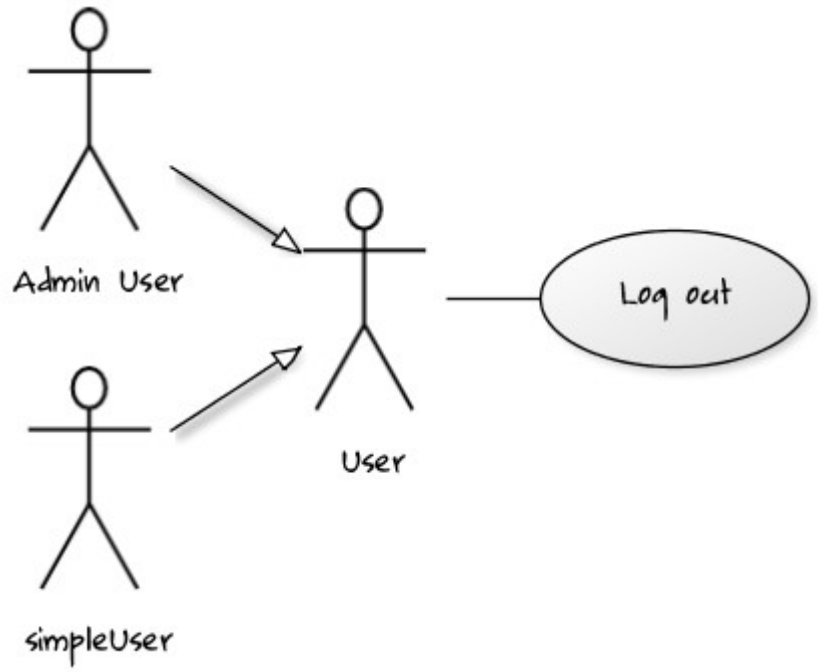
3. Ειδικές απαιτήσεις

Ενεργή σύνδεση internet.

4. Κατάσταση εισόδου (preconditions):Δεν υπάρχει.

5. Κατάσταση εξόδου (postconditions):

1. Η Έξοδος έγινε επιτυχημένα από την εφαρμογή.
2. Η εφαρμογή απέτυχε να εξέλθει από το Session του χρήστη.



Εικόνα 6 Περίπτωση χρήσης Έξοδος από το σύστημα(Log Out).

Περίπτωση χρήσης Διαγραφή λογαριασμού χρήστη.

1. Τίτλος περίπτωσης χρήσης : Δημιουργία σχεδίου.

1.1. Σύντομη περιγραφή:

Η περίπτωση χρήσης Δημιουργία σχεδίου επιτρέπει στον χρήστη να δημιουργήσει σχέδιο η διάγραμμα.

1.2. Χειριστές: χρήστης

2. Ροή γεγονότων.

2.1 Βασική ροή

Αυτή η περίπτωση χρήσης αρχίζει όταν ο πελάτης επιλέξει τη Διαγραφή λογαριασμού.

8. Η εφαρμογή παρουσιάζει τη φόρμα τροποποίησης
9. Ο χρήστης εισάγει τις τροποποιήσεις που θέλει να κάνει.
10. Η εφαρμογή ελέγχει για την ορθότητα των στοιχείων και επιστρέφει μήνυμα επιβεβαίωσης.
11. Ο χρήστης επιβεβαιώνει την τροποποιήσει στοιχείων πατώντας το οκ
12. Το σύστημα παρουσιάζει μήνυμα ορθής τροποποίησης στοιχείων.

Η περίπτωση χρήσης τελειώνει.

2.2 Εναλλακτικές ροές

2.2.1 Εναλλακτική ροή 1

Ο χρήστης ακυρώνει την τροποποίηση. Το σύστημα επιτρέπει την «Διαχείριση λογαριασμού» σε όσους χρήστες έχουν λογαριασμό.

2.2.1 Εναλλακτική ροή 2

Το σύστημα δε μπόρεσε να αναγνωρίσει ορθά στοιχεία στην εισαγωγή (λάθος ημερομηνία).

2α. Το σύστημα παρουσιάζει στο χρήστη σχετικό μήνυμα.

2β. Η ροή μεταφέρεται στο βήμα 2.

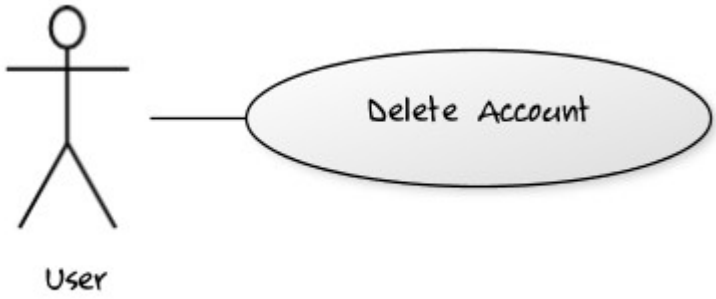
3. Ειδικές απαιτήσεις

Η εισαγωγή στοιχείων πρέπει να είναι στα αγγλικά.

4. Κατάσταση εισόδου (preconditions):Δεν υπάρχει.

5. Κατάσταση εξόδου (postconditions):

13. Η εισαγωγή έγινε επιτυχημένα από το σύστημα.
14. Το σύστημα απέτυχε να τροποποίηση τα στοιχεία που άλλαξαν λόγο μη συμβατότητας ορθότητας τις εισαγωγής.



Εικόνα 7 Περίπτωση χρήσης Διαγραφή λογαριασμού χρήστη

Κεφάλαιο 5. Τεχνικές δημιουργίας εφαρμογών

Στις περισσότερες περιπτώσεις μια εφαρμογή δεν δημιουργείται για να εκτελεστεί μόνο μια φορά. Θα πρέπει να είναι εύκολο να συντηρηθεί, τροποποιηθεί και επεκταθεί όταν αυτό κριθεί αναγκαίο. Για αυτό το λόγο απαιτείται ένα μοντέλο ανάπτυξης που θα επιτρέπει την εύκολη συντήρηση του λογισμικού.
Οι ομάδες ανάπτυξης θέλοντας να δημιουργήσουν αυτό το μοντέλο έχουν καταγράψει διάφορα κριτήρια καθώς και διάφορα πλαίσια τα οποία επιτρέπουν την δημιουργία του επιθυμητού αποτελέσματος με προσεκτική σχεδίαση. Μερικά κριτήρια από αυτά είναι:

- Επαναχρησιμοποίηση
- Συνδεσιμότητα
- Συνοχή
- Πληρότητα

και θα περιγραφούν περιληπτικά στη συνέχεια.

Κριτήρια

Επαναχρησιμοποίηση (reusability)

Όσο πιο γενική είναι μια κλάση τόσο πιο εύκολα μπορούμε να την επαναχρησιμοποιήσουμε. Πριν ενσωματώσουμε μια μέθοδο σε μια συγκεκριμένη κλάση πρέπει να σκεφτούμε τη δυνατότητα επαναχρησιμοποίησης της κλάσης.

Συνδεσιμότητα (coupling)

Ίσως και το ποιο σημαντικό κριτήριο από όλα γιατί μας δίνει την δυνατότητα να μπορούμε να επεκτείνουμε και να συντηρήσουμε την εφαρμογής μας πιο εύκολα. Στόχος μας είναι η σχεδόν μηδενική σύνδεση των μεθόδων κλάσεων και της εφαρμογής έτσι ώστε να μπορούμε τροποποιήσουμε την εφαρμογή χωρίς να έχουμε πρόβλημα. Υπάρχουν αρκετά πλαίσια που μας δίνουν την δυνατότητα να έχουμε χαμηλή σύζευξη έτσι ώστε να μπορούμε να έχουμε τα επιθυμητά αποτελέσματα.

Συνοχή (cohesion)

Θα πρέπει κάθε κλάση μας να ενθυλακώνει αντικείμενα του προβλήματος μας ώστε να μπορεί να είναι αυτόνομη και λογικά πλήρης, ως τμήμα του συστήματος, έτσι ώστε το υπόλοιπο της εφαρμογής να μπορεί να επικοινωνεί αποτελεσματικά με αυτήν.

Πληρότητα (completeness)

Θα πρέπει να δώσουμε μεγάλη προσοχή στον βαθμό πληρότητας που έχει η κάθε κλάση έτσι ώστε να είναι επαναχρησιμοποιήσιμη κατά την διάρκεια τις εφαρμογής.

Ο Νόμος τις Δήμητρας (LoD) ή Principle of Least Knowledge

Είναι μια σχεδιαστική μεθοδολογία για την ανάπτυξη λογισμικού και αντικατοπτρίζει τον αντικειμενοστραφή προγραμματισμό. Στην γενική της μορφή αναφέρεται στην συνδεσιμότητα .

Ο Νόμος στο αντικειμενοστραφή προγραμματισμό περιγράφεται ως ακολούθως:

Η μέθοδος m ενός αντικειμένου ο μπορεί να καλέσει μέθοδο από τα ακόλουθα αντικείμενα

1. Του εαυτού της.
2. Αντικείμενα δημιουργημένα μέσα στο m.
3. Αντικείμενα του ο.
4. Καθολικές μεταβλητές προσβάσιμες μόνο από ο και μέσα στην εμβέλεια του m.

«Κάθε κλάση θα πρέπει να γνωρίζει τις μονάδες που σχετίζονται στενά με αυτή»

Πότε να εφαρμόσουμε το Νόμο της Δήμητρας.

Οι παρακάτω περιπτώσεις θεωρούνται πεδίο άμεσης εφαρμογής του Νόμου της Δήμητρας. Όταν έχουμε τα εξής δείγματα τότε έχουμε πάρα πολλές δυνατότητες χρήσης του νόμου.

1. Αλυσιδωτές get δηλώσεις

Έστω ότι έχουμε

Price = object.getX ().getY ().getTheValue ();

Ακριβώς το ίδιο με

License = person.getWallet ().getDriversLicense ();

2. Προσωρινά αντικείμενα .

```
Wallet tempWallet = person.getWallet ();
License = tempWallet.getDriversLicense ();
```

είναι ακριβώς το ίδιο με τις get δηλώσεις αλλά πιο δύσκολο να εντοπιστούν.

3. Εισαγωγή κλάσεων Imports

Όταν δηλαδή κάνουμε αρκετά import’s στην εφαρμογή μας τότε δημιουργούμε μεγάλη συνδεσιμότητα στην εφαρμογή .

```
import crud.enitys.Person;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
```

Ως αποτέλεσμα της χρήσης του Νόμου της Δήμητρας θα έχουμε καλή επαναχρησιμοποίηση του κώδικα μας, επιτυγχάνοντας την εύκολη συντήρηση και την επεκτασιμότητα του .

Κεφάλαιο 6.Frameworks

Μετά την εμφάνιση τις Java και την ανάπτυξη του διαδικτύου άρχισε η δομή συναρτήσεων που διευκολύνουν την δημιουργία κώδικα. Έτσι εμφανίστηκαν τα πρώτα Framework που χρησιμοποιούνται για την παράγωγη κώδικα.
Το framework είναι ένα καλά φτιαγμένο μοντέλο που έχει ήδη έτοιμες τις συναρτήσεις που θέλουμε να χρησιμοποιήσουμε έτσι ώστε να μην χρειάζεται να τις ξαναγράφουμε από την αρχή. Ας σκεφτούμε την περίπτωση ότι θα έπρεπε να γράψουμε από την αρχή όλες τις συναρτήσεις που χρησιμοποιούμε και μπορούμε να καταλάβουμε την ευκολία που μας δίνει η χρήση framework.
Με τα framework έχουμε την δυνατότητα να περιορίσουμε τον χρόνο μας από τα έξης

Data Validation
Το Validation σε όλα τα Framework είναι ήδη έτοιμο να παραμετροποιηθει και να λειτουργήσει.
Έχουμε τα Validations για τον έλεγχο εισόδου από τον χρήστη.
Έχουμε τα Validations για τον έλεγχο κάποιων String.

Page Rules
Μπορούμε να δημιουργήσουμε κάποιους κανόνες που θα υπακούουν η σελίδες μας
Ποια θα ξεκινά πρώτη
Τι θα εμφανίζεται σε κάθε σελίδα

Http Request Response
Έχουμε την λειτουργία του Http request.
Έχουμε την λειτουργία του Http response.

DBConnection
Η σύνδεση με την βάση ήταν ένα πρόβλημα που απασχολούσε από πάντα τους προγραμματιστές
Γι την επίλυση του εμφανίστηκαν και τα framework που έχουν την δυνατότητα να κάνουν την σύνδεση με την βάση πιο εύκολη και πιο απλή.

Security
Ίσως ένα από τα πιο σημαντικά σημεία που έχει ένα framework είναι η ασφάλεια που θα προσφέρει έτσι ώστε τα δεδομένα μας να είναι ασφαλείς από παρείσακτους.
Το περιβάλλον το οποίο θα έχουμε θα είναι ήδη έτοιμο και δοκιμασμένο για κοινές τακτικές ασφάλειας .

Για να γίνει κατανοητή η χρήση κάποιου framework έχουμε το εξής.

Για να γράψουμε τον εξής κώδικα χρειάζεται πολλής κόπος και χρόνος.

```
<table>
   <tr>
      <td><input name="j_idt6:j_idt8" id="j_idt6:j_idt8:0" value="1"
            type="checkbox" checked="checked" />
         <label for="j_idt6:j_idt8:0" class=""> Item 1</label>
      </td>
      <td><input name="j_idt6:j_idt8" id="j_idt6:j_idt8:1" value="2"
            type="checkbox" checked="checked" />
         <label for="j_idt6:j_idt8:1" class=""> Item 2</label>
      </td>
   </tr>
```

```
</table>
```

Αντί αυτού με το framework έχουμε το εξής

```
<h:selectManyCheckbox value="#{userData.data}">
   <f:selectItem itemValue="1" itemLabel="Item 1" />
   <f:selectItem itemValue="2" itemLabel="Item 2" />
</h:selectManyCheckbox>
```

Οπότε με την χρήση τις h:selectManyCheckbox έχουμε την ίδια ακριβώς λειτουργία με το κώδικα παραπάνω μόνο που δεν χρειάζεται να γράψουμε όλο των κώδικα που έχουμε πιο πάνω.

Με τον ίδιο ακριβώς τρόπο για το

```
<a href="/hello/page.jsf">Page</a>
```

Μπορούμε να έχουμε το εξής

```
<h:link value="Page" outcome="page" />
```

Οπότε έχουμε πιο εύκολη την δημιουργία του κώδικα μας .
Κάθε framework έχει φτιαχτεί για να αντιμετωπίσει κάποια συνηθισμένα προβλήματα .Τα προβλήματα αυτά έχουν εμφανιστεί με την πάροδο των χρόνων και την εμπειρία που αποκτήθηκε με αυτά .

Struts Framework

Το Struts Framework μας δίνει λύση στο πρόβλημα του MVC μια τακτική η οποία χρησιμοποιείται σχεδόν από τις περισσότερες εταιρίες.
Το Mvc αποσκοπεί στην δημιουργία κλάσεων που θα αποθηκεύουν τα δεδομένα στην βάση για παράδειγμα , toView κομμάτι που θα υποδεχτεί την είσοδο από τον χρήστη και την Λογική Controller (Business Logic) που θα όλο αυτό να λειτουργήσει και να τρέχει. Εχει προσαρμοστεί πιο πολλή στο κομμάτι του Controller (Business Logic)δηλαδή θα μας δώσει τα εργαλεία που θέλουμε για την πιο εύκολη δημιουργία και χειρισμό του Controller .

Jsf Framework

Το JSF μας δίνει την λύση στο πρόβλημα του Mvc με μια διαφορετική μορφή. Το Jsf δίνει τα εργαλεία που χρειαζόμαστε για να αναπτύξουμε το view κομμάτι μιας εφαρμογής και όχι όπως το struts όπου είναι πιο καλά φτιαγμένο για την δημιουργία του Controller. θα κάνουμε μια μικρή αναφορά για το JSF αλλά αφενός μεν δεν είναι στο θέμα τις διπλωματικής η ανάλυση του αφετέρου μια αναφορά για το συγκεκριμένο framework θα ήταν πολλή λιτή και περιορισμένη που θα αδικούσε το Jsf και δεν θα αναδείκνυε τις δυνατότητες του.

Spring
Τα Spring είναι ένα framework με πολλές λειτουργίες. Ίσως είναι ότι το πιο δημοφιλές υπάρχει σήμερα στην αγορά χωρίζεται σε κομμάτια.

Core όπου έχει τα βασικά καθώς και IoC και DI.
Bean όπου έχουμε BeanFactory και την Factory pattern.
Context όπου έχουμε την βάση από το Core και το Bean για την πρόσβαση σε αντικείμενα.

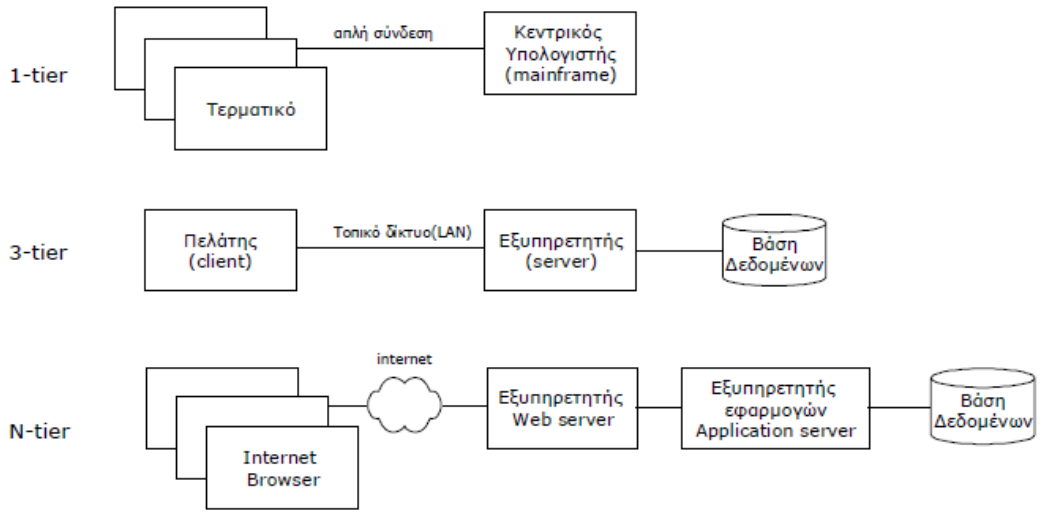
Expression Language όπου έχουμε την EL για πανίσχυρη εκμετάλλευση αντικειμένων.
JDBC όπου έχουμε την σύνδεση με την βάση.
ORM όπου έχουμε δημοφιλή ORM σαν το Hibernate JPA JDO.
OXM όπου έχουμε JAXB Castor XMLbeans.
JMS όπου έχουμε το γνωστό Java messaging service.
Transaction όπου έχουμε τα γνωστά Transaction.
Web όπου έχουμε IoC.
Web-Servlet όπου έχουμε MVC
Web-Struts όπου έχουμε το κλασικό Struts Framework μέσα σε μια spring εφαρμογή.
Web-portlet όπου έχουμε το MVC μέσα σε εφαρμογή για portlet.

Έχουν δημιουργηθεί διάφορα framework τα οποία μας δίνουν την δυνατότητα να γράψουμε κατευθείαν εκτελέσιμο κώδικα και να έχουμε γρήγορα ένα αποτέλεσμα.

Μερικά γνωστά framework είναι τα Jsf ,Spring,Struts, για την γλώσσα Java.

Κεφάλαιο 7. Αρχιτεκτονικές πολλών επιπέδων

Ιστορικά οι πρώτες εφαρμογές χαρακτηρίζονταν ενός επιπέδου, με την έννοια ότι το τερματικό το οποίο χειριζόταν ο χρήστης συνδεόταν άμεσα με τον κεντρικό υπολογιστή. Σήμερα είναι αρκετά συνηθισμένο να χρησιμοποιούνται αρχιτεκτονικές πολλών επιπέδων.



Εικόνα 8. Αρχιτεκτονικές πολλών επιπέδων

Αντίστοιχα στη δική μας περίπτωση θα χρησιμοποιήσουμε μια αρχιτεκτονική πολλαπλών επιπέδων όπου ο χρήστης θα έχει πρόσβαση στην εφαρμογή μέσω ενός προγράμματος πλοήγησης, το οποίο επικοινωνεί μέσω διαδικτύου με τον εξυπηρετητή Διαδικτύου (Web server) και αυτός με την εφαρμογή μας και τη βάση δεδομένων.

Κεφάλαιο 8.Σχεδίαση οθονών συστήματος

Εφόσον έχουμε βρει το ανθρώπινο δυναμικό και το χρόνο που θα χρειαστεί για να γίνει η ανάπτυξη μπορούμε να προχωρήσουμε στη σχεδίαση των οθονών της εφαρμογής.
Θα πρέπει να λάβουμε υπόψη μας απλούς κανόνες για την δομή τους έτσι ώστε να έχουμε κατάλληλη λειτουργία και αποδοτικότητα από τους χρήστες. Με την δημιουργία των οθονών θα είναι πάρα πολύ πιο εύκολη η δημιουργία της εφαρμογής καθώς θα υπάρχει μια κατανόηση του τι ακριβώς γίνεται.
Υπάρχουν εταιρίες που δίνουν άμεση προτεραιότητα στην δημιουργία του οπτικού μέρους καθώς μετά η εργασία που χρειάζεται είναι απλά η ένωση τον οθονών με την λογική της εφαρμογής. Πολλές από αυτές έχουν ολόκληρες ομάδες που ασχολούνται μόνο με την συγγραφή των View /Gui κομματιών έτσι ώστε να έχουν τα επιθυμητά αποτελέσματα. Συνήθως η συγγραφή των View's γίνεται πρώτα με μολύβι και χαρτί έτσι ώστε να γίνει μια παρουσίαση του έργου και του αποτελέσματος αυτού. Έχουν δημιουργηθεί πολλές μέθοδοι για την δημιουργία View's σας τα Wireframes τα Mock ups κ.α.

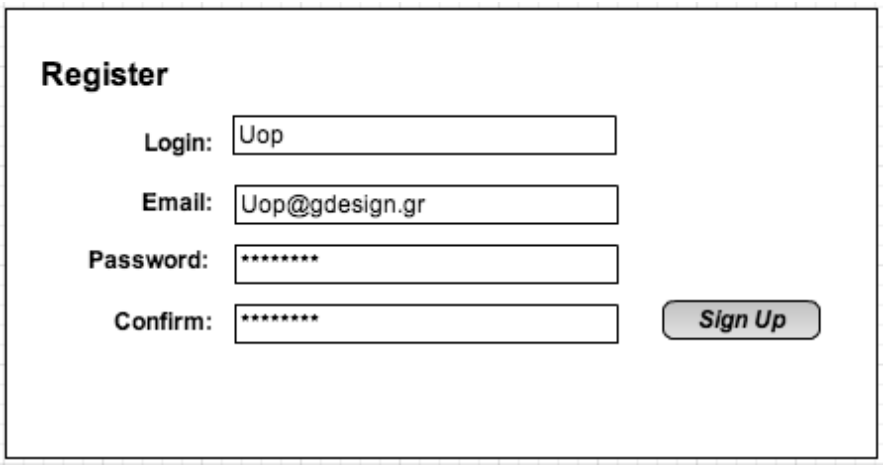
Αφού γίνει παρουσίαση του View και εξηγηθεί η λειτουργικότητα κάθε οθόνης αλλά και το σκεπτικό αυτής τότε αναλύεται η εφαρμογή κομμάτι-κομμάτι και παρουσιάζεται με κάποια από τις μεθοδολογίες που είπαμε.

Οθόνες

Η πρώτη οθόνη θα είναι η οθόνη υποδοχής του χρήστη και θα την ονομάσουμε starpage και θα έχει χαρακτηριστικά για την εφαρμογή μας και διάφορα ενδιαφέροντα σημεία για την εφαρμογή μας νέα ,κάποιο RSS feed κάποια στοιχεία για τα δικαιώματα των developer και menu με τις διάφορες επιλογές που θα είναι διαθέσιμες .

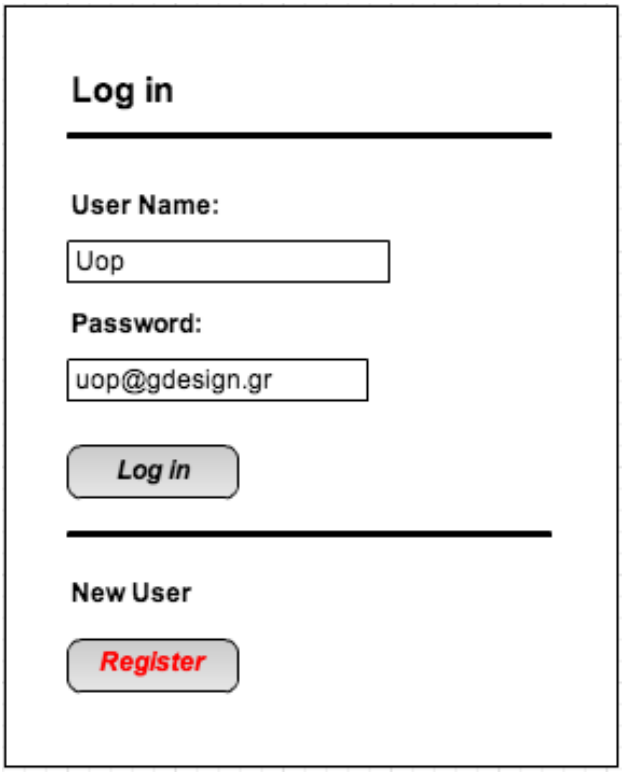
θα υπάρχει μια φόρμα επικοινωνίας γνωστή και ως contactForm όπου θα μπορεί κάποιος να επικοινωνήσει με την ομάδα η των developer που έχει την εφαρμογή καθώς και να στείλει προτάσεις για την καλυτέρευση τις εφαρμογής, παράπονα η διάφορες άλλες ιδέες που θα μπορούσαν να βοηθήσουν στην υλοποίηση και τη βελτίωση αυτής.

Θα υπάρχει επίσης και μια οθόνη για να διαβάσει κάποιος τις οδηγίες χρήσης του σχεδιαστικού εργαλείου.



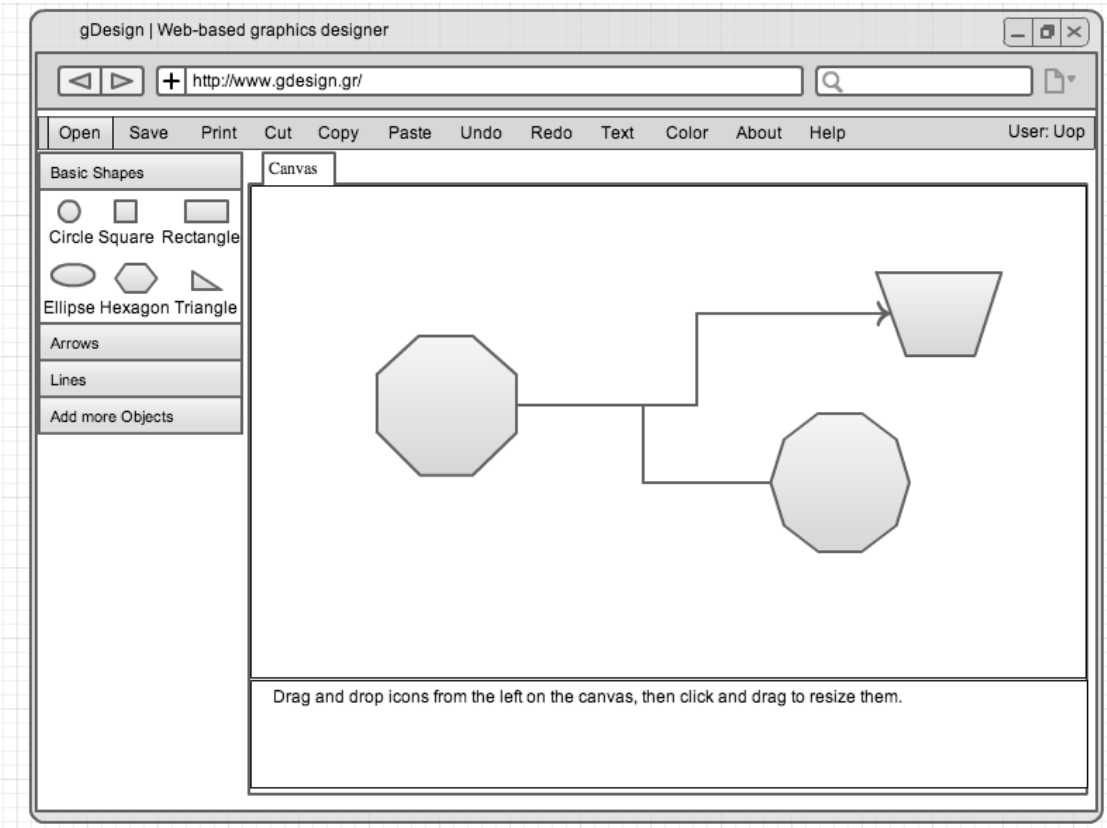
Εικόνα 9. Οθόνη εγγραφής χρήστη

Η οθόνη εγγραφής χρήστη θα είναι αρκετά απλή, καθώς για τη δοκιμαστική εφαρμογή το μόνο που χρειαζόμαστε είναι η δυνατότητα πιστοποίησης του χρήστη. Σε επόμενες εκδόσεις θα μπορούσαν να προστεθούν επιπλέον δημογραφικά στοιχεία (έτος γέννησης, φύλλο, ώστε να έχουν καλύτερη εικόνα των χρηστών της εφαρμογής), τηλέφωνο επικοινωνίας, κλπ.



Εικόνα 10. Οθόνη σύνδεσης με το σύστημα

Όταν ο χρήστης έχει εγγραφεί μπορεί να χρησιμοποιήσει την οθόνη σύνδεσης με το σύστημα για να αποκτήσει πρόσβαση στις λειτουργίες του συστήματος. Στην περίπτωση που δεν έχει εγγραφεί υπάρχει η δυνατότητα μέσω του πλήκτρου register μετάβασης στην οθόνη εγγραφής.



Εικόνα 11. Αρχική οθόνη συστήματος

Μετά την επιτυχημένη σύνδεση με την εφαρμογή ο χρήστης μεταβαίνει στην αρχική οθόνη της εφαρμογής. Μέσω αυτής της οθόνης ο χρήστης μπορεί να ανακτήσει παλαιότερα σχέδια, να αποθηκεύσει αυτά που σχεδιάζει, να εκτυπώσει, να εκτελέσει τις συνήθεις λειτουργίες επεξεργασίας (αποκοπή, αντιγραφή, επικόλληση, αναίρεση και επανάληψη), να επιλέξει τον τύπο γραμματοσειράς και χρώματος και να βρει βοήθεια.

Στα αριστερά βρίσκονται οι εργαλειοθήκες με τα έτοιμα σχήματα που μπορεί να εισάγει ο χρήστης στο σχήμα του. Επιπλέον δίνεται η δυνατότητα προσθήκης επιπλέον κατηγοριών έτοιμων σχημάτων.

Οι οθόνες θα είναι φτιαγμένες σε JSF που θα μεταφερθεί σε HTML και θα γίνει και χρήση JSF. Η κάθε οθόνη θα έχει την λειτουργία της ανάλογα με τις συνθήκες που επικρατούν. Η λογική της εφαρμογής (Business Logic) θα γραφτεί σε Java όπου και θα είναι το backend της εφαρμογής και η βάση της εφαρμογής όπου αποθηκεύονται τα δεδομένα θα είναι σε MySql.

Θα χρησιμοποιήσουμε για την διαμόρφωση της κάθε οθόνης CSS για να δημιουργήσουμε την κατάλληλη μορφοποίηση που θέλουμε χωρίς ιδιαίτερη δυσκολία στην συγγραφή κώδικα.

Θα γίνει και χρήση JavaScript για την δημιουργία διαδραστικότητας στο επίπεδο του client με κάποιο framework όπως π.χ jQuery. Έτσι ώστε να επιτύχουμε πιο δυναμικά εφέ στην εφαρμογή μας.

Τεχνολογίες δημιουργίας οθόνης.

Στη συνέχεια θα παρουσιάσουμε τις βασικές τεχνολογίες που θα χρησιμοποιήσουμε για τη δημιουργία των οθονών.

HTML CSS

HTML

Το HTML σημαίνει **H**yper **T**ext **M**arkup **L**anguage και είναι η πιο διαδεδομένη γλώσσα δημιουργίας web σελίδων .

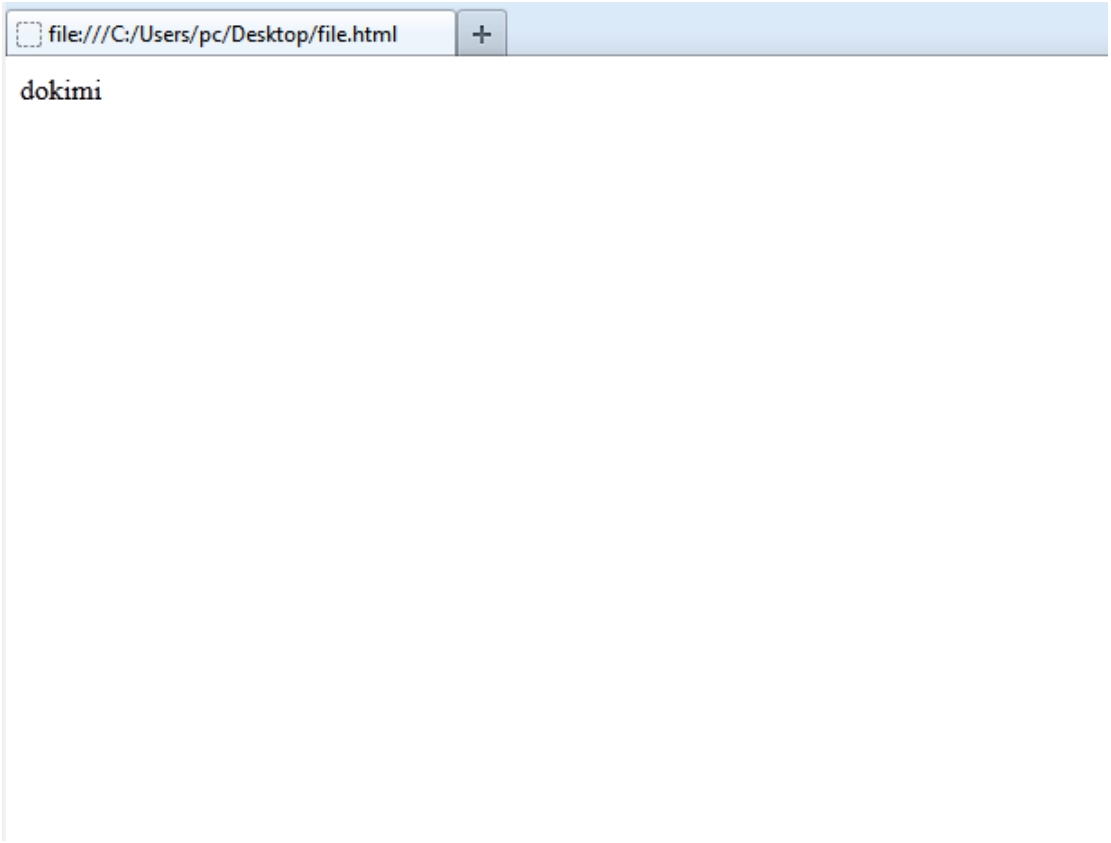
Το **H**yper**T**ext επιτρέπει τη σύνδεση μεταξύ των ιστοσελίδων με τη χρήση υπεσυνδέσμων, ώστε να είναι δυνατή η μετάβαση από μια σελίδα σε άλλη. Το **M**arkup **L**anguage χρησιμοποιείται για να εξηγήσει στο πρόγραμμα πλοήγησης πώς θα παρουσιάσει μια σελίδα. Η συγγραφή μιας σελίδας σε HTML είναι αρκετά απλή. Η HTML γράφεται υπό μορφή στοιχείων HTML τα οποία αποτελούνται από ετικέτες, οι οποίες περικλείονται μέσα σε σύμβολα «<» και «>» (για παράδειγμα <html>), μέσα στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες HTML συνήθως λειτουργούν ανά ζεύγη (για παράδειγμα <h1> και </h1>), με την πρώτη να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης. Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ. Πρακτικά οι ετικέτες χρησιμοποιούνται για να εξηγήσουν στο πρόγραμμα πλοήγησης πώς θα παρουσιάσει το κείμενο που βρίσκεται ανάμεσα τους.
Ένα μικρό παράδειγμα σελίδας HTML είναι το εξής.

```
<html>
<head>
<title>This is document title</title>
</head>
<body>
<h1>This is a heading</h1>
<p>Document description goes here.....</p>
</body>
</html>
```

Θα πρέπει επίσης το αρχείο να το αποθηκεύσουμε με την κατάληξη .html η .htm δηλαδή ονομαΑρχειου.html ή ονομαΑρχειου.htm.

Εδώ το αποθηκεύουμε με το όνομα file.html

Το συγκεκριμένο script θα μας δώσει το εξής.



Έχουμε γράψει την λέξη dokimi για να φανεί η σελίδα αλλιώς θα είναι άδεια.

CSS

Η CSS (Cascading Style Sheets-Διαδοχικά Φύλλα Στυλ) είναι μια γλώσσα υπολογιστή που ανήκει στην κατηγορία των γλωσσών φύλλων στυλ που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης. Χρησιμοποιείται δηλαδή για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε στις γλώσσες HTML και XHTML, δηλαδή για τον έλεγχο της εμφάνισης μιας ιστοσελίδας και γενικότερα ενός ιστοτόπου. Με τη CSS μπορούμε εύκολα να διατηρήσουμε ένα ομοιόμορφο στυλ εμφάνισης στις σελίδες ενώ παράλληλα είναι ιδιαίτερα απλή και η τροποποίηση του.

JSF

Το Java Server Faces είναι ένα MVC framework που απλουστεύει την δημιουργία της διεπαφής για server side διαδικτυακές εφαρμογές επαναχρησιμοποιώντας UI αντικείμενα στην σελίδα και στην μετάβαση από σελίδα σε σελίδα και ενεργές σύνδεσης και με την σύνδεση σε βάση δεδομένων.

Πλεονεκτήματα χρήσης Jsf

- ✓ Δυναμικά και επαναχρησιμοποιήσιμα UI.
- ✓ Εύκολη μεταφορά δεδομένων μεταξύ σελίδων και αντικειμένων.
- ✓ Καλύτερο control ανάμεσα στα server requests.
- ✓ Σύνδεση του client side με το server side τις εφαρμογής.

Jsf αντικείμενα

Δίνει την δυνατότητα στους developers να δημιουργήσουν web εφαρμογές μέσα από βιβλιοθήκες που χρησιμοποιεί. Έτσι ώστε να παρουσιαστεί διαφορετικά μέσα από διαφορετικές συσκευές (Html ,wireless ,wap).

Αρχιτεκτονική

Managed Beans
Managed Beans που μπορούν να χρησιμοποιηθούν σαν Model στην εφαρμογή μας δηλαδή να βοηθήσουν στην αποθήκευση. Έχουν get/set και business logic ακόμα και τα γνωστά Baked Beans.

Page Navigation
Κανόνες για την ανακατεύθυνση μπορούν να οριστούν στο faces-config.xml και να διαφοροποιήσουμε τη ροή των σελίδων στην εφαρμογή μας.
Οι κανόνες μπορούν να γίνουν με την βοήθεια των Managed Beans.

Οι κανόνες μπορούν να έχουν συνθήκες για το τι μπορεί να θεαθεί ανάλογα με την κατάσταση.

Validation

Για να μπορούμε να ελέγξουμε την είσοδο του χρήστη έχουμε την δυνατότητα της πιστοποίησης (validation).

Μπορούμε να ελέγξουμε την δυνατότητα ορθότητας και λογικής της εισόδου μας. Το framework JSF μας δίνει την δυνατότητα να δημιουργήσουμε κάποιους κανόνες για το validation που θέλουμε να έχει η εφαρμογή μας. Έχει κάποια έτοιμα validation tags και μπορεί να γίνει άμεση χρήση τους.

Μερικά validation Tags είναι

Tag & Description
f:validateLength ελέγχει το μήκος του String
f:validateLongRange ελέγχει το αριθμητικό μέγεθος
f:validateDoubleRange ελέγχει το μήκος των Float
f:validateRegex Ελέγχει τα JSF αντικείμενα με μια Κανονική έκφραση.
CustomValidator δημιουργεί ένα τυπικό validator

Βασικά Tags

Αναλύονται περιληπτικά μερικά βασικά Tags όπως έχουν δοθεί από την ομάδα κατασκευής.

Tag & Description
h:inputText Μεταφέρει HTML είσοδο τύπου ="text", text box.
h:inputSecret Μεταφέρει HTML είσοδο τύπου ="password", text box.

h:inputTextarea
Μεταφέρει HTML textarea field.

h:inputHidden
Μεταφέρει HTML είσοδο τύπου ="hidden".

h:selectBooleanCheckbox
Μεταφέρει HTML μόνο HTML check box.

h:selectManyCheckbox
Μεταφέρει HTML ένα γκρουπ από HTML check boxes.

h:selectOneRadio
Μεταφέρει μόνο HTML radio button.

h:selectOneListbox
Μεταφέρει HTML μονού τύπου list box.

h:selectManyListbox
Μεταφέρει HTML πολλαπλό list box.

h:selectOneMenu
Μεταφέρει HTML combo box.

h:outputText
Μεταφέρει HTML text.

h:outputFormat
Μεταφέρει HTML text. Και δέχεται παραμέτρους.

h:graphicImage
Μεταφέρει εικόνα.

h:outputStylesheet
Ενσωματώνει το CSS stylesheet σε HTML έξοδο.

h:outputScript
Ενσωματώνει ένα script σε HTML έξοδο.

h:commandButton
Μεταφέρει HTML είσοδο τύπου ="submit" button.

h:Link
Μεταφέρει HTML anchor.

h:commandLink
Μεταφέρει HTML anchor.

h:outputLink
Μεταφέρει HTML anchor.

h:panelGrid
Μεταφέρει HTML Tableσε μορφή Grid.

h:message
Μεταφέρει μήνυμα για JSF UI αντικείμενο.

h:messages
Μεταφέρει όλα τα μηνύματα για JSF UI αντικείμενα.

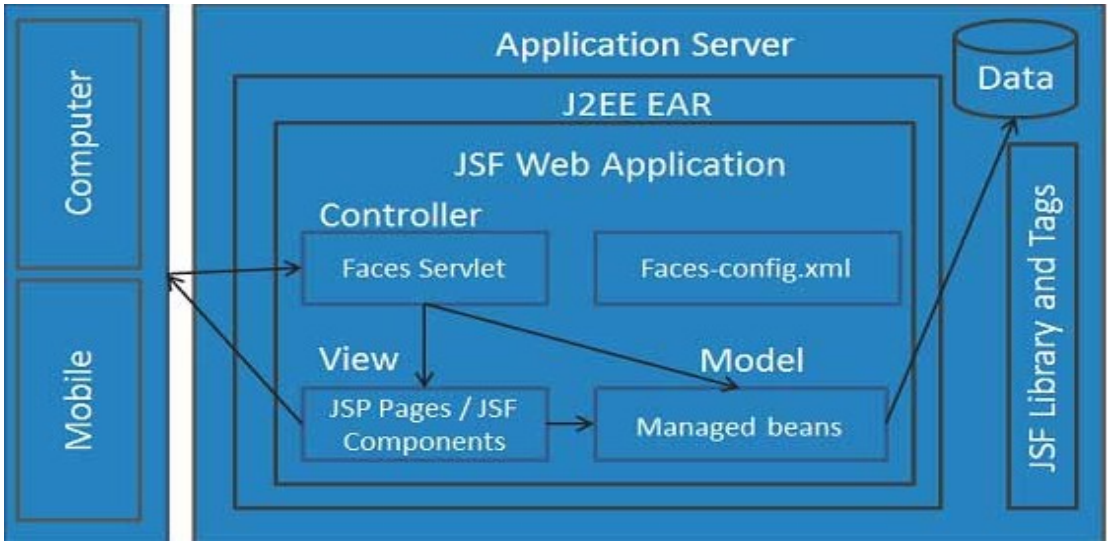
f:param
Θέτει παραμέτρους στο JSF UI αντικείμενο.

f:attribute
Θέτει χαρακτηριστικά στο JSF UI αντικείμενο.

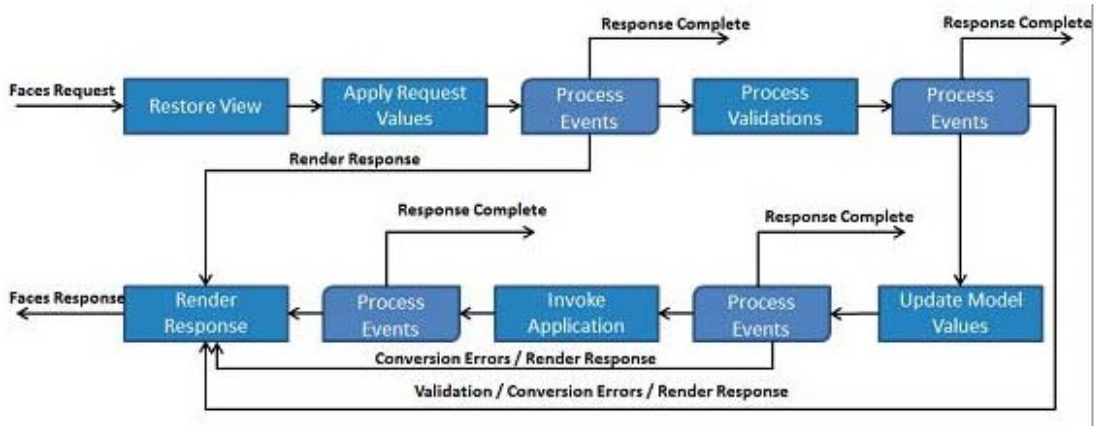
f:setPropertyActionListener
Βάζει Sets value από την ιδιότητα του managed bean.

Αρχιτεκτονική

Βλέπουμε ότι το Framework JSF υποστηρίζει πλήρως το μοντέλο που χρησιμοποιείται σήμερα στην αγορά το MVC.



Κύκλος ζωής jsf (Jsf Life Cycle).



Στην εφαρμογή μας θα χρησιμοποιήσουμε την τεχνολογία JSF όπου εξηγείται παρακάτω για τις δυνατότητες τις και τις λειτουργίες τις.

Η δημιουργία των ιστοσελίδων θα είναι το αποτέλεσμα σε HTML.
Αναλυτικά οι όλες οι σελίδες θα εμφανιστούν σε HTML γλώσσα στον Browser μας και θα έχουν το ακόλουθο μοτίβο.

```
<html>
<head>
<title>Τίτλος σελίδας</title>
</head>
<body>
</body>
</html>
```

Πάνω σε αυτό θα χρησιμοποιήσουμε ένα από τα κριτήρια, αυτό της επαναχρησιμοποίησης έτσι ώστε να μπορέσουμε να έχουμε πιο γρήγορη ανάπτυξη τις εφαρμογής καθώς και πιο εύκολη συντήρηση.

Οπότε θα χωρίσουμε την ιστοσελίδα σε κομμάτια ανάλογα με τις λειτουργίες που θα έχουν αυτά και θα κάνουμε ένα πρότυπο σελίδας που θα έχει η εφαρμογή μας και θα μπορεί να χρησιμοποιήσει ανάλογα με την στιγμή που θα βρίσκεται.

Δηλαδή όλη η σελίδα θα διαμορφωθεί ανάλογα με την κατάσταση που βρίσκεται εκείνη την στιγμή η εφαρμογή.

Δημιουργούμε τα εξής.

Header

Commons

Contents

Sidebar

Footer

Άμα κοιτάξουμε τα προσχέδια και τεχνικές δημιουργίας σελίδας θα παρατηρήσουμε ότι σχεδόν κάθε σελίδα μοιάζει με την άλλη.
Η επανάληψη του κώδικα δεν είναι μια καλή τακτική και την αποφεύγουμε έτσι για να αποφύγουμε αυτό το πρόβλημα τον κοινών Views.

Ένας τρόπος είναι να τα προσθέσουμε σε κάθε σελίδα. Και έτσι γίνεται η ιστοσελίδα στο εξής πρότυπο(blueprint).

Αλλά ο Header και ο Footer δεν έχει valid Html. Οπότε θα χρησιμοποιήσουμε ένα design pattern για την επίλυση του προβλήματος.

Και αυτό είναι η Decorator Design Pattern.

Η Decorator Design Pattern μας λύνει το πρόβλημα δηλαδή το template μας γεμίζει εφόσον το περιεχόμενο έχει προωθηθεί στο global template και λέγεται πλέον Layout.

Και ο κώδικας σε HTML για το συγκεκριμένο Layout σαν αποτέλεσμα θα είναι το εξής.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"><head></head><body>
      <div style="border-width:2px; border-color:green; border-
style:solid;">
         <h1>Default Header</h1>
      </div>
      <br />
     <div style="border-width:2px; border-color:green; border-
style:solid ;">
         <h1>Default SideBar</h1>
      </div>
      <br />
      <div style="border-width: 2px; border-color: black; border-
style: solid ;">
         <h1>Default Content</h1>
      </div>
      <br />
      <div style="border-width: 2px; border-color: red; border-style:
solid ;">
         <h1>Default Footer</h1>
      </div></body>
```

```
</html>
```

Για τον Header θα φτιάξουμε την header σελίδα header.xhtml και σε τεχνολογία jsf θα έχουμε.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:ui="http://java.sun.com/jsf/facelets">
  <body>
    <ui:composition>
      <h1>Default Header</h1>
    </ui:composition>
  </body>
</html>
```

Σε HTML θα έχουμε

```
<div style="border-width: 2px; border-color: green; border-style:
solid ;">
         <h1>Default Header</h1>
      </div>
```

Για τον Footer θα φτιάξουμε την footer.xhtml και σε τεχνολογία jsf θα έχουμε.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:ui="http://java.sun.com/jsf/facelets">
   <body>
      <ui:composition>
```

```
         <h1>Default Footer</h1>
      </ui:composition>
   </body>
</html>
```

Σε HTML θα έχουμε

```
<div style="border-width: 2px; border-color: red; border-style: solid
;">
         <h1>Default Footer</h1>
      </div>
```

Για SideBar θα φτιάξουμε την SideBar.xhtml και σε τεχνολογία jsf θα έχουμε.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:ui="http://java.sun.com/jsf/facelets">
   <body>
      <ui:composition>
         <h1>Default SideBar </h1>
      </ui:composition>
   </body>
</html>
```

Σε HTML θα έχουμε

```
<div style="border-width: 2px; border-color: green; border-style:
solid ;">
         <h1>Default SideBar</h1>
      </div>
```

Για commons θα φτιάξουμε την commons.xhtml και σε τεχνολογία jsf θα έχουμε.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:ui="http://java.sun.com/jsf/facelets">
   <body>
      <ui:composition>
         <h1>Default  Content  </h1>
      </ui:composition>
   </body>
</html>
```

Σε HTML θα έχουμε

```
<div style="border-width: 2px; border-color: black; border-style:
solid ;">
         <h1>Default Content</h1>
      </div>
```

θα φτιάξουμε και τις σελίδες commons όπου θα είναι το περιεχόμενο της σελίδας μας και εκεί μέσα θα ενθυλακώνονται τα υπόλοιπα κομμάτια τα οποία θα είναι μέρη της σελίδας .

Αναλυτικά.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

  xmlns:h="http://java.sun.com/jsf/html"

  xmlns:ui="http://java.sun.com/jsf/facelets">

  <h: head></h: head>

  <h:body>

    <div style="border-width: 2px; border-color: green; border-style: solid ;">
```

```
        <ui: insert name="header" >
          <ui: include src="/templates/header.xhtml" />
        </ui: insert>
      </div>
      <br/>
     <div style="border-width: 2px; border-color: green; border-style: solid ;">
        <ui: insert name="sidebar" >
          <ui: include src="/templates/sidebar.xhtml" />
        </ui: insert>
      </div>
      <br/>
      <div style="border-width: 2px; border-color: black; border-style: solid ;">
        <ui: insert name="content" >
          <ui: include src="/templates/contents.xhtml" />
        </ui: insert>
      </div>
      <br/>
      <div style="border-width: 2px; border-color: red; border-style: solid ;">
        <ui: insert name="footer" >
          <ui: include src="/templates/footer.xhtml" />
        </ui: insert>
      </div>
    </h:body>
</html>
```

Και μέσα σε αυτή θα γίνουν τα includes δηλαδή η τοποθέτηση των προηγουμένων σελίδων και σαν αποτέλεσμα θα έχουμε την σελίδα που παρουσιάσαμε πιο πριν.

Οπότε το αποτέλεσμα της σελίδας μας και σε τεχνολογία JSF είναι η σελίδα index.xhtml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:ui="http://java.sun.com/jsf/facelets">
   <h:head></h:head>
   <h:body>
      <div style="border-width:2px; border-color:green; border-style:solid;">
         <ui:insert name="header" >
            <ui:include src="/templates/header.xhtml" />
         </ui:insert>
      </div>
      <br/>
     <div style="border-width:2px; border-color:green; border-style:solid;">
         <ui:insert name="sidebar" >
            <ui:include src="/templates/sidebar.xhtml" />
         </ui:insert>
      </div>
      <br/>
      <div style="border-width:2px; border-color:black; border-style:solid;">
         <ui:insert name="content" >
            <ui:include src="/templates/contents.xhtml" />
         </ui:insert>
      </div>
      <br/>
      <div style="border-width:2px; border-color:red; border-style:solid;">
         <ui:insert name="footer" >
            <ui:include src="/templates/footer.xhtml" />
         </ui:insert>
      </div>
   </h:body>
</html>
```

Για χάρη ευκολίας της εφαρμογής και επειδή είναι στα πλαίσια διπλωματικής εργασίας θα παραμείνουμε λιτοί στην δημιουργία και θα γίνει αναφορά στον κώδικα στην τελική έκδοση. καθώς θα υπάρχει και το πέρασμα σε τεχνολογία υλοποίησης οθόνης JSF και η τελική έκδοση των οθονών.

Στο κάθε ένα από αυτά θα προσθέσουμε για την μορφοποίηση του css.

Χρήση CSS με JSF

Cascading Style Sheets (css) όπως αναφέρθηκε είναι μια απλή μορφή γλώσσας σχεδιασμού μορφοποίησης που μας δίνει την δυνατότητα να τυποποιήσουμε την εμφάνισης μιας ιστοσελίδας.

Με την χρήση του css μπορούμε να χειριστούμε το χρώμα, το στυλ των γραμμάτων, την μορφολογία την διαίρεση τις σελίδας σε κομμάτια το κενό μεταξύ παραγράφων το μέγεθος και τις φωτογραφίες που υπάρχουν καθώς και μια μεγάλη ποικιλία από άλλες δυνατότητες σε σχέση πάντα με την εμφάνιση .

Τα πλεονεκτήματα του css είναι

Κερδίζουμε χρόνο στην δημιουργία και μορφοποίηση τις σελίδας.

Δεν χρειάζεται να γράψουμε κώδικα HTML πλέον για την μορφοποίηση, έτσι χρησιμοποιούμε τον χρόνο που κερδίζουμε για τη δημιουργία τις σελίδας μιας και μπορούμε να το επαναχρησιμοποιήσουμε σε περισσότερες σελίδες σε μια εφαρμογή η και σε άλλες εφαρμογές μας .

Ευκολία στην συντήρηση και αλλαγή.

Μπορούμε εύκολα να μετατρέψουμε την μορφή το χρώμα μιας σελίδας καθώς να αλλάξουμε την θέση μιας φωτογραφίας να την μεγαλώσουμε χωρίς να πειράξουμε των HTML κώδικα που έχουμε δημιουργήσει .

Καλύτερη Εμφάνιση.

Δίνει καλύτερη εμφάνιση στην web σελίδα μας από ότι η HTML καθώς έχει μια πιο μεγάλη γκάμα ιδιοτήτων από την HTML και χωρίς μεγάλη προσπάθεια αλλά με πολλή φαντασία έχουμε το επιθυμητό αποτέλεσμα.

Πολυλειτουργικο .

Το Css μπορεί να χρησιμοποιηθεί σε πολλές συσκευές. Με την χρήση του μπορούμε να έχουμε εμφάνιση σε πάρα πολλές συσκευές όπως σελίδες web σελίδες σε android εφαρμογές φτιαγμένες για pda’s καθώς και εκτυπωτές .

Παγκόσμιο εργαλείο εμφάνισης σελίδων .

Από την W3C προτείνεται και ελέγχεται η χρήση του σε συνδυασμό με την χρήση σελίδων web έτσι ώστε να αποφεύγεται η χρήση γλώσσας HTML για την εμφάνιση.

Οπότε γίνεται χρήση τις HTML μόνο για την δόμηση τις σελίδας και τα υπόλοιπα τα αναλαμβάνει το css.

Πρέπει να αναφερθεί ότι κάνοντας χρήση του css έχουμε πιο γρήγορο φόρτωμα τις σελίδας μας από τον client έτσι μπορεί να αυξηθεί η κίνηση της σελίδας που υποστηρίζουμε παρακάμπτοντας τα προβλήματα της ταχύτητας.

Παράδειγμα css.

```
/* This is an external style sheet file */

h1, h2, h3 {
color: #36C;
font-weight: normal;
letter-spacing: .4em;
margin-bottom: 1em;
text-transform: lowercase;
}

.scroll{
        display:block;
        border: 1px solid red;
        padding:5px;
        margin-top:5px;
        width:300px;
        height:50px;
        overflow:scroll;
        }
.auto{
        display:block;
        border: 1px solid red;
        padding:5px;
        margin-top:5px;
        width:300px;
        height:50px;
        overflow:auto;
        }


/* end of style rules. */
```

Η χρήση του είναι πολύ απλή, έχουμε τους εξής τρόπους.

Ενσωμάτωση Css .

Το Css μπαίνει μέσα στην σελίδα και την μορφοποιεί. Με την χρήση του <style> μπορούμε να το βάλουμε μέσα στην σελίδα html που έχουμε και να ενεργήσει δυναμικά.

Παράδειγμα ενσωματώσεις css.

Κώδικας ενσωματώσεις css.

```
<head>
<style type="text/css" media="...">
Style Rules
.....
</style>
</head>
```

Εσωτερική εισαγωγή Css σε οποιοδήποτε στοιχείο τις σελίδας.

<element style="...style rules....">

αναλυτικά

<h1 style ="color:#36C;"> This is inline CSS </h1>

Και ο τελευταίος τρόπος και ο πιο χρησιμοποιημένος.

Εξωτερικά σε ένα άλλο αρχείο σε ένα άλλο μέρος που δεν έχει καμία σχέση με την σελίδα μας και κάνουμε αναφορά στην σελίδα που θα βρει την κωδικοποίηση css.

Έχει χρησιμοποιηθεί η μέθοδος αυτή μιας και είναι η πιο αξιόπιστη από όλες .

Παράδειγμα σελίδας με κωδικοποίηση css με αναφορά σε αρχείο.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Associating Styles with Documents</title>
<link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
<meta name="Keywords" content="CSS, Tutorials, Learning, Beginners, CSS, Basics, Advanced, Cascading Style Sheet, CSS2, Prope
<meta name="description" content="Associating Styles with Documents - CSS Tutorials for beginners to advanced developers - Le
<meta http-equiv="pragma" content="no-cache">
<meta http-equiv="cache-control" content="no-cache">



<link rel="stylesheet" type="text/css" href="css_inclusion.css" media="all">
</head>
<body>
<center>
<a name="top"></a>
<table class="main" border="0" cellpadding="2" cellspacing="0">
<tbody><tr>
<td class="content" align="left" valign="top">
<a href="http://www.tutorialspoint.com/index.htm"><img src="tp-logo.gif" alt="Tutorials Point" border="0"></a>
<br>
<br>
<div class="search">
```

Και η αναφορά γίνεται ως εξής .

```
<head>
<link type="text/css" href="css/css.css" rel="stylesheet">
</head>
```

Σε τεχνολογία jsf.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html">
    <h:head>
        <title>Index gDesign</title>
        <h:outputStylesheet name="css/jobs.css"/>
    </h:head>
    <h:body>
       <h1>
            <ui:insert name="title">Default Title</ui:insert>
        </h1>
        <p>
            <ui:insert name="body">Default Body</ui:insert>
        </p>
    </h:body>
</html>
```

Στην εφαρμογή μας έχουμε δημιουργήσει css για κάθε σελίδα ξεχωριστά.

Για την εφαρμογή μας θα χρησιμοποιήσουμε το css για να μορφοποιήσουμε την σελίδα μας κατάλληλα έτσι έχει γίνει επιλογή τις δυνατότητας και φορτώνει το css μόνο η σελίδα που το χρειάζεται.
Θα φτιαχτεί ένα πλαίσιο που θα ενσωματώνει τα άλλα πλαίσια και θα χωρίσουμε την σελίδα σε διαφορετικά μέρη.

Τα μέρη που θα δεχτούν την μορφοποίηση του css είναι όλες οι σελίδες που έχουμε στην εφαρμογή μας.
Τα αρχεία με το css είναι τα εξής και βρίσκονται στον φάκελο resources.

Css.css
cssLayout.css
cssStyle.css
default.css
rocanvas.css

Και στον συγκεκριμένο Header ανάλογα με την σελίδα θα εφαρμοστεί ο κανόνας για το css κάθε φορά που χρειάζεται.
Με βάση την επαναχρησιμοποίηση το συγκεκριμένο Header θα είναι για τις σελίδες που χρειάζονται την συγκεκριμένη μορφοποίηση και δεν θα χρειάζεται να το αλλάζουμε άλλα και να το τοποθετούμε στην καθεμία σελίδα ξεχωριστά .

Θα έχουμε τουλάχιστον 3 διαφορετικούς Header για να έχουμε τις διάφορες επιλογές που θέλουμε .

Η σελίδες θα είναι

- Η Commons όπου θα έχει όλη την σελίδα μέσα της θα είναι δηλαδή ο container.

- Contents όπου θα είναι και η κύρια σελίδα το εσωτερικό της σελίδας μας.

- Sidebar όπου θα είναι και η μπάρα που θα έχει τα εργαλεία για σχεδίαση και τα εργαλεία για τις λειτουργίες στον λογαριασμό του χρήστη.

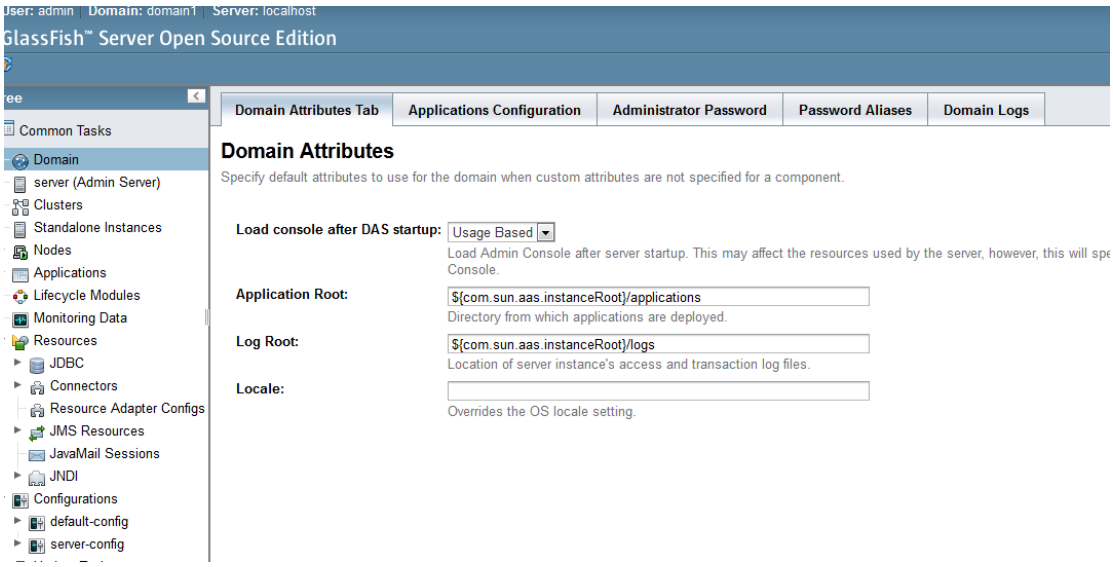
- Footer όπου και θα είναι για την εισαγωγή στοιχείων επικοινωνίας και τα στοιχεία του δημιουργού.

- Δηλαδή όλη η σελίδα θα διαμορφωθεί ανάλογα με την κατάσταση που βρίσκεται εκείνη την στιγμή η εφαρμογή.

Containers

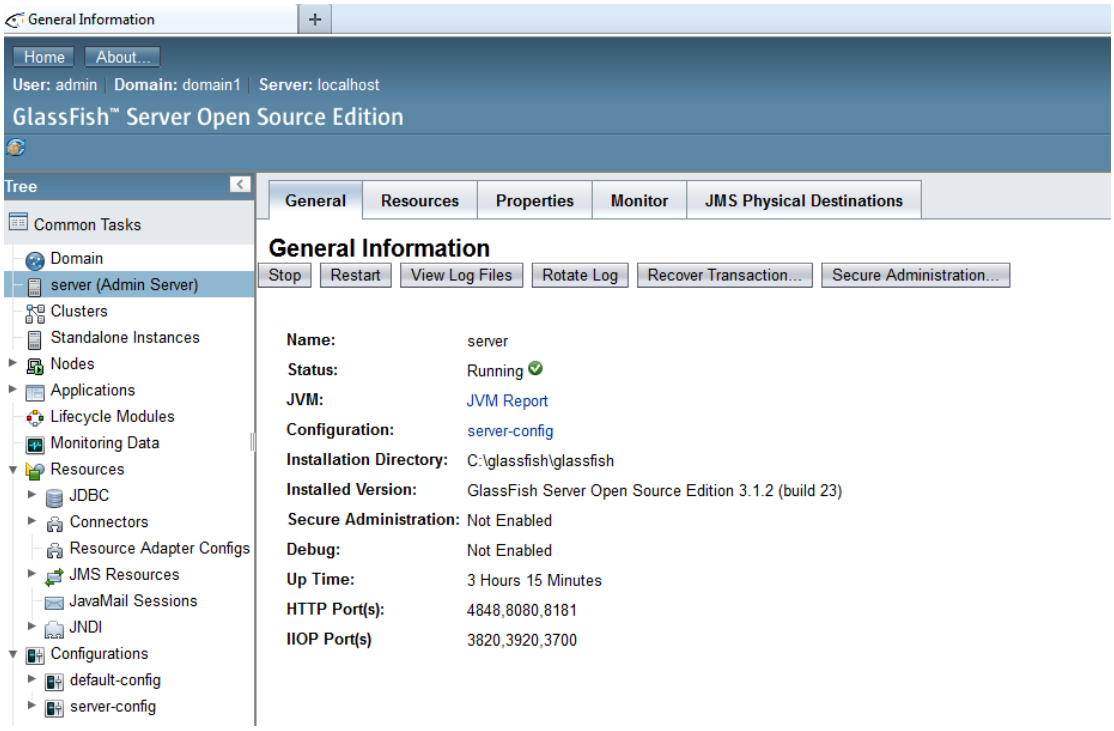
Glassfish Server Open Source Edition

Για να τρέξει η εφαρμογή πρέπει να υπάρχει κάτι που θα την δεχτεί και θα μπορέσει να την βγάλει στον αέρα έτσι ώστε να την χρησιμοποιήσουν οι χρήστες στο διαδίκτυο.



Χρησιμοποιούμε τον container Glassfish για να ανεβάσουμε την εφαρμογή και να την εκτελέσουμε. Διανέμεται δωρεάν από την Oracle και υπάρχουν αρκετές δωρεάν εκδόσεις.

Στην εφαρμογή θα γίνει χρήση του GlassFish Server 3.1
Έχουμε την δυνατότητα να δούμε την κατάσταση του server μέσα από το μενού που μας δίνει το admin panel του server.



Στις δυνατότητες που έχουμε είναι ότι μπορούμε να δημιουργήσουμε κάποιον εικονικό Host, να επεξεργαστούμε τις εφαρμογές που θα τρέχουμε στον server, να βρούμε τυχόν διαρροές που έχει κάποια εφαρμογή που τρέχει στον server .
Στα μειονεκτήματα του μπορεί να θεωρηθεί ότι δεν υπάρχουν πολλές εφαρμογές που να τρέχουν σε java Web 6 αλλά έχει πάρα πολλές δυνατότητες για το μέλλον μιας και έχει τεχνολογίες για την χρήση java 6 και java 7 .

Μετά την εγκατάσταση και την δημιουργία χρηστών θα μπορούμε να κάνουμε deploy την εφαρμογή μας.

Login Authentication

Δημιουργώντας τις σελίδες που θα έχουμε για την χρήση της εφαρμογής πρέπει να έχουμε μια αρχική σελίδα για τον χρήστη όπου θα καλωσορίζει τον ενδιαφερόμενο για την εφαρμογή.
Θα έχει την δυνατότητα να κάνει εγγραφή στην εφαρμογή εάν δεν είναι ήδη εγγεγραμμένος χρήστης και να την δοκιμάσει.

Οπότε θα πρέπει να δημιουργήσουμε και μια σελίδα εγγραφής (Registration Page) δηλαδή θα πρέπει να εισάγουμε τύπο ασφάλειας στην εφαρμογή μας έτσι ώστε να απορρίψουμε την πρόσβαση σε άτομα που δεν είναι εγγεγραμμένα.
Θα έχουμε 2 ειδών χρήστες τους admin και user οπότε θα πρέπει να ετοιμάσουμε την εφαρμογή να μπορεί να τους δεχτεί.
Πέρα του απλού login θα πρέπει να δώσουμε και τα ανάλογα δικαιώματα στους χρήστες έτσι ώστε να μπορούμε να ξέρουμε τι μπορεί να κάνει ο admin και τι μπορεί να κάνει ο User.
Έτσι θα αναθέσουμε ρόλους στον κάθε χρήστη.
Θα μπορούμε να δώσουμε την δυνατότητα στον χρήστη να κάνει login στον λογαριασμό του και να επιλέξουμε τι θα μπορεί να δει, θα πρέπει επίσης να δώσουμε τη δυνατότητα στον admin να μπορεί η όχι να δει και τον λογαριασμό του χρήστη.
Ο χρήστης user θα κάνει login στον λογαριασμό του και θα έχει κάποια συγκεκριμένα δικαιώματα δεν θα μπορεί να δει την οθόνη με τις λειτουργίες του admin ούτε και να τις χρησιμοποιήσει.
Ο admin θα πρέπει να μπορεί να κάνει login από την εφαρμογή να χρησιμοποιήσει τα χαρακτηριστικά τις εφαρμογής και να τα τροποποίηση και να δει την λειτουργία του λογαριασμού του χρήστη σε περίπτωση που έχει κάποιο πρόβλημα ο χρήστης.

Η εφαρμογή δημιουργείται με τον σκοπό να χρησιμοποιηθεί ως ένα εργαλείο που μελλοντικά θα μπορεί να προσφέρει στον χρήστη ευχρηστία και σημαντική βελτίωση στον χρόνο εργασίας του. Πρέπει να καλύπτει τον χρήστη σε αρκετά σημεία όπως απόδοση ευχρηστία και ασφάλεια.
Για να μπορούμε να έχουμε ασφάλεια θα πρέπει να μπορούμε να εγγυηθούμε στον χρήστη την **εμπιστευτικότητα** την **ακεραιότητα** και την **διαθεσιμότητα** τις εφαρμογής.

Όταν προσπαθούμε να δούμε προστατευόμενο περιεχόμενο σε μια σελίδα ο Container ενεργοποιεί τον μηχανισμό πιστοποίησης που έχει δημιουργηθεί για την εφαρμογή εμποδίζοντας έτσι την παράνομη είσοδο και την κοινή χρήση στοιχείων εργασίας κάποιου χρήστη από κάποιους άλλους.

Για να γίνει αυτό χρειάζεται να δημιουργήσουμε τις κατάλληλες συνθήκες για τους χρήστες καθώς και τα δικαιώματα που θα υπάρχουν για τον καθένα. Η ασφάλεια μιας εφαρμογής είναι από τα σημαντικότερα σημεία που θα χρειαστεί να υλοποιηθεί για την εφαρμογή μιας και θα μπορεί ο χρήστης να αγοράσει και να αποθηκεύσει κάποια σχέδια.

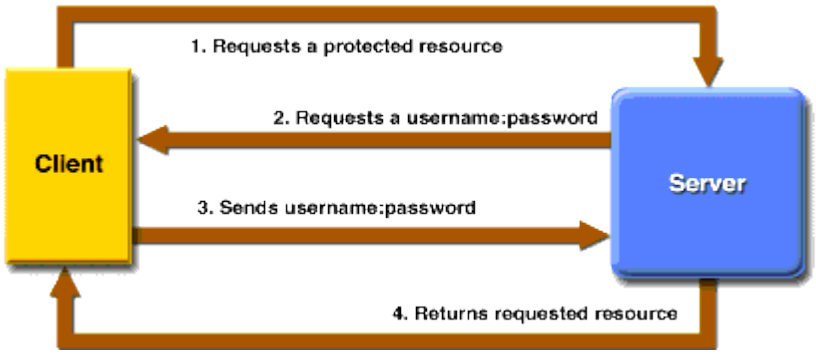
Είμαστε βέβαιοι ότι ο χρήστης δεν θα θέλει να μοιραστεί το νούμερο τις πιστωτικής τους καθώς και τα προσωπικά του στοιχεία με κάποιους άλλους. Οπότε θα δημιουργηθεί τρόπος πιστοποίησης του χρήστη και θα έχει συγκεκριμένα δικαιώματα .
Όλοι οι χρήστες θα χρειάζονται πιστοποίηση για την ασφαλή είσοδο στην εφαρμογή, με την διαφορά στα δικαιώματα που θα έχουν ανάλογα με τις απαιτήσεις τους .

Τρόποι πιστοποίησης είναι η εξής:

- Http Basic Authentication (Http Βασική πιστοποίηση)
- Form-based login authentication (πιστοποίηση με φόρμα)
- Client certificate authentication (πιστοποίηση με πιστοποιητικό)
- Mutual authentication (κοινή πιστοποίηση)
- Digest authentication (πιστοποίηση με κωδικοποίηση)

Http Basic Authentication (Http Βασική πιστοποίηση)

1 Ο Client ζητά πρόσβαση σε μια προστατευμένη σελίδα.
2 Ο Web Server επιστρέφει ένα μήνυμα που ζητά το user name και το password.
3 Ο Client στέλνει τα στοιχεία του στον server.
4 Ο Server επιβεβαιώνει τα στοιχεία και επιστρέφει το προστατευόμενο περιεχόμενο .

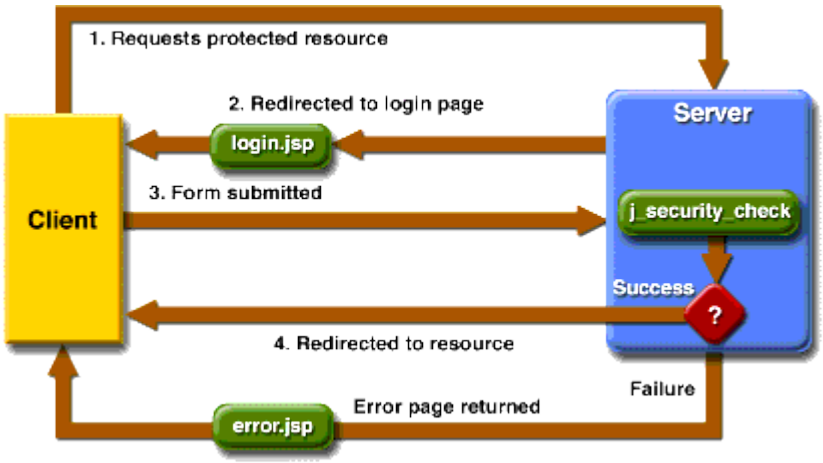


Form-based login authentication (πιστοποίηση με φόρμα)

1 Ο Client ζητά πρόσβαση σε μια προστατευμένη σελίδα.
2 Εάν ο χρήστης δεν είναι πιστοποιημένος ο Server τον στέλνει στην φόρμα πιστοποίησης.

3 Ο Client στέλνει τα στοιχεία του στον server.

4 Εάν η πιστοποίηση είναι επιτυχής ο Server στέλνει τον client στο περιεχόμενο αλλιώς του βγάζει μήνυμα λάθους.



Client certificate authentication (πιστοποίηση με πιστοποιητικό)

Είναι μια πιο ασφαλές και εξελιγμένη μέθοδος πιστοποίησης σε σχέση με τις προηγούμενες. Χρησιμοποιεί το SSL στο Http που έχουν την πιστοποίηση του client και του Server με χρήση public key certificate.

Το Secure Socket Layer SSL προσφέρει κρυπτογράφηση δεδομένων ακεραιότητα μηνυμάτων καθώς και την client πιστοποίηση για TCP/IP συνδέσεις .

Mutual authentication (κοινή πιστοποίηση)

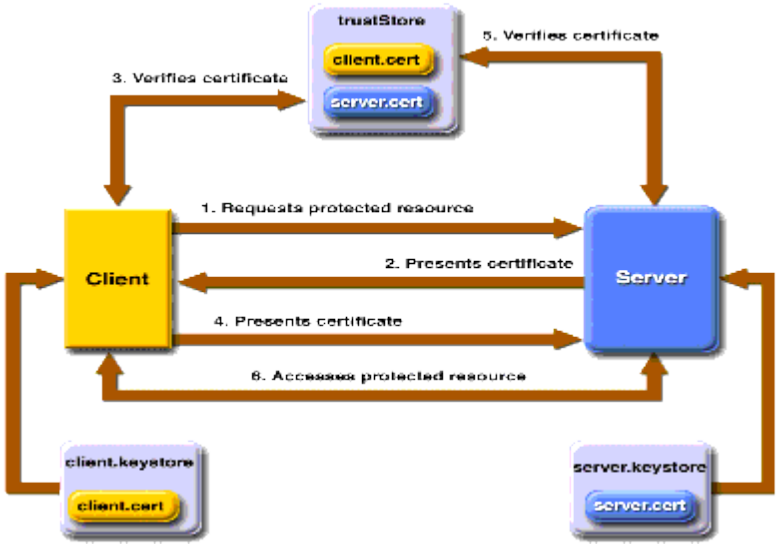
Με την κοινή πιστοποίηση ο Server και ο Client πιστοποιούν ο ένας τον άλλο.

Οι τρόποι που υπάρχουν για την κοινή πιστοποίηση.

Βάση κάποιου πιστοποιητικού.

Username και password με βάση κάποια κοινή πιστοποίηση.

1 Ο Client ζητά πρόσβαση σε μια προστατευμένη σελίδα.
2 Ο Server παρουσιάζει στον client το πιστοποιητικό του .
3 Ο Client πιστοποιεί το πιστοποιητικό του Server .
4 Εάν έχουμε επιτυχία ο client στέλνει το username και το password στον server που πιστοποιεί τον client.
4 Εάν η πιστοποίηση είναι επιτυχής ο Server παραχωρεί στον client πρόσβαση στο περιεχόμενο.



Digest authentication (πιστοποίηση με κωδικοποίηση)

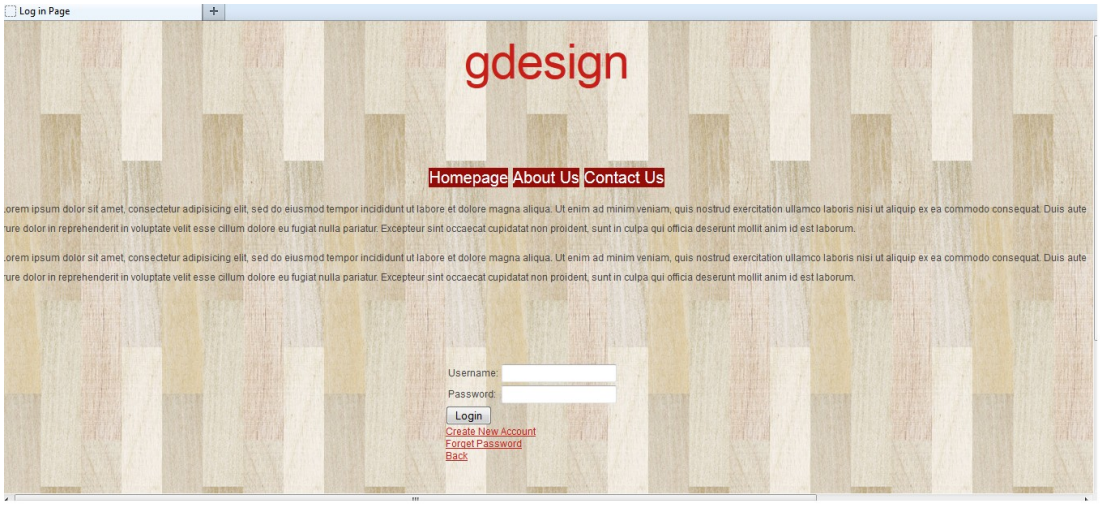
Όπως και η Http Basic Authentication (Http Βασική πιστοποίηση) έτσι και το Digest Authentication λειτουργεί με βάση το username και το password με την διαφορά ότι κρυπτογραφεί τα στοιχεία με ένα πιο ασφαλές σύστημα από ότι με βάση την 64bit κωδικοποίηση που χρησιμοποιείται στην Http Βασική πιστοποίηση.

Η Digest authentication (πιστοποίηση με κωδικοποίηση) δεν είναι τόσο δημοφιλής.

Θεωρώντας τους τρόπους που μπορούμε να έχουμε ένα login και αξιολογώντας την περίπτωση που έχουμε που είναι ανάλογη με την ασφάλεια που θέλουμε να δώσουμε στην εφαρμογή μας χρησιμοποιούμε την λειτουργία login με την χρήση φόρμας.

Η Οθόνη welcome

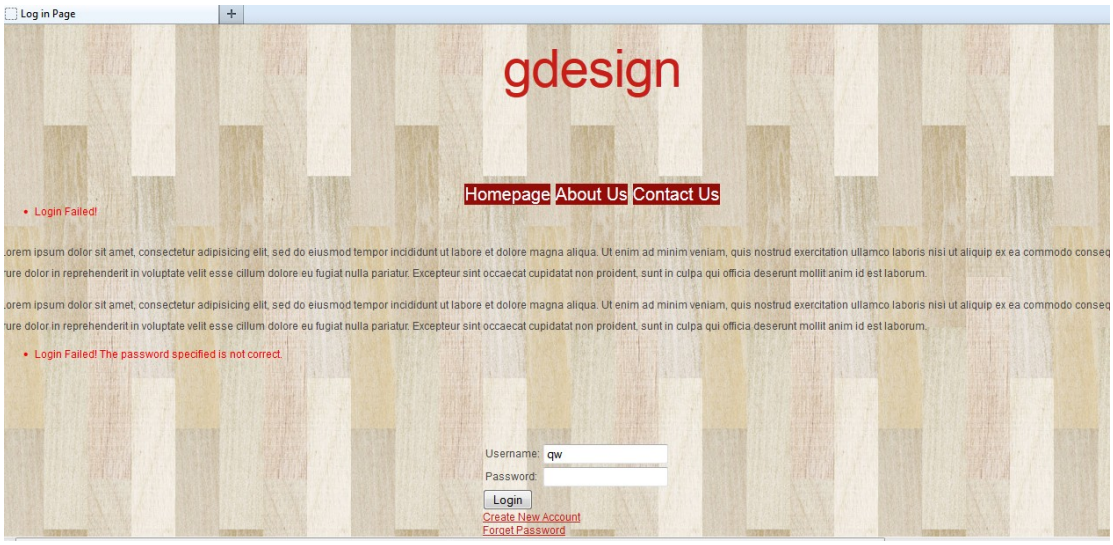
Η οθόνη Login Register



Η οθόνη About Us

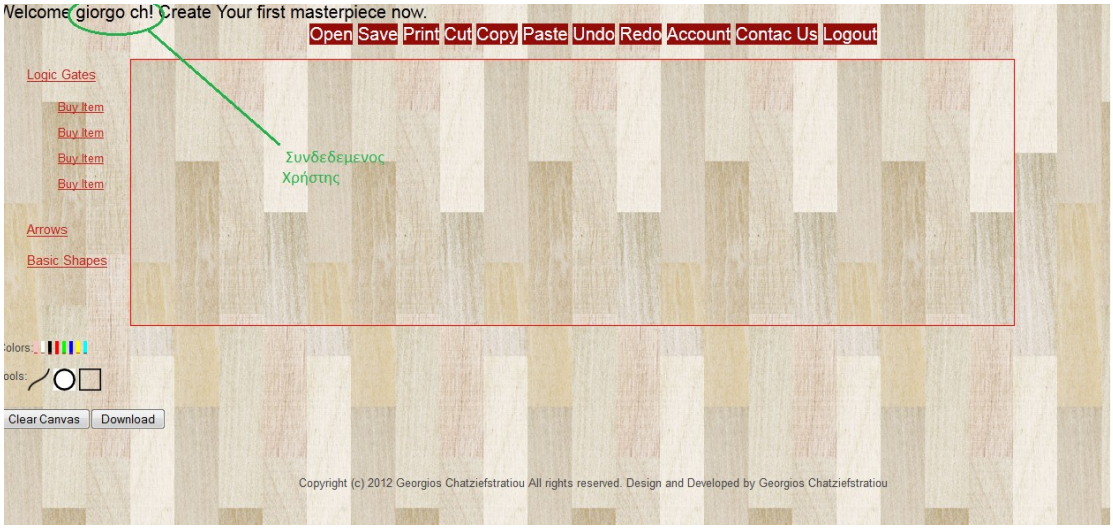
Η οθόνη Contact Us

Η οθόνη όταν έγινε λανθασμένο login



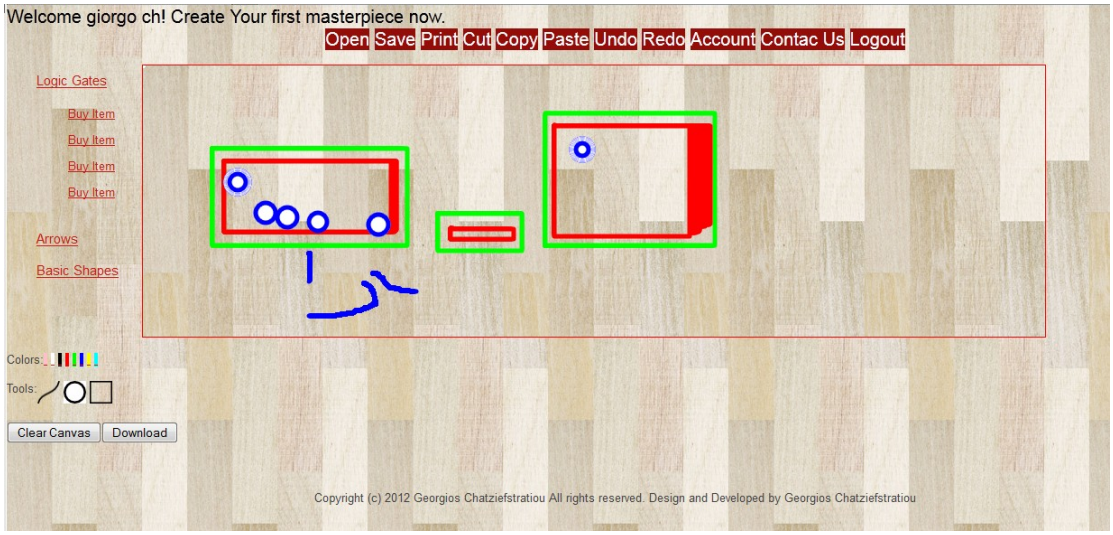
Η οθόνη με ένα χρήστη που έχει κάνει login Και δημιουργώντας ένα σχέδιο

Έχει την δυνατότητα να βλέπει τον συνδεδεμένο χρήστη

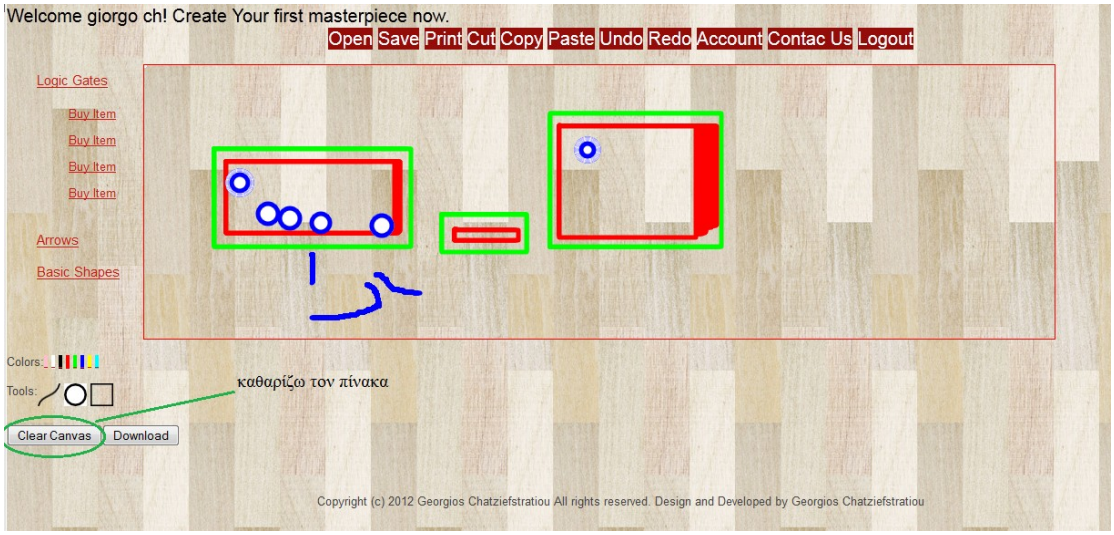


Υπάρχει η δυνατότητα δημιουργίας ελεύθερου σχεδίου

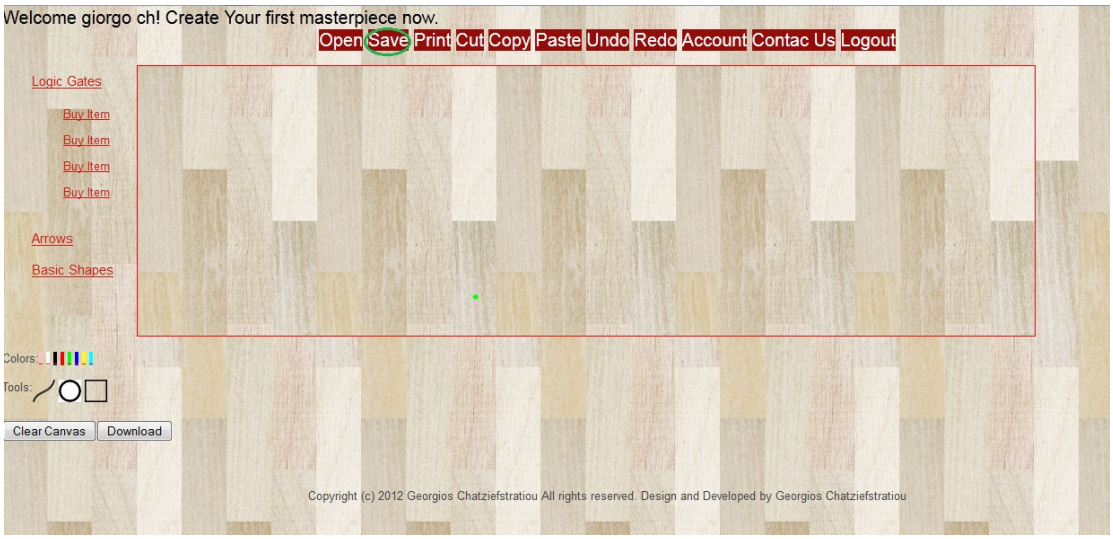
Αφού έχω διαλέξει την γραμμή που δίνεται από την αρχική έκδοση και το χρώμα πράσινο μπορώ να σχεδιάσω μια εικόνα



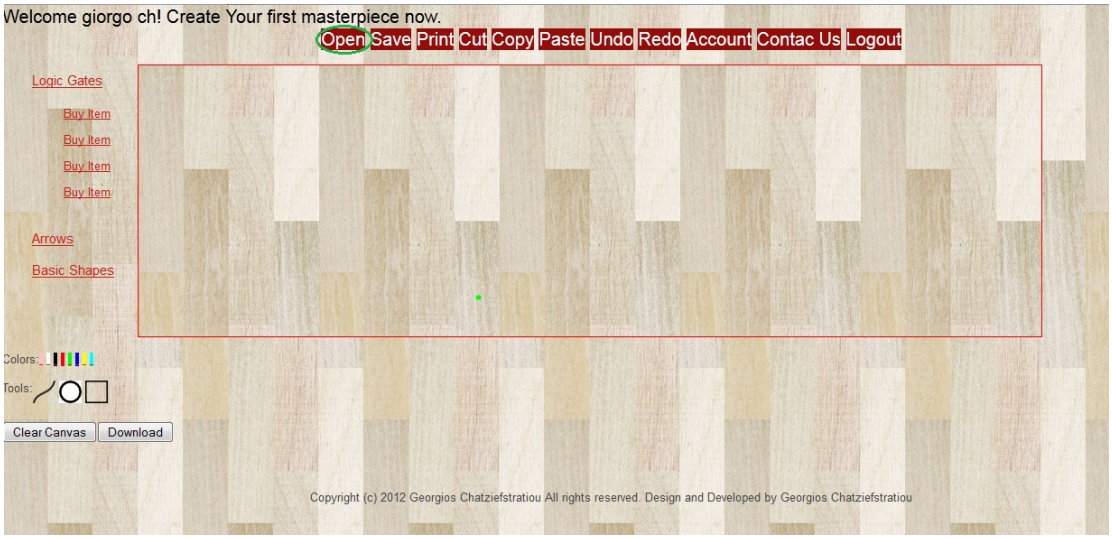
Με το κουμπί clear canvas καθαρίζω τον πίνακα μου



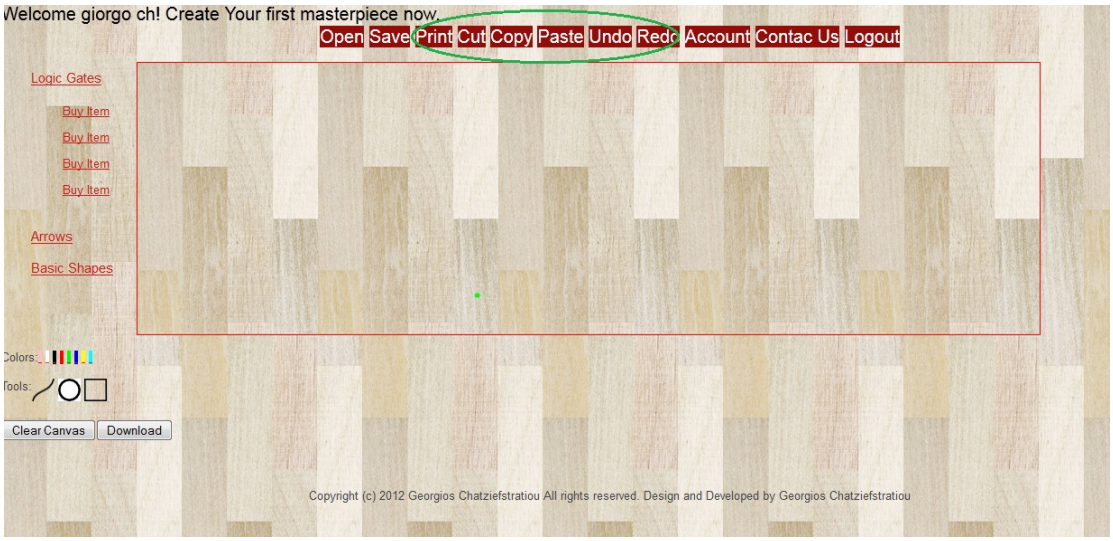
Στην πλήρη εφαρμογή θα έχουμε τη δυνατότητα να αποθηκεύσουμε στην βάση το σχέδιο μου με το πλήκτρο save



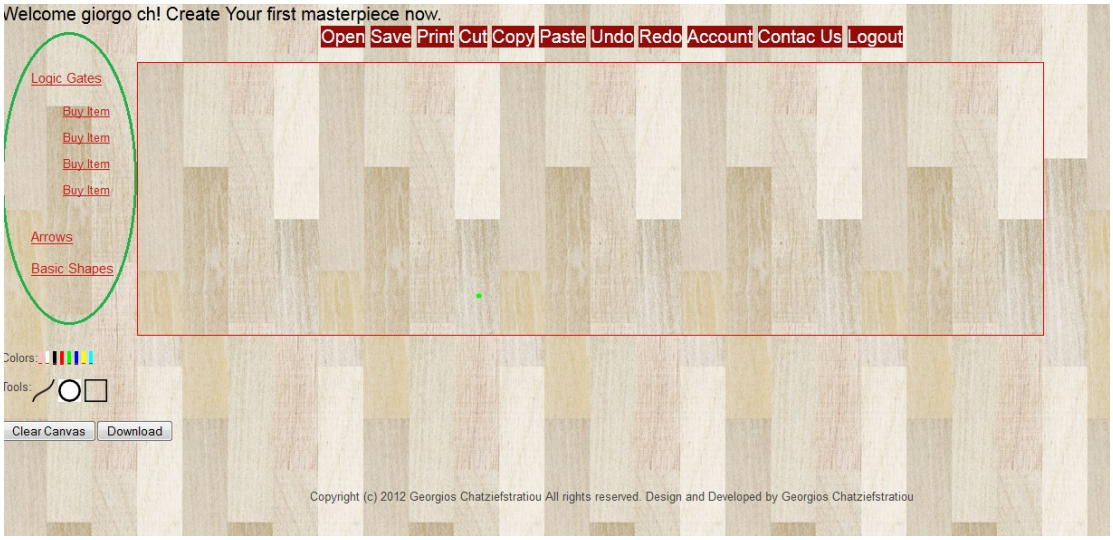
Να ανοίξουμε κάποιο αρχείο από τον υπολογιστή



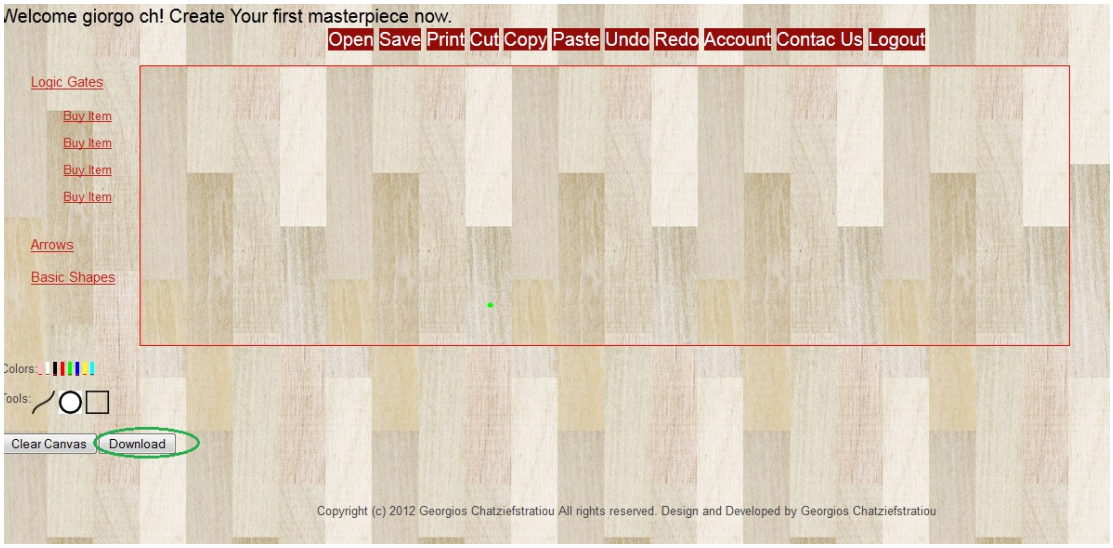
Να εκτυπώσουμε αρχείο καθώς και επικόλληση, αποκοπή, αναίρεση, επανάληψη



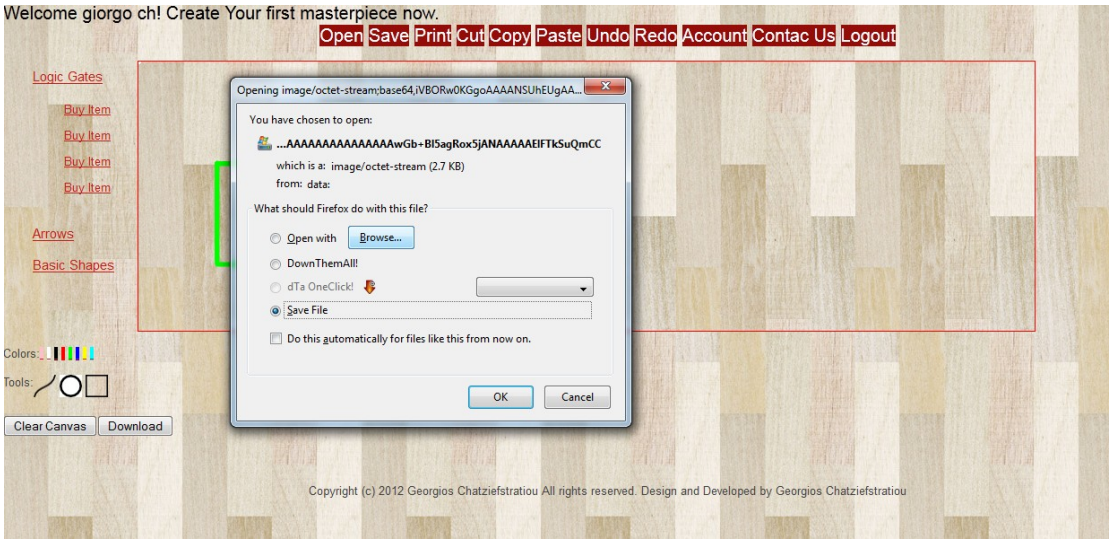
Μπορούμε να αγοράσουμε αντικείμενα για να τα χρησιμοποιήσουμε



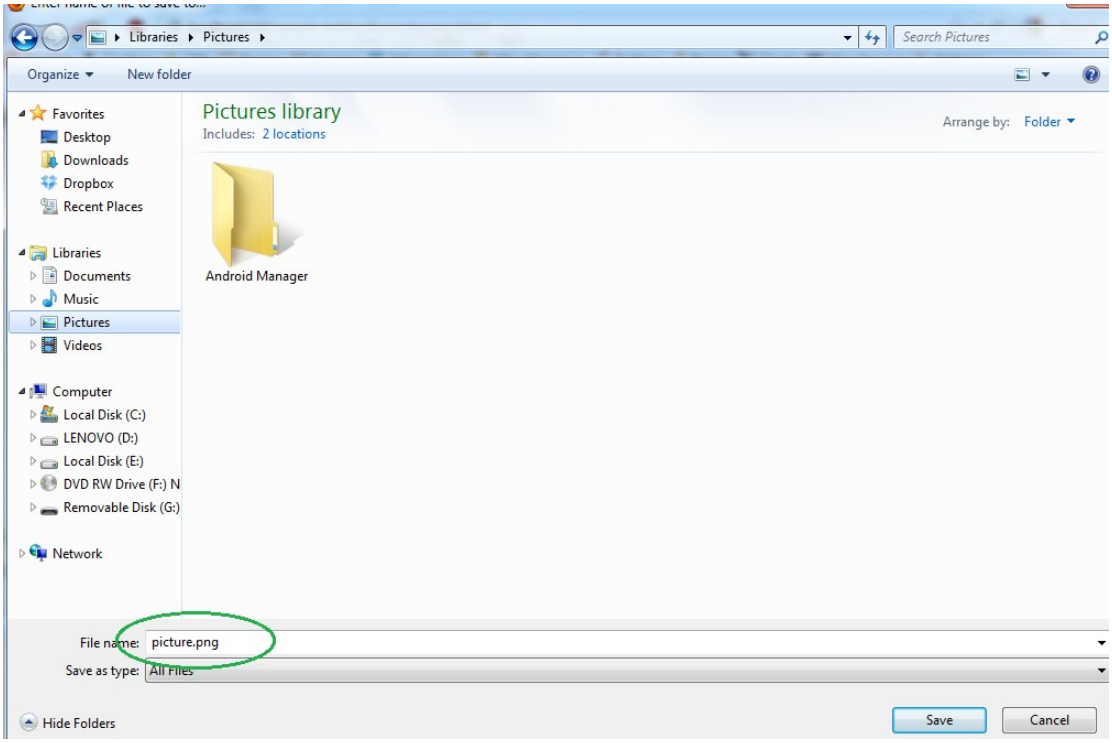
Η αρχική έκδοση (Demo) μας δίνει την δυνατότητα να κατεβάσουμε το σχέδιο και να το αποθηκεύσουμε στον υπολογιστή



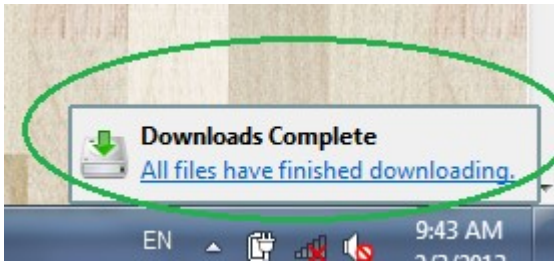
Πατώντας εμφανίζεται πλαίσιο διαλόγου αποθήκευσης



Διαλέγουμε το μέρος και δίνουμε το όνομα που θέλουμε ακολουθώντας την κατάληξη του αρχείου π.χ. θέλω να είναι picture.png

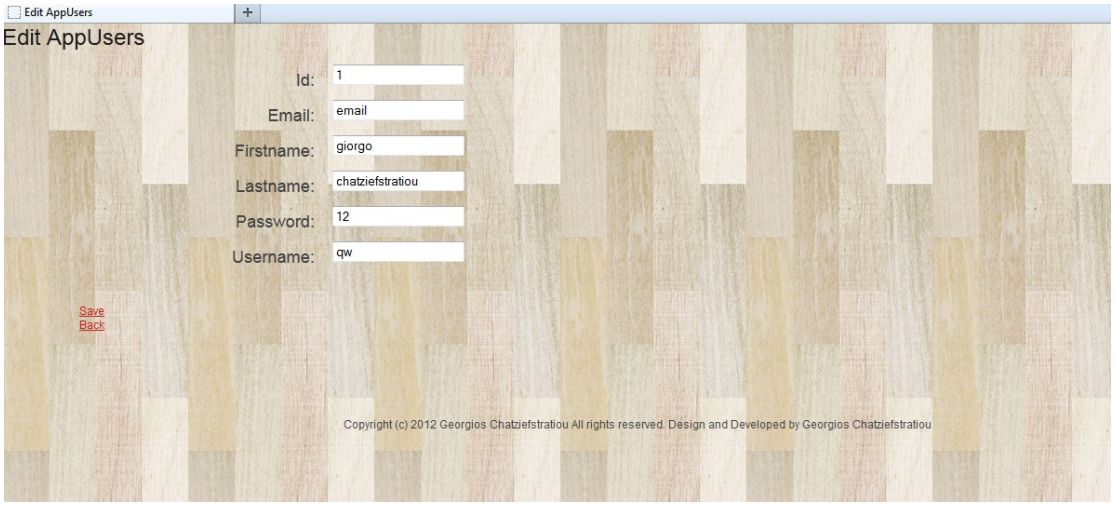


Και με το save αποθηκεύεται στον υπολογιστή.



Μπορούμε να δούμε και να επεξεργαστούμε τον λογαριασμό του χρήστη

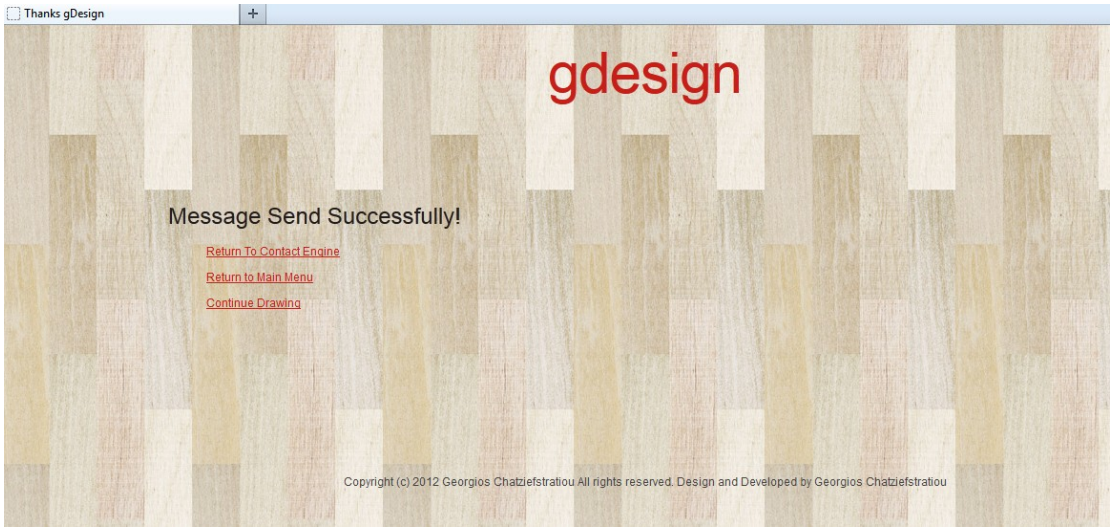
Και μεταφερόμαστε στον λογαριασμό του χρήστη



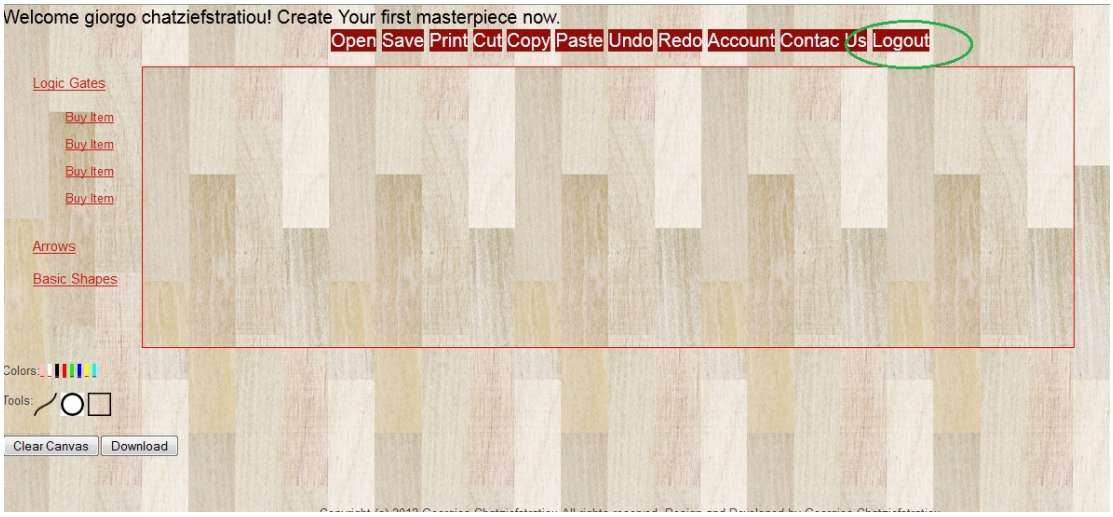
Έχουμε την δυνατότητα να επιστρέψουμε πίσω στο σχεδιασμό ή να κάνουμε κάποια αλλαγή και να την αποθηκεύσουμε στην βάση .

Μπορούμε να στείλουμε κάποια μηνύματα μέσω της φόρμας επικοινωνίας που υπάρχει

Το όνομα και τα υπόλοιπα στοιχεία τα παίρνει η φόρμα έτοιμα μιας και έχουμε συνδεθεί έχουμε πιστοποιηθεί και τα τραβά από την βάση με Session έτσι θα στείλουμε μήνυμα στον δημιουργό της εφαρμογής. Με την αποστολή μηνύματος θα μας εμφανιστεί μήνυμα επιτυχούς αποστολής μηνύματος και οι επιλογές για επιστροφή στα μηνύματα ή στο μενού ή στο σχεδιασμό. Πρέπει να συνδεθεί η εφαρμογή με ένα έγκυρο email έτσι ώστε να γίνει δυνατή η αποστολή του μηνύματος στο δημιουργό της εφαρμογής .



Τέλος όταν τελειώσουμε μπορούμε να κάνουμε log out και να τελειώσουμε την σύνοδο (Session) που έχουμε αρχίσει και να επιστρέψουμε στην αρχική οθόνη.

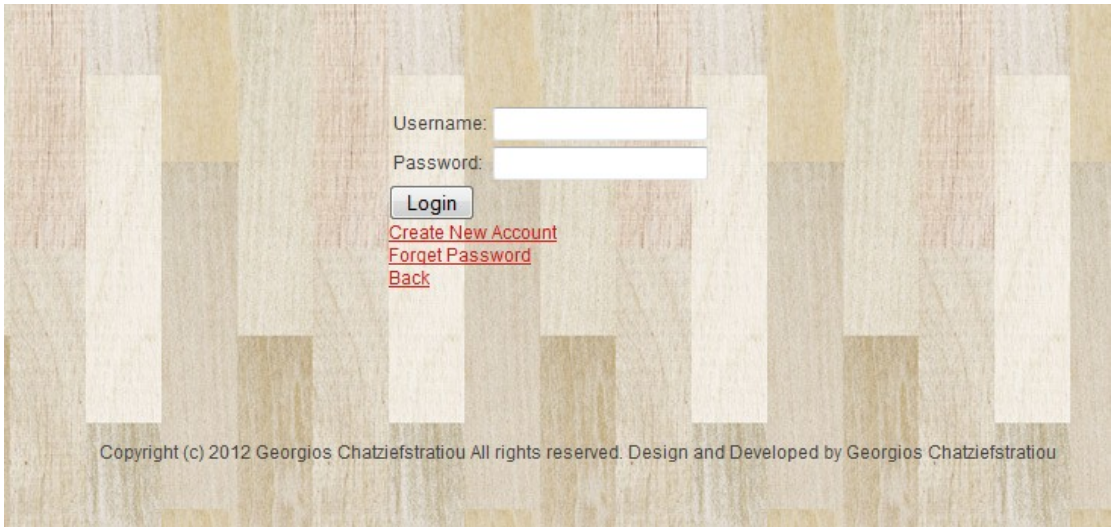


Οθόνη Login Register

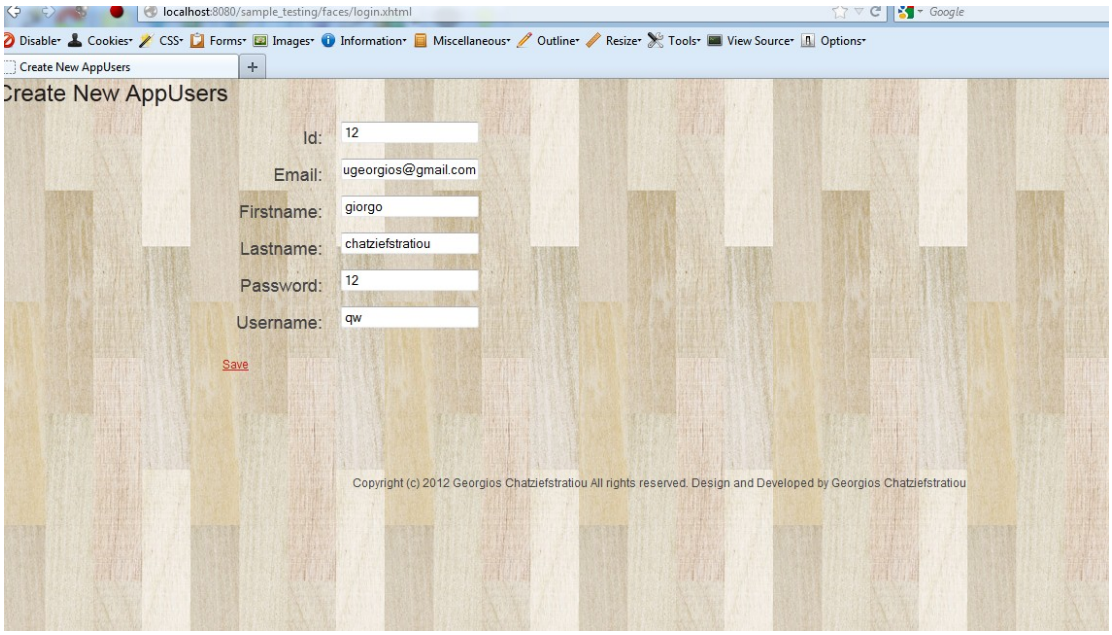
Στην οθόνη αυτή μπορούμε να κάνουμε Login στην εφαρμογή και θα γίνει αυθεντικοποιηση του Username και του password δημιουργώντας ένα Session με το όποιο θα μπορεί ο χρήστης να σχεδιάσει.

Εάν δεν έχουμε λογαριασμό τότε έχουμε την ευκαιρία να δημιουργήσουμε ένα έτσι ώστε να χρησιμοποιήσουμε την εφαρμογή.

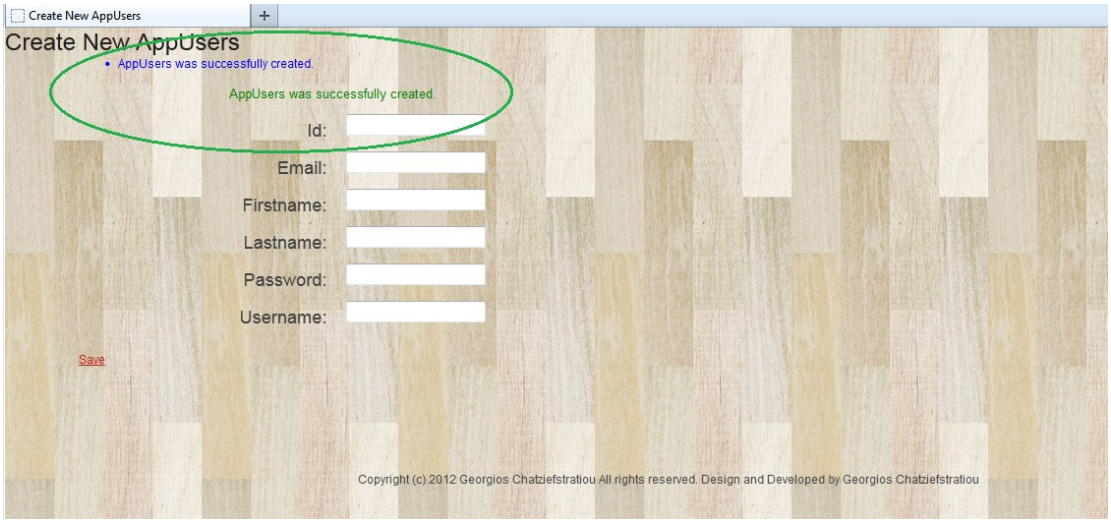
Στην ίδια οθόνη έχουμε την επιλογή να πατήσουμε το link Create New User Account και θα μεταφερθούμε στην δημιουργία νέου λογαριασμού .



Μεταφερόμαστε στην οθόνη που μπορούμε να δώσουμε τα στοιχεία μας έτσι ώστε να γίνει η εγγραφή μας στην εφαρμογή.

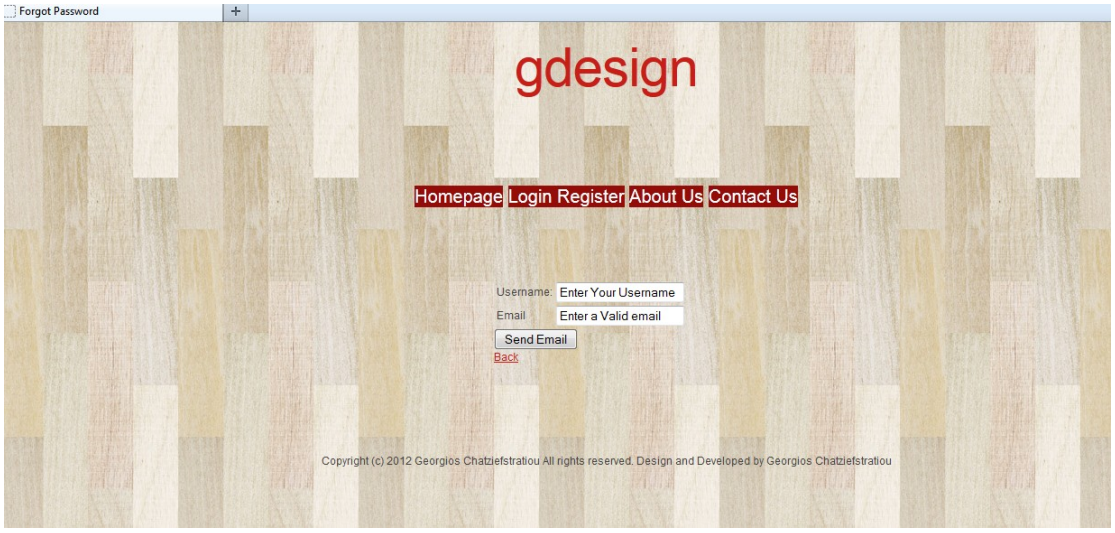


Η εφαρμογή θα μας ενημέρωση για την επιτυχή δημιουργία χρήστη με μήνυμα στην οθόνη

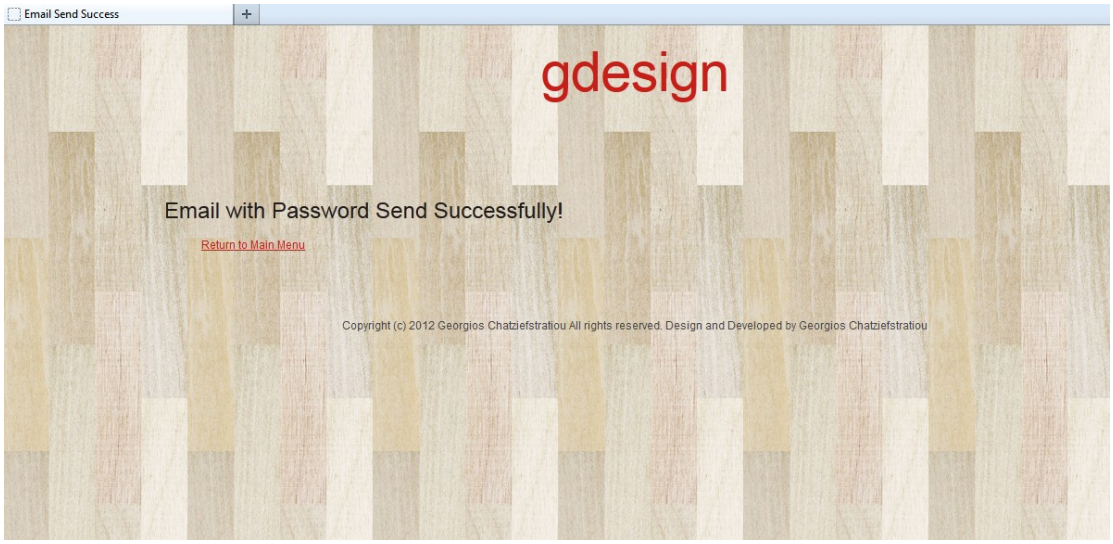


Μετά μπορούμε να επιστρέψουμε στην αρχική οθόνη login και να συνδεθούμε με τα στοιχεία που μόλις δώσαμε .

Σε περίπτωση που έχουμε ξεχάσει το password μπορούμε να χρησιμοποιήσουμε την φόρμα που θα μας στείλει τον κωδικό μας



Όταν πατήσουμε το πλήκτρο send mail έχουμε την οθόνη επιβεβαίωσης αποστολής



Πρέπει να σημειωθεί ότι για να γίνει αποστολή του password στο email που έχουμε δώσει χρειάζεται να έχει γίνει σύνδεση τις εφαρμογής με ένα έγκυρο email για να σταλεί το email.

Κεφάλαιο 9.Βάση δεδομένων

Ο κύριος λόγος που έχουν αναπτυχθεί οι βάσεις δεδομένων είναι για την αποθήκευση και εκμετάλλευση δεδομένων. Θα μπορούσαμε να έχουμε στην θέση τους αρχείο με απλές εγγραφές αλλά αυτό θα χρειαζόταν κόπο και χρόνο για την εύρεση μιας εγγραφής ανάμεσα σε χιλιάδες.

Αρκεί να φανταστούμε ότι σε μια σχετικά μεγάλη βάση δεδομένων που θα συντηρεί μια τράπεζα θα έχει τουλάχιστον 1000 άτομα με στοιχεία, μπορούμε να δούμε πόσο δύσκολο θα ήταν για την εύρεση ενός ατόμου μέσα στα 1000 όταν αυτό θα ήταν γραμμένο σε ένα έγγραφο.

Έτσι η χρήση βάσης δεδομένων έγινε θεσμός από την δεκαετία του 70’ και αναπτύχθηκαν με ταχύτατο τρόπο.

Από τα έγγραφα περάσαμε σε απλές βάσεις δεδομένων οι οποίες σίγουρα συνέβαλαν στην αντιμετώπιση των προβλημάτων ταχύτητας και κόστους αλλά δημιούργησαν νέα προβλήματα καθαρά χρήσης και εισαγωγής αλλά και επεξεργασίας δεδομένων. Μια καλή τακτική είναι επίσης να γίνεται χρήση παραπάνω από μιας βάσης δεδομένων έτσι ώστε να υπάρχει ταχύτητα στην εφαρμογή. Χωρίζουμε την εφαρμογή σε κομμάτια για να έχουμε καλύτερη επικοινωνία και ταχύτητα αλλά και ευκολία στην κατασκευή της εφαρμογής μας.

Μπορούμε να έχουμε την βάση την οποία θα γίνεται η σύνδεση με τον χρήστη και θα μπορεί να είναι μια σχεσιακή βάση δεδομένων (postgresSQL).

Θα υπάρχουν πινάκες που έχουν σχέση με την κίνηση του λογαριασμού και τις λειτουργίες του. Έχουμε βγάλει κλάσεις που θα υλοποιήσουμε σε κώδικα και θα χρησιμοποιηθούν για την αποθήκευση και χρήση της εφαρμογής. Έχουμε κάνει ανάλυση των απαιτήσεων στο έγγραφο προδιαγραφών και έχουμε φτάσει σε ένα σημείο να έχουμε το μοντέλο που θα υλοποιήσουμε σε σχέδιο.

Μια διαδεδομένη τεχνική για την δημιουργία της βάσης είναι η χαρτογράφηση των κλάσεων με κάποιο ORM εργαλείο σε sql και σε δημιουργία βάσης, μια άλλη τεχνική είναι ακριβώς το αντίστροφο δηλαδή η δημιουργία της βάσης και η σύνδεση της με κώδικα. Εδω θα γίνει χρήση του JPA Java Persistence API.

Με τον ένα τρόπο η τον άλλο έχουμε σύνδεση βάσης δεδομένων, λειτουργικότητας και αποθήκευσης αλλά και σύνδεση με κώδικα. Οπότε το αποτέλεσμα είναι σχεδόν το ίδιο με την μόνη διαφορά τις ασφάλειας σε SQL Injection .

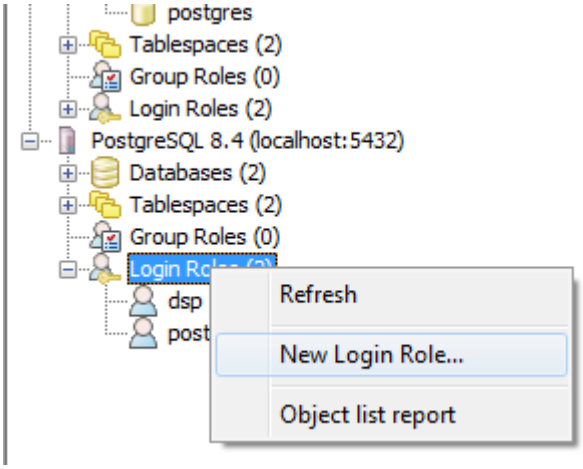
Η σχέσεις που θα δημιουργηθούν οι ιδιότητες και όλα τα απαιτούμενα στοιχεία που χρειάζονται η κλάσεις η οι πίνακες θα είναι χαρτογραφησιμες είτε από sql σε κάποια

γλώσσα προγραμματισμού είτε από κάποια γλώσσα προγραμματισμού σε sql και θα είναι έτοιμες για χρήση.
Η βάση δεδομένων θα δημιουργηθεί σε postgres όπου διατίθεται ελεύθερη.

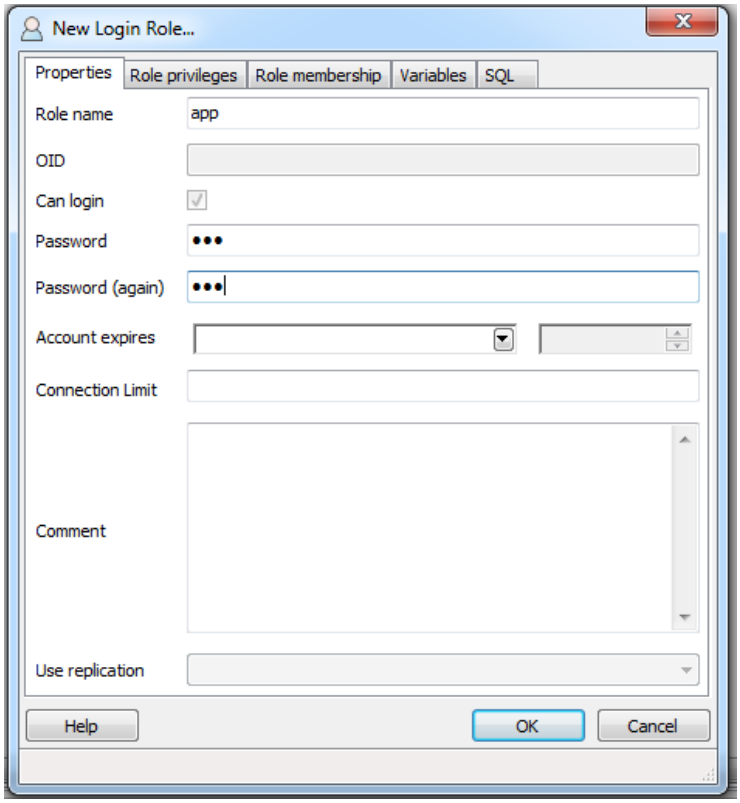
Το παρόν θα είναι ένα μικρό δείγμα υλοποίησης τις βάσης μιας και η βάση σε πλήρη ανάπτυξη μπορεί να έχει πάρα πολλούς πίνακες .

Δημιουργία βάσης δεδομένων εφαρμογής

Και η βάση δεδομένων θα είναι η εξής
Πρώτα θα δημιουργήσουμε κάποιο Role για τη βάση που θα δημιουργηθεί

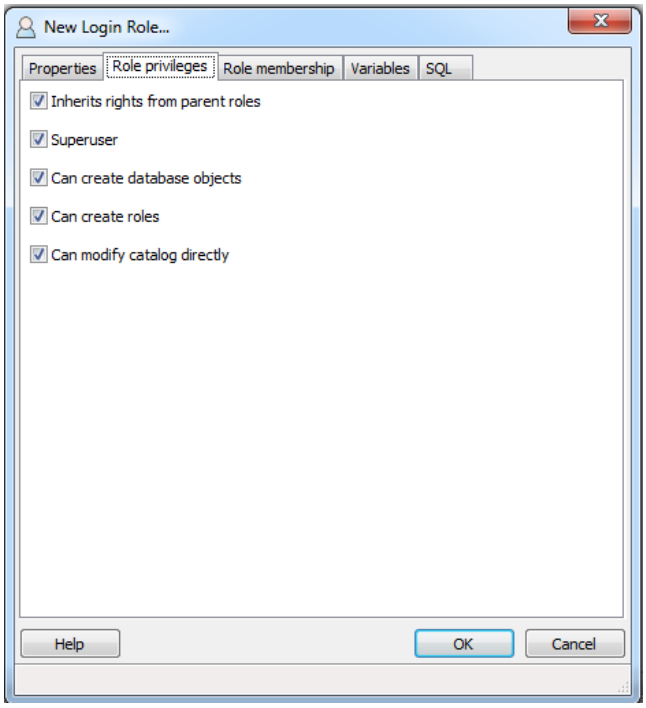


Θα δημιουργηθεί ο χρήστης που θα έχει τον έλεγχο της βάσης οπότε

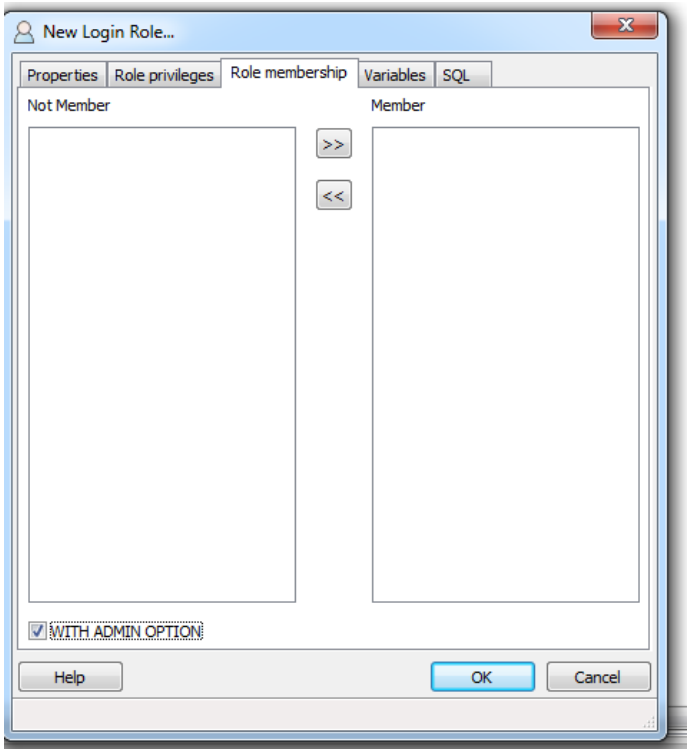


Θα του δώσουμε δικαιώματα υπερχρήστη και θα δώσουμε το όνομα του Role και το password .

Δεν βάζουμε ημερομηνία λήξης στον χρήστη και πατάμε οκ.



Και θα επιλέξουμε την επιλογή Admin



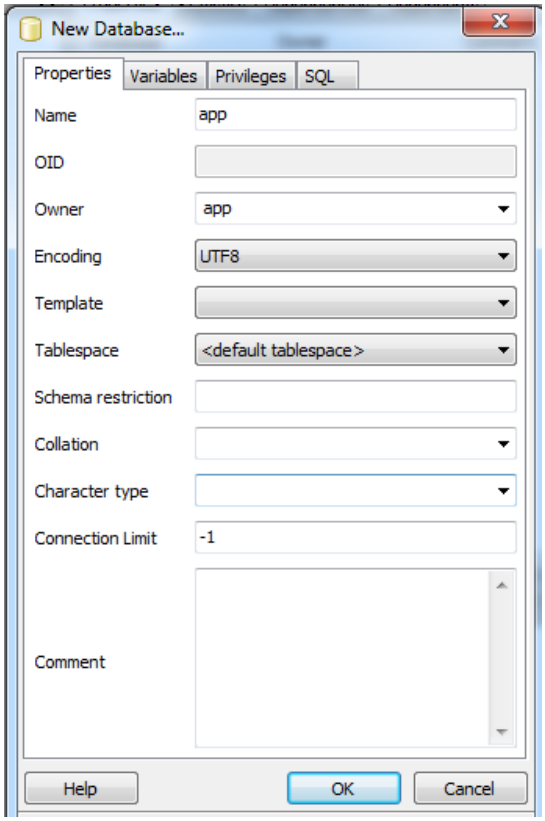
Και όλη η δημιουργία χρήστη admin μπορεί να μεταφραστεί σε Sql ως

```
CREATE ROLE app LOGIN ENCRYPTED PASSWORD 'md5fe63fea7be31b0200b496d08bc6b517d'
  SUPERUSER CREATEDB CREATEROLE
   VALID UNTIL 'infinity';
```

Στο επόμενο βήμα είναι η δημιουργία της βάσης μας μπορούμε να δημιουργήσουμε την βάση επίσης με τον οδηγό που μας δίνει η postgress αλλά και με ένα script

```
CREATE DATABASE app
  WITH ENCODING='UTF8'
       OWNER=app
       CONNECTION LIMIT=-1;
```

Με τον οδηγό έχουμε



μετά θα κάνουμε εισαγωγή των πινάκων που θα έχουν τα δεδομένα μας .

Θα υπάρξει ο πίνακας app_users για την αποθήκευση των χρηστών

Τα πεδία αυτά θα χρησιμοποιούνται για το login καθώς και όταν θα χρειάζεται να γίνει κάποια άλλη διεργασία.
Τα πεδία είναι πάρα πολλή απλά στην δημιουργία τους έχουν τα χαρακτηριστικά του χρήστη καθώς και τον ρόλο του στην εφαρμογή.

Αναλυτικά η δημιουργία του πίνακα app_users σε sql θα είναι

```
-- Table: app_users

-- DROP TABLE app_users;

CREATE TABLE app_users
(
  id integer NOT NULL,
```

```
  email character varying(255),
  firstname character varying(255),
  lastname character varying (255),
  "password" character varying(255),
  usercode integer NOT NULL,
  username character varying(255),
  CONSTRAINT app_users_pkey PRIMARY KEY (id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE app_users OWNER TO app;
```

Στον πίνακα profile έχουμε τα χαρακτηριστικά για το είδος των χρηστών που θα έχουμε. Όλοι θα έχουν ένα id που θα είναι primary key και θα έχουν codename όπου θα είναι μια συντόμευση του είδους χρήστη που θα είναι.
Θα έχουν επίσης description δηλαδή εάν θα είναι record απλά εγγραφή εάν θα είναι simple user απλός χρήστης εάν θα είναι Admin διαχειριστής με κάποια δικαιώματα και οποιοσδήποτε είδος χρήστη μπορεί να προστεθεί μετά σε περίπτωση που θέλουμε να αναπτύξουμε την βάση μας και τα είδη χρηστών.

Αναλυτικά η δημιουργία του πίνακα profile σε sql θα είναι.

```
-- Table: profile

-- DROP TABLE profile;

CREATE TABLE profile
(
  id bigint NOT NULL,
  codename character varying(255),
  description character varying(255),
  "name" character varying(255),
  CONSTRAINT profile_pkey PRIMARY KEY (id)
)
WITH (
  OIDS=FALSE
```

```
);
ALTER TABLE profile OWNER TO app;
```

Ο πίνακας app_users_profile έχει δυο ξένα κλειδιά με τα οποία ενώνει δυο άλλους πινάκες

Αναλυτικά η δημιουργία του πίνακα app_users_profile σε sql θα είναι.

```
-- Table: app_users_profile

-- DROP TABLE app_users_profile;

CREATE TABLE app_users
(
  id int NOT NULL,
  email character varying(255),
  firstname character varying(255),
    lastname character varying(255),
  "password" character varying(255),
  username character varying(255),
  CONSTRAINT app_users_pkey PRIMARY KEY (id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE app_users OWNER TO dsp;;
```

Ο πίνακας graph που θα αποθηκεύει τα στοιχεία που θα μπορεί να χρησιμοποιήσει ο χρήστης.

```
-- Table: graph
-- DROP TABLE graph;

CREATE TABLE graph
(
  grh_id bigint NOT NULL,
  grh__item_model character varying(255),
```

```
  grh_registrationnumber character varying(255),
  uuid bigint,
  CONSTRAINT graph_pkey PRIMARY KEY (grh_id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE graph OWNER TO dsp;
```

Ο πίνακας logicgates_items που θα αποθηκεύει τα στοιχεία που θα μπορεί να χρησιμοποιήσει ο χρήστης.

```
-- Table: logicgates_items
-- DROP TABLE logicgates_items;

CREATE TABLE logicgates_items
(
  lg_id_item bigint NOT NULL,
  lg_item_model character varying(255),
  lg_item_type character varying(255),
  lg_registrationnumber character varying(255),
  uuid bigint,
  CONSTRAINT logicgates_items_pkey PRIMARY KEY (lg_id_item)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE logicgates_items OWNER TO dsp;
```

Ο πίνακας uml_items που θα αποθηκεύει τα στοιχεία που θα μπορεί να χρησιμοποιήσει ο χρήστης.

```
-- Table: uml_items
-- DROP TABLE uml_items;

CREATE TABLE uml_items
```

```
(
  uml_id_item bigint NOT NULL,
  item_model character varying(255),
  item_type character varying(255),
  registrationnumber character varying(255),
  uuid bigint,
  CONSTRAINT uml_items_pkey PRIMARY KEY (uml_id_item)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE uml_items OWNER TO dsp;
```

JavaScript

Η Java έχει πολύ καλή εφαρμογή στη δημιουργία γραφικών σε μη διαδικτυακές εφαρμογές (desktop). Με την γενίκευση της χρήσης της και την ανάπτυξη του διαδικτύου ανάπτυξε τεχνολογίες που κάνουν εφικτή την εκτέλεση της σε προγράμματα πλοήγησης. Τα μικρά αυτά προγράμματα ονομάζονται Applet. Τα Applet σήμερα λόγω δυσκολίας συγγραφής και ταχύτητας δεν χρησιμοποιούνται σε μεγάλη κλίμακα και την θέση τους την έχει πάρει η JavaScript και το Flash.

Η JavaScirpt δεν έχει ανάγκη ιδιαίτερων εργαλείων συγγραφής και δεν χρειάζεται μεταγλωττιστή. Αυτό που χρειάζεται για να λειτουργήσει είναι να γραφτεί ένα script και να το εισάγουμε στην σελίδα που θέλουμε να εκτελεστεί. Μετά το πρόγραμμα πλοήγησης -που θα πρέπει να υποστηρίζει την JavaScirpt- θα διαβάσει το script που έχουμε γράψει και θα το τρέξει.

Διαφορές Java και JavaScript

Java	**JavaScript**
Περίπλοκη στην χρήση	Σχετικά εύκολη στην χρήση
Θέλει το Jdk	Δεν χρειάζεται τίποτα
Τα προγράμματα μεταγλωττίζονται	Τα προγράμματα ενσωματώνονται στην σελίδα
Είναι αργή στο internet	Τρέχει πολύ γρήγορα
Έχει περισσότερες δυνατότητες	Είναι καλή για σχετικά απλές εφαρμογές

Είναι προσανατολισμένη στο αντικείμενο	Είναι βασισμένη στο αντικείμενο

Η JavaScript χρειάζεται ένα browser για να λειτουργήσει και είναι φτιαγμένη για τον browser. Στην εφαρμογή θα χρησιμοποιήσουμε JavaScript για την σχεδίαση που θέλουμε να κάνουμε έχουμε την δυνατότητα να χρησιμοποιήσουμε αρκετά εργαλεία που θα μπορέσουμε να κάνουμε κάποια σχεδίαση.

Η επιλογή έγινε για το εργαλείο RoCanvas γραμμένο σε JavaScript και μπορεί να βρεθεί στο http://re.trotoys.com/article/rocanvas/ .

Για την αποθήκευση του σχεδίου χρησιμοποιούμε το εργαλείο Canvas2Image και base64.και μπορούν να βρεθούν στο http://www.nihilogic.dk/labs/canvas3d/

Για το Accordion και τα αλλά functions χρησιμοποιούμε το Jquery. Αναλυτικά έχουμε τα jquery-1.7.2 , jquery.ui.accordion, jquery.ui.core , jquery.ui.widget.js, jquery.ui.mouse, jquery.ui.draggable, jquery, jquery.effects.core, jquery-draggable-droppable.custom.min, jquery-ui-1.8.20.custom. και μπορούν να βρεθούν στο http://www.Jquery.com

Για να γίνει η χρήση του εργαλείου αυτού έχουμε κάνει τοποθέτηση του κώδικα που μας δίνεται στον φάκελο που έχουμε για τα διάφορα βοηθητικά αρχεία που θέλουμε τον resources.

Κεφάλαιο 10. Συμπεράσματα - Μελλοντικές επεκτάσεις

Για την υλοποίηση της εφαρμογής έχουν αξιοποιηθεί πολλές διαφορετικές τεχνολογίες. Η java χρησιμοποιείται για το Back End αλλά και για την διαχείριση των σελίδων καθώς και για την σύνδεση και τις συνεδρίες (sessions) των χρηστών . Η JavaScript αξιοποιείται στο κομμάτι της δημιουργίας των σχεδίων και αποθήκευσης σε διάφορα είδη .png ,jpeg κ.α.

Θα μπορούσαμε να δημιουργήσουμε την εφαρμογή με μια μόνο τεχνολογία π.χ Java και όχι με συνδυασμό Java, JavaScript, Jsf αλλά δεν θα είχαμε την δυνατότητα και την ευκολία να έχουμε σε πολύ μικρό χρονικό διάστημα το αποτέλεσμα που θέλουμε.

Υπάρχει η δυνατότητα να δημιουργήσουμε σε Java Applets τα οποία θα φορτώναμε κάθε φορά και θα μπορούσαμε να έχουμε την σχεδίαση. Όμως δε θα μπορούσαμε να έχουμε το αποτέλεσμα που έχουμε σήμερα καθώς τα Applets χρειάζονται αρκετό χρόνο για να φορτώσουν και με τις σημερινές τεχνολογίες είναι αρκετά αργά. Βλέπουμε ότι η Java υποστηρίζει πάρα πολύ καλά το Back End μέρος της εφαρμογής μας αλλά δεν μπορεί να ικανοποίηση το μέρος που εκτελείται στο πρόγραμμα πλοήγησης.

Παρατηρήσαμε ότι με την βοήθεια ενός framework μπορούμε να ελαχιστοποιήσουμε τον κόπο ανάπτυξης. Στη συγκεκριμένη περίπτωση έχουμε χρησιμοποιήσει το JSF, έτσι έχουμε την δυνατότητα να γράψουμε το δυνατό ελάχιστο κώδικα που θέλουμε για την ανάπτυξη. Με τη χρήση της UML μπορούμε να βρούμε τις απαιτήσεις και με απλό τρόπο να τις αποτυπώσουμε σε χαρτί για να τις περάσουμε στην εφαρμογή μας.

Παρατηρούμε επίσης ότι η δημιουργία τις εφαρμογής μπορεί να γίνει με πολλούς διαφορετικούς τρόπους και τεχνολογίες και κάθε προγραμματιστής μπορεί να επιλέξει το δικό συνδυασμό τεχνολογιών για να υλοποιήσει την εφαρμογή του. Συμπεραίνουμε ότι για την δημιουργία μιας διαδικτυακής εφαρμογής χρειάζεται να έχουμε γνώση από πολλές διαφορετικές τεχνολογίες καθώς θα πρέπει να έχουμε και ικανότητες ανάλυσης προβλημάτων. Θα πρέπει ένας προγραμματιστής να έχει την δυνατότητα να μπορεί να δει πιο μακριά από όσο φτάνει έτσι ώστε η δημιουργία να είναι όσο πιο ρεαλιστική και ζωντανή γίνεται.

Η εφαρμογή έχει δημιουργηθεί στα πλαίσια της διπλωματικής και είναι κυρίως πρόγραμμα επίδειξης. ΟΙ μελλοντικές επεκτάσεις είναι αρκετές . Θα μπορεί κάποιος να αναπτύξει κάποια plug ins τα οποία θα δίνουν την δυνατότατα στον χρήστη να επιλέξει επιπλέον εργαλεία (τα οποία είναι απενεργοποιημένα).

Θα μπορεί να έχει πρόσβαση σε λογικές πύλες για την δημιουργία κάποιου ψηφιακού σχεδίου , θα μπορεί να κάνει κάποια διαγράμματα UML τα οποία θα μπορεί να συνδέσει,

θα μπορεί να σχεδιάσει κάποια βασικά σχήματα (3D κύβοι ,κύκλοι ,τετράγωνα, να γίνεται εισαγωγή κάποια εικόνας και να έχει την δυνατότητα να συνδεθεί με αυτή κάποιο άλλο σχήμα , να μετατρέψουμε την εικόνα η το σχέδιο σε κάτι άλλο (σαν το Photoshop).

Μπορούμε επίσης να δημιουργήσουμε ένα Forum και να το συνδέσουμε με την εφαρμογή. Έτσι θα έχουμε την επικοινωνία των μελών και των διαχειριστών ώστε να μοιράζεται γνώση για τα πιθανόν προβλήματα αλλά και να μπορούν να αναφέρουν πιθανά προβλήματα.

Στις μελλοντικές επεκτάσεις είναι η δυνατότητα συνεργατικού σχεδιασμού από 2 ή περισσότερους χρήστες ταυτόχρονα.

Κεφάλαιο 11. Βιβλιογραφία

1 The Design Patterns, Java Companion -- by James W. Cooper

2 Sun's core J2EE Patterns

3 Category patterns

4 Java Design Patterns Reference and Examples

5 Law of Demeter Wikipedia, the free encyclopedia

6 Ανάπτυξη συστήματος βάσει την μεθοδολογία iconix

Αλέξανδρος Ν.Χατζηγεωργιου ΕΑΠ πλη 24 2008 Σχεδιασμός Λογισμικού

7 Σύστημα έλεγχου ανελκυστήρων Βασίλης Χ.Γερογιαννης ΕΑΠ πλη 24 2004 Σχεδιασμός Λογισμικού

8 Μελέτη Περίπτωσης Ηλεκτρονικό κατάστημα Πάνος φιτσιλης ΕΑΠ πλη 24 2004 Σχεδιασμός Λογισμικού

9 The J2EE 1.4 Tutorial

10 PostgresSQL

11 Retro toys http://re.trotoys.com/article/rocanvas/

12 Wikipedia javascript http://el.wikipedia.org/wiki/JavaScript

13 http://www.nihilogic.dk/labs/canvas3d/

14 http://www.JQuery.com

15 http://www.netbeans.org

16 http://www.eclipse .org

17 http://en.wikipedia.org/wiki/Java_Persistence_API

18 http://en.wikipedia.org/wiki/JavaServer_Faces

19 http://en.wikipedia.org/wiki/GlassFish

20 http://en.wikipedia.org/wiki/Session_%28computer_science%29

21 http://en.wikipedia.org/wiki/Object-relational_mapping

Κεφάλαιο 12. Παράρτημα 1 - Οδηγός χρήσης

Για την λειτουργία της εφαρμογής πρέπει να κάνουμε/εγκαταστήσουμε τα εξής

Δημιουργούμε το Path θα εισάγουμε τα Path CLASSPATH JAVA_HOME
Το PATH διατηρεί μια λίστα με τα προγράμματα που είναι εκτελέσιμα.
Το CLASSPATH διατηρεί λίστα και χρησιμοποιείται από Java εφαρμογές μόνο.

PATH(Win)
Για Java εφαρμογές πρέπει να βάλουμε τα εξής
JDK's "bin" φάκελος (e.g., "c:\Program Files\java\jdk1.7.0_xx\bin"),
Που έχει JDK Προγράμματα σαν "javac.exe" Και Java Runtime "java.exe".

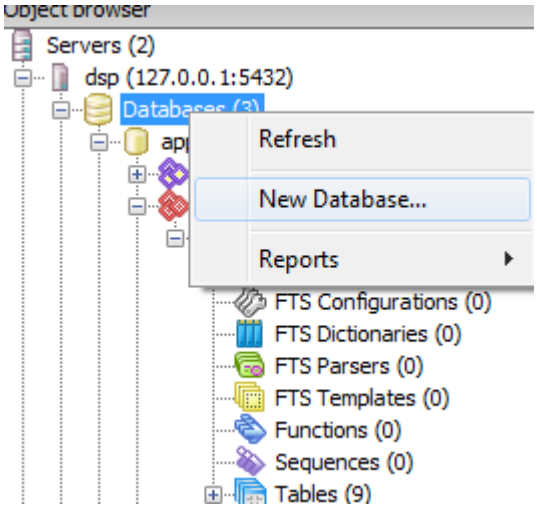
Για να δημιουργήσουμε το PATH
Control Panel=> System=>Advanced system settings => αλλαγή σε Advanced tab=> "Enviroment variables"=>επιλέγουμε System Variables (Για όλους τους χρήστες)η User Variables για τον τρέχον χρήστη =>Επιλέγουμε "PATH" και "Edit" => Insert το JDK bin μονοπάτι π.χ "c:\Program Files\java\jdk.1.7.0\bin" και βάζουμε στο τέλος το « ; »

CLASSPATH(win)
Για να δημιουργήσουμε το CLASSPATH
Control Panel=> επιλέγουμε Edit=> μεταβλητή Value και βάζουμε τα μονοπάτια

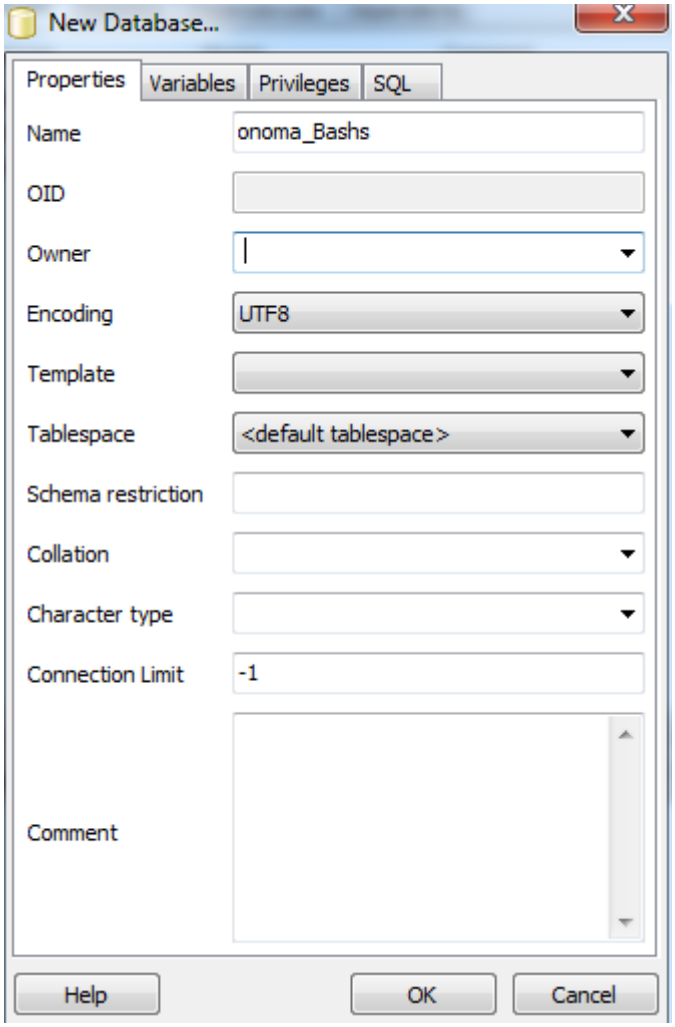
JAVA_HOME και JRE_HOME
Το JRE (Java Runtime) Χρειάζεται για το τρέξιμο Java Εφαρμογών.
Set JAVA_HOME στο JDK Μονοπάτι
Π.χ "c:\Program Files \java\jdk1.7.0_xx").
Το JAVA_HOME χρειάζεται για να τρέξει ο Tomcat αλλά και άλλες εφαρμογές .
Το JRE_HOME είναι προαιρετικό .

Χρειαζόμαστε ένα Database Server για την βάση που θα έχουμε. Θα χρησιμοποιήσουμε τον PostgresSQL και διατίθεται δωρεάν.

Κάνουμε εγκατάσταση τον PostgresSQL και δημιουργούμε ένα νέο λογαριασμό χρηστή.
Εφόσον έχουμε δημιουργήσει τον Νέο χρήστη τότε δημιουργούμε μια καινούργια βάση.



Δίνουμε το όνομα που θέλουμε στην βάση ορίζουμε δικαιώματα και έχουμε



Σε SQL

```
CREATE DATABASE "onoma_Bashs"
 WITH ENCODING='UTF8'
     CONNECTION LIMIT=-1;
```

Την στιγμή που θα φτιάξουμε την βάση θα είναι κενή δεν θα έχει ούτε ένα table
Για να την γεμίσουμε θα χρησιμοποιήσουμε το Cmd shell και θα φτιάξουμε ένα αρχείο που είναι εκτελέσιμο .

Το Script είναι το εξής
echo on
SET PGPASSFILE=C:\Webapp\onoma_me_to_arxeio_tou_password.conf
SET PGPASSWORD=dsp1234
"C:\Webapp\PostgresSQL\8.4\bin\dropdb.exe" -U app -i app
"C:\Webapp\PostgresSQL\8.4\bin\psql.exe" -U app -f C:\Webapp\ onoma_me_to_arxeio_tou backup.backup

Θα λέγεται
pg_restoreall.bat

θα φτιάξουμε και το αρχείο με το οποίο θα κάνουμε το backup στην βάση μας

```
echo off
SET PGPASSFILE=C:\Webapp\ onoma_me_to_arxeio_tou_password.conf
SET PGPASSWORD=dsp1234
"C:\webapp\pgsql\bin\pg_dumpall.exe" -U App C:\webapp\ onoma_me_to_arxeio_tou backup.backup current
```

Όνομα αρχείου pg_dumpall.bat

Θα κάνουμε και ένα ακόμα αρχείο το onoma_me_to_arxeio_tou_password.conf θα το βάλουμε στο ίδιο φάκελο με το αρχείο τις βάσης για να μπορεί να πάρει το password και το username από μόνο το backup script.
Το περιεχόμενο θα είναι

```
hostname:127.0.0.1
port:
database:app
username:app
password:app1234
```

```
Sql
CREATE TABLE app_users
(
  id int NOT NULL,
  email character varying(255),
  firstname character varying(255),
    lastname character varying(255),
  "password" character varying(255),
  username character varying(255),
  CONSTRAINT app_users_pkey PRIMARY KEY
(id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE app_users OWNER TO dsp;

-- Table: draw

-- DROP TABLE draw;

CREATE TABLE draw
(
  id integer NOT NULL,
  "appUser_id" integer,
  CONSTRAINT draw_pkey PRIMARY KEY (id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE draw OWNER TO dsp;

-- Table: graph

-- DROP TABLE graph;

CREATE TABLE graph
(
  grh_id bigint NOT NULL,
          grh__item_model     character
varying(255),
      grh_registrationnumber   character
varying(255),
  uuid bigint,
    CONSTRAINT  graph_pkey  PRIMARY  KEY
(grh_id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE graph OWNER TO dsp;


-- Table: logicgates_items

-- DROP TABLE logicgates_items;

CREATE TABLE logicgates_items
(
  lg_id_item bigint NOT NULL,
  lg_item_model character varying(255),
  lg_item_type character varying(255),
      lg_registrationnumber   character
varying(255),
  uuid bigint,
      CONSTRAINT   logicgates_items_pkey
PRIMARY KEY (lg_id_item)
)
WITH (
  OIDS=FALSE
);
ALTER  TABLE  logicgates_items  OWNER  TO
dsp;

-- Table: pert_items

-- DROP TABLE pert_items;

CREATE TABLE pert_items
(
  prt_id_item bigint NOT NULL,
           prt_item_model     character
varying(255),
      prt_registrationnumber   character
varying(255),
  uuid bigint,
    CONSTRAINT  pert_items_pkey  PRIMARY
KEY (prt_id_item)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE pert_items OWNER TO dsp;
```

```
-- Table: profile

-- DROP TABLE profile;

CREATE TABLE profile
(
  id bigint NOT NULL,
  codename character varying(255),
  description character varying(255),
  "name" character varying(255),
   CONSTRAINT profile_pkey PRIMARY KEY
(id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE profile OWNER TO dsp;

-- Table: uml_items

-- DROP TABLE uml_items;

CREATE TABLE uml_items
(
  uml_id_item bigint NOT NULL,
  item_model character varying(255),
  item_type character varying(255),
        registrationnumber    character
varying(255),
  uuid bigint,
  CONSTRAINT uml_items_pkey PRIMARY KEY
(uml_id_item)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE uml_items OWNER TO dsp;

Java
AppUserController
package org.com.Classes;

import org.com.Entitys.AppUsers;
import org.com.Classes.util.JsfUtil;
import
org.com.Classes.util.PaginationHelper;
import
org.com.JPAControllers.AppUsersJpaContr
oller;

import java.io.Serializable;
import java.util.ResourceBundle;
import javax.annotation.Resource;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import
javax.faces.component.UIComponent;
import
javax.faces.context.FacesContext;
import javax.faces.convert.Converter;
import
javax.faces.convert.FacesConverter;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;
import javax.faces.model.SelectItem;
import
javax.persistence.EntityManagerFactory;
import
javax.persistence.PersistenceUnit;
import
javax.transaction.UserTransaction;

@ManagedBean(name                    =
"appUsersController")
@SessionScoped
public     class     AppUsersController
implements Serializable {

    @Resource
    private UserTransaction utx = null;
          @PersistenceUnit(unitName   =
"sample_testingPU")
     private EntityManagerFactory emf =
null;
    private AppUsers current;
    private DataModel items = null;
         private  AppUsersJpaController
jpaController = null;
             private   PaginationHelper
pagination;
    private int selectedItemIndex;

    public AppUsersController() {
    }

    public AppUsers getSelected() {
        if (current == null) {
            current = new AppUsers();
            selectedItemIndex = -1;
        }
        return current;
    }

         private  AppUsersJpaController
getJpaController() {
        if (jpaController == null) {
```

```
                    jpaController = new
AppUsersJpaController(utx, emf);
        }
        return jpaController;
    }

              public    PaginationHelper
getPagination() {
        if (pagination == null) {
                      pagination = new
PaginationHelper(10) {

                @Override
                             public int
getItemsCount() {
                                  return
getJpaController().getAppUsersCount();
                }

                @Override
                       public DataModel
createPageDataModel() {
                              return new
ListDataModel(getJpaController().findAp
pUsersEntities(getPageSize(),
getPageFirstItem()));
                }
            };
        }
        return pagination;
    }

    public String prepareList() {
        recreateModel();
        return "List";
    }

    public String prepareView() {
                current = (AppUsers)
getItems().getRowData();
                   selectedItemIndex =
pagination.getPageFirstItem()         +
getItems().getRowIndex();
        return "View";
    }

    public String prepareCreate() {
        current = new AppUsers();
        selectedItemIndex = -1;
        return "Create";
    }

    public String create() {
        try {
            getJpaController().create(c
urrent);
            JsfUtil.addSuccessMessage(R
esourceBundle.getBundle("/Bundle").getS
tring("AppUsersCreated"));
            return prepareCreate();
        } catch (Exception e) {
             JsfUtil.addErrorMessage(e,
ResourceBundle.getBundle("/Bundle").get
String("PersistenceErrorOccured"));
            return null;
        }
    }

    public String prepareEdit() {
                current = (AppUsers)
getItems().getRowData();
                   selectedItemIndex =
pagination.getPageFirstItem()         +
getItems().getRowIndex();
        return "Edit";
    }

    public String update() {
        try {
            getJpaController().edit(cur
rent);
            JsfUtil.addSuccessMessage(R
esourceBundle.getBundle("/Bundle").getS
tring("AppUsersUpdated"));
            return "View";
        } catch (Exception e) {
             JsfUtil.addErrorMessage(e,
ResourceBundle.getBundle("/Bundle").get
String("PersistenceErrorOccured"));
            return null;
        }
    }

    public String destroy() {
                current = (AppUsers)
getItems().getRowData();
                   selectedItemIndex =
pagination.getPageFirstItem()         +
getItems().getRowIndex();
        performDestroy();
        recreatePagination();
        recreateModel();
        return "List";
    }

    public String destroyAndView() {
```

```
        performDestroy();
        recreateModel();
        updateCurrentItem();
        if (selectedItemIndex >= 0) {
            return "View";
        } else {

            recreateModel();
            return "List";
        }
    }

    private void performDestroy() {
        try {
             getJpaController().destroy(
current.getId());
             JsfUtil.addSuccessMessage(R
esourceBundle.getBundle("/Bundle").getS
tring("AppUsersDeleted"));
        } catch (Exception e) {
              JsfUtil.addErrorMessage(e,
ResourceBundle.getBundle("/Bundle").get
String("PersistenceErrorOccured"));
        }
    }

    private void updateCurrentItem() {
                        int   count   =
getJpaController().getAppUsersCount();
         if (selectedItemIndex >= count)
{

             selectedItemIndex = count -
1;

                                     if
(pagination.getPageFirstItem()       >=
count) {
                 pagination.previousPage
();
             }
        }
        if (selectedItemIndex >= 0) {
                              current   =
getJpaController().findAppUsersEntities
(1, selectedItemIndex).get(0);
        }
    }

    public DataModel getItems() {
        if (items == null) {
                                items   =
getPagination().createPageDataModel();
        }
        return items;
    }

    private void recreateModel() {
        items = null;
    }

    private void recreatePagination() {
        pagination = null;
    }

    public String next() {
        getPagination().nextPage();
        recreateModel();
        return "List";
    }

    public String previous() {
        getPagination().previousPage();
        recreateModel();
        return "List";
    }

                  public    SelectItem[]
getItemsAvailableSelectMany() {
                                     return
JsfUtil.getSelectItems(getJpaController
().findAppUsersEntities(), false);
    }

                  public    SelectItem[]
getItemsAvailableSelectOne() {
                                     return
JsfUtil.getSelectItems(getJpaController
().findAppUsersEntities(), true);
    }

            @FacesConverter(forClass   =
AppUsers.class)
               public   static   class
AppUsersControllerConverter  implements
Converter {

                         public   Object
getAsObject(FacesContext  facesContext,
UIComponent component, String value) {
                 if (value == null ||
value.length() == 0) {
                return null;
            }
                      AppUsersController
controller    =    (AppUsersController)
```

```
facesContext.getApplication().getELReso
lver().
                    getValue(facesConte
xt.getELContext(),                null,
"appUsersController");
                                return
controller.getJpaController().findAppUs
ers(getKey(value));
        }

         java.lang.Integer getKey(String
value) {
            java.lang.Integer key;
                                  key   =
Integer.valueOf(value);
            return key;
        }

                                  String
getStringKey(java.lang.Integer value) {
                 StringBuffer  sb  =  new
StringBuffer();
            sb.append(value);
            return sb.toString();
        }

                        public   String
getAsString(FacesContext   facesContext,
UIComponent component, Object object) {
            if (object == null) {
                return null;
            }
                 if  (object  instanceof
AppUsers) {
                AppUsers o = (AppUsers)
object;
                                return
getStringKey(o.getId());
            } else {
                              throw  new
IllegalArgumentException("object   "   +
object   +   "   is   of   type   "   +
object.getClass().getName()     +     ";
expected        type:        "        +
AppUsersController.class.getName());
            }
        }
    }
}

WuserController
package org.com.Classes;

import org.com.Entitys.Wuser;
import org.com.Classes.util.JsfUtil;
import
org.com.Classes.util.PaginationHelper;
import
org.com.JPAControllers.WuserJpaControll
er;

import java.io.Serializable;
import java.util.ResourceBundle;
import javax.annotation.Resource;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import
javax.faces.component.UIComponent;
import
javax.faces.context.FacesContext;
import javax.faces.convert.Converter;
import
javax.faces.convert.FacesConverter;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;
import javax.faces.model.SelectItem;
import
javax.persistence.EntityManagerFactory;
import
javax.persistence.PersistenceUnit;
import
javax.transaction.UserTransaction;

@ManagedBean(name = "wuserController")
@SessionScoped
public class WuserController implements
Serializable {

    @Resource
    private UserTransaction utx = null;
          @PersistenceUnit(unitName   =
"sample_testingPU")
     private EntityManagerFactory emf =
null;
    private Wuser current;
    private DataModel items = null;
           private   WuserJpaController
jpaController = null;
             private   PaginationHelper
pagination;
    private int selectedItemIndex;

    public WuserController() {
    }

    public Wuser getSelected() {
```

```
        if (current == null) {
            current = new Wuser();
            selectedItemIndex = -1;
        }
        return current;
    }

           private   WuserJpaController
getJpaController() {
        if (jpaController == null) {
                  jpaController  =  new
WuserJpaController(utx, emf);
        }
        return jpaController;
    }

              public    PaginationHelper
getPagination() {
        if (pagination == null) {
                     pagination  =  new
PaginationHelper(10) {

                @Override
                            public  int
getItemsCount() {
                                 return
getJpaController().getWuserCount();
                }

                @Override
                       public  DataModel
createPageDataModel() {
                             return  new
ListDataModel(getJpaController().findWu
serEntities(getPageSize(),
getPageFirstItem()));
                }
            };
        }
        return pagination;
    }

    public String prepareList() {
        recreateModel();
        return "List";
    }

    public String prepareView() {
                    current   =   (Wuser)
getItems().getRowData();
                  selectedItemIndex   =
pagination.getPageFirstItem()         +
getItems().getRowIndex();
        return "View";
    }

    public String prepareCreate() {
        current = new Wuser();
        selectedItemIndex = -1;
        return "Create";
    }

    public String create() {
        try {
             getJpaController().create(c
urrent);
             JsfUtil.addSuccessMessage(R
esourceBundle.getBundle("/Bundle").getS
tring("WuserCreated"));
            return prepareCreate();
        } catch (Exception e) {
              JsfUtil.addErrorMessage(e,
ResourceBundle.getBundle("/Bundle").get
String("PersistenceErrorOccured"));
            return null;
        }
    }

    public String prepareEdit() {
                    current   =   (Wuser)
getItems().getRowData();
                   selectedItemIndex   =
pagination.getPageFirstItem()         +
getItems().getRowIndex();
        return "Edit";
    }

    public String update() {
        try {
             getJpaController().edit(cur
rent);
             JsfUtil.addSuccessMessage(R
esourceBundle.getBundle("/Bundle").getS
tring("WuserUpdated"));
            return "View";
        } catch (Exception e) {
              JsfUtil.addErrorMessage(e,
ResourceBundle.getBundle("/Bundle").get
String("PersistenceErrorOccured"));
            return null;
        }
    }

    public String destroy() {
                    current   =   (Wuser)
getItems().getRowData();
```

```
                    selectedItemIndex = pagination.getPageFirstItem() + getItems().getRowIndex();
        performDestroy();
        recreatePagination();
        recreateModel();
        return "List";
    }

    public String destroyAndView() {
        performDestroy();
        recreateModel();
        updateCurrentItem();
        if (selectedItemIndex >= 0) {
            return "View";
        } else {

            recreateModel();
            return "List";
        }
    }

    private void performDestroy() {
        try {
             getJpaController().destroy(current.getId());
             JsfUtil.addSuccessMessage(ResourceBundle.getBundle("/Bundle").getString("WuserDeleted"));
        } catch (Exception e) {
              JsfUtil.addErrorMessage(e, ResourceBundle.getBundle("/Bundle").getString("PersistenceErrorOccured"));
        }
    }

    private void updateCurrentItem() {
                        int count = getJpaController().getWuserCount();
         if (selectedItemIndex >= count) {

            selectedItemIndex = count - 1;

                                    if (pagination.getPageFirstItem() >= count) {
                pagination.previousPage();
            }
        }
        if (selectedItemIndex >= 0) {
                                current = getJpaController().findWuserEntities(1, selectedItemIndex).get(0);
        }
    }

    public DataModel getItems() {
        if (items == null) {
                                items = getPagination().createPageDataModel();
        }
        return items;
    }

    private void recreateModel() {
        items = null;
    }

    private void recreatePagination() {
        pagination = null;
    }

    public String next() {
        getPagination().nextPage();
        recreateModel();
        return "List";
    }

    public String previous() {
        getPagination().previousPage();
        recreateModel();
        return "List";
    }

                public SelectItem[] getItemsAvailableSelectMany() {
                                    return JsfUtil.getSelectItems(getJpaController().findWuserEntities(), false);
    }

                public SelectItem[] getItemsAvailableSelectOne() {
                                    return JsfUtil.getSelectItems(getJpaController().findWuserEntities(), true);
    }

            @FacesConverter(forClass = Wuser.class)
               public static class WuserControllerConverter implements Converter {
```

```
                        public Object
getAsObject(FacesContext facesContext,
UIComponent component, String value) {
                 if (value == null ||
value.length() == 0) {
                return null;
            }
             WuserController controller
=                    (WuserController)
facesContext.getApplication().getELReso
lver().
                    getValue(facesConte
xt.getELContext(),                null,
"wuserController");
                                return
controller.getJpaController().findWuser
(getKey(value));
        }

        java.lang.Integer getKey(String
value) {
            java.lang.Integer key;
                                key   =
Integer.valueOf(value);
            return key;
        }

                                String
getStringKey(java.lang.Integer value) {
                StringBuffer sb = new
StringBuffer();
            sb.append(value);
            return sb.toString();
        }

                        public   String
getAsString(FacesContext  facesContext,
UIComponent component, Object object) {
            if (object == null) {
                return null;
            }
                 if (object instanceof
Wuser) {
                    Wuser o = (Wuser)
object;
                                return
getStringKey(o.getId());
            } else {
                            throw new
IllegalArgumentException("object   "   +
object   +   "   is   of   type   "   +
object.getClass().getName()    +    ";
expected       type:       "       +
WuserController.class.getName());
            }
        }
    }
}

JsfUtil
package org.com.Classes.util;

import java.util.List;
import
javax.faces.application.FacesMessage;
import
javax.faces.component.UIComponent;
import
javax.faces.context.FacesContext;
import javax.faces.convert.Converter;
import javax.faces.model.SelectItem;

public class JsfUtil {

          public  static  SelectItem[]
getSelectItems(List<?>        entities,
boolean selectOne) {
              int  size  =  selectOne  ?
entities.size() + 1 : entities.size();
             SelectItem[] items = new
SelectItem[size];
        int i = 0;
        if (selectOne) {
                        items[0]  =  new
SelectItem("", "---");
            i++;
        }
        for (Object x : entities) {
                     items[i++]  =  new
SelectItem(x, x.toString());
        }
        return items;
    }

               public    static    void
addErrorMessage(Exception   ex,   String
defaultMsg) {
                        String   msg   =
ex.getLocalizedMessage();
        if (msg != null && msg.length()
> 0) {
            addErrorMessage(msg);
        } else {
             addErrorMessage(defaultMsg)
;
```

```
        }
    }

                public    static    void
addErrorMessages(List<String>  messages)
{
        for (String message : messages)
{
            addErrorMessage(message);
        }
    }

                public    static    void
addErrorMessage(String msg) {
           FacesMessage facesMsg = new
FacesMessage(FacesMessage.SEVERITY_ERRO
R, msg, msg);
        FacesContext.getCurrentInstance
().addMessage(null, facesMsg);
    }

                public    static    void
addSuccessMessage(String msg) {
           FacesMessage facesMsg = new
FacesMessage(FacesMessage.SEVERITY_INFO
, msg, msg);
        FacesContext.getCurrentInstance
().addMessage("successInfo", facesMsg);
    }

               public    static    String
getRequestParameter(String key) {
                                  return
FacesContext.getCurrentInstance().getEx
ternalContext().getRequestParameterMap(
).get(key);
    }

               public    static    Object
getObjectFromRequestParameter(String
requestParameterName,          Converter
converter, UIComponent component) {
                       String   theId   =
JsfUtil.getRequestParameter(requestPara
meterName);
                                  return
converter.getAsObject(FacesContext.getC
urrentInstance(), component, theId);
    }
}
```

PaginationHelper

```
package org.com.Classes.util;

import javax.faces.model.DataModel;

public abstract class PaginationHelper
{

    private int pageSize;
    private int page;

           public   PaginationHelper(int
pageSize) {
        this.pageSize = pageSize;
    }

                public    abstract    int
getItemsCount();

           public   abstract   DataModel
createPageDataModel();

    public int getPageFirstItem() {
        return page * pageSize;
    }

    public int getPageLastItem() {
          int i = getPageFirstItem() +
pageSize - 1;
         int count = getItemsCount() -
1;
        if (i > count) {
            i = count;
        }
        if (i < 0) {
            i = 0;
        }
        return i;
    }

    public boolean isHasNextPage() {
         return (page + 1) * pageSize +
1 <= getItemsCount();
    }

    public void nextPage() {
        if (isHasNextPage()) {
            page++;
        }
```

```
    }

     public boolean isHasPreviousPage()
{
        return page > 0;
    }

    public void previousPage() {
        if (isHasPreviousPage()) {
            page--;
        }
    }

    public int getPageSize() {
        return pageSize;
    }
}
```

AppUsers

```
/*
  * To change this template, choose
Tools | Templates
 * and open the template in the editor.
 */
package org.com.Entitys;

import java.io.Serializable;
import javax.persistence.*;
import
javax.validation.constraints.NotNull;
import
javax.validation.constraints.Size;
import
javax.xml.bind.annotation.XmlRootElemen
t;

/**
 *
 * @author giorgoCh
 */
@Entity
@Table(name = "app_users", catalog =
"app", schema = "public")
@XmlRootElement
@NamedQueries({
                    @NamedQuery(name    =
"AppUsers.findAll", query = "SELECT a
FROM AppUsers a"),
                    @NamedQuery(name    =
"AppUsers.findById", query = "SELECT a
FROM AppUsers a WHERE a.id = :id"),
                    @NamedQuery(name    =
"AppUsers.findByEmail", query = "SELECT
a FROM AppUsers a WHERE a.email =
:email"),
                    @NamedQuery(name    =
"AppUsers.findByFirstname",   query   =
"SELECT a FROM AppUsers a WHERE
a.firstname = :firstname"),
                    @NamedQuery(name    =
"AppUsers.findByLastname",   query   =
"SELECT a FROM AppUsers a WHERE
a.lastname = :lastname"),
                    @NamedQuery(name    =
"AppUsers.findByPassword",   query   =
"SELECT a FROM AppUsers a WHERE
a.password = :password"),
                    @NamedQuery(name    =
"AppUsers.findByUsername",   query   =
"SELECT a FROM AppUsers a WHERE
a.username = :username")})
public   class   AppUsers   implements
Serializable {
          private   static   final   long
serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
      @Column(name = "id", nullable =
false)
    private Integer id;
        //  @Pattern(regexp="[a-z0-9!#$
%&'*+/=?^_`{|}~-]+(?:\\.[a-z0-9!#$
%&'*+/=?^_`{|}~-]+)*@(?:[a-z0-9](?:[a-
z0-9-]*[a-z0-9])?\\.)+[a-z0-9](?:[a-z0-
9-]*[a-z0-9])?",       message="Invalid
email")//if the field contains email
address consider using this annotation
to enforce field validation
    @Size(max = 255)
      @Column(name = "email", length =
255)
    private String email;
    @Size(max = 255)
     @Column(name = "firstname", length
= 255)
    private String firstname;
    @Size(max = 255)
    @Column(name = "lastname", length =
255)
    private String lastname;
    @Size(max = 255)
    @Column(name = "password", length =
255)
    private String password;
    @Size(max = 255)
```

```
    @Column(name = "username", length =
255)
    private String username;

    public AppUsers() {
    }

    public AppUsers(Integer id) {
        this.id = id;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getEmail() {
        return email;
    }

     public void setEmail(String email)
{
        this.email = email;
    }

    public String getFirstname() {
        return firstname;
    }

       public  void  setFirstname(String
firstname) {
        this.firstname = firstname;
    }

    public String getLastname() {
        return lastname;
    }

        public  void  setLastname(String
lastname) {
        this.lastname = lastname;
    }

    public String getPassword() {
        return password;
    }

        public  void  setPassword(String
password) {
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

        public  void  setUsername(String
username) {
        this.username = username;
    }

    @Override
    public int hashCode() {
        int hash = 0;
              hash  +=  (id  !=  null  ?
id.hashCode() : 0);
        return hash;
    }

    @Override
         public  boolean  equals(Object
object) {
         // TODO: Warning - this method
won't work in the case the id fields
are not set
              if  (!(object  instanceof
AppUsers)) {
            return false;
        }
           AppUsers other = (AppUsers)
object;
             if  ((this.id  ==  null  &&
other.id != null) || (this.id != null
&& !this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
                                    return
"org.com.Entitys.AppUsers[ id=" + id +
" ]";
    }

}
Wuser
/*
  *  To  change  this  template,  choose
Tools | Templates
 * and open the template in the editor.
 */
```

```
package org.com.Entitys;

import java.io.Serializable;
import java.util.Date;
import javax.persistence.*;
import
javax.validation.constraints.NotNull;
import
javax.validation.constraints.Size;
import
javax.xml.bind.annotation.XmlRootElemen
t;

/**
 *
 * @author giorgoCh
 */
@Entity
@Table(name = "wuser", catalog = "app",
schema = "public")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Wuser.findAll",
query = "SELECT w FROM Wuser w"),
                @NamedQuery(name    =
"Wuser.findById",  query  =  "SELECT  w
FROM Wuser w WHERE w.id = :id"),
                @NamedQuery(name    =
"Wuser.findByFirstname",    query    =
"SELECT   w   FROM   Wuser   w   WHERE
w.firstname = :firstname"),
                @NamedQuery(name    =
"Wuser.findByLastname", query = "SELECT
w  FROM  Wuser  w  WHERE  w.lastname  =
:lastname"),
                @NamedQuery(name    =
"Wuser.findByPassword", query = "SELECT
w  FROM  Wuser  w  WHERE  w.password  =
:password"),
                @NamedQuery(name    =
"Wuser.findBySince", query = "SELECT w
FROM Wuser w WHERE w.since = :since"),
                @NamedQuery(name    =
"Wuser.findByUsername", query = "SELECT
w  FROM  Wuser  w  WHERE  w.username  =
:username")})
public    class    Wuser    implements
Serializable {
          private  static  final  long
serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
      @Column(name  =  "id",  nullable  =
false)
    private Integer id;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 255)
         @Column(name  =  "firstname",
nullable = false, length = 255)
    private String firstname;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 255)
    @Column(name = "lastname", nullable
= false, length = 255)
    private String lastname;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 255)
    @Column(name = "password", nullable
= false, length = 255)
    private String password;
    @Column(name = "since")
    @Temporal(TemporalType.TIMESTAMP)
    private Date since;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 255)
    @Column(name = "username", nullable
= false, length = 255)
    private String username;

    public Wuser() {
    }

    public Wuser(Integer id) {
        this.id = id;
    }

      public  Wuser(Integer  id,  String
firstname,   String   lastname,   String
password, String username) {
        this.id = id;
        this.firstname = firstname;
        this.lastname = lastname;
        this.password = password;
        this.username = username;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
```

```
        this.id = id;
    }

    public String getFirstname() {
        return firstname;
    }

       public  void  setFirstname(String
firstname) {
        this.firstname = firstname;
    }

    public String getLastname() {
        return lastname;
    }

        public  void  setLastname(String
lastname) {
        this.lastname = lastname;
    }

    public String getPassword() {
        return password;
    }

        public  void  setPassword(String
password) {
        this.password = password;
    }

    public Date getSince() {
        return since;
    }

    public void setSince(Date since) {
        this.since = since;
    }

    public String getUsername() {
        return username;
    }

        public  void  setUsername(String
username) {
        this.username = username;
    }

    @Override
    public int hashCode() {
        int hash = 0;
              hash  +=  (id  !=  null  ?
id.hashCode() : 0);
        return hash;
    }

    @Override
         public  boolean  equals(Object
object) {
         // TODO: Warning - this method
won't work in the case the id fields
are not set
        if (!(object instanceof Wuser))
{
            return false;
        }
        Wuser other = (Wuser) object;
             if  ((this.id  ==  null  &&
other.id != null) || (this.id != null
&& !this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
                                    return
"org.com.Entitys.Wuser[  id="  +  id  +
" ]";
    }

}
```

IllegalOrphanException

```
package
org.com.JPAControllers.exceptions;

import java.util.ArrayList;
import java.util.List;

public   class   IllegalOrphanException
extends Exception {
    private List<String> messages;
                                   public
IllegalOrphanException(List<String>
messages) {
            super((messages != null &&
messages.size() > 0 ? messages.get(0) :
null));
        if (messages == null) {
                  this.messages  =  new
ArrayList<String>();
        }
        else {
```

```
            this.messages = messages;
        }
    }
    public List<String> getMessages() {
        return messages;
    }
}
```

NonexistentEntityException

```
package
org.com.JPAControllers.exceptions;

public class NonexistentEntityException
extends Exception {
                                public
NonexistentEntityException(String
message, Throwable cause) {
        super(message, cause);
    }
                                public
NonexistentEntityException(String
message) {
        super(message);
    }
}
```

PreexistingEntityException

```
package
org.com.JPAControllers.exceptions;

public class PreexistingEntityException
extends Exception {
                                public
PreexistingEntityException(String
message, Throwable cause) {
        super(message, cause);
    }
                                public
PreexistingEntityException(String
message) {
        super(message);
    }
}
```

RollbackFailureException

```
package
org.com.JPAControllers.exceptions;

public  class  RollbackFailureException
extends Exception {
                                public
RollbackFailureException(String
message, Throwable cause) {
        super(message, cause);
    }
                                public
RollbackFailureException(String
message) {
        super(message);
    }
}
```

AuthenticationPhaseListener

```
package org.com.login.pack;

import javax.el.ELContext;
import javax.el.ValueExpression;
import javax.faces.FacesException;
import
javax.faces.context.ExternalContext;
import
javax.faces.context.FacesContext;
import javax.faces.event.PhaseEvent;
import javax.faces.event.PhaseId;
import javax.faces.event.PhaseListener;

public                          class
AuthenticationPhaseListener  implements
PhaseListener {

        private  static  final  String
USER_LOGIN_OUTCOME = "login";

     public void afterPhase(PhaseEvent
event) {
               FacesContext  context  =
event.getFacesContext();

        if (userExists(context)) {

            return;
        } else {

                                    if
(requestingSecureView(context)) {
                context.responseComplet
e();
                context.getApplication(
).
                        getNavigationHa
ndler().handleNavigation(context,
                       null,
                        USER_LOGIN_OUTC
OME);
            }
```

```
        }
    }

     public void beforePhase(PhaseEvent
event) {
    }

    public PhaseId getPhaseId() {
        return PhaseId.RESTORE_VIEW;
    }

                    private     boolean
userExists(FacesContext context) {
           ExternalContext extContext =
context.getExternalContext();
                                  return
(extContext.getSessionMap().containsKey
(UserManager.USER_SESSION_KEY));
    }

                    private     boolean
requestingSecureView(FacesContext
context) {
           ExternalContext extContext =
context.getExternalContext();
                       String   path   =
extContext.getRequestPathInfo();
                                  return
(!"/login.jsp".equals(path)
&& !"/create.jsp".equals(path));
    }
}
```

UserManager

```
package org.com.login.pack;

import java.util.logging.Level;
import java.util.logging.Logger;
import
javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import
javax.faces.context.FacesContext;
import javax.persistence.EntityManager;
import
javax.persistence.NoResultException;
import
javax.persistence.PersistenceContext;

import javax.annotation.Resource;
import javax.servlet.http.HttpSession;
import javax.transaction.*;
import org.com.Entitys.AppUsers;

@ManagedBean(name    =    "usermanager",
eager = true)
@SessionScoped
public class UserManager {

         public  static  final  String
USER_SESSION_KEY = "user";
    @PersistenceContext
    private EntityManager em;
    @Resource
    private UserTransaction utx;
    private String username;
    private String password;
    private String passwordv;
    private String email;
    private String firstname;
    private String lastname;

    public String getEmail() {
        return email;
    }

     public void setEmail(String email)
{
        this.email = email;
    }

    public String getFirstname() {
        return firstname;
    }

       public  void  setFirstname(String
firstname) {
        this.firstname = firstname;
    }

    public String getLastname() {
        return lastname;
    }

        public  void  setLastname(String
lastname) {
        this.lastname = lastname;
    }

    public String getUsername() {
        return username;
    }
```

```
        public  void  setUsername(String
username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

        public  void  setPassword(String
password) {
        this.password = password;
    }

    public String getPasswordv() {
        return passwordv;
    }

       public  void  setPasswordv(String
passwordv) {
        this.passwordv = passwordv;
    }

    public String validateUser() {
                FacesContext  context  =
FacesContext.getCurrentInstance();
        AppUsers user = getUser();
        if (user != null) {
                                  if   (!
user.getPassword().equals(password)) {
                 FacesMessage message =
new
FacesMessage(FacesMessage.SEVERITY_ERRO
R,
                                   "Login
Failed!",
                           "The password
specified is not correct.");
                context.addMessage(null
, message);
                return null;
            }

             context.getExternalContext(
).getSessionMap().put(USER_SESSION_KEY,
user);
            return "drawing";
        } else {
             FacesMessage message = new
FacesMessage(FacesMessage.SEVERITY_ERRO
R,
                    "Login Failed!",
                    "Username '"
                        + username
                             + "' does not
exist.");
                context.addMessage(null,
message);
            return null;
        }
    }

    public String createUser() {
                FacesContext  context  =
FacesContext.getCurrentInstance();
        AppUsers user = getUser();
        if (user == null) {
                                  if   (!
password.equals(passwordv)) {
                 FacesMessage message =
new     FacesMessage("The     specified
passwords  do  not  match.   Please  try
again");
                 context.addMessage(null
, message);
                return null;
            }
            user = new AppUsers();
             user.setFirstname(firstname
);
            user.setLastname(lastname);
            user.setUsername(username);
            user.setPassword(password);
            try {
                utx.begin();
                em.persist(user);
                utx.commit();
                return "/login";
            } catch (Exception e) {
                 FacesMessage message =
new
FacesMessage(FacesMessage.SEVERITY_ERRO
R,
                          "Error creating
user!",
                              "Unexpected
error   when   creating   your   account.
Please      contact      the      system
Administrator");
                 context.addMessage(null
, message);
                 Logger.getAnonymousLogg
er().log(Level.SEVERE,
                               "Unable to
create new user",
                          e);
```

```
                return null;
            }
        } else {
             FacesMessage message = new
FacesMessage(FacesMessage.SEVERITY_ERRO
R,
                    "Username '"
                    + username
                           + "' already
exists!  ",
                       "Please choose a
different username.");
               context.addMessage(null,
message);
            return null;
        }
    }

    public String logout() {
              HttpSession  session  =
(HttpSession)
FacesContext.getCurrentInstance().getEx
ternalContext().getSession(false);
        if (session != null) {
            session.invalidate();
        }
        return "/index";

    }

    private AppUsers getUser() {
        try {
             AppUsers user = (AppUsers)
em.createNamedQuery("AppUsers.findByUse
rname").
                    setParameter("usern
ame", username).getSingleResult();
            return user;
        } catch (NoResultException nre)
{
            return null;
        }
    }
}
```

Userdetails

```
/*
  *  To  change  this  template,  choose
Tools | Templates
 * and open the template in the editor.
 */
package org.com.login.pack;

/**
 *
 * @author giorgoCh
 */
public class userdetails {

    public userdetails() {
    }

    private String messages;

    public userdetails(String messages)
{
        this.messages = messages;
    }

    public String getMessages() {
        return messages;
    }

       public  void  setMessages(String
messages) {
        this.messages = messages;
    }

}
```