ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Χρήση java τεχνολογιών για την υλοποίηση μιας 3-tier αρχιτεκτονικής παροχής μηχανισμού ελέγχου χρηστών και εκχώρησης δικαιωμάτων πρόσβασης

Του φοιτητή
Σωτήρη Ρέμελη
Αριθμός Μητρώου: 06/3094

Επιβλέπουσα Καθηγήτρια
Κερκίρη Σ.

ΘΕΣΣΑΛΟΝΙΚΗ 2011

.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα ξεκινώντας να ευχαριστήσω τα άτομα που με βοήθησαν στην εκπόνηση αυτής της εργασίας.

- Πρώτα, την κυρία Τάνια Κερκίρη, επιβλέπουσα της εργασίας αυτής. Το ενδιαφέρον της ήταν παραπάνω από συγκινητικό και ειλικρινά, ώρες-ώρες έμοιαζε να αγχώνεται περισσότερο και από εμένα τον ίδιο για την πορεία της πτυχιακής μου. Η ηρεμία που μου μετάγγιζε, οι γνώσεις που μου προσέφερε και ο τρόπος που οργάνωσε τη δουλειά μου έπαιξαν τον πιο καθοριστικό ρόλο στην εξέλιξη του έργου αυτού και αποτέλεσαν ένα μεγάλο σχολείο για μένα.
- Όλους τους καθηγητές μου στο τμήμα Πληροφορικής του ΑΤΕΙΘ καθ' όλη τη διάρκεια των σπουδών μου, από το Σεπτέμβρη του 2006 έως και σήμερα, για τις γνώσεις που μου μετέδωσαν και ακόμα περισσότερο για τη δίψα για μάθηση που μου εμφύσησαν. Λίγο περισσότερο όμως θα ήθελα να ευχαριστήσω τους καθηγητές που μου έδωσαν τη σχετική γνώση για τη συγγραφή της πτυχιακής μου και συγκεκριμένα: τους κυρίους Παναγιώτη Αδαμίδη και Θέμη Κότσιαλο για τη διδασκαλία τους και την αγάπη που μου ενέπνευσαν για τον προγραμματισμό και ειδικά για τη Java, τους κυρίους Δεληγιάννη και Αμπατζόγλου για τη διδασκαλία τους στη Μηχανική Λογισμικού και τον κύριο Δημήτρη Δέρβο ο οποίος είναι ο λόγος που συμπάθησα τις βάσεις δεδομένων.
- Τέλος, δε θα ήμουν καθόλου δίκαιος αν δεν ανέφερα την οικογένειά μου, η οποία στο σύνολό της έκανε όλη τη δέουσα υπομονή ώστε να ολοκληρώσω την πτυχιακή μου. Συγκεκριμένα, τη σύζυγό μου, Μαρία Καρβουνίδου, για την ψυχολογική υποστήριξη που μου προσέφερε και την ώθηση που μου έδινε στις φάσεις που τη χρειαζόμουν και το γιο μου, Αλέξανδρο, για τα βλέμματα, τα λογάκια, τα χαμόγελα και όλα όσα έκανε και με ξεκολλούσαν όταν όλοι οι άλλοι αποτύγχαναν.

ΠΕΡΙΛΗΨΗ

Αυτή η εργασία περιλαμβάνει την ανάπτυξη μίας εφαρμογής διαχείρισης δικαιωμάτων χρηστών καθώς και τη μελέτη των τεχνολογιών υλοποίησής της. Συγκεκριμένα, στο έγγραφο περιγράφονται το πρότυπο MVC σύμφωνα με το οποίο επιβάλλεται ο διαχωρισμός της επεξεργασίας των δεδομένων από την παρουσίαση της πληροφορίας, η αρχιτεκτονική 3-tier η οποία επιτρέπει σε ένα σύστημα να κατανεμηθεί σε τρία φυσικά επίπεδα-στρώματα (δεδομένα-λογική-παρουσίαση), η πλατφόρμα Java 2 Enterprise Edition (J2EE) πάνω στην οποία θα βασιστεί η υλοποίηση του κώδικα της εφαρμογής και το πλαίσιο ιστού Java Serer Faces (JSF) που αποτελεί τμήμα της J2EE και είναι η μετεξέλιξη των Java Server Pages (JSP). Επιπλέον, γίνεται αναφορά σε έξυπνες προγραμματιστικές τεχνικές που διευκολύνουν και επιταχύνουν τη συγγραφή κώδικα και βελτιώνουν δραματικά την απόδοση του συστήματος, όπως η Javascript, τα Cascading Style Sheets (CSS) και η Asynchronous Javascript and XML (AJAX). Δίνεται μία πλήρης ανάλυση για το σχεδιασμό του λογισμικού που αναπτύχθηκε στα πλαίσια της πτυχιακής και λεπτομερής αναφορά στα αρχεία που την απαρτίζουν. Στο τελευταίο κομμάτι αποδεικνύεται πώς η κάθε τεχνολογία που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής επιτέλεσε το σκοπό για τον οποίο επιστρατεύτηκε, κρίνεται και αξιολογείται η εργασία στο σύνολό της και εκφράζονται στόχοι και βλέψεις για μελλοντική δράση πάνω στην εφαρμογή που αναπτύχθηκε.

ABSTRACT

This thesis consists of the development of a user access management system as well as the study of its implementation technologies. Specifically, this document describes the MVC pattern which enforces separation between data processing and data presentation, the 3-tier architecture that allows a system to be distributed into three physical layers (data, logic, view), Java 2Enterprise Edition (J2EE) that is the implementation platform and the Java Server Faces (JSF) web framework which is part of the J2EE platform and is the evolution of the Java Server Pages (JSP). Moreover, this script mentions some smart programming techniques that facilitate code development and boost system performance, such as Javasxript, Cascading Style Sheets (CSS) and Asynchronous Javascript and XML (AJAX). The design of the software developed in the scope of this thesis and the files that compose the application are fully analysed. In the final part of the document every used technology is proven against its intent, the thesis is assessed in its entirety and some aspirations are expressed for future action upon the implemented application.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

1. ΕΙΣΑΓΩΓΗ

Ο ιστός σήμερα αποτελεί το μέσο υποστήριξης των εργασιών και των προσφερόμενων υπηρεσιών όλων σχεδόν των επιχειρήσεων / οργανισμών.

Πλέον πλήθος εφαρμογών που στηρίζονται στον ιστό, διαμοιράζουν δισεκατομμύρια bytes πληροφορίας ανάμεσα σε εκατομμύρια ανθρώπους κατανεμημένους σε όλο τον κόσμο και στηρίζουν την ανάπτυξη ολόκληρου του πλανήτη.

Αυτή η πολυπλοκότητα, από τη μεριά του ειδικού ανάπτυξης εφαρμογών, απαιτεί εξειδικευμένες γνώσεις και ειδικές τεχνικές και τεχνολογίες στη σύλληψη / ανάλυση και υλοποίηση των εφαρμογών αυτών. Από τη μεριά των χρηστών των εφαρμογών, όλη αυτή η πολυπλοκότητα απαιτεί αυξημένες δυνατότητες διαχείρισης. Με άλλα λόγια, απαιτείται η δυνατότητα διαχωρισμού των αρμοδιοτήτων των εργαζομένων που σαφώς θα ορίζει τη δράση τους μέσα στην επιχείρηση / οργανισμό αλλά και θα διευκολύνει την καθημερινή τους ρουτίνα. Επίσης, λόγω της ολοκλήρωσης των πολύπλοκων πληροφοριακών συστημάτων που πλέον υλοποιείται και στη χώρα μας, προκύπτει και η ανάγκη για έλεγχο ή περιορισμό της πρόσβασης πάνω σε ευαίσθητα δεδομένα.

1.1. Περιοχή Έρευνας

Για αυτούς τους πολύ σοβαρούς λόγους προκύπτει **για τον χρήστη,** από τη μια, η ανάγκη μιας εφαρμογής με αυξημένες δυνατότητες στη διαχείριση της πρόσβασής του μέσα στην πληροφορία και στην οριοθέτηση των εργασιών και των δικαιωμάτων του και **για τον προγραμματιστή**, από την άλλη, η εύκολη αναδιάταξη αλλά και επέκταση των δυνατοτήτων του λογισμικού που διαχειρίζεται τις εργασίες της εταιρείας / οργανισμού ανάλογα με τις νέες ανάγκες που θα προκύψουν.

Ευτυχώς πλέον οι σύγχρονες γλώσσες διαδικτυακού προγραμματισμού και οι τεχνολογίες σχεδιασμού τους έχουν ωριμάσει προς την κατεύθυνση αυτή. Έχουν δημιουργηθεί πρότυπα (standards) και έχουν υλοποιηθεί μεθοδολογίες καθώς και καλές τεχνικές προγραμματισμού πρέπει να ακολουθούνται για να διευκολύνουν την υλοποίησή των εφαρμογών του ιστού και να εξασφαλίζουν την ασφάλεια και ταχύτητα εκτέλεσής τους.

Τέτοιες τεχνολογίες είναι η MVC προσέγγιση, η υλοποίηση 3-tier αρχιτεκτονικής, το client-server μοντέλο και η χρήση τεχνολογιών όπως η Java Server Pages και ajax. Αυτές αποτελούν καλές πρακτικές (best practices) και ανεξαρτητοποιούν την

υλοποίηση από την υποστηριζόμενη βάση δεδομένων και την πλατφόρμα υλοποίησης, προσφέροντας ευέλικτο / ευκολοσυντήρητο / επαναχρησιμοποιήσιμο κώδικα.

Εκτός των παραπάνω, οι τεχνολογίες αυτές έχουν το επιπλέον πλεονέκτημα ότι είναι ανοιχτού κώδικα και επίσης διαθέτουν μεγάλη ερευνητική κοινότητα που ανατροφοδοτεί την ανάπτυξή τους.

1.2. Αντικείμενο Πτυχιακής

Σκοπός της πτυχιακής είναι η διερεύνηση και οικειοποίηση των παραπάνω τεχνολογιών, με απτό αποτέλεσμα τη δημιουργία μιας εφαρμογής η οποία υλοποιεί μια 3-tier αρχιτεκτονική χρησιμοποιώντας τεχνολογίες ανοιχτού λογισμικού όπως: java, JSP/JSF και Tomcat [19] σαν εξυπηρετητή ιστού (web-server), προκειμένου να διαχειριστεί ειδικές δυνατότητες / αρμοδιότητες / προσβασιμότητα ομάδων χρηστών πάνω στις επιλογές μιας εφαρμογής.

Η πτυχιακή περιλαμβάνει τη συγγραφή μιας εφαρμογής λογισμικού για τη διαχείριση της πρόσβασης, υλοποιεί μία γενικευμένη προσέγγιση για τη διαχείριση δικαιωμάτων και καταγράφει το θεωρητικό υπόβαθρο των τεχνολογιών / προτύπων και αρχιτεκτονικών που απαιτούνται για την υλοποίησή της.

1.3. Δομή Εγγράφου

Παρακάτω συνοψίζονται τα περιεχόμενα των κεφαλαίων που ακολουθούν.

Στο **κεφάλαιο 2** αναλύονται τα σχεδιαστικά πρότυπα και οι αρχιτεκτονικές υλοποίησης της εφαρμογής καθώς και οι τεχνολογίες λογισμικού που χρησιμοποιήθηκαν. Συγκεκριμένα, γίνεται αναφορά στο πρότυπο MVC και την αρχιτεκτονική τριών επιπέδων βάσει των οποίων χτίστηκε η εφαρμογή, παρουσιάζεται η τεχνολογία JSF με την οποία γράφτηκαν οι ιστοσελίδες της εφαρμογής και το λογισμικό Hibernate ORM που χρησιμοποιήθηκε για αντιστοίχιση αντικειμένων στο σχεσιακό μοντέλο.

Στο **κεφάλαιο 3** αρχικά περιγράφονται οι προδιαγραφές του συστήματος διαχείρισης δικαιωμάτων πρόσβασης, αιτιολογούνται οι αποφάσεις που λήφθηκαν προκειμένου να ικανοποιηθούν οι τρέχουσες ανάγκες αλλά και να δοθεί άμεση δυνατότητα επέκτασης. Ακολουθεί ο σχεδιασμός της παρουσίασης (view) της εφαρμογής, με ιδιαίτερη έμφαση στην ανάλυση των τεχνολογιών CSS, JavaScript, AJAX και των

ειδικευμένων συστατικών (custom components) του προτύπου JSF. Το κεφάλαιο ολοκληρώνεται με το σχεδιασμό του μοντέλου οντοτήτων-συσχετίσεων (ER model).

Στο **κεφάλαιο 4** του εγγράφου αναλύεται η υλοποίηση της εφαρμογής και συγκεκριμένα τα πακέτα της java με τις κλάσεις τους που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής. Περιγράφεται επίσης η δομή των αρχείων της Java, των ιστοσελίδων JSF και των ειδικών αρχείων XML.

Η αξιολόγηση της εργασίας γίνεται στο **5^{ο} κεφάλαιο**. Εδώ αξιολογούνται οι αντικειμενικοί στόχοι της πτυχιακής και παρατίθενται αποδείξεις για το πώς το καθένα από τα πρότυπα, τεχνολογίες και τεχνικές που αναλύονται σε προηγούμενα κεφάλαια βοηθούν στην υλοποίηση αυτού του στόχου.

Στο **έκτο και τελευταίο κεφάλαιο** του εγγράφου γίνονται προτάσεις για το πώς η παρούσα εφαρμογή μπορεί να βελτιωθεί και να επεκταθεί σε υψηλότερο επίπεδο. Εκφράζονται στόχοι και βλέψεις από την βελτίωση της εργασίας που υλοποιείται και εξάγονται συμπεράσματα από την εκπόνησή της.

2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

2.1. Σχεδιαστικά Πρότυπα και Αρχιτεκτονική

Η πτυχιακή αυτή έχει διπλό στόχο:

- από τη μια, την οικειοποίηση των τεχνολογιών υλοποίησης οι οποίες διευκολύνουν στην ανάπτυξη πολύπλοκων εφαρμογών που απαιτούν διευρυμένες δυνατότητες διαχείρισης των αυξημένων απαιτήσεων των χρηστών σε έξυπνες δυνατότητες πρόσβασης
- και από την άλλη, την απόδειξη της οικειοποίησης με την υλοποίηση μιας εφαρμογής η οποία επίσης θα εφαρμόζει καινοτόμες τεχνολογίες.

Για να καλυφθούν οι παραπάνω απαιτήσεις έγινε ενδελεχής μελέτη του προγραμματιστικού προτύπου Μοντέλο-Παρουσίαση-Ελεγκτής (Model-View-Controller - MVC) και της αρχιτεκτονικής τριών επιπέδων (3-tier architecture) και αναπτύχθηκε μια εφαρμογή με αυτά.

Στη συνέχεια, υλοποιήθηκε μια εφαρμογή που εφαρμόζει τις τεχνολογίες αυτές και με επαγγελματικές προδιαγραφές ώστε να παρέχει επιμελής έλεγχος σφαλμάτων με ανάλαφρες τεχνολογίες στο κομμάτι του πελάτη, έλεγχο και τυποποίηση των εντολών sql πριν την αποστολή τους στη βάση δεδομένων, σύγχρονη και ασύγχρονη επικοινωνία με τον διακομιστή ιστού για επιτάχυνση των εργασιών και ελαχιστοποίηση της διακίνησης πληροφορίας, με JSP τεχνολογία για ανεξαρτητοποίηση της εμφάνισης από τα ίδια τα δεδομένα και τους επιχειρησιακούς κανόνες (business rules) της εφαρμογής.

Το αποτέλεσμα της εργασίας είναι ένα αυτόνομο λογισμικό που μπορεί να εγκατασταθεί σαν πρόσθετο σε διάφορες εφαρμογές βασισμένες σε java και να επιτελέσει τον έλεγχο πρόσβασης και χωρισμού επιπλέον δραστηριοτήτων των εφαρμογών αυτών στους χρήστες τους.

Στο κεφάλαιο αυτό θα παρουσιαστούν αυτές οι τεχνικές και τεχνολογίες που αποτελούν το απαραίτητο θεωρητικό υπόβαθρο με το οποίο γίνεται κατανοητή η παρούσα πτυχιακή.

2.1.1. Πρότυπο Model-View-Controller (MVC)

Με την έναρξη της ψηφιακής εποχής η πληροφορία έγινε το πιο πολύτιμο αγαθό. Με την πάροδο του χρόνου, οι ανάγκες των οργανισμών για επεξεργασία της και

αποτελεσματική διαχείρισή της έγιναν επιτακτικές κι εξεζητημένες. Αναδείχτηκαν ανάγκες όπως: ταχεία επεξεργασία της, παρουσίασή της σε πολλαπλά μέσα κ.λπ.

Τους λόγους αυτούς έρχεται να καλύψει ο διαχωρισμός της επεξεργασίας και διαχείρισης των δεδομένων (δηλαδή οι **κανόνες επιχειρησιακής λογικής** με τους οποίους τα δεδομένα διαχειρίζονται) από την προβολή τους.

Η **επιχειρησιακή λογική** είναι ένα σύνολο από κανόνες και αλγορίθμους υπεύθυνους για την ανταλλαγή πληροφοριών μεταξύ της βάσης δεδομένων και της οθόνης του χρήστη. Περιλαμβάνει όλες τις διαδικασίες που εκτελούνται στο παρασκήνιο και είναι αόρατα στο χρήστη, όπως τα ερωτήματα δημιουργίας, ανάκτησης, επεξεργασίας και διαγραφής δεδομένων καθώς και άλλες εξειδικευμένες λειτουργίες ανάλογα με την εφαρμογή και τα δεδομένα που διαχειρίζεται. Πρόκειται δηλαδή για το παρασκηνιακό κομμάτι ενός συστήματος, σε αντίθεση με την παρουσίαση των δεδομένων που αποτελεί και την όψη του συστήματος.

Στα συστήματα διαχείρισης πληροφορίας είναι ζωτικής σημασίας η ανάκτηση δεδομένων από αποθήκες δεδομένων και η παρουσίασή τους στο χρήστη. Μετά τη διαχείριση / μεταβολή των δεδομένων από το χρήστη, το σύστημα ενημερώνει τη βάση δεδομένων. Επειδή η κύρια ροή πληροφοριών είναι αυτή μεταξύ της βάσης δεδομένων και της διεπαφής της εφαρμογής, είναι λογικό να σκεφτεί κανείς να δέσει αυτά τα δύο τμήματα ώστε να μειωθεί ο απαιτούμενος κώδικας και να βελτιωθεί η απόδοση της εφαρμογής. Παρόλα αυτά, αυτή η φαινομενικά λογική προσέγγιση παρουσιάζει σημαντικά προβλήματα:

- Η διεπαφή χρήστη (user interface - UI) έχει την τάση να αλλάζει πολύ πιο γρήγορα από τα συστήματα αποθήκευσης δεδομένων. Όταν συνδέονται η επιχειρησιακή λογική με την παρουσίαση, κάθε αλλαγή στη διεπαφή χρήστη επιφέρει αναπόφευκτες αλλαγές και στον κώδικα διαχείρισης των δεδομένων, οδηγώντας σε πιθανά σφάλματα και άσκοπες νέες δοκιμές.
- Η διεπαφή χρήστη έχει πιο ισχυρή εξάρτηση από τις συσκευές στις οποίες θα εμφανίζεται. Αν δε διαχωρίζεται η λογική από την παρουσίαση, η μετατροπή, για παράδειγμα, μίας εφαρμογής PC σε εφαρμογή για κινητή συσκευή, απαιτεί την επανεγγραφή όλου του κώδικα.
- Σε πολλές περιπτώσεις, απαιτείται οι εφαρμογές να παρουσιάζουν τα δεδομένα σε διάφορες μορφές. Αυτό μπορεί να χρειάζεται να γίνει είτε επειδή τα δεδομένα της εφαρμογής θα παρουσιαστούν σε διαφορετικές ομάδες

χρηστών (π.χ. από τη μία οι μέτοχοι μίας μεγάλης εταιρίας, με μικρό ή μηδαμινό διοικητικό υπόβαθρο, και από την άλλη τα υψηλόβαθμα εκτελεστικά στελέχη της εταιρίας), είτε γιατί η εφαρμογή απευθύνεται πρακτικά σε όλους (μία ιστοσελίδα για παράδειγμα), οπότε, για να είναι φιλική προς το χρήστη, πρέπει να του επιτρέπει την επέμβαση σε βασικά στοιχεία της εμφάνισης, όπως χρώμα και μέγεθος γραμματοσειράς, ή ίσως ακόμα και παρουσίαση αριθμητικών δεδομένων με γραφικό τρόπο (διαγράμματα), εφαρμόζοντας το γνωστό ως *tailoring*.

- Η σχεδίαση ελκυστικών ιστοσελίδων απαιτεί εντελώς διαφορετικές ικανότητες και γνώσεις από τη συγγραφή του κώδικα που υλοποιεί τη λογική της εφαρμογής. Εφόσον σπάνια συναντάμε ένα πρόσωπο που να συνδυάζει και τα δύο, καλύτερα να διαχωρίζεται το ένα κομμάτι από το άλλο.
- Η συγγραφή αυτόματων δοκιμών για τη διεπαφή χρήστη είναι πιο χρονοβόρα και δυσκολότερη από τις αντίστοιχες δοκιμές πάνω στον κώδικα της επιχειρησιακής λογικής. Επομένως, η μείωση του κώδικα που αφορά στη διεπαφή χρήστη διευκολύνει τις δοκιμές και εξοικονομεί χρόνο.

Είναι φανερό λοιπόν ότι έπρεπε να βρεθεί τρόπος να ανεξαρτητοποιηθεί η λειτουργικότητα από τη λογική της παρουσίασης ώστε να είναι εύκολη η τροποποίηση των ατομικών στοιχείων.

Τη λύση παρέχει το πρότυπο *Model-View-Controller (MVC)*.

Το MVC είναι ένας τρόπος διάσπασης μίας εφαρμογής, ή ενός τμήματός της, σε τρία μέρη: το *model (μοντέλο)*, το *view (όψη)* και τον *controller (ελεγκτής)*. Τα αντικείμενα αυτά χειρίζονται αντίστοιχα τη μοντελοποίηση του εξωτερικού περιβάλλοντος, την οπτική απεικόνιση (έξοδο) και την είσοδο του χρήστη. Το πρότυπο MVC αναφέρεται στη διάκριση της αποθήκευσης και διαχείρισης των **δεδομένων** (*model*) από την **παρουσίασή** τους (*view*) και τις **ενέργειες** που πυροδοτούνται από τον χρήστη (*controller*). Εισηγήθηκε από τον Trygve Reenskaug της Smalltalk και την ομάδα του, όπου εφαρμόσθηκε για να ενθυλακώσει τα δεδομένα μαζί με τις διαδικασίες επεξεργασίας τους (το model) και να τα απομονώσει από το χειρισμό τους (controller) και την παρουσίασή τους (view) σε κάποια διεπαφή χρήστη (**user interface**) [2].

Συνοπτικά, το μοντέλο διαχειρίζεται τη συμπεριφορά και τα δεδομένα της εφαρμογής, απαντά σε αιτήσεις για ενημέρωση σχετικά με την κατάστασή του (συνήθως προέρχονται από την όψη) και απαντά σε εντολές για αλλαγή της κατάστασής του (προερχόμενες συνήθως από τον ελεγκτή). Η όψη ασχολείται μόνο με την εμφάνιση της πληροφορίας. Ο ελεγκτής ερμηνεύει την είσοδο του χρήστη από οποιαδήποτε συσκευή εισόδου και πληροφορεί το μοντέλο και την όψη ώστε να αλλάξουν την κατάστασή τους ανάλογα. [7]

2.1.1.1. Model (Μοντέλο)

Τα δεδομένα που σχετίζονται με ένα πρόβλημα περιγράφονται αφαιρετικά με τη βοήθεια ενός εννοιολογικού μοντέλου από το οποίο προκύπτει το φυσικό μοντέλο δεδομένων, όπως είναι για παράδειγμα το μοντέλο οντοτήτων-συσχετίσεων (ER model) που οργανώνει τα δεδομένα σε πίνακες.

Ωστόσο, όσον αφορά στο MVC πρότυπο, ένα μοντέλο δεν αναφέρεται μόνο στη φυσική οργάνωση, διάταξη και αποθήκευση των δεδομένων σε βάσεις, αλλά περιέχει και όλη την απαραίτητη λειτουργικότητα που σχετίζεται με τα εν λόγω δεδομένα, άρα περιγράφει τον τρόπο λειτουργίας του συστήματος, ενθυλακώνοντας παράλληλα την τρέχουσα κατάστασή του.

Επομένως, το μοντέλο είναι υπεύθυνο για την **αφαιρετική απεικόνιση της περιοχής προβλήματος** και την πλήρη διαχείριση (αποθήκευση, οργάνωση και επεξεργασία) των δεδομένων και την ενημέρωση των χρηστών του συστήματος όταν αυτά αλλάζουν. Όσο για την **τεχνική της αφαίρεσης**, αυτή επιτυγχάνει μία γενίκευση η οποία βοηθά τον προγραμματιστή να επικεντρωθεί σε αυτά που πραγματικά έχουν σημασία και όχι σε τεχνικές λεπτομέρειες. Επίσης, αυτή η γενίκευση οδηγεί στην ανάπτυξη επαναχρησιμοποιήσιμου κώδικα, καθώς τα μοντέλα δύο διαφορετικών προβλημάτων μπορεί τελικά να μην είναι τόσο ανόμοια.

Όσον αφορά για το πόσα μοντέλα μπορεί να αναπτυχθούν κατά το σχεδιασμό ενός συστήματος, αρκεί να ειπωθεί ότι ένα μοντέλο περιέχει μόνο δεδομένα και λειτουργικότητα που σχετίζονται με τον ίδιο σκοπό. Αυτό έχει σαν συνέπεια ότι αν χρειαστεί η μοντελοποίηση δύο ομάδων από ανόμοια δεδομένα, τότε πρέπει να δημιουργηθούν δύο ξεχωριστά μοντέλα, καθένα από τα οποία θα αναπαριστά και μία ομάδα.

Ακόμα κι έτσι όμως, δεν είναι απαραίτητο ότι ένα μοντέλο μπορεί να παρουσιαστεί με έναν μόνο τρόπο.

2.1.1.2. View (Όψη/Παρουσίαση)

Η όψη ή παρουσίαση (view) είναι υπεύθυνη για το σχεδιασμό γραφικών σε μία συσκευή εξόδου (συνήθως μία οθόνη). Μία όψη έχει τυπικά σχέση ένα-προς-ένα με μία επιφάνεια παρουσίασης και γνωρίζει πώς να εμφανιστεί σε αυτήν. Προσκολλάται σε ένα μοντέλο και εμφανίζει τα περιεχόμενά του στην επιφάνεια παρουσίασης.

Επίσης, όταν το μοντέλο αλλάζει, η παρουσίαση αυτόματα επανασχεδιάζει το επηρεασμένο κομμάτι της εικόνας ώστε να αποδώσει την αλλαγή.

Επιπλέον, μία όψη μπορεί να είναι σύνθετη: μπορεί δηλαδή να περιλαμβάνει αρκετές υπο-όψεις, η καθεμία από τις οποίες μπορεί να περιλαμβάνει με τη σειρά της άλλες υπο-όψεις κ.λπ.

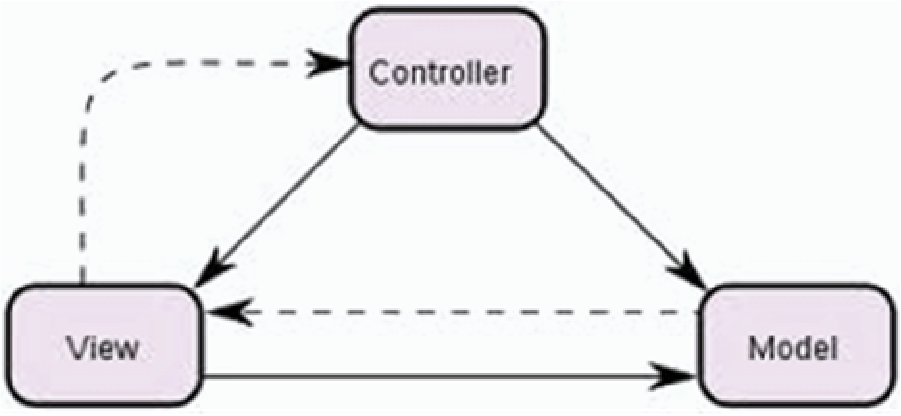
Για τους λόγους που παρουσιάστηκαν παραπάνω μπορεί να συνυπάρχουν πολλές όψεις στο ίδιο μοντέλο και καθεμία από αυτές να εμφανίζει τα περιεχόμενα του μοντέλου σε διαφορετική επιφάνεια παρουσίασης.

2.1.1.3. Controller (Ελεγκτής)

Ο ελεγκτής είναι το μέσο με το οποίο ο χρήστης αλληλεπιδρά με την εφαρμογή. Ένας ελεγκτής δέχεται είσοδο από το χρήστη και αναθέτει στα αντικείμενα του μοντέλου και της παρουσίασης να εκτελέσουν ενέργειες βάσει αυτής της εισόδου. Ο ελεγκτής είναι υπεύθυνος για την αντιστοίχιση μίας ενέργειας του τελικού χρήστη με την κατάλληλη απόκριση της εφαρμογής. Τελικά, ο κώδικας της επιχειρησιακής λογικής εκτελείται, τα δεδομένα τροποποιούνται και η αλλαγή αντανακλάται στην οθόνη του χρήστη [8].

2.1.1.4. Εξάρτηση model, view, controller

Η σχέση των 3 αυτών συστατικών απεικονίζεται στην Εικόνα 1, από όπου συνάγεται ότι η όψη και ο ελεγκτής εξαρτώνται από το μοντέλο, αλλά το μοντέλο δεν εξαρτάται



Εικόνα 1 - Εξαρτήσεις μεταξύ model, view και controller

ούτε από την όψη ούτε από τον ελεγκτή. Αυτό αποτελεί το μείζον όφελος του διαχωρισμού, καθώς επιτρέπει στο μοντέλο να δημιουργείται και να δοκιμάζεται ανεξάρτητα από την οπτική παρουσίαση.

Η διάκριση μεταξύ όψης και ελεγκτή σε πολλές περιπτώσεις δεν είναι εύκολο να εντοπιστεί. Σε ορισμένες εφαρμογές, σε αντίθεση με όσα προστάζει το MVC, η όψη και ο ελεγκτής επικαλύπτονται και οι ρόλοι αυτοί υλοποιούνται σε ένα αντικείμενο το οποίο είναι υπεύθυνο και για τον έλεγχο της εισόδου του χρήστη και για την παρουσίαση του περιεχομένου. Ωστόσο, στις εφαρμογές ιστού, η διάκριση μεταξύ της όψης, η οποία παρουσιάζεται σε ένα παράθυρο φυλλομετρητή, και του ελεγκτή, που υλοποιείται στην πλευρά του διακομιστή και χειρίζεται τα αιτήματα HTTP, είναι σαφής.

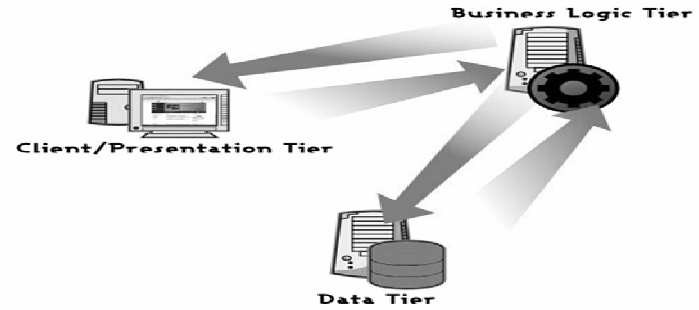
2.1.2. Αρχιτεκτονική 3 επιπέδων (3-tier architecture)

Η αρχιτεκτονική 3 επιπέδων είναι ένας εξελιγμένος τύπος της αρχιτεκτονικής πελάτη-διακομιστή (client-server) η οποία διαχωρίζει σε φυσικό επίπεδο την υποδομή του συστήματος σε στρώματα (ή επίπεδα), συνήθως τα εξής τρία:

- το **στρώμα της παρουσίασης**,
- το **στρώμα της λογικής** (ή εφαρμογής, ή, αλλιώς, πρόσβασης δεδομένων)
- και το **στρώμα των δεδομένων**.

Στο κατώτερο στρώμα (των δεδομένων) γίνεται η διαχείριση του συστήματος αποθήκευσης των δεδομένων (π.χ. βάσεων δεδομένων), στο μεσαίο στρώμα (της λογικής) βρίσκονται οι διακομιστές που εκτελούν τον κατάλληλο κώδικα πάνω σε αυτά τα δεδομένα για να τα παρουσιάσουν στο ανώτερο στρώμα, αυτό της παρουσίασης.

Το κάθε στρώμα μπορεί να υλοποιηθεί σε διαφορετικό φυσικό μέσο (υπολογιστή) από τα άλλα και μάλιστα σε πολλές περιπτώσεις ένα στρώμα μπορεί να είναι μοιρασμένο σε περισσότερα του ενός μέσα (Εικόνα 2).



Εικόνα 2 - Τα τρία επίπεδα μίας 3-tier εφαρμογής τρέχουν σε ξεχωριστές μηχανές

2.1.2.1. Η λειτουργία της 3-tier αρχιτεκτονικής

Στην 3-tier αρχιτεκτονική, η λογική της εφαρμογής, σχεδόν στο σύνολό της, εκτελείται σε έναν ενδιάμεσο διακομιστή, το διακομιστή εφαρμογών (application server) και απομένει ένα πολύ μικρό μέρος επιχειρησιακής λογικής στον πελάτη. Ο πελάτης απευθύνει τις αιτήσεις του για δεδομένα στο διακομιστή εφαρμογών και εκείνος τα διαβιβάζει στο DBMS που χειρίζεται τα δεδομένα της βάσης. Στη συνέχεια, τα δεδομένα ακολουθούν αντίστροφη πορεία: αφού τα ανασύρει από τη βάση, το DBMS τα στέλνει στο διακομιστή, όπου γίνεται η επεξεργασία τους και από εκεί καταλήγουν στον πελάτη του τελικού χρήστη για μορφοποίηση και προβολή.

Στο μοντέλο αυτό ο πελάτη δεν επικοινωνεί **ποτέ** απευθείας με το DBMS. Το σύνολο των δεδομένων που ανταλλάσσονται μεταξύ αυτών των δύο επιπέδων περνά πάντα από τον ενδιάμεσο διακομιστή εφαρμογών.

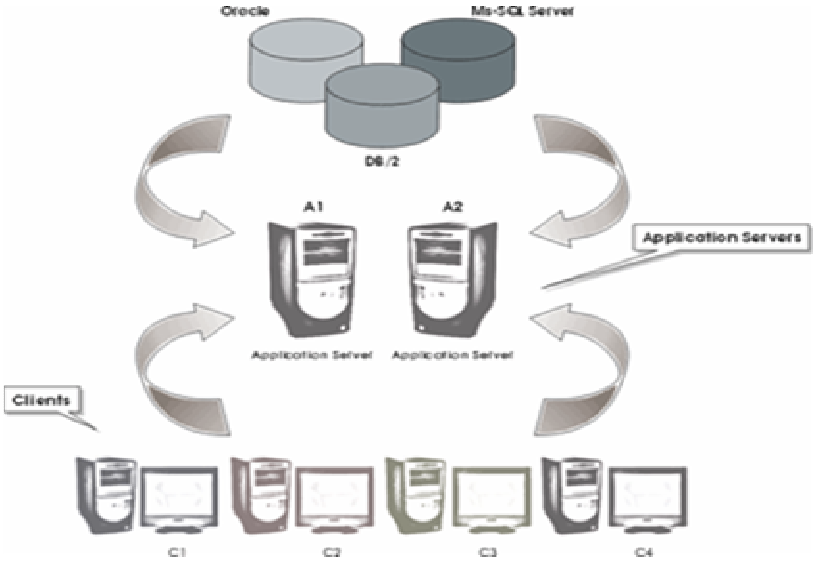
Όπως εύκολα συνεπάγεται από τις περιγραφές του MVC και της 3-tier αρχιτεκτονικής, η 3-tier αρχιτεκτονική δένει απόλυτα με το μοντέλο MVC. Στο πρώτο (υψηλότερο) επίπεδο υλοποιείται η όψη (view), στο δεύτερο επίπεδο ο ελεγκτής (controller) και στο τρίτο το μοντέλο.

2.1.2.2. Πρώτο Επίπεδο - Διακομιστής Δεδομένων

Ο διακομιστής δεδομένων (*data server)* περιέχει και διαχειρίζεται τα δεδομένα. Εδώ είναι εγκατεστημένο το DBMS, το οποίο παρέχει όλους τους απαραίτητους μηχανισμούς για την αποθήκευση, ενημέρωση, συντήρηση και ανάκτηση των δεδομένων. Να σημειωθεί ότι για λόγους ταχύτητας, αξιοπιστίας αλλά και όγκου δεδομένων, μπορούν να χρησιμοποιηθούν όχι ένας αλλά πολλοί διακομιστές δεδομένων και μάλιστα με διαφορετικά DBMS εγκατεστημένα στον καθένα.

2.1.2.3. Δεύτερο Επίπεδο - Διακομιστής Εφαρμογών

Το μεσαίο στρώμα, γνωστό και ως διακομιστής εφαρμογών (*application server*), παρέχει την επιχειρησιακή λογική και τον κώδικα για πρόσβαση των δεδομένων. Αποτελεί το κύριο τμήμα του λογισμικού, στο οποίο εκτελούνται οι περισσότερες λειτουργίες, εκτός εκείνων που σχετίζονται με τη διαμόρφωση των οθονών εργασίας. Όμοια με τους διακομιστές δεδομένων, στις μεγάλες εφαρμογές χρησιμοποιούνται περισσότεροι του ενός διακομιστές εφαρμογών σε διαφορετικά μηχανήματα, αξιοποιώντας, με τον τρόπο αυτό, οποιαδήποτε διαθέσιμη υπολογιστική ισχύ (Εικόνα 3).



Εικόνα 3 - Χρήση δύο application servers

Επίσης, με την κατανομή των διακομιστών εφαρμογών σε ανεξάρτητα μηχανήματα, **επιτυγχάνεται αποσυμφόρηση του συνολικού φόρτου του συστήματος**, αφού καθένας από αυτούς είναι σε θέση να υποστηρίξει ένα μέρος του συνόλου των απομακρυσμένων πελατών.

2.1.2.4. Τρίτο Επίπεδο - Πελάτης

Ο πελάτης (client) περιέχει μόνο τη λογική της παρουσίασης καθώς επίσης ελέγχους και επικυρώσεις της εισόδου του χρήστη. Αποτελεί το σημείο επαφής του χρήστη με το σύστημα μέσω της διεπαφής χρήστη και είναι το μόνο κομμάτι που αλληλεπιδρά με τον τελικό χρήστη. Στο επίπεδο αυτό, πραγματοποιείται η διαχείριση των οθονών εργασίας καθώς επίσης και η μορφοποίηση των δεδομένων που εμφανίζονται [9].

2.1.2.5. Πλεονεκτήματα της αρχιτεκτονικής επιπέδων

Με την 3-tier αρχιτεκτονική προκύπτουν σημαντικά πλεονεκτήματα.

- Η τροποποίηση ή αλλαγή ολόκληρου στρώματος γίνεται χωρίς να επηρεάζονται τα υπόλοιπα στρώματα.
- Ο διαχωρισμός της λειτουργικότητας της εφαρμογής από αυτήν της βάσης δεδομένων έχει ως αποτέλεσμα καλύτερη κατανομή φόρτου.
- Ευελιξία και ανεξαρτησία στην επιλογή του διακομιστή δεδομένων, αφού μπορούν να επιλεγούν όλοι οι τύποι συστημάτων διαχείρισης βάσεων δεδομένων (Oracle, SQL Server, MySQL, PostgeSQL, DB2 κλπ), σε

οποιοδήποτε λειτουργικό σύστημα (αρκεί βέβαια το λειτουργικό σύστημα να παρέχει τη δυνατότητα επικοινωνίας μέσω του πρωτοκόλλου TCP/IP).

- Μπορούμε να εφαρμόσουμε οποιεσδήποτε πολιτικές ασφαλείας στα επίπεδα των διακομιστών χωρίς να επιβαρυνθούν καθόλου οι πελάτες [5].

2.2. Λογισμικό υλοποίησης

Οι τεχνολογίες που περιγράφηκαν παραπάνω υποστηρίζονται από τις πλέον γνωστές και ευρέως διαδεδομένες γλώσσες προγραμματισμού και πλατφόρμες, όπως είναι η Java της Oracle [14] και το .NET Framework της Microsoft [12]. Απεριόριστες δυνατότητες προσφέρουν τα JSF [15], τα οποία είναι ένα πλαίσιο ιστού (web framework) οδηγούμενο από αιτήματα του χρήστη (request-driven). Η υποστήριξη ανάπτυξης που παρέχεται από ολοκληρωμένα περιβάλλοντα (IDEs), όπως είναι το NetBeans [13] και το Eclipse για Java και το Visual Studio για .NET, βελτιώνουν την εμπειρία της συγγραφής του κώδικα και επιταχύνουν τη διαδικασία της ανάπτυξης.
Η εφαρμογή βασίζεται εξ ολοκλήρου σε τεχνολογίες ανοιχτού κώδικα και συγκεκριμένα:

- **Java Enterprise Edition 6 (J2EE)** για την υλοποίηση της λογικής της εφαρμογής. Η J2EE είναι η έκδοση της Java για server programming, εμπλουτισμένη σε σχέση με τη Java SE με APIs για διαχείριση βάσεων δεδομένων (JDBC), απομακρυσμένες κλήσεις (RMI και web services), υπηρεσίες μηνυμάτων (JMS), e-mail, XML και άλλα.. Με δεδομένο ότι στην υλοποίηση της εφαρμογής χρησιμοποιήθηκαν μόνο τεχνολογίες ανοιχτού κώδικα, η επιλογή της Java ως γλώσσα υλοποίησης ήταν πρακτικά επιβεβλημένη.
    - **Glassfish v3.1 ως Web Container**: από τους πλέον διαδεδομένους application servers ανοιχτού κώδικα για Java, επιτρέπει την πλήρη εκμετάλλευση των δυνατοτήτων που προσφέρει η J2EE. **Επιλέχθηκε ο Glassfish Application Server με γνώμονα** τη πολύ καλή υποστήριξη της γλώσσας προγραμματισμού Java.
    - **MySQL 5.5.6 ως DBMS**: το πιο διαδεδομένο RDBMS λογισμικό ανοιχτού κώδικα. Διαθέτει εγκαταστάσεις σε linux και windows και ικανοποιεί τις ανάγκες υλοποίησης της εργασίας αυτής. Παρόλο που το JDBC θα επέτρεπε τη χρήση οποιουδήποτε λογισμικού διαχείρισης βάσεων δεδομένων,

επιλέχθηκε ο MySQL Server, λόγω του ότι είναι το πιο διαδεδομένο DBMS ανοιχτού κώδικα.

- **Java Server Faces (JSF) 2.03 RI:** κομμάτι της J2EE που χρησιμοποιείται για την παρουσίαση των σελίδων. Επιλέχτηκαν έναντι άλλων τεχνολογιών όπως π.χ. η JSP λόγω του ότι είναι πλέον standard.
  Αναλύονται διεξοδικά παρακάτω.
- **NetBeans IDE 7.0:** εργαλείο για την ανάπτυξη της εφαρμογής. Το NetBeans προσφέρει πλήρη ολοκλήρωση της J2EE και των JSF μέσα στο περιβάλλον του και σε πολλά σημεία μπορεί να παράγει αυτόματα τον κοινά χρησιμοποιούμενο κώδικα.

Στη συνέχεια θα περιγραφούν οι τεχνολογίες και τα πρότυπα αυτά.

2.2.1. Java Server Faces (JSF)

Στην παρουσίαση των δεδομένων και την επικοινωνία με το χρήστη θα μπορούσαν να χρησιμοποιηθούν είτε απλές JSP σελίδες είτε μία πιο καινούρια τεχνολογία με πολύ περισσότερες δυνατότητες από τα JSP και αυτή δεν ήταν άλλη από τα JSF. Η πλήρης ενσωμάτωση των JSF στην J2EE είναι η καλύτερη απόδειξη για την καταλληλότητα της επιλογής αυτής.

Τα JSF υλοποιούνται πλέον με τη δοκιμασμένη πλατφόρμα των Java Server Pages (JSP), βασίζονται στο σαφή διαχωρισμό της λογικής της εφαρμογής και της παρουσίασης των ιστοσελίδων και ταυτόχρονα διευκολύνουν τη σύνδεση του στρώματος της παρουσίασης με τον κώδικα της εφαρμογής. Αυτή η τεχνική επιτρέπει σε κάθε μέλος της ομάδας ανάπτυξης να επικεντρωθεί στο δικό του κομμάτι και παρέχει ένα προγραμματιστικό μοντέλο για τη συνένωση αυτών των κομματιών.

Η JSF τεχνολογία περιλαμβάνει ένα σύνολο από APIs τα οποία είναι υπεύθυνα για:

- την αναπαράσταση και τη διαχείριση της κατάστασης των UI components,
- το χειρισμό γεγονότων και την επικύρωση της εισόδου του χρήστη (user input),
- τον ορισμό της πλοήγησης ανάμεσα στις ιστοσελίδες και
- την υποστήριξη i18n (πολυγλωσικής υποστήριξης - internationalization) και προσβασιμότητας.

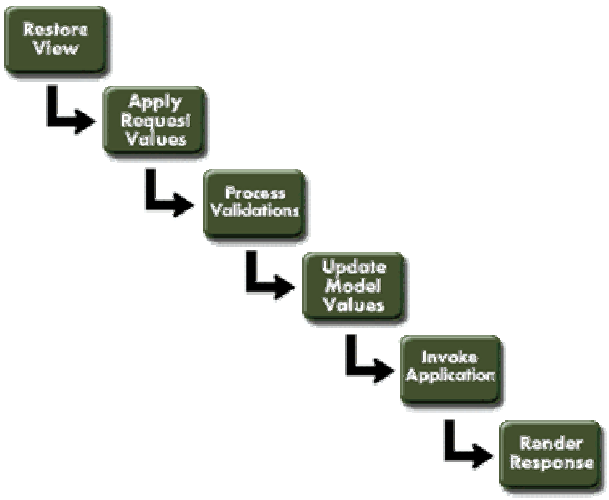
Όλα αυτά γίνονται εφικτά μέσα από μία κατάλληλη βιβλιοθήκη (tag library) που αποτελεί το λεξιλόγιο για την υλοποίηση των συστατικών των JSF μέσα σε μία σελίδα JSP.

Τα JSF δεν χρησιμοποιούν πλέον tags εναλλαγής από συγγραφή HTML σε JSP (<%...%>) και χωρίς να απαιτούν από τον προγραμματιστή την ανάπτυξη καμιάς κλάσης java για επεξεργασία εξόδου HTML (servlet), κάνουν τη συγγραφή ιστοσελίδων ακόμη πιο εύκολη και ενδιαφέρουσα.

Τα συστατικά μίας σελίδας JSF (UI components) αντί να ενθυλακώνουν την παρουσίαση και εμφάνισή τους σε κάθε πελάτη ξεχωριστά, περιγράφουν τη συμπεριφορά και λειτουργικότητά τους, απελευθερώνοντας έτσι τον προγραμματιστή από περιορισμούς συσκευής, αφού το ίδιο component θα χρησιμοποιηθεί και για εφαρμογή ιστού και για κινητή εφαρμογή [16].

2.2.1.1. Ο κύκλος ζωής μίας JSF σελίδας

Κατά τη διάρκεια ζωής της, μία σελίδα JSF περνάει σειριακά από τις παρακάτω έξι φάσεις : restore view, apply request values, process validations, update model values, invoke application και render response. Στην Εικόνα 4 φαίνονται αυτές οι φάσεις.



Εικόνα 4 - Ο κύκλος ζωής των JSF

1. ***Restore View***: Μόλις ληφθεί αίτημα (request) για μία σελίδα, τα JSF ξεκινούν τη φάση restore view. Σε αυτή τη φάση δημιουργείται η όψη (view) της σελίδας, συνδέονται όλοι οι listeners (event handlers, validators κλπ) με τα αντίστοιχα συστατικά της σελίδας που είναι γραμμένα σε JSF (components) και αποθηκεύεται ένα στιγμιότυπο της κλάσης `javax.faces.context.FacesContext`. Το αντικείμενο αυτό περιέχει όλες τις απαραίτητες πληροφορίες για την επεξεργασία του αιτήματος. Εάν πρόκειται για αρχικό αίτημα (δηλαδή αν καλούμε για πρώτη φορά τη σελίδα, άρα η όψη δεν έχει δημιουργηθεί), τα JSF δημιουργούν μία άδεια όψη και ο

κύκλος συνεχίζει από τη φάση render response, αφού δεν υπάρχει είσοδος χρήστη για να επεξεργαστούν τα JSF. Εάν όμως το αίτημα είναι τύπου επανάκλησης (postback), υπάρχει ήδη μία όψη που αντιστοιχεί στη σελίδα αυτή, οπότε τα JSF την ανακτούν βάσει των πληροφοριών που υπάρχουν στον πελάτη ή/και στο διακομιστή.

2. ***Apply Request Values***: Αφού ανακτηθούν τα συστατικά της σελίδας, καθένα από αυτά εξάγει τη νέα του τιμή από τις παραμέτρους της αίτησης και αποθηκεύει τη νέα του τιμή τοπικά. Εάν υπάρχουν events στην ουρά του στιγμιοτύπου FacesContext, τα JSF τα προωθούν στους αντίστοιχους listeners και εάν έστω και ένας listener καλέσει τη μέθοδο `renderResponse` του τρέχοντος στιγμιοτύπου FacesContext, τότε μεταφερόμαστε στη φάση render response.
3. ***Process Validations***: Εδώ τα JSF εκτελούν τους validators, οι οποίοι δεν κάνουν άλλο από το να επικυρώνουν τις νέες τιμές των συστατικών της σελίδας, με άλλα λόγια την είσοδο του χρήστη. Εάν προκύψει σφάλμα επικύρωσης, τα JSF μεταπηδούν στη φάση render response και εμφανίζουν το αντίστοιχο μήνυμα λάθους.
4. ***Update Model Values***: Αφού επικυρώσουν τις νέες τιμές, τα JSF μπορούν πλέον να αντιστοιχίσουν τις τοπικές τιμές του κάθε συστατικού με τα αντικείμενα στο διακομιστή (server-side objects). Σε περίπτωση που κάποια αντιστοίχιση είναι άκυρη και δεν έχει «πιαστεί» κατά τη φάση process validations, μεταφερόμαστε στη φάση render response όπου και εμφανίζεται κατάλληλο μήνυμα σφάλματος.
5. ***Invoke Application***: Σε αυτό το στάδιο, τα JSF χειρίζονται events επιπέδου εφαρμογής (application-level events), όπως είναι η υποβολή μίας φόρμας ή η ανακατεύθυνση σε άλλη σελίδα.
6. ***Render Response***: Πλέον, τα JSF είναι έτοιμα να μεταβιβάσουν την αρμοδιότητα για την εμφάνιση των σελίδων στον JSP container (πρακτικά ένα servlet που μεταφράζει τη JSP σύνταξη σε HTML). Εάν πρόκειται για αρχικό αίτημα, τα συστατικά που περιέχονται στη σελίδα προστίθενται στο δέντρο συστατικών (component tree) καθώς ο JSP container εκτελεί τη σελίδα και τα components εμφανίζονται. Αφού το περιεχόμενο της όψης εμφανιστεί, η κατάσταση της απάντησης (response) αποθηκεύεται για να μπορεί να

ξαναχρησιμοποιηθεί κατά τη φάση restore view μεταγενέστερων αιτημάτων [10].

2.2.2. Object-relational Mapping (ORM)

Το ORM είναι μία τεχνική για τη μετατροπή δεδομένων μεταξύ μίας αντικειμενοστραφούς γλώσσας και ενός ασύμβατου συστήματος Στην ουσία δημιουργείται μία εικονική αντικειμενοστρεφής βάση δεδομένων, η οποία είναι προσβάσιμη μέσω της γλώσσας. Τα εργαλεία που διατίθενται για το σκοπό αυτό είναι είτε εμπορικά είτε ελεύθερα. Στην παρούσα εργασία χρησιμοποιήθηκε ένα ευρέως διαδεδομένο εργαλείο ORM, το Hibernate. Η επιλογή αυτή έγινε επειδή το Hibernate γράφτηκε ειδικά για αντιστοίχιση αντικειμένων Java στο ER μοντέλο και αντίστροφα.

2.2.2.1. Hibernate

Το Hibernate είναι ένα εργαλείο ανοιχτού λογισμικού ORM για τη μετατροπή δεδομένων σε αντικείμενα της Java και το αντίστροφο. Η μετατροπή μπορεί να γίνει με δύο τρόπους· είτε με τη χρήση ενός ειδικού XML αρχείου, είτε με *Java annotations*. Στην πρώτη περίπτωση, ορίζονται αντιστοιχίες (mappings) πινάκων της βάσης δεδομένων με κλάσεις της Java, ενώ στη δεύτερη η αντιστοίχιση δηλώνεται μέσα στον κώδικα της κλάσης (πάνω ακριβώς από τη δήλωση της κλάσης και κάθε μεταβλητής/μέλους της) με σχόλιο (annotation) ειδικής σύνταξης, όπου αναφέρεται το όνομα του αντίστοιχου πίνακα ή πεδίου στη βάση.

Το Hibernate παρέχει ένα σύνολο από ευκολίες για δημιουργία ερωτημάτων και ανάκτηση δεδομένων. Με τον τρόπο αυτό απελευθερώνει τον προγραμματιστή από τη χειροκίνητη διαχείριση των ομάδων αποτελεσμάτων (result sets) και τη μετατροπή δεδομένων από σχεδόν οποιοδήποτε σχεσιακό DBMS στο αντικειμενοστρεφές μοντέλο, διατηρώντας ταυτόχρονα τη μεταφερσιμότητα της εφαρμογής σε υψηλά επίπεδα.

ΑΝΑΚΕΦΑΛΑΙΩΣΗ

Στο κεφάλαιο αυτό περιγράφηκαν οι τεχνολογίες που αποτελούν την πλατφόρμα υλοποίησης της πτυχιακής και αιτιολογήθηκαν οι αποφάσεις για τις οποίες επιλέχθηκαν.

Συγκεκριμένα, περιγράφηκε το μοντέλο MVC που ικανοποιεί πλήρως την αρχιτεκτονική 3-tier για κατανεμημένα συστήματα, και οι τεχνολογίες: J2EE, JSF και Hibernate οι οποίες υποστηρίζουν την υλοποίηση των μοντέλων. Ειδικά για τα JSF,

αναλύθηκε ιδιαίτερα ο κύκλος ζωής μίας εφαρμογής JSF, ώστε να γίνουν καλύτερα κατανοητές οι φάσεις από τις οποίες περνάει ένα HTTP αίτημα μέχρι να εμφανιστεί η αντίστοιχη απάντηση.

Όλες οι παραπάνω τεχνολογίες έχουν το επιπλέον πλεονέκτημα ότι είναι ανοιχτού κώδικα.

3. ΠΡΟΔΙΑΓΡΑΦΕΣ ΚΑΙ ΣΧΕΔΙΑΣΜΟΣ

3.1. ΠΡΟΔΙΑΓΡΑΦΕΣ

Ο έλεγχος πρόσβασης δεδομένων είναι μείζονος σημασίας για τους οργανισμούς, όπως είναι επιχειρήσεις και δημόσιοι φορείς. Τα εταιρικά ή κρατικά δεδομένα και ιδιαιτέρως εκείνα που έχουν ευαίσθητη φύση πρέπει να προστατεύονται από μη εξουσιοδοτημένη προβολή, πόσω μάλλον από κακόβουλες προθέσεις που αποσκοπούν σε αλλοίωση η ακόμη και διαγραφή τους. Δεν υπάρχει οργανισμός σήμερα που να μην εφαρμόζει μια γενική ή εξατομικευμένη και προσαρμοσμένη στις ανάγκες και το περιβάλλον του πολιτική προστασίας. Συνεπώς, κάθε πληροφοριακό σύστημα πρέπει να παρέχει δυνατότητες εκχώρησης δικαιωμάτων με τον τρόπο που καθορίζει η πολιτική προστασίας δεδομένων του εκάστοτε οργανισμού.

Ένα τυπικό σύστημα ελέγχου προσπέλασης περιλαμβάνει υποκείμενα που προσπελαύνουν αντικείμενα μέσω κατάλληλων λειτουργιών. Συνήθως, το «ποιος» (*υποκείμενο*) θα δει «τι» (*αντικείμενο*) καθορίζεται από το ποια και πόση γνώση χρειάζεται να έχει πάνω σε συγκεκριμένα δεδομένα. Ο καθένας έχει την απαραίτητη πρόσβαση στα δεδομένα που χρειάζεται για να φέρει σε πέρας το στόχο του, αλλά όχι περισσότερη.

Ο διαχωρισμός της πρόσβασης μπορεί να είναι **κάθετος** ή **οριζόντιος**.

- Στον **κάθετο διαχωρισμό** η πρόσβαση που δίνεται στα δεδομένα και τις πληροφορίες χωρίζεται σε διαβαθμίσεις. Για παράδειγμα, ανώτατα στελέχη μπορούν να επεξεργαστούν όλες τις πληροφορίες, τα αμέσως επόμενα στην ιεραρχία λιγότερες, ενώ όσο κατεβαίνουμε στην ιεραρχία τα στελέχη έχουν πρόσβαση σε μικρό μόνο μέρος των πληροφοριών. Το σύστημα αυτό εφαρμόζεται σε περιβάλλοντα με σαφή και ρητή ιεραρχία, όπως π.χ. ο στρατός ή επιχειρήσεις με λειτουργική οργάνωση (functional organisation).
- Στον **οριζόντιο διαχωρισμό**, τα δεδομένα δε χωρίζονται κάθετα σε διαβαθμίσεις, γιατί υπάρχουν περιπτώσεις όπου υποκείμενα που ανήκουν στο ίδιο επίπεδο εμπιστευτικότητας δεν μπορούν να μοιράζονται τις ίδιες πληροφορίες. Για παράδειγμα, στα ευαίσθητα δεδομένα των ιατρικών εξετάσεων ενός ασθενούς πρέπει να έχει πρόσβαση μόνο ο προσωπικός του ιατρός και όχι όλοι οι ιατροί.

Η πρόσβαση σε ένα αντικείμενο έχει τρεις βασικές μορφές.

1. Η πρώτη είναι η πλήρης απουσία δικαιωμάτων πρόσβασης του υποκειμένου πάνω στο αντικείμενο. Σε αυτήν την κατηγορία ανήκουν υποκείμενα χωρίς ανάγκη κι επομένως εκκαθάριση να επεξεργαστούν κάποια ομάδα πληροφοριών.
2. Ακολουθεί το δικαίωμα προβολής ή ανάγνωσης, όπου το υποκείμενο μπορεί να προβάλει, να διαβάσει, να «δει» δηλαδή τα δεδομένα αλλά δεν μπορεί να τα αλλάξει.
3. Δυνατότητα μεταβολής των δεδομένων έχουν μόνο όσοι έχουν δικαίωμα εγγραφής πάνω στο συγκεκριμένο αντικείμενο και ανήκουν στην τρίτη και τελευταία κατηγορία.

Ας δούμε κάποιες περιπτώσεις όπου τα παραπάνω μπορούν να εφαρμοστούν:

α) το εταιρικό περιβάλλον, έχει σαφή ιεραρχία, με τα διοικητικά και εκτελεστικά στελέχη να έχουν ανάγκη επεξεργασίας μεγάλου όγκου ευαίσθητων δεδομένων, ενώ οι υπάλληλοι πάντα έχουν πρόσβαση στα δεδομένα που πρέπει να γνωρίζουν ώστε να επιτελέσουν το καθημερινό τους έργο.

β) με διαφορετικές ομάδες πληροφοριών ασχολούνται ένας οικονομικός διευθυντής κι ένας διευθυντής παραγωγής.

γ) οι κατά τόπους διευθυντές μίας πολυεθνικής εταιρίας επεξεργάζονται δεδομένα και πληροφορίες που αφορούν την τοπική αγορά του καθενός, ενώ ο γενικός εκτελεστικός διευθυντής της πολυεθνικής θα έχει πρόσβαση σε όλα τα στοιχεία της εταιρίας.

Με βάση όλα όσα αναφέρθηκαν παραπάνω γίνεται φανερή η ανάγκη για μια εφαρμογή η οποία θα διαχειρίζεται κατάλληλα την πρόσβαση στην πληροφορία της επιχείρησης ή του οργανισμού.

Θεωρούμε ότι η εκχώρηση δικαιωμάτων μπορεί να γίνει είτε σε **επίπεδο χρήστη** (ατομικό επίπεδο) είτε σε **επίπεδο ομάδας** (ομαδικό επίπεδο). Η δεύτερη περίπτωση μπορεί να εφαρμοστεί μόνο σε περιπτώσεις κάθετης διαβάθμισης, όπου όλα τα μέλη μιας ομάδας χρηστών μοιράζονται τα ίδια δικαιώματα στα δεδομένα που δικαιούνται να διαχειριστούν.

Η εφαρμογή θα εκχωρεί ειδικές δυνατότητες πρόσβασης σε ομάδες χρηστών με παρεμφερή εργασιακά αντικείμενα αλλά ταυτόχρονα θα εξατομικεύει τις δυνατότητες πρόσβασης του καθενός αναλόγως με τις ανάγκες και τις αρμοδιότητές του. Η εφαρμογή αυτή θα διαθέτει δυνατότητες εξατομίκευσης στην διαχείριση της πρόσβασης που θα αφορούν τόσο στο τι μπορεί να δει κανείς, τι θα μπορεί να κάνει

πάνω σε αυτή την πληροφορία που βλέπει και σε ποια γλώσσα θα το βλέπει. Ταυτόχρονα η υλοποίησή της βασίζεται σε καινοτόμες τεχνολογίες του ιστού, «έξυπνες» προγραμματιστικές τεχνικές και φιλικές τεχνικές παρουσίασης βασισμένες σε πρότυπα της αλληλεπίδρασης ανθρώπου-υπολογιστή.

Στην πτυχιακή αυτή έγινε η σύλληψη ο σχεδιασμός και η υλοποίηση μιας εφαρμογής που δίνει λύσεις στα παραπάνω.

Ο σχεδιασμός της εφαρμογής έγινε με βάση το μοντέλο MVC που αναλύθηκε στο προηγούμενο κεφάλαιο.

3.2 ΣΧΕΔΙΑΣΜΟΣ

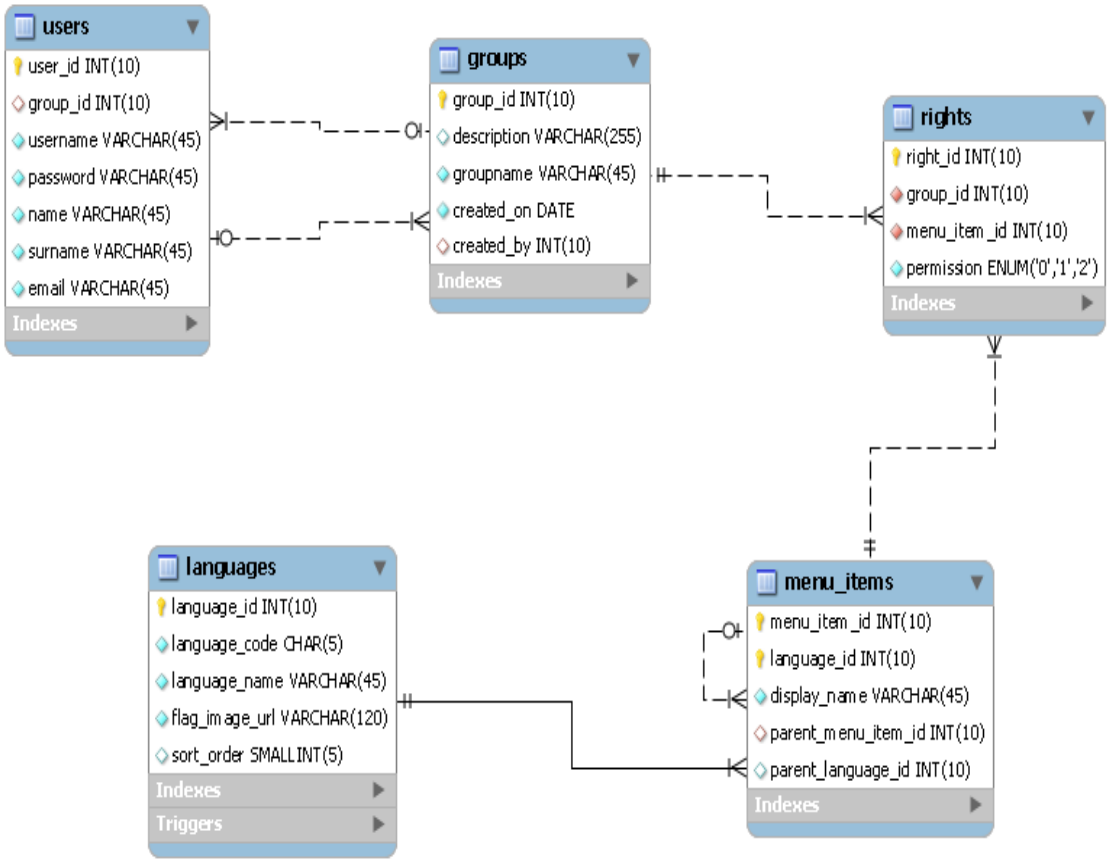
3.2.1. Μοντελοποίηση οντοτήτων-συσχετίσεων

Αρχικά, σχεδιάστηκαν οι οντότητες της εφαρμογής και οι συσχετίσεις μεταξύ τους, όπως ορίζει το μοντέλο οντοτήτων-συσχετίσεων (ER Model - Εικόνα 5). Οντότητα είναι ένα αντικείμενο του πραγματικού κόσμου το χαρακτηρίζεται να έχει διακριτή ύπαρξη σε σχέση με τα υπόλοιπα αντικείμενα, ενώ συσχέτιση αποτελεί η διασύνδεση δύο ή περισσότερων οντοτήτων [4]

Σε αυτό διακρίνονται τέσσερεις οντότητες:

1. *Users*, η οποία αντιπροσωπεύει τους χρήστες του συστήματος,
2. *Groups*, η οποία αντιπροσωπεύει τις ομάδες χρηστών,
3. *Menu_Items* η οποία απεικονίζει τα στοιχεία μενού και
4. *Languages*, δομή στην οποία αποθηκεύονται οι πληροφορίες των διαθέσιμων γλωσσών της εφαρμογής. Με τον πίνακα αυτόν το μενού και συνεπώς η πρόσβαση στις λειτουργίες της επιχείρησης ανεξαρτητοποιείται από τη γλώσσα.

Η συσχέτιση της οντότητας Groups με την οντότητα Menu_Items είναι «πολλά-προς-πολλά», εφόσον μία ομάδα μπορεί να έχει δικαιώματα σε περισσότερα από ένα στοιχεία μενού και σε ένα στοιχείο μενού μπορούν να έχουν δικαιώματα πολλές ομάδες. Για το λόγο αυτό ήταν επιβεβλημένη η δημιουργία ενός πίνακα συσχέτισης, ονόματι *Rights*. Ο πίνακας αυτός κρατά όλη την πληροφορία για την πεμπτουσία του συστήματος, καθώς κάθε εγγραφή του αποθηκεύει το δικαίωμα πρόσβασης των μελών μίας ομάδας σε συγκεκριμένο στοιχείο μενού.



Εικόνα 5 - Το διάγραμμα οντοτήτων-συσχετίσεων (ER) της εφαρμογής

Στη συνέχεια περιγράφονται αναλυτικά οι πίνακες που υλοποιούν τις οντότητες καθώς και τις επιλογές που ελήφθησαν υπόψη στο σχεδιασμό τους:

Ο πίνακας ***Users*** περιέχει τις εξής ιδιότητες:

1. *user_id*, τύπου ακεραίου, που αντιστοιχεί στο μοναδικό αναγνωριστικό του χρήστη, αποτελεί το κύριο κλειδί της οντότητας και παράγεται αυτόματα από το DBMS.
2. *username*, τύπου αλφαριθμητικού, που δηλώνει το όνομα χρήστη με το οποίο ο χρήστης ταυτοποιείται από το σύστημα. Λόγω του ότι πρέπει να καλύπτει δυνατότητες επιλεκτικής πρόσβασης στην πληροφορία πρέπει να είναι μοναδικό συνεπώς χρησιμοποιώντας τις δυνατότητες της MySQL απαγορεύουμε τα διπλότυπα.
3. *password*, τύπου αλφαριθμητικού, το οποίο είναι το συνθηματικό που θα πρέπει να εισάγει ο χρήστης για την πιστοποίηση-αυθεντικοποίησή του.
4. *name*, τύπου αλφαριθμητικού, το όνομα του χρήστη.
5. *surname*, τύπου αλφαριθμητικού, το επώνυμο του χρήστη.
6. *email*, τύπου αλφαριθμητικού, η ηλεκτρονική διεύθυνση του χρήστη.

Ο πίνακας ***Groups*** αποτελείται από τα εξής πεδία:

1. *group_id*, τύπου ακεραίου, που αντιστοιχεί στο μοναδικό αναγνωριστικό της ομάδας, αποτελεί το κύριο κλειδί της οντότητας και παράγεται αυτόματα από το DBMS.
2. *groupname*, τύπου αλφαριθμητικού, που δηλώνει το όνομα της ομάδας. Και εδώ απαγορεύονται τα διπλότυπα.
3. *description*, τύπου αλφαριθμητικού, όπου δηλώνεται μία περιγραφή για την ομάδα.
4. *created_on*, τύπου ημερομηνίας, για να δηλώνει πότε δημιουργήθηκε η ομάδα.
5. *created_by*, αναφορά στο πεδίο *user_id* του πίνακα Users, στο δημιουργό δηλαδή της ομάδας αυτής.

Οι χρήστες καταχωρούνται σε μία ομάδα μόνο και αποκτούν τις δυνατότητες πρόσβασης της ομάδας. Ο κάθε χρήστης έχει συγκεκριμένη πρόσβαση στο σύστημα η οποία καθορίζεται από την πολιτική της συγκεκριμένης ομάδας. Με αυτό τον τρόπο έχει συνεπή (πάντα την ίδια) πρόσβαση στην εφαρμογή, βάσει των δικαιωμάτων που έχουν εκχωρηθεί στην ομάδα.

Ο πίνακας ***Menu_Items*** αποτελείται από τα παρακάτω πεδία:

1. *menu_item_id*, τύπου ακεραίου, που αντιστοιχεί στο μοναδικό αναγνωριστικό του πεδίου, είναι μέρος του κύριου κλειδιού της οντότητας.
2. *language_id*, τύπου ακεραίου, είναι αναφορά σε μία από τις γλώσσες στις οποίες δύναται η εφαρμογή να εμφανίσει μενού και συναποτελεί με το παραπάνω πεδίο το κύριο κλειδί του πίνακα.
3. *display_name*, τύπου αλφαριθμητικού, το οποίο χρησιμοποιείται ως το περιγραφικό του στοιχείου.
4. *parent_ menu_item_id* και
5. *parent_language*_id, τύπου ακεραίου αμφότερα, τα οποία είναι αναφορά στο κύριο κλειδί του ίδιου πίνακα. Δηλώνουν κάτω από ποιο στοιχείο μενού βρίσκεται (αν δεν είναι πατρικό) το συγκεκριμένο στοιχείο.

Ο πίνακας αυτός υλοποιεί δενδρικές δομές για την αποθήκευση των στοιχείων μενού, καθώς κάθε στοιχείο μπορεί να είναι απόγονος άλλου στοιχείου και είναι εν δυνάμει γονέας άλλων στοιχείων. Έτσι, με τη διαρκή συμπλήρωση στοιχείων, δημιουργούνται διάφορα δένδρα, το καθένα με ρίζα το πατρικό στοιχείο και φύλλα τα στοιχεία που δεν έχουν κανέναν απόγονο.

Ο πίνακας ***Languages*** περιέχει τις εξής πληροφορίες:

1. *language_id*, τύπου ακεραίου, που είναι το κύριο κλειδί της οντότητας και παράγεται αυτόματα από τη MySQL.
2. *language_code*, τύπου ακεραίου, είναι ο κωδικός της γλώσσας κατά ISO-639 και πρέπει να είναι μοναδικό.
3. *language_name*, τύπου ακεραίου, είναι το μοναδικό όνομα της γλώσσας στο σύστημα.
4. *flag_image_url*, τύπου αλφαριθμητικού, αναφέρει τη διαδρομή για το αρχείο εικόνας με το οποίο συμβολίζεται η γλώσσα στην εφαρμογή.
5. *sort_order*, που καθορίζει τη σειρά στην οποία θα εμφανίζεται η γλώσσα στη λίστα με τις διαθέσιμες γλώσσες της εφαρμογής.

Ο πίνακας συσχέτισης ***Rights*** διαχειρίζεται τα δικαιώματα του κάθε χρήστη σε καθένα από τα πεδία μενού της εφαρμογής και περιέχει τρία (3) πεδία:

1. *right_id*, αυτόματα παραγόμενο κύριο κλειδί τύπου ακεραίου.
2. *group_id*, αναφορά στο ομώνυμο πεδίο του πίνακα Groups, το οποίο αντιπροσωπεύει την ομάδα την οποία αφορά το δικαίωμα.
3. *menu_item_id*, αναφορά στο ομώνυμο πεδίο του πίνακα Menu_Items, το οποίο αντιπροσωπεύει τη σελίδα και αποτελεί μαζί με το παραπάνω το κύριο κλειδί του πίνακα (***σημαντική παρατήρηση*:** παρόλο που το κύριο κλειδί του πίνακα Menu_Items είναι σύνθετο, εδώ μας χρειάζεται μόνο η πληροφορία του id, καθώς τα στοιχεία που αφορούν το ίδιο ουσιαστικά μενού αλλά σε διαφορετική γλώσσα αντιμετωπίζονται ενιαία όσον αφορά τα δικαιώματα. Δηλαδή, το δικαίωμα που θα έχει μία ομάδα σε κάποιο στοιχείο γραμμένο στα ελληνικά, είναι το ίδιο που έχει και στο αντίστοιχο στοιχείο που είναι γραμμένο στα αγγλικά).
4. *permission*, τύπου enumerator. Δηλώνει τα δικαιώματα που έχει ο χρήστης στο συγκεκριμένο πεδίο του πίνακα. Οι πιθανές τιμές είναι 0, 1 και 2 και αντιστοιχούν σε: κανένα δικαίωμα, δικαίωμα ανάγνωσης και δικαίωμα εγγραφής.

Έγινε προσπάθεια να μοντελοποιηθεί η περιοχή του προβλήματος όσο πιο απλά και όσο πιο κοντά στην πραγματικότητα γίνεται.

Παρόλο που φαίνεται ότι οι πίνακες είναι λίγοι για τη μοντελοποίηση που απαιτεί η εφαρμογή, πόσω μάλλον όταν κανένας πίνακας από τους παραπάνω δε διαθέτει περισσότερα από έξι πεδία, ωστόσο ικανοποιούν το σκοπό της εφαρμογής, κάνοντας

σωστή αντιστοίχιση των ιδιοτήτων της κάθε οντότητας στο ER μοντέλο, χωρίς αναφορές σε περιττά χαρακτηριστικά. Είναι χαρακτηριστικό καλού σχεδιασμού ότι με τόσα λίγα εργαλεία δόθηκε τόση σημαντική και πολύπλευρη δυνατότητα στην πρόσβαση στην πληροφορία μιας επιχείρησης.

Στη συνέχεια παρατίθεται ο σχεδιασμός των οντοτήτων αλλά και της όψης του συστήματος με βάση το μοντέλο MVC και τις δυνατότητες των εργαλείων προγραμματισμού.

3.2.2. Σχεδιασμός Model

Στο επίπεδο του *μοντέλου* είναι οι οντότητες της εφαρμογής, όπως περιγράφηκαν στην ενότητα 3.2.1., αντιστοιχισμένες σε κλάσεις της java και τα EJB (Enterprise Java Beans) τα οποία ορίζουν και χειρίζονται τις ενέργειες που επιτρέπονται πάνω σε στιγμιότυπα των κλάσεων αυτών. Η χρήση των EJB προϋποθέτει την ύπαρξη ενός *web container* (αλλιώς *application server*) καθώς δεν εκτελούνται πάνω σε απλό διακομιστή ιστού (*web server*). Για την περίπτωσή μας προτιμήθηκε Glassfish v3.1 για το σκοπό αυτό.

3.2.3. Σχεδιασμός View

Το ρόλο του *view* έχουν τα JSF (Java Server Faces) που σε συνδυασμό με τις τεχνολογίες CSS (*Cascading Style Sheets*), Javascript, jQuery και AJAX (*Asynchronous Javascript and XML*) προσφέρουν απεριόριστες δυνατότητες όσον αφορά την εμφάνιση αλλά και την ταχύτητα της εφαρμογής. Επίσης, χρησιμοποιήθηκαν custom JSF components προς αποφυγή συγγραφής ίδιου κώδικα ξανά και ξανά.

Παρακάτω αναλύεται το πώς το μοντέλο MVC υλοποιείται χρησιμοποιώντας καθεμία από τις τεχνολογίες αυτές.

3.2.3.1. CSS

Τα CSS scripts τρέχουν αποκλειστικά στο φυλλομετρητή και ρυθμίζουν την εμφάνιση του περιεχομένου σε αυτόν. Ιδιότητες απλές, όπως το πλάτος και ύψος των στοιχείων ή η ευθυγράμμιση του κειμένου, αλλά και πιο σύνθετες και πολύπλοκες ρυθμίζονται εύκολα και γρήγορα σε λίγες μόνο γραμμές.

Στην παρούσα εφαρμογή, έγινε ένα εναλλακτικό theme για να καταδειχθεί η πλήρης ανεξαρτησία της παρουσίασης των δεδομένων από την επεξεργασία τους, όπως ορίζει το MVC μοντέλο.

3.2.3.2. Javascript – jQuery

Η Javascript εκτελείται επίσης στη πλευρά του πελάτη (client programming) και έχει τη δυνατότητα επιρροής και μεταβολής του ίδιου του περιεχομένου της σελίδας. Μπορεί να εισάγει ή να αφαιρέσει περιεχόμενο από τη σελίδα, να αλλάξει το κείμενο ενός πεδίου, ακόμα και να μεταβάλει τον τρόπο εμφάνισης της σελίδας στέλνοντας κώδικα css στη σελίδα.

Η jQuery είναι μία ισχυρή βιβλιοθήκη της Javascript η οποία προσφέρει ευκολία στον προγραμματιστή, καθώς περιέχει συναρτήσεις που αντικαθιστούν πολλές γραμμές κώδικα σε απλή Javascript.

Υπάρχουν δύο αρχεία javascript στον αντίστοιχο φάκελο της εφαρμογής, το `menu_js.js` το οποίο ρυθμίζει τη συμπεριφορά των μενού και το `script.js` που ασχολείται με όλα τα άλλα, όπως επικύρωση εισόδου του χρήστη και εμφάνιση αντίστοιχων μηνυμάτων (για να μη φορτώνεται ο διακομιστής για κάτι που μπορεί να ρυθμίσει ο φυλλομετρητής), επιβεβαίωση μη αναστρέψιμων ενεργειών όπως η διαγραφή, αλλαγή χρωμάτων για εφέ και άλλα. Η δε jQuery βοήθησε στην παραγωγή οπτικών εφέ σε περιπτώσεις σφάλματος ή παράλειψης

3.2.3.3. AJAX

Η ασύγχρονη επικοινωνία πελάτη-διακομιστή είναι μία τεχνολογία που μειώνει το φόρτο του διακομιστή και βελτιώνει την εμπειρία του χρήστη.

Η φιλοσοφία είναι να αποφεύγονται οι λεγόμενες πλήρεις υποβολές σελίδας (full page submit), το να αποστέλλονται δηλαδή όλα τα δεδομένα της φόρμας στο διακομιστή και να πρέπει να περιμένει ο χρήστης από το διακομιστή να παράξει το ανανεωμένο περιεχόμενο. Αντί για αυτό, με τη χρήση XML μπορούν να σταλθούν μόνο τα απαραίτητα δεδομένα στο διακομιστή, να παράξει αυτός την κατάλληλη απάντηση και μετά με τη χρήση Javascript να ανανεωθεί μέρος του περιεχομένου της σελίδας. Με τον τρόπο αυτό το όφελος είναι πολλαπλό:

- μειώνεται η κίνηση στο δίκτυο και ο φόρτος εργασίας του διακομιστή.
- η σελίδα δεν συμπεριφέρεται όπως στην πλήρη υποβολή όπου η διεπαφή χρήστη σταματά να αποκρίνεται και πολλές φορές εξαφανίζεται μέχρι να

«φτάσει» το ανανεωμένο περιεχόμενο, αλλά τα στοιχεία ανανεώνονται μέσω Javascript μετά την παραλαβή του περιεχομένου.

Στην εφαρμογή χρησιμοποιήθηκε η τεχνική AJAX ως ενσωματωμένη δυνατότητα στο framework του JSF, πράγμα που σημαίνει ότι δε χρειάστηκε να γραφεί κώδικας σε javascript ή XML.

Οι περιπτώσεις στις οποίες χρησιμοποιήθηκε περιλαμβάνουν μεταξύ άλλων την ανανέωση πινάκων αποτελεσμάτων κατόπιν αιτήματος για αλλαγή σελίδας ή προσθήκης/αφαίρεσης στοιχείου, την αλλαγή θέσης στοιχείων μέσα σε πίνακα (συγκεκριμένα στις γλώσσες του συστήματος) και την εμφάνιση και απόκρυψη κάποιων φορμών.

3.2.3.4. Custom components

Πολλές φορές χρειάζεται να χρησιμοποιηθεί το ίδιο κομμάτι κώδικα σε περισσότερες από μία περιπτώσεις. Για το λόγο αυτό, τα JSF δίνουν τη δυνατότητα δημιουργίας ενός εξατομικευμένου component με τη χρήση των διαθέσιμων components των JSF. Για να γίνει αυτό, πρέπει να δημιουργηθεί ένα αρχείο με κατάληξη `.xhtml` στο φάκελο `./resources/cc` της εφαρμογής και να δηλωθεί το αρχείο αυτό στο `xmlns` attribute του html tag της σελίδας όπου πρόκειται να χρησιμοποιηθεί.

Στην παρούσα εφαρμογή, για παράδειγμα, το κομμάτι της διαχείρισης των σελίδων των πινάκων της εφαρμογής (είτε αφορούν χρήστες είτε ομάδες) που αποτελείται από κάποιες δεκάδες γραμμές κώδικα, έπρεπε να εισαχθεί σε τέσσερα διαφορετικά σημεία. Επίσης, σε τρία σημεία έπρεπε να χρησιμοποιηθούν αναδυόμενα tooltips. Έτσι, δημιουργήθηκαν δύο components για να εξυπηρετήσουν αυτούς τους δύο σκοπούς.

3.2.4. Σχεδιασμός Controller

Στο ρόλο του *controller* τα Managed Beans της Java, τα οποία δένουν με τα JSF και βρίσκονται μόνιμα σε αναμονή γεγονότων που πυροδοτούνται από το χρήστη, λαμβάνουν στοιχεία από αυτόν και εκτελούν κατάλληλες διαδικασίες, όπως έλεγχο και επικύρωση των στοιχείων της φόρμας, πριν την κλήση των μεθόδων των EJB για επικοινωνία με την ίδια τη βάση.

ΑΝΑΚΕΦΑΛΑΙΩΣΗ

Στο κεφάλαιο αυτό έγινε λεπτομερής περιγραφή του σχεδιασμού της εφαρμογής και των επιλογών που οδήγησαν στο συγκεκριμένο σχεδιασμό, καθώς επίσης και στη χρήση βοηθητικών τεχνολογιών ανάπτυξης.

Παρουσιάστηκαν οι απαραίτητες τεχνολογίες όπως το css, η javascript, η AJAX και τα custom components που απαιτούνται για να προσφέρουν επιπλέον ταχύτητα στη διαδικασία συγγραφής κώδικα και έκαναν την εφαρμογή πιο γρήγορη, αλλά ταυτόχρονα και πιο λιτή.

Στη συνέχεια, θα ακολουθήσει η περιγραφή της υλοποίησης της εφαρμογής η οποία βασίζεται στον σχεδιασμό που παρουσιάστηκε σε αυτό το κεφάλαιο.

4. ΥΛΟΠΟΙΗΣΗ

Στο κεφάλαιο αυτό αναλύονται οι κλάσεις και οι ιστοσελίδες της εφαρμογής καθώς και η δομή των αρχείων, όπως αυτά αναπτύσσονται μέσα στο διακομιστή.

4.1. Οντότητες της εφαρμογής (πακέτο *entities*)

Οι κύριες κλάσεις τις εφαρμογής είναι αυτές που αντιπροσωπεύουν τις οντότητες του προβλήματος. Πρακτικά, πρόκειται για οπτικοποίηση των πινάκων της βάσης δεδομένων, καθώς κάθε εγγραφή που ανακτάται από τη βάση ή που πρόκειται να δημιουργηθεί ενθυλακώνεται σε στιγμιότυπο της αντίστοιχης κλάσης.

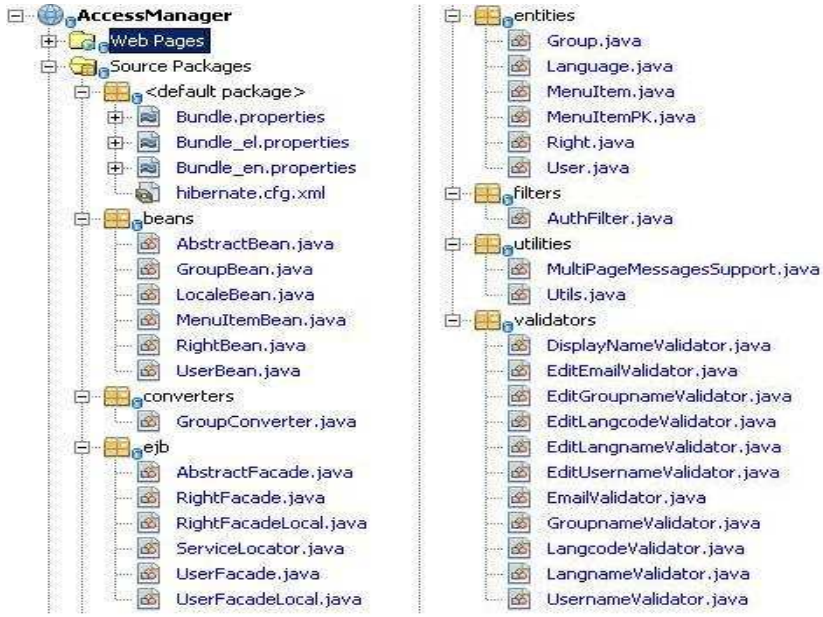
Οι κλάσεις της εφαρμογής περιέχουν ερωτήματα σε HQL (Hibernate Query Language) που ονομάζονται *named queries*, τα οποία εκτελούνται από τα EJB όπως αναφέρεται παρακάτω. Βρίσκονται στο πακέτο *entities* και συγκεκριμένα:

- ***User***: κάθε αντικείμενο της κλάσης αυτής είναι και ένας χρήστης του συστήματος. Τα μέλη της είναι τα προσωπικά στοιχεία του χρήστη και τα στοιχεία εισόδου του στο σύστημα (login credentials).
- ***Group***: αντικατοπτρίζει τις ομάδες χρηστών. Τα μέλη της είναι στοιχεία που αφορούν το όνομα και την περιγραφή της ομάδας καθώς και πληροφορίες για τη δημιουργία της (δημιουργός και timestamp δημιουργίας).
- ***Language***: δεν είναι άλλο από τις γλώσσες που είτε παρέχονται έτοιμες από την εφαρμογή είτε μπορούν να δημιουργηθούν από τους διαχειριστές του συστήματος. Στοιχεία όπως όνομα, κωδικός αλλά και σειρά εμφάνισης αποτελούν τα μέλη της.
- ***MenuItem***: ενθυλακώνει τα στοιχεία του μενού (είτε τα παρεχόμενα από την εφαρμογή είτε αυτά που κατασκευάζουν οι διαχειριστές της) σε αντικείμενα με πληροφορίες όπως το όνομα εμφάνισης του στοιχείου, τη γλώσσα του και το γονέα του. Αυτό το τελευταίο είναι το χαρακτηριστικό που ξεχωρίζει την οντότητα αυτή από τις άλλες τις εφαρμογής καθώς έχει αναφορά σε αντικείμενα του ίδιου της του εαυτού. Τα submenus έχουν ως γονέα το στοιχείο μενού που βρίσκεται ακριβώς από πάνω τους στην ιεραρχία.
- ***Right***: ο ακρογωνιαίος λίθος της παρούσας εργασίας. Κάθε στιγμιότυπο της οντότητας Right αντιπροσωπεύει τα δικαιώματα που έχει μία ομάδα πάνω σε συγκεκριμένο αντικείμενο της `MenuItem`. Εύκολα συμπεραίνει κανείς ότι μέλη της κλάσης Right αποτελούν η ομάδα, το στοιχείο μενού και ένα

αλφαριθμητικό (στο επίπεδο της βάσης το πεδίο αυτό είναι υλοποιημένο με enumerator, τύπος τον οποίο το Hibernate αντιστοιχίζει στον τύπο `java.lang.String`) που συμβολίζει το δικαίωμα που έχει στο στοιχείο η ομάδα αυτή.

- ***MenuItemPK:*** δεν πρόκειται ακριβώς για οντότητα αλλά για κλάση που προκύπτει από την ανάγκη του Hibernate να γίνεται ξεχωριστή κλάση το πρωτεύον κλειδί μίας οντότητας όταν αυτό είναι σύνθετο (αποτελείται δηλαδή από περισσότερα του ενός πεδία). Επειδή ακριβώς η `MenuItem` δεν μπορεί να υλοποιηθεί με μοναδικό αναγνωριστικό (οι εγγραφές που αφορούν πρακτικά το ίδιο στοιχείο σε διαφορετικές γλώσσες πρέπει να μοιράζονται το ίδιο αναγνωριστικό, διαφέροντας μόνο στο πεδίο της γλώσσας), αυτό το σύνθετο κλειδί μοντελοποιήθηκε σε ξεχωριστή κλάση με μέλη το αναγνωριστικό του στοιχείου μενού και τη γλώσσα του (πεδία που αποτελούν και το πρωτεύον κλειδί του αντίστοιχου πίνακα menu_items στη βάση).

Η δομή των κλάσεων της εφαρμογής παρουσιάζεται στην Εικόνα 6.



Εικόνα 6 - Η δομή των κλάσεων της εφαρμογής (NetBeans IDE)

4.2. EJB Facades (πακέτο *ejb*)

Τα EJB façades είναι οι κλάσεις που επιφορτίζονται με το ρόλο της επικοινωνίας με τη βάση για ανάκτηση, εγγραφή, επεξεργασία ή διαγραφή δεδομένων, υλοποιώντας `Container-Managed Transactions (CMT)` βάσει της οποίας δεν υπάρχει ανάγκη πουθενά στον κώδικα για έναρξη, εκτέλεση ή ακύρωση μίας συναλλαγής με

τη βάση, ούτε καν σύνδεση με ή αποσύνδεση από το διακομιστή. Τα EJB αναλαμβάνουν όλες τις συναλλαγές με τη βάση και είναι υπεύθυνα για την επιτυχή εκτέλεσή τους ή για την ακύρωσή τους σε περίπτωση σφάλματος [3].

Στις κλάσεις των EJB ορίζονται οι μέθοδοι που εκτελούν τα named queries που αναφέρθηκαν στην προηγούμενη ενότητα. Η μοναδική τους μεταβλητή / μέλος είναι ένα αντικείμενο τύπου "`javax.persistence.EntityManager`", το οποίο είναι υπεύθυνο για το άνοιγμα της σύνδεσης με τη βάση, την ολοκλήρωση της κάθε συναλλαγής (επιτυχούς ή μη) και το κλείσιμο της σύνδεσης.

Όσον αφορά το πόσα EJB façades θα γραφτούν σε μία εφαρμογή, υπάρχει η δυνατότητα να γίνει ένα κοινό façade για όλες τις οντότητες τις εφαρμογής, κάτι που θα παρήγαγε ένα πρόγραμμα μαμούθ με αμέτρητες γραμμές κώδικα. Από την άλλη, μπορεί να γίνει ένα façade για κάθε οντότητα, με αποτέλεσμα να υπάρχουν πολλά μικρά προγράμματα στο συγκεκριμένο πακέτο. Η λύση που προτιμήθηκε είναι ακριβώς ανάμεσα στις δύο προαναφερθείσες, να ομαδοποιηθούν δηλαδή οι οντότητες ανάλογα με τη φύση και τη λειτουργικότητά τους και να γίνει ένα façade για κάθε ομάδα. Αξίζει να σημειωθεί ότι το NetBeans IDE έχει τη δυνατότητα αυτόματης δημιουργίας όλων των EJB με τις βασικές μεθόδους τους.

- ***AbstractFacade***: αφηρημένο façade που υλοποιεί τέσσερεις βασικές μεθόδους CRUD (Create-Read-Update-Delete) και από το οποίο κληρονομούν όλα τα υπόλοιπα façades. Οι γραμμές αυτής της κλάσης παράχθηκαν αυτόματα από το IDE.
- ***UserFacade***: το EJB που ορίζει και χειρίζεται τις επιπλέον (πέρα από αυτές που ορίζει το αφηρημένο) ενέργειες πάνω στις οντότητες των χρηστών και των ομάδων (`User` και `Group`). Περιέχει κάποια ειδικά ερωτήματα προς τη βάση, υλοποιώντας ουσιαστικά τα named queries των κλάσεων User και Group.
- ***RightFacade***: αφορά τις ενέργειες πάνω στις υπόλοιπες οντότητες (`Right`, `MenuItem` και `Language`) και περιέχει κώδικα εκτέλεσης των named queries των κλάσεων αυτών.
- ***ServiceLocator***: Αυτή η κλάση δεν είναι EJB αλλά υλοποιεί το pattern *Service Locator*, για εύκολη ανάκτηση των EJB. Ο κώδικας που χρησιμοποιήθηκε είναι από την ιστοσελίδα της java όπου και περιγράφεται αναλυτικά το pattern [6].

4.3. Managed Beans (πακέτο *beans*)

Ο ενδιάμεσος κώδικας που παρεμβάλλεται μεταξύ της υλοποίησης των σελίδων (των αρχείων JSF) και των EJB είναι τα managed beans και περιέχονται στο πακέτο beans. Αντιδρώντας στην είσοδο που παρέχει ο χρήστης από το φυλλομετρητή του, εκτελεί τις προπαρασκευαστικές ενέργειες πριν τελικά αποφασιστεί η εκτέλεση των κατάλληλων μεθόδων των EJB. Επειδή εδώ, σε αντίθεση με τα EJB που περιέχουν αρκετό κοινό κώδικα για κάθε κλάση, απαιτούνται διαφορετικοί έλεγχοι και προετοιμασίες για κάθε αντικείμενο, προτιμάται η δημιουργία ενός managed bean για τα αντικείμενα κάθε κλάσης που διαχειρίζεται η εφαρμογή, με εξαίρεση την κλάση Right.

- ***AbstractBean***: επειδή η υλοποίηση της σελιδοποίησης των πινάκων που προβάλλουν χρήστες και ομάδες του συστήματος είναι ίδια και στις δύο περιπτώσεις, προς αποφυγή εγγραφής όμοιου κώδικα χρησιμοποιήθηκε ένα αφηρημένο managed bean το οποίο περιέχει αυτές ακριβώς τις μεθόδους και κληρονομείται από τα `UserBean` και `GroupBean`.
- ***UserBean***: ότι ενέργεια έχει να κάνει με αντικείμενα της κλάσης `User` περνάει πρώτα από εδώ. Αυτό το managed bean εκτελεί τις απαραίτητες ενέργειες για τη διαχείριση χρηστών πριν κληθούν οι μέθοδοι των EJB που θα μονιμοποιήσουν το αποτέλεσμα. Επίσης, είναι επιφορτισμένο με τη διαδικασία εισόδου / εξόδου από το σύστημα (login / logout).
- ***GroupBean***: αντίστοιχα με το προηγούμενο, το managed bean αυτό είναι υπεύθυνο για την προετοιμασία προβολής, εισαγωγής, διαγραφής και ενημέρωσης ενός στιγμιοτύπου της κλάσης Group.
- ***LocaleBean***: αφορά το χειρισμό αντικειμένων της κλάσης `Language` αλλά και ρυθμίζει την από το χρήστη αιτούμενη αλλαγή γλώσσας των ιστοσελίδων. Η εφαρμογή χρησιμοποιεί εξ ορισμού την αγγλική γλώσσα κι έχει τα πάντα μεταφρασμένα και στα ελληνικά, ώστε με το πάτημα του αντίστοιχου συνδέσμου να αλλάζει η γλώσσα της σελίδας.
- ***MenuItemBean***: αυτό το managed bean, μαζί φυσικά με τα αντικείμενα της κλάσης `MenuItem`, χειρίζεται και αυτά της `Right`. Ο λόγος είναι ότι δεν υπάρχει ξεχωριστή ιστοσελίδα για τα δικαιώματα καθώς ρυθμίζονται από τη σελίδα των στοιχείων μενού.

4.4. Converters (*πακέτο converters*)

Οι converters (*μετατροπείς*) είναι ένα πολύ χρήσιμο εργαλείο στην ανάπτυξη ιστοσελίδων με JSF, καθώς μετατρέπουν τα αντικείμενα σε αλφαριθμητικά και αντίστροφα, χωρίς να χρειάζεται να γράφεται επιπλέον κώδικας κάθε φορά. Στην εφαρμογή μας για παράδειγμα, όταν χρησιμοποιείται μία λίστα επιλογής σε html (`"<select><option>GROUP1</option><option>GROUP2</option>…</select>"`) στην οποία θέλουμε να φορτώσουμε αντικείμενα της κλάσης Group, τα JSF χρησιμοποιώντας τον registered converter της κλάσης Group, την `GroupConverter.class`, κάνουν αυτόματα τη μετατροπή σε `java.lang.String` (φορτώνοντας στη λίστα ότι ορίζεται στον converter, που, ως είθισται, είναι η έξοδος της μεθόδου `toString()` της κλάσης). Αυτό όμως που γλιτώνει τον προγραμματιστή από πολύ κόπο είναι το αντίστροφο. Όταν αποσταλεί η φόρμα στο διακομιστή, τα JSF και πάλι αυτόματα με τη χρήση του converter μετατρέπουν το αλφαριθμητικό που παραλήφθηκε από το HTTP request σε αντικείμενο της κλάσης Group, ώστε να το χειριστεί απευθείας το Managed bean, χωρίς πρώτα να πρέπει να τρέξει κάποιο ερώτημα στη βάση βασισμένο στο αλφαριθμητικό που έστειλε ο φυλλομετρητής.
Ο μοναδικός converter που χρειάστηκε να υλοποιηθεί στην εφαρμογή είναι ο GroupConverter.

4.5. Validators (*πακέτο validators*)

Οι validators (θα μπορούσαν ίσως να μεταφραστούν ως *επικυρωτές*) επικυρώνουν την είσοδο του χρήστη, ανάλογα με τις απαιτήσεις τις εκάστοτε εφαρμογής. Οι validators μπορούν να χρησιμοποιηθούν σε περιπτώσεις όπου η είσοδος πρέπει να έχει συγκεκριμένη μορφή ή όταν πρέπει να ελεγχθεί κάποια εγγραφή στη βάση ώστε να συγκριθεί με την είσοδο του χρήστη (όπως για παράδειγμα το επιλεγμένο όνομα χρήστη κατά τη διαδικασία εγγραφής νέου χρήστη). Αν και στην πρώτη περίπτωση είναι μάλλον περιττοί (η Javascript «κάνει τη δουλειά» πιο γρήγορα, χωρίς φόρτο στο διακομιστή και χωρίς καμία έκπτωση στην ασφάλεια), στη δεύτερη περίπτωση είναι απαραίτητοι και βολικοί, καθώς και αυτοί γλιτώνουν σε κάποιες περιπτώσεις τον προγραμματιστή από επαναλαμβανόμενο κώδικα.

- ***UsernameValidator***: επικυρώνει τη μοναδικότητα του επιλεγμένου ονόματος χρήστη κατά τη διαδικασία δημιουργίας νέου χρήστη του συστήματος από το διαχειριστή, όπως στο παραπάνω παράδειγμα.
- ***GroupnameValidator***: κάνει το ίδιο με τον προηγούμενο, αλλά για το επιλεγμένο όνομα της ομάδας.
- ***EmailValidator***: και αυτός επικυρώνει τη μοναδικότητα ενός πεδίου σε έναν πίνακα στη βάση και συγκεκριμένα της ηλεκτρονικής διεύθυνσης του νέου χρήστη.
- ***LangnameValidator***: κάθε γλώσσα που εισάγεται στο σύστημα πρέπει να έχει μοναδικό όνομα (π.χ. «Ελληνικά» ή "English").
- ***LangcodeValidator***: επίσης, κάθε γλώσσα θα πρέπει να αναγνωρίζεται από το μοναδικό κωδικό (αντίστοιχα με το παραπάνω παράδειγμα, "EL" ή "EN" ή "EN_GB" κλπ.)
- ***DisplayNameValidator***: κάθε στοιχείο μενού που εισάγεται στο σύστημα πρέπει κι αυτό να έχει μοναδικό όνομα.

Για καθένα από τους παραπάνω validators που εκτελούνται κάθε φορά που ένα αντικείμενο προσπαθεί να αποθηκευτεί στη βάση, υπάρχει και ο αντίστοιχος validator που εκτελείται κάθε φορά που ένα υπάρχον αντικείμενο επιθυμεί να ενημερωθεί. Ο λόγος που δε χρησιμοποιούνται οι ίδιοι validators είναι ότι στην περίπτωση της ενημέρωσης θα πρέπει να ελεγχθεί το σχετικό πεδίο όλων των εγγραφών του αντίστοιχου πίνακα της βάσης και να συγκριθούν με την είσοδο του χρήστη εκτός φυσικά από την εγγραφή που περιέχει το προηγούμενο όνομα.

Σκεφτείτε για παράδειγμα την περίπτωση ο χρήστης να εκκινήσει διαδικασία ενημέρωσης των στοιχείων μίας ομάδας, αλλάξει την περιγραφή της ομάδας αλλά αφήσει ανέπαφο το όνομα της ομάδας. Ο `GroupnameValidator` θα δημιουργούσε πρόβλημα αν το όνομα αυτό υπάρχει ήδη στη βάση! Δυστυχώς, ο τρόπος κατά τον οποίο καλείται ένας validator σε μία JSF σελίδα δεν επιτρέπει τη χρήση παραμέτρων ώστε να ενημερωθεί ο κώδικας για το αν πρόκειται για δημιουργία αντικειμένου ή ενημέρωση, οπότε προτιμήθηκε η λύση των δύο ξεχωριστών validators.

4.6. Φίλτρα (πακέτο *filters*)

Υπάρχει η δυνατότητα να εκτελεστούν κάποιες ενέργειες πριν και μετά τη βασική επεξεργασία ενός HTTP αιτήματος, κάποιες από τις οποίες καθορίζουν εάν θα συνεχιστεί ή όχι η επεξεργασία του αιτήματος, ενώ άλλες χειρίζονται τα εισερχόμενα ή εξερχόμενα δεδομένα σε πιο κατάλληλη μορφές για περαιτέρω επεξεργασία [1]. Η διεπαφή `javax.servlet.Filter`, όπως προδίδει και το όνομά της, διεξάγει ενέργειες φιλτραρίσματος ενός request (αίτησης) ή ενός response (απάντησης) ή ακόμη και των δύο. Υπάρχουν πολλά είδη φίλτρων όπως φίλτρα αυθεντικοποίησης, ελέγχου και logging, κρυπτογράφησης, συμπίεση και άλλα. Η κλάση `AuthFilter` χρησιμοποιείται για να περιορίσει την πρόσβαση στο σύστημα των χρηστών που δεν έχουν συνδεθεί. Όποια URL κι αν προσπαθήσει να προσπελάσει ο χρήστης πριν αναγνωριστεί από το σύστημα, η εφαρμογή θα τον ανακατευθύνει στη σελίδα εισόδου (login page).

4.7. «Ευκολίες» (πακέτο *utilities*)

Στο πακέτο αυτό περιέχονται δύο ακόμα κλάσεις της εφαρμογής οι οποίες είναι γενικότερης φύσης.

- ***Utils***: εδώ υπάρχουν μέθοδοι για πρόσβαση στο τρέχον στιγμιότυπο του κάθε managed bean ξεχωριστά, μέθοδοι για δημιουργία και εμφάνιση JSF μηνυμάτων, μία ευκολία για MD5 hashing (για την αποθήκευση του password κατά τη διαδικασία εγγραφής νέου χρήστη και τη σύγκρισή του κατά τη διαδικασία εισόδου) και άλλο ένα μέλος το οποίο επιστρέφει το συνδεδεμένο χρήστη. Όλες οι μέθοδοι είναι στατικές, πράγμα που σημαίνει ότι δεν δημιουργούνται πουθενά στην εφαρμογή στιγμιότυπα της κλάσης αυτής.
- ***MultiPageMessagesSupport***: χάρη στον κώδικα που συνέγραψαν οι Jesse Wilson και Lincoln Baxter III [11] και μας επιτρέπουν να χρησιμοποιούμε, εκμεταλλευόμενοι έτσι την άνεση που παρέχει η επαναχρησιμοποίηση κώδικα, είναι δυνατή η εμφάνιση JSF μηνυμάτων ακόμα και μετά από ανακατεύθυνση σε άλλη σελίδα. Στα JSF, όταν δημιουργείται ένα μήνυμα, εμφανίζεται μόνο στην τρέχουσα σελίδα. Πολλές φορές όμως, υπάρχει η ανάγκη να παραχθεί και να εμφανιστεί στο χρήστη ένα μήνυμα, όντας παράλληλα απαραίτητο να μεταφερθεί η ροή σε άλλη σελίδα. Η λύση που υλοποιήθηκε αρχικά ήταν να δημιουργηθεί ένα μέλος ονόματι *message*, τύπου

String, μέσα στο `UserBean`, το οποίο μεταβαλλόταν και εμφανιζόταν κατά βούληση. Το στιγμιότυπο του `UserBean` παρέμενε το ίδιο ανεξάρτητα από τη σελίδα στην οποία πηδούσε η ροή του προγράμματος, οπότε αυτό έλυνε βολικά το πρόβλημα. Με τη χρήση όμως της κλάσης `MultiPageMessagesSupport`, δόθηκε η ευκαιρία να εξερευνηθεί και ένα άλλο στοιχείο των JSF, οι ***Phase Listeners***. Πρόκειται για κλάσεις που υλοποιούν τη διεπαφή `javax.faces.event.PhaseListener` και ανάλογα με τη φάση στην οποία βρίσκεται η εφαρμογή JSF (βλέπε ενότητα 2.2.1.1.), εκτελείται ο επιθυμητός κώδικας. Έτσι και η `MultiPageMessagesSupport`, στο τέλος κάθε φάσης αποθηκεύει τα καθολικά (global) JSF μηνύματα για να τα προβάλει τελικά στη φάση *RENDER_RESPONSE*.

4.8. Πολυγλωσσική υποστήριξη – i18n (default πακέτο)

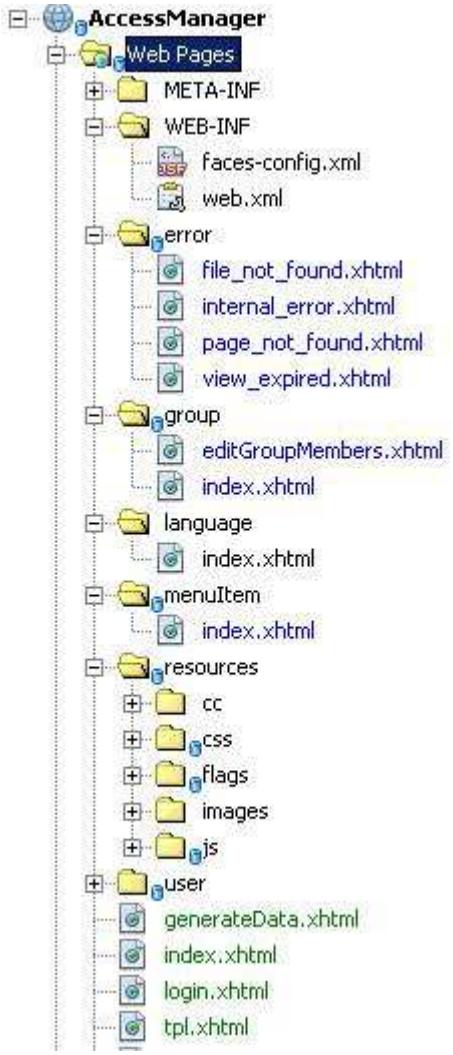
Για τη χρήση δεύτερης γλώσσας στις ιστοσελίδες της εφαρμογής δεν απαιτείται η συγγραφή διπλού κώδικα, αλλά η ύπαρξη ASCII αρχείων με την κατάληξη ".properties".

Υπάρχουν τόσα αρχεία όσες είναι και οι γλώσσες. Εναλλακτικά, μπορεί να δηλωθεί και να χρησιμοποιηθεί ένα εξ ορισμού αρχείο. Η ονομασία των διαφορετικών αρχείων είναι ίδια με το εξ ορισμού αρχείο, με την προσθήκη μίας underscore και του κωδικού της γλώσσας στο τέλος (π.χ. "_el" για ελληνικά).

Στη συγγραφή των σελίδων χρησιμοποιείται ένα αντικείμενο που δείχνει στα αρχεία αυτά και μεταβλητές που δείχνουν σε συγκεκριμένες εγγραφές στο αρχείο αυτό. Για παράδειγμα, γράφουμε `<h:outputLabel for="username" value=#{messages.UsernameLabel}/>` (προσέξτε τις αγκύλες της Expression Language) και όταν η επιλεγμένη γλώσσα είναι ελληνικά θα εμφανιστεί «Όνομα χρήστη», ενώ όταν είναι τα αγγλικά ο χρήστης θα δει "Username".

4.9. Διεπαφή της Εφαρμογής

Η επικοινωνία του χρήστη με την εφαρμογή γίνεται μέσω ενός φυλλομετρητή, στον οποίο φορτώνεται το περιεχόμενο των ιστοσελίδων της από το διακομιστή. Όλα τα αρχεία των ιστοσελίδων, των ρυθμίσεων και των πηγών (resources) βρίσκονται στο φάκελο `web` του root της εφαρμογής και η δομή τους φαίνεται στην Εικόνα 7.



Εικόνα 7 - Η δομή των ιστοσελίδων της εφαρμογής

4.9.1. Web root

Στο web document root βρίσκονται μόνο το αρχείο της αρχικής σελίδας (`./index.xhtml`), το αρχείο προτύπου της εφαρμογής (`./tpl.xhtml`) και η σελίδα εισόδου (`./login.xhtml`).

4.9.2. Φάκελος WEB-INF

Σε αυτόν τον υποκατάλογο υπάρχουν τα αρχεία ρυθμίσεων `faces-config.xml` και `web.xml`. Στο πρώτο ορίζονται, με σύνταξη xml όπως δηλώνει και η κατάληξή του, το αρχείο με τα πολύγλωσσα μηνύματα, οι κανόνες πλοήγησης ανάμεσα στις σελίδες, οι validators, τα managed beans και οι phase listeners. Στο δεύτερο υπάρχουν πολύ περισσότερες ρυθμίσεις, όπως τα servlets (αν χρειάζεται να δηλωθούν ξεχωριστά servlet πέρα από τη βασική servlet των JSF), η αρχική σελίδα, ο χρόνος λήξης του session, οι σελίδες σφαλμάτων (είτε ανά HTTP κωδικό σφάλματος είτε

ανά εξαίρεση που προκαλείται) και τα φίλτρα. Στο `web.xml` μπορεί επίσης να οριστεί με περιγραφικό τρόπο η πολιτική ασφάλειας της εφαρμογής, εάν προτιμηθεί η επιλογή διαχείρισης της ασφάλειας από το web container (glassfish στην παρούσα περίπτωση).

4.9.3. Φάκελος Resources

Είναι οι πηγές των σελίδων, οτιδήποτε άλλο δηλαδή πέρα από τα αρχεία xhtml που χρησιμοποιείται για να παραχθεί ο τελικός HTML κώδικας που θα αποσταλεί στο φυλλομετρητή. Οι υποκατάλογοί του είναι οι εξής:

- css: περιέχει τα cascading style sheets που ρυθμίζουν την εμφάνιση των ιστοσελίδων και περιγράφηκαν στην ενότητα 3.2.
- flags: εδώ αποθηκεύονται οι εικόνες των σημαιών των γλωσσών που εισάγουν οι διαχειριστές στο σύστημα.
- js: περιέχει τα javascript αρχεία όπως κι αυτά περιγράφηκαν στην ενότητα 3.2.
- images: όλα τα απαραίτητα αρχεία εικόνας για την εμφάνιση της εφαρμογής βρίσκονται σε αυτό τον υποκατάλογο (εικόνες για κουμπιά, backgrounds κλπ.).
- cc: τέλος, στο φάκελο cc υπάρχουν τα custom components της εφαρμογής (βλ. ενότητα 3.2.).

4.9.4. Φάκελος user

Περιέχει μόνο το αρχείο διαχείρισης των χρηστών (`./user/index.xhtml`) στο οποίο οι χρήστες με τα απαραίτητα δικαιώματα μπορούν να προβάλλουν τους χρήστες που είναι αποθηκευμένοι στη βάση της εφαρμογής και να δημιουργήσουν καινούριους.

4.9.5. Φάκελος group

Εδώ περιέχονται δύο αρχεία, το `./group/index.xhtml` στο οποίο δίνεται η δυνατότητα διαχείρισης και προβολής των ομάδων και το `./group/editGroupMembers.xhtml` όπου ο χρήστης μπορεί να επεξεργαστεί (προσθέσει ή αφαιρέσει) τα μέλη μίας υπάρχουσας ομάδας. Αρχικά, τα δύο αυτά αρχεία ήταν ενωμένα σε ένα φυσικό αρχείο, αλλά για λόγους απλότητας και αποφυγής πολυπλοκότητας προτιμήθηκε να «σπάσουν» σε δύο ξεχωριστά.

4.9.6. Φάκελος menuItem

Όλη η κεντρική διαχείριση των στοιχείων μενού γίνεται από ένα αρχείο (`./menuItem/index.xhtml`) το οποίο και βρίσκεται σε αυτόν τον κατάλογο. Πέρα από τις ενέργειες CRUD που μπορούν να γίνουν επάνω στα στοιχεία, εδώ γίνεται και ορισμός των δικαιωμάτων των ομάδων σε καθένα από αυτά. Στη σελίδα αυτή χρησιμοποιείται το `tree`, ένα component των primefaces-2.2.1 (μία από τις υλοποιήσεις των JSF) για την παρουσίαση των στοιχείων του μενού υπό τη μορφή δένδρου, μιας και τα JSF δεν παρέχουν κάποιο component που να παρουσιάζει δεδομένα σε δενδρική μορφή.

4.9.7. Φάκελος language

Για τη διαχείριση των γλωσσών του συστήματος, υπεύθυνο είναι το μοναδικό αρχείο αυτού του καταλόγου (`./languages/index.xhtml`). Το συγκεκριμένο αρχείο είναι και ο μοναδικός λόγος που επέβαλε τη χρήση ξεχωριστών βιβλιοθηκών και συγκεκριμένα της tomahawk-2.0, commons-io-2.1 και commons-fileupload-1.2.1 (όλες του Apache Foundation) ώστε να υπάρχει δυνατότητα για ανέβασμα αρχείου στο διακομιστή με τη σημαία που συμβολίζει την κάθε νέα γλώσσα που εισάγεται στο σύστημα, αφού τα JSF δεν περιέχουν κάποιο component για τη διαχείριση των uploads.

ΑΝΑΚΕΦΑΛΑΙΩΣΗ

Στο κεφάλαιο αυτό περιγράφηκε η δομή των αρχείων και ο σκελετός των ιστοσελίδων. Αναλύθηκαν λεπτομερώς όλες οι κλάσεις, κατηγοριοποιημένες σε πακέτα, και οι σελίδες της εφαρμογής, καθώς και κάποια αρχεία με ειδικό σκοπό. Έγινε αναφορά στα πακέτα κλάσεων της java συν το default πακέτο που περιέχει τα αρχεία πολυγλωσσικής υποστήριξης και στους διαφορετικούς φακέλους του εναρκτήριου καταλόγου (web root) της εφαρμογής.

5. ΑΞΙΟΛΟΓΗΣΗ

Όπως αναφέρθηκε στο εισαγωγικό κεφάλαιο, ο στόχος της παρούσας πτυχιακής ήταν συγκεκριμένος και διπλός:

- η εξοικείωση με τεχνολογίες και πρότυπα όπως το MVC, το 3-tier architecture, η Java, και τα JSF,
- και η δημιουργία μίας εφαρμογής διαχείρισης δικαιωμάτων πρόσβασης.

Στο κεφάλαιο αυτό θα αξιολογηθεί η εφαρμογή με βάση τους αντικειμενικούς της στόχους.

5.1. ΑΞΙΟΛΟΓΗΣΗ ΩΣ ΠΡΟΣ ΤΟ ΜΟΝΤΕΛΟ MVC

Στην ενότητα 2.1.1.περιγράφτηκαν με λεπτομέρεια οι βασικές έννοιες που αφορούν στο σχεδιαστικό πρότυπο MVC, καθώς και οι ανάγκες που οδήγησαν στη δημιουργία του. Μπορεί κανείς να πει ότι διαχωρίζει με σαφήνεια τους τρεις ρόλους του μοντέλου, της παρουσίασης και του ελεγκτή. Μεταφράζοντας τα προγραμματιστικά αντικείμενα που χρησιμοποιήθηκαν στην συγκεκριμένη εφαρμογή, τη μοντελοποίηση αναλαμβάνουν οι οντότητες εκφρασμένες σε κλάσεις Java και τα Enterprise Java Beans (EJB), η παρουσίαση είναι ευθύνη των σελίδων JSF και τα Managed Beans χειρίζονται την είσοδο του χρήστη και τις επικοινωνία μεταξύ της βάσης δεδομένων (μοντέλου) με τη διεπαφή χρήστη (παρουσίαση).

Ειδικά για το πόσο διαχωρισμένο είναι το κομμάτι της παρουσίασης, αρκεί να δειχθεί πώς μπορούν τα ίδια δεδομένα να εμφανιστούν με εντελώς διαφορετικούς τρόπους, χωρίς να γίνει καμία κλήση σε κώδικα της επιχειρησιακής λογικής και επίσης χωρίς να γραφεί διπλός κώδικας για καμία σελίδα.

Στην εφαρμογή υπάρχουν τρία css αρχεία, το `cssLayout.css` που φορτώνεται πάντα και ορίζει την εμφάνιση και το διαχωρισμό των τμημάτων της σελίδας μέσα στο template (`tpl.xhtml`) και δύο εναλλακτικά αρχεία για υλοποίηση ξεχωριστού στυλ εμφάνισης (Εικόνα 8, Εικόνα 9).

Εικόνα 8 - Το βασικό στυλ εμφάνισης της εφαρμογής

Εικόνα 9 - Το εναλλακτικό στυλ εμφάνισης της εφαρμογής

Η δυνατότητα αλλαγής των χρωμάτων και του όλου θέματος της εφαρμογής έχει υλοποιηθεί με τέτοιο τρόπο ώστε να προσφέρει τη μέγιστη ευκολία στον χρήστη. Συγκεκριμένα, με το πάτημα ενός συνδέσμου στο πάνω δεξιό άκρο κάθε σελίδας, ο χρήστης μπορεί να αλλάξει την εμφάνιση των χρωμάτων της εφαρμογής στιγμιαία, κάτι που όπως φαίνεται και από τις παραπάνω εικόνες, δε θα επηρεάσει φυσικά καθόλου το περιεχόμενο της σελίδας.

Η τεχνική αυτή εφαρμόζεται σε μεγάλες εφαρμογές, όπου προσφέρονται πολύ περισσότερα στυλ εμφάνισης στους χρήστες για να επιλέξουν.

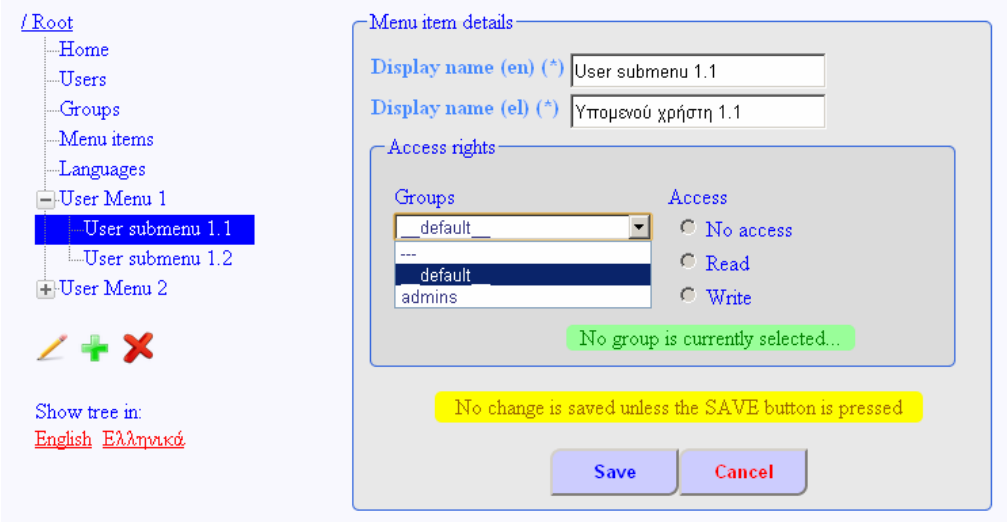
Παρακάτω θα στο κεφάλαιο αυτό θα επεξηγήσουμε την εφαρμογή και θα δείξουμε το πώς αυτή πληροί τις προδιαγραφές της εκφώνησης.

5.2. Αξιολόγηση ως προς 3-tier Αρχιτεκτονική

Όπως αναλύθηκε στην ενότητα 2.1.2., η αρχιτεκτονική επιπέδων επιτρέπει τη διανομή των στοιχείων του λογισμικού και του υλικού σε περισσότερους από έναν υπολογιστές. Η συγκεκριμένη εφαρμογή εύκολα χωρίζεται σε τρία φυσικά επίπεδα, αφού μπορούμε να έχουμε σε έναν υπολογιστή τον application server (Glassfish), σε άλλο υπολογιστή το DBMS (MySQL) και φυσικά να έχουμε πρόσβαση στη διεπαφή της εφαρμογής από οποιονδήποτε υπολογιστή διαθέτει σύνδεση στο διαδίκτυο και ένα φυλλομετρητή. Κατά τη διάρκεια της ανάπτυξης μάλιστα, δοκιμάστηκε με επιτυχία το μοντέλο αυτό, καθώς στήθηκε σε τοπικό μηχάνημα ο application server αλλά σε απομακρυσμένο υπολογιστή το DMBS και φυσικά η εφαρμογή ήταν προσβάσιμη από οπουδήποτε στον κόσμο (με τη χρήση dynamic DNS).

5.3. Αξιολόγηση Εφαρμογής

Δείχτηκε ήδη πώς επετεύχθη ο διαχωρισμός μεταξύ μοντέλου και παρουσίασης και πώς γίνεται η εφαρμογή αυτή να κατανεμηθεί σε τρία επίπεδα. Η εφαρμογή όμως, είχε κι έναν αντικειμενικό στόχο: να παράσχει ένα εργαλείο διαχείρισης δικαιωμάτων πρόσβασης. Από τις σελίδες του συστήματος, δίνεται η δυνατότητα στο χρήστη να δημιουργήσει και να επεξεργαστεί χρήστες και να τους κατανείμει σε ομάδες. Υπενθυμίζεται εδώ ότι όλα ανεξαιρέτως τα μέλη μίας ομάδας μοιράζονται τα ίδια δικαιώματα σε όλα τα στοιχεία μενού. Επομένως, παρέχεται και η λειτουργία ανάθεσης δικαιωμάτων ομάδας πάνω σε στοιχεία του μενού (Εικόνα 10), με αποτέλεσμα τη διάκριση των επιτρεπόμενων λειτουργιών ανάμεσα σε χρήστες δύο



Εικόνα 10 - Επιλογή ομάδας για ανάθεση δικαιωμάτων σε στοιχείο μενού

διαφορετικών ομάδων. Συγκεκριμένα, ο χρήστης της περίπτωσης που απεικονίζεται στην Εικόνα 11, επιτρέπεται να δημιουργήσει μία νέα ομάδα, καθώς και να επεξεργαστεί τις πληροφορίες της και ακόμα και να τη διαγράψει (επειδή έτσι έχουν οριστεί τα δικαιώματα της ομάδας στην οποία ανήκει). Στην περίπτωση της Εικόνα 12 όμως, παρατηρείται η απουσία των αντίστοιχων με την προηγούμενη εικόνα συνδέσμων που εκτελούν τις προαναφερθείσες λειτουργίες.

Εικόνα 11 - Στο συγκεκριμένο χρήστη επιτρέπονται λειτουργίες εγγραφής

Εικόνα 12 - Σε αυτήν την περίπτωση, ο χρήστης έχει μόνο δικαιώματα ανάγνωσης

Παράλληλα με την παραπάνω αξιολόγηση για πιο επισταμένη αξιολόγηση της εφαρμογής έγιναν περαιτέρω έλεγχοι οι οποίοι φέρνουν τον χρήστη στο κέντρο της διαδικασίας χρήσης του.

Η αξιολόγηση αυτή βασίζεται σε αξιοποίηση των κανόνων ευχρηστίας οι οποίοι γράφτηκαν από τον Jakob Nielsen [18] και αποτελούν απαύγασμα καλών

προγραμματιστικών πρακτικών, με στόχο την παραγωγή λογισμικού φιλικού προς το χρήστη, υπό την έννοια της ευχρηστίας.

5.4. ΤΥΠΙΚΟΙ ΕΥΡΕΤΙΚΟΙ ΚΑΝΟΝΕΣ ΕΥΧΡΗΣΤΙΑΣ

Ως προς τη διεπαφή χρήστη (UI), η εφαρμογή θα έπρεπε να συμφωνεί με τους κανόνες που έθεσε ο Jakob Nielsen περί ευχρηστίας λογισμικού. Στη συνέχεια παρατίθενται οι κανόνες αυτοί και ελέγχεται η πιστότητα της εφαρμογής με αυτούς.

1. **Ορατότητα Κατάστασης Συστήματος (Visibility of System Status)**: «*Το σύστημα πρέπει να παρέχει συνεχή ανάδραση στο χρήστη σχετικά με το τι συμβαίνει σε εύλογο χρονικό διάστημα*».

Όλη η αλληλεπίδραση του χρήστη με τις διάφορες επιλογές του μενού ελέγχεται και κατευθύνεται αναλόγως. Σαφή μηνύματα κατευθύνουν τον χρήστη για την επόμενη ενέργειά του ή περιγράφουν τα λάθη στις ενέργειές του. Όταν υλοποιείται μία ενέργεια εμφανίζεται μήνυμα σε πράσινο φόντο με το οποίο ο χρήστης ενημερώνεται ότι έγινε αυτό ακριβώς που ήθελε να γίνει (Εικόνα 13). Από την άλλη, όταν προκύψει κάποιο σφάλμα, ο χρήστης ενημερώνεται για την αποτυχία εκτέλεσης της ενέργειας με μήνυμα σε κόκκινο ή κίτρινο φόντο (ανάλογα με την κρισιμότητα), ώστε να του τραβήξει την προσοχή και να το διορθώσει πριν συνεχίσει (Εικόνα 14).

Εικόνα 13 - Μήνυμα επιτυχούς δημιουργίας χρήστη

Εικόνα 14 - Μήνυμα αποτυχίας διαγραφής ομάδας η οποία περιέχει μέλη

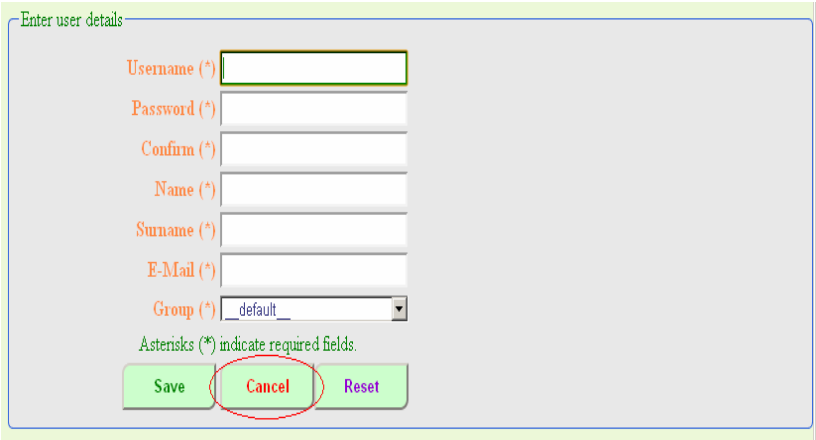
2. **Αναλογία του συστήματος με τον πραγματικό κόσμο (Match between the system and the real world)**: «*Το σύστημα πρέπει να μιλά τη γλώσσα του χρήστη με λέξεις, φράσεις και σύμβολα οικεία σε αυτόν…*».

Έγινε προσπάθεια να αποφευχθούν οι εξεζητημένοι όροι της πληροφορικής και να χρησιμοποιηθεί γλώσσα φυσική, καθημερινή, μαζί με σύμβολα που κάθε άλλο παρά διφορούμενα μπορεί να είναι. Αυτό ισχύει σε κάθε μενού και σε κάθε σε κάθε σελίδα

ανεξαιρέτως, με αποτέλεσμα η εφαρμογή να είναι προσιτή και εύχρηστη στον καθένα και όχι μόνο σε έμπειρους χρήστες.

3. **Έλεγχος και ελευθερία κίνησης στο χρήστη (User control and freedom)**: «*Οι χρήστες συχνά επιλέγουν λειτουργίες εκ παραδρομής και χρειάζονται μία σαφή 'έξοδο κινδύνου' ώστε να αποφύγουν την ανεπιθύμητη κατάσταση χωρίς εκτενής διαλόγους*».

Σε κάθε οθόνη δίνεται η ευχέρεια στο χρήστη να ακυρώσει κάθε ενέργεια, με μόνο ένα κουμπί (Εικόνα 15).



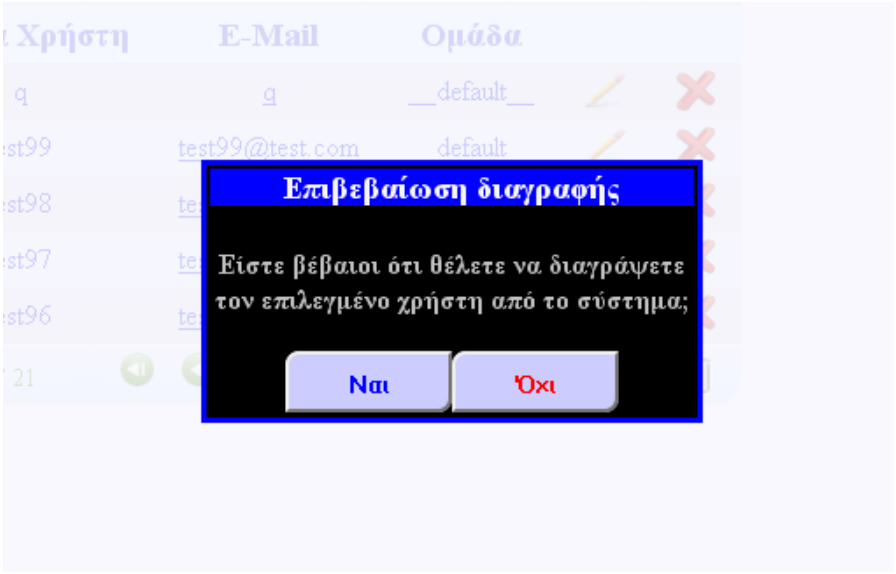
Εικόνα 15 - Δυνατότητα ακύρωσης λειτουργίας με το πάτημα ενός κουμπιού

4. **Συνέπεια και πρότυπα (Consistency and standards)**: «*Οι χρήστες δεν πρέπει να αναρωτιούνται αν δύο ή περισσότεροι διαφορετικοί όροι σημαίνουν το ίδιο πράγμα*».

Όλα τα αντικείμενα εμφανίζονται σε κάθε σελίδα πάντα με τον ίδιο τρόπο, και πάντα στην ίδια θέση. Συνεπώς υπάρχει συνέπεια των όρων που χρησιμοποιούνται σε κάθε περίπτωση.

5. **Αποφυγή σφαλμάτων (Error prevention)**: «*Ο σχεδιασμός που αποτρέπει εξ αρχής τα λάθη να συμβούν είναι πολύ καλύτερος από τα μηνύματα σφάλματος. Είτε εξαλείψτε τις καταστάσεις που είναι επιρρεπείς σε λάθη, είτε παράσχετε στο χρήστη την επιλογή επιβεβαίωσης πριν προχωρήσει στην ενέργεια*».

Οι μόνες περιπτώσεις όπου η ενέργεια είναι μη αναστρέψιμη είναι αυτές στις οποίες διαγράφεται κάποιο αντικείμενο (χρήστης, ομάδα χρηστών ή στοιχείο μενού). Και στις τρεις αυτές περιπτώσεις ο χρήστης είναι υποχρεωμένος να επιβεβαιώσει τη διαγραφή, απαντώντας στο παράθυρο διαλόγου, το οποίο από μόνο του επισημαίνει τη σοβαρότητα της κατάστασης καθώς εμφανίζεται και απενεργοποιεί όλο το υπόλοιπο παράθυρο του φυλλομετρητή (Εικόνα 16).



Εικόνα 16 - Επιβεβαίωση διαγραφής χρήστη

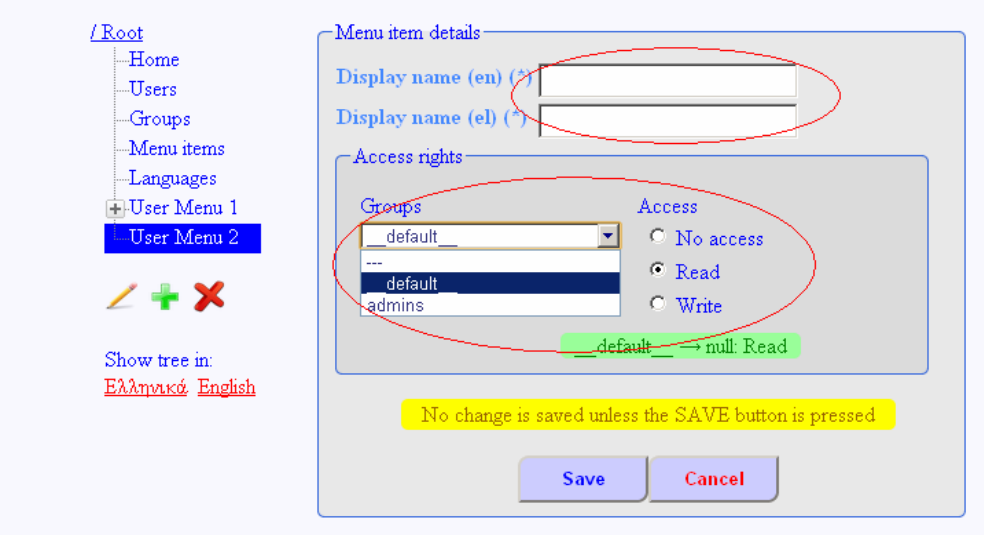
6. **Αναγνώριση παρά ανάκληση από τη μνήμη του χρήστη (Recognition rather than recall)**: «*Ελαχιστοποιήστε το φόρτο μνήμης του χρήστη κάνοντας τα αντικείμενα, τις ενέργειες και τις επιλογές ορατές…*».

Έχοντας πάντα τις ενέργειες που χρειάζεται σε εύκολα να εντοπιστούν σημεία (είτε με το μενού που είναι σταθερό, είτε με τις επιλογές ανά σελίδα), ο χρήστης δε χρειάζεται να θυμάται πρακτικά τίποτα για προηγούμενες ενέργειές του ώστε να προχωρήσει στην επόμενη. Η εμφάνιση πληροφοριακών tooltips όπου κρίθηκε σκόπιμο βοήθησε στο σκοπό αυτό. Η Εικόνα 17 αποδεικνύει όλα τα παραπάνω.

Εικόνα 17 - Οι απαραίτητες πληροφορίες και ενέργειες σε ευδιάκριτο σημείο

7. **Ελαστικότητα και αποτελεσματικότητα χρήσης (Flexibility and efficiency of use)**: «*Συντομεύσεις – αόρατες στον άπειρο χρήστη – μπορούν να επιταχύνουν την αλληλεπίδραση με τον πεπειραμένο χρήστη*».

Η δυνατότητα αυτή έχει προστεθεί στο σημείο που χρειάζεται: στην λειτουργία δημιουργίας στοιχείου μενού δίνεται η δυνατότητα ανάθεσης σε κάθε ομάδα ξεχωριστά δικαιωμάτων πάνω στο προς δημιουργία αντικείμενο, ώστε να μη χρειάζεται να το επεξεργαστεί αργότερα ο χρήστης (Εικόνα 18). Σε αντίθετη περίπτωση, ο χρήστης θα έπρεπε να δημιουργήσει το στοιχείο και μετά να το επιλέξει εκ νέου και να επεξεργαστεί τα δικαιώματα των ομάδων πάνω σε αυτό.



Εικόνα 18 - Δημιουργία στοιχείου μενού με ταυτόχρονη ανάθεση δικαιωμάτων

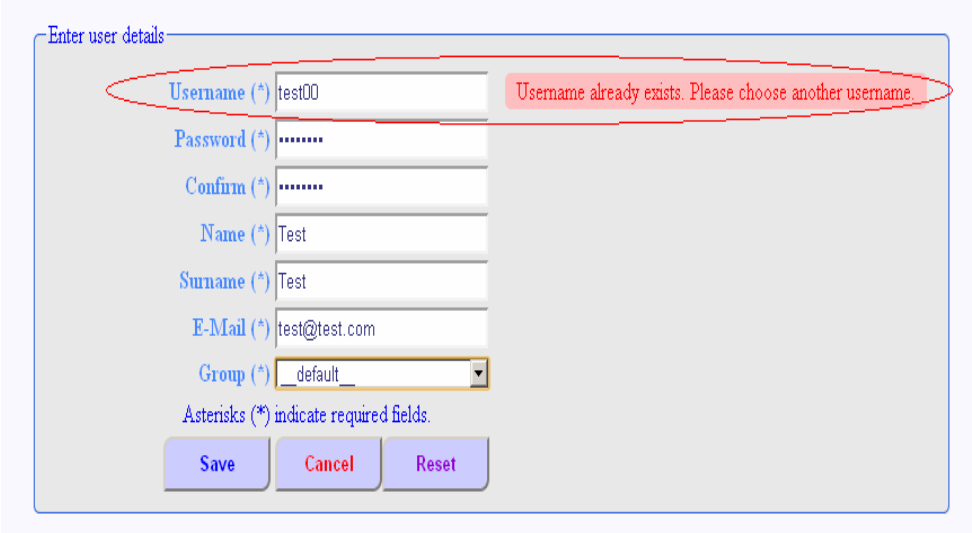
8. **Αισθητική και μινιμαλιστικός σχεδιασμός (Aesthetic and minimalist design)**: «*Οι διάλογοι δε θα πρέπει να περιέχουν άσχετες πληροφορίες ή πληροφορίες που ίσως χρειάζονται σπάνια…*».

Η εφαρμογή είναι λακωνική, χωρίς όμως να στερεί από τον χρήστη τα απαραίτητα δεδομένα για να προχωρήσει σε μία ενέργεια.

9. **Βοηθήστε τους χρήστες να αναγνωρίσουν, διαγνώσουν και διορθώσουν τα σφάλματά τους (Help users recognize, diagnose and recover from errors)**: «*Τα μηνύματα σφάλματος θα πρέπει να εκφράζονται σε απλή γλώσσα, να καταδεικνύουν με σαφήνεια το πρόβλημα και να προτείνουν εποικοδομητικά μία λύση*».

Σε περίπτωση που το λάθος δεν είναι εξωτερικό (π.χ. σφάλμα σύνδεσης με τη βάση ή τον application server), ο χρήστης ενημερώνεται με περιγραφικό τρόπο για το πού εντοπίστηκε το σφάλμα και την ενέργεια στην οποία πρέπει να προβεί για να ανακάμψει. Συγκεκριμένα, σε λάθη συμπλήρωσης στοιχείων σε φόρμες τα μηνύματα εμφανίζονται ακριβώς δίπλα στο πεδίο όπου εντοπίστηκε το λάθος, κάτι που σημαίνει ότι ο χρήστης ίσως να μη χρειαστεί καν να διαβάσει το μήνυμα. Στο παράδειγμα της

Εικόνα 19 φαίνεται ότι κατά τη διαδικασία εγγραφής νέου χρήστη, επιλέχθηκε όνομα χρήστη που χρησιμοποιείται ήδη.



Εικόνα 19 - Υπόδειξη σφάλματος για συγκεκριμένο πεδίο (δημιουργία χρήστη)

10. **Βοήθεια και τεκμηρίωση (Help and documentation)**: «*Αν και είναι καλύτερο το σύστημα να μπορεί να χρησιμοποιηθεί χωρίς τεκμηρίωση, ίσως είναι απαραίτητη η παροχή της. Τέτοιου είδους πληροφορίες πρέπει να είναι εύκολο να τις ψάχνει κανείς, να είναι επικεντρωμένες στις εργασίες του χρήστη, να παραθέτουν συγκεκριμένα βήματα και να μην είναι πολύ ογκώδεις*». Το σύστημα μπορεί εύκολα να χρησιμοποιηθεί από μετρίου επιπέδου (γνώσεων χρήσης υπολογιστή) χρήστες, παρόλα αυτά παρατίθεται στο έγγραφο αυτό ένα σαφές εγχειρίδιο χρήσης για όποιον επιθυμεί να το συμβουλευτεί.

[18]

ΑΝΑΚΕΦΑΛΑΙΩΣΗ

Στο κεφάλαιο αυτό γίνεται αξιολόγηση της πτυχιακής σε σχέση με τους αντικειμενικούς στόχους της. Όπως αποδεικνύεται σχεδιάζεται και υλοποιείται μια εφαρμογή οι οποία οικειοποιείται τις τεχνολογίες αιχμής που απαιτούνται, συγκεκριμένα υλοποιείται μια εφαρμογή που:

- μπορεί να στηθεί σε τρία διαφορετικά επίπεδα (3-tier),
- υλοποιεί το διαχωρισμό της παρουσίασης από τα δεδομένα (MVC),
- χρησιμοποιεί καλές μεθοδολογίες προγραμματισμού (Javascript, AJAX κλπ)

- χρησιμοποιεί καλές πρακτικές προγραμματισμού δημιουργώντας κώδικα που είναι εύκολο να συντηρηθεί και να επεκταθεί καθώς επίσης περιέχει πολλά τμήματα που μπορούν να επαναχρησιμοποιηθούν.

Πολύ περισσότερο, η εφαρμογή πραγματοποιεί αυτά που υπόσχεται και καλύπτει όλες τις βασικές λειτουργίες και δυνατότητες μίας εφαρμογής διαχείρισης δικαιωμάτων.

6. ΣΥΜΠΕΡΑΣΜΑΤΑ - ΜΕΛΛΟΝΤΙΚΟ ΕΡΓΟ

6.1. Συμπεράσματα

Έλεγχος πρόσβασης δεδομένων γίνεται παντού, σε κάθε οργανισμό, μικρό ή μεγάλο, ιδιωτικό ή κρατικό, εμπορικό ή μη κερδοσκοπικό. Ιδιαίτερα στο διαδίκτυο όπου ο όγκος των δεδομένων αλλά και των χρηστών είναι τεράστιος, τα συστήματα ελέγχου πρόσβασης παίζουν καταλυτικό ρόλο στην εύρυθμη λειτουργία του φορέα που τα υλοποιεί. Επιπλέον το λογισμικό εξελίσσεται πιο γρήγορα από ποτέ. Τα βήματα προς τα μπρος του παρελθόντος έχουν γίνει άλματα προόδου τα τελευταία χρόνια, παρέχοντας απεριόριστες δυνατότητες στην ανάπτυξη λογισμικού.

Σκοπός της εργασίας αυτής είναι όχι μόνο η δημιουργία μίας εφαρμογής διαχείρισης δικαιωμάτων, αλλά κυρίως η εξοικείωση και εμπέδωση πρωτοποριακών τεχνολογιών και προτύπων λογισμικού, τα οποία χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής.

Στην πορεία της εκπόνησης της εργασίας και για την απόκτηση θεωρητικού υπόβαθρου, ερευνήθηκαν ενδελεχώς τα πρότυπα MVC και 3-tier αλλά και κάποιες βασικές αρχές ασφάλειας συστημάτων. Για την υλοποίηση της εφαρμογής μελετήθηκαν η γλώσσα J2EE με τα JSF, τα CSS, η Javascript/jQuery και η AJAX.

Στο έγγραφο αναλύθηκαν οι παραπάνω γνώσεις και η χρησιμότητά τους και η πρακτική τους εφαρμογή απεικονίζεται στο προϊόν της πτυχιακής εργασίας, μίας εφαρμογής που εκτελεί όλες τις βασικές λειτουργίες που αναμένονται από ένα σύστημα ανάθεσης και διαχείρισης δικαιωμάτων πρόσβασης.

Το πιο ουσιαστικό όμως αποτέλεσμα της δουλειάς αυτής είναι η γνώση που προσέφερε στο δημιουργό της αλλά και θα προσφέρει σε όσους επιλέξουν να τη συμβουλευτούν για να πάρουν μια γεύση από ένα σύστημα διαχείρισης δικαιωμάτων πρόσβασης ή από την ανάπτυξη λογισμικού με J2EE/MVC γενικότερα.

6.2. Επόμενα Βήματα

Στην εξέλιξή του το έργο αυτό ανοίγει επιπλέον προοπτικές.

Μια πρώτη δυνατότητά του είναι ότι μπορεί να λειτουργήσει σαν μηχανισμός ελέγχου πρόσβασης σε κάθε εφαρμογή.

Σαν συνέχεια, η εφαρμογή αυτή θα μπορούσε και αποτελέσει ένα μικρό πλαίσιο ασφάλειας (security framework), ένα πρόσθετο (plug-in) το οποίο θα έχει τη

δυνατότητα να ορίσει και να ελέγξει εξ ολοκλήρου την πρόσβαση σε εφαρμογές τρίτων.

Είναι δυνατό ακόμα να δημιουργηθούν προγραμματιστικά (δυναμικά) νέα στοιχεία μενού τα οποία θα ανοίγουν φόρμες ορισμένες από το χρήστη που τα δημιούργησε. Για παράδειγμα, θα μπορούσε ο χρήστης να δηλώνει ένα νέο στοιχείο μενού με το όνομά του και τα αντίστοιχα δικαιώματα των ομάδων χρηστών πάνω σε αυτό (όπως γίνεται στην παρούσα εφαρμογή), αλλά με τη διαφορά ότι θα ήταν δυνατό να δηλώσει, κατά τη στιγμή της δημιουργίας, σε ποια σελίδα θα κάνει ανακατευθύνεται ο έλεγχος με το νέο στοιχείο ή αν θα ανοίγει μία φόρμα για συμπλήρωση στοιχείων με πεδία που θα ορίζει πάντα ο χρήστης.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Alur D., Crupi J., Malks D.. (2003). "Core J2EE Patterns: Best Practices and Design Strategies", 2nd edition, Prentice Hall.
2. Fowler M (2003). "Patterns of Enterprise Application Architecture", Addison-Wesley.
3. Johnson R. (2002). "Expert One-on-One J2EE Design and Development (Programmer to Programmer)", Wrox.
4. Ramakrishnan R., Gehrke J (2000), "Database Management systems", 2nd edition, McGraw-Hill.
5. Ariel Ortiz Ramirez. "Three-Tier Architecture". Internet: http://www.linuxjournal.com/article/3508, Jul 01, 2000.
6. Internet: http://java.sun.com/blueprints/corej2eepatterns/Patterns/ServiceLocator.html.
7. Internet: http://msdn.microsoft.com/en-us/library/ff649643.aspx.
8. Internet: http://ootips.org/mvc-pattern.html, May 14, 1998.
9. Internet: http://www.altec.gr/index.php/software/technology/3-tier-client-server.html.
10. Internet: http://www.ananta.com/docs/j2eetutorial14/doc/JSFIntro10.html.
11. Internet: http://www.java2s.com/Open-Source/Java-Document/Content-Management-System/contineo/org/contineo/web/MultiPageMessagesSupport.java.htm
12. http://www.microsoft.com/net
13. http://www.netbeans.com
14. Internet: http://www.oracle.com/technetwork/java/index.html
15. Internet: http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html
16. Internet: http://www.oracle.com/technetwork/java/javaee/overview-140548.html.
17. Internet: http://www.updateguide.gr/glossary.asp?type=3tier.
18. Internet: http://www.useit.com/papers/heuristic/heuristic_list.html
19. Κερκίρη Τάνια, (2006). "Java Προχωρημένες Τεχνικές, Εφαρμογές πελάτη - διακομιστή για διαχείριση της My-SQL σε περιβάλλον Tomcat". Εκδόσεις Κλειδάριθμος

ΠΑΡΑΡΤΗΜΑ Α - ΚΩΔΙΚΑΣ ΔΗΜΙΟΥΡΓΙΑΣ ΠΙΝΑΚΩΝ

```
-----------------------------------------------------
-- CREATE TABLE `Access_Manager`.`Users`
-----------------------------------------------------
 CREATE  TABLE IF NOT EXISTS `Access_Manager`.`Users` (
  `user_id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `group_id` INT UNSIGNED,
  `username` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `surname` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`user_id`),
  UNIQUE KEY `username_UNIQUE` (`username` ASC),
  UNIQUE KEY `email_UNIQUE` (`email` ASC)
);

-----------------------------------------------------
-- CREATE TABLE `Access_Manager`.`Groups`
-----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `Access_Manager`.`Groups` (
   `group_id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
   `description` VARCHAR(255 ,
   `groupname` VARCHAR(45) NOT NULL,
   `created_on` DATE NOT NULL,
   `created_by` INT UNSIGNED NULL,
   PRIMARY KEY (`group_id`),
   UNIQUE KEY `groupname_UNIQUE` (`groupname` ASC),
   INDEX `FK_created_by` (`created_by` ASC),
   CONSTRAINT `FK_created_by`
     FOREIGN KEY (`created_by`)
     REFERENCES `Access_Manager`.`Users` (`user_id`)
     ON DELETE SET NULL
     ON UPDATE CASCADE
);
-----------------------------------------------------
ALTER TABLE `Access_Manager`.`Users`
-----------------------------------------------------
```

```
ALTER TABLE `Access_Manager`.`Users`
  ADD CONSTRAINT `FK_group_id`
    FOREIGN KEY (`group_id`)
    REFERENCES `Access_Manager`.`Groups` (`group_id`);
-------------------------------------------------------
CREATE TABLE `Access_Manager`.`Languages`
-------------------------------------------------------
CREATE  TABLE IF NOT EXISTS `Access_Manager`.`Languages` (
  `language_id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `language_code` CHAR(5) NOT NULL ,
  `language_name` VARCHAR(45) NOT NULL ,
  `flag_image_url` VARCHAR(120) NOT NULL ,
  `sort_order` SMALLINT UNSIGNED NULL,
  PRIMARY KEY (`language_id`),
  UNIQUE KEY `language_code_UNIQUE` (`language_code` ASC),
  UNIQUE KEY `language_name_UNIQUE` (`language_name` ASC),
  UNIQUE KEY `sort_order_UNIQUE` (`sort_order` ASC) );
-------------------------------------------------------
-- CREATE TRIGGER TO INCREASE SORT ORDER BY ONE
-------------------------------------------------------
USE access_manager;
DELIMITER $$
DROP TRIGGER IF EXISTS access_manager.increment_sort_order$$
USE `access_manager`$$

CREATE TRIGGER `access_manager`.`increment_sort_order`
BEFORE INSERT ON `access_manager`.`languages`
FOR EACH ROW
BEGIN
    DECLARE maxOrder SMALLINT;
    SELECT COUNT(*) INTO maxOrder FROM languages;
    SET NEW.sort_order=maxOrder+1;
END$$
DELIMITER ;
-------------------------------------------------------
-- CREATE TABLE `Access_Manager`.`Menu_Items`
-------------------------------------------------------
CREATE TABLE IF NOT EXISTS `Access_Manager`.`Menu_Items` (
  `menu_item_id` INT UNSIGNED NOT NULL,
  `language_id` INT UNSIGNED NOT NULL,
  `display_name` VARCHAR(45) NOT NULL,
```

```
  `parent_menu_item_id` INT UNSIGNED,
  `parent_language_id` INT UNSIGNED,
  PRIMARY KEY (`menu_item_id`, `language_id`),
  INDEX `FK_language` (`language_id` ASC),
  INDEX `FK_parent_menu_item_id` (`parent_menu_item_id` ASC),
  CONSTRAINT `FK_parent1`
    FOREIGN KEY (`parent_menu_item_id`)
    REFERENCES `Access_Manager`.`Menu_Items` (`menu_item_id`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE ,
  CONSTRAINT `FK_language`
    FOREIGN KEY (`language_id`)
    REFERENCES `Access_Manager`.`Languages` (`language_id`)
    ON DELETE RESTRICT
    ON UPDATE CASCADE );
-----------------------------------------------------
-- CREATE TABLE `Access_Manager`.`Rights`
-----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Access_Manager`.`Rights` (
  `right_id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `group_id` INT UNSIGNED NOT NULL ,
  `menu_item_id` INT UNSIGNED NOT NULL ,
  `permission` ENUM('0','1','2') NOT NULL DEFAULT '0' ,
  PRIMARY KEY (`right_id`),
  UNIQUE KEY `unique_group_menu_item` (`group_id`, `menu_item_id`) ,
  INDEX `FK_group` (`group_id` ASC),
  INDEX `FK_menu_item` (`menu_item_id` ASC),
  CONSTRAINT `FK_group`
    FOREIGN KEY (`group_id`)
    REFERENCES `Access_Manager`.`Groups` (`group_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `FK_right_menu_item`
    FOREIGN KEY (`menu_item_id`)
    REFERENCES `Access_Manager`.`Menu_Items` (`menu_item_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

ΟΔΗΓΟΣ ΧΡΗΣΗΣ ΛΟΓΙΣΜΙΚΟΥ

Εγκατάσταση της Εφαρμογής

Προαπαιτούμενα

Για την εγκατάσταση της εφαρμογής απαιτείται προηγουμένως η εγκατάσταση του Glassfish Server και να δημιουργηθεί ένα νέο domain (αν δε δημιουργείται ήδη κατά τη διαδικασία της εγκατάστασης του server). Επίσης, θα πρέπει να υπάρχει στον ίδιο ή σε άλλο υπολογιστή εγκατεστημένος ο MySQL Server ο οποίος να περιέχει μία βάση δεδομένων με το όνομα `access_manager` και να δημιουργηθεί στον Glassfish μέσω του admin console ένα JDBC Resource με το αντίστοιχο connection pool τα οποία να δείχνουν στη συγκεκριμένη βάση. Για το σκοπό αυτό είναι απαραίτητος ο Connector/J (java connector) για το MySQL Server, ένα jar αρχείο το οποίο πρέπει να τοποθετηθεί στο φάκελο lib του domain του Glassfish Server. Τέλος, για να είναι έτοιμη να τρέξει η εφαρμογή μόλις εγκατασταθεί, πρέπει να εκτελεστούν τα sql scripts `create_script.sql` και `initial_data.sql`. Τα script αυτά δημιουργούν τους απαραίτητους για τη λειτουργία του συστήματος πίνακες στη βάση καθώς και αρχικά δεδομένα για να είναι εφικτή η χρήση του εξαρχής. Δημιουργούνται δύο ομάδες, `__default__` και `admins`, και ένας χρήστης που ανήκει στην ομάδα admins με όνομα χρήστη `admin` και κωδικό `test`. Επίσης, παράγεται και το αρχικό μενού της εφαρμογής.

Διαδικασία εγκατάστασης

Η εφαρμογή πακετάρεται σε ένα Web Archive (αρχείο με κατάληξη war) μεγέθους περίπου 13MB. Το αρχείο αυτό πρέπει να φορτωθεί στον Glassfish ακολουθώντας την παρακάτω διαδικασία:

1. Ανοίγουμε την κονσόλα διαχείρισης του Glassfish σε ένα φυλλομετρητή. Εξ ορισμού η κονσόλα ακούει στη θύρα 4848, έτσι για παράδειγμα αρκεί να γράψουμε `http://192.168.178.25:4848`, αντικαθιστώντας βέβαια τη συγκεκριμένη IP με την IP ή το όνομα του υπολογιστή στον οποίο είναι εγκατεστημένος ο Glassfish.
2. Από το μενού που βρίσκεται αριστερά, επιλέγουμε `Applications`. Εμφανίζεται μία λίστα με τις ήδη φορτωμένες εφαρμογές στο server. Εάν πρόκειται για νέα εγκατάσταση, αυτή η λίστα θα είναι κενή.

3. Πατάμε το κουμπί `Deploy`. Στη σελίδα που εμφανίζεται και στο πεδίο `Location` είναι επιλεγμένο το radio button `Packaged File to Be Uploaded to the Server`.
4. Πατάμε το αντίστοιχο κουμπί του φυλλομετρητή για ανέβασμα αρχείου (`browse` για Firefox, `choose file` για Chrome κλπ) και επιλέγουμε τοπικά το αρχείο war της εφαρμογής.
5. Στο πεδίο `Type` επιλέγουμε `Web Application`. Στη νέα φόρμα που προκύπτει αφήνουμε τις επιλογές ως έχουν, εκτός κι αν επιθυμούμε να αλλάξουμε το όνομα της εφαρμογής ή να δώσουμε μία περιγραφή.
6. Τερματίζουμε τη διαδικασία πατώντας OK στο κάτω μέρος της φόρμας. Το αρχείο θα μεταφορτωθεί στο server και θα εμφανιστεί η λίστα με τις φορτωμένες εφαρμογές, η οποία πλέον περιέχει και την εφαρμογή Access Manager.

Η εφαρμογή είναι πλέον προσβάσιμη από οποιονδήποτε φυλλομετρητή στη διεύθυνση `http://<όνομα ή IP υπολογιστή>:<αριθμός http θύρας Glassfish>/AccessManager`, για παράδειγμα `http://192.168.178.25:8080/AccessManager`.

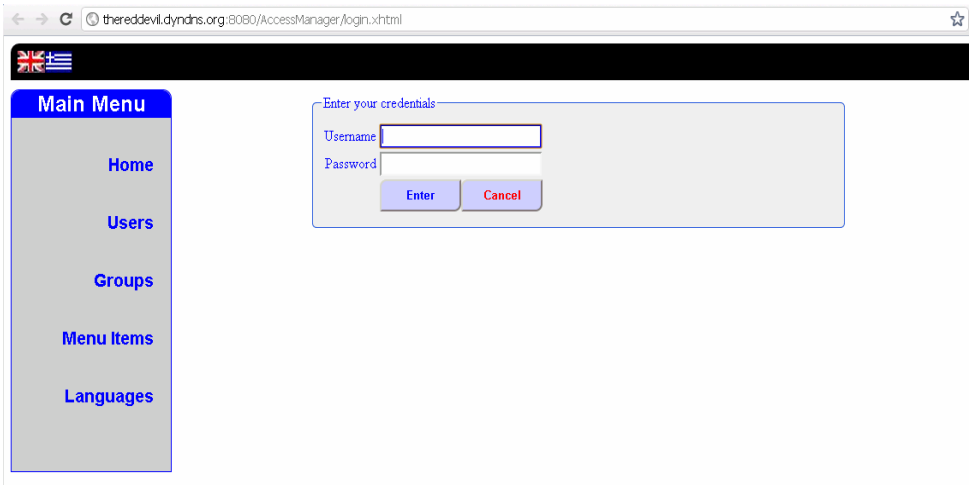
Χρήση της Εφαρμογής

Κάθε σελίδα χρησιμοποιεί ένα συγκεκριμένο πρότυπο εμφάνισης. Το αρχικό μενού της εφαρμογής βρίσκεται πάντα αριστερά σε γκρίζο πλαίσιο. Τα στοιχεία μενού που έχουν εισαχθεί στο σύστημα μετά την εγκατάστασή του βρίσκονται στο επάνω μέρος των σελίδων, κάτω από τις σημαίες για επιλογή γλώσσας εμφάνισης, και στα δεξιά τους βρίσκονται το ονοματεπώνυμο του συνδεδεμένου χρήστη και ο σύνδεσμος αποσύνδεσης. Όλα τα μηνύματα της εφαρμογής εμφανίζονται στο κάτω δεξιό άκρο της οθόνης και είναι ρυθμισμένα να εξαφανίζονται αυτόματα με την πάροδο κάποιων δευτερολέπτων.

Για ορισμένες ενέργειες απαιτούνται τα ανώτερα δικαιώματα που ορίζει το σύστημα, οπότε είναι πιθανό για κάποιους χρήστες μερικοί σύνδεσμοι να είναι απενεργοποιημένοι ή αόρατοι. Σε κάθε ενέργεια που περιγράφεται, αναφέρεται και το επίπεδο πρόσβασης που πρέπει να έχει ο χρήστης για να εκτελέσει την ενέργεια αυτή.

Είσοδος στο σύστημα

Η εφαρμογή AccessManager είναι κλειδωμένη σε εξωτερικούς χρήστες, δηλαδή σε όσους δε διαθέτουν λογαριασμό σύνδεσης στο σύστημα. Για το λόγο αυτό, όποια url της εφαρμογής κι αν εισάγετε στο φυλλομετρητή, το σύστημα θα σας ανακατευθύνει στη σελίδα εισόδου (`login.xhtml`). Εκεί παρέχετε στην εφαρμογή τα διαπιστευτήρια εισόδου σας, δηλαδή το όνομα χρήστη και το συνθηματικό σας, στα αντίστοιχα πεδία (Εικόνα 20).

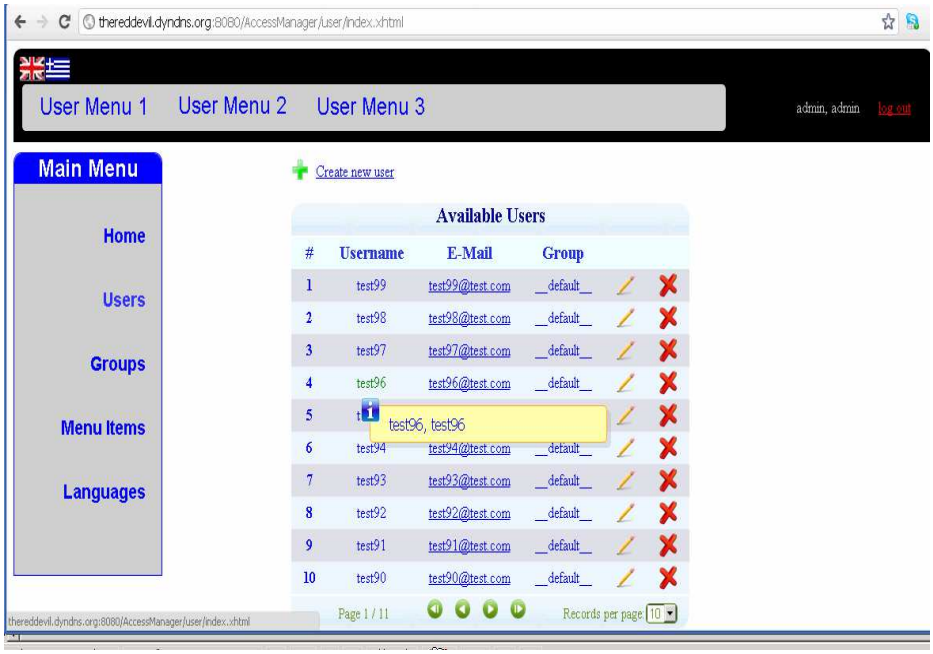


Εικόνα 20 - Η φόρμα εισόδου

Διαχείριση Χρηστών

Προβολή χρηστών

Εφόσον έχετε τουλάχιστον δικαίωμα ανάγνωσης στο μενού `Χρήστες` (`Users`), ο αντίστοιχος σύνδεσμος είναι ενεργοποιημένος στο αριστερό μενού. Πατήστε τον και θα εμφανιστεί ένας πίνακας με όλους τους χρήστες που είναι αποθηκευμένοι στη βάση (Εικόνα 21).



Εικόνα 21 - Προβολή χρηστών

Τα στοιχεία ανά χρήστη περιέχουν βασικές πληροφορίες, όπως το όνομα χρήστη, η ηλεκτρονική του διεύθυνση και η ομάδα στην οποία ανήκει. Για την εμφάνιση του ονοματεπωνύμου του χρήστη σε tooltip, αφήστε το ποντίκι πάνω από το όνομα χρήστη. Για να αποστείλετε ηλεκτρονικό μήνυμα σε κάποιον από τους χρήστες, απλώς πατήστε πάνω στην ηλεκτρονική του διεύθυνση.

Στο κάτω μέρος του πίνακα υπάρχει αρίθμηση σελίδων, κουμπιά πλοήγησης μεταξύ των διαφόρων σελίδων του πίνακα καθώς και επιλογή για το πλήθος των εγγραφών ανά σελίδα που θα εμφανίζονται από το dropdown menu `Records per page` (`Εγγραφές ανά σελίδα`).

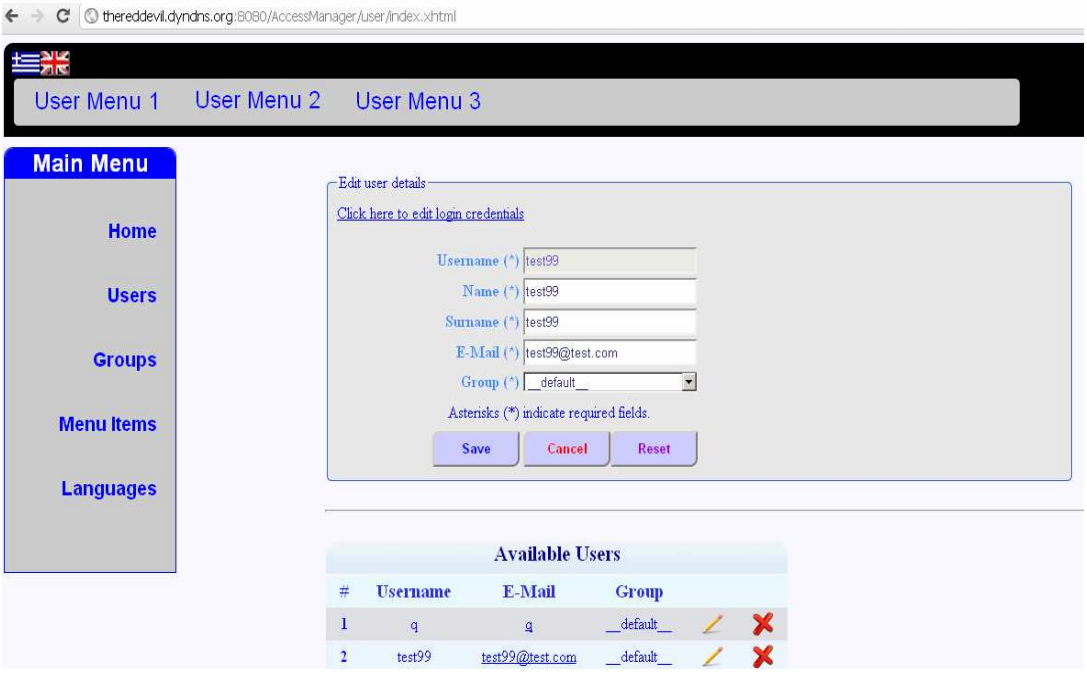
Δημιουργία νέου χρήστη

Εμφανίστε τη σελίδα προβολής χρηστών ακολουθώντας τα βήματα της προηγούμενης ενότητας. Εάν έχετε δικαιώματα εγγραφής, θα εμφανιστεί ο σύνδεσμος `Δημιουργία νέου χρήστη` (`Create new user`) δίπλα σε ένα πράσινο κουμπί  (Εικόνα 21). Αμέσως μόλις το πατήσετε (είτε το κουμπί είτε το σύνδεσμο) εμφανίζεται η φόρμα δημιουργίας χρήστη, όπου συμπληρώνετε τα στοιχεία του νέου χρήστη.

Επεξεργασία στοιχείων χρήστη

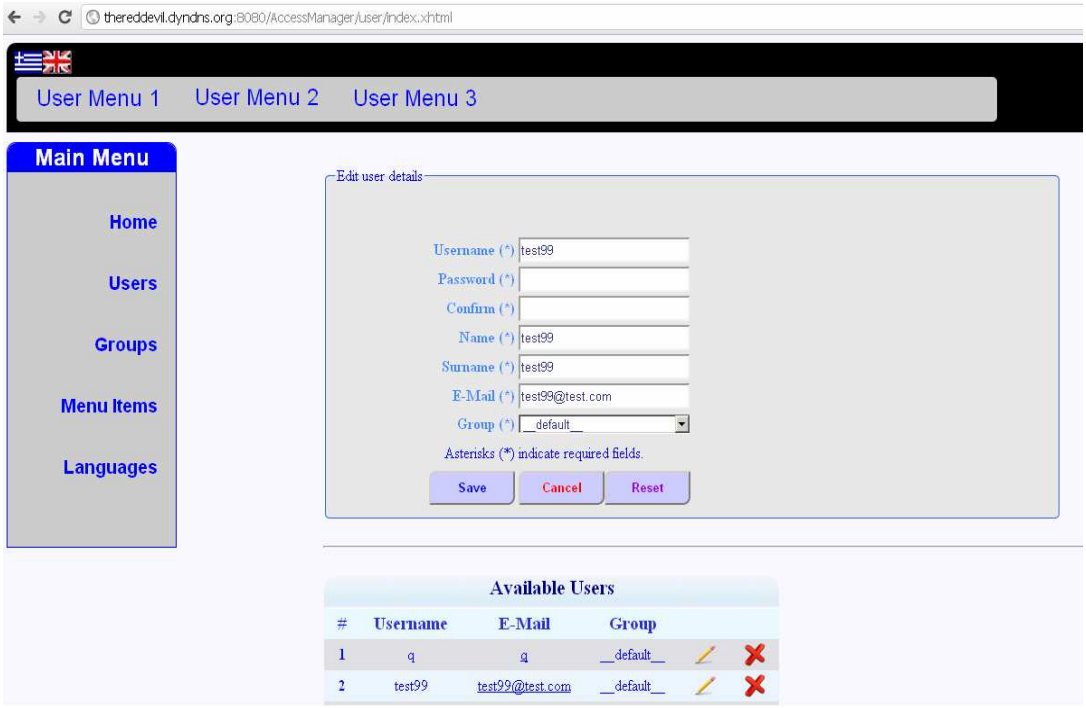
Τα στοιχεία των χρηστών μπορούν να ενημερωθούν από κάποιον χρήστη με δικαιώματα εγγραφής στο μενού χρηστών. Για να εκκινήσετε τη διαδικασία

ενημέρωσης στοιχείων, εμφανίστε τη σελίδα προβολής χρηστών πατώντας στο σύνδεσμο Users (Χρήστες) από το βασικό μενού αριστερά. Πλοηγηθείτε στις σελίδες του πίνακα ώσπου να εμφανιστεί ο προς επεξεργασία χρήστης. Πατήστε το κουμπί  δίπλα από το χρήστη του οποίου τα στοιχεία επιθυμείτε να ενημερώσετε (Εικόνα 21). Αμέσως εμφανίζεται η φόρμα ενημέρωσης στοιχείων χρήστη (Εικόνα 22).



Εικόνα 22 - Φόρμα ενημέρωσης στοιχείων χρήστη

Όπως θα παρατηρήσετε, όλα τα πεδία είναι έτοιμα να δεχτούν νέες τιμές, εκτός του πεδίου Username (Όνομα χρήστη). Επίσης, δεν εμφανίζεται πουθενά πεδίο για αλλαγή του κωδικού εισόδου στο σύστημα (password). Εάν δεν επιθυμείτε να αλλάξετε τα στοιχεία εισόδου του χρήστη παρά μόνο κάποιο ή κάποια από τα πεδία όνομα, επώνυμο, ηλεκτρονική διεύθυνση και ομάδα, κάντε τις αλλαγές σας και πατήστε το κουμπί "Save" («Αποθήκευση»). Εάν σκοπός σας είναι να μεταβάλετε τα στοιχεία εισόδου του χρήστη, πατήστε στο σύνδεσμο "Click here to edit login credentials" («Πατήστε εδώ για να επεξεργαστείτε τα διαπιστευτήρια εισόδου») για να ενεργοποιηθεί η δυνατότητα επεξεργασίας στο πεδίο του ονόματος χρήστη και να εμφανιστούν τα πεδία του κωδικού και της επιβεβαίωσης κωδικού (Εικόνα 23).



Εικόνα 23 - Πλήρης επεξεργασία στοιχείων χρήστη (με όνομα χρήστη και κωδικό)

Διαγραφή χρήστη

Οι υπάρχοντες λογαριασμοί χρηστών μπορούν να διαγραφούν από το σύστημα μόνο από κάποιο χρήστη με δικαιώματα εγγραφής στο μενού χρήστη. Για να διαγράψετε ένα χρήστη, εμφανίστε τη σελίδα προβολής χρηστών και πλοηγηθείτε στη σελίδα του πίνακα που εμφανίζει το χρήστη που θέλετε να διαγράψετε. Πατήστε το κουμπί της στήλης του αντίστοιχου χρήστη για να ξεκινήσετε τη διαδικασία διαγραφής. Το σύστημα θα σας ζητήσει να επιβεβαιώσετε την ενέργειά σας, εμφανίζοντας ένα modal παράθυρο διαλόγου (**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**) στο οποίο θα πρέπει να απαντήσετε καταφατικά εάν επιθυμείτε όντως να διαγράψετε το χρήστη.

Διαχείριση Ομάδων

Προβολή ομάδων

Μπορείτε να προβάλετε τον πίνακα με τις υπάρχουσες ομάδες στο σύστημα, μόνο εάν έχετε τουλάχιστον δικαιώματα ανάγνωσης στο αντίστοιχο μενού. Πατήστε στο σύνδεσμο Ομάδες (Groups) για να εμφανιστεί η σελίδα προβολής των ομάδων του συστήματος (Εικόνα 24). Οι πληροφορίες που περιέχει ο πίνακας είναι το όνομα της ομάδας, ο χρήστης που δημιούργησε την ομάδα και η ημερομηνία δημιουργίας.

Μπορείτε επίσης να εμφανίσετε σε tooltip μία σύντομη περιγραφή της ομάδας (εάν είναι διαθέσιμη) αφήνοντας το ποντίκι πάνω από το όνομα της ομάδας.

Εικόνα 24 - Προβολή ομάδων

Προβολή μελών ομάδας

Για να προβάλετε τα μέλη μίας ομάδας πρέπει να έχετε τουλάχιστον δικαιώματα ανάγνωσης στο μενού των ομάδων. Εμφανίστε τη σελίδα προβολής ομάδων ακολουθώντας τις οδηγίες της παραπάνω διαδικασίας. Πλοηγηθείτε στις σελίδες του πίνακα μέχρι να βρείτε την ομάδα των οποίων τα μέλη θέλετε να εμφανίσετε. Πατήστε το κουμπί  της αντίστοιχης στήλης για να μεταφερθείτε στη σελίδα των μελών της ομάδας (Εικόνα 25). Εδώ υπάρχουν δύο πίνακες. Στο δεξιό φαίνονται τα μέλη που ανήκουν στη συγκεκριμένη ομάδα, ενώ στον αριστερό όλα τα υπόλοιπα.

Εικόνα 25 - Προβολή των μελών της ομάδας __default__

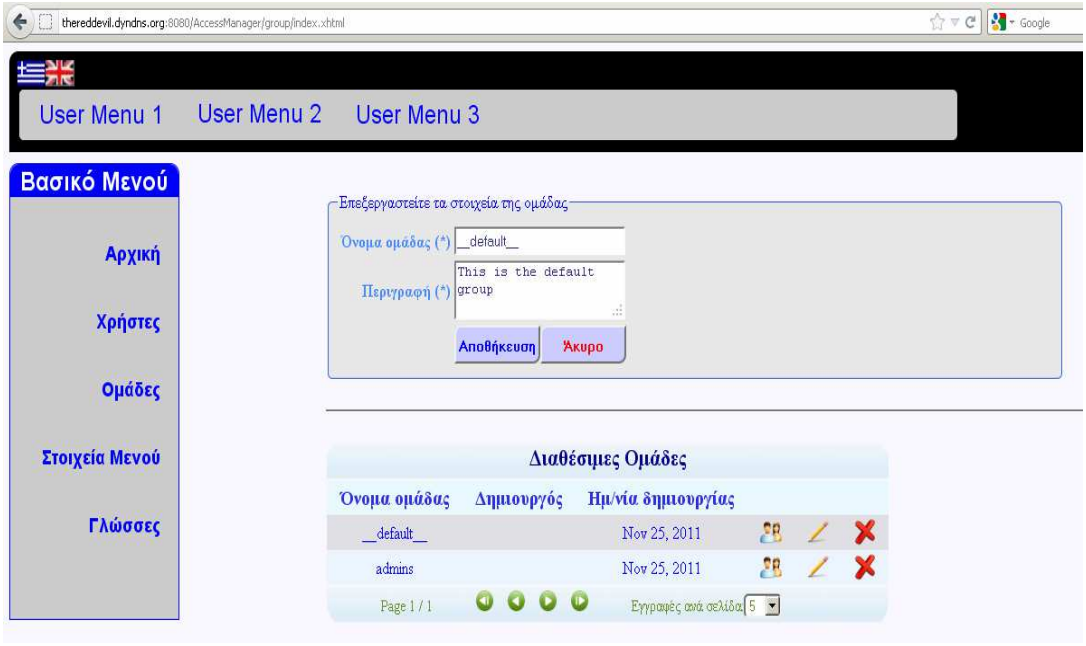
Δημιουργία νέας ομάδας

Η προσθήκη νέας ομάδας στο σύστημα προϋποθέτει να έχετε δικαιώματα εγγραφής στις ομάδες. Αρχικά εμφανίστε τη σελίδα προβολής ομάδων χρησιμοποιώντας το σύνδεσμο "Groups" («Ομάδες») από το βασικό μενού της εφαρμογής. Πάνω από τον πίνακα με τις ομάδες υπάρχει ο σύνδεσμος "Create a new group" («Δημιουργία νέας ομάδας») και αριστερά του το κουμπί , όπως φαίνεται στην Εικόνα 24. Πατώντας στο κουμπί ή στο σύνδεσμο, εμφανίζετε η φόρμα δημιουργίας νέας ομάδας (Εικόνα 26).

Εικόνα 26 - Φόρμα δημιουργίας νέας ομάδας

Επεξεργασία στοιχείων ομάδας

Η επεξεργασία των στοιχείων κάποιας ομάδας απαιτεί να έχετε δικαιώματα εγγραφής στο μενού των ομάδων. Εφόσον πληροίτε αυτήν την προϋπόθεση, εμφανίστε τη σελίδα προβολής ομάδων (Εικόνα 24) από το σύνδεσμο "Groups" («Ομάδες») του βασικού μενού της εφαρμογής. Εκεί εμφανίστε την ομάδα που θέλετε να επεξεργαστείτε και πατήστε το κουμπί της αντίστοιχης στήλης. Αμέσως θα εμφανιστεί η φόρμα ενημέρωσης στοιχείων ομάδας (Εικόνα 27).



Εικόνα 27 - Φόρμα ενημέρωσης στοιχείων ομάδας

Επεξεργασία μελών ομάδας

Για να αλλάξετε τη σύνθεση μίας ομάδας πρέπει να έχετε δικαιώματα εγγραφής στο μενού των ομάδων. Εμφανίστε τη σελίδα προβολής μελών για την ομάδα που επιθυμείτε (Εικόνα 25) ακολουθώντας τις αντίστοιχες οδηγίες. Στο δεξιό πίνακα είναι τα μέλη της ομάδας, ενώ στον αριστερό πίνακα φαίνονται όλοι οι χρήστες που ανήκουν σε άλλες ομάδες, είναι δηλαδή διαθέσιμοι για μεταφορά στη συγκεκριμένη ομάδα. Πατώντας το κουμπί μίας στήλης του αριστερού πίνακα, μεταφέρουμε τον αντίστοιχο χρήστη στην ομάδα που επεξεργαζόμαστε, αφαιρώντας τον ταυτόχρονα βέβαια από την προηγούμενη ομάδα του και οι δύο πίνακες ενημερώνονται αυτόματα. Αντίθετα, πατώντας το κουμπί μίας στήλης του δεξιού πίνακα, διαγράφουμε τον αντίστοιχο χρήστη από την ομάδα που επεξεργαζόμαστε και οι πίνακες και πάλι αντικατοπτρίζουν την αλλαγή. Σε αυτήν την περίπτωση, ο διαγεγραμμένος χρήστης δεν ανήκει σε καμία ομάδα, με αποτέλεσμα να μην έχει κανένα δικαίωμα σε κανένα στοιχείο μενού. Σημειώστε ότι καμία αλλαγή δεν αποθηκεύεται αν δεν πατηθεί το κουμπί "Save" («Αποθήκευση») που βρίσκεται κάτω από τον δεξιό πίνακα με τα μέλη της επεξεργαζόμενης ομάδας. Μόλις δηλαδή πατηθεί το κουμπί "Cancel" («Ακύρωση»), η φόρμα εξαφανίζεται και οι αλλαγές παραβλέπονται. Επίσης, μπορείτε να επαναφέρετε την ομάδα στην αρχική της σύνθεση χωρίς να κλείσετε τη φόρμα επεξεργασίας μελών, πατώντας το κουμπί "Reset", ακυρώνοντας κάθε αλλαγή που είχε γίνει μέχρι το πάτημα του κουμπιού.

Διαγραφή ομάδας

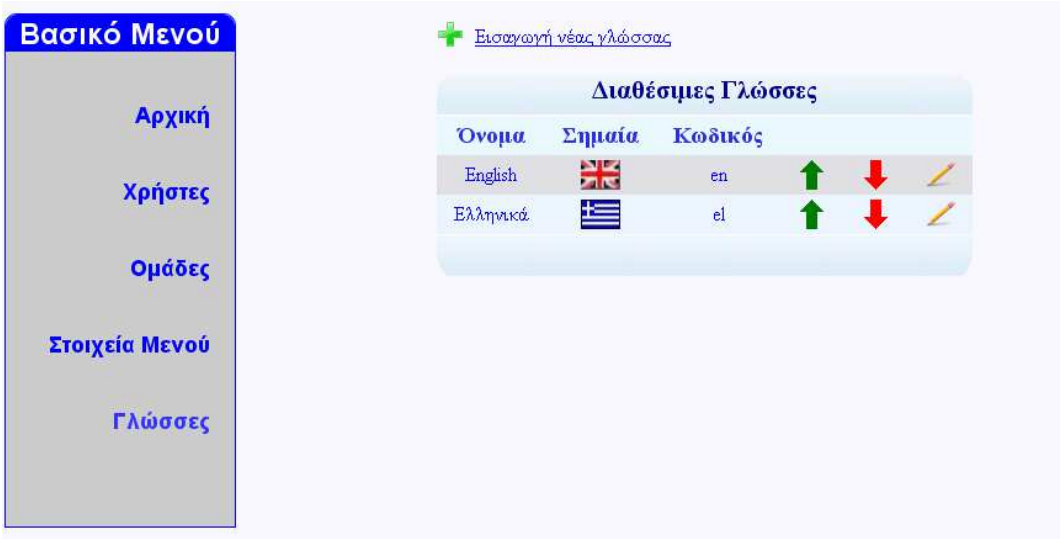
Η διαδικασία διαγραφής μίας ομάδας είναι πανομοιότυπη με την αντίστοιχη διαδικασία για τους χρήστες. Πρωτίστως, πρέπει να έχετε δικαιώματα εγγραφής στις ομάδες για να εμφανίσετε τη φόρμα της Εικόνα 24 και να βλέπετε το κουμπί . Πατήστε αυτό που βρίσκεται στην ίδια στήλη με την ομάδα που επιθυμείτε να διαγράψετε. Θα εμφανιστεί κι εδώ ένα modal παράθυρο διαλόγου, παρόμοιο με της **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**, το οποίο θα σας ζητήσει να επιβεβαιώσετε την ενέργεια διαγραφής. Προσοχή: για να είναι εφικτή η διαγραφή μίας ομάδας, θα πρέπει να μην περιέχει κανένα χρήστη.

Διαχείριση Γλωσσών Εφαρμογής

Η εφαρμογή εγκαθίσταται με δύο γλώσσες, την αγγλική και την ελληνική. Κάθε σελίδα μπορεί να εμφανίσει το περιεχόμενό της σε οποιαδήποτε από αυτές τις δύο γλώσσες. Συγκεκριμένα για τα μενού όμως, μπορούν να οριστούν όποιες και όσες γλώσσες επιθυμούν οι χρήστες της. Καλό θα είναι βέβαια (χωρίς να είναι υποχρεωτικό) να μεταφραστούν στις αντίστοιχες γλώσσες όλα τα μενού (και το βασικό μενού της εφαρμογής, αλλά και τα μενού που έχουν οριστεί από χρήστες). Για το σκοπό αυτό, η εφαρμογή δίνει τη δυνατότητα εισαγωγής νέων γλωσσών.

Προβολή γλωσσών

Για να προβάλετε τις υπάρχουσες γλώσσες της εφαρμογής πρέπει να έχετε τουλάχιστον δικαίωμα ανάγνωσης στο μενού των γλωσσών. Επιλέξτε το σύνδεσμο "Languages" («Γλώσσες») από το βασικό μενού στα αριστερά.



Εικόνα 28 - Προβολή γλωσσών

Εμφανίζονται οι διαθέσιμες γλώσσες της εφαρμογής σε πίνακα με στήλες το όνομά τους, τον κωδικό τους κατά ISO 639-1 και τη σημαία που συμβολίζει την καθεμία (Εικόνα 28).

Εισαγωγή νέας γλώσσας – ρύθμιση σειράς εμφάνισης γλωσσών

Για τις διαδικασίες αυτές επιβάλλεται να έχετε δικαιώματα εγγραφής στο μενού των γλωσσών. Εμφανίστε τη σελίδα προβολής γλωσσών όπως περιγράφεται στην παραπάνω ενότητα. Για να εισάγετε μία νέα γλώσσα στο σύστημα, πατήστε το σύνδεσμο "`Import new language`" («`Εισαγωγή νέας γλώσσας`»), όπως φαίνεται στην Εικόνα 28. Θα παρουσιαστεί η φόρμα εισαγωγής νέας γλώσσας (Εικόνα 29).

Εικόνα 29 - Φόρμα εισαγωγής νέας γλώσσας

Η φόρμα περιέχει τρία υποχρεωτικά πεδία. Πρέπει να δηλώσετε το όνομα της νέας γλώσσας, τον κωδικό της (ISO 639-1) που αποτελείται από δύο λατινικά γράμματα (π.χ. "el" για ελληνικά) και να μεταφορτώσετε στο διακομιστή ένα αρχείο εικόνας το οποίο θα χρησιμοποιείται στη γραμμή επιλογής γλώσσας, στο πάνω μέρος της κάθε σελίδας.

Για να αλλάξετε τη σειρά εμφάνισης των γλωσσών στη γραμμή επιλογής γλώσσας της εφαρμογής, δεν έχετε παρά να πατήσετε στα κουμπιά ⬆ ή ⬇ της αντίστοιχης στήλης και η γλώσσα που μετακινήσατε θα αλλάξει σειρά και στον πίνακα αλλά και στη γραμμή επιλογής γλώσσας στο πάνω μέρος της σελίδας. Σημειώστε ότι η ενέργεια εκτελείται άμεσα, δε χρειάζεται δηλαδή να πατήσετε κάποιο κουμπί για να αποθηκευτούν οι αλλαγές.

Επεξεργασία γλώσσας

Για την ενέργεια αυτή χρειάζεστε δικαιώματα εγγραφής. Ομοίως με τις περιπτώσεις επεξεργασίας χρήστη και ομάδας, εμφανίστε τη σελίδα προβολής γλωσσών και πατήστε το κουμπί  για να εμφανίσετε τη φόρμα επεξεργασίας ομάδας (Εικόνα 30).

Εικόνα 30 - Φόρμα επεξεργασίας λεπτομερειών γλώσσας

Διαχείριση Μενού Χρήστη

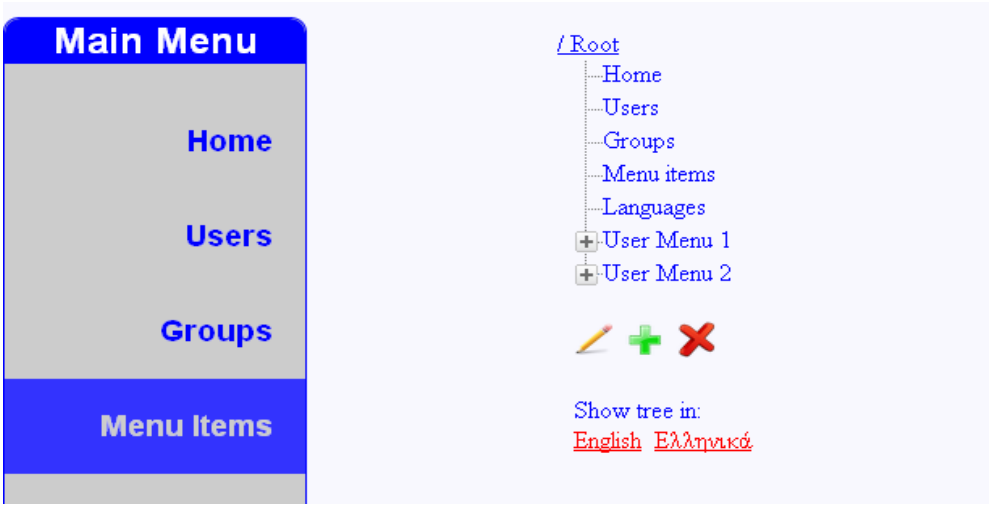
Όπως έχει αναφερθεί ήδη, η εφαρμογή έχει το δικό της μενού το οποίο εμφανίζεται στην αριστερά πλευρά κάθε σελίδας. Παράλληλα όμως, δίνει τη δυνατότητα για δημιουργία προσαρμοσμένων στοιχείων μενού, τα οποία θα φαίνονται στο πάνω μέρος της κάθε σελίδας, κάτω από τη γραμμή επιλογής γλώσσας. Υπάρχει επίσης η δυνατότητα να δημιουργηθούν και στοιχεία μενού εμφωλιασμένα μέσα σε ήδη υπάρχοντα γονεϊκά στοιχεία (Εικόνα 31). Αρκεί να αφήσετε το ποντίκι πάνω από ένα στοιχείο της γραμμής μενού για να δείτε αν έχει ή όχι εμφωλιασμένα μενού. Σημειώστε ότι δεν μπορείτε να δημιουργήσετε δεύτερο επίπεδο εμφωλιασμού, δηλαδή δεν είναι δυνατή η δημιουργία στοιχείου κάτω από στοιχείο που έχει ήδη κάποιο γονέα.

Εικόνα 31 - Εμφωλευμένα μενού κάτω από το μενού *User Submenu 2*

Προβολή στοιχείων μενού

Για να εμφανίσετε τα στοιχεία μενού της εφαρμογής (είτε τα προκαθορισμένα είτε τα καθορισμένα από χρήστες) πρέπει να έχετε δικαιώματα ανάγνωσης στο μενού των στοιχείων μενού. Χρησιμοποιήστε το σύνδεσμο "Menu Items" («Στοιχεία Μενού») από το βασικό μενού και θα μεταβείτε στην αντίστοιχη σελίδα (Εικόνα 32).

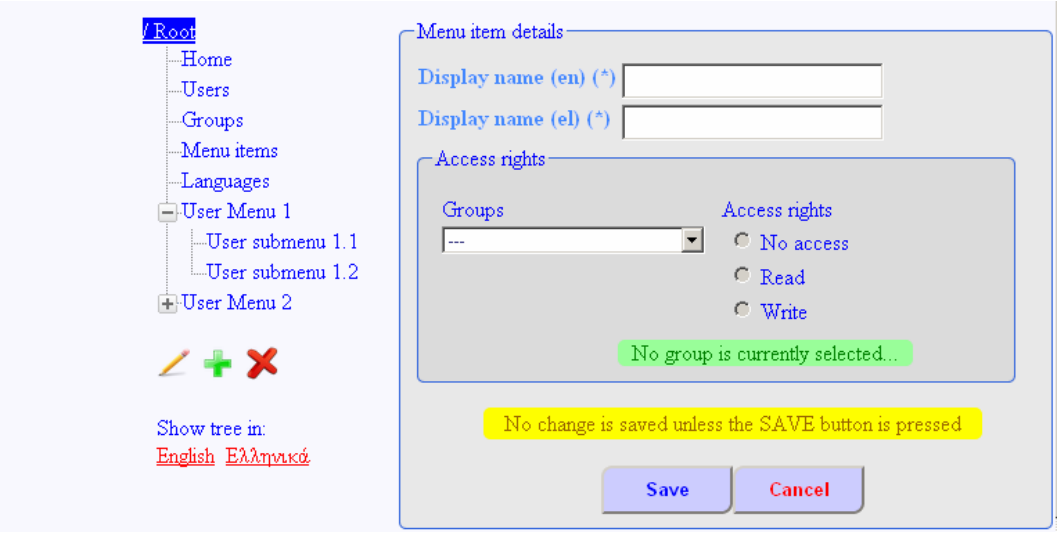
Τα στοιχεία εμφανίζονται με τη μορφή δένδρου, του οποίου η ρίζα συμβολίζεται με "/ Root" («/ Ρίζα») αλλά δεν αντιπροσωπεύει κάποιο στοιχείο, απλά την αρχή της δενδρικής δομής. Δίπλα από κάθε στοιχείο το οποίο περιέχει ένα ή περισσότερα εμφωλιασμένα στοιχεία, εμφανίζεται το σύμβολο «+». Πατώντας το εμφανίζονται τα στοιχεία που εμπεριέχονται στο συγκεκριμένο γονέα. Όσον αφορά τη γλώσσα εμφάνισης του δένδρου, είναι άσχετη με την επιλεγμένη γλώσσα της εφαρμογής και μπορεί εύκολα να αλλάξει πατώντας στον αντίστοιχο σύνδεσμο της γλώσσας που επιθυμούμε (κάτω από το δένδρο).



Εικόνα 32 - Τα στοιχεία μενού της εφαρμογής

Δημιουργία νέου στοιχείου μενού

Για την ενέργεια αυτή χρειάζεστε δικαιώματα εγγραφής στα στοιχεία μενού. Αφού λοιπόν εμφανίσετε τη σελίδα προβολής των στοιχείων μενού (Εικόνα 32), επιλέξτε με μονό κλικ το στοιχείο μέσα στο οποίο επιθυμείτε να εμφωλεύσετε το νέο στοιχείο. Σε περίπτωση που θέλετε στοιχείο πρώτο επιπέδου, επιλέξτε τη ρίζα του δένδρου. Έπειτα, πατήστε το κουμπί  κάτω από το δένδρο για να εμφανιστεί η φόρμα δημιουργίας νέου στοιχείου (Εικόνα 33). Σημειώστε ότι αν επιλέξετε εμφωλιασμένο στοιχείο για να δημιουργήσετε το νέο στοιχείο κάτω από αυτό, η εφαρμογή δε θα σας το επιτρέψει. Επίσης, δεν επιτρέπεται η δημιουργία εμφωλιασμένων στοιχείων κάτω από τα πέντε (5) αρχικά στοιχεία της εφαρμογής (Home, Users, Groups, Menu Items και Languages).



Εικόνα 33 - Φόρμα δημιουργίας νέου στοιχείου μενού

Η φόρμα περιέχει υποχρεωτικά πεδία κειμένου για την ονομασία του στοιχείου σε καθεμία από τις γλώσσες που έχουν οριστεί στην εφαρμογή. Εδώ δηλαδή πρέπει να δώσετε την ονομασία εμφάνισης του νέου στοιχείου σε κάθε διαθέσιμη γλώσσα. για να ολοκληρώσετε τη διαδικασία πρέπει να πατήσετε το κουμπί "Save" («Αποθήκευση»).

(Υπάρχει η προαιρετική δυνατότητα να ορίσετε ταυτόχρονα και τα δικαιώματα που θα έχουν στο νέο στοιχείο οι ομάδες χρηστών. Για την ανάθεση δικαιωμάτων σε ομάδες πάνω σε συγκεκριμένο στοιχείο, διαβάστε παρακάτω στην ενότητα *Ανάθεση δικαιωμάτων*.)

Επεξεργασία στοιχείου μενού

Και εδώ χρειάζεστε δικαιώματα εγγραφής στα στοιχεία μενού. Αφού έχετε εμφανίσει τη δενδρική δομή των στοιχείων, επιλέξτε αυτό που επιθυμείτε να επεξεργαστείτε (επιτρέπεται να επιλέξετε οποιοδήποτε στοιχείο, είτε αρχικό είτε user-defined, εκτός φυσικά από τη ρίζα του δένδρου). Πατήστε το κουμπί  για να εμφανιστεί η φόρμα επεξεργασίας, η οποία είναι ίδια ακριβώς με τη φόρμα δημιουργίας (Εικόνα 33), με τη διαφορά ότι τα πεδία των ονομάτων θα είναι συμπληρωμένα. Ενημερώστε τα ονόματα σε όσες και όποιες γλώσσες επιθυμείτε και πατήστε το κουμπί "Save" («Αποθήκευση»).

Διαγραφή στοιχείου μενού

Τα ανώτερα δικαιώματα (εγγραφής) απαιτούνται και για τη διαγραφή κάποιου στοιχείου. Εμφανίστε το δένδρο με τα στοιχεία και επιλέξτε με μονό κλικ το στοιχείο που επιθυμείτε να διαγράψετε από το user-defined μενού της εφαρμογής. Πατήστε το κουμπί  για να εκκινήσετε τη διαδικασία διαγραφής. Θα σας ζητηθεί επιβεβαίωση της ενέργειας, όπως και στην περίπτωση διαγραφής χρήστη ή ομάδας χρηστών (**Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**). Απαντήστε καταφατικά για να συνεχίσετε και να διαγράψετε το στοιχείο. Θα ενημερωθείτε για την επιτυχή ή όχι εξέλιξη του αιτήματός σας με κατάλληλο μήνυμα. Σημειώστε όμως ότι δεν μπορείτε να αφαιρέσετε κάποιο από τα πέντε (5) αρχικά στοιχεία της εφαρμογής, αλλά ούτε κάποιο γονέα (κάποιο στοιχείο δηλαδή που περιέχει εμφωλευμένα στοιχεία). Στην τελευταία περίπτωση θα πρέπει προηγουμένως να έχετε διαγράψει με την ίδια διαδικασία που περιγράφεται σε αυτήν την ενότητα όλα τα στοιχεία-παιδιά του.

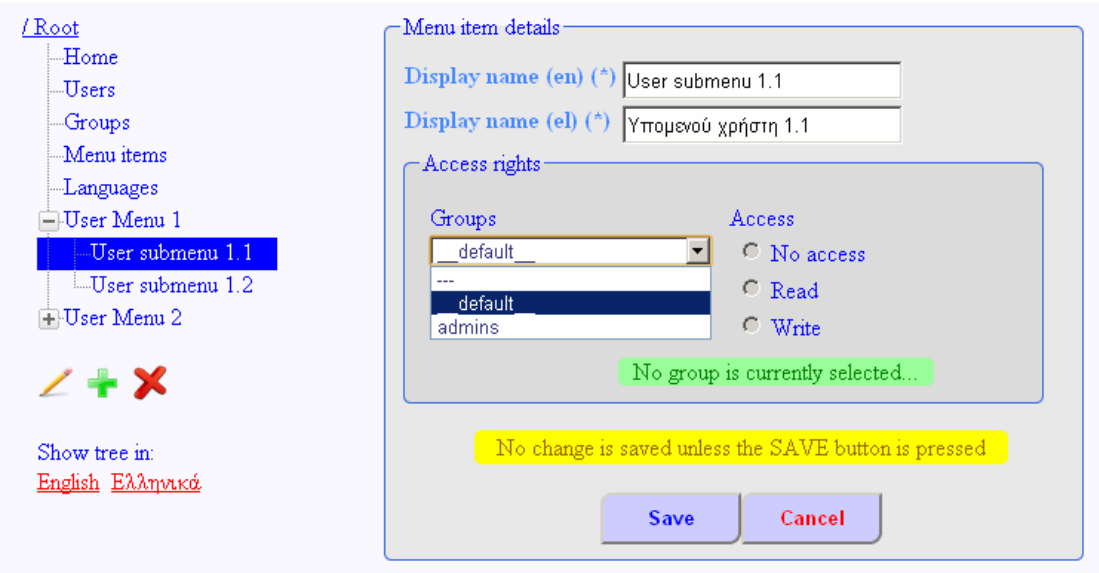
Διαχείριση Δικαιωμάτων

Όπως είναι αναμενόμενο, για να έχει κάποιος χρήστης του συστήματος πρόσβαση στα δικαιώματα των ομάδων χρηστών θα πρέπει και ο ίδιος να έχει δικαίωμα εγγραφής στα στοιχεία μενού. Τα δικαιώματα πάνω σε κάποιο στοιχείο διακρίνονται στα εξής τρία:

1. ***Κανένα δικαίωμα***. Σε αυτή την περίπτωση δεν μπορεί ο χρήστης ούτε να προβάλει δεδομένα ούτε φυσικά και να τα μεταβάλει.
2. ***Δικαίωμα ανάγνωσης***. Εδώ υπάρχει η δυνατότητα προβολής δεδομένων, χωρίς όμως να επιτρέπεται η επεξεργασία.
3. ***Δικαίωμα εγγραφής***. Το ανώτερο δικαίωμα. Η προβολή αλλά και η επεξεργασία και μεταβολή δεδομένων είναι εφικτή.

Ανάθεση δικαιωμάτων

Η ανάθεση δικαιωμάτων σε ομάδα χρηστών πάνω σε συγκεκριμένο στοιχείο μενού μπορεί να γίνει είτε κατά τη στιγμή δημιουργίας του στοιχείου είτε μεταγενέστερα. Η ρύθμιση για τα δικαιώματα που θα έχει κάθε ομάδα πάνω σε ένα στοιχείο γίνεται από τη φόρμα επεξεργασίας ή δημιουργίας ενός στοιχείου μενού. Για να εμφανίσετε αυτή τη φόρμα, ακολουθήστε τις οδηγίες για επεξεργασία ή δημιουργία στοιχείου μενού. Κάτω από τα πεδία ονόματος του στοιχείου, υπάρχει ένα πιο σκούρο πλαίσιο με τίτλο "`Access rights`" («`Δικαιώματα πρόσβασης`»), το οποίο περιέχει ένα drop-down menu με τίτλο "`Groups`" («`Ομάδες`») και ένα σετ από τρία radio buttons με την ετικέτα "`Access`" («`Πρόσβαση`»), καθένα από τα οποία αντιστοιχεί σε ένα από τα τρία προκαθορισμένα επίπεδα πρόσβασης.



Εικόνα 34 - Επιλογή ομάδας για ανάθεση δικαιώματος

Για να οριστεί ένα δικαίωμα πάνω στο προς επεξεργασία στοιχείο, πρέπει πρώτα να επιλεγεί μία ομάδα χρηστών. Μόλις γίνει αυτό, εμφανίζεται επιλεγμένο το radio button που αντιστοιχεί στο επίπεδο πρόσβασης που έχει η επιλεγμένη ομάδα στο τρέχον στοιχείο. Σε περίπτωση που δεν έχουν οριστεί δικαιώματα για την ομάδα αυτή στο τρέχον στοιχείο (είτε γιατί είστε στη δημιουργία ενός νέου στοιχείου είτε γιατί κανείς δεν τα όρισε προηγουμένως), εμφανίζεται επιλεγμένο το εξ ορισμού επίπεδο (No access – Χωρίς πρόσβαση). Μπορείτε να επιλέξετε οποιοδήποτε από τα τρία επίπεδα για την επιλεγμένη ομάδα. Επίσης, μπορείτε να αλλάξετε ομάδα από το drop-down και να ρυθμίσετε το επίπεδο πρόσβασης και για τη νέα επιλεγμένη ομάδα. Η διαδικασία αυτή μπορεί να επαναληφθεί για όσες και όποιες ομάδες χρηστών επιθυμείτε (πάντα πάνω στο τρέχον στοιχείο μενού), αλλά καμία αλλαγή δε θα αποθηκευτεί εάν δεν πατήσετε το κουμπί "Save" («Αποθήκευση»). Μην ξεχνάτε ότι ανά πάσα στιγμή μπορείτε να διακόψετε τη διαδικασία (είτε βρίσκεστε σε δημιουργία νέου είτε σε επεξεργασία υπάρχοντος στοιχείου) πατώντας το κουμπί "Cancel" (Άκυρο»).