ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΚΑΤΕΥΘΥΝΣΗ : ΨΗΦΙΑΚΕΣ ΕΠΙΚΟΙΝΩΝΙΕΣ ΚΑΙ ΔΙΚΤΥΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«ΔΗΜΙΟΥΡΓΙΑ ΔΙΑΔΡΑΣΤΙΚΗΣ e-ΥΠΗΡΕΣΙΑΣ ΥΠΟΣΤΗΡΙΞΗΣ ΣΥΝΑΛΛΑΓΩΝ ΜΕ ΧΡΗΣΗ XML, JAVA & WEB SERVICES»

ΤΣΑΡΑΒΑΣ ΧΑΡΑΛΑΜΠΟΣ - ΜΕ/07074

ΕΠΙΒΛΕΠΩΝ: ΜΑΡΙΝΟΣ ΘΕΜΙΣΤΟΚΛΕΟΥΣ

ΠΕΙΡΑΙΑΣ, ΔΕΚΕΜΒΡΙΟΣ 2009

UNIVERSITY OF PIRAEUS

Department of Digital Systems

POSTGRADUATE PROGRAM

Communication Systems and Networks

Master Course Thesis

«Development of an Interactive e- Service, that is based on XML, JAVA & WEB SERVICES »

(Administration system of postgraduate program)

Tsaravas Charalampos - ME/07074

Supervisor: Marinos Themistocleous

Piraeus, December 2009

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα, Επίκουρο Καθηγητή κ. Μαρίνο Θεμιστοκλέους για την πολύτιμη βοήθεια και καθοδήγησή του καθ'όλη τη διάρκεια εκπόνησης της διπλωματικής αυτής. Ευχαριστώ επίσης τα μέλη της επιτροπής αξιολόγησης κα. Φλώρα Μαλαματένιου και κα. Ανδριάνα Πρέντζα για την αξιολόγηση της διπλωματικής και τις χρήσιμες παρατηρήσεις τους. Τέλος οφείλω ένα μεγάλο ευχαριστώ στους γονείς μου για την υποστήριξη που μου παρείχαν όλα αυτά τα χρόνια.

Τσαραβάς Χαράλαμπος,
22 Δεκεμβρίου 2009

Ευρετήριο Εικόνων

Ευρετήριο Πινάκων

Περίληψη

Η παρούσα εργασία μελετά την εξέλιξη και τη χρήση των τεχνολογιών web services με σκοπό την αυτοματοποίηση των διαδικασιών. Στόχος της εργασίας αυτής είναι η δημιουργία μιας ηλεκτρονικής υπηρεσίας η οποία βασίζεται στην αυτοματοποίηση των διαδικασιών του Προγράμματος Μεταπτυχιακών Σπουδών (Π.Μ.Σ.) του Τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς.

Η εφαρμογή αυτή αποτελεί μια πρόταση για την αυτοματοποίηση και διαχείριση των διάφορων σταδίων από τα οποία περνά ένας υποψήφιος, από την κατάθεση της υποψηφιότητάς του έως την αποδοχή του και την απόκτηση μεταπτυχιακού διπλώματος. Αποτελεί επίσης ένα Portal διαχείρισης των κατευθύνσεων των Π.Μ.Σ. και των εγγεγραμμένων σε αυτές φοιτητών, παρέχοντας έτσι οργάνωση και ευκολία τόσο στους φοιτητές όσο και στην γραμματεία. Επίσης είναι εμφανής και ενεργός και ο ρόλος των καθηγητών, οι οποίοι αποκτούν το δικό τους λογαριασμό και μέσα από τον οποίο μπορούν να βλέπουν τα μαθήματά που τους αντιστοιχούν και να τροποποιούν τα προσωπικά τους στοιχεία. Γενικά οι ρόλοι και οι λειτουργίες των συμμετεχόντων στο σύστημα, προέκυψαν ύστερα απο έρευνα η οποία αφορούσε την ροή δραστηριοτήτων της γραμματείας. Στη συνέχεια θα αναλυθεί και θα παρουσιαστεί η λειτουργία της εφαρμογής με τις απαιτήσεις και τις παραμέτρους σύμφωνα με τις οποίες σχεδιάστηκε.

Η παραπάνω ιδέα υλοποιήθηκε με αρχιτεκτονική client – server, δημιουργώντας αντίστοιχα Server και Client projects JAVA Web Application σύμφωνα με τις αρχές της πλατφόρμας NetBeans IDE 6.7.1, από τα οποία γίνεται και η ανάπτυξη του web service και ολοκληρώνεται τόσο σε επίπεδο Web Application, με το interface του client να αποτελείται από Java Server Pages (jsp) όσο και σε επίπεδο Web Service. Για την αποθήκευση και εξόρυξη των δεδομένων χρησιμοποιείται το σύστημα διαχείρισης βάσεων δεδομένων MySQL.

ΚΕΦΑΛΑΙΟ 1 : ΕΙΣΑΓΩΓΗ

Στο κεφάλαιο αυτό, γίνεται μια εισαγωγή στην τεχνολογία των web services τα οποία ερευνά η παρούσα εργασία και πάνω στα οποία θα βασιστεί στη συνέχεια η για τη δημιουργία μιας εφαρμογής. Επίσης, θα θεσπιστούν οι στόχοι και το αντικείμενο της εργασίας και θα περιγραφεί η δομή του τόμου.

1.1. Εισαγωγή στα Web Services

Η παρούσα εργασία ερευνά τις υπηρεσίες ιστού *(web services)* και πως αυτές μπορούν να απλοποιήσουν και να αυτοματοποιήσουν διάφορες διαδικασίες ενός πληροφοριακού συστήματος. Ένα πληροφοριακό σύστημα αποτελείται από υλικό, λογισμικό, ανθρώπους, δεδομένα και διαδικασίες και αποσκοπεί στην απόκτηση, αποθήκευση, επεξεργασία και διαχείριση πληροφοριών. Με την ανάπτυξη κυρίως των web services αλλά και με την χρήση και άλλων τεχνολογιών, επιτυγχάνεται η απλοποίηση και η αυτοματοποίηση πολύπλοκων, χρονοβόρων, γραφειοκρατικών και δυσλειτουργικών διαδικασιών στους οργανισμούς και στις επιχειρήσεις αλλά παρέχεται και η δυνατότητα στους διαχειριστές να έχουν συγκεντρωμένες και ασφαλείς πληροφορίες.

Στο παρελθόν η δημιουργία και η παροχή υπηρεσιών από επιχειρήσεις στο Internet δεν γίνονταν με έναν προκαθορισμένο και καθολικά αποδεκτό τρόπο με αποτέλεσμα κάθε επιχείρηση να χρησιμοποιεί δικές της μεθόδους. Παρά την ύπαρξη ενός μεγάλου συνόλου διαδικτυακών υπηρεσιών για να είναι δυνατή η χρησιμοποίηση μίας υπηρεσίας, έπρεπε να μελετηθεί ο τρόπος με τον οποίο θα κληθεί, να ελεγχτεί το πρωτόκολλο επικοινωνίας (TCP/IP, HTTP, κλπ) και να προσαρμοστεί όλο το σύστημά του πελάτη της υπηρεσίας ώστε να γίνει συμβατό με αυτό του παροχέα της υπηρεσίας.

Σήμερα με την πρόοδο της τεχνολογίας των υπηρεσιών ιστού οι περισσότερες επιχειρήσεις που δημιουργούν υπηρεσίες στο διαδίκτυο, βασίζονται στην αρχιτεκτονική των υπηρεσιών ιστού, όπως αυτή καθορίζεται από το W3C. (Chappell and Jewell). Αυτό πρακτικά σημαίνει ότι επικρατεί ένας καθολικά αποδεκτός τρόπος δημιουργίας και χρήσης διακτυακών υπηρεσιών και η εκμετάλευσή τους είναι πολύ ευκολότερη και αποδοτικότερη.

1.2. Σκοπός και αντικειμενικοί στόχοι εργασίας

Σκοπός

Στην παρούσα εργασία προτείνεται και υλοποιείται, μια διαδραστική ηλεκτρονική υπηρεσία η οποία αυτοματοποιεί και διευκολύνει την διαδικασία μετάβασης ενός υποψηφίου σε μεταπτυχιακό φοιτητή, ανάλογα με τις κατευθύνσεις και τα Προγράμματα στα οποία γίνεται αποδεκτός και υποστηρίζει (η εφαρμογή) τις δραστηριότητες και τις λειτουργίες της γραμματείας πριν και μετά την επιλογή των υποψηφίων. Η εφαρμογή που αναπτύχθηκε αποτελεί επίσης και ένα portal για τους φοιτητές και τους καθηγητές του Μεταπτυχιακού Προγράμματος, το οποίο περιλαμβάνει τις κατευθύνσεις και μαθήματα των τρέχοντων Πργραμμάτων Μεταπτυχιακών Σπουδών ενώ παράλληλα παρέχει την δυνατότητα καταχώρησης βαθμολογιών και παρουσιών για κάθε φοιτητή. Τέλος, μέσα απο την δικτυακή αυτή εφαρμογή οι φοιτητές έχουν την δυνατότητα αποστολής αιτήσεων στην γραμματεία για την έκδοση πιστοποιητικών σπουδών και αναλυτικών βαθμολογιών.

Αντικειμενικοί στόχοι εργασίας

Οι αντικειμενικοί στόχοι που θέτονται και επιθυμούμε να επιτύχουμε στην παρούσα εργασία είναι:

- Μελέτη βιβλιογραφίας που αφορά την ιδιότητα, τα χαρακτηριστικά και τα οφέλη που προκύπτουν από την χρήση των web services και των σύγρονων τεχνολογιών
- Καταγραφή των σημερινών λειτουργιών Π.Μ.Σ.
- Πρόταση βελτίωσης των λειτουργιών αυτών με τη χρήσης διαγράμματος ροής δραστηριοτήτων
- Ανάλυση, σχεδιασμός και ανάπτυξη συστήματος υποστήριξης και διαχείρισης των Προγραμμάτων Μεταπτυχιακών Σπουδών του Τμήματος,

1.3. Δομή Εργασίας

Σύμφωνα με την έρευνα που πραγματοποιήθηκε, προέκυψε ένα πλάνο για την συγγραφή της παρούσας εργασίας το οποίο περιλαμβάνει 5 κεφάλαια.

Στο κεφάλαιο 1, γίνεται μια εισαγωγή στις τεχνολογίες των υπηρεσιών ιστού που θα αναλυθούν και στη συνέχεια, παρουσιάζονται οι στόχοι και το αντικείμενο της εργασίας.

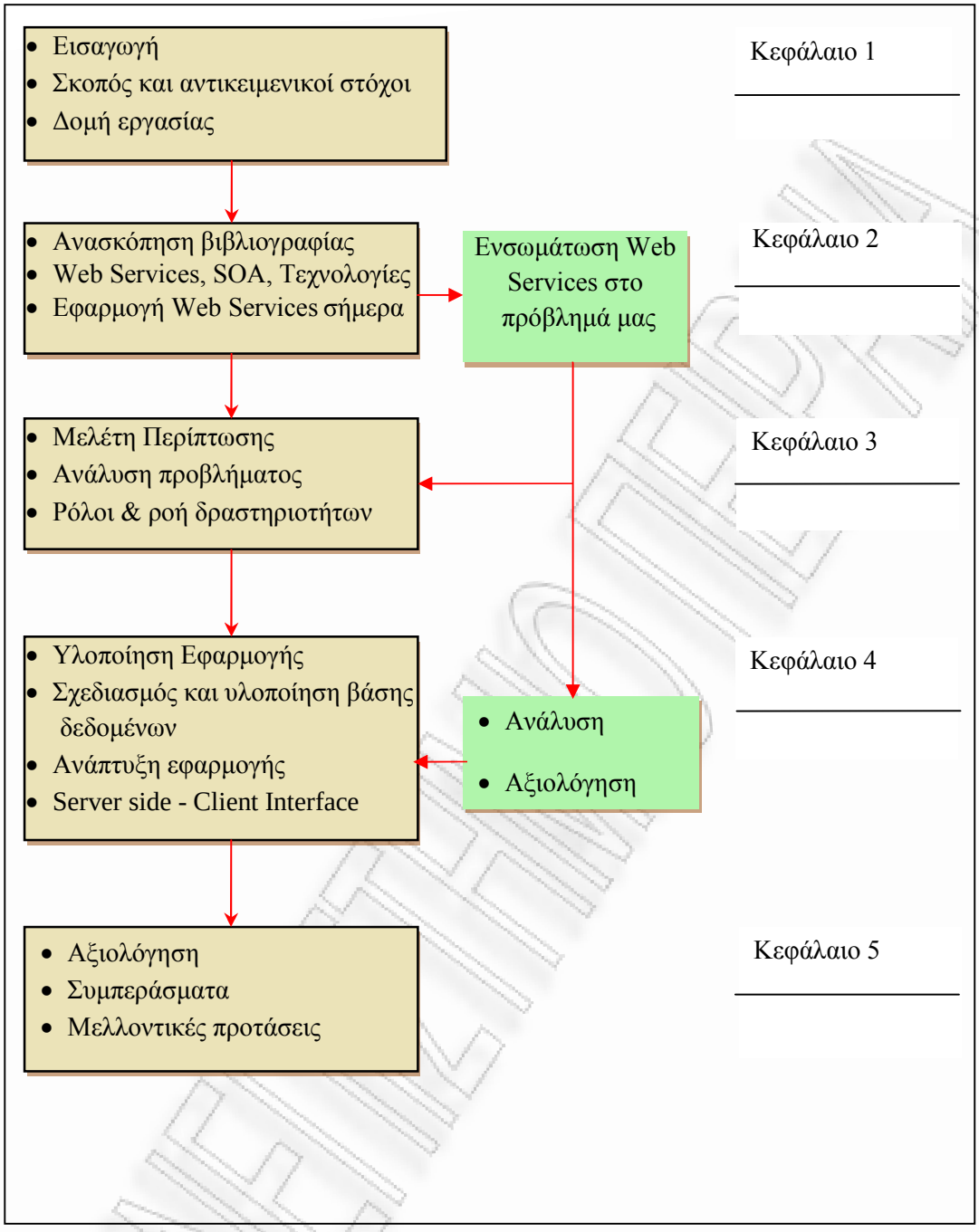
Στο κεφάλαιο 2 γίνεται μια ανασκόπηση της βιβλιογραφίας όσον αφορά τα web services, τα χαρακτηριστικά τους και τα πλεονεκτήματά τους καθώς επίσης και περιπτώσεις όπου βρίσκουν εφαρμογή.

Στο κεφάλαιο 3 αναλύεται η μελέτη περίπτωσης, οι ρόλοι και η ροή δραστηριοτήτων στην οποία βασίστηκε η εφαρμογή.

Στο κεφάλαιο 4 αναλύεται η αρχιτεκτονική του συστήματος και της βάσης δεδομένων καθώς και η ανάπτυξη της εφαρμογής σε προγραμματιστικό επίπεδο.

Στο κεφάλαιο 5, παραθέτονται συμπεράσματα και παρατηρήσεις για το μέλλον και γίνεται μια πρώτη αξιολόγηση της εφαρμογής.

Το παρακάτω διάγραμμα απεικονίζει την δομή στην οποία βασίστηκε η συγγραφή της εργασίας καθώς επίσης και την σύνδεση μεταξύ των κεφαλαίων.



Εικόνα 1: Διάγραμμα δομής εργασίας

ΚΕΦΑΛΑΙΟ 2: ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΑΝΑΣΚΟΠΗΣΗ

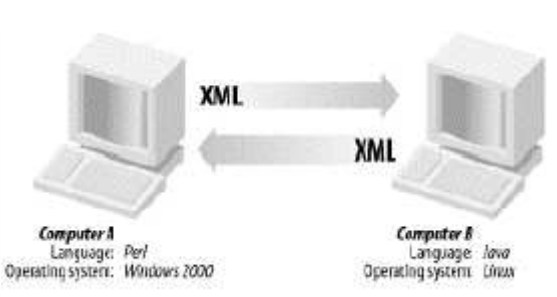
Στο κεφάλαιο αυτό, γίνεται μια ανασκόπησης της βιβλιογραφίας πάνω στις υπηρεσίες ιστού παρουσιάζοντας τεκμηριωμένες θεωρίες και παραδείγματα και αναλύοντας τα χαρακτηριστικά, τα πλεονεκτήματα και τα μειονεκτήματα των web services καθώς και το προγραμματιστικό τους μοντέλο. Στη συνέχεια αναλύεται η αρχιτεκτονική των υπηρεσιών ιστού, τα επίπεδα λειτουργίας της και τα πλεονεκτήματά της. Θα παρουσιαστούν αναλυτικά οι καθιερωμένες τεχνολογίες των web services και τέλος θα αναφερθούν κάποιες περιπτώσεις όπου βρίσκουν εφαρμογή τα web services σήμερα

.

2.1 Web Services – Θεωρητικό Υπόβαθρο

Η IBM ορίζει τα web services ως *μια τεχνολογία που επιτρέπει στις εφαρμογές να επικοινωνούν μεταξύ τους ανεξαρτήτως πλατφόρμας και γλώσσας προγραμματισμού. Ένα web service είναι μια διεπαφή λογισμικού (software interface) που περιγράφει μια συλλογή από λειτουργίες οι οποίες μπορούν να προσεγγιστούν από το δίκτυο μέσω πρότυπων μηνυμάτων XML. Χρησιμοποιεί πρότυπα βασισμένα στη γλώσσα XML για να περιγράψει μία λειτουργία (operation) προς εκτέλεση και τα δεδομένα προς ανταλλαγή με κάποια άλλη εφαρμογή. Μια ομάδα από web services οι οποίες αλληλεπιδρούν μεταξύ τους καθορίζει μια εφαρμογή web service.*

Τα Web Services είναι μια αναδυόμενη τεχνολογία. Δίνουν τη δυνατότητα σε ανόμοιες εφαρμογές οι οποίες τρέχουν σε διαφορετικές μηχανές να ανταλλάσσουν δεδομένα και να συνεργάζονται μεταξύ τους χωρίς τη χρήση επιπρόσθετου λογισμικού ή υλικού. Οι εφαρμογές που βασίζονται σε Web Services μπορούν να ανταλλάσσουν δεδομένα αδιαφορώντας για την γλώσσα προγραμματισμού, το λειτουργικό σύστημα και το πρωτόκολλο διαδικτύου, όπως φαίνεται και από το σχήμα της εικόνας 1.



Εικόνα 2: Απλό web service (Chappell and Jewell, 2002)

Είναι αυτόνομες και ανεξάρτητες εφαρμογές οι οποίες μπορούν να περιγραφούν, διατεθούν, ανατεθούν και εκτελεστούν σε ένα ενιαίο δικτυακό περιβάλλον. Επίσης είναι αυτό-περιγραφικές και αυτόνομες δικτυοκεντρικές μονάδες, εκτελούν ολοκληρωμένες επιχειρησιακές λειτουργίες και υλοποιούνται με ευκολία, επειδή βασίζονται σε κοινά πρότυπα και υπάρχουσες τεχνολογίες όπως α)XML και β)HTTP. Όπως θα δούμε και παρακάτω, τα web services επιτρέπουν στους προγραμματιστές πληροφοριακών συστημάτων να δομήσουν τις εφαρμογές τους χρησιμοποιώντας υπάρχοντα κώδικα που διατίθεται από τρίτους. Τα Web Services αποτελούν λογική συνέχεια των object oriented και component based applications. Παράλληλα τα web services είναι λογισμικές μονάδες διαθέσιμες μέσω ενός δικτύου (internet), οι οποίες δίνουν τη δυνατότητα για ολοκλήρωση εργασιών, επίλυση προβλημάτων ή διεξαγωγή συναλλαγών εκ μέρους ενός χρήστη ή μιας εφαρμογής. (Θεμιστοκλεους, 2009a, Papazoglou, 2008)

Μια υπηρεσία συνοδεύεται και από μία διεπαφή και παρέχει πρόσβαση μέσω ενός συγκεκριμένου μηχανισμού. Η διεπαφή αυτή υλοποιείται με *SOAP/XML,* περιγράφεται με *WSDL* και καταχωρείται και ανακαλύπτεται μέσω UDDI. (Οι προαναφερθείσες τεχνολογίες αναλύονται παρακάτω)

Σε γενικές γραμμές ένα Web Service μπορεί να είναι:

- Μια επιχειρησιακή διαδικασία η οποία δρα αυτόνομα
- Μια ολοκληρωμένη επιχειρησιακή εργασία
- Μια εφαρμογή
- Ένας πόρος παροχής υπηρεσιών

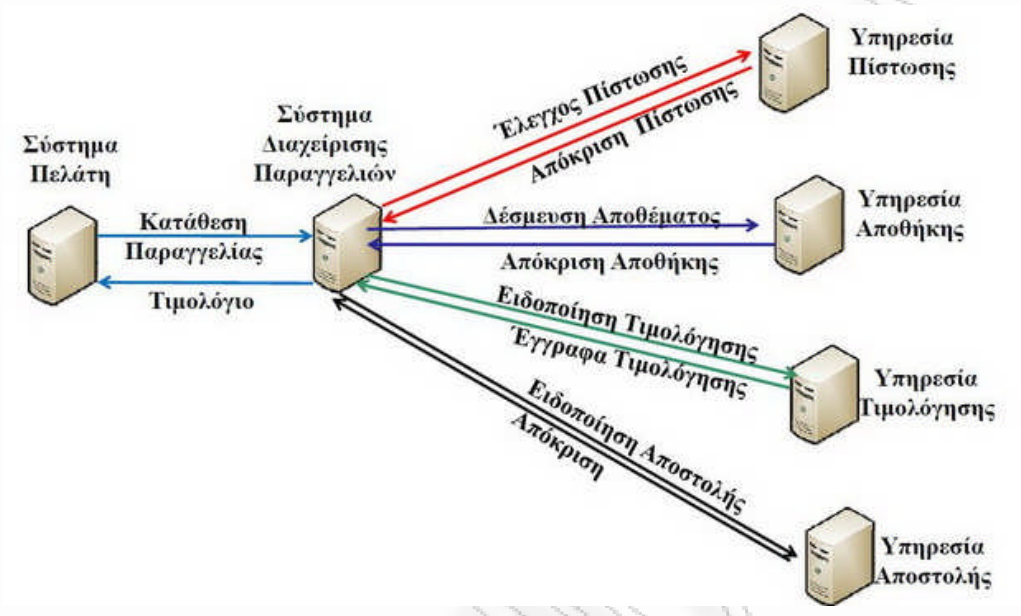
Η παρακάτω εικόνα απεικονίζει ένα απλό παράδειγμα μιας διαδικασίας παραγγελίας με την χρήση web service και δείχνει πώς μπορεί να υλοποιηθεί μια τέτοια διαδικασία από την πλευρά του προμηθευτή στα πλαίσια της αλληλεπίδρασης των web services που αφορούν που εντολές αγοράς, ελέγχους πίστωσης, αυτόματες χρεώσεις, ενημερώσεις αποθήκης και μεταφοράς εμπορευμάτων.

Όταν πραγματοποιηθεί η λήψη μιας παραγγελίας από κάποιον πελάτη, μπορούν και εκτελούνται πολλές λειτουργίες ταυτόχρονα όπως: ο έλεγχος της φερεγγυότητας των χρηστών, η εξακρίβωση ή μη της διαθεσιμότητας ενός προϊόντος στα αποθέματα, ο υπολογισμός της τελικής τιμής και η χρέωση του πελάτη για την εκάστοτε παραγγελία, η επιλογή του μεταφορέα και ο προγραμματισμός της παραγωγής και της αποστολή της παραγγελίας.

Όσο μερικές από τις διαδικασίες εκτελούνται ταυτόχρονα, θα υπάρχουν εξαρτήσεις συγχρονισμού μεταξύ αυτών των διαδικασιών. Για παράδειγμα, πρέπει πρώτα να εξακριβώνεται η εγκυρότητα του χρήστη πριν γίνει αποδεκτή η παραγγελία, επίσης για να υπολογιστεί το τελικό κόστος της παραγγελίας θα πρέπει να υπολογιστούν και τα έξοδα μεταφοράς και τέλος για να εκπληρωθεί το πλήρες πρόγραμμα μιας παραγγελίας και να φτάσει εις πέρας, πρέπει να καθοριστεί και μια ημερομηνία αποστολής του προϊόντος. Όταν αυτές οι εργασίες ολοκληρωθούν με επιτυχία μπορεί να ξεκινήσει η διαδικασία τιμολόγησης και να σταλεί το αντίστοιχο τιμολόγιο στον πελάτη.

Μια διαδικασία παραγγελίας με τη χρήση web services σαν αυτή που περιγράφηκε παραπάνω, μπορεί να χειριστεί και πιο εξειδικευμένες εργασίες. Για παράδειγμα θα μπορούσε να παρέχει ευκολίες ανίχνευσης και προσαρμογής οι οποίες θα ανίχνευαν και θα ρύθμιζαν διαδικασίες παραγγελιών λόγω απρόσμενων γεγονότων όπως αλλαγή παραγγελίας ή ακύρωσή της από τον πελάτη. Οι εργασίες αυτές απαιτούν μεγάλο φόρτο εργασίας και την χρήση αντιδραστικών web services. Εάν δηλαδή παραστεί ανάγκη για κάποια αλλαγή ή ακύρωση της διαδικασίας παραγγελίας, η όλη διαδικασία μπορεί θα συνεχίσει να εκτυλίσσεται ομαλά. Η χρησιμοποίηση μιας συλλογής web services που λειτουργούν μαζί για την ρύθμιση τέτοιων καταστάσεων σε μια διαδικασία παραγγελίας δημιουργεί μια αυτοματοποιημένη λύση σε αυτό το πρόβλημα. Σε περίπτωση ακύρωσης μιας εντολής αγοράς, η web service διαδικασία παραγγελίας κρατά ένα κατάλληλο προϊόν αντικατάστασης και ειδοποιεί τις υπηρεσίες χρέωσης και απογραφής αποθεμάτων για τις αλλαγές. Όταν όλες αυτές οι web service αλληλεπιδράσεις ολοκληρωθούν και ο νέος προσαρμοσμένος προγραμματισμός

είναι πλέον διαθέσιμος, η web service εντολή αγοράς ειδοποιεί τον πελάτη αποστέλλοντάς του το ενημερωμένο τιμολόγιο.



Εικόνα 3: Παράδειγμα Web Service – Διαδικασία Παραγγελίας (Papazoglou, 2008)

Η επιλογή της δημιουργίας web services από μια επιχείρηση έχει και το ανάλογο κόστος αφού απευθύνεται σε προγραμματιστές και όχι σε απλούς χειριστές. Η δημιουργία ενός web service από μία επιχείρηση, σημαίνει πως στοχεύει είτε στην πώληση των υπηρεσιών της σε άλλες επιχειρήσεις είτε στη δημοσιοποίηση και διαφήμιση των παρεχόμενων υπηρεσιών της. Είναι λοιπόν στην ευχέρεια κάθε επιχειρηματία να αποφασίσει αν πρέπει να επενδύσει χρόνο και χρήμα σε δημιουργία υπηρεσιών που θα παρέχονται μέσω Internet είτε δωρεάν είτε επί πληρωμή.

Με την αρχιτεκτονική των υπηρεσιών ιστού αντιμετωπίστηκαν τα περισσότερα προβλήματα συμβατότητας και διαλειτουργικότητας μεταξύ των δικτυακών λογισμικών συστημάτων. Παρόλα αυτά υπάρχουν ακόμα αρκετά ανοικτά θέματα που έχουν να κάνουν με:

- την ασφάλεια,
- την ποιότητα επικοινωνίας και
- με τον αυτόματο τρόπο αναζήτησης και ενσωμάτωσης έτοιμων υπηρεσιών σε ένα λογισμικό σύστημα.

Η τεχνολογία των web services εξελίσσεται συνεχώς προτείνοντας και θεσπίζοντας νέα standards που αφορούν τον τρόπο ανάπτυξης και λειτουργίας των υπηρεσιών δίνοντας ιδιαίτερη έμφαση στο «semantic web» δηλαδή στη σημασιολογική ερμηνεία όλων των κόμβων πληροφορίας του παγκόσμιου ιστού (Θεμιστοκλεους, 2009a, Papazoglou, 2008)

2.1.1 Πλεονεκτήματα & Μειονεκτήματα Web Services

Τα πλεονεκτήματα από τη χρήση των υπηρεσιών ιστού είναι πολλά, ορισμένα από αυτά είναι:

- *Διαλειτουργικότητα*: Ένα web service δίνει ανεξαρτησία τόσο σε επίπεδο λειτουργικού συστήματος όσο και σε επίπεδο υλικού (hardware). Οποιοδήποτε πρόγραμμα μπορεί εύκολα να προσπελάσει μία υπηρεσία ιστού
- *Ενσωμάτωση*: Ένα web service μπορεί εύκολα να ενσωματωθεί σε οποιοδήποτε λογισμικό που λειτουργεί μέσα στο Internet χωρίς να απαιτούνται αλλαγές στον μηχανισμό του συστήματος
- *Διαθεσιμότητα και δημοσίευση*: Τα web services δημοσιεύονται στο Internet οπότε είναι πολύ εύκολο να αναζητηθούν και ενσωματωθούν σε οποιοδήποτε σύστημα.
- *Επέκταση*: Ένα web service είναι εύκολο να επεκταθεί αν προστεθούν σε αυτό νέες λειτουργίες με σκοπό την παροχή νέων υπηρεσιών προς τους χρήστες.
- *Μικρό κόστος δημιουργίας και χρήσης*: Αφού σε ένα σύστημα υπάρχει ήδη κάποια διαδικασία η οποία θα επεκταθεί σε online υπηρεσία, το κόστος της δημιουργίας της υπηρεσίας είναι ελάχιστο. Επίσης η ενσωμάτωση μιας υπηρεσίας σε κάποια ιστοσελίδα ή δικτυακή εφαρμογή κοστίζει ελάχιστα.
- *Χρήση λογισμικών συστημάτων*: Όλες οι ιστοσελίδες οι οποίες χρησιμοποιούν έτοιμες υπηρεσίες είναι πιο λειτουργικές αφού παρέχουν περισσότερες υπηρεσίες στους χρήστες (Options, 2003)

Γενικά με την χρήση των Web Services επιτυγχάνεται γρήγορη και εύκολη ανάπτυξη πληροφοριακών συστημάτων και αποδοτική ανάπτυξη εφαρμογών. Καίριο πλεονέκτημα των υπηρεσιών ιστού είναι η διασυνδεσιμότητα και η άμεση ολοκλή-

ρωση που προσφέρουν (just in time integration). Παράλληλα περιορίζεται η πολυπλοκότητα (encapsulation) ενώ μειώνεται το κόστος διασύνδεσης της εφαρμογής (interface cost). Τέλος, τα web services παρέχουν έναν γενικής χρήσης μηχανισμό για την υιοθέτηση της επιχειρηματικής διαδικασίας και αναπτύσσονται τόσο στο εσωτερικό της εταιρείας όσο και σε πολλαπλούς οργανισμούς. (Chappell and Jewell, 2002, Papazoglou, 2008)

Πέρα από όλα αυτά τα πλεονεκτήματα και την αποτελεσματικότητα την οποία προσφέρουν, τα web services θα πρέπει να ξεπεράσουν στο μέλλον κάποια μειονεκτήματα τα οποία δεν μπορούν να περάσουν απαρατήρητα. Τα πιο σημαντικά από αυτά είναι:

- Τα πρότυπα των Web services, για λειτουργίες όπως οι συναλλαγές (transactions) είναι προς το παρόν σχεδόν ανύπαρκτα ή τουλάχιστον σε πολύ πρώιμα στάδια αντίθετα με άλλα πολύ πιο ώριμα ανοικτά πρότυπα συστημάτων όπως η CORBA.
- Σε σύγκριση με άλλες κατανεμημένες προσεγγίσεις, όπως RMI ή CORBA τα Web services υστερούν σε απόδοση. Αυτό είναι ένα κοινό μειονέκτημα των text-based διαμορφώσεων, καθώς η XML δεν στοχεύει ούτε στην σαφήνεια της κωδικοποίησης ούτε στην αποτελεσματικότητα της λεκτικής ανάλυσης (parsing).
- Η παρουσία του HTTP στα web services, επιτρέπει σε αυτά την αποφυγή των υπαρχόντων μέτρων ασφαλείας (firewall), σκοπός των οποίων είναι να εμποδίσουν ή να ελέγξουν την επικοινωνία μεταξύ των προγραμμάτων από κάθε πλευρά του firewall.

2.1.2 Χαρακτηριστικά Web Services

Οι υπηρεσίες ιστού, έχουν ειδικά χαρακτηριστικά συμπεριφοράς. Τα κυριότερα από αυτά είναι:

- *eXtensible Markup Language*: Ίσως το κυριότερο χαρακτηριστικό είναι η αξιοποίηση και χρήση της XML, αφού χρησιμοποιείται για την μεταφορά δεδο-

μένων, αποβάλλει κάθε δικτύωση, λειτουργικό σύστημα ή δέσμευση πλατφόρμας που έχει ένα πρωτόκολλο.

- *Loosely Coupled:* Ένας πελάτης μιας υπηρεσίας διαδικτύου δεν είναι δεσμευμένος με μια συγκεκριμένη υπηρεσία άμεσα. Το Interface της υπηρεσίας μπορεί να αλλάξει με την πάροδο του χρόνου, χωρίς να επηρεάσει τη δυνατότητα που έχει ο πελάτης να αλληλεπιδρά με την υπηρεσία. Ένα *tightly coupled* σύστημα δεσμεύει τους πελάτες με τον εξυπηρετητή και σε περίπτωση που αλλάξει μια διεπαφή, θα πρέπει να ενημερώνεται και η άλλη. Με την Loosely Coupled μέθοδο, τα συστήματα γίνονται περισσότερο εύχρηστα.
- *Coarse Grained:* Οι αντικειμενοστραφείς τεχνολογίες όπως η JAVA, υλοποιούν τις υπηρεσίες τους μέσω ξεχωριστών μεθόδων. Μια μέθοδος είναι μια πάρα πολύ λεπτή λειτουργία που παρέχει κάθε χρήσιμη ικανότητα σε ένα συνεταιρικό επίπεδο. Η ανάπτυξη ενός προγράμματος Java από την αρχή απαιτεί δημιουργία πολλών "fine-grained" μεθόδων οι οποίες προσχωρούν έπειτα σε μια "coarse-grained" υπηρεσία που εκτελείται είτε από έναν πελάτη είτε από μια άλλη υπηρεσία. Οι επιχειρήσεις και οι διεπαφές που υλοποιούν πρέπει να είναι "coarse-grained". Τα web services παρέχουν έναν φυσικό τρόπο ορισμού υπηρεσιών "coarse-grained", που έχουν πρόσβαση στην επιχειρησιακή λογική.
- *Δυνατότητα να είναι σύγχρονες και ασύγχρονες:* Ο συγχρονισμός αναφέρεται στη σύνδεση του πελάτη με την εκτέλεση της υπηρεσίας. Στις σύγχρονες επικλήσεις, ο πελάτης στέλνει ένα μήνυμα αιτήματος και περιμένει ένα μήνυμα ανταπόκρισης πριν συνεχίσει με τους υπολογισμούς του. Στα ασύγχρονα web services, ο πελάτης στέλνει ένα μήνυμα αιτήματος το οποίο περιλαμβάνει όλο το έγγραφο και όχι ένα τμήμα του και μπορεί να συνεχίσει να εκτελεί κάποιες άλλες λειτουργίες. Η υπηρεσία, μπορεί να αποστείλει ή να μην αποστείλει μήνυμα απάντησης.
- *Υποστήριξη διαδικασίας εξ' αποστάσεως κλήσεων (Remote Procedure Calls, RPCs)*: Οι υπηρεσίες ιστού επιτρέπουν στους πελάτες να καλέσουν τις διαδικασίες, τις λειτουργίες, και τις μεθόδους σε απομακρυσμένα αντικείμενα χρησιμοποιώντας ένα "XML-based" πρωτόκολλο. Οι απομακρυσμένες διαδικασίες εκθέτουν τις παραμέτρους εισαγωγής και εξαγωγής που μια υπηρεσία Ιστού πρέπει να υποστηρίξει. Μια υπηρεσία Ιστού υποστηρίζει RPC με την παροχή υπηρεσιών της, ισοδύναμων με εκείνες ενός παραδοσιακού συσ-

τατικού, ή με την μετάφραση των εισερχόμενων κλήσεων σε μια κλήση ενός EJB ή ενός .NET συστατικού.

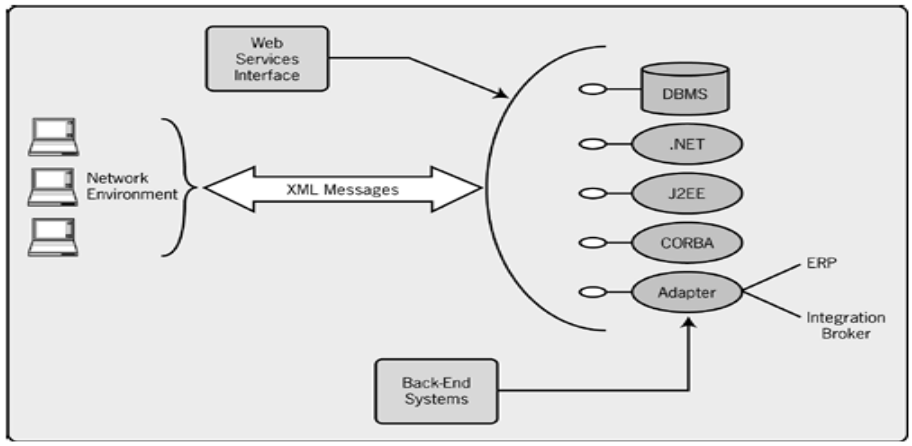
- *Υποστήριξη ανταλλαγής κειμένων:* Ένα από τα βασικά χαρακτηριστικά της XML είναι ο γενικός τρόπος του αναπαριστά όχι μόνο τα στοιχεία, αλλά και σύνθετα έγγραφα. Αυτά τα έγγραφα μπορούν να είναι απλά, όπως όταν αντιπροσωπεύουν μια τρέχουσα διεύθυνση, ή μπορούν να είναι σύνθετα, αντιπροσωπεύοντας ένα ολόκληρο βιβλίο. Οι υπηρεσίες ιστού υποστηρίζουν τη διαφανή ανταλλαγή των εγγράφων για να διευκολύνουν την επιχειρησιακή ενοποίηση. (Chappell and Jewell, 2002)

2.1.3 Τεχνικά Χαρακτηριστικά

Οι υπηρεσίες ιστού αποτελούν εφαρμογές που βασίζονται στη γλώσσα *Extensible Markup Language (XML)* και αποτελούν διεπαφές σε εφαρμογές, αντικείμενα, βάσεις δεδομένων ή επιχειρησιακές διαδικασίες.

Για την εκτέλεση μιας υπηρεσίας ιστού απαιτείται η αποστολή ενός εγγράφου XML σε μορφή μηνύματος μέσω του δικτύου και μετά την εκτέλεση της υπηρεσίας, προαιρετικά, μπορεί να αποσταλεί απάντηση σε μορφή εγγράφου XML. Οι τυποποιήσεις των υπηρεσιών ιστού καθορίζουν :

- Τυποποιήσεις για τη μορφοποίηση των μηνυμάτων
- Τη διεπαφή στην οποία στέλνονται τα μηνύματα
- Αντιστοιχίσεις μεταξύ των δεδομένων των μηνυμάτων και των δεδομένων των εφαρμογών που κρύβονται πίσω από τις υπηρεσίες ιστού
- Μηχανισμούς για τη δημοσίευση και την αναζήτηση υπηρεσιών ιστού.



Εικόνα 4: Web Services Structure (Chappell and Jewell, 2002)

Οι περισσότερες υπηρεσίες εκτελούνται μέσω του διαδικτύου με την συμπλήρωση των απαραίτητων φορμών εισόδου για την υπηρεσία σε μια σελίδα Hyper Text Markup Language (HTML) και με την αποστολή αυτών των δεδομένων στην υπηρεσία ενσωματωμένα σε ένα Uniform Resource Locator (URL). Απλές συναλλαγές στο διαδίκτυο, όπως είναι η αναζήτηση ή η αγορά προϊόντων πραγματοποιούνται μέσω διαδικτύου απλά με την εισαγωγή παραμέτρων και λέξεων κλειδιών σε ένα URL.

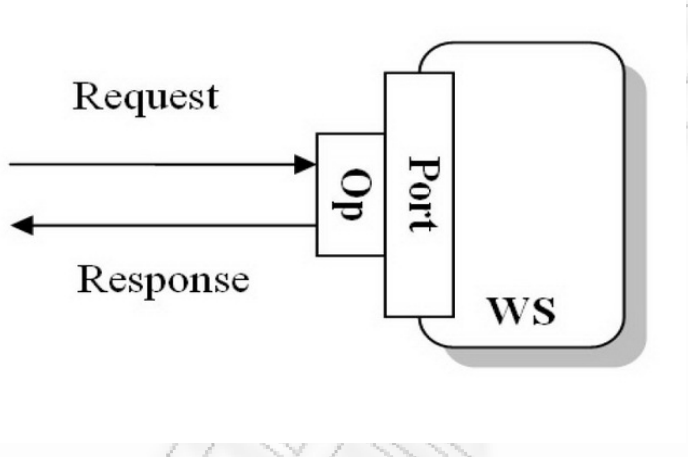
Η αποστολή της αίτησης σε μορφή XML εγγράφου έχει πολλά πλεονεκτήματα, όπως καλύτερη υποστήριξη για τύπους δεδομένων, μεγαλύτερη ευελιξία και επεκτασιμότητα. Η XML μπορεί να αναπαραστήσει διάφορους τύπους δεδομένων και επίσης έχει δυνατότητα μεταφοράς περισσοτέρων δεδομένων από ότι ένα κείμενο σε ένα URL.

Ένα web service είναι μια αυτόνομη λογισμική μονάδα και εκτελεί όπως προαναφέρθηκε μια συγκεκριμένη εργασία. Για την προσπέλαση λειτουργιών μέσω του διαδικτύου με την ανταλλαγή τυποποιημένων μηνυμάτων σε μορφή XML χρησιμοποιείται μια διεπαφή (interface). Για την αλληλεπίδραση με ένα web service χρησιμοποιείται η περιγραφή της υπηρεσίας (service description) σε μορφή XML, που παρέχει όλες τις απαραίτητες λεπτομέρειες για την αλληλεπίδραση με την υπηρεσία, συμπεριλαμβανομένου των μηνυμάτων που ανταλλάσσονται, τα πρωτόκολλα επικοινωνίας και την τοποθεσία της υπηρεσίας. Οι περιγραφές των υπηρεσιών εκφράζονται στη γλώσσα WSDL *(Web Service Description Language)* (Πουλημενοπουλου, 2009)

2.1.4 Τοπολογίες Web Services

Τα Web Services χωρίζονται σε δύο κύριες κατηγορίες. Τις *Απλές* ή *Πληροφοριακές* ή *τύπου I* και τις *Περίπλοκες* ή *τύπου II.* Ας δούμε κάποιες πληροφορίες για καθέναν από αυτούς τους δυο τύπους υπηρεσιών ιστού.

- *Απλές ή Πληροφοριακές, ή τύπου I:* Είναι τα Web Services τα οποία υποστηρίζουν μόνο απλές λειτουργίες (αίτημα/απόκριση) και περιμένουν πάντα για το αίτημα. Έπειτα το επεξεργάζονται και αποκρίνονται.



Εικόνα 5: Web Services τύπου I (Papazoglou, 2008)

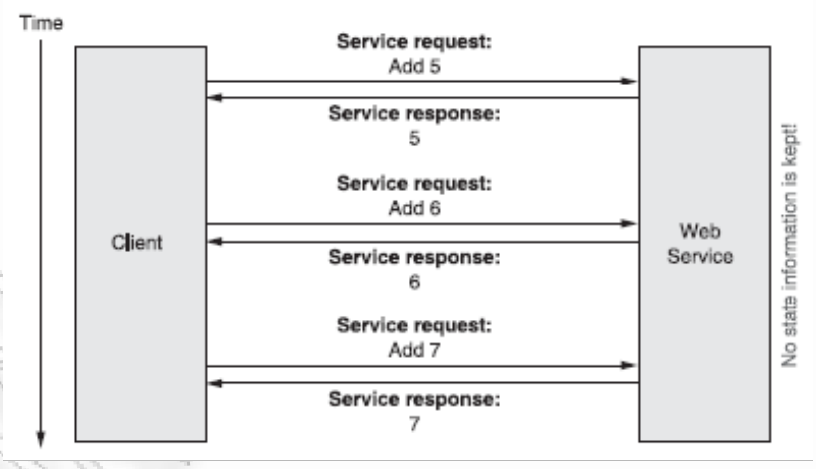
Τα Web Services τύπου I, χωρίζονται σε τρεις υποκατηγορίες οι οποίες είναι οι εξής :

- *Υπηρεσίες καθαρού περιεχομένου:* Οι υπηρεσίες αυτές, δίνουν προγραμματιστική πρόσβαση σε περιεχόμενο όπως ειδήσεις, πληροφορίες για τον καιρό κλπ.
- *Υπηρεσίες εμπορίου:* Σε αυτήν την κατηγορία, εντάσσονται οι πιο σύνθετες υπηρεσίες, οι οποίες απαιτούν συλλογή πληροφοριών από διάφορα πληροφοριακά συστήματα (π.χ. Logistics)
- *Υπηρεσίες σύνδεσης πληροφοριών:* Είναι οι υπηρεσίες προστιθέμενης αξίας, οι οποίες επιτρέπουν την πρόσβαση σε εμπορικά sites (π.χ. σύστημα κράτησης θέσεων). Σε γενικές γραμμές οι υπηρεσίες αυτές παρέχονται από τρίτους και τρέχουν όλο το φάσμα από εμπορικές υπηρεσίες, όπως οι μεταφορές και εφοδιασμοί, η εκπλήρωση πληρωμών και οι υπηρεσίες παρακολούθησης μέχρι και εμπορικές υπηρεσίες προστιθέμενης αξίας, όπως οι υπηρεσίες βαθμολόγησης. Κλασσι-

κά παραδείγματα τέτοιων υπηρεσιών μπορεί να είναι υπηρεσίες κρατήσεων σε ταξιδιωτικά site ή υπηρεσίες που παρέχει κάποια site ασφαλιστικών εταιρειών .

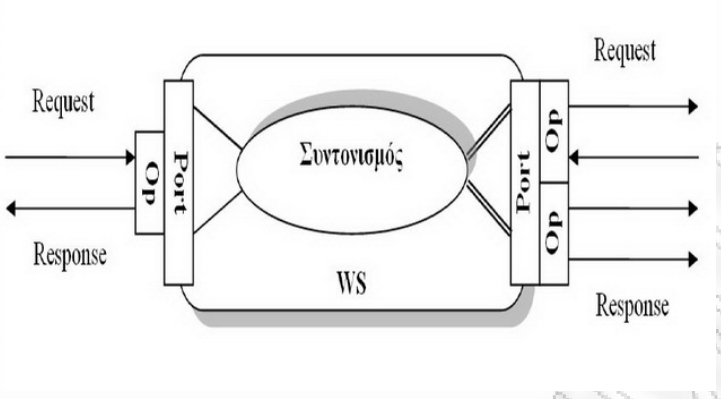
Οι υπηρεσίες αυτού του τύπου, είναι σχετικά απλές στη φύση τους και ονομάζονται *stateless web services.* Ο όρος stateless σημαίνει ότι το Web Service δεν έχει τη δυνατότητα να κρατά στην μνήμη πληροφορίες της ροής δραστηριοτήτων μεταξύ των διαδοχικών αιτημάτων (από την μια κλήση στην άλλη).

Για να γίνει καλύτερα κατανοητός ο όρος stateless web services, ας δούμε ένα πολύ απλό web service η λειτουργία του οποίου είναι η άθροιση ακεραίων. Η αρχική τιμή του αθροιστή είναι ίση με μηδέν, και εν συνεχεία αθροίζονται κάποιοι αριθμοί με αυτόν. Στην πρώτη κλήση προστίθεται ο αριθμός 5, και η απάντηση στον client είναι ο αριθμός 5. Στην δεύτερη κλήση, στον αριθμό 5 προστίθεται ο αριθμός 6. Θα περίμενε κανείς πως η απάντηση του server στον client θα ήταν ο αριθμός 11, λόγω όμως του ότι πρόκειται για stateless web service, ο server δεν έχει την δυνατότητα να γνωρίζει το αποτέλεσμα της προηγούμενης κλήσης και έτσι επιστρέφει στον client τον αριθμό 6. Ομοίως στην τρίτη κλήση, με την πρόσθεση του αριθμού 7, δεν επιστρέφεται ο αριθμός 18, αλλά ο αριθμός 7.



Εικόνα 6: Stateless web service (Μακρης, 2009)

- *Περίπλοκες ή τύπου ΙΙ:* Πρόκειται για web services τα οποία υλοποιούν κάποιας μορφής συντονισμό μεταξύ εισερχομένων και εξερχομένων εργασιών.



Εικόνα 7: Web Services τύπου II (Papazoglou, 2008)

Οι πολύπλοκες ή τύπου II web services συνήθως περιλαμβάνουν ή συνθέτουν πολλές προϋπάρχουσες υπηρεσίες και χωρίζονται σε:

- *Πολύπλοκες Web Services που συνθέτουν προγραμματιστικές υπηρεσίες:* Τις συνθέτουν οι πελάτες τους για να δημιουργήσουν σύνθετες υπηρεσίες (π.χ. η υπηρεσία ελέγχου αποθεμάτων, αποτελεί μέρος της διαδικασίας διαχείρισης αποθεμάτων)
- *Πολύπλοκες Web Services που συνθέτουν διαδραστικές υπηρεσίες:* Οι πελάτες αυτής υποκατηγορίας υπηρεσιών ιστού, ενσωματώνουν διαδραστικές εφαρμογές διαχείρισης μέσα σε web εφαρμογές, συνθέτοντας ολοκληρωμένες υπηρεσίες (χρησιμοποιώντας εξωτερικούς παρόχους υπηρεσίας).

Οι πολύπλοκες υπηρεσίες επιδεικνύουν coarse-grained λειτουργικότητα και είναι *stateful*. Σε αντίθεση με τις stateless υπηρεσίες, οι stateful δίνουν τη δυνατότητα στο web service να κρατά στη μνήμη πληροφορίες της ροής δραστηριοτήτων μεταξύ διαδοχικών κλήσεων. (Θεμιστοκλεους, 2009a, Papazoglou, 2008)

2.2 Service Oriented Architecture (SOA) – Υπηρεσιοστρεφής Αρχιτεκτονική

Η *Service Oriented Architecture (SOA)* είναι ένας λογικός τρόπος για τον σχεδιασμό ενός λογισμικού και παροχής υπηρεσιών στον τελικό χρήστη ή σε άλλες υπηρεσίες μοιρασμένες σε ένα δίκτυο, μέσω διεπαφών έκδοσης και εξερεύνησης. Με την ανάπτυξη υπηρεσιοστρεφούς αρχιτεκτονικής δημιουργούνται είτε εφαρμογές που χρησιμοποιούν υπηρεσίες (services), είτε εφαρμογές που λειτουργούν σαν υπηρεσίες προς όφελος άλλων εφαρμογών, είτε συνδυασμός και των δυο. Αναπτύσσονται δηλαδή εφαρμογές, οι οποίες χρησιμοποιούν υπηρεσίες και αυτές με την σειρά τους αποτελούν υπηρεσίες για άλλες εφαρμογές.

Η SOA παρέχει μια αρχιτεκτονική, η οποία προσφέρει ένα πλαίσιο ολοκλήρωσης πάνω στο οποίο οι αρχιτέκτονες λογισμικού μπορούν να σχεδιάσουν εφαρμογές χρησιμοποιώντας μια συλλογή από επαναχρησιμοποιήσιμα λειτουργικά τμήματα με ευκρινείς διεπαφές. Οι εφαρμογές ολοκληρώνονται στις διεπαφές και όχι στο επίπεδο της υλοποίησης. Ένα ακόμη χαρακτηριστικό της SOA είναι ότι επιτρέπει πολλές –προς– πολλές ολοκληρώσεις. Για παράδειγμα, μια πληθώρα από καταναλωτές, χρησιμοποιούν ξανά και ξανά εφαρμογές με πολλούς διαφορετικούς τρόπους.

Οι οργανισμοί οι οποίοι χρησιμοποιούν SOA διαχωρίζονται σε τρία σημεία εισόδου, με βάση τις επιχειρηματικές τους απαιτήσεις και προτεραιότητες

- Υλοποίηση ενορχήστρωσης εταιρικών υπηρεσιών,
- Υπηρεσίες ενδυνάμωσης όλου του οργανισμού,
- Υλοποίησης από edge-to-edge συνεργασίες επιχειρηματικών διαδικασιών.

Μια υπηρεσία παρέχει μια συγκεκριμένη διαδικασία, όπως για παράδειγμα η επεξεργασία μιας αγοράς. Επίσης μια υπηρεσία μπορεί να αποτελείται μόνο από μια διαδικασία, για παράδειγμα η μετατροπή συναλλάγματος από ένα νόμισμα σε κάποιο άλλο, ή να διαχειρίζεται πολλές διαδικασίες οι οποίες σχετίζονται μεταξύ τους, όπως για παράδειγμα σε ένα σύστημα κράτησης θέσεων.

Το χαρακτηριστικό το οποίο κάνει την αρχιτεκτονική SOA ξεχωριστή σε σχέση με άλλες αρχιτεκτονικές, είναι το γεγονός ότι ο πελάτης (client) μιας υπηρεσίας είναι ανεξάρτητος από την ιδία την υπηρεσία . Ο τρόπος με τον οποίο ένας πελάτης (ή μια άλλη υπηρεσία) καλεί μια υπηρεσία δεν σχετίζεται με την ανάπτυξη της υπη-

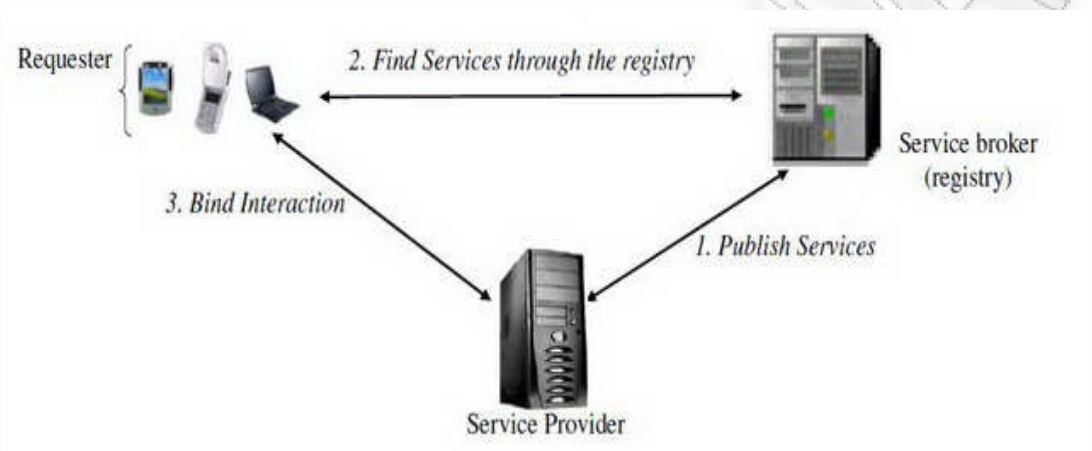
ρεσίας αυτής. Στην πράξη αυτό σημαίνει ότι ο client δεν είναι απαραίτητο να γνωρίζει οτιδήποτε για την υπηρεσία που επιθυμεί να χρησιμοποιήσει. Δεν είναι απαραίτητο να γνωρίζει την γλωσσά προγραμματισμού στην οποία είναι υλοποιημένη είναι η υπηρεσία ή σε τι λειτουργικό σύστημα θα τρέχει. Ο πελάτης επικοινωνεί με την υπηρεσία μέσω ενός interface και από αυτό το σημείο και έπειτα η υπηρεσία εκτελεί τις όποιες λειτουργίες απαιτούνται. Σε περίπτωση κατά την οποία τροποποιηθεί η υπηρεσία, χωρίς να υπάρξει κάποια μετατροπή στο interface , τότε ο πελάτης θα εξακολουθήσει να καλεί την συγκεκριμένη υπηρεσία χωρίς κανένα πρόβλημα. Για παράδειγμα ένα σύστημα κρατήσεων θέσεων κάποια στιγμή αναβαθμιστεί, οι πελάτες θα εξακολουθήσουν να το χρησιμοποιούν με την προϋπόθεση το interface της εφαρμογής να παραμένει ίδιο.

Είναι φανερό πως τα web services έχουν μια μεγάλη ανεξαρτησία η οποία τους παρέχει τη δυνατότητα να είναι *document – centric*. Με τον όρο αυτό μια υπηρεσία μπορεί να δέχεται σαν είσοδο ένα «έγγραφο» αντί για κάποιο αντικείμενο Java. Όπως προαναφέρθηκε παραπάνω, έτσι και τώρα ο πελάτης πάλι δεν ενδιαφέρεται για την διαδικασία επεξεργασίας του εγγράφου. Η υπηρεσία είναι αυτή που έχει τον τρόπο να επεξεργαστεί και να απαντήσει στην αίτηση. Η ιδιότητα αυτή σε συνδυασμό με την ραγδαία εξάπλωση της XML είναι πολύ σημαντική, καθώς η SOA αποκτά πολύ μεγάλα πλεονεκτήματα, ευελιξία και δυναμική, γεγονός που την καθιστά κυρίαρχη για τα επόμενα χρόνια στο χώρο της Πληροφορικής.

Τα κύρια χαρακτηριστικά της SOA είναι βασισμένα σε τρείς ρόλους που αποτελούνται από τις εξής αρχιτεκτονικές μονάδες όπως φαίνονται και στην εικόνα 8:

- *Ο πάροχος υπηρεσίας (service provider)*: Από επιχειρησιακής απόψεως, είναι ο ιδιοκτήτης της υπηρεσίας. Από αρχιτεκτονικής απόψεως, είναι η πλατφόρμα που φιλοξενεί και ελέγχει την είσοδο στην υπηρεσία.
- *Ο πελάτης υπηρεσίας (service requester)*: Από επιχειρησιακής απόψεως, είναι η επιχείρηση που αναζητά την ικανοποίηση συγκεκριμένων λειτουργιών από άλλη επιχείρηση. Από αρχιτεκτονικής απόψεως, είναι η εφαρμογή που αναζητά, καλεί και εκτελεί μια υπηρεσία. Ο ρόλος του πελάτη υπηρεσίας μπορεί να υλοποιηθεί είτε από μια εφαρμογή που εκτελείται από ένα χρήστη μέσω ενός περιηγητή ιστού (web browser) είτε από μία εφαρμογή χωρίς σύστημα διεπαφής για τους χρήστες (user interface), όπως για παράδειγμα μια άλλη υπηρεσία ιστού

- *Μητρώο καταγραφής υπηρεσίας (service registry)*: Αποτελεί ένα ευρετήριο υπηρεσιών ιστού, όπου οι πάροχοι υπηρεσίας δημοσιεύουν τις υπηρεσίες τους. Οι πελάτες υπηρεσίας προσπελάζουν τα μητρώα καταχώρησης υπηρεσιών για την αναζήτηση υπηρεσιών και για την ανάκτηση πληροφορίας σύνδεσης με αυτές. (Chappell and Jewell, 2002, Θεμιστοκλεους, 2009a, Πουλημενοπουλου, 2009, Papazoglou, 2008)



Εικόνα 8: Service Oriented Architecture - Βασική Τεχνολογία – Ρόλοι (Waluyo et al., 2008)

Το μητρώο καταχώρησης υπηρεσιών είναι προαιρετικό στην αρχιτεκτονική των υπηρεσιών web, λόγω του ότι ένας πάροχος υπηρεσίας μπορεί να στείλει την περιγραφή μιας υπηρεσίας απευθείας στον πελάτη της υπηρεσίας. Επίσης, οι πελάτες υπηρεσίας μπορούν να ανακτήσουν περιγραφές υπηρεσιών από άλλες πηγές, όπως είναι ένα τοπικό αρχείο, ένας τόπος FTP (FTP site) και ένας δικτυακός τόπος (web site).

Τα αντικείμενα είναι τα οποία συμμετέχουν στην αρχιτεκτονική των Web Services είναι :

- Η υπηρεσία (service) και
- Η περιγραφή της υπηρεσίας (service description).

Οι λειτουργίες που εκτελούνται από τους παίκτες σε αυτά τα αντικείμενα είναι

- Δημοσίευση (publish),
- Εύρεση (find) και
- Σύνδεση (bind).

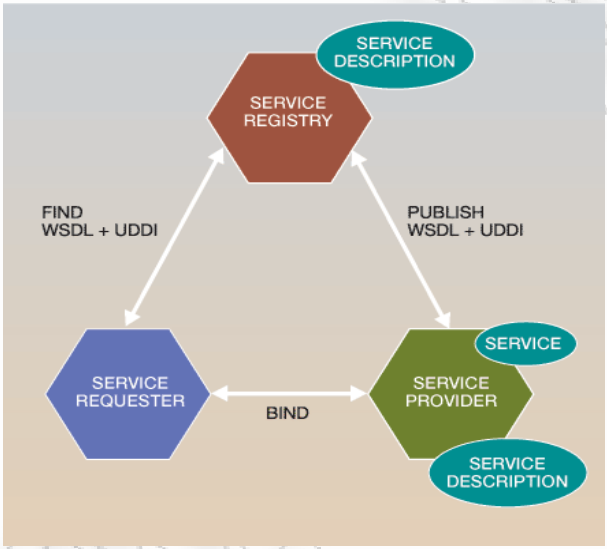
Για τη δημιουργία και εκτέλεση μιας υπηρεσίας απαιτείται η καταγραφή και ο σχεδιασμός μιας ροής ενεργειών. Η ροή αυτή περιγράφει τις δραστηριότητες που πρέπει να εκτελεστούν βήμα – βήμα ώστε να εκτελεστεί η υπηρεσία.

Βήμα 1: Ο πάροχος μιας υπηρεσίας δημιουργεί μια υπηρεσία ιστού και την περιγραφή της και κατόπιν δημοσιεύει την υπηρεσία σε μια αποθήκη υπηρεσιών με βάση την τυποποίηση Universal Description Discovery and Integration *(UDDI)*

Βήμα 2: Με τη δημοσίευση μιας υπηρεσίας ιστού, οποιοσδήποτε αναζητά υπηρεσίες μπορεί να βρει την υπηρεσία μέσω της διεπαφής UDDI.

Βήμα 3: Η αποθήκη με βάση την προδιαγραφή UDDI παρέχει στους πελάτες υπηρεσιών την WSDL περιγραφή της υπηρεσίας και τη διεύθυνση URL της υπηρεσίας.

Βήμα 4: Οι πελάτες υπηρεσίας μπορούν έπειτα να χρησιμοποιήσουν αυτήν την πληροφορία για να συνδεθούν κατευθείαν με την υπηρεσία και να την εκτελέσουν.



Εικόνα 9: Ροή ενεργειών για τη δημιουργία και εκτέλεση μιας υπηρεσίας (Papazoglou, 2008)

Η λειτουργία της έκδοσης, συνίσταται από δύο ισάξιες λειτουργίες:

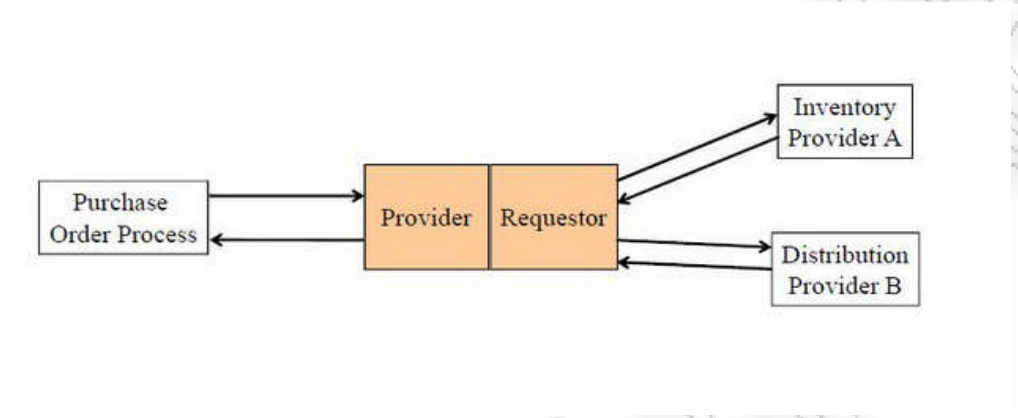
- Την περιγραφή της υπηρεσίας, όπου περιγράφονται πληροφορίες επιχειρηματικής φύσης και τεχνικές πληροφορίες
- Την εγγραφή της υπηρεσίας, όπου εγγράφεται στο μητρώο των τριών χαρακτηριστικών.

Η λειτουργία της εύρεσης, συνίσταται επίσης από δύο λειτουργίες:

- Την εύρεση των υπηρεσιών από τον πράκτορα εξεύρεσης και,
- Τον εντοπισμό της επιθυμητής από τα αποτελέσματα της έρευνας.

Τέλος, η λειτουργία της σύνδεσης, αποτελείται και αυτή από δύο πιθανές λειτουργίες:

- Την απευθείας σύνδεση, και
- Την σύνδεση μέσω διαμεσολαβητή. (Πουλημενοπουλου, 2009)



Εικόνα 10: Παράδειγμα SOA - Composite Service (Papazoglou, 2008)

2.2.1 Επίπεδα Λειτουργίας SOA

Στην εικόνα 11, απεικονίζεται η πολλών επιπέδων προσέγγιση στην ανάπτυξη SOA. Κάθε ένα από τα επίπεδα αυτά περιγράφει ένα λογικό διαχωρισμό σχέσεων ορίζοντας ένα σύνολο κοινών επιχειρηματικών στοιχείων. Κάθε επίπεδο χρησιμοποιεί την λειτουργικότητα του παρακάτω επιπέδου, προσθέτοντας μια νέα λειτουργικότητα για την επίτευξη του στόχου του. Τα επίπεδα αυτά είναι:

Επίπεδο 1: Όλες οι επιχειρηματικές διαδικασίες έχουν ως στόχο ένα συγκεκριμένο επιχειρηματικό πεδίο.

Επίπεδο 2: Το επίπεδο αυτό δημιουργήθηκε από την υποδιαίρεση ενός επιχειρηματικού πεδίου σε μικρότερα τμήματα, από βασικές επιχειρηματικές διαδικασίες.

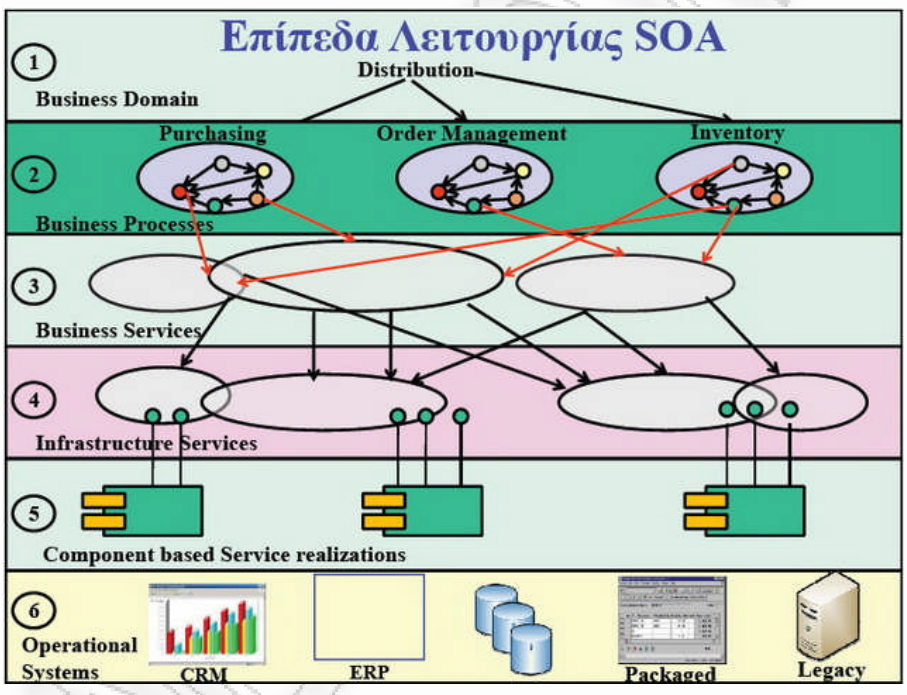
*Επίπεδο 3: Σ*το επίπεδο 3 περιλαμβάνονται υποδιαδικασίες στο χαμηλότερο βαθμό υποδιαίρεσης, με τις υπόλοιπες υπο-διαδικασίες να αποτελούν μοναδικές (invisible-singular) επιχειρηματικές υπηρεσίες προς ανάπτυξη.

*Επίπεδο 4: Σ*το επίπεδο 4, συμμετέχουν οι τεχνικές υπηρεσίες που παρέχουν το τεχνικό υπόβαθρο που επιτρέπει:

- την δημιουργία,
- την παράδοση,
- την συντήρηση και
- τον εφοδιασμό μοναδικών επιχειρηματικών υπηρεσιών.

Επίπεδο 5: Το επίπεδο 5, αποτελεί την πραγματοποίηση του παράγοντα για την υλοποίηση υπηρεσιών από ήδη υπάρχουσες εφαρμογές και συστήματα.

Επίπεδο 6: Το επίπεδο 6, χρησιμοποιείται από παράγοντες για την υλοποίηση επιχειρηματικών υπηρεσιών και διαδικασιών. (Θεμιστοκλεους, 2009a, Papazoglou, 2008)



Εικόνα 11: Επίπεδα λειτουργίας SOA (Papazoglou, 2008)

2.2.2 Πλεονεκτήματα SOA

Σε μια εποχή όπου η ανάπτυξη του ιντερνέτ και των δικτύων είναι ραγδαία η υιοθέτηση της αρχιτεκτονικής SOA είναι αναπόφευκτη. Οι επιχειρήσεις, για την επίτευξη των εταιρικών στόχων τους βασίζονται σε μεγάλο βαθμό στην Πληροφορική Υποδομή τους. Μία πετυχημένη υλοποίηση SOA προσφέρει αξία στην εταιρεία, καθώς καλύπτει τις μεταβαλλόμενες επιχειρηματικές ανάγκες. Σε αυτήν την ενότητα θα

αναλυθούν τα πλεονεκτήματα της αρχιτεκτονικής SOA, τα οποία την καθιστούν ελκυστική και αποτελεσματική. (Μαρκατος, 2005)

2.2.2.1 Επαναχρησιμοποίηση (reusability)

Το πρώτο σημαντικό πλεονέκτημα της αρχιτεκτονικής SOA είναι η επαναχρησιμοποίηση υπηρεσιών. Ο προγραμματιστής σε ένα project έχει την δυνατότητα να χρησιμοποιήσει υπάρχον κώδικα υλοποίησης από άλλες εφαρμογές, να τον χρησιμοποιήσει ως υπηρεσία για την κάλυψη των δικών του αναγκών. Με την δυνατότητα επαναχρησιμοποίησης μειώνεται αισθητά ο προγραμματιστικός φόρτος, και επωφελούμαστε πρακτικά μείωση χρόνου αλλά και χρήματος, κάτι που λαμβάνεται σοβαρά υπόψη από όλες τις μεγάλες εταιρίες ανάπτυξης λογισμικού.

Το κυριότερο μειονέκτημα κατά την επαναχρησιμοποίηση υπάρχοντα κώδικα, είναι οτι κάθε εφαρμογή είναι μοναδική και εξειδικευμένη και εξυπηρετεί διαφορετικές ανάγκες κάθε φορα. Τις περισσότερες φορές οι έτοιμες προγραμματιστικές λύσεις διάφορων projects, έχουν κάποια πολύ ξεχωριστά χαρακτηριστικά. «Τρέχουν» σε διαφορετικά λειτουργικά συστήματα, αναπτύχθηκαν σε διαφορετικές γλώσσες προγραμματισμού και χρησιμοποιούν διαφορετικά interfaces και πρωτόκολλα. Για να μπορέσει κάποιος να χρησιμοποιήσει τις εφαρμογές αυτές, θα πρέπει να αντιληφθεί πλήρως που βρίσκονται οι εφαρμογές αυτές. Το πρόβλημα για τις εταιρίες πληροφορικής προκύπτει όταν έχουν να ολοκληρώσουν μια λύση είτε ενδοεπιχειρησιακά είτε διεπιχειρησιακά.

Όμως στην SOA, το μονό που πρέπει να γνωρίζει μια υπηρεσία για να επικοινωνήσει με κάποια ήδη υπάρχουσα, είναι η δημοσιά διεπαφή της. Έτσι η ολοκλήρωση εφαρμογών και συστημάτων απλοποιείται σε μεγάλο βαθμό.

2.2.2.2 Διαλειτουργικότητα (interoperability)

Εξαιτίας του ότι ο πελάτης και ο πάροχος, δεν ενδιαφέρονται για το λειτουργικό σύστημα, η SOA παρέχει μεγάλη διαλειτουργικότητα. Αυτό επιτυγχάνεται επικοινωνώντας ο πελάτης και ο πάροχος με προτυποποιημένο τρόπο, ο οποίος υλοποιείται κάθε φορά με τα εργαλεία της εκάστοτε πλατφόρμας ή γλωσσάς προγραμματισμού.

Αυτό εφαρμόζεται και στα web services. Οι υπηρεσίες ιστού περιλαμβάνουν μια συλλογή πρωτοκόλλων και τεχνολογιών ευρύτατα διαδομένων και γενικώς αποδεκτών, κάτι που τα καθιστά ανεξάρτητα πλατφόρμας, συστήματος και γλωσσάς προγραμματισμού. Με την δημιουργία του WS-I basic profile, που παρουσιάστηκε από τον οργανισμό για τη διαλειτουργικότητα των web services (web services interoperability organization) η διαλειτουργικότητα ενισχύεται ακόμη περισσότερο. Το WS-I είναι υπεύθυνο για την αναγνώριση των τεχνολογιών οι οποίες όταν εφαρμοστούν παρέχουν την δυνατότητα να δημιουργηθούν υπηρεσίες ανεξάρτητες πλατφόρμας, συστήματος ή γλωσσάς προγραμματισμού. Ήδη το WS-I basic profile τυγχάνει αναγνώρισης στον κλάδο των υπολογιστών λόγω των εγκύρων αποτελεσμάτων του.

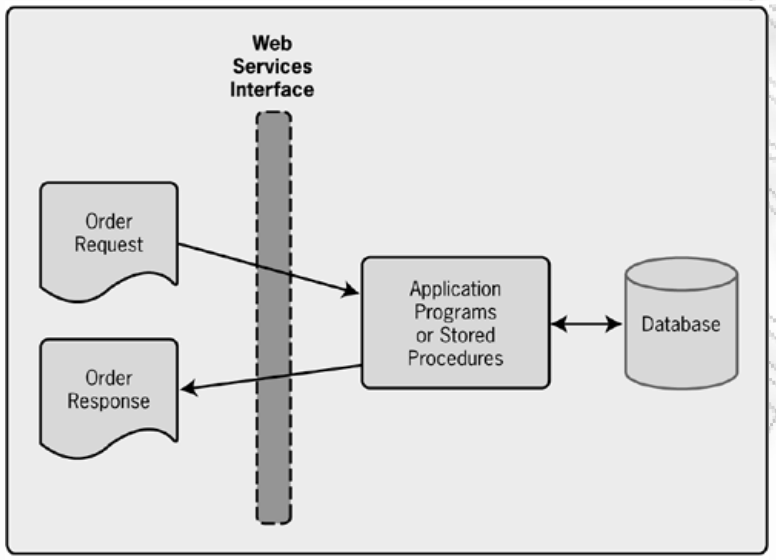
2.2.2.3 Κλιμάκωση (scalability)

Οι υπηρεσίες που βασίζονται στην SOA έχουν μικρή συνεκτικότητα μεταξύ τους και έτσι η μεταξύ τους κλιμάκωση είναι ευκολότερη σε σχέση με υπηρεσίες που βρίσκονται σε πιο συνεκτικά περιβάλλοντα. Αυτό γίνεται γιατί υπάρχουν ορισμένες εξαρτήσεις ανάμεσα στην εφαρμογή που καλεί το web service και στο ίδιο το service. Οι εξαρτήσεις αυτές στην περίπτωση μιας εφαρμογής που λειτουργεί σε σφιχτά συνδεδεμένο περιβάλλον είναι πολύπλοκες και η οποιαδήποτε επέκτασης της εφαρμογής απαιτεί ευρύ προγραμματιστικό φόρτο καθώς οι αλλαγές είναι πολλές.

Είδαμε προηγουμένως πως τα web services είναι coarse grained, δηλαδή μπορούν να προσφέρουν πολλές σχετιζόμενες μεταξύ τους υπηρεσίες. Αυτό δίνει πλεονέκτημα σε σχέση με τα fine grained. Για παράδειγμα μια υπηρεσία αρχιτεκτονικής SOA μπορεί να χειρίζεται όλη τη διαδικασίας μιας online αγοράς ενώ μια άλλη μονό ένα μέρος της συναλλαγής.

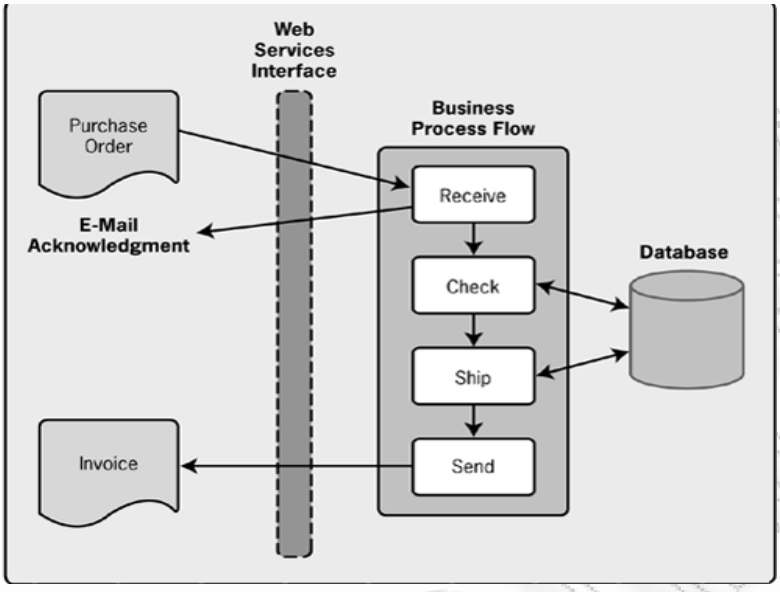
Μια άλλη ιδιότητα των υπηρεσιών διαδικτύου με SOA, είναι ότι υποστηρίζουν δύο τύπους συναλλαγής. Την Απομακρυσμένη κλήση διαδικασιών – *Remote Procedure Calls (online)* και την Εγγραφοκεντρική συναλλαγή – *Document Oriented Interaction.* Στην πρώτη περίπτωση, η συναλλαγή με απομακρυσμένη κλήση διαδικασιών στέλνει ένα έγγραφο σε μία συγκεκριμένη εφαρμογή ή βάση δεδομένων. Εάν όλα εκτελεστούν επιτυχώς η εφαρμογή επιστρέφει ένα XML έγγραφο σε ένδειξη της

εκτέλεσης της λειτουργίας. Τα μηνύματα αίτησης και απάντησης σε αυτήν την περίπτωση είναι σύγχρονα. Αυτό φαίνεται στο παρακάτω διάγραμμα.



Εικόνα 12: Remote Procedure Call (Papazoglou, 2008)

Σύμφωνα με την εγγραφοκεντρική συναλλαγή, η αίτηση για μια υπηρεσία ιστού είναι ένα πλήρες έγγραφο XML που πρόκειται να επεξεργαστεί. Μετά την παραλαβή της αίτησης συνήθως θα αποσταλεί ένα email ή κάποιο άλλο είδος ειδοποίησης στον αποστολέα της αίτησης για να δείξει ότι η αίτηση παρελήφθη και το αίτημα θα επεξεργαστεί. Μετά την ολοκλήρωση της επεξεργασίας μπορεί να αποσταλεί ένα μήνυμα απάντησης όπως φαίνεται στην εικόνα 13. Τα μηνύματα αίτησης και απάντησης σε αυτήν την περίπτωση είναι ασύγχρονα. (Πουλημενοπουλου, 2009)



Εικόνα 13: Document-Oriented Interactions (Papazoglou, 2008)

Δεν είναι δηλαδή αναγκαίο, να περιμένει η υπηρεσία την απόκριση του πελάτη για να συνεχίσει. Το ίδιο ισχύει και για τον πελάτη καθώς είναι δυνατόν να στείλει μια αίτηση και να μην κολλήσει περιμένοντας να αποκριθεί η υπηρεσία. Για την εξυπηρέτηση και άλλων αιτήσεων η διαδικασία προχωρά στο χρόνο που μεσολαβεί.

Συνοψίζοντας τα παραπάνω χαρακτηριστικά παρατηρείται ότι η αλληλεπίδραση ανάμεσα σε εφαρμογή και πελάτη είναι σχετικά περιορισμένη. Έτσι γίνεται ευκολότερη η επέκταση της εφαρμογής, αφού το φορτίο κίνησης που επιβαρύνει το δίκτυο τελικά είναι περιορισμένο. (Πουλημενοπουλου, 2009, Μαρκατος, 2005)

2.2.2.4 Ευελιξία (flexibility)

Οι χαλαρά συνδεδεμένες υπηρεσίες είναι πιο ευέλικτες από τις σφιχτές εφαρμογές. Στις πολύ σφιχτές αρχιτεκτονικές, η σύνδεση των διάφορων τμημάτων είναι πολύπλοκη και έτσι η παραμετροποίηση και επέκταση γίνεται δυσκολότερη και χρονοβόρα με αποτέλεσμα οι εφαρμογές να μην είναι ευέλικτες σε περίπτωση που χρειαστεί κάποια αλλαγή στις προδιαγραφές.

Αντίθετα, εφαρμογές οι οποίες υιοθετούν την αρχιτεκτονική SOA, είναι δυνατόν να παραμετροποιηθούν, να τροποποιηθούν κάποια μικρά τμήματα κώδικα σε συγκεκριμένα σημεία τα όποια είναι εύκολο να διαγνωσθούν και όλα αυτά με ελάχιστο προγραμματιστικό κόστος. (Μαρκατος, 2005)

2.2.2.5 Κόστος (Cost efficiency)

Οι προσπάθειες για την ολοκλήρωση συστημάτων επιχειρήσεων είναι εξαιρετικά δαπανηρές καθώς απαιτούν ακριβές αναλύσεις, αρκετό προγραμματιστικό φόρτο και χρόνο και γενικότερα μεγάλη προσπάθεια. Επίσης, επειδή συνήθως είναι άμεσα συνδεμένες οι εφαρμογές μεταξύ τους η αλλαγή ενός τμήματος προκαλεί και την αλλαγή σε αλλά τμήματα που επικοινωνούν με αυτό.

Κατά την υλοποίηση των εφαρμογών με SOA, όλα τα παραπάνω δεν ισχύουν λόγω της προτυποποίησης. Η οποιαδήποτε αλλαγή πραγματοποιείται σύμφωνα με συγκεκριμένους κανόνες και όχι με προσαρμοσμένες τεχνικές και εκτεταμένο προγραμματισμό. Το πιο σημαντικό όμως χαρακτηριστικό, που μειώνει δραστικά το κόστος αυτής της αρχιτεκτονικής είναι η επαναχρησιμοποίηση του υπάρχοντα κώδικα. (Μαρκατος, 2005)

2.2.3 Μειονεκτήματα SOA

Πέρα όμως από τα πλεονεκτήματα και τα οφέλη που προκύπτουν από την εφαρμογή της αρχιτεκτονικής SOA, φαίνεται πως τελευταία γνωρίζει την παρακμή της. Η SOA και συνεπώς και τα web services έχουν υποστεί τεράστια εως ανεπανόρθωτη ζημία λόγω της πρόσφατης οικονομικής ύφεσης και επιβιώνουν χάρη στους «απογόνους» τους όπως mashups, SaaS, Cloud Computing και άλλες αρχιτεκτονικές προσεγγίσεις οι οποίες εξαρτώνται απο τα services.

Με την εμφάνιση της, η SOA θεωρήθηκε από πολλούς πως είναι ο νέος στυλοβάτης της πληροφορικής, όμως αντίθετα μετατράπηκε σε ένα μεγάλο αποτυχημένο πείραμα τουλάχιστον για τους περισσότερους οργανισμούς. Με τη χρήση της αρχιτεκτονικής SOA αναμενόταν τεράστια μείωση του κόστους και αύξηση της λειτουργικότητας, απέτυχε όμως να εκπληρώσει για πολλούς τα πολλά υποσχόμενα οφέλη της. Σε πολλούς οργανισμούς, τα πράγματα έγιναν χειρότερα: το κόστος είναι υψηλότερο, τα έργα απαιτούν περισσότερο χρόνο, και τα συστήματα είναι πιο ευάλωτη από ποτέ και με τους περιορισμένους σε μεγάλο βαθμό προϋπολογισμούς για το 2009 οι περισσότεροι οργανισμοί περιέκοψαν τις χρηματοδοτήσεις για SOA πρωτοβουλίες.

Εάν καταρρεύσει ολοκληρωτικά η SOA, θα είναι κάτι τραγικό για την βιομηχανία. Οι οργανισμοί χρειάζονται απεγνωσμένα αρχιτεκτονικές βελτιώσεις στα χαρ-

τοφυλάκιά τους. Ο υπηρεσιοστρεφής προσανατολισμός αποτελεί προϋπόθεση για την ταχεία ολοκλήρωση των επιχειρηματικών διαδικασιών και ενεργοποιεί μοντέλα ανάπτυξης όπως τα mashups, ενώ είναι επίσης η θεμελιώδης αρχιτεκτονική SaaS και Cloud Computing.

Αν σήμερα η λέξη SOA είναι συνώνυμη της αποτυχίας, η απαίτηση για service-oriented αρχιτεκτονική είναι ισχυρότερη από ποτέ. Οι άνθρωποι ξεχάσανε τι σημαίνει SOA εξαιτίας της εμπλοκής του σε ανούσιες συζητήσεις περί τεχνολογίας χάνοντας έτσι τα σημαντικότερα πράγματα που προσέφερε: την αρχιτεκτονική και τις υπηρεσίες.

Για να έχουμε και πάλι επιτυχημένη SOA, απαιτείται επανασχεδιασμός και μια μαζική μετατόπιση στον τρόπο λειτουργίας του. Όσοι οργανισμοί γνώρισαν θεαματικά κέρδη από την SOA, πρέπει να βοηθήσουν και στην αναγέννηση και τον μετασχηματισμό του. Ακόμα και οι νέες τεχνολογίες δεν θα κάνουν τα πράγματα καλύτερα αν δεν τηρηθεί κάποια δέσμευση για την αλλαγή. Είναι επιτακτική η ανάγκη για τους οργανισμούς η σημαντική μείωση του κόστους και η αύξηση της ευελιξίας, και δεν θα εκπληρωθεί με στοιχειώδης κινήσεις και επιφανειακές βελτιώσεις.

2.2.4 Προγραμματιστικό Μοντέλο Web Services

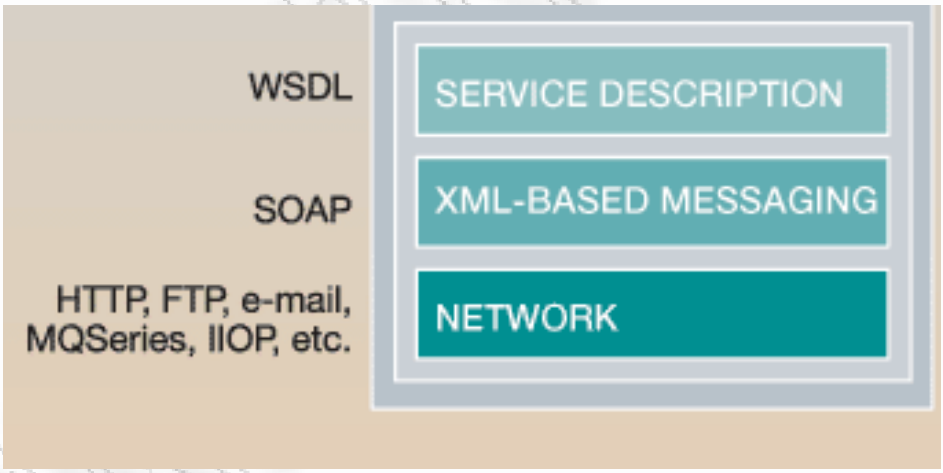
Το προγραμματιστικό μοντέλο των web services αποτελείται από μια συλλογή από τυποποιημένων πρωτόκολλων και διεπαφών εφαρμογών (application programming interfaces - APIs), οι οποίες χρησιμοποιούνται από ανθρώπους και εφαρμογές για τον εντοπισμό και τη χρήση των υπηρεσιών ιστού. Βασική αρχή της δημιουργίας των υπηρεσιών ιστού είναι η τυποποίηση (standardization) απλών και ανοικτών (open) πρωτοκόλλων και διεπαφών, ώστε να είναι εφικτή η ευρεία διάδοση τους μέσω του διαδικτύου. Τα επίπεδα του προγραμματιστικού μοντέλου των υπηρεσιών ιστού είναι:

1^{ο} Επίπεδο: Το επίπεδο δικτύου (network) είναι το δομικό επίπεδο του προγραμματιστικού μοντέλου των υπηρεσιών ιστού. Όλες οι υπηρεσίες ιστού πρέπει να είναι διαθέσιμες μέσω του δικτύου. Το επίπεδο δίκτυο βασίζεται συνήθως στο πρωτόκολλο Hyper Text Transport Protocol (HTTP protocol), αλλά μπορούν να χρησιμο-

ποιηθούν και άλλα πρωτόκολλα όπως είναι το Internet Inter-ORB Protocol (IIOP) ή το SMTP.

2ο Επίπεδο: Πάνω από το επίπεδο δικτύου είναι το επίπεδο XML μηνυμάτων που παρέχει την επικοινωνία μεταξύ των υπηρεσιών και των πελατών που τα καλούν. Αυτό το επίπεδο επικοινωνίας βασίζεται στο πρωτόκολλο Simple Object Access Protocol (SOAP). Το πρωτόκολλο SOAP είναι ένα XML πρωτόκολλο που παρέχει τις λειτουργίες δημοσίευσης (publish), εύρεσης (find) και σύνδεσης (bind) που περιγράφηκαν στην παράγραφο 2.2.

3ο Επίπεδο: Πάνω από το επίπεδο XML μηνυμάτων είναι το επίπεδο περιγραφής υπηρεσιών. Αυτό το επίπεδο βασίζεται στην προδιαγραφή Web Services Description Language (WSDL), που περιγράφει σε μορφή XML εγγράφων διαθέσιμες υπηρεσίες ιστού για τους πελάτες. Αυτά τα 3 επίπεδα είναι υποχρεωτικά για τη δημιουργία διαλειτουργικών (interoperable) υπηρεσιών ιστού και τη δημοσίευση τους στο Internet. Τα ακόλουθα επίπεδα είναι προαιρετικά και χρησιμοποιούνται όταν απαιτείται ανάλογα με τις επιχειρησιακές ανάγκες.



Εικόνα 14: Απαραίτητα επίπεδα προγραμματιστικού μοντέλου Web Services (Papazoglou, 2008)

4ο Επίπεδο: Στο επίπεδο δημοσίευσης ενός web service περιλαμβάνεται η ενέργεια που εκτελεί ο προμηθευτής μιας υπηρεσίας για να κάνει το έγγραφο WSDL διαθέσιμο στους πιθανούς αιτούντες της υπηρεσίας. Έτσι, π.χ. η αποστολή του εγγράφου WSDL (ή της διεύθυνσης URL του εγγράφου) με ένα e-mail σε κάποιον θεωρείται δημοσίευση της υπηρεσίας. Επίσης, η δημοσίευση του εγγράφου WSDL μιας

υπηρεσίας σε μια αποθήκη UDDI, ώστε να είναι διαθέσιμο στους αιτούντες υπηρεσίας, αποτελεί τη δημοσίευση της υπηρεσίας.

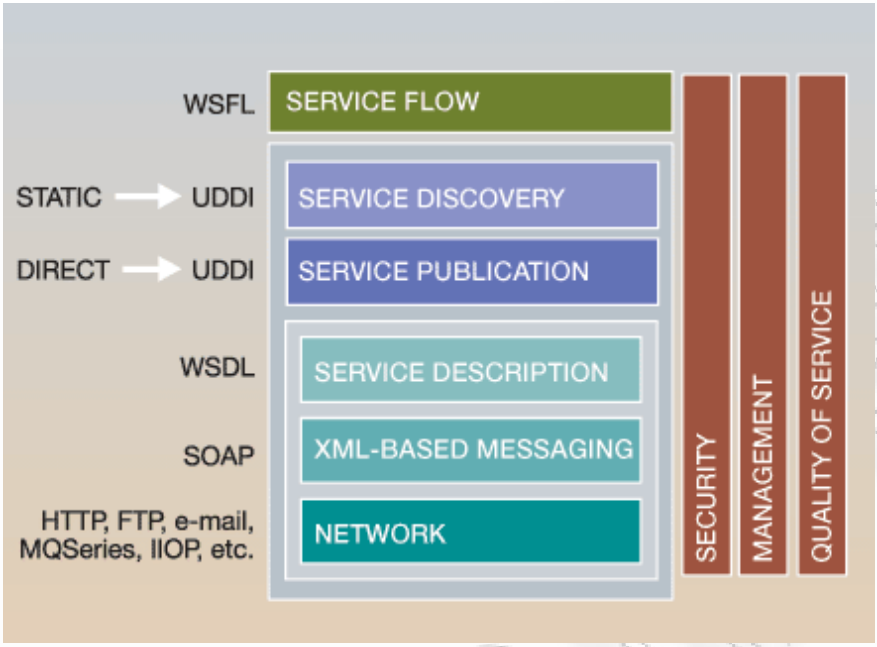
5ο Επίπεδο: Ομοίως, στο επίπεδο ανακάλυψης μιας υπηρεσίας ιστού περιλαμβάνεται κάθε ενέργεια που εκτελείται ώστε ο αιτών υπηρεσίας να προσπελάσει το WSDL έγγραφο της υπηρεσίας. Η ενέργεια μπορεί να είναι απλή, όπως η προσπέλαση του εγγράφου WSDL ή η επίσκεψη της διεύθυνσης που βρίσκεται το έγγραφο WSDL ή πολύπλοκη, όπως είναι η αναζήτηση σε μια αποθήκη UDDI και η χρήση των εγγράφων WSDL για την επιλογή ενός ή περισσοτέρων υπηρεσιών.

6ο Επίπεδο: Το επίπεδο ροής υπηρεσιών (services flow layer) παρέχει τη σύνθεση των υπηρεσιών ιστού σε ροές εργασίας (workflows) και την αναπαράσταση αυτής της συσσώρευσης των υπηρεσιών ως μια υπηρεσία ιστού υψηλότερου επιπέδου. Για αυτό το επίπεδο η IBM έχει προτείνει την τυποποίηση Web Services Flow Language (WSFL) και η Microsoft την τυποποίηση XLANG. Η προδιαγραφή Business Process Execution Language for Web Services (BPEL4WS) αποτελεί μια προσπάθεια για ενοποίηση των προδιαγραφών WSFL και XLANG, συνδυάζοντας τα προτερήματα της κάθε μιας προσέγγισης, με στόχο την τυποποίηση (standard) της για τη σύνθεση υπηρεσιών web.

Σε όλα τα επίπεδα του προγραμματιστικού μοντέλου των web services πρέπει να ληφθούν υπόψη

- η ασφάλεια των υπηρεσιών,
- η διαχείριση των υπηρεσιών και
- η διαχείριση της ποιότητας παροχής υπηρεσιών.

Η λύση που θα επιλεχθεί σε κάθε επίπεδο μπορεί να είναι ανεξάρτητη από τα άλλα επίπεδα. (Πουλημενοπουλου, 2009)



Εικόνα 15: Τα 6 επίπεδα προγραμματιστικού μοντέλου Web Services (Papazoglou, 2008)

2.3 Τεχνολογίες Web Services

Η ραγδαία ανάπτυξη των υπηρεσιών διαδικτύου οφείλεται κατά ένα μεγάλο ποσοστό στις τεχνολογίες που εφαρμόζονται στις υπηρεσίες αυτές και στα πολλά λειτουργικά οφέλη τα οποία παρέχουν. Οι τεχνολογίες που χρησιμοποιούνται στα web services ταξινομούνται σε ένα μοντέλο επιπέδων. Ξεκινώντας από το χαμηλότερο επίπεδο, το οποίο επιτρέπει τη μεταφορά δεδομένων από το ένα μηχάνημα στο άλλο, χτίζονται τα ανώτερα επίπεδα βασισμένα στα κατώτερα, αποκρύπτοντας όμως τις πολλές λεπτομέρειες.

Σήμερα, τέσσερις είναι οι βασικές τεχνολογίες οι οποίες έχουν προκύψει ως παγκόσμια πρότυπα και αποτελούν τον πυρήνα των τεχνολογιών των web services. Οι τεχνολογίες αυτές, είναι (Chappell and Jewell, 2002, Papazoglou, 2008):

- Επεκτάσιμη Γλώσσα Σήμανσης (eXtensible Markup Language, XML)
- Πρωτόκολλο Πρόσβασης Απλού Αντικειμένου (Simple Object Access Protocol, SOAP)
- Γλώσσα Περιγραφής Υπηρεσιών Διαδικτύου (Web Service Description Language, WSDL)
- Παγκόσμια Περιγραφή, Ανακάλυψη και Ολοκλήρωση (Universal Description, Discovery, and Integration UDDI)

2.3.1 Επεκτάσιμη Γλώσσα ΣήμαΣνσης (eXtensible Markup Language, XML)

Όλα τα έγγραφα υπηρεσιών διαδικτύου είναι γραμμένα σε γλώσσα XML και χρησιμοποιείται το XML σχήμα με το οποίο καθορίζονται τα στοιχεία τα οποία χρησιμοποιούνται στην επικοινωνία των υπηρεσιών ιστού. Τα αρχικά XML προκύπτουν από τις λέξεις eXtensible Markup Language. Πρόκειται για μια γλώσσα σήμανσης η οποία αναπτύχθηκε από το W3C (World Wide Web Consortium) με σκοπό να ξεπεραστούν οι περιορισμοί της HTML. Όταν μιλάμε για HTML, γίνεται λόγος για την πιο δημοφιλή ίσως γλώσσα σήμανσης. Υπάρχουν πάνω από 1 δισεκατομμύριο HTML ιστοσελίδες στο διαδίκτυο. Υποστηρίζεται από χιλιάδες εφαρμογές συμπεριλαμβανομένων των πλοηγών διαδικτύου, των επεξεργαστών κειμένου, των ηλεκτρονικών ταχυδρομείων, των βάσεων δεδομένων και άλλων.

Η XML δεν αποτελεί αντικαταστάτη της HTML. Η XML και η HTML έχουν σχεδιαστεί για διαφορετικούς σκοπούς. Η XML έχει σχεδιαστεί για τη μεταφορά και την αποθήκευση δεδομένων, με έμφαση στον τύπο των δεδομένων ενώ η HTML έχει σχεδιαστεί για την εμφάνιση των δεδομένων.

Ανάλυση της XML

Με μια γενική προσέγγιση, η XML θα μπορούσε να χαρακτηριστεί ως ένα σύνολο προτύπων για την ανταλλαγή και δημοσίευση πληροφοριών με ένα δομημένο τρόπο. Είναι σαφής η έμφαση που δίνεται στην δομή και ακριβώς λόγω αυτής της δομής, η XML χρησιμοποιείται στην περιγραφή και τον χειρισμό δομημένων εγγράφων. Τα XML έγγραφα μπορεί να είναι και αντικείμενα σε μια εφαρμογή πελάτη/εξυπηρετητή. Η XML παρέχει μια δενδροειδή δομή σε όλα τα είδη εγγράφων και δεν υποδεικνύει ή επιβάλλει τις λεπτομέρειες αυτής της δομής. Η XML αποτελεί είναι έναν μηχανισμό, ο οποίος παρέχει την δομή συγκεκριμένων εφαρμογών. Αυτή η δομή, παρέχει έναν μηχανισμό κωδικοποίησης των πληροφοριών της εφαρμογής. Τέλος παρέχει μηχανισμούς οι οποίοι είναι υπεύθυνοι για τον χειρισμό των πληροφοριών, την εμφάνισή και την ανάκτησή τους από τρίτους. Ένα απλό παράδειγμα XML δίνεται παρακάτω:

```
<bookstore>
<book category="COOKING">
<title lang="en">Everyday Italian</title>
<author>Giada De Laurentiis</author>
<year>2005</year>
<price>30.00</price>
</book>
<book category="CHILDREN">
<title lang="en">Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>29.99</price>
</book>
</bookstore>
```

Τα χαρακτηριστικά (attributes), συχνά παρέχουν πληροφορία που δεν αποτελεί μέρος των δεδομένων. Για παράδειγμα, ο τύπος του αρχείου είναι άσχετος με τα δεδομένα, αλλά είναι σημαντικός για το λογισμικό που θέλει να επεξεργαστεί το στοιχείο:

```
<file type="gif">computer.gif</file>
```

Τα χαρακτηριστικά πρέπει να είναι σε εισαγωγικά (quotes), είτε μονά είτε διπλά π.χ.

```
<person’s sex="female"> ή
<person’s sex='female'>
```

Well Formed XML Έγγραφα

Πολύ σημαντικό στοιχείο για ένα σωστά δομημένο XML έγγραφο είναι η σωστή σύνταξη. Τα κύρια χαρακτηριστικά ενός καλά σχηματισμένου XML εγγράφου είναι:

- Τα XML έγγραφα πρέπει να έχουν ένα κεντρικό στοιχείο
- Τα XML στοιχεία πρέπει να έχουν ένα tag στην αρχή και ένα στο τέλος
- Τα XML tags είναι case sensitive
- Τα XML στοιχεία πρέπει να εσωκλείονται σωστά
- Οι τιμές των XML χαρακτηριστικών πρέπει να εσωκλείονται σε εισαγωγικά

Ένα XML έγγραφο είναι έγκυρο (valid) όταν είναι καλά σχηματισμένο (well formed) και επίσης υπακούει στους κανόνες του Document Type Definition (DTD) ή XML Schema

XML Schema

Ο σκοπός του XML Schema είναι να καθορίσει τα νόμιμα δομικά στοιχεία ενός XML εγγράφου. Ένα XML Schema καθορίζει :

- Τα στοιχεία που μπορούν να εμφανιστούν σε ένα έγγραφο
- Τα χαρακτηριστικά που μπορούν να εμφανιστούν σε ένα έγγραφο
- Ποια στοιχεία είναι παιδιά άλλων στοιχείων
- Τη σειρά των στοιχείων παιδιών
- Τον αριθμό των στοιχείων παιδιών
- Εάν ένα στοιχείο είναι κενό ή περιλαμβάνει κείμενο
- Τους τύπους δεδομένων των στοιχείων και των χαρακτηριστικών
- Προκαθορισμένες και αρχικές τιμές των στοιχείων και των χαρακτηριστικών

Τα χαρακτηριστικά πλεονεκτήματα που καθιστούν ισχυρά τα XML Schemas έναντι των DTDs είναι:

- Τα XML Schemas είναι επεκτάσιμα
- Τα XML Schemas είναι γραμμένα σε XML
- Τα XML Schemas υποστηρίζουν τύπους δεδομένων
- Τα XML Schemas υποστηρίζουν namespaces.

Τα XML Schemas είναι επεκτάσιμα επειδή ακριβώς είναι γραμμένα σε XML. Με την επεκτασιμότητα των XML Schemas επιτυγχάνεται η επαναχρησιμοποίηση

ενός Schema σε άλλα Schemas. Επίσης είναι εφικτή η δημιουργία ενός νέου τύπου δεδομένων, από τους υπάρχοντες τύπους δεδομένων. Τέλος, είναι δυνατή η αναφορά σε πολλά Schemas στο ίδιο έγγραφο. Ακολουθεί ένα παράδειγμα ενός XML εγγράφου και XML Schema:

XML έγγραφο:

```
<?xml version="1.0"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

XML Schema:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">
<xs:element name="note">
        <xs:complexType>
        <xs:sequence>
                <xs:element name="to" type="xs:string"/>
                <xs:element name="from" type="xs:string"/>
                <xs:element name="heading" type="xs:string"/>
                <xs:element name="body" type="xs:string"/>
        </xs:sequence>
        </xs:complexType>
</xs:element>
</xs:schema>
```

Στον παρακάτω πίνακα, φαίνεται η σύνταξη των στοιχείων και των χαρακτηριστικών

Στοιχείο	Χαρακτηριστικό
<xs:element name="xxx" type="yyy"/>	<xs:attribute name="xxx" type="yyy"/>

Πίνακας 1: Σύνταξη XML στοιχείου

Οι πιο συνήθεις τύποι δεδομένων είναι οι:

- xs: string
- xs: demical
- xs: integer
- xs: boolean
- xs: date
- xs: time

Το στοιχείο <schema> μπορεί να περιέχει κάποια χαρακτηριστικά. Για παράδειγμα:

<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">
</xs:schema>

Ας δούμε τα χαρακτηριστικά αυτά:

- xmlns: xs=http://www.w3.org/2001/XMLSchema. Καθορίζει ότι τα στοιχεία και οι τύποι δεδομένων που χρησιμοποιούνται στο σχήμα προέρχονται από το "http://www.w3.org/2001/XMLSchema" namespace και ότι τα στοιχεία και οι τύποι δεδομένων πρέπει να χρησιμοποιούν το πρόθεμα **xs:**
- Το targetNamespace="http://www.w3schools.com" καθορίζει ότι τα στοιχεία αυτού του σχήματος (note, to, from, heading, body.) προέρχονται από το "http://www.w3schools.com" namespace.
- xmlns="http://www.w3schools.com" καθορίζει αυτό το namespace ως προκαθορισμένο

- elementFormDefault="qualified" καθορίζει ότι κάθε στοιχείο των XML εγγράφων που δημιουργούνται με βάση αυτό το σχήμα πρέπει να υπακούουν στους κανόνες του namespace

XML Namespaces

Τα XML namespaces χρησιμοποιούνται για το διαχωρισμό των ονομάτων των στοιχείων (elements) μέσα σε ένα έγγραφο. Έτσι, δημιουργούν μοναδικά προθέματα για τα στοιχεία που προέρχονται από διαφορετικά έγγραφα ή εφαρμογές αλλά χρησιμοποιούνται μαζί. Επίσης χρησιμοποιούνται ως μοναδικές λέξεις κλειδιά που προσδιορίζουν συγκεκριμένη σημασιολογία που πρέπει να ληφθεί υπόψη κατά την επεξεργασία των εγγράφων όπως φαίνεται και στο παράδειγμα παρακάτω.

<h:html xmlns:xdc="http://www.xml.com/books"
xmlns:h="http://www.w3.org/HTML/1998/html4">
<h:head>
<h:title>Book Review</h:title>
</h:head>
<h:body>
<xdc:bookreview>
<xdc:title>XML: A Primer</xdc:title> </xdc:bookreview>
</h:body>
</h:html>

Τα namespaces έχουν ιδιαίτερη σημασία για τις υπηρεσίες ιστού εξαιτίας της ταυτόχρονης ύπαρξης πολλών σχετιζόμενων XML εγγράφων. Π.χ., μία υπηρεσία ιστού έχει τουλάχιστον 4 σχετιζόμενα XML έγγραφα:

- Το XML έγγραφο που περιέχει τα δεδομένα
- Το SOAP schema του φακέλου SOAP που καθορίζει τη μορφή του μηνύματος
- Το WSDL έγγραφο με το οποίο περιγράφεται η διεπαφή
- Το WSDL schema του εγγράφου που επαληθεύει τη διεπαφή.

Τα Namespaces συνήθως μοντελοποιούνται ως uniform resource locators (URLs), στην γνωστή μορφή του Internet. Για παράδειγμα: *xmlns:myns=http://www.xmlbus.com/namespaces/WSDL.*

Σε αυτό το παράδειγμα το πρόθεμα του namespace που προέρχεται από το URI xmlbus.com είναι το myns. Γενικά όταν χρησιμοποιούνται namespaces ισχύει ότι:

το πλήρες όνομα ενός στοιχείου = namespace URL + όνομα στοιχείου

Παράδειγμα:

<xdc:bookreview> = http://www.xml.com/books +bookreview

Για λόγους συντομίας χρησιμοποιούνται τα προθέματα αντί το namespace URL.

Τα namespaces επίσης χρησιμοποιούνται για να περιγράψουν συγκεκριμένη συμπεριφορά σε έναν επεξεργαστή XML εγγράφων. Για παράδειγμα, όταν χρησιμοποιείται το namespace του SOAP encoding σε ένα μήνυμα SOAP, ο επεξεργαστής του SOAP ξέρει ότι πρέπει να χρησιμοποιήσει το μηχανισμό κωδικοποίησης για το serializing και το deserializing του μηνύματος. Ένα namespace URL, είναι συνήθως ένα μοναδικό αναγνωριστικό για το χαρακτηρισμό των ονομάτων των στοιχείων μέσα σε ένα έγγραφο XML. Μερικές φορές είναι πιθανό στο τέλος του namespace URL να υπάρχει ένα σχήμα XML (XML schema) το οποίο να περιέχει πληροφορία για τα πεδία XML, όπως να βοηθάει στην επικύρωση των στοιχείων ή των τύπων τους.

Τα namespaces συνήθως δηλώνονται στην αρχή ενός εγγράφου, αλλά μπορούν να δηλωθούν και σε επίπεδο ενός στοιχείου. Για παράδειγμα: <sktb:boot xmlns:sktb=http://www.xmlbus.com/skateboots/>

Το boot είναι το στοιχείο XML, xmlns είναι η δομή για τον προσδιορισμό του προθέματος sktb ως ένα namespace για το στοιχείο. Το namespace URL είναι http://www.xmlbus.com/skateboots, αλλά θα μπορούσε να είναι και το 1234:abcdefg. Μερικές υπηρεσίες ιστού, χρησιμοποιούν τα namespaces για να δείξουν την τοποθεσία των μεθόδων που πρόκειται να εκτελεστούν, όπως φαίνεται και στο παρακάτω παράδειγμα:

<p:GetOrderStatus xmlns:p="http://www.xmlbus.com/Skateboots/OrderEntry">
<OrderID>12345</OrderID>
</p:GetOrderStatus> (Papazoglou, 2008, Πουλημενοπουλου, 2009)

2.3.2 Πρωτόκολλο Πρόσβασης Απλού Αντικειμένου (Simple Object Access Protocol - SOAP)

Το πρωτόκολλο SOAP αποτελεί ένα ελαφρύ (lightweight) πρωτόκολλο που βασίζεται στη γλώσσα XML για την ανταλλαγή τυποποιημένων (structured) δεδομένων μεταξύ εφαρμογών. Τα κύρια χαρακτηριστικά του είναι:

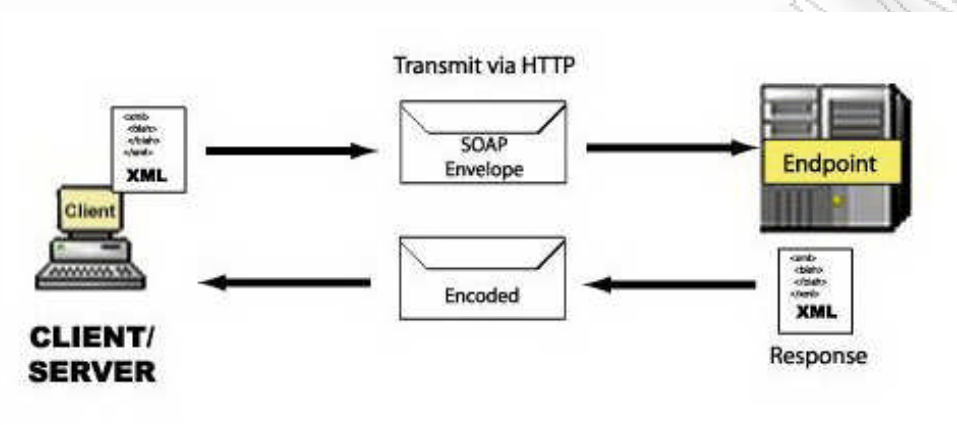
- Απλό και επεκτάσιμο
- Βασίζεται στην XML
- Ανεξάρτητο πρωτοκόλλου, λειτουργικού συστήματος και γλώσσας προγραμματισμού (Το SOAP με το HTTP είναι το πιο συχνά χρησιμοποιούμενο)
- Απομακρυσμένη κλήση μεθόδων (Remote Procedure Calls-RPC)) ή ασύγχρονη αποστολή μηνυμάτων (asynchronous message style)

Οι χρήσεις του SOAP είναι :

- Χρησιμοποιείται συχνά ως πρωτόκολλο για απομακρυσμένες κλήσεις πάνω από το πρωτόκολλο Hypertext Transfer Protocol (HTTP).
- Δεν είναι αναγκαία η χρήση του SOAP με το πρωτόκολλο HTTP ενώ παράλληλα υποστηρίζει και άλλα μοντέλα εκτός του μοντέλου αίτηση/απάντηση.
- Το SOAP χρησιμοποιείται με κάθε πρωτόκολλο που υποστηρίζει τη μεταφορά XML δεδομένων από τον αποστολέα στο παραλήπτη.

Η Microsoft και η IBM έχουν υλοποιήσει μηνύματα SOAP πάνω από το πρωτόκολλο SMTP, το οποίο σημαίνει ότι τα μηνύματα SOAP μπορούν να δρομολογηθούν μέσω e-mail servers. Το κατώτερο επίπεδο του SOAP είναι ένα ελαφρύ πρωτόκολλο ανταλλαγής μηνυμάτων που μπορεί να χρησιμοποιηθεί για την αποστολή μηνυμάτων μεταξύ δύο ή περισσοτέρων σημείων. Ο κύριος στόχος του SOAP είναι η παροχή ενός κοινού τρόπου για το πακετάρισμα των δεδομένων των μηνυμάτων και ο καθο-

ρισμός κανόνων για τη κωδικοποίηση και αποκωδικοποίηση των δεδομένων κατά τη μεταφορά. Στο παρακάτω σχήμα φαίνεται η διαδικασία SOAP. (Πουλημενοπουλου, 2009, Papazoglou, 2008)



Εικόνα 16: Διαδικασία SOAP μηνύματος (http://wiki.eeng.dcu.ie:8888/ee557/g2/740-EE.html)

Μοντέλα Ανταλλαγής Δεδομένων – Message Exchange Models

Το πρωτόκολλο SOAP είναι κατά κύριο λόγο πρωτόκολλο μιας κατεύθυνσης (One-way), παρόλα αυτά μπορεί να υποστηρίξει και άλλα μοντέλα ανταλλαγής δεδομένων, αρκεί να τα υποστηρίζει το πρωτόκολλο που χρησιμοποιείται σε συνδυασμό με το SOAP. Τέτοια μοντέλα ανταλλαγής δεδομένων είναι:

- πολλαπλή αποστολή (multicast),
- αίτηση/απάντηση (request/response).

Για παράδειγμα, το μοντέλο αίτηση/απάντηση υλοποιείται με 2 SOAP μηνύματα: 1 για την αίτηση και 1 για την απάντηση.

HTTP Bindings

Αρχικά το SOAP συνδέθηκε με το πρωτόκολλο HTTP επειδή σχεδιάστηκε ως ένα πρωτόκολλο που θα υποστηρίζει υπηρεσίες στο Internet. Ελάχιστες είναι οι απαιτήσεις για το συνδυασμό του SOAP με HTTP.

- Μία αίτηση SOAP μπορεί να είναι μια HTTP POST ή μια HTTP GET αίτηση.

- Η HTTP POST αίτηση καθορίζει τουλάχιστον 2 επικεφαλίδες: Content-Type and Content-Length.

Το πεδίο *Content-Type* για μια αίτηση ή απάντηση SOAP, καθορίζει τον τύπο δεδομένων ενός μηνύματος και προαιρετικά την κωδικοποίηση που χρησιμοποιείται στο κυρίως σώμα της αίτησης ή της απάντησης.

Το πεδίο *Content-Length* για μία αίτηση ή απάντηση SOAP καθορίζει τον αριθμό των bytes στο κυρίως σώμα (body) της αίτηση ή της απάντησης αντίστοιχα. Στον παρακάτω πίνακα, παρουσιάζεται η σύνταξη των δυο επικεφαλίδων και ένα παράδειγμα για την καθεμία.

	Σύνταξη	**Παράδειγμα**
Content-Type	Content-Type: MIMEType; charset=character-encoding	POST/item HTTP/1.1 Content-Type: application/soap+xml; charset=utf-8
	Σύνταξη	**Παράδειγμα**
Content-Length	Content-Length: bytes	POST /item HTTP/1.1 Content-Type: application/soap+xml; charset=utf-8 Content-Length: 250

Πίνακας 2: Σύνταξη επικεφαλίδων Content-Type και Content Length

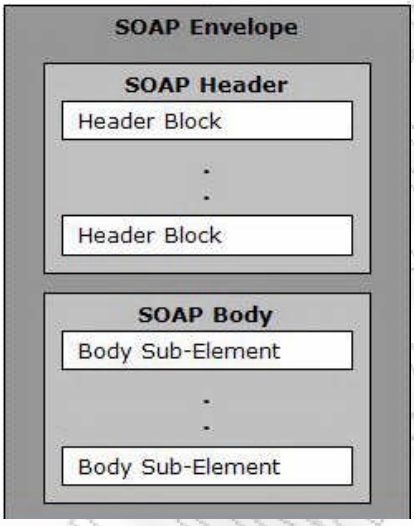
HTTP Response

Η επικεφαλίδα μιας SOAP απάντησης δεν περιέχει κάτι περισσότερο από ότι μια επικεφαλίδα μιας SOAP αίτησης, παρά μόνο τους κωδικούς κατάστασης (status codes). Για κάθε επιτυχή λήψη ενός HTTP SOAP μηνύματος επιστρέφεται ένας κωδικός κατάστασης. Σε περίπτωση προβλήματος, ο κωδικός κατάστασης που επιστρέφεται είναι 500, “Internal Server Error”. Εάν ο κωδικός κατάστασης είναι 500, τότε το SOAP μήνυμα πρέπει να περιέχει ένα στοιχείο Fault (fault element).

SOAP Message

Ένα SOAP μήνυμα αποτελείται από το φάκελο του SOAP (SOAP envelope) μηνύματος. Μέσα στο φάκελο εσωκλείονται:

- η επικεφαλίδα (header) και
- τα κυρίως δεδομένα που βρίσκονται στο σώμα (body).



Εικόνα 17: Φάκελος SOAP (Papazoglou, 2008)

Ο SOAP φάκελος δεν διαφέρει πολύ από έναν πραγματικό φάκελο, παραδοσιακού γράμματος. Όπως προαναφέρθηκε, στον φάκελο περιέχονται τα κύρια δεδομένα του μηνύματος, τα οποία κωδικοποιούνται σε ένα SOAP φορτίο. Στα κύρια αυτά δεδομένα, συμπεριλαμβάνονται και πληροφορίες που αφορούν τον παραλήπτη και τον αποστολέα όπως επίσης και λεπτομέρειες για το ίδιο το μήνυμα. Για παράδειγμα, από την επικεφαλίδα του SOAP φακέλου μπορεί να καθοριστεί ο τρόπος με τον οποίο το μήνυμα πρέπει επεξεργαστεί. Πριν συνεχιστεί η διαδικασία επεξεργασίας ενός μηνύματος από μια εφαρμογή, είναι δυνατόν να προσδιοριστεί η πληροφορία για το ίδιο το μήνυμα.

Επίσης ένα τυπικό SOAP μήνυμα, μπορεί να περιλαμβάνει τον τρόπο κωδικοποίησης. Σε ένα SOAP message μπορούν να χρησιμοποιηθούν οι κοινοί τύποι δεδομένων :

- XSD types: int, String, date (π.χ. <age xsi:type=xsd:int>66</age>) και

- Πιο πολύπλοκες δομές δεδομένων οι οποίες περιγράφονται σε ένα σχήμα XML.

Κωδικοποίηση καλείται η διαδικασία μετάφρασης των δεδομένων σε XML. Στην παρακάτω εικόνα, φαίνεται ο SOAP φάκελος με συγκεκριμένη κωδικοποίηση

```
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
soap:encodingStyle="http://myHost.com/encodings/secureEncoding"
>
<soap:Body>
<article xmlns="http://www.ibm.com/developer">
<name>Soapbox</name>
<url>
http://www-106.ibm.com/developerworks/library/x-soapbx1.html
</url>
</article>
</soap:Body>
</soap:Envelope>
```

Εικόνα 18: Παράδειγμα SOAP Φακέλου (IBM-Web Services)

Το χαρακτηριστικό encodingStyle, δείχνει πως θα αναπαρίστανται τα δεδομένα στο μήνυμα. Αυτό το χαρακτηριστικό μπορεί να εμφανιστεί οπουδήποτε μέσα στο μήνυμα, αλλά συνήθως εμφανίζεται στο φάκελο (Envelope). Στην εικόνα, φαίνεται ότι η κωδικοποίηση ορίζεται μέσα στον φάκελο, έτσι η εφαρμογή μπορεί να καθορίσει, χρησιμοποιώντας την τιμή του encodingStyle, εάν μπορεί να διαβάσει το εισερχόμενο μήνυμα το οποίο βρίσκεται μέσα στο σώμα (body).

Το πρόθεμα του namespace "soap" χρησιμοποιείται στα περισσότερα πεδία ενός SOAP μηνύματος. Στο παράδειγμα της εικόνας 17, αυτό το πρόθεμα σχετίζεται με το namespace URL http://schemas.xmlsoap.org/soap/envelope/, και αναγνωρίζει τα πεδία που είναι μέρος ενός τυπικού SOAP μηνύματος. Όπως όλα τα προθέματα namespace, η επιλογή του soap είναι τυχαία. Έτσι για παράδειγμα αντί για «soap» οτιδήποτε άλλο μπορεί να χρησιμοποιηθεί για πρόθεμα όπως π.χ.

<blah:Envelope
xmlns:blah="http://schemas.xmlsoap.org/soap/envelope/"
blah:encodingStyle="http://myHost.com/encodings/SecureEncoding/">

Το namespace του εγγράφου SOAP μπορεί να παραληφθεί εντελώς όταν αυτό που χρησιμοποιείται είναι το τυπικό για το έγγραφο. Αυτό δηλώνεται με τη χρήση του χαρακτηριστικού xmlns. Για παράδειγμα :

<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

Ένα Namespace για το Envelope είναι το URL "http://schemas.xmlsoap.org/soap/envelope/". Μηνύματα που δε χρησιμοποιούν αυτό το namespace είναι άκυρα και οπότε αυτοί που τα λαμβάνουν επιστρέφουν λάθος.

Επίσης, σε ένα μήνυμα SOAP δεν είναι υποχρεωτική η επικεφαλίδα. Εάν όμως υπάρχει, τότε αυτή είναι και το πρώτο παιδί μετά το χαρακτηριστικό Envelope. Όταν υπάρχει επικεφαλίδα, το κυρίως σώμα ακολουθεί την επικεφαλίδα αλλιώς είναι το πρώτο παιδί μετά το χαρακτηριστικό Envelope.

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

<soap:Header>……</soap:Header>
<soap:Body>…..</soap:Body>
</soap:Envelope>

Χρήσεις του SOAP Header

Οι χρήσεις της επικεφαλίδας συνοπτικά, θα μπορούσαμε να πούμε ότι είναι:

- Η επικεφαλίδα SOAP, μπορεί να περιέχει πληροφορία αυθεντικοποίησης, όπως όνομα χρήστη και συνθηματικό
- Επίσης, μπορεί να περιέχει πληροφορία για την συναλλαγή, όπως κάποιο κλειδί συναλλαγής
- Μπορεί να χρησιμοποιηθεί για την υλοποίηση πολλαπλής μεταφοράς, όπου στην επικεφαλίδα περιέχεται πληροφορία για τον επόμενο παραλήπτη του μηνύματος

- Τέλος, μπορεί να περιέχει πληροφορία για την κατάσταση του μηνύματος

Actor Attribute

Ένα μήνυμα μπορεί να περάσει από πολλούς εξυπηρετητές. Το χαρακτηριστικό actor χρησιμοποιείται σε μια κεφαλίδα για να δείξει ποιος πρέπει να διαχειριστεί την πληροφορία.

Εάν το χαρακτηριστικό actor καθορίζει κάποιον εξυπηρετητή, τότε αυτός πρέπει να αφαιρέσει αυτό το χαρακτηριστικό. Κατόπιν, ο εξυπηρετητής μπορεί να προσθέσει νέο πεδίο Header πριν το στείλει σε άλλο εξυπηρετητή.

Εάν η επικεφαλίδα δεν περιέχει χαρακτηριστικό actor, αυτό δείχνει ότι ο παραλήπτης του μηνύματος είναι ο actor. Παρόλα αυτά δεν είναι σίγουρο ότι ο παραλήπτης πρέπει ή μπορεί να καταλάβει το πεδίο Header.

MustUnderstand Attribute

Αυτό το χαρακτηριστικό χρησιμοποιείται για να δείξει εάν ο παραλήπτης πρέπει να καταλαβαίνει το μήνυμα. Οι τιμές που παίρνει το χαρακτηριστικό είναι 0 ή 1 (λογικές τιμές).

- Εάν η τιμή είναι 0, τότε σημαίνει ότι δεν είναι απαραίτητη η κατανόηση του μηνύματος από τον παραλήπτη
- Εάν η τιμή είναι 1 τότε σημαίνει ότι είναι απαραίτητη η κατανόηση του μηνύματος από τον παραλήπτη.
- Εάν το χαρακτηριστικό mustUnderstand δε χρησιμοποιείται, τότε παίρνει αυτόματα την τιμή 0.

SOAP Body

Στο πεδίο αυτό περιλαμβάνονται τα κυρίως δεδομένα του SOAP μηνύματος.

- Το στοιχείο Body περιέχει τα στοιχεία Body entries.
- Κάθε Body entry πρέπει να κωδικοποιηθεί ως ένα ανεξάρτητο στοιχείο με χαρακτηριστικά id και href.
- Κάθε Body entry πρέπει να ακολουθεί κάποιο namespace και είναι δυνατόν να περιέχει ένα χαρακτηριστικό encodingRules.
- Τα Body entries δεν περιέχουν χαρακτηριστικά actor και mustUnderstand όπως η επικεφαλίδα.
- Όλα τα στοιχεία του κυρίως σώματος του μηνύματος προορίζονται για τον τελικό παραλήπτη του μηνύματος.
- Είναι απαραίτητο να καταλαβαίνει το μήνυμα ο τελικός παραλήπτης
- Τα στοιχεία Header και Body δεν είναι άμεσα συνδεδεμένα μεταξύ τους, ά συνδέονται όμως μέσω των στοιχείων actor και την τιμή του mustUnderstand=1.

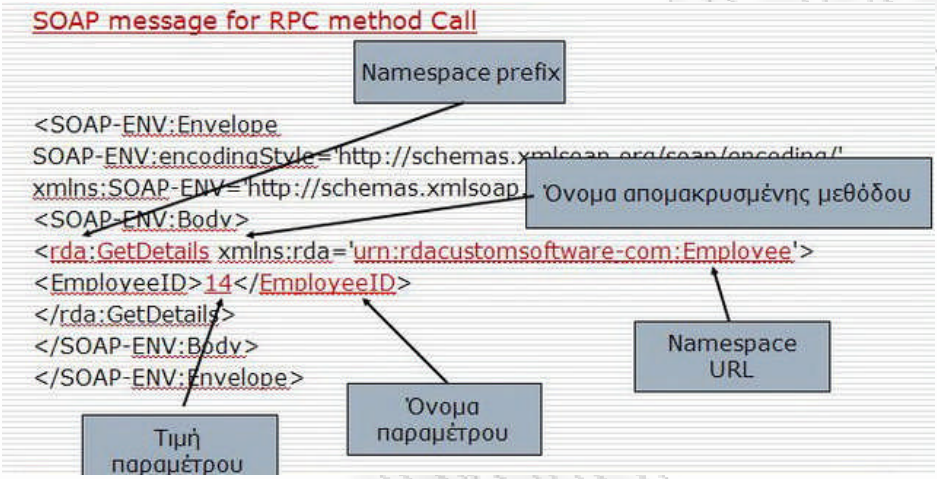
Κανόνες Υλοποίησης SOAP για RPC

Όπως ειπώθηκε στην αρχή, ένα από τα κύρια χαρακτηριστικά του SOAP είναι και η απομακρυσμένη κλήση μεθόδων, Remote Procedure Call. Οι κανόνες υλοποίησης SOAP για RPC είναι:

- Η μέθοδος που καλείται, εμφανίζεται στο SOAP μήνυμα σε μορφή XML με όνομα ίδιο με της μεθόδου. Ο τύπος δεδομένων πρέπει να αντιστοιχεί στον τύπο δεδομένων που επιστρέφει η μέθοδος
- Οι παράμετροι της μεθόδου περιλαμβάνονται στο μήνυμα με τη σειρά που εμφανίζονται στη μέθοδο και συνοδεύονται από τον τύπο δεδομένων τους.
- Οι απαντήσεις δομούνται με τον ίδιο τρόπο που δομούνται οι κλήσεις των μεθόδων. Το όνομα του πεδίου απάντησης δεν προσδιορίζεται, αλλά εισάγεται το “Response” στο όνομα της μεθόδου.
- Στην περίπτωση όπου η απάντηση επιστρέφει μια τιμή, τότε η τιμή αυτή περιέχεται στο πρώτο πεδίο. Ο τύπος δεδομένων πρέπει να ταυτίζεται με τον τύπο δεδομένων της επιστροφής της μεθόδου.

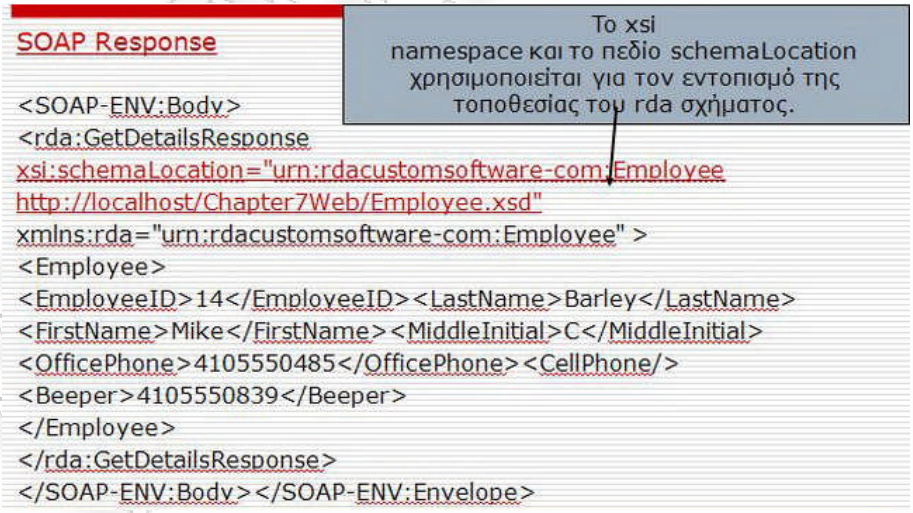
- Κάθε άλλη παράμετρος που επιστρέφεται τοποθετείται μετά το πεδίο "Response". Η σειρά των παραμέτρων πρέπει να είναι ίδια με τη σειρά που κατέχουν και στην μέθοδο.
- Εάν παρουσιαστεί κάποιο σφάλμα κατά την κλήση της μεθόδου, επιστρέφεται το πεδίο SOAP Fault

Στην παρακάτω εικόνα, φαίνεται ένα SOAP μήνυμα για μια απομακρυσμένη κλήση μεθόδων.



Εικόνα 19: SOAP message for RPC method Call (Πουλημενοπουλου, 2009)

Το SOAP μήνυμα απάντησης θα είναι όπως φαίνεται στην παρακάτω εικόνα:



Εικόνα 20: SOAP Response (Πουλημενοπουλου, 2009)

Ο αποστολέας και ο παραλήπτης ενός μηνύματος SOAP πρέπει να χρησιμοποιούν την ίδια μορφή serialization ώστε να είναι δυνατή η σωστή επεξεργασία και

ερμηνεία του μηνύματος. Ο αποστολέας και ο παραλήπτης ενός μηνύματος SOAP είναι απαραίτητο να έχουν :

- Το σχήμα για το φάκελο SOAP,
- Ένα σχήμα για τον προαιρετικό μηχανισμό SOAP κωδικοποίησης,
- Σχήματα για όποια προαιρετική επικεφαλίδα και
- Σχήματα για τα στοιχεία στο κυρίως σώμα ενός μηνύματος (application-specific data elements).

Τα σχήματα για την επικύρωση των διαφόρων μερών ενός μηνύματος SOAP συνήθως τοποθετούνται στο Internet, ώστε το κάθε εμπλεκόμενο μέρος της συναλλαγής να τα κατεβάζει και να ελέγχει ότι το μήνυμα είναι σωστό και έχει επεξεργαστεί σωστά. Τα σχήματα μπορούν επίσης να βρεθούν σε μια αποθήκη UDDI ή με τη χρήση της γλώσσας Web services inspection language (WS-Inspection).

Τα προαιρετικά σχήματα για τις επικεφαλίδες, καθορίζουν επιπλέον πληροφορία για τις επικεφαλίδες των μηνυμάτων, όπως προτεραιότητα και ημερομηνία λήξης, πληροφορία ασφάλειας κ.α. Οι επικεφαλίδες γενικά χρησιμοποιούνται ως ένας μηχανισμός επεκτασιμότητας για την προσθήκη δυνατοτήτων στο SOAP. Γενικά, ο αποστολέας και ο παραλήπτης ενός μηνύματος πρέπει να υποστηρίζουν τις ίδιες επικεφαλίδες, χρησιμοποιώντας τα ίδια σχήματα ή παρόμοια σχήματα. Σε μερικές υλοποιήσεις είναι πιθανή η διαπραγμάτευση μεταξύ του αποστολέα και του παραλήπτη στη μη χρήση της ίδιας επικεφαλίδας. Ένας επεξεργαστής SOAP είναι μια οντότητα που επεξεργάζεται ένα μήνυμα SOAP σύμφωνα με τους κανόνες της προδιαγραφής SOAP, για την προσπέλαση των υπηρεσιών που παρέχονται με τη χρήση του SOAP. Μπορεί να είναι ένας αποστολέας, ένας παραλήπτης ή ένας ενδιάμεσος, ο οποίος είναι αποστολέας και παραλήπτης μαζί. Ένας επεξεργαστής SOAP πρέπει να επικυρώσει το φάκελο SOAP χρησιμοποιώντας το σχήμα για το φάκελο SOAP, ενώ μπορεί να καταλάβει το σχήμα του SOAP και να επιστρέψει λάθος εάν το μήνυμα παραβιάζει ένα μέρος των XML ορισμών που χρησιμοποιούνται στο SOAP.

Πέρα από όλες αυτές τις ιδιότητες και τα πλεονεκτήματα που χαρακτηρίζουν το SOAP, αυτό έχει ένα μειονέκτημα. Δεν περιγράφει τα λειτουργικά χαρακτηριστικά των Web Services και πως τα δεδομένα μπορούν να εναλλαχθούν μεταξύ υπηρεσιών που αλληλεπιδρούν μεταξύ τους. Μια υπηρεσία SOAP χρειάζεται μερικά έγγραφα τα οποία να παρέχουν τις λεπτομέρειες της υπηρεσίας μαζί με τις παραμέτρους.

Το πρόβλημα αυτό, μπορεί να το λύσει μια γλώσσα περιγραφής υπηρεσιών. Η γλώσσα αυτή είναι άλλη μια εδραιωμένη τεχνολογία των Web Services και ονομάζεται Web Services Description Language. Στην παράγραφο 2.3.3, θα αναλυθεί ο σκοπός και η λειτουργία της γλώσσας αυτής καθώς επίσης και τα χαρακτηριστικά της, ενώ παράλληλα θα παρουσιάζονται και κάποια παραδείγματα. (Papazoglou, 2008, Πουλημενοπουλου, 2009)

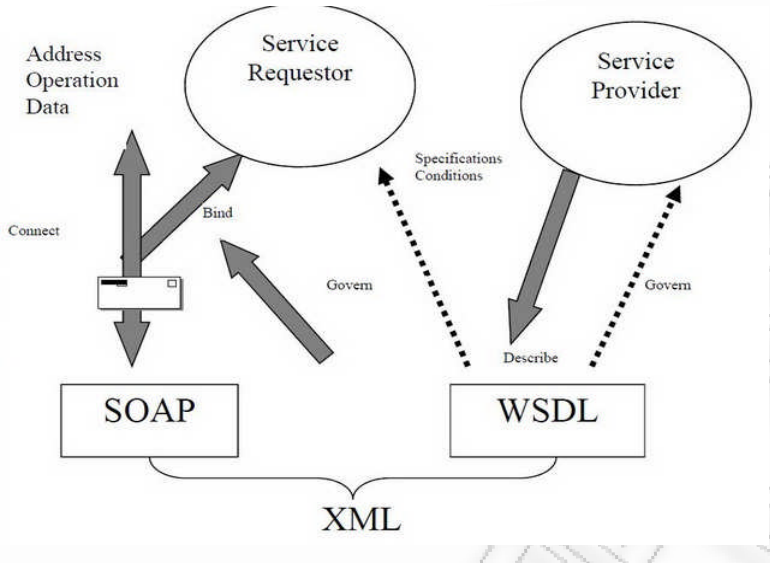
2.3.3 Γλώσσα Περιγραφής Υπηρεσιών Διαδικτύου (Web Services Description Language – WSDL)

Η Web Services Description Language (WSDL) αποτελεί μια γλώσσα βασισμένη σε XML και περιγράφει τους μηχανισμούς της αλληλεπίδρασης με μια συγκεκριμένη υπηρεσία ιστού. Συνοπτικά η WSDL περιγράφει το τι κάνει ένα web service, που ανήκει και πως καλείται.

Η WSDL βασίζεται στην XML για την περιγραφή των δημόσιων διεπαφών ενός Web Service όπως :

- Όλες τις δημόσιες δραστηριότητες
- Τα XML πρωτόκολλα μηνυμάτων υποστηριζόμενα από τα Web Services
- Πληροφορίες δέσμευσης για το συγκεκριμένο πρωτόκολλο μεταφοράς που πρόκειται να χρησιμοποιηθεί
- Πληροφορίες διεύθυνσης για τον εντοπισμό του web service

Αποτελεί ένα είδος δέσμευσης, μεταξύ του πελάτη της υπηρεσίας και του παρόχου της υπηρεσίας. Είναι όμως ανεξάρτητη της πλατφόρμας και της γλώσσας προγραμματισμού ενώ χρησιμοποιείται και για να περιγράψει υπηρεσίες τύπου SOAP και όχι μόνο.

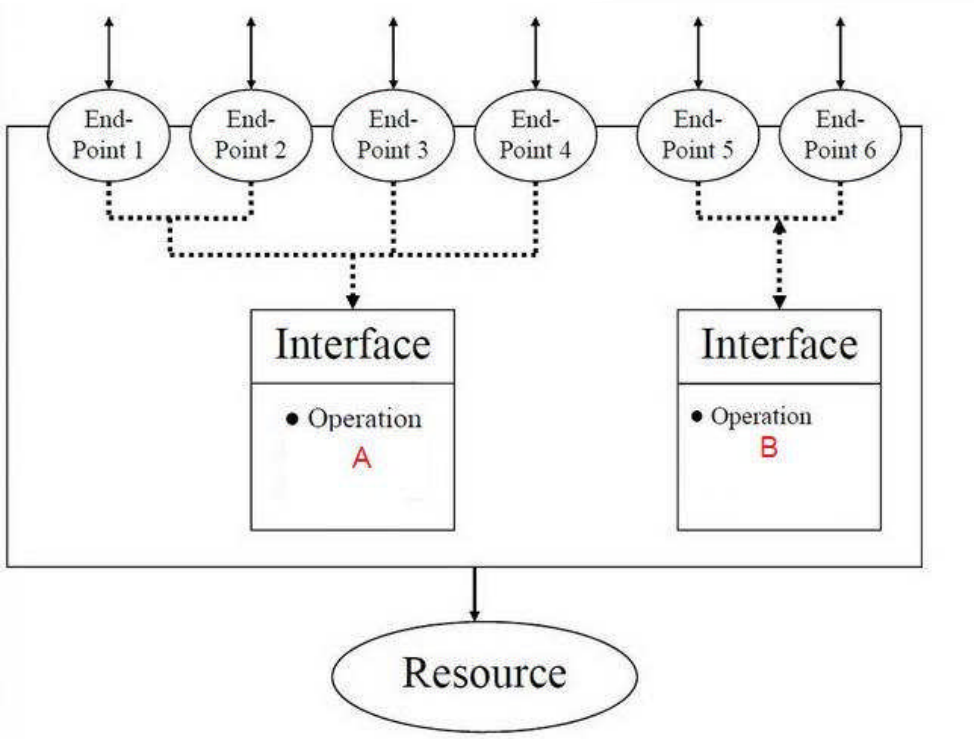


Εικόνα 21: WSDL περιγραφή υπηρεσίας (Papazoglou, 2008)

Οι προδιαγραφές της WSDL χωρίζονται σε δύο μέρη:

- Τον καθορισμό της διεπαφής της υπηρεσίας που περιγράφει την γενική δομή του Web Service, και
- Την υλοποίηση της υπηρεσίας που συνδέει το αφηρημένο (abstract) σε μια συγκεκριμένη διεύθυνση δικτύου, σε ένα συγκεκριμένο πρωτόκολλο και σε πραγματικές δομές δεδομένων

Η γραμματική και η σύνταξη που περιγράφουν ένα web service, καθορίζονται από την WSDL σαν μια συλλογή από επικοινωνιακά άκρα. Η WSDL χρησιμοποιεί την XML και τοποθετείται πάνω από ένα XML Schema. Μπορεί να παρέχει τα μέσα για την ομαδοποίηση των μηνυμάτων σε λειτουργίες και τις λειτουργίες σε διεπαφές. Όπως φαίνεται στο σχήμα της εικόνας 22, τα άκρα 1 εως 4 ομαδοποιούνται σε μια λειτουργία A (operation A) μέσω ενός interface ενώ η λειτουργία β (operation b) παράγει το interface των άκρων 5-6.



Εικόνα 22: WSDL Ομαδοποίηση (Θεμιστοκλεους, 2009b)

Ένα έγγραφο WSDL αποτελείται από τα ακόλουθα στοιχεία (elements):

- <types>
- <messages>
- <operation>
- <port type>
- <port>
- <binding>
- <service>

Τα στοιχεία αυτά τα οποία και συνοψίζονται παρακάτω περιγράφουν το αφηρημένο μέρος (abstract) ενός web service.

Types είναι οι τύποι δεδομένων οι οποίοι θα χρησιμοποιηθούν στα μηνύματα σε μορφή σχημάτων XML.

Message είναι ένας αφηρημένος ορισμός των δεδομένων σε μορφή μηνύματος. Παρουσιάζεται είτε ως ολόκληρο έγγραφο, είτε ως παράμετροι που θα αντιστοιχηθούν σε μια μέθοδο.

Operation είναι ένας αφηρημένος καθορισμός της μεθόδου του μηνύματος.

Port Type είναι ο αφηρημένος καθορισμός ομάδας μεθόδων για τα bindings. Οι μέθοδοι μπορούν να αντιστοιχηθούν σε διάφορα πρωτόκολλα μεταφοράς, με τη χρήση διάφορων bindings.

Port είναι ένα μοναδικό τελικό σημείο που ορίζεται σαν συνδυασμός μιας σύνδεσης και μιας διεύθυνσης δικτύου.

Binding είναι τα πραγματικά πρωτόκολλα και τύποι δεδομένων των μεθόδων και των μηνυμάτων για κάθε port type.

Service είναι ένας συνδυασμός ενός binding και μιας διεύθυνσης. Καθορίζει τη διεύθυνση για την επικοινωνία για την υπηρεσία.
Στη συνέχεια βλέπετε ένα παράδειγμα WSDL.

```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl"
targetNamespace="your namespace here"
xmlns:tns="your namespace here"
xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap">
<wsdl:types>
<xs:schema targetNamespace="your namespace here (could be another) "
xmlns:xsd="http://www.w3.org/2001/XMLSchema“>……..
</schema>
</wsdl:types>
<wsdl:message name="some operation input">…parts…</wsdl:message>
<wsdl:message name="some operation output">..parts..</wsdl:message>
<wsdl:portType name="your type name">…….……..</wsdl:portType>
<wsdl:binding name="your binding name" type="tns:porttype name above">
<!-- define style and transport in general and use per operation -->
```

</wsdl:binding>

<wsdl:service><!-- define a port using the above binding and a URL -->

</wsdl:service>

</wsdl:definitions>

Σε ένα web service είναι απαραίτητος ο ορισμός των εισερχόμενων και εξερχόμενων δεδομένων των μηνυμάτων και η σύνδεσή τους με τα εισερχόμενα και τα εξερχόμενα δεδομένα των μεθόδων των υπηρεσιών. Για αυτό το σκοπό χρησιμοποιούνται τα WSDL types. Τα πεδία types είναι XML έγγραφα ή μέρη εγγράφων.

Οι τύποι δεδομένων μπορεί να είναι ακέραιοι (integer), σύνολο χαρακτήρων (string), λογικές τιμές (Boolean) και ημερομηνίες (date) και υποστηρίζονται από σχήματα XML. Επιπλέον, μπορεί να είναι σύνθετοι τύποι δεδομένων, όπως είναι οι δομές (structures) και οι πίνακες (arrays). Ένας ή περισσότεροι τύποι δεδομένων (types) αντιστοιχούνται σε μηνύματα (messages) και κατόπιν το ίδιο μήνυμα μπορεί να αντιστοιχηθεί σε μία ή πολλές μεθόδους.

Στο πεδίο <portType> δηλώνεται το σύνολο των λειτουργιών. Στο πεδίο <operation> περιλαμβάνεται η αφηρημένη περιγραφή μιας λειτουργίας και τα μηνύματα (ένα εισερχόμενο και ένα ή περισσότερα εξερχόμενα μηνύματα). Η WSDL μέσω των port types, υποστηρίζει τους ακόλουθους τύπους λειτουργίας:

- Μιας κατεύθυνσης (One way): Το τερματικό σημείο λαμβάνει ένα μήνυμα
- Αίτηση – Απάντηση (request - response): Το τερματικό σημείο λαμβάνει ένα μήνυμα και στέλνει πίσω ένα άλλο
- Κοινοποίηση (notification): Είναι το αντίθετο της one way. Το τερματικό σημείο, στέλνει ένα μήνυμα
- Αίτηση απάντησης (solicit response): Το τερματικό σημείο, στέλνει ένα μήνυμα και περιμένει απάντηση

One way

<message name="newTermValues">

<part name="term" type="xs:string"/>

```
        <part name="value" type="xs:string"/>
</message>
<portType name="glossaryTerms">
        <operation name="setTerm">
                <input name="newTerm" message="newTermValues"/>
        </operation>
</portType >
```

Request/Response

```
<message name="getTermRequest">
        <part name="term" type="xs:string"/>
</message>
<message name="getTermResponse">
        <part name="value" type="xs:string"/>
</message>
<portType name="glossaryTerms">
<operation name="getTerm">
        <input message="getTermRequest"/>
        <output message="getTermResponse"/>
        <fault message=”faultMessage”/>
</operation>
</portType>
```

Notification Operation

```
<operation name=”deliveryStatus”>
<output message=”trackingInformation”/>
</operation>
```

Solicit/Response Operation

```
<operation name=”clientQuery”>
```

<output message=”bandwidthRequest”/>
<input message=”bandwidthInfo”/>
<fault message=”faultMessage”/>
</operation>

Όσον αφορά το πεδίο Binding, αυτό καθορίζει τις λεπτομέρειες σχετικά με το πρωτόκολλο μεταφοράς και ουσιαστικά συνδέει τη μορφή των μηνυμάτων που αποστέλλονται για την κλήση των λειτουργιών, με τα μηνύματα που καθορίζονται στα portType. Επίσης, καθορίζει ένα πρωτόκολλο από τα SOAP (SOAP over HTTP, SOAP over SMTP) και HTTP Get/Post ενώ παράλληλα παρέχει και μηχανισμούς επεκτασιμότητας.

Το πεδίο <binding> παρέχει τη σύνδεση ανάμεσα στο φυσικό κομμάτι (δεδομένα) και το λογικό κομμάτι (μηνύματα και port types). Συγκεκριμένα, συνδέει τη λειτουργία που είχε οριστεί στο port type με το πώς θα μεταφερθεί μέσω του SOAP. Για παράδειγμα :

```
<wsdl:binding name="LabManagementSoapBinding" type="intf:LabManagement">
        <wsdlsoap:binding style="document"
        transport="http://schemas.xmlsoap.org/soap/http"/>
```

Οι πιθανές τιμές για το Style είναι το RPC και το Document. Πιθανότερο είναι το Document αφού και η διαφορά μεταξύ τους είναι μικρή. Το πεδίο αυτό, αφορά όλες τις λειτουργίες. Με το transport, ορίζεται το πρωτόκολλο με το οποίο θα μεταφερθεί μέσω του SOAP.

Τέλος, το πεδίο <service> καθορίζει μια φυσική τοποθεσία, δίνοντας μια web διεύθυνση ή ένα URL όπου τρέχει η υπηρεσία. Όπως για παράδειγμα εδώ:

```
<wsdl:service name="LabManagementService">
        <wsdl:portbinding="intf:LabManagementSoapBinding"
        name="LabManagement">
        <wsdlsoap:address
        location="http://localhost:9080/WSProject/services/LabManagement"/>
</wsdl:port>
 </wsdl:service>
```

(Papazoglou, 2008, Θεμιστοκλεους, 2009b, Πουλημενοπουλου, 2009)

2.3.4 Παγκόσμια Περιγραφή, Ανακάλυψη και Ολοκλήρωση (Universal Discovery Description and Integration - UDDI)

Το UDDI παρέχει ένα παγκόσμιο αρχείο καταγραφής υπηρεσιών δικτύου για να διευκολύνει τις επιχειρήσεις στη γρήγορη, εύκολη και δυναμική εύρεση και συναλλαγή με άλλες επιχειρήσεις. Συγκεκριμένα, το UDDI επιτρέπει σε μια επιχείρηση να:

- περιγράψει την επιχείρηση και τις υπηρεσίες της
- ανακαλύψει άλλες επιχειρήσεις που προσφέρουν επιθυμητές υπηρεσίες
- συνεργαστεί με άλλες επιχειρήσεις

Η πρόταση του UDDI προήλθε από την συνεργασία των IBM, Microsoft και Ariba στις προσπάθειες B2B (IBM-Ariba), XML SOAP (IBM, Microsoft) και BizTalk, cXML (Microsoft, Ariba). Το UDDI, ορίζει τον τρόπο καταχώρησης των Web Services στο μητρώο και είναι βασισμένο και αυτό σε XML. Στόχος του είναι η παροχή αναγκαίας υποδομής για την περιγραφή και αναζήτηση Web Services.

Στις αρχές της ανάπτυξής του, το UDDI είχε θεωρηθεί ως ένα "Universal Business Registry", παρόμοιο με μηχανές αναζήτησης (π.χ. Google). Σήμερα, το UDDI είναι πολύ περισσότερο ρεαλιστικό και αναγνωρίζει την επικρατούσα κατάσταση στις B2B αλληλεπιδράσεις

Σε αυτό το σημείο θα δούμε τις προδιαγραφές του UDDI :

- η έκδοση 1 καθόρισε τη βάση για ένα business service registry
- η έκδοση 2 προσάρμοσε την εργασία του registry στα SOAP και WSDL
- η έκδοση 3 επαναπροσδιορίζει τον ρόλο και σκοπό των UDDI registries, δίνει μεγάλη προσοχή στον ρόλο των private εφαρμογών, και αντιμετωπίζει το πρόβλημα αλληλεπίδρασης μεταξύ private και public UDDI registries

Ο αριθμός των UDDI registries που είναι σε λειτουργία είναι μικρός και αξιοποιούνται και συντηρούνται κυρίως από τις IBM, Microsoft, SAP, κτλ. Αυτά τα web registries αυτά αποτελούν συνήθως την πρώτη επιλογή για αναζήτηση web services. Δυστυχώς, τα registries είναι ακόμη πολύ μικρά και οι περισσότερες καταχωρήσεις

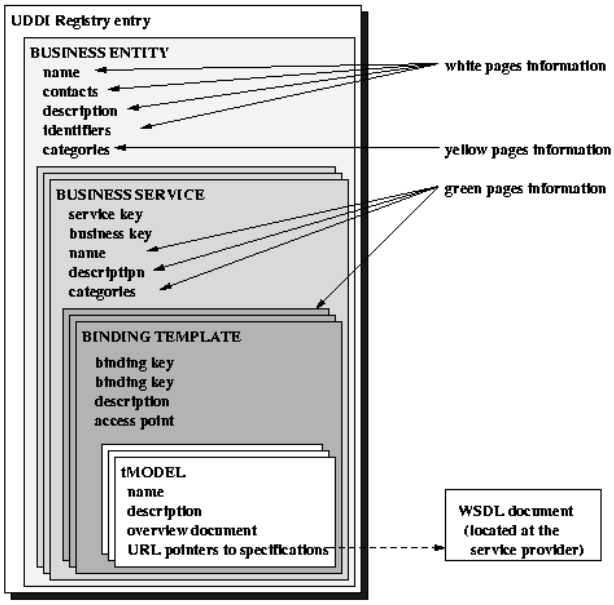
σε αυτά δεν λειτουργούν ή δεν αντιστοιχούν σε υπάρχουσες υπηρεσίες. Με αφορμή αυτό το γεγονός ασκήθηκε γενικότερη κριτική στα Web Services, όχι άδικη αλλά με λάθος τρόπο: το πρόβλημα δεν βρίσκεται στο UDDI αλλά στον τρόπο με τον οποίο αυτό χρησιμοποιείται και στις υποθέσεις σχετικά με τη δυναμικότητα των Web Services. Το UDDI είναι χρήσιμο και λειτουργικό, και αυτό θα αποδειχθεί εάν αντιμετωπιστεί ως μια υποδομή υποστήριξης των Web Services σε περιβάλλοντα χωρίς δημόσια πρόσβαση. Τα περισσότερα UDDI registries σήμερα είναι private και λειτουργούν ενδοεπιχειρησιακά ή συντηρούνται από ένα σύνολο εταιριών με private τρόπο. Πλέον το UDDI αποτελεί έγκυρο τρόπο τεκμηρίωσης των Web Services και παρέχει πληροφορία η οποία δεν παρέχεται από τις WSDL περιγραφές τους.

Οι διαθέσιμες μέσω Internet υπηρεσίες απαιτούν πολύ περισσότερες πληροφορίες από μια τυπική middleware υπηρεσία. Σε πολλές middleware υλοποιήσεις, η υπηρεσία και η εφαρμογή που την χρησιμοποιεί υλοποιούνται από τα ίδια άτομα. Με τον καιρό όμως αυτός ο τρόπος ξεπεράστηκες, και έτσι η χρησιμοποίηση μιας υπηρεσίας απαιτεί μεγαλύτερο όγκο πληροφοριών από αυτόν που είναι συνήθως διαθέσιμος σε ενδοεταιρικές υπηρεσίες. Η τεκμηρίωση αυτή έχει τρείς συνιστώσες:

- βασική πληροφορία
- κατηγοριοποίηση
- δεδομένα τεχνικής φύσεως

Μια καταχώρηση σε ένα UDDI registry είναι ένα έγγραφο XML που αποτελείται από διαφορετικούς τύπους δεδομένων. Οι πιο σημαντικοί από αυτούς φαίνονται και στην εικόνα 23 και είναι οι εξής :

- businessEntity: περιγράφει μια επιχείρηση ή οργανισμό που παρέχει την υπηρεσία.
- businessService: περιγραφή των Web services που προσφέρονται από την businessEnity.
- bindingTemplate: περιγράφει τις τεχνικές λεπτομέρειες οι οποίες απαιτούνται εκτέλεση μιας υπηρεσίας
- tModel: (“technical model”): ένας γενικής φύσεως τύπος δεδομένου ο οποίος χρησιμοποιείται για την αποθήκευση επιπλέον πληροφορίας σχετικά με την υπηρεσία

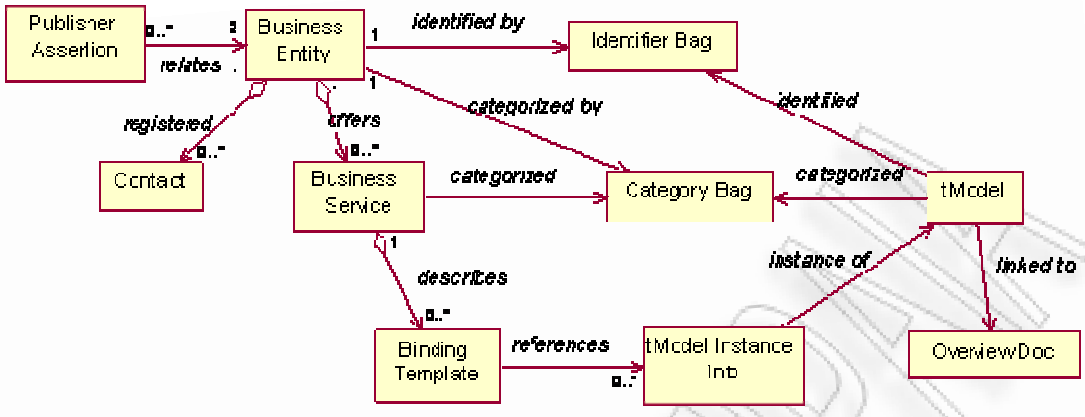


Εικόνα 23: Στοιχεία UDDI Registry (Δημητρακοπουλος, 2008)

Οι υπηρεσίες που μπορεί να προσφέρει το UDDI, είναι οι εξής :

- πληροφορίας white pages: δεδομένα σχετικά με τον πάροχο της υπηρεσίας (όνομα, διεύθυνση, πρόσωπο επικοινωνίας κτλ.)
- πληροφορίας yellow pages: περιγράφουν τί είδους υπηρεσίες προσφέρονται, καθώς και μια λίστα όλων των διαφορετικών υπηρεσιών
- πληροφορίας green pages: περιγράφονται οι τεχνικές λεπτομέρειες που αφορούν το πώς χρησιμοποιείται καθεμιά από τις προσφερόμενες υπηρεσίες, συμπεριλαμβάνοντας δείκτες προς την WSDL περιγραφή τους (η οποία δεν βρίσκεται στο UDDI registry)

Το εννοιολογικό μοντέλο UDDI όπως φαίνεται στην εικόνα 24, απεικονίζει τις συσχετίσεις μεταξύ των τύπων δεδομένων του UDDI ενώ στη συνέχεια περιγράφονται οι λειτουργίες αυτών των τύπων δεδομένων.



Εικόνα 24 Εννοιολογικό Μοντέλο UDDI (Δημητρακοπουλος, 2008)

Business Entity

Το BusinessEntity αποθηκεύει την πληροφορία white και yellow pages σχετικά με έναν πάροχο υπηρεσιών και οποίο περιλαμβάνει τα εξής δεδομένα:

- κάθε businessEntity έχει ένα businessKey
  discoveryURLs: μια λίστα των URLs που δείχνουν σε εναλλακτικούς, βασισμένους σε αρχεία μηχανισμούς αναζήτησης υπηρεσιών.
- Name: πληροφορία υπό τη μορφή κειμένου
- Business Description: πληροφορία υπό τη μορφή κειμένου
- Contacts: πληροφορία υπό τη μορφή κειμένου
- businessServices: μια λίστα με τις υπηρεσίες που παρέχονται από το συγκεκριμένο businessEntity
- identifierBag: μια λίστα με εξωτερικούς identifiers
- categoryBag: μια λίστα με κατηγορίες επιχειρήσεων

Το businessEntity δεν αντιπροσωπεύει υποχρεωτικά η εταιρία. Χρησιμοποιείται για την αναπαράσταση οποιασδήποτε οντότητας η οποία προσφέρει υπηρεσίες. Μπορεί να είναι ένα παράρτημα, μια ομάδα ανθρώπων, ένας server, ένα σύνολο servers κτλ.

Business Service

Οι υπηρεσίες που παρέχονται από ένα business entity περιγράφονται με την χρήση στοιχείων businessService. Ένα στοιχείο businessService περιγράφει ένα μεμονωμένο Web Service ή ένα σύνολο σχετικών μεταξύ τους Web services τα οποία προσφέρονται από το ίδιο businessEntity.

Ένα businessEntity μπορεί να έχει πολλά businessServices αλλά ένα businessService ανήκει σε ένα businessEntity. Το businessService μπορεί να παρέχεται από ένα διαφορετικό businessEntity από αυτό, στο οποίο βρίσκεται το στοιχείο. Το φαινόμενο αυτό ονομάζεται projection και επιτρέπει την περίληψη υπηρεσιών που παρέχονται από άλλους οργανισμούς.
Το στοιχείο businessService περιέχει:

- ένα serviceKey που προσδιορίζει μονοσήμαντα την υπηρεσία και το businessEntity όχι απαραίτητα αυτό στο οποίο περιέχεται το businessService
- Name
- Description
- categoryBag
- bindingTemplates: μια λίστα με όλα τα bindingTemplates της υπηρεσίας, με τεχνικές πληροφορίες, σχετικά με το πως πραγματοποιείται η προσπέλαση και χρήση της υπηρεσίας

Binding Template

Ένα binding template περιέχει την τεχνική πληροφορία που σχετίζεται με μια συγκεκριμένη υπηρεσία. Η πληροφορία που παρέχεται είναι η εξής:

- bindingKey
- serviceKey
- description

- accessPoint: η δικτυακή διεύθυνση της υπηρεσίας που παρέχεται (ουσιαστικά ένα URL αν και μπορεί να είναι οτιδήποτε καθώς το πεδίο αυτό είναι ένα string: π.χ., μια e-mail διεύθυνση ή ακόμη ένα τηλέφωνο)
- tModels: μια λίστα από εγγραφές που αντιστοιχούν σε tModels που σχετίζονται με το συγκεκριμένο binding. Η λίστα περιλαμβάνει αναφορές στα tModels, έγγραφα που περιγράφουν αυτά τα tModels, σύντομες περιγραφές, κλπ.
- categoryBag: πρόσθετη πληροφορία σχετικά με την υπηρεσία και το binding

Ένα businessService μπορεί να έχει διάφορα bindingTemplates αλλά ένα binding Template έχει μόνο ένα businessService. Το binding template μπορεί να θεωρηθεί ως ένας φάκελος στον οποίο συγκεντρώνεται όλη η τεχνική πληροφορία για ένα service.

tModel

Ένα tModel είναι ένας γενικός αποδέκτης πληροφοριών όπου οι σχεδιαστές μπορούν να γράψουν οποιαδήποτε τεχνική πληροφορία που σχετίζεται με τη χρήση ενός Web Service όπως το interface και το πρωτόκολλο που χρησιμοποιούνται συμπεριλαμβανομένου ενός δείκτη στην WSDL περιγραφή καθώς επίσης και την περιγραφή του επιχειρησιακού πρωτοκόλλου και των συναλλαγών που υποστηρίζονται από την υπηρεσία. Ένα tModel έγγραφο περιέχει τις εξής πληροφορίες:

- tModelKey
- name
- description
- overviewDoc: με το overviewURL και useType που υποδεικνύει που μπορεί να βρεθεί η πληροφορία και η μορφή της, π.χ., “text” or “wsdldescription”
- identifierBag
- categoryBag

Ένα tModel μπορεί να δείχνει σε άλλα tModels και τελικά θα προτυποποιηθούν διαφορετικές μορφές tModels (tModel για WSDL services, tModels για EDI based services, κλπ.)

Εικόνα 25: Παράδειγμα tModel (Δημητρακοπουλος, 2008)

Η προδιαγραφή του UDDI παρέχει έναν αριθμό από Προγραμματιστικές Διεπαφές (Application Programming Interfaces - APIs) που παρέχουν πρόσβαση σε ένα UDDI σύστημα. Οι διεπαφές αυτές είναι :

- UDDI Inquiry: Αφορά τον εντοπισμό λεπτομερειών καταχώρησης σε ένα UDDI registry και την υποστήριξη διαφόρων μοτίβων (αναζήτησης, drill-down, κλήσεις)
- UDDI Publication: Για την δημοσίευση και τροποποίηση πληροφορίας σε ένα UDDI registry.
- UDDI Security: Για έλεγχο πρόσβασης στο UDDI registry
- UDDI Subscription: Επιτρέπει στους clients να εγγραφούν σε αλλαγές στην πληροφορία στο UDDI registry (οι αλλαγές μπορούν να είναι στο αίτημα συνδρομής)
- UDDI Replication: Για δημιουργία αντιγράφων πληροφοριών διαμέσου των κόμβων σε ένα UDDI registry
- UDDI Custody and Ownership transfer: Για την αλλαγή του ιδιοκτήτη της πληροφορίας και τη μεταφορά της επιμέλειας από έναν κόμβο σε έναν άλλο μέσα σε ένα UDDI registry

Το UDDI παρέχει επίσης ένα σύνολο από APIs για clients ενός UDDI συστήματος:

- UDDI Subscription Listener: η πλευρά του πελάτη του subscription API
- UDDI Value Set: για την επικύρωση της πληροφορίας που παρέχεται σε ένα UDDI registry

Συμπερασματικά, διαπιστώνεται πως η προδιαγραφή UDDI είναι σχετικά πλήρης και περιλαμβάνει πολλές πλευρές ενός UDDI registry από τη χρήση του μέχρι την κατανομή του σε διάφορους κόμβους και τη συνοχή των δεδομένων σε ένα κατανεμημένο registry

Τα περισσότερα UDDI registries είναι ιδιωτικά και τυπικά χρησιμεύουν ως πηγή τεκμηρίωσης για προσπάθειες βασισμένες σε web services. Τα UDDI registries δεν προορίζονται απαραίτητα ως το τελικό σημείο αποθήκευσης πληροφοριών που αφορούν σε web services. Ακόμη και στην γενική εκδοχή του UDDI registry, το σκεπτικό είναι η προτυποποίηση των βασικών λειτουργιών και έπειτα η κατασκευή ιδιόκτητων εργαλείων για την αξιοποίηση του βασικού registry. Με αυτό τον τρόπο είναι δυνατή τόσο η προσαρμογή του σχεδιασμού όσο και η διατήρηση της απαραίτητης συμβατότητας μεταξύ των διαφορετικών χώρων αποθήκευσης.

Αν και είναι το πιο σαφές μέρος της προσπάθειας γύρω από τα Web Services, το UDDI είναι ίσως το λιγότερο κρίσιμο λόγω της πολυπλοκότητας των διεπιχειρησιακών αλληλεπιδράσεων όπως η καθιέρωση εμπιστοσύνης, διάφορες συμβάσεις, νομικοί περιορισμοί και διαδικασίες, κ.λπ. Ο βασικός στόχος είναι η πλήρης αυτοματοποίηση αλλά έως ότου συμβεί αυτό εκκρεμούν πολλά προβλήματα που πρέπει να επιλυθούν (Δημητρακοπουλος, 2008).

2.4 Εφαρμογή Web Services σήμερα

Σήμερα τα web services χρησιμοποιούνται σε ένα μεγάλο εύρος εφαρμογών που ποικίλουν και έχουν αναπτυχθεί για να λύσουν προβλήματα διαφορετικού τύπου. Τα web services έτυχαν ιδιαίτερης προσοχής και αποδοχής από τις επιχειρήσεις οι οποίες τα χρησιμοποιούν για εμπορικές και ενδοεπιχειρηματικές ολοκληρωμένες εφαρμογές. Με την ανάπτυξη των υπηρεσιών ιστού και την εφαρμογή τους από τις επιχειρήσεις εξαπλώθηκε το ηλεκτρονικό εμπόριο, το οποίο εκτός από την υποστήριξη της εσωτερικής λειτουργίας των επιχειρήσεων και την επικοινωνία και συνεργασία

τους με άλλες επιχειρήσεις, αποτελεί ένα θαυμάσιο εργαλείο για τον σχεδιασμό και την υλοποίηση της γενικότερης στρατηγικής με στόχο την εξασφάλιση ανταγωνιστικών πλεονεκτημάτων.

Οι κυριότερες μορφές ηλεκτρονικού εμπορίου είναι οι εξής:

- Επιχείρηση με Επιχείρηση (Business to Business ή B2B)
- Επιχείρηση με Καταναλωτή (business-to-consumer ή B2C) Online κρατήσεις & ασύρματες επικοινωνίες, Amazon.com
- Business-to-Employee (B2E) Online training & banking
- Καταναλωτή με καταναλωτή (consumer-to consumer ή C2C) – ebay.com
- Επιχείρηση με Κράτος (business-to-government).
- P2P Peer to Peer – gnutella.
- M-Commerce. Χρήση ασύρματων ψηφιακών συσκευών για συναλλαγές στο διαδίκτυο.

ΚΕΦΑΛΑΙΟ 3: ΜΕΛΕΤΗ ΠΕΡΙΠΤΩΣΗΣ

Το κεφάλαιο αυτό αναλύει την μελέτη περίπτωσης που ερευνούμε και μελετά σε βάθος την διαδικασία πάνω στην οποία βασίστηκε η εφαρμογή που αναπτύξαμε. Παρουσιάζονται αναλυτικά τα αποτελέσματα αυτής της έρευνας, καθώς γίνεται ο διαχωρισμός των ρόλων και της ροής δραστηριοτήτων που αφορά την περίπτωσή μας.

3.1 Πρόβλημα

Όπως έχει ήδη αναφερθεί, η παρούσα εργασία μελετά, αναλύει και αυτοματοποιεί λειτουργίες της γραμματείας του Προγράμματος Μεταπτυχιακών Σπουδών, του Τμήματος Ψηφιακών Συστημάτων. Η αυτοματοποίηση γίνεται μέσω της ανάπτυξης εφαρμογών web services.

Για να γίνει κατανοητό πώς ακριβώς λειτουργεί στην πράξη σήμερα η γραμματεία του τμήματος, γίνεται μια καταγραφή της ροής δραστηριοτήτων. Με βάση λοιπόν την ροή δραστηριοτήτων της γραμματείας, έγινε μια προσπάθεια ανάπτυξης ενός portal, στο οποίο θα ενσωματώνονται όλες αυτές τις λειτουργίες, και οι οποίες θα απευθύνονται στον αντίστοιχο χρήστη, κάθε φορά.

Για παράδειγμα, ένας υποψήφιος φοιτητής από την στιγμή που καταθέτει την απαραίτητη αίτηση και δικαιολογητικά, θα αποκτά ένα λογαριασμό για την πρόσβαση του στο σύστημα. Μέσα στο σύστημα θα έχει τη δυνατότητα να παρακολουθεί την κατάσταση στην οποία βρίσκεται η υποψηφιότητά του. Όταν θα ξεκινήσει η διαδικασία κατάρτισης προγράμματος συνεντεύξεων, θα προβάλλεται στην οθόνη του κάθε υποψήφιου η ημερομηνία και ώρα, στην οποία καλείται να παρευρεθεί για συνέντευξη από την αρμόδια επιτροπή συνεντεύξεων. Το σύστημα θα του δίνει τη δυνατότητα να αποδεχθεί ή να ζητήσει αλλαγή ημερομηνίας διεξαγωγής της συνέντευξής του σε περίπτωση που δεν είναι διαθέσιμος στην ημερομηνία που του ορίζεται. Δεν θα μπορεί όμως να ζητήσει κάποια συγκεκριμένη ημερομηνία, διότι αυτή η λειτουργία είναι στην δικαιοδοσία της γραμματείας. Η γραμματεία είναι αυτή που καταρτίζει το πρόγραμμα συνεντεύξεων των υποψηφίων. Άλλωστε, εάν ο κάθε υποψήφιος είχε τη δυνατότητα να ζητά και την ημερομηνία η οποία των εξυπηρετεί είναι αυτονόητο πως θα

επικρατούσε μια τεράστια σύγχυση στον τρόπο με τον οποίο θα καταρτιζόταν το τελικό πρόγραμμα συνεντεύξεων.

3.1.1 Διαδικασία Π.Μ.Σ.

Για να είμαστε σε θέση να σχεδιάσουμε και να αναπτύξουμε ένα σύστημα αυτοματοποίησης λειτουργιών γραμματείας, είναι απαραίτητη η μελέτη της διαδικασίας που ακολουθείται σήμερα στο Π.Μ.Σ. του Τμήματος. Λέγοντας διαδικασία εννοούμε όλες τις δραστηριότητες και λειτουργίες που διενεργούνται μέσα στο πλαίσιο λειτουργίας του Π.Μ.Σ.

Η αρχική έρευνα αφορούσε τον τρόπο λειτουργίας του μεταπτυχιακού προγράμματος σπουδών, στη συνέχεια μελετήθηκε η ροή δραστηριοτήτων του και μέσα από αυτήν πρόεκυψαν οι ρόλοι που συμμετέχουν στην όλη διαδικασία.

Μελετώντας τον τρόπο λειτουργίας, απορρέει ένα σύνολο λειτουργιών που εκτελούνται κυρίως από την γραμματεία, συνήθως όμως κατόπιν εντολής των αρμόδιων επιτροπών ή του Προέδρου. Στα πλαίσια του μεταπτυχιακού προγράμματος λοιπόν σήμερα, η γραμματεία δέχεται τις αιτήσεις των υποψηφίων, ελέγχει τα δικαιολογητικά τους, ενημερώνει τους υποψήφιους τηλεφωνικώς για τυχόν ελλείψεις των δικαιολογητικών τους, δημιουργεί το πρόγραμμα των συνεντεύξεων αξιολόγησης και εγγράφει τους επιτυχόντες στις κατευθύνσεις που γίνονται δεκτοί. Όλες αυτές οι ενέργειες αποτελούν την διαδικασία επιλογής υποψηφίων.

Παρατηρούνται όμως ορισμένα μειονεκτήματα. Για να είναι έγκυρα τα δικαιολογητικά των υποψηφίων, πρέπει να είναι επικυρωμένα από κάποιο δημόσιο φορέα κάτι το οποίο δεν επιτρέπει την ηλεκτρονική αποστολή αιτήσεων και δικαιολογητικών. Έτσι, η γραμματεία πρέπει να διαχειριστεί έναν μεγάλο όγκο εγγράφων πιστοποιημένων δικαιολογητικών, να τα καταχωρεί και να τα επεξεργάζεται. Το γεγονός αυτό προκαλεί καθυστερήσεις οι οποίες θα ήταν δυνατό να αποφευχθούν εάν γινόταν χρήση κάποιας ηλεκτρονικής υπηρεσίας.

Πέρα την διαδικασία που αφορά τους υποψήφιους και την επιλογή τους, η γραμματεία διαχειρίζεται οτιδήποτε σχετίζεται με το πρόγραμμα σπουδών, δηλαδή: μαθήματα, φοιτητές, παρουσίες, βαθμολογίες, διδακτικό προσωπικό. Για να πραγματοποιείται επιτυχώς αυτή η διαχείριση, η γραμματεία έρχεται σε επαφή και αλληλεπιδρά και με άλλους ρόλους, όπως επιτροπές αξιολόγησης, επιτροπές Π.Μ.Σ., Γενικές Συνελεύσεις, φοιτητές, ακαδημαϊκό προσωπικό κλπ. Έτσι λοιπόν προκύπτει ένα σύ-

νολο συμμετεχόντων, για το οποίο έγινε μια προσπάθεια διαχωρισμού ρόλων και δραστηριοτήτων που έχει ο καθένας από αυτούς μέσα στην όλη διαδικασία.

Η διαδικασία του Π.Μ.Σ. ουσιαστικά, χωρίζεται σε τρία στάδια: α) Προεγγραφής, το οποίο περιλαμβάνει την διαδικασία επιλογής υποψηφίων, β) εγγραφής μεταπτυχιακών φοιτητών, όπου εγγράφονται οι επιτυχόντες υποψήφιοι και γ) διαχείριση προγράμματος σπουδών, όπου γίνεται η διοικητική διαχείριση των μαθημάτων, των φοιτητών κλπ. Η συνολική διαδικασία λειτουργίας του Π.Μ.Σ., φαίνεται στον παρακάτω πίνακα.

Με την ισχίουσα διαδικασία, παρατηρεί κανείς πως οι μεταπτυχιακοί φοιτητές έχουν έναν παθητικό ρόλο. Εγγράφονται και οργανώνονται από την γραμματεία και δεν μπορούν να έχουν κάποια αλληλεπίδραση με αυτήν, πέρα από την τηλεφωνική επικοινωνία ή την φυσική παρουσία τους στο χώρο λειτουργίας της. Το γεγονός αυτό δυσκολεύει την εξυπηρέτηση των φοιτητών η οποία θα γίνοταν ευκολότερη με την ύπαρξη μιας ηλεκτρονικής υπηρεσίας.

Τρέχουσα Διαδικασία Π.Μ.Σ.					
	Διεργασία	**Περιγραφή**	**Εκτελείται από**	**Απαιτείται**	**Τρόπος εκτέλεσης**
Στάδιο Α	Α.1	Κατάθεση αιτήσεων - δικαιολογητικών	Υποψήφιους	-	Φυσική παρουσία
	Α.2	Παραλαβή αιτήσεων	Γραμματεία	Α.1	Χειρονακτικά
	Α.3	Έλεγχος αιτήσεων - δικαιολογητικών	Γραμματεία	Α.2	Χειρονακτικά
	Α.4	Ενημέρωση υποψηφίων για τυχόν ελλείψεις	Γραμματεία	Α.3	Τηλεφωνικώς
	Α.5	Ολοκλήρωση Ελέγχου δικαιολογητικών	Γραμματεία	Α.4	Χειρονακτικά
	Α.6	Πρόγραμμα συνεντεύξεων αξιολόγησης	Γραμματεία	Α.5	Η/Υ
	Α.7	Επίβλεψη αποτελεσμάτων αξιολόγησης	Συντονιστική επιτροπή Π.Μ.Σ.	Α.6	Παρουσία μελών επιτροπής
	Α.8	Τελική έγκριση επιτυχόντων	Γενική συνέλευση ειδικής σύνθεσης (Γ.Σ.Ε.Σ)	Α.7	Παρουσία μελών Γ.Σ.Ε.Σ.
	Α.9	Ενημέρωση υποψηφίων για την αξιολόγησή τους	Γραμματεία	Α.8	Τηλεφωνικώς
	Α.10	Κατάθεση προκαταβολής διδάκτρων	Επιτυχόντες	Α.9	Χειρονακτικά
	Α.11	Εγγραφή μεταπτυχιακών φοιτητών	Γραμματεία	Α.10	Χειρονακτικά & Η/Υ
Στάδιο Β	Β.1	Οργάνωση φοιτητών ανά κατεύθυνση	Γραμματεία	-	Χειρονακτικά & Η/Υ
	Β.2	Διαχείριση δεδομένων φοιτητών	Γραμματεία	Β.1	Η/Υ
	Β.3	Εξυπηρέτηση φοιτητών	Γραμματεία	-	Τηλεφωνικώς, φυσική παρουσία
Στάδιο Γ	Γ.1	Κατάρτιση Π.Μ.Σ	Γραμματεία	-	Η/Υ
	Γ.2	Οργάνωση Κατευθύνσεων	Γραμματεία	Γ.1	Χειρονακτικά & Η/Υ
	Γ.3	Οργάνωση προγράμματος σπουδών ανά κατεύθυνση	Γραμματεία	Γ.2	Χειρονακτικά & Η/Υ
	Γ.4	Διαχείριση δεδομένων προγράμματος σπουδών	Γραμματεία	Γ.3	Η/Υ

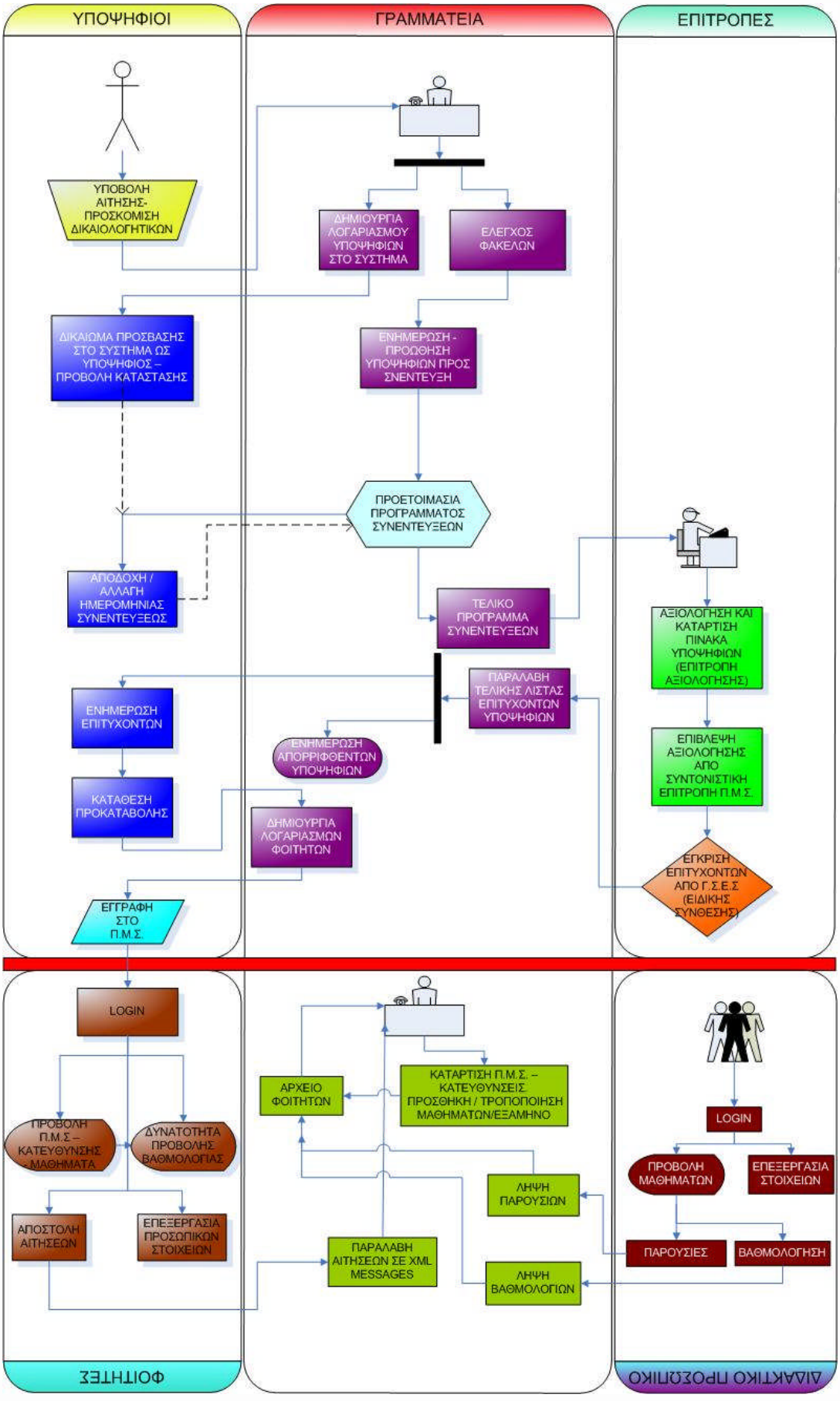
Πίνακας 3: Διαδικασία Π.Μ.Σ.

3.2 Ρόλοι και ροή δραστηριοτήτων

Αφού μελετήθηκε η ροή δραστηριοτήτων της διαδικασίας του Π.Μ.Σ. έγινε ο διαχωρισμός των ρόλων και των δραστηριοτήτων τους. Σχεδιάστηκε ένα διάγραμμα δραστηριοτήτων, το οποίο βασίστηκε στις πληροφορίες που λάβαμε απο την γραμματεία του Τμήματος και αφορούσαν τον τρόπο λειτουργίας του Π.Μ.Σ. και παρουσιάζει τον τρόπο λειτουργίας του υπό υλοποίηση συστήματος.

Στην παρακάτω εικόνα απεικονίζεται το διάγραμμα αυτό, όπου διακρίνονται οι χρήστες-ρόλοι οι οποίοι θα συμμετέχουν στο σύστημα καθώς επίσης και οι δραστηριότητες τις οποίες εκτελεί ο κάθε συμμετέχον στο σύστημα με τη σειρά. Στις επόμενες παραγράφους, περιγράφονται οι ρόλοι και οι δραστηριότητες αυτών αναλυτικά. Το διάγραμμα ροής δραστηριοτήτων το οποίο φαίνεται στην εικόνα 25, υλοποιήθηκε στο πρόγραμμα Visio, της πλατφόρμας Microsoft Office 2007

Το σύστημα που υλοποιήθηκε, χωρίζεται ουσιαστικά σε δύο μέρη. Το πρώτο μέρος, αφορά την διαδικασία επιλογής των υποψήφιων φοιτητών ενώ το δεύτερο μέρος αφορά τις λειτουργίες και υπηρεσίες που προσφέρει το σύστημα για την διαχείριση των μεταπτυχιακών φοιτητών.



Εικόνα 26: Διάγραμμα ροής δραστηριοτήτων

3.2.1 Ο ρόλος του Υποψήφιου

Ο πρώτος ρόλος λοιπόν που προκύπτει είναι αυτός του υποψήφιου. Για να είναι κάποιος υποψήφιος του Μεταπτυχιακού Προγράμματος Σπουδών, θα πρέπει να καταθέσει αίτηση υποψηφιότητας, συνοδευόμενη από τα απαραίτητα δικαιολογητικά και πιστοποιητικά. Στην συγκεκριμένη περίπτωση, ο υποψήφιος μπορεί να κατεβάσει την αίτηση από την ιστοσελίδα του τμήματος http://www.ted.unipi.gr/msc/, στην οποία εκτός των άλλων θα ενημερωθεί και για τις λεπτομέρειες υποβολής. Άρα η πρώτη ενέργεια του υποψηφίου, είναι η κατάθεση του φακέλου της υποψηφιότητάς του με τα απαραίτητα έγγραφα στην γραμματεία. Οι επόμενες ενέργειες που έχει τη δυνατότητα να πραγματοποιήσει ο υποψήφιος, εξαρτώνται από την γραμματεία όπου κάπου εδώ αρχίζει να κάνει αισθητό τον πρωταγωνιστικό της ρόλο. Για καλύτερη όμως τεκμηρίωση των εκάστοτε ρόλων και των δραστηριοτήτων αυτών, οι ενέργειες του κάθε ρόλου θα αναλυθούν στην αντίστοιχη παράγραφο. Έτσι, αφού αναφερθεί ότι η γραμματεία αφότου λάβει τον φάκελο υποψηφιότητας του υποψηφίου, δημιουργεί ένα λογαριασμό χρήστη για τον κάθε υποψήφιο, ο υποψήφιος όταν εισέλθει στο σύστημα θα έχει τη δυνατότητα να βλέπει σε ποια «κατάσταση» βρίσκεται η υποψηφιότητα του. Για παράδειγμα, όταν καταθέσει τον φάκελο υποψηφιότητάς του και δημιουργηθεί ο λογαριασμός του, όταν θα εισέλθει στο σύστημα με το όνομα χρήστη και τον κωδικό πρόσβασης που παραλαμβάνει από τη γραμματεία, θα δει στην οθόνη του το μήνυμα: *Ο φάκελος υποψηφιότητάς είναι υπό εξέταση.*

Στη συνέχεια, εάν ο φάκελος ο οποίος κατατέθηκε είναι πλήρης, θα ενημερωθεί από το σύστημα για την ημέρα και ώρα κατά την οποία πρέπει να προσέλθει προς συνέντευξη. Στο σύστημα θα έχει τη δυνατότητα είτε να αποδεχτεί την προτεινόμενη ημερομηνία, είτε να αιτηθεί αλλαγής ώρας. Σε αυτό το σημείο πρέπει να αναφερθεί πως η νέα καθορισμένη ώρα συνέντευξης που θα προκύψει από την αίτηση αλλαγής της αρχικής από τον υποψήφιο, θα είναι και η τελική. Δεν θα υπάρχει δυνατότητα επαναδιαπραγμάτευσης ή αλλαγής της τελικής ώρας. Αυτό γίνεται για ευνόητους λόγους λειτουργικότητας και θα πρέπει να φροντίσει ο κάθε υποψήφιος να προσαρμόσει το πρόγραμμα του ώστε να είναι διαθέσιμος προς συνέντευξη στην τελική καθορισμένη ημέρα και ώρα.

Ο υποψήφιος, έχει τη δυνατότητα να δηλώσει στην αίτησή του υποψηφιότητα για πέντε κατευθύνσεις, στα δύο Προγράμματα Μεταπτυχιακών Σπουδών τα οποία διαθέτει το Τμήμα Ψηφιακών Συστημάτων. Σε περίπτωση αποδοχής του, σε 2 ή πα-

ραπάνω κατευθύνσεις, η σειρά προτίμησης είναι αυτή η οποία θα καθορίσει τελικά την κατεύθυνση στην οποία θα γίνει αποδεκτός.

Όταν ολοκληρωθούν οι συνεντεύξεις του υποψηφίου, θα μεσολαβήσουν κάποιες διαδικασίες, οι οποίες αφορούν κάποιον άλλον ρόλο αυτόν των επιτροπών τον οποίο θα αναλύσουμε παρακάτω, έως ότου ληφθεί η τελική απόφαση που αφορά την αποδοχή ή όχι του εκάστοτε υποψηφίου στα Προγράμματα Μεταπτυχιακών Σπουδών.

Στην περίπτωση που κάποιος υποψήφιος γίνει δεκτός σε κάποιο Πρόγραμμα Μεταπτυχιακών Σπουδών, ενημερώνεται από το σύστημα, πάντα έχοντας εισέλθει πιστοποιημένα, και από την στιγμή που καταθέσει την απαραίτητη προκαταβολή διδάκτρων, αναβαθμίζεται στο σύστημα (από την γραμματεία) σε φοιτητής. Στην περίπτωση που κάποιος υποψήφιος δεν γίνει δεκτός, ενημερώνεται και πάλι από το σύστημα για την απόρριψή του.

Όταν κάποιος υποψήφιος γίνει δεκτός και τον εγγράψει η γραμματεία πλέον ως φοιτητή, ο λογαριασμός του υποψηφίου παραμένει ενεργός και καταχωρημένος στη βάση δεδομένων. Μπαίνοντας ο υποψήφιος στον προσωπικό του λογαριασμό θα βλέπει κάποιο μήνυμα με το οποίο θα ενημερώνεται πως έχει εγγραφεί από την γραμματεία και θα καλείται να εισέλθει ως φοιτητής, από την αντίστοιχη σελίδα jsp η οποία αφορά τον client φοιτητή, με όνομα χρήστη και κωδικό πρόσβασης που παράγονται αυτόματα το σύστημα

ΡΟΛΟΣ: Υποψήφιος	
Ενέργεια	**Σύντομη Περιγραφή**
• Υποβολή αίτησης	Κατάθεση αίτησης και απαραίτητων δικαιολογητικών στην γραμματεία
• Πρόσβαση στο σύστημα	Ο υποψήφιος αποκτά δικαίωμα πρόσβασης στο σύστημα με username και password, τα οποία εκδίδονται αυτόματα από το σύστημα και παρέχονται στον υποψήφιο από την γραμματεία.
• Αποδοχή/αλλαγή ώρας συνέντευξης	Ο υποψήφιος αποδέχεται ή ζητά αλλαγής της προκαθορισμένης από την γραμματεία ώρα συνέντευξης
• Ενημέρωση αξιολόγησης υποψηφίων	Οι υποψήφιοι ενημερώνονται από το σύστημα για το αποτέλεσμα της αξιολόγησής τους
• Εγγραφή στο Π.Μ.Σ.	Οι υποψήφιοι που έγιναν δεκτοί, εγγράφονται στο Π.Μ.Σ. από την γραμματεία ως μεταπτυχιακοί φοιτητές, εφόσον καταθέσουν την προκαταβολή των διδάκτρων

Πίνακας 4: Ρόλος υποψήφιου

3.2.2 Ο ρόλος της Γραμματείας

Ο ρόλος της γραμματείας είναι ο σηματνικότερος και πιο νευραλγικός ρόλος του συστήματος. Η γραμματεία είναι υπεύθυνη για τις λειτουργίες οι οποίες πρέπει να εκτελεστούν στην διάρκεια της διαδικασίας της επιλογής των υποψηφίων (στα όρια του συστήματος), για την διαχείριση του Προγράμματος Μεταπτυχιακών Σπουδών στο μετέπειτα στάδιο της επιλογής των υποψηφίων και είναι επίσης εκείνη η οποία δημιουργεί τους λογαριασμούς χρηστών (υποψηφίων και φοιτητών), τα μαθήματα, τα προφίλ των καθηγητών και πολλά αλλά τα οποία θα δούμε σε αυτήν την παράγραφο.

Η γραμματεία ξεκινά την δραστηριότητά της από την στιγμή που ο υποψήφιος καταθέτει τον φάκελο υποψηφιότητάς του. Σε αυτό το σημείο, η γραμματεία δημιουργεί στο σύστημα έναν λογαριασμό χρήστη για τον εκάστοτε υποψήφιο. Το σύστημα παράγει αυτόματα ονόματα χρήστη και κωδικούς πρόσβασης σύμφωνα με κάποιο κριτήριο το οποίο θέσαμε. Το κριτήριο αυτό είναι το τρέχον έτος και ένας τυχαίος τετραψήφιος κωδικός που θα δημιουργείται κάθε φορά. Αυτό συμβαίνει για την αποφυγή περιπτώσεων με κοινά ονόματα χρήστη, όπου θα γινόταν πιο πολύπλοκη η όλη διαδικασία. Με την δημιουργία του λογαριασμού υποψηφίου, αυτός έχει τη δυνατότητα όπως προαναφέρθηκε να εισέρχεται στο σύστημα πιστοποιημένα πλέον και να παρακολουθεί την κατάσταση της υποψηφιότητάς του.

Παράλληλα με την δημιουργία των λογαριασμών των υποψηφίων, η γραμματεία ελέγχει το περιεχόμενο των φακέλων. Οι υποψήφιοι των οποίων οι φάκελοι υποψηφιότητας πληρούν τις προϋποθέσεις, ενημερώνονται από το σύστημα (έχοντας κάνει login) πως ο φάκελος τους έγινε δεκτός και είναι πλήρης. Οι υποψήφιοι των οποίων οι υποψηφιότητες δεν ήταν πλήρεις, ενημερώνονται τηλεφωνικώς από τη γραμματεία για την προσκόμιση κάποιων ελλιπών ίσως δικαιολογητικών και προωθούνται και αυτοί στο επόμενο βήμα. Για να έχει αυτή την ενημέρωση ο φοιτητής, περνάει από το σύστημα από διάφορες καταστάσεις (candidate mode). Παραδίδοντας αρχικά την αίτησή του εντάσσεται σε μια κατηγορία που περιλαμβάνει τους υποψήφιους που κατέθεσαν αιτήσεις. Όταν η γραμματεία ελέγξει το φάκελο υποψηφιότητας του υποψηφίου, θα τον προάγει στην κατηγορία με τους υποψήφιους των οποίων οι αιτήσεις έγιναν αποδεκτές.

Σε αυτό το σημείο ξεκινάει η κατάρτιση ενός προγράμματος συνεντεύξεων. Η γραμματεία έχει την δυνατότητα μέσα στο σύστημα να ορίσει ημερομηνία και ώρα διεξαγωγής συνεντεύξεων για την κάθε κατεύθυνση. Στο πρόγραμμα συνεντεύξεων θα συμμετέχουν οι υποψήφιοι των οποίων τα δικαιολογητικά ήταν πλήρη (έστω και ετεροχρονισμένα), εκτός εξαιρετικών περιπτώσεων. Η γραμματεία ορίζει ένα προσωρινό πρόγραμμα συνεντεύξεων και ενημερώνεται κάθε υποψήφιος εισέρχοντας στον προσωπικό του λογαριασμό, για μια προσωρινή ημερομηνία και ώρα συνέντευξης. Ο υποψήφιος όπως προαναφέρθηκε στο ρόλο του, μπορεί είτε να αποδεχτεί την ημερομηνία και ώρα που προτείνεται από την γραμματεία, είτε να ζητήσει αλλαγή της ώρας διεξαγωγής της συνέντευξης. Εάν ζητηθεί αλλαγή ώρας, η γραμματεία βλέπει πόσοι και ποιοι έστειλαν αίτημα αλλαγής ώρας συνέντευξης. Όταν η γραμματεία ορίσει νέα ώρα συνέντευξης για τους υποψήφιους που το ζήτησαν, οι υποψήφιοι αυτοί δεν θα

έχουν τη δυνατότητα για νέο αίτημα αλλαγής ώρας. Ο υποψήφιος θα ενημερωθεί εκ νέου για την νέα ώρα που ορίστηκε. Όταν ολοκληρωθεί η δημιουργία του προγράμματος συνεντεύξεων, η γραμματεία θα το προωθήσει στην αρμόδια επιτροπή συνεντεύξεων.

Ακολουθεί μια σειρά διαδικασιών, στις οποίες δεν συμμετέχει ενεργά η γραμματεία. Είναι η σειρά των επιτροπών να δραστηριοποιηθούν, και θα δούμε παρακάτω τον ρόλο και τις ενέργειές τους οι οποίες έχουν άμεση σχέση με την γραμματεία. Αφού λοιπόν ολοκληρωθούν οι διαδικασίες των επιτροπών, η γραμματεία θα λάβει μια τελική λίστα επιτυχόντων οι οποίοι γίνονται δεκτοί στις πέντε κατευθύνσεις των Προγραμμάτων Μεταπτυχιακών Σπουδών.

Η γραμματεία μέσω του συστήματος, ενημερώνει κατάλληλα και τους επιτυχόντες και τους υποψήφιους οι οποίοι απορρίφθηκαν. Οι υποψήφιοι οι οποίοι έγιναν δεκτοί, εφόσον καταβάλουν το ποσό της προκαταβολής, θα εγγραφούν στο Π.Μ.Σ. από την γραμματεία και θα δημιουργηθεί ο φοιτητικός τους λογαριασμός. Ο ρόλος τους ως υποψήφιοι παύει να υπάρχει και με τα ίδια στοιχεία που έχουν καταχωρηθεί στη βάση δεδομένων, αλλάζει η κατάστασή και ο ρόλος τους σε φοιτητές.

Σε αυτό το σημείο περνάμε στο δεύτερο μέρος του συστήματος, όπου γίνεται ουσιαστικά και η διαχείριση των Προγραμμάτων Μεταπτυχιακών Σπουδών από την γραμματεία. Πριν δούμε όμως τους ρόλους που συμμετέχουν στο δεύτερο ουσιαστικά μέρος του συστήματος, θα περιγράψουμε το ρόλο και τις δραστηριότητες που έχει η γραμματεία στο μετέπειτα της επιλογής υποψηφίων στάδιο και θα δούμε στην επόμενη παράγραφο και έναν ρόλο ο οποίος δεν έχει ενεργή συμμετοχή στο υπό ανάπτυξη σύστημα, αλλά η παρουσία του είναι σημαντική για την ομαλή διεξαγωγή της επιλογής υποψηφίων. Ο ρόλος αυτός είναι ο ρόλος των Επιτροπών.

Όπως προαναφέρθηκε, όταν ο υποψήφιος γίνεται δεκτός σε κάποιο Π.Μ.Σ., ο ρόλος τους ως υποψήφιοι παύει να υπάρχει. Οι επιτυχόντες υποψήφιοι αλλάζουν την ιδιότητα και αποκτούν λογαριασμό φοιτητή. Η γραμματεία, μπαίνοντας στο σύστημα και για να έχει τη δυνατότητα διαχείρισης του portal, θα πρέπει πρώτα να δημιουργήσει κάποιο Πρόγραμμα Μεταπτυχιακών Σπουδών. Από εκεί και μετά, δημιουργεί τις κατευθύνσεις οι οποίες αντιστοιχούν σε κάθε Π.Μ.Σ. Στη συνέχεια μπορεί να δημιουργήσει τα μαθήματα τα οποία αντιστοιχούν σε κάθε κατεύθυνση και να ορίσει σε αυτά το εξάμηνο και τις ώρες διεξαγωγής τους. Αφού οριστεί η πρώτη ημέρα παρακολούθησης του μαθήματος, έχει τη δυνατότητα να προσθέτει την επόμενη ημέρα παρακολούθησης του μαθήματος. Αυτό συμβαίνει, διότι πολλές φορές το πρόγραμμα

διδασκαλίας δεν ακολουθείται πιστά, λόγω διάφορων κωλυμάτων (π.χ. απεργίες, απουσία εκπαιδευτικού κλπ). Με αυτόν τον τρόπο επίσης, θα είναι δυνατή η καταχώρηση των παρουσιών των φοιτητών καθώς επίσης και η προσθήκη της βαθμολογίας. Σύμφωνα με την σημερινή διαδικασία, η γραμματεία παραλαμβάνει από τον εκάστοτε καθηγητή μια λίστα με τα ονόματα και τις υπογραφές των φοιτητών που παρακολούθησαν το μάθημα την κάθε φορά ξεχωριστά. Υπάρχει η δυνατότητα στο σύστημα μας, να καταχωρεί η γραμματεία τις παρουσίες των φοιτητών, ενώ παράλληλα θα είναι ορατός σε παρένθεση ο αριθμός των απουσιών κάθε φοιτητή. Εάν κάποιος έχει περισσότερες από δύο απουσίες τις οποίες δικαιούται με βάση τον ισχύοντα κανονισμό Π.Μ.Σ., ο αριθμός των απουσιών εμφανίζεται με κόκκινο χρώμα. Η βαθμολογία μπορεί να καταχωρηθεί, μόνο εάν έχει πραγματοποιηθεί «κλείσιμο» του εκάστοτε μαθήματος.

Η γραμματεία ως διαχειριστής, έχει ασφαλώς τη δυνατότητα να βλέπει ανά πάσα στιγμή τους εγγεγραμμένους φοιτητές ανά κατεύθυνση και ανά εξάμηνο. Στο σημείο αυτό μπορεί επίσης να βλέπει τα στοιχεία των φοιτητών, τα μαθήματα του καθενός και εκεί θα έχει την επιλογή προώθησής του στο επόμενο εξάμηνο. Αυτό σημαίνει πως ολοκλήρωσε το τρέχον εξάμηνο και καταχωρείται πλέον στο επόμενο. Μία ακόμα πολύ σημαντική δυνατότητα, είναι η αλλαγή βαθμολογίας του κάθε φοιτητή. Στην περίπτωση δηλαδή στην οποία ένας φοιτητής Α, έχει λάβει σε κάποιο μάθημα Χ, βαθμολογία μικρότερη του προβιβάσιμου 5 και επανεξετάζεται στο συγκεκριμένο μάθημα και πετυχαίνει προβιβάσιμη βαθμολογία, η γραμματεία έχει τη δυνατότητα να αλλάξει πλέον την βαθμολογία του.

Μία ακόμα λειτουργία της γραμματείας είναι η παραλαβή αιτημάτων για έκδοση αναλυτικών βαθμολογίων και πιστοποιητικών σπουδών. Οι φοιτητές μέσα από το λογαριασμό τους, εκτός των άλλων θα έχουν τη δυνατότητα να αιτούνται αναλυτικές βαθμολογίες ή πιστοποιητικά σπουδών. Οι αιτήσεις αυτές θα φτάνουν στη γραμματεία σαν XML μηνύματα στη βάση δεδομένων.

Επίσης είναι σωστό και χρήσιμο, να γνωρίζει και η γραμματεία αλλά και οι φοιτητές τον διδάσκοντα κάθε μαθήματος. Για το λόγο αυτό η γραμματεία ως διαχειριστής είναι υπεύθυνη και για την δημιουργία των λογαριασμών των καθηγητών. Έτσι, αντιστοιχεί κάθε καθηγητή στα αντίστοιχα μαθήματα τα οποία αυτός διδάσκει, και αυτό φαίνεται και στην διεπαφή των φοιτητών.

Γενικά είναι πολύ κρίσιμος ο διαχωρισμός του ρόλου και των λειτουργιών της γραμματείας. Είναι μια σειρά από πολύπλοκες προγραμματιστικά διαδικασίες οι οποίες θα αναλυθούν στο κεφάλαιο 4 που αφορά την υλοποίηση του συστήματος.

ΡΟΛΟΣ: Γραμματεία	
Ενέργεια	**Σύντομη Περιγραφή**
Παραλαβή αιτήσεων	Παραλαμβάνει όλες τις αιτήσεις και δημιουργεί τους λογαριασμούς των υποψηφίων
Κατάρτιση προγράμματος συνεντεύξεων	Δημιουργεί ένα πρόγραμμα συνεντεύξεων, το οποίο πιθανώς θα τροποποιηθεί έπειτα από αιτήματα υποψηφίων για αλλαγές στις ώρες των συνεντεύξεων
Παραλαβή αποτελεσμάτων	Η γραμματεία λαμβάνει τα αποτελέσματα της αξιολόγησης και ενημερώνει κατάλληλα τους υποψηφίους
Δημιουργία φοιτητικών λογαριασμών	Δημιουργεί τους φοιτητικούς λογαριασμούς, για τους υποψήφιους που έγιναν δεκτοί και κατέθεσαν την προκαταβολή των διδάκτρων
Δημιουργία φοιτητικών λογαριασμών	Οι υποψήφιοι που έγιναν δεκτοί, εγγράφονται στο Π.Μ.Σ. από την γραμματεία ως μεταπτυχιακοί φοιτητές, εφόσον καταθέσουν την προκαταβολή των διδάκτρων
Κατάρτιση Π.Μ.Σ. – Κατευθύνσεις	Δημιουργεί τα Π.Μ.Σ. και τις κατευθύνσεις που αντιστοιχούν στο καθένα
Προσθήκη μαθημάτων	Προσθέτει και τροποποιεί τα μαθήματα για κάθε κατεύθυνση
Προσθήκη παρουσιών και βαθμολογιών	Εισάγει τις παρουσίες και τις βαθμολογίες των φοιτητών, τις οποίες παραλαμβάνει από τους αρμόδιους διδάσκοντες
Λήψη αιτήσεων	Λαμβάνει από τους φοιτητές, αιτήσεις για έκδοση πιστοποιητικών σπουδών και αναλυτικών βαθμολογιών
Δημιουργία λογαριασμών διδασκόντων	Δημιουργεί τους λογαριασμούς πρόσβασης στο σύστημα για το διδακτικό προσωπικό

Πίνακας 5: Ρόλος γραμματείας

3.2.3 Ο ρόλος των επιτροπών

Ο ρόλος των επιτροπών είναι αφανής όσον αφορά την διαδικτυακή τους διεπαφή στο σύστημα, είναι όμως πολύ σημαντικός για την ορθή λειτουργία της διαδικασίας της επιλογής των υποψηφίων. Υπάρχουν τρείς επιτροπές, η καθεμία με τον δικός της ρόλο. Οι επιτροπές αυτές είναι οι εξής:

- Επιτροπή συνεντεύξεων (ή αξιολόγησης)
- Συντονιστική Επιτροπή Π.Μ.Σ.
- Γενική Συνέλευση Ειδικής Σύνθεσης (Γ.Σ.Ε.Σ.)

Για κάθε κατεύθυνση, ορίζεται διαφορετική επιτροπή συνεντεύξεων από τις επιτροπές Π.Μ.Σ. των δύο Προγραμμάτων Σπουδών. Επίσης για κάθε κατεύθυνση διεξάγεται διαφορετική συνέντευξη. Οι επιτροπές συνεντεύξεων είναι υπεύθυνες για την αξιολόγηση των συνεντευξιαζόμενων υποψηφίων. Τα αποτελέσματα της αξιολόγησης των συνεντεύξεων της κάθε κατεύθυνσης λαμβάνονται από τις συντονιστικές επιτροπές Π.Μ.Σ.

Οι συντονιστικές επιτροπές Π.Μ.Σ. αποτελούνται από τρία ή τέσσερα μέλη ΔΕΠ, τα οποία ορίζονται από την Γενική Συνέλευση του Τμήματος. Τα μέλη των συντονιστικών επιτροπών Π.Μ.Σ. επιβλέπουν με τη σειρά τους τα αποτελέσματα της αξιολόγησης από τις επιτροπές συνεντεύξεων. Τέλος, τα αποτελέσματα αξιολογούνται από την Γενική Συνέλευση Ειδικής Σύνθεσης (Γ.Σ.Ε.Σ.) του Τμήματος όπου και γίνεται η τελική επιλογή των επιτυχόντων υποψηφίων. Τα τελικά αποτελέσματα αποστέλλονται στη γραμματεία η οποία ενημερώνει με τη σειρά της τους υποψήφιους.

ΡΟΛΟΣ: Επιτροπές	
Ενέργεια	**Σύντομη Περιγραφή**
Αξιολόγηση	Η επιτροπή αξιολόγησης πραγματοποιεί τις συνεντεύξεις και αξιολογεί τους υποψήφιους
Επίβλεψη αξιολόγησης	Η συντονιστική επιτροπή ελέγχει τα αποτελέσματα της αξιολόγησης
Έγκριση	Η τελική έγκριση των επιτυχόντων πραγματοποιείται από την Γ.Σ.Ε.Σ.

Πίνακας 6: Ρόλος επιτροπών

3.2.4 Ο ρόλος των φοιτητών

Με την ενημέρωση για την επιτυχία τους και την κατάθεση ενός μέρους των διδάκτρων από τους επιτυχόντες, περνάμε ουσιαστικά στο δεύτερο μέρος του συστήματος. Εδώ οι υποψήφιοι που έγιναν δεκτοί από τα Προγράμματα Μεταπτυχιακών Σπουδών, αποκτούν πλέον την ιδιότητα του φοιτητή. Αυτό γίνεται από την γραμματεία, η οποία εισέρχεται στο προφίλ κάθε οριστικοποιεί την συμμετοχή του στο Πρόγραμμα Μεταπτυχιακών Σπουδών.

Όταν εισέλθει στον λογαριασμό του ο εκάστοτε φοιτητής θα έχει τη δυνατότητα να δει τα εξάμηνα και τα μαθήματα της κατεύθυνσής του, να επεξεργαστεί τα προσωπικά του στοιχεία (εκτός του ονόματος χρήστη, του κωδικού πρόσβασης και του ονοματεπώνυμου του), καθώς επίσης να δει τη βαθμολογία του στα μαθήματα στα οποία έχει καταχωρηθεί.

Όπως προαναφέρθηκε και στον ρόλο της γραμματείας, οι φοιτητές θα έχουν τη δυνατότητα μέσα από το σύστημα που αναπτύξαμε να στέλνουν αιτήσεις για παραλαβή αναλυτικών βαθμολογιών και πιστοποιητικών σπουδών. Κάθε αίτηση η οποία θα διεκπεραιώνεται από τη γραμματεία για λογαριασμό κάθε φοιτητή ξεχωριστά, θα είναι διαθέσιμη σαν προεπισκόπηση για προβολή από τον λογαριασμό του αντίστοιχου φοιτητή. Δυστυχώς δεν είναι δυνατόν να αποφευχθεί η φυσική παρουσία του κάθε ενδιαφερόμενου φοιτητή στα γραφεία της γραμματείας για την παραλαβή του πρωτότυπου της αναλυτικής βαθμολογίας ή του πιστοποιητικού σπουδών, εξαιτίας γραφειοκρατικών κωλυμάτων.

ΡΟΛΟΣ: Φοιτητές	
Ενέργεια	**Σύντομη Περιγραφή**
Είσοδος	Όλοι οι μεταπτυχιακοί φοιτητές, έχουν τη δυνατότητα πιστοποιημένης εισόδου στο σύστημα
Προβολή μαθημάτων και βαθμολογίας	Οι φοιτητές μπορούν να δουν τα μαθήματα της κατεύθυνσής τους ανά εξάμηνο καθώς επίσης και την βαθμολογία τους, για όσα μαθήματα έχει καταχωρηθεί
Επεξεργασία προσωπικών στοιχείων	Δυνατότητα επεξεργασίας προσωπικών στοιχείων όπως διεύθυνση, τηλέφωνο και email
Αποστολή αιτήσεων	Οι φοιτητές μπορούν να στείλουν στην γραμματεία αιτήσεις για πιστοποιητικά σπουδών και αναλυτικές βαθμολογίες

Πίνακας 7: Ρόλος φοιτητών

3.2.5 Ο ρόλος του διδακτικού προσωπικού

Όπως σε κάθε σύστημα που αφορά τον τομέα της εκπαίδευσης, έτσι και στο δικό μας, συμμετέχουν και οι καθηγητές. Οι λειτουργίες που μπορούν να εκτελέσουν μέσα στο σύστημά μας, είναι σχετικά απλές αλλά αυτό συμβαίνει αφενός διότι σκοπός της εργασίας δεν είναι η δημιουργία portal ηλεκτρονικής μάθησης και αφετέρου διότι υπάρχουν ήδη λογισμικά ηλεκτρονικής μάθησης ανοιχτού κώδικα.

Στο δικό μας σύστημα λοιπόν, ο κάθε καθηγητής αποκτά το δικό του προφίλ. Εκεί θα μπορεί να βλέπει τα προσωπικά του στοιχεία και να επεξεργαστεί ορισμένα μη «ζωτικά» στοιχεία, όπως η διεύθυνση και το τηλέφωνό του. Επίσης, επιλέγοντας κάποιο μάθημα που του αντιστοιχεί θα έχει τη δυνατότητα να δει τους φοιτητές ανά μάθημα και ανά εξάμηνο, καθώς επίσης και τις παρουσίες και απουσίες αυτών. Μία ακόμα δυνατότητα που παρέχει το σύστημα στους διδάσκοντες, είναι η προβολή των βαθμολογιών του κάθε φοιτητή, φυσικά πάντα ταξινομημένα ανά μάθημα και ανά εξάμηνο.

Εδώ να σημειωθεί, πως είναι δυστυχώς αδύνατο να αποφευχθεί η παραδοσιακή μέθοδος αποστολής των παρουσιών από τους διδάσκοντες στην γραμματεία και

αυτό διότι για την πιστοποίηση της παρουσίας κάποιου φοιτητή είναι απαραίτητη η υπογραφή του. Έτσι ο διδάσκοντας πρέπει να παραδώσει στην γραμματεία το παρουσιολόγιο με τις υπογραφές των φοιτητών που προσήλθαν στο εκάστοτε μάθημα και αυτή με τη σειρά της όπως προαναφέρθηκε να καταχωρήσει τις παρουσίες.

Για λόγους λειτουργικότητας και σαφούς διαχωρισμού των ρόλων αλλά και λόγω του ότι αυτό και σήμερα, η βαθμολογία δεν καταχωρείται στο σύστημα από τους καθηγητές αλλά από την γραμματεία. Φυσικά οι διδάσκοντες είναι αυτοί που θα βαθμολογήσουν τους φοιτητές, αλλά για την καταχώρηση της βαθμολογίας στο σύστημα υπεύθυνη είναι η γραμματεία. Ο καθηγητής φυσικά, όπως προαναφέρθηκε, θα μπορεί να δει τις βαθμολογίες τις οποίες έχει καταχωρήσει η γραμματεία, και σε περίπτωση κάποιου λάθους θα μπορεί να παρέμβει και να επικοινωνήσει με την γραμματεία για την ορθότητα της καταχώρησης.

Για να εξασφαλισθεί η ορθή λειτουργία του συστήματος, θα ήταν καλό να υπάρχει μια συνέπεια όσον αφορά τις ενέργειες των διδασκόντων. Είναι δηλαδή επιθυμητό οι λειτουργίες των βαθμολογιών και των παρουσιών να εκτελούνται έγκαιρα. Εάν για παράδειγμα αργήσει κάποιος διδάσκοντας να στείλει τις βαθμολογίες του για κάποιο μάθημα, θα μείνει εκκρεμότητα από την γραμματεία η καταχώρηση των βαθμών, και παράλληλα το «κλείσιμο» του μαθήματος. Γενικά θα επικρατήσει μια σύγχυση η οποία μπορεί να επηρεάσει αρνητικά την λειτουργία του συστήματος.

ΡΟΛΟΣ: Διδακτικό Προσωπικό	
Ενέργεια	**Σύντομη Περιγραφή**
• Είσοδος	Όλοι οι διδάσκοντες, έχουν τη δυνατότητα πιστοποιημένης εισόδου στο σύστημα
• Προβολή μαθημάτων	Ο κάθε διδάσκοντας έχει τη δυνατότητα να βλέπει τα μαθήματα τα οποία διδάσκει και τους φοιτητές που συμμετέχουν σε αυτά
• Επεξεργασία προσωπικών στοιχείων	Δυνατότητα επεξεργασίας προσωπικών στοιχείων όπως διεύθυνση, τηλέφωνο και email

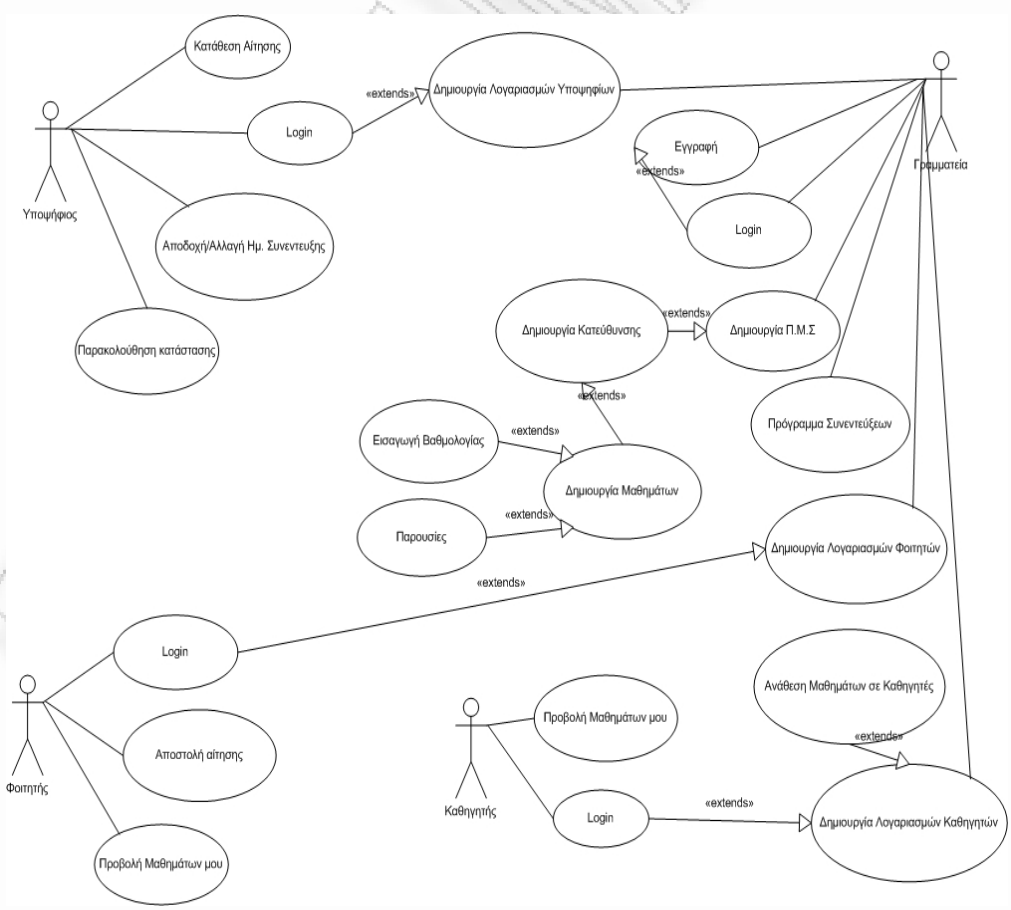
Πίνακας 8: Ρόλος διδακτικού προσωπικού

3.2.6 Διάγραμμα περιπτώσεων χρήσης

Στις παραπάνω παραγράφους αναλύθηκαν οι ρόλοι και οι δραστηριότητες τις οποίες θα έχει τη δυνατότητα να εκτελεί κάθε ρόλος μέσα στο σύστημα με μια σειρά. Αυτές οι λειτουργίες και ρόλοι προήλθαν από τη μελέτη της ροής δραστηριοτήτων της γραμματείας και του Προγράμματος Μεταπτυχιακών Σπουδών γενικά.

Αφού μελετήθηκε και καταγράφηκε η ροή δραστηριοτήτων της γραμματείας και ο τρόπος λειτουργίας του Προγράμματος Μεταπτυχιακών Σπουδών του Τμήματος, προχωρήσαμε στον σχεδιασμό ενός διαγράμματος περιπτώσεων χρήσης, στο οποίο απεικονίζεται ***τι*** μπορεί να κάνει και να χειριστεί ο κάθε χρήστης-ρόλος του συστήματος.

Με τον τρόπο αυτό απεικονίζεται το σύνολο της λειτουργικότητας του συστήματος και όχι όλες τις λεπτομέρειες που απαρτίζουν το σύστημα. Όπως και για το παραπάνω διάγραμμα ροής δραστηριοτήτων, έτσι για το διάγραμμα περιπτώσεων χρήσης χρησιμοποιήθηκε το πρόγραμμα Microsoft Visio.



Εικόνα 27: Διάγραμμα Περιπτώσεων Χρήσης

Ίσως κανείς να παρατηρήσει πως απουσιάζει από διάγραμμα περιπτώσεων χρήσης ο ρόλος των επιτροπών. Όπως αναφέρθηκε και στην παράγραφο 3.2.3, οι επιτροπές δεν έχουν ενεργό ρόλο στην πραγματική λειτουργία του συστήματος. Είναι όμως βασικός ρόλος για την ομαλή διεξαγωγή της διαδικασίας επιλογής των υποψηφίων του μεταπτυχιακού προγράμματος. Για λόγους όμως λειτουργικότητας κρίθηκε μη αναγκαία η ενεργή τους συμμετοχή στο σύστημα.

Έτσι η υλοποίηση του web service θα μας δίνει ένα interface στο οποίο υλοποιούνται μέθοδοι, κλήσεις και λειτουργίες που αφορούν τους ενεργούς ρόλους του συστήματός μας. Δηλαδή τους υποψήφιους, την γραμματεία, τους φοιτητές και τους καθηγητές.

Φαίνεται ξεκάθαρα και στο διάγραμμα περιπτώσεων χρήσης πως για την πρόσβαση κάθε χρήστη στο σύστημα απαιτείται ασφαλής είσοδος με όνομα χρήστη και κωδικό πρόσβασης. Οι υποψήφιοι, οι φοιτητές και οι καθηγητές θα λαμβάνουν προκαθορισμένα από την γραμματεία τα στοιχεία για την πρόσβασή τους. Οι υπεύθυνοι της γραμματείας είναι και οι μόνοι οι οποίοι μπορούν να δημιουργήσουν μόνοι τους τον λογαριασμό τους.

Άλλωστε φαίνεται και στο διάγραμμα πως στην ενέργεια «Login» των υποψηφίων, των φοιτητών και των καθηγητών, υπάρχει η σχέση «extends» με την ενέργεια «Δημιουργία Λογαριασμών Υποψηφίων», «Δημιουργία Λογαριασμών Φοιτητών» και «Δημιουργία Λογαριασμών Καθηγητών» αντίστοιχα. Αυτό σημαίνει πως για να πραγματοποιήσει ο εκάστοτε υποψήφιος, φοιτητής ή καθηγητής Login στο σύστημα, είναι απαραίτητη προϋπόθεση η δημιουργία του αντίστοιχου λογαριασμού από την γραμματεία.

Από κει και πέρα, απεικονίζονται διάφορα «σενάρια» τα οποία είναι ουσιαστικά οι ενέργειες που μπορεί να πραγματοποιήσει ο κάθε χρήστης, ανάλογα με το ρόλο του. Κάθε ενέργεια, αφορά και κάποιο αντίστοιχο βήμα στην ροή δραστηριοτήτων. Το ενδιαφέρον βέβαια επικεντρώνεται στον ρόλο της γραμματείας, η οποία έχει και την διαχείριση του Π.Μ.Σ. Η διαχείριση αυτή, πέρα από ότι αφορά τους χρήστες, τους λογαριασμούς τους κλπ, απαιτεί για παράδειγμα την δημιουργία κάποιου Προγράμματος Μεταπτυχιακών Σπουδών και φυσικά την δημιουργία των κατευθύνσεων που αντιστοιχούν στο κάθε Πρόγραμμα. Φαίνεται και στο διάγραμμα πως για την δημιουργία κατεύθυνσης, είναι απαραίτητη η δημιουργία πρώτα κάποιου προγράμματος.

Γενικά στην εφαρμογή, πραγματοποιήθηκαν όσο το δυνατόν περισσότεροι έλεγχοι ώστε να εξασφαλισθεί η αξιοπιστία και η ορθή και ρεαλιστική λειτουργία και του συστήματός. Στο επόμενο κεφάλαιο, αναλύεται ο τρόπος υλοποίησης όλης της εφαρμογής και παρουσιάζεται η λειτουργικότητα και ο τρόπος λειτουργίας της.

ΚΕΦΑΛΑΙΟ 4: ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ

Στο κεφάλαιο αυτό, αναφέρονται τα εργαλεία που χρησιμοποιήθηκαν και βοήθησαν στην υλοποίηση της εφαρμογής. Αναλύεται ο σχεδιασμός και η ανάπτυξη της βάσης δεδομένων, οι πίνακες που την αποτελούν και οι μεταξύ τους συσχετίσεις. Τέλος παρουσιάζεται ο τρόπος δημιουργίας των project server και client καθώς και το δικτυακό interface της εφαρμογής μέσα από το οποίο πραγματοποιούνται όλες οι λειτουργίες των συμμετεχόντων στο σύστημα.

4.1. Πλατφόρμες και Προγραμματιστικά εργαλεία

Ως λειτουργικό σύστημα για την ανάπτυξη και τον έλεγχο της εφαρμογής χρησιμοποιήθηκαν τα Microsoft Windows XP Professional SP3. Αυτό φυσικά δεν αποτελεί πρόβλημα για την εγκατάσταση και χρήση της εφαρμογής από οποιοδήποτε άλλο διαδεδομένο λειτουργικό σύστημα. Αυτό είναι άλλωστε και το μεγαλύτερο πλεονέκτημα όπως προαναφέρθηκε των web services.

Η ανάπτυξη της εφαρμογής πραγματοποιήθηκε με την χρήση του εργαλείου NetBeans IDE 6.7.1, το οποίο παρέχει ένα πλήρες περιβάλλον ανάπτυξης εφαρμογών (IDE: Integrated Development Environment) και δίνει στο προγραμματιστή βοηθήματα τα οποία κάνουν ευκολότερη και αποδοτικότερη την ανάπτυξη των εφαρμογών. Το εργαλείο NetBeans, παρέχει την δυνατότητα για ανάπτυξη είτε ανεξάρτητων εφαρμογών είτε web εφαρμογών client-server. Για την web εφαρμογή η οποία παρουσιάζεται στην παρούσα εργασία χρησιμοποιήθηκε ο server Apache Tomcat 6.0.18.

Το πιο σημαντικό εργαλείο όμως το οποίο χρησιμοποιήθηκε για την σωστή λειτουργία, διαχείριση και έλεγχο της εφαρμογής είναι μια εξαιρετικά διαδεδομένη για web εφαρμογές βάση δεδομένων, η MySQL Administrator 1.2.17 και MySQL Query Browser 1.2.17. Ένα πολύ σημαντικό πλεονέκτημα της MySQL είναι ότι δεν είναι ιδιαίτερα απαιτητική σε πόρους (μνήμη, επεξεργαστική ισχύς, χώρος αποθήκευσης) και γι' αυτό το λόγο χρησιμοποιήθηκε κατά τη διαδικασία ανάπτυξης του συστήματος. Στην επόμενη παράγραφο αναλύεται η αρχιτεκτονική και ο σχεδιασμός της βάσης δεδομένων που αναπτύχθηκε στην παρούσα εφαρμογή. Τέλος για την ανάπτυξη του client interface του web service, αναπτύχθηκαν Java Server Pages (JSP) σελίδες με συνδυασμό στοιχείων HTML και CSS. Η τεχνολογία των Java Server

Pages επιτρέπει την ανάπτυξη και διατήρηση δυναμικών σελίδων Web και Web-based εφαρμογών χωρίς να υπάρχει αλληλεξάρτηση από την πλατφόρμα. Η JSP τεχνολογία επιτρέπει το διαχωρισμό του User interface από το περιεχόμενο, με αποτέλεσμα να υπάρχει δυνατότητα αλλαγής ολόκληρου του σχεδιασμού της σελίδας χωρίς όμως να μεταβάλλεται το βασικό δυναμικό περιεχόμενό της. Η τεχνολογία Java Server Pages επιτρέπει την δημιουργία Web Content το οποίο έχει τόσο στατικά όσο και δυναμικά components. Τα βασικά χαρακτηριστικά της είναι τα εξής:

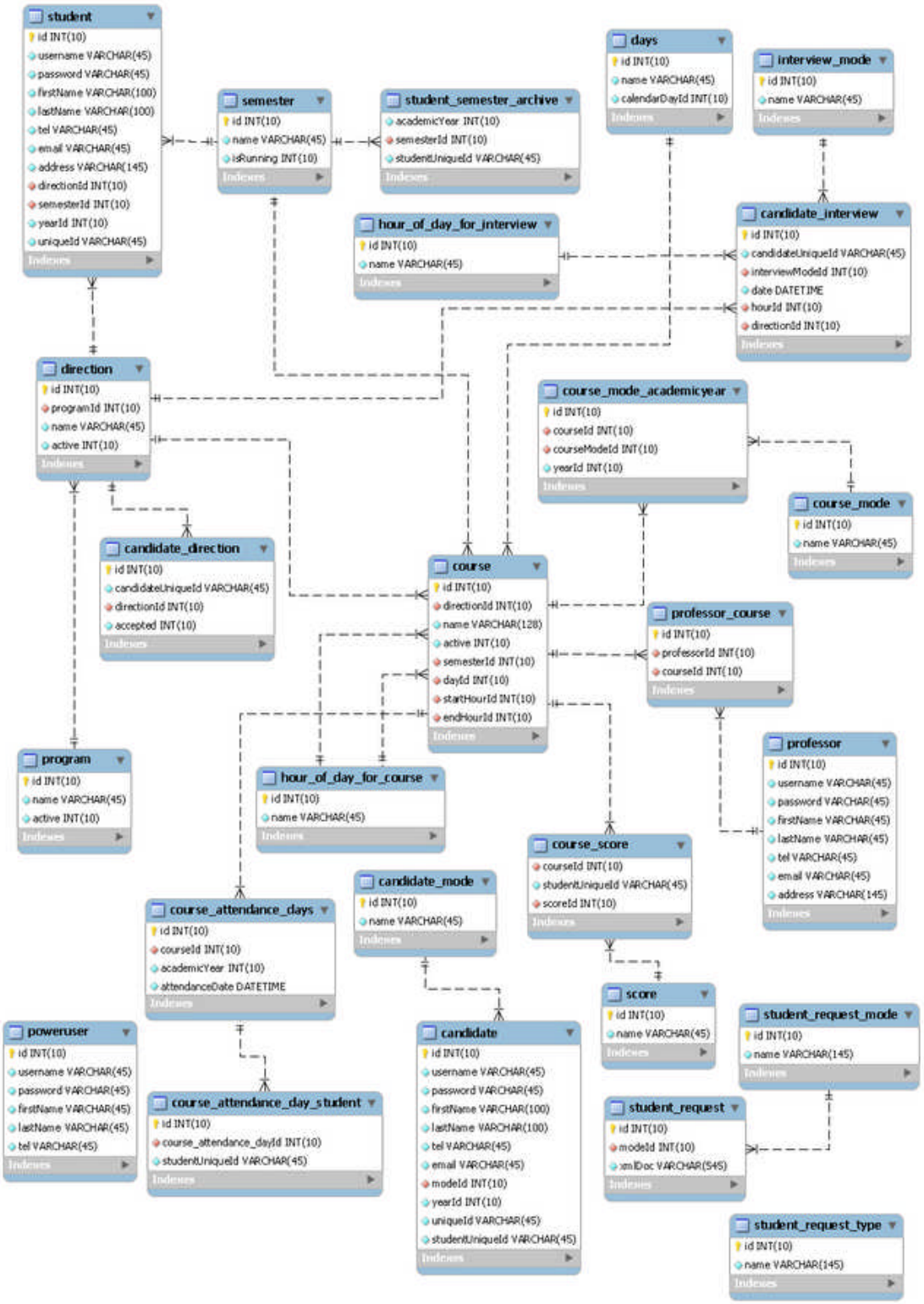
- Περιγράφει τον τρόπο εκτέλεσης των requests στην βάση δεδομένων και τον τρόπο εκτέλεσης των responses σε αυτά τα requests
- Παρέχει μηχανισμούς για τον καθορισμό επεκτάσεων της JSP γλώσσας
- Παρέχει πρόσβαση στα αντικείμενα από την πλευρά του server

Όπως προαναφέρθηκε σε μια JSP σελίδα περιλαμβάνονται και στατικά δεδομένα σε μορφή HTML, WML ή XML κλπ, στοιχεία τα οποία δίνουν τη δυνατότητα στο χρήστη να τροποποιήσει κατ' επιλογή τον τρόπο εμφάνισης του Interface της εφαρμογής του.

4.2. Σχεδιασμός βάσης δεδομένων

Για την ανάπτυξη μιας web εφαρμογής είναι αυτονόητη η ύπαρξη μιας βάσης δεδομένων. Επειδή όλα ξεκινούν από αυτή πρέπει να δοθεί μεγάλη έμφαση και προσοχή στον σχεδιασμό των πινάκων που θα απαρτίζουν την συγκεκριμένη βάση δεδομένων.

Έχοντας λοιπόν καθορίσει από την αρχή τους ρόλους-χρήστες που θα συμμετέχουν στο σύστημα μας, ξεκίνησαν να σχεδιάζονται οι πίνακες οι οποίοι αφορούν αρχικά τους ρόλους αυτούς. Στη συνέχεια και με την πρόοδο του σχεδιασμού του συστήματος προέκυψαν όπως είναι φυσικό και άλλοι πίνακες οι οποίοι αφορούν διάφορες δομές ή καταστάσεις του συστήματος. Η λογική που επικρατεί στο σχεδιασμό είναι απλή και γίνεται μια προσπάθεια για τον συσχετισμό των διάφορων στοιχείων των πινάκων μεταξύ τους. Στο παρακάτω σχήμα, φαίνεται το διάγραμμα E-R της βάσης δεδομένων το οποίο απεικονίζει την αρχιτεκτονική της βάσης, τους πίνακες που την αποτελούν καθώς επίσης και τον βαθμός συσχέτισης των πινάκων αυτών μεταξύ τους.



Εικόνα 28: Διάγραμμα E-R βάσης δεδομένων

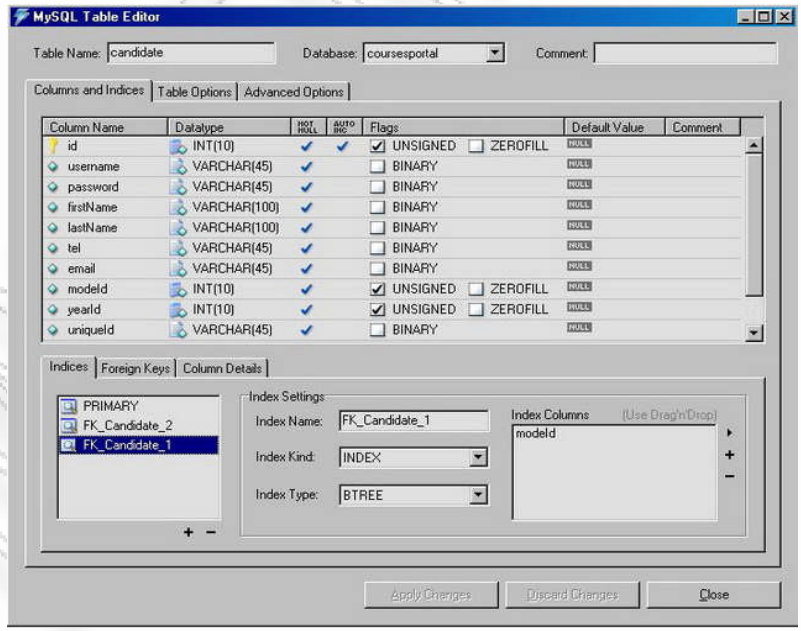
4.2.1 Πίνακες Βάσης Δεδομένων

Οι πίνακες και τα πεδία τους δημιουργήθηκαν στον MySQL Administrator. Με την βοήθεια του MySQL Query Browser προκύπτει και το script δημιουργίας όλων των πινάκων που θα ακολουθήσουν, άρα και συνολικά της βάσης. Λαμβάνοντας το script δημιουργίας των πινάκων, το οποίο ονομάσαμε CreateTables, έχουμε τη δυνατότητα να δημιουργήσουμε σε οποιοδήποτε υπολογιστή που υποστηρίζει βάση δεδομένων MySQL την δική μας βάση απλά κάνοντας ένα execute το script CreateTables. Επίσης, με το script αυτό έχουμε τη δυνατότητα να πραγματοποιήσουμε κάποια Insert με τα οποία θα εισάγουμε κάποιες προκαθορισμένες τιμές όπου είναι απαραίτητο.

Η τελική μορφή την οποία κατέληξε να πάρει η βάση μας, αποτελείται από 26 πίνακες. Κάθε ένας από αυτούς περιέχει κάποια πεδία. Στη συνέχεια θα δούμε αναλυτικά τα πεδία, τις ιδιότητες και την χρήση κάθε πίνακα ξεχωριστά.

Οι πίνακες που αφορούν τους βασικούς clients του συστήματος είναι οι candidate, poweruser, student και professor.

- **Πίνακας candidate**



Εικόνα 29: Πίνακας Candidate

Ο πίνακας αυτός αφορά τους λογαριασμούς πρόσβασης των υποψήφιων. Αποτελείται από τα εξής πεδία:

- **Id** (primary key): αύξον αριθμός – πρωτεύον κλειδί.
- **Username**: όνομα χρήστη το οποίο παράγεται αυτόματα από το σύστημα
- **Password**: κωδικός πρόσβασης ο οποίος παράγεται αυτόματα από το σύστημα
- **firstName**: Όνομα υποψηφίου το οποίο καταχωρεί η γραμματεία κατά το άνοιγμα του λογαριασμού του εκάστοτε υποψηφίου
- **lastName:** Επίθετο υποψηφίου το οποίο καταχωρεί η γραμματεία κατά το άνοιγμα του λογαριασμού του εκάστοτε υποψηφίου
- **tel:** Αριθμός τηλεφώνου του υποψηφίου τον οποίο καταχωρεί η γραμματεία κατά το άνοιγμα του λογαριασμού του υποψηφίου
- **email:** Email του υποψηφίου τον οποίο καταχωρεί η γραμματεία κατά το άνοιγμα του λογαριασμού του υποψηφίου
- **uniqueId:** Μοναδικό id κάθε υποψηφίου, το οποίο είναι το ίδιο κάθε φορά με το username του
- **StudentUniqueId:** Το μοναδικό id το οποίο αποκτά κάθε υποψήφιος όταν εγγραφεί σαν φοιτητής.

o **Foreign Keys**

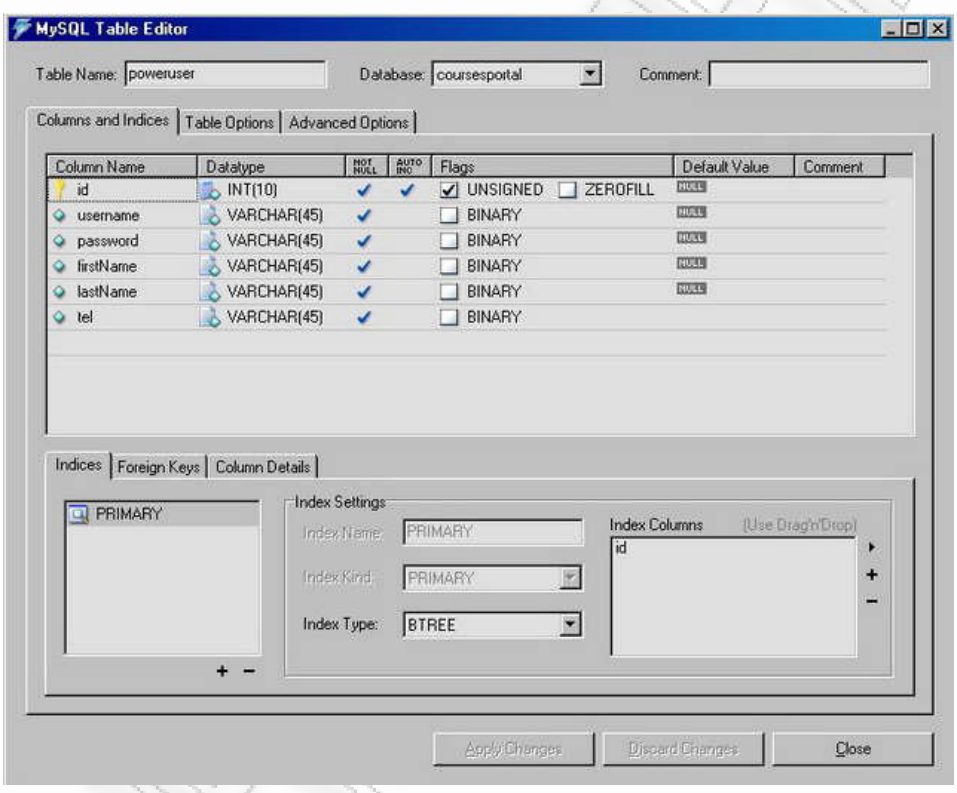
Για την σωστή λειτουργία όμως του συστήματος, θα πρέπει να συσχετίσουμε σε πολλές περιπτώσεις κάποιους πίνακες με κάποια πεδία από κάποιους άλλους πίνακες. Εφόσον θέλουμε να γνωρίζουμε την κατάσταση στην οποία βρίσκεται η αξιολόγηση ενός υποψηφίου, πρέπει να συνδέσουμε τον υποψήφιο με αυτό το κλειδί.

Στο διάγραμμα οντοτήτων συσχετίσεων E-R της βάσης βλέπουμε έναν πίνακα που ονομάζεται **candidate_mode**. Σε αυτόν τον πίνακα δίνουμε id και name. Εισάγουμε δηλαδή για κάθε id και μια τιμή (ονομασία του Mode). Για παράδειγμα το id 0, σημαίνει ότι ο υποψήφιος βρίσκεται σε mode *DELIVERED_DOCUMENTS.* Έτσι, έχοντας το *modeId* κάθε υποψηφίου, μπορούμε να καταλάβουμε σε ποια κατάσταση βρίσκεται η αξιολόγησή του. Η εισαγωγή των id και των καταστάσεων έγινε με τον παρακάτω κώδικα:

```
insert into `candidate_mode` (`id`, `name`) values(0, 'DELIVERED_DOCUMENTS');
insert into `candidate_mode` (`id`, `name`) values(1, 'DOCUMENTS_ACCEPTED');
```

```
insert into `candidate_mode` (`id`, `name`) values(2, 'INTERVIEW_SCHEDULED');
insert into `candidate_mode` (`id`, `name`) values(3, 'INTERVIEW_TO_RESCHEDULE');
insert into `candidate_mode` (`id`, `name`) values(4, 'INTERVIEW_SET');
insert into `candidate_mode` (`id`, `name`) values(5, 'ACCEPTED_BY_GS');
insert into `candidate_mode` (`id`, `name`) values(6, 'REJECTED_BY_GS');
insert into `candidate_mode` (`id`, `name`) values(7, 'HAS_BECOME_STUDENT');
```

- **Πίνακας poweruser**



Εικόνα 30: Πίνακας Poweruser

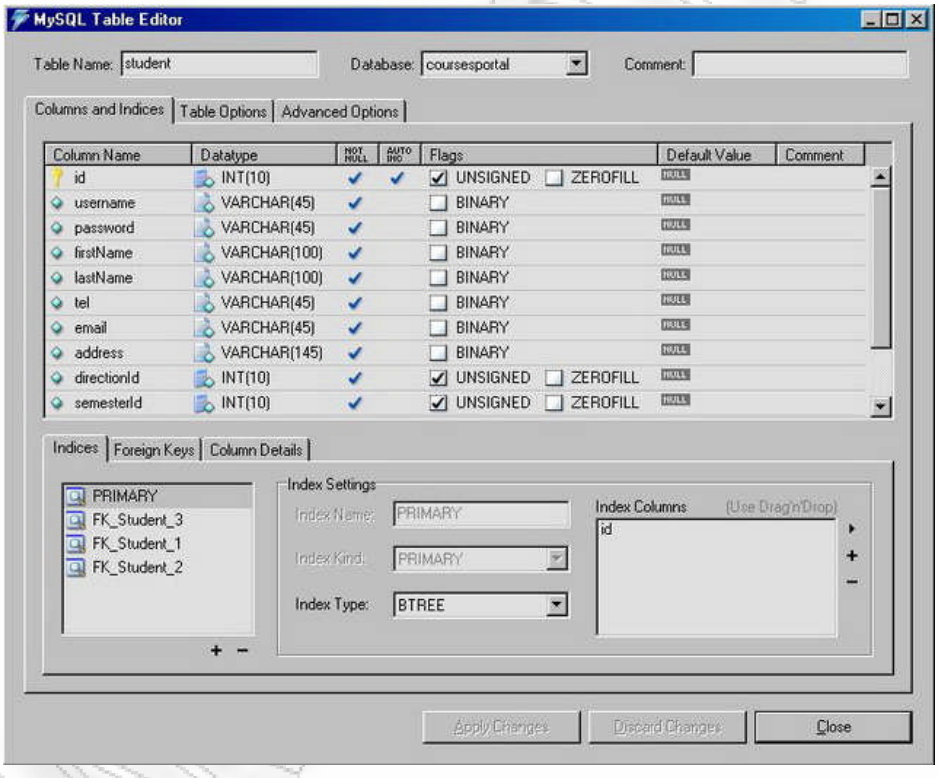
Ο πίνακας αυτός αφορά την δημιουργία λογαριασμού για τους χρήστες της γραμματείας. Γενικά πάντα μέσα στο σύστημα μας, με τον όρο poweruser αναφερόμαστε στον client γραμματεία. Ο πίνακας αποτελείται από τα παρακάτω πεδία:

- **id**(primary key)
- **Username**
- **Password**
- **firstName**

- **lastName**
- **tel**

Τα πεδία αυτά, είναι απαραίτητα για την εγγραφή και δημιουργία ενός λογαριασμού τον οποίο θα χειρίζονται οι αρμόδιοι εργαζόμενοι στην γραμματεία του Τμήματος. Τα στοιχεία που θα δηλώνονται σε κάθε νέα εγγραφή, θα αποθηκεύονται στον πίνακα poweruser στα αντίστοιχα πεδία. Τα πεδία του poweruser δεν συσχετίζονται με κάποιον άλλον πίνακα.

➢ **Πίνακας student**



Εικόνα 31: Πίνακας Student

Αυτός είναι ο πίνακας στον οποίο καταχωρούνται οι εγγεγραμμένοι φοιτητές. Τα στοιχεία username, password παράγονται αυτόματα μόλις γίνει η οριστικοποίηση ενός υποψηφίου σε φοιτητή. Για λόγους ευκολίας και ταχύτητας, τα πεδία firstName και lastName λαμβάνονται από τα αντίστοιχα πεδία στον πίνακα candidate. Συνολικά τα πεδία που αποτελούν τον πίνακα Student είναι:

- **Id** (primary key): αύξον αριθμός – πρωτεύον κλειδί.

- **Username**: όνομα χρήστη το οποίο παράγεται αυτόματα από το σύστημα
- **Password**: κωδικός πρόσβασης ο οποίος παράγεται αυτόματα από το σύστημα
- **firstName**: Όνομα φοιτητή (ίδια εγγραφή από με το firstName του υποψηφίου)
- **lastName:** Επίθετο φοιτητή (ίδια εγγραφή από με το firstName του υποψηφίου)
- **tel:** Αριθμός τηλεφώνου του φοιτητή τον οποίο καταχωρεί η γραμματεία κατά το άνοιγμα του λογαριασμού του φοιτητή
- **email:** Email του υποψηφίου τον οποίο καταχωρεί η γραμματεία κατά το άνοιγμα του λογαριασμού του υποψηφίου
- **address:** Διεύθυνση του φοιτητή
- **uniqueId:** Το μοναδιαίο id του φοιτητή

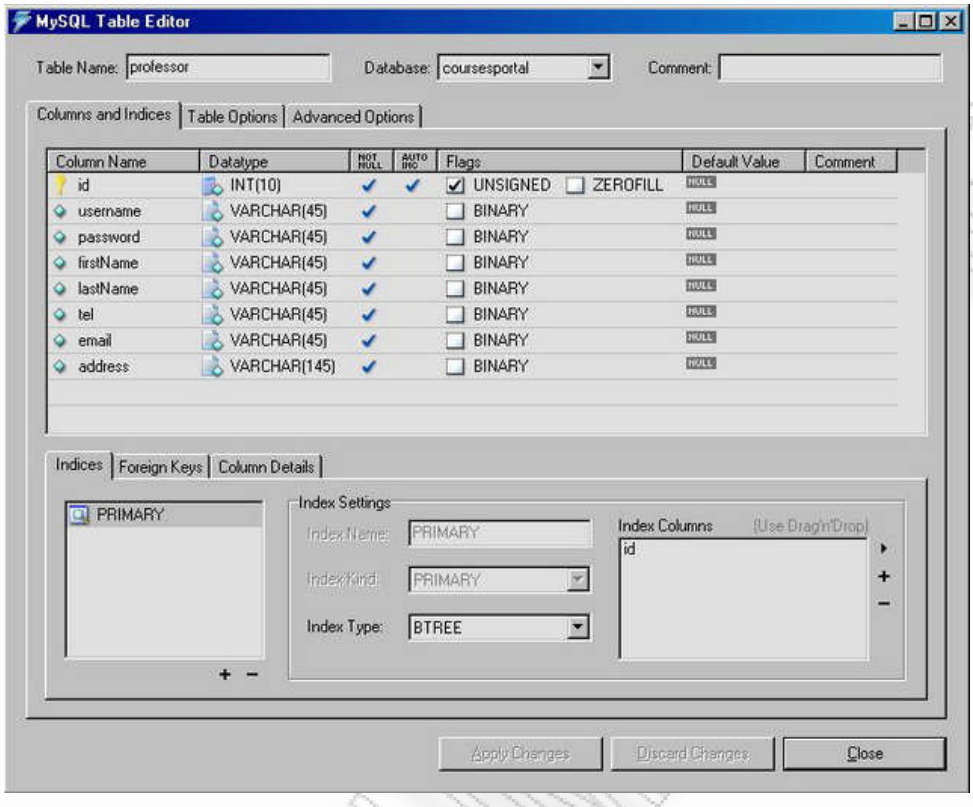
o *Foreign Keys*

Αφού λοιπόν, για έναν student πρέπει να γνωρίζουμε σε ποιο εξάμηνο βρίσκεται, και σε ποια κατεύθυνση εγγράφηκε, ο πίνακας student, συνδέεται με τα εξής πεδία:

- *directionId*: Αναφέρεται στον πίνακα **direction** στο πεδίο id. Παίρνουμε δηλαδή το id της κατεύθυνσης στην οποία εγγράφηκε ο φοιτητής
- *semesterId***:** Αναφέρεται στον πίνακα **semester** στο πεδίο id. Έτσι συνδέεται ο φοιτητής και με τον πίνακα **semester** ωστέ να γνωρίζουμε σε ποιο εξάμηνο βρίσκεται κάθε φοιτητής

```
CONSTRAINT `FK_Student_1` FOREIGN KEY (`directionId`) REFERENCES `direction` (`id`),
CONSTRAINT `FK_Student_2` FOREIGN KEY (`semesterId`) REFERENCES `semester` (`id`)
```

- **Πίνακας professor**



Εικόνα 32: Πίνακας Professor

Ο πίνακας αυτός αφορά τους καθηγητές. Σε αυτόν τον πίνακα αποθηκεύονται τα στοιχεία των λογαριασμών των καθηγητών. Τα στοιχεία αυτά είναι:

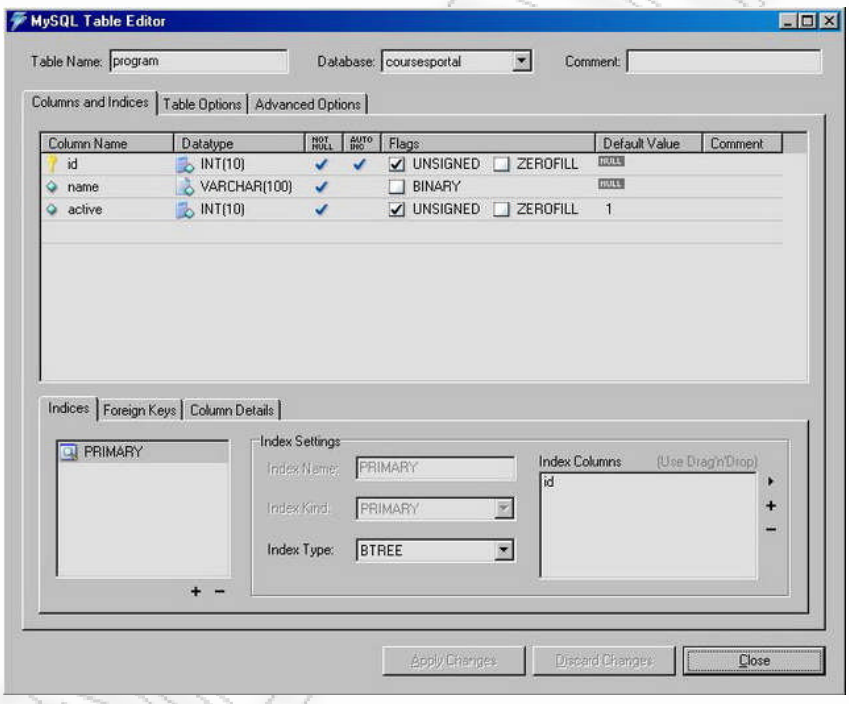
- **Id** (primary key): αύξον αριθμός – πρωτεύον κλειδί.
- **Username**: όνομα χρήστη το οποίο θέτει η γραμματεία κατ' επιλογή της
- **Password**: κωδικός πρόσβασης τον οποίο επίσης ορίζει η γραμματεία για κάθε διδάσκοντα ξεχωριστά.
- **firstName**: Όνομα διδάσκοντα
- **lastName:** Επίθετο διδάσκοντα
- **tel:** Αριθμός τηλεφώνου του διδάσκοντα
- **email:** Email διδάσκοντα
- **address:** Διεύθυνση διδάσκοντα

Οι εκπαιδευτικοί, θα έχουν τη δυνατότητα επεξεργασίας των προσωπικών τους στοιχείων: τηλέφωνο, διεύθυνση και email.

Αυτοί είναι οι πίνακες που αφορούν τους clients του συστήματος μας. Στη συνέχεια θα παρουσιάσουμε τους πίνακες που αφορούν βασικές δομές του συστήματος.

Από τις πλέον βασικές δομές του συστήματός μας είναι τα Προγράμματα Μεταπτυχιακών Σπουδών, οι κατευθύνσεις των προγραμμάτων αυτών καθώς επίσης τα μαθήματα που πραγματοποιούνται σε κάθε κατεύθυνση και τα εξάμηνα στα οποία πραγματοποιούνται τα μαθήματα αυτά. Είναι μια περίπλοκη διαδικασία η οποία απαιτούσε μεγάλη προσοχή στον σχεδιασμό και την συσχέτιση όλων αυτών των πινάκων μέσα στη βάση δεδομένων. Σχεδόν όλοι οι υπόλοιποι πίνακες συνδέονται με αυτές τις δομές.

- **Πίνακας program**



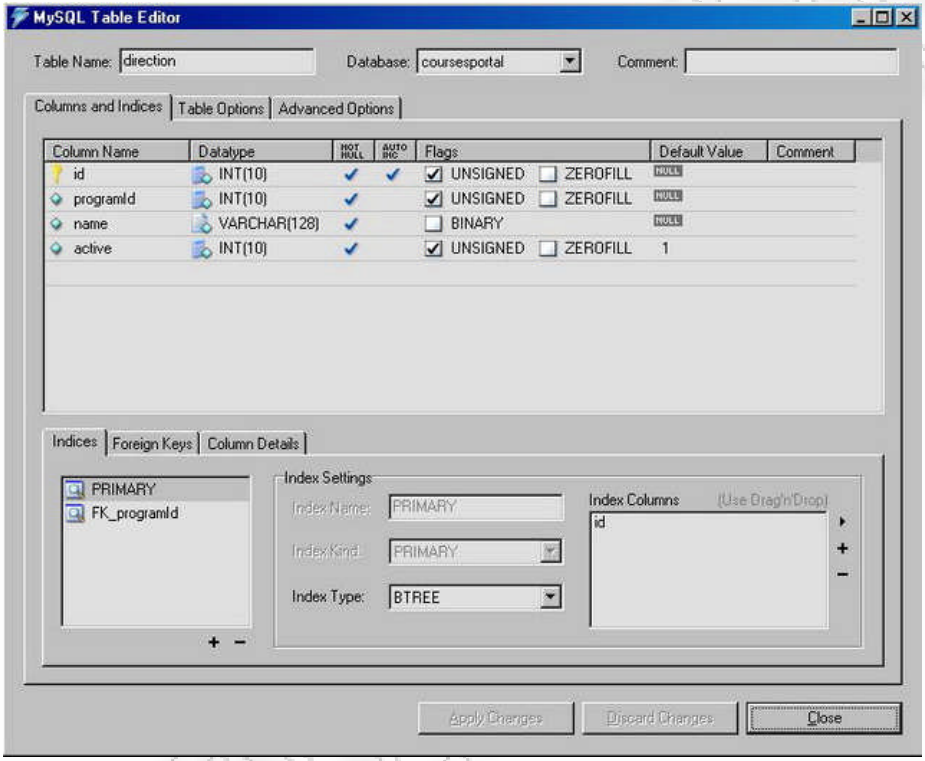
Εικόνα 33: Πίνακας Program

Στον πίνακα αυτό αποθηκεύονται τα Προγράμματα Μεταπτυχιακών Σπουδών τα οποία δημιουργεί ο αρμόδιος χρήστης από την γραμματεία. Τα πεδία του πίνακα όπως βλέπουμε είναι τα :

- **id** (primary key): αύξον αριθμός – πρωτεύον κλειδί.
- **name:** Το όνομα του προγράμματος
- **active:** Εάν η τιμή του πεδίου είναι μηδέν, τότε θεωρείται ανενεργό και δεν εμφανίζεται. Αντί δηλαδή για delete, θέτουμε την τιμή μηδέν και δεν εμφανί-

ζεται το πρόγραμμα. Ουσιαστικά όμως δεν χρησιμοποιείται κάπου το πεδίο αυτό, διότι όλες οι δομές που δημιουργούνται μπορούν να τεθούν σε επεξεργασία και όχι σε διαγραφή από την στιγμή που θα συνδεθούν με διάφορα δεδομένα (πχ Π.Μ.Σ. με κατευθύνσεις, εξάμηνα, φοιτητές κλπ)

- **Πίνακας Direction**

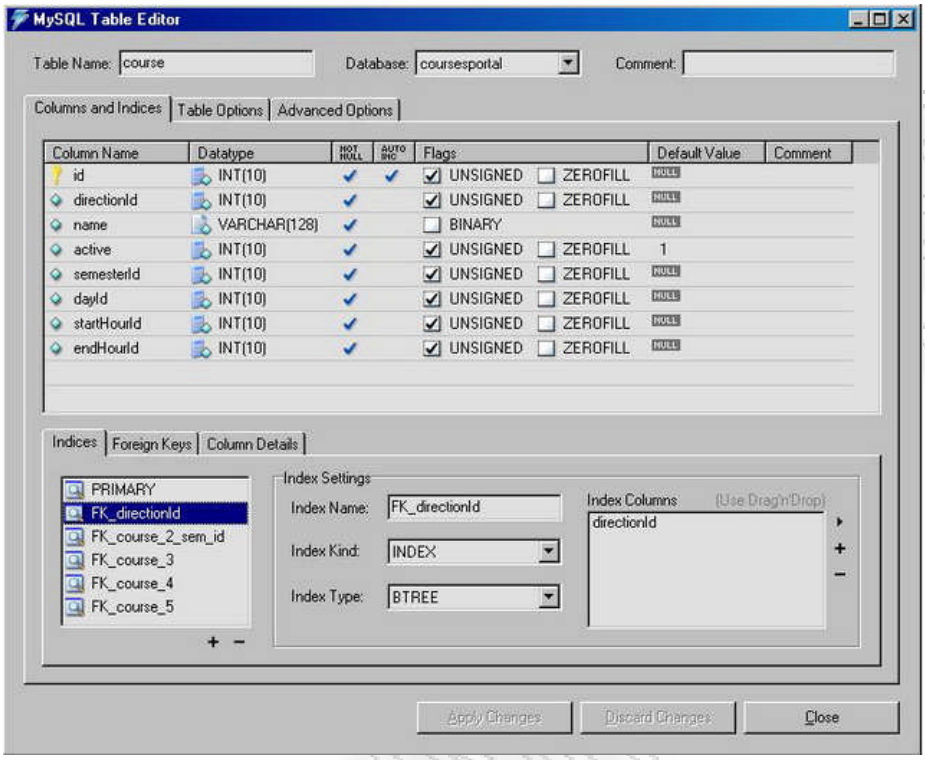


Εικόνα 34: Πίνακας Direction

Ο πίνακας αυτός, χρησιμοποιείται για την αποθήκευση των κατευθύνσεων των προγραμμάτων μεταπτυχιακών σπουδών και περιέχει τα ίδια πεδία με τον πίνακα program, μόνο που εδώ υπάρχει η πολύ σημαντική σύνδεση με το κλειδί *programId* του πίνακα **program**. Έτσι, γνωρίζουμε κάθε φορά σε ποιο πρόγραμμα σπουδών αντιστοιχεί η κάθε κατεύθυνση.

```
CONSTRAINT `FK_programId` FOREIGN KEY (`programId`) REFERENCES `program` (`id`)
```

- **Πίνακας course**



Εικόνα 35: Πίνακας Course

Άλλος ένας πολύ σημαντικός πίνακας. Ο πίνακας Course αναφέρεται στα μαθήματα των μεταπτυχιακών προγραμμάτων σπουδών. Τα πεδία που περιλαμβάνει ο πίνακας αυτός είναι:

- **Id (**primary key): αύξον αριθμός – πρωτεύον κλειδί
- **name:** Όνομα του μαθήματος
- **active:** Εάν η τιμή του πεδίου είναι μηδέν, τότε θεωρείται ανενεργό και δεν εμφανίζεται

o *Foreign Keys*

Για κάθε μάθημα, πρέπει να γνωρίζουμε σε ποια κατεύθυνση ορίζεται (και εφόσον η κατεύθυνση συνδέεται με το πρόγραμμα μεταπτυχιακών σπουδών, γνωρίζουμε και το πρόγραμμα στο οποίο ορίζεται το μάθημα) επομένως πραγματοποιούμε συσχέτιση με το κλειδί *directionId* που αναφέρεται στον πίνακα **direction**. Αμέσως γνωρίζουμε σε ποια κατεύθυνση αντιστοιχεί κάθε μάθημα.

Θέλουμε επίσης να γνωρίζουμε και το εξάμηνο στο οποίο πραγματοποιείται το μάθημα. Επομένως θα συνδέσουμε και το κλειδί *semesterId*, από τον πίνακα **semester**. Επίσης, επειδή θέλουμε να ορίζουμε και ποια μέρα θα πραγματοποιείται το μάθημα, συνδέουμε τον πίνακα courses και με το κλειδί *dayId* του πίνακα **day**.

```
CONSTRAINT `FK_course_2_sem_id` FOREIGN KEY (`semesterId`) REFERENCES `semester` (`id`),
CONSTRAINT `FK_course_3` FOREIGN KEY (`dayId`) REFERENCES `days` (`id`),
CONSTRAINT `FK_directionId` FOREIGN KEY (`directionId`) REFERENCES `direction` (`id`)
```

Τέλος για κάθε μάθημα ορίζεται και η ώρα έναρξης και λήξης παρακολούθησης. Για λόγους ευκολίας, θελήσαμε να θέσουμε μια ακολουθία διάφορων ωρών, οι οποίες να είναι διαθέσιμες για επιλογή από τον χρήστη με ένα drop down menu. Αυτό συμβαίνει στον ορισμό της ώρας έναρξης και λήξης της διδασκαλίας καθώς επίσης και στον ορισμό των συνεντεύξεων των υποψηφίων. Για το λόγο αυτό δημιουργήθηκαν επίσης οι πίνακες **hour_of_day_for_course** και **hour_of_day_for_interview** στους οι οποίους δημιουργήσαμε κάποια Insert και θέσαμε id και name. Κάθε id αντιστοιχεί σε κάποια ώρα. Ενδεικτικά το script με το οποίο ορίζονται οι αντιστοιχίες id και name :

```
insert into `hour_of_day_for_course` (`id`, `name`) values(1,'07:00');
insert into `hour_of_day_for_course` (`id`, `name`) values(2,'07:15');
insert into `hour_of_day_for_course` (`id`, `name`) values(3,'07:30');
insert into `hour_of_day_for_course` (`id`, `name`) values(4,'07:45');
insert into `hour_of_day_for_course` (`id`, `name`) values(5,'08:00');
insert into `hour_of_day_for_course` (`id`, `name`) values(6,'08:15');
insert into `hour_of_day_for_course` (`id`, `name`) values(7,'08:30');
insert into `hour_of_day_for_course` (`id`, `name`) values(8,'08:45');
insert into `hour_of_day_for_course` (`id`, `name`) values(9,'09:00');
insert into `hour_of_day_for_course` (`id`, `name`) values(10,'09:15');
insert into `hour_of_day_for_course` (`id`, `name`) values(11,'09:30');
…… …… …… …… …
…… …… …… …… …
```

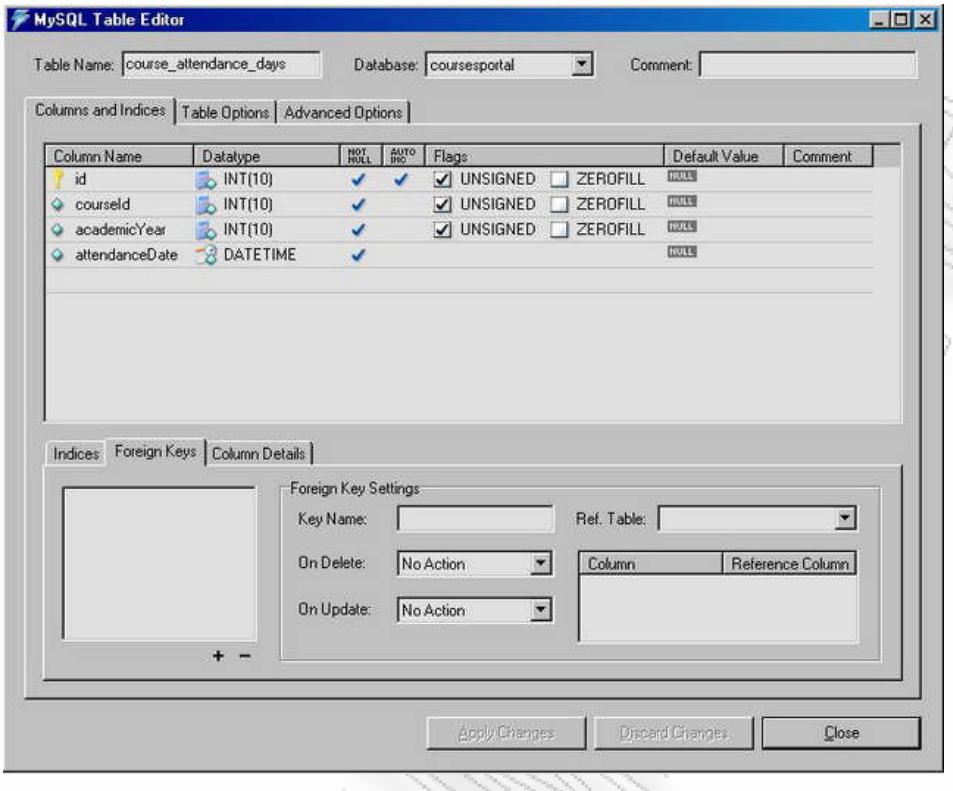
Και αντίστοιχα για τον πίνακα hour_of_day_for_interview:

```
insert into `hour_of_day_for_interview` (`id`, `name`) values(1,'07:00');
insert into `hour_of_day_for_interview` (`id`, `name`) values(2,'07:15');
insert into `hour_of_day_for_interview` (`id`, `name`) values(3,'07:30');
insert into `hour_of_day_for_interview` (`id`, `name`) values(4,'07:45');
insert into `hour_of_day_for_interview` (`id`, `name`) values(5,'08:00');
insert into `hour_of_day_for_interview` (`id`, `name`) values(6,'08:15');
insert into `hour_of_day_for_interview` (`id`, `name`) values(7,'08:30');
insert into `hour_of_day_for_interview` (`id`, `name`) values(8,'08:45');
insert into `hour_of_day_for_interview` (`id`, `name`) values(9,'09:00');
insert into `hour_of_day_for_interview` (`id`, `name`) values(10,'09:15');
insert into `hour_of_day_for_interview` (`id`, `name`) values(11,'09:30');
…… …… …… …… …
…… …… …… …… …
```

Έτσι λοιπόν συνδέουμε τα πεδία τον πίνακα course με τα κλειδιά ***startHourId*** και ***endHourId*** που αναφέρονται στον πίνακα **hour_of_day_for_course**.

```
CONSTRAINT `FK_course_4` FOREIGN KEY (`startHourId`) REFERENCES `hour_of_day_for_course` (`id`),
 CONSTRAINT `FK_course_5` FOREIGN KEY (`endHourId`) REFERENCES `hour_of_day_for_course` (`id`),
```

- **Πίνακας course_attendance_days**



Εικόνα 36: Πίνακας course_attendance_days

Ένας ακόμη πίνακας ο οποίος συσχετίζεται με τον πίνακα course. Ο πίνακας αυτός δημιουργήθηκε ώστε να είναι εφικτή η καταχώρηση των ημερών παρακολούθησης (DATETIME τιμή) που αφορούν κάποιο μάθημα, δηλαδή κάποιο courseId και να συσχετιστεί με το academinYear, δηλαδή με το τρέχον ακαδημαϊκό έτος. Τα πεδία του πίνακα είναι τα εξής:

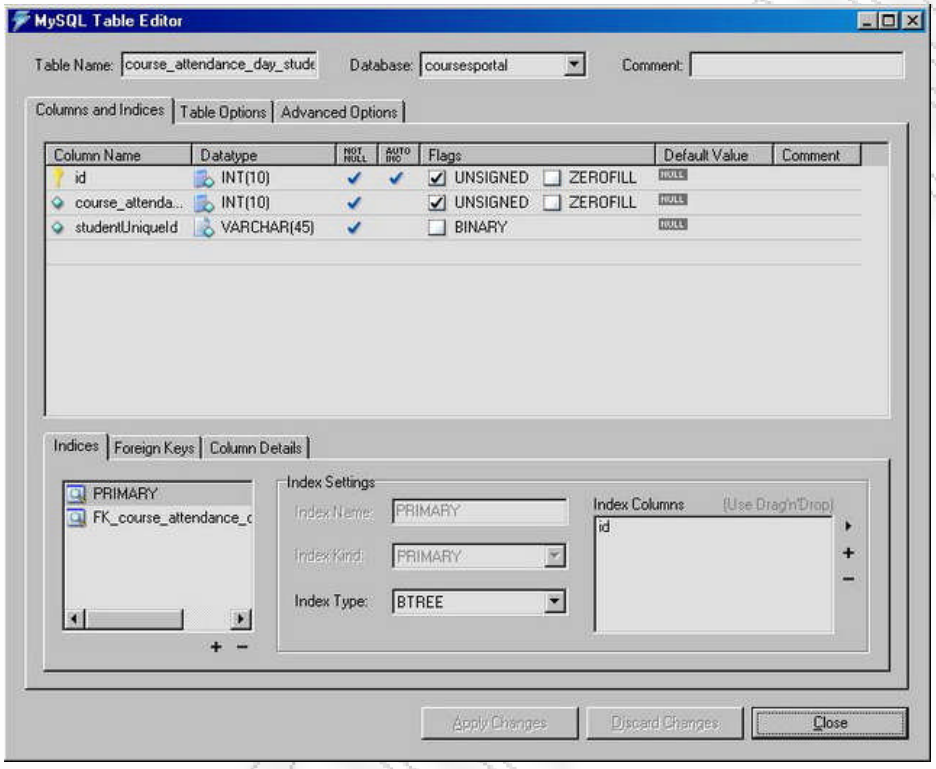
- **id** (primary key): αύξον αριθμός – πρωτεύον κλειδί.
- **academicYear:** Το τρέχον ακαδημαϊκό έτος
- **attendanceDate:** Η ημερομηνία ημέρας παρακολούθησης ενός μαθήματος

o *Foreign Keys*

Φυσικά από την στιγμή που αναφερόμαστε σε μαθήματα, είναι αναμενόμενη η χρήση κάποιου foreign key του πίνακα **course**. Το κλειδί αυτό είναι το id του μαθήματος, δηλαδή το *courseId*. Και αυτό διότι, σε κάθε μάθημα ορίζεται μια μέρα παρακολούθησης.

```
CONSTRAINT `FK_course_attendance_days_1` FOREIGN KEY (`courseId`) REFERENCES `course` (`id`)
```

- **Πίνακας course_attendance_day_student**



Εικόνα 37: Πίνακας course_attendanvce_day_student

Ο πίνακας αυτός, αφορά τις παρουσίες των φοιτητών στα μαθήματα. Αποτελείται από τα εξής πεδία:

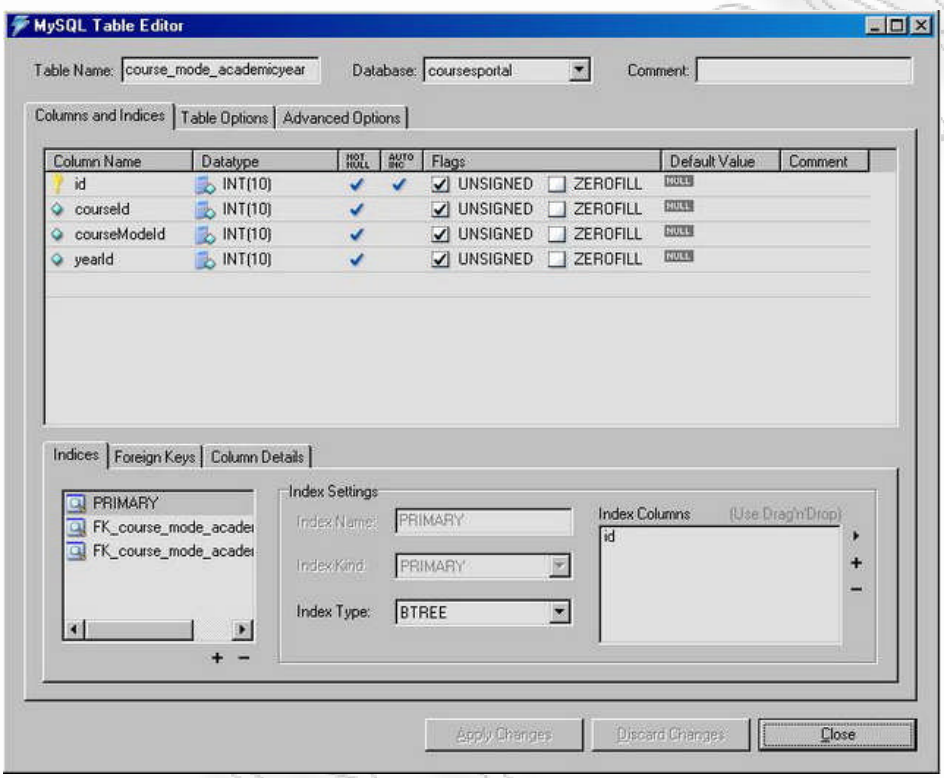
- **id** (primary key): αύξον αριθμός – πρωτεύον κλειδί.
- **studentUniqueId:** Το μοναδικό id κάθε φοιτητή

o *Foreign Keys*

Για να είναι δυνατή η λήψη των παρουσιών πρέπει να γνωρίζουμε ότι ο φοιτητής παρακολούθησε το μάθημα μιας συγκεκριμένης ημερομηνίας. Επομένως συνδέουμε τον πίνακα course_attendance_day_student, με το κλειδί *course_attendance_dayId*, το οποίο αναφέρεται στον πίνακα

course_attendance_days. Τον συνδέουμε δηλαδή έμμεσα και με ημερομηνία και με ακαδημαϊκό έτος γιατί αυτό το Id είναι μοναδικό, δεν είναι απλά μια ημερομηνία.

- **Πίνακας course_mode_academicyear**



Εικόνα 38: Πίνακας course_mode_academicyear

Με την δημιουργία του πίνακα **course_mode** ορίζουμε τα διάφορα στάδια που περνάει ένα μάθημα (course).

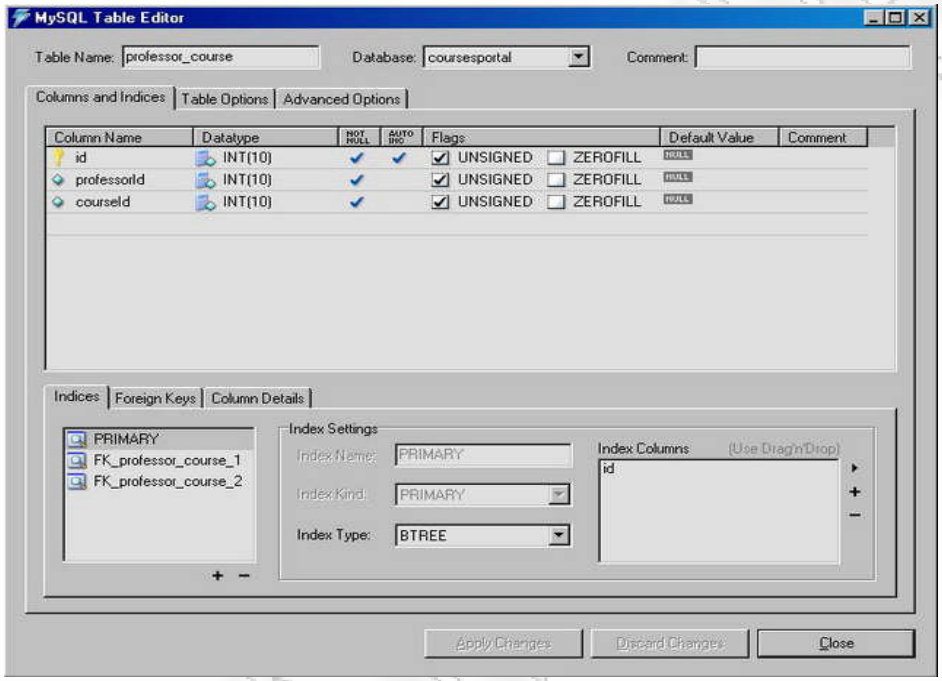
```
insert into `course_mode` (`id`, `name`) values(0, 'STARTED');
insert into `course_mode` (`id`, `name`) values(1, 'FINISHED');
insert into `course_mode` (`id`, `name`) values(2, 'CLOSED');
```

Με τον πίνακα **course_mode_academicyear**, ο οποίος αποτελείται από τα εξής πεδία:

- **id** (primary key): αύξον αριθμός – πρωτεύον κλειδί.
- **yearId**

προσδιορίζουμε ποιο είναι το mode του εκάστοτε μαθήματος για το τρέχον ακαδημαϊκό έτος συσχετίζοντας το κλειδί *courseModeId* που αναφέρεται στον πίνακα **course_mode** και το κλειδί *courseId* που αναφέρεται στον πίνακα **course.**

- **Πίνακας professor_course**



Εικόνα 39: Πίνακας professor_course

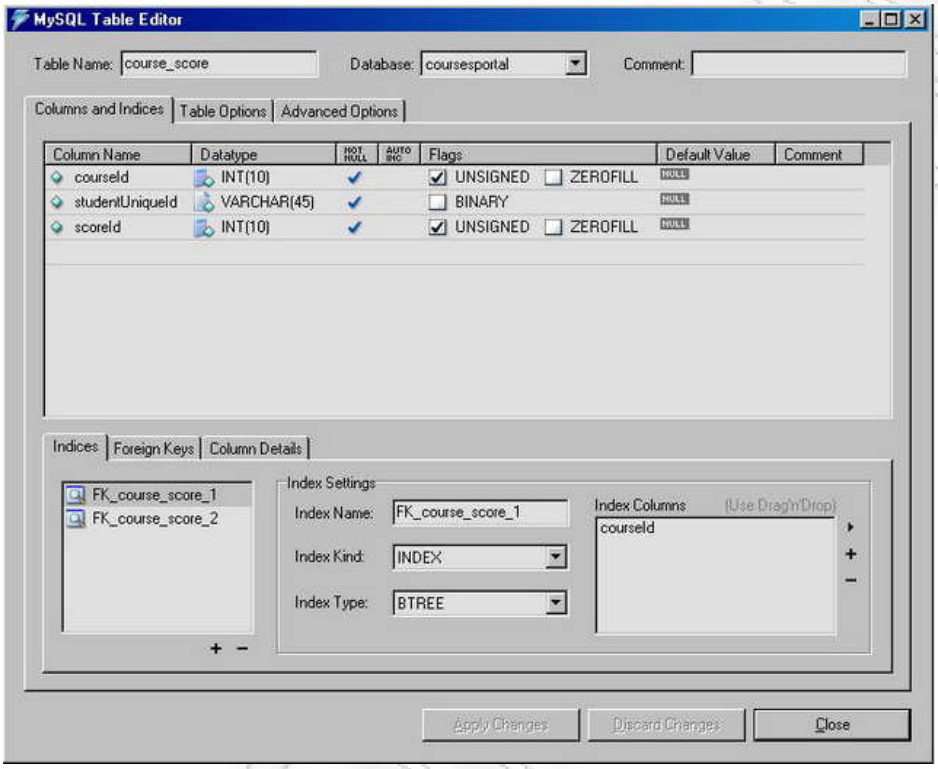
Συνεχίζοντας με πίνακες που συσχετίζονται με τα μαθήματα, συναντάμε τον πίνακα professor_course. Θα μπορούσαμε να πούμε πως είναι ο συνδετικός κρίκος ανάμεσα στους πίνακες course και professor, ώστε να είναι δυνατή η ανάθεση των μαθημάτων στους καθηγητές. Έτσι ο πίνακας professor_course αποτελείται από τα πεδία:

- **id** (primary key): αύξον αριθμός – πρωτεύον κλειδί
- **professorId**: foreign key
- **courseId**: foreign key

Για να είναι δυνατή η σύνδεση του κάθε διδάσκοντα με κάποια μαθήματα, χρησιμοποιούμε τα κλειδιά *professorId* και *courseId* τα οποία αναφέρονται στους πί-

νακες **professor** και **course** αντίστοιχα. Έτσι για παράδειγμα γνωρίζουμε ότι ο διδάσκοντας με id: 3, διδάσκει τα μαθήματα με courseId:6,8,11.

- **Πίνακας course_score**



Εικόνα 40: Πίνακας course_score

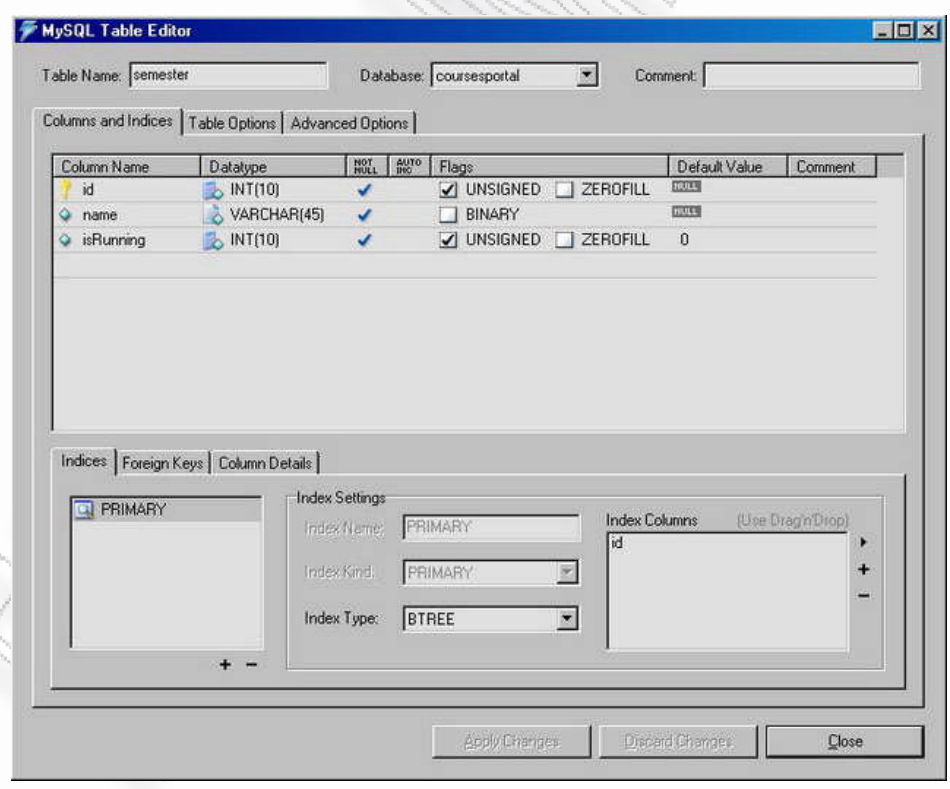
Ο πίνακας αυτός είναι ο συνδετικός κρίκος ανάμεσα στους πίνακες course και score. Αφορά την βαθμολογία κάθε φοιτητή για κάθε μάθημα. Αυτό γίνεται εύκολα αντιληπτό αν δούμε τα πεδία του:

- **courseId**: Το Id του μαθήματος, από τον πίνακα course (foreign key)
- **studentUniqueId**: Το μοναδιαίο Id του φοιτητή
- **scoreId**: Το id από τον πίνακα **score** (foreign key)

Ο πίνακας score, αποτελείται από δύο πεδία id και name. Όπως και σε άλλες περιπτώσεις, έτσι και εδώ το scoreId μας φέρνει σαν αποτέλεσμα κάποια βαθμολογία. Για παράδειγμα, το id: 6, από τον πίνακα score, μας φέρνει σαν αποτέλεσμα την βαθμολογία : 5. Τις τιμές αυτές τις έχουμε προκαθορίσει εμείς με τα κατάλληλα Insert στο script της βάσης:

```
insert into `score` (`id`, `name`) values(0, '-');
insert into `score` (`id`, `name`) values(1, '0');
insert into `score` (`id`, `name`) values(2, '1');
insert into `score` (`id`, `name`) values(3, '2');
insert into `score` (`id`, `name`) values(4, '3');
insert into `score` (`id`, `name`) values(5, '4');
insert into `score` (`id`, `name`) values(6, '5');
insert into `score` (`id`, `name`) values(7, '6');
insert into `score` (`id`, `name`) values(8, '7');
insert into `score` (`id`, `name`) values(9, '8');
insert into `score` (`id`, `name`) values(10, '9');
insert into `score` (`id`, `name`) values(11, '10');
```

- **Πίνακας semester**



Εικόνα 41: Πίνακας semester

Πολύ σημαντική οντότητα στη λειτουργία του συστήματος αποτελούν τα εξάμηνα. Το εξάμηνο στο οποίο διδάσκεται κάποιο μάθημα αλλά και το εξάμηνο στο οποίο βρίσκεται κάποιος φοιτητής είναι κάτι που πρέπει να αποθηκεύεται. Έτσι ο πί-

νακας semester, δημιουργήθηκε για να μας δίνει κάποιες προκαθορισμένες τιμές διότι γνωρίζουμε ότι το μέγιστο χρονικό διάστημα σπουδών στα μεταπτυχιακά του τμήματος είναι τα 4 εξάμηνα.

Έτσι, όταν η γραμματεία θέλει για παράδειγμα να προσθέσει ένα μάθημα σε κάποια κατεύθυνση, θα μπορεί να επιλέγει από ένα drop down menu το εξάμηνο στο οποίο θα διδάσκεται το συγκεκριμένο μάθημα. Στο μενού αυτό θα εμφανίζονται τέσσερα εξάμηνα. Ο πίνακας αυτός δημιουργήθηκε και «γέμισε» με δεδομένα που του δώσαμε με τα κατάλληλα Insert στο script της βάσης:

```
insert into `semester` (`id`, `name`, `isRunning`) values(0, 'Εξάμηνο 1', 1);
insert into `semester` (`id`, `name`, `isRunning`) values(1, 'Εξάμηνο 2', 1);
insert into `semester` (`id`, `name`, `isRunning`) values(2, 'Εξάμηνο 3', 1);
insert into `semester` (`id`, `name`, `isRunning`) values(3, 'Εξάμηνο 4', 1);
```

Αποτελείται από τα παρακάτω πεδία:

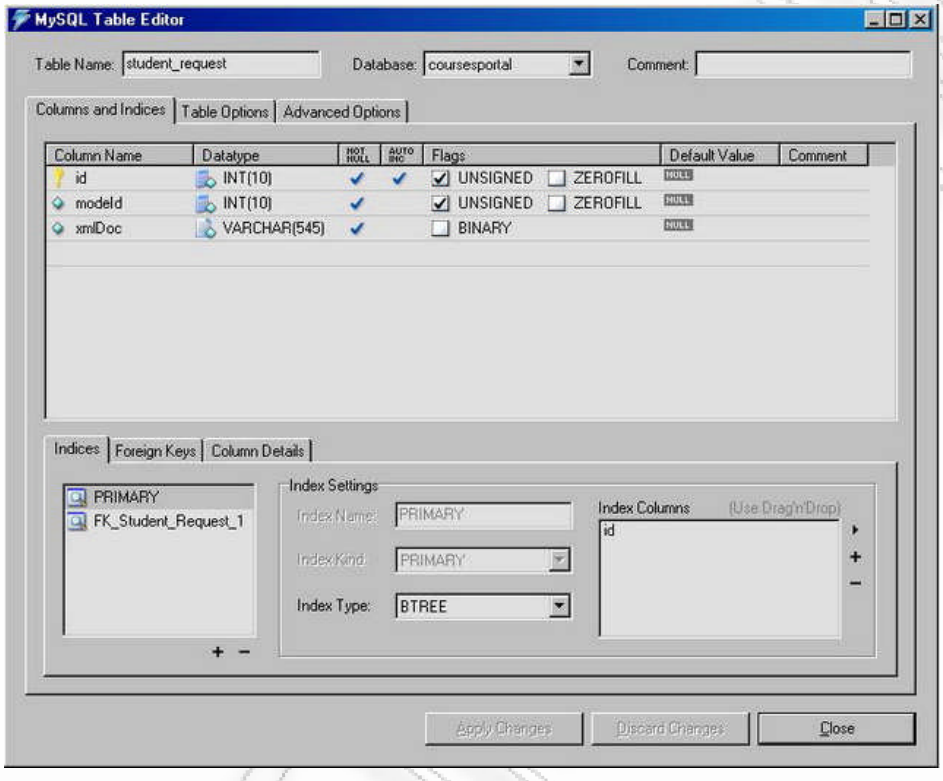
- **id**: Το id του εξαμήνου
- **name**: Το όνομα που αντιστοιχεί σε κάθε id
- **isRunning**: Χρησιμοποιήθηκε κατά το αρχικό στάδιο της υλοποίησης, για να ορίζεται αν το εξάμηνο είναι τρέχον. Πλέον αυτό καθορίζεται από την χρονολογία.

Ο πίνακας semester συσχετίζεται όπως προαναφέρθηκε και με τον πίνακα **student** ώστε να είναι δυνατή η συσχέτιση ενός φοιτητή με κάποιο εξάμηνο, αλλά και με τον πίνακα **student_semester_archive**, με την βοήθεια του οποίου καταχωρούνται οι φοιτητές οι οποίοι περνάνε στο επόμενο εξάμηνο. Δηλαδή αν ένας φοιτητής έχει περάσει στο δεύτερο εξάμηνο, μπορούμε να καταλάβουμε ότι έχει περάσει από το πρώτο και έτσι μπορούμε ελέγξουμε για καταχωρήσεις για παρακολούθηση και βαθμολογία και αυτό γιατί ο συσχετισμός ανάμεσα στους φοιτητές και στα μαθήματα, πραγματοποιείται μέσω εξαμήνου.

Σε ότι αφορά τους φοιτητές (students) και τις λειτουργίες τους, προσθέσαμε σε αυτές και τη δυνατότητα αποστολής αιτήματος για κάποιες αιτήσεις, στην γραμ-

ματεία. Οι αιτήσεις αυτές είναι δύο ειδών: α) αναλυτική βαθμολογία, β) πιστοποιητικό σπουδών. Άρα λοιπόν έπρεπε να δημιουργηθούν και οι ανάλογοι πίνακες.

- **Πίνακας student_request**



Εικόνα 42: Πίνακας student_request

Ο πίνακας αυτός περιέχει τα πεδία **id, xmlDoc** το οποίο είναι το xml που παίρνει την πληροφορία με τα χαρακτηριστικά και τις λεπτομέρειες της αίτησης και **modeId** το οποίο είναι το κλειδί με το ποίο συνδέεται ο πίνακας student_request και αναφέρεται στον πίνακα **student_request_mode**. Όπως και σε άλλους πίνακες, όπως για παράδειγμα στον πίνακα score, έτσι και εδώ, ορίζουμε με τα κατάλληλα Insert στο script της βάσης τις τιμές για τα αντίστοιχα Id. Έτσι λοιπόν, στο id:1, δίνουμε την τιμή με Name: OLD και στο id:2, την τιμή με Name: NEW,

```
insert into `student_request_mode` (`id`, `name`) values(1, 'OLD');
insert into `student_request_mode` (`id`, `name`) values(0, 'NEW');
```

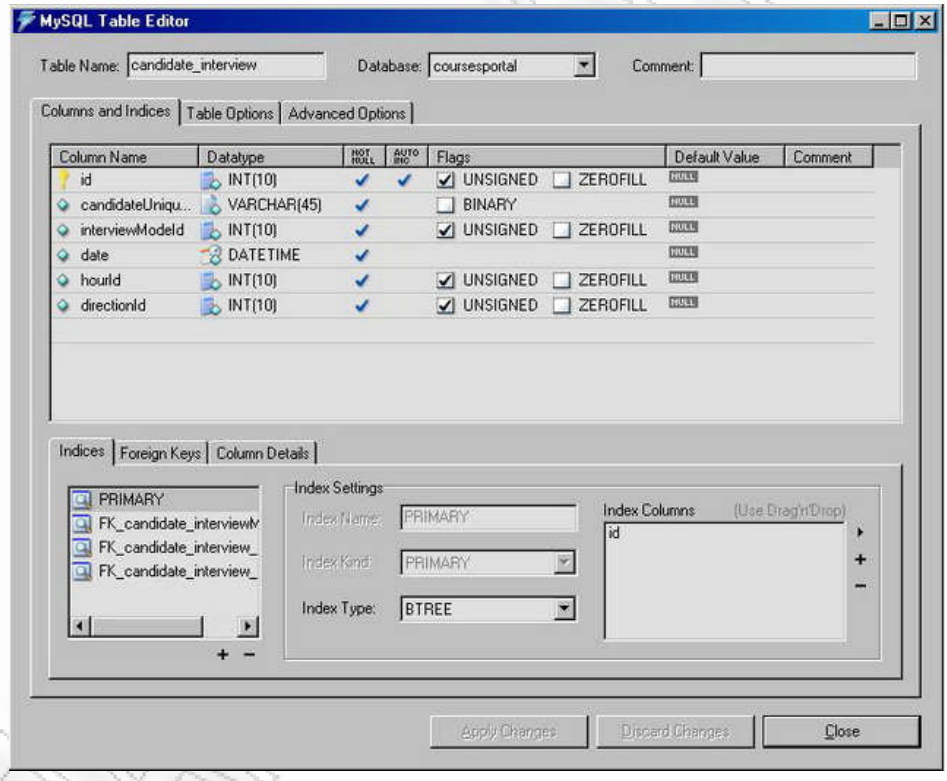
Στον τον πίνακα **student_request_type** εισάγουμε με τα κατάλληλα Insert τον τύπο των αιτήσεων.

```
insert into `student_request_type` (`id`, `name`) values(0, 'Βεβαίωση σπουδών');
insert into `student_request_type` (`id`, `name`) values(1, 'Αναλυτική Βαθμολογία');
```

Μέχρι τώρα είδαμε τους πίνακες που αφορούν την γραμματεία, τους φοιτητές, τους καθηγητές, τα Προγράμματα Μεταπτυχιακών Σπουδών και τις κατευθύνσεις τους καθώς και τις συσχετίσεις όλων αυτών με τα μαθήματα. Είδαμε επίσης τον πίνακα που αφορά τους υποψηφίους και τα διάφορα modes που θέτονται κατά την διάρκεια της αξιολόγησής τους.

Στη συνέχεια θα δούμε τους πίνακες που σχετίζονται με τις συνεντεύξεις των υποψηφίων.

- **Πίνακας candidate_interview**



Εικόνα 43: Πίνακας candidate_interview

Με τον πίνακα αυτό, είμαστε σε θέση να θέσουμε ημερομηνία και ώρα συνέντευξης ενός υποψηφίου, ανάλογα με την κατεύθυνσή του. Τα πεδία που αποτελούν τον πίνακα αυτό είναι:

- **Id** (primary key): Αύξον αριθμός και πρωτεύον κλειδί
- **candidateUniqueId**: Το μοναδικό id κάθε υποψηφίου
- **date**: Ημερομηνία συνέντευξης

o *Foreign Keys*

Για να γίνει μπορεί να γίνει ο συσχετισμός της ημερομηνίας συνέντευξης κάθε υποψηφίου ανάλογα με την κατεύθυνση την οποία έχει επιλέξει, είναι απαραίτητη η σύνδεση του πίνακα αυτού και με κάποια foreign keys. Καταρχήν, πρέπει να γίνει η σύνδεση με την κατεύθυνση για την οποία θα οριστεί ημερομηνία συνέντευξης. Επομένως συνδέουμε τον πίνακα, με το κλειδί *directionId* του πίνακα **direction.**

```
CONSTRAINT `FK_candidate_interview_3` FOREIGN KEY (`directionId`) REFERENCES `direction` (`id`)
```

Στον πίνακα courses είχε αναφερθεί ο τρόπος και ο λόγος δημιουργίας των πινάκων hour_of_day_for_course και hour_of_day_for_interview. Με τον πίνακα **hour_of_day_for_interview**, ορίζουμε με τα κατάλληλα Insert κάποιες ώρες. Οι ώρες αυτές αντιστοιχούν σε κάποιο Id. Έτσι η γραμματεία, θα μπορεί να επιλέξει όποια ως συνέντευξης, κάποια από αυτές που εμφανίζονται στο drop down menu. Οι ώρες αυτές είναι αντιπροσωπευτικές και έτσι θα μπορεί η γραμματεία να καταρτίσει ένα λειτουργικό για αυτήν και τους καθηγητές πρόγραμμα συνεντεύξεων. Επομένως γίνεται πολύ εύκολα και ο συσχετισμός του πίνακα **candidate_interview**, με το κλειδί *hourId* του πίνακα **hour_of_day_for_interview.**

```
CONSTRAINT `FK_candidate_interview_2` FOREIGN KEY (`hourId`) REFERENCES `hour_of_day_for_interview` (`id`)
```

Τέλος, δημιουργήθηκε και ένας ακόμη πίνακας, ο οποίος μας βοηθάει στον επαναπροσδιορισμό μιας ώρας συνέντευξης, όταν αυτό ζητηθεί από κάποιον υποψήφιο. Ο πίνακας αυτός είναι ο **interview_mode**, τον οποίο δημιουργούμε και πάλι με τα κατάλληλα Insert.

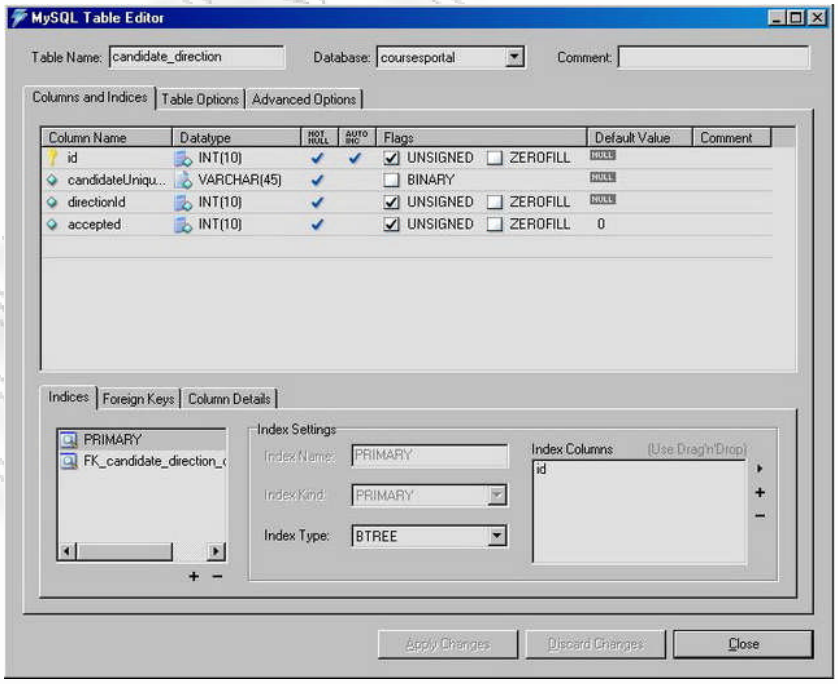
```
insert into `interview_mode` (`id`, `name`) values(0, 'INIT');
insert into `interview_mode` (`id`, `name`) values(1, 'ACCEPT');
insert into `interview_mode` (`id`, `name`) values(2, 'REQUEST_RESCHEDULE');
insert into `interview_mode` (`id`, `name`) values(3, 'FINLALIZE');
```

Συνδέοντας έτσι τον πίνακα candidate_interview, με το *interviewModeId* του πίνακα interviewMode, είμαστε σε θέση να ορίζουμε συνεντεύξεις για κάθε υποψήφιο ξεχωριστά, σε διαφορετική ώρα και με δυνατότητα αλλαγής της προτεινόμενης ώρας εφόσον ζητηθεί από τον υποψήφιο.

```
CONSTRAINT `FK_candidate_interview_1` FOREIGN KEY (`interviewModeId`) REFERENCES `interview_mode` (`id`),
```

Σε περίπτωση που ζητηθεί αλλαγή της ώρας από τον υποψήφιο, το mode που τον αφορά για την συνέντευξη, ορίζεται σε *'REQUEST_RESCHEDULE'*. Με την αλλαγή της ώρας από την γραμματεία, το mode του υποψηφίου θα μετατραπεί σε *'FINLALIZE'*.

- **Πίνακας candidate_direction**



Εικόνα 44:Πίνακας candidate_direction

Τέλος, σε ότι αφορά τον υποψήφιο, υλοποιήθηκε ένας ακόμη πίνακας ο οποίος είναι χρήσιμος κατά την δημιουργία του λογαριασμού του από την γραμματεία. Όπως προαναφέρθηκε και στους ρόλους, ο υποψήφιος μπορεί να δηλώσει υποψηφιότητα για πέντε συνολικά κατευθύνσεις του Π.Μ.Σ. του τμήματος. Έτσι για κάθε κατεύθυνση που επιλέγει ο υποψήφιος, δημιουργείται και μία καταχώρηση.

Ο πίνακας αυτός περιέχει τα εξής πεδία:

- **Id** (primary key): Αύξον αριθμός και πρωτεύον κλειδί
- **candidateUniqueId**: Το μοναδικό id κάθε υποψηφίου
- **accepted**: Εάν έχει γίνει δεκτός σε κάποια κατεύθυνση

o *Foreign Keys*

Όπως είπαμε ο υποψήφιος δηλώνει κάποιες κατευθύνσεις. Για να μπορεί να γίνει αυτό, συνδέουμε τον πίνακα candidate_direction με το κλειδί *directionId* που αναφέρεται στον πίνακα **direction**

```
CONSTRAINT `FK_candidate_direction_directionId` FOREIGN KEY (`directionId`) REFERENCES `direction` (`id`)
```

4.3. Ανάπτυξη εφαρμογής

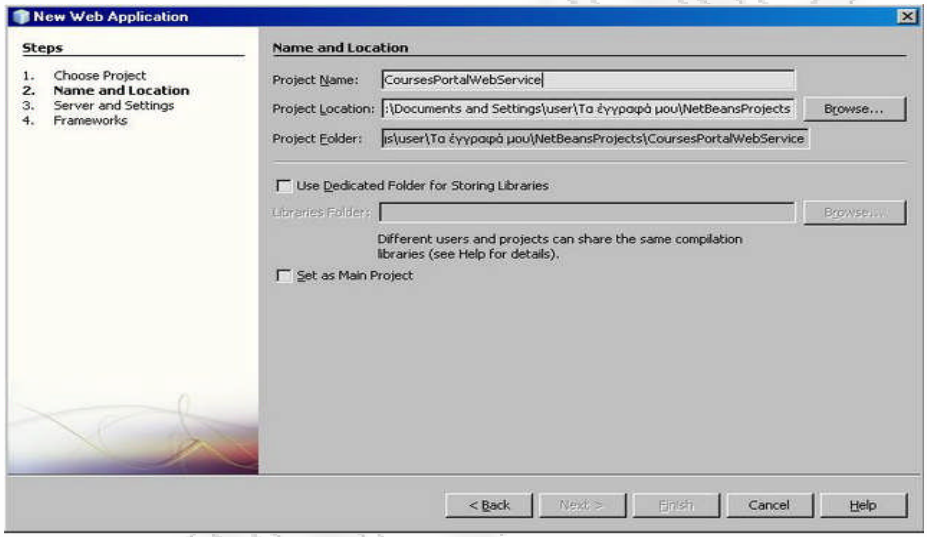
Αφού σχεδιάσαμε την αρχιτεκτονική της βάσης δεδομένων και καταλήξαμε σε αυτό το τελικό μοντέλο, ξεκινάει η υλοποίηση των αντικειμένων και των μεθόδων για τα διάφορα αντικείμενα.

Η εφαρμογή αποτελείται από δύο projects. Το CoursesPortalWebService και το CoursesPortalWebServiceClient. Στο CoursesPortalWebService υλοποιούνται προγραμματιστικά όλα τα αντικείμενα και οι μέθοδοι που θα χρειαστούμε για την σωστή λειτουργία του συστήματος. Επίσης στο CoursesPortalWebService, υλοποιείται το web service της εφαρμογής παράγοντας το WSDL. Το portal.java, όπως θα δούμε παρακάτω, είναι ουσιαστικά το web interface της εφαρμογής, στο οποίο υλοποιούνται όλες οι μέθοδοι του Client από τα jsp αρχεία.

Στο CoursesPortalWebServiceClient project, έχουν δημιουργηθεί όλες οι jsp οι οποίες δίνουν που δίνουν πρόσβαση στο interface της εφαρμογής καλώντας διάφορες μεθόδους κάθε φορά.

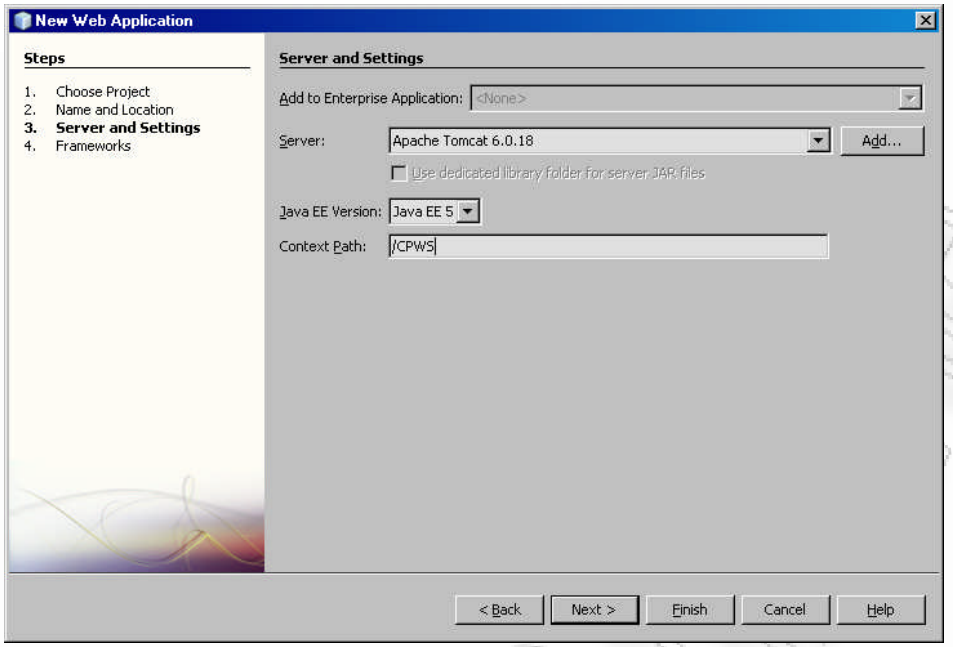
4.3.1 Υλοποίηση Project Web Service

Όπως έχει αναφερθεί, για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε το εργαλείο NetBeans IDE 6.7.1.. Τα βήματα δημιουργίας του web service project είναι τα εξής: Δημιουργήσαμε ένα New Project-Java Web-Web Application και το ονομάσαμε CoursesPortalWebServiceClient.



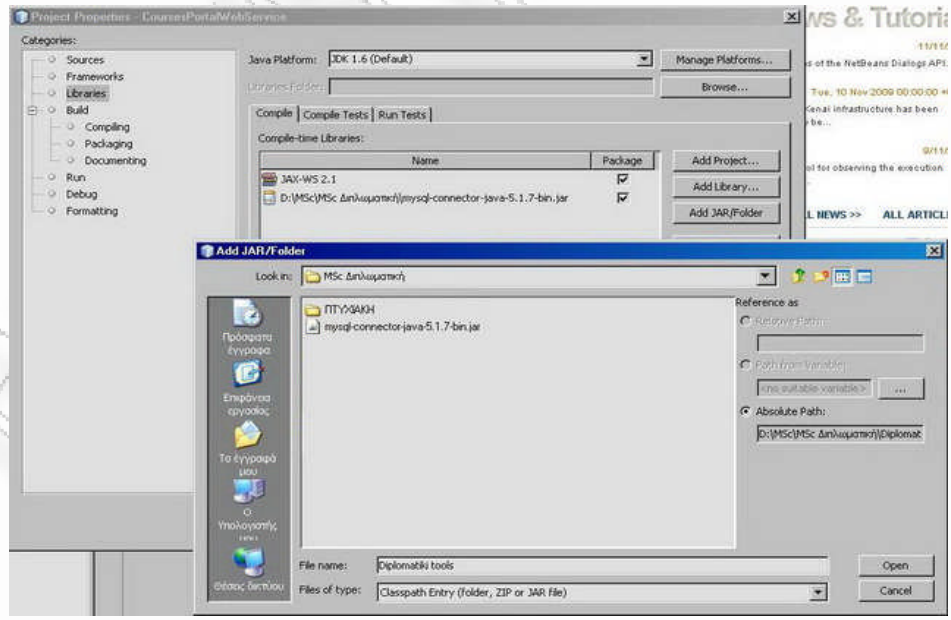
Εικόνα 45: Δημιουργία κυρίως project – CoursesPortalWebService

Στη συνέχεια ορίζουμε τον server και το Context Path που θα «βλέπει» ο server. Για λόγους συντόμευσης το Context Path ονομάστηκε CPWS. Ως server χρησιμοποιήθηκε ο Apache Tomcat 6.0.18.



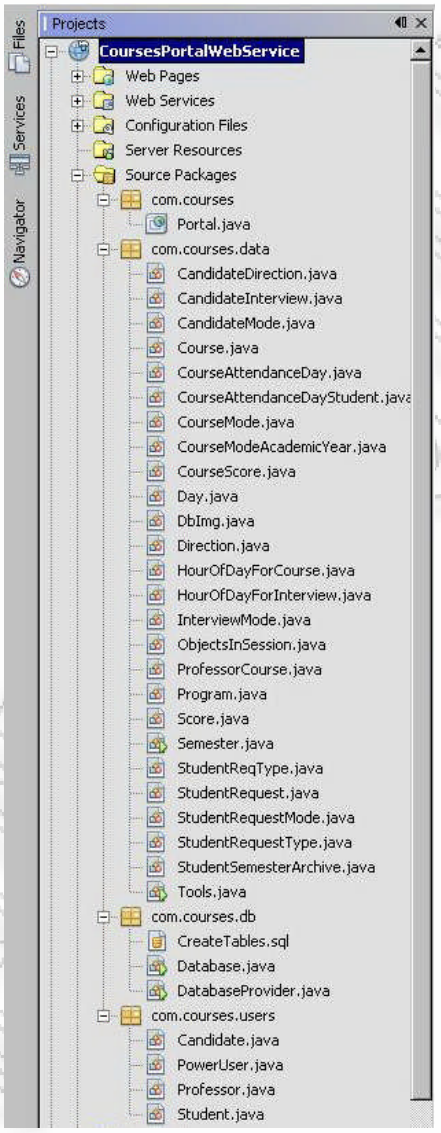
Εικόνα 46: Εγκατάσταση Server & Context Path

Με τα βήματα αυτά δημιουργήθηκε το κυρίως project CoursesPortalWebService. Αφού δημιουργήθηκε και είναι διαθέσιμο στα projects του NetBeans, ακολουθεί ένα πολύ σημαντικό βήμα. Η προσθήκη του αρχείου connector για την σύνδεση με την βάση δεδομένων. Ο connector που χρησιμοποιήθηκε είναι ο mysql-connector-java-5.1.7-bin.jar. Αυτό φαίνεται και στην παρακάτω εικόνα.



Εικόνα 47: Προσθήκη mysql-Connector

Στη συνέχεια δημιουργήσαμε ένα νέο source package, το οποίο ονομάσαμε *com* και μέσα στο οποίο δημιουργήσαμε τον φάκελο courses όπου υλοποιήθηκαν όλα τα αντικείμενα τα οποία είναι απαραίτητα για το σύστημά μας. Για λόγους λειτουργικότητας τα πακέτα αυτά διαχωρίζονται σε τέσσερις κατηγορίες, όπως φαίνεται και στην εικόνα 48 :



Εικόνα 48: Sourse Packages - Objects

- **com.courses**: Περιέχει το *portal.java*, το οποίο είναι ουσιαστικά το web interface της εφαρμογής, και υλοποιεί όλες τις μεθόδους του web service που καλούνται από τα jsp του client. Ακολουθεί ένα τμήμα του κώδικα **Portal.java**

```
package com.courses;

import com.courses.data.CandidateDirection;
import com.courses.data.CandidateInterview;
import com.courses.data.CandidateMode.CANDIDATE_MODE;
import com.courses.data.Course;
import com.courses.data.CourseAttendanceDay;
import com.courses.data.CourseAttendanceDayStudent;
import com.courses.data.CourseModeAcademicYear;
import com.courses.data.CourseScore;
import com.courses.data.Day;
import com.courses.data.Direction;
import com.courses.data.Program;
import com.courses.data.Semester;
import com.courses.data.DbImg;
import com.courses.data.HourOfDayForCourse;
import com.courses.data.HourOfDayForInterview;
import com.courses.data.ProfessorCourse;
import com.courses.data.Score;
import com.courses.data.StudentRequest;
import com.courses.data.StudentReqType;
import com.courses.data.StudentSemesterArchive;
import com.courses.db.Database;
import com.courses.users.Candidate;
import com.courses.users.PowerUser;
import com.courses.users.Professor;
import com.courses.users.Student;
import java.util.List;
import javax.jws.Oneway;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;

/**
 *
 * @author harris
 */

@WebService()
public class Portal {
```

```
    private Database db;

    public Portal() {
        db = new Database();

    }
    /**
     * Web service operation
     */
    @WebMethod(operationName = "t1")
    public String t1(@WebParam(name = "parameter")
    String parameter) {
        //TODO write your implementation code here:
        return null;
    }


    @WebMethod(operationName="semesterGetAll")
    public List<Semester> semesterGetAll(@WebParam(name = "isRunning") boolean isRun-
ning)
    {
        return db.semesterGetAll(isRunning);
    }

    @WebMethod(operationName = "semesterUpdateName")
    @Oneway
    public void semesterUpdateName(@WebParam(name = "id") int id, @WebParam(name =
"name")  String name)
    {
        db.semesterUpdateName(id, name);
    }

    @WebMethod(operationName = "semesterUpdateStatus")
    @Oneway
    public void semesterUpdateStatus(@WebParam(name = "id") int id, @WebParam(name =
"isRunning")  int isRunning)
    {
        db.semesterUpdateStatus(id, isRunning);
    }
```

```
    @WebMethod(operationName = "createNewCandidate")
    @Oneway
    public void createNewCandidate(@WebParam(name = "username")
    String username, @WebParam(name = "password")
    String password, @WebParam(name = "firstName")
    String firstName, @WebParam(name = "lastName")
    String lastName, @WebParam(name = "tel")
    String tel, @WebParam(name = "email")
    String email, @WebParam(name = "directionIds")
    List<Integer> directionIds
          ) {
        db.createNewCandidate(username, password, firstName, lastName, tel, email, directio-
nIds);
    }



    @WebMethod(operationName = "candidateRemoveByUniqueId")
    @Oneway
    public void candidateRemoveByUniqueId(@WebParam(name = "uniqueId") String unique-
Id) {
        db.candidateRemoveByUniqueId(uniqueId);
    }




    @WebMethod(operationName = "getCandidatesAllForThisYear")
    public List<Candidate> getCandidatesAllForThisYear()
    {
          return db.getCandidatesAllForThisYear();

    }
```

- **com.data**: Εδώ κατασκευάστηκαν όλα τα αντικείμενα τα οποία έχουν σχέση με την εφαρμογή μας. (CandidateDirection.java, Course.java, Day.java, Student.java κλπ). Καθένα από αυτά περιέχει απλές μεταβλητές, τους απαραίτητους constructors και getters και setters. Ένα τμήμα της υλοποίησης ενός αντικειμένου, πχ του **Course.java** παρατίθεται παρακάτω.

```
package com.courses.data;

import java.sql.ResultSet;
import java.sql.SQLException;

public class Course
{

   private int id;

private String name;
            private int active; //0 false, 1 true
            private int directionId;
            private int semesterId;
           private int dayId;
           private int startHourId;
           private int endHourId;



   public Course() {
      super();
      // TODO Auto-generated constructor stub
   }

   public Course(int id, String name, int active, int directionId, int semesterId, int dayId, int startHourId, int endHourId) {
      this.id = id;
      this.name = name;
      this.active = active;
      this.directionId = directionId;
      this.semesterId = semesterId;
      this.dayId = dayId;
      this.startHourId = startHourId;
      this.endHourId = endHourId;
   }

   public Course(ResultSet rs) throws SQLException {
      this.id = rs.getInt("id");
      this.name = rs.getString("name");
      this.active = rs.getInt("active");
      this.directionId = rs.getInt("directionId");
```

```
        this.semesterId = rs.getInt("semesterId");
        this.dayId = rs.getInt("dayId");
        this.startHourId = rs.getInt("startHourId");
        this.endHourId = rs.getInt("endHourId");
    }

    public int getId() {
        return id;
    }

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

    public int getDirectionId() {
        return directionId;
    }

    public void setDirectionId(int directionId) {
        this.directionId = directionId;
    }
```

- **com.courses.users**: Η μόνη διαφορά του πακέτου αυτού με το courses.data είναι ότι τοποθετήσαμε μαζί και κατασκευάσαμε για λόγους ευκολίας, όλους τους χρήστες οι οποίοι θα αλληλεπιδρούν με το σύστημα (candidate.java, poweruser.java, student.java, professor.java). Ομοίως παρατίθεται τμήμα κώδικα για την υλοποίηση του user **PowerUser.java**

```
package com.courses.users;

import java.sql.ResultSet;
import java.sql.SQLException;

public class PowerUser
{
        private String username;
        private String password;
```

```
	private String firstName;
	private String lastName;
	private String tel;



	public PowerUser()
	{
		super();
		// TODO Auto-generated constructor stub
	}
	public PowerUser(String username, String password, String firstName,
			String lastName, String tel)
	{
		super();
		this.username = username;
		this.password = password;
		this.firstName = firstName;
		this.lastName = lastName;
		this.tel = tel;

	}

	public PowerUser(ResultSet rs) throws SQLException
	{
		this.username = rs.getString("username");
		this.password = rs.getString("password");
		this.firstName = rs.getString("firstName");
		this.lastName = rs.getString("lastName");
		this.tel = rs.getString("tel");

	}


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

```
	public String getPassword()
	{
		return password;
	}
	public void setPassword(String password)
	{
		this.password = password;
	}
	public String getFirstName()
	{
		return firstName;
	}
	public void setFirstName(String firstName)
	{
		this.firstName = firstName;
	}
```

- **com.courses.db**: Τέλος το πακέτο αυτό περιλαμβάνει το script δημιουργίας της βάσης δεδομένων CreateTables.sql και το αρχείο DatabaseProvider.java με το οποίο πραγματοποιείται η σύνδεση του web service με την βάση δεδομένων και η οποία είναι static. Με την μέθοδο *DatabaseProvider.getConnection* δημιουργείται μία φορά η σύνδεση και στη συνέχεια χρησιμοποιείται οπουδήποτε μέσα στον κώδικα χωρίς να χρειάζεται *newConnection.* Παρακάτω παρουσιάζεται τμήμα κώδικα του **DatabaseProvider.java**

```
public class DatabaseProvider
{
	private static Connection con;


	private DatabaseProvider()
	{

	}



	public static Connection getConnection()
```

```
        {

                if(con == null)
                {

                        String                                                  url
="jdbc:mysql://localhost:3306/coursesportal?characterEncoding=UTF-8";

                    try {
                        try {
                                Class.forName("com.mysql.jdbc.Driver");
                        } catch (ClassNotFoundException e) {
                                e.printStackTrace();
                        }
                        con = DriverManager.getConnection(url,"root", "***");
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }

                }

                return con;
```

Το αρχείο Database.java είναι το πιο σημαντικό αντικείμενο του Project. Αρχικά γίνεται ένα Numeration, μια νοητή συσχέτιση με τα πραγματικά objects που θα χρησιμοποιηθούν. Στη συνέχεια ορίζουμε όλα τα Prepare Statements (Microsystems, 1995 - 2008) που θα τρέχουν στη βάση δεδομένων χωρίς τα πραγματικά δεδομένα. Για κάθε δομή δημιουργούμε τα απαραίτητα insert, create, update, remove, getAll κλπ. Στη συνέχεια αρχικά δημιουργούμε στον constructor το academicYear το οποίο θα χρησιμοποιηθεί στη συνέχεια και έπειτα δημιουργούμε όλες τις μεθόδους που αντιστοιχούν στα παραπάνω Prepare Statements. Ο τρόπος με τον οποίο γίνεται το αρχικό numeration φαίνεται παρακάτω

```
public class Database {

   private static enum dBStatementsEnum {
```

```
    SemesterTruncate,
    SemesterCreateNewEntry,
    SemesterGetAll,
    SemesterGetAllRunning,
    SemesterUpdateName,
    SemesterUpdateStatus,
    //          SemesterGetNameById,

    CandidateModeTruncate,
    CandidateModeCreateNewEntry,

    InterviewModeCreateNewEntry,

    StudentRequestModeCreateNewEntry,
    StudentRequestTypeCreateNewEntry,

    CourseModeCreateNewEntry,

    CourseModeAcademicYearCreateNewEntry,
    CourseModeAcademicYearGetAll,
    CourseModeAcademicYearUpdate,

    StudentRequestCreateNewEntry,
    StudentRequestGetAll,
    StudentRequestUpdate,

    CourseAttendanceDayCreateNewEntry,
    CourseAttendanceDayGetAll,

    CourseAttendanceDayStudentCreateNewEntry,
    CourseAttendanceDayStudentGetAll,
    CourseAttendanceDayStudentRemove,
    CourseAttendanceDayStudentRemoveByCourseIdAndYearId,

    ScoreCreateNewEntry,
    ScoreGetAll,

    CourseScoreCreateNewEntry,
    CourseScoreRemoveByCidSid,
    CourseScoreGetAll,
```

```
    ProfessorCreateNewEntry,
    ProfessorGetAll,
    ProfessorGetByUsernameAndPassword,
    ProfessorUpdateMe,
```

Ενώ η δήλωση των prepare statements έγινε ως εξής:

```
private static PreparedStatement[] dBStatements;

  static {

    dBStatements = new PreparedStatement[dBStatementsEnum.values().length];

    dBStatements[dBStatementsEnum.SemesterTruncate.ordinal()] =
Tools.createPs(DatabaseProvider.getConnection(), "truncate semester");
    dBStatements[dBStatementsEnum.SemesterCreateNewEntry.ordinal()] =
Tools.createPs(DatabaseProvider.getConnection(), "insert into `semester` (`id`, `name`) val-
ues(?, ?)");
    dBStatements[dBStatementsEnum.SemesterGetAll.ordinal()] =
Tools.createPs(DatabaseProvider.getConnection(), "SELECT * FROM semester ");
    dBStatements[dBStatementsEnum.SemesterGetAllRunning.ordinal()] =
Tools.createPs(DatabaseProvider.getConnection(), "SELECT * FROM semester where
isRunning=1");
    dBStatements[dBStatementsEnum.SemesterUpdateName.ordinal()] =
Tools.createPs(DatabaseProvider.getConnection(), "UPDATE semester SET name=? where
id=?");
    dBStatements[dBStatementsEnum.SemesterUpdateStatus.ordinal()] =
Tools.createPs(DatabaseProvider.getConnection(), "UPDATE semester SET isRunning=?
where id=?");
//                  dBStatements[dBStatementsEnum.SemesterGetNameById.ordinal()] =
Tools.createPs(DatabaseProvider.getConnection(), "SELECT name FROM semester s where
s.id = ?");

    dBStatements[dBStatementsEnum.CandidateModeTruncate.ordinal()] =
Tools.createPs(DatabaseProvider.getConnection(), "truncate candidate_mode");
    dBStatements[dBStatementsEnum.CandidateModeCreateNewEntry.ordinal()] =
Tools.createPs(DatabaseProvider.getConnection(), "insert into `candidate_mode` (`id`,
`name`) values(?, ?)");
```

```
dBStatements[dBStatementsEnum.studentRequestTypeGetAll.ordinal()] = Tools.createPs(DatabaseProvider.getConnection(), "SELECT * FROM student_request_type order by id");

dBStatements[dBStatementsEnum.InterviewModeCreateNewEntry.ordinal()] = Tools.createPs(DatabaseProvider.getConnection(), "insert into `interview_mode` (`id`, `name`) values(?, ?)");

dBStatements[dBStatementsEnum.StudentRequestModeCreateNewEntry.ordinal()] = Tools.createPs(DatabaseProvider.getConnection(), "insert into `student_request_mode` (`id`, `name`) values(?, ?)");
dBStatements[dBStatementsEnum.StudentRequestTypeCreateNewEntry.ordinal()] = Tools.createPs(DatabaseProvider.getConnection(), "insert into `student_request_type` (`id`, `name`) values(?, ?)");

dBStatements[dBStatementsEnum.CourseModeCreateNewEntry.ordinal()] = Tools.createPs(DatabaseProvider.getConnection(), "insert into `course_mode` (`id`, `name`) values(?, ?)");

dBStatements[dBStatementsEnum.StudentSemesterArchiveCreateNewEntry.ordinal()] = Tools.createPs(DatabaseProvider.getConnection(), "insert into `student_semester_archive` (`academicYear`, `semesterId`, `studentUniqueId`) values(?, ?, ?)");
dBStatements[dBStatementsEnum.StudentSemesterArchiveGetAll.ordinal()] = Tools.createPs(DatabaseProvider.getConnection(), "SELECT * FROM student_semester_archive order by academicYear, semesterId");
```

Λόγω του τεράστιου όγκου, είναι αδύνατον να παρατεθεί ο κώδικας όλου του project. Ο παραπάνω κώδικας είναι απλά ένα τμήμα από τα αντίστοιχα αντικείμενα Java που υλοποιήθηκαν.

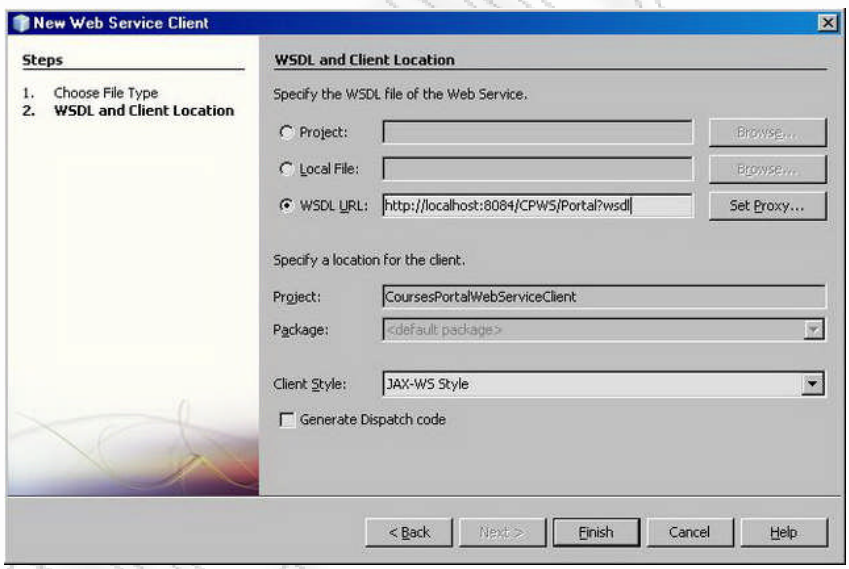
4.3.2 Υλοποίηση Client Project

Η μεθοδολογία για την δημιουργία του client είναι η ίδια με την δημιουργία του web service project. Αυτό που διαφοροποιείται σε αυτό το Project, CoursesPortalWebServiceClient, είναι ότι πλέον το Context Path το ονομάσαμε για λόγους συντόμευσης CPWSC.

Για να βεβαιωθούμε ότι το web service λειτουργεί κανονικά, κάνουμε deploy το web service project CoursesPortalWebService. Όταν ολοκληρωθεί το deploy, πληκτρολογούμε σε έναν internet browser το URL: http://localhost:8084/CPWS/Portal. Εαν δούμε την παρακάτω οθόνη, το web service λειτουργεί κανονικά και βλέπουμε το WSDL το οποίο μας παράγει.

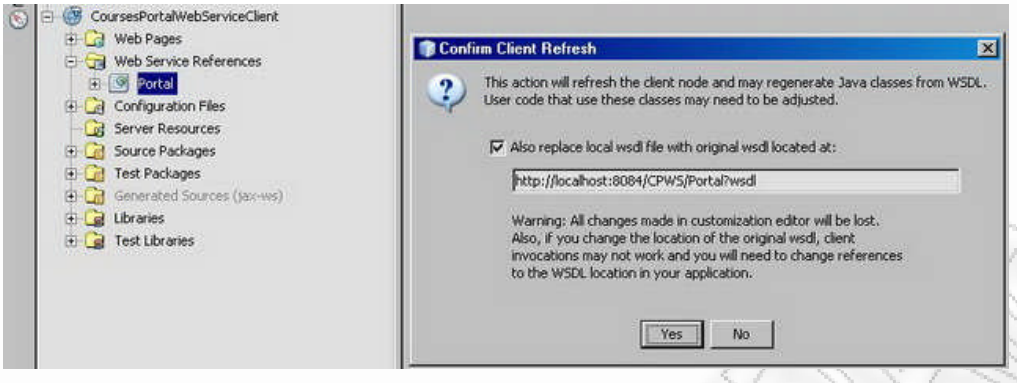
Εικόνα 49: Web Service deploy

Αυτό είναι το WSDL του Web Service μας το οποίο πρέπει να οριστεί στον client.



Εικόνα 50: Δήλωση WSDL στον client

Για να περιέχει κάθε φορά την σωστή πληροφορία, το wdsl θα πρέπει να αντι-καθίσταται κάθε φορά πριν το deploy του client, από το νέο wdsl που παράγεται κατά το deploy του web service.



Εικόνα 51: WSDL refresh

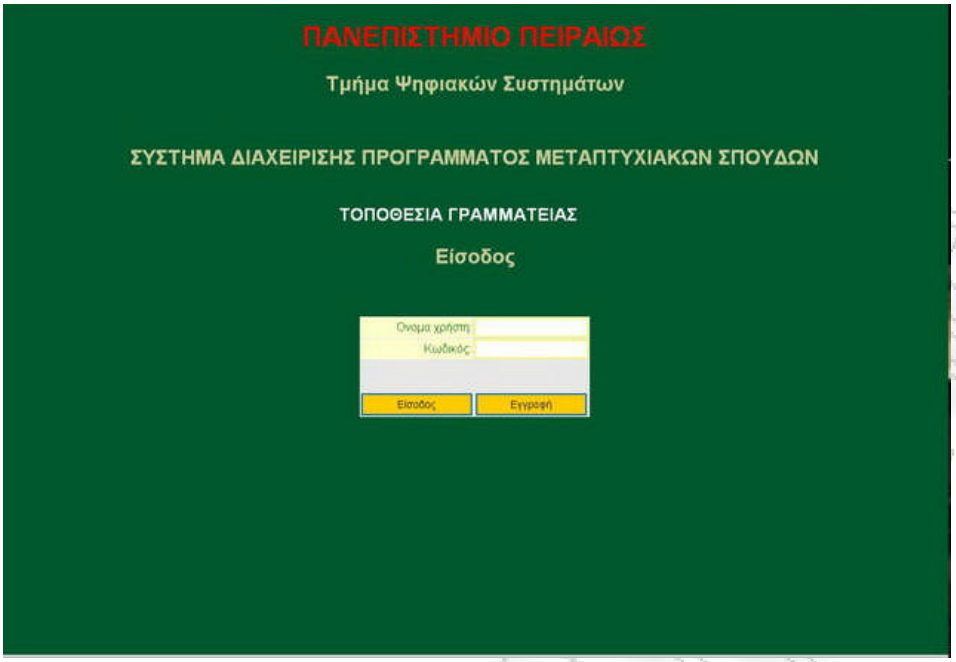
Όλα αυτά τα βήματα είναι απαραίτητα για την σωστή λειτουργία της εφαρμογής σε επίπεδο web service. Για να αποκτήσουμε πρόσβαση και σε επίπεδο web interface αναπτύχθηκαν οι κατάλληλες jsp σελίδες, με τις οποίες οι clients του συστήματος (poweruser, candidate, student, professor) καλούν κάποιες μεθόδους του web service και αλληλεπιδρούν μεταξύ τους.

4.4 Java Server Pages – Client Interface

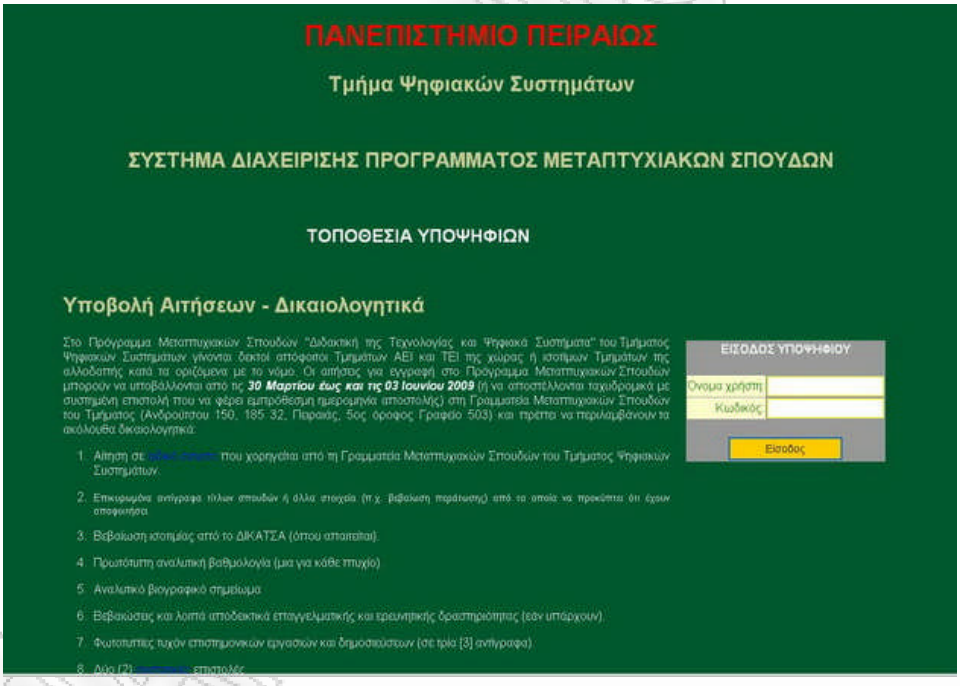
Έχοντας πλέον δημιουργήσει και συνδέσει με την βάση δεδομένων όλα τα απαραίτητα Objects, δημιουργούμε τις σελίδες για την διαδικτυακή διεπαφή με την οποία ο χρήστης θα αλληλεπιδρά με το σύστημα.

4.4.1 Clients Login

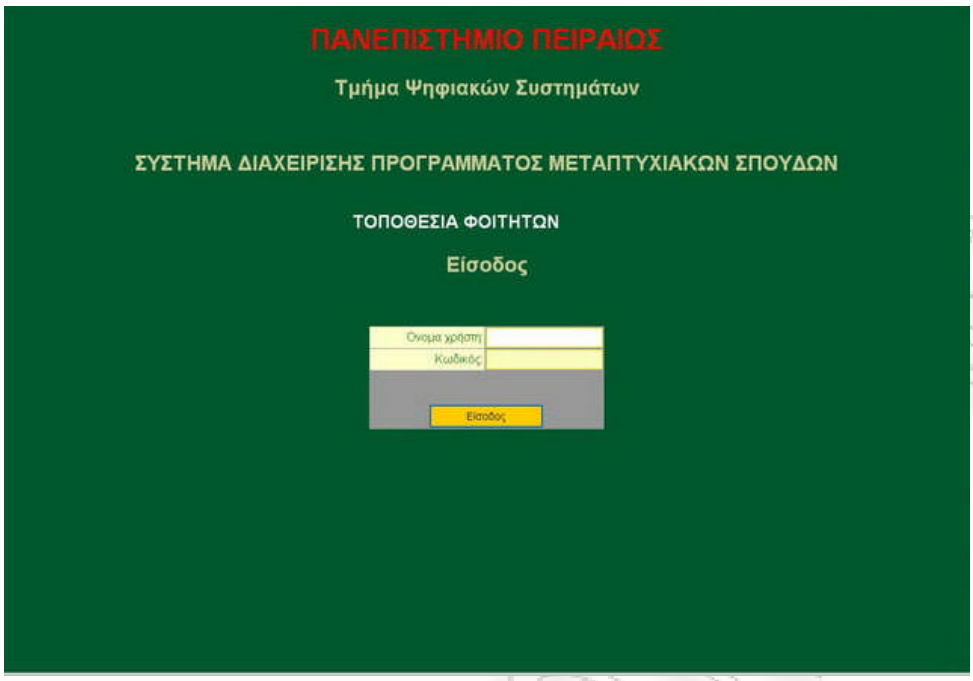
Καταρχήν, δημιουργήσαμε για κάθε client (poweruser, candidate, student, professor) αρχικές σελίδες εισόδου (login) ώστε να εισέρχονται στο σύστημα όλοι πιστοποιημένα. Μόνο η γραμματεία μπορεί να δημιουργήσει τον δικό της λογαριασμό, μέσω της επιλογής *Εγγραφή*. Παρατηρείτε πως οι άλλοι clients έχουν μόνο την επιλογή της εισόδου. Η γραμματεία είναι υπεύθυνη για την δημιουργία των λογαριασμών των υποψηφίων, των φοιτητών και των καθηγητών και τους παρέχει τα αντίστοιχα usernames και passwords.



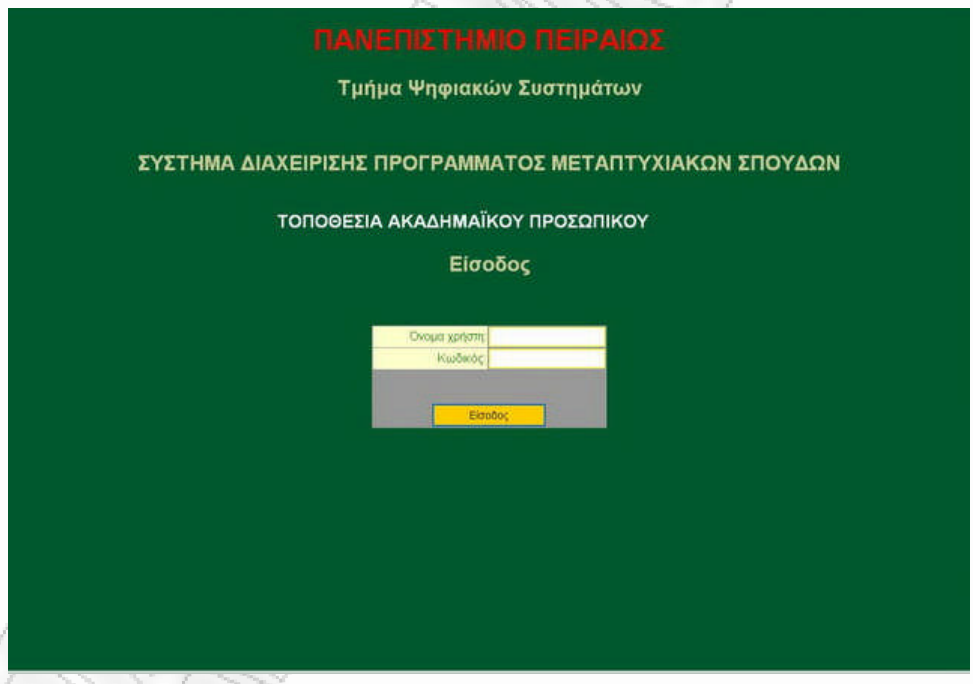
Εικόνα 52: PowerUserLogin.jsp



Εικόνα 53: CandidateLogin.jsp



Εικόνα 54:StudentLogin.jsp



Εικόνα 55: ProfessorLogin.jsp

Αρχικά, το σύστημα μάς δίνει ένα session και κάθε φορά που μπαίνει ο εκάστοτε client στην αντίστοιχη Login σελίδα του, ξαναδημιουργείται το session για να διαγράφονται τυχόν δεδομένα που έχουν μείνει στο session.

```
session.invalidate();
session = request.getSession(true);
```

Το πρώτο που αποθηκεύεται στο session είναι το Portal portal.

```
Portal portal = (Portal) session.getAttribute("portal");
```

Και στη συνέχεια ελέγχουμε εάν το portal είναι null. Εάν γίνει κάποια προσπάθεια εισόδου σε κάποια άλλη σελίδα χωρίς την ασφαλή είσοδο από τις login σελίδες θα γίνεται redirect στην σελίδα login, διότι αυτό θα σημαίνει είτε ότι προσπάθησε κάποιος να μπει παρακάμπτοντας την αρχική διαδικασία του Login, είτε ότι έγινε time-out στο session.

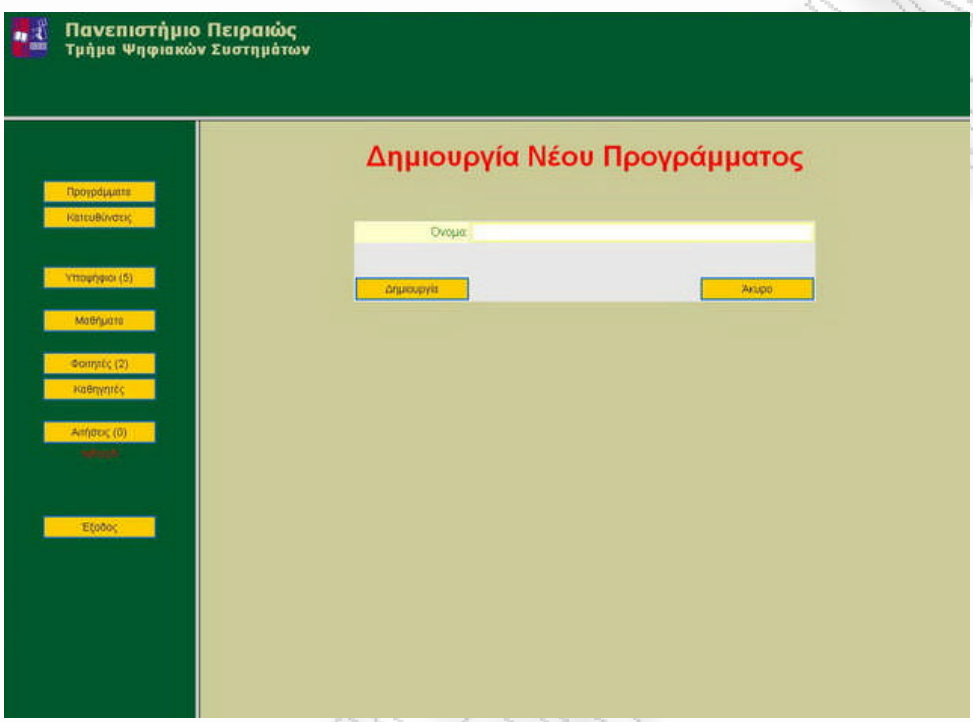
4.4.2 Interface και λειτουργίες γραμματείας (PowerUser)

Ξεκινώντας από τον βασικό client του συστήματος μας, την γραμματεία (poweruser), έχουμε καθορίσει κατά τον σχεδιασμό του συστήματος τις βασικές λειτουργίες τις οποίες θα μπορεί να εκτελεί. Οι λειτουργίες αυτές είναι:

- Δημιουργία Προγραμμάτων Μεταπτυχιακών Σπουδών
- Δημιουργία Κατευθύνσεων ανά Πρόγραμμα Μεταπτυχιακών Σπουδών
- Δημιουργία Μαθημάτων
- Δημιουργία Λογαριασμών Υποψηφίων
- Φοιτητές
- Δημιουργία Λογαριασμών Καθηγητών
- Αιτήσεις

Το πρώτο και πιο σημαντικό βήμα για την όλη διαδικασία, είναι η δημιουργία Προγράμματος Μεταπτυχιακών Σπουδών. Εάν δεν έχει δημιουργηθεί κάποιο πρόγραμμα, δεν μπορούμε να καταχωρήσουμε κατευθύνσεις, μαθήματα και να δημιουργήσουμε λογαριασμούς υποψηφίων. Έτσι μετά από την επιτυχή του είσοδο στο σύστημα (PowerUserLoginSuccessful.jsp),ο χρήστης της γραμματείας έχει τη δυνατότητα από την επιλογή *Προγράμματα* (PowerUserViewPrograms.jsp) να επιλέξει την *Δημιο-*

υργία Νέου Προγράμματος και να μεταφερθεί στην PowerUserCreateNewProgram.jsp όπου και μπορεί να καταχωρήσει ένα νέο Π.Μ.Σ όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 56: PowerUserCreateNewProgram.jsp

Πλέον μπορούμε να δημιουργήσουμε με παρόμοιο τρόπο και τις κατευθύνσεις που αντιστοιχούν σε κάθε μεταπτυχιακό πρόγραμμα. Επιλέγοντας από την επιλογή *Κατευθύνσεις* (PowerUserViewDirections.jsp) την *Δημιουργία Νέας Κατεύθυνσης* (PowerUserCreateNewDirection.jsp), ο χρήστης μεταφέρεται στην αντίστοιχη σελίδα όπου μπορεί να καταχωρήσει τις κατευθύνσεις που επιθυμεί.

Κάθε κατεύθυνση όμως, αντιστοιχεί σε κάποιο μεταπτυχιακό πρόγραμμα. Για να πετύχουμε τη σύνδεση της κάθε κατεύθυνσης που θα δημιουργήσουμε, με το μεταπτυχιακό πρόγραμμα στο οποίο στην πραγματικότητα ανήκει, τόσο η κλάση Direction όσο και ο πίνακας direction στην βάση, περιέχουν το πεδίο programId. Έτσι κάθε κατεύθυνση είναι συνδεδεμένη με ένα πρόγραμμα. Στα SaveDirectionsToSession.jsp και SaveAll.jsp αποθηκεύεται μια λίστα από αντικείμενα τύπου Direction μέσα στο session. Έτσι με την μέθοδο getProgramId() γνωρίζουμε σε ποιο πρόγραμμα αντιστοιχεί μια συγκεκριμένη κατεύθυνση.

```
List<Direction> directions = dbImg.getDirections();
    Hashtable<Integer, Direction> directionsById = new Hashtable<Integer, Direction>();
    Hashtable<Integer, Vector<Direction>> directionsByProgId = new Hashtable<Integer, Vector<Direction>>();

    Direction direction;
    for(int i=0;i<directions.size();i++)
    {
       direction = directions.get(i);
       directionsById.put(direction.getId(), direction);

       if(directionsByProgId.get(direction.getProgramId()) == null)
          directionsByProgId.put(direction.getProgramId(), new Vector<Direction>());

       directionsByProgId.get(direction.getProgramId()).add(direction);

    }

    session.setAttribute(ObjectsINSESSION.DIRECTIONS.toString(), directions);
    session.setAttribute(ObjectsINSESSION.DIRECTIONS_BY_ID.toString(), directionsById);
    session.setAttribute(ObjectsINSESSION.DIRECTIONS_BY_PROG_ID.toString(), directionsByProgId);
```

Ένα από τα αντικείμενα που δημιουργήσαμε είναι και το DbImg.java. Αυτό μας δίνει την εικόνα της βάσης. Κάνοντας μία κλήση στο portal,

```
DbImg dbImg = portal.getDbImg();
```

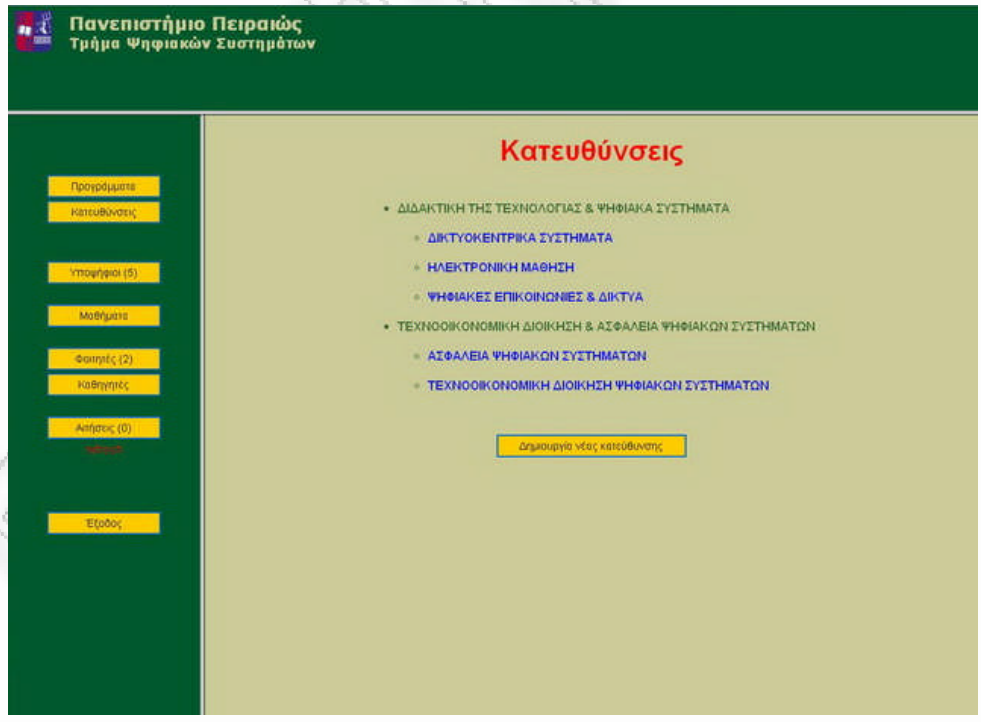
παίρνουμε όλα τα δεδομένα της βάσης οργανωμένα μέσα στο DbImg. Μέσα στην DbImg, έχουμε δημιουργήσει όλους τους απαραίτητους getters και setters για να παίρνει όλα τα δεδομένα που έχει αποθηκεύσει αυτή μέσα. Για να μπορέσουμε να εμφανίζουμε τις κατευθύνσεις ανάλογα με τα προγράμματα, παίρνουμε μια λίστα με τα Directions. Θέλουμε ένα hash table με βάση το id και θέλουμε και μία οργάνωση με βάση το πρόγραμμα.

Επομένως, θέλουμε έναν hash table ο οποίος να έχει ένα κλειδί Integer, το programId δηλαδή, και έναν Vector από Directions. Γίνεται parsing η λίστα με τα directions που μας φέρνει η DbImg και στον hash table που περιέχει το id τοποθετούμε τα directions με κλειδί το id. Στον άλλον hash table που περιέχει Vector, ελέγχου-

με πρώτα εάν έχει δημιουργηθεί κάποιος vector γιατί πρέπει να δημιουργηθεί μόνο μια φορά. Εάν είναι null, τότε τον δημιουργούμε και άρα εξασφαλίζουμε ότι πάντα έχει δημιουργηθεί ο vector. Οπότε κάνοντας ένα get στον hash table, μας επιστρέφει τον vector και κάνουμε add πλέον το direction μέσα στον Vector. Επομένως κάνοντας ένα get με κλειδί το programId, παίρνουμε έναν vector με όλα τα directions που αντιστοιχούν στο programId.

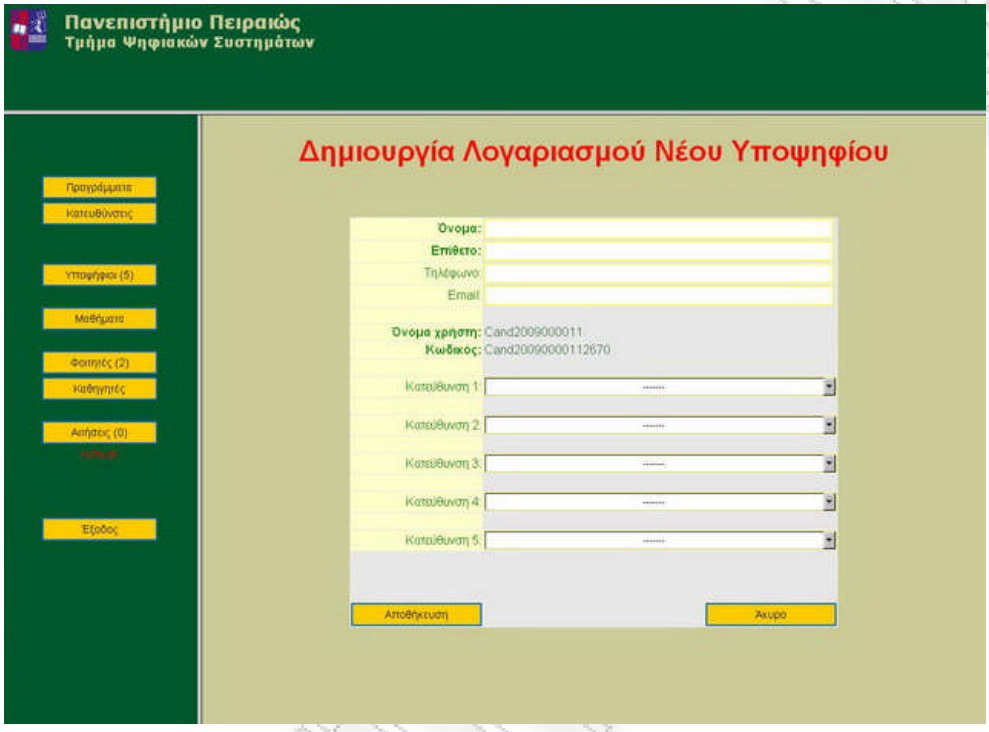
Σε γενικές γραμμές, χρησιμοποιούμε hashtables για ταχύτητα αφού μπορούμε σε ένα βήμα (με την χρήση του κλειδιού) να έχουμε την πληροφορία που θέλουμε. Διαφορετικά θα έπρεπε να παρσάρουμε κάθε φορά όλη την λίστα. Εναλλακτικά, θα μπορούσαμε να κάνουμε SELECT στην βάση και να πάρουμε την πληροφορία που θέλουμε αλλά αυτό θα ήταν θεαματικά χρονοβόρο για να το κάνουμε μέσα σε όλα αυτά τα for loops που έχουμε. Έτσι κάνουμε το SELECT μια φορά στην αρχή και αποθηκεύω το αποτέλεσμα σε λίστες και hashtables. Αυτή η λύση κοστίζει λίγο παραπάνω σε μνήμη αλλά έχουμε την ταχύτητα που θέλουμε.

Με τον τρόπο αυτό, βλέπουμε τις κατευθύνσεις που δημιουργήσαμε, ανά Πρόγραμμα και επομένως έχουμε εξασφαλίσει την βασική λειτουργική δομή συνδέοντας τα μεταπτυχιακά προγράμματα με τις κατευθύνσεις τους.



Εικόνα 57: PowerUserViewDirections.jsp

Καθώς οι υποψήφιοι αποστέλλουν ή καταθέτουν τις αιτήσεις τους, η γραμματεία δημιουργεί τους λογαριασμούς τους για είσοδό τους στο σύστημα. Η λειτουργία αυτή, γίνεται από την επιλογή *Υποψήφιοι,* επιλέγοντας *Δημιουργία νέου Υποψηφίου* (CreateNewCandidate.jsp).



Εικόνα 58: CreateNewCandidate.jsp

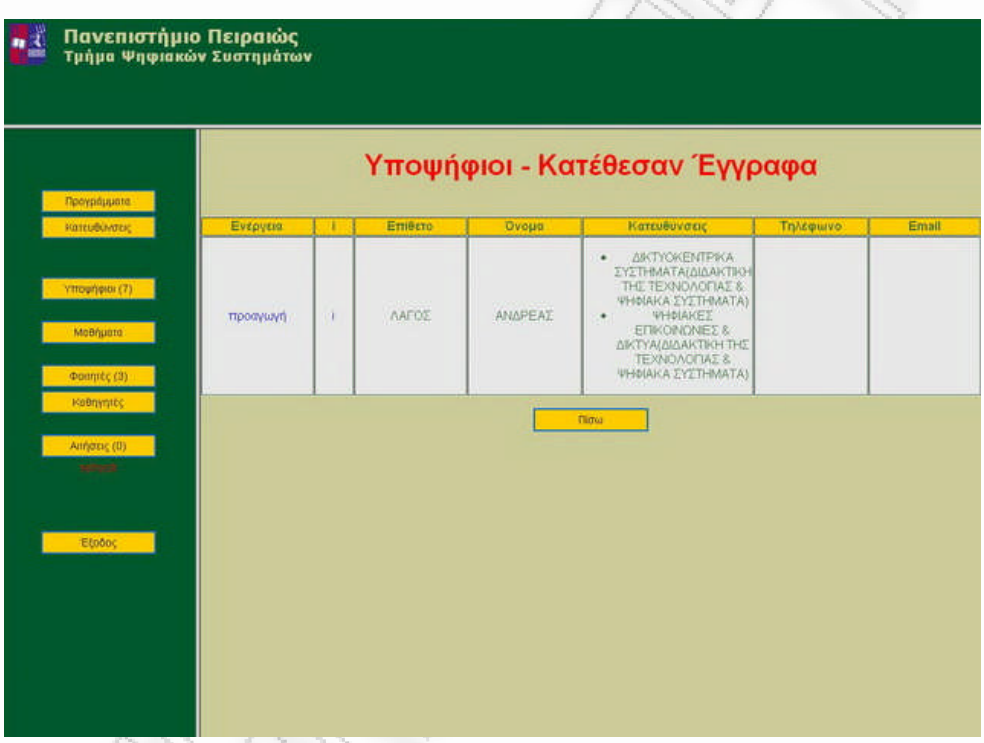
Τα στοιχεία καταχωρούνται στην βάση δεδομένων, και κάθε υποψήφιος αποκτά τον λογαριασμό του στο σύστημα. Όπως έχει ήδη αναφερθεί, για την αποφυγή όμοιας επιλογής ονόματος χρήστη από τους υποψήφιους, το σύστημα παράγει μόνο του όνομα χρήστη και κωδικό πρόσβασης της μορφής : Cand200900000x.

```
public String getProposedCandidateUsername() {
      int nextId = getCandidateCntFotThisYear() + 1;
      return "Cand" + Tools.getCurrentYear() + String.format("%06d", nextId);

   }

   public String getProposedCandidatePassword() {
      int nextId = getCandidateCntFotThisYear() + 1;
      return "Cand" + Tools.getCurrentYear() + String.format("%06d", nextId) + getRandomNumber();
```

Επίσης, κατά την δημιουργία του λογαριασμού του υποψηφίου, η γραμματεία καταχωρεί στο σύστημα τις κατευθύνσεις που επέλεξε ο υποψήφιος με σειρά προτεραιότητας. Αυτό θα φανεί πολύ σημαντικό στη συνέχεια, για τον καθορισμό της κατεύθυνσης στην οποία θα γίνει τελικά αποδεκτός ένας υποψήφιος.

Όταν η γραμματεία δημιουργήσει τους λογαριασμούς των υποψηφίων, στην επιλογή *Υποψήφιοι* βλέπει τις εξής επιλογές (Σε παρένθεση το πλήθος των υποψηφίων στην εκάστοτε κατάσταση την τρέχουσα στιγμή):

- Δείτε υποψηφίους που έχουν καταθέσει έγγραφα(1)

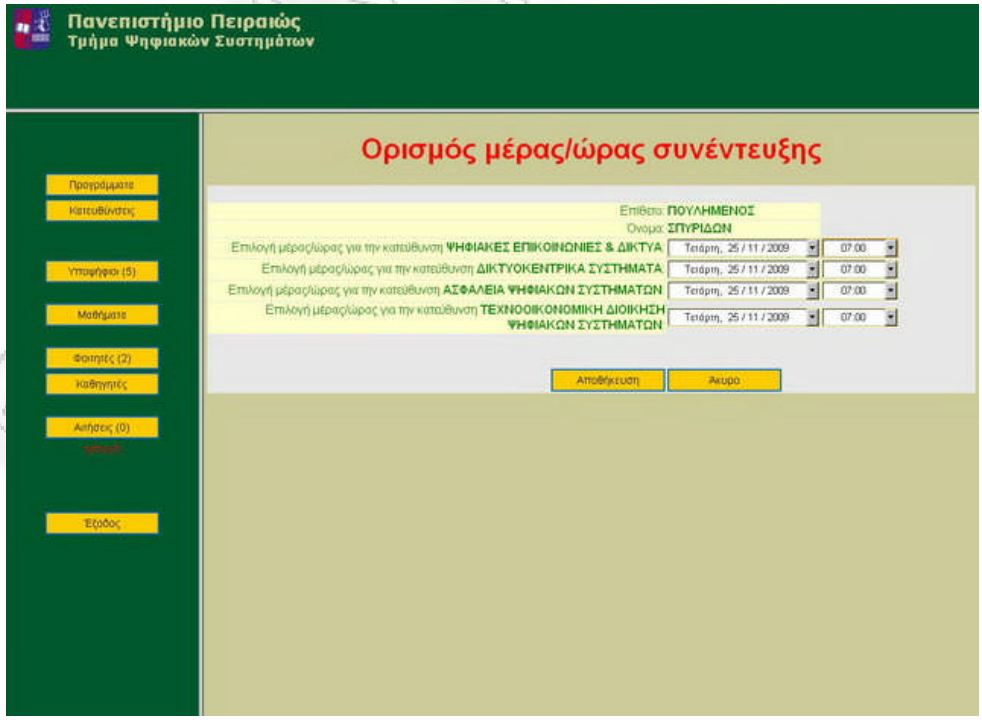


Εικόνα 59: PowerUserViewDeliveredDocumentsCandidates.jsp

- Δείτε υποψηφίους των οποίων τα έγγραφα έγιναν δεκτά(2)

Εικόνα 60: PowerUserViewDocumentsAcceptedCandidates.jsp

Στο σημείο αυτό, επιλέγοντας *προαγωγή* μεταβαίνουμε στη σελίδα όπου θα οριστούν ημερομηνίες συνεντεύξεων για τον συγκεκριμένο υποψήφιο (δίπλα από το όνομα του οποίου επιλέχθηκε *προαγωγή*) και για την κάθε κατεύθυνση που επέλεξε. (PowerUserViewDocumentsAcceptedCandidatesScheduleInterviews.jsp)



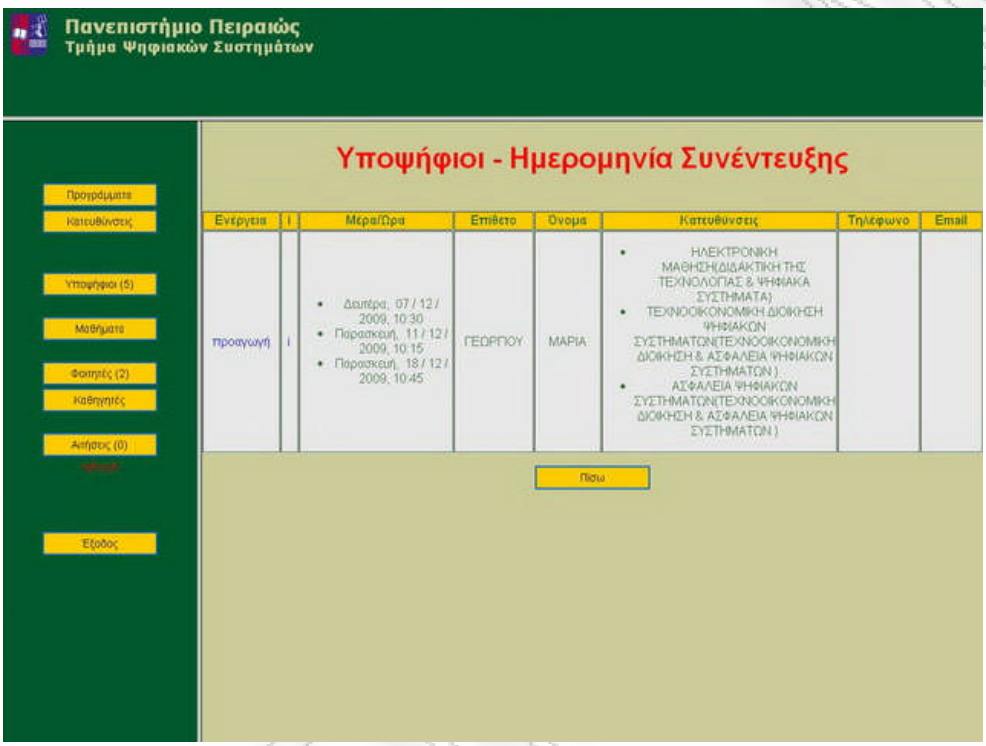
Εικόνα 61: PowerUserViewDocumentsAcceptedCandidatesScheduleInterviews.jsp

Το σύστημα μας δίνει τη δυνατότητα να επιλέξουμε μια από τις επόμενες εργάσιμες ημέρες, για τους επόμενους τρείς μήνες και κάποια ώρα από τις 7 το πρωί έως τις 22:45 το βράδυ για κάθε συνέντευξη.

```
<%
            for (CandidateDirection candidateDirection : candidateDirections) {
 %>


        <tr>
          <td align="right" width=200><b>Επιλογή μέρας/ώρας για την κατεύθυνση
<%=directionsById.get(candidateDirection.getDirectionId()).getName()%>:</b></td>

          <td align="center">
            <select name="interviewDate_<%=candidateDirection.getDirectionId()%>"
STYLE="width: 200px; text-align:center">
              <%

          for (Long workingDay : next100WorkingDays) {
            cal.setTimeInMillis(workingDay);
%>
              <option value="<%=workingDay%>"><%=daysByCalendarId.get(cal.get(Calendar.DAY_OF_WEEK)).getName(
)%>,  <%=getDateString(cal)%> </option>
              <%

              }
              %>
            </select>

          </td>
          <td width=200>
            <select name="interviewHourId_<%=candidateDirection.getDirectionId()%>"
STYLE="width: 100px; text-align:center">
              <%
        for (HourOfDayForInterview hourOfDayForInterview : hoursOfDayForInterview) {
              %>
              <option value="<%=hourOfDayForInterview.getId()%>"><%=hourOfDayForInterview.getName()%> </option>
              <%
}
 %>
```

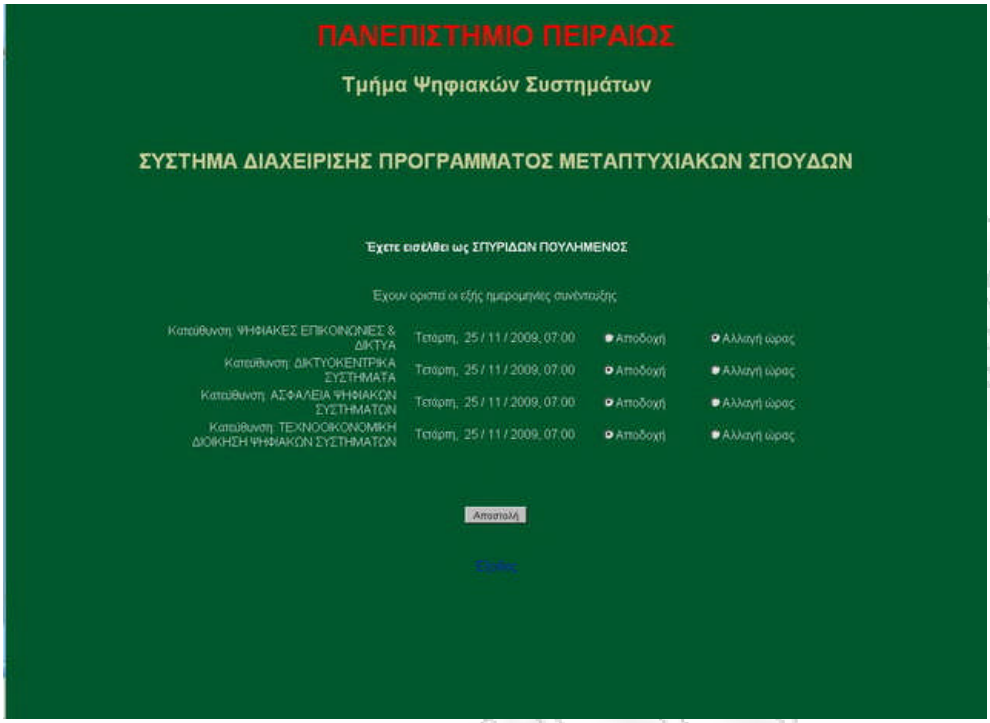
Με την αποθήκευση των ημερομηνιών για την πραγματοποίηση των συνεντεύξεων κάθε υποψηφίου, μεταφερόμαστε στην επόμενη κατηγορία, όπου ο διαχειριστής μπορεί να δει τους υποψήφιους για τους οποίους μόλις όρισε συνεντεύξεις.

- Δείτε υποψηφίους για τους οποίους έχει οριστεί μέρα συνέντευξης.(1)



Εικόνα 62: PowerUserViewScheduledInterviewsCandidates.jpg

Εάν ο υποψήφιος αποδεχτεί τις προτεινόμενες από την γραμματεία ημερομηνίες, τότε καταχωρείται στους υποψήφιους με οριστικοποιημένη ώρα συνέντευξης (PowerUserViewSetInterviewsCandidates.jsp).

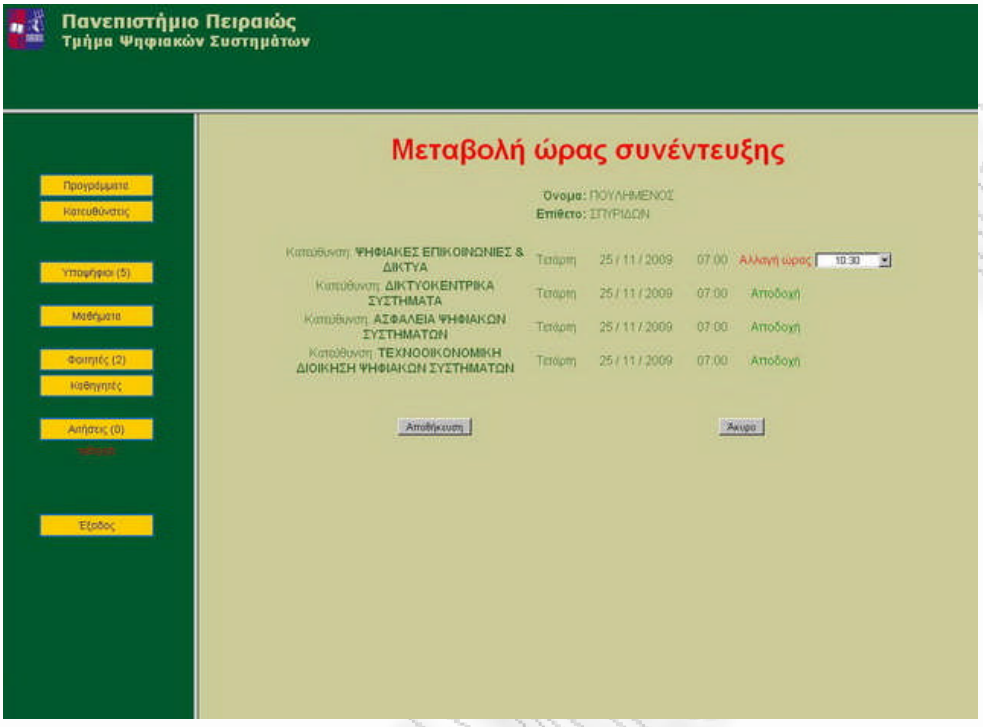


Εικόνα 63: Αποδοχή/Αλλαγή ώρας συνέντευξης απο τον υποψήφιο

Εικόνα 64: PowerUserViewSetInterviewsCandidates.jsp

Εάν ο υποψήφιος ζητήσει αλλαγή ώρας, τότε καταχωρείται στην αντίστοιχη κατηγορία και μέσα από την PowerUserRescheduleInterview.jsp η γραμματεία αλλάζει την ώρα συνέντευξης, χωρίς δυνατότητα αναπροσαρμογής.

- Δείτε υποψηφίους που έχουν ζητήσει μεταβολή της ώρας συνέντευξης.(0)



Εικόνα 65: PowerUserRescheduleInterview.jsp

Όταν πραγματοποιηθούν οι αλλαγές στις ώρες συνέντευξης που ζητήθηκαν, οι συγκεκριμένοι υποψήφιοι μεταβαίνουν στην κατηγορία *Υποψήφιοι – Προγραμματισμένες Συνεντεύξεις* (εικόνα 64).

Με την πραγματοποίηση των συνεντεύξεων η γραμματεία λαμβάνει από την Γενική Συνέλευση Ειδικής Σύνθεσης τα τελικά αποτελέσματα των επιτυχόντων υποψηφίων. Ανάλογα με αυτά τα αποτελέσματα, η γραμματεία επιλέγει *Αποδοχή ή Απόρριψη* για κάθε υποψήφιο (εικόνα 64).

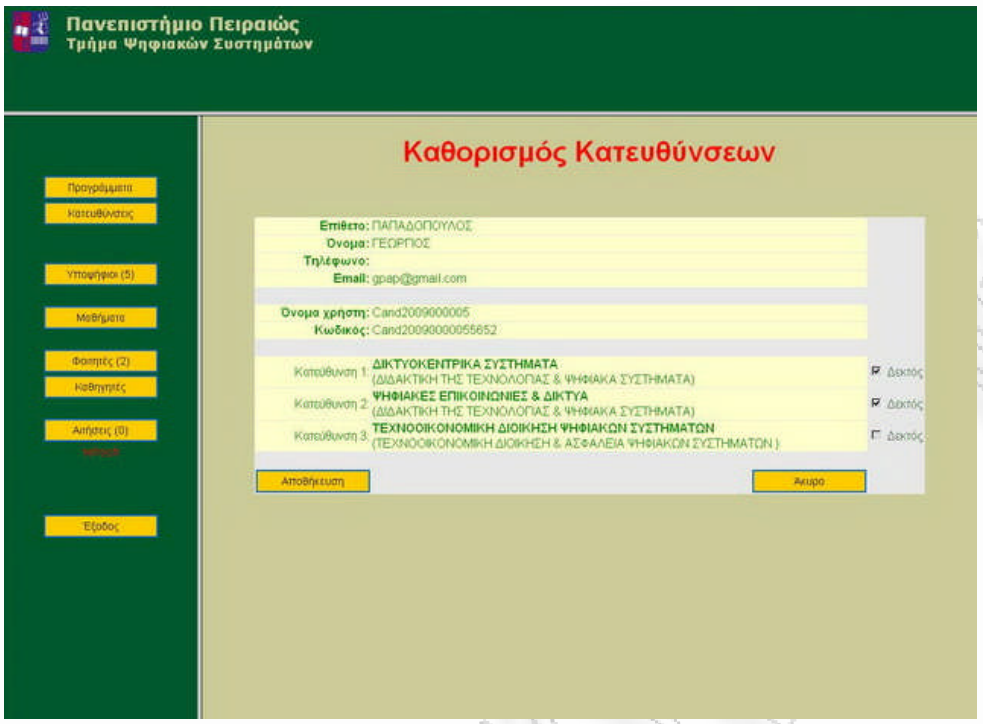
Εάν ο υποψήφιος απορριφθεί το modeId του παίρνει τιμή 6: *REJECTED_BY_GS,* καταχωρείται στην κατηγορία υποψηφίων που δεν έγιναν δεκτοί από την Γενική Συνέλευση και μπαίνοντας στο λογαριασμό του ενημερώνεται στον με κατάλληλο μήνυμα για την απόρριψή του.

Επιλέγοντας *Αποδοχή*, το modeId του υποψηφίου αλλάζει σε id:5, *ACCEPTED_BY_GS* καταχωρείται στην κατηγορία υποψηφίων που έγιναν δεκτοί από την Γενική Συνέλευση (PowerUserViewAcceptedByGsCandidates.jpg - εικόνα 66) και ενημερώνεται για αυτό με κατάλληλο μήνυμα στον λογαριασμό του (εικόνα 67).

Εικόνα 66: PowerUserViewAcceptedByGsCandidates.jsp

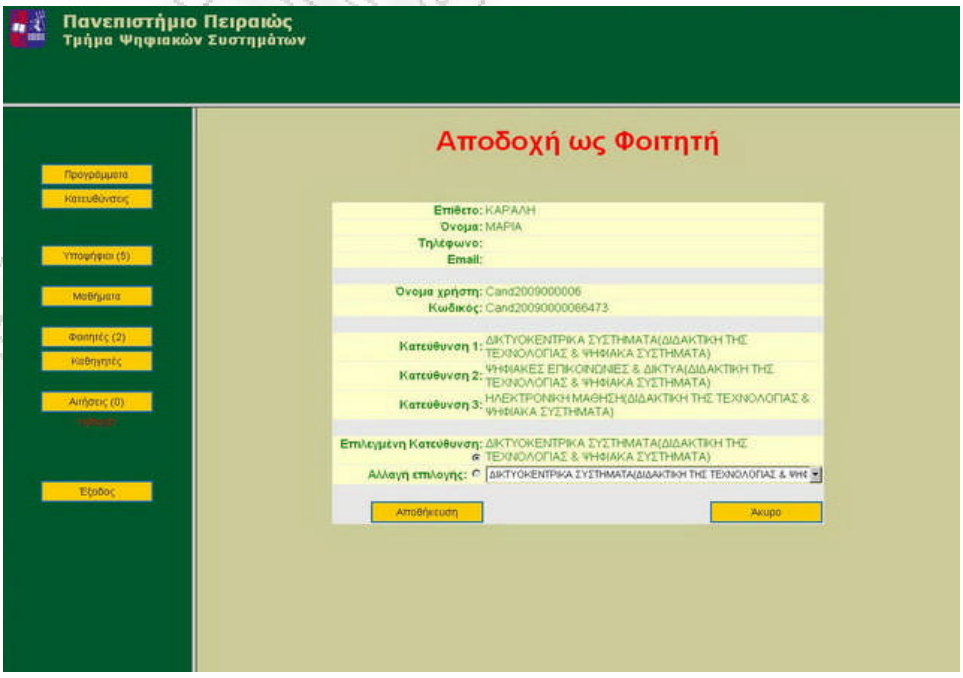
Εικόνα 67: Ενημέρωση υποψηφίου για την αποδοχή του

Όταν κάποιος υποψήφιος γίνει δεκτός, η γραμματεία θα καταχωρήσει τις κατευθύνσεις για τις οποίες έγινε αποδεκτός (εικόνα 68) και ο υποψήφιος θα ενημερωθεί από τον λογαριασμό του πως πρέπει να καταβάλει μια προκαταβολή διδάκτρων ώστε να οριστικοποιηθεί η αποδοχή του και να δημιουργηθεί ο φοιτητικός του λογαριασμός.



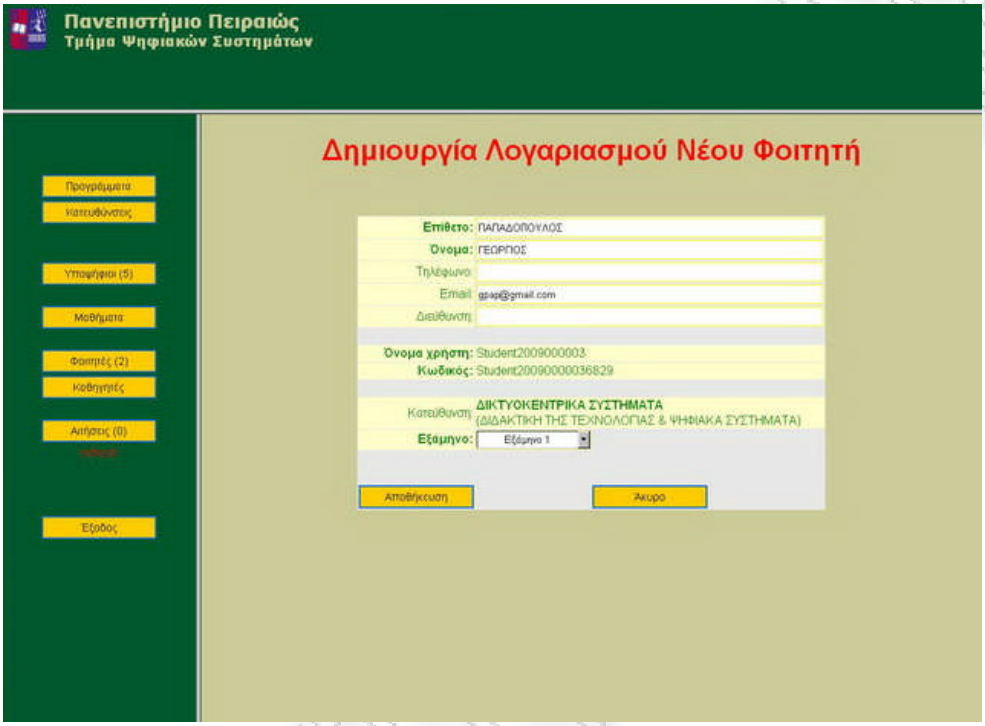
Εικόνα 68: PowerUserProcessAcceptedDirections.jsp

Το σύστημά μας είναι τόσο ευέλικτο ώστε στην συνήθη περίπτωση όπου ένας υποψήφιος γίνει δεκτός σε παραπάνω από μια κατευθύνσεις με την κατάθεση της προκαταβολής να οριστικοποιηθεί η αποδοχή του στην κατεύθυνση την οποία είχε δηλώσει πρώτη. Όμως για να υπάρχει ευελιξία στην τελική επιλογή, έχει προστεθεί και επιλογή αλλαγής της προεπιλεγμένης κατεύθυνσης εάν για κάποιο λόγο γίνει τελικά αποδεκτός σε μια από τις υπόλοιπες κατευθύνσεις που επέλεξε.



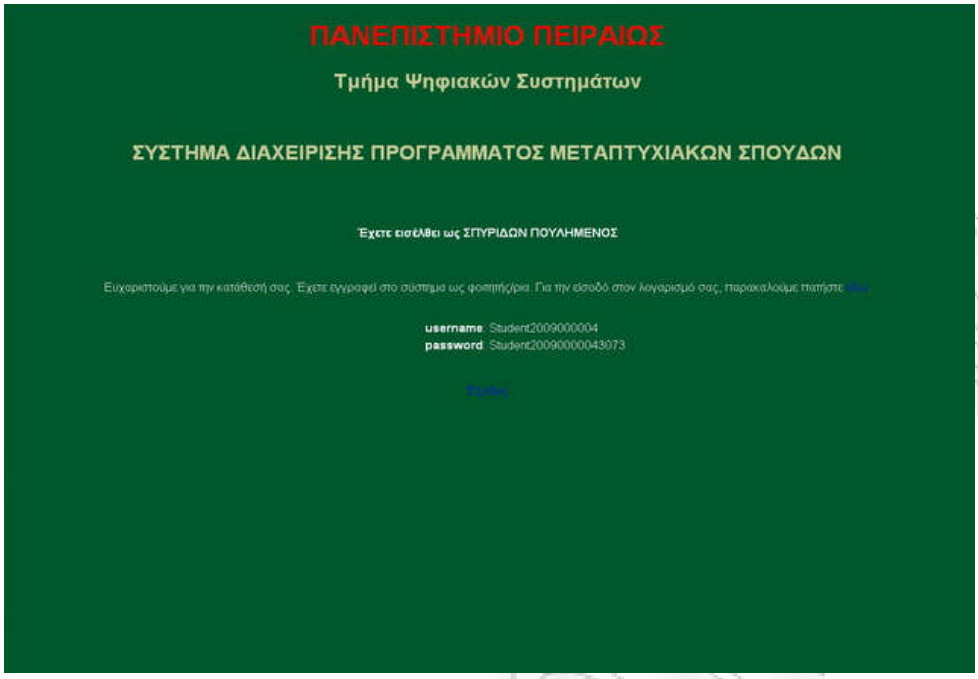
Εικόνα 69: PowerUserFinalizeCandidate.jsp

Όταν ο υποψήφιος καταθέσει το απαραίτητο χρηματικό ποσό, ο διαχειριστής αρχικά θα οριστικοποιήσει την αποδοχή του υποψηφίου με την τελική κατεύθυνση στην οποία γίνεται αποδεκτός (εικόνα 70) και έπειτα θα δημιουργήσει τον φοιτητικό λογαριασμό του υποψηφίου.



Εικόνα 70: PowerUserCreateNewStudent.jsp

Επιλέγοντας αποθήκευση, ο υποψήφιος καταχωρείται πλέον στην βάση ως student. Παρόλα αυτά δεν διαγράφεται από τον πίνακα Candidate, όμως το mode του πλέον αλλάζει σε id=7 που αντιστοιχεί σε mode : *HAS_BECOME_STUDENT*. Με τον τρόπο αυτό, όταν κάποιος φοιτητής γίνει δεκτός, μπαίνοντας στον λογαριασμό του ως υποψήφιος ακόμα, ενημερώνεται πως έχει γίνει δεκτός και καλείται να συνδεθεί στην τοποθεσία φοιτητών με το δικό του όνομα χρήστη και κωδικό πρόσβασης.

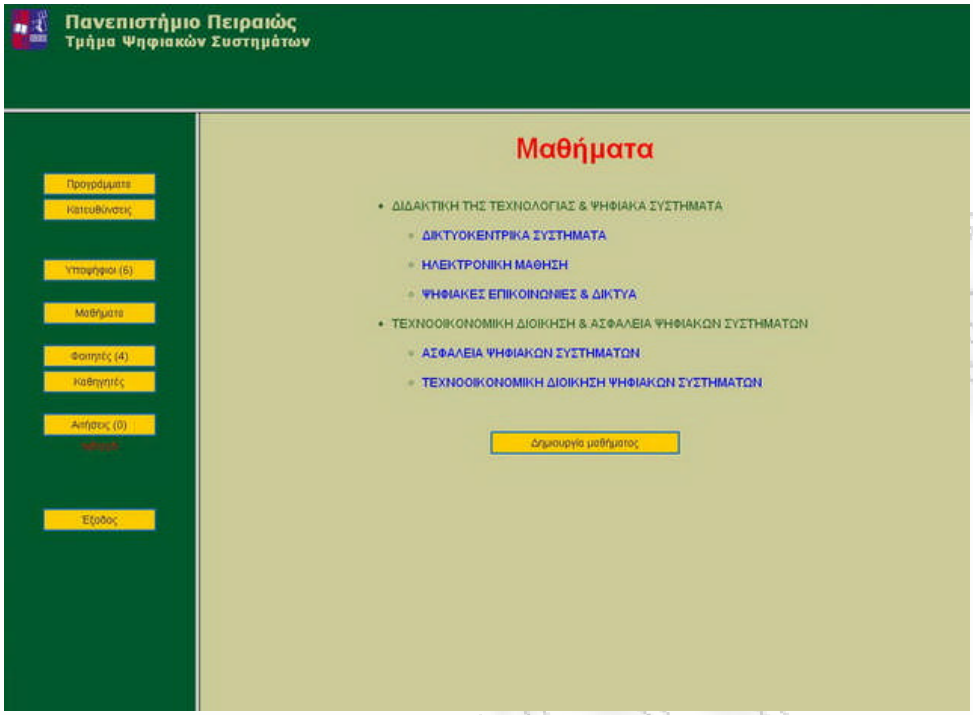


Εικόνα 71: Ενημέρωση υποψηφίου για τον φοιτητικό του λογαριασμό

Όλες αυτές οι λειτουργίες, από τη δημιουργία λογαριασμού του υποψηφίου μέχρι και την δημιουργία του λογαριασμού του πλέον ως φοιτητή, αποτελούν τη διαδικασία επιλογής υποψηφίων για τα μεταπτυχιακά προγράμματα του Τμήματος Ψηφιακών Συστημάτων, όπως την καταγράψαμε από την έρευνά μας και αναλύσαμε στο κεφάλαιο 3.

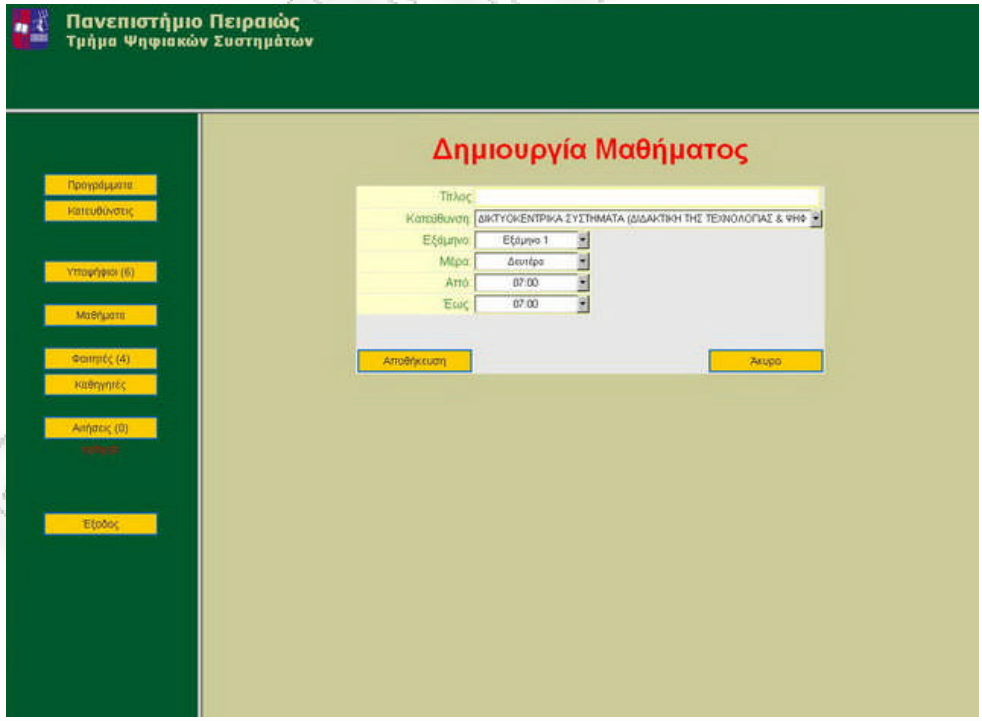
Στη συνέχεια της υλοποίησης των υπόλοιπων λειτουργιών της γραμματείας συναντάμε την δομή *Μαθήματα* (PowerUserViewCourses.jsp). Εδώ η γραμματεία έχει τη δυνατότητα να καταχωρήσει τα μαθήματα του προγράμματος σπουδών. Για λόγους παρουσίασης της εργασίας, δημιουργήσαμε τα μαθήματα που περιλαμβάνουν το νέα προγράμματα σπουδών των δύο μεταπτυχιακών προγραμμάτων.

Επιλέγοντας το κουμπί *Μαθήματα* εμφανίζονται στο main frame τα προγράμματα μεταπτυχιακών σπουδών και οι αντίστοιχες κατευθύνσεις τους καθώς και η επιλογή *Δημιουργία Μαθήματος*.



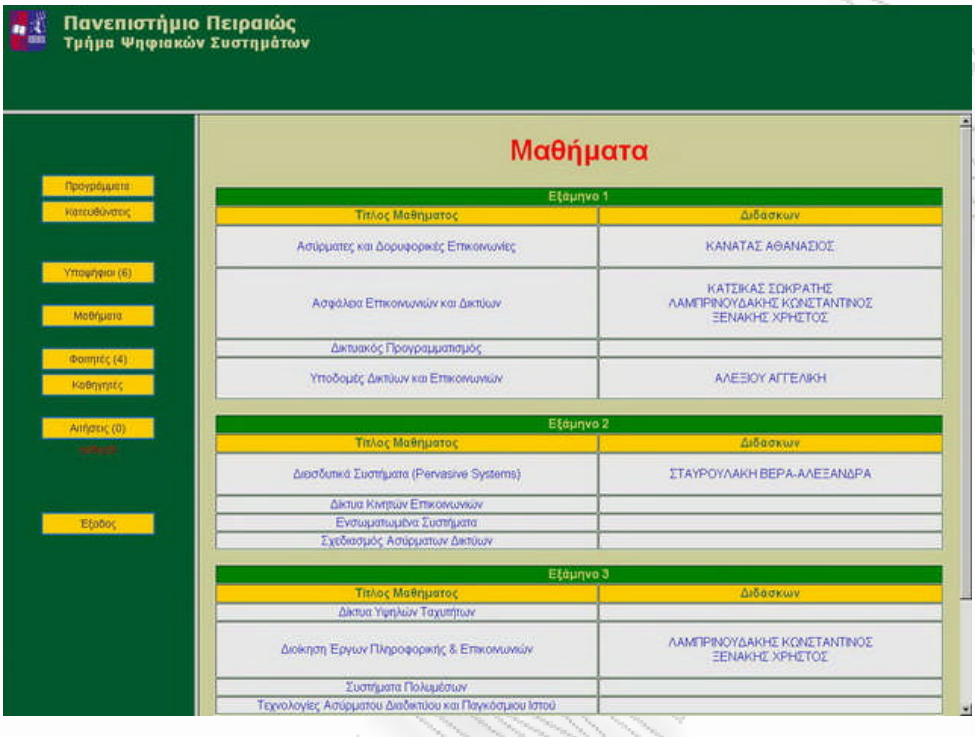
Εικόνα 72: PowerUserViewCourses.jsp

Επιλέγοντας την, μεταφερόμαστε στη σελίδα δημιουργίας νέου μαθήματος (PowerUserCreateCourse.jsp), όπου δηλώνουμε τον τίτλο του, την κατεύθυνση στην οποία αντιστοιχεί, το εξάμηνο του, την μέρα και την ώρα διεξαγωγής.



Εικόνα 73: PowerUserCreateCourse.jsp

Επιλέγοντας κάθε μια κατεύθυνση βλέπουμε τα μαθήματα που δημιουργήσαμε για την επιλεγμένη κατεύθυνση (PowerUserViewCoursesByDirId.jsp).



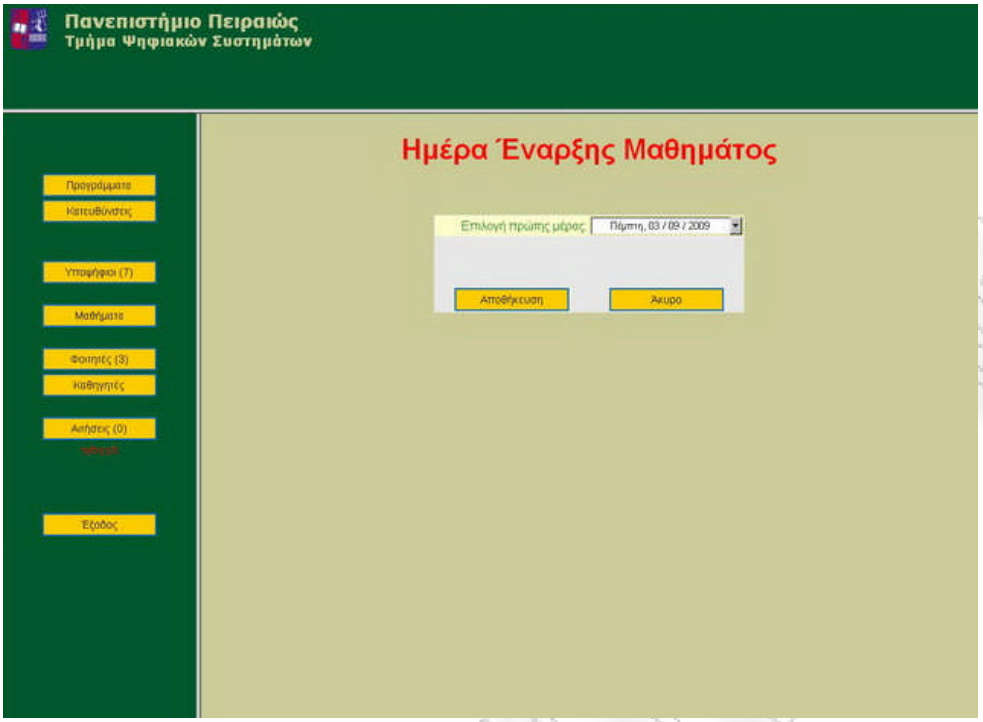
Εικόνα 74: PowerUserViewCoursesByDirId.jsp

Επιλέγοντας κάθε μάθημα αρχικά έχουμε δύο επιλογές. :

- Επεξεργασία Μαθήματος και,
- Ορισμό πρώτης μέρας παρακολούθησης για το τρέχον ακαδημαϊκό έτος

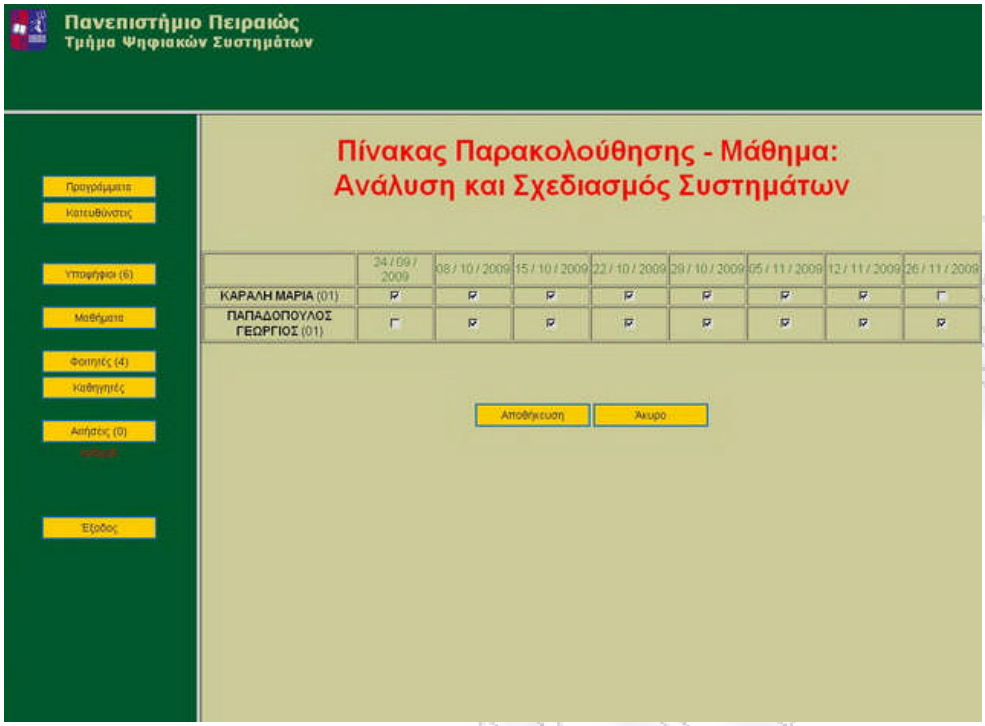
Στην επεξεργασία μαθήματος μπορούμε να αλλάξουμε τον τίτλο, την κατεύθυνση το εξάμηνο και την μέρα και ώρα διεξαγωγής του. Τονίζεται ότι είναι καίριας σημασίας, η όποια τροποποίηση αφορά κάποιο μάθημα θα πρέπει να γίνεται πριν συνδεθεί με δεδομένα (φοιτητές, βαθμολογίες κλπ).

Με την επιλογή *Ορισμός πρώτης μέρας παρακολούθησης για το τρέχον ακαδημαϊκό έτος*, ορίζουμε ουσιαστικά την ημέρα έναρξης των μαθημάτων για το συγκεκριμένο μάθημα.



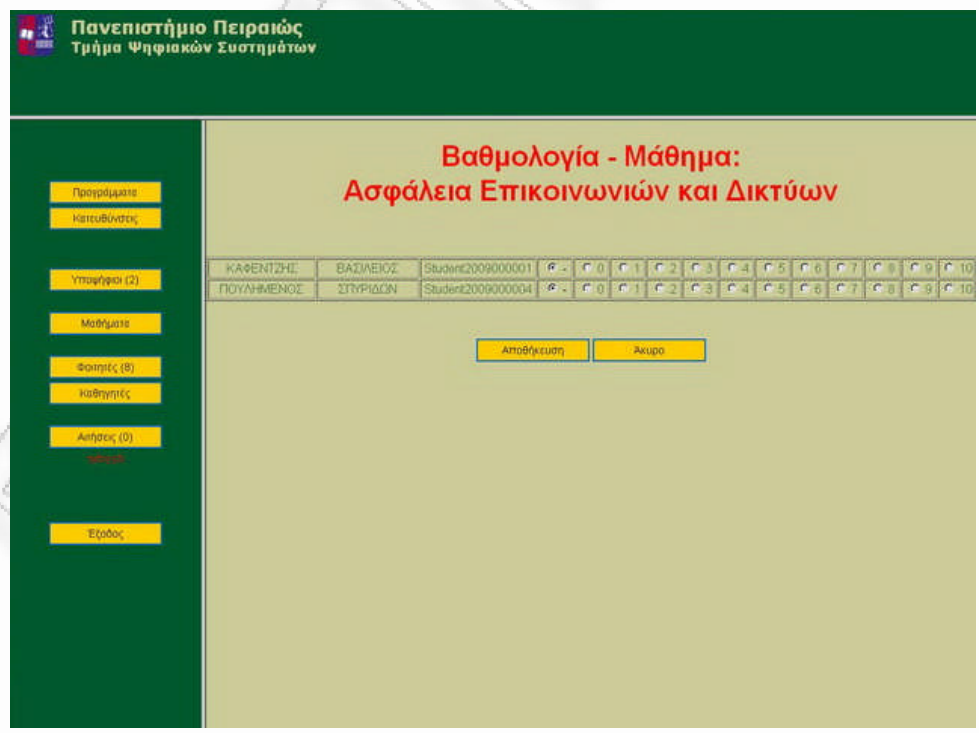
Εικόνα 75: PowerUserCreateFirstAttendanceDayForCourse.jsp

Αφού οριστεί η μέρα έναρξης του κάθε μαθήματος, η γραμματεία μπορεί να προσθέτει την επόμενη μέρα διεξαγωγής του μαθήματος, όπου το σύστημα προτείνει τις επόμενες επτά πιθανές ημέρες (εάν το μάθημα γίνεται Τετάρτη, για την προσθήκη νέας μέρας παρακολούθησης το σύστημα προτείνει τις επόμενες επτά Τετάρτες). Ο λόγος είναι ότι για κάποια απρόσμενη αιτία μπορεί να μην διεξαχθεί κάποιο μάθημα την αναμενόμενη ημέρα. Έτσι δώσαμε στο σύστημα τη δυνατότητα να γνωρίζει πότε έγινε πράγματι κάποιο μάθημα και για αυτήν την παρακολούθηση να προστεθούν και οι αντίστοιχες παρουσίες των φοιτητών τις οποίες λαμβάνει από τους διδάσκοντες με τις υπογραφές των φοιτητών και να τις καταχωρεί στο σύστημα, στον *Πίνακα Παρακολουθήσεων*.



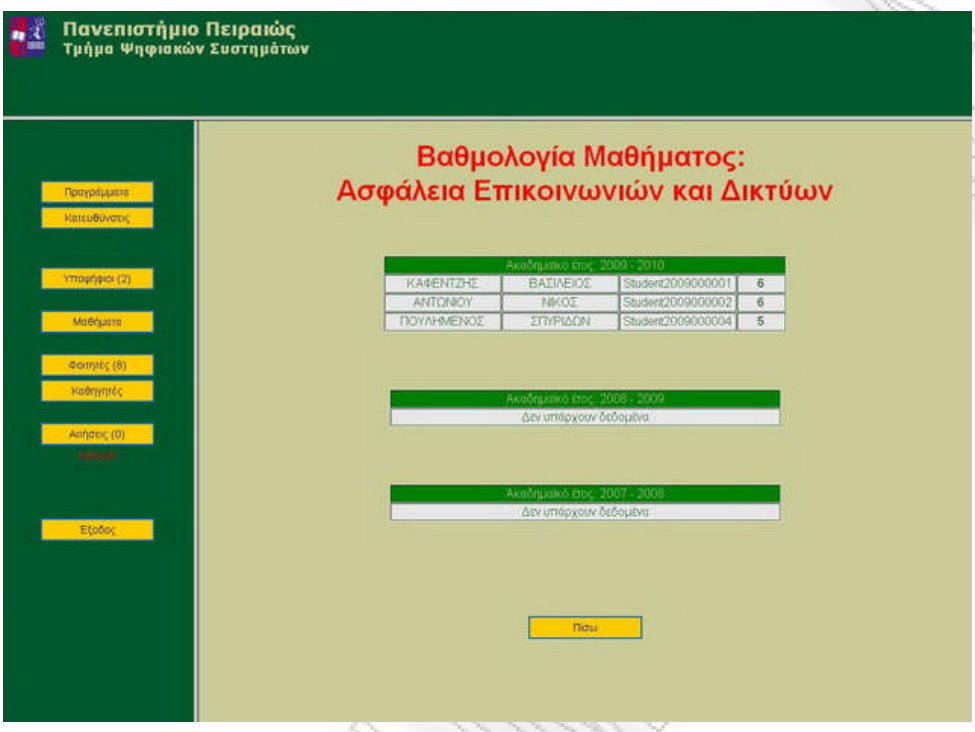
Εικόνα 76: PowerUserAttendanceTable.jsp

Τέλος, με την επιλογή *Κλείσιμο Παρακολούθησης για αυτό το εξάμηνο* το modeId του μαθήματος παίρνει τιμή 2, δηλαδή CLOSED, και μόνο τότε η γραμματεία μπορεί να καταχωρήσει τις βαθμολογίες που εκδίδουν οι διδάσκοντες.



Εικόνα 77: PowerUserViewScoreForCourse.jsp

Αφού καταχωρήσει η γραμματεία την βαθμολογία κάθε φοιτητή, μπορεί να επιλέξει *Κλείσιμο μαθήματος για αυτό το εξάμηνο* που θα σημαίνει ότι το μάθημα παύει να είναι ενεργό για το τρέχον ακαδημαϊκό έτος.



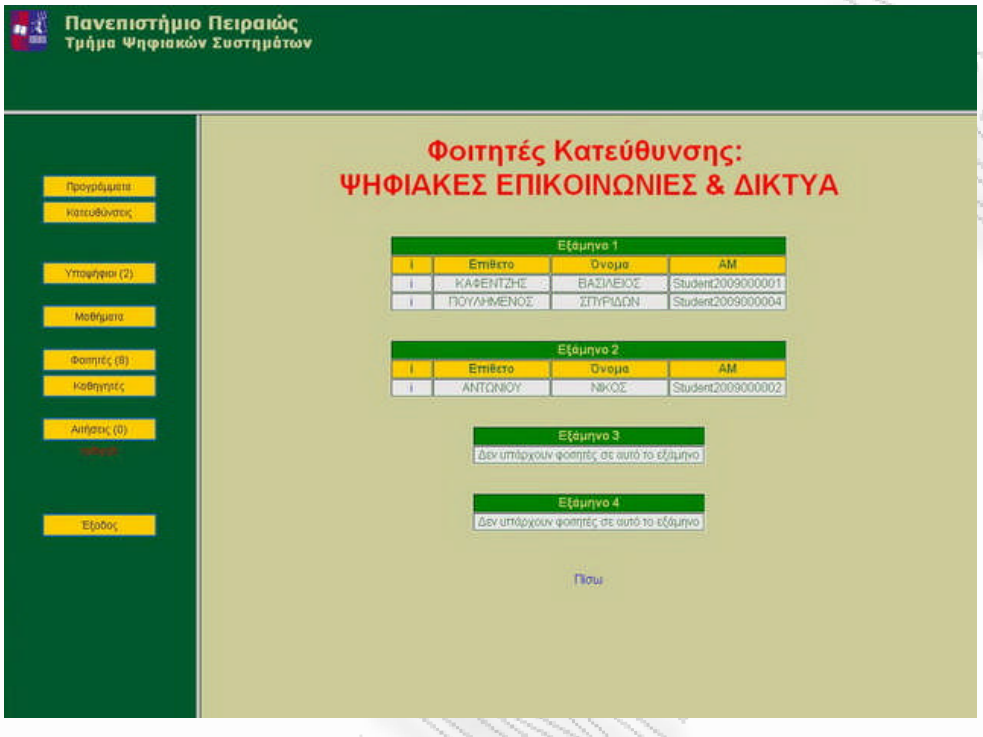
Εικόνα 78: PowerUserViewScoreForCourseClosed.jsp

Συνοψίζοντας, σε ότι αφορά τα μαθήματα οι λειτουργίες που υλοποιούνται είναι οι εξής:

- Δημιουργία μαθήματος (επεξεργασία προαιρετικά)
- Ορισμός πρώτης μέρας παρακολούθησης
- Προσθήκη παρουσιών είτε με το πέρας της εκάστοτε παρακολούθησης, είτε κάποια άλλη στιγμή, υποχρεωτικά όμως πριν το κλείσιμο της παρακολούθησης
- Κλείσιμο παρακολούθησης για το τρέχον εξάμηνο και δυνατότητα προσθήκης βαθμολογίας
- Κλείσιμο μαθήματος για το τρέχον εξάμηνο

Στη συνέχεια της υλοποίησης των λειτουργιών της γραμματείας, είναι η δομή που αφορά τους φοιτητές. Επιλέγοντας αρχικά την επιλογή *Φοιτητές* (PowerUserViewStudents.jsp) εμφανίζονται στην οθόνη του διαχειριστή τα μεταπτυχιακά πργράμματα με τις αντίστοιχες κατευθύνσεις τους και σε παρένθεση ένας αριθμός. Ο

αριθμός αυτός υποδηλώνει το πλήθος των φοιτητών σε κάθε κατεύθυνση. Επιλέγοντας μια κατεύθυνση (PowerUserViewStudentsByDirId.jsp),



Εικόνα 79: PowerUserViewStudentsByDirId.jpg

εμφανίζονται οι εγγεγραμμένοι σε αυτήν φοιτητές. Πατώντας στις πληροφορίες (i) κάθε φοιτητή μεταφερόμαστε στο προφίλ του (PowerUserViewStudentProfile.jpg).

Εικόνα 80: PowerUserViewStudentProfile.jsp

Σε αυτό το σημείο, ο διαχειριστής είτε μπορεί να δει τα μαθήματα και την βαθμολογία του φοιτητή, είτε να πραγματοποιήσει αλλαγές στην βαθμολογία εάν για παράδειγμα έχει προηγηθεί κάποιο λάθος στην αρχική καταχώρηση. Στην παρακάτω εικόνα διακρίνεται με πορτοκαλί χρώμα ο πίνακας που αφορά το τρέχον εξάμηνο το οποίο διανύει ο φοιτητής.

Εικόνα 81: PowerUserChangeStudentScores.jsp

Η πιο σημαντική ενέργεια όμως είναι η *Προαγωγή στο επόμενο εξάμηνο*. Θεωρητικά αλλά και πρακτικά όταν τελειώσει για παράδειγμα το 1^{ο} εξάμηνο, με την επιλογή αυτή ο φοιτητής περνάει στο επόμενο. Αυτό πρακτικά για την βάση δεδομένων σημαίνει ότι το semesterId του φοιτητή θα αλλάξει και θα πάρει την επόμενη τιμή. Με τον τρόπο αυτό, έχουμε ένα καλά δομημένο και οργανωμένο αρχείο φοιτητών.

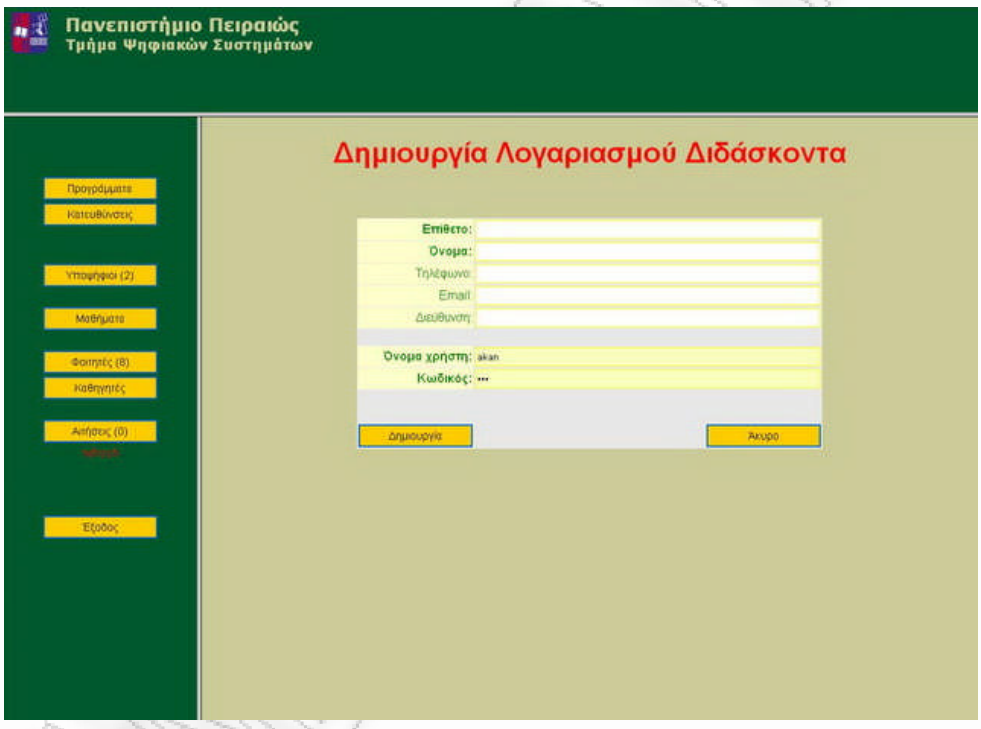
Όπως είναι ήδη εμφανές, τα μαθήματα και οι καταστάσεις τους είναι άμεσα συνδεδεμένα με τους φοιτητές. Έτσι θα πρέπει να δοθεί μεγάλη προσοχή στο κλείσιμο των μαθημάτων για τα εξάμηνα και η προαγωγή του φοιτητή στο επόμενο εξάμηνο να γίνεται μόνο εφόσον συμβαδίζει με την πραγματική ροή του προγράμματος σπουδών αλλά και με το κλείσιμο του μαθήματος εντός του συστήματος.

Άλλη μια λειτουργία της γραμματείας είναι η διαχείριση των καθηγητών. Όπως ίσως παρατηρεί κανείς, στις παραπάνω εικόνες που αφορούν τα μαθήματα, εκτός

από τον τίτλο του μαθήματος υπάρχει και η στήλη διδάσκον. Στην στήλη αυτή αναφέρεται ο εισηγητής καθηγητής κάθε μαθήματος. Η αντιστοίχηση όμως κάθε μαθήματος με τους αντίστοιχους διδάσκοντες γίνεται από την επιλογή *Καθηγητές* (PowerUserViewProfessors.jsp).

Στην οθόνη που εμφανίζεται αρχικά μας ενημερώνει το σύστημα πως δεν υπάρχουν εγγραφές. Για λόγους παρουσίασης και επεξήγησης έχουμε προσθέσει τα ονόματα του διδακτικού προσωπικού όπως αυτό είναι αναρτημένο στην ιστοσελίδα του τμήματος.

Για την δημιουργία ενός λογαριασμού καθηγητή επιλέγουμε το αντίστοιχο button το οποίο μας μεταφέρει στη σελίδα δημιουργίας του προφίλ του καθηγητή.(PowerUserCreateProfessors.jsp).

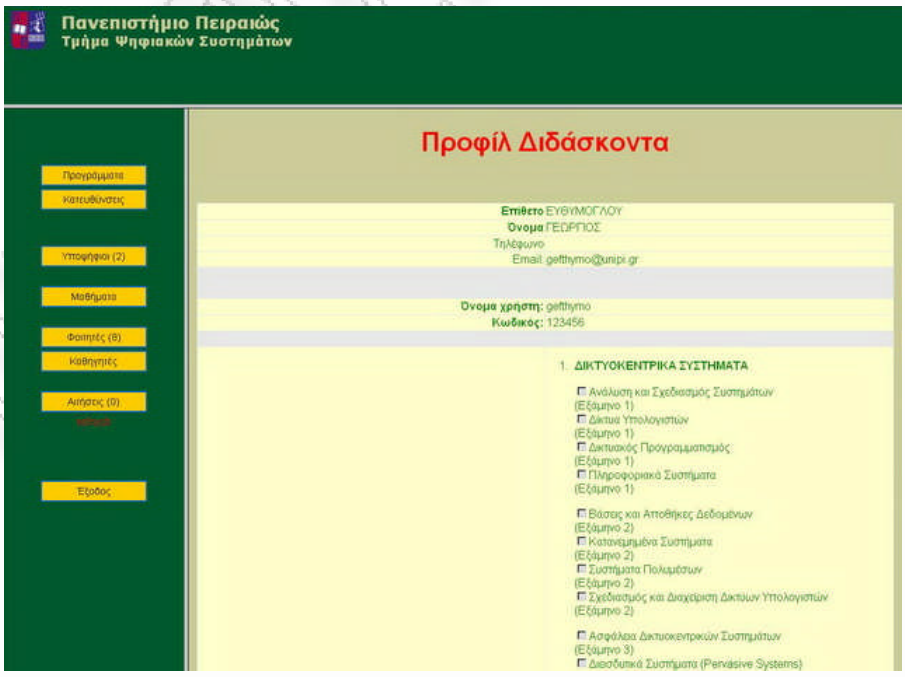


Εικόνα 82: PowerUserCreateProfessors.jpg

Όπως σε όλες τις περιπτώσεις δημιουργίας λογαριασμών, έτσι και εδώ η γραμματεία είναι αυτή που δημιουργεί τον λογαριασμό για τους εκπαιδευτικούς. Συμπληρώνοντας τουλάχιστον τα απαραίτητα στοιχεία και πατώντας αποθήκευση δημιουργείται ο λογαριασμός του αντίστοιχου διδάσκοντα. Έτσι έχουμε μια λίστα με όλο το εκπαιδευτικό προσωπικό του τμήματος.

Εικόνα 83: PowerUserViewProfessors.jsp

Επιλέγοντας τις πληροφορίες (*i*) δίπλα από το όνομα κάθε εκπαιδευτικού μεταφερόμαστε στην σελίδα του λογαριασμού του, όπου εκτός από τα στοιχεία του βλέπουμε τα μεταπτυχιακά προγράμματα με τις αντίστοιχες κατευθύνσεις και τα μαθήματα που αντιστοιχούν σε κάθε κατεύθυνση. Ο διαχειριστής επιλέγει στα αντίστοιχα checkboxes τα μαθήματα που διδάσκει ο κάθε εκπαιδευτικός και έτσι γίνεται η αντιστοίχηση μαθημάτων και διδασκόντων. Η πληροφορία αυτή είναι διαθέσιμη οπουδήποτε μέσα στο σύστημα όπου συμμετέχουν σαν δομή τα μαθήματα.



Εικόνα 84: PowerUserViewProfessorProfile.jsp

Τέλος, επεκτείναμε τις λειτουργίες της γραμματείας προσθέτοντας μια ακόμα δυνατότητα, την δυνατότητα αποστολής (από τους φοιτητές) και παραλαβής (από την γραμματεία) αιτήσεων. Θα φτάνουν δηλαδή στη βάση, μηνύματα xml τα οποία θα περιέχουν πληροφορίες για τις αιτήσεις. Ποιος την ζήτησε, πόσα αντίτυπα επιθυμεί και τι είδους πιστοποιητικό ζήτησε. Τα πιστοποιητικά που θα αιτούνται οι φοιτητές θα είναι δύο ειδών: α) βεβαιώσεις σπουδών και β) αναλυτική βαθμολογία. Δημιουργήσαμε έτσι τρία απαραίτητα αντικείμενα: α)StudentReqType.java, β)StudentRequestMode.java και γ) StudentRequest.java. Το StudentReqType αντιστοιχεί στον πίνακα student_request_type και είναι οι δύο τύποι αιτήσεων (Βεβαίωση σπουδών, Αναλυτική βαθμολογία). Το StudentRequestMode αφορά το αν οι αιτήσεις είναι NEW ή OLD. Οι αιτήσεις που εκτυπώνονται γίνονται αυτομάτως OLD. Το StudentRequest περιέχει όλα τα απαραίτητα στοιχεία για την αίτηση. Ας δούμε με ποιόν τρόπο δημιουργείται το xml μήνυμα που φτάνει στη βάση, και πως διαβάζεται στη συνέχεια.

Στο αντικείμενο Database.java δημιουργήσαμε όπως προαναφέραμε κάποια Prepared Statements. Για της αιτήσεις χρειαζόμαστε τα *StudentRequestCreateNewEntry* (δημιουργούμε New Entry), *StudentRequestGetAll* (παίρνουμε την πληροφορία) και *StudentRequestUpdate* (κάνουμε Update, συγκεκριμένα μόνο το modeId).

Έπειτα δημιουργούμε την StudentRequestCreateNewEntry,

```
public void studentRequestCreateNewEntry(int modeId, int typeId, int cntOfCopies, String studentU-
niqueId)
```

η οποία δέχεται σαν ορίσματα το modeId, typeId, cntOfCopies και StudentUniqueId και δημιουργείται το xml που καταχωρείται στην βάση.

```
p.setString(2, Tools.createStudentRequestXML(typeId, cntOfCopies, studentUniqueId));
```

Η createStudentRequestXML είναι μια μέθοδος static γενικής χρήσης που δημιουργήθηκε στο αντικείμενο Tools.java, το οποίο περιέχει γενικά τέτοιες μεθόδους γενικής χρήσης, που δεν είναι απαραίτητο να βρίσκονται σε κάποιο αντικείμενο. Η createStudentRequestXML λοιπόν, δέχεται σαν arguments τα typeId, cntOfCopies και StudenUniqueId και ετοιμάζει έναν πίνακα από Elements τον οποίο τροφοδοτούμε στην createXML (return *createXML*(“request”,elements)). Και η createXML είναι γενικής χρήσης και περιέχει δύο στοιχεία.

```
public static String createXml(String mainNode, String [][] elements)
```

Το όνομα του mainNode (<request> </request>) και έναν πίνακα με τα elements δύο διαστάσεων. Στη συνέχεια, γίνεται parsing αυτού του πίνακα και δημιουργείται το document και ο mainNode σαν element.

```
Document document = documentBuilder.newDocument();
Element re = document.createElement(mainNode);
```

Με την χρήση της appendChild(createTextNode)

```
re.appendChild(createTextNode(elements[i][0], elements[i][1], document));
```

δημιουργείται ένα element και το ενσωματώνει(append) σε ένα έγγραφο(document).

```
public static Element createTextNode(String nodeName, String nodeText, Document document) {
    Element em = document.createElement(nodeName);//.appendChild(document.createTextNode("el2"));
    em.appendChild(document.createTextNode(nodeText));
    return em;
```

Με αυτόν τον τρόπο δημιουργούμε TextNodes που περιέχουν τις μεταβλητές typeId, cntOfCopies και StudentUniqueId με βάση των δισδιάστατο πίνακα. Η πρώτη διάσταση αφορά το όνομα του TextNode και η δεύτερη το περιεχόμενο του.

```
String [][] elements = new String[][]{
            {"typeId", typeId + ""}
            {"cntOfCopies", cntOfCopies + ""},
            {"studentUniqueId", studentUniqueId}
```

Το XML που δημιουργείται θα έχει αυτή τη μορφή:

<request>

<typeId> περιεχόμενο <typeId/>

< cntOfCopies > περιεχόμενο <cntOfCopies/>

< studentUniqueId > περιεχόμενο <studentUniqueId/>

</request>

Έτσι δημιουργούμε το XML με την πληροφορία για την αίτηση που ζητήθηκε. Για να το διαβάσουμε, η μέθοδος

```
public List<StudentRequest> studentRequestGetAll() {
```

πρέπει να επιστρέψει StudentRequest.

```
PreparedStatement p =  dBStatements[dBStatementsEnum.StudentRequestGetAll.ordinal()];

    List<StudentRequest> toReturn = null;
```

Αυτή (στο StudentRequest.java) περιέχει έναν constructor και ένα ResultSet. Από την βάση, παίρνουμε το xmlDoc

```
String xml = rs.getString("xmlDoc");
```

το οποίο είναι τύπου String και το κάνουμε parse.

```
parseXML(xml);
```

Αυτό γίνεται μέσα στην,

```
public void parseXML(String xml)
```

με έναν Stream XML parser

```
XMLStreamReader parser = factory.createXMLStreamReader(new StringReader(xml));
```

Αυτός ο parser, παράγει events τα οποία γνωρίζουμε τι τύπου είναι . (START_ELEMENT ή CHARACTERS, διαβάζει δηλαδή text). Σε αυτό το σημείο θέτουμε ποιος είναι ο node ο οποίος διαβάζουμε

```
currentNode = parser.getName().toString();
```

Και γνωρίζοντας ποιος είναι ο node ο οποίος διαβάζουμε, κάνουμε τους παρακάτω ελέγχους: αν είναι τύπου typeId, να παρσάρει το Text και να το κάνει Integer. Ομοίως για τους άλλους ελέγχους.

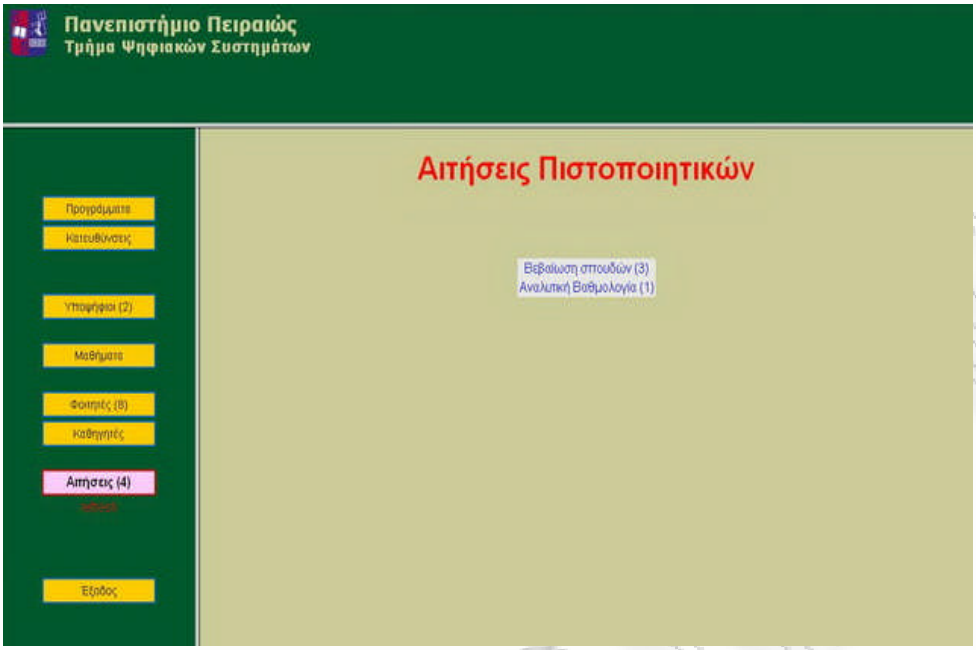
```
if(currentNode.equalsIgnoreCase("typeId"))
            {
                this.typeId = Integer.parseInt(parser.getText());
            }
            else if(currentNode.equalsIgnoreCase("cntOfCopies"))
            {
                this.cntOfCopies = Integer.parseInt(parser.getText());
            }
            else if(currentNode.equalsIgnoreCase("studentUniqueId"))
            {
                this.studentUniqueId = parser.getText();
            }
```

Οπότε, επιστρέφεται πίσω, μια δομή, StudentRequest η οποία περιέχει τις μεταβλητές που θέλουμε.

```
public class StudentRequest {

   private int id;
   private int modeId;
   private int typeId;
   private int cntOfCopies;
   private String studentUniqueId;
```

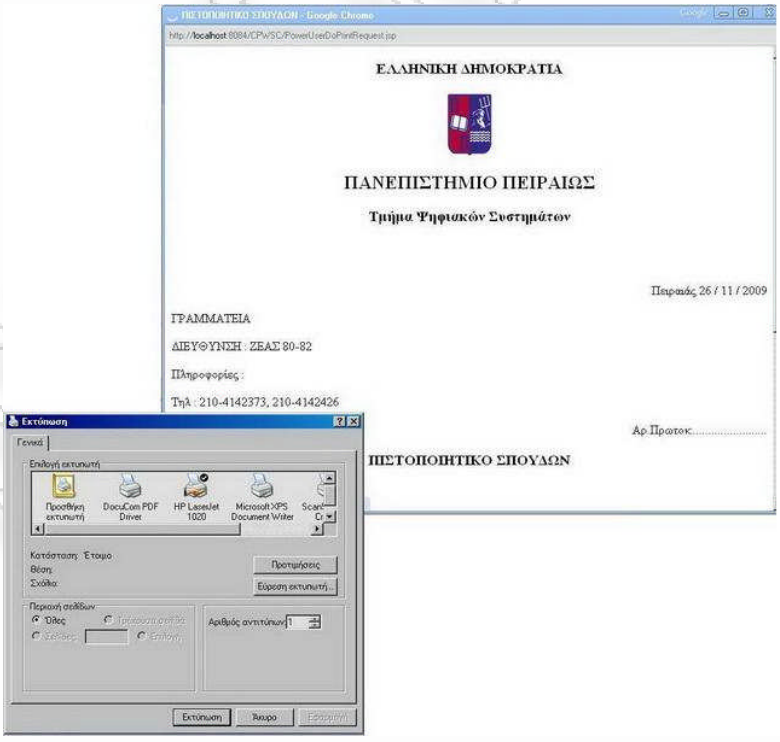
Κατ' αυτόν τον τρόπο το XML που φτάνει στη βάση, δημιουργείται και διαβάζεται εσωτερικά μέσα στις δομές χωρίς να φαίνεται προς το Web Service.

Έτσι λοιπόν, οι η γραμματεία όταν λαμβάνει αιτήσεις από τους φοιτητές, και μπορεί να τις δει επιλέγοντας το αντίστοιχο button *Αιτήσεις* (PowerUserViewRequests.jsp). Επίσης μπορεί να βλέπει πόσες νέες αιτήσεις έχει σε εκκρεμότητα. Με την επιλογή refresh, ανανεώνεται όλο το frame και έτσι μπορεί να βλέπει κάθε φορά μόνο τις νέες αιτήσεις που εκκρεμούν.



Εικόνα 85: PowerUserViewRequests.jsp

Επιλέγοντας ο χρήστης της γραμματείας τον αντίστοιχο τύπο αίτησης, βλέπει πόσες και ποιος ζήτησε αιτήσεις. Επιλέγοντας εκτύπωση (PowerUserPrintRequest.jsp) προβάλλεται το όνομα του φοιτητή για τον οποίο θα τυπωθεί η αίτηση, και πατώντας *Εκτύπωση* ανοίγει ένα παράθυρο προεπισκόπησης της αίτησης με επιλογή για άμεση εκτύπωση (PowerUserDoPrintRequest.jsp)



Εικόνα 86: PowerUserDoPrintRequest.jsp

Η παραπάνω προεπισκόπηση αφορά μια αίτηση για πιστοποιητικό σπουδών. Η μορφοποίηση υλοποιήθηκε με την προσθήκη διάφορων HTML στοιχείων. Τα δεδομένα που αφορούν τον χρήστη λαμβάνονται από την βάση και εμφανίζονται αυτόματα ενώ εμφανίζεται αυτόματα και η ημερομηνία εκτύπωσης.

```
<body onload="window.print()">

    <%if(studentRequest.getTypeId() == 0)
{
        %>



        <h3 align="center">ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ</h3>
        <p align="center"> <img align="middle" src="logo_new.gif" width="60" height="85"
alt="logo"></p>
        <h2 align="center">ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ</h2>
        <h3 align="center">Τμήμα Ψηφιακών Συστημάτων</h3>
        <br/>
…….
…….
  <%
    }
else
{

         %>
…….
…….
          <p> Ο φοιτητής με τα παρακάτω στοιχεία: <br/></p>
        <p>ONOMA.....: <b><%=student.getFirstName() %></b> </p>
        <p>ΕΠΙΘΕΤΟ.....: <b><%=student.getLastName() %></b> </p>

 <%
    Semester semester;
    for(int i=0;i<semesters.size();i++)
    {
       semester = semesters.get(i);

        courses = coursesByDidSid.get(student.getDirectionId() + "_" + semester.getId());
```

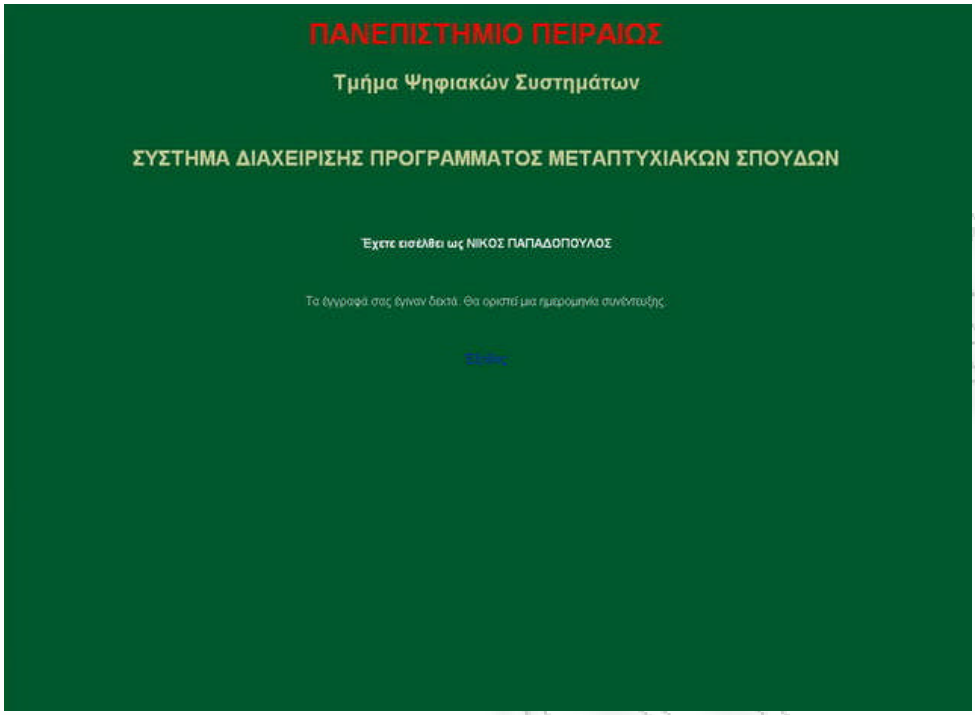
```
        %>

        <table align="center" border="1">
          <tr bgcolor="<%= semester.getId() == student.getSemesterId() ? "orange" : "grey" %>">
            <th align="center" colspan="3" width=650
style="color:black"><b><%=semester.getName() %></b>

            </th>
          </tr>
```

Έτσι ολοκληρώνονται οι λειτουργίες της γραμματείας, προσφέροντας έλεγχο, οργάνωση και πραγματική διαχείριση στην υπηρεσία. Παρακάτω θα δούμε τις λειτουργίες και την επικοινωνία που έχουν με την γραμματεία οι υπόλοιποι clients του συστήματος, υποψήφιοι, φοιτητές και καθηγητές.

4.4.3 Client Interface Υποψηφίου (candidate)

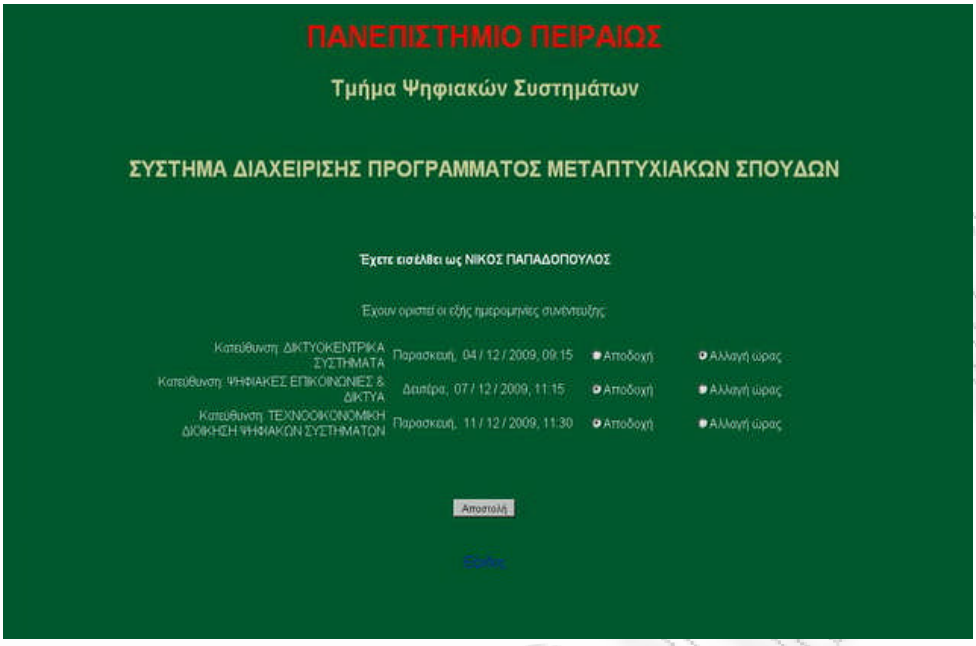
Έχουμε δει ήδη την συμπεριφορά και τον ρόλο των υποψηφίων γενικά στο σύστημα. Δυστυχώς λόγω γραφειοκρατικών κωλυμάτων είναι αδύνατο να αυτοματοποιηθούν σε απόλυτο βαθμό οι ενέργειες των υποψηφίων. Στο δικό μας σύστημα έγινε μια προσπάθεια για όσο μέγιστη δυνατή αυτοματοποίηση των ενεργειών των υποψηφίων.

Όπως έχουμε ήδη πει, ο υποψήφιος με την κατάθεση της υποψηφιότητάς του, αποκτά από τη γραμματεία ένα λογαριασμό χρήστη. Μπαίνοντας από την αντίστοιχη διεύθυνση http:localhost:8084/CPWSC/CandidateLogin.jsp (εικόνα 53), εισάγει τα στοιχεία του και εισέρχεται πλέον στον προσωπικό του λογαριασμό. Σκοπός μας είναι να ενημερώνεται ο υποψήφιος συνεχώς μέσα από το λογαριασμό του, για την κατάσταση της αξιολόγησής του. Έτσι, κάθε φορά που η γραμματεία προάγει στην επόμενη κατηγορία τους υποψήφιους, αυτοί θα βλέπουν το αντίστοιχο μήνυμα στο λογαριασμό τους. Για παράδειγμα, έστω ότι ο υποψήφιος με όνομα χρήστη Cand2009000008 έχει καταθέσει τα έγγραφά του και αυτά έχουν γίνει αποδεκτά, οπότε είναι σε αναμονή ορισμού συνέντευξης για τις κατευθύνσεις που επέλεξε. Μπαίνοντας στο λογαριασμό του θα ενημερωθεί ακριβώς για αυτήν την κατάσταση.



Εικόνα 87: CandidateLoginSuccessfull.jsp - Προβολή κατάστασης αξιολόγησης

Ομοίως, κάθε φορά που προάγεται από τη γραμματεία στις επόμενες κατηγορίες μέχρι την αποδοχή ή απόρριψή του, θα ενημερώνεται απ' τον λογαριασμό του για την κατάσταση στην οποία βρίσκεται η αξιολόγησή του. Μια χαρακτηριστική λειτουργικότητα που προσφέρει το σύστημα και στον υποψήφιο και στη γραμματεία, είναι η αποδοχή ή αλλαγή της προκαθορισμένης ώρας συνέντευξης. Όταν η γραμματεία ορίσει ημερομηνία και ώρα συνέντευξης για κάποιον υποψήφιο, μπαίνοντας ο συγκεκριμένος στο λογαριασμό του θα δει την παρακάτω εικόνα.

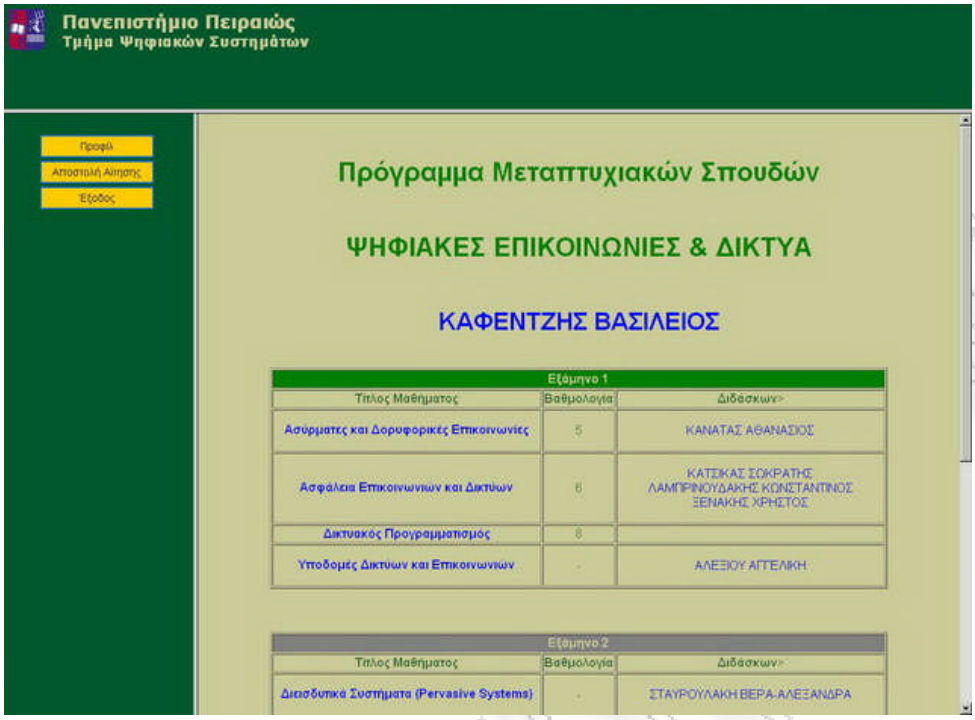


Εικόνα 88: Αποδοχή /Αλλαγή ώρας συνέντευξης

Εκεί ο υποψήφιος, έχει τη δυνατότητα να αποδεχτεί την προκαθορισμένη ημερομηνία και ώρα ή να ζητήσει αλλαγή ώρας. Έχουμε ήδη αναφέρει πως μετά τον ορισμό νέας ώρας από τη γραμματεία, ο υποψήφιος θα ενημερωθεί για την αλλαγή, από τον λογαριασμό του και τέλος θα ενημερωθεί για την τελική έκβαση της υποψηφιότητάς του, εάν δηλαδή απορρίφθηκε ή έγινε αποδεκτός (εικόνα 67). Στην περίπτωση που γίνει δεκτός, ενημερώνεται για την κατάθεση της προκαταβολής ,ώστε να εγγραφεί ως φοιτητής. Αφού καταθέσει την προκαταβολή των διδάκτρων ενημερώνεται για την εγγραφή του στο σύστημα ως φοιτητής (εικόνα 71).

4.4.4 Client Interface Φοιτητή (student)

Με την εγγραφή ενός υποψηφίου ως φοιτητή και τη δημιουργία του φοιτητικού λογαριασμού, ο φοιτητής αποκτά πρόσβαση σε αυτόν και άρα δυνατότητα συναλλαγών με τη γραμματεία. Αφού επισκεφθεί τη σελίδα εισόδου (εικόνα 54), και εισέλθει στο λογαριασμό του πιστοποιημένα θα εμφανιστεί στην οθόνη του η παρακάτω εικόνα.



Εικόνα 89: StudentLoginSuccessfull.jsp

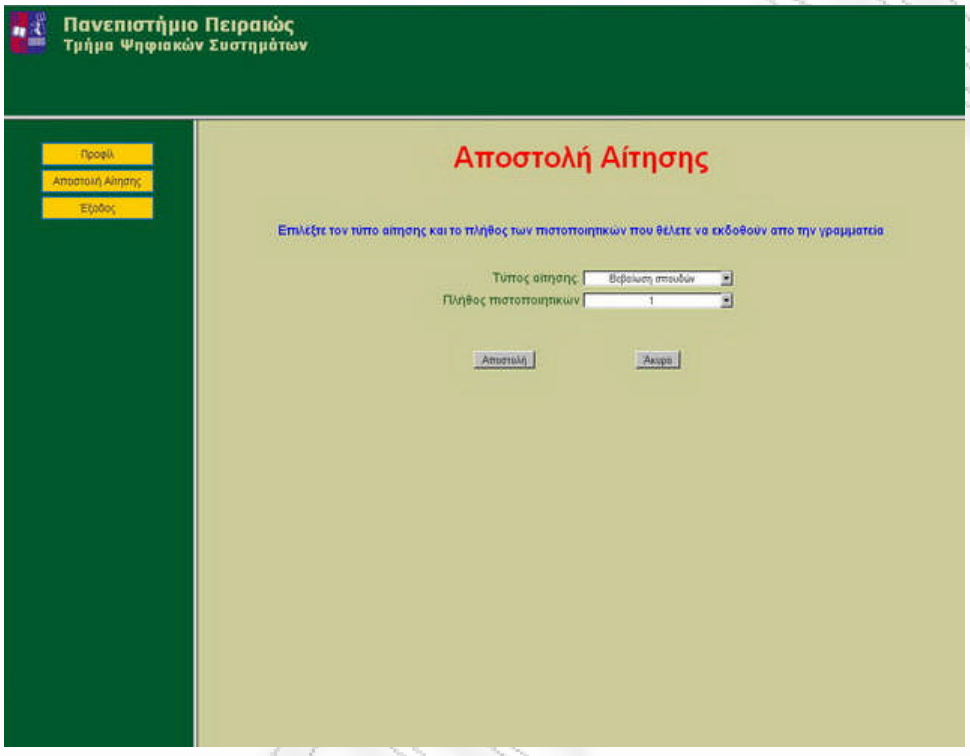
Έτσι κάθε φοιτητής, βλέπει τα μαθήματά του ανά εξάμηνο, την βαθμολογία του για όσα μαθήματα έχει καταχωρηθεί καθώς επίσης και τον εισηγητή του κάθε μαθήματος. Επίσης, το εξάμηνο το οποίο διανύει κάθε φορά και είναι το τρέχον, εμφανίζεται με πράσινο χρώμα. Τα υπόλοιπα εξάμηνα εμφανίζονται με γκρι χρώμα.

```
<body class="ex2">

    <%
    Hashtable<Integer, Direction> directionsById = (Hashtable<Integer, Direction>) session.getAttribute(ObjectsINSESSION.DIRECTIONS_BY_ID.toString());

    List<Semester> semesters = (List<Semester>) session.getAttribute(ObjectsINSESSION.SEMESTERS.toString());
    Hashtable<Integer , Semester> semestersById = (Hashtable<Integer , Semester>) session.getAttribute(ObjectsINSESSION.SEMESTERS_BY_ID.toString());

    Student student = (Student) session.getAttribute("student");

    Direction direction = directionsById.get(student.getDirectionId());
    //System.out.println("student.getAddress(): " + student.getAddress());

    Semester semester = semestersById.get(student.getSemesterId());
```

```
    //session.setAttribute("student", student);


    Hashtable<String, Vector<Course>> coursesByDidSid = (Hashtable<String, Vector<Course>>) session.getAttribute(ObjectsINSESSION.COURSES_BY_DID_SID.toString());
    Hashtable<Integer, Vector<Integer>> professorsIdByCid = (Hashtable<Integer, Vector<Integer>>) session.getAttribute(ObjectsINSESSION.PROFESSORS_ID_BY_CID.toString());
    Hashtable<Integer, Professor> professorsById = (Hashtable<Integer, Professor>) session.getAttribute(ObjectsINSESSION.PROFESSORS_BY_ID.toString());

    Hashtable<Integer, Score> scoresById = (Hashtable<Integer, Score>) session.getAttribute(ObjectsINSESSION.SCORES_BY_ID.toString());
    Hashtable<String, CourseScore> courseScoreByCidSid = (Hashtable<String, CourseScore>) session.getAttribute(ObjectsINSESSION.COURSE_SCORE_BY_CID_SID.toString());

    Vector<Course> courses;
    //courses = coursesByDidSid.get(student.getDirectionId() + "_" + student.getSemesterId());
    Vector<Integer> professorsIds;
%>

    <%--<h1 align="center">Προφίλ του <%=student.getLastName() %> <%=student.getFirstName() %>
        </h1>--%>

    <h5 align="center">Πρόγραμμα Μεταπτυχιακών Σπουδών </h5>
    <h5 align="center"><%=direction.getName() %></h5>


    <h4 align="center"><%=student.getLastName() %> <%=student.getFirstName() %></h4>
      <%
    //Semester semester;
    for(int i=0;i<semesters.size();i++)
    {
       semester = semesters.get(i);

       courses = coursesByDidSid.get(student.getDirectionId() + "_" + semester.getId());
       %>
       <table align="center" border="1">
          <tr bgcolor="<%= semester.getId() == student.getSemesterId() ? "green" : "gray" %>">
```

```
            <th align="center" colspan="3" width=300
style="color:#cccc99"><%=semester.getName() %>

            </th>
          </tr>
          <%
          if(courses!=null)
          {

            %>
          <tr>
            <td align="center" width=150><b>Τίτλος Μαθήματος</b></td>
            <td align="center" width=60><b>Βαθμολογία</b></td>
            <td align="center" width=150><b>Διδάσκων</b>></td>
          </tr>

          <%
            Course course;
            Score score;
            CourseScore courseScore;
            for(int j=0;j<courses.size();j++)
            {
              course = courses.get(j);

              professorsIds = professorsIdByCid.get(course.getId());

              int scoreId = 0;
              courseScore = courseScoreByCidSid.get(course.getId() + "_" + student.getUniqueId());
              if(courseScore!=null)
                scoreId = scoresById.get(courseScore.getScoreId()).getId();
%>
              <tr>
                <td align="center" width=350><b><a
href="ViewCourseProfile.jsp?courseId=<%=course.getId() %>"><%=course.getName()
%></b></a></td>
                <td align="center" width=60><%= scoresById.get( scoreId ).getName() %></td>
                <td align="center" width=350>
                <%
                if(professorsIds == null)
                {
%>
```

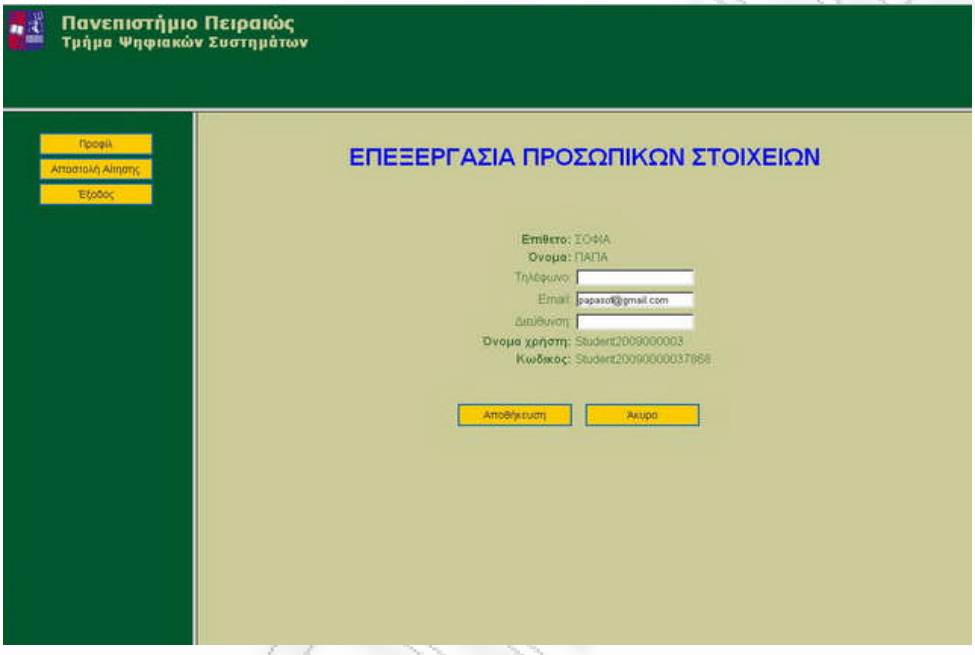
Όπως είδαμε παραπάνω, οι φοιτητές έχουν τη δυνατότητα αποστολής δύο τύπων αιτήσεων. Μπορούν να επιλέξουν έναν τύπο πιστοποιητικού κάθε φορά και πλήθος πιστοποιητικών (StudentSendRequest.jpg). Εφόσον ολοκληρωθεί η αποστολή με επιτυχία, ο φοιτητής ενημερώνεται για τον χρόνο παραλαβής των αιτήσεών του από την γραμματεία.



Εικόνα 90: StudendSendRequest.jsp

Εικόνα 91: StudentSendRequestSuccess.jsp

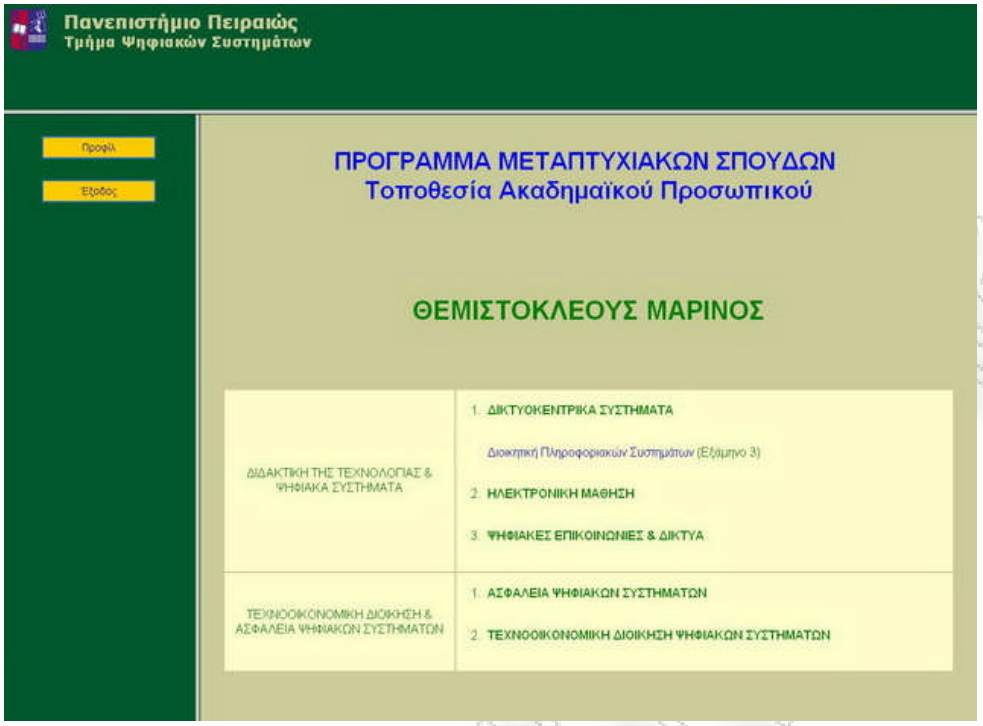
Τέλος οι φοιτητές, όπως και οι καθηγητές που θα δούμε στη συνέχεια, έχουν τη δυνατότητα να τροποποιήσουν κάποια προσωπικά τους στοιχεία. Από την επιλογή *Προφίλ,* μεταφέρονται στην StudentProfile.jsp και εκτός του ότι βλέπουν τα στοιχεία του λογαριασμού τους, μπορούν να αλλάξουν τα πεδία τηλέφωνο, email και διεύθυνση. Χρήσιμα στοιχεία για την γραμματεία, όχι όμως ζωτικής σημασίας για το σύστημα.



Εικόνα 92: StudentProfile.jsp

4.4.5 Client Interface Καθηγητή (Professor)

Ο χρήστης με τον οποίο ολοκληρώνονται οι clients και κατ' επέκταση και η όλη λειτουργία σε επίπεδο interface και συναλλαγών, είναι ο καθηγητής. Όλο το ακαδημαϊκό προσωπικό του τμήματος μπορεί να έχει τον δικό του λογαριασμό. Παρέχεται όπως είναι ήδη γνωστό, όνομα χρήστη και κωδικός πρόσβασης από την γραμματεία στους εκπαιδευτικούς και με αυτά τα στοιχεία αποκτούν πρόσβαση στον λογαριασμό τους.



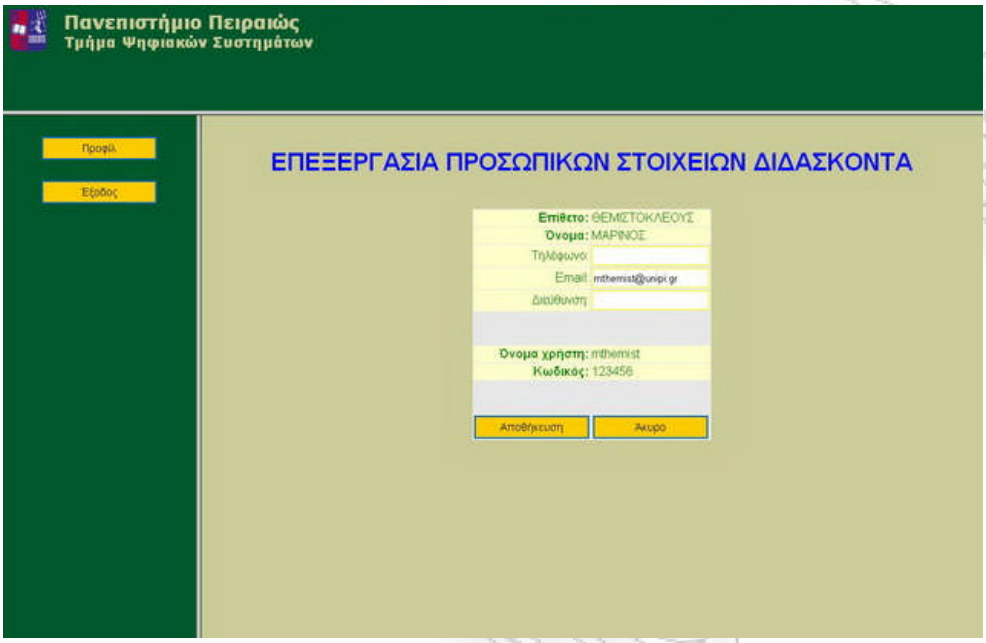
Εικόνα 93: ProfessorLoginSuccessful.jsp

```
<%
        Course course;
        for(int k=0;k<courses.size();k++)
        {
           course = courses.get(k);

           if(coursesForThisProf!=null && coursesForThisProf.contains(course.getId()))
           {
              %>
              <dt><a href="ProfessorViewCourseProfile.jsp?courseId=<%=course.getId() %>">
                 <%=course.getName() %></a> (<%=semester.getName() %>) </dt><br/>
              <%                                                                    }
   }
```

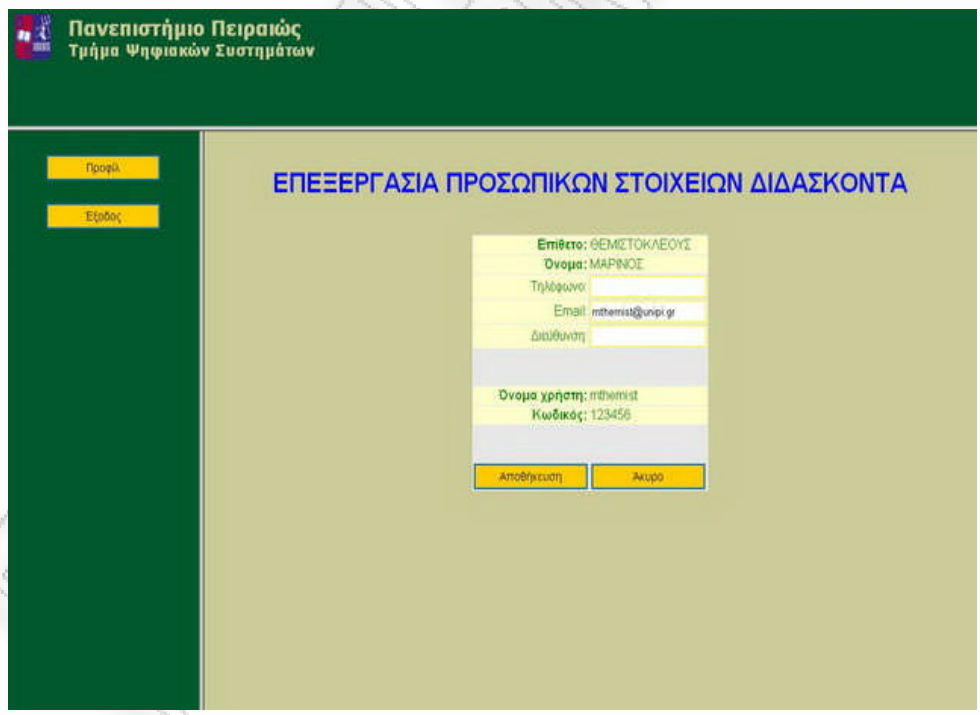
Εισέρχοντας εκεί, ο κάθε εκπαιδευτικός μπορεί να δει τα μεταπτυχιακά προγράμματα του τμήματος, τις κατευθύνσεις του κάθε προγράμματος και το πιο σημαντικό είναι το ότι βλέπει τα μαθήματα που ο ίδιος είναι εισηγητής. Η πληροφορία αυτή έχει καταχωρηθεί από την γραμματεία (εικόνα 84).

Επιλέγοντας κάποιο από τα μαθήματα που διδάσκει, βλέπει τις αντίστοιχες πληροφορίες του μαθήματος.

Τέλος, κάθε καθηγητής μπορεί να επεξεργαστεί και να τροποποιήσει κατά τα γνωστά, κάποια προσωπικά του στοιχεία όπως τηλέφωνο, email και διεύθυνση.



Εικόνα 94: ProfessorProfile.jsp και διεύθυνση.



Εικόνα 95: ProfessorProfile.jsp

ΚΕΦΑΛΑΙΟ 5: ΣΥΜΠΕΡΑΣΜΑΤΑ

Στο τελευταίο κεφάλαιο της παρούσας εργασίας, θα γίνει η αξιολόγηση του συστήματος που υλοποιήθηκε ώστε να δούμε εαν πραγματοποιήθηκαν οι στόχοι που αρχικά είχαν τεθεί. Θα παραθέσουμε επίσης συμπεράσματα τα οποία προήλθαν απο την γενικότερη μελέτη του θέματος και θα προτείνουμε κάποιες προτάσεις για βελτίωση στο μέλλον.

5.1 Αξιολόγηση Εφαρμογής - Συμπεράσματα

Σε μια προσπάθεια αξιολόγησης της εφαρμογής που αναλύθηκε και παρουσιάστηκε, θα λέγαμε ότι καταφέραμε να υλοποιήσουμε ένα σύστημα το οποίο αυτοματοποιεί σε ένα μεγάλο βαθμό την διαδικασία του Μεταπτυχιακού Προγράμματος του Τμήματος Ψηφιακών Συστημάτων, από την επιλογή των υποψηφίων μέχρι τις βαθμολογίες και παρουσίες των φοιτητών ανά μάθημα ανά εξάμηνο και ανά κατεύθυνση.

Ο κυρίαρχος ρόλος του διαχειριστή – γραμματείας είναι αυτός ο οποίος κινεί όλες τις διαδικασίες μέσα στο σύστημα ακολουθώντας τις εντολές και τις οδηγίες που έχει λάβει από την επιτροπή του Μεταπτυχιακού Προγράμματος και της Γενικής Συνέλευσης.

Χάρη στον ρόλο του διαχειριστή, καταφέραμε να υλοποιήσουμε αυτοματοποιημένη επικοινωνία και συναλλαγή με τους υποψήφιους και τους φοιτητές οι οποίοι από την δική τους μεριά εκτελούν με τη σειρά τους τις όποιες λειτουργίες έχουν διαθέσιμες. Φυσικά για λόγους ασφάλειας και ελεγχόμενης διαχείρισης οι ενέργειες που μπορούν να εκτελέσουν υποψήφιοι και φοιτητές είναι περιορισμένες και σύμφωνα με τη βούληση του διαχειριστή.

Τέλος από ένα τέτοιο σύστημα δεν θα μπορούσε να λείπει ο ρόλος των καθηγητών. Η αντιστοίχιση τους με κάθε μάθημα προσδίδει στο σύστημα μια ολοκληρωμένη εικόνα και μια σημαντική σειρά πληροφοριών τόσο για την γραμματεία όσο και για τους φοιτητές.

5.2 Μελλοντικές βελτιώσεις

Αν και το σύστημα αυτοματοποιεί σε μεγάλο βαθμό όπως προείπαμε την διαδικασία του μεταπτυχιακού προγράμματος, υπάρχουν σαφώς πολλά περιθώρια βελτίωσης.

Για παράδειγμα θα ήταν καλό, σε μια ενδεχόμενη προσπάθεια αναβάθμισης, ο αριθμός των αιτήσεων που καταφθάνουν στη βάση και στο interface της γραμματείας, να ανανεώνεται αυτόματα χωρίς να χρειάζεται refresh από τον χρήστη. Σε μια τέτοια περίπτωση θα χρειαζόταν κώδικας AJAX με χρήση Javascript που να «μιλάει» με τον server ανά τακτά χρονικά διαστήματα κλπ. Είναι αρκετά περίπλοκη και εξειδικευμένη διαδικασία και ξεφεύγει από τα όρια μιας μεταπτυχιακής διπλωματικής εργασίας.

Ένας από τους λόγους για τους οποίους το σύστημά μας δεν είναι τόσο ευέλικτο όσο θα θέλαμε για ενδεχόμενη πραγματική χρήση (αλλά παρόλα αυτά για τα επίπεδα μιας διπλωματικής εργασίας πολύ ευέλικτο), είναι τα διάφορα γραφειοκρατικά κωλύματα. Για παράδειγμα, θα μπορούσε η γραμματεία να στέλνει τις αιτήσεις και τα πιστοποιητικά που αιτούνται οι φοιτητές απευθείας στον φοιτητικό τους λογαριασμό. Κάτι τέτοιο όμως δεν είναι πρακτικό και ουσιαστικό διότι δεν θα ήταν εμφανής πουθενά η αυθεντικότητα του πιστοποιητικού ελλείψει της σφραγίδας και της υπογραφής του Γραμματέα του τμήματος.

Ένα ακόμα χαρακτηριστικό το οποίο θα έχριζε την εφαρμογή απολύτως χρησιμοποιήσιμη, θα ήταν η αυτόματη παραγωγή αριθμών πρωτοκόλλου ώστε να αποφεύγεται η όλη χειροκίνητη διαδικασία. Με αυτό το χαρακτηριστικό όμως η εφαρμογή αποκτά πλέον χαρακτήρα εφαρμογής μηχανογράφησης κάτι το οποίο αποτελεί ίσως υλικό για περεταίρω έρευνα.

ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΑΝΑΦΟΡΕΣ

CHAN, P. 2002. Java Examples from The Java Developers Almanac 1.4. Pearson Education, Inc.

CHAPPELL, D. A. & JEWELL, T. 2002. Java Web Services, O'Reilly & Associates.

DEITEL, H. M. & DEITEL, P. J. 2008. Java Προγραμματισμός Έκτη Έκδοση, Μ. Γκιούρδας.

DEITEL, H. M., DEITEL, P. J., NIETO, T. R., LIN, T. M. & SADHU, P. 2001. XML How to Program, Prentince Hall.

MICROSYSTEMS, S. 1995 - 2008. MySQL 5.0 Reference Manual. *http://dev.mysql.com/doc/refman/5.0/en/sql-syntax.html*. SUN Microsystems, Inc

MORGAN, C. 2004. Php5 and MySQL Bible, Wiley Publishinh, Inc.

OPTIONS, O. E. 2003. Η Τεχνολογία στην Επιχείρηση. Available: http://www.go-online.gr/ebusiness/specials/article.html?article_id=213.

PAPAZOGLOU, M. P. 2008. Web Services: Principles and Technology, Pearson Prentice Hall.

WALUYO, A. B., TANIAR, D., RAHAYU, W. & SRINIVASAN, B. 2008. Mobile serviceorientedarchitecturesforNN-queries. NetworkandComputerApplications.

ΔΗΜΗΤΡΑΚΟΠΟΥΛΟΣ, Γ. 2008. Κατανεμημένα Συστήματα, Σημειώσεις μαθήματος. Κατανεμημένα Συστήματα Universal Description and Discovery Integration (UDDI). Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Πειραώς.

ΘΕΜΙΣΤΟΚΛΕΟΥΣ, Μ. 2009a. Δικτυοκεντρικά ΠΣ, Σημειώσεις Μαθήματος. Ηλεκτρονικές Υπηρεσίες. Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Πειραιώς.

ΘΕΜΙΣΤΟΚΛΕΟΥΣ, Μ. 2009b. Δικτυοκεντρικά ΠΣ, Σημειώσεις μαθήματος. Ηλεκτρονικές Υπηρεσίες (e-Services). Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Πειραιώς.

ΜΑΚΡΗΣ, Ν. 2009. Σχεδιασμός και ανάπτυξη τεχνικών μέτρησης Πόρων και Υπηρεσιών σε Συστήματα Πλέγματος (Grid). Διπλωματική Εργασία. Αθηνα: Εθνικό Μετσόβιο Πολυτεχνείο.

ΜΑΡΚΑΤΟΣ, Δ. Ε. 2005. SOA and Web Services. Διπλωματική εργασία. Αθήνα: Εθνικό Μετσόβιο Πολυτεχνείο.

ΠΟΥΛΗΜΕΝΟΠΟΥΛΟΥ, Μ. 2009. Ηλεκτρονικές Υπηρεσίες Διαδικτύου, Σημειώσεις Μαθήματος. Web Services. Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Πειραιώς.

ΗΛΕΚΤΡΟΝΙΚΕΣ ΠΗΓΕΣ

HTTP://WIKI.EENG.DCU.IE:8888/EE557/G2/740-EE.HTML. Dublin City University, School of Electronic Engineering. *Web Services, Chapter 11* [Online].

HTTP://WWW.JSPTUT.COM/ *JSP Tutorial*

HTTP://WWW.ROSEINDIA.NET 2007. *Java Create XML File,Java Create XML Document,Create XML Java,Create XML Files*. *http://www.roseindia.net/xml/dom/CreatXMLFile.shtml*

MICROSYSTEMS, S. 1995 - 2008. *MySQL 5.0 Reference Manual*. *http://dev.mysql.com/doc/refman/5.0/en/sql-syntax.html* SUN

OPTIONS, O. E. 2003. Η Τεχνολογία στην Επιχείρηση. Available: http://www.go-online.gr/ebusiness/specials/article.html?article_id=213

W3SCOOLS.COM. *CSS Tutorial - http://www.w3schools.com/css/default.asp*

WHEATON, P. 1998 - 2009. *Java Ranch*. *http://www.javaranch.com/*

WWW.JAVA-TIPS.ORG 2005-2008a. *Java Tips - Home*. http://www.java-tips.org/index.html

WWW.JAVA-TIPS.ORG 2005-2008b. *Java Tips - Introducing the Sun Java Streaming XML Parser*. http://www.java-tips.org/java-ee-tips/enterprise-java-beans/introducing-the-sun-java-streaming-xml-p.html

WWW.SPIRITUS-TEMPORIS.COM Web service *http://www.spiritus-temporis.com/web-service/disadvantages-of-web-services.html*