TEI Κρήτης

Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Τμήμα Μηχανικών Πληροφορικής

Σύστημα Διαχείρισης Πτυχιακών Εργασιών Για Το Τμήμα Ηλεκτρολογίας

Ειρήνη Τσακού

ΑΜ: 2454

ΥΠΕΥΘΥΝΟΣ ΚΑΘΗΓΗΤΗΣ: ΝΙΚΟΛΑΟΣ ΠΑΠΑΔΑΚΗΣ

ΕΥΧΑΡΙΣΤΗΡΙΟ ΣΗΜΕΙΩΜΑ

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω όσους στήριξαν την προσπάθειά μου να ολοκληρώσω την πτυχιακή μου εργασία με θέμα την διαχείριση των πτυχιακών εργασιών του τμήματος ηλεκτρολογίας.

Πρώτον από όλους να ευχαριστήσω τον υπεύθυνο της πτυχιακής μου εργασίας κ. Νικόλαο Παπαδάκη που μου έδωσε την ευκαιρία να καταλήξω σε κάποιο θέμα πτυχιακής και να ολοκληρώσω με αυτό τον τρόπο τις σπουδές μου και την πορεία μου στο ίδρυμα. Ύστερα,θα ήθελα να ευχαριστήσω την οικογένεια μου για τη στήριξη που μου προσέφεραν καθ' όλη τη διάρκεια της εκπόνησης της εργασίας μου, καθώς επίσης και φίλους-συναδέλφους που με βοήθησαν να κατανοήσω βαθύτερα την εργασία. Και τέλος, ένα μεγάλο ευχαριστώ σε όλους τους καθηγητές του τμήματος Μηχανικών Πληροφορικής που μας εξοπλίζουν με τις γνώσεις τους και μας προετοιμάζουν για το μέλλον όλα αυτά τα χρόνια!

Περιεχόμενα

Πίνακας Εικόνων

Σύνοψη

Τα τελευταία χρόνια οι μηχανικοί λογισμικού στρέφονται όλο και περισσότερα σε διαδικτυακές εφαρμογές για την δυναμική διαχείριση δεδομένων. Η μεταστροφή αυτή οφείλεται στην διείσδυση των φορητών συσκευών και στην ανάγκη ανάπτυξης λογισμικού για την υποστήριξη όλων των διαφορετικών τύπων συσκευών. Στην παρούσα εργασία παρουσιάζουμε μια διαδικτυακή εφαρμογή διαχείρισης πτυχιακών εργασιών η οποία σχεδιάστηκε για το τμήμα Ηλεκτρολογίας. Σκοπός της πτυχιακής αυτής είναι η μελέτη των τεχνολογιών που αποσκοπούν για τον σχεδιασμό και υλοποίηση μιας διαδικτυακής εφαρμογής μηχανογράφησης πτυχιακών εργασιών.

Λέξεις – Κλειδιά: HTML5, RESTful API, AngularJS, NodeJS, MySQL

1 Εισαγωγή

Από την εμφάνιση του Web 2.0, web τεχνολογίες όπως AJAX, Java και JavaScript επιτρέπουν την ανάπτυξη εμπλουτισμένων web εφαρμογών. Ωστόσο, οι περισσότερες τεχνολογίες που απαιτούνται για την ανάπτυξη τέτοιων εφαρμογών χρειάζονται επιπρόσθετες βιβλιοθήκες. Τα τελευταία χρόνια γίνονται προσπάθειες για την επίλυση αυτού του προβλήματος με την έλευση του HTML5.

Η HTML5 υιοθετείται όλο και περισσότερο από τους μηχανικούς λογισμικού με τις περισσότερες ιστοσελίδες να «επανά–αναπτύσσονται» ώστε να την υποστηρίξουν και να μην εξαρτώνται σε άλλες βιβλιοθήκες ή σε τεχνολογίες άλλων όπως Java και Flash. Αυτό επιτρέπει την θέσπιση των web εφαρμογών σε συσκευές που υποστηρίζουν HTML5 όπως μεταξύ άλλων φορητές συσκευές, tablets, και έξυπνες τηλεοράσεις

Η παρούσα εργασία εστιάζεται και υιοθετεί πλήρως το Web 2.0 και σε όλες τις τεχνολογίες που το περικλείουν. Η επισκόπηση της εργασίας έχει ως εξής: αρχικά γίνεται αναφορά και ιστορική αναδρομή στο Web 2.0 και τις τεχνολογίες. Έπειτα περιγράφεται η σχεδίαση της εφαρμογής που αποσκοπεί στην μηχανογράφηση των πτυχιακών εργασιών λαμβάνοντας υπόψιν λειτουργίες και απαιτήσεις ενός τέτοιου συστήματος. Κεντρικός άξονας της σχεδίασης είναι η μελλοντική επεκτασιμότητα και η λειτουργικότητα. Στην συνέχεια περιγράφονται οι τεχνολογίες και η λογική της υλοποίησης. Τέλος περιγράφονται κάποια σενάρια χρήση της εφαρμογής παρουσιάζοντας ταυτόχρονα το αποτέλεσμα της εργασίας.

1.1 Web 2.0

Η εξέλιξη του Web 2.0 την τελευταία δεκαετία επέτρεψε την μετάλλαξη από τις υπάρχουσες εφαρμογές επιτραπέζιων υπολογιστών σε πλούσιες εφαρμογές που τρέχουν αποκλειστικά στον περιηγητή. Τέτοια παραδείγματα είναι τα Google Apps ή Microsoft Office Web Apps που από desktop εφαρμογές μπορεί κανείς να τα χρησιμοποιήσει από τον περιηγητή του. Αυτό είναι πιθανό με τις Web 2.0 τεχνολογίες και κυρίως με την τεχνολογία AJAX που επιτρέπει την ανάπτυξη ασύγχρονων web εφαρμογών. Ουσιαστικά με την τεχνολογία AJAX οι εφαρμογές μπορούν να ενημερώνονται με νέα δεδομένα χωρίς να γίνεται ανανέωση της ιστοσελίδας.

Με την ανάδυση του Web 2.0, οι εφαρμογές για δυναμική διαχείριση δεδομένων από βάσεις δεδομένων έχουν αυξηθεί σημαντικά. Αυτό οφείλεται στην ευκολία που έχουν οι web εφαρμογές όπως δεν χρειάζεται να γίνει κάποιου είδους εγκατάσταση λόγω ότι τρέχουν σε ένα παράθυρο ενός απλού περιηγητή, ενώ οι δυνατότητες και το γραφικό περιβάλλον είναι τουλάχιστον εφάμιλλες με αυτές μιας παραδοσιακής desktop εφαρμογής.

1.1.1 HTML - JavaScript

Η HTML, είναι μια τυπική γλώσσα σήμανσης που χρησιμοποιείται για τη δημιουργία ιστοσελίδων. Μαζί με το CSS, και JavaScript, η HTML χρησιμοποιείται από τις περισσότερες ιστοσελίδες για να δημιουργήσουν οπτικά ελκυστικές ιστοσελίδες, διεπαφών χρήστη για εφαρμογές web, και διεπαφές χρήστη για πολλές κινητές εφαρμογές. Η HTML περιγράφει τη δομή μιας ιστοσελίδας σημασιολογικά καθιστώντας τη μια γλώσσα σήμανσης, παρά μια γλώσσα προγραμματισμού. Η HTML γράφεται υπό μορφή στοιχείων HTML τα οποία αποτελούνται από ετικέτες (tags), οι οποίες περικλείονται μέσα σε σύμβολα «μεγαλύτερο από» και «μικρότερο από» (για παράδειγμα <html>), μέσα στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες HTML συνήθως λειτουργούν ανά ζεύγη, με την πρώτη να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης (ή σε άλλες περιπτώσεις ετικέτα ανοίγματος και ετικέτα κλεισίματος αντίστοιχα). Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ. Ο σκοπός ενός περιηγητή είναι να διαβάζει τα έγγραφα HTML και τα συνθέτει σε σελίδες που μπορεί κανείς να διαβάσει ή να ακούσει. Ο περιηγητής δεν εμφανίζει τις ετικέτες HTML, αλλά τις χρησιμοποιεί για να ερμηνεύσει το περιεχόμενο της σελίδας. Τα στοιχεία της HTML χρησιμοποιούνται για να κτίσουν όλους του ιστότοπους. Η HTML επιτρέπει την ενσωμάτωση εικόνων και άλλων αντικειμένων μέσα στη σελίδα, και μπορεί να χρησιμοποιηθεί για να εμφανίσει δια δραστικές φόρμες. Παρέχει τις μεθόδους δημιουργίας δομημένων εγγράφων (δηλαδή εγγράφων που αποτελούνται από το περιεχόμενο που μεταφέρουν και από τον κώδικα μορφοποίησης του περιεχομένου) καθορίζοντας δομικά στοιχεία σημαντικά για το κείμενο, όπως κεφαλίδες, παραγράφους, λίστες, συνδέσμους, παραθέσεις και άλλα. Μπορούν επίσης να ενσωματώνονται σενάρια εντολών σε γλώσσες όπως η JavaScript τα οποία επηρεάζουν τη συμπεριφορά των ιστοσελίδων HTML. Οι περιηγητές μπορούν επίσης να αναφέρονται σε στυλ μορφοποίησης CSS για να ορίζουν την εμφάνιση και τη διάταξη του κειμένου και του υπόλοιπου υλικού. Ο οργανισμός W3C, ο οποίος δημιουργεί και συντηρεί τα πρότυπα για την HTML και τα CSS, ενθαρρύνει τη χρήση των CSS αντί διαφόρων στοιχείων της HTML για σκοπούς παρουσίασης του περιεχομένου.

1.1.1.1 Ιστορική Αναδρομή HTML

Το 1980, ο φυσικός Τιμ Μπέρνερς Λι, ο οποίος εργαζόταν στο CERN επινόησε το ENQUIRE ένα σύστημα χρήσης και διαμοιρασμού εγγράφων για τους ερευνητές του CERN, και κατασκεύασε ένα πρωτότυπό του. Αργότερα, το 1989, πρότεινε ένα σύστημα βασισμένο στο διαδίκτυο, το οποίο θα χρησιμοποιούσε υπερκείμενο. Έτσι, έφτιαξε την προδιαγραφή της HTML και έγραψε τον περιηγητή και το λογισμικό εξυπηρετητή στα τέλη του 1990. Τον ίδιο χρόνο, ο Μπέρνερς Λι και ο μηχανικός συστημάτων πληροφορικής του CERN Robert Cailliau συνεργάστηκαν σε μια κοινή προσπάθεια εύρεσης χρηματοδότησης, αλλά το έργο δεν υιοθετήθηκε ποτέ επίσημα από το CERN. Στις προσωπικές του σημειώσεις από το 1990 ο Μπέρνερς Λι αριθμεί «μερικές από τις πολλές χρήσεις του υπερκειμένου», και αναφέρει πρώτα από όλες μια εγκυκλοπαίδεια.

Η πρώτη δημόσια διαθέσιμη περιγραφή της HTML ήταν ένα έγγραφο με το όνομα Ετικέτες HTML, το οποίο πρωτοαναφέρθηκε στο διαδίκτυο από τον Μπέρνερς Λι στα τέλη του 1991. Περιέγραφε τα 20 στοιχεία τα οποίο αποτελούσαν τον αρχικό και σχετικά απλό σχεδιασμό της HTML. Εκτός από την ετικέτα υπερ-συνδέσμου, οι υπόλοιπες ήταν έντονα επηρεασμένες από την SGML guid, μια μορφή δημιουργίας τεκμηρίωσης, φτιαγμένη στο CERN και βασισμένη στην SGML. Δεκατρία από εκείνα τα αρχικά στοιχεία υπάρχουν ακόμα σήμερα στην HTML 4.

1.1.1.2 Ιδιότητες και στοιχεία

Τα έγγραφα HTML αποτελούνται από στοιχεία HTML τα οποία στην πιο γενική μορφή τους έχουν τρία συστατικά: ένα ζεύγος από ετικέτες, την «ετικέτα εκκίνησης» και την «ετικέτα τερματισμού», μερικές ιδιότητες μέσα στην ετικέτα εκκίνησης, και τέλος το κείμενο ή το γραφικό περιεχόμενο μεταξύ των ετικετών, το οποίο μπορεί να περιλαμβάνει και άλλα στοιχεία εμφωλευμένα μέσα του. Το στοιχείο HTML μπορεί να είναι οτιδήποτε ανάμεσα στις ετικέτες εκκίνησης και τερματισμού. Τέλος, κάθε ετικέτα περικλείεται σε σύμβολα «μεγαλύτερο από» και «μικρότερο από», δηλαδή < και >. Επομένως, η γενική μορφή ενός στοιχείου HTML είναι: <ετικέτα ιδιότητα1="τιμή1" ιδιότητα2="τιμή2">περιεχόμενο</ετικέτα>. Μερικά στοιχεία HTML περιγράφονται ως άδεια στοιχεία, έχουν τη μορφή <ετικέτα ιδιότητα1="τιμή1" ιδιότητα2="τιμή2">, και δεν έχουν καθόλου περιεχόμενο. Το όνομα κάθε στοιχείου HTML είναι το ίδιο όνομα που χρησιμοποιείται στις αντίστοιχες ετικέτες. Το όνομα της ετικέτας τερματισμού ξεκινά με μια κάθετο «/», η οποία παραλείπεται στα άδεια στοιχεία. Τέλος, αν δεν αναφέρονται ρητά οι ιδιότητες ενός στοιχείου, τότε χρησιμοποιούνται οι προεπιλογές σε κάθε περίπτωση.

Οι περισσότερες ιδιότητες των στοιχείων είναι ζεύγη ονομάτων και τιμών, τα οποία διαχωρίζονται με ένα «=» και γράφονται μέσα στην ετικέτα εκκίνησης ενός στοιχείου, μετά το όνομα του στοιχείου. Η τιμή μπορεί να περικλείεται σε μονά ή διπλά εισαγωγικά, παρότι τιμές που αποτελούνται από συγκεκριμένους χαρακτήρες μπορούν να γράφονται χωρίς εισαγωγικά στην HTML, αλλά όχι στην XHTML. Το να μένουν οι τιμές των ιδιοτήτων χωρίς εισαγωγικά θεωρείται ανασφαλές. Εκτός από τις ιδιότητες που γράφονται ως ζεύγη ονομάτων και τιμών, υπάρχουν και μερικές οι οποίες επηρεάζουν το στοιχείο απλά με την παρουσία τους μέσα στην ετικέτα εκκίνησης. Από την έκδοση 4.0 και μετά, στην HTML ορίζεται ένα σύνολο από 252 αναφορές οντοτήτων χαρακτήρων και ένα σύνολο από 1.114.050 αναφορές οντοτήτων αριθμών. Και τα δύο σύνολα επιτρέπουν τη γραφή μοναδικών χαρακτήρων ως σήμανση, αντί χρησιμοποιώντας τους ίδιους τους χαρακτήρες. Ένας χαρακτήρας και η αντίστοιχη σήμανση γι' αυτόν θεωρούνται ισοδύναμες οντότητες, και εμφανίζονται ίδια.

1.1.1.3 JavaScript

Η JavaScript δημιουργήθηκε το 1995 από τον Brendan Eich, έναν μηχανικό της Netscape και εκδόθηκε με τον Netscape 2 στις αρχές του 1996. Το αρχικό της όνομα ήταν LiveScript αλλά πιθανώς για λόγους μάρκετινγκ, καθώς η εξάπλωση της Java ήταν μεγάλη, μετονομάστηκε σε JavaScript. Είναι μία γλώσσα προγραμματισμού επηρεασμένη από αντικειμενοστραφείς γλώσσες προγραμματισμού όπως η C++ και Java, αλλά η ίδια δεν είναι αντικειμενοστραφής γλώσσα προγραμματισμού.

Ο κώδικας της JavaScript γράφεται σε καθαρό κείμενο (ASCII μορφή) και ενσωματώνεται μέσα στον κώδικα της HTML, μπορεί δε να εκτελεστεί αμέσως ή όταν λαμβάνει χώρα ένα συμβάν (event). Δεν γίνεται μεταγλώττιση (compilation) του κώδικα της JavaScript, αρκεί μόνο ο περιηγητής να υποστηρίζει την JavaScript. Μερικά από τα κυρίως χαρακτηριστικά της JavaScript είναι:

- εύκολη στην χρήση
- εκτελείται γρήγορα
- τα προγράμματα ενσωματώνονται στις σελίδες της html και δεν χρειάζονται μεταγλώττιση ή διερμήνευση
- είναι κατάλληλο για απλές χρήσεις αλλά είναι αρκετό ικανό ακόμα και σε περίπλοκα σενάρια.
- αν και δεν είναι αντικειμενοστραφής γλώσσα είναι βασισμένη στην φιλοσοφία της αντικειμενοστρέφειας

1.1.2 Web Services

Μια υπηρεσία Web προσφέρεται από μια ηλεκτρονική συσκευή σε μια άλλη. Επικοινωνούν μεταξύ τους μέσω του διαδικτύου. Μια υπηρεσία web, όπως το HTTP έχει σχεδιαστεί για τη μεταφορά αναγνώσιμων αρχείων όπως XML και JSON. Στην πράξη, μια υπηρεσία Web παρέχει τυπικά ένα αντικείμενο με προσανατολισμό web-based interface σε ένα διακομιστή της βάσης δεδομένων, που χρησιμοποιούνται για παράδειγμα από ένα άλλο διακομιστή ιστού, ή από μια κινητή εφαρμογή, που παρέχει μια διεπαφή χρήστη για τον τελικό χρήστη.

Ο όρος "Web service" περιγράφει έναν τυποποιημένο τρόπο ενσωμάτωσης των Web-based. Μια υπηρεσία Web είναι μια μέθοδος επικοινωνίας μεταξύ δύο ηλεκτρονικών συσκευών σε ένα δίκτυο. Είναι μια λειτουργία του λογισμικού που παρέχεται σε μια διεύθυνση δικτύου μέσω του Web με την υπηρεσία πάντα ως προς την έννοια της υπολογιστικής χρησιμότητας. Το W3C ορίζει μια υπηρεσία Web:

"*Ένα σύστημα λογισμικού σχεδιασμένο για να υποστηρίζει δια λειτουργική αλληλεπίδραση υπολογιστή-σε-υπολογιστή μέσω ενός δικτύου.*"

Πολλοί οργανισμοί χρησιμοποιούν πολλαπλά συστήματα λογισμικού για τη διαχείριση δεδομένων. Διαφορετικά συστήματα λογισμικού συχνά χρειάζεται να ανταλλάσσουν δεδομένα μεταξύ τους, καθώς και μια Web service είναι μια μέθοδος επικοινωνίας που επιτρέπει σε δύο συστήματα λογισμικού την ανταλλαγή των δεδομένων μέσω του διαδικτύου. Το σύστημα λογισμικού που ζητά τα δεδομένα ονομάζεται *service requester*, ενώ το σύστημα λογισμικού που θα επεξεργαστεί το request και να παρέχουν τα δεδομένα ονομάζεται service provider. Διαφορετικά το λογισμικό μπορεί να χρησιμοποιήσει διαφορετικές γλώσσες προγραμματισμού, και ως εκ τούτου υπάρχει ανάγκη για μία μέθοδο ανταλλαγής δεδομένων που δεν εξαρτάται από μια συγκεκριμένη γλώσσα προγραμματισμού. Οι περισσότεροι τύποι λογισμικού μπορούν, ωστόσο, να ερμηνεύσουν τις ετικέτες XML. Έτσι, οι υπηρεσίες Web μπορούν να χρησιμοποιήσουν τα αρχεία XML για την ανταλλαγή δεδομένων.

Ένας κατάλογος που ονομάζεται UDDI (Universal Description, Discovery και Ενσωμάτωσης) καθορίζει ποιο σύστημα λογισμικού θα πρέπει να έρθει σε επαφή για το είδος των δεδομένων. Έτσι, όταν ένα λογισμικό σύστημα χρειάζεται μία συγκεκριμένη report/data, θα πάει στο UDDI και θα καθορίσει ποιο άλλο σύστημα μπορεί να επικοινωνήσει για τη λήψη αυτών των στοιχείων. Μόλις το σύστημα λογισμικού ανακαλύψει με ποιο άλλο σύστημα θα πρέπει να επικοινωνήσει, θα επικοινωνήσει χρησιμοποιώντας ένα ειδικό πρωτόκολλο που ονομάζεται SOAP (Simple Object Access Protocol). Το σύστημα παροχής υπηρεσιών θα επικυρώσει πρώτα το αίτημα, και στη συνέχεια επεξεργάζεται την αίτηση και στέλνει τα δεδομένα σύμφωνα με το πρωτόκολλο SOAP.

1.1.3 Αρχιτεκτονική Representational State Transfer (Rest)

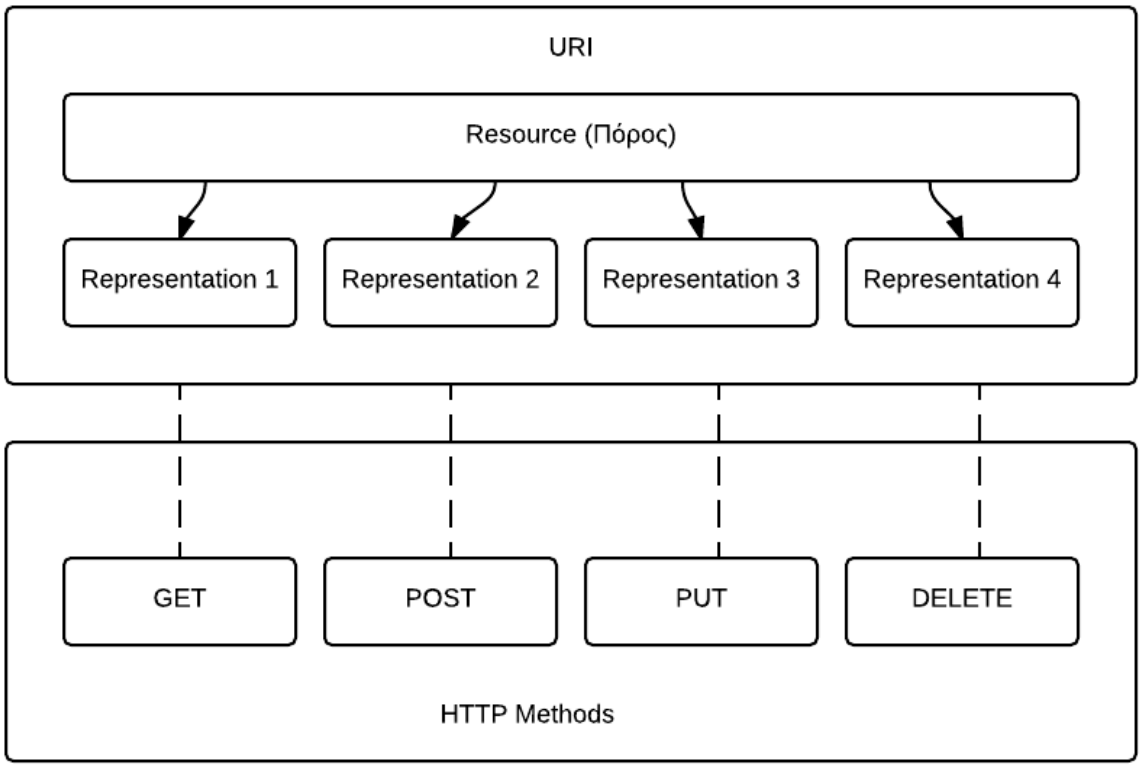
Στο προηγούμενο υπό κεφάλαιο είδαμε πως ορίζονται γενικά οι web υπηρεσίες. Σε αυτήν την ενότητα θα δούμε την εναλλακτική μεθοδολογία υλοποίησης υπηρεσιών web βασισμένη στην αρχιτεκτονική REST.

Η αρχιτεκτονική REST ορίστηκε στο διδακτορικό του Fielding “Architectural Styles and the Design of Network-based Software Architectures” και αναφέρεται σε μια αρχιτεκτονική υλοποίησης υπηρεσιών ανάμεσα σε κατανεμημένα συστήματα υπερμέσων. Το κύριο αντιπροσωπευτικό τέτοιο σύστημα είναι ο παγκόσμιος ιστός και είναι ο λόγος που αναπτύχθηκε η αρχιτεκτονική REST, ώστε να περιγράψει ένα σύνολο από κανόνες και λειτουργίες που πρέπει να διέπουν ένα σύστημα υπηρεσιών. Η αρχιτεκτονική REST βασίζεται σε μερικές βασικές αρχές:

- *Ομοιομορφία διεπαφής*: Οι αλληλεπιδράσεις μεταξύ των μηχανημάτων θα πρέπει να ακολουθούν τα καθιερωμένα HTTP πρότυπα.
- *Σύστημα πολλαπλών επιπέδων*: Η χρήση της ομοιόμορφης διεπαφής επιτρέπει την εισαγωγή ενδιάμεσων επιπέδων ανάμεσα στον εξυπηρετητή και τον πελάτη με σκοπό την

καλύτερη εξυπηρέτηση του δικτύου. Ένα ενδιάμεσο επίπεδο πρέπει να μην παίζει ρόλο στην τελική διεπαφή και να είναι αόρατο κατά τη διάρκεια της επικοινωνίας.

- *Αρχιτεκτονική client*-server: Το βασικό στοιχείο πάνω στο οποίο βασίζονται όλα τα υπόλοιπα είναι ότι το σύστημα ακολουθεί την μορφή πελάτη-εξυπηρετητή, με τρόπο έτσι ώστε η τεχνολογική κατάσταση του κάθε ενός να είναι εντελώς ανεξάρτητη από τον άλλον, διατηρώντας διακριτούς ρόλους στην επικοινωνία τους.
- *Cache των δεδομένων*: Με τη δυνατότητα αποθήκευσης σε μνήμη cache σε οποιοδήποτε από τα επίπεδα επικοινωνίας ελαττώνεται σημαντικά ο όγκος των δεδομένων προς μεταφορά, προσδίδοντας έτσι μεγαλύτερη ταχύτητα και αξιοπιστία στην εφαρμογή.
- *Ανεξαρτησία κατάστασης*: Βασική προϋπόθεση είναι ότι ο εξυπηρετητής δεν είναι υποχρεωμένος να γνωρίζει συνεχώς τις προηγούμενες καταστάσεις που βρισκόταν ο πελάτης, απαιτώντας έτσι σε κάθε επικοινωνία τους ο πελάτης να περιέχει όλες τις απαραίτητες παραμέτρους που χρειάζονται για να επιτευχθεί η σωστή μεταξύ τους επικοινωνία



Εικόνα 1: Αρχιτεκτονική REST

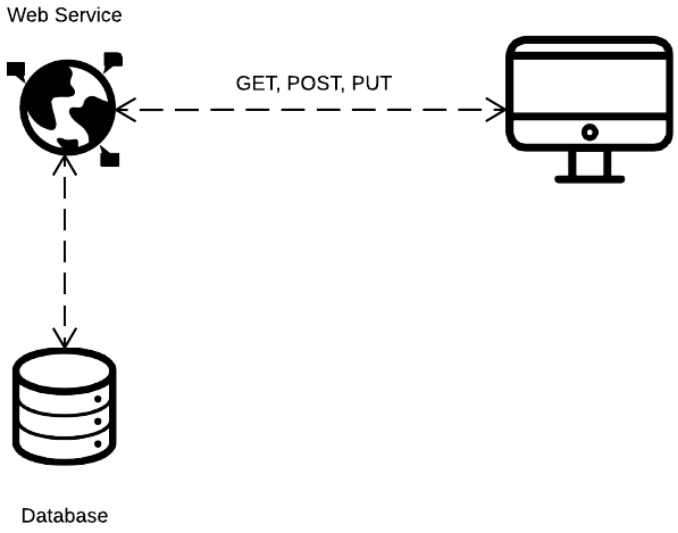
Συνοπτικά τα κυρίως πλεονεκτήματα των REST υπηρεσιών είναι:

- προσφέρουν μια απλοποιημένη και ελαφρύτερη υλοποίηση
- είναι βασισμένα σε ανοιχτού κώδικα πρότυπα όπως HTTP, URI και XML.
- ο περιορισμός της διεπαφής ορίζει ότι τα μηνύματα πρέπει να είναι αυτό-περιγραφικά ώστε να αποτρέπονται λάθη σε ενδιάμεσες καταστάσεις
- η αναπαράσταση των δεδομένων είναι αποσυνδεδεμένα από τους πόρους οπότε μια REST υπηρεσία μπορεί να αναπαραστήσει δεδομένα με διάφορους τρόπους όπως XML, JSON κ.α.

Όπως θα δούμε και στο επόμενο κεφάλαιο ένα κομμάτι της υλοποίησης μας βασίζεται στην δημιουργία ενός RESTful API για την επικοινωνία της διαδικτυακής μας εφαρμογής με την βάση δεδομένων.

2 Σχεδιασμός εφαρμογής

Στο κεφάλαιο αυτό θα παρουσιαστουν τα στάδια σχεδιασμού της εφαρμογής. Αρχικά θα παρουσιαστεί η φιλοσοφία και η σχεδίαση της βάσης δεδομένων, στην συνέχεια της διαδικτυακής υπηρεσίας. Τέλος θα κλείσουμε με την σχεδίαση της διεπαφής. Η σχεδίαση της εφαρμογής χωρίζεται σε τρία κυρίως κομμάτια. Την **βάση δεδομένων** που αποτελεί το μοντέλο της εφαρμογής μας και αποθηκεύονται όλα τα απαραίτητα δεδομένα. Ο δεύτερος κορμός είναι η **διαδικτυακή υπηρεσία** η οποία είναι υπεύθυνη για την ενδιάμεση επικοινωνία της διεπαφής και της βάσης δεδομένων. Ενώ θα μπορούσαμε να απλοποιήσουμε την υλοποίηση και να επικοινωνούσε η διεπαφή απευθείας με την βάση δεδομένων επιλέχθηκε η ύπαρξη της ενδιάμεσης διαδικτυακής υπηρεσίες ώστε να υποστηρίζεται η επεκτασιμότητα της εφαρμογής. Ένα παράδειγμα θα ήταν να αναπτυσσόταν μια εφαρμογή για φορητές συσκευές, στην περίπτωση που δεν υπήρχε η υπηρεσία θα έπρεπε να στηθεί πάλι από την αρχή ένας μηχανισμός επικοινωνίας με την βάση δεδομένων. Το τελευταίο κομμάτι της εφαρμογής είναι η **διεπαφή** της εφαρμογής η οποία ουσιαστικά είναι μια ιστοσελίδα που τρέχει σε έναν περιηγητή. Το σημαντικότερο κομμάτι της υλοποίησης αποτελεί η διεπαφή καθώς είναι υπεύθυνή για την λογική της εφαρμογής η οποία θα παρουσιαστεί στο κεφάλαιο 4. Στο παρακάτω σχήμα διακρίνεται ο αφηρημένος σχεδιασμός της εφαρμογής διαχείρισης πτυχιακών.



Εικόνα 2: Σχεδιασμός Εφαρμογής

2.1 Βάση Δεδομένων

Για την σχεδίαση της βάσης δεδομένων βασιστήκαμε σε 5 κύριες οντότητες όπως φαίνεται στο παρακάτω διάγραμμα οντοτήτων συσχετίσεων. Τις *Πτυχιακή*, *Φοιτητής*, *Καθηγητής*, *Χρήστης*, *Τομέας*. Στην σχεδίαση έγινε προσπάθεια αντιστοίχισης κάθε οντότητας με τον πραγματικό κόσμο. Για κάθε πτυχιακή είναι απαραίτητο να αποθηκεύουμε κάποια δεδομένα, όπως τι τίτλο και περιγραφή θα έχει η πτυχιακή ή ακόμα και τον βαθμό δυσκολίας και την ημερομηνία ανάληψης ώστε να μπορούμε να εφαρμόζουμε περιορισμούς στην διάρκεια ανάληψης. Για την οντότητα Πτυχιακή καταλήξαμε στα εξής γνωρίσματα:

- *Τίτλος*
- *Περιγραφή*
- *Βαθμός δυσκολίας*
- *Προϋποθέσεις*
- *Αριθμό Σπουδαστών*
- *Ημερομηνία Καταχώρησης*
- *Ημερομηνία Ανάληψης*

Όσο αφορά τους χρήστες, οι οποίοι χωρίζονται σε 2 κατηγορίες Φοιτητής - Καθηγητής, είναι απαραίτητο να έχουν username και password ώστε να έχουν λογαριασμούς στην εφαρμογή. Επίσης είναι απαραίτητο να αποθηκεύουμε τα ονοματεπώνυμα τους. Για τον φοιτητή είναι απαραίτητο να γνωρίζουμε τον αριθμό μητρώου του καθώς και το έτος εισαγωγής για τυχόν περιορισμούς όπως το ότι ένας φοιτητής μπορεί να αναλάβει μια πτυχιακή εφόσον είναι σε κάποιο συγκεκριμένο εξάμηνο φοίτησης. Για την οντότητα Φοιτητής τα γνωρίσματα είναι:

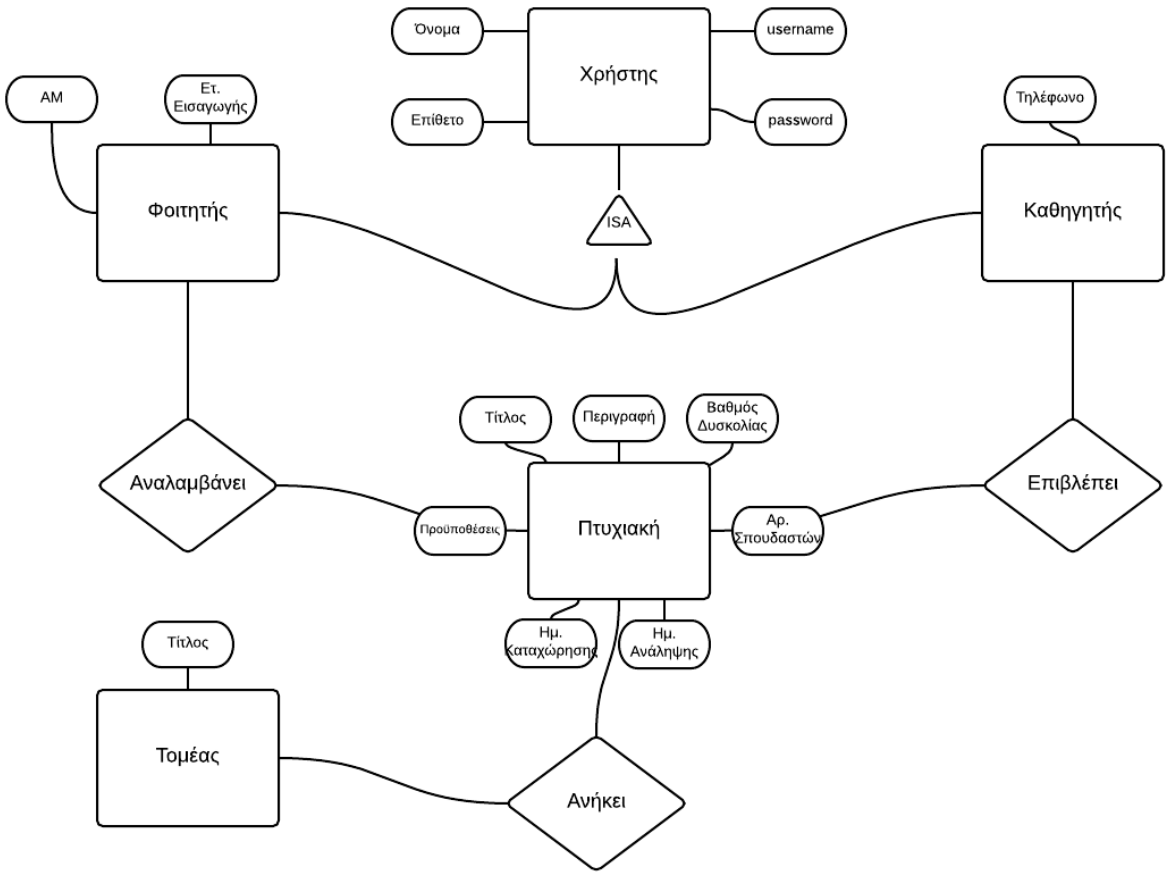
- *ΑΜ*
- *Έτος εισαγωγής*
- *Όνομα*
- *Επίθετο*

Η οντότητα Καθηγητής, στον πραγματικό κόσμο ο επιβλέπων της πτυχιακής εργασίας, έχει τα γνωρίσματα:

- *Όνομα*
- *Επίθετο*
- *Τηλέφωνο*

Κάθε πτυχιακή εργασία ανήκει σε κάποιον τομέα η οποία έχει το γνώρισμα *τίτλος*. Για τις συσχετίσεις μεταξύ των οντοτήτων βασιστήκαμε σε διάφορες εκφράσεις οι οποίες αργότερα μετατράπηκαν στο παρακάτω μοντέλο ή σε σχεσιακή γλώσσα όπως περιγράφεται στο επόμενο κεφάλαιο. Οι εκφράσεις αυτές είναι:

- *Ένας ή περισσότεροι φοιτητές αναλαμβάνουν μια πτυχιακή*
- *Ένας ή περισσότεροι καθηγητές επιβλέπουν μια ή περισσότερες πτυχιακές*
- *Μια πτυχιακή εργασία ανήκει σε έναν τομέα*
- *Ο φοιτητής είναι χρήστης*
- *Ο καθηγητής είναι χρήστης*



Εικόνα 3: Διάγραμμα Οντοτήτων-Συσχετίσεων

2.2 Web Service – RESTful API

Για τον σχεδιασμό της υπηρεσίας web επιλέχθηκε η αρχιτεκτονική REST καθώς ήταν η πιο κατάλληλη για εξόρυξη ή ανάγνωση δεδομένων από την βάση δεδομένων καθώς και για τα πλεονεκτήματα που αναφέρθηκαν στο προηγούμενο κεφάλαιο. Ο κάθε πόρος αντιπροσωπεύεται από μια μοναδική διεύθυνση (URI). Στην υπηρεσία γίνεται αίτημα για συγκεκριμένο πόρο και υπάρχουν μέθοδοι για την επιστροφή του σε διάφορες αναπαραστάσεις. Για παράδειγμα, αν χρειαζόμασταν όλους τους φοιτητές της βάσης και η απάντηση της υπηρεσίας να ήταν σε JSON το URI θα ήταν της μορφής:

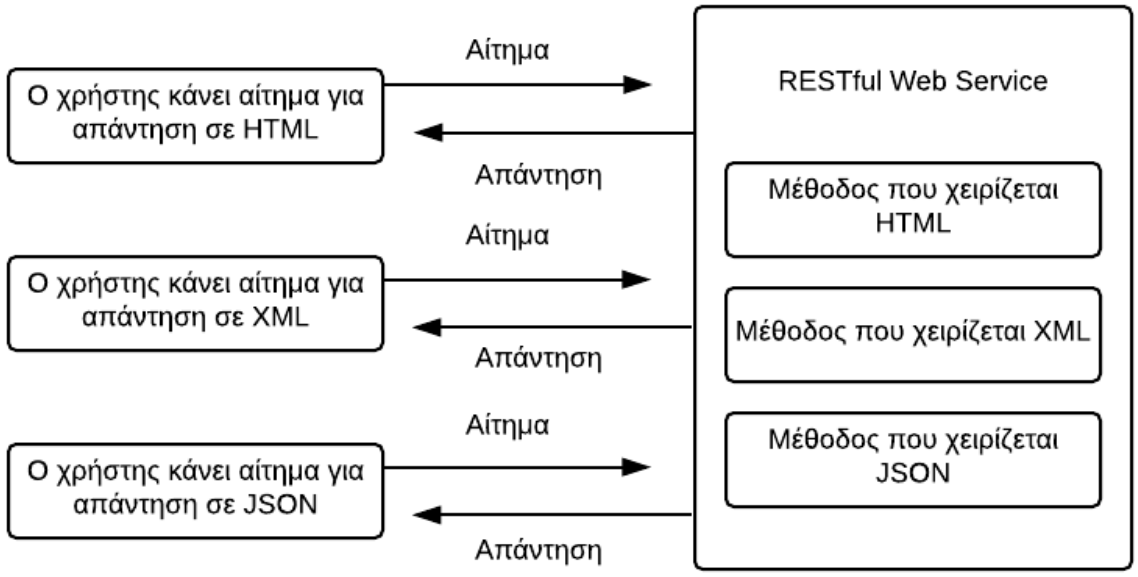
…./webresource/**foitites**?content=**JSON**

Ενώ αν χρειαζόμασταν τον φοιτητή με το μοναδικό κωδικό 1 και η απάντηση σε XML το URI θα

ήταν της μορφής:

…./webresource/**foitites/1**?content=**XML**

Η γενικότερη φιλοσοφία ενός RESTful API είναι να κάνεις αίτημα για συγκεκριμένο πόρο και να απαιτείς η απάντηση να είναι μια συγκεκριμένη αναπαράσταση όπως φαίνεται στο παρακάτω σχήμα.



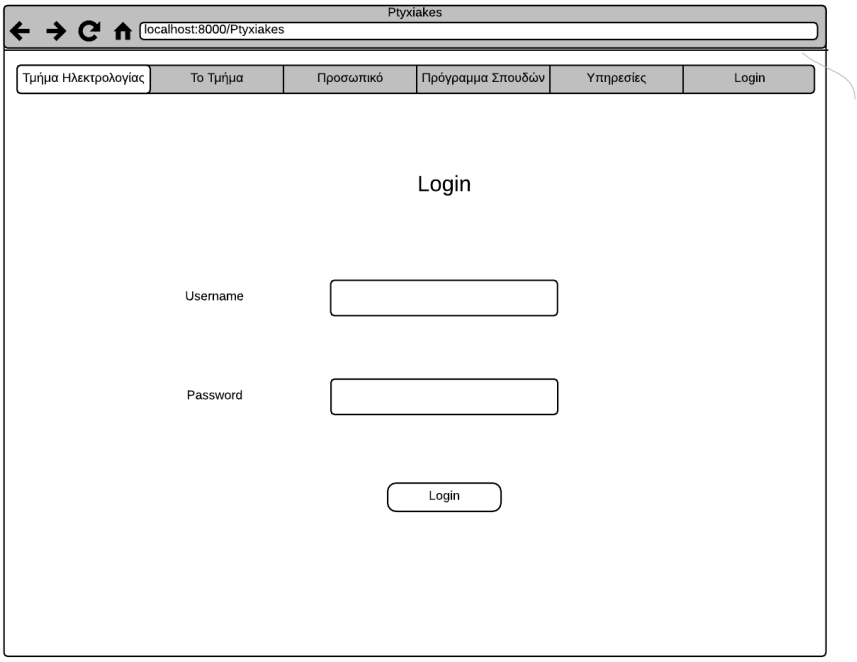
Εικόνα 4: RESTful API

2.3 Διεπαφή εφαρμογής

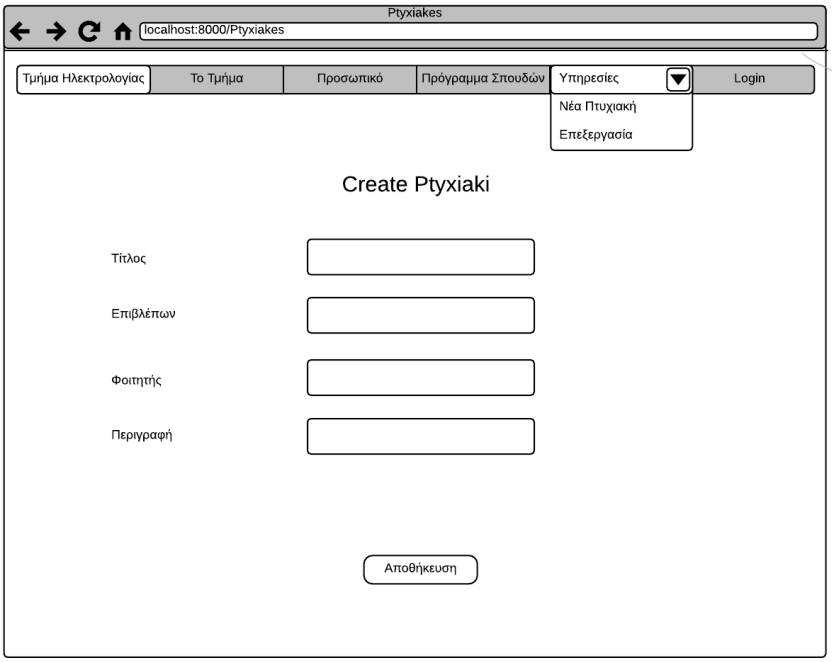
Η σχεδίαση της διεπαφής έγινε με γνώμονα τις απαιτήσεις ενός τέτοιου συστήματος. Μια κύρια απαίτηση θα ήταν ένας μηχανισμός login/logout και ο διαχωρισμός της διεπαφής ανάλογα τον τύπο του χρήστη. Ένας φοιτητής δεν θα έπρεπε να μπορεί να δημιουργήσει μια πτυχιακή, θα έπρεπε μόνο να του δίνεται η δυνατότητα να δει ποιες ελεύθερες πτυχιακές υπάρχουν και να επιλέξει μια. Ενώ αν έκανε σύνδεση στην εφαρμογή ένας καθηγητής θα έπρεπε να του δίνονταν οι δυνατότητες δημιουργίας μιας πτυχιακής και διαχείριση υπαρχουσών. Άλλη απαίτηση θα ήταν για την δημιουργία ή επεξεργασία μιας πτυχιακής να δίνονταν όλα τα απαραίτητα εργαλεία στον χρήστη. Η γενική ιδέα για τη διεπαφή της εφαρμογής μας συνίσταται στη χρήση

συστατικών που τοποθετούνται στις σελίδες της και της προσδίδουν την απαιτούμενη λειτουργικότητα. Συστατικά όπως φόρμες, κουμπιά, μενού επιλογής και πίνακες, γνώριμα στους χρήστες του διαδικτύου, τα οποία δομούνται ακολουθώντας ένα νοητό περίγραμμα ώστε να συνθέσουν μια διεπαφή φιλική στο χρήστη. Οι φόρμες χρησιμοποιούνται όπου προβλέπεται ή αναμένεται είσοδος από το χρήστη, τα κουμπιά για την αλληλεπίδραση του με την εφαρμογή μιας και επιτρέπουν την πυροδότηση γεγονότων, οι πίνακες για παρουσίαση δεδομένων, ενώ τα μενού επιλογής σε σημεία όπου η είσοδος που αναμένεται από το χρήστη είναι τυποποιημένη, οπότε του δίνεται η δυνατότητα να επιλέξει παρά να πληκτρολογήσει. Επιπλέον, σε κάθε περίπτωση εκτέλεσης κάποιας εντολής ή απαγόρευσης κάποιας εντολής, ο χρήστης ενημερώνεται μέσω μηνύματος που παρουσιάζεται σε παράθυρο απόκρισης.

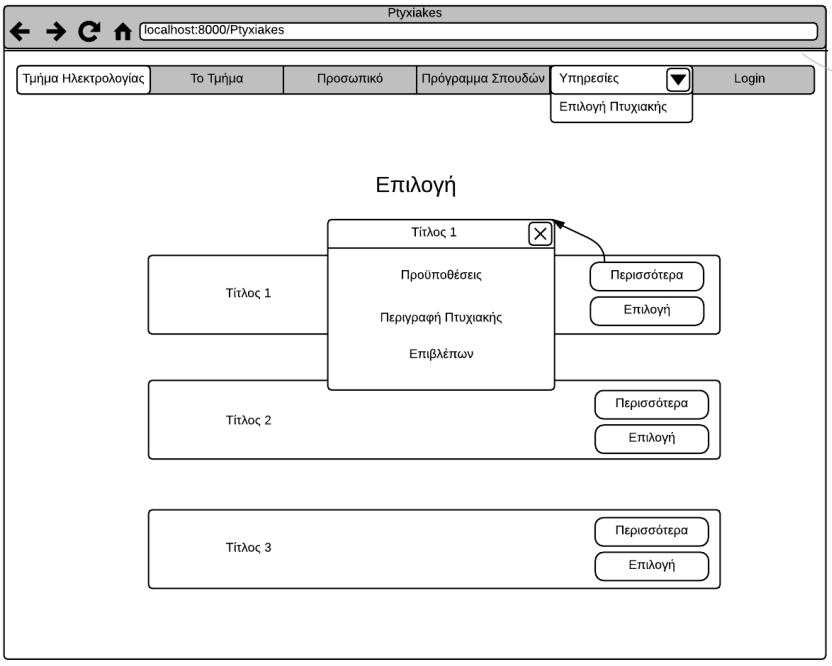
Χρησιμοποιώντας εργαλεία δημιουργίας αφηρημένων διεπαφών καταλήξαμε σε μερικές διεπαφές για την εφαρμογή.



Εικόνα 5: Σύνδεση χρήστη



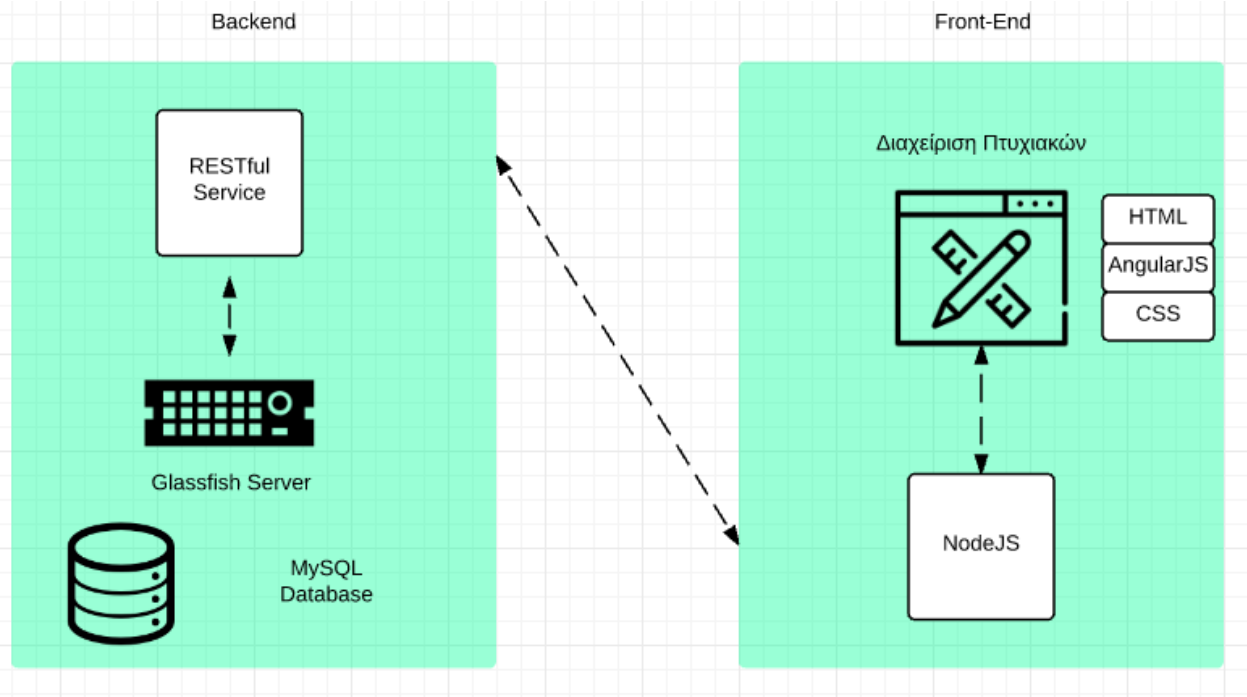
Εικόνα 6: Δημιουργία πτυχιακής



Εικόνα 7: Επιλογή πτυχιακής

3 Υλοποίηση Εφαρμογής

Στο παρόν κεφάλαιο θα περιγράψουμε πως από την σχεδίαση του προηγούμενου κεφαλαίου υλοποιήσαμε την εφαρμογή. Θα δούμε αρχικά την υλοποίηση της βάσης δεδομένων, στην συνέχεια της διαδικτυακής υπηρεσίας REST καθώς και την διεπαφή όπως φαίνονται στο παρακάτω σχήμα. Για την βάση δεδομένων χρησιμοποιήθηκε η δημοφιλής MySQL ενώ για την υλοποίηση της διαδικτυακής υπηρεσίας χρησιμοποιήθηκε η γλώσσα προγραμματισμού Java με το ολοκληρωμένο περιβάλλον εργασίας NetBeans, το οποίο με την χρήση της βιβλιοθήκης Jersey μπορούμε με απλό τρόπο να δημιουργούμε Restful υπηρεσίες. Τέλος για την υλοποίηση του front-end που περιλαμβάνει την διεπαφή καθώς και την λογική του προγραμματισμού χρησιμοποιήθηκαν η HTML5, CSS και AngularJS τα οποία θα αναλυθούν στην συνέχεια.



Εικόνα 8: Υλοποίηση Συστήματος

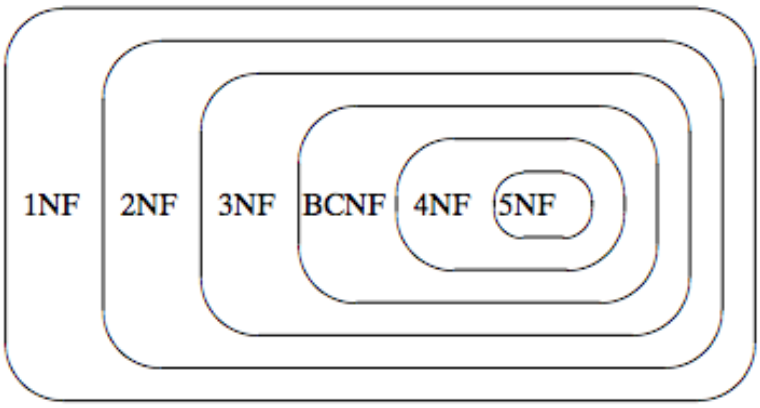
3.1 MySQL

Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων. Με τη χρήση της MySQL είναι εύκολη η πρόσβαση σ' αυτές τις πληροφορίες χρησιμοποιώντας μια γλώσσα συγγραφής σεναρίων στην πλευρά του διακομιστή (server-side scripting languages). Η MySQL αναπτύσσεται, διανέμεται, και υποστηρίζεται από την εταιρεία MySQL AB, η οποία είναι μια εμπορική επιχείρηση ανάπτυξης ανοικτού λογισμικού. Ανοικτό λογισμικό, σημαίνει ότι είναι δυνατό για οποιονδήποτε να χρησιμοποιήσει και να τροποποιήσει το λογισμικό, χωρίς καμία οικονομική επιβάρυνση. Ο διακομιστής MySQL ελέγχει την πρόσβαση στα δεδομένα μας για να

διασφαλίσει ότι πολλοί χρήστες θα μπορούν να δουλεύουν ταυτόχρονα, για να παρέχει γρήγορη πρόσβαση και για να διασφαλίσει ότι μόνο οι πιστοποιημένοι χρήστες θα μπορούν να έχουν πρόσβαση.

Η MySQL λειτουργεί με το μοντέλο Πελάτη/Εξυπηρετητή (Client/Server). Ο εξυπηρετητής δέχεται αιτήματα από τους πελάτες σχετικά με τη διαχείριση μίας ή περισσότερων βάσεων δεδομένων και προβαίνει στις απαιτούμενες ενέργειες. Οι κύριες λειτουργίες διαχείρισης βάσεων δεδομένων τις οποίες υλοποιεί ο εξυπηρετητής μετά από αίτηση του πελάτη, δε διαφέρουν σημαντικά από τις αντίστοιχες λειτουργίες των άλλων εφαρμογών βάσεων δεδομένων και περιλαμβάνουν τη δημιουργία-διαγραφή βάσεων δεδομένων, την εισαγωγή τροποποίηση-διαγραφή πινάκων (tables) και πεδίων (fields), την εισαγωγή τροποποίηση-διαγραφή εγγραφών (records) και τέλος την ανάκτηση δεδομένων από τη βάση με τη χρήση συγκεκριμένων κριτηρίων. Για την υλοποίηση των παραπάνω λειτουργιών, η MySQL χρησιμοποιεί την SQL (Δομημένη Γλώσσα Ερωτήσεων – Structured Query Language). Η SQL είναι η ευρύτερα χρησιμοποιούμενη τυποποιημένη γλώσσα πρόσβασης στις βάσεις δεδομένων και αποτελείται από εντολές οι οποίες επιτρέπουν την ανάκτηση και ενημέρωση δεδομένων σε μια βάση. Εκτός από τη MySQL, η γλώσσα SQL συνεργάζεται με άλλα προγράμματα βάσεων δεδομένων όπως είναι η Access, Microsoft SQL Server, Oracle, Sybase κ.α.

Για την βάση δεδομένων της εφαρμογής δημιουργήσαμε το σχεσιακό σχήμα από το διάγραμμα οντοτήτων-συσχετίσεων χρησιμοποιώντας τεχνικές ομαλοποίηση της βάσης όπως τις θεωρίες κανονικότητας. Η κανονικοποίηση αποτελεί ένα σημαντικό βήμα κατά το σχεδιασμό μιας βάσης δεδομένων. Οι πίνακες που προκύπτουν μετά την μετατροπή του μοντέλου οντοτήτων συσχετίσεων συνήθως περιέχουν επαναλαμβανόμενη πληροφορία. Για παράδειγμα, η ύπαρξη ενός χαρακτηριστικού πολλαπλών τιμών ενός συνόλου οντοτήτων έχει ως αποτέλεσμα να αποθηκευτούν πολλαπλές γραμμές στον αντίστοιχο πίνακα. Η κανονικοποίηση σχέσεων ουσιαστικά είναι μια απλοποίηση των σχέσεων με στόχο την εξάλειψη του πλεονασμού των δεδομένων ο οποίος δημιουργεί διάφορα προβλήματα. Για την κανονικοποίηση ακολουθούμε τις κανονικές μορφές που στο σύνολο τους είναι έξι. Η πιο κρίσιμη βέβαια για το σχεσιακό μοντέλο αποτελεί η πρώτη κανονική μορφή ενώ οι υπόλοιπες είναι προαιρετικές.



Εικόνα 9: Κανονικές Μορφές

Στην συνέχεια της κανονικοποίηση είναι η υλοποίηση της βάσης χρησιμοποιώντας την γλώσσα SQL. Ενδεικτικά μερικές εντολές είναι:

- *Create Schema*: Για την δημιουργία του σχήματος.
- *Create Table*: Για την δημιουργία ενός πίνακα
- *Insert into * Values **: Εισαγωγή δεδομένων σε έναν πίνακα

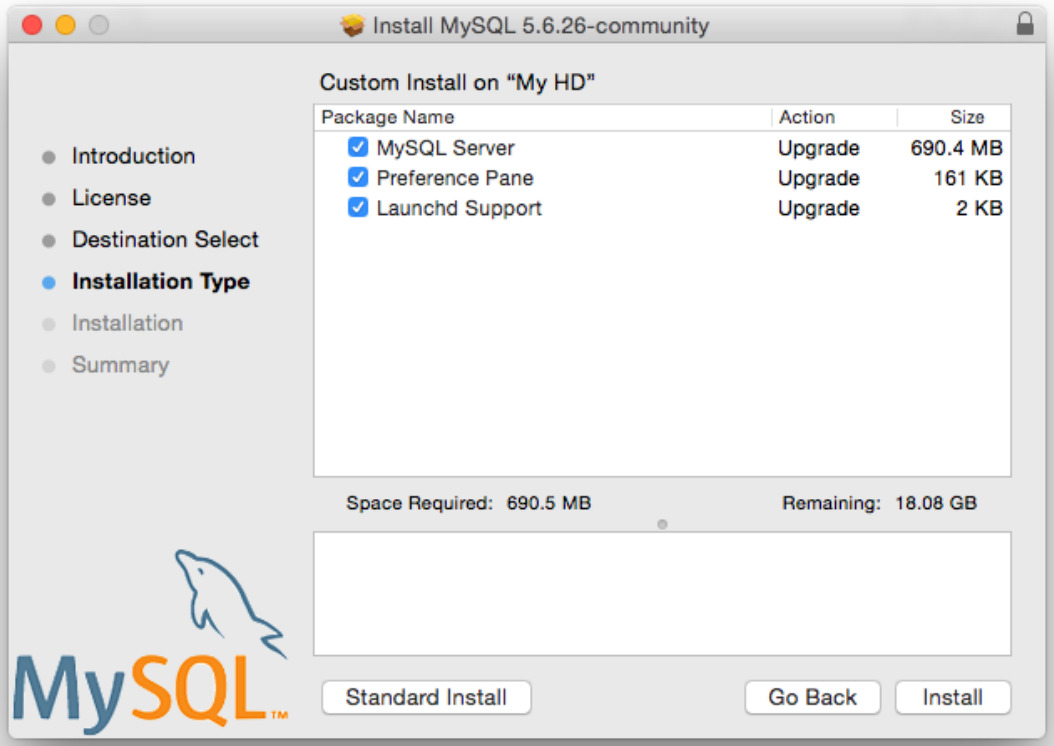
3.1.1 Εγκατάσταση MySQL Server & MySQL Workbench

Για την εγκατάσταση του MySQL Server Community Edition ακολουθούμε τα εξής βήματα:

- Ακολουθούμε τον σύνδεσμο http://dev.mysql.com/downloads/mysql/. Ο σύνδεσμος αυτός περιέχει τις δωρεάν εκδόσεις ανοιχτού κώδικα.
- Διαλέγουμε πλατφόρμα που θέλουμε να εγκαταστήσουμε
- Κατεβάζουμε και ανοίγουμε το εκτελέσιμο

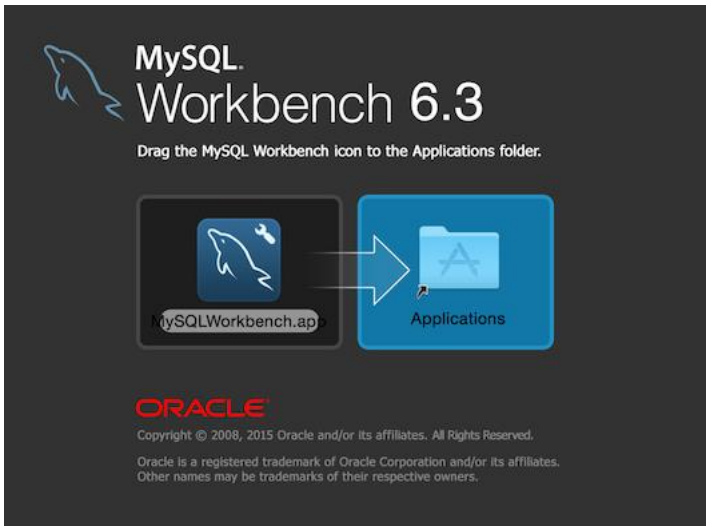
Εικόνα 10: Εγκατάσταση MySQL Server

- Ακολουθούμε τις προεπιλεγμένες ρυθμίσεις



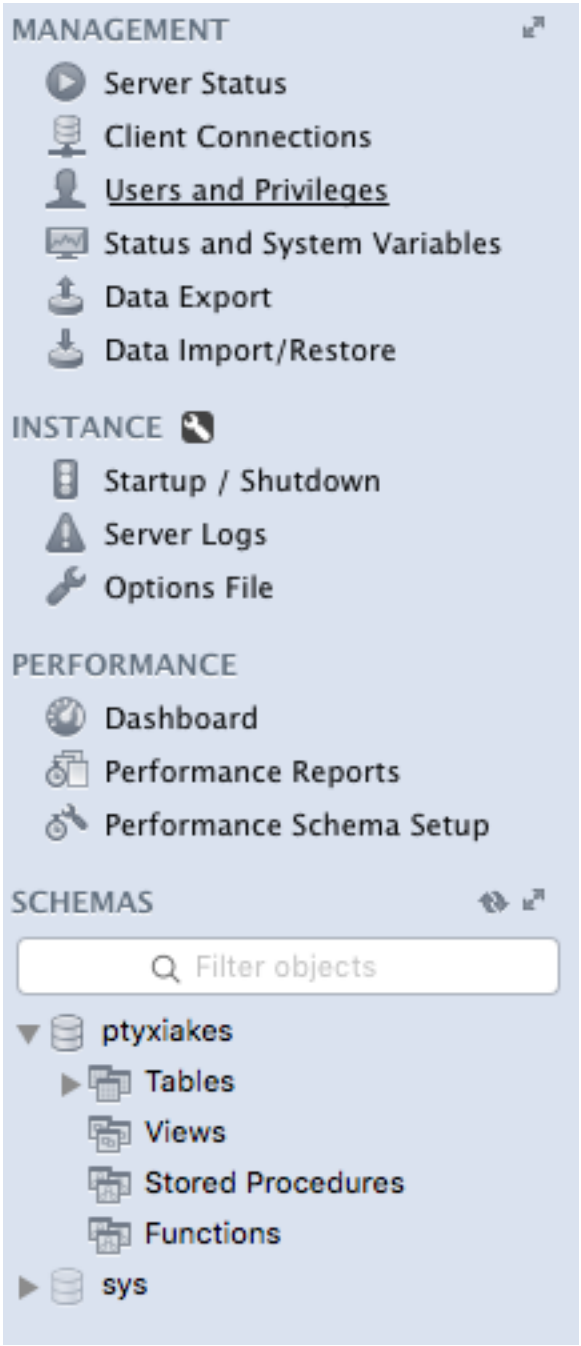
Εικόνα 11: Εγκατάσταση MySQL (2)

Στην συνέχεια θα εγκαταστήσουμε ένα βοηθητικό εργαλείο το MySQL Workbench το οποίο μας βοηθάει στην διαχείριση των σχημάτων που υπάρχουν στην βάση δεδομένων.



Εικόνα 12: MySQL Workbench

Αφού εγκαταστήσουμε την βοηθητική εφαρμογή μπορούμε να δούμε και να διαχειριστούμε όλα τα σχήματα της βάσης χωρίς να γράφουμε SQL queries. Στην παρούσα εργασία το workbench μας βοήθησε στο να δούμε αν όλα τα δεδομένα στην βάση είναι ορθά στις δοκιμές της διαδικτυακής υπηρεσίας που υλοποιήθηκε.

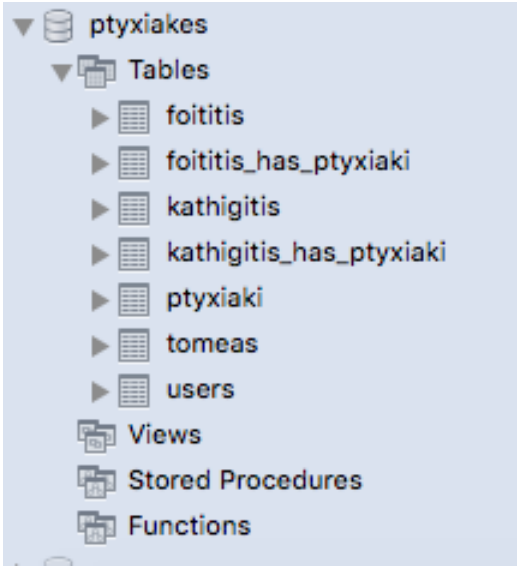


Εικόνα 13: MySQL Workbench Management

Με το MySQL Workbench μπορούμε να εκτελέσουμε ενέργειες όπως:

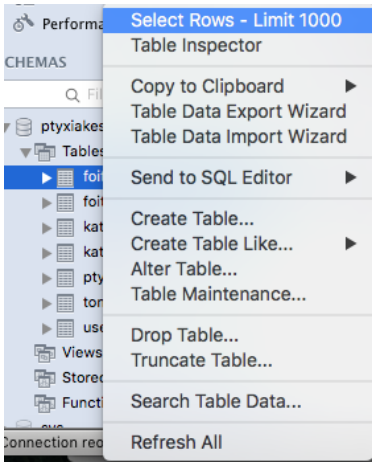
- εκκίνηση ή σταμάτημα της βάσης,
- εξαγωγή δεδομένων βάσης και σχημάτων,
- εισαγωγή δεδομένων στην βάση από αρχεία,
- εμφάνιση ή εκτύπωση επιδόσεων βάσης,
- δημιουργία σχεσιακής βάσης από σχεδιάγραμμα οντοτήτων συσχετίσεων
- κ.α

Στην παρακάτω εικόνα διακρίνεται το σχήμα ptyxiakes το οποίο περιγράφεται στο επόμενο υποκεφάλαιο



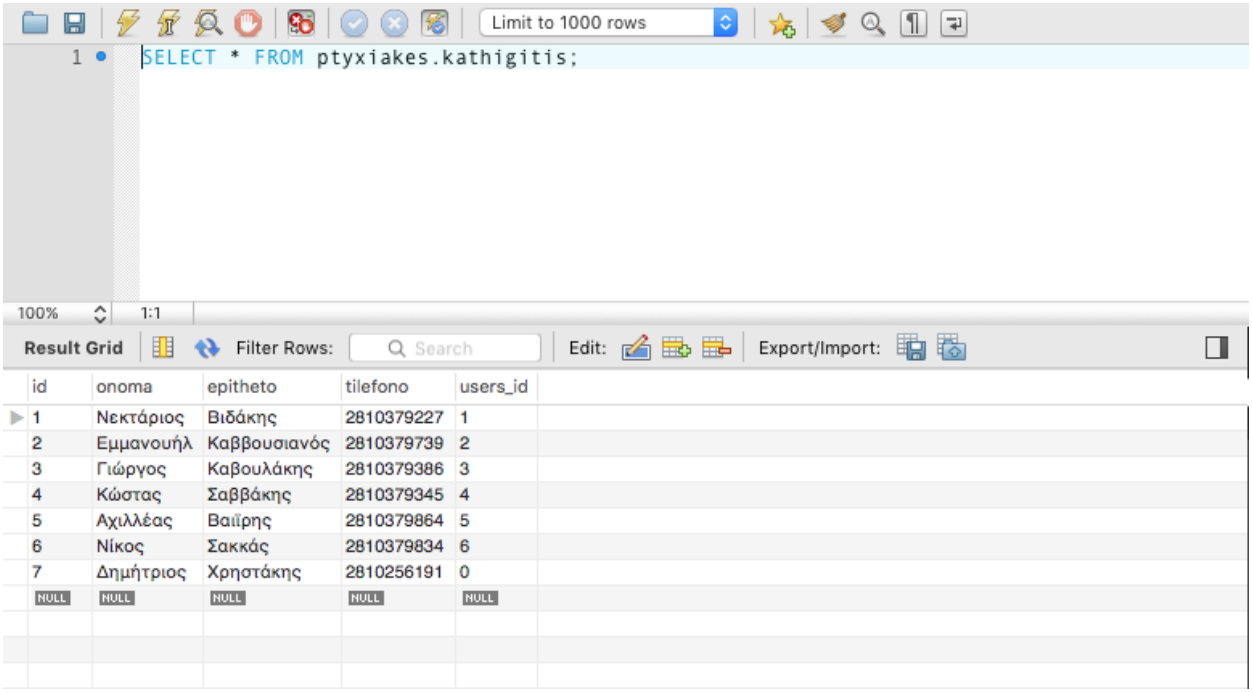
Εικόνα 14: Πίνακεσ βάσης

Σε κάθε πίνακα της βάσης μπορούμε δούμε τι δεδομένα έχει επιλέγοντας τον πίνακα και πατώντας δεξί κλικ. Στην συνέχεια πατώντας στο "Select Rows – Limit 1000"



Εικόνα 15: Εμφάνιση δεδομένων πίνακα

Στην συνέχεια μας εμφανίζεται στο παράθυρο όλες οι εγγραφές του πίνακα. Μπορούμε και από την εφαρμογή να αλλάξουμε δεδομένα ή να εισάγουμε δεδομένα. Με τον τρόπο αυτό η διαχείριση της βάσης γίνεται γρηγορότερη και πιο αποδοτική διότι δεν χρειάζεται να φτιάχνουμε συνεχώς νέα ερωτήματα στην βάση.



Εικόνα 16: Εγγραφές Πίνακα

3.1.2 Δημιουργία της σχεσιακής βάσης δεδομένων

Ενώ παρακάτω εμφανίζεται ο κώδικας που αναπτύχθηκε για την δημιουργία του πίνακα **foititis**..

```
CREATE TABLE `foititis` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `am` int(11) DEFAULT NULL,
  `onoma` varchar(45) DEFAULT NULL,
  `epitheto` varchar(45) DEFAULT NULL,
  `etosEisagogis` varchar(11) DEFAULT NULL,
  `users_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `am_UNIQUE` (`am`),
  KEY `fk_foititis_users_idx` (`users_id`),
  CONSTRAINT `fk_foititis_users` FOREIGN KEY (`users_id`) REFERENCES `users` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
)
```

Ο πίνακας foitis έχει πρωτεύον κλειδί γιατί θέλουμε να διασφαλίσουμε την μοναδικότητα κάθε εγγραφής. Θα μπορούσαμε στην παρούσα φάση να χρησιμοποιούσαμε μαζί και το ΑΜ του φοιτητή ως συμπληρωματικό πρωτεύον κλειδί αλλά δεν ήταν απαραίτητο λόγω της μη πολύπλοκης εφαρμογής.

Για τον ενδιάμεσο πίνακα **foititis_has_ptyxiaki** στον οποίο εγγράφονται οι φοιτητές που έχουν

```
CREATE TABLE `foititis_has_ptyxiaki` (
  `foititis_id` int(11) NOT NULL,
  `ptyxiaki_id` int(11) NOT NULL,
  PRIMARY KEY (`foititis_id`,`ptyxiaki_id`),
  KEY `fk_foititis_has_ptyxiaki_ptyxiaki1_idx` (`ptyxiaki_id`),
  KEY `fk_foititis_has_ptyxiaki_foititis1_idx` (`foititis_id`),
  CONSTRAINT `fk_foititis_has_ptyxiaki_foititis1` FOREIGN KEY (`foititis_id`) REFERENCES `foititis` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_foititis_has_ptyxiaki_ptyxiaki1` FOREIGN KEY (`ptyxiaki_id`) REFERENCES `ptyxiaki` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

αναλάβει μια συγκεκριμένη πτυχιακή:

Στον πίνακα αυτό πρωτεύον κλειδιά είναι το id του φοιτητή καθώς και το id της πτυχιακής. Έτσι διασφαλίζουμε μια μοναδική εγγραφή ενός φοιτητή με την συγκεκριμένη πτυχιακή.

Για τον πίνακα **kathigitis**:

```
CREATE TABLE `kathigitis` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `onoma` varchar(45) DEFAULT NULL,
  `epitheto` varchar(45) DEFAULT NULL,
  `tilefono` varchar(45) DEFAULT NULL,
  `users_id` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_kathigitis_users1_idx` (`users_id`),
  CONSTRAINT `fk_kathigitis_users1` FOREIGN KEY (`users_id`) REFERENCES `users` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
)
```

Στον πίνακα kathigitis έχει πρωτεύον κλειδί το id γιατί θέλουμε να διασφαλίσουμε την μοναδικότητα κάθε εγγραφής.

Για τον ενδιάμεσο πίνακα **kathigitis_has_ptyxiaki** στον οποίο αναγράφονται όσοι καθηγητές επιβλέπουν πτυχιακές.

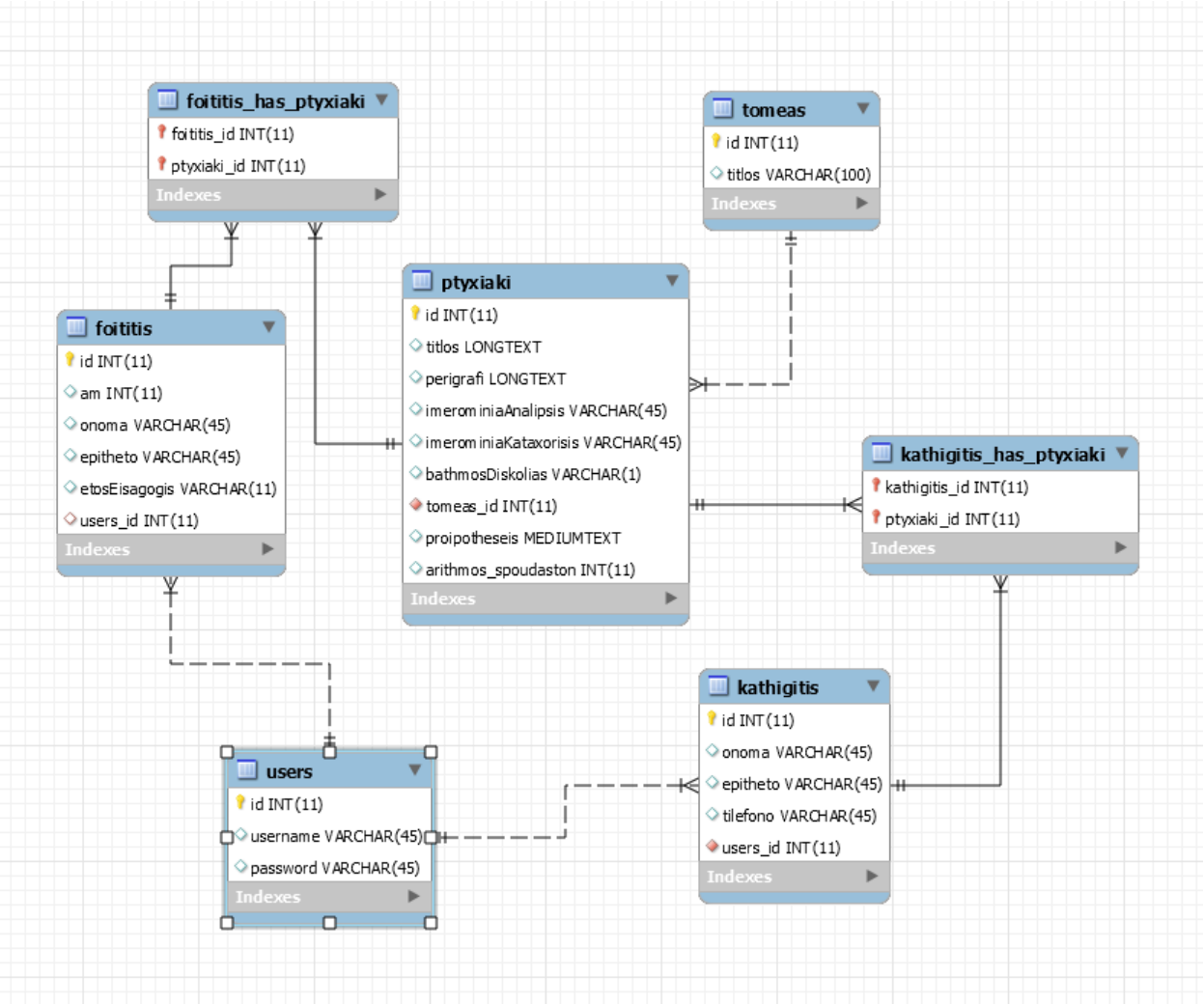
Στον πίνακα αυτό πρωτεύον κλειδιά είναι το id του καθηγητή καθώς και το id της πτυχιακής. Έτσι διασφαλίζουμε μια μοναδική εγγραφή ενός καθηγητή με την συγκεκριμένη πτυχιακή.

```
CREATE TABLE `kathigitis_has_ptyxiaki` (
  `kathigitis_id` int(11) NOT NULL,
  `ptyxiaki_id` int(11) NOT NULL,
  PRIMARY KEY (`kathigitis_id`,`ptyxiaki_id`),
  KEY `fk_kathigitis_has_ptyxiaki_ptyxiaki1_idx` (`ptyxiaki_id`),
  KEY `fk_kathigitis_has_ptyxiaki_kathigitis1_idx` (`kathigitis_id`),
  CONSTRAINT `fk_kathigitis_has_ptyxiaki_kathigitis1` FOREIGN KEY (`kathigitis_id`) REFERENCES `kathigitis` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_kathigitis_has_ptyxiaki_ptyxiaki1` FOREIGN KEY (`ptyxiaki_id`) REFERENCES `ptyxiaki` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
)
```

Για τον πίνακα **ptyxiaki**:

```
CREATE TABLE `ptyxiaki` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `titlos` longtext,
  `perigrafi` longtext,
  `imerominiaAnalipsis` varchar(45) DEFAULT NULL,
  `imerominiaKataxorisis` varchar(45) DEFAULT NULL,
  `bathmosDiskolias` varchar(1) DEFAULT NULL,
  `tomeas_id` int(11) NOT NULL,
  `proipotheseis` mediumtext,
  `arithmos_spoudaston` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_ptyxiaki_tomeas1_idx` (`tomeas_id`),
  CONSTRAINT `fk_ptyxiaki_tomeas1` FOREIGN KEY (`tomeas_id`) REFERENCES `tomeas` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION
`
```

Αφού τρέξουμε τον κώδικα χρησιμοποιώντας το εργαλείο MySQL Workbench μπορούμε να δούμε γραφικά την βάση που μόλις φτιάξαμε



Εικόνα 17 Μοντέλο ΟΣ στο Workbench

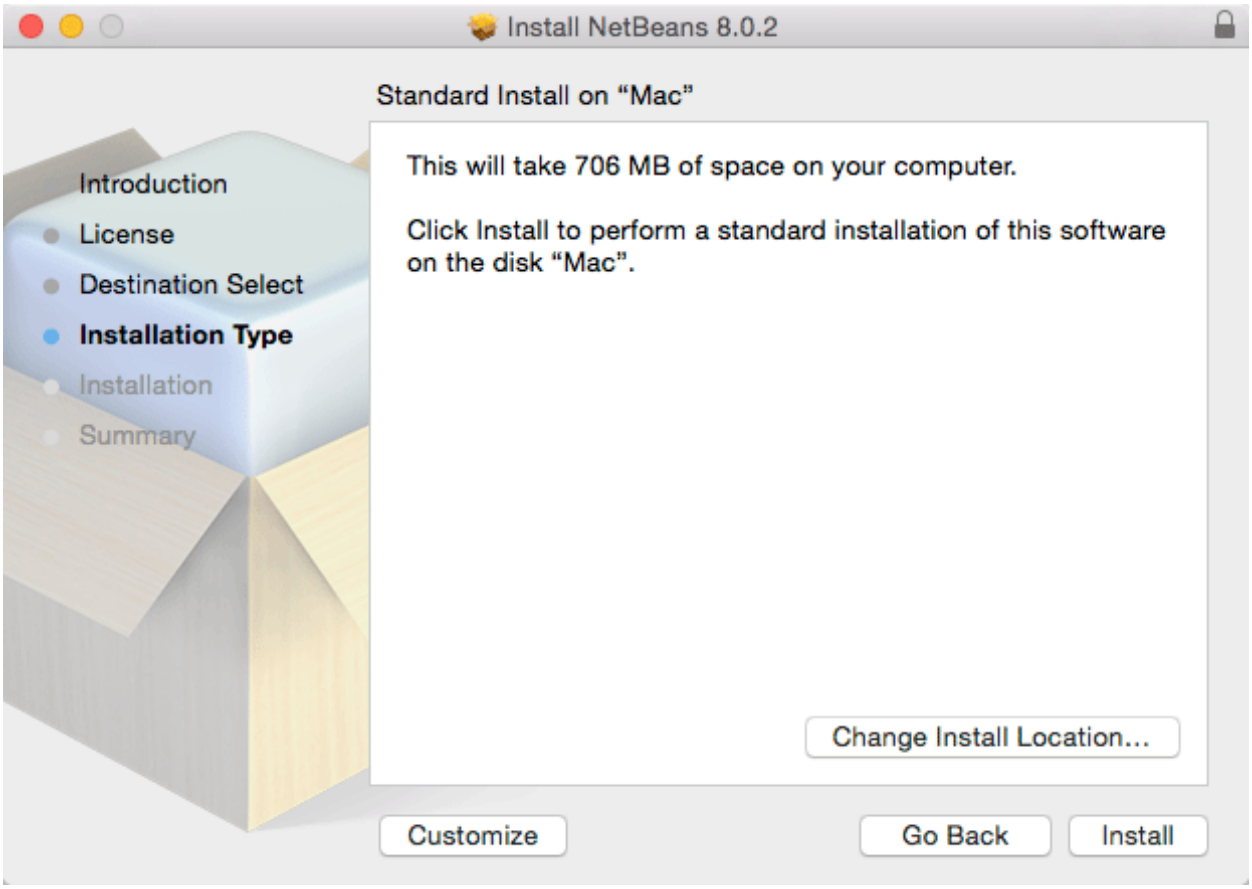
3.2 RESTful API

Για την υλοποίηση της υπηρεσίας χρησιμοποιήθηκε η γλώσσα προγραμματισμού Java και το ενσωματωμένο περιβάλλον ανάπτυξης (IDE) Netbeans. Η ανάπτυξη REST υπηρεσιών στο περιβάλλον του Netbeans στηρίζεται στην χρήση του Java API για rest υπηρεσίες JAX-RS και στην υλοποίηση του, η οποία αποτελεί μέρος του πακέτου Jersey. Συγκεκριμένα για να δημιουργήσουμε μια REST υπηρεσία με το Netbeans είναι απαραίτητο να δημιουργήσουμε ένα νέο project το οποίο να είναι Web Application. Στην συνέχεια επιλέγουμε να δημιουργήσουμε οντότητες από μια βάση δεδομένων (Entities from database). Η ενέργεια αυτή θα μας φτιάξει ένα Java Bean για κάθε πίνακα στην βάση 1 προς 1 με όλα τα γνωρίσματα.

3.2.1 Εγκατάσταση Netbeans

Για την εγκατάσταση του Netbeans ακολουθούμε τα εξής βήματα:

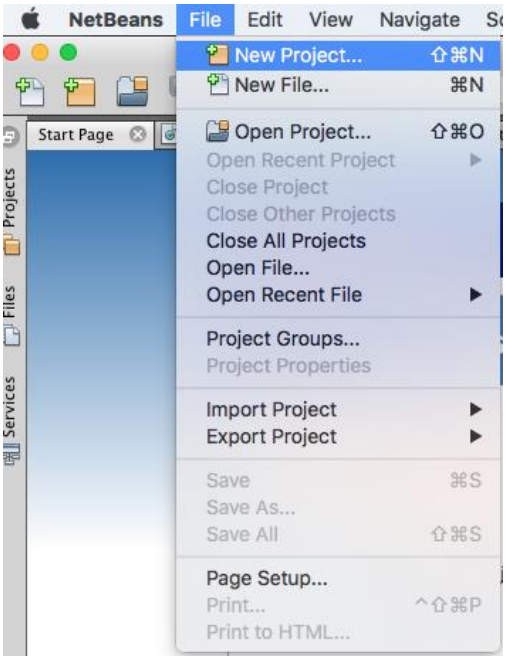
- Μεταβαίνουμε στην ιστοσελίδα https://netbeans.org/downloads/index.html
- Κατεβάζουμε και ανοίγουμε το εκτελέσιμο
- Ακολουθούμε τις προεπιλεγμένες ρυθμίσεις



Εικόνα 18: Εγκατάσταση Netbeans

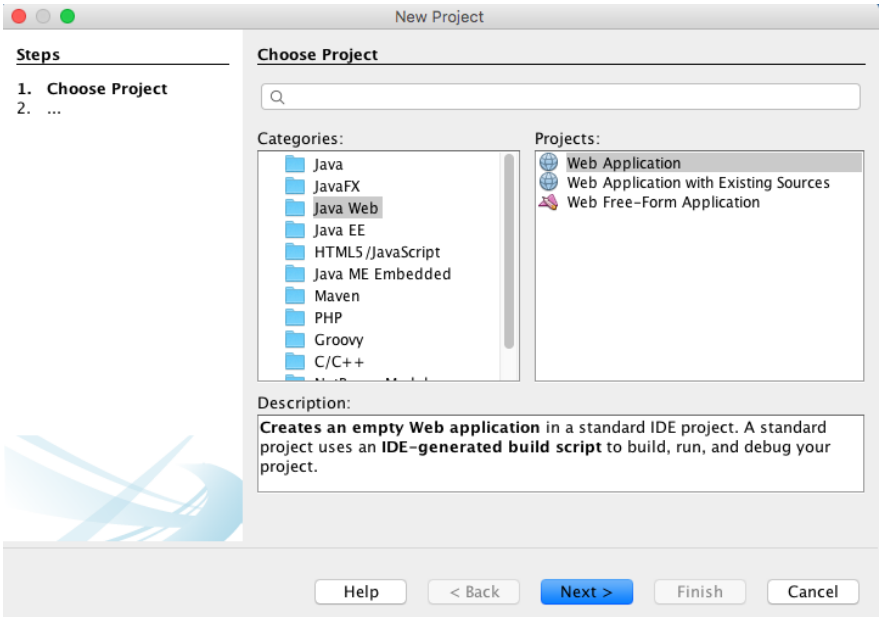
3.2.2 Δημιουργία Project

Για την δημιουργία project επιλέγουμε File->New Project.



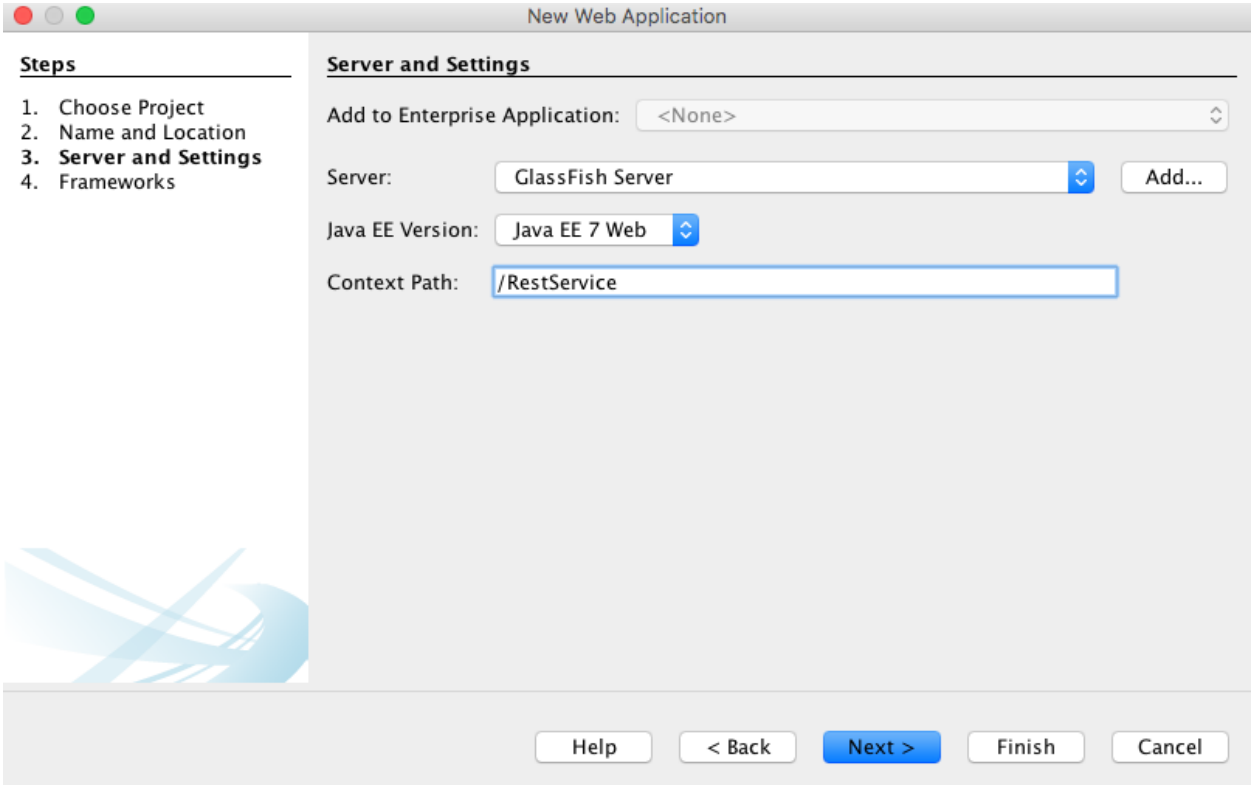
Εικόνα 19: Δημιουργία Project

Στην συνέχεια θα πρέπει να δημιουργήσουμε ένα Web Application καθώς η διαδικτυακή υπηρεσία θα τρέχει σε έναν server. (GlassFish Server)



Εικόνα 20: Επιλογή είδος project στο Netbeans

Αφού επιλέξουμε το όνομα του project καθώς και σε ποια τοποθεσία θα αποθηκευτεί επιλέγουμε σε ποιον τύπο server θα εγκατασταθεί ώστε να τρέχει η εφαρμογή. Στην παρούσα εφαρμογή επιλέγουμε τον Glassfish server.



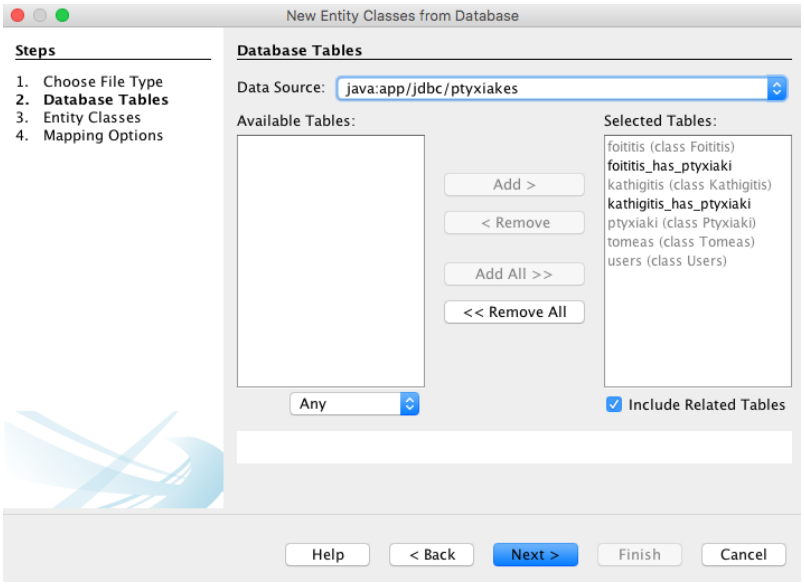
Εικόνα 21: Επιλογή server Glassfish

Στο τέλος του οδηγού το Netbeans θα μας φτιάξει όλες τα απαραίτητα αρχεία και ρυθμίσεις ώστε να μπορούμε να τρέχουμε την εφαρμογή στον server.

3.2.3 Δημιουργία οντοτήτων από την βάση

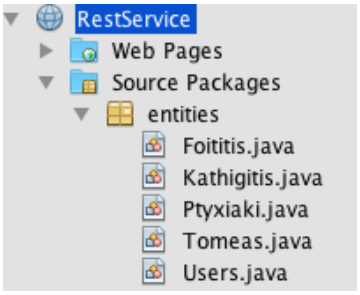
Στην συνέχεια θα πρέπει στο project μας να φτιάξουμε κάποιες κλάσεις οι οποίες αντιπροσωπεύουν τους πίνακες στην βάση έναν-προς-ένα. Το περιβάλλον Netbeans έχει οδηγό το οποίο εισάγεις την διεύθυνση, username, και κωδικό της βάσης και ανακτά όλα τα σχήματα. Διαλέγουμε ποιους πίνακες θέλουμε να συμπεριλάβουμε και κάθε γνώρισμα ενός πίνακα γίνεται μεταβλητή της κλάσης και κάθε πίνακας γίνεται κλάση.

Για να εκτελέσουμε τον οδηγό αυτό στο Netbeans με δεξί κλίκ επιλέγουμε στο project να δημιουργήσουμε *Restful web service from entities.*



Εικόνα 22: Δημιουργία αντικειμένων απο βάση

Επιλέγουμε το σχήμα αλλά και από ποιους πίνακες θέλουμε να δημιουργήσουμε Java Bean οντότητες.

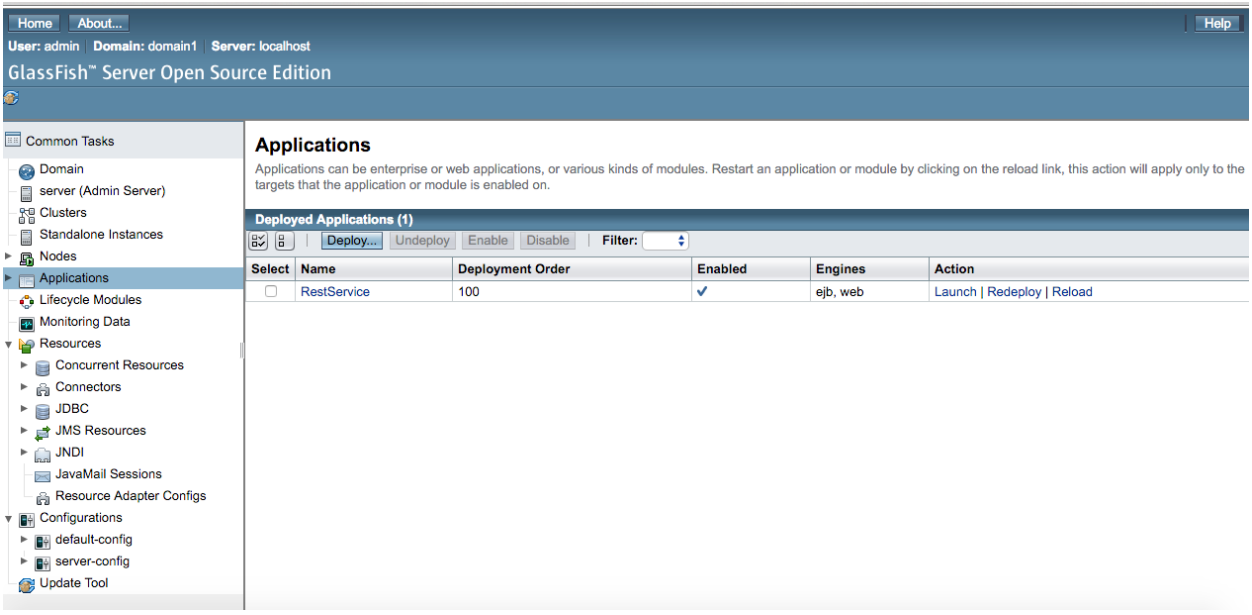


Εικόνα 23: Java Beans απο βάση

Στο τέλος του οδηγού επιπλέον φτιάχνονται όλες οι απαραίτητες κλάσεις και μέθοδοι ώστε οι πόροι μας να έχουν διάφορες αναπαραστάσεις αν ζητηθεί από τον χρήστη.

```
17  import javax.ws.rs.PUT;
18  import javax.ws.rs.Path;
19  import javax.ws.rs.PathParam;
20  import javax.ws.rs.Produces;
21  import javax.ws.rs.core.MediaType;
22
23  /**
24   *
25   * @author Rena
26   */
27  @Stateless
28  @Path("foititis")
    public class FoititisFacadeREST extends AbstractFacade<Foititis> {
30
31      @PersistenceContext(unitName = "RestServicePU")
32      private EntityManager em;
33
34      public FoititisFacadeREST() {
35          super(Foititis.class);
36      }
37
38      @POST
39      @Override
40      @Consumes({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
        public void create(Foititis entity) {
42          super.create(entity);
43      }
44
45      @PUT
46      @Path("{id}")
47      @Consumes({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
```

Αφού φτιάξουμε την διαδικτυακή υπηρεσία θα πρέπει να εγκατασταθεί σε έναν web server ώστε να μπορούμε να την χρησιμοποιήσουμε. Για τις ανάγκες της πτυχιακής χρειάστηκε ένας ελαφρύς και εύκολος στην χρήση web server καθώς δεν είναι απαιτητική η υπηρεσία που φτιάξαμε. Καταλήξαμε στον web server Glassfish 4 από την Oracle ο οποίος είναι λογισμικό ανοιχτού κώδικα και δωρεάν στην χρήση. Για την εγκατάσταση εφαρμογών η διαδικασία είναι εύκολη καθώς υπάρχει ένα web portal στο οποίο όταν μπούμε μπορούμε να του μεταφορτώσουμε τις εφαρμογές μας.

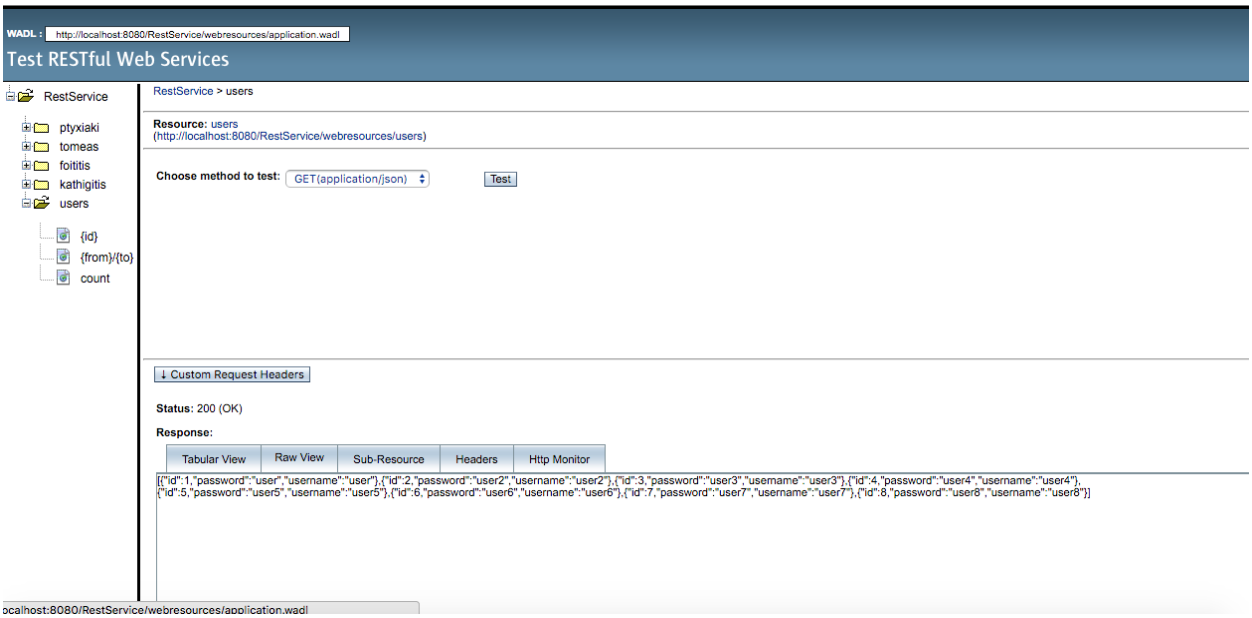


Εικόνα 24: Glassfish Management

Ανοίγοντας το portal του glassfish server μπορούμε να εκτελέσουμε διάφορες ενέργειες όπως:

- Φόρτωση web εφαρμογών
- Εκκίνηση, παύση ή σταμάτημα εφαρμογών
- Διαχείριση χρηστών του server
- Εισαγωγή Java βιβλιοθηκών για υποστήριξη σε διάφορες εφαρμογές
- Εισαγωγή Connectors για την σύνδεση με βάσεις δεδομένων
- Παρακολούθηση και διαχείριση πόρων του server

Ο οδηγός του προηγούμενου βήματος μας έχει δημιουργήσει και μια ιστοσελίδα ώστε να ελέγχουμε την ορθότητα της υπηρεσίας. Μπορούμε να επιλέξουμε κάθε διαθέσιμο πόρο και δούμε τις διαθέσιμες αναπαραστάσεις με τις HTTP μεθόδους.



Εικόνα 25: Δοκιμή της Restful υπηρεσίας

3.3 Node.js & AngularJS

Η υλοποίηση της διεπαφής βασίστηκε εξολοκλήρου στην βιβλιοθήκη AngularJS της Google. Η βιβλιοθήκη AngularJS έχει γίνει η "defacto" επιλογή για την ανάπτυξη διαδικτυακών εφαρμογών ενώ το NodeJS χρησιμοποιείται κυρίως για εφαρμογές που χρειάζονται γρήγορη ανάπτυξη και δεν έχουν τρομερά πολλές απαιτήσεις.

3.3.1 AngularJS

Η πρώτη έκδοση σε AngularJS κυκλοφόρησε το 2012. Το AngularJS αναπτύχθηκε αρχικά το 2009 από έναν υπάλληλο της Google, τον Misko Hevery οποίος άρχισε να εργάζεται πάνω σε αυτό. Η ιδέα του τελικώς αποδείχθηκε πολύ καλή και το έργο του πλέον υποστηρίζεται από την Google. Το AngularJS είναι ένα open-source πλαίσιο ανάπτυξης Web, το οποίο συντηρείται από την Google και είναι γραμμένο σε JavaScript. Το AngularJS είναι χτισμένο γύρω από την πεποίθηση ότι ο δηλωτικός προγραμματισμός θα πρέπει να χρησιμοποιείται για την κατασκευή της διεπαφής χρήστη και τη σύνδεση στοιχείων λογισμικού. Λέγοντας δηλωτικός προγραμματισμός(declarative programming) εννοούμε τις επιθυμητές ιδιότητες ενός ζητούμενου

αποτελέσματος, ένα παράδειγμα είναι η γλώσσα HTML και η CSS κ.ά. Στόχος του είναι να απλοποιεί την ανάπτυξη και τη δοκιμή των εφαρμογών με την παροχή ενός πλαισίου για client-side, model-view-controller (MVC) αρχιτεκτονική. Το MVC αποτελείται από τρία βασικά μέρη: το οπτικό μέρος, το μοντέλο δεδομένων που περιέχει αντικείμενα σε JavaScript της εκάστοτε εφαρμογής και έναν ελεγκτή που προσθέτει συμπεριφορά σε προστακτικές γλώσσες (imperative languages). Εάν ο χρήστης θέλει να αλληλοεπιδράσει με το εικαστικό μέρος, καλεί τον ελεγκτή (controller) και χειρίζεται το μοντέλο το οποίο αλληλοεπιδρά με τον διακομιστή (server). Το AngularJS ανιχνεύει τις αλλαγές του μοντέλου και ενημερώνει το view.

Στην εφαρμογή χρησιμοποιήθηκε εκτενώς η βιβλιοθήκη για την υλοποίηση της λογικής αλλά και για την διεπαφή. Συνοπτικά για κάθε view της εφαρμογής υπάρχει και ένας controller που χειρίζεται την απαραίτητη λογική ή πραγματοποιεί κλήση στην Rest υπηρεσία. Ενδεικτικός κώδικας παρουσιάζεται παρακάτω για το view της δημιουργίας νέας πτυχιακής.

View

```
<%- include nav.ejs %>
<link rel='stylesheet' href='/stylesheets/textAngular.css'>
<link rel='stylesheet' href='/stylesheets/font-awesome.min.css'>
<div class="container" ng-controller="createCtl">
  <div layout = "row">
    <h2 class="md-display-1" flex="70">Δημιουργία Νέας Πτυχιακής</h2>
    <md-button class="md-primary md-raised" ng-click="saveData()" flex="30" flex-gt-md="auto">
    Αποθηκευση αλλαγων
  </md-button>
</div>
<md-content md-theme="docs-dark">
<div layout="row">
  <md-input-container flex="70">
  <label>Τίτλος</label>
  <input ng-model="ptyxiaki.titlos">
</md-input-container>

<md-input-container class="md-block" flex="30">
<label>Εισηγητής</label>
<md-select placeholder="Επιλογή Εισηγητή" ng-model="professor" md-on-open="getProfessors()">
<md-option ng-value="professor" ng-repeat="professor in professors">{{professor.onoma}} {{professor.epitheto}}</md-option>
</md-select>
</md-input-container>

</div>
<div layout="row">
  <md-input-container>
  <label>Τηλέφωνο</label>
  <input ng-model="professor.tilefono">
</md-input-container>
</div>
</md-content>

<md-content layout-padding="">
<div>
  <form name="userForm">

    <div layout="row">
      <md-input-container flex="70">
      <label>Προϋποθέσεις Ανάληψης Πτυχιακής</label>
      <input ng-model="ptyxiaki.proipotheseis">
    </md-input-container>
    <md-input-container class="md-block" flex="30">
    <label>Τομέας</label>
    <md-select placeholder="Επιλογή τομέα" ng-model="ptyxiaki.tomeas" md-on-open="getTomeas()">
    <md-option ng-value="tomeas" ng-repeat="tomeas in tomeis">{{tomeas.titlos}}</md-option>
  </md-select>
</md-input-container>
</div>

<div layout-gt-sm="row">
  <md-input-container class="md-block" flex="30">
  <label>Εισηγητής (2)</label>
  <md-select placeholder="Επιλογή Εισηγητή (2)" ng-model="professor2" md-on-open="getProfessors2()">
  <md-option ng-value="professor2" ng-repeat="professor2 in professors">{{professor2.onoma}} {{professor2.epitheto}}</md-option>
</md-select>
</md-input-container>

<md-input-container class="md-block" flex="30">
<label>Βαθμός Δυσκολίας</label>
<md-select ng-model="ptyxiaki.diskolia">
<md-option ng-repeat="d in difficulty" value="{{d.name}}">
```

Εικόνα 26: Ενδεικτικό παράδειγμα View

Για την διεπαφή της εφαρμογή έχουμε υλοποιήσει μια μπάρα περιήγησης με επιλογές ώστε να επιλέγει ο χρήστης που θέλει να περιηγηθεί. Χρησιμοποιώντας την JavaScript μηχανή "templating" EJS μπορούμε να υλοποιήσουμε την μπάρα μια φορά και να την συμπεριλαμβάνουμε σε κάθε view μας ώστε να μην επαναλαμβάνουμε κώδικα συνεχώς. Η σύνδεση του controller με κάθε view γίνεται με την επιλογή "**ng-controller"** η οποία ανήκει στην βιβλιοθήκη της AngularJS. Στο controller μπορούμε πλέον να μοιραζόμαστε μεταβλητές δεδομένα χρησιμοποιώντας το πρόθεμα **$scope** στο controller για κάθε μεταβλητή.

```
<md-select ng-model="ptyxiaki.diskolia">
<md-option ng-repeat="d in difficulty" value="{{d.name}}">
{{d.name}}
</md-option>
</md-select>
</md-input-container>
<md-input-container>
<label>Α. Σπουδαστών</label>
<input ng-model="ptyxiaki.arithmos_spoudaston">
</md-input-container>

<div layout="column" flex>
  <label class="md-caption">Περίοδος Ανάληψης</label>
  <md-datepicker name="toDate" ng-model="ptyxiaki.periodos_analipsis" required md-placeholder=""></md-datepicker>
</div>
</div>


<div layout="row">
  <md-input-container >
  <label style="padding-bottom: 30px"> Περιγραφή πτυχιακής </label>
  <text-angular ng-model="htmlVariable"></text-angular>
</md-input-container>
</div>
<h2 class="md-display-1">Ανάθεση Πτυχιακής Σε Φοιτητές</h2>
<div layout="row" ng-repeat="student in students">
  <md-input-container class="md-block" flex-gt-xs="">
  <label>{{student.onoma}} {{student.epitheto}}</label>
  <input ng-model="student" disabled="">
</md-input-container>
</div>
<div layout="row">
  <md-input-container flex = "40">
  <label>Ονομα Φοιτητή</label>
  <input ng-model="tmpStudent.epitheto" disabled="">
</md-input-container>
<md-input-container flex = "20">
<label>ΑΜ φοιτητή</label>
<input ng-model="am" >
</md-input-container>
<md-button class="md-primary md-raised" ng-click="findStudent()" ng-disabled = "disable_findBtn" flex="20" flex-gt-md="auto">
Ευρεση Φοιτητων
</md-button>
<md-button class="md-primary md-raised" ng-click="addStudent()" ng-disabled = "tmpStudent==null" flex="20" flex-gt-md="auto">
Προσθηκη Φοιτητων
</md-button>
</div>

</form>
</div>
</md-content>

</div>
<!-- <%- include footer.ejs %> -->
```

Εικόνα 27: View της Δημιουργία πτυχιακής

Controller

```
app.controller('createCtl', [
  '$scope','DataFactory', '$http', '$mdDialog', '$mdMedia',
  function($scope, DataFactory, $http, $mdDialog, $mdMedia){

    $scope.ptyxiaki = {};

    $scope.professor = null;
    $scope.professor2 = null;
    $scope.professors =null;
    $scope.professors2 =null;


    $scope.getProfessors = function () {
     return DataFactory.getKathigitis()
     .success(function (data) {
       $scope.professors = data;
     });
   }

   $scope.getProfessors2 = function () {
     return DataFactory.getKathigitis()
     .success(function (data) {
       $scope.professors2 = data;
     });
   }

   $scope.tomeas = null;
   $scope.tomeis = null;

   $scope.getTomeas = function () {
    return DataFactory.getTomeas()
    .success(function (data){
      $scope.tomeis = data;
    });
  }

  $scope.am = null;
  $scope.disable_findBtn = false;
  $scope.disable_addBtn = true;
  $scope.tmpStudent = null;

  $scope.findStudent = function(){
    console.log($scope.am);
    if($scope.am!=null){
      DataFactory.getFoititis()
      .success(function (data){

        for (var i=0; i<data.length; i++){
          if(data[i].am == $scope.am){
            $scope.tmpStudent = data[i];
            $scope.disable_addBtn = false;
            break;
          }
        }
      });
    }

  }

  $scope.students = [];

  $scope.addStudent = function(){
```

Εικόνα 28: Controller του view Δημιουργία πτυχιακής

Στον controller της παραπάνω εικόνας παρουσιάζεται η λογική της κλήσης της Rest υπηρεσίας. Για τον σκοπό αυτό έχουμε υλοποιήσει ένα factory, DataFactory, το οποίο περιέχει μεθόδους για την κλήση της υπηρεσίας για κάθε διαφορετικό πόρο. Για παράδειγμα, η μέθοδος **DataFactory.getKathigitis()** επιστρέφει όλες τις εγγραφές των καθηγητών που υπάρχουν στην βάση μας. Αυτό είναι δυνατό εισάγοντας μια ειδική μεταβλητή στο factory, την $http, ώστε να πραγματοποιούνται http κλήσεις. Στην παρακάτω εικόνα παρουσιάζονται όλες οι μέθοδοι της DataFactory.

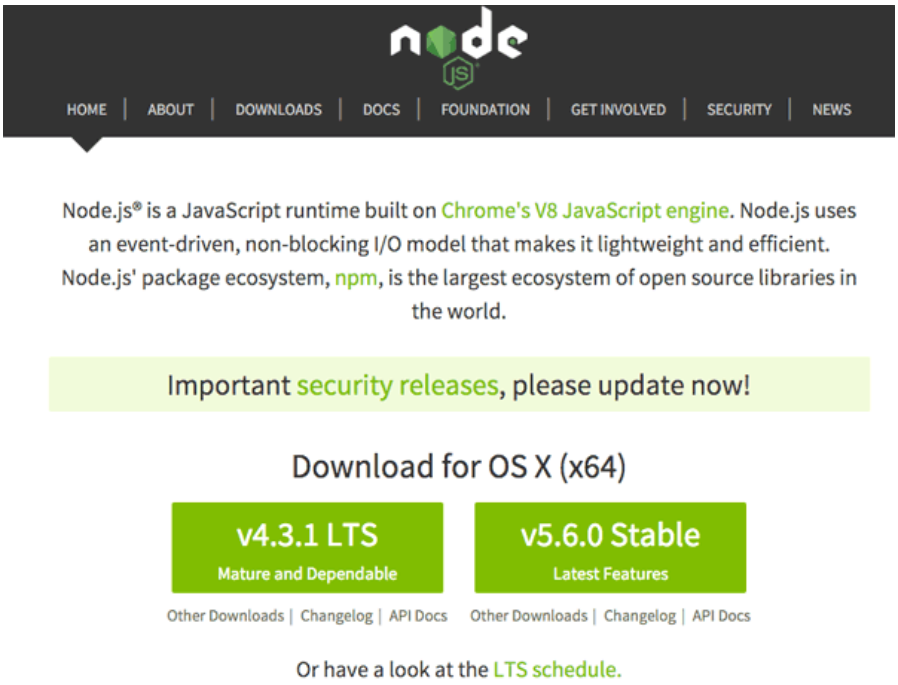
```
app.factory('DataFactory', ['$http', function ($http) {

  var urlBase = 'http://localhost:8080/RestService/webresources/';
  var dataFactory = {};

  dataFactory.getTomeas = function () {
    return $http.get(urlBase+'tomeas');
  };
  dataFactory.getFoititis = function () {
    return $http.get(urlBase+'foititis');
  };
  dataFactory.getPtyxiaki = function () {
    return $http.get(urlBase+'ptyxiaki');
  };
  dataFactory.getSpecificPtyxiaki = function (id) {
    return $http.get(urlBase+'ptyxiaki/'+id);
  };
  dataFactory.getKathigitis = function () {
    return $http.get(urlBase+'kathigitis');
  };
  dataFactory.getUser = function () {
    return $http.get(urlBase+'users');
  }
  dataFactory.getSpecificUser = function (id) {
    return $http.get(urlBase+'users/'+id);
  }
  return dataFactory;
}]);
```

3.3.2 NodeJS

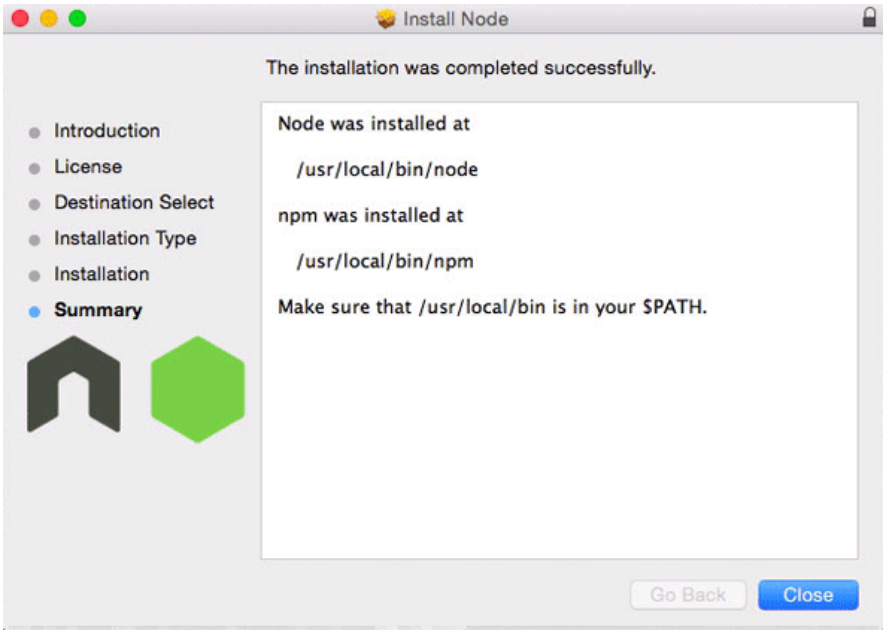
Το NodeJS είναι ανοιχτού κώδικα περιβάλλον runtime σε JavaScript για χρήση σε εφαρμογές που τρέχουν από την μεριά του διακομιστή. Είναι βασισμένο στο Google V8 Javascript Engine, που σημαίνει ότι οι εφαρμογές που φτιάχνονται για το NodeJS γράφονται σε JavaScript χρησιμοποιώντας ίδιο συντακτικό, αντικείμενα, μεθόδους που χρησιμοποιούνται σε εφαρμογές από την μεριά του πελάτη. Το NodeJS έρχεται με ενσωματωμένη βιβλιοθήκη που επιτρέπει εφαρμογές να συμπεριφέρονται σαν διακομιστή web ώστε να φιλοξενούν ιστοσελίδες.

3.3.2.1 Εγκατάσταση NodeJS

Για την εγκατάσταση του NodeJS μεταβαίνουμε στην ιστοσελίδα https://nodejs.org/en/download/ και διαλέγουμε την επιθυμητή πλατφόρμα. Στην συνέχεια εκτελούμε το εκτελέσιμο.



Εικόνα 29: Ιστότοπος NodeJS



Εικόνα 30: Εγκατάσταση NodeJS

Μετά την εγκατάσταση μας τοποθετεί το node αλλά και το npm σε τοποθεσία τέτοια ώστε να είναι διαθέσιμο από το τερματικό. Μερικές λειτουργίες του node και npm από το τερματικό είναι:

- *npm init:* Δημιουργεί ένα νέο προτζεκτ με τις προεπιλεγμένες ρυθμίσεις
- *npm install package:* Εγκαθιστά το πακέτο που έχουμε ορίσει
- *node filename.*js: τρέχει το αρχείο

3.3.2.2 Υλοποίηση δρομολόγησης με ExpressJS

Στην παρούσα εργασία το NodeJS επιλέχθηκε για τον ρόλο του διακομιστή που φιλοξενεί την εφαρμογή. Μερικά από τα πλεονεκτήματα που εχει το NodeJS έναντι συμβατικών διακομιστών είναι:

- Δεν χρησιμοποιεί πολλούς πόρους συστήματος και είναι αρκετά γρήγορο
- Εύκολο στην ρύθμιση και στην προσαρμογή
- Χρησιμοποιεί το npm ως διαχειριστή επιπρόσθετων πακέτων το οποίο περιέχει πάνω από 150 χιλιάδες πακέτα
- Ο κώδικας μπορεί να χρησιμοποιηθεί και από την μεριά του διακομιστή αλλά και από την μεριά του πελάτη αφού είναι και τα δυο JavaScript.

Εκτός τον ρόλο του διακομιστή για την υλοποίηση, το NodeJS όπως αναφέρθηκε παραπάνω εμπεριέχει διάφορα πακέτα τα οποία μας βοηθάνε είτε με την δρομολόγηση των σελίδων είτε με την πιστοποίηση των χρηστών. Στην πρώτη περίπτωση χρησιμοποιήθηκε το πακέτο ExpressJS. Η βιβλιοθήκη αυτή χειρίζεται με ιδιαίτερη ευκολία την δρομολόγηση των ιστοσελίδων και την διαχείριση των αιτήσεων από τον πελάτη.

```
// app/routes.js
module.exports = function(app, passport) {

    // =====================================
    // HOME PAGE (with login links) ========
    // =====================================
    app.locals.user = 'empty';

    app.get('/', function(req, res) {
        if(typeof req.user != 'undefined'){
        res.cookie('userOnline', true);
        console.log('reqqqq '+req.user);
        res.render('index.ejs', {
            "user" : req.user // get the user out of session and pass to template
        });
    }else{
        console.log('undefined');
        res.cookie('userOnline', 'false');
        res.cookie('userType', 'guest');
        res.render('index.ejs', {"user": 'empty'}); // load the index.ejs file
    }

    });
    app.get('/create', function(req, res) {
        res.render('create.ejs'); // load the index.ejs file
    });
    app.get('/manage', function(req, res) {
        res.render('manage.ejs'); // load the index.ejs file
    });
    app.get('/select', function(req, res) {
        res.render('select-thesis.ejs'); // load the index.ejs file
    });
    app.get('/manage/:id', function(req, res, next) {
        var id = req.params.id;
        console.log(id);
        res.render('manage-thesis', { "id": id });
    });
```

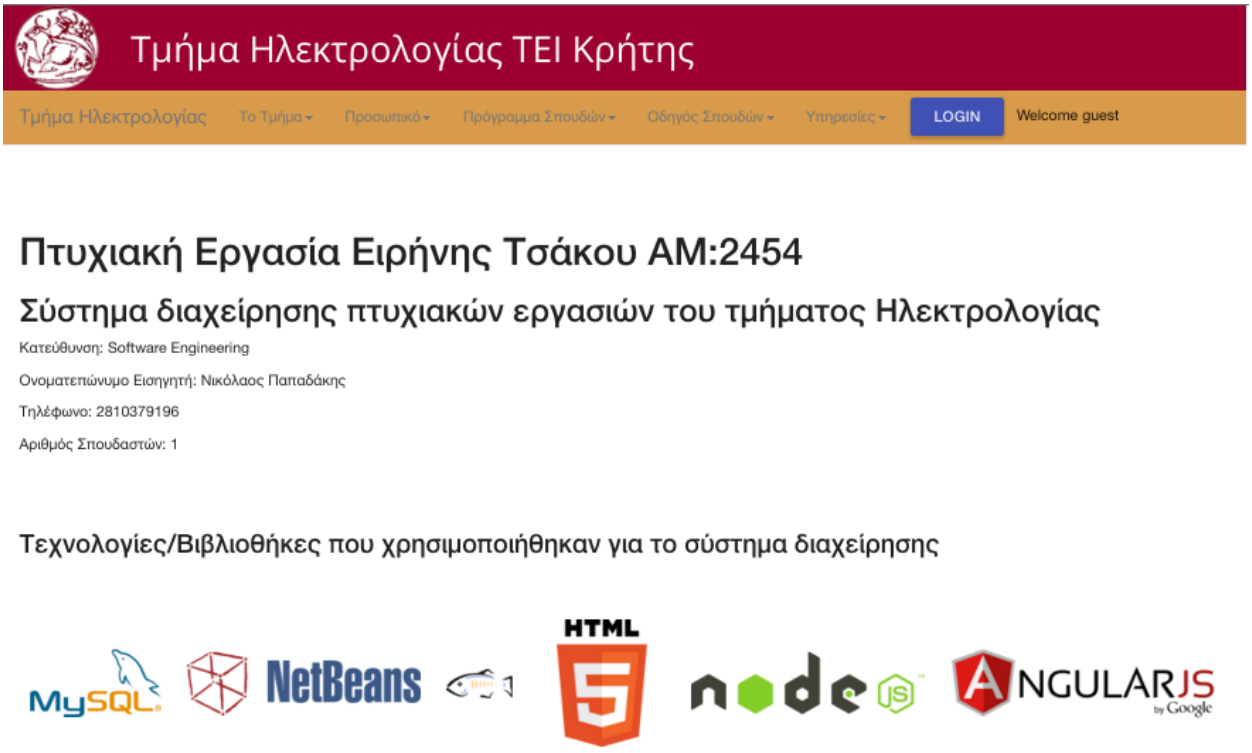
Εικόνα 31: Δρομολόγηση με ExpressJS

Για κάθε σελίδα που θέλουμε να δρομολογήσουμε δημιουργούμε μια μέθοδο app.get και παίρνει παράμετρο την υπο-διεύθυνση που θέλουμε να δρομολογήσουμε. Στο σώμα της μεθόδου ορίζουμε ποια σελίδα θα μεταμορφώσει και θα προσφέρει στον πελάτη.

4 Παρουσίαση Εφαρμογής & Σενάρια Χρήσης

4.1 Εισαγωγή

Για την αλληλεπίδραση του χρήστη με την εφαρμογή έγιναν προσπάθειες να γίνει εύχρηστο και όλα τα δεδομένα να είναι ευδιάκριτα. Στο πάνω μέρος της οθόνης υπάρχει η μπάρα περιήγησης η οποία συμπίπτει με την περιήγηση της ιστοσελίδας του ΤΕΙ Κρήτης και των τμημάτων του ιδρύματος. Όταν εισέρχεται ο χρήστης στην εφαρμογή μπορεί να επιλέξει να πραγματοποιήσει σύνδεση ή να περιηγηθεί στις υποδιευθύνσεις του τμήματος. Για να υπάρχει πρόσβαση στις διάφορες υπηρεσίες θα πρέπει αρχικά να γίνει σύνδεση, να πιστοποιηθεί ο χρήστης και ανάλογα τον τύπο να εμφανίσει τις διαθέσιμες λειτουργίες.

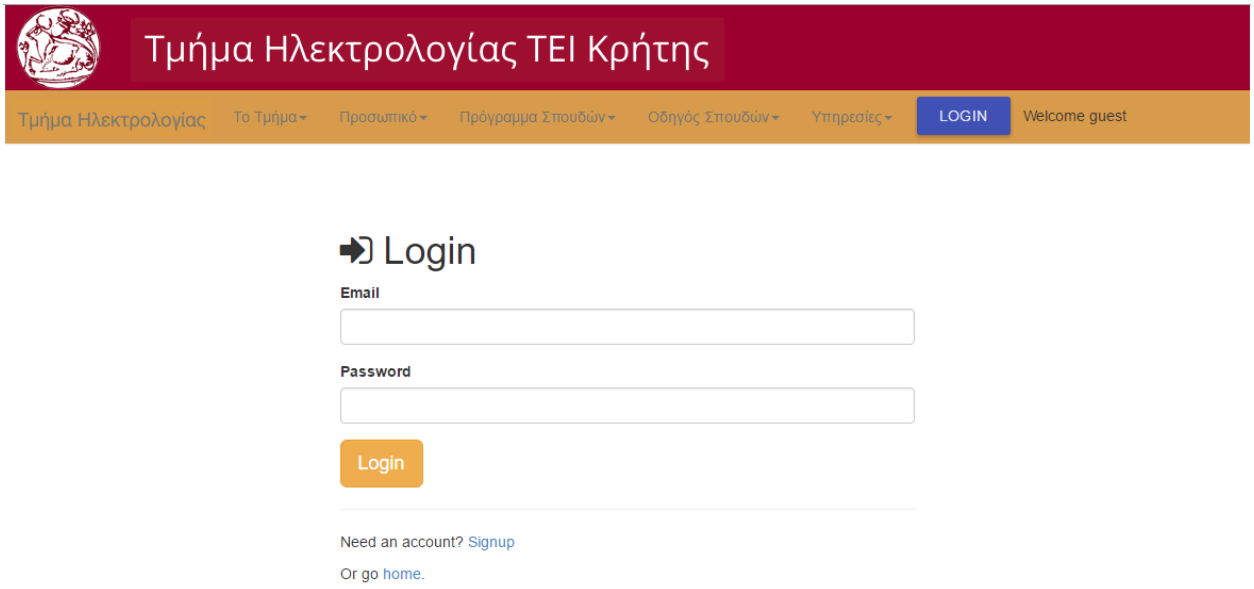


Εικόνα 32: Αρχική σελίδα εφαρμογής

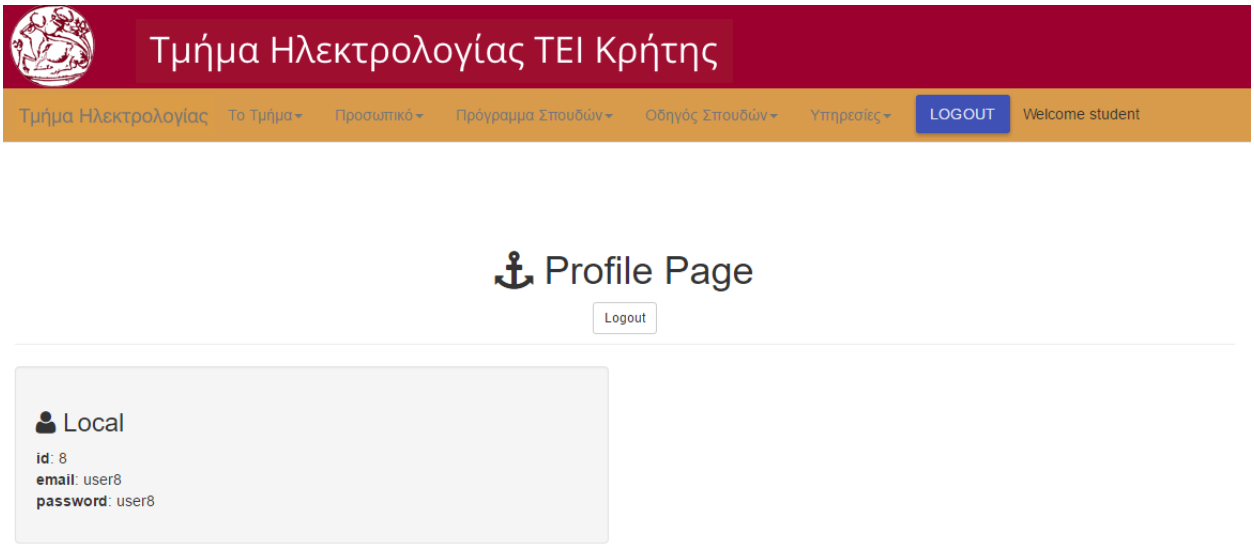
Στο κάτω μέρος της οθόνης παρουσιάζονται για πληροφοριακούς λόγους στον επισκέπτη οι τεχνολογίες που χρησιμοποιήθηκαν και αν ενδιαφέρεται για περισσότερες πληροφορίες μπορεί να πατήσει πάνω σε κάθε σύνδεσμο.

4.2 Σύνδεση/ Αποσύνδεση χρήστη

Για να γίνει σύνδεση και πιστοποίηση του χρήστη πατιέται το κουμπί "Login" στην μπάρα περιήγησης. Ο χρήστης καλείται να συμπληρώσει τα στοιχεία του ή να δημιουργήσει ένα νέο λογαριασμό και να γίνει ταυτοποίηση με το ΑΜ του. Αν τα στοιχεία είναι σωστά εισέρχεται στο σύστημα και ο λογαριασμός του αποθηκεύεται στα cookies του περιηγητή μέχρι να πραγματοποιήσει logout. Αν τα στοιχεία δεν είναι σωστά τότε εμφανίζεται μήνυμα ανάλογα το λάθος, π.χ. λάθος στοιχεία ή δεν υπάρχει χρήστης.



Εικόνα 33: Σύνδεση χρήστη

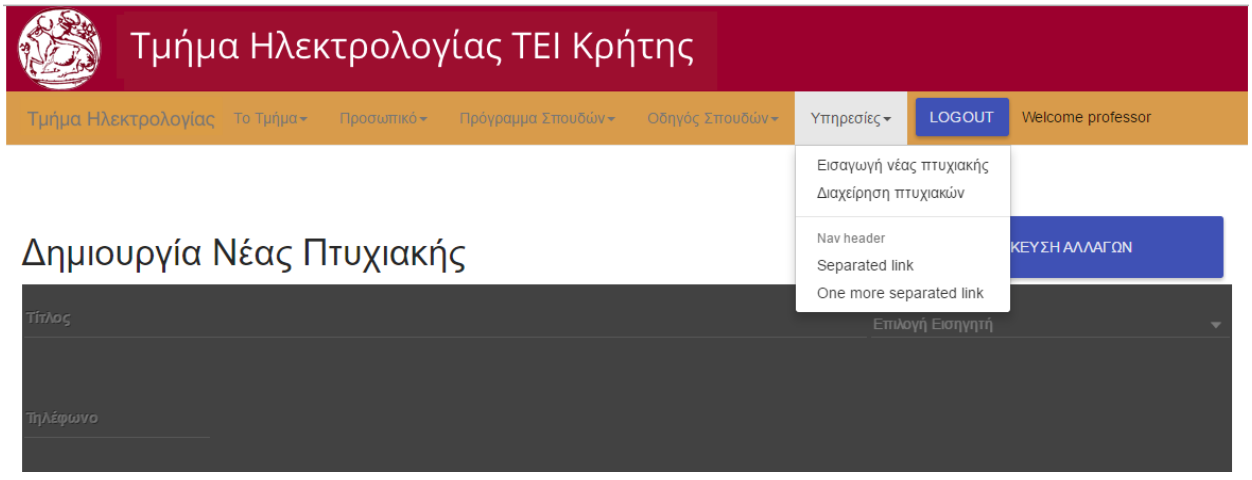


Εικόνα 34: Προφίλ χρήστη

Μετά την σύνδεση ο χρήστης καλωσορίζεται με την οθόνη του προφίλ του όπου βλέπει πληροφορίες για τα στοιχεία του όπως όνομα, ΑΜ αν υπάρχει κλπ. Σε μελλοντικές εκδόσεις θα μπορούσαν να προστεθούν επιλογές για την διαχείριση του λογαριασμού του.

4.3 Δημιουργία πτυχιακής

Αν ο χρήστης που εισέρχεται στην εφαρμογή είναι καθηγητής του δίνεται η επιλογή στις υπηρεσίες αν θέλει να εισάγει νέα πτυχιακή ή να επεξεργαστεί μια υπάρχουσα.



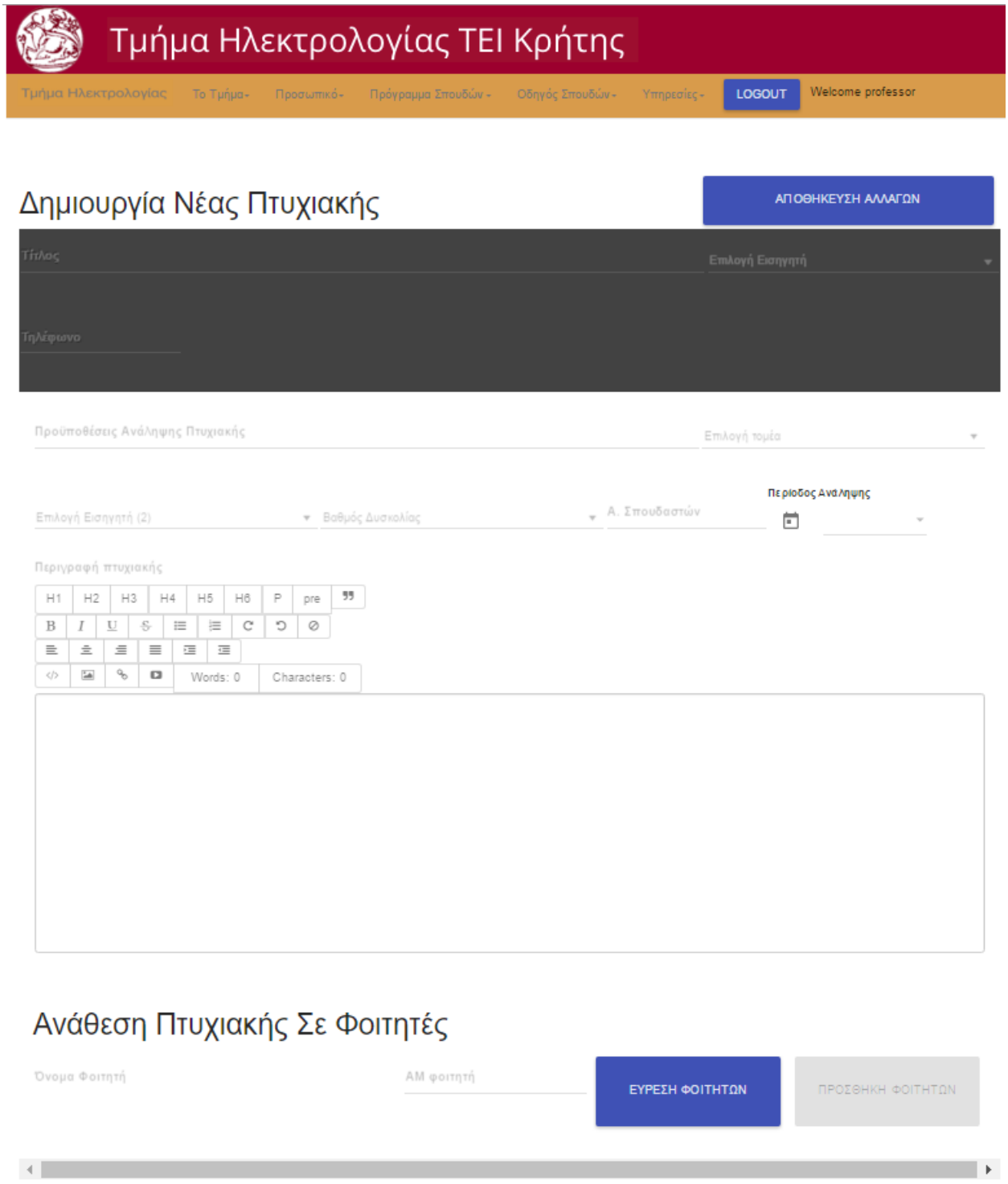
Εικόνα 35: Διαθέσιμες επιλογές καθηγητή

Στην περίπτωση εισαγωγής νέας πτυχιακής εμφανίζονται στον καθηγητή όλα τα απαραίτητα πεδία που απαιτούνται από την μηχανογράφηση πτυχιακών. Για να είναι πιο εύχρηστο στο πεδία που είναι πολλαπλής επιλογής εμπλουτίζονται με εγγραφές από την βάση δεδομένων. Για παράδειγμα, αν επαυξήσουμε το πεδίο εισηγητής θα εμπλουτιστεί με τις εγγραφές όλων των καθηγητών του τμήματος. Έτσι δεν χρειάζονται να συμπληρωθούν τα επιπρόσθετα πεδία που αφορούν τον καθηγητή όπως τηλέφωνο.

Για την περιγραφή της πτυχιακής υπάρχει πεδίο εισαγωγής το οποίο δημιουργεί HTML5 μορφοποίηση. Έτσι αν υπάρχει περίπτωση να δημοσιευθούν οι διαθέσιμες πτυχιακές σε ιστοσελίδα μπορούμε να χρησιμοποιήσουμε την HTML5.

Ο καθηγητής έχει την επιλογή να αναθέσει απευθείας την πτυχιακή που δημιουργεί σε φοιτητές. Στο πεδίο Ανάθεση Πτυχιακής σε Φοιτητές συμπληρώνει μονάχα το ΑΜ του και αν υπάρχει εγγραφή στην βάση συμπληρώνονται και τα υπόλοιπα απαραίτητα στοιχεία του φοιτητή όπως Ονοματεπώνυμο και προστίθεται στην λίστα με τους φοιτητές προς ανάθεση.

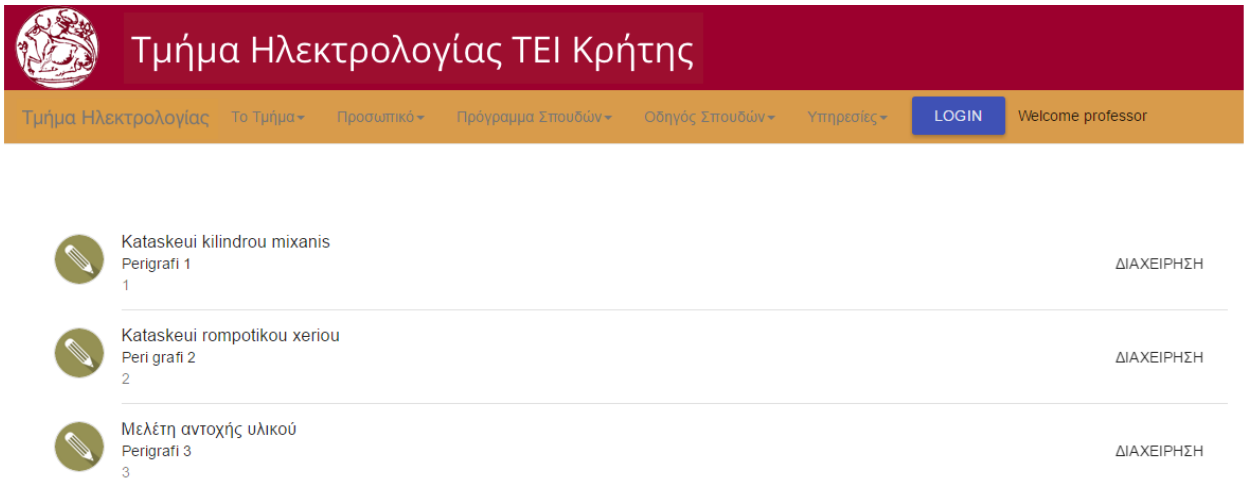
Όταν είναι έτοιμα όλα τα απαραίτητα πεδία, ο καθηγητής επιλέγει την αποθήκευση αλλαγών και πραγματοποιείται έλεγχος ακεραιότητας των δεδομένων. Αν τα δεδομένα είναι σωστά τότε προωθείται στην διαδικτυακή υπηρεσία και αποθηκεύεται στην βάση.



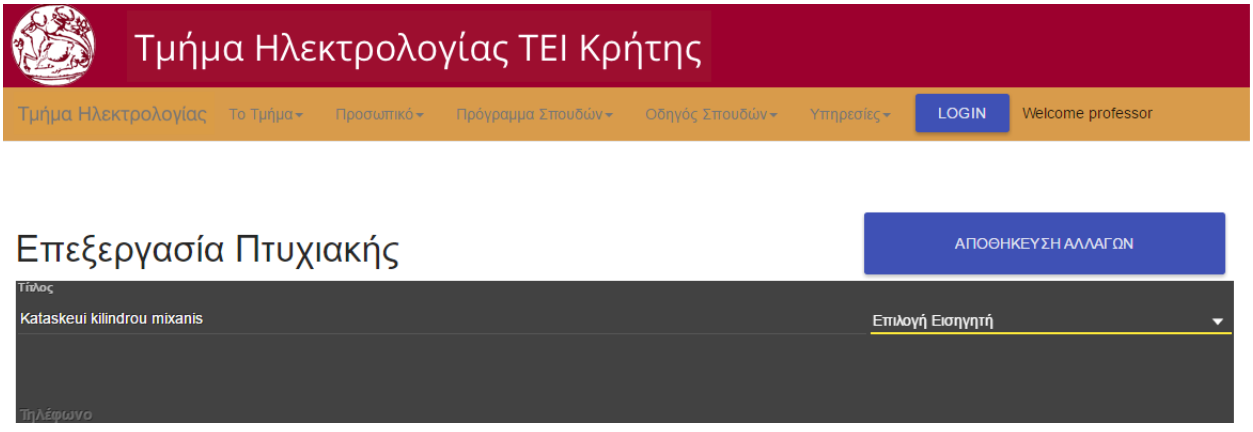
Εικόνα 36: Δημιουργία νέας πτυχιακής

4.4 Διαχείριση πτυχιακών

Στην περίπτωση επιλογής διαχείρισης πτυχιακών φορτώνονται όλες οι διαθέσιμες πτυχιακές που δεν έχουν πραγματοποιηθεί από την διαδικτυακή υπηρεσία. Εμφανίζονται στον καθηγητή ο τίτλος της πτυχιακής, μια συνοπτική περιγραφή και ο κωδικός της πτυχιακής. Επιλέγοντας διαχείριση, ο χρήστης μεταβιβάζεται σε μια νέα διεπαφή παρόμοια με αυτήν της δημιουργίας πτυχιακής. Η μόνη διαφορά είναι ότι πλέον τα πεδία είναι συμπληρωμένα και ο καθηγητής μπορεί να τα επεξεργαστεί.



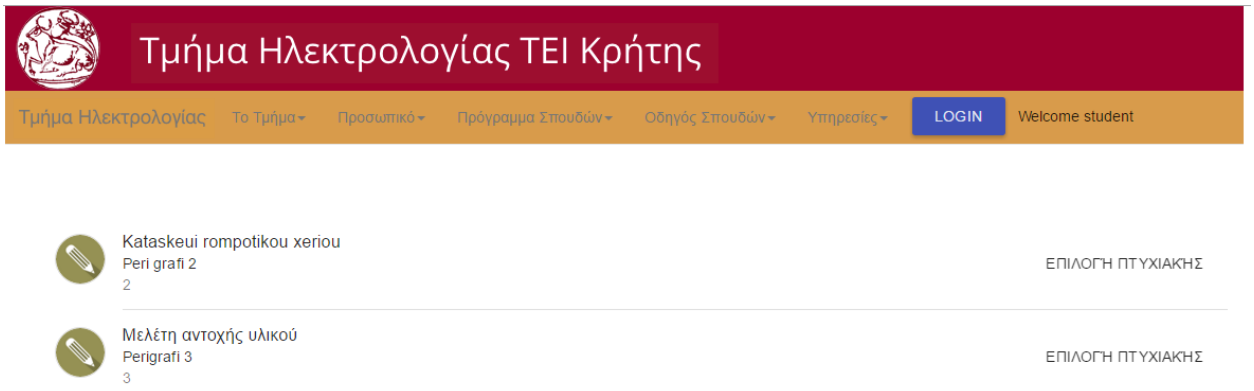
Εικόνα 37: Διαχείριση πτυχιακών



Εικόνα 38: Επεξεργασία υπαρχουσας πτυχιακής

4.5 Επιλογή πτυχιακής

Στην περίπτωση που εισέρχεται στην εφαρμογή φοιτητής του δίνεται η δυνατότητα να επιλέξει κάποια από τις υπάρχουσες πτυχιακές που δεν έχουν ανατεθεί σε αριθμό περισσότερο από αυτόν που ορίζει το πεδίο "Αριθμός Σπουδαστών" της εκάστης πτυχιακής. Του παρουσιάζονται συνοπτικές πληροφορίες για την κάθε πτυχιακή αλλά μπορεί και να επιλέξει πάνω σε μια και να δει περισσότερες πληροφορίες. Στην περίπτωση που επιθυμεί να την επιλέξει πατάει το κουμπί Επιλογή Πτυχιακής και εμφανίζεται ένας διάλογος για να επαληθεύσει την επιλογή του. Στην συνέχεια εγγράφεται με την διαδικτυακή υπηρεσία ότι ο φοιτητής έχει επιλέξει την πτυχιακή.



Εικόνα 39: Επιλογή πτυχιακής

5 Επίλογος – Μελλοντικές Επεκτάσεις

Στην παρούσα εργασία μελετήθηκαν οι περισσότερες τεχνολογίες για την ανάπτυξη διαδικτυακών εφαρμογών. Επιλέχθηκαν με γνώμονα κυρίως την δημοτικότητα αλλά και την ευκολία στην γρήγορη ανάπτυξη λογισμικού που απαιτείται στις μέρες μας. Η κυριότητα των σύγχρονων προγραμματιστών χρησιμοποιεί τις ίδιες τεχνολογίες που υιοθετήθηκαν σε αυτήν την εργασία όχι μόνο της απλότητας που προσφέρουν αλλά λόγω της εξελιξιμότητας αυτών.

Λόγω της αρχιτεκτονικής της εφαρμογής που υλοποιήθηκε είναι εύκολη η επεκτασιμότητα της εφαρμογής. Μερικά χαρακτηριστικά που θα μπορούσαν να προστεθούν είναι η σύνδεση με τον διακομιστή LDAP του ιδρύματος ώστε να μην υπάρχει ξεχωριστή βάση για τους φοιτητές και να παίρνονται όλα τα απαραίτητα δεδομένα από εκεί. Άλλη μια επέκταση θα ήταν να αποθηκεύονται αυτούσια οι πτυχιακές στην βάση όταν αυτές πραγματοποιούνται.

6 Βιβλιογραφία

1. Wikipedia. Code Examples on Ajax, CGI, Java, Javascript, Actionscript, Asp, Php, C++ and .Net. www.wikipedia.com.

2. W3 Schools. On-line Tutorials and Examples on XML, Javascript, PHP, Asp. http://www.w3schools.com/.

3. Rising Stack Blog. 2016. The Beginner's Guide to Terminal for Node.js Development. [ONLINE] Available at: https://blog.risingstack.com/terminal-guide-for-nodejs/. [Accessed 23 March 16].

4. NodeJS Docs. 2016. Docs | NodeJS. [ONLINE] Available at: https://nodejs.org/en/docs/. [Accessed 23 March 16].

5. AngularJS Docs. 2016. AngularJS Tutorials. [ONLINE] Available at: https://docs.angularjs.org/tutorial. [Accessed 20 March 16].

6. MySQL Tutorial. 2016. TutorialsPoint | MySQL Tutorials. [ONLINE] Available at: http://www.tutorialspoint.com/mysql/. [Accessed 20 March 16].

7. ExpressJS Routing. 2016. Basic Routing . [ONLINE] Available at: http://expressjs.com/en/starter/basic-routing.html. [Accessed 17 March 16].

8. Elated.com. 2016. MySQL for Beginners. [ONLINE] Available at: http://www.elated.com/articles/mysql-for-absolute-beginners/. [Accessed 15 March 16].

9. Wikipedia.org. 2016. Representational State Transfer. [ONLINE] Available at: https://en.wikipedia.org/wiki/Representational_state_transfer. [Accessed 21 March 16].

10. Restapitutorial.com. 2016. Rest API Tutorial. [ONLINE] Available at: http://www.restapitutorial.com/lessons/whatisrest.html. [Accessed 24 February 16].

11. W3Schools. 2016. AngularJS Tutorials. [ONLINE] Available at: http://www.w3schools.com/angular/. [Accessed 10 February 16].

12. Crunchify.com. 2016. Restful API with Jersey. [ONLINE] Available at: http://crunchify.com/how-to-build-restful-service-with-java-using-jax-rs-and-jersey/. [Accessed 24 February 16].

13. Code Academy. 2016. Learn AngularJS. [ONLINE] Available at: https://www.codecademy.com/learn/learn-angularjs. [Accessed 24 February 16].

14. Envato Tuts. 2016. Build Applications with PassportJS. [ONLINE] Available at: http://code.tutsplus.com/tutorials/authenticating-nodejs-applications-with-passport--cms-21619. [Accessed 10 March 16].

15. EggHead. 2016. Getting Started with ExpressJS. [ONLINE] Available at: https://egghead.io/series/getting-started-with-express-js. [Accessed 10 March 16].