ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΜΣ ΔΙΔΑΚΤΙΚΗ ΤΗΣ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ
ΚΑΤΕΥΘΥΝΣΗ: ΔΙΚΤΥΟΚΕΝΤΡΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

"Διαδικτυακό πληροφοριακό σύστημα αυτοματοποίησης χρονοπρογραμματισμού πόρων "

Επιβλέπων καθηγητής: Κυριαζής Δημοσθένης

Φοιτήτρια: Μαρίνη Ελένη, Α.Μ.: ΜΕ13047

Φεβρουάριος 2016

UNIVERSITY OF PIRAEUS

DEPARTMENT OF DIGITAL SYSTEMS

POSTGRADUATE PROGRAMME "DIGITAL SYSTEMS & SERVICES"
NETWORK-ORIENTED INFORMATION SYSTEMS

DISSERTATION

"Online automation information system of

scheduling resources"

Supervisor: Kyriazis Dimosthenis

Student: Marini Eleni, ME13047

February 2016

Ευχαριστίες

Για την ολοκλήρωση της παρούσας διπλωματικής εργασίας, συνετέλεσε, εκτός από την ατομική προσπάθεια, η στήριξη ορισμένων ανθρώπων τους οποίους θα ήθελα να ευχαριστήσω.

Πρώτα θέλω να ευχαριστήσω τον επιβλέποντα καθηγητή της διπλωματικής εργασίας, τον Επίκουρο Καθηγητή Δημοσθένη Κυριαζή, για την πολύτιμη καθοδήγηση του και την εμπιστοσύνη που μου έδειξε καθώς και για την υπομονή του.

Ακόμα, ευχαριστώ τους γονείς μου Δημήτρη και Δήμητρα, καθώς και τον αδερφό μου Βασίλη, που μου πρόσφεραν την απαραίτητη ηθική συμπαράσταση για την ολοκλήρωση της παρούσας εργασίας

Τέλος, ιδιαίτερες ευχαριστίες θέλω να απευθύνω στον συνάδελφο και φίλο Χάρο Αλέξανδρο για την βοήθεια που μου προσέφερε.

Περιεχόμενα

1 Εισαγωγή

1.1 Αντικείμενο της εργασίας

Η παρούσα διπλωματική εργασία έχει ως αντικείμενο εξέτασης τον Χρονοπρογραμματισμό Πόρων Βασισμένο σε Περιορισμούς (Constraint Based Scheduling). Αποτελεί μία προσπάθεια αυτοματοποίησης του χρονοπρογραμματισμού των υποψήφιων διδακτόρων του Πανεπιστημίου Πειραιώς που απαιτούνται για την επίβλεψη των εξεταζόμενων μαθημάτων κατά της διάρκεια της εξεταστικής περιόδου, διαδικασία που μέχρι σήμερα πραγματοποιείται χειρωνακτικά από κάποιο μέλος του εκπαιδευτικού προσωπικού του τμήματος.

Αναλυτικότερα, η διαδικασία που καλείται να αυτοματοποιήσει η διαδικτυακή κυρίως εφαρμογή, είναι η επιλογή των κατάλληλων υποψήφιων διδακτόρων για την επιτήρηση των μαθημάτων της εξεταστικής περιόδου, δεδομένων κάποιων προσωπικών χρονικών περιορισμών που μπορεί να θέσει κάθε υποψήφιος διδάκτορας, αλλά και ενός συγκεκριμένου αριθμού επιτηρητών που απαιτούνται για κάθε μάθημα.

Για το σκοπό αυτό σχεδιάστηκε και αναπτύχθηκε ένα πλήρως αυτοματοποιημένο πληροφοριακό σύστημα (web application) καθώς και μία εφαρμογή για κινητές συσκευές (mobile application) , η οποία διατίθεται τόσο για λειτουργικό Android όσο και για λειτουργικό iOS. Η διαδικτυακή εφαρμογή δημιουργεί το πρόγραμμα των επιτηρήσεων, ενώ μέσω της mobile εφαρμογής παρέχεται η δυνατότητα τόσο της παρακολούθησης του προγράμματος από τους ενδιαφερόμενους (καθηγητές και υποψήφιους διδάκτορες) όσο και της δήλωσης της παρουσίας των υποψηφίων διδακτόρων στις επιτηρήσεις που τους έχουν ανατεθεί.

1.2 Θεωρητικό υπόβαθρο

Συχνά παρουσιάζονται καταστάσεις οι οποίες θα πρέπει να αντιμετωπιστούν και οι οποίες μπορούν να αναγνωριστούν ως προβλήματα συνδυαστικής βελτιστοποίησης. Η επίλυση ενός προβλήματος συνδυαστικής βελτιστοποίησης

συνίσταται στην επιλογή κατάλληλων τιμών για μεγέθη που μπορούν να λάβουν διακριτές τιμές με στόχο την επίτευξη του καλύτερου δυνατού αποτελέσματος ενώ ταυτόχρονα θα πρέπει να ικανοποιείται ένα σύνολο από περιορισμούς[12].

Ιδιαίτερη κατηγορία προβλημάτων συνδυαστικής βελτιστοποίησης αποτελούν τα προβλήματα χρονοπρογραμματισμού. Ως πρόβλημα χρονοπρογραμματισμού μπορεί να οριστεί ένα πρόβλημα, στο οποίο ζητείται η ανάθεση δραστηριοτήτων στο χρόνο και στο διαθέσιμο προσωπικό (ανθρώπους ή μηχανήματα). Σε κάθε περίπτωση, η οποιαδήποτε ανάθεση πόρων στο χρόνο θα πρέπει να ικανοποιεί ένα σύνολο περιορισμών αλλά ταυτόχρονα να επιδιώκει την εύρεση της βέλτιστης λύσης [12]. Μη αυστηρά θα μπορούσαμε να ορίσουμε ως χρονοπρογραμματισμό τη διαδικασία ανάθεσης πόρων (resources) σε εργασίες (jobs), για ένα ορισμένο χρονικό διάστημα, δεδομένων κάποιων περιορισμών και ενός κριτηρίου - στόχου. Ένα πρόβλημα χρονοπρογραμματισμού μπορεί να οριστεί ως ένα πρόβλημα στο οποίο εμπλέκονται οι τέσσερις ακόλουθοι παράμετροι: πεπερασμένο σύνολο χρονικών περιόδων, πόρων, εργασιών και περιορισμών. Λαμβάνοντας υπόψη τις παραμέτρους αυτές, ένα πρόβλημα χρονοπρογραμματισμού ορίζεται ως η ανάθεση χρονικών περιόδων και πόρων σε εργασίες, έτσι ώστε οι δεδομένοι περιορισμοί να ικανοποιούνται στον μεγαλύτερο δυνατό βαθμό.

Υπάρχουν δεκάδες παραδείγματα προβλημάτων χρονοπρογραμματισμού (χρονοπρογραμματισμός εκπαιδευτικών ιδρυμάτων, χρονοπρογραμματισμός ανθρώπινου δυναμικού, χρονοπρογραμματισμός μέσων μεταφοράς κλπ).

1.3 Διάθρωση της εργασίας,

Η παρούσα διπλωματική εργασία αποτελείται από οκτώ κεφάλαια. Το Κεφάλαιο 1 αποτελεί μια εισαγωγή στο θέμα που πραγματεύεται η διπλωματική. Στο Κεφάλαιο 2 παρουσιάζεται και αναλύεται η αρχιτεκτονική της προτεινόμενης λύσης στο πρόβλημα του χρονοπρογραμματισμού των υποψήφιων διδακτόρων που απαιτούνται για τις επιτηρήσεις των μαθημάτων κατά τις εξεταστικές περιόδους. Οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση τόσο της web όσο και της

mobile εφαρμογής αναφέρονται και παρουσιάζονται αναλυτικά στο τρίτο Κεφάλαιο. Στο Κεφάλαιο 4 περιγράφεται αναλυτικά η λειτουργικότητα των εφαρμογών που αναπτύχθηκαν ενώ, στο Κεφάλαιο 5 παρουσιάζεται η ροή τόσο των δεδομένων όσο και των διαδικασιών των δυο εφαρμογών. Η περιγραφή της υλοποίησης των εφαρμογών δίνεται στο Κεφάλαιο 6 και στο κεφάλαιο 7 παρουσιάζονται και αναλύονται οι οθόνες που παρήχθησαν τόσο για την διαδικτυακή όσο και για την mobile εφαρμογή. Τέλος, το κεφάλαιο 8 αναλύονται τα συμπεράσματα της διπλωματικής εργασίας.

2 Αρχιτεκτονική προτεινόμενης λύσης

Για την αποθήκευση των δεδομένων τόσο της web εφαρμογής όσο και της mobile εφαρμογής χρησιμοποιήθηκε βάση τεχνολογίας MongoDB η οποία φιλοξενείται στο Cloud και συγκεκριμένα στο Modulus [30]. Για την ανάπτυξη της web εφαρμογής επιλέχθηκε αρχιτεκτονική τύπου MVC (Model-View-Controller). Η web εφαρμογή φιλοξενείται στο Cloud και συγκεκριμένα στο Heroku [31]. Έτσι η mobile εφαρμογή έχει τη δυνατότητα να μιλάει με τα rest services της web εφαρμογής ώστε να έχει πρόσβαση στα δεδομένα. Στην Εικόνα 2 παρουσιάζεται συνοπτικά η αρχιτεκτονική κάτω από την οποία αναπτύχθηκαν οι δυο εφαρμογές.

2.1 Η αρχιτεκτονική MVC

Το MVC είναι ένα μοντέλο αρχιτεκτονικής λογισμικού το οποίο χρησιμοποιείται για την δημιουργία περιβαλλόντων αλληλεπίδρασης χρήστη. Πρωτοεμφανίστηκε στην γλώσσα προγραμματισμού smalltalk [32]. Η smalltalk είναι μία αντικειμενοστραφής γλώσσα προγραμματισμού η οποία έχει την δυνατότητα κατά τη διάρκεια που τρέχουμε ένα πρόγραμμα να αλλάζει τη δομή και την συμπεριφορά των δεδομένων του καθώς και διαφόρων συναρτήσεων αυτού.

Στο μοντέλο αυτό η εφαρμογή διαιρείται σε τρία μέρη ώστε να μπορέσει να ανεξαρτητοποιηθεί η παρουσίαση της πληροφορίας στον χρήση από την μορφή με

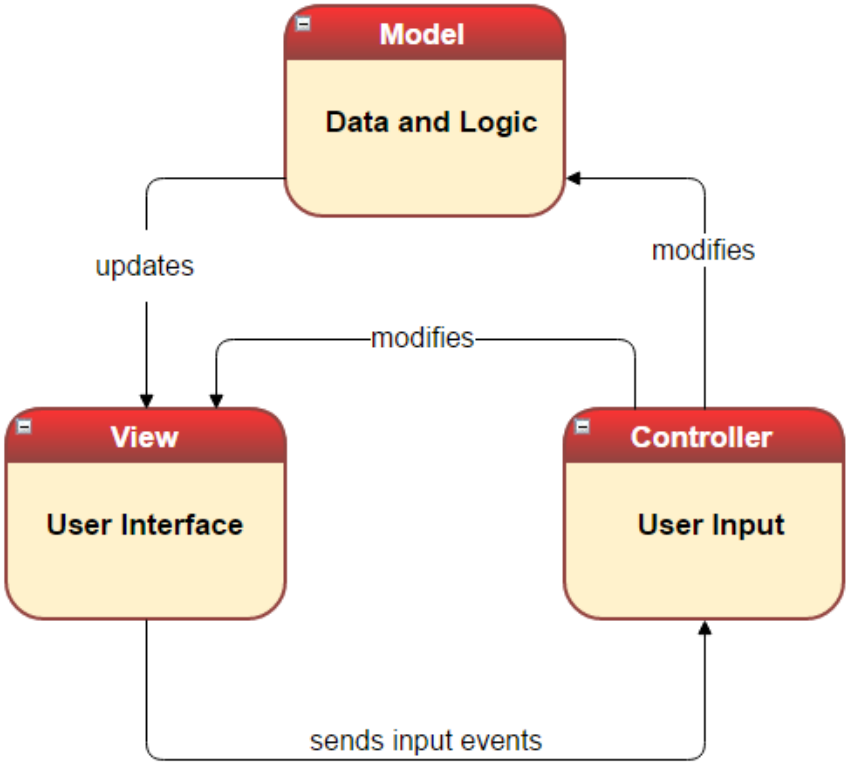
την οποία έχει αποθηκευτεί. Το κύριο μέρος του μοντέλου είναι το αντικείμενο *Model* το οποίο διαχειρίζεται την ανάκτηση και την αποθήκευση των δεδομένων στο σύστημα. Το αντικείμενο *View* χρησιμοποιείται μόνο για την παρουσίαση της πληροφορίας στον χρήστη. Το τρίτο μέρος είναι ο *Controller* ο οποίος δέχεται την είσοδο από τον χρήστη και στέλνει εντολές στο αντικείμενο Model και στο View. Στην MVC αρχιτεκτονική ορίζονται οι ακόλουθες αλληλεπιδράσεις μεταξύ των τριών τμημάτων που την αποτελούν:

- Ο ***Controller*** στέλνει εντολές στο model και να ενημερώνει την κατάστασή του. Επιπλέον στέλνει εντολές ώστε να γίνει η αντίστοιχη αναπαράσταση των δεδομένων του μοντέλου μέσω του view. Αποτελεί στην ουσία τον διαμεσολαβητή μεταξύ Model και View. Επικοινωνεί με το Model, παίρνει τα δεδομένα που ζητά το View και αφού τα επεξεργαστεί τα στέλνει πίσω στο View για απεικόνιση.
- Το ***Model*** ενημερώνει τα αντίστοιχα views και τους controllers όταν υπάρχει αλλαγή στα δεδομένα. Αυτή η ενημέρωση επιτρέπει στα views να ενημερώνουν την γραφική απεικόνιση. Στο model αναπτύσσουμε όλες τις λειτουργίες της εφαρμογής οι οποίες σχετίζονται με την πρόσβαση στη βάση δεδομένων. Οι λειτουργίες αυτές είναι με τη μορφή μεθόδων με τις οποίες εκτελούμε διάφορες λειτουργίες διαχείρισης των δεδομένων που λαμβάνουμε από τη βάση. Για παράδειγμα αν θέλουμε σε μία σελίδα να εμφανίσουμε όλους καθηγητές από τη βάση δεδομένων ,θα πρέπει στο model των καθηγητών να υπάρχει κάποια μέθοδος, έστω «getAllProfessors()», η οποία περιέχει κώδικα που μιλάει με τη βάση και μας επιστρέφει τα δεδομένα που ζητήσαμε.
- Το ***View*** αναπαριστά με γραφικό τρόπο την πληροφορία που περιέχει το model δημιουργώντας γραφική παρουσίαση στο χρήστη. Μέσα στο view υπάρχει το HTML της σελίδας της εφαρμογής μας. Είναι αυτό που βλέπουμε. Τις περισσότερες φορές ένα View μιλάει με έναν controller και αφού ο controller κάνει τις διάφορες επεξεργασίες των δεδομένων στέλνει στη View συγκεκριμένα δεδομένα να εμφανίσει.

Οι περισσότερες εφαρμογές και συστήματα ανακτούν δεδομένα από κάποια πηγή (βάση δεδομένων) και τα εμφανίζουν στο χρήστη. Όταν ο χρήστης τροποποιεί

διαγράφει ή εισάγει δεδομένα στην εφαρμογή, αυτή ενημερώνει την βάση δεδομένων με τα ανανεωμένα δεδομένα. Όπως βλέπουμε η βασική επικοινωνία και ανταλλαγή πληροφορίας φαίνεται να πραγματοποιείται μεταξύ βάσης δεδομένων και διεπαφής χρήστη (User Interface-UI). Για το λόγο θα φάνταζε ιδανικό και λογικό η σύνδεση των δυο αυτών μερών να είναι άμεση ώστε να αυξηθεί η απόδοση της εφαρμογής αλλά και να μειωθεί αρκετά η έκταση και το μέγεθος του κώδικα για την υλοποίηση μιας εφαρμογής.

Ωστόσο η προσέγγιση αυτή της άμεσης σύνδεσης της διεπαφής του χρήστη με τη βάση δεδομένων, παρουσιάζει αρκετά προβλήματα. Ένα από αυτά και ίσως το βασικότερο είναι ,ότι επειδή η λογική της εφαρμογής (business logic) είναι ενσωματωμένη με την διεπαφή του χρήστη ,που έχει την τάση να αλλάζει συχνότερα, θα πρέπει κάθε φορά που παρουσιάζεται η ανάγκη για προσθήκη νέων οθονών ή έστω για τροποποίηση των στυλ απεικόνισης της εφαρμογής να πραγματοποιούνται αλλαγές και στο τμήμα που περιέχει όλη τη λογική, γεγονός που καθιστά πιθανή την περίπτωση εισαγωγής λαθών στην εφαρμογή επομένως και αναγκαίο τον επανέλεγχο του τμήματος της λογικής.



Εικόνα 1: Η MVC Αρχιτεκτονική

Η αρχιτεκτονική Model-View-Controller δίνει λύση στο πρόβλημα που αναφέρθηκε παραπάνω διαχωρίζοντας το τμήμα της απεικόνισης των δεδομένων, τις λειτουργίες της εφαρμογής που διαχειρίζονται την εισαγωγή δεδομένων από τον χρήστη και την αποθήκευση των δεδομένων σε τρία διακριτά μέρη/αντικείμενα.

Συγκεκριμένα το μέρος του View διαχειρίζεται την απεικόνιση των δεδομένων ενώ το αντικείμενο Model διατηρεί τη δομή των δεδομένων, δέχεται αιτήματα για την ενημέρωσή τους από τον Controller αλλά και για την ανάκτησή τους από το View [3]. Το αντικείμενο Controller μεταφράζει την εισόδου που λαμβάνει από το χρήστη, η οποία μπορεί να προέρχεται από πληκτρολόγηση η κλικ του ποντικιού, και ενημερώνει το αντικείμενο Model και το αντικείμενο View ώστε να πραγματοποιηθεί η κατάλληλη τροποποίηση.

Στο σημείο αυτό πρέπει να σημειωθεί ότι τα αντικείμενα View και Controller εξαρτώνται από το Model, ενώ το Model δεν εξαρτάται ούτε από το View ούτε από το Controller. Το γεγονός αυτό αποτελεί ένα από τα βασικά πλεονεκτήματα όταν πρέπει να απομονωθούν οι κύριες λειτουργίες μιας εφαρμογής. Ο διαχωρισμός αυτός επιτρέπει στο Model να αναπτυχθεί και να ελεγχθεί ανεξάρτητα από τα υπόλοιπα, χωρίς δηλαδή να είναι απαραίτητη η ανάπτυξη λειτουργίας που θα απεικονίζει τα δεδομένα (View). Στις εφαρμογές Παγκόσμιου Ιστού, ο διαχωρισμός ανάμεσα στο View (browser) και στον Controller (τα server-side τμήματα που χειρίζονται τα HTTP αιτήματα) είναι συνήθως πολύ καλά ορισμένος, με αποτέλεσμα η MVC αρχιτεκτονική να είναι πολύ δημοφιλής σε εφαρμογές που απαιτείται διαχωρισμός της διεπαφής χρήστη από τη λογική της εφαρμογής.

Για την αρχιτεκτονική MVC υπάρχουν δυο διαφοροποιήσεις (Burbeck, 1992) [32] :

- το passive model και
- το active model,

που μπορούν να επιλεχθούν ανάλογα με τις απαιτήσεις της εκάστοτε εφαρμογής. Στο passive model το τμήμα του Controller χειρίζεται το Model αποκλειστικά, δηλαδή ο Controller τροποποιεί το Model και έπειτα ενημερώνει το View ότι το Model έχει τροποποιηθεί, οπότε πρέπει ανανεωθεί (refresh) και το View (Εικόνα). Συνεπώς δεν υπάρχει τρόπος το Model να ενημερώσει το View για αλλαγές

που μπορεί να συμβούν στο ίδιο, καθώς δεν υπάρχει κάποιος απλός τρόπος ώστε ένας browser να λαμβάνει ασύγχρονες ενημερώσεις από το server. Ο browser δε μπορεί να ανιχνεύσει αλλαγές που πιθανόν συμβαίνουν στα δεδομένα στο server. Μόνο όταν ο χρήστης κάνει ένα αίτημα (request) για ανανέωση δεδομένων, διερευνάται αν ο server έχει υποστεί αλλαγές. Επομένως το Model σε αυτήν την εκδοχή είναι τελείως ανεξάρτητο από το View και το Controller.

Από την άλλη, στο active model της αρχιτεκτονικής MVC το Model δύναται να αλλάζει την κατάστασή του χωρίς την ανάμιξη του Controller. Αυτό συμβαίνει όταν άλλες πηγές προκαλούν αλλαγές στα δεδομένα και αυτές οι αλλαγές πρέπει να απεικονιστούν στο View. Καθώς μόνο το Model μπορεί να αντιληφθεί τις αλλαγές στα δεδομένα, θα πρέπει το ίδιο να ενημερώσει το View για να ανανεωθούν τα δεδομένα στην οθόνη του χρήστη. Ωστόσο, ένα από τα κύρια χαρακτηριστικά της MVC αρχιτεκτονικής είναι η ανεξαρτησία του Model από το View. Αν το Model έπρεπε να ειδοποιήσει το View για τις όποιες αλλαγές, τότε θα υπήρχε μεταξύ τους εξάρτηση, γεγονός που θέλουμε να αποφευχθεί. Στην περίπτωση αυτή, για να μπορέσουμε να διατηρήσουμε την ανεξαρτησία των τριών τμημάτων και κυρίως του Model από το View, χρησιμοποιούμε ένα επιπλέον τμήμα ή διεπαφή (interface), ο Observer. Στο τμήμα αυτό οποίο διατηρείται μια λίστα εξαρτώμενων αντικειμένων από ένα κύριο αντικείμενο, και κάθε φορά που υπάρχει κάποια αλλαγή στην κατάσταση του κύριου αντικειμένου, ο Observer ειδοποιεί τα περιεχόμενα στη λίστα αντικείμενα αυτόματα, καλώντας μια από τις μεθόδους τους. Με αυτόν τον τρόπο εξαλείφεται η ανάγκη της αλληλεπίδρασης του Model με το View. Κάθε View υλοποιεί ένα μηχανισμό Observer που αλληλεπιδρά με το Model και παρατηρεί αλλαγές. Όταν στο Model συμβεί οποιαδήποτε αλλαγή, τότε ενημερώνεται αποκλειστικά ο Observer και κατ' επέκταση και το View. Συνεπώς, δεν υπάρχει καμία επικοινωνία μεταξύ Model και View. Στις περιπτώσεις κατά τις οποίες ο Controller πρέπει να ενημερωθεί για αλλαγές που έχουν γίνει στο Model (για παράδειγμα, να ενεργοποιηθούν ή να απενεργοποιηθούν κάποιες επιλογές στο μενού της εφαρμογής), τότε ο Controller θα πρέπει να υλοποιήσει επίσης ένα τμήμα Observer. Στις περιπτώσεις όπου υπάρχουν περισσότερα από ένα View, μπορεί κάθε ένα από αυτά να παραμετροποιηθεί ώστε να δέχεται μόνο ορισμένου τύπου αλλαγές.

Ένα ακόμα χαρακτηριστικό της αρχιτεκτονικής MVC είναι ότι διευκολύνει σημαντικά τη δυνατότητα για έλεγχο (testing) της εφαρμογής. Κατά τον έλεγχο των τμημάτων μιας εφαρμογής παρουσιάζονται δυσκολίες όταν δεν υπάρχει ανεξαρτησία μεταξύ τους. Ειδικότερα όταν αυτή η διεπαφή χρήστη δεν είναι ανεξάρτητη παρουσιάζονται επιπλέον προβλήματα, καθώς απαιτείται ένας σύνθετος μηχανισμό ελέγχου, ακόμα και για τον έλεγχο μιας απλής συνάρτησης. Ακόμα χειρότερα, όταν εμφανίζεται ένα λάθος είναι δύσκολο να απομονωθεί το πρόβλημα σε ένα συγκεκριμένο τμήμα της εφαρμογής. Για το λόγο αυτό ο διαχωρισμός των τμημάτων/αντικειμένων μιας εφαρμογής είναι πολύ σημαντικός στο σχεδιασμό των αρχιτεκτονικών, και σε αυτό δίνει τη λύση η αρχιτεκτονική MVC διαχωρίζοντας τις έννοιες της αποθήκευσης, της εμφάνισης και της ενημέρωσης των δεδομένων σε τρία ξεχωριστά τμήματα τα οποία μπορούν να λειτουργούν και ελέγχονται ανεξάρτητα.

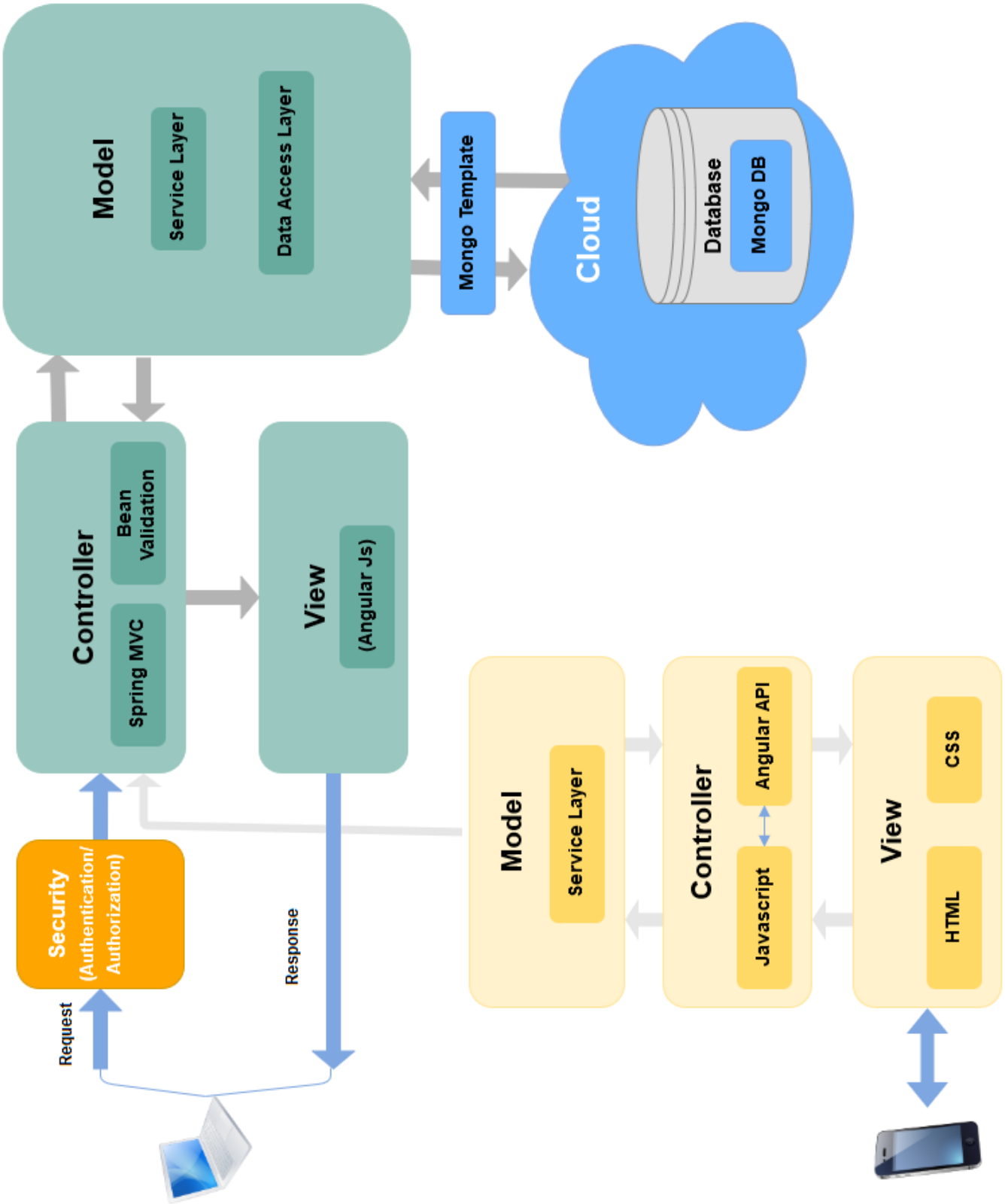
Πρέπει να σημειωθεί πως, εκτός από τα προβλήματα αλληλεξάρτησης ανάμεσα στα τμήματα μιας εφαρμογής, έρχεται και το πρόβλημα του ελέγχου της διεπαφής του χρήστη η οποία είναι δύσκολο από τη φύση της να ελεγχθεί. Το testing στις διεπαφές χρήστη απαιτεί χειροκίνητους ελέγχους, ή ειδικές μεθόδους που προσομοιώνουν τις κινήσεις των χρηστών, και απαιτούν πολύ χρόνο για να υλοποιηθούν και να σταθεροποιηθούν. Σε καμία περίπτωση, η χρήση της αρχιτεκτονικής MVC δεν εξαλείφει την ανάγκη για έλεγχο της διεπαφής χρήστη, ωστόσο ο διαχωρισμός του Model από το View , μας επιτρέπει να ελέγχουμε ανεξάρτητα τα δυο αυτά τμήματα γεγονός που μειώνει τον αριθμό των περιπτώσεων για τις οποίες πρέπει να ελεγχθεί η διεπαφή χρήστη.

Όπως προαναφέρθηκε, το View είναι πλήρως διαχωρισμένο από το Model και δεν υπάρχει άμεση εξάρτηση μεταξύ τους. Επομένως, εάν προσπαθούσαμε να συγκεντρώσουμε τα πλεονεκτήματα της αρχιτεκτονικής MVC θα έπρεπε ανάμεσα σε αυτά να αναφερθεί η δυνατότητα απεικόνισης πολλαπλών views που αφορούν τα ίδια δεδομένα την ίδια χρονική στιγμή. Για παράδειγμα, πολλαπλές σελίδες σε μια εφαρμογή Παγκόσμιου Ιστού μπορεί να χρησιμοποιούν τα ίδια Model. Ένα ακόμα παράδειγμα που θα μπορούσαμε να αναφέρουμε είναι, μια εφαρμογή που επιτρέπει στο χρήστη να αλλάζει την εμφάνιση μιας σελίδας. Αυτές οι σελίδες απεικονίζουν τα ίδια δεδομένα μέσω ενός κοινού μοντέλου, αλλά το εμφανίζουν στην οθόνη με διαφορετικό τρόπο.

Οι απαιτήσεις μιας διεπαφής χρήστη έχουν την τάση να αλλάζουν πιο συχνά από ότι η λογική της εφαρμογής καθώς για παράδειγμα, οι χρήστες μπορεί να προτιμούν διαφορετικά χρώματα, μεγέθη, γραμματοσειρές, ή ακόμα και να χρησιμοποιούν εναλλακτικές συσκευές απεικόνισης όπως κινητά ή PDA. Η ανεξαρτησία μεταξύ Model και View, επιτρέπει την προσθήκη νέων τύπων View στο σύστημα χωρίς να επηρεαστεί καθόλου το Model.

Επιπλέον , λόγω της ανεξαρτησίας μεταξύ των τμημάτων της εφαρμογής, η χρήση της MVC διευκολύνει σε μεγάλο βαθμό τις διαδικασίες ελέγχου της τελικής εφαρμογής.

Από την άλλη, η ύπαρξη των διακριτών αυτών τμημάτων στην αρχιτεκτονική MVC, αυξάνει την πολυπλοκότητα της τελικής εφαρμογής. Κατά την ανάπτυξη του κώδικα θα πρέπει να λαμβάνεται υπόψη από τους προγραμματιστές ο τρόπος ανάπτυξης και λειτουργίας του View αλλά και η γενικότερη φύση τους. Για παράδειγμα, αν το Model υφίσταται συχνές αλλαγές μπορεί το View να κατακλυστεί από αιτήματα για ενημέρωση με αποτέλεσμα την καθυστέρηση στην τελική εμφάνιση των αποτελεσμάτων.



Εικόνα 2: Αρχιτεκτονική προτεινόμενης λύσης

3 Τεχνολογίες που χρησιμοποιήθηκαν

Στο παρόν κεφάλαιο θα παρουσιαστούν και θα αναλυθούν οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της web εφαρμογής και της mobile εφαρμογής που έχουν σκοπό την αυτοματοποίηση της διαδικασίας δημιουργίας του προγράμματος των επιτηρήσεων των μαθημάτων κατά τη διάρκεια της εξεταστικής περιόδου.

3.1 Mongo DB

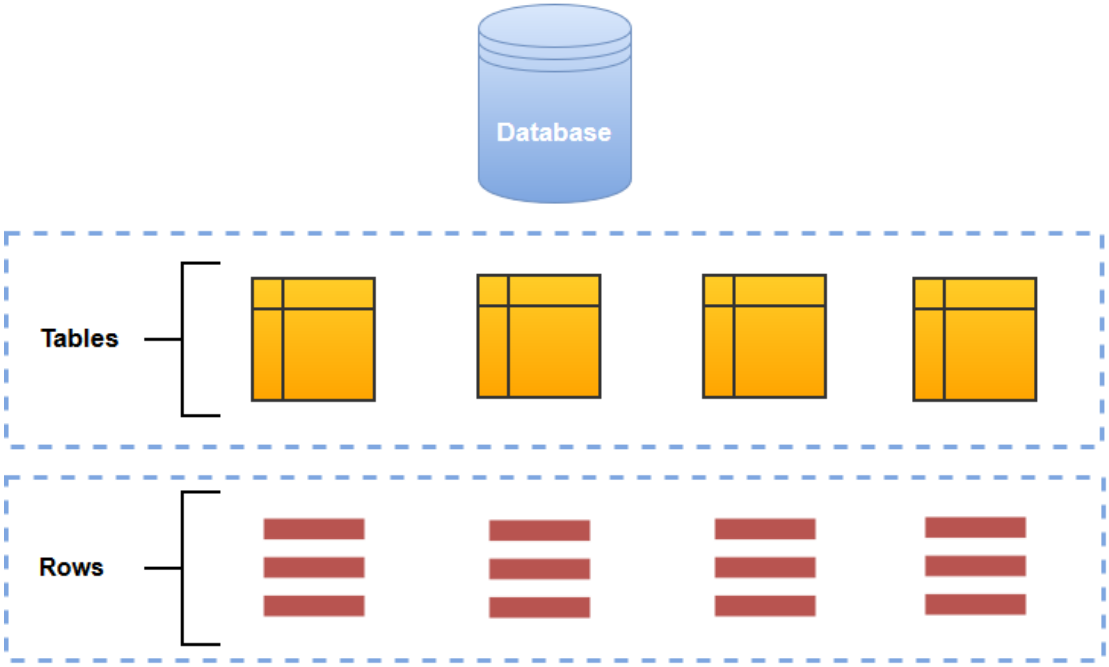
Το πρώτο μέλημά μας πριν ξεκινήσουμε την ανάπτυξη μιας web εφαρμογής αφορά την βάση δεδομένων που θα χρησιμοποιηθεί για την αποθήκευση των δεδομένων μας. Ο σκοπός ήταν να φιλοξενηθεί η βάση δεδομένων στο Cloud επομένως επιλέχθηκε βάση δεδομένων τεχνολογίας Mongo DB.

Οι βάσεις δεδομένων τύπου Mongo DB ανήκουν σε μια ευρύτερη κατηγορία βάσεων δεδομένων, αυτή των NoSQL (Not Only SQL). Οι βάσεις δεδομένων που ανήκουν στην κατηγορία αυτή έχουν σαν κύριο χαρακτηριστικό την μη τήρηση του μοντέλου RDBMS (Relation Database Management System) το οποίο χρησιμοποιεί η συντριπτική πλειοψηφία των βάσεων δεδομένων. Αντίθετα οι NoSQL βάσεις υιοθετούν έναν εντελώς διαφορετικό τρόπο για την διαχείριση των δεδομένων μέσα στη βάση (data manipulation), καθώς δεν υπακούν σε κάποιο δομημένο σύστημα για τα δεδομένα που φιλοξενούν, όπως είναι οι πίνακες, ούτε χρησιμοποιούν κάποια Structured Query Language (SQL) για την διαχείριση των δεδομένων. Χρησιμοποιούν αποκλειστικά non-relational τρόπους οργάνωσης και διαχείρισης δεδομένων [18].

Τα βασικά χαρακτηριστικά των NoSQL συστημάτων βάσεων δεδομένων είναι [20]:

1) Η μη χρήση της SQL για την διαχείριση των δεδομένων
2) δεν μπορεί να εγγυηθεί ότι οι διεργασίες στην βάση δεδομένων θα γίνονται αξιόπιστα
3) είναι βελτιστοποιημένα ώστε να ανακτούν και να επισυνάπτουν δεδομένα
4) έχουν μια ιδιαίτερη αρχιτεκτονική και φιλοσοφία λειτουργίας.

Οι βάσεις δεδομένων τύπου NoSQL αναπτύχθηκαν και εξελίχθηκαν καθώς δημιουργήθηκε η ανάγκη για την διαχείριση τεράστιου όγκου δεδομένων από τις μεγαλύτερες εταιρίες πληροφορικής όπως η Google ή η Amazon, τον οποίο δεν μπορούσαν να διαχειριστούν τα παραδοσιακά RDBMS συστήματα. Τα NoSQL συστήματα έχουν την δυνατότητα να διαχειρίζονται μεγάλες ποσότητες δεδομένων χωρίς να διατηρούν απαραίτητα κάποια συγκεκριμένη δομή. Επιπλέον οι μέθοδοι που χρησιμοποιούν αξιοποιούν μια αρχιτεκτονική που επιτρέπει την κατανεμημένη λειτουργία του συστήματος. Θεωρητικά μπορούν να προστεθούν άπειροι servers για την κατανεμημένη επεξεργασία των δεδομένων του συστήματος.



Εικόνα 3: Μοντέλο σχεσιακής βάσης δεδομένων

Στα πλεονεκτήματα που παρουσιάζουν οι NoSQL βάσεις δεδομένων μπορεί να προστεθεί η ελαστικότητα που παρουσιάζουν στην βελτίωση της επίδοσής τους [20]. Επιπλέον από οικονομική άποψη συμφέρουν, σε αντίθεση με τα παραδοσιακά RDBMS συστήματα, καθώς χρησιμοποιούν τεχνολογίες Cloud. Πιο συγκεκριμένα, όταν θέλουμε να επεκτείνουμε η να βελτιώσουμε τα RDBMS συστήματα συνήθως προσθέτουμε καλύτερους επεξεργαστές ή μνήμη RAM ενώ στις NoSQL βάσεις προσθέτουμε νέους κόμβους για την επεξεργασία ακόμα περισσότερων δεδομένων. Επιπλέον τα NoSQL συστήματα είναι φθηνότερα και στη συντήρηση καθώς δεν απαιτούν μεγάλη διαχείριση από τον άνθρωπο αφού έχουν πιο απλά μοντέλα

δεδομένων και χρησιμοποιούν αυτόματες μεθόδους για επιδιόρθωση και διανομή δεδομένων.

Οι NoSQL βάσεις δεδομένων χωρίζονται σε ορισμένες κατηγορίες:

1) οι Key-values βάσεις,
2) οι Column Family βάσεις
3) οι Document (εγγραφοκεντρικες) βάσεις και
4) οι Graph βάσεις.

Στην πρώτη κατηγορία, η κύρια ιδέα είναι η ύπαρξη ενός hash table στον οποίο υπάρχει ένα μοναδικό κλειδί και κάποιος δείκτης που δείχνουν σε έναν συγκεκριμένο στοιχείο. Για την καλύτερη απόδοση του συστήματος, χρησιμοποιούνται μηχανισμοί cache.

Οι Column family βάσεις σχεδιάστηκαν για την διαχείριση μεγάλων ποσων δεδομένων τα οποία είναι κατανεμημένα σε διάφορους servers. Όπως και στην προηγούμενη κατηγορία, έτσι κι εδώ υπάρχουν κλειδιά τα οποία όμως δείχνουν σε περισσότερα από ένα στοιχεία. Οι γραμμές εδώ αναγνωρίζονται από ένα μοναδικό κλειδί γραμμής ενώ οι στήλες είναι οργανωμένες σε «οικογένειες στηλών», από όπου και προκύπτει η ονομασία της κατηγορίας.

Οι Document βάσεις, είναι παρόμοιες με τις key-value βάσεις. Ωστόσο τα δεδομένα στην περίπτωση αυτή είναι οργανωμένα από συλλογές key-valued συλλογών δεδομένων.

Η κατηγορία των Graph databases βασίζεται σε κόμβους (nodes), στις σχέσεις μεταξύ αυτών των κόμβων αλλά και στις ιδιότητές τους. Εδώ αντί για πίνακες με στήλες και γραμμές, υπάρχει ένα ευέλικτο γραφικό μοντέλο το οποίο μπορεί να χρησιμοποιηθεί και να αναπτυχθεί παράλληλα σε πολλά μηχανήματα.

Οι πιο γνωστές NoSQL βάσεις δεδομένων είναι οι:

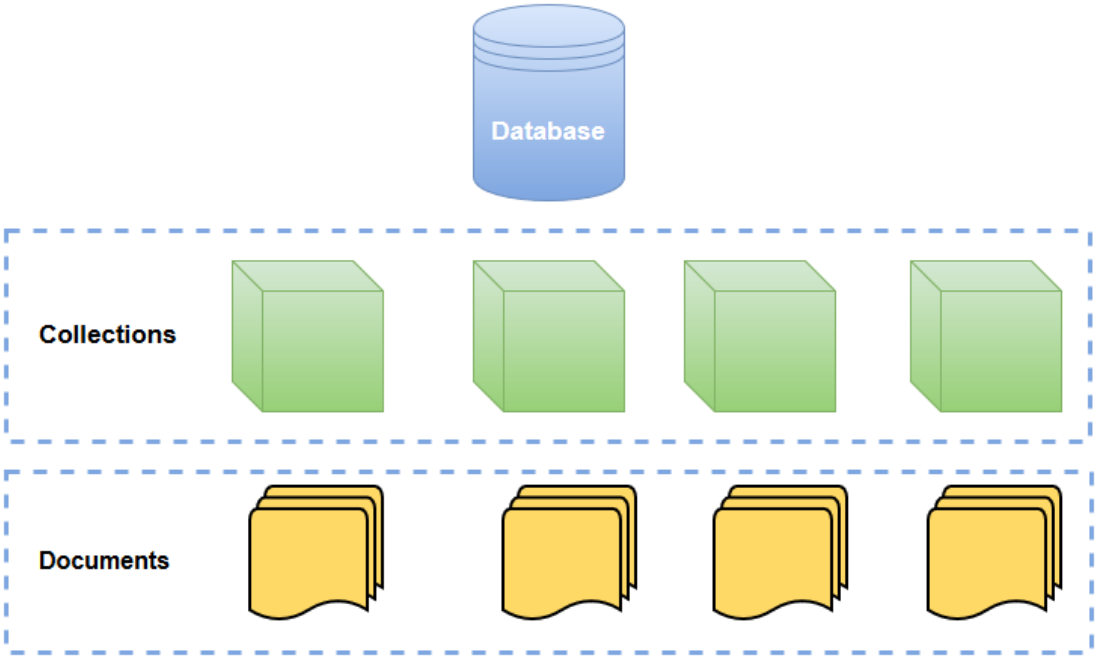
- Cassandra: Η Cassandra δημιουργήθηκε από την Facebook ωστόσο ανήκει πια στην Apache. Ανήκει στις column oriented NoSQL βάσεις και είναι ανοικτού κώδικα.
- Dynamo και SimpleDB: οι βάσεις αυτές δημιουργήθηκαν από την Amazon. Η Dynamo είναι η πιο γνωστή Key-Value NoSQL βάση

δεδομένων, ενώ η SimpleDB αποτελεί μέρος των Cloud Services που προσφέρει η Amazon.

- BigTable: η βάση αυτή κατασκευάστηκε από τη Google, και η χρήση της είναι περιορισμένη καθώς πραγματοποιείται μόνο μέσω του Google App Engine. Τεχνικά, είναι μια column oriented βάση κλειστού κώδικα.
- Neo4J: αποτελεί μια graph NoSQL βάση ελεύθερου κώδικα.
- CouchDB και MongoDB: αυτές οι δύο βάσεις αποτελούν τις πιο γνωστές open source document oriented NoSQL βάσεις.

Η MongoDB είναι ίσως η δημοφιλέστερη εγγραφοκεντρική βάση δεδομένων αυτή τη στιγμή. Η ονομασία της προέρχεται από τη λέξη "humongous" [33] που σημαίνει τεράστιος, τερατώδης. Για τη βάση αυτή οι πίνακες, τα κλειδιά, τα joins είναι άγνωστα. Το αντίστοιχο της εγγραφής (record) των σχεσιακών βάσεων, στην MongoDB ονομάζεται έγγραφο (document) και περιέχει πεδία που απαρτίζονται από ζεύγη κλειδιών-τιμών (key-value pairs). Τα έγγραφα της MongoDB είναι παρόμοια με τα αντικείμενα σε μορφή JSON. Οι τιμές των πεδίων μπορούν να περιέχουν άλλα έγγραφα, πίνακες ακόμα και πίνακες από έγγραφα. Το αντίστοιχο του πίνακα (table) των σχεσιακών βάσεων, στην MongoDB ονομάζεται συλλογή (collection). Μια συλλογή μοιάζει με ένα κουτί με μια ετικέτα πάνω του. Θα μπορούσαμε να το φανταστούμε με ένα απλό κουτί αποθήκευσης που έχει ετικέτα "Βιβλία". Το λογικό είναι στο κουτί αυτό να αποθηκευτούν βιβλία. Ωστόσο δεν απαγορεύεται με κάποιον τρόπο να αποθηκεύσουμε στο κουτί αυτό και μολύβια. Κάτι αντίστοιχο συμβαίνει και στην MongoDB. Ένα παράδειγμα εγγράφου της MongoDB μπορεί να είναι το παρακάτω:

```
{
    "id" : 18,
    "lastName" : "Μαρίνη",
    "firstName" : "Ελένη",
    "fathersName" : "Δημήτριος",
    "grades" : [
        {
            "lessonName" : "Java",
            "grade" : 9
        },
        {
            "lessonName" : "Cloud",
            "grade" : 9
        }
    ]
}
```



Εικόνα 4: Μοντέλο της βάσης δεδομένων MongoDB

Η MongoDB για να αποθηκεύσει δεδομένα χρησιμοποιεί ένα ανοιχτού τρόπου πρότυπο δεδομένων, το BSON (Binary - JSON), γεγονός που την καθιστά ακόμα πιο γρήγορη στην αναζήτηση εγγράφων.

Μερικά από τα πλεονεκτήματα που παρουσιάζει η χρήση εγγράφων αντί των κλασικών εγγραφών δίνονται ακολούθως. Τα έγγραφα που χρησιμοποιούν οι εγγραφοκεντρικές βάσεις δεδομένων, αντιστοιχούν στους (έμφυτους -native) τύπους δεδομένων πολλών μοντέρνων γλωσσών προγραμματισμού (όπως η javascript) διευκολύνοντας πολύ τον προγραμματισμό εφαρμογών. Επιπλέον η δυνατότητα αποθήκευσης ενσωματωμένων έγγραφων και πινάκων μειώνει την ανάγκη για δαπανηρές πράξεις τύπου join. Τέλος, η δυνατότητα χρήσης αδόμητου ή ημιδομημένου σχήματος υποστηρίζει την εύκολη επέκταση των εφαρμογών.

Συγκεκριμένα η MongoDB παρέχει υψηλή απόδοση, υψηλή διαθεσιμότητα και αυτόματη κλιμάκωση. Μερικά από τα χαρακτηριστικά της βάσης που συντελούν στα παραπάνω είναι [16][17]:

- Υποστηρίζει την ενσωμάτωση αντικειμένων στο μοντέλο των δεδομένων της, γεγονός που μειώνει την ανάγκη για πολλαπλές αναγνώσεις και εγγραφές στους χώρους αποθήκευσης.
- Παρέχει δείκτες (indexes) οι οποίοι μπορούν να δεικτοδοτήσουν κλειδιά και σε ενσωματωμένα έγγραφα (documents).
- Η υπηρεσία λειτουργίας αντιγράφων της βάσης παρέχει την αυτόματη ανάκαμψή της από βλάβες και από πλεονασμό.
- Παρέχει οριζόντια κλιμάκωση ως και προσφέρει τη δυνατότητα κατακερματισμού των δεδομένων σε ένα σύνολο υπολογιστών.

Γενικότερα η MongoDB αποτελεί μια βάση δεδομένων που είναι βελτιστοποιημένη σε ταχύτητα και επεκτασιμότητα και μπορεί να εκτελεστεί σε οποιοδήποτε λειτουργικό σύστημα.

Κάποιες βασικές εντολές της MongoDB είναι οι ακόλουθες.

Δημιουργία Collection:

```
db.createCollection(COLLECTION_NAME, options);
```

Διαγραφή collection:

```
db.COLLECTION_NAME.drop();
```

Εισαγωγή document σε collection:

```
db.COLLECTION_NAME.insert({json object});
```

Αναζήτηση documents σε collection που εχουν την τιμή Eleni στο Attribute name:

```
db.COLLECTION_NAME.find({"name":"Eleni"});
```

3.2 Apache Maven

Το Apache Maven αποτελεί ένα build automation εργαλείο και έχει σαν σκοπό να περιγράφει πώς είναι δομημένο το λογισμικό και να συγκεντρώνει τις απαραίτητες για το έργο (project) εξαρτήσεις από εξωτερικές πηγές ή καταλόγους του συστήματος [21]. Αυτές οι εξαρτήσεις καταγράφονται σε ένα αρχείο αυστηρής δομής xml που ονομάζεται pom (project object management). Η λέξη maven σημαίνει

"συσσωρευτής της γνώσης". Για να κάνουμε build ένα maven project αρκεί να τρέξουμε την εντολή `mvn install` στην γραμμή εντολών και στον φάκελο που περιέχει το project μας. Η εντολή αυτή τρέχει και τα tests που έχουμε γράψει στο project μας. Εάν επιθυμούμε να αγνοηθούν τα test κατά το build αρκεί να τρέχουμε την εντολή `mvn install -DskipTests.`

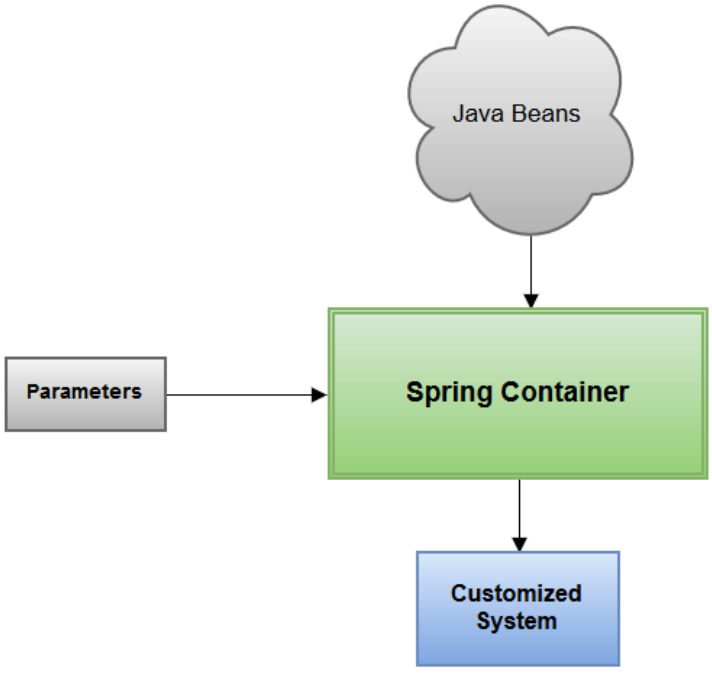
3.3 Spring Framework

Το Spring είναι προϊόν ανοιχτού κώδικα, της εταιρείας Springsource, και αποτελεί ένα ελαφρύ πλαίσιο Java εφαρμογών. Το Spring διακρίνεται από τα εξής βασικά χαρακτηριστικά [4][5]:

- Διαχειρίζεται τις ρυθμίσεις της εφαρμογής, βασιζόμενο σε JavaBeans και εφαρμόζοντας την αρχιτεκτονική αρχή του Dependency Injection.
- Συνδυάζεται με ένα πλήθος τεχνολογιών
- Οι δευτερεύουσες λειτουργίες διαχωρίζονται από το business logic της εφαρμογής.
- Έρχεται με το δικό του MVC Web framework που παρέχει πολλαπλές τεχνολογίες στο αντικείμενο View.
- Παρέχει διαδικασίες για authentication / authorization, μέσω της βιβλιοθήκης, Spring security, που υποστηρίζουν διάφορα πρωτόκολλα (όπως το LDAP) .

Το Spring Framework έχει σαν κύριο στόχο την απλοποίηση της διαδικασίας ανάπτυξης εφαρμογών. Επιπλέον, διευκολύνει την συνεργασία με άλλες τεχνολογίες γεγονός που το κάνει ένα εξαιρετικά ευέλικτο πλαίσιο ανάπτυξης εφαρμογών.

Αποτελείται από έναν Container ο οποίος βασίζεται στη λογική του Dependency Injection, κατά την οποία υποδεικνύεται σε ένα τμήμα μιας εφαρμογής ποια άλλα κομμάτια έχει δικαίωμα να χρησιμοποιήσει. Σε όλες τις εφαρμογές που αναπτύσσονται με βάση το Spring Framework, υπάρχει ένας Container ο οποίος αποτελεί την βάση του συστήματος. Όλα τα Beans που περιέχει γίνονται δηλώνονται και δημιουργούνται από τον Container. Τον ρόλο του Container παίζει Το interface org.springframework.context.ApplicationContext και το implementation του interface αυτού είναι το WebApplicationContext. Το κεντρικό servlet (το οποίο κάνει extend το HttpServlet) πάνω στο οποίο είναι σχεδιασμένο το Spring Web MVC Framework είναι το DispatcherServlet. Το DispatcherServlet αποτελεί έναν front controller ο οποίος αναλαμβάνει την παραλαβή των requests και την προώθησή τους σε άλλους Controllers και Views. Όπως όλα τα servlets σε μια τυπική web εφαρμογή, ο DispatcherServlet δηλώνεται, όπως φαίνεται στην Εικόνα 6, μέσα στο web.xml μαζί με το mapping των url-patterns που θα εξυπηρετεί.



Εικόνα 5: Βασική λειτουργία του Spring

```
<web-app>
...
<servlet>
<servlet-name>Spring MVC Dispatcher Servlet</servlet-name>
<servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
<init-param>
<param-name>contextConfigLocation</param-name>
<param-value>/WEB-INF/spring/*.xml</param-value>
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>Spring MVC Dispatcher Servlet</servlet-name>
<url-pattern>*.info</url-pattern>
</servlet-mapping>
...
</web-app>
```

Εικόνα 6: Δήλωση του Dispatcher Servlet στο web.xml

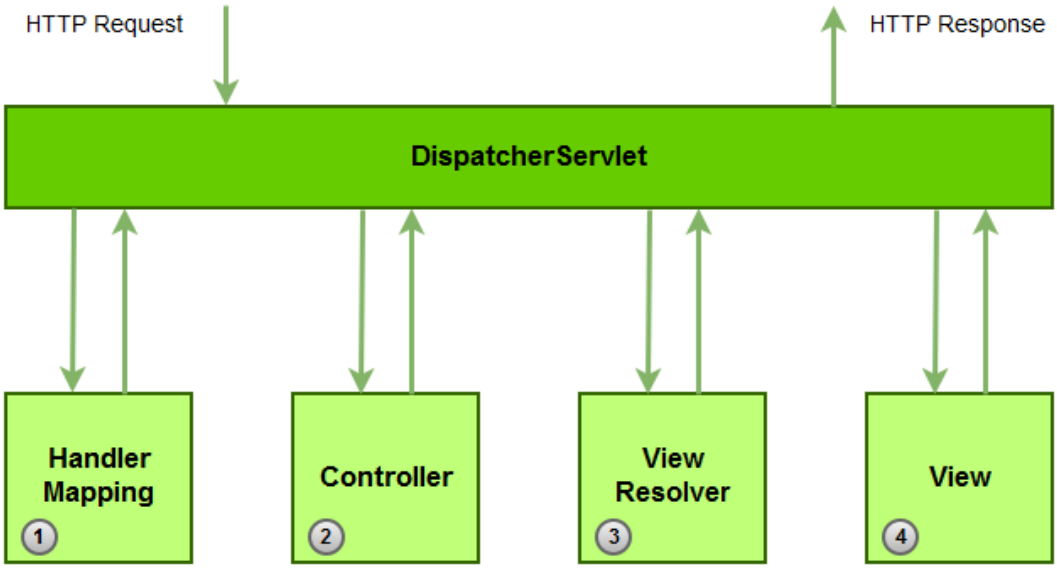
Σε μία Spring MVC εφαρμογή ο DispatcherServlet χρησιμοποιεί Beans (Controllers, Handlers, Resolvers , Services, DAOs κτλ.) του Spring Framework για να επεξεργαστεί τα requests και να τα περάσει στα αντίστοιχα views. Τα beans αυτά δηλώνονται στα παραμετρικό αρχείο app-config.xml. Ο DispatcherServlet πρέπει να γνωρίζει πώς θα διαχειριστεί τα requests που λαμβάνει. Αυτό επιτυγχάνεται με την παραμετροποίηση του στα αρχεία app-config.xml και mvc-config.xml που βρίσκονται στο /WEB-INF/spring/. Μέσα στο app-config.xml δηλώνοντας το εξής: `<context:component-scan base-package="com.package.test" />` ο DispatcherServlet ενημερώνεται ότι πρέπει να ψάξει για τα beans στο πακέτο com.package.test. Ουσιαστικά όταν γίνεται instantiated ο DispatcherServlet, ψάχνει σε αυτό το πακέτο ποιές κλάσεις χρησιμοποιούν το annotation @Component ή πιο συγκεκριμένα τα @Controller και @Repository. Για να γνωρίζει ο DispatcherServlet ότι πρέπει να ψάξει για annotations θα πρέπει στο mvc-config.xml να δηλώσουμε το `<mvc:annotation-driven />`. Επιπλέον στο mvc-config.xml δηλώνεται ένα ειδικό bean του Spring MVC, ο InternalResourceViewResolver, με τον εξής τρόπο:

```
<bean
class="org.springframework.web.servlet.view.InternalResourceViewResol
ver">
        <property name="prefix" value="/WEB-INF/views/"/>
        <property name="suffix" value=".jsp"/>
</bean>
```

Ο InternalResourceViewResolver είναι ένας resolver για τα views του εφαρμογής τα οποία δηλώνεται ότι είναι τύπου jsp και βρίσκονται στο

φάκελο /WEB-INF/views/. Με τις παραπάνω κινήσεις ενημερώνεται ο DispatcherServlet για τους Controllers και τα Views του συστήματος και επομένως γνωρίζει που να προωθεί κάθε φορά τα requests.

Η πορεία επεξεργασίας των αιτημάτων (requests) που δέχεται ο DispatcherServlet του Spring παριστάνεται στην εικόνα 7 και αποτελείται από τα εξής βήματα [5]. Όταν φτάσει ένα HTTP request στον DispatcherServlet, αυτός συμβουλεύεται τον HandlerMapping έτσι ώστε να καλέσει τον σωστό Controller. Ο Controller στη συνέχεια παίρνει το αίτημα (request) και καλεί το κατάλληλο service το οποίο τροποποιεί το μοντέλο και επιστρέφει το όνομα του view στον DispatcherServlet, ο οποίος με τη βοήθεια του ViewResolver επιλέγει το κατάλληλο view και αφού του "δώσει" τα δεδομένα του model, το view αυτό εμφανίζεται στον browser.



Εικόνα 7: Πορεία HTTP request στο Spring

3.3.1 Spring Beans

Τα αντικείμενα που αποτελούν τον πυρήνα μιας εφαρμογής βασισμένης στο Spring Framework και που διαχειρίζονται από τον container του Spring ονομάζονται beans. Το bean αποτελεί ένα αντικείμενο το οποίο αρχικοποιείται, συναρμολογείται και διαχειρίζεται από τον container του Spring. Τα beans κατασκευάζονται μέσω των

configuration metadata που δηλώνουμε στον container, για παράδειγμα με τη μορφή ενός xml element μέσα σε `<bean/>` tags. Παράδειγμα δήλωσης ενός bean είναι

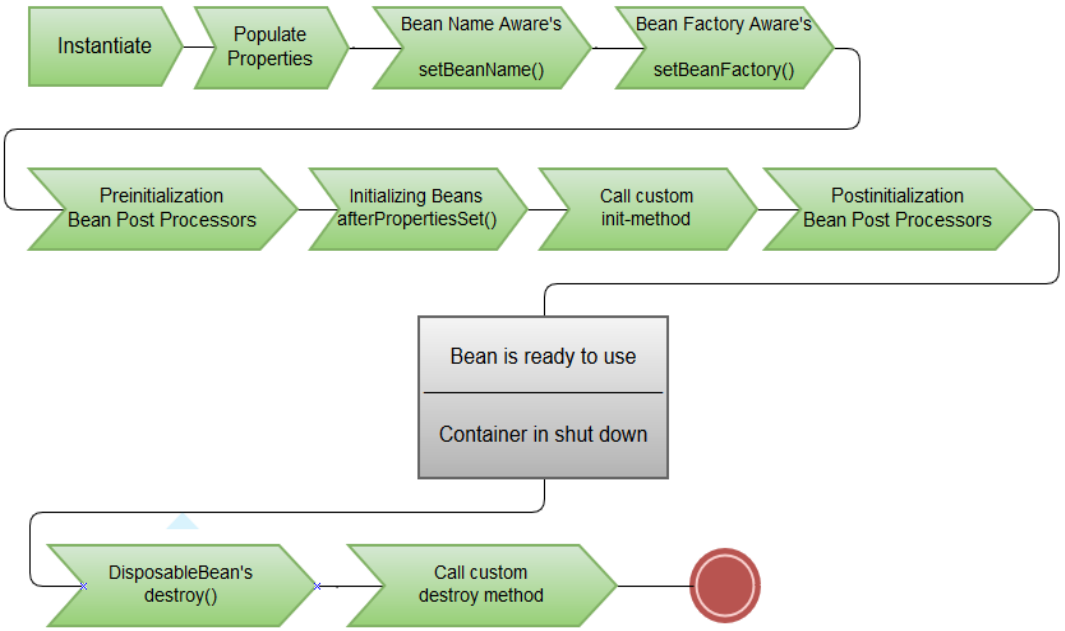
```
<!-- A simple bean definition -->
<bean id="..." class="...">
    <!-- collaborators and configuration for this bean go here -->
</bean>
```

Ο ορισμός ενός bean περιέχει πληροφορίες γι αυτό οι οποίες στο σύνολό τους ονομάζονται **configuration metadata** και είναι απαραίτητες στον container ώστε να γνωρίζει τα ακόλουθα [5].

- πώς να κατασκευάσει ένα bean
- λεπτομέρειες για τον κύκλο ζωής του bean
- τις εξαρτήσεις του bean

Τα configuration metadata αρχεία μπορεί να έχουν την μορφή xml ή να βασίζονται απλά σε annotations.

Όλη η παραπάνω πληροφορία που έχει δηλωθεί στο configuration metadata, μεταφράζεται σε ένα σύνολο από τις ακόλουθες λεπτομέρειες οι οποίες συνθέτουν την δήλωση ενός bean.



Εικόνα 8: Κύκλος ζωής ενός bean

Πίνακας 1: Βασικά attributes ενός Bean [5]

class	Το attribute αυτό είναι υποχρεωτικό και προσδιορίζει την κλάση που πρέπει να χρησιμοποιηθεί για να δημιουργηθεί το

	bean.
name	Το attribute αυτό προσδιορίζει το μοναδικό αναγνωριστικό του bean. Στα configuration metadata που έχουν τη δομή XML, χρησιμοποιούμε το id και/ή το name attribute για να ορίσουμε τα αναγνωριστικά του bean.
scope	Το attribute αυτό προσδιορίζει το scope των αντικειμένων που δημιουργούνται από μια συγκεκριμένη δήλωση bean.
constructor-arg, properties, autowiring mode	Το attribute αυτό χρησιμοποιείται για να δηλώσεις τις εξαρτήσεις του bean.
lazy-initialization mode	Τα bean που περιέχουν αυτό το attribute δηλώνουν στον container να τα δημιουργήσει την πρώτη φορά που κάποιος θα τα ζητήσει και όχι απο την αρχή.
initialization method	Το attribute αυτό δηλώνει μια callback μέθοδο η οποία θα κληθεί αφού όλες οι απαραίτητες ιδιότητες του bean έχουν κατασκευαστεί από τον container.
destruction method	Το attribute αυτό δηλώνει μια callback μέθοδο η οποία θα κληθεί όταν ο container που περιέχει το bean καταστραφεί.

3.4 Javascript

Η javascript είναι μια scripting γλώσσα η οποία έχει σχεδιαστεί και χρησιμοποιείται για να εισάγουμε την έννοια της διαδραστικότητας στις html σελίδες. Είναι μια ερμηνευτική γλώσσα, δηλαδή το script εκτελείται χωρίς να έχει περάσει από την διαδικασία της σύνταξης [25]. Με την πάροδο των χρόνων γίνεται όλο και δημοφιλέστερη και έχουν δημιουργηθεί πανίσχυρες βιβλιοθήκες που την υποστηρίζουν και διευκολύνουν την ανάπτυξη εφαρμογών σε web, πλατφόρμες και κινητά. Το γεγονός ότι η γλώσσα υποστηρίζει τόσο τον functional προγραμματισμό όσο και τον αντικειμενοστραφή (object oriented) την κάνει ακόμα πιο δημοφιλή. Η javascript επιτρέπει γρήγορη επικοινωνία ανάμεσα στον browser και τον server. Χρησιμοποιείται με ενιαίο τρόπο στο διαδίκτυο, ανεξαρτήτως από την γλώσσα προγραμματισμού στον server (PHP, Java κλπ) και δουλεύει ακόμα και όταν ο browser είναι offline. 'Εχει εδραιωθεί στο διαδίκτυο, ενώ παράλληλα έχει αρχίσει να επεκτείνεται και σε άλλους τομείς καθώς οι βιβλιοθήκες που έχουν υλοποιηθεί σε αυτήν την κάνουν ακόμα πιο ευέλικτη και πιο αναγνωρίσιμη.

Μια από αυτές είναι η JQuery η οποία χρησιμοποιείται ευρέως και απλοποιεί τον προγραμματισμό καθώς με την βοήθεια της μπορούν να υλοποιηθούν λειτουργίες

με πολύ απλό τρόπο, οι οποίες με την συμβατική JavaScript θα ήταν ιδιαίτερα πολύπλοκες. Έχει πολύ μεγάλο αριθμό δυνατοτήτων και λειτουργιών, την υποστηρίζει ευρύ φάσμα open-source community και θεωρείται πλέον από τις απαραίτητες βιβλιοθήκες για την υλοποίηση μιας διαδικτυακής εφαρμογής.

3.5 Angular JS

Η Angular JS αποτελεί ένα ενδιαφέρον client - side framework ανοιχτού λογισμικού της Google, το οποίο αλλάζει τα δεδομένα στον τομέα στον διαδικτυακών εφαρμογών, καθώς έχει σχεδιαστεί ώστε να αποδεσμεύει το DOM από την λογική της εφαρμογής γεγονός που διευκολύνει τον έλεγχο του κώδικα [6]. Ο έλεγχος της εφαρμογής είναι τόσο σημαντικός όσο και η διαδικασία υλοποίησης της. Η δυσκολία των δοκιμών επηρεάζεται δραματικά από τον τρόπο με τον οποίο είναι δομημένος ο κώδικας. Ανεξαρτητοποιεί πλήρως τον πελάτη (browser) από τον διακομιστή (server) γεγονός που επιτρέπει επιτρέπει την επαναχρησιμοποίηση κώδικα και στις δύο πλευρές. Ακολουθεί το μοτίβο MVC της μηχανικής λογισμικού και ενθαρρύνει τη χαλαρή σύνδεση μεταξύ view, controller και model [8]. Πρέπει επίσης να αναφερθεί ότι μεγάλο μέρος της επιβάρυνσης στο back-end μειώνεται και δημιουργούνται πολύ ελαφρύτερες web εφαρμογές.

Γενικότερα, το AngularJS δένει την HTML (views) με αντικείμενα JavaScript (models). Σημαντικό είναι το γεγονός ότι οταν συμβαίνει μια αλλαγή στα models, η σελίδα ανανεώνεται αυτόματα. Συμβαίνει επίσης και το αντίθετο, όταν δηλαδή ένα υπάρχει κάποια αλλαγή στο view, το AngularJS χειρίζεται την κατάσταση αυτόματα χωρίς να χρειάζεται να γίνει κάποια επιπλέον ενέργεια στην HMTL, ή στην JQuery με την σύνδεση κάποιου event (για παράδειγμα κλικ σε κάποιο κουμπί).

Ένα από τα μεγαλύτερα πλεονεκτήματα της Angular JS είναι ότι παρέχει έναν τρόπο αυτόματης ενημέρωσης του τμήματος του view κάθε φορά που υπάρχει κάποια αλλαγή στο model, αλλά επιτρέπει και την ενημέρωση του model κάθε φορά που έχουμε κάποια αλλαγή στο view [7]. Ο αυτόματος αυτός μηχανισμός ονομάζεται data binding και ουσιαστικά μας απελευθερώνει από το να ασχοληθούμε με το DOM.

Επιπλέον λόγω της MVC αρχιτεκτονικής που ακολουθεί, παρέχει Controllers που συγκεντρώνουν όλη τη συμπεριφορά πίσω από το DOM, η οποία είναι γραμμένη

με καθαρή μορφή χρησιμοποιώντας απλή JavaScript. Επίσης βασικές λειτουργίες, όπως η επικύρωση μιας φόρμας στον browser μπορούν να γίνουν χωρίς να χρειαστεί να γράψουμε καθόλου κώδικα JavaScript, καθώς η Angular JS επιτρέπει τη δήλωση κανόνων επικύρωσης στο κομμάτι της HTML. Οι ενσωματωμένες υπηρεσίες που παρέχει για την επικοινωνία με τρίτες βιβλιοθήκες ενισχύουν το χειρισμό της ασύγχρονης επιστροφής δεδομένων και απλοποιούν τον κώδικα. Ένα μοναδικό χαρακτηριστικό που κάνει την Angular ακόμα πιο ξεχωριστή είναι η δυνατότητα χρήσης directives που μας επιτρέπουν να έχουμε ειδική HTML σύνταξη για την εφαρμογή μας αλλά και να επαναχρησιμοποιούμε πολλά στοιχεία. Ακόμα η Angular μπορεί να συνδυαστεί άψογα με άλλες τεχνολογίες χωρίς να απαιτεί πλήρη δέσμευση. Τέλος, έχει σχεδιαστεί έτσι ώστε να είναι ελέγξιμη παρέχοντας μηχανισμούς για Unit testing και front end testing.

Η κατανόηση των ακόλουθων πέντε συστατικών της Angular, αποτελεί το πρώτο βήμα για κάποιον που επιθυμεί να εξοικειωθεί μαζί της και να την χρησιμοποιήσει για την ανάπτυξη εφαρμογών. Τα βασικά αυτά συστατικά είναι: τα directives και οι controllers , που αναφέρθηκαν και παραπάνω , το scope, τα services, και η δήλωση των εξαρτήσεων της εφαρμογής μας.

Το πρώτο πράγμα που πρέπει να κατανοήσουμε είναι η έννοια του scope. Για την επικοινωνία μεταξύ του view και του controller η Angular χρησιμοποιεί το scope το οποίο αποτελεί ουσιαστικό το πεδίο μεταβλητών που είναι ορατό τόσο από τη JavaScript όσο και από την HTML. Οι μεταβλητές που ανήκουν στο scope δηλώνονται με τον χαρακτήρα "$". Για παράδειγμα αν δηλώσουμε μια μεταβλητή user στο scope όπως φαίνεται στην εικόνα 9

```
1
2   angular.module('MyApp')
3
4     .controller('MyContoller',function($scope){
5
6       $scope.user={username:'eleni',
7                    password:'1234'};
8
9   });
10
```

Εικόνα 9: Δήλωση μεταβλητής user στο scope στον controller

τότε αρκεί στο view να χρησιμοποιήσουμε τα {{}} όπως φαίνεται στην εικόνα 10, για να εμφανίσουμε την τιμή της. Το ίδιο μπορεί να συμβεί και με την αντίθετη φορά. Δηλαδή να περάσουμε σε κάποια μεταβλητή που είναι δηλωμένη στο scope, κάποια τιμή που του έχει δοθεί απευθείας από το view. Αυτό γίνεται με τη χρήση της λέξης-κλειδιού ng-model όπως φαίνεται στις εικόνες 11 και 12. Εδώ δηλώνουμε και ένα κουμπί το οποίο όταν πατηθεί καλείται η μέθοδος printEmail() η οποία είναι και αυτή ορισμένη στο scope για να μπορεί να τη "δει" το κουμπί. Όταν λοιπόν κληθεί η μέθοδος αυτή εκτυπώνεται η τιμή που έχει εισαχθεί το πεδίο που έχουμε ορίσει για το email.

```
1
2
3   <span>Username: {{user.username}}</span>
4   <span>Username: {{user.password}}</span>
5
```

Εικόνα 10: Εμφάνιση μεταβλητής user στο view

```
6
7   <input type="text" ng-model="email"></input>
8
9   <button ng-click="printEmail()">Click me!</button>
10
```

Εικόνα 11: Αυτόματη δήλωση στο view της μεταβλητή email στο scope

```
9
10      $scope.printEmail= function(){
11          console.log("Email is "+$scope.email);
12      }
13
14  });
15
```

Εικόνα 12: Μέθοδος για την εκτύπωση της μεταβλητής email στον controller

Όπως φαίνεται στην εικόνα 9 για να διαχειριστούμε και να οργανώσουμε τη λειτουργικότητα τις σελίδας μας, χρησιμοποιούμε τους controllers μέσα στους οποίους δηλώνονται και αναπτύσσονται όλες οι μέθοδοι που θα χρησιμοποιήσουμε. Περιέχουν δηλαδή όλη τη λογική της εφαρμογής μας. Έχουμε τη δυνατότητα για διαφορετικά view να ορίζουμε διαφορετικούς controllers. Αυτό το πετυχαίνουμε με την δήλωση του ονόματος του controller στην λέξη-κλειδί ng-controller μέσα σε

κάποιο tag element του view (HTML). Στο παράδειγμα με τον user στις εικόνες 9 και 10 αρκεί να δηλώσουμε τον controller μας όπως δείχνει η εικόνα 13, ώστε να μπορέσει το view να "δει" τη μεταβλητή user.

```
1  <div ng-controller="MyController">
2
3      <span>Username: {{user.username}}</span>
4      <span>Username: {{user.password}}</span>
5
6  </div>
```

Εικόνα 13: Δήλωση επιθυμητού Controller στο view

Τα Directives αποτελούν το πιο δύσκολο στην κατανόηση συστατικό της Angular. Πρακτικά μας δίνουν τη δυνατότητα να δημιουργήσουμε δικά μας tag elements τύπου html τα οποία θα δώσουν πρόσθετη λειτουργικότητα στην εφαρμογή μας. Το βασικό πλεονέκτημα των directives είναι ότι είναι επαναχρησιμοποιήσιμα. Για παράδειγμα όταν θέλουμε να εισάγουμε μια εικόνα στην HTML αρκεί να δηλώσουμε το tag <img> μέσα σε αυτην. Αν θέλουμε για πάραδειγμα να έχουμε ένα ειδικό επαναχρησιμοποιήσιμο tag της εμφάνιση σελιδοποίησης, για μια φόρμα login αλλά ακόμα και για μεγαλύτερα τμήματα κώδικα, η Angular μας δίνει τη δυνατότητα να δημιουργήσουμε τα δικά μας directives για το σκοπό αυτό. Η Angular παρέχει ένα σύνολο από δικά της directives - λέξεις κλειδιά τα οποία αναγνωρίζονται και οδηγούν στην εκτέλεση συγκεκριμένων εντολών. Για παράδειγμα η λέξη κλειδί ng-repeat χρησιμοποιείται στο επίπεδο του view όταν θέλουμε να εμφανίσουμε για ένα πλήθος όμοιων αντικειμένων κάποια τιμή, όπως συμβαίνει συνήθως στις λίστες ή στους πίνακες τις HTML. Έχοντας δηλώσει στον controller μας και στο $scope, έναν πίνακα από objects, όπως φαίνεται στην εικόνα 14, τότε μπορούμε να εμφανίσουμε τα βιβλία αυτά σε έναν πίνακα όπως φαίνεται στην εικόνα 15.

```
16  $scope.books=[{name:'Book1', author:'Author1'},
17                {name:'Book2', author:'Author2'}];
18
```

Εικόνα 14: Δήλωση μεταβλητής books στο scope

```
12
13 ▼ <table>
14       <tr ng-repeat="book in books ">
15             <td>{{ book.name }}</td>
16          <td>{{ book.author }}</td>
17       </tr>
18   </table>
19
```

Εικόνα 15: Χρήση λέξης-κλειδιού ng-repeat

η οποία τελικά θα μας εμφανίσει στο view το:

Book1 Author1

Book2 Author2

Όταν ένας Controller θέλει να χρησιμοποιήσει μια εξωτερική βιβλιοθήκη. Αν για παράδειγμα θέλουμε να εκτυπώσουμε κάτι στην κονσόλα, θα πρέπει να δηλώσουμε το $log στον controller για να χρησιμοποιήσουμε το αντικείμενο log όπως φαίνεται στην εικόνα 16. Με τον ίδιο τρόπο δηλώνονται και τα Services.

```
1
2   angular.module('MyApp')
3
4     .controller('MyContoller',function($scope, $log){
5
6       $scope.printEmail= function(){
7           console.log("Email is "+$scope.email);
8       }
9
10  });
```

Εικόνα 16: Δήλωση εξωτερικών εξαρτήσεων

Τα Services αποτελούν επίσης βασική έννοια της Angular JS και είναι αντικείμενα που εκτελούν κατά βάση παρόμοιες εντολές σε πολλά σημεία της εφαρμογής. Μέσω των Services για παράδειγμα μπορούμε να καλέσουμε rest services τα οποία θα μας φέρουν τα δεδομένα που χρειαζόμαστε. Δημιουργούνται μια φορά και δηλώνονται σαν dependency στους controllers που επιθυμούμαι να τα χρησιμοποιήσουν.

3.6 Τύποι εφαρμογών έξυπνων κινητών συσκευών

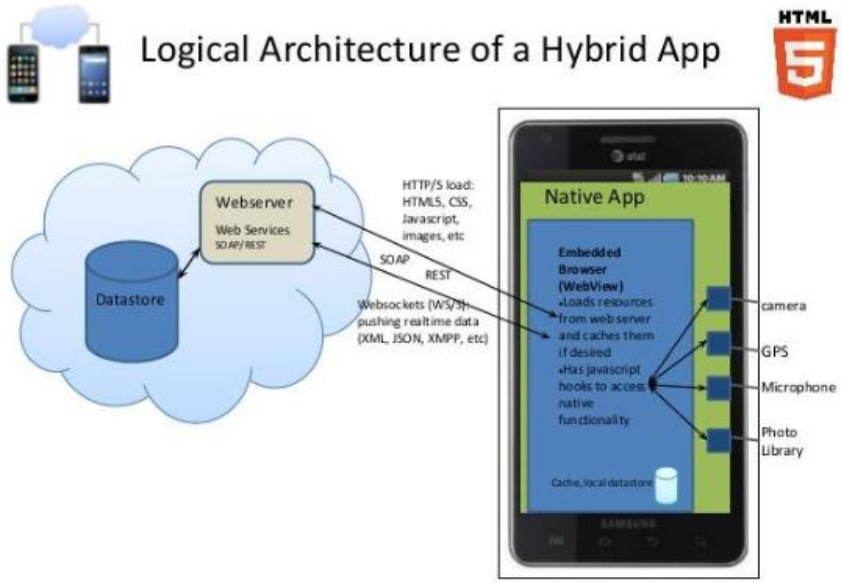
Οι εφαρμογές που προορίζονται για έξυπνες κινητές συσκευές, μπορεί να είναι μητρικές (native), διαδικτυακές (web), ή υβριδικές (hybrid) [13]. Οι native εφαρμογές προορίζονται και είναι κατάλληλες μόνο για ένα συγκεκριμένο λειτουργικό σύστημα/πλατφόρμα όπως είναι το Android ή το iOS. Κάθε πλατφόρμα αντιλαμβάνεται και υποστηρίζει εφαρμογές που έχουν αναπτυχθεί με τη χρήση συγκεκριμένης γλώσσας προγραμματισμού. Το Android αντιλαμβάνεται για παράδειγμα τη Java ενώ το iOS την Objective-C.

Οι native εφαρμογές εγκαθίστανται στο κινητό είτε από τον ίδιο τον χρήστη μέσω του APK αρχείου που παράγεται, από το build της εφαρμογής, είτε προϋπάρχουν στο λειτουργικό σύστημα, είτε μπορεί να τις "κατεβάσει" ο χρήστης από το αντίστοιχο διαδικτυακό κατάστημα εφαρμογών. Η πλειοψηφία των εφαρμογών , και ιδίως τα παιχνίδια, στις έξυπνες κινητές συσκευές είναι native, αφού παρέχουν πολύ καλό επίπεδο γραφικών. Για την ανάπτυξη μιας native εφαρμογής, θα πρέπει να αναπτυχθεί ο απαραίτητος κώδικας στη γλώσσα που αντιλαμβάνεται η πλατφόρμα της συσκευής για την οποία προορίζεται η εφαρμογή. Επομένως βασικό μειονέκτημα των εφαρμογών που ανήκουν σε αυτήν την κατηγορία είναι ότι δεν υπάρχει η δυνατότητα να μεταφερθούν σε οποιοδήποτε λειτουργικό σύστημα.

Οι web εφαρμογές υλοποιούνται με τεχνολογίες δικτύου. Για τον λόγο αυτό είναι απαραίτητος κάποιος browser για την απεικόνισή τους. Παράδειγμα τέτοιων εφαρμογών αποτελούν οι εφαρμογές ηλεκτρονικού ταχυδρομείου. Βασικό πλεονέκτημα των εφαρμογών αυτών είναι ότι δεν προορίζονται για κάποιο συγκεκριμένο λειτουργικό σύστημα αλλά παρέχουν τη δυνατότητα εκτέλεσης σε διαφορετικά λειτουργικά συστήματα μέσω τεχνολογιών δικτύου. Οι εφαρμογές αυτές δεν εγκαθίστανται στη συσκευή του χρήστη, αλλά είναι προσβάσιμες κάθε φορά που επισκεπτόμαστε τη διεύθυνση της εφαρμογής στον browser. Σημαντικό μειονέκτημα των εφαρμογών τέτοιου τύπου είναι ότι δεν υπάρχει έλεγχος και προστασία των ευαίσθητων προσωπικών δεδομένων του χρήστη αφού εξαρτώνται πάντα από κάποιον browser. Η εξάρτησή τους αυτή αποτελεί επιπλέον αιτία για τη μειωμένη

απόδοσή τους καθώς δεν εξαρτώνται μόνο από τη συσκευή και την επεξεργαστική ισχύ της αλλά και από το δίκτυο.

Οι υβριδικές εφαρμογές αποτελούν κατά κάποιο τρόπο έναν συνδυασμό των native και web εφαρμογών. Χρησιμοποιούμε και εδώ διαδικτυακές τεχνολογίες (HTML, CSS, javascript) οι οποίες είναι κατανοητές από όλα τα λειτουργικά συστήματα, όπως και στις web εφαρμογές , έχοντας πρόσβαση στα χαρακτηριστικά της συσκευής όπως στις native εφαρμογές. Οι εφαρμογές αυτές μπορούν επίσης να εγκατασταθούν μέσω του apk αρχείου ή από το διαδικτυακό κατάστημα κάθε πλατφόρμας. Μπορούν να χρησιμοποιήσουν έτοιμα εργαλεία, όπως το Apache Cordova που αναλύεται στην επόμενη ενότητα και αποτελεί μια ανοιχτού κώδικα (open source) βιβλιοθήκη που παρέχει στους προγραμματιστές πρόσβαση στα native χαρακτηριστικά των κινητών συσκευών, όπως είναι η κάμερα ή η δόνηση, μέσα από ένα σύνολο APIs που μπορούν να κληθούν μέσω της Javascript.



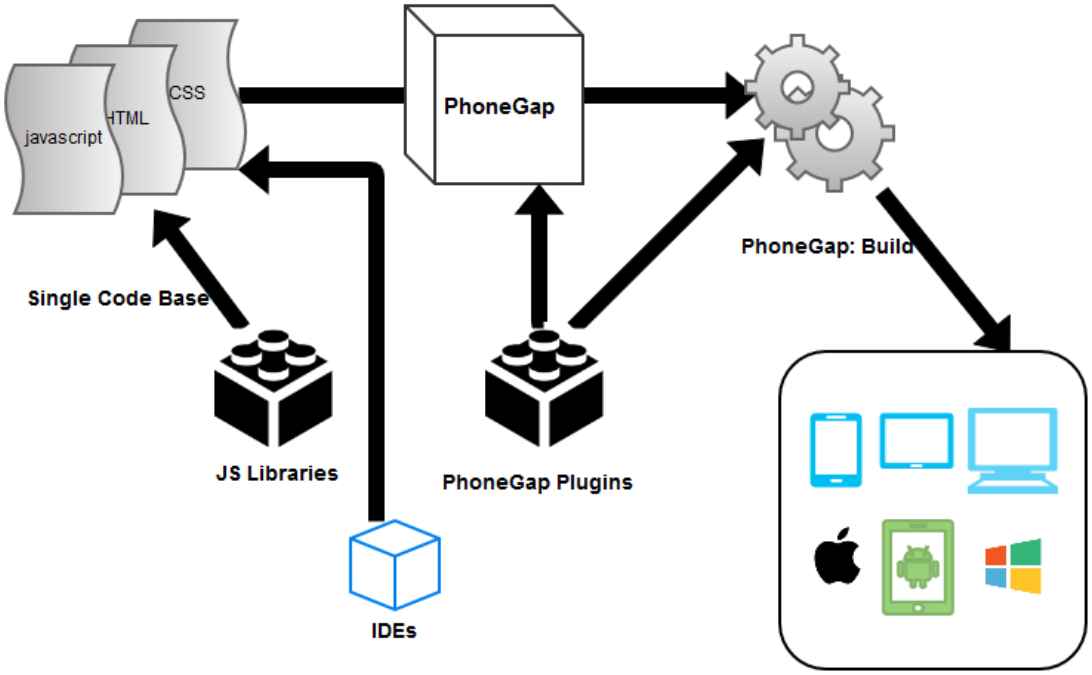
Εικόνα 17: Αρχιτεκτονική υβριδικής εφαρμογής (πηγή:Erik Paulsson-slideshare.net)

3.7 Apache Cordova ή PhoneGap

Η javascript και κυρίως η Angular, έχουν εισχωρήσει και στον τομέα της ανάπτυξης εφαρμογών για κινητές συσκευές. Έτσι έχουν δημιουργηθεί αρκετά frameworks για την ανάπτυξη λογισμικού σε κινητές συσκευές. Ένα από αυτά είναι το PhoneGap ή Apache Cordova όπως έχει πλέον ονομαστεί.

Το Apache Cordova παρέχει τη δυνατότητα δημιουργίας εφαρμογών για κινητές συσκευές με τη χρήση HTML5, CSS3 για την εμφάνιση και javascript για τη λογική τους, και όχι στις γλώσσες που υποστηρίζεται ανά συσκευή [9]. Για το λόγο αυτό, οι εφαρμογές που κατασκευάζονται με το framework του Apache Cordova είναι υβριδικές, καθώς δεν βασίζονται στα framework του λειτουργικού κάθε συσκευής, αλλά αποτελούν πλήρως web-based εφαρμογές. Τα λειτουργικά κινητών συσκευών όπως είναι το Android, το iOS και τα Windows, ακολουθούν το καθένα διαφορετικούς κανόνες και είναι συμβατά με διαφορετικές γλώσσες προγραμματισμού. Εδώ λοιπόν έρχεται να δώσει λύση το Apache Cordova καθώς μας παρέχει τη δυνατότητα να αναπτύξουμε εφαρμογές οι οποίες θα είναι συμβατές με όλες τις πλατφόρμες. Επιπλέον παρέχει ένα set απο APIs τα επιτρέπουν στον προγραμματιστή να έχει πρόσβαση σε native μεθόδους τις συσκευής όπως η κάμερα, μέσω της javascript.

Εικόνα 18: Λογότυπο του Apache Cordova



Εικόνα 19: Ανάπτυξη Mobile εφαρμογής με το PhoneGap Framework

3.8 Ionic Framework

Το Ionic Framework αποτελεί ένα software development kit (SDK) που έχει στόχο την ανάπτυξη υβριδικών εφαρμογών για έξυπνες κινητές συσκευές. Για το σκοπό αυτό χρησιμοποιεί αποκλειστικά διαδικτυακές τεχνολογίες όπως η HTML5, το CSS3 και η javascript. Συνεργάζεται με το Apache Cordova παρέχοντας τεχνολογίες για το front-end, επεκτείνοντας στην ουσία τις δυνατότητές του Apache Cordova δίνοντας έμφαση στον τρόπο αλληλεπίδρασης του χρήστη με την εφαρμογή και γενικότερα με το User Interface (UI). Μέσω του Ionic η διαδικασία υλοποίησης μιας εφαρμογής επιταχύνεται καθώς παρέχει έτοιμα πρότυπα εφαρμογών (templates), όπως είναι το blank, η tabs. Έχει αναπτυχθεί από τα πρώτα στάδια υλοποίησης του

Ionic μια πολύ καλή σχέση μεταξύ αυτού και της Angular JS γεγονός που έχει σαν αποτέλεσμα οι εντολές του ionic είναι απόλυτα κατανοητές από την Angular.

3.9 Bootstrap Framework

Το Bootstrap είναι ένα ισχυρό και καλαίσθητο front-end framework το οποίο επιταχύνει και διευκολύνει την ανάπτυξη web εφαρμογών. Βασίζεται στις διαδικτυακές τεχνολογίες HTML, CSS και JavaScript και υποστηρίζει όλους τους γνωστούς browsers στις περισσότερες εκδόσεις τους. Το Bootstrap αναπτύχθηκε από τους Mark Otto και Jacob Thornton για λογαριασμό του Twitter ως ένα framework που θα εξασφάλιζε την ενιαία αισθητική στις διάφορες λειτουργίες του. Ωστόσο τον Αύγουστο του 2011 εκδόθηκε σαν project ανοιχτού κώδικα. Είναι εύκολο στην εγκατάσταση και τη χρήση και παρέχει ένα σύνολο από web components και εικονίδια σε μορφή συμβολοσειράς (glyph icons). Μπορεί επιπλέον να επεκταθεί με τη συμβολή διαφόρων JavaScript βιβλιοθηκών που υποστηρίζει. Παρέχει ένα από τα πιο σταθερά και ευέλικτα συστήματα διάταξης (Grid) και προσαρμόζεται σε όλες τις διαστάσεις του browser.

3.10 Node.JS

Το Node.js αποτελεί μια πλατφόρμα ανάπτυξης λογισμικού η οποία έχει αναπτυχθεί σε περιβάλλον Javascript και έχει σαν στόχο να διευκολύνει τη δημιουργία διαδικτυακών εφαρμογών [15]. Ενώ τα περισσότερα σύγχρονα περιβάλλοντα ανάπτυξης διαδικτυακών εφαρμογών βασίζονται στην πολυνηματικότητα το *node* στηρίζεται στην ασύγχρονη επικοινωνία εισόδου/εξόδου και δίνει έμφαση στην ασύγχρονη επικοινωνία μεταξύ των υπολογιστικών πόρων η οποία επιτυγχάνεται με τη χρήση των callback συμβάντων (events) που παρέχει η JavaScript. Υπάρχει ένα μεγάλο σύνολο βιβλιοθηκών που είναι συμβατές με το *node*.

4 Λειτουργικότητα των εφαρμογών

4.1 Web εφαρμογή

Στην ενότητα αυτή θα περιγραφεί η λειτουργικότητα της διαδικτυακής εφαρμογής, που αναπτύχθηκε με σκοπό τον χρονοπρογραμματισμό των υποψηφίων διδακτόρων ώστε να καλυφθούν οι ανάγκες επιτήρησης των εξεταζόμενων μαθημάτων κατά τη διάρκεια των εξεταστικών περιόδων.

Ο αρχικός και μοναδικός χρήστης μέχρι την δημιουργία λογαριασμών νέων χρηστών είναι ο διαχειριστής της εφαρμογής, τα στοιχεία του οποίου έχουν προ-αποθηκευτεί στη βάση δεδομένων μας. Έστω η περίπτωση που ο διαχειριστής επιχειρεί να συνδεθεί στην εφαρμογή.

Η σύνδεση στην εφαρμογή είναι δυνατή μέσω της αρχικής σελίδας, η οποία φέρει μια φόρμα σύνδεσης (Log in) στην οποία υπάρχουν δυο πεδία. Πρόκειται για το πεδία που αφορούν όνομα χρήστη (username) και για τον κωδικό (password). Εισάγοντας σαν όνομα χρήστη το email που έχει αποθηκευτεί στη βάση για την συγκεκριμένο χρήστη καθώς και το κωδικό, πραγματοποιείται έλεγχος ώστε να επιβεβαιωθούν τα στοιχεία και ο ρόλος του χρήστη. Εάν δεν σωστά, ο διαχειριστής παραμένει στη σελίδα μέχρι τα στοιχεία που θα δώσει να επιβεβαιωθούν. Στην περίπτωση που τα στοιχεία του χρήστη δεν είναι λανθασμένα, τότε αυτός ανακατευθύνεται στην αρχική σελίδα του διαχειριστή.

Μέσω της αρχικής σελίδας αυτής, ο διαχειριστής της εφαρμογής έχει την δυνατότητα να αποσυνδεθεί από αυτήν, πατώντας στο αντίστοιχο "Log out" κουμπί. Επιπλέον μπορεί να περιηγηθεί στις υπόλοιπες σελίδες που είναι διαθέσιμες για τον δικό του ρόλο. Ας τις αναλύσουμε μια - μια.

Η πρώτη σελίδα που μπορεί να επιλέξει ο διαχειριστής αφορά τη διαχείριση όλων των χρηστών που συνδέονται στην εφαρμογή (*Registrations*) . Κάθε φορά που συνδέεται ένας νέος χρήστης στο σύστημα, ο διαχειριστής ειδοποιείται. Η ειδοποίηση αυτή πραγματοποιείται μέσω ενός notification bubble στο μενού περιήγησης δίπλα στην επιλογή των χρηστών. Ο διαχειριστής πρέπει να επιλέξει έναν ρόλο για τον χρήστη που επιχείρησε να συνδεθεί για πρώτη φορά. Αν επιλέξει για το χρήστη αυτό, το ρόλο του καθηγητή, τότε αποθηκεύεται στη βάση ένας νέος καθηγητής, αντίθετα αν στο νέο χρήστη ανατεθεί ο ρόλος του υποψήφιου διδακτορικού, τότε δημιουργείται ένας νέος υποψήφιος διδακτορικός. Το σύνολο των χρηστών

παρουσιάζονται στον διαχειριστή με τη μορφή πίνακα. Στη σελίδα αυτή ο διαχειριστής έχει τη δυνατότητα να αλλάξει ρόλο στον χρήστη αλλά και να τον διαγράψει.

Μια ακόμα σελίδα στην οποία μπορεί να μεταβεί ο διαχειριστής , είναι αυτή της διαχείρισης των καθηγητών (*Professors*).Εδώ, το σύνολο των καθηγητών παρουσιάζονται στον διαχειριστή με τη μορφή πίνακα και του παρέχεται η δυνατότητα να επεξεργαστεί τα στοιχεία του καθηγητή αλλά και να τον διαγράψει. Επίσης, είναι διαθέσιμοι για προβολή όλοι οι υποψήφιοι διδακτορικοί ανά καθηγητή, αλλά και τα μαθήματα τα οποία διδάσκει.

Στην σελίδα διαχείρισης των υποψηφίων διδακτόρων (*Phd Students*), ο διαχειριστής μπορεί δει συγκεντρωτικά σε πίνακα όλους τους υποψήφιους διδάκτορες του συστήματος, αλλά και να δημιουργήσει νέους. Επιπλέον, έχει τη δυνατότητα να επεξεργαστεί τα στοιχεία τους αλλά και να τους διαγράψει από το σύστημα. Τέλος, στη σελίδα αυτή φαίνονται τα μαθήματα στα οποία είναι υπεύθυνος ο κάθε υποψήφιος διδακτορικός, αλλά και το σύνολο των προσωπικών χρονικών περιορισμών που έχει θέσει αναφορικά με την εξεταστική περίοδο.

Το σύνολο των μαθημάτων συγκεντρώνεται στη σελίδα των μαθημάτων (*Lessons*). Εδώ ο διαχειριστής μπορεί να δημιουργήσει νέα μαθήματα και να τα αναθέσει σε κάποιον καθηγητή. Στην ουσία δημιουργεί τη σχέση καθηγητή - μαθήματος. Επιπλέον μπορεί να εισάγει τον αριθμό υποψηφίων διδακτόρων που απαιτείται σε κάθε μάθημα για την κάλυψη των αναγκών κατά τη διάρκεια των επιτηρήσεων των εξεταζόμενων μαθημάτων της εξεταστικής περιόδου. Δεν επιτρέπεται στο διαχειριστή ωστόσο, να εισάγει η να τροποποιήσει την ημερομηνία και τις ώρες εξέτασης των μαθημάτων. Αυτό, όπως θα δούμε στη συνέχεια, αποτελεί αρμοδιότητα του καθηγητή. Του παρέχεται βέβαια η δυνατότητα να επεξεργαστεί την περιγραφή του μαθήματος καθώς και τον καθηγητή στον οποίο αυτό έχει ανατεθεί.

Η σελίδα δημιουργίας και επεξεργασίας του προγράμματος αποτελεί την τελευταία επιλογή του διαχειριστή. Στη σελίδα αυτή παρουσιάζεται σε μορφή συμβάντων ημερολογίου το συνολικό πρόγραμμα επιτηρήσεων όλων των μαθημάτων. Κάθε συμβάν ισοδυναμεί με την επιτήρηση ενός υποψήφιου διδάκτορα. Πατώντας ένα κουμπί ("*Update Program*"), ο διαχειριστής μπορεί να δημιουργήσει ή να ανανεώσει το πρόγραμμα. Αξίζει να σημειωθεί ότι του παρέχεται η δυνατότητα να

ενημερώνεται για τον αν υπάρχουν αλλαγές οι οποίες θα μπορούσαν να επηρεάσουν το πρόγραμμα. Αν για παράδειγμα ένας υποψήφιος διδάκτορας, προσθέσει , αφαιρέσει η τροποποιήσει κάποιον περιορισμό, ο διαχειριστής ειδοποιείται με νέο bubble notification το οποίο του υποδεικνύει ότι πρέπει να ανανεώσει το πρόγραμμα των επιτηρήσεων. Στο ημερολόγιο με τις επιτηρήσεις, ο διαχειριστής επιτρέπεται να μετακινήσει κάποιον υποψήφιο διδακτορικό σε κάποιο άλλο μάθημα, τηρώντας βέβαια κάποια κριτήρια. Για παράδειγμα, δεν μπορεί να τον μετακινήσει σε μάθημα που ήδη επιτηρεί ούτε σε ημερομηνία που δεν εξετάζεται κανένα μάθημα. Οι χρωματικές υποδείξεις των επιτηρήσεων επάνω στο ημερολόγιο, βοηθούν επίσης τον διαχειριστή να καταλάβει εάν σε κάποιο μάθημα υπάρχουν περισσότεροι υποψήφιοι από όσοι χρειάζονται, αλλά αντίστοιχα και να ενημερωθεί εάν υπάρχει έλλειψη επιτηρητών. Για παράδειγμα εάν υπάρχει έλλειψη υποψηφίων το συμβάν στο ημερολόγιο φέρει κόκκινο χρώμα. Αντίθετα, αν οι επιτηρητές είναι υπεράριθμοι, ο χρωματισμός είναι πορτοκαλί, ή πράσινος στην περίπτωση που οι επιτηρητές είναι όσοι έχει δηλώσει ο διαχειριστής ότι απαιτούνται για το συγκεκριμένο μάθημα. Στο σημείο αυτό αξίζει να σημειωθεί πως με κάθε αλλαγή στο πρόγραμμα, ο καθηγητής που διδάσκει το μάθημα στο οποίο έγινε η αλλαγή αλλά και οι υποψήφιοι διδακτορικοί που τους αφορά η επιτήρηση που τροποποιήθηκε, ενημερώνονται με ανάλογο μήνυμα στο email τους (το οποίο χρησιμοποιούν ως όνομα χρήστη για να συνδεθούν στην εφαρμογή).

Ας προχωρήσουμε στην περιγραφή της λειτουργικότητας της διαδικτυακής εφαρμογής, για τον ρόλο του καθηγητή. Εάν ένας καθηγητής επιχειρήσει να συνδεθεί για πρώτη φορά, πρέπει να εισάγει τα προσωπικά του στοιχεία. Μεταξύ αυτών, είναι το email τους το οποίο θα χρησιμοποιούν σαν όνομα χρήστη (username) για να συνδεθούν στο σύστημα. Μετά την δημιουργία λογαριασμού στο σύστημα, και αφού ο διαχειριστής αναθέσει στο νέο χρήστη κάποιο ρόλο, στην περίπτωση αυτή το ρόλο του καθηγητή, ο χρήστης ειδοποιείται στο email που δήλωσε κατά τη δημιουργία του λογαριασμού του, ότι ο λογαριασμός του έχει ενεργοποιηθεί. Μπορεί πλέον να συνδεθεί να το ρόλο του καθηγητή. Αφού εισάγει στη φόρμα σύνδεσης σωστά τα στοιχεία που δήλωσε κατά την εγγραφή του στο σύστημα, ο καθηγητής ανακατευθύνεται στην αρχική σελίδα που είναι διαθέσιμη για τον δικό του ρόλο.

Από τη σελίδα αυτή μπορεί να μεταβεί στις υπόλοιπες υποσελίδες. Πρόκειται για την σελίδα διαχείρισης των υποψήφιων διδακτορικών που επιβλέπει, την σελίδα διαχείρισης των μαθημάτων του αλλά και τη σελίδα που αφορά το πρόγραμμα των επιτηρήσεων.

Στη σελίδα διαχείρισης των υποψήφιων διδακτορικών που επιβλέπει, μπορεί να δει συγκεντρωτικά σε πίνακα όλους τους υποψηφίους καθώς και τα μαθήματα που τους έχουν ανατεθεί. Από όλους τους υποψηφίους έχει τη δυνατότητα να επιλέξει αυτούς που ο ίδιος επιτηρεί και να αποθηκεύσει τις επιλογές του.

Για να δει τα μαθήματά που διδάσκει ο ίδιος, πρέπει να προχωρήσει στην αντίστοιχη σελίδα των μαθημάτων όπου μπορεί να τα δει συγκεντρωτικά σε μορφή πίνακα. Έχει επιπλέον τη δυνατότητα να επιλέξει τους υποψήφιους διδακτορικούς που θα τον βοηθήσουν στο κάθε μάθημα. Στη λίστα των διδακτορικών προς επιλογή, εμφανίζονται μόνο αυτοί που ο ίδιος επέλεξε στην σελίδα των υποψηφίων.

Τέλος, στη σελίδα που αφορά το πρόγραμμα, οι καθηγητές βλέπουν τις επιτηρήσεις των μαθημάτων που οι ίδιοι διδάσκουν και όχι όλων όπως μπορεί ο διαχειριστής. Το πρόγραμμα των επιτηρήσεων εμφανίζεται και εδώ με μορφή ημερολογίου και κάθε μάθημα αποτελεί ένα ημερολογιακό συμβάν. Για να εμφανιστεί η ώρα εξέτασης του μαθήματος αλλά και οι επιτηρητές του, αρκεί να περάσουμε το ποντίκι επάνω από το συμβάν-μάθημα που έχει δημιουργηθεί στο ημερολόγιο. Ο καθηγητής δεν έχει τη δυνατότητα να προχωρήσει σε οποιαδήποτε μετατροπή του προγράμματος. Ο μόνος τρόπος με τον οποίο μπορεί να το κάνει είναι η μετατροπή της ημερομηνίας και ώρας εξέτασης του μαθήματος από τη σελίδα των μαθημάτων που του παρέχεται.

Οι δυνατότητες που παρέχονται στους χρήστες που έχουν το ρόλο του υποψήφιου διδακτορικού διαφέρουν. Η διαδικασία δημιουργίας και ενεργοποίησης λογαριασμού ωστόσο είναι όμοια με αυτή του καθηγητή. Μετά την δημιουργία λογαριασμού στο σύστημα, και αφού του ανατεθεί ο ρόλος του υποψήφιου διδακτορικού από τον διαχειριστή, ο χρήστης ειδοποιείται με email, ότι ο λογαριασμός του έχει ενεργοποιηθεί και μπορεί πλέον να συνδεθεί να το ρόλο του υποψήφιου διδακτορικού. Αφού εισάγει στη φόρμα σύνδεσης σωστά τα στοιχεία που δήλωσε κατά την εγγραφή του στο σύστημα, ο διδακτορικός ανακατευθύνεται στην αρχική σελίδα που είναι διαθέσιμη για τον ρόλο του.

Από την αρχική σελίδα αυτή μπορεί να μεταβεί είτε στην σελίδα διαχείρισης των προσωπικών του περιορισμών, είτε σε αυτή του προγράμματος. Στην πρώτη έχει τη δυνατότητα να δημιουργήσει, να επεξεργαστεί αλλά και να διαγράψει περιορισμούς. Κατά τη δημιουργία των περιορισμών του , ο υποψήφιος καλείται να δώσει ημερομηνία και ώρα έναρξης και λήξης του περιορισμού, αλλά και κάποια περιγραφή για τον περιορισμό αυτό. Επιλέγοντας την σελίδα του προγράμματος, του παρέχεται η δυνατότητα να δει συγκεντρωτικά τα μαθήματα που ο ίδιος επιτηρεί και ποιες ώρες. Δεν μπορεί ωστόσο να παρέμβει άμεσα με οποιονδήποτε τρόπο και να μετατρέψει το πρόγραμμα. Όπως προαναφέρθηκε, το δικαίωμα αυτό το έχει μόνο ο διαχειριστής του συστήματος. Βέβαια τροποποιώντας κάποιον προσωπικό του περιορισμό, "αναγκάζει" τον διαχειριστή να ανανεώσει το πρόγραμμα, αφού όπως είπαμε για οποιαδήποτε μετατροπή σε μέρα και ώρα εξέτασης μαθήματος, ή περιορισμού υποψήφιου διδακτορικό, ο διαχειριστής ενημερώνεται.

4.2 Mobile εφαρμογή

Στην ενότητα αυτή θα περιγραφεί η λειτουργικότητα της εφαρμογής για κινητές συσκευές, που αναπτύχθηκε με σκοπό την δήλωση της παρουσίας των υποψήφιων διδακτορικών στις επιτηρήσεις που τους έχουν αναταθεί, αλλά και την προβολή του προγράμματος των επιτηρήσεων συγκεντρωτικά.

Συγκεκριμένα, στην mobile εφαρμογή μπορούν να συνδεθούν μόνο οι λογαριασμοί των καθηγητών και των υποψήφιων διδακτορικών που δημιουργήθηκαν μέσω της διαδικτυακής εφαρμογής. Εάν η τα στοιχεία σύνδεσης που εισάγει ο χρήστης είναι λανθασμένα, τότε ειδοποιείται με ανάλογο μήνυμα και καλείται να ξαναπροσπαθήσει.

Για τον χρήστη - καθηγητή, μια από τις διαθέσιμες επιλογές είναι η προβολή σε μορφή λίστας όλων των επιτηρήσεων των μαθημάτων που διδάσκει, αλλά και των επιτηρητών που έχουν οριστεί για κάθε μάθημα. Δίπλα από κάθε επιτηρητή μπορεί να δει εάν αυτός παρουσιάστηκε στην επιτήρηση που ανατέθηκε ή όχι, με ένα ενδεικτικό

σύμβολο (ν και x αντίστοιχα). Το πρόγραμμα των επιτηρήσεων είναι διαθέσιμο κάτω από το tab *Program*.

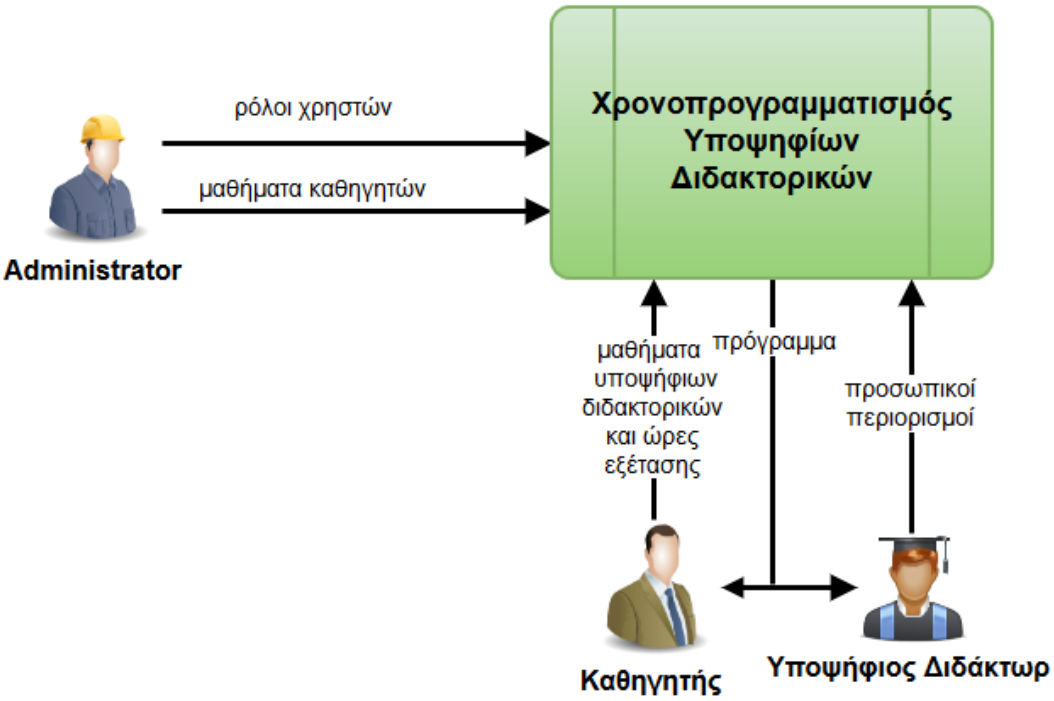
Για να μπορέσει να δηλώσει την παρουσία του ένας επιτηρητής στο μάθημα που του ανατέθηκε, θα πρέπει ο καθηγητής να έχει δημιουργήσει έναν κωδικό τύπου QR τον οποίο και θα διαθέσει προς "σκανάρισμα" από τους επιτηρητές. Ο καθηγητής, λοιπόν εκτός από το να δει το πρόγραμμα επιτηρήσεων των μαθημάτων του, έχει και τη δυνατότητα μέσω της mobile εφαρμογής να κατασκευάσει έναν QR κωδικό για κάθε μάθημα, επιλέγοντας το αντίστοιχο tab (*QR codes*).

Εκτός από τα δυο βασικά tabs που προαναφέρθηκαν (*Program* και *QR codes*), διατίθεται στον χρήστη - καθηγητή, και ένα μικρότερο float μενού, μέσω του οποίου μπορεί να πραγματοποιήσει την έξοδό του από την εφαρμογή (log out) αλλά και να μεταβεί στη σελίδα της web εφαρμογής.

Αντίστοιχες επιλογές παρέχονται και στον χρήστη - υποψήφιο διδακτορικό. Επιλέγοντας το tab *Program* μπορεί να δει το πρόγραμμα των επιτηρήσεών του σε μορφή λίστας, ενώ επιλέγοντας το tab *QR codes,* βλέπει όλους τους κωδικούς QR που έχει σκανάρει. Για κάθε σκαναριμένο κωδικό μπορεί είτε να επιλέξει με ποιο μάθημα θα τον αντιστοιχίσει, είτε να τον διαγράψει. Για να σκανάρει έναν κωδικό, αρκεί να επιλέξει την επιλογή *Scan* από το float menu. Επιλέγοντας να σκανάρει έναν κωδικό, ανοίγει η κάμερα του κινητού, σκανάρει και αποθηκεύει το κρυφό περιεχόμενο του κωδικού. Μέσω του ίδιου μενού μπορεί και αυτός, όπως και ο καθηγητής να πραγματοποιήσει έξοδο από το σύστημα ή να μεταβεί στη σελίδα της web εφαρμογής.

5 Ροή (Flow)

5.1 Ροή (Flow) των δεδομένων στη web εφαρμογή



Εικόνα 20: Flow of data στη web εφαρμογή

Η βασική λειτουργία της διαδικτυακής εφαρμογής, που αναπτύχθηκε με σκοπό τον χρονοπρογραμματισμό των υποψηφίων διδακτόρων ώστε να καλυφθούν οι ανάγκες επιτήρησης των εξεταζόμενων μαθημάτων κατά τη διάρκεια των εξεταστικών περιόδων, είναι η παραγωγή του προγράμματος των επιτηρήσεων δεδομένων των ωρών εξέτασης των μαθημάτων και των προσωπικών περιορισμών των υποψηφίων διδακτόρων.

Ο διαχειριστής τις σελίδας αναθέτει ρόλους στους χρήστες που συνδέονται στο σύστημα για πρώτη φορά και δημιουργεί και αναθέτει στους αντίστοιχους καθηγητές τα εξεταζόμενα μαθήματα για τα οποία πρέπει να παραχθεί το πρόγραμμα των επιτηρήσεων. Επιπλέον ορίζει το πλήθος των υποψήφιων διδακτόρων που απαιτούνται για το κάθε μάθημα.

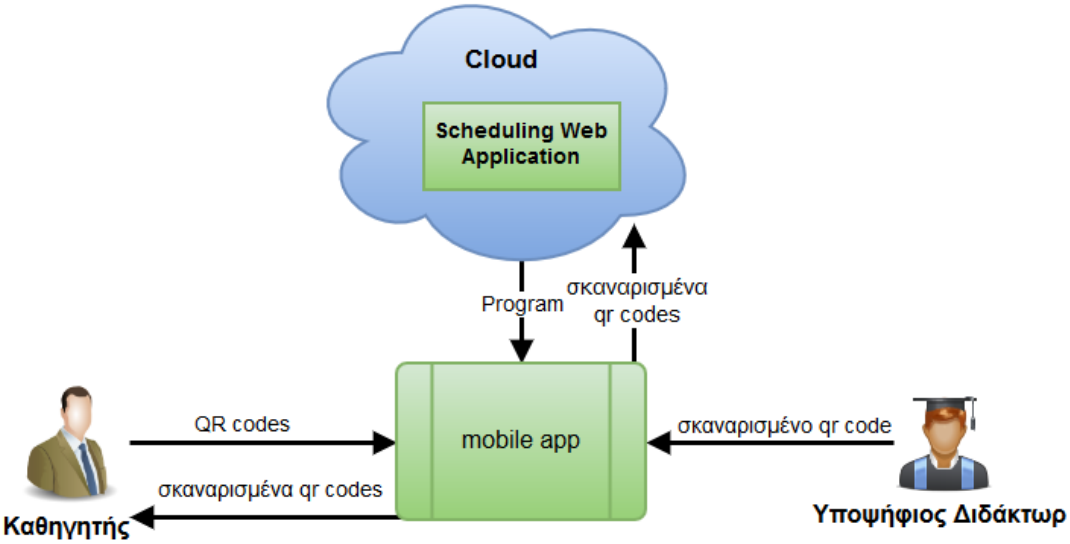
Με τη σειρά τους οι καθηγητές συνδέονται στο σύστημα και αναθέτουν τα μαθήματά τους στους υποψήφιους διδακτορικούς που επιβλέπουν. Επιπλέον εισάγουν τις ώρες εξέτασης για κάθε μάθημα.

Οι υποψήφιοι διδακτορικοί πρέπει να εισάγουν στο σύστημα, εάν έχουν, τις χρονικές περιόδους κατά τις οποίες δεν μπορούν να παραστούν σε κάποια εξέταση. Οι περιορισμοί αυτοί λαμβάνονται υπόψη από την εφαρμογή, μαζί με τα δεδομένα που έχουν εισάγει οι άλλοι ρόλοι (ο διαχειριστής τον αριθμό επιτηρητών ανά μάθημα και οι καθηγητές τις ώρες εξέτασης των μαθημάτων και τους υποψήφιους διδακτορικούς κάθε μαθήματος) η οποία τελικά παράγει το τελικό πρόγραμμα, το οποίο είναι διαθέσιμο στους χρήστες. Η διαδικασία που περιγράφηκε παρουσιάζεται στην εικόνα 20.

5.2 Ροή (Flow) των δεδομένων στη mobile εφαρμογή

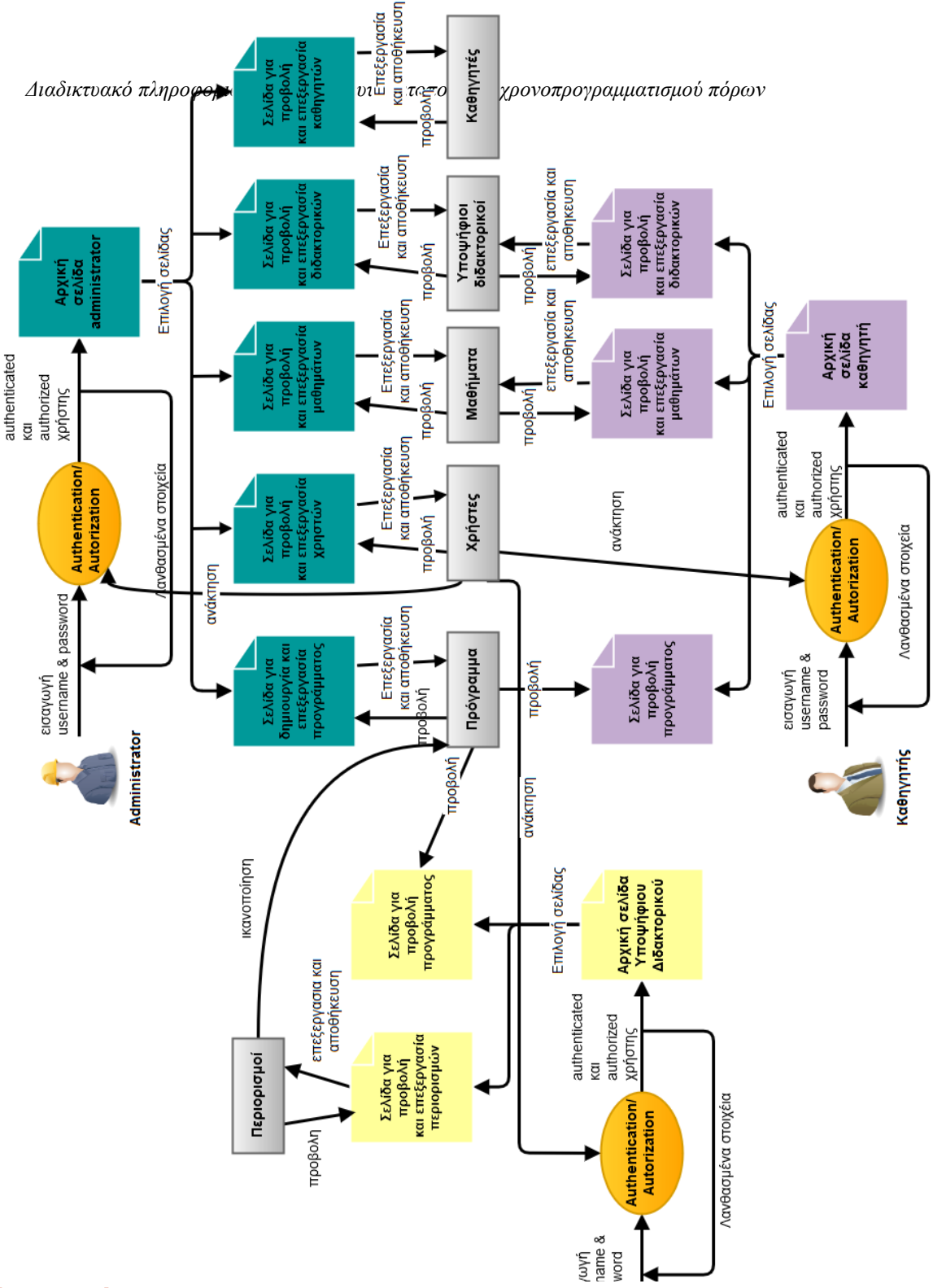
Κύριος σκοπός της εφαρμογής που αναπτύχθηκε για κινητές συσκευές, είναι η προβολή του προγράμματος στους χρήστες (καθηγητές και υποψηφίους διδακτορικούς) αλλά και η δήλωση παρουσίας των υποψηφίων διδακτορικών στις επιτηρήσεις μαθημάτων που τους έχουν ανατεθεί. Επιπλέον οι καθηγητές μπορούν να δουν ποιοι από τους υποψήφιους διδακτορικούς όντως παραβρέθηκαν στην επιτήρηση που τους είχε ανατεθεί. Η παραπάνω λειτουργία επιτυγχάνεται με τη βοήθεια των qr codes.

Οι καθηγητές μπορούν να δημιουργήσουν, μέσω της mobile εφαρμογής που αναπτύχθηκε, qr codes τα οποία στη συνέχεια μπορούν να σκανάρουν οι υποψήφιοι διδακτορικοί κατά της προσέλευσή τους στην εξέταση του μαθήματος που τους έχει ανατεθεί να επιτηρήσουν, δηλώνοντας έτσι την παρουσία τους. Οι σκαναρισμένοι κωδικοί είναι έπειτα διαθέσιμοι στους καθηγητές οι οποίοι με την τρόπο αυτό είναι σε θέση να γνωρίζουν ποιοι υποψήφιοι παραβρέθηκαν στις εξετάσεις και ποιοι όχι. Στην εικόνα 21 παρουσιάζεται αυτή η ροή των δεδομένων στην mobile εφαρμογή.



Εικόνα 21: Flow of data στη mobile εφαρμογή

5.3 Flow chart της web εφαρμογής



Εικόνα 22: Flow chart της web εφαρμογής

Στην εικόνα 22 παρουσιάζεται διαγραμματικά η ροή των διαδικασιών που ακολουθούνται στην διαδικτυακή εφαρμογή. Όπως φαίνεται όλοι οι χρήστες αφού εισάγουν τα στοιχεία τους για να εισαχθούν στο σύστημα, γίνεται επιβεβαίωση των στοιχείων των και οδηγούνται στην αντίστοιχη αρχική σελίδα με βάση το ρόλο τους.

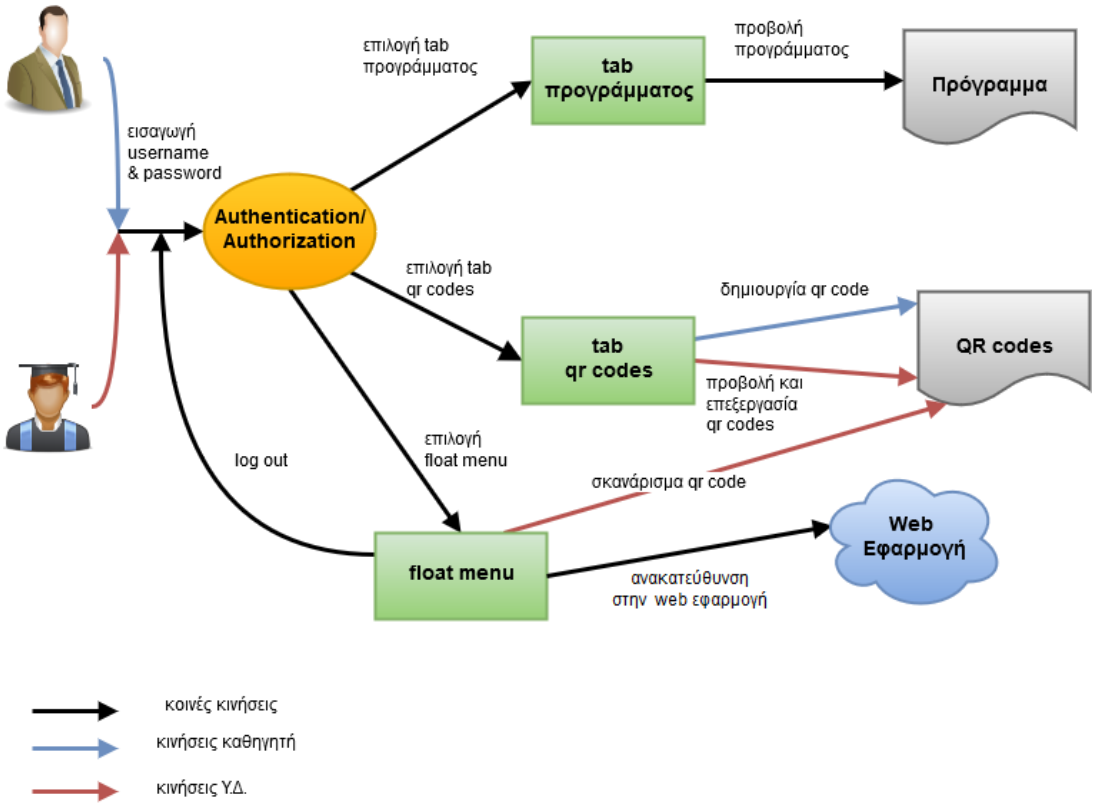
Αν πρόκειται για τον διαχειριστή της σελίδας, αυτός έχει τη δυνατότητα να περιηγηθεί στη συνέχεια είτε στη σελίδα που περιέχει όλους τους καθηγητές, είτε σε αυτή των υποψήφιων διδακτορικών, είτε στην σελίδα που αφορά τα μαθήματα είτε σε αυτή των αιτήσεων νέων χρηστών. Επίσης έχει τη δυνατότητα να μεταβεί στη σελίδα παραγωγής και επεξεργασίας του προγράμματος. Στις σελίδες των καθηγητών και των υποψήφιων διδακτόρων ο διαχειριστής μπορεί να επεξεργαστεί, ακόμα και να διαγράψει κάποιον καθηγητή ή υποψήφιο. Αντίστοιχα στη σελίδα των μαθημάτων έχει τη δυνατότητα να δημιουργήσει, να επεξεργαστεί αλλά και να διαγράψει μαθήματα. Επιπλέον στη σελίδα με τις αιτήσεις νέων χρηστών βλέπει συγκεντρωτικά όλους τους χρήστες που έχουν συνδεθεί στο σύστημα και τους αναθέτει ρόλους (Καθηγητής, Υποψήφιος Διδακτορικός). Τέλος, στη σελίδα του προγράμματος μπορεί να δημιουργήσει αλλά και να επεξεργαστεί το παραχθέν πρόγραμμα.

Από την άλλη, εάν ο χρήστης που συνδεθεί στο σύστημα, είναι καθηγητής, έχει πρόσβαση στη σελίδα όπου μπορεί να δει συγκεντρωτικά τα μαθήματά του, αλλά και να τα επεξεργαστεί ορίζοντάς τους ημερομηνία εξέτασης και υποψήφιο διδακτορικό. Μπορεί επίσης, να μεταβεί στη σελίδα των υποψηφίων, στην οποία επιλέγει όλους τους υποψήφιους διδακτορικούς που επιβλέπει. Μπορεί ακόμα να δει το πρόγραμμα επιτηρήσεων των μαθημάτων που διδάσκει.

Υπάρχει, τέλος, η πιθανότητα ο χρήστης που θα συνδεθεί να έχει το ρόλο του υποψηφίου διδακτορικού, ο οποίος μπορεί να εισαγάγει στο σύστημα ,στην αντίστοιχη σελίδα, όλους τους χρονικούς περιορισμούς που τον αφορούν αλλά και να δει συγκεντρωτικά το πρόγραμμα όλων των επιτηρήσεών του.

Από όλες τις προαναφερθείσες σελίδες, οι χρήστες μπορούν να πραγματοποιήσουν έξοδο από την εφαρμογή (log out).

5.4 Flow Chart της mobile εφαρμογής



Εικόνα 23: Flow chart της mobile εφαρμογής

Οι χρήστες που συνδέονται, στην mobile εφαρμογή που αναπτύχθηκε ,έχουν και διαφορετικές επιλογές ανάλογα με το ρόλο τους. Οι καθηγητές μετά την επιτυχή είσοδό τους στο σύστημα έχουν την επιλογή να δουν το πρόγραμμα των μαθημάτων τους, αλλά και τους επιτηρητές που τα επιβλέπουν. Επίσης μπορούν να δημιουργήσουν κωδικούς τύπου QR οι οποίοι στη συνέχεια σκανάρονται από τους υποψηφίους που επιτηρούν τα μαθήματά του ώστε να δηλώσουν την παρουσία τους. Οι χρήστες που έχουν το ρόλο του υποψήφιου διδακτορικού μπορούν να δουν τα μαθήματα και τι ώρες που επιτηρούν αλλά και να σκανάρουν τους QR κωδικούς που έχει δημιουργήσει ο καθηγητής μέσω της εφαρμογής. Επιπλέον μπορούν να δουν όλους τους κωδικούς που έχουν σκανάρει και να τους αντιστοιχήσουν με τα μαθήματα που επιθυμούν. Τέλος είναι δυνατή η ανακατεύθυνση όλων των χρηστών στην web εφαρμογή αλλά και η έξοδος από την mobile εφαρμογή.

6 Υλοποίηση εφαρμογών

6.1 Web εφαρμογή

Η εφαρμογή χτίστηκε επάνω στο εργαλείο maven το οποίο όπως αναφέρθηκε και στο κεφάλαιο 3.2 επιτρέπει την οργάνωση και αυτοματοποίηση της μετάφρασης του κώδικα αλλά και της δημιουργίας των εκτελέσιμων αρχείων. Επιπλέον προσφέρει αυτόματη εύρεση και ανάκτηση όλων των προγραμματιστικών εξαρτήσεων από εξωτερικές βιβλιοθήκες και τεχνολογίες. Όλα αυτά μέσω του αρχείου pom.xml το οποίο καθοδηγεί το Maven για το πως πρέπει να χτιστεί το project αλλά και να εκτελέσει όλες τις παραπάνω λειτουργίες.

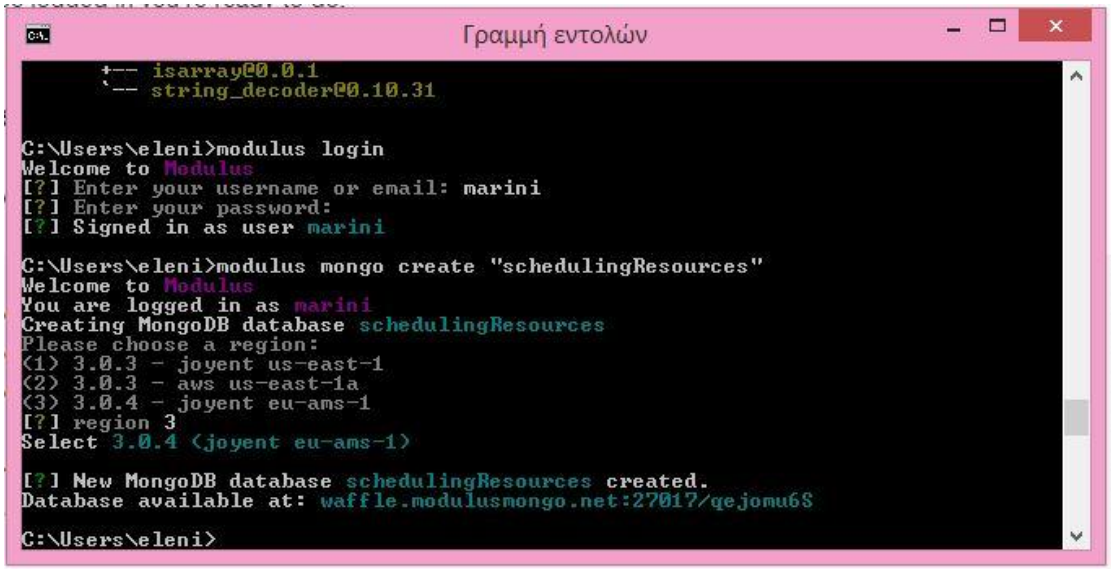
Επομένως στο *pom.xml* δηλώθηκαν όλες οι εξωτερικές βιβλιοθήκες και τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής. Αρχικά δηλώνεται η έκδοση του java development kit που χρησιμοποιείται, κατά τη μεταγλώττιση του κώδικά από το maven. Επίσης το *Spring Framework* και διάφορες βοηθητικές βιβλιοθήκες του, όπως αυτή του *spring security*, η οποία προσφέρει την ασφάλεια που απαιτείται σε μια διαδικτυακή εφαρμογή, έπρεπε να δηλωθεί στο αρχείο pom.xml για να είναι δυνατή η χρήση του, καθώς η αρχιτεκτονική του συστήματος βασίζεται στην MVC αρχιτεκτονική που προσφέρει το Spring. Ενδεικτικά να αναφέρουμε ακόμα, την βιβλιοθήκη *javax.mail* η οποία χρησιμοποιήθηκε για την αποστολή email σε διάφορα σημεία της εφαρμογής. Επιπλέον για τη χρήση του spring θα πρέπει να δηλωθεί στο web.xml της εφαρμογής, το οποίο πρέπει να υλοποιεί κάθε διαδικτυακή εφαρμογή, το αρχείο που αντιπροσωπεύει τον DispatcherServlet του Spring. Η δήλωση αυτή γίνεται ως εξής:

```
<servlet>
	<servlet-name>mvc-dispatcher</servlet-name>
	<servlet-class>
			org.springframework.web.servlet.DispatcherServlet
```

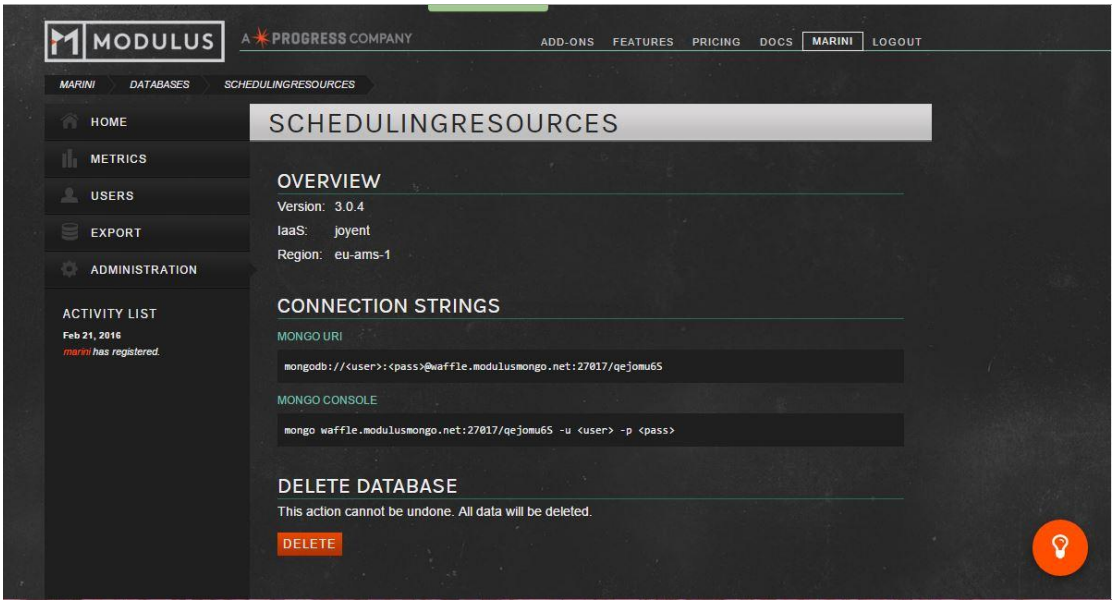
```
		</servlet-class>
		<load-on-startup>1</load-on-startup>
	</servlet>

	<servlet-mapping>
		<servlet-name>mvc-dispatcher</servlet-name>
		<url-pattern>/</url-pattern>
	</servlet-mapping>


	<context-param>
		<param-name>contextConfigLocation</param-name>
		<param-value>/WEB-INF/mvc-dispatcher-servlet.xml</param-value>
	</context-param>
```

Η πρώτη και βασική κίνηση ενός προγραμματιστή πριν αναπτυχθεί μια εφαρμογή, διαδικτυακή ή μη, είναι ο σχεδιασμός και η δημιουργία μιας βάσης δεδομένων στην οποία θα φιλοξενούνται τα δεδομένα της εφαρμογής. Για να γίνει αυτό πρέπει αρχικά να αποφασιστεί ο τύπος της βάσης, ανάλογα με τον οποίο σχεδιάζεται και η μορφή της. Ο τύπος της βάσης δεδομένων που επιλέχθηκε για την διαδικτυακή εφαρμογή που αναπτύχθηκε στα πλαίσια της εργασίας αυτής είναι NoSQL (Not Only SQL) και συγκεκριμένα η MongoDB. Όπως αναφέρθηκε στο κεφάλαιο 3.1, οι βάσεις δεδομένων τέτοιου τύπου είναι εγγραφοκεντρικές, δηλαδή αντί για εγγραφές, περιέχουν έγγραφα και αντί για πίνακες, συλλογές. Ο στόχος ήταν, οι δυο εφαρμογές να χρησιμοποιούν την ίδια βάση. Η φιλοξενία της λοιπόν στο νέφος (Cloud) ήταν μια επιλογή που εξυπηρετούσε το σκοπό αυτό. Κατασκευάστηκε λοιπόν μια βάση δεδομένων τεχνολογίας MongoDB την οποία φιλοξενεί το Modulus.

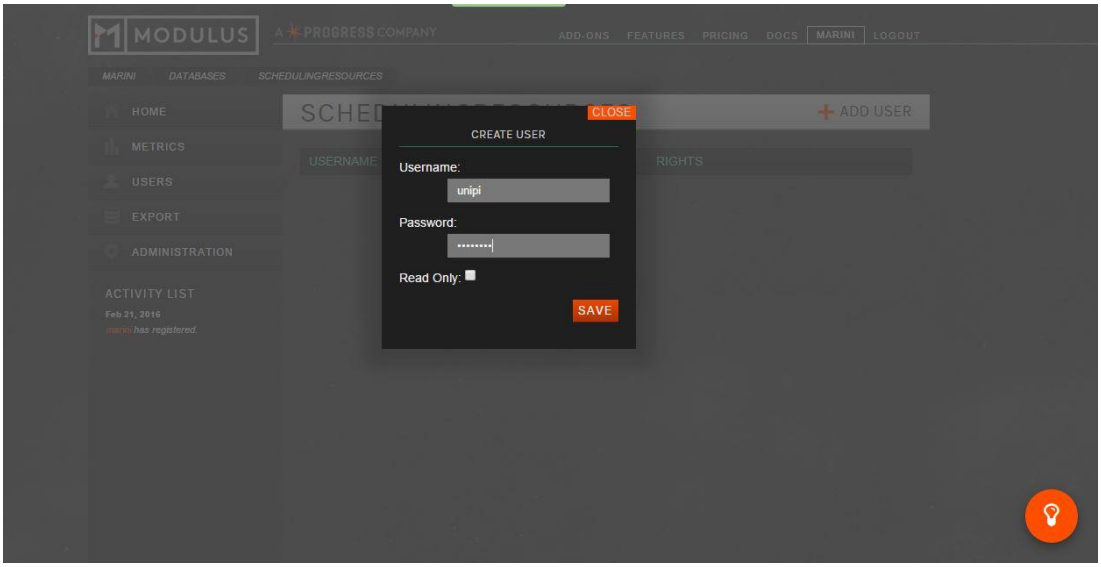
Μετά τη δημιουργία λογαριασμού στο Modulus, εκτελέστηκαν συγκεκριμένες εντολές μέσω της γραμμής εντολών (cmd) για τη δημιουργία της βάσης. Αρχικά πραγματοποιήθηκε η σύνδεση του λογαριασμού που κατασκευάστηκε για το σκοπό αυτό με την εντολή `modulus login`. Μετά την επιτυχημένη σύνδεση, η εκτέλεση την εντολής `modulus mongo create "schedulingResources"`, δημιουργείται η βάση μας και το Modulus μας ενημερώνει για το που φιλοξενείται αυτή. Η βάση που έχει δημιουργηθεί ονομάζεται schedulingResources και είναι διαθέσιμη στον server `waffle.modulusmongo.net:27017/qejomu6S`.



Εικόνα 24: Δημιουργία βάσης δεδομένων



Εικόνα 25: Η βάση στο modulus



Εικόνα 26: Δημιουργία χρήστη για σύνδεση με τη βάση

Με κάποιον τρόπο πρέπει να γνωρίζει η εφαρμογή, ότι ο τύπος βάσης δεδομένων που θα χρησιμοποιήσουμε θα είναι MognoDB. Αφού η αρχιτεκτονική μας βασίζεται στο spring αρκεί να δηλώσουμε στο pom.xml την απαραίτητη βιβλιοθήκη του spring που είναι υπεύθυνη για την διαχείριση των δεδομένων μας έτσι ώστε να

συμπεριληφθεί αυτόματα ο συμβατός MongoDB java driver κατά τη μεταγλώττιση της εφαρμογής. Δηλώνουμε επομένως στο pom.xml

```
<dependency>
	<groupId>org.springframework.data</groupId>
	<artifactId>spring-data-mongodb</artifactId>
	<version>${spring.data.mongo.version}</version>
</dependency>
```

όπου spring.data.mongo.version η έκδοση της βιβλιοθήκης που επιθυμούμε να χρησιμοποιήσουμε. Το βασικό πλεονέκτημα της βιβλιοθήκης Spring Data MongoDB είναι ότι δε χρειάζεται να ανησυχούμε για την μετατροπή ενός java bean σε αντικείμενο Mongo DBObject, και το αντίστροφο

Για να μπορέσουμε να συνδεθούμε μέσω της εφαρμογής με τη βάση που δημιουργήσαμε, πρέπει να δηλώσουμε σε beans στο αρχείο mvc-dispatcher-servlet.xml του spring τα στοιχεία της βάσης και τον χρήστη που ορίσαμε. Το όνομα της βάσης μας όπως προκύπτει από το url που μας επέστρεψε το modulus είναι ***qejomu6S***, και τα στοιχεία του χρήστη είναι username: ***unipi*** και password: ***123456q!***. Το bean που δημιουργείται και μας επιτρέπει τη σύνδεση με τη βάση είναι της κλάσης org.springframework.data.mongodb.core.MongoTemplate.

Ενδεικτικά η δήλωση των απαραίτητων beans για τη βάση γίνεται ως εξής:

```
<bean id="mongoTemplate"
class="org.springframework.data.mongodb.core.MongoTemplate">
	<constructor-arg name="mongo" ref="mongo"/>
	<constructor-arg name="databaseName" value="qejomu6S"/>
	<constructor-arg name="userCredentials"
	ref="mongoCredentials"/>
</bean>

<bean id="mongoCredentials"
class="org.springframework.data.authentication.UserCredentials">
	<constructor-arg name="username" value="unipi" />
	<constructor-arg name="password" value="123456q!" />
</bean>

<bean class="com.mongodb.MongoURI" id="mongoURI">
	<constructor-arg
value="mongodb://unipi:123456q!@waffle.modulusmongo.net:27017/qejomu6
S"  />
</bean>

<bean class="com.mongodb.Mongo" id="mongo">
	<constructor-arg ref="mongoURI" />
</bean>
```

Το bean αυτό της κλάσης org.springframework.data.mongodb.core. MongoTemplate, το χρησιμοποιούν όλες οι κλάσεις που θέλουν να εκτελέσουν κάποιο query στη βάση. Για να μπορέσουν να το χρησιμοποιήσουν αρκεί να το δηλώσουν με annotation Autowire αλλά και να ορίσουν μεθόδους get και set για το bean αυτό.

```
@Autowired
@Qualifier("mongoTemplate")
MongoTemplate mongoTemplate;

public MongoTemplate getMongoTemplate() {
	return mongoTemplate;
}

public void setMongoTemplate(MongoTemplate mongoTemplate) {
	this.mongoTemplate = mongoTemplate;
}
```

Οι βασικές οντότητες που ορίστηκαν στο μοντέλο της εφαρμογής είναι:

- **Professor** (Καθηγητής): με πεδία για όνομα(fname), επίθετο (lname), email, τηλέφωνο (phone)και ένα μοναδικό id.
- **Lesson** (Μάθημα): με πεδία για περιγραφή (description), αριθμό απαιτούμενων επιτηρητών, ημερομηνία και ώρα έναρξης και λήξης της εξέτασης, καθηγητή (Professor) και ένα μοναδικό id.
- **Phd Student** (Υποψήφιος Διδάκτορας): με πεδία για όνομα(fname), επίθετο (lname), email, τηλέφωνο (phone), ένα σύνολο από μαθήματα (Lesson), έναν καθηγητή (Professor)καθώς και ένα μοναδικό id.
- **Request** : με πεδία για όνομα(fname), επίθετο (lname), email, κωδικό σύνδεσης (password), ρόλο (role)και ένα μοναδικό id. Η οντότητα αυτή αφορά τους νέους χρήστες που δημιουργούν λογαριασμό στο σύστημα.
- **Constraint** (Περιορισμός Υποψήφιου Διδάκτορα): με πεδία ημερομηνίας και ώρας έναρξης και λήξης, τίτλο και υποψήφιο διδάκτορα (Phd Student) και ένα μοναδικό id.
- **Exam** (Εξέταση μαθήματος): με πεδία μάθημα (Lesson), μια λίστα επιτηρητών και ένα μοναδικό id.

- **QrCodeItem** : με πεδία έναν υποψήφιο διδάκτορα (Phd Student), ένα μάθημα (Lesson), μια περιγραφή καθώς και ημερομηνία και ώρα σκαναρίσματος. Η οντότητα αυτή αφορά τους κωδικούς τεχνολογίας QR που σκανάρουν οι επιτηρητές προκειμένου να δηλώσουν την παρουσία τους.

Αφού δημιουργήθηκαν οι οντότητές μας, μένει να δημιουργήσουμε το repository interface και το αντίστοιχο implementation του. Για κάθε οντότητα κατασκευάζουμε ένα DAO (Data Access Object) interface, το οποίο αποτελεί το επίπεδο στην εφαρμογή μας που περιλαμβάνει όλες τις κλάσεις που είναι υπεύθυνες για την επικοινωνία με τη βάση δεδομένων. Οι υλοποιήσεις αυτών των interfaces, προκειμένου να επικοινωνήσουν με την βάση ώστε να λάβουν η να επεξεργαστούν δεδομένα, χρησιμοποιούν το MongoTemplate bean που όπως προαναφέρθηκε έχει δηλωθεί στο mvc-dispatcher-servlet.xml του spring. Για να μπορέσουν να εκτελέσουν queries (ερωτήματα) στη βάση είναι απαραίτητη η χρήση των κλάσεων org.springframework.data.mongodb.core.query.Query και org.springframework.data.mongodb.core.query.Criteria.

Για κάθε υλοποίηση των DAO interfaces δηλώνουμε στο mvc-dispatcher-servlet.xml ένα αντίστοιχο bean. Για το παράδειγμα του phdStudentDAO δηλώνουμε το παρακάτω bean, το οποίο έχει ένα property mongoTemplate ώστε να μπορέσει να χρησιμοποιήσει το bean κλάσης org.springframework.data.mongodb.core, για την εκτέλεση ερωτημάτων στη βάση δεδομένων.

```
<bean id="phdStudentDAO"
      class="unipi.scheduling.dao.impl.PhdStudentDAO">
      <property name="mongoTemplate" ref="mongoTemplate"></property>
</bean>
```

Αν για παράδειγμα θέλουμε να ζητήσουμε από τη βάση δεδομένων όλους τους καθηγητές, πρέπει να γράψουμε

```
mongoTemplate.findAll( Professor.class, PROFESSOR_COLLECTION);
```

όπου δηλώνουμε το collection στο οποίο θέλουμε να ψάξουμε (`PROFESSOR_COLLECTION`) αλλά και τον τύπο που θα μας επιστραφεί (`Professor.class`).

Σε άλλες περιπτώσεις που επιθυμούμε την ικανοποίηση συγκεκριμένων κριτηρίων η σύνταξη είναι διαφορετική. Έστω ότι θέλουμε να διαγράψουμε έναν

καθηγητή με βάση το μοναδικό id του. Πρέπει αρχικά να ορίσουμε το κριτήριο που πρέπει να πληρείται ως εξής:

```
Query query = new Query(Criteria.where("_id").is(professorId));
```

Στο κριτήριο αυτό δηλώνουμε ότι θέλουμε το _id που υπάρχει στη βάση να είναι ίδιο με το professorId που δίνουμε εμείς. Στη συνέχεια πρέπει να δώσουμε το κριτήριο αυτό μαζί με την εντολή για διαγραφή (remove):

```
mongoTemplate.remove(query, Professor.class, PROFESSOR_COLLECTION );
```

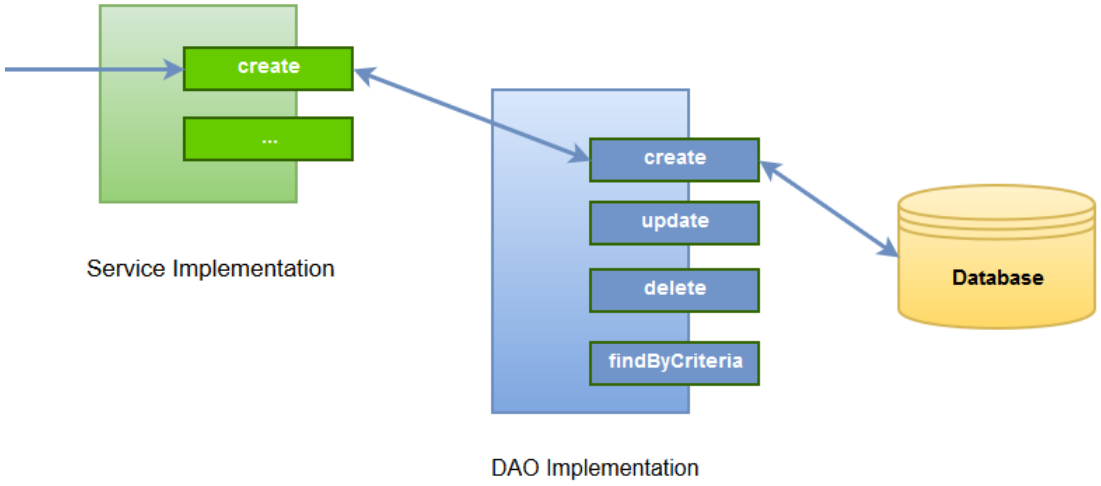
Οι υλοποιήσεις των DAO interfaces χρησιμοποιούνται από τα services που έχουν υλοποιηθεί για κάθε οντότητα, και τα services με τη σειρά τους καλούνται από τους controllers. Για κάθε service δηλώνεται στο mvc-dispatcher-servlet.xml του spring ένα bean με τόσα properties όσα και οι αντίστοιχες κλάσεις DAO που θα χρησιμοποιήσει. Για παράδειγμα, το service που περιέχει μεθόδους που θέλουν να επεξεργαστούν δεδομένα στο collection της βάσης μας, που αφορά τους υποψήφιους διδάκτορες, πρέπει να δηλωθεί ως εξής:

```
<bean id="phdStudentService"
      class="unipi.scheduling.service.impl.PhdStudentService">
            <property name="phdStudentDAO"
      ref="phdStudentDAO"></property>
            <property name="constraintDAO"
      ref="constraintDAO"></property>
</bean>
```

Όπως βλέπουμε για το phdStudentService έχει οριστεί property τόσο για το phdStudentDAO που εκτελεί ερωτήματα στο collection της βάσης μας, που αφορά τους υποψήφιους διδάκτορες όσο και για το constraintDAO που εκτελεί ερωτήματα στο collection της βάσης μας, που τους περιορισμούς των υποψήφιων διδακτόρων. Στη συνέχεια στην υλοποίηση της κλάσης του PhdStudentService πρέπει να δηλωθούν τα αντίστοιχα DAOs καθώς και μέθοδοι get και set για αυτά.

```
PhdStudentDAO phdStudentDAO;
ConstraintDAO constraintDAO;

public ConstraintDAO getConstraintDAO()
{
      return constraintDAO;
}
```

```
public void setConstraintDAO(ConstraintDAO constraintDAO)
{
	this.constraintDAO = constraintDAO;
}

public PhdStudentDAO getPhdStudentDAO()
{
	return phdStudentDAO;
}

public void setPhdStudentDAO(PhdStudentDAO phdStudentDAO)
{
	this.phdStudentDAO = phdStudentDAO;
}
```



Εικόνα 27: Υλοποίηση επιπέδου DAO

Το τμήμα του View για να έχει πρόσβαση στο μοντέλο και κατ' επέκταση στη βάση δεδομένων καλεί τον αντίστοιχο controller κάθε φορά ο οποίος καλεί τα beans των services που έχουν δηλωθεί στον DispatcherServlet του Spring αφού πρώτα τα δηλώσει με annotation Autowired το interface του service ως εξής:

```
@Autowired
IPhdStudentService  phdStudentService;
```

Για κάθε οντότητα του μοντέλου της εφαρμογής έχουν υλοποιηθεί ξεχωριστά services αλλά και controllers. Για να καλέσει το view έναν controller και τις μεθόδους που περιέχει θα πρέπει να κάνει μια κλήση στο url που αντιστοιχεί στον controller και

συγκεκριμένα στη μέθοδο που θέλει να καλέσει. Όλοι οι controllers βρίσκονται κάτω από το root (βασικό) path της διαδικτυακής εφαρμογής το οποίο δηλαδή κάτω από το *SchedulingResources*. Έστω ότι η εφαρμογή μας τρέχει τοπικά, και ότι ένα τμήμα του view θέλει να εμφανίζει όλους τους καθηγητές του συστήματος. Στην περίπτωση αυτή θα πρέπει να καλέσει το service που βρίσκεται υπό το url *http://localhost:8080/SchedulingWebApp/professor/all*.

Το *url http://localhost:8080/SchedulingWebApp* αποτελεί το root url της εφαρμογής μας, δηλαδή όλο το πληροφοριακό σύστημα βρίσκεται κάτω από αυτό, ενώ κάτω από το *http://localhost:8080/SchedulingWebApp/professor* ακούει ο controller που καλεί services τα οποία αφορούν τον controller. Συγκεκριμένα στο */all* ακούει η μέθοδος του controller που καλεί το service το οποίο με τη σειρά του καλεί το DAO που επιστρέφει τελικά από τη βάση όλους τους καθηγητές οι οποίοι εμφανίζονται στο view. Όπως βλέπουμε το view δεν έχει άμεση επικοινωνία με το μοντέλο της εφαρμογής.

Οι σελίδες του view βρίσκονται κάτω από το root path της εφαρμογής, συγκεκριμένα στο φάκελο views, και είναι της μορφής HTML. Το spring πρέπει να γνωρίζει που βρίσκεται το view. Αυτό δηλώνεται με ένα bean κλάσης `org.springframework.web.servlet.view.InternalResourceViewResolver` στον DispatcherServlet του Spring. Η δήλωση αυτή γίνεται ως εξής:

```
<bean class="org.springframework.web.servlet.view.
InternalResourceViewResolver">
      <property name="prefix">
            <value>/views/</value>
      </property>
      <property name="suffix">
            <value>.html</value>
      </property>
</bean>
```

Από την άλλη το spring πρέπει να ξέρει και πού πρέπει να ψάξει για τους controllers. Αυτό δηλώνεται στον DispatcherServlet του Spring υποδιεκνύοντάς του σε ποιο πακέτο της εφαρμογής βρίσκονται οι controllers ως εξής: `<context:component-scan base-package="unipi.scheduling.controller" />`

Στο σημείο αυτό πρέπει να τονισθεί ότι δεν ήταν επιθυμητή η πρόσβαση όλων των σελίδων από όλους τους χρήστες. Για το λόγο αυτό έπρεπε να οριστούν ρόλοι χρηστών αλλά και να προσδιοριστεί ποιοι ρόλοι έχουν πρόσβαση σε ποιες σελίδες.

Αυτό επιτεύχθηκε με τη χρήση της βιβλιοθήκης του Spring Framework που αφορά την ασφάλεια, την spring-security. Κατά τα γνωστά, για να καταστεί δυνατή η χρήση της βιβλιοθήκης αυτής πρέπει να δηλωθεί στο pom.xml του maven ως εξής:

```
<dependency>
	<groupId>org.springframework.security</groupId>
	<artifactId>spring-security-web</artifactId>
	<version>${spring.security.version}</version>
</dependency>

<dependency>
	<groupId>org.springframework.security</groupId>
	<artifactId>spring-security-config</artifactId>
	<version>${spring.security.version}</version>
</dependency>
```

Για τη χρήση της βιβλιοθήκης spring-security είναι απαραίτητη η σύνταξη ενός αρχείου μορφής xml το οποίο περιέχει όλες τις απαραίτητες δηλώσεις για τους ρόλους των χρηστών του συστήματος τη διαχείριση τους και τις σελίδες στις οποίες έχουν πρόσβαση. Το αρχείο αυτό πρέπει να δηλωθεί στο web.xml της εφαρμογής. Επιπλέον στο web.xml πρέπει να δηλωθεί ότι η πρόσβαση των σελίδων θα "φιλτράρεται" από την βιβλιοθήκη spring security. Δηλαδή πρέπει να προσθέσουμε στο web.xml τα:

```
<filter>
	<filter-name>springSecurityFilterChain</filter-name>
	<filter-
class>org.springframework.web.filter.DelegatingFilterProxy
	</filter-class>
</filter>

<filter-mapping>
	<filter-name>springSecurityFilterChain</filter-name>
	<url-pattern>/*</url-pattern>
</filter-mapping>
```

και

```
<context-param>
	<param-name>contextConfigLocation</param-name>
	<param-value>/WEB-INF/mvc-dispatcher-servlet.xml  /WEB-
INF/security-app-context.xml  </param-value>
</context-param>
```

Για να αποφασιστεί από το spring security αν ένας χρήστης επιτρέπεται να προβάλει μια σελίδα θα πρέπει αρχικά να ταυτοποιηθούν τα στοιχεία του με τα στοιχεία που υπάρχουν στη βάση. Επιπλέον θα πρέπει να έχουν δηλωθεί στο αρχείο security-app-context.xml οι συνθήκες που πρέπει να επαληθεύονται ώστε να

επιτραπεί η πρόσβαση στην επιθυμητή σελίδα. Για την ταυτοποίηση των στοιχείων δηλώνεται ένα Service το οποίο κάνει extend την κλάση UserDetailsService του spring και αναζητά στη βάση τον χρήστη με βάση τα στοιχεία του. Επιπλέον στον authentication-manager δηλώνουμε τον τρόπο με τον οποίο το spring security θα αναζητήσει τον χρήστη που συνδέθηκε.

Η δήλωση γίνεται ως εξής:

```
<authentication-manager>
      <authentication-provider user-service-ref="loginService">
      </authentication-provider>
</authentication-manager>

<beans:bean id="loginService"
class="unipi.scheduling.auth.LoginService" />
```

Επιπλέον θα πρέπει να δηλωθεί ένα bean για την διαχείριση της επιτυχούς ταυτοποίησης του χρήστη. Η κλάση του bean είναι unipi.scheduling.auth.CustomSuccessHandler η οποία κάνει extend την AuthenticationSuccessHandler του Spring.

```
<beans:bean id="customSuccessHandler"
            class="unipi.scheduling.auth.CustomSuccessHandler" />
```

Στην περίπτωση που τα στοιχεία δεν ταυτοποιηθούν, τότε αναλαμβάνει η κλάση unipi.scheduling.auth.RestAuthenticationFailureHandler η οποία κάνει extend την AuthenticationFailureHandler του Spring. Για την κλάση αυτή ορίζεται επίσης ένα bean στο οποίο δηλώνουμε και την σελίδα στην οποία θέλουμε να ανακατευθυνθεί ο χρήστης στην περίπτωση που δεν εισάγει τα σωστά στοιχεία. Δηλαδή δηλώνεται στο security-app-context.xml:

```
<beans:bean id="restAuthenticationFailureHandler"

      class="unipi.scheduling.auth.RestAuthenticationFailureHandler">
      <beans:property name="defaultFailureUrl"
      value="/views/error.html"/>
</beans:bean>
```

Εκτός από τα παραπάνω, είναι απαραίτητη και η δήλωση ενός bean που είναι κλάσης unipi.scheduling.auth.RestAuthenticationEntryPoint η οποία κάνει extend την AuthenticationEntryPoint του Spring και αποφασίζει ανάλογα με το ρόλο του χρήστη εάν θα του επιτραπεί η προβολή κάποιας σελίδας. Για να το αποφασίσει αυτό θα

πρέπει να έχουν δηλωθεί κανόνες πρόσβασης για κάθε σελίδα ή για ένα σύνολο σελίδων. Αν κάποια σελίδα ή υποφάκελος σελίδων δεν συμπεριληφθεί στους κανόνες αυτός τότε το RestAuthenticationEntryPoint θεωρεί ότι η πρόσβαση επιτρέπεται σε όλους τους χρήστες. Η δήλωση του bean είναι η εξής:

```
<beans:bean id="restAuthenticationEntryPoint"

      class="unipi.scheduling.auth.RestAuthenticationEntryPoint" />
```

ενώ για τους κανόνες που ακολουθεί γράφουμε:

```
<http use-expressions="true" entry-point-
      ref="restAuthenticationEntryPoint">
      <intercept-url pattern="/views/login**" access="permitAll"/>
      <intercept-url pattern="/views/error**" access="permitAll"/>
      <intercept-url pattern="/views/admin**"
            access="hasRole('Admin')" />
      <intercept-url pattern="/views/professor**"
            access="hasRole('Professor')" />
      <intercept-url pattern="/views/phd**"
            access="hasRole('Phd')" />

      <form-login login-page='/index.jsp'
         authentication-success-handler-ref="customSuccessHandler"
         authentication-failure-handler-
            ref="restAuthenticationFailureHandler"/>

      <logout logout-success-url="/"
            logout-url="/j_spring_security_logout" />
</http>
```

Οι σελίδες που βρίσκονται κάτω από το path /views/login είναι προσβάσιμες από όλους τους χρήστες αφού δηλώνεται σαν τιμή του access attibute το permitAll. Το ίδιο ισχύει και για το path /views/error. Αντίθετα όσες σελίδες είναι κάτω από το path /views/admin είναι προσβάσιμες μόνο από όσους χρήστες έχουν τον ρόλο admin αφού έχει δηλωθεί τιμή του access attibute το hasRole('Admin'). Αντίστοιχα ότι βρίσκεται κάτω από το /views/professor μπορεί να το δει μόνο κάποιος χρήστης με ρόλο Professor ενώ για το path /views/phd, οι χρήστες που έχουν πρόσβαση είναι όσοι έχουν ρόλο Phd.

Οι χρήστες έχουν πρόσβαση στην εφαρμογή και στα δεδομένα της μέσω του τμήματος view. Το view (front-end) έχει υλοποιηθεί σε Angular JS. Στο βασικό αρχείο του τμήματος αυτού, το app.js, δηλώνονται όλες οι εξωτερικές βιβλιοθήκες που χρησιμοποιούνται καθώς και η διαμόρφωση της ανακατεύθυνσης των σελίδων

δηλαδή η αντιστοίχιση των url του browser με τη εκάστοτε σελίδα της εφαρμογής και τον controller στον οποίο "ακούει". Οι βιβλιοθήκες που έχουν οριστεί στο app.js είναι

- *'isteven-multi-select'*, που επιτρέπει τη χρήση ενός html στοιχείου για την πολλαπλή επιλογή σε drop down μένου
- *'ngResource'*, που χρησιμοποιείται για τη δημιουργία των απαραίτητων services που καλούν το τμήμα του controller του back end
- *'ui.bootstrap'*, για τη χρήση του bootstrap front-end framework
- *'nsPopover'*, που επιτρέπει τη χρήση html στοιχείων τύπου pop up
- *'angularjs-datetime-picker'*, για την χρήση html στοιχείου για επιλογή ημερομηνίας και ώρας
- *'schedulingServices'*, αποτελούν όλα τα services του front end τμήματος, τα οποία βρίσκονται στο αρχείο schedulingServices.js
- *'ngRoute'*, για την ανακατεύθυνση των σελίδων της εφαρμογής
- *'mwl.calendar'*, για τη χρήση του ημερολογίου με τη βοήθεια του οποίου αναπαρίσταται το εξαγόμενο πρόγραμμα επιτηρήσεων
- *'ngDialog'*, που επιτρέπει τη χρήση ενός html στοιχείου τύπου παραθύρου
- *'jp.ng-bs-animated-button'*, για τη χρήση ιδιαίτερων κουμπιών
- *'ngToast'*, για την εμφάνιση ενημερωτικών μηνυμάτων.

Για να είναι δυνατή η χρήση των βιβλιοθηκών που δηλώνονται στο app.js είναι απαραίτητη και η δήλωση των απαραίτητων script εντολών και διαμόρφωσης εμφάνισης (css), η οποία γίνεται σε όποια html σελίδα επιθυμεί να χρησιμοποιήσει κάποια από τις προαναφερθείσες βιβλιοθήκες.

Ενδεικτική δήλωση ανακατεύθυνση σελίδας στο app.js είναι η

```
.config(function ($routeProvider) {
   $routeProvider
      .when('/adminProfessors', {
            templateUrl: '../views/adminProfessors.html',
            controller: 'adminProfController'
       })
    });
```

στην οποία δηλώνουμε ότι όταν το url του browser γίνει /adminProfessors τότε η σελίδα που πρέπει να εμφανιστεί είναι η adminProfessors.html που βρίσκεται στο φάκελο views και ακούει στον controller adminProfController.

Για την επικοινωνία με το τμήμα του controller και κατ' επέκταση με το model, στο τμήμα του view, που κατασκευάστηκε και αυτό με την MVC αρχιτεκτονική που παρέχει η Angular JS, υλοποιήθηκαν services που "χτυπούν" τα url paths των controller που επιθυμούν. Ας πάρουμε για παράδειγμα μια σελίδα στην οποία εμφανίζονται όλοι οι καθηγητές του τμήματος. Αρχικά δημιουργήθηκε ένα αρχείο schedulingServices.js στο οποία δηλώθηκαν όλα τα απαιτούμενα services του view. Η μέθοδος του service που επιστρέψει στο view όλους καθηγητές πρέπει να επικοινωνήσει με τον controller που βρίσκεται κάτω από το /professor και συγκεκριμένα να καλέσει τη μέθοδο που βρίσκεται κάτω από το /all. Για τη δήλωση του service με την μέθοδο που προαναφέρθηκε γράφουμε:

```
schedulingServices.factory('Professor', function($resource){
      var service = $resource('../professor', {}, {

            getAllProfessors:{
                  method:'GET',
                  url:'../professor/all'

            }

       });
       return service;
});
```

όπου Professor το όνομα του service και getAllProfessors η μέθοδος που επιστρέφει όλους τους καθηγητές.

Ο controller τουν view που θέλει να χρησιμοποιήσει οποιαδήποτε μέθοδο του service Professor, θα πρέπει να το δηλώσει ως εξής:

```
angular.module('schedulingApp')

  .controller('adminProfController', ['$scope', 'Professor',
function($scope, Professor) {
.....
```

και τελικά να χρησιμοποιήσει την κατάλληλη μέθοδο. Με την εντολή:

```
$scope.professors = Professor.getAllProfessors();
```

καταχωρούμε σε μια μεταβλητή του scope τους καθηγητές που μας επιστρέφει η μέθοδος getAllProfessors του service Professor για να μπορέσουμε να τους

εμφανίσουμε στην HTML σελίδα που ακούει στον συγκεκριμένο controller. Σε μορφή πίνακα. Οι controllers από το backend επιστρέφουν στο view δεδομένα με τη μορφή json αντικειμένων. Για να γίνει αυτό, έχουμε δηλώσει στο pom.xml τη βιβλιοθήκη Jackson η οποία μετατρέπει τα δεδομένα υπό μορφή json. Δηλώνουμε δηλαδή:

```
<dependency>
	<groupId>com.fasterxml.jackson.core</groupId>
	<artifactId>jackson-databind</artifactId>
	<version> 2.4.3 </version>
</dependency>
<dependency>
	<groupId>org.codehaus.jackson</groupId>
	<artifactId>jackson-mapper-asl</artifactId>
	<version>1.9.13</version>
</dependency>
<dependency>
	<groupId>org.codehaus.jackson</groupId>
	<artifactId>jackson-core-lgpl</artifactId>
	<version>1.9.13</version>
</dependency>

<dependency>
	<groupId>org.codehaus.jackson</groupId>
	<artifactId>jackson-mapper-lgpl</artifactId>
	<version>1.9.13</version>
</dependency>
```

Αντίστοιχα, το service που έχει υλοποιηθεί στο τμήμα του view, θα πρέπει να τροποποιήσει κατάλληλα τα δεδομένα που επιστρέφονται από τον controller του back-end για να μπορέσει να τα χρησιμοποιήσει. Η προσθήκη των ακόλουθων γραμμών στη μέθοδο του service Professor που επιστρέφει όλους καθηγητές, εξυπηρετεί ακριβώς αυτό το σκοπό:

```
isArray: true,
transformResponse: function(data) {
	try {
		data = (angular.fromJson(data));
	} catch (Error) {
		console.log('Error');
	}
	return data;
}
```

Στην μεταβλητή $scope.professors έχουν καταχωρηθεί όλοι οι καθηγητές τους οποίους θέλουμε να εμφανίσουμε συγκεντρωτικά σε έναν πίνακα. Στην HTML σελίδα που επιθυμούμε να γίνει αυτό, χρησιμοποιώντας τη λέξη κλειδί ng-repeat που παρέχει η Angular, εμφανίζουμε όλους τους καθηγητές γράφοντας:

```
<table>
      <tr ng-repeat="professor in professors>
      <td>{{professor.fname}}</td>
      <td>{{professor.lname}}</td>
      <td>{{professor.mail}}</td>
      <td>{{professor.phone}}</td>
</table>
```

Με τον τρόπο αυτό για κάθε καθηγητή (professor) που υπάρχει στη μεταβλητή professors, εμφανίζουμε το όνομα (fname), το επίθετο (lname), το email και το τηλέφωνό του (phone). Ακολουθώντας την αντίστοιχη διαδικασία μπορούμε να εμφανίσουμε στο view δεδομένα από το model της εφαρμογής.

Το σημαντικότερο τμήμα της εφαρμογής αποτελεί το service το οποίο δεδομένων όλων των απαραίτητων στοιχείων κατασκευάζει το πρόγραμμα των επιτηρήσεων, και κρίνεται αναγκαίο να περιγραφεί η υλοποίηση και η λειτουργία του. Το service αυτό είναι το ExamService και η μέθοδος που κατασκευάζει το πρόγραμμα είναι η generateProgram(). Εδώ αρχικά χρειαζόμαστε όλα τα μαθήματα που υπάρχουν στη βάση δεδομένων μας, επομένως πρέπει μέσω του Data Access Object που έχει κατασκευαστεί για την οντότητα του μαθήματος (Lesson), να τα ζητήσουμε από τη βάση. Για κάθε μάθημα αναζητούνται τόσοι επιτηρητές όσους έχει ορίσει ο διαχειριστής της σελίδας. Αρχικά η αναζήτηση γίνεται στους υποψήφιους διδάκτορες του συγκεκριμένου μαθήματος. Για κάθε υποψήφιο εξετάζουμε εάν οι προσωπικοί περιορισμοί του δεν συμπίπτουν χρονικά με τις ημερομηνίες έναρξης και λήξης του μαθήματος για το οποίο ψάχνουμε επιτηρητές. Εάν όντως δεν συμπίπτουν τότε εξετάζεται και ο συνολικός αριθμός των επιτηρήσεων του υποψήφιου διδάκτορα. Εάν υπάρχει διαθέσιμος υποψήφιος με μικρότερο αριθμό επιτηρήσεων, τότε η επιτήρηση ανατίθεται σε αυτόν με τις λιγότερες επιτηρήσεις. Ο επιτηρητής που επιλέγεται, προστίθεται στη λίστα των διαθέσιμων επιτηρητών. Εάν δεν βρεθεί κανένας υποψήφιος διδάκτορας διαθέσιμος ανάμεσα σε αυτούς του μαθήματος, τότε η αναζήτηση συνεχίζεται στο σύνολο των υποψήφιων διδακτόρων που ανήκουν στον καθηγητή που διδάσκει το μάθημα, ακολουθώντας την ίδια διαδικασία. Στην περίπτωση που δεν βρεθεί κάποιος επιτηρητής από το σύνολο αυτό των υποψήφιων διδακτόρων, τότε η αναζήτηση γενικεύεται στο σύνολο όλων των υποψήφιων διδακτόρων του συστήματος.

Ανάλογα με τα αποτελέσματα της αναζήτησης, ενημερώνονται κατάλληλα οι ενδιαφερόμενοι καθηγητές και οι υποψήφιοι διδάκτορες. Στην περίπτωση που το σύνολο των υποψήφιων διδακτόρων που βρέθηκε, δεν καλύπτει το αριθμό των απαιτούμενων επιτηρητών, τότε ο καθηγητής του μαθήματος ενημερώνεται μέσω email με κατάλληλο μήνυμα.

Για την αποστολή email έχει δηλωθεί στο pom.xml η βιβλιοθήκη `javax.mail`.

```
<dependency>
	<groupId>javax.mail</groupId>
	<artifactId>mail</artifactId>
	<version>1.4.3</version>
</dependency>
```

Επιπλέον θα πρέπει να δηλωθεί στον DispatcherServlet του Spring ένα bean κλάσης org.springframework.mail.javamail.JavaMailSenderImpl, στο οποίο δηλώνουμε τον λογαριασμό mail στον οποίο θα συνδεθεί η εφαρμογή για να αποσταλεί στο email. Για το σκοπό αυτό δημιουργήθηκε λογαριασμός με email schedulingresourcesunipi@gmail.com και password unipi1234.

```
<bean id="mailSender"
class="org.springframework.mail.javamail.JavaMailSenderImpl">
	<property name="host" value="smtp.gmail.com"/>
	<property name="port" value="587"/>
	<property name="username"
	value="schedulingresourcesunipi@gmail.com"/>
	<property name="password" value="unipi1234"/>
	<property name="javaMailProperties">
		<props>
			<prop key="mail.transport.protocol">smtp</prop>
			<prop key="mail.smtp.auth">true</prop>
			<prop key="mail.smtp.starttls.enable">true</prop>
			<prop key="mail.debug">true</prop>
		</props>
	</property>
</bean>
```

Στη συνέχεια θα πρέπει να δηλωθεί ένα bean για το Service που έχει υλοποιηθεί με σκοπό την αποστολή email. Το service αυτό είναι το MailService, το οποίο χρησιμοποιεί το bean με id mailSender που ορίστηκε νωρίτερα.

```
<bean id="mail" class="unipi.scheduling.service.impl.MailService">
	<property name="mailSender" ref="mailSender"></property>
</bean>
```

Το MailService για να χρησιμοποιήσει το bean mailSender θα πρέπει να το δηλώσει με annotation AutoWired ως εξής:

```
@Autowired
JavaMailSender mailSender;
```

αλλά και να ορίσει μεθόδους get και set για το bean αυτό:

```
public JavaMailSender getMailSender() {
	return mailSender;
}

public void setMailSender(JavaMailSender mailSender) {
	this.mailSender = mailSender;
}
```

Το MailService μπορεί πλέον να κατασκευάσει και να στείλει ένα mail καλώντας

```
mailSender.send(message);
```

όπου message είναι ένα αντικείμενο τύπου SimpleMailMessage στο οποίο ορίζουμε αποστολέα, παραλήπτη, θέμα και κυρίως μήνυμα.

Η διαδικτυακή εφαρμογή φιλοξενείται πλέον στο Cloud και συγκεκριμένα στο Heroku, στην διεύθυνση https:/unipischeduling.herokuapp.com. Η διαδικασία που ακολουθήθηκε για να "ανέβει" η εφαρμογή στο Heroku περιγράφεται στη συνέχεια. Αρχικά έπρεπε να δημιουργηθεί ένας λογαριασμός στο Heroku, το οποίο αποτελεί μια πλατφόρμα εφαρμογών στο cloud και προσφέρεται σαν PaaS (Platform-as-a-Service). Έπειτα έπρεπε να εγκατασταθεί ένα εργαλείο για να μπορούν να εκτελεσθούν οι απαραίτητες εντολές. Το εργαλείο αυτό είναι το Heroku Toolbelt το οποίο μας δίνει πρόσβαση στο Heroku Command Line Interface (CLI). Μετά τη σύνδεση στο heroku (με την εκτέλεση της ενρολής heroku login) και τη δημιουργία ενός τοπικού git repository στο φάκελο που περιέχει τον κώδικα της εφαρμογής μας (εκτελώντας την εντολή git init) σειρά έχει η δημιουργία ενός απαραίτητου αρχείου για το Heroku, το Procfile, το οποίο περιέχει την `web: java $JAVA_OPTS -jar target/dependency/webapp-runner.jar --port $PORT target/*.war`, εντολή για την εκτέλεση του jar αρχείου της εφαρμογής. Για την εκτέλεση της εντολής αυτής είναι απαραίτητη η χρήση της βιβλιοθήκης webapp-runner την οποία δηλώνουμε στο pom.xml ως εξής:

```
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-dependency-plugin</artifactId>
   <version>2.3</version>
   <executions>
   <execution>
       <phase>package</phase>
       <goals><goal>copy</goal></goals>
       <configuration>
            <artifactItems>
                 <artifactItem>
                      <groupId>com.github.jsimone</groupId>
                      <artifactId>webapp-runner</artifactId>
                      <version>8.0.30.2</version>
                      <destFileName>webapp-runner.jar</destFileName>
                  </artifactItem>
             </artifactItems>
        </configuration>
    </execution>
    </executions>
 </plugin>
```

Στη συνέχεια πρέπει να ανεβάσουμε τον κώδικα του τοπικού μας repository στο απομακρυσμένο master repository του Heroku (`git push heroku master`). Το Heroku βρίσκει το Procfile αρχείο και εκτελεί την εντολή που περιέχει. Η εφαρμογή μας πλέον είναι στο Heroku. Για να την ανοίξουμε αρκεί να εκτελέσουμε την εντολή `heroku open`.

6.2 Mobile εφαρμογή

Για την υλοποίηση της mobile εφαρμογής εγκαταστάθηκαν αρχικά κάποια βασικά εργαλείο μέσω του διαχειριστή πακέτων (package manager) του Node.Js. Με την εκτέλεση της εντολής `npm install -g cordova ionic` στη γραμμή εντολών εγκαθιστούμε το Apache Cordova και το Ionic. Στη συνέχεια δημιουργούμε μια mobile εφαρμογή `εκτελώντας ionic start {appname} {template}`, όπου appname είναι το όνομα της εφαρμογής μας και template το βασικό template που θέλουμε να έχει η εφαρμογή. Το Ionic Framework προσφέρει τρία έτοιμα templates,τα blank, tabs και .Για την δημιουργία της mobile εφαρμογής που αναπτύχθηκε εκτελέσθηκε η εντολή `ionic start scheduling tabs`, η οποία παρήγαγε τα αρχεία:

```
├── hooks          // custom cordova hooks to execute on specific commands
├── platforms      // iOS/Android specific builds will reside here
├── plugins        // where your cordova/ionic plugins will be installed
├── resources        // icon and splash screen
├── scss           // scss code, which will output to www/css/
|── www         // application - JS code and libs, CSS, images, etc.
    |---------css                //customs styles
    |---------img               //app images
    |---------js                 //angular app and custom JS
    |---------lib               //third party libraries
    |---------index.html  //app master page
├── bower.json     // bower dependencies
├── config.xml     // cordova configuration
├── gulpfile.js    // gulp tasks
├── ionic.project  // ionic configuration
├── package.json   // node dependencies
```

Εικόνα 28: βασικά αρχεία Mobile εφαρμογής

Στη συνέχεια, στο φάκελο scheduling που δημιουργήθηκε από την παραπάνω εντολή, εκτελέσθηκαν οι εντολές :

```
ionic platform add ios
ionic platform add android
```

ώστε να είναι συμβατή η εφαρμογή τόσο σε λειτουργικό Android όσο και σε iOS.

Στη συνέχεια κάνοντας χρήση της Angular JS και εφαρμόζοντας την MVC αρχιτεκτονική της, κατασκευάστηκαν οι απαραίτητες σελίδες (view) , controllers και services. Στο βασικό αρχείο της εφαρμογής , στο app.js δηλώθηκαν οι απαραίτητες βιβλιοθήκες που χρησιμοποιήθηκαν ως εξής:

```
angular.module('starter', ['ionic', 'starter.controllers',
'starter.services', 'ng-mfb' ,'ngCordova'])
```

Συγκεκριμένα:

- *ionic*, για τη χρήση του ionic framework
- *starter.controllers*, όπου συγκεντρώνονται όλοι οι controllers της εφαρμογής
- *starter.services*, όπου συγκεντρώνονται όλα τα services της εφαρμογής
- *ng-mfb*, για τη χρήση ενός floating μενού
- *ngCordova*, για τη χρήση του Apache Cordova

Για να είναι δυνατή η χρήση των βιβλιοθηκών που δηλώνονται στο app.js είναι απαραίτητη και η δήλωση των απαραίτητων script εντολών και διαμόρφωσης εμφάνισης (css), η οποία γίνεται σε όποια html σελίδα επιθυμεί να χρησιμοποιήσει κάποια από τις προαναφερθείσες βιβλιοθήκες.

Επιπλέον στο αρχείο αυτό δηλώνεται η διαμόρφωση της ανακατεύθυνσης των σελίδων δηλαδή η αντιστοίχιση των url με τη εκάστοτε σελίδα της εφαρμογής και τον controller στον οποίο "ακούει". Η εφαρμογή αποτελείται από μια βασική σελίδα με δυο tabs. Ενδεικτικά για την ανακατεύθυνση στη βασική σελίδα των tabs αλλά και την σελίδα του προγράμματος γράφουμε:

```
.state('tab', {
    url: '/tab',
    abstract: true,
    templateUrl: 'templates/tabs.html'
})

.state('tab.program', {
    cache: false,
    url: '/program',
    views: {
      'tab-program': {
        templateUrl: 'templates/tab-program.html',
        controller: 'ProgramCtrl'
      }
    }
})
```

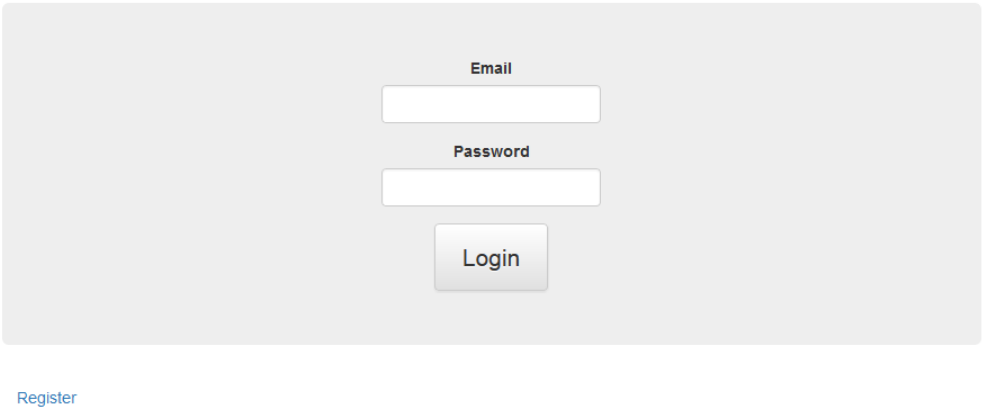
Τα services που αναπτύχθηκαν και καταναλώνονται από την mobile εφαρμογή, "χτυπούν" τους controllers της διαδικτυακής εφαρμογής η οποία όπως προαναφέρθηκε φιλοξενείται στο Heroku. Επομένως, τα urls που δηλώνουμε είναι κάτω από το url https:/unipischeduling.herokuapp.com. Για παράδειγμα για να υλοποιήσουμε μια μέθοδο που θα καλούμε για να σώσουμε έναν κωδικό QR στη βάση, γράφουμε:

```
services.factory('Service', function($resource) {

var service = $resource('https://unipischeduling.herokuapp.com/', {},
{

   saveQR:{
        method:'POST',
        url:'https://unipischeduling.herokuapp.com/qr/save',
        params:{qr:'@qr'}
   }
});
return service;
});
```

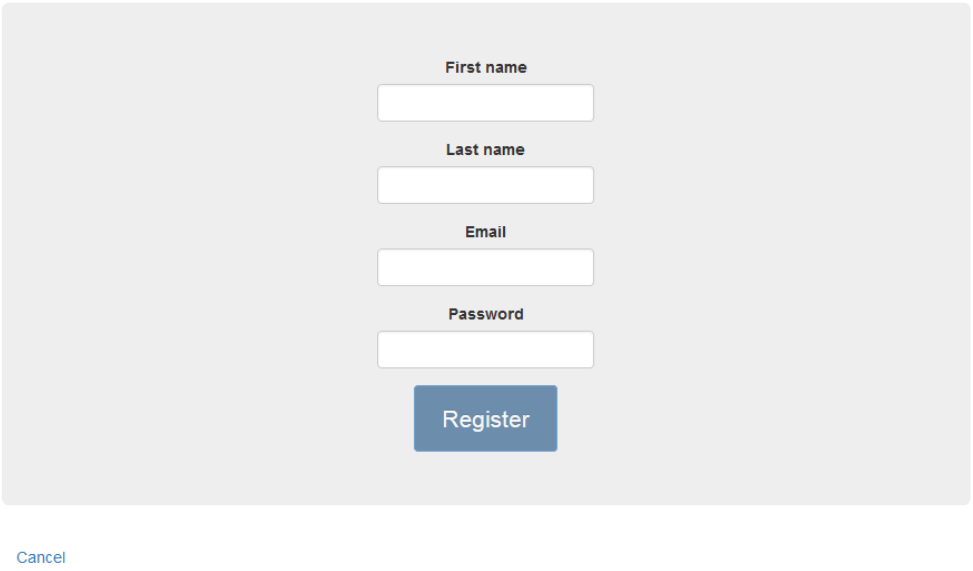
7 Παρουσίαση εφαρμογών

7.1 Web εφαρμογή

Η αρχική σελίδα της εφαρμογής περιλαμβάνει μια φόρμα εισαγωγής στοιχείων (εικόνα 29), διεύθυνσης email και κωδικού σύνδεσης, τα οποία πρέπει να εισάγει ο χρήστης για να συνδεθεί στην εφαρμογή. Τα στοιχεία αυτά τα είχε δηλώσει ο ίδιος κατά την δημιουργία του λογαριασμού του (εικόνα 30).



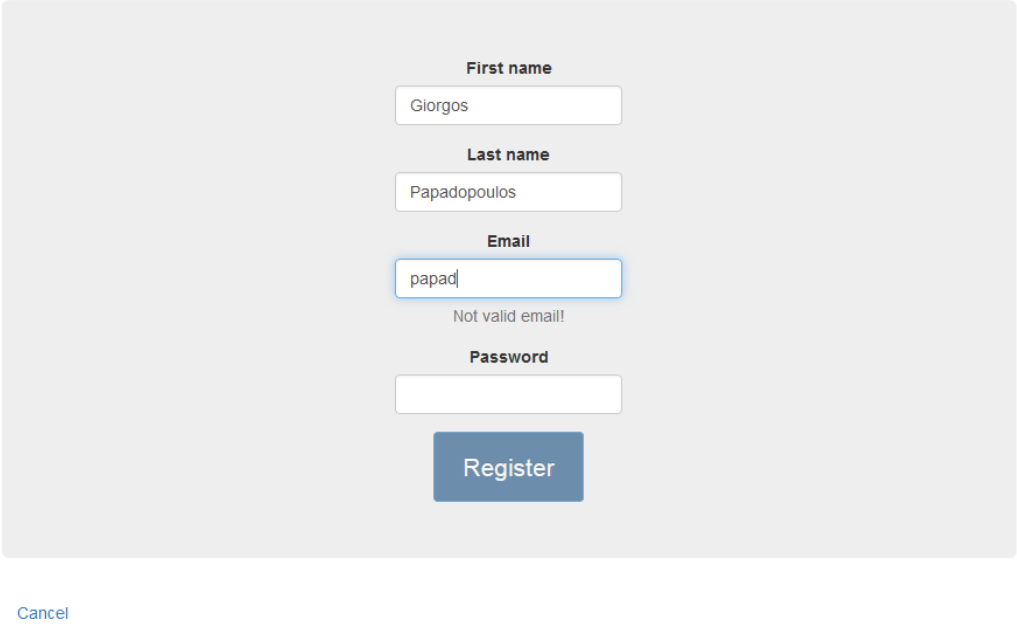
Εικόνα 29: Φόρμα σύνδεσης με το σύστημα



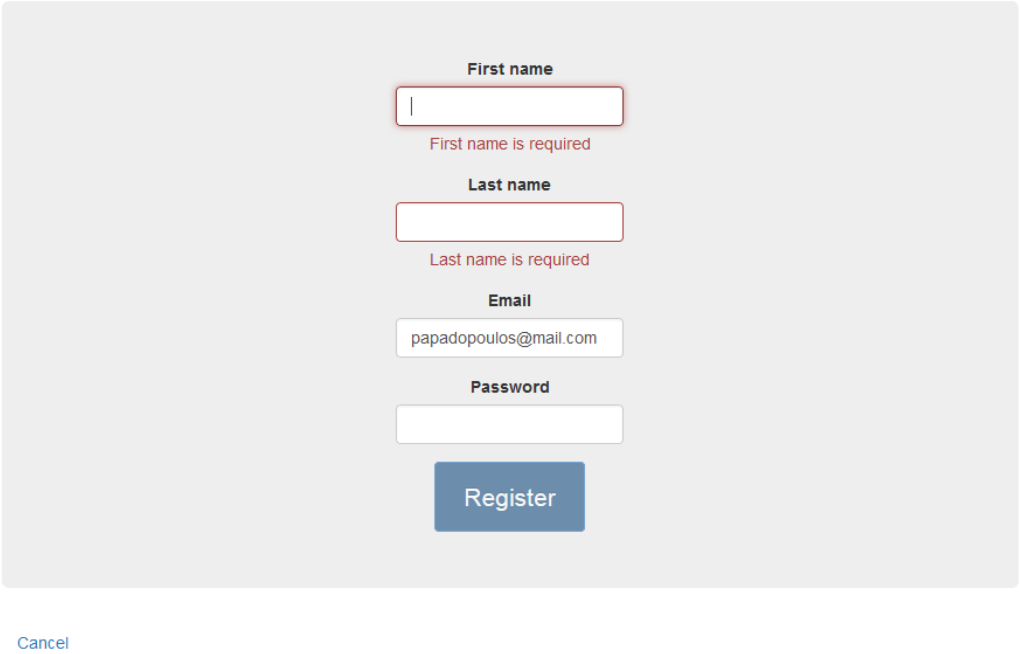
Εικόνα 30: Φόρμα δημιουργίας λογαριασμού

Η αρχική σελίδα είναι προσβάσιμη από οποιονδήποτε χρήστη. Στην περίπτωση που κάποιος νέος χρήστης επιθυμεί να δημιουργήσει νέο λογαριασμό, τότε αρκεί να πατήσει στο "*Register*" κουμπί-λινκ το οποίο κρύβει την φόρμα σύνδεσης (εικόνα 29) και εμφανίζει την φόρμα δημιουργίας λογαριασμού (εικόνα 30). Εάν ο χρήστης επιθυμεί να ακυρώσει την κίνηση του αυτή, μπορεί να πατήσει το κουμπί-λινκ "*Cancel*".

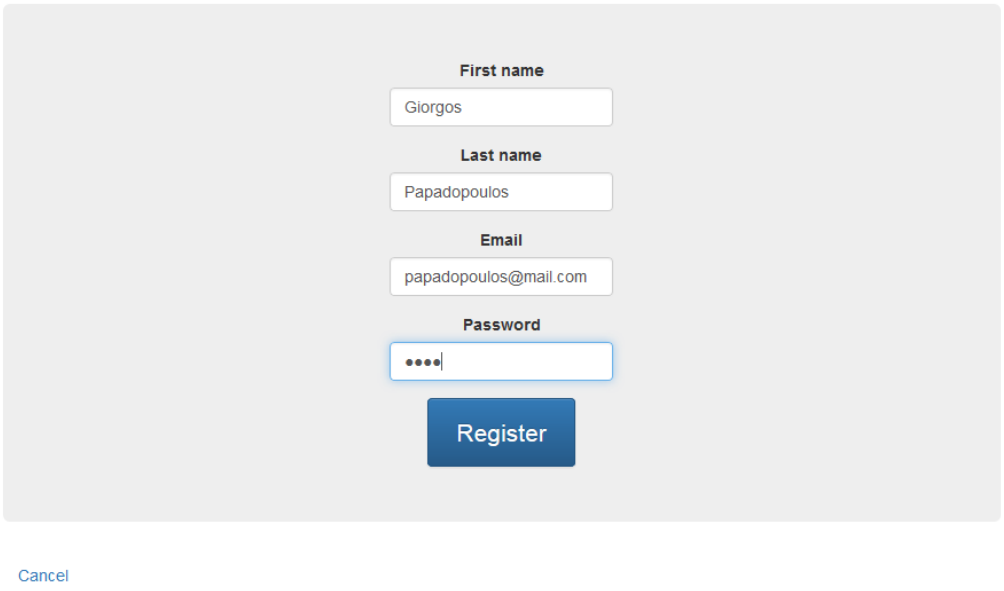
Κατά την εισαγωγή των στοιχείων του χρήστη στη φόρμα δημιουργίας νέου λογαριασμού, πραγματοποιείται έλεγχος εγκυρότητας των πεδίων κατά την πληκτρολόγηση. Όλα τα πεδία είναι υποχρεωτικά (εικόνα 32), και το email πρέπει να έχει την μορφή ενός email δηλαδή να περιέχει το σύμβολο @ (εικόνα 31). Εάν δεν πληρούνται όλες οι προϋποθέσεις το κουμπί δημιουργίας του λογαριασμού παραμένει ανενεργό. Μόνο εάν έχουν συμπληρωθεί όλα τα πεδία και το email είναι της σωστής μορφής μπορεί να προχωρήσει ο χρήστης στη δημιουργία του λογαριασμού του, πατώντας το κουμπί "*Register*" που έχει πλέον ενεργοποιηθεί (εικόνα 33). Η εφαρμογή καλωσορίζει το νέο χρήστη και τον ενημερώνει πως θα του αποσταλεί email ,στη διεύθυνση που δήλωσε, το οποίο θα τον ειδοποιεί ότι ο λογαριασμός του είναι έτοιμος(εικόνα 34).



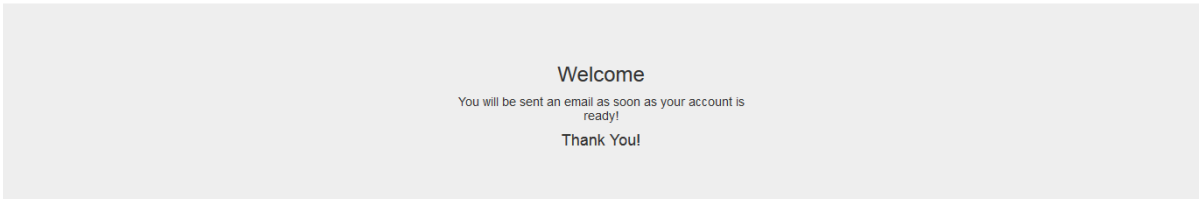
Εικόνα 29: Μη έγκυρη μορφή email



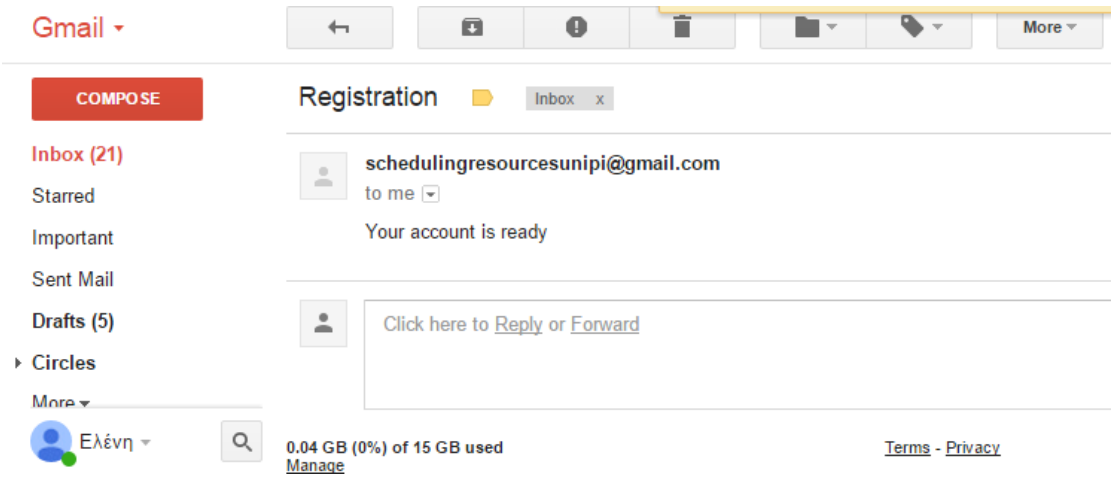
Εικόνα 30: Υποχρεωτικά πεδία



Εικόνα 31: Ενεργοποίηση κουμπιού Register



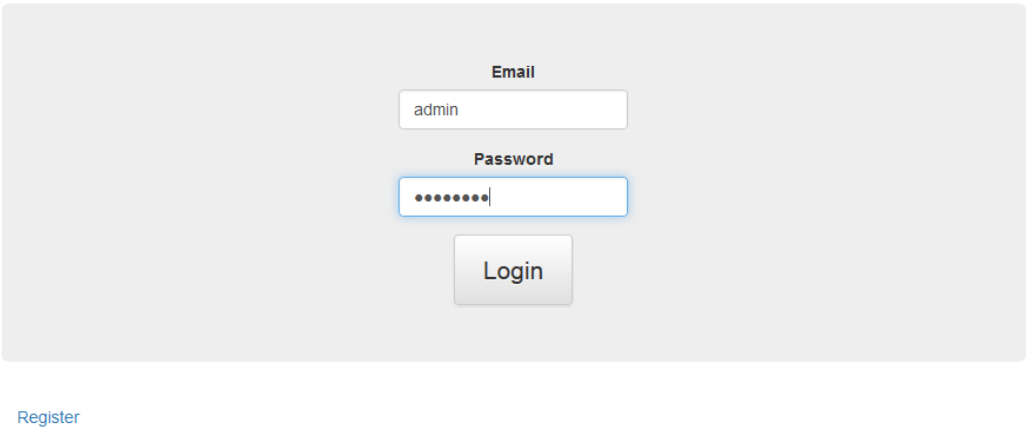
Εικόνα 32: Καλώς όρισμα νέου χρήστη



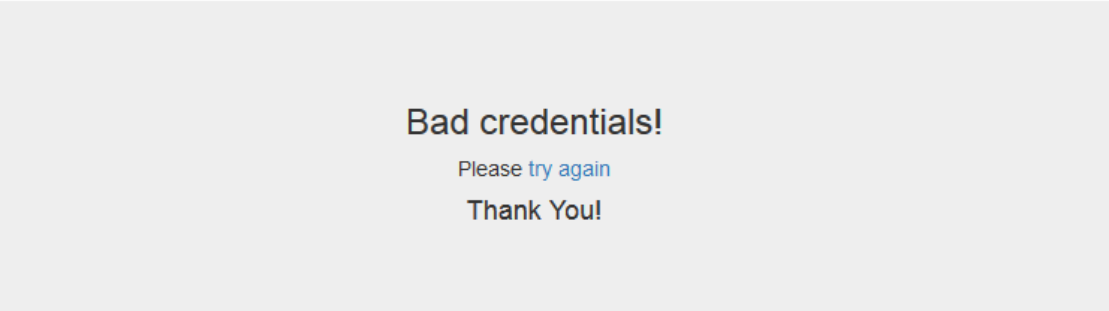
Εικόνα 33: Email ειδοποίησης για την ενεργοποίηση του νέου λογαριασμού

Στην περίπτωση που κάποιος χρήστης επιχειρήσει να συνδεθεί με λάθος στοιχεία ειδοποιείται με κατάλληλο μήνυμα (εικόνα 37) και καλείται να επιστρέψει στην αρχική σελίδα σύνδεσης για να ξαναπροσπαθήσει.

Η εφαρμογή διαθέτει έναν χρήστη - διαχειριστή, ο οποίος δημιουργήθηκε κατά την ανάπτυξη της εφαρμογής και αποθηκεύτηκε απευθείας στη βάση δεδομένων που φιλοξενείται στο Cloud. Ο χρήστης αυτός έχει όνομα χρήστη admin.



Εικόνα 34: Είσοδος διαχειριστή

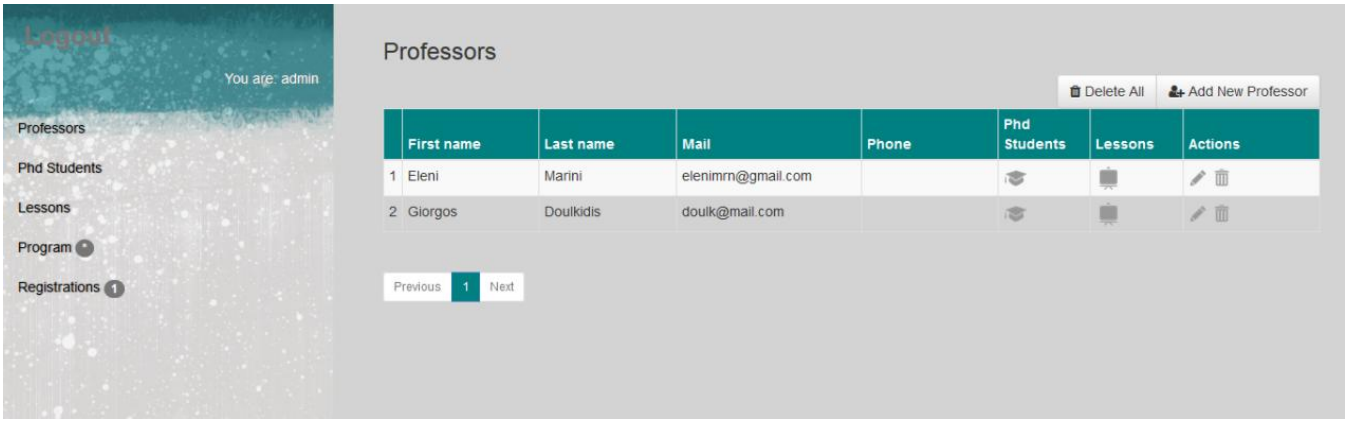


Εικόνα 35: Λανθασμένα στοιχεία εισόδου

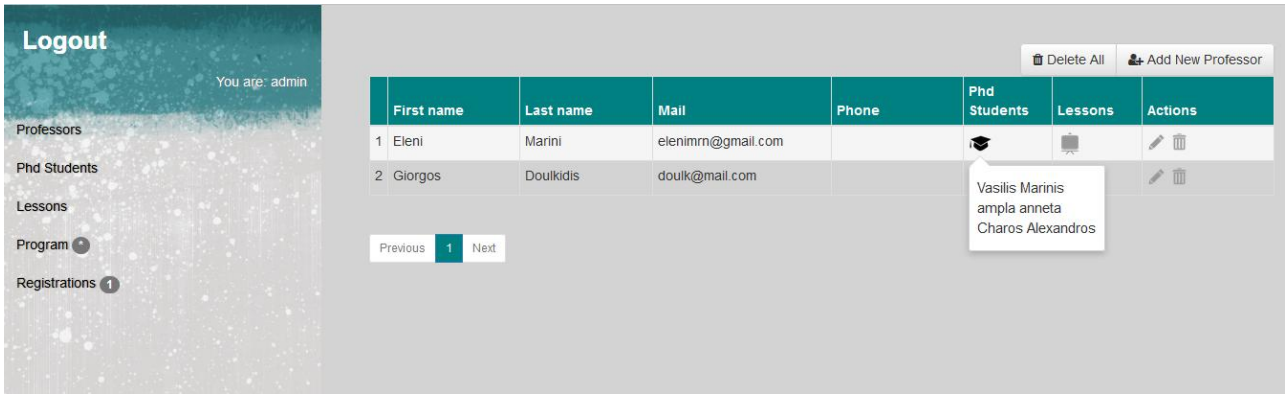
Μετά την επιτυχημένη είσοδο του διαχειριστή στο σύστημα, αυτός οδηγείται στην αρχική σελίδα που αφορά τον δικό του ρόλο (εικόνα 38). Στη σελίδα αυτή υπάρχει στην αριστερή πλευρά, ένα κάθετο μενού περιήγησης μέσω του οποίου ο διαχειριστής μπορεί να μεταβεί στις υπόλοιπες σελίδες. Επιπλέον στο μενού αυτό υπάρχει και ένα κουμπί "*LogOut*" πατώντας το οποίο πραγματοποιείται αποσύνδεση του χρήστη από τη σελίδα. Οι διαθέσιμες επιλογές στο μενού περιήγησης είναι οι :*Professors, Phd Students, Lessons, Program* και *Registrations.*

Εικόνα 36: Αρχική σελίδα διαχειριστή

Επιλέγοντας από το μενού, "*Professors*", ανοίγει η σελίδα διαχείρισης των καθηγητών στην οποία ο διαχειριστής μπορεί να δει όλους τους καθηγητές συγκεντρωμένους σε έναν πίνακα (εικόνα 39). Για κάθε καθηγητή, στον πίνακα, φαίνονται άμεσα τα προσωπικά του στοιχεία, όνομα, επίθετο, email, τηλέφωνο. Επιπλέον υπάρχουν 3 ακόμα στήλες με εικονίδια. Όταν κάνουμε "κλικ" με το ποντίκι στο εικονίδιο στη στήλη "*Phd Students*" που ανήκει σε έναν καθηγητή, τότε εμφανίζεται ένα λευκό κουτάκι που περιέχει όλους τους υποψήφιους διδακτορικούς του συγκεκριμένου καθηγητή (εικόνα 40).



Εικόνα 37: Υποσελίδα καθηγητών



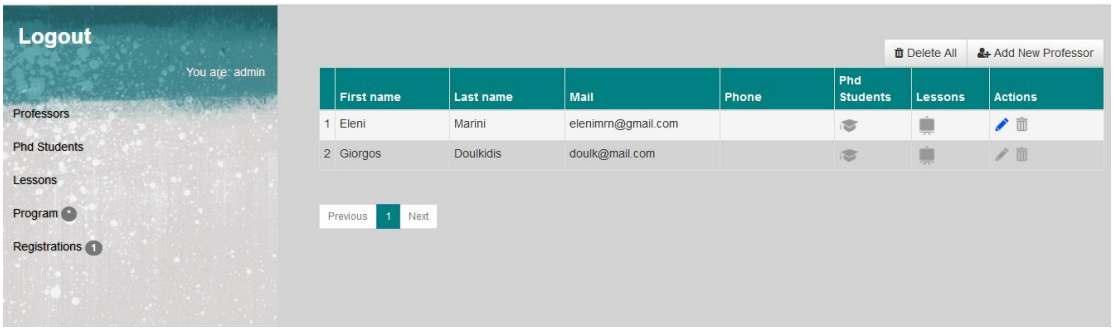
Εικόνα 38: Προβολή υποψήφιων διδακτορικών καθηγητή

Αντίστοιχα όταν κάνουμε "κλικ" με το ποντίκι στο εικονίδιο στη στήλη "*Lessons*" που ανήκει έναν καθηγητή, τότε στο λευκό κουτάκι εμφανίζονται όλα τα μαθήματα που διδάσκει ο καθηγητής (εικόνα 41).

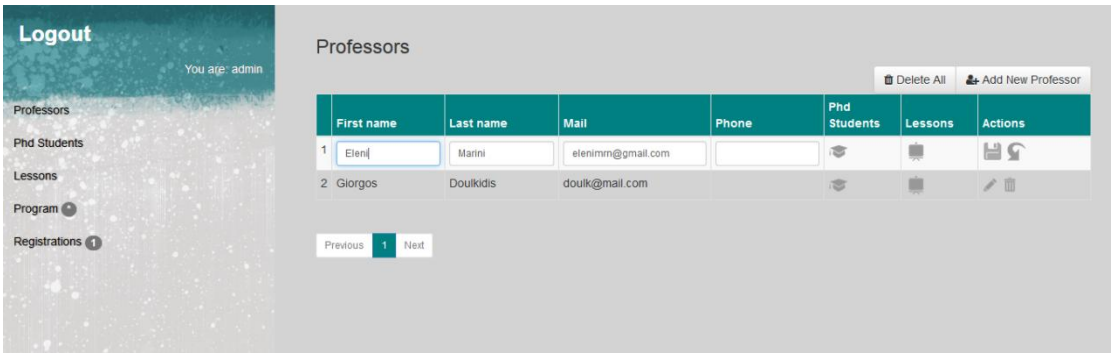
Εικόνα 39: Προβολή μαθημάτων καθηγητή

Στην τελευταία στήλη "*Actions*" βλέπουμε αρχικά δύο κουμπιά, τα οποία αλλάζουν χρώμα όταν το ποντίκι περάσει από πάνω τους (εικόνα 42, εικόνα 47). Τα κουμπιά αυτά προορίζονται για την επεξεργασία για την διαγραφή του καθηγητή που βρίσκεται στην ίδια γραμμή του πίνακα. Για να επεξεργαστούμε τα στοιχεία του καθηγητή αρκεί να πατήσουμε επάνω στο εικονίδιο που έχει το σχήμα μολυβιού , ενώ για να τον διαγράψουμε πρέπει να πατήσουμε το εικονίδιο που μοιάζει με κάδο σκουπιδιών .

Όταν επιλέξουμε να επεξεργαστούμε έναν καθηγητή, τότε ολόκληρη η γραμμή στην οποία ανήκει μπαίνει σε κατάσταση επεξεργασίας (εικόνα 43). Τα κουμπιά επεξεργασίας και διαγραφής της τελευταίας στήλης αντικαθίστανται από τα κουμπιά αποθήκευσης και ακύρωσης .

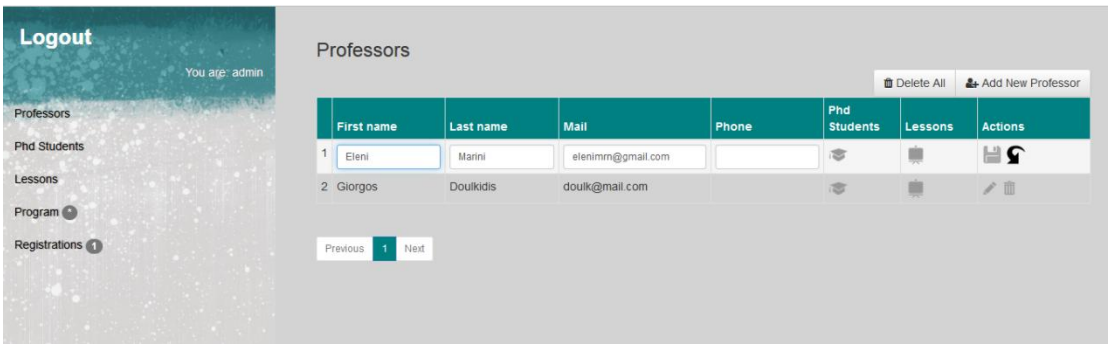


Εικόνα 40: Επιλογή για επεξεργασία καθηγητή

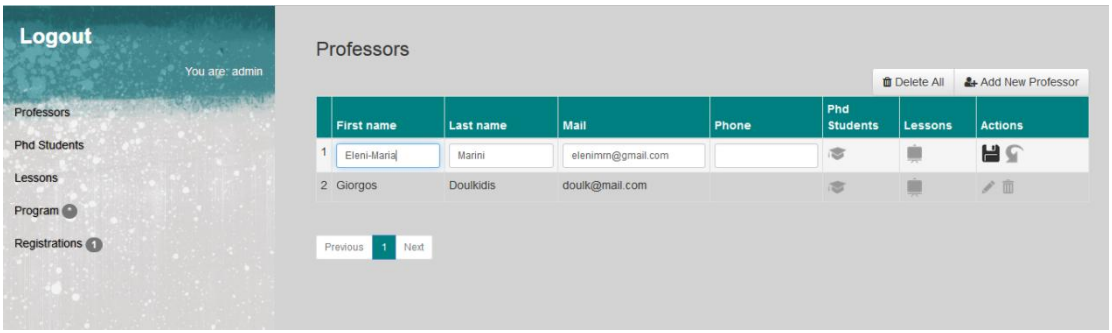


Εικόνα 41: Inline επεξεργασία καθηγητή

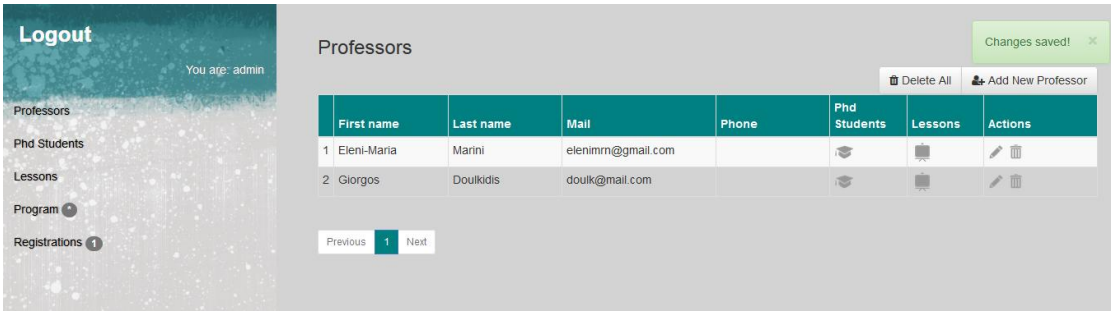
Εάν ακυρωθούν οι αλλαγές που έγιναν πατώντας το κουμπί ακύρωσης (εικόνα 44), τα στοιχεία του καθηγητή επιστρέφουν στην αρχική τους κατάσταση. Εάν ο διαχειριστής πατήσει το κουμπί αποθήκευσης (εικόνα 45) τότε ο καθηγητής σώζεται με τα νέα του στοιχεία στη βάση δεδομένων και ο διαχειριστής ενημερώνεται με κατάλληλο μήνυμα για την επιτυχημένη αποθήκευση (εικόνα 46).



Εικόνα 42: Ακύρωση της επεξεργασίας του καθηγητή



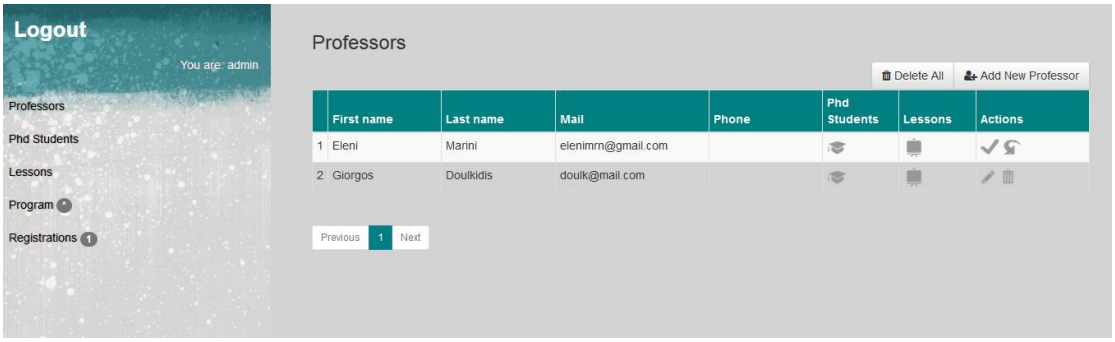
Εικόνα 43: Αποθήκευση αλλαγών



Εικόνα 44: Μήνυμα επιβεβαίωσης αλλαγών

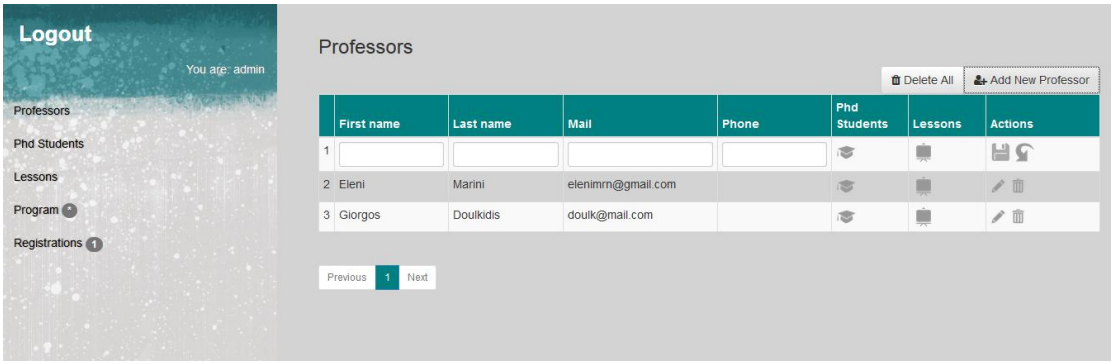
Εικόνα 45: Επιλογή για διαγραφή καθηγητή

Εάν ο διαχειριστής επιλέξει να διαγράψει έναν καθηγητή τότε για λόγους ασφαλείας, αυτό δε γίνεται άμεσα, αλλά εμφανίζονται πρώτα δύο κουμπιά μέσω των οποίων ο επιβεβαιώνεται ή ακυρώνεται η διαγραφή του επιλεγμένου καθηγητή. Το κουμπί επιβεβαίωσης είναι το ✓ ενώ το κουμπί ακύρωσης είναι το ↶ (εικόνα 48).



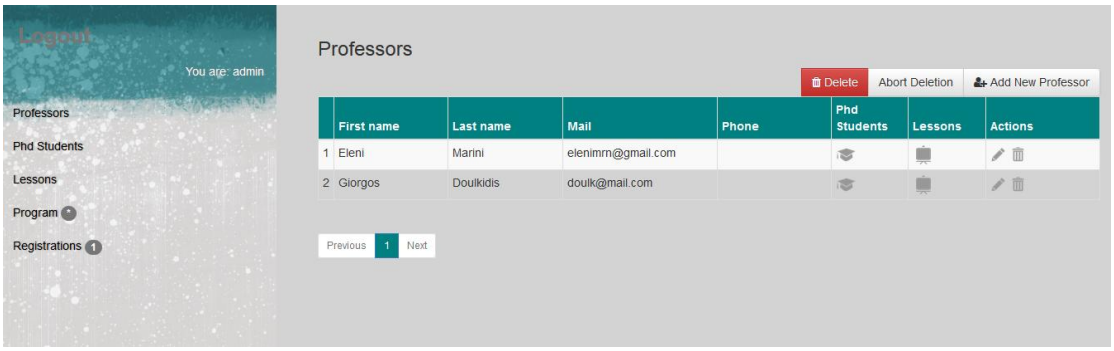
Εικόνα 46:Επιλογές επιβεβαίωσης και ακύρωσης διαγραφής

Ο διαχειριστής έχει τη δυνατότητα να δημιουργήσει νέους καθηγητές πατώντας το κουμπί "*Add New Professor*" που βρίσκεται πάνω από τον πίνακα. Όταν γίνει αυτό μια νέα γραμμή προστίθεται στον πίνακα των καθηγητών με πεδία έτοιμα προς συμπλήρωση. Πατώντας το κουμπί αποθήκευσης , ο νέος καθηγητής σώζεται στη βάση. Αντίθετα αν πατηθεί το κουμπί ακύρωσης , η γραμμή που προστέθηκε στον πίνακα αφαιρείται και δεν δημιουργείται ο καθηγητής (εικόνα 49).



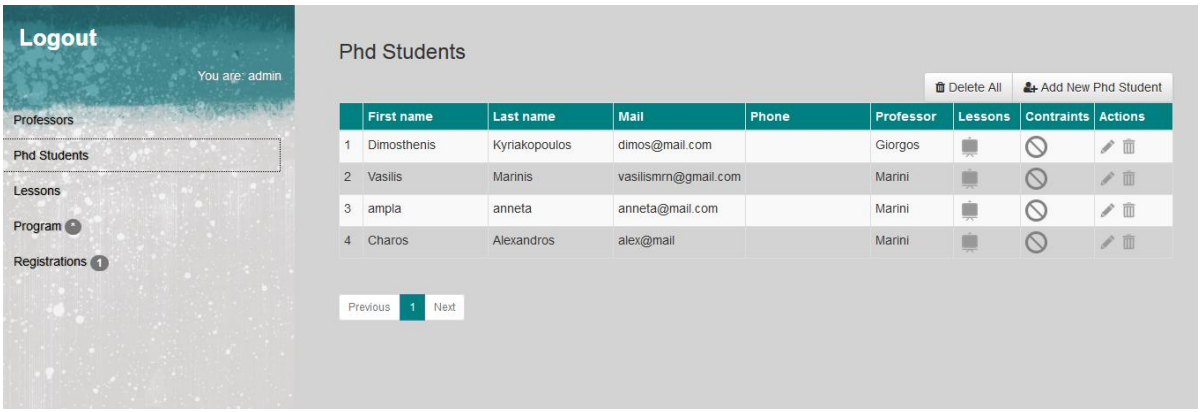
Εικόνα 47: Δημιουργία νέου καθηγητή

Ακόμα, υπάρχει η δυνατότητα να διαγραφούν όλοι οι καθηγητές , πατώντας το κουμπί "*Delete All*". Ωστόσο για λόγους ασφαλείας, εμφανίζονται δυο νέα κουμπιά ("*Delete All*" και "*Abort Deletion*"). Για να ολοκληρωθεί η διαγραφή, θα πρέπει να πατήσουμε στο κουμπί "*Delete*". Αν ο διαχειριστής επιθυμεί να ακυρώσει την διαγραφή αρκεί να πατήσει στο κουμπί "*Abort Deletion*" (εικόνα 50).

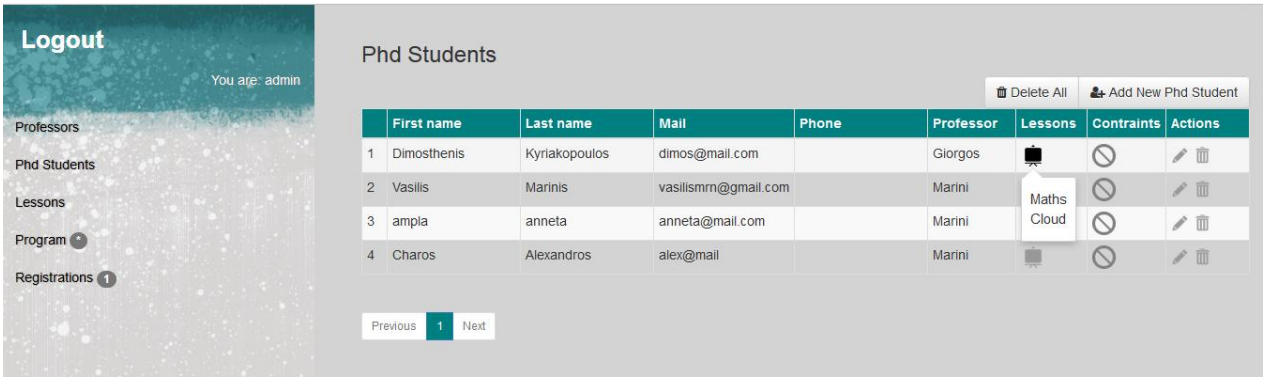


Εικόνα 48: Κουμπιά επιβεβαίωσης και ακύρωσης διαγραφής όλως των καθηγητών

Επιλέγοντας από το μενού, "*Phd Students*", ανοίγει η σελίδα διαχείρισης των υποψηφίων διδακτόρων, στην οποία ο διαχειριστής μπορεί να δει όλους τους υποψήφιους διδάκτορες συγκεντρωμένους σε έναν πίνακα (εικόνα 51). Για κάθε υποψήφιο διδάκτορα, στον πίνακα, φαίνονται άμεσα τα προσωπικά του στοιχεία, όνομα, επίθετο, email, τηλέφωνο, αλλά και ο καθηγητής ο οποίος τον επιβλέπει. Επιπλέον υπάρχουν 3 ακόμα στήλες με εικονίδια. Όταν κάνουμε "κλικ" με το ποντίκι στο εικονίδιο στη στήλη "*Lessons*" που ανήκει σε έναν υποψήφιο διδάκτορα, τότε εμφανίζεται ένα λευκό κουτάκι που περιέχει όλα τα μαθήματα του συγκεκριμένου υποψήφιο διδάκτορα (εικόνα 52).

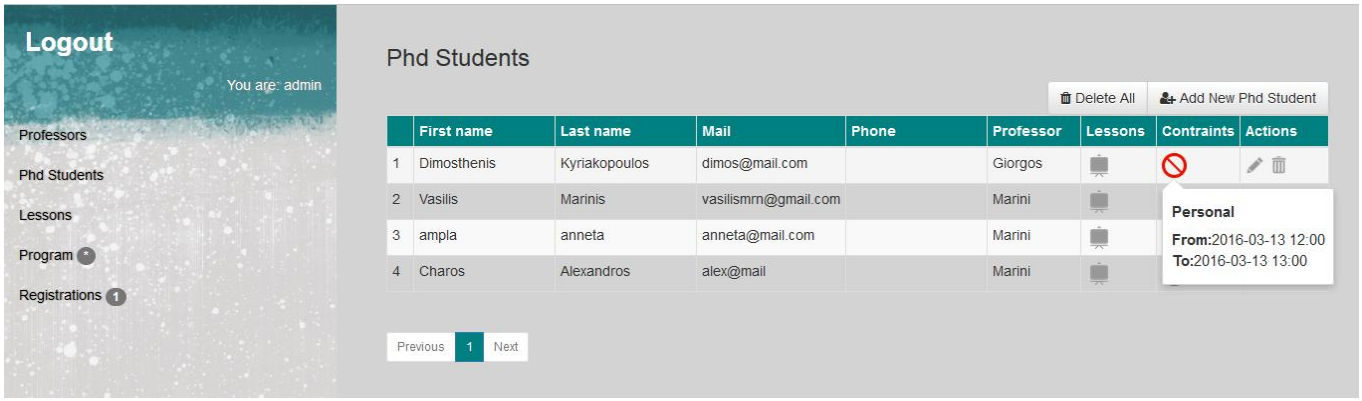


Εικόνα 49: Υποσελίδα υποψηφίων διδακτορικών



Εικόνα 50: Μαθήματα υποψηφίου διδάκτορα

Αντίστοιχα όταν κάνουμε "κλικ" με το ποντίκι στο εικονίδιο ⃠ στη στήλη "*Constraints*" που ανήκει σε έναν υποψήφιο διδάκτορα, τότε στο λευκό κουτάκι εμφανίζονται όλοι οι προσωπικοί περιορισμοί του συγκεκριμένου υποψήφιου διδάκτορα (εικόνα 53). Η διαδικασία αποθήκευσης και διαγραφής είναι όμοια με αυτή που περιγράφηκε στην σελίδα των καθηγητών.

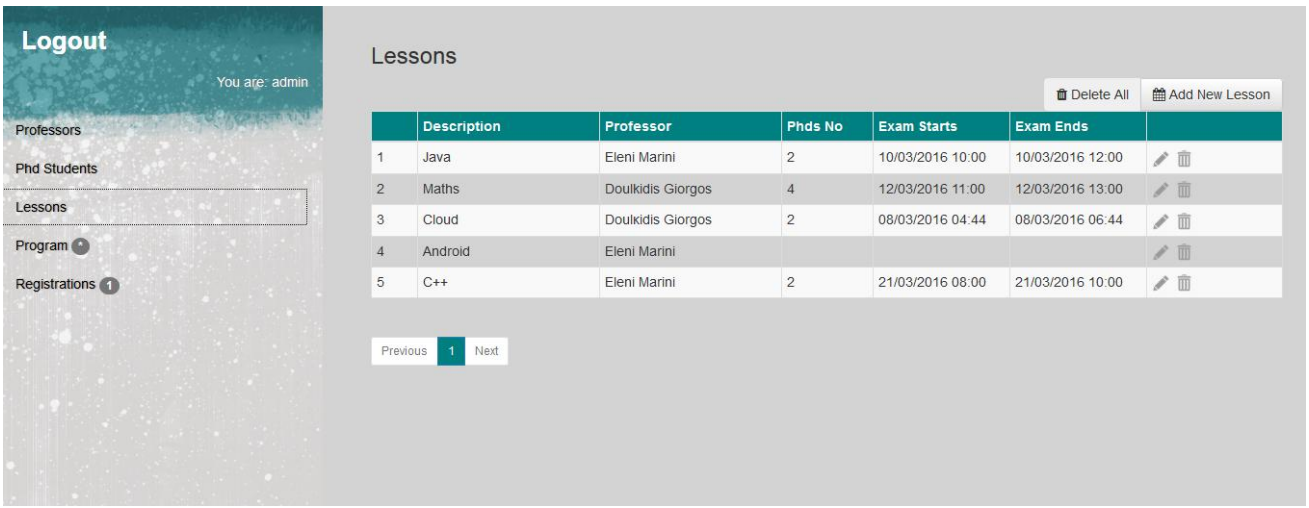


Εικόνα 51:Προσωπικοί περιορισμοί υποψηφίου διδακτορικού

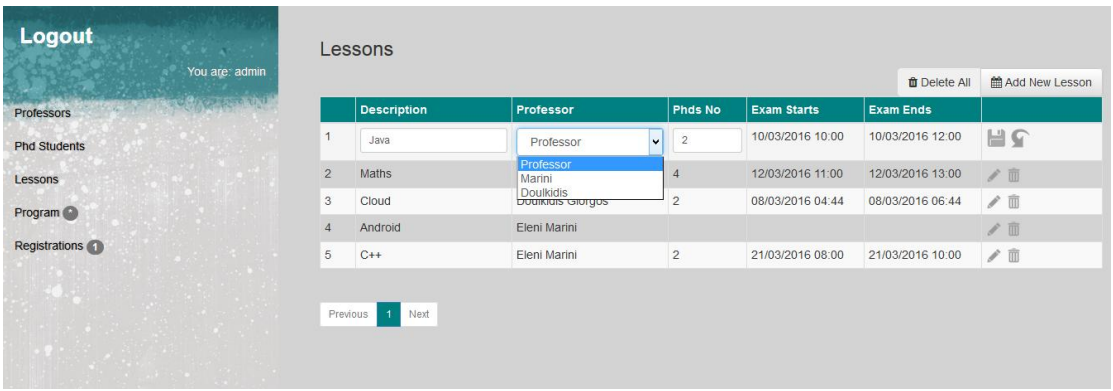
Ο διαχειριστής έχει τη δυνατότητα να δημιουργήσει νέους υποψήφιους διδάκτορες πατώντας το κουμπί "*Add New Phd Student*" αλλά και να του διαγράψει όλους ακολουθώντας την ίδια διαδικασία που περιγράφηκε στην σελίδα των καθηγητών.

Επιλέγοντας από το μενού, "*Lessons*", ανοίγει η σελίδα διαχείρισης των μαθημάτων, στην οποία ο διαχειριστής μπορεί να δει όλα τα μαθήματα συγκεντρωμένα σε έναν πίνακα (εικόνα 54). Για κάθε μάθημα, στον πίνακα,

φαίνονται η περιγραφή του, το όνομα του καθηγητή που το διδάσκει, ο αριθμός των υποψηφίων διδακτόρων που απαιτούνται για την επιτήρηση του μαθήματος αλλά και οι ημερομηνίες έναρξης και λήξης της εξέτασής του. Στην τελευταία στήλη, βλέπουμε τα γνωστά πλέον εικονίδια επεξεργασίας και διαγραφής. Κατά την επεξεργασία των μαθημάτων, ο διαχειριστής επιτρέπεται να τροποποιήσει μόνο την περιγραφή του μαθήματος, τον αριθμό των απαιτούμενων επιτηρητών, αλλά και τον καθηγητή που το διδάσκει. Η επιλογή του καθηγητή γίνεται από ένα dropdown μενού, το οποίο περιέχει όλους τους καθηγητές του συστήματος (εικόνα 55).

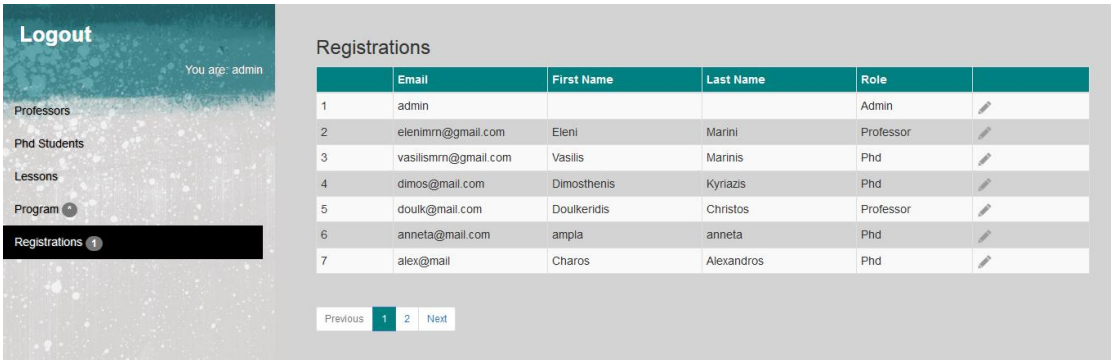


Εικόνα 52: Υποσελίδα μαθημάτων

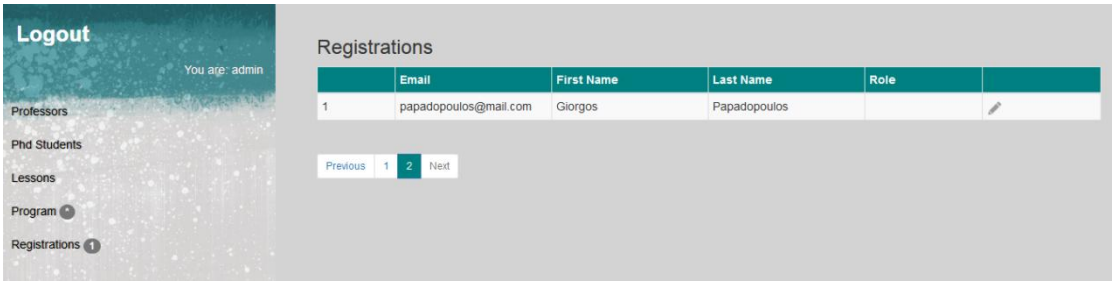


Εικόνα 53: Inline επεξεργασία μαθήματος-επιλογή καθηγητή μαθήματος

Επιλέγοντας από το μενού, "*Registrations*", ανοίγει η σελίδα διαχείρισης των χρηστών του συστήματος, στην οποία ο διαχειριστής μπορεί να δει όλους τους χρήστες συγκεντρωμένους σε έναν πίνακα (εικόνα 56). Για κάθε χρήστη, στον πίνακα, φαίνονται το email του, το όνομα, το επίθετό του, αλλά και ο ρόλος του. Στην τελευταία στήλη, βλέπουμε το εικονίδιο επεξεργασίας . Κατά την επεξεργασία των χρηστών, ο διαχειριστής επιτρέπεται να τροποποιήσει μόνο ρόλο του χρήστη. Οι διαθέσιμοι ρόλοι είναι Professor και Phd Student. Η επιλογή του ρόλου γίνεται από ένα dropdown μενού, το οποίο περιέχει όλους τους ρόλους (εικόνα 58).



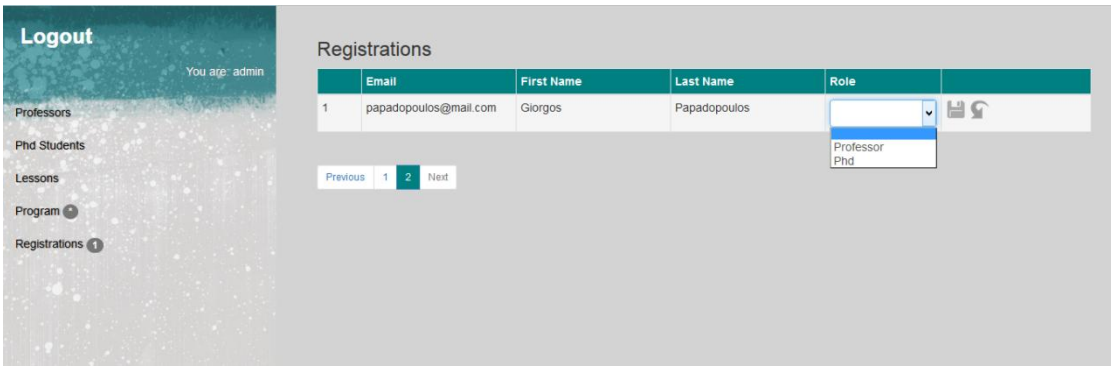
Εικόνα 54: Υποσελίδα χρηστών συστήματος - ειδοποίηση νέου χρήστη



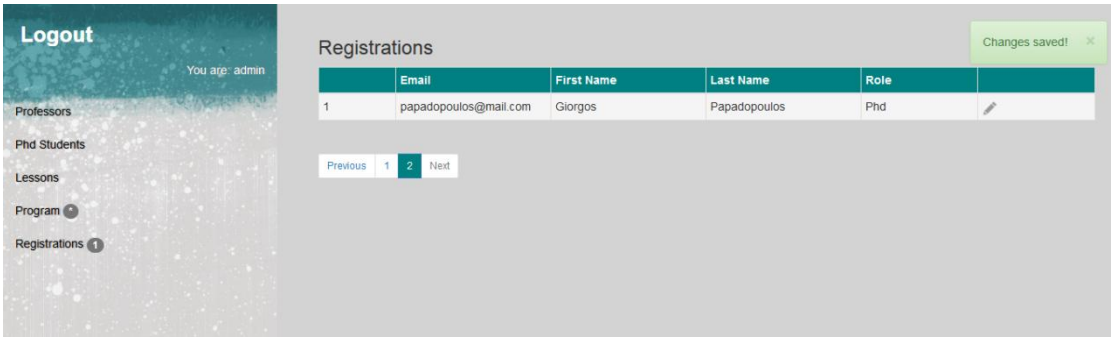
Εικόνα 55: Δεύτερη σελίδα χρηστών, νέος χρήστης

Στη σελίδα αυτή μπορούμε να δούμε και τη σελιδοποίηση του πίνακα. Όπως βλέπουμε στην εικόνα 57, οι χρήστες του πίνακα έχουν κατανεμηθεί σε δύο σελίδες. Επιπλέον δίπλα από την επιλογή "*Registrations*" στο μενού περιήγησης, διακρίνεται

ένα bubble notification το οποίο ενημερώνει τον διαχειριστή ότι έχει δημιουργηθεί ένας νέος λογαριασμός στον οποίο δεν έχει ανατεθεί ρόλος.

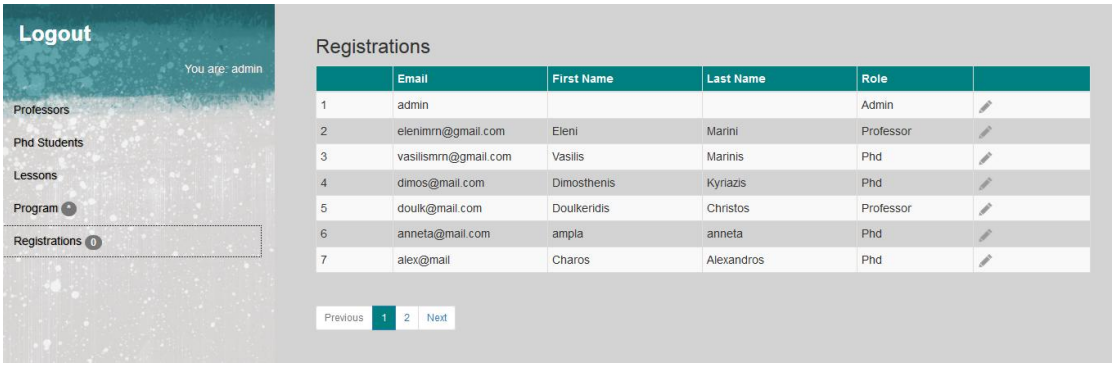


Εικόνα 56: Επιλογή ρόλου νέου χρήστη



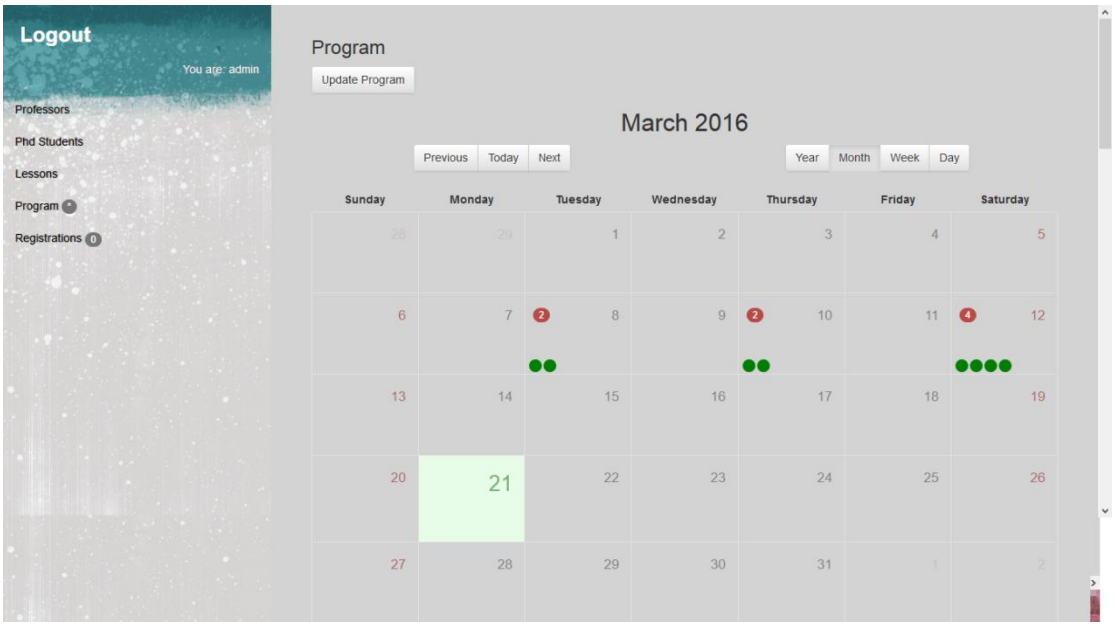
Εικόνα 57: Αποθήκευση αλλαγών

Αφού ο διαχειριστής επιλέξει ρόλο για το νέο χρήστη και αποθηκεύσει τις επιλογές του, ο αριθμός των notifications μειώνεται (εικόνα 60).



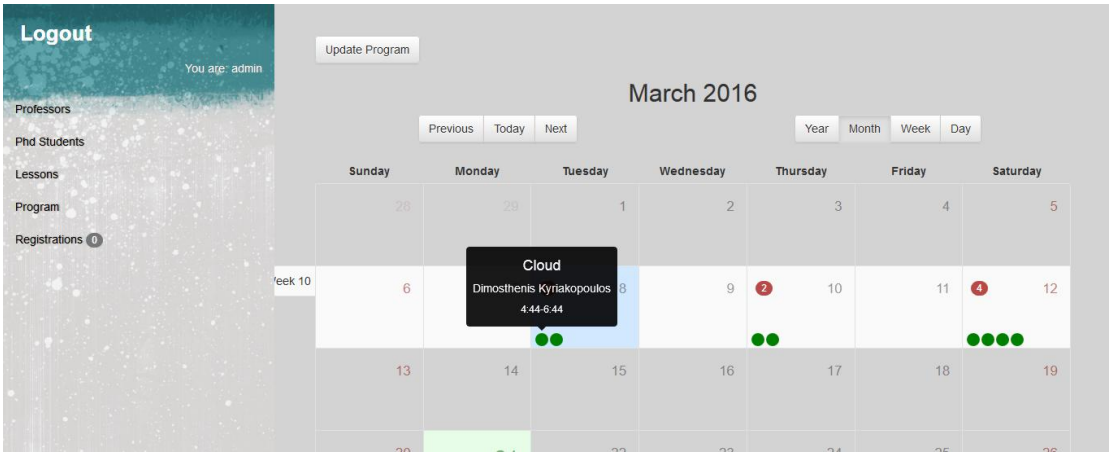
Εικόνα 58: Αλλαγή στο notification bubble

Επιλέγοντας από το μενού πλοήγησης, "*Program*", ανοίγει η σελίδα διαχείρισης και δημιουργίας του προγράμματος επιτηρήσεων, στην οποία ο διαχειριστής μπορεί να δημιουργήσει και να τροποποιήσει το πρόγραμμα των επιτηρήσεων, το οποίο παρουσιάζεται με μορφή ημερολογίου και οι επιτηρητές αποτελούν ξεχωριστά ημερολογιακά συμβάντα (εικόνα 61). Κάθε κουκίδα υποδεικνύει την επιτήρηση ενός υποψηφίου διδάκτορα. Επιπλέον σε κάθε ημερομηνία υπάρχει υπόδειξη για το πλήθος των επιτηρήσεων. Για να δημιουργήσει η να ανανεώσει το πρόγραμμα αρκεί να πατήσει το κουμπί "*Update Program*".



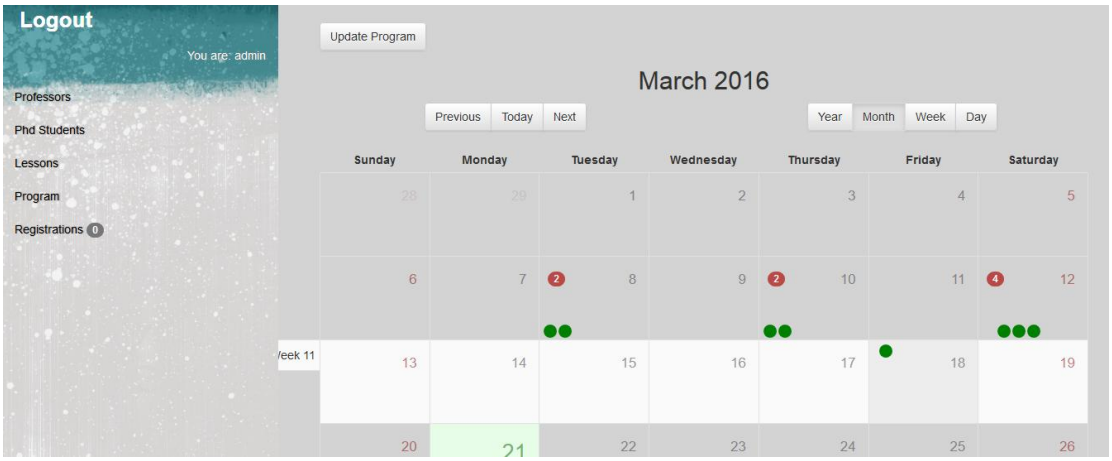
Εικόνα 59: Πρόγραμμα επιτηρήσεων

Για να δούμε περισσότερες πληροφορίες για κάθε επιτήρηση αρκεί να περάσουμε το ποντίκι πάνω από την κουκίδα που επιθυμούμε. Οι πληροφορίες που εμφανίζονται με τη μορφή tooltip είναι το μάθημα που εξετάζεται, η ώρα εξέτασής του, αλλά και ο υποψήφιος διδάκτορας που αντιστοιχεί στην κουκίδα. Για παράδειγμα στην εικόνα 56 πληροφορούμαστε ότι ο Δημοσθένης Κυριακόπουλος επιτηρεί το μάθημα Java στις 4:44 έως 6:44 στις 8 Μαρτίου 2016 (εικόνα 62).



Εικόνα 60: Tooltip επιτήρησης

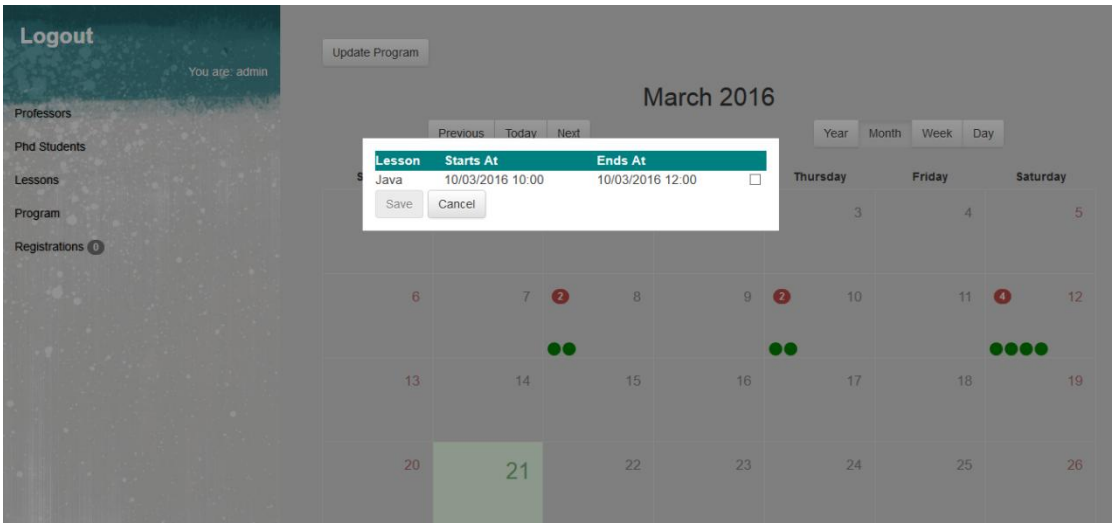
Ο διαχειριστής έχει τη δυνατότητα να τροποποιήσει το πρόγραμμα επιτηρήσεων μετακινώντας έναν επιτηρητή σε κάποιο άλλο μάθημα. Η μετακίνηση αυτή πραγματοποιείται "τραβώντας" την κουκίδα που αντιστοιχεί στον επιτηρητή που επιθυμεί να μετακινήσει, "σέρνοντάς" την και "αφήνοντάς'" την στο κουτί της ημερομηνίας που επιθυμεί (λειτουργία drag and drop - εικόνα 63).



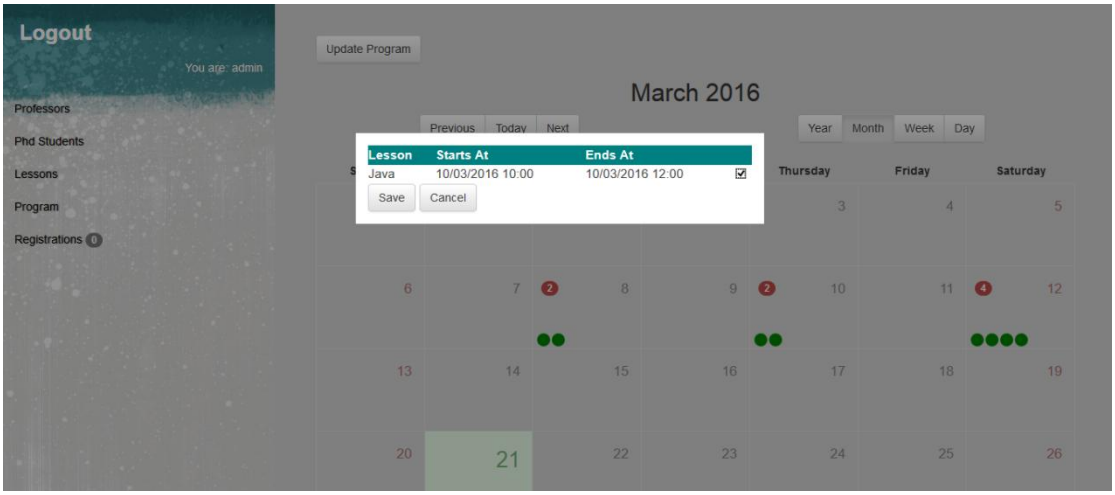
Εικόνα 61: Μετακίνηση επιτηρητή

Όταν η κουκίδα του επιτηρητή αφεθεί ελεύθερη, εμφανίζεται ένα παράθυρο με τα μαθήματα που εξετάζονται την ημερομηνία στην οποία μεταφέρθηκε ο επιτηρητής (εικόνα 64). Από αυτά τα μαθήματα καλείται να επιλέξει ο διαχειριστής σε ποιο επιθυμεί τελικά να τον μετακινήσει. Αφού επιλέξει ένα μάθημα , ενεργοποιείται το κουμπί "*Save*" και δεν μπορεί να επιλέξει δεύτερο μάθημα (εικόνα 65). Αν επιθυμεί να επιλέξει διαφορετικό μάθημα, τότε πρέπει να απο-επιλέξει το επιλεγμένο μάθημα

και έπειτα να επιλέξει κάποιο άλλο. Ωστόσο εάν επιθυμεί να ακυρώσει την μετακίνηση, κάνει κλικ στο κουμπί "*Cancel*" και η κουκίδα που αντιστοιχεί στον επιτηρητή παραμένει στην θέση της. Εάν τελικά ολοκληρωθεί η μετακίνηση του επιτηρητή, η κουκίδα μετακινείται στο κουτί της ημερομηνίας που επιλέχθηκε. Κατά τη μετακίνηση ενός επιτηρητή, εάν δεν υπάρχει εξεταζόμενο μάθημα στη νέα ημερομηνία, ή αν ο υποψήφιος διδάκτορας έχει θέσει χρονικούς περιορισμούς που δεν επιτρέπουν την μετακίνηση αυτή, τότε ο διαχειριστής ειδοποιείται με κατάλληλο μήνυμα (εικόνα 67).

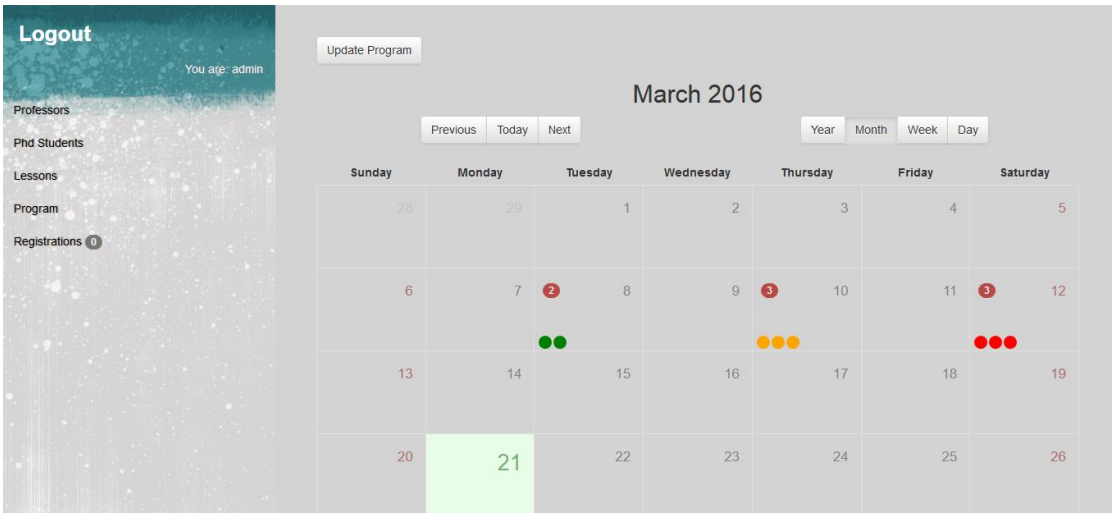


Εικόνα 62: Επιλογή μαθήματος για τον επιτηρητή στη νέα ημερομηνία

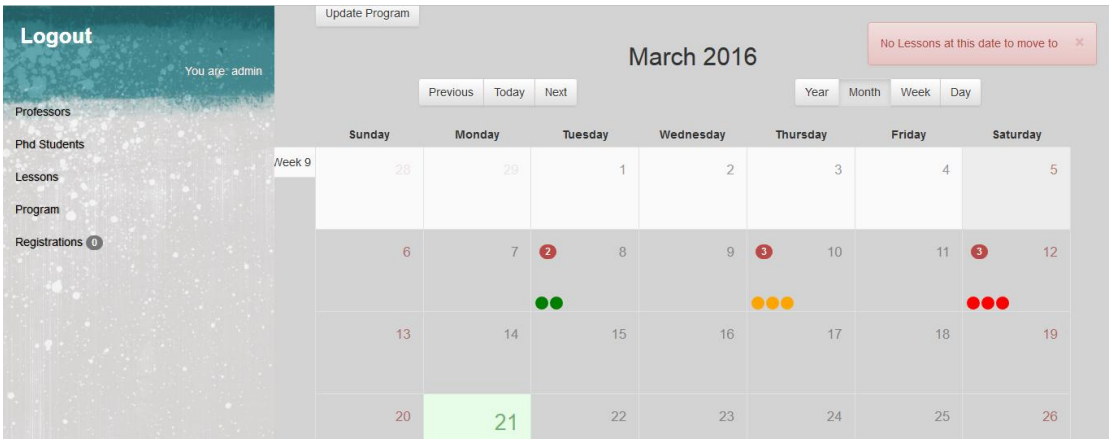


Εικόνα 63: Ενεργοποίηση κουμπιού save

Ο χρωματισμός των κουκίδων, βοηθά τον διαχειριστή να αντιληφθεί αν οι απαιτούμενες θέσεις επιτηρητών για το εξεταζόμενο μάθημα είναι καλυμμένες. Συγκεκριμένα εάν ο αριθμός των επιτηρητών είναι ακριβώς αυτός που απαιτείται, τότε το χρώμα των κουκίδων είναι πράσινο. Αντίθετα, εάν απαιτούνται περισσότεροι επιτηρητές από όσους έχουν ανατεθεί τότε οι κουκίδες είναι κόκκινες. Τέλος εάν σε ένα μάθημα έχουν ανατεθεί περισσότεροι επιτηρητές από όσους απαιτούνται τότε οι κουκίδες χρωματίζονται πορτοκαλί (εικόνα 66).

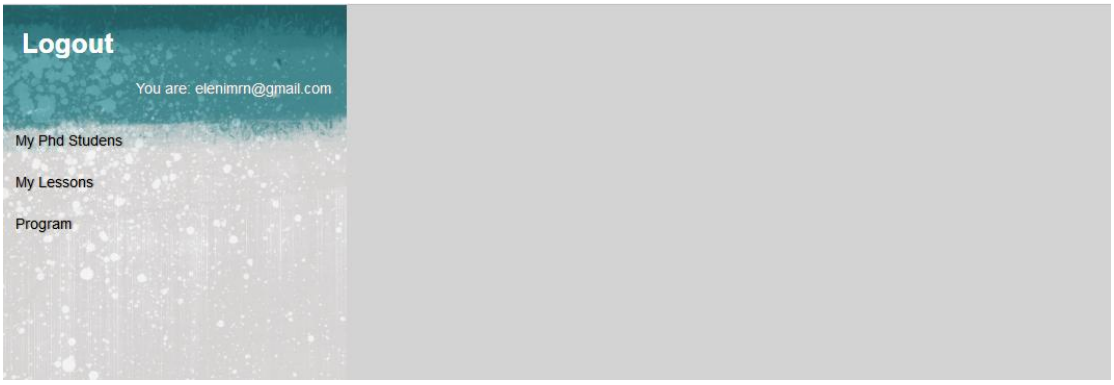


Εικόνα 64: Το πρόγραμμα μετά τη μετακίνηση του επιτηρητή



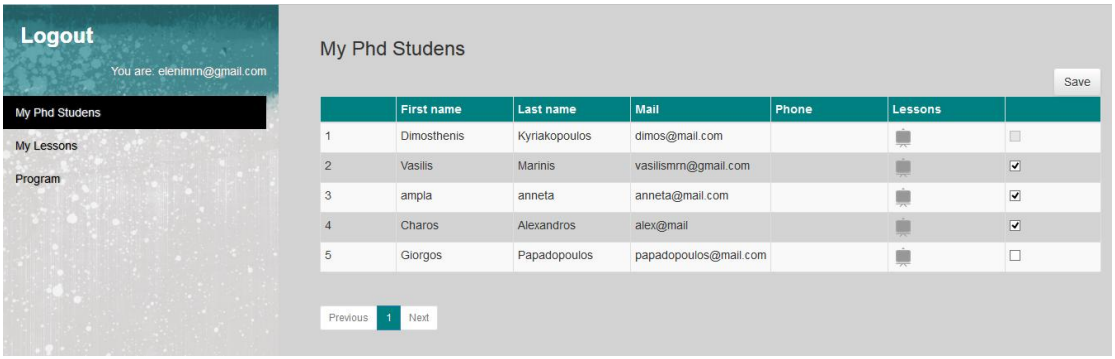
Εικόνα 65: Μη επιτρεπτή μετακίνηση

Μετά την επιτυχημένη είσοδο του καθηγητή στο σύστημα, αυτός οδηγείται στην αρχική σελίδα που αφορά τον δικό του ρόλο (εικόνα 68). Στη σελίδα αυτή το μενού περιήγησης στην αριστερή πλευρά οδηγεί τον καθηγητή στις υπόλοιπες σελίδες. Επιπλέον στο μενού αυτό υπάρχει και ένα κουμπί "*LogOut*" πατώντας το οποίο πραγματοποιείται αποσύνδεση του χρήστη από τη σελίδα. Οι διαθέσιμες επιλογές στο μενού περιήγησης είναι οι : *My Phd Students, My Lessons* και *Program.*

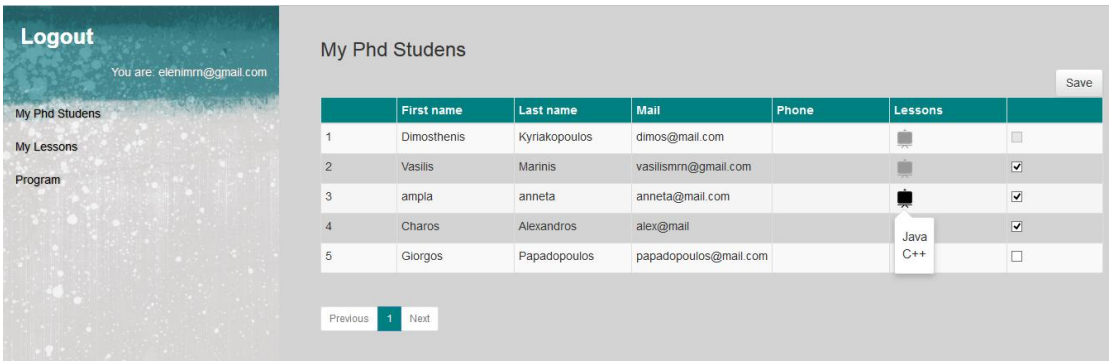


Εικόνα 66: Αρχική σελίδα καθηγητή

Επιλέγοντας από το μενού, " *My Phd Students* ", ανοίγει η σελίδα διαχείρισης των υποψήφιων διδακτόρων στην οποία ο καθηγητής μπορεί να δει όλους τους υποψήφιους διδάκτορες, που επιτηρεί, συγκεντρωμένους σε έναν πίνακα (εικόνα 69). Για κάθε υποψήφιο διδάκτορα, στον πίνακα, φαίνονται άμεσα τα προσωπικά του στοιχεία, όνομα, επίθετο, email, τηλέφωνο. Επιπλέον, υπάρχουν 2 ακόμα στήλες. Όταν κάνουμε "κλικ" με το ποντίκι στο εικονίδιο στη στήλη "*Lessons*" που ανήκει σε έναν υποψήφιο διδάκτορα, τότε εμφανίζεται ένα λευκό κουτάκι που περιέχει όλα τα μαθήματα του συγκεκριμένου υποψήφιου διδάκτορα (εικόνα 70). Στην τελευταία στήλη για κάθε υποψήφιο διδάκτορα, υπάρχει ένα κουτί επιλογής, το οποίο επιλέγει ο καθηγητής αν επιτηρεί τον συγκεκριμένο υποψήφιο διδάκτορα. Για να αποθηκεύσει τις επιλογές του αρκεί να πατήσει το κουμπί "*Save*" που βρίσκεται πάνω από τον συγκεντρωτικό πίνακα.



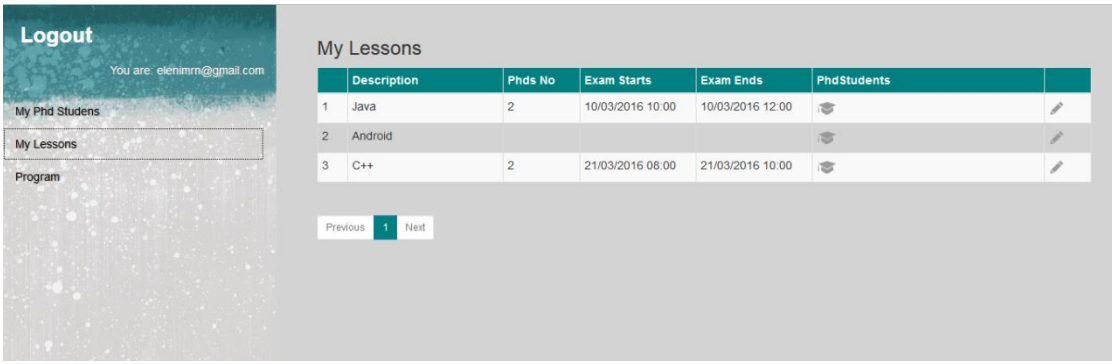
Εικόνα 67: Υποσελίδα υποψήφιων διδακτόρων



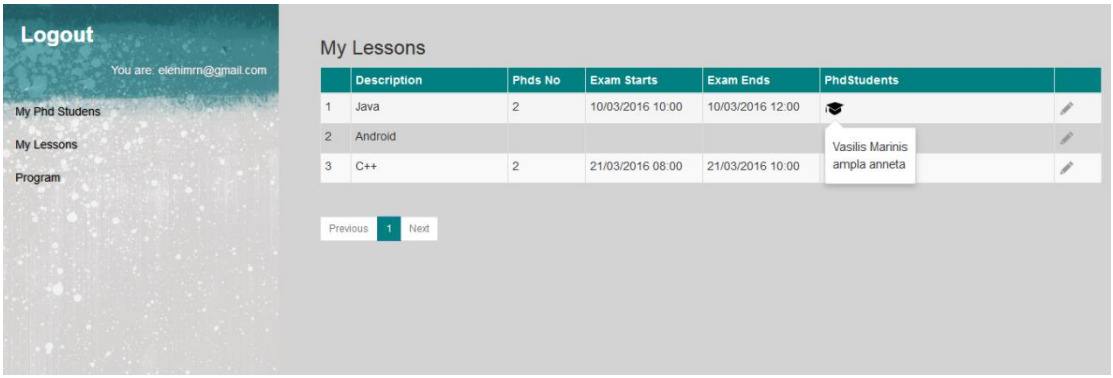
Εικόνα 68: Μαθήματα υποψηφίων διδακτόρων

Επιλέγοντας από το μενού, " *My Lessons* ", ανοίγει η σελίδα διαχείρισης των μαθημάτων στην οποία ο καθηγητής μπορεί να δει όλα τα μαθήματα που διδάσκει, συγκεντρωμένα σε έναν πίνακα (εικόνα 71). Για κάθε μάθημα, στον πίνακα, φαίνονται η περιγραφή του, το όνομα του καθηγητή που το διδάσκει, ο αριθμός των υποψηφίων διδακτόρων που απαιτούνται για την επιτήρηση του μαθήματος αλλά και οι ημερομηνίες έναρξης και λήξης της εξέτασής του. Όταν κάνουμε "κλικ" με το ποντίκι στο εικονίδιο στη στήλη "*Phd Students*" που ανήκει σε έναν μάθημα, τότε εμφανίζεται ένα λευκό κουτάκι που περιέχει όλους τους υποψήφιους διδακτορικούς που ανήκουν στο μάθημα αυτό (εικόνα 72).

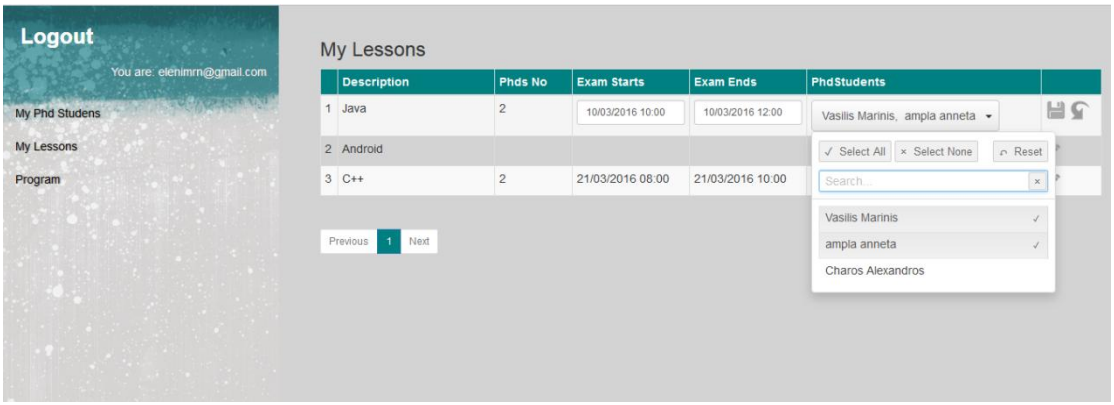
Στην τελευταία στήλη, βλέπουμε το εικονίδιο επεξεργασίας. Κατά την επεξεργασία των μαθημάτων, ο καθηγητής επιτρέπεται να τροποποιήσει μόνο τις ημερομηνίες έναρξης και λήξης της εξέτασής τους. Αυτό επιτυγχάνεται μέσα από ένα μενού πολλαπλής επιλογής το οποίο περιέχει όλους τους υποψήφιους διδάκτορες που επιτηρεί ο καθηγητής (εικόνα 67). Από το drop down αυτό μπορούν να επιλεγούν ταυτόχρονα, περισσότεροι από ένας υποψήφιοι διδάκτορες.



Εικόνα 69: Μαθήματα συνδεδεμένου καθηγητή

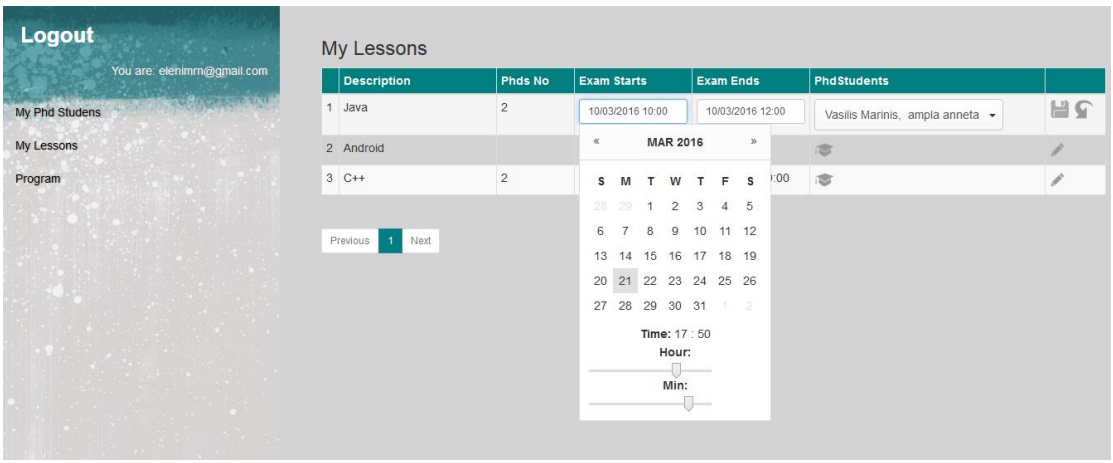


Εικόνα 70: Υποψήφιοι διδακτορικοί μαθημάτων



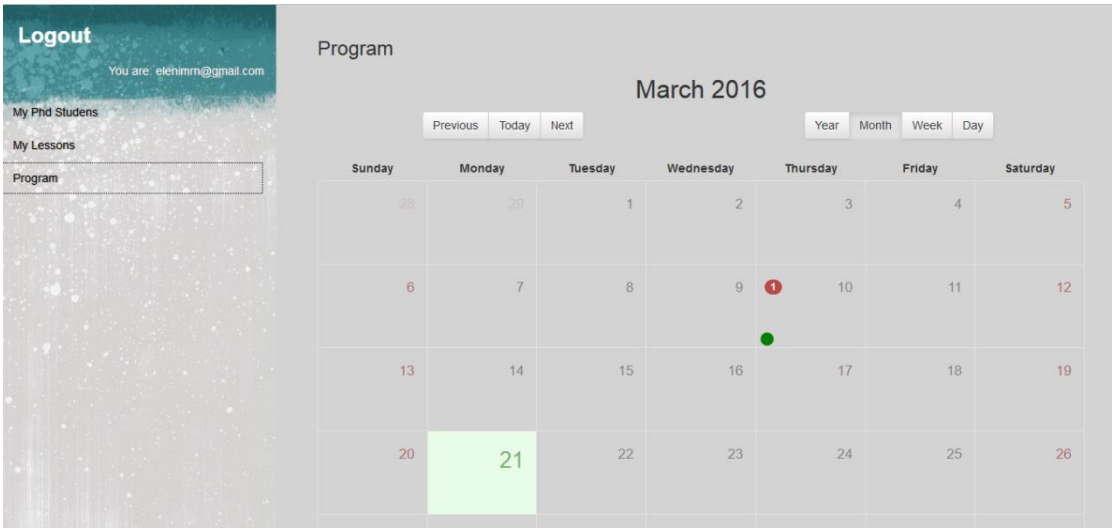
Εικόνα 71: Επεξεργασία μαθήματος-πολλαπλή επιλογή διδακτορικών

Για την εισαγωγή και την επεξεργασία των ημερομηνιών έναρξης και λήξης της εξέτασης του μαθήματος προσφέρονται date-time pickers , δηλαδή πεδία που επιτρέπουν την επιλογή τόσο ημερομηνίας όσο και ώρας (εικόνα 74).



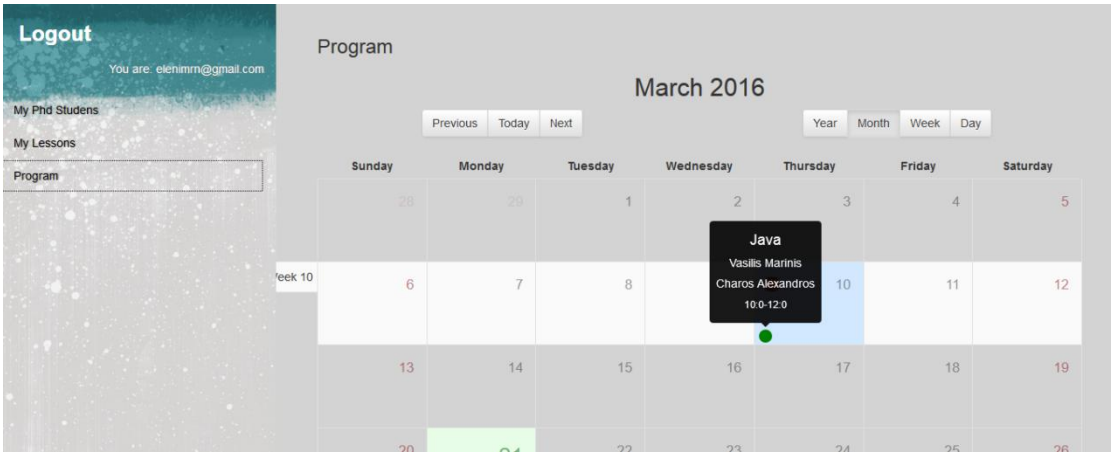
Εικόνα 72:Επεξεργασία μαθήματος-επιλογή ημερομηνίας εξέτασης

Επιλέγοντας από το μενού πλοήγησης, "*Program*", ανοίγει η σελίδα για την προβολή του προγράμματος επιτηρήσεων, στην οποία ο καθηγητής μπορεί να δει το πρόγραμμα των επιτηρήσεων των μαθημάτων που διδάσκει ο ίδιος, το οποίο παρουσιάζεται με μορφή ημερολογίου και κάθε εξεταζόμενο μάθημα αποτελεί ξεχωριστό ημερολογιακό συμβάν που παρουσιάζεται με τη μορφή μιας πράσινης κουκίδας (εικόνα 75). Επιπλέον σε κάθε ημερομηνία υπάρχει υπόδειξη για το πλήθος των μαθημάτων που εξετάζονται κατά τη διάρκεια της ημέρας.



Εικόνα 73: Πρόγραμμα επιτηρήσεων των μαθημάτων του συνδεδεμένου καθηγητή

Για την προβολή περισσότερων πληροφοριών για κάθε ημερολογιακό γεγονός κουκίδα, ο καθηγητής μπορεί να περάσει το ποντίκι πάνω από αυτό, ώστε να εμφανιστεί ένα πλαίσιο με το μάθημα που εξετάζεται, τους υποψήφιους διδάκτορες που το επιτηρούν, καθώς και την ώρα της εξέτασης του μαθήματος. Στο σημείο αυτό, αξίζει να σημειωθεί ότι ο καθηγητής δεν έχει τη δυνατότητα να επεξεργαστεί το πρόγραμμα των επιτηρήσεων.

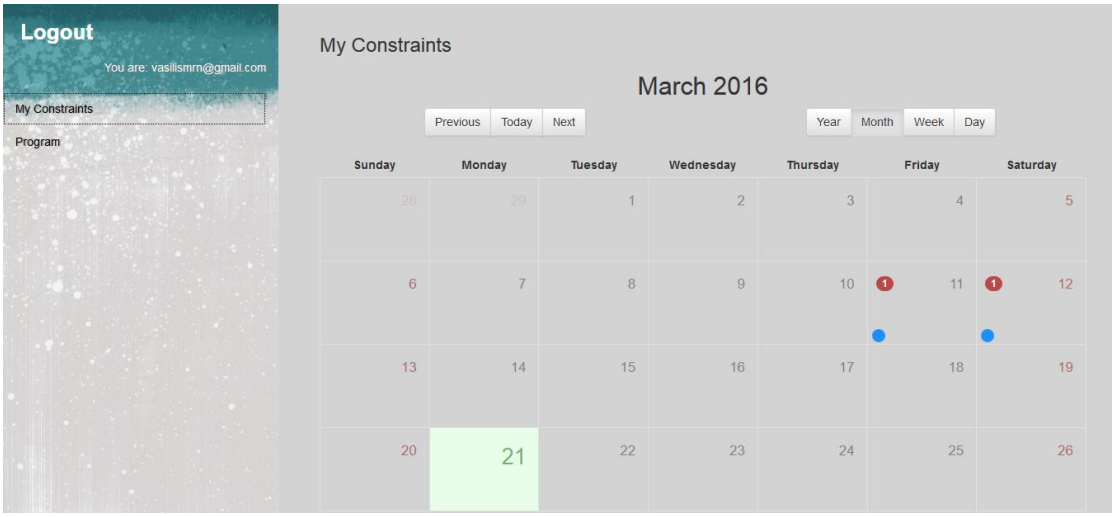


Εικόνα 74: Προβολή μαθήματος , ώρας εξέτασης και επιτηρητών, μέσω tooltip

Για το ρόλο του υποψήφιου διδάκτορα, παρέχονται διαφορετικές δυνατότητες. Μετά την επιτυχημένη είσοδο ενός χρήστη με ρόλο υποψήφιου διδάκτορα στο σύστημα, αυτός οδηγείται στην αρχική σελίδα που αφορά τον δικό του ρόλο (εικόνα 71). Στη σελίδα αυτή το μενού περιήγησης στην αριστερή πλευρά οδηγεί τον χρήστη στις υπόλοιπες σελίδες. Επιπλέον στο μενού αυτό υπάρχει και ένα κουμπί "*LogOut*" πατώντας το οποίο πραγματοποιείται αποσύνδεση του υποψήφιου διδάκτορα από το σύστημα. Οι διαθέσιμες επιλογές στο μενού περιήγησης είναι οι : *My Constraints* και *Program.*

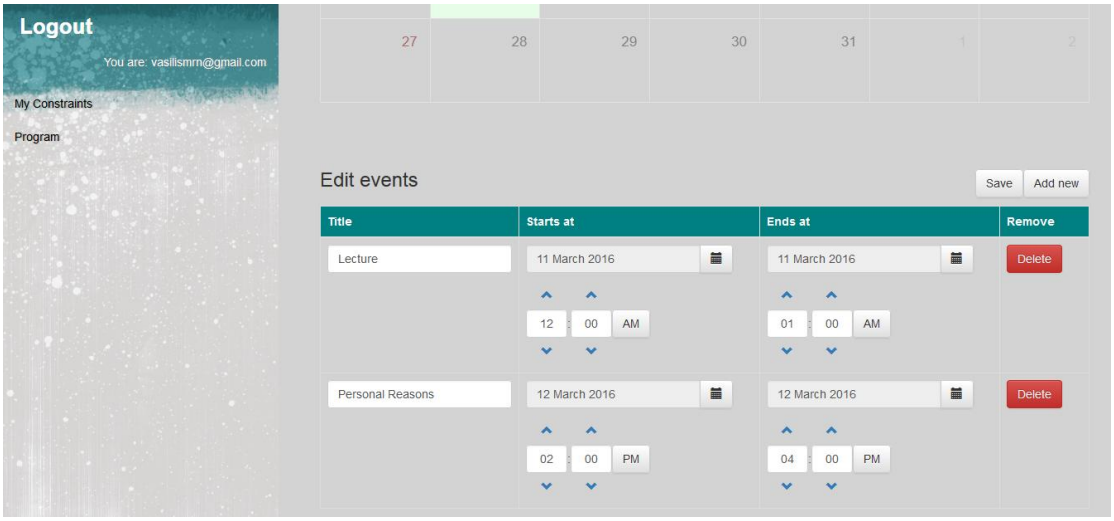
Εικόνα 75: Αρχική σελίδα για τον χρήστη-υποψήφιο διδακτορικό

Επιλέγοντας από το μενού πλοήγησης, "*My Constraints*", ανοίγει η σελίδα διαχείρισης των προσωπικών χρονικών περιορισμών που μπορεί να θέσει ο υποψήφιος διδάκτορας. Μέσα από τη σελίδα αυτή ο τελευταίος, μπορεί να αποθηκεύσει και να διαγράψει από το σύστημα όσους προσωπικούς χρονικούς περιορισμούς επιθυμεί, οι οποίοι στην συνέχεια θα ληφθούν υπόψη κατά την δημιουργία του προγράμματος επιτηρήσεων των εξεταζόμενων μαθημάτων. Στο επάνω μέρος της σελίδας παρουσιάζονται συγκεντρωτικά όλοι οι περιορισμοί που έχει θέσει ο χρήστης που έχει συνδεθεί σε μορφή ημερολογίου. Επίσης στο κάτω μέρος της ίδιας σελίδας, υπάρχουν πεδία μέσων των οποίων πραγματοποιείται η δημιουργία των περιορισμών. Για κάθε περιορισμό ο υποψήφιος διδάκτορας πρέπει να ορίσει την ημερομηνία και ώρα έναρξής του, αλλά και την ημερομηνία και ώρα λήξης του. Επιπλέον μπορεί να ορίσει μια περιγραφή ή επεξήγηση που αφορά τον περιορισμό αυτό (εικόνες 79, 80). Η εισαγωγή ημερομηνίας και ώρας έναρξης και λήξης πραγματοποιείται μέσω date - time picker πεδίων (εικόνα 80).

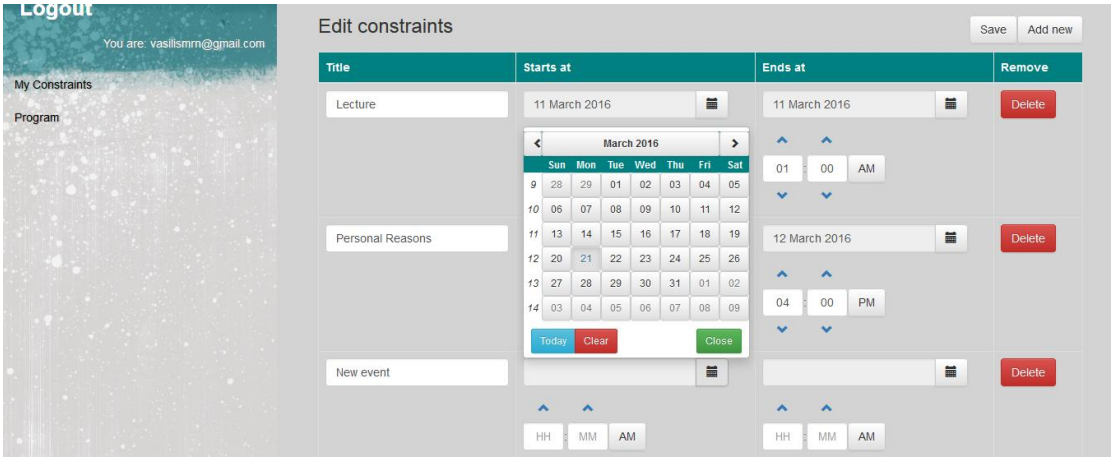


Εικόνα 76: Περιορισμοί υποψηφίου διδακτορικού

Για να ξεκινήσει η δημιουργία ενός νέου περιορισμού, αρκεί να πατηθεί το κουμπί "*Add New*" το οποίο ανοίγει τα πεδία που φαίνονται στις εικόνες 79 και 80. Για να αποθηκευτεί ο περιορισμός αρκεί να πατήσουμε το κουμπί "*Save*" μετά την εισαγωγή των απαραίτητων στοιχείων. Η διαγραφή ενός περιορισμού πραγματοποιείται μέσω του κουμπιού "*Delete*" που βρίσκεται στη δεξιά πλευρά των πεδίων εισαγωγής στοιχείων για κάθε περιορισμό (εικόνες 79 ,80).

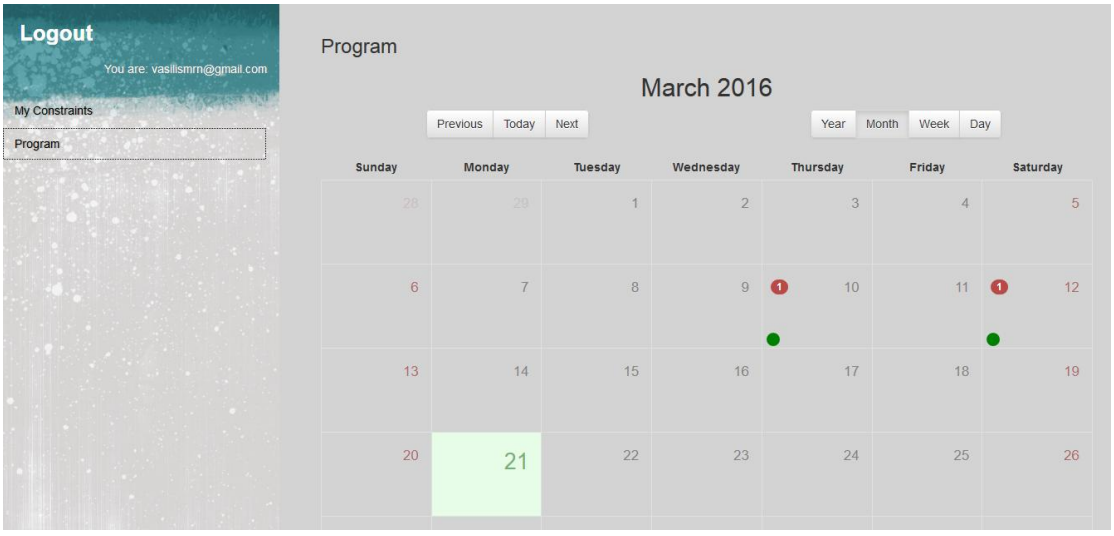


Εικόνα 77: Εισαγωγή και επεξεργασία περιορισμών



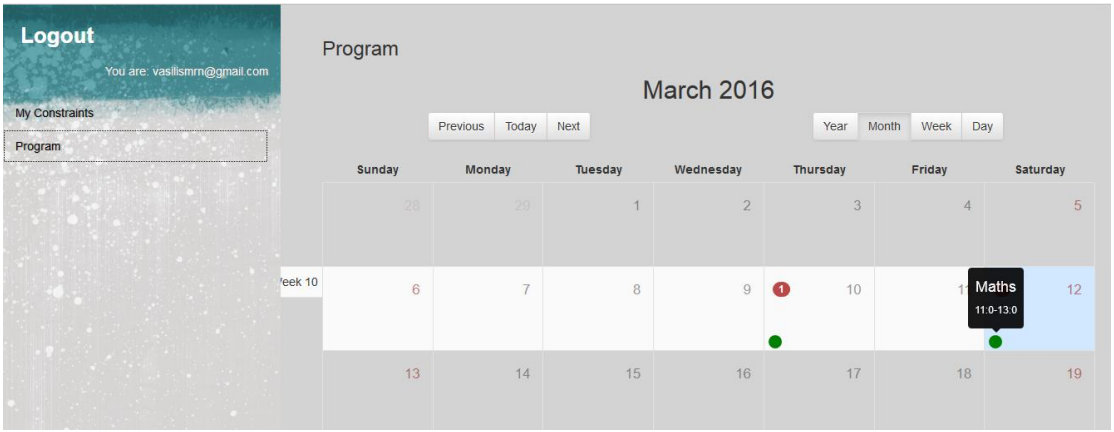
Εικόνα 78: Δημιουργία νέου περιορισμού-επιλογή ημερομηνίας και ώρας

Επιλέγοντας από το μενού πλοήγησης, "*Program*", ανοίγει η σελίδα για την προβολή του προγράμματος επιτηρήσεων, στην οποία ο υποψήφιος διδακτορικός μπορεί να δει το πρόγραμμα των επιτηρήσεων στις οποίες συμμετέχει ο ίδιος. Το πρόγραμμα παρουσιάζεται με μορφή ημερολογίου και κάθε μάθημα που επιτηρεί ο χρήστης αποτελεί ξεχωριστό ημερολογιακό συμβάν το οποίο παρουσιάζεται με τη μορφή μιας πράσινης κουκίδας. Επιπλέον σε κάθε ημερομηνία υπάρχει υπόδειξη για το πλήθος των μαθημάτων που επιτηρεί χρήστης κατά τη διάρκεια της ημέρας (εικόνα 81).



Εικόνα 79: Πρόγραμμα επιτηρήσεων υποψήφιου διδακτορικού

Για την προβολή περισσότερων πληροφοριών για κάθε επιτήρηση, ο υποψήφιος διδάκτορας μπορεί να περάσει το ποντίκι πάνω από την κουκίδα που την αντιπροσωπεύει, ώστε να εμφανιστεί ένα πλαίσιο με το μάθημα καθώς και την ώρα της εξέτασής του (εικόνα 82). Ο υποψήφιος διδάκτορας δεν έχει τη δυνατότητα να επεξεργαστεί το πρόγραμμα των επιτηρήσεων.



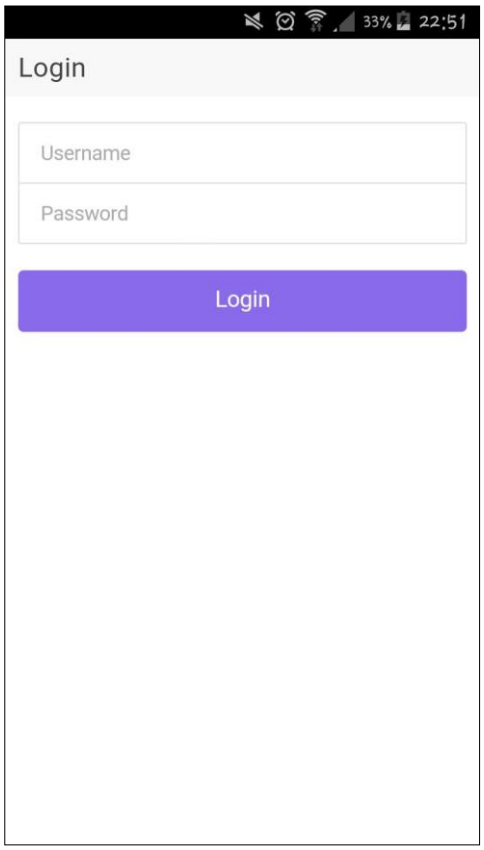
Εικόνα 80: Προβολή μαθήματος και ώρας εξέτασης προς επιτήρηση

7.2 Mobile εφαρμογή

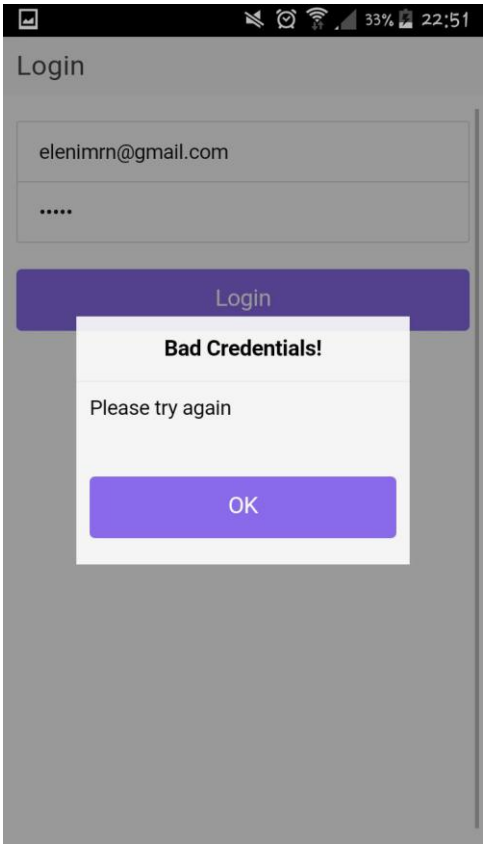
Στην συμπληρωματική εφαρμογή που αναπτύχθηκε για κινητές συσκευές, μπορούν να συνδεθούν μονάχα οι καθηγητές και οι υποψήφιοι διδάκτορες που έχουν δημιουργήσει λογαριασμό στην διαδικτυακή εφαρμογή.

Η αρχική οθόνη της mobile εφαρμογής (εικόνα 84) αποτελείται από μια φόρμα εισαγωγής email και κωδικού χρήστη, μέσω της οποίας πραγματοποιείται η είσοδος στην εφαρμογή. Στην περίπτωση που τα στοιχεία που εισάγει ο χρήστης είναι λανθασμένα, ένα κατάλληλο μήνυμα τον ενημερώνει (εικόνα 83).

Η εφαρμογή είναι διαφορετικής μορφής για κάθε έναν από τους δυο δυνατούς ρόλους (Καθηγητής, Υποψήφιος διδάκτορας). Και στους δύο ρόλους είναι διαθέσιμα δύο ξεχωριστά Tabs με τίτλους Program και QR codes, αλλά και ένα μικρό μενού, στην κάτω δεξιά πλευρά, το οποίο "επιπλέει" στην οθόνη (floating menu). Η διαφορά της εφαρμογής για τον κάθε ρόλο έγκειται στο περιεχόμενο των tabs και του μενού που προαναφέρθηκε.

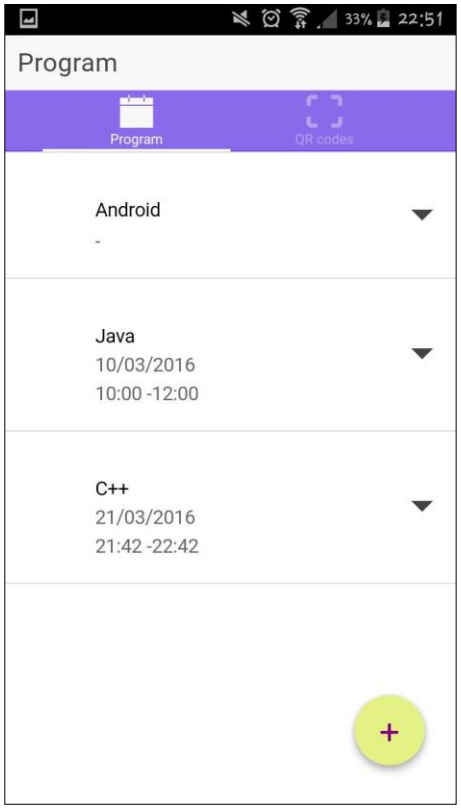


Εικόνα 82: Φόρμα login

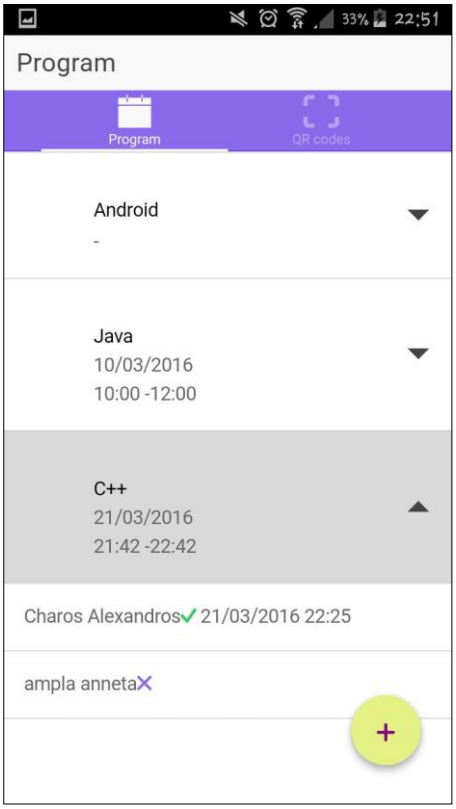


Εικόνα 81: Μήνυμα λανθασμένων στοιχείων σύνδεσης

Ο καθηγητής στο tab με τίτλο "*Program*" μπορεί να δει σε μορφή λίστας της ημερομηνίες και ώρες εξέτασης κάθε μαθήματος που διδάσκει (εικόνα 85). Επιπλέον πατώντας πάνω σε ένα στοιχείο - μάθημα της λίστας, εμφανίζονται κάτω από το επιλεχθέν στοιχείο οι επιτηρητές του αντίστοιχου μαθήματος σε μορφή λίστας. Δίπλα σε κάθε επιτηρητή εμφανίζεται ένα εικονίδιο το οποίο υποδεικνύει αν έχει παρουσιαστεί ο υποψήφιος διδάκτορας στην επιτήρηση του μαθήματος. Θα περιγράψουμε στη συνέχεια τη διαδικασία δήλωσης της παρουσίας των επιτηρητών στα εξεταζόμενα μαθήματα. Εάν ο επιτηρητής έχει προσέλθει μεταξύ των ωρών έναρξης και λήξης της εξέτασης το εικονίδιο που εμφανίζεται είναι το ✓ . Σε αντίθετη περίπτωση το εικονίδιο είναι της μορφής ✗ (εικόνα 86).

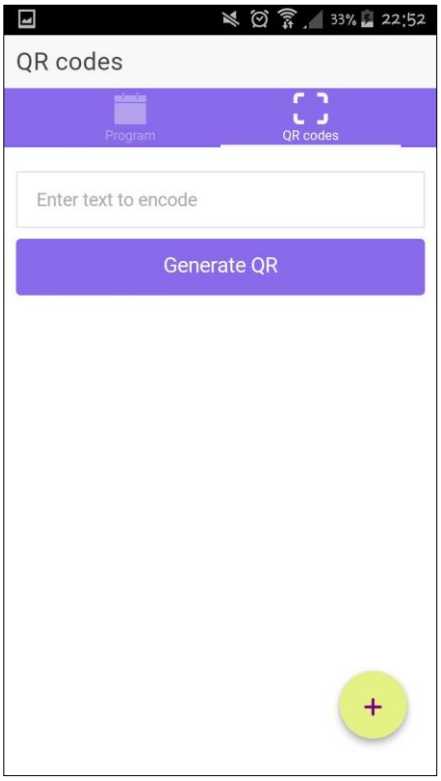


Εικόνα 83: Πρόγραμμα επιτηρήσεων (χρήστης-καθηγητής, tab Program)

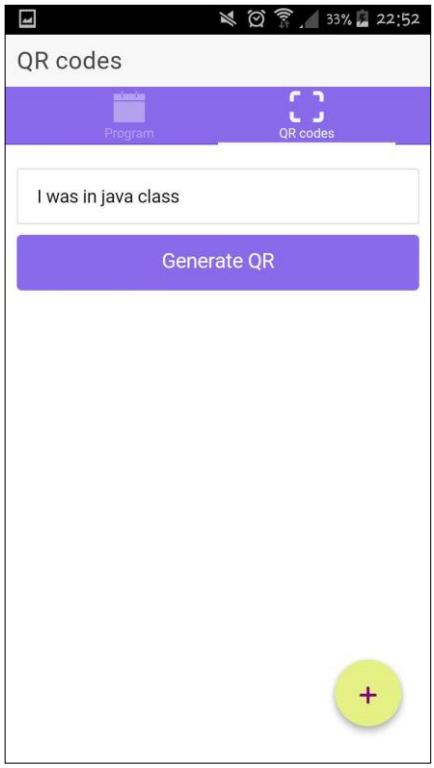


Εικόνα 84: επιτηρητές μαθήματος (χρήστης-καθηγητής, tab Program)

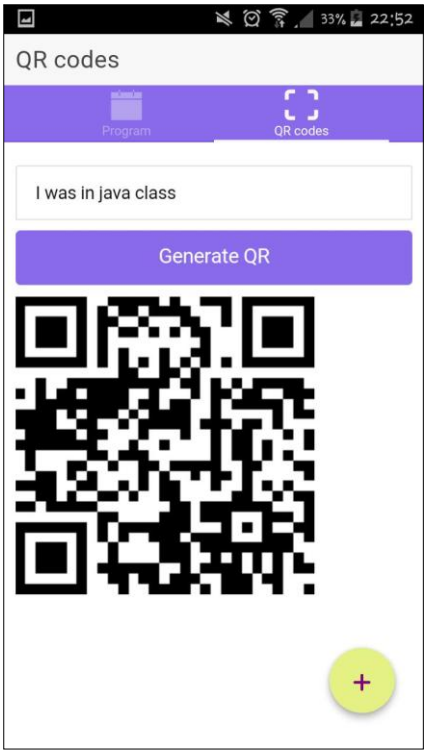
Η δήλωση της παρουσίας των επιτηρητών πραγματοποιείται μέσω του σκαναρίσματος ενός κωδικού τεχνολογίας QR. Ο καθηγητής στο tab με τίτλο "*QR codes*" έχει τη δυνατότητα να δημιουργήσει τέτοιους κωδικούς οι οποίοι προορίζονται για σκανάρισμα από τους επιτηρητές. Στο tab αυτό περιέχονται ένα πεδίο εισαγωγής κειμένου και ένα κουμπί με τίτλο "*Generate QR*" (εικόνα 88). Ο καθηγητής αφού εισάγει την λέξη ή φράση που επιθυμεί στο πεδίο εισαγωγής (εικόνα 89), μπορεί να κωδικοποιήσει το κείμενο που πληκτρολόγηση σε μορφή QR Κωδικού πατώντας το κουμπί "*Generate QR*". Τελικά εμφανίζεται κάτω από το κουμπί αυτό ένας κωδικός μορφής QR (εικόνα 89). Στην εικόνα 90 φαίνεται το floating μενού το οποίο για τον καθηγητή διαθέτει δύο κουμπιά. Όταν το μενού είναι κλειστό έχει τη μορφή , ενώ όταν ανοίξει εμφανίζονται τα 3 κουμπιά κάθετα και προς τα πάνω. Το κουμπί αποσυνδέει τον χρήστη από το σύστημα ,ενώ το τον οδηγεί στην διαδικτυακή εφαρμογή.



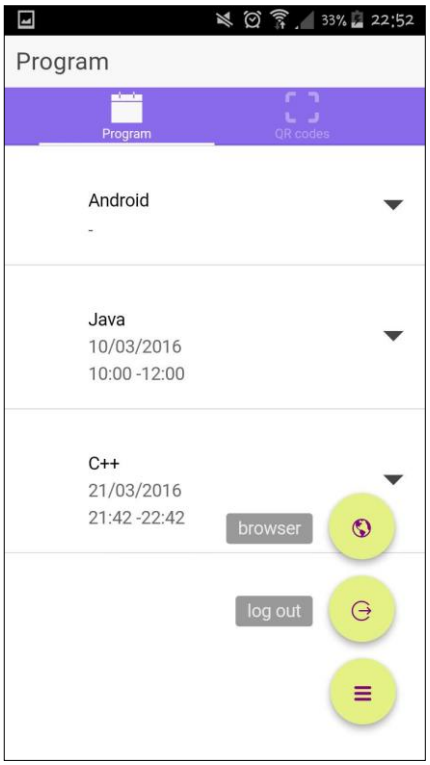
Εικόνα 86: Δημιουργία QR code (χρήστης-καθηγητής)



Εικόνα 85: Δημιουργία QR code 2

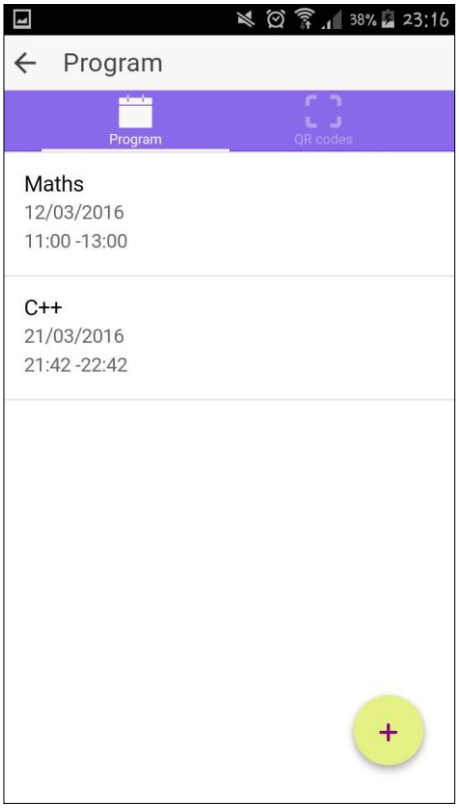


Εικόνα 87: Δημιουργία QR code 3

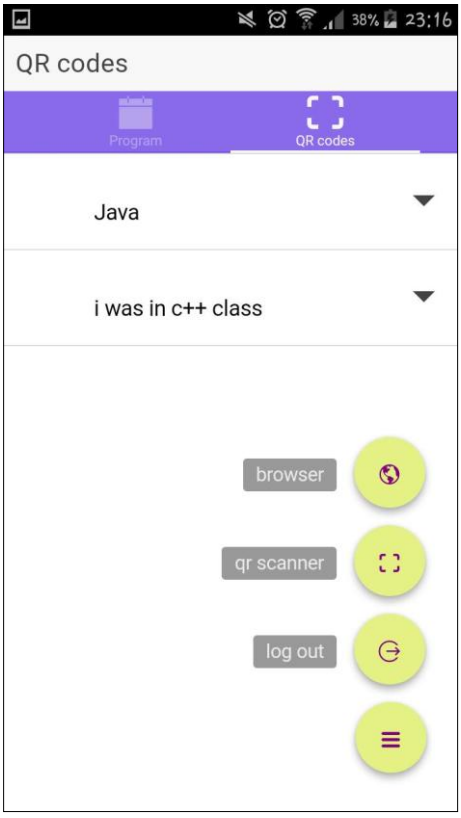


Εικόνα 88: float menu (χρήστης καθηγητής)

Ο καθηγητής στο tab με τίτλο "*Program*" μπορεί να δει σε μορφή λίστας της ημερομηνίες και ώρες εξέτασης κάθε μαθήματος που επιτηρεί (εικόνα 92) ενώ στο tab με τίτλο "*QR codes*" έχει τη δυνατότητα να δει συγκεντρωτικά σε μορφή λίστας όλους τους κωδικούς που έχει σκανάρει μέσω της εφαρμογής (εικόνα 91). Αν επιλέξουμε ένα στοιχείο - κωδικό της λίστας, τότε κάτω από αυτό εμφανίζεται το μάθημα με το οποίο είναι συνδεδεμένος ο κωδικός αυτός (εικόνα 93). Ωστόσο εάν για κάποιον κωδικό δεν έχουμε επιλέξει κανένα μάθημα τότε εμφανίζεται λίστα επιλογής μαθήματος (εικόνα 94). Για να αποθηκεύσουμε τις αλλαγές ή να διαγράψουμε έναν κωδικό αρκεί να σύρουμε του στοιχείο - κωδικό της λίστας προς τα αριστερά. Με την κίνησή μας αυτή εμφανίζονται δύο κουμπιά για την αποθήκευση ("*Save*") ή την διαγραφή του κωδικού ("*Delete*") (εικόνα 95). Το floating μενού έχει την ίδια λειτουργικότητα με αυτό του καθηγητή, ωστόσο περιέχει ένα επιπλέον κουμπί το οποίο παρέχει τη δυνατότητα στον υποψήφιο διδάκτορα να σκανάρει έναν qr κωδικό (εικόνα 96).

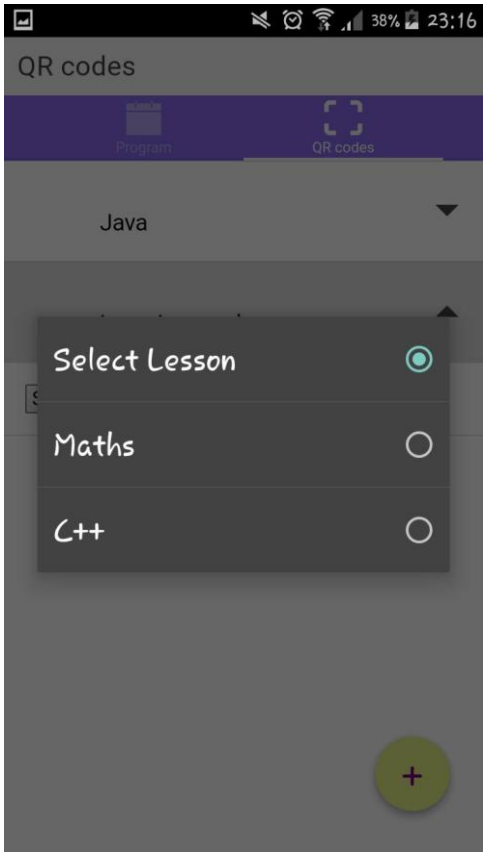


Εικόνα 90: Πρόγραμμα (χρήστης-υποψήφιος διδακτορικός)

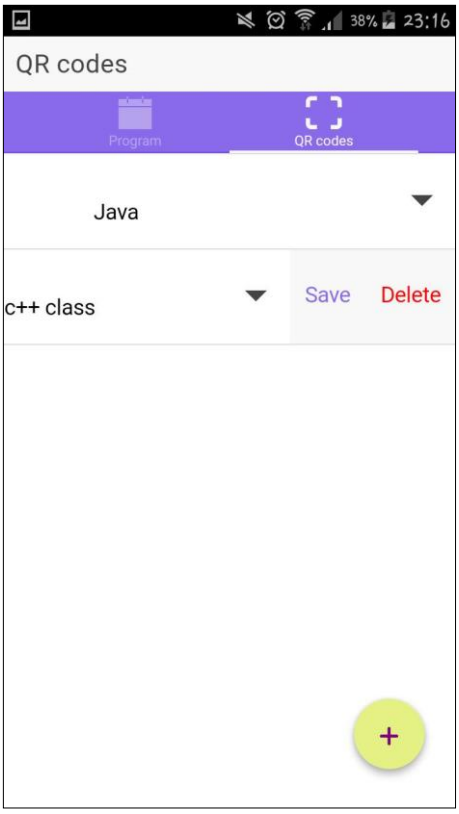


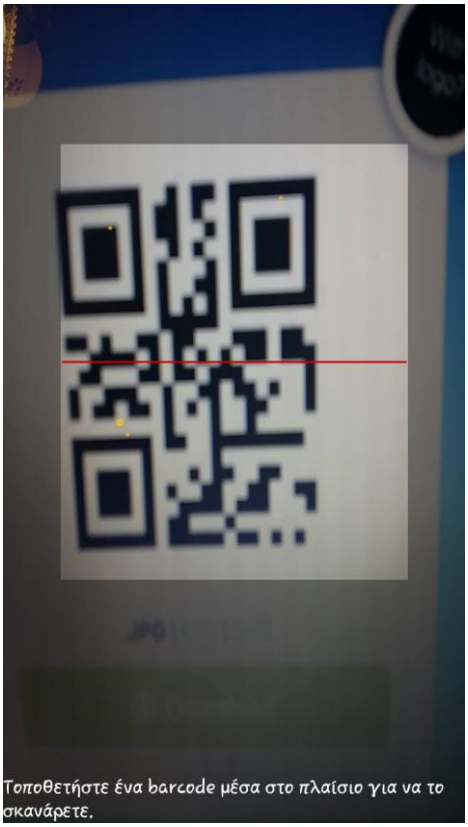
Εικόνα 89: QR codes (χρήστης υποψήφιος διδακτορικός), float menu

Εικόνα 91: επιλεγμένος QR code



Εικόνα 92: Επιλογής μαθήματος για QR code





Εικόνα 93: Διαγραφή/ Αποθήκευση Qr code **Εικόνα 94: Σκανάρισμα QR code**

8 Συμπεράσματα

Στόχος της παρούσας μελέτης ήταν η υλοποίηση ενός διαδικτυακού πληροφοριακού συστήματος αυτοματοποίησης του χρονοπρογραμματισμού των υποψήφιων διδακτόρων, δεδομένων κάποιων περιορισμών, ώστε να καλυφθούν οι ανάγκες επιτήρησης των εξεταζόμενων μαθημάτων κατά τη διάρκεια της εξεταστικής περιόδου. Η λειτουργικότητα της αναπτυχθείσας εφαρμογής εμπλουτίστηκε με την υλοποίηση μιας εφαρμογής για έξυπνες κινητές συσκευές.

Τα προβλήματα χρονοπρογραμματισμού παρουσιάζουν ιδιαίτερο ενδιαφέρον όχι μόνο λόγω της ευρείας εμφάνισής τους στην καθημερινότητα και της ποικιλομορφίας τους, αλλά και λόγω της πολυπλοκότητάς τους [9]. Τα προβλήματα χρονοπρογραμματισμού, αποτελούν προβλήματα αξιοποίησης συγκεκριμένων πόρων, δεδομένων κάποιων περιορισμών, με σκοπό την βέλτιστη ικανοποίηση συγκεκριμένων αντικειμενικών στόχων.

Τα περισσότερα προβλήματα χρονοπρογραμματισμού είναι συνδυαστικά και πολύπλοκα (NP-complete προβλήματα), γεγονός που καθιστά δύσκολη, χρονοβόρα και ακριβή τη βέλτιστη επίλυσή τους. Για το λόγο αυτό, πολλές φορές η επίλυσή τους, η οποία πραγματοποιείται χειροκίνητα, αρκείται απλά στο να ικανοποιεί τους περιορισμούς χωρίς να αποτελεί τη βέλτιστη επιλογή απαραίτητα. Επομένως ένα εργαλείο βελτιστοποίησης και ταυτόχρονα αυτοματοποίησης της διαδικασίας χρονοπρογραμματισμού των υποψήφιων θα διδακτόρων βοηθά αυτούς που μέχρι τώρα κατασκευάζουν το πρόγραμμα και μειώνει το χρόνο της διαδικασίας.

Η αναπτυχθείσες εφαρμογές συνιστώνται για την αυτόματη παραγωγή προγράμματος επιτηρήσεων αλλά και την παρακολούθηση της έγκαιρης παρουσίας των επιτηρητών στην εξέταση του μαθήματος που τους έχει ανατεθεί.

Υπάρχει ένα σύνολο μελλοντικών επεκτάσεων που θα μπορούσαν να υλοποιηθούν ώστε να βελτιώσουν τη λειτουργικότητα και τη χρησιμότητα των δύο εφαρμογών. Αναφορικά με την εφαρμογή για κινητές συσκευές, εύστοχο θα ήταν να ειδοποιείται με push notification, ο ενδιαφερόμενος καθηγητής κάθε φορά που κάποιος επιτηρητής στο μάθημά του δηλώνει την παρουσία του μέσω του σκαναρίσματος του QR κωδικού που ο ίδιος έχει παράγει μέσω της εφαρμογής.

Επιπλέον, μια καλή ιδέα θα ήταν, όταν υπάρχει κάποια αλλαγή στο πρόγραμμα, να ενημερώνονται, όχι μόνο μέσω email όπως συμβαίνει τώρα, αλλά και με push notifications. Μία ακόμα πρόταση είναι η δυνατότητα αποθήκευσης των QR κωδικών που παράγει ο καθηγητής μέσω της εφαρμογής για κινητές συσκευές, έτσι ώστε να μπορεί να τους ανακτήσει όποτε επιθυμεί. Ακόμα, η δημιουργία λογαριασμού θα μπορούσε να αντικατασταθεί από την σύνδεση στο σύστημα μέσω κάποιου κοινωνικού δικτύου, όπως facebook ή gmail, διαδικασία που αρχικά απορρίφθηκε λόγω της δυσκολίας ανάθεσης ρόλων στους χρήστες.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Leff, Avraham, and James T. Rayfield. "Web-application development using the model/view/controller design pattern." *Enterprise Distributed Object Computing Conference, 2001. EDOC'01. Proceedings. Fifth IEEE International*. IEEE, 2001.
2. Wren, Anthony. "Scheduling, timetabling and rostering—a special relationship?." *Practice and theory of automated timetabling*. Springer Berlin Heidelberg, 1995. 46-75.
3. Ladd, Seth, et al. *Expert Spring MVC and Web Flow*. Vol. 1. Berkeley, CA: Apress, 2006.
4. Gupta, Praveen, Pacheri Bari, and M. C. Govil. "Spring Web MVC Framework for rapid open source J2EE application development: a case study." *Interface* 2.6 (2010): 1684-1689.
5. Johnson, Rod, et al. "The Spring Framework–Reference Documentation." *Interface* 21 (2004).
6. Green, Brad, and Shyam Seshadri. *AngularJS*. " O'Reilly Media, Inc.", 2013.
7. Kozlowski, Pawel. *Mastering Web Application Development with AngularJS*. Packt Publishing Ltd, 2013.
8. Jain, Nilesh, Priyanka Mangal, and Deepak Mehta. "AngularJS: A Modern MVC Framework in JavaScript." *Journal of Global Research in Computer Science* 5.12 (2015): 17-23.
9. Cordova, Apache. "About Apache Cordova." (2013).
10. Wargo, John M. *Apache Cordova 4 Programming*. Pearson Education, 2015.
11. Fromherz, Markus PJ. "Constraint-based scheduling." *American Control Conference, 2001. Proceedings of the 2001*. Vol. 4. IEEE, 2001.
12. Baptiste, Philippe, Claude Le Pape, and Wim Nuijten. "Incorporating E cient Operations Research Algorithms in Constraint-Based Scheduling." *Proceedings of the First International Joint Workshop on Artificial Intelligence and Operations Research, Timberline Lodge, Oregon*. 1995.
13. Charland, Andre, and Brian Leroux. "Mobile application development: web vs. native." *Communications of the ACM* 54.5 (2011): 49-53.

14. Joorabchi, Mona Erfani, Ali Mesbah, and Philippe Kruchten. "Real challenges in mobile app development." *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on*. IEEE, 2013.
15. Tilkov, Stefan, and Steve Vinoski. "Node. js: Using javascript to build high-performance network programs." *IEEE Internet Computing* 14.6 (2010): 80.
16. Chodorow, Kristina. *MongoDB: the definitive guide*. " O'Reilly Media, Inc.", 2013.
17. Boicea, Alexandru, Florin Radulescu, and Laura Ioana Agapin. "MongoDB vs Oracle--Database Comparison." *2012 Third International Conference on Emerging Intelligent Data and Web Technologies*. IEEE, 2012.
18. Cattell, Rick. "Scalable SQL and NoSQL data stores." *ACM SIGMOD Record* 39.4 (2011): 12-27.
19. Membrey, Peter, Eelco Plugge, and DUPTim Hawkins. *The definitive guide to MongoDB: the noSQL database for cloud and desktop computing*. Apress, 2011.
20. Pokorny, Jaroslav. "NoSQL databases: a step to database scalability in web environment." *International Journal of Web Information Systems* 9.1 (2013): 69-82.
21. Miller, Frederic P., Agnes F. Vandome, and John McBrewster. "Apache Maven." (2010).
22. Salewski, Frank, Andreas Schirmer, and Andreas Drexl. "Project scheduling under resource and mode identity constraints: Model, complexity, methods, and application." *European Journal of Operational Research* 102.1 (1997): 88-110.
23. Ferland, Jacques A., and Serge Roy. "Timetabling problem for university as assignment of activities to resources." *Computers & operations research* 12.2 (1985): 207-218.
24. Selfa, Diana M., Maya Carrillo, and Ma del Rocío Boone. "A database and web application based on MVC architecture." *Electronics, Communications and Computers, 2006. CONIELECOMP 2006. 16th International Conference on*. IEEE, 2006.

25. Flanagan, David. *JavaScript: the definitive guide*. " O'Reilly Media, Inc.", 2006.
26. Crockford, Douglas. *JavaScript: The Good Parts: The Good Parts*. " O'Reilly Media, Inc.", 2008.
27. Middleton, Neil, and Richard Schneeman. *Heroku: Up and Running*. " O'Reilly Media, Inc.", 2013.
28. Heroku, I. "What is Heroku?." *Stand* 30 (2011): 2011.
29. Kemp, Chris, and Brad Gyger. *Professional Heroku Programming*. John Wiley & Sons, 2013.
30. https://my.modulus.io/
31. https://dashboard.heroku.com/
32. Burbeck, Steve. "Applications programming in smalltalk-80 (tm): How to use model-view-controller (mvc)." *Smalltalk-80 v2* 5 (1992).
33. Liu, Yimeng, Yizhi Wang, and Yi Jin. "Research on the improvement of MongoDB Auto-Sharding in cloud environment." *Computer Science & Education (ICCSE), 2012 7th International Conference on*. IEEE, 2012.