ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Διδακτικής της Τεχνολογίας και Ψηφιακών Συστημάτων

Μεταπτυχιακή διπλωματική εργασία

Σχεδίαση και ανάπτυξη εφαρμογής με Web Services

Πνευματικός Δημήτριος
Α.Μ.: ΜΕ08028

Επιβλέπων καθηγητής : Θεμιστοκλέους Μαρίνος, Επίκουρος καθηγητής.

Περίληψη

Η μεγάλη ανάπτυξη του διαδικτύου τα τελευταία χρόνια έφερε στο προσκήνιο ένα σύνολο από νέες τεχνολογίες. Ταυτόχρονα, ο προσωπικός υπολογιστής με τη χρήση των φυλλομετρητών (Browsers) έπαψε να είναι ο κυρίαρχος τρόπος πρόσβασης στο διαδικτύου, καθώς ο μέσος χρήστης μπορεί πλέον να χρησιμοποιήσει είτε την ταμπλέτα του είτε το κινητό τηλέφωνο. Οι διαφορετικοί τρόποι πρόσβασης προσέφεραν εύφορο έδαφος στην ανάπτυξη νέων, καινοτόμων εφαρμογών ή επέβαλλαν την ανάγκη επέκτασης ήδη υπάρχοντων εφαρμογών ώστε να υποστηρίξουν τις νέες πλατφόρμες. Τα Web Services είναι μια τεχνολογία που συνηγορεί σε αυτή τη κατεύθυνση καθιστώντας ευέλικτη την επέκταση είτε ήδη υπάρχοντων εφαρμογών είτε την ανάπτυξη νέων, προσφέροντας την απαραίτητη ασφάλεια, ευχρηστία αλλά και εξασφαλίζοντας μια κοινή διεπαφή ανεξαρτήτου πλατφόρμας, "διαγράφοντας" έτσι τα υπάρχοντα τεχνολογικά σύνορα.

Παρ'όλα αυτά συνεχίζουν να υφίστανται τομείς όπου δεν έχουν ακόμα εφαρμοστεί αυτές οι τεχνολογίες. Ο τομέας των συστημάτων διαχείρισης μαθητών δείχνει να είναι ένας από αυτούς. Στην παρούσα διπλωματική εργασία θα παρουσιάσουμε ένα σύστημα διαχείρισης μαθητών,το Dreskt, το οποίο χρησιμοποιεί Web Services ώστε να εκθέσει μέρος της λειτουργικότητας του στην μαθητική κοινότητα. Οι μαθητές με τη χρήση μιας εφαρμογής πελάτη σε κινητό τηλέφωνο Android, αποκτούν πρόσβαση σε λειτουργίες του Dreskt διευκολύνοντας έτσι τη διεπαφή με τη γραμματεία αλλά και μειώνοντας όγκο εργασίας από αυτή εξασφαλίζοντας την απαιτούμενη ταχύτητα,ασφάλεια διατηρώντας παράλληλα της υπάρχουσες εσωτερικές ροές εργασίας.

Περιεχόμενα

Πίνακας εικόνων

Κεφάλαιο 1

Εισαγωγή

1.1 Ορισμός του προβλήματος

Τα τελευταία χρόνια έχουμε γίνει μάρτυρες της διαρκώς αυξανόμενης διάδοσης του διαδικτύου. Ο παγκόσμιος ιστός, από τα κλειστά συστήματα του CERN, έχει πλέον φτάσει σε σχεδόν κάθε γειτονιά του πλανήτη αλλάζοντας μορφή σε πολύ μικρά χρονικά διαστήματα. Από τις απλές στατικές ιστοσελίδες του παρελθόντος ο παγκόσμιος ιστός φιλοξενεί πλέον περίπλοκες διαδικτυακές εφαρμογές σαν και αυτές που κάποτε συνηθίζαμε να τρέχουμε στους προσωπικούς μας υπολογιστές όπως για παράδειγμα online κειμενογράφους, εφαρμογές επεξεργασίας εικόνας και πολλές άλλες. Η έλευση του περίφημου Web 2.0 έφερε μαζί της και ένα σύνολο από τεχνολογίες οι οποίες συνεισφεραν σημαντικά προς αυτή τη κατεύθυνση και διευκόλυναν δραματικά την ανάπτυξη τέτοιων online εφαρμογών. Βρέθηκαν επίσης έξυπνοι τρόποι για να απομακρυνθεί το βάρος από τους εξυπηρετητές και να μεταφερθεί μέρος του στους πελάτες και κάπως έτσι άρχισε να εισάγεται η έννοια του λογισμικού ως υπηρεσία (SaaS software as a Service).

Η παράλληλη ανάπτυξη και διάδοση έξυπνων συσκευών σε συνδιασμό με τα αυξημένα τεχνικά χαρακτηριστικά που παρουσιάζουν άρχισαν να δημιουργούν την ανάγκη για λογισμικό ίδιου επιπέδου με αυτό που τρέχαμε μέχρι πρότινος στους προσωπικούς υπολογιστές μας. Η επαναδημιουργία των εφαρμογών σε κάθε

διαφορετική πλατφόρμα θα επέφερε μεγάλο κόστος στις επιχειρήσεις και κατά συνέπεια μια άλλη λύση θα έπρεπε να εφαρμοστεί. Η λύση αυτή ήρθε με την έλευση της τεχνολογίας των Web Services. Στον παγκόσμιο ιστό του παρόντος σχεδόν όλοι οι μεγάλοι ιστοτόποι εκθέτουν μέρος της λειτουργικότητας τους με τη χρήση προγραμματιστικών διεπαφών και επιτρέπουν διαλειτουργικότητα ανεξαρτήτως πλατοφόρμας ή συσκευής. Η έκθεση αυτή, δημιούργησε προυποθέσεις ώστε οποιοσδήποτε προγραμματιστής να μπορεί να χρησιμοποιεί λειτουργικότητα από υπηρεσίες σε δικά του προγράμματα είτε ακόμα και να την επεκτείνει χρησιμοποιώντας διαφορετικές υπηρεσίες.

Παρ'όλα αυτά και επειδή οι τεχνολογίες αυτές είναι ακόμα στα πρώτα τους βήματα υπάρχει ένα μέρος διαδικτυακών εφαρμογών οι οποίες δεν έχουν ακολουθήσει αυτές τις τεχνολογικές εξελίξεις. Ένα παράδειγμα είναι τα συστήματα διαχείρισης μαθητών με τα οποία ασχολείται και η παρούσα διπλωματική εργασία.

Κατά συνέπεια διαφαίνεται η ύπαρξη ανάγκης υλοποίησης καινούριων συστημάτων ή ακόμα και επέκταση των ήδη υπαρχόντων προς μια κατεύθυνση ποιο υπηρεσιοστρεφή ώστε να μπορεί να επωφεληθεί των πλεονεκτημάτων, πιο σύγχρονων αρχικτεκτονικών.

Σκοπός της διπλωματικής εργασίας είναι να υλοποιήσει ένα παρόμοιο σύστημα με κύρια φιλοσοφία τη χρήση web services για την επέκταση του και να δείξει πως αυτά μπορούν να χρησιμοποιηθούν για να διευκολύνουν τη λειτουργία ενός εκπαιδευτικού ιδρύματος.

1.2 Σκοπός της εργασίας

Στα πλαίσια της εργασίας αυτής θα παρουσιάσουμε της επικρατέστερες αρχιτεκτονικές Web Services. Θα δούμε τα πλεονεκτήματα και τα μειονεκτήματα

της κάθε αρχιτεκτονικής και θα μελετήσουμε τον τρόπο ανάπτυξης μιας διαδικτυακής εφαρμογής με τη χρήση της REST αρχιτεκτονικής.

Σκοπός της εργασίας είναι η ανάπτυξη ενός συστήματος διαχείρισης μαθητών μικρής κλίμακας το οποίο, με τη βοήθεια των web services, θα εκθέτει μέρος της λειτουργικότητας του στους μαθητές.

Οι **στόχοι** μας είναι να δείξουμε ποια είναι η διαδικασία που πρέπει να ακολουθείται για τη δημιουργία συστημάτων που προσφέρουν υπηρεσίες. Επιπρόσθετα θα μελετήσουμε τα οφέλη που απορρέουν από τέτοιου τύπου αρχιτεκτονικές και το εύρος των επεκτάσεων που μπορούν να δεχτούν με παράδειγμα ένα πρόγραμμα πελάτη για κινητή συσκευή Android.

1.3 Δομή της εργασίας

Η εργασία χωρίζεται σε τέσσερα βασικα κεφάλαια.

Στο **δεύτερο κεφάλαιο** παρουσιάζεται η έννοια του Web Service. Ορίζονται οι βασικές συνιστώσες ενός web service και συντίθεται ένας ορισμός με βάση άλλους ορισμούς που δόθηκαν από μεγάλες εταιρίες λογισμικού. Επιπλέον αναλύουμε τα βασικά συστατικά από τα οποία αποτελείται ένα web service και γίνεται μελέτη των τεχνολογιών που χρησιμοποιήθηκαν μέχρι τώρα στην ανάπτυξη τους. Δύο είναι οι κύριες αρχιτεκτονικής που αναλύονται : τα Big Web Services και η αρχιτεκτονική REST. Τέλος παρουσιάζονται αναλύτικα τα πλεονεκτήματα και μειονεκτήματα της κάθε αρχιτεκτονικής.

Στο **τρίτο κεφάλαιο** γίνεται μια έρευνα σχετικά με τα συστήματα διαχείρισης μαθητών. Μελετάμε τον ορισμό της έννοιας αυτής και παρουσιάζουμε μερικά από τα πιο διαδομένα συστήματα διαχείρισης μαθητών προσπαθώντας να εντοπίσουμε κατά πόσο υποστηρίζουν web services. Τέλος παρουσιάζουμε έναν πίνακα με σύνοψη ως σύνοψη της σύγκρισης των αποτελεσμάτων που βρέθηκαν

αποδεικνύοντας την ανάγκη δημιουργίας ενός νέου.

Στο **κεφάλαιο τέσσερα** δίνεται η πρώτη παρουσίαση του συστήματος Dreskt. Δείχνουμε της αρχιτεκτονικές αποφάσεις που ελήφθησαν καθώς και όλα τα εμπλεκόμενα μέρη του συστήματος. Παρουσιάζονται οι λειτουργικές και οι μη-λειτουργικές απαιτήσεις του συστήματος και δίνονται οι περιπτώσεις χρήσεις του.

Τέλος στο **πέμπτο κεφάλαιο** γίνεται η παρουσίαση του συστήματος Dreskt μέσα από σενάρια χρήσης. Εκεί δείχνουμε πως μπορούν να υλοποιηθούν πραγματικά σενάρια με τη χρήση του συστήματος και αποδεικνύουμε γιατί ένα τέτοιο σύστημα μπορεί να λειτουργήσει προς μια θετική κατεύθυνση συνηγορώντας στην ακόμα ομαλότερη λειτουργία ενός εκπαιδευτικού ιδρύματος.

Κεφάλαιο 2

Βιβλιογραφική επισκόπηση

2.1 Εισαγωγή

Στο κεφάλαιο που ακολουθεί θα περιγράψουμε τη θεωρία που πλαισιώνει τα web services. Συγκεκριμένα, θα μελετήσουμε τους υπάρχοντες ορισμούς που δίνονται από μεγάλες εταιρίες και οργανισμούς στον χώρο της πληροφορικής και θα προσπαθήσουμε να εξάγουμε έναν ορισμό ο οποίος θα καλύπτει όλες τις περιπτώσεις. Επιπρόσθετα, θα αναλύσουμε τις δύο επικρατέστερες αρχιτεκτονικής υλοποίησης web services , την αρχιτεκτονική **REST** και τα **Big Web Services**, με έμφαση στην πρώτη μια και αυτή είναι η επιλεχθείσα αρχιτεκτονική για την εφαρμογή που θα αναπτύξουμε.

2.2 Ορισμός του web service

Στο διαδίκτυο μπορεί κανείς να βρει ένα σύνολο από ορισμούς που δίνονται ως προς την έννοια web service. Οι σημαντικότεροι δίνονται από την Microsoft, την IBM και τον Διεθνή Οργανισμό Προτύπων του διαδικτύου (W3C). Σύμφωνα, λοιπόν, με την Microsoft ένα web service είναι :

> *" ... μια εφαρμογή που χρησιμοποιεί καθιερωμένους τρόπους μεταφοράς, κωδικοποίησης, και καθιερωμένα πρωτόκολλα για να*

μεταφέρει πληροφορία [...] τα web services καθιστούν, υπολογιστικά συστήματα ανεξαρτήτου πλατφόρμας, ικανά να επικοινωνήσουν πάνω είτε από εταιρικά δίκτυα είτε στο ίντερνετ με ασφάλεια, αξιόπιστα μηνύματα και κατανεμημένες ενέργειες." [1]

Ο ορισμός που δίνεται από την IBM δεν απέχει πολύ από αυτόν της Microsoft :

" Τα web services είναι αυτοτελείς, πολυμελείς εφαρμογές που μπορούν να περιγραφούν, να εντοπιστούν, και να χρησιμοποιηθούν μέσα από ένα οποιοδήποτε δίκτυο, γενικότερα όμως, μέσα από το ίντερνετ." [2]

Οι παραπάνω ορισμοί είναι σχεδόν ταυτόσημοι με μοναδική εξαίρεση ότι η IBM θέτει μια επιπλέον προϋπόθεση : να μπορούν να εντοπιστούν. Ας δούμε και τον τελευταίο ορισμό που δίνεται από τον οργανισμό W3C. Σύμφωνα με τον W3C :

" Ένα web service είναι ένα σύστημα λογισμικού που υποστηρίζει την επικοινωνία μεταξύ συστημάτων υπολογιστών μέσα από ένα δίκτυο. Διαθέτει μια διεπαφή, η οποία περιγράφεται σε επεξεργάσιμη από άλλον υπολογιστή, μορφή (και συγκεκριμένα WSDL)." [3]

Όπως παρατηρούμε, όλοι οι παραπάνω ορισμοί μοιράζονται κοινά στοιχεία. Συγκεκριμένα και οι τρείς ορισμοί αποδέχονται οτι ένα web service :

---

1 *"Web services are applications that use standard transports, encodings, and protocols to exchange information. Web services enable computer systems on any platform to communicate over corporate intranets, extranets, and across the Internet with support for end-to-end security, reliable messaging, distributed transactions, and more"* - Web Services and the Microsoft Platform. Payam Shodjai.

2 *"Web Services are self-contained, modular applications that can be described, published, located, and invoked over a network, generally, the World Wide Web."* - IBM Services Architecture Team.

3 *"A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL)."* - Web services Glossary. W3C.

1. είναι ένα αυτοτελές σύστημα λογισμικού
2. επιτρέπει την επικοινωνία μεταξύ διαφορετικών υπολογιστικών συστημάτων με τη χρήση ενός δικτύου.
3. Εκθέτει μια συγκεκριμένη λειτουργικότητα προς τον "έξω κόσμο" η οποία περιγράφεται στη διεπαφή της και η οποία διεπαφή της είναι ορισμένη με έναν τρόπο ώστε να είναι επεξεργάσιμος από μια μηχανή.
4. Είναι χρήσιμο, όχι όμως και απαραίτητο, να χρησιμοποιούνται τεχνολογίες οι οποίες έχουν καθιερωθεί στην πλειονότητα των υπολογιστικών συστημάτων.

Επομένως αν θέλουμε να συνθέσουμε έναν πλήρη ορισμό για το τι ακριβώς είναι ένα web service θα λέγαμε ότι :

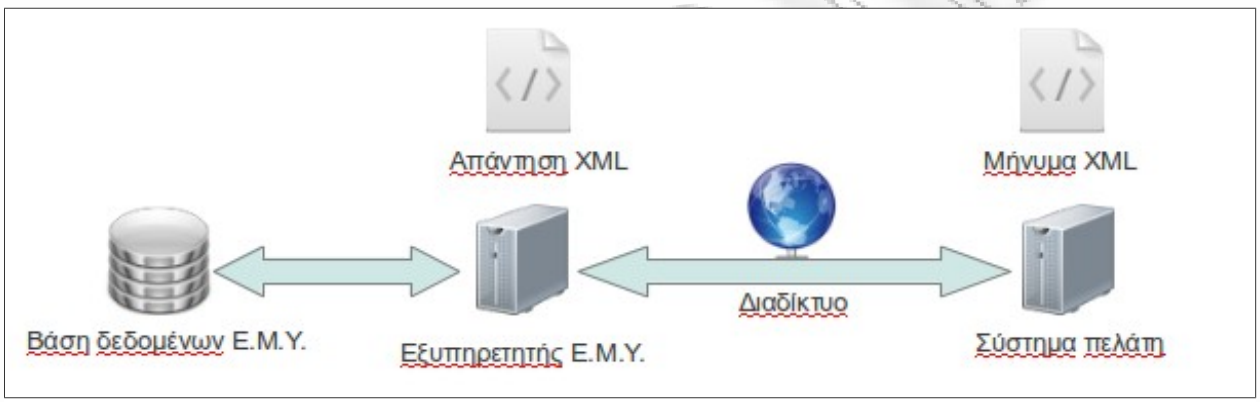
> " ***Web service είναι ένα αυτοτελές σύστημα λογισμικού το οποίο, μέσω δικτύου, εκθέτει συγκεκριμένη λειτουργικότητα μέσω μιας διεπαφής ορισμένη με τρόπο επεξεργάσιμο από άλλο υπολογιστή και καθιστά εφικτή την επικοινωνία μεταξύ ετερόκλητων υπολογιστικών συστημάτων."*** [4]

Για να κατανοήσουμε καλύτερα την έννοια web service ας δούμε ένα απλό παράδειγμα. Ας υποθέσουμε οτι η Εθνική Μετεωρολογική υπηρεσία θέλει να δημοσιοποιήσει ένα σύνολο από πληροφορίες που αφορούν καιρικές προβλέψεις σε μια δεδομένη γεωγραφική περιοχή. Αυτές οι πληροφορίες θα έπρεπε να είναι προσβάσιμες από τα υπολογιστικά συστήματα των Μ.Μ.Ε., του Λιμενικού Σώματος, του Πολεμικού Ναυτικού αλλά και μια εφαρμογή που τρέχει σε smartphone και βοηθά απλούς χρήστες να προγραμματίσουν τις καλοκαιρινές διακοπές τους.

Στο Web όπως το γνωρίζαμε μέχρι τώρα οι πληροφορίες της Ε.Μ.Υ θα ήταν

4 Από τον ορισμό έχουμε αφαιρέσει το μέρος εκείνο που αναφέρεται στη χρήση του WSDL για τη δημιουργία της διεπαφής, παρ'όλο που αναφερόταν και στον ορισμό της IBM και στον ορισμό της Microsoft, για λόγους που θα αναφέρουμε παρακάτω.

διαθέσιμες είτε μέσα από την ιστοσελίδα της είτε μέσα από τα δελτία καιρού στα Μ.Μ.Ε . Η χρήση της πληροφορίας όμως, σε αυτή την περίπτωση θα ήταν κάπως περιορισμένη. Για παράδειγμα το Πολεμικό Ναυτικό θα ήθελε να "τροφοδοτήσει" τα υπολογιστικά του συστήματα με αυτές τις πληροφορίες και να εξάγει δικά του, επεξεργασμένα δεδομένα για τον καιρό. Θα ήταν εξαιρετικά χρονοβόρα, δύσκολη και με έφεση σε λάθη η χειροκίνητη αντιγραφή των πηροφοριών αυτών μέσα από τον ιστότοπο της Ε.Μ.Υ . Το πρόβλημα αυτό θα μπορούσε άνετα να λυθεί με τη χρήση web services.



Εικόνα 1: Τοπολογία δικτύου με web services

Σε αυτή τη περίπτωση η Ε.Μ.Υ θα εγκαθιστούσε στους εξυπηρετητές της μια εφαρμογή η οποία θα δέχεται μηνύματα δομημένα με ένα συγκεκριμένο τρόπο, θα τα επεξεργάζεται και στη συνέχεια θα αποστέλλει πίσω στον ενδιαφερόμενο την απάντηση στα ερωτήματα που έθεσε. Στη συνέχεια ο παραλήπτης του μηνύματος θα εισήγαγε στο σύστηματα του αυτά τα μηνύματα και μετά από επεξεργασία θα εξήγαγε τα δικά του δεδομένα. Η διαδικασία αυτή είναι πλήρως αυτοματοποιημένη και δεν απαιτεί ανθρώπινη παρέμβαση. Η κωδικοποίηση και αποκωδικοποίηση του μηνύματος γίνεται από εντελώς διαφορετικά συστήματα ,με διαφορετικό τρόπο. Η ουσία βρίσκεται στην "κοινή γλώσσα" χρήσης των web services της Ε.Μ.Υ. Η "γλώσσα" αυτή ορίζεται από την Ε.Μ.Υ και όσοι επιθυμούν να χρησιμοποιήσουν τις

υπηρεσίες της (τα web services δηλαδή) είναι υποχρεωμένοι να την μιλούν. Η γλώσσα είναι η διεπαφή της υπήρεσίας και το συντακτικό της μπορεί να είναι είτε η ***XML*** (Extensible Markup Language) είτε ***JSON*** (Javascript Notation Object) είτε οποιαδήποτε σύνταξη επιθυμεί ο δημιουργός της υπηρεσίας. Τέλος, όλοι όσοι θέλουν να χρησιμοποιήσουν αυτές τις υπηρεσίες είναι υποχρεωμένοι να έχουν πρόσβαση στο διαδίκτυο.

2.3 Η αρχιτεκτονική των Web Services

Τα τελευταία χρόνια, και ειδικά με την διείσδυση όλο και περισσοτέρων χρηστών στο διαδίκτυο, έχουν καθιερωθεί δύο αρχιτεκτονικές web services. Σύμφωνα με το W3C :

> *" Μπορούμε να αναγνωρίσουμε δύο μεγάλες κατηγορίες web services :*
>
> - *REST-συμβατές υπηρεσίες, όπου ο πρωταρχικός ρόλος των υπηρεσιών είναι να χειριστούν αναπαραστάσεις διαδικτυακών πόρων σε XML μορφή χρησιμοποιώντας ένα ενιαίο σύνολο λειτουργιών.*
> - *Αυθαίρετες υπηρεσίες web, όπου η κάθε υπηρεσία ορίζει ένα αυθαίρετο σύνολο λειτουργιών. "* [5]

Η πρώτη κατηγορία είναι ευρύτερα γνωστή ως *Web API* και βασίζεταιι εξ'ολοκλήρου στο HTTP πρωτόκολλο ενώ η δεύτερη ως Big *Web Services* ή *WS-**

5 "We can identify two major classes of Web services: REST-compliant Web services, in which the primary purpose of the service is to manipulate XML representations of Web resources using a uniform set of "stateless" operations; and arbitrary Web services, in which the service may expose an arbitrary set of operations." - Web Services Architecture. W3C. February 2004 βλ. <http://www.w3.org/TR/ws-arch/#relwwwrest>

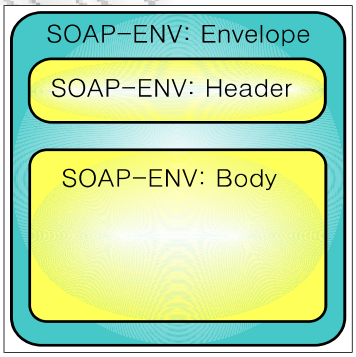
stack και στηρίζονται στην ανταλλαγή μηνυμάτων SOAP.

2.4 Big Web Services – Service Oriented Architecture (SOA)

2.4.1 Το πρωτόκολλο SOAP

Τα Big Web Services είναι ευρέως διαδεδομένα στις επιχειρήσεις. Η XML διαδραματίζει σημαντικό ρόλο καθώς όλες οι επικοινωνίες μεταξύ των συστημάτων γίνονται με τη χρήση της. Τα μηνύματα σε αυτή τη μορφή ακολουθούν το πρότυπο *Simple Object Access Protocol* ή αλλιώς *SOAP.* Το SOAP σχεδιάστηκε και υλοποιήθηκε από την Microsoft και αποτελεί το δομικό στοιχείο στη δημιουργία web services. Αποτελείται από τρία βασικά μέρη (βλ. Εικόνα 2):

- **Φάκελος** (Envelope) – προσδιορίζει τι είναι μέσα στο μήνυμα και πως πρέπει να το διαχειριστεί κάποιος
- **Επικεφαλίδα** (Header) – προσδιορίζει πληροφορίες που αφορούν τη συγκεκριμένη υπηρεσία
- **Σώμα μηνύματος** (Body) – περιέχει το κυρίως μήνυμα προς επεξεργασία



Εικόνα 2: Δομή ενός μηνύματος SOAP (πηγή wikipedia)

Για παράδειγμα ένα μήνυμα SOAP προς ένα web service που παρέχει πληροφορίες για τα στοκ προιόντων ενός καταστήματος θα μπορούσε να συνταχθεί με τον εξής τρόπο :

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

Εικόνα 3: Μήνυμα SOAP προς web service ενός e-shop

Όπως φαίνεται στην Εικόνα 3, στην αρχή του <soap:Envelop> περιέχονται πληροφορίες σχετικά με την δομή και την κωδικοποίηση του μηνύματος. Αυτές οι πληροφορίες επεξεργάζονται από την πλευρά του web service του e-shop. Στο κυρίως σώμα το μήνυμα που έχουμε συντάξει ζητά την τιμή (κλήση της μεθόδου GetStockPrice) για το προιόν IBM (<m:StockName>). Μόλις η υπηρεσία ολοκληρώσει την εσωτερική διεργασία για την εύρεση της τιμής θα επιστρέψει το παρακάτω μήνυμα :

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

Εικόνα 4: SOAP απάντηση από το e-shop

Η κλήση της μεθόδου GetStockPrice επιστρέφει την GetStockPriceResponse περιλμβάνοντας την τιμή που ζητήσαμε, σε ένα πανομοιότυπο δομικά SOAP μήνυμα.

Το πρωτόκολλο SOAP φέρει τρία βασικά χαρακτηριστικά :

- **Επεκτασιμότητα** – δεν υφίσταται κανένας περιορισμός στο μήνυμα που μπορεί να μεταφέρει. Το σώμα ενός SOAP μηνύματος μπορεί να επεκταθεί για να μεταφέρει οποιαδήποτε πληροφορία.
- **Ουδετερότητα** – ένα SOAP μήνυμα μπορεί να μεταφερθεί μέσα από ένα σύνολο διαφορετικών πρωτοκόλλων όπως HTTP, TCP, JMS.
- **Ανεξαρτησία** – Μηνύματα SOAP μπορούν να δημιουργηθούν μέσα από τη συντριπτική πλειοψηφία γλωσσών προγραμματισμού.

Η χρήση του HTTP για τη μεταφορά μηνυμάτων SOAP είναι αυτή που χρησιμοποείται περισσότερο σήμερα καθώς οι υπάρχουσες υποδομές σε μεγάλα εταιρικά δίκτυα συνεργάζονται ήδη απροβλημάτιστα με αυτό.

2.4.2 Web Services Description Language

Με τη θέσπιση του SOAP σαν πρότυπο τρόπο επικοινωνίας, προστέθηκε ένα ανώτερο επίπεδο το οποίο υποδεικνύει το σύνολο των λειτουργιών που προσφέρονται από μια υπηρεσία. Το επίπεδο αυτό ονομάζεται **Web Services Description Language** ή WSDL. Και εδώ, η λίστα με τις διαθέσιμες λειτουργίες μιας υπηρεσίας είναι κατεγεγγραμένη σε XML μορφή ώστε να είναι επεξεργάσιμη από υπολογιστή.

```
1  <message name="getTermRequest">
2    <part name="term" type="xs:string"/>
3  </message>
4
5  <message name="getTermResponse">
6    <part name="value" type="xs:string"/>
7  </message>
8
9  <portType name="glossaryTerms">
10   <operation name="getTerm">
11     <input message="getTermRequest"/>
12     <output message="getTermResponse"/>
13   </operation>
14 </portType>
```

Εικόνα 5: παράδειγμα WSDL εγγράφου

Εξηγώντας τον κώδικα της Εικόνα 5 με προγραμματιστικούς όρους, θα λέγαμε ότι, η ετικέτα με το όνομα glossaryTerms αντιστοιχεί σε μια βιβλιοθήκη συναρτήσεων (function library). Μέλος της βιβλιοθήκης είναι η συνάρτηση getTerm η οποία παίρνει σαν όρισμα το getTermRequest και το getTermRequestResponse είναι η παράμετρος επιστροφής.

2.4.3 UDDI

Τον Αύγουστο του 2000 και σε επέκταση του WS stack παρουσιάστηκε το *Universal Description, Discovery and Integration* ή UDDI. Το UDDI ήταν μια ανοιχτή πρωτοβουλία της βιομηχανίας, υπό την αιγίδα του Οργανισμού για την Προώθηση Προτύπων των Δομημένων πληροφοριών (OASIS), που επιτρέπει στις επιχειρήσεις να δημοσιεύουν προγράμματα παροχής υπηρεσιών, να ανακαλύπτουν η μία την άλλη και επίσης καθορίζει πως οι υπηρεσίες και οι εφαρμογές λογισμικού πρέπει να αλληλεπιδρούν μέσω του Διαδικτύου. Είναι λοιπόν ένα μητρώο ανεξάρτητο πλατφόρμας, βασισμένο σε XML, όπου σκοπός είναι να παρέχει μια λίστα από υπηρεσίες που προσφέρονται από διάφορες επιχειρήσεις ανά τον κόσμο.

Δυστυχώς για τις φιλοδοξίες των αρχικών σχεδιαστών, το UDDI, δεν υιοθετήθηκε ως ήλπιζαν. Το κίνημα του open source τάχθηκε ενάντια σε αυτή τη κίνηση καταγραφής των υπηρεσιών. Το 2006 μεγάλες εταιρίες στον χώρο της πληροφορικής απέσυραν τους εν λειτουργία κόμβους και μετά από λίγο καιρό η OASIS διέκοψε της εργασίες της. Για αυτούς τους λόγους και επειδή η αρχιτεκτονική της εφαρμογής που θα αναπτύξουμε δεν θα είναι η SOAP, το UDDI δε θα αναλυθεί περαιτέρω.

2.4.4 Κριτική στην SOAP αρχιτεκτονική

Η SOAP αρχιτεκτονική παρ'όλο που είναι ευρέως διαδεδομένη και είναι ίσως η πρώτη αρχιτεκτονική web services που χρησιμοποιήθηκε σε τόσο ευρεία κλίμακα έχει δεχτεί πολλές επικρίσεις. Πολλοί πιστεύουν οτι το μεγαλύτερο πρόβλημα είναι η μεγάλη πολυπλοκότητα που εισάγουν τα διάφορα πρότυπα υλοποίησης μιας υπηρεσίας. Παρ'όλο που πολλές εταιρίες λογισμικού προσφέρουν εξειδικευμένα εργαλεία για να απλοποίησουν αυτή τη διαδικασία εν τούτοις πολλοί αντιπαραθέτουν οτι η ανάπτυξη τέτοιων εφαρμογών δε πρέπει να βασίζεται σε

"κλειστό λογισμικό". Κατά δεύτερον αυτός που υλοποιεί μια εφαρμογή βασισμένη σε web services τρίτου, πρέπει να έχει άριστη γνώση του τρόπου που λειτουργεί η υπηρεσία. Για παράδειγμα πρέπει να έχει γνώση των δομών δεδομένων που δέχεται η υπηρεσία, ενώ οποιαδήποτε αλλαγή στη δομή της υπηρεσίας επιφέρει τεράστιες αλλαγές και στην πλευρά του. Για αυτό πολλοί πιστεύουν οτι η SOAP αρχιτεκτονική δεν είναι τελικά τόσο χαλαρά συνδεδεμένη[6] όσο μπορεί να νομίζαμε και έχει τεράστιες συνέπειες σε μεγάλης κλίμακας συστήματα όπου υποστηρίζεται μεγάλος αριθμός χρηστών.

2.5 Η αρχιτεκτονική REST

Η Κατάσταση Παραστατικής Μεταφοράς (Representational State Transfer) ή αλλιώς REST, είναι μια αρχιτεκτονική λογισμικού για κατανεμημένα συστήματα όπως ο παγκόσμιος ιστός. Το REST εμφανίζεται τα τελευταία χρόνια να είναι το ανερχόμενο μοντέλο για web services. Λόγω της απλότητας του έχει εκτοπίσει σε ένα μεγάλο βαθμό το SOAP και το WSDL.

Ο όρος REST εισήχθη για πρώτη φορά το 2000 στη διδακτορική διατριβή[7] του *Roy Fielding*[8] και η ανάπτυξη του έγινε παράλληλα με αυτή του HTTP 1.1 στηριζόμενο σε αυτή του HTTP 1.0. Η μεγαλύτερη υλοποίηση του REST είναι ο ίδιος ο παγκόσμιος ιστός (WWW).

Βάση του REST αποτελούν οι *πελάτες* και οι *εξυπηρετητές*. Οι πελάτες ξεκινούν αιτήματα προς τους εξυπηρετητές, οι οποίοι με τη σειρά τους τα επεξεργάζονται και επιστρέφουν τις ανάλογες απαντήσεις. Τα αιτήματα και οι απαντήσεις είναι χτισμένα γύρω από *αναπαραστάσεις* των *πόρων*. Οι πόροι μπορεί να είναι οτιδήποτε πληροφοριακό αντικείμενο με κάποια σημασία και οι αναπαραστάσεις

6 Η ακριβής αγγλική ορολογία είναι *loosely coupled*.
7 Βλ. κεφάλαιο 5, *Representational State Transfer* <*http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm*>
8 Ο Roy Fielding είναι ένας εκ των συγγραφέων των προδιαγραφών του πρωτοκόλλου HTTP.

τους αποτυπώνουν ένα στιγμιότυπο του πόρου σε μια δεδομένη χρονική στιγμή ή μετά από κάποια εκτελεθείσα ενέργεια. Στα κεφάλαια που ακολουθούν θα εξηγήσουμε αναλυτικότερα τις παραπάνω έννοιες.

2.5.1 Πόροι

Για να κατανοήσουμε καλύτερα την αρχιτεκτονική αυτή, πρέπει πρωτίστως να ορίσουμε τον όρο *πόρος*. Η πρώτη απόπειρα να οριστεί η έννοια πόρος έγινε από τον δημιουργό του WWW, Tim Berners-Lee και από συνεργάτες του στο τεχνικό κείμενο της *Internet Engineering Task Force*, *RFC-2396 :*

> " *Πόρος μπορεί να είναι οτιδήποτε έχει μια ταυτότητα. Γνώριμα παραδείγματα είναι ένα ηλεκτρονικό έγγραφο, μια εικόνα, μια υπηρεσία (π.χ. Το σημερινό δελτίο καιρού για την πόλη του Los Angeles) και ένα σύνολο από πόρους. Μπορούν να υπάρχουν πόροι οι οποίοι δεν είναι προσβάσιμοι μέσω δικτύου όπως για παράδειγμα τα ανθρώπινα όντα, οι εταιρίες και τα βιβλία μιας βιβλιοθήκης. Και αυτά μπορούν να χαρακτηριστούν πόροι.* " [9]

Ο Fielding κρατάει τον ορισμό και σε μια προσπάθεια περαιτέρω αποσαφήνησης γράφει :

> " *Ένας πόρος είναι μια νοητική αντιστοίχιση σε ένα σετ από οντότητες, και όχι η οντότητα που αντιστοιχεί σε αυτό το σετ σε μια*

9 "A resource can be anything that has identity. Familiar examples include an electronic document, an image, a service(e.g., "today's weather report for Los Angeles"), and a collection of other resources. Not all resources are network "retrievable"; e.g., human beings, corporations, and bound books in a library can also be considered resources." - βλ. Berners-Lee, Tim. Uniform Resource Identifiers (URI) : Generic Syntax. August 1998. <http://tools.ietf.org/html/rfc2396> Κεφ.: 1.1 Overview of URI.

δεδομένη χρονική στιγμή. "[10]

Αυτό μπορεί να γίνει καλύτερα κατανοητό με ένα απλό παράδειγμα. Η προτεινόμενη έκδοση ενός ακαδημαικού εγγράφου του συγγραφέα Χ σε ένα γνωστικό αντικείμενο, έχει μια τιμή που δύναται να αλλάξει με την πάροδο του χρόνου καθώς το κείμενο θα μπορούσε να υποστεί βελτιώσεις ή και αλλαγές. Αυτό συνιστά έναν *μεταβλητό πόρο*. Αντίθετα η έκδοση ενός ακαδημαικού συγγράματος στο συνέδριο Χ είναι *στατικός πόρος* διότι στο συγκεκριμένο συνέδριο υπάρχει μόνο μια έκδοση του συγγράματος. Οι πόροι αυτοί είναι δυο διαφορετικοί πόροι ακόμα και αν σε κάποια δεδομένη χρονική στιγμή αντιστοιχούν στην ίδια οντότητα.

2.5.2 Αναπαραστάσεις

Οι αναπαραστάσεις των πόρων μας δείχνουν, αυτά που στην ορολογία του αντικειμενοστρεφούς προγραμματισμού ονομάζουμε, *στιγμιότυπα* των πόρων. Όπως αναφέρει και ο Fielding :

"Η αναπαράσταση είναι μια αλληλουχία bytes, με επιπλέον μετα-δεδομένα, που περιγράφουν την τρέχουσα ή την αποσκοπούσα κατάσταση ενός πόρου. Μια αναπαράσταση αποτελείται ένα σύνολο από δεδομένα ή και μετα-δεδομένα που περιγράφουν. " [11]

---

10 "A resource is a conceptual mapping to a set of entities, not the entity that corresponds to the mapping at any particular point in time." βλ. Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. UNIVERSITY OF CALIFORNIA, 2000. <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm> Κεφ. 5.2.1.1 Resources and Resource Identifiers.

11 "A representation is a sequence of bytes, plus representation metadata to describe those bytes. [...] A representation consists of data, metadata describing the data, and, on occasion, metadata to describe the metadata [...] " - βλ. Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. UNIVERSITY OF CALIFORNIA, 2000. <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm> Κεφ. 5.2.1.2 Representations

2.5.3 Connectors

Οι connectors είναι μια διεπαφή για την επικοινωνία μεταξύ των εμπλεκομένων μελών στην αρχιτεκτονική REST (πελάτης, εξυπηρετητής). Οι connectors ενισχύουν την απλότητα στην αρχιτεκτονική με το να κρύβουν την πολυπλοκότητα υλοποίησης των μηχανισμών επικοινωνίας παρέχοντας διαχωρισμό στις ανάγκες των εμπλεκόμενων μελών. Υπάρχουν πέντε τύποι connectors (βλ. Εικόνα 6).

Table 5-2: REST Connectors

Connector	Modern Web Examples
client	libwww, libwww-perl
server	libwww, Apache API, NSAPI
cache	browser cache, Akamai cache network
resolver	bind (DNS lookup library)
tunnel	SOCKS, SSL after HTTP CONNECT

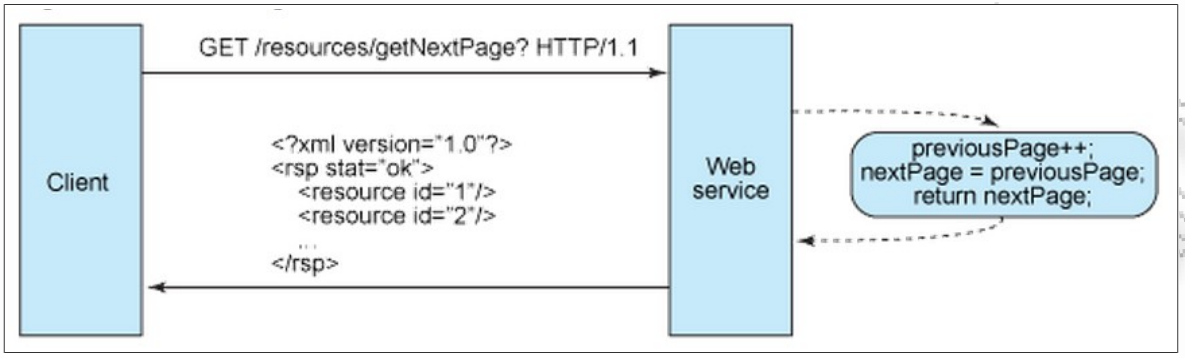
Εικόνα 6: Λίστα με connectors

Κάθε connector μπορεί να μεσολαβήσει για ένα αίτημα χωρίς να γνωρίζει τίποτα για τα αιτήματα που μεσολαβούν οι υπόλοιποι. Με την εισαγωγή αυτού του επίπεδου αφαίρεσης, μια εφαρμογή αλληλεπιδρά με τους πόρους γνωρίζοντας μόνο δύο πράγματα : το *αναγνωριστικό (identifier)* του πόρου και την *ενέργεια* που πρέπει να πραγματοποιηθεί και δεν είναι ανάγκη να γνωρίζει με ποιο τρόπο επιτυγχάνεται η επικοινωνία.

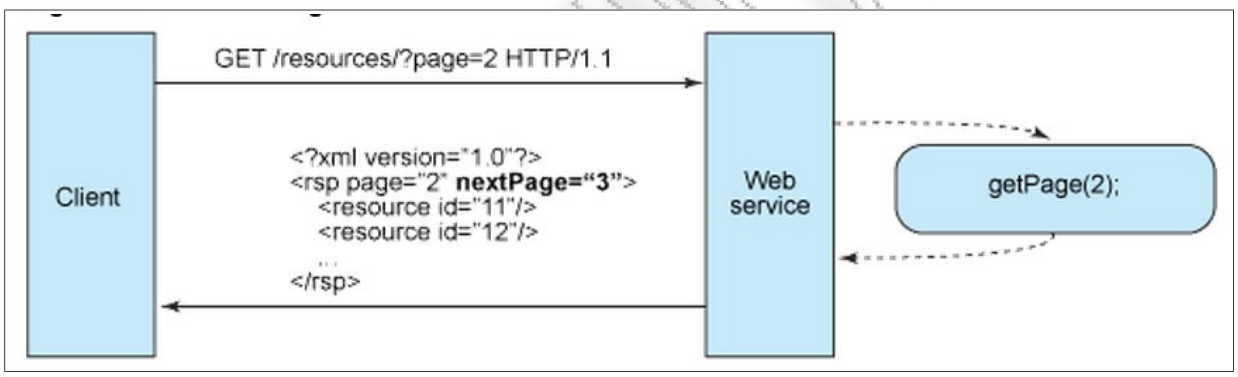
2.5.4 Αρχές της διεπαφής REST

Η αρχιτεκτονική REST επιβάλλει έξι σημεία τα οποία πρέπει να εφαρμόζονται από μια εφαρμογή ώστε να χαρακτηρίζεται σαν REST εφαρμογή. Τα σημεία αυτά είναι ανεξάρτητα από τον τρόπο που θα υλοποιηθούν στα επιμέρους μέλη της εφαρμογής. Κατά συνέπεια, για να ονομάσουμε μια εφαρμογή REST πρέπει να ισχύουν τα εξής :

1. **πελάτης – εξυπηρετητής** : μια ενιαία διεπαφή που να διαχωρίζει τη λειτουργικότητα πελάτη-εξυπηρετητή. Για παράδειγμα ο πελάτης δεν χρειάζεται να γνωρίζει τον τρόπο που χρησιμοποιεί ο εξυπηρετητής για να αποθηκεύει τα δεδομένα. Αυτές οι διαδικασίες ορίζονται εσωτερικά στον εξυπηρετητή. Αντίστοιχα ο εξυπηρετητής δεν χρειάζεται να γνωρίζει το τρόπο που παρουσιάζει ο πελάτης τα δεδομένα αυτά στο χρήστη. Χρησιμοποιώντας αυτό το διαχωρισμό ενδιαφέροντων θα μπορούσαμε να αντικαταστήσουμε τη λειτουργικότητα του εξυπηρετητή (για παράδειγμα να αλλάξουμε τον τύπο της βάσης δεδομένων) χωρίς ο πελάτης να επηρρεαστεί από αυτή τη διαδικασία. Το ίδιο ισχύει και για τον πελάτη: τροποποιώντας τη διεπαφή με τον χρήστη δεν επηρρεάζονται οι διαδικασίες του εξυπηρετητή.
2. **Χωρίς κατάσταση ή stateless** : σε καμία περίπτωση δεν πρέπει να αποθηκεύονται καταστάσεις των πόρων στον εξυπηρετητή αλλά και σε κανέναν ενδιάμεσο. Κάθε αίτημα του πελάτη πρέπει να περιλαμβάνει όλες τις απαραίτητες πληροφορίες, και αν κρατείται κάποια κατάσταση πρέπει να κρατείται στον πελάτη.



Εικόνα 7: stateful ή με κατάσταση. Ο εξυπηρετητής κρατάει τον τρέχοντα αριθμό της σελίδας(previousPage) που βλέπει ο πελάτης και υπολογίζει την επόμενη (nextPage=previousPage++)



Εικόνα 8: stateless κύκλωμα. Ο πελάτης γνωρίζοντας την τρέχουσα σελίδα, ζητά από τον εξυπηρετητή την επόμενη. Ο εξυπηρετητής δεν προβαίνει σε κανέναν υπολογισμό.

3. **Αποθήκευση πληροφοριών ή Cacheable** : Οι πελάτες πρέπει να φέρουν τη δυνατότητα να αποθηκεύουν τις απαντήσεις των εξυπηρετητών για μελλοντική χρήση. Αυτή η τεχνική χρησιμοποιείται ευρύτερα στο διαδίκτυο για να παρουσιάζει ίδια αιτήματα στους πελάτες χωρίς να αποστέλλονται στον εξυπηρετητή. Έτσι μειώνεται ο αριθμός των αλληλεπιδράσεων μεταξύ πελάτη – εξυπηρετητή.
4. **Σύστημα σχεδιασμένο σε επίπεδα ή layered system :** ο πελάτης δεν είναι σε θέση να γνωρίζει αν συνδέεται στον εξυπηρετητή

απευθείας ή με τη χρήση κάποιου ενδιάμεσου. Η σχεδίαση σε επίπεδα βοηθά στην ομοιόμορφη κατανομή του όγκου των αιτημάτων σε ένα σύνολο από εξυπηρετητές.

5. **Κώδικας κατ'απαίτηση ή code on demand (προαιρετικό) :** πολλές φορές οι εξυπηρετητές μπορούν να μεταφέρουν κομμάτια κώδικα στους πελάτες, επεκτείνοντας έτσι τη λειτουργικότητα τους. Παράδειγμα είναι τα applets, μικρές εφαρμογές που κατεβαίνουν από τον εξυπηρετητή στον πελάτη και τρέχουν ως πρόγραμμα πελάτη.
6. **Ενιαία διεπαφή ή uniform interface :** η ενιαία διεπαφή προσφέρει τη δυνατότητα κάθε μέρος του συστήματος να εξελίσσεται ανεξάρτητα.

Κάθε εφαρμογή που υλοποιεί τα παραπάνω χαρακτηριστικά αποκτά : ταχύτητα, επεκτασιμότητα, απλότητα, ευκολία στην τροποποίηση και αξιοπιστία.

2.5.5 HTTP, "καθαρά" URL's και web services

Στα προηγούμενα κεφάλαια δώσαμε τις βασικές αρχές της REST αρχιτεκτονικής και καταγράψαμε τις βασικές έννοιες και τα εμπλεκόμενα μέλη της. Πως όμως συνδέονται όλα αυτά μεταξύ τους και κυρίως πως μπορούν να χρησιμοποιηθούν για να δημιουργήσουμε web services ?

Όπως αναφέραμε, ο πόρος είναι ένα οποιοδήποτε αντικείμενο που φέρει ένα αναγνωριστικό. Στη περίπτωση του REST το αναγνωριστικό αυτό είναι ένα *URI* ή *Uniform Resource Identifier* (ενιαίο αναγνωριστικό πόρου). Το URI χρησιμοποιείται από το πρωτόκολλο HTTP. Για να επιδράσουμε πάνω στην κατάσταση ενός πόρου πρέπει να γνωρίζουμε αυτό το αναγνωριστικό. Η αλλαγή στην κατάσταση ενός πόρου θα παράγει μια *νέα αναπαράσταση* του και συμβαίνει επίσης με τη χρήση του πρωτοκόλλου HTTP. Άρα συνοψίζοντας την παραπάνω λογική καταλήγουμε

στον εξής αλγόριθμο :

1. Δίνουμε σε κάθε "πράγμα" ένα αναγνωριστικό
2. Χρησιμοποιούμε της HTTP μεθόδους για να αλλάξουμε την οντότητες του πόρου
3. Οι πόροι μας μπορούν να εκφράζονται με πολλαπλές αναπαραστάσεις
4. Η επικοινωνία γίνεται ελλείψει κατάστασης των πόρων (statelessly)

Όταν χρησιμοποιείται η αυτή η λογική τότε μπορούμε να λέμε οτι έχουμε υλοποιήσει ένα *RESTful web service*. Οι μέθοδοι HTTP συνιστούν μια ιδανική διεπαφή για αλλαγές στις οντότητες του πόρου αφού είναι αρκετά γενικές (βλ Εικόνα 9).

HTTP methods
GET
POST
PUT
DELETE

Εικόνα 9: Πίνακας HTTP μεθόδων

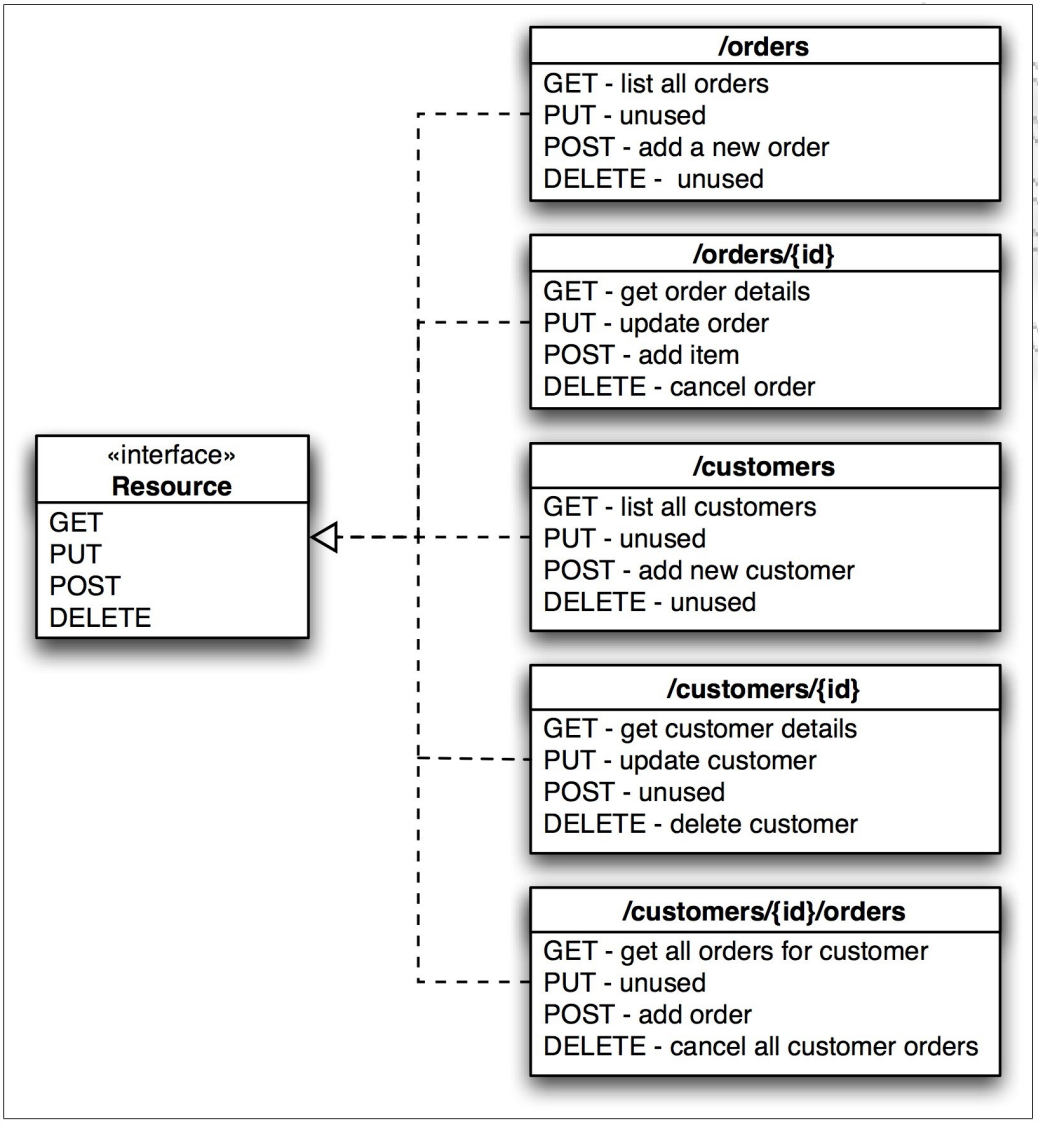
Πως επιτυγχάνεται η αλλαγή των οντοτήτων του πόρου? Χρησιμοποιώντας την αρχή *CRUD* ή *Create,Retrieve,Update* και *Delete* (δημιουργία, ανάκτηση, ανανέωση και διαγραφή), έχουμε πρόσβαση σε όλες τις βασικές λειτουργίες πάνω στον πόρο. Ακόμα και πιο σύνθετες λειτουργίες μπορούν να εκτελεστούν με τη χρήση των βασικών. Η αντιστοιχία μεταξύ των μεθόδων HTTP και της CRUD φαίνεται στον πίνακα 10 :

HTTP methods	CRUD operations
GET	Retrieve
POST	Create
PUT	Update
DELETE	Delete

Εικόνα 10: Αντιστοιχία μεθόδων HTTP - CRUD

Η μεταφορά του βασικού κορμού της πληροφορίας επιτυγχάνεται κυρίως είτε με τη χρήση XML είτε με τη χρήση JSON αντικειμένων. Αξίζει να αναφέρουμε οτι μπορούν ακόμα να χρησιμοποιηθούν και SOAP μηνύματα.

Για να δούμε πως εφαρμόζονται όλα αυτά στην πράξη ας υποθέσουμε οτι θέλουμε να εκθέσουμε μέρος της λειτουργικότητας ενός ηλεκτρονικού καταστήματος στο διαδίκτυο. Οι υπάλληλοι του καταστήματος θα μπορούν να δουν λίστες με τους πελάτες, με τις παραγγελίες που έχουν κατατεθεί αλλά και πιο σύνθετες λεπτομέρειες όπως το ιστορικό παραγγελιών ενός πελάτη. Η λειτουργικότητα αυτή μέσα από την REST αρχιτεκτονική απεικονίζεται στο παρακάτω σχήμα.

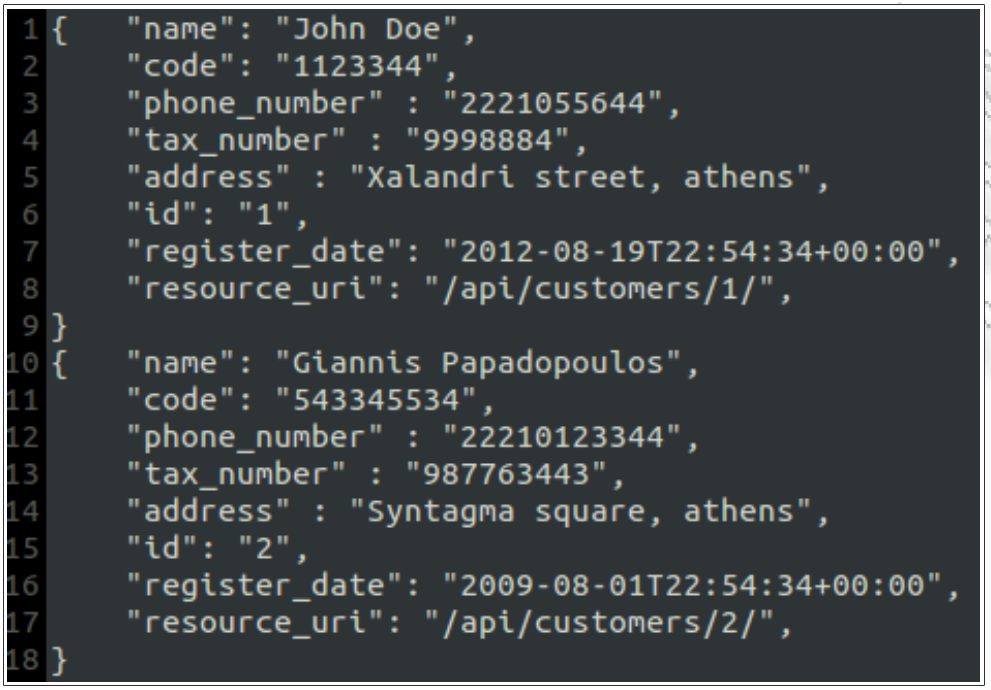


Εικόνα 11: Σχηματική αναπαράσταση των λειτουργιών ενός e-shop

Για να αποκτήσει κάποιος πρόσβαση στη λίστα των πελατών του καταστήματος πρέπει να απευθυνθεί στη διεύθυνση :

http://www.e-shop.com/customers/

και εάν η μορφοποίηση των μηνυμάτων που έχει επιλεγεί από τους σχεδιαστές του συστήματος είναι σε JSON, τότε ο χρήστης θα λάβει ένα μήνυμα απάντησης το οποίο θα είναι όπως φαίνεται παρακάτω :



Εικόνα 12: απάντηση του εξυπηρετητή σε μορφή JSON

Μπορούμε να παρατηρήσουμε οτι στην απάντηση του εξυπηρετητή περιλαμβάνεται και η διεύθυνση του πόρου : *"resource_uri" : "/api/customer/2/"* το οποίο πρακτικά σημαίνει οτι έαν θέλουμε να δούμε τις πληροφορίες που αφορούν τον πελάτη John Doe μπορούμε να πληκτρολογήσουμε κατευθείαν τη διεύθυνση :

http://www.e-shop.com/customers/2/

και η απάντηση που θα περιλαμβάνει μόνο τα στοιχεία του πελάτη που ζητήσαμε.

Αυτός είναι και ο τρόπος που γίνεται η ανάκτηση δεδομένων από τα web services. Το ίδιο ακριβώς ισχύει για όλους τους πόρους που διαθέτει το ηλεκτρονικό μας κατάστημα.

Η διαδικασία της καταχώρησης νέου πελάτη είναι και αυτή εξίσου απλή, μόνο που τώρα πρέπει να παρέχουμε τα πλήρη στοιχεία του, στο web service :

```
var data = JSON.stringify({
    "name": "Jim Newbie",
    "code": "342342",
    "phone_number" : "22210334566",
    "tax_number" : "9144331",
    "address" : "Neos kosmos, athens",
    "register_date": "2012-08-19T09:54:34+00:00",
});

$.ajax({
  url: 'http://localhost:8000/api/customers/',
  type: 'POST',
  contentType: 'application/json',
  data: data,
  dataType: 'json',
  processData: false
})
```

Εικόνα 13: Καταχώρηση νέου πελάτη με τη χρήση Javascript

Η τροποποίηση των στοιχείων ενός επιλεγμένου πελάτη γίνεται με τον ίδιο περίπου κώδικα, αλλάζοντας το όνομα της HTTP μεθόδου, χρησιμοποιώντας την μέθοδο PUT και "μιλώντας" στο URL με το αναγνωριστικό του πελάτη που καταχωρήσαμε (id = 3) :

```
var data = JSON.stringify({
    "name": "Jim Newbie",
    "code": "111111",
    "phone_number" : "22210334566",
    "tax_number" : "9144331",
    "address" : "Neos kosmos, athens",
    "register_date": "2012-08-19T09:54:34+00:00",
});

$.ajax({
  url: 'http://localhost:8000/api/customers/3',
  type: 'PUT',
  contentType: 'application/json',
  data: data,
  dataType: 'json',
  processData: false
})
```

Εικόνα 14: Τροποποίηση στοιχείων πελάτη με τη χρήση της HTTP μεθόδου PUT

Τέλος η διαγραφή του πελάτη γίνεται ως εξής :

```
$.ajax({
  url: 'http://localhost:8000/api/customers/3',
  type: 'DELETE',
})
```

Εικόνα 15: Διαγραφή πελάτη με τη χρήση της HTTP μεθόδου PUT

Είναι προφανές οτι η χρήση μιας κοινής διεπαφής, αυτή που παρέχει το HTTP πρωτόκολλο με τις μεθόδους GET, POST, PUT και DELETE, σε συνδιασμό με τα καθαρά URL's διευκολύνει σε μεγάλο βαθμό τη διαχείριση των πόρων της εφαρμογής. Αυτό δίνει μεγάλο πλεονέκτημα και στη χρήση της υπηρεσίας αλλά και στις δυνατότητες της προς επέκταση, καλύπτοντας ακόμα και πιο σύνθετες περιπτώσεις.

2.5.6 Κριτική στην αρχιτεκτονική REST

Το πλήθος της βιβλιογραφίας σχετιζόμενης με την κριτική στην αρχιτεκτονική REST είναι εξαιρετικά μικρό. Αυτό οφείλεται κυρίως στο γεγονός οτι χρησιμοποιείται ευρύτερα τα τελευταία χρόνια και επομένως ούτε έχουν αποσαφηνιστεί πλήρως οι βέλτιστες πρακτικές αλλά, ούτε και τα εργαλεία ανάπτυξης έχουν ωριμάσει σε ένα σημαντικό βαθμό. Ενδεικτικό είναι το γεγονός οτι οι εταιρίες που αναπτύσσουν εταιρικό λογισμικό προτιμούν ακόμα τα Big Web Services χαρακτηρίζοντας τα REST web services ως μια τεχνολογία ανώριμη. Σημαντικό μειονέκτημα θεωρείται η έλλεψη διαφορετικών διεπαφών, πλήρως ορισμένων, αντίστοιχων με αυτές που ορίζει το WSDL. Παρ'όλα αυτά υπάρχουν

μεγάλες εταιρίες της πληροφορικής που έχουν μεταναστεύσει τα συστήματα τους σε REST αρχιτεκτονική, με κύριο παράδειγμα τη Google.

2.5.7 Συμπεράσματα

Η διαμάχη μεταξύ των υποστηρικτών των Big Web Services και αυτών των REST web services παραμένει ακόμα στο προσκήνιο. Αμφότερες οι πλευρές συμφωνούν οτι την παρούσα χρονική στιγμή είναι σαφές οτι τα Big Web Services είναι πιο ώριμα τεχνολογικά και θα ήταν σοφό να χρησιμοποιούνται σε εταιρικές εφαρμογές ενώ τα REST web services παρ'όλο το μικρό σχετικά επίπεδο ωρίμανσης μπορούν να χρησιμοποιηθούν σε εφαρμογές που ο απαιτούμενος χρόνος υλοποίησης είναι μικρότερος και εκεί που ενδεχομένως δεν υπάρχει αντίστοιχη υποδομή.

Στα επόμενα κεφάλαια θα επιχειρήσουμε να αναπτύξουμε μια εφαρμογή σχολικής γραμματείας που θα εκθέτει μέρος των υπηρεσιών της με REST web services προς εξυπηρέτηση των μαθητών. Ένα τέτοιο σύστημα λειτουργεί επικουρικά σε ένα σύστημα διαχείρισης μαθητών (Student Information System) και δίνει πρόσβαση σε λειτουργίες που αφορούν τη φοίτηση μαθητών σε ένα εκπαιδευτικό ίδρυμα.

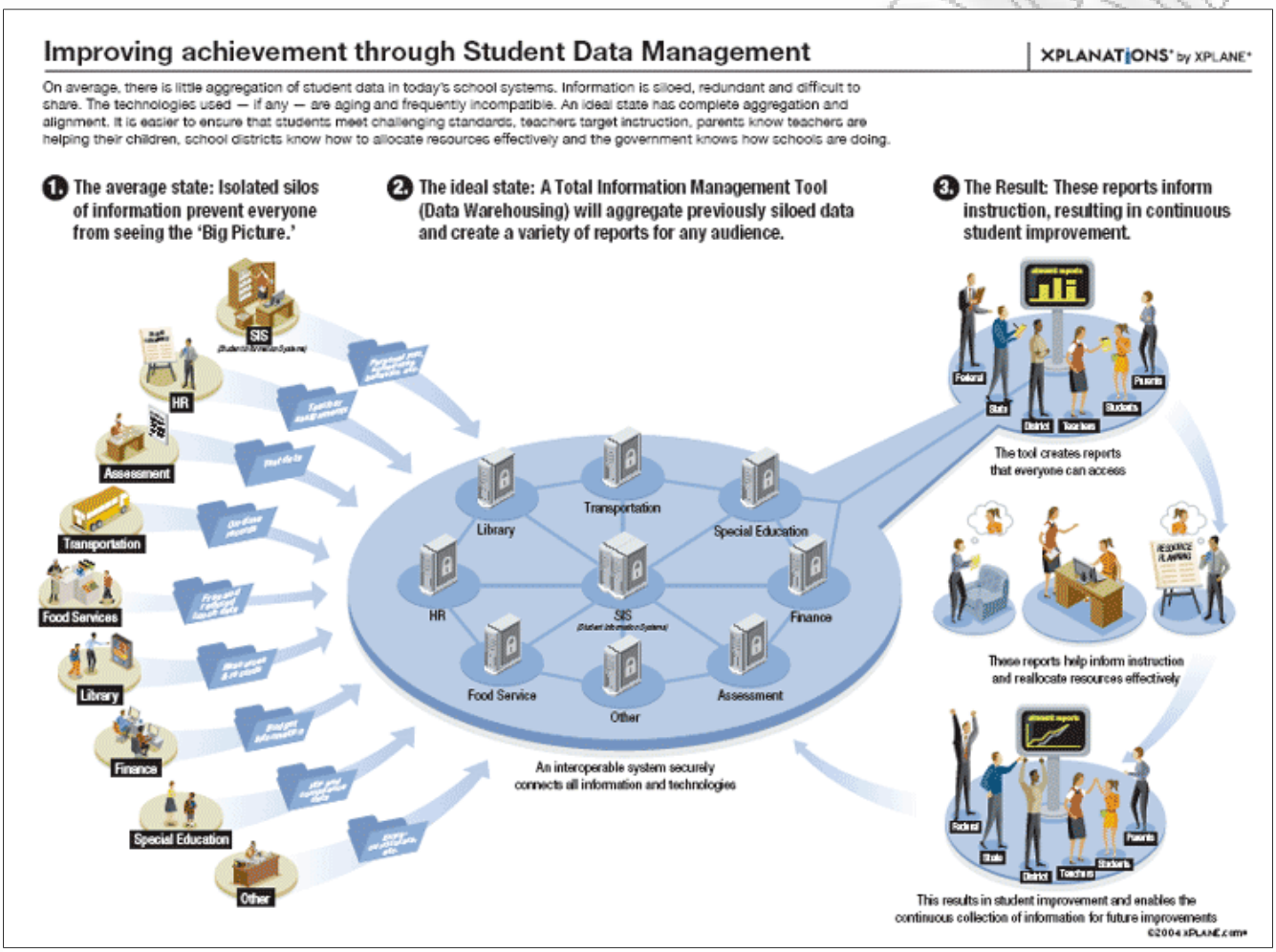
Κεφάλαιο 3

Συστήματα Διαχείρισης Μαθητών

3.1 Όρισμος

Σύστημα διαχείρισης μαθητών ονομάζεται μια εφαρμογή λογισμικού η οποία διαχειρίζεται πληροφορίες που έχουν σχέση με τους μαθητές ενός εκπαιδευτικού ιδρύματος. Ένα σύστημα διαχείρισης μαθητών παρουσιάζει παρόμοια λειτουργία με αυτή ενός συστημάτος διαχείρισης πελατών ή Customer Relationship Management. Τα συστήματα αυτά ποικίλουν σε όγκο λειτουργικότητων : από μικρές εφαρμογές που καλύπτουν τις ανάγκες ενός μικρού φροντιστηρίου, σε μεγάλες εφαρμογές διαδικτύου οι οποίες εξυπηρετούν τις ανάγκες ενός πανεπιστημίου. Μερικές από τις λειτουργίες τους είναι :

- διαχείριση αιτήσεων υποψήφιων μαθητών
- διαχείριση της διαδικασίας αιτήσεων
- δημιουργία ορολόγιου προγράμματος σχολής
- διαχείριση μαθημάτων, καθηγητών και γενικά πόρων της σχολής
- δημιουργία στατιστικών δεδομένων
- διατήρηση απουσιολόγιου
- διαχείριση πληροφοριών που αφορούν τα οικονομικά της σχολής
- καρτέλες υγείας μαθητών



Εικόνα 16: Η σημασία ενός Συστήματος διαχείρισης μαθητών (πηγή : wikipedia)

Όπως βλεπουμε, τέτοια συστήματα είναι εξαιρετικά σύνθετα σε πολυπλοκότητα μια και κάθε εκπαιδευτικό ίδρυμα παρουσιάζει δικές του ροές εργασίας και έχει τις δικές του εσωτερικές δομές.

Παρακάτω θα δούμε δύο συστήματα διαχείρισης μαθητών τα οποία είναι ανοιχτού λογισμικού, θα παρουσιάσουμε εν συντομία τις λειτουργίες τους και θα εξετάσουμε αν εκθέτουν υπηρεσίες ως web services.

3.2 Σύντομη επισκόπηση συστημάτων διαχείρισης μαθητών

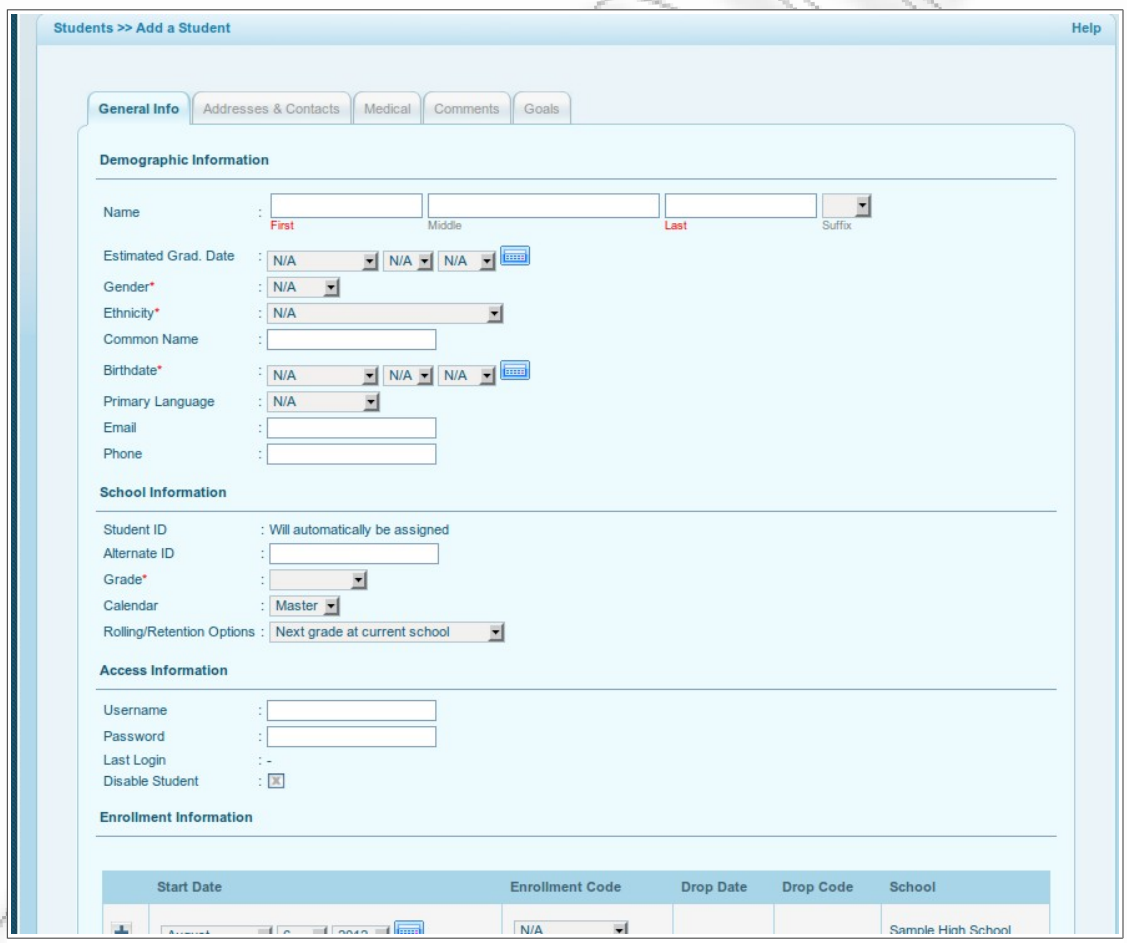
3.2.1 OpenSiS

Εικόνα 17: Η αρχική ιστοσελίδα του OpenSis

Το OpenSis είναι μια διαδικτυακή εφαρμογή ανοιχτού λογισμικού που είναι διαθέσιμη για μεγάλους εκπαιδευτικούς οργανισμούς. Το προιόν αυτό αναπτύσσεται για αρκετά χρόνια και φαίνεται πως υποστηρίζει πλήθος από λειτουργίες που μέχρι τώρα κάλυπταν εταιρικές εφαρμογές. Η εφαρμογή έχει αναπτυχθεί με τη χρήση της MySQL βάσης δεδομένων και της PHP γλώσσας προγραμματισμού. Η εφαρμογή διατίθεται δωρεάν προς κατέβασμα αλλά και προς χρήση.

Το OpenSis καλύπτει πλήθος λειτουργιών που απαιτούνται από μικρούς εως μεγάλους εκπαιδευτικούς οργανισμούς όπως :

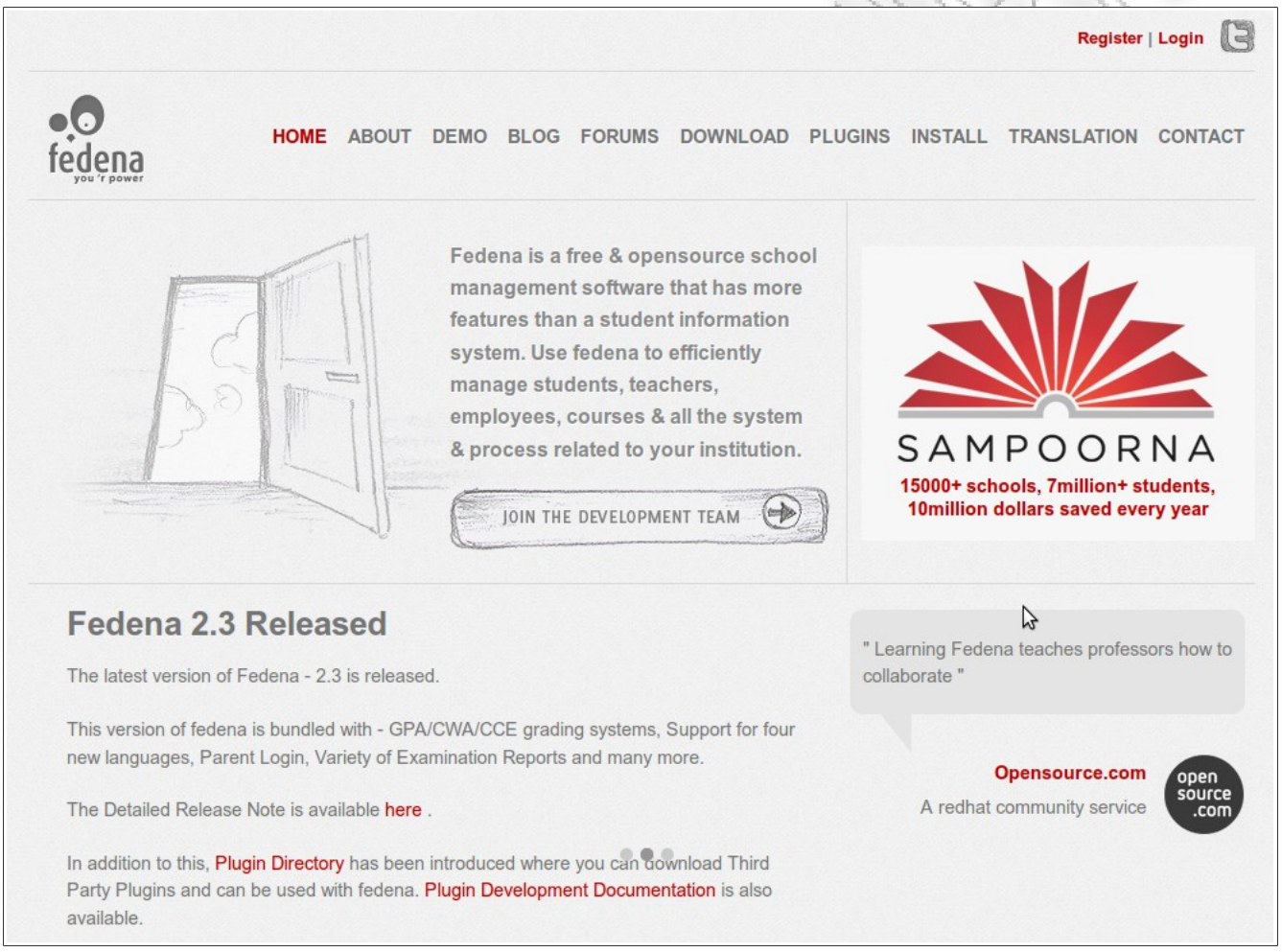
- διαχείριση λογαριασμών μαθητών
- προγραμματισμός σχολής
- κάρτες προόδου μαθητών
- πληροφορίες υγείας
- ειδικά σχεδιασμένες αναφορές, εξειδικευμένες κατά περίπτωση



Εικόνα 18: Οθόνη εγγραφής νέου μαθητή

Τέλος, να προσθέσουμε οτι από τη λειτουργικότητα του OpenSis εξαιρείται αυτή της έκθεσης γραμματειακών υπηρεσιών με web services.

3.2.2 Το project Fedena

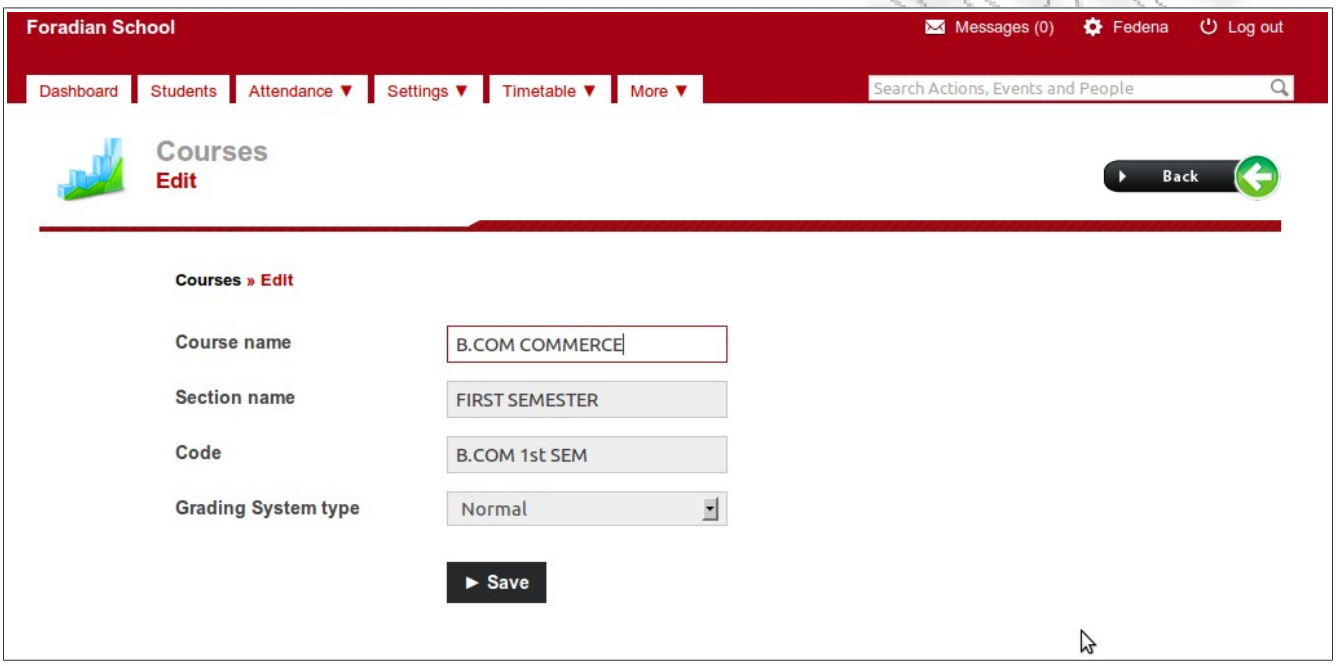


Εικόνα 19: Η αρχική σελίδα του project Fedena

Το Fedena είναι μια διαδικτυακή εφαρμογή διαχείρισης εκπαιδευτικών ιδρυμάτων, κυρίως όμως απευθύνεται σε μικρότερους οργανισμούς. Όπως και το OpenSis, είναι ανοιχτού κώδικα γραμμένη στη γλώσσα προγραμματισμού Ruby on Rails. Ανάμεσα στα χαρακτηριστικά της περιλαμβάνονται :

- διαχείριση μαθητών

- διαχείριση καθηγητών
- διαχείριση μαθημάτων
- σχεδιασμός ορολόγιου προγράμματος



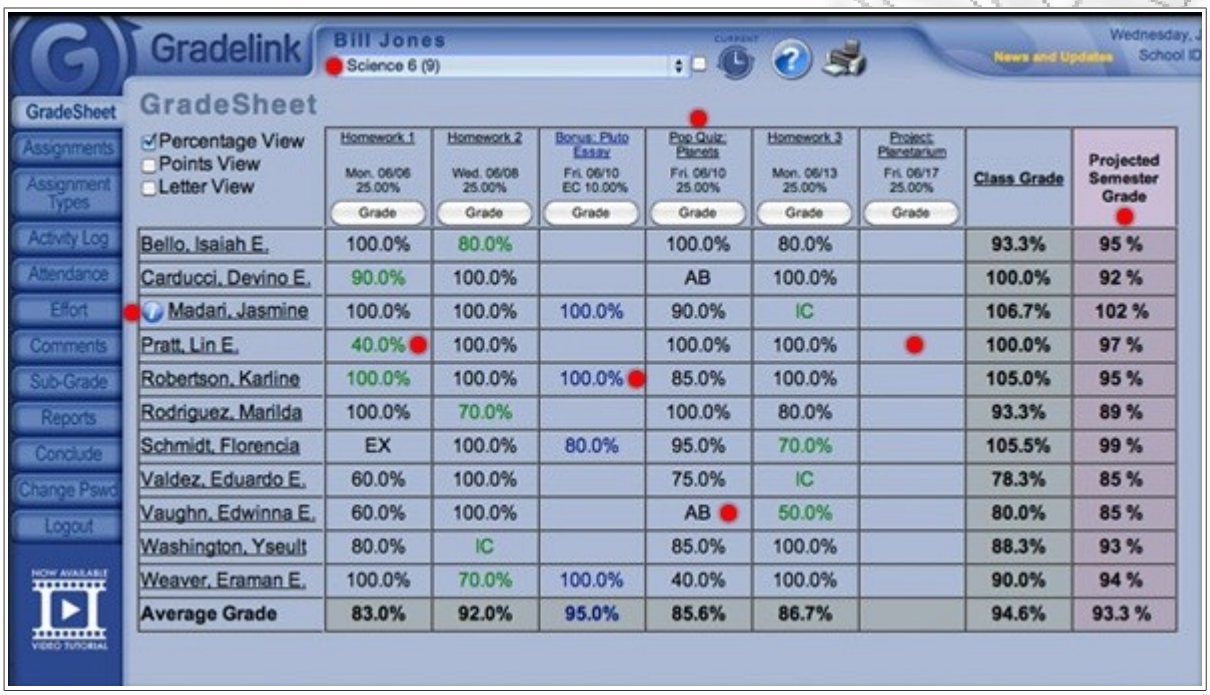
Εικόνα 20: Επεξεργασία πληροφοριών μαθήματος στο Fedena

Όπως είδαμε και στο OpenSis έτσι και εδώ δεν υποστηρίζονται Web Services για τη χρήση γραμματειακών υπηρεσιών από τους φοιτητές.

3.2.3 Gradelink

Το Gradelink είναι και αυτό μια διαδικτυακή εφαρμοφή η οποία απευθύνεται σε μικρούς και μεγάλους εκπαιδευτικούς οργανισμούς. Το μεγαλύτερο τμήμα αυτής της εφαρμογής είναι χτισμένο και βασισμένο στο Flash, οπότε οι εκάστοτε φυλλομετρητές που το τρέχουν θα πρέπει να υποστηρίζουν flash. Διαθέτει φιλικό προς το χρήστη περιβάλλον και είναι εμπλουτισμένο με αρκετές λειτουργικότητες.

Το Gradelink είναι κλειστού κώδικα εμπορική εφαρμογή.



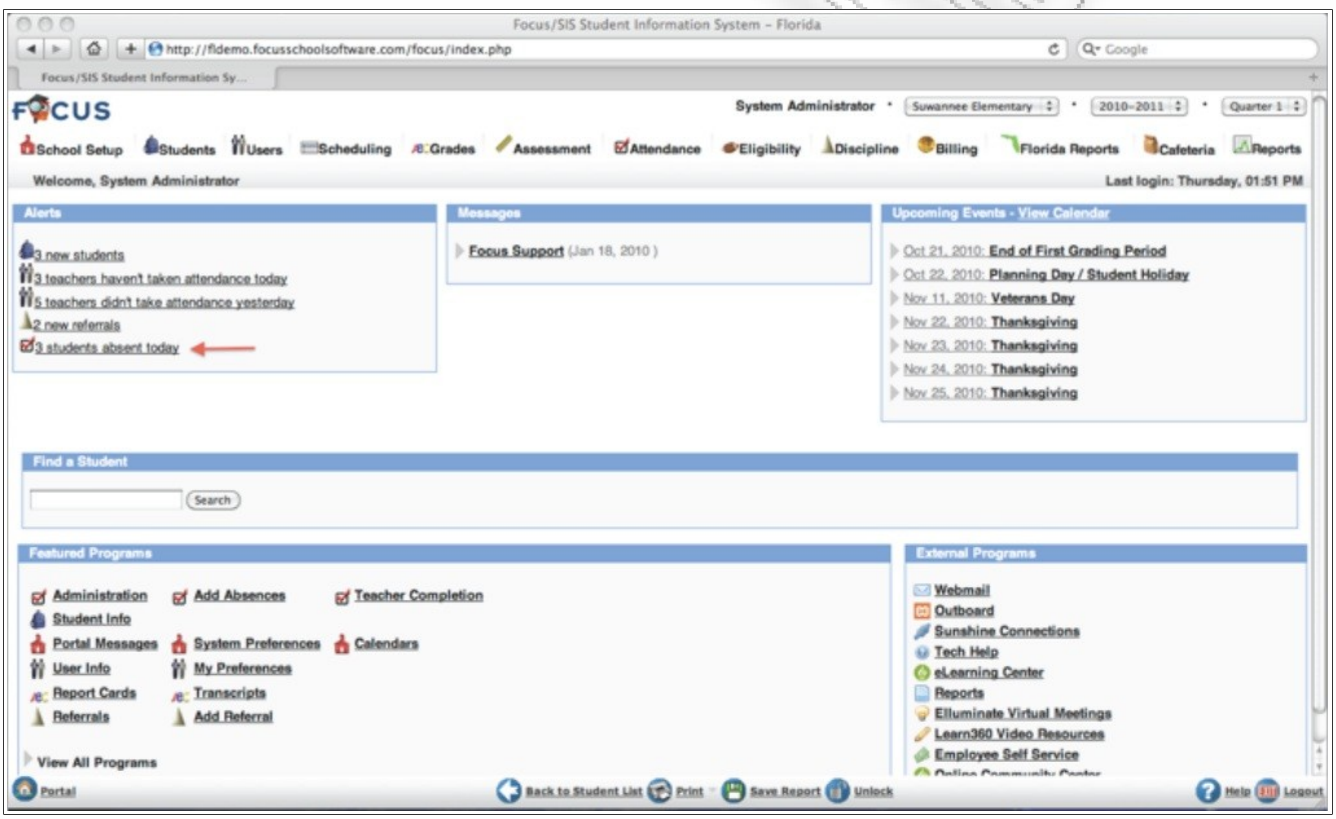
Εικόνα 21: Διαχείριση βαθμολογιών στο Gradelink

Διαθέσιμα χαρακτηριστικά:

1. διαχείριση μαθητών
2. διαχείριση αιθουσών
3. κάρτες προόδου μαθητών
4. επικοινωνία με γονείς
5. πληροφορίες υγείας
6. εισαγωγή και εξαγωγή δεδομένων σε μορφή excel

3.2.4 SIS – Student Information System

Το SIS είναι μια διαδικτυακή εφαρμοφή η οποία απευθύνεται κυρίως σε μικρούς εκπαιδευτικούς οργανισμούς. Αποτελείται απο αρκετές λειτουργικές μονάδες, όπου η κάθε μια εκ αυτών υλοποιεί συγκεκριμένες ροές εργασίων αναγκαίες για εκπαιδευτικους οργανισμούς, όσον αφορά την διαχείρηση μαθητών, αιθουσών, καθηγητών.



Εικόνα 22: Το σύστημα SIS

Διαθέσιμα χαρακτηριστικά:

1. διαχείριση μαθητών
2. διαχείριση αιθουσών

3. κάρτες προόδου μαθητών
4. επικοινωνία με γονείς
5. απουσιολόγιο
6. δυνατότητα λογαριασμών γονέων για παρακολούθηση βαθμολογιών
7. δυνατότητα δημιουργίας εβδομαδιαίου ορολόγιου προγράμματος

3.3 Σύγκριση συστημάτων διαχείρισης μαθητών

Λαμβάνοντας υπόψη τις λειτουργίες των παραπάνω λογισμικών, προκύπτει ο παρακάτω συνοπτικός πίνακας συγκρίσεως των συστημάτων.

Εργαλεία	**OpenSIS**	**Fedena**	**Gradelink**	**SIS**
Λειτουργίες				
Διαχείριση Μαθητών	✔	✔	✔	✔
Διαχείριση Καθηγητών	✔	✔	✘	✘
Κάρτες προόδου μαθητών	✔	✘	✔	✔
Απουσιολόγιο	✘	✘	✘	✔
Δημιουργία ορολόγιου προγράμματος	✘	✔	✘	✔
Πληροφορίες υγείας μαθητών	✔	✘	✔	✘
Πολύγλωσσο	✘	✔	✘	✘
Βασισμένο στον ιστό	✔	✔	✔	✔
Εισαγωγή και εξαγωγή δεδομένων σε excel	✘	✘	✔	✘
Δυνατότητα διασύνδεσης του συστήματος με web-services	✘	✘	✘	✘

3.4 Συμπεράσματα

Η αναζήτηση μας στο διαδίκτυο υπέδεξε έναν μεγάλο αριθμό συστημάτων διαχείρισης μαθητών. Ο λόγος παρουσίασης των παραπάνω αφορά στην επισκόπηση καθιερωμένων λειτουργιών που εκτελούνται στα πλαίσια ενός εκπαιδευτικού ιδρύματος, όπως για παράδειγμα την εγγραφή ενός νέου μαθητή. Ένας ακόμα λόγος που επιλέχθηκαν τα παραπάνω συστήματα είναι γιατί ήταν τα μοναδικά που διέθεταν δοκιμαστικούς λογαριασμούς (demo accounts) και μας κατέστη δυνατό να προσεγγίσουμε τη λειτουργικότητα τους με μεγαλύτερη λεπτομέρεια. Η ευρεία χρήση τους είναι ένας ακόμα λόγος παρουσίασης, διότι υποδεικνύει οτι μεγάλος αριθμός εκπαιδευτικών ιδρυμάτων καλύπτεται από τη λειτουργικότητα που υποστηρίζουν. Παρ'όλα αυτά η λειτουργικότητα τους έχει περιορισμούς. Κανένα από τα παραπάνω συστήματα δεν υποστηρίζει web services με σκοπό να εκθέσει μέρος των λειτουργιών του στους χρήστες.

Κατά συνέπεια, υπάρχει χώρος για την ανάπτυξη μιας εφαρμογής η οποία θα είναι σε θέση να καλύπτει βασικές λειτουργίες διαχείρισης ενός εκπαιδευτικού ιδρύματος αλλά και, με τη χρήση web services, να επιτρέπει στην κοινότητα της να χρησιμοποιεί μέρος των λειτουργιών της για καλύτερη και πιο γρήγορη εξυπηρέτηση, αφαιρώντας έτσι φόρτο εργασίας από το προσωπικό. Η εφαρμογή που προτείνουμε εστιάζει σε αυτούς του τομείς. Στα κεφάλαια που ακολουθούν θα δείξουμε πως υλοποιείται ένα τέτοιο σύστημα παρουσιάζοντας τα απαιτούμενα εργαλεία, τις περιπτώσεις χρήσης αλλά και ένα πρόγραμμα κινητού τηλεφώνου το οποίο επικοινωνεί με τα web services.

Κεφάλαιο 4

Μεθοδολογία ανάλυσης & σχεδίασης της εφαρμογής με Web Services

4.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζεται η ανάλυση του συστήματος. Για την δημιουργία του, χρειάζεται πρωτίστως να γίνει μια περιγραφή των αναγκών και απαιτήσεων του, καταγραφή των λειτουργικών και μη λειτουργικών απαιτήσεων καθώς επίσης και των χρηστών του συστήματος. Τέλος θα δούμε και τις περιπτώσεις χρήσης του συστήματος από τους χρήστες.

4.2 Χρήστες

Οι χρήστες του συστήματος είναι οι άνθρωποι εκείνοι οι οποίοι θα χρησιμοποιούν το σύστημα και κατά περίπτωση θα τους επιτρέπεται να έχουν δικαιώματα να εκτελέσουν ένα σύνολο από εργασίες. Οι χρήστες στην περίπτωση του δικού μας συστήματος χωρίζονται σε δύο κατηγορίες :

- προσωπικό της γραμματείας
- μαθητές

Είναι σαφές οτι οι δύο παραπάνω κατηγορίες ενδιαφέρονται να αλληλεπιδράσουν

με το σύστημα με εντελώς διαφορετικό τρόπο. Επιπρόσθετα, στο δικό μας σύστημα αλληλεπιδρούν και μέσα από διαφορετικές πλατφόρμες, χρησιμοποιώντας διαφορετικού τύπου συσκευές.

4.2.1 Οι χρήστες της γραμματείας

Οι χρήστες της γραμματείας είναι ουσιαστικά το διοικητικό προσωπικό που εργάζεται στη γραμματεία ενός εκπαιδευτικού ιδρύματος. Η συγκεκριμένοι χρήστες αλληλεπιδρούν με το σύνολο του συστήματος και ουσιαστικά έχουν εξουσιοδότηση στο σύνολο των ενεργειών του. Αναλυτικά ένας χρήστης της γραμματείας μπορεί :

1. να διαχειριστεί καρτέλες μαθητών
2. να διαχειριστεί τα μαθήματα
3. να διαχειριστεί πληροφορίες που αφορούν την τρέχουσα κατάσταση του ιδρύματος όπως για παράδειγμα να κάνει τροποποιήσεις ή και ακόμα να συντάξει το ορολόγιο πρόγραμμα της σχολής.
4. Να δημοσιεύσει ανακοινώσεις προς τους μαθητές.

4.2.2 Οι χρήστες μαθητές

Οι μαθητές έχουν περιορισμένα δικαιώματα σε σχέση με τη γραμματεία. Οι ενέργειες που μπορούν να εκτελέσουν περιορίζονται σε :

1. προβολή ειδοποιήσεων της σχολής
2. προβολή προσωπικής καρτέλας
3. προβολή μαθημάτων που παρακολουθούνται την τρέχουσα σχολική περίοδο
4. υποβολή αίτησης προς τη γραμματεία για την χορήγηση κάποιας

πιστοποίησης

4.3 Απαιτήσεις συστήματος

Με τον όρο απαίτηση συστήματος εννοούμε την εκάστοτε λειτουργία που πρέπει να αντιστοιχεί σε κάποιον χρήστη. Όπως είδαμε παραπάνω, οι τύποι χρηστών είναι δύο και επομένως οι λειτουργίες που αντιστοιχούν στην κάθε ομάδα είναι διαφορετικές. Οι απαιτήσεις ενός συστήματος χωρίζονται σε δύο κατηγορίες, τις λειτουργικές και τις μη λειτουργικές απαιτήσεις. Επιπλεόν από τις προαναφερθείσες απαιτήσεις προκύπτουν και οι τεχνικές απαιτήσεις που σχετίζονται με το τι τεχνικό εξοπλισμό ή τι εργαλεία θα χρειαστούμε προκειμένου να καταστεί εφικτή η υλοποίηση των απαιτήσεων.

4.3.1 Λειτουργικές απαιτήσεις συστήματος

Οι λειτουργικές απαιτήσεις του συστήματος είναι οι εξής :

- δημιουργία καρτέλας νέου μαθητή
- διαχείριση καρτέλας μαθητή παρέχοντας δυνατότητες τροποποιήσης είτε ακόμα και πλήρους διαγράφης
- δημιουργία ανακοινώσεων σχολής
- προβολή ανακοινώσεων σχολής
- δημιουργία νέων μαθηματων
- εγγραφή μαθητών σε μαθήματα
- δημιουργία ειδικών εγγράφων π.χ. Πιστοποιήσεις παρακολούθησης, αναλυτικές βαθμολογίες.

- Διαχείριση αιτήσεων μαθητών
- διαχείριση βαθμολογιών

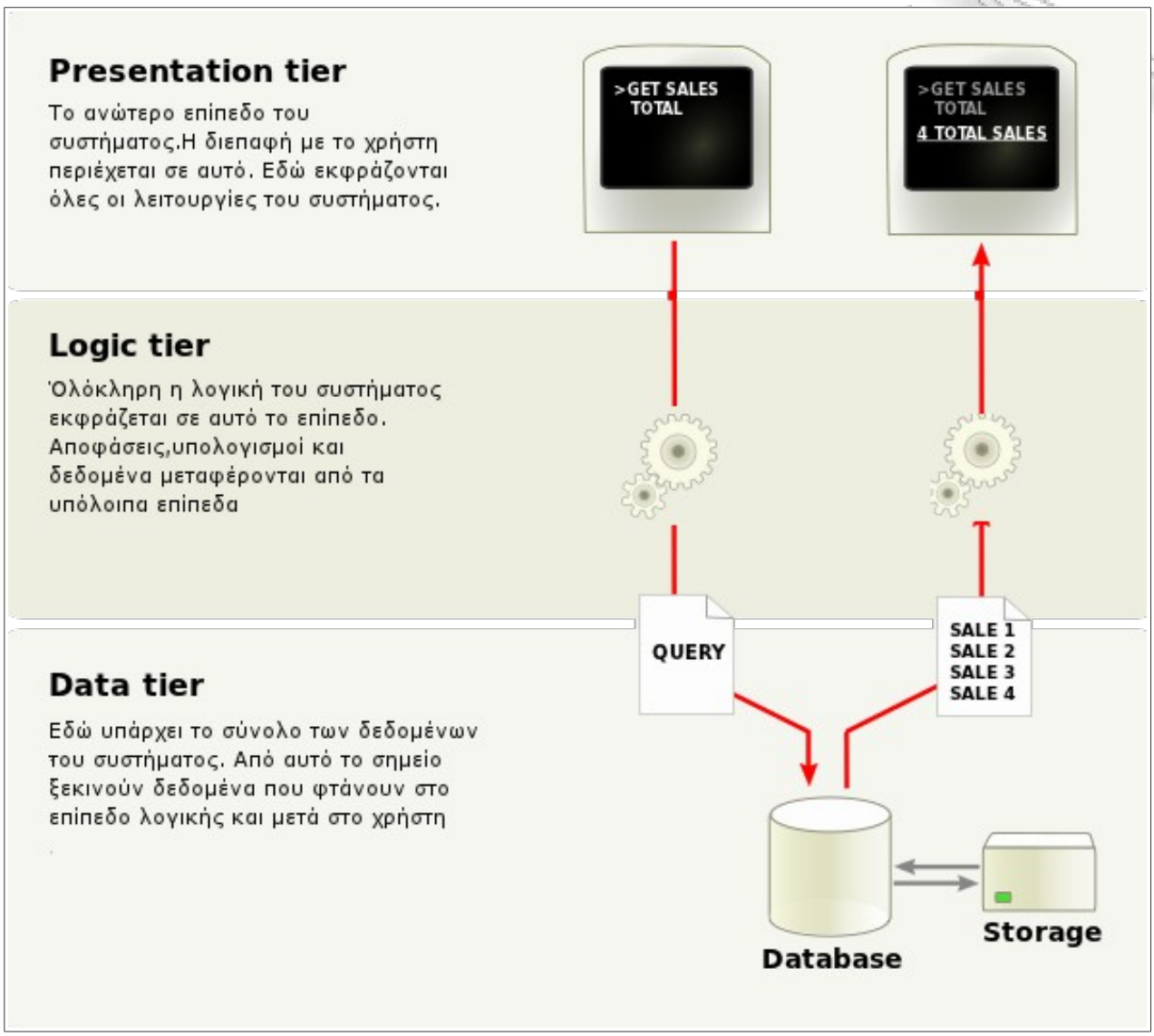
4.3.2 Μη λειτουργικές απαιτήσεις συστήματος

Μη λειτουργικές απαιτήσεις του συστήματος είναι οι εξής :

- Το σύστημα να λειτουργεί αδιάλειπτα.
- Η πρόσβαση στη λειτουργία του συστήματος να δίνεται μόνο σε εξουσιοδοτημένους χρήστες.
- τα στοιχεία των χρηστών δε πρέπει να εκτίθονται.
- Το σύστημα θα πρέπει να λειτουργεί στο διαδίκτυο
- Το σύστημα πρέπει να είνα εύχρηστο και γρήγορο.
- Το σύστημα θα πρέπει να μπορεί να εξυπηρετήσει μεγάλο αριθμό ταυτόχρονα συνδεδεμένων χρηστών (scalability).

4.4 Η αρχιτεκτονική του συστήματος Dreskt

Η σχεδίαση του συστήματος ακολουθεί τη λογική της πολυεπίπεδης αρχιτεκτονικής. Σύμφωνα με αυτή τη λογική τα λειτουργικά μέρη ενός συστήματος διαχωρίζονται σε ένα αριθμό από επίπεδα, όπου κάθε επίπεδο είναι αυτοτελές και εξειδικεύεται στην υποστήριξη συγκεκριμένης λειτουργικότητας. Με αυτό το τρόπο επιτυγχάνεται πιο εύκολα ο σχεδιασμός και η υλοποίηση μεγάλων εφαρμογών. Επίσης διευκολύνεται και η συντήρηση αλλά και η επεκτασιμότητα τους.



Εικόνα 23: Πολυεπίπεδη σχεδίαση συστημάτων

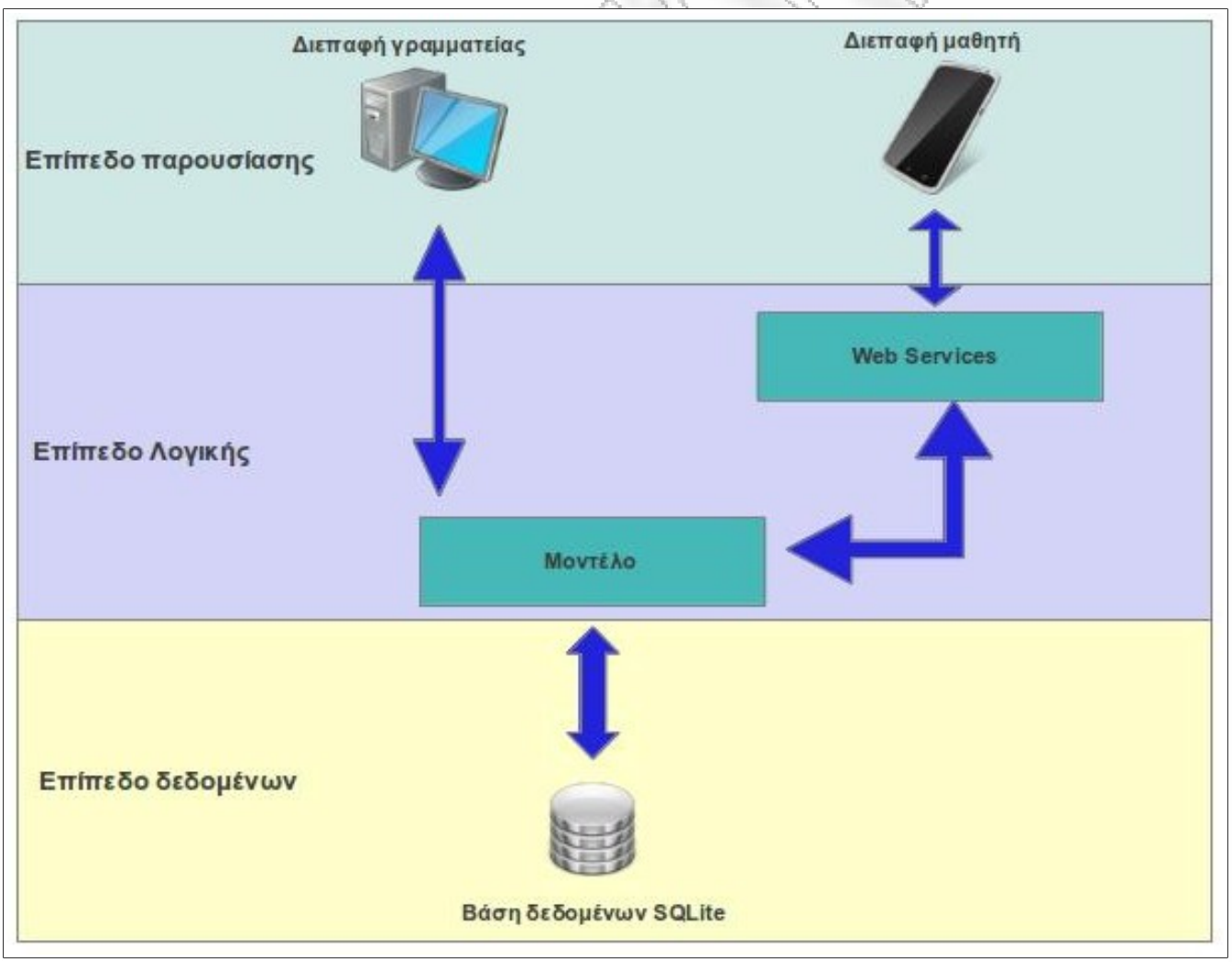
Η βάση του συστήματος βρίσκεται στο επίπεδο δεδομένων ή **Data Tier.** Η κύρια οντότητα εδώ είναι η βάση δεδομένων. Η βάση μπορεί να είναι οποιουδήποτε τύπου και επικοινωνία με αυτήν γίνεται με τη γλώσσα SQL. Στη βάση δεδομένων αποθηκεύονται όλες οι πληροφοριές του συστήματος.

Στο επίπεδο της λογικής, ευρύτερα γνωστό ώς **Logic Tier** ή **Business Tier** υπάρχει το μοντέλο του συστήματος, δηλαδή οι διακριτές οντότητες-αντικείμενα από τα οποία αποτελείται το σύστημα. Εδώ καθορίζονται οι σχέσεις και οι

λειτουργίες τους αλλά και οι κανόνες χειρισμού των δεδομένων.

Τέλος στο ανώτερο επίπεδο της αρχιτεκτονικής βρίσκεται το επίπεδο παρουσίασης ή **Presentation Tier** το οποίου ουσιαστικά είναι η διεπαφή του συστήματος. Εδώ οι κύριες λειτουργίες εκφράζονται με οπτικό, συνήθως, τρόπο στο χρήστη. Το επίπεδο λογικής οπτικοποιείται και όλη η αλληλεπίδραση του χρήστη με αυτό γίνεται μέσω της διεπαφής και μέσω των συσκευών εισόδου της εκάστοτε συσκευής.

Αν προσαρμόσουμε την παραπάνω αρχιτεκτονική στα δικά μας δεδομένα μπορούμε να συνοψίσουμε στο παρακάτω διάγραμμα :

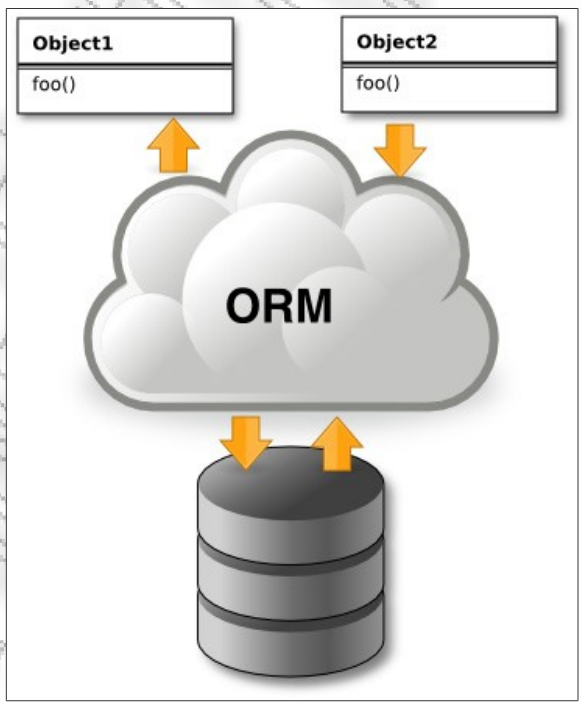


Εικόνα 24: Η αρχιτεκτονική του συστήματος Dreskt

4.5 Ανάλυση των δομικών μερών του συστήματος

4.5.1 Βάση δεδομένων

Στο χαμηλότερο επίπεδο του συστήματος βρίσκεται η βάση δεδομένων. Όπως προαναφέραμε όλα τα δεδομένα του συστήματος βρίσκονται στη βάση δεδομένων. Αυτό ακριβώς το χαρακτηριστικό είναι που την κάνει μια ακριβή αποτύπωση του μοντέλου του συστήματος, δηλαδή σχεδόν για κάθε ένα αντικείμενου του μοντέλου υπάρχει και ένας αντίστοιχος πίνακας στη βάση δεδομένων. Με αυτό το τρόπο διευκολύνεται κατά πολύ ο τρόπος προγραμματισμού και χειρισμού της βάσης δεδομένων. Στην περίπτωση του συστήματος Dreskt έχει γίνει ένα ακόμα βήμα προς αυτή τη κατεύθυνση : έχουμε χρησιμοποιήσει το πρότυπο **ORM** που μας παρέχει η γλώσσα προγραμματισμού Python. Το πρότυπο ORM ή **object relational mapping** μας επιτρέπει να συνδέσουμε με ευθύ τρόπο το μοντέλο μας με τη βάση δεδομένων.

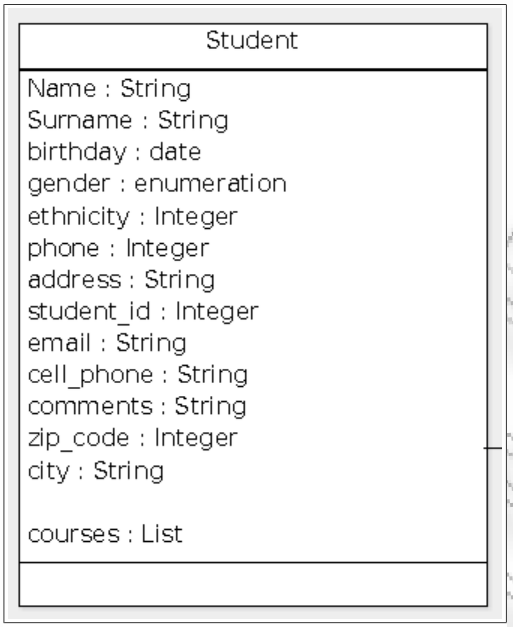


Εικόνα 25: Το ORM πρότυπο

Όπως βλέπουμε και στην εικόνα 25 το ORM λειτουργεί ανάμεσα από το μοντέλο και τη βάση δεδομένων ως ένας διαμεσολαβητής που αποκρύπτει λεπτομέρειες που αφορούν τη βάση δεδομένων. Για παράδειγμα αποκρύπτει από τον χρήστη του μοντέλου με ποιο τρόπο γίνεται η αποθήκευση στη βάση ή τι τύπος είναι η βάση δεδομένων που χρησιμοποιείται. Το προγραμματιστικό πλαίσιο που χρησιμοποιήσαμε για τη χρήση του προτύπου ORM αναλαμβάνει και τη δημιουργία των πινάκων στη βάση δεδομένων. Πρακτικά αυτό σημαίνει οτι αν θελήσουμε μπορούμε με τη χρήση του προτύπου να μεταφέρουμε το σύστημα Dreskt σε οποιοδήποτε τύπο βάσης δεδομένων επιθυμούμε. Για τους σκοπούς αυτής της εργασίας χρησιμοποιούμε την SQLite ως κεντρική βάση δεδομένων.

4.5.2 Το μοντέλο

Το μοντέλο αποτελεί την "καρδιά" του συστήματος. Απεικονίζεται πλήρως και με σαφήνεια κάθε μέρος της λειτουργικότητας του και των οντοτήτων που το απαρτίζουν. Στον αντικειμενοστρεφή προγραμματισμό το μοντέλο αντιστοιχεί στο μοντέλο κλάσεων ενός συστήματος. Κάθε οντότητα του μοντέλου είναι μια κλάση με τις δικές της μεταβλητές και συναρτήσεις που επιδρούν επάνω σε αυτές. Για παράδειγμα η οντότητα του μαθητή, στο σύστημα Dreskt, αντιστοιχεί στην κλάση Student η οποία έχει ως φαίνεται στην εικόνα 26.



Εικόνα 26: Η κλάση Student

Όλα τα δεδομένα του συστήματος περνάνε από εδώ καθώς μεταφέρονται από το επίπεδο παρουσίασης στη βάση δεδομένων, αφού βέβαια υποστούν την ανάλογη επεξεργασία και τον έλεγχο-επικύρωση που απαιτείται.

Τα web services είναι μέρος του μοντέλου. Όπως για κάθε οντότητα υπάρχει μια κλάση έτσι και για κάθε οντότητα υπάρχει ένα **πόρος** (resource) πράγμα το οποίο επιτάσσει η αρχιτεκτονική REST. Για την κλάση Notification η οποία αφορά την οντότητα ειδοποιήσεις υπάρχει ο πόρος NotificationResource ο οποίος την εκθέτει σαν web service στους εξουσιοδοτημένους πελάτες (βλ. Εικόνα 28).

```
class Notification(models.Model):
    text = models.TextField(max_length=500)
    subject = models.CharField(max_length=40)
    date_created = models.DateTimeField('Created',default=datetime.now())
    expiration_date = models.DateTimeField('Expires')

    def __unicode__(self):
        return self.subject
```

Εικόνα 27: Ο ορισμός της οντότητας Notification

```
class NotificationResource(ModelResource):

    class Meta:
        start_date = datetime.now()
        end_date = datetime(2013,12,20)

        queryset = Notification.objects.filter(expiration_date__gte=start_date,expiration_date__lte=end_date)
        resource_name = 'notifications'
        include_resource_uri = True

    '''
    def dehydrate(self,bundle):
        #return notification text only
        return bundle.data['text']
    '''
    def alter_list_data_to_serialize(self,request,data_dict):
        print "----------------"
        #print request.META["HTTP_USER"]
        if isinstance(data_dict, dict):
            if 'meta' in data_dict:
                # Get rid of the "meta".
                del(data_dict['meta'])
                # Rename the objects.
                data_dict['notifications'] = data_dict['objects']
                del(data_dict['objects'])
        return data_dict
```

Εικόνα 28: Η οντότητα Notification εκφρασμένη ως πόρος

Με βάση επίσης την αρχιτεκτονική REST κάθε πόρος πρέπει να φέρει μια χαρακτηριστική διεύθυνση την οποία πρέπει να ορίσουμε (βλ. Εικόνα 28). Επομένως το υπερσύνολο των πόρων που ορίζουμε χρησιμοποιούν το μοντέλο και το επεκτείνουν εκτίθοντας μέρος της λειτουργικότητας του και προσφέρουν μια διεπαφή για χρήση από υπολογιστές. Η διεπαφή μπορεί να χρησιμοποιηθεί από οποιαδήποτε γλώσσα προγραμματισμού αρκεί η πλευρά του πελάτη να ανταποκρίνεται στις συμβάσεις που έχουν καθοριστεί από τον δημιουργό του προγράμματος, και ονομάζεται **Web API** (application programming interface) ή διαδικτυακή διεπαφή προγραμματισμού εφαρμογής.

Η μορφή που επιλέξαμε να εκφράσουμε τους πόρους στα web services είναι τα **JSON** ή **Javascript Notation objects**. Το JSON είναι ένας ελαφρύς τύπος ανταλλαγής δεδομένων. Είναι ευανάγνωστο από τον άνθρωπο και είναι εύκολο για έναν υπολογιστή να το διατρέξει. Αποτελεί υποσύνολο της γλώσσας προγραμματισμού Javascript. Είναι ουσιαστικά απλό κείμενο δομημένο με έναν συγκεκριμένο τρόπο :

[*λίστα μεταβλητών-τιμών*] :

{

[*όνομα μεταβλητής*]: [*τιμή μεταβλητής*]

}

Το κυριότερο πλεονέκτημα του είναι οτι το μικρό του μέγεθος το καθιστά ευέλικτο στη μεταφορά ενώ δεν επιβαρύνει τον web server με μεγαλύτερο όγκο όπως η XML[12]. Αυτός είναι και ο λόγος που μεγάλες εταιρίες λογισμικού έχουν αρχίσει να το χρησιμοποιούν ευρύτατα.

	JSON	XML
Trial 1 Number Of Objects	20000	20000
Trial 1 Total Time (ms)	2213.15	61333.68
Trial 1 Average Time (ms)	0.11	3.07
Trial 2 Number Of Objects	40000	40000
Trial 2 Total Time (ms)	3127.99	123854.59
Trial 2 Average Time (ms)	0.08	3.10
Trial 3 Number Of Objects	60000	60000
Trial 3 Total Time (ms)	4552.38	185936.27
Trial 3 Average Time (ms)	0.08	3.10
Trial 4 Number Of Objects	80000	80000
Trial 4 Total Time (ms)	6006.72	247639.81
Trial 4 Average Time (ms)	0.08	3.10
Trial 5 Number Of Objects	100000	100000
Trial 5 Total Time (ms)	7497.36	310017.47
Trial 5 Average Time (ms)	0.07	3.10

Εικόνα 29: Συγκριτικός πίνακας χρόνου απόκρισης ενός web server μετά από αποστολή Ν αντικειμένων JSON και XML.

12 Για μια αναλυτικότερη σύγκριση μεταξύ των δύο τύπων δεδομένων βλέπε "*Comparison of JSON and XML Data Interchange Formats: A Case Study "* από τους Nurzhan Nurseitov, Michael Paulson, Randall Reynolds, Clemente Izurieta .

Στο σύστημα Dreskt το μοντέλο έχει υλοποιηθεί με τη γλώσσα προγραμματισμού Python και τη χρήση της βιβλιοθήκης Django που διευκολύνει στην ανάπτυξη εφαρμογών διαδικτύου. Επίσης έχει χρησιμοποιηθεί η βιβλιοθήκη tastypie που δρα επικουρικά με το Django στη δημιουργία REST web services.

4.5.4 Το επίπεδο παρουσίασης & διεπαφή χρήστη

Το επίπεδο παρουσίασης είναι το υψηλότερο επίπεδο στην πολυεπίπεδη αρχιτεκτονική. Ο χρήστης έρχεται σε επαφή με το μοντέλο μέσα από τη γραφική διεπαφή. Η γραφική διεπαφή του δίνει πρόσβαση σε όλο το εύρος της λειτουργικότητας του συστήματος διευκολύνοντας τον μέσα από ένα φιλικό και κατανοητό περιβάλλον. Το σύστημα Dreskt παρουσιάζει ουσιαστικά δύο γραφικές διεπαφές. Η πρώτη αφορά το περιβάλλον των χρηστών της γραμματείας και είναι υλοποιημένη με τη χρήση διαδικτυακών τεχνολογιών (HTML,Javascript) ενώ η δεύτερη είναι αφορά τους μαθητές και είναι υλοποιημένη στο περιβάλλον Android (βλ. Εικόνα 24).

Η γραφική διεπαφή των χρηστών της γραμματείας είναι προσαρμοσμένη σε περιβάλλον γραφείου. Οι χρήστες με τη βοήθεια του ηλεκτρονικού τους υπολογιστή μπαίνουν σε μια ιστοσελίδα και αφού εισάγουν τα αναγνωριστικά τους τότε μπορούν να χρησιμοποιήσουν το σύστημα. Μερικές από τις λειτουργίες που εκτελούνται εδώ είναι καταχώρηση νέας καρτέλας μαθητή, δημιουργία μαθήματος κ.α. Η γραφική διεπαφή είναι προσαρμοσμένη έτσι ώστε να μπορούν να δουλέψουν γρήγορα και αποδοτικά, και είναι βασισμένη σε φόρμες HTML.

Σε αντίθεση με τους χρήστες της γραμματείας οι μαθητές χρησιμοποιούν ένα γραφικό περιβάλλον με προσανατολισμό στην ευχρηστία και την απλότητα που παρέχουν τα smartphones έχοντας επίσης σα δεδομένη την εξοικείωση που έχουν

οι νεότεροι σε ηλικία χρήστες με αυτά. Με αυτό το τρόπο έχουν πρόσβαση σε ανακοινώσεις της σχολής που φοιτούν, μπορούν να δούν βαθμολογίες κ.α. Οι μαθητές είναι ουσιαστικά και οι κύριοι χρήστες των web services του συστήματος διότι χρησιμοποιούνται από τον πελάτη που υλοποιήσαμε στην πλατφόρμα Android.

Για τη δημιουργία της γραφικής διεπαφής της γραμματείας χρησιμοποιήθηκε το Django admin interface το οποίο αποτελεί μέρος του Django framework ενώ για αυτή των μαθητών χρησιμοποιήσαμε το Android-SDK το οποίο αποτελεί την καρδιά του προγραμματιστικού περιβάλλοντος για όλα τα Android smartphones.

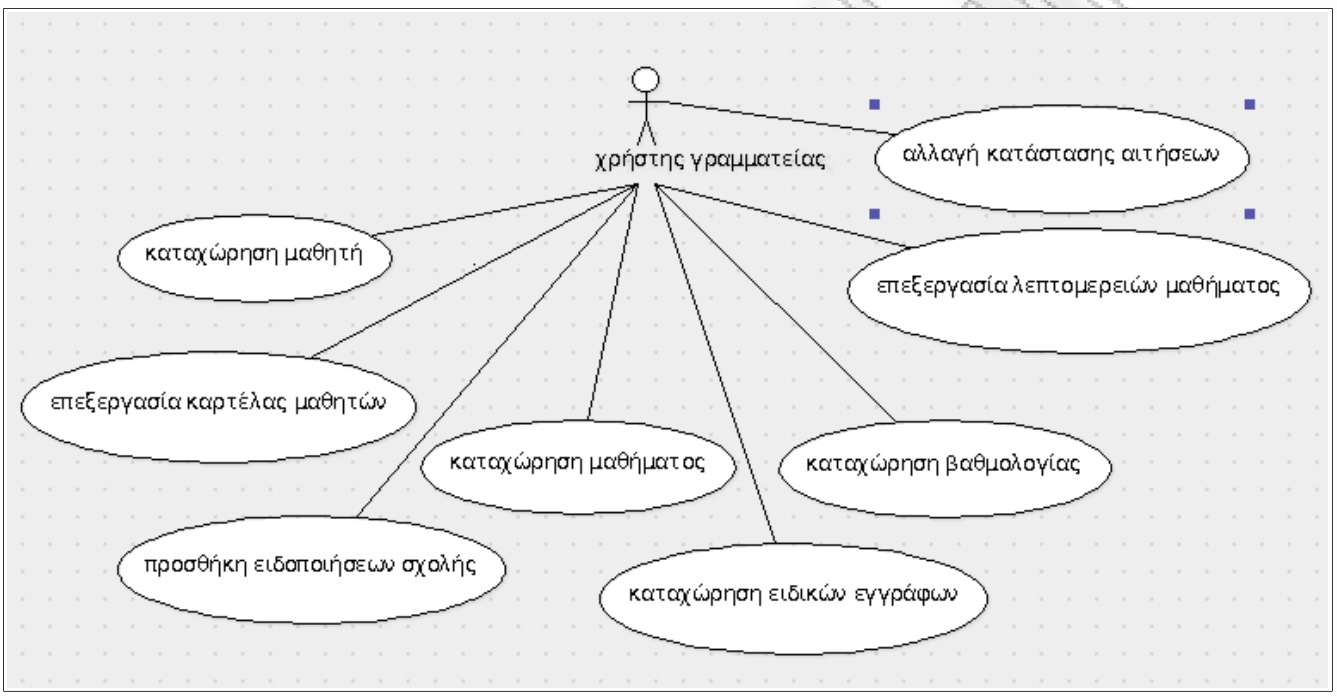
4.6 Περιπτώσεις χρήσεις του συστήματος Dreskt

Περίπτωση χρήσης ονομάζουμε τη σχέση που υπάρχει ανάμεσα σε ένα ρόλο-χρήστη και στη λειτουργικότητα ενός συστήματος. Τα εξειδικευμένα διαγράμματα που χρησιμοποιούνται για να οπτικοποιήσουμε τη συνολική λειτουργία ενός συστήματος είναι μέρος της **Unified Modelling Language** ή **UML** και ονομάζονται **use case diagrams**.

Οι περιπτώσεις χρήσεις του συστήματος Dreskt χωρίζονται σε δύο κατηγορίες. Η πρώτη αφορά τους χρήστες-προσωπικό της γραμματείας και η δεύτερη τους μαθητές. Και στις δύο περιπτώσεις η πρόσβαση στη λειτουργικότητα του συστήματος γίνεται μόνο μετά από εξουσιοδότηση. Καμία λειτουργία δεν είναι προσβάσιμη σε μη εγγραγραμένους χρήστες, καθώς αυτό θα συντελούσε σε ουσιώδες κένο στην ασφάλεια του συστήματος και σε έκθεση προσωπικών δεδομένων των μαθητών.

4.6.2 Περιπτώσεις χρήσεις για τους χρήστες της γραμματείας

Στα προηγούμενα κεφάλαια κάναμε διάφορες αναφορές στο ρόλο των χρηστών της γραμματείας. Οι λειτουργίες που μπορούν να εκτελέσουν συνοψίζονται στο παρακάτω διάγραμμα περιπτώσεων χρήσης :



Εικόνα 30: Διάγραμμα περιπτώσεων χρήσης για τους χρήστες της γραμματείας

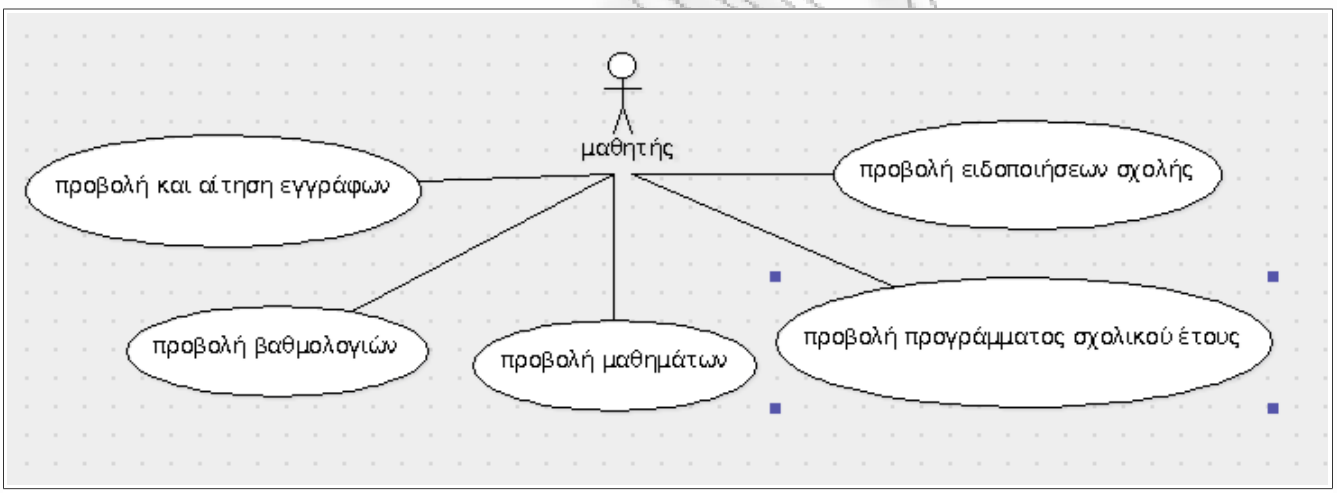
Όπως λοπόν βλέπουμε στο διάγραμμα 30 ένας χρήστης της γραμματείας μπορεί :

1. να καταχωρήσει-εγγράψει νέο μαθητή
2. να επεξεργαστεί την καρτέλα ενός υπάρχοντα μαθητή
3. να καταχωρήσει νέο μάθημα
4. να τροποποιήσει λεπτομέρειες του μαθήματος
5. να εγγράψει μαθητές στο μάθημα αυτό
6. να καταχωρήσει βαθμολογία για τον μαθητή σε ένα συγκεκριμένο μάθημα

7. να καταχωρήσει τύπους ειδικών εγγράφων (για παράδειγμα η αναλυτική βαθμολογία είναι ένας ειδικός τύπος εγγράφου)
8. και τέλος να αλλάξει την κατάσταση σε μια αίτηση για ειδικό έγγραφο (αν για παράδειγμα εγκριθεί κάποια αίτηση του μαθητή)

4.6.3 Περιπτώσεις χρήσεις για τους μαθητές

Ο ρόλος των μαθητών έχει σαφώς λιγότερες λειτουργίες. Συνοψίζοντας λοιπόν έχουμε το παρακάτω διάγραμμα περιπτώσεων χρήσεις για τους μαθητές :



Εικόνα 31: Διάγραμμα περιπτώσεων χρήσεις του συστήματος Dreskt από τους μαθητές

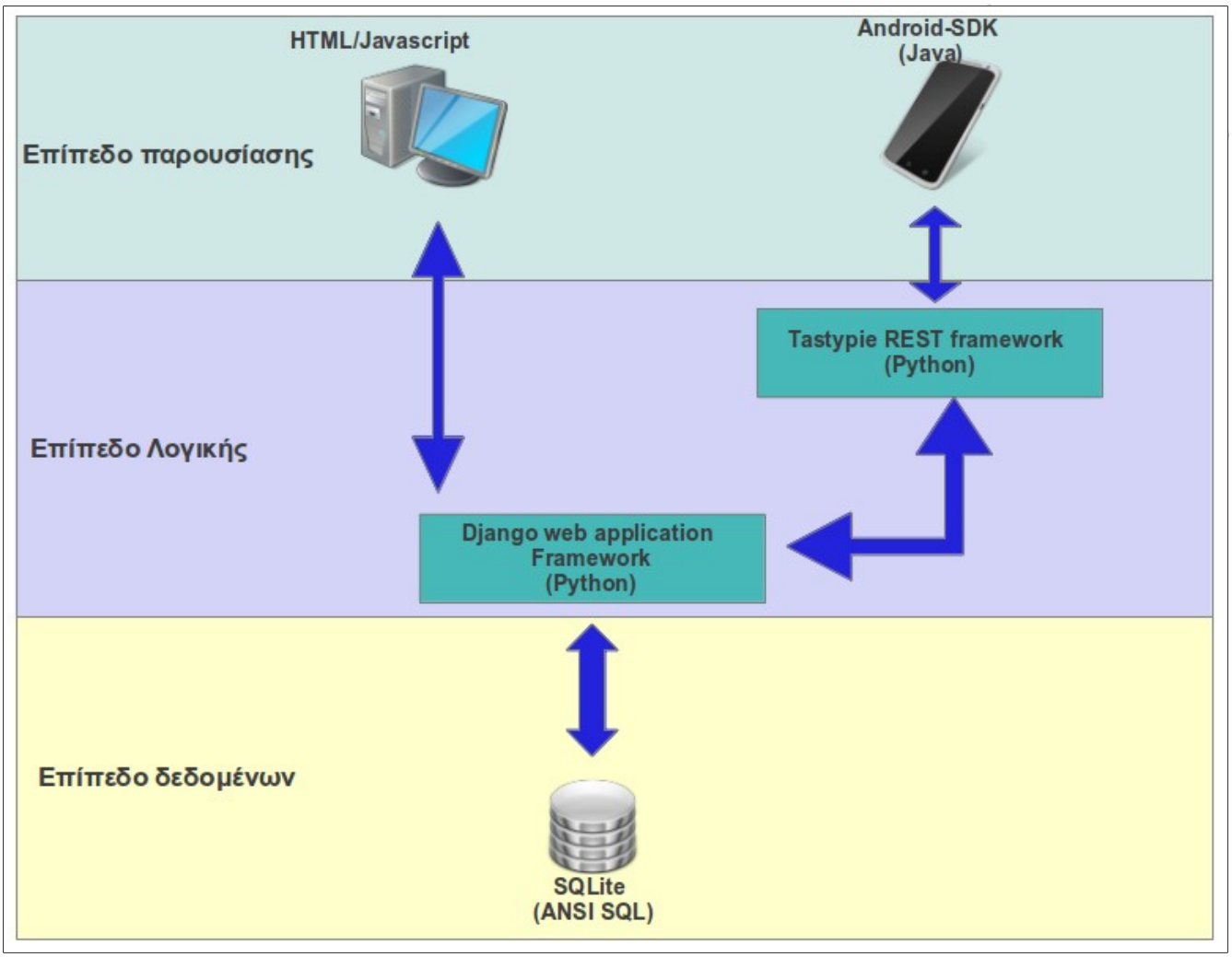
Έτσι, ένας μαθητής μπορεί :

1. Να προβάλλει τις βαθμολογίες στα έως τώρα μαθήματα που έχει παρακολουθήσει
2. να δει ειδοποιήσεις από τη γραμματεία της σχολής
3. να προβάλλει το ορολόγιο πρόγραμματα μαθημάτων
4. να προβάλλει τα μαθήματα που παρακολουθεί

5. να καταχωρήσει νέες αιτήσεις
6. και να δει την κατάσταση ολοκλήρωσης των αιτήσεων που έχει ζητήσει

4.7 Υλοποίηση του συστήματος Dreskt

Για την υλοποίηση του συστήματος Dreskt χρησιμοποιήθηκαν δύο γλώσσες προγραμματισμού, δύο περιβάλλοντα ανάπτυξης καθώς επίσης και διάφορες βοηθητικές βιβλιοθήκες. Συνοψίζοντας το σύνολο των τεχνολογιών στο παρακάτω διάγραμμα μπορούμε να δούμε τι χρησιμοποιήθηκε και σε ποιο επίπεδο της αρχιτεκτονικής :



Εικόνα 32: Επισκόπηση των τεχνολογιών που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος Dreskt

Κεφάλαιο 5

Παρουσίαση του Dreskt

5.1 Εισαγωγή

Στο κεφάλαιο 5 παρουσιάζεται το σύστημα **Dreskt** το οποίο σχεδιάστηκε με σκοπό να δείξει πως, με τη βοήθεια των web services, μπορούν να εκτεθούν συγκεκριμένες λειτουργίες μιας γραμματείας ενός εκπαιδευτικού ιδρύματος στα μέλη της κοινότητας της, και συγκεκριμένα στους μαθητές. Όπως είδαμε το κυρίως σύστημα, αυτό της γραμματείας, υλοποιήθηκε εξ'ολοκλήρου σαν μια διαδικτυακή εφαρμογή επιτρέποντας εύκολη πρόσβαση στους χρήστες καθώς όλοι, πλέον, οι υπολογιστές φέρουν φυλλομετρητή διαδικτύου. Από την άλλη πλευρά το πρόγραμμα που χρησιμοποιούν οι μαθητές, **Dreskt Student**, είναι υλοποιημένο στο πλαίσιο μιας φορητής συσκευής τελευταίας τεχνολογίας και ευρείας χρήσης καθώς η μεγάλη πλειοψηφία των νέων ανθρώπων είναι κάτοχοι μιας τέτοιας συσκευής και άριστα εξοικειωμένοι με τη χρήση της.

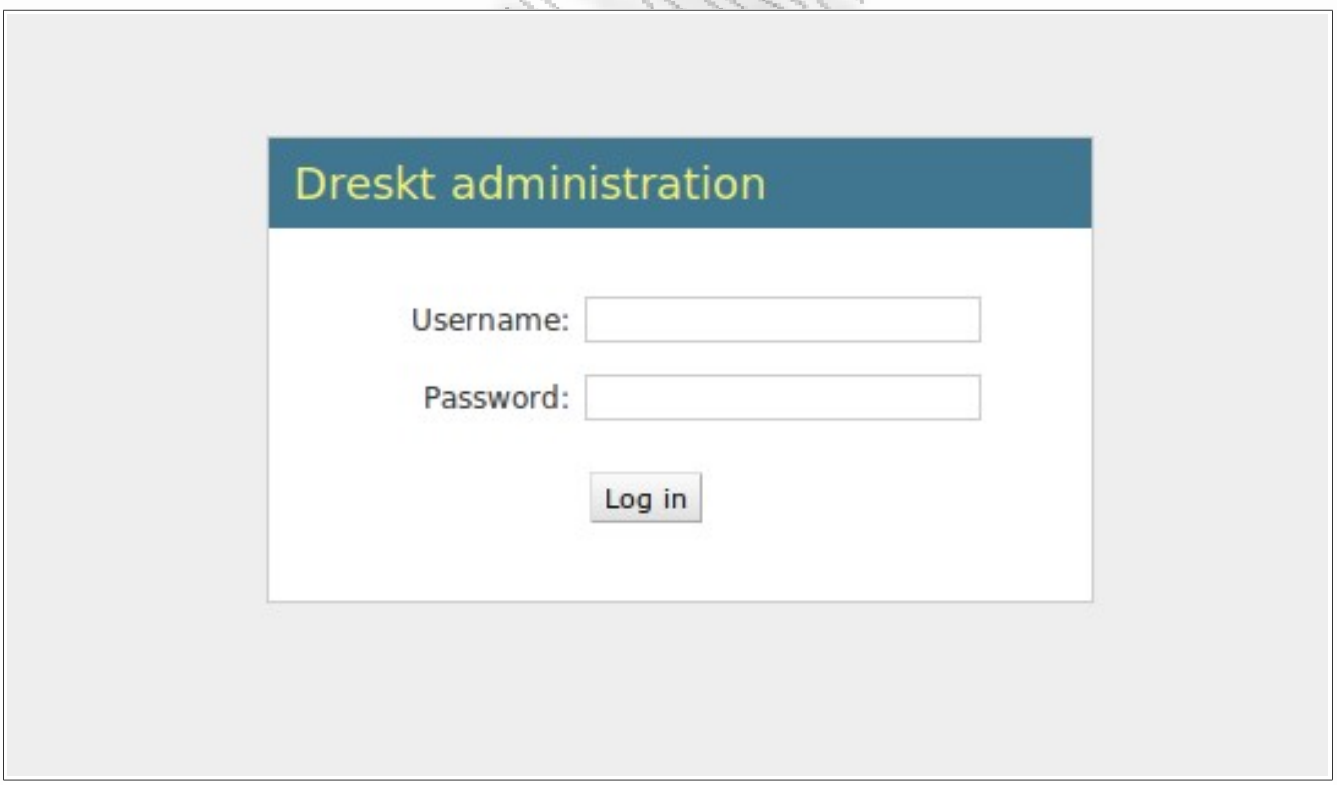
5.2 Σενάρια χρήσης Dreskt

Τα σενάρια χρήσης του Dreskt σχετίζονται μόνο με τους εξουσιοδοτημένους χρήστες του συστήματος. Οποιοσδήποτε άλλος χρήστης αποκλείεται από το σύστημα. Κάθε μαθητής υποχρεούται να εξουσιοδοτηθεί από το σύστημα του

εκπαιδευτικού ιδρύματος ώστε να έχει πρόσβαση στο δικό του περιβάλλον πληροφοριών. Αντίθετα η γραμματεία χειρίζεται ένα κοινό περιβάλλον πληροφοριών καθώς οι χρήστες της είναι εξουσιοδοτημένοι να εκτελούν κοινές ενέργειες.

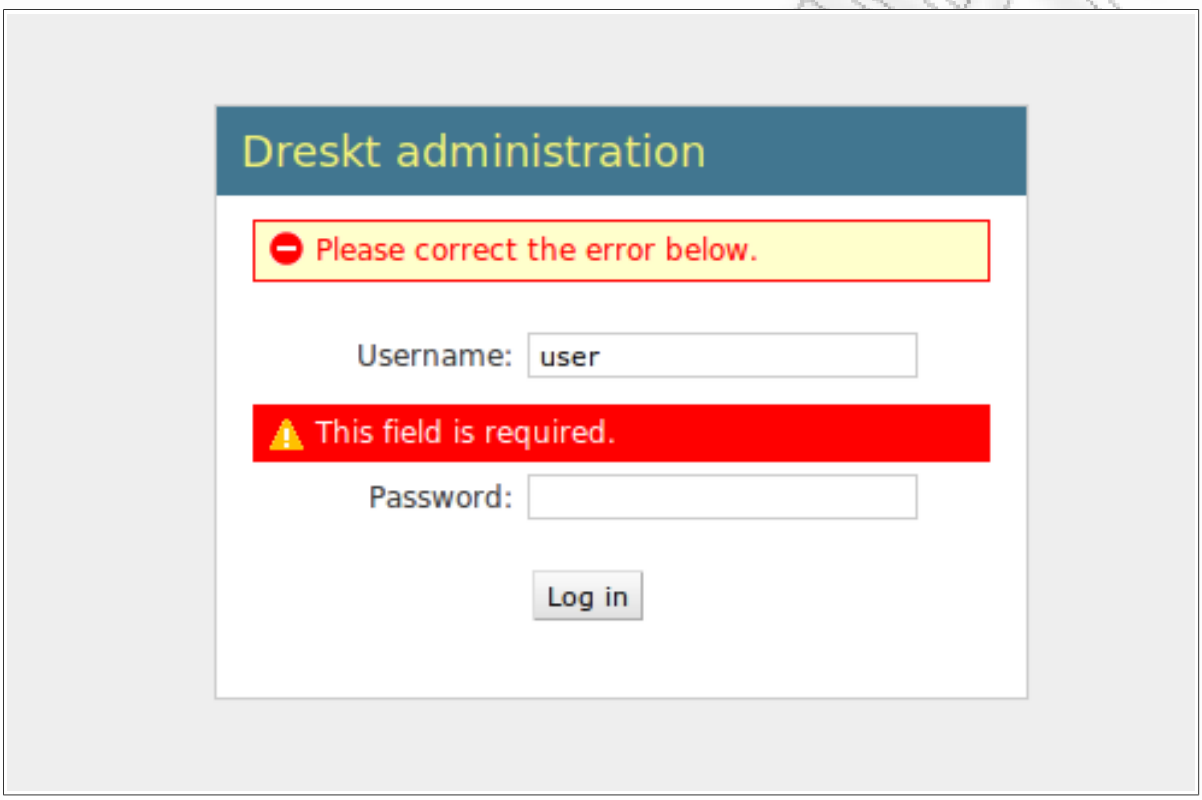
5.2.1 Είσοδος στο σύστημα της γραμματείας

Η είσοδος στο σύστημα γίνεται με τη χρήση ονόματος χρήστη και κωδικού. Τα στοιχεία εισόδου διατίθονται στο σύστημα μόνο από τους διαχειριστές του συστήματος καθώς δεν επιτρέπονται νέες εγγραφές χρηστών. Για να έχει πρόσβαση ο χρήστης της γραμματείας χρησιμοποιεί την οθόνη εισόδου. Η οθόνη εισόδου είναι και η πρώτη οθόνη που βλέπουν οι χρήστες της γραμματείας.



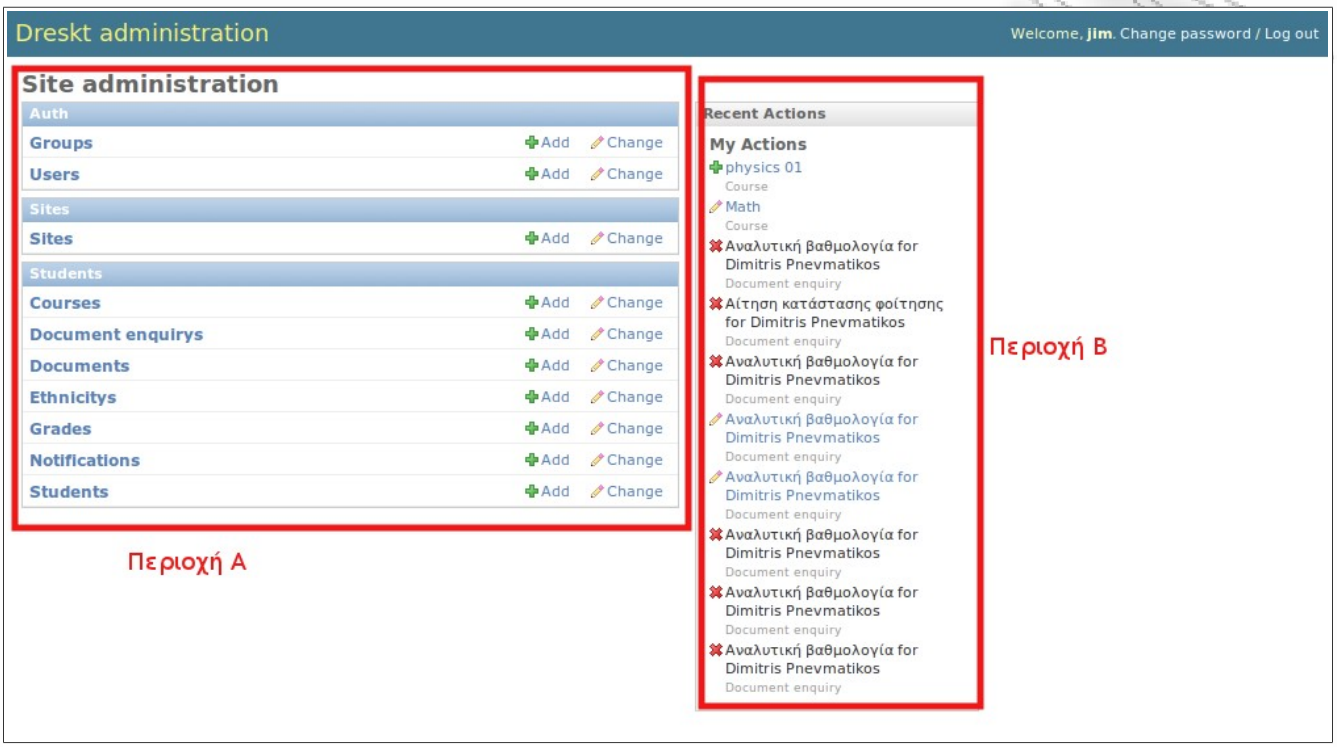
Εικόνα 33: Η οθόνη εισόδου στο Dreskt

Στο πεδίο username πληκτρολογούμε το όνομα χρήστη και στο πεδίο password το συνθηματικό. Σε περίπτωση λανθασμένης καταχώρησης, το σύστημα μας ειδοποιεί για το λάθος που έχουμε κάνει ή έαν δεν καταχωρήσουμε καθόλου στοιχεία μας ενημερώνει ποια στοιχεία είναι υποχρεωτικά.



Εικόνα 34: Λανθασμένα ή ελλειπή στοιχεία χρήστη

Με το πέρας της καταχώρησης των στοιχείων χρήστη και αφού πιστοποιηθούμε από το σύστημα, τότε εμφανίζεται η αρχική σελίδα.



Εικόνα 35: Η αρχική σελίδα του Dreskt

Η αρχική σελίδα του Dreskt χωρίζεται σε δύο βασικές περιοχές όπως φαίνεται στο σχήμα 35.

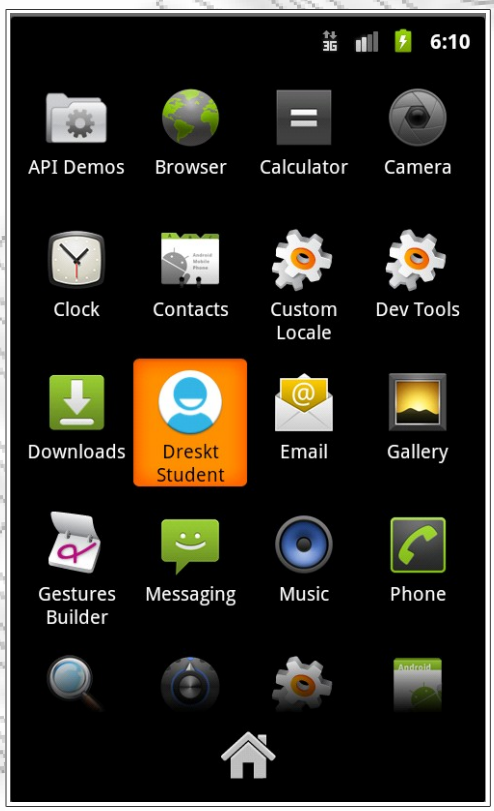
Η **περιοχή Α** περιλαμβάνει λίστα με τις διαθέσιμες οντότητες του ιδρύματος και περιγράφει τις λειτουργίες διαθέσιμες για κάθε μια από αυτές. Εκτός από τις οντότητες περιλαμβάνονται και λειτουργίες που αφορούν το ίδιο το σύστημα όπως για παράδειγμα η διαχείριση χρηστών του συστήματος. Όπως θα δούμε παρακάτω για κάθε ένα μαθητή υπάρχει μια οντότητα χρήστη, η οποία φέρει το όνομα και κωδικό χρήστη.

Η **περιοχή Β** προβάλλει λίστα μια σύνοψη των τελευταίων ενεργειών που έχουν εκτελεσθεί από εμάς, λειτουργεί δηλαδή ως ιστορικό. Κάθε καινούρια ενέργεια που εκτελούμε εμφανίζεται πάντα στο πάνω μέρος της λίστας και οι παλαιότερες αφαιρούνται τελείως.

Η αρχική οθόνη αποτελεί το σημείο εκκίνησης για οποιοδήποτε σενάριο χρήσης περιλαμβάνει το Dreskt.

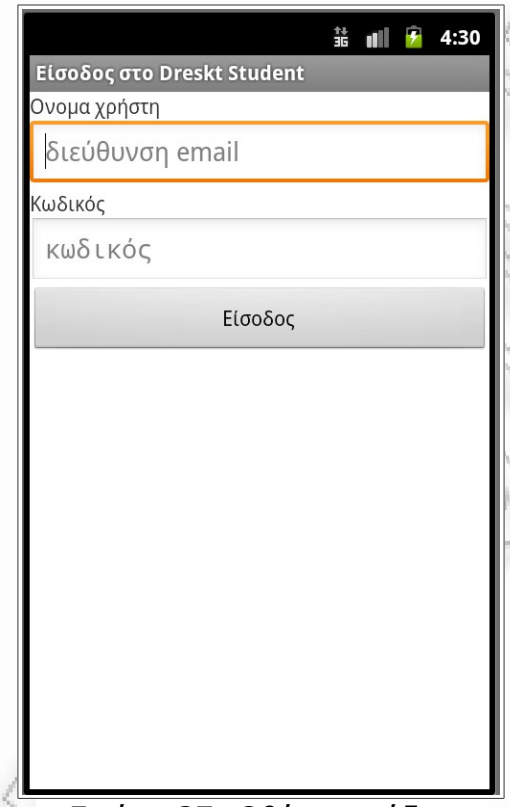
5.2.2 Είσοδος στο Dreskt Student

Η είσοδος στο Dreskt Student έχει την ίδια λογική με αυτόν της εισόδου των χρηστών της γραμματείας. Προαπαιτούμενα είναι το όνομα χρήστη και το συνθηματικό που διαθέτει στους μαθητές η γραμματεία. Έτσι, για να αποκτήσει πρόσβαση στο σύστημα, ο μαθητής τρέχει σε πρώτο στάδιο την εφαρμογή Dreskt Student, ακουμπώντας το αντίστοιχο εικονίδιο στην κινητή συσκευή.



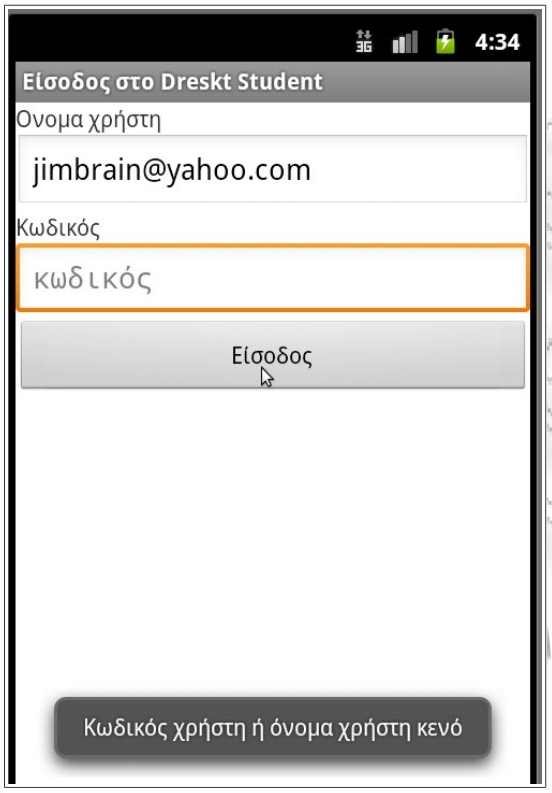
Εικόνα 36: Άνοιγμα της εφαρμογής Dreskt Student

Μόλις ανοίξει το παράθυρο της εφαρμογής ο χρήστης καλείται να εισάγει στα κατάλληλα πεδία, τα στοιχεία εισόδου.



Εικόνα 37: Οθόνη εισόδου στο Dreskt Student

Σε περίπτωση που ο μαθητής εισάγει ελλειπή στοιχεία το σύστημα τον ειδοποιεί αναλόγως, όπως και στο σύστημα της γραμματείας.

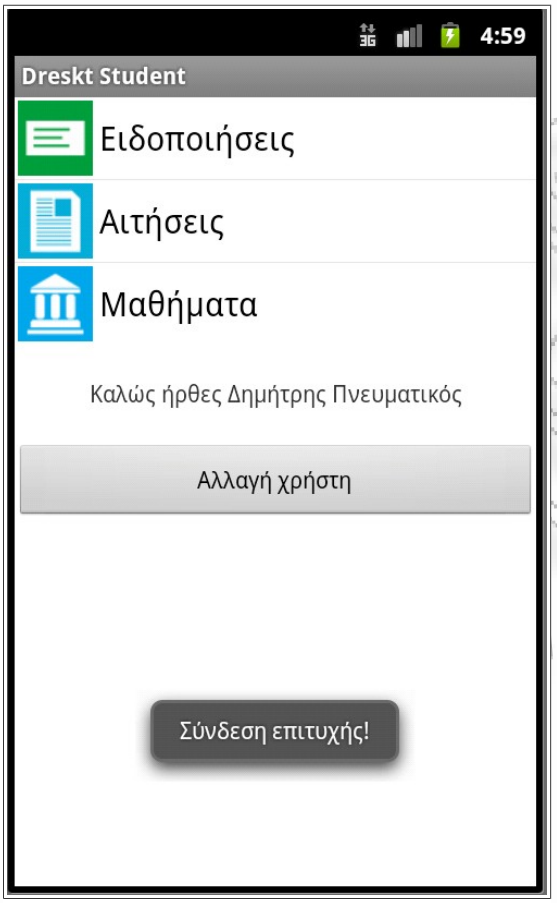


Εικόνα 38: Ελλειπής καταχώρηση στοιχείων χρήστη

Αφού τα στοιχεία εισόδου πιστοποιηθούν από το σύστημα[13], τότε ο μαθητής ενημερώνεται για την επιτυχή είσοδου του, και οδηγείται στη αρχική οθόνη της εφαρμογής (βλ. Εικόνα 39).

Η αρχική οθόνη προσφέρει πρόσβαση σε όλες τις υπηρεσίες που παρέχονται από τη γραμματεία του εκπαιδευτικού ιδρύματος.

13 Αξίζει να αναφερθεί ότι η εφαρμογή Dreskt Student ζητά τα στοιχεία του μαθητή μόνο μια φορά. Τα στοιχεία κρατούνται σε εσωτερικές δομές αποθήκευσης που προσφέρει το Android και δεν μπορούν να αποκαλυφθούν σε κανέναν. Έτσι καθίστανται πλήρως ασφαλή και διευκολύνουν στη χρήση της εφαρμογής καθώς δεν χρειάζεται να εισάγονται κάθε φορά που εκκινείται.



Εικόνα 39: Αρχική οθόνη της εφαρμογής Dreskt Student

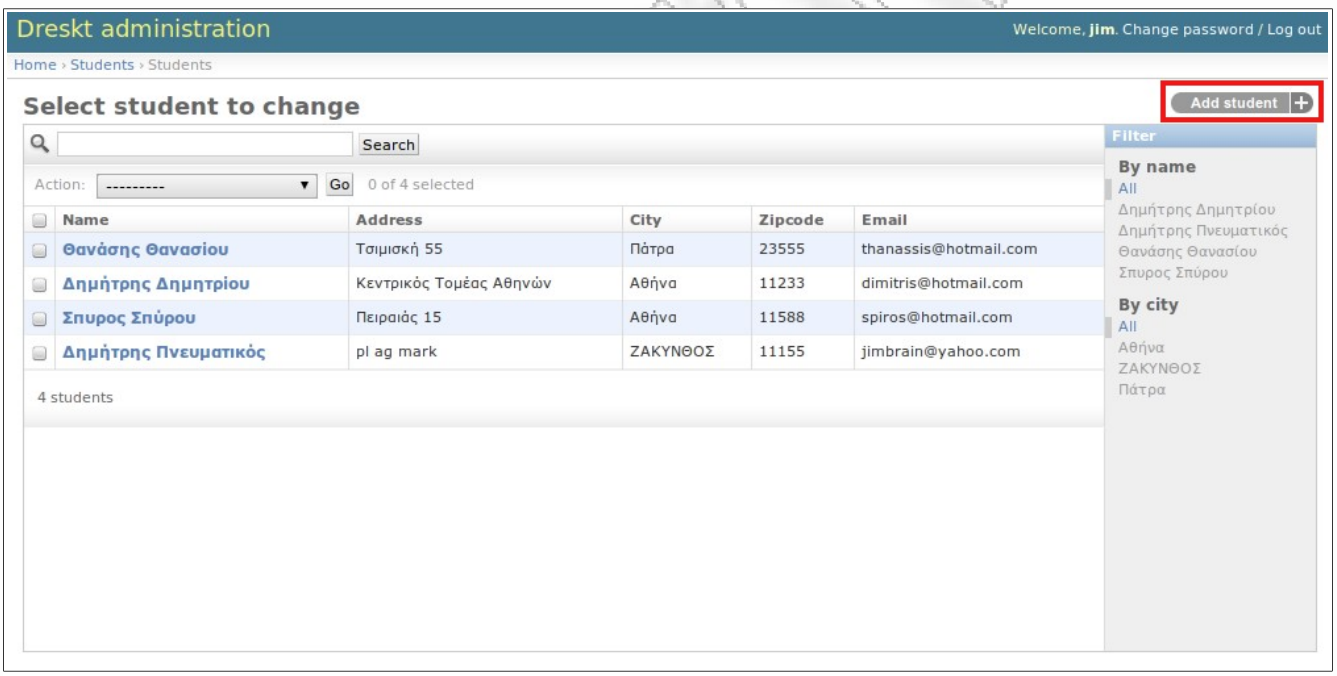
5.2.3 Μαθητές

5.2.3.1 Εγγραφή νέου μαθητή

Η απαραίτητη πιστοποίηση εισόδου δίνεται μόνο σε εξουσιοδοτημένους μαθητές. Η γραμματεία αναλαμβάνει να κάνει την εγγραφή τους και εκτός από την

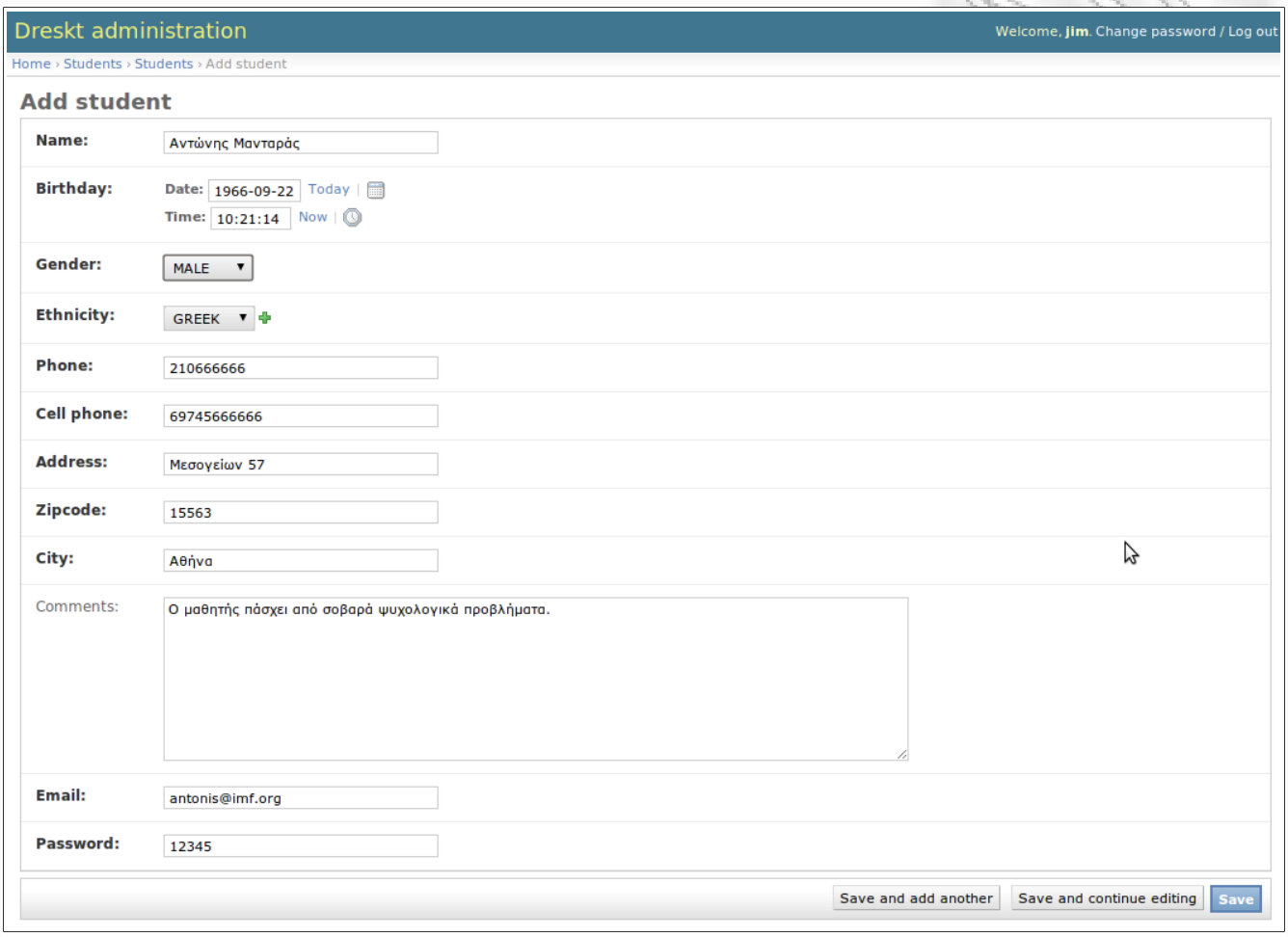
καταχώρηση των προσωπικών τους στοιχείων, παρέχει και ένα αυτόματο σύστημα δημιουργίας χρηστών. Αυτό σημαίνει οτι για κάθε εγγραφή νέου μαθητή στο εκπαιδευτικό ίδρυμα δημιουργείται και ένας νέος χρήστης ο οποίος φέρει τα δικαιώματα της ομάδας των μαθητών και μπορεί να τα χρησιμοποιήσει για να έχει πρόσβαση στα web services της γραμματείας.

Για την εγγραφή νέου μαθητή στο σύστημα ακολουθούνται τα εξής βήματα : από την αρχική οθόνη της γραμματείας, ο χρήστης επιλέγει "Students" και οδηγείται στην οθόνη διαχείρισης των μαθητών. Απο εκεί επιλέγει το "Add Student" στο πάνω δεξιά μέρος της οθόνης, όπως φαίνεται στην εικόνα 40.



Εικόνα 40: Εγγραφή νέου μαθητή

Η επιλογή αυτή τον οδηγεί στη φόρμα εγγραφής νέου μαθητή όπου πρέπει να καταχωρήσει προσωπικά στοιχεία του μαθητή όπως, έτος γέννησης, όνομα, διεύθυνση ηλεκτρονικού ταχυδρομείου κ.α.

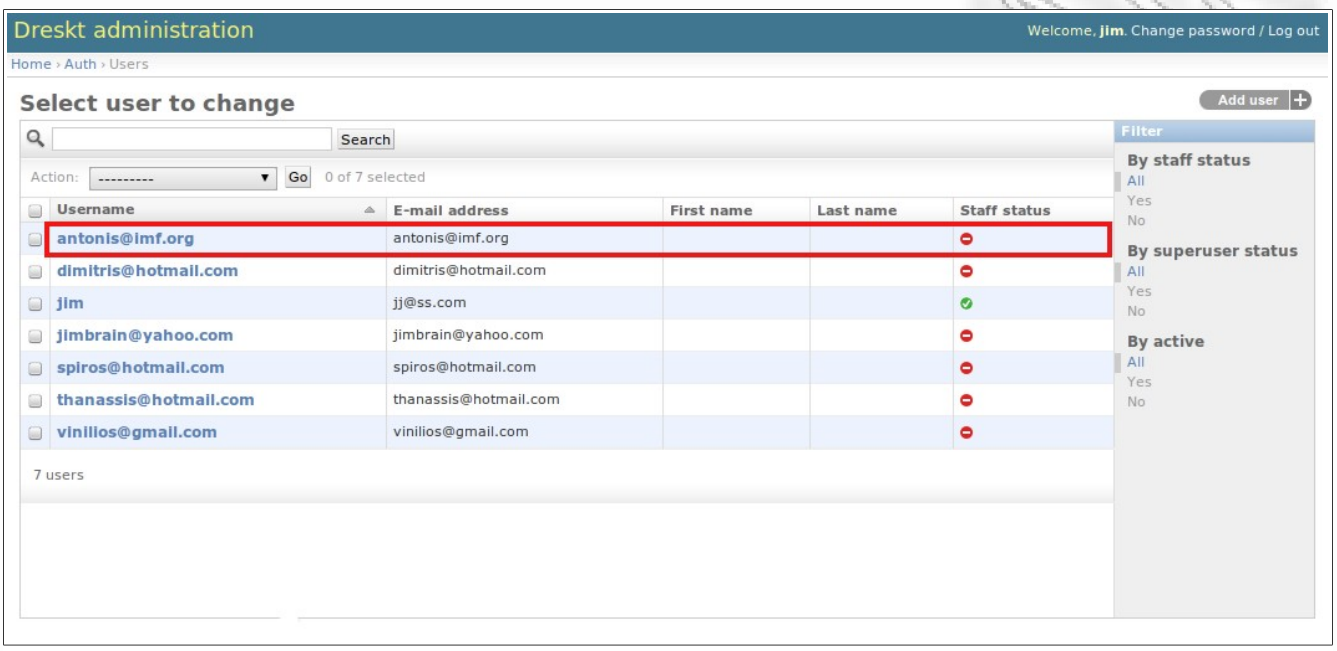


Εικόνα 41: Φόρμα εγγραφής νέου μαθητή

Τα βασικά στοιχεία που είναι απαιτούμενο να καταχωρηθούν για να είναι ικανό το σύστημα να δημιουργήσει νέο χρήστη είναι η διεύθυνση email και ο προσωπικός κωδικός-συνθηματικό. Η διεύθυνση email χρησιμοποιείται ως όνομα χρήστη.

Μόλις αποθηκεύσουμε την εγγραφή του νέου μαθητή, η διαδικασία δημιουργίας του νέου χρήστη έχει ολοκληρωθεί με επιτυχία. Για να το διαπιστώσουμε πλοηγούμαστε και πάλι στην αρχική σελίδα του Dreskt. Εκεί επιλέγουμε την οντότητα "Users" και στη λίστα που θα εμφανιστεί η οποία περιλαμβάνει τους χρήστες του συστήματος περιλαμβάνεται τώρα και ο νέος μαθητής, όπως φαίνεται

στην εικόνα 42.

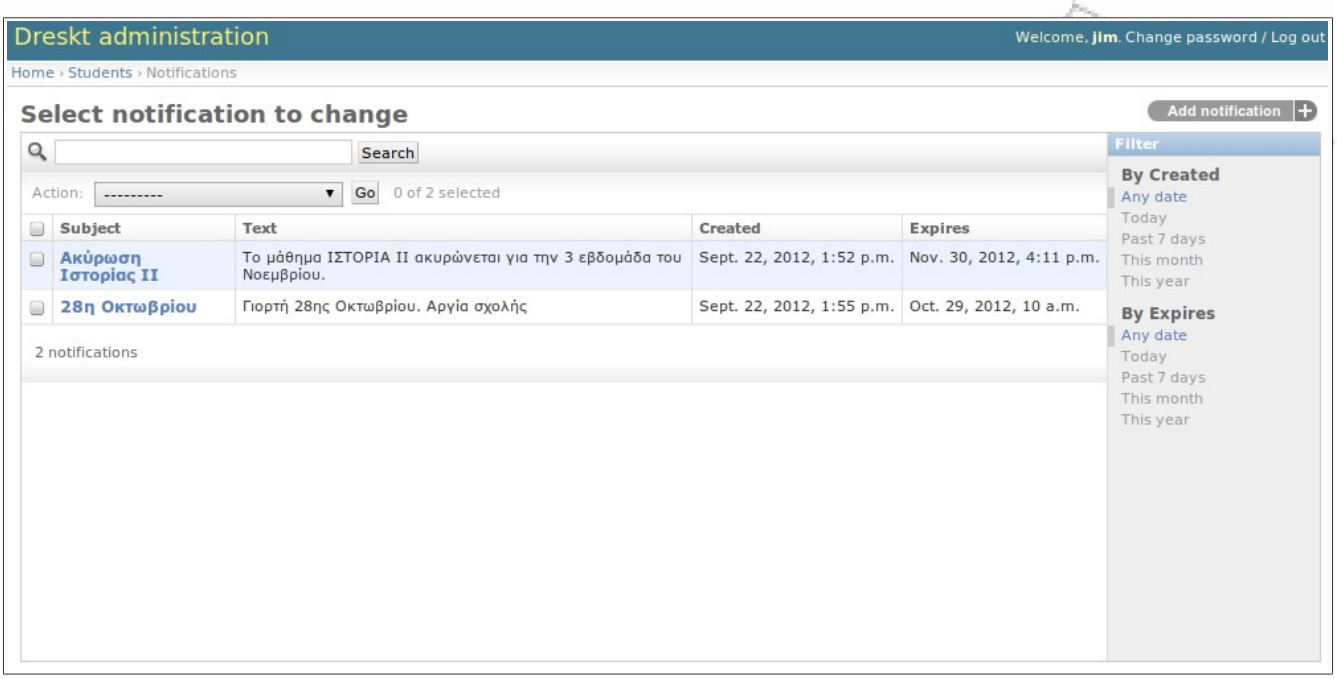


Εικόνα 42: Αυτόματη δημιουργία νέου χρήστη για κάθε εγγραφή μαθητή

5.2.4 Ειδοποιήσεις

5.2.4.1 Προβολή ειδοποιήσεων

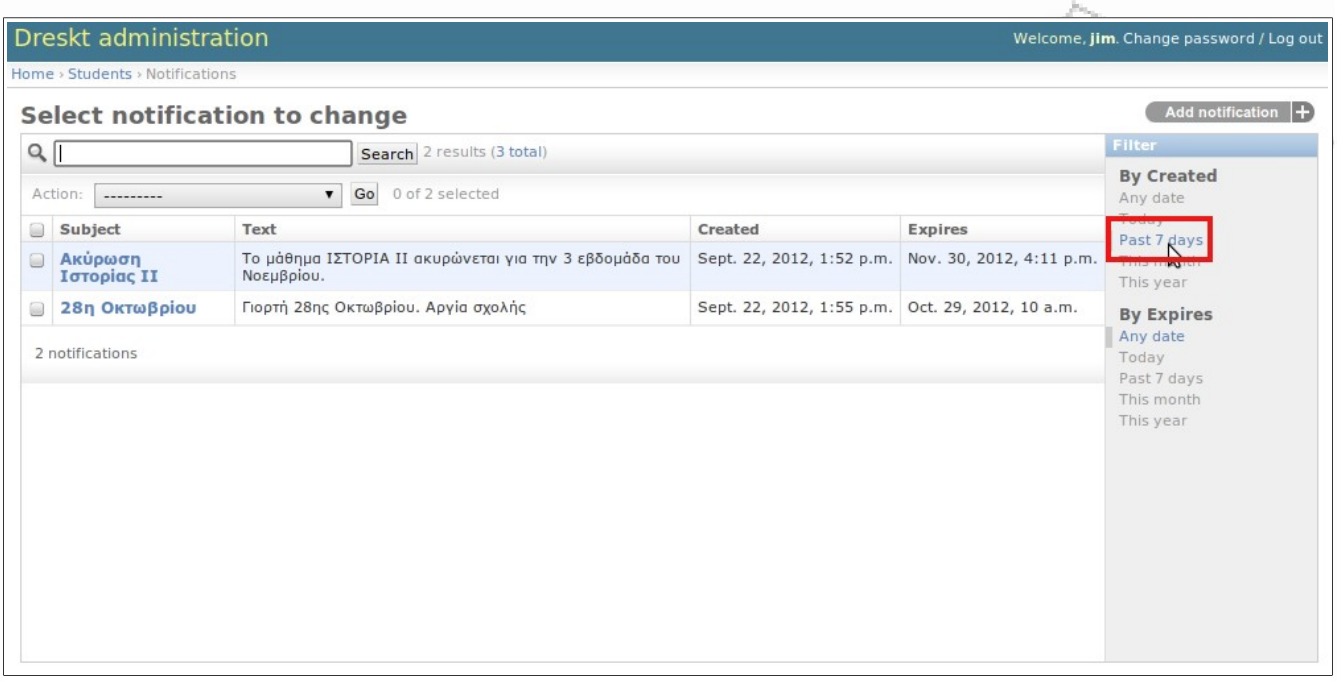
Όλες οι οντότητες του Dreskt παρουσιάζουν κοινή συμπεριφορά στη λειτουργικότητα. Για να προβάλλουμε το σύνολο των καταχωρήσεων κάνουμε κλικ πάνω σε μια οντότητα. Έτσι για να δούμε το σύνολο των ειδοποιήσεων που είναι αυτή τη στιγμή διαθέσιμες στη σχολή κάνουμε κλικ πάνω στο Notifications.



Εικόνα 43: Η λίστα με τις ειδοποιήσεις της σχολής

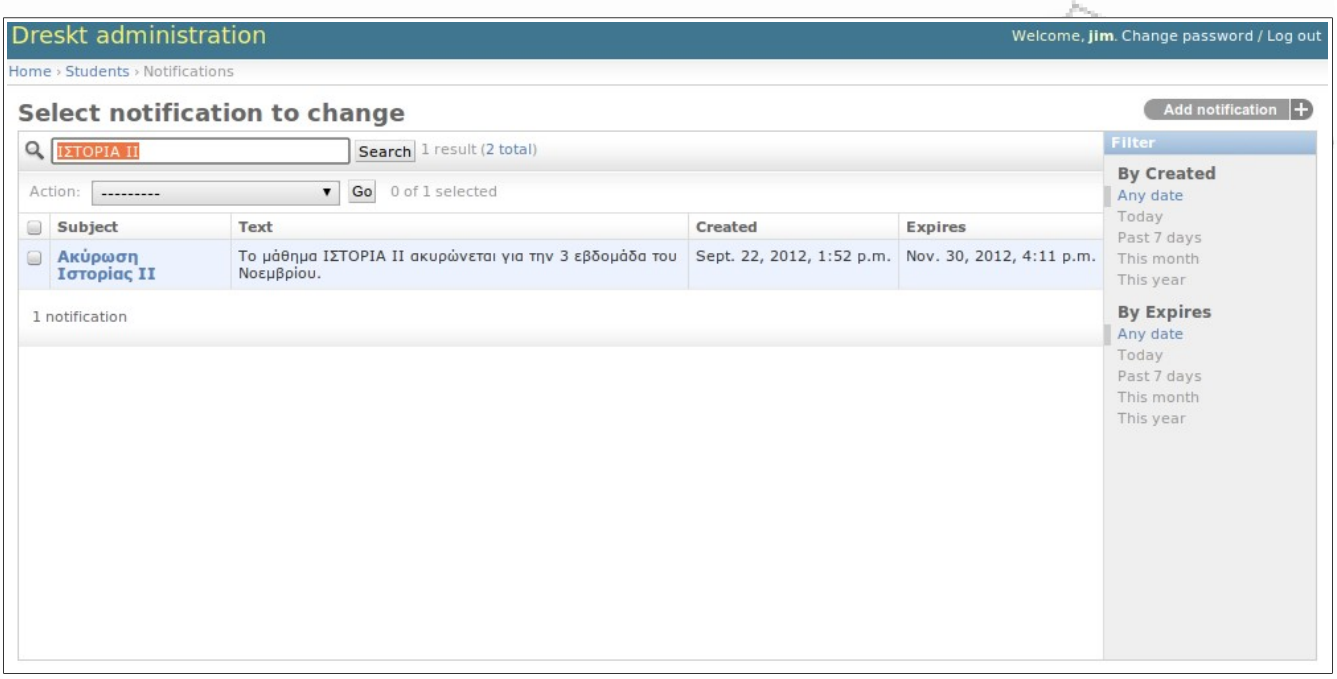
Παρόμοια με λογική της αρχική σελίδα του συστήματος, η σελίδα των ειδοποιήσεων προβάλλει τις διαθέσιμες ειδοποιήσεις και παρέχει πρόσβαση σε όλες τις λειτουργίες που μπορούμε να εκτελέσουμε.

Εκτός από τη βασική λειτουργικότητα που περιγράφηκε στο κεφάλαιο της αρχιτεκτονικής του συστήματος, προσφέρονται και κάποιες επιπλέον δυνατότητες. Μπορούμε, για παράδειγμα να προσαρμόσουμε την προβολή της λίστας με τη χρήση φίλτρων. Τα φίλτρα βρίσκονται στη δεξιά πλευρά της οθόνης. Οι ειδοποιήσεις μπορούν να φιλτραριστούν με βάση την ημερομηνία δημιουργίας τους και τη λήξη της ισχύς τους.



Εικόνα 44: Προβολή ειδοποιήσεων που δημιουργήθηκαν τις τελευταίες 7 μέρες

Επιπρόσθετα, χρησιμοποιώντας το πεδίο κειμένου που βρίσκεται ακριβώς πάνω από τη λίστα μπορούμε να περιορίσουμε τα αποτελέσματα, πληκτρολογώντας κάποιον όρο αναζήτησης. Πατώντας το πλήκτρο ENTER το σύστημα θα εκτελέσει αναζήτηση στο κείμενο(Notification text) και στο θέμα (Notification subject) των ειδοποιήσεων και η λίστα θα αναπροσαρμοστεί (βλ. Εικόνα 45).

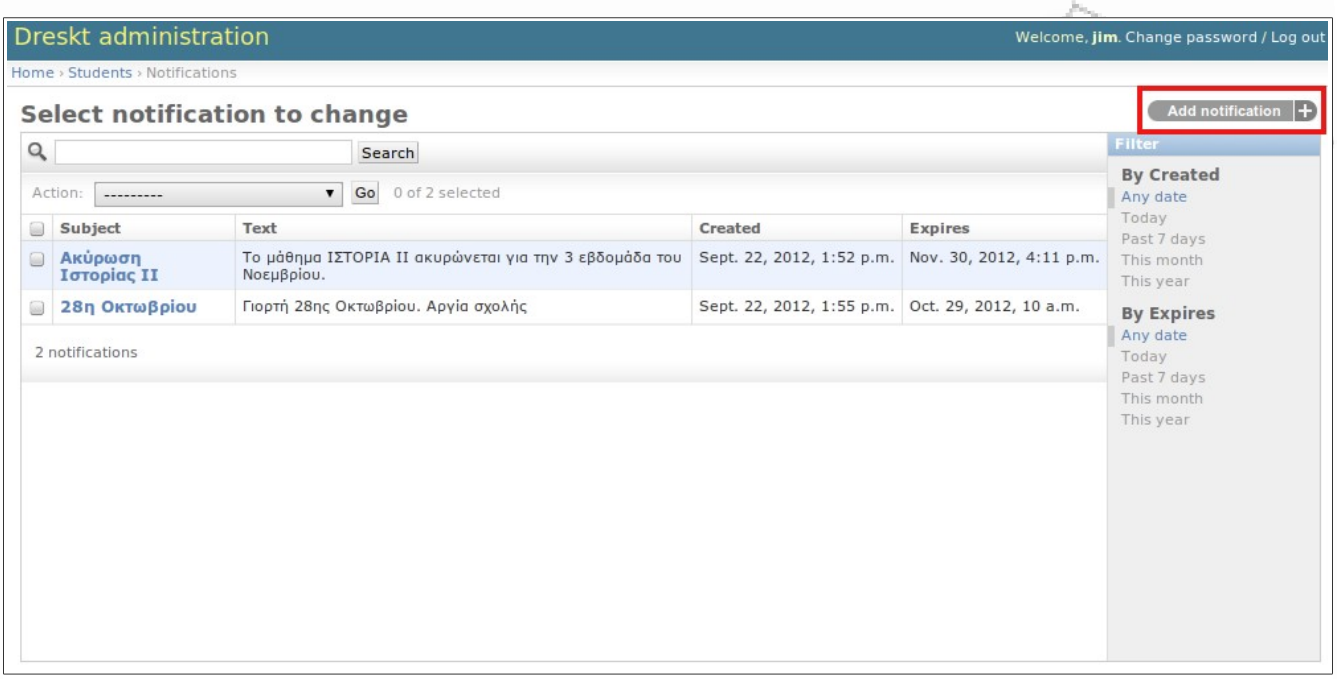


Εικόνα 45: Αναζήτηση με βάση κείμενο ειδοποίησης

Τέλος υπάρχει η δυνατότητα της ταξινόμησης των αποτελεσμάτων με αλφαβητική σειρά. Αυτό επιτυγχάνεται με κλικ στην αντίστοιχη επικεφαλίδα στήλης.

5.2.4.2 Καταχώρηση νέας ειδοποίησης

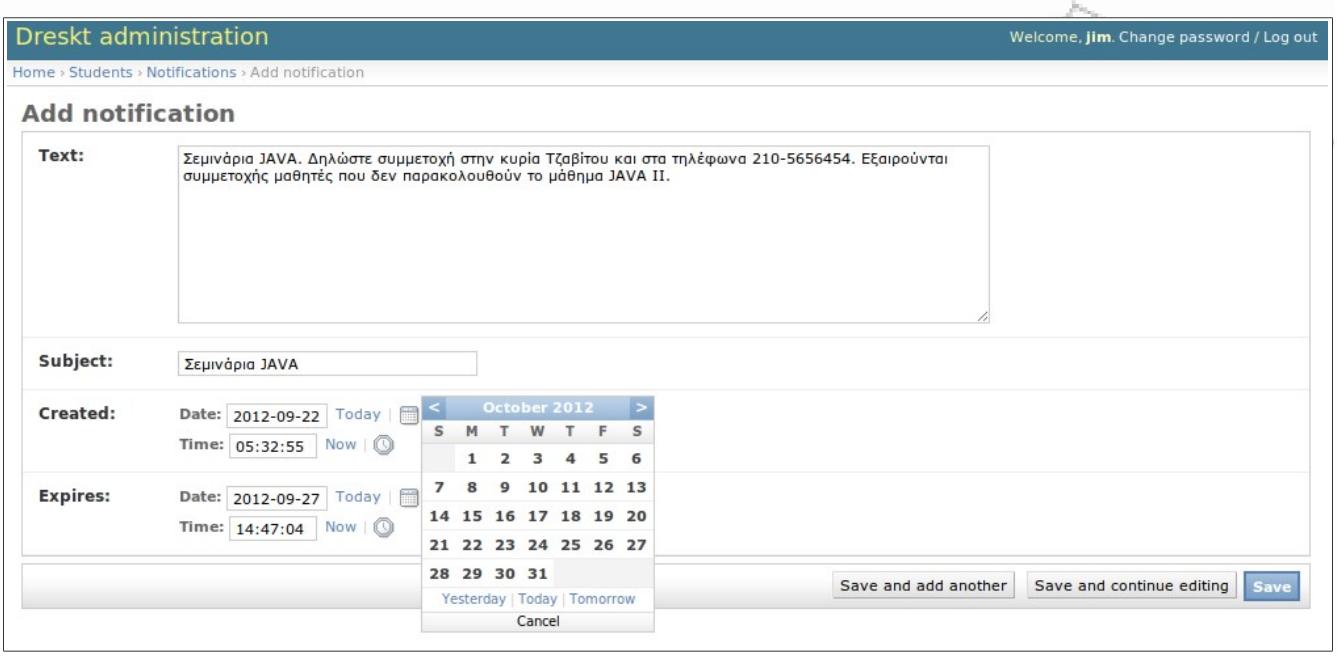
Η καταχώρηση της ειδοποίησης γίνεται με τη χρήση του κουμπιού "Add Notification" που βρίσκεται στο πάνω δεξιά μέρος της οθόνης.



Εικόνα 46: Εκκίνηση διαδικασίας δημιουργίας νέας ειδοποίησης με το κουμπί Add Notification

Αμέσως, οδηγούμαστε στη φόρμα καταχώρησης της ειδοποίησης όπου βάζουμε τις λεπτομέρειες της ειδοποίησης, δηλαδη :

1. Το κείμενο της ειδοποίησης
2. το θέμα
3. την ημερομηνία από την οποία ισχύει η ειδοποίηση, με προκαθορισμένη τιμή τη σημερινή ημερομηνία και τέλος
4. την ημερομηνία μέχρι την οποία θα διαρκέσει η συγκεκριμένη ειδοποίηση



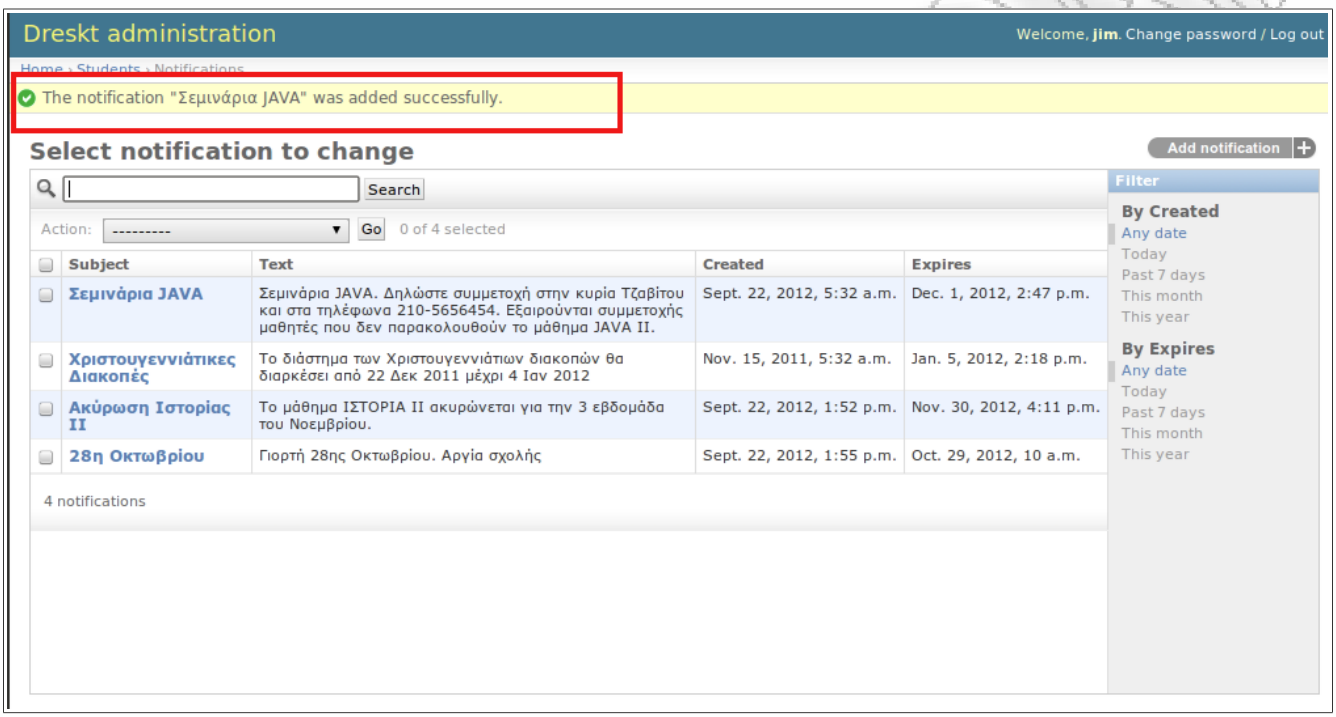
Εικόνα 47: Καταχώρηση νέας ειδοποίησης

Μόλις ολοκληρώσουμε την καταγραφή των στοιχείων, έχουμε τρείς επιλογές, εκφρασμένες με τα τρία κουμπιά που βρίσκονται στο κάτω δεξιά μέρος της οθόνης :

- Αποθήκευση και πρόσθεση άλλης (κουμπί "Save and add another")
- Αποθήκευση και συνέχιση τροποποίησης στοιχείων (κουμπί "Save and continue editing")
- αποθήκευση (κουμπί "Save")

Πατώντας το πρώτο η ειδοποίηση θα αποθηκευτεί και μετά θα οδηγηθούμε πάλι στη φόρμα νέας ειδοποιήσης με καθαρισμένα τα πεδία. Η δεύτερη θα αποθηκεύσει την ειδοποιήση και θα παραμείνει στην παρούσα φόρμα ώστε να συνεχίσουμε να κάνουμε τροποποιήσεις ενώ πατώντας το κουμπί "save" η ειδοποίηση αποθηκεύεται και το σύστημα μας επιστρέφει στην αρχική οθόνη των

ειδοποιήσεων. Σε κάθε περίπτωση το σύστημα μας ειδοποιεί για την αποθήκευση της καταχώρησης εμφανίζοντας αντίστοιχο μήνυμα στο πάνω μέρος της οθόνης.



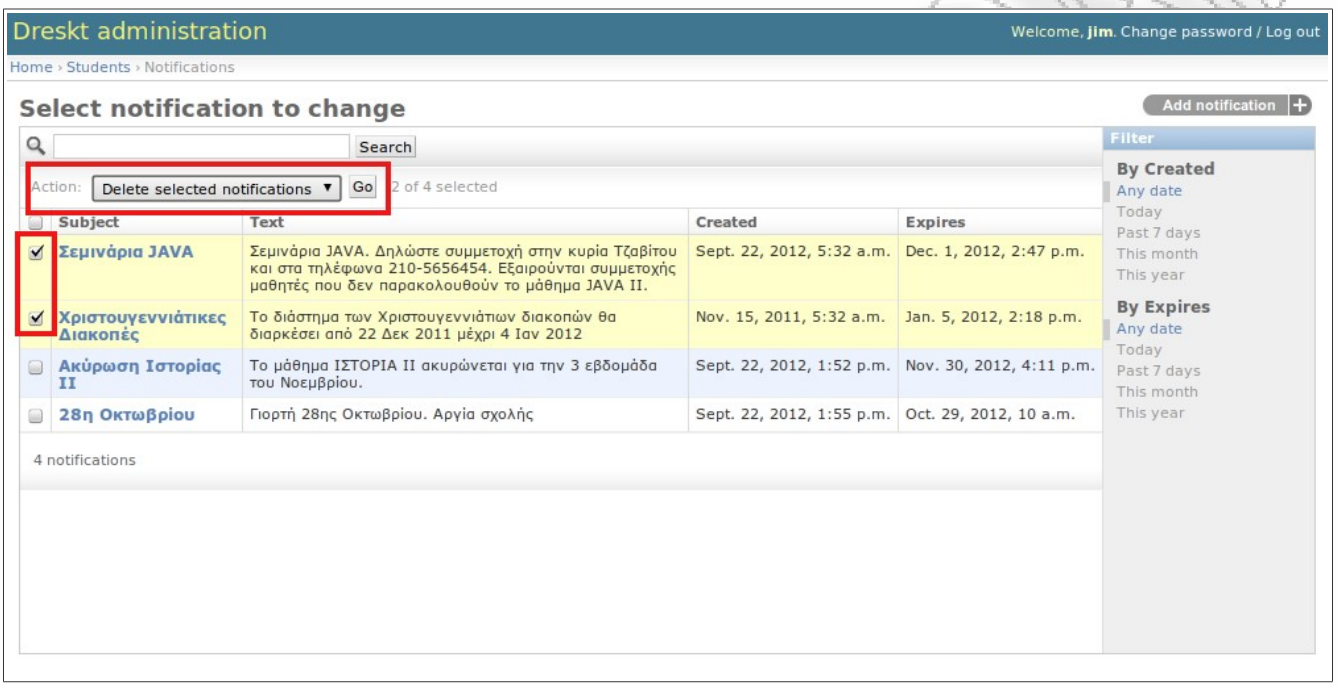
Εικόνα 48: Επιτυχής καταχώρηση ειδοποίησης

5.2.4.3 Τροποποίηση & Διαγραφή

Για να τροποποιήσουμε μια ειδοποίηση επισκεπτόμαστε την αρχική σελίδα των ειδοποιήσεων και στη λίστα κάνουμε κλικ πάνω στο θέμα. Εκεί μας ανοίγει η φόρμα με όλες τις λεπτομέρειες της ειδοποίησης όπου και ακολουθούμε τη διαδικασία που περιγράφηκε στο προηγούμενο κεφάλαιο.

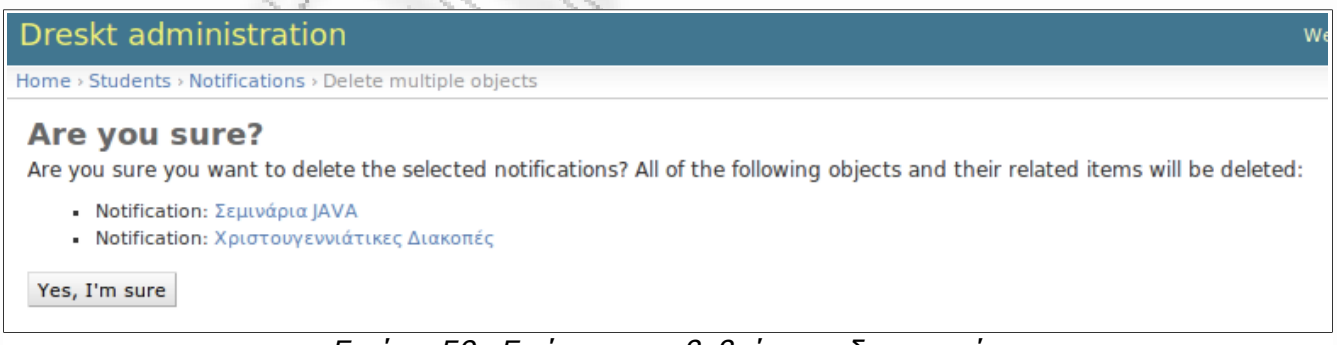
Η διαγραφή ειδοποίησης μπορεί να επιτευχθεί με δύο τρόπους. Ο πρώτος είναι από την αρχική οθόνη των ειδοποιήσεων. Επιλέγουμε μια ή περισσότερες ειδοποιήσεις με τη χρήση των checkboxes που βρίσκονται στο αριστερό μέρος κάθε γραμμής της λίστας. Στη συνέχεια χρησιμοποιώντας το dropdown μενού που βρίσκεται

πάνω από τη λίστα, επιλέγουμε "Delete selected notifications" και πατάμε το κουμπί "Go".



Εικόνα 49: Επιλογή ειδοποιήσεων για διαγραφή

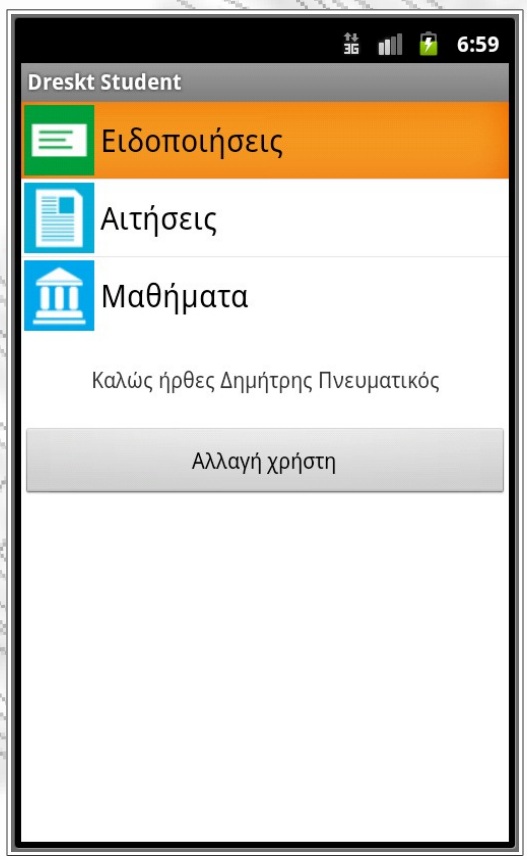
Αμέσως, οδηγούμαστε σε ερώτηση επιβεβαίωσης διαγραφής. Η ερώτηση αυτή γίνεται για λόγους ασφαλείας, αποτρέποντας τον χρήστη στο να κάνει μια βιαστική ενέργεια.



Εικόνα 50: Ερώτηση επιβεβαίωσης διαγραφής

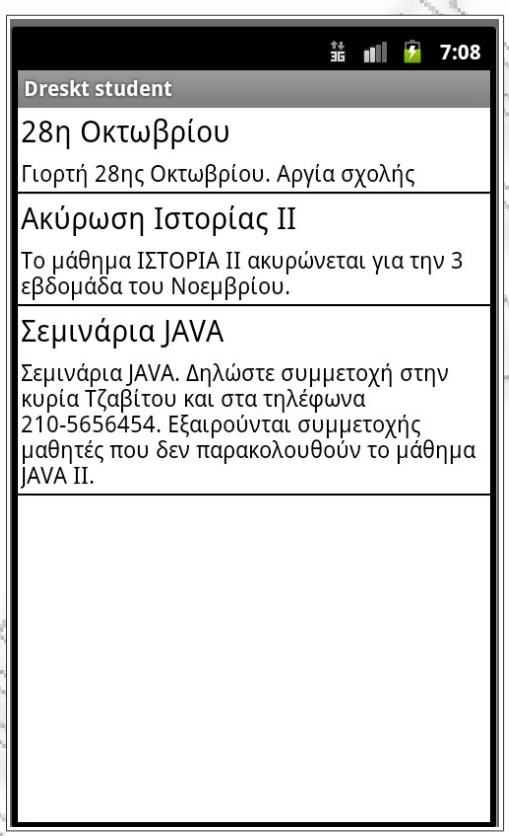
5.2.4.4 Επισκόπηση ειδοποιήσεων στο Dreskt Student

Σε αυτή τη φάση του σεναρίου, οι υπάλληλοι της γραμματείας, έχουν καταχωρήσει κάποιες ειδοποιήσεις. Σκοπός είναι οι ειδοποιήσεις αυτές να προβληθούν στους μαθητές και να τους ενημερώσουν για έκτατα γεγονότα που αφορούν τη λειτουργία της σχολής. Οι ειδοποιήσεις εκτός από τη φυσική τους ύπαρξη στη βάση δεδομένων υφίστανται τώρα και ως αναπαραστάσεις πόρων εκτεθιμένες στο διαδίκτυο με τη βοήθεια των web services. Αφού ο μαθητής εξουσιοδοτηθεί από το σύστημα μπορεί, πάλι με τη χρήστη της εφαρμογής πελάτη να δει όλες τις ειδοποιήσεις. Ο μαθητής επιλέγει ειδοποιήσεις από την αρχική οθόνη της εφαρμογής πατώντας πάνω στο αντίστοιχο εικονίδιο.



Εικόνα 51: Επιλογή ειδοποιήσεων

Στη συνέχεια το Dreskt Student επικοινωνεί με τον κεντρικό server της γραμματείας, και αφού στείλει τα στοιχεία εισόδου[14], τότε λαμβάνει τη λίστα ειδοποιήσεων. Η λίστα με τις ειδοποιήσεις έρχεται σε μορφή JSON αντικειμένων τα οποία μεταφράζονται από το Dreskt Student σε αντικείμενα Java και στη συνέχεια προβάλλονται με τη μορφή λίστας όπως φαίνεται στην εικόνα 52.



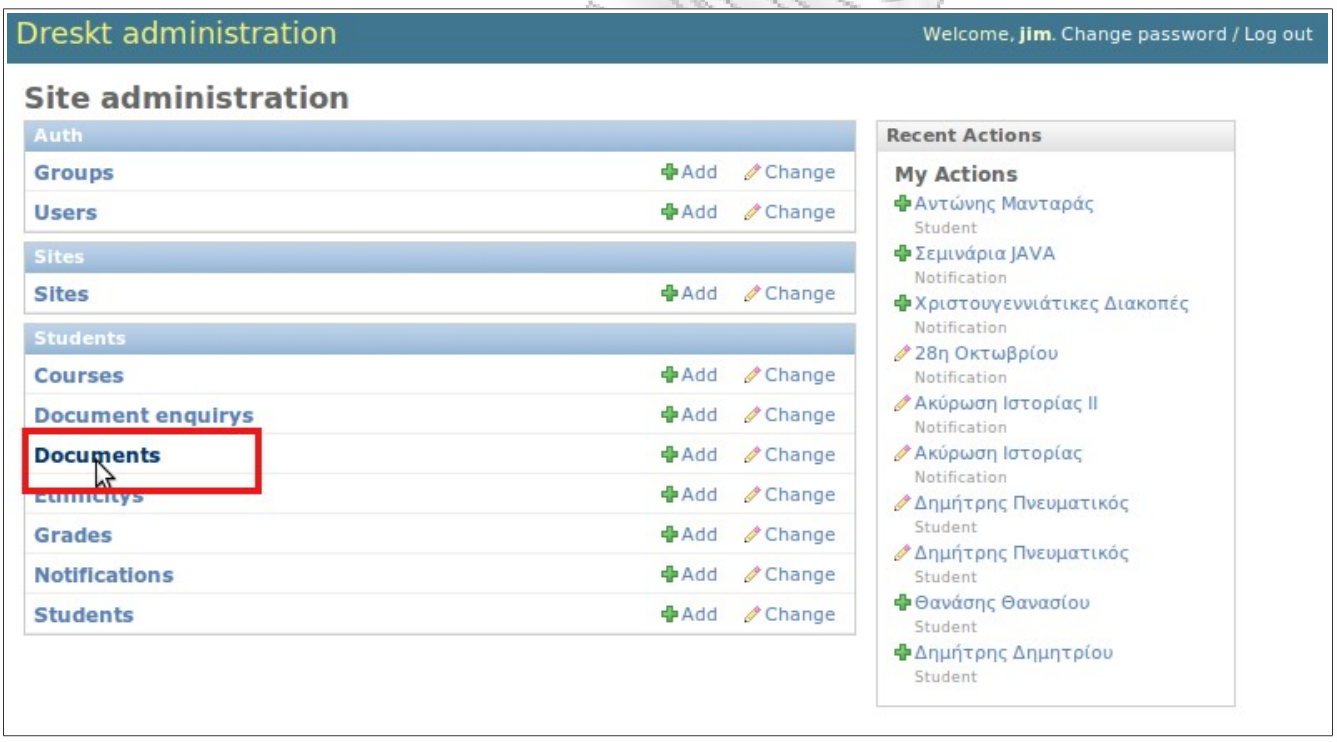
Εικόνα 52: Οι ειδοποιήσεις της σχολής

14 Σημειώνεται οτι μπορεί το Dreskt Student να αποθηκεύει τα στοιχεία εισόδου του μαθητή παρ'όλα αυτά επειδή η αρχιτεκτονική REST είναι χωρίς κατάσταση (stateless) στον εξυπηρετητή, επιβάλλεται να στέλνονται σε κάθε αίτημα.

5.2.5 Έγγραφα

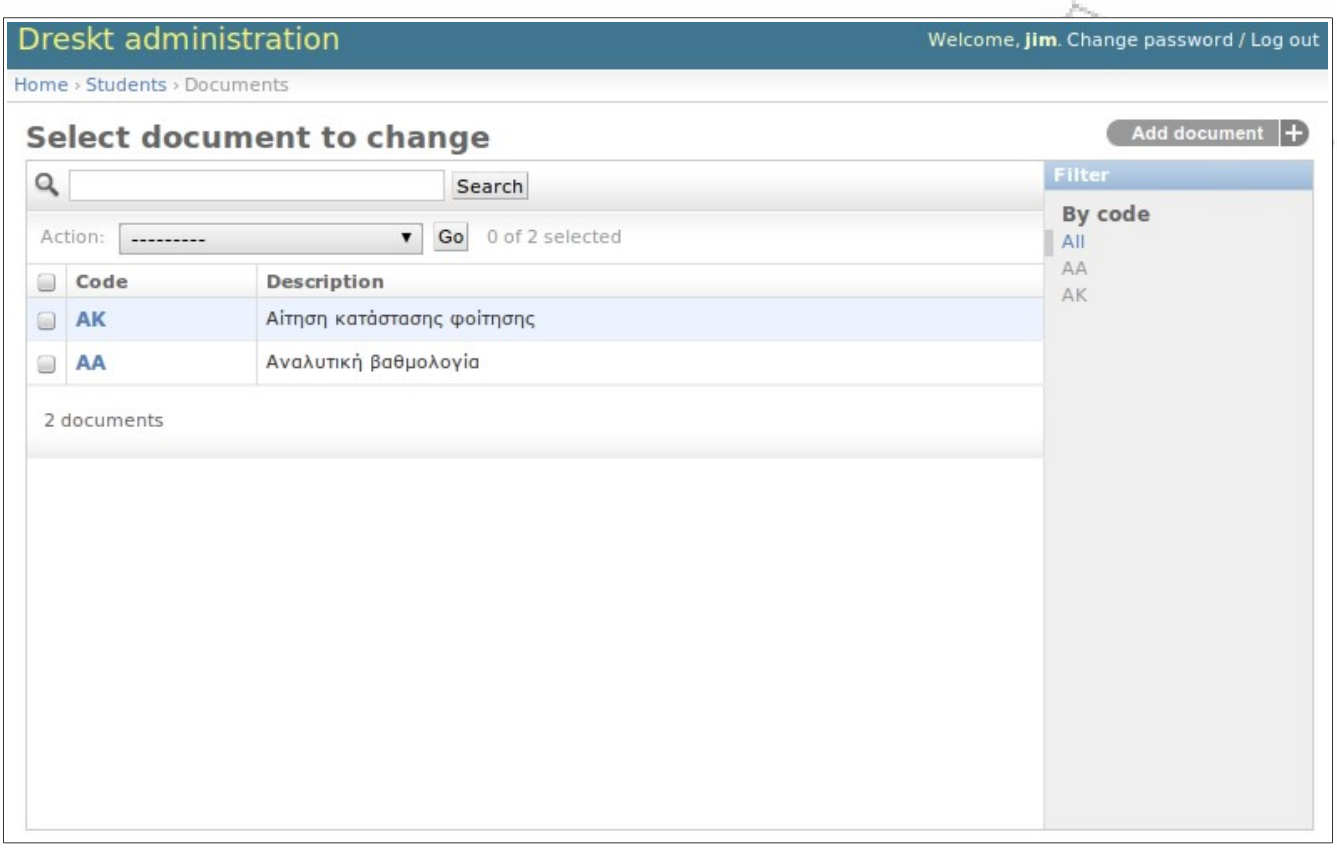
5.2.5.1 Προβολή εγγράφων

Τα έγγραφα είναι κυρίως ειδικές αιτήσεις που απευθύνει ένας μαθητής προς τη γραμματεία για ειδικούς σκοπούς. Για παράδειγμα ένας μαθητής μπορεί να αιτηθεί ένα έγγραφο αναλυτικής βαθμολογίας το οποίο καταγράφει τη βαθμολογική κατάσταση στα μαθήματα που έχει παρακολουθήσει έως τώρα. Για να προβάλλουμε τα διαθέσιμα έγγραφα ή τους διαθέσιμους τύπους εγγράφων, επιλέγουμε Documents από την αρχική οθόνη του συστήματος της γραμματείας.



Εικόνα 53: Επιλογή Εγγράφων από την αρχική σελίδα του Dreskt

Στη συνέχεια εμφανίζεται η λίστα με τους τύπους των εγγράφων που έχει διαθέσιμα η γραμματεία.

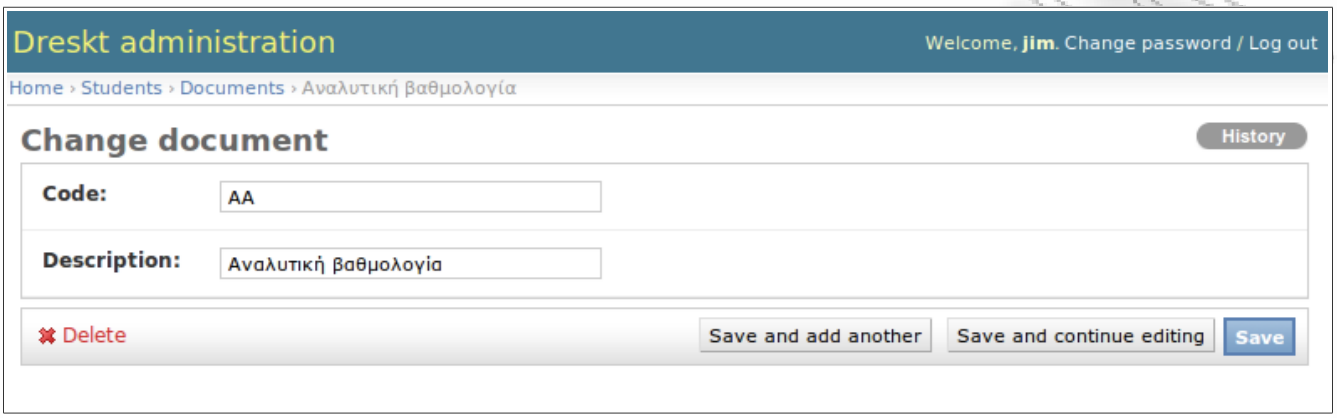


Εικόνα 54: Καταχωρημένοι τύποι εγγράφων

Η λειτουργικότητα της οθόνης των εγγράφων είναι γνώριμη και συναντά πολλές ομοιότητες με αυτή που είδαμε προηγουμένως στις ειδοποιήσεις. Τα φίλτρα βρίσκονται στο δεξί μέρος της οθόνης και το πλαίσιο αναζήτησης το οποίο αναζητά με βάση τον κωδικό ή την περιγραφή, στο πάνω μέρος της λίστας.

5.2.5.2 Καταχώρηση νέου εγγράφου

Η καταχώρηση νέου τύπου εγγράφου επιτυγχάνεται μέσα από τη φόρμα εγγράφων. Δύο στοιχεία απαιτούνται για την δημιουργία ενός νέου τύπου εγγράφου : ο διψήφιος κωδικός του και μια περιγραφή.

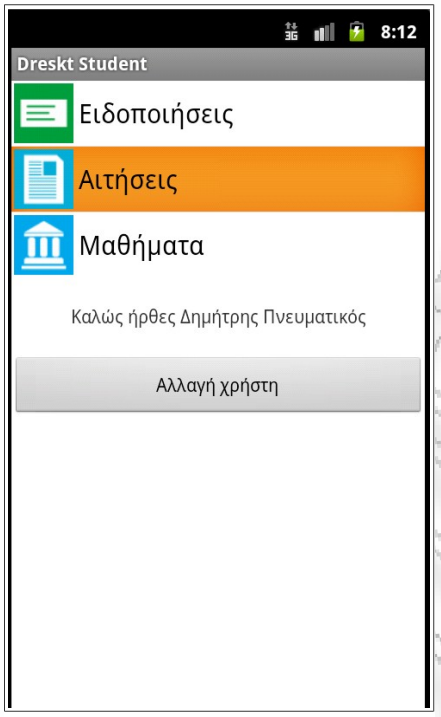


Εικόνα 55: Καταχώρηση νέου τύπου εγγράφου

Με την ολοκλήρωση της καταχώρησης το έγγραφο αυτό είναι διαθέσιμο και στους μαθητές προς αίτηση.

5.2.5.3 Αίτηση εγγράφων από τους μαθητές

Για την αίτηση αιτήσεων εγγράφων από τους μαθητές είναι μια εξαιρετικά απλή διαδικασία. Ο μαθητής επιλέγει "Αιτήσεις" από το αρχικό μενού επιλογών του Dreskt Student.

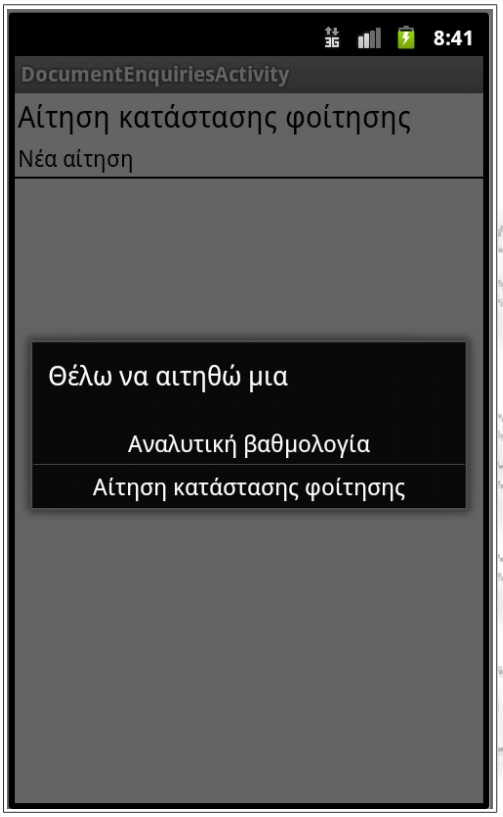


Εικόνα 56: Επιλογή "Αιτήσεις"

Στη συνέχεια οδηγείται στη λίστα με τις αιτήσεις που έχει υποβάλλει ως τώρα. Στη λίστα φαίνονται ο τύπος του εγγράφου που έχει αιτηθεί και η κατάσταση της αιτήσεως στο κάτω μέρος (εικόνα 57). Εάν έχει υποβάλλει κάποια αίτηση αυτή θα εμφανίζεται στη λίστα, εναλλακτικά μπορεί να προχωρήσει στην υποβολή μιας νέας αίτησης εγγράφου. Πατώντας το κουμπί της κινητής συσκευής "Menu" εμφανίζεται η επιλογή "Νέα Αίτηση" όπως φαίνεται στην εικόνα 57.

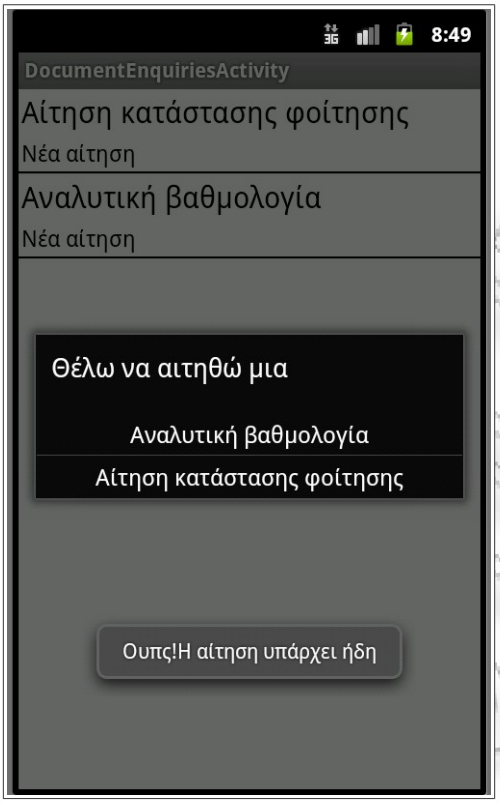
Εικόνα 57: Λίστα αιτήσεων & υποβολή νέας

Το Dreskt Student σε αυτή τη φάση, επικοινωνεί και πάλι με τον κεντρικό εξυπηρετητή της γραμματείας και φέρνει τους διαθέσιμους τύπους εγγράφων.



Εικόνα 58: Υποβολή νέας αίτησης από διαθέσιμους τύπους εγγράφων

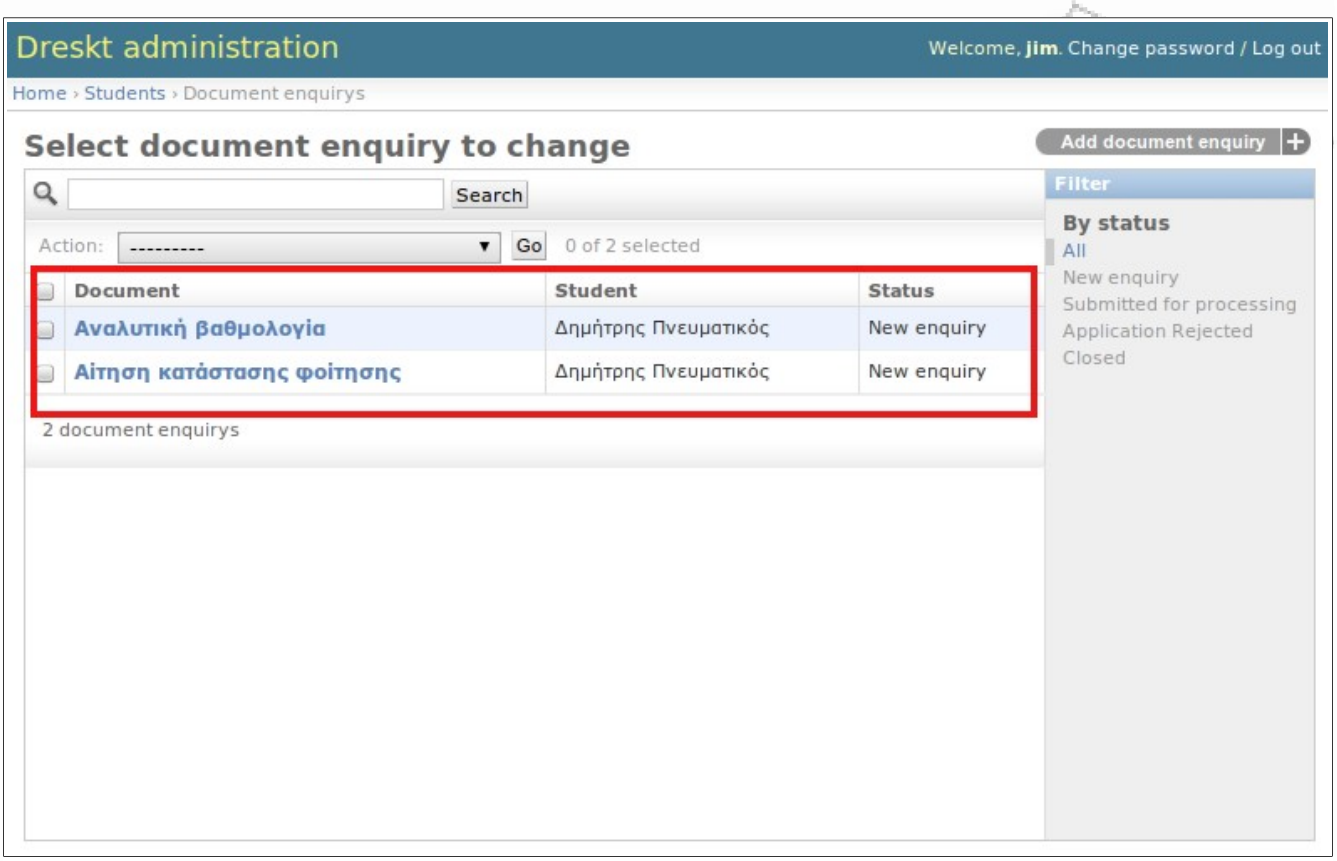
Το μοναδικό πράγμα που έχει να κάνει ο μαθητής είναι να πατήσει πάνω σε κάποια από τις εμφανιζόμενες επιλογές και η αίτηση θα αποσταλλεί στο σύστημα της γραμματείας. Αξίζει να αναφερθεί πως ο μαθητής μπορεί να επιβάλλει μόνο μια αίτηση για κάθε διαθέσιμο τύπο εγγράφων κάθε φορά (εικόνα 59). Μόνο όταν η κατάσταση της αίτησης γίνει "Ολοκληρώθηκε" θα μπορεί να αιτηθεί καινούρια.



Εικόνα 59: Απόρριψη αίτησης από το σύστημα

5.2.5.4 Διαχείριση αιτήσεων από τη γραμματεία

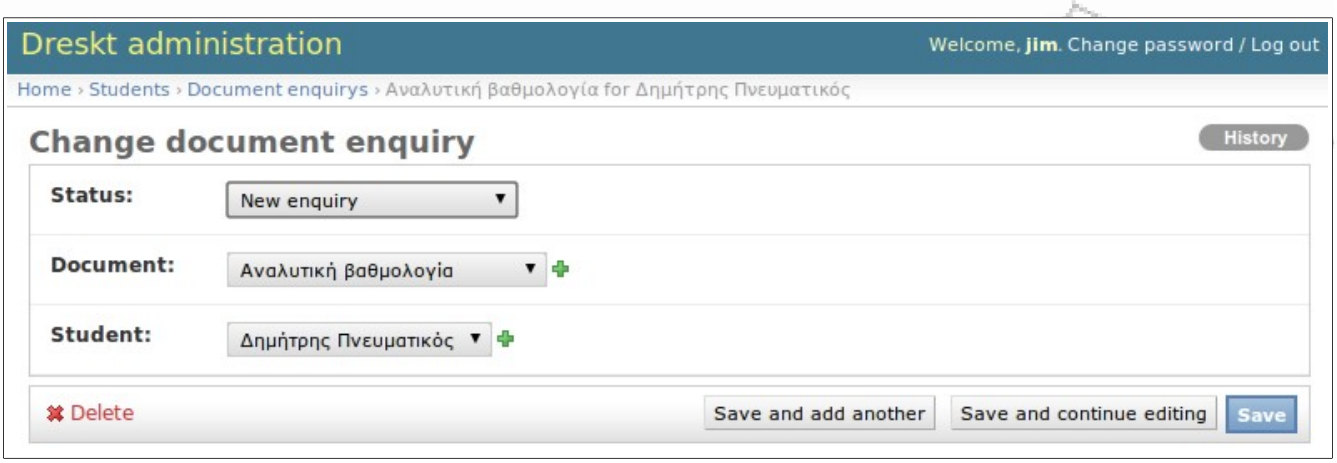
Η διαχείριση της πορείας των αιτήσεων από τη γραμματεία γίνεται μέσα από το μενού "DocumentEnquiries" που βρίσκεται στην αρχική οθόνη του Dreskt. Εκεί προβάλλεται η λίστα όλων των αιτήσεων που έχουν γίνει από τους μαθητές της σχολής. Έχουμε ήδη καταχωρήσει δύο αιτήσεις οπότε αυτές πρέπει να βρίσκονται στη λίστα μας.



Εικόνα 60: Οι αιτήσεις που υποβλήθηκαν από τον μαθητή Δημήτρη Πνευματικό.

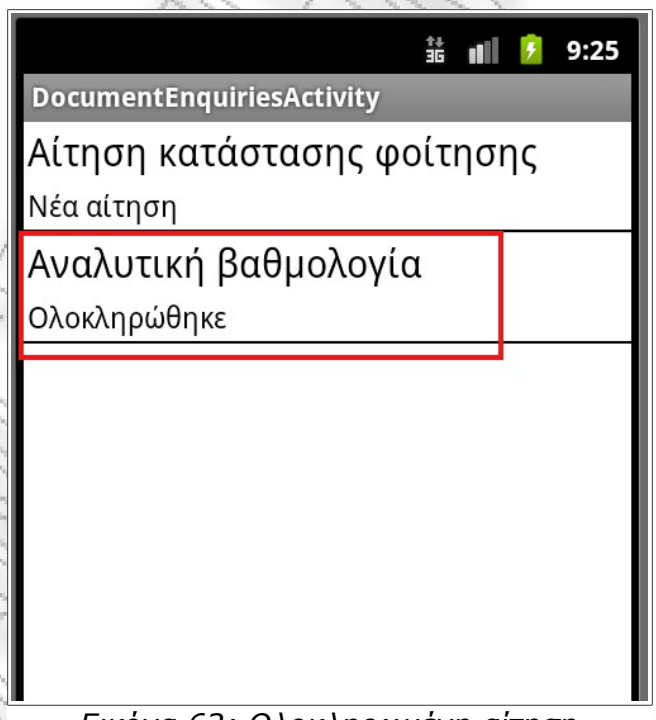
Κάθε νέα αίτηση καταχωρείται αυτόματα με κατάσταση "New enquiry". Η χρήστες της γραμματείας έχουν τη δυνατότητα να αλλάξουν τη κατάσταση της αίτησης σε ορισμένες τιμές :

1. Υποβλήθηκε για επεξεργασία (Submitted for processing)
2. Απορρίφθηκε (Application Rejected)
3. Ολοκληρώθηκε (Closed)



Εικόνα 61: Ανάλυση αίτησης

Ο μαθητής έχει τη δυνατότητα να βλέπει την κατάσταση των αιτήσεων του χρησιμοποιώντας και πάλι τη λίστα των Αιτήσεων. Μόλις η κατάσταση γίνει "Ολοκληρώθηκε" ο μαθητής μπορεί να προσέλθει στη γραμματεία της σχολής για να παραλάβει το έγγραφο.



Εικόνα 62: Ολοκληρωμένη αίτηση

5.3 Συμπεράσματα και μελλοντικές επεκτάσεις

Στο παρόν κεφάλαιο μελετήσαμε μερικά από τα σενάρια χρήσης του συστήματος Dreskt. Κύριος σκοπός ήταν να δείξουμε πως με τη χρήση των web services εκθέσαμε μέρος της λειτουργικότητας μια γραμματείας καθιστώντας την εξυπηρέτηση των μαθητών πιο εύκολη και πιο γρήγορη, και διατηρώντας την αξιοπιστία και την ασφάλεια. Επιπλέον με τη χρήση της κινητής συσκευής Android κάναμε τις διαδικασίες πιο φιλικές προς του μαθητές και πολύ πιο εύκολες κυριολεκτικά με τη χρήση ενός κουμπιού. Η χρήση της πλατφόρμας Android για την ανάπτυξη της εφαρμογής του πελάτη αποτελεί τρανό παράδειγμα της ευελιξίας που προσφέρουν τα web services. Εφαρμογές πελάτες θα μπορούσαν να αναπτυχθούν σε οποιαδήποτε άλλη πλατφόρμα εφ'όσων υλοποιούσαν τις συμβάσεις που προέρχονται από την αρχιτεκτονική του κεντρικού συστήματος. Παρ'όλα αυτά τέτοιες υλοποιήσαν θα ήταν ασφαλέστερο να προέρχονται από το ίδιο το εκπαιδευτικό ίδρυμα και όχι από τρίτους γιατί ενδεχομένως, στην περίπτωση του Dreskt να είχαμε απώλεια κωδικών πρόσβασης των χρηστών. Για αυτό το λόγο το Dreskt επιλέχθηκε να εκθέσει ένα υποσύνολο λειτουργιών της γραμματείας όπου ακόμα και η απώλεια κωδικού πρόσβασης να μη συνιστά κίνδυνο για το σύστημα.

Επιπρόσθετα, η χρήση του JSON σαν κύριο πρωτόκολλο μεταφοράς των πόρων έχει πολλά οφέλη. Το βασικότερο όλων είναι το μικρό του μέγεθος. Οι τεχνικές υποδομές δε θα έπρεπε να είναι κατ'ανάγκη υψηλές και επομένως ακόμα και μικρότεροι εκπαιδευτικοί οργανισμοί, που δε θα μπορούσαν να διαθέσουν ένα σημαντικό οικονομικό κεφάλαιο σε υποδομές, να ήταν ικανοί λειτουργήσουν χωρίς προβλήματα.

Οι κατασκευαστές συστημάτων διαχείρισης μαθητών σίγουρα θα μπορούσαν να επωφεληθούν από τη χρήση των web services και αποτελεί άποψη του γράφοντος οτι στο κοντινό μέλλον θα δούμε τέτοια συστήματα να επεκτείνονται. Το Dreskt θα μπορούσε επίσης να επεκταθεί αφομοιώνοντας λειτουργικότητα από το OpenSis ή

από το Project Fedena ώστε να γίνει πιο ολοκληρωμένο σαν σύστημα.

Βιβλιογραφία

[1] Fielding, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures.* UNIVERSITY OF CALIFORNIA, 2000. <*http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm*>

[2] Alex Rodriguez. *RESTful Web services: The basics.* 2008. <http://www.ibm.com/developerworks/webservices/library/ws-restful/>

[3] IBM Services Architecture Team. *Web Services architecture overview.* 2000 <http://www.ibm.com/developerworks/webservices/library/w-ovr/>

[4] Richardson, Leonard and Ruby, Sam. *Restful web services.* O'Reilly. May 2007.

[5] Haas, Hugo and Brown Allen. *Web Services Glossary.* February 2004. <http://www.w3.org/TR/ws-gloss/ >

[6] Payam, Shodjai. *Web Services and the Microsoft Platform*. June 2006. <http://msdn.microsoft.com/en-s/library/aa480728.aspx#wsmsplat_topic2 >

[7] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, David Orchard. *Web Services Architecture.* February 2004. <http://www.w3.org/TR/ws-arch/>

[8] Nurzhan Nurseitov, Michael Paulson, Randall Reynolds, Clemente Izurieta . *Comparison of JSON and XML Data Interchange Formats: A Case Study .* 2009

[9] W3C. *SOAP tutorial.* <http://www.w3schools.com/soap/>

[10] W3C. *SOAP Version 1.2 Messaging Framework.* April 2007. <http://www.w3.org/TR/soap12-part1/>

[11] Bellwood, Tom. *Understanding UDDI.* July 2002.

<http://www.ibm.com/developerworks/webservices/library/ws-featuddi/>

[12] W3C. *WSDL Tutorial.* <http://www.w3schools.com/wsdl/default.asp>

[13] Lacey, Pete. *S stands for simple*. November 2011. <http://wanderingbarque.com/nonintersecting/2006/11/15/the-s-stands-for-simple/>

[14] Haas, Hugo. *Reconcilling web services and REST services.* W3C. November 2005. <http://www.w3.org/2005/Talks/1115-hh-k-ecows/#(1)>

[15] Berners-Lee, Tim. *A short history of "Resource" in web architecture. August 2009.* <http://www.w3.org/DesignIssues/TermResource.html>

[16] Berners-Lee, Tim. *Uniform Resource Identifiers (URI) : Generic Syntax*. August 1998. <http://tools.ietf.org/html/rfc2396>

[17] Fielding, Roy. *Method Definitions, Hypertext Transfer Protocol – HTTP/1.1*. <http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>

[18] Berners-Lee, Tim. *Cool URI's don't change.* Style guide for online hypertext. Retrieved 19 August 2012. <http://www.w3.org/Provider/Style/URI.html>

[19] Project Fedena. <http://www.projectfedena.org/>

[20] OpenSiS. <http://opensis.com/>

[21] Wikipedia. *Student Information System.* <http://en.wikipedia.org/wiki/Student_information_system>

[22] Wikipedia. Multitier architecture. <http://en.wikipedia.org/wiki/Multitier_architecture>

[23] Microsoft. *N-Tier Data Applications Overview*. <http://msdn.microsoft.com/en-us/library/bb384398.aspx>

[24] JBoss Community. *What is object relational mapping.* <http://www.hibernate.org/about/orm>

[25] Wikipedia. *Object-relational mapping.* <http://en.wikipedia.org/wiki/Object-relational_mapping>

[26] SQLite.org. *The SQLite database.* <http://www.sqlite.org/docs.html>

[27] Tastypie. <http://django-tastypie.readthedocs.org/en/latest/index.html>

[28] The Django Framework. <https://www.djangoproject.com/>

[29] The Django book. <http://www.djangobook.com/>

[30] JSON. <http://www.json.org/>

Παράρτημα Α: Οδηγίες εγκατάστασης συστήματος Dreskt

Το σύστημα Dreskt αναπτύχθηκε στο λειτουργικό σύστημα Ubuntu Linux με τη χρήση των :

Python	γλώσσα προγραμματισμού	Έκδοση 2.7
Django	Βιβλιοθήκη ανάπτυξης εφαρμογών στο διαδίκτυο	Έκδοση 1.4
South	Πρόσθετο του Django	Έκδοση 0.7.6
Tastypie REST Framework	Πρόσθετο του Django	Έκδοση 0.9.12
Android-SDK	Περιβάλλον ανάπτυξης εφαρμογών Android	API Έκδοση 10
Eclipse SDK	Ολοκληρωμένο περιβάλλον ανάπτυξης	Έκδοση 4.2
SQLite	Βάση δεδομένων	Έκδοση 3.7.1

Α.1 Εγκατάσταση του εξυπηρετητή & του περιβάλλοντος της γραμματείας

Για να γίνει εγκατάσταση του συστήματος τα εξής βήματα απαιτούνται :

1. Εγκατάσταση της γλώσσας προγραμματισμού Python. Τη γλώσσα μπορούμε να την κατεβάσουμε από την επίσημη ιστοσελίδα της στο διαδίκτυο στη διεύθυνση : http://www.python.org/download/. Εναλλακτικά στην κονσόλα του Ubuntu Linux πληκτρολογούμε :

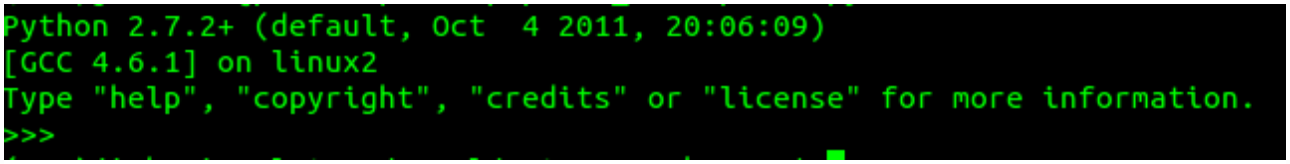
   sudo apt-get install python

2. Μόλις ολοκληρωθεί η εγκατάσταση της Python πληκτρολογούμε στην

κονσόλα :

python

και κανονικά πρέπει να μας εμφανίσει :



Εικόνα 63: Επιτυχής εγκατάσταση της Python

3. Στη συνέχεια εγκαθιστούμε το Django που απαιτείται για χρήση της εφαρμογής της γραμματείας. Πληκτρολογούμε στην κονσόλα :

   sudo pip install django

   Μόλις τελειώσει η εγκατάσταση του Django τρέχουμε :

   django-admin.py

   και ελέγχουμε οτι όντως αυτή η εντολή είναι διαθέσιμη.

4. Εγκαθιστούμε το django extension με το όνομα South :

   pip install south

5. Εγκαθιστούμε το Tastypie REST framework :

   pip install django-tastypie

6. Έχουμε ολοκληρώσει την εγκατάσταση του περιβάλλοντος που τρέχει το Dreskt. Στη συνέχεια παίρνουμε το project Dreskt από το GitHub ή εναλλακτικά το αντιγράφουμε από το CD της διπλωματικής σε έναν φάκελο στον υπολογιστή μας. Για να πάρουμε το project από το GitHub πρέπει να έχουμε εγκατεστημένο έναν Git πελάτη[15]. Το project βρίσκεται ανεβασμένο στη διεύθυνση :

15 Το Git είναι ένα Version Control System ή Σύστημα Διαχείρισης Εκδόσεων κώδικα. Για περισσότερα βλέπε: http://git-scm.com/book/en/Git-Basics-Getting-a-Git-Repository.

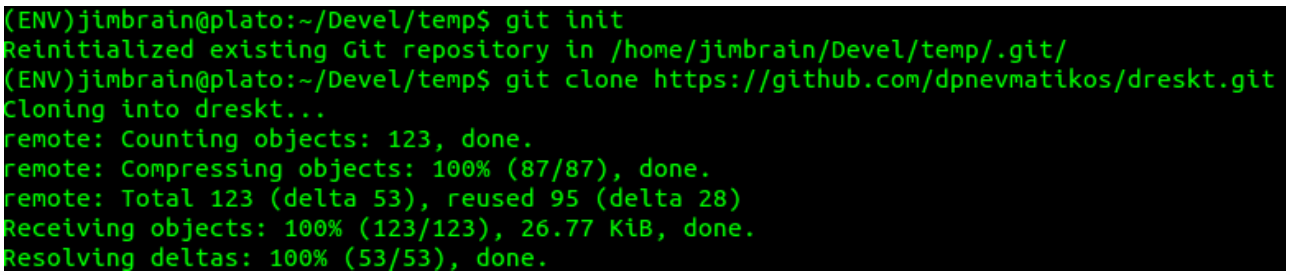
https://github.com/dpnevmatikos/dreskt

Για να το κατεβάσουμε από τη διεύθυνση αυτή και αφού έχουμε ολοκληρώσει την εγκατάσταση του Git στο Ubuntu Linux πληκτρολογούμε διαδοχικά :

git init

git clone https://github.com/dpnevmatikos/dreskt.git

Η πρώτη εντολή είναι για να δημιουργήσει ένα Git repository στο φάκελο που βρισκόμαστε και η δεύτερη κατεβάζει το project από το online repository. Μετά τη λήψη πρέπει να δούμε την παρακάτω οθόνη :



Εικόνα 64: Επιτυχής λήψη του Dreskt από το GitHub

7. Προχωράμε στη δημιουργία της βάσης δεδομένων για το σύστημα :

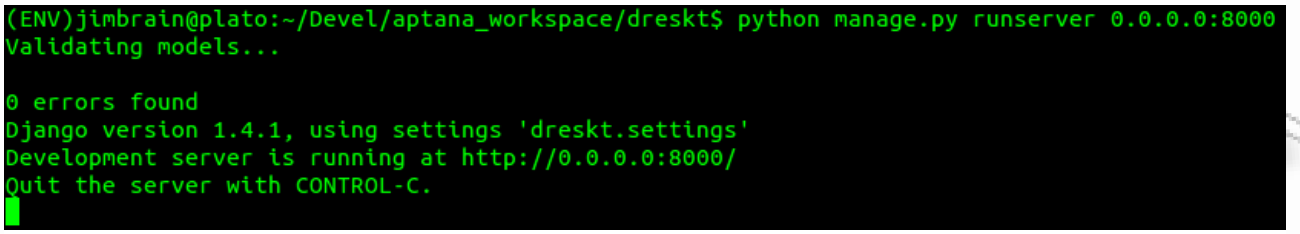
   python manage.py schemamigration students –initial

   python manage.py syncdb

   python manage.py migrate students

8. Τέλος και έχοντας ακολουθήσει τα βήματα σωστά είμαστε έτοιμοι να εκκινήσουμε τον εξυπηρετητή του συστήματος. Πληκτρολογούμε :

   python manage.py runserver 0.0.0.0:8000



Εικόνα 65: Επιτυχής εκκίνηση του εξυπηρετητή του Django

9. Ανοίγουμε τον φυλλομετρητή του διαδικτύου και πληκτρολογούμε τη διεύθυνση :

   http://localhost:8000/admin

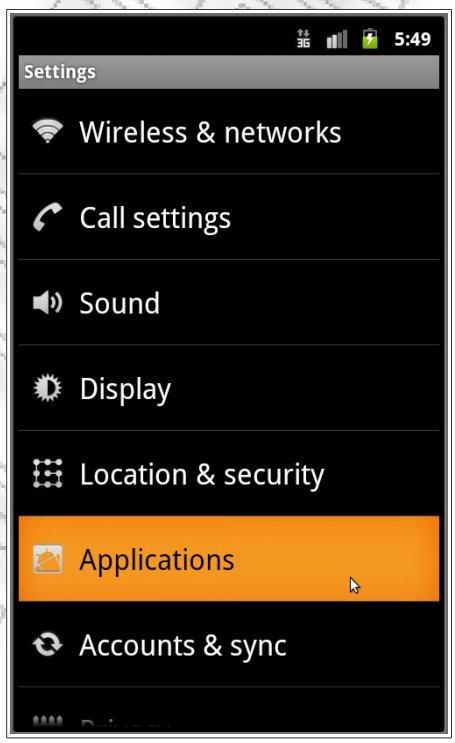
Α.2 Εγκατάσταση της εφαρμογής πελάτη στο Android

Η εγκατάσταση στο περιβάλλον Android απαιτεί την ύπαρξη συσκευής με έκδοση **Android 2.3.3 GingerBread**.

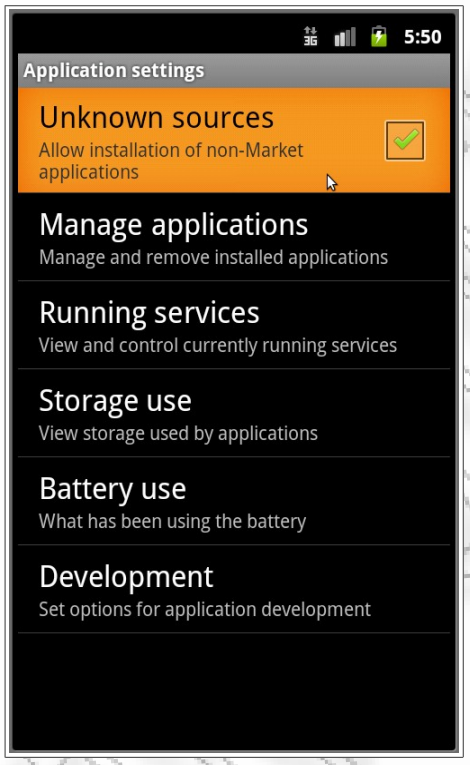
Για να εγκαταστήσουμε την εφαρμογή του πελάτη στο Android ακολουθούμε τα εξής βήματα :

1. Από το αρχικό μενού επιλογών της συσκευής επιλέγουμε Settings.

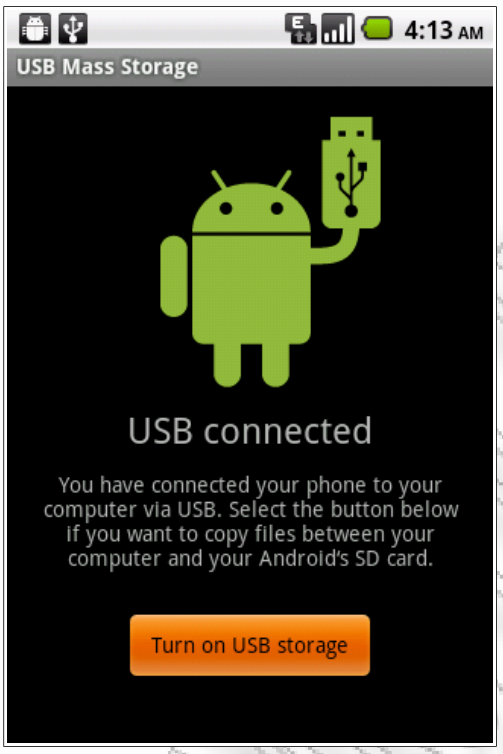
2. Στη λίστα που εμφανίζεται επιλέγουμε Applications (Εφαρμογές) :



3. Τσεκάρουμε την επιλογή Uknown sources (άγνωστες πηγές) η οποία μας επιτρέπει να εγκαθιστούμε εφαρμογές εκτός από αυτές του Google Play Store :



4. Στη συνέχεια συνδέουμε τη συσκευή στον υπολογιστή μας και μόλις συνδεθεί ενεργοποιούμε την πρόσβαση στον αποθηκευτικό χώρο της συσκευής όπως φαίνεται στην παρακάτω εικόνα:



5. Τέλος αντιγράφουμε το αρχείο με όνομα "*Dreskt student.apk*" στην sdcard της συσκευής και κάνοντας κλικ πάνω στο αρχείο ξεκινάει η διαδικασία εγκατάστασης. Αν όλα γίνουν σωστά τότε θα δούμε το εικονίδιο της εφαρμογής στη λίστα με τις εφαρμογές μας.