ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ

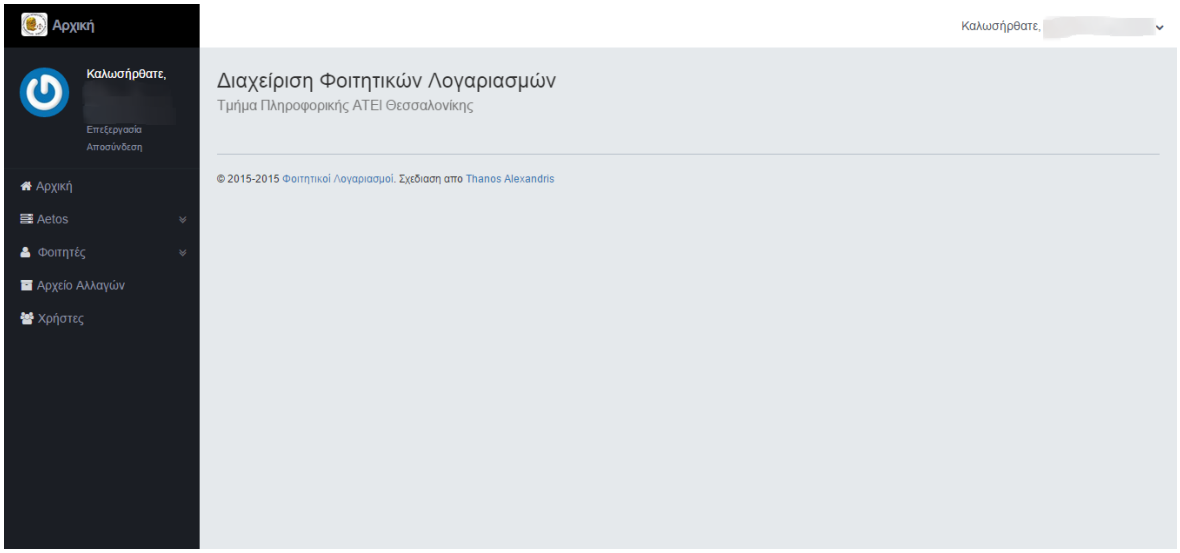
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

Τμήμα Μηχανικών Πληροφορικής ΑΤΕΙΘ

Πτυχιακή εργασία

«Σύστημα διαχείρισης φοιτητικών λογαριασμών»



Του φοιτητή
Αλεξανδρή Αθανάσιου
Αρ. Μητρώου: 042526

Επιβλέπων καθηγητής
Αμανατιάδης Δημήτριος

Θεσσαλονίκη 2015

ΠΡΟΛΟΓΟΣ

Η δημιουργία ενός πληροφοριακού συστήματος που έχει να κάνει με φοιτητές και λογαριασμούς αυτών, είναι ένα από τα κυριότερα εργαλεία που πρέπει να διαθέτει ένα τμήμα μιας οποιασδήποτε σχολής. Το σύστημα αυτό θα πρέπει να πληροί όλες τις προϋποθέσεις, καθώς έχει να κάνει με προσωπικά δεδομένα και πληροφορίες φοιτητών.

Ο στόχος της παρούσας πτυχιακής εργασίας είναι η δημιουργία μιας νέας διαδικτυακής εφαρμογής που θα ασχολείται με τους λογαριασμούς των φοιτητών. Η εργασία αποτελείται από δυο μέρη. Το θεωρητικό το οποίο αποτελεί το παρών έγγραφο και το πρακτικό μέρος. Στο πρακτικό μέρος δημιουργήθηκε μια διαδικτυακή εφαρμογή μέσω της οποίας μπορεί και ενημερώνετε η βάση των φοιτητών και των καταστάσεών τους, δίνεται η δυνατότητα αλλαγής των πληροφοριών των λογαριασμών τους και ακόμα η παρουσίαση των πληροφοριών αυτών με στατιστικά διαγράμματα. Έτσι μπορεί εύκολα ο διαχειριστής του συστήματος να εξάγει συμπεράσματα σχετικά με τους χρόνους φοίτησης, διαγραφών κλπ.

Η πτυχιακή αυτή δημιουργήθηκε με σκοπό την επίλυση προβλημάτων που αντιμετωπίζει το σύστημα διαχείρισης φοιτητικών λογαριασμών του server hydra του τμήματος μηχανικών πληροφορικής του ΑΤΕΙΘ.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία είχε στόχο την ανάπτυξη ενός νέου συστήματος διαχείρισης των φοιτητικών λογαριασμών του τμήματος πληροφορικής, γεγονός το οποίο προέκυψε ως επιτακτική ανάγκη της εξάλειψης των ατελειών του υπάρχοντος συστήματος. Στα κεφάλαια αυτής της εργασίας γίνεται ανάλυση του υπάρχοντος μοντέλου, περιγραφή των αδύναμων σημείων του καθώς και οι λόγοι ανάπτυξης του νέου συστήματος. Επίσης παρατίθενται κάποια χρήσιμα συμπεράσματα αλλά και τα σημαντικότερα τμήματα κώδικα που χρησιμοποιήθηκαν στην ανάπτυξη του νέου συστήματος.

ABSTRACT

This thesis target, was to develop a new system for managing student accounts of IT department, as an imperative of eliminating the current system imperfections. During the chapters of this thesis, you can see the analysis of existing model, description of its weak points and also the reasons of new system's development. However you can see such some useful conclusions as the major code sections of code used in order the new system to be developed.

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία με τίτλο «Σύστημα διαχείρισης φοιτητικών λογαριασμών», εκπονήθηκε στα πλαίσια της πτυχιακής εργασίας του τμήματος Μηχανικών Πληροφορικής του Αλεξάνδρειου Τεχνολογικού Εκπαιδευτικού Ιδρύματος.

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου κ. Δημήτριο Αμανατιάδη για τη συνεχή καθοδήγηση, την συνεχή υποστήριξη του, καθώς και τις συμβουλές που μου παρείχε σε όλο αυτό το διάστημα της πτυχιακής μου εργασίας.

Θέλω να ευχαριστήσω την οικογένεια μου, τους συναδέλφους μου και την κοπέλα μου Υφαντή Πηνελόπη για την στήριξή που μου πρόσφεραν σ ’αυτήν την προσπάθεια μου, και που με την καθημερινή τους συμπαράσταση και την υπομονή τους, συνέβαλαν στην εκπλήρωση του στόχου μου.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ ΚΑΙ ΠΙΝΑΚΩΝ

ΕΙΣΑΓΩΓΗ

Η παρούσα πτυχιακή εργασία, δημιουργήθηκε με σκοπό την αντικατάσταση του υπάρχοντος συστήματος διαχείρισης φοιτητικών λογαριασμών. Η απουσία της πρόβλεψης στον αρχικό σχεδιασμό του συστήματος για την προσθήκη μεταπτυχιακού προγράμματος σπουδών στο τμήμα οδήγησε σε ανεπάρκεια και προβλήματα στο σύστημα. Ωστόσο και η διαδικασία ενημέρωσης των φοιτητών και των καταστάσεων τους αποτελούσε μια χρονοβόρα και αρκετά επίπονη εργασία για τον διαχειριστή του συστήματος, καθώς στην ενημέρωση αυτή λαμβάνουν μέρος 2 server του τμήματος πληροφορικής(hydra και aetos) και ως εκ τούτου μπορεί να προκύψουν προβλήματα συγχρονισμού των πληροφοριών. Με αυτό λοιπόν το σύστημα, προσπάθησε να γίνει μια προσέγγιση ώστε να εξαλειφθούν τα παραπάνω θέματα.

Έτσι, θα μπορούν να γίνουν ευκολότεροι οι τρόποι φιλτραρίσματος των στοιχείων, η ανάλυση και επεξεργασία των πληροφοριών και επίσης θα είναι εφικτή η σύγκριση και η ορθή αξιολόγηση σε σχέση με παλαιότερα δεδομένα των φοιτητών.

Η εργασία αναλύεται σε 4 βασικά κεφάλαια. Το 1ο έχει να κάνει με την προϋπαρξη του συστήματος και τις βασικές αδυναμίες του, το 2ο με τους τρόπους αντιμετώπισης αυτών των αδυναμιών και ελλείψεων, το 3ο με τις τεχνολογίες που χρησιμοποιήθηκαν για την κατασκευή του νέου συστήματος και στο 4ο παρουσιάζονται κάποια συμπεράσματα που εξάχθηκαν.

Οι παραπάνω θεματικές ενότητες, αναλύονται σταδιακά σε κεφάλαια στις παρακάτω σελίδες της εργασίας.

ΚΕΦΑΛΑΙΟ 1

ΥΠΑΡΧΟΥΣΑ ΔΙΑΔΙΚΑΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ ΛΟΓΑΡΙΑΣΜΩΝ

ΕΙΣΑΓΩΓΗ

Το πρώτο κεφάλαιο περιγράφει την υπάρχουσα διαδικασία ενημέρωσης των πληροφοριών και των καταστάσεων των φοιτητικών λογαριασμών που χρησιμοποιούσε ως τώρα ο διαχειριστής του συστήματος.

1.1 ΑΠΟΚΤΗΣΗ ΦΟΙΤΗΤΙΚΟΥ ΛΟΓΑΡΙΑΣΜΟΥ

Η διαδικασία εγγραφής των φοιτητών στο τμήμα, αποτελεί το πρώτο και βασικό βήμα. Αφού η γραμματεία έχει λάβει από το υπουργείο τα στοιχεία των φοιτητών, οι φοιτητές με τη σειρά τους οφείλουν να μεταβούν σ ’αυτή ώστε να ολοκληρώσουν την εγγραφή τους στο τμήμα.

1.2 ΠΡΟΑΠΑΙΤΟΥΜΕΝΑ ΣΤΟΙΧΕΙΑ ΣΥΓΧΡΟΝΙΣΜΟΥ

1.2.1 ΛΗΨΗ ΣΤΟΙΧΕΙΩΝ ΦΟΙΤΗΤΩΝ ΑΠΟ ΤΗΝ ΓΡΑΜΜΑΤΕΙΑ

Ο διαχειριστής του συστήματος λαμβάνει ανά τακτά διαστήματα από το κέντρο διαχείρισης δικτύου(noc), ένα αρχείο τύπου tsv(tab separated values) με όνομα “students.txt”, στο οποίο περιέχονται όλες οι νέες πληροφορίες αλλά και αλλαγές που έχουν υπάρξει στους λογαριασμούς των φοιτητών.

1.2.2 ΑΝΑΛΥΣΗ ΑΡΧΕΙΟΥ STUDENTS.TXT

Το αρχείο αυτό περιέχει όλες τις απαραίτητες πληροφορίες που χρειάζονται για την ενημέρωση της βάσης των φοιτητών τα πεδία του οποίου αναλύονται στον πίνακα 1.

Πίνακας 1: Επεξήγηση πεδίων αρχείου students.txt

student_ID	Αύξων αριθμός για τον κάθε φοιτητή
spec_aem	Ο αριθμός μητρώου του κάθε φοιτητή
first	Όνομα φοιτητή
last	Επώνυμο φοιτητή
fname	Όνομα πατρός φοιτητή
in_year	Έτος εισαγωγής φοιτητή
in_period_ID	Περίοδος εισαγωγής φοιτητή(χειμερινό/εαρινό)
cond_ID	Κατάσταση φοιτητή(ενεργός, διαγραμμένος, απόφοιτος, σε αναστολή σπουδών, από μετεγγραφή, σε αναμονή πτυχίου
dgr_logos	Λόγος διαγραφής φοιτητή(αν υπάρχει)
in_exam_ID	Σε ποιο εξάμηνο ήρθε
exam_ID	Σε ποιο εξάμηνο είναι(τρέχον)
aetos_username	Το username που έχει ο φοιτητής στον aeto

Ο κάθε φοιτητής, αποτελεί μια εγγραφή στο αρχείο, μια εγγραφή ανά γραμμή και για την ακρίβεια είναι τύπου tsv καθώς τα στοιχεία των πεδίων σε κάθε γραμμή είναι χωρισμένα με ένα tab.

1.3 ΣΥΓΧΡΟΝΙΣΜΟΣ ΜΕΤΑΞΥ ΕΞΥΠΗΡΕΤΗΤΩΝ

1.3.1 ΛΗΨΗ ΠΛΗΡΟΦΟΡΙΩΝ ΑΠΟ ΤΟΝ SERVER AETOS

Μετά τη λήψη αυτού του αρχείου, ο διαχειριστής δικτύου είναι υπεύθυνος για την ενημέρωση των δυο server με τα νέα στοιχεία. Αφού λοιπόν συνδεθεί στον server aetos με δικαιώματα διαχειριστή, αντιγράφει το αρχείο “/etc/passwd” στον δικό του τοπικό κατάλογο και έπειτα το κατεβάζει στον προσωπικό υπολογιστή του μέσω sftp σύνδεσης.

1.3.2 ΕΝΗΜΕΡΩΣΗ ΕΓΓΡΑΦΩΝ ΤΟΥ SERVER HYDRA ΑΠΟ ΤΟΝ ΑΕΤΟ

Στο web panel της hydras, ο διαχειριστής δικτύου συνδέεται με τα στοιχεία του, και έπειτα επιλέγει το αρχείο “/etc/passwd” από τον τοπικό του δίσκο μέσω των επιλογών του μενού: IT Students->Import->Password File. Στη συνέχεια πατάει upload χωρίς να είναι τσεκαρισμένη η επιλογή Store data ώστε σε περίπτωση σφάλματος να μην προχωρήσει η ενημέρωση. Αν όλα πάνε καλά, την 2^{η} φορά τσεκάρει το Store data και επιλέγει το upload. Με αυτόν τον τρόπο ενημερώνει τη βάση της hydras με τους λογαριασμούς των φοιτητών στον aeto.

1.4 ΕΝΗΜΕΡΩΣΗ ΦΟΙΤΗΤΩΝ ΤΟΥ SERVER HYDRA

Σε μια διαφορετική καρτέλα του φυλλομετρητή που χρησιμοποιεί ήδη, επιλέγει πάλι από το αριστερό μενού του web panel της hydras την επιλογή: IT Students->Import->Students. Εδώ, παρομοίως με την προηγούμενη διαδικασία ενημέρωσης του "/etc/passwd” αρχείου, κάνει την πρώτη φορά upload το αρχείο students.txt χωρίς να τσεκάρει την επιλογή Update database, και αφού δεν υπάρξει κάποιο πρόβλημα το τσεκάρει και ξαναεπιλέγει το upload.

1.5 ΔΗΜΙΟΥΡΓΙΑ ΠΡΟΤΕΙΝΟΜΕΝΩΝ USERNAMES

Μετά την περάτωση της παραπάνω ενέργειας, μεταβαίνει στην επιλογή IT Students->Students, και από τις λειτουργίες επιλέγει το New Accounts. Εδώ εμφανίζονται όλοι οι νέοι φοιτητές(όχι οι υπάρχοντες στην βάση), οι οποίο προέκυψαν από την ενημέρωση του αρχείου “students.txt”. Σε κάθε γραμμή υπάρχει και ένας φοιτητής με το ονοματεπώνυμο του και το προτεινόμενο από το σύστημα username για αυτόν. Αν θέλει μπορεί να αλλάξει κάποιο από αυτά απλά, και αφού αποφασίσει ότι όλα είναι εντάξει επιλέγει το copy με το οποίο αντιγράφονται στο clipboard του συστήματος τα στοιχεία αυτά, που θα χρησιμοποιηθούν για την μετέπειτα ενημέρωση του aetos server. Το βασικό μειονέκτημα σε αυτήν την διαδικασία είναι ότι ενδέχεται το σύστημα να προτείνει το ίδιο username σε φοιτητές που μπορεί να έχουν το ίδιο όνομα στο αρχείο “students.txt” που ανέβηκε.

1.6 ΕΝΗΜΕΡΩΣΗ ΤΟΥ SERVER AETOS ΜΕ ΤΟΥΣ ΝΕΟΥΣ ΛΟΓΑΡΙΑΣΜΟΥΣ

Τα δεδομένα των προτεινόμενων username, είναι πλέον έτοιμα για εισαγωγή στον aeto. Αφού συνδεθεί μέσω ssh στον aeto(αν δεν είναι ήδη συνδεδεμένος), ο διαχειριστής δημιουργεί ένα αρχείο Stud_xxxx στο οποίο κάνει επικόλληση τα δεδομένα από το clipboard. Στην συνέχεια τρέχει ένα script από τον κατάλογο «/usr/local/adm/oper/adduser» το οποίο δημιουργεί δυο αρχεία. Αν δεν υπάρχει ο αντίστοιχος κατάλογος στον φάκελο «/home/student» τον δημιουργεί. Τέλος τρέχει ένα ακόμα script και με αυτόν τον τρόπο δημιουργούνται οι λογαριασμοί στον aeto. Σε περίπτωση που υπάρχει μήνυμα “Permission denied” το μόνο που χρειάζεται είναι να αλλάξει τα δικαιώματα του script.

1.7 ΕΠΑΝΑΛΗΨΗ ΔΙΑΔΙΚΑΣΙΑΣ ΣΥΓΧΡΟΝΙΣΜΟΥ

Στην φάση αυτή, ο διαχειριστής επαναλαμβάνει την διαδικασία ενημέρωσης(κεφ. 1.3) για να ενημερώσει τις νέες αλλαγές που επήλθαν στον aeto.

1.8 ΑΠΟΘΗΚΕΥΣΗ USERNAMES ΝΕΩΝ ΛΟΓΑΡΙΑΣΜΩΝ ΣΤΗΝ HYDRA

Μετά τον δεύτερο συγχρονισμό της hydras με το νέο αρχείο «/etc/passwd», μεταβαίνει στην πρώτη καρτέλα της hydras New accounts που υπάρχει στον φυλλομετρητή, και επιλέγοντας το apply, ελέγχει τα υπάρχοντα username για προβλήματα. Αν όλα είναι εντάξει χωρίς να προκύψει κάποιο σφάλμα, όλα τα username θα πρέπει να είναι ίδια με τα existing. Στο τέλος τσεκάρει στην στήλη store την επιλογή όλα για όλες τις εγγραφές και έπειτα το κουμπί store δίπλα στο apply.

1.9 ΕΝΗΜΕΡΩΣΗ ΚΩΔΙΚΩΝ ΦΟΙΤΗΤΙΚΩΝ ΛΟΓΑΡΙΑΣΜΩΝ HYDRAS

Το τελευταίο βήμα αυτής της διαδικασίας ενημέρωσης έχει ως εξής. Ο διαχειριστής συνδέεται στον server της hydras μέσω ssh ως χρήστης “plir”. Αφού μεταβεί στον κατάλογο «talos2/itusers/scripts» αντιγράφει το δεύτερο αρχείο που δημιούργησε στον server aetos, και το τοποθετεί σ ’αυτόν τον φάκελο. Έπειτα εκτελεί ένα script, και αφού του ζητήσει τον κωδικό της βάσης δεδομένων, θα το εκτελέσει. Το αποτέλεσμα είναι η εμφάνιση των στοιχείων των φοιτητών με αντιστοιχία ΑΜ, username και password, τα οποία αντιγράφει και τοποθετεί σε κάποιο πρόγραμμα επεξεργασίας κειμένου, προκειμένου να τα επεξεργαστεί, να τα μορφοποιήσει, και να τα εκτυπώσει σε αυτοκόλλητα για να τα μοιράσει στους νέους φοιτητές.

1.10 ΠΡΟΒΛΗΜΑΤΑ ΚΑΙ ΕΛΛΕΙΨΕΙΣ ΠΟΥ ΠΡΟΚΥΠΤΟΥΝ ΑΠΟ ΤΗΝ ΔΙΑΔΙΚΑΣΙΑ ΕΝΗΜΕΡΩΣΗΣ

- Ένα σημαντικό κενό του συστήματος ήταν το γεγονός ότι κατά την πρωταρχική σχεδίαση του, δεν είχε προβλεφθεί η υποστήριξη για μελλοντική υποστήριξη προγράμματος μεταπτυχιακών σπουδών στο τμήμα. Έτσι προέκυψαν νέα δεδομένα και καταστάσεις οι οποίες έπρεπε να αντιμετωπιστούν.

- Στο παρόν σύστημα δεν μπορεί να γίνει ταξινόμηση των φοιτητών βάση του τρέχοντος εξαμήνου τους και της κατάστασής τους, παρά μόνο του εξαμήνου εισαγωγής τους στο τμήμα, γεγονός που καθιστά δύσκολη την εύρεση κάποιου φοιτητή ή φοιτητών βάση ορισμένων φίλτρων.

- Κατά την δημιουργία προτεινόμενων usernames, όπως προαναφέρθηκε στο βήμα 1.5, δεν ελέγχεται αν κάποιο username έχει ήδη δημιουργηθεί από την ίδια ενημέρωση, με αποτέλεσμα να δημιουργούνται ίδια usernames και να δημιουργούνται σφάλματα.

- Επίσης το σύστημα δεν ελέγχει εάν οι προτεινόμενοι κωδικοί είναι reversed dictionary word, με αποτέλεσμα να μην μπορεί να γίνει σύνδεση των φοιτητών που απέκτησαν τον κωδικό αυτό, στον aeto.

- Η διαδικασία της ενημέρωσης, δεν δίνει στο σύστημα την δυνατότητα να δημιουργηθούν στατιστικά των φοιτητών από ενημέρωση σε ενημέρωση.

ΕΠΙΛΟΓΟΣ

Σ’ αυτό το κεφάλαιο έγινε μια καταγραφή των αναγκών και των προβλημάτων που μπορεί να δημιουργηθούν, όπως επίσης η αστάθεια σε ορισμένα σημεία και η πολυπλοκότητα του υπάρχοντος συστήματος, στοιχεία που αποτέλεσαν ισχυρά κίνητρα για την προσέγγιση μιας πιο ολοκληρωμένης λύσης η οποία θα μπορεί να διαχειρίζεται τους φοιτητικούς λογαριασμούς με μεγαλύτερη ασφάλεια, απλότητα, ταχύτητα, αποδοτικότητα και επεκτασιμότητα για καλύτερη απόδοση του συστήματος και ευκολία σε πιθανή μελλοντική της ανάπτυξή. Τα προαναφερθέντα αναλύονται στο επόμενο κεφάλαιο.

ΚΕΦΑΛΑΙΟ 2

ΠΡΟΣΕΓΓΙΣΗ ΝΕΟΥ ΣΥΣΤΗΜΑΤΟΣ

ΕΙΣΑΓΩΓΗ

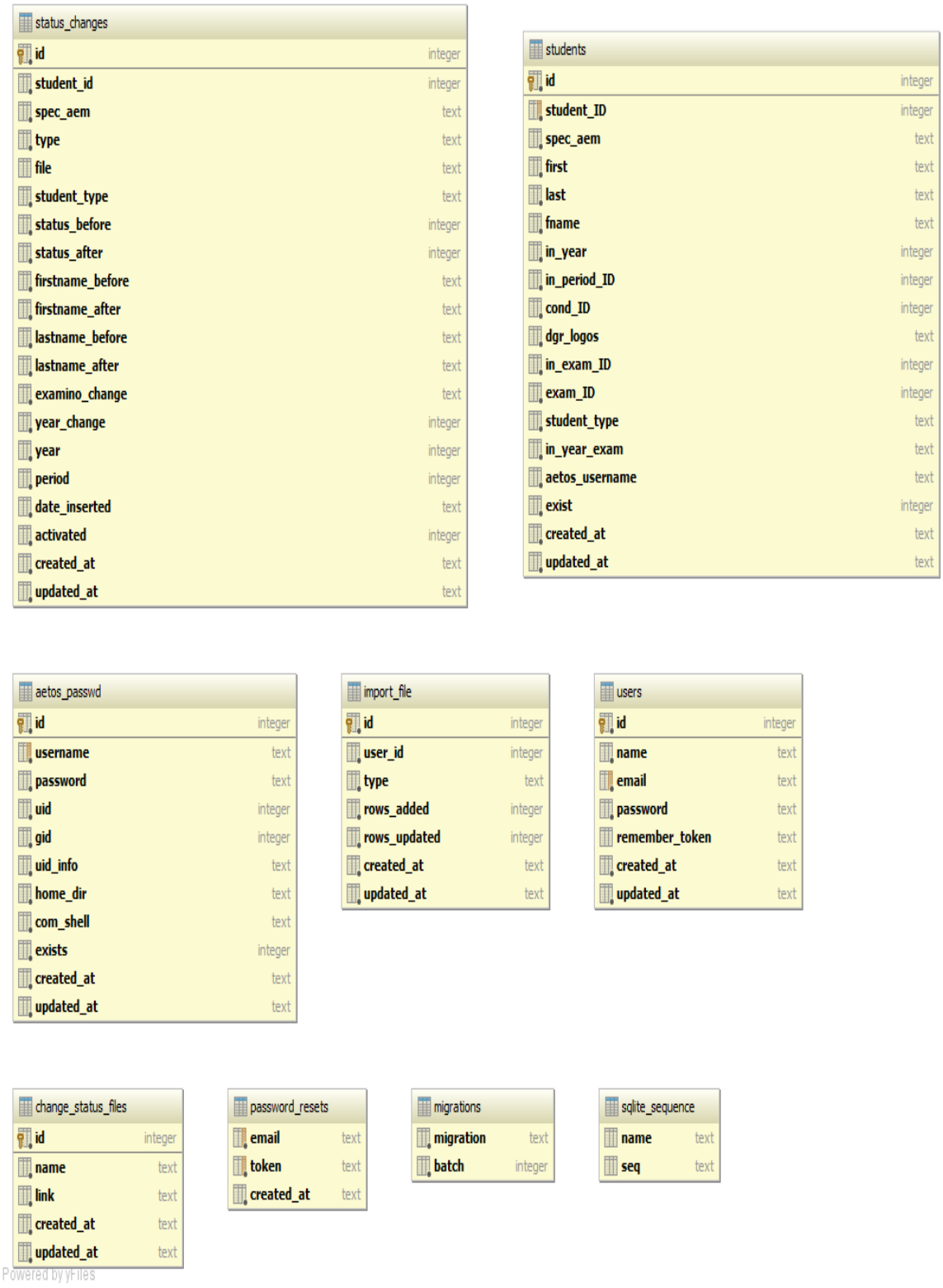
Στο κεφάλαιο που ακολουθεί, θα παρουσιαστούν αναλυτικά οι νέες τεχνικές και διαδικασίες που χρησιμοποιήθηκαν για την κατασκευή του νέου συστήματος.

2.1 ΔΟΜΗ ΑΡΧΕΙΟΥ «STUDENTS.TXT»

Η βασική δομή και ιεραρχία του tsv αρχείου «students.txt» παρέμεινε ίδια και σε αυτή 1η έκδοση του νέου συστήματος. Δεν έχει αλλάξει κάτι ως προς τον αριθμό των πεδίων, και έχουν την ίδια μορφή όπως τα έστελνε το κέντρο διαχείρισης δικτύου(noc) στο τμήμα πληροφορικής(βλέπε πίνακα 1, κεφ. 1.1).

2.2 ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ ΤΟΥ ΝΕΟΥ ΣΥΣΤΗΜΑΤΟΣ

Ο σχεδιασμός της βάσης δεδομένων αποτελεί ένα σημαντικό μέρος της διαδικασίας της σχεδίασης ενός διαδικτυακού συστήματος. Είναι στην ουσία η πηγή άντλησης πληροφοριών του. Για αυτό το λόγο λοιπόν, θα πρέπει ο σχεδιασμός της βάσης να μην την αφήνει σε ασυνεπή κατάσταση, να μπορεί να ανακτά γρήγορα τα δεδομένα καθώς και να τηρεί όλες εκείνες τις λογικές συνδέσεις που προκύπτουν από τους πίνακες της βάσης όσον αφορά τα κύρια, ξένα κλειδιά και τους περιορισμούς που ίσως διαθέτουν.



Εικόνα 1: Πίνακες βάσης δεδομένων του νέου συστήματος

2.3 ΕΝΗΜΕΡΩΣΗ ΒΑΣΗΣ ΦΟΙΤΗΤΩΝ ΜΕΣΩ ΑΡΧΕΙΟΥ «STUDENTS.TXT»

Η διαδικασία της ενημέρωσης αποτελεί ίσως και το πιο σύνθετο κομμάτι καθώς διασπάτε σε αρκετά μικρά μέρη τα οποία γίνονται πλέον με πιο αυτοματοποιημένο τρόπο στο νέο σύστημα. Τα αρχεία των φοιτητών πλέον που έχουν σταλεί από το παρελθόν στο τμήμα της πληροφορικής, έχουν τοποθετηθεί στον αρχικό κατάλογο της εφαρμογής στον φάκελο «storage/uploads/student_seeds/ExportPlhrData YYYY_MM_DD». Ο κάθε τέτοιος φάκελος περιέχει ένα αρχείο «students.txt» και το “YYYY_MM_DD” εκφράζει την ημερομηνία λήψης του από το NOC. Τοποθετώντας λοιπόν όλους αυτούς τους φακέλους, η εφαρμογή έχει την εναλλακτική να ξαναεισάγει τα δεδομένα των φοιτητών στη βάση από τα αρχεία «students.txt» που περιέχονται σε αυτούς, εάν προκύψει κάποιο σφάλμα στον server και χαθούν δεδομένα. Πρόκειται δηλαδή για ένα είδος backup αλλά και σημείο αφετηρίας σε περίπτωση επανεγκατάστασης του συστήματος.

2.4 ΕΠΕΞΕΡΓΑΣΙΑ ΤΟΥ ΑΡΧΕΙΟΥ «STUDENTS.TXT»

Αφού μετατραπούν όλες οι γραμμές του αρχείου σε πίνακα φοιτητών με τα πεδία τους, φιλτράρονται από μια μέθοδο η οποία αναλαμβάνει την επιδιόρθωση της κωδικοποίησης των ελληνικών γραμμάτων του αρχείου, της προσθήκης «παύλας» στα διπλά ονόματα και επίθετα ώστε να μην υπάρχει πρόβλημα αργότερα με τον server aetos. Κατά την διάρκεια της επεξεργασίας του αρχείου, το σύστημα ελέγχει την ύπαρξη ή μη ενός φοιτητή στο σύστημά και σε κάθε περίπτωση αντίστοιχα ενημερώνει τα στοιχεία του με τα νέα δεδομένα του αρχείου ή τον δημιουργεί στη βάση. Ωστόσο, οι φοιτητές των οποίων το ΑΜ(πεδίο «spec_aem» του αρχείου «students.txt») περιλαμβάνει αστεράκι(που σημαίνει ότι ο φοιτητής βρίσκεται σε κατάσταση αναμονής εγγραφής από την γραμματεία ή δεν έχει προσκομίσει τα απαραίτητα δικαιολογητικά ακόμα) ή ξεκινάει με ER/er(που δηλώνει ότι ο φοιτητής είναι από Erasmus), δεν συμμετέχουν στην διαδικασία ενημέρωσης και αποκλείονται όταν βρεθούν. Παράλληλα με την ενημέρωση/δημιουργία του κάθε φοιτητή, το σύστημα ενημερώνει και τον πίνακα “status_changes”(σχήμα 1) με τα απαραίτητα πεδία του . Στην ουσία κρατάει ένα αρχείο ενημέρωσης φοιτητών με

όλες τις ενημερώσεις των φοιτητών που προκύπτουν κατά την επεξεργασία του αρχείου.

2.5 ΕΝΗΜΕΡΩΣΗ ΣΥΣΤΗΜΑΤΟΣ ΜΕ ΤΟ ΑΡΧΕΙΟ «/ETC/PASSWD»

Ένα βασικό στοιχείο που έπρεπε να προστεθεί στο νέο σύστημα ήταν η δυνατότητα να μπορεί να βλέπει το σύστημα άμεσα τις εγγραφές των φοιτητών όπως αυτές φαίνονται στον aetos, πληροφορία που πλέον μπορεί να αποκτηθεί αυτόματα ή χειροκίνητα από το νέο σύστημα με την τροφοδότηση του αρχείου «/etc/passwd». Υπάρχει δηλαδή η δυνατότητα ο διαχειριστής να ενημερώσει την βάση με ένα αρχείο «passwd» που έχει ήδη κατεβασμένο στον τοπικό του δίσκο και θεωρεί ότι είναι ενημερωμένο ή ακόμα να επιλέξει την αυτόματη ενημέρωση του συστήματος. Η βασική πληροφορία που απαιτεί το σύστημα από το αρχείο «passwd» ήταν και είναι αν κατά την διαδικασία προσθήκης ενός νέου φοιτητή υπάρχει ήδη αυτό το username στον aeto. Αυτό επίσης εξαλείφεται με την επιλογή της αυτόματης ενημέρωσης με το αρχείο «passwd» από τον server του aetos, κατά την διάρκεια της ενημέρωσης των φοιτητών.

2.6 ΔΗΜΙΟΥΡΓΙΑ ΑΡΧΕΙΟΥ ΑΛΛΑΓΩΝ

Όταν ολοκληρωθεί η εισαγωγή και η ενημέρωση των φοιτητών και του πίνακα “status_changes”, επόμενη δραστηριότητα για το σύστημα είναι η δημιουργία ενός αρχείου που να καταγράφει αναλυτικά τις αλλαγές, προσθήκες και διαγραφές των στοιχείων των φοιτητών κατά την διάρκεια της ενημέρωσης, ομαδοποιημένες ανά ΑΜ(αριθμό μητρώου). Το αρχείο αυτό μπορεί να αποτελέσει μια ανασκόπηση στις συνέπειες της εκάστοτε ενημέρωσης και μπορεί να κατέβει τοπικά στον σκληρό δίσκο του διαχειριστή με μορφή «changes_YYYY_MM_DD_Τυχαίο_αριθμό(4ψήφιο).txt»

2.7 ΠΡΟΤΕΙΝΟΜΕΝΑ USERNAMES

Η επιτυχής ενημέρωση της βάσης με το ανέβασμα του αρχείου «students.txt» μπορεί να περιέχει και νέες εγγραφές φοιτητών οι οποίοι δεν έχουν αποκτήσει ακόμα username, και τα στοιχεία τους βρίσκονται μόνο στην διάθεση της γραμματείας. Ωστόσο η προαναφερθείσα διαδικασία στο υποκεφάλαιο 2.5 μπορεί να διορθώσει αυτό το πρόβλημα καθώς προηγείται η ενημέρωση των usernames από το αρχείο «/etc/passwd» του aetos server το οποίο λαμβάνεται αυτόματα μέσω ssh σύνδεση στον server aetos. Όσον αφορά τον τρόπο δημιουργίας των usernames, το σύστημα δημιουργεί το προτεινόμενο username από το πρώτο γράμμα του ονόματος και τα υπόλοιπα του επιθέτου. Στις περιπτώσεις όπου υπάρχει το ίδιο username στην λίστα των νέων φοιτητών, τώρα αυτό το πρόβλημα επιλύεται καθώς το σύστημα ελέγχει και στα ήδη υπάρχοντα usernames που υπάρχουν στον aeto(αν έγινε αυτόματη ενημέρωση αλλιώς από το τελευταίο πρόσφατο αρχείο «passwd» που έχει κρατήσει το σύστημα) αλλά και μεταξύ των προτεινόμενων usernames. Για να μην υπάρχουν ίδια usernames, το σύστημα αντικαθιστά τον τελευταίο χαρακτήρα του username με έναν τυχαίο αριθμό, και αν πάλι υπάρχει ίδια εγγραφή αντικαθιστά το προτελευταίο κ.ο.κ έως ότου αυτό είναι μοναδικό.

2.8 ΤΡΟΠΟΣ ΠΑΡΟΥΣΙΑΣΗΣ ΤΩΝ ΦΟΙΤΗΤΩΝ

Στο νέο σύστημα, ο τρόπος με τον οποίο εμφανίζονται οι φοιτητές είναι λίγο διαφορετικός, καθώς πλέον εμφανίζονται από προεπιλογή χωρίς τη χρήση φίλτρων, οι προπτυχιακοί και ενεργοί φοιτητές, με εξαίρεση αυτών που ανήκουν σε προγράμματα Erasmus. Υπάρχει η δυνατότητα φιλτραρίσματος ως προς το τρέχον εξάμηνο πλέον και όχι ως προς το εξάμηνο εισαγωγής που προϋπήρχε στο σύστημα της hydras(εξακολουθεί να υπάρχει και σ ’αυτό το σύστημα). Εκτός από αυτά τα φίλτρα υπάρχει και αυτό της κατάστασης των φοιτητών που εμφανίζει τους φοιτητές σύμφωνα με την κατάσταση τους στο σύστημα(π.χ. ενεργοί, διαγραμμένοι κλπ.).

2.9 ΕΞΑΓΩΓΗ ΣΤΟΙΧΕΙΩΝ ΦΟΙΤΗΤΩΝ ΣΕ ΑΡΧΕΙΟ ΤΥΠΟΥ “CSV”

Στην σελίδα της εμφάνισης των φοιτητών, με ή χωρίς χρήση φίλτρων, έχει προστεθεί και η δυνατότητα εξαγωγής των στοιχείων των φοιτητών που βρέθηκαν βάση των φίλτρων(αν έχουν εφαρμοστεί). Το αποτέλεσμα είναι να μπορεί να κατεβάσει ο διαχειριστής του συστήματος το εν λόγω αρχείο με τα στοιχεία των φοιτητών, και έπειτα να τα εισάγει στο excel ή κάποιο αντίστοιχο πρόγραμμα της επιλογής του, προκειμένου να τα χρησιμοποιήσει με όποιο τρόπο αυτός θέλει(στατιστικούς λόγους κλπ.).

2.10 ΣΤΑΤΙΣΤΙΚΑ ΦΟΙΤΗΤΩΝ

Ως τώρα η συλλογή των πληροφοριών από διάφορα αρχεία και η ένωση τους γινόταν από τον διαχειριστή του συστήματος με αρκετά περίπλοκο τρόπο. Στο νέο σύστημα αυτό ωστόσο γίνεται αρκετά απλά χωρίς να χρειαστεί κάποια επεξεργασία ή ανάλυση καθώς το σύστημα προετοιμάζει από μόνο του τα δεδομένα και τα παρουσιάζει σε γραφήματα(ραβδογράμματα) για την εξαγωγή χρήσιμων και πολύτιμων στατιστικών για το τμήμα. Συγκεκριμένα υπάρχουν 4 ειδών γραφήματα. Στο 1ο συγκεντρώνονται αναλυτικά για όλα τα έτη ύπαρξης του τμήματος, οι εισακτέοι ανά έτος εισαγωγής, οι διαγραμμένοι, οι απόφοιτοι και οι φοιτητές που έχουν αναστείλει τις σπουδές τους. Στα υπόλοιπα 3 εκφράζονται οι ποσότητες των αποφοίτων, σε αναστολή και διαγραμμένων φοιτητών ανά εξάμηνο σπουδών. Το μεγάλο πλεονέκτημα αυτών των γραφημάτων είναι ότι μπορούν να εξαχθούν σε διάφορες μορφές όπως εικόνες, διανύσματα αλλά και csv. Ποιο σημαντικό είναι ότι αν επιλέξει τα csv αρχεία, μπορούν να εισαχθούν σε κάποιο πρόγραμμα όπως το excel, και να παρουσιάσουν αναλυτικά όλες τις πληροφορίες που περιέχουν τα γραφήματα.

ΕΠΙΛΟΓΟΣ

Στο κεφάλαιο αυτό παρουσιάστηκαν εκτενέστερα οι τρόποι, οι διαδικασία και τα νέα χαρακτηριστικά του νέου συστήματος, τα οποία απέδωσαν ένα νέο χαρακτήρα γενικότερα στην εφαρμογή που θα βοηθήσει καλύτερα στην εύρυθμη λειτουργία του από τον διαχειριστή του τμήματος. Όλα τα παραπάνω δεν θα ήταν δυνατόν να υλοποιηθούν χωρίς την χρήση προγραμματιστικών εργαλείων και βοηθημάτων που περιγράφονται αναλυτικά στο επόμενο κεφάλαιο αυτής της πτυχιακής.

ΚΕΦΑΛΑΙΟ 3

ΤΕΧΝΟΛΟΓΊΕΣ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ

ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο, θα αναλυθούν τα εργαλεία που χρησιμοποιήθηκαν στην εφαρμογή, οι λόγοι που προτιμήθηκαν τα συγκεκριμένα, καθώς και μια μικρή περιγραφή για την ιστορία του καθενός.

3.1 LARAVEL FRAMEWORK

Εικόνα 2: Laravel logo

Το σύστημα απαιτούσε λύσεις που χρειαζόταν να δημιουργηθούν από το μηδέν γι' αυτό δεν προτιμήθηκε κάποια έτοιμη λύση CMS για την κατασκευή του συστήματος. Το Laravel Framework, είναι ένα μοντέρνο PHP framework το οποίο μπορεί να χρησιμοποιηθεί για την κατασκευή διαδικτυακών εφαρμογών και μετράει χιλιάδες προγραμματιστές στην κοινότητά του. Η αρχιτεκτονική του Laravel βασίζεται στο MVC σχεδιαστικό πρότυπο, ενώ αυτό που το κάνει να ξεχωρίζει από τα υπόλοιπα framework είναι το απλό και διαισθητικό συντακτικό, το οποίο μας επιτρέπει με λίγες γραμμές κώδικα να πετύχουμε πολλά. Σύμφωνα με τον δημιουργό του, Taylor Otwell, το Laravel θέλει να κάνει το προγραμματισμό διασκεδαστικό και για αυτό δημιουργήθηκε ώστε να είναι απλό, κομψό και κυρίως καλά τεκμηριωμένο.

- **Απλό** – Οι λειτουργίες του Laravel είναι απλές στην κατανόηση και στην υλοποίηση. Αν κάποιος έχει δουλέψει στο παρελθόν με το CodeIgniter ή κάποιο άλλο MVC framework τότε η μετάβαση στο Laravel θα είναι εύκολη.

- **Κομψό** - Το Laravel βασίζεται στα τρέχοντα standard της βιομηχανίας και σαν αποτέλεσμα δεν απαιτούνται περίπλοκοι κώδικες για να εκτελεστούν λειτουργίες που κανονικά είναι απλές. Σε γενικές γραμμές το Laravel χρειάζεται ελάχιστη παραμετροποίηση για να ρυθμιστεί και να δουλέψει.
- **Τεκμηρίωση** – Η τεκμηρίωση του Laravel είναι πλήρης και πάντα ενημερωμένη. Εξίσου σημαντικό για κάποιον που θέλει να μάθει Laravel είναι ότι υπάρχουν πάρα πολλές πηγές με οδηγούς και συμβουλές στο internet.

Το σύστημα χρησιμοποιεί την τελευταία έκδοση 5 του Laravel και έκδοση PHP 5.6.

3.2 COMPOSER

Εικόνα 3: Composer logo

Ο Composer είναι μία εφαρμογή η οποία αναλαμβάνει να κατεβάσει και να ενημερώσει τα διάφορα πακέτα PHP, που έχουμε δηλώσει ότι χρησιμοποιούμε στην εφαρμογή μας. Το ίδιο το Laravel είναι ένα PHP πακέτο το οποίο στηρίζεται με τη σειρά του σε άλλα πακέτα. Ο Composer βλέπει όλες αυτές τις συσχετίσεις (dependencies) και αναλαμβάνει να κατεβάσει τα απαραίτητα πακέτα/βιβλιοθήκες. Γενικά δημιουργείται ένα αρχείο package.json το οποίο τοποθετείται στο root της εφαρμογής και δέχεται διάφορες παραμέτρους. Μια από αυτές είναι και τα πακέτα που πρόκειται να χρησιμοποιηθούν στο σύστημα μαζί με την έκδοση τους ή ακόμα και το repository τους. Για τις ανάγκες τις εφαρμογής χρησιμοποιήθηκαν τα εξής πακέτα:

Πίνακας 2: Πακέτα που χρησιμοποιήθηκαν από τον composer

Πακέτο	Έκδοση	Περιγραφή
“Laravel/framework”	5.0	Το βασικό σύστημα του Laravel
“league/csv”	~7.0	Πακέτο για την εισαγωγή και επεξεργασία csv αρχείων
“barryvdh/Laravel-snappy”	0.2.x	Πακέτο για την παραγωγή pdf αρχείων που χρησιμοποιήθηκαν στην εκτύπωση των στοιχείων των φοιτητών.

3.3 NODE PACKAGE MANAGER

Εικόνα 4: Node.js logo

Το node package manager(npm), είναι ένας διαχειριστής πακέτων μέσω του οποίου μπορεί κάποιος να κατεβάσει ένα πακέτο ή module τα οποία μπορεί να χρησιμοποιήσει στα πλαίσια της εφαρμογής του. Στο σύστημα το npm χρησιμοποιήθηκε για την απόκτηση του πακέτου gulp που περιγράφεται στην συνέχεια.

3.4 GULP.JS

Εικόνα 5: Gulp.js logo

Το gulp.js χρησιμοποιεί το node.js για την λειτουργία του. Δημιουργεί ένα ειδικό αρχείο γνωστό και ως gulpfile που τοποθετείται στο root της εφαρμογής. Σ ’αυτό το αρχείο σημειώνονται διάφορες διεργασίες όπως η αυτόματη παρακολούθηση των αλλαγών ενός αρχείου .less ώστε σε κάθε αλλαγή του να γίνεται αυτόματα το compile στον κώδικα χωρίς να υπάρχουν καθυστερήσεις και κενά στον τελικό χρήστη. Αυτές οι διεργασίες του gulp αποτελούν server-side λύσεις και δεν εκτελούνται στον υπολογιστή του τελικού χρήστη. Όσον αφορά την συγκεκριμένη εφαρμογή έχει εγκατασταθεί μια πιο «ενοποιημένη» και απλή λύση του gulp καθώς συνεργάζεται καλύτερα με το Laravel Framework και είναι το Laravel-elixir. Το Laravel-elixir περιέχει πολλά προεγκατεστειμένα modules για το gulp και είναι πιο εύκολο στην χρήση του από το να γινόταν ξεχωριστά η εγκατάστασης του κάθε module. Φυσικά έχει προστεθεί αυτόματα και η δυνατότητα autoprefixer ώστε να είναι συμβατό το τελικό css με οποιοδήποτε browser σε οποιαδήποτε έκδοσή του.

3.5 ΑΡΧΕΙΑ LESS

Εικόνα 6: Less logo

Η χρήση css αρχείων δεν είναι πάντα και η πιο βολική ιδίως όταν πρόκειται για μια εφαρμογή στην οποία χρειάζεται διαφορετικές μεταβλητές για την παρουσίαση των σελίδων. Το less αποτελεί ένα προεπεξεργαστή της css, δηλαδή μια προέκταση της

css γλώσσας δίνοντας την ευκαιρία να χρησιμοποιηθούν mixins, μεταβλητές, functions κλπ. γεγονός που δίνει την δυνατότητα στο less να είναι πιο διατηρήσιμο, επεκτάσιμο και με περισσότερες επιλογές. Το less τρέχει πάνω στο Node, είναι κι αυτό server side module δηλαδή, και σε συνδυασμό με το προηγούμενο εργαλείο(gulp.js) μπορεί να γίνει χειροκίνητα από τον developer, compile μέσω του gulp-less(module του gulp) ή ακόμα και αυτόματα με την προσθήκη του (gulp-watch) το οποίο παρακολουθεί τα αρχεία Less και όταν αλλάξει κάτι ξανακάνει από μόνο του το compile του αρχείου σε css.

3.6 BOOTSTRAP

Εικόνα 7: Bootstrap logo

Το πακέτα “Laravel/framework” έγινε διάσημο από τη συμμετοχή του στην ανάπτυξη του γνωστού ιστότοπου Twitter. Στην ανάπτυξη του επίσης βοήθησε και ένα άλλο εργαλείο το οποίο χρησιμοποιήθηκε και στην παρούσα εφαρμογή και ονομάζεται Bootstrap. Το bootstrap αποτελεί ένα εργαλείο για την παραγωγή responsive, mobile-first εφαρμογών. Το Bootstrap φυσικά δεν είναι απαραίτητο για να δουλέψει το Laravel, όμως χρησιμοποιείται για να γίνει πιο όμορφο και λειτουργικό το περιβάλλον της εφαρμογής. Είναι open source και για αυτό το λόγο χρησιμοποιήθηκε η less και όχι η minified έκδοσή του. Έτσι κάθε αλλαγή που γίνεται στις μεταβλητές τους γίνεται αυτόματα compile και στην συνέχεια minified css ώστε να μπορεί να το διαβάσει ο browser.

3.7 JQUERY

Εικόνα 8: jQuery logo

Όσον αφορά την «κίνηση» και τα εφέ της εφαρμογής, χρησιμοποιήθηκε ή πασίγνωστη βιβλιοθήκη της javascript, το jQuery. Το jQuery είναι ένα μικρό, γρήγορο και πλούσιο σε χαρακτηριστικά javascript framework. Η χρήση του έγινε κυρίως στα ajax request που απαιτήθηκαν σε διάφορα σημεία της εφαρμογής, στον τρόπο εμφάνισης και παρουσίασης των modals και κάποιων παραθύρων της εφαρμογής αλλά ακόμα και για την χρήση του στον διαχειρισμό των κλάσεων, divs και άλλων στοιχείων των σελίδων.

3.8 HTML5

Εικόνα 9: HTML5 logo

Η παρουσίαση των αποτελεσμάτων και γενικά όλου του υλικού, έγινε στην Hypertext Markup Language ή αλλιώς HTML. Η HTML5 αντικατάστησε την HTML 4.01, την XHTML 1.0, και την DOM Level 2 HTML. Ο σκοπός της είναι η μείωση της ανάγκης για ιδιόκτητα plug-in και πλούσιες διαδικτυακές εφαρμογές (RIA) όπως το Adobe Flash, το Microsoft Silverlight, το Apache Pivot, και η Sun JavaFX. Για την παραγωγή κάθε σελίδας της εφαρμογής, χρησιμοποιήθηκε το σύστημα παραγωγής της Laravel που ονομάζεται Blade Engine. Δημιουργήθηκε δηλαδή ένα master layout το οποίο η σελίδα που θα δημιουργηθεί το κάνει επέκταση, και έπειτα εμφανίζει τα δικά της δεδομένα. Αποτελεί δηλαδή την βάση (ή κάτι σαν θέμα) της εμφάνισης των σελίδων.

ΣΥΜΠΕΡΆΣΜΑΤΑ

Η κατασκευή ενός συστήματος το οποίο θα χρησιμοποιείται σε κάποιο τμήμα για την ικανοποίηση των αναγκών του σχετικά με τους φοιτητές του, ήταν και θα συνεχίζει να είναι ένα σημαντικό εργαλείο για το άτομο που το διαχειρίζεται αλλά πάνω από όλα για το ίδιο το τμήμα.

Ο αρχικός σχεδιασμός θα πρέπει να περιλαμβάνει πολλές περιπτώσεις χρήσης όσον αφορά τα σενάρια που μπορεί να προκύψουν σε ένα τμήμα μιας σχολής και να τα εντάσσει από την αρχή στο σχεδιασμό. Τα μεγαλύτερα προβλήματα σε μεγάλες εφαρμογές οι οποίες τείνουν να υστερούν τεχνολογικά σε καινοτόμα και νέα εργαλεία, είναι αυτός ακριβώς ο ίδιος λόγος που δεν τις αφήνει να υποστούν αλλαγές από ένα μικρό κομμάτι κώδικα, ως ακόμα και ριζικές. Έτσι προκύπτει εύλογα το συμπέρασμα ότι μια εφαρμογή θα πρέπει να είναι ευέλικτη όσον το δυνατό περισσότερο, ώστε να μπορεί να υποστηρίξει μελλοντικά νέες τεχνολογίες, νέα δεδομένα και πληροφορίες και πάνω από όλα να παραμένει λειτουργική, εύχρηστη και ασφαλής!

ΒΙΒΛΙΟΓΡΑΦΙΑ

Ο τρόπος γραφής των βιβλιογραφικών αναφορών γίνεται σύμφωνα με το APA Formatting and Style Guide2.

- Laravel – Wikipedia, the free encyclopedia. Από *Wikipedia*. 19 Μαΐου 2015, από http://en.wikipedia.org/wiki/Laravel
- Εισαγωγή στο Laravel – Μέρος 1ο. 19 Μαΐου 2015, από http://www.wdf.gr/articles/php-sql/%CE%B5%CE%B9%CF%83%CE%B1%CE%B3%CF%89%CE%B3%CE%AE-%CF%83%CF%84%CE%BF-laravel-%CE%BC%CE%AD%CF%81%CE%BF%CF%82-1%CE%BF.html
- An introduction to Gulp.js – SitePoint, In SitePoint,19 Μαΐου 2015, από http://www.sitepoint.com/introduction-gulp-js/
- HTML5 – Βικιπεδία, Από Wikipedia, 19 Μαΐου 2015, από http://el.wikipedia.org/wiki/HTML5
- Gulp.js – the streaming build system, Από gulpjs.com, 19 Μαΐου 2015, από http://gulpjs.com/
- Getting started | Less.js , Από Lesscss.org,19 Μαΐου 2015, από http://lesscss.org/

ΠΑΡΑΡΤΗΜΑΤΑ

ΠΑΡΑΡΤΗΜΑ 1

```
/**
 * Μετατρέπει το αρχείο students.txt σε πίνακα
 *
 * @param \Symfony\Component\HttpFoundation\File\UploadedFile $file
 * @return array
 * @throws \Exception
 */
private function toArray($file)
{
    try {

        $students=array();
        $interpreter = new Interpreter();
        $interpreter->addObserver(function (array $row) use (&$students) {
            $students[] = array(
                'student_ID' => $row[0],
                'spec_aem' => $row[1],
                'first' => $row[2],
                'last' => $row[3],
                'fname' => $row[4],
                'in_year' => $row[5],
                'in_period_ID' => $row[6],
                'cond_ID' => $row[7],
                'dgr_logos' => $row[8],
                'in_exam_ID' => $row[9],
                'exam_ID' => $row[10]
            );
        });
```

```
        $this->lexer->parse($file->getPathname(), $interpreter);
        array_shift($students);
        return $students;
    } catch (\Exception $e) {
        throw new \Exception("File type is not in students.txt format");
    }
}


/**
 * Προετοιμάζει και κανονικοποιεί τα δεδομένα του φοιτητή πριν την ενημέρωση του
πίνακα
 *
 * @param $student
 * @param boolean|null $no_file_update
 * @return mixed
 */
public function normalizeStudentData($student,$no_file_update=null)
{
    $student['student_ID'] = intval($student['student_ID']);
    $year=intval($student['in_year']) > 0 ? intval($student['in_year']) : $this-
>file_options['year_change'];
    $student['in_year'] = $year;


    $student['first'] =$this->fix_double_name($student['first'],$no_file_update);
    $student['last'] = $this->fix_double_name($student['last'],$no_file_update);
    $student['fname'] = $this-
>fix_double_name($student['fname'],$no_file_update);


    $period= intval($student['in_period_ID']) > 0 ? intval($student['in_period_ID']) :
$this->file_options['period'];
    $student['in_period_ID'] = $period;
```

```
    $student['cond_ID'] = intval($student['cond_ID']);
    $student['dgr_logos'] = strtoupper(iconv('greek', 'UTF-8',
$student['dgr_logos']));
    $student['in_exam_ID'] = filter_var($student['in_exam_ID'],
FILTER_VALIDATE_INT) ? intval($student['in_exam_ID']) : 0;
    $student['exam_ID'] = intval($student['exam_ID']);

    if(!$no_file_update){
        $student['student_type']= $this->isMsc($student['spec_aem'])? 'msc' : 'bsc'
;
    }

    $student['spec_aem'] = $this->fix_spec_aem($student['spec_aem'],$year);

    $in_year_exam = $this->createExam($year, $period);
    $student['in_year_exam'] = $in_year_exam;

    return $student;
}


/**
 * Ελέγχει αν είναι διπλό όνομα(αν υπάρχει κενό), και προσθέτει ένα σύμβολο για
να τα διαχωρίσει
 *
 * @param string $name
 * @param $no_file_update
 * @param string $symbol
 * @return string $name
 */
private function fix_double_name($name,$no_file_update,$symbol="-")
{
    $words=explode(" ",$name);
```

```
    foreach($words as $k=>$v){

        if(is_null($no_file_update)){
            $words[$k]=ucfirst(iconv('greek', 'UTF-8',$words[$k]));
        }
        else{
            $words[$k]=mb_convert_case($words[$k],MB_CASE_UPPER, "UTF-8")
        }
    }

    if(count($words)>1){
        foreach($words as $k=>$v){
            $words[$k]=str_replace(["-"," "],"",$words[$k]);
            if(preg_match('/-/i',$words[$k] && strlen($words[$k])<2) || $words[$k]=="")
unset($words[$k]);
        }

        return(implode($symbol,$words));
    }

    return $words[0];
}
```

```
/**
 * Συμπληρώνει με 0 το spec_aem των φοιτητών κατά την δημιουργία τους, μέχρι να αποκτήσει μήκος 6 χαρακτήρων
 *
 * @param $spec_aem
 * @return string mixed
 */
private function fix_spec_aem($spec_aem,$in_year)
{
    if(substr($spec_aem, -1) =="M") return substr($spec_aem, 0, -1);


    $spec_aem=strval($spec_aem);
    if(in_array($in_year,range(2000,2009))){
        while(strlen($spec_aem)<6){
            $spec_aem="0".$spec_aem;
        }
    }
    return strval($spec_aem);
}



/**
 * Δημιουργεί το εξάμηνο εισαγωγής του φοιτητή βάση του έτους εισαγωγής και της περιόδου
 *
 * @param integer $year
 * @param  integer $period
 * @return int|string
 */
private function createExam($year,$period)
{
    $final_exam=0;
```

```
    $year = intval($year);
    $date = \DateTime::createFromFormat("Y", $year);
    $fyear = $date->format('y');
    $date = \DateTime::createFromFormat("Y", $year + 1);
    $lyear = $date->format('y');
    if (intval($period) == 1) {
        $final_exam = "X" . strval($fyear) . strval($lyear);
    } elseif (intval($period) == 2) {
        $final_exam = "E" . strval($fyear) . strval($lyear);
    }
    return $final_exam;
}



/**
 * Ελέγχει αν οι δυο φοιτητές έχουν τα ίδια στοιχεία
 *
 * @param $student1 φοιτητής από αρχείο
 * @param Student $student2 υπάρχων φοιτητής
 * @return boolean
 */
public function areSame($student1,Student $student2)
{
    if($student1['first']!=$student2->first) return false;
    if($student1['last']!=$student2->last) return false;
    if($student1['cond_ID']!=$student2->cond_ID) return false;

    return true;
}
```

```
/**
 * Ελέγχει εάν ο φοιτητής της εγγραφής πρέπει να αγνοηθεί από την εισαγωγή στην βάση(είναι Erasmus ή σε αναμονή εγγραφής)
 *
 * @param array $student_data
 * @return bool
 */
private function isIgnored($student_data)
{

    if(preg_match('/\*/',$student_data['spec_aem']) ||
        preg_match('/^er/i',$student_data['spec_aem']) ||
        $student_data['spec_aem']=='000001' ||
        $student_data['spec_aem']=='113697' && $this->file_options['date_inserted']=='2011_10_13'
    ) return true;

    return false;



/**
 * Ενημέρωση πίνακα φοιτητών της βάσης
 *
 * @param array $students
 * @return mixed
 * @throws StudentException
 */
private function import($students)
{
    DB::disableQueryLog();

    $changes=DB::transaction(function () use ($students) {
```

```
        if(DB::select("SELECT * FROM status_changes WHERE activated=0;")){
            DB::table('status_changes')->where('activated',0)-
>update(['activated'=>1]);
            Student::where('exist',0)->update(['exist'=>1]);
        }

        $changes=[
            'rows_added'=>0,
            'rows_updated'=>0
        ];

        $studentRepo=new StudentsRepository();

        foreach ($students as $student) {

            $spec_aem=$student['spec_aem'];

            $student = $this->normalizeStudentData($student);

            $student_exist = Student::where('student_ID', '=',
intval($student['student_ID']))->first();

            if(!$this->isIgnored($student) && strlen(strval($spec_aem))>4){

                if (!$student_exist) {

                    $this->updateStatusChangesTable($student);
                    $new_student=Student::create($student);

                    if($this->file_options['real_update_mode']) $studentRepo-
>setUsername($new_student);
                    $changes['rows_added']+=1;
                } else {
```

```
                if ($this->areSame($student,$student_exist)) {
                    $student_exist->update(['exam_ID'=>$student['exam_ID']]);
                }
                else{
                    $this->updateStatusChangesTable($student,$student_exist);
                    $student_exist->update($student);
                    $changes['rows_updated']+=1;
                }
            }
        }
    }
    $this->update_imports($changes);
    return $changes;
});

return $changes;
}



/**
 * Εισάγει μια νέα εγγραφή στον πίνακα status changes για τον συγκεκριμένο
φοιτητή
 *
 * @param $student
 * @param Student $student_exist
 */
public function updateStatusChangesTable($student,Student
$student_exist=null)
{
    $date=Carbon::now()->toDateTimeString();

    DB::disableQueryLog();
```

```
    DB::transaction(function () use ($student,$student_exist,$date) {
        DB::table('status_changes')->insert([
            'student_id' => intval($student['student_ID']),
            'status_before' => is_null($student_exist)? $student['cond_ID']:
$student_exist->cond_ID,
            'status_after' => $student['cond_ID'],
            'student_type' => $student['student_type'],
            'firstname_before' => is_null($student_exist)? $student['first'] :
$student_exist->first,
            'firstname_after' => $student['first'],
            'lastname_before' => is_null($student_exist)? $student['last'] :
$student_exist->last,
            'lastname_after' => $student['last'],
            'type' => is_null($student_exist) ? 'created' : 'updated',
            'year' => $student['in_year'],
            'examino_change'=>$this->file_options['examino_change'],
            'year_change'=>$this->file_options['year_change'],
            'period' => $student['in_period_ID'],
            'activated'=>0,///(/*$this->file_options['sync_aetos'] &&
*/is_null($student_exist))? 0 : 1,
            'date_inserted'=>$this->file_options['date_inserted'],
            'spec_aem'=> is_null($student_exist)? $student['spec_aem'] :
$student_exist->spec_aem,
            'file'=>$this->file_options['file'],
            'created_at' => is_null($student_exist) ? $date : $student_exist -
>created_at,
            'updated_at' => $date
        ]);
    });

}
```

```php
/**
 * Δημιουργεί το αρχείο που περιέχει τις αλλαγές που έγιναν
 *
 * @return string
 */
private function createDownloadChanges_file()
{
    $link=URL::to('files/changes/'.$this->file_options['file']);
    DB::disableQueryLog();
    DB::transaction(function () use ($link) {
        DB::table('change_status_files')->insert([
            'name' => $this->file_options['file'],
            'link' => $link,
            'created_at' => $this->file_options['date'],
            'updated_at' => $this->file_options['date']
        ]);
    });

    $changes=$this->generateStudentsChangedData();

    $content = $this->build_changes_file($changes);

    $this->activateAll();

    File::put(public_path().'/files/changes/'.$this->file_options['file'], $content);

    return URL::to('files/changes/' . $this->file_options['file']);
}
```

```
/**
 * Δημιουργεί τα περιεχόμενα του αρχείου αλλαγών βάση των δεδομένων του πίνακα status changes
 *
 * @param array $pinakas
 * @return string $str
 */
private function build_changes_file($pinakas)
{
    $date=Carbon::now()->toDateTimeString();
    $file_date=Carbon::createFromFormat('Y_m_d',$this->file_options['file_date'])->format('Y-m-d H:i:s');

    $str = "**********************************************************************\n";
    $str .= "               Αρχείο αλλαγών " . $file_date . "\n\n";
    $str .= "                        Συνολικά: \n";
    $str .= "               ---------------------------------\n";
    $str .= "                        ADDED: " . (array_key_exists('rows_added', $pinakas) ? count($pinakas['rows_added']) : 0) . "\n";
    $str .= "                        UPDATED: " . (array_key_exists('rows_updated', $pinakas) ? count($pinakas['rows_updated'], COUNT_RECURSIVE) - count($pinakas['rows_updated'], 0) : 0) . "\n";
    $str .= "                        DELETED: " . (array_key_exists('rows_deleted', $pinakas) ? count($pinakas['rows_deleted']) : 0) . "\n";
    $str .= "**********************************************************************\n\n\n";

    $str .= "================================================================\n";
    $added_count = array_key_exists('rows_added', $pinakas) ? count($pinakas['rows_added']) : 0;
    $str .= 'Added: ' . $added_count . "\n";
    $str .= "================================================================
```

```
=========\n\n";
    if (array_key_exists('rows_added', $pinakas)) {
        foreach ($pinakas['rows_added'] as $added) {
            $str .= "    " . $added . "\n";
        }
    }

    $str .=
"\n\n============================================================
============\n";
    $updated_count = array_key_exists('rows_updated', $pinakas) ?
count($pinakas['rows_updated'], COUNT_RECURSIVE) -
count($pinakas['rows_updated'], 0) : 0;
    $str .= "Updated: " . $updated_count . "\n";
    $str .=
"============================================================
=========\n";
    if (array_key_exists('rows_updated', $pinakas)) {
        foreach ($pinakas['rows_updated'] as $type => $updated) {

            if ($type == "activated" &&
count($pinakas['rows_updated']["activated"])) {
                $str .= "\n   ~Επαναενεργοποίησαν: " .
count($pinakas['rows_updated']["activated"]) . "\n\n";
                foreach ($pinakas['rows_updated']["activated"] as $activated) {
                    $str .= "    " . $activated . "\n";
                }
            } elseif ($type == "grd" && count($pinakas['rows_updated']["grd"])) {
                $str .= "\n   ~Αποφοίτησαν: " .
count($pinakas['rows_updated']["grd"]) . "\n\n";
                foreach ($pinakas['rows_updated']["grd"] as $grd) {
                    $str .= "    " . $grd . "\n";
                }
            } elseif ($type == "halted" &&
```

```
count($pinakas['rows_updated']["halted"])) {
            $str .= "\n   ~Σε Αναστολή: " .
count($pinakas['rows_updated']["halted"]) . "\n\n";
            foreach ($pinakas['rows_updated']["halted"] as $halted) {
               $str .= "    " . $halted . "\n";
            }
         } elseif ($type == "moved" &&
count($pinakas['rows_updated']["moved"])) {
            $str .= "\n   ~Μετεγγράφησαν: " .
count($pinakas['rows_updated']["moved"]) . "\n\n";
            foreach ($pinakas['rows_updated']["moved"] as $moved) {
               $str .= "    " . $moved . "\n";
            }
         } elseif ($type == "togrd" && count($pinakas['rows_updated']["togrd"])) {
            $str .= "\n   ~Ανακυρηχθείς πτυχιούχοι: " .
count($pinakas['rows_updated']["togrd"]) . "\n\n";
            foreach ($pinakas['rows_updated']["togrd"] as $togrd) {
               $str .= "    " . $togrd . "\n";
            }
         } /*elseif ($type == "waitregister" &&
count($pinakas['rows_updated']["waitregister"])) {
            $str .= "\n   ~Εν αναμονή εγγραφής: " .
count($pinakas['rows_updated']["waitregister"]) . "\n\n";
            foreach ($pinakas['rows_updated']["waitregister"] as $waitregister) {
               $str .= "    " . $waitregister . "\n";
            }
         }*/ elseif ($type == "first" && count($pinakas['rows_updated']["first"])) {
            $str .= "\n   ~Αλλαγή ονόματος: " .
count($pinakas['rows_updated']["first"]) . "\n\n";
            foreach ($pinakas['rows_updated']["first"] as $firstname) {
               $str .= "    " . $firstname . "\n";
            }
         } elseif ($type == "last" && count($pinakas['rows_updated']["last"])) {
            $str .= "\n   ~Αλλαγή επιθέτου: " .
```

```
count($pinakas['rows_updated']["last"]) . "\n\n";
                foreach ($pinakas['rows_updated']["last"] as $lastname) {
                    $str .= "    " . $lastname . "\n";
                }
            }
        }
    }

    $str .=
"\n\n============================================================
============";
    $deleted_count = array_key_exists('rows_deleted', $pinakas) ?
count($pinakas['rows_deleted']) : 0;
    $str .= "\nDeleted: " . $deleted_count . "\n";
    $str .=
"============================================================
=========\n\n";
    if (array_key_exists('rows_deleted', $pinakas)) {
        foreach ($pinakas['rows_deleted'] as $removed) {
            $str .= "    " . $removed . "\n";
        }
    }

    return $str;
}
```

```
/**
 * Επιστρέφει έναν πίνακα με τις εγγραφές που αλλάχθηκαν/δημιουργήθηκαν/διαγράφθηκαν.
 *
 * @param string|null $file Όνομα αρχείου που έγινε η αλλαγή
 * @return array $total
 */
private function generateStudentsChangedData($file=null){

    DB::disableQueryLog();
    $changes=DB::select("SELECT * FROM status_changes WHERE activated=0 ORDER BY spec_aem;");

    $total=DB::transaction(function() use ($changes,$file) {
        $total=[];
        foreach($changes as $change) {

            $change=(array)$change;
            $data = Student::where('student_id',$change['student_id'])->first()->toArray();


            if( ($change['status_before'] ==1 && $change['status_after']==1) /*|| ($change['status_before']=$change['status_after']) */){
                $total['rows_added'][] =
                    $data['spec_aem'] . ": " .
                    $data['first'] . " " .
                    $data['last'] . " " .
                    $data['in_year_exam'] . " " .
                    $data['exam_ID'];
            }
            else{
                if ( $change['status_after'] == 1 && $change['status_before'] != 1) {
                    $total['rows_updated']["activated"][] =
```

```
                $data['spec_aem'].": (status: ".$change['status_before']." =>
".$change['status_after']. ") "
                .$data['last']." ".$data['first']." ".$data['fname'];
            }
            if ( $change['status_after'] == 2  && $change['status_before'] != 2) {
                $total["rows_deleted"][]=
                $data['spec_aem']."  ".$data['last']." ".$data['first']."
".$data['fname'];
            }
            if ( $change['status_after'] == 3 && $change['status_before']!=3) {
                $total['rows_updated']["grd"][]=
                $data['spec_aem'].": (status: ".$change['status_before']." =>
".$change['status_after']. ") "
                .$data['last']." ".$data['first']." ".$data['fname'];
            }
            if ( $change['status_after'] == 4 && $change['status_before'] != 4) {
                $total['rows_updated']["halted"][]=
                $data['spec_aem'].": (status: ".$change['status_before']." =>
".$change['status_after']. ") "
                .$data['last']." ".$data['first']." ".$data['fname'];
            }
            if ( $change['status_after'] == 5 &&  $change['status_before'] != 5) {
                $total['rows_updated']["moved"][]=
                $data['spec_aem'].": (status: ".$change['status_before']." =>
".$change['status_after']. ") "
                .$data['last']." ".$data['first']." ".$data['fname'];
            }
            if ( $change['status_after'] == 6 && $change['status_after'] != 6) {
                $total['rows_updated']["togrd"][]=
                $data['spec_aem'].": (status: ".$change['status_before']." =>
".$change['status_after']. ") "
                .$data['last']." ".$data['first']." ".$data['fname'];
            }
            if($change['firstname_after']!=$change['firstname_before']){
```

```
                $total['rows_updated']["first"][]=
                    $data['spec_aem'].": (firstname: ".$change['firstname_before']."
=> ".$change['firstname_after']. ") "
                    .$data['last']." ".$data['first']." ".$data['fname'];
            }
            if($change['lastname_after']!=$change['lastname_before']){
                $total['rows_updated']["last"][]=
                    $data['spec_aem'].": (lastname: ".$change['lastname_before']."
=> ".$change['lastname_after']. ") "
                    .$data['last']." ".$data['first']." ".$data['fname'];
            }

        }

    }

    return $total;
});

return $total;
}
```

ΠΑΡΑΡΤΗΜΑ 2

```php
/**
 * Μετατρέπει το αρχείο passwd σε πίνακα
 *
 * @param $file
 * @return array
 * @throws \Exception
 */
private function toArray($file){
    try {
        $aetos=array();
        $interpreter = new Interpreter();
        $interpreter->addObserver(function (array $row) use (&$aetos) {
            $aetos[] = [
                'username'=>$row[0],
                'password'=>$row[1],
                'uid'=>intval($row[2]),
                'gid'=>intval($row[3]),
                'uid_info'=>mb_convert_case($row[4],MB_CASE_UPPER, "UTF-8"),
                'home_dir'=>$row[5],
                'com_shell'=>$row[6]
            ];
        });

        $this->lexer->parse($file->getPathname(), $interpreter);
        return $aetos;
    } catch (\Exception $e) {
        throw new \Exception("File type is not in passwd format");
    }

}
```

```
/**
 * Ενημέρωση πίνακα φοιτητών της βάσης
 *
 * @param array $aetos
 * @return array $changes
 */
public function import($aetos)
{
    DB::disableQueryLog();

    $changes=\DB::transaction(function() use ($aetos) {

        $changes=[
            'rows_added'=>0,
            'rows_updated'=>0
        ];

        foreach ($aetos as $aetos_row) {
            $exists = Aetos::where('username','=', $aetos_row['username'])->first();

            try {
                if(!$exists) {
                    Aetos::create($aetos_row);
                    $changes['rows_added']+=1;
                }
                else{
                    if (!$this->areSame($aetos_row,$exists)) {
                        $exists->update($aetos_row);
                        $exists->save();
                        $changes['rows_updated']+=1;
                    }
                }

            }catch (Exception $e) {}
```

```
        }
        $this->update_imports($changes);
        return $changes;
    });

    return $changes;
}


/**
 * Ελέγχει εάν οι δυο εγγραφές είναι ίδιες
 *
 * @param $aetos_row
 * @param $exists
 * @return bool
 */
private function areSame($aetos_row,$exists)
{
    if($aetos_row['username']!=$exists->username) return false;
    if($aetos_row['password']!=$exists->password) return false;
    if($aetos_row['uid']!=$exists->uid) return false;
    if($aetos_row['gid']!=$exists->gid) return false;
    if($aetos_row['uid_info']!=$exists->uid_info) return false;
    if($aetos_row['home_dir']!=$exists->home_dir) return false;
    if($aetos_row['com_shell']!=$exists->com_shell) return false;

    return true;
}
```

ΠΑΡΑΡΤΗΜΑ 3

```php
/**
 * Προσθέτει το username του φοιτητή
 *
 * @param Student $student
 */
public function setUsername(Student $student)
{
    $username = $this->buildSugestUsername($student);

    while ($this->username_exists($username)) {
        $username = $this->buildSugestUsername($student);
    }

    $student->update(['aetos_username' => $username]);
}
```

```php
/**
 * Επιστρέφει το προτεινόμενο username για τον φοιτητή
 *
 * @param Student $student
 * @return string
 * @throws \InvalidArgumentException
 */
private function buildSugestUsername($student)
{
    $name = htmlspecialchars(mb_substr($student->first, 0, 1)) .
htmlspecialchars($student->last);
    $name = (mb_substr($name, 0, 7));
```

```
    $suggestion = strtr($name, array(
        'Α' => 'A', 'Β' => 'V', 'Γ' => 'G', 'Δ' => 'D', 'Ε' => 'E', 'Ζ' => 'Z',
        'Η' => 'I', 'Θ' => 'TH', 'Ι' => 'I', 'Ϊ' => 'I', 'Κ' => 'K', 'Λ' => 'L', 'Μ' => 'M',
        'Ν' => 'N', 'Ξ' => 'X', 'Ο' => 'O', 'Π' => 'P', 'Ρ' => 'R', 'Σ' => 'S',
        'Τ' => 'T', 'Υ' => 'Y', 'Φ' => 'F', 'Χ' => 'X', 'Ψ' => 'PS', 'Ω' => 'O',
        'α' => 'a', 'β' => 'v', 'γ' => 'g', 'δ' => 'd', 'ε' => 'e', 'ζ' => 'z',
        'η' => 'i', 'θ' => 'th', 'ι' => 'i', 'κ' => 'k', 'λ' => 'l', 'μ' => 'm',
        'ν' => 'n', 'ξ' => 'x', 'ο' => 'o', 'π' => 'p', 'ρ' => 'r', 'σ' => 's',
        'τ' => 't', 'υ' => 'y', 'φ' => 'f', 'χ' => 'x', 'ψ' => 'ps', 'ω' => 'o',
        'ς' => 's', 'ά' => 'a', 'έ' => 'e', 'ή' => 'i', 'ί' => 'i', 'ό' => 'o',
        'ύ' => 'y', 'ώ' => 'o', 'ΐ' => 'i', 'ϋ' => 'y', 'ΰ' => 'i', 'ΰ' => 'y'));

    $username = strtolower($suggestion);

    if (strlen($username) > 7) {
        $username = (mb_substr($username, 0, 7));
    } elseif (strlen($username) < 7) {
        while ((strlen($username) < 7)) {
            $username .= rand(1, 9);
        }
    }

    if (Aetos::count()) {
        while (Aetos::where('username', $username)->first()) {
            $username = substr($username, 0, -1) . rand(1, 9);
        }
    }

    return $username;
}
```

```
/**
 * Ενεργοποιεί τους νέους λογαριασμούς
 * @param $inputs
 * @return bool
 * @internal param int|null $id
 */
public function activate($inputs)
{
    foreach ($inputs as $student_id => $suggested_username) {
        $student = Student::find($student_id);
        if ($student->where('aetos_username', '=', $suggested_username)->where('id', '!=', $student_id)->count() > 0 || Aetos::where('username', '=', $suggested_username)->count() > 0) return false;

        $student->update(['aetos_username' => $suggested_username, 'exist' => 1]);
        DB::table('status_changes')->where('activated', 0)->update(['activated' => 1]);
    }

    return true;
}



/**
 * Επιστρέφει τα στατιστικά των φοιτητών
 *
 * @return array
 */
public function get_students_statistics()
{
    $statistics_repo = new StatisticsRepository();
```

```
    return [
        'students_stats_per_year' => $statistics_repo->students_stats_per_year(),
        'student_stats_graduated_per_exam' => $statistics_repo->student_stats_graduated_per_exam(),
        'student_stats_halted_per_exam' => $statistics_repo->student_stats_halted_per_exam(),
        'student_stats_deleted_per_exam' => $statistics_repo->student_stats_deleted_per_exam()
    ];

}
```

ΟΔΗΓΟΣ ΧΡΗΣΗΣ ΛΟΓΙΣΜΙΚΟΥ

ΑΠΑΙΤΗΣΕΙΣ ΣΥΣΤΗΜΑΤΟΣ

Η εφαρμογή για την εγκατάστασή της, απαιτεί:

- Έκδοση PHP μεγαλύτερη ή ίση της 5.4
- Λειτουργικό σύστημα LINUX
- Composer
- Εγκατεστημένο το git
- Node
- Αποθηκευτικό χώρο 285 MB

ΕΓΚΑΤΑΣΤΑΣΗ

1. Η εφαρμογή μπορεί να εγκατασταθεί σε οποιοδήποτε κατάλογο του συστήματος αλλά για λόγους ομοιοτυπίας χρησιμοποιείται συνήθως ο κατάλογος /var/www
2. Έπειτα γίνεται λήψη της εφαρμογής από το repository του vcs(Github,bitbucket κλπ.) που είναι ανεβασμένη με την εντολή git clone «το_URL_του_repository» «το_όνομα_του_φακέλου». (Σε περίπτωση που το repository είναι private, το σύστημα θα ζητήσει και τον κωδικό του χρήστη)
3. Αφού γίνει η λήψη, ο χρήστης εισέρχεται στον φάκελο που δημιουργήθηκε με την εντολή cd «όνομα_φακέλου». Εκεί εκτελεί την εντολή composer install, npm install και bower install. Αυτό έχει ως αποτέλεσμα την εγκατάσταση των πακέτων που είναι δηλωμένα στα ομώνυμα αρχεία τους(τύπου .json) που περιέχουν τις πληροφορίες για αυτά τα πακέτα.
4. Τέλος, το τελευταίο βήμα για να ολοκληρωθεί η διαδικασία της εγκατάστασης είναι η εκτέλεση της εντολής chmod –R 0777 storage(αν χρειάζεται δικαιώματα διαχειριστή προστίθεται το sudo στην αρχή της εντολής), ώστε να γίνει εγγράψιμος ο φάκελος storage στον οποίο αποθηκεύονται η cache της εφαρμογής, τα session της κλπ.

ΡΥΘΜΙΣΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

ΕΞΥΠΗΡΕΤΗΤΕΣ ΕΦΑΡΜΟΓΗΣ

Το σύστημα μπορεί να δουλέψει με δυο ειδών εξυπηρετητές. Τον Apache και τον nginx. Συνήθως προτιμάτε η επιλογή του nginx καθώς έχει καλύτερη απόδοση στην εξυπηρέτηση των στατικών αρχείων στον server, δεν χρειάζεται .htaccess αρχεία για την ρύθμιση της ασφάλειας και των φακέλων και μπορεί να χρησιμοποιηθεί ως proxy server πίσω από τον apache για λόγους ασφαλείας και ενθυλάκωσης. Σε περίπτωση που δεν είναι εγκατεστημένη η PHP ή ο nginx server θα πρέπει να εκτελεστεί η εντολή «sudo apt-get install nginx php5-fpm php5-cli php5-mcrypt git».

ΔΗΜΙΟΥΡΓΙΑ VHOST

- Δημιουργία ενός νέου vhost αρχείου με την εντολή sudo nano /etc/nginx/sites-available/default από το τερματικό
  Στο αρχείο που ανοίγει φαίνονται οι προεπιλεγμένες ρυθμίσεις του vhost. Το βασικό που πρέπει να αλλαχτεί είναι το «onoma_fakelou_efarmogis». Τα υπόλοιπα αποτελούν πρόσθετες ρυθμίσεις.

```
server {
        listen   80 default_server;

        root /var/www/onoma_fakelou_efarmogis/public/;
        index index.php index.html index.htm;

        location / {
            try_files $uri $uri/ /index.php$is_args$args;
        }

        # pass the PHP scripts to FastCGI server listening on /var/run/php5-fpm.sock
        location ~ \.php$ {
                try_files $uri /index.php =404;
```

```
            fastcgi_pass unix:/var/run/php5-fpm.sock;
            fastcgi_index index.php;
            fastcgi_param SCRIPT_FILENAME
$document_root$fastcgi_script_name;
            include fastcgi_params;
      }
}
```

Αφού ολοκληρωθούν οι αλλαγές, το αρχείο αποθηκεύεται με το ctrl+x και έπειτα yes και enter. Μετά την περάτωση των παραπάνω εντολών πρέπει να γίνει επανεκκίνηση του nginx server με τις εντολές:

```
service php5-fpm restart
service nginx restart
```

ΑΡΧΙΚΟΠΟΙΗΣΗ ΒΑΣΗΣ

Την αρχικοποίηση της βάσης δεδομένων την αναλαμβάνει εξ ’ολοκλήρου το framework της laravel . Στον φάκελο ./database/migrations της εφαρμογής είναι τοποθετημένοι οι πίνακες της βάσης, και στον φάκελο ./database/seeds τα αντίστοιχα (εάν υπάρχουν) αρχεία προκειμένου να δημιουργούνται οι πίνακες της βάσης και μετά να εισέρχονται τα δεδομένα στη βάση.

Η δημιουργία των πινάκων γίνεται με την εντολή:

```
php artisan migrate
```

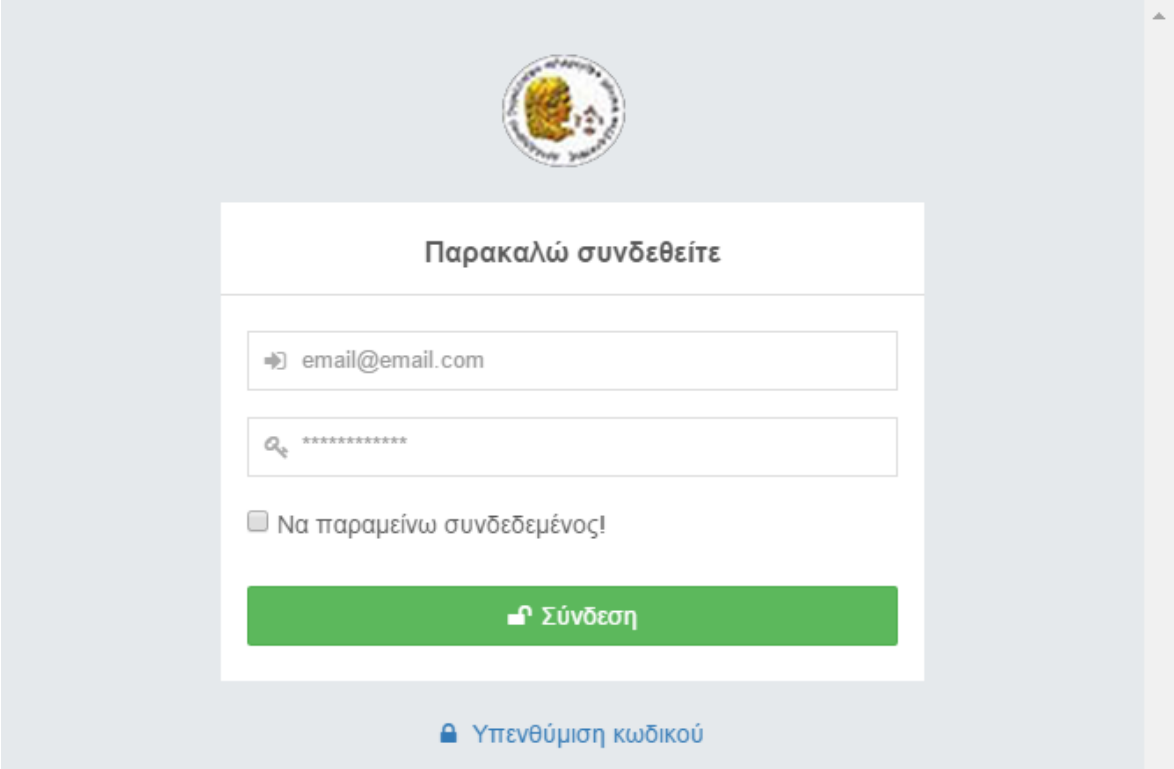
και στη συνέχεια για την εισαγωγή των δεδομένων στους πίνακες που δημιουργήθηκαν η εντολή:

```
php artisan db:seed
```

ΛΕΙΤΟΥΡΓΙΑ ΣΥΣΤΗΜΑΤΟΣ

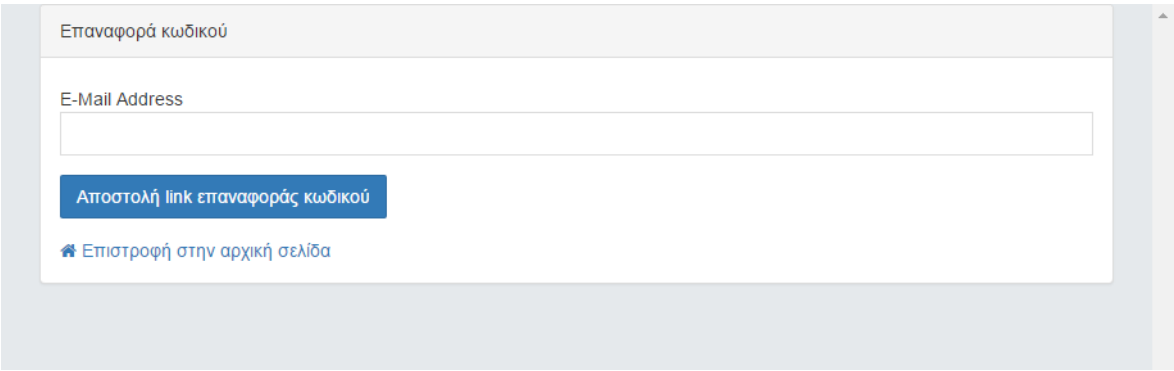
ΕΊΣΟΔΟΣ ΧΡΗΣΤΗ

Σ ’αυτή την οθόνη ο χρήστης πρέπει να δώσει τα συνθηματικά του ώστε να έχει πρόσβαση στο περιβάλλον της εφαρμογής(email, κωδικός χρήστη).



Εικόνα 10: Οθόνη σύνδεσης

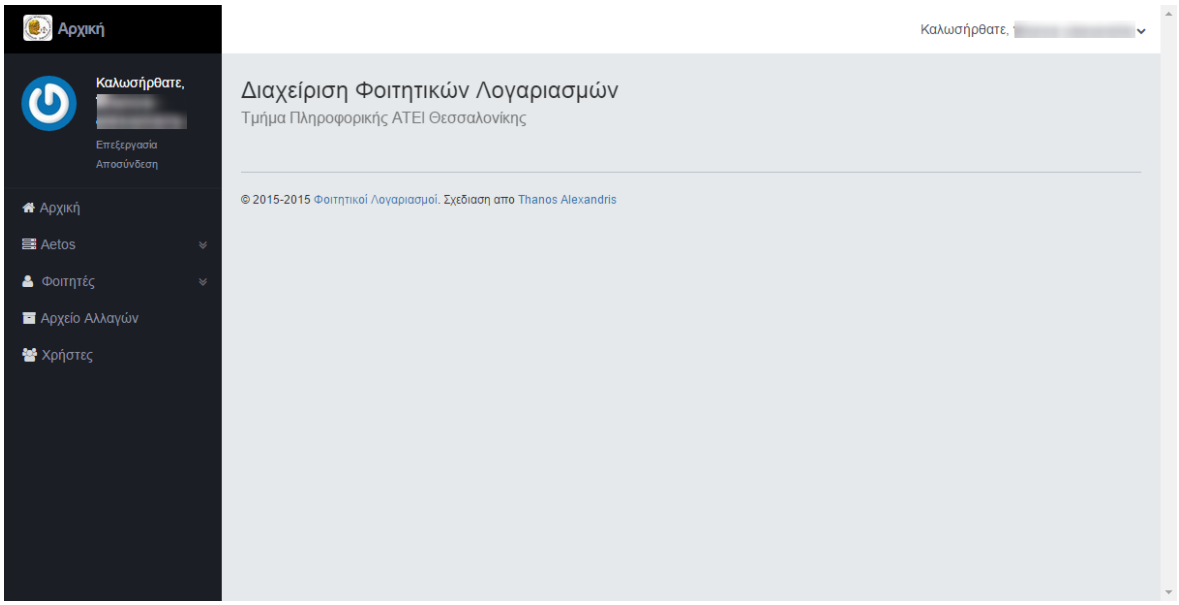
Σε περίπτωση που ξεχάσει τα συνθηματικά του μπορεί μεταβεί στην υπενθύμιση κωδικού όπου συμπληρώνοντας το email του θα του αποσταλεί ένα link στο email με οδηγίες για επαναφορά του κωδικού του.



Εικόνα 11:Σελίδα επαναφοράς κωδικού

ΑΡΧΙΚΗ ΣΕΛΙΔΑ

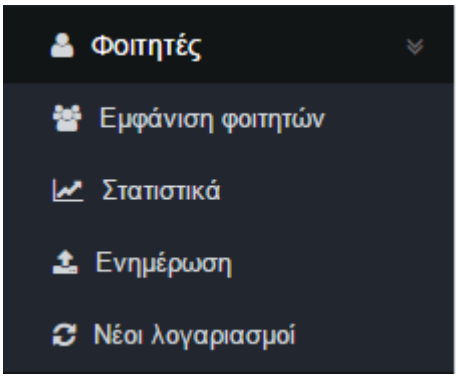
Μετά την επιτυχή είσοδο, ο χρήστης οδηγείτε στην κύρια οθόνη όπου βλέπει μια σειρά από επιλογές.



Εικόνα 12: Αρχική Οθόνη

ΦΟΙΤΗΤΕΣ

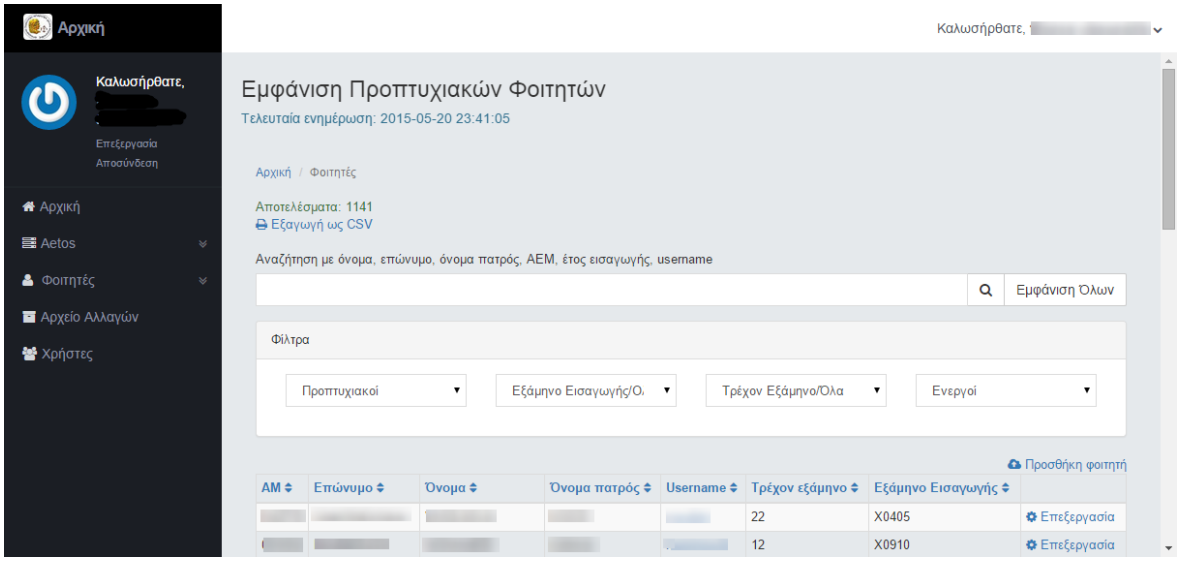
Η καρτέλα των φοιτητών περιλαμβάνει μια σειρά από επιλογές που μπορεί να επιλέξει ο χρήστης.



Εικόνα 13: Καρτέλα φοιτητών

ΕΜΦΑΝΙΣΗ ΦΟΙΤΗΤΩΝ

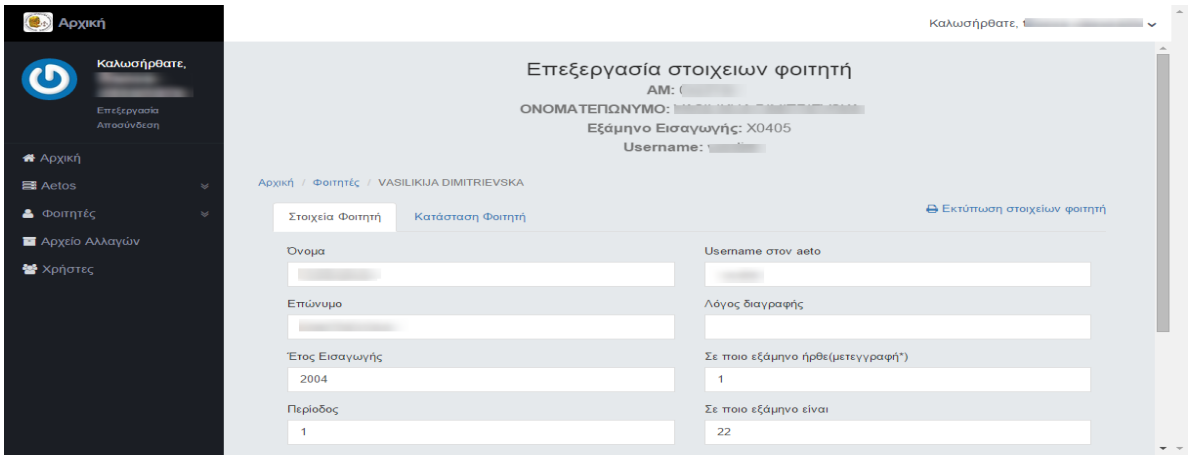
Εδώ μπορεί να γίνει εμφάνιση, αναζήτηση των φοιτητών με κριτήρια, φιλτράρισμα και ταξινόμησή τους όπως επίσης και αποθήκευσης των αποτελεσμάτων σε αρχείο csv. Ακόμα μπορεί να γίνει και εμφάνιση της καρτέλας ενός φοιτητή επιλέγοντας την επιλογή επεξεργασία στη τελευταία στήλη της κάθε εγγραφής.



Εικόνα 14: Σελίδα εμφάνισης φοιτητών

ΕΠΕΞΕΡΓΑΣΙΑ ΦΟΙΤΗΤΗ

Εδώ μπορεί να γίνει επεξεργασία ή διαγραφή της καρτέλας του φοιτητή καθώς επίσης και αποθήκευση των στοιχείων του σε αρχείο τύπου pdf.



Εικόνα 15: Σελίδα επεξεργασίας φοιτητή

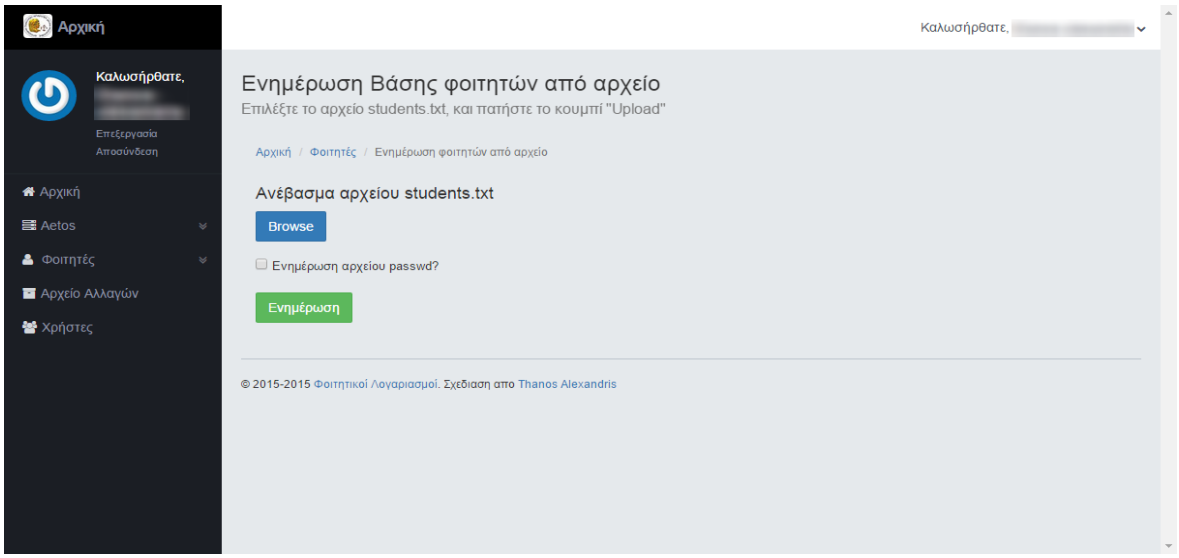
ΣΤΑΤΙΣΤΙΚΑ

Σ' αυτή τη σελίδα ο χρήστης μπορεί να δεί τα στατιστικά των φοιτητών σε διάφορα γραφήματα, καθώς και να εξάγει τα αποτελέσματα σε διάφορους τύπους αρχεία(csv,xls,png κλπ) για περαιτέρω μελέτη και ανάλυση.

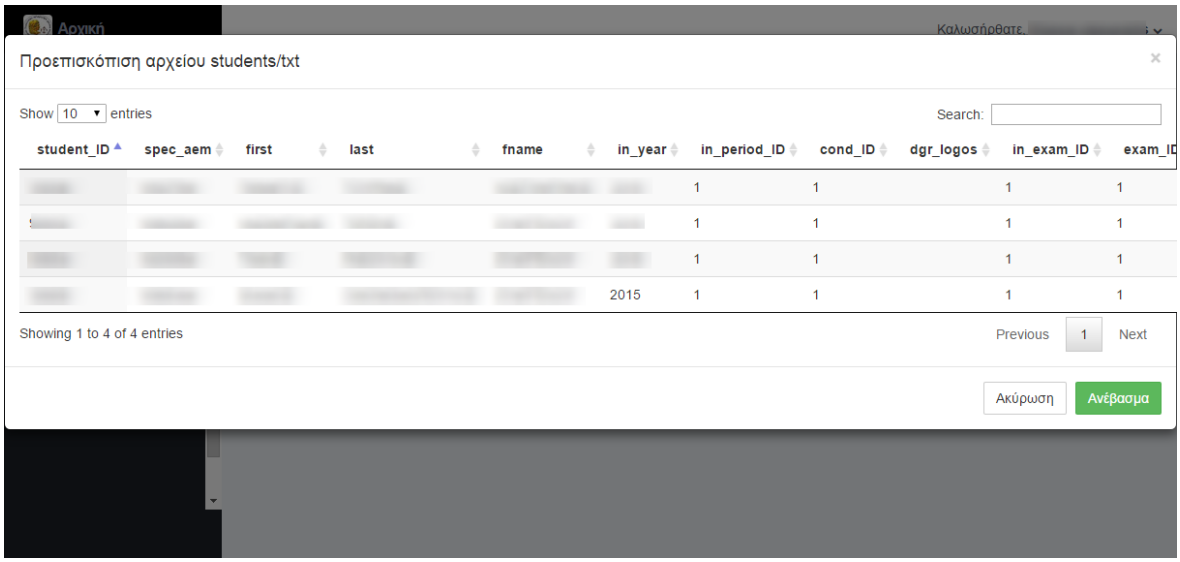
Εικόνα 16: Σελίδα στατιστικών φοιτητών

ΕΝΗΜΕΡΩΣΗ

Εδώ γίνεται η ενημέρωση των φοιτητών μέσω αρχείου students.txt



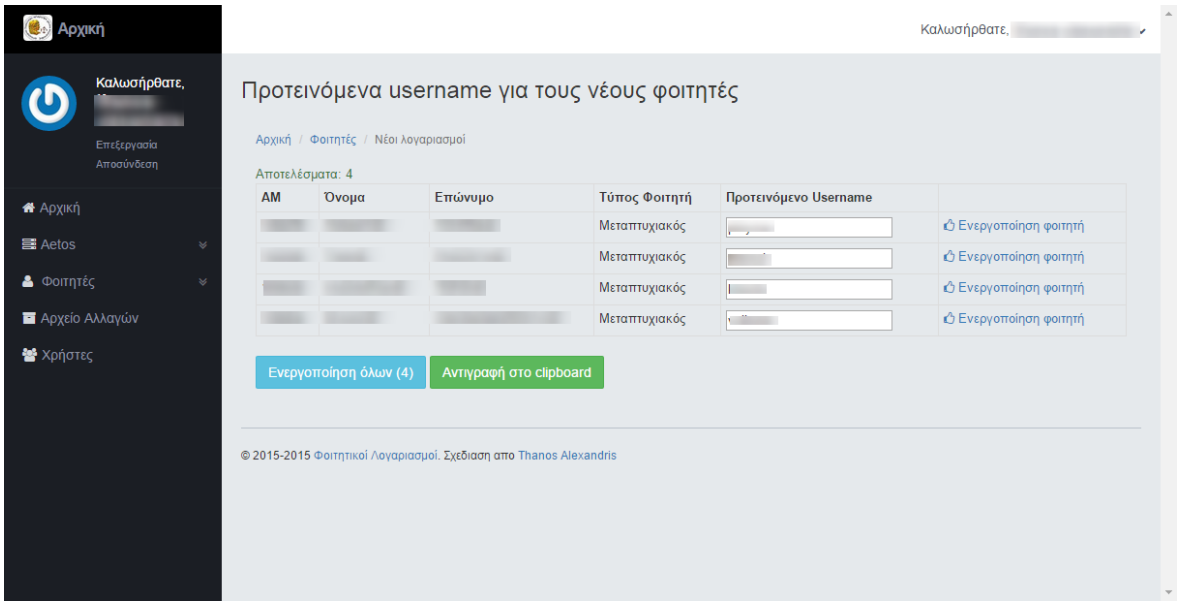
Εικόνα 17:Σελίδα ενημέρωσης φοιτητών



Εικόνα 18:Έλεγχος αρχείου students.txt πριν το ανέβασμα στον server

ΝΕΟΙ ΛΟΓΑΡΙΑΣΜΟΙ

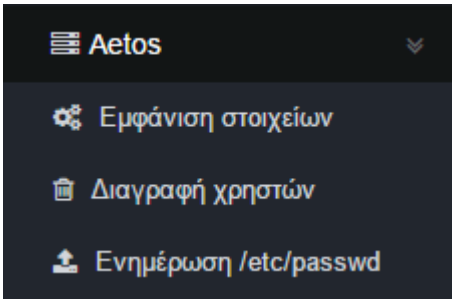
Αυτή η σελίδα περιέχει τα username των νέων φοιτητών τα οποία μπορεί να αλλάξει ο διαχειριστής του συστήματος



Εικόνα 19:Σελίδα προτεινόμενων username για τους νέους φοιτητές

AETOS

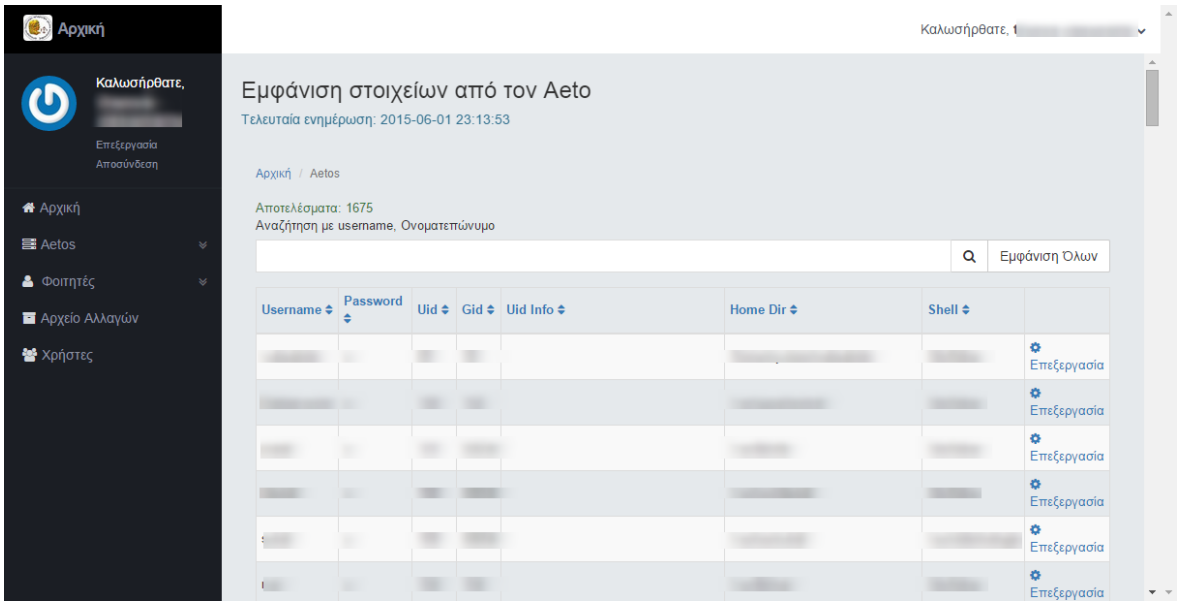
Σ ’αυτήν την καρτέλα εμφανίζονται οι επιλογές που έχει ο χρήστης σχετικά με τις εγγραφές που έχουν να κάνουν σχέση με τον server aetos.



Εικόνα 20: Καρτέλα Aetou

ΕΜΦΑΝΙΣΗ ΣΤΟΙΧΕΙΩΝ

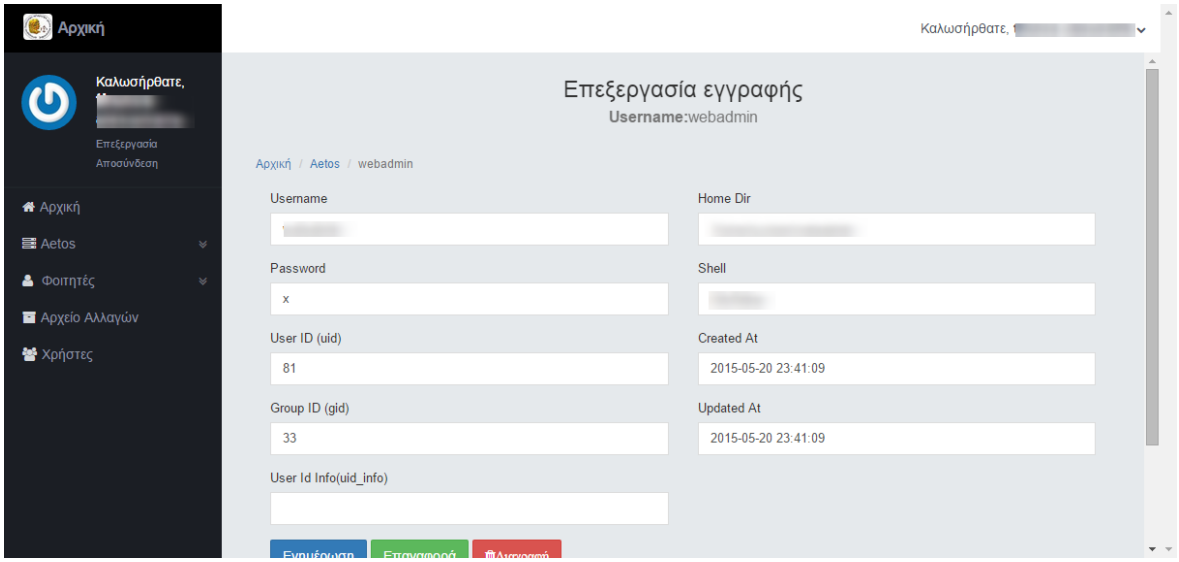
Εδώ εμφανίζονται όλα τα στοιχεία των εγγραφών που περιέχονται στο αρχείο /etc/passwd, με δυνατότητα αναζήτησης και ταξινόμησης. Επίσης μπορεί να γίνει επεξεργασία της κάθε εγγραφής επιλέγοντας την επιλογή επεξεργασία της τελευταίας στήλης.



Εικόνα 21:Εμφάνιση στοιχείων aetou

ΕΠΕΞΕΡΓΑΣΙΑ ΕΓΓΡΑΦΗΣ

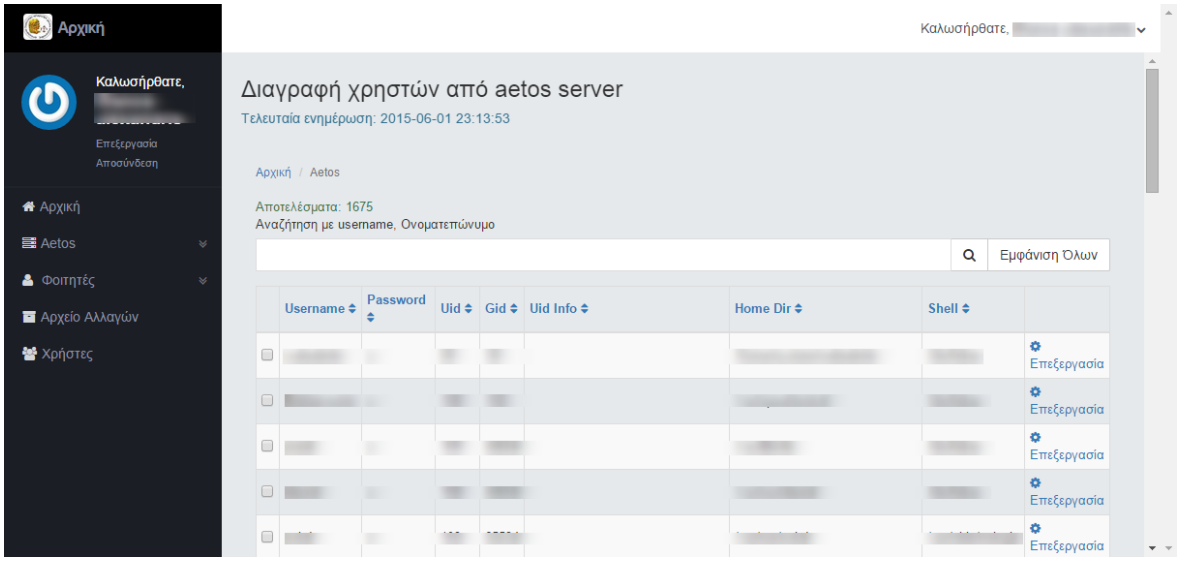
Εδώ μπορεί να γίνει επεξεργασία ή διαγραφή της συγκεκριμένης εγγραφής.



Εικόνα 22:Επεξεργασία εγγραφής aetou

ΔΙΑΓΡΑΦΗ ΧΡΗΣΤΩΝ

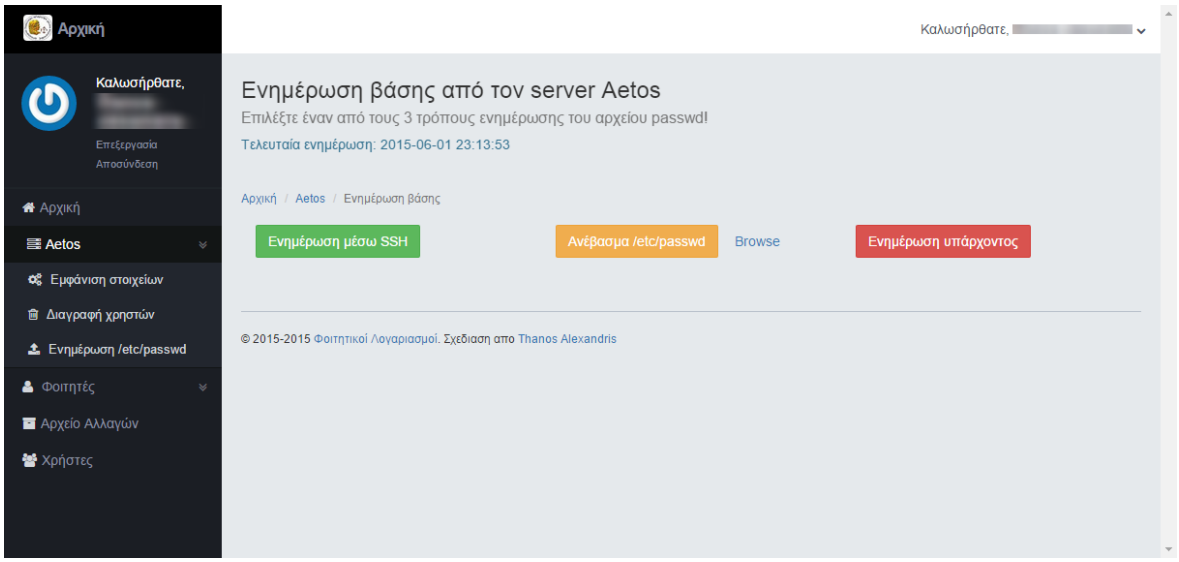
Επιλέγοντας αυτή την επιλογή της καρτέλας «Aetos», ο διαχειριστής μπορεί να διαγράψει χειροκίνητα τους χρήστες από τη βάση του aetos server.



Εικόνα 23:Σελίδα διαγραφής φοιτητών από τη βάση του aetou

ΕΝΗΜΕΡΩΣΗ /ETC/PASSWD

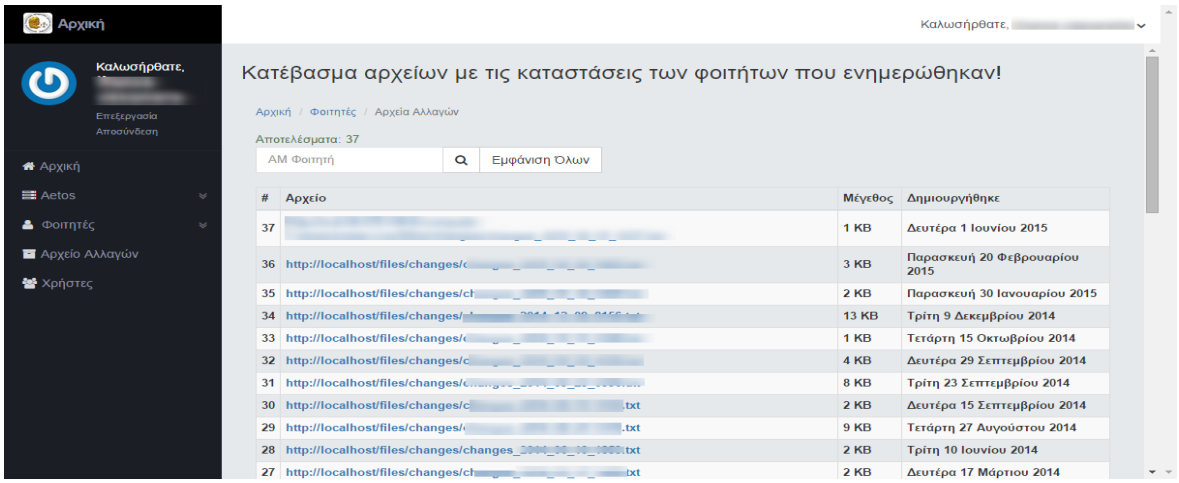
Σ ‘αυτήν την καρτέλα μπορεί να γίνει ενημέρωση της βάσης του aetou με 3 τρόπους όπως φαίνεται και στην εικόνα 24.



Εικόνα 24: Σελίδα ενημέρωσης aetou

ΑΡΧΕΙΟ ΑΛΛΑΓΩΝ

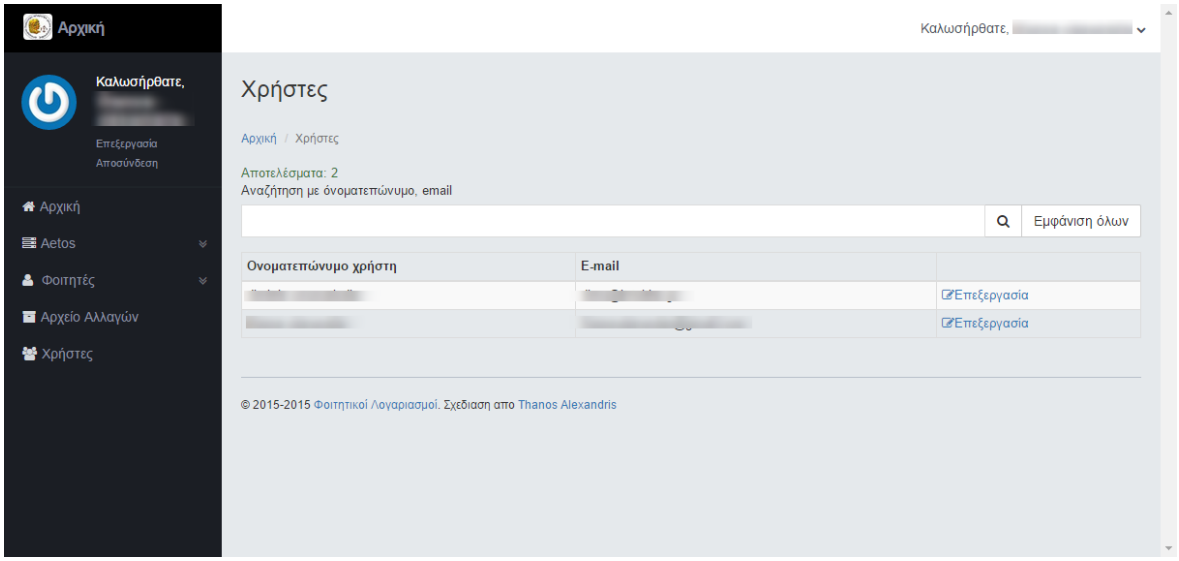
Οι αλλαγές που γίνονται στην βάση από το ανέβασμα των αρχείων students.txt παρουσιάζονται σ ’αυτή τη σελίδα, με δυνατότητα μεταφόρτωσης στον τοπικό δίσκο του υπολογιστή του διαχειριστή. Επίσης δίνεται και η δυνατότητα αναζήτησης με βάση το ΑΜ ενός φοιτητή.



Εικόνα 25: Σελίδα αρχείου αλλαγών

ΧΡΗΣΤΕΣ

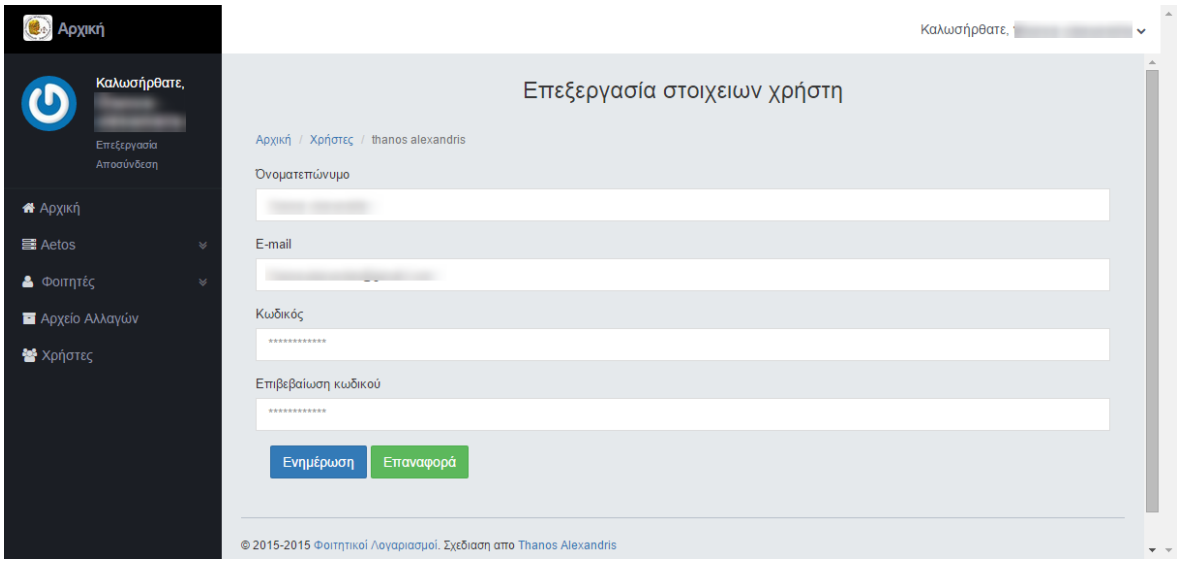
Στην σελίδα χρήστες ο διαχειριστής μπορεί να εμφανίσει τους χρήστες του συστήματος.



Εικόνα 26:Σελίδα παρουσίασης χρηστών

ΠΡΟΦΙΛ ΧΡΗΣΤΗ

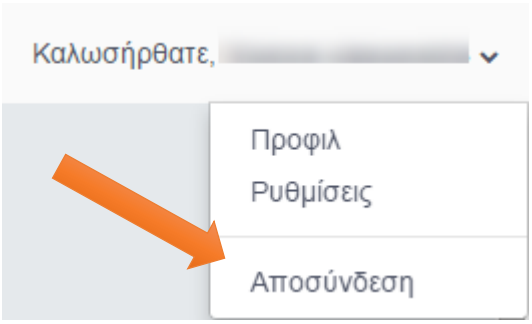
Στην επεξεργασία του προφίλ χρήστη, δύναται ο συνδεδεμένος χρήστης να αλλάξει κάποια στοιχεία του προφίλ του όπως ονοματεπώνυμο, email και κωδικό πρόσβασης.



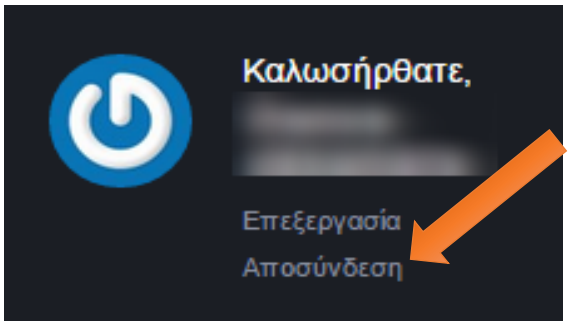
Εικόνα 27:Σελίδα προφίλ χρήστη

ΑΠΟΣΥΝΔΕΣΗ

Η αποσύνδεση του χρήστη μπορεί να γίνει με δυο τρόπους όπως φαίνεται και στις εικόνες παρακάτω.



Εικόνα 28:1ος τρόπος αποσύνδεσης



Εικόνα 29: 2ος τρόπος αποσύνδεσης