ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ
ΙΔΡΥΜΑ ΛΑΡΙΣΑΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΜΕ ΘΕΜΑ:

Υπηρεσία ενοποιημένης παροχής και ταξινόμησης διαφόρων υπηρεσιών και πληροφοριών του ΤΕΙ Λάρισας με χρήση κοινών λογαριασμών σε LDAP Server και υλοποίηση σε Django Web Framework

ΟΙ ΣΠΟΥΔΑΣΤΕΣ:

Σπανός Ιωάννης Τ-1105

Χατζημίχος Θεώδωρος Τ-1387

Ο ΕΠΙΒΛΕΠΩΝ:

Χρήστος Σωμαράς

ΛΑΡΙΣΑ 2010

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Λάρισα …/…../2010

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1.

2.

3.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα θέλαμε να ευχαριστήσουμε θερμά τον υπεύθυνο καθηγητή κύριο Χρήστο Σωμαρά που μας καθοδήγησε ώστε να επιτευχθεί ένα άρτιο αποτέλεσμα. Επίσης, τους φίλους μας από το Linux Team του ΤΕΙ Λάρισας και την ελληνική κοινότητα Gentoo Linux, για την απεριόριστη βοήθεια που μας προσέφεραν, και την προώθηση της εφαρμογής ώστε να επιτύχει από πολύ νωρίς. Ελπίζουμε για πολλά χρόνια ακόμα να βρισκόμαστε και να ανταλλάσουμε γνώσεις και εμπειρίες πάνω στο ελεύθερο λογισμικό.

Με εκτίμηση,

Σπανός Ιωάννης

Χατζημίχος Θεόδωρος

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Μέσα από διάφορα sites και υπηρεσίες, η πληροφορία διασκορπίζεται και χάνεται. Από την αρχή της φοίτησής μας μας φαινόταν εξαιρετικά δύσκολο να παρακολουθούμε τις ανακοινώσεις που παρέχονταν από τις διάφορες υπηρεσίες του ΤΕΙ Λάρισας και να μένουμε συνεχώς ενημερωμένοι. Συνοπτικά, για ένα φοιτητή του τμήματος Τεχνολογίας Πληροφορικής και Τηλεπικοινωνιών για παράδειγμα, υπάρχουν ανακοινώσεις που αφορούν τη σχολή του στην κεντρική σελίδα του ΤΕΙ Λάρισας, στην ιστοσελίδα του τμήματός του, στην ιστοσελίδα / προφίλ του κάθε καθηγητή στην ιστοσελίδα της πληροφορικής καθώς και στην ιστοσελίδα /προφίλ του κάθε μαθήματος στην Πλατφόρμα Ασύγχρονης Τηλεκπαίδευσης (E-class). Επίσης, ένας καθηγητής επιλέγει πότε την ιστοσελίδα της Πληροφορικής και πότε το E-class για να αναρτήσει μια ανακοίνωση. Φυσικά δεν αναφέρονται άλλες υπηρεσίες που υπάρχουν στο ΤΕΙ και δεν έχουν άμεση σχέση με τη σχολή το μάθημα ή το φοιτητή, αλλά μπορεί να επηρεάζουν τη φοίτησή του. Εμφανές παράδειγμα η ιστοσελίδα του Γραφείου Διασυνδέσεως που περιλαμβάνει ανακοινώσεις σχετικά με διαθέσιμες πρακτικές εργασίες από διάφορες εταιρίες εντός και εκτός της Λάρισας για όλες τις σχολές.

Είναι πλέον εμφανής η δυσκολία παρακολούθησης όλων αυτών των πηγών σε όλη τη διάρκεια του εξαμήνου. Παράλληλα, δημιουργείται και το πρόβλημα των διαφόρων ονομάτων χρήστη και κωδικών πρόσβασης που δίνονται στο φοιτητή, τα οποία πρέπει να απομνημονεύει.

Ένα άλλο συχνό φαινόμενο που παρατηρείται, είναι η άγνοια των φοιτητών για την ύπαρξη όλων αυτών των υπηρεσιών, οι οποίες θα μπορούσαν να τον διευκολύνουν σε πολλές περιπτώσεις, όπως για παράδειγμα η εύρεση πρακτικής άσκησης μέσα από την πλατφόρμα του Γραφείου Διασύνδεσης που προαναφέρθηκε.

Το πρόβλημα γίνεται πιο έντονο όταν κάποια υπηρεσία που παρέχει ανακοινώσεις και έγγραφα μεγάλης σημασίας για το φοιτητή είναι για κάποιο λόγο μη διαθέσιμη, φαινόμενο πολύ συχνό στη σχολή μας. Ιδιαίτερα σε περιόδους εξεταστικής δημιουργείται τεράστιο πρόβλημα. Η πτυχιακή μας εργασία μέσα από την ιστοσελίδα Cronos έρχεται να καλύψει ακριβώς αυτή την ανάγκη συγκέντρωσης πληροφοριών από το φοιτητή.

Κύριος στόχος μας ήταν η δημιουργία ενός συστήματος, στο οποίο κατά την εγγραφή

του χρήστη θα ζητάει τα στοιχεία του από τις διάφορες υπηρεσίες του ΤΕΙ, και θα τα αποθηκεύει στο προφίλ του. Στο τέλος, ο φοιτητής θα επιλέγει ένα όνομα χρήστη και κωδικό πρόσβασης της επιλογής του, το οποίο θα χρησιμοποιεί στην ιστοσελίδα μας, και μέσα από ένα συμπαγές γραφικό περιβάλλον να μπορεί να προβάλει τις ανακοινώσεις που τον ενδιαφέρουν, καθώς και άλλα προσωπικά (βαθμολογία, δήλωση, e-mail) και μη (μαθήματα e-class) στοιχεία. Για την ανάπτυξη της πλατφόρμας μας χρησιμοποιήθηκε το Django Web Framework, το οποίο είναι γραμμένο στη γλώσσα προγραμματισμού Python, και η αποθήκευση των στοιχείων γίνεται στην γνωστή υπηρεσία καταλόγου openLDAP. Η επιλογή των παραπάνω εργαλείων δεν έγινε τυχαία, καθώς θέλαμε το αποτέλεσμα να είναι μια εφαρμογή εύχρηστη, φιλική προς το χρήστη, ευπαρουσίαστη, λειτουργική, ανοιχτή, και πλήρως επεκτάσιμη.

ΚΕΦΑΛΑΙΟ 1ο: ΤΟ ΣΥΣΤΗΜΑ ΔΙΑΧΕΙΡΙΣΗΣ ΚΩΔΙΚΑ SUBVERSION

1.1 Περιγραφή του Subversion

Η πτυχιακή εργασία έχει ανατεθεί σε δύο άτομα, οπότε το πρώτο πρόβλημα που έπρεπε να λυθεί ήταν να βρεθεί τρόπος ώστε να είμαστε πλήρως ενημερωμένοι μεταξύ μας για την δουλειά που έχει κάνει ο άλλος, ώστε να μην χρειάζεται να είμαστε σε διαρκή επικοινωνία, και παράλληλα να γνωρίζουμε ακόμα και τα μελλοντικά σχέδια του άλλου, ώστε να μην τύχει να δουλέψουμε στον ίδιο τομέα και σπαταλήσουμε άδικα χρόνο και κόπο. Φυσικά αυτό είναι πολύ εύκολο να γίνει με ένα σύστημα διαχείρισης κώδικα. Επιλέξαμε το Subversion, το οποίο είναι πολύ απλό στη χρήση του και με το οποίο είχαμε δουλέψει ήδη και οι δύο στο παρελθόν και είμασταν εξοικειωμένοι με τη χρήση του.

Το Subversion και κάθε σύστημα διαχείρισης κώδικα αποθηκεύει τα αρχεία σε μια μικρή βάση δεδομένων του, και με κάθε αλλαγή που υποβάλεται στα αρχεία, το σύστημα εμφανίζει ποιος χρήστης έκανε την αλλαγή, πότε, και ποιες είναι οι γραμμές που αλλάχτηκαν. Κάθε ανανέωση παίρνει έναν αύξοντα αριθμό, οπότε είναι εύκολη η μετάβαση σε μια συγκεκριμένη προηγούμενη έκδοση. Με κάθε υποβολή ο χρήστης γράφει και ένα μικρό μήνυμα, στο οποίο περιγράφει τις αλλαγές που έκανε και ότι άλλο ενδιαφέρει τους υπόλοιπους προγραμματιστές της ομάδας. Οπότε διατηρείται ένα πλήρες ιστορικό με την πορεία ανάπτυξης μιας εφαρμογής στο συγκεκριμένο αποθετήριο. Μπορούν να οριστούν δικαιώματα στους χρήστες ως προς ποιοι θα έχουν πρόσβαση εγγραφής ή ανάγνωσης στο αποθετήριο ή σε υποκαταλόγους αυτού.

1.2 Δημιουργία του αποθετηρίου

Υπάρχουν πολλοί τρόποι παραμετροποίησης του subversion, όπως με τον ενσωματωμένο svn δαίμονα (η πιο μη ασφαλής μέθοδος), svn+ssh, μέσω apache (http και https). Επιλέξαμε το δεύτερο τρόπο, svn μέσω ασφαλούς κελύφους ssh. Για τη δημιουργία του αποθετηρίου δημιουργήθηκε ένας χρήστης με το όνομα svn στον εξυπηρετητή. Οι προγραμματιστές θα παίρναν πρόσβαση σε αυτό μέσω εικονικών χρηστών, και η ταυτοποίηση των στοιχείων τους θα γινόταν με κλειδιά ssh αντί για passwords, για αυξημένη ασφάλεια.

Οι εντολές για τη δημιουργία ενός χρήστη και ενός αποθετηρίου στον εξυπηρετητή είναι οι εξής:

```
(Δημιουργία χρήστη και ορισμός του home folder του)

# useradd -m -s /bin/bash -d /var/svn svn

# su – svn

(Δημιουργία του αποθετηρίου)

$ svnadmin create ptixiaki

$ ls ptixiaki

conf db format hooks locks README.txt
```

Το επόμενο βήμα ήταν να εισάγουμε τα δημόσια κλειδιά μας στο σύστημα ώστε να δημιουργηθούν οι εικονικοί χρήστες:

```
$ cd .ssh

$ cat /path/to/cephalon.pub >> .ssh/authorized_keys

$ cat /path/to/tampakrap.pub >> .ssh/authorized_keys
```

Το δημόσιο κλειδί κάθε χρήστη είναι απλά ένα αλφαριθμητικό το οποίο κάνει ταυτοποίηση με το αντίστοιχο ιδιωτικό κλειδί. Το ζεύγος κλειδιών έχει δημιουργηθεί από τον κάθε χρήστη. Τροποποιούμε το παρακάτω αρχείο, ώστε οι χρήστες να έχουν δικαίωμα να εκτελέσουν μόνο εντολές που αφορούν το subversion, και να μην έχουν περαιτέρω πρόσβαση στον εξυπηρετητή:

```
$ vim /var/svn/.ssh/authorized_keys

command="svnserve -t -r /var/svn --tunnel-user=tampakrap",no-port-forwarding,no-agent-forwarding,no-X11-forwarding,no-pty ssh-dss AAAAalskdfj.....

command="svnserve -t -r /var/svn --tunnel-user=cephalon",no-port-forwarding,no-
```

```
agent-forwarding,no-X11-forwarding,no-pty ssh-dss AAAAsdfjklslf.....
```

Με αυτό τον τρόπο ενημερώνουμε το μηχάνημα ότι εκάστοτε ο χρήστης κάθε φορά που θα συνδέεται στον εξυπηρετητή με το πρωτόκολλο svn+ssh θα του δίνεται το όνομα χρήστη tampakrap ή cephalon (και όχι svn που θα γινόταν σε αντίθετη περίπτωση), και παράλληλα απορρίπτουμε τα δικαιώματα κάθε άλλης εντολής που θα χρησιμοποιηθεί πέραν των εντολών για το svn.

1.3 Δικαιώματα των χρηστών

Επόμενο βήμα ήταν να ρυθμίσουμε τα δικαιώματα των χρηστών αυτών ώστε να έχουν πρόσβαση εγγραφής στο νέο αποθετήριο, και όλοι οι υπόλοιποι χρήστες να έχουν μόνο δικαιώματα ανάγνωσης. Αυτό γίνεται στο αρχείο /var/svn/ptixiaki/conf/authz:

```
# cat /var/svn/ptixiaki/conf/authz

[groups]

developers = cephalon, tampakrap

[ptixiaki:/]

@developers = rw

* = r
```

Δημιουργήθηκε ένα group developers, οι οποίοι έχουν δικαιώματα εγγραφής (rw) και όλοι οι υπόλοιποι (*) δικαιώματα ανάγνωσης.

Τελευταίο βήμα ήταν να δημιουργήσουμε ένα μικρό bash script, το οποίο μετά από κάθε αποστολή κώδικα θα εξάγει τη νέα έκδοση σε ένα φάκελο, ο οποίος θα είναι προσβάσιμος από τον web server και το νέο αποτέλεσμα θα φαίνεται απευθείας στην ιστοσελίδα. Κάθε σύστημα διαχείρισης κώδικα υποστηρίζει τέτοιες λειτουργίες, οι οποίες μπορούν να γίνουν και ποιο περίπλοκες, δηλαδή να εκτελούνται συγκεκριμένες εντολές και πριν την αποστολή. Τα αρχεία ρυθμίσεων βρίσκονται στο /var/svn/ptixiaki/hooks, και στην περίπτωσή μας μας ενδιαφέρει το αρχείο post-commit, το οποίο κάνουμε εκτελέσιμο και συμπληρώνουμε το κατάλληλο περιεχόμενο:

```
# cat /var/svn/ptixiaki/hooks/post-commit

#!/bin/bash

/usr/bin/svn update /home/code/cronos >> /var/log/svnserve.log
```

Το αποτέλεσμα καταγράφεται για την περίπτωση που υπάρξει αποτυχία.

Τεστάρουμε την νέα μας εγκατάσταση στέλνοντας ένα δοκιμαστικό αρχείο:

```
(τοπικά, κατεβάζουμε το αποθετήριο)

$ svn co svn+ssh://svn@cronos.teilar.gr/ptixiaki

Checked out revision 0.
```

Δημιουργείται ένας άδειος φάκελος, στον οποίο φτιάχνουμε ένα άδειο αρχείο test και το υποβάλουμε.

```
$ svn add test -m “initial commit”

$ svn commit
```

Για την αποφυγή αποστολής δυαδικών παραγόμενων αρχείων, δημιουργώντας απλά ένα αρχείο με όνομα .svnignore το svn αρνείται να αποστείλει τα περιεχόμενα του εν λόγω αρχείου. Δημιουργούμε το αρχείο με τα κάτωθι περιεχόμενα και το αποστέλουμε όπως προηγουμένως:

```
$ cat .svnignore

*.pyc
```

1.4 WebSVN και Apache

Για πιο εύκολη προβολή των αλλαγών στο αποθετήριο, χρησιμοποιήσαμε μια γραφική διεπαφή εν ονόματι WebSVN. Η εγκατάσταση ήταν αρκετά εύκολη. Η διαδικασία εγκατάστασης apache web server και ενός web application στο λειτουργικό σύστημα Gentoo Linux (που έτρεχε ο εξυπηρετητής) ήταν η εξής:

```
# emerge -av apache

# emerge -av websvn

# /etc/init.d/apache2 restart

# rc-update add apache2 default

# webapp-config -I -h cronos.teilar.gr -d websvn websvn 2.3.0
```

Οι πρώτες δύο εντολές πραγματοποιούν την εγκατάσταση, η τρίτη πραγματοποιεί την εκκίνηση της υπηρεσίας του apache, η τέταρτη προσθέτει την υπηρεσία στο default runlevel ώστε να ξεκινάει με κάθε εκκίνηση του συστήματος, και η τελευταία μεταφέρει το websvn στο virtual host cronos.teilar.gr ώστε να είναι εμφανές από τον apache web server. Ο φυλλομετρητής μας πλέον μας εμφανίζει την διεπαφή του websvn sto http://cronos.teilar.gr/websvn μαζί με τις πρόσφατες αλλαγές που γίναν.

Τελευταίο βήμα ήταν η εγκατάσταση της μονάδας mod_python στον apache web server, όπου και θα επέτρεπε στον apache να τρέξει αρχεία python ως αρχεία ιστοσελίδας και να προβάλλεται στον φυλλομετρητή το αποτέλεσμα (σε αντίθετη περίπτωση θα βλέπαμε τον κώδικα αντί για το αποτέλεσμα αυτού). Η εγκατάσταση και η ρύθμισή του συνοψίζεται στα παρακάτω απόσπασμα:

```
# emerge -av mod_python

# cat /etc/conf.d/apache2 (Προσθέτουμε το -D PYTHON)

...

APACHE2_OPTS="-D DEFAULT_VHOST -D INFO -D LANGUAGE -D SSL -D SSL_DEFAULT_VHOST -D PHP5 -D PERL -D PYTHON -D USERDIR"

...
```

85	47d 20h	tampakrap /	Change admin mail
84	47d 20h	tampakrap /	Add eclass to cron, pending lesson ids
83	47d 23h	tampakrap /	Finished cron for teilar.gr, teachers-teilar, career, noc, linuxteam, pending eclass
82	47d 23h	cephalon /	
81	48d 00h	cephalon /	
80	48d 00h	cephalon /	comment everything
79	48d 00h	cephalon /	cron for teacher_id and school_id database table
78	48d 01h	tampakrap /	Cron: added teachers' announcements Models: slightly updated database system: now we use 4 databases
77	48d 03h	tampakrap /	Add try except, now it doesn't fail for new db entries

Το subversion προσφέρει έναν αναλυτικό καταγραφέα (το παραπάνω screenshot παρουσιάζει ένα μέρος του) με όλες τις υποβολές που έχουν γίνει, καθώς και εύκολη μετάβαση σε μια προηγούμενη έκδοση του project. Κάθε υποβολή συνοδεύεται από ένα περιγραφικό μήνυμα προς τα υπόλοιπα μέλη της ομάδας ανάπτυξης-Επίσης, ένα πολύ δυνατό του σημείο είναι η υποστήριξη διαφορών στα αρχεία κώδικα ανάμεσα από δύο (οποιεσδήποτε στη γραμμή του χρόνου) υποβολές, και από οποιαδήποτε άτομα. Παράλληλα, μπορούμε εύκολα να απομονώσουμε τις αλλαγές που έγιναν σε μια συγκεκριμένη χρονική περίοδο, από συγκεκριμένους χρήστες, σε συγκεκριμένα αρχεία κτλ

/recover/views.py

18,6 → 18,15

```
                if form.is_valid():
                        new_password = ''.join([choice(string.letters + string.digits) for i in
range(8)])
                        try:
                                user = User.objects.get(username = request.POST.get('username'))
                                if not user:
                                        msg = 'Ο χρήστης δεν υπάρχει'
                                        raise

                                if user.email[-13:] == 'emptymail.com':
                                        msg = 'Ο χρήστης δεν έχει δηλώσει email'
                                        raise

                                l = ldap.initialize(settings.LDAP_URL)
                                l.simple_bind_s(settings.BIND_USER, settings.BIND_PASSWORD)
                                mod_attrs = [(ldap.MOD_DELETE, 'userPassword', None)]
```

26,7 → 35,6

```
                                l.modify_s('uid=%s,ou=teilarStudents,dc=teilar,dc=gr' %
(request.POST.get('username')), mod_attrs)
                                l.unbind_s()

                                user = User.objects.get(username = request.POST.get('username'))
                                user.set_password(new_password)
                                user.save()
```

ΚΕΦΑΛΑΙΟ 2ο: Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ PYTHON

2.1: Περιγραφή

Το Django είναι ένα ανοιχτού κώδικα υψηλού επιπεδου Web Framework γραμμένο σε python που κυκλοφορεί υπό την άδεια χρήσης BSD.

Η Python είναι μια διερμηνευόμενη (interpreted) γλώσσα προγραμματισμού υψηλού επιπέδου, που σχεδιάστηκε με έμφαση στην εύκολη ανάγνωση του κώδικα της. Η ανάπτυξή της ξεκίνησε το Δεκέμβριο του 1989 από τον Guido van Rossum στο Εθνικό Ινστιτούτο Έρευνας Πληροφορικής και Μαθηματικών της Ολλανδίας (CWI). Ο δημιουργός της εμπνεύστηκε το όνομα της από τους Monty Pythons, το γνωστό γκρουπ κωμικών από την Μεγάλη Βρετανία.

Υποστηρίζει πολλά είδη προγραμματισμού, όπως αντικειμενοστραφή, διαδικαστικό, συναρτησιακό και άλλα, και παρέχει ένα πλήρως δυναμικό σύστημα τύπων και αυτόματη διαχείριση μνήμης. Είναι μια γλώσσα προγραμματισμού γενικής χρήσης, ιδανική για κατασκευή εφαρμογών όλων των ειδών ακόμα και δυναμικών ιστοσελίδων. Η βασική της βιβλιοθήκη είναι αρκετά μεγάλη και διαθέτει εκτενή τεκμηρίωση.

Η ανάπτυξή της ακολουθεί ένα ανοικτό πρότυπο που βασίζεται στην κοινότητα των χρηστών και συντονίζεται από το Python Software Foundation. Είναι λογισμικό ανοιχτού κώδικα (open source) και εκδίδεται υπό την άδεια PSFL. Οι δύο τρέχουσες εκδόσεις της είναι η 2.6 και η 3, έχουν αρκετές διαφορές μεταξύ τους και αναπτύσσονται παράλληλα. Επίσης η έκδοση 3 δεν είναι συμβατή προς τα πίσω με τις παλαιότερες εκδόσεις καθώς έχει αλλάξει η σύνταξη και η λειτουργία πολλών εντολών και εκφράσεων. Τέλος, η Python υποστηρίζει διάφορες αρχιτεκτονικές και λειτουργικά συστήματα (Windows, Linux, διάφορα συστήματα Unix).

2.2 Το διαδραστικό κέλυφος της Python

Ένα πολύ χρήσιμο χαρακτηριστικό της Python, είναι ότι ο διερμηνέας της, είναι ταυτόχρονα και ένα διαδραστικό κέλυφος (interactive shell), κάτι σαν μια γραμμή εντολών για την Python, στην οποία μπορούμε να εισάγουμε εντολές και να δούμε απευθείας τα αποτελέσματα. Εάν εκτελέσουμε στο τερματικό του λειτουργικού μας συστήματος την

εντολή python χωρίς καμία παράμετρο, στην οθόνη μας εμφανίζεται το διαδραστικό κέλυφος του διερμηνέα, το οποίο μας παροτρύνει να εισάγουμε κάποια εντολή.

```
$ python
Python 2.5.5 (r255:77872, Apr 21 2010, 08:44:16)
[GCC 4.4.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Αυτό μπορεί να αποδειχθεί εξαιρετικά χρήσιμο σε ορισμένες περιπτώσεις, για παράδειγμα μπορούμε να εκτελέσουμε μια συνάρτηση του προγράμματος μας μεμονωμένα χωρίς να χρειαστεί να εκτελέσουμε ολόκληρο το πρόγραμμα.

2.3 Η βασική λειτουργία της Python

2.3.1 Αριθμοί και εκφράσεις

Ο διαδραστικός διερμηνέας της Python μπορεί να χρησιμοποιηθεί και ως αριθμομηχανή, για παράδειγμα:

```
>>> 10+5
15
>>> 30*25
750
```

Οι περισσότεροι αριθμητικοί τελεστές λειτουργούν όπως είναι αναμενόμενο, με εξαίρεση την διαίρεση ακεραίων, η οποία σε εκδόσεις παλαιότερες της 3 κάνει στρογγυλοποίηση στο αποτέλεσμα ώστε αυτό να αποτελεί ακέραιο αριθμό:

```
>>> 1/2
0
```

```
>>> 7/2

3
```

Για να πάρουμε το αποτέλεσμα χωρίς στρογγυλοποίηση πρέπει να χρησιμοποιήσουμε πραγματικούς αριθμούς (floats) ως εξής:

```
>>> 1.0/2.0

0.5
```

Ένας ακόμα χρήσιμος τελεστής είναι ο % που μας δίνει το υπόλοιπο διαίρεσης:

```
>>> 9 % 4

1
```

Όπως επίσης και ο τελεστής δύναμης ** :

```
>>> 3 ** 2

9
```

Αξίζει ακόμα να σημειωθεί ότι ο διερμηνέας τηρεί την προτεραιότητα πράξεων, δηλαδή οι παραστάσεις 4+8*7 και (4+8)*7 θα μας δώσουν διαφορετικό αποτέλεσμα:

```
>>> 4+8*7

60

>>> (4+8)*7

84
```

2.3.2 Μεταβλητές

Μια μεταβλητή είναι στην ουσία ένα όνομα που αναπαριστά μια τιμή. Εάν για παράδειγμα θέλουμε η μεταβλητή y να αναπαριστά την τιμή 7 θα πρέπει να της αναθέσουμε την τιμή αυτή ως εξής:

```
>>>y = 7
```

Αυτή η διαδικασία λέγεται ανάθεση τιμής και έπειτα μπορούμε να χρησιμοποιήσουμε την μεταβλητή y σε εκφράσεις:

```
>>>y*3

21
```

2.3.3 Εντολές

Με τις εντολές, σε αντίθεση με τις εκφράσεις, μπορούμε να πούμε στον διερμηνέα να κάνει κάτι, για παράδειγμα με την εντολή print μπορούμε να τυπώσουμε στην οθόνη:

```
>>> print "Hello, Cronos!!!"

Hello, Cronos!!!
```

2.3.4 Συναρτήσεις

Οι συναρτήσεις είναι μικρά υποπρογράμματα που μπορούμε να καλέσουμε για να εκτελέσουμε μια εργασία. Για παράδειγμα με την συνάρτηση pow() μπορούμε να υψώσουμε έναν αριθμό σε μία δύναμη:

```
>>> pow(2,8)

256
```

Όταν χρησιμοποιούμε μια συνάρτηση λέμε ότι την “καλούμε” ενώ οι τιμές που εσωκλείονται στην παρένθεση ονομάζονται παράμετροι της συνάρτησης. Οι συναρτήσεις μπορούν να χρησιμοποιηθούν σε εκφράσεις. Η Python διαθέτει αρκετές ενσωματωμένες συναρτήσεις και φυσικά ο χρήστης μπορεί να ορίσει τις δικές του.

Μια αρκετά χρήσιμη συνάρτηση είναι η input() με την οποία μπορούμε να διαβάσουμε μια τιμή που εισάγει ο χρήστης από το πληκτρολόγιο και να την αναθέσουμε σε κάποια μεταβλητή ή να την διαχειριστούμε όπως επιθυμούμε:

```
>>> a=input("Dose mou to a:")
```

```
Dose mou to a:30

>>> b=input("Dose mou to b:")

Dose mou to b:44

>>> print a+b

74
```

2.3.5 Modules

Τα modules είναι κάτι σαν επεκτάσεις που μπορούμε να εισάγουμε στην Python για να επεκτείνουμε της δυνατότητές της. Η βασική βιβλιοθήκη της Python διαθέτει πάρα πολλά για κάθε είδους λειτουργία και υπάρχουν ακόμα περισσότερα third-party modules που μπορούμε να εγκαταστήσουμε από διάφορες on-line πηγές. Για παράδειγμα το module math της βασικής βιβλιοθήκης διαθέτει πολλές χρήσιμες συναρτήσεις όπως την floor η οποία επιστρέφει την μικρότερη δυνατή άρτια τιμή η οποία είναι μικρότερη ή ίση με τον αριθμό που εισάγουμε ως παράμετρο της συνάρτησης. Για να εισάγουμε ένα module στην Python χρησιμοποιούμε την εντολή import και στην συνέχεια καλούμε την συνάρτηση ως εξής:

```
>>> import math

>>> math.floor(22.9)

22.0
```

Εάν θέλουμε να χρησιμοποιήσουμε μόνο μία συνάρτηση από ένα module, και δεν θέλουμε κάθε φορά να την καλούμε γράφοντας και το όνομα του module, μπορούμε να την εισάγουμε με έναν λίγο διαφορετικό τρόπο, και στην συνέχεια να την καλούμε χρησιμοποιώντας μόνο το όνομά της ως εξής:

```
>>> from math import floor

>>> floor(22.9)

22.0
```

2.3.6 Αποθήκευση και εκτέλεση των προγραμμάτων

Για να δημιουργήσουμε ένα πρόγραμμα στην Python, το οποίο θα παραμείνει στον Η/Υ μας, σε αντίθεση με όσα εισάγουμε στον διαδραστικό διερμηνέα τα οποία χάνονται μόλις τον κλείσουμε, αρκεί να γράψουμε της εντολές που θέλουμε σε κάποιον επεξεργαστή κειμένου και στην συνέχεια να τα αποθηκεύσουμε σε ένα αρχείο.

Εάν για παράδειγμα τοποθετήσουμε την εντολή print 'Hello, Cronos!!!' στο αρχείο cronos.py, τότε έχουμε ένα πρόγραμμα σε Python. Για να το εκτελέσουμε αρκεί να γράψουμε στο τερματικό του λειτουργικού μας συστήματος την εντολή python ακολουθούμενη από το όνομα του αρχείο:

```
$ python cronos.py

Hello, Cronos!!!
```

Όπως και σε άλλες γλώσσες προγραμματισμού, όταν γράφουμε μεγάλα προγράμματα χρειάζεται να προσθέσουμε κάποια σχόλια για να είναι πιο κατανοητός ο κώδικάς μας. Για να βάλουμε ένα σχόλιο στα προγράμματα python που γράφουμε αρκεί μπροστά από το σχόλιο να βάλουμε το σύμβολο της δίεσης #. Έτσι ο διερμηνέας κατά την εκτέλεση του προγράμματος θα αγνοήσει όλους του χαρακτήρες που βρίσκονται μετά από την δίεση.

Σε περιβάλλοντα UNIX, μπορούμε να κάνουμε το πρόγραμμά μας να συμπεριφέρεται σαν οποιοδήποτε κανονικό πρόγραμμα κατά την εκτέλεση του, και να μην χρειάζεται να δώσουμε την εντολή python πριν από το όνομα του για να εκτελεστεί. Αυτό γίνεται προσθέτοντας στην πρώτη γραμμή του προγράμματός μας τα σύμβολα της δίεσης και του θαυμαστικού ακολουθούμενα από την διαδρομή όπου βρίσκεται ο διερμηνέας της Python, δηλαδή #!/usr/bin/python , και στην συνέχεια δίνοντας δικαιώματα εκτέλεσης στο αρχείο μας. Έτσι μπορούμε να εκτελέσουμε για παράδειγμα το αρχείο cronos.py που δημιουργήσαμε πριν απλά πληκτρολογώντας το όνομά του, δεδομένου ότι ο τρέχον φάκελος είναι αυτός που το περιέχει:

```
$ ./cronos.py

Hello, Cronos!!!
```

2.3.7 Strings

Η Python υποστηρίζει πολλούς τύπους δεδομένων, όπως int, float, long int, list κ.α..

Ένας από του βασικότερους όμως που συναντούμε σχεδόν σε κάθε πρόγραμμα γραμμένο σε Python είναι τα strings. Η κύρια χρήσης των strings είναι η αναπαράσταση κειμένου και αποτελούν τιμές όπως και οι ακέραιοι (int). Τα strings πρέπει να εσωκλείονται σε διπλά ή μονά εισαγωγικά, για παράδειγμα 'Hello, Cronos' ή “Hello, Cronos”. Στην περίπτωση που ένα string περιέχει εισαγωγικά, τα οποία όμως χρησιμοποιούμε για να ορίσουμε που αρχίζει και που τελειώνει το string, πρέπει να χρησιμοποιήσουμε πριν από αυτά την ανάποδη πλαγιοκάθετο για να “καταλάβει” ο διερμηνέας ότι πρέπει να το τυπώσει, δηλαδή:

```
>>> 'let\'s go'

"let's go"
```

Σε αντίθετη περίπτωση ο διερμηνέας θα μας τυπώσει ένα σφάλμα:

```
>>> 'let's go'

File "<stdin>", line 1

'let's go'

^
```

2.4 Λίστες και tuples

Η Python υποστηρίζει αρκετές δομές δεδομένων, οι οποίες είναι συλλογές στοιχείων δεδομένων. Η πιο βασική από αυτές είναι η ακολουθία (sequence). Σε κάθε στοιχείο της ακολουθίας ανατίθεται ένας αριθμός-ευρετηρίου (index). Η αρίθμηση του ευρετηρίου ξεκινάει από το μηδέν, το δεύτερο στοιχείο έχει αριθμό ένα και ούτω καθεξής. Επίσης είναι δυνατή η αρίθμηση ξεκινώντας από πίσω, με το τελευταίο στοιχείο να έχει αριθμό ευρετηρίου -1, το επόμενο -2 κλπ.

Υπάρχουν έξι ενσωματωμένοι τύποι ακολουθιών με κυριότερους από αυτούς τις λίστες (lists) και τα tuples. Η βασική διαφορά τους είναι ότι μπορούμε να μεταβάλουμε τα

στοιχεία μια λίστας, να προσθέσουμε καινούρια ή να αφαιρέσουμε κάποια , αλλά δεν μπορούμε να μεταβάλουμε τα στοιχεία σε ένα tuple. Οι ακολουθίες είναι χρήσιμες όταν θέλουμε να δουλέψουμε με μια συλλογή δεδομένων. Τα στοιχεία μιας λίστας περικλείονται σε αγκύλες και χωρίζονται με κόμμα, για παράδειγμα:

```
>>> list = ['cronos', 'teilar.gr', 3]

>>> print list

['cronos', 'teilar.gr', 3]
```

Επίσης τα στοιχεία μια ακολουθίας μπορεί να είναι και τα ίδια ακολουθίες, πέρα από strings, integers, floats κτλπ.

2.4.1 Συνηθισμένες λειτουργίες των ακολουθιών

Υπάρχουν ορισμένες λειτουργίες που λειτουργούν σε όλων των ειδών τις ακολουθίες όπως: indexing, slicing, adding, multiplying και έλεγχος μέλους. Επιπλέον η Python διαθέτει αρκετές ενσωματωμένες συναρτήσεις για τον υπολογισμό του μήκους των ακολουθιών και για την εύρεση του μεγαλύτερου και του μικρότερου στοιχείου τους.

Indexing

Όπως είπαμε όλα τα στοιχεία κάθε ακολουθίας είναι αριθμημένα ξεκινώντας από το μηδέν. Έτσι μπορούμε να προσπελάσουμε κάθε στοιχείο ξεχωριστά ως εξής:

```
>>> a='CrOnoS'

>>> a[1]

'r'

>>> a[4]

'o'

>>> a[-1]
```

```
'S'
```

Τα strings αποτελούν και αυτά ακολουθίες και μπορούμε να προσπελάσουμε τα στοιχεία τους ακόμα και χωρίς την χρήση μεταβλητής:

```
>>> 'Cronos'[-2]

'o'
```

Slicing

Με το slicing (τεμαχισμός) μπορούμε να προσπελάσουμε ένα εύρος στοιχείων μια ακολουθίας χρησιμοποιώντας δύο δείκτες χωρισμένους με άνω κάτω τελεία.

```
>>> numbers=[1,2,3,4,5,6,7,8,9,10]

>>> numbers[3:6]

[4, 5, 6]
```

Το slicing είναι πολύ χρήσιμο όταν θέλουμε να αποσπάσουμε συγκεκριμένα κομμάτια από μια ακολουθία. Ο πρώτος δείκτης είναι ο αριθμός ευρετηρίου του στοιχείου που θέλουμε να συμπεριλάβουμε πρώτο στον τεμαχισμό, ενώ ο δεύτερος είναι ο αριθμός ευρετηρίου του πρώτου στοιχείου αμέσως μετά τον τεμαχισμό. Τέλος μπορούμε να ορίσουμε ακόμα έναν δείκτη ο οποίος αποτελεί το βήμα του τεμαχισμού:

```
>>> numbers[0:6]

[1, 2, 3, 4, 5, 6]

>>> numbers[0:6:2]

[1, 3, 5]
```

Adding

Με το adding (πρόσθεση) μπορούμε να συνενώσουμε ακολουθίες χρησιμοποιώντας τον τελεστή της πρόσθεσης +.

```
>>> [1,2,3]+[4,5,6]+[7,8,9]

[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Multiplication

Με το multiplication (πολλαπλασιασμός) μπορούμε να δημιουργήσουμε μία νέα ακολουθία, στην οποία η ακολουθία που πολλαπλασιάζουμε με τον τελεστή * και μία τιμή x, επαναλαμβάνεται x φορές.

```
>>> 'Cronos'*3

'CronosCronosCronos'
```

Membership

Για να ελέγξουμε εάν μία τιμή υπάρχει σε μια ακολουθία, χρησιμοποιούμε τον τελεστή “in”. Ο τελεστής αυτός μας επιστρέφει True εάν η τιμή υπάρχει και False εάν δεν υπάρχει. Τέτοιου είδους τελεστές ονομάζονται boolen τελεστές:

```
>>> students = ['nikos', 'maria', 'dimitris']

>>> 'nikos' in students

True

>>> 'Petros' in students

False
```

Μήκος, Μέγιστο και Ελάχιστο

Οι ενσωματωμένες συναρτήσεις της Python len, max και min μπορεί να φανούν αρκετά χρήσιμες όταν δουλεύουμε με ακολουθίες. Η συνάρτηση len επιστρέφει τον αριθμό των στοιχείων που υπάρχουν σε μια ακολουθία, ενώ οι max και min μας επιστρέφουν το μεγαλύτερο και το ελάχιστο στοιχείο μιας ακολουθίας αντίστοιχα:

```
>>> numbers = [200, 567, 908]
```

```
>>> len(numbers)
3
>>> max(numbers)
908
>>> min(numbers)
200
>>> max(7,8)
8
>>> min (4,2,3,1,9)
1
```

2.4.2 Λίστες

Όπως είδαμε ένας πολύ χρήσιμος τύπος ακολουθίας της Python είναι οι λίστες και η βασική τους διαφορά σε σχέση με τα tuples και τα strings είναι ότι μπορούμε να μεταβάλλουμε τα περιεχόμενά τους. Υπάρχει ακόμα η δυνατότητα να δημιουργήσουμε μία λίστα από τα στοιχεία ενός string ώστε να είναι δυνατή η διαχείρισή τους, αυτό γίνεται με τον τύπο list() ως εξής:

```
>>> list('teilar')
['t', 'e', 'i', 'l', 'a', 'r']
```

Μεταβολή των στοιχειών μιας λίστας

Η μεταβολή ενός στοιχείου μιας λίστας είναι πολύ απλή και μοιάζει με την ανάθεση τιμής σε μεταβλητή, χρησιμοποιώντας όμως και τον δείκτη ευρετηρίου ώστε να δείξουμε ποιο στοιχείο θέλουμε να μεταβάλλουμε:

```
>>> x = [3, 3, 3,]
```

```
>>> x[1] = 4

>>> x

[3, 4, 3]

>>>
```

Επίσης μπορούμε να μεταβάλουμε ένα μόνο κομμάτι της λίστα χρησιμοποιώντας slicing:

```
>>> school = list('Cronos')

>>> school

['C', 'r', 'o', 'n', 'o', 's']

>>> school[4:] = list('job')

>>> school

['C', 'r', 'o', 'n', 'j', 'o', 'b']

>>>
```

Διαγραφή των στοιχείων μιας λίστας

Για να διαγράψουμε ένα στοιχείο μιας λίστας χρησιμοποιούμε την εντολή del και φυσικά τον δείκτη ευρετηρίου:

```
>>> x

[3, 4, 3]

>>> del x[1]

>>> x

[3, 3]
```

Μέθοδοι λιστών

Οι μέθοδοι μοιάζουν συναρτήσεις οι οποίες είναι στενά συνδεδεμένες με κάποια αντικείμενο (object), όπως ένα string, μια λίστα ή οποιοδήποτε άλλο. Οι μέθοδοι καλούνται όπως τις συναρτήσεις αλλά βάζοντας μπροστά το όνομα του αντικειμένου και τελεία: object.method(argmunets)

Οι λίστες οι οποίες είναι αντικείμενα, έχουν αρκετές μεθόδους που μας επιτρέπουν να εξετάσουμε αλλά και να μεταβάλλουμε τα στοιχεία τους. Μερικές αρκετά χρήσιμες από αυτές είναι:

Η μέθοδος append χρησιμοποιείται για να προσθέσουμε ένα στοιχείο στο τέλος μιας λίστας:

```
>>> numbers = [1,2,3]

>>> numbers.append(4)

>>> numbers

[1, 2, 3, 4]
```

Η μέθοδος count μετράει πόσες φορές εμφανίζεται ένα στοιχείο σε μια λίστα:

```
>>> numbers = [1,2,3,1,4,5,1]

>>> numbers.count(1)

3
```

Η μέθοδος extend μας επιτρέπει να προσθέσουμε πολλά στοιχεία στο τέλος μιας λίστας, στην ουσία επεκτείνουμε μια λίστα με τα στοιχεία κάποιας άλλης.

```
>>> numbers = [1,2,3]

>>> a = [4,5,6]

>>> numbers.extend(a)
```

```
>>> numbers

[1, 2, 3, 4, 5, 6]
```

Η μέθοδος index αναζητεί το δείκτη ευρετηρίου της πρώτης εμφάνισης ενός στοιχείου σε μια λίστα:

```
>>> users = ['nick', 'mary', 'john']

>>> users.index('mary')

1
```

Η μέθοδος insert χρησιμοποιείται για να προσθέσουμε ένα στοιχείο σε μια συγκεκριμένη θέση μιας λίστας:

```
>>> users = ['nick', 'mary', 'john']

>>> users.insert(1, 'peter')

>>> users

['nick', 'peter', 'mary', 'john']
```

Η μέθοδος pop αφαιρεί ένα στοιχείο από μια λίστα (από ορισμού το τελευταίο) και το επιστρέφει. Είναι η μόνη μέθοδος λίστας η οποία και μεταβάλει την λίστα και επιστρέφει μια τιμή:

```
>>> users

['nick', 'peter', 'mary', 'john']

>>> users.pop()

'john'

>>> users

['nick', 'peter', 'mary']
```

```
>>> users.pop(1)

'peter'

>>> users

['nick', 'mary']
```

Η μέθοδος remove χρησιμοποιείται για την αφαίρεση της πρώτης εμφάνισης ενός αντικειμένου μιας λίστας:

```
>>> a = [2,4,5,2,7]

>>> a.remove(2)

>>> a

[4, 5, 2, 7]
```

Η μέθοδος sort ταξινομεί τα στοιχεία μιας λίστας, μεταβάλλοντας την ίδια την λίστα:

```
>>> a = [8, 3, 12, 1, 9, 25]

>>> a.sort()

>>> a

[1, 3, 8, 9, 12, 25]
```

2.4.3 Tuples

Τα tuples είναι ακολουθίες, όπως και οι λίστες, με την διαφορά ότι δεν μπορούμε να τις μεταβάλουμε. Η σύνταξη ενός tuple είναι πολύ απλή, αρκεί να χωρίσουμε μερικές τιμές με κόμμα και έχουμε ένα tuple. Προαιρετικά οι τιμές αυτές μπορούν να εσωκλείονται σε παρενθέσεις:

```
>>> 1,2,3

(1, 2, 3)
```

```
>>> (1,2,3)

(1, 2, 3)
```

Η ενσωματωμένη συνάρτηση tuple(), παίρνει ως παράμετρο μια ακολουθία και την μετατρέπει σε tuple. Εάν η ακολουθία είναι ήδη tuple επιστρέφεται ως έχει:

```
>>> tuple([1,2,3,4,5])

(1, 2, 3, 4, 5)
```

Τα tuples είναι απλές δομές δεδομένων και το μόνο που μπορούμε να κάνουμε είναι να τις δημιουργήσουμε και να προσπελάσουμε τα στοιχεία τους, με τον ίδιο τρόπο που το κάνουμε σε άλλες ακολουθίες και ο βασικός λόγος ύπαρξής τους είναι ότι επιστρέφονται από ορισμένες ενσωματωμένες συναρτήσεις της Python.

2.5 Λεξικά

Τα λεξικά (dictionaries) είναι μια δομή δεδομένων της Python, στα στοιχεία της οποίας μπορούμε να αναφερόμαστε ονομαστικά. Τα στοιχεία ενός λεξικού δεν έχουν κάποια συγκεκριμένη σειρά, αλλά είναι αποθηκευμένα με βάση ένα κλειδί (όνομα), το οποίο μπορεί να είναι ένα string, ένας αριθμός ή ακόμα και ένα tuple.

Τα λεξικά απαρτίζονται από ζευγάρια κλειδιών και τιμών. Κάθε κλειδί χωρίζεται από την τιμή του με άνω κάτω τελεία, και όλα τα ζευγάρια εσωκλείονται σε αγκύλες {}. Στο παρακάτω παράδειγμα ενός τηλεφωνικού καταλόγου, τα ονόματα αποτελούν τα κλειδιά του λεξικού ενώ τα τηλέφωνα τις τιμές του:

```
>>> katalogos = {'nikos':'213123', 'maria':'234409', 'petros':'431214'}

>>> katalogos

{'petros': '431214', 'nikos': '213123', 'maria': '234409'}
```

Η αναζήτηση μιας τιμής γίνεται ως εξής:

```
>>> katalogos['maria']
```

```
'234409'
```

Για την κατασκευή ενός λεξικού από άλλες ακολουθίες (κλειδιών, τιμών) ή ακόμα και από άλλα λεξικά υπάρχει η ενσωματωμένη συνάρτηση dict():

```
>>> pairs = [('onoma', 'Nikos'), ('hlikia','56')]
>>> d = dict(pairs)
>>> d
{'hlikia': '56', 'onoma': 'Nikos'}
>>> d['onoma']
'Nikos'
```

Επίσης η συνάρτηση dict() μπορεί να χρησιμοποιηθεί απευθείας περνώντας της ως παραμέτρους τα ζευγάρια κλειδιών-τιμών:

```
>>> d = dict(onoma='Nikos', hlikia='56')
>>> d
{'hlikia': '56', 'onoma': 'Nikos'}
```

2.5.1 Βασικές λειτουργίες των λεξικών

Η βασική λειτουργία των λεξικών σε πολλές περιπτώσεις είναι ίδια με αυτή των ακολουθιών:

len(d) επιστρέφει τον αριθμό των ζευγαριών που υπάρχουν στο λεξικό d.

d[k] επιστρέφει την τιμή που ανταποκρίνεται στο κλειδί k.

d[k] = v συσχετισμός της τιμής v με το κλειδί k.

del d[k] διαγραφή του ζευγαριού με κλειδί k.

k in d ελεγχος εάν υπάρχει ζευγάρι στο λεξικό d με κλειδί k.

Παρόλο που οι λίστες και τα λεξικά έχουν αρκετά κοινά χαρακτηριστικά, έχουν και ορισμένες σημαντικές διαφορές:

Τύποι κλειδιών: Τα κλειδιά ενός λεξικού δεν χρειάζεται να είναι απαραιτήτως ακέραιοι αριθμοί. Μπορεί να είναι πραγματικοί αριθμοί, strings ή ακόμα και tuples.

Αυτόματη προσθήκη: Μπορούμε να αποδώσουμε μια τιμή σε ένα κλειδί ακόμα και αν αυτό το κλειδί δεν υπάρχει στο λεξικό, έτσι θα δημιουργηθεί αυτόματα ένα νέο ζευγάρι. Αντίθετα σε μια λίστα αυτό δεν είναι δυνατό και θα πρέπει να χρησιμοποιήσουμε την μέθοδο append.

Έλεγχος μέλους: Η έκφραση k in d (όπου d ένα λεξικό) ψάχνει για κλειδί και όχι για τιμή.

Από την άλλη η έκφραση v in l (όπου l μια λίστα) ψάχνει για τιμή και όχι για αριθμό ευρετηρίου.

2.5.2 Μέθοδοι λεξικών

Τα λεξικά έχουν κάποιες χρήσιμες μεθόδους που αξίζει να αναφερθούν, καθώς καθιστούν την διαχείρισή τους αρκετά εύκολη:

Η μέθοδος clear, διαγράφει όλα τα στοιχεία ενός λεξικού και δεν επιστρέφει τίποτα:

```
>>> katalogos = {'nikos':'213123', 'maria':'234409', 'petros':'431214'}
>>> katalogos
{'petros': '431214', 'nikos': '213123', 'maria': '234409'}
>>> katalogos.clear()
>>> katalogos
{}
```

Η μέθοδος copy επιστρέφει ένα λεξικό με τα ίδια ζευγάρια κλειδιών-τιμών. Το λεξικό αυτό είναι ένα αντίγραφο του πρώτου άλλα οι τιμές του είναι οι ίδιες με το πρώτο και όχι αντίγραφα:

```
>>> katalogos = {'nikos':'213123', 'maria':'234409', 'petros':'431214'}

>>> y = katalogos.copy()

>>> y

{'maria': '234409', 'nikos': '213123', 'petros': '431214'}
```

Στο παραπάνω παράδειγμα, εάν αντικαταστήσουμε μια τιμή στο λεξικό y, στο λεξικό katalogos δεν γίνεται καμία αλλαγή. Εάν όμως επεξεργαστούμε μία τιμή στο y χωρίς να την αντικαταστήσουμε, η αλλαγή αυτή αντικατοπτρίζεται και στο λεξικό katalogos μιας και πρόκειται για την ίδια τιμή και όχι αντίγραφό της. Εάν θέλουμε να δημιουργήσουμε τελείως ανεξάρτητα αντίγραφα μπορούμε να χρησιμοποιήσουμε την συνάρτηση deepcopy, από το module copy.

Η μέθοδος items επιστρέφει όλα τα στοιχεία ενός λεξικού σε μια λίστα, στην οποία τα στοιχεία είναι της μορφής (κλειδί, τιμή). Τα στοιχεία δεν επιστρέφονται με κάποια συγκεκριμένη σειρά:

```
>>> katalogos = {'nikos':'213123', 'maria':'234409', 'petros':'431214'}

>>> katalogos.items()

[('petros', '431214'), ('nikos', '213123'), ('maria', '234409')]
```

Η μέθοδος keys επιστρέφει όλα τα κλειδιά ενός λεξικού σε μια λίστα:

```
>>> katalogos.keys()

['petros', 'nikos', 'maria']
```

Η μέθοδος pop, επιστρέφει την τιμή ενός συγκεκριμένου κλειδιού, και στην συνέχεια αφαιρεί ολόκληρο το ζευγάρι από την λίστα.

```
>>> katalogos.pop('nikos')

'213123'
```

```
>>> katalogos

{'petros': '431214', 'maria': '234409'}
```

Η μέθοδος popitem μοιάζει με την μέθοδο pop των λιστών, παρόλα αυτά επειδή τα λεξικά δεν έχουν τελευταίο αντικείμενο, η συνάρτηση επιστρέφει αυθαίρετα ένα τυχαίο στοιχείο του λεξικού και στην συνέχεια το διαγράφει:

```
>>> katalogos = {'nikos':'213123', 'maria':'234409', 'petros':'431214'}

>>> katalogos.popitem()

('petros', '431214')

>>> katalogos

{'nikos': '213123', 'maria': '234409'}
```

Η μέθοδος update ενημερώνει τα στοιχεία ενός λεξικού με τα στοιχεία ενός άλλου:

```
>>> katalogos = {'nikos':'213123', 'maria':'234409', 'petros':'431214'}

>>> x = {'giorgos':'098123', 'dimitra':'434353'}

>>> katalogos.update(x)

>>> katalogos

{'dimitra': '434353', 'petros': '431214', 'giorgos': '098123', 'nikos': '213123', 'maria': '234409'}
```

Η μέθοδος values επιστρέφει μια λίστα με όλες τις τιμές ενός λεξικού:

```
>>> katalogos.values()

['434353', '431214', '098123', '213123', '234409']
```

2.6 Συνθήκες και βρόχοι

2.6.1 Στοιχειοθέτηση και Blocks κωδικά

Τα blocks είναι σύνολα εντολών που εκτελούνται εάν μια συνθήκη είναι αληθής ή εκτελούνται πολλές φορές (βρόχοι). Σε αντίθεση με τις περισσότερες γλώσσες προγραμματισμού που χρησιμοποιούν κάποιους χαρακτήρες όπως αγκύλες "{}" για την οριοθέτηση των blocks, η Python χρησιμοποιεί την στοιχειοθέτηση (indentation), δηλαδή την προσθήκη κενών ή tabs μπροστά από κάθε γραμμή κώδικα. Όλες οι γραμμές κώδικα που ανήκουν στο ίδιο block πρέπει να έχουν ίσο αριθμό κενών ή tabs.

2.6.2 Εκτέλεση υπό συνθήκες και η εντολή if

Η εντολή if επιτρέπει την εκτέλεση κώδικα υπό συνθήκες. Η σύνταξή της είναι ως εξής:

```
if συνθήκη:

εντολή 1

εντολή 2

εντολή 1 εκτός block
```

Στην περίπτωση που η συνθήκη, δηλαδή η έκφραση πριν την άνω κάτω τελεία, είναι αληθής θα εκτελεστούν οι εντολές 1 και 2 με την σειρά, η οποίες ανήκουν στο ίδιο block. Στην συνέχεια θα συνεχιστεί η εκτέλεση των εντολών εκτός του block. Φυσικά στην περίπτωση που η συνθήκη δεν ισχύει οι εντολές εντός του block δεν θα εκτελεστούν.

Εάν θέλουμε να εισάγουμε ακόμα μία συνθήκη και να εκτελέσουμε κάποιο block στην περίπτωση που ισχύει μπορούμε να χρησιμοποιήσουμε την εντολή elif, της οποίας η σύνταξη είναι ίδια με αυτή της if. Τέλος εάν θέλουμε να εκτελέσουμε κάποιο block στην περίπτωση που δεν ισχύει καμία από τις προηγούμενες συνθήκες, δηλαδή σε κάθε περίπτωση, χρησιμοποιούμε την εντολή else.

```
>>> a = input('Δώσε έναν αριθμό: ')

Δώσε έναν αριθμό: 15

>>> if a > 0:
```

```
... print 'Θετικός Αριθμός'
... elif a < 0:
... print 'Αρνητικό Αριθμός'
... else:
... print 'Μηδέν'
...
Θετικός Αριθμός
```

Στο παραπάνω παράδειγμα ο χρήστης εισάγει έναν αριθμό. Στην συνέχεια το πρόγραμμα ελέγχει εάν ο αριθμός είναι μεγαλύτερος από το μηδέν οπότε και τυπώνει στην οθόνη 'Θετικός Αριθμός', εάν αυτό δεν ισχύει ελέγχει εάν είναι μικρότερος από το μηδέν οπότε τυπώνει 'Αρνητικό Αριθμός' και τέλος, εάν δεν ισχύει καμία από τις παραπάνω συνθήκες τυπώνει 'Μηδέν'.

Πρέπει να σημειωθεί ότι οι εντολές elif και else δεν μπορούν να χρησιμοποιηθούν μόνες τους χωρίς να έχει προηγηθεί μια εντολή if. Ακόμα η εντολή if μπορεί να χρησιμοποιηθεί και μέσα στο block κώδικα μιας άλλης εντολής if. Τότε το block κώδικα της δεύτερης εντολής if ονομάζεται εμφωλευμένο (nested) block.

2.6.3 Βρόχοι

Με τους βρόχους μπορούμε να επαναλαμβάνουμε μια διαδικασία, δηλαδή την εκτέλεση ενός block κώδικα, πολλές φορές ή όσο μια συνθήκη είναι αληθής.

Ο βρόχος while

Με τον βρόχο while μια διαδικασία επαναλαμβάνεται όσο μια συνθήκη είναι αληθής. Για παράδειγμα μπορούμε να τυπώσουμε τους αριθμούς από το 1 έως το 100 χωρίς να χρειάζεται να γράψουμε 100 διαφορετικές εντολές print:

```
>>> while x <= 100:
```

```
... print x
... x +=1
...
1
2
…
99
100
```

Οι εντολή print x που τυπώνει τον αριθμό και η εντολή x+=1 που αυξάνει τον αριθμό κατά 1 επαναλαμβάνονται μέχρι η τιμή του x να γίνει 100, όσο δηλαδή η συνθήκη x<= 100 είναι αληθής.

Ο βρόχος for

Με τον βρόχο for μπορούμε επαναλαμβάνουμε μια διαδικασία για ένα σύνολο στοιχείων, για παράδειγμα για όλα τα στοιχεία μιας λίστας:

```
>>> names = ['nikos', 'maria', 'giorgos', 'makis']
>>> for name in names:
... print name
...
nikos
maria
giorgos
```

```
makis
```

Εάν θέλουμε για οποιοδήποτε λόγο να διακόψουμε την εκτέλεση ενός βρόχου, χρησιμοποιούμε την εντολή break. Η εντολή αυτή “σπάει” το βρόχο και συνεχίζεται η εκτέλεση των υπόλοιπων εντολών του προγράμματος. εάν αυτές υπάρχουν.

2.7 Δημιουργία συναρτήσεων

Η δημιουργία μιας συνάρτησης γίνεται με την εντολή def, και στην συνέχεια η συνάρτηση μπορεί να χρησιμοποιηθεί ακριβώς όπως τις ενσωματωμένες συναρτήσεις της Python:

```
>>> def hello(name):

...

...

>>> print hello('Cronos')

Hello, Cronos!
```

Με την εντολή return η συνάρτηση επιστρέφει το αποτέλεσμα όταν την καλέσουμε. Σε ορισμένες περιπτώσεις οι συναρτήσεις δεν επιστρέφουν τίποτα, για παράδειγμα μπορεί return 'Hello, ' + name + '!' απλά να τυπώνουν στην οθόνη, αυτές οι συναρτήσεις δεν έχουν στο block τους την εντολή return ή εάν την έχουν δεν επιστρέφει τίποτα.

Στο παραπάνω παράδειγμα το name αποτελεί παράμετρο της συνάρτησης. Επίσης οι μεταβλητές και οι παράμετροι της συνάρτησης μπορούν να χρησιμοποιηθούν μόνο μέσα σε αυτήν, εκτός αν οριστούν ως global.

2.8 Δημιουργία τάξεων

Οι τάξεις (classes) είναι ένα είδος αντικειμένου, και όλα τα αντικείμενα που ανήκουν σε μια τάξη αποτελούν ένα στιγμιότυπο αυτής. Η δημιουργία μιας τάξης γίνεται με την εντολή class:

```
class Person:

def setName(self, name):

    self.name = name

def getName(self):

    return self.name

def greet(self):

    print "Hello, world! I'm %s." % self.name
```

Στο παραπάνω παράδειγμα δημιουργούμε την τάξη Person, οι setName(), getName() και greet() αποτελούν τις μεθόδους της συνάρτησης. Οι μέθοδοι μοιάζουν με συναρτήσεις και ορίζονται με τον ίδιο τρόπο αλλά μέσα στο block κώδικα της τάξης. Η βασική διαφορά με τις συναρτήσεις είναι η παράμετρος self η οποία παίρνει την τιμή της από το στιγμιότυπο της τάξης κάθε φορά, οπότε ο χρήστης δεν χρειάζεται να την ορίσει.

2.9 Εξαιρέσεις

Κατά τη συγγραφή προγραμμάτων είναι δυνατό να συναντήσουμε κάποιες περίεργες ή εξαιρετικές καταστάσεις, όπως λάθη ή καταστάσεις που δεν περιμένουμε να συμβούν αρκετά συχνά, τις εξαιρέσεις (exceptions). Η διαχείριση τους στην Python γίνεται με τα exception αντικείμενα. Όταν συμβεί κάτι τέτοιο η Python κάνει raise ένα exception:

```
>>> 1/0

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

ZeroDivisionError: integer division or modulo by zero
```

Στο παραπάνω παράδειγμα γίνεται raise ένα exception το οποίο είναι στιγμιότυπο της τάξης ZeroDivisionError. Η Python διαθέτει πολλές ενσωματωμένες εξαιρέσεις και υπάρχει η δυνατότητα ο χρήστης να ορίσει και δικές του. Το ενδιαφέρον με τις εξαιρέσεις είναι ότι

μπορούμε να τις διαχειριστούμε και να ενεργήσουμε ανάλογα όταν αυτές συμβούν, αυτό γίνεται με τις εντολές try και except:

```
try:
    x = input('Enter the first number: ')
    y = input('Enter the second number: ')
    print x/y
except ZeroDivisionError:
    print "The second number can't be zero!"
```

Στο παραπάνω παράδειγμα εάν ο χρήστης δώσει τον δεύτερο αριθμό μηδέν, η Python θα τυπώσει το μήνυμα

'The second number can't be zero' αντί της εξαίρεσης και του σφάλματος. Ακόμα υπάρχει η δυνατότητα να χρησιμοποιήσουμε περισσότερες από μία εντολές except μέσα στην ίδια try ή να ορίσουμε περισσότερες από μία εξαιρέσεις με μία εντολή except.

ΚΕΦΑΛΑΙΟ 3ο: ΔΗΜΙΟΥΡΓΙΑ ΤΩΝ CRON JOBS

3.1 Third-Party Django/Python Modules

Όπως αναφέραμε και στο κεφαλαιο 2 τα modules στην python είναι πολύ χρήσιμα γιατί δεν χρειάζεται να ανακαλύπτουμε τον τροχό κάθε φορα. Έτσι και εμείς για την πτυχιακή μας χρειαστήκαμε αρκετά third-party modules. Τα ποιο σημαντικά είναι:

- PycURL
- BeautifulSoup
- PyCrypto
- Python-LDAP
- LDAP_Groups

3.1.1 >>> import pycurl

Το PycURL αποτελεί μια Python διεπαφή για την βιβλιοθήκη libcurl. Το PycURL μπορεί να χρησιμοποιηθεί ώστε να πάρει αντικείμενα από το διαδίκτυο από ένα συγκεκριμένο υπερσύνδεσμο.

Η libcurl είναι μια δωρεάν και εύκολη στην χρήση βιβλιοθήκη που λειτουργεί σαν φυλλομετρητής αλλα με υποστήριξη πολλών πρωτόκολλων. Υποστηρίζει FTP, FTPS, HTTP, HTTPS, SCP, SFTP, TFTP, TELNET, DICT, LDAP, LDAPS, FILE, IMAP, SMTP, POP3 και RTSP. Επίσης υποστηρίζει SSL πιστοποιητικά, HTTP POST, HTTP PUT, μεταφόρτωση FTP, μεταφόρτωση HTTP μέσω φόρμας, proxies, cookies, πιστοποίηση μέσω όνοματος χρήστη και κωδικού πρόσβασης και άλλα.

Η PycURL ήταν ένα από τα βασικότερα εργαλεία μας για την ολοκλήρωση τις πτυχιακής μας. Έπρεπε με κάποιον τρόπο να καταφέρουμε να πάρουμε δεδομένα από διάφορες πλατφόρμες του ΤΕΙ στις οποίες οι φοιτητές έχουν πρόσβαση μεμονωμένα. Αυτό έπρεπε να πραγματοποιηθεί χωρίς εμείς να έχουμε άμεση πρόσβαση στους εξυπηρετητές που τα φιλοξενούν ή στις βάσεις δεδομένων των υπηρεσιών αυτών. Έπρεπε απλά να τραβήξουμε

τα δεδομένα που θα ήταν απαραίτητα να προβάλουμε στους φοιτητές μόνο με τα στοιχεία που θα μας παρέχουν οι ίδιοι, όπως για παράδειγμα όνομα χρηστη και κωδικό πρόσβασης από το dionysos.teilar.gr.

Αυτή ακριβώς είναι και η λειτουργία της PycURL. Ο προγραμματιστής μπορεί με την κατάλληλη σύνταξη να τραβήξει από το "διαδίκτυο" οποιαδήποτε πληροφορία. Ο κώδικας που ακολουθεί είναι ένα μικρό παράδειγμα για το πώς με την χρήση Python και PycURL παίρνουμε τα αποτελέσματα από την αναζήτηση τις βιβλιοθήκης του ΤΕΙ Λάρισας.

```
link='http://hermes.lib.teilar.gr/ipac20/ipac.jsp?
session=A26772NR74250.24315&menu=search&aspect=subtab22&npp=40&ipp=20&spp=
20&profile=multbl--1&ri=&term=%s&index=.GEN&x=0&y=0&aspect=subtab22' %
(str(request.GET.get('search')))

b = StringIO.StringIO()

conn = pycurl.Curl()

conn.setopt(pycurl.URL, link)

conn.setopt(pycurl.WRITEFUNCTION, b.write)

conn.perform()

output = unicode(b.getvalue(), 'utf-8', 'ignore')
```

Σύντομη εξήγηση του παραπάνω αποσπάσματος:

Στην πρώτη γραμμή δημιουργούμε το URL στο οποίο θα εισέλθει η PycURL, το οποίο είναι δυναμικά οριζόμενο συναρτήσει του στοιχείου αναζήτησης που δίνεται από τον επισκέπτη (request.GET.get('search'))

Στην δεύτερη γραμμή ορίζουμε έναν Input/Output Buffer, ο οποίος θα αποθηκεύσει το αποτέλεσμα της σύνδεσης (δηλαδή την παραγώμενη HTML ιστοσελίδα μετά τη σύνδεση). Στην τρίτη γραμμή ορίζουμε ένα αντικείμενο τύπου pycurl το οποίο θα πραγματοποιήσει τη σύνδεση, και στις επόμενες δύο γραμμές χρησιμοποιούμε συναρτήσεις πάνω σε αυτό το αντικείμενο όπου δηλώνουμε το URL που θα πρέπει να εισέλθει και τον buffer που θα πρέπει

να γραφτεί το αποτέλεσμα (τον οποίο δηλώσαμε προηγουμένως). Στις δύο τελευταίες γραμμές πραγματοποιούμε τη σύνδεση και γράφουμε το αποτέλεσμα στη μεταβλητή output αφού κωδικοποιήσουμε το αποτέλεσμα σε UTF-8. Το παραγόμενο αποτέλεσμα είναι μια HTML σελίδα.

3.1.2 >>> import BeautifulSoup

Με την βοήθεια του PycURL καταφέραμε να πάρουμε την ιστοσελίδα που μας ενδιαφέρει σε μορφή HTML. Πρέπει όμως από τον HTML κώδικα να πάρουμε συγκεκριμένα δεδομένα, όπως στο παραπάνω παράδειγμα παίρνουμε με την χρήση της PycURL τα αποτελέσματα τις αναζήτησης στην βιβλιοθήκη του TEI σε μορφή HTML. Για να πάρουμε από τον HTML κώδικα μονο το κομμάτι που μας ενδιαφέρει θα χρησιμοποιήσουμε τον BeautifulSoup HTML/XML parser.

Το BeautifulSoup είναι ένας Python HTML/XML parser ο οποιος παίρνει ως είσοδο τον HTML κώδικα και επιτρέπει με την χρήση κατάλληλων μεθόδων να απομονώσουμε μεμονωμένα κομμάτια του κώδικα. Μπορούμε δηλαδή να πάρουμε συγκεκριμένες HTML ετικέτες ή ακόμη και τα περιεχόμενα των ετικετών (ή και τα περιεχόμενα των περιεχομένων των ετικετών...). Μερικά παραδείγματα συναρτήσεων:

findAll('a', 'bold'): Βρίσκει όλες τις ετικέτες τύπου <a class='bold'> και τις επιστρέφει σε μια λίστα (python array), το οποίο είναι πολύ χρήσιμο καθώς τα αποτελέσματα μπορούν εύκολα να κρατηθούν σε μεταβλητή και να αξιοποιηθούν κατάλληλα. Αν το a αντικατασταθεί με οποιαδήποτε άλλη έγκυρη ετικέτα HTML όπως td, table,

.find('a', 'bold'): Όπως η προηγούμενη, αλλά επιστρέφει μόνο το πρώτο αποτέλεσμα σε αλφαριθμητικό αντί για λίστα

.contents: Μια λίστα με όλα τα περιεχόμενα της ετικέτας. Η λίστα έχει ως στοιχείο κάθε υπο-ετικέτα

.a .b .span .title .head: Κάθε μία από αυτές τις συναρτήσεις όταν κληθεί σε αντικείμενο τύπου BeautifulSoup επιστρέφει την πρώτη ετικέτα που το όνομά της αντιστοιχεί με το όνομα της συνάρτησης

.pretiffy(): Μία πολύ χρήσιμη συνάρτηση η οποία τακτοποιεί τον παραγόμενο HTML

κώδικα, ώστε είτε να εκτυπωθεί, είτε να μας βοηθήσει να υπολογίσουμε τα στοιχεία που μας ενδιαφέρουν.

Παράδειγμα χρήσης της BeautifulSoup στην εργασία μας:

```
1   soup = BeautifulSoup(output).findAll('table')[24]
2   results = []
3   i = 4
4   k = 0
5   for item in (soup.findAll('a', 'mediumBoldAnchor')):
6       title = str(soup.findAll('td')[i].contents[0].contents[0])
7       try:
8           author = ''.join(str(soup.findAll('td')[i+1].contents[0].contents[1:]))[4:-5]
9           editor = str(soup.findAll('td')[i+2].contents[0].contents[0]).split(' : ')[1]
10          city = str(soup.findAll('td')[i+2].contents[0].contents[0]).split(' : ')[0]
11      except:
12          pass
13      i += 10
14      results.append([title, author, editor, city])
15  if (results == []):
16      msg = 'Δεν υπάρχουν αποτελέσματα'
```

Ο παραπάνω κώδικας είναι συνέχεια του παραπάνω αποσπάσματος χρήσης της PycURL. Εδώ, στην πρώτη γραμμή ορίζουμε το αντικείμενο soup τύπου BeautifulSoup και ταυτόχρονα καλούμε τη συνάρτηση findAll, η οποία με την παράμετρο table επιστρέφει σε

ένα python array όλα τα <table> tags από το παραγόμενο output. Εμείς αποθηκεύουμε μόνο το 24ο στοιχείο του array, το οποίο παράλληλα είναι και το table με τα αποτελέσματα των βιβλίων που μας ενδιαφέρει.

Στην συνέχεια εκτελούμε έναν βρόγχο για κάθε <a> ετικέτα που βρίσκουμε με όνομα CSS κλάσης mediumBoldAnchor. Από κάθε επανάληψη του βρόγχου θα πάρουμε τον τίτλο του βιβλίου που μας έχει επιστραφεί σαν αποτέλεσμα, ο οποίος βρίσκεται στο περιεχόμενο τις ετικέτας <td>. Με την ίδια λογική συνεχίζουμε και παίρνουμε τον εκδοτικό οίκο, τον συγγραφέα και την πόλη. Τα αποτελέσματα αποθηκεύονται σε 2D Array με όνομα results.

Γενικά βλέπουμε πως ο συνδυασμός PycURL και BeautifulSoup μας επιστρέφει δεδομένα που θα κοπιάζαμε πολύ να πάρουμε χωρίς την ύπαρξη τους.

3.1.3 >>> import PyCrypto

Το PyCrypto (Python Cryptograpic Toolkit – Κουτί εργαλείων κρυπτογράφησης) αποτελεί μία εξαιρετικά χρήσιμη βιβλιοθήκη για την κρυπτογράφηση ή την κωδικοποίηση δεδομένων. Η διαφορά των δύο εννοιών είναι τεράστια, και στην περίπτωση της πτυχιακής μας έπρεπε να είναι ξεκάθαρη, καθώς και οι δύο θα μας ήταν ιδιαίτερα χρήσιμες για την εξασφάλιση της προστασίας των δεδομένων των εγγεγραμμένων χρηστών.

Η κρυπτογράφηση είναι η διαδικασία απόκρυψης ενός κειμένου με τη χρήση ενός αλγορίθμου και ενός μυστικού κλειδιού. Το μυστικό κλειδί στην ουσία είναι ένα σύνολο τυχαίων χαρακτήρων, όπως ένα συνθηματικό. Ο αλγόριθμος χρησιμοποιεί το μυστικό κλειδί και παράγει ένα νέο κείμενο το οποίο αποτελείται από φαινομενικά τυχαίους χαρακτήρες. Η αντίστροφη διαδικασία ονομάζεται αποκρυπτογράφηση και μπορεί να παράγει το αρχικό κείμενο από το παραγόμενο, μόνο με τη χρήση του ίδου μυστικού κλειδιού.

Για την αποτελεσματικότητα της απόκρυψης και προστασίας των δεδομένων θα πρέπει το μυστικό κλειδί να αποτελείται από πολλούς χαρακτήρες, να περιέχει σύμβολα, αριθμούς και κεφαλαία/μικρά γράμματα σε διάφορους συνδιασμούς, να είναι καλά φυλαγμένο στον εξυπηρετητή, και προστατευμένο με τα κατάλληλα δικαιώματα (για παράδειγμα δικαίωμα εγγραφής / ανάγνωσης μόνο στον υπερχρήστη) και ο αλγόριθμος να είναι επίσης δυνατός, ώστε να παράγει αποτέλεσμα που να είναι αδύνατη η αποκρυπτογράφηση. Για τις ανάγκες της πτυχιακής χρησιμοποιήθηκε ο αλγόριθμος Blowfish

που είναι ιδιαίτερα διαδεδομένος και αποτελεσματικός.

Ακολουθεί παράδειγμα υλοποίησης κρυπτογράφησης και αποκρυπτογράφησης με τον αλγόριθμο Blowfish σε python:

```
>>> from Crypto.Cipher import Blowfish
>>> obj = Blowfish.new('%$s3cr37@k3y#')
>>> password = ('th1$|zAp4$$w0rD')
>>> result = obj.encrypt(password)
>>> print result
ZLMO%b cT
 
>>> print obj.decrypt(result)
th1$|zAp4$$w0rD!
```

Δημιουργήσαμε ένα αντικείμενο τύπου Blowfish, στο οποίο δηλώνουμε ποιο θα είναι το μυστικό του κλειδί, στη συνέχεια δηλώνουμε το συνθηματικό στη μεταβλητή password και χρησιμοποιούμε τις συναρτήσεις encrypt και decrypt για να πάρουμε τα αποτέλεσματα και παρατηρούμε ότι το decrypt μας δίνει το αρχικό μας συνθηματικό.

Η κωδικοποίηση είναι η διαδικασία απόκρυψης ενός κειμένου σε ένα δεύτερο, με τη βοήθεια πάλι ενός μυστικού κλειδιού. Η διαφορά τους έγκειται στο γεγονός ότι το μυστικό κλειδί (το οποίο ονομάζεται Salt) είναι τώρα ένα τυχαίο σύνολο χαρακτήρων και όχι προκαθορισμένο, και είναι εξαιρετικά δύσκολη η αντίστροφη διαδικασία. Με αυτό τον τρόπο, ένα συνθηματικό κωδικοποιείται και αποθηκεύεται στη βάση δεδομένων μαζί με το μυστικό κλειδί, και για να επαληθευτεί η ταυτότητα του χρήστη, το σύστημα ζητάει από τον ίδιο το χρήστη να εισάγει το συνθηματικό του, το οποίο κωδικοποιείται με το ίδιο κλειδί και επαληθεύονται τα δύο αποτελέσματα (το αποθηκευμένο και το παραγόμενο). Για την

πτυχιακή χρησιμοποιήθηκε ο αλγόριθμος SHA1 (αρχικά από το Secure Hash Algorithm 1), ο οποίος πρόσφατα αντικαταστάθηκε από τον SHA2. Ακολουθεί παράδειγμα σε python:

```
>>> import hashlib
>>> import os
>>> h = hashlib.sha1('th1$|zAP4$$!')
>>> h.update(salt)
>>> h.hexdigest()
'dd90757e333c9d9f67573533447487dd5e51e217'
```

Το αποτέλεσμα που πρέπει να επιστραφεί διαφέρει ανάλογα με το σύστημα αποθήκευσης. Για παράδειγμα, το django κρατάει το αποτέλεσμα στη μορφή {SSHA} $SALT$HASH. Η βιβλιοθήκη hashlib που χρησιμοποιήθηκε υπάρχει μέσα στην python.

Η ασφάλεια των δεδομένων ήταν και ένα θέμα που απασχόλησε άμεσα την πτυχιακή εργασία μας, μιας και οι χρήστες κατά την εγγραφή μας παραδίδουν ονόματα χρήστη και συνθηματικά. Στο κεφάλαιο 7 αναφέρουμε εκτενέστερα όλα τα μέτρα που λάβαμε ώστε να εξασφαλιστεί η προστασία των δεδομένων, καθώς και πως οι παραπάνω απλές συναρτήσεις έπαιξαν καθοριστικό ρόλο στον τομέα αυτό.

3.1.4 >>> import LDAP-Python

Όπως προαναφέραμε, η αποθήκευση των δεδομένων γίνεται σε LDAP server και γι αυτό μας ήταν απαραίτητη μια βιβλιοθήκη η οποία θα αναλάμβανε να πραγματοποιεί τις παρακάτω διαδικασίες:

- Σύνδεση με τον LDAP server
- Αναζήτηση στις εγγραφές και επιστροφή αποτελεσμάτων σε επεξεργάσιμη μορφή
- Εισαγωγή νέας εγγραφής / νέας τιμής σε υπάρχουσα εγγραφή

- Επεξεργασία δεδομένων του LDAP
- Διαγραφή εγγραφής / τιμής σε υπάρχουσα εγγραφή

Η εν λόγω βιβλιοθήκη πραγματοποιούσε όλες τις παραπάνω διεργασίες με απόλυτη επιτυχία, και μας επέτρεπε να γράψουμε τις δικιές μας συναρτήσεις ώστε να έχουμε πλήρη πρόσβαση στον LDAP μέσα από την εφαρμογή Django που γράψαμε, και όχι μόνο από καθαρή python. Ακολουθούν μερικά παραδείγματα που πραγματοποιούν όλα τα παραπάνω βήματα:

Σύνδεση

```
>>> import ldap
>>> l = ldap.initialize('ldap://localhost')
>>> username = 'uid=root,dc=teilar,dc=gr'
>>> password = 'secret'
>>> l.simple_bind_s(username, password)
```

Αν η σύνδεση αποτύχει το παραπάνω απόσπασμα θα επιστρέψει exception το οποίο εύκολα με ένα try-except block μπορούμε να εκμεταλευτούμε.

Αναζήτηση

```
>>> l.search_s('ou=teilarStudents,dc=teilar,dc=gr', ldap.SCOPE_SUBTREE, 'uid=*',
['dionysosUsername', 'registrationNumber']
```

Στο παραπάνω απόσπασμα, το πρώτο όρισμα είναι το υποδέντρο που θα γίνει η αναζήτηση (σύμφωνα με τη δεντρική δομή του LDAP, το δεύτερο δηλώνει αν θα πρέπει να γίνει αναζήτηση και στα υποστοιχεία του δέντρου (το SUBTREE δηλώνει πως πρέπει να γίνει), το τρίτο είναι το στοιχείο βάσει του οποίου θα γίνει αναζήτηση (εδώ στην ουσία θα μας επιστραφούν όλοι οι εγγεγραμένοι χρήστες), και το τελευταίο είναι τα attributes τα οποία θέλουμε να μας επιστραφούν. Το αποτέλεσμα που λαμβάνουμε είναι αρκετά περίεργο: μια λίστα, κάθε στοιχείο της οποίας είναι ένα tupple. Κάθε tupple είναι κάθε στοιχείο που

ανταποκρίνεται στο φίλτρο. Το πρώτο στοιχείο του tupple είναι το DN του χρήστη (κατά κάποιο τρόπο το όνομα χρήστη του στον LDAP) και το δεύτερο είναι ένα λεξικό με τα attributes που ζητήσαμε να μας επιστραφούν. Το λεξικό όμως έχει ως κλειδιά απλά αλφαριθμητικά, αλλά η τιμή κάθε κλειδιού είναι λίστα, καθώς κάθε attribute στον LDAP έχει τη δυνατότητα να είναι multi value (πολλαπλών τιμών), και κάθε τιμή είναι ένα στοιχείο της λίστας. Εκτενέστερη αναφορά στην παρούσα συνάρτηση θα γίνει στο κεφάλαιο του LDAP όπου θα αναφερθεί και η δομή της υπηρεσίας καταλόγου.

Προσθήκη

```
>>> attrs = { 'objectClass': ['top', 'person', 'teilarStudent'], 'cn': ['firstname'], 'sn': ['lastname'], 'uid': ['username'], 'registrationNumber': ['t1387'] }

>>> ldif = modlist.addModlist(attrs)

>>> l.add_s('uid=username,ou=teilarStudents,dc=teilar,dc=gr', ldir)
```

Δημιουργούμε ένα λεξικό, του οποίου τα κλειδιά είναι πεδία του LDAP και οι τιμές των κλειδιών είναι λίστες με μία ή περισσότερες τιμές, και τις αποθηκεύουμε στο λογαριασμό του χρήστη.

Διόρθωση

```
>>> old = { 'registrationNumber': ['t1387'] }

>>> new = { 'registrationNumber': ['t1105'] }

>>> ldif = modlist.modifyModlist(old, new)

>>> l.modify_s('uid=username,ou=teilarStudents,dc=teilar,dc=gr', ldif)
```

Διαγραφή

```
>>> l.delete_s('uid=username,ou=teilarStudents,dc=teilar,dc=gr')
```

Αποσύνδεση

```
>>> l.unbind_s()
```

3.1.5 >>> import LDAP_Groups

Το LDAP_Groups δεν είναι βιβλιοθήκη της python όπως οι παραπάνω, αλλά ένα django application, το οποίο βρίσκεται σε πρώιμο στάδιο, και με τη χρήση της βιβλιοθήκης python-ldap αναλαμβάνει να κάνει τη σύνδεση των χρηστών ανάμεσα στη βάση δεδομένων του Django και τον LDAP server. Το παρών πρόγραμμα δεν υπήρχε σε πακέτο στη διανομή Gentoo, οπότε έπρεπε ή να το δημιουργήσουμε εμείς ή να το ενσωματώσουμε στον κώδικά μας. Επιλέξαμε το δεύτερο, διότι έπρεπε να αλλαχτεί πάρα πολύ ο κώδικάς του λόγω της εμβρυακής κατάστασής του. Εκτενέστερη αναφορά πάνω στη λειτουργία του θα γίνει στο κεφάλαιο 6 όπου θα περιγραφεί και η δομή του LDAP.

3.2 Δημιουργία Models

Εφόσον συλλέξαμε τα εργαλεία που μας ενδιέφεραν, το επόμενο βήμα μας ήταν να βρούμε όλες τις ιστοσελίδες του TEI που παρέχουν ανακοινώσεις, να τις συλλέξουμε και να τις ξεκαθαρίσουμε, φυσικά με τη χρήση κυρίως των εργαλείων PycURL και BeautifulSoup.

Στα προηγούμενα παραδείγματα χρησιμοποιήσαμε αυτές τις βιβλιοθήκες και καταφέραμε να αποθηκεύσουμε τα δεδομένα σε ένα δισδιάστατο Array, το οποίο όμως δεν ήταν ο επιθυμητός τρόπος αποθήκευσης των δεδομένων, αν αναλογιστούμε το πλήθος των ιστοσελίδων των ανακοινώσεων και των φοιτητών. Για παράδειγμα, ένας αλγόριθμος ταξινόμησης πίνακα σε ένα τέτοιο μέγεθος θα έπαιρνε πάρα πολλή ώρα να εκτελεστεί και θα καθυστερούσε αισθητά την ιστοσελίδα μας. Φυσικά και άλλοι παράγοντες, όπως η δυσκολία εώς και αδυναμία παροχής ανακοινώσεων ανά άτομο, μας ανάγκασαν πολύ γρήγορα να στραφούμε στην αποθήκευση των ανακοινώσεων σε βάση δεδομένων.

Το Django χρησιμοποιεί τα models για την δημιουργία βασεων δεδομένων και πινάκων. Δημιουργήσαμε λοιπόν το παρακάτω model:

```
class Id(models.Model):
```

```
        urlid = models.CharField("URL id", max_length = 30, unique = True)

        name = models.CharField("Teacher, School or Lesson name", max_length =
100)

        email = models.EmailField("Teacher's mail", null = True)

        department = models.CharField("Teacher's department", max_length = 100,
null = True)



    class Announcements(models.Model):

        title = models.CharField("Title", max_length = 500)

        date_fetched = models.DateTimeField(auto_now = True)

        urlid = models.ForeignKey(Id)

        unique = models.CharField("Unique entry", max_length = 250, unique = True
)

        url = models.URLField()

        description = models.TextField("Attachment Text", null = True)

        attachment_text = models.CharField("Attachment Text", max_length = 200,
null = True)

        attachment_url = models.CharField("Attachment URL", max_length = 200,
null = True)
```

Το model αυτό περιγράφει την δημιουργία δυο πινάκων.

3.2.1 Πίνακας Id

Ο πίνακας Id περιέχει τα εξής πεδία:

- Το πεδίο urlid είναι μια μοναδική τιμή που χαρακτηρίζει μια πηγή

ανακοινώσεων, συνήθως είναι το id τις ιστοσελίδας από όπου παίρνουμε την αντίστοιχη ανακοίνωση.

- Το πεδίο name είναι είτε το όνομα του καθηγητή του οποίου την ανακοίνωση κάνουμε parse, είτε το όνομα του μαθήματος στο e-class, είτε το όνομα τις σχολής από την οποία παίρνουμε τις ανακοινώσεις, είτε γενικά το όνομα της πηγής των ανακοινώσεων.

- Το πεδίο email κρατάει την διεύθυνση ηλεκτρονικού ταχυδρομείου του κάθε καθηγητή. Με μια πρώτη ματιά ίσως και να φαίνεται άχρηστο, αλλά απλά θεωρήσαμε ότι πρέπει να υπάρχουν τα στοιχεία αυτά συγκεντρωμένα στη βάση δεδομένων ώστε να μπορούν να εξαχθούν ανά πάσα ώρα και στιγμή.

- Το πεδίο department είναι το πεδίο στο οποίο αποθηκεύεται το όνομα του τμήματος στο οποίο ανήκει ο κάθε καθηγητή.

3.2.2 Πίνακας Announcements

Ο πίνακας Announcements περιέχει τα εξής πεδία:

- Το πεδίο title όπου κρατάμε τον τίτλο τις εκάστοτε ανακοίνωσης και μπορεί να έχει μήκος μέχρι 500 χαρακτήρες.

- Το πεδίο date_fetched κρατάει την ημερομηνία και ώρα που έγινε parse μια ανακοίνωση. Επειδή οι εκάστοτε ανακοίνωσης δεν έχουν ημερομηνία δημοσίευσης έτσι έπρεπε να κρατάμε μια ημερομηνία και ώρα για να ξέρουμε περίπου ποτε βγήκε η ανακοίνωση. Φτιάξαμε το πεδίο date_fetched το οποιο συμπληρώνεται αυτόματα με την ημερομηνία και ώρα την στιγμή ακριβώς που εισάγεται μια νέα ανακοίνωση στην βάση.

- Το πεδίο urlid το οποιο είναι ξένο κλειδί από τον πινακα Id και μας δείχνει από ποια πηγή προέρχεται η κάθε ανακοίνωση

- Το πεδίο unique είναι ένα πεδίο στο οποιο δεν επιτρέπονται οι διπλοεγγραφές. Στο πεδίο unique βάζουμε κάποιο χαρακτηριστικό γνώρισμα τις ανακοίνωσης το οποιο μπορεί να είναι είτε ο υπερσύνδεσμος τις ανακοίνωσης, είτε οι πρώτοι 250 χαρακτήρες μιας ανακοίνωσης η ακόμη και όλα μαζί προκειμενου να εξασφαλίσουμε ένα πεδίο με μοναδικό περιεχόμενο για κάθε ανακοίνωση. Με αυτόν τον τρόπο αποφεύγουμε να εισάγουμε την ίδια

ανακοίνωση δύο φορες στην βάση μας.

- Το πεδίο url είναι αυτό που κρατάει το URL τις ανακοίνωσης.

- Το πεδίο description είναι αυτό στο οποίο αποθηκεύεται το πλήρες περιεχόμενο της κάθε ανακοίνωσης. Το να πάρουμε το πλήρες περιεχόμενο τις κάθε ανακοίνωσης δεν ήταν πάντα εύκολο. Σε μερικές ιστοσελίδες του TEI υπήρχε μίνο σύνοψη της ανακοίνωσης και υπερσύνδεσμος με το πλήρες κείμενο τις ανακοίνωσης. Εμείς λοιπόν έπρεπε να τις εντοπίσουμε αυτές τις ανακοίνωσεις και να κάνουμε εκ νέου parsing με PycURL και BeautifulSoup και να πάρουμε το πλήρες κείμενο της ανακοίνωσης.

- Το πεδίο attachment_text. Πολλές ανακοίνωσεις έχουν και επισυναπτόμενα αρχεία. Σε αυτό το πεδίο κρατάμε το κείμενο ή τον τίτλο του επισυναπτόμενου αρχείου.

- Το πεδίο attachment_url στο οποίο κρατάμε τον υπερσύνδεσμο του επισυναπτόμενου αρχείου, δηλαδή από πού μπορούν να κατεβάσουν το επισυναπτόμενο.

3.3 Από Python Script σε Stand-Alone Django Application

Αφού δημιουργήσαμε τους πίνακες όπου θα αποθηκεύσουμε τα δεδομένα μας διαπιστώσαμε πως η εφαρμογή που φτιάξαμε πιο πριν για να κάνει parsing όλες τις ανακοίνωσης δεν μπορεί να εισάγει δεδομένα στην βάση δεδομένων του django. Έπρεπε δηλαδή να μετατρέψουμε την εφαρμογή python σε django εφαρμογή. Αυτό θα ήταν αρκετά απλό να το κάνουμε αν θέλαμε να βάλουμε την εφαρμογή να εκτελείται από τον Django εξυπηρετητή.

Για να έχουμε όμως την βάση δεδομένων μας με τις ανακοίνωσεις πάντοτε ενημερωμένη και να έχει όλες τις τελευταίες ανακοίνωσεις, έπρεπε αυτήν η εφαρμογή να εκτελείται ανά τακτά χρονικά διαστήματα. Δηλαδή έπρεπε να δημιουργήσουμε μια προγραμματισμένη εργασία η οποία θα εκτελείται κάθε μια ώρα για παράδειγμα. Έτσι έπρεπε να κάνουμε την python εφαρμογή μας stand-alone django εφαρμογή ώστε να μπορούμε να εισάγουμε τα δεδομένα μας στην βάση δεδομένων του django.

3.3.1 Ρυθμίσεις

Έτσι το μονο που είχαμε να κάνουμε για να το πετύχουμε αυτό ήταν να εισάγουμε τις

παρακάτω γραμμές στην αρχή της εφαρμογής python:

```
from proj_root import *

import os

import sys

sys.path.append(PROJ_ROOT)

os.environ['DJANGO_SETTINGS_MODULE'] = 'cronos.settings'
```

Και στον ίδιο φάκελο με την εφαρμογή δημιουργήσαμε ένα αρχείο με όνομα proj_root.py με περιεχόμενο:

```
# -*- coding:utf-8 -*-

PROJ_ROOT = '/home/cronos/'
```

...το οποίο προσθέσαμε και στο .svnignore

```
$ cat .svnignore

*.pyc

proj_root.py
```

Το αρχείο proj_root.py το καλούμε στην πρώτη γραμμή από την εφαρμογή, και είναι το αρχείο το οποίο φανερώνει την πλήρη διαδρομή που βρίσκεται το django project μας, στην μεταβλητή PROJ_ROOT. Με τη συνάρτηση os.environ δηλώνουμε το όνομα της Django εφαρμογής μας. Ο λόγος που το συγκεκριμένο αρχείο προστέθηκε στο .svnignore είναι διαφορετικός από το λόγο που προσθέσαμε τα δυαδικά αρχεία. Με αυτό τον τρόπο το περιεχόμενο του συγκεκριμένου αρχείου μπορεί να είναι διαφορετικό στον εξυπηρετητή από τα μηχανήματα στα οποία γίνεται η ανάπτυξη της εφαρμογής μας. Η βάση δεδομένων μας πλέον μπορεί να παραχθεί από το Django, δίνοντας απλά την εντολή

```
# python manage.py syncdb
```

Η παραπάνω εντολή είναι υπεύθυνη να δημιουργήσει ή να τροποποιήσει τους νέους πίνακες.

3.3.2 Cron Jobs

Έχουμε φτιάξει μια προγραμματισμένη εργασία η οποια κάθε μια ώρα εκτελείται και τραβάει όλες τις νέες ανακοίνωσεις από όλες τις ιστοσελίδες του TEI.

Οι πηγές από τις οποιες τραβάμε τις ανακοίνωσης μας δεν είναι όλες σταθερές. Είναι πολύ πιθανό νέοι καθηγητές να προστεθούν και θα πρέπει να πάρουμε και τις δικές τους ανακοίνωσεις. Για να το πετύχουμε δημιουργήσαμε μια ακόμη stand-alone django εφαρμογή η οποια εκτελείται μια φορά την εβδομάδα και η οποια ελέγχει αν έχουν δηλωθεί νέοι καθηγητές στην κεντρική σελίδα του TEI και αν υπάρχουν καινουργή τους πρόσθετη στον πινακα Id.

Έτσι την επομενη φορα που θα εκτελεστεί η προγραμματισμένη εργασία για να τραβήξει όλες τις ανακοίνωσης θα συμπεριλάβει και τις νέες πηγές που προστέθηκαν στον πίνακα Id.

Στο Gentoo Linux οι προγραμματισμένες διεργασίες δηλώνονται στον κατάλληλο κατάλογο με όνομα /etc/cron.{hourly, daily, weekly, monthly}. Τοποθετούμε δύο απλά bash scripts, ένα στον κατάλογο cron.hourly και ένα στον κατάλογο cron.monthly με το εξής περιεχόμενο:

```
# cat /etc/cron.hourly/cronos.sh

#!/bin/bash

python /home/code/cronos/cron/announcements.py > /dev/null 2>&1

# cat /etc/cron.weekly/cronos.sh

#!/bin/bash

python /home/code/cronos/cron/id.py > /dev/null 2>&1
```

ΚΕΦΑΛΑΙΟ 4ο: ΤΟ DJANGO WEB FRAMEWORK

4.1 Ιστορικό

Το Django framework ξεκίνησε σαν εσωτερικό εφαρμογή από την εφημερίδα Lawrence Journal-World το 2003. Η ομάδα ανάπτυξης διαδικτυακών εφαρμογών ήταν αναγκασμένη πολύ συχνά να δημιουργήσει νέες δυνατότητες ή ακόμη και ολόκληρες εφαρμογές μέσα σε μερικές ώρες. Έτσι δημιουργήθηκε το Django για να ανταποκρίνεται στα στενά χρονικά περιθώρια που έχει συνήθως μια ειδησιογραφική ιστοσελίδα. Ταυτόχρονα όμως καθιστούσε την διαδικασία ανάπτυξης ξεκάθαρη και την συντήρηση της εφαρμογής εύκολη. Μέχρι το καλοκαίρι του 2005 το Django είχε ωριμάσει αρκετά ώστε να χρησιμοποιείται σε μερικές ιστοσελίδες με πολύ υψηλά επίπεδα επισκεψιμότητας. Τότε οι δημιουργοί του προγράμματος αποφάσισαν να το δημοσιοποιήσουν σαν εφαρμογή Ανοιχτού Κώδικα. Η εφαρμογή πήρε το όνομα της από τον διάσημο κιθαρίστα της Jazz, Django Reinhardt.

4.2 Η αρχή DRY

Το Django είναι ένα υψηλού επιπέδου framework βασισμένο στην γλώσσα προγραμματισμού Python το οποιο ενθαρρύνει την ταχύτατη ανάπτυξη διαδικτυακών εφαρμογών με πολύ καθαρό κώδικα. Επίσης με τα Templates του Django μπορούμε να δημιουργήσουμε καλοσχεδιασμένες και όμορφες ιστοσελίδες σε πολύ λίγο χρόνο. Το Django επικεντρώνεται στο να αυτοματοποιεί όσο το δυνατόν περισσότερες λειτουργίες και παραμένει πάντοτε προσκολλημένο στην αρχή DRY (Don't Repeat Yourself - Μην επαναλαμβάνεις τον εαυτό σου).

Η λογική αυτής της αρχής, είναι ότι όταν κάποτε χρειαστεί να αλλάξουμε την συμπεριφορά σε κάποιο σημείο της εφαρμογής, δεν χρειάζεται να αλλάξουμε τον κώδικα σε περισσότερα από ένα σημεία.

Αυτό γίνεται εφικτό με τον εξής τρόπο: αντί να γράφουμε συνεχώς τον ίδιο κώδικα σε διάφορα σημεία της εφαρμογής, γράφουμε μια φορά τον κώδικα σε κάποιο “κεντρικό” σημείο και όταν θέλουμε να χρησιμοποιήσουμε αυτό το τμήμα του κώδικα κάπου στην εφαρμογή, απλά κάνουμε μια αναφορά στο μέρος όπου βρίσκεται γραμμένος ο κώδικας. Έτσι αν θέλουμε να αλλάξουμε την συμπεριφορά της εφαρμογής δεν χρειάζεται να αλλάξουμε τα

σημεία όπου καλείται ο κώδικας, αλλά μόνο το σημείο που είναι γραμμένος ο κώδικας.

Ένα παράδειγμα για το πως το Django υποστηρίζει αυτή την αρχή είναι το εξής. Σε αντίθεση με την JAVA δεν χρειάζεται να ορίζουμε συνεχώς την βάση δεδομένων και να κάνουμε συνεχώς σύνδεση με αυτή μέσα στην ίδια εφαρμογή. Πολύ απλά ορίζουμε μια φορά την βάση δεδομένων και όταν χρειάζεται να αντληθούν πληροφορίες από την βάση το Django, χρησιμοποιεί τον ήδη υπάρχοντα ορισμό. Ένα ακόμη παράδειγμα αυτής της αρχής είναι η ανάπτυξη τεχνικών όπως το AJAX (asynchronous javascript and XML), όπου όταν κάποιος browser δεν υποστηρίζει αυτή την τεχνική, ο προγραμματιστής πρέπει να εμφανίσει τα αποτελέσματα χωρίς να χρησιμοποιήσει την τεχνική AJAX. Σε τέτοιες περιπτώσεις πολλοί προγραμματιστές πιάνουν τον εαυτό τους να γράφει τον ίδιο κώδικα πολλές φορές. Αυτό μπορεί πολύ εύκολα να το παρακάμψει το Django Framewrork χρησιμοποιώντας την αρχή DRY.

4.3 Κύρια πλεονεκτήματα

4.3.1 Object-relational mapper

Το Django διαχειρίζεται την βάση με το Object Relational Mapping. Σύμφωνα με αυτή την λογική, ένας πίνακας σε μια βάση δεδομένων διαχειρίζεται ως μία κλάση, τα πεδία του πίνακα ως τα δεδομένα-μεταβλητές της κλάσης και οι εγγραφές του πίνακα ως αντικείμενα της κλάσης. Οι κλάσεις αυτές έχουν αρκετές συναρτήσεις για την διαχείριση της βάσης. Έτσι για παράδειγμα εάν υπάρχει ένας πίνακας με το όνομα Announcements, θα υπάρχει και μια κλάση στην εφαρμογή με το όνομα Announcements, η οποία θα αντιστοιχίζεται με τον πίνακα στην βάση δεδομένων.

Με τον Object-relational mapper έχουμε τεράστια ευελιξία καθώς η ιστοσελίδα μας είναι εντελώς ανεξάρτητη από το σύστημα διαχείρισης βάσεων δεδομένων. Δηλαδή μπορούμε να φτιάξουμε την ιστοσελίδα μας ή την εφαρμογή μας σε MySQL και μετά με την αλλαγή μόνο μερικών ρυθμίσεων να χρησιμοποιήσουμε PostgresSQL χωρίς να χρειαστεί να αλλάξουμε ούτε μια γραμμή από τον κώδικα τις εφαρμογής μας.

4.3.2 Αυτοματοποιημένη διεπαφή διαχείρισης

Ένα από τα ισχυρότερα τμήματα και εργαλεία του Django framework είναι το

αυτοματοποιημένο σύστημα διαχείρισης. Το σύστημα παίρνει τα δεδομένα από το μοντέλα με σκοπό να παράγει μια πανίσχυρη και έτοιμη για άμεση χρήση διαδικτυακή επαφή διαχείρισης.

Για μια συγκεκριμένη κλάση τις διαδικτυακή σελίδας, η διεπαφή διαχείρισης αποτελεί ουσιώδες μέρος τις υποδομής. Πρόκειται για μια διεπαφή βασισμένη στο διαδίκτυο στην οποία έχουν περιορισμένη πρόσβαση μόνο όσοι είναι πιστοποιημένοι και αξιόπιστοι διαχειριστές τις ιστοσελίδας. Έχοντας εξουσιοδοτημένοι πρόσβαση στην ιστοσελίδα μπορούν να προσθέσουν να αλλάξουν ή ακόμη και να διαγράψουν περιεχόμενο τις ιστοσελίδας. Η διεπαφή που χρησιμοποιείται για να κάνετε μια νέα δημοσίευση στο blog σας, το backend που χρησιμοποιούν οι διαχειριστές τις ιστοσελίδας για να διαχειριστούν τα σχόλια τον αναγνωστών, το εργαλείο που χρησιμοποιούν οι πελάτες σας για να διαχειριστούν την ιστοσελίδα που κατασκευάσατε γι'αυτούς – όλα αυτά είναι παραδείγματα διεπαφών διαχείρισης.

Ένα σημαντικό πρόβλημα των διεπαφών διαχείρισης είναι ότι η διαδικασία δημιουργία τους δεν είναι καθόλου διασκεδαστική. Συνήθως, η διαδικασία δημιουργίας μιας ιστοσελίδας είναι συναρπαστική, διασκεδαστική και ενδιαφέρων, αλλά η δημιουργία της διεπαφής διαχείρισης είναι σχεδόν πάντα η ίδια. Ο χρήστης πρέπει να πιστοποιηθεί ώστε να έχει πρόσβαση στην διεπαφή διαχείρισης, να του επιτραπεί να διαχειρίζεται φόρμες καθώς και τις εισόδους των χρηστών. Πραγματικά, μια διαδικασία χωρίς ιδιαίτερο δημιουργικό ενδιαφέρων.

Εκεί ακριβώς είναι και το δυνατό σημείο του Django, το οποίο μας απαλλάσσει από όλη αυτήν την βαρετή διαδικασία της δημιουργίας διεπαφής διαχείρισης και τα αναλαμβάνει όλα να γίνουν αυτοματοποιημένα απλά και μόνο τροποποιώντας ελάχιστα τον κώδικα. Με το Django η δημιουργία διεπαφής διαχείρισης δεν είναι πλέον πρόβλημα για τον προγραμματιστή.

4.3.3 Όμορφος σχεδιασμός των URL

Ένα καθαρό και εκλεπτυσμένο URL σχήμα είναι μια αρκετά σημαντική λεπτομέρεια σε μια υψηλής ποιότητας διαδικτυακή εφαρμογή. Το Django ενθαρρύνει τον όμορφο και ευδιάκριτο σχεδιασμό τον URL, αποφεύγοντας τις ανούσιες καταλήξεις όπως συμβαίνει μ την .php ή την .asp.

Το μόνο που χρειάζεται για να επιτευχθεί ένα σύνολο καθαρών URL για μια Django εφαρμογή είναι η δημιουργία και κατάλληλη παραμετροποίηση του Django module εν ονόματι URLconf..

4.3.4 Template system

Στο Django το template είναι ένα αλφαριθμητικό το προορίζεται να διαχωρίσει την παρουσίαση του εγγράφου από τα δεδομένα. Ένα template αποτελείται από placeholders και από αρκετές ετικέτες οι οποίες είναι υπεύθυνες για την εμφάνιση τις ιστοσελίδας. Συνήθως τα templates χρησιμοποιούνται για να παράγουν σαν έξοδο HTML κώδικα, αν και στο Django τα templates μπορούν κάλλιστα να παράγουν κείμενο με οποιαδήποτε μορφή.

Ο κώδικας που ακολουθεί είναι ένα δείγμα template που θα παράγει μια ιστοσελίδα HTML η οποία καλωσορίζει τον χρήστη που μόλις εγγράφτηκε στο Cronos:

```
{% extends "intro.html" %}



{% block title %}

    Καλώς Ήρθατε |

{% endblock %}



{% block content %}

    <div id="login">

        <div id="info">

            <h3>Καλώς Ήρθατε</h3>

            <p>Καλώς ήρθατε στην υπηρεσία cronos του ΤΕΙ Λάρισας.</p>

            <h3>Ρυθμίσεις</h3>
```

```
                    <p>Είστε πλέον έτοιμοι να κάνετε εισαγωγή με το νέο σας
λογαριασμό. Στο προφίλ σας θα μπορέσετε να επιλέξετε ποιες ανακοινώσεις θέλετε να
λαμβάνετε, είτε από υπηρεσίες του ΤΕΙ Λάρισας (Κόμβος, Γραμματεία, Βιβλιοθήκη) είτε
από καθηγητές, είτε από μαθήματα του e-class. Για να μεταβείτε στην αρχική σελίδα και να
κάνετε εισαγωγή στην υπηρεσία πατήστε <a href="/">εδώ</a></p>

                </div>

                <div id="welcome">

                    <h3>Ονόματα Χρήστη</h3><br />

                    <ul>

                        <li><b>Cronos:</b> {{ username }}</li>

                        <li><b>Dionysos:</b> {{ dionysos_username }}</li>

                        {% if eclass_username %}

                        <li><b>E-class:</b> {{ eclass_username }}</li>

                        {% endif %}

                        {% if webmail_username %}

                        <li><b>Webmail:</b> {{ webmail_username }}</li>

                        {% endif %}

                    </ul>

                    <h3>Πλήρη Στοιχεία</h3><br />

                    <ul>

                        <li><b>Ονοματεπώνυμο:</b> {{ first_name }}
{{ last_name }}</li>
```

```
                    <li><b>Εξάμηνο:</b> {{ semester }}</li>
                    <li><b>Έτος Εισαγωγής:</b> {{ introduction_year }}</li>
                    <li><b>Σχολή:</b> {{ school }}</li>
                </ul>
            </div>
        </div>
    {% endblock %}
```

Αυτό το template είναι κατά κύριο λόγο HTML με μερικές μεταβλητές και template ετικέτες. Ακολουθεί μια μικρή ανάλυση:

Οποιοδήποτε κείμενο που περιβάλλεται από διπλες αγκύλες – για παράδειγμα {{first_name}} – είναι μεταβλητή. Αυτό σημαίνει πως πρέπει σε εκείνο το σημείο να μπε η μεταβλητή με το όνομα first_name..

Οποιοδήποτε κείμενο που περιβάλλεται από μονή αγκύλη και επί τις εκατό – για παράδειγμα {% if webmail_username %} - είναι σύνολο από ετικέτες. Το σύνολο τον ετικετών απλά λέει στο template να εκτελέσει μια συγκεκριμένη εντολή.

4.3.5 Λανθάνουσα μνήμη

Το Django έχει ένα αρκετά εξελιγμένο σύστημα λανθάνουσας μνήμης. Το σύστημα λανθάνουσας μνήμης κάθε φορά που ένας χρήστης ζήτησει μια σελίδα κάνει τους απαραίτητους υπολογισμούς και αποθηκεύει τα δεδομένα αυτά σε κάποιο σημείο του δίσκου προσωρινά. Έτσι κάθε φορά που ένας άλλος χρήστης ζητήσει την ίδια σελίδα δεν θα χρειαστεί να εκτελέσει όλους τους υπολογισμούς και τα SQL ερωτήματα.

4.3.6 Διεθνοποίηση

Το Django έχει πλήρη υποστήριξη για πολυγλωσσικό σύστημα. Το πολυγλωσσικό του

σύστημα υποστηρίζει μετατροπή κειμένου, αριθμών, ημερομηνιών κ.α.

4.4 MVC (models – views – controllers)

4.4.1 Μοντέλα (Models)

Το 1979 Trygve Reenskaug εμπνεύστηκε μια νέα αρχιτεκτονική για την ανάπτυξη διαδραστικών εφαρμογών. Σε αυτή την αρχιτεκτονική η εφαρμογή "σπάει" σε τρία μέρη: στα models, στα views και στους controllers. Το Django framework χρησιμοποιεί αυτή την αρχιτεκτονική.

Το μοντέλο (model) αντιστοιχίζεται στα δεδομένα της εφαρμογής. Τα δεδομένα αυτά μπορεί να είναι παροδικά, όταν υπάρχει μεγάλη αλληλεπίδραση με τον χρήστη, ή μπορεί να είναι σταθερά, και τότε συνήθως πρέπει να αποθηκευτούν κάπου εκτός της εφαρμογής, για παράδειγμα σε μια βάση δεδομένων.

Το μοντέλο δεν είναι μόνο τα δεδομένα της εφαρμογής. Είναι αυτό που επιβάλλει όλες τις ενέργειες και τους περιορισμούς που αφορούν τα δεδομένα της εφαρμογής. Αναπτύσσοντας λοιπόν αυτές τις ενέργειες και τους περιορισμούς μέσα στο ίδιο το μοντέλο, είμαστε σίγουροι πως στην εφαρμογή μας τίποτα, εκτός από το μοντέλο, δεν μπορεί να "πειράξει" τα δεδομένα μας. Έτσι λοιπόν ένα μοντέλο είναι ταυτόχρονα διαχειριστής και φύλακας των δεδομένων της εφαρμογής μας.

4.4.2 Όψεις (Views)

Η όψη (view) είναι υπεύθυνη για την παραγωγή της διασύνδεσης χρήστη (user interface), που συνήθως βασίζεται στα δεδομένα του μοντέλου. Για παράδειγμα, στο Cronos η λίστα των καθηγητών είναι προσπελάσιμη από το μοντέλο, αλλά η όψη είναι αυτή που μέσω του μοντέλου τους παρουσιάζει στον χρήστη ως λίστα. Αν και η όψη μπορεί να παρουσιάσει στον χρήστη αποτελέσματα με διάφορους τρόπους, δεν μπορεί ποτέ να χειριστεί δεδομένα που έρχονται από την χρήστη στην εφαρμογή. Η δουλειά του view τελειώνει όταν τελικά παρουσιαστούν τα δεδομένα στον χρήστη. Στο ίδιο μοντέλο μπορούν να υπάρχουν πολλές όψεις, κάθε μια για διαφορετικό σκοπό.

4.4.3 Ελεγκτές (Controllers)

Οι ελεγκτές (controllers) είναι αυτοί που οργανώνουν την εφαρμογή. Οι ελεγκτές δέχονται δεδομένα από τον έξω κόσμο (χρήστη) αλληλεπιδρούν με το μοντέλο και παρουσιάζουν τα κατάλληλα αποτελέσματα μέσω της όψης στον χρήστη.

Το Django είναι ένα framework που χρησιμοποιεί την τεχνολογία MVC. Δημιουργεί στον προγραμματιστή μια τέτοια δομή, και όταν ο προγραμματιστής αναπτύξει τα μοντέλα, τις όψεις και τους ελεγκτές, το Django τα συνδυάζει και παράγεται η τελική εφαρμογή. Ένα από τα πλεονεκτήματα του Django είναι ότι έχει την "ευφυΐα" να τα συνδυάσει μεταξύ τους και έτσι ο προγραμματιστής δεν χρειάζεται να γράψει επιπλέον κώδικα ώστε να τα κάνει να δουλέψουν όλα μαζί. Και αυτό είναι ένα παράδειγμα της ευκολίας που προσφέρει το Django Framework.

Σε μια Django εφαρμογή, ένα εισερχόμενο αίτημα στέλνεται αρχικά σε κάποιον δρομολογητή, ο οποίος αποφασίζει που πρέπει να σταλεί και πως πρέπει να επεξεργαστεί το αίτημα. Σε αυτό το σημείο καλείται η κατάλληλη μέθοδος στον ελεγκτή, ο ελεγκτής θα επεξεργαστεί το αίτημα, αν χρειαστεί θα αλληλεπιδράσει με το μοντέλο για να εξάγει κάποια δεδομένα από την βάση δεδομένων και τελικά θα εμφανίσει τα αποτελέσματα στον χρήστη μέσω της όψης.

4.5 Δημιουργία του νέου μας Project

4.5.1 Αρχική δομή

Έχοντας εφοδιαστεί με τις παραπάνω γνώσεις, ήρθε η ώρα να προχωρήσουμε στη δημιουργία ενός django project, το οποίο θα στεγάσει τη νέα μας ιστοσελίδα. Η εντολή για να πραγματοποιηθεί η αρχική δομή είναι η εξής:

```
$ django-admin.py startproject cronos

$ ls cronos

__init__.py  manage.py  settings.py  urls.py
```

Δημιουργήθηκαν 4 αρχεία:

- *__init__.py* Ένα άδειο αρχείο, το οποίο πληροφορεί στο Django ότι ο συγκεκριμένος κατάλογος πρέπει να αντιμετωπιστεί ως πακέτο της Python

- *manage.py* Ένα εργαλείο κονσόλας το οποίο μας επιτρέπει να αλληλοεπιδρούμε με το Django Project με διάφορους τρόπους

- settings.py Αρχείο ρυθμίσεων για το συγκεκριμένο Django project

- urls.py Οι δηλώσεις των URL για το συγκεκριμένο Django Project, το οποίο στην ουσία είναι ο πίνακας των περιεχομένων της ιστοσελίδας.

4.5.2 Ο ενσωματωμένος διακομιστής

Ο ενσωματωμένος διακομιστής ανάπτυξης του Django είναι ένας ελαφρύς διακομιστής γραμμένος καθαρά σε python. Η συγκεκριμένη λειτουργία είναι απίστευτα χρήσιμη, καθώς μπορούμε εύκολα και γρήγορα να έχουμε εν λειτουργία μια ιστοσελίδα με όλες τις απαραίτητες λειτουργίες που θα περιμέναμε από ένα διακομιστή, χωρίς να χρειάστηκε να μπούμε στον κόπο να στήσουμε Apache Web Server, παρά μόνον όταν η σελίδα έπρεπε να βγει στην παραγωγή. Ο συγκεκριμένος διακομιστής χρησιμοποιείται μόνο για την ανάπτυξη καθώς δεν μπορεί να καλύψει τις απαιτήσεις μιας πλήρους ιστοσελίδας. Ο συγκεκριμένος server καλείται με την παρακάτω εντολή, όπου η IP και η port μπορούν εύκολα να μεταβληθούν:

```
python manage.py runserver 127.0.0.1:8000
```

4.5.3 Ρύθμιση της βάσης δεδομένων

Ήρθε η ώρα να εξερευνήσουμε το αρχείο ρυθμίσεων. Είναι απλά ένα αρχείο με μεταβλητές, οι οποίες μπορούν να καλεστούν καθ'όλη τη διάρκεια του project. Επίσης δηλώνονται μεταβλητές που αφορούν το project αυτό καθ' αυτό, όπως τα ονόματα χρήστη και email των διαχειριστών, στοιχεία που αφορούν τη βάση δεδομένων κτλ.

Αφού δημιουργήσουμε μια βάση δεδομένων από την MySQL, το μόνο που έχουμε να κάνουμε είναι να το δηλώσουμε στις κατάλληλες μεταβλητές:

```
DATABASE_ENGINE = 'mysql'
```

```
DATABASE_NAME = 'cronos'

DATABASE_PASSWORD = 'secret'

DATABASE_HOST = ''

DATABASE_PORT = ''
```

4.5.4 Δημιουργία του πρώτου application

Πριν δημιουργήσαμε ένα project που θα στεγάσει τη μελλοντική ιστοσελίδα μας. Το project είναι στην ουσία ένα σύνολο από applications και αρχείων ρυθμίσεων που συνδέονται άμεσα μεταξύ τους και παράγουν το τελικό αποτέλεσμα. Η δημιουργία ενός app γίνεται με την παρακάτω εντολή:

```
$ python manage.py startapp announcements

$ ls announcements/

__init__.py models.py tests.py views.py
```

Αυτό που θα μας απασχολήσει περισσότερο είναι το models.py και το views.py, όπου στο πρώτο θα γραφτεί το μοντέλο για τη βάση δεδομένων και στο δεύτερο η όψη της ιστοσελίδας. Το μοντέλο το έχουμε ήδη γράψει κατά τη δημιουργία των cron jobs, οπότε το εισάγουμε στο κατάλληλο αρχείο. Η ύπαρξή του δηλώνεται στο settings.py:

```
INSTALLED_APPS = ( ....   'cronos.announcements' ... )
```

...και η ανανέωση της βάσης δεδομένων με την εξής εντολή

```
python manage.py syncdb
```

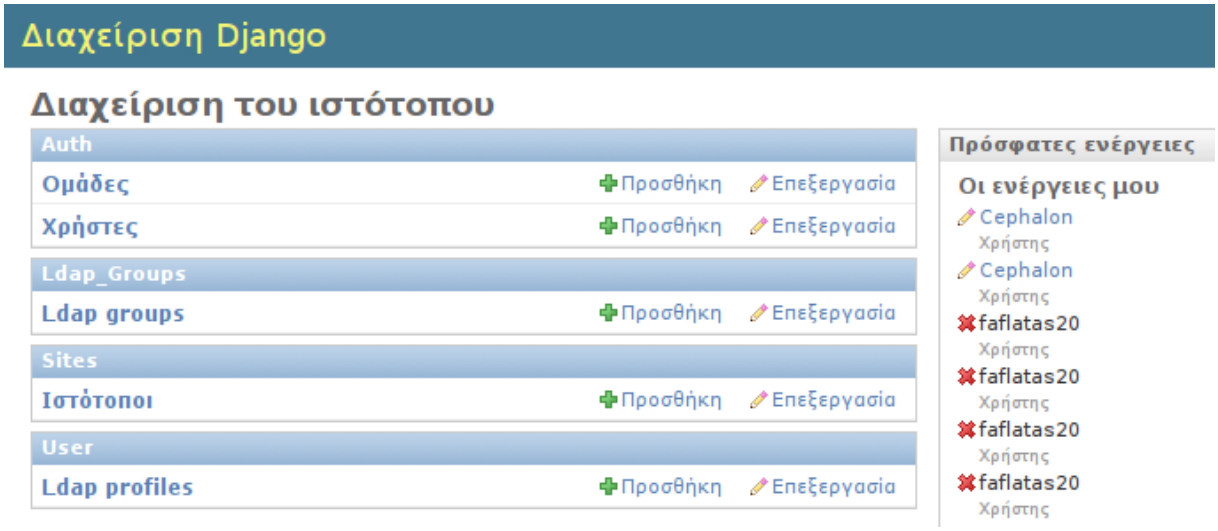
4.5.5 Ρύθμιση του admin panel

Είδαμε πιο πριν ότι ένα από τα μεγαλύτερα θετικά σημεία του Django είναι η ύπαρξη της διεπαφής διαχείρισης (admin panel) και πώς αυτό μπορεί να στηθεί σε λίγα μόνο λεπτά, προσφέροντας απεριόριστες δυνατότητες. Το μόνο που χρειάστηκε ήταν η ρύθμιση του URL του admin panel στο urls.py:

```
(r'^admin/', include(admin.site.urls)),
```

...και η αλλαγή ορισμένων μεταβλητών στο settings.py, ώστε να το έχουμε διαθέσιμο στα ελληνικά:

```
USE_I18N = True

USE_L10N = True

LANGUAGE_CODE = 'el-gr'
```



ΚΕΦΑΛΑΙΟ 5ο: Η ΙΣΤΟΣΕΛΙΔΑ CRONOS

5.1 Ανακοινώσεις

Σε αυτό το σημείο, έχουμε στην κατοχή μας δύο προγραμματισμένες εργασίες, οι οποίες τραβούν ανακοινώσεις από διάφορες υπηρεσίες του ΤΕΙ Λάρισας, καθώς και ένα υποτυπώδες Django project, το οποίο είναι υπεύθυνο ως τώρα για την σύνδεση των εργασιών με τη βάση δεδομένων . Η επιλογή των υπηρεσιών που θα τραβάμε τις ανακοινώσεις τους στην αρχή αφορούσε μόνο τις σχολές, τους καθηγητές και τα μαθήματα (e-class) των φοιτητών της τμήματος Τεχνολογίας Πληροφορικής και Τηλεπικοινωνιών, αλλά είδαμε πολύ σύντομα ότι μπορούσαν εύκολα να επεκταθούν και να παρθούν ανακοινώσεις όλων των σχολών, όλων των καθηγητών και όλων των μαθημάτων eclass. Στην πορεία προστέθηκαν και επιπλέον σελίδες, οπότε η βάση μας μεγάλωσε και τα cron jobs γίνονταν όλο και πιο αργά.

5.1.1 Κεντρική σελίδα ΤΕΙ Λάρισας

Η κεντρική σελίδα του ΤΕΙ Λάρισας (http://www.teilar.gr) παρέχει πολλών ειδών ανακοινώσεις χωρισμένες σε κατηγορίες. Εμείς τις χωρίσαμε σε τρεις μεγάλες κατηγορίες, και κάθε μία την μεταχειριστήκαμε διαφορετικά: Η λογική της συλλογής των ανακοινώσεων ήταν η ίδια για όλες τις κατηγορίες της σελίδας: πρώτα ελέγχουμε ποια σχολή/καθηγητής (Id δηλαδή) έχει αναρτήσει ανακοίνωση, και στην πορεία μπαίνουμε μέσα στο προφίλ του και συλλέγουμε το περιεχόμενο της ανακοίνωσης. Πιο αναλυτικά:

Γενικές Ανακοινώσεις

Για την ακρίβεια, είναι μια υπερκατηγορία κάποιων μικρότερων, που αφορούν ανακοινώσεις όλου του ΤΕΙ, αποτελέσματα συνελεύσεων κτλ. Για καλή μας τύχη οι δέκα τελευταίες εμφανίζονται στην κεντρική σελίδα και μας αρκούσαν, οπότε προστέθηκαν με τον τίτλο συγγραφέα “Κεντρική Σελίδα ΤΕΙ Λάρισας”

Ανακοινώσεις Τμημάτων

Παρέχονται επίσης ανακοινώσεις από όλα τα τμήματα του ΤΕΙ, τα οποία αντιπροσωπεύονται από ένα cid στο URL του καθενός, το οποίο αποτελεί και το ξένο κλειδί

που συνδέει τον πίνακα Id με τον πίνακα Announcements. Επίσης, κάποιες ανακοινώσεις μπορεί να ήταν άδειες και να αποτελούνταν μόνο από ένα επισυναπτόμενο αρχείο, το οποίο αποφασίσαμε να κρατάμε σε ξεχωριστή εγγραφή στον πίνακα (και μάλιστα ξεχωριστά τον υπερσύνδεσμο από το κείμενο που το συντρόφευε), ώστε στο μέλλον να μας διευκολύνει στην εκτύπωση της ανακοίνωσης στην ιστοσελίδα και το RSS Feed. Το ΤΕΙ αποτελείται από 25 περίπου σχολές, και πολύ εύκολα με ένα βρόγχο for καταφέραμε να συλλέξουμε όλες τις ανακοινώσεις από όλες τις σχολές, και παράλληλα να επεκτείνουμε εύκολα την εφαρμογή μας ώστε να απευθύνεται σε όλους τους φοιτητές.

Ανακοινώσεις καθηγητών

Η ίδια λογική με την παραπάνω εφαρμόστηκε και εδώ. Ο κάθε καθηγητής αντιπροσωπεύονταν από ένα pid, και απλά εισερχόμαστε στο προφιλ του καθηγητή που έχει αναρτήσει την ανακοίνωση και αποθηκεύουμε το περιεχόμενό της. Το πρόβλημα με τα επισυναπτόμενα αρχεία εμφανίστηκε και εδώ και αντιμετωπίστηκε με τον ίδιο τρόπο. Λόγω του πλήθους των καθηγητών (περίπου 330), εδώ αντιμετωπίσαμε και τα πρώτα προβλήματα ταχύτητας στην προγραμματισμένη εργασία και αναγκαστήκαμε να βελτιώσουμε τον αλγόριθμό του, αλλά χωρίς πολύ αισθητό αποτέλεσμα, μιας και το μηχάνημα στο οποίο στεγάζεται προς το παρών ο cronos είναι αρκετά αδύναμο.

5.1.2 E-Class

Η Πλατφόρμα Ασύγχρονης Τηλεκπαίδευσης (Openclass / E-Class) χρησιμοποιείται ευρέως από πολλές σχολές του ΤΕΙ Λάρισας, μιας και προσφέρει ανακοινώσεις, έγγραφα και αποστολή εργασιών. Δυστυχώς όμως ήταν και η υπηρεσία που μας δυσκόλεψε περισσότερο από όλες ώστε να έχουμε ένα υποτυπώδες αποτέλεσμα. Η συγκεκριμένη υπηρεσία απαιτεί όνομα χρήστη και κωδικό πρόσβασης για την εισαγωγή, και επίσης έπρεπε να έχουμε ένα λογαριασμό ο οποίος να είναι εγγεγραμμένος σε όλα τα μαθήματα, οπότε αναγκαστικά θυσιάσαμε το έναν από τους δύο λογαριασμούς που είχαμε και εγγραφτήκαμε σε όλα τα μαθήματα που υπάρχουν στη λίστα.

Το δεύτερο πρόβλημα που αντιμετωπίσαμε ήταν η συλλογή των ID και τίτλων των μαθημάτων. Η λίστα των μαθημάτων δεν ήταν προκαθορισμένη, αλλά ο οποιοσδήποτε με τα κατάλληλα δικαιώματα του γκρουπ καθηγητή μπορούσε να δημιουργήσει ένα μάθημα και να το ονομάσει όπως θέλει. Αν και αυτή η αναρχία στην ονοματοποίηση χαλούσε την αισθητική,

παρ'όλα αυτά αναγκαστήκαμε να συμβιβαστούμε, μιας και η ιδανική αντιστοίχηση καθηγητών και μαθημάτων δεν υφίσταται.

Έχοντας στην κατοχή μας λίστα με τα Ids που θα χρησιμοποιούσαμε ως εμφανιζόμενο όνομα του συντάκτη της ανακοίνωσης, επόμενο βήμα ήταν να αρχίσουμε να συλλέγουμε τις ανακοινώσεις, οι οποίες όμως προς μεγάλη μας έκπληξη ανακαλύψαμε ότι όλες οι ανακοινώσεις ήταν στην ίδια σελίδα, και όχι σε ξεχωριστές η κάθε ανακοίνωση, όπως συνέβαινε με την ιστοσελίδα του ΤΕΙ Λάρισας. Αυτό μας δημιούργησε το πρόβλημα ότι το πεδίο unique, το οποίο προσδιόριζε ως τώρα το URL της ανακοίνωσης, το οποίο θεωρούσαμε ότι ήταν μοναδικό, ώστε να αποφύγουμε τις διπλοεγγραφές, έπρεπε να αλλάξει το στυλ του και να βρούμε εναλλακτική. Καταλήξαμε τελικά να παίρνουμε τον τίτλο και ένα μέρος από το περιεχόμενο της κάθε ανακοίνωσης, τα οποία θεωρούσαμε ότι ο συνδιασμός τους θα ήταν πάντα μοναδικός. Φυσικά η λύση μας πάσχει, αλλά θεωρούμε ότι είναι καθαρή αδυναμία του συστήματος του e-class η οποία δεν μπορούσε να επιλυθεί καλύτερα.

Τελευταίο βήμα ήταν να αρχίσουμε να συλλέγουμε τις ανακοινώσεις και να τις αποθηκεύουμε στη βάση δεδομένων. Όμως παρουσιάστηκαν διάφορα προβλήματα, όπως ότι υπήρχαν ανακοινώσεις χωρίς τίτλο, ανακοινώσεις χωρίς κείμενο, και ανακοινώσεις που είχαν HTML κείμενο και φυσικά είτε επέστρεφαν exception είτε έκαναν λάθος parsing, μιας και μπέρδευαν τη διαδικασία του αλγορίθμου που έκανε parsing δεδομένα με τη σειρά. Το γεγονός αυτό μπορεί να γίνει εμφανές κοιτώντας τον κώδικα των προγραμματισμένων εργασιών, όπου το τμήμα που χειρίζεται το e-class έχει τους περισσότερους try -except βρόγχους από όλα τα υπόλοιπα.

5.1.3 Άλλες Ανακοινώσεις

Οι υπόλοιπες σελίδες που επιλέχτηκαν ήταν οι εξής:

- Βιβλιοθήκη
- Κέντρο Διαχείρισης Δικτύου
- Linux Team
- Πρόγραμμα Γραμματείας

- Γραφείο Διασύνδεσης
- Γραφείο Δημοσίων και Διεθνών Σχέσεων

Όλες οι παραπάνω δεν μας απασχόλησαν ιδιαίτερα, μιας και είχαν σχετικά καθαρόHTML κώδικα και δεν είχαν πλήθος περιεχομένου για να τραβήξουμε, καθώς είναι σελίδες με μικρή επισκεψιμότητα και οι περισσότερες χρησιμοποιούσαν κάποιο δημοφιλές CMS (Joomla, Drupal).

5.2 Προβολή των ανακοινώσεων

Έχοντας ένα πλήθος ανακοινώσεων στη βάση δεδομένων μας, υπολείπονταν να τις δούμε εκτυπωμένες σε ιστοσελίδα.

5.2.1 Ανάκτηση από τη βάση δεδομένων

Ως πρώτο βήμα, τα αποτελέσματα έπρεπε να ανακτηθούν από τη βάση δεδομένων, και να αποσταλούν με κάποιο τρόπο από το view στο template. Ο κώδικας ο οποίος κάνει ανάκτηση από τη βάση είναι ο παρακάτω:

```
for item in Announcements.objects.all()[:30]:
    announcements.append([item.author(), length, img, str(item.id), item__unicode__(), date])
 return render_to_response('announcements.html', {
'    announcements': announcements,
}, context_instance = RequestContext(request))
```

Στο παραπάνω απόσπασμα δημιουργήσαμε ένα δισδιάστατο πίνακα, ο οποίος αποτελείται από την ανακοίνωση με τα στοιχεία κατά σειρά: συγγραφέας, id, τίτλος, ημερομηνία. Το Announcements.objects.all()[:30] στην ουσία είναι και η αναζήτηση στη βάση δεδομένων όλων των αποτελεσμάτων αλλά επιστρέφουμε μόνο τις τελευταίες 30. Στη συνέχεια ο πίνακας αυτός στέλνεται στο template announcements.html. Ο κώδικας του announcements.html:

```
{% for i in announcements %}

{% for j in i %}

{{j|safe}}

 {% endfor %}

{% endfor %}
```

Στο παραπάνω απόσπασμα δημιουργήσαμε ένα διπλό βρόγχο for για να παίρνουμε το κάθε στοιχείο του δισδιάσταατου πίνακα announcements. Το αποτέλεσμα θα είναι το ένα στοιχείο κάτω από το άλλο. Για να μπορέσουμε να εκμεταλλευτούμε την έξοδο, χρησιμοποιούμε τη μεταβλητή forloop.counter0 η οποία δείχνει σε ποιο σημείο κάθε φορά είναι ο μετρητής της for, οπότε μπορούμε να εφαρμόζουμε διαφορετικές ρυθμίσεις στο στυλ κάθε αρχείου. Στην περίπτωσή μας έχουμε γράψει ένα μικρό CSS και έχουμε εμφανίσει τα αποτελέσματα σε ένα HTML table, του οποίου το αποτέλεσμα φαίνεται στην παρακάτω εικόνα

5.2.2 Δημιουργία RSS Feed

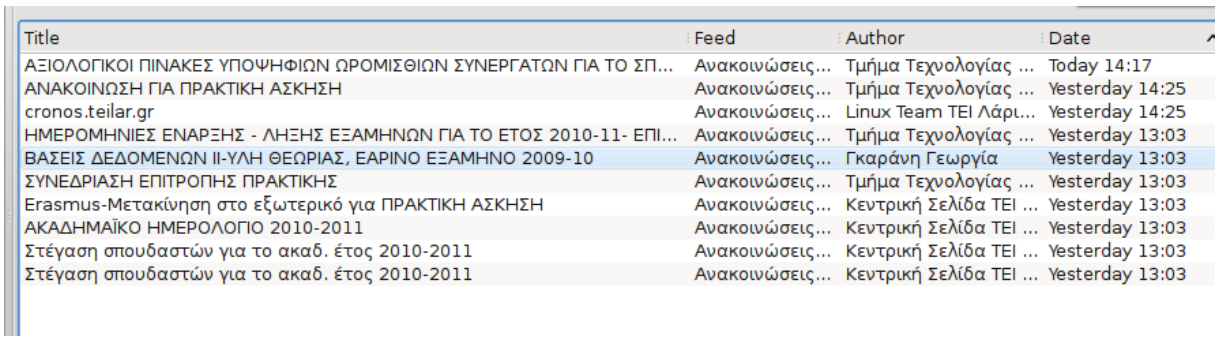
Πριν προχωρήσουμε στη δημιουργία χρηστών, ώστε να μπορεί ο καθένας να έχει τις ανακοινώσεις που τον ενδιαφέρουν, δοκιμάσαμε να εκτυπώσουμε τα αποτελέσματα σε ένα RSS Feed. Το Django διαθέτει κλάση η οποία δημιουργεί RSS Feeds από αναζητήσεις σε βάση δεδομένων, και ο κώδικας φαίνεται παρακάτω:

```
class AnnouncementFeed(Feed):

    title = 'Ανακοινώσεις ΤΕΙ Λάρισας'

    link = 'http://cronos.teilar.gr'

     description = 'Ανακοινώσεις διαφόρων ιστοσελίδων του ΤΕΙ Λάρισας. Παρέχονται από το http://cronos.teilar.gr'

def items(self):

    return Announcements.objects.all()[:30]
```

Μια επιπλέον ρύθμιση που απαιτήθηκε ήταν να αναφέρουμε φυσικά το σύνδεσμο που θα βρίσκονται οι ανακοινώσεις, κάνοντας τις εξής προσθήκες στο urls.py

```
feeds = { 'announcements': AnnouncementFeed }

(r'^feeds/(?P<url>.*)/$', 'django.contrib.syndication.views.feed', {'feed_dict': feeds}),
```

Δημιουργήσαμε ένα dictionary, και δηλώσαμε ως feed URL το /feeds/sannouncements. Με αυτές τις απλές γραμμές κώδικα καταφέραμε να δούμε τις ανακοινώσεις του ΤΕΙ μαζεμένες σε ένα RSS



Title	Feed	Author	Date
ΑΞΙΟΛΟΓΙΚΟΙ ΠΙΝΑΚΕΣ ΥΠΟΨΗΦΙΩΝ ΩΡΟΜΙΣΘΙΩΝ ΣΥΝΕΡΓΑΤΩΝ ΓΙΑ ΤΟ ΣΠ...	Ανακοινώσεις...	Τμήμα Τεχνολογίας ...	Today 14:17
ΑΝΑΚΟΙΝΩΣΗ ΓΙΑ ΠΡΑΚΤΙΚΗ ΑΣΚΗΣΗ	Ανακοινώσεις...	Τμήμα Τεχνολογίας ...	Yesterday 14:25
cronos.teilar.gr	Ανακοινώσεις...	Linux Team ΤΕΙ Λάρι...	Yesterday 14:25
ΗΜΕΡΟΜΗΝΙΕΣ ΕΝΑΡΞΗΣ - ΛΗΞΗΣ ΕΞΑΜΗΝΩΝ ΓΙΑ ΤΟ ΕΤΟΣ 2010-11- ΕΠΙ...	Ανακοινώσεις...	Τμήμα Τεχνολογίας ...	Yesterday 13:03
ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΙΙ-ΥΛΗ ΘΕΩΡΙΑΣ, ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2009-10	Ανακοινώσεις...	Γκαράνη Γεωργία	Yesterday 13:03
ΣΥΝΕΔΡΙΑΣΗ ΕΠΙΤΡΟΠΗΣ ΠΡΑΚΤΙΚΗΣ	Ανακοινώσεις...	Τμήμα Τεχνολογίας ...	Yesterday 13:03
Erasmus-Μετακίνηση στο εξωτερικό για ΠΡΑΚΤΙΚΗ ΑΣΚΗΣΗ	Ανακοινώσεις...	Κεντρική Σελίδα ΤΕΙ ...	Yesterday 13:03
ΑΚΑΔΗΜΑΪΚΟ ΗΜΕΡΟΛΟΓΙΟ 2010-2011	Ανακοινώσεις...	Κεντρική Σελίδα ΤΕΙ ...	Yesterday 13:03
Στέγαση σπουδαστών για το ακαδ. έτος 2010-2011	Ανακοινώσεις...	Κεντρική Σελίδα ΤΕΙ ...	Yesterday 13:03
Στέγαση σπουδαστών για το ακαδ. έτος 2010-2011	Ανακοινώσεις...	Κεντρική Σελίδα ΤΕΙ ...	Yesterday 13:03

5.3 Ταυτοποίηση στοιχείων στο Django

Το επόμενο σημαντικό βήμα ήταν η δημιουργία λογαριασμών χρηστών, ώστε ο κάθε χρήστης να μπορεί να διαλέγει ποιες ανακοινώσεις θέλει να παρακολουθεί. Το σύστημα διαχείρισης χρηστών του Django ήταν υπερπλήρες για τη δική μας περίπτωση, καθώς θα αναλάμβανε να κρατήσει τους χρήστες στη βάση δεδομένων του και να μπορούμε εύκολα να ορίζουμε δικαιώματα για τον κάθε χρήστη.

5.3.1 Περιγραφή συστήματος εγγραφής

Στην περίπτωση της δικής μας εφαρμογής, η εγγραφή ενός χρήστη δεν θα ήταν η κλασική του τύπου "παρακαλώ εισάγετε όνομα χρήστη και κωδικό πρόσβασης". Ο χρήστης θα έπρεπε από την αρχή να μας παραχωρήσει όλους τους λογαριασμούς που έχει στις διάφορες υπηρεσίες του ΤΕΙ ώστε να αποθηκευτούν από το σύστημά μας. Επίσης, κατά την εγγραφή του χρήστη όλα αυτά τα στοιχεία έπρεπε πρώτα να επαληθευτούν πριν προχωρήσουμε στη δημιουργία του λογαριασμού του χρήστη. Οι μόνες υπηρεσίες που απαιτούσαν λογαριασμό ήταν η Γραμματεία, το e-class και το Webmail. Η ταυτοποίηση θα έπρεπε πρώτα να γίνει με τα στοιχεία της γραμματείας, μιας και χωρίς κωδικό από τη γραμματεία δεν υφίσταται φοιτητής, και τα υπόλοιπα να θεωρηθούν προαιρετικά. Στα προαιρετικά όμως έπρεπε να γίνει έλεγχος και εκεί έλεγχος αν τα στοιχεία υπάρχουν ήδη. Τέλος, κατά την εγγραφή του χρήστη έπρεπε κατευθείαν να γίνουν parse όλα τα προσωπικά του στοιχεία (δήλωση, σχολή, εξάμηνο, βαθμολογία, μαθήματα eclass κτλ) και όλα μαζί να κρατηθούν στη βάση δεδομένων του Django

5.3.2 Ο πίνακας UserProfile

Δυστυχώς ο πίνακας των χρηστών στο Django δεν υποστήριζε όλα αυτά τα χαρακτηριστικά που θέλαμε, αλλά φυσικά η λύση ήταν και πάλι απλή. Δημιουργήθηκε ένας δεύτερος πίνακας ο οποίος κρατάει όλα αυτά τα χαρακτηριστικά και συνδέεται με σχέση ενα-προς-ένα με τον πίνακα των χρηστών. Μια πολύ απλή λύση, η οποία στην πορεία μπόρεσε να εξελιχθεί τρομερά, και με την παρακάτω γραμμή στο αρχείο admin.py να μπορέσουμε να δούμε και τον δεύτερο πίνακα στο admin panel να συνδέεται με τον πρώτο και να μας εμφανίζει τα προσωπικά στοιχεία των χρηστών (όπου UserProfile το όνομα.του προαναφερθέντος συμπληρωματικού πίνακα):

```
from cronos.user.models import *

from django.contrib import admin

admin.site.register(LdapProfile)
```

5.3.3 Η πολλαπλή φόρμα εγγραφής

Έχοντας ξεπεράσει το πρόβλημα του πού θα αποθηκευτούνε τα στοιχεία, έπρεπε να δημιουργηθεί η φόρμα εγγραφής, η οποία θα έπρεπε να αποτελούνταν από 4 διαδοχικές υποφόρμες που θα διαδέχονταν η μία την άλλη και θα ζητούσαν όλα τα απαραίτητα συνθηματικά. Η δημιουργία φόρμας στο Django γίνεται με τη χρήση της κλάσης Form, η οποία εκτός από απλές φόρμες υποστηρίζει και φόρμες τύπου Wizard, ιδανική για την περίπτωσή μας. Ο τρόπος λειτουργίας μιας φόρμας είναι απλός, και πριν προχωρήσουμε στη δημιουργία της φόρμας wizard θα δούμε μια απλή φόρμα αναζήτησης που δημιουργήσαμε για την ανάκτηση δεδομένων από την ιστοσελίδα της βιβλιοθήκης του ΤΕΙ Λάρισας.

```
from django import forms

class SearchForm(forms.Form):

    search = forms.CharField(label = 'Αναζήτηση', max_length = 100)
```

Στο παραπάνω απόσπασμα δημιουργήθηκε ένα στοιχείο τύπου αλφαριθμητικού. Στο view η φιλοσοφία που ακολουθείται είναι η εξής: ελέγχουμε αν έχουμε request από τον εξυπηρετητή, όπου θα εμπεριέχει και το στοιχείο της αναζήτησης. Αν είναι άδειο, τότε τυπώνουμε μια άδεια φόρμα. Αν όμως έχει κάποιο στοιχείο μέσα, τότε ελέγχουμε αν η φόρμα είναι έγκυρη (αλλιώς τυπώνεται μήνυμα λάθους) και αν έχει αποτελέσματα (αλλιώς τυπώνεται μήνυμα τύπου δεν υπάρχουν αποτελέσματα. Ένα γενικό view είναι το παρακάτω:

```
    if request.method == 'GET':

        form = SearchForm(request.GET)

        if form.is_valid():
```

```
                results = (...try to get results...)

                if (results == ''):

                        msg = 'Δεν υπάρχουν αποτελέσματα'

    else:

            form = SearchForm()

    return render_to_response('library.html', {

                'form': form,

                'msg': msg,

                'results': results,

            }, context_instance = RequestContext(request))
```

Στην περίπτωση της φόρμας τύπου wizard έπρεπε να δημιουργήσουμε τέσσερις φόρμες (δηλαδή 4 κλάσεις) με ονόματα SignupDionysos(forms.Form) SignupEclass(forms.Form) SignupWebmail(forms.Form) SignupCronos(forms.Form).

5.3.4 Εισαγωγή χρήστη στη βάση

Μετά το τέλος της εγγραφής, τα αποτελέσματα στέλνονται όλα μαζί στο view, οπότε μας είναι εύκολο να πάρουμε το γενικό request και να ελέγξουμε αν καθένα από τα αποτελέσματα είναι έγκυρο, και ανάλογα να κάνουμε raise exception με το κατάλληλο μήνυμα. Στην περίπτωση που όλα είναι έγκυρα, πρέπει πρώτα να κάνουμε parse όλα τα προσωπικά στοιχεία του χρήστη από τη σελίδα της γραμματείας και του eclass, και μετά απλά να τα κάνουμε όλα μαζί εισαγωγή στη βάση ως εξής:

```
user = User( username = request.POST.get('username'), first_name = first_name, last_name = last_name, email = email)

user.is_staff = None
```

```
user.is_superuser = None

user.set_password(request.POST.get('password')

user.save()
```

Μέχρι εδώ αποθηκεύσαμε έναν απλό χρήστη στη βάση δεδομένων, όπου δεν έχει πρόσβαση στο admin panel,και ο κωδικός του δημιουργείται από τη συνάρτηση set_password όπου τον κρυπτογραφεί κατευθείαν, και τέλος τον αποθηκεύουμε. Ακολουθεί η είσοδος των υπόλοιπων στοιχείων, στον πίνακα UserProfile

```
userprofile = UserProfile(user = user, dionysos_username =
request.POST.get('dionysosUsername'), ....)

userprofile.save()
```

Μετά με μία απλή φόρμα μπορέσαμε να δημιουργήσουμε και το σύστημα όπου ο χρήστης κάνει εισαγωγή των στοιχείων του, και με μια απλή αναζήτηση στη βάση ταυτοποιούσαμε τα στοιχεία του.

5.4 Κύρια σελίδα του χρήστη

5.4.1 Προβολή στοιχείων του χρήστη

Έχοντας στην κατοχή μας όλα τα στοιχεία του χρήστη στη βάση δεδομένων, με απλά ερωτήματα στη βάση μπορούσαμε να ανακτήσουμε το οποιοδήποτε και να το εκτυπώσουμε στην κατάλληλη σελίδα. Οπότε δημιουργήθηκε η αρχική σελίδα που περιείχε τα στοιχεία του χρήστη, η σελίδα της γραμματείας που περιείχε τη βαθμολογία και τη δήλωση του, η σελίδα E-Class με τα μαθήματα που παρακολουθεί ο χρήστης και η σελίδα με τα emails του. Τα emails δεν κρατούνται στη βάση δεδομένων, αλλά γίνονται parse την ώρα που ο χρήστης εισέρχεται στη σελίδα.

5.4.2 Άλλες Υπηρεσίες

Παράλληλα, το μενού των περιεχομένων αυξήθηκε με μια λίστα όλων των καθηγητών και με δυνατότητα αναζήτησης βιβλίων από τη σελίδα της βιβλιοθήκης.

Στην πρώτη περίπτωση τα δεδομένα ανακτούνται από τον πίνακα Id με την εξής εντολή:

```
for item in Id.objects.filter(urlid__startswith = 'pid')

  teacher.append([item.name, item.email, item,school])
```

Στην περίπτωση της βιβλιοθήκης τα δεδομένα γίνονται parse στο κατάλληλο URL ανάλογα με το στοιχείο της αναζήτησης

5.4.3 Η σελίδα ρυθμίσεων του χρήστη

Πλέον υπήρχαν συναρτήσεις για την ανάκτηση των στοιχείων του χρήστη και για την προσθήκη τους στη βάση δεδομένων. Οπότε επόμενο ήταν η δημιουργία από πολλές φόρμες, καθεμία από τις οποίες θα εξυπηρετούσε το ρόλο ανανέωσης συγκεκριμένων στοιχείων του χρήστη (βαθμολογία, δήλωση, όνομα χρήστη ή συνθηματικό σε κάποια υπηρεσία). Το τελευταίο που έπρεπε να γίνει όμως ήταν και η επιλογή των συγκεκριμένων ανακοινώσεων που ενδιέφεραν το χρήστη. Η πρώτη λύση ήταν η αυτόματη δημιουργία φόρμας τύπου πολλαπλών κουτιών επιλογής (checkbox), όμως δημιουργούνταν το πρόβλημα ότι ο χρήστης αν ήθελε απλά να προσθέσει έναν καινούργιο καθηγητή στη λίστα του, έπρεπε να ξαναεπιλέξει τους παλιούς.

Γι αυτό και στραφήκαμε στην επιλογή του multiple input box, ενισχυμένο με jQuery. Στην ουσία χρειαζόμασταν δύο κουτιά, το ένα με τους καθηγητές που θέλουμε να παρακολουθούμε και ένα με τους υπόλοιπους και θα έπρεπε να μας δίνεται η ικανότητα να μπορούμε να μεταφέρουμε κάποιο όνομα από το ένα κουτί στο άλλο. Παράλληλα, με έναν έλεγχο στη βάση δεδομένων θα μπορούσαμε να βλέπουμε ποια ονόματα είναι ήδη στη λίστα μας, ώστε να εμφανίζονται στο κατάλληλο κουτί. Την πλήρη λίστα των ονομάτων φυσικά θα την ανακτούσαμε από τον πίνακα Id. Ακολουθεί ο κώδικας του view:

```
    teacher_announcements_selected = []

    try:

        for teacher in Id.objects.filter(urlid__in =
request.user.get_profile().teacher_announcements.split(',')).order_by('name'):

            teacher_announcements_selected.append([teacher.urlid, teacher.name])
```

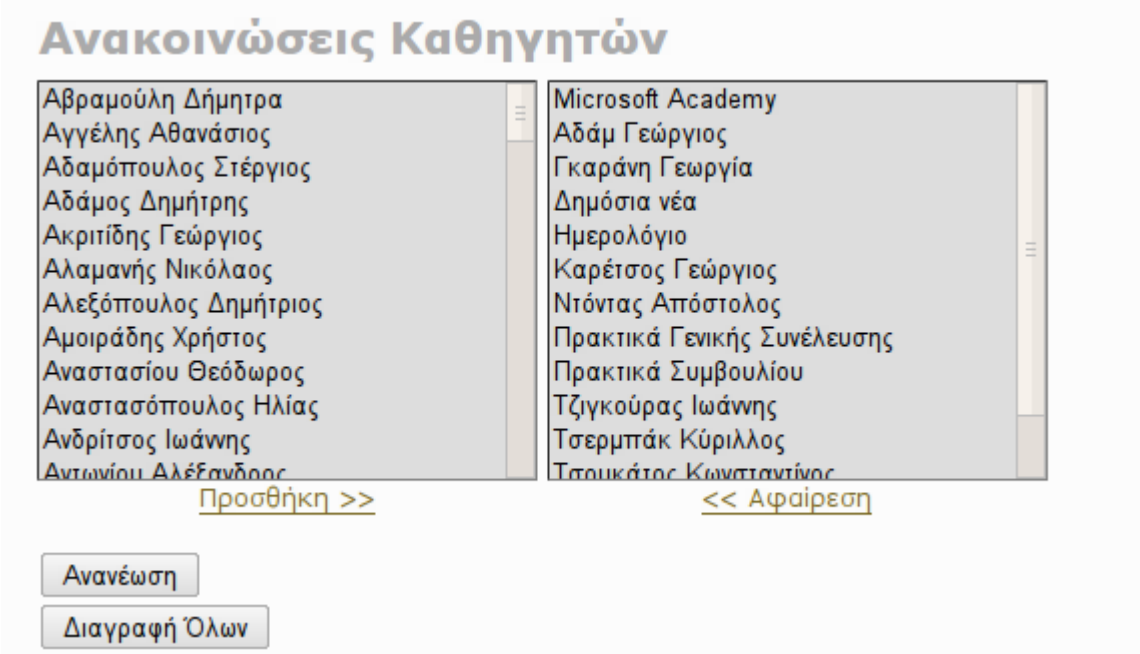
```
    except:
        pass
    teacher_announcements_all = []
    teachers = Id.objects.filter(urlid__startswith = 'pid').order_by('name')
    if request.user.get_profile().teacher_announcements:
        teachers = teachers.exclude(urlid__in =
request.user.get_profile().teacher_announcements.split(','))
    for teacher in teachers:
            teacher_announcements_all.append([teacher.urlid, teacher.name])
```

Στην ουσία δημιουργούμε δύο array, ένα με τους καθηγητές που έχουν επελεγεί, και ένα το οποίο απορρίπτει όσους είναι στο συγκεκριμένο array και εκτυπώνει τους υπόλοιπους της βάσης. Ο κώδικας του jQuery είναι ο παρακάτω:

```
$().ready(function() {
    $('#add1').click(function() {
        return !$('#id_teacher_announcements
option:selected').remove().appendTo('#teacherann_selected');
    });
    $('#remove1').click(function() {
        return !$('#teacherann_selected
option:selected').remove().appendTo('#id_teacher_announcements');
    });
    $('#teacherann').submit(function() {
```

```
            $('#teacherann_selected option').each(function(i) {

                $(this).attr("selected", "selected");

            });

        });

});
```

Η λειτουργία του είναι να μεταφέρει τα στοιχεία από το ένα κουτί στο άλλο, και όταν πατηθεί το κουμπί της υποβολής να επιλέγει αυτόματα όλα τα στοιχεία του δεύτερου κουτιού και να τα στέλνει σαν παράμετρο στη φόρμα.



5.4.4 Μικρή αλλαγή στην προβολή των ανακοινώσεων

Πλέον οι επιλεγμένες ανακοινώσεις κρατιούνται στον πίνακα UserProfile του κάθε χρήστη και γι αυτό το λόγο έπρεπε το announcements/views.py να τροποποιηθεί κατάλληλα ώστε να φιλτράρει τις ανακοινώσεις που μας ενδιαφέρουν. Υπάρχουν δύο πεδία, ένα το οποίο περιέχει τα pid των καθηγητών (πεδίο teacher_announcements) και ένα που περιέχει cid άλλων υπηρεσιών (πεδίο other_announcements). Ακολουθεί ο κώδικας:

```
all_announcements = []

all_announcements_ids = [request.user.get_profile().school]

try:

    all_announcements_ids += request.user.get_profile().eclass_lessons.split(',')

    all_announcements_ids += request.user.get_profile().other_announcements.split(',')

    all_announcements_ids += request.user.get_profile().teacher_announcements.split(',')

except:

    pass

for item in Announcements.objects.filter(urlid__urlid__in = all_announcements_ids
).order_by('-date_fetched')[:30]:

    all_announcements.append([item.author(), length, img, str(item.id), item.__unicode__(),
date])
```

ΚΕΦΑΛΑΙΟ 6ο: ΥΠΗΡΕΣΙΑ ΚΑΤΑΛΟΓΟΥ opneLDAP

6.1Ενίσχυση με LDAP backend

6.1.1 Περιγραφή του LDAP server

Το LDAP (Lightweight Directory Access Protocol) είναι ένα πρωτόκολλο το οποίο έχει τη δυνατότητα να αποθηκεύει δεδομένα, να τα οργανώνει και να παρέχει πρόσβαση σε αυτά ανάλογα με τον καθορισμό των δικαιωμάτων. Κάθε στοιχείο ανήκει σε ένα σύνολο "γρουπ" που ονομάζονται objectClass και ανάλογα παίρνει έξτρα χαρακτηριστικά. Είναι δηλαδή μια υπηρεσία καταλόγου, μια ειδική βάση δεδομένων και χρησιμοποιείται για συχνά ερωτήματα αλλά για μη συχνές ενημερώσεις. Διαφέρει από τις γενικές βάσεις δεδομένων στο γεγονός ότι αυτές δεν προσφέρουν υποστήριξη για απομακρυσμένη αλληλεπίδραση ή λειτουργικότητα roll-back. Η δομή του είναι δενδρική, οπότε όλα τα δεδομένα είναι ιεραρχικά δομημένα, οπότε όταν θα θελήσουμε να εισάγουμε δεδομένα μέσα στην υπηρεσία καταλόγου, θα πρέπει να ακολουθήσουμε ακριβώς το δέντρο ώστε η υπηρεσία να ξέρει που θα τοποθετήσει το νέο δεδομένο.

Οι χρήσεις του είναι απεριόριστες. Μπορεί να χρησιμοποιηθεί για κεντρική διαχείριση χρηστών, κρατώντας όλους τους λογαριασμούς σε μια συγκεκριμένη περιοχή στον LDAP. Άλλες χρήσεις του είναι η κοινή χρήση κλειδιών ασφαλούς κελύφους ανάμεσα από τα στοιχεία, η ενοποιημένη διαχείριση κόμβων κτλ

6.1.2 Αρχική Ρύθμιση

Για τις ανάγκες της πτυχιακής χρησιμοποιήθηκε το λογισμικό openLDAP. Η εγκατάστασή του έγινε με την εξής εντολή:

```
# emerge -av openldap
```

Τα αρχείο ρυθμίσεών του βρίσκεται στο φάκελο /etc/openldap, και πρώτο και βασικό είναι το slapd.conf, όπου δηλώνουμε το όνομα χρήστη και κωδικό πρόσβασης του υπερχρήστη. Ο κωδικός πρόσβασης δημιουργείται κρυπτογραφημένος με την εντολή slappasswd, και απλά αντιγράφουμε το hash στο εν λόγω αρχείο:

```
rootdn    'cn=root,dc=teilar,dc=gr'

rootpw    {SSHA}Ezp6I82DrNw+ou5lYiXhGXsPs0w2Xg4
```

Στη συνέχεια θα πρέπει να δημιουργήσουμε ένα δικό μας schema, το οποίο θα περιέχει τα attributes και το objectClass teilarStudent. Τα attributes θα είναι σε πλήρη αντιστοιχία με τα πεδία του πίνακα UserProfile (τον οποίο και μετονομάσαμε σε LdapProfile). Ακολουθεί ένα απόσπασμα από το teilar.schema

```
attributetype ( 2.51.20100504.20100608.32.13 NAME 'teacherAnnouncements'
        DESC 'teacher Announcements pids'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.44{5} )
attributetype ( 2.51.20100504.20100608.32.14 NAME 'otherAnnouncements'
        DESC 'other Announcements ids'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.44{5} )
attributetype ( 2.51.20100504.20100608.32.16 NAME 'cronosEmail'
        DESC 'Email defined by user, @teilar.gr is the default if applicable'
        SUP name )
attributetype ( 2.51.20100504.20100608.32.17 NAME 'grades'
        DESC 'dionysos.teilar.gr grades'
```

```
        SUP name )

objectclass ( 2.51.20100504.20100608.33.1 NAME 'teilarStudent'

        DESC 'Credentials of a Student of Tei Larissas'

        AUXILIARY

        MUST ( registrationNumber $ dionysosUsername $ dionysosPassword $ semester $
introductionYear $ school )

        MAY ( declaration $ eclassUsername $ eclassPassword $ eclassLessons $
webmailUsername $ webmailPassword $ teacherAnnouncements $ otherAnnouncements $
teilarAccess $ cronosEmail $ grades ) )
```

Η δομή του είναι η εξής: δίνεται στην αρχή ένας μοναδικός αριθμός και ένα όνομα για το attribute, και μετά η περιγραφή του. Τέλος, ή δηλώνουμε κάποιον από τους υπάρχοντες τύπους δεδομένων ή κληρονομούμε τις ιδιότητες από κάποιο άλλο (πολύ συχνά κληρονομούσαμε το NAME το οποίο υποστήριζε και ελληνικούς χαρακτήρες).

Δηλώνουμε το αρχείο teilar.schema στη λίστα με τα άλλα schema στο αρχείο ρυθμίσεων slapd.conf και ξεκινούμε την υπηρεσία στο σύστημά μας

```
# /etc/init.d/slapd start
```

6.1.3 Εγκατάσταση phpLDAPadmin

Έπειτα εγκαθιστούμε το phpldapadmin στο σύστημά μας, το οποίο είναι μια εύχρηστη διεπαφή web για LDAP

```
# emerge -av phpldapadmin

# webapp-config -I -h localhost -d phpldapadmin phpldapadmin 1.2.0.4
```

6.1.4 Ενσωμάτωση του LDAP στο Cronos

Το Django υποστηρίζει εναλλακτικά authentication backends πέρα από το δικό του ενσωματωμένο σύστημα της βάσης δεδομένων. Στην ουσία, κατά την εισαγωγή των στοιχείων το σύστημα καλεί τη συνάρτηση authentication() η οποία κάνει τον έλεγχο για το αν τα στοιχεία υπάρχουν στη βάση και κάνει τις κατάλληλες ενέργειες. Παράλληλα όμως, δίνεται η δυνατότητα αλλάζοντας απλά μια παράμετρο στο settings.py να μην καλείται η συνάρτηση authentication του django αλλά μια χειρόγραφη συνάρτηση με το ίδιο όνομα η οποία δεν θα κάνει αναζήτηση στη βάση δεδομένων των χρηστών, αλλά σε κάποιο άλλο μέσο αποθήκευσης, όπως ο openLDAP. Ευτυχώς για μας, υπήρχε ήδη έτοιμο πακέτο το οποίο έκανε ακριβώς αυτή τη διεργασία, δηλαδή ταυτοποίηση των στοιχείων από LDAP server. Στο settings.py έγιναν οι εξής αλλαγές:

```
BIND_USER = 'uid=root,dc=teilar,dc=gr'

BIND_PASSWORD = 'secret'

LDAP_SERVER = 'localhost'

LDAP_POERT = 389

LDAP_URL = 'ldap://%s%s' % (LDAP_SERVER, LDAP_PORT)

SEARCH_DN = 'ou=teilarStudents,dc=teilar,dc=gr'
```

```
SEARCH_FIELDS = ['*']

AUTHENTICATION_BACKENDS = (

'cronos.ldap_groups.accounts.backends.ActiveDirectoryGroupMembershipSSLBackend',

'django.contrib.auth.backends.ModelBackend',

)

AUTH_PROFILE_MODULE = 'user.LdapProfile'
```

Προστέθηκαν οι καθολικές μεταβλητές που αφορούν τον LDAP, καθώς και η μεταβλητή AUTHENTICATION_BACKENDS πλέον έχει ως πρώτη επιλογή τη δικιά μας κλάση, και η κλάση του Django υπάρχει ως δεύτερη σε περιίπτωση που αποτύχει η πρώτη. Στο κεφάλαιο 3 είδαμε τη χρήση του python-ldap module για αλληλεπίδραση με τον LDAP. Όλες οι διεργασίες που αναφέρθηκαν έχουν ενσωματωθεί σε συναρτήσεις, και χρησιμοποιήθηκαν πολύ εύκολα. Ακολουθεί η συνάρτηση authentication του ldap_groups

```
def bind_ldap(self, username, password):
        try:
            ldap.set_option(ldap.OPT_X_TLS_CACERTFILE,settings.CERT_FILE)
        except AttributeError:
            pass
        ldap.set_option(ldap.OPT_REFERRALS,0)
        l = ldap.initialize(settings.LDAP_URL)
        l.set_option(ldap.OPT_PROTOCOL_VERSION, 3)
        binddn = "uid=%s,%s" % (username, settings.SEARCH_DN)
        l.simple_bind_s(binddn,password)
```

```
        return l
    def authenticate(self,username=None,password=None):
        try:
            if len(password) == 0:
                return None
            l = self.bind_ldap(username, password)
            l.unbind_s()
            return self.get_or_create_user(username,password)
        except ImportError:
            pass
        except ldap.INVALID_CREDENTIALS as e:
            pass
    def get_or_create_user(self, username, password):
```

Η συνάρτηση authentication στην ουσία συνδέεται με τον LDAP server και ελέγχει αν ο χρήστης υπάρχει ήδη στη βάση δεδομένων του Django ή πρέπει να δημιουργηθεί. Αν δεν υπάρχει, αναλαμβάνει η συνάρτηση get_or_create_user η οποία όμως δεν ήταν καθόλου παραμετροποιήσιμη και στην ουσία έπρεπε να ξαναγραφτεί, ώστε να συμπεριλαμβάνει όλα τα χαρακτηριστικά που έχουμε δηλώσει το LdapProfile και στο teilar.schema.

6.1.5 Τελικό συμπέρασμα

Η όλη διαδικασία σύνδεσης με τον LDAP δίνει νέες πτυχές στην εφαρμογή. Πλέον μπορεί πολύ εύκολα να γραφτεί μια δεύτερη υπηρεσία η οποία θα μπορεί ακόμα και να τρέχει σε ξεχωριστό μηχάνημα, αλλά να κρατάει τους ίδιους λογαριασμούς, όπως για παράδειγμα μια αντικατάσταση της ιστοσελίδας της Γραμματείας ή του e-class.

ΚΕΦΑΛΑΙΟ 7ο: ΑΣΦΑΛΕΙΑ ΣΤΟ CRONOS

7.1 Περιγραφή

Φτάνοντας στο τέλος της ιστοσελίδας, έμενε να δημοσιοποιηθεί, ώστε να λάβει εκτενείς ελέγχους από τους χρήστες. Πιστεύαμε ότι μόνο αυτοί μπορούσαν να μας αναφέρουν όλα τα προβλήματα που έχει η σελίδα (σημαντικά και μη) τα οποία είτε δεν μπορέσαμε να εντοπίσουμε μέχρι τώρα είτε δεν μας επηρέαζαν. Για παράδειγμα, έπρεπε να γίνει εγγραφή από φοιτητές και άλλων σχολών ώστε να σιγουρευτούμε ότι το σύστημα εγγραφής δουλεύει όπως σχεδιάστηκε. Όμως πριν ανακοινώσουμε το οτιδήποτε στο ευρύ κοινό, έπρεπε πρώτα να σιγουρευτούμε ότι όλοι οι εγγεγραμμένοι χρήστες θα ήταν απόλυτα προστατευμένοι και τα συνθηματικά και οι προσωπικές τους πληροφορίες δεν θα ήταν διαθέσιμα σε τρίτους, είτε αυτοί είναι άλλοι εγγεγραμμένοι χρήστες είτε άλλοι οι οποίοι εκτέλεσαν κακόβουλες επιθέσεις. Λόγω των διαφόρων τεχνολογιών που χρησιμοποιήθηκαν, έπρεπε να γίνει μια σειρά σε κάθε μία από αυτές ώστε να εξασφαλιστεί η εύρυθμη λειτουργία του όλου συστήματος, και παράλληλα να μη γίνει δύσχρηστο ως προς το χρήστη αλλά και ως προς τους διαχειριστές.

7.2 Αποθήκευση συνθηματικών του συστήματος

7.2.1 Χρησιμότητα του local_settings.py

Το σύστημά μας χρειαζόταν να κάνει ανάκτηση των συνθηματικών για το eclass, το χρήστη cronos της MySQL και το λογαριασμό root του LDAP. Όλα τα παραπάνω συνθηματικά αποθηκεύτηκαν στο αρχείο local_settings.py αντί του αρχείου settings.py. Το local_settings.py γινόταν import από το settings.py και ήταν αναγνώσιμο μόνο από τον χρήστη apache, και εγγράψιμο από κανέναν (εκτός φυσικά από τον χρήστη root του συστήματος)

```
# ls -l local_settings.py

-r-x------ 1 apache root 1138 Jun 22 13:04 local_settings.py
```

Μέσα το αρχείο μεταξύ άλλων υπάρχουν δηλωμένες οι εξής μεταβλητές:

```
DATABASE_USER = 'user'

DATABASE_PASSWORD = 'secret'

ECLASS_USER = 'user'

ECLASS_PASSWORD = 'secret'

BIND_USER = 'uid=root,dc=teilar,dc=gr'

BIND_PASSWORD = 'user'
```

Η κλήση των παραπάνω μεταβλητών γίνεται με την συάρτηση settings του Django:

```
>>> from django.conf import settings

>>> print settings.ECLASS_USER

'user'
```

7.2.3 Hardened Profile

Οπότε ο επιτιθέμενος θα έπρεπε να πάρει πρόσβαση root στον εξυπηρετητή ώστε να ανακτήσει τα παραπάνω συνθηματικά. Για την προστασία του εξυπηρετητή χρησιμοποιήθηκαν εργαλεία του Hardened Project του Gentoo Linux. Το Hardened Project είναι το υπεύθυνο Project που ειδικεύεται στην ασφάλεια των εξυπηρετητών, και φροντίζει ο πυρήνας και όλα τα διαθέσιμα πακέτα να είναι πλήρως ενημερωμένα με όλες τις ενημερώσεις ασφάλειας σε όλες τις εκδόσεις των πακέτων. Η ανανέωση σε hardened σύστημα γίνεται εύκολα, διαλέγοντας hardened profile και κάνοντας update το σύστημα:

```
# eselect profile set hardened

# emerge -uDNav world –keep-going
```

7.2.3 Απόρριψη από το subversion

Καθώς όμως το local_settings.py πρέπει να βρίσκεται μέσα στα αρχεία του

αποθετηρίου, θα πρέπει και να βρεθεί ένας τρόπος ώστε το συγκεκριμένο αρχείο να μην εισάγεται στο αποθετήριο, καθώς δεν χρησιμοποιείται μόνο για την προστασία των κωδικών, αλλά και για την αποθήκευση προσωπικών ρυθμίσεων από σύστημα σε σύστημα (για παράδειγμα, ο χρήστης MySQL στον εξυπηρετητή να μπορεί να είναι διαφορετικός από αυτόν στο μηχάνημα που γίνεται η ανάπτυξη της εφαρμογής). Πέραν αυτού, στην περίπτωση που το αρχείο ανέβαινε στο αποθετήριο, θα γινόταν εμφανές όχι μόνο στους χρήστες που θα έκαναν svn clone τον κώδικα, αλλά και από όλους όσους θα έβλεπαν το WebSVN. Η απόρριψη ενός αρχείου από το αποθετήριο γίνεται εύκολα, απλά προσθέτοντας και το όνομα του αρχείου στο αρχείο .svnignore, το οποίο στην αρχή χρησιμοποιήθηκε για να αποτραπεί η αποστολή των binaries της python (*.pyc αρχεία) και του proj_root.py:

```
$ cat .svnignore

*.pyc

proj_root.py

local_settings.py
```

Με όλα τα παραπάνω καταφέραμε να κρατήσουμε ασφαλή τα συνθηματικά της εφαρμογής, είτε από τους απλούς χρήστες που έχουν πρόσβαση στον εξυπηρετητή είτε από τους svn χρήστες που συντελούν στην ανάπτυξη αυτής. Στη δεύτερη περίπτωση, οι προγραμματιστές που έχουν κάνει checkout τοπικά τον κώδικα, θα πρέπει ομοίως να δημιουργήσουν αρχείο local_settings.py το οποίο να περιέχει αντίστοιχα ονόματα χρήστη και συνθηματικά, τα οποία όμως θα ανταποκρίνονται στους MySQL και LDAP servers που τρέχουν τοπικά.

7.3 Αποθήκευση συνθηματικών του χρήστη για τις υπηρεσίες του TEI

7.3.1 Αλγόριθμος Blowfish

Κατά την εγγραφή ο χρήστης να καλείται να μας παραχωρήσει από ένα έως τρία συνθηματικά άλλων υπηρεσιών. Τα στοιχεία αυτά αποθηκεύονται στον LDAP server και στη βάση δεδομένων των χρηστών του Django. Οπότε έπρεπε να βρεθεί ένας δυνατός τρόπος κρυπτογράφησης των συνθηματικών, και η λύση βρέθηκε στον αλγόριθμο Blowfish, που υποστηρίζεται από την βιβλιοθήκη PyCrypto της python που προαναφέρθηκε. Η λογική ενός

αλγορίθμου κρυπτογράφησης είναι πολύ απλή: από δύο αριθμούς μπορεί εύκολα να παραχθεί ένας τρίτος, αλλά είναι εξαιρετικά δύσκολο από τον παραγόμενο να βρούμε τους πρώτους δύο. Αρχικά η κρυπτογραφία ήταν αποκλειστικό φαινόμενο του στρατού και των διαφόρων κυβερνητικών υπηρεσιών. Σήμερα υπάρχουν διάφοροι αλγόριθμοι κρυπτογράφησης όπως οι AES, Serpent, Blowfish και IDEA. Ένας αλγόριθμος κρυπτογράφησης θεωρείται ασφαλής όταν μείνει ελεύθερος για μεγάλο χρονικό διάστημα χωρίς να καταφέρει κάποιος να τον αποκρυπτογραφήσει εξ'ολοκλήρου ή μερικώς. Η κρυπτογραφία δεν χωρίζεται μόνο σε αλγόριθμους αλλά και σε τρόπους εφαρμογής. Ο τρόπος εφαρμογής είναι και το δυνατό σημείο ενός προγράμματος που χρησιμοποιεί την κρυπτογραφία.

Το Blowfish είναι ένα συμμετρικό μπλοκ κρυπτογράφησης αναπτυγμένο από τον Bruce Schneier. Ο αλγόριθμος χρησιμοποιεί 64 bit μπλοκ για κρυπτογράφηση και το μήκος του κλειδιού ποικίλει σε 448 bit. Αυτό παρέχει στο σύστημα ένα υψηλό επίπεδο ασφαλείας. Τα πιο σημαντικά χαρακτηριστικά του αλγόριθμου είναι: η ταχύτητά του, η πυκνότητά του και η απλότητά του. Ο Blowfish κρυπτογραφεί δεδομένα σε 32 bit μικροεπεξεργαστών σε μια ακτίνα των 18 χρονοκύκλων ανά bit. Μπορεί να τρέξει σε λιγότερο από 5 K μνήμης και η δομή του είναι πολύ απλή από την στιγμή που χρησιμοποιεί απλές λειτουργίες. Μέχρι στιγμής η ασφάλεια του Blowfish δεν έχει απειληθεί από οποιαδήποτε επίθεση.

Το κοινό κλειδί κρυπτογράφησης είναι ένας όρος που χρησιμοποιείται για να καλύψει την περιοχή της ασύμμετρης κρυπτογραφίας. Το κύριο χαρακτηριστικό όπου διαφέρει η ασύμμετρη κρυπτογραφία από την συμμετρική, είναι ότι χρησιμοποιούν διαφορετικό κλειδί για κρυπτογράφηση και αποκρυπτογράφηση. Επιπλέον έχουν το πλεονέκτημα ότι το κλειδί κρυπτογράφησης μπορεί να γίνει κοινό ενώ το κλειδί αποκρυπτογράφησης πρέπει πάντα να είναι κρυφό. Κάποιος μπορεί να αναφερθεί στο κλειδί κρυπτογράφησης σαν κοινό κλειδί και το κλειδί αποκρυπτογράφησης σαν μυστικό κλειδί.

7.3.2 Εφαρμογή στο Cronos

Ακολουθεί ο κώδικας υλοποίησης κρυπτογράφησης και αποκρυπτογράφησης κωδικού που χρησιμοποιήθηκε, με βάση τον αλγόριθμο κρυπτογράφησης Blowfish:

```
def encryptPassword(password):

    from Crypt.Cipher import Blowfish
```

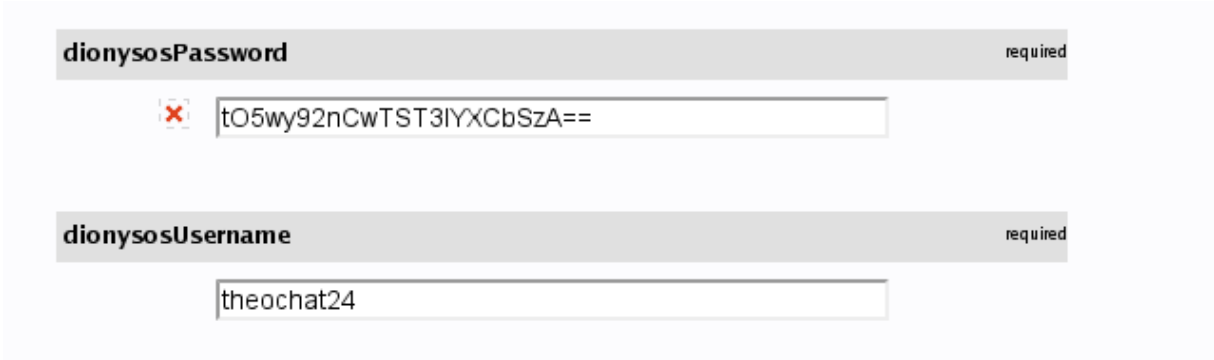
```
            from django.conf import settings
            from random import choice
            import base64
            import string


            obj = Blowfish.new(settings.BLOWFISH_KEY)
            if len(str(len(password))) == 1:
                length = '0' + str(len(password))
            else:
                length = str(len(password))
            new_password = length + password
            if len(new_password) % 8 != 0:
             new_password += (",join([choice(string.letters + string.digits) for i in range(8 –
len(new_password) % 8))]))
            return base64.b64encode(obj.encrypt(new_password))


        def decryptPassword(password):
            from Crypto.Cipher import Blowfish
            from django.conf import settings
            from random import choice
            import base64
```

```
    obj = Blowfish.new(settings.BLOWFISH_KEY)
    original_password = obj.decrypt(base64.b64decode(password))
    return original_password[2:int(original_password)[:2]) + 2]
```

Το Blowfish απαιτεί ο κωδικός να είναι πολλαπλάσιος του 8, πράγμα το οποίο ήταν απίθανο να γίνει για όλους τους χρήστες. Οπότε για να επιλυθεί το πρόβλημα θα έπρεπε πρώτα να γίνει έλεγχος αν ο κωδικός είναι όντως πολλαπλάσιος του 8, και αν δεν είναι, υπολογίζονται πόσα ψηφία υπολείπονται ώστε να πραγματοποιηθεί αυτό, και με τη χρήση της συνάρτησης random δημιουργούνται τυχαίοι χαρακτήρες οι οποίοι προστίθενται στο τέλος. Με τον τρόπο αυτό χάνεται όμως το μέγεθος του αρχικού συνθηματικού, το οποίο για αυτό το λόγο κρατείται σε έναν διψήφιο αριθμό στην αρχή, όπου το πρώτο ψηφίο είναι 0 αν ο κωδικός είναι μονοψήφιος. Το μυστικό κλειδί είναι αποθηκευμένο στο αρχείο local_settings.py στη μεταβλητή BLOWFISH_KEY. Το κρυπτογραφημένο αποτέλεσμα όμως εμφανίζεται ως στοιχεία μνήμης (σκουπίδια) στην οθόνη, και υπήρχε πρόβλημα αποθήκευσής του στον LDAP. Οπότε ως τελευταίο βήμα ήταν να επανακρυπτογραφηθεί το αποτέλεσμα, αυτή τη φορά όμως με τη συνάρτηση base64, η οποία στηρίζεται σε έναν πολύ αδύναμο αλγόριθμο κρυπτογράφησης, και ο σκοπός της πλέον είναι να παράγει ένα hash το οποίο θα αποτελείται από έγκυρους χαρακτήρες για το URL ή για άλλα μέσα αποθήκευσης.

Η αποκρυπτογράφηση στηρίζεται στον τρόπο που έγινε η κρυπτογράφηση: πρώτα αποκωδικοποιούμε το συνθηματικό με την base64 και αμέσως με το blowfish χρησιμοποιώντας το ίδιο μυστικό κλειδί. Το αποτέλεσμα είναι ένα αλφαριθμητικό, το οποίο γνωρίζουμε ότι οι δύο πρώτοι χαρακτήρες του είναι το μέγεθος του συνθηματικού, οπότε εκμεταλευόμενοι τον τρόπο που χειρίζεται η python τα αλφαριθμητικά, εκτυπώνουμε το αποτέλεσμα ξεκινώντας από τον τρίτο χαρακτήρα μέχρι όσο είναι το νούμερο των δύο πρώτων χαρακτήρων.

Με την παραπάνω διαδικασία καταφέραμε να χρησιμοποιήσουμε έναν ισχυρό αλγόριθμο κρυπτογράφησης, όπου το μυστικό κλειδί που απαιτείται είναι καλά προστατευμένο από όλους, και το αποτέλεσμα να είναι αδύνατο να φανερωθεί, δίνοντας παράλληλα τη δυνατότητα στο σύστημα να δουλεύει αυτόνομα και με ασφάλεια.



7.4 Αποθήκευση συνθηματικών του χρήστη για το Cronos

7.4.1 Αλγόριθμος SHA-1

Τα παραπάνω συνθηματικά έπρεπε να κωδικοποιηθούν με τρόπο που να μπορούν να αποκωδικοποιηθούν ώστε να επαναχρησιμοποιούνται εύκολα από το σύστημα όποτε ζητηθούν. Στην περίπτωση όμως του συνθηματικού του χρήστη για την υπηρεσία Cronos κάτι τέτοιο δεν ισχύει, αφού το συνθηματικό χρησιμοποιείται μόνο κατά την είσοδο του χρήστη όπου πραγματοποιείται και η ταυτοποίηση των στοιχείων του χρήστη. Για το λόγο αυτό χρησιμοποιήθηκε η κρυπτογραφική συνάρτηση κατακερματισμού SHA-1.

Η κρυπτογραφική συνάρτηση κατατεμαχισμού (cryptographic hash function) είναι μια συνάρτηση κατακερματισμού (hash function) η οποία είναι σχεδιασμένη για να χρησιμοποιείται στην κρυπτογραφία. Γενικά η συνάρτηση κατατεμαχισμού είναι μια μαθηματική συνάρτηση που έχοντας ως είσοδο μια αυθαίρετου μεγέθους ομάδα δεδομένων δίνει έξοδο μια καθορισμένου μεγέθους συστοιχεία (η συμβολοσειρά είναι συνήθως μικρότερη σε μέγεθος από την αρχική είσοδο). Η έξοδος δεν μπορεί με αντιστροφή (με κανένα τρόπο) να μας παράγει την αρχική είσοδο. Η έξοδος αποκαλείται συνήθως "σύνοψη" (digest).

Μια ιδεατή κρυπτογραφική συνάρτηση κατατεμαχισμού έχει τις παρακάτω ιδιότητες:

- Είναι εύκολο να υπολογιστεί η σύνοψη για οποιαδήποτε είσοδο.
- Δεν είναι εφικτό να βρεις την είσοδο από την σύνοψη.
- Δεν είναι εφικτό να τροποποιήσεις την είσοδο χωρίς να τροποποιηθεί η

σύνοψη.

- Δεν είναι εφικτό να βρεθούν δύο διαφορετικές είσοδοι που δίνουν την ίδια σύνοψη.

Οι συναρτήσεις κατατεμαχισμού βρίσκουν εφαρμογή στις εφαρμογές ασφάλειας πληροφοριών, ενδεικτικά στις ψηφιακές υπογραφές και άλλες μορφές πιστοποίησης αυθεντικότητας των δεδομένων.

Διάσημες συναρτήσεις κατατεμαχισμού είναι MD5 και η SHA-1. Το 2008 βρέθηκαν προβλήματα ασφάλειας στην συνάρτηση MD5 σε επίθεση στο πρωτόκολλο Secure Socket Layer (SSL). Η συνάρτηση SHA-0 και ηSHA-1 αναπτύχθηκαν από την Υπηρεσία Εθνικής Ασφάλειας (National Security Agency: NSA) των ΗΠΑ. Τον Φεβρουάριο 2005, μια επιτυχημένη επίθεση στην συνάρτηση SHA-1 δημοσιεύθηκε. Η θεωρητική αδυναμία της συνάρτησης είναι γνωστή και έτσι αναπτύχθηκε η συνάρτηση SHA-2.

7.4.2 Εφαρμογή στο Cronos

Ακολουθεί η συνάρτηση της Python που χρησιμοποιήθηκε για την κρυπτογράφηση του κωδικού πρόσβασης του χρήστη

```
def sha1Password(password):
    from base64 import encodestring as encode
    import base64
    import hashlib
    import os

    salt = os.urandom(4)
    h = hashlib.sha1(password)
    h.update(salt)
    return "{SSHA}" + encode(h.digest() + salt)
```

Η βάση δεδομένων του Django έχει δικιά της συνάρτηση για να δημιουργεί SHA-1 hash, η οποία συνοψίζεται στο παρακάτω απόσπασμα

```
from django.auth.models import User

user = User(username = 'username', first_name = 'firstname', last_name = 'lastname', email = 'email')

user.is_staff = False

user.is_superuser = False

user.set_password(password)

user.save()
```

Ως προεπιλογή, η συνάρτηση user.set_password() χρησιμοποιεί κρυπτογράφηση SHA-1 και αποθηκεύει το αποτέλεσμα στη μορφή sha1$salt$hash

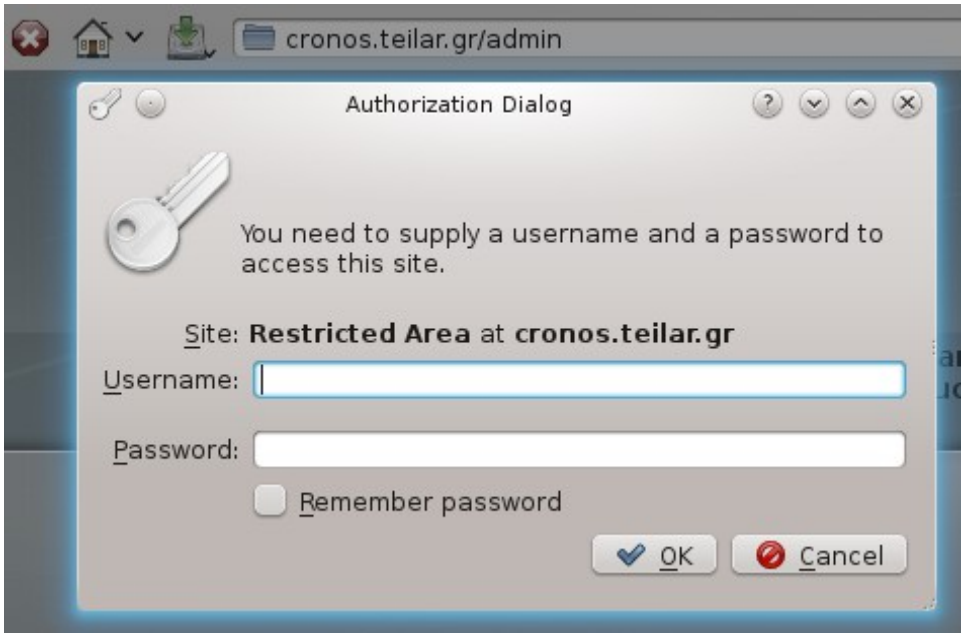
7.5 Προστασία στον Apache Web Server

7.5.1 Εφαρμογές Web που χρησιμοποιήθηκαν

Για την καλύτερη διαχείριση της εφαρμογής χρειάστηκε η εγκατάσταση βοηθητικών εφαρμογών web, όπως τα phpLDAPadmin και phpMyAdmin τα οποία προσέφεραν ένα εύχρηστο γραφικό περιβάλλον για τη MySQL και τον openLDAP server. Μια τρίτη αντίστοιχη υπηρεσία που χρησιμοποιήθηκε ήταν το Admin Panel που προσφέρει το Django. Όλα τα παραπάνω έχουν πρόσβαση στα (κρυπτογραφημένα μεν) συνθηματικά των χρηστών, καθώς και σε όλα τα προσωπικά στοιχεία τους, όπως το ονοματεπώνυμο και το email τους.

Όλες οι παραπάνω εφαρμογές ζητάνε όνομα χρήστη και κωδικό πρόσβασης για την εισαγωγή και κάθε μία από αυτές έχει ιδιαίτερο τρόπο για την ρύθμιση των δικαιωμάτων του χρήστη. Πριν όμως γίνει οποιαδήποτε ρύθμιση σε αυτές καθ'αυτές και στην αντίστοιχη υπηρεσία που τις συντροφεύει, πρώτα εξασφαλίστηκε από τον apache web server ότι κανένας μη εξουσιοδοτημένος χρήστης θα είχε πρόσβαση σε αυτές. Οπότε, έπρεπε να δημιουργήσουμε ένα επιπλέον σύνολο χρηστών το οποίο θα μπορούσε να εισέλθει στους υποκαταλόγους /phpmyadmin /phpldapadmin /admin και σε όλους τους θυγατρικούς τους. Η τεχνική γίνεται με τη δημιουργία ενός αρχείου εν ονόματι htpasswd, το οποίο αποθηκεύει μια

λίστα με ονόματα χρήστη και συνθηματικά κωδικοποιημένα σε SHA-1. Όποτε ο χρήστης ζητάει να εισέλθει στις παραπάνω ιστοσελίδες, ένα παράθυρο ζητάει τα στοιχεία του χρήστη και κλείνει μόνο αν τα στοιχεία επαληθευτούν. Ένα επιπλέον θετικό της τεχνικής αυτής είναι ότι τα διάφορα bot attacks δεν θα μπορέσουν καν να αντιληφθούν την ύπαρξη htpasswd, και απλά θα θεωρήσουν ότι η ιστοσελίδα δεν υπάρχει.



7.5.2 Δημιουργία και χρήση htpasswd

Η διαδικασία δημιουργίας αρχείου htpasswd είναι η εξής:

```
# htpasswd -c /etc/apache2/passwd tampakrap

New password:

Re-type new password:

Adding password for user tampakrap
```

Παρατηρούμε ότι το αρχείο έχει δημιουργηθεί (με τη σημαία -c) οπότε προσθέτουμε και επιπλέον χρήστη:

```
# htpasswd /etc/apache2/passwd cephalon
```

```
New password:

Re-type new password:

Adding password for user cephalon
```

Τα δικαιώματα του αρχείου είναι τα εξής:

```
# ls -l /etc/apache2/passwd

-rw-------  1 apache root   477 Jun  8 15:34 passwd
```

7.5.3 Αρχείο ρυθμίσεων του apache

Επόμενο βήμα είναι να δηλώσουμε την ύπαρξη του συγκεκριμένου αρχείου στο αρχείο ρυθμίσεων του apache:

```
...

<Directory "/var/www/cronos.teilar.gr/htdocs/phpldapadmin">

    DirectoryIndex index.php

    AllowOverride None

    Options FollowSymlinks

    Order allow,deny

    Allow from all

    AuthType Basic

    AuthName "Restricted Area"

    AuthUserFile /etc/apache2/passwd
```

```
        Require user cephalon tampakrap
        <limit GET POST>
            require valid-user
        </limit>
        RedirectMatch ^/phpldapadmin/$ /phpldapadmin/index.php
</Directory>
...
<Location "/">
        SetHandler python-program
        PythonHandler django.core.handlers.modpython
        PythonPath "['/home/code/'] + sys.path"
        SetEnv DJANGO_SETTINGS_MODULE cronos.settings
        PythonDebug On
        #PythonOption django.root /cronos
</Location>
...
Alias /media "/var/www/cronos.teilar.gr/htdocs/media"
<location "/media/">
        SetHandler None
</Location>
```

```
Alias /phpldapadmin "/var/www/cronos.teilar.gr/htdocs/phpldapadmin"
<Location "/phpldapadmin">
    SetHandler application/x-httpd-php
</Location>
<LocationMatch "\.(jpg|gif|png|css)$">
    SetHandler None
</LocationMatch>
CustomLog /var/log/apache2/cronos.teilar.gr/access.log combined
ErrorLog /var/log/apache2/cronos.teilar.gr/error.log
LogLevel warn
...
```

Δηλώσαμε σε ετικέτες Directory όλες τις φυσικές διαδρομές που υπάρχουν οι διάφορες υπηρεσίες, και δηλώσαμε τον κατάλληλο χειριστή για αυτές, είτε αυτός είναι το python module του apache είτε το php module. Στις υπηρεσίες που μας ενδιέφερε να είναι προστατευμένες υπάρχουν οι εντολές Auth*. Παρατηρούμε ότι στα σημεία που καλείται το συγκεκριμένο αρχείο htpasswd, δηλώνουμε επ'ακριβώς τους χρήστες οι οποίοι θέλουμε να έχουν πρόσβαση στην υπηρεσία που περιλαμβάνεται μέσα στην ετικέτα Directory.

Μια άλλη μορφή προστασίας που χρησιμοποιήθηκε για τον εν λόγω web server ήταν ο χειρισμός των στατικών αρχείων, όπως εικόνες και αρχεία CSS. Ο apache είναι πολύ πιο γρήγορος και αποδοτικός στον χειρισμό τέτοιων αρχείων αντί του python module για τον apache. Γι αυτό και δηλώσαμε όλες τις διαδρομές (είτε φυσικές, είτε διαδρομές URL) που περικλείουν στατικά αρχεία και δηλώσαμε SetHandler None, το οποίο αποδεσμεύει το mod_python από το χειρισμό των αρχείων. Το επιθυμητό αποτέλεσμα είναι η προστασία από επιθέσεις τύπου άρνησης πρόσβασης.

Στο τέλος έχουμε δηλώσει το μέρος όπου θα κρατούνται οι καταγραφές της κίνησης,

ώστε να γνωρίζουμε την κίνηση της σελίδας και τυχόν προβλήματα που παρουσιάζονται.

7.6 Απόκρυψη στοιχείων στον LDAP server

Αν και η ύπαρξη του μυστικού κλειδιού και του htpasswd στον apache φαίνονται αρκετές ώστε να κρατήσουν μακρυά τον επιτιθέμενο από τα στοιχεία των εγγεγραμμένων χρηστών, παρ' όλα αυτά παραμένει το πρόβλημα των χρηστών που έχουν πρόσβαση στον εξυπηρετητή (ή την αποκτούν με πλάγιο τρόπο). Οι χρήστες αυτοί, αν έχουν πρόσβαση και στο Cronos, μπορούν να εκτελέσουν εντολές αναζήτησης στον LDAP και να έχουν εύκολα πρόσβαση στα στοιχεία των εγγεγραμμένων χρηστών. Αυτό μπορεί να γίνει είτε χρησιμοποιώντας το python-ldap module ή την εντολή ldapsearch

```
$ ldapsearch -x -D 'uid=user,ou=teilarStudents,dc=teilar,dc=gr' -W
```

Για να προστατευτούν πλήρως τα στοιχεία των χρηστών και να γίνουν διαθέσιμα μόνο από τον υπερχρήστη του LDAP, εφαρμόστηκαν οι εξής τροποποιήσεις στο αρχείο ρυθμίσεων του LDAP Server /etc/openldap/slapd.conf:

```
access to dn.base="" by * read

access to dn.base="uid=Subschema" by * read

access to attrs =
userPassword,dionysosUsername,dionysosPassword,eclassUsername,eclassPassword,webma
ilUsername,webmailPassword,semester,school,cn,sn,declaration,eclassLessons,grades,introd
uctionYear,registrationNumber,cronosEmail,teacherAnnouncements,otherAnnouncements

        by anonymous auth

        by self write

        by * none

access to *

        by self write
```

```
        by users read

        by anonymous auth
```

Με το παραπάνω αρχείο, δίνουμε τη δυνατότητα στους ανώνυμους χρήστες να μπορούν μόνο να κάνουν ταυτοποίηση των στοιχείων τους, στους ταυτοποιημένους χρήστες τη δυνατότητα να διαβάζουν και να τροποποιούν όλες τις μεταβλητές τους και να μπορούν να μην προβάλουν όλες τις μεταβλητές των άλλων χρηστών που βρίσκονται στη λίστα με τα attrs. Η λίστα attrs περιλαμβάνει όλα τα ονόματα χρήστη (εκτός αυτού του Cronos) και όλα τα συνθηματικά, μαζί με πολλά προσωπικά στοιχεία.

Με όλες τις παραπάνω ενέργειες καταφέραμε να κάνουμε εξαιρετικά δύσκολη τη δυνατότητα κάποιου να μπορέσει να κάνει την οποιαδήποτε ενέργεια κατευθείαν στον LDAP server, αλλά και στην ακραία περίπτωση που ο επιτιθέμενος το καταφέρει, να μην μπορέσει να αποσπάσει σημαντικές πληροφορίες. Πλέον μόνο ο υπερχρήστης του LDAP Server μπορεί να προβάλει τις όλες μεταβλητές των χρηστών ανεξαρτήτως και να τις τροποποιήσει. Ο κωδικός του υπερχρήστη έχει δηλωθεί στο local_settings.py και είναι αρκετά ισχυρός.

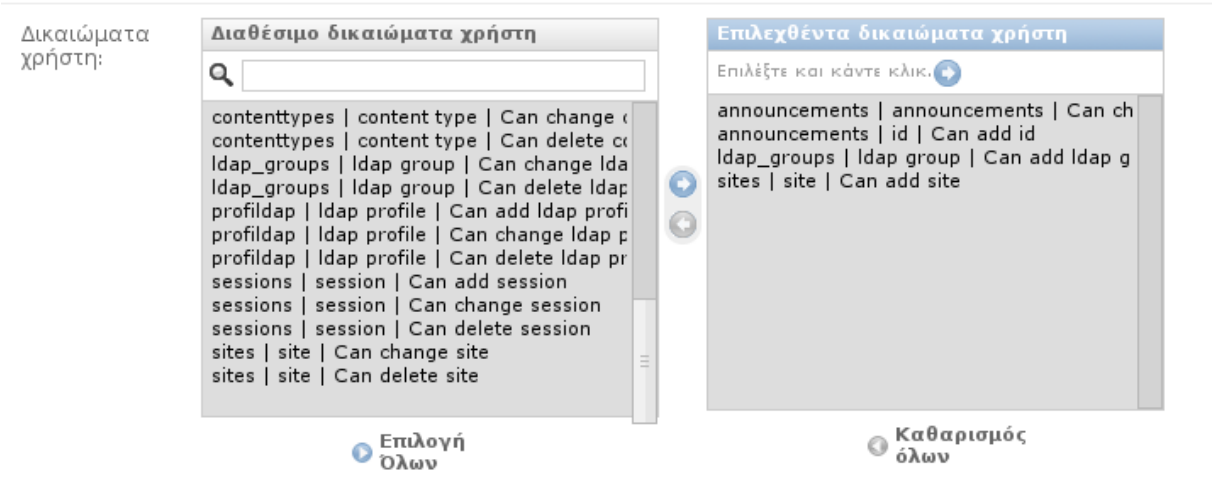
7.7 Ασφάλεια στην MySQL

Η MySQL προστατεύεται από μόνη της από χρήστες. Ο χρήστης που έχει πρόσβαση στη βάση δεδομένων του Cronos είναι και ο χρήστης ο οποίος μπορεί στην ουσία να κάνει ταυτοποίηση των στοιχείων του στο phpMyAdmin, καθώς φυσικά και ο υπερχρήστης. Και οι δύο χρήστες προστατεύονται από ισχυρά συνθηματικά, και ο χρήστης που έχει πρόσβαση στη βάση δεδομένων του Cronos έχει επίσης δηλωθεί στο local_settings.py. Η μέθοδος αυτή, σε συνδιασμό με το htpasswd, είναι αρκετά ασφαλής, αφού οι εγγεγραμμένοι χρήστες δεν μπορούν απλά με τα στοιχεία που έχουν στο Cronos να κάνουν και εισαγωγή στη MySQL.

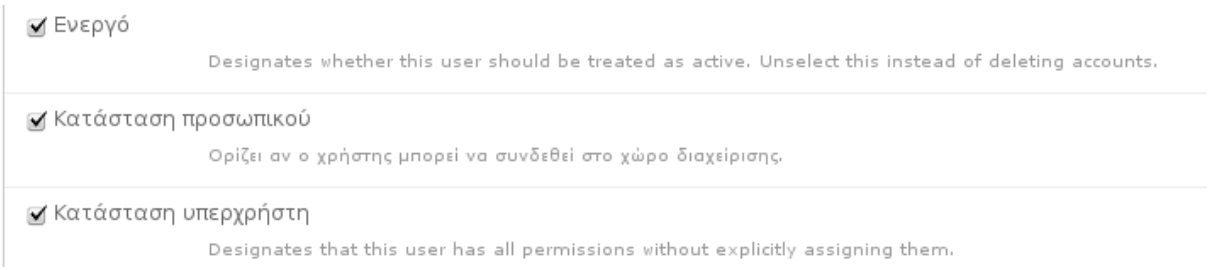
7.8 Χειρισμός δικαιωμάτων στο Django

Το Django έχει έναν άψογο χειρισμό των δικαιωμάτων των χρηστών του. Κατά τη δημιουργία του χρήστη μπορεί να επιλεγεί αν ο εν λόγω χρήστης θα έχει πρόσβαση στο admin panel, και μάλιστα τι δικαιώματα θα έχει μέσα σε αυτό. Επίσης, ο οποισδήποτε

χρήστης μπορεί να οριστεί ως υπερχρήστης (από κάποιον άλλο υπερχρήστη φυσικά). Η επιλογή των δικαιωμάτων γίνεται με το παρακάτω κουτί επιλογών:



Υπάρχει μια πλήρης λίστα δικαιωμάτων, και μάλιστα μπορούμε εύκολα να δημιουργήσουμε και νέα. Στην περίπτωση της δικής μας εργασίας, απλά περιορίσαμε την πρόσβαση στο admin panel από όλους τους χρήστες, το οποίο είναι αρκετά ασφαλές σε συνεργασία με το htpasswd.



7.9 Αποφυγή sniffing

Ένα network packet sniffer είναι λογισμικό με δυνατότητες παρακολούθησης και καταγραφής της κίνησης ή αλλιώς των πακέτων δικτύου που αποστέλλει και δέχεται η διεπαφή δικτύου (network interface) του Η/Υ.

Σε δίκτυα συνδεδεμένα με διανομείς (network hubs) και ασύρματα δίκτυα, ένα τέτοιο λογισμικό μπορεί να καταγράψει και τα πακέτα που απευθύνονται σε άλλους Η/Υ ή συσκευές του δικτύου. Αντίθετα σε δίκτυα συνδεδεμένα με μεταγωγείς (network switch) που παρέχουν

επιπλέον ασφάλεια, αυτό είναι εφικτό μόνο σε συγκεκριμένες περιπτώσεις και λόγω κάποιων αδυναμιών ασφαλείας των πρωτοκόλλων που χρησιμοποιούνται.

Όλες οι πλατφόρμες του TEI που οι φοιτητές πρέπει να εισάγουν όνομα χρήστη και κωδικό πρόσβασης λαμβάνουν τα δεδομένα σε μη κρυπτογραφημένη μορφή. Αυτό σημαίνει πως αν κάποιος βρίσκεται στο ίδιο δίκτυο με κάποιον χρήστη, είναι δυνατή η χρησιμοποίηση κάποιας άλλης τεχνικής ώστε να μπορέσει να πάρει κάποια πακέτα με την χρήση ενός network packet sniffer. Μέσα σε αυτά τα πακέτα είναι πολύ πιθανό να υπάρχουν και το όνομα χρήστη και ο κωδικός του.

Με το Cronos δεν υπάρχει αυτός ο κίνδυνος γιατί τα δεδομένα φτάνουν στον εξυπηρετητή μας σε κρυπτογραφημένη μορφή. Επίσης την επικοινωνία με τις διάφορες υπηρεσίες του TEI τις διεκπεραιώνει επίσης ο εξυπηρετητής μας και έτσι το πρόβλημα του packet sniffing σχεδόν εξαλείφεται.

ΕΠΙΛΟΓΟΣ: ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΕΡΑΙΤΕΡΩ ΕΞΕΛΙΞΗ

Τη στιγμή που γράφονται αυτές οι γραμμές, η εφαρμογή έχει ήδη κλείσει ένα μήνα κανονικής λειτουργίας και μέχρι στιγμής έχουν εγγραφεί 45 φοιτητές από διάφορες σχολές του ΤΕΙ Λάρισας. Τα σχόλια που έχουμε λάβει ως τώρα είναι παραπάνω από θετικά. Την ημέρα που γνωστοποιήσαμε την ιστοσελίδα ανοίξαμε και τον κώδικα και στην περιγραφή της κάναμε εμφανή τη διάθεσή μας για περαιτέρω ανάπτυξη της εφαρμογής από άλλους φοιτητές είτε στα πλαίσια κάποιας πτυχιακής εργασίας είτε απλά για εκμάθηση python και Django. Ήδη έχουν εμφανιστεί άτομα τα οποία μας έχουν αποστείλει τις προτάσεις τους, οι οποίες κυρίως αφορούσαν το design. Έχει γίνει ήδη κίνηση από φοιτητές να εξελίξουν την εν λόγω ιστοσελίδα, και μάλιστα και από φοιτητές εκτός του τμήματος Πληροφορικής. Ενδεικτικά τα θέματα των φοιτητών αφορούν ηλεκτρονική εγγραφή εργαστηρίων, Φυτολόγιο σε βάση δεδομένων για τους φοιτητές της Φυτικής Παραγωγής και Μηχανολογικό Σχέδιο για ψυγεία και προβολή σε ιστοσελίδα για τους φοιτητές Μηχανολογίας.

Η επαφή μας με τη γλώσσα προγραμματισμού Python ήταν πολύ μικρή πριν αναλάβουμε την πτυχιακή εργασία, αν και υπήρχε προηγούμενη εμπειρία σε άλλες γλώσσες προγραμματισμού και κυρίως scipting. Πολύ γρήγορα όμως καταφέραμε να ακολουθήσουμε τη φιλοσοφία της και να γράψουμε τα πρώτα scripts τα οποία δούλευαν άψογα, σε λίγες μόνο γραμμές κώδικα. Χαρακτηριστικό παράδειγμα ήταν η χρήση του PycURL για την εισαγωγή στις διάφορες υπηρεσίες του ΤΕΙ, η οποία γίνεται μόνο σε 5 γραμμές κώδικα, και μάλιστα χειρίζεται άψογα την ύπαρξη cookies για την είσοδο με συγκεκριμένο λογαριασμό. Η δομή της γλώσσας, με την απώλεια των αγκυλών και την αναγκαστική σύνταξη του κώδικα μας επέτρεψε να δημιουργήσουμε καθαρό και ευανάγνωστο κώδικα από την αρχή της συγγραφής, με αποτέλεσμα την πανεύκολη εξέλιξή του κατά την πορεία της εργασίας. Ευελπιστούμε το συγκεκριμένο μεγάλο προτέρημα θα αντιληφθούν και οι συμφοιτητές μας και θα το εκμεταλλευτούν στο έπακρο για την πιθανή εξέλιξη της εφαρμογής. Φυσικά και τα διάφορα modules που κυκλοφορούν στο διαδίκτυο μας επέτρεψαν να δημιουργήσουμε με λίγο κόπο δικές μας κλάσεις.

Το Django όμως ήταν η πρώτη μας επαφή με κάποιο Web Framework. Έχοντας προηγούμενη εμπειρία με διάφορα CMS (Joomla, Drupal, Wordpress) είχαμε δει σελίδες έτοιμες να δημιουργούνται με λίγα κλικ, και ένα πλήρες σύστημα χρηστών και δικαιωμάτων

να εμφανίζεται από το πουθενά. Όμως μας ήταν δύσκολο να αντιληφθούμε ότι από το μηδέν και μόνο με λίγες γραμμές κώδικα θα μπορούσαμε απλά καλώντας ορισμένες προκαθορισμένες κλάσεις να δημιουργήσουμε κάτι αντίστοιχο. Και το πιο εντυπωσιακό ήταν ότι όλα ήταν πλήρως παραμετροποιήσιμα, κατανοητά και κυρίως δημιουργημένα από μας τους ίδιους, γεγονός που μας επέτρεπε να γνωρίζουμε εις βάθος τη λειτουργία τους.

Το μεγαλύτερο προτέρημα της εφαρμογής μας όμως έγκειται στο γεγονός ότι γράφτηκε από φοιτητές για φοιτητές και έτσι θα παραμείνει, ο κώδικας θα μείνει ανοιχτός και ευελπιστούμε όλοι να βάλουν το χέρι τους ώστε να καλύψουν κάθε κενό που πιστεύουν ότι υπάρχει είτε στη συγκεκριμένη εφαρμογή, είτε γενικώς στο σύστημα που λειτουργεί το ΤΕΙ και να δημιουργήσουν κάτι αντίστοιχο που θα τους καλύπτει. Ακριβώς με αυτό το σκεπτικό ξεκινήσαμε και εμείς οι ίδιοι όταν σκεφτόμασταν το θέμα της εργασίας μας, και γι αυτό πιστεύουμε ότι υπήρξε ανταπόκριση από το πλήθος.

Φυσικά μένουν πολλά να γίνουν. Είναι εμφανείς οι ελλείψεις της εφαρμογής, μιας και δεν γινόταν να πραγματοποιηθούν όλα στο διάστημα που μας δόθηκε. Ένα χαρακτηριστικό παράδειγμα θα ήταν να η δημιουργία ενός πλήρους πελάτη για τη διαχείριση των e-mail του χρήστη, καθώς το Django διαθέτει ήδη κλάσεις για τη συγκεκριμένη διεργασία. Προς το παρών η εφαρμογή απλά κάνει parse τα emails του χρήστη, κάνοντας αδύνατη την αποστολή ή την προβολή των διαγραμμένων και των απεσταλμένων για παράδειγμα.

Επίσης θα μπορούσε να πραγματοποιηθεί ένα σύστημα ανακοινώσεων, με έναν εγγραφέα τύπου WYSIWYG (ότι βλέπεις είναι ότι παίρνεις), ώστε να παρακολουθούνται τα νέα της ιστοσελίδας από τους ενδιαφερόμενους. Η ύπαρξη LDAP server αυτόματα δίνει την δυνατότητα στην εφαρμογή να εξαπλωθεί απίστευτα, όπως για παράδειγμα το όνομα χρήστη και το συνθηματικό να δίνεται από το ΤΕΙ κατά την εισαγωγή του φοιτητή, και να χρησιμοποιείται στα εργαστήρια της σχολής, στο ασύρματο δίκτυο, στη βιβλιοθήκη, στην ιστοσελίδα της Γραμματείας, στο e-class και στο email του. Η υποδομή για να γίνει το παραπάνω υπάρχει, μιας και ο LDAP server του Cronos μπορεί εύκολα να ενοποιηθεί με τον υπάρχον επίσημο LDAP του ΤΕΙ, και στα πλαίσια μιας πτυχιακής τα υπόλοιπα συστήματα να ενσωματωθούν στον παραπάνω LDAP, μαζί με τα εργαστήρια, τα οποία θα λειτουργούν ως LDAP clients.

Ενώ από πλευράς backend η σελίδα λειτουργεί άψογα, από άποψη σχεδιασμού και

design πάσχει όμως πάρα πολύ, καθώς ο συγκεκριμένος τομέας δεν είναι κάτι που κατέχουμε πλήρως. Αυτό που καταφέραμε είναι απλά να δημιουργήσουμε ένα υποτυπώδες CSS το οποίο σε συνεργασία με τα κατάλληλα γραφικά χρησιμοποιήθηκε σωστά σε διάφορα σημεία και δημιουργήθηκε ένα περιβάλλον απλό και φιλικό προς το χρήστη. Ήδη έχουν δηλώσει άτομα ενδιαφέρον για το design, και λόγω του διαχωρισμού του Django των αρχείων python από τα HTML και CSS αρχεία, το project μπορεί να εξελιχτεί πάρα πολύ στον συγκεκριμένο τομέα. Επιπλέον, η ύπαρξη της τεχνολογίας jQuery χρησιμοποιείται πλέον ευρέως στα περισσότερα Django applications, δίνοντας ακόμη πιο δυναμικό χαρακτήρα στο αποτέλεσμα.

Τέλος, το πιο δύσκολο κομμάτι που θα μπορούσε να υλοποιηθεί στην εφαρμογή είναι η αντικατάσταση της σελίδας των ρυθμίσεων με το Admin Panel του Django, και η ενσωμάτωση σε αυτό όλων των λειτουργιών που ήδη υπάρχουν ή θα χρειαστούν στην πορεία. Η ανάπτυξη του Admin Panel του Django ακολουθεί άλλη πορεία από όλα τα υπόλοιπα μέρη του, καθώς υποστηρίζει για παράδειγμα πολλές φόρμες με ένα μόνο κουμπί υποβολής, και παρέχει πολλές κλάσεις για δημιουργία διαφόρων τύπων ρυθμίσεων. Το πρόβλημα όμως θα ήταν τα θέματα ασφάλειας, τα οποία εύκολα μπορούν να αντιμετωπιστούν, καθώς και η ενσωμάτωση του design του admin panel με την υπόλοιπη σελίδα. Και το δεύτερο πρόβλημα μπορεί να αντιμετωπιστεί, όχι και τόσο εύκολα βέβαια, μιας και τα αρχεία CSS και HTML είναι διαθέσιμα προς όλους.

Απώτερος σκοπός μας θα ήταν φυσικά να δούμε όλες τις υπηρεσίες του ΤΕΙ να τρέχουν κάτω από μία κοινή ομπρέλα, ένα συμπαγές γραφικό περιβάλλον, όπου θα προβάλλονται όλες οι ανακοινώσεις, θα βρίσκονται όλα τα επισυναπτόμενα αρχεία, σημειώσεις όλων των καθηγητών από όλα τα μαθήματα, θα γίνεται η επικοινωνία φοιτητών-καθηγητών για εργασίες, και άλλα πολλά.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία

- Magnus Lie Hetland “Beginning Python: From Novice to Professional, Second Edition”, εκδόσεις Apress 2008, ISBN13: 978-1-59059-982-2
- Mark Lutz “Learning Python, Third Edition”, εκδόσεις O'Reilly 2008, ISBN-13: 978-0-596-51398-6
- Noah Gift & Jeremy M. Jones “Python for Unix and Linux System Administration”, εκδόσεις O'Reilly 2008, ISBN: 978-0-596-51582-9
- The Definitive Guide to Django: Web Development Done Right, Second Edition, ISBN-10: 143021936X
- Pro Django (Expert's Voice in Web Development) ISBN-10: 1430210478
- Python Web Development with Django ISBN-10: 0132356139
- LDAP System Administration ISBN-10: 1565924916
- Mastering OpenLDAP: Configuring, Securing and Integrating Directory Services ISBN-10: 1847191029

Ιστοσελίδες:

- http://www.djangoproject.com/
- http://en.wikipedia.org/wiki/Django_(web_framework)
- http://www.django.gr/
- http://www.python.org/
- http://en.wikipedia.org/wiki/Python_(programming_language)
- http://www.openldap.org/

- http://en.wikipedia.org/wiki/OpenLDAP
- http://www.crummy.com/software/BeautifulSoup/documentation.html
- http://pycurl.sourceforge.net/
- http://curl.haxx.se/libcurl/python/
- http://www.python-ldap.org/
-
- http://code.google.com/p/django-ldap-groups/
- http://www.gentoo.org/
- http://www.gentoo.org/proj/en/hardened/
- http://en.wikipedia.org/wiki/Hardened_Gentoo
- http://www.gentoo.org/doc/en/ldap-howto.xml
- http://en.gentoo-wiki.com/wiki/Subversion
- http://subversion.apache.org/
- http://en.wikipedia.org/wiki/Apache_Subversion
- http://blog.tampakrap.gr/subversion-setup-and-gorg/