ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εφαρμογή Διαμοιρασμού Αρχείων μέσω Υπηρεσιών Cloud (Cloud Sharing Application)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

'Αλ-Σαμίστι Φάντι-Μαχμούντ

Επιβλέπων : Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2014

Η σελίδα είναι σκόπιμα λευκή.

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εφαρμογή Διαμοιρασμού Αρχείων μέσω Υπηρεσιών Cloud (Cloud Sharing Application)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

'Αλ-Σαμίστι Φάντι-Μαχμούντ

Επιβλέπων : Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 17η Φεβρουαρίου 2014

.....	.....	.....
Νεκτάριος Κοζύρης Καθηγητής Ε.Μ.Π.	Νικόλαος Παπασπύρου Αν.Καθηγητής Ε.Μ.Π.	Κωνσταντίνος Κοντογιάννης Αν.Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2014

.....

Άλ-Σαμίστι Φάντι-Μαχμούντ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Το Cloud Computing είναι το επόμενο βήμα στην εξέλιξη του διαδικτύου. Υπηρεσίες, όπως η υπολογιστική δύναμη, οι υπολογιστικές υποδομές και οι εφαρμογές, μπορούν να καταναλωθούν από τον τελικό χρήστη οπουδήποτε και όποτε τις χρειαστεί.

Η άνθηση του, που ξεκίνησε στην αρχή της χιλιετίας, τράβηξε την προσοχή πολλών εταιρειών, παλαιών και νέων, για την αξιοποίηση της νέας αυτής προσέγγισης στην επίλυση προβλημάτων και ικανοποίησης αναγκών.

Μια από τις πιο δημοφιλείς υπηρεσίες είναι η παροχή χώρου για την αποθήκευση αρχείων στις υποδομές του εκάστοτε παρόχου. Μια εταιρεία που δραστηριοποιείται στον τομέα αυτό εφοδιάζει τον χρήστη με κάποιο, εύκολο συνήθως, τρόπο να διαχειρίζεται τα αρχεία του, είτε μέσω κάποιου προγράμματος περιήγησης στο διαδίκτυο ή μέσω μιας εφαρμογής στον υπολογιστή. Μέσα από αυτά τα προγράμματα ο χρήστης μπορεί, με προσαρμόσιμά δικαιώματα, να μοιραστεί τα αρχεία του με άλλους. Πολλοί χρήστες χρησιμοποιούν περισσότερες από μια υπηρεσίες για να αποθηκεύσουν και να μοιραστούν τα αρχεία τους, γεγονός που οδηγεί στην εγκατάσταση πολλαπλών εφαρμογών στον υπολογιστή. Το παραπάνω φαινόμενο δημιουργεί την ανάγκη διαμοιρασμού αρχείων σε ένα εύρος υπηρεσιών χωρίς την εγκατάσταση της εφαρμογής που απαιτεί η κάθε υπηρεσία.

Στο πλαίσιο της διπλωματικής εργασίας θα παρουσιαστούν οι διαδικασίες της ανάλυσης των απαιτήσεων, της σχεδίασης του συστήματος και της υλοποίησης μιας εφαρμογής με στόχο να απαλλάξει τον τελικό χρήστη από το προαναφερθέν πρόβλημα. Το κύριο χαρακτηριστικό της είναι η παροχή μιας ενοποιημένης και παράλληλα απλής διαπροσωπείας μέσω της οποίας ο χρήστης μπορεί να ανεβάσει και να μοιραστεί αρχεία μέσω των υπηρεσιών Dropbox, Pithos+ και GoogleDrive καθώς και να διαχειριστεί το ιστορικό των αρχείων αυτών.

Τέλος, η εργασία έχει σκοπό να λειτουργήσει ως οδηγός για κάποιον που θα επιχειρήσει να δημιουργήσει μια ανάλογη εφαρμογή, ανεξαρτήτως πλατφόρμας ή γλώσσας προγραμματισμού. Αν κάποιο τμήμα δεν τον ικανοποιεί, βάσει της ανάλυσης και του τρόπου σκέψης που παρουσιάζονται, έχει τα εργαλεία για να σχεδιάσει ένα σύστημα βάσει των προσδοκιών του.

Λέξεις Κλειδιά:

αρχιτεκτονική λογισμικού, τεχνολογία λογισμικού, προγραμματισμός, αλληλεπίδραση ανθρώπου μηχανής, διεπαφή χρήστη, Python, MVC

Η σελίδα είναι σκόπιμα λευκή.

Abstract

Cloud Computing is the next step in the evolution of the internet. Services, like computation power, computing infrastructure and applications, can be consumed by the end user wherever and whenever they are needed.

The bloom of Cloud Computing, which started at the beginning of the millennium, drew the attention of many companies, both old and new, to utilize this new approach in problem solving and need satisfaction.

One of the most popular services is the space provision for file storage at the infrastructure of each provider. A company that is active in this field provides the user with a, usually easy, way to manage his files through a web interface or an application on the computer. Using these programs the user is able to share, with customizable permissions, his files with others. A large portion of users interact frequently with more than one services in order to store and share their files, a fact that leads to multiple application installations in their computers. As a result of this phenomenon a need emerges, that of sharing files to an array of services without installing the application that is required by each service.

In the scope of the thesis we will present the procedures of requirements analysis, system design and implementation of an application in order to relieve the end user from the aforementioned problem. The main feature of the application is to provide a streamlined and easy to use interface which the user can use to upload and share files through Dropbox, Pithos+ and GoogleDrive as well as manage the history of these files.

Finally, this work aims to act as a guide for someone who will attempt to create a similar application, regardless of the platform or the programming language. If any part does not satisfy him, based on the analysis and the way of thought that are presented, he is given the tools to design a system based on his vision.

Keywords:

Software architecture, software engineering, programming, human computer interaction, user interface, Python, MVC

Η σελίδα είναι σκόπιμα λευκή

Πίνακας περιεχομένων

Πίνακας Εικόνων

1

Εισαγωγή

1.1 Cloud Computing

Το “Cloud Computing” ή αλλιώς “the cloud” είναι ένας γενικός όρος που χρησιμοποιείται για να περιγράψει υπηρεσίες που διανέμονται μέσω του διαδικτύου και φιλοξενούνται στις υποδομές των παρόχων. Το κύριο χαρακτηριστικό του cloud είναι ο βέλτιστος διαμοιρασμός υπολογιστικών πόρων δυναμικά σε επίπεδο φόρτου ανά χρήστη αλλά και συνολικά ανά υπηρεσία.

Μια υπηρεσία που παρέχεται μέσω του cloud διακρίνεται για τρία βασικά χαρακτηριστικά:

- Παρέχεται κατ’ απαίτηση, είτε επί πληρωμή, συνήθως ανά λεπτό ή ωριαία, είτε δωρεάν,
- Είναι ευέλικτη και κλιμακώσιμη, όπως προαναφέρθηκε, με την έννοια ότι ένας χρήστης μπορεί να την χρησιμοποιήσει όποτε και όσο θέλει,
- Συντηρείται αποκλειστικά από τον πάροχο, που σημαίνει ότι ο καταναλωτής χρειάζεται μόνο έναν υπολογιστή και πρόσβαση στο διαδίκτυο για την χρήση της.

Με την συνεχώς αυξανόμενη δημοτικότητα των cloud υπηρεσιών, ο διαμοιρασμός αρχείων για πολλούς έχει γίνει αναπόσπαστο κομμάτι της ζωής τους. Η διαδικασία αποστολής αρχείων εξαρτάται από τον τρόπο που έχει επιλέξει η εκάστοτε υπηρεσία να γίνεται και συνήθως η κάθε μια απαιτεί την εγκατάσταση δικού της λογισμικού που, αν και πολύ-χρηστικό, χρειάζεται χρόνο εξοικείωσης και έχει ως αποτέλεσμα ο υπολογιστής να εκτελεί δυο και τρία προγράμματα ταυτόχρονα για τον ίδιο σκοπό.

Υπάρχουν περιπτώσεις όμως που ο χρήστης θέλει μόνο να μοιράζεται αρχεία μέσω των υπηρεσιών αυτών όσο πιο εύκολα γίνεται χωρίς να τα κρατάει αποθηκευμένα στον υπολογιστή του ή να τα συγχρονίζει με άλλους υπολογιστές, όπως κάνουν οι εφαρμογές των Dropbox και GoogleDrive ξοδεύοντας έτσι υπολογιστικούς πόρους.

1.1.1 Ιστορική Αναδρομή

Αξίζει να κάνουμε μια επισκόπηση για το πώς εξελίχθηκε η ιδέα του cloud computing. Θα ξεκινήσουμε από την σύλληψη της έννοιας πριν ακόμα της δοθεί ονομασία μέχρι και σήμερα που έχει εισχωρήσει σε όλους τους τομείς της τεχνολογίας και των επιστημών.

1950-1960

Τα θεμέλια των διαμοιραζόμενων υπολογιστικών πόρων ετέθησαν την δεκαετία του ’50, όταν μεγάλα συστήματα υπολογιστών έγιναν διαθέσιμα σε πανεπιστήμια, σχολεία και οργανισμούς. Οι υποδομές για αυτά τα συστήματα είχαν έκταση ολόκληρων δωματίων και η πρόσβαση σε αυτά γινόταν μέσω “στατικών” τερματικών καθώς δεν είχαν δυνατότητες επεξεργασίας δεδομένων παρά μόνο έδιναν τα μέσα για την επικοινωνία.

Για την πιο αποτελεσματική χρήση των κοστοβόρων υποδομών, αναπτύχθηκε μια πρακτική που επέτρεπε σε πολλούς χρήστες να μοιράζονται από πολλαπλά τερματικά την υπολογιστική ισχύ και τον χώρο. Ως αποτέλεσμα οι περίοδοι αδράνειας των συστημάτων εξαλείφτηκαν και η απόδοση της επένδυσης αυξήθηκε. Αυτή η πρακτική ονομάστηκε χρονο-μοιρασμός.

1960-1990

Λόγω των μεγάλων επενδύσεων που γίνανε για πολύ ισχυρά υπολογιστικά συστήματα, πολλές εταιρείες και οργανισμοί, όπως η θυγατρική της IBM SBC(Service Bureau Corporation) που ιδρύθηκε το 1957, η Tymshare(1966), η National CSS(1967), η Dial Data(1968) και Bolt, Beranek, and Newman(BBN Technologies), διαφημίζανε τον χρονομοιρασμό ως μια κερδοφόρα επένδυση.

Την δεκαετία του ’70 η IBM κυκλοφόρησε ένα λειτουργικό με την ονομασία VM που επέτρεπε στους διαχειριστές συστημάτων να τρέχουν πολλαπλά εικονικά μηχανήματα(VMs, Virtual Machines) σε έναν φυσικό κόμβο. Ήταν η φυσική εξέλιξη των τερματικών της δεκαετίας του ’50 μιας και πλέον μπορούσαν να ζήσουν ξεχωριστά υπολογιστικά περιβάλλοντα σε ένα φυσικό περιβάλλον.

Οι κύριες δυνατότητες των εικονικών μηχανημάτων ήταν να τρέχουν δικό τους λειτουργικό σύστημα, να έχουν δικιά τους μνήμη, CPU, σκληρό δίσκο και συσκευές ανάγνωσης για CD-ROM. Η “Εικονοποίηση” ήταν ο καταλύτης για επαναστατικές εξελίξεις στους τομείς της πληροφορικής και των επικοινωνιών.

1990-2000

Οι εταιρείες τηλεπικοινωνιών μέχρι εκείνη την περίοδο προσέφεραν δίκτυα δεδομένων από σημείο σε σημείο. Ξεκίνησαν να προσφέρουν εικονικά ιδιωτικά δίκτυα με συγκρίσιμη ποιότητα αλλά με χαμηλότερο κόστος. Ρυθμίζοντας την κυκλοφορία μπορούσαν να εξισορροπούν τον φόρτο των εξυπηρετητών με αποτέλεσμα το δίκτυο να αξιοποιείται αποδοτικότερα.

Καθώς οι υπολογιστές σιγά-σιγά επικρατούσαν, οι επιστήμονες και οι τεχνολόγοι έψαχναν τρόπους για να καταστήσουν διαθέσιμους σε μεγαλύτερο κοινό την υπολογιστική ισχύ μεγάλης κλίμακας μέσω χρονο-μοιρασμού. Για το λόγο αυτό πειραματίζονταν με αλγορίθμους για την βέλτιστη χρήση της υποδομής, της πλατφόρμας και των εφαρμογών.

2000-Σήμερα

Η συγκεκριμένη χρονική περίοδος είναι η πιο επαναστατική όσον αφορά τις εξελίξεις στον τομέα του cloud computing.

Το 2002 η Amazon εισήγαγε τις διαδικτυακές υπηρεσίες Amazon(Amazon Web Services, AWS) που με τις σημαντικές προσθήκες των Amazon Elastic Compute(EC2) και Simple Storage Service(S3) το 2006 αποτελούν την διαδικτυακή πλατφόρμα υπηρεσιών της Amazon.

Το 2004 ιδρύθηκε η εταιρεία Facebook που κατάφερε και μετέτρεψε την το cloud σε κάτι προσωπικό και δωρεάν.

Το 2007 η Salesforce.com ξεκίνησε το force.com που είναι μια πλατφόρμα ως υπηρεσία(Platform-as-a-service, PaaS), με στόχο να δώσει στους πελάτες της τα απαραίτητα εργαλεία για να κατασκευάσουν, να αποθηκεύσουν και να εκτελέσουν όλες τις εφαρμογές και τις ιστοσελίδες που χρειαζόντουσαν.

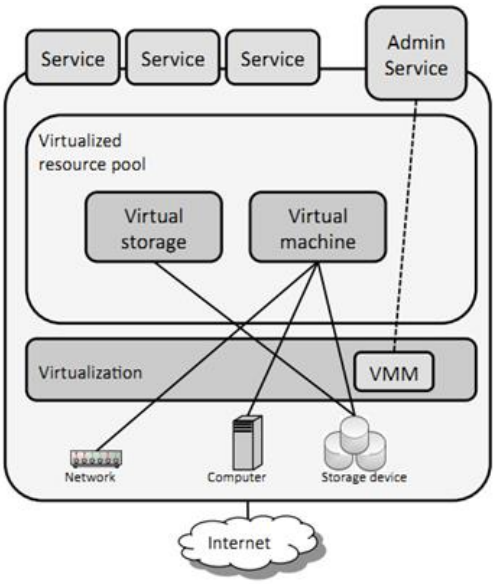
Το 2009 η Google διέθεσε την υπηρεσία Google Apps που επιτρέπει την δημιουργία και τον διαμοιρασμό εγγράφων στο cloud.

Βλέπουμε πως το cloud computing, αν και σχετικά πρόσφατη έννοια, τυγχάνει ευρείας αποδοχής και η απαρίθμηση που έγινε δεν είναι παρά ένα μικρό δείγμα τεχνολογιών.

1.1.2 Κατηγοριοποίηση

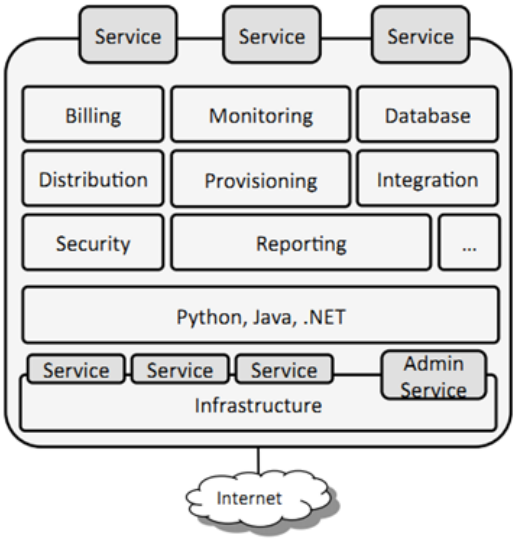
Το εύρος των cloud υπηρεσιών που προσφέρονται είναι πολύ μεγάλο για αυτό οι πάροχοι ακολουθούν κάποια μοντέλα υπηρεσιών. Τα βασικότερα εξ' αυτών είναι:

- Υποδομή ως υπηρεσία(Infrastructure-as-a-Service, IaaS): Είναι από τα βασικότερα μοντέλα και παρέχει στον χρήστη φυσικά ή εικονικά μηχανήματα, τα οποία συντονίζονται από κάποιο επιβλέπον λογισμικό(Xen, KVM, ESX/ESXi). Μαζί με αυτά τα μηχανήματα προσφέρονται προ-εγκατεστημένα πακέτα λογισμικού, firewalls, εικονικά τοπικά δίκτυα(VLANs) και πολλά ακόμη.



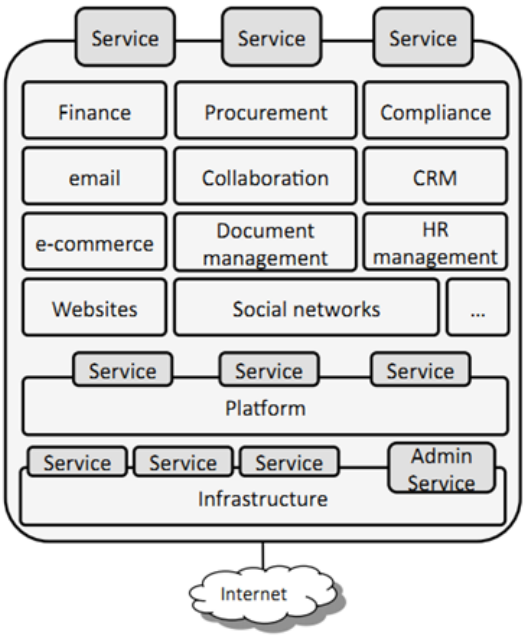
Εικόνα 1: Στοίβα τεχνολογιών του IaaS

- Πλατφόρμα ως υπηρεσία(Platform-as-a-Service, PaaS): Το μοντέλο αυτό περιγράφει μια πλατφόρμα που περιλαμβάνει λειτουργικό σύστημα, περιβάλλον για επιλεγμένες γλώσσες προγραμματισμού, βάσεις δεδομένων και δικτυακούς εξυπηρετητές. Οι προγραμματιστές αναπτύσσουν και εκτελούν εφαρμογές στην πλατφόρμα χωρίς να χρειάζεται να ανησυχούν για τα χαμηλότερα επίπεδα.



Εικόνα 2: Στοίβα τεχνολογιών του PaaS

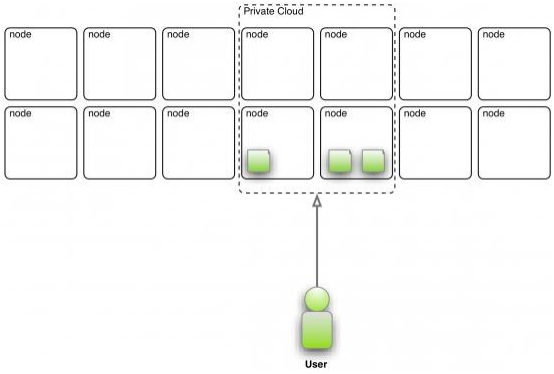
- Λογισμικό ως υπηρεσία(Software-as-a-Service, SaaS): Οι πάροχοι του SaaS διαθέτουν στον τελικό χρήστη μια ολοκληρωμένη εφαρμογή ή σουίτα εφαρμογών έτοιμων προς κατανάλωση. Την πλατφόρμα και την αναγκαία υποδομή την διαχειρίζεται ο πάροχος και η πρόσβαση στην υπηρεσία γίνεται μέσω διαδικτυακής πύλης, γεγονός που καθιστά περιττή την εγκατάσταση και την συντήρηση εφαρμογών στον υπολογιστή.



Εικόνα 3: Στοίβα τεχνολογιών του SaaS

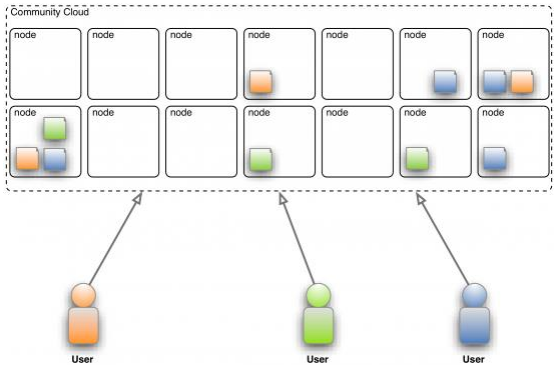
Οι υπηρεσίες και οι υποδομές που παρέχονται στοχεύουν συγκεκριμένο κοινό και οι βασικότερες ομάδες είναι:

- Το ιδιωτικό νέφος(Private Cloud): Δίνεται αποκλειστική πρόσβαση στα στελέχη ενός οργανισμού ή εταιρείας.



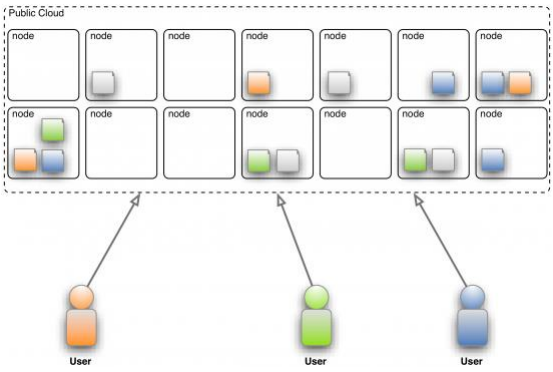
Εικόνα 4: Private Cloud

- Νέφος κοινότητας(Community Cloud): Τα μέλη οργανισμών που ανήκουν σε μια κοινότητα με κοινά ενδιαφέροντα και ανησυχίες, για παράδειγμα πρότυπα ασφάλειας ή ζητήματα συμμόρφωσης, έχουν πρόσβαση σε αυτού του είδους το νέφος.



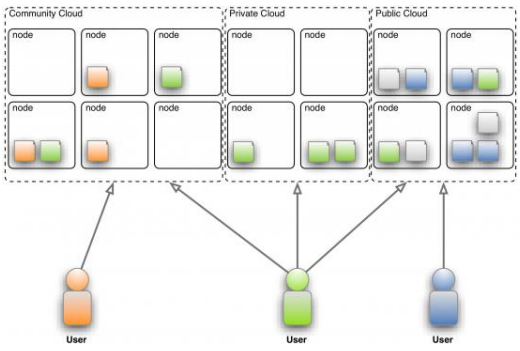
Εικόνα 5: Community Cloud

- Το δημόσιο νέφος(Public Cloud): Παρέχεται για χρήση στο ευρύ κοινό.



Εικόνα 6:Public Cloud

- Το υβριδικό νέφος(Hybrid Cloud): Οι υποδομές και οι υπηρεσίες διανέμονται βάσει συνδυασμού δυο ή περισσότερων ειδών.



Εικόνα 7: Hybrid Cloud

1.2 Αντικείμενο διπλωματικής

Η διπλωματική εργασία έχει ως στόχο την ανάλυση, την σχεδίαση και την ανάπτυξη εφαρμογής που διευκολύνει την αποστολή και τον διαμοιρασμό αρχείων σε πολλαπλές υπηρεσίες ταυτόχρονα με ενοποιημένο και απλό τρόπο.

Τα χαρακτηριστικά του συστήματος προς υλοποίηση θα είναι τα ακόλουθα:

- **Μη-επεμβατική**: Δεν θα διακόπτει την κανονική ροή των ενεργειών του χρήστη εκτός αν επιθυμεί να χρησιμοποιήσει την εφαρμογή,
- **Απλό και κατανοητό γραφικό περιβάλλον**: Η χρήση της εφαρμογής δεν θα απαιτεί εγχειρίδιο και ο χρήστης δεν θα χρειάζεται να κάνει πολλές ενέργειες για να πετύχει αυτό που θέλει,
- **Άμεσα αποκρίσιμα παράθυρα**: Η αποστολή αρχείων είναι χρονοβόρα διαδικασία και πρέπει να εκτελείται στο προσκήνιο έτσι ώστε η εμπειρία χρήσης να είναι όσο πιο λειτουργική και άμεση γίνεται,
- **Ανεξάρτητο του λειτουργικού**: Η εφαρμογή, λόγω της γλώσσας προγραμματισμού στην οποία θα υλοποιηθεί, θα έχει την δυνατότητα να εκτελεστεί σε πληθώρα λειτουργικών συστημάτων,
- **Κώδικας σχεδιασμένος για την ανεξαρτησία των υπομονάδων:** Ο κώδικας της εφαρμογής θα δομηθεί καταλλήλως με τον διαχωρισμό των μονάδων κατά νου έτσι ώστε κάθε προσθήκη λειτουργιών να απαιτεί την λιγότερη προγραμματιστική προσπάθεια,
- **Χρήση δημοφιλών υπηρεσιών:** Οι υπηρεσίες που αρχικά θα υποστηρίζονται θα επιλεγούν βάσει της δημοτικότητάς τους για να ικανοποιεί το ευρύ κοινό,
- **Χαμηλή κατανάλωση υπολογιστικών πόρων,**

1.3 Οργάνωση του τόμου

Η δομή της εργασίας έχει ως εξής:

- Στο **Κεφάλαιο 2,** αναλύονται οι απαιτήσεις του συστήματος,
- Στο **Κεφάλαιο 3**, παρουσιάζεται το απαραίτητο τεχνικό υπόβαθρο για την δημιουργία της εφαρμογής,
- Στο **Κεφάλαιο 4,** καταγράφονται οι λειτουργίες των παραθύρων που απαρτίζουν το γραφικό περιβάλλον και αναλύονται οι λειτουργικές και ποιοτικές απαιτήσεις του συστήματος,
- Στο **Κεφάλαιο 5**, παρουσιάζεται σε βάθος η αρχιτεκτονική του συστήματος και οι βασικότερες κλάσεις, περιγράφονται οι πληροφορίες που αποθηκεύονται στον τοπικό δίσκο και δίνεται η ευρύτερη εικόνα της υλοποίησης συνδυάζοντας όλες τις υπομονάδες,
- Στο **Κεφάλαιο 6**, αναλύονται θέματα υλοποίησης στο επίπεδο του κώδικα, παρέχεται το εγχειρίδιο χρήσης και γίνονται αναφορές στην πλατφόρμα ανάπτυξης, στα προγραμματιστικά εργαλεία και στις απαιτήσεις της εφαρμογής,

- Στο **Κεφάλαιο 7,** γίνεται μια σύνοψη στο έργο που έγινε, καταγραφή των συμπερασμάτων και των ιδεών για την εξέλιξη του συστήματος,
- Στο **Κεφάλαιο 8**, παρατίθεται η βιβλιογραφία.

Η σελίδα είναι σκόπιμα λευκή

2

Ανάλυση Απαιτήσεων Συστήματος

Στην ενότητα αυτή, κάνουμε μια ανάλυση των απαιτήσεων που θέλουμε το υπό ανάπτυξη σύστημα να έχει. Δίνεται μια μικρή περιγραφή για τις μονάδες που απαρτίζουν το σύστημα σε αφηρημένο επίπεδο, δηλαδή πως πρέπει να λειτουργούν οι υπομονάδες. Για κάθε υπομονάδα καταγράφουμε τι δυνατότητες πρέπει να προσδίδει στο σύστημα έτσι ώστε να προκύψει κάτι ολοκληρωμένο και ανεκτικό στα σφάλματα.

2.1 Απαιτήσεις Συστήματος

Η εφαρμογή που κατασκευάζουμε πρέπει να έχει απλή διαπροσωπεία με τον χρήστη. Όμως πρέπει να έχει την δυνατότητα να διαχειριστεί οποιοδήποτε κομμάτι της εφαρμογής μπορεί να του χρειαστεί, συγκεκριμένα να έχει την εποπτεία και τον πλήρη έλεγχο των αρχείων που ανεβαίνουν ή έχουν διακοπεί, να βλέπει ποια αρχεία έχει ανεβάσει παλιότερα, να δύναται να αλλάξει τις ρυθμίσεις και να προσθαφαιρεί λογαριασμούς υπηρεσιών. Παρ' όλα αυτά η κυριότερη απαίτηση του συστήματος είναι όσο ανεβαίνουν αρχεία η εφαρμογή να είναι πλήρως απόκρισιμη.

Απλή Διαπροσωπεία

Για την ικανοποίηση της συγκεκριμένης απαίτησης πρέπει να σχεδιάσουμε ένα παράθυρο που δεν είναι παρεμβατικό, με την έννοια ότι δεν θα αλλάξει τις συνήθειες που είχε μέχρι τώρα για να χρησιμοποιεί σε μόνιμη βάση την εφαρμογή. Όταν όμως θέλει να την χρησιμοποιήσει, να είναι εύκολα προσβάσιμη από σχεδόν οποιαδήποτε κανονική κατάσταση λειτουργίας του υπολογιστή.

Παραμετροποιήσιμη

Η εφαρμογή πρέπει να προσφέρει επιλογές στον χρήστη για την γενική συμπεριφορά της καθώς και για κάθε υπηρεσία που χρησιμοποιείται, για παράδειγμα σε ποιο φάκελο θα αποθηκεύονται τα αρχεία που ανεβαίνουν ή προσθαφαίρεση υπηρεσίας.

Ανεκτικότητα και ανάνηψη από λάθη

Κάθε περίπτωση σφάλματος πρέπει να αντιμετωπίζεται και να ενημερώνει τον χρήστη ότι συνέβη και να του προτείνει τρόπο επίλυσης. Ακολουθούν τα τυπικά λάθη που μπορεί να συμβούν και προτείνονται τρόποι επίλυσης για το κάθε ένα:

- Διακοπή δικτύου: Επανεκκίνηση του αρχείου,
- Μη ύπαρξη του αρχείου: Παρακίνηση του χρήστη να ελέγξει ότι υπάρχει το αρχείο και να δοκιμάσει την επανεκκίνησή του,
- Λανθασμένα/Ληγμένα κλειδιά: Ο χρήστης να πάρει νέα κλειδιά για τν υπηρεσία από την διαδικασία εξουσιοδότησης και να δοκιμάσει την επανεκκίνηση του αρχείου,
- Εξάντληση διαθέσιμου χώρου: Ο χρήστης πρέπει να διαγράψει κάποιο αρχείο από την υπηρεσία και να ξαναδοκιμάσει,
- Κατάρρευση της εφαρμογής: Αποθήκευση της τρέχουσας κατάστασης αν είναι δυνατόν έτσι ώστε την επόμενη φορά που θα εκτελεστεί η εφαρμογή να μην υπάρχουν μεγάλες απώλειες στην πρόοδο των αρχείων,
- Μη διαθέσιμη υπηρεσία: Ο χρήστης πρέπει να κάνει προσπάθεια επανεκκίνησης του αρχείου.

Εποπτεία και έλεγχος επί της κατάστασης των αρχείων

Για να καλύψουμε την απαίτηση αυτή κρίνεται απαραίτητη η δημιουργία ενός παραθύρου που απεικονίζει την κατάσταση όλων των αρχείων που είναι ενεργά, δηλαδή σε κατάσταση ανεβάσματος, σταματημένα ή σε κατάσταση λάθους. Όσον αφορά τον έλεγχο το παράθυρο πρέπει να περιλαμβάνει ευδιάκριτα κουμπιά με τα οποία μπορεί να προσθέσει, να επανεκκινήσει, να σταματήσει ή να διαγράψει ένα αρχείο.

Ενημέρωση κατά την ολοκλήρωση ανεβάσματος αρχείου

Για να το επιτευχθεί αυτό πρέπει να δημιουργηθεί ένα παράθυρο που θα εμφανίζεται αυτόματα κατά την ολοκλήρωση ανεβάσματος ενός αρχείου που θα απεικονίζει πληροφορίες όπως:

- Το τελικό όνομα του αρχείου στην υπηρεσία: Συνήθως το όνομα θα είναι το ίδιο με αυτό του αρχείου που ανέβασε αρχικά ο χρήστης αλλά υπάρχει η πιθανότητα να υπάρξει κάποια σύγκρουση ονομάτων που λύνεται βάσει της πολιτικής της κάθε υπηρεσίας.
- Την ημερομηνία και την ώρα ολοκλήρωσης,
- Την υπηρεσία που ανέβηκε το αρχείο,
- Τον σύνδεσμο για τον διαμοιρασμό του αρχείου: Με το πάτημα ενός κουμπιού ο χρήστης θα αντιγράφει τον σύνδεσμο στον πρόχειρο χώρο του λειτουργικού του συστήματος(clipboard).

Λογικό είναι πως στο παράθυρο δεν θα υπάρχει χώρος για όλα τα αρχεία που έχουν ανέβει με την εφαρμογή, για το λόγο αυτό θα προβάλλονται μόνο τα πιο πρόσφατα.

Αποθήκευση παραμέτρων στον δίσκο

Οι παράμετροι πάσης φύσεως, γενικού σκοπού ή ανά υπηρεσία, πρέπει να αποθηκεύονται στον σκληρό δίσκο για μελλοντική χρήση. Κάποια από αυτά τα δεδομένα είναι ευαίσθητα για αυτά οφείλουν να αναπτυχθούν μηχανισμοί για την προστασία τους. Οι πληροφορίες που πρέπει να αποθηκευτούν είναι οι ακόλουθες:

- Επικοινωνία με τις υπηρεσίες: Περιέχει τα κλειδιά που είναι απαραίτητα για την επικοινωνία της εφαρμογής με την κάθε υπηρεσία καθώς και τον φάκελο που θέλει να αποθηκεύονται τα αρχεία.
- Ιστορικό: Είναι αποθηκευμένα όλα τα αρχεία που έχουν ανέβει μέσω της εφαρμογής και έχει τις ακόλουθες πληροφορίες: ημερομηνία ολοκλήρωσης, τελικό όνομα στην υπηρεσία, τοπικό μονοπάτι του αρχείου και τον σύνδεσμο διαμοιρασμού,
- Εφαρμογή: Εδώ γράφονται οι ρυθμίσεις του χρήστη, οι υπηρεσίες που χρησιμοποιούνται και συντεταγμένες οθόνης για να τοποθετούνται τα παράθυρα στην θέση όπου επιθυμεί ο χρήστης,
- Αρχεία των οποίων το ανέβασμα έχει διακοπεί: Για κάθε αρχείο που έχει διακοπεί αποθηκεύεται η αναγκαία πληροφορία για να επανεκκινήσει, όπως το τοπικό μονοπάτι του αρχείου, πόσα bytes εστάλησαν και η κατάστασή του, δηλαδή ενεργή, σταματημένη ή εσφαλμένη.

Επανεκκινήσιμα αρχεία

Αν για οποιοδήποτε λόγο το ανέβασμα ενός αρχείου διακοπεί πρέπει να υπάρχει μηχανισμός επανεκκίνησης. Οι λόγοι που μπορεί να διακοπεί ένα ανέβασμα είναι από ενέργεια του χρήστη, διακοπή του δικτύου, κανονικός τερματισμός ή κατάρρευση της εφαρμογής, λανθασμένα κλειδιά για επικοινωνία.

Χαμηλή κατανάλωση πόρων

Ο στόχος της εφαρμογής είναι να είναι ενεργή για μεγάλα χρονικά διαστήματα για αυτό πρέπει να γίνεται καλή διαχείριση της μνήμης. Καλές πρακτικές είναι:

- Όσο ανεβαίνει ένα αρχείο δεν πρέπει να φορτώνεται εξ' ολοκλήρου στην μνήμη αλλά σε μικρά κομμάτια,
- Όταν ολοκληρωθεί πρέπει να δίνεται ιδιαίτερη προσοχή στα απομεινάρια του αρχείου, δηλαδή σε μεταβλητές που έχουν κομμάτια του.

Αποκρίσιμη

Η διαδικασία ανεβάσματος αρχείου είναι χρονοβόρα άρα πρέπει να προνοήσουμε και να ενσωματώσουμε μηχανισμό που θα δημιουργεί νήμα(thread) για κάθε ενεργό αρχείο κατά την εκκίνησή του. Το αποτέλεσμα του παραπάνω μηχανισμού είναι πως το γραφικό περιβάλλον δεν θα "κολλάει" όσα αρχεία και να ανεβαίνουν.

Η σελίδα είναι σκόπιμα λευκή.

3

Τεχνικό υπόβαθρο

Έχοντας καταγράψει τα χαρακτηριστικά του συστήματός μας, πρέπει να εξετάσουμε μακροσκοπικά τα εργαλεία, δηλαδή τις τεχνολογίες και τις έννοιες, που θα μας επιτρέψουν να υλοποιήσουμε ένα σταθερό και πλήρες σύστημα.

Περιληπτικά θα παρουσιαστούν τα παρακάτω:

- Γλώσσα Προγραμματισμού Python,
- Νήματα,
- Ουρές,
- Μικρή εισαγωγή στον αντικειμενοστραφή προγραμματισμό(Object Oriented Programming, OOP),
- Χρήση των Μηχανών Πεπερασμένων Καταστάσεων(Finite State Automata, FSM),
- Παρουσίαση του πρωτοκόλλου εξουσιοδότησης OAuth 2.0,
- Τι είναι ένα API και πως θα τα χρησιμοποιήσουμε,
- Η βιβλιοθήκη για την σχεδίαση του γραφικού περιβάλλοντος, PyQt4.

3.1 Γλώσσα Προγραμματισμού Python

Η Python είναι μια γλώσσα σεναρίων(scripting), γενικού σκοπού, υψηλού επιπέδου και δημιουργήθηκε το 1991 από τον Guido Van Rossum. Σχεδιάστηκε με στόχο να είναι ευανάγνωστη , να βοηθάει τον προγραμματιστή να είναι παραγωγικός και να χρειάζονται λιγότερες γραμμές κώδικα σε σχέση με γλώσσες προγραμματισμού όπως η C ή η Java.

Τα χαρακτηριστικά της είναι τα εξής:

- Έχει δυναμικό σύστημα τύπων : Είναι η διαδικασία κατά την οποία η ασφάλεια των τύπων ελέγχεται κατά την εκτέλεση του κώδικα,
- Υποστηρίζει πολλαπλά είδη προγραμματισμού: Ένα πρόγραμμα Python μπορεί να περιέχει στοιχεία αντικειμενοστραφούς, προστακτικού, συναρτησιακού και διαδικαστικού προγραμματισμού,
- Αυτόματη διαχείριση μνήμης: Περιέχει συλλέκτη σκουπιδιών(garbage collector) που ελευθερώνει και δεσμεύει την μνήμη έτσι ώστε να μην χρειάζεται να το κάνει ο προγραμματιστής,
- Μεταγλωττίζεται σε ενδιάμεση γλώσσα: Το αποτέλεσμα της μεταγλώττισης ενός προγράμματος Python δεν μπορεί να εκτελεστεί άμεσα από τον υπολογιστή. Ο παραγόμενος κώδικας ονομάζεται bytecode, όπως και στην Java, και για να τρέξει χρειάζεται ένας διερμηνέας ή αλλιώς ένα εικονικό μηχάνημα(Virtual machine).
- Είναι επεκτάσιμη: Σχεδιάστηκε με τέτοιο τρόπο που κάποιος μπορεί να δημιουργήσει δικιά του έκδοση της γλώσσας αλλάζοντας στοιχεία της αρχικής. Οι κυριότερες εκδόσεις της Python είναι:
    - CPython: Η πρώτη έκδοση που δημιουργήθηκε. Είναι γραμμένη στη γλώσσα C και η βιβλιοθήκη που συμπεριλαμβάνει καλύπτει μεγάλο εύρος αναγκών. Ο μηχανισμός που πρέπει να αναφερθεί είναι το καθολικό κλείδωμα του διερμηνευτή(Global Interpreter Lock, GIL) που επιτρέπει μόνο ένα νήμα να εκτελείται σε οποιαδήποτε στιγμή καθώς ο κώδικας της εν λόγω υλοποίησης δεν περιέχει μηχανισμούς προστασίας από την παράλληλη εκτέλεση κώδικα,
    - PyPy: Πιο γρήγορη έκδοση της CPython,
    - IronPython: Η υλοποίηση αυτή γράφτηκε στην γλώσσα C# εξ' ολοκλήρου με στόχο να τρέχει στο περιβάλλον .NET της Microsoft,
    - Jython: Μια υλοποίηση της Python στην γλώσσα Java που παράγει ενδιάμεσο κώδικα για το εικονικό μηχάνημα της Java,
    - Stackless Python: Επεκτείνει την CPython προσθέτοντας μικρονήματα, δηλαδή μηχανισμός για την ταυτόχρονη εκτέλεση κώδικα σε μικροεπεξεργαστές.
- Τρέχει ανεξαρτήτως του λειτουργικού: Λόγω της χρήσης του εικονικού μηχανήματος ο κώδικας, σχεδόν πάντα, τρέχει το ίδιο σε όλα τα λειτουργικά συστήματα.

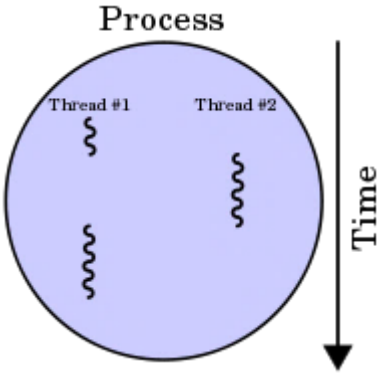
Είναι πολύ δημοφιλής γλώσσα γιατί είναι ιδανική για εκπαιδευτικούς σκοπούς αλλά και πολύ πλούσια για να χρησιμοποιείται σε επαγγελματικό και επιστημονικό επίπεδο. Ονομαστικά κάποιες εταιρείες και οργανισμοί που γράφουν τα έργα τους σε Python είναι η Google, η NASA, το Dropbox, ο ~okeanos, η Blender 3D, η Nokia, η Yahoo!, το CERN και πολλές άλλες.

3.1.1 Νήματα

Τα πολυεπίπεδα και πολύπλοκα συστήματα χρειάζονται να εκτελούν πολλές ενέργειες ταυτόχρονα. Η λύση σε αυτό το πρόβλημα είναι η χρήση νημάτων. Όμως η προσθήκη νημάτων υπαγορεύει ότι πρέπει να δημιουργηθούν υποσυστήματα για την διαχείρισή τους.

Νήμα είναι μια ακολουθία εντολών προς τον επεξεργαστή. Στην σύγχρονη εποχή που ζούμε, σχεδόν όλοι οι υπολογιστές περιέχουν επεξεργαστές με πολλούς πυρήνες. Κάθε πυρήνας μπορεί να εκτελεί ένα νήμα τη φορά αλλά ο χρονοπρογραμματιστής του λειτουργικού συστήματος είναι υπεύθυνος για την δίκαιη εναλλαγή των νημάτων που περιμένουν στην ουρά.

Η διεργασία χρειάζεται το λιγότερο ένα νήμα για να τρέξει, δεν υπάρχει όμως μέγιστο. Τα νήματα μιας πολυνηματικής εφαρμογής μοιράζονται την μνήμη που έχει δεσμεύσει, τον κώδικα της και το περιβάλλον εκτέλεσής της, δηλαδή τις τιμές που έχουν οι μεταβλητές σε οποιαδήποτε στιγμή.



Εικόνα 8: Διεργασία και νήματα

Στην εικόνα που παρατίθεται, απεικονίζεται μια διεργασία με δυο νήματα που εκτελείται σε έναν επεξεργαστικό πυρήνα. Παρατηρούμε πως όσο τρέχει ένα νήμα το εναπομείναν είναι σε κατάσταση παύσης.

Για την εργασία αυτή θα χρησιμοποιηθεί η ενσωματωμένη βιβλιοθήκη της Python, η *threading*, γιατί είναι πλήρης για τις ανάγκες μας και πολυχρησιμοποιημένη για να θεωρήσουμε πως είναι σταθερή. Η υπό εξέταση βιβλιοθήκη είναι μια υψηλού επιπέδου διαπροσωπεία για την μονάδα *thread*. Ας εξετάσουμε εποπτικά την κύρια κλάση που θα μας απασχολήσει.

threading.Thread

Αντιπροσωπεύει μια δραστηριότητα που τρέχει σε ξεχωριστό νήμα. Για να εκμεταλλευτούμε αυτή τη δυνατότητα θα επεκτείνουμε την κλάση προσθέτοντας τον δικό μας κώδικα.

Οι συναρτήσεις που αξίζουν αναφοράς είναι οι ακόλουθες:

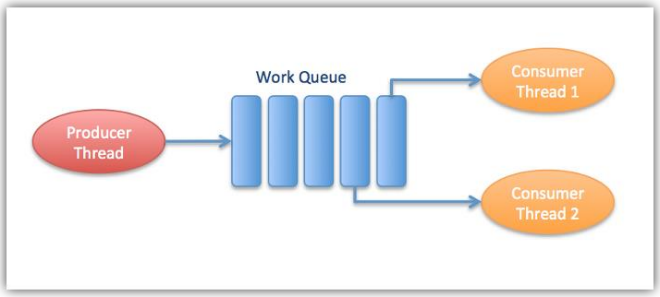
- Κατασκευαστής: Στο τμήμα αυτό αρχικοποιούμε όσες μεταβλητές είναι αναγκαίες για την εκτέλεση του κώδικα που θέλουμε να τρέξει ταυτόχρονα,
- daemon: Αν η εφαρμογή προσπαθήσει να τερματίσει θα περιμένει όσα νήματα δεν είναι daemon, η εν λόγω μεταβλητή είναι λογική, δηλαδή παίρνει τιμές *True* ή *False,*
- run: Η συνάρτηση αποτελεί τον σκελετό της κλάσης καθώς ότι κώδικας βρίσκεται εδώ θα τρέξει ταυτόχρονα με το κύριο νήμα της εφαρμογής,
- start: Μόλις κληθεί η συνάρτηση, ξεκινά η εκτέλεση του κώδικα στην *run*,
- join: Ένα νήμα μπορεί να καλέσει την μέθοδο *join* ενός άλλου νήματος για να περιμένει να το πρώτο να ολοκληρώσει την εκτέλεσή του το δεύτερο.

3.1.2 Ουρές

Όπως προαναφέρθηκε, με την προσθήκη νημάτων στο σύστημα αυξάνεται η πολυπλοκότητα. Λόγω του ότι τα νήματα εκτελούνται ταυτόχρονα, δηλαδή είναι αδύνατον να γνωρίζουμε εκ των προτέρων την σειρά με την οποία θα ολοκληρωθούν, αναγκαζόμαστε να κάνουμε χρήση ουρών που μας επιτρέπουν να επεξεργαστούμε τα αποτελέσματα σειριακά.

Η βιβλιοθήκη της Python, με την ονομασία *Queue*, είναι ασφαλής όσον αφορά τα νήματα, που σημαίνει ότι ως δομή δεδομένων έχει μηχανισμούς για την προστασία της από την ταυτόχρονη πρόσβαση νημάτων που θα είχε ως αποτέλεσμα να βρεθεί σε άκυρη κατάσταση.

Επειδή η διαδικασία αναμονής για να φτάσει ένα μήνυμα στην ουρά είναι χρονοβόρα, ένα νήμα είναι το πλέον κατάλληλο για να περιμένει καθώς το κύριο νήμα της εφαρμογής είναι υπεύθυνο για όλα τα συστήματα και δεν μπορεί να "κολλάει". Στο πνεύμα αυτό θα αξιοποιήσουμε μια προσέγγιση σχεδίασης που ονομάζεται παραγωγός-καταναλωτής. Η ακόλουθη εικόνα απεικονίζει την διαδικασία:



Εικόνα 9: Παραγωγός και καταναλωτές

Το κόκκινο νήμα είναι ο παραγωγός μηνυμάτων που γράφει στην ουρά (μπλε). Τα δυο πορτοκαλί νήματα περιέχουν έναν ατέρμονο βρόχο που περιμένει για μήνυμα και μόλις διαβάσουν κάποιο το επεξεργάζονται και η διαδικασία επαναλαμβάνεται.

Η κλάση της βιβλιοθήκης *Queue* που θα χρησιμοποιήσουμε είναι η *Queue*. Είναι ουρά FIFO(First In, First Out) στην οποία η είσοδος γίνεται από το ένα άκρο της και η ανάγνωση από το άλλο άκρο της. Οι κύριες συναρτήσεις της είναι:

- *put*: Προσθέτει ένα αντικείμενο στην ουρά,
- *get*: Μπλοκάρει την εκτέλεση του νήματος μέχρι να διαβάσει ένα αντικείμενο.

Δηλαδή το νήμα παραγωγός χρησιμοποιεί την *put* ενώ το νήμα καταναλωτής την συνάρτηση *get*.

3.2 Αντικειμενοστραφής Προγραμματισμός

Πολλές σύγχρονες γλώσσες όπως είναι οι Java, C++, C#, Scala είναι αντικειμενοστραφείς, δηλαδή τα αντικείμενα είναι πολίτες πρώτης τάξης. Η Python υποστηρίζει ως κύριο είδος τον αντικειμενοστραφή προγραμματισμό αλλά δεν περιορίζεται σε αυτόν.

Τα στοιχεία των προγραμμάτων μας που αποκαλούμε αντικείμενα είναι μια μεταφορική έννοια. Περιέχουν πεδία δεδομένων που ουσιαστικά είναι τα χαρακτηριστικά με τα οποία τα περιγράφουμε. Επίσης περιλαμβάνουν συναρτήσεις που δρουν πάνω στα πεδία δεδομένων.

Είναι χρήσιμο να δούμε ένα παράδειγμα κλάσης πριν μπούμε στα ενδότερα της αντικειμενοστρέφειας. Ακολουθεί κώδικας σε Python:

```
class Rect(object):
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

    def isSquare(self):
        return self.width == self.height

r = Rect(5, 5)
print r.area() #25
print r.isSquare() #True
```

Η παραπάνω κλάση έχει όνομα *Rect.* Ο κατασκευαστής(constructor) είναι η πρώτη συνάρτηση που καλείται όταν δημιουργούμε ένα αντικείμενο της κλάσης και της περνάμε σαν παραμέτρους το πλάτος και το ύψος του ορθογωνίου οι οποίες αποθηκεύονται στο αντικείμενο με όνομα ίδιο με αυτό των παραμέτρων. Οι άλλες δυο συναρτήσεις, η *area* και η *isSquare,* δρουν πάνω στα δεδομένα, δηλαδή περιέχουν την λογική για να τα επεξεργαστούμε ή να εξάγουμε συμπεράσματα.

Υπάρχουν ορισμένες πρακτικές στην οργάνωση και ανάπτυξη προγραμμάτων τις οποίες προωθεί η αντικειμενοστραφής προσέγγιση:

- Η **ενθυλάκωση** οδηγεί σε ανεξάρτητες μονάδες που συνδέουν τις επεξεργαστικές μεθόδους με τα δεδομένα της μονάδας. Όπως προαναφέρθηκε οι μονάδες ονομάζονται κλάσεις και το στιγμιότυπο της, αντικείμενο. Έστω ότι έχουμε δυο στιγμιότυπα της κλάσης Rect, τα ASquare και NotASquare. Η ενθυλάκωση σιγουρεύει την ανεξάρτητη φύση τους, γεγονός που κρατάει διαχωρισμένες τις μεθόδους τους και τις καθιστά λιγότερο ευάλωτες σε συγκρούσεις,
- Με την **κληρονομικότητα** μπορούμε να οργανώσουμε τον κώδικα ιεραρχικά. Κάθε επίπεδο ιεραρχίας "γνωρίζει" ότι και το προηγούμενο επίπεδο, για παράδειγμα έχουμε την κλάση Animal που ξέρει πώς να τρέφεται. Δημιουργούμε την κλάση Bird που ξέρει να πετάει και κληρονομεί την Animal από την οποία μαθαίνει να τρέφεται. Σε πολύπλοκες εφαρμογές η προσπάθεια που πρέπει να κάνουμε για την επέκταση μιας κλάσης είναι η ελάχιστη δυνατή καθώς προσθέτουμε μόνο την καινούρια λειτουργικότητα,
- Ο **πολυμορφισμός** επιτρέπει στους προγραμματιστές να δημιουργούν διαδικασίες για αντικείμενα των οποίων ο τύπος δεν είναι γνωστός πριν την εκτέλεση του προγράμματος. Συνεχίζοντας το παράδειγμα από την κληρονομικότητα, δημιουργούμε τις κλάσεις Cat, Dog και Fox οι οποίες είναι Animal και ξέρουν πώς να τρέφονται, μπορεί με διαφορετικό τρόπο, αλλά η διαδικασία έχει ίδιο όνομα, έστω *feed*. Δημιουργούμε μια συνάρτηση που τυπώνει το φαγητό που τρώει το κάθε ζώο, δηλαδή καλείται η αντίστοιχη *feed*, με μια παράμετρο που έχει τύπο Animal χωρίς να χρειάζεται να αντιμετωπίσουμε κάθε περίπτωση ζώου ξεχωριστά.

Για να γίνουν κατανοητές η παραπάνω πρακτικές παρατίθεται ένα απλό παράδειγμα κώδικα σε Python:

```
class Animal(object):
    def feed(self):
        raise NotImplementedError("Subclass must implement
abstract method")

class Cat(Animal):
    def feed(self):
        return 'fish'

class Dog(Animal):
    def feed(self):
        return 'meat'

class Fox(Animal):
    def feed(self):
        return 'rabbits'

animals = [Cat(), Dog(), Fox()]
for animal in animals:
    print animal.feed()
```

Ο αντικειμενοστραφής προγραμματισμός έχει και τα μειονεκτήματά του, εκ των οποίων τα κυριότερα είναι:

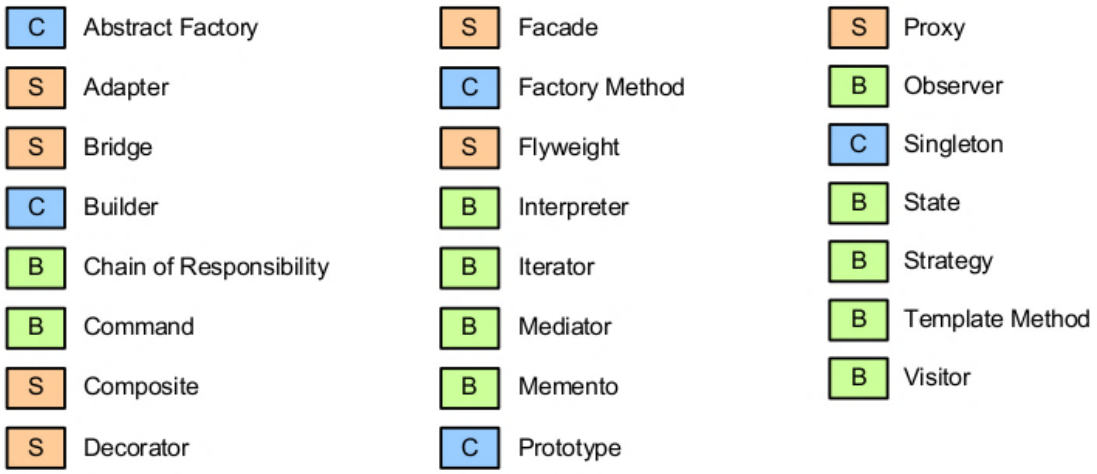
- Η μοντελοποίηση των κλάσεων είναι περίπλοκη διαδικασία,
- Ο αντικειμενοστραφής προγραμματισμός δεν είναι ικανός από μόνος του να προσδώσει στο έργο που αναπτύσσεται τα πλεονεκτήματα που προαναφέρθηκαν,
- Μεγάλη καμπύλη εκμάθησης του διαφορετικού τρόπου σκέψης,
- Όσο κάποιος μυείται στην αντικειμενοστραφή προσέγγιση συχνά θα αναρωτιέται αν γράφει "σωστά" τον κώδικα, με αποτέλεσμα η παραγωγικότητα του να ελαττώνεται όσο δοκιμάζει νέες ιδέες, στοχεύοντας σε μια καλή αντικειμενοστραφή λύση,
- Ο κώδικας συνήθως είναι μεγαλύτερος σε έκταση σε σύγκριση με την δομημένη του προσέγγιση,
- Αν δεν γίνει σωστή μοντελοποίηση ο κώδικας δεν θα συντηρείται εύκολα.

Τα χαρακτηριστικά του αντικειμενοστραφούς προγραμματισμού μας δίνουν τα εργαλεία για τον καθορισμό κανόνων προς την στοχευμένη επίλυση ποικίλων προβλημάτων, που ονομάζονται σχεδιαστικά μορφήματα(design patterns). Δεν αποτελούν έτοιμες λύσεις που θα προσαρμοστούν στην κάθε περίπτωση και θα είναι έτοιμα για εκτέλεση. Ο προγραμματιστής καλείται να το υλοποιήσει στην εφαρμογή ακολουθώντας τους κανόνες που υπαγορεύονται.

Ο αποτελεσματικός σχεδιασμός λογισμικού απαιτεί προνοητικότητα σε προβλήματα που μπορεί να μην είναι εμφανή εξ αρχής. Η ανακύκλωση των σχεδιαστικών μορφημάτων βοηθάει στην αντιμετώπιση των δύσκολων στον εντοπισμό λαθών που προκαλούν μεγάλα προβλήματα και βελτιώνει την αναγνωσιμότητα του κώδικα για προγραμματιστές και αρχιτέκτονες λογισμικού που είναι εξοικειωμένοι με τα μορφήματα.

Κατά την ενσωμάτωση τους όμως στο πρόγραμμα εισαγάγουν επιπρόσθετα επίπεδα ανακατεύθυνσης που σε ορισμένες περιπτώσεις περιπλέκουν το τελικό αποτέλεσμα και μειώνουν την απόδοση της εφαρμογής.

Στο βιβλίο "Design Patterns: Elements of Reusable Object-Oriented Software" [GHJV] αναλύονται 23 συνήθη σχεδιαστικά μορφήματα, που καταγράφονται στην εικόνα που ακολουθεί.



Εικόνα 10: Κατηγορίες Μορφημάτων

3.3 Μηχανές Πεπερασμένων Καταστάσεων

Τα αρχεία που είναι ενεργά στην εφαρμογή, δηλαδή ανεβαίνουν, είναι σταματημένα ή σε κάποια κατάσταση λάθους, μπορούν να μεταβούν σε προκαθορισμένες καταστάσεις αναλόγως της τρέχουσας κατάστασης και του είδους μηνύματος που το αφορά.

Για την μοντελοποίηση του παραπάνω μηχανισμού θα χρησιμοποιηθεί μια μηχανή πεπερασμένων καταστάσεων. Δεν θα περιγραφεί ο γενικός ορισμός και η χρήση τους αλλά θα γίνει μια στοχευμένη περιγραφή για τον σκοπό που έχουν στην παρούσα εργασία. Επομένως, οι καταστάσεις κάθε ενεργού αρχείου θα ακολουθούν την ροή της μηχανής πεπερασμένων καταστάσεων και ανά ένα αρχείο θα έχουμε μια μηχανή.

Ορισμοί

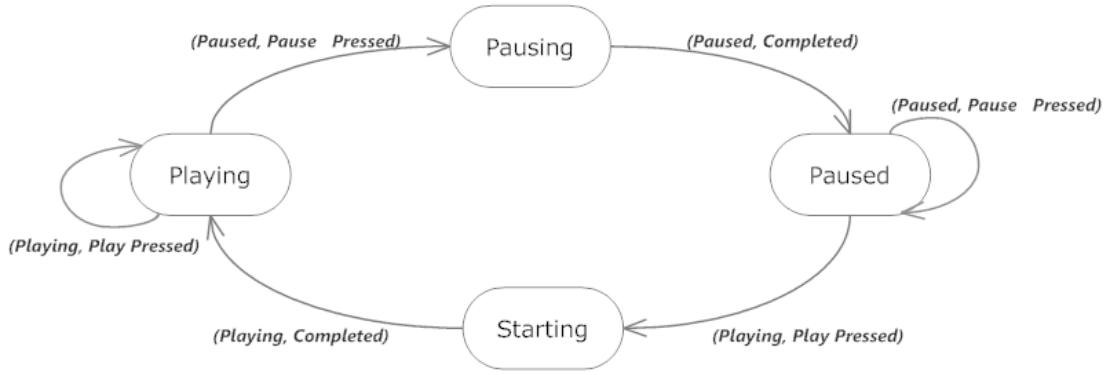
- *S*: Το σύνολο των σταθερών καταστάσεων, δηλαδή οι καταστάσεις που αλλάζουν μόνο από ενέργεια του χρήστη ή από εμφάνιση λάθους,
- *T*: Το σύνολο των μεταβατικών καταστάσεων, που έχουν ως στόχο τον καλύτερο έλεγχο της διαδικασίας από την μεριά του σχεδιαστή και την βελτίωση της εμπειρίας για τον χρήστη,
- *Πίνακας Καταστάσεων*: Είναι ο πίνακας που έχει καταγεγραμμένη την συμπεριφορά της μηχανής ανάλογα με την είσοδο και την τρέχουσα κατάσταση,
- *Μετάβαση*: Είναι μια συνάρτηση που διαβάζει τον πίνακα καταστάσεων και επιστρέφει ως αποτέλεσμα μια νέα κατάσταση. Ο τυπικός ορισμός της είναι:

  $$f: (S \cup T) \rightarrow (S \cup T)$$

  Και στα διαγράμματα θα την συμβολίζουμε ως:

  $$(ΚατάστασηΣτόχος, ΑιτίαΜετάβασης)$$

Θα ολοκληρώσουμε αναλύοντας την απλοποιημένη μηχανή καταστάσεων ενός προγράμματος αναπαραγωγής μουσικής:



Εικόνα 11: Μηχανή Πεπερασμένων καταστάσεων

Το διάγραμμα μηχανής καταστάσεων έχει δυο σταθερές καταστάσεις, τις Playing και Paused, και δυο μεταβατικές καταστάσεις, την Pausing και την Starting. Έστω ότι η τρέχουσα κατάσταση είναι η 'Playing'. Αν ο χρήστης πατήσει το κουμπί 'Play' δεν θα αλλάξει κάτι γιατί η μουσική παίζει ήδη. Αν όμως πατήσει το 'Pause', το τραγούδι που αναπαράγεται εκείνη τη στιγμή θα μεταβεί στην 'Pausing' και κατά την ολοκλήρωση της διαδικασίας θα μεταβεί αυτόματα στην 'Paused'.

3.4 Το πρωτόκολλο εξουσιοδότησης OAuth

Το OAuth δημιουργήθηκε ως ανοιχτό πρότυπο με στόχο την πρόσβαση των εφαρμογών-πελατών στους πόρους του εξυπηρετητή εκ μέρους των κατόχων των πόρων. Επίσης, παρέχει μια διαδικασία εξουσιοδότησης έτσι ώστε ο τελικός χρήστης να μην εισαγάγει ευαίσθητα δεδομένα, όπως το όνομα χρήστη και το συνθηματικό, στην εφαρμογή αλλά τελικά να αποκτήσει πρόσβαση στους πόρους. Η διαδικασία ονομάζεται ροή εξουσιοδότησης(*authorization flow*).

Η πρώτη έκδοση του εν λόγω πρωτοκόλλου δημοσιεύτηκε τον Απρίλιο του 2010 ως RFC 5849. Είναι ιδιαίτερα περίπλοκο πρότυπο από άποψη υλοποίησης με αποτέλεσμα ορισμένες σχεδιαστικές αποφάσεις να περιέχουν τρύπες ασφαλείας.

Πέντε χρόνια μετά δημοσιεύτηκε η δεύτερη προσέγγιση, το OAuth 2.0 Framework, ως RFC 6750, τον Οκτώβρη του 2012. Πολλές σχεδιαστικές αποφάσεις εγκαταλείφθηκαν για να απλοποιηθεί η υλοποίηση που έπρεπε να γίνει από τους εξυπηρετητές και η ροή εξουσιοδότησης.

Στην εφαρμογή που υλοποιήθηκε, όσες υπηρεσίες είχαν ροές εξουσιοδότησης προτείνανε την χρήση της OAuth 2.0 ροής η οποία θα αναλυθεί στη συνέχεια.

Ορισμοί

- Μυστικό καταναλωτή(*consumer secret*): Κωδικός που δίνεται από την cloud υπηρεσία για την εφαρμογή για την εκκίνηση των χειραψιών OAuth,
- Κλειδί καταναλωτή(*consumer key*): Ο δεύτερος κωδικός για την χειραψία OAuth,
- Παράμετρος: Κομμάτια πληροφορίας των επικεφαλίδων από τα HTTP πακέτα που ανταλλάσσονται κατά την διαδικασία εξουσιοδότησης,
- Κωδικός εξουσιοδότησης(*authorization code*): ανταλλάσσεται για το τελικό κλειδί,
- Κλειδί OAuth(*OAuth token*): Το τελικό κλειδί για την επικοινωνία της εφαρμογής με τον εξυπηρετητή.

Η ροή που χρησιμοποιήθηκε ονομάζεται Three-Legged, εξαιτίας τον προσώπων που συμμετέχουν στην διαδικασία, η εφαρμογή, ο χρήστης και η υπηρεσία. Υπάρχουν περιπτώσεις που η εφαρμογή δεν χρειάζεται την συμμετοχή του χρήστη στην ροή για αυτό ονομάζεται Two-Legged.

Η Three-Legged ροή αποτελείται από 6 βήματα:

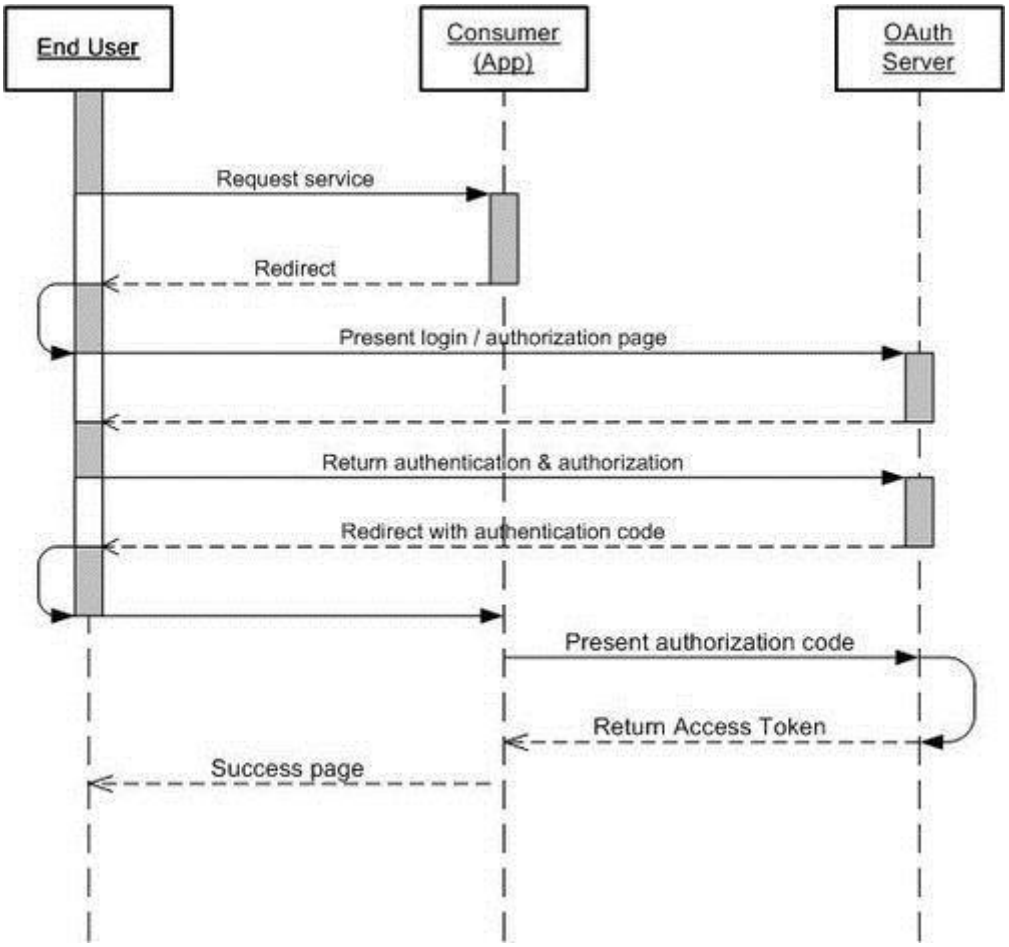
1. Η εφαρμογή ανοίγει το πρόγραμμα διαδικτυακής περιήγησης του χρήστη και τον ανακατευθύνει σε μια σελίδα της υπηρεσίας. Οι πληροφορίες που χρειάζονται ως παράμετροι στο βήμα αυτό είναι οι εξής: κλειδί καταναλωτή, σελίδα ανακατεύθυνσης όταν γίνει η είσοδος στην υπηρεσία, τύπος απάντησης που πρέπει να τεθεί στην τιμή '*code*', εμβέλεια που αντιπροσωπεύει τα δικαιώματα της εφαρμογής στα δεδομένα του χρήστη. Ένας σύνδεσμος για παράδειγμα έχει την ακόλουθη μορφή:

```
https://oauth_service/login/oauth/authorize?client_id=3MVG9lKcPoNINVB&redire
ct_uri=http://localhost/oauth/code_callback&scope=user
```

2. Ο χρήστης εισάγαγει τα προσωπικά του στοιχεία στην ιστοσελίδα της υπηρεσίας,
3. Η υπηρεσία ανακατευθύνει τον χρήστη στην σελίδα που είχε οριστεί στην σελίδα ανακατεύθυνσης στην αποστολή του πρώτου πακέτου, που περιέχει τον κωδικό εξουσιοδότησης,
4. Η εφαρμογή στέλνει τον κωδικό και τον ανταλλάσσει για το κλειδί OAuth συνοδεύοντας τον με τις απαραίτητες πληροφορίες όπως είναι ο κωδικός και το κλειδί καταναλωτή, ο κωδικός εξουσιοδότησης και ο τύπος του κωδικού(*grant type*) με την τιμή '*authorization code*',
5. Αν τα στοιχεία είναι έγκυρα η υπηρεσία θα απαντήσει με το κλειδί OAuth,
6. Η εφαρμογή αποθηκεύει το κλειδί για μελλοντική χρήση.

Ορισμένες υπηρεσίες προσφέρουν ζεύγος κλειδιών, ένα με μικρό χρόνο ζωής που χρησιμοποιείται για την επικοινωνία και ένα με πολύ μεγάλο χρόνο ζωής για την ανανέωση του πρώτου.

Η διαδικασία συνοψίζεται παρακάτω:



Εικόνα 12: Ροή OAuth 2.0

3.5 *Επικοινωνία με τις υπηρεσίες*

Κάθε υπηρεσία που θέλει να την χρησιμοποιήσουν οι προγραμματιστές στις εφαρμογές τους πρέπει να προσφέρουν εργαλεία για εύκολη προγραμματιστική επικοινωνία. Τα εργαλεία συνήθως καλύπτουν μεγάλο εύρος τεχνολογιών και γλωσσών προγραμματισμού. Σε αυτή τη φιλοσοφία αναπτύσσουν και δημοσιεύουν τις προγραμματιστικές διαπροσωπείες εφαρμογών(Application Programming Interfaces, APIs).

3.5.1 Προγραμματιστική Διαπροσωπεία Εφαρμογής

Ορίζεται ως ένα σύνολο κανόνων και προδιαγραφών, το οποίο ακολουθούν οι εφαρμογές για να επικοινωνούν μεταξύ τους. Οι δημιουργοί των APIs σχεδιάζουν με πολύ προσοχή τον τρόπο με τον οποίο θα επικοινωνούν άλλες εφαρμογές με τις δικές τους για να είναι όσο πιο λειτουργικές γίνεται για τους τρίτους και ταυτόχρονα να μην εκτίθενται, από άποψη ασφαλείας, κομμάτια της δικιάς τους υποδομής.

Τα APIs υπάρχουν παντού, από το διαδίκτυο μέχρι την κάρτα γραφικών και τον επεξεργαστή του υπολογιστή. Ας δούμε ονομαστικά κάποια:

- Το σύνολο εντολών για επεξεργαστές της Intel με αρχιτεκτονική 32bit(x86),
- OpenGL, για την δημιουργία γραφικών και παιχνιδιών,
- WinAPI, για κλήσεις συστήματος σε λειτουργικά Microsoft Windows,
- Κλάσεις και μέθοδοι στις βασικές βιβλιοθήκες της Python,
- Το Document Object Model(DOM) που το εκθέτουν οι διαδικτυακοί περιηγητές στην JavaScript,
- Διαδικτυακές υπηρεσίες, που δημοσιεύουν τα APIs τους όπως το Twitter API, το Flickr API και το Facebook Graph API,
- Υλοποιήσεις πρωτοκόλλων, για παράδειγμα η βιβλιοθήκη JNI της Java.

Στον παρακάτω κώδικα οι μέθοδοι .nextLine() και .close() είναι μέρος του API της κλάσης Scanner στην Java:

```
public class Test {
   public static void main(String[] args) {
        System.out.println("Enter your name: ");
        Scanner inputScanner = new Scanner(System.in);
        String name = inputScanner.nextLine();
        System.out.println("Your name is " + name + ".");
        inputScanner.close();
  }
}
```

Ολοκληρώνοντας, ένα API ανεξαρτήτως της ποιότητας του, είναι απαραίτητο να συνοδεύεται από λεπτομερή τεκμηρίωση και παραδείγματα χρήσης.

3.5.2 Διαπροσωπείες Υπηρεσιών

Στην εφαρμογή που κατασκευάστηκε επικοινωνούμε με τις υπηρεσίες Dropbox, GoogleDrive και Pithos+. Χρησιμοποιήθηκαν τα πιο πρόσφατα APIs της κάθε υπηρεσίας που ήταν διαθέσιμα μέχρι την στιγμή της ολοκλήρωσης της εφαρμογής.

 **Dropbox**

Όνομα: dropbox,

Έκδοση: 1.6, τελευταία ανανέωση 7 Ιουλίου 2013,

+ Καλογραμμένη και κατανοητή βιβλιοθήκη,
+ Λεπτομερής τεκμηρίωση,
+ Εύκολα επεκτάσιμη προγραμματιστικά,
– Ορισμένοι αδικαιολόγητοι περιορισμοί

Λόγω του ότι η εταιρεία προσφέρει στοχευμένη και ώριμη υπηρεσία, είναι αναμενόμενο να είναι ιδιαιτέρως προσεγμένη.

 **GoogleDrive**

Όνομα: google-api-python-client

Έκδοση: 1.2, τελευταία ανανέωση 7 Αυγούστου 2013,

+ Καλογραμμένη βιβλιοθήκη,
+ Λεπτομερής τεκμηρίωση,
– Μέτρια καμπύλη εκμάθησης μέχρι να μπορέσει ο προγραμματιστής να χρησιμοποιεί με ευχέρεια την βιβλιοθήκη και την τεκμηρίωση,

Η Google χρησιμοποιεί την βιβλιοθήκη αυτή όχι μόνο για το Drive αλλά για όλες τις υπηρεσίες της με αποτέλεσμα να είναι πολύ γενική και σε πρώτο στάδιο δυσνόητη. Όταν εξοικειωθεί κάποιος με το API και την τεκμηρίωση, δεν υπάρχει κάτι που δεν μπορεί να επιτύχει.

 **Pithos+**

Όνομα: astakosclient,

kamaki

Έκδοση: astakosclient: 0.15-py2.7, τελευταία ανανέωση 17 Δεκεμβρίου 2013

kamaki: 0.11.3-py2.7, τελευταία ανανέωση 23 Οκτώβρη 2013,

+ Καλογραμμένες βιβλιοθήκες,
+ Εύκολα επεκτάσιμες προγραμματιστικά,
– Μεγάλη καμπύλη εκμάθησης όσον αφορά την εξοικείωση με την παρεχόμενη βιβλιοθήκη και την τεκμηρίωση,

Η βιβλιοθήκη astakosclient χρησιμοποιήθηκε στην διαδικασία εξουσιοδότησης και η kamaki για την αποστολή των αρχείων στο Pithos+.

3.6 Αποθήκευση τοπικών δεδομένων

Όλες οι εφαρμογές χρειάζονται να αποθηκεύουν δεδομένα στον υπολογιστή που εκτελούνται έτσι ώστε ο τελικός χρήστης να έχει την καλύτερη εμπειρία χρήσης. Η πλειοψηφία των δημοφιλών προϊόντων χρησιμοποιεί δικό της τύπο αρχείων, που μόνο οι δημιουργοί γνωρίζουν πώς να τα διαβάζουν.

Η υπό ανάπτυξη εφαρμογή, για λόγους απλότητας, θα αποθηκεύει τα δεδομένα που χρειάζεται σε απλά αρχεία κειμένου με την κατάληξη .ini. Το θέμα ασφαλείας των κλειδιών είναι γνωστό αλλά η επιλογή και αξιολόγηση οποιουδήποτε σχήματος προστασίας είναι χρονοβόρα διαδικασία με αποτελέσματα τα οποία μπορεί να μην είναι ικανοποιητικά.

Τα δεδομένα που χρειάζονται να αποθηκευτούν είναι τα εξής:

- Πληροφορίες για την επανεκκίνηση των αρχείων, που ποικίλουν ανάλογα με την υπηρεσία,
- Ιστορικό διαμοιρασμένων αρχείων,
- Κλειδιά για την επικοινωνία με τις υπηρεσίες,
- Ρυθμίσεις του παραθυρικού περιβάλλοντος.

Η μορφή των αρχείων είναι πολύ απλή και αναγνώσιμη από ανθρώπους σε περίπτωση που χρειαστεί να γίνει επεξεργασία. Η γενική μορφή τους είναι η παρακάτω:

```
#Filename: general.ini
# This is the 'initial_comment'
# Which may be several lines
keyword1 = value1

[section 1]
# This comment goes with keyword 2
keyword 2 = value 2
keyword 3 = value 3, value 4, 'value 5'

    [[ sub-section ]]    # an inline comment
    # sub-section is inside "section 1"
    'keyword 4' = 'value 6'

        [[[ sub-sub-section ]]]
        # sub-sub-section is *in* 'sub-section'
        # which is in 'section 1'
        'keyword 5' = 'value 7'

[section 2]
Keyword 6 = 'value 8'
Keyword 7 = value 9     # an inline comment
# The 'final_comment'
```

Για προγραμματιστική διευκόλυνσή θα χρησιμοποιήσουμε την βιβλιοθήκη ConfigObj της Python. Φορτώνει τα δεδομένα στη μορφή λεξικού(dictionary), μια από τις βασικές δομές δεδομένων της Python, γεγονός που μας επιτρέπει την εύκολη χρήση τους στον κώδικα. Παραδείγματος χάριν αν θέλουμε να διαβάσουμε το keyword 4 θα γράψουμε:

```
settings = configobj.ConfigObj('general.ini')
keyword4 = settings['section1']['sub-section']['keyword 4']
```

3.7 Γραφικό Περιβάλλον

Η γλώσσα προγραμματισμού που επιλέχτηκε για την δημιουργία της εφαρμογής είναι ανεξάρτητη του λειτουργικού συστήματος. Θέλοντας να μείνουμε στο ίδιο εύρος λειτουργικών συστημάτων, φυσικό επόμενο είναι να επιλέξουμε μια βιβλιοθήκη για γραφικό περιβάλλον με την ίδια ιδιότητα.

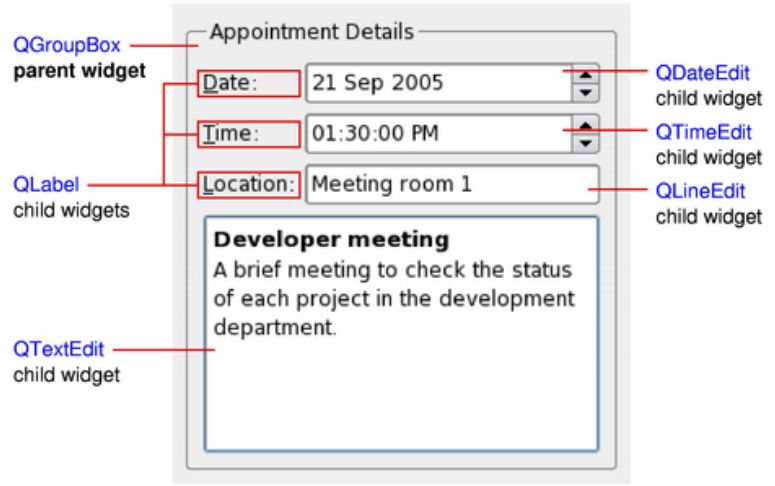
Τα κριτήρια αναζήτησης ήταν τα εξής:

- Απλή στον προγραμματισμό,
- Πλήρης από άποψη εργαλείων,
- Ελευθερία παραμετροποίησης της διαδικασίας σχεδιασμού στην οθόνη,
- Να συνδέεται ανώδυνα με τα υπόλοιπα υποσυστήματα της εφαρμογής,
- Ενεργή κοινότητα,
- Συχνές ανανεώσεις,
- Κατανοητή και ολοκληρωμένη τεκμηρίωση,
- Πληθώρα παραδειγμάτων,

Η βιβλιοθήκη που ικανοποιεί στον μεγαλύτερο βαθμό τα παραπάνω κριτήρια είναι η PyQt. Λειτουργεί ως μεσάζων για την δημοφιλή βιβλιοθήκη δημιουργίας γραφικού περιβάλλοντος Qt, που είναι γραμμένη σε C++. Η PyQt δημιουργήθηκε από την αγγλική Riverbank Computing και παρέχεται με την άδεια χρήσης GNU General Public License. Λόγω της δεσμευτικής άδειας χρήσης σε περίπτωση πώλησης της εφαρμογής, η Nokia δημιούργησε την βιβλιοθήκη PySide που είναι πανομοιότυπη με την PyQt αλλά με άδεια χρήσης LGPL.

Τα εργαλεία που παρέχονται στον προγραμματιστή καλύπτουν ένα μεγάλο εύρος των αναγκών που υπάρχουν σήμερα, όπως είναι η ενσωμάτωση βάσεων δεδομένων, η αναπαραγωγή ήχου, η απεικόνιση γραφικών με την OpenGL, οι δικτυακές εφαρμογές. Επίσης, παρέχεται το πρόγραμμα QtDesigner που, χωρίς κώδικα, επιτρέπει την σχεδίαση παραθύρων. Με την ολοκλήρωση της διαδικασίας όμως ο προγραμματιστής καλείται να προσθέσει την λειτουργικότητα που χρειάζεται γράφοντας τον κατάλληλο κώδικα.

Για να αποκτήσει ο αναγνώστης μια ιδέα για τα στοιχεία που χρειάζονται για ένα απλό παράθυρο, παραθέτουμε μια εικόνα στην οποία έχουν επισημανθεί οι ονομασίες των κλάσεων για την κάθε συνιστώσα:



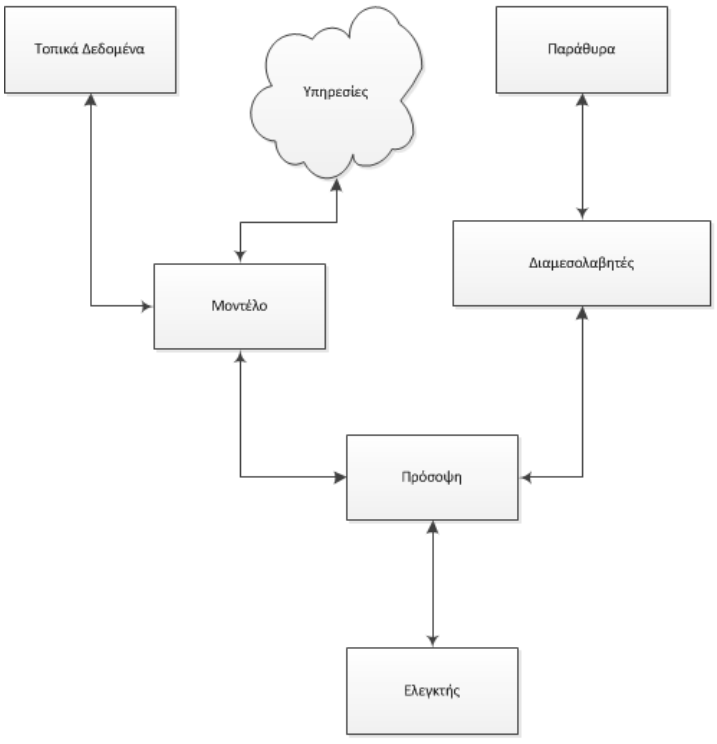
Εικόνα 13: Εργαλεία της PyQt4

4

Λειτουργικότητα του Συστήματος

Στο παρόν κεφάλαιο θα παρουσιαστούν τα παράθυρα που απαρτίζουν το γραφικό περιβάλλον της εφαρμογής, θα αναλυθούν λεπτομερώς οι λειτουργικές και θα καταγραφούν οι ποιοτικές του συστήματος μας.

Παρακάτω απεικονίζονται οι υπομονάδες του συστήματός και η σύνδεση τους σε απλοποιημένη μορφή:



Εικόνα 14: Απλοποιημένη παρουσίαση του συστήματος

4.1 Περιγραφή Λειτουργιών

Στην ενότητα αυτή θα γίνει μια επισκόπηση στις λειτουργίες του γραφικού περιβάλλοντος της εφαρμογής ανεξαρτήτως της υποδομής που θα το υποστηρίζει. Τα παράθυρα του γραφικού περιβάλλοντος είναι ο μοναδικός τρόπος που δίνεται στον χρήστη για να αλληλεπιδράσει με την εφαρμογή, επομένως οφείλει να είναι όσο πιο φιλικό γίνεται στην χρήση του και να είναι πλήρες από άποψη χαρακτηριστικών. Ανεξαρτήτως της επαναστατικότητας μιας ιδέας και της υλοποίησης που υποστηρίζει το γραφικό περιβάλλον, αν προορίζεται για εμπορική εφαρμογή, και όχι μόνο, πρέπει να καλύπτει τα παραπάνω χαρακτηριστικά. Για να το πετύχει αυτό το σύστημα που αναπτύσσουμε, θα περιγραφούν και θα αναλυθούν τα παράθυρα που ακολουθούν από την πλευρά των λειτουργικών απαιτήσεων του χρήστη.

4.1.1 Εργαλείο στην επιφάνεια εργασίας

Το παράθυρο αυτό θα βρίσκεται στην επιφάνεια εργασίας του χρήστη και θα λειτουργεί ως το κύριο μέσο για τον διαμοιρασμό αρχείων. Για κάθε υπηρεσία που έχει εξουσιοδοτήσει την εφαρμογή ο χρήστης, θα απεικονίζεται το εικονίδιο της και θα λειτουργεί ως στόχος στον οποίον θα σέρνει ο χρήστης τα αρχεία του για τον διαμοιρασμό τους.

Τα χαρακτηριστικά που πρέπει να έχει είναι τα εξής:

- Ανατροφοδότηση για τις ενέργειες του χρήστη, δηλαδή αν ανεβαίνει στην υπηρεσία ένα αρχείο το προσκήνιο του εικονιδίου να γίνεται πράσινο, σε περίπτωση σφάλματος να γίνεται κόκκινο αλλιώς να είναι διάφανο,
- Οι υπηρεσίες να σχεδιάζονται σε γραμμή κάθετα ή οριζόντια με την δυνατότητα εναλλαγής ανάμεσα στις δυο προβολές,
- Ο χρήστης να μπορεί να μετακινήσει το παράθυρο σε οποιοδήποτε τμήμα της οθόνης του επιθυμεί,
- Όσο είναι στη διαδικασία μεταφοράς των αρχείων προς το παράθυρο, αν ο δείκτης του ποντικιού είναι πάνω από κάποιο εικονίδιο θα γίνεται ελαφρώς μεγαλύτερο από τα υπόλοιπα,
- Στην περίπτωση που καμία υπηρεσία δεν έχει εξουσιοδότηση από τον χρήστη θα είναι αόρατο μέχρι την προσθήκη κάποιας.

4.1.2 Παράθυρο λεπτομερειών

Το προηγούμενο παράθυρο καλύπτει την αμεσότητα και την απλότητα της εφαρμογής αλλά ο χρήστης πρέπει να έχει πρόσβαση σε περισσότερες πληροφορίες και έλεγχο επί αυτών. όπως η κατάσταση των αρχείων που ανεβαίνουν, το ιστορικό, η αλλαγή των παραμέτρων και η προσθαφαίρεση λογαριασμών.

Το παράθυρο λεπτομερειών πρέπει να έχει τα ακόλουθα γνωρίσματα:

- Στο πάνω μέρος του παραθύρου θα υπάρχουν τα κουμπιά ελέγχου με αντίστοιχο του ρόλου τους εικονίδιο. Οι ρόλοι που πρέπει να καλύψουν είναι της προσθήκης αρχείων μέσω ενός ειδικού παραθύρου, της εκκίνησης, της παύσης και της διαγραφής αρχείων. Τα εν λόγω κουμπιά θα λειτουργούν για οποιαδήποτε οθόνη ενεργοποιώντας την κατάλληλη ενέργεια,
- Στην οθόνη των ενεργών αρχείων για κάθε ένα αρχείο, θα προβάλλονται πληροφορίες όπως είναι το όνομα του, η πρόοδος, η υπηρεσία στην οποία ανεβαίνει, η κατάσταση του, ο απομακρυσμένος φάκελος αποθήκευσης και ο τρόπος με τον οποίο θα επιλυθεί μια πιθανή σύγκρουση ονόματος,
- Παραμένοντας στην ίδια οθόνη, ο χρήστης θα έχει την δυνατότητα να επιλέξει, με οικείο τρόπο, πολλαπλά αρχεία και να ελέγξει την κατάστασή τους με ένα πάτημα κάποιου κουμπιού,
- Συνεχίζοντας, στην οθόνη ιστορικού θα καταγράφονται όσα αρχεία έχουν ανέβει μέσω της εφαρμογής. Οι πληροφορίες που θα προβάλλονται για το κάθε αρχείο θα είναι το όνομά του, ο απομακρυσμένος φάκελος, η υπηρεσία και η ημερομηνία και ώρα ολοκλήρωσης του ανεβάσματος,
- Στην οθόνη του ιστορικού τα κουμπιά εκκίνησης και παύσης θα είναι ανενεργά, καθώς δεν έχουν νόημα στο περιεχόμενο αυτό. Επιτρέπεται η μαζική διαγραφή αρχείων, η προσθήκη νέων για διαμοιρασμό και η αντιγραφή συνδέσμων στον πρόχειρο χώρο του λειτουργικού συστήματος,
- Η οθόνη ρυθμίσεων θα χωριστεί σε δυο λογικά σκέλη, των γενικών ρυθμίσεων που αφορούν την γενική συμπεριφορά της εφαρμογής και της διαχείρισης των λογαριασμών. Στο τελευταίο ο χρήστης θα μπορεί να εξουσιοδοτήσει την εφαρμογή για μια υπηρεσία καθώς και να ρυθμίσει μια, ήδη εξουσιοδοτημένη, υπηρεσία.

4.1.3 Παράθυρο για την ενημέρωση ολοκλήρωσης αποστολής

Ο σκοπός του παραθύρου ενημέρωσης είναι να εμφανίζεται μόλις ολοκληρωθεί ο διαμοιρασμός κάποιου αρχείου και θα δίνεται η δυνατότητα στον χρήστη να αντιγράψει στον πρόχειρο χώρο του λειτουργικού του τον σύνδεσμο για το αρχείο.

Τα χαρακτηριστικά του παραθύρου αυτού πρέπει να είναι:

- Προβολή των αναγκαίων πληροφοριών, όπως είναι το όνομα του αρχείου στην υπηρεσία, η ημερομηνία ολοκλήρωσης και το όνομα της υπηρεσίας, με τρόπο που καλύπτει όσο λιγότερο χώρο γίνεται,
- Λόγω της φύσης του παραθύρου δεν είναι πρακτικό να δείχνουμε όλο το ιστορικό παρά μόνο τα πιο πρόσφατα,
- Ο σύνδεσμος διαμοιρασμού συνήθως είναι μεγάλος σε μήκος για αυτό δεν θα προβάλλεται στο παράθυρο. Με το πέρασμα του δείκτη του ποντικιού πάνω από μια εγγραφή θα εμφανίζεται ένα κουμπί το οποίο μπορεί να πατήσει για να αντιγράψει τον σύνδεσμο,
- Το παράθυρο δεν θα μετακινείται αλλά θα εμφανίζεται, ανάλογα με την θέση της γραμμής εργαλείων, σε κατάλληλη θέση.

4.1.4 Εικονίδιο στην γραμμή εργαλείων

Αποτελεί το σημείο διαχείρισης της εφαρμογής, δίνοντας επιλογές στον χρήστη για την εμφάνιση του παραθύρου λεπτομερειών, της οθόνης ρυθμίσεων, της προσθαφαίρεσης λογαριασμών, του παραθύρου αποστολής σχολίων για την εφαρμογή. Επίσης, είναι το μόνο παράθυρο που δίνει την δυνατότητα ομαλής εξόδου από την εφαρμογή. Τέλος, στις παραπάνω δυνατότητες θα μπορούσαν να προστεθούν συντομεύσεις για προκαθορισμένες ενέργειες παραδείγματος χάριν η παύση/εκκίνηση όλων των αρχείων.

4.1.5 Παράθυρο αποστολής σχολίων

Το τελευταίο παράθυρο θα αναλαμβάνει την αποστολή ηλεκτρονικού μηνύματος στον δημιουργό με σχόλια, προτάσεις και σφάλματα προς διόρθωση. Ο χρήστης θα εισαγάγει αν επιθυμεί το όνομά του, τη διεύθυνση ηλεκτρονικού ταχυδρομείου και το μήνυμα του.

4.2 Λειτουργικές απαιτήσεις συστήματος

Στην ενότητα αυτή θα εξετάσουμε πως το σύστημα που αναπτύσσουμε πρέπει να αντιδρά στις εισόδους του χρήστη και πως θα συμπεριφερθεί σε ιδιαίτερες περιπτώσεις. Οι απαιτήσεις που αφορούν τα παραπάνω θέματα ονομάζονται λειτουργικές και καθορίζουν τον σχεδιασμό του συστήματος. Για να είναι χρήσιμες στους αναγνώστες τους πρέπει να είναι γραμμένες βάσει συγκεκριμένων κανόνων εκ των οποίων οι βασικότεροι είναι:

- Κάθε απαίτηση να έχει πολύ συγκεκριμένο θέμα,
- Δεν πρέπει να είναι σχεδιαστική απόφαση, για παράδειγμα η πρόταση "Το σύστημα θα αποθηκεύει τα δεδομένα του στην MySQL βάση δεδομένων" αναφέρει πως θα χρησιμοποιηθεί η MySQL, που μπορεί να είναι καλή επιλογή τελικά, αλλά δεν είναι το στάδιο για την επιλογή τεχνολογιών,
- Σε κάθε απαίτηση πρέπει να περιλαμβάνεται η λογική για την εισαγωγή της,
- Να είναι εύκολος ο έλεγχος τους γιατί θα είναι η κύρια πηγή για την συγγραφή κώδικα που θα ελέγχει την ορθή λειτουργία τους,

4.2.1 Αποστολή αρχείων στις υπηρεσίες

Αιτιολόγηση: Αποτελεί τον στόχο του συστήματος.

Αναγνωριστικό: Λ-1

Περιγραφή: Μέσω των παραθύρων που παρουσιάστηκαν στην προηγούμενη ενότητα ο χρήστης θα μπορεί να ανεβάσει ένα ή πολλαπλά αρχεία σε μια από τις υπηρεσίες που έχει εξουσιοδοτήσει την εφαρμογή.

Ενεργοποίηση/Ακολουθία αντίδρασης:

1. Ο χρήστης εισαγάγει μέσω ενός παραθύρου τα αρχεία προς ανέβασμα,
2. Το σύστημα εκκινεί την διαδικασία ανεβάσματος,
3. Το σύστημα ανεβάζει τα αρχεία,
4. Ο χρήστης ενημερώνεται για την έκβαση της διαδικασίας, σε περίπτωση επιτυχίας μπορεί να αντιγράψει τους συνδέσμους, αλλιώς εμφανίζεται ανάλογο μήνυμα λάθους,

4.2.2 Εξουσιοδότηση Λογαριασμού

Αιτιολόγηση: Χωρίς αυτή την λειτουργία ο χρήστης δε θα μπορεί να ανεβάσει κάποιο αρχείο.

Αναγνωριστικό: Λ-2

Περιγραφή: Για να μπορέσει να ανεβάσει ο χρήστης αρχεία σε κάποια υπηρεσία πρέπει να έχει προηγηθεί η διαδικασία της εξουσιοδότησης.

Ενεργοποίηση/Ακολουθία αντίδρασης:

1. Ο χρήστης ανοίγει το παράθυρο προσθαφαίρεσης λογαριασμών,
2. Πατάει το κουμπί που εκκινεί την διαδικασία εξουσιοδότησης,
3. Εμφανίζεται η ιστοσελίδα της υπηρεσίας,
4. Εισαγάγει τα στοιχεία που απαιτεί η υπηρεσία για ταυτοποίηση, συνήθως όνομα χρήστη και συνθηματικό,
5. Αποδέχεται την εξουσιοδότηση της εφαρμογής μόλις ερωτηθεί,
6. Αντιγράφει τον κωδικό εξουσιοδότησης,
7. Επικολλά τον κωδικό στην εφαρμογή στο κατάλληλο πεδίο,
8. Πατάει το κουμπί που ελέγχει αν ο κωδικός είναι έγκυρος,
9. Αν είναι έγκυρος μπορεί να χρησιμοποιήσει την υπηρεσία στην εφαρμογή αλλιώς πρέπει να ξαναπροσπαθήσει.

4.2.3 Περιπτώσεις Χρήσης

Για κάθε μια από τις απαιτήσεις θα δημιουργήσουμε διαγράμματα περιπτώσεων χρήσης που στοχεύουν στην σχηματική αναπαράσταση της διαδικασίας για κάθε δυνατή περίπτωση και στην προσθήκη απαραίτητων πληροφοριών που δεν αφορούν την λειτουργική απαίτηση.

4.2.3.1 Αποστολή αρχείων στις υπηρεσίες

Συντόμευση: ΔΧ-1,

Στόχος: Αποστολή ενός ή περισσότερων αρχείων σε επιλεγμένη υπηρεσία,

Πηγή: Χρήστης,

Δράστες: Χρήστης, Υπηρεσία,

Συνθήκες: Ο χρήστης έχει εξουσιοδοτήσει την υπηρεσία μέσω της εφαρμογής, ροή ΔΧ-2,

Αποτέλεσμα: Τα αρχεία βρίσκονται στους εξυπηρετητές της υπηρεσίας και ο χρήστης έχει στην κατοχή του τους συνδέσμους για τον διαμοιρασμό τους,

Ροη Γεγονότων:

1. Βασική ροή:
    1. Ο χρήστης δίνει στην εφαρμογή τα αρχεία που επιθυμεί να ανεβάσει,
    2. Το σύστημα εκκινεί την διαδικασία ανεβάσματος,
    3. Το σύστημα ανεβάζει τα αρχεία,
    4. Η υπηρεσία ενημερώνει το σύστημα ότι η διαδικασία ολοκληρώθηκε επιτυχώς,
    5. Ο χρήστης αποκτά τους συνδέσμους για τα αρχεία.
2. Εναλλακτική Ροή 1: Πριν την ολοκλήρωση του βήματος 4 ο χρήστης διέκοψε την διαδικασία,
    1. Περίπτωση <Μη-Λειτουργική ροή - Διακοπή Διαδικασίας>

2. Επιστροφή στο βήμα 1.1

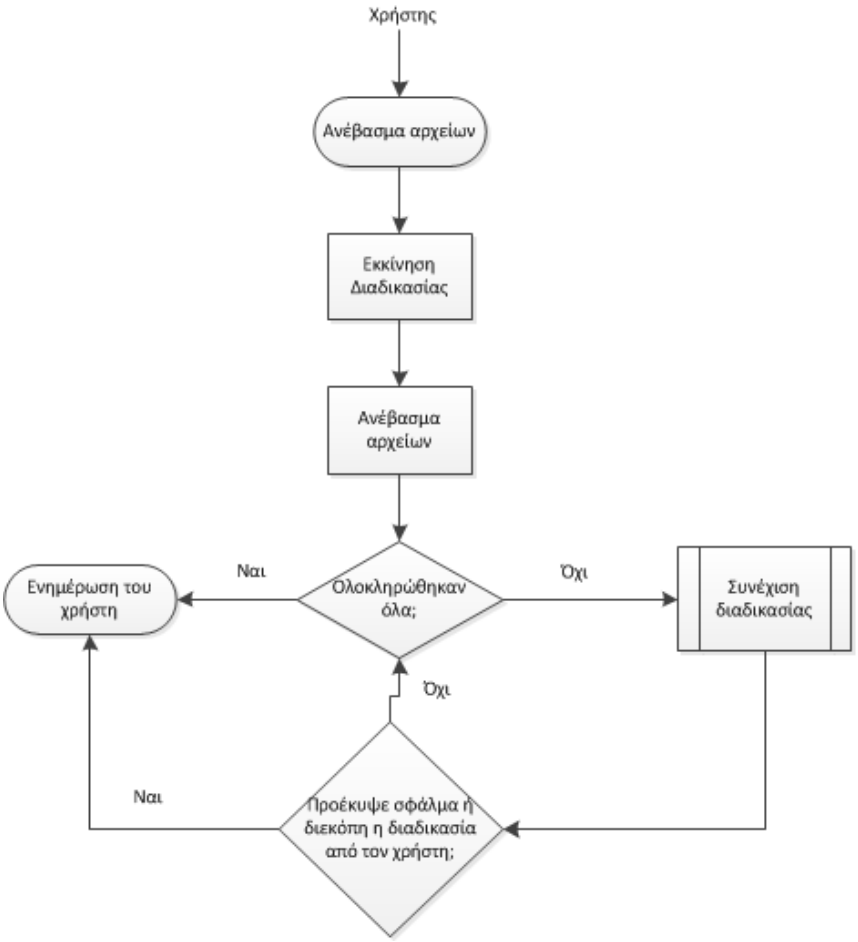
3. Ροή Σφάλματος: Πριν την ολοκλήρωση του βήματος 4 προέκυψε σφάλμα
    1. Περίπτωση <Μη-Λειτουργική ροή – Σφάλμα>
    2. Επιστροφή στο βήμα 1.1

Περιλαμβάνει: <Μη-Λειτουργική ροή - Διακοπή Διαδικασίας, Μη-Λειτουργική ροή-Σφάλμα>

Ειδικές απαιτήσεις: Καμία

Σημειώσεις: Καμία

Το διάγραμμα για την παραπάνω ροή είναι:



Εικόνα 15: Διάγραμμα ροής αποστολής αρχείου

4.2.3.2 Εξουσιοδότηση Λογαριασμού

Συντόμευση: ΔΧ-2,

Στόχος: Ανέβασμα ενός ή πολλών αρχείων σε επιλεγμένη υπηρεσία,

Πηγή: Χρήστης,

Δράστες: Χρήστης, Υπηρεσία,

Συνθήκες: Καμία,

Αποτέλεσμα: Ο χρήστης μπορεί να ανεβάσει αρχεία στην υπηρεσία,

Ροη Γεγονότων:

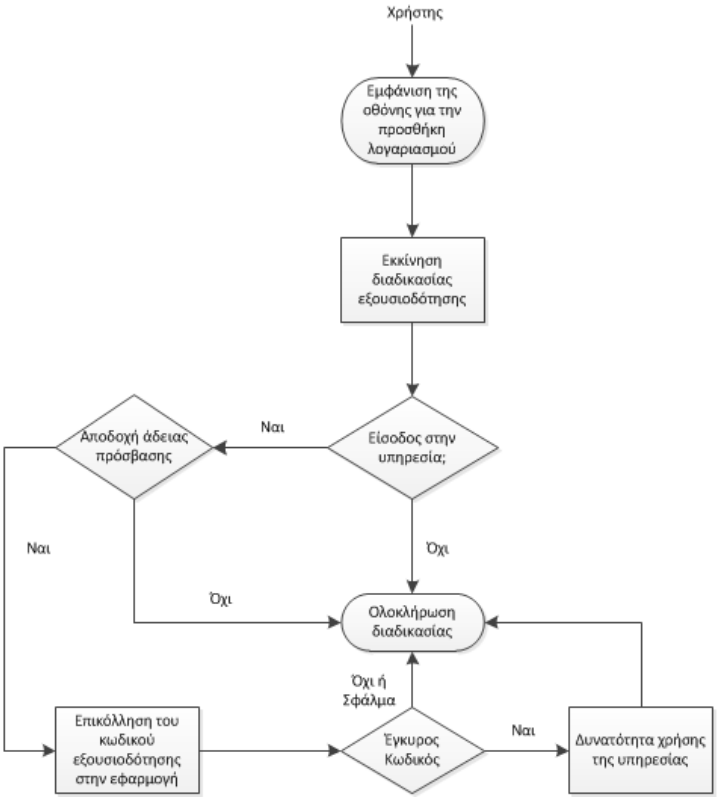
1. Βασική ροή:
    1. Ο χρήστης εμφανίζει την οθόνη προσθήκης λογαριασμού,
    2. Ο χρήστης εκκινεί την διαδικασία εξουσιοδότησης,
    3. Ο χρήστης εισαγάγει τα δεδομένα στην ιστοσελίδα της υπηρεσίας,
    4. Ο χρήστης αποδέχεται την άδεια πρόσβασης μόλις ερωτηθεί,
    5. Δίνει στην εφαρμογή τον κωδικό εξουσιοδότησης,
    6. Η Υπηρεσία απαντάει με το κλειδί εξουσιοδότησης,
    7. Ο χρήστης μπορεί να χρησιμοποιήσει την υπηρεσία,
2. Εναλλακτική Ροή 1: Στο βήμα 4 ο χρήστης αρνείται την άδεια πρόσβασης,
    1. Επιστροφή στο βήμα 1.1,
3. Εναλλακτική Ροή 1: Στο βήμα 6 η υπηρεσία απαντάει πως ο κωδικός εξουσιοδότησης είναι λανθασμένος,
    1. Ενημέρωση του χρήστη,
    2. Επιστροφή στο βήμα 1.1
4. Ροή Λάθους: Πριν την ολοκλήρωση του βήματος 6 προκύπτει σφάλμα:
    1. Προβολή κατάλληλου μηνύματος λάθους,
    2. Επιστροφή στο βήμα 1.1

Περιλαμβάνει: Τίποτα

Ειδικές απαιτήσεις: Καμία

Σημειώσεις: Καμία

Σχηματικά η ροή εξουσιοδότησης απεικονίζεται ως ακολούθως:



Εικόνα 16: Διάγραμμα ροής εξουσιοδότησης

4.3 Ποιοτικές απαιτήσεις συστήματος

Εκτός από τις λειτουργικές απαιτήσεις, πρέπει να ορίσουμε και τις ποιοτικές απαιτήσεις. Οι απαιτήσεις κινούνται σε ένα μεγάλο εύρος επιλογών που μπορεί να αφορούν τα ποιοτικά χαρακτηριστικά της εφαρμογής, την τεκμηρίωσή της, την ανοχή στα σφάλματα και πολλά ακόμη. Γενικά καθορίζουν το πώς θα υλοποιηθεί ένα χαρακτηριστικό της εφαρμογής και είναι ο κυριότερος παράγοντας για τις τεχνικές αποφάσεις στο σύστημα.

Για να γίνει κατανοητή η διαφορά ας εξετάσουμε τις λειτουργικές και ποιοτικές απαιτήσεις μέσω ενός παραδείγματος. Η προβολή εγγραφών από μια βάση δεδομένων είναι λειτουργική απαίτηση αλλά πόσες εγγραφές θα προβληθούν ή κάθε πόσο θα ανανεώνονται είναι ποιοτική απαίτηση.

Στην συνέχεια θα απαριθμήσουμε τις ποιοτικές απαιτήσεις, θα εξετάσουμε πως μεταβάλλουν τις ήδη υπάρχουσες ροές και αν είναι νέα διαδικασία θα εισάγουμε νέα ροή.

- Αποκρίσιμη εφαρμογή κατά το ανέβασμα: Μεταβάλλει την διαδικασία Λ-1, συγκεκριμένα τα βήματα 2 και 3, εισάγοντας νήματα για κάθε αρχείο που ανεβαίνει,
- Προβολή κατάστασης των αρχείων στο εργαλείο της επιφάνεια εργασίας,
- Αποστολή σχολίων στους δημιουργούς,
- Συντομεύσεις ενεργειών και έλεγχος των παραθύρων από το εικονίδιο στην γραμμή εργαλείων,
- Έλεγχος της κατάστασης των αρχείων που ανεβαίνουν με κουμπιά στο πάνω μέρος του παραθύρου λεπτομερειών,
- Αποθήκευση της κατάστασης των παραθύρων, όπως είναι η θέση και το μέγεθος,
- Ανάνηψη από σφάλματα και δυνατότητα αντιμετώπισης τους,
- Παραμετροποίηση της εφαρμογής,
- Ενημέρωση μόλις ολοκληρωθεί το ανέβασμα αρχείου,
- Αποκρίσιμη εφαρμογή κατά την διαδικασία εξουσιοδότησης καθώς ο έλεγχος της εγκυρότητας του κωδικού εξουσιοδότησης διαρκεί περίπου 2 δευτερόλεπτα,
- Υλοποίηση ανεξάρτητη του λειτουργικού συστήματος,
- Αποθήκευση της κατάστασης της εφαρμογής ανά τακτά χρονικά διαστήματα έτσι ώστε σε περίπτωση σοβαρού σφάλματος, παραδείγματος χάριν κατάρρευση, οι απώλειες να μην είναι μεγάλες,
- Αποθήκευση, προβολή και διαχείριση ιστορικού,
- Χαμηλή κατανάλωση υπολογιστικών πόρων,
- Επανεκκινήσιμα αρχεία.

Βλέπουμε πως οι ποιοτικές απαιτήσεις είναι πολύ περισσότερες σε αριθμό σε σχέση με τις λειτουργικές απαιτήσεις καθώς οι πρώτες είναι εκείνες που θα δώσουν “ταυτότητα” στην εφαρμογή και θα την κάνουν να ξεχωρίσει ανάμεσα σε άλλες εφαρμογές.

5

Σχεδίαση Συστήματος

Στο κεφάλαιο αυτό, θα περιγράψουμε τις αρχιτεκτονικές αποφάσεις που πάρθηκαν και θα εξετάσουμε αναλυτικά την δομή και την λειτουργία του συστήματος μας. Επίσης, θα παρουσιαστούν οι πληροφορίες που κρίθηκαν απαραίτητες να αποθηκευτούν έτσι ώστε να λειτουργήσει η εφαρμογή αποδοτικά και να βελτιώσουν την εμπειρία του χρήστη.

Οι έννοιες που θα μελετηθούν συνοψίζονται ως ακολούθως:

- Αρχιτεκτονικό μόρφημα Model-View-Controller(MVC),
- Μηχανή πεπερασμένων καταστάσεων που ακολουθούν τα αρχεία,
- Πέρασμα μηνυμάτων,
- Βασικές κλάσεις,
- Αλληλεπίδραση των υπομονάδων.

5.1 Εισαγωγή

Πριν προχωρήσουμε στα ενδότερα της αρχιτεκτονικής του συστήματος, θα παρουσιάσουμε και θα αιτιολογήσουμε τις αποφάσεις που πάρθηκαν για την οργάνωση του. Στο υπό ανάπτυξη σύστημα είχαμε ως στόχο οι υπομονάδες να είναι τελείως ανεξάρτητες μεταξύ τους για να επιτύχουμε τον διαχωρισμό των λειτουργιών.

5.1.1 Αρχιτεκτονικό μόρφημα MVC

Στο πνεύμα της ανεξαρτησίας των υπομονάδων, χρησιμοποιήθηκε το μόρφημα Model-View-Controller. Το ξεκίνημα του έγινε με τις γλώσσες προγραμματισμού Smalltalk-76 και Smalltalk-80 τις δεκαετίες ’70 και ’80 αντιστοίχως. Όμως σε ένα άρθρο το 1988 εξεφράσθη ως γενική έννοια και όχι ως μια υλοποίηση στις προαναφερθείσες γλώσσες προγραμματισμού. Το MVC αναπτύχθηκε για να ικανοποιήσει την ανάγκη για επαναχρησιμοποιήσιμα στοιχεία ελέγχου στα γραφικά περιβάλλοντα.

Ακολουθώντας την λογική του μορφήματος MVC μας δίνεται η δυνατότητα να ανεξαρτητοποιήσουμε το γραφικό περιβάλλον από τη λογική που απαιτείται για την λειτουργία του. Αυτό θα μας διευκολύνει στην διαχείριση, στην συντήρηση και στην επεκτασιμότητα του συστήματος.

Το MVC χωρίζεται σε δυο τομείς, της *παρουσίασης* και της *λογικής*.

Τομέας λογικής

Το πιο σύνηθες όνομα του είναι μοντέλο. Τα χαρακτηριστικά του είναι τα εξής:

- Δεν περιέχει καμία αναφορά στην παρουσίαση, δηλαδή λειτουργεί χωρίς να γνωρίζει ότι υπάρχει κάποιο παράθυρο που προβάλλει τα δεδομένα του,
- Επόμενο του προηγούμενου σημείου είναι πως μπορεί να προβάλλει τα δεδομένα σε πολλαπλές παρουσιάσεις, παραδείγματος χάριν ταυτόχρονη προβολή σε μια ιστοσελίδα, σε μια εφαρμογή κινητού και σε μια κονσόλα εντολών,
- Στέλνει μηνύματα στους ελεγκτές που έχουν δηλώσει ενδιαφέρον,
- Δεν δέχεται μηνύματα το ίδιο αλλά όποιος το χρειάζεται έχει απευθείας πρόσβαση σε αυτό.

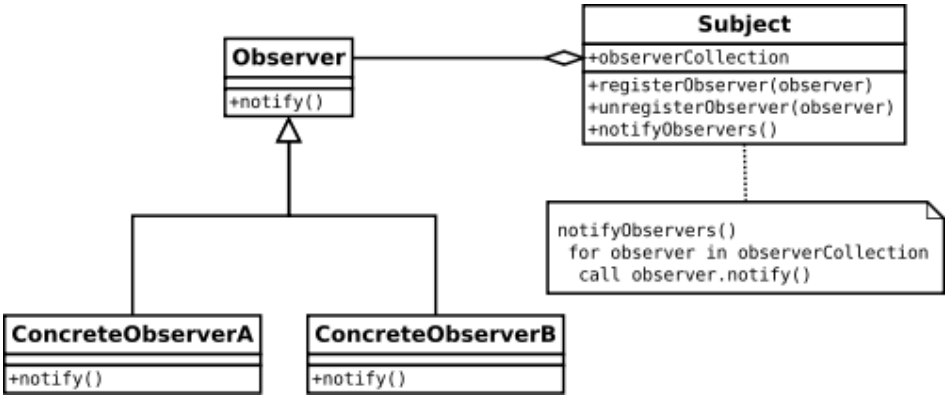
Τομέας παρουσίασης

Σε αυτόν ανήκουν ο ελεγκτής και η όψη. Κάθε στοιχείο γραφικού περιβάλλοντος που θέλουμε να έχει γενικευμένη χρήση, στο MVC υλοποιείται ως ένα ζεύγος όψης-ελεγκτή. Ο σκοπός του ελεγκτή είναι να αποφασίζει τι ενέργειες θα γίνονται για κάθε αλληλεπίδραση του χρήστη με το γραφικό περιβάλλον, στο πάτημα του κουμπιού για παράδειγμα. Η όψη αναλαμβάνει την λογική και την απεικόνιση των στοιχείων του γραφικού περιβάλλοντος τροφοδοτώντας τα με δεδομένα από το μοντέλο.

Σχεδιαστικό Μόρφημα Παρατηρητή

Είναι αναπόσπαστο κομμάτι του MVC καθώς όλοι οι μηχανισμοί του βασίζονται σε αυτό. Η λογική του είναι πως οι ενδιαφερόμενοι δηλώνουν ενδιαφέρον για κάποιο γεγονός και μόλις συμβεί, το υποκείμενο τους ενημερώνει.

Το διάγραμμα κλάσεων που περιγράφει το σχεδιαστικό μόρφημα του παρατηρητή είναι το παρακάτω:



Εικόνα 17: Σχεδιαστικό μόρφημα Παρατηρητή

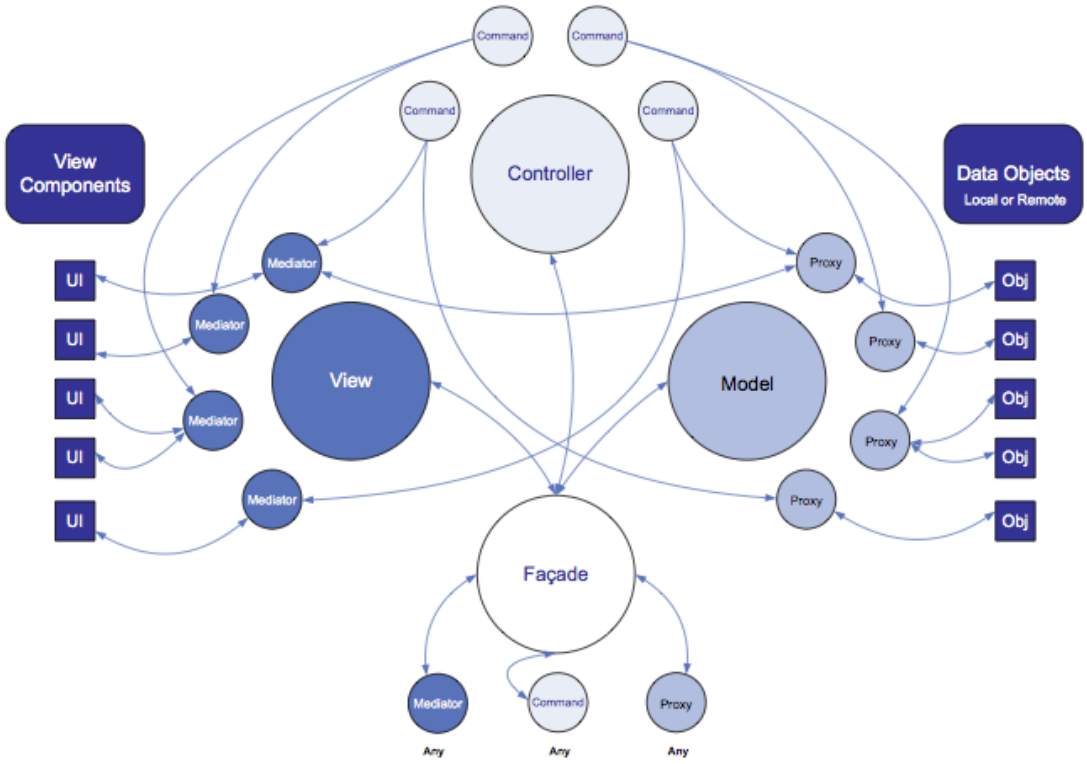
Οι ConcreteObserverA και ConcreteObserverB δηλώνουν ενδιαφέρον μπαίνοντας σε μια λίστα παρατηρητών. Ο Observer καλεί την *registerObserver* για τους δυο παρατηρητές και μόλις συμβεί το γεγονός για το οποίο ενδιαφέρονται, το Subject καλεί την συνάρτηση *notify* που τους αντιστοιχεί.

Υπάρχουν πολλές διαφοροποιήσεις σε κάθε υλοποίηση του MVC σε σχέση με το αρχικό πρότυπο και μεταξύ αυτών. Στην υλοποίησή μας τα Model, View, Controller και Façade είναι *singletons*, δηλαδή μπορεί να υπάρχει μόνο ένα αντικείμενο από την κάθε μια σε οποιαδήποτε στιγμή, που κρατάνε αναφορές προς τους διαχειριστές του εκάστοτε τμήματος του συστήματος. Πιο αναλυτικά έχουμε:

- Το Μοντέλο περιέχει αναφορές προς τους Proxies που είναι υπεύθυνοι για την διαχείριση των τοπικών ή απομακρυσμένων δεδομένων,
- Τα Commands, μέρος του Ελεγκτή, είναι οι δρομολογητές των μηνυμάτων που προέρχονται από την Όψη ή το Μοντέλο,
- Στην Όψη περιέχονται αναφορές προς τους Mediators, που φροντίζουν για την επεξεργασία της εισόδου από τον χρήστη μέσω του παραθύρου που τους έχει ανατεθεί,
- Η Πρόσοψη(Façade) έχει αποθηκευμένα όλα τα μηνύματα που μπορεί να εκπεμφθούν καθώς παρέχει μια διαπροσωπεία για την εύκολη αποστολή τους.

Η παραπάνω περιγραφή είναι για την βιβλιοθήκη *PureMVC* της Python, που μας παρείχε το πλαίσιο για την υλοποίηση. Η εν λόγω βιβλιοθήκη γράφτηκε αρχικά για την οργάνωση προγραμμάτων σε ActionScript3, για χρήση σε συνεργασία με την Flash, Adobe Flex και AIR. Με τον καιρό έγινε δημοφιλής και δημοσιεύσανε υλοποιήσεις για πάνω από 10 γλώσσες προγραμματισμού.

Το εννοιολογικό διάγραμμα της PureMVC είναι το εξής:



Εικόνα 18: Εννοιολογικό διάγραμμα της PureMVC

5.1.2 Μηχανή πεπερασμένων καταστάσεων των αρχείων

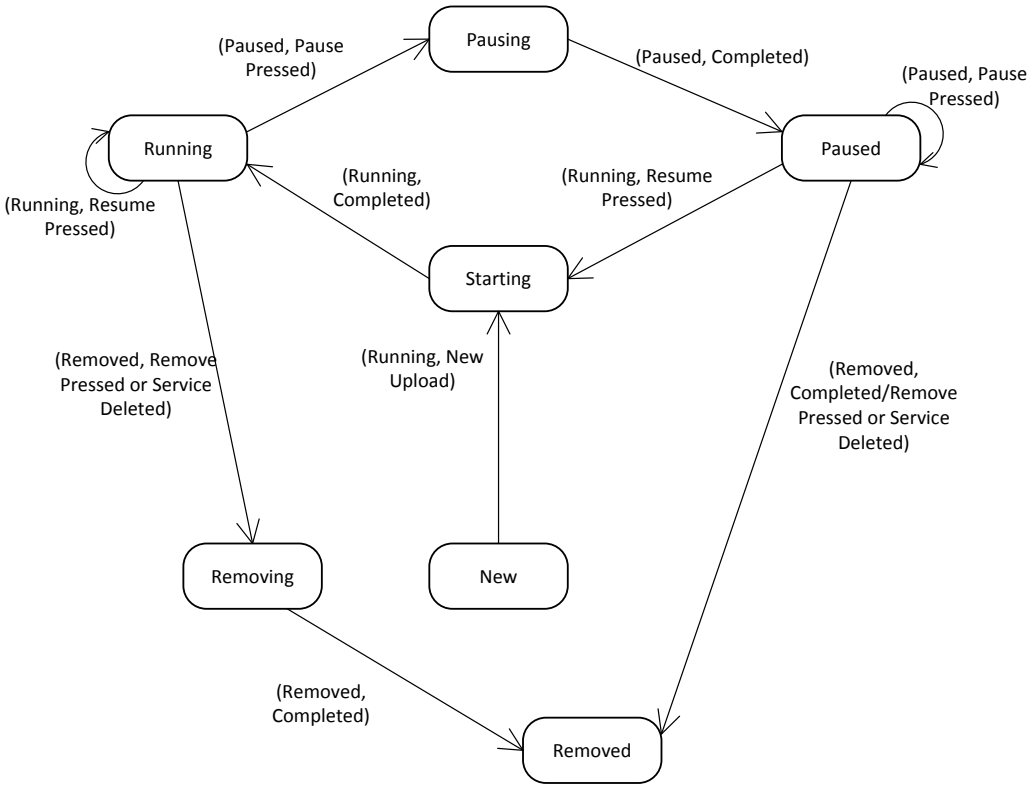
Τα ενεργά αρχεία στην εφαρμογή πρέπει να ακολουθούν κανόνες. Για τον λόγο αυτό κάθε αρχείο θα βρίσκεται σε μια από τις καταστάσεις που επιβάλλει μια μηχανή πεπερασμένων καταστάσεων.

Οι καταστάσεις είναι οι ακόλουθες:

1. *Running*: Το αρχείο αποστέλλει δεδομένα στην υπηρεσία,
2. *Paused*: Βρίσκεται σε παύση,
3. *Removed*: Η κατάσταση αυτή δεν εμφανίζεται στο γραφικό περιβάλλον αλλά αν ολοκληρωθεί το αρχείο διαγράφεται,
4. *New*: Δείχνει το σημείο εισόδου για ένα νέο αρχείο,
5. Pausing: Μεταβατική κατάσταση από την Running στην Paused,
6. Starting: Μεταβατική κατάσταση από την Paused στην Running,
7. Removing: Μεταβατική κατάσταση από την Running στην Removed.

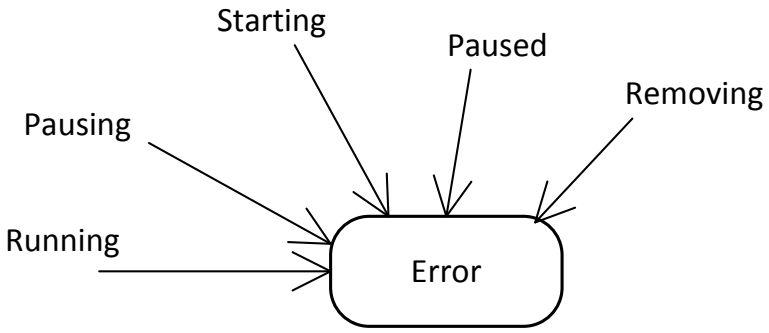
Στο διάγραμμα που ακολουθεί, παρουσιάζονται οι καταστάσεις και οι μεταβάσεις τους. Ο αναγνώστης παροτρύνεται να ανατρέξει στην υποενότητα 3 του κεφαλαίου "Τεχνικό Υπόβαθρο" στην περίπτωση που δεν θυμάται κάτι.

Διάγραμμα καταστάσεων για εκτέλεση δίχως σφάλματα:



Εικόνα 19: Μηχανή πεπερασμένων καταστάσεων αρχείων

Διάγραμμα καταστάσεων για εκτέλεση με σφάλμα:



Προέκυψε σφάλμα και το αρχείο μεταβαίνει στην κατάσταση ‘Error’ από τις καταστάσεις που φαίνονται.

Εικόνα 20: Διάγραμμα καταστάσεων για τα σφάλματα

5.2 Περιγραφή Κλάσεων

Στην ενότητα αυτή θα αναλυθούν λειτουργικά και ποιοτικά οι κλάσεις που σχηματίζουν τις μονάδες του συστήματός μας.

5.2.1 Managers

Είναι η ιεραρχία των κλάσεων που έχει υπό την ευθύνη της την διαχείριση και τον έλεγχο εγκυρότητας των τοπικών δεδομένων καθώς και την δημιουργία κλάσεων για την εύκολη επικοινωνία με τις υπηρεσίες.

Manager

Είναι η βασική κλάση που ελέγχει κατά την κατασκευή αντικειμένων αν υπάρχει ο φάκελος στον οποίο αποθηκεύονται οι αναγκαίες πληροφορίες. Επίσης περιέχει τα εξής μέλη:

- *filedir*: ο γονικός φάκελος του φακέλου που θα κρατήσει τα τοπικά αρχεία της εφαρμογής,
- *basepath:* ο φάκελος στον οποίο θα αποθηκεύσουμε τα δεδομένα,
- *services:* είναι όλες οι υπηρεσίες που υποστηρίζονται από την εφαρμογή.

LocalDataManager

Διαχειρίζεται τις πληροφορίες για την εξουσιοδότηση της εφαρμογής εκ μέρους του χρήστη, όπως είναι το κλειδί και ο απομακρυσμένος φάκελος αποθήκευσης δεδομένων. Επίσης σε κάθε κλήση των συναρτήσεών της ελέγχεται η ύπαρξη του αρχείου από το οποίο γράφει και διαβάζει. Στην περίπτωση που δεν εντοπίσει το αρχείο δημιουργείται εκ νέου με αρχικοποιημένες τιμές που έχουμε επιλέξει.

LocalUploadManager

Είναι της ίδιας λογικής με την κλάση *LocalDataManager* αλλά τα αρχεία για τα οποία είναι υπεύθυνη είναι αυτά του ιστορικού και των αρχείων των οποίων η αποστολή έχει διακοπεί.

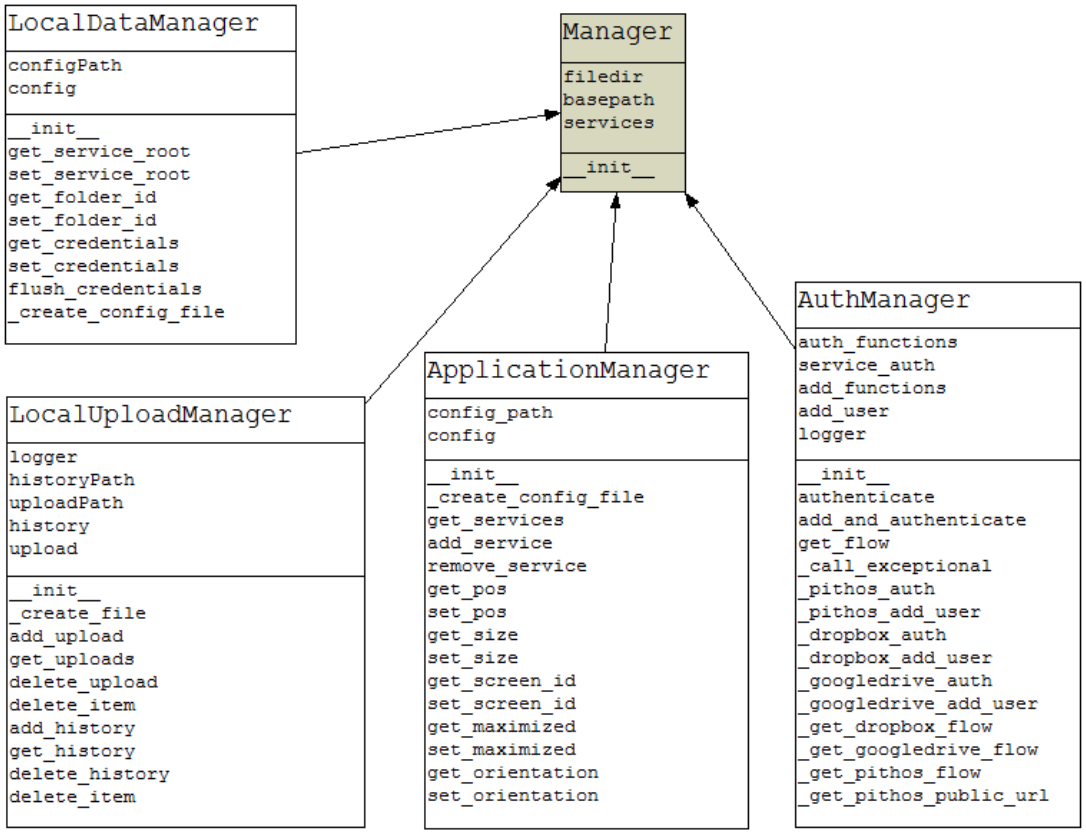
ApplicationManager

Η εφαρμογή πρέπει να αποθηκεύσει πληροφορίες για το γραφικό περιβάλλον της, όπως η θέση και το μέγεθος των παραθύρων. Η κλάση αυτή διαχειρίζεται το αρχείο που αποθηκεύει πληροφορίες όμοιας φύσης με τις προαναφερθείσες.

AuthManager

Η κλάση αυτή αναλαμβάνει την έκθεση μιας απλής διαπροσωπείας για την δημιουργία κλάσεων πελατών. Η κλάση-πελάτης είναι μέρος της βιβλιοθήκης, που παρέχεται από την ίδια την υπηρεσία συνήθως, για την εύκολη επικοινωνία με αυτές. Σε περίπτωση σφάλματος ο καλών κώδικας πρέπει να φροντίσει για την επίλυσή του.

Η ιεραρχία των *Managers* είναι το ακόλουθη:



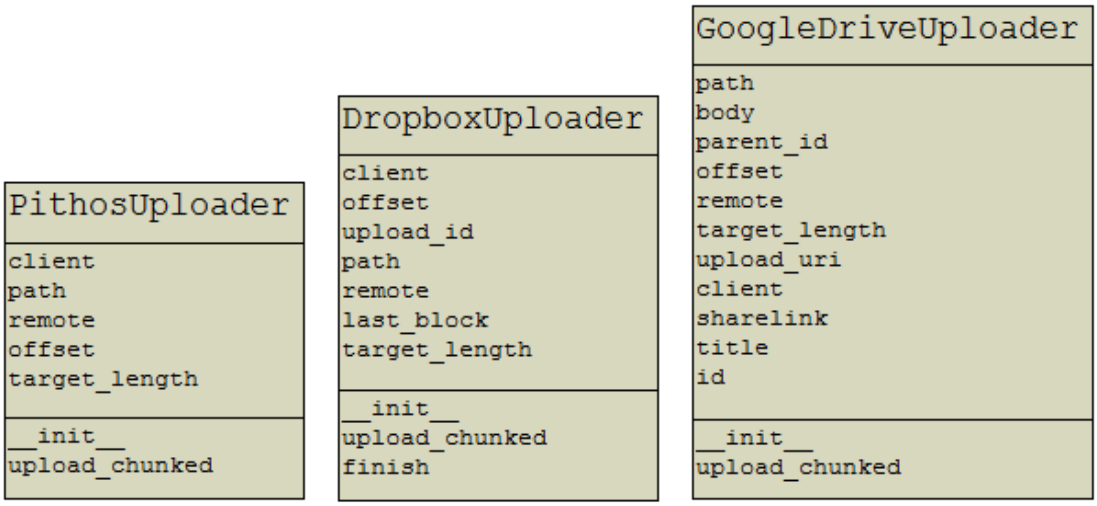
Εικόνα 21: Ιεραρχία των κλάσεων Managers

5.2.2 Uploaders

Αυτή η οικογένεια κλάσεων φροντίζει για την αποστολή δεδομένων προς τις υπηρεσίες. Διαβάζουν και στέλνουν κομμάτια από το αρχείο που τους έχει ανατεθεί μέχρι να ολοκληρωθεί όλο. Σε περίπτωση σφάλματος μεταφράζουν τα συγκεκριμένα στην υπηρεσία λάθη σε νούμερα που αντιπροσωπεύουν την κατηγορία σφάλματος στην υλοποίησή μας. Για παράδειγμα η διακοπή του δικτύου θα αντιπροσωπεύεται από το νούμερο 22.

Κάθε υπηρεσία που ενσωματώθηκε στην εφαρμογή έχει έναν Uploader που θα στέλνει το αρχείο βάσει της διαδικασίας που ορίζεται στην τεκμηρίωσή της. Η διαδικασία αποστολής δεδομένων θέλουμε να είναι ενιαία για να μην γράφουμε κώδικα συγκεκριμένα για κάθε υπηρεσία. Για αυτό οι Uploaders πρέπει να έχουν όλη την λογική σε μια συνάρτηση με ίδιο όνομα.

Η δομή των κλάσεων που περιγράφηκαν είναι η εξής:



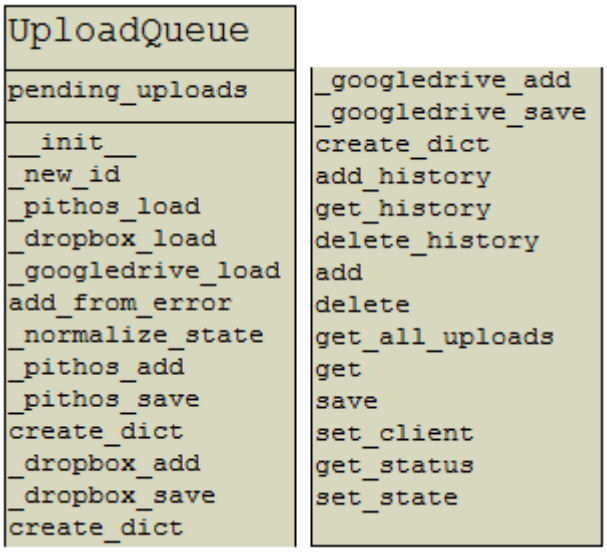
Εικόνα 22: Κλάσεις τύπου Uploader

Παρατηρούμε πως η κλάση *DropboxUploader* περιέχει την συνάρτηση *finish*. Με την ολοκλήρωση της αποστολής των δεδομένων πρέπει να κληθεί αυτή η συνάρτηση για να ορίσουμε τον απομακρυσμένο φάκελο και το όνομα του αρχείου.

5.2.3 UploadQueue

Η κλάση λειτουργεί ως μεταφραστής των τοπικών δεδομένων σε μορφή που μπορεί να την αξιοποιήσει το ModelProxy και το αντίστροφο. Το μοναδικό μέλος της κλάσης είναι ένα λεξικό που έχει αποθηκευμένους uploaders ανάλογα με την υπηρεσία μαζί με κάποια μετα-δεδομένα, συγκεκριμένα η κατάσταση του αρχείου και η πολιτική επίλυσης συγκρούσεων.

Παρατηρούμε πως στο παρακάτω διάγραμμα η συνάρτηση *create_dict* εμφανίζεται τρεις φορές γιατί είναι εμφωλευμένη στις συναρτήσεις που προηγούνται.



Εικόνα 23: Κλάση UploadQueue

5.2.4 ModelProxy

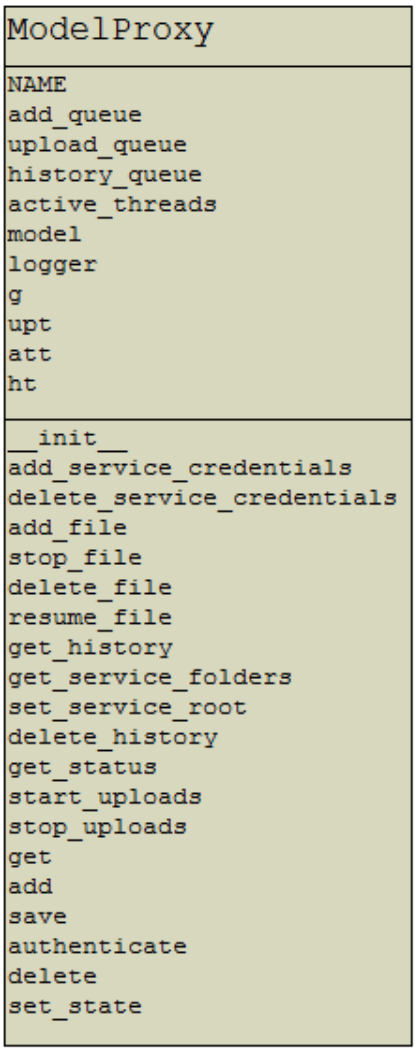
Είναι η κλάση η οποία διαχειρίζεται τον μεγαλύτερο φόρτο στην εφαρμογή μας. Οι λειτουργίες που εκτελεί είναι οι ακόλουθες:

- Έναρξη, επανεκκίνηση, διαγραφή και παύση αρχείων,
- Δημιουργία νημάτων για την αποστολή δεδομένων προς τις υπηρεσίες,
- Αποθήκευση και ανάγνωση πληροφοριών όπως το ιστορικό, οι απομακρυσμένοι φάκελοι και η κατάσταση των αρχείων,
- Εκκίνηση των τριών daemon νημάτων-διαχειριστών που ζουν όσο εκτελείται η εφαρμογή.

Τα μέλη της κλάσης που αξίζουν αναφοράς είναι τα εξής:

- *active_threads*: Ένα λεξικό που κρατάει αναφορές προς τα νήματα-εργάτες που ανεβάζουν δεδομένα με κλειδί το μοναδικό αναγνωριστικό του αρχείου,
- *model*: Ένα αντικείμενο που αναφέρεται στις κλάσεις που περιέχουν την λογική για την αποστολή αρχείων και για την δημιουργία κλάσεων-πελατών,
- *upt, att, ht*: Είναι τα daemon νήματα, τα οποία αρχικοποιούνται στον κατασκευαστή της κλάσης και δεν ξαναχρησιμοποιούνται από αυτή καθώς είναι τελείως αυτόνομα.

Οι συναρτήσεις και τα μέλη της κλάσης φαίνονται παρακάτω:



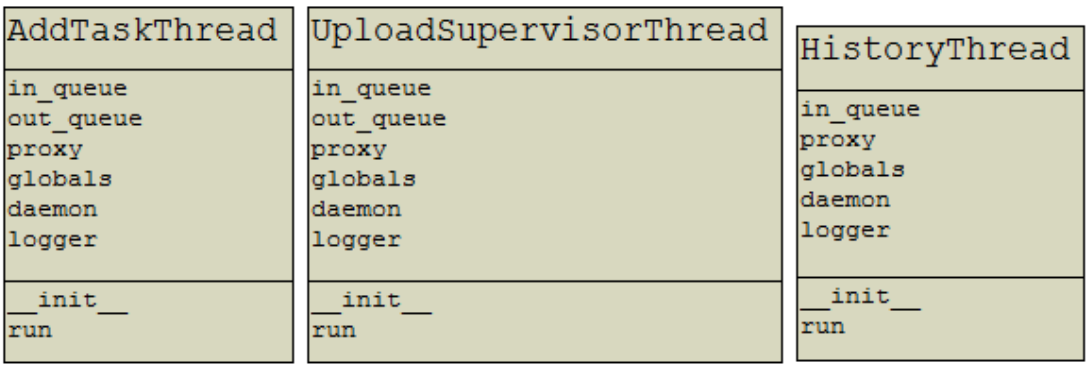
Εικόνα 24: Κλάση ModelProxy

5.2.5 *Νήματα-Διαχειριστές*

Πέρα από την χρήση νημάτων για την ταυτόχρονη αποστολή δεδομένων προς τις υπηρεσίες, οι υπόλοιπες ενέργειες επιλέξαμε να γίνονται σειριακά. Για αυτό το λόγο δημιουργήσαμε τα νήματα-διαχειριστές που διαβάζουν μηνύματα από ουρές, τα επεξεργάζονται και στη συνέχεια είτε παράγουν μήνυμα γράφοντάς το σε διαφορετική ουρά ή το στέλνουν μέσω της Πρόσοψης σε άλλες μονάδες.

Θα χρησιμοποιήσουμε τρία νήματα-διαχειριστές:

- *AddTaskThread*: Καταναλώνει μηνύματα από την ουρά του ελέγχου κατάστασης αρχείων και προετοιμάζει τους Uploaders κατασκευάζοντας για αυτούς επικυρωμένο client για την επικοινωνία με την υπηρεσία. Επίσης ενημερώνει το γραφικό περιβάλλον για την κατάσταση από την οποία ξεκινά το αρχείο. Γράφει μηνύματα στην ουρά προετοιμασίας αρχείων για ανέβασμα,
- *UploadSupervisorThread*: Διαβάζει τα μηνύματα που φτάνουν στην προηγούμενη ουρά και δημιουργεί ή καταστρέφει νήματα ανάλογα με το είδος του μηνύματος,
- *HistoryThread*: Η πρόσβαση σε κοινούς πόρους πρέπει να γίνεται σειριακά, για αυτό όταν ολοκληρωθεί η αποστολή ενός αρχείου, το νήμα γράφει στην ουρά του ιστορικού και αυτό με την σειρά του ανανεώνει το τοπικό αρχείο και ενημερώνει το γραφικό περιβάλλον για την αλλαγή.



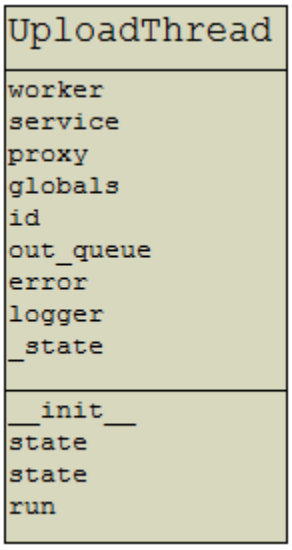
Εικόνα 25: Μακρόβια νήματα του συστήματος

5.2.6 *UploadThread*

Είναι το νήμα “εργάτης” καθώς αναλαμβάνει τον συντονισμό της διαδικασίας αποστολής δεδομένων και την ενημέρωση του γραφικού περιβάλλοντος για κάθε περίπτωση, δηλαδή ανά κομμάτι αρχείου που ολοκληρώνεται, σε σφάλμα ή στην ολοκλήρωση του διαμοιρασμού του. Αποθηκεύει αρκετές πληροφορίες που θα του χρειαστούν καθ’ όλη την ροή όπως σε η υπηρεσία που ανεβαίνει, το αναγνωριστικό του αρχείου, η ουρά ιστορικού και η κατάσταση του.

Λόγω του τρόπου με τον οποίο αποστέλλουμε δεδομένα δεν είναι δυνατόν να διακοπεί η διαδικασία ανεβάσματος κάποιου κομματιού. Για να γίνει οποιαδήποτε αλλαγή, π.χ. από Running σε Paused, πρέπει να ολοκληρωθεί η αποστολή του κομματιού.

Στο σχήμα λεπτομερειών της κλάσης βλέπουμε δυο συναρτήσεις με το όνομα *state.* Οφείλεται στο γεγονός πως η *state* είναι μέλος της κλάσης και οι δυο αυτές συναρτήσεις είναι ουσιαστικά ο *getter* και ο *setter* της με επιπρόσθετη λογική για την ασφάλειά της.



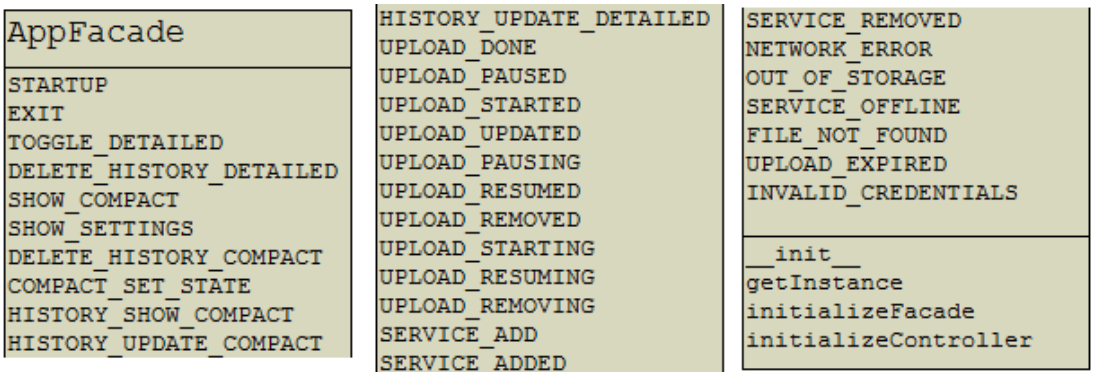
Εικόνα 26: Κλάση UploadThread

5.2.7 Logger

Αποτελεί την υποδομή που μας επιτρέπει με εύκολο τρόπο να καταγράφουμε πληροφορίες για τα γεγονότα που συμβαίνουν στην εφαρμογή. Προστατεύεται από την ταυτόχρονη πρόσβαση των νημάτων, με την έννοια πως τα μηνύματα που γράφονται δεν θα ανακατευτούν μεταξύ τους ή θα χαθούν. Τα μηνύματα διακρίνονται σε απλές ενημερώσεις, ενημερώσεις σφαλμάτων και ενημερώσεις κατά την ανάπτυξη της εφαρμογής.

5.2.8 AppFacade

Περιέχει όλα τα μηνύματα που θα χρησιμοποιηθούν απ' την εφαρμογή και διευκολύνει την επικοινωνία μεταξύ του Μοντέλου, της Όψης και του Ελεγκτή.



Εικόνα 27: Κλάση AppFacade(Πρόοψη)

5.2.9 Commands

Βρίσκονται υπό τον έλεγχο του Ελεγκτή και λειτουργούν ως διαμεσολαβητές μεταξύ της επικοινωνίας του Μοντέλου και της Όψης. Τα τελευταία μπορούν να στείλουν μηνύματα απευθείας μεταξύ τους αλλά λόγω τεχνικού θέματος προστέθηκαν τα *Commands*.

Τα *Commands* που υλοποιήθηκαν είναι τα ακόλουθα:

- *StartUpCommand*: Καλείται κατά την εκκίνηση της εφαρμογής και αρχικοποιεί τους διαμεσολαβητές του γραφικού περιβάλλοντος, το *ModelProxy*, τα παράθυρα της εφαρμογής και εκκινεί την διαδικασία αποστολής των αρχείων που διακόπηκαν,
- *HistoryCommand*: Δρομολογεί τα μηνύματα με κατεύθυνση προς τους διαμεσολαβητές του γραφικού περιβάλλοντος που αφορούν το ιστορικό, όπως η ολοκλήρωση αποστολής και η διαγραφή εγγραφής του ιστορικού,
- *UploadCommand*: Ομοίως με το *HistoryCommand* αλλά για μηνύματα με θέμα την αποστολή αρχείων, όπως η εκκίνηση, η παύση, η διαγραφή, η πρόοδος κλπ,
- *ErrorCommand*: Έχει υπό την ευθύνη του την ενημέρωση των διαμεσολαβητών για τυχόν σφάλματα που προέκυψαν στο ModelProxy,
- *ExitAppCommand:* Καλείται κατά την ομαλή έξοδο από την εφαρμογή. Αποθηκεύει ότι πληροφορία χρειάζεται στα τοπικά αρχεία.

5.2.10 Διαμεσολαβητές του γραφικού περιβάλλοντος

Η είσοδος του χρήστη στα παράθυρα της εφαρμογής που αλλάζει την κατάσταση αρχείων ή γενικά επηρεάζει το ModelProxy, διαχειρίζεται από τους διαμεσολαβητές των παραθύρων. Επίσης ενημερώνονται μέσω του Ελεγκτή για αλλαγές στο ModelProxy και ανανεώνουν τα παράθυρά τους.

Στη συνέχεια αναφέρουμε τους διαμεσολαβητές που χρειάστηκαν και ποιο είναι το παράθυρο ευθύνης τους.

- *DetailedWindowMediator*: Παράθυρο λεπτομερειών,
- *CompactWindowMediator*: Εργαλείο στην επιφάνεια εργασίας,
- SysTrayMediator: Εικονίδιο στην γραμμή εργαλείων,
- SettingsMediator: Παράθυρο ρυθμίσεων,
- HistoryWindowMediator: Αναδυόμενο παράθυρο ενημέρωσης για την ολοκλήρωση της αποστολής αρχείου.

Παρατηρούμε, όπως και στο ModelProxy, πως οι μεσολαβητές επωμίζονται μεγάλο φόρτο εργασιών.

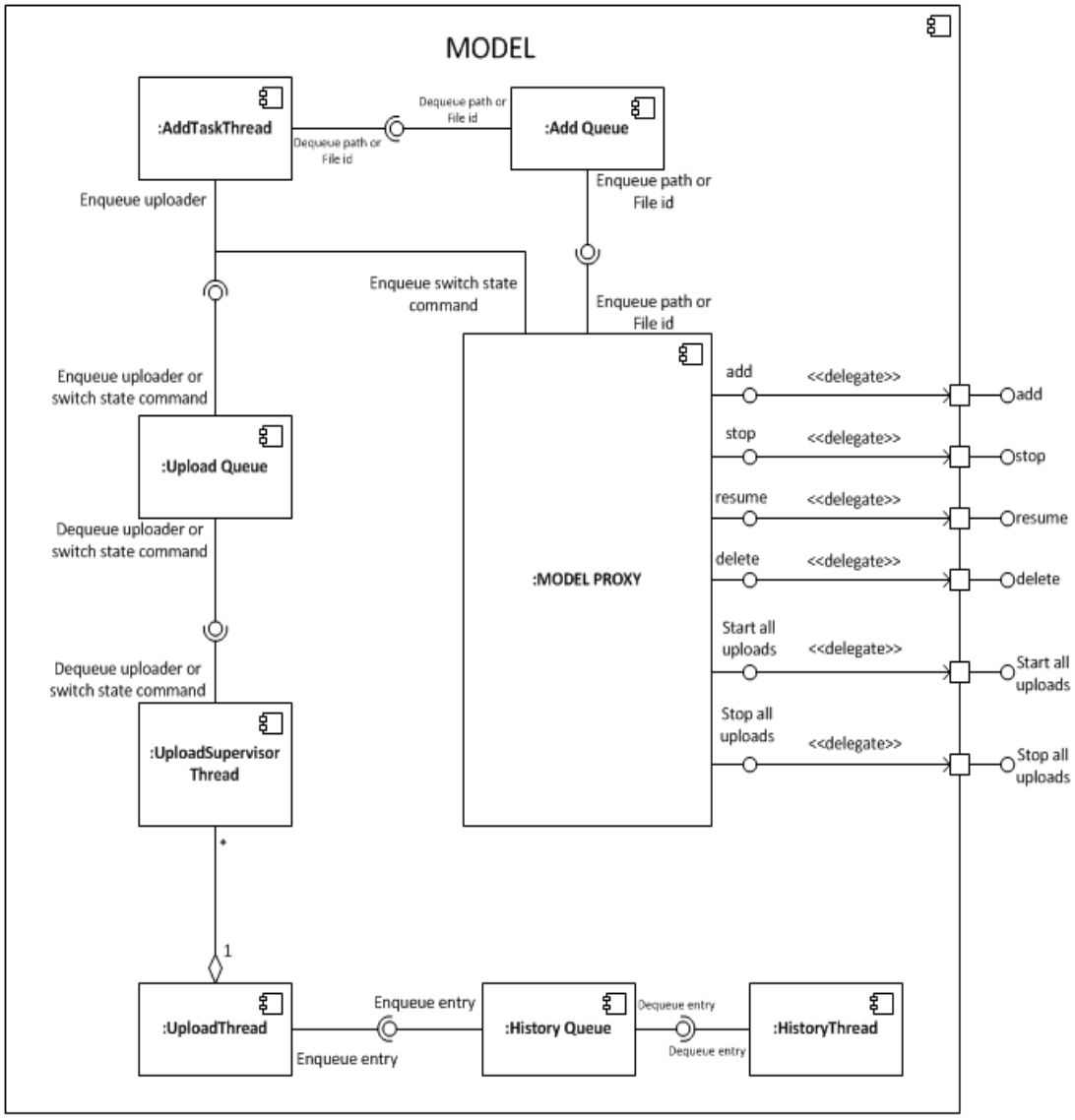
Εικόνα 28: Οι κλάσεις διαμεσολαβητών(Mediators)

5.3 Η ευρύτερη εικόνα

Έχοντας εξετάσει τα επί μέρους κομμάτια του συστήματος μας, θα μελετήσουμε εκ του μακρόθεν την αλληλεπίδραση των στοιχείων που συνιστούν τις βασικές μονάδες. Θα το πετύχουμε μέσω των διαγραμμάτων συνιστωσών(*component diagrams*) και των διαγραμμάτων ακολουθίας ενεργειών(*sequence diagrams*). Πρέπει να σημειωθεί πως δεν απεικονίζονται όλες οι λειτουργίες των μονάδων παρά μόνο οι βασικότερες εξ αυτών.

5.3.1 Μοντέλο

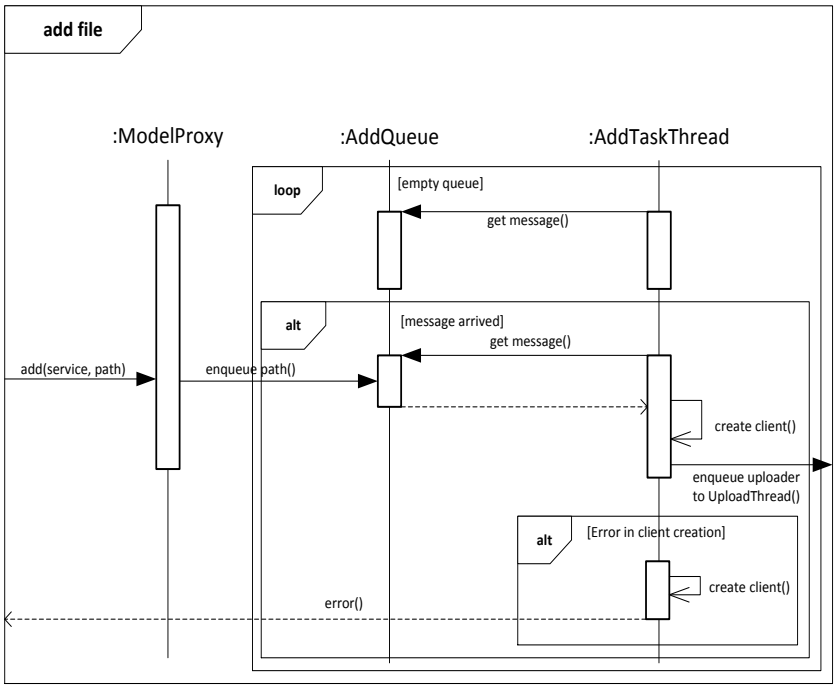
Διάγραμμα συνιστωσών



Εικόνα 29: Διάγραμμα συνιστωσών ModelProxy

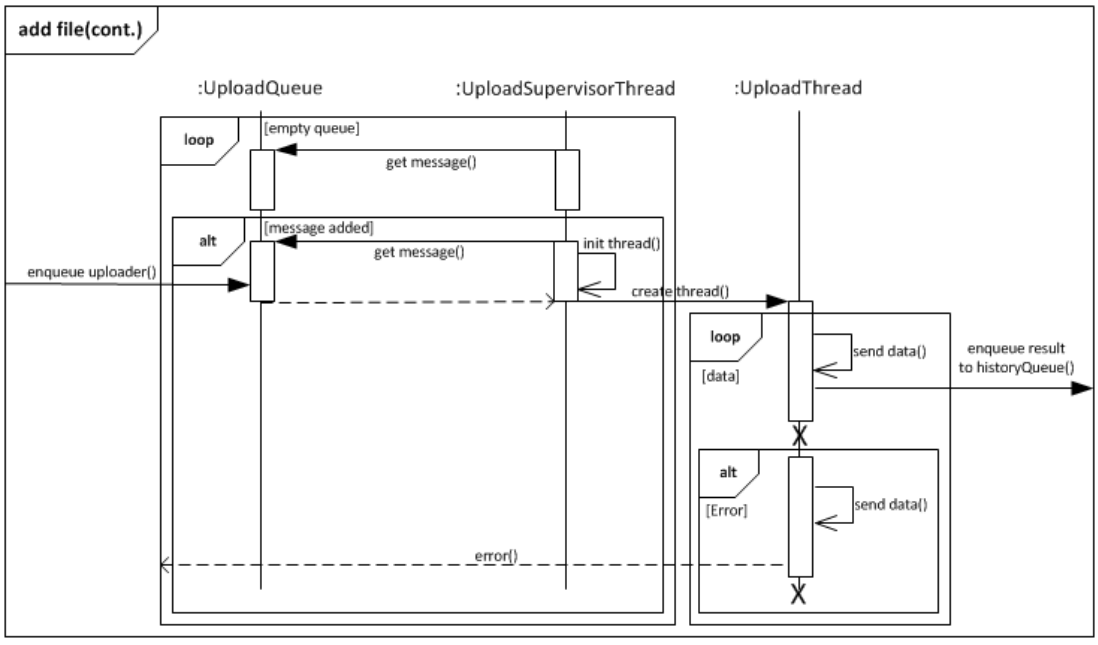
Διάγραμμα ακολουθίας ενεργειών

Η πρώτη ροή που θα δούμε είναι αυτή της προσθήκης αρχείου για ανέβασμα, δηλαδή το μοντέλο πήρε μήνυμα από το άκρο "*add*" με παράμετρο την υπηρεσία και το τοπικό μονοπάτι του αρχείου. Οι διαδικασίες "resume" και "start all uploads" είναι παρόμοιες με την "add".



Εικόνα 30: Ακολουθιακό διάγραμμα αποστολής αρχείου(1)

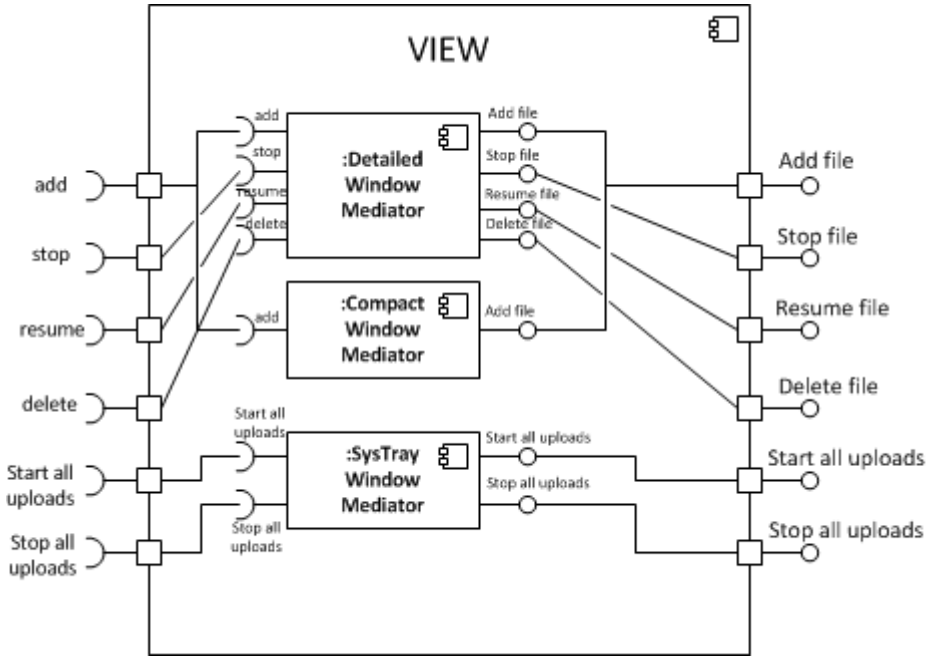
Η διαδικασία συνεχίζεται με την τριάδα *UploadQueue*, *UploadQueue* και *UploadSupervisorThread,* η οποία αποτελεί και το σημείο εισόδου για τις ροές "*stop*", "*delete*" και "*stop all uploads*".



Εικόνα 31: Ακολουθιακό διάγραμμα αποστολής αρχείου(2)

5.3.2 Όψη

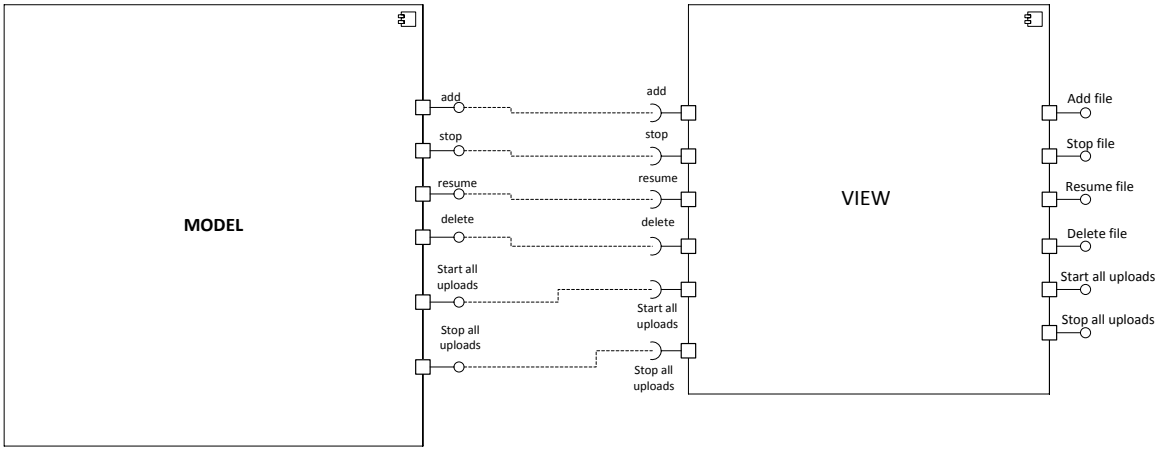
Διάγραμμα συνιστωσών



Εικόνα 32: Διάγραμμα συνιστωσών της Όψης(View)

5.3.3 Σύστημα

Με την παρουσίαση του Μοντέλου και της Όψης μπορούμε να συνδυάσουμε τα διαγράμματα συνιστωσών τους για να δημιουργήσουμε το αντίστοιχο διάγραμμα για το σύστημά μας. Στην επικοινωνία μεταξύ του Μοντέλου και της Όψης παρεμβάλλεται ο Ελεγκτής.



Εικόνα 33: Σύνδεση συνιστωσών

6

Υλοποίηση

Στις προηγούμενες ενότητες εξετάσαμε σε θεωρητικό καθώς και σε ποιοτικό επίπεδο την λειτουργικότητα και την σχεδίαση του συστήματος μας. Στο κεφάλαιο αυτό θα παρουσιαστούν σε όλο τους το εύρος οι λεπτομέρειες υλοποίησης, οι οποίες περιλαμβάνουν την ανάλυση τεχνικών προβλημάτων που αντιμετωπίσθηκαν και επιλογών που έγιναν, τις σημαντικότερες βιβλιοθήκες που χρησιμοποιήθηκαν, τις πληροφορίες που αποθηκεύονται στον σκληρό δίσκο και τις οδηγίες χρήσης της εφαρμογής.

Τέλος, θα αναλυθεί βήμα προς βήμα η διαδικασία προετοιμασίας της υπηρεσίας Pithos+ για την εκκίνηση αποστολής αρχείου και θα παρουσιαστεί ο κώδικας από τον Uploader της ίδιας υπηρεσίας.

6.1 Λεπτομέρειες υλοποίησης

Στην παρούσα υποενότητα θα παρουσιαστούν και θα αιτιολογηθούν οι τεχνικές επιλογές που έγιναν κατά την υλοποίηση της εφαρμογής.

6.1.1 Βιβλιοθήκες

Οι κυριότερες βιβλιοθήκες που χρησιμοποιήθηκαν, είτε είναι ενσωματωμένες στην Python είτε εξωτερικές, είναι οι ακόλουθες:

1. *Simpleflake – 0.1.2* :
   Το πρόβλημα για το οποίο επιλέχτηκε να επιλύσει η βιβλιοθήκη ήταν αυτό του μοναδικού αναγνωριστικού για κάθε αρχείο που αποστέλλεται. Ο χρήστης μπορεί να ανεβάσει το ίδιο αρχείο πολλές φορές, καθιστώντας τον υπολογισμό του hash του ανεπαρκές μέτρο. Η λύση στην σύγκρουση των απλών hashes είναι η επικόλληση στο τέλος του αρχείου της τρέχουσας ώρας, γεγονός που κάνει το hash μοναδικό. Όμως, η λύση αυτή δεν είναι εγγυημένα χωρίς συγκρούσεις για αυτό επιλέχτηκε η Simpleflake που παράγει αριθμούς μήκους 64bit με θεωρητική πιθανότητα σύγκρουσης $1.0787 * 10^{-9}$ σε ρυθμό δημιουργίας 100 ανά δευτερόλεπτο.

2. *Python – os* :
   Είναι κομμάτι της Python και παρέχει εργαλεία για την διαχείριση τοπικών μονοπατιών ανεξαρτήτως λειτουργικού συστήματος. Η μονάδα που χρησιμοποιήθηκε κυρίως είναι η *path* και οι βασικότερες συναρτήσεις της είναι οι εξής:
   - join(path1[,path2[,...]]): Συνδυάζει όσα μονοπάτια της περαστούν σαν παράμετροι,
   - basename(path): Επιστρέφει το όνομα του αρχείου, παραλείποντας το μονοπάτι μέχρι το αρχείο,

3. *Python – webbrowser* :
   Επιτρέπει στην εφαρμογή να ανοίγει τις ιστοσελίδες κατά την διάρκεια της διαδικασίας εξουσιοδότησης στο προεπιλεγμένο πρόγραμμα περιήγησης του υπολογιστή.

4. *PyQt4* :
   Από την βιβλιοθήκη γραφικών, η εφαρμογή χρειάστηκε τις ακόλουθες μονάδες:
   - Qt: Περιέχει μοναδικά αναγνωριστικά για κάθε λειτουργία της PyQt,
   - QtCore: Περιέχει όλη την λογική για τις μη-γραφικές λειτουργίες,
   - QtGui: Είναι η επέκταση της QtCore προσδίδοντας της λειτουργίες απαραίτητες για το γραφικό περιβάλλον.

5. *ConfigObj* :
   Αναφέρθηκε στην υποενότητα 3.6 και χρησιμοποιήθηκε για την προγραμματιστική διευκόλυνσή στην διαχείριση των δεδομένων που αποθηκεύονται στον υπολογιστή.

6.1.2 Φιλτράρισμα καταστάσεων κατά την αποθήκευση

Αν κατά τον τερματισμό της εφαρμογής κάποιο αρχείο βρίσκεται σε μεταβατική κατάσταση, δηλαδή σε μια από τις 'Starting', 'Resuming', 'Pausing' ή 'Removing', τότε το αρχείο θα αποθηκευτεί στην τερματική κατάσταση που θα μετέβαινε εάν προλάβαινε να ολοκληρωθεί.

Συγκεκριμένα οι δυο πρώτες από τις προαναφερθείσες έχουν ως στόχο την 'Running' αλλά όταν ξαναεκτελεστεί η εφαρμογή πρέπει να δημιουργηθούν clients γεγονός που μας οδηγεί στο να αποθηκεύσουμε τα αρχεία σε κατάσταση 'Starting'. Στην ίδια φιλοσοφία αν είναι στην 'Pausing' τότε θα αποθηκευτεί ως 'Paused'. Ολοκληρώνοντας, εάν διακοπεί η εφαρμογή κατά την διαγραφή της αποστολής, δηλαδή στην 'Removing', τότε οι πληροφορίες του αρχείου δεν θα αποθηκευτούν καθόλου.

6.1.3 Νήματα ή διεργασίες

Μια από τις πιο σημαντικές επιλογές που έγιναν ήταν ο τρόπος με τον οποίο η εφαρμογή μας θα παραμένει αποκρίσιμη σε κάθε περίπτωση. Τα κριτήρια που πρέπει να ικανοποιεί ο ζητούμενος μηχανισμός είναι η δυνατότητα πρόσβασης σε κοινούς πόρους και η μικρότερη προγραμματιστική επιβάρυνση.

Οι πιθανοί μηχανισμοί ήταν το νήμα ή η διεργασία. Τα νήματα, σε αντίθεση με τις διεργασίες, βλέπουν την ίδια μνήμη άρα ο διαμοιρασμός δεδομένων μεταξύ τους γίνεται απρόσκοπτα. Το πρόβλημα που προκύπτει από αυτό, που δεν έχουν οι διεργασίες, είναι πως πρέπει να ενσωματώσουμε μηχανισμούς για την πρόσβαση και επεξεργασία των δεδομένων που είναι κοινά.

Είναι ξεκάθαρο πως δεν υπάρχει μηχανισμός που ικανοποιεί και τα δυο ζητούμενα κριτήρια αλλά είναι απαραίτητη η κοινή χρήση των δεδομένων άρα εν τέλει επιλέχτηκε η χρήση των νημάτων.

Το δεύτερο ζήτημα είναι η δομή δεδομένων που θα φιλοξενήσει τα νήματα. Το δίλημμα ήταν ανάμεσα στο λεξικό(dictionary) της Python, που στην ουσία είναι ένα hash table, και σε ένα pool από νήματα, που είναι μια δομή με προκαθορισμένο αριθμό διαθέσιμων νημάτων και με κάθε αίτηση η εφαρμογή δεσμεύει ένα από αυτά μέχρι να εξαντληθούν. Οι προϋποθέσεις ήταν, όπως και προηγουμένως, η εύκολη προγραμματιστική διαχείριση της δομής και άμεση αναφορά στα νήματα.

Το hash table με κλειδί το αναγνωριστικό του κάθε αρχείου αποδείχθηκε η καλύτερη επιλογή εκ των δυο καθώς στο pool νημάτων δεν μπορούμε να αναφερθούμε καθόλου σε κάθε νήμα ξεχωριστά. Το αρνητικό με το hash table είναι πως χρειάζεται παραπάνω προγραμματιστική προσπάθεια για την ομαλή και σωστή λειτουργία.

6.1.4 Τοποθέτηση αναδυόμενου παραθύρου ιστορικού

Ένα πολύ σημαντικό ζήτημα που προέκυψε, όσον αφορά το γραφικό περιβάλλον, ήταν η τοποθέτηση του αναδυόμενου παραθύρου κατά την ολοκλήρωση της αποστολής κάποιου αρχείου. Μπορεί να είναι πολύ μικρή λεπτομέρεια που λογικά θα αγνοηθεί αλλά εάν δεν προβλέψουμε για αυτό τότε ίσως να ενοχλήσει κάποιους χρήστες της εφαρμογής και να χαλάσει την εμπειρία χρήσης τους.

Με την λύση που υλοποιήθηκε η τοποθέτηση του παραθύρου γίνεται αυτόματα αναλόγως της θέσης της γραμμής εργασιών(taskbar) του λειτουργικού. Η PyQt4 έχει την δυνατότητα παροχής πληροφοριών σχετικά με την οθόνη ή τις οθόνες του χρήστη. Οι πληροφορίες που χρειάστηκαν για την αυτόματη τοποθέτηση του παραθύρου ήταν οι εξής:

- Διαθέσιμος χώρος: Είναι οι διαστάσεις της οθόνης οι οποίες μπορούν να αξιοποιηθούν από εφαρμογές. Για παράδειγμα αν η οθόνη του χρήστη έχει μέγεθος 1600x900 pixels τότε, θεωρώντας πως η γραμμή εργασιών είναι ορατή και ότι βρίσκεται στο κάτω μέρος της οθόνης, ο αξιοποιήσιμος χώρος είναι 1520x900 με αρχή το (0, 0). Η συνάρτηση της PyQt που μας δίνει αυτή τη πληροφορία είναι η:
  QtGui.QApplication.desktop().availableGeometry()
- Συνολικός χώρος: Είναι η διαστάσεις της οθόνης, π.χ. 1600x900 με θέση (0, 0), συμπεριλαμβανομένης της γραμμής εργαλείων. Η συνάρτηση που μας δίνει αυτή την πληροφορία είναι η:
  QtGui.QApplication.desktop().screenGeometry()

Δομούμε τις παραπάνω μετρικές στην εξής μορφή:

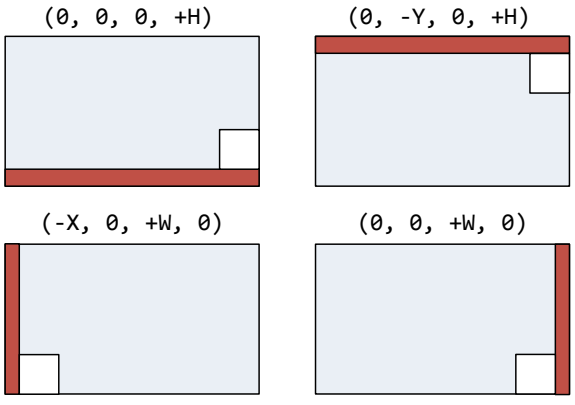
$$(\theta\acute{\epsilon}\sigma\eta X, \theta\acute{\epsilon}\sigma\eta Y, \pi\lambda\acute{\alpha}\tau\sigma\varsigma, \acute{\upsilon}\psi\sigma\varsigma)$$

Βάσει παρατήρησης του αποτελέσματος της αφαίρεσης του διαθέσιμου χώρου από τον συνολικό χώρο((X, Y, W, H)), προσπαθούμε να εξάγουμε την μορφή που πρέπει να έχει η παραπάνω δομή για την αναγνώριση κάθε θέσης στην γραμμή εργαλείων.

Το μοτίβο είναι το ακόλουθο(X, Y, W, H είναι απόλυτες τιμές):

- Κάτω: (0, 0, 0, +H), τοποθετείται στο δεξί άκρο,
- Δεξιά: (0, 0, +W, 0), τοποθετείται στο κάτω άκρο,
- Πάνω: (0, -Y, 0, +H), τοποθετείται στο δεξί άκρο,
- Αριστερά: (-X, 0, +W, 0), τοποθετείται στο κάτω άκρο.

Σχηματικά έχουμε:



Εικόνα 34: Πιθανές θέσεις της γραμμής εργαλείων

6.1.5 Διαδικασία προετοιμασίας πελατών για το Pithos+

Λόγω της συνεχούς ανάπτυξης της πλατφόρμας ~okeanos δεν υπάρχουν ακόμα οδηγίες βήμα προς βήμα για τους προγραμματιστές που θέλουν να αλληλεπιδράσουν με τον Pithos+. Για τον λόγο αυτό θα αναλυθεί η σειρά των εντολών που χρειάζονται για την δημιουργία μιας κλάσης πελάτη έτοιμη για επικοινωνία με την υπηρεσία.

1. Φόρτωση(import) της κλάσης AstakosClient από την βιβλιοθήκη astakosclient,
2. Φόρτωση της κλάσης PithosClient από την υπομονάδα kamaki.clients.Pithos,
3. Αρχικοποίηση των ακολούθων παραμέτρων:
    a. url: *https://accounts.okeanos.grnet.gr/identity/v2.0*,
    b. token: το κλειδί επικοινωνίας με την υπηρεσία ~okeanos μέσω του rest API, το οποίο μπορεί να το αποκτήσει κανείς από τον παρακάτω σύνδεσμο: *https://accounts.okeanos.grnet.gr/ui/api_access*
4. Δημιουργία της μεταβλητής s που είναι τύπου AstakosClient: $s = AstakosClient(token, url)$,
5. Κλήση της συνάρτησης *authenticate* για την απόκτηση συνδέσμων για διάφορες υπηρεσίες που υπάγονται στην ~okeanos: *auth_data = s.authenticate()*. Η endpoints είναι της μορφής json/dictionary και για την απόκτηση του τελικού συνδέσμου πρέπει να κάνουμε το εξής:
    a. $catalog = auth\_data['access']['serviceCatalog']$
    b. Αναζήτηση για τον τύπο 'object-store', δηλαδή τον Pithos+:

```
for entry in catalog:
    if entry['type'] == 'object-store':
        pithosurl = entry['endpoints'][0]['publicURL']
        break
```

6. Η επόμενη πληροφορία που χρειαζόμαστε είναι το *uuid*, το μοναδικό αναγνωριστικό του χρήστη στον ~okeanos:

```
uuid = auth_data['access']['user']['id']
```

7. Τώρα έχουμε όλες τις πληροφορίες για να δημιουργήσουμε την κλάση πελάτη για τον Pithos+:

```
p = PithosClient(pithosurl, token, uuid)
```

Η διαφοροποίηση με την διαδικασία που υλοποιήθηκε στην εφαρμογή έγκειται στο γεγονός πως δεν χρησιμοποιήθηκε η κλάση *PithosClient* αλλά μια παραλλαγή της μέσω κληρονομικότητας.

6.1.6 Αποθηκευμένες πληροφορίες

Οι πληροφορίες που πρέπει αποθηκευτούν χωρίζονται σε τέσσερα τμήματα:

6.1.6.1 Επανεκκίνηση αρχείων

Κάθε υπηρεσία χρειάζεται διαφορετικά στοιχεία για την επανεκκίνηση των αρχείων που ανεβαίνουν σε αυτές. Τα κοινά στοιχεία είναι η κατάσταση, ο απομακρυσμένος φάκελος, η συμπεριφορά στην περίπτωση σύγκρουσης ονόματος και το τοπικό μονοπάτι.

Οι ανά υπηρεσία πληροφορίες είναι:

- *Dropbox*
  Upload-id: Το αναγνωριστικό του αρχείου στην υπηρεσία. Με αυτό θα συνεχιστεί το ανέβασμα και έχει χρόνο ζωής μια μέρα,
  Offset: Ο αριθμός των bytes που απεστάλησαν πριν την διακοπή,
- *GoogleDrive*
  Parent-id: Το αναγνωριστικό του φακέλου που θα αποθηκευτεί το αρχείο,
  Offset: Ο αριθμός των bytes που απεστάλησαν πριν την διακοπή,
  Upload-uri: Το αναγνωριστικό του αρχείου στην υπηρεσία. Με αυτό θα συνεχιστεί το ανέβασμα και έχει χρόνο ζωής μια μέρα,
- *Pithos+*
  Offset: Ο αριθμός των bytes που απεστάλησαν πριν την διακοπή.
  Η μόνη πληροφορία που χρειάζεται για να την επανεκκίνηση είναι το τοπικό όνομα του αρχείου και το όνομα του απομακρυσμένου φακέλου. Το offset αποθηκεύεται για την απεικόνιση στο γραφικό περιβάλλον.

Παράδειγμα αποθηκευμένων δεδομένων για το Dropbox και το Pithos+:

```
[Dropbox]
[[36747746358543850 11]]
status = Paused
destination = /
upload_id = gKEbXAan5QO_q_suOPUiaw
offset = 131072
path = C:/Users/MyUsername/Desktop/db.pdf
conflict = KeepBoth

[Pithos]
[[36747902288579514 55]]
status = Paused
path = C:/Users/MyUsername/Music/music.mp3
destination = cloudy
conflict = KeepBoth
offset = 4194304
```

6.1.6.2 *Ιστορικό Αρχείων*

Για κάθε εγγραφή στο ιστορικό αποθηκεύεται η ημερομηνία και ώρα ολοκλήρωσης, το απομακρυσμένο μονοπάτι και η υπηρεσία.

Παράδειγμα από δυο εγγραφές αρχείων που ανεβήκανε στο Pithos+:

```
[Pithos]
[[3674790228857951455]]
date = YYYY-MM-DD HH:MM:SS
path = pithos
link = https://pithos.okeanos.grnet.gr/public/B94iEYclymI3CiBbkTcPq8
name = mymusic1.mp3
[[3674802119461354286]]
date = YYYY-MM-DD HH:MM:SS
path = cloudy
link = https://pithos.okeanos.grnet.gr/public/gbrmcEmeG9E3DPupPDbP
name = mymusic2.m4a
```

6.1.6.3 *Κλειδιά επικοινωνίας με τις υπηρεσίες*

Περιέχονται τα κλειδιά για την επικοινωνία και ο τρέχων απομακρυσμένος φάκελος. Το σύστημα πρέπει να έχει προνοήσει για την προστασία αυτών των δεδομένων καθώς είναι ευαίσθητης φύσεως.

Παράδειγμα πληροφοριών για το Dropbox και το Pithos+:

```
[Pithos]
ROOT = cloudy
credentials = myPithosToken
[Dropbox]
ROOT = /
credentials= myDropboxToken
```

6.1.6.4 *Πληροφορίες του γραφικού περιβάλλοντος*

Εδώ αποθηκεύονται πληροφορίες όπως οι χρησιμοποιούμενες υπηρεσίες, οι θέσεις και τα μεγέθη των παραθύρων και οι ρυθμίσεις της εφαρμογής.

Απόσπασμα πληροφοριών για την εφαρμογή που αναπτύχθηκε:

```
Services = Dropbox, Pithos
[Detailed]
pos = 825, 58
size = 720, 454
maximized = False
screen_id = 0
```

6.1.7 Ο ρόλος του Ελεγκτή

Στο σύστημά που υλοποιήθηκε ο Ελεγκτής έχει υποστηρικτικό ρόλο και αυτός είναι να επιτευχθεί η επικοινωνία ανάμεσα στο Μοντέλο και την Όψη. Το Μοντέλο είναι πολυνηματικό υποσύστημα ενώ το γραφικό περιβάλλον εκτελείται κυρίως στο κύριο νήμα.

Η PyQt4 δεν επιτρέπει αλλαγές στην κατάσταση των παραθύρων της από νήματα πέραν του βασικού, γεγονός που δεν επιτρέπει την άμεση αποστολή μηνυμάτων από το Μοντέλο προς την Όψη.

Για τον λόγο αυτό ο Ελεγκτής λαμβάνει αυτά τα μηνύματα και χρησιμοποιώντας έναν από τους βασικότερους μηχανισμούς της PyQt4, τα σήματα, τα μεταφέρει στην Όψη. Η διαδικασία για να το επιτύχουμε αυτό είναι η ακόλουθη, όπως υλοποιήθηκε στην εφαρμογή:

1. Ορίζουμε σε μια καθολική κλάση, έστω Globals, από την οποία μπορεί να υπάρχει μόνο ένα στιγμιότυπο, δηλαδή είναι singleton, για παράδειγμα:

```
class Signals(QtCore.QObject):
    upload_detailed_start = QtCore.pyqtSignal(list)
class Globals(object):
    def __init__(self):
        self.signals = Signals()
def get_globals(_singleton=Globals()):
    return _singleton
```

2. Στην κλάση αυτή περιλαμβάνουμε τα σήματα που μας ενδιαφέρουν με ένα προς ένα αντιστοιχία με τα μηνύματα που στέλνει το Μοντέλο,
3. Κατά την κατασκευή των διαμεσολαβητών δηλώνονται, μέσω της συνάρτησης connect των σημάτων στην Globals, οι συναρτήσεις που θα κληθούν όταν σταλεί από τον ελεγκτή το αντίστοιχο σήμα, όπως φαίνεται παρακάτω:

```
(στην συνάρτηση __init__)
    get_globals().signals.upload_detailed_start.connect(self.onUploadStart)
(στο σώμα της κλάσης, εκτός της __init__)
    def onUploadStart(self, id):
        print ‘Upload started with id’, id
```

4. Όταν συμβεί το ζητούμενο γεγονός και το Μοντέλο στείλει το μήνυμα, ο Ελεγκτής θα καλέσει την συνάρτηση *emit* του κατάλληλου σήματος, δηλαδή:

```
get_globals().signals.upload_detailed_start.emit(upload_id)
```

5. Μόλις εκπεμφθεί το παραπάνω σήμα, ο διαμεσολαβητής το λαμβάνει και καλεί την *onUploadStart*.

6.1.8 Κώδικας του PithosUploader

Εφόσον παρουσιάστηκε η θεωρητική διαδικασία αποστολής δεδομένων, στην υποενότητα αυτή θα εξεταστεί η υλοποίησή ενός από τους Uploaders στο επίπεδο του κώδικα.

Το πρώτο ενδιαφέρον κομμάτι της κλάσης PithosUploader είναι ο κατασκευαστής, στον οποίο αρχικοποιούνται οι μεταβλητές που θα χρειαστούν στην συνάρτηση που αποστέλλει δεδομένα, δηλαδή:

```
class PithosUploader(object):
    def __init__(self, path, remote='pithos', offset=0, client=None):
        self.client = client
        self.path = path
        self.remote = remote #Container
        self.offset = offset #Progress
        self.target_length = os.path.getsize(path)
```

Οι μεταβλητές που απαιτείται να αρχικοποιηθούν είναι το τοπικό μονοπάτι του αρχείου, το απομακρυσμένο μονοπάτι, που στην περίπτωση του Pithos+ είναι ο φάκελος, το μέγεθος του αρχείου και τα bytes που έχουν σταλεί μέχρι εκείνη την στιγμή ενώ στην περίπτωση που δεν είναι 0 σημαίνει πως για κάποιο λόγο είχε διακοπεί.

Η δεύτερη και τελευταία συνάρτηση είναι η *upload_chunked* η οποία δέχεται ως παράμετρο το μέγεθος των κομματιών αλλά στην περίπτωση της υπηρεσίας Pithos+, κατά την συγγραφή της εργασίας, δεν ήταν δυνατόν να μεταβληθεί το μέγεθος πέρα από αυτό των 4 MB. Ο κώδικας της συνάρτησης έχει ως ακολούθως:

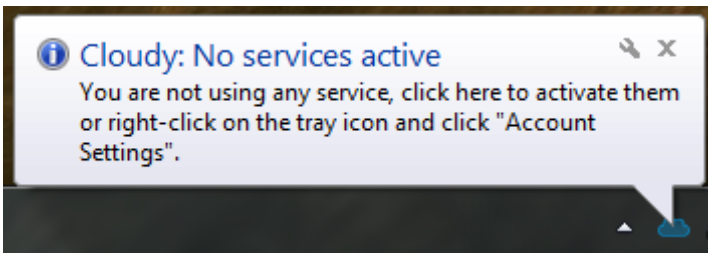
```
def upload_chunked(self, chunk_size=128*1024): #Chunk size unused.
    try:
        with open(self.path, 'rb') as f:
            try:
                for i in self.client.upload_object(os.path.basename(self.path), f,
public=True):
                    self.offset += i
                    try:
                        yield (float(self.offset)/self.target_length, self.path)
                    except ZeroDivisionError:
                        #The file was empty, it's 100% by default.
                        yield (1.0, self.path)
            except ClientError as e:
                    if e.status == 413: #Out of quota
                        yield (3, None)
                    elif e.status in [401, 404]:
                        yield (12, None)
                    elif 'Errno 11004' in e.message:
                        yield (22, None)
                    return
    except IOError as e:
        yield (2, None)
```

Όπως φαίνεται το πρώτο βήμα είναι να ανοίξει το αρχείο για ανάγνωση και στην περίπτωση που δεν υπάρχει επιστρέφεται σφάλμα τύπου #2. Αν το άνοιγμα του αρχείου γίνει με επιτυχία καλείται η *upload_object* της κλάσης πελάτη δίνοντας το όνομα του αρχείου και τον δείκτη για την ανάγνωση του καθώς και μια παράμετρο True που συμβολίζει ότι το αρχείο μόλις ολοκληρωθεί θέλουμε να είναι δημόσιο. Στην συνέχεια, ξεκινάει η επανάληψη για την αποστολή των κομματιών και με κάθε επιτυχημένη αποστολή κάποιου κομματιού ανανεώνεται η πρόοδος και επιστρέφεται στο καλών νήμα για να ενημερώσει με την σειρά του το γραφικό περιβάλλον. Τέλος, καθ' όλη την διαδικασία γίνεται έλεγχος για σφάλματα εξουσιοδότησης και δικτύου.

6.2 *Εγχειρίδιο Χρήσης*

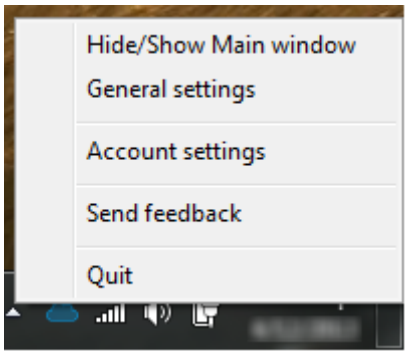
Κάθε εφαρμογή χρειάζεται κάποιο χρόνο εξοικείωσης από τον χρήστη μέχρι να ενσωματωθεί στη καθημερινότητά του. Ο ελάχιστος χρόνος που απαιτείται για την πλήρη εξοικείωση με μια εφαρμογή μπορεί να μειωθεί αισθητά αν υπάρχει ένα εγχειρίδιο χρήσης για αυτή.

Κατά την πρώτη εκτέλεση της εφαρμογής ο χρήστης δεν θα την έχει εξουσιοδοτήσει για ανεβάζει σε κάποια υπηρεσία. Για τον λόγο αυτό εμφανίζεται ένα μήνυμα κατά την εκκίνηση το οποίο τον ενημερώνει για την απουσία υπηρεσιών και με το πάτημα του αριστερού κουμπιού του ποντικιού πάνω σε αυτό εμφανίζεται η οθόνη προσθήκης και διαχείρισης υπηρεσιών όπως φαίνεται παρακάτω:



Εικόνα 35: Απουσία χρηστών κατά την εκκίνηση

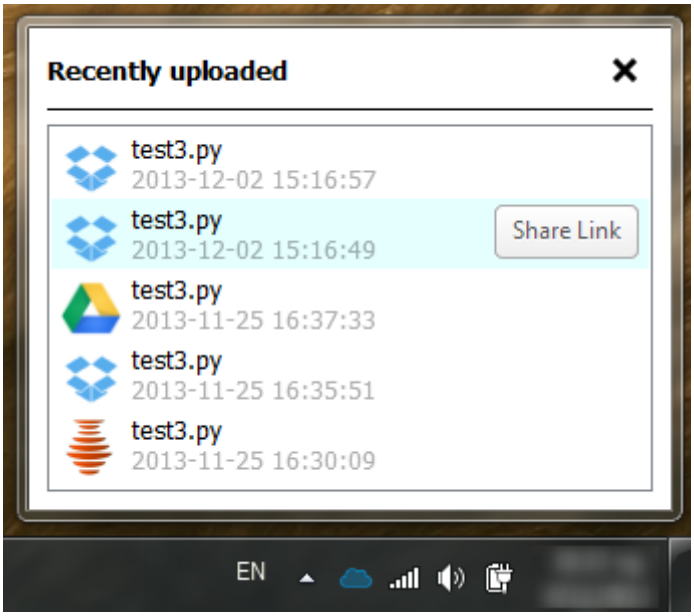
Συνεχίζοντας στο εικονίδιο της γραμμής εργαλείων, ο χρήστης μπορεί να πατήσει με το δεξί κουμπί το ποντικιού πάνω του για να εμφανίσει το ακόλουθο μενού, που λειτουργεί ως είσοδο για τα υπόλοιπα μέρη της εφαρμογής:



Εικόνα 36: Μενού του εικονιδίου της γραμμής εργαλείων

- *Hide/Show Main window*: Κρύβει/Εμφανίζει το παράθυρο με τις αναλυτικές πληροφορίες,
- *General Settings*: Εμφανίζει το παράθυρο ρυθμίσεων στις γενικές επιλογές,
- *Account Settings*: Ομοίως με το *General Settings* αλλά εμφανίζει τις ρυθμίσεις λογαριασμών, π.χ. προσθήκη υπηρεσίας, επιλογή φακέλου,
- *Send Feedback:* Εμφανίζει ένα παράθυρο για αποστολή σχολίων στον δημιουργό,
- *Quit:* Έξοδος από την εφαρμογή.

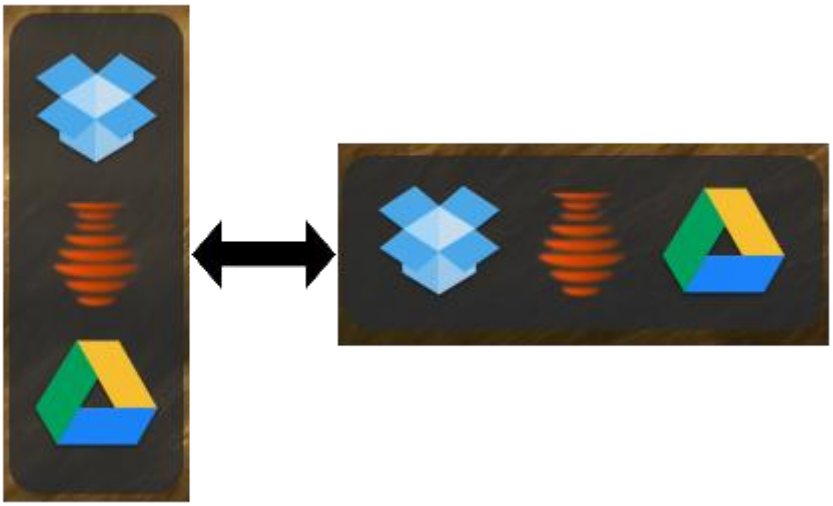
Η τελευταία λειτουργία του εν λόγω εικονιδίου είναι η εμφάνιση και η απόκρυψη των πρόσφατων διαμοιρασμένων αρχείων με το αριστερό κουμπί του ποντικιού:



Εικόνα 37: Παράθυρο πρόσφατων διαμοιρασμών αρχείου

Στην παραπάνω εικόνα φαίνονται οι πέντε πρόσφατες αποστολές αρχείων. Επίσης, περνώντας τον δείκτη του ποντικιού πάνω από κάποια εγγραφή εμφανίζεται το κουμπί '*Share Link*' το οποίο μπορεί να πατήσει ο χρήστης για να αντιγράψει στο προσωρινό χώρο(clipboard) του λειτουργικού του τον σύνδεσμο για διαμοιρασμό.

Το κύριο παράθυρο της εφαρμογής, το εργαλείο στην επιφάνεια εργασίας, μπορεί να βρίσκεται σε δυο καταστάσεις, την οριζόντια και την κάθετη, όπως φαίνεται στην επόμενη εικόνα:



Εικόνα 38: Πιθανές διατάξεις του εργαλείου στην επιφάνεια εργασίας

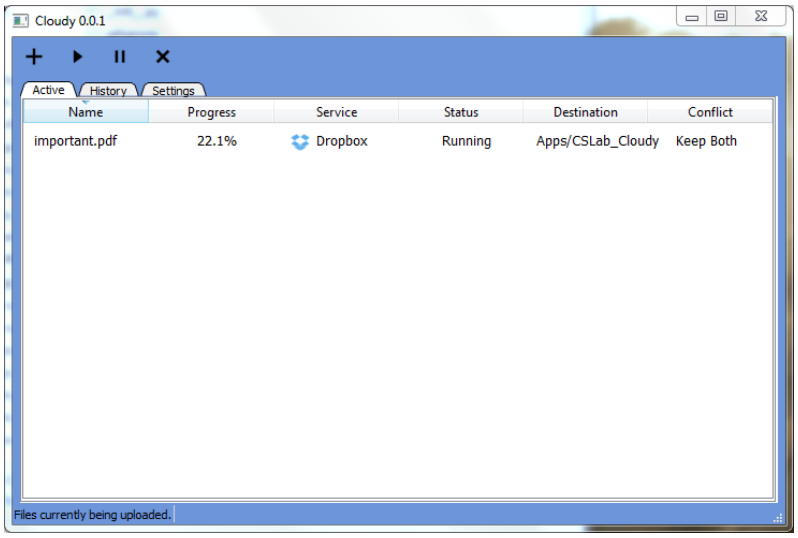
Η εναλλαγή ανάμεσα στις καταστάσεις γίνεται με γρήγορο διπλό πάτημα του αριστερού κουμπιού του ποντικιού πάνω στο παράθυρο. Επίσης, το παράθυρο μπορεί να μετακινηθεί κρατώντας πατημένο το αριστερό κουμπί του ποντικιού και σέρνοντας το στην επιθυμητή θέση.

Έστω ότι ο χρήστης επιθυμεί να ανεβάσει ένα αρχείο μέσω του παραθύρου αυτού στην υπηρεσία Pithos+.

Εικόνα 39: Διαδικασία αποστολής αρχείου μέσω του εργαλείου στην επιφάνεια εργασίας

Βλέπουμε πως το εικονίδιο της υπηρεσίας έχει μεγαλώσει σε κλίμακα για να δώσει το μήνυμα πως αν αφήσει το αρχείο εκεί θα αποσταλεί στην υπηρεσία Pithos+.

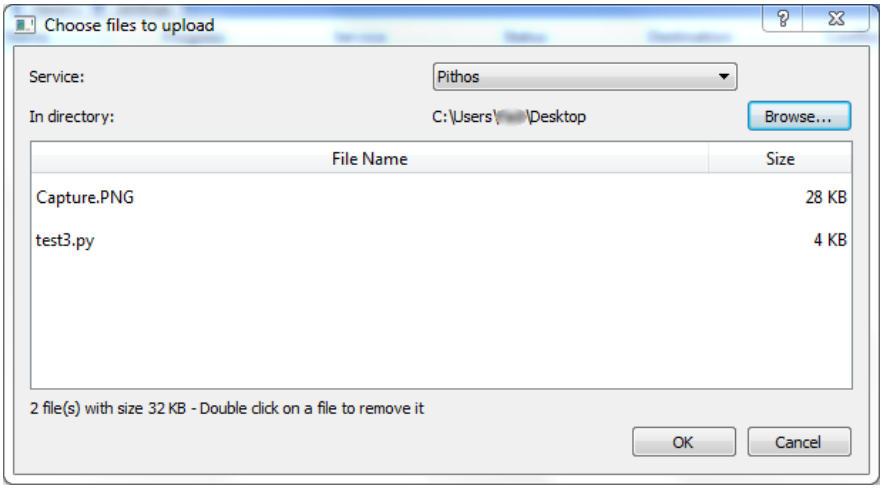
Πατώντας στην επιλογή '*Hide/Show Main window*' στο μενού του εικονιδίου στην γραμμή εργαλείων, εμφανίζεται το παράθυρο λεπτομερειών όπως φαίνεται παρακάτω:



Εικόνα 40: Παράθυρο λεπτομερειών

Βλέπουμε πως στο πάνω μέρος βρίσκονται τα κουμπιά ελέγχου *'Add', 'Resume', 'Pause', 'Delete'*. Ας δούμε την λειτουργία κάθε κουμπιού με την σειρά:

- '*Add*': Εμφανίζει ένα παράθυρο για την αποστολή αρχείων που προσφέρει μεγαλύτερο έλεγχο από ότι το παράθυρο στην επιφάνεια εργασίας. Ένα παράδειγμα χρήσης φαίνεται παρακάτω:

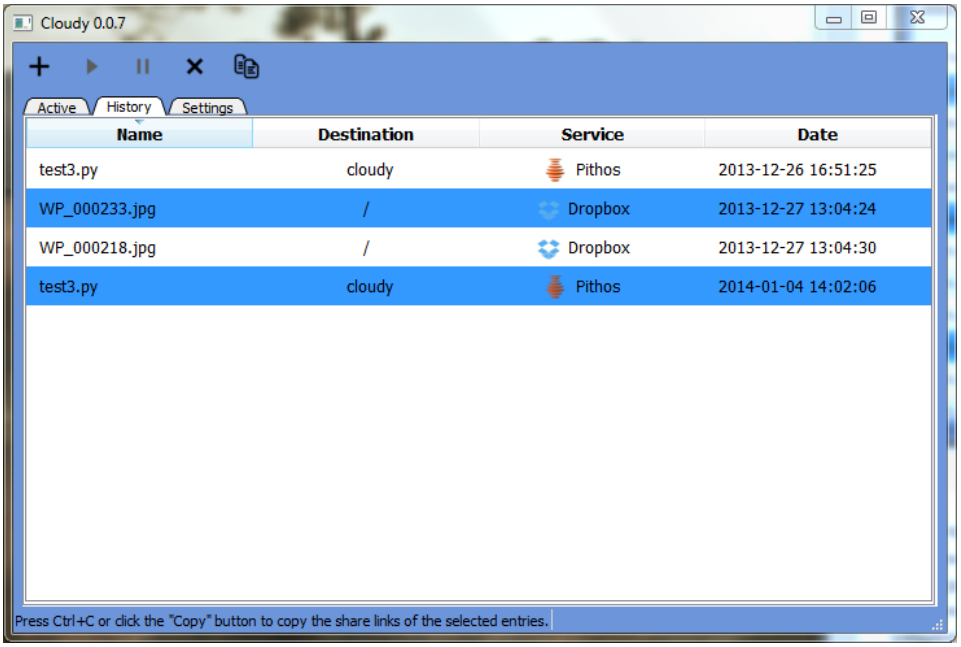


Εικόνα 41: Αναλυτικό παράθυρο αποστολής αρχείου

- '*Resume*': Επανεκκινεί τα επιλεγμένα αρχεία,

- '*Pause*': Τα επιλεγμένα αρχεία που αποστέλλουν δεδομένα σταματούν,
- '*Delete*': Διαγράφονται τα επιλεγμένα αρχεία,

Στο tab με το όνομα '*History*' περιέχεται το ιστορικό των αρχείων που έχουν ανέβει μέσω της εφαρμογής.



Εικόνα 42:Οθόνη ιστορικού

Αν ο χρήστης πατήσει τον συνδυασμό πλήκτρων Ctrl και C ή το δεξιότερο κουμπί στην γραμμή εργαλείων του παραθύρου τότε στον προσωρινό χώρο θα αντιγραφούν οι σύνδεσμοι των αρχείων για τον διαμοιρασμό τους. Αν ένα μόνο αρχείο είναι επιλεγμένο τότε στον προσωρινό χώρο θα αντιγραφεί μόνο ο σύνδεσμος αλλιώς θα αντιγραφούν οι σύνδεσμοι συνοδευόμενοι από τα ονόματα των αρχείων στην εκάστοτε υπηρεσία, για παράδειγμα:

Ένα αρχείο:

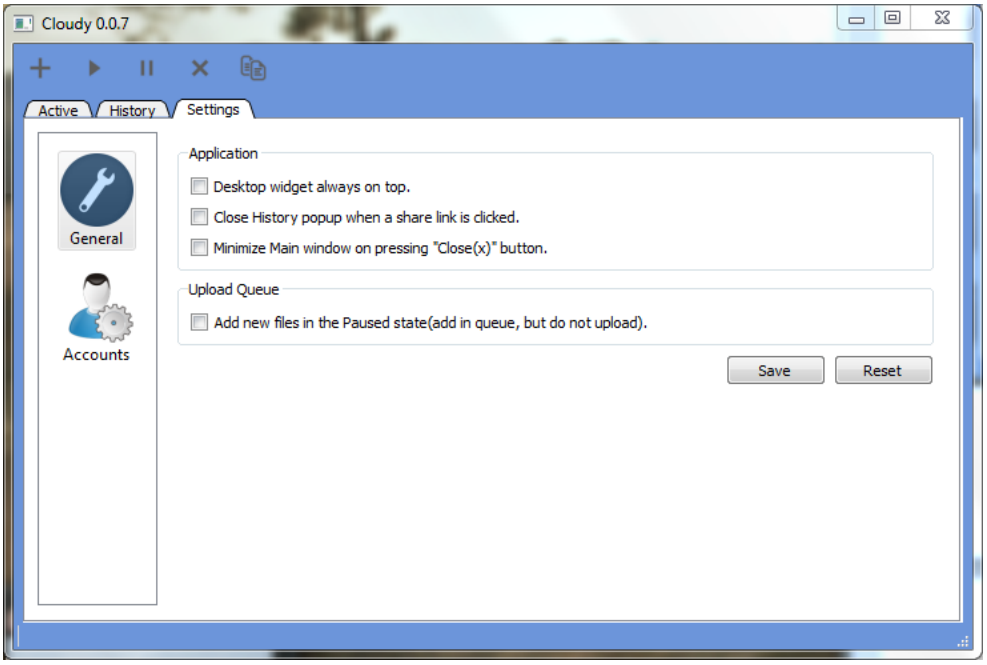
https://www.dropbox.com/s/rh1ix1rvms2zNlp/test3.py

Πολλαπλά αρχεία:

WP_000218.jpg: *https://www.dropbox.com/s/w8sb36tg6j6uSVv/*WP_000218.jpg

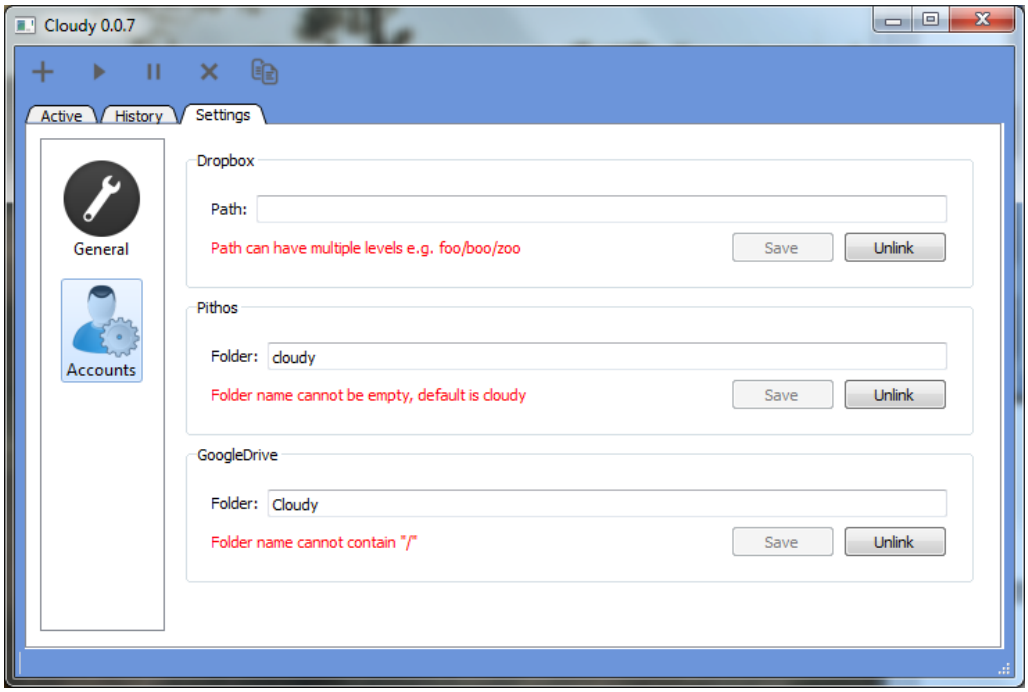
test3.py: *https://pithos.okeanos.grnet.gr/public/hGl2vfVloRZlzvG8V9YZY1*

Στο tab με όνομα '*Settings*' απενεργοποιούνται τα κουμπιά ελέγχου καθώς δεν χρησιμοποιούνται στο περιεχόμενο των ρυθμίσεων. Στο πρώτο μενού ρυθμίσεων, το '*General*', βλέπουμε δυο ομάδες επιλογών οι οποίες αφορούν ρυθμίσεις που αλλάζουν την συμπεριφορά της εφαρμογής και ρυθμίσεις σχετικές με την αποστολή των αρχείων.



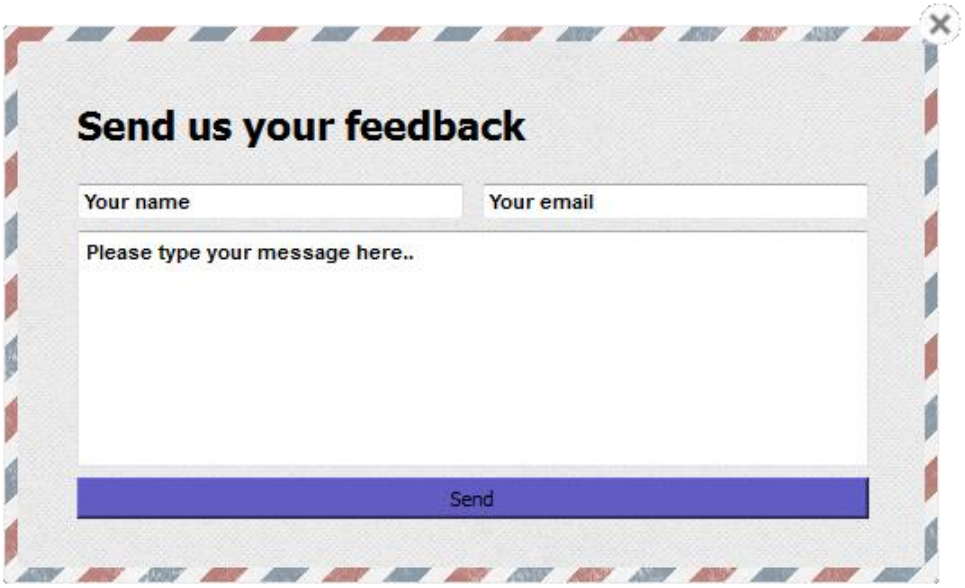
Εικόνα 43: Οθόνη γενικών ρυθμίσεων

Ο χρήστης επιλέγοντας το 'Accounts', μπορεί να προσθέσει κάποια υπηρεσία ή να επιλέξει τον φάκελο στον οποίο θα αποθηκεύονται τα αρχεία του σε κάθε εξουσιοδοτημένη υπηρεσία. Ένα στιγμιότυπο του παραθύρου αυτού είναι το ακόλουθο:



Εικόνα 44: Οθόνη ρυθμίσεων λογαριασμών

Το τελευταίο παράθυρο της εφαρμογής είναι αυτό της αποστολής σχολίων στους δημιουργούς:



Εικόνα 45: Παράθυρο αποστολής σχολίων

6.3 Προγραμματιστικά εργαλεία και απαιτήσεις εφαρμογής

Κατά την ανάπτυξη της εφαρμογής δεν χρησιμοποιήθηκαν εργαλεία που έπρεπε να χρησιμοποιηθούν για την ολοκλήρωση της αλλά βοήθησαν αισθητά στην εξερεύνηση και εκμάθηση των βιβλιοθηκών καθώς και στην υλοποίηση και δοκιμή ανεξάρτητων κομματιών κώδικα. Τα εργαλεία αυτά είναι:

- Ipython, 1.1.0: Είναι μια βελτιωμένη έκδοση του διερμηνευτή της Python, προσθέτοντας, μεταξύ πολλών άλλων, αυτόματη συμπλήρωση κώδικα και πολυχρηστικής ιστοσελίδας(notebook) που εκτελεί κώδικα Python,
- Dreampie: Είναι μια εναλλακτική έκδοση του διερμηνευτή της Python, που επιτρέπει την επεξεργασία και εκτέλεση κώδικα που καταλαμβάνει πολλές γραμμές.

Η σελίδα είναι σκόπιμα λευκή.

7

Επίλογος

Η εργασία ολοκληρώνεται σε αυτό το κεφάλαιο, ανακεφαλαιώνοντας την ροή σχεδιασμού του συστήματος σε μακροσκοπικό επίπεδο και των αποφάσεων που παρθήκαν μέχρι να πάρει την τελική της μορφή η εφαρμογή που υλοποιήθηκε στο πλαίσιο της διπλωματικής εργασίας. Επιπρόσθετα, καταγράφονται προτάσεις για μελλοντικές επεκτάσεις της εφαρμογής έτσι ώστε να βελτιωθεί σε τομείς όπως είναι η εμπειρία του χρήστη και η γενικότερη υποδομή της.

7.1 Σύνοψη και συμπεράσματα

Μετά από πολλαπλές επαναλήψεις πάνω στον σχεδιασμό του συστήματος και στην δομή της υλοποίησης, παρουσιάστηκε στην παρούσα εργασία ο κατασταλαγμένος τρόπος σκέψης και η διαδικασία για την λεπτομερή ανάλυση, την στοχευμένη σχεδίαση και πλήρη ανάπτυξη της εφαρμογής.

Ο στόχος της ήταν να προσφέρει στον χρήση μια πολύ απλή διαπροσωπεία για τον εύκολο και άμεσο διαμοιρασμό αρχείων χωρίς όμως να πάσχει από άποψη ρυθμίσεων και παροχής πληροφοριών επί της τρέχουσας κατάστασης. Επιλέχτηκε να γίνει διαχωρισμός των μονάδων χρησιμοποιώντας το αρχιτεκτονικό μόρφημα Model-View-Controller, γεγονός που μας οδήγησε στην ομαλή επικοινωνία μεταξύ τους και στην ευκολία κλιμάκωσης της εφαρμογής.

Αρχικά, παρουσιάστηκαν και αναλύθηκαν οι διάφορες τεχνικές έννοιες που χρησιμοποιήθηκαν καθ' όλη την έκταση της εργασίας. Στην συνέχεια, κατεγράφησαν οι απαιτήσεις του συστήματός μας καθώς και οι λειτουργικές και ποιοτικές απαιτήσεις της εφαρμογής που πρέπει να ικανοποιεί. Στο πέμπτο κεφάλαιο, παρουσιάστηκαν οι βασικότερες κλάσεις της υλοποίησης και ο τρόπος με τον οποίο αυτές απαρτίζουν το συνολικό σύστημά μας. Τέλος, εξετάστηκαν ορισμένες λεπτομέρειες της υλοποίησής οι

οποίες απαρτίζονται από αποφάσεις τεχνικής φύσεως και από διαδικασίες που θεωρήθηκαν πως χρήζουν αναφοράς.

Συνοψίζοντας, οι τεχνολογίες cloud, παρότι άρχισαν να αξιοποιούνται σε μεγάλη κλίμακα από τις αρχές της χιλιετίας, ακμάζουν και υιοθετούνται από εταιρείες και οργανισμούς ραγδαίως. Πολλές υπηρεσίες τύπου IaaS, τείνουν να μην περιορίζονται στο κομμάτι των υποδομών αλλά προσφέρουν μια ευρεία οικογένεια υπηρεσιών για παράδειγμα η υπηρεσία ~okeanos που έχει συνδέσει την Pithos+, που χρησιμοποιούμε στην εργασία αυτή, και την υπηρεσία cyclades, η οποία αφορά την δημιουργία και την διαχείριση παραμετροποιήσιμων εικονικών μηχανημάτων μεταξύ πολλών άλλων λειτουργιών. Όμοιες, υπηρεσίες προσφέρουν η Amazon, με την ονομασία Amazon Web Services, και η Google, της οποίας η οικογένεια cloud υπηρεσιών ονομάζεται Google App Engine.

Συγκεκριμένα, η υλοποιημένη εφαρμογή, έχει τα χαρακτηριστικά που θέσαμε ως στόχο στην αρχή της παρούσας εργασίας δηλαδή:

- Να είναι αποκρίσιμη σε κάθε στιγμή ανεξαρτήτως των έργων που εκτελούνται στο προσκήνιο,
- Να έχει όσο πιο απλή διαπροσωπεία γίνεται αλλά ταυτοχρόνως να προσφέρει εργαλεία για πλήρη έλεγχο επί της κατάστασης της στον χρήστη,
- Να έχει ανοχή σε ποικίλα σφάλματα, όπως είναι σφάλματα δικτύου, εξουσιοδότησης, τοπικών αρχείων και υπηρεσιών,

Η ροή σκέψης και ανάλυσης που ακολουθήθηκε σε όλη την έκταση της εργασίας μπορεί να λειτουργήσει ως οδηγός για την δόμηση και την οργάνωση εφαρμογών που δεν περιορίζονται στο είδος της συγκεκριμένης υλοποίησης αλλά και στην επίλυση προβλημάτων όμοιας φύσεως.

7.2 Μελλοντικές επεκτάσεις

Το *Cloudy*, η εφαρμογή που υλοποιήθηκε, δεν είναι σε καμία περίπτωση πλήρης από την άποψη χαρακτηριστικών και μπορεί να λειτουργήσει ως βάση για περαιτέρω ανάπτυξη. Οι αναβαθμίσεις αφορούν ήδη υπάρχοντα χαρακτηριστικά αλλά και νέα χαρακτηριστικά που μπορούν να προστεθούν.

Ένα από τα χαρακτηριστικά που μπορούν να προστεθούν είναι η προσθήκη επιλογής στο μενού που εμφανίζεται όταν πατηθεί το δεξί κουμπί του ποντικιού πάνω σε επιλεγμένα αρχεία η οποία θα στέλνει τα εν λόγω αρχεία μέσω του *Cloudy* σε κάποια υπηρεσία. Η γενική ιδέα απεικονίζεται παρακάτω:

Εικόνα 46: Επέκταση εφαρμογής – Δεξί κλικ στα αρχεία

Οι υπόλοιπες βελτιώσεις που προτείνονται για την εφαρμογή είναι οι ακόλουθες:

- Θα μπορούσε να έχει υποδομή για την προσθήκη υπηρεσιών που δεν υποστηρίζονται εξ' αρχής, για παράδειγμα αν ο χρήστης δεν χρησιμοποιεί καμία από τις Dropbox, Pithos+ ή GoogleDrive, να έχει την δυνατότητα να εγκαταστήσει μέσω μιας εύκολης διαδικασίας να προσθέσει την δικιά του. Με αυτόν τον τρόπο θα αυξηθεί η χρήση της καθώς ο κάθε χρήστης θα μπορεί να την παραμετροποιήσει όπως επιθυμεί.
- Είναι σημαντικό η εφαρμογή να δίνει την δυνατότητα στον χρήστη να μοιραστεί το αρχείο που ανέβασε με πολλούς τρόπους. Αυτή τη στιγμή ο μόνος τρόπος που υποστηρίζεται είναι μέσω αντιγραφής των συνδέσμων στον πρόχειρο χώρο του λειτουργικού. Η διαδικασία θα μπορούσε να επεκταθεί προσθέτοντας την επιλογή να αποστέλλεται αυτόματα μέσω e-mail συνοδευόμενο από κάποιο προσωπικό μήνυμα.
- Όσον αφορά την παραμετροποίηση της εφαρμογής, θα μπορούσε να προστεθεί στις ρυθμίσεις μια επιλογή για περιορισμό των ενεργών αποστολών και της ταχύτητας αποστολής είτε σε συνολικό επίπεδο είτε ανά υπηρεσία.
- Μέσω της οθόνης ιστορικού θα μπορούσε να προστεθεί ένα κουμπί που αν πατηθεί θα διαγράφει τα επιλεγμένα αρχεία από τους απομακρυσμένους αποθηκευτικούς χώρους τους της αντίστοιχης υπηρεσίας.
- Για λεπτομερέστερη παραμετροποίηση, η προσθήκη ενός προγραμματιστή που ρυθμίζει, παραδείγματος χάριν ανάλογα με την ώρα της ημέρας, το όριο του ρυθμού αποστολής δεδομένων.
- Για να δοθεί περισσότερος έλεγχος στην συμπεριφορά της εφαρμογής όσον αφορά τις υπηρεσίες, μπορεί να προστεθεί στο σύστημα η επιλογή καθορισμού της προτεραιότητας των υπηρεσιών. Για παράδειγμα εάν ο χρήστης χρησιμοποιεί δυο υπηρεσίες, μπορεί να ορίσει την μια με μεγαλύτερη προτεραιότητα από την άλλη έτσι ώστε μόλις αρχίσει την αποστολή στην υπηρεσία με την μεγαλύτερη προτεραιότητα, αυτές τις δεύτερης υπηρεσίας να σταματήσουν μέχρι να ολοκληρωθούν της πρώτης.
- Με το πέρας του χρόνου, το αρχείο του ιστορικού θα μεγαλώνει σε μέγεθος κρατώντας εγγραφές που δεν ενδιαφέρουν πια τον χρήστη. Για αυτό το λόγο, θα μπορούσε να εισαχθεί μια επιλογή έτσι ώστε να κρατάει το πιο πρόσφατο ιστορικό, για παράδειγμα της τελευταίας εβδομάδας.
- Ορισμένες φορές ο χρήστης ανεβάζει ένα μεγάλο αρχείο και απομακρύνεται από τον υπολογιστή του. Έχοντας αυτό κατά νου, ο χρήστης πρέπει να έχει την δυνατότητα να θέσει, πριν την απομάκρυνσή, μια αυτόματη ενέργεια που θα εκτελεστεί κατά την ολοκλήρωση των αποστολών, όπως το να κλείσει τον υπολογιστή ή να στείλει με e-mail τους συνδέσμους των αρχείων που απεστάλησαν.
- Θα μπορούσε να δοθεί η δυνατότητα στον χρήστη να αποθηκεύσει σε ελεγχόμενους πόρους τις ρυθμίσεις του, έτσι ώστε αν εγκαταστήσει την εφαρμογή σε διαφορετικό υπολογιστή να κατεβάσει τις αποθηκευμένες ρυθμίσεις του με αποτέλεσμα ο χρόνος εγκατάστασης και παραμετροποίησης να γίνει ο ελάχιστος δυνατός.
- Σε πολλές περιπτώσεις το παράθυρο στην επιφάνεια εργασίας, αναλόγως της ρύθμισης, ορισμένες φορές είτε θα καλύπτεται εξ' ολοκλήρου είτε θα είναι πάντα πάνω από οποιοδήποτε άλλο παράθυρο(always on top). Για τον λόγο αυτό είναι επιθυμητή η ενσωμάτωση μηχανισμού που θα εμφανίζει αυτόματα το παράθυρο όταν ο χρήστης σέρνει τα αρχεία πάνω από αυτό ακόμα και αν δεν είναι ορατό.

Η σελίδα είναι σκόπιμα λευκή.

8

Βιβλιογραφία

[SMCC] Service Models in Cloud Computing,
http://www.cloud-competence-center.com/understanding/cloud-computing-service-models/

[WCC] Wikipedia – Cloud Computing, http://en.wikipedia.org/wiki/Cloud_computing

[GHJV] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*,

[NDCC] NIST Definition of Cloud Computing,
http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[WPUI] Wikipedia - Principles of user interface design
http://en.wikipedia.org/wiki/Principles_of_user_interface_design

[DMCC] Deployment Models in Cloud Computing,
http://www.cloud-competence-center.com/understanding/cloud-computing-deployment-models/

[P27D] Python 2.7 Documentation, http://docs.python.org/2.7/

[WPPL] Wikipedia - Python Programming Language,
http://en.wikipedia.org/wiki/Python_(programming_language)

[PQCD] PyQt4 Class Documentation, http://pyqt.sourceforge.net/Docs/PyQt4/classes.html

[SD] Synnefo Documentation, http://www.synnefo.org/docs/synnefo/latest/index.html

[DD] Dropbox Developers, https://www.dropbox.com/developers

[GD] GoogleDrive Developers, https://developers.google.com/drive/

[OOP] Object-Oriented Programming,

http://searchsoa.techtarget.com/definition/object-oriented-programming

[WOOP] Wikipedia - Object-Oriented Programming,

http://en.wikipedia.org/wiki/Object-oriented_programming

[WFR] Wikipedia – Functional Requirements,

http://en.wikipedia.org/wiki/Functional_requirement

[WNFR] Wikipedia – Non-Functional Requirements,

http://en.wikipedia.org/wiki/Non-functional_requirement

[FRSS] Functional Requirements of Software Systems,

http://www.eogogics.com/talkgogics/infocenter/functional-requirements

[NFR] Non-Functional Requirements,

http://www.utd.edu/~chung/RE/NFR-18-4-on-1.pdf

[NFRC] Non-Functional Requirements Checklist,

http://thoughts-agile.blogspot.gr/2013/09/non-functional-requirements.html

[RFC5849] RFC 5849 – The OAuth 1.0 Protocol,

http://tools.ietf.org/html/rfc5849

[RFC6749] RFC 6749 – The OAuth 2.0 Authorization Framework,

http://tools.ietf.org/html/rfc6749

[TOB] The OAuth Bible,

https://github.com/Mashape/mashape-oauth/blob/master/FLOWS.md#terminology--reference

[WOA] Wikipedia - OAuth, http://en.wkipedia.org/wiki/OAuth

[UML] Wikipedia - Unified Modeling Language,

http://en.wikipedia.org/wiki/Unified_Modeling_Language

[UBTCD] UML Basics: The component diagram,

http://www.ibm.com/developerworks/rational/library/dec04/bell/

[UDD] UML Deployment Diagrams,

http://www.uml-diagrams.org/deployment-diagrams.html

[USD] UML Sequence Diagrams,

http://www.uml-diagrams.org/sequence-diagrams.html