ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Διαχείριση δεδομένων με χρήση Νέφους Αποθήκευσης για το Διαδίκτυο των Πραγμάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΝΙΚΟΛΑΟΣ ΦΥΤΡΟΣ

Επιβλέπων : Θεοδώρα Βαρβαρίγου

Καθηγήτρια Ε.Μ.Π.

Αθήνα, Αύγουστος 2014

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Διαχείριση δεδομένων με χρήση Νέφους Αποθήκευσης για το Διαδίκτυο των Πραγμάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΝΙΚΟΛΑΟΣ ΦΥΤΡΟΣ

Επιβλέπων : Θεοδώρα Βαρβαρίγου

Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 25η Αυγούστου 2014.

.....	.....	.....
Θεοδώρα Βαρβαρίγου Καθηγήτρια Ε.Μ.Π.	Ελευθέριος Καγιάφας Καθηγητής Ε.Μ.Π.	Βασίλειος Λούμος Καθηγητής Ε.Μ.Π.

Αθήνα, Αύγουστος 2014

.....

ΝΙΚΟΛΑΟΣ ΦΥΤΡΟΣ

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

Περίληψη

Η παρούσα διπλωματική εργασία πραγματεύεται τον τρόπο με τον οποίο μπορεί να επιτευχθεί αξιόπιστη και γρήγορη διαχείριση δεδομένων τα οποία προέρχονται από συσκευές συνδεδεμένες στο διαδίκτυο των πραγμάτων.
Τα δεδομένα αυτά θα φτάνουν σε έναν τοπικό διακομιστή ιστού στον οποίο θα βρίσκεται η εφαρμογή που είναι υπεύθυνη για την επεξεργασία τους και την αποθήκευσή τους σε ένα νέφος αποθήκευσης αντικειμένων.
Το νέφος αυτό είναι εγκατεστημένο σε μία εικονική μηχανή μίας διαδικτυακής υπηρεσίας παροχής εικονικών μηχανών. Θα εξηγηθεί ο τρόπος με τον οποίο εγκαθίσταται και εδραιώνεται η λειτουργία του νέφους αποθήκευσης αντικειμένων
της Openstack καθώς και ο τρόπος επικοινωνίας του τοπικού διακομιστή με το νέφος αυτό. Το κύριο κομμάτι υλοποίησης της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη της εφαρμογής που θα είναι υπεύθυνη για την λήψη των δεδομένων από τις συσκευές αλλά και την μετατροπή και αποθήκευση αυτών ως αντικείμενα στο νέφος. Η γλώσσα προγραμματισμού που χρησιμοποιείται είναι η Java και η τεχνολογία είναι τα JavaServlets. Τέλος εξηγείται η REST αρχιτεκτονική καθώς το νέφος αποθήκευσης της Openstack είναι φτιαχμένο με βάση αυτήν. Η διαχείριση μεγάλης ποσότητας δεδομένων που προέρχονται από συσκευές είναι ένα μείζον θέμα της σημερινής εποχής. Το διαδίκτυο των πραγμάτων προσπαθεί να εκμεταλλευτεί όλα τα δεδομένα αυτά, για να παράγει δυναμικά αποτελέσματα. Έτσι θα δημιουργηθούν έξυπνα σπίτια, εργοστάσια, πόλεις και άλλα, τα οποία θα έχουν τις ιδιότητες της αυτοδιαχείρισης, του αυτοελέγχου και γενικά της δυνατότητας να παίρνουν αποφάσεις για τις λειτουργίες τους, διευκολύνοντας έτσι τον άνθρωπο.

Λέξεις Κλειδιά: διαδίκτυο των πραγμάτων, JAVA, J2EE,
νέφος αποθήκευσης αντικειμένων, υπολογιστικό νέφος, REST, JavaServlets, JSP

Abstract

This diploma thesis addresses the means in which reliable and fast data management could be achieved, with data comming from devices connected to the Internet of Things. These data will reach a local web server to which a web application will reside. This application is responsible for processing and storing these data in an object storage cloud. This object cloud is established within a vitrual machine of a web service that provides virtual machines.
The process of installing and affirmation of the Openstack cloud functionality will be explained as well as the way that the local server communicates with the cloud. The programmin language that is used is Java and the
technology is JavaServlets. Lastly REST architecture is explained because the object storage cloud is build based on this. The management of large amount of data -known as big data- is a major issue of today's technology advance and Internet of Things tries to set it to the next level because the things on this network will not be cellphones or pcs but devices like sensors etc that can provide usefull information through measurments and behaviors.

Keywords: Internet of Things, JAVA, J2EE,Object Storage Cloud, Cloud Computing, REST, JavaServlets, JSP

Ευχαριστίες

Θα ήθελα αρχικά να ευχαριστήσω την Καθηγήτρια Ε.Μ.Π. κ. Θεοδώρα Βαρβαρίγου για τη δυνατότητα που μου έδωσε να δουλέψω κάτω από την επίβλεψη της και τη αμέριστη εμπιστοσύνη που μου έδειξε.

Επίσης, θα ήθελα να ευχαριστήσω τον διδάκτορα Ε.Μ.Π. κ. Σπυρίδωνα Β. Γωγουβίτη και τον υποψήφιο διδάκτορα Ε.Μ.Π. κ. Αχιλλέα Μαρινάκη για τις συμβουλές που μου έδωσαν, την καθοδήγηση στην εκπόνηση της εργασίας αυτής και για τον άπλετο χρόνο που μου αφιέρωσαν.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου για τη αγάπη και στήριξη που μου δίνουν αδιάκοπα όλα αυτά τα χρόνια για να εκπληρωθούν τα όνειρα και οι φιλοδοξίες μου.

Πίνακας Περιεχομένων

Πίνακας Εικόνων

Πίνακας Ακρονύμων

API	Application Programming Interface
CIFS	Common Internet File System
CSS	Cascading Style Sheets
ERP	Enterprise Resource Planning
FASP	Fetal Anomaly Screening Programme
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IaaS	Infrastructure as a Service
IDE	Integrated Development Environment
IoT	Internet of Things
ISCS	Internet Software Company Smart
J2EE	Java 2 Platform Enterprise Edition
JSON	JavaScript Object Notation
JSP	Java Server Pages
M2M	Machine-to-Machine
mac address	media access control address
NFS	Network File System
OPEX	Operating expenditure
PaaS	Platform as a Service
REST	REpresentational State Transfer
RFID	Radio-Frequency Identification
RTT	Round-trip Time
SaaS	Software as a Service
SAIO	Swift All In One
SCSI	Small Computer System Interface
Sla	Service-level agreement
SOAP	Simple Object Access Protocol
SRA	Solicitors Regulation Authority
TCP	Transmission Control Protocol
ΤΠΕ	Τεχνολογίες Πληροφοριών και Επικοινωνιών
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VM	Virtual Machine
XML	Extensible Markup Language

Κεφάλαιο 1: Εισαγωγή

1.1. Είδη υπολογιστικών νεφών και υπηρεσίες που παρέχουν

Εικόνα 2.1 Cloud Computing

Ένα υπολογιστικό νέφος είναι ένα σύνολο απομακρυσμένων υπολογιστών οι οποίοι είναι συνδεδεμένοι και συνεργάζονται μεταξύ τους. Βασίζεται λοιπόν στην ιδέα του διαμοιρασμού υπολογιστικών πόρων ή εφαρμογών οι οποίες βρίσκονται σε απομακρυσμένες τοποθεσίες αντί να βρίσκονται σε τοπικούς εξυπηρετητές (servers) ή προσωπικές συσκευές και να καταπονούνται αυτές. Είναι δηλαδή ένα σύνολο αξιόπιστων υπηρεσιών στο οποίο μπορούμε να στηριζόμαστε και να αξιοποιούμε.
Το Cloud computing παρέχει ένα δυναμικό και ελαστικό περιβάλλον διάθεσης υπηρεσιών το οποίο μπορεί να είναι ανθεκτικό σε ραγδαίες και γιγαντιαίας κλίμακας μεταβολές των συνθηκών του. Αυτό επιτυγχάνεται με τα εγγενή χαρακτηριστικά του,

που είναι η αυτόματη ανάκαμψη, η αυτό-επιτήρηση, η αυτό-διαχείριση, η αυτόματη επαναδιαμόρφωση, και οι υψηλές δυνατότητες (αυτό)κλιμάκωσης.

Είδη υπηρεσιών υπολογιστικών νεφών

Το Cloud Computing μπορεί να διαχωριστεί σε δυο κατηγορίες:

- ως προς το είδος της υπηρεσίας που προσφέρεται
- ως προς το sourcing μοντέλο.

Ξεκινώντας από τα είδη των υπηρεσιών, τα διαθέσιμα μοντέλα του cloud computing είναι τα εξής:

- **Software-as-a-Service(SaaS)** (παροχή λογισμικού σαν υπηρεσία)
- **Platform-as-a-Service(Paas)** (παροχή πλατφόρμας σαν υπηρεσία)
- **Infrastructure-as-a-Service(IaaS)** (παροχή υποδομής σαν υπηρεσία)

Το κάθε ένα από αυτά, εξυπηρετεί διαφορετικές ανάγκες και προσφέρει διαφορετικές υπηρεσίες οι οποίες αναλύονται παρακάτω:

Το **Software-as-a-Service** βασίζεται στη λογική της υπερενοικίασης λογισμικού από έναν πάροχο υπηρεσιών, αντί της αγοράς της άδειας χρήσης. Το λογισμικό λειτουργεί σε ένα κεντρικοποιημένο δίκτυο servers προκειμένου να διατίθεται ως υπηρεσία από το web ή το διαδίκτυο. Επίσης αποκαλείται και «software on demand» και αποτελεί τον πλέον γνωστό τύπο cloud computing λόγω της μεγάλης ευελιξίας, ποιότητας υπηρεσιών, υψηλής σταθερότητας και της ελάχιστης δυνατής συντήρησης που απαιτεί. Ο πάροχος της υπηρεσίας φιλοξενεί και την εφαρμογή, αλλά και τα δεδομένα. Έτσι οι χρήστες μπορούν να την χρησιμοποιήσουν όπου και να βρίσκονται. Το SaaS μοντέλο είναι πολύ αποτελεσματικό στη μείωση του κόστους, αφού παρέχεται στην επιχείρηση ως μηνιαίο λειτουργικό κόστος, το οποίο συνήθως είναι κατά πολύ οικονομικότερο από την αγορά των αντίστοιχων αδειών χρήσης και υποδομής. Στο SaaS μοντέλο δεν απαιτείται καμία συντήρηση ή αναβάθμιση, αφού ο τελικός αποδέκτης δε χρειάζεται να μεριμνήσει για τη διαθεσιμότητα, την κλιμάκωση και τη χωρητικότητα.

Ως συνέχεια του **SaaS**, το **Platform-as-a-Service** παρέχει μια cloud πλατφόρμα εφαρμογών για εταιρείες ή ιδιώτες που κατασκευάζουν λογισμικό, είτε για χρήση από τους ίδιους είτε για τρίτους. Το μοντέλο αυτό παρέχει τις κατάλληλες υπηρεσίες προκειμένου κάποιος να μπορέσει να αναπτύξει, να δοκιμάσει, να διαθέσει και να συντηρήσει εφαρμογές και υπηρεσίες μέσα σε ένα ενιαίο περιβάλλον πλατφόρμας το οποίο είναι εγγενώς υψηλά διαθέσιμο, ελαστικό και ευέλικτο, με δυνατότητες πλήρης αυτό-διαχείρισης, αυτό-συντήρησης και αυτό-κλιμάκωσης της υποδομής, του λειτουργικού συστήματος και της πλατφόρμας εφαρμογών. Δηλαδή, με το PaaS δεν

χρειάζεται να ασχολείται κανείς με τη συντήρηση του λειτουργικού συστήματος και της πλατφόρμας. Όμως από τη άλλη πλευρά, δεν θα έχει και την δυνατότητα λεπτομερούς ελέγχου αυτών. Το PaaS βασίζεται στο μοντέλο «Pay-per-use» με τέτοιο τρόπο ώστε να επιτυγχάνεται η πλήρης αξιοποίηση των υπολογιστικών πόρων που χρησιμοποιούνται σε σχέση με το κόστος χρήσης. Αν συνδυαστεί με το χαρακτηριστικό της αυτο-κλιμάκωσης μπορούμε να πετύχουμε τη διάθεση υπηρεσιών που να μπορούν να ανταποκρίνονται σε οποιαδήποτε ραγδαία ή αναμενόμενη μεταβολή χωρητικότητας (ισχύ, μνήμη, αποθηκευτικό χώρο, δίκτυο) που θα απαιτηθεί ανά πάσα στιγμή, χωρίς να έχει δεσμευτεί εκ των προτέρων, είτε με αγορά υποδομής, λογισμικού πλατφόρμας, δικτυακής γραμμής υψηλής χωρητικότητας κλπ., είτε με ένα συμβόλαιο παροχής υπηρεσιών φιλοξενίας υποδομής και πλατφόρμας συγκεκριμένης χωρητικότητας και χρονικής διάρκειας.

Το τρίτο και τελευταίο μοντέλο είναι το **Infrastructure-as-a-Service,** το οποίο είναι η παροχή υπολογιστικών και δικτυακών υποδομών ως μια πλήρως outsourced υπηρεσία. Η εταιρεία ή ο ιδιώτης μπορεί να υπερνοικιάσει μια υποδομή (όχι όμως και πλατφόρμα όπως στο PaaS), ανάλογα με τις απαιτήσεις εκείνης της χρονικής στιγμής, όπως και στο PaaS, «Pay as you go», αντί να προβεί στην αγορά εξοπλισμού (υπολογιστικού, δικτυακού, κλπ) ή στη σύναψη συμβολαίου παροχής υπηρεσιών φιλοξενίας υποδομής για συγκεκριμένο χρονικό διάστημα. Σημαντικό πλεονέκτημα του IaaS είναι επίσης η δυνατότητα μεταφοράς εικονικών μηχανών από το ιδιόκτητο περιβάλλον της εταιρείας ή του ιδιώτη στο cloud, με συνοπτικές διαδικασίες. [1]

Sourcing Μοντέλα στο Cloud Computing

Το sourcing μοντέλο των cloud computing υπηρεσιών διαχωρίζεται σε:

- Public Cloud (δημόσιο νέφος)
- Dedicated Cloud (αφοσιωμένο νέφος)
- Private Cloud (ιδιωτικό νέφος) και
- Private Cloud Appliance (ιδιωτικό νέφος συσκευών)

Το **Public cloud** αποτελεί ένα σύνολο από υπολογιστικούς πόρους, οι οποίοι διατίθενται πάνω από το διαδίκτυο. Το Public Cloud computing έχει τα ακόλουθα πλεονεκτήματα: μεγάλη ευελιξία λόγω της άμεσης διάθεσης υπηρεσιών, υπάρχει άμεση κλιμάκωση σε μεγαλύτερη ή μικρότερη χωρητικότητα σε μόλις μερικά λεπτά, και όλες οι υπηρεσίες προσφέρονται με βελτιωμένη και συνεχή διαθεσιμότητα, ελαστικότητα, ασφάλεια και διαχειρισιμότητα.

Το **Dedicated Cloud** παρέχει ό,τι και το Public Cloud με τη διαφορά ότι λειτουργεί σε υποδομή αποκλειστικής χρήσης. Χαρακτηριστικά όπως η ασφάλεια, η αποδοτικότητα

και -σε μερικές περιπτώσεις- οι δυνατότητες αποκλειστικής προσαρμογής, είναι υψηλότερου επιπέδου, αφού μπορούν να προσαρμοστούν για συγκεκριμένο καταναλωτή με ειδικές απαιτήσεις.

Το **Private Cloud** αποτελεί ένα σύνολο από υπολογιστικούς πόρους που προσφέρονται ως ένα προτυποποιημένο σύνολο υπηρεσιών, οι οποίες καθορίζονται, σχεδιάζονται και ελέγχονται από ένα συγκεκριμένο οργανισμό. Η επιλογή ανάπτυξης ενός Private Cloud συνήθως καθοδηγείται από την ανάγκη για τη διατήρηση του πλήρους ελέγχου ενός παραγωγικού περιβάλλοντος εξ' αιτίας ιδιαίτερων απαιτήσεων των εφαρμογών από πλευράς απόδοσης, ωριμότητας ή νομικού πλαισίου λειτουργίας.

Το **Private Cloud Appliance**, το οποίο αποτελεί ένα αποκλειστικό περιβάλλον που μπορεί να μεταφερθεί (συνήθως σε μορφή container) που παρέχεται και κατασκευάζεται από έναν κατασκευαστή, ο οποίος έχει τον αρχιτεκτονικό έλεγχο του, την ευθύνη διαχείρισης και συντήρησης των φυσικών υποδομών, ενώ η λογική διαχείριση του παραμένει στον τελικό καταναλωτή. Έτσι συνδυάζονται τα πλεονεκτήματα χρήσης προκαθορισμένης λειτουργικής αρχιτεκτονικής, μειώνοντας το ρίσκο διάθεσης υπηρεσιών μέσω της εσωτερικής ασφάλειας και ελέγχου. [2]

1.2. Πλεονεκτήματα cloud computing

Παρακάτω παρατίθενται κάποια πλεονεκτήματα της χρήσης cloud computing:

Οι χρήστες της υποδομής νέφους μπορούν να έχουν πρόσβαση σε εφαρμογές και τα δεδομένα τους οποιανδήποτε στιγμή από οπουδήποτε. Τα δεδομένα δεν περιορίζονται πια σε ένα σκληρό δίσκο στον υπολογιστή ενός χρήστη ή έστω εσωτερικά του δικτύου ενός οργανισμού αλλά πλέον η πρόσβαση μπορεί να γίνει απομακρυσμένα από οπουδήποτε.
Το κόστος αγοράς υλικού και αναβαθμίσεων μειώνεται. Η παροχή υπηρεσιών νέφους μειώνει στη πλευρά του πελάτη το κόστος για σύγχρονο και τελευταίας γενιάς υλικό. Πλέον δεν απαιτούνται ταχύτατοι υπολογιστές με πάρα πολλή μνήμη γιατί αυτά τα αναλαμβάνει η υποδομή του νέφους. Το μόνο που απαιτείται είναι ένα τερματικό, ελαχίστων απαιτήσεων, με περιηγητή για πρόσβαση στο διαδίκτυο. Το τερματικό μπορεί να αποτελείται από μια οθόνη, συσκευές εισόδου όπως πληκτρολόγιο και ποντίκι και όση ισχύ και μνήμη χρειάζεται για να τρέχουν οι εφαρμογές πρόσβασης (middleware) στις υπηρεσίες του νέφους.
Πάρα πολλές εταιρίες και οργανισμοί εξαρτώνται από τη χρήση του κατάλληλου λογισμικού στους υπολογιστές των υπαλλήλων τους. Το λογισμικό αυτό μπορεί να είναι κάποια σουίτα γραφείου όπως το Microsoft Office, κάποια λογιστική πλατφόρμα όπως το QuickBooks ή κάποιο πακέτο φτιαγμένο αποκλειστικά για τον

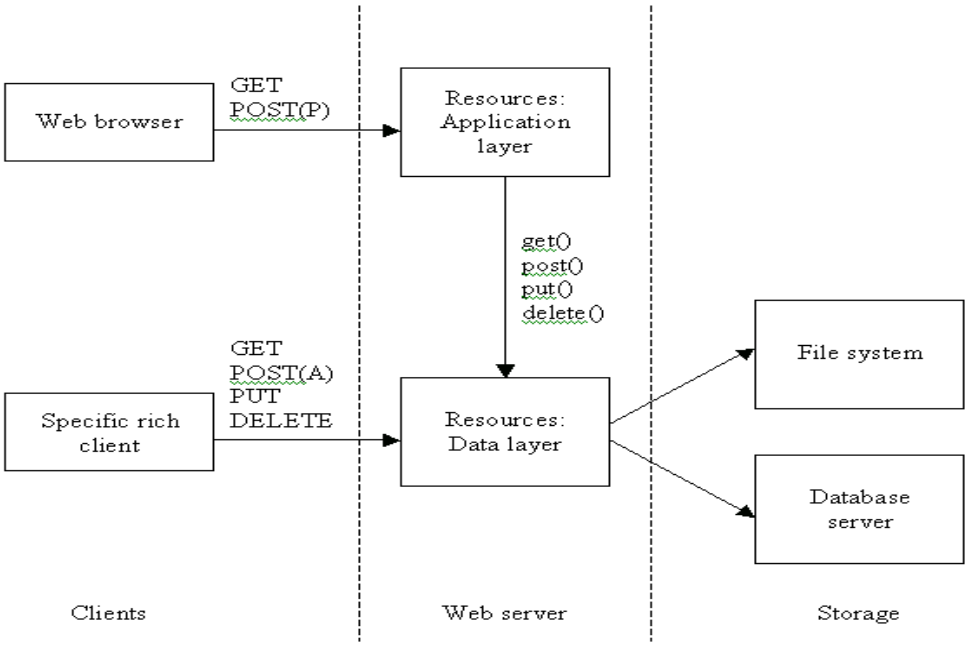
οργανισμό αυτό. Για τη χρήση του λογισμικού απαιτείται η αγορά του καθώς και των αδειών χρήσης του για κάθε υπολογιστή του οργανισμού, η εγκατάσταση και οι αναβαθμίσεις του. Όλα αυτά αυξάνουν το κόστος κατακόρυφα και πολλές φορές το κ αταντούν ασύμφορο. Τα οφέλη από τη παροχή λογισμικού από πάροχο υπηρεσιών νέφ ους είναι ότι μέσα σε λίγα λεπτά μπορείς να έχει έτοιμο και να τρέχει με τις άδειες χρή σης του το επιθυμητό λογισμικό. Το αντίτιμο για τις υπηρεσίες αυτές είναι μικρό και συ νήθως χρεώνεται με τον μήνα ή χρησιμοποιείται το μοντέλο “pay as you use” και υπάρχ ουν περιπτώσεις όπως web mail εφαρμογές (Google Gmail) και online σουίτες γραφείου (Google Docs) που είναι δωρεάν. Η αναβάθμιση και η συντήρηση γίνεται από τον πάροχο υπηρεσιών έτσι δεν υπάρχει η ανάγκη για κλήση τεχνικού κάθε φορά που χρειάζεται μια νέα αναβάθμιση.

Πολλές φορές οι ανάγκες και οι απαιτήσεις για περισσότερο υλικό και λογισμικό είναι μόνο για μικρό χρονικό διάστημα όμως οι απαιτήσεις αυτές θα πρέπει να καλυφθούν. Περιπτώσεις όπως, υπηρεσίες του κράτους που αναλαμβάνουν τη κατάθεση φορολογικών δηλώσεων, σώματα του στρατού που αναλαμβάνουν τη εισαγωγή νεοσυλλέκτων και γραμματείες σχολείων και πανεπιστημίων που πρέπει στη αρχή της χρονιάς να περάσουν τους νεοεισερχόμενους στο σύστημα τους είναι μόνο μερικά παραδείγματα όπου για κάποιο χρονικό διάστημα υπάρχει ανάγκη για κάλυψη των απαιτήσεων αυτών. Η αγορά επιπλέον υλικού αλλά και αδειών χρήσης λογισμικού για το διάστημα αυτό είναι ασύμφορο και το πιθαν ότερο είναι ότι μετά τη πάροδο του διαστήματος που τα έχουμε ανάγκη θα παραγκωνιστούν. Ακόμη τη επόμενη χρονιά πάλι θα απαιτείται η ανάγκη για υλικό ή έστω για αναβάθμιση καθώς η τεχνολογία δεν μένει στάσιμη. Επομένως τα μέχρι στιγμής πλεονεκτήματα που παρουσιάστηκαν θα μπορούσαν να συνδυαστούν έτσι ώστε σε περιπτώσεις σαν και αυτές να μειωθούν τα κόστη στο ελάχιστο.

Δεν είναι λίγες οι φορές όπου εταιρίες επιβαρύνονται με επιπλέον κόστος για αγορά ή ενοικίαση χώρου για τη τοποθέτηση των εξυπηρετητών και των ψηφιακών αποθηκευτικών μονάδων τους. Ακόμη στους χώρους αυτούς υπάρχει και η ανάγκη για συντήρηση, για κλιματισμό και φυσικά για φύλαξη του χώρου. Κλασσικό παράδειγμα είναι όταν μια εταιρία βλέπει να αυξάνονται οι ανάγκες της για υλικό και τη περιορίζει ο χώρος ή το υπάρχον υλικό έχει προβλήματα συμβατότητας με τη σημερινή τεχνολογία. Τα προβλήματα αυτά μπορούν να αποφευχθούν, αν όχι τουλάχισ τον να μειωθούν, με τη ενοικίαση της υποδομής από πάροχο υπηρεσιών νέφους.

Τέλος στις περιπτώσεις όπως πανεπιστημιακές κοινότητες, οργανισμοί και εταιρίες με τμήματα ερευνών που χρειάζονται μεγάλη υπολογιστική ισχύ και μνήμη για την εκτέλεση περίπλοκων υπολογισμών συχνά χρειάζονται μέρες ή και εβδομάδες για να εκ τελεστούν σε ένα υπολογιστή και απαιτείται δέσμευση πόρων για όσο χρόνο διαρκούν. Με τη χρήση νέφους που το back end του αποτελείται από πλέγμα υπολογιστών (grid computing system) μπορούμε να εκμεταλλευτούμε την επεξεργαστική ισχύ αλλά και μν ήμη ολόκληρου του δικτύου μειώνοντας το χρόνο εκτέλεσης των υπολογισμών.

1.3. Αρχιτεκτονική παραστατικής κατάστασης μεταφοράς (RESTful Architecture)



Εικόνα 3.2 REST Architecture

Τι είναι η RESTful αρχιτεκτονική

Η παραστατική κατάσταση μεταφοράς (REST) είναι ένα αρχιτεκτονικό στυλ που αποτελείται από ένα συντονισμένο σύνολο περιορισμών που εφαρμόζονται σε κατασκευαστικά στοιχεία, συνδέσμους και στοιχεία δεδομένων, μέσα σε ένα κατανεμημένο σύστημα υπερμέσων. Η REST αγνοεί τις λεπτομέρειες της υλοποίησης των στοιχείων και της σύνταξης του πρωτοκόλλου, προκειμένου να επικεντρωθεί στους ρόλους των στοιχείων, τους περιορισμούς στην αλληλεπίδρασή τους με άλλα στοιχεία, και την ερμηνεία τους από σημαντικά στοιχεία δεδομένων. Βασίζεται στη μεταφορά μηνυμάτων μέσω του πρωτοκόλλου HTTP μεταξύ του διακομιστή και του πελάτη.[3]

Περιορισμοί της αρχιτεκτονικής REST

Οι αρχιτεκτονικές ιδιότητες της REST μπορούν να πραγματοποιηθούν με την εφαρμογή ειδικών περιορισμών αλληλεπίδρασης στα συστατικά, στους συνδέσμους, και τα στοιχεία των δεδομένων. Οι τυπικοί περιορισμοί της REST είναι:

Πελάτης-διακομιστής (Client-server)
Μια ενιαία διεπαφή χωρίζει τους πελάτες από τους διακομιστές. Αυτός ο διαχωρισμός σημαίνει ότι, για παράδειγμα, οι πελάτες δεν ασχολούνται με την αποθήκευση δεδομένων -η οποία παραμένει στο εσωτερικό κάθε διακομιστή- έτσι ώστε ο κώδικας του πελάτη να είναι βελτιωμένη και πιο αποδοτικός. Οι διακομιστές δεν ασχολούνται με το περιβάλλον εργασίας του χρήστη ή της κατάστασης του χρήστη, έτσι ώστε ο κώδικας των διακομιστών να μπορεί να είναι απλούστερος και πιο επεκτάσιμος. Οι κώδικες των διακομιστών και των πελατών μπορούν επίσης να αντικατασταθούν και να αναπτύσσονται ανεξάρτητα, εφ 'όσον η διασύνδεση μεταξύ τους δεν μεταβάλλεται.

Χωρίς αποθήκευση κατάστασης (Stateless)
Κάθε αίτηση από οποιονδήποτε πελάτη περιέχει όλες τις απαραίτητες πληροφορίες έτσι ώστε ο διακομιστής να εξυπηρετήσει το αίτημα και η κατάσταση συνεδρίας διεξάγεται στον πελάτη. Σημαντικό να σημειωθεί είναι ότι η κατάσταση περιόδου λειτουργίας μπορεί να μεταφερθεί από τον server σε άλλη υπηρεσία , όπως μια βάση δεδομένων για να διατηρήσει μια πιο μόνιμη κατάσταση για κάποιο χρονικό διάστημα και να επιτρέψει τον έλεγχο ταυτότητας. Ο πελάτης ξεκινά την αποστολή των αιτήσεων όταν είναι έτοιμοι να κάνουν τη μετάβαση σε μια νέα κατάσταση και όταν έχουν πιστοποιηθεί τα στοιχεία του.

Προσωρινή αποθηκεύσιμη (Cacheable)
Όπως και στον παγκόσμιο ιστό (World Wide Web), οι πελάτες μπορούν να αποθηκεύουν προσωρινά τις απαντήσεις του διακομιστή. Οι απαντήσεις θα πρέπει , ως εκ τούτου, γραπτά ή ρητά, να αυτοπροσδιορίζονται ως προσωρινά αποθηκεύσιμες (cacheable) ή όχι, για να αποτρέψουν τους πελάτες να επαναχρησιμοποίησουν απαρχαιωμένα ή ακατάλληλα δεδομένα ως απάντηση σε περαιτέρω αιτήματα. Μία καλά διαχειριζόμενη προσωρινή αποθήκευση (caching), εν μέρει ή πλήρως, καταργεί ορισμένες αλληλεπιδράσεις client-server, βελτιώνοντας περαιτέρω την επεκτασιμότητα και την απόδοση.

Πολυεπίπεδο σύστημα
Ένας πελάτης δεν μπορεί κανονικά να καταλάβει αν είναι συνδεδεμένος απευθείας με τον διακομιστή ή σε έναν ενδιάμεσο μεταξύ αυτού και του διακομιστή. Ενδιάμεσοι διακομιστές μπορεί να βελτιώσουν την επεκτασιμότητα του συστήματος, με το να

επιτρέπουν εξισορρόπηση φορτίου και με το να παρέχουν κοινές προσωρινές μνήμες (caches). Μπορούν επίσης να επιβάλουν και τις πολιτικές ασφάλειας.

Κώδικας ανάλογα με τη ζήτηση (Code on demand)
Οι διακομιστές μπορούν να παρατείνουν προσωρινά ή να προσαρμόσουν την λειτουργικότητα ενός πελάτη με τη μεταφορά εκτελέσιμου κώδικα . Παραδείγματα είναι συστατικά όπως βοηθητικές εφαρμογές Java και προγράμματα από την μεριά του πελάτη (client-side scripts), όπως είναι αυτά που γράφονται με χρήση της γλώσσας προγραμματισμού JavaScript. Ο "κώδικας ανάλογα με τη ζήτηση" είναι ο μόνος προαιρετικός περιορισμός της αρχιτεκτονικής REST . [3]

Ενιαία διεπαφή της REST

Ο περιορισμός της ενιαίας διεπαφής είναι θεμελιώδους σημασίας για τον σχεδιασμό οποιασδήποτε υπηρεσίας REST. Το ενιαίο περιβάλλον απλοποιεί την αρχιτεκτονική, η οποία με τη σειρά της επιτρέπει σε κάθε σκέλος της να εξελιχθεί ανεξάρτητα. Η βασική ιδέα είναι ότι ο πελάτης στέλνει αιτήσεις σε κάποια συγκεκριμένα ενιαία αναγνωριστικά πόρων (uri) και παίρνει τα δεδομένα από τον διακομιστή μέσω του πρωτοκόλλου HTTP. Οι τέσσερις κατευθυντήριες αρχές αυτής της διεπαφής είναι οι:

Ταυτοποίηση των πόρων
Οι μεμονομένοι πόροι που ζητούνται από τον χρήστη-πελάτη μέσω των αιτήσεων είναι πλήρως διαχωρισμένοι από την παρουσίασή τους στον χρήστη. Για παράδειγμα μετά από μια αίτηση για κάποια δεδομένα από τον χρήστη ο διακομιστής δε στέλνει τη βάση δεδομένων αλλά στέλνει τα δεδομένα με τη μορφή κάποιας αναπαράστασης όπως HTML, XML, JSON. Δηλαδή με κάποια γλώσσα σήμανσης (markup language).

Επεξεργασία των πόρων μέσω των αναπαραστάσεων
Όταν ένας πελάτης κατέχει μία αναπαράσταση ενός πόρου - συμπεριλαμβανομένων οποιωνδήποτε μεταδεδομένων που επισυνάπτονται – έχει αρκετές πληροφορίες για να τροποποιήσει ή να διαγράψει τον πόρο αυτό.

Αυτο-περιγραφικά μηνύματα (αιτήσεις και απαντήσεις)
Κάθε μύνημα περιέχει αρκετές πληροφορίες για να περιγράψει πώς να το επεξεργαστεί ο διακομιστής.

Τα υπερμέσα ως κινητήρια δύναμη της κατάστασης της εφαρμογής
Οι πελάτες κάνουν μεταβάσεις μόνο μέσα από δράσεις που προσδιορίζονται δυναμικά μέσω υπερμέσων από τον διακομιστή (π.χ. μέσω hyperlinks).

Βασικές μέθοδοι του HTTP που χρησιμοποιούνται στη REST αρχιτεκτονική

Έστω ότι θέλουμε να στείλουμε μία αίτηση στο uri: http://example.com/resources, στο οποίο βρίσκεται μια συλλογή με πόρους (resources).
Οι βασικές μέθοδοι/λειτουργίες που μπορούμε να στείλουμε ως παράμετρο με την HTTP αίτηση στον διακομιστή είναι οι εξής:

Μέθοδοι			
GET	PUT	POST	DELETE
Επιστέφει μία λίστα, πιθανώς μαζί με άλλες πληροφορίες με όλη τη συλλογή των πόρων.	Αντικαθιστά ολόκληρη τη συλλογή των πόρων με μία άλλη.	Δημιουργεί έναν καινούργιο πόρο (resource) και τον προσθέτει στην συλλογή με τους άλλους πόρους.	Διαγράφει ολόκληρη την συλλογή με τους πόρους.

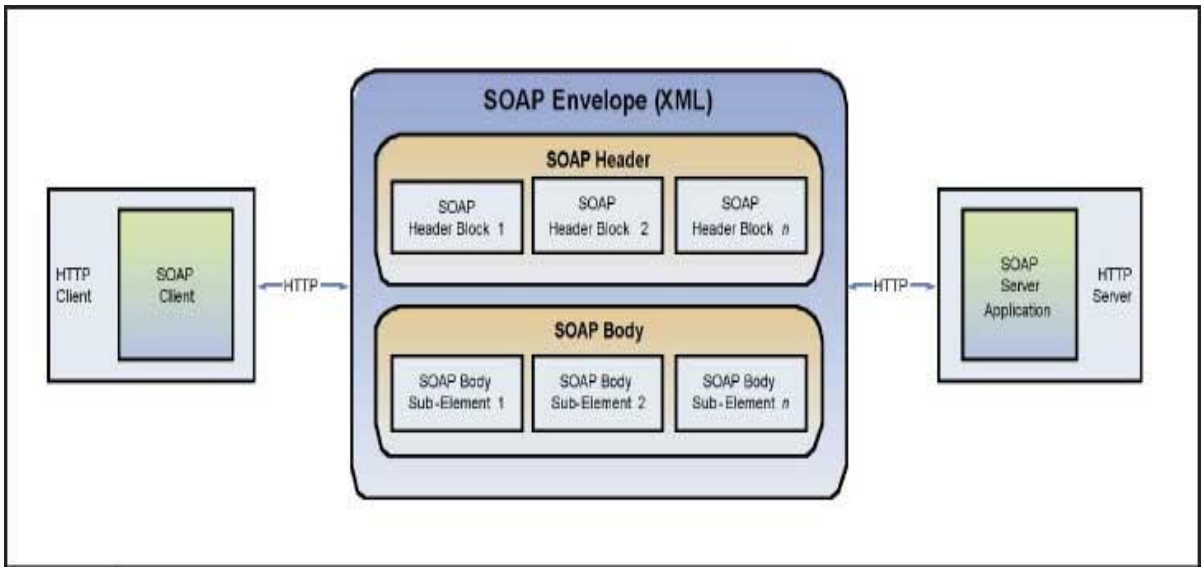
Τέλος αν θέλουμε να στείλουμε μία αίτηση στο uri:
http://example.com/resources/item17, δηλαδή σε έναν συγκεκριμένο πόρο της συλλογής πόρων (resources), θα είχαμε τις παρακάτω επιλογές/μεθόδους:

Μέθοδοι			
GET	PUT	POST	DELETE
Επιστέφει τις πληροφορίες του συγκεκριμένου πόρου (item17).	Αντικαθιστά τον συγκεκριμένο πόρο ή αν δεν υπάρχει τον δημιουργεί.	Δε χρησιμοποιείται γενικά για λειτουργίες σε συγκεκριμένους πόρους	Διαγράφει τον συγκεκριμένο πόρο που του προσδιορίσαμε (item17)

1.4. Διαφορές μεταξύ REST και SOAP αρχιτεκτονικής

Η REST αρχιτεκτονική χρησιμοποιείται από το νέφος αποθήκευσης και όλες οι λειτουργίες που εκθέτει γίνονται μέσω ενός API.
Όμως η εφαρμογή η οποία θα βρίσκεται στον τοπικό διακομιστή (η οποία είναι το βασικό κομμάτι υλοποίησης της παρούσας διπλωματικής εργασίας) ακολουθεί ένα πρότυπο συνδυαστικό. Χρησιμοποιήθηκαν HttpServlets τα οποία ακούνε μόνο σε GET ή POST http methods. Δεν είναι δηλαδή μία εξ ολοκλήρου REST εφαρμογή.
Η εφαρμογή χρησιμοποιεί λειτουργίες της REST αρχιτεκτονικής γιατί είναι πιο γρήγορη και πιο αποτελεσματική για την παρούσα χρήση. Τα πράγματα του διαδικτύου των πραγμάτων θα είναι πάρα πολλά και τα δεδομένα θα στέλνονται συνεχώς. Οπότε η ταχύτητα και η ευελιξία που παρέχει η REST είναι πολύ σημαντικά.
Η ενότητα έχει σκοπό να εξηγήσει το SOAP πρωτόκολλο για να γίνει κατανοητός ο λόγος που επιλέχθηκε η REST.

Γρήγορη επισκόπιση του SOAP (Simple Object Access Protocol)



Εικόνα 4.3 SOAP protocol

Το SOAP βασίζεται αποκλειστικά σε XML για να παρέχει υπηρεσίες μηνυμάτων. Το Microsoft SOAP αναπτύχθηκε αρχικά για να πάρει τη θέση των παλαιότερων τεχνολογιών που δεν λειτουργούν καλά στο διαδίκτυο, όπως η Distributed Component Object Model (DCOM) και η κοινής Αίτησης Αντικείμενου Broker αρχιτεκτονική (CORBA).

Οι τεχνολογίες αυτές αποτυγχάνουν, επειδή βασίζονται στα δυαδικά μηνύματα. Τα XML μηνύματα που η SOAP χρησιμοποιεί λειτουργούν καλύτερα μέσω του Διαδικτύου.
Μετά από μια αρχική έκδοση, η Microsoft υπέβαλε το SOAP στο Engineering Task Force Διαδικτύου (IETF), όπου είχε σταθεροποιηθεί. Το SOAP έχει σχεδιαστεί για να υποστηρίξει την επέκταση, έτσι ώστε να έχει όλα τα είδη των άλλων ακρωνύμιων και συντομογραφιών που σχετίζονται με αυτό, όπως το WS-Addressing, WS-Πολιτική, WS-Security, WS-Federation, WS-ReliableMessaging, WS-Συντονισμός, WS-AtomicTransaction και WS-Remote Portlets. Στην πραγματικότητα, μπορεί να βρεθεί ένας ολόκληρος κατάλογος των προτύπων αυτών στα Web Πρότυπα Υπηρεσιών.
Το θέμα είναι ότι το SOAP είναι εξαιρετικά επεκτάσιμο, αλλά μπορούν να χρησιμοποιηθούν μόνο τα κομμάτια που χρειάζονται για μια συγκεκριμένη εργασία. Για παράδειγμα, όταν χρησιμοποιείται μια δημόσια υπηρεσία Web που είναι ελεύθερα διαθέσιμη σε όλους, δεν έχει πολύ ανάγκη για WS-Security.
Η XML χρησιμοποιείται για να κάνει τις αιτήσεις και τη λήψη απαντήσεων σε SOAP πολύ πιο εύκολο. Σε κάποιες γλώσσες προγραμματισμού, θα πρέπει να οικοδομήσουμε τα αιτήματα αυτά με το χέρι, το οποίο καθίσταται προβληματικό, επειδή το SOAP δεν είναι ανεκτικό σε λάθη.
Μέρος της μαγείας είναι η Web Services Description Language (WSDL). Αυτό είναι ένα άλλο αρχείο που είναι συνδεδεμένο με το SOAP. Παρέχει έναν ορισμό για το πώς λειτουργεί η υπηρεσία Web, έτσι ώστε όταν δημιουργείται μια αναφορά σε αυτό, το IDE να μπορεί να αυτοματοποιήσει πλήρως τη διαδικασία. Έτσι, η δυσκολία χρησιμοποιώντας SOAP εξαρτάται σε μεγάλο βαθμό από τη γλώσσα που χρησιμοποιείται.
Ένα από τα πιο σημαντικά χαρακτηριστικά του SOAP είναι η ενσωματωμένη αντιμετώπιση των λαθών. Αν υπάρχει κάποιο πρόβλημα με το αίτημα κάποιου, η απάντηση περιέχει πληροφορίες για το λάθος που μπορεί να χρησιμοποιηθεί για να διορθωθεί το πρόβλημα. Δεδομένου ότι μπορεί να μην έχουν στην ιδιοκτησία τους την υπηρεσία Web, το συγκεκριμένο χαρακτηριστικό είναι εξαιρετικά σημαντικό. Η αναφορά σφάλματος παρέχει ακόμη τυποποιημένους κωδικούς, έτσι ώστε να είναι δυνατή η αυτοματοποίηση κάποιων εργασιών χειρισμού σφαλμάτων σε κάποιον κώδικα.
Ένα ενδιαφέρον χαρακτηριστικό του SOAP είναι ότι δεν θα πρέπει απαραίτητα να χρησιμοποιηθεί με το Hypertext Transfer Protocol (HTTP) πρωτόκολλο μεταφορών. Υπάρχει μια προδιαγραφή για τη χρήση SOAP πάνω από το πρωτόκολλο Simple Mail Transfer Protocol (SMTP) και δεν υπάρχει κανένας λόγος για το οποίο να μην μπορεί να το χρησιμοποιηθεί σε σχέση με άλλες μεταφορές. Στην πραγματικότητα, οι προγραμματιστές σε ορισμένες γλώσσες, όπως Python, κάνουν ακριβώς αυτό.

Βασικές διαφορές του SOAP και του REST

Το SOAP είναι ένα πρωτόκολλο ανταλλαγής μηνυμάτων με βάση το XML ενώ η REST δεν είναι ένα πρωτόκολλο, αλλά ένα αρχιτεκτονικό στυλ.
Το SOAP έχει μια τυπική προδιαγραφή, αλλά δεν υπάρχει καμία για τη REST.
Υπηρεσίες web, ακόμη και αυτές που βασίζονται σε SOAP μπορούν να εφαρμοστούν σε REST στυλ.
Το SOAP είναι διανεμημένο υπολογιστικό στυλ και η REST είναι στυλ web (web είναι επίσης ένα κατανεμημένο υπολογιστικό μοντέλο).
Τα μηνύματα REST θα πρέπει να είναι αυτόνομα και να βοηθάνε τους καταναλωτές στον έλεγχο της αλληλεπίδρασης μεταξύ παρόχου και καταναλωτή (π.χ. συνδέσεις στο μήνυμα για να αποφασίσει το επόμενο σχέδιο δράσης). Αλλά το SOAP δεν έχει τέτοιες απαιτήσεις.
Η REST δεν επιβάλλει μορφή μηνύματος όπως XML ή JSON κλπ. Αλλά το SOAP επιβάλλει την χρήση του XML πρωτόκολλου μηνύματος.
Η REST ακολουθεί stateless μοντέλο. Το SOAP έχει προδιαγραφές για stateful υλοποίηση.
Το SOAP έχει αυστηρές προδιαγραφές για κάθε τμήμα της εφαρμογής του, αλλά η REST είναι λιγότερο περιοριστική για την εφαρμογή.
Ως εκ τούτου, οι εφαρμογές βασισμένες στην REST είναι απλές σε σύγκριση με τις εφαρμογές σε SOAP.
Το SOAP χρησιμοποιεί διεπαφές και διεργασίες για να εκθέσει την επιχειρηματική λογική. Η REST χρησιμοποιεί (γενικά) URI και μεθόδους, όπως (GET, PUT, POST, DELETE) για να εκθέσει τους πόρους.
Το SOAP έχει ένα σύνολο συγκεκριμένων προδιαγραφών. Το WS-Security είναι η προδιαγραφή για την ασφάλεια κατά την εφαρμογή. Πρόκειται για ένα λεπτομερές πρότυπο που παρέχει τους κανόνες για την ασφάλεια στην υλοποίηση εφαρμογών. Με τον ίδιο τρόπο έχουμε ξεχωριστές προδιαγραφές για την ανταλλαγή μηνυμάτων, τις συναλλαγές, κλπ. Σε αντίθεση με το SOAP, η REST δεν έχει αφιερώσει έννοιες για καθεμία από αυτές. Η REST βασίζεται κυρίως σε HTTPS. Πάνω απ 'όλα όμως και το SOAP και η REST εξαρτώνται από τον σχεδιασμό και την υλοποίηση της εφαρμογής.

Κεφάλαιο 2: Νέφος αποθήκευσης SWIFT Openstack

2.1. Γενικά για τα νέφη αποθήκευσης αντικειμένων

Αρχιτεκτονική αποθηκευτικού νέφους

Η αρχιτεκτονική ενός αποθηκευτικού νέφους έχει ως κύριο στόχο την παράδοση χώρου αποθήκευσης κατόπιν ζήτησης με τέτοιον τρόπο ώστε να επιτυγχάνεται υψηλή κλιμάκωση, διαχείριση και αυτοσυντήρηση. Γενικά, η αρχιτεκτονική αποτελείται από το front-end το οποίο εξάγει ένα API για πρόσβαση στις λειτουργίες του χώρου αποθήκευσης και των λειτουργιών που προσφέρονται. Σε ένα τυπικό αποθηκευτικό σύστημα διαχείρισης αρχείων, το API αυτό θα ήταν το πρωτόκολλο SCSI όμως σε αποθηκευτικά νέφη τα πρωτόκολλα συνεχώς εξελίσσονται, ώστε να βλέπουμε ακόμη και διαδικτυακές υπηρεσίες ως front-end. Πίσω από το front-end βρίσκουμε το επίπεδο του middleware το οποίο συχνά ονομάζουμε και “storage logic”. Στο επίπεδο αυτό βρίσκουμε πάρα πολλά χαρακτηριστικά όπως η αναπαραγωγή των δεδομένων (backups ασφαλείας), η συμπίεση των δεδομένων για μείωση του χώρου που καταλαμβάνουν και η (γεωγραφική) επιλογή της τοποθεσίας που θα αποθηκευτούν τα δεδομένα. Τέλος, πίσω από το middleware, βρίσκουμε το back-end που υλοποιεί τα χαρακτηριστικά εκείνα που είναι αναγκαία για τη φυσική αποθήκευση των δεδομένων.
Παρακάτω περιγράφονται κάποια από τα σημαντικότερα χαρακτηριστικά που πρέπει να διαθέτει ένα αποθηκευτικό νέφος[4]:

Χαρακτηριστικά αποθηκευτικού νέφους

Διαχειρισιμότητα: Ίσως το πιο σημαντικό πράγμα που λαμβάνει μια εταιρία υπόψην πριν επενδύσει σε αποθηκευτικό νέφος είναι το κόστος. Αν μια εταιρία μπορεί να αγοράσει και να διαχειρίζεται τα δεδομένα της τοπικά πολύ πιο φθηνά από ότι σε νέφος, τότε δεν υπάρχει λόγος να επιζήσει η αγορά αποθηκευτικών νεφών. Το κόστος χωρίζεται σε δυο παράγοντες. Ο πρώτος είναι η αγορά του εξοπλισμού και ο δεύτερος είναι η διαχείριση των πόρων του συστήματος στη οποία δεν μπορούμε να αποδώσουμε μια τιμή εξ αρχής. Για το λόγο αυτό θέλουμε να επιτύχουμε την

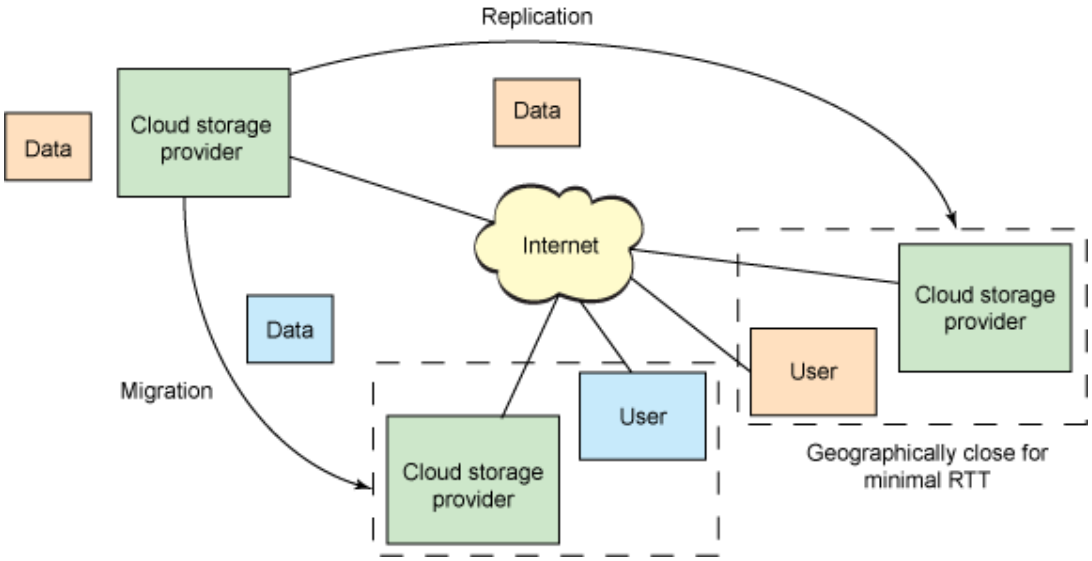
αυτό-διαχείριση στο μεγαλύτερο ποσοστό που μπορούμε. Σε αποθηκευτικά νέφη έχουμε τη δυνατότητα όταν χρειαζόμαστε (on-demand) να προσθέτουμε μεγαλύτερο χώρο αποθήκευσης χωρίς να αγοράζουμε τον φυσικό εξοπλισμό και το σημαντικότερο χωρίς να νοιαζόμαστε για τη συμβατότητα.

Μέθοδος πρόσβασης: Η μέθοδος πρόσβασης στα δεδομένα αποτελεί μια από τις μεγαλύτερες διαφορές μεταξύ αποθηκευτικού νέφους και παραδοσιακές μεθόδους αποθήκευσης. Οι περισσότεροι πάροχοι υπηρεσιών αποθήκευσης σε νέφος υλοποιούν πολλαπλές μεθόδους πρόσβασης όμως είναι πολύ διαδεδομένα τα διαδικτυακά APIs. Τα περισσότερα APIs είναι υλοποιημένα με βάση τη αρχιτεκτονική REST και τους κανόνες της, στη οποία έχουμε ένα σχήμα με βάση αντικείμενα (object-based schema) και η αρχιτεκτονική είναι υλοποιημένη πάνω στο πρωτόκολλο HTTP. Τα "RESTful" APIs είναι άνευ κατάστασης (stateless) έτσι παραμένουν απλά στη χρήση και αποδοτικά. Οι πιο γνωστοί πάροχοι υπηρεσιών αποθήκευσης που χρησιμοποιούν RESTful API είναι το Amazon Simple Storage Service, Windows Azure και το Mezeo Cloud Storage Platform. Άλλοι μέθοδοι πρόσβασης που χρησιμοποιούνται από πάροχους όπως Nirvanix, Zetta και η Cleversafe χρησιμοποιούν file-based πρωτόκολλα όπως το πρωτόκολλο NFS ή το CIFS ή ακόμη και FTP ενώ συχνά χρησιμοποιείται και το iSCSI που είναι block-based. Τ έλος ένα πολύ ενδιαφέρον πρωτόκολλο είναι το WebDAV το οποίο βασίζεται στο HTTP και θεωρεί το διαδίκτυο ως πόρος για διάβασμα και εγγραφή.

Απόδοση: Υπάρχουν αρκετοί παράγοντες που επηρεάζουν την απόδοση. Όμως η ικανότητα να μεταφέρονται δεδομένα από και προς τον χρήστη στο αποθηκευτικό νέφος που δεν βρίσκεται τοπικά είναι η μεγαλύτερη πρόκληση για τους παρόχους υπηρεσιών. Το πρόβλημα, το οποίο είναι και πρόβλημα του διαδικτύου, έχει να κάνει με το πρωτόκολλο μεταφοράς TCP. Το TCP είναι το πρωτόκολλο που ελέγχει τη μεταφορά δεδομένων και βασίζεται σε πακέτα αναγνώρισης. Η απώλεια πακέτων και η καθυστερημένη άφιξη είναι αιτίες συμφόρησης οι οποίες επηρεάζουν την απόδοση του δικτύου. Το TCP είναι ιδανικό για τη μεταφορά μικρών σε μέγεθος πακέτων, όμως για τη διακίνηση μεγάλου μεγέθους δεδομένων, αυξάνεται κατακόρυφα το round trip time (RTT). Η Amazon σε μια προσπάθεια επίλυσης του προβλήματος της συμφόρησης έχει να προτείνει ένα νέο πρωτόκολλο, το FASP, το οποίο αυξάνει τη μεταφορά δεδομένων και αντιμετωπίζει το μεγάλο RTT χρησιμοποιώντας το πρωτόκολλο UDP στο επίπεδο μεταφοράς και μεταφέροντας τον έλεγχο της συμφόρησης στο επίπεδο εφαρμογών όπου λειτουργεί το FASP.

Multi-tenancy: Είναι ένα από τα κυριότερα χαρακτηριστικά των αποθηκευτικών νεφών. Στον χώρο αποθήκευσης μπορούν να έχουν πρόσβαση αρκετοί χρήστες. Σε multi-tenancy περιβάλλον μόνο ένα φυσικό στιγμιότυπο μιας εφαρμογής υπάρχει και εξυπηρετεί πολλαπλούς χρήστες και οργανισμούς. Η λογική αυτή βασίζεται στο ότι ο κάθε χρήστης και οργανισμός χρησιμοποιεί ένα εικονικό τμήμα της εφαρμογής και των δεδομένων και είναι ειδικά ρυθμισμένο στις ανάγκες τους στο τμήμα αυτό. Τέλος το multi-tenancy βρίσκει εφαρμογή και στο επίπεδο του δικτύου, με το εύρος ζώνης που δίνεται σε κάθε χρήστη για πρόσβαση στο νέφος και μεταφορά των δεδομένων του στο νέφος, να παίζει σημαντικό ρόλο στη ποιότητα των υπηρεσιών που προσφέρει ο πάροχος.

Κλιμάκωση: Είναι και αυτή ένα από τα σημαντικότερα στοιχεία σε αποθηκευτικά νέφη. Αυτό που κάνει πολύ σημαντική την κλιμάκωση είναι η δυνατότητα, κατόπιν ζήτησης, να αυξομειώνουμε ανάλογα με τις ανάγκες μας τον χώρο αποθήκευσης μειώνοντας το κόστος στο ελάχιστο. Έτσι η κοστολόγηση γίνεται μόνο για τους δεσμευμένους πόρους. Στον αντίποδα όμως, είναι η αύξηση της πολυπλοκότητας του αποθηκευτικού νέφους για τον πάροχο που πρέπει να μεριμνήσει για την κλιμάκωση. Η κλιμάκωση δεν πρέπει να γίνεται μόνο στην παροχή επιπλέον χώρου αποθήκευσης αλλά και στο εύρος ζώνης, στη φόρτωση και κατέβασμα των δεδομένων αλλά και στη κατανομή (γεωγραφικά) των δεδομένων έτσι ώστε να βρίσκονται πάντα κοντά στον χρήστη.



Εικόνα 2.1. Κλιμάκωση αποθηκευτικού νέφους

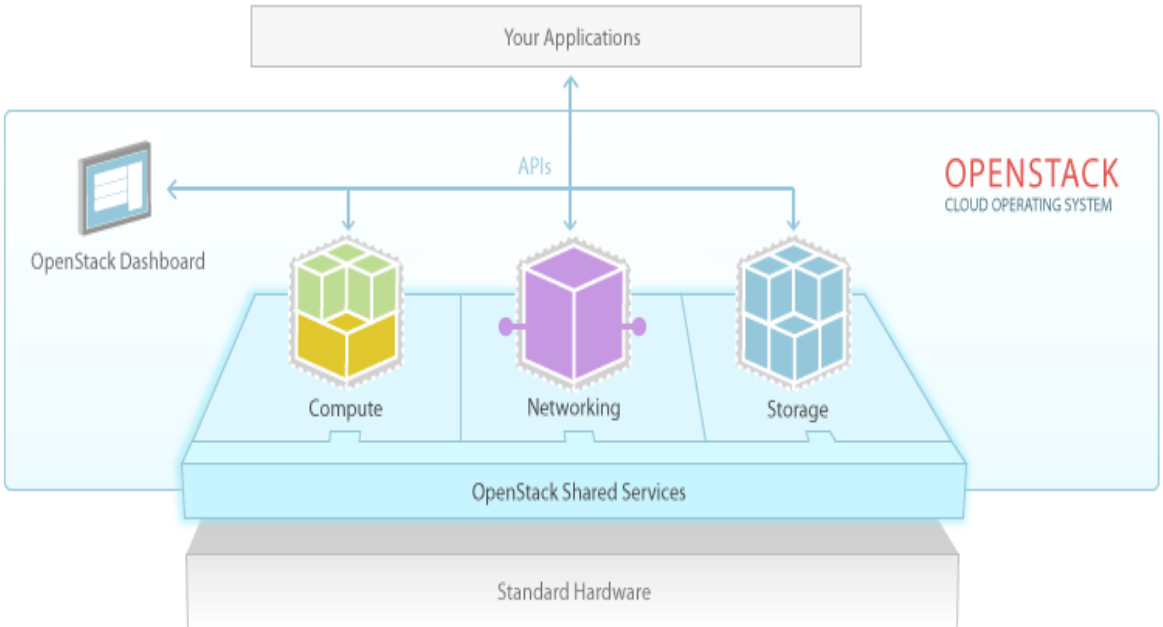
Διαθεσιμότητα: Είναι πολύ σημαντικό, όταν ο χρήστης ανεβάσει στο νέφος τα δεδομένα του, έπειτα να μπορεί να τα έχει πάντα διαθέσιμα κατόπιν ζήτησης. Προβλήματα στο δίκτυο, σφάλματα στην πλευρά του πελάτη και φυσικές απώλειες, όπως απώλειες σκληρών δίσκων είναι μερικά προβλήματα που πρέπει

να αντιμετωπίσουν οι πάροχοι έτσι ώστε να προσφέρουν αξιόπιστες υπηρεσίες και να εγγυηθούν τα δεδομένα του χρήστη. Για τον λόγο αυτό οι πάροχοι έχουν αναπτύξει διάφορες μεθόδους για αντιμετώπιση απωλειών όπως ο αλγόριθμος IDA[15], όπου πολύ παρόμοια με το RAID μπορούν να ανακατασκευαστούν τα δεδομένα από ένα υποσύνολό τους.

2.2. Το νέφος αποθήκευσης Openstack

Εξυπηρετητής Νέφους αποθήκευσης αντικειμένων (Cloud Object Storage Server)

Από τις πολλές εφαρμογές της υπολογιστικότητας νέφους, εμείς θα ασχοληθούμε με το νέφος αποθήκευσης και συγκεκριμένα με το νέφος αποθήκευσης αντικειμένων. Το λογισμικό που χρησιμοποιήθηκε για την εργασία αυτή είναι το ανοιχτού κώδικα λογισμικό Openstack για εξυπηρετητές νέφους. [4]
Το openstack παρέχει το κατάλληλο λογισμικό για την ανάπτυξη, το στήσιμο, τη λειτουργία και την επικοινωνία των υπολογιστικών πόρων και του χώρου αποθήκευσης με το δίκτυο και έναν πίνακα ελέγχου για τις λειτουργίες όπως φαίνεται παρακάτω:

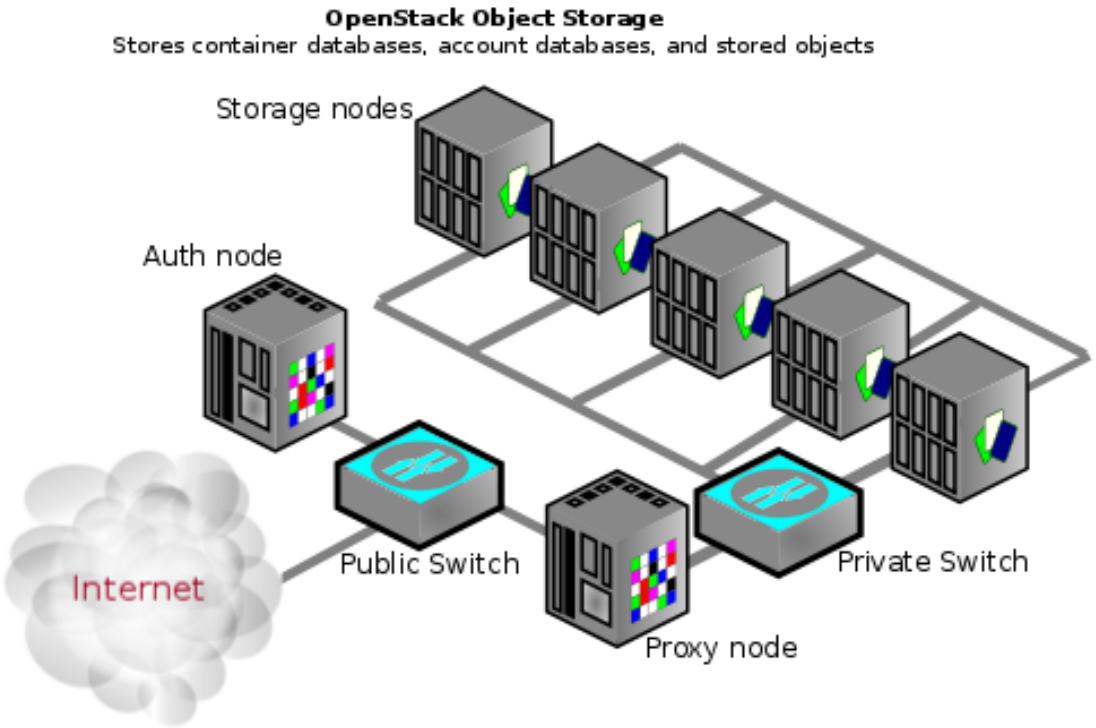


Εικόνα 2.2.1 Openstack cloud operating system

Στη συγκεκριμένη εργασία χρησιμοποιήθηκε η πλατφόρμα SAIO, Swift all in one (όλα σε ένα) του Openstack για να στήθει σε μία εικονική μηχανή το σύνολο των κόμβων που χρειάζονται ώστε να λειτουργήσει ένας εξυπηρετητής νέφους αποθήκευσης αντικειμένων (Cloud object storage server). Η επικοινωνία με τον εξυπηρετητή αποθήκευσης γίνεται με ένα API (Διασύνδεση προγραμματισμού εφαρμογών), βασισμένο στην αρχιτεκτονική REST (παραστατική κατάσταση μεταφοράς), η οποία θα αναλυθεί παρακάτω.
Συγκεκριμένα πάνω στην εικονική μηχανή στήθηκαν:

- ένας κόμβος ελέγχου ταυτότητας (authentication node)
- ένας κόμβος διακομιστή μεσολάβησης (proxy node)
- ένας κόμβος αποθήκευσης (storage node)

Ο κόμβος διακομιστή μεσολάβησης τρέχει τις υπηρεσίες διαμεσολάβησης και επικοινωνίας μεταξύ των άλλων κόμβων ενώ ο κόμβος αποθήκευσης τρέχει τις υπηρεσίες του λογαριασμού χρήστη (account services), των δοχείων των αντικειμένων (container services) και των αντικειμένων (object services). Ο κόμβος ελέγχου ταυτότητας είναι υπεύθυνος για τον έλεχγο των διακριτικών πιστοποίησης (authentication tokens) και την έκδοσή τους. Το δίκτυο έχει την εξής μορφή:



Εικόνα 2.2.2 Openstack object storage

Με την διαφορά πως όλοι οι κομβοι βρίσκονται στην εικονική μηχανή.

2.3. Βήματα εγκατάστασης του SWIFT σε απομακρυσμένη εικονική μηχανή

Στήσιμο εικονικής μηχανής και λειτουργικού

Αρχικά έγινε εγγραφή σε μία υπηρεσία νέφους παροχής εικονικών μηχανών με μοναδικό IP έτσι ώστε να μπορεί το νέφος αποθήκευσης να είναι προσβάσιμο από οποιοδήποτε μηχάνημα συνδεδεμένο στο διαδίκτυο.
Αφού έγινε η εγγραφή, στη συνέχεια έφτιαξα μία εικονική μηχανή με επεξεργαστή δύο πυρήνες (dual core) και 6GB Ram. Η χωρητικότητα της εικονικής μηχανής είναι 100GB.
Για λειτουργικό σύστημα επιλέχθηκαν τα Ubuntu 12.04 LTS (Precise Pangolin) Server όπως υποδεικνύει και ο οδηγός εγκατάστασης του SAIO – Swift All In One. [6]
Τα στατικά IP που δώθηκαν σε αυτήν την εικονική μηχανή είναι τα:

Public IPv4 Network	83.212.121.116
Public IPv6 Network	2001:648:2ffc:1225:a800:ff:feda:2685

Το αποτέλεσμα μέτα την ολοκλήρωση δημιουργίας της εικονικής μηχανής φαίνεται στην παρακάτων εικόνα στην οποία εικονίζεται η διεπαφή διαχείρησης της εικονικής αυτής μηχανής.

Εικόνα 2.3 Okeanos vm control

Το όνομα χρήστη (username) του λειτουργικού συστήματος που δώθηκε είναι το user και το όνομα της εικονικής μηχανής είναι: IoT_Cloud_Server

Στήσιμο του Swift All In One object server στην εικονική μηχανή IoT_Cloud_Server

Αφού συνδέθηκα στην εικονική μηχανή μέσω του ssh address: snf-219455.vm.okeanos.grnet.gr, ξεκίνησα εγκαθιστώντας τις απαιτήσεις συστήματος, πληκτρολογώντας εντολές για apt συστήματα. Το apt (advanced packaging tool) είναι ένα σύνολο εργαλείων το οποίο δουλεύει με τις βασικές βιβλιοθήκες ενός λειτουργικού συστήματος και βοηθάει στην εγκατάσταση και απεγκατάσταση λογισμικού σε αυτό. [7]

Οι εντολές είναι οι εξής:

[6]

```
sudo apt-get update
sudo apt-get install curl gcc memcached rsync sqlite3 xfsprogs\
                     git-core libffi-dev python-setuptools
sudo apt-get install python-coverage python-dev python-nose \
                     python-simplejson python-xattr python-
eventlet \
                     python-greenlet python-pastedeploy \
                     python-netifaces python-pip python-
dnspython \
                     python-mock
```

Στην συνέχεια, έπρεπε να επιλεγεί ο τρόπος με τον οποίο θα διαχωριστεί ο αποθηκευτικός χώρος της εικονικής μηχανής, έτσι ώστε να εγκατασταθούν οι κόμβοι του νέφους αποθήκευσης αντικειμένων (ο κόμβος ελέγχου ταυτότητας (authentication node), ο κόμβος διακομιστή μεσολάβησης (proxy node) και ο κόμβος αποθήκευσης (storage node).
Οι δύο πιθανές επιλογές ήταν να χρησιμοποιήσουμε διαμερίσματα για τον αποθηκευτικό χώρο (Using a partition for storage) και η άλλη ήταν να χρησιμοποιήσουμε μία συσκευή βρόχου επιστροφής για τον αποθηκευτικό χώρο (Using a loopback device for storage).
Από τις δύο αυτές επιλογές επιλέχθηκε η χρησιμοποίηση μίας συσκευής βρόχου επιστροφής καθώς για την $1^{η}$ επιλογή θα έπρεπε να διαμερίσουμε τον χώρο κατά τη δημιουργία της εικονικής μηχανής. Εφόσον όμως δεν είχαμε τη δυνατότητα ελέγχου της δημιουργίας της εικονικής μηχανής, (εφόσον ήταν αυτοματοποιημένη από την υπηρεσία νέφους παροχής εικονικών μηχανών) η $1^{η}$ επιλογή απορρίφθηκε.

Για την δημιουργία της συσκευής βρόχου επιστροφής έτρεξα τις εξής εντολές:

- Δημιουργία του αρχείου για την συσκευή:

```
sudo mkdir /srv
sudo truncate -s 1GB /srv/swift-disk
sudo mkfs.xfs /srv/swift-disk
```

- Επεξεργασία του αρχείου */etc/fstab:*

```
/srv/swift-disk /mnt/sdb1 xfs
loop,noatime,nodiratime,nobarrier,logbufs=8 0 0
```

- Δημιουργία σημείου προσάρτισης:

```
sudo mkdir /mnt/sdb1
sudo mount /mnt/sdb1
sudo mkdir /mnt/sdb1/1 /mnt/sdb1/2 /mnt/sdb1/3 /mnt/sdb1/4
sudo chown ${USER}:${USER} /mnt/sdb1/*
for x in {1..4}; do sudo ln -s /mnt/sdb1/$x /srv/$x; done
sudo mkdir -p /srv/1/node/sdb1 /srv/2/node/sdb2
/srv/3/node/sdb3 /srv/4/node/sdb4 /var/run/swift
sudo chown -R ${USER}:${USER} /var/run/swift
# **Make sure to include the trailing slash after /srv/$x/**
for x in {1..4}; do sudo chown -R ${USER}:${USER} /srv/$x/;
done
```

Στην συνέχεια πήρα τον κώδικα του swift, έκανα την εγκατάσταση του swift και των εξαρτήσεων του:

```
git clone https://github.com/openstack/swift.git
cd $HOME/swift; sudo python setup.py develop; cd -
sudo pip install -r swift/test-requirements.txt
```

Μετά δημιούργησα το rsync.conf αρχείο που περιέχει τις ρυθμίσεις των κόμβων, το ενεργοποίησα και το ξενικάμε ώστε να συγχρονίζονται οι κόμβοι μεταξύ τους:

```
sudo cp $HOME/swift/doc/saio/rsyncd.conf /etc/
sudo sed -i "s/user/${USER}/" /etc/rsyncd.conf
RSYNC_ENABLE=true
sudo service rsync restart
rsync rsync://pub@localhost/
```

Τέλος με τις παρακάτω εντολές φτιάχνονται τα αρχεία ρυθμίσεων του κόμβου διακομιστή μεσολάβησης (proxy-server.conf), του κόμβου διακομιστή λογαριασμού χρήστη (account-server), του κόμβου διακομιστή δοχείου (container-server) και του κόμβου διακομιστή αντικειμένου (object-server).

```
sudo rm -rf /etc/swift
cd $HOME/swift/doc; sudo cp -r saio/swift /etc/swift; cd -
sudo chown -R ${USER}:${USER} /etc/swift
find /etc/swift/ -name \*.conf | xargs sudo sed -i
"s/user/${USER}/"
```

2.4. Έλεγχος σωστής λειτουργίας του νέφους αποθήκευσης μετά την εγκατάσταση.

Για να ελέγξουμε ότι λετουργεί το νέφος αποθήκευσης που μόλις εγκαταστάθηκε τρέχουμε τις εξής εντολές με την σειρά:

- Πέρνουμε ένα X-Storage-Url και ένα X-Auth-Token:

```
curl -v -H 'X-Storage-User: test:tester' -H 'X-Storage-Pass:
testing' http://127.0.0.1:8080/auth/v1.0
```

- Στέλνουμε ένα HTTP μήνυμα τύπου GET για να πάρουμε τις πληροφορίες του λογαριασμού χρήστη:

```
curl -v -H 'X-Auth-Token: <token-from-x-auth-token-above>'
<url-from-x-storage-url-above>
```

Από την πρώτη εντολή πρέπει να επιστραφεί ένα μήνυμα HTTP το οποίο θα έχει μέσα στις επικεφαλίδες του, μεταξύ άλλων, μία τιμή για το X-Storage-User και μία τιμή για το X-Auth-Token. Το X-Auth-Token θα πρέπει να το στέλνουμε κάθε φορά στον διακομιστή του νέφους που βρίσκεται στο url: X-Storage-Url σαν επικεφαλίδες ενός HTTP μηνύματος για να κάνουμε τις αλληλεπιδράσεις με το νέφος.
Με τη δεύτερη εντολή ζητάμε τις πληροφορίες του λογαριασμού χρήστη.
Οι αλληλεπιδράσεις με το νέφος και ο τρόπος που γίνονται θα εξηγηθεί πιο αναλυτικά σε επόμενο κεφάλαιο.

2.5. Οργάνωση και λειτουργία του νέφους αποθήκευσης αντικειμένων ως RESTful υπηρεσία

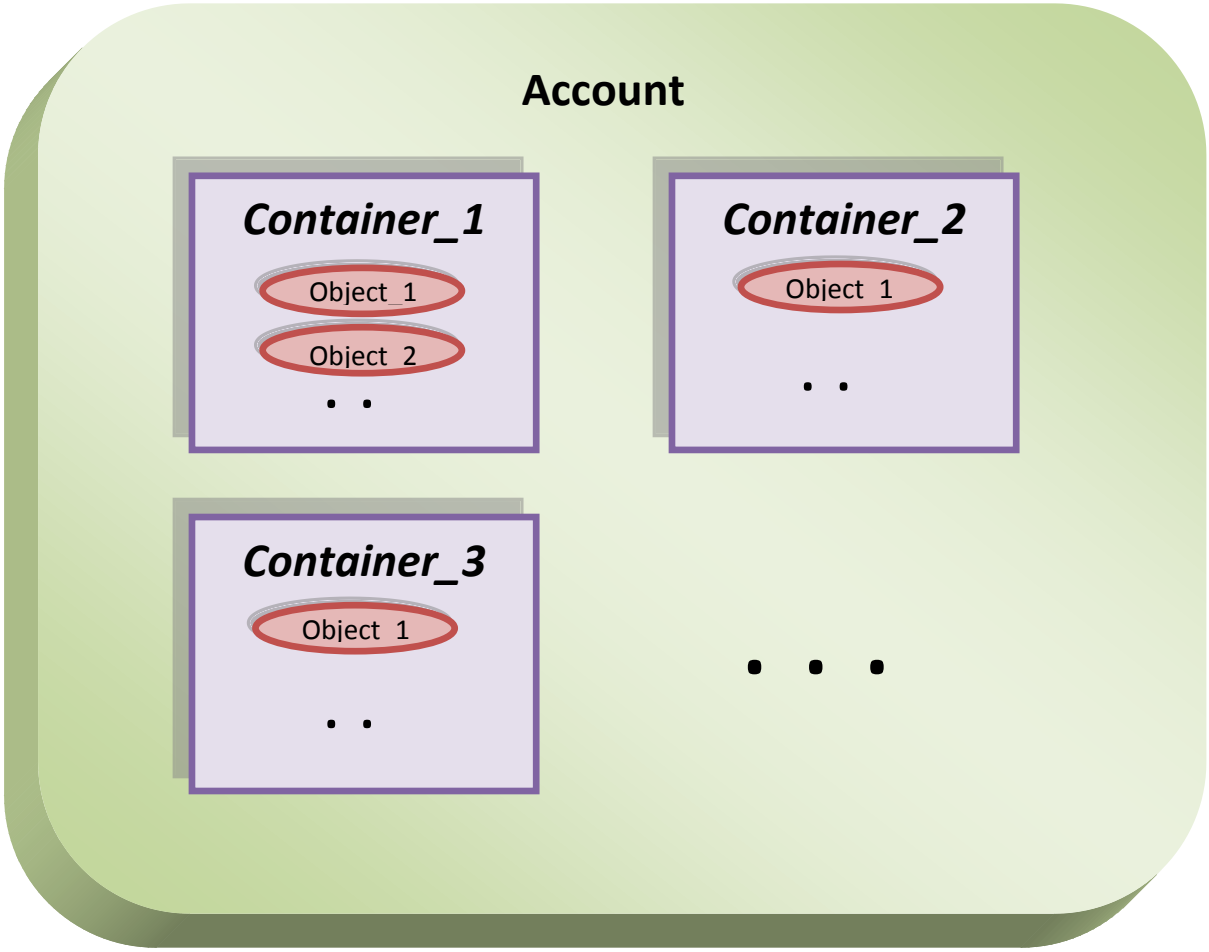
Γενικά για το νέφος και το REST

Όπως έχει αναφερθεί στην προηγούμενη ενότητα το νέφος αποθήκευσης αντικειμένων Swift της Openstack χρησιμοποιεί το πρωτόκολλο HTTP για την μεταφορά εντολών και δεδομένων από και προς αυτό.
Συγκεκριμένα χρησιμοποιεί τις μεθόδους PUT, POST, DELETE, GET, HEAD για τις βασικές λειτουργίες του. Η μέθοδος HEAD δεν αναφέρθηκε νωρίτερα, καθώς δεν είναι μία από τις συνηθησμένες μεθόδους του HTTP αλλά θα εξηγηθεί η χρησιμότητά της στο νέφος παρακάτω.
Ο βασικός λόγος για τον οποίο επιλέχθηκε το νέφος αποθήκευσης αντικειμένων της Openstack είναι λόγο της REST ιδιότητας που έχει. Χάρη σε αυτήν μπορούμε όχι μόνο να αποθηκεύουμε πληροφορίες πολύ γρήγορα και εύκολα, αλλά και να αποθηκεύουμε εκτός από τα κύρια δεδομένα της πληροφορίας και κάποια δευτερεύοντα σαν μεταδεδομένα. Με αυτόν τον τρόπο γίνεται πολύ πιο εύκολη η αναζήτηση μέσω αυτών των “ετικετών” που θα έχει το κάθε αντικείμενο. Έτσι θα ξεχωρίζει από τα άλλα και θα είναι εύκολα αναζητήσιμο. Στο επόμενο κεφάλαιο που θα αναλυθεί το διαδίκτυο των πραγμάτων, θα γίνει ευκολότερα κατανοητό γιατί είναι τόσο σημαντική η γρήγορη αναζήτηση δεδομένων.

Οργάνωση του νέφους αποθήκευσης Swift της Openstack

Όπως φαίνεται στο παρακάτω σχήμα, το Swift έχει οργανώσει το νέφος αποθήκευσης σε “δοχεία”. Αρχικά έχουμε το account το οποίο αναφέρεται στον λογαριασμό χρήστη, δηλαδή αυτόν που εγκατέστησε το νέφος. Ρυθμίζεται δηλαδή κατά την εγκατάσταση, αλλά στην συνέχεια μπορούν να δημιουργηθούν και άλλα accounts.

Μέσα στο account βρίσκονται τα containers, δηλαδή οι υποδοχείς αντικειμένων. Μπορούμε να έχουμε όσα containers θέλουμε. Το κάθε container μπορεί να έχει τα δικά του μεταδεδομένα, όπως π.χ. το αναγνωριστικό του (id:1). Τα μεταδεδομένα είναι της μορφής: “key” : “value” .
Μέσα σε κάθε container μπορούμε να έχουμε όσα αντικείμενα θέλουμε. Το κάθε αντικείμενο μπορεί να έχει τα δικά του μεταδεδομένα όπως π.χ. τον δημιουργό του (creator:Nick).
Τώρα γίνεται εμφανές σιγά σιγά πως τα μεταδεδομένα μπορούν να βοηθήσουν στην ευκολότερη αναζήτηση αντικειμένων.
Εκτός από τα μεταδεδομένα όμως, κάθε αντικείμενο έχει και το κυρίως σώμα του. Το κυρίως σώμα ενός αντικειμένου περιέχει οποιαδήποτε πληροφορία εμείς επιθυμούμε και την εισάγουμε κατά τη δημιουργία του.



Τρόπος λειτουργίας του REST στο νέφος αποθήκευσης Swift της

Openstack

Στο νέφος αποθήκευσης Swift της Openstack όπως ειπώθηκε προηγουμένως χρησιμοποιούνται οι μέθοδοι HEAD, PUT, POST, DELETE, GET. Στον παρακάτω πίνακα εμφανίζεται ακριβώς τι μπορούμε να κάνουμε χρησιμοποιώντας κάθε μία από αυτές τις μεθόδους στο account, στο container και στο object.
Ας πούμε δηλαδή ότι το νέφος είναι προσβάσιμο από το σύνδεσμο, στον οποίο εγκαταστήσαμε προηγουμένως την εικονική μηχανή. Δηλαδή, το νέφος βρίσκεται στο: 83.212.121.116.
Επομένως οι μεθόδοι των μηνυμάτων HTTP στέλνονται στο: http://83.212.121.116/v1, όπου v1 είναι η έκδοση του εγκατεστημένου νέφους. [8]

- Μέθοδοι που μπορούμε να καλέσουμε στο account. Στην περίπτωση του δικού μας νέφους το account είναι το AUTH_test.

HTTP μέθοδος	URI	Περιγραφή
GET	http://83.212.121.116/v1/AUTH_test	Επιστρέφει όλα τα container που περιέχονται στο account AUTH_test στο HTTP response body.
HEAD	http://83.212.121.116/v1/AUTH_test	Επιστρέφει όλα τα μεταδεδομένα του account σαν HTTP headers.
POST	http://83.212.121.116/v1/AUTH_test	Δημιουργεί ή ενημερώνει τα μεταδεδομένα του container τα οποία έχουμε τοποθετήσει στα HTTP headers του μήνυματός μας.
POST	http://83.212.121.116/v1/AUTH_test	Διαγράφει τα μεταδεδομένα που έχουμε τοποθετήσει στα HTTP headers ως: *X-Remove-Account-Meta-name: arbitrary value*

- Μέθοδοι που μπορούμε να καλέσουμε στο container. Στην περίπτωση του δικού μας νέφους ένα container του account AUTH_test είναι το IoTcontainer. Άρα θα έχουμε τα εξής:

HTTP μέθοδος	URI	Περιγραφή
GET	http://83.212.121.116/v1/AUTH_test/IoTcontainer	Επιστρέφει όλα τα objects που περιέχονται στο container IoTcontainer στο HTTP response body.
HEAD	http://83.212.121.116/v1/AUTH_test/IoTcontainer	Επιστρέφει όλα τα μεταδεδομένα του container σαν HTTP headers.
PUT	http://83.212.121.116/v1/AUTH_test/IoTcontainer	Δημιουργεί ένα καινούργιο container του οποίου το όνομα το βάζουμε στο τέλος του uri. Π.χ.: *http://83.212.121.116/v1/AUTH_test/ new_container*
DELETE	http://83.212.121.116/v1/AUTH_test/IoTcontainer	Διαγράφει το container του οποίου το όνομα τοποθετούμε στο τέλος του uri. Π.χ.: *http://83.212.121.116/v1/AUTH_test/ new_container*

- Μέθοδοι που μπορούμε να καλέσουμε στο object. Στην περίπτωση του δικού μας νέφους ένα object του container IoTcontainer είναι το newobj. Άρα θα έχουμε τα εξής:

HTTP μέθοδος	URI	Περιγραφή
GET	http://83.212.121.116/v1/AUTH_test/IoTcontainer/newobj	Επιστρέφει όλα τα δεδομένα του object newobj μαζί με τα μεταδεδομένα.
HEAD	http://83.212.121.116/v1/AUTH_test/IoTcontainer/newobj	Επιστρέφει όλα τα μεταδεδομένα του object σαν HTTP headers.
POST	http://83.212.121.116/v1/AUTH_test/IoTcontainer/newobj	Ενημερώνει τα μεταδεδομένα του object τα οποία έχουμε τοποθετήσει στα HTTP headers του μήνυματός μας.
DELETE	http://83.212.121.116/v1/AUTH_test/IoTcontainer/newobj	Διαγράφει το object newobj.
PUT	http://83.212.121.116/v1/AUTH_test/IoTcontainer/newobj	Δημιουργεί το object newobj ή το ενημερώνει.
PUT	http://83.212.121.116/v1/AUTH_test/IoTcontainer/newobj	Μπορεί να χρησιμοποιηθεί για θριμματισμένη μεταφορά και δημιουργία ενός object στο νέφος σε περίπτωση που είναι πολύ μεγάλο.

Κεφάλαιο 3: Internet of Things (Διαδίκτυο των Πραγμάτων)

3.1. Γενικά για το Διαδίκτυο των Πραγμάτων (Internet of Things)

Η έκφραση «Διαδίκτυο των Πραγμάτων» (IoT), επινοήθηκε το 1999 από τον Kevin Ashton, Βρετανό πρωτοπόρο της τεχνολογίας που ίδρυσε από κοινού το Auto-ID Center στο Massachusetts Institute of Technology, και γίνεται όλο και πιο δημοφιλής. Ενδεικτικά αναφέρεται ότι η Cisco προβλέπει 50 δισεκατομμύρια συσκευές συνδεδεμένες μέχρι το 2020, δημιουργώντας μία πιθανή αγορά που θα υπερβαίνει το ποσό των 14 τρισεκατομμυρίων δολαρίων καθώς και υποστηρίζει πως το Διαδίκτυο των Πραγμάτων είναι ήδη εδώ και σε λειτουργία. [9]

Εικόνα 3.1 Internet of things

Την ίδια στιγμή, το Διαδίκτυο των Πραγμάτων (IoT) δεν είναι μία εμπειρία η οποία βιώνεται από μόνη της. Αυτό που θα συμβεί και συμβαίνει είναι ότι όλο και περισσότερα αντικείμενα θα συνδέονται με αυτό. Με τον όρο αντικείμενα εννοούμε τα έξυπνα αντικείμενα όπως π.χ. έξυπνα πλυντήρια, ψυγεία, παρκόμετρα, αυτοκίνητα κ.τ.λ. Ακόμα δεν γνωρίζουμε και τις συνέπειες του να τα συνδέσουμε όλα αυτά στο διαδίκτυο. Καθημερινές δραστηριότητες που γίνονταν ξεχωριστά θα διαπλέκονται σε νέες μορφές και νέα επιχειρηματικά μοντέλα. [10]
Έτσι, στην πραγματικότητα, το Διαδίκτυο των Πραγμάτων είναι ένας συνδυασμός της τεχνολογικής ώθησης και της ανθρώπινης έλξης για περισσότερη και διαρκή αυξανόμενη συνδεσιμότητα με οτιδήποτε συμβαίνει στο άμεσο και ευρύτερο περιβάλλον - μια λογική επέκταση της υπολογιστικής δύναμης σε ένα μηχάνημα στο περιβάλλον: το περιβάλλον ως διεπαφή. Αυτός ο συνδιασμός ώθησης-έλξης (push-pull) το καθιστά πολύ ισχυρό, ασταμάτητο, γρήγορο και εξαιρετικά διασπαστικό.

3.2. Εφαρμογές του Διαδικτύου των Πραγμάτων

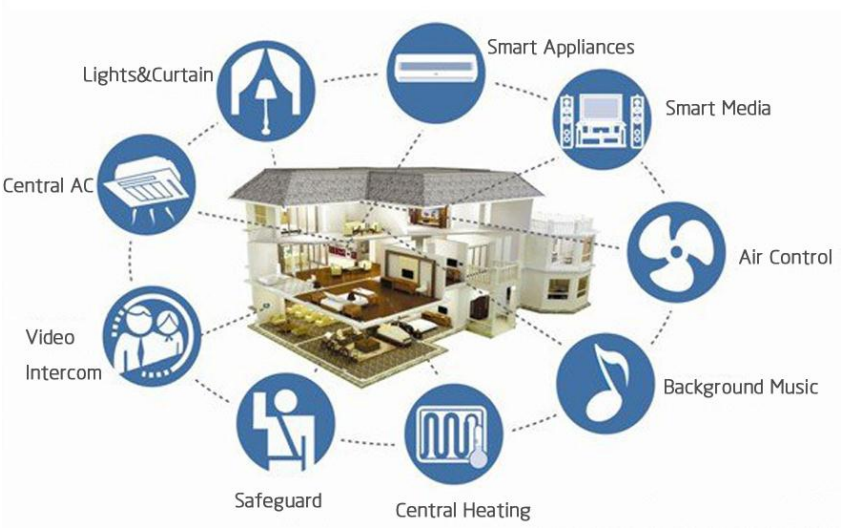
Από τα παρακάτω παραδείγματα γίνεται εμφανής η χρησιμότητα και η δύναμη που έχει το διαδίκτυο των πραγμάτων όπως ακόμα και η ευρεία γκάμα εφαρμογών στις οποίες μπορεί να χρησιμοποιηηθεί.

- **Παράδειγμα 1^{ο}** : Μεταφορές/Εφοδιασμός

Στον εφοδιασμό και τις μεταφορές, το IoT δεν βελτιώνει μόνο τα συστήματα ροής των υλικών, αλλά και το παγκόσμιο σύστημα εντοπισμού στίγματος και την αυτόματη αναγνώριση των εμπορευμάτων. Επίσης, αυξάνει την ενεργειακή απόδοση και έτσι μειώνει την κατανάλωση ενέργειας.
Εν κατακλείδι, το IoT αναμένεται να επιφέρει ριζικές αλλαγές στην παγκόσμια
αλυσίδα εφοδιασμού, μέσω της έξυπνης κίνησης του φορτίου. Αυτό θα επιτευχθεί μέσα από τον συνεχή συγχρονισμό των πληροφοριών της εφοδιαστικής αλυσίδας και την αδιάλειπτη σε πραγματικό χρόνο παρακολούθηση και τον εντοπισμό των αντικειμένων. Θα καταστήσει την αλυσίδα εφοδιασμού διαφανή, ορατή και ελεγχόμενη, επιτρέποντας την έξυπνη επικοινωνία μεταξύ των ανθρώπων και των εμπορευμάτων / αγαθών.

❖ **Παράδειγμα 2^{ο}** : Το έξυπνο σπίτι

Τα μελλοντικά έξυπνα σπίτια θα έχουν επίγνωση για το τι συμβαίνει μέσα σε ένα κτίριο, κυρίως επηρεάζοντας τρεις πτυχές: τη χρήση των πόρων (εξοικονόμηση νερού και κατανάλωση ενέργειας), την ασφάλεια και την άνεση. Ο στόχος είναι να επιτευχθούν καλύτερα επίπεδα άνεσης, ενώ παράλληλα να μείωθούν οι συνολικές δαπάνες. Επιπλέον, τα έξυπνα σπίτια εξετάζουν θέματα ασφάλειας μέσω πολύπλοκων συστημάτων ασφαλείας για την ανίχνευση πυρκαγιάς, κλοπής ή παράνομης εισόδου.



Εικόνα 3.2.1 Smart home

Οι φορείς που εμπλέκονται σε αυτό το σενάριο αποτελούν μια πολύ ετερογενή ομάδα. Διαφορετικοί φορείς θα συνεργαστούν στο σπίτι του χρήστη, όπως εταιρίες παροχής Internet, κατασκευαστές συσκευών, φορείς εκμετάλλευσης τηλεπικοινωνιών, πάροχοι υπηρεσιών μέσων επικοινωνίας, εταιρείες security, εταιρείες κοινής ωφέλειας ηλεκτρικής ενέργειας, κλπ.

❖ **Παράδειγμα 3^{ο}** : Έξυπνες πόλεις

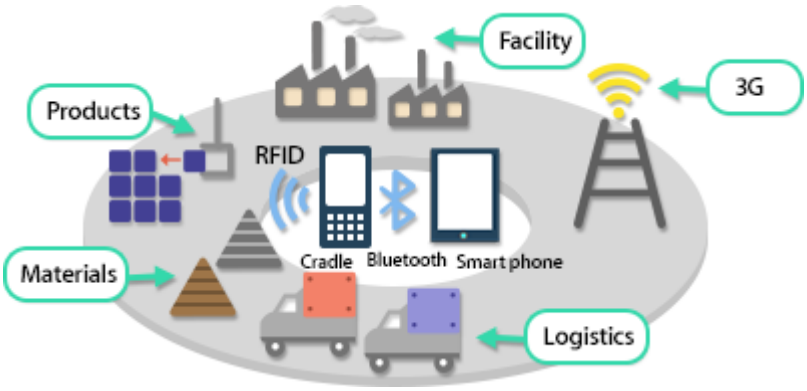
Ενώ ο όρος έξυπνη πόλη εξακολουθεί να είναι μια ασαφής έννοια, υπάρχει η γενική συμφωνία ότι είναι μια αστική περιοχή που δημιουργεί βιώσιμη ανάπτυξη και υψηλή ποιότητα ζωής. Το μοντέλο του Giffinger «et al» διαφωτίζει τα χαρακτηριστικά μιας έξυπνης πόλης, η οποία περιλαμβάνει την οικονομία, τους ανθρώπους, τη διακυβέρνηση, την κινητικότητα, το περιβάλλον και την ζωή μέσα σε αυτή.[11]

Εικόνα 3.2.2 Smart city

Η υπεραπόδοση σε αυτούς τους βασικούς τομείς μπορεί να επιτευχθεί μέσω ισχυρού ανθρώπινου ή κοινωνικού κεφαλαίου ή και των υποδομών ΤΠΕ. Για το τελευταίο, μία αρχική επιχειματική ανάλυση καταλήγει στο συμπέρασμα ότι οι διάφοροι τομείς / κλάδοι θα επωφεληθούν από πιο ψηφιοποιημένες και έξυπνες πόλεις.

- **Παράδειγμα 4^{ο}** : Έξυπνα εργοστάσια

Σε μια παγκόσμια αλυσίδα εφοδιασμού, οι εταιρείες θα είναι σε θέση να παρακολουθούν όλα τα προϊόντα τους μέσω των ετικετών ταυτοποίησης ραδιοσυχνοτήτων (RFID). Κατά συνέπεια, οι εταιρείες θα μειώσουν τα λειτουργικά τους έξοδα (OPEX) και θα βελτιώσουν την παραγωγικότητα λόγω της μεγαλύτερης ολοκλήρωσης μέσω του προγραμματισμού επιχειρηματικών πόρων (ERP) και άλλων συστημάτων. Επίσης, η συντήρηση των μηχανημάτων θα διευκολυνθεί μέσω των συνδεδεμένων αισθητήρων, επιτρέποντας την παρακολούθηση σε πραγματικό χρόνο της υγείας και της απόδοσης του εξοπλισμού του εργοστασίου.
Σε γενικές γραμμές, το IoT θα παρέχει αυτόματες διαδικασίες οι οποίες θα επιφέρουν δραστική μείωση του αριθμού των εργαζομένων που απαιτούνται. Οι εργαζόμενοι θα αντικατασταθούν από bar code scanners, αναγνώστες, αισθητήρες και ενεργοποιητές, και στο τέλος από πολύπλοκα ρομπότ τόσο αποτελεσματικά και αποδοτικά όσο ένα ανθρώπινο ον.



Εικόνα 3.2.3 Smart factory

Χωρίς καμία αμφιβολία, αυτές οι τεχνολογίες θα δημιουργήσουν θέσεις εργασίας και ευκαιρίες για υπάλληλους γραφείου και ένας μεγάλος αριθμός τεχνικών θα είναι απαραίτητος για τον προγραμματισμό και την επισκευή αυτών των μηχανών. Αυτό είναι συνώνυμο με μια μεταφορά σε θέσεις εργασίας συντήρησης, αλλά αποτελεί ακόμα και μια νέα πρόκληση για την παροχή ευκαιρίας σε όλους τους εργάτες να κινηθούν προς αυτούς τους τύπους θέσεων εργασίας και έτσι να προληφθεί η ανεργία.

3.3. Το Διαδίκτυο των Πραγμάτων σε συνδιασμό με συναφείς μελλοντίκες τεχνολογίες διαδικτύου

- **IoT και υπολογιστικό νέφος** (Cloud computing)

Από τη δημοσίευση του SRA του 2011, το cloud computing έχει καθιερωθεί
ως ένα από τα σημαντικότερα δομικά στοιχεία του διαδικτύου του μέλλοντος (Future Internet). Νέα τεχνολογικά μέσα έχουν σταδιακά προωθήσει την εικονικοποίηση σε διάφορα επίπεδα και επέτρεψαν τα διάφορα παραδείγματα που είναι γνωστά ως «Software-as-a-Service»,«Platforms as a Service» και «Infrastructure and Networks as a Service».
Οι τάσεις αυτές έχουν συμβάλει σε μεγάλο βαθμό στη μείωση του κόστους της ιδιοκτησίας και της διαχείρισης των συνδεδεμένων εικονικών πόρων, τη μείωση του ορίου εισόδου στην αγορά για νέους παίκτες και τη δυνατότητα πρόβλεψης των νέων υπηρεσιών. Η εικονικοποίηση των αντικειμένων είναι το επόμενο φυσικό βήμα σε αυτή την τάση, η σύγκλιση του υπολογιστικού νέφους και του Διαδικτύου των Πραγμάτων θα επιτρέψει πρωτοφανείς ευκαιρίες στην αρένα υπηρεσιών του IoT.[12]

Ως μέρος αυτής της σύγκλισης, οι εφαρμογές IoT (όπως οι υπηρεσίες που βασίζονται σε αισθητήρες) θα παραδοθούν on-demand μέσω ενός περιβάλλοντος cloud [13]. Αυτό εκτείνεται πέρα από την ανάγκη για εικονικοποίηση και αποθήκευση των δεδομένων από αισθητήρες με ένα κλιμακούμενο τρόπο.Ζητά την εικονικοποίηση των συνδεδεμένων στο Internet αντικειμένων και την ικανότητά τους να ενορχηστρωθούν σε on-demand υπηρεσίες (όπως το Sensing-as-a-Service).

Επιπλέον, γενικεύοντας το εξυπηρετούμενο πεδίο ενός συνδεδεμένου στο διαδίκτυο αντικειμένου πέρα από το «Sensing-as-a-Service», δεν είναι δύσκολο να φανταστεί κανείς εικονικά αντικείμενα που θα ενσωματωθούν στις μελλοντικές υπηρεσίες του Διαδικτύου των πραγμάτων και θα μοιράζονται και θα επαναχρησιμοποιούνται σε διαφορετικά πλαίσια, προβάλλοντας ένα «Object as a Service» με στόχο, όπως

και σε άλλους τομείς εικονικοποιημένων πόρων, την ελαχιστοποίηση του κόστους της ιδιοκτησίας και της συντήρησης των αντικειμένων, και προωθώντας τη δημιουργία καινοτόμων υπηρεσιών IoT.

Ως εκ τούτου, συναφή θέματα για την ατζέντα της έρευνας θα είναι:

• Η περιγραφή των αιτήσεων για την παροχή υπηρεσιών σε υποδομές cloud / IoT,

• Η εικονικοποίηση των αντικειμένων,

• Εργαλεία και τεχνικές για τη βελτιστοποίηση των υποδομών cloud με γνώμονα την χρησιμότητα και τα κριτήρια SLA,

• Η έρευνα των:

◦ μετρήσεων κοινής ωφέλειας και

◦ (η ενίσχυση) τεχνικών μάθησης που θα μπορούσαν να χρησιμοποιηθούν για την μέτρηση των on-demand υπηρεσίων IoT σε περιβάλλον cloud,

• Τεχνικές για την αλληλεπίδραση σε πραγματικό χρόνο των συνδεδεμένων στο διαδίκτυο αντικειμένων μέσα σε ένα περιβάλλον cloud μέσω της εφαρμογής ελαφρών αλληλεπιδράσεων και η προσαρμογή της λειτουργίας του συστήματος σε πραγματικό χρόνο.

❖ IoT και σημασιολογικές τεχνολογίες

Το SRA του 2010 έχει αναγνωρίσει τη σημασία των σημασιολογικών τεχνολογιών προς την ανακάληψη συσκευών, καθώς και στην επίτευξη σημασιολογικής διαλειτουργικότητας.

Κατά τη διάρκεια των τελευταίων ετών, οι τεχνολογίες σημασιολογικού ιστού έχουν επίσης αποδείξει την ικανότητα τους να συνδέουν δεδομένα που σχετίζονται μεταξύ τους (web-of-data concept) [14], ενώ τα σχετικά εργαλεία και οι τεχνικές έχουν μόλις βγει . Η μελλοντική έρευνα για το IoT είναι πιθανό να αγκαλιάσει την ιδέα του Linked Open Data. Αυτό θα μπορούσε να βασιστεί στην προγενέστερη ολοκλήρωση των οντολογιών (π.χ., οντολογίες αισθητήρα) σε υποδομές IoT και εφαρμογές.

Οι σημασιολογικές τεχνολογίες θα έχουν επίσης σημαντικό ρόλο στην επίτευξη της κοινής χρήσης και επαναχρησιμοποίησης των εικονικών αντικειμένων ως υπηρεσία μέσω του cloud, όπως φαίνεται στην προηγούμενη παράγραφο. Ο σημασιολογικός εμπλουτισμός των περιγραφών των εικονικών αντικειμένων θα δείξει στο IoT το πως ο σημασιολογικός σχολιασμός των ιστοσελίδων είναι ενεργοποιημένος στον Σημασιολογικό Ιστό.
Η συσχετισμένη σημασιολογική συλλογιστική θα βοηθήσει τους χρήστες του IoT
να βρουν πιο ανεξάρτητα τα σχετικά αποδεδειγμένα εικονικά αντικείμενα για τη βελτίωση της απόδοσης ή της αποτελεσματικότητας των εφαρμογών IoT που προτίθενται να χρησιμοποιήσουν.

- **IoT και αυτονομία**

Οι θεαματικές εξελίξεις στην τεχνολογία έχουν εισαγάγει όλο και πιο σύνθετα και
μεγάλης κλίμακας συστήματα πληροφορικής και επικοινωνιών. Η αυτόνομη υπολογιστική, εμπνευσμένη από τα βιολογικά συστήματα, έχει προταθεί ως μια μεγάλη πρόκληση που θα επιτρέψει στα συστήματα να αυτο-διαχειρίζονται αυτήν την πολυπλοκότητα, χρησιμοποιώντας στόχους υψηλού επιπέδου και πολιτικές που καθορίζονται από τον άνθρωπο.
Ο στόχος είναι να παρέχει κάποια αυτό-x ιδιότητα του συστήματος, όπου x μπορεί να είναι η προσαρμογή, η οργάνωση, η βελτιστοποίηση, η διαμόρφωση, η προστασία, η θεραπεία, η ανακάλυψη, η περιγραφή, κλπ.
Το Ίντερνετ των πραγμάτων θα αυξήσει εκθετικά την κλίμακα και την πολυπλοκότητα
των υπαρχόντων συστημάτων πληροφορικής και επικοινωνιών. Η αυτονομία είναι έτσι μία επιτακτική ιδιοκτησία την οποία θα πρέπει τα συστήματα IoT να έχουν. Ωστόσο, εξακολουθεί να υπάρχει έλλειψη έρευνας για το πώς να προσαρμοστούν και να προσαρμόζουν την υπάρχουσα έρευνα σχετικά με την αυτόνομη υπολογιστική
με τα ειδικά χαρακτηριστικά του διαδικτύου των πραγμάτων, όπως η υψηλή δυναμικότητα και διανομή, η πραγματικού χρόνου φυση που θα πρέπει να έχουν, οι περιορισμένοι πόροι και τα περιβάλλοντα με απώλειες.

3.4. Υποδομή του Διαδικτύου των Πραγμάτων

Το Διαδίκτυο των πραγμάτων θα γίνει μέρος της καθημερινής ζωής. Θα γίνει μέρος της συνολικής υποδομής μας ακριβώς όπως το νερό, το ηλεκτρικό ρεύμα, το τηλέφωνο, η τηλεόραση και πιο πρόσφατα το διαδίκτυο. Λαμβάνοντας υπόψη ότι το σημερινό διαδίκτυο τυπικά συνδέει υπολογιστές πλήρους κλίμακας, το Διαδίκτυο των Πραγμάτων (ως μέρος του Future Internet) θα συνδέει αντικείμενα καθημερινής χρήσης με μία ισχυρή ενοποίηση στον φυσικό κόσμο.

Plug and Play ενσωμάτωση (τοποθέτηση και άμεση λειτουργία)

Αν κοιτάξουμε την συσχετιζόμενη με το IoT τεχνολογία που είναι διαθέσιμη σήμερα, υπάρχει μια τεράστια ανομοιογένεια.
Είναι συνήθως ανεπτυγμένη για πολύ συγκεκριμένους σκοπούς και οι ρυθμίσεις απαιτούν σημαντικές τεχνικές γνώσεις και μπορεί να είναι επαχθής. Για την επίτευξη ενός αληθινού Διαδικτύου των πραγμάτων θα πρέπει να απομακρυνθούμε από αυτές τις μικρής κλίμακας, κάθετων εφαρμογών, προς μια οριζόντια υποδομή στην οποία μια ποικιλία εφαρμογών μπορούν να τρέχουν ταυτόχρονα.
Αυτό είναι δυνατό μόνο εάν η συνδεση ενός πράγματος στο Διαδίκτυο των πραγμάτων γίνεται τόσο απλά όσο το να συνδεθεί και το τεθεί σε λειτουργία μια ηλεκτρική συσκευή. Τέτοιες λειτουργίες plug and play απαιτούν μια υποδομή που να τις υποστηρίζουν, ξεκινώντας από το επίπεδο δικτύου και πέρα από αυτό το επίπεδο εφαρμογών. Αυτό συνδέεται στενά με τα ζητήματα που συζητούνται στο τμήμα για αυτονομία. Σε επίπεδο δικτύωσης, η λειτοργία plug & play πρέπει να επιτρέπει την επικοινωνία, και χαρακτηριστικά όπως αυτά που παρέχονται από το IPv6 είναι στις κατευθύνσεις για να βοηθήσουν σε αυτή τη διαδικασία.
Κατάλληλα συστατικά υποδομών πρέπει στη συνέχεια να ανακαλυφθούν για να καταστεί δυνατή η ένταξη των πραγμάτων στο Διαδίκτυο των Πραγμάτων. Αυτό περιλαμβάνει το να ανακοινωθούν οι λειτουργίες που παρέχονται, όπως το τι μπορεί να ανιχνεύεται ή τι μπορεί να ενεργοποιείται.

Λειτουργικότητα υποδομής

Η υποδομή πρέπει να υποστηρίζει εφαρμογές για την εξεύρεση των πραγμάτων που απαιτούνται.
Μια εφαρμογή μπορεί να τρέξει οπουδήποτε, συμπεριλαμβανομένων των ίδιων των πραγμάτων. Η εξεύρεση των πραγμάτων δεν περιορίζεται στο χρόνο εκκίνησης της εφαρμογής. Χρειάζεται αυτόματη προσαρμογή κάθε φορά που σχετικά νέα πράγματα γίνονται διαθέσιμα, γίνονται μη διαθέσιμα ή η κατάσταση τους αλλάζει. Η υποδομή πρέπει να υποστηρίζει την παρακολούθηση αυτών των αλλαγών και την προσαρμογή που απαιτείται ως αποτέλεσμα των αλλαγών αυτών.

Σημασιολογική μοντελοποίηση των πραγμάτων

Για να επιτευχθεί το πλήρες δυναμικό του Διαδικτύου των πραγμάτων, οι σημασιολογικές πληροφορίες οι οποίες αφορούν τα πράγματα, οι πληροφορίες που μπορούν να παρέχουν ή οι ενεργοποιήσεις που απαιτούνται για να μπορεί να εκτελέστει μια ανάγκη θα πρέπει να είναι διαθέσιμες. Δεν αρκεί να γνωρίζουμε ότι υπάρχει ένας αισθητήρας θερμοκρασίας ή ένας ηλεκτρικός κινητήρας, αλλά είναι σημαντικό να γνωρίζουμε ποια θερμοκρασία μέτρα ο αισθητήρας: την εσωτερική θερμοκρασία του δωματίου ή την θερμοκρασία του ψυγείου, και αν ο ηλεκτροκινητήρας μπορεί να ανοίξει ή να κλείσει τις περσίδες ή να μετακινήσει κάτι σε μια διαφορετική θέση.
Δεδομένου ότι μπορεί να μην είναι δυνατή η παροχή τέτοιων σημασιολογικών πληροφορίων με την απλή μετάβαση στο πράγμα, η υποδομή θα πρέπει να κάνει εύκολη την πρόσθεση νέων σημασιών για τους χρήστες. Επίσης, μπορεί να είναι δυνατό να αντλήσει σημασιολογική πληροφορία, δεδομένου ότι κάποιες βασικές πληροφορίες και συμπληρωματικές γνώσεις π.χ. οι πληροφορίες που αφορούν ένα δωμάτιο, είναι βασισμένες στις πληροφορίες από ένα συγκεκριμένο αισθητήρα που είναι τοποθετημένος στο δωμάτιο. Αυτό θα έπρεπε να ενεργοποιείται από την υποδομή.

Φυσική τοποθεσία και θέση

Δεδομένου ότι το Διαδίκτυο των πραγμάτων είναι βαθιά ριζωμένο στον φυσικό κόσμο, η έννοια της φυσικής τοποθεσίας και της θέσης είναι πολύ σημαντική, ειδικά για την εξεύρεση πραγμάτων, αλλά επίσης και για την εξαγωγή της γνώσης. Ως εκ τούτου, η υποδομή πρέπει να υποστηρίζει τον εντοπισμό αυτών των πραγμάτων ανάλογα με την τοποθεσία (π.χ. geo-location βάση της ανακάλυψης).

Λαμβάνοντας την κινητικότητα υπόψη, οι τεχνολογίες εντοπισμού θα διαδραματίσουν σημαντικό ρόλο για το Διαδίκτυο των πραγμάτων και μπορεί να ενσωματωθούν στην υποδομή του Διαδικτύου των πραγμάτων.

Ασφάλεια και προστασία προσωπικών δεδομένων

Επιπλέον, η υποδομή πρέπει να παρέχει στήριξη για την ασφάλεια και για λειτουργίες προστασίας της ιδιωτικής ζωής όπως την αναγνώριση, την εμπιστευτικότητα, την ακεραιότητα, τη μη άρνηση ταυτότητας και αδειοδότησης.
Εδώ η ετερογένεια και η ανάγκη για διαλειτουργικότητα μεταξύ των διαφορετικών συστημάτων ΤΠΕ τα οποία έχουν αναπτυχθεί στην υποδομή και οι περιορισμοί των πόρων των συσκευών του IoT (π.χ., Nano αισθητήρες) πρέπει να λαμβάνονται υπόψη.

3.5. Διαχείρηση δεδομένων (Data Management)

Εικόνα 3.5.1 Data management

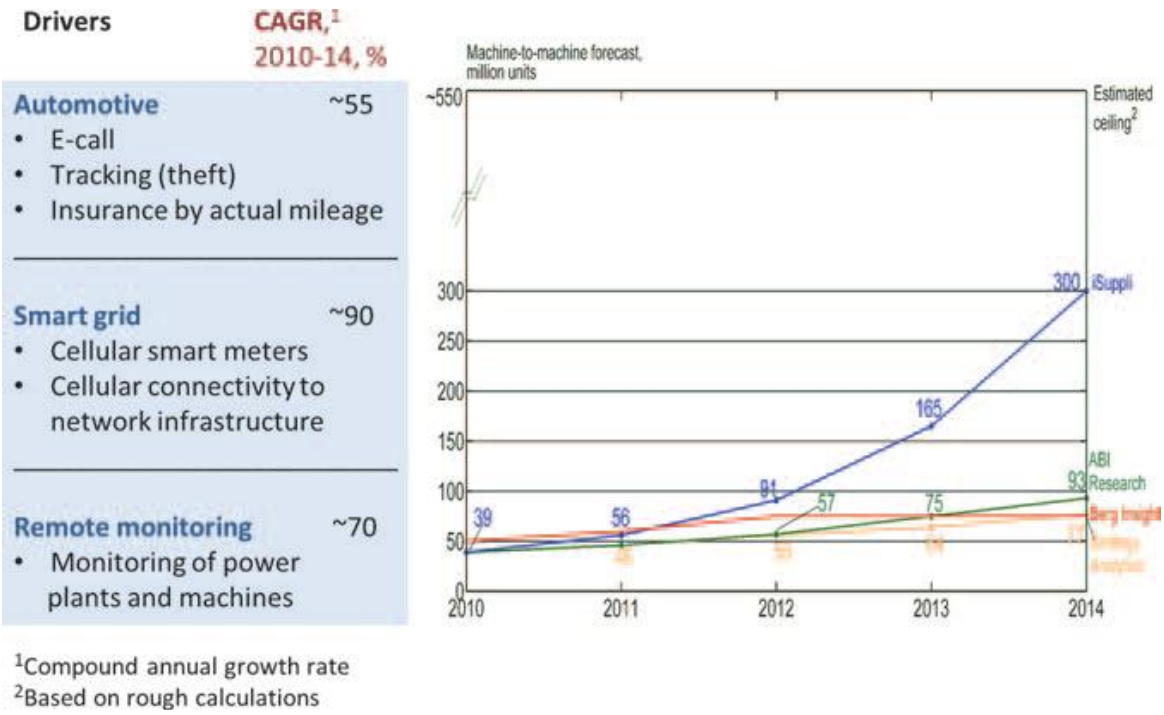
Η διαχείριση των δεδομένων είναι μια κρίσιμη πτυχή του Διαδικτύου των πραγμάτων. Αν έχουμε υπόψιν ένα κόσμο αντικειμένων τα οποία είναι διασυνδεδεμένα και συνεχώς ανταλλάζουν όλων των ειδών πληροφορίες, η ένταση των παραχθέντων δεδομένων και

οι διαδικασίες που συμπεριλαμβάνονται στην διαχείρηση αυτών των δεδομένων είναι κρίσιμες.

Μια μακροπρόθεσμη ευκαιρία για κατασκευαστές τσιπ ασύρματων επικοινωνιών είναι η αύξηση της μηχανής-προς-μηχανή (M2M) πληροφορικής, η οποία είναι μία από τις τεχνολογίες επίτρεψης (enabling) για το Ίντερνετ των πραγμάτων.

Η τεχνολογία αυτή εκτείνεται στο εξωτερικό φάσμα των εφαρμογών. Ενώ δεν υπάρχει συναίνεση ότι το M2M είναι ένας πολλά υποσχόμενος τομέας ανάπτυξης, οι εκτιμήσεις των αναλυτών για το μέγεθος της ευκαιρίας αποκλίνουν κατά ένα συντελεστή τεσσάρων μονάδων. Σύμφωνα με συντηρητικές εκτιμήσεις υποθέτουν περίπου 80 εκατομμύρια έως 90 εκατομμύρια μονάδες M2M θα πωληθούν το 2014, ενώ προβλέπονται πιο αισιόδοξες προβλέψεις για πωλήσεις 300 εκατ. μονάδων.

Με βάση την ανάλυση των ιστορικών καμπυλών των εκδόσεων παρόμοιων επαναστατικών τεχνολογιών, όπως τα φορητά MP3 players και τα αντι κλειδωτικά συστήματα πέδησης για τα αυτοκίνητα, πιστεύεται ότι η μονάδα πωλήσεων για τα M2M θα μπορούσε να αυξηθεί έως και κατά ένα συντελεστή δεκαπλάσιο για τα επόμενα πέντε χρόνια, όπως φαίνεται στο παρακάτω σχήμα:



Εικόνα 3.5.2 M2M forecast diagram

Υπάρχουν πολλές τεχνολογίες και παράγοντες που εμπλέκονται στη “διαχείριση των δεδομένων” στο πλαίσιο του IoT.
Μερικές από τις πιο σημαντικές έννοιες που θα μας επιτρέψουν να κατανοήσουμε το προκλήσεις και τις ευκαιρίες της διαχείρισης των δεδομένων είναι:

- Συλλογή δεδομένων και ανάλυση
- Big Data
- Σημασιολογικά δίκτυα αισθητήρων
- Εικονικοί αισθητήρες
- Επεξερασία περίπολων συμβάντων

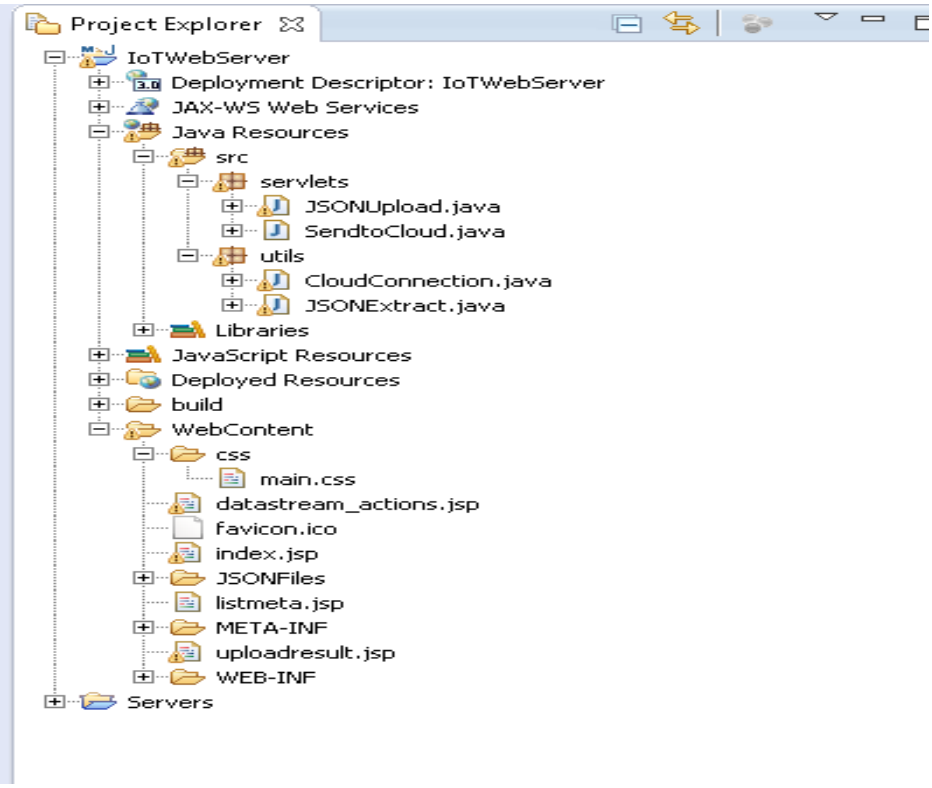
3.6. Το Διαδίκτυο των πραγμάτων στην παρούσα εργασία

Στην παρούσα διπλωματική εργασία επεξεργάζονται δεδομένα τα οποία θα περιέχουν πληροφορίες αισθητήρων ή άλλων πραγμάτων του Διαδικτύου των πραγμάτων.
Τα δεδομένα αυτά αφού επεξεργαστούν και φτάσουν στην επιθυμητή μορφή αποθηκεύονται στο αποθηκευτικό νέφος αντικειμένων το οποίο στήθηκε στην προηγούμενη ενότητα.
Στην συνέχεια θα μπορούμε να ζητάμε πληροφορίες από το αποθηκευτικό νέφος, να αποθηκεύουμε και άλλα αντικείμενα, να διαγράφουμε αντικείμενα και να αναβαθμίζουμε αντικείμενα.
Σημαντικό πλεονέκτημα είναι η χρησιμοποιήση μεταδεδομένων με τον τρόπο που αναφέρθηκε στο προηγούμενο κεφάλαιο. Με τα μεταδεδομένα μπορούμε να κάνουμε εύκολη και γρήγορα αναζήτηση στα αντικείμενα. Τα μεταδεδομένα μπορούν και αυτά να αναβαθμιστούν και να διαγραφούν.
Αυτή η επεξεργασία θα γίνεται μέσω ενός ενδιάμεσου διακομιστή ο οποίος θα είναι υπεύθυνος για την επικοινωνία του χρήστη(ο οποίος στέλνει την ενέργεια για το “πράγμα” του IoT) και του αποθηκευτικού νέφους αντικειμένων της Openstack.
Το πως λειτουργούν και πως έχουν αναπτυχθεί όλα αυτά θα αναλυθούν στην επόμενη ενότητα.

Κεφάλαιο 4: Υλοποίηση

4.1. Αρχιτεκτονική και δομή της εφαρμογής

Ο ενδιάμεσος διακομιστής ο οποίος επιλέχθηκε είναι ο tomcat της apache. Συγκεκριμένα η έκδοση του είναι η 7η. Σε αυτόν τον διακομιστή ο οποίος έχει στηθεί σε τοπικό υπολογιστή βρίσκεται η εφαρμογή που είναι υπεύθυνη για την επικοινωνία του χρήστη με το νέφος αποθήκευσης αντικειμένων. Ο χρήστης μπορεί να επικοινωνεί και να δίνει εντολές στον τοπικό ενδιάμεσο διακομιστή μέσω του φυλλομετρητή ιστού (internet browser). Αυτό έγινε για λόγους ευκολίας εφόσον δεν είναι το θέμα της παρούσας διπλωματικής εργασίας ο τρόπος με τον οποίο επικοινωνούν τα πράγματα με τον διακομιστή αλλά το πως γίνονται οι λειτουργίες αποθήκευσης αναζήτησης και ομαδοποίησης των δεδομένων σε ένα υπολογιστικό νέφος. Δηλαδή η διαχείρηση των δεφομένων.
Η εφαρμογή είναι γραμμένη σε γλώσσα προγραμματισμού Java αλλά χρησιμοποιούνται και άλλες τεχνολογίες όπως HTML (Hyper Text Markup Language), CSS (Cascading Style Sheets)και JSP (Java Server Pages), JSON (JavaScript Object Notation). Συγκεκριμένα χρησιμοποιείται η τεχνολογία Java Servlets. Στην παρακάτω εικόνα φαίνεται ο τρόπος με τον οποίο είναι ο φάκελος της εφαρμογής που περιέχει τον κώδικα και τα δεδομένα.



Εικόνα 4.1 Project file organization

Το project έχει όνομα IoTWebServer και μέσα σε αυτό φαίνονται όλα τα αρχεία κώδικα. Ακόμα πρέπει να αναφερθεί ότι τα πράγματα του IoT στέλνουνε τις πληροφορίες τους σε αρχεία της μορφής JSON.
Το JSON είναι μια ελαφριά μορφή ανταλλαγής δεδομένων. Είναι εύκολο για τους ανθρώπους να το διαβάζουν και να το γράφουν. Είναι εύκολο για τις μηχανές να το αναλύσουν και να το δημιουργήσουν. Βασίζεται σε ένα υποσύνολο της γλώσσας προγραμματισμού JavaScript, το Standard ECMA-262 3rd Edition - Δεκέμβριος 1999. Το JSON είναι μια μορφή κειμένου που είναι εντελώς ανεξάρτητη ως γλώσσα, αλλά χρησιμοποιεί τις συμβάσεις που είναι γνωστές στους προγραμματιστές της C-οικογένειας γλωσσών, συμπεριλαμβανομένων των C , C + +, C #, Java, JavaScript, Perl, Python, και πολλών άλλων. [15]

Ο φάκελος src περιέχει όλο τον κώδικα Java οποίος είναι υπεύθυνος για τις λειτουργίες της εφαρμογής μέσω του ενδιάμεσου διακομιστή. Χωρίζεται σε 2 πακέτα (packages) τα οποία όπως φαίνεται από την εικόνα είναι τα servlets και utils. Στον φάκελο servlets βρίσκεται ο κώδικας που είναι υπεύθυνος για την επικοινωνία του χρήστη με τον τοπικό διακομιστή (tomcat_v7 server) ενώ στον φάκελο utils βρίσκονται τα utilities δηλαδή τα βοηθητικά προγράμματα.

- **Φάκελος servlets**

Περιέχει 2 αρχεία: το JSONUpload.java που είναι υπεύθυνο για να ανεβάσουμε στον τοπικό διακομιστή το αρχείο που έχει τις πληροφορίες του πράγματος και το SendtoCloud.java το οποίο περιέχει τις λειτουργίες επικοινωνίας με τον χρήστη για να οριστεί η αλληλεπίδραση που θα γίνει με το νέφος.

- **Φάκελος utils**

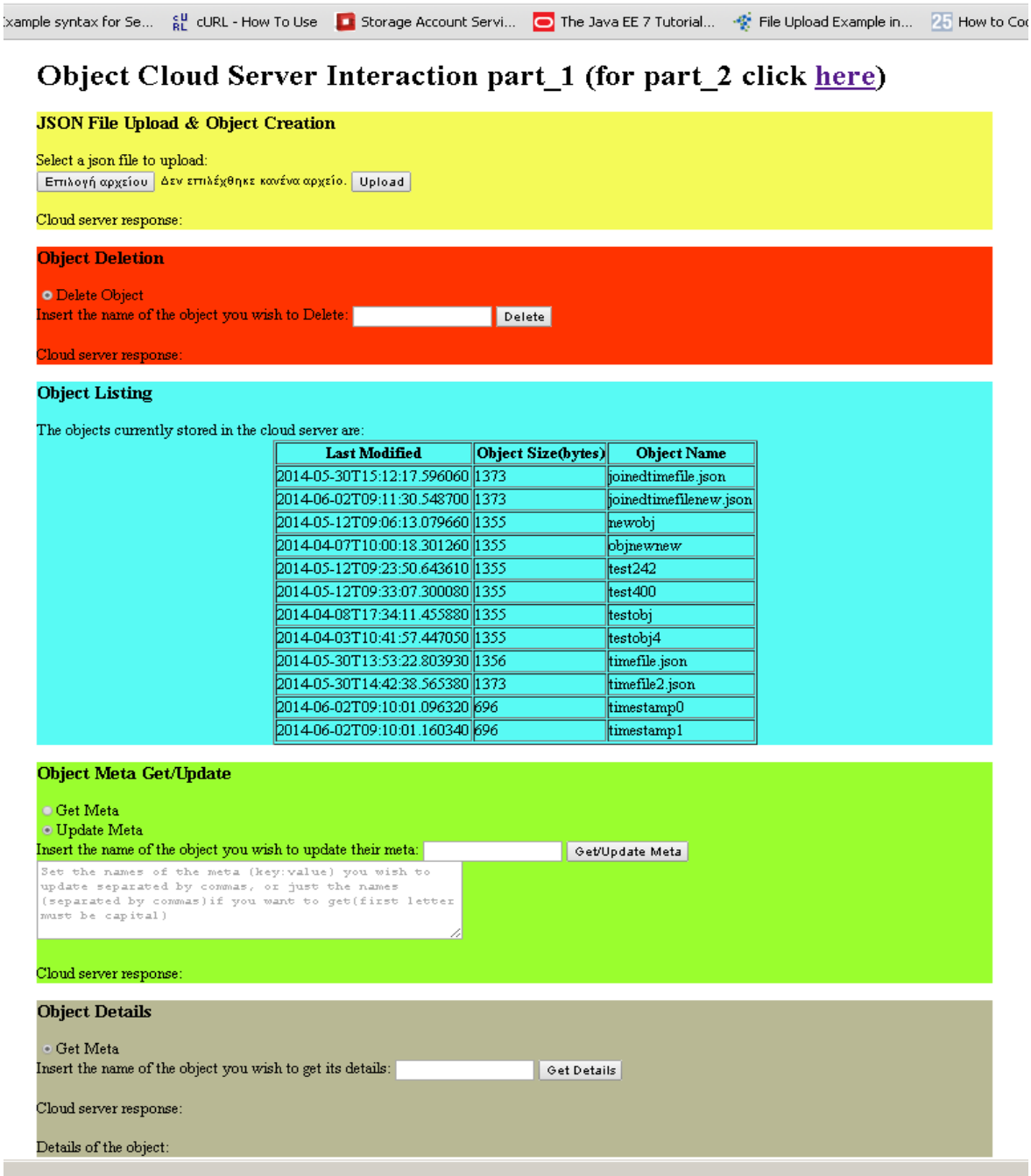
Περιέχει 2 αρχεία: το JSONExtract.java το οποίο περιέχει τις λειτουργίες για να πάρουμε τα δεδομένα από τα αρχεία JSON τα οποία περιέχουν τα δεδομένα των πραγμάτων. Ακόμα περιέχει και το CloudConnection.java που είναι το σημαντικότερο αρχείο της εφαρμογής. Είναι υπεύθυνο να μεταφράζει τις εντολές που παίρνει από το SendtoCloud.java σε εντολές προς το νέφος αποθήκευσης και να επιστρέφει την απάντηση που παίρνει.

Στον φάκελο WebContent βρίσκονται αρχεία τα οποία είναι υπεύθυνα για την εμφάνιση του περιεχομένου και των επιλογών που έχει ο χρήστης. Είναι δηλαδή η διεπαφή η οποία εμφανίζεται μέσω του φυλλομετρητή στον χρήστη. Τα αρχεία αυτά είναι τύπου JSP δηλαδή είναι αρχεία HTML & CSS εμπλουτισμένα με κώδικα Java. Ακόμα μέσα στον

φάκελο WebContent υπάρχει και ένας άλλος φάκελος ο JSONFiles όπου περιέχονται όλα τα αρχεία JSON τα οποία ή τα έχει ανεβάσει στον τοπικό διακομιστή ο χρήστης ή έχουν σταλεί από πράγματα του IoT και βρίσκονται εκεί.

4.2. Ανάλυση της υλοποίησης του πρώτου μέρους της εφαρμογής

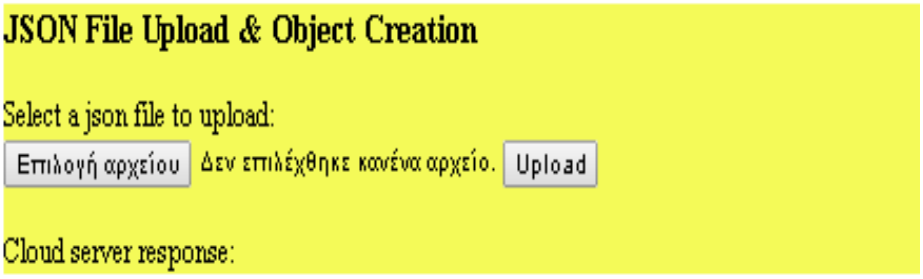
Η πλήρης διεπαφή του πρώτου μέρους της εφαρμογής φαίνεται στην παρακάτω εικόνα:



Εικόνα 4.2.1 Διεπαφή πρώτου μέρους

4.2.1 Ανέβασμα JSON και δημιουργία αντικειμένου στο νέφος

Στο μέρος αυτό έχουμε το πρώτο μέρος της διεπαφής (με κίτρινο χρώμα) η οποία επιτρέπει στον χρήστη να επιλέξει ένα αρχείο JSON και να το ανεβάσει στον τοπικό ενδιάμεσο διακομιστή. Η διεπαφή του πρώτου μέρους φαίνεται στην παρακάτω εικόνα:



Εικόνα 4.2.2 Φόρμα ανεβάσματος αρχείου

Ο κώδικας της διεπαφής αυτής φαίνεται εδώ:

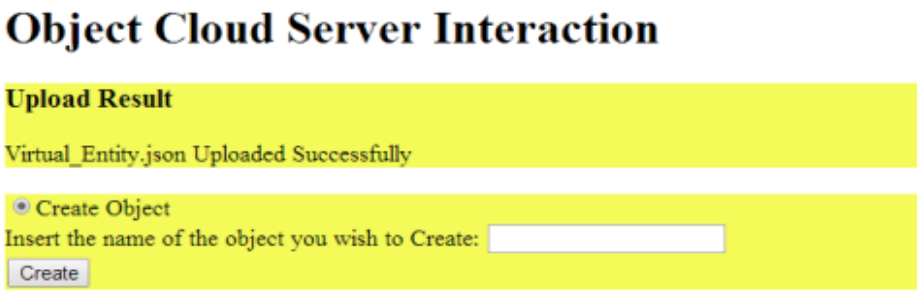
```
<div id="upload">
            <h3>JSON File Upload & Object Creation</h3>
            Select a json file to upload: <br />
                  <form action="JSONUpload" method="post"
enctype="multipart/form-data">
                  <input type="file" name="file" size="50" />
                  <input type="submit" value="Upload" />
                  </form>
            <br>
            Cloud server response: ${requestScope["response"]}
            <h3>${requestScope["message"]}</h3>
</div>
```

Στον κώδικα αυτό το κομμάτι που μας ενδιαφέρει είναι το form action το οποίο μας δείχνει ότι όταν κάνουμε submit την φόρμα αυτήν θα κληθεί το JSONUpload.java το οποίο είναι υπεύθυνο για το ανέβασμα του αρχείου στον τοπικό διακομιστή.
Εδω βλέπουμε τον κώδικα της JSONUpload κλάσσης:

```
protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        // Check that we have a file upload request
        //process only if its multipart content
        String name = null;
        String rootDir=getServletContext().getRealPath("");
        HttpSession session= request.getSession();
        session.setAttribute("rootDir", rootDir);

if(ServletFileUpload.isMultipartContent(request)){
```

```
                    try {
                        List<FileItem> multiparts = new
ServletFileUpload(new
DiskFileItemFactory()).parseRequest(request);
                        for(FileItem item : multiparts){
                            if(!item.isFormField()){
                                String filename = new
File(item.getName()).getName();
                                item.write( new File(rootDir +
File.separator + UPLOAD_DIRECTORY + File.separator + filename));
                                name=filename;
                            }
                        }
                       //File uploaded successfull
                       session.setAttribute("filename", name);
                       request.setAttribute("message", name +"
Uploaded Successfully");

request.getRequestDispatcher("/uploadresult.jsp").forward(request
, response);
                    } catch (Exception ex) {
                       request.setAttribute("message", "File
Upload Failed due to " + ex);

request.getRequestDispatcher("/index.jsp").forward(request,
response);
                    }
                }else{
                    request.setAttribute("message","Sorry this
Servlet only handles file upload request");

request.getRequestDispatcher("/index.jsp").forward(request,
response);
                }

            }
```

Εδώ παρατηρούμε ότι αφού ανεβάσουμε το αρχείο στον τοπικό διακομιστή αποθηκεύουμε το όνομα του στο session της σύνδεσης έτσι ώστε να μπορούμε να το χρησιμοποιήσουμε αργότερα.
Αμέσως αφού ολοκληρωθεί το ανέβασμα ο παραπάνω κώδικας μας κατευθύνει στην σελίδα /uploadresult.jsp.
Την σελίδα αυτή τη βλέπουμε το παρακάτω:



Εικόνα 4.2.3 Φόρμα δημιουργίας αντικειμένου

Στην φόρμα αυτήν λοιπόν μπορούμε να βάλουμε το όνομα το οποίο θέλουμε να δώσουμε στο νέο αντικείμενο που θα αποθηκευτεί στο νέφος και πατάμε Create.
Από τον παρακάτω κώδικα της σελίδας uploadresult.jsp που βλέπουμε στην εικόνα το κομμάτι: `<form autocomplete="off" action="SendtoCloud" method="post">`
φαίνεται ότι όταν πατήσουμε το Create στην φόρμα θα κλιθεί η κλάσση SendtoCloud η οποία είναι υπεύθυνη να πάρει την εντολή που δώσαμε (create) και να καλέσει την κατάλληλη μέθοδο για να αποθηκευτεί το αντικείμενο.

Αρχικά στην αρχή της κλάσης SendtoCloud υπάρχει ένα σημαντικό κομμάτι κώδικα το οποίο φαίνεται εδώ:

```
String filename = (String) session.getAttribute("filename");
        String action = request.getParameter("action");
        String objname= request.getParameter("objname");
        String rootDir = getServletContext().getRealPath("");
        CloudConnection cc = new CloudConnection();
        String[] connresult = cc.getAuthToken();
```

Στο κομμάτι αυτό αρχικά παίρνουμε από το session το όνομα του αρχείου που ανεβάσαμε. Στην συνέχεια παίρνουμε το action που δηλώνει το τι θέλουμε να κάνουμε με το νέφος, το όνομα του νέου αντικειμένου όπως το ορίσαμε στο uploadresult.jsp και τέλος την τοποθεσία του root φακέλου μέσα στον οποίο βρίσκονται όλα τα αρχεία του κώδικα στον τοπικό διακομιστή.
Ο λόγος για τον οποίο χρειαζόμαστε αυτή την τοποθεσία είναι για να μπορούμε εύκολα και γρήγορα να έχουμε πρόσβαση σε οποιοδήποτε αρχείο βρίσκεται ή έχει έρθει στον τοπικό διακομιστή και είναι έτοιμο για να αποσταλεί στο νέφος.
Τέλος από το παραπάνω κομμάτι κώδικα φτιάχνουμε ένα αντικείμενο της κλάσης CloudConnection για να μπορέσουμε να έχουμε πρόσβαση στις λειτουργίες του νέφους. Μία από αυτές είναι η κλήση της getAuthToken η οποία γίνεται αμέσως μετά και μας επιστρέφει ένα πίνακα με το αποτέλεσμα της κλήσης από το νέφος και ένα Authorization Token το οποίο χρειαζόμαστε για οποιαδήποτε αλληλεπίδραση με το νέφος αποθήκευσης αντικειμένων.

Η μέθοδος **getAuthToken** της κλάσης CloudConnection φαίνεται στο παρακάτω κομμάτι κώδικα:

```
public String[] getAuthToken(){
        String[] result=new String[3];
        String token=null,storageurl=null;
        try {
            URL url = new URL (urlStr);
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setDoOutput(true);
            connection.setRequestMethod(method);
```

```
            connection.setRequestProperty("X-Storage-User",
"test:tester"); // to get an X-Auth-Token
            connection.setRequestProperty("X-Storage-Pass",
"testing"); // to get an X-Auth-Token
            //connection.setRequestProperty("X-Auth-Token",
"AUTH_tkfac274ed86294b36b6793062f6e83e7d");

            int responseCode = connection.getResponseCode();
            if (responseCode==200){
                result[0]="200 OK";
            }
            BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
            String inputLine;
            StringBuffer response = new StringBuffer();

            //getting the http response body
            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }
            storageurl=connection.getHeaderField(1);
            token=connection.getHeaderField(2);
            in.close();
            connection.disconnect();
        } catch (MalformedURLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        // result[0] holds the response code, result[1] holds the
token and result[2] holds the storage url
        result[1]= token;
        result[2]= storageurl;
        return result;
    }
```

Η συγκεκριμένη μέθοδος αρχικά δημιουργεί μία Httpurl σύνδεση με το νέφος και στην συνέχεια στέλνουμε σαν επικεφαλίδες HTTP το X-Storage-User και το X-Storage-Password. Μετά από αυτό διαβάζουμε την απάντηση του νέφους και αποθηκεύουμε σε ένα πίνακα το αποτέλεσμα της κλήσης και το Authorization Token όπως επίσης και το url στο οποίο θα γίνονται οι επόμενες συνδέσεις με το νέφος.

Για να δούμε λοιπόν το κομμάτι κώδικα της SendtoCloud κλάσης το οποίο είναι υπεύθυνο για για να καλέσει την ανάλογη μέθοδο της CloudConnection και έτσι να αποθηκεύσει τελικά το νέο αντικείμενο στο νέφος:

```
if (action.equals("create")){
                if (connresult[0]=="200 OK"){
```

```
                    result=cc.createObject(connresult[1],
filename, objname,rootDir);
                    request.setAttribute("response", result[1]);

request.getRequestDispatcher("/index.jsp").forward(request,
response);
                }else {
                    request.setAttribute("response", "Could not
retrieve authorization token from object cloud server");
                }
```

Εδώ εφόσον το action είναι create αρχικά παίρνει ένα Authorization token (με την getAuthToken όπως δείχθηκε προηγουμένως) και ελέγχει αν η απάντηση του νέφους είναι 200 OK. Αν ναι όλα πήγαν καλά και έχουμε ένα authorization token. Στην συνέχεια καλεί την μέθοδο createObject της κλάσης CloudConnection που όπως έχει προαναφερθεί είναι υπεύθυνη για να επικοινωνεί με το νέφος και να εκτελεί τις ενέργειες που ζητάμε.

Σαν ορίσματα στην μέθοδο αυτή δίνουμε το Authorization Token το όνομα του αρχείου που ανεβάσαμε το οποίο όπως είπαμε είναι αποθηκευμένο στο session της σύνδεσης, το όνομα του νέου αντικειμένου όπως ορίστηκε από την uploadresult.jsp και τέλος τον φάκελο root στον οποίο βρίσκεται το αρχείο από το οποίο θέλουμε να δημιουργήσουμε το νέο αντικείμενο.

Η μέθοδος **createObject** της CloudConnection φαίνεται παρακάτω:

```
public String[] createObject(String token, String filename,
String objname, String rootDir){
        String[][] meta;
        String[] result = new String[5];
        JSONExtract jsonex = new JSONExtract();
        String pathtojson=rootDir+"/JSONFiles/"+filename;
        try {
            meta =jsonex.ParseJSON(pathtojson);
            URL url = new URL
("http://83.212.121.116:8080/v1/AUTH_test/IoTcontainer/"
+objname);
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setDoOutput(true);
            connection.setRequestMethod("PUT");
            connection.setRequestProperty("Content-Type",
"multipart/form-data");
            connection.setRequestProperty("X-Auth-Token", token);
// use the X-Auth-Token you have
            connection.setRequestProperty("Content-
Length","200");
            //gemizw to request me ta meta tou neou object
            for (int i=0; i<meta.length; i++){
                connection.setRequestProperty("X-Object-Meta-
"+meta[i][0], meta[i][1]);
            }
```

```
            OutputStream outputStream =
connection.getOutputStream();
            //apostoli tou arxeiou json sto cloud
            File fileup = new
File(rootDir+"/JSONFiles/"+filename);//gia to anevasma olou tou
.json ston server pairnw to arxeio json
            FileInputStream streamFileInputStream = new
FileInputStream(fileup);
            BufferedInputStream streamFileBufferedInputStream =
new BufferedInputStream(streamFileInputStream);
            byte[] streamFileBytes = new byte[(int)
fileup.length()];
            int bytesRead = 0;
            int totalBytesRead = 0;
            while ((bytesRead =
streamFileBufferedInputStream.read(streamFileBytes)) > 0) {
                outputStream.write(streamFileBytes, 0,
bytesRead);
                outputStream.flush();
                totalBytesRead += bytesRead;
            }
            streamFileBufferedInputStream.close();
            outputStream.close();

            int responseCode = connection.getResponseCode();
            System.out.println("\nSending 'PUT' request to URL :
" + url);
            System.out.println("Response Code : " +
responseCode);
            //to result[0] exei to response code
            result[0]=""+responseCode;
            BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
            String inputLine;
            StringBuffer response = new StringBuffer();

            //getting the http response body
            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }

            //to result[1] exei tin apadisi tou server
            result[1]=connection.getHeaderField(0);

            in.close();
            connection.disconnect();
        } catch (MalformedURLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        catch (JSONException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
```

```
        }
        return result;
    }
```

Η μέθοδος αυτή είναι υπεύθυνη για την δημιουργία του αντικειμένου στο νέφος αποθήκευσης αντικειμένων.
Το όνομα του container στο οποίο θα δημιουργηθεί το αντικείμενο βρίσκεται στο τέλος του url. Σαν μέθοδος http ορίστηκε η PUT έτσι ώστε το νέφος να καταλάβει ότι πρόκειται για δημιουργία αντικειμένου.
Αρχικά φτιάχνει ένα πίνακα για να αποθηκεύσει το αποτέλεσμα αυτής της διενέργειας το οποίο θα επιστρέψει ο διακομιστής του νέφους. Στην συνέχεια φτιάχνει ένα αντικείμενο της JSONExtract που όπως έχει προαναφερθεί είναι υπεύθυνη για το JSON parsing των αρχείων .json. Δηλαδή παίρνει το αρχείο που ανεβάζουμε και καλώντας την μέθοδο parseJSON που παίρνει σαν παράμετρο την τοποθεσία του αρχείου το αποσυνθέτει και αποθηκεύει σε ένα πίνακα όλα τα δεδομένα του σε μορφή (key : value) ώστε να μπορούμε να τα αποθηκεύσουμε ευκολότερα σαν μεταδεδομένα στο καινούργιο αντικείμενο που θα δημιουργηθεί στο νέφος. Ο κώδικας και η αναλυτική εξήγηση για την λειτουργία του JSONExtract θα δωθεί παρακάτω.
Αφού λοιπόν αποθηκεύσουμε όλα τα δεδομένα σε ένα πίνακα μέσω της JSONExtract δημιουργείται μία Httpurlconnection δηλαδή μία σύνδεση με το νέφος μέσω http και αφού ορίσουμε το μήκος και το Authorization Token σαν επικεφαλίδες γίνεται η επόμενη επανάληψη:

```
for (int i=0; i<meta.length; i++){
                connection.setRequestProperty("X-Object-Meta-
"+meta[i][0], meta[i][1]);
            }
```

Στην επανάληψη αυτή παίρνουμε ένα ένα τα δεδομένα από τον πίνακα που φτιάξαμε και γεμίσαμε προηγουμένως και τα αποθηκεύουμε σαν μεταδεδομένα του αντικειμένου βάζοντας τα σαν επικεφαλίδες.
Σημαντικό να σημειωθεί είναι πως οι επικεφαλίδες για τα μεταδεδομένα στην http σύνδεση θα πρέπει να ξεκινάνε με “X-Object-Meta-” και στην συνέχεια το όνομα.
Αφού λοιπόν βάλουμε όλα τα μεταδεδομένα δημιουργούμε ένα OutputStream μέσω του οποίου θα ανέβει όλο το JSON αρχείο σαν σώμα (body) του αντικειμένου στο νέφος. Έτσι στην συνέχεια το περνάμε στο σώμα (HTTP body) της http κλήσης για να το στείλουμε στο νέφος.
Τέλος παίρνοντας το αποτέλεσμα βλέπουμε αν ήταν επιτυχής η μεταφορά ή εάν απέτυχε. Σε οποιαδήποτε περίπτωση επιστρέφεται το αποτέλεσμα μέσω του πίνακα αποτελεσμάτων που αναφέραμε προηγουμένως.
Έτσι λοιπόν έχει δημιουργηθεί ένα αντικείμενο στο νέφος.

4.2.2 Διαγραφή αντικειμένου από το νέφος αποθήκευσης

Η επόμενη λειτουργία η οποία φαίνεται στην παρακάτω εικόνα είναι η διαγραφή ενός αντικειμένου από το νέφος:

Εικόνα 4.2.4 Φόρμα διαγραφής αντικειμένου

Ο κώδικας για την εμφάνιση της διεπαφής αυτής είναι ο παρακάτω:

```
<div id="delete">
      <h3>Object Deletion</h3>
      <form autocomplete="off" action="SendtoCloud" method="post">
      <input type="radio" name="action" value="delete" checked>Delete
Object<br>
                  Insert the name of the object you wish to Delete:
<input type="text" name="objname" size="20">
                  <input type="submit" value="Delete" />
      </form>
      <br>
      Cloud server response: ${requestScope["responsed"]}
</div>
```

Πάλι το κομμάτι το οποίο μας ενδιαφέρει πιο πολύ είναι το ότι όταν αυτή η φόρμα προσωρήσει θα κληθεί πάλι η κλάση SendtoCloud με action = delete.

Με βάση τα παραπάνω θα κληθεί το κομμάτι του κώδικα της SendtoCloud που φαίνεται εδώ:

```
else if (action.equals("delete")){
                  if (connresult[0]=="200 OK"){
                      result=cc.deleteObject(connresult[1],
objname);
                      request.setAttribute("responsed", result[1]);

request.getRequestDispatcher("/index.jsp").forward(request,
response);
                  }else {
                      request.setAttribute("response", "Could not
retrieve authorization token from object cloud server");}}
```

Εφόσον δηλαδή το action ήταν delete θα μπούμε σε αυτό το else if και αφόσον καταφέρουμε να πάρουμε το Authorization Token καλούμε την μέθοδο deleteObject της κλάσης CloudConnection. Αυτή η μέθοδος παίρνει σαν ορίσματα το Authorization Token και το όνομα του αντικειμένου το οποίο θέλουμε να διαγράψουμε από το νέφος αποθήκευσης.
Αν γίνει η διαγραφή και έχουμε επιτυχία επιστρέφουμε το αποτέλεσμα πίσω στην διεπαφή σαν απάντηση από το νέφος.

Όσο αφορά την **deleteObject** μέθοδο αυτή φαίνεται εδώ:

```
public String[] deleteObject(String token, String objname){
        String[] result = new String[5];
        try {
            URL url = new URL
("http://83.212.121.116:8080/v1/AUTH_test/IoTcontainer/"
+objname);
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setDoOutput(true);
            connection.setRequestMethod("DELETE");
            connection.setRequestProperty("X-Auth-Token", token);
// use the X-Auth-Token you have
            int responseCode = connection.getResponseCode();
            System.out.println("\nSending 'DELETE' request to URL
: " + url);
            System.out.println("Response Code : " +
responseCode);
            //to result[0] exei to response code
            result[0]=""+responseCode;
            BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
            String inputLine;
            StringBuffer response = new StringBuffer();

            //getting the http response body
            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }

            //to result[1] exei tin apadisi tou server
            result[1]=connection.getHeaderField(0);

            in.close();
            connection.disconnect();
        } catch (MalformedURLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return result;
```

```
}
```

Όπως και με την δημιουργία αντικειμένου πρέπει να κάνουμε μία σύνδεση http μέσω url. Αφού εδραιωθεί η σύνδεση στέλνουμε όπως πάντα το Authorization Token με μόνη διαφορά ότι δεν χρειάζεται να στείλουμε άλλες επικεφαλίδες. Το μόνο που πρέπει να κάνουμε είναι να βάλουμε σαν url της κλήσης το url:

"http://83.212.121.116:8080/v1/AUTH_test/IoTcontainer/" +objname

δηλαδή το url που βρίσκεται το νέφος, το account name το container και τέλος στο url πρέπει να υπάρχει το όνομα του αντικειμένου που θέλουμε να διαγράψουμε.
Η μέθοδος http είναι η DELETE οπότε το νέφος ξέρει ότι θα πρέπει να διαγράψει το αντικείμενο.
Στην συνέχεια απλά παίρνουμε το αποτέλεσμα της διαγραφής από το νέφος και το επιστρέφουμε.

4.2.3 Απεικόνιση αντικειμένων που βρίσκονται στο νέφος αποθήκευσης

Είναι σημαντικό, για να μπορεί να κάνει όλες τις αλληλεπιδράσεις ο χρήστης με το νέφος, να μπορεί να βλέπει όλα τα αντικείμενα τα οποία βρίσκονται ανά πάσα στιγμή στο νέφος.
Η απεικόνιση των αντικειμένων φαίνεται στην παρακάτω εικόνα:

Object Listing

The objects currently stored in the cloud server are:

Last Modified	Object Size(bytes)	Object Name
2014-05-30T15:12:17.596060	1373	joinedtimefile.json
2014-06-02T09:11:30.548700	1373	joinedtimefilenew.json
2014-05-12T09:06:13.079660	1355	newobj
2014-04-07T10:00:18.301260	1355	objnewnew
2014-05-12T09:23:50.643610	1355	test242
2014-05-12T09:33:07.300080	1355	test400
2014-04-08T17:34:11.455880	1355	testobj
2014-04-03T10:41:57.447050	1355	testobj4
2014-05-30T13:53:22.803930	1356	timefile.json
2014-05-30T14:42:38.565380	1373	timefile2.json
2014-06-02T09:10:01.096320	696	timestamp0
2014-06-02T09:10:01.160340	696	timestamp1

Εικόνα 4.2.5 Εμφάνιση αντικειμένων νέφους

Ο κώδικας αυτής της διεπαφής δίνεται εδώ:

```
<div id="list">
            <h3>Object Listing</h3>
            <%    ArrayList<String> objlist = new ArrayList<String>();
                  CloudConnection cc = new CloudConnection();
                  objlist=cc.listObjects("IoTcontainer");
                  int i=0;
            %>
            The objects currently stored in the cloud server are:
            <table border=1 id="objtable">
                  <tr><th>Last Modified</th><th>Object
Size(bytes)</th><th>Object Name</th></tr>
                  <tr><% for (String s : objlist) { %>
                        <% i=i+1;%>
                        <td><%= s %></td>
                        <%    if ((i % 3)==0){%>
                              </tr><tr>
                        <%} %>
                  <%} %>
                  </tr>
            </table>
</div>
```

Εδώ βλέπουμε την γλώσσα προγραμματισμού Java η οποία περιέχεται μέσα στον κώδικα HTML. Αυτό αποκαλείται JSP.
Εδώ φτιάχνουμε μία λίστα με String μέσα στην οποία αποθηκεύονται τα στοιχεία των αντικειμένων που βρίσκονται στο νέφος. Στην συνέχεια απεικονίζουμε αυτά τα στοιχεία φτιάχνοντας έναν πίνακα και γεμίζοντας τον με ένα for loop.
Η λίστα γεμίζει με τα στοιχεία αυτά καλώντας την μέθοδο listObjects της κλάσης CloudConnection. Η listObjects παίρνει μία παράμετρο και αυτή είναι το container στο οποίο θέλουμε να δούμε τα αντικείμενα.
Η μέθοδος listObjects φαίνεται παρακάτω:

```
public ArrayList<String> listObjects (String container){
        ArrayList<String> objlist = new ArrayList<String>();
        String[] result=new String[3];
        //pairnw to auth token kai to exw sto conn[1]
        String[] conn = new String[3];
        //pairnw to token gia na kanw to GET
        conn=getAuthToken();
        URL url;
        try {
            url = new URL
("http://83.212.121.116:8080/v1/AUTH_test/"+container+"?format=js
on");
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            //connection.setDoOutput(true);
            connection.setRequestMethod("GET");
```

```
            connection.setRequestProperty("X-Auth-Token",
conn[1]);
            connection.setRequestProperty("Content-Length","0");
            //to result[0] exei to response code
            int responseCode = connection.getResponseCode();
            System.out.println("\nSending 'GET' request to URL :
" + url);
            System.out.println("Response Code : " +
responseCode);
            result[0]=""+responseCode;
            //pairnw to body tou response pou tha periexei ta
objects
            BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
            String inputLine;
            StringBuffer response = new StringBuffer();

            //getting the http response body
            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }
            //to result[1] exei ta object names
            result[1]=response.toString();
            result[2]=connection.getHeaderField(2);
            String[][] objinfo = new
String[Integer.parseInt(result[2])][3];
            JSONExtract jex = new JSONExtract();
            objinfo = jex.objJSONparse(result[1], result[2]);
            for (int i=0; i<Integer.parseInt(result[2]); i++){
                objlist.add(objinfo[i][0]);
                objlist.add(objinfo[i][1]);
                objlist.add(objinfo[i][2]);
            }
            in.close();
            connection.disconnect();
        } catch (MalformedURLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return objlist;
}
```

Αρχικά φτιάχουμε μία λίστα για να αποθηκεύσουμε τα δεδομένα όλων των αντικειμένων. Στην συνέχεια δημιουργούμε μία σύνδεση με το νέφος και στο url βάζουμε στο τέλος το όνομα του container στο οποίο θέλουμε να δούμε τα αντικείμενα που έχει.
Η μέθοδος http που στέλνουμε στο νέφος είναι η GET έτσι ώστε το νέφος να μας επιστρέψει όλα τα αντικείμενα του container που προσδιορίσαμε.

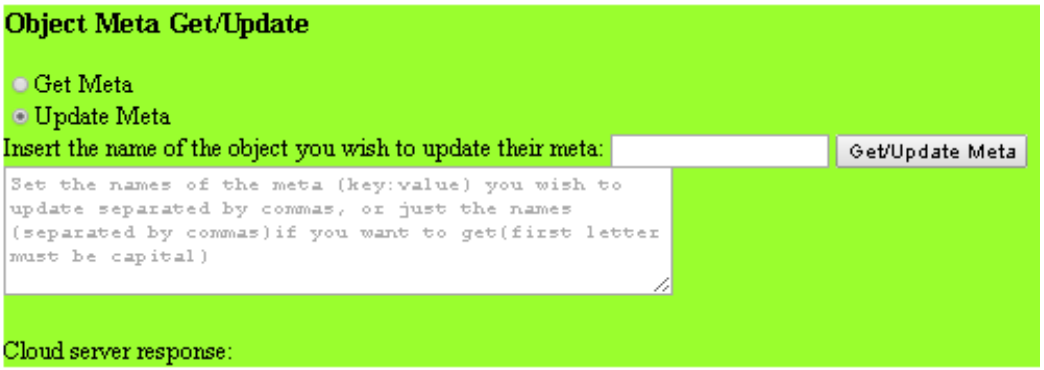
Αφού στείλουμε την http κλήση φτιάχνουμε ένα InpuStreamReader με έναν BufferedReader για να πάρουμε τα δεδομένα από το νέφος τα οποία βρίσκονται στο σώμα του http response(απάντησης).
Αφού τα πάρουμε τα αποθηκεύουμε κάνοντας μία επανάληψη for στην λίστα και τέλος επιστρέφουμε την λίστα.

4.2.4 Ανάκτηση και ενημέρωση των μεταδεδομένων των αντικειμένων

Μία από τις πιο σημαντικές λειτουργίες είναι αυτή που παρουσιάζεται σε αυτήν την παράγραφο. Η ανάκτηση και η ενημέρωση των μεταδεδομένων των αντικειμένων που βρίσκονται ήδη αποθηκευμένα στο νέφος αποθήκευσης αντικειμένων.

- **Ανάκτηση μεταδεδομένων**

Αρχικά θα αναλυθεί η **ανάκτηση μεταδεδομένων**. Η εφαρμογή είναι προγραμματισμένη έτσι ώστε ο χρήστης να έχει την δυνατότητα να ανακτά συγκεκριμένα μεταδεδομένα τα οποία θέλει από οποιοδήποτε αντικείμενο του container.
Όπως φαινεται στην παρακάτων εικόνα της διεπαφής για την ανάκτηση ή την ενημέρωση ο χρήσης αρχικά επιλέγει την ενέργεια που θέλει (Get Meta = ανάκτηση μεταδεδομένων ή Update Meta = ενημέρωση μεταδεδομένων) και στην συνέχεια εισάγει στο μικρό κενό το όνομα του αντικειμένου που θέλει να ανακτήσει/ενημερώσει. Στο μεγάλο κενό εισάγει τα μεταδεδομένα των οποίων την τιμή θέλει να ανακτήσει με την μορφή: key, key, key, … και του επιστρέφονται τα values δηλαδή οι τιμές.



Εικόνα 4.2.6 Φόρμα ενημέρωσης/ανάκτησης μεταδεδομένων

Ο κώδικας της παραπάνω διεπαφής φαίνεται εδώ:

```
<div id="meta">
            <h3>Object Meta Get/Update</h3>
            <form id="metadata" autocomplete="off" action="SendtoCloud"
method="post">
            <input type="radio" name="action" value="getmeta"
checked>Get Meta<br>
            <input type="radio" name="action" value="updatemeta"
checked>Update Meta<br>
                  Insert the name of the object you wish to update
their meta: <input type="text" name="objname" size="20">
                  <input type="submit" value="Get/Update Meta" />
            </form>
            <textarea name="objmeta" form="metadata" rows="5" cols="50"
placeholder=" to get(first letter must be capital)"></textarea>
            <br><br>Cloud server response: ${requestScope["responseu"]}
</div>
```

Εδώ το action που αναφέρθηκε προηγουμένως παίρνει την τιμή getmeta αν επιλεχθεί η ανάκτηση μεταδεδομένων ή την τιμή updatemeta αν επιλεχθεί η ενημέρωση μεταδεδομένων.
Στην συνέχεια στέλνεται στην SendtoCloud κλάση η οποία είναι υπεύθυνη για την κλήση της σωστής μεθόδου για την επικοινωνία με το νέφος.

Παρακάτω φαίνεται το κομμάτι του κώδικα της SendtoCloud κλάσης το οποίο είναι υπεύθυνο γι' αυτό:

```
else if (action.equals("getmeta")){
                  if (connresult[0]=="200 OK"){
                        request.setAttribute("objname", objname);
                        String metadata =
request.getParameter("objmeta");
                        System.out.println(metadata);
                        if (metadata!=null){
                              ArrayList<String> metadatafromobjserver =
cc.listMeta(connresult[1], objname, "IoTcontainer");//pairnw apo
ton server ola ta meta se lista
                              ArrayList<String> metaresult = new
ArrayList<>(); //tha apothikeuw auta pou tha deiksw telika
                              String metadataarray[]=
metadata.split(",");//parnw ena pinaka me ta obj name pou exei
dwsei o xrhsths
                              for (int
z=0;z<metadatafromobjserver.size();z++){
                                    String metafromobjserver =
metadatafromobjserver.get(z);
                                    String[] getkey =
metafromobjserver.split(":");//getkey[0] exei to key toy kathe
meta apo ton object server
```

```
                                for (int
k=0;k<metadataarray.length;k++){
                                    if (("X-Object-Meta-
"+metadataarray[k]).equals(getkey[0].replaceAll("\\s",""))){

metaresult.add(getkey[0]+":"+getkey[1]);//kanw loop kai krataw
mono auta pou exei zhthsei o xrhsths
                                    }
                                }
                            }
                            request.setAttribute("metalist",
metaresult);

request.getRequestDispatcher("/listmeta.jsp").forward(request,
response);
                        }else{
                            request.setAttribute("responseu", "Please
set at least one meta to update as: 'key:value' or : 'key' if you
want to get");

request.getRequestDispatcher("/index.jsp").forward(request,
response);
                        }
                    }else {
                        request.setAttribute("response", "Could not
retrieve authorization token from object cloud server");
                    }
                }
```

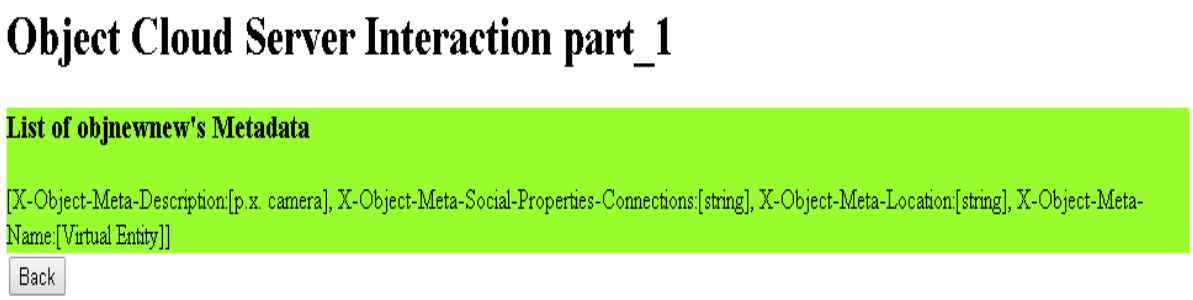
Στο κομμάτι κώδικα αυτό καλείται η listMeta μέθοδος της CloudConnection κλάσης η οποία παίρνει σαν ορίσματα το Authorization Token, το όνομα του αντικειμένου του νέφους του οποίου τα μεταδεδομένα θέλουμε να ανακτήσουμε και το όνομα του container στο οποίο περιέχεται το αντικείμενο αυτό.
Η listMeta λοιπόν (η οποία θα δούμε πως δουλεύει παρακάτω) επιστρέφει μία λίστα με τα μεταδεδομένα του αντικειμένου. Επειδή δεν υπάρχει κάποια μέθοδος του API του νέφους για να παίρνουμε συγκεκριμένα μόνο τα μεταδεδομένα που θέλουμε αναγκαστικά τα παίρνουμε όλα και μετά φιλτράρονται από τον κώδικα που φαίνεται παραπάνω μετά την κλήση της listMeta.
Κάνοντας μία επανάληψη ελέγχω τα στοιχεία της λίστας που επιστράφηκαν από το νέφος με τα μεταδεδομένα που ζήτησε ο χρήστης και επιστρέφονται μόνο αυτά που έχουν ζητηθεί. Έτσι η διεπαφή είναι πιο σαφής και δεν περιέχει περιττές πληροφορίες.

Ο κώδικας της listMeta φαίνεται εδώ:

```
public ArrayList<String> listMeta (String token, String objname,
String container){
        ArrayList<String> metalist = new ArrayList<String>();
        try {
```

```
            URL url = new URL
("http://83.212.121.116:8080/v1/AUTH_test/"+container+"/"
+objname);
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setDoOutput(true);
            connection.setRequestMethod("HEAD");
            connection.setRequestProperty("X-Auth-Token", token);
            int responseCode = connection.getResponseCode();
            System.out.println("\nSending 'HEAD' request to URL :
" + url);
            System.out.println("Response Code : " +
responseCode);
            Map headerfields = connection.getHeaderFields();
            Set headers = headerfields.entrySet();
            for(Iterator i = headers.iterator(); i.hasNext();){
                Map.Entry map = (Map.Entry)i.next();
                //System.out.println(map.getKey() + " : " +
map.getValue() + "");
                metalist.add(map.getKey()+":"+map.getValue());
            }
            connection.disconnect();
        } catch (MalformedURLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return metalist;
}
```

Είναι αρκετά απλός και το μόνο που χρειάζεται να σημειωθεί είναι πως στέλνουμε στο νέφος με την μέθοδο http HEAD για να μας επιστρέψει μεταδεδομένα. Η εμφάνιση των μεταδεδομένων στον χρήστη φαίνεται στην παρακάτω εικόνα:



Εικόνα 4.2.7 Λίστα εμφάνισης μεταδεδομένων

Εδώ ζητήσαμε να μας φέρει μεταδεδομένα από το αντικείμενο objnewnew και συγκεκριμένα ζητήσαμε να μας φέρει τα:
Description, Social-Properties-Connections,Location και τέλος το Name.

- **Ενημέρωση μεταδεδομένων**

Για την ενημέρωση των αντικειμένων έχοντας επιλέξει αντίστοιχα το update meta στην φόρμα που είδαμε στην αρχή καλείται η SendtoCloud με action = updatemeta .Ο κώδικας που καλείται είναι ο εξής:

```
else if (action.equals("updatemeta")){
                if (connresult[0]=="200 OK"){
                    /* pairnw ta metadata pou tha kanw update kai
ta pernaw apo ena string se mia lista me kathe stoixeio na exei
morfi key:value
                     * meta kalw tin sunartish poy kanei update
kai vrisketai sto CloudConnection.java */
                    String metadata =
request.getParameter("objmeta");
                    if (metadata!=null){
                        String metadataarray[]=
metadata.split(",");
                        result=cc.updateMeta(connresult[1],
objname,metadataarray);
                        request.setAttribute("responseu",
result[1]);//i apadisi tou server gia na thn deiksw

request.getRequestDispatcher("/index.jsp").forward(request,
response);
                    }else {
                        request.setAttribute("responseu", "Please
set at least one meta to update as: 'key:value'");

request.getRequestDispatcher("/index.jsp").forward(request,
response);
                    }
                }else {
                    request.setAttribute("response", "Could not
retrieve authorization token from object cloud server");
                }
}
```

Στον οποίο το σημαντικό κομμάτι είναι ότι καλείται η μέθοδος **updateMeta** της CloudConnection η οποία φαίνεται εδώ:

```
public String[] updateMeta(String token, String objname, String[]
update_meta_list){
        String[] result = new String[5];
```

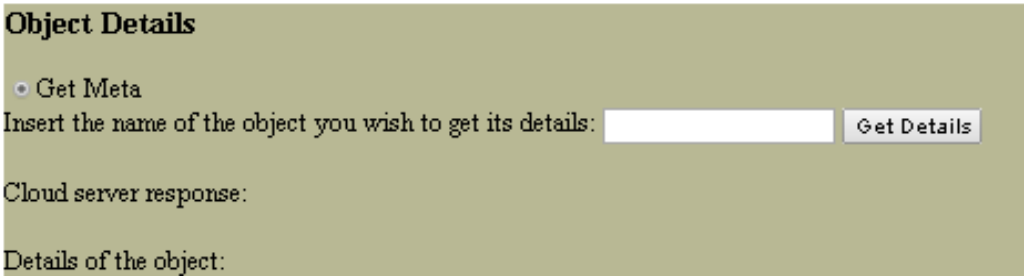
```
        try {
            URL url = new URL
("http://83.212.121.116:8080/v1/AUTH_test/IoTcontainer/"
+objname);
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setDoOutput(true);
            connection.setRequestMethod("POST");
            connection.setRequestProperty("X-Auth-Token", token);
// use the X-Auth-Token you have
            for (int i=0; i<update_meta_list.length; i++){
                String[] keyvalue =
update_meta_list[i].split(":");//o pinakas autos exei sti thesi 0
to key kai stin 1 to value
                connection.setRequestProperty("X-Object-Meta-
"+keyvalue[0], keyvalue[1]);//ta vazw san header gia to update
            }
            int responseCode = connection.getResponseCode();
            System.out.println("\nSending 'POST' request to URL :
" + url);
            System.out.println("Response Code : " +
responseCode);
            //to result[0] exei to response code
            result[0]=""+responseCode;
            BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
            String inputLine;
            StringBuffer response = new StringBuffer();

            //getting the http response body
            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }

            //to result[1] exei tin apadisi tou server
            result[1]=connection.getHeaderField(0);

            in.close();
            connection.disconnect();
        } catch (MalformedURLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return result;
}
```

Με την συγκεκριμένη μέθοδο στέλνουμε ένα http POST στο νέφος βάζοντας σαν επικεφαλίδες εκτός από το Authorization Token (το οποίο τοποθετείται πάντα σαν επικεφαλίδα όταν μιλάμε με το νέφος), τα μεταδεδομένα τα οποία θέλουμε να

ενημερώσουμε. Αυτά τα παρέχει ο χρήστης με την μορφή που είπαμε προηγουμένως δηλαδή “key : value” ζευγάρια.
Το μόνο πρόβλημα είναι ότι ο χρήστης ακόμα και να μην θέλει να ενημερώσει όλα τα μεταδεδομένα ενός αντικειμένου θα πρέπει να τα στείλει όλα. Αυτα τα οποία δεν θέλει να ενημερώσει θα πρέπει να τα στέλνει πάλι έτσι όπως ήταν. Αυτό το πρόβλημα προκύπτει διότι το νέφος αποθήκευσης αντικειμένων της Openstack δεν έχει κάποια μέθοδο στο API του έτσι ώστε να μας δίνεται η δυνατότητα να κάνουμε επιλεκτική ενημέρωση στα μεταδεδομένα ενός αντικειμένου στο νέφος.
Στην συνέχεια εφόσον στείλουμε το αίτημα παίρνουμε και επιστρέφουμε την απάντηση τους νέφους πίσω στον χρήστη.

4.2.5 Εμφάνιση μεταδεδομένων και κύριου σώματος ενός αντικειμένου

Η διεπαφή/φόρμα μέσω της οποίας ο χρήστης μπορεί να δεί όλες της πληροφορίες που περιέχει ένα αντικείμενο στο νέφος φαίνεται στην παρακάτω εικόνα:



Εικόνα 4.2.8 Φόρμα ανάκτησης δεδομένων αντικειμένου

Ο κώδικας της φόρμας αυτής είναι ο εξής:

```
<div id="details">
            <h3>Object Details</h3>
            <form id="metadata" autocomplete="off" action="SendtoCloud"
method="post">
            <input type="radio" name="action" value="getdetails"
checked>Get Meta<br>
                  Insert the name of the object you wish to get its
details: <input type="text" name="objname" size="20">
                  <input type="submit" value="Get Details" />
            </form>
            <br>Cloud server response: ${requestScope["responsedet"]}
```

```
            <br><br>Details of the object: <%if
(request.getAttribute("objdetails")!=null){ %>
            <% ArrayList<String> objdetails = (ArrayList<String>)
request.getAttribute("objdetails");%>
            <%int j=0; %>
                  <% for (String s : objdetails) { %>
                        <% j=j+1;%>
                        <br>
                        <%= s %>
                  <%} %>
            <%} %>
</div>
```

Με τον κώδικα αυτό αφού σταλεί στην κλάση SendtoCloud όπως φαίνεται στο form με action = getdetails επιστρέφεται από αυτήν μία λίστα η οποία περιέχει όλα τα δεδομένα του αντικειμένου. Μετά με μία επανάληψη δημιουργείται ένας πίνακας και γεμίζει με τα δεδομένα αυτά.

Το κομμάτι κώδικα της SendtoCloud που καλείται είναι το εξής:

```
else if (action.equals("getdetails")){
                  if (connresult[0]=="200 OK"){
                        ArrayList<String>
details=cc.getObject(objname);
                        request.setAttribute("objdetails", details);

request.getRequestDispatcher("/index.jsp").forward(request,
response);
                  }else {
                        request.setAttribute("responsedet", "Could
not retrieve authorization token from object cloud server");
                  }
}
```

και το μόνο που κάνει είναι να καλεί την μέθοδο getObject της κλάσης CloudConnection. Η getObject παίρνει σαν παράμετρο μόνο το όνομα του αντικειμένου του οποίου τα δεδομένα θέλουμε να εμφανίσουμε.

Η μέθοδος **getObject** φαίνεται παρακάτω:

```
public ArrayList<String> getObject (String objname){
        ArrayList<String> objdetails = new ArrayList<String>();
        //swzw topika metavlites pou pernw apo to GET ston server
gia ta objects
        String[] result=new String[3];
        String container="IoTcontainer";
        //pairnw to auth token kai to exw sto conn[1]
        String[] conn = new String[3];
```

```
        //pairnw to token gia na kanw to GET
        conn=getAuthToken();
        URL url;
        try {
            url = new URL
("http://83.212.121.116:8080/v1/AUTH_test/"+container+"/"+objname
+"?format=json");
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            //connection.setDoOutput(true);
            connection.setRequestMethod("GET");
            connection.setRequestProperty("X-Auth-Token",
conn[1]);
            connection.setRequestProperty("Content-Length","0");
            //to result[0] exei to response code
            int responseCode = connection.getResponseCode();
            System.out.println("\nSending 'GET' request to URL :
" + url);
            System.out.println("Response Code : " +
responseCode);
            result[0]=""+responseCode;
            //pairnw to body tou response pou tha periexei ta
objects
            BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
            String inputLine;
            StringBuffer response = new StringBuffer();

            //getting the http response body
            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }
            //to result[1] exei ta object names
            result[1]=response.toString();
            //to result[2] exei ton arithmo twn objects pou
xreiazomai gia na kanw swsta to parse tou json meta
            result[2]=connection.getHeaderField(2);
            Map headerfields = connection.getHeaderFields();
            Set headers = headerfields.entrySet();
            for(Iterator it = headers.iterator(); it.hasNext();){
                Map.Entry map = (Map.Entry)it.next();
                objdetails.add(map.getKey()+" :
"+map.getValue());
            }
            objdetails.add(result[1]);
            in.close();
            connection.disconnect();
        } catch (MalformedURLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return objdetails;
}
```

Με αυτήν τη μέθοδο στέλνουμε ένα http GET στο νέφος προσδιορίζοντας το όνομα του αντικειμένου στο url και στην συνέχεια διαβάζουμε το body του http response το οποίο περιέχει όλο το κύριο σώμα του αντικειμένου. Τέλος διαβάζουμε και τις επικεφαλίδες οι οποίες περιέχουν τα μεταδεδομένα. Όλα αυτά αποθηκεύονται σε μία λίστα για να επιστραφούν στο χρήστη μέσω του προηγούμενου κώδικα jsp.

Το αποτέλεσμα φαίνεται εδώ:

Εικόνα 4.2.9 Λίστα δεδομένων αντικειμένου

όπου πήραμε όλα τα δεδομένα του αντικειμένου objnewnew.
Με αυτό το κομμάτι ολοκληρώνεται το πρώτο μέρος της υλοποίησης.
Το επόμενο κομμάτι αφορά κάποιες επιπλέον λειτουργίες ξανά σε αρχεία .json και αντικείμενα στο νέφος αποθήκευσης.
Το είδος των .json αρχείων τα οποία χρησιμοποιήθηκαν και μέσω αυτών φτιάχτηκαν τα αντικείμενα στο νέφος καθώς και ο τρόπος μαζί με τις μεθόδους που χρησιμοποιήθηκαν για να αναλυθούν αναφέρονται στην ενότητα 4.4.

Ο λόγος που δεν γίνεται η αναλυτική αναφορά τους νωρίτερα είναι γιατί ο κώδικας μπορεί πολύ εύκολα να αλλαχθεί να λαμβάνει οποιασδήποτε μορφής αρχεία JSON τα οποία θα αναλύει και θα αποθηκεύει στο νέφος.

4.3. Ανάλυση της υλοποίησης του δεύτερου μέρους της εφαρμογής

Το δεύτερο κομμάτι της εφαρμογής αφορά περεταίρω διεργασίες πάνω σε αρχεία και αποθήκευση αντικειμένων στο νέφος.
Η διεπαφή του δεύτερου μέρους με τις φόρμες για τις διάφορες λειτουργίες φαίνεται στην παρακάτω εικόνα:

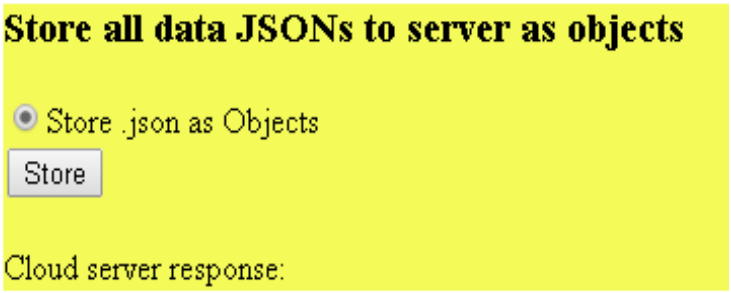
Εικόνα 4.3.1 Διαπαφή δεύτερου μέρους

Η μόνη λειτουργία που υπάρχει ξανά σε αυτή την διεπαφή είναι η εμφάνιση των αντικειμένων. Αυτό το οποίο αλλάζει είναι ότι δείχνουμε τα αντικείμενα στο άλλο container το οποίο είναι για το δεύτερο μέρος της εργασίας. Το container αυτό είναι το JsonStream.

4.3.1 Δημιουργία πολλαπλών αντικειμένω από πολλά αρχεία JSON

Στην πρώτη λειτουργία αυτή σκοπός είναι να προσομοιωθεί ένα "ποτάμι" (stream) πληροφοριών. Αυτές είναι αρχεία JSON τα οποία έρχονται απευθείας στον τοπικό διακομιστή και αποθηκεύονται σε ένα φάκελο. Ξανά όπως και πριν δεν είναι θέμα της παρούσας διπλωματικής ο τρόπος με τον οποίο τα πράγματα (συσκευές) του διαδικτύου των πραγμάτων στέλνουν τις πληροφορίες στον τοπικό διακομιστή.
Έτσι θεωρούμε ότι αποθηκεύονται σε ένα τοπικό φάκελο και από εκεί παίρνουμε τα JSON και φτιάχνουμε τόσα αντικείμενα όσα και τα αρχεία.

Η διεπαφή του χρήστη για αυτήν την λειτουργία είναι η εξής:



Εικόνα 4.3.2 Φόρμα αποθήκευσης πολλαπλών αρχείων

Το μόνο που έχει να κάνει ο χρήστης είναι να πατήσει το κουμπί Store και αυτόματα όλα τα αρχεία που βρίσκονται στον φάκελο που αναφέρθηκε προηγουμένως γίνονται αντικείμενα στο νέφος.

Ο κώδικας της παραπάνω φόρμας φαίνεται εδώ:

```
<div id="upload">
            <h3>Store all data JSONs to server as objects</h3>
            <form autocomplete="off" action="SendtoCloud"
method="post">
```

```
            <input type="radio" name="action" value="datajsonstore"
checked>Store .json as Objects<br>
                <input type="submit" value="Store" />
            </form>
            <br>
      Cloud server response: ${requestScope["response"]}
</div>
```

Εδώ μας ενδιαφέρει το κομμάτι της φόρμας στο οποίο καλείται πάλι η κλάση SendtoCloud. Έτσι στέλνεται στην κλάση αυτή το action με τιμή datajsonstore.
Παρακάτω φαίνεται το κομμάτι της SendtoCloud το οποίο αναλαμβάνει την επικοινωνία με το νέφος και την εκτέλεση αυτής της ενέργειας:

```
else if (action.equals("datajsonstore")){
                if (connresult[0]=="200 OK"){

result=cc.createDataObjects(connresult[1],rootDir);
                    request.setAttribute("response", result[1]);

request.getRequestDispatcher("/datastream_actions.jsp").forward(r
equest, response);
                }else {
                    request.setAttribute("response", "Could not
retrieve authorization token from object cloud server");
                }
}
```

Όπως φαίνεται παραπάνω καλείται η μέθοδος createDataObjects της CloudConnection. Η μέθοδος αυτή παίρνει σαν παράμετρους το Authorization Token και την τοποθεσία του φακέλου root της εφαρμογής για να μπορεί να βρει τον φάκελο στον οποίο είναι αποθηκευμένα τα αρχεία JSON.

Η μέθοδος **createDataObjects** της κλάσης CloudConnection φαίνεται εδώ:

```
public String[] createDataObjects(String token, String rootDir){
        String[][] meta;
        String[] result = new String[5];
        JSONExtract jsonex = new JSONExtract();
        File[] jsondatafiles = new
File(rootDir+"/JSONFiles/DataJSONStream").listFiles();
        for (File file : jsondatafiles) {
            String
pathtojson=rootDir+"/JSONFiles/DataJSONStream/"+file.getName();
            try {
                meta =jsonex.ParseJSONstream(pathtojson);
                String objname="datajson_"+meta[0][1];//gia na
exei to kathe object ston server diaforetiko onoma (efoson to id
einai pada diaforetiko)
```

```
                URL url = new URL
("http://83.212.121.116:8080/v1/AUTH_test/JsonStream/" +objname);
                HttpURLConnection connection =
(HttpURLConnection) url.openConnection();
                connection.setDoOutput(true);
                connection.setRequestMethod("PUT");
                connection.setRequestProperty("Content-Type",
"multipart/form-data");
                connection.setRequestProperty("X-Auth-Token",
token); // use the X-Auth-Token you have
                connection.setRequestProperty("Content-
Length","100");
                //gemizw to request me ta meta tou neou object
                for (int i=0; i<2; i++){
                    connection.setRequestProperty("X-Object-Meta-
"+meta[i][0], meta[i][1]);
                }
                JSONObject jsonobj = new JSONObject();
                jsonobj.put(meta[2][0], meta[2][1]);
                jsonobj.put(meta[3][0], meta[3][1]);
                jsonobj.put(meta[4][0], meta[4][1]);
                jsonobj.put(meta[5][0], meta[5][1]);
                try {
                    FileWriter filewriter = new
FileWriter(rootDir+"/JSONFiles/DataJSONStream/junk/temporaryjson.
json");
                    filewriter.write(jsonobj.toJSONString());
                    filewriter.flush();
                    filewriter.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                OutputStream outputStream =
connection.getOutputStream();
                //apostoli tou arxeiou json sto cloud
                File fileup = new
File(rootDir+"/JSONFiles/DataJSONStream/junk/temporaryjson.json")
;
                FileInputStream streamFileInputStream = new
FileInputStream(fileup);
                BufferedInputStream streamFileBufferedInputStream
= new BufferedInputStream(streamFileInputStream);
                byte[] streamFileBytes = new byte[(int)
file.length()];
                int bytesRead = 0;
                int totalBytesRead = 0;
                while ((bytesRead =
streamFileBufferedInputStream.read(streamFileBytes)) > 0) {
                    outputStream.write(streamFileBytes, 0,
bytesRead);
                    outputStream.flush();
                    totalBytesRead += bytesRead;
                }
                streamFileBufferedInputStream.close();
                outputStream.close();
```

```
                int responseCode = connection.getResponseCode();
                System.out.println("\nSending 'PUT' request to
URL : " + url);
                System.out.println("Response Code : " +
responseCode);
                //to result[0] exei to response code
                result[0]=""+responseCode;
                BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
                String inputLine;
                StringBuffer response = new StringBuffer();

                //getting the http response body
                while ((inputLine = in.readLine()) != null) {
                    response.append(inputLine);
                }

                //to result[1] exei tin apadisi tou server
                result[1]=connection.getHeaderField(0);

                in.close();
                connection.disconnect();
            } catch (MalformedURLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            catch (JSONException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
        }
        return result;
}
```

Όλα στον κώδικα αυτό είναι όπως και η προηγούμενη μέθοδος δημιουργίας αντικειμένων στο νέφος με μία διαφορά.
Εδώ θέλουμε να αποθηκεύουμε σαν μεταδεδομένα του αντικειμένου μόνο το id και το mac address της συσκευής. Τα υπόλοιπα δεδομένα του .json θα πρέπει να αποθηκεύονται σαν body του αντικειμένου. Έτσι αφού εξάγουμε όλα τα δεδομένα από το αρχείο με την JSONExtract κλάση, που όπως είπαμε θα αναλυθεί στο επόμενο κεφάλαιο, φτιάχνεται ένα νέο αρχείο json το οποίο περιέχει μόνο τα δεδομένα που θα μπουν σαν body του αντικειμένου.
Οπότε αφού βάλουμε σαν επικεφαλίδες του http τα δύο αυτά μεταδεδομένα κάνουμε upload στο νέφος το δεύτερο αρχείο που δημιουργήσαμε.
Το upload γίνεται μέσω ενός BufferedInputStream όπως και την άλλη μέθοδο δημιουργίας αντικειμένων.

Τέλος παίρνουμε την απάντηση του νέφους και την επιστρέφουμε στην φόρμα του χρήστη.

Η δημιουργία του επιπλέον αρχείου γίνεται για κάθε αντικείμενο που θέλουμε να αποθηκεύσουμε στο νέφος αλλά εφόσον το αρχείο αυτό δεν μας είναι απαραίτητο στην συνέχεια αποθηκεύεται το ένα πάνω στο άλλο για εξοικονόμιση χώρου.

Επιπλέον είναι απαραίτητη η δημιουργία νέου αρχείου γιατί πάλι για λόγους εξοικονόμισης χώρου δεν θέλουμε να αποθηκεύονται δύο φορές οι ίδιες πληροφορίες. Μία φορά στα μεταδεδομένα του αντικειμένου και μία στο body. Οπότε με τον τρόπο αυτό αποφεύγουμε τις διπλές επαναλήψεις ίδιων πληροφοριών.

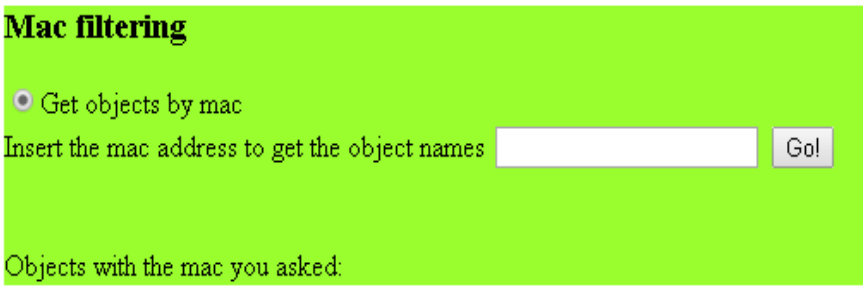
4.3.2 Απεικόνιση αντικειμένων που βρίσκονται στο νέφος αποθήκευσης

Το κομμάτι αυτό γίνεται με τον ακριβώς ίδιο τρόπο όπως και στην προηγούμενη ενότητα με την μόνη διαφορά ότι αλλάζει το container.

4.3.3 Αναζήτηση αντικειμένων με βάση το mac μεταδεδομένο

Αυτή η λειτουργία αφορά την εξειδικευμένη αναζήτηση αντικειμένων στο νέφος με βάση την διεύθυνση mac της συσκευής η οποία τα έστειλε.

Η διεπαφή του χρήστη φαίνεται παρακάτω:



Εικόνα 4.3.3 Φόρμα ανάκτησης αντικειμένου μέσω mac

Εδώ ο χρήστης πρέπει απλά να βάλει την διεύθυνση mac της οποίας τα αντικείμενα θέλει να πάρει και πατώντας το “Go!” του επιστρέφονται τα αντικείμενα που έχουν

σταλεί από την συσκευή της οποίας την mac έχει συμπληρώσει με τα δεδομένα τους αναλυτικά.

Ο κώδικας της διεπαφής αυτή παρουσιάζεται εδώ:

```
<div id="meta">
            <h3>Mac filtering</h3>
            <form id="getobjwithmac" autocomplete="off"
action="SendtoCloud" method="post">
            <input type="radio" name="action" value="getobjwithmac"
checked>Get objects by mac<br>
                  Insert the mac address to get the object names <input
type="text" name="mac" size="20">
                  <input type="submit" value="Go!" />
            </form>
            <br><br>Objects with the mac you asked: <%if
(request.getAttribute("objmac")!=null){ %>
            <% ArrayList<String> objinfo = (ArrayList<String>)
request.getAttribute("objbymac");%>
            <%int j=0; %>
            <table border=1 id="objtable">
                  <tr><th>Object Name</th><th colspan="2">Object
Meta</th><th>Object Body</th></tr>
                  <tr><% for (String s : objinfo) { %>
                        <% j=j+1;%>
                        <td><%= s %></td>
                        <%    if ((j % 4)==0){%>
                              </tr><tr>
                        <%} %>
                  <%} %>
                  </tr>
            </table><%} %>
</div>
```

Με τον κώδικα αυτό στέλνεται ένα αίτημα στην SendtoCloud με το action = getobjwithmac. Αφού σταλεί αυτή μέσω της φόρμας τότε επιστρέφεται μία λίστα με τα αντικείμενα που έχουν στα μεταδεδομένα τους την mac αυτή μαζί με όλα τα υπόλοιπα δεδομένα που περιέχουν.
Τέλος με μία επανάληψη εμφανίζουμε τα στοιχεία της λίστας και τα παρουσιάζουμε στον χρήστη.

Το κομμάτι της SendtoCloud στο οποίο στέλνεται το action είναι το εξής:

```
else if (action.equals("getobjwithmac")){
                  if (connresult[0]=="200 OK"){
                        String mac = request.getParameter("mac");
                        ArrayList<String> objbymac = new
ArrayList<String>();
                        objbymac=cc.getObjectsByMac(connresult[1],
mac);
```

```
                    request.setAttribute("objbymac", objbymac);

request.getRequestDispatcher("/datastream_actions.jsp").forward(r
equest, response);
                }else {
                    request.setAttribute("responseu", "Could not
retrieve authorization token from object cloud server");
                }
}
```

Εδώ απλά καλείται η getObjectsByMac μέθοδος με μοναδική παράμετρο το mac το οποίο έχει ορίσει ο χρήστης μέσω της προηγούμενης φόρμας.

Η μέθοδος **getObjectsByMac** της CloudConnection κλάσης φαίνεται παρακάτω:

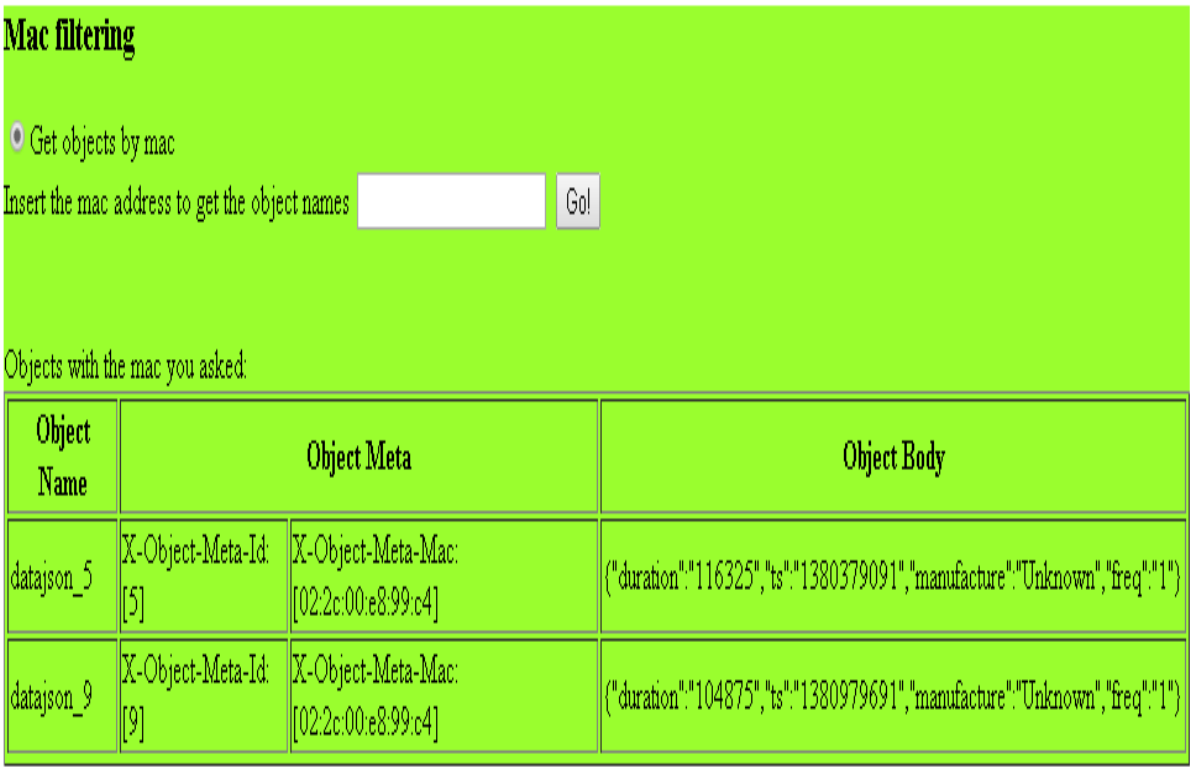
```
public ArrayList<String> getObjectsByMac (String token, String
mac){
        ArrayList<String> objectsmatchingwithmac = new
ArrayList<String>();
        ArrayList<String> objlist = new ArrayList<String>();
        String container="JsonStream";
        String mac_requested = "["+mac+"]";
        objlist=listObjects (container);//pairnw ta object names
tou container pou me endiaferei kai ua psaksw kathe ena apo auta
gia ta mac tous
        int p=2; //gia na pairnw ta swsta data ths listas efoson
thelw ta obj names
        try {
            for (int c=0;c<objlist.size()/3;c++){
                URL url = new URL
("http://83.212.121.116:8080/v1/AUTH_test/"+container+"/"
+objlist.get(p));
                HttpURLConnection connection =
(HttpURLConnection) url.openConnection();
                connection.setDoOutput(true);
                connection.setRequestMethod("GET");
                connection.setRequestProperty("X-Auth-Token",
token);
                int responseCode = connection.getResponseCode();
                System.out.println("\nSending 'GET' request to
URL : " + url);
                System.out.println("Response Code : " +
responseCode);
                Map headerfields = connection.getHeaderFields();
                Set headers = headerfields.entrySet();
                for(Iterator it = headers.iterator();
it.hasNext();){
                    Map.Entry map = (Map.Entry)it.next();
                    if (map.getKey()!=null){//upoxrewtiko giati
alliws prokuptei severe error
                        if (map.getKey().equals("X-Object-Meta-
Mac")){//pairnw ta mac apo ta meta kai elegnxw an einai idia me
```

```
to mac pou edwse o xrhsths to apothikeuw stin
objnamesthatmatchwithmac
                                String
objmac=map.getValue().toString();;
                                if (objmac.equals(mac_requested)){
                                    //edw pairnw ta metadata tou
object kai ta apothikeuw sto objmeta Array
                                    ArrayList <String> objectmeta =
listMeta(token,objlist.get(p),container);
                                    //edw pairnw to body tou object
kai to apothikeuw ston StringBuilder sb.
                                    InputStream is =
connection.getInputStream();
                                    BufferedReader br = null;
                                    StringBuilder sb = new
StringBuilder();
                                    String line;
                                    try {

                                        br = new BufferedReader(new
InputStreamReader(is));
                                        while ((line = br.readLine())
!= null) {
                                            sb.append(line);
                                        }

                                    } catch (IOException e) {
                                        e.printStackTrace();
                                    }
                                    //edw kanw to StringBuilder ->
String
                                    String objbody = sb.toString();
                                    //edw tha ta valw ola se ena
ArrayList<String> to opoio tha epistrepsw ston xrhsth

objectsmatchingwithmac.add(objlist.get(p));//edw vazw to name tou
object
                                    //for (int d=0;
d<objectmeta.size();d++){//loop gia na valw ola ta meta stin
lista

objectsmatchingwithmac.add(objectmeta.get(0));

objectsmatchingwithmac.add(objectmeta.get(8));
                                    //}

objectsmatchingwithmac.add(objbody);//edw vazw to body stin lista
                                }break;
                            }
                        }
                    }
                    p=p+3;
                    connection.disconnect();
                }
            } catch (MalformedURLException e) {
                // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return objectsmatchingwithmac;
}
```

Στην συγκεκριμένη μέθοδο αρχικά στέλνεται η μέθοδος GET του http για να πάρουμε τα αντικείμενα που βρίσκονται στο container JsonStream. Στην συνέχεια με ένα iteration μέσα σε ένα for loop παίρνουμε όλα τα μεταδεδομένα όλων των αντικειμένων που βρίσκονται στο container αυτό και ελέγχουμε ποια από αυτά είναι ίδια με το mac address που όρισε ο χρήστης.
Αν βρούμε ίδια mac σε ένα αντικείμενο τότε αρχικά αποθηκεύονται όλα τα μεταδεδομένα αυτού του αντικειμένου σε μία λίστα και στην συνέχεια με έναν BufferedReader διαβάζεται και το σώμα του αντικειμένου το οποίο αποθηκεύεται και αυτό στην λίστα.
Αυτό επαναλαμβάνεται για όλα τα αντικείμενα μέχρι που τελικά έχουμε μία λίστα με αποθηκευμένα όλα τα μεταδεδομένα και το body των αντικειμένων με τη σωστή mac.
Αυτή η λίστα επιστρέφεται πίσω στον χρήστη και παρουσιάζεται μέσω του jsp που είδαμε στην αρχή.

Το αποτέλεσμα της αναζήτησης με mac φαίνεται στην παρακάτω εικόνα:



Εικόνα 4.3.4 Εμφάνιση αντικειμένων αναζήτης μέσω mac

Το αποτέλεσμα αυτό το πήραμε βάζοντας στην φόρμα την mac: 02:2c:00:e8:99:c4
Επιστρέφονται δηλαδή το όνομα των αντικειμένων του νέφους, τα μεταδεδομένα τους (από τα οποίο μπορούμε να δούμε ότι το mac που βάλαμε είναι όντως αυτό που αναγράφεται) και το body των αντικειμένων αυτών.

4.3.4 Συνένωση αρχείων σε ένα αντικείμενο

Η τελευταία λειτουργία είναι και η πιο πολύπλοκη. Στο συγκεκριμένο κομμάτι της διπλωματικής εργασίας θα πρέπει για λόγους εξοικονόμισης χώρου στο νέφος να μπορούμε δύο αρχεία τα οποία έχουν έρθει από την ίδια συσκευή και περιέχουν μετρήσεις ανά κάποια χρονικά διαστήματα να συνενωθούν σε ένα αντικείμενο.
Η συνένωση αυτή θα γίνεται με τέτοιο τρόπο ώστε στο σώμα του αντικειμένου οι μετρήσεις να ξεκινάνε από την αρχική μέτρηση και να συνεχίζουν με την σειρά μέχρι και την τελευταία του δεύτερου αρχείου. Το κοινό αντικείμενο θα έχει τελικά σαν μεταδεδομένα εκτός από το όνομά του και το id του, την χρονική στιγμή που ξεκίνησαν οι μετρήσεις (start_time) καθώς και την χρονική στιγμή που τελείωσαν (end_time).
Υπάρχει όμως και ένας ακόμα περιορισμός που τέθηκε.
Για να μπορεί να πραγματοποιηθεί αυτή η συνένωση θα πρέπει το συνολικό μέγεθος του συνενωμένου αρχείου να μην περνάει ένα συγκεκριμένο κατώφλι. Το κατώφλι αυτό μπορεί να πάρει ότι τιμές θέλει ο εκάστοτε χρήστης του κώδικα. Το κατώγλι καθορίζεται με μία μεταβλητή μέσα στον κώδικα που λέγεται “joined_size_accepted” και μετριέται σε bytes.

Η διεπαφή του χρήστη για την λειτουργία αυτή φαίνεται εδώ:

Εικόνα 4.3.5 Φόρμα συνένωσης αρχείων σε κοινό αντικείμενο

Πάλι το μόνο που έχει να κάνει ο χρήστης είναι να πατήσει το “Join & Store” κουμπί και τότε τα δύο αρχεί που βρίσκονται στον φάκελο στον τοπικό διακομιστή θα ελεγχθούν, αναλόγως θα ενωθούν ή όχι και τέλος θα αποθηκευτούν στο νέφος σαν αντικείμενα, είτε μαζί είτε χωριστά.

Τα αρχεία βρίσκονται σε φάκελο στον τοπικο διακομιστή γιατί θεωρείται ότι έχουν στελεί εκεί από μία συσκευή (πράγμα) του διαδικτύου των πραγμάτων.

Ο κώδικας της παραπάνω διεπαφής δίνεται εδώ:

```
<div id="timestamp">
            <h3>Timestamp file join</h3>
            <form autocomplete="off" action="SendtoCloud"
method="post">
            <input type="radio" name="action" value="timejoin"
checked>Join and store timestamp json.<br>
                  <input type="submit" value="Join & Store" />
            </form>
            <br>
      Cloud server response: ${requestScope["responsetime"]}
</div>
```

Με την φόρμα αυτή στέλνεται ένα action με τιμή timejoin στην κλάση SendtoCloud.

Το τελευταίο κομμάτι της SendtoCloud που έμεινε να δειχθεί είναι αυτό που διαχειρίζεται το action = timejoin και φαίνεται παρακάτω:

```
else if (action.equals("timejoin")){
                  if (connresult[0]=="200 OK"){

result=cc.createJoinedObject(connresult[1],rootDir);
                        request.setAttribute("responsetime",
result[1]);

request.getRequestDispatcher("/datastream_actions.jsp").forward(r
equest, response);
                  }else {
                        request.setAttribute("responsetime", "Could
not retrieve authorization token from object cloud server");
                  }
}
```

Στο κομμάτι αυτό φαίνεται πως καλείται η createJoinedObject της CloudConnection κλάσης με ορίσματα το Authentication Token όπως πάντα μαζί με την τοποθεσία του root φακέλου της εφαρμογής για να μπορεί η μέθοδος να βρει τα αποθηκευμένα αρχεία τα οποίο θα κληθεί να ενώσει ή να αποθηκεύσει ξεχωριστά.

Η μέθοδος **createJoinedObject** αναλύεται παρακάτω:

```
public String[] createJoinedObject(String token, String rootDir){
        String[] result = new String[5];
        JSONExtract jsonex = new JSONExtract();
        long joined_size_accepted = 1000;// in bytes
```

```
        String joinedtimejson_name="joinedtimefile.json"; // name
of the new joined timestamp json
        boolean joined = false; // thn xrisimopoiw gia na elenksw
an telika enothikan. Alliws ta apothikeuw ksexwrista
        try {
            String[][] data_start_time =
jsonex.getJSONtimestampmeta(rootDir+"\\JSONFiles\\JsonTimestamps\
\timestamp.json");
            String[][] data_end_time =
jsonex.getJSONtimestampmeta(rootDir+"\\JSONFiles\\JsonTimestamps\
\timestamp1.json");
            if
(data_start_time[0][1].equals(data_end_time[0][1])){
                try {
                    //ta kanw join kai ftiaxnw neo arxeio mono me
ta timestamps
                    FileWriter filewriter = new
FileWriter(rootDir+"\\JSONFiles\\JoinedFile\\"+joinedtimejson_nam
e);
                    filewriter.write("{\n");
                    filewriter.write("\"Temperatures\":");

filewriter.write(data_start_time[3][1].replace("]", ",")+"\n");

filewriter.write(data_end_time[3][1].replace("[", "")+"\n");
                    filewriter.write("}");
                    filewriter.flush();
                    filewriter.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
```

Ο κώδικας χωρίστηκε για να είναι δυνατό να εξηγηθεί καλύτερα. Μέχρι εδώ έχουμε πάρει τα start_time και τα end_time των αρχείων και ελέγχεται αν το δεύτερο ξεκινάει από εκεί που τελειώνει το πρώτο και αν προέρχονται από την ίδια συσκευή.
Αν ισχύουν τα παραπάνω τα ενώνουμε σε ένα αρχείο.

```
                // elegxw an einai katw apo to kathorismeno
megethos
                File joinedfile = new
File(rootDir+"\\JSONFiles\\JoinedFile\\"+joinedtimejson_name);
                //an einai katw apo to katofli megethous to
apothikeuw me body mono ta data kai oxi ta id, start_time,
end_time
                System.out.println("Joined File Size:
"+joinedfile.length());
                if (joinedfile.length()<=joined_size_accepted){
                    joined=true;//telika ta stelnw enomena
                    URL url = new URL
("http://83.212.121.116:8080/v1/AUTH_test/IoTcontainer/"
+joinedtimejson_name);
                    HttpURLConnection connection =
(HttpURLConnection) url.openConnection();
```

```
                    connection.setDoOutput(true);
                    connection.setRequestMethod("PUT");
                    connection.setRequestProperty("Content-Type",
"multipart/form-data");
                    connection.setRequestProperty("X-Auth-Token",
token); // use the X-Auth-Token you have
                    connection.setRequestProperty("Content-
Length","200");
                    //gemizw to request me ta meta tou neou
object
                    connection.setRequestProperty("X-Object-Meta-
"+data_start_time[0][0], data_start_time[0][1]);
                    connection.setRequestProperty("X-Object-Meta-
"+data_start_time[1][0], data_start_time[1][1]);
                    connection.setRequestProperty("X-Object-Meta-
"+data_end_time[2][0], data_end_time[2][1]);
                    OutputStream outputStream =
connection.getOutputStream();
                    //apostoli tou arxeiou json sto cloud
                    File fileup = new
File(rootDir+"/JSONFiles/JoinedFile/"+joinedtimejson_name);//gia
to anevasma olou tou .json ston server pairnw to arxeio json
                    FileInputStream streamFileInputStream = new
FileInputStream(fileup);
                    BufferedInputStream
streamFileBufferedInputStream = new
BufferedInputStream(streamFileInputStream);
                    byte[] streamFileBytes = new byte[(int)
fileup.length()];
                    int bytesRead = 0;
                    int totalBytesRead = 0;
                    while ((bytesRead =
streamFileBufferedInputStream.read(streamFileBytes)) > 0) {
                        outputStream.write(streamFileBytes, 0,
bytesRead);
                        outputStream.flush();
                        totalBytesRead += bytesRead;
                    }
                    streamFileBufferedInputStream.close();
                    outputStream.close();

                    int responseCode =
connection.getResponseCode();
                    System.out.println("\nSending 'PUT' request
to URL : " + url);
                    System.out.println("Response Code : " +
responseCode);
                    //to result[0] exei to response code
                    result[0]=""+responseCode;
                    BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
                    String inputLine;
                    StringBuffer response = new StringBuffer();

                    //getting the http response body
                    while ((inputLine = in.readLine()) != null) {
```

```
                    response.append(inputLine);
                }

                //to result[1] exei tin apadisi tou server
                result[1]=connection.getHeaderField(0);

                in.close();
                connection.disconnect();
            }
        }
```

Μετά ελέγχουμε αν το μέγεθος του ενωμένου αρχείου είναι κάτω από το κατώφλι που έχει οριστεί στην αρχή του κώδικα και αν είναι αποθηκεύεται στο νέφος το ενωμένο αρχείο που περιέχει τα δεδομένα και των δύο.
Αν δεν έχουν ενωθεί λόγω του ότι το μέγεθος του ενωμένου αρχείο είναι μεγαλύτερο από το κατώφλι αποθηκεύουμε το κάθε ένα σαν ξεχωριστά αντικείμενα στο νέφος παρακάτω:

```
        if (!joined){
            for (int k=0;k<2;k++){
                try {
                    //ftiaxnw to kathe arxeio ksexwrista gia
na to anevasw sto body xwris ta metadata
                    FileWriter filewriter = new
FileWriter(rootDir+"\\JSONFiles\\JsonTimestamps\\timestamp_new_"+
k+".json");
                    filewriter.write("{\n");
                    filewriter.write("\"Temperatures\":");
                    if (k==0){

filewriter.write(data_start_time[3][1]+"\n");
                    }else if (k==1){

filewriter.write(data_end_time[3][1]+"\n");
                    }
                    filewriter.write("}");
                    filewriter.flush();
                    filewriter.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                File fileup;
                //kai ftiaxnw ta objects ston server
                URL url = new URL
("http://83.212.121.116:8080/v1/AUTH_test/IoTcontainer/timestamp"
+k);
                HttpURLConnection connection =
(HttpURLConnection) url.openConnection();
                connection.setDoOutput(true);
                connection.setRequestMethod("PUT");
                connection.setRequestProperty("Content-Type",
"multipart/form-data");
```

```
                    connection.setRequestProperty("X-Auth-Token",
token); // use the X-Auth-Token you have
                    connection.setRequestProperty("Content-
Length","200");
                    //gemizw to request me ta meta tou neou
object
                    if (k==0){
                        connection.setRequestProperty("X-Object-
Meta-"+data_start_time[0][0], data_start_time[0][1]);
                        connection.setRequestProperty("X-Object-
Meta-"+data_start_time[1][0], data_start_time[1][1]);
                        connection.setRequestProperty("X-Object-
Meta-"+data_start_time[2][0], data_start_time[2][1]);
                    }else{
                        connection.setRequestProperty("X-Object-
Meta-"+data_end_time[0][0], data_end_time[0][1]);
                        connection.setRequestProperty("X-Object-
Meta-"+data_end_time[1][0], data_end_time[1][1]);
                        connection.setRequestProperty("X-Object-
Meta-"+data_end_time[2][0], data_end_time[2][1]);
                    }
                    OutputStream outputStream =
connection.getOutputStream();
                    fileup = new
File(rootDir+"/JSONFiles/JsonTimestamps/timestamp_new_"+k+".json"
);
                    FileInputStream streamFileInputStream = new
FileInputStream(fileup);
                    BufferedInputStream
streamFileBufferedInputStream = new
BufferedInputStream(streamFileInputStream);
                    byte[] streamFileBytes = new byte[(int)
fileup.length()];
                    int bytesRead = 0;
                    int totalBytesRead = 0;
                    while ((bytesRead =
streamFileBufferedInputStream.read(streamFileBytes)) > 0) {
                        outputStream.write(streamFileBytes, 0,
bytesRead);
                        outputStream.flush();
                        totalBytesRead += bytesRead;
                    }
                    streamFileBufferedInputStream.close();
                    outputStream.close();

                    int responseCode =
connection.getResponseCode();
                    System.out.println("\nSending 'PUT' request
to URL : " + url);
                    System.out.println("Response Code : " +
responseCode);
                    //to result[0] exei to response code
                    result[0]=""+responseCode;
                    BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
                    String inputLine;
```

```
                    StringBuffer response = new StringBuffer();

                    //getting the http response body
                    while ((inputLine = in.readLine()) != null) {
                        response.append(inputLine);
                    }

                    //to result[1] exei tin apadisi tou server
                    result[1]=connection.getHeaderField(0);

                    in.close();
                    connection.disconnect();
                }
            }
        } catch (MalformedURLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        catch (JSONException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        return result;
}
```

Ο κώδικας είναι μακροσκελής στο τελευταίο κομμάτι γιατί πρέπει να αποθηκευτούν δύο αντικείμενα στο νέφος με διαφορετικά μεταδεδομένα το καθένα. Η αποθήκευση γίνεται όπως έχει εξηγηθεί και στις προηγούμενες διαδικασίες δημιουργίας αντικειμένων στο νέφος.

Στην ενότητα που ακολουθεί θα δειχθούν τα json αρχεία που χρησιμοποιούνται στις λειτουργίες αυτές καθώς και το τρόπος που αναλύουμε και αντλούμε τα δεδομένα από αυτά.

4.4. Ανάλυση των αρχείων JSON

Στην ενότητα αυτή που είναι και η τελευταία της υλοποίησης θα εξηγηθεί ο τρόπος ανάλυσης και ανάκτησης των δεδομένων από τα αρχεία JSON για να αποθηκευτούν στην συνέχεια σαν αντικείμενα στο νέφος με τους τρόπους που εξηγήθηκαν στις παραπάνω ενότητες.
Ο λόγος για τον οποίο αναλύονται αυτά στην τελευταία ενότητα είναι γιατί τα αρχεία JSON μπορούν να έχουν όποια μορφή χρειαστεί και δεν είναι απαραίτητο να έχουν την

μορφή αυτών που χρησιμοποιήθηκαν. Αυτό γίνεται για να είναι ο κώδικας και γενικά όλη η εφαρμογή προσαρμόσιμη σε οποιαδήποτε ανάγκη. Για να γίνει η προσαρμογή αυτή το μόνο που θα πρέπει να αλλάξει είναι οι μέθοδοι που θα δειχθούν σε αυτήν την ενότητα μαζί με το εκάστοτε JSON αρχείο που επεξεργάζεται.
Έτσι η λειτουργικότητα της εφαρμογής που αναλύθηκε στις παραπάνω ενότητες μένει αναλύωτη και προσαρμόσιμη.

4.4.1. Ανάλυση JSON ενότητας 4.2.1

Στην ενότητα 4.2 είδαμε πως ανεβάζεται και δημιουργείται ένα αντικείμενο από ένα αρχείο JSON το οποίο το ονομάσαμε Virtual_Entity.

Η μορφή του αρχείου αυτού είναι η εξής:

```
{
  "name" : " Virtual Entity",
  "type" : "Physical Object(device) or network",
  "description" : "p.x. camera",
  "location" : "string",
     "computational properties" :
     {
        "cpu" : "number",
        "memory" : "number"
     },
     "social properties" :
     {
        "intentions" : "string",
        "relationships" : "string",
        "connections" : "string",
        "objectives" : "string",
        "plans" : "string",
        "goals" : "string",
        "administration" : "string",
        "experiences" : [
           {
             "type": "functional",
             "resource": "string"
           },
           {
             "type": "functional",
             "optimization": "string"
           },
           {
             "type": "functional",
             "healing": "string"
           },
```

```
            {
              "type": "functional",
              "protection": "string"
            },
            {
              "type": "non - functional",
              "trust/reputation": "number"
            }
            ]
}
```

Δηλαδή έχουμε ένα αντικείμενο JSONObject που περιέχει τις αρχικές πληροφορίες (π.χ. name, type) και μετά ένα εμφωλευμένο JSONObject με όνομα computational_properties με τις δικές του πληροφορίες. Στην συνέχεια έχει ένα εμφωλευμένο στο αρχικό JSONObject με όνομα social_properties. Τέλος μέσα σε αυτό υπάρχει ένας πίνακας JSONArray με όνομα experiencies και τις δικές του πληροφορίες.
Το αρχείο αυτό δηλαδή περιγράφει πλήρως ένα πράγμα (ή συσκευή) του διαδικτύου των πραγμάτων και όλες τις πληροφορίες του για την λειτουργία του και την κατάστασή του.
Όπως φαίνεται τα δεδομένα σε κάθε JSON αρχείο είναι της μορφής “key : value” pairs.
Για την ανάλυση αυτού του JSON αρχείου χρησιμοποιήθηκε η μέθοδος ParseJSON της κλάσης JSONExtract που όπως έχει προαναφερθεί περιέχει όλες τις μεθόδους χειρισμού και ανάλυσης των αρχείων JSON.

Ο κώδικας της **ParseJSON** είναι ο εξής:

```
public String[][] ParseJSON(String pathtofile) throws
JSONException{
        JSONObject jsonobj;
        String[][] meta = new String[18][18];
        try {
            //get the json file and parse it to json object
            jsonobj = (JSONObject) parser.parse(new
FileReader(pathtofile));

            //get the strings keys and the values
            String name = (String) jsonobj.get("name");
            meta[0][0]="name";
            meta[0][1]=name;

            String type = (String) jsonobj.get("type");
            meta[1][0]="type";
            meta[1][1]=type;

            String description = (String)
jsonobj.get("description");
            meta[2][0]="description";
            meta[2][1]=description;

            String location = (String) jsonobj.get("location");
```

```
            meta[3][0]="location";
            meta[3][1]=location;

            JSONObject cp = (JSONObject)
jsonobj.get("computational properties");
            String cpu = (String) cp.get("cpu");
            meta[4][0]="computational_properties-cpu";
            meta[4][1]=cpu;

            String memory = (String) cp.get("memory");
            meta[5][0]="computational_properties-memory";
            meta[5][1]=memory;

            JSONObject sp = (JSONObject) jsonobj.get("social
properties");
            String intentions = (String) sp.get("intentions");
            meta[6][0]="social_properties-intentions";
            meta[6][1]=intentions;

            String relationships = (String)
sp.get("relationships");
            meta[7][0]="social_properties-relationships";
            meta[7][1]=relationships;

            String connections = (String) sp.get("connections");
            meta[8][0]="social_properties-connections";
            meta[8][1]=connections;

            String objectives = (String) sp.get("objectives");
            meta[9][0]="social_properties-objectives";
            meta[9][1]=objectives;

            String plans = (String) sp.get("plans");
            meta[10][0]="social_properties-plans";
            meta[10][1]=plans;

            String goals = (String) sp.get("goals");
            meta[11][0]="social_properties-goals";
            meta[11][1]=goals;

            String administration = (String)
sp.get("administration");
            meta[12][0]="social_properties-administration";
            meta[12][1]=administration;

            JSONArray names = (JSONArray)
sp.get("experiences");//get the experience array with the objects
            JSONObject experience1 =
(JSONObject)names.get(0);//get the first object of the experience
table
            String exptype1 = (String) experience1.get("type");
            String expresource = (String)
experience1.get("resource");
            if (exptype1=="functional"){
                meta[13][0]="experience-functional-resource";
                meta[13][1]=expresource;
```

```
            }else {
                meta[13][0]="experience-nonfunctional-resource";
                meta[13][1]=expresource;
            }
            JSONObject experience2 =
(JSONObject)names.get(1);//get the second object of the
experience table
            String exptype2 = (String) experience2.get("type");
            String expoptimization = (String)
experience2.get("optimization");
            if (exptype2=="functional"){
                meta[14][0]="experience-functional-optimization";
                meta[14][1]=expoptimization;
            }else {
                meta[14][0]="experience-nonfunctional-
optimization";
                meta[14][1]=expoptimization;
            }
            JSONObject experience3 =
(JSONObject)names.get(2);//get the third object of the experience
table
            String exptype3 = (String) experience3.get("type");
            String exphealing = (String)
experience3.get("healing");
            if (exptype3=="functional"){
                meta[15][0]="experience-functional-healing";
                meta[15][1]=exphealing;
            }else {
                meta[15][0]="experience-nonfunctional-healing";
                meta[15][1]=exphealing;
            }
            JSONObject experience4 =
(JSONObject)names.get(3);//get the fourth object of the
experience table
            String exptype4 = (String) experience4.get("type");
            String expprotection = (String)
experience4.get("protection");
            if (exptype4=="functional"){
                meta[16][0]="experience-functional-protection";
                meta[16][1]=expprotection;
            }else {
                meta[16][0]="experience-nonfunctional-
protection";
                meta[16][1]=expprotection;
            }
            JSONObject experience5 =
(JSONObject)names.get(4);//get the fifth object of the experience
table
            String exptype5 = (String) experience5.get("type");
            String exptrustreputation = (String)
experience5.get("trust/reputation");
            if (exptype5=="functional"){
                meta[17][0]="experience-functional-
trust_reputation";
                meta[17][1]=exptrustreputation;
            }else {
```

```
                meta[17][0]="experience-nonfunctional-
trust_reputation";
                meta[17][1]=exptrustreputation;
            }

        } catch (IOException | ParseException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return meta;
}
```

Ο παραπάνω κώδικας παίρνει σαν είσοδο την τοποθεσία του αρχείου και επιστρέφει ένα πίνακα δισδυάστατο με τα δεδομένα του παραπάνω JSON σε μορφή key : value όπως ειπώθηκε πριν. Τα keys αποθηκεύονται στην $1^{η}$ στήλη του πίνακα ενώ τα values στην $2^{η}$. Ο τρόπος εξαγωγής των δεδομένων από το αρχείο είναι ότι ορίζουμε JSONObjects ή JSONArrays ανάλογα τι έχουμε κάθε φορά και καλώντας την .get() μέθοδο με όρισμα το key του οποίου το value θέλουμε παίρνουμε το key : value pair και το αποθηκεύουμε στον πίνακα όπως εξηγήθηκε.

4.4.2. Ανάλυση JSON ενότητας 4.3.1

Στην ενότητα 4.3.1 αποθηκεύαμε ένα σύνολο πολλών αρχείων JSON που όμως έχουν όλα την ίδια μορφή.
Τα αρχεία αυτά έχουν όνομα datax όπου x=1,2,3,... μέχρι όσα αρχεία χρειάζεται να γίνουν αντικείμενα.

Η μορφή των αρχείων data φαίνεται παρακάτω παίρνοντας το data1.json:

```
{
    "id":1,
    "mac":"02:3c:00:c0:63:94",
    "freq":1,
    "ts":1380378452,
    "duration":116336,
    "manufacture":"Unknown"
}
```

Εδώ έχουμε ένα μόνο JSONObject. Αυτό που μας ενδιέφερε εδώ είναι να αντλείσουμε όλα τα δεδομένα αλλά μόνο το id και το mac θα μπουν σαν μεταδεδομένα γιατί όπως είχε εξηγηθεί στην ενότητα 4.3.1 τα υπόλοιπα δεδομένα θα τα αποθηκεύαμε στο body του αντικειμένου στο νέφος.

Τα αντλούμε όλα όμως για να μπορεί να δημιουργηθεί νέο αρχείο μόνο με τα δεδομένα του body έτσι ώστε να ανέβει αυτό στο νέφος. Ο τρόπος εξηγείται στην 4.3.1
Άρα με την μέθοδο **ParseJSONStream** φτιάχνουμε ένα JSONObject και από αυτό με την get μέθοδο παίρνουμε τα key : value pairs τα οποία και αποθηκεύονται στον πίνακα όπως πριν.

```
public String[][] ParseJSONstream(String pathtofile) throws
JSONException{
        JSONObject jsonobj;
        String[][] meta = new String[6][6];
        try {
            //get the json file and parse it to json array full
of json objects
            jsonobj = (JSONObject) parser.parse(new
FileReader(pathtofile));
            //pairnw mono auta pou thelw na apothikeusw san meta
            String id = (String)
String.valueOf(jsonobj.get("id"));
            meta[0][0]="id";
            meta[0][1]=id;
            String mac = (String) jsonobj.get("mac");
            meta[1][0]="mac";
            meta[1][1]=mac;
            String freq = (String)
String.valueOf(jsonobj.get("freq"));
            meta[2][0]="freq";
            meta[2][1]=freq;
            String ts = (String)
String.valueOf(jsonobj.get("ts"));
            meta[3][0]="ts";
            meta[3][1]=ts;
            String duration = (String)
String.valueOf(jsonobj.get("duration"));
            meta[4][0]="duration";
            meta[4][1]=duration;
            String manufacture = (String)
jsonobj.get("manufacture");
            meta[5][0]="manufacture";
            meta[5][1]=manufacture;
        } catch (IOException | ParseException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return meta;
}
```

Ο πίνακας τελικά επιστρέφεται πίσω στο πρόγραμμα που την κάλεσε.

4.4.3. Ανάλυση JSON ενότητας 4.3.4

Στην ενότητα 4.3.4 εξηγήθηκε ο τρόπος με τον οποίο ελέγχουμε δύο αρχεία και αν προέρχονται από την ίδια συσκευή και στο ένα συνεχίζουν οι μετρήσεις του από εκεί που τελειώνουν του πρώτου θα πρέπει να ενωθούν σε ένα.

Στην επόμενη σελίδα έχουμε τα δύο αυτά αρχεία JSON με μετρήσεις θερμοκρασιών τα οποία ονομάζονται timestamp.json και timestamp1.json.

- timestamp.json:


```
{
    "Id": "2",
    "Start-Time": "2014-04-10T00:00:00",
    "End-Time": "2014-04-10T24:00:00",
    "Temperatures": [
        { "time": "2014-04-10T00:00:00", "temperature": "10 C" },
        { "time": "2014-04-10T02:00:00", "temperature": "09 C" },
        { "time": "2014-04-10T04:00:00", "temperature": "08 C" },
        { "time": "2014-04-10T06:00:00", "temperature": "10 C" },
        { "time": "2014-04-10T08:00:00", "temperature": "11 C" },
        { "time": "2014-04-10T10:00:00", "temperature": "12 C" },
        { "time": "2014-04-10T12:00:00", "temperature": "13 C" },
        { "time": "2014-04-10T14:00:00", "temperature": "14 C" },
        { "time": "2014-04-10T16:00:00", "temperature": "13 C" },
        { "time": "2014-04-10T18:00:00", "temperature": "12 C" },
        { "time": "2014-04-10T20:00:00", "temperature": "11 C" },
        { "time": "2014-04-10T22:00:00", "temperature": "10 C" },
        { "time": "2014-04-10T24:00:00", "temperature": "10 C" }
    ]
}
```


- timestamp1.json:


```
{
    "Id": "2",
    "Start-Time": "2014-05-10T00:00:00",
    "End-Time": "2014-05-10T24:00:00",
    "Temperatures": [
        { "time": "2014-05-10T00:00:00", "temperature": "11 C" },
        { "time": "2014-05-10T02:00:00", "temperature": "09 C" },
        { "time": "2014-05-10T04:00:00", "temperature": "09 C" },
        { "time": "2014-05-10T06:00:00", "temperature": "13 C" },
        { "time": "2014-05-10T08:00:00", "temperature": "14 C" },
```

```
        { "time": "2014-05-10T10:00:00", "temperature": "15 C" },
        { "time": "2014-05-10T12:00:00", "temperature": "15 C" },
        { "time": "2014-05-10T14:00:00", "temperature": "14 C" },
        { "time": "2014-05-10T16:00:00", "temperature": "14 C" },
        { "time": "2014-05-10T18:00:00", "temperature": "12 C" },
        { "time": "2014-05-10T20:00:00", "temperature": "13 C" },
        { "time": "2014-05-10T22:00:00", "temperature": "12 C" },
        { "time": "2014-05-10T24:00:00", "temperature": "11 C" }
    ]
}
```

Όπως φαίνεται έχουν το ίδιο id άρα προέρχονται από την ίδια συσκευή και όντως το End-Time του πρώτου είναι ίδιο με το Start-Time του δεύτερου.
Και τα δύο αποτελούνται από ένα αρχικό JSONObject μέσα στο οποίο υπάρχουν πληροφορίες της μορφής key : value και ένας εμφωλευμένος πίνακας JSONArray με όνομα temperatures. Ο πίνακας αυτός περιέχει τις μετρήσεις θερμοκρασιών συγκεκριμένων χρονικών στιγμών.
Όπως έχει εξηγηθεί και στην ενότητα 4.3.4 σαν μεταδεδομένα στο αντικείμενο στο νέφος θα μπει το Start-Time του πρώτου αρχείου, το End-Time του δεύτερου και το κοινό τους Id εάν αυτά ενωθούν.
Εάν όχι τότε θα αποθηκευτούν σαν ξεχωριστά αντικείμενα με τα δικά του μεταδεδομένα το καθένα με τον τρόπο που εξηγήθηκε στην ενότητα 4.3.4.

Σε κάθε περίπτωση όμως θα πρέπει να εξάγουμε όλα τα μεταδεδομένα (Id, Start-Time, End-Time) του κάθε JSON αρχείου και αυτό γίνεται με την getJSONTimestampmeta μέθοδο της JSONExtract κλάσης.

Η μέθοδος **getJSONTimestampmeta** είναι η εξής:

```
public String[][] getJSONtimestampmeta(String pathtofile) throws
JSONException{
        JSONObject jsonobj;
        String[][] meta = new String[4][2];
        try {
            //get the json file and parse it to json array full
of json objects
            jsonobj = (JSONObject) parser.parse(new
FileReader(pathtofile));
            //pairnw mono auta pou thelw na apothikeusw san meta
            String id = (String)
String.valueOf(jsonobj.get("Id"));
            meta[0][0]="id";
            meta[0][1]=id;
            String start_time = (String) jsonobj.get("Start-
Time");
            meta[1][0]="starttime";
            meta[1][1]=start_time;
            String end_time = (String)
String.valueOf(jsonobj.get("End-Time"));
```

```
            meta[2][0]="endtime";
            meta[2][1]=end_time;
            //sto meta [3][1] vrisketai to array me ta
temperatures
            JSONArray temperatures = (JSONArray)
jsonobj.get("Temperatures");
            meta[3][0]="temperatures";
            meta[3][1]=temperatures.toJSONString();
        } catch (IOException | ParseException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return meta;
}
```

Στην μέθοδο αυτή χρησιμοποιώντας ακριβώς τον ίδιο τρόπο και με τις προηγούμενες εξάγονται τα δεδομένα και αποθηκεύονται σε ένα δισδυάστατο πίνακα ο οποίος με την σειρά του επιστρέφεται πίσω στην μέθοδο που την κάλεσε.

Έτσι τελειώνει το κομμάτι της υλοποίησης της εφαρμογής της διαχείρισης δεδομένων.

4.5. Μορφοποίηση διεπαφής χρήστη

Η μορφοποίηση της διεπαφής του χρήστη η οποία δείχθηκε στις παραπάνω εικόνες είναι αποτέλεσμα ενός αρχείου css. Τα αρχεία css είναι αρχεία κώδικα για να μορφοποιούν το περιεχόμενα ενός html αρχείου το οποίο σερβίρεται στην χρήστη μέσω του φυλλομετρητή ιστού.
Το CSS είχε σχεδιαστεί αρχικά για να επιτρέψει τον διαχωρισμό του περιεχομένου ενός εγγράφου από την παρουσίασή του, δηλαδή στοιχεία όπως τα χρώματα και οι γραμματοσειρές. Ο διαχωρισμός αυτός βελτιώνει την πρόσβαση περιεχομένου, παρέχει περισσότερη ευελιξία και έλεγχο στα χαρακτηριστικά παρουσίασης, επιτρέπει πολλαπλές σελίδες να μοιράζονται την ίδια παρουσίαση και μειώνει την πολυπλοκότητα και την επανάληψη στα δομημένα δεδομένα. [16]

Το αρχείο CSS που χρησιμοποιήθηκε στην παρούσα εφαρμογή είναι σχετικά απλό και έχει ως βασικό σκοπό τον διαχωρισμό των διαφόρων λειτουργιών της διεπαφής του χρήστη για να είναι ευδιάκριτο.

Ο κώδικας είναι ο εξής:

```
@CHARSET "UTF-8";
#upload{
	background-color:#F4FA58;
}
#create{
	background-color:#F4FA58;
}
#delete{
	background-color:#FF3300;
}
#upresult{
	background-color:#F4FA58;
}
#objaction{
	background-color:#F4FA58;
}
#list{
	background-color:#58FAF4;
}
#main{
	width:960px;
	margin-left:auto;
	margin-right:auto;
}
#objtable{
	margin-left:auto;
	margin-right:auto;
}
#meta{
	background-color:#9AFE2E;
}
#timestamp{
	background-color:#FF00BF;
}
#details{
	background-color:#B8B894;
}
```

Αρχικά ρυθμίζεται το σύνολο χαρακτήρων σε UTF-8 για να δέχεται τα ελληνικά χωρίς πρόβλημα και στην συνέχει έχει ένα σύνολο κανόνων κυρίως χρωματικών και τοποθέτησης περιεχομένου.

Κεφάλαιο 5: Συμπεράσματα – Μελλοντικές επεκτάσεις

Αυτό που ίσως κάνει τις δικτυακές υπηρεσίες μοναδικές είναι το γεγονός ότι ο κόσμος της πληροφορίας ελπίζει ότι μέσω αυτών θα καταφέρουν οι επιχειρήσεις και οι δημόσιες υπηρεσίες να οργανώσουν τις προσφερόμενες υπηρεσίες τους με τέτοιο τρόπο ώστε η αναζήτηση, η εύρεση και η χρήση αυτών από τους ενδιαφερόμενους τους να γίνεται εύκολα και γρήγορα.

Έτσι το διαδίκτυο των πραγμάτων περνάει ένα βήμα μπροστά προσπαθώντας μέσω της άντλησης πληροφοριών από οποιαδήποτε συσκευή μπορεί να μετρήσει ή να εξάγει δεδομένα να αποθηκεύει τα δεδομένα αυτά. Στην συνέχεια τα δεδομένα αυτά θα χρησιμοποιούνται για να ρυθμίσουν την συμπεριφορά άλλων συσκευών εγκαταστάσεων κ.α. Η αναζήτηση αυτών των δεδομένων είναι πολύ εύκολη και γρήγορη μέσω του υπολογιστικού νέφους στο οποίο αποθηκεύονται σαν αντικείμενα.

Με τον τρόπο αυτό θα μπορουν να αυτοματοποιηθούν πολλές λειτουργίες και συμπεριφορές με βάση τις καταστάσεις και τις εμπειρίες άλλων συσκευών μέσα στο σύστημα. Αυτή η αυτοματοποίηση θα επιφέρει ταχύτητα στις διάφορες λειτουργίες που θα πρέπει να πραγματοποιηθούν.

Όπως έχει αναφερθεί και στην παρούσα διπλωματική με το διαδίκτυο των πραγμάτων θα μπορούν να δημιουργηθούν έξυπνα σπίτια, εργοστάσια ακόμα και πόλεις στις οποίες πολλές απο τις λειτουργίες που θα έπρεπε να επιτελέσει ο άνθρωπος θα γίνονται αυτόματα.

Κεφάλαιο 6: Βιβλιογραφία και αναφορές

[1]Cloud computing: http://blogs.msdn.com/b/gkanel/archive/2010/10/29/cloud-computing.aspx
[2] Cloud computing: http://tech.in.gr/short-news/?aid=1231078190
[3] REST: http://www.lynda.com/OData-tutorials/What-REST/126131/145957-4.html
[4] Openstack: https://www.openstack.org/software/
[5] Openstack install Swift: https://wiki.openstack.org/wiki/Swift
[6] Openstack Swift: http://docs.openstack.org/developer/swift/development_saio.html
[7] Advanced packaging tool: http://sernaonubuntu.wikidot.com/advanced-packaging-tool
[8] Object storage cloud: http://docs.openstack.org/api/openstack-object-storage/1.0/content/storage-account-services.html
[9] Thorsten Kramp, Rob van Kranenburg, and Sebastian Lange "Enabling Things to Talk Designing IoT solutions with the IoT Architectural Reference Model" Springer, 2013
[10] On the LinkedIn Group "Internet of Things" strueker@iig.uni-freiburg.de
[11] Smart cities: http://www.smart-cities.eu/download/smart_cities_final_report.pdf
[12] M. M. Hassan, B. Song, and E. Huh, "A framework of sensor-cloud integration opportunities
and challenges", in Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication, ICUIMC 2009, Suwon, Korea, January 15–16, pp. 618–626, 2009
[13] M. Yuriyama and T. Kushida, "Sensor-Cloud Infrastructure — Physical Sensor Management
with Virtualized Sensors on Cloud Computing", NBiS 2010: 1–8
[14] C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee, "Linked Data on theWeb", Proceedings
of the 17th International Conference on World Wide Web (WWW'08), New York, NY, USA, ACM, pp. 1265–1266, 2008
[15] JSON: http://json.org/
[16] CSS: http://www.w3.org/Style/CSS/