Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής

Μελέτη και ανάπτυξη διαδικτυακής υπηρεσίας για την επίβλεψη νέφους αποθήκευσης δεδομένων (storage cloud)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτρης Τριχινάς

Επιβλέπων: Θεοδώρα Βαρβαρίγου

Καθηγήτρια Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2012

Η σελίδα αυτή είναι σκόπιμα λευκή.

Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής

Μελέτη και ανάπτυξη διαδικτυακής υπηρεσίας για την επίβλεψη νέφους αποθήκευσης δεδομένων (storage cloud)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτρης Τριχινάς

Επιβλέπων: Θεοδώρα Βαρβαρίγου

Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 18η Σεπτεμβρίου 2012

……..	…….	……..
Θεοδώρα Βαρβαρίγου	Συμεών Παπαβασιλείου	Δημήτριος Ασκούνης

Αθήνα, Σεπτέμβριος 2012

Η σελίδα αυτή είναι σκόπιμα λευκή.

.....

Δημήτρης Τριχινάς

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

Η σελίδα αυτή είναι σκόπιμα λευκή.

Περίληψη

Σκοπός αυτής της διπλωματικής εργασίας υπήρξε η διερεύνηση και η μελέτη των κατάλληλων τεχνολογιών με στόχο τη ανάπτυξη και σχεδιασμό διαδικτυακής υπηρεσίας για τη επίβλεψη νέφους αποθήκευσης δεδομένων (storage cloud).

Για τη αποκόμιση των κατάλληλων γνώσεων αρχικά μελετήθηκε το τι είναι cloud computing, ο διαχωρισμός του σε κατηγορίες, ποια τα πλεονεκτήματα και οι ανησυχίες που έχουν οργανισμοί για τη μεταφορά των υπηρεσιών τους σε νέφος. Ειδικό βάρος δόθηκε στη μελέτη των χαρακτηριστικών και την αρχιτεκτονική των νεφών αποθήκευσης δεδομένων (storage clouds). Ακολούθως μελετήθηκαν τα χαρακτηριστικά του αρχιτεκτονικού μοντέλου REST και τα πλεονεκτήματα του έναντι άλλων μοντέλων όπως το SOAP. Έπειτα, μελετήθηκε η γλώσσα σήμανσης HTML5 και δόθηκε ειδικό βάρος στο EventSource API που ειδικεύεται στη λήψη Server-Side Events (SSEs) στη πλευρά του πελάτη. Έπειτα, μελετήθηκε το πρωτόκολλο CDMI (Cloud Data Management Interface) σε αποθηκευτικά νέφη. Τέλος, αναλύθηκαν οι προκλήσεις που συναντάμε στην επίβλεψη νεφών και ειδικά στην επίβλεψη νεφών από τη μεριά του πελάτη (client-side cloud monitoring).

Η διαδικτυακή υπηρεσία που αναπτύχθηκε παρέχει τη δυνατότητα σε χρήστες αποθηκευτικού νέφους, οι οποίοι εγγράφονται στη υπηρεσία μέσω διαδικτύου, να λαμβάνουν πληροφορίες για γεγονότα (events) που συμβαίνουν στο νέφος. Η διαδικτυακή υπηρεσία τρέχει στον περιηγητή του χρήστη χωρίς τη μεσολάβηση αλλού λογισμικού εγκατεστημένο τοπικά στον υπολογιστή. Με τον τρόπο αυτό μεταφέρουμε την επίβλεψη του αποθηκευτικού νέφους στη πλευρά του πελάτη. Η διαδικτυακή υπηρεσία ακολουθεί το αρχιτεκτονικό μοντέλο REST για την επικοινωνία της με το νέφος, υλοποιεί το πρωτόκολλο CDMI και ειδικότερα τις ουρές ειδοποίησης (Notification Queues) ενώ η ενημέρωση του χρήστη γίνεται με τη χρήση HTML5 EventSource.

Αρχικά ο χρήστης δίνει τα στοιχεία του για επαλήθευση από το σύστημα, ακολούθως δημιουργεί μια ουρά γεγονότων και μεταβαίνει στη σελίδα παρακολούθησης γεγονότων όπου αρχίζει η ροή ενημερώσεων από το νέφος προς τον περιηγητή του. Στη σελίδα παρακολούθησης γεγονότων ο χρήστης έχει στη διάθεση του διάφορα εργαλεία όπως κονσόλα άφιξης νέου event και γραφικές παραστάσεις για γραφική πλοήγηση ανάμεσα στα αντικείμενα που φθάνουν μέσω των event από το νέφος.

Λέξεις Κλειδιά

υπολογιστικό νέφος, νέφος αποθήκευσης δεδομένων, σύστημα επίβλεψης, διαδικτυακή υπηρεσία, REST, EventSource, server-side events, HTML5, CDMI, AJAX, ουρές ειδοποίησης

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

The objective of this thesis was to research and study the most suitable technologies available to develop and design a web service for monitoring a storage cloud system.

To obtain the proper knowledge initially studied was cloud computing, its separation in different categories, what are the benefits and concerns that organizations face when thinking of transferring their services to the cloud. Special emphasis was given to studying the characteristics and architecture of storage clouds. Next studied were the characteristics of the REST architectural style and its advantages over other models such as SOAP. Afterwards studied was the HTML5 markup language with emphasis given to the EventSource API which specializes in receiving Server-Side Events (SSEs) on the client side. Next, studied was the CDMI (Cloud Data Management Interface) protocol for storage clouds. Finally, we analyzed the challenges encountered in cloud monitoring and especially in client-side cloud monitoring.

The web service developed gives the opportunity to storage cloud users, who subscribe to the service via the internet, to receive information about events that occur in the cloud. The web service runs on the user's browser without the need of other software installed locally on his desktop. By doing this, we transfer the monitoring of the storage cloud to the client side, discharging the cloud from the work load. The web service uses the REST architectural style for the communication with the cloud, implements the CDMI protocol and especially notification queues while informing the user of new events is done using HTML5 EventSource.

Initially, the user provides the system with his credentials for verification and then creates an event queue. With the creation of a queue, a stream of new events can now start from the cloud to the user's browser. At the Event Monitoring Page the user has a variety of tools to use such as Event Arrival Console and Graphs for graphical navigation between objects processed that arrived from the cloud.

Keywords

cloud computing, storage cloud, monitoring system, web service, REST, Event Source, server-side events, HTML5, CDMI, AJAX, notification queues

Η σελίδα αυτή είναι σκόπιμα λευκή.

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 Δρ. Σπυρίδων Β. Γωγουβίτη και τον Βασίλη Αλεξάνδρου για τις συμβουλές που μου έδωσαν, την καθοδήγηση στη 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.

Η σελίδα αυτή είναι σκόπιμα λευκή.

Περιεχόμενα

Μέρος Α

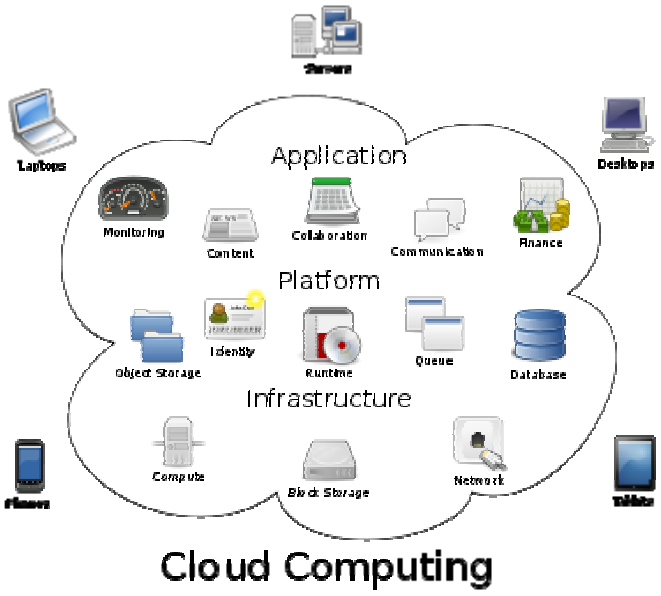
Στο πρώτο μέρος αυτής της εργασίας παρουσιάζονται οι τεχνολογίες που μελετήθηκαν με στόχο τη αποκόμιση των γνώσεων απαραίτητων ώστε να αναπτυχθεί μια διαδικτυακή υπηρεσία με σκοπό τη επίβλεψη ενός νέφους αποθήκευσης δεδομένων.

Στο μέρος αυτό αρχικά παρουσιάζεται το τι είναι cloud computing, ο διαχωρισμός του σε κατηγορίες, ποια τα πλεονεκτήματα και οι ανησυχίες που έχουν οργανισμοί για τη μεταφορά των υπηρεσιών τους σε νέφος. Ειδικό βάρος δίνεται στη μελέτη των χαρακτηριστικών και αρχιτεκτονική των νεφών αποθήκευσης δεδομένων (storage clouds). Ακολούθως παρουσιάζονται τα χαρακτηριστικά του αρχιτεκτονικού μοντέλου REST και τα πλεονεκτήματα του έναντι άλλων μοντέλων όπως το SOAP. Έπειτα, παρουσιάζεται η γλώσσα σήμανσης HTML5 και δίνεται ειδικό βάρος στο EventSource API που ειδικεύεται στη λήψη Server-Side Events (SSEs) στη πλευρά του πελάτη. Παρουσιάζονται ακόμη οι διαφορές των SSEs σε σχέση με long polling και web sockets. Έπειτα, παρουσιάζεται το πρωτόκολλο CDMI (Cloud Data Management Interface) σε αποθηκευτικά νέφη και οι αρχές του Vision Cloud. Τέλος, αναλύθηκαν οι προκλήσεις που συναντάμε στην επίβλεψη νεφών και ειδικά στην επίβλεψη νεφών από τη μεριά του πελάτη (client-side cloud monitoring).

Cloud Computing

Ο όρος cloud computing αναφέρεται σε εφαρμογές και υπηρεσίες οι οποίες τρέχουν σε κατανεμημένο δίκτυο με πρόσβαση μέσω κοινών πρωτοκόλλων του διαδικτύου και πρότυπα δικτύωσης. Το σύννεφο διακρίνεται από την ιδέα ότι οι πόροι είναι εικονικοί και απεριόριστοι και ότι οι λεπτομέρειες της υλοποίησης της υποδομής και τον τρόπο με τον οποίο τρέχουν οι εφαρμογές και οι υπηρεσίες κρύβονται από τον τελικό χρήστη[1].

Επειδή ο όρος cloud computing έχει πολλές πτυχές μπορούμε να γενικεύσουμε τη έννοια με τέτοιο τρόπο ώστε "*Cloud*" να θεωρείται κάθε σύστημα το οποίο επιτρέπει τη ανάθεση υπολογιστικών και αποθηκευτικών υπηρεσιών σε τελικούς χρήστες με κύριο χαρακτηριστικό το διαδίκτυο. Λόγω της γενίκευσης αυτής έχουμε το διαχωρισμό των σύννεφων σε δυο διαφορετικές κλάσεις. Ο πρώτος διαχωρισμός γίνεται με βάση τις προσφερόμενες υπηρεσίες και ο δεύτερος, όπως θα δούμε και παρακάτω, με βάση το μοντέλο ανάπτυξης[1][3].



Εικόνα 1: Cloud Computing Πηγή: Wikipedia, Cloud Computing[7]

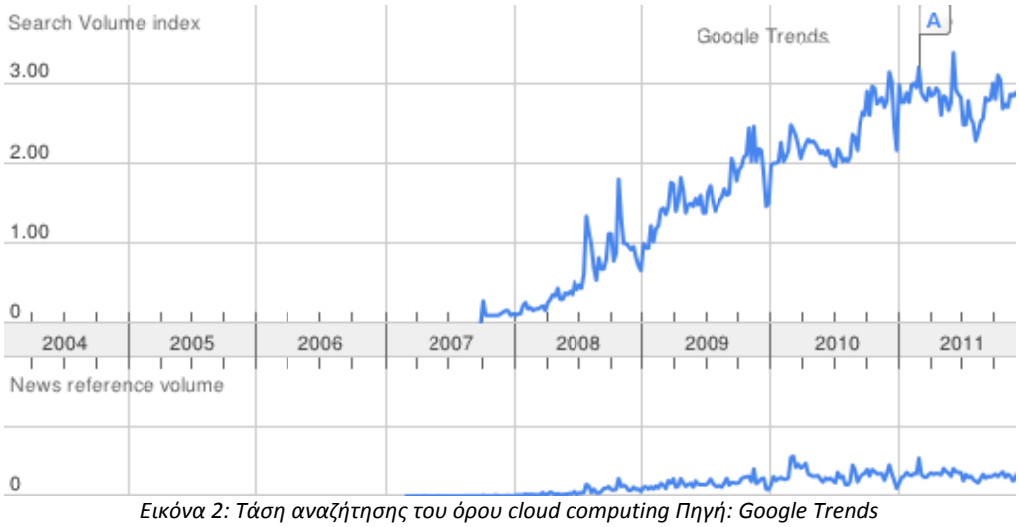
Το όνομα του cloud computing προέρχεται από το σχήμα του σύννεφου που συχνά χρησιμοποιούμε για να περιγραφή δικτύων και υποδεικνύει τη στενή σχέση του cloud computing με τη δικτύωση και το διαδίκτυο. Όμως το σύννεφο δεν είναι απλά ένας όμορφος όρος για το διαδίκτυο. Παρόλο που το διαδίκτυο είναι αναγκαίο για τη υποδομή του σύννεφου, το σύννεφο είναι κάτι παραπάνω από το διαδίκτυο. Το σύννεφο είναι ο "*χώρος*" όπου πάμε για να χρησιμοποιήσουμε τη "*τεχνολογία*" (εφαρμογές και υπηρεσίες) που θέλουμε όταν τη χρειαζόμαστε και

ούτε λεπτό παραπάνω. Δεν εγκαθιστούμε τίποτα στον υπολογιστή μας και δεν πληρώνουμε για "τεχνολογία" που δεν χρησιμοποιήσαμε[2].

Σύμφωνα με τη εταιρία ερευνών IDC[5], εκτιμάται ότι το ποσό των 42 δισεκατομμυρίων δολαρίων αναμένεται να δαπανηθεί στο έτος που διανύουμε από ιδιώτες και εταιρίες για υπηρεσίες σε υποδομή σύννεφου. Για να γίνει πιο ξεκάθαρη η αύξηση αυτή, το 2008 το ποσό αυτό ανήλθε μόλις στα 16 δισεκατομμύρια δολάρια.

Βλέπουμε cloud computing εφαρμογές παντού. Εφαρμογές σε smartphones χρησιμοποιούν τεχνολογία νέφους για να μπορεί ο χρήστης να έχει πρόσβαση στα αποθηκευμένα δεδομένα του που δεν χωράνε στη μνήμη μιας μικρής κινητής συσκευής. Εκπαιδευτικά και ερευνητικά ιδρύματα χρησιμοποιούν υποδομή νέφους για να αποθηκεύουν τις τεράστιες βιβλιοθήκες δεδομένων τους και τέλος εταιρίες κατασκευής ψηφιακών παιχνιδιών πειραματίζονται με τρόπους ώστε η πρόσβαση σε παιχνίδια τελευταίας λέξης της τεχνολογίας να μην απαιτούν αγορά ακριβού υλικού από τους χρήστες. Τα παραπάνω παραδείγματα δείχνουν πως το cloud computing έχει αρχίσει να μπαίνει στη ζωή μας.

Στη Εικόνα 2 βλέπουμε τη αύξηση του ενδιαφέροντος για αναζήτηση του όρου "*cloud computing*" σε σχέση με όρους όπως "*grid computing*", "*P2P*" και "*networking*" όπως καταγράφονται από το *Google Trends*[4]:



Εικόνα 2: Τάση αναζήτησης του όρου cloud computing Πηγή: Google Trends

Πλεονεκτήματα Cloud Computing

Παρακάτω παρουσιάζονται μερικά πλεονεκτήματα χρήσης υποδομών νέφους και παραδείγματα όπου εταιρίες και ιδρύματα μπορούν να ωφεληθούν και να μειώσουν τα κόστη τους αν εκμεταλλευτούν τις δυνατότητες που τους προσφέρει το cloud computing [1][2][3][11]:

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 καταντούν ασύμφορο. Τα οφέλη από τη παροχή λογισμικού από πάροχο υπηρεσιών νέφους είναι ότι μέσα σε λίγα λεπτά μπορείς να έχει έτοιμο και να τρέχει με τις άδειες χρήσης του το επιθυμητό λογισμικό. Το αντίτιμο για τις υπηρεσίες αυτές είναι μικρό και συνήθως χρεώνεται με τον μήνα ή χρησιμοποιείται το μοντέλο "*pay as you use*" και υπάρχουν 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 πιθανότερο 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 τουλάχιστον 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 τη εκτέλεση περίπλοκων υπολογισμών συχνά χρειάζονται μέρες ή και εβδομάδες για να εκτελεστούν 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 τη επεξεργαστική ισχύ αλλά και μνήμη ολόκληρου του δικτύου μειώνοντας το χρόνο εκτέλεσης των υπολογισμών.

Ανησυχίες για Cloud Computing

Οι μεγαλύτερες ανησυχίες γύρω από το cloud computing έχουν να κάνουν με τη προστασία-ιδιωτικότητα των προσωπικών δεδομένων και τη ασφάλεια. Η ιδέα της παράδοσης δεδομένων είτε είναι προσωπικά των χρηστών είτε ευαίσθητα για ένα οργανισμό σε τρίτη εταιρία πάροχο υποδομής νέφους είναι ο λόγος που τα διοικητικά στελέχη οργανισμών διστάζουν να εκμεταλλευτούν τις δυνατότητες που προσφέρει το cloud computing[1][11].

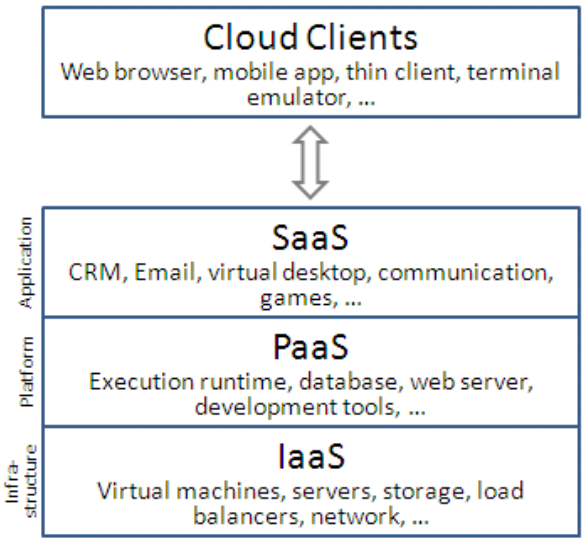
Οι άνθρωποι συνήθως αισθάνονται πιο ήσυχοι όταν τα δεδομένα τους βρίσκονται στον χώρο τους ακόμη και αν η προσφερόμενη ασφάλεια από πάροχους υπηρεσιών νέφους είναι η μέγιστη δυνατή. Σαν αντεπιχείρημα τις άποψης αυτής είναι ότι οι εταιρίες παροχής υπηρεσιών νέφους εξαρτούνται από τη φήμη τους. Αν ένας πάροχος δεν προσφέρει αξιόπιστη ασφάλεια τότε θα χάσει τους πελάτες της έτσι θα πρέπει πάντα να προσφέρει τη μέγιστη δυνατή ασφάλεια.

Ακόμη ένα θέμα είναι η ιδιωτικότητα των προσωπικών δεδομένων. Αν ένας χρήστης μπορεί να έχει πρόσβαση από οποιαδήποτε τοποθεσία τότε είναι πολύ πιθανόν η ιδιωτικότητα των δεδομένων του να διακυβεύεται. Για τη προστασία των προσωπικών δεδομένων όμως οι πάροχοι θα πρέπει να προσφέρουν τεχνικές εξουσιοδότησης (authentication) δίδοντας σε κάθε χρήστη *username* και *password* και τέλος να έχει πρόσβαση ο κάθε χρήστης μόνο σε αρχεία και δεδομένα που αφορούν τον ίδιο και τις αρμοδιότητες του στον οργανισμό.

Το τελευταίο θέμα που προκαλεί ανησυχία είναι και κάπως φιλοσοφικό. Όταν ένας οργανισμός χρησιμοποιεί τις υπηρεσίες νέφους ενός πάροχου για τη φύλαξη και διαμοιρασμό των δεδομένων του, τότε σε ποιον ανήκουν τα δεδομένα. Έτσι τίθεται το ερώτημα, *έχει ο πάροχος πρόσβαση και δικαίωμα διαμοιρασμού των δεδομένων του οργανισμού;* Η απάντηση σε αυτό το ερώτημα αποτελεί πηγή μελέτης νομικών εταιριών, κυβερνήσεων και πανεπιστημίων.

Διαχωρισμός ως προς προσφερόμενες υπηρεσίες

Ανάλογα με τις προσφερόμενες υπηρεσίες προς χρήστες ή ενδιαφερόμενες εταιρίες μπορούμε να διαχωρίσουμε το cloud computing σε 3 κατηγορίες [1][8][9][10]:



Εικόνα 3: Διαχωρισμός cloud computing ως προς τις προσφερόμενες υπηρεσίες Πηγή: Wikipedia[7]

SaaS (Software-as-a-Service)

Αναφέρεται συχνά και ως "*software on demand*" και αποτελεί τον πιο γνωστό τύπο cloud computing λόγω κυρίως της μεγάλης ευελιξίας, ποιότητας υπηρεσιών, υψηλής σταθερότητας και της ελάχιστης δυνατής συντήρησης που

απαιτεί. Με τον όρο SaaS εννοούμε τη παράδοση-παροχή λογισμικού από ένα πάροχο υπηρεσιών αντί της αγοράς του ίδιου του λογισμικού και της άδειας χρήσης του. Οι εφαρμογές και τα δεδομένα φιλοξενούνται κεντρικά στο σύννεφο από τον πάροχο και έτσι ο χρήστης μπορεί να έχει πρόσβαση στις εφαρμογές και δεδομένα που επιθυμεί από οπουδήποτε. Το μόνο που χρειάζεται ο χρήστης να έχει είναι ένα περιηγητή και πρόσβαση στο διαδίκτυο. Το μοντέλο αυτό είναι πολύ αποτελεσματικό στη μείωση του κόστους, ειδικά για οργανισμούς και επιχειρήσεις, αφού παρέχεται ως μηνιαίο λειτουργικό κόστος ή ανάλογα με τη χρήση (pay as you use) το οποίο συνήθως είναι κατά πολύ οικονομικότερο από την αγορά των αντίστοιχων αδειών χρήσης και υποδομής. Ακόμη δεν απαιτείται καμία συντήρηση ή αναβάθμιση, αφού ο τελικός χρήστης δε χρειάζεται να μεριμνήσει για τη διαθεσιμότητα, την κλιμάκωση, τη χωρητικότητα και το SLA (Συμφωνητικό Παροχής Υπηρεσιών) της υποδομής, της πλατφόρμας και της υπηρεσίας. Μερικά παραδείγματα SaaS είναι οι δωρεάν υπηρεσίες webmail όπως *Microsoft hotmail* και *Google gmail*, η σουίτα γραφείου *Google Docs*, η εφαρμογή λογιστικού περιβάλλοντος *Quickbooks Online* και η εφαρμογή διαχείρισης πωλήσεων της *salesforce.com*.

PaaS (Platform-as-a-Service)

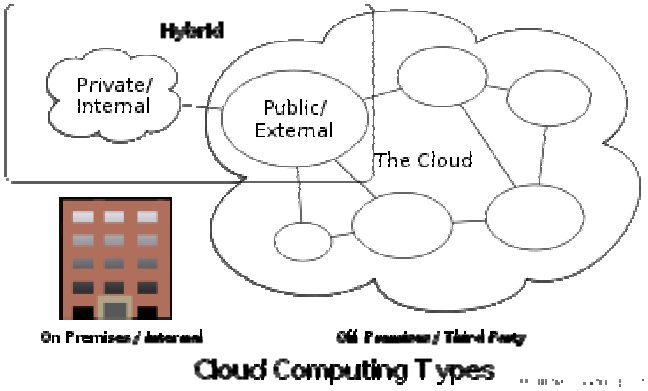
Στο μοντέλο αυτό δίδεται η δυνατότητα σε χρήστες ή εταιρίες να αναπτύσσουν εφαρμογές χρησιμοποιώντας τη υποδομή του νέφους. Το μοντέλο αυτό παρέχει τις κατάλληλες υπηρεσίες προκειμένου κάποιος να μπορέσει να αναπτύξει, να δοκιμάσει, να διαθέσει και να συντηρήσει εφαρμογές και υπηρεσίες μέσα ένα ενιαίο 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. Πολύ απλά, οι βιβλιοθήκες και τα εργαλεία ανάπτυξης εφαρμογών βρίσκονται κεντρικά στο νέφος και το μόνο που χρειάζεται ο χρήστης είναι ένα τερματικό με σύνδεση στο διαδίκτυο. Χρησιμοποιώντας το μοντέλο αυτό δεν χρειάζεται να ασχοληθεί ο χρήστης με τη συντήρηση του λειτουργικού συστήματος και της πλατφόρμας. Στο PaaS η χρέωση γίνεται ανάλογα με τη χρήση (pay as you use) έτσι ώστε να επιτυγχάνεται η πλήρης αξιοποίηση των υπολογιστικών πόρων που χρησιμοποιούνται σε σχέση με το κόστος χρήσης. Αν συνδυαστεί με το χαρακτηριστικό της αυτό-κλιμάκωσης μπορούμε να πετύχουμε τη διάθεση υπηρεσιών που να μπορούν να ανταποκρίνονται σε οποιαδήποτε ραγδαία ή αναμενόμενη μεταβολή χωρητικότητας (ισχύς, μνήμη, αποθηκευτικό χώρο, δίκτυο) που θα απαιτηθεί ανά πάσα χρονική στιγμή χωρίς να υπάρχει η δέσμευση εκ των προτέρων είτε με αγορά υποδομής, λογισμικού πλατφόρμας και δικτυακή γραμμή υψηλής χωρητικότητας. Μερικά παραδείγματα Paas είναι το *Google App Engine*, *Microsoft Azure*, η πλατφόρμα της Amazon *Amazon Elastic Beanstalk* και το *force.com* της salesforce.com.

IaaS (Infrastructure-as-a-Service)

Στο μοντέλο αυτό έχουμε τη παροχή υπολογιστικών και δικτυακών υποδομών ως υπηρεσία. Ο χρήστης ή η ενδιαφερόμενη εταιρεία μπορεί να υπενοικιάσει υποδομή ανάλογα με τις απαιτήσεις εκείνης της χρονικής στιγμής. Η χρέωση γίνεται ανάλογα με τη χρήση (pay as you use) αντί να απαιτείται η 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 νέφος. Μερικά παραδείγματα Iaas είναι το *Amazon Elastic Compute Cloud (EC2),* το *Rackspace Cloud* και το *Google Compute Engine*.

Διαχωρισμός ως προς το μοντέλο ανάπτυξης

Ανάλογα σε ποιους απευθύνονται οι προσφερόμενες υπηρεσίες αλλά και που βρίσκεται το νέφος μπορούμε να διαχωρίσουμε το cloud computing σε τρία μοντέλα ανάπτυξης [1][6][7]:



Εικόνα 4: Διαχωρισμός cloud computing ως προς το μοντέλο ανάπτυξης Πηγή: Wikipedia, Cloud Computing[7]

Public Cloud

Στο μοντέλο αυτό υπολογιστικοί πόροι, εφαρμογές και χώρος αποθήκευσης διατίθενται στο ευρύ κοινό από ένα πάροχο υπηρεσιών. Η πρόσβαση στις υπηρεσίες αυτές γίνεται μόνο μέσω διαδικτύου έτσι το μόνο που χρειάζεται ο χρήστης είναι ένα τερματικό με πρόσβαση στο διαδίκτυο. Η χρέωση των υπηρεσιών αυτών γίνονται συνήθως ανάλογα με τη χρήση (pay as you use) και τα πλεονεκτήματα προσφοράς υπηρεσιών σε Public Cloud είναι η χρέωση της υπηρεσίας μόνο για τους πόρους που έχουν χρησιμοποιηθεί, υπάρχει 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.

Private Cloud

Στο μοντέλο αυτό ολόκληρη η υποδομή του νέφους ανήκει σε ένα οργανισμό και μπορεί να διαχειρίζεται από αυτόν ή κάποιο τρίτο και η υποδομή μπορεί να βρίσκεται εσωτερικά στις κτηριακές εγκαταστάσεις του. 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. Σημαντικό χαρακτηριστικό του είναι το πολύ υψηλό κόστος απόκτησης και λειτουργίας του και αυτό είναι ίσως και το μεγαλύτερο μειονέκτημα του. Το Private cloud συχνά συγχέεται με το Virtualization, το όποιο όμως αποτελεί μόνο ένα μικρό μέρος αυτού, αφού ακόμα και ως ιδιωτικό σύννεφο θα πρέπει να έχει τα χαρακτηριστικά αυτόματης ανάκαμψης, αυτό-επιτήρησης, αυτό-διαχείρισης, αυτόματης επαναδιαμόρφωσης, δυνατότητας καθορισμού SLAs, και δυνατότητες κλιμάκωσης.

Community Cloud

Στο μοντέλο αυτό έχουμε διαμοιραζόμενη υποδομή μεταξύ μερικών οργανισμών με κοινές ανησυχίες όπως η ασφάλεια, η δικαιοδοσία και θέματα διαμοιρασμού και προστασίας δεδομένων που είναι και η μεγαλύτερη ανησυχία σε δημόσια σύννεφα. Η διαχείριση μπορεί να γίνεται εσωτερικά μέσω της κοινότητας ή μέσω ενός τρίτου οργανισμού και η υποδομή μπορεί να βρίσκεται σε χώρο της κοινότητας. Το σημαντικότερο πλεονέκτημα ενός Community Cloud είναι ότι μοιράζεται μεταξύ μερικών οργανισμών και όχι μόνο έναν έτσι αν δεν φθάνουν τα χρήματα για ιδιωτικό σύννεφο τότε η λύση είναι το νέφος κοινότητας.

Hybrid Cloud

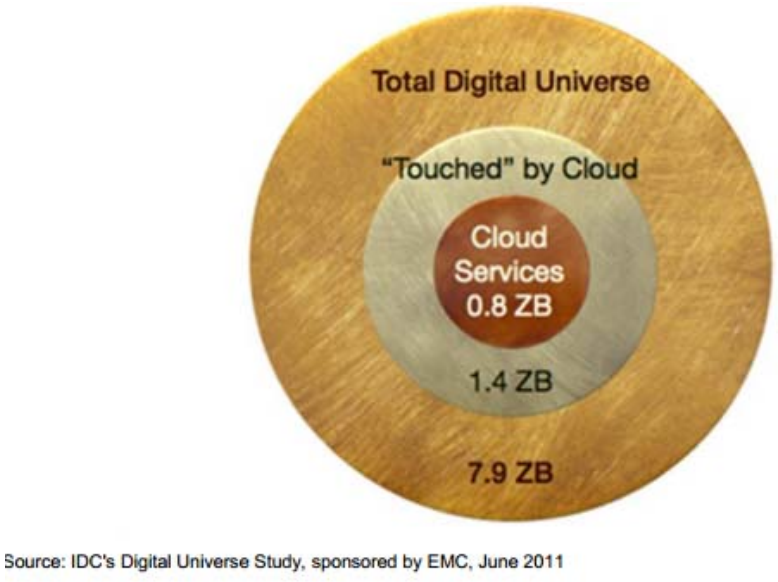
Στο μοντέλο αυτό έχουμε σύνθεση δύο ή περισσοτέρων μοντέλων ανάπτυξης (ιδιωτικό, δημόσιο ή κοινότητας) τα οποία αν και παραμένουν μοναδικές οντότητες στο νέο μοντέλο έρχονται και δένουν μαζί προσφέροντας τα πλεονεκτήματα πολλαπλών μοντέλων ανάπτυξης. Στο μοντέλο αυτό είναι σύνηθες ο οργανισμός να έχει δικό του ιδιωτικό νέφος όμως υπάρχει η δυνατότητα διασύνδεσης με δημόσιο νέφος για συγκεκριμένους σκοπούς.

Αποθηκευτικά νέφοι (Storage Clouds)

Η αποθήκευση δεδομένων σε αποθηκευτικό νέφος είναι η αξιοποίηση της τεχνολογίας του cloud computing για τη διαχείριση αποθηκευτικού χώρου κατόπιν ζήτησης (on demand) μέσω διεπαφής με τέτοιο τρόπο ώστε να επιτυγχάνεται η απόκρυψη (abstraction) της αρχιτεκτονικής και τρόπου διαχείρισης του αποθηκευτικού χώρου από τον πελάτη. Ακόμη η διεπαφή αυτή, αποκρύπτει τη τοποθεσία του χώρου αποθήκευσης έτσι ώστε να είναι αδιάφορο για τον πελάτη που βρίσκονται τα δεδομένα του.

Η SNIA[12] έχει δημιουργήσει τον όρο "*Data Storage as a Service (DaaS)*" για να περιγράψει τη παροχή υπηρεσιών αποθήκευσης κατόπιν ζήτησης (on demand) σε πελάτες πάνω σε κατανεμημένο σύστημα. Υπάρχουν μερικοί που προτιμούν να περιγράφουν τα συστήματα αυτά με τον όρο "*Storage as a Service* (*STaaS*)"[1].

Με τον ρυθμό παραγωγής δεδομένων σήμερα, δεν είναι έκπληξη το γεγονός ότι τα αποθηκευτικά νέφοι γίνονται ολοένα και πιο δημοφιλής. Σύμφωνα με τη IDC[13], υπολογίζεται ότι το 2015 θα παράγονται δεδομένα της τάξης των 7.9ZB και ότι το 10% εξ αυτών θα αποθηκεύονται σε αποθηκευτικά νέφοι ενώ ένα ποσοστό της τάξης του 17% θα περνάνε από αποθηκευτικά νέφοι.



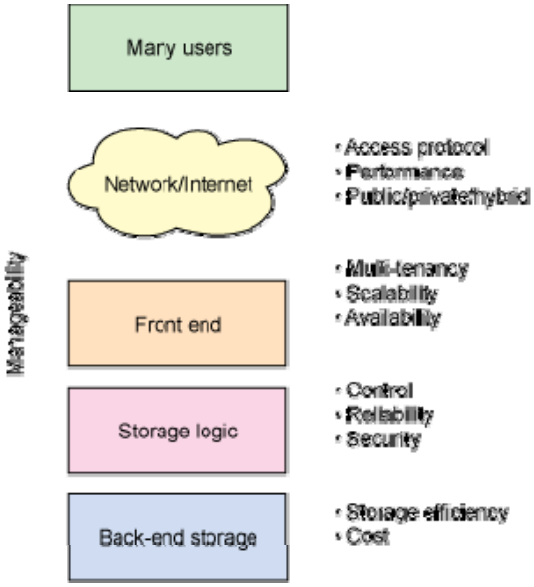
Εικόνα 5: Το ποσοστό των δεδομένων παγκοσμίως που συσχετίζονται με αποθηκευτικά νέφοι Πηγή: IDC[13]

Αρχιτεκτονική αποθηκευτικού νέφους

Η αρχιτεκτονική ενός αποθηκευτικού νέφους έχει ως κύριο στόχο τη παράδοση χώρου αποθήκευσης κατόπιν ζήτησης με τέτοιο τρόπο ώστε να επιτυγχάνεται υψηλή κλιμάκωση, διαχείριση και αυτοσυντήρηση.

Γενικά, η αρχιτεκτονικά 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 νέφοι 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.

Στη εικόνα 6 παρατηρούμε μια γενική αρχιτεκτονική αποθηκευτικού νέφους. Παρακάτω περιγράφονται κάποια από τα σημαντικότερα χαρακτηριστικά που πρέπει να διαθέτει ένα αποθηκευτικό νέφος[14]:



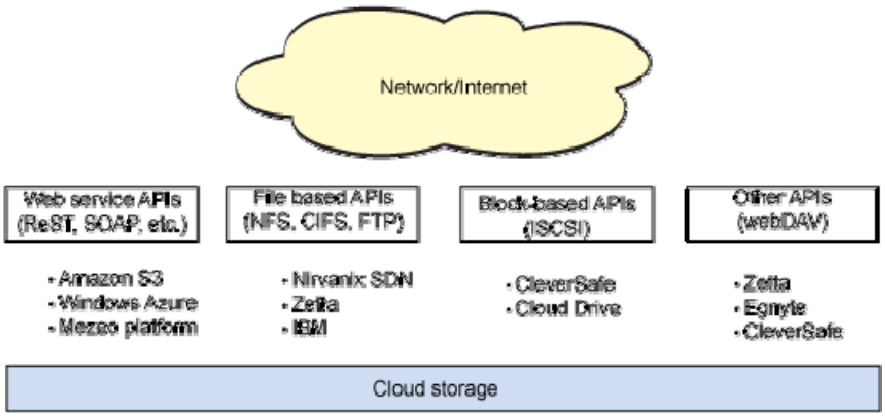
Εικόνα 6: Γενική αρχιτεκτονική αποθηκευτικού νέφους Πηγή: IBM developerWorks[14]

Χαρακτηριστικά αποθηκευτικού νέφους

Διαχειρισιμότητα: Ίσως το πιο σημαντικό πράγμα που λαμβάνει μια εταιρία υπόψη 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 παράγοντας είναι η διαχείριση των πόρων του συστήματος στη οποία δεν μπορούμε να αποδώσουμε μια τιμή εξ αρχής. Για το λόγο αυτό θέλουμε να

επιτύχουμε τη 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. Τέλος ένα πολύ ενδιαφέρον πρωτόκολλο είναι το WebDAV το οποίο βασίζεται στο HTTP και θεωρεί το διαδίκτυο ως πόρος για διάβασμα και εγγραφή.



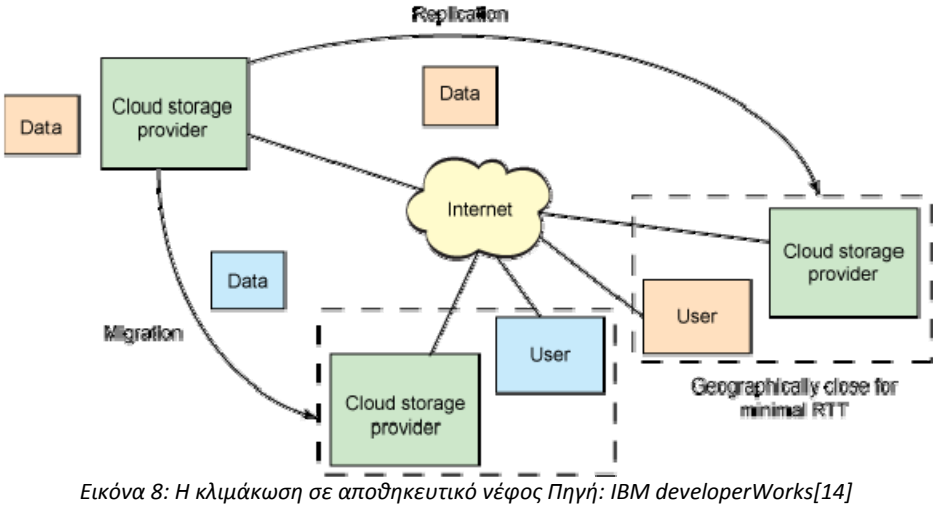
Εικόνα 7: Μέθοδοι πρόσβασης σε αποθηκευτικά νέφοι Πηγή: IBM developerWorks[14]

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 τη 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. Στο 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 το 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 τη 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 τη κλιμάκωση. Η κλιμάκωση δεν πρέπει να γίνεται μόνο στη 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 στο χρήστη.



Εικόνα 8: Η κλιμάκωση σε αποθηκευτικό νέφος Πηγή: IBM developerWorks[14]

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 στη 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 υποσύνολο τους.

REST

Είναι ένα υβριδικό αρχιτεκτονικό μοντέλο για δικτυακά συστήματα, το οποίο συνδυάζει υπάρχοντα αρχιτεκτονικά μοντέλα (όπως το μοντέλο client-server), σε συνδυασμό με αρχές (principles) και περιορισμούς (constrains) για το σχεδιασμό υπηρεσιών του διαδικτύου (web services) [15][16][17].

Ο όρος "*REpresentational State Transfer*" παρουσιάστηκε για πρώτη φορά το 2000 στη διδακτορική διατριβή του Roy Fielding, με τίτλο "*Architectural Styles and the Design of Network-based Software Architectures*"[16].

Το αρχιτεκτονικό μοντέλο REST

Ένα σύστημα το οποίο εφαρμόζει τους κανόνες και περιορισμούς της αρχιτεκτονικής REST, λέμε ότι είναι "*RESTful*" [18].

Ένα "*RESTful*" σύστημα αποτελείται από πελάτες (clients) και εξυπηρετητές (servers). Η διαδικτυακή υπηρεσία αποτελείται από πόρους (Resource Oriented Service). Ο πελάτης αποστέλλει στον εξυπηρετητή αίτημα (request) και ο εξυπηρετητής επεξεργάζεται κατάλληλα το αίτημα, επιστρέφοντας στον πελάτη απάντηση (response). Οι αιτήσεις και οι απαντήσεις είναι σχεδιασμένες γύρω από τη μεταφορά αναπαράστασης πόρων. Ένας πόρος μπορεί να είναι οποιαδήποτε έννοια (κάποιος χρήστης, ένα προϊόν κ.α) στον οποίο δίνουμε μια μοναδική διεύθυνση. Για το σκοπό αυτό χρησιμοποιούμε καθολικούς ταυτοποιητές (URIs). Η αναπαράσταση (representation) ενός πόρου (resource) είναι τυπικά ένα έγγραφο το οποίο απεικονίζει τη τρέχουσα κατάσταση (state) ενός πόρου.

Ο όρος REST προέρχεται από τη έννοια της μεταφοράς κατάστασης στον πελάτη. Με κάθε νέο αίτημα του πελάτη, επιστρέφεται από τον εξυπηρετητή, μια απάντηση, με αποτέλεσμα να μεταβαίνει σε μια νέα κατάσταση.

Το μεγαλύτερο αλλά και πιο γνωστό σύστημα, το οποίο εφαρμόζει το αρχιτεκτονικό μοντέλο REST, είναι ο Παγκόσμιος Ιστός (World Wide Web). Όπως ο Παγκόσμιος Ιστός, έτσι και κάθε καλο-σχεδιασμένη "*RESTful*" διαδικτυακή υπηρεσία, θα πρέπει να συμπεριφέρεται ως ένα δίκτυο από ιστοσελίδες (virtual state machine), όπου ο κάθε χρήστης περιηγείται μέσα στη εφαρμογή επιλέγοντας συνδέσμους (μεταφορείς κατάστασης), με αποτέλεσμα μια νέα ιστοσελίδα (η αναπαράσταση της νέας κατάστασης) η οποία μεταφέρθηκε ως απάντηση από τον εξυπηρετητή [16].

Οι βασικές αρχές και περιορισμοί της αρχιτεκτονικής REST

Οι ακόλουθες αρχές και περιορισμοί θα πρέπει να ισχύουν σε μια διαδικτυακή υπηρεσία για να θεωρείται “RESTful”. Είναι σημαντικό να αναφερθεί ότι αν και πρέπει να εφαρμόζονται οι περιορισμοί αυτοί, δεν γίνεται καμία αναφορά στο τρόπο υλοποίησης τους[15][16][17][19].

- **Μοντέλο πελάτη-εξυπηρετητή**

Μια ομοιόμορφη διεπαφή (uniform interface) διαχωρίζει τους πελάτες από τους εξυπηρετητές. Με τον διαχωρισμό αυτό για παράδειγμα, οι πελάτες δεν ενδιαφέρονται για τη αποθήκευση των δεδομένων. Αυτό βοηθάει στη βελτίωση της φορητότητας του κώδικα πελάτη. Αντίστοιχα, οι εξυπηρετητές δεν ενδιαφέρονται για τη διεπαφή του χρήστη (user interface) ή για τη κατάσταση του χρήστη (user state). Αυτό βοηθάει τους εξυπηρετητές ώστε να παραμένουν απλοί και επεκτάσιμοι (scalable). Με τον διαχωρισμό αυτό, τελικά, οι δυο πλευρές μπορούν να αναπτυχθούν ανεξάρτητα.

- **Ρητή χρήση του πρωτοκόλλου HTTP και των μεθόδων του**

Πρέπει να γίνεται ρητή χρήση των μεθόδων του HTTP πρωτοκόλλου. Με τον κανόνα αυτό πετυχαίνουμε μια-προς-μια αντιστοίχιση ανάμεσα στις CRUD λειτουργίες, δηλ. Δημιουργία(Create)-Ανάγνωση(Read)-Ενημέρωση(Update)-Διαγραφή(Delete), και των HTTP μεθόδων.

Σύμφωνα με τη αντιστοίχιση αυτή:

- Για τη δημιουργία ενός πόρου στο εξυπηρετητή, χρησιμοποιούμε τη μέθοδο POST ή PUT.
- Για τη ανάκτηση πόρου από τον εξυπηρετητή, χρησιμοποιούμε τη μέθοδο GET.
- Για τη αλλαγή κατάστασης πόρου ή για τη ενημέρωση του, χρησιμοποιούμε τη μέθοδο POST ή PUT.
- Για τη διαγραφή πόρου από τον εξυπηρετητή, χρησιμοποιούμε τη μέθοδο DELETE.

Στους πίνακες 1 και 2 φαίνονται δυο παραδείγματα χρήσης μιας διαδικτυακής “*RESTful*” υπηρεσίας που ορίζεται στη Εικόνα 9:

Πίνακας 1: Παράδειγμα ΑΙΤΗΣΗΣ/ΑΠΑΝΤΗΣΗΣ δημιουργίας νέου πελάτη με HTTP POST

Πίνακας 1a: Παράδειγμα ΑΙΤΗΣΗΣ δημιουργίας νέου πελάτη με HTTP POST
`POST /customers/1234 HTTP/1.1` `Host: mydomain.com`

Πίνακας 1b: Παράδειγμα ΑΠΑΝΤΗΣΗΣ δημιουργίας νέου πελάτη με HTTP POST
`201 Created HTTP/1.1`

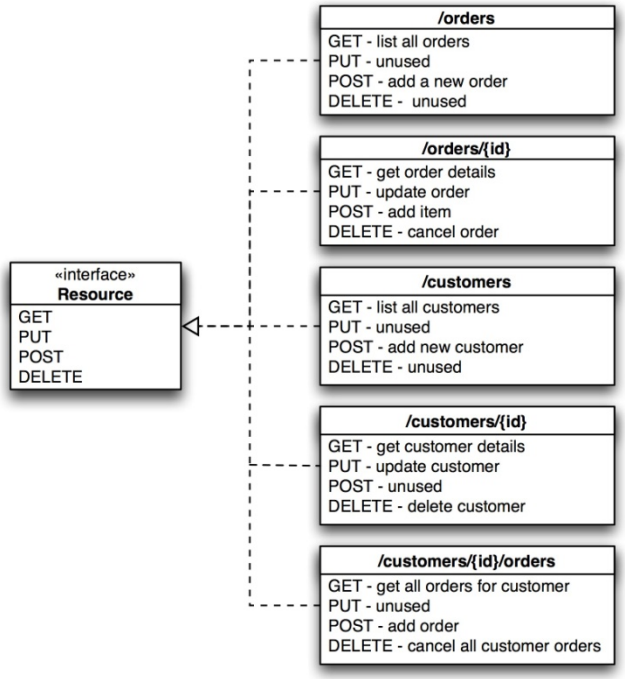
Πίνακας 2: Παράδειγμα ΑΙΤΗΣΗΣ/ΑΠΑΝΤΗΣΗΣ λήψης λίστας πελατών με HTTP GET

Πίνακας 2a: Παράδειγμα ΑΙΤΗΣΗΣ λήψης λίστας πελατών με HTTP GET

```
GET /customers HTTP/1.1
Host: mydomain.com
Accept: application/xml
```

Πίνακας 2b: Παράδειγμα ΑΠΑΝΤΗΣΗΣ λήψης λίστας πελατών σε XML με HTTP GET

```
<customers>
   <customer>
      <id>1233</id>
      <name>Pavlos Leggos</name>
      <address>25, Othonos, Athens, Greece</address>
   </customer>
   <customer>
      <id>1234</id>
      <name>Giorgos Aggelou</name>
      <address>12, Akropoleos, Athens, Greece</address>
   </customer>
</customers>
```



Εικόνα 9: Παράδειγμα διαδικτυακής υπηρεσίας με RESTful HTTP προσέγγιση

Όπως φαίνεται και από τα παραδείγματα χρησιμοποιούμε ένα μικρό σύνολο από ρήματα (οι HTTP μέθοδοι: GET, POST, PUT, DELETE) και τα εφαρμόζουμε σε ένα άπειρο σύνολο ουσιαστικών (URIs)[20]. Έτσι δουλεύει και ο Παγκόσμιος Ιστός. Όταν πληκτρολογούμε στον περιηγητή μας τη διεύθυνση που θέλουμε, ο περιηγητής χρησιμοποιεί τη HTTP μέθοδο GET για τη ανάκτηση της σελίδας που ζητάμε.

Η στενή σχέση της αρχιτεκτονικής REST με το πρωτόκολλο HTTP δεν είναι τυχαία. Η αρχιτεκτονική REST αναπτύχθηκε παράλληλα με το πρωτόκολλο

HTTP/ 1.1, με τον Roy Fielding να είναι ένας από τους συγγραφείς των αρχών και κανόνων των προδιαγραφών του HTTP/1.1.

- **Stateless Server-Side (άνευ κατάστασης πλευρά εξυπηρετητή)**

Η επικοινωνία πελάτη-εξυπηρετητή πρέπει να ακολουθείται από τον περιορισμό ότι κανένα περιεχόμενο που αφορά τη κατάσταση του χρήστη δεν θα πρέπει να αποθηκεύεται στη πλευρά του εξυπηρετητή κατά τη μεταφορά των αιτημάτων. Έτσι, κάθε αίτημα από τον πελάτη θα πρέπει να περιέχει όλη τη πληροφορία αναγκαία για τη επεξεργασία του αιτήματος και οποιαδήποτε κατάσταση συνεδρίας (session state) πρέπει να κρατείται στη πλευρά του πελάτη.

- **Αυτοπροσδιοριστικά URIs**

Σε κάθε πόρο της διαδικτυακής υπηρεσίας πρέπει να δίνεται URI που να δείχνει τη έννοια και τη λειτουργία που εξυπηρετεί. Για ακόμη καλύτερη κατανόηση είναι βοηθητικό η διεύθυνση να έχει μορφή που ακολουθεί τη διαδρομή σε κατάλογο με τις υποενότητες του όπως παρακάτω:

```
http://www.mydomain.com/discussion/2012/07/19/{topic}
```

- **Μεταφορά δεδομένων από τον εξυπηρετητή σε XML ή JSON**

Η αναπαράσταση ενός πόρου, που επιστρέφει ως απάντηση ο εξυπηρετητής, μας δείχνει τη τρέχουσα κατάσταση του πόρου τη στιγμή της αίτησης του πελάτη. Επομένως αυτό που πρέπει να επιστρέφει ο εξυπηρετητής είναι το στιγμιότυπο του πόρου τη δεδομένη χρονική στιγμή. Το στιγμιότυπο αυτό μπορεί να είναι κάτι απλό όπως η εγγραφή από μια βάση δεδομένων ή η αναπαράσταση ενός αντικειμένου αν το σύστημα μας χρησιμοποιεί κάποιο μοντέλο δεδομένων. Στη δεύτερη περίπτωση τα αντικείμενα που αποτελούν το μοντέλο δεδομένων συνδέονται μεταξύ τους με σχέσεις αλληλεπίδρασης και είναι χρήσιμο οι σχέσεις αυτές να τις μεταφέρουμε στο περιεχόμενο της αναπαράστασης, δίνοντας τη δυνατότητα να παρέχονται τα URIs των σχετιζομένων αντικειμένων. Παράδειγμα τέτοιας αναπαράστασης σε XML είναι το παρακάτω:

Πίνακας 3: XML αναπαράσταση νήματος σχολιασμού

```
<?xml version="1.0"?>
<discussion date="{date}" topic="{topic}">
    <comment>{comment}</comment>
    <replies>
       <reply from=joe@mail.com href="/discussion/topics/{topic}/joe"/>
       <reply from="bob@mail.com" href="/discussion/topics/{topic}/bob"/>
    </replies>
</discussion>
```

Πίνακας 4: Τα πιο συνηθισμένα MIME Types που χρησιμοποιούμε σε RESTful υπηρεσίες

JSON	application/json
XML	application/xml
XHTML	application/xhtml+xml

- **Cachable**

Στον Παγκόσμιο Ιστό οι πελάτες μπορούν να αποθηκεύουν (cache) απαντήσεις (responses). Οι απαντήσεις μιας RESTful υπηρεσίας θα πρέπει να αναφέρουν ρητά αν είναι επαναχρησιμοποιήσιμες (cachable) ή όχι, έτσι ώστε ο πελάτης να λαμβάνει πάντα τα σωστά δεδομένα.

- **Layered System**

Ο πελάτης δεν ξέρει αν είναι συνδεδεμένος απευθείας με τον τελικό εξυπηρετητή (end server) ή σε κάποιο ενδιάμεσο. Οι ενδιάμεσοι εξυπηρετητές βοηθούν στη βελτίωση της επεκτασιμότητας εφαρμόζοντας ισολογισμό του φόρτου εργασίας με μοιραζόμενες cache και ακόμη εφαρμόζουν πολιτικές και ελέγχους ασφαλείας κρατώντας απλό τον τελικό εξυπηρετητή.

- **Code on Demand** (προαιρετικό)

Οι εξυπηρετητές μπορούν να επεκτείνουν τη λειτουργικότητα του πελάτη μεταφέροντας σε αυτόν εκτελέσιμο κώδικα. Τέτοια παραδείγματα είναι Java Applets ή JavaScript scripts.

REST έναντι SOAP (Simple Object Access Protocol)

Η ευρεία ανταπόκριση που βρήκε η αρχιτεκτονική REST για τη ανάπτυξη υπηρεσιών διαδικτύου έναντι υπηρεσιών βασισμένες σε SOAP ή WSDL, βρίσκεται στο ότι είναι -με διαφορά- ένα αρχιτεκτονικό μοντέλο πολύ πιο απλό και προσιτό.

Ίσως η πιο σημαντική διαφορά της αρχιτεκτονικής REST σε σχέση με το SOAP ή άλλα παρεμφερή RPC (Remote Procedure Call) είναι ότι σε RPC λέμε "*όρισε κάποιες μεθόδους οι οποίες κάνουν κάτι*" ενώ σε REST λέμε "*όρισε κάποιους πόρους και θα έχουν αυτές τις μεθόδους*". Κάποια πλεονεκτήματα της αρχιτεκτονικής REST για διαδικτυακές υπηρεσίες είναι[21]:

- **Δουλεύει πολύ καλά σε περιορισμένο bandwidth και πόρους**: Ένας από τους λόγους για τους οποίους λέμε ότι η αρχιτεκτονική REST είναι "*lightweight*" οφείλεται στο ότι η επιστρεφόμενη δομή μπορεί να είναι οτιδήποτε, XML, JSON ή HTML. Ένας άλλος λόγος είναι ότι μπορείς να

χρησιμοποιήσεις οποιονδήποτε περιηγητή γιατί η REST προσέγγιση χρησιμοποιεί τις γνωστές HTTP μεθόδους. Τέλος με REST μπορούμε να χρησιμοποιήσουμε το αντικείμενο *XMLHttpRequest* το οποίο υποστηρίζεται από όλους τους περιηγητές, δίνοντας ως μπόνους τη χρήση AJAX για τις κλήσεις μας.

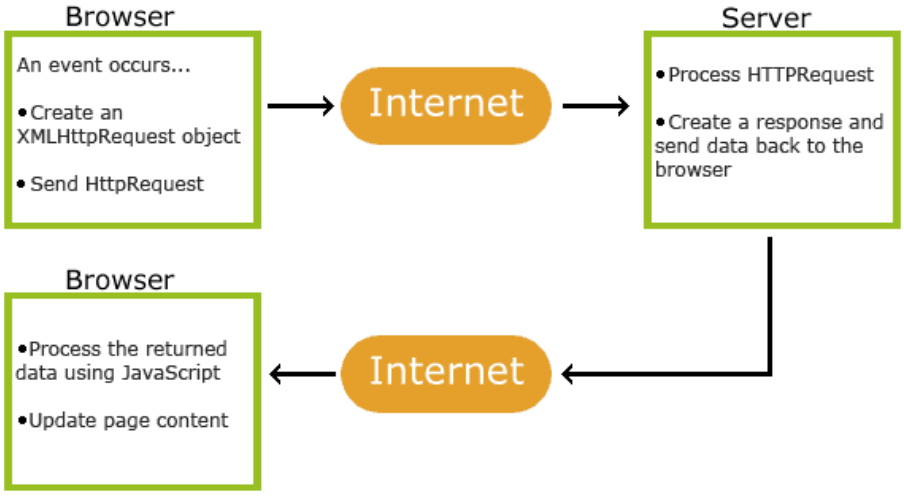
- **Με REST έχουμε stateless λειτουργίες**: Αν η εφαρμογή μας θα χρησιμοποιεί stateless CRUD λειτουργίες τότε κερδίζουμε στο ότι μπορούμε να επαναχρησιμοποιούμε (cache) τις απαντήσεις που παίρνουμε από τον εξυπηρετητή.

AJAX - Asynchronous JavaScript and XML

Είναι ένα σύνολο αλληλένδετων τεχνικών του διαδικτύου, τις οποίες χρησιμοποιούμε στη πλευρά του πελάτη για τη δημιουργία ασύγχρονων διαδικτυακών εφαρμογών. Με AJAX, οι διαδικτυακές εφαρμογές μπορούν να στείλουν αλλά και να λάβουν από τον εξυπηρετητή δεδομένα ασύγχρονα στο παρασκήνιο (background) χωρίς να επηρεάζεται η συμπεριφορά και η εμφάνιση της τρέχουσας σελίδας.

Τα δεδομένα μπορούν να ανακτηθούν χρησιμοποιώντας το αντικείμενο *XMLHttpRequest* του περιηγητή. Παρά το όνομα, η χρήση XML δεν απαιτείται και συχνά αντί αυτού χρησιμοποιείται JSON και οι αιτήσεις δεν χρειάζεται να είναι ασύγχρονες[22].

Το AJAX δεν είναι γλώσσα προγραμματισμού ούτε είναι μια τεχνική από μόνη της αλλά όπως αναφέρθηκε είναι ένα σύνολο αλληλένδετων τεχνικών και πρότυπων του διαδικτύου. Χρησιμοποιούμε HTML και CSS για τη σήμανση και παρουσίαση των δεδομένων. Στο DOM έχουμε πρόσβαση μέσω JavaScript για τη δυναμική εμφάνιση των δεδομένων αλλά και για τη αλληλεπίδραση με τον χρήστη. Με JavaScript και μέσω του API του αντικειμένου XMLHttpRequest μας παρέχεται μέθοδος για ανταλλαγή δεδομένων, ασύγχρονα μεταξύ του περιηγητή του πελάτη και του εξυπηρετητή για να αποφεύγουμε τις άσκοπες πλήρεις επαναφορτώσεις τις σελίδας[23].



Εικόνα 10: Πως δουλεύει το AJAX Πηγή: w3shools[23]

Παρακάτω παρουσιάζεται κώδικας για τη μεριά του πελάτη όπου με χρήση JavaScript έχουμε πρόσβαση στις μεθόδους *open()* και *send()* του αντικειμένου `XMLHttpRequest`. Η επεξεργασία της απάντησης από τον εξυπηρετητή γίνεται

από τη συνάρτηση που αναθέτουμε στο πεδίο `onreadystatechange` του αντικειμένου `XMLHttpRequest`:

Πίνακας 5: Παράδειγμα κώδικα πελάτη για δημιουργία AJAX κλήσης

```
var xmlhttp;
if (window.XMLHttpRequest){
  // code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
}
else{
  // code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.onreadystatechange=
  function(){
      if (xmlhttp.readyState==4 && xmlhttp.status==200){
         document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
      }
  }
xmlhttp.open("GET","URL_TO_SERVER",true);
xmlhttp.send();
```

Μερικά πλεονεκτήματα χρήσης AJAX είναι ότι μειώνεται η κίνηση δεδομένων μεταξύ πελάτη-εξυπηρετητή έτσι μειώνεται ο χρόνος απόκρισης αυξάνοντας τη επίδοση και τη ταχύτητα της διαδικτυακής υπηρεσίας. Με τη χρήση εναλλακτικών της XML όπως JSON έχουμε άμεση πρόσβαση των δεδομένων μας, σαν να πρόκειται για πίνακα, σε ζεύγη (κλειδί, τιμή). Υπάρχουν έτοιμες open source βιβλιοθήκες όπως JQuery, Prototype κ.α. που κάνουν τη χρήση AJAX πιο απλή, παρέχοντας έτοιμες συναρτήσεις και τέλος η επικοινωνία με AJAX γίνεται μέσω του πρωτοκόλλου HTTP.

Μερικά μειονεκτήματα που σιγά-σιγά όμως εξαλείφονται είναι ότι στη προ HTML5 εποχή με τη χρήση AJAX δεν υπήρχε τρόπος να καταχωρηθεί η σελίδα στο ιστορικό του περιηγητή με αποτέλεσμα τη μη σωστή χρήση του "*back button*". Οι μηχανές αναζήτησης όπως αυτή της Google δεν μπορεί να προσθέσει στο ευρετήριο της AJAX σελίδες. Αν για λόγους ασφαλείας είναι απενεργοποιημένη η JavaScript στον περιηγητή, τότε δεν δουλεύει η διαδικτυακή υπηρεσία και τέλος AJAX κλήσεις για λόγους ασφαλείας δεν μπορούν να γίνουν σε host διαφορετικό από αυτόν που μας "*σέρβιρε*" τη τρέχουσα σελίδα[24].

HTML5

Η HTML5 είναι μια συλλογή από χαρακτηριστικά, τεχνολογίες και APIs, που φέρνουν τη δύναμη ενός υπολογιστή desktop και τη ζωντανή εμπειρία πολυμέσων στο διαδίκτυο, ενώ ενισχύεται η διαδραστικότητα και η συνδεσιμότητα που είναι χαρακτηριστικά του πυρήνα του διαδικτύου.

Η HTML5 περιλαμβάνει την πέμπτη αναθεώρηση της γλώσσας σήμανσης HTML, CSS3 και μια σειρά από JavaScript APIs. Μαζί, οι τεχνολογίες αυτές επιτρέπουν τη δημιουργία πολύπλοκων εφαρμογών και υπηρεσιών του διαδικτύου όπου προηγουμένως μόνο σε desktop πλατφόρμες μπορούσαν να δημιουργηθούν[25]. Η HTML5 θα αντικαταστήσει τη HTML4, XHTML1 και DOM Level 2.

Αρκετά χαρακτηριστικά της HTML5 έχουν ενσωματωθεί στους σημερινούς περιηγητές, αν και η τελική έκδοση της HTML5 (final recommendation) σύμφωνα με τη κοινότητα W3C[26] αναμένεται το 2014.

Η HTML5 δεν ανήκει σε κάποια εταιρεία ή περιηγητή αλλά είναι δημιούργημα της κοινότητας WHATWG[27] σε συνδυασμό με τεχνολογικούς κολοσσούς όπως η Google, Microsoft, Apple, Mozilla Foundation, Facebook, IBM, HP, Adobe κ.α. Στόχος της κοινότητας και των εταιριών αυτών είναι να συνεργαστούν έτσι ώστε να δημιουργήσουν καθολικά πρότυπα (universal standards) για τους περιηγητές και να αυξήσουν τις δυνατότητες του διαδικτύου.

Η επόμενη γενιά των διαδικτυακών υπηρεσιών θα μπορούν να τρέχουν υψηλής επίδοσης γραφικών, να δουλεύουν offline, να αποθηκεύουν μεγάλο μέγεθος πληροφορίας στον πελάτη, να εκτελούν γρήγορους υπολογισμούς και να αλληλεπιδρούν με τον χρήστη σε ένα νέο και βελτιωμένο επίπεδο.

Τα τρία κυριότερα χαρακτηριστικά της HTML5 που χρησιμοποιήσαμε για τη διαδικτυακή υπηρεσία που φτιάξαμε είναι:

- Το στοιχείο <canvas> για τον δυναμικό σχεδιασμό γραφικών παραστάσεων.
- Τη αποθήκευση δεδομένων στη πλευρά του πελάτη χρησιμοποιώντας το API του sessionStorage.
- Τη αποστολή events από τη πλευρά του εξυπηρετητή (Server-Side Events) και τη επεξεργασία τους στη πλευρά του πελάτη με το EventSource API.

<canvas>

Ο “καμβάς” είναι μια σχεδιαστική περιοχή (drawable region) η οποία καθορίζεται από κώδικα HTML, δίνοντας για παραμέτρους το ύψος και το πλάτος του:

Πίνακας 6: Παράδειγμα ορισμού canvas

```
<canvas id="myCanvas" width="525" height="350">[No canvas support]</canvas>
```

Η μεγάλη δύναμη του καμβά είναι ότι μπορούμε να έχουμε πρόσβαση στη περιοχή αυτή μέσω JavaScript, χρησιμοποιώντας τις συναρτήσεις που μας δίνονται οι οποίες είναι πολύ όμοιες με κοινά 2D APIs έτσι μπορούμε να δημιουργήσουμε δυναμικά γραφικά. Μερικές χρήσεις που δείχνουν τις δυνατότητες του καμβά είναι η κατασκευή δυναμικών γραφικών παραστάσεων, κινούμενα σχέδια, παιχνίδια κ.α.[28]

HTML5 Web Storage και sessionStorage

Μια από τις σπουδαιότερες καινοτομίες που φέρνει η HTML5 είναι η δυνατότητα που δίνεται στις διαδικτυακές υπηρεσίες, να αποθηκεύουν δεδομένα τοπικά στον περιηγητή του χρήστη. Τη δυνατότητα αυτή τη λέμε “*Web Storage*”[29]. Παλαιότερα αυτό γινόταν μόνο με τη χρήση “*cookies*”.

Το web storage είναι ασφαλέστερο και ταχύτερο έναντι της χρήσης cookies. Μερικά από τα πλεονεκτήματα του web storage έναντι των cookies είναι[29][30]:

- Με τη χρήση web storage, τα δεδομένα δεν συμπεριλαμβάνονται σε κάθε κλήση προς τον εξυπηρετητή αλλά μόνο εφόσον ζητηθούν σε αντίθεση με τα cookies όπου όλα τα δεδομένα στέλνονται σε κάθε αίτημα.
- Με web storage μπορούμε να αποθηκεύσουμε μεγάλο μέγεθος δεδομένων στον περιηγητή μας χωρίς να επηρεάζεται η απόδοση της διαδικτυακής υπηρεσίας, εφόσον τα δεδομένα αυτά αποθηκεύονται τοπικά.
- Το μέγιστο μέγεθος ενός cookie είναι 4KB ενώ τοπικά μπορούμε να αποθηκεύσουμε δεδομένα της τάξης των MB. Σε Microsoft Internet Explorer 8+ μπορούμε να αποθηκεύσουμε μέχρι 10MB ενώ σε Mozilla Firefox και Google Chrome μέχρι 5MB.

Υπάρχουν δυο είδη web storage. Το πρώτο είδος ονομάζεται localStorage με τα δεδομένα να μην έχουν “*ημερομηνία λήξης*”.

Το δεύτερο είδος ονομάζεται sessionStorage και τα δεδομένα αποθηκεύονται μόνο για τη τρέχουσα συνεδρία. Αυτό σημαίνει ότι αν ο χρήστης κλείσει το tab ή τον περιηγητή του, τότε τα δεδομένα χάνονται.

Ένα από τα μεγαλύτερα πλεονεκτήματα του sessionStorage, το οποίο εκμεταλλευόμαστε και στη δική μας διαδικτυακή εφαρμογή, είναι ότι μπορούμε με sessionStorage στη πλευρά του πελάτη να πετύχουμε ότι και το HTTP Session στη πλευρά του εξυπηρετητή. Αυτό είναι ένα πολύ μεγάλο πλεονέκτημα γιατί αν για παράδειγμα θέλουμε να κρατάμε το username ενός χρήστη σε μια "*RESTful*" διαδικτυακή υπηρεσία, τότε για να παραμένει stateless η πλευρά του εξυπηρετητή δεν θα πρέπει να αποθηκεύεται στο HTTP session.

Πίνακας 7: Παράδειγμα χρήσης sessionStorage με JavaScript – Αποθήκευση/Ανάκτηση δεδομένων

```
var user="demetris"
if(sessionStorage == null)
    alert("sessionStorage not supported by the browser");
else
    sessionStorage.setItem('username',user);
```

```
if(sessionStorage == null)
    alert("sessionStorage not supported by the browser");
else
    alert(sessionStorage.username);
```

Server-Side Events (SSEs) – EventSource API

Τα Server Side Events (SSE) είναι η τεχνολογία η οποία παρέχει τη δυνατότητα στους εξυπηρετητές να προωθούν κοινοποιήσεις (notifications) στον περιηγητή του πελάτη σε μορφή DOM Event [31].

Η ιδέα πίσω από τα SSEs είναι ότι ο πελάτης μέσω μιας διαδικτυακής υπηρεσίας "*εγγράφεται*" (subscribe) σε ένα "*ρεύμα ενημερώσεων*" (stream of updates) που παράγεται από τον εξυπηρετητή και κάθε φορά που συμβαίνει ένα νέο "*γεγονός*" (event), ενημερώνεται ο περιηγητής του[32].

EventSource API

Τα SSEs είναι πρότυπο της HTML5 και περιγράφεται πλήρως από τη κοινότητα W3C[45]. Μπορούμε να εκμεταλλευτούμε το πρότυπο αυτό μέσω JavaScript χρησιμοποιώντας το EventSource API όπως φαίνεται παρακάτω:

Πίνακας 8: Εγγραφή πελάτη σε ρεύμα ενημερώσεων

```
if (window.EventSource) {
  var source = new EventSource('URL_TO_SERVER');
}
else {
  alert("your browser does not support Server Side Events");
}
```

Πίνακας 9: Χειρισμός ενημερώσεων (events)

```
source.onopen = function () {
                    console.log("new stream opened");
                };

source.onmessage =  function (event) {
                        console.log("new event: " + event.data);
                        //κώδικας για επεξεργασία του event
                    };

source.onerror = function () {
                    console.log("error");
                };
```

Ο εξυπηρετητής μπορεί να υλοποιηθεί σε PHP, NODE JS ή να είναι κάποιο Servlet. Στο παράρτημα υπάρχει παράδειγμα υλοποίησης εξυπηρετητή σε Servlet χρησιμοποιώντας το Jetty API. Ο εξυπηρετητής αυτός, χρησιμοποιήθηκε για "testing" της διαδικτυακής υπηρεσίας που υλοποιήθηκε.

Η αποστολή event από τον περιηγητή γίνεται με επικεφαλίδα Content-Type “*text/event-stream*” στη απάντηση. Στην βασική του μορφή, το σώμα της απάντησης πρέπει να περιέχει το string "`data:`", ακολουθούμενο από το μήνυμά, ακολουθούμενο από “`\n\n`”.

Πίνακας 10: Παράδειγμα αποστολής event που περιέχει JSON μήνυμα

```
data: {\n
data: "msg": "hello world",\n
data: "id": 12345\n
data: }\n\n
```

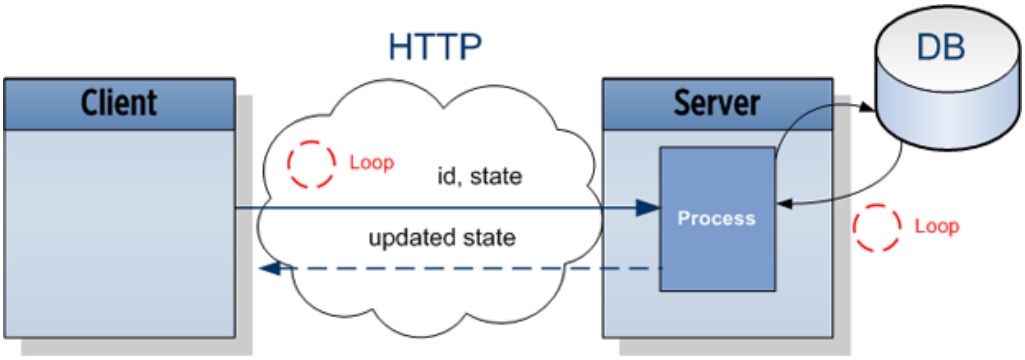
Πίνακας 11: Επεξεργασία JSON μηνύματος στη πλευρά του πελάτη

```
source.onmessage =  function (event) {
                        var data = JSON.parse(event.data);
                        console.log(data.id, data.msg);
                    };
```

Server-Sent Events vs WebSockets vs Long-polling

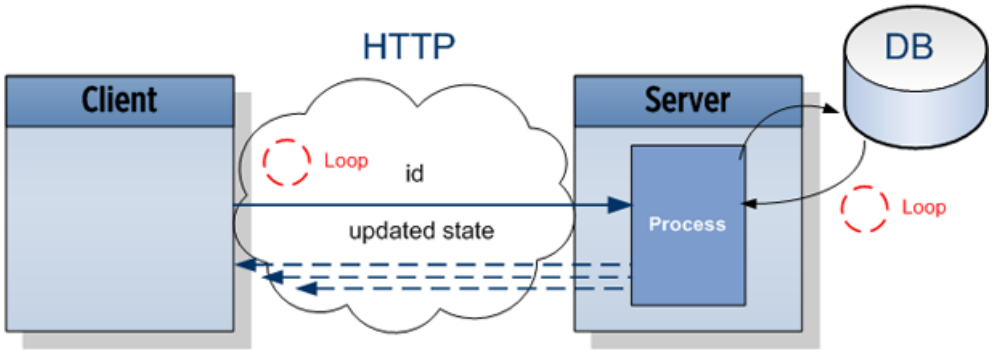
Η τεχνική **Polling** είναι μια παραδοσιακή τεχνική η οποία χρησιμοποιείται από AJAX εφαρμογές. Η βασική ιδέα είναι ότι η εφαρμογή καλεί επανειλημμένα τον εξυπηρετητή για δεδομένα. Σύμφωνα με το πρωτόκολλο HTTP όταν ο πελάτης κάνει μια αίτηση στο εξυπηρετητή, αυτός από τη μεριά του πρέπει να στείλει πίσω μια απάντηση. Στη περίπτωση που δεν υπάρχουν νέα δεδομένα τότε αποστέλλεται πίσω μια άδεια απάντηση. Το μειονέκτημα αυτής της τεχνικής είναι ότι στη περίπτωση που δεν υπάρχουν νέα δεδομένα έχουμε επιβάρυνση του δικτύου με τα ζεύγη HTTP Αίτησης/Απάντησης.

Για τη επίλυση της επιβάρυνσης αυτής υπάρχει η τεχνική του **Long Polling** όπου αν στη πλευρά του εξυπηρετητή δεν υπάρχουν νέα δεδομένα, τότε ο εξυπηρετητής κρατάει το αίτημα που του έχει αποστείλει ο πελάτης ανοικτό μέχρι νέα δεδομένα να είναι διαθέσιμα. Η τεχνική αυτή ονομάζεται και "*Hanging GET*" γιατί όταν δεδομένα γίνουν διαθέσιμα τότε ο εξυπηρετητής απαντάει και κλείνει τη σύνδεση και η διαδικασία επαναλαμβάνεται. Το μειονέκτημα της τεχνικής αυτής είναι ότι μπορεί εύκολα να γίνει χακεριά προσαρτώντας σκριπτάκια σε ένα άπειρο iframe (infinite iframe).



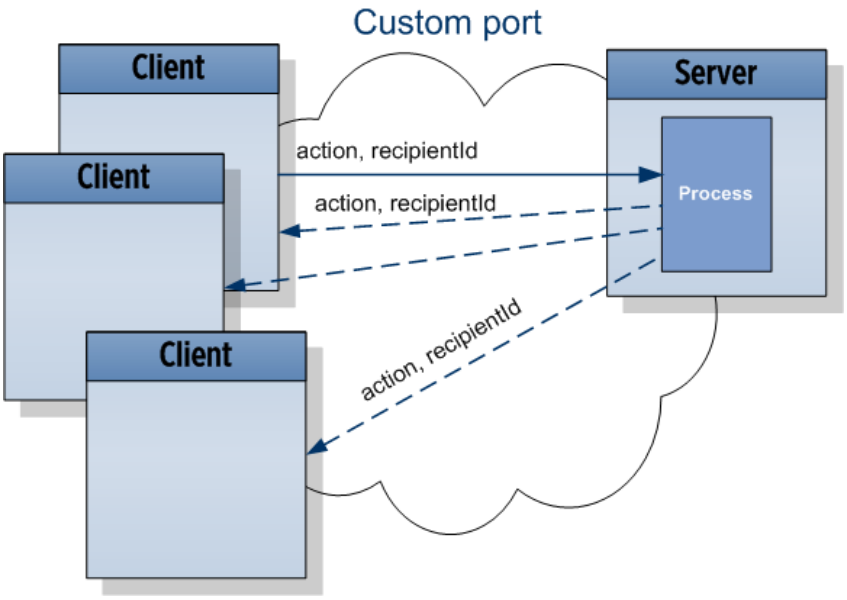
Εικόνα 11: Long Polling

Τα **SSEs** έχουν σχεδιαστεί από τη αρχή με τέτοιο τρόπο ώστε να είναι αποδοτικά. Στη επικοινωνία με SSEs ο εξυπηρετητής όταν έχει νέα δεδομένα, τα προωθεί στο περιηγητή του πελάτη χωρίς να χρειάζονται HTTP Αιτήματα μετά τη εγκατάσταση της σύνδεσης. Έτσι με SSEs δημιουργείται ένα κανάλι μόνο προς μια κατεύθυνση, από τον εξυπηρετητή προς τον πελάτη. Τα SSEs σε αντίθεση με τις polling τεχνικές τα χειρίζεται απευθείας ο περιηγητής και ο πελάτης το μόνο που έχει να κάνει είναι να "*ακούει*" για νέα μηνύματα.



Εικόνα 12: SSEs

Πολύ όμοια με τα SSEs είναι τα WebSockets. Τα WebSocket APIs μας παρέχουν ένα πλουσιότερο πρωτόκολλο και μπορούμε να έχουμε κανάλι επικοινωνίας και προς τις δυο κατευθύνσεις. Το να υπάρχει κανάλι και προς τις δυο κατευθύνσεις είναι χρήσιμο σε διαδικτυακές υπηρεσίες όπως παιχνίδια, εφαρμογές ανταλλαγής μηνυμάτων και εφαρμογές με σχεδόν πραγματικό χρόνο ενημέρωσης (near real-time apps).



Εικόνα 13: WebSockets

Όμως τις περισσότερες φορές δεν χρειαζόμαστε επικοινωνία και προς τις δυο κατευθύνσεις αλλά μόνο ενημερώσεις από τον εξυπηρετητή. Τέτοια παραδείγματα είναι διαδικτυακές υπηρεσίες με ενημέρωση friend status, ενημέρωση τιμών μετοχών χρηματιστηρίου, news feeds κ.α.

Τα βασικότερα πλεονεκτήματα των SSEs (και του EventSource API) έναντι των WebSockets είναι ότι τα SSEs μπορούν να σταλούν μέσω HTTP. Αυτό σημαίνει ότι δεν χρειάζεται κάποιο ειδικό πρωτόκολλο ή κάποια ειδική υλοποίηση εξυπηρετητή για να δουλέψουν. Τα WebSockets απ' την άλλη χρειάζονται full-duplex συνδέσεις και Web Socket εξυπηρετητές για να χειριστούν το πρωτόκολλο. Ακόμη τα SSEs σε αντίθεση με τα WebSockets παρέχουν στον σχεδιασμό τους χαρακτηριστικά όπως αυτόματη επανασύνδεση, event IDs και τη δυνατότητα να στέλνουν αυθαίρετα events[32][33].

CDMI (Cloud Data Management Interface)

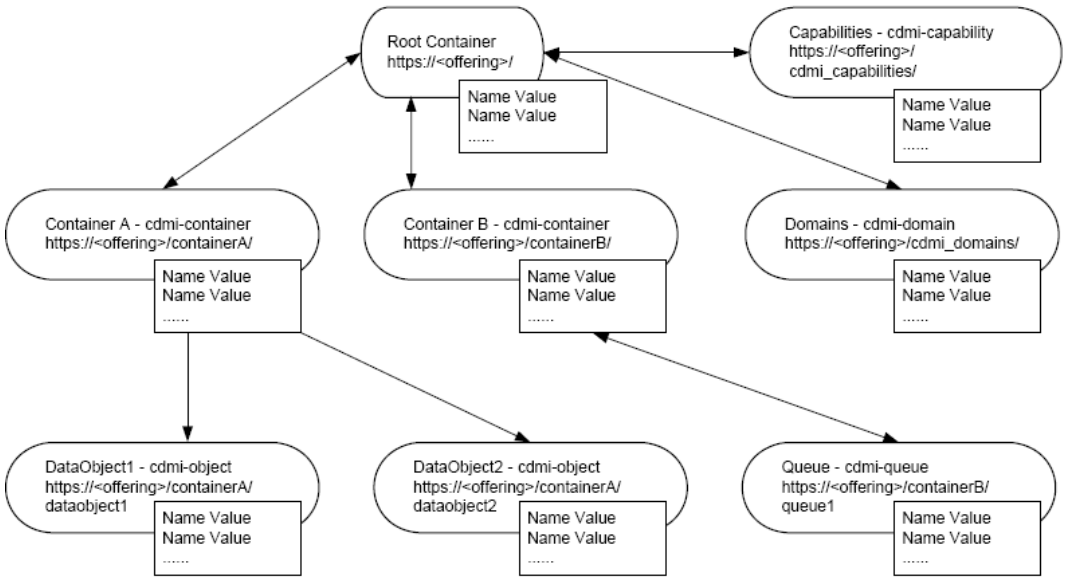
Είναι ένα διεθνές ανοικτό πρότυπο, από τη SNIA (Storage Networking Industry Association)[34] που καθορίζει το πρωτόκολλο για τη πρόσβαση σε αποθηκευτικό νέφος και τη διαχείριση των δεδομένων που είναι αποθηκευμένα σε αυτό[35].

Στο CDMI καθορίζονται "*RESTful*" HTTP λειτουργίες για πρόσβαση στις δυνατότητες (capabilities) του αποθηκευτικού νέφους, πρόσβαση και CRUD λειτουργίες σε containers και αντικείμενα, διαχείριση χρηστών και ομάδων, υλοποίηση εφαρμογών ελέγχου, σύνδεση δεδομένων με μεταδεδομένα, εκτέλεση ερωτημάτων (queries), μεταφορά δεδομένων από ένα νέφος σε άλλο και εξαγωγή δεδομένων μέσω πρωτοκόλλων όπως το iSCSI και NFS.

Στο CDMI χρησιμοποιείται ο ίδιος μηχανισμός εξουσιοδότησης (authorization mechanism) με το NFS σύστημα αρχείων. Η κατάτμηση (partition) του αποθηκευτικού χώρου γίνεται σε μονάδες οι οποίες ονομάζονται containers. Ένα σύνολο από δεδομένα αποθηκεύεται σε container και με βάση αυτό μπορούμε να αναζητήσουμε τα δεδομένα μας[1].

Το Object Model του CDMI

Τα αντικείμενα είναι παρόμοια με τα αρχεία σε ένα παραδοσιακό σύστημα αρχείων, αλλά με τη διαφορά ότι υπάρχει η δυνατότητα εμπλουτισμού τους με μεταδεδομένα.



Εικόνα 14: Το Object Model του CDMI

Ο πελάτης μπορεί να έχει πρόσβαση στα αντικείμενα μέσω του ονόματός του ή του μοναδικού αναγνωριστικού Object ID (OID) του. Όταν η πρόσβαση γίνεται μέσω του ονόματος του αντικειμένου, τότε ο πελάτης δίνει το URL που περιέχει το πλήρες μονοπάτι για τη δημιουργία, διάβασμα, ενημέρωση ή διαγραφή του αντικειμένου. Όταν η πρόσβαση γίνεται με το OID του αντικειμένου, τότε η πρόσβαση γίνεται μέσω του χώρου ονομάτων *cdmi-objectid container*.

Πίνακας 12: Παράδειγμα URL πρόσβασης αντικειμένου μέσω OID και ονόματος

```
http://cloud.example.com/cdmi_objectid/00006FFD0010AA33D8CEF9711E0835CA/
http://cloud.example.com/MyContainer/MyDataObject.txt
```

Αυτό που ενδιαφέρει τη μεριά του πελάτη είναι τα containers και τα αντικείμενα. Η δημιουργία ενός container γίνεται μέσω ενός HTTP PUT αιτήματος δίνοντας στο URI το επιθυμητό όνομα και προαιρετικά αν θέλουμε μπορούμε να συμπεριλάβουμε μεταδεδομένα για αυτό. Μετά μπορούμε να δημιουργήσουμε, και πάλι με HTTP PUT αίτημα, αντικείμενα στο container και τέλος μπορούμε να έχουμε πρόσβαση σε αυτά με ένα HTTP GET αίτημα.

Ουρές Αντικειμένων (Object Queues)

Οι ουρές αντικειμένων παρέχουν First-In First-Out (FIFO) πρόσβαση για τη αποθήκευση και ανάκτηση δεδομένων. Με ένα *queue object writer* προσθέτουμε (POST) δεδομένα στη ουρά και με ένα *queue object reader* διαβάζουμε (GET) δεδομένα από τη ουρά, διαγράφοντας παράλληλα τα δεδομένα αυτά από τη ουρά, ως αναγνώριση της παραλαβής τους από τον πελάτη. Ο μηχανισμός διαχείρισης ουρών επιτρέπει σε ένα *queue object writer* να στέλνει δεδομένα σε ένα ή περισσότερους *queue object reader* με αξιόπιστο τρόπο[35].

Η πρόσβαση σε ουρά γίνεται όπως σε ένα αντικείμενο ή container χρησιμοποιώντας είτε το όνομα της (URI) είτε με το OID όπως φαίνεται παρακάτω:

- Με βάση το URI πχ. `http://cloud.example.com/queueobject`
- Με βάση το OID πχ.
  `http://cloud.example.com/cdmi_objectid/0000706D0010B84FAD185C425D8B537E`

Μια ουρά μπορεί να έχει για πρόγονο ένα άλλο αντικείμενο. Στη περίπτωση αυτή η ουρά κληρονομεί τα μεταδεδομένα του προγόνου της. Στη παρακάτω περίπτωση η ουρά `receipt.queue` κληρονομεί τα μεταδεδομένα του container finance:

`http://cloud.example.com/finance/receipts.queue`

Χρησιμοποιούμε "*RESTful*" HTTP αιτήματα για τη κλήση των επιτρεπτών λειτουργιών που μπορούμε να κάνουμε σε ουρά. Τέτοιες λειτουργίες είναι:

- Η δημιουργία ουράς με HTTP PUT αίτημα
- Η ανάκτηση της ουράς ή μέρος της (χρησιμοποιώντας παραμέτρους στο αίτημα) με HTTP GET αίτημα
- Η προσθήκη δεδομένων στη ουρά με HTTP POST αίτημα
- Η ενημέρωση μέρους των δεδομένων ή μεταδεδομένων της ουράς με HTTP PUT αίτημα
- Η διαγραφή ουράς ή μέρος των δεδομένων της (χρησιμοποιώντας παραμέτρους στο αίτημα) με HTTP DELETE αίτημα

Στο δεύτερο μέρος θα γίνει αναλυτική παρουσίαση της αποστολής αιτημάτων και του χειρισμού των απαντήσεων που επιστρέφει το Vision cloud.

Πίνακας 13: Παράδειγμα αίτησης/απάντησης Δημιουργία ουρας

Πίνακας 13a: Παράδειγμα αίτησης Δημιουργία ουρας

```
PUT /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-queue
Content-Type: application/cdmi-queue
X-CDMI-Specification-Version: 1.0.1
{
  "metadata" : {
  }
}
```

Πίνακας 13b: Παράδειγμα απάντησης Δημιουργία ουράς

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-queue
X-CDMI-Specification-Version: 1.0.1
{
  "objectType" : "application/cdmi-queue",
  "objectID" : "00007E7F00104BE66AB53A9572F9F51E",
  "objectName" : "MyQueue",
  "parentURI " : "/MyContainer/",
  "parentID" : "0000706D0010B84FAD185C425D8B537E",
  "domainURI" : "/cdmi_domains/MyDomain/",
  "capabilitiesURI" : "/cdmi_capabilities/queue/",
  "completionStatus" : "Complete",
  "metadata" : {},
  "queueValues" : ""
}
```

Πίνακας 14: Παράδειγμα αίτησης/απάντησης Διάβασμα ουρας

Πίνακας 14a: Παράδειγμα αίτησης Διάβασμα ουράς

```
GET /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-queue
X-CDMI-Specification-Version: 1.0.1
```

Πίνακας 14b: Παράδειγμα απάντησης Ανάκτησης ουρας

```
HTTP/1.1 200 OK
Content-Type: application/cdmi-queue
X-CDMI-Specification-Version: 1.0.1
{
  "objectType": "application/cdmi-queue",
  "objectID": "00007E7F00104BE66AB53A9572F9F51E",
  "objectName": "MyQueue",
  "parentURI": "/MyContainer/",
  "parentID" : "0000706D0010B84FAD185C425D8B537E",
  "domainURI": "/cdmi_domains/MyDomain/",
  "capabilitiesURI": "/cdmi_capabilities/queue/",
  "completionStatus": "Complete",
  "metadata": {},
  "queueValues": "1-2",
  "mimetype": [
    "text/plain"
  ],
  "valuerange": [
    "0-19"
  ],
  "valuetransferencoding": [
    "utf-8"
  ],
  "value": [
    "First Enqueued Value"
  ]
}
```

Πίνακας 15: Παράδειγμα αίτησης/απάντησης Προσθήκη αντικειμένου στη ουρά

Πίνακας 15a: Παράδειγμα αίτησης Προσθήκη αντικειμένου στη ουρά

```
POST /MyContainer/MyQueue HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-queue
X-CDMI-Specification-Version: 1.0.1
{
  "mimetype" : [
    "text/plain"
  ],
  "value" : [
     "Value to Enqueue"
  ]
}
```

Πίνακας 15b: Παράδειγμα απάντησης Προσθήκη αντικειμένου στη ουρά

```
HTTP/1.1 204 No Content
```

Vision Cloud

Το Vision Cloud είναι ένα ευρωπαϊκό πρόγραμμα το οποίο ξεκίνησε τον Οκτώβριο του 2010, αναμένεται να ολοκληρωθεί τον Σεπτέμβριο του 2013 και το κόστος εκτιμάται στα 15.7 εκατομμύρια ευρώ[37].

Οι εταιρίες, φορείς και πανεπιστημιακά ιδρύματα που συμμετέχουν στο έργο είναι: IBM Haifa Research Lab (Ισραήλ), SAP Research (Γερμανία), Telefónica Investigación y Desarrollo (Ισπανία), Siemens AG (Γερμανία), Engineering Ingegneria Informatica Spa (Ιταλία), Deutsche Welle (Γερμανία), RAI-Radiotelevisione Italiana Spa (Ιταλία), Umeå University (Σουηδία), SNIA Europe (Ηνωμένο Βασίλειο), Telenor (Νορβηγία), France Telecom (Γαλλία), Swedish Institute of Computer Science (Σουηδία), University of Messina (Ιταλία), iTricity B.V. (Ολλανδία) και φυσικά ο Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής του Εθνικού Μετσόβιου Πολυτεχνείου[38].

Στόχος του Vision project είναι να προωθήσει την ανταγωνιστικότητα της οικονομίας της ΕΕ με τη θέσπιση ισχυρών πληροφοριακών και επικοινωνιακών υποδομών τεχνολογίας (*ICT Infrastructure*) για την αξιόπιστη και αποτελεσματική παροχή υπηρεσιών αποθήκευσης δεδομένων. Η υποδομή αυτή θα υποστηρίζει την εγκατάσταση και ανάπτυξη υπηρεσιών αποθήκευσης δεδομένων *on demand*, με ανταγωνιστικό κόστος, παρέχοντας ποιοτικές (QoS) και υψηλής ασφαλείας υπηρεσίες[39].

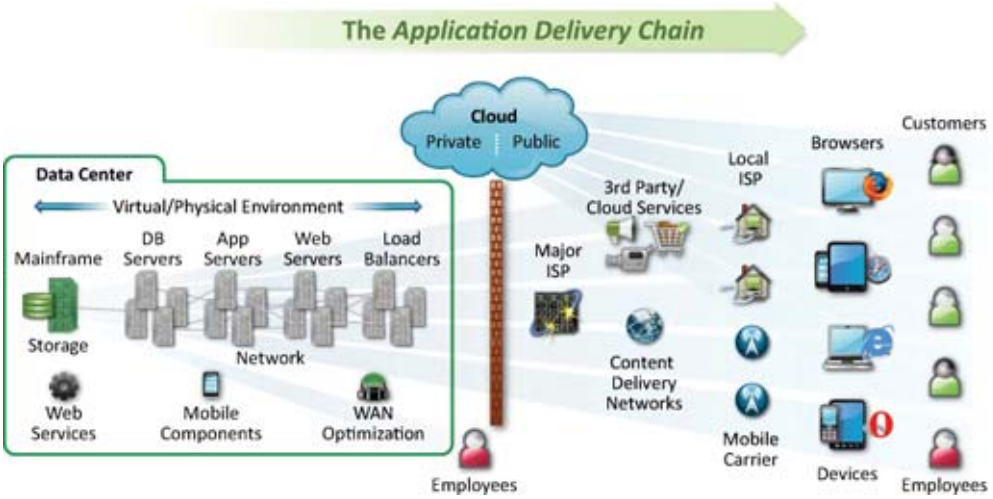
Παραδοτέο του Vision project θα είναι η αρχιτεκτονική και οδηγός αναφοράς υποδομής νέφους (cloud infrastructure) βασισμένη σε ανοικτά πρότυπα και νέες τεχνολογίες, ικανές να βελτιώσουν τη μεταφορά και αποθήκευση δεδομένων. Η υποδομή αυτή, όπου το *virtualization* έχει τον κύριο ρόλο, θα αναδείξει τα ακόλουθα σημεία:

- Νέο μοντέλο αποθήκευσης δεδομένων, χρησιμοποιώντας αντικείμενα με μεταδεδομένα για την υποστήριξη αποτελεσματικής πρόσβασης και διαχείρισης, επιτρέποντας τη κλιμάκωση και την απλοποίηση των υπηρεσιών αποθήκευσης δεδομένων.
- Τη μεταφορά δεδομένων, επιτρέποντας ολοκληρωμένη μετάδοση των δεδομένων και τη πρόσβαση από απομακρυσμένες τοποθεσίες, καθώς και την εξάλειψη των σημερινών περιορισμών.
- Υπολογιστική αποθήκευση (Computational Storage), εισάγοντας τη τεχνολογία για τον καθορισμό και την εκτέλεση υπολογισμών κοντά στην αποθήκευση.

- Αποθήκευση με βάση το περιεχόμενο, παρέχοντας έτσι τη διευκόλυνση της πρόσβασης στα δεδομένα με βάση το περιεχόμενο, χωρίς τη ανάγκη γνώσης της λειτουργίας της υποδομής, από τις εφαρμογές και τους χρήστες.
- Αύξηση δυνατοτήτων νεφών αποθήκευσης δεδομένων (storage clouds), υποστηρίζοντας την παράδοση υπηρεσιών αποθήκευσης δεδομένων με ασφάλεια, με τη επιθυμητή ποιότητα (QoS) και με ανταγωνιστικό κόστος.

Cloud Monitoring

Με μια γρήγορη αναζήτηση στο διαδίκτυο θα παρατηρήσει κανείς ότι υπάρχει πάρα πολλή βιβλιογραφία και άρθρα για ανάπτυξη εφαρμογών και υποδομών για υπηρεσίες σε νέφος. Υπάρχει όμως μια τρύπα στη βιβλιογραφία για τη παρακολούθηση της επίδοσης εφαρμογών και υπηρεσιών που τρέχουν σε νέφος.



Εικόνα 15: Η αλυσίδα παράδοσης εφαρμογών σε υποδομή σύννεφου

Μια άποψη για τον λόγο της ελλιπής βιβλιογραφίας είναι ότι η παρακολούθηση εφαρμογών σε νέφος δεν διαφέρει και πολύ από τη παρακολούθηση κλασσικών εφαρμογών. Όμως η αλυσίδα παράδοσης εφαρμογών και υπηρεσιών που τρέχουν σε νέφος είναι πολύ πιο σύνθετη από κλασσικές εφαρμογές που περιορίζονται και προστατεύονται πίσω από το firewall του οργανισμού που ανήκουν και αυτό γιατί η αλυσίδα εκτείνεται από τον περιηγητή του χρήστη στον υπολογιστή ή κινητή συσκευή του, δια μέσω του διαδικτύου ή WAN οργανισμών ή άλλων νεφών και μέσω περίπλοκων υποδομών σε data centers.

Προκλήσεις στη παρακολούθηση νέφους (cloud monitoring)

Η παραπάνω άποψη απέχει πολύ από τη αλήθεια και αυτό γιατί σε εφαρμογές που τρέχουν σε νέφος έχουμε τις παρακάτω προκλήσεις που δεν εμφανίζονται σε κλασσικές εφαρμογές[41][43]:

Εικονικοί πόροι (Virtualized Resourcing): Το virtualization τόσο σε ιδιωτικά όσο και σε δημόσια σύννεφα αποτελεί πρόκληση για τα εργαλεία παρακολούθησης νεφών. Σε απλά συστήματα δεν έχουμε παρά να παρακολουθούμε τη διάθεση και χρήση φυσικών πόρων αλλά σε σύννεφα το virtualization δημιουργεί ένα δυναμικό χώρο

από υπολογιστικούς και αποθηκευτικούς πόρους οι οποίοι πρέπει να παρακολουθούνται και να διαχειρίζονται με εντελώς διαφορετικό τρόπο.

Χρόνοι απόκρισης τελικού χρήστη: Η παρακολούθηση της επίδοσης του πελάτη είναι διαφορετική σε εφαρμογές που τρέχουν σε νέφος. Ο πρώτος λόγος είναι ότι οι εφαρμογές σε νέφος λειτουργούν μέσω δικτύων και ειδικά το διαδίκτυο. Ο δεύτερος λόγος έχει να κάνει με το γεγονός ότι σε SaaS οι τελικοί χρήστες μπορεί να είναι διασκορπισμένοι σε πάρα πολλές τοποθεσίες σε όλο τον κόσμο. Οι δυο παραπάνω λόγοι καθιστούν το καθορισμό του χρόνου απόκρισης δύσκολο καθώς αν για παράδειγμα ένας χρήστης από τη Ελλάδα παρουσιάζει χαμηλή επίδοση δεν γνωρίζουμε άμεσα αν ο λόγος οφείλεται στο δίκτυο, στη τοποθεσία του ή στη ίδια τη εφαρμογή.

Κλιμάκωση και δοκιμές φόρτωσης: Μια διαφορά μεταξύ εφαρμογών που περιορίζονται εντός ενός οργανισμού και εφαρμογών που τρέχουν σε νέφος είναι η κλιμάκωση. Η κλιμάκωση σε εφαρμογές οργανισμών είναι φραγμένη στα όρια του οργανισμού, προβλέψιμη και πάνω από όλα μετρήσιμη. Σε εφαρμογές και υπηρεσίες που τρέχουν σε νέφος όμως η κλιμάκωση δεν μπορεί να έχει φράγμα, ούτε να προβλεφθεί και αυτό γιατί τα όρια είναι θεωρητικά άπειρα. Δεν ξέρουμε από πριν πόσοι χρήστες θα χρησιμοποιήσουν τη εφαρμογή τη ίδια στιγμή, από που στον κόσμο προσπαθούν να έχουν πρόσβαση με αποτέλεσμα να μην μπορούμε να προβλέψουμε και να μετρήσουμε τον χρόνο φόρτωσης της εφαρμογής.

Multi-tenancy: Το multi-tenancy αποτελεί αρχιτεκτονικό χαρακτηριστικό της υποδομής νέφους που όμως εισάγει προκλήσεις στον τρόπο παρακολούθησης της επίδοσης εφαρμογών που τρέχουν σε νέφος. 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 το τμήμα αυτό. Έτσι δημιουργείται η ανάγκη όχι μόνο να παρακολουθείται η εφαρμογή αλλά και συγκεκριμένα η επίδοση που έχει κάθε τμήμα για τον ανάλογο χρήστη του.

Client-side monitoring

Πριν μερικά χρόνια δεν θα μπορούσε κανείς να φανταστεί ότι θα υπήρχε συζήτηση γύρω από *client-side monitoring* και δεν υπήρχε και λόγος. Όλο το monitoring μπορούσε να γίνει από τη πλευρά του εξυπηρετητή, γιατί η επεξεργασία στη πλευρά του πελάτη ήταν περιορισμένη, και ο πελάτης το μόνο που χρειαζόταν να κάνει είναι να ζητήσει τα αποτελέσματα του monitoring.

Τα δεδομένα αυτά όμως έχουν αλλάξει. Τα αποτελέσματα από το monitoring στη πλευρά του εξυπηρετητή μπορεί να λένε μόνο τη μισή αλήθεια. Αυτό συμβαίνει για πλέον έχουμε πιο πλούσιους πελάτες με τον εμπλουτισμό τους με APIs και εργαλεία που μας προσφέρει η HTML5. Είναι πολύ σημαντικό να παρακολουθούμε τη επίδοση και διαχειρισιμότητα της υποδομής ενός νέφους όμως δεν πρέπει να ξεχνάμε ότι ο πιο σημαντικός ρόλος στο νέφος είναι ο τελικός χρήστης.

Η εισαγωγή της HTML5 βοηθάει στο να έχουμε πιο πλούσιους πελάτες με της δυνατότητες επεξεργασίας δεδομένων και εκτέλεσης πολύπλοκων υπολογισμών να έχουν αυξηθεί. Πλέον μπορούμε να πετύχουμε καλύτερη επίδοση γιατί διάφορες περίπλοκες διαδικασίες όπως έλεγχος εγκυρότητας των δεδομένων, ταξινόμηση και φιλτράρισμα μπορούν να γίνουν χωρίς τη μετακίνηση δεδομένων στο δίκτυο μεταξύ του εξυπηρετητή και πελάτη. Άμεση αναπληροφόρηση (feedback) από τη διεπαφή του χρήστη γιατί η ετοιμασία του GUI γίνεται στη πλευρά του πελάτη και ο εξυπηρετητής δεν ασχολείται πλέον με αυτό και τέλος καλύτερη ασφάλεια γιατί μπορούμε να ελέγχουμε ποια δεδομένα μετακινούνται στο δίκτυο. Αυτό όμως δημιουργεί και πιθανά προβλήματα καθώς η πλευρά του εξυπηρετητή έχει άγνοια για τη συμβαίνει στη πλευρά του πελάτη[40][42].

Δυστυχώς, τα εργαλεία που έχουμε στη διάθεση μας για τη παρακολούθηση στη πλευρά του πελάτη είναι περιορισμένα και περιορίζονται κυρίως μόνο στη καταγραφή στατιστικών όπως η καθυστέρηση του δικτύου και χρόνοι φόρτωσης όπως πχ. αντικειμένων του DOM. Για τον λόγο αυτό ένας οργανισμός θα πρέπει να καταφύγει στη ανάπτυξη διαδικτυακών εφαρμογών εξειδικευμένες στις ανάγκες του νέφους για τη παρακολούθηση του τη συμβαίνει στη πλευρά του πελάτη.

Μέρος Β

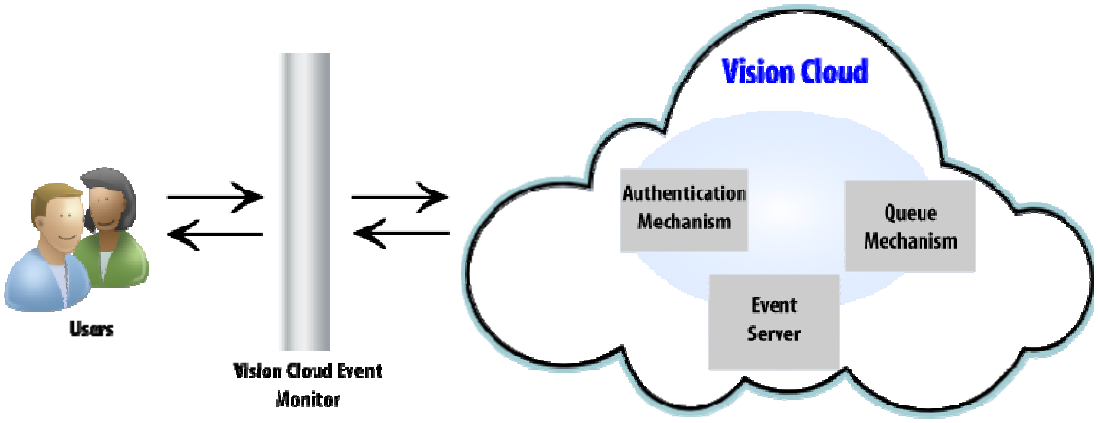
Σκοπός αυτής της διπλωματικής εργασίας ήταν η μελέτη και ανάπτυξη διαδικτυακής υπηρεσίας για τη επίβλεψη αποθηκευτικού νέφους. Στο προηγούμενο μέρος παρουσιάστηκε το θεωρητικό υπόβαθρο που μελετήθηκε, απαραίτητο ώστε να πάρουμε τα γνωστικά εφόδια για τη ανάπτυξη της διαδικτυακής υπηρεσίας.

Στο δεύτερο μέρος αρχικά δίνεται εν συντομία μια περιγραφή της διαδικτυακής υπηρεσίας. Ακολούθως δίνονται οι λειτουργικές και μη λειτουργικές απαιτήσεις που ήταν οδηγοί για τη ανάπτυξη της υπηρεσίας. Ακολούθως παρουσιάζεται ένα ακολουθιακό διάγραμμα (sequence diagram) που δείχνει τη ροή των μηνυμάτων από και προς τη διαδικτυακή υπηρεσία. Έπειτα, περιγράφονται αναλυτικά οι διαφορές υπηρεσίες της εφαρμογής καθώς και τα εργαλεία που αναπτύχθηκαν για να δοκιμαστεί η εφαρμογή αλλά και διάφορες μετρήσεις που έγιναν για να δούμε αν ικανοποιούνται οι απαιτήσεις που θέσαμε. Τέλος δίνονται και κάποιες πιθανές επεκτάσεις της διαδικτυακής υπηρεσίας.

Σύντομη περιγραφή της διαδικτυακής υπηρεσίας

Σκοπός μας είναι η ανάπτυξη μιας διαδικτυακής εφαρμογής για τη επίβλεψη ενός αποθηκευτικού νέφους (storage cloud). Συγκεκριμένα, η εφαρμογή προορίζεται για το Vision Cloud όμως μπορεί να χρησιμοποιηθεί σε οποιοδήποτε νέφος υλοποιεί το πρωτόκολλο CDMI και έχει μηχανισμό παραγωγής events. Η εφαρμογή θα παρέχει την δυνατότητα σε χρήστες του υπολογιστικού νέφους, οι οποίοι μέσω διαδικτύου θα εγγράφονται στη υπηρεσία, να λαμβάνουν πληροφορίες για γεγονότα (events) που συμβαίνουν στο νέφος. Η διαδικτυακή υπηρεσία θα τρέχει στον περιηγητή του χρήστη χωρίς τη μεσολάβηση αλλού λογισμικού. Με τον τρόπο αυτό μεταφέρουμε την επίβλεψη του αποθηκευτικού νέφους στη πλευρά του πελάτη, αποφορτίζοντας το νέφος από τη εργασία αυτή. Η υπηρεσία θα υλοποιεί το πρωτόκολλο CDMI (Cloud Data Management Interface) και ειδικότερα τις Ουρές Ειδοποίησης (Notification Queues), ενώ η ενημέρωση του χρήστη θα γίνεται με χρήση HTML5 EventSource.

Η διαδικτυακή υπηρεσία θα πρέπει να είναι απλή στη χρήση, με όλες τις εργασίες να γίνονται στο παρασκήνιο, αποκρύπτοντας τον τρόπο με τον οποίο γίνονται οι λειτουργίες της εφαρμογής από τον χρήστη. Ο χρήστης θα περιορίζεται μόνο στο να δώσει τα στοιχεία του, όνομα για τη δημιουργία ουράς και κατά επιθυμία του, κλείσιμο της ροής γεγονότων προς τον περιηγητή του.



Εικόνα 16: Vision Cloud Event Monitor

Απαιτήσεις

Για να θεωρηθεί πλήρης η διαδικτυακή εφαρμογή θα πρέπει να ικανοποιούνται οι λειτουργικές και μη λειτουργικές απαιτήσεις που περιγράφονται πιο κάτω:

Λειτουργικές Απαιτήσεις

Το πρώτο που θέλουμε να κάνει η διαδικτυακή υπηρεσία είναι να επαληθεύει ότι ο χρήστης που επιθυμεί να χρησιμοποιήσει τη υπηρεσία, είναι και χρήστης του Vision Cloud. Επομένως θέλουμε ένα μηχανισμό επαλήθευσης των στοιχείων που δίνει ο χρήστης μέσω του Vision Cloud.

Η δεύτερη απαίτηση που έχουμε είναι να υπάρχει κάποιος μηχανισμός διατήρησης της συνεδρίας στη πλευρά του πελάτη για να μην παραβιάζεται η αρχή της αρχιτεκτονικής REST για *stateless server-side*. Ο μηχανισμός αυτός σε πρώτο στάδιο είναι αναγκαίος για να μην μπορεί να έχει πρόσβαση κάποιος στις σελίδες της υπηρεσίας αν δεν έχουν επαληθευτεί τα στοιχεία του και κατά δεύτερο να αποθηκεύονται τα στοιχεία του χρήστη για να έχουν πρόσβαση σε αυτά οι σελίδες της διαδικτυακής υπηρεσίας.

Η τρίτη απαίτηση που έχουμε είναι να υπάρχει κάποιος μηχανισμός για επικοινωνία με το νέφος έτσι ώστε να αιτείται η διαδικτυακή υπηρεσία, κατά επιθυμία του χρήστη, την δημιουργία ουράς. Ο μηχανισμός αυτός πρέπει να είναι σε θέση να διαχειρίζεται όλες τις πιθανές απαντήσεις που μπορεί να επιστρέψει πίσω το νέφος και αναφέρονται στο πρωτόκολλο CDMI[35].

Η επόμενη απαίτηση που έχουμε είναι η διαδικτυακή υπηρεσία να μπορεί να λαμβάνει και να επεξεργάζεται SSEs (*Server-Side-Events*). Σε πρώτο στάδιο θα πρέπει να στηθεί μηχανισμός λήψης γεγονότων και σε δεύτερο στάδιο να υπάρχει μηχανισμός ο οποίος θα αναλαμβάνει την επεξεργασία των events και την εξαγωγή της χρήσιμης πληροφορίας από αυτά.

Η τελευταία απαίτηση που έχουμε από τη διαδικτυακή υπηρεσία είναι η παρουσίαση των αποτελεσμάτων σε μορφή γραφικών παραστάσεων. Για την καλύτερη ενημέρωση του χρήστη θα πρέπει να υπάρχουν τρεις γραφικές όπως δίνονται παρακάτω:

- Μια κύρια γραφική παράσταση όπου θα παρουσιάζονται τα αντικείμενα που εξάγουμε από τα γεγονότα με τη σειρά που έρχονται. Τα αντικείμενα είναι πολλά και θα πρέπει να γίνεται σελιδοποίηση (paging) έτσι ώστε να

παρουσιάζονται μερικά αντικείμενα και με τη χρήση κουμπιών πλοήγησης να διασχίζουμε τη λίστα αντικειμένων.

- Δυο μικρότερες γραφικές παραστάσεις οι οποίες θα παρουσιάζουν τα top *Reads/Writes* του χρήστη.

Μη Λειτουργικές Απαιτήσεις

Η πρώτη μη λειτουργική απαίτηση αφορά τη ασφάλεια της εφαρμογής και έχει άμεση σχέση με τη πρώτη λειτουργική απαίτηση καθώς θέλουμε μόνο εγγεγραμμένους χρήστες του Vision Cloud να έχουν πρόσβαση στη διαδικτυακή υπηρεσία.

Η δεύτερη απαίτηση έχει να κάνει με τη απόδοση της διαδικτυακής υπηρεσίας. Η υπηρεσία θα πρέπει να μπορεί να διαχειρίζεται τα γεγονότα που λαμβάνει ταχύτατα έτσι ώστε να μπορεί να παρουσιάζει *Real-Time Statistics*.

Ως λογική συνέχεια της δεύτερης απαίτησης, θέλουμε να μπορεί η διαδικτυακή υπηρεσία να λαμβάνει και να διαχειρίζεται άμεσα γεγονότα τα οποία έρχονται με συχνότητα ενός δευτερολέπτου και άνω. Για τη προστασία του χρήστη από πιθανές καθυστερήσεις λόγω έλευσης γεγονότων με μικρότερη του ενός δευτερολέπτου απόσταση μεταξύ τους αλλά και λόγω αύξησης των μεγεθών των πινάκων της εφαρμογής θέλουμε να υπάρχει μηχανισμός προστασίας στη πλευρά του πελάτη. Ο μηχανισμός αυτός θα προστατεύει τον περιηγητή του χρήστη από τη υπερβολική ζήτηση μνήμης από τη υπηρεσία και θα δείχνει πάντα *Real-Time-Statistics*.

Η τρίτη μη λειτουργική απαίτηση που έχουμε είναι εφόσον θέλουμε να φτιάξουμε μια διαδικτυακή υπηρεσία για επίβλεψη νέφους από τη πλευρά του πελάτη, θέλουμε *non-persistence server-side*. Δηλαδή, δεν θα πρέπει να αποθηκεύεται τίποτα στη πλευρά του εξυπηρετητή, ούτε γεγονότα, ούτε στοιχεία του χρήστη έτσι ώστε να είμαστε και συμβατοί με τις αρχές της αρχιτεκτονικής REST.

Όσον αφορά τη πλευρά του πελάτη, τα στατιστικά που μαζεύονται θα αποθηκεύονται στη μνήμη σε πίνακες (javascript) έτσι ώστε να παρουσιάζονται σε γραφικές παραστάσεις. Όταν ο χρήστης κλείσει το tab στον περιηγητή του ή τον ίδιο τον περιηγητή ή πατήσει stop στη έλευση γεγονότων ή refresh (F5) τότε τα στατιστικά αυτά πολύ απλά τα “πετάμε”. Δεν χρειάζεται σε μια υπηρεσία με *Real-Time Statistics* να φυλάμε σε βάση δεδομένων στοιχεία.

Η τέταρτη απαίτηση που θέλουμε να ικανοποιείται αφορά το *tenancy*. Η υπηρεσία που θα φτιαχτεί θα πρέπει να είναι ανεξάρτητη του αριθμού των χρηστών που θα τη χρησιμοποιούν ταυτοχρόνως και ακόμη θα πρέπει να δίδεται στον χρήστη η δυνατότητα της σύνδεσης από διαφορετικούς λογαριασμούς στη υπηρεσία από όσα tab ή περιηγητές θέλει.

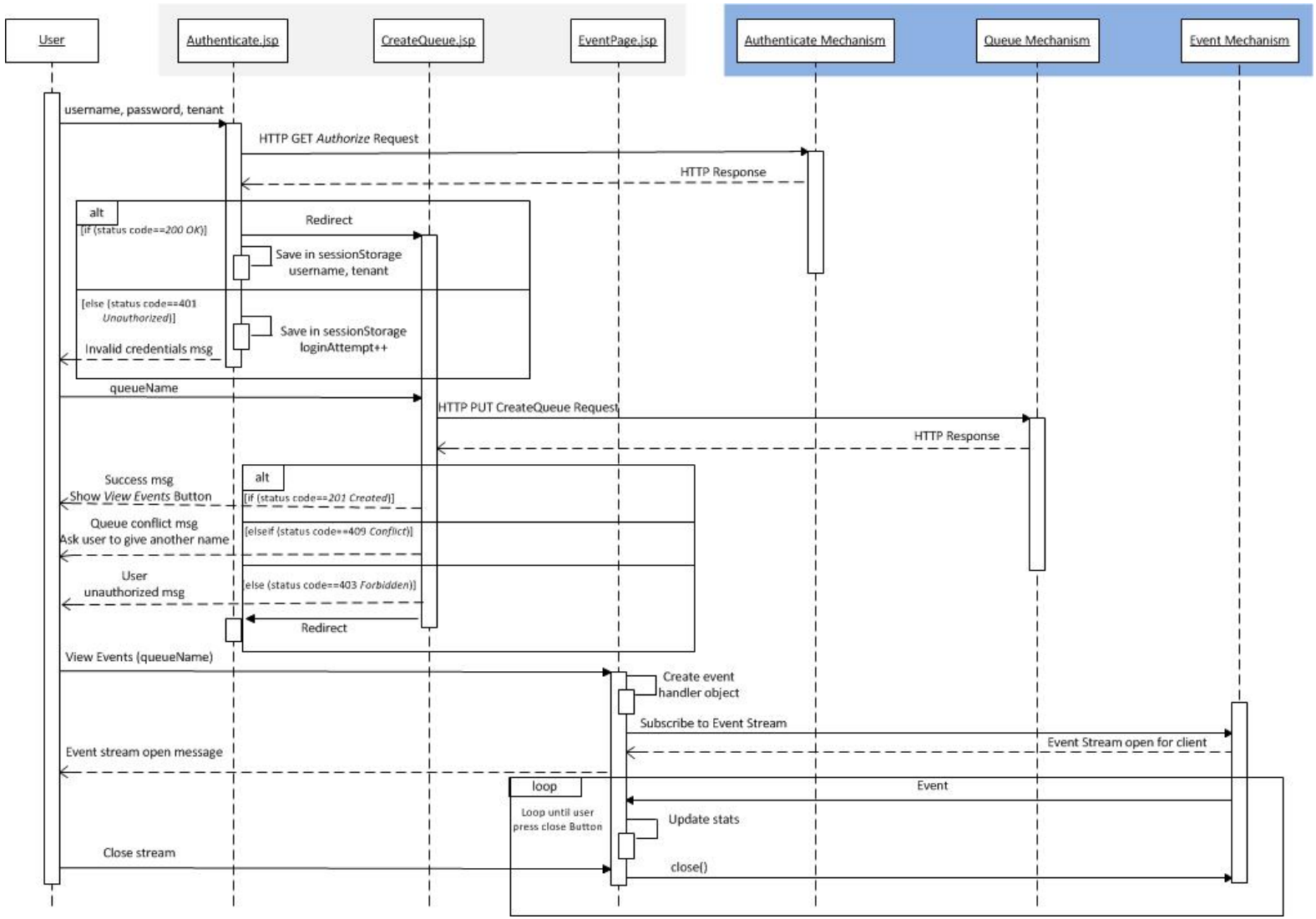
Η πέμπτη απαίτηση που θέλουμε να ικανοποιείται έχει να κάνει με τη φορητότητα της εφαρμογής. Η εφαρμογή θα πρέπει να είναι ανεξάρτητη αρχιτεκτονικής και λειτουργικού συστήματος.

Τέλος, η τελευταία απαίτηση που θέλουμε να ικανοποιείται αφορά το *cross-browser compatibility*. Η υπηρεσία θα πρέπει να δουλεύει σε όλους τους περιηγητές, άρα θα πρέπει να ακολουθηθούν όλα τα διεθνή πρότυπα σύμφωνα με το W3C[44] για τη λήψη Server-Side Events και για τη χρήση HTML5 (EventSource, SessionStorage, canvas κ.α).

Η παραπάνω απαίτηση τη στιγμή που γράφεται το κείμενο αυτό ικανοποιείται εν μέρει καθώς η υπηρεσία δουλεύει σε *Mozilla Firefox* και *Google Chrome* αλλά όχι σε *Microsoft Internet Explorer*. Ο λόγος όμως που δεν δουλεύει σε *Microsoft Internet Explorer* έχει να κάνει με το γεγονός ότι η Microsoft δεν έχει ακόμη υλοποιήσει τη δυνατότητα να λαμβάνει ο περιηγητής *Server-Side Events* (SSEs). Όταν όμως η Microsoft υλοποιήσει αυτή τη δυνατότητα τότε η υπηρεσία θα πρέπει να δουλεύει και στον *Internet Explorer* καθώς ακολουθήθηκαν όλα τα διεθνή πρότυπα για τη ανάπτυξη της εφαρμογής.

Ακολουθιακό διάγραμμα (Sequence diagram)

Στη σελίδα που ακολουθεί παρουσιάζεται το ακολουθιακό διάγραμμα που περιγράφει τη ανταλλαγή μηνυμάτων μεταξύ του χρήστη, της διαδικτυακής υπηρεσίας και του νέφους. Οι οθόνες της υπηρεσίας παρουσιάζονται στο γκρι ορθογώνιο ενώ οι μηχανισμοί που ανήκουν στο νέφος παρουσιάζονται στο γαλάζιο ορθογώνιο.


User
Authenticate.jsp
CreateQueue.jsp
EventPage.jsp
Authenticate Mechanism
Queue Mechanism
Event Mechanism
username, password, tenant
HTTP GET Authorize Request
HTTP Response
alt
[if (status code==200 OK)]
Redirect
Save in sessionStorage
username, tenant
[else [status code==401 Unauthorized]]
Save in sessionStorage
loginAttempt++
Invalid credentials msg
queueName
HTTP PUT CreateQueue Request
HTTP Response
Success msg
Show View Events Button
alt
[if (status code==201 Created)]
Queue conflict msg
Ask user to give another name
[elseif (status code==409 Conflict)]
User
unauthorized msg
[else (status code==403 Forbidden)]
Redirect
View Events (queueName)
Create event
handler object
Subscribe to Event Stream
Event Stream open for client
Event stream open message
loop
Loop until user
press close Button
Event
Update stats
Close stream
close()

Περιγραφή λειτουργιών διαδικτυακής υπηρεσίας

Παρακάτω παρουσιάζονται οι σελίδες της διαδικτυακής υπηρεσίας και αναλύονται οι λειτουργίες που επιτελούνται σε κάθε μια από αυτές.

Σελίδα επαλήθευσης στοιχείων χρήστη (Authenticate.jsp)

Η πρώτη σελίδα που εμφανίζεται στον χρήστη ο οποίος επιθυμεί να χρησιμοποιήσει τη διαδικτυακή υπηρεσία, ζητάει από αυτόν να επαληθεύσει τα στοιχεία του.

Όταν ο χρήστης δώσει τα στοιχεία του και πατήσει *Sign-in*, τότε η διαδικτυακή υπηρεσία φτιάχνει ένα αίτημα σύμφωνα με το REST API του Vision Cloud[46] και το αποστέλλει στο νέφος για έλεγχο των στοιχείων του χρήστη. Έτσι, σε αντίθεση με κλασσικές client-server υπηρεσίες, εδώ δεν χρειαζόμαστε βάση δεδομένων αποκλειστικά για τη εφαρμογή για να φυλάμε τα στοιχεία του χρήστη. Με τον τρόπο αυτό ο χρήστης δεν χρειάζεται να θυμάται επιπλέον στοιχεία για τη υπηρεσία αυτή. Ακόμη, χρήστης ο οποίος δεν έχει ID του Vision Cloud δεν έχει και νόημα να χρησιμοποιήσει τη υπηρεσία καθώς η υπηρεσία αυτή παρέχει τη δυνατότητα επίβλεψης των αντικειμένων που χειριζόμαστε στο νέφος.

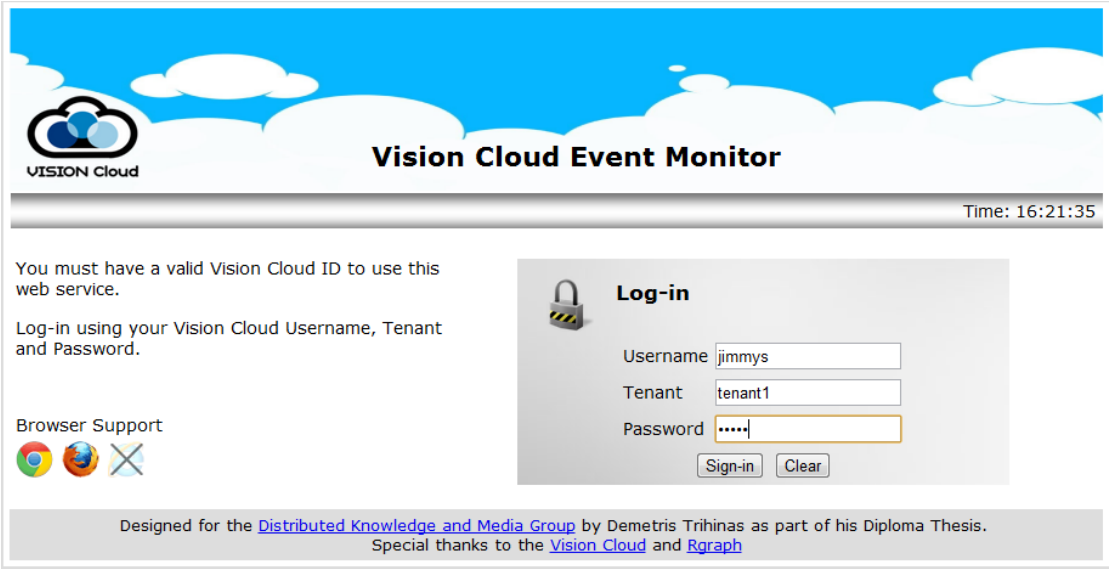
Σύμφωνα με το REST API του Vision Cloud, το αίτημα επαλήθευσης στοιχείων χρήστη θα πρέπει να είναι ένα HTTP GET αίτημα προς τον authorization server του νέφους με επικεφαλίδα Authorization. Η μορφή της επικεφαλίδας Authorization θα πρέπει ξεκινάει με τη λέξη “Basic” ακολουθούμενη από το κωδικοποιημένο σε base64 String που περιέχει το *username*, *tenant* και *password* του χρήστη όπως φαίνεται παρακάτω:

Πίνακας 16: Μορφή αιτήματος επαλήθευσης στοιχείων χρήστη

```
GET <root>/authenticate HTTP/1.1
Authorization: Basic base64Encoded{username@tenantName:password}
```

Οι πιθανές απαντήσεις του νέφους, σύμφωνα και με το REST API του Vision Cloud, στο αίτημα επαλήθευσης στοιχείων χρήστη είναι δύο. Η πρώτη περίπτωση είναι τα στοιχεία που δίνονται να ανήκουν πράγματι σε χρήστη του Vision Cloud, έτσι η επιστρεφόμενη απάντηση θα έχει status code *200 OK* και στο σώμα της απάντησης σε μορφή JSON String θα περιέχονται το username και tenant του χρήστη. Για λόγους ασφαλείας δεν περιέχεται το password. Στη δεύτερη περίπτωση, τα στοιχεία που δίνονται δεν είναι έγκυρα επομένως η επιστρεφόμενη απάντηση θα έχει status code *401 Unauthorized* με κενό σώμα.

Στη *Εικόνα 17* που ακολουθεί φαίνεται ένα παράδειγμα επαλήθευσης έγκυρων στοιχείων χρήστη και στους *Πίνακες 17α-β* το HTTP αίτημα και η επιστρεφόμενη απάντηση:



Εικόνα 17: Screenshot σελίδας επαλήθευσης στοιχείων χρήστη

Πίνακας 17: Αίτημα/Απάντηση επαλήθευσης στοιχείων χρήστη

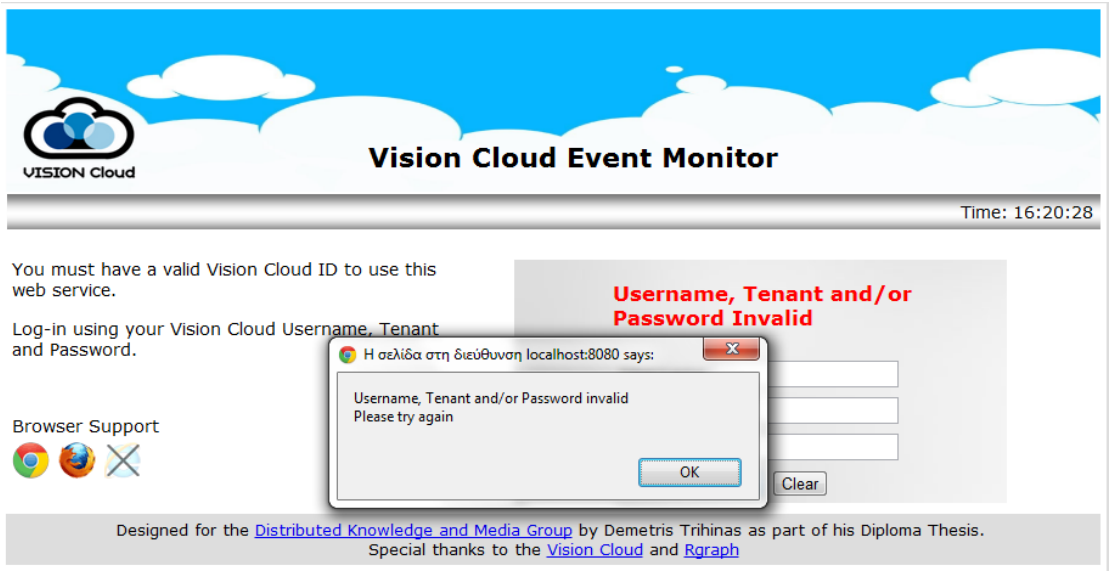
```
GET http://localhost:8080/EventMonitor/authenticate HTTP/1.1
Authorization: Basic amltbXlzQHRlbmFudDE6JDEyMzQ=
```

Πίνακας 17α: Αίτημα επαλήθευσης στοιχείων χρήστη

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  “userName”: jimmys
  “tenantName”: tenant1
}
```

Πίνακας 17β: Απάντηση έγκυρης επαλήθευσης στοιχείων χρήστη

Στη *Εικόνα 18* που ακολουθεί φαίνεται ένα παράδειγμα όπου ο χρήστης δίνει μη έγκυρα στοιχεία και στους *Πίνακες 18α-β* το HTTP αίτημα και η επιστρεφόμενη απάντηση:



Εικόνα 18: Screenshot σελίδας επαλήθευσης στοιχείων χρήστη – λάθος στοιχεία

Πίνακας 18: Αίτημα/Απάντηση επαλήθευσης με λάθος στοιχείων χρήστη

`GET http://localhost:8080/EventMonitor/authenticate HTTP/1.1` `Authorization: Basic ampppbW15c0B0ZW4xOiQxMTEx`
Πίνακας 18α: Αίτημα επαλήθευσης με λάθος στοιχείων χρήστη

`HTTP/1.1 401 Unauthorized`
Πίνακας 18β: Απάντηση μη επαλήθευσης στοιχείων χρήστη

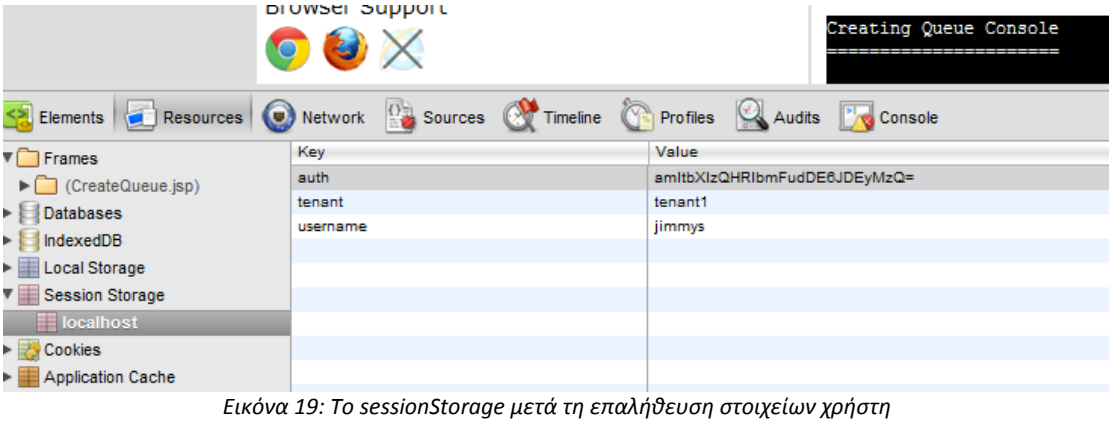
Μια από τις βασικότερες λειτουργικές απαιτήσεις της διαδικτυακής υπηρεσίας, αναφέρεται σε μηχανισμό διατήρησης συνεδρίας του χρήστη. Ο μηχανισμός αυτός όπως αναφέρθηκε και πιο πριν είναι αναγκαίος για δυο λόγους. Ο πρώτος λόγος έχει να κάνει με τη ασφάλεια της υπηρεσίας καθώς δεν επιτρέπεται με τον τρόπο αυτό να παρακαμφθεί η επαλήθευση των στοιχείων του χρήστη για πρόσβαση σε σελίδες της υπηρεσίας. Ο δεύτερος λόγος, είναι ότι ο μηχανισμός αυτός μας βοηθάει στο να αποθηκεύουμε χρήσιμα στοιχεία που αφορούν τον χρήστη έτσι ώστε να έχουν πρόσβαση σε αυτά οι σελίδες της υπηρεσίας χωρίς να τα ζητάμε κάθε φορά από τον χρήστη.

Ο μηχανισμός που χρησιμοποιήθηκε στη διαδικτυακή υπηρεσία είναι το sessionStorage που αποτελεί μια από τις καινοτομίες της HTML5. Με το sessionStorage όπως αναφέρθηκε και στο πρώτο μέρος, μπορούμε να αποθηκεύουμε δεδομένα τοπικά, στη πλευρά του πελάτη. Με τον τρόπο αυτό πετυχαίνουμε να έχουμε stateless Server-Side, μια από τις βασικές αρχές της αρχιτεκτονικής REST, καθώς δεν διατηρούμε τη συνεδρία στη πλευρά του εξυπηρετητή αλλά στέλνουμε μόνο όταν χρειάζεται στοιχεία του χρήστη μέσα στο αίτημα.

Τα δεδομένα που αποθηκεύουμε στο sessionStorage και η χρησιμότητα τους είναι τα ακόλουθα:

- Αποθηκεύουμε μια μεταβλητή "*login_attempt*" η οποία αυξάνεται κάθε φορά που τα στοιχεία του χρήστη δεν είναι ορθά. Με τον τρόπο αυτό μετράμε τις προσπάθειες επαλήθευσης στοιχείων που κάνει ο χρήστης, δίνοντας του μόνο 3 προσπάθειες. Η μεταβλητή αυτή μετά τη επαλήθευση των στοιχείων του χρήστη δεν είναι αναγκαία έτσι τη αφαιρούμε από το sessionStorage.
- Αποθηκεύουμε μια μεταβλητή "*username*" στη οποία αποθηκεύουμε το username του χρήστη μετά τη επαλήθευση των στοιχείων του χρήστη για να χρησιμοποιηθεί στις επόμενες σελίδες της υπηρεσίας.
- Αποθηκεύουμε μια μεταβλητή "*tenant*" στη οποία αποθηκεύουμε το tenant του χρήστη μετά τη επαλήθευση των στοιχείων του χρήστη για να χρησιμοποιηθεί στις επόμενες σελίδες της υπηρεσίας.
- Αποθηκεύουμε μια μεταβλητή "a*uth*" στη οποία αποθηκεύουμε το κωδικοποιημένο authorization string μετά τη επαλήθευση των στοιχείων του χρήστη για να χρησιμοποιηθεί στις επόμενες σελίδες της υπηρεσίας. Ο λόγος που αποθηκεύουμε κωδικοποιημένο το string αυτό είναι για ασφάλεια καθώς δεν θέλουμε να εκθέτουμε το password του χρήστη.

Στη *Εικόνα 3* φαίνεται από τη κονσόλα JavaScript του Google Chrome οι μεταβλητές που βρίσκονται στο sessionStorage μετά τη ορθή επαλήθευση των στοιχείων χρήστη:



Εικόνα 19: Το sessionStorage μετά τη επαλήθευση στοιχείων χρήστη

Σελίδα δημιουργίας ουράς (CreateQueue.jsp)

Η επόμενη σελίδα μετά τη επαλήθευση των στοιχείων του χρήστη είναι η σελίδα δημιουργίας ουράς γεγονότων.

Στη σελίδα αυτή ο χρήστης καλείται να δώσει ένα όνομα για δημιουργία μιας καινούργιας ουράς γεγονότων. Αν η ουρά δημιουργηθεί στο νέφος με επιτυχία τότε παρουσιάζεται στο χρήστη η επιλογή να αρχίσει να παρακολουθεί (monitoring) σε πραγματικό χρόνο τη ουρά.

Όταν ο χρήστης δώσει όνομα για ουρά και πατήσει το κουμπί *Create* τότε αποστέλλεται στο νέφος REST HTTP αίτημα τύπου PUT για δημιουργία ουράς. Η μορφή του αιτήματος καθορίζεται σύμφωνα με το πρωτόκολλο CDMI[35]. Για να είναι σωστή η μορφή του αιτήματος είναι απαραίτητο εκτός από το να αποσταλεί στο σωστό URL, θα πρέπει να προσθέσουμε τις επικεφαλίδες *Accept*, *Content-Type*, *X-CDMI-Specification-Version* και για λόγους ασφαλείας τη επικεφαλίδα Authorization όπως φαίνεται παρακάτω:

Πίνακας 19: Μορφή αιτήματος δημιουργίας ουράς

```
PUT <root URI>/<ContainerName>/<QueueName>

Accept: application/cdmi-queue
Content-Type: application/cdmi-queue
X-CDMI-Specification-Version: <CDMI Version>
Authorization: Basic base64Encoded{username@tenantName:password}
```

Δεν απαιτείται από τον χρήστη να δώσει ξανά το username και password του καθώς έχουμε αποθηκεύσει από τη προηγούμενη σελίδα στο sessionStorage τη μεταβλητή *auth*.

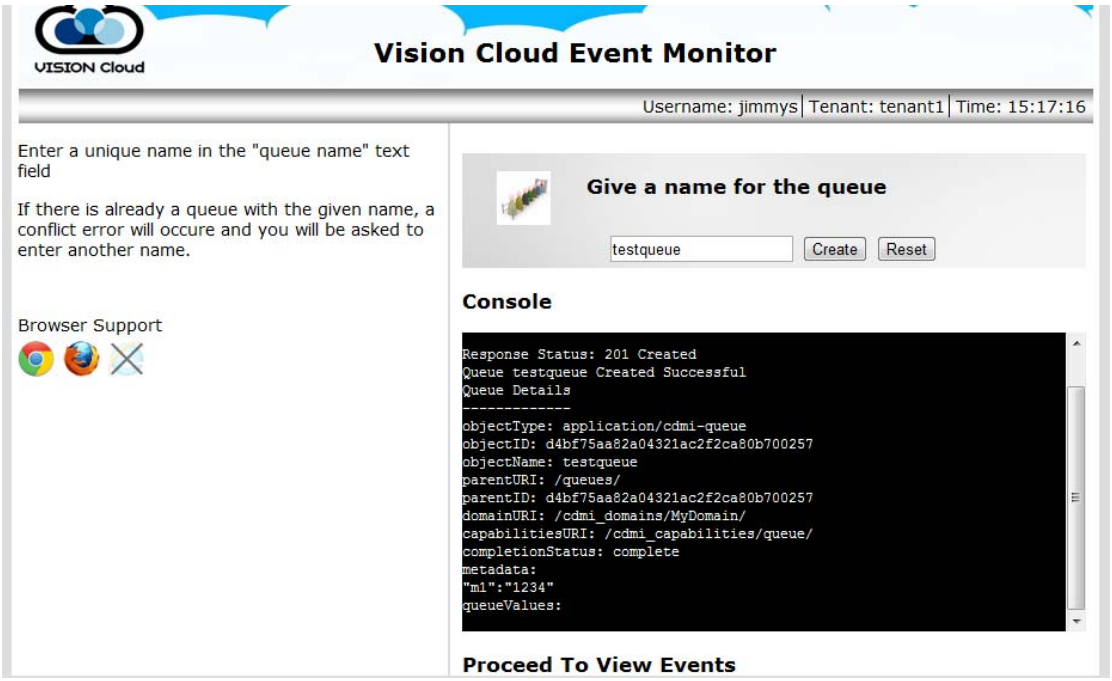
Οι πιθανές απαντήσεις του νέφους, σύμφωνα και με το πρωτόκολλο CDMI, στο αίτημα δημιουργίας ουράς είναι αρκετές και θα πρέπει η διαδικτυακή υπηρεσία να μπορεί να τις διαχειριστεί:

- Αν είναι επιτυχής η δημιουργία της ουράς τότε το νέφος επιστρέφει απάντηση με HTTP status code *201 Created* με τα στοιχεία της νεοδημιουργηθείσας ουράς στο σώμα της απάντησης ως JSON string.
- Αν υπάρχει ήδη ουρά με το όνομα που έχει δώσει ο χρήστης τότε το νέφος απαντάει με HTTP status code *409 Conflict* και θα πρέπει να ζητηθεί από τον χρήστη να δώσει άλλο όνομα.
- Αν ο χρήστης για κάποιο λόγο δεν έχει δικαίωμα δημιουργίας ουράς, σύμφωνα με το νέφος, τότε το νέφος απαντάει με HTTP status code *403 Forbidden* και θα πρέπει να ζητηθεί από τον χρήστη να συνδεθεί με κάποιο άλλο username αν θέλει να δημιουργήσει ουρά.

- Τέλος, παρόλο που γίνεται μέριμνα ώστε το αίτημα να μην πάει σε λάθος URL (HTTP status code *404 Not Found*), να μην είναι λανθασμένου τύπου πχ. τύπου GET (HTTP status code *400 Bad Request*) και να έχει ήδη κάνει Sign-in ο χρήστης (HTTP status code *401 Unauthorized*) η διαδικτυακή υπηρεσία φροντίζει ώστε να μπορεί να διαχειριστεί και τέτοιες απαντήσεις ώστε να είναι απόλυτα συμβατή με το πρωτόκολλο CDMI.

Ακολουθούν μερικά παραδείγματα για να διαφανούν οι λειτουργίες της διαδικτυακής υπηρεσίας κατά τη δημιουργία ουράς αναλόγως απαντήσεων που λαμβάνει από το νέφος.

Στη *Εικόνα 20* φαίνεται ένα παράδειγμα επιτυχής δημιουργίας ουράς με το όνομα "*testqueue*" και στους *Πίνακες 20α-β* το RESTful αίτημα και η απάντηση του νέφους:



Εικόνα 20: Screenshot Επιτυχής δημιουργία ουράς

Πίνακας 20: Αίτημα/Απάντηση δημιουργίας ουράς

```
PUT  http://localhost:8080/EventMonitor/queues/testqueue HTTP/1.1
Host: localhost:8080
Accept: application/cdmi-queue
Content-Type: application/cdmi-queue
X-CDMI-Specification-Version: 1.0.1
Authorization: Basic amltbXlzQHRlbmFudDE6JDEyMzQ=

{
    "metadata" : {
    }
}
```

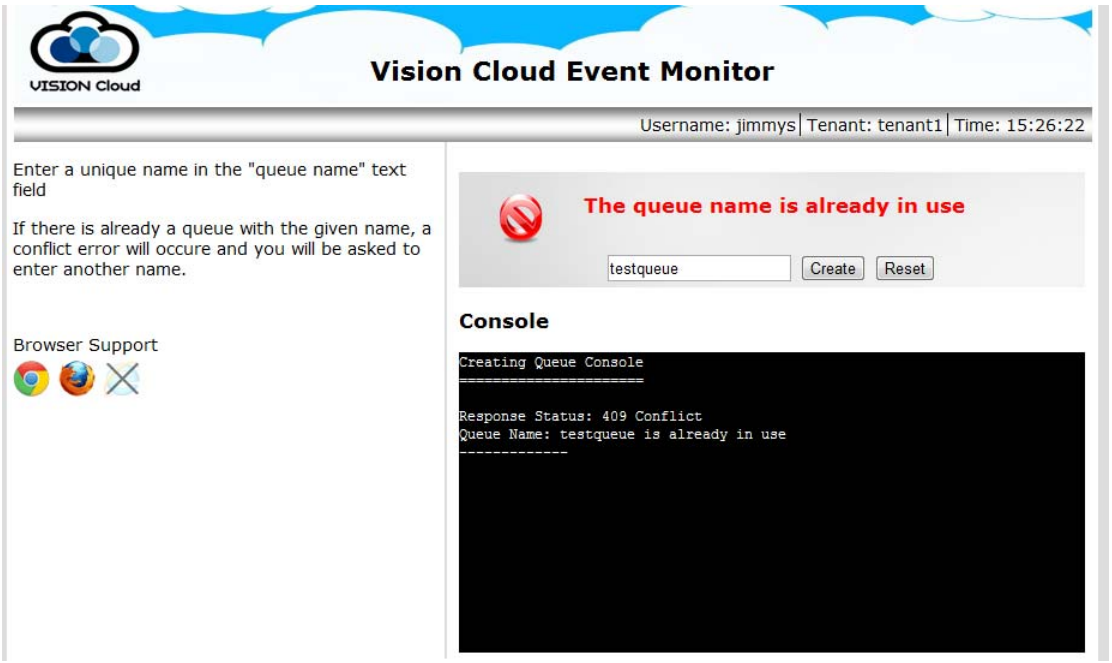
Πίνακας 20α: Αίτημα δημιουργίας ουράς

```
HTTP/1.1 201 Created
Content-Type: application/cdmi-queue
X-CDMI-Specification-Version: 1.0.1

{
    "objectType" : "application/cdmi-queue",
    "objectID" : "6241e1402e46454bb401e0277b82ddd6",
    "objectName" : "testqueue",
    "parentURI " : "/queues/",
    "parentID" : "6241e1402e46454bb401e0277b82ddd6",
    "domainURI" : "/cdmi_domains/MyDomain/",
    "capabilitiesURI" : "/cdmi_capabilities/queue/",
    "completionStatus" : "complete",
    "metadata" : {
        "m1":"1234"
    },
    "queueValues" : ""
}
```

Πίνακας 20β: Απάντηση Επιτυχής δημιουργίας ουράς

Στη *Εικόνα 21* φαίνεται ένα παράδειγμα όπου παρουσιάζεται conflict κατά τη δημιουργία ουράς με το όνομα "*testqueue*" που πλέον ήδη υπάρχει στο νέφος και στον *Πίνακα 21* η απάντηση του νέφους:



Εικόνα 21: Screenshot Conflict στη δημιουργία ουράς

Πίνακας 21: Απάντηση conflict στη δημιουργία ουράς

```
HTTP/1.1 409 Conflict
```

Στη *Εικόνα 6* φαίνεται ένα παράδειγμα όπου ο χρήστης δεν έχει δικαίωμα να δημιουργίας ουράς και του ζητείται να συνδεθεί με άλλο λογαριασμό και στον *Πίνακα 7* η απάντηση του νέφους:

Εικόνα 22: Screenshot Απάντηση forbidden στη δημιουργία ουράς

Πίνακας 22: Απάντηση forbidden στη δημιουργία ουράς

```
HTTP/1.1 403 Forbidden
```

Με τη επιτυχής δημιουργία ουράς εμφανίζεται στον χρήστη η επιλογή να παρακολουθήσει σε πραγματικό χρόνο τη ουρά. Με το που πατάει ο χρήστης "*View Events*" μεταβαίνει στη σελίδα παρακολούθησης γεγονότων (EventPage.jsp) με query parameter το όνομα της ουράς που δημιούργησε ο χρήστης:

Εικόνα 23: Screenshot Κουμπί μετάβασης στη σελίδα παρακολούθησης γεγονότων

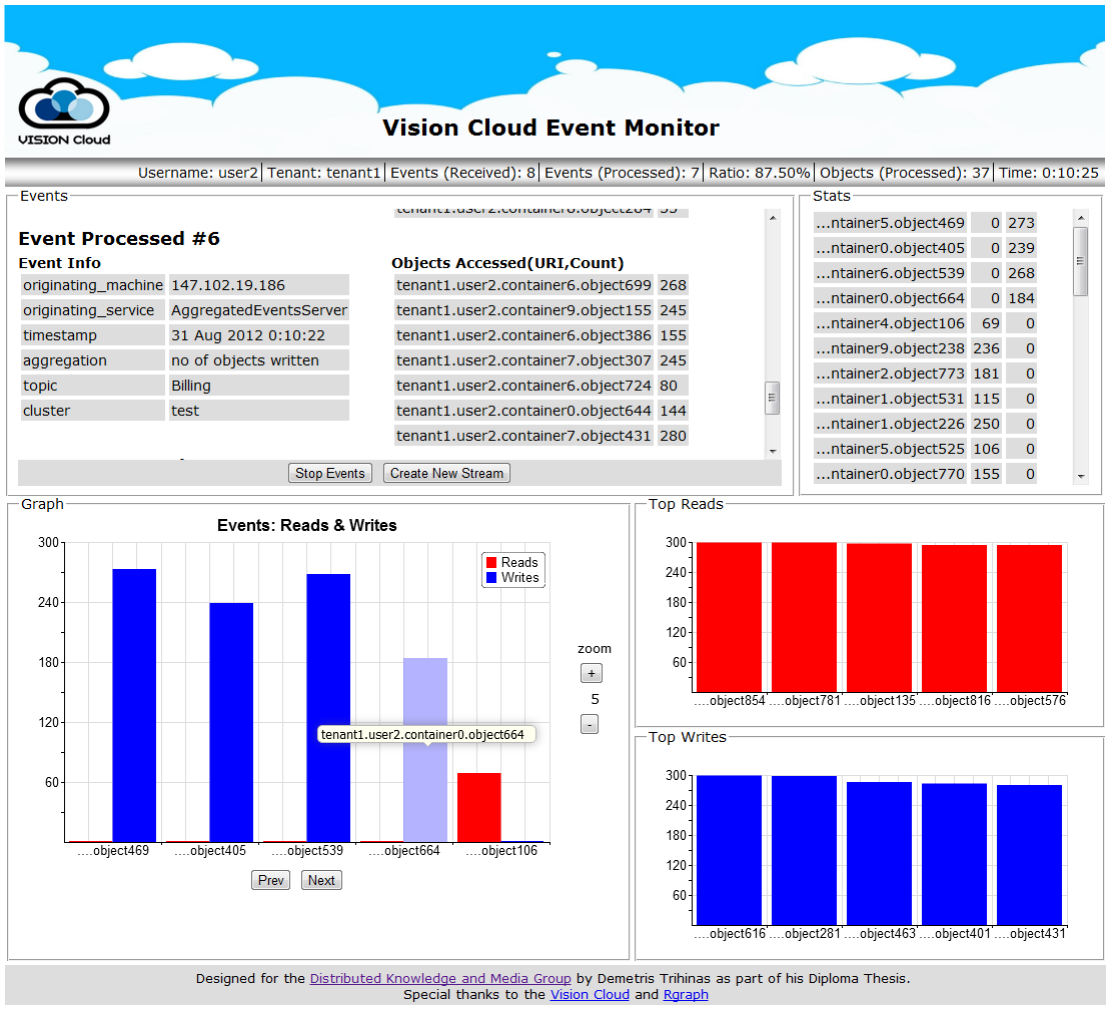
Πίνακας 23: URL προς σελίδα παρακολούθησης γεγονότων

```
<root URI>/EventMonitor/EventPage.jsp?queueName=just_another_queue
```

Σελίδα παρακολούθησης event σε πραγματικό χρόνο (EventPage.jsp)

Η επόμενη σελίδα μετά τη δημιουργία ουράς από τον χρήστη είναι η σελίδα παρακολούθησης γεγονότων.

Στη σελίδα παρακολούθησης γεγονότων ο χρήστης δεν χρειάζεται να κάνει τίποτα. Η διαδικτυακή υπηρεσία αλλά και κατ' επέκταση η σελίδα αυτή είναι σχεδιασμένη έτσι ώστε να γίνονται τα πάντα στο παρασκήνιο, διατηρώντας με τον τρόπο αυτό τη απλότητα στη χρήση.



Εικόνα 24: Screenshot σελίδας παρακολούθησης γεγονότων

Μια από τις βασικότερες λειτουργικές απαιτήσεις της διαδικτυακής εφαρμογής είναι να στηθεί στη πλευρά του πελάτη μηχανισμός λήψης events. Αυτό μπορεί να γίνει χρησιμοποιώντας το EventSource API της HTML5 όπως αναφέρθηκε στο πρώτο μέρος.

Για να λαμβάνουμε Server-Side Events (SSEs) ο περιηγητής του χρήστη μέσω της διαδικτυακής υπηρεσίας θα πρέπει να "*εγγραφεί*" (subscribe) στο "*ρεύμα ενημερώσεων*" (stream of updates) που παράγεται από τον Event Server του νέφους

έτσι ώστε κάθε φορά που συμβαίνει ένα νέο "*γεγονός*" (event) να ενημερώνεται γι' αυτό. Για τον λόγο αυτό με το άνοιγμα της σελίδας αυτής φτιάχνουμε ένα αντικείμενο τύπου *EventSource* ως *event handler* δίνοντας του το URL που βρίσκεται ο Event Server. Ο *event handler* στέλνει αίτημα προς τον Event Server του νέφους με παράμετρο το όνομα της ουράς ώστε να μας προσθέσει ως παραλήπτες events, όπως φαίνεται παρακάτω:

Πίνακας 24: Δημιουργία event handler και εγγραφή στη υπηρεσία λήψης events

```
var eSource = new EventSource("URL_TO_EVENTSERVER?queueName="+queueName);
```

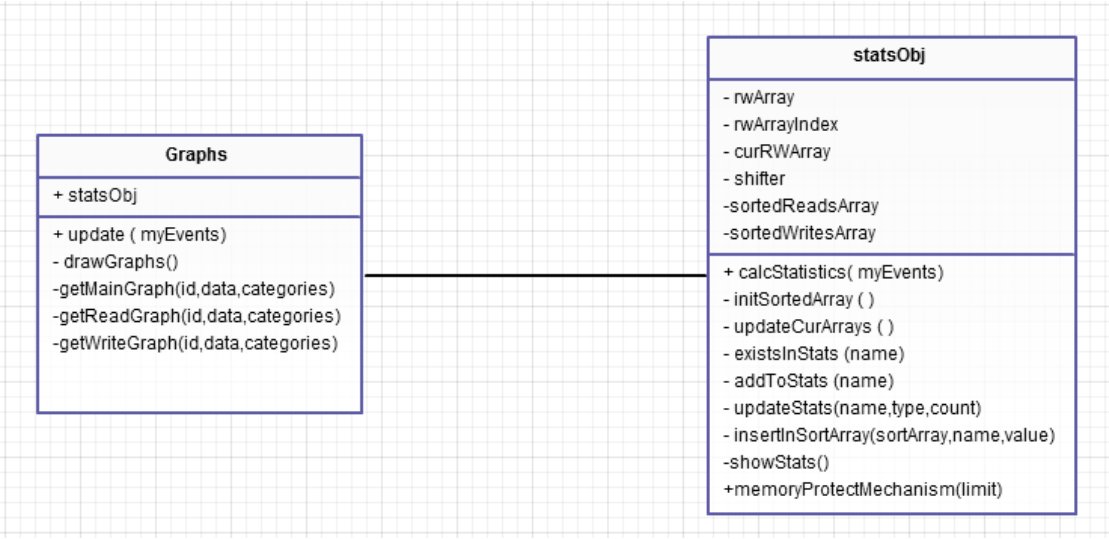
Μετά τη εγγραφή στη υπηρεσία λήψης SSEs μπορεί να αρχίσει η ροή events προς τον περιηγητή του χρήστη. Η JavaScript μας δίνει τη δυνατότητα να μπορούμε να διαχειριστούμε τη ροή των γεγονότων προσθέτοντας λειτουργίες στα πεδία του *event handler*. Στα πεδία *onopen* (άνοιγμα ροής events) και *onerror* (εμφάνιση σφάλματος) προσθέτουμε μόνο σχόλια κονσόλας.

Το πιο σημαντικό πεδίο είναι το πεδίο *onmessage* (άφιξη event) το οποίο μας ενημερώνει για τη άφιξη event και από κει μπορούμε να το διαχειριστούμε. Στο πεδίο αυτό προσθέτουμε μια συνάρτηση ώστε κατά τη άφιξη event το πρώτο που κάνουμε είναι να το αναλύσουμε (parse) κατά JSON και αμέσως μετά ελέγχουμε αν το event αυτό περιέχει χρήσιμη πληροφορία που αφορά τον χρήστη. Αν δεν περιέχει χρήσιμη πληροφορία, "*πετάμε*" το event και περιμένουμε το επόμενο. Αν όμως περιέχει χρήσιμη πληροφορία, αποθηκεύουμε τα αντικείμενα που αφορούν τον χρήστη σε μια προσωρινή λίστα (*myEvents*) και ενεργοποιούμε τις απαραίτητες συναρτήσεις, οι οποίες είναι υπεύθυνες για τη άντληση, επεξεργασία και παρουσίαση της πληροφορίας.

Πίνακας 25: Παράδειγμα event

```
{ "topic" : "Billing",
  "aggregation" : "no of objects written",
  "timestamp" : "2012-09-15 17:01:57.664",
  "tstart" : "2012-09-15 17:01:57.664",
  "tend" : "2012-09-15 17:01:57.664",
  "cluster" : "test",
  "originating_machine" : "147.102.19.186",
  "originating_service" : "AggregatedEventsServer",
  "units" : "",
  "users" :  [{ "tenant" : "tenant1",
                "count" : 156,
                "container" : "container4",
                "object" : "object20",
                "user" : "user6"
              },
              { "tenant" : "tenant1",
                "count" : 138,
                "container" : "container8",
                "object" : "object30",
                "user" : "user2"
              }]
}
```

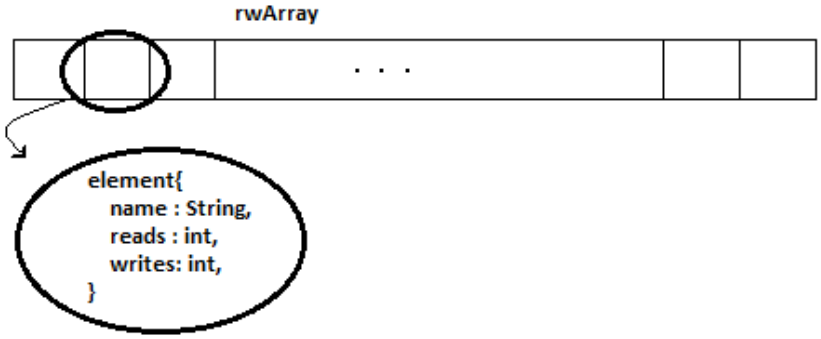
Δυο είναι τα κύρια αντικείμενα που αναλαμβάνουν την επεξεργασία και παρουσίαση των αποτελεσμάτων στη διαδικτυακή υπηρεσία. Το πρώτο αντικείμενο ονομάζεται *statsObj* και είναι υπεύθυνο για την επεξεργασία της λίστας *myEvents* αλλά και για την αποθήκευση των στατιστικών σε πίνακες και το δεύτερο αντικείμενο ονομάζεται *Graphs* και είναι υπεύθυνο για την διαχείριση και σχεδιασμό των γραφικών παραστάσεων.



Εικόνα 25: class diagram

Με τη άφιξη νέου event, πρώτα το επεξεργαζόμαστε παίρνοντας τα αντικείμενα που αφορούν τον χρήστη και τα βάζουμε στη λίστα *myEvents*, έπειτα δίνουμε τη λίστα αυτή στο αντικείμενο *statsObj* για περαιτέρω επεξεργασία.

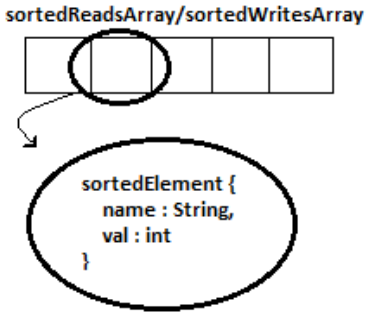
Για τη διαχείριση και αποθήκευση των αντικειμένων που έχει προσπελάσει ο χρήστης στο νέφος, χρησιμοποιούμε ένα πίνακα με όνομα *rwArray* στο *statsObj*. Ο πίνακας *rwArray* είναι πίνακας αντικειμένων με κάθε στοιχείο του να είναι αντικείμενο τύπου element όπως φαίνεται παρακάτω:



Εικόνα 26: Μορφή rwArray

Έτσι, για κάθε στοιχείο της λίστας *myEvents*, ελέγχουμε αν υπάρχει ήδη στον πίνακα *rwArray*. Αν δεν υπάρχει, τότε προσθέτουμε ένα νέο στοιχείο στον πίνακα *rwArray*. Αν υπάρχει, τότε απλά κάνουμε update τους read/write counters του.

Εκτός από τον *rwArray*, στο αντικείμενο *statsObj* βρίσκονται ακόμη δυο πίνακες (*sortedReadsArray* και *sortedWritesArray*), οι οποίοι είναι ταξινομημένοι. Οι δυο αυτοί πίνακες κρατάνε τα πέντε αντικείμενα με τα περισσότερα reads/writes αντίστοιχα και έχουν τη παρακάτω μορφή:



Εικόνα 27: Μορφή sorted arrays

Οι δυο αυτοί πίνακες είναι αναγκαίοι έτσι ώστε να μπορούμε να παρουσιάζουμε τα top reads/writes του χρήστη σε μορφή γραφικών παραστάσεων. Θα μπορούσαμε να μην χρησιμοποιούσαμε τους δυο αυτούς πίνακες όμως τότε θα έπρεπε μετά τη επεξεργασία κάθε event, να ταξινομούμε τον *rwArray* με κόστος O(nlogn) κάθε φορά. Όμως με τη αύξηση του πλήθους των στοιχείων του *rwArray* να αυξάνεται συνεχώς θα σπαταλείται χρόνος άδικα στη ταξινόμηση. Έτσι, προτιμήθηκε η χρήση πινάκων πέντε στοιχείων αφού μόνο πέντε στοιχεία θα παρουσιάζονται στις γραφικές παραστάσεις. Ο αλγόριθμος που χρησιμοποιήθηκε είναι ο εξής:

Για κάθε αντικείμενο στο event που έφθασε, ελέγχεται αν υπάρχει ήδη σε ταξινομημένο πίνακα (αν είναι τύπου read στον *sortedReadsArray* ή write στον *sortedWritesArray*). Αν υπάρχει, τότε γίνεται update η τιμή του και ελέγχεται με τα γειτονικά του στοιχεία και αν χρειάζεται το μετακινούμε σε καλύτερη θέση. Αν δεν υπάρχει στον ανάλογο ταξινομημένο πίνακα, τότε ελέγχουμε τη τιμή του με τη τιμή του αντικειμένου στη τελευταία θέση του πίνακα και αν είναι μεγαλύτερη τότε παίρνει τη θέση του. Αν η τιμή του δεν είναι μεγαλύτερη από τη τιμή του τελευταίου στοιχείου δεν κάνουμε κάτι. Με τον αλγόριθμο αυτό έχουμε σταθερό χρόνο αφού ουσιαστικά ελέγχουμε το πολύ πέντε στοιχεία και γλιτώνουμε τη ταξινόμηση ολόκληρου του πίνακα.

Αμέσως μετά τη επεξεργασία του event και τη ενημέρωση των στατιστικών, τη σκυτάλη αναλαμβάνει το αντικείμενο *Graphs* για τη ανανέωση των τιμών των γραφικών παραστάσεων.

Για τον σχεδιασμό των γραφικών παραστάσεων χρησιμοποιούμε το εργαλείο Rgraph (περιγραφή και ο λόγος που επιλέχθηκε εξηγείται αργότερα). Το Rgraph μας επιτρέπει να διαχειριζόμαστε τις γραφικές παραστάσεις ως JavaScript αντικείμενα έτσι, με τη δημιουργία μιας γραφικής παράστασης και τη αρχικοποίηση των παραμέτρων της (είδος, χρώματα, μέγεθος κ.α) μπορούμε να τη "φορτώσουμε" στο αντικείμενο *window* του περιηγητή του χρήστη έτσι ώστε με τη αλλαγή των δεδομένων (αλλαγή δεδομένων έχουμε με έλευση νέου event) να μην χρειάζεται κάθε φορά εκ νέου κατασκευή από τη αρχή νέας γραφικής παράστασης. Με τον τρόπο αυτό κάνουμε μόνο αλλαγή των τιμών των στοιχείων που δείχνει η γραφική και γλιτώνουμε τη δημιουργία, σε κάθε έλευση event, νέων αντικειμένων (γραφική) που καταλαμβάνουν χώρο στη μνήμη και επιβαρύνουν τον περιηγητή.

Σύμφωνα με τη λειτουργική απαίτηση που αφορά τη κύρια γραφική παράσταση, θέλουμε να υπάρχει ένας μηχανισμός σελιδοποίησης (paging) των αντικειμένων που αφορούν τον χρήστη με βάση των χρόνο άφιξης τους. Ο λόγος που είναι αναγκαίος ο μηχανισμός αυτός είναι επειδή τα αντικείμενα είναι πολλά και το πλήθος τους άγνωστο. Έτσι, αρχικά η κύρια γραφική παράσταση εμφανίζει τα πρώτα πέντε αντικείμενα και ο χρήστης εφόσον το επιθυμεί, μπορεί με τα κουμπιά πλοήγησης (Next/Prev) να διασχίσει τη λίστα αντικειμένων. Ακόμη του δίνεται η δυνατότητα με τα κουμπιά αύξησης/μείωσης Zoom να μπορεί να αυξήσει ή να μειώσει αντίστοιχα, τα πλήθος των αντικειμένων που βλέπει στη γραφική.

Για να πετύχουμε τη σελιδοποίηση χρησιμοποιούμε ένα επιπλέον πίνακα με όνομα *curRWArray* και αρχικό μέγεθος πέντε στοιχείων. Το μέγεθος του πίνακα εξαρτάται από το πόσα αντικείμενα επιθυμεί να βλέπει σε μια σελίδα της γραφικής ο χρήστης και μεταβάλλεται με τα κουμπιά του Zoom. Δουλειά του πίνακα *curRWArray* είναι να κρατάει τα στατιστικά των στοιχείων της σελίδας που δείχνει αυτή τη στιγμή η γραφική παράσταση. Έτσι, με τη επεξεργασία ενός event, μετά τη ενημέρωση του *rwArray* με τις νέες τιμές των αντικειμένων, ελέγχεται αν χρειάζεται να ενημερωθεί και ο *curRWArray* γιατί τα αντικείμενα αυτά βρίσκονται στη σελίδα της γραφικής που βλέπει ο χρήστης.

Μηχανισμός προστασίας κατανάλωσης μνήμης

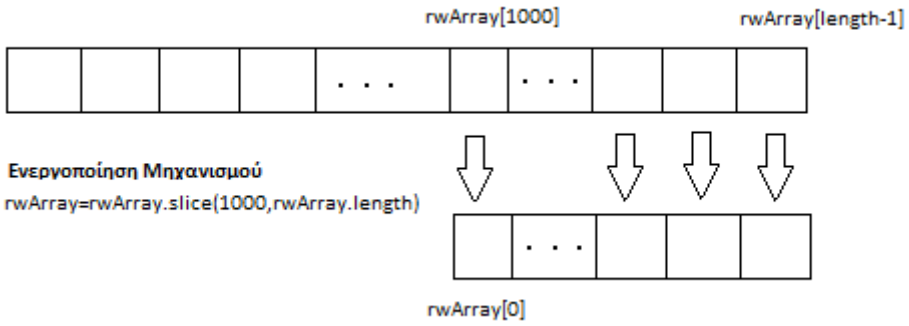
Ίσως το μεγαλύτερο μειονέκτημα της χρήσης Server-Side Events σε διαδικτυακή εφαρμογή είναι ότι δεν μπορούμε να γνωρίζουμε από πριν τη μνήμη που καταναλώνει η εφαρμογή. Αυτό οφείλεται στο γεγονός ότι δεν γνωρίζουμε από πριν, τη συχνότητα με τη οποία φθάνουν τα events και δεν ξέρουμε το μέγεθος του κάθε event. Οι δυο αυτοί παράμετροι επηρεάζουν τη κατανάλωση μνήμης που δεσμεύει η διαδικτυακή υπηρεσία στον υπολογιστή του χρήστη.

Για να προστατευθεί ο χρήστης από τη υπερβολική ζήτηση μνήμης από τη διαδικτυακή υπηρεσία που μπορεί να προκαλέσει από καθυστέρηση στη

επεξεργασία των event ή/και στη παρουσίαση των αποτελεσμάτων μέχρι και παγώματα του περιηγητή του χρήστη, απαιτείται η προσθήκη στη διαδικτυακή υπηρεσία ενός μηχανισμού προστασίας τον χρήστη από οριακές συνθήκες.

Ο μηχανισμός που επιλέξαμε να υλοποιήσουμε αναλαμβάνει τη εποπτεία του πλήθους των αντικειμένων που βρίσκονται στον πίνακα *rwArray*. Ο πίνακας αυτός είναι ο πιο σημαντικός πίνακας της εφαρμογής καθώς εκεί αποθηκεύονται τα στατιστικά των αντικειμένων που έχει προσπελάσει στο νέφος ο χρήστης, και από αυτόν τον πίνακα εξαρτώνται όλα τα εργαλεία παρουσίασης της εφαρμογής. Η συνεχής αύξηση του μεγέθους του *rwArray* αυξάνει τη ανάγκη για δέσμευση περισσότερης μνήμης, για τον ίδιο τον πίνακα αλλά και στον DOM για τα εργαλεία παρουσίασης, και ακόμη αυξάνει τον χρόνο επεξεργασίας των event καθώς η αναζήτηση, η ενημέρωση και η ταξινόμηση των στοιχείων του πίνακα χρειάζεται περισσότερο χρόνο.

Επομένως ο μηχανισμός που υλοποιήθηκε ελέγχει τον πίνακα έτσι ώστε το μέγεθος του να μην ξεπερνάει ποτέ τα 1000 αντικείμενα. Μετά τη επεξεργασία ενός καινούργιου event γίνεται έλεγχος του πλήθους των αντικειμένων στον *rwArray* και αν το πλήθος είναι μεγαλύτερο του 1000 τότε ενεργοποιείται ο μηχανισμός ο οποίος αναλαμβάνει “κόψει” (slice) τον πίνακα, “διώχνοντας” έτσι τα 1000 παλαιότερα αντικείμενα από τον πίνακα και διατηρώντας μόνο τα νεότερα. Αυτό είναι δυνατόν να συμβεί με ένα πέρασμα, *O(n)*, γιατί η προσθήκη νέων στοιχείων του πίνακα πάντα γίνεται στο τέλος του, έτσι κρατάμε τα αντικείμενα από τη θέση `rwArray[1000]` μέχρι `rwArray[length-1]` όπως φαίνεται παρακάτω:



Εικόνα 28: Αναπαράσταση μηχανισμού προστασίας χρήστη στη μνήμη

Μετά τον περιορισμό του μεγέθους του πίνακα πρέπει να ενημερωθούν οι πίνακες που κρατάνε τα στοιχεία με τις μεγαλύτερες προσβάσεις και οι πίνακες που δείχνουν τα τρέχοντα αντικείμενα στη κύρια γραφική έτσι ώστε οι γραφικές παραστάσεις να ενημερωθούν και αυτές με τη σειρά τους.

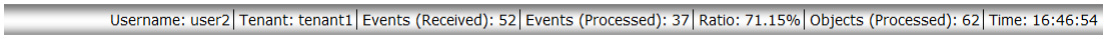
Με τη προσθήκη του μηχανισμού όπως θα δούμε παρακάτω στις μετρήσεις που πήραμε για τη κατανάλωση μνήμης στη πλευρά του πελάτη πετυχαίνουμε να

σταθεροποιήσουμε τις ανάγκες για μνήμη και μένουμε συνεπής στη μη λειτουργική απαίτηση για *Real-Time Statistics* γιατί κρατάμε πάντα τα 1000 τελευταία αντικείμενα που προσπέλασε ο χρήστης και έτσι δείχνουμε τη πραγματικά συμβαίνει εκείνη τη στιγμή στο νέφος και τα τρέχοντα στοιχεία δεν κρύβονται από παλαιότερα με μεγαλύτερες τιμές.

Εργαλεία παρακολούθησης (monitoring)

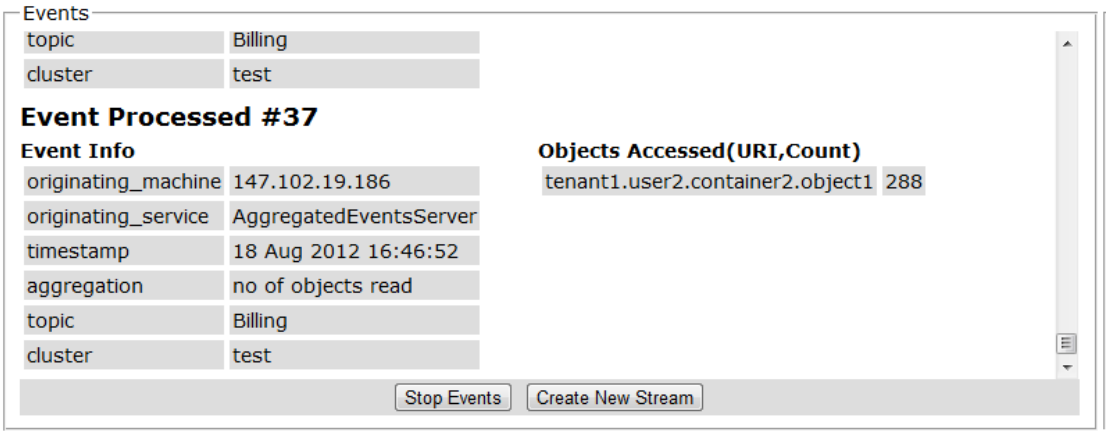
Τα εργαλεία που βρίσκονται στη διάθεση του χρήστη για παρακολούθηση (monitoring) του νέφους είναι:

Status Bar: Στη μπάρα αυτή εκτός από το username και tenant του χρήστη, αναγράφεται ο συνολικός αριθμός των events (Events Received), ο αριθμός των events που αφορούν τον χρήστη (Events Processed), ο αριθμός των αντικειμένων που επεξεργάστηκαν (Objects Processed) και ο λόγος *events που αφορούν τον χρήστη προς events που φθάνουν* (Ratio).



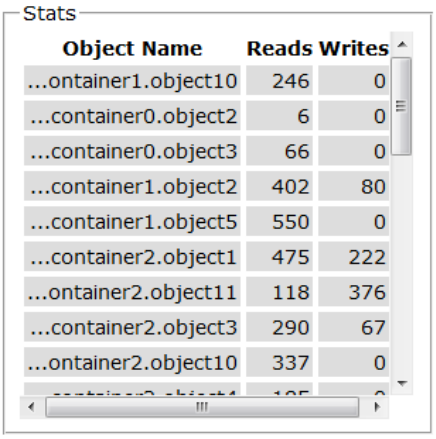
Εικόνα 29: Status Bar – EventPage.jsp

Κονσόλα παρουσίασης event: Στη κονσόλα αυτή παρουσιάζονται τα στοιχεία κάθε event καθώς και τα αντικείμενα που αφορούν τον χρήστη σε μορφή λίστας.



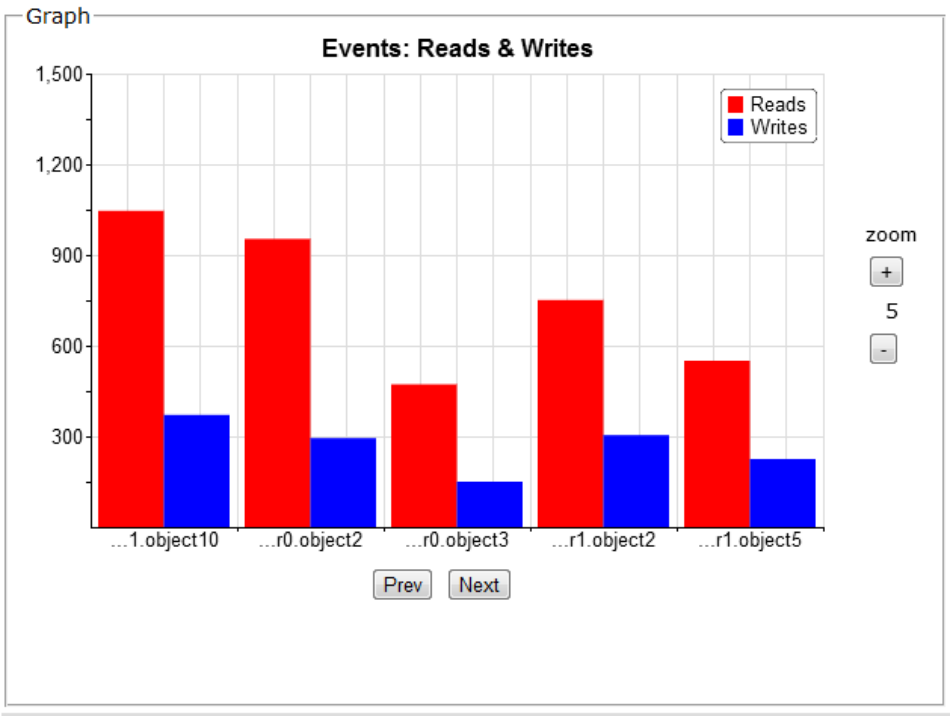
Εικόνα 30: Κονσόλα παρουσίασης Event – EventPage.jsp

Πίνακας αντικειμένων: Στον χώρο αυτό δίνεται σε μορφή πίνακα η λίστα όλων των αντικειμένων που επεξεργάστηκαν, καθώς και ο αριθμός των Reads/Writes τους.



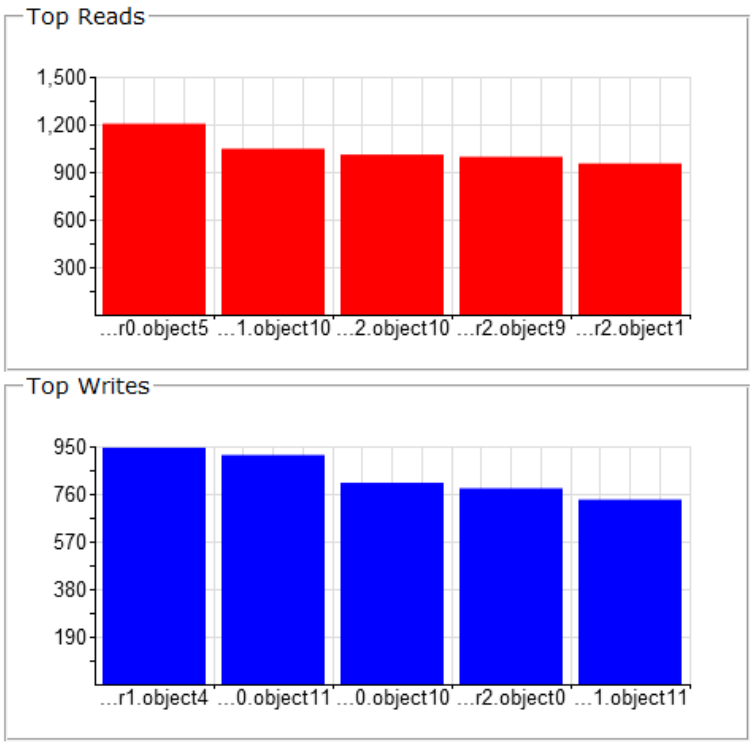
Εικόνα 31: Πίνακας αντικειμένων – EventPage.jsp

Κύρια γραφική παράσταση: Σε αυτή τη γραφική παράσταση μπορούμε να πλοηγηθούμε μέσα από τη λίστα των αντικειμένων που επεξεργάστηκαν σε χρονολογική σειρά. Αρχικά, στη λίστα φαίνονται τα πρώτα πέντε αντικείμενα. Για να δούμε τα επόμενα ή για να γυρίζουμε σε προηγούμενα υπάρχουν τα κουμπιά πλοήγησης *Next/Prev*. Αν θέλουμε να βλέπουμε περισσότερα ή λιγότερα από πέντε αντικείμενα στη γραφική, μπορούμε να χρησιμοποιήσουμε τα κουμπιά στο πλάι για να κάνουμε Zoom.



Εικόνα 32: Κύρια γραφική παράσταση – EventPage.jsp

Top Reads/Writes γραφικές: Στις γραφικές αυτές δίνονται τα πέντε πρώτα σε προσβάσεις για διάβασμα/γράψιμο αντικείμενα που αφορούν το χρήστη.



Εικόνα 33: Top Reads/Writes γραφικές - EventPage.jsp

Εργαλεία για δοκιμές

Εργαλεία προσομοίωσης μηχανισμών του νέφους

Στα πλαίσια της ανάπτυξης της διαδικτυακής υπηρεσίας χρησιμοποιήθηκαν διάφορα εργαλεία και Java APIs για να προσομοιώσουμε τη πλευρά του νέφους κάτω από διαφορετικές συνθήκες λειτουργίας. Η προσομοίωση αυτή ήταν αναγκαία έτσι ώστε να δοκιμαστεί η διαδικτυακή υπηρεσία και να πάρουμε διάφορες μετρήσεις για να δούμε αν η υπηρεσία πληρεί τις απαιτήσεις που θέσαμε πιο πριν. Ο κώδικας που υλοποιεί όλους τους μηχανισμούς που αναπτύχθηκαν βρίσκεται στο παράρτημα.

Αρχικά χρειαστήκαμε ένα web server ο οποίος αναλαμβάνει τον ρόλο του νέφους στις δοκιμές που πραγματοποιήσαμε. Για τον ρόλο αυτό επιλέχθηκε ο **Apache Tomcat 7**[47]. Ο *Apache Tomcat* είναι ένας ελεύθερου λογισμικού web server και servlet container ο οποίος αναπτύσσεται από το Apache Software Foundation. Ο *Tomcat* υλοποιεί τις προδιαγραφές των Java Servlets και Java Server Pages (JSPs) που θέτει η Oracle και παρέχει στον προγραμματιστή έναν γνήσιο Java HTTP web server περιβάλλον για να τρέχει κώδικας Java στο διαδίκτυο.

Επόμενο βήμα ήταν η υλοποίηση ενός μηχανισμού εξυπηρέτησης "*RESTful*" HTTP αιτημάτων για να προσομοιώσουμε τη επικοινωνία μεταξύ της διαδικτυακής υπηρεσίας και του νέφους σύμφωνα με το πρωτόκολλο CDMI. Για τη υλοποίηση ενός τέτοιου μηχανισμού χρησιμοποιήθηκε το **Jersey API**[48][49]. Το Jersey API είναι ένα ανοικτού κώδικα API το οποίο υλοποιεί το *JAX-RS* (Java API for RESTful Web Services) για τη δημιουργία διαδικτυακών υπηρεσιών σύμφωνα με το αρχιτεκτονικό μοντέλο REST.

Για τη επαλήθευση των στοιχείων που δίνει ο χρήστης κατά τη είσοδο του στο σύστημα, αναπτύχθηκε ένας *AuthenticateServer*. Ο *AuthenticateServer* παραλαμβάνει το "RESTful" HTTP GET αίτημα από τη πλευρά του πελάτη, επαληθεύει αν τα στοιχεία είναι σωστά και αναλόγως στέλνει απάντηση στο χρήστη. Αυτό που ελέγχει ο *AuthenticateServer* είναι η κωδικοποιημένη (base64) *authenticate* επικεφαλίδα και αν τα στοιχεία είναι ορθά απαντάει με *HTTP status 200 OK* ενώ αν είναι λάθος απαντάει με *401 Unauthorized*.

Για τη προσομοίωση της δημιουργίας ουράς, αναπτύχθηκε ένας *QueueServer*. Ο *QueueServer* παραλαμβάνει το "*RESTful*" HTTP PUT αίτημα από τη πλευρά του πελάτη. Αρχικά, ελέγχει αν οι επικεφαλίδες του αιτήματος είναι σύμφωνες με το CDMI πρωτόκολλο, αν δεν είναι επιστρέφει HTTP απάντηση με status *400 Bad request*. Ακολούθως ελέγχει αν υπάρχει ήδη κάποια ουρά με το

όνομα που δόθηκε από τον χρήστη. Αν υπάρχει ήδη το δοθέν όνομα τότε απαντάει με HTTP απάντηση με status *409 Conflict*. Αν το αίτημα περνάει και από τους δυο αυτούς ελέγχους τότε ο *QueueServer* "φτιάχνει" μια καινούργια ουρά σε μια MySQL βάση δεδομένων. Ακολούθως τα στοιχεία της ουράς επιστρέφονται ως *JSON String* στο σώμα HTTP απάντησης με status *201 Created*.

Ακολούθως χρειαστήκαμε να προσομοιώσουμε ένα μηχανισμό παραγωγής event που θα δημιουργεί events και θα τα προωθεί στους πελάτες που θα εγγράφονται μέσω της δημιουργίας ουράς στη υπηρεσία. Για τη δημιουργία ενός τέτοιου μηχανισμού χρησιμοποιήθηκε το **Jetty web server**[50]. Το *Jetty* είναι ένας HTTP server και Servlet container, ελεύθερου λογισμικού, γραμμένο σε Java και αναπτύσσεται από το Eclipse Foundation. Με το *Jetty* μπορούμε να δημιουργήσουμε μηχανισμούς προώθησης events, παρέχοντας μας μια πλήρης βιβλιοθήκη από κλάσεις όπως οι *EventSourceServlet* και *EventSource*.

Για το μηχανισμό αυτό, αναπτύχθηκε ένας *EventServer* όπου δέχεται αίτηση από τη πλευρά για εγγραφή στη υπηρεσία. Αν η αίτηση ακολουθεί το πρότυπο που θέσαμε τότε δημιουργείται η ανάλογη σύνδεση με τον πελάτη και θα δέχεται events από τον εξυπηρετητή. Με ανάλογη αίτηση ο πελάτης μπορεί να κλείσει τη σύνδεση. Είναι σημαντικό να τονιστεί ότι μόνο ένας εξυπηρετητής υπάρχει και εξυπηρετεί όλες τις συνδέσεις, προωθώντας κάθε φορά το event προς όλους τους ενδιαφερόμενους.

Επιλογή εργαλείου σχεδιασμού γραφικών παραστάσεων

Για το σχεδιασμό τον γραφικών παραστάσεων δοκιμάσαμε τρία διαφορετικά JavaScript frameworks για να δούμε ποιο είναι το καταλληλότερο για χρήση στη διαδικτυακή υπηρεσία. Τα τρία JavaScript frameworks που επιλέχθηκαν για δοκιμή ήταν το jqPlot, το extJS 4.1 και το Rgraph. Μετά τη δοκιμή και των τριών frameworks επιλέχθηκε τελικά το Rgraph.

Το **jqPlot**[51] είναι πρόσθετο (plugin) του JQuery JavaScript framework, κατάλληλο για τη κατασκευή γραφικών παραστάσεων. Το jqPlot εκμεταλλεύεται τις συναρτήσεις του JQuery έτσι αποτελεί ένα ελαφρύ εργαλείο για κατασκευή γραφικών παραστάσεων. Δεν επιλέχθηκε γιατί χρησιμοποιεί το DOM αντί να εκμεταλλεύεται τη δυναμική του HTML5 canvas και γιατί για κάθε αλλαγή στα δεδομένα θα πρέπει να κατασκευαστεί εξ αρχής η κάθε γραφική παράσταση με αποτέλεσμα επιπλέον φόρτο.

Το **ext-JS 4.1**[52] αποτελεί ένα πολύ δυνατό JavaScript framework κατάλληλο για τη κατασκευή γραφικών παραστάσεων αλλά και σύνθετων γραφικών. Εκμεταλλεύεται τις δυνατότητες του HTML5 canvas όμως στα αρνητικά του, εκτός από το γεγονός ότι για κάθε αλλαγή στα δεδομένα θα πρέπει να κατασκευαστεί εξ αρχής η κάθε γραφική παράσταση, είναι ένα πολύ βαρύ εργαλείο καθιστώντας το ακατάλληλο για τη εργασία που θέλουμε να κάνει.

Το **Rgraph**[53] project ξεκίνησε το 2008 από τον Richard Heyes με στόχο τη δημιουργία ενός ισχυρού JavaScript framework με βάση τη HTML5 για το σχεδιασμό γραφικών παραστάσεων. Το Rgraph εκμεταλλεύεται το HTML5 στοιχείο canvas, είναι μικρό σε μέγεθος και μπορούν να φορτωθούν μόνο οι αναγκαίες βιβλιοθήκες για ακόμη μεγαλύτερη μείωση στο χρόνο φόρτωσης. Το Rgraph επιλέχθηκε έναντι των άλλων δυο frameworks επειδή μας δίνεται η δυνατότητα μέσω JavaScript να φορτώνουμε στο αντικείμενο *window* του περιηγητή τη γραφική μας παράσταση έτσι ώστε με αλλαγή των δεδομένων να μην χρειάζεται εκ νέου η κατασκευή από τη αρχή της γραφικής αλλά μόνο η αλλαγή των τιμών των στοιχείων.

Πειραματικές μετρήσεις

Στα πλαίσια των δοκιμών της διαδικτυακής εφαρμογής έγιναν κάποιες μετρήσεις στη πλευρά του πελάτη για να δούμε την επίδραση που έχει η διαδικτυακή υπηρεσία στη κατανάλωση μνήμης.

Όπως αναφέρθηκε στη περιγραφή της σελίδας παρακολούθησης γεγονότων, το μεγαλύτερο μειονέκτημα του EventSource είναι ότι δεν γνωρίζουμε από πριν τη μνήμη που καταναλώνει η εφαρμογή. Αυτό οφείλεται στο γεγονός ότι δεν γνωρίζουμε κάθε πόσο έρχονται τα events και το μέγεθος του κάθε event.

Για να δούμε τη επίδραση της διαδικτυακής υπηρεσίας στη πλευρά του πελάτη πραγματοποιήθηκαν δυο είδη μετρήσεων. Το πρώτο είδος μετρήσεων έγινε χωρίς τη ύπαρξη μηχανισμού προστασίας στη διαδικτυακή υπηρεσία και το δεύτερο είδος με τη ενσωμάτωση του μηχανισμού στη υπηρεσία.

Οι μετρήσεις που πραγματοποιήθηκαν έγιναν σε υπολογιστή Toshiba laptop με επεξεργαστή Intel Core 2 duo στα 2GHz με 3GB RAM. Χρησιμοποιήθηκε ο περιηγητής Google Chrome (έκδοση 20.0.1132.57) και για τη μέτρηση της χρησιμοποιημένης μνήμης χρησιμοποιήθηκε το εργαλείο "*about:memory*"[54].

Χωρίς μηχανισμό προστασίας

Παρακάτω παρουσιάζονται δυο σετ μετρήσεων για να δούμε την επίδραση που έχει η διαδικτυακή υπηρεσία στη κατανάλωση μνήμης στη πλευρά του πελάτη. Στο πρώτο σετ μετρήσεων θέλουμε να δούμε την επίδραση που έχει η συχνότητα με την οποία φθάνουν στον περιηγητή του χρήστη τα events και στο δεύτερο σετ θέλουμε να δούμε τη επίδραση που έχει το ratio *events που αφορούν τον χρήστη προς events που φθάνουν*. Οι μετρήσεις αυτές έγιναν χωρίς τη ύπαρξη μηχανισμού προστασίας του πελάτη για να μπορούμε να δούμε σε βάθος χρόνου την επίδραση που έχουν οι παράμετροι που θέλουμε να μετρήσουμε και ακόμη να συγκρίνουμε και με μετρήσεις που έγιναν χρησιμοποιώντας τον μηχανισμό.

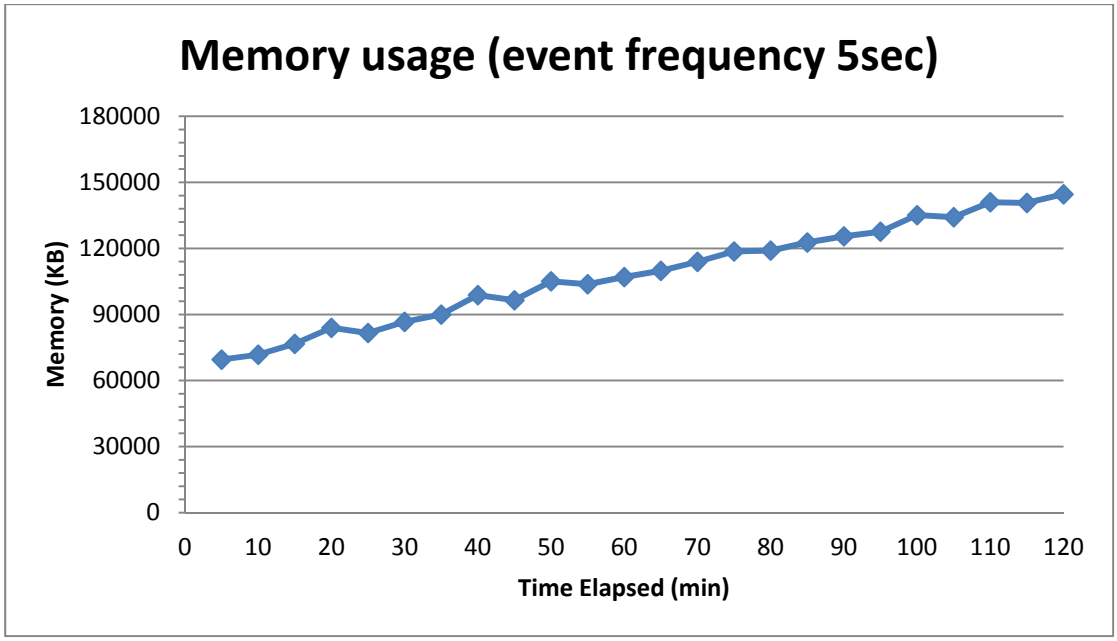
1^{ο} set μετρήσεων

Στο πρώτο set μετρήσεων καταγράφονταν τα εξής στατιστικά: τα events που φθάνουν, τα events που αφορούν τον συγκεκριμένο χρήστη, τα αντικείμενα σε κάθε event που αφορούν τον χρήστη και τέλος η χρησιμοποιημένη μνήμη. Τα στατιστικά αυτά καταγράφονταν κάθε 5 λεπτά για διάστημα 2 ωρών. Κάναμε τρεις δοκιμές. Για να έχουμε μια σωστή βάση μέτρησης στις μετρήσεις μας, η αποστολή event από τον εξυπηρετητή γινόταν σε σταθερό χρόνο. Η πρώτη δοκιμή έγινε με συχνότητα

εμφάνισης event κάθε 5 δευτερόλεπτα, η δεύτερη με συχνότητα κάθε 1 δευτερόλεπτο και η τρίτη με συχνότητα κάθε 8 δευτερόλεπτα. Για τις μετρήσεις αυτές είχαμε ένα υψηλό λόγο *events που αφορούν τον χρήστη προς events που φθάνουν* που άγγιζε το 72% με σκοπό να δούμε τη χρήση της εφαρμογής σε συνθήκες πάρα πολύ υψηλής καταπόνησης.

1η μέτρηση: *Συχνότητα εμφάνισης event κάθε 5 δευτερόλεπτα*

A/A	Time Elapsed (min)	Events Received	Events Processed	Objects Processed	Memory (KB)
0	0	0	0	0	0
1	5	58	44	75	69512
2	10	122	86	153	71744
3	15	178	123	222	76668
4	20	236	164	292	83936
5	25	297	209	364	81592
6	30	356	255	443	86664
7	35	416	297	522	89976
8	40	477	347	612	98808
9	45	540	392	689	96464
10	50	599	439	776	105080
11	55	565	477	839	103757
12	60	717	519	940	106992
13	65	766	559	1001	109834
14	70	836	608	1097	113924
15	75	897	656	1184	118628
16	80	962	701	1268	119036
17	85	1018	740	1354	122692
18	90	1076	780	1423	125493
19	95	1136	813	1478	127616
20	100	1200	860	1561	135116
21	105	1258	903	1641	134196
22	110	1316	939	1704	140984
23	115	1365	973	1764	140676
24	120	1412	1021	1823	144593

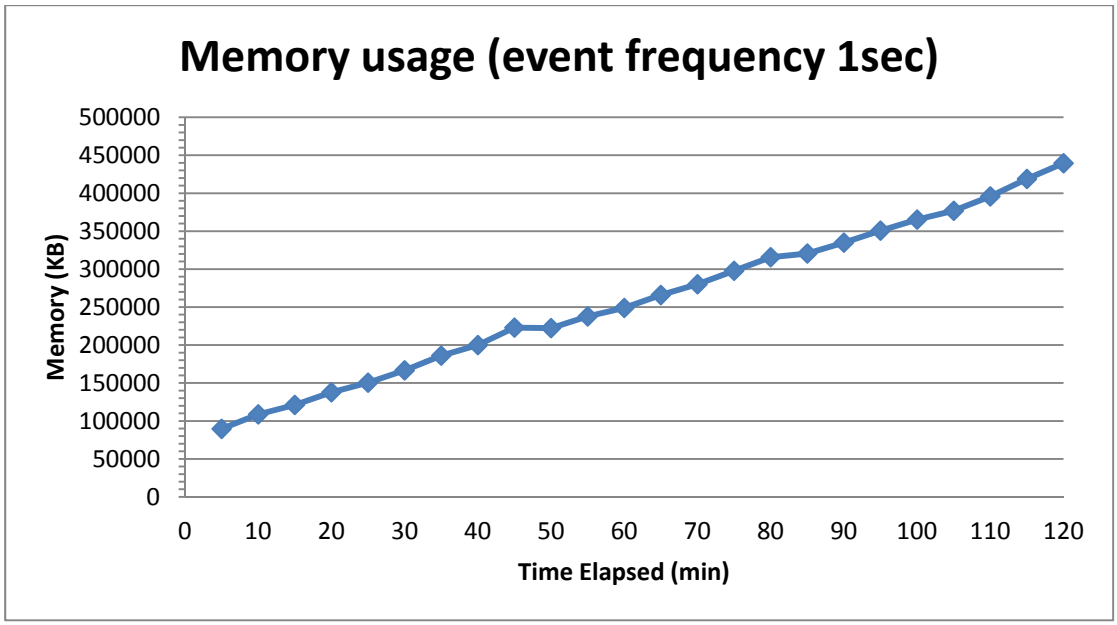


Εικόνα 34: Γραφική 1, Memory usage (event frequency 5sec)

2η μέτρηση: *Συχνότητα εμφάνισης event κάθε 1 δευτερόλεπτο*

A/A	Time Elapsed (min)	Events Received	Events Processed	Objects Processed	Memory (KB)
0	0	0	0	0	0
1	5	316	224	400	89612
2	10	603	427	773	108700
3	15	918	646	1176	121120
4	20	1206	858	1561	137796
5	25	1503	1082	1949	150400
6	30	1800	1308	2368	166708
7	35	2107	1529	2777	186064
8	40	2400	1747	3166	200040
9	45	2702	1961	3580	222952
10	50	3059	2133	3927	222388
11	55	3358	2361	4330	237748
12	60	3682	2591	4748	248868
13	65	3961	2789	5134	265916
14	70	4268	3028	5536	280112
15	75	4584	3255	5943	297796
16	80	4867	3466	6327	315732
17	85	5171	3674	6702	320636
18	90	5462	3897	7113	334944
19	95	5754	4108	7516	350752
20	100	6070	4332	7927	365304
21	105	6362	4547	8313	376888

22	110	6663	4764	8719	395904
23	115	6993	4999	9150	418836
24	120	7301	5291	9429	439516

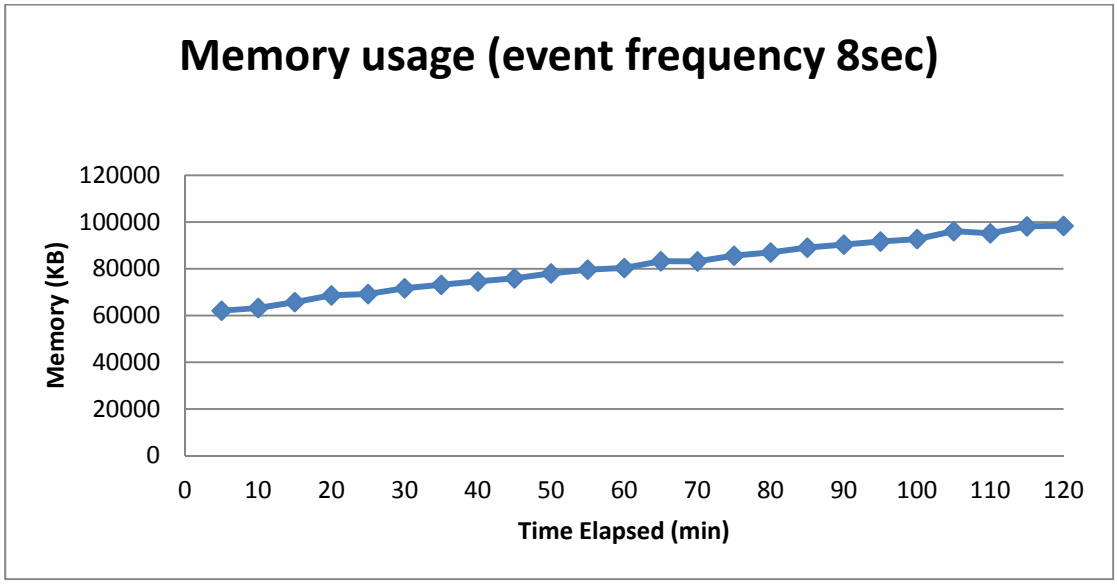


Εικόνα 35: Γραφική 2, Memory usage (event frequency 1sec)

3η μέτρηση: *Συχνότητα εμφάνισης event κάθε 8 δευτερόλεπτα*

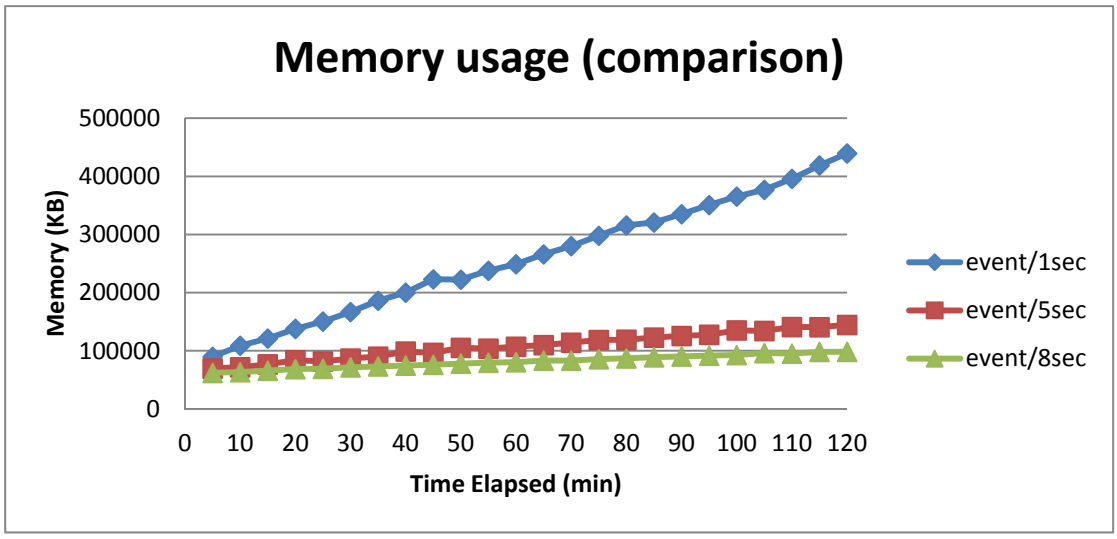
A/A	Time Elapsed (min)	Events Received	Events Processed	Objects Processed	Memory (KB)
0	0	0	0	0	0
1	5	34	23	37	62040
2	10	71	47	72	63220
3	15	111	71	108	65720
4	20	150	94	143	68622
5	25	183	113	169	69200
6	30	220	135	195	71660
7	35	258	158	225	73126
8	40	296	180	254	74564
9	45	333	200	282	75904
10	50	371	227	319	78020
11	55	410	249	352	79624
12	60	447	266	373	80336
13	65	487	288	400	83236
14	70	520	303	424	83164
15	75	558	327	466	85596
16	80	596	343	489	86960
17	85	633	359	513	89036

18	90	671	377	538	90344
19	95	708	404	574	91664
20	100	756	431	614	92700
21	105	782	445	633	96088
22	110	820	459	649	95168
23	115	859	479	674	98160
24	120	905	507	711	98336



Εικόνα 36: Γραφική 3, Memory usage (event frequency 8sec)

Παρακάτω φαίνονται τα αποτελέσματα των τριών μετρήσεων σε μια γραφική για καλύτερη κατανόηση της διαφοράς των αποτελεσμάτων:

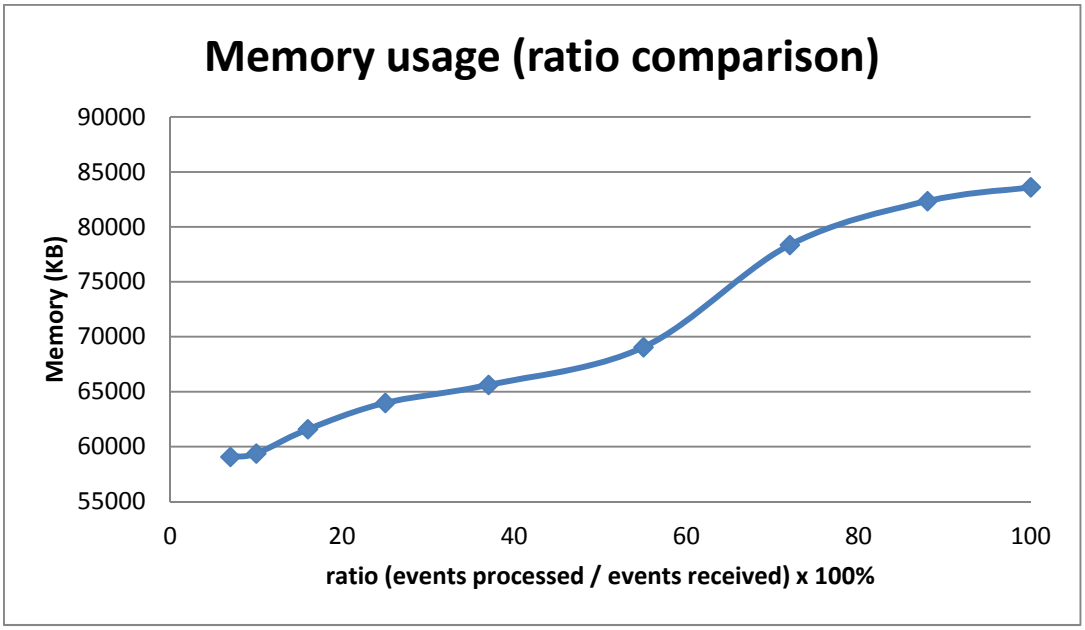


Εικόνα 37: Γραφική 4, Memory usage

2ο set μετρήσεων

Στο δεύτερο set μετρήσεων καταγράφονταν τα εξής στατιστικά: τα events που φθάνουν, τα events που αφορούν τον συγκεκριμένο χρήστη και τέλος η χρησιμοποιημένη μνήμη. Τα στατιστικά αυτά καταγράφονταν κάθε 5 λεπτά για διάστημα 45 λεπτών με συχνότητα εμφάνισης event κάθε 5 δευτερόλεπτα με διαφορετικό λόγο *events που αφορούν τον χρήστη προς events που φθάνουν κάθε φορά.* Στο τέλος κάθε μέτρησης βρίσκαμε τον μέσο όρο της χρησιμοποιημένης μνήμης.

Ratio (events processed / events received) x 100%	Μέσος όρος χρησιμοποιημένης μνήμης (KB)
100	83604
88	80342
72	78353
55	69046
37	65617
25	63978
16	61581
10	59369
7	59063



Εικόνα 38: Γραφική 5, Memory usage (ratio comparison)

Παρατηρήσεις

Από το 1^{ο} set μετρήσεων παρατηρούμε ότι η συχνότητα με τη οποία φθάνουν τα events στον περιηγητή του χρήστη έχει άμεση σχέση με τη μνήμη που καταναλώνει η διαδικτυακή υπηρεσία.

Από τη 2^{η} γραφική βλέπουμε ότι για συχνότητα άφιξης event της τάξης του ενός δευτερολέπτου η μνήμη που χρειάζεται η υπηρεσία για 2 ώρες λειτουργίας αγγίζει τα 430MB και κρίνεται πολύ μεγάλο το μέγεθος. Μάλιστα, σε δοκιμή που έγινε για 4 ώρες λειτουργίας παρατηρήθηκαν μικρά παγώματα του περιηγητή του χρήστη και μικρή καθυστέρηση στη παρουσίαση των events. Βέβαια το γεγονός ότι ο λόγος *events που αφορούν τον χρήστη προς events που φθάνουν κάθε φορά* ήταν πολύ μεγάλος (72%) έπαιξε και αυτό τον ρόλο του όμως όπως αναφέρθηκε και πιο πάνω θέλαμε να δούμε τη θα συμβεί σε συνθήκες πάρα πολύ μεγάλης καταπόνησης.

Από τις άλλες δυο μετρήσεις για συχνότητα άφιξης event στα 5 και 8 δευτερόλεπτα αντίστοιχα, παρατηρούμε ότι αν και αυξάνεται η μνήμη που δεσμεύεται για τη διαδικτυακή υπηρεσία όσο περνάει ο χρόνος, η αύξηση αυτή είναι ελάχιστη. Μάλιστα, από τη γραφική 4 βλέπουμε ότι σε σχέση με χρόνο άφιξης 1sec μπορούμε να χαρακτηρίσουμε τη κατανάλωση σταθερή.

Από το 2^{ο} set μετρήσεων παρατηρούμε ότι ο λόγος *events που αφορούν τον χρήστη προς events που φθάνουν κάθε φορά* έχει σχέση με τη μνήμη που καταναλώνει η διαδικτυακή υπηρεσία.

Αυτό είναι λογικό γιατί αν σε event δεν περιέχεται χρήσιμη πληροφορία για τον χρήστη τότε απλά “πετάμε” το event ενώ αν περιέχει τότε προχωράμε στη επεξεργασία του. Επομένως η αύξηση του λόγου αυτού, αυξάνει τη δέσμευση μνήμης στη πλευρά του πελάτη.

Συμπέρασμα

Από τα δυο σετ μετρήσεων παρατηρούμε ότι η συχνότητα άφιξης event αλλά και ο λόγος *events που αφορούν τον χρήστη προς events που φθάνουν κάθε φορά* έχουν σχέση με τη κατανάλωση μνήμης στη πλευρά του πελάτη. Και στις δυο περιπτώσεις δοκιμάσαμε πολύ οριακές συνθήκες καθώς events κάθε 1sec για διάστημα 2 ωρών με ratio 72% είναι σχεδόν αδύνατον να συμβεί όπως και ratio 100%. Αυτό όμως δεν αναιρεί το γεγονός ότι η πλευρά του πελάτη δεν μπορεί να γνωρίζει από πριν αν η κατανάλωση μνήμης ανεβαίνει κατακόρυφα για να μπορέσει να προστατευθεί.

Για τον λόγο αυτό είναι χρήσιμο να προστεθεί ένας μηχανισμός προστασίας στη διαδικτυακή υπηρεσία, προστατεύοντας τον χρήστη από οριακές συνθήκες.

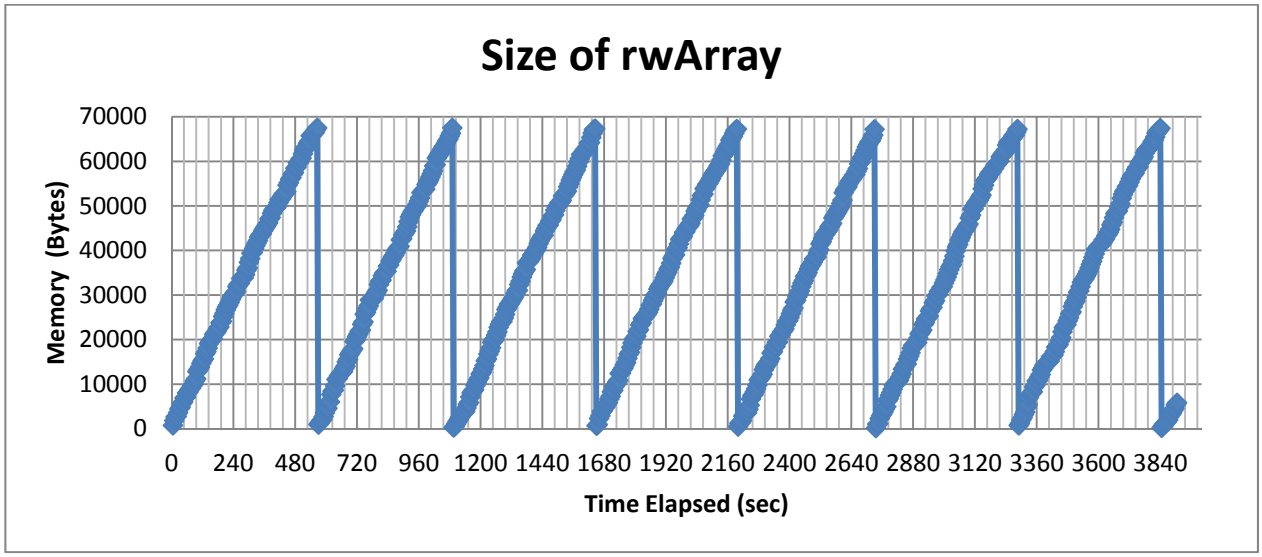
Με μηχανισμό προστασίας

Παρακάτω παρουσιάζονται τα αποτελέσματα μετρήσεων που έγιναν για να δούμε τη επίδραση της διαδικτυακής υπηρεσίας στη κατανάλωση μνήμης στη πλευρά του πελάτη. Οι μετρήσεις αυτές έγιναν με τον μηχανισμό προστασίας ενεργοποιημένο.

Για να έχουμε μια σωστή βάση μέτρησης στις μετρήσεις μας, η αποστολή event από τον εξυπηρετητή γινόταν και πάλι με σταθερό χρόνο και συγκεκριμένα κάθε 2 δευτερόλεπτα. Ακόμη, διατηρούσαμε ένα πολύ υψηλό λόγο *events που αφορούν τον χρήστη προς events που φθάνουν* που άγγιζε το 94% με σκοπό να δούμε τη χρήση της εφαρμογής σε συνθήκες πάρα πολύ υψηλής καταπόνησης. Στις μετρήσεις αυτές καταγράφαμε τη κατανάλωση μνήμης για διάστημα 1 ώρας.

Τα αποτελέσματα των μετρήσεων παρουσιάζονται στις δυο γραφικές παραστάσεις που ακολουθούν.

Η πρώτη γραφική παράσταση (γραφική 6) μας δείχνει πως μεταβάλλεται το μέγεθος του *rwArray* σε σχέση με τον χρόνο. Για τον σκοπό αυτό καταγράφαμε κάθε 5 δευτερόλεπτα το πλήθος των στοιχείων του πίνακα και το συνολικό μέγεθος του πίνακα σε Bytes. Για να πάρουμε το μέγεθος του πίνακα σε Bytes, σε κάθε μέτρηση παίρναμε το άθροισμα της αναπαράστασης σε String (String representation) κάθε αντικειμένου του πίνακα. Για τη παρουσίαση των μετρήσεων θα χρειάζονται σελίδες ολόκληρες, για τον λόγο αυτό παραλείπονται.



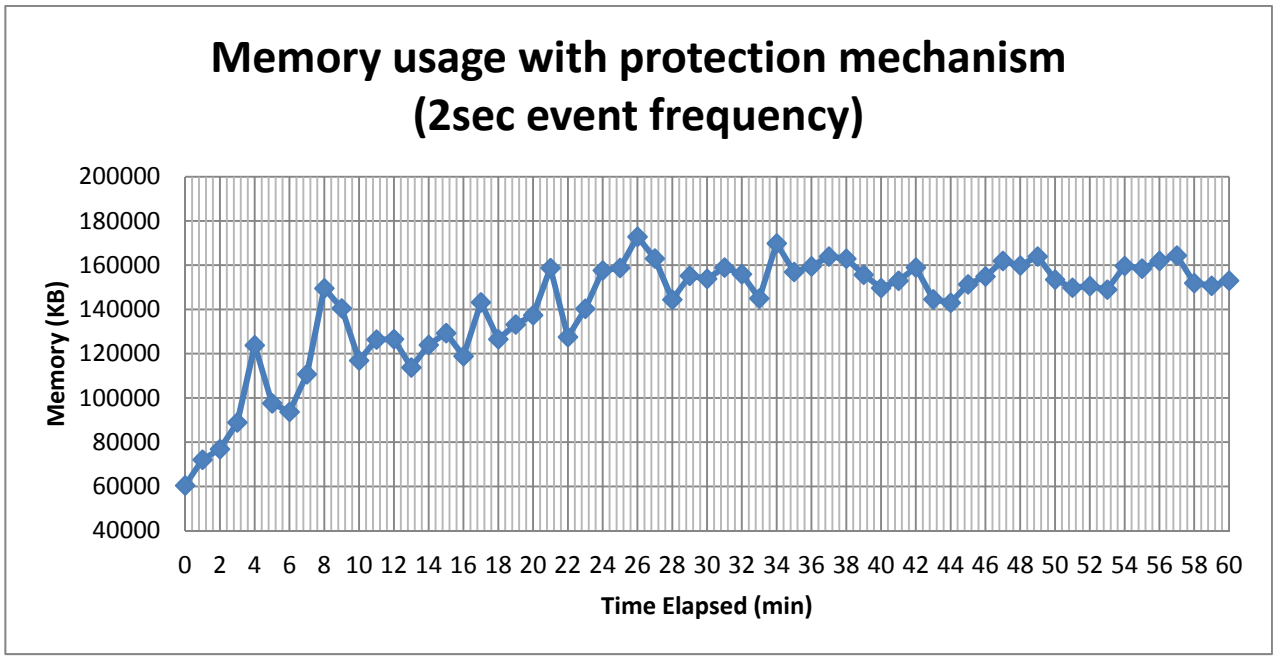
Εικόνα 39: Γραφική 6, Μέγεθος rwArray σε Bytes

Η δεύτερη γραφική παράσταση (γραφική 7) μας δείχνει τη κατανάλωση μνήμης, όμοια με τις γραφικές 1,2 και 3. Η μόνη διαφορά σε σχέση με τις τρεις αυτές γραφικές είναι ότι η καταγραφή των στατιστικών γινόταν κάθε λεπτό αντί

κάθε 5 λεπτά για να παρατηρηθούν με μεγαλύτερη ακρίβεια οι μεταβολές στη κατανάλωση μνήμης.

Time Elapsed (min)	Memory (KB)
0	60444
1	72092
2	76884
3	88908
4	123816
5	97588
6	93708
7	110732
8	149572
9	140516
10	116932
11	126384
12	126594
13	113776
14	123976
15	129324
16	118876
17	143222
18	126554
19	133226
20	137427
21	158745
22	127540
23	140333
24	157662
25	158792
26	172876
27	163040
28	144453
29	155232
30	153875
31	158957
32	155956
33	144965
34	169864
35	156960
36	159485
37	163949
38	162959
39	155697

40	149686
41	152986
42	158965
43	144595
44	143059
45	151349
46	154934
47	161949
48	159864
49	163959
50	153495
51	149858
52	150596
53	148958
54	159695
55	158485
56	161959
57	164368
58	151949
59	150697
60	153040



Εικόνα 40: Γραφική 7, Memory usage (event frequency 2sec)

Παρατηρήσεις

Στις μετρήσεις αυτές θέλαμε να δούμε τη κατανάλωση μνήμης σε πολύ υψηλή καταπόνηση με τη διαδικτυακή υπηρεσία να διαθέτει μηχανισμό προστασίας. Για τον λόγο αυτό χρησιμοποιήσαμε μηχανισμό παραγωγής και προώθησης event με συχνότητα αποστολής event στα 2sec και διατηρούσαμε ratio που άγγιζε το 93%.

Από τη γραφική 6 παρατηρούμε τη μεταβολή στο μέγεθος του πίνακα *rwArray*. Ο *rwArray* αποθηκεύει τα αντικείμενα που έχει προσπελάσει ο χρήστης στο νέφος.

Η γραφική 7 είναι αυτή που μας ενδιαφέρει. Εδώ παρουσιάζεται η συνολική μνήμη που καταναλώνει η υπηρεσία και παρατηρούμε την επίδραση που έχει η μεταβολή του μεγέθους του *rwArray* στις ανάγκες για μνήμη. Στη γραφική 2, αλλά και στις γραφικές 1 και 3 σε μικρότερο βαθμό, παρατηρούμε ότι όσο περνάει ο χρόνος βλέπουμε μια συνεχής αύξηση στις ανάγκες για μνήμη. Στη γραφική 7 όπου η διαδικτυακή υπηρεσία διαθέτει μηχανισμό προστασίας, παρατηρούμε ότι μετά από ένα διάστημα αρχικοποίησης με συνεχής αύξηση στις ανάγκες για μνήμη, βλέπουμε ότι η κατανάλωση μνήμης σταθεροποιείται και μεταβάλλεται εντός του διαστήματος των 140-160MB με μέσο όρο τα 154MB. Η σταθεροποίηση αυτή της κατανάλωσης μνήμης μας δείχνει πόσο πολύ ήταν αναγκαίος ο μηχανισμός προστασίας κατανάλωσης μνήμης στη πλευρά του πελάτη.

Πιθανές επεκτάσεις

Παρακάτω παρουσιάζονται τρεις ενδιαφέρον πιθανές επεκτάσεις στη διαδικτυακή υπηρεσία. Οι επεκτάσεις αυτές θα βοηθήσουν έτσι ώστε να βελτιωθεί η παρουσίαση των στατιστικών αλλά και να αυξηθούν οι επιλογές που θα έχει στα χέρια του ο χρήστης έτσι ώστε να βελτιωθεί ακόμη περισσότερο το monitoring:

1η Επέκταση: *Αποθήκευση στατιστικών τοπικά στη πλευρά του πελάτη*

Μια πιθανή επέκταση στη διαδικτυακή υπηρεσία είναι η καταγραφή των στατιστικών και η αποθήκευση τους σε αρχείο τοπικά στον υπολογιστή του χρήστη.

Η καταγραφή αυτή μπορεί να γίνει προσθέτοντας ένα κουμπί REC στη εφαρμογή για έναρξη της συλλογής στατιστικών ενώ η ολοκλήρωση να γίνεται με ακόμη ένα πάτημα του.

Η αποθήκευση των στατιστικών που μαζεύουμε από τη *EventPage* της εφαρμογής θα γίνεται τοπικά στον υπολογιστή του χρήστη χωρίς τη αποστολή αιτημάτων προς το σύννεφο. Αυτό είναι δυνατόν μέσω του *FileSystem API* που ανήκει στη HTML5 και μας επιτρέπει μέσω JavaScript να διαχειριζόμαστε αρχεία. Το *FileSystem API* είναι καινούργιο και το μεγαλύτερο μειονέκτημα του και ο βασικός λόγος που δεν χρησιμοποιήθηκε στη έκδοση αυτή της διαδικτυακής υπηρεσίας είναι το γεγονός ότι υποστηρίζεται μόνο από τον περιηγητή *Google Chrome* (και όχι πλήρως με προβλήματα ασφάλειας) και έτσι δεν πληρείται η μη λειτουργική απαίτηση για *cross-browser compatibility*.

2η Επέκταση: *Διαχείριση ουρών*

Μια δεύτερη πιθανή επέκταση της διαδικτυακή υπηρεσίας είναι η προσθήκη σελίδας διαχείρισης ουρών που ανήκουν στον χρήστη και η προσθήκη των ανάλογων λειτουργιών.

Στη τρέχουσα έκδοση της διαδικτυακής υπηρεσίας υπάρχει μόνο η δυνατότητα δημιουργίας ουράς. Με τη προσθήκη σελίδας διαχείρισης ουρών θα μπορούσαν να προστεθούν δυνατότητες όπως η προβολή όλων των ουρών του χρήστη, η διαγραφή όλων των ουρών, η διαγραφή συγκεκριμένων ουρών και άλλες πιθανές λειτουργίες.

Για να είναι δυνατή η προσθήκη των λειτουργικοτήτων αυτών θα μπορούσε να προστεθεί ένα ενδιάμεσο επίπεδο μεταξύ περιηγητή χρήστη και νέφους έτσι ώστε τα αιτήματα ουρών να φθάνουν πρώτα εκεί. Βασική προϋπόθεση είναι τα αιτήματα να είναι RESTful. Παραδείγματα αιτημάτων θα μπορούσαν να είναι τα παρακάτω:

Πίνακας 26: Πιθ. Επεκτάσεις, Αίτηση λίστας ουρών που ανήκουν στον χρήστη

```
GET /MyContainer/queues?user="CURRENT_USER" HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-queue
X-CDMI-Specification-Version: 1.0.1
```

Πίνακας 27: Πιθ. Επεκτάσεις, Αίτηση διαγραφής όλων των ουρών που ανήκουν στον χρήστη

```
DELETE /MyContainer/queues?user="CURRENT_USER" HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-queue
X-CDMI-Specification-Version: 1.0.1
```

Πίνακας 28: Πιθ. Επεκτάσεις, Αίτηση διαγραφής μιας ουράς που ανήκει στον χρήστη

```
DELETE /MyContainer/queues/QUEUE_TO_DELETE?user="CURRENT_USER"
HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-queue
X-CDMI-Specification-Version: 1.0.1
```

3η Επέκταση: *Ειδικές Ουρές*

Ως συνέχεια της δεύτερης πιθανής επέκτασης της διαδικτυακής υπηρεσίας θα μπορούσε να είναι η κατηγοριοποίηση της δημιουργίας ουράς.

Έτσι αντί να υπάρχει μόνο η δυνατότητα δημιουργίας απλής ουράς να μπορεί ο χρήστης για παράδειγμα, να δημιουργεί *READ ONLY* ουρές όπου μας ενδιαφέρουν μόνο READS, *WRITE ONLY* ουρές όπου μας ενδιαφέρουν μόνο WRITES, ουρές που θα μπορούν να έχουν πρόσβαση ομάδα χρηστών και όχι μόνο ο δημιουργός της ουράς κ.α.

Παράρτημα

Στο παράρτημα αυτό παρουσιάζεται ο κώδικας της διαδικτυακής υπηρεσίας. Αρχικά παρουσιάζεται ο κώδικας και τα JavaScript scripts της σελίδας επαλήθευσης στοιχείων, ακολούθως της σελίδας δημιουργίας ουράς και τέλος της σελίδας παρακολούθησης event. Στο παράρτημα δεν βρίσκεται ο κώδικας των αρχείων του Rgraph και του JQuery που χρησιμοποιήθηκαν και ακόμη παραλείπονται τα css αρχεία.

Κώδικας σελίδας επαλήθευσης στοιχείων χρήστη

Authenticate.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
   <title>Vision Cloud Event Monitor - Authenticate</title>

   <!-- jquery -->
   <script language="javascript" type="text/javascript"
                                          src="scripts/jquery-1.7.min.js"></script>
   <!-- base64 -->
   <script language="javascript" type="text/javascript"
                                          src="scripts/jquery.base64.min.js"></script>
   <!-- my javascript source -->
   <script language="javascript" type="text/javascript"
                            src="scripts/authenticateScripts/authenticate.js"></script>
   <script language="javascript" type="text/javascript"
                                          src="scripts/utilities.js"></script>
   <!-- css -->
   <link rel="stylesheet" type="text/css" href="css/AuthenticateStyle.css" />
</head>
<body>
   <div id="container">
      <div id="header">
         <div id="logo">
             <a href="Authenticate.jsp"><img src="images/vision_logo.png" height="70"
                                       width="100" alt="vision logo" border="0" /></a>
         </div>
         <div id="header_title">
            <h2>Vision Cloud Event Monitor</h2>
         </div>
         <div class="seperator"></div>
     </div><!-- header -->

     <div id="statusbar" align="right">
        <table>
            <tr>
              <td>Time:</td><td id="clockid"></td>
```

```
            </tr>
        </table>
    </div><!-- statusbar -->

    <div id="mainPanel">
        <div id="leftPanel">
            <p>You must have a valid Vision Cloud ID to use this web service.</p>
            <p>Log-in using your Vision Cloud Username, Tenant and Password.</p> <br />
            <p>Browser Support<br />
            <img src="images/compatible_chrome.gif" alt="chrome compatible"
                                                    style="padding:5px 2px 0px 0px" />
            <img src="images/compatible_firefox.gif" alt="firefox compatible"
                                                    style="padding:5px 2px 0px 0px" />
            <img src="images/incompatible_ie.gif" alt="Internet Explorer compatible"
                                                    style="padding:5px 2px 0px 0px" />
            </p>
        </div><!-- leftPanel -->

        <div id="rightPanel" align="center">
            <div id="loginForm">
                <div id="msgDiv">
                    <div id="errorImage">
                        <img src="images/login_icon.png" height="45px" width="45px"
                                                        alt=login_icon />
                    </div>
                    <div id="msg"><h3>Log-in</h3></div>
                    <div class="seperator"></div>
                </div><!-- msgDiv -->

            <table>
                <tr>
                  <td><label>Username</label></td>
                  <td><input type="text" id="user"/></td>
                </tr>
                <tr>
                  <td><label>Tenant</label></td>
                  <td><input type="text" id="tenant"/></td>
                </tr>
                <tr>
                  <td><label>Password</label></td>
                  <td><input type="password" id="pass"/></td>
                </tr>
                <tr>
                  <td colspan="2" align="center">
                    <input type="button" value="Sign-in" id="authenticateButton"/>
                    <input type="button" value="Clear" id="clearButton"/>
                  </td>
                </tr>
            </table>
          </div><!-- loginForm -->

      </div><!-- rightPanel -->
      <div class="seperator"></div>
   </div><!-- mainPanel -->

   <div id="footer">
       Designed for the <a href="http://grid.ece.ntua.gr/" target="_blank">Distributed
      Knowledge and Media Group</a> by Demetris Trihinas as part of his Diploma Thesis.
      <br />Special thanks to the <a href="http://www.visioncloud.eu/"
                      target="_blank">Vision Cloud</a> and <a href="http://rgraph.net/"
                      target="_blank">Rgraph</a>
   </div><!-- footer -->
 </div><!-- container -->
</body>
</html>
```

authenticate.js

```
$(document).ready(function() {
	$("#authenticateButton").click(authFunc);
	$("#clearButton").click(clearFunc);

	//clock on the status bar
	window.setInterval(
		function(){
			$("#clockid").html(clock()); // clock() in utilities.js
		}, 1000
	);
});

/*
 * this is the most important function of the authenticate mechanism.
 * We first check for sessionStorage support. After we get the users credentials
 * inserted by the user and make an AJAX call to the cloud following the specified
 * from the RESTful Vision user API format for authenticating a user.
 */
function authFunc(){
	if(!checkSessionStorageSupport()){
		alert("Error: Your browser does not support HTML5 sessionStorage");
		return;
    }
	if (typeof(sessionStorage.login_attempt)=='undefined'){
		sessionStorage.setItem('login_attempt',1);
	}

	var username=$("#user").val();
	var tenant=$("#tenant").val();
	var password=$("#pass").val();
	var base64=$.base64.encode(username+"@"+tenant+":"+password);

	$.ajax({
		type: "get",
		url: "authenticate/",
		beforeSend: function(xhr) {
			xhr.setRequestHeader("Accept", "application/json");
		  xhr.setRequestHeader("Authorization", "Basic " + base64);
		},
		dataType: "json",
		success: function(jsonData){
					showResults(jsonData,base64);
				},
		statusCode: {
			401 : unauthorizedResponse
		}
	});
}

/*
 *  this function checks the users browser to see if it supports
 *  HTML5 sessionStorage.
 */
function checkSessionStorageSupport(){
	//ex. IE8 as browser
	if (typeof(sessionStorage)=='undefined'){
		$("#msg").html("Cannot proceed\nYour browser does not support HTML5
                                                                sessionStorage."
                    +" If you are using Mozilla Firefox or Google Chrome, please
                     update your browser");
		return false;
	}
	return true;
}
```

```
/*
 *  this function is called when the cloud returns an 200 OK
 *  http status, meaning the user is authenticated. We then save
 *  in the sessionStorage the credentials of the user
 *  and redirect the user to the CreateQueue.jsp page
 */
function showResults(jsonData,base64){
	var username=jsonData.userName;
	var tenant=jsonData.tenantName;

	sessionStorage.setItem('username',username);
	sessionStorage.setItem('tenant',tenant);
	sessionStorage.setItem('auth',base64);
	sessionStorage.removeItem("login_attempt");

	document.location.href='CreateQueue.jsp';
}

/*
 * this function is called when the cloud returns 401 Unauthorized
 * http status.
 */
function unauthorizedResponse(){
	if(++sessionStorage.login_attempt>3)
		document.location.href='loginErrorPage.html';
	else{
		clearFunc();
		$("#errorImage").html("<img src=\"images/error_icon.png\" height=\"45px\"
			width=\"45px\" alt=error_icon />");
		$("#msg").html("<h3 style=\"color:red\">Username, Tenant and/or Password
			Invalid<br /></h3>");
		alert("Username, Tenant and/or Password invalid\nPlease try again");
	}
}

/*
 * this fuction when called clears the fields of the loginForm
 */
function clearFunc(){
	$("#user").val('');
	$("#tenant").val('');
	$("#pass").val('');
}
```

loginErrorPage.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
	<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
	<title>Vision Cloud Event Monitor - Login Error</title>
</head>

<body>
	<h1>Login Error</h1>
	<h2>You have attempted to log-in 3 times</h2>
</body>
</html>
```

utilities.js

```
/*
 * this function returns an array of
 * the parameters that are contained in a URL address
 */
function getURLParams() {
    var urlArray2={};
       var e,
        a = /\+/g,  // Regex for replacing addition symbol with a space
        r = /([^&=]+)=?([^&]*)/g,
        d = function (s) { return decodeURIComponent(s.replace(a, " ")); },
        q = window.location.search.substring(1);

    while (e = r.exec(q))
       urlArray2[d(e[1])] = d(e[2]);
    return urlArray2;
}

/*
 * this function is used as a clock
 */
function clock() {
   var now = new Date();
   var minutes=now.getMinutes();
   var seconds=now.getSeconds();
   if (minutes < 10)
          minutes = "0" + minutes;

   if (seconds < 10)
          seconds = "0" + seconds;

   var outStr = now.getHours()+':'+minutes+':'+seconds;
   return outStr;
}

/*
 * this function returns the string representation of a Unix timestamp
 */
function timeConverter(timestamp){
      var a = new Date(timestamp);
      var months =
['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'];
      var year = a.getFullYear();
      var month = months[a.getMonth()];
      var date = a.getDate();
      var hour = a.getHours();
      var min = a.getMinutes();
      if (min < 10)
               min = "0" + min;
      var sec = a.getSeconds();
      if (sec < 10)
               sec = "0" + sec;
      var time = date+' '+month+' '+year+' '+hour+':'+min+':'+sec ;
      return time;
}
```

Κώδικας σελίδας δημιουργίας ουράς

CreateQueue.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
       <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
       <title>Vision Cloud Event Monitor - Create Queue</title>
       <!-- jquery -->
       <script language="javascript" type="text/javascript" src="scripts/jquery-
                                                              1.7.min.js"></script>
       <!-- my javascript source -->
       <script language="javascript" type="text/javascript"
                                 src="scripts/createQueueScripts/createQueue.js"></script>
       <script language="javascript" type="text/javascript"
                                 src="scripts/utilities.js"></script>
       <!-- css -->
       <link rel="stylesheet" type="text/css" href="css/CreateQueueStyle.css" />
</head>
<body>
   <div id="container">
       <div id="header">
           <div id="logo">
              <a href="Authenticate.jsp"><img src="images/vision_logo.png" height="70"
                                    width="100" alt="vision logo" border="0" /></a>
           </div>
           <div id="header_title">
              <h2>Vision Cloud Event Monitor</h2>
           </div>
           <div class="seperator"></div>
       </div><!-- header -->

       <div id="statusbar" align="right">
          <table>
             <tr>
               <td>Username:</td><td id="user"></td><td class="st_sep"> </td>
               <td>Tenant:</td><td id="tenant"></td><td class="st_sep"> </td>
               <td>Time:</td><td id="clockid"></td>
             </tr>
          </table>
       </div><!-- statusbar -->

       <div id="mainPanel">
           <div id="leftPanel">
              <p>Enter a unique name in the "queue name" text field</p>
              <p>If there is already a queue with the given name, a conflict error will
                 occur and you will be asked to enter another name.</p>    <br />
              <p>Browser Support<br />
                 <img src="images/compatible_chrome.gif" alt="chrome compatible"
                                    style="padding:5px 2px 0px 0px" />
                 <img src="images/compatible_firefox.gif" alt="firefox compatible"
                                    style="padding:5px 2px 0px 0px" />
                 <img src="images/incompatible_ie.gif" alt="Internet Explorer compatible"
                                    style="padding:5px 2px 0px 0px" />
              </p>
            </div><!-- leftPanel -->

            <div id="rightPanel">
               <div id="inputForm">
                   <div id="msgDiv">
```

```
                <div id="errorImage">
                    <img src="images/queue_icon.jpg" height="45px" width="45px"
                                                            alt=queue_icon />
                </div>
                <div id="msg"><h3>Give a name for the queue</h3></div>
                <div class="seperator"></div>
            </div><!-- msgDiv -->
            <table  align="center">
                <tr>
                    <td><input type="text" id="queueNameField"/></td>
                    <td><input type="button" value="Create"
                                                id="createQueueButton"/></td>
                     <td><input type="reset" value="Reset"/></td>
                </tr>
            </table>
            </div><!-- inputQueue -->
                <h3>Console</h3>
                <div id="console">
                </div>
            <div id="viewEventsDiv">
                    <h3>Proceed To View Events</h3>
                    <div id="ve">
                    <input type="button" value="View Events" id="viewEventsButton"/>
                    </div>
            </div>
        </div><!-- rightPanel -->
        <div class="seperator"></div>
    </div><!-- mainPanel -->

    <div id="footer">
        Designed for the <a href="http://grid.ece.ntua.gr/" target="_blank">Distributed
       Knowledge and Media Group</a> by Demetris Trihinas as part of his Diploma Thesis.
       <br />Special thanks to the <a href="http://www.visioncloud.eu/"
                        target="_blank">Vision Cloud</a> and <a href="http://rgraph.net/"
                        target="_blank">Rgraph</a>
    </div><!-- footer -->
  </div><!-- container -->
</body>
</html>
```

CreateQueue.js

```
$(document).ready(function() {
	$("#createQueueButton").click(ajaxCall);
	$("#viewEventsDiv").hide();
	$("#viewEventsButton").click(viewFunc);

	/*if the user did not logg in successfully or navigated
	  direct to this page then we redirect him to the Authenticate.jsp page*/
	if(!checkIfLoggedIn()){
		alert("Please Log-in first");
		document.location.href="Authenticate.jsp";
	}

	//clock on the status bar
	window.setInterval(
		function(){
			$("#clockid").html(clock()); // clock() in utilities.js
		}, 1000
	);

	$("#user").html(getUsername());
	$("#tenant").html(getTenant());

	$("#console").append("Creating Queue Console<br />======================<br />
				<br />");

});

/**
 * this function checks the sessionStorage to see
 * if username and the auth variables are defined.
 * @returns {Boolean}
 */
function checkIfLoggedIn(){
	if (typeof(sessionStorage.username)=='undefined'){
		return false;
	}
	if (typeof(sessionStorage.auth)=='undefined'){
		return false;
	}
	return true;
}

/**
 * get the username of the user from the sessionStorage
 * @returns void
 */
function getUsername(){
	if (typeof(sessionStorage.username)!='undefined')
		return sessionStorage.username;
}

/**
 * get the tenant of the user from the sessionStorage
 * @returns void
 */
function getTenant(){
	if (typeof(sessionStorage.tenant)!='undefined')
		return sessionStorage.tenant;
}

/**
 * this function is the most import in the CreateQueue.jsp page.
 * first we get the queueName the user inserted. After we send
```

```
 * HTTP PUT request to cloud for creating a queue and process the response.
 */
function ajaxCall() {
	var queueName=$("#queueNameField").val();
	if (queueName==""){
		$("#errorImage").html("<img src=\"images/error_icon.png\" height=\"45px\"
width=\"45px\" alt=error_icon />");
		alert("Please give a name to the queue");
		return;
	}

	//base64 authentication stored in sessionStorage
	var auth=sessionStorage.auth;

	/*
	 * these headers are mandatory
	 * 1. Accept=application/cdmi-queue
	 * 2. X-CDMI-Specification-Version
	 * 3. Content-Type=application/cdmi-queue
	 *
	 * 201: if the queue is succesfully created
	 * 409: if the queue name given already exists
	 * 403: if user does not have permission to create queues.
	 */
	var metadata={};
	$.ajax({
		type: "put",
		url: "queues/"+queueName,
		beforeSend: function(req) {
			req.setRequestHeader("Accept", "application/cdmi-queue");
		 req.setRequestHeader("X-CDMI-Specification-Version", "1.0.1");
		 req.setRequestHeader("Authorization", "Basic " + auth);
	    },
		contentType: "application/cdmi-queue",
		dataType: "json",
		data:metadata,
		success: showResults,
		statusCode: {
			409: conflictResponse,
			400: badRequestResponse,
			401: unauthorizedResponse,
			403: forbiddenResponse,
			404: notfoundResponse,
		  },
	});
}

/**
 * this function is called if the cloud returned a 201 Created http status code.
 * We process the response by parsing the body as a JSON string and the obtained details
 * of the queue we print to the console.
 * @param jsonData
 */
function showResults(jsonData){
	alert("Creating Queue Successfull");
	var meta=extractMetadata(jsonData.metadata);
	var queueName=$("#queueNameField").val();

	var s="Response Status: 201 Created<br />"
		 +"Queue "+queueName+" Created Successful<br />"
		 +"Queue Details<br />"
		 +"-------------<br />"
		 +"objectType: "+jsonData.objectType+"<br />"
		 +"objectID: "+jsonData.objectID+"<br />"
		 +"objectName: "+jsonData.objectName+"<br />"
		 +"parentURI: "+jsonData.parentURI+"<br />"
		 +"parentID: "+jsonData.parentID+"<br />"
		 +"domainURI: "+jsonData.domainURI+"<br />"
```

```
                +"capabilitiesURI: "+jsonData.capabilitiesURI+"<br />"
                +"completionStatus: "+jsonData.completionStatus+"<br />"
                +"metadata: "+meta+"<br />"
                +"queueValues: "+jsonData.queueValues+"<br /><br />";
        $("#console").append(s);
        $("#console").animate({ scrollTop: $("#console").prop("scrollHeight") -
                                $('#console').height() }, 1000);

        $("#viewEventsDiv").show();
}

/**
 * this function is called to parse the metadata
 * @param group
 * @returns {String}
 */
function extractMetadata(group){
        var st="";
        for (var name in group) {
            if (group.hasOwnProperty(name)) {
                var value = group[name];

                st=st+"<br />\""+name+"\""+":"+"\""+value+"\"";
            }
        }
        return st;
}

/**
 * this function is called if the cloud returned a 409 Conflict http status code.
 */
function conflictResponse(){
        $("#errorImage").html("<img src=\"images/error_icon.png\" height=\"45px\"
width=\"45px\" alt=error_icon />");
        $("#msg").html("<h3 style=\"color:red\">The queue name is already in use</h3>");
        alert("409 Conflict Error\n The queue name is already in use");
        var queueName=$("#queueNameField").val();
        var s="Response Status: 409 Conflict<br />"
                +"Queue Name: "+queueName+" is already in use<br />"
                +"-------------<br /><br />";
        $("#console").append(s);
        $("#console").animate({ scrollTop: $("#console").prop("scrollHeight") -
                                $('#console').height() }, 1000);
}

/**
 * this function is called if the cloud returned a 400 Bad request http status code.
 */
function badRequestResponse(){
        $("#errorImage").html("<img src=\"images/error_icon.png\" height=\"45px\"
width=\"45px\" alt=error_icon />");
        $("#msg").html("<h3 style=\"color:red\">Bad Request Error</h3>");
        alert("400 Bad Request Error\n");
        var s="Response Status: 400 Bad Request<br />"
                +"-------------<br /><br />";
        $("#console").append(s);
        $("#console").animate({ scrollTop: $("#console").prop("scrollHeight") -
                                $('#console').height() }, 1000);
}

/**
 * this function is called if the cloud returned a 401 Unauthorized http status code.
 */
function unauthorizedResponse(){
        alert("401 unauthorized Error\n username and/or password invalid\n");
        document.location.href="Authenticate.jsp";
}
```

```
/**
 * this function is called if the cloud returned a 403 Forbidden http status code.
 * Redirect user to Authenticate.jsp page
 */
function forbiddenResponse(){
	alert("403 forbidden Error\n Client lacks the proper authorization to perform
		this request\nSign-in using another account with proper authorization");
	document.location.href="Authenticate.jsp";
}

/**
 * this function is called if the cloud returned a 404 Not Found http status code.
 */
function notfoundResponse(){
	$("#errorImage").html("<img src=\"images/error_icon.png\" height=\"45px\"
width=\"45px\" alt=error_icon />");
	$("#msg").html("<h3 style=\"color:red\">Bad Request Error</h3>");
	alert("404 Not Found Error\n");
	var s="Response Status: 404 Not Found<br />"
		+"-------------<br /><br />";
	$("#console").append(s);
	$("#console").animate({ scrollTop: $("#console").prop("scrollHeight") -
				$('#console').height() }, 1000);
}

/**
 * this function is called to redirect user to EventPage.jsp page
 */
function viewFunc(){
	var queueName=$("#queueNameField").val();
	document.location.href='EventPage.jsp?queueName='+queueName;
}
```

Σελίδα παρακολούθησης γεγονότων

EventPage.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
       <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
       <title>Showing Events</title>
       <!-- jquery -->
       <script language="javascript" type="text/javascript" src="scripts/jquery-
                                                                  1.7.min.js"></script>
       <!-- my javascript source -->
       <script language="javascript" type="text/javascript"
                                  src="scripts/eventPageScripts/eventPage.js"></script>
       <script language="javascript" type="text/javascript"
                                  src="scripts/utilities.js"></script>
       <script language="javascript" type="text/javascript"
                                  src="scripts/eventPageScripts/charttools.js"></script>
       <script language="javascript" type="text/javascript"
                                  src="scripts/eventPageScripts/eventTable.js"></script>
       <script language="javascript" type="text/javascript"
                                  src="scripts/eventPageScripts/Graph.js"></script>
       <script language="javascript" type="text/javascript"
                               src="scripts/eventPageScripts/processRawEvent.js"></script>
       <script language="javascript" type="text/javascript"
                                  src="scripts/eventPageScripts/StatsObj.js"></script>
       <!-- rgraph -->
       <script language="javascript" type="text/javascript"
                                  src="scripts/rgraph/RGraph.common.core.js"></script>
       <script language="javascript" type="text/javascript"
                                  src="scripts/rgraph/RGraph.bar.js"></script>
       <script language="javascript" type="text/javascript"
                                  src="scripts/rgraph/RGraph.common.key.js"></script>
       <script language="javascript" type="text/javascript"
                                src="scripts/rgraph/RGraph.common.tooltips.js"></script>
       <script language="javascript" type="text/javascript"
                                 src="scripts/rgraph/RGraph.common.dynamic.js"></script>
       <!-- css -->
       <link rel="stylesheet" type="text/css" href="css/EventPageStyle.css" />
</head>
<body>
   <div id="container">
       <div id="header">
          <div id="logo">
              <a href="Authenticate.jsp"><img src="images/vision_logo.png" height="70"
                                       width="100" alt="vision logo" border="0" /></a>
          </div>
          <div id="header_title">
              <h2>Vision Cloud Event Monitor - EventPage.jsp</h2>
          </div>
          <div class="seperator"></div>
       </div><!-- header -->

       <div id="statusbar" align="right">
          <table>
              <tr>
               <td>Username:</td><td id="user"></td><td class="st_sep"> </td>
              <td>Tenant:</td><td id="tenant"></td><td class="st_sep"> </td>
```

```
            <td>Events (Received):</td><td id="eventsRec"></td><td class="st_sep">
            </td>
            <td>Events (Processed):</td><td id="eventsProc"></td><td class="st_sep">
            </td>
            <td>Ratio:</td><td id="ratio"></td><td class="st_sep"> </td>
            <td>Objects (Processed):</td><td id="objectsProc"></td><td class="st_sep">
             </td>
            <td>Time:</td><td id="clockid"></td>
            </tr>
        </table>
    </div>

    <div id="topPanel">
        <div id="topPanelEvents">
            <fieldset>
              <legend>Events</legend>
              <div id="showEventsDiv">
              </div><!-- showEventsDiv -->
              <div id="stopEventsDiv">
                   <input type="button" value="Stop Events" id="stopEventsButton"/>
                   <input type="button" value="Create New Stream" id="refreshButton"/>
              </div>
           </fieldset>
        </div><!-- topPanelEvents -->

        <div id="topPanelStats">
            <fieldset>
               <legend>Stats</legend>
               <div id="stats">
                   <table id="statsTable"></table>
               </div>    <!-- stats -->
            </fieldset>
        </div><!-- topPanelStats -->
    </div><!-- topPanel -->

    <div class="seperator"></div>

    <div id="bottomPanel">
        <div id="graph">
            <fieldset>
                <legend>Graph</legend>
                <div id="chartdiv">
                    <canvas id="mainchart" width="555" height="350">[No canvas
                                                                    support]</canvas>
                     <div id=chartmsg></div>
                     <div id="chartnavdiv">
                          <input type="button" value="Prev" id="previousButton" />
                          <input type="button" value="Next" id="nextButton" />
                     </div>
                </div>
                <div id="zoomdiv">
                   <table>
                      <tr><td style="font-size:small">zoom</td></tr>
                      <tr><td><input type="button" value="+" id="zoomInc" /></td></tr>
                      <tr><td id="zoomDisplayText" align="center"></td></tr>
                      <tr><td><input type="button" value="-" id="zoomDec" /></td></tr>
                   </table>
               </div>
            </fieldset>
        </div><!-- graph -->

        <div id="sortGraphsDiv">
            <fieldset>
                 <legend>Top Reads</legend>
                <div id="readsSortGraph">
                   <canvas id="readchart" width="445" height="200">[No canvas
                                                                   support]</canvas>
                </div>
```

```
            </fieldset>

             <fieldset>
                 <legend>Top Writes</legend>
                 <div id="writesSortGraph">
                    <canvas id="writechart" width="445" height="200">[No canvas
                                                                    support]</canvas>
                 </div>
             </fieldset>
         </div><!-- sortGraphsDiv -->
     </div><!-- bottomPanel -->

     <div class="seperator"></div>

     <div id="footer">
       Designed for the <a href="http://grid.ece.ntua.gr/" target="_blank">Distributed
      Knowledge and Media Group</a> by Demetris Trihinas as part of his Diploma Thesis.
      <br />Special thanks to the <a href="http://www.visioncloud.eu/"
                      target="_blank">Vision Cloud</a> and <a href="http://rgraph.net/"
                      target="_blank">Rgraph</a>
     </div><!-- footer -->

   </div><!-- container -->
</body>
</html>
```

eventPage.js

```
$(document).ready(function() {
	//get the url parameters
	var urlParams = getURLParams();   //getURLParams() in utilities.js
	//get the queue name
	var queueName=urlParams["queueName"];
	if (queueName==null){
		alert("Could not get queueName.\nPlease Log-in and create a queue first");
		document.location.href="Authenticate.jsp";
		return;
	}

	/*
	 * events received, events Processed, objects processed variables
	 * shown in the status bar
	 */
	var eventsRec=0;
	var eventsProc=0;
	var objectsProc=0;

	//the source of the events
	var eSource = new EventSource("getEvents1?queueName="+queueName);

	eSource.onopen =
		function () {
			console.log("opened stream");
			alert("new stream opened");
		};

	//new event has arrived. JSON parse it and process it.
	eSource.onmessage =
		function (event) {
			eventCounterIncrement(++eventsRec,"#eventsRec");
			console.log("new event: " + event.data);
			var jsonEvent =JSON.parse(event.data);

			var username=getUsername();
			var tenant=getTenant();

			var myEvents=getmyEvents(jsonEvent,username,tenant);

			/*
			 * if event is relevant to the user
			 */
			if (myEvents.length>0){
				eventCounterIncrement(++eventsProc,"#eventsProc");
				objectsProc=objectsProc+myEvents.length;
				eventCounterIncrement(objectsProc,"#objectsProc");
				console.log("this event contains user: "+username+" objects
					that need to be calculated");

				//trigger the 2 binded dom objects
				$("#showEventsDiv").trigger("eventTableEvt",
					{Param1:jsonEvent,Param2:myEvents});
				$("#chartdiv").trigger("chartdivEvt",{Param1:myEvents});
			}
			else
				console.log("this event does not contain user: "+username+"
					objects");

			eventCounterIncrement((eventsProc/eventsRec*100).toFixed(2)
				+"%","#ratio");
		};
```

```
	eSource.onerror =
		function () {
			console.log("error");
		};

	//in StatsObj.js
	var statsObj=new StatsObj();
	//in Graph.js
	var graph=new Graph(statsObj);

	//called when a new event arrives - in eventTable.js
	$("#showEventsDiv").bind("eventTableEvt",
		function(e,par){
			var jsonEvent=par.Param1;
			var myEvents=par.Param2;

			var len=$('#showEventsDiv').children().length;
			if (len>20)
				$('#showEventsDiv').find("div:lt(1)").html("");

			$("#showEventsDiv").append(
					addEventToConsole(jsonEvent,myEvents,eventsProc));
			$("#showEventsDiv").animate({ scrollTop:
					$("#showEventsDiv").prop("scrollHeight")
				- $('#showEventsDiv').height() }, 1000);
		}
	);


	//called when a new event arrives - in graph.js
	$("#chartdiv").bind("chartdivEvt",
		function(e,par){
			var myEvents=par.Param1;
			graph.update(myEvents);
		}
	);

	//basic chart tools - in charttools.js
	$("#nextButton").click(function(){
		nextButtonFunc(graph);
	});
	$("#previousButton").click(function(){
		previousButtonFunc(graph);
	});

	$("#zoomInc").click(function(){
		zoomIncFunc(graph);
	});
	$("#zoomDec").click(function(){
		zoomDecFunc(graph);
	});

	$("#zoomDisplayText").html(graph.statsObj.shifter);

	//close the incoming event stream
	$("#stopEventsButton").click(
		function(){
			eSource.close();
			console.log("closed stream");
		}
	);

	//refresh EventPage to start a new stream
	$("#refreshButton").click(
			function(){
				//in case user did not close stream
				eSource.close();
				console.log("closed stream");
```

```
                        document.location.href='EventPage.jsp?queueName='+queueName;
                    }
        );

        //clock on the status bar
        window.setInterval(
                function(){
                        $("#clockid").html(clock()); // clock() in utilities.js
                }, 1000
        );

        $("#statsTable tr").mouseover(function() {
          $("#statsTable tr td").text($(this).attr("title"));
    });

        $("#user").html(getUsername());
        $("#tenant").html(getTenant());
});

/**
 * get the username of the user from the sessionStorage
 */
function getUsername(){
        if (typeof(sessionStorage.username)=='undefined'){
                alert("Please Log-in first");
                document.location.href='Authenticate.jsp';
        }
        return sessionStorage.username;
}

/**
 * get the tenant of the user from the sessionStorage
 */
function getTenant(){
        if (typeof(sessionStorage.tenant)=='undefined'){
                alert("Please Log-in first");
                document.location.href='Authenticate.jsp';
        }
        return sessionStorage.tenant;
}

/**
 * increment the e variable and show it on the EventPage status bar
 * @param e
 * @param id
 * @returns
 */
function eventCounterIncrement(e,id){
        $(id).html(e);
        return e;
}

/**
 * calculate the size of the rwArray by summing up
 * the string representation of each object it contains
 * @param arr
 * @returns
 */
function approxByteSize(arr){
        return arr.map(function(e){return JSON.stringify(e).length;})
                  .reduce(function(x,y){return x+y;});
}
```

#

processRawEvent.js

```
/**
 * this function processes the JSON Event and returns
 * an array of the objects that are relevant to user
 * @param jsonEvent
 * @param username
 * @param tenant
 */
function getmyEvents(jsonEvent,username,tenant){
	var userArray=jsonEvent.users;

	return userArray.filter(
				function(u) {
				  return u.user == username && u.tenant==tenant;
			 })
			.map(function(u) {
			    if (jsonEvent.aggregation=="no of objects read")
					u.operation="read";
			    else u.operation="write";
					return u;
			    }
			);
}
```

eventTable.js

```
//extract the fields from the json object and return a string
function addEventToConsole(jsonEvent,myEvents,eventsProc){
	var timestamp=timeConverter(jsonEvent.timestamp);

	var s= "<div><div class=\"eventDivHeader\">Event Processed #"+eventsProc+"</div>"
	   +"<div class=\"eventDivInfo\"><div class=\"eventDivInfoHead\">Event
Info</div><table>"
	   +"<tr><td>originating_machine</td>"
	   +"<td>"+jsonEvent.originating_machine+"</td></tr>"
	   +"<tr><td>originating_service</td>"
	   +"<td>"+jsonEvent.originating_service+"</td></tr>"
	   +"<tr><td>timestamp</td>"
	   +"<td>"+timestamp+"</td></tr>"
	   +"<tr><td>aggregation</td>"
	   +"<td>"+jsonEvent.aggregation+"</td></tr>"
	   +"<tr><td>topic</td>"
	   +"<td>"+jsonEvent.topic+"</td></tr>"
	   +"<tr><td>cluster</td>"
	   +"<td>"+jsonEvent.cluster+"</td></tr>"
	   +"</table></div>"

	   +"<div class=\"eventDivObjects\"><b>Objects Accessed(URI,Count)</b><table>" ;
	for(var i=0;i<myEvents.length;i++){
	  s= s +"<tr><td>"+myEvents[i].tenant+"."+myEvents[i].user+"."
	       + myEvents[i].container+"."+myEvents[i].object+"</td>"
	   +"<td>"+myEvents[i].count+"</td></tr>";
	}
	s=s+"</table></div><div class=\"seperator\"></div></div>";

	return(s);
}
```

charttools.js

```
/*
 * change value of the rwArrayIndex field,
 * update current arrays and re-draw the graph
 */
function nextButtonFunc(graph){
	var g=graph.statsObj;
	var newIndex=g.rwArrayIndex + g.shifter;

	if(newIndex<g.rwArray.length){
		g.rwArrayIndex=newIndex;
		g.updateCurArrays();
		graph.drawGraph();
	}
	else
		$("#chartmsg").html("These are the most resent events in the list");
}

/*
 * change value of the rwArrayIndex field,
 * update current arrays and re-draw the graph
 */
function previousButtonFunc(graph){
	var g=graph.statsObj;
	var newIndex=g.rwArrayIndex - g.shifter;
	if(newIndex>=0){
		g.rwArrayIndex=newIndex;
	}
	else{
		$("#chartmsg").html("These are the oldest events in the list");
		g.rwArrayIndex=0;
	}
	g.updateCurArrays();
	graph.drawGraph();
}

/*
 * change value of the shifter field, update current arrays
 * and re-draw the graph
 */
function zoomIncFunc(graph){
	var g=graph.statsObj;
	newShifter=g.shifter+1;
	if (newShifter<9){
		console.log("zoom out. going to show "+newShifter+" objects in graph");
		g.shifter=newShifter;
		g.updateCurArrays();
		graph.drawGraph();
		$("#zoomDisplayText").html(g.shifter);
		$("#chartmsg").html("");
	}
	else
		$("#chartmsg").html("Cant show more than 8 events");
}

/*
 * change value of the shifter field, update current arrays
 * and re-draw the graph
 */
function zoomDecFunc(graph){
	var g=graph.statsObj;
	newShifter=g.shifter-1;
	if (newShifter>1){
		console.log("zoom in. going to show "+newShifter+" objects in graph");
		g.shifter=newShifter;
```

```
                g.updateCurArrays();
                graph.drawGraph();
                $("#zoomDisplayText").html(g.shifter);
                $("#chartmsg").html("");
        }
        else
                $("#chartmsg").html("Cant show less than 2 events");
}
```

Graph.js

```
/**
 * the Graph object is responsible for drawing the graphs
 * When a new event arrives, we call the update method.
 * the update method calls the calcStatistics method of the field
 * statsObj so we can process the event. When the processing is over
 * we draw the 3 graphs.
 */
function Graph(statsObj){
	this.statsObj=statsObj;
}

/**
 * this is the public method called when a new event arrives.
 * we also check if there is need to trigger the memory porotection
 * mechanism
 * @param myEvents
 */
Graph.prototype.update=function(myEvents){
	this.statsObj.calcStatistics(myEvents);  //in StatsObj.js
	this.drawGraphs();

	var limit=1000;
	if(this.statsObj.rwArray.length>limit){
		this.statsObj.memoryProtectMechanism(limit);
		this.drawGraphs();
	}

};

/**
 * this method is responsible for drawing the 3 graphs
 */
Graph.prototype.drawGraphs=function(){
	RGraph.Clear(document.getElementById("mainchart"));
	RGraph.Clear(document.getElementById("readchart"));
	RGraph.Clear(document.getElementById("writechart"));

	var data = this.statsObj.curRWArray;
	var categories=this.statsObj.curNamesArray;
	var graph = this.getMainGraph('mainchart', data, categories);
	graph.Draw();

	var sortedReadArray=this.statsObj.sortedReadsArray;
	var sortedWriteArray=this.statsObj.sortedWritesArray;

	for (var i=0;i<sortedWriteArray.length;i++){
	   this.statsObj.sortedReadData[i]= sortedReadArray[i].val;
	   this.statsObj.sortedReadCat[i]= sortedReadArray[i].name;
	   this.statsObj.sortedWriteData[i]= sortedWriteArray[i].val;
	   this.statsObj.sortedWriteCat[i]= sortedWriteArray[i].name;
	}

	var graph2 = this.getReadGraph("readchart", this.statsObj.sortedReadData,
	                                                this.statsObj.sortedReadCat);
	graph2.Draw();

	var graph3 = this.getWriteGraph("writechart", this.statsObj.sortedWriteData,
	                                               this.statsObj.sortedWriteCat);
	graph3.Draw();
};

/**
 * we only create the main graph once and then store it in the
```

```
 * window object.
 */
Graph.prototype.getMainGraph=function(id, data1,categories1){
    if (!window.__rgraph_bar__) {
       window.__rgraph_bar__ = new RGraph.Bar(id, data1);
       window.__rgraph_bar__.Set('chart.labels', categories1);
       window.__rgraph_bar__.Set('chart.tooltips',  function (idx) {return
                                                  categories1[Math.floor(idx/2)];});
       window.__rgraph_bar__.Set('chart.tooltips.effect', 'fade');
       window.__rgraph_bar__.Set('chart.tooltips.event', 'onmousemove');
       window.__rgraph_bar__.Set('chart.key', ['Reads', 'Writes']);
       window.__rgraph_bar__.Set('chart.title', 'Events: Reads & Writes');
       window.__rgraph_bar__.Set('chart.background.barcolor1', 'white');
       window.__rgraph_bar__.Set('chart.background.barcolor2', 'white');
       window.__rgraph_bar__.Set('chart.background.grid', true);
       window.__rgraph_bar__.Set('chart.colors', ['red','blue']);
       window.__rgraph_bar__.Set('chart.gutter.left', 45);
    }
    return window.__rgraph_bar__;
};

/**
 * we only create the top reads graph once and then store it in the
 * window object.
 */
Graph.prototype.getReadGraph=function(id, data1,categories1){
       if (!window.__rgraph_bar2__) {
              window.__rgraph_bar2__ = new RGraph.Bar(id, data1);
           window.__rgraph_bar2__.Set('chart.labels', categories1);
           window.__rgraph_bar2__.Set('chart.tooltips',  function (idx) {return
                                                  categories1[Math.floor(idx/2)];});
           window.__rgraph_bar2__.Set('chart.tooltips.effect', 'fade');
           window.__rgraph_bar2__.Set('chart.tooltips.event', 'onmousemove');
           window.__rgraph_bar2__.Set('chart.background.grid', true);
           window.__rgraph_bar2__.Set('chart.colors', ['red']);
           window.__rgraph_bar2__.Set('chart.gutter.left', 45);
       }
       return window.__rgraph_bar2__;

};

/**
 * we only create the top wirtes graph once and then store it in the
 * window object.
 */
Graph.prototype.getWriteGraph=function(id, data1,categories1){
       if (!window.__rgraph_bar3__) {
              window.__rgraph_bar3__ = new RGraph.Bar(id, data1);
           window.__rgraph_bar3__.Set('chart.labels', categories1);
           window.__rgraph_bar3__.Set('chart.tooltips',  function (idx) {return
                                                 categories1[Math.floor(idx/2)];});
           window.__rgraph_bar3__.Set('chart.tooltips.effect', 'fade');
           window.__rgraph_bar3__.Set('chart.tooltips.event', 'onmousemove');
           window.__rgraph_bar3__.Set('chart.background.grid', true);
           window.__rgraph_bar3__.Set('chart.colors', ['blue']);
           window.__rgraph_bar3__.Set('chart.gutter.left', 45);
       }
       return window.__rgraph_bar3__;
};
```

StatsObj.js

```
/**
 * the rwArray is an Array of elements. for each object in the
 * cloud we keep its name and read/write counters
 */
function element(name,reads,writes){
	this.name=name;
	this.reads=reads;
	this.writes=writes;
}

/**
 * the sorted Arrays are arrays of sortedElements.
 */
function sortedElement(name,val){
	this.name=name;
	this.val=val;
}

/**
 * for a Graph object we have a StatsObj field.
 * the StatsObj field is responsible for processing a json parsed
 * event. The StatsObj is responsible for updating the rwArray and
 * the sorted Arrays.
 */
function StatsObj(){
	/**
	 * the basic array that holds the objects
	 * that the user has accessed in the cloud
	 */
	this.rwArray=new Array();
	/**
	 * this is an index the keeps the position in the rwArray
	 * of the first object currently shown in the main graph
	 * (also the curRWArray and curNamesArray)
	 */
	this.rwArrayIndex=0;
	/**
	 * the length of the current arrays. this changes
	 * using the zoom buttons
	 */
	this.shifter=5; //default shift is 5
	/**
	 * the current arrays keep the objects show currently in the
	 * main graph
	 */
	this.curNamesArray=new Array();
	this.curRWArray=new Array();

	/**
	 * the sorted arrays keep the top reads/writes
	 * objects
	 */
	this.sortedReadsArray=this.initSortedArray();
	this.sortedWritesArray=this.initSortedArray();

	/**
	 * these arrays are helper arrays used for preparing
	 * the data that will change when updating the graphs
	 */
	this.sortedReadData=new Array();
	this.sortedReadCat=new Array();
	this.sortedWriteData=new Array();
	this.sortedWriteCat=new  Array();
```

```
}

/**
 * this method initializes the sorted arrays
 * @returns {Array}
 */
StatsObj.prototype.initSortedArray=function(){
	var len=5; //sorted arrays have length=5
	var arr=new Array();
	for (var i=0;i<len;i++)
		arr.push(new sortedElement("",0));
	return arr;
};


/**
 * this method is called
 * 1. when the user presses the next/prev/zoom buttons-> in charttools.js
 * 2. if the event that just came is currently showing in the main graph
 * 3. memory protection mechanism is triggered
 */
StatsObj.prototype.updateCurArrays=function(){
	console.log("updating current arrays to show requested data
			starting from Read/Write Array index: "
			+this.rwArrayIndex+" to: "
			+ parseInt(this.rwArrayIndex+this.shifter));
	//zero length the current arrays
	this.curNamesArray.length=0;
	this.curRWArray.length=0;
	var k=0;
	for(var i=this.rwArrayIndex;i<(this.rwArrayIndex+this.shifter)
	    && (i<this.rwArray.length);i++){
		this.curNamesArray[k]=this.rwArray[i].name;
		this.curRWArray[k++]=
			new Array(this.rwArray[i].reads,this.rwArray[i].writes);
	}
	if(this.rwArrayIndex!=0)
		$("#chartmsg").html("");
};

/**
 * this is the public method that is called from the graph object.
 * IMPORTANT: objName and objType depend on the elements of myEvents
 * array having the correct fields.
 */
StatsObj.prototype.calcStatistics=function(myEvents){
	for(var i=0;i<myEvents.length;i++){
		var objName=myEvents[i].tenant+"."+myEvents[i].user+"."
				+myEvents[i].container+"."+myEvents[i].object;
		var objType=myEvents[i].operation; //read or write
		var objCount=myEvents[i].count; //no. of reads/writes on this object
		console.log("resource name: "+objName+" resource type: "+objType);

		if (!this.existsInStats(objName))
			this.addToStats(objName);
		else
			console.log(objName+" not a new object");

		this.updateStats(objName,objType,objCount);
		this.showStats();
	}
};

/**
 * this method checks if the object processed is already
 * in the rwArray or it is a new object
 * @param name
 * @returns {Boolean}
```

```
 */
StatsObj.prototype.existsInStats=function(name){
	for (var i=0;i<this.rwArray.length;i++)
		if (name==this.rwArray[i].name)
			return true;
	return false;
};

/**
 * this method adds a new object to the rwArray
 * @param name
 */
StatsObj.prototype.addToStats=function(name){
	this.rwArray.push(new element(name,0,0)); //[name,reads,writes]
	console.log("added object: "+name+" to stats");

	//if needed insert to current arrays
	if(this.rwArrayIndex+this.shifter>=this.rwArray.length){
		this.curNamesArray.push(name);
		this.curRWArray.push(new Array(0,0));
	}
};

/**
 * this method updates the arrays with the new stats
 */
StatsObj.prototype.updateStats=function(name,type,count){
	var index=0;

	for (var i=0;i<this.rwArray.length;i++){
		if (name==this.rwArray[i].name){
			if(type=="read")
				this.rwArray[i].reads=this.rwArray[i].reads + count;
			if(type=="write")
				this.rwArray[i].writes=this.rwArray[i].writes + count;
			index=i;
			break;
		}
	}

	//if needed update current arrays
	if(index>=this.rwArrayIndex && index<this.rwArrayIndex+this.shifter){
		console.log("current arrays index: "+i+" is going to be updated");
		for(var i=0;i<this.curNamesArray.length;i++){
			if (name==this.curNamesArray[i]){
				if(type=="read")
					this.curRWArray[i][0]=this.rwArray[index].reads;
				if(type=="write")
					this.curRWArray[i][1]=this.rwArray[index].writes;
				break;
			}
		}
	}

	//if needed update sorted arrays
	var len=this.sortedReadsArray.length;
	//reads and writes array have the same length
	var sortArray=new Array();

	if(type=="read"){
		value=this.rwArray[index].reads;
		sortArray=this.sortedReadsArray;
	}
	if(type=="write"){
		value=this.rwArray[index].writes;
		sortArray=this.sortedWritesArray;
	}
	//is it a top read or write
```

```
	if(value>=sortArray[len-1].val)
		this.insertInSortArray(sortArray,name,value);
};

/**
 * this method checks if the object processed is a top read/write
 * if it is we add it to the sorted arrays and update the
 * sorted arrays
 * @param sortArray
 * @param name
 * @param value
 */
StatsObj.prototype.insertInSortArray=function(sortArray,name,value){
	var len=sortArray.length;
	var pos=-1;
	for(var i=len-1;i>=0;i--){
		if(sortArray[i].name==name){
			//already a top read or write
			pos=i;
			sortArray[i].val=value;
			break;
		}
	}
	if(pos==-1){
		//insert it to the sorted read or write array
		sortArray.pop();
		sortArray.push(new sortedElement(name,value));
		pos=len-1;
	}

	sortArray.sort(function(a,b){return b.val - a.val;});
};

/**
 * this method updates the #statsTable to show the rwArray as a
 * list
 */
StatsObj.prototype.showStats=function(){
	$("#statsTable").html("");
	var s="<tr><th>Object Name</th><th>Reads</th><th>Writes</th></tr>";
	var len=18;

	for (var i=0;i<this.rwArray.length;i++){
		var name=this.rwArray[i].name;
		var srow="<tr><td title=\""+name+"\">"+((name.length<=len)? name :
				("..."+name.substr(name.length-len,name.length)))+"</td>"
		   +"<td>"+this.rwArray[i].reads+"</td>"
		 +"<td>"+this.rwArray[i].writes+"</td></tr>";
		s=s+srow;
	}$("#statsTable").html(s);
};

/**
 * this method is triggered when there is a need to slice the
 * rwArray. first we slice the rwArray then we update the current arrays
 * and after the sorted arrays
 * @param limit
 */
StatsObj.prototype.memoryProtectMechanism=function(limit){
	console.log("Memory Protection Mechanism kicked in");
	this.rwArray=this.rwArray.slice(limit,this.rwArray.length);
	this.rwArrayIndex=0;
	this.updateCurArrays();
	this.showStats();

	this.sortedReadsArray=this.initSortedArray();
	this.sortedWritesArray=this.initSortedArray();
	//get a copy of the rwArray. don't want to change it
```

```
        var tempArr=this.rwArray.slice(0);
        //for the sortedWritesArray
        tempArr.sort(function(a,b){return b.writes - a.writes;});

        for(var i=0;i<5 && i<tempArr.length;i++){
                (tempArr[i].writes>0) ? this.sortedWritesArray[i].name=tempArr[i].name
                                      : this.sortedWritesArray[i].name="";
                this.sortedWritesArray[i].val=tempArr[i].writes;
        }

        //for the sortedReadsArray
        tempArr.sort(function(a,b){return b.reads - a.reads;});

        for(var i=0;i<5 && i<tempArr.length;i++){
                (tempArr[i].reads>0) ? this.sortedReadsArray[i].name=tempArr[i].name
                                     : this.sortedReadsArray[i].name="";
                this.sortedReadsArray[i].val=tempArr[i].reads;
        }

        var s= "<div><div class=\"eventDivHeader\">Memory Protection Mechanism kicked
in</div>"
                  +"Object holding array reducing its size, deleting the oldest 1000
objects</div>";
        $("#showEventsDiv").append(s);

};
```

Testing tools

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">
  <display-name>EventMonitor</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

   <servlet>
    <servlet-name>jersey-serlvet-AuthenticateServer</servlet-name>
    <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
    <init-param>
      <param-name>com.sun.jersey.config.property.packages</param-name>
      <param-value>authenticatePackage</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>jersey-serlvet-AuthenticateServer</servlet-name>
    <url-pattern>/authenticate/*</url-pattern>
  </servlet-mapping>

  <servlet>
    <servlet-name>jersey-serlvet-QueueServer</servlet-name>
    <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
    <init-param>
      <param-name>com.sun.jersey.config.property.packages</param-name>
      <param-value>queueServerPackage</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>jersey-serlvet-QueueServer</servlet-name>
    <url-pattern>/queues/*</url-pattern>
  </servlet-mapping>

   <servlet>
    <servlet-name>EventServer1</servlet-name>
    <servlet-class>eventPackage1.EventServer1</servlet-class>
    <async-supported>true</async-supported>
  </servlet>
  <servlet-mapping>
    <servlet-name>EventServer1</servlet-name>
    <url-pattern>/getEvents1/*</url-pattern>
  </servlet-mapping>

</web-app>
```

AuthenticateServer.java

```
package authenticatePackage;

import java.util.StringTokenizer;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.HttpHeaders;
import javax.ws.rs.core.Response;

import org.json.JSONException;
import org.json.JSONObject;

import com.sun.jersey.core.util.Base64;

@Path("/")
public class AuthenticateServer {
	String[] userArray={"jimmys","user1","user2"};
	String[] passArray={"$1234","asdf","1234"};

	@GET
	public Response getauthenticated(@Context HttpHeaders headers){
	   String authHeader=headers.getRequestHeader("Authorization").get(0);

	    StringTokenizer stokenz=new StringTokenizer(authHeader);
	    String coded=stokenz.nextToken();
	    coded=stokenz.nextToken();
	    System.out.println("EncodedText: "+coded);

	    String decodedText=Base64.base64Decode(coded);
	    System.out.println("DecodedText: "+decodedText);

	    int ch1=decodedText.indexOf('@');
	    String userName=decodedText.substring(0, ch1);
	    int ch2=decodedText.indexOf(':');
	       String tenantName=decodedText.substring(ch1+1, ch2);
	       String password=decodedText.substring(ch2+1);

	       if(!checkCreds(userName,password)){
	              return unauthorizedResponse();
	       }

	       JSONObject object = new JSONObject();

	       try {
	              object.put("userName", userName);
	              object.put("tenantName", tenantName);
	       } catch (JSONException e) {
	              e.printStackTrace();
	       }

	       return Response.status(Response.Status.OK)
	                           .header("Content-Type","application/json")
	                         .entity(object.toString())
	                           .build();
	}

	private boolean checkCreds(String username,String password){
	       for(int i=0;i<userArray.length;i++){
	              if (userArray[i].equals(username) && passArray[i].equals(password))
	                     return true;
	       }
	       return false;
```

```
        }

        private Response unauthorizedResponse(){
                return Response.status(Response.Status.UNAUTHORIZED).build();
        }
}
```

QueueServer.java

```
package queueServerPackage;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.UUID;

import javax.servlet.http.HttpServletRequest;
import javax.ws.rs.Consumes;
import javax.ws.rs.DELETE;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.HttpHeaders;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;


import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import databasePackage.DBManager;

@Path("/")
public class QueueServer {
	private String currentCDMIVersion="1.0.1";
	DBManager dbManager;

	public QueueServer() {
		dbManager= new DBManager();
    }

	@PUT
	@Path("/{param}")
	@Consumes("application/cdmi-queue")
	@Produces({"application/cdmi-queue",MediaType.APPLICATION_JSON})
	public Response createQueue(@PathParam("param") String queueName,
                                                    @Context HttpHeaders headers,
                                                    @Context HttpServletRequest req) {

		if (!validateRequestHeaders(headers)){
			return badRequestResponse();
		}
		if(queueNameExists(queueName)){
			return conflictResponse();
		}

		JSONObject object = new JSONObject();
		JSONObject metadata = new JSONObject();

		String uuid;
		String objectID="";
		String objectType="";
		String objectName="";
		String parentURI="";
		String parentID="";
		String domainURI="";
		String capabilitiesURI="";
```

```
		String queueValues="";
		try {
			uuid = UUID.randomUUID().toString();
			objectID=uuid.replaceAll("-", "");

			objectType="application/cdmi-queue";
			object.put("objectType", objectType);

			object.put("objectID", objectID);

			objectName=queueName;
			object.put("objectName", objectName);

			parentURI="/queues/";
			object.put("parentURI", parentURI);

			parentID=objectID;
			object.put("parentID", parentID);

			domainURI="/cdmi_domains/MyDomain/";
			object.put("domainURI", domainURI);

			capabilitiesURI="/cdmi_capabilities/queue/";
			object.put("capabilitiesURI", capabilitiesURI);

			String completionStatus="complete";
			object.put("completionStatus", completionStatus);

			//metadata is a json object not a string
			metadata.put("m1", "1234");
			object.put("metadata", metadata);

			queueValues="";
			object.put("queueValues", queueValues);

		}
		catch (JSONException e) {
			e.printStackTrace();
		}

		insertQueueToDB(queueName,objectID,parentURI,parentID);

		return Response.status(Response.Status.CREATED)
					.header("Content-Type","application/cdmi-queue")
				.header("X-CDMI-Specification-Version", currentCDMIVersion)
					.entity(object.toString())
					.build();
	}


	 @GET
	@Path("/{param}")
	@Consumes("application/cdmi-queue")
	@Produces("application/cdmi-queue")
	public Response doGet(@PathParam("param") String queueName) {
		return badRequestResponse();
	}


	@POST
	@Path("/{param}")
	@Consumes("application/cdmi-queue")
	@Produces("application/cdmi-queue")
	public Response doPost(@PathParam("param") String queueName) {
		return badRequestResponse();
	}

	@DELETE
```

```
	@Path("/{param}")
	@Consumes("application/cdmi-queue")
	@Produces("application/cdmi-queue")
	public Response doDelete(@PathParam("param") String queueName) {
		return badRequestResponse();
	}

	private boolean validateRequestHeaders(HttpHeaders headers){
		boolean isValid=true;
		try{
			String accept=headers.getRequestHeader("Accept").get(0);
			String contentType=headers.getRequestHeader("Content-Type").get(0);
			String xcdmiVersion=headers.getRequestHeader("X-CDMI-Specification-
                                                                Version").get(0);

			if(!accept.equals("application/cdmi-queue"))
				isValid=false;
			else if(!contentType.equals("application/cdmi-queue"))
				isValid=false;
			else if(!xcdmiVersion.equals(currentCDMIVersion))
				isValid=false;
		}
		catch(NullPointerException  e){
			isValid=false;
		}

		return isValid;
	}

	private boolean queueNameExists(String queueName){
		System.out.println(queueName);
		boolean exists=false;
		dbManager.dbConnect();
		String qry="SELECT `queueName` FROM `queues` WHERE
                                                        `queueName`='"+queueName+"'";
		ResultSet rs=dbManager.dbExecuteQuery(qry);
		try {
			if (rs.next())
				exists=true;
		} catch (SQLException e) {
			e.printStackTrace();
		}
		dbManager.dbDisconnect();

		return exists;
	}

	private void insertQueueToDB(String queueName, String objectID, String parentURI,
                                                                String parentID){
		dbManager.dbConnect();
		String qry="INSERT INTO `queues`(`objectID`, `queueName`, `parentURI`,
                                                                `parentID`)"+
                  " VALUES
                 ('"+objectID+"','"+queueName+"','"+parentURI+"','"+parentID+"')";
		dbManager.dbInsert(qry);
		dbManager.dbDisconnect();
	}

	protected Response badRequestResponse(){
		return Response.status(Response.Status.BAD_REQUEST)
                                .header("Content-Type","application/cdmi-queue")
                                .header("X-CDMI-Specification-Version",
                                                                currentCDMIVersion)
                                .build();
	}

	protected Response conflictResponse(){
		return Response.status(Response.Status.CONFLICT)
```

```
                                .header("Content-Type","application/cdmi-queue")
                                .header("X-CDMI-Specification-Version",
                                                                    currentCDMIVersion)
                                .build();
        }
}
```

EventServer.java

```
package eventPackage1;

import java.io.IOException;
import java.sql.Timestamp;
import java.util.Date;
import java.util.Random;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;

import org.eclipse.jetty.servlets.EventSource;
import org.eclipse.jetty.servlets.EventSourceServlet;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;


public class EventServer1 extends EventSourceServlet{
    private static final long serialVersionUID = 1L;

    private static final Logger LOG = Logger.getLogger("EventMonitor");
    private volatile ScheduledExecutorService executor;

    @Override
    public void init() throws ServletException{
        super.init();
        this.executor = Executors.newSingleThreadScheduledExecutor();
    }

    @Override
    public void destroy(){
        this.executor.shutdown();
        super.destroy();
    }

    @Override
    protected EventSource newEventSource(HttpServletRequest request){
       String queueName=request.getParameter("queueName");
              System.out.println("Getting events for queue: "+queueName);
        return new ObjEventSource(queueName);
    }

    final class ObjEventSource implements EventSource{
        private volatile Emitter emitter;
        private volatile boolean closed = false;
        private String queueName;

        ObjEventSource(String queueName){
             this.queueName=queueName;
        }

        @Override
        public void onOpen(Emitter emitter) throws IOException{
            this.emitter = emitter;
            System.out.println("Stream open for queue: "+queueName);
            this.sendEvent(0L);
        }
```

```
        @Override
        public void onClose(){
            closed = true;
            System.out.println("client closed stream for queue: "+queueName);
        }

        private void sendEvent(final long count) throws IOException {
            if (this.closed)
                return;

            Random randomGenerator = new Random();
            int numOfRows=2+randomGenerator.nextInt(50);
            String optype=(randomGenerator.nextInt(2)==0) ? "read" : "write";

            System.out.println("starting index is: "+count+" operation: "+optype+"
                               number of rows: "+numOfRows);

            JSONObject eventObj = new JSONObject();
            JSONObject object;
            JSONArray usersArray = new JSONArray();

                    try{
                           String originating_machine="147.102.19.186";
                           eventObj.put("originating_machine", originating_machine);
                           Date date= new Date();
                           Timestamp timestamp=new Timestamp(date.getTime());
                           eventObj.put("timestamp", timestamp );
                           String aggregation="no of objects ";
                           aggregation+=(optype.equals("read")) ? "read":"written";
                           eventObj.put("aggregation", aggregation);
                           String topic="Billing";
                           eventObj.put("topic", topic);
                           String cluster="test";
                           eventObj.put("cluster", cluster);
                           String originating_service="AggregatedEventsServer";
                           eventObj.put("originating_service", originating_service);
                           String units="";
                           eventObj.put("units", units);
                           eventObj.put("tstart", timestamp);
                           eventObj.put("tend", timestamp);

                           for(int i=0;i<numOfRows;i++){
                    int objcount=1+randomGenerator.nextInt(300);
                    String object_name="object"+randomGenerator.nextInt(1000);
                           String
                              container_name="container"+randomGenerator.nextInt(10);
                           String user="user"+(1+randomGenerator.nextInt(7));
                           String tenant="tenant1";

                           object = new JSONObject();
                           object.put("count", objcount);
                           object.put("object", object_name);
                           object.put("container", container_name);
                           object.put("user", user);
                           object.put("tenant", tenant);

                           usersArray.put(object);
                     }

                           eventObj.put("users", usersArray);

                           System.out.println(eventObj.toString());

                           this.emitter.data(eventObj.toString()+"\n\n");
                           executor.schedule(new UpdateSender(count+1),2L, TimeUnit.SECONDS);

                    }
                     catch (JSONException e) {
```

```
                        e.printStackTrace();
                    }

        }

        class UpdateSender implements Runnable{
             private long c;
             public UpdateSender(long c){
                    this.c=c;
             }
             @Override
            public void run(){
                try{
                    sendEvent(c);
                }
                catch (IOException e){
                    LOG.log(Level.SEVERE, "could not send update to client", e);
                }
            }
        }

    }
}
```

Βιβλιογραφία

[1]: "*Cloud Computing Bible*", Barrie Sosinsky, 2011, Wiley Publishing

[2]: "*Cloud Application Architectures: Building Applications and Infrastructure in the Cloud*", George Reese, 2009, O'Reilly

[3]: "*Cloud Computing Explained: Implementation Handbook for Enterprises*", John Rhoton, 2009, Recursive

[4]: Google Trends, http://www.google.com/trends/

[5]: IDC, "*IT Cloud Services Forecast – 2008, 2012: A Key Driver of New Growth*", http://blogs.idc.com/ie/?p=224/

[6]: George Kanellopoulos's Blog, "*Τι είναι το Cloud Computing*", Γιώργος Κανελλόπουλος, 2010, http://blogs.msdn.com/b/gkanel/archive/2010/10/29/cloud-computing.aspx

[7]: Wikipedia, "*Cloud Computing*", http://en.wikipedia.org/wiki/Cloud_computing

[8]: Wikipedia, "*Saas*"-"*PaaS*"-"*IaaS*", http://en.wikipedia.org/wiki/Software_as_a_Service

[9]: Webopedia, "*Saas*"-"*PaaS*"-"*IaaS*", http://www.webopedia.com/TERM/saas

[10]: Cloud Tweaks, "*Cloud Computing For Dummies: SaaS, PaaS, IaaS And All That*", http://www.cloudtweaks.com/2011/02/cloud-computing-for-dummies-saas-paas-iaas-and-all-that-was/

[11]: HowStuffWorks, "*How Cloud Computing Works*", Jonathan Strickland, http://computer.howstuffworks.com/cloud-computing/cloud-computing2.htm

[12]: SNIA, "*terms and standards*", http://cloud-standards.org/wiki/index.php?title=SNIA_Terms_and_Diagrams

[13]: IDC, "*2011 Digital Universe Study*", 2011, http://www.emc.com/collateral/demos/microsites/emc-digital-universe-2011/index.htm

[14]: IBM developerWorks, "*Anatomy of a cloud storage infrastructure*", 2010, http://www.ibm.com/developerworks/cloud/library/cl-cloudstorage/

[15]: Wikipedia, REST, http://en.wikipedia.org/wiki/Representational_state_transfer

[16]: Roy Fielding, 2000, Doctoral Dissertation, "*Architectural Styles and the Design of Network-based Software Architectures*"

[17]: IBM, developerWorks, "*RESTful Web services: The basics*", 2008, http://www.ibm.com/developerworks/webservices/library/ws-restful/

[18]: "*RESTful Web Services*", Leonard Richardson & Sam Ruby, 2007, O'Reily Media

[19]: infoQ, "*A Brief Introduction to REST*", Stefan Tilkov, 2007, http://www.infoq.com/articles/rest-introduction

[20]: "*Representational State Transfer (REST)*", 2011, http://www.peej.co.uk/articles/rest.html

[21]: infoQ," REST *and SOAP: When Should I Use Each (or Both)*" , Mike Rozlog, 2010, http://www.infoq.com/articles/rest-soap-when-to-use-each/

[22]: Wikipedia, "*AJAX*", http://en.wikipedia.org/wiki/Ajax_(programming)

[23]: w3schools, "*AJAX Basic*", http://www.w3schools.com/ajax/ajax_intro.asp

[24]: Pritam Baldota, "*What is AJAX*", http://www.pritambaldota.com/index.php/what-is-ajax/

[25]: HTML5 Rocks, "*What is HTML5*", http://www.html5rocks.com/en/why

[26]: Wikipedia, "*HTML5*", http://en.wikipedia.org/wiki/HTML5

[27]: Κοινότητα WHATWG, http://wiki.whatwg.org/wiki/FAQ#What_is_the_WHATWG.3F

[28]: Wikipedia, *"Canvas element*", http://en.wikipedia.org/wiki/Canvas_element

[29]: w3schools, "*HTML5 Web Storage*", http://www.w3schools.com/html5/html5_webstorage.asp

[30]: webReference, "*HTML 5: Client-side Storage*", Arpan Dhandhania , http://www.webreference.com/authoring/languages/html/HTML5-Client-Side/index.html

[31]: Wikipedia, "*Server Side Events*", http://en.wikipedia.org/wiki/Server-sent_events

[32]: HTML5 Rocks, "*Stream updates with server side events*", Eric Bidelman, 2010, http://www.html5rocks.com/en/tutorials/eventsource/basics/

[33]: Dmitry Sheiko' Blog, "*Server-Sent Events vs WebSockets vs Long-polling*", http://dsheiko.com/weblog/websockets-vs-sse-vs-long-polling/

[34]: SNIA, "*About the SNIA*", http://www.snia.org/about

[35]: SNIA, "CDMI-*SNIA Technical Position*", http://snia.org/sites/default/files/CDMI_SNIA_Architecture_v1.0.1.pdf

[36]: Wikipedia, "*Cloud Data Management Interface*", http://en.wikipedia.org/wiki/Cloud_Data_Management_Interface

[37]: cordis.europa.eu, "*Vision Cloud*", http://cordis.europa.eu/fp7/ict/ssai/docs/call5-visioncloud.pdf

[38]: Vision Cloud, "*Vision Cloud: The Fact Sheet*", http://www.visioncloud.eu/content.php?s=30,47

[39]: Distributed Knowledge and Media Systems Group NTUA, "*VISION Cloud Project*", http://grid.ece.ntua.gr/?projects=vision-cloud

[40]: "*The Client Side of Cloud Computing*", Mario Höfer & Gernot Howanitz, 2009, Universität Salzburg

[41]: ebizq, "*Five Challenges to Monitoring Cloud Applications*", Andre Yee, 2011, http://www.ebizq.net/blogs/cloudtalk/2011/08/five_challenges_to_monitoring.php

[42]: "*Monitoring End User Experience for Virtualized and Cloud Hosted Applications*", Bernd Harzog, 2011, http://www.virtualizationpractice.com/monitoring-end-user-experience-for-virtualized-and-cloud-hosted-applications-10786/

[43]: Compuware, "*What's So Hard About Cloud Management*", http://www.compuware.com/application-performance-management/dont-blindly-trust-the-cloud.html

[44]: W3C, "About W3C", http://www.w3.org/Consortium/

[45]: W3C, "Server-Sent-Events", 2012, http://www.w3.org/TR/eventsource/

[46]: Vision Cloud, "*REST API Exposed by User Management Service*", Vision Cloud

[47]: "*Apach Tomcat 7 documentation*", Apache Foundation, http://tomcat.apache.org/tomcat-7.0-doc/index.html

[48]: "*Jersey*", Java.net, http://jersey.java.net/

[49]: "*REST with Java (JAX-RS) using Jersey*", Lars Vogel, 2012, http://www.vogella.com/articles/REST/article.html

[50]: github, "*Jetty event server documentation*", https://github.com/jetty-project/jetty-eventsource-servlet/wiki

[51]: "*jqplot documentation*", jqplot, http://www.jqplot.com/docs/files/usage-txt.html

[52]: "*ext-js 4.1 documentation*", sencha, http://docs.sencha.com/ext-js/4-1/

[53]: "*rgraph documentation*", rgraph, http://www.rgraph.net/docs/index.html

[54]: How-To Geek, "*Monitor and Control Memory Usage in Google Chrome*", http://www.howtogeek.com/howto/16102/monitor-and-control-memory-usage-in-google-chrome/

Ευρετήριο Εικόνων

Ευρετήριο Πινάκων