Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**(Σχεδίαση και ανάπτυξη εφαρμογής τρισδιάστατης απεικόνισης ανοιχτών γεωλογικών δεδομένων σε δημόσιο υπολογιστικό νέφος)** **(3D Design and developing application presenting open geospatial data over public cloud)**
Ονοματεπώνυμο Φοιτητή	**Θωμάς Χαβάκης**
Πατρώνυμο	**Ιωάννης**
Αριθμός Μητρώου	**ΜΠΣΠ/ 12088**
Επιβλέπων	**Χρήστος Δουληγέρης, Καθηγητής**

Τρίτη 16 **ΙΟΥΝΙΟΥ 2014**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα

ΕΥΧΑΡΙΣΤΙΕΣ

Για την εκπόνηση αυτής της μεταπτυχιακής διατριβής συνεργάστηκε μια ομάδα ανθρώπων, οι οποίοι με βοήθησαν με διάφορους τρόπους. Μεταξύ αυτών ο καθηγητής Χ. Δουληγέρης ο οποίος ήταν ο άνθρωπος για τις δύσκολες στιγμές και η γνώμη του είναι βαρυσήμαντη. Θα ήθελα να ευχαριστήσω τον κ. Κ. Χίμο ο οποίος πίστεψε σε μένα και στις δυνατότητες μου και αυτό ήταν φανερό από την πρώτη κιόλας στιγμή της συνεργασίας μας. Ιδιαίτερα ευχαριστήρια θέλω να εκφράσω για τον κ. Καλλέργη με τον οποίο δαπανήσαμε πολλές ώρες και έναν πολύ μεγάλο αριθμό email γεγονός που αποδεικνύει πόσο επαγγελματίας και πόσο συνεπής είναι στις υποχρεώσεις του. Θα ήταν παράλειψη αν δεν ανέφερα την κ. Παπαχαραλάμπου η οποία βοήθησε την ομάδα με τις εξειδικευμένες γνώσεις της.

Επίσης θα ήθελα να ευχαριστήσω τον κ. Γ. Καπνιά, ο οποίος πάντα με βοηθούσε με την εμπειρία και τις γνώσεις του.

Τέλος τα πιο μεγάλα ευχαριστήρια και την αγάπη μου, θα ήθελα να τα μεταφέρω στους γονείς μου οι οποίοι αν δεν με βοηθούσαν σε όλους τους τομείς αυτό το μεταπτυχιακό πιθανόν και να μην το έκανα ποτέ.

Χαβάκης Θωμάς

Πειραιάς , Μάιος 2014

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΛΙΣΤΑ ΕΙΚΟΝΩΝ

ΛΙΣΤΑ ΠΙΝΑΚΩΝ

ΚΕΦΑΛΑΙΟ 1

1.1 Πρόλογος

Η εργασία αυτή εντάσσεται στο πεδίο των Τεχνολογιών Πληροφορικής & Επικοινωνιών (ΤΠΕ) και της αλληλεπίδρασής τους με άλλα πεδία επιστημών. Πιο συγκεκριμένα, στο πλαίσιο εκπόνησης της παρούσας μεταπτυχιακής διατριβής χρησιμοποιήθηκαν χωρικά δεδομένα για να απεικονιστούν σε τρεις διαστάσεις (3D) μέσω διαδικτυακής εφαρμογής αντικειμενοστρεφούς λογικής σε αρχιτεκτονική υπολογιστικού νέφους (Cloud computing).

Η εργασία χρησιμοποιεί πρωτογενή δεδομένα (raw data) από πραγματικές γεωτρήσεις σε συγκεκριμένο γεωγραφικό χώρο της ελληνικής επικράτειας. Στη συνέχεια, τα απεικονίζει σύμφωνα με τους κανόνες και τους περιορισμούς που θέτουν οι επιστήμονες του κλάδου. Για τους σκοπούς της εργασίας υποθέτουμε ότι παρόμοια δεδομένα θα προέρχονται από τη βάση δεδομένων του «Εθνικού Μητρώου Υδρογεωτρήσεων» (ΥΠΕΚΑ, αρ.πρωτ. 1042/12.9.2013, ΑΔΑ: ΒΛ9Κ0-ΝΝΤ). Υποθέτουμε, επίσης, ότι τα δεδομένα της παραπάνω βάσης θα ενταχθούν στα «Δημόσια Ανοικτά Δεδομένα» του ΥΔΜΗΔ (http://geodata.gov.gr) και στο «Μητρώο Ελληνικών Ανοικτών Δεδομένων» (http://ckan.okfn.gr/).

Οι χρήστες της διαδικτυακής εφαρμογής που παρέχεται ως υπηρεσία επιλέγουν μια γεωγραφική περιοχή της χώρας και περιηγούνται στις γεωαναφερμένες (geo-located) γεωτρήσεις της. Στο χώρο των τριών διαστάσεων, μπορούν να οπτικοποιήσουν συγκεκριμένες πληροφορίες που αφορούν τις γεωτρήσεις (βάθη, στοιχεία υδροσωλήνων κ.λπ.) και τους υδροφόρους ορίζοντες.

Για την ολοκλήρωση της παρούσας εργασίας χρησιμοποιήθηκαν εργαλεία για την ανάπτυξη κώδικα (IDE) με αντικειμενοστρεφή λογική, καθώς και αντίστοιχα για την απεικόνιση γεωγραφικών πληροφοριών (Γεωγραφικά Συστήματα Πληροφοριών). Σε μια πρωταρχική προσέγγιση αξιοποιούμε την υποδομή υπολογιστικού νέφους του Εθνικού Δικτύου Έρευνας

και Τεχνολογίας (Okeanos), για να φιλοξενηθούν τα υπό υλοποίηση υποσυστήματα. Η εργασία χωρίζεται σε δύο φάσεις:

(α) ενοποίησης των διαφορετικών και ετερογενών συστημάτων, και

(β) υλοποίησης και ελέγχου σε αρχιτεκτονική υπολογιστικού νέφους.

Τα αποτελέσματα της παρούσας εργασίας μπορούν να καταταχθούν στο πεδίο της Υδροπληροφορικής (HydroInformatics), στόχος της οποίας είναι η συλλογή, ερμηνεία και διαχείριση δεδομένων που σχετίζονται με τον ευρύτερο τομέα του νερού (περιβάλλον, οικονομία, υποδομές, κοινωνία, νομοθεσία) και στην ανάπτυξη εφαρμογών που θα προσεγγίζουν τις υπάρχουσες και μελλοντικές προκλήσεις που σχετίζονται με το νερό σε συστημικό επίπεδο (*π.χ.* λεκάνης απορροής).

Παρόμοιες υλοποιήσεις οπτικοποίησης γεωλογικών και χωρικών δεδομένων μόνο για σταθερούς χρήστες και σε απεικόνιση χώρου δύο διαστάσεων, συναντάμε **(α)** στο Υπουργείο Άμυνας της Ομοσπονδιακής Ελβετικής Κυβέρνησης *(http://www.swisstopo.admin.ch)*, **(β)** στο έργο «Υγροσκόπιο», της ερευνητικής ομάδας ΙΤΙΑ της Σχολής Πολιτικών Μηχανικών του ΕΜΠ υπό την αιγίδα του ΥΠΕΚΑ *(http://thyamis.itia.ntua.gr/Hydro_Base/)*.

1.2 Σκοπός Μεταπτυχιακής Διατριβής

Σκοπός της διατριβής είναι η αξιοποίηση ανοικτών χωρικών δεδομένων και η απεικόνισή τους στο χώρο των τριών διαστάσεων με τη χρήση της γεωαναφοράς τους. Το τελικό παραδοτέο αποτελείται από μια διαδικτυακή εφαρμογή απεικόνισης γεωτρήσεων και υπόγειων Υδροφορέων, η οποία αναπτύσσεται σε αρχιτεκτονική υπολογιστικού νέφους με αντικειμενοστρεφή λογική.

Σκοπός:

- Τρισδιάστατη απεικόνιση γεωτρήσεων και υπόγειων Υδροφορέων σε περιοχές της ελληνικής επικράτειας,

- Υλοποίηση πληροφοριακού συστήματος με τη χρήση *υποδομής ως υπηρεσίας* (Infrastructure as a Service – IaaS) και, τελικά, την παροχή *λογισμικού ως υπηρεσία* (Software as a Service – SaaS) σε δημόσιο υπολογιστικό νέφος,
- Αποτίμηση συνθηκών υψηλής ζήτησης των υπηρεσιών και εφαρμογή πολιτικών εξισορρόπησης φορτίου δικτυακής κίνησης,
- Δυναμική προσέγγιση για την ανάπτυξη συστήματος σε μεγαλύτερη κλίμακα.
- Αποτελέσματα συγκριτικών μετρήσεων σε δομές Υπολογιστικού Νέφους

1.3 Αντικείμενα Μελέτης Μεταπτυχιακής Διατριβής

Αντικείμενο μελέτης της μεταπτυχιακής διατριβής αποτελούν η διασύνδεση με ψηφιακά αποθετήρια ανοικτής και δημόσιας πρόσβασης, η χρήση μηχανών παραγωγής χώρων τριών διαστάσεων και η ανάπτυξη εφαρμογής με το παράδειγμα υπολογιστικών νεφών.

Αντικείμενα μελέτης:

- Δημιουργία Web Service για τη διασύνδεση με χωρικές ΒΔ *(Spatial RDBS)*.
- Υποδομή συστήματος που θα υποστηρίζει τη βέλτιστη ανεύρεση, πρόσβαση και χρησιμοποίηση χωροχρονικών δεδομένων σε υπολογιστικό νέφος *(Spatial Cloud Computing),*
- Καταστάσεις υψηλής ζήτησης των υπηρεσιών και εξισορρόπηση φορτίου,
- Ευθυγράμμιση τρισδιάστατου ανάγλυφου μοντέλου εδάφους του ελλαδικού χώρου,
- Εξαγωγή ασπρόμαυρης και έγχρωμης εικόνας height map μέσω μετατροπής ισοϋψών δεδομένων,
- Διασύνδεση γεωτρήσεων και υπόγειων Υδροφορέων μέσω αλγορίθμων τριγωνοποίησης σημείων στο χώρο.
- Γεννήτριες Φορτίου Προσομοίωσης Χρηστών

1.4 Δομή Μεταπτυχιακής Διατριβής

Στο πρώτο μέρος της εργασίας αναλύονται οι βασικές έννοιες και διατυπώνονται κάποιοι ορισμοί οι οποίοι είναι απαραίτητοι για την κατανόηση της διατριβής.

Στη συνέχεια, στο Δεύτερο Κεφάλαιο, περιγράφονται οι αλγόριθμοι που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής. Επίσης παρουσιάζεται το θεωρητικό υπόβαθρο για θέματα εξυπηρετητών και υπολογιστικού νέφους.

Στο Τρίτο Κεφάλαιο, αναλύεται η διαδικασία υλοποίησης της εφαρμογής καθώς και ο τρόπος διασύνδεσης και λειτουργικότητας των επί μέρους συστημάτων του συστήματος.

Στο Τέταρτο κεφάλαιο παρουσιάζονται τα αποτελέσματα από την πειραματική διαδικασία παραγωγής μετρήσεων υπό μορφή γραφημάτων και περιγραφής των συγκριτικών αποτελεσμάτων.

Τέλος, στο Πέμπτο Κεφάλαιο, παρουσιάζονται οι μελλοντικές προοπτικές του συστήματος καθώς και η αξιοποίηση της υφιστάμενης διατριβής.

ΚΕΦΑΛΑΙΟ 2 ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

2.1 Βασικές Έννοιες-Ορισμοί

Πριν αναλυθεί η εργασία παρατίθονται κάποιοι ορισμοί, οι οποίοι αναφέρονται συχνά στο κείμενο της Διπλωματικής διατριβής.

Λεκάνη απορροής (*Catchment*):

Ο γεωμετρικός τόπος των σημείων από τα οποία ένα επιφανειακό ή/και ένα υπόγειο υδάτινο σώμα συγκεντρώνει («κληρονομεί) τους υδατικούς πόρους του. **[18]**

Υδροφόρος Ορίζοντας (*Aquifer*):

Ένα σώμα από διαπερατά πετρώματα (π.χ ενοποιημένα τμήματα χαλικιού ή μία στρώση άμμου), το οποίο προστατεύεται από αδιαπέραστο υλικό και είναι ικανό να συγκρατήσει σημαντικές ποσότητες νερού μέσα από κινήσεις υπόγειων υδάτων. **[18]**

Γεώτρηση / Πηγάδι (*Borehole/Well*):

Μία περιστροφική τρύπα στην επιφάνεια της γης, η οποία διανοίγεται για να επιτρέψει την αξιολόγηση των χαρακτηριστικών ενός γεωλογικού σχηματισμού και των υγρών που περιέχονται σε αυτόν (πχ υπόγειων υδάτων). Συνήθως αποκαλείται και ως πλήρης γεώτρηση. **[18]**

"Φίλτρα" (*well screens*):

Ένα σύστημα ελεγχόμενου πλέγματος (ή τρύπες) που έχει σχεδιαστεί για να επιτρέπει την είσοδο του νερού σε μία γεώτρηση χωρίς την αδικαιολόγητη απώλεια κάποιου τμήματος, αλλά για να απομακρυνθεί η άμμος, η λάσπη ή άλλα αιωρούμενα στερεά υλικά από το νερό. **[18]**

Είναι σημαντικό να αναφερθεί επίσης ότι οι γεωτρήσεις αφού διανοιχτούν, σωληνώνονται. Σύμφωνα με τις οδηγίες του επιβλέποντα γεωλόγου, αξιολογείται σε ποια βάθη θα βρεθεί νερό (αυτό εξαρτάται από τα χαρακτηριστικά των πετρωμάτων) και προτείνεται η τοποθέτηση των απαραίτητων σωλήνων. Για τα βάθη που δεν υπάρχει νερό (απουσία

υδροφορίας), τοποθετείται "τυφλό", δηλαδή απλοί σωλήνες. Για εκείνα τα βάθη που προβλέπεται υδροφορία στους γεωλογικούς σχηματισμούς, τοποθετούνται σωλήνες με φίλτρο.

Για λόγους κατανόησης και ενιαίας ορολογίας, κάνουμε την παραδοχή ότι όταν αναφέρεται ο όρος «φίλτρο» στο κείμενο εννοούμε τα βάθη εκείνα μέσα στα πηγάδια όπου αναπτύσσεται ο υδροφόρος ορίζοντας.

2.2 Γεωδαιτικό Σύστημα Αναφοράς Ε.Γ.Σ.Α. ’87

Στην συγκεκριμένη εφαρμογή θα απεικονίζονται σε Χώρο Τριών Διαστάσεων το Γεωγραφικό Ανάγλυφο του Ελλαδικού Χώρου καθώς και οι Γεωτρήσεις στον χώρο αυτό.

Για να υλοποιηθεί η υποστήριξη αυτής της δυνατότητας είναι προαπαιτούμενη η γεωαναφορά των χωρικών δεδομένων.

Κάθε εφαρμογή διαχείρισης και παρουσίασης χωρικών δεδομένων πρέπει να έχει ένα επίπεδο (layer) το οποίο σχετίζεται με τον γεωγραφικό προσδιορισμό των συντεταγμένων. Το επίπεδο αυτό είναι το υπόβαθρο στο οποίο διαρθρώνεται και απεικονίζεται η γεωγραφικά προσδιορισμένη πληροφορία.

Ως Γεωδαιτικό Σύστημα Αναφοράς ορίζουμε την επιλογή ενός Γεωδαιτικού (Datum) συστήματος, που δίνει αρχικές συντεταγμένες σε σημεία και τις διαστάσεις ενός ελλειψοειδούς αναφοράς [1].

Κατά την χρονική εξέλιξη έχουν αναπτυχθεί και παρουσιαστεί διάφορα γεωγραφικά πρότυπα, τα οποία στις περισσότερες των περιπτώσεων εξυπηρετούν τις ανάγκες κάθε χώρας. Ο λόγος για τον οποίο χρειαζόμαστε ένα Γεωδαιτικό Σύστημα Αναφοράς είναι για να έχουμε ένα κοινό Γεωγραφικό Σύστημα, προκειμένου να προσδιορίζουμε στο χώρο συγκεκριμένα σημεία με την χρήση των συντεταγμένων.

Οι Συντεταγμένες προσδιορίζουν μονοσήμαντα τη γεωγραφική θέση ενός σημείου, επομένως ένα σημείο στο χώρο αποτελείται από ένα μόνο ζεύγος σημείων. Οι Συντεταγμένες σε Καρτεσιανές και Γεωδαιτικές Συντεταγμένες.

Ως καρτεσιανές συντεταγμένες ορίζουμε τις συντεταγμένες σε ορθοκανονικό σύστημα αξόνων, ενώ τις γεωδαιτικές τις ορίζουμε ως τις γωνίες που ορίζουν την θέση ενός σημείου στην επιφάνεια της ελλειψοειδούς αναφοράς.

Τα συστήματα των συντεταγμένων που χρησιμοποιούνται στη Γεωδαισία στα συστήματα αναφοράς είναι: οι καρτεσιανές τρισδιάστατες συντεταγμένες (X,Y,Z), και οι ελλειψοειδείς συντεταγμένες (φ,λ,h)

Στην Ελλάδα εφαρμόζονται διάφορα γεωδαιτικά συστήματα [2]. Το πιο διαδεδομένο είναι το ΕΓΣΑ ’87 καθώς είναι το πρότυπο αυτό με το οποίο έχουν αναπτυχθεί οι περισσότερες εφαρμογές που κάνουν χρήση γεωγραφικών δεδομένων στην Ελλάδα όπως το κτηματολόγιο.

Το ΕΓΣΑ ‘ 87 είναι το πιο πρόσφατο προβολικό σύστημα στην Ελλάδα και είναι αποτέλεσμα συνεργασίας του Εργαστηρίου Ανώτερης Γεωδαισίας του Τμήματος Αγρονόμων Τοπογράφων του Ε.Μ.Π, της Γεωγραφικής Υπηρεσίας Στρατού και του ΟΚΧΕ.

Το ΕΓΣΑ χρησιμοποιεί σαν ελλειψοειδές αναφοράς το GRS80, εξ ορισμού προσανατολισμένο παράλληλα **[1]** με το ITRF89 (International Terrestrial Reference Frame -1989). Γεωκεντρικές συντεταγμένες στο δίκτυο δόθηκαν από την υπολογισμένη στο ITRF89 θέση του κεντρικού βάθρου του σταθμού του Διονύσου. Ο προσανατολισμός και η κλίμακα δόθηκαν (ΒΕΗΣ, 1987) από δίκτυο 6 σταθμών laser, οι οποίοι συνδέθηκαν με το υπόλοιπο δίκτυο με δορυφορικές μετρήσεις. Η θέση του μετατεθειμένου (ως προς το ITRF89) γεώκεντρου έχει υπολογισθεί ώστε η επιφάνεια του ελλειψοειδούς να είναι η βέλτιστη για την Ελλάδα. Σαν προβολικό σύστημα χρησιμοποιείται σε μία μόνο ζώνη, με κεντρικό μεσημβρινό στις 24º - δηλαδή ακριβώς στην γραμμή διαχωρισμού των ζωνών 34 και 35 του UTM ,και λοιπές παραμέτρους όπως το UTM. **[1]**

Ορίζεται από τις συντεταγμένες του Κεντρικού Βάθρου:

φ=38^{o} 04’ 33"8107

λ=23^{o} 55’ 51"0095, και

h=481,743m και

Αυτό σημαίνει ότι το κέντρο του ελλειψοειδούς αναφοράς (GRS 80) που χρησιμοποιήθηκε μετατοπίστηκε παράλληλα ως προς το γήινο σύστημα κατά :

ΔX=+199,652m

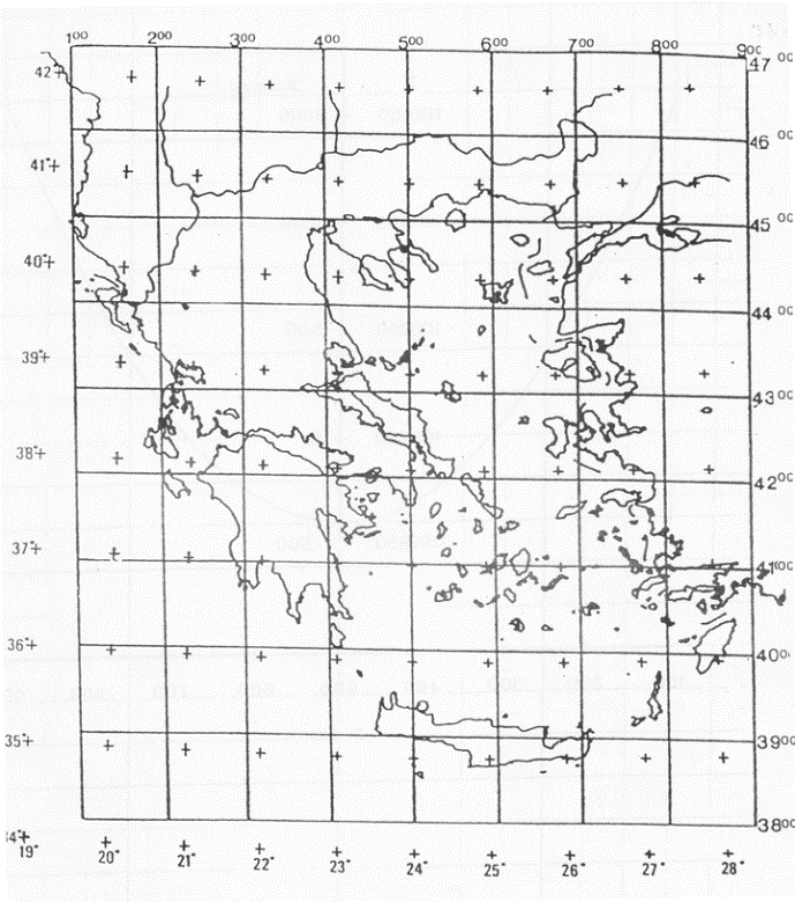
ΔY= - 74,759m

ΔZ = -246,055m

ώστε να ταιριάζει καλύτερα στο γεωειδές της Ελλάδας.

Για προβολικό σύστημα επιλέχθηκε η Εγκάρσια Μερκατορική Προβολή σε μία ζώνη. Για την επιλογή του κεντρικού μεσημβρινού λ_0 και της κλίμακας σε αυτόν K_0 έγινε μία βελτιστοποίηση ώστε να ελαχιστοποιούνται οι παραμορφώσεις κλίμακας στην έκταση της Ηπειρωτικής Χώρας. Το αποτέλεσμα ήταν $\lambda_0 = 24^0$ και $K_0 = 0,9996$. [2]

Οι παραμορφώσεις με αυτόν τον τρόπο μπορούν να φτάσουν μέχρι και το ένα (1) μέτρο (1m) σε απόσταση 1 χλμ (1 Km). Για να αποφευχθούν αρνητικές τιμές το σύστημα αυτό έχει ως τετμημένη 500.000 μέτρα και αρχή των τεταμένων θεωρείται ο ισημερινός ($\phi=0^0$)



Εικόνα 2.1 *Γεωγραφικός Προσδιορισμός Ελλάδας σε Γεωγραφικό Πρότυπο ΕΓΣΑ ' 87 [2]*

2.3 Επεξεργασία Δεδομένων της μορφής `.asc`

Στα γεωγραφικά συστήματα υπάρχουν δύο μοντέλα δεδομένων που χρησιμοποιούνται ευρέως. Αυτά είναι τα δεδομένα Raster και τα δεδομένα Vector. Η διαφορά τους είναι στον τρόπο με τον οποίο αποθηκεύουν και αναπαριστούν τα δεδομένα.

Στα δεδομένα Vector οι πληροφορίες που θέλουμε να αποθηκεύσουμε - μοντελοποιήσουμε ορίζονται ως ζεύγη συντεταγμένων. Αυτά μπορεί να είναι δεδομένα σημείων (point), πολυγώνων (polygons), ευθειών (polylines), ή και ακόμα περιοχές (area). Αυτά τα δεδομένα μπορούν να αποθηκευτούν σε Γεωγραφικές Βάσεις Δεδομένων (Spatial Databases) για χρήση και επεξεργασία.

Στα δεδομένα Raster οι πληροφορίες υπάρχουν σε ένα αρχείο, το οποίο τις περισσότερες φορές είναι ένα αρχείο εικόνας περιλαμβάνοντας δεδομένα, DEM (Data Elevation Model), δηλαδή δεδομένα ύψους.

Τα γεωγραφικά δεδομένα προέρχονται από αεροφωτογραφίες, δορυφορικές φωτογραφίες και αποθηκεύονται υπό την μορφή JPEG2000, GeoTiff, Png, JPG και συνοδεύονται με ένα αρχείο συνήθως της μορφής .PRJ.

Το αρχείο αυτό είναι ένα κείμενο το οποίο αναφέρει τη γεωαναφορά, δηλαδή την κλίμακα εύρους των συντεταγμένων και το προβολικό σύστημα της φωτογραφίας. Σαν δεδομένα Raster όμως ορίζονται και τα αρχεία .ASC που είναι αρχεία κειμένου.

Η τελική μορφή των .asc αρχείων, περιέχει δεδομένα σε μορφή ASCII, όπου μας παρέχει δύο είδη πληροφοριών. Η μορφοποίηση των δεδομένων αυτών έχει ως ακολούθως :

```
NCOLS 480
NROWS 450
XLLCORNER 378922
YLLCORNER 4072345
CELLSIZE 30
NODATA_VALUE -9999
43 2 45 7 3 56 2 5 23 65 34 6 32 54 57 34
35 45 65 34 2 6 78 4 2 6 89 3 2 7 45 23 5 ...
```

Το πρώτο μέρος του αρχείου παρέχει μετά – δεδομένα (meta-data) δηλαδή τον αριθμό των στηλών και των γραμμών του πίνακα των δεδομένων , το που βρίσκονται οι συντεταγμένες των σημείων στο κάτω αριστερά και πάνω δεξιά σημείο, πόσο είναι η απόσταση του βήματος δηλαδή η απόσταση μεταξύ δύο διαδοχικών σημείων και, τέλος, μία σταθερή τιμή που είναι το -9999 που αντιστοιχεί στην τιμή όπου δεν υπάρχουν δεδομένα ύψους (π.χ Θάλασσα).

Στο δεύτερο μέρος περιέχονται όλες εκείνες οι τιμές που περιγράφονται στα μετά – δεδομένα με κάθε τιμή να αντιστοιχεί στις τιμές των ισοϋψών.

2.4 Τριγωνοποίηση - Δημιουργία Αντικειμένων

2.4.1 Αλγόριθμος Ταξινόμησης Γεωτρήσεων

Κάθε γεώτρηση υπάρχει στο χώρο σε συγκεκριμένες συντεταγμένες σύμφωνα με το προβολικό σύστημα ΕΓΣΑ ’87. Επειδή κάθε γεώτρηση μπορεί να απέχει μεγάλη απόσταση από την πλησιέστερη σε αυτήν και ενδιάμεσα να παρεμβάλλεται μία άλλη γεώτρηση είναι αναγκαίο να θέσουμε τις γεωτρήσεις σε σωστή γεωγραφική σειρά προκειμένου η σύνδεση των φίλτρων που γίνεται στο επόμενο στάδιο της εφαρμογής να μπορεί να δέχεται τις συντεταγμένες με διατεταγμένη σειρά προκειμένου να ενωθούν οι γεωτρήσεις με τα σχετιζόμενα φίλτρα τους.

Η ταξινόμηση των γεωτρήσεων έγινε ως ακολούθως :

Αρχικά επιλέχθηκε ένα σημείο εκκίνησης που αντιστοιχεί στο νότιο - ανατολικότερο σημείο προκειμένου να υπάρχει μία βάση και να υπολογιστούν τα υπόλοιπα σημεία σύμφωνα με αυτό.

Στη συνέχεια επιλέχτηκε η κοντινότερη σε νότιο – ανατολικό σημείο γεώτρηση και κατά τον ίδιο τρόπο συνεχίστηκε η διαδικασία ψάχνοντας με τους δείκτες του ρολογιού την πιο κοντινή γεώτρηση έχοντας κάθε φορά σαν σημείο Αναφοράς την εκάστοτε Γεώτρηση.

Με αυτόν τον τρόπο όταν ολοκληρωθεί η διαδικασία έχει δημιουργηθεί μία λίστα η οποία παρέχει αναδιατεταγμένα τα γεωγραφικά σημεία των γεωτρήσεων σε αύξουσα σειρά επιλέγοντας ως σημείο εκκίνησης την νότιο – ανατολικότερη γεώτρηση.

2.4.2 Αλγόριθμος Τριγωνοποίησης Ear Clipping

Η τριγωνοποίηση συνήθως χρησιμοποιείται στα γραφικά υπολογιστών για την δημιουργία γεωγραφικού Ανάγλυφου **[3]** και σε εφαρμογές στις οποίες έχουμε ένα σύνολο σημείων

(Point Cloud) και επιθυμούμε να ενώσουμε τα σημεία αυτά έτσι ώστε να φαίνονται σαν ένα ενιαίο τμήμα όπως ένα πολύγωνο ή την αντίστροφη διαδικασία δηλαδή τη διαδικασία αποσύνθεσης ενός πολύγωνου σε ένα σύνολο τριγώνων. Οι ακμές του πολύγωνου περιέχουν τις κορυφές του τριγώνου.**[3]**

Για την τριγωνοποίηση - ένωση των φίλτρων των γεωτρήσεων χρησιμοποιήθηκε ο αλγόριθμος Ear Clipping. Σύμφωνα με το Θεώρημα του αλγορίθμου αυτού, κάθε πολύγωνο P με τουλάχιστον τέσσερις κορυφές n ≥ 4 χωρίς κενά (holes) έχει τουλάχιστον δύο «αυτιά» όπου είναι και τα τρίγωνα με δύο πλευρές του πολύγωνου ενώ η τρίτη πλευρά είναι εσωτερική. Έπειτα ο αλγόριθμος ψάχνει για να βρει ένα «αυτί» όπου το αφαιρεί από το πολύγωνο δημιουργώντας ένα νέο πολύγωνο. Αυτή η διαδικασία επαναλαμβάνεται έως ότου μείνει μόνον ένα τρίγωνο.

Ο αλγόριθμος αυτός προτάθηκε και εξηγήθηκε από τον Joseph O' Rourke **[3]**.

Για την τοποθέτηση ενοποιημένων τμημάτων ήταν αναγκαία η ανεύρεση των διαδοχικών κοντινότερων σημείων στο χώρο καθώς και η μεταξύ τους γωνία σε σχέση με το επίπεδο προκειμένου η σύνδεση των σημείων να είναι ταξινομημένη και με τη σωστή διεύθυνση.

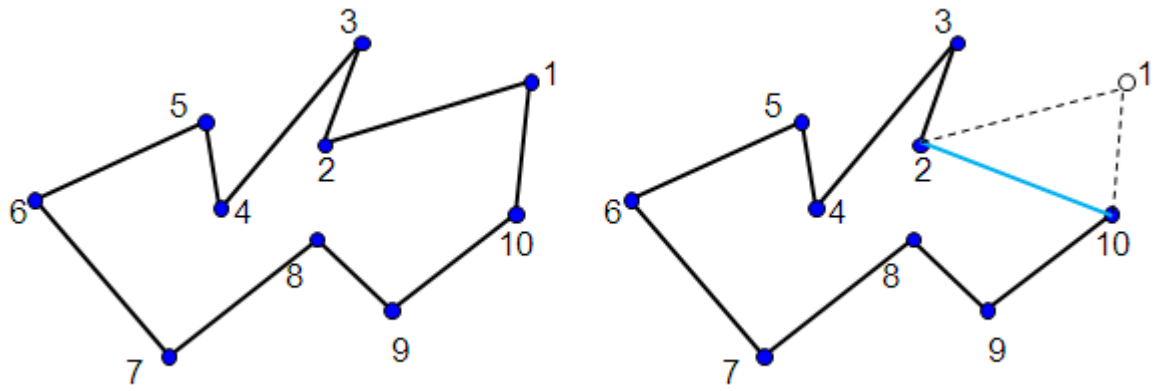
Αλγόριθμος Ear Clipping

Initialize the ear trip status of each vertex

```
While n>3 do
     Locate an ear tip Vi
     Delete Vi
     Update the ear trip status of Vi-1 and Vi+1
```

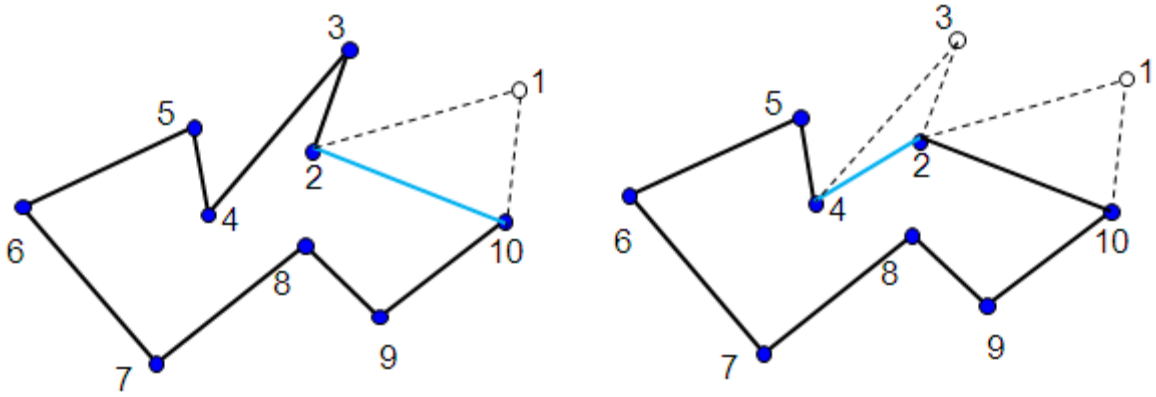
Παράδειγμα εφαρμογής Αλγόριθμου Ear Clipping

Έστω ένα πολύγωνο που αποτελείται από 10 σημεία. Σύμφωνα με τον αλγόριθμο αυτό το σημείο 1 απομακρύνεται και πλέον το Τρίγωνο T_1={10,1,2} είναι το πρώτο τρίγωνο της μεθόδου αυτής και η λίστα με τα σημεία της διαγώνιου είναι το Δ = {(2,10)} όπως φαίνεται στο σχήμα 2.1 .



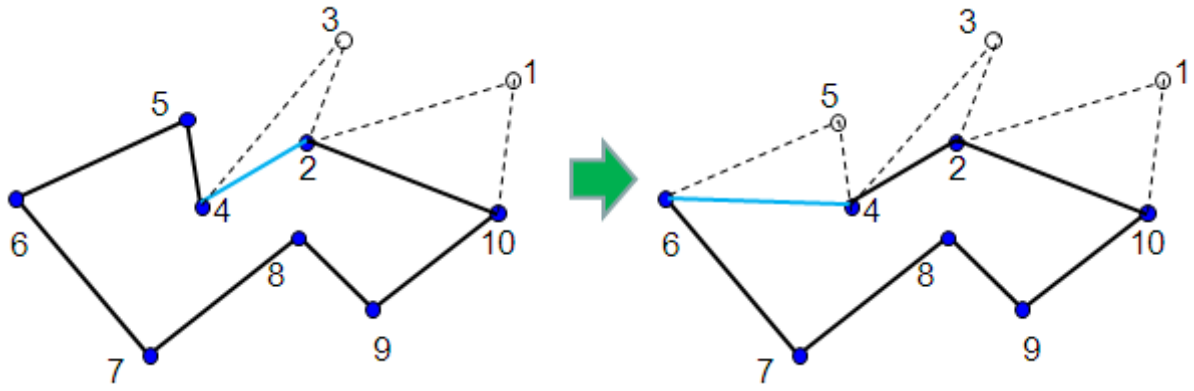
Σχήμα 2.1 *Απομάκρυνση σημείου 1*

Στην συνέχεια αν απομακρυνθεί το σημείο 3 θα έχουμε Τ ={(10,1,2),(4,3,2)} και Δ = {(2,10),(2,4)} όπως φαίνεται στο Σχήμα 2.2 .

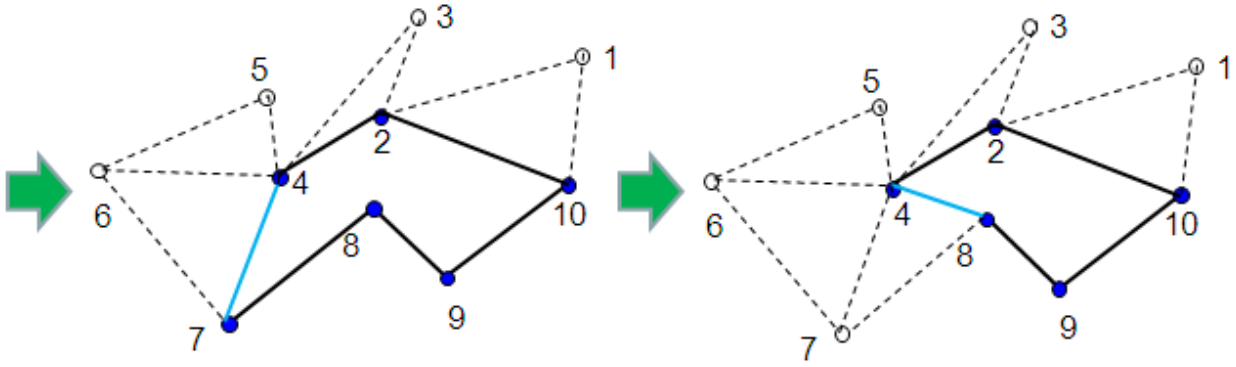


Σχήμα 2.2 *Απομάκρυνση σημείου 3*

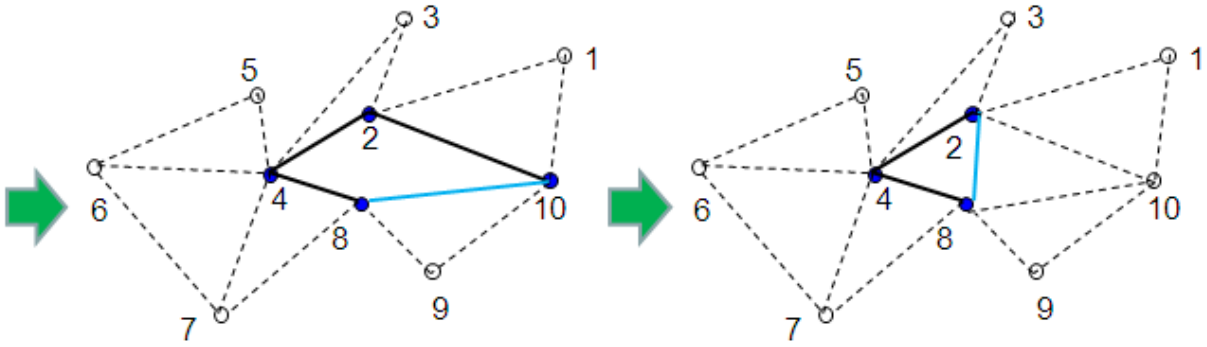
Ακολουθώντας την ίδια μέθοδο έχουμε απομάκρυνση του σημείου 5 όπως φαίνεται στο σχήμα 2.3 , απομάκρυνση του σημείου 6,7 στο σχήμα 2.4 και απομάκρυνση των σημείων 9,10 του σχήματος 2.5 .



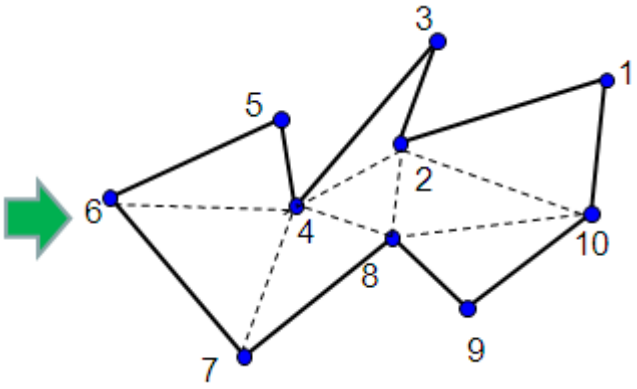
Σχήμα 2.3 *Απομάκρυνση σημείου 5*



Σχήμα 2.4 *Απομάκρυνση σημείων 6,7*



Σχήμα 2.5 *Απομάκρυνση σημείων 9,10*



Σχήμα 2.6 *Τέλος Αλγόριθμου Ear Clipping*

Οπότε όπως φαίνεται και στο σχήμα 2.6 έχουμε Δ = {(2,10),(2,4),(4,6),(4,7),(4,8),(8,10),(2,8)} και T={(2,1,10),(4,3,2),(6,5,4),(6,4,7),(7,4,8),(8,9,10),(2,10,8),(4,2,8)}.

Ο συγκεκριμένος αλγόριθμος παρουσιάζει προβλήματα στη σύνδεση σημείων μεγάλης απόστασης. Συγκεκριμένα δημιουργείται ένα μεγαλύτερο τμήμα από το αναμενόμενο.

Ένας άλλος αλγόριθμος τριγωνοποίησης, ο οποίος έχει ευρεία χρήση και πολύ καλά αποτελέσματα είναι ο αλγόριθμος Delaunay **[4]** ο οποίος θα συνδέσει και θα δημιουργήσει ένα ενιαίο πλέγμα τριών διαστάσεων.

2.5 Εξυπηρετητές (Web Servers)

2.5.1 Σύγκριση Εξυπηρετητών

Οι εξυπηρετητές ή διακομιστές (Web Server) είναι υλικό ή λογισμικό το οποίο μας παρέχει τη δυνατότητα διαφόρων υπηρεσιών που βασίζονται στην αρχιτεκτονική πελάτη - εξυπηρετητή (Client-Server) μέσω του πρωτοκόλλου HTTP, δηλαδή μέσω δικτύου.

Ο εξυπηρετητής (Server) στεγάζει τα αρχεία του σε ένα συγκεκριμένο αποθετήριο και τα διαθέτει στον πελάτη (Client) όποτε εκείνος ζητήσει τα αρχεία αυτά υπό την μορφή διεύθυνσης δικτύου (URL).

Υπάρχουν διάφοροι τύποι εξυπηρετητών οι οποίοι είναι ανεπτυγμένοι από Εταιρίες Πληροφορικής .

Κάθε εξυπηρετητής μπορεί να υποστηρίξει διαφορετικές τεχνολογίες και παρέχει ένα σύνολο από λειτουργίες που εξασφαλίζουν την ασφάλεια των δεδομένων καθώς και την ταχύτητα της εκτέλεσης και μεταφοράς των δεδομένων.

Στους ακόλουθους πίνακες συγκρίνονται διαφορετικοί τύποι εξυπηρετητών σύμφωνα με τις εταιρίες που τους έχουν χρησιμοποιήσει, το λογισμικό το οποίο μπορεί να τους υποστηρίξει, τις Βάσεις Δεδομένων που μπορούν να φέρουν , τις γλώσσες προγραμματισμού που δέχονται, όπως επίσης και τους αλγόριθμους Εξισορρόπησης Φορτίου που έχουν υλοποιημένους, και τις ημερομηνίες όπου έχουν κάνει την τελευταία ενημέρωσή τους.

Εταιρίες	Λειτουργικό Σύστημα	Βάσεις Δεδομένων	Αλγόριθμοι	Γλώσσες Προγραμματισμού	Τελευταία Έκδοση
Facebook	FreeBSD 3 -10	MySQL	Round Robin	PHP	1.4.4 (2013/11/19)
Pinterest	FreeBSD 5 — 10	Postgresql	Weight Round Robin	Python	1.5.8 (2013/12/17)
WordPress.com	Solaris 9,10	MariaDB	Least Connection	Perl	
GitHub	AIX 7.1				
Zynga	HP-UX 11.31 / ia64				
Eventbrite	Mac OS X / ppc, i386				
NetDNA	Windows XP, Windows Server 2003				
Hulu					
NetFlix					
Dropbox					

Πινακας 2.6 *Χαρακτηριστικά εξυπηρετητή NGiNX*

Λειτουργικό Σύστημα	Βάσεις Δεδομένων	Αλγόριθμοι	Γλώσσες Προγραμματισμού	Τελευταία Έκδοση
Red Hat	MySQL	Request Counting	PHP	2.4
CentOs	Postgresql	Weight Traffic Counting	Python	2.2
Windows	MariaDB	Pending Request Counting	Perl	2.0

Mac Os	Cassandra		Tcl	
	CouchDB			
	Hadoop			

Πίνακας 7.2 *Χαρακτηριστικά εξυπηρετητή Apache*

Λειτουργικό Σύστημα	Βάσεις Δεδομένων	Αλγόριθμοι	Γλώσσες Προγραμματισμού	Τελευταία Έκδοση
Windows	MySql	**Weighted round robin**	PHP	8
	SQL Server	**Weighted total traffic**	C++	7.5
		Least current request	C#	7
		Least response time	VB	
		Server variable hash	F#	
		Query string hash	ASP.NET	
		Request hash		

Πίνακας 2.8*Χαρακτηριστικά εξυπηρετητή IIS*

Server	Basic Access Authentication	SSL/TLS	CGI	FCGI	Java Servlets	Administration Console	IPv6
Apache	√	√	√	√	X	√	√
IIS	√	√	√	√	X	√	√
NGiNX	√	√	√	√	√	√	√

Πίνακας 2.9 *Χαρακτηριστικά Εξυπηρετητών*

Server	Windows	Linux	OS X	BSD	Solaris	Open VMS	IBM	HP-UX
Apache	√	√	√	√	√	√	√	√
IIS	√	X	X	X	X	X	X	X
NGiNX	√	√	√	√	√	X	X	√

Πίνακας 2.5 *Λειτουργία Εξυπηρετητών σε Λειτουργικά Συστήματα*

Σε αυτούς τους πίνακες βλέπουμε τα κυριότερα χαρακτηριστικά των εξυπηρετητών Διαδικτύου.

Επιλέχτηκε ο NGiNX καθώς υπερτερεί σε ταχύτητα έναντι των Apache και IIS, και επειδή μας παρέχει επιπλέον κάποια χαρακτηριστικά τα οποία είναι ιδιαίτερα σημαντικά για τον τύπο και την μορφή της εφαρμογής. Αυτά είναι η αναγνώριση τοποθεσίας του χρήστη και αν η υποδομή στεγάζεται κάτω από την αρχιτεκτονική ενός εξισορροπητή Φορτίου (Load Balancer) τότε γίνεται αναγνώριση του πλησιέστερου εξυπηρετητή και εξυπηρετεί εκείνος τον χρήστη. Για αυτό το λόγο προτιμάται από μεγάλες εταιρίες οι οποίες έχουν σαν κύριο χαρακτηριστικό τους την εύρεση της τοποθεσίας του χρήστη για να μειώσουν το κόστος μεταφοράς δεδομένων λόγω της απόστασης που προκαλεί μία καθυστέρηση αν υποθέσουμε ότι έχουμε την ίδια υποδομή σε όλο το δίκτυο.

Επιπλέον ο NGiNX κατόπιν ολοκλήρωσης του προγράμματος παρατηρήθηκε ότι διόρθωνε ένα πρόβλημα που υπήρχε ή καλύτερα δεν το εμφάνισε ποτέ και αυτό είχε να κάνει με απώλειες στη μνήμη του συστήματος στεγάζοντας την εφαρμογή των τριών διαστάσεων σε αυτόν σε σύγκριση με τον IIS, ο οποίος προκαλούσε μία επαναλαμβανόμενη και συνεχή αύξηση του φορτίου της Μνήμης χωρίς να ελαττώνεται ποτέ με αποτέλεσμα η εφαρμογή να τελματώνει μετά από ένα χρονικό διάστημα λόγω του συγκεκριμένου προβλήματος.

Επίσης ο NGiNX είναι ευρέως διαδεδομένος για τις ταχύτητες που προσφέρει ,καθώς θεωρείται από τους γρηγορότερους εξυπηρετητές διαδικτύου.

2.5.2 Εξισορροπητές Φορτίου (Load Balancers)

Η εξέλιξη της τεχνολογίας αλλά και η συνεχής και αδιάκοπη προσθήκη δεδομένων, καθώς και η ανάπτυξη των εφαρμογών σε μεγάλη κλίμακα και σε παγκόσμιο επίπεδο έχει σαν αποτέλεσμα να ζητούνται συνεχώς υπολογιστικοί πόροι οι οποίοι θα συνδυάζονται μεταξύ τους προκειμένου να έχουμε γρηγορότερα αποτελέσματα όσον αφορά τους χρόνους

απόκρισης των δεδομένων αλλά και να παρέχουν καλύτερες και γρηγορότερες υπηρεσίες στους χρήστες.

Για να είναι δυνατόν να υποστηριχτούν αυτές οι λειτουργίες αλλά και οι τεράστιες ποσότητες δεδομένων (Big Data), χωρίς να εμφανίζονται καθυστερήσεις και προβλήματα, απαιτείται μία πληθώρα εργαλείων αλλά και μεγάλη ποσότητα σε υλικό (hardware).

Για αυτόν το λόγο για να είναι δυνατή η υποστήριξή τους χρειαζόμαστε παραπάνω από έναν εξυπηρετητές διαδικτύου (Web Servers) οι οποίοι μπορεί να βρίσκονται σε διαφορετικά μηχανήματα και διαφορετικά σημεία σε όλο τον κόσμο.

Αν χρησιμοποιούσαμε έναν μόνο εξυπηρετητή αυτό θα είχε ως αποτέλεσμα η εφαρμογή να είναι προβληματική και ο τελικός χρήστης να έμενε δυσαρεστημένος από την εμπειρία της εφαρμογής και αυτό θα οδηγούσε στην παραγκώνιση του εκάστοτε λογισμικού αφού δεν θα είχε το επιθυμητό αποτέλεσμα.

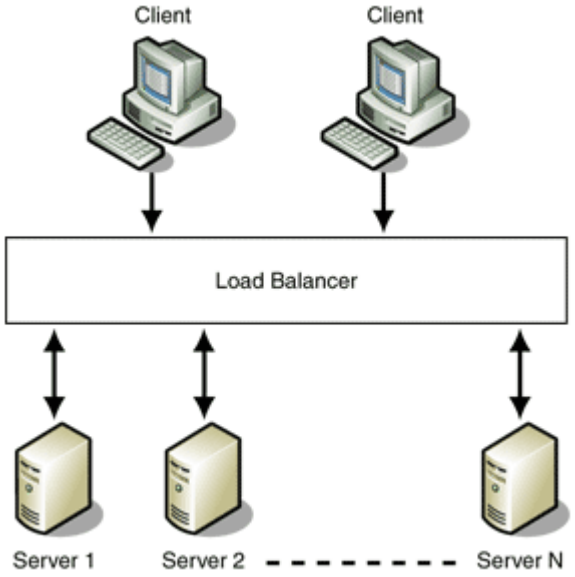
Επομένως, η τεχνολογία και οι απαιτήσεις της σύγχρονης πραγματικότητας οδήγησαν στην δημιουργία μίας Φάρμας Εξυπηρετητών (Server Farm) καθώς η απάντηση στο ερώτημα πόσους εξυπηρετητές χρειάζεται μία εφαρμογή, είναι ένα από τα πιο δύσκολα ερωτήματα αφού η δυναμική και η αβέβαιη φύση της εφαρμογής προκαλεί αβέβαια αποτελέσματα στο φορτίο που απαιτείται κάθε φορά. **[5]**

Μία ακόμα δύσκολη κατάσταση έγκειται στο γεγονός ότι πολλά συστήματα στην υποδομή μας μπορεί να είναι σε ετερογενή υποδομή, οπότε χρειαζόμαστε μία λύση η οποία να είναι ανεξάρτητη του λογισμικού, δηλαδή να μπορεί να υποστηρίζει ετερογενή συστήματα.

Αυτή η λύση έρχεται με τους εξισορροπητές φορτίου (Load Balancer). Αυτές οι λύσεις παρέχονται τόσο σε υλικό (Hardware) όσο και σε λογισμικό (Software).

Οι κατανεμητές φορτίου μπορούν να διαχειρίζονται ένα πολύ μεγάλο πλήθος αιτημάτων (web Requests). Έχουν την δυνατότητα να ελέγχουν την υποδομή, και ανάλογα με τον αλγόριθμο που έχουν επιλεγμένο να διαχειρίζονται τους πόρους τους ανάλογα, προκειμένου σε όλη τη διάρκεια που η εφαρμογή είναι ενεργοποιημένη να βρίσκονται σε μία συνέργεια και όλοι να αντιμετωπίζονται ισόποσα.**[6]**

Το σημαντικότερο γεγονός είναι ότι αυτά τα λογισμικά είναι ανεξάρτητα της τελικής εφαρμογής καθώς στην διαχείρισή τους δηλώνονται οι διευθύνσεις και ο αλγόριθμος που θα εκτελούν ο οποίος εξαρτάται κάθε φορά από τις απαιτήσεις μας.



Εικόνα 2.2 *Αρχιτεκτονική εξισορροπητή φορτίου*

Πηγή : http://msdn.microsoft.com/en-us/library/ff648960.aspx

Όπως φαίνεται και από την εικόνα 2.2 έχουμε N αριθμό εξυπηρετητών οι οποίοι διασυνδέονται μεταξύ τους μέσω ενός εξισορροπητή φορτίου (Load Balancer). Οι πελάτες (clients) όταν θα πρέπει να εκτελούν κάποιο ερώτημα και να αντλούν δεδομένα της εφαρμογής στέλνουν ένα αίτημα στον κατανεμητή φορτίου και εκείνος ανάλογα με το φορτίο που υπάρχει σε κάθε εξυπηρετητή και ανάλογα το αλγόριθμο που χρησιμοποιεί στέλνουν το αίτημα στον αντίστοιχο εξυπηρετητή προκειμένου ο πελάτης να δεχτεί την απάντηση και τα αρχεία που επιθυμεί κάθε φορά.

2.5.3 Σύγκριση Αλγορίθμων Εξισσοροπητών Φορτίου

Round Robin	Least Connection	Destination Hashing	Never Queue	Request Counting	Least Loaded
Weight Round Robin	Weighted Least Connection	Source Hashing	Shortest Queue First	Weighted Traffic Counting	
	Locally-Based Least Connection			Pending Request Counting	

Πίνακας 2.10 *Αλγόριθμοι Εξισορρόπησης Φορτίου*

Υπάρχουν διαφορετικές τεχνολογίες εξισορροπητών φορτίου (Load Balancers) οι οποίες εξαρτώνται από τους διαφορετικούς αλγόριθμους οι οποίοι είναι υλοποιημένοι στο λογισμικό ή στο υλικό ανάλογα με την ιδιότητα και τις δυνατότητες του εκάστοτε εξισορροπητή φορτίου.

Αρχικά οι αλγόριθμοι αυτοί όπως φαίνονται και στο πίνακα 2.6 χωρίζονται στις εξής κατηγορίες :

Αλγόριθμος Round Robin(RR): Σύμφωνα με αυτόν τον αλγόριθμο ο εξισορροπητής μπορεί να διαχειριστεί διαφορετικούς εξυπηρετητές τυχαία, χωρίς να εξαρτάται κάθε φορά από το φορτίο που υπάρχει στους Server.**[7]**

Weighted Round Robin: Η τροποποίηση αυτού του αλγόριθμου βασίζεται στον Round Robin. Σύμφωνα με αυτόν τον αλγόριθμο σε κάθε εξυπηρετητή τίθεται ένας ακέραιος αριθμός που ανάλογα με την τιμή του προωθεί τα αιτήματα. Ο εξυπηρετητής που φέρει το μεγαλύτερο νούμερο είναι αυτός που θα εξυπηρετήσει πρώτος το αίτημα και αυτός που έχει το χαμηλότερο νούμερο τίθεται σε χαμηλότερη προτεραιότητα.

Αλγόριθμος Least Connection (LC): Σύμφωνα με αυτόν τον αλγόριθμο το νέο αίτημα θα εξυπηρετηθεί από τον εξυπηρετητή που έχει τις ελάχιστες HTTP συνδέσεις. Πρόκειται για έναν δυναμικό αλγόριθμο καθώς χειρίζεται τα αιτήματα ανάλογα με τον ελάχιστο αριθμό συνδέσεων που υπάρχουν σε κάθε εξυπηρετητή.**[7]**

Weighted Least Connection: Ο αλγόριθμος βασίζεται στον Least Connection του οποίου υπάρχει δυνατότητα να τεθεί η απόδοση υπό μορφή βάρους σε κάθε εξυπηρετητή. Οι εξυπηρετητές με μεγαλύτερο βάρος εξυπηρετούν μεγαλύτερο ποσοστό ενεργών συνδέσεων κάθε φορά.

Locally-Based Least Connection (LBLR): Σε αυτό τον αλγόριθμο εξυπηρετείται κάθε φορά ο χρήστης από μία λίστα εξυπηρετητών που σχετίζονται με την τοποθεσία της διεύθυνσης δικτύου. Έτσι ελέγχεται η διαθεσιμότητα των εξυπηρετητών και εξυπηρετείται από τον εξυπηρετητή με τον λιγότερο φορτίο.

Αλγόριθμος Destination Hashing: Ο αλγόριθμος αυτός αναθέτει τις συνδέσεις του δικτύου στους εξυπηρετητές ελέγχοντας το πίνακα hash ανάλογα με την διεύθυνση δικτύου (IP Address) του αποστολέα. Την ίδια λογική ακολουθεί και ο **Source Hashing** με την μόνη διαφορά ότι ελέγχεται η διεύθυνση του παραλήπτη.

Αλγόριθμος Never Queue: Ο αλγόριθμος αυτός κατανέμει το νέο αίτημα στον εξυπηρετητή ο οποίος είναι ανενεργός αντί να αναμένει να αναμένει εκείνος που μπορεί να είναι πιο γρήγορος, όταν δεν υπάρχει ανενεργός εξυπηρετητής το νέο αίτημα θα κατανεμηθεί στον εξυπηρετητή ο οποίος έχει την ελάχιστη καθυστέρηση.

Αλγόριθμος Request Counting: Ο αλγόριθμος αυτός κατανέμει τα αιτήματα σε όλους τους εξυπηρετητές και το νέο αίτημα εξυπηρετείται ανάλογα με τις ιδιότητες που έχει ο εκάστοτε εξυπηρετητής **[6]**

Weight Traffic Counting: Αυτός ο αλγόριθμος βασίζεται στον αλγόριθμο Request Counting. Επεξεργάζεται τα αιτήματα με παρόμοιο τρόπο αλλά ο μέγιστος αριθμός των αιτημάτων ανά εξυπηρετητή καθορίζεται από την κίνηση του δικτύου. **[6]**

Pending Request Traffic: Με αυτόν τον αλγόριθμο καταμετρώνται τα αιτήματα που υπάρχουν σε κάθε εξυπηρετητή και τα νέα αιτήματα ανατίθενται στον εξυπηρετητή εκείνον που έχει τα λιγότερα αιτήματα.

Αλγόριθμος Least Loaded (baseline): Ο αλγόριθμος αυτός αναθέτει το επόμενο αίτημα στον εξυπηρετητή ο οποίος έχει το ελάχιστο φορτίο. Το φορτίο αυτό καθορίζεται από το άθροισμα του χρόνου εξυπηρέτησης που είναι σε αναμονή στον εξυπηρετητή. Πρόκειται για έναν αλγόριθμο ο οποίος είναι δύσκολος στην υλοποίησή του καθώς οι χρόνοι αναπροσαρμόζονται συνέχεια και εξαρτώνται κάθε φορά από τον αριθμό των χρηστών που χρησιμοποιούν τον κάθε εξυπηρετητή.[7]

2.6 Το Σύννεφο (Cloud)

Με την εξέλιξη της τεχνολογίας, καθημερινά αναπτύσσονται εφαρμογές και ιστοσελίδες παγκοσμίου βεληνεκούς οι οποίες έχουν μεγάλη ανάγκη χρησιμοποίησης και συντήρησης τεράστιου όγκου δεδομένων που αντιμετωπίζουν μεγάλες δυσκολίες ως προς την λειτουργικότητα και την συντήρησή τους.

Η λύση σε αυτά τα προβλήματα έρχεται δια μέσου του Σύννεφου (Cloud). Με τη χρήση αυτής της τεχνολογίας, παρέχεται η δυνατότητα αύξησης της ασφάλειας των δεδομένων τόσο από πλευράς φυσικής ασφάλειας, όσο και από πλευράς απώλειας των δεδομένων, κάνοντας χρήση Βάσεων Δεδομένων που είναι προσαρμοσμένες στο Υπολογιστικό Νέφος και δύναται να κρατιούνται αντίτυπα όπου υπάρχει υλικό (Hardware) ή λογισμικό της υποδομής.

Πολύ βασικό χαρακτηριστικό αυτής της τεχνολογίας είναι ότι δίνεται η δυνατότητα, σύντομης και πολύ εύκολης επέκτασης της υποδομής, καθώς το μεγαλύτερο μέρος των εφαρμογών στεγάζεται σε εικονικές μηχανές (Virtual Machines) οι οποίες έχουν ευελιξία στην διαχείρισή τους.

Το Υπολογιστικό Νέφος επίσης προσφέρει τη δυνατότητα, η εφαρμογή να είναι διασυνδεδεμένη μέσω διαδικτύου σε πολλά σημεία σε ολόκληρο τον πλανήτη και κάθε φορά που χρειάζεται να εξυπηρετήσει κάποιο πελάτη να εξυπηρετείται δια μέσου του συστήματος που είναι πιο κοντά σε αυτόν (σε ιδεατό περιβάλλον).

Επομένως, με την χρήση της υποδομής του Σύννεφου (Cloud) αυξάνονται χαρακτηριστικά όπως η αξιοπιστία της εφαρμογής και του υλικού, η διαθεσιμότητα της εφαρμογής, η επεκτασιμότητα και η ασφάλεια καθώς το σύστημα είναι κατανεμημένο και δεν στεγάζεται μόνο σε έναν εξυπηρετητή.**[8]**

Ο ορισμός για το Υπολογιστικό Νέφος, σύμφωνα με το Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας των ΗΠΑ (NIST : National Institute of Standards and Technology) είναι ο ακόλουθος:

«Το Υπολογιστικό Νέφος είναι ένα μοντέλο που επιτρέπει την εύκολη πρόσβαση στο δίκτυο σε κοινόχρηστους πόρους, που μπορούν να τροφοδοτηθούν γρήγορα μέσω της ελάχιστης προσπάθειας διαχείρισης ή αλληλεπίδρασης από τους παρόχους» **[9]**

Χαρακτηριστικές εφαρμογές που κάνουν χρήση του Υπολογιστικού Νέφους είναι οι ιστοσελίδες κοινωνικής δικτύωσης, όπως τα Facebook, Twitter, Pinterest, Reddit, και StackOverFlow, όπου καθημερινά και σε πολύ σύντομο χρονικό διάστημα γίνονται ανανεώσεις σε ενημερώσεις. Οι χρήστες είναι από όλο τον κόσμο και ανανεώνονται συνέχεια με την πάροδο του χρόνου.

Αυτές οι εφαρμογές έχουν την υποδομή τους σε Υπηρεσίες Σύννεφου (Cloud) όπως το Amazon Web Services, Microsoft Azure,Google Cloud όπου διαχειρίζονται τα δεδομένα μέσω Βάσεων Δεδομένων προσαρμοσμένων σε μεγάλο όγκο δεδομένων όπως με τη χρήση των Hadoop και MongoDB **[8]**.

2.6.1 Μοντέλα Υπηρεσιών

Η τεχνολογία του Υπολογιστικού Νέφους αποτελείται από μία διαστρωμάτωση παρεχόμενων υπηρεσιών, οι οποίες μπορούν να καλύψουν τις ανάγκες όλων των πελατών καθώς παρέχουν μία πληθώρα επιλογών σε όλα τα επίπεδα.

Αποτελείται από τρία βασικά επίπεδα ξεκινώντας από το πιο χαμηλό επίπεδο προς το πιο υψηλό και αυτά είναι τα :

- IAAS (Infrastructure As A Service) / HAAS (Hardware As A Service),
- PAAS (Platform As A Service)
- SAAS (Software As A Service)

Το χαμηλότερο επίπεδο είναι το **HAAS** (Hardware As A Service), είναι το επίπεδο του Υλικού, το φυσικό επίπεδο, και συνήθως επιλέγεται από εταιρίες μεγάλου βεληνεκούς που έχουν μεγάλες απαιτήσεις σε υλικό.

Το επόμενο ενδιάμεσο επίπεδο είναι ο πυρήνας Λογισμικού (Software Kernel). Αυτό το επίπεδο λειτουργεί σαν γέφυρα μεταξύ του υλικού και του λογισμικού. Είναι το χαμηλότερο επίπεδο στο επίπεδο του λογισμικού και σαν σκοπό έχει την διαχείριση των πόρων των εξυπηρετητών. Τέτοιο λογισμικό αποτελούν οι εξισορροπητές φορτίου (Load Balancer).

Το επόμενο επίπεδο μετά το Software Kernel είναι η υποδομή του λογισμικού (Software Infrastructure).

Αυτό το επίπεδο περιλαμβάνει τους κύριους δικτυακούς πόρους, με σκοπό να διασυνδέσει το περιβάλλον του λογισμικού και τις εφαρμογές στους τελικούς χρήστες υπό την μορφή υπηρεσιών και αποκαλείται **IAAS (Infrastructure As A Service)**

Σε αυτό το επίπεδο προσφέρονται οι δυνατότητες υπολογιστικής ισχύος, αποθηκευτικού χώρου και δικτυακής υποδομή με τις οποίες οι πελάτες είναι σε θέση να αναπτύξουν και να φέρουν λογισμικό το οποίο ενσωματώνεται στα λειτουργικά συστήματα και σε εφαρμογές. **[10]**

Επίσης στους πελάτες που έχουν επιλέξει αυτή την υποδομή παρέχεται εικονικό υλικό όπως επεξεργαστική ισχύς (CPU), μνήμη (RAM), τροφοδοσία, και αποθηκευτικό μέσο υπό την μορφή διαχείρισης εικονικών μηχανών (Virtual Machines) και διαχείρισης αποθετηρίων. Τέτοιου τύπου υπηρεσίες είναι το AmazonEC2 και το S3. Οι χρήστες συνήθως πληρώνουν αυτές τις υπηρεσίες σύμφωνα με τη χρήση τους (pay-per-use basis).

Η επόμενη βαθμίδα στην πυραμίδα του Υπολογιστικού Νέφους είναι η Πλατφόρμα σαν υπηρεσία **PAAS (Platform As A Service).**

Μέσω αυτής της υπηρεσίας παρέχονται εφαρμογές ή μία πλατφόρμα ανάπτυξης με την οποία οι χρήστες έχουν την δυνατότητα να δημιουργήσουν τις δικές τους εφαρμογές που στεγάζονται και είναι λειτουργικές στο Υπολογιστικό Νέφος.

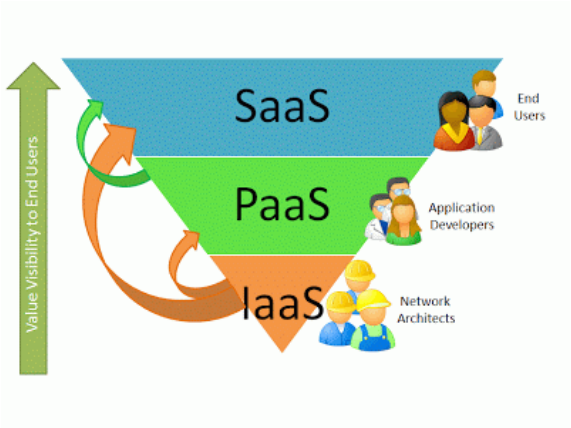
Συνήθως παρέχονται και λύσεις ενός Πλαισίου (framework) ή μέσω ενός συνόλου εντολών (API) τα οποία μπορούν να χρησιμοποιηθούν από τους πελάτες για να αναπτύξουν τις δικές τους εφαρμογές πάνω σε αυτή την υποδομή.

Χαρακτηριστικές λύσεις τέτοιου τύπου είναι το Google App Engine **[19]** και Microsoft Azure **[20]**.

Το τελευταίο και πιο υψηλό Επίπεδο είναι η υπηρεσία «Λογισμικό σαν Υπηρεσία» **(SAAS Software as a Service)**.

Σε αυτό το επίπεδο παρέχονται οι δυνατότητες στους πελάτες να χρησιμοποιούν τις εφαρμογές που λειτουργούν στην υποδομή του νέφους. Οι εφαρμογές είναι προσπελάσιμες από πολλούς πελάτες τόσο στον αριθμό όσο και στον τύπο των συσκευών. Επίσης, η σύνδεση σε αυτές τις εφαρμογές μπορεί να πραγματοποιηθεί από παντού και συνήθως είναι εφαρμογές διαδικτύου.

Χαρακτηριστικές λύσεις είναι τα Google Mail, DropBox και SalesForce.com **[10]**



Εικόνα 2.3*Επίπεδα Υπηρεσιών Υπολογιστικού Νέφους*

Πηγή : http://rapidsol.blogspot.gr/2013/11/what-is-cloud-computing-and-iaas-paas.html

2.6.2 Αξιοποίηση Τεχνολογιών υπολογιστικού νέφους

Το Σύννεφο εμφανίστηκε για να καλύψει πολλές ανάγκες που μέχρι πρότινος δεν ήταν δυνατό. Με τη χρήση του Σύννεφου λόγω του χαμηλού κόστους και των αυξημένων πλεονεκτημάτων που περιλαμβάνει όλο ένα και περισσότερες εταιρίες μεσαίου και υψηλού επιπέδου μεταβαίνουν σε υποδομή Σύννεφου και οι περισσότερες είναι ικανοποιημένες και έχουν μειώσει τα λειτουργικά τους έξοδα και ως αποτέλεσμα έχουν μεγαλύτερα έσοδα σε σύγκριση με το κόστος που προκαλούσε η προηγούμενη υποδομή τους.

Με τη χρήση του Σύννεφου το κυριότερο πλεονέκτημα είναι η ταχύτητα και η απόδοση της εφαρμογής με ίδια αποτελέσματα σε όλους τους χρήστες ανεξαρτήτου τοποθεσίας και ανεξαρτήτου συσκευής χρήσης.

Οι ανάγκες που δύναται να καλυφθούν σχετίζονται τόσο ως προς το λογισμικό όσο και ως προς το υλικό που οι λύσεις είναι ελαστικές και απεριόριστες καθώς η ίδια υποδομή μπορεί να αναπτυχθεί από έναν ως ένα πάρα πολύ μεγάλο αριθμό εξυπηρετητών .

Επίσης λόγω αυτής της μεγάλης υποδομής έχουν μειωθεί τα προβλήματα ασφάλειας των δεδομένων και χώρου καθώς μπορούν να διατηρούνται αντίγραφα των βάσεων δεδομένων

σε όλο το πλανήτη χωρίς να είναι ανάγκη να βρίσκονται στην ίδια τοποθεσία ή σε δύο μόνο διαφορετικούς εξυπηρετητές.

Ο συγχρονισμός των δεδομένων λόγω των μεγάλων ταχυτήτων δικτύου της υποδομής γίνονται σε πολύ σύντομο χρονικό διάστημα γεγονός που αυξάνει το λειτουργικό και αποδοτικό χρόνο.

Τα ποσοστά ασφάλειας και λειτουργίας των εφαρμογών έχουν αυξηθεί πάρα πολύ και σε μεγάλες περιπτώσεις φτάνουν το 99.9% ασφάλειας απώλειας των δεδομένων ή πτώσης της υποδομής του συστήματος.

Ωστόσο, λόγω της τόσο μεγάλης υποδομής αυξάνονται τα προβλήματα διαχείρισης του υλικού και λογισμικού και είναι αναγκαία η χρήση επιπρόσθετων λογισμικών για την διαχείριση του υλικού και του λογισμικού.

Επίσης πολλές εταιρίες δεν ήταν προετοιμασμένες ώστε να μεταφέρουν την υποδομή τους σε περιβάλλον Σύννεφου με αποτέλεσμα να έχουν αρνητικά αποτελέσματα λόγω της γρήγορης μετάπτωσης τους από την προηγούμενη υποδομή στην νέα.

2.6.3 Υποδομή ΕΔΕΤ

Το Εθνικό Δίκτυο Έρευνας και Τεχνολογίας (ΕΔΕΤ) έχει αναπτύξει μία Κεντρική Υπηρεσία το επονομαζόμενο okeanos όπου ανήκει στην Υποδομή ως Υπηρεσία (Infrastructure As a Service). **[11]**

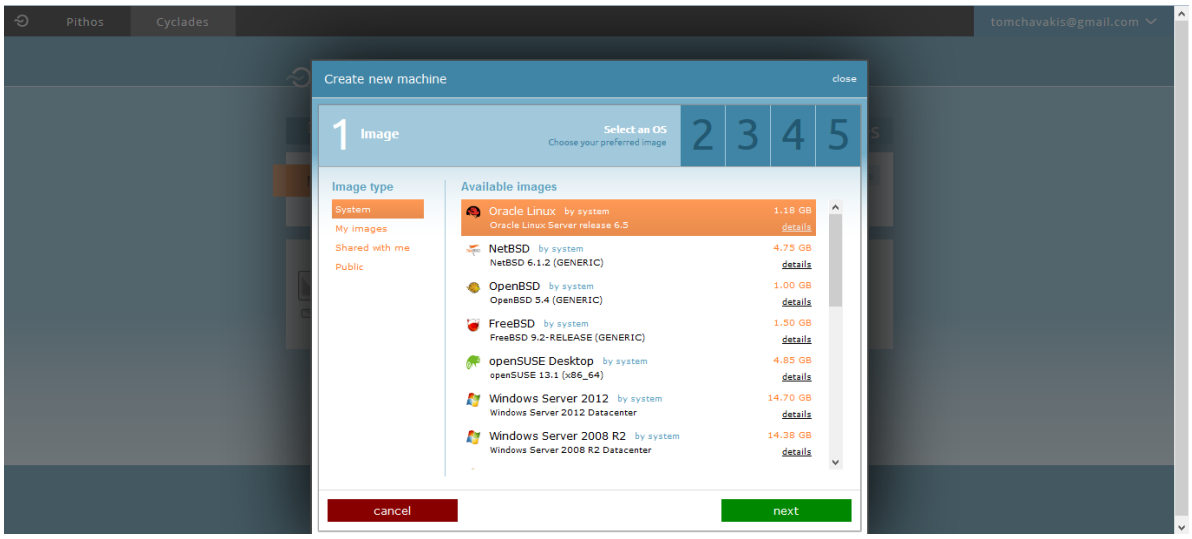
Ο okeanos ανήκει στο δημόσιο Σύννεφο (Public Cloud). Αναπτύχθηκε για να υποστηρίξει την ακαδημαϊκή και ερευνητική κοινότητα, προκειμένου να ελαχιστοποιήσει το κόστος που χρειαζόταν από τα Πανεπιστήμια και τα Ιδρύματα για να υποστηρίξουν τις δυνατότητες όσον αφορά το Υλικό (Hardware) και το Λογισμικό (Software), γεγονός που επιλύθηκε μέσω μίας κεντρικής Πλατφόρμας που καλύπτει τις περισσότερες ανάγκες των Φοιτητών και των Ακαδημαϊκών για έρευνα και ανάπτυξη συστημάτων Πληροφορικής.

Ο okeanos περιλαμβάνει 2 υπηρεσίες τις Cyclades, που είναι IAAS (Infrastructure As A Service), και τον pithos+, που είναι SAAS (Software As A Service) αντίστοιχα. Οι δύο αυτές υπηρεσίες ανήκουν σε οριζόντιες ηλεκτρονικές υποδομές (e-Infrastructures) υψηλής αξιοπιστίας, τις οποίες παρέχει σε όλα τα μέλη της εκπαιδευτικής και ερευνητικής κοινότητας, με προηγμένα και περιβαλλοντικά φιλικά υπολογιστικά κέντρα (green data centers). **[12]**

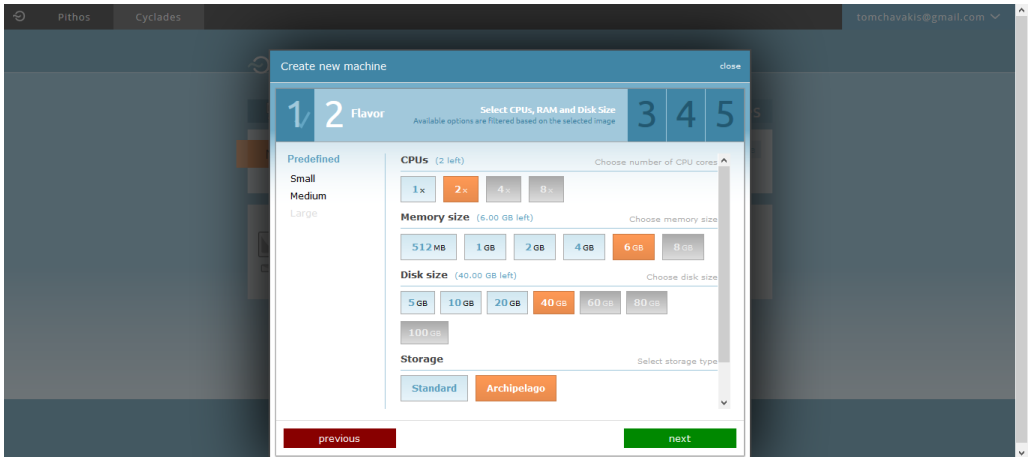
Οι Cyclades ανήκουν στην κατηγορία της Υποδομής ως Υπηρεσία (IAAS Infrastructure As A Service), παρέχοντας εικονικές μηχανές (VMs) με διάφορα λειτουργικά συστήματα (OS Images) τα οποία είναι προσαρμοσμένα στο υλικό της υποδομής.

Μέσω της υπηρεσίας αυτής δίνεται η δυνατότητα εγκατάστασης όλων των διανομών λειτουργικών συστημάτων είτε Ανοιχτού είτε Λογισμικού Εμπορίου σε πολύ σύντομο χρονικό διάστημα προσφέροντας υψηλές δυνατότητες οι οποίες καλύπτουν τις περισσότερες από τις ανάγκες των χρηστών.

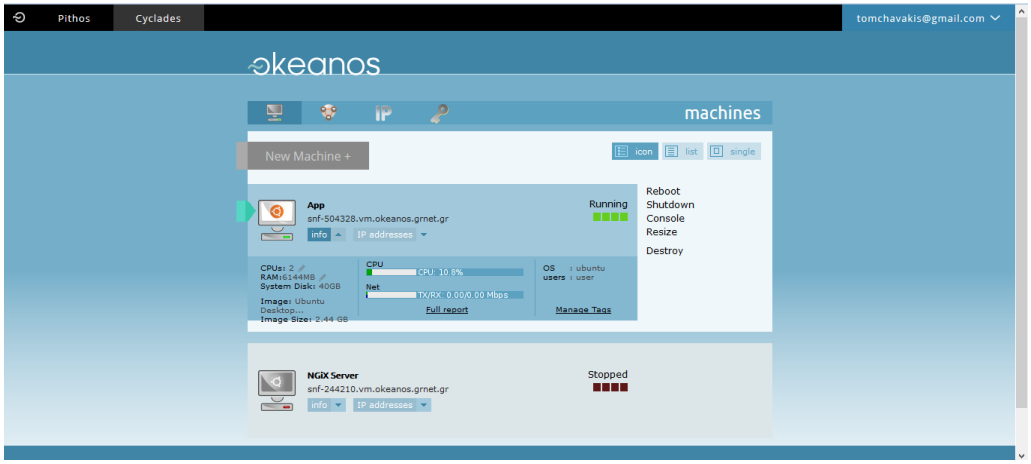
Η υπηρεσία ύστερα από την επιτυχή εκκίνηση και δοκιμή της πρώτης έκδοσης τώρα βρίσκεται στη δεύτερη έκδοσή της και στη συνέχεια θα μεταβεί σε beta έκδοση.



Εικόνα 2.4 *Δημιουργία Εικονικής Μηχανής σε Περιβάλλον Cyclades*



Εικόνα 2.5 *Σύνθεση Χαρακτηριστικών Εικονικής Μηχανής*



Εικόνα 2.6 *Περιβάλλον Διαχείρισης Εικονικών Μηχανών σε Υποδομή Cyclades*

Όπως φαίνεται και από την εικόνα 2.4 κατά τη δημιουργία μίας εικονικής μηχανής (Virtual Machine) μπορούμε να επιλέξουμε από μία πληθώρα Λειτουργικών Συστημάτων (OS), διανομές Ανοιχτού Λογισμικού καθώς και Λογισμικό Εμπορίου.

Σε περίπτωση που διαθέτουμε το αρχείο μιας εικονικής μηχανής μπορούμε να δημιουργήσουμε μία εικονική μηχανή από μία ήδη εγκατεστημένη εικονική μηχανή επιλέγοντας το MyImages αφού πρωτίστως ανεβάσουμε την προ εγκατεστημένη εικονική μηχανή στο pithos + .

Στην επόμενη φάση αφού επιλέξουμε Λειτουργικό Σύστημα θα πρέπει να εισαγάγουμε τις ρυθμίσεις των χαρακτηριστικών της εικονικής μηχανής. Ουσιαστικά εισάγουμε τα μεγέθη που επιθυμούμε σε Μνήμη (RAM), τον αριθμό των Πυρήνων επεξεργαστικής ισχύος (CPU), και τη χωρητικότητα του Σκληρού Δίσκου που θέλουμε να έχει η εικονική μηχανή.

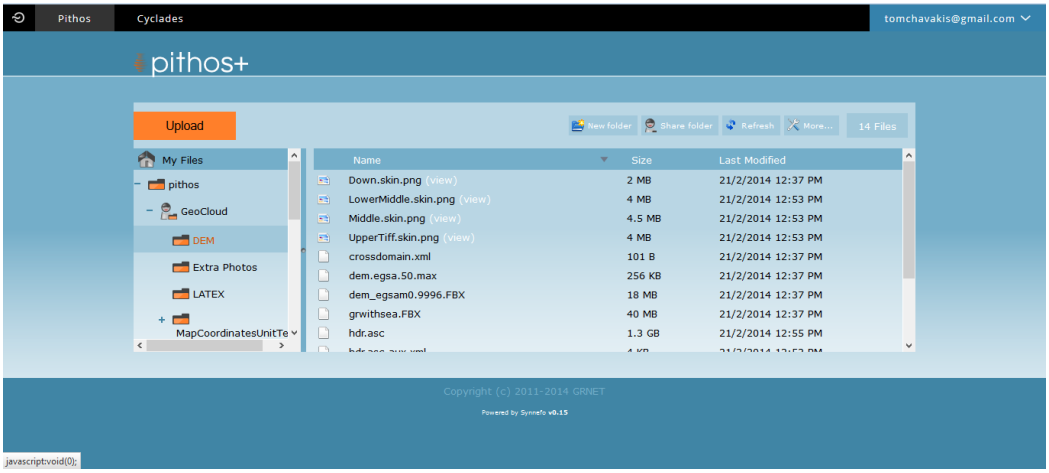
Στην τελική φάση όπως φαίνεται και στην Εικόνα 2.6, μας δίνεται η δυνατότητα να διαχειριστούμε την εικονική αυτή μηχανή ως προς την κατάστασή της, και μπορούμε να επιβλέπουμε και να ελέγχουμε το φορτίο της σε μνήμη και σε επεξεργαστική ισχύ. Επίσης μπορούμε να αλλάξουμε τις ρυθμίσεις σε υλικό (Hardware) της Εικονικής Μηχανής επιλέγοντας να κάνουμε Resize την μηχανή.

Τέλος μας δίνεται η δυνατότητα να αποδεσμεύσουμε την διεύθυνση δικτύου από μία συγκεκριμένη εικονική μηχανή και να την δεσμεύσουμε σε μία άλλη καθώς στην τελευταία έκδοση του okeanos έχουμε μία μόνο public IP.

Επίσης μέσω της επιλογής του Δικτύου μπορούμε να δημιουργήσουμε εικονικά δίκτυα τα οποία ενώνουν τις εικονικές μηχανές.

Η άλλη υπηρεσία του okeanos περιλαμβάνει τον pithos + . Ο pithos + αποτελεί εξέλιξη του Pithos που ήταν στην προηγούμενη έκδοση του συστήματος. Το σύστημα αυτό ανήκει στην κατηγορία του SAAS (Software As A Service).

Με αυτή την υπηρεσία προσφέρεται στους φοιτητές και στην ακαδημαϊκή κοινότητα η δυνατότητα να ανεβάζουν δεδομένα καθώς διατίθεται χωρητικότητα ώς 50 GB αποθηκευτικού χώρου για τα αρχεία στο Σύννεφο (Cloud).



Εικόνα 2.7 *Σελίδα του Pithos + από το okeanos.grnet.gr*

Όπως φαίνεται και στην εικόνα 2.7 μέσω του pithos+ μπορούμε να διαχειριστούμε τα αρχεία που βρίσκονται στον λογαριασμό μας , μπορούμε να ανεβάσουμε (upload) νέα αρχεία καθώς και να δημιουργήσουμε κοινόχρηστους φακέλους.

ΚΕΦΑΛΑΙΟ 3 ΕΡΓΑΛΕΙΑ ΚΑΙ ΜΕΘΟΔΟΙ

3.1 Δημιουργία Τρισδιάστατου Γεωγραφικού Ανάγλυφου

Στην ενότητα αυτή θα αναλυθεί η διαδικασία δημιουργίας του Γεωγραφικού Ανάγλυφου.Η διαδικασία αυτή περιλαμβάνει μία ακολουθία από συγκεκριμένα βήματα τα οποία ολοκληρώνονται με την απεικόνιση του Γεωγραφικού Μοντέλου της Ελλάδας σε Τρεις Διαστάσεις. Η μέθοδος αυτή βασίζεται στην κατασκευή εικόνων Χαρτών Ύψους (height map) από πρωτογενή και ανεπεξέργαστα δεδομένα, και στην επεξεργασία τους σε περιβάλλοντα Τριών Διαστάσεων.

3.1.1 Επεξεργασία Πρωτογενών Δεδομένων μέσω του λογισμικού QGIS

Για την επεξεργασία των Γεωγραφικών δεδομένων χρησιμοποιήθηκε το εργαλείο Ανοιχτού Λογισμικού QGIS (QuantumGIS)**[13]**, το οποίο μας δίνει τη δυνατότητα να δημιουργήσουμε, αναλύσουμε και να επεξεργαστούμε χωρικά δεδομένα.

Το QGIS διαθέτει μία ποικιλία από λειτουργίες και δυνατότητες οι οποίες καλύπτουν ένα ευρύ φάσμα απαιτήσεων του τομέα της γεωπληροφορικής.

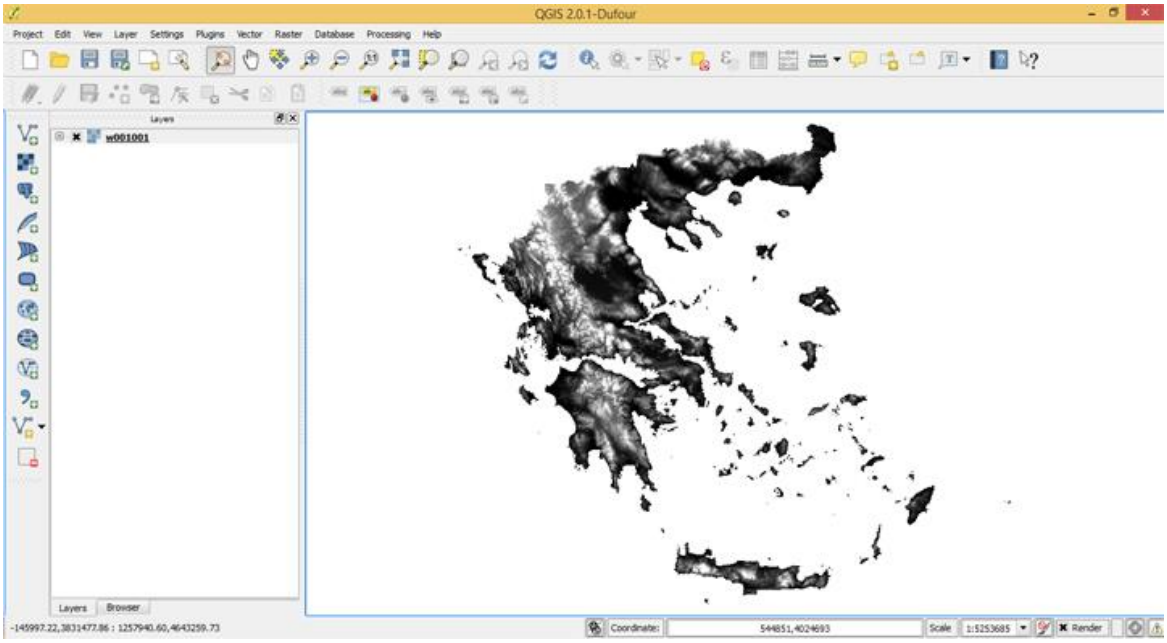
Τα πρωτογενή δεδομένα που έχουμε στη διάθεσή μας είναι ψηφιακά δεδομένα ύψους (DEM Digital Elevation Model) από το Ψηφιακό Μοντέλο Ανάγλυφου σε συγκεκριμένο γεωγραφικό χώρο της Ελληνικής επικράτειας υπό τη μορφή `.adf` .Πρόκειται για έναν από τους δύο τύπους Arc/InfoBinaryGrid αρχείων που αναπτύχθηκαν από την ESRI. Το άλλο είναι το Arc/InfoASCIIGrid.

Όνομα	Περιγραφή
`dblbnd.adf`	Τα όρια του grid
`hdr.adf`	Πληροφορίες επικεφαλίδας με τον αριθμό των tiles και το μέγεθός τους.
`sta.adf`	Στατιστικές πληροφορίες όπως το μέγιστο, το ελάχιστο και το μέσο όρο
`vat.adf`	Πίνακας τιμών των χαρακτηριστικών
`Prj.adf`	Γεωαναφορά συνδυασμένο με τις παραμέτρους αυτού του αρχείου.
`W001001.adf`	Πραγματικά raster δεδομένα

`w001001x.adf`	Ευρετήριο με δείκτες για κάθε ένα από τα tiles του αρχείου w001001.adf

Πίνακας 3.1 *Περιγραφή Αρχείων* `.adf`

Μετά από την εισαγωγή των δεδομένων αυτών στο πρόγραμμα QGIS έχουμε την ακόλουθη εικόνα :

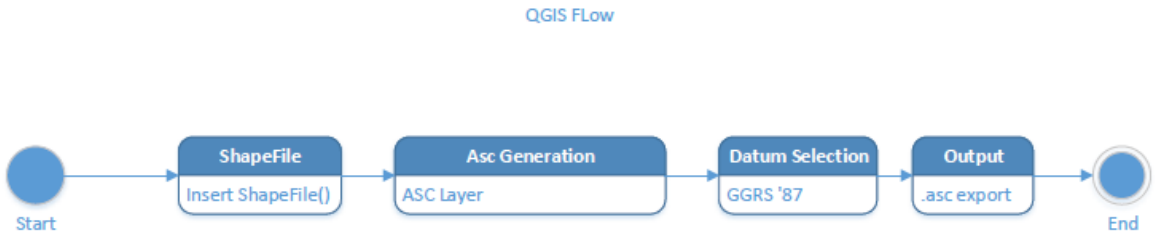


***Εικόνα* 3.1** Επεξεργασία .adf Δεδομένων μέσω QGIS

Το DEM (Digital Elevation Model) έχει γεωαναφερθεί στο προβολικό σύστημα ΕΓΣΑ ’87(GGRS 87).

Στην συνέχεια εξάγουμε σε δεδομένα raster τα αρχεία `.adf` σε αρχείο της μορφής `.asc` όπως περιγράφτηκε στο Κεφάλαιο 2 ενότητα 2.3 .

Η διαδικασία περιγράφεται από το ακόλουθο διάγραμμα καταστάσεων της εικόνας 3.2.



Εικόνα 3.2 *Διαδικασία μετατροπής ShapeFile σε Raster της μορφής* `.asc`

3.1.2 Παραγωγή Χάρτη Ύψους (Height Map) μέσω `.asc` Δεδομένων

Από την προηγούμενη διαδικασία έχουμε εξάγει δεδομένα Raster της μορφής `.asc` μέσω του προγράμματος QGIS.

Τα δεδομένα αυτά έχουν πληροφορίες ύψους και διαθέτουν μία ελάχιστη και μία μέγιστη τιμή.

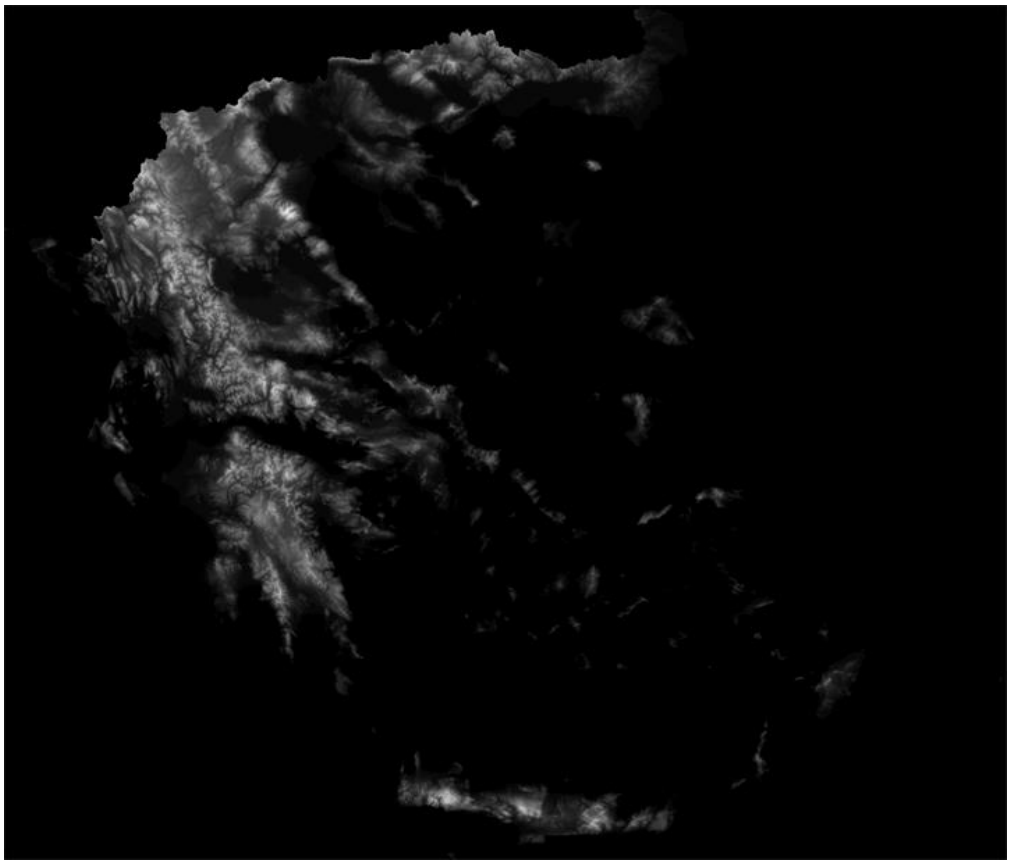
Σύμφωνα με το παραγόμενο αρχείο η ελάχιστη τιμή αντιστοιχεί στην τιμή -9999 που είναι η τιμή όπου δεν έχουμε δεδομένα, για παράδειγμα η θάλασσα που έχει μηδενικό υψόμετρο, και η μέγιστη τιμή στο υψηλότερο υψόμετρο του χώρου που μελετάται με τιμή 2.918 m.

Η τιμή όπου δεν υπάρχουν δεδομένα είναι πολύ σημαντική, καθώς όσα τμήματα δεν διαθέτουν τιμή θα πρέπει να αποκλειστούν από το τρισδιάστατο γεωγραφικό ανάγλυφο καθώς δεν έχουν λόγο ύπαρξης στον τρισδιάστατο υψομετρικό Χάρτη.

Για την δημιουργία του τρισδιάστατου γεωγραφικού ανάγλυφου είναι απαραίτητο να έχουμε μία εικόνα height map απόχρωσης και κλίμακας του γκρι.

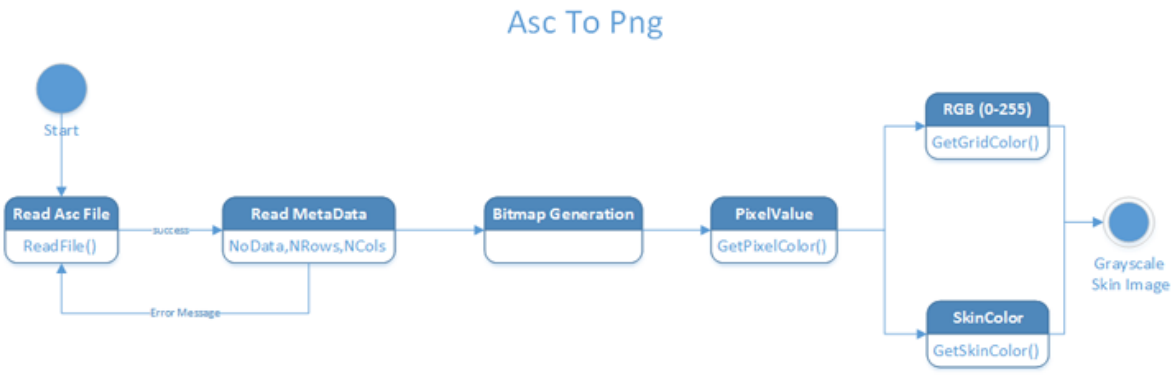
Για αυτό το λόγο δημιουργήθηκε ένα πρόγραμμα το οποίο διαβάζει τα δεδομένα ύψους από το αρχείο `.asc` και μετατρέπει τα δεδομένα αυτά σε χρώμα διαβάθμισης του γκρι βασισμένο στο πρότυπο ARGB (alpha,red,green και blue).

Με αυτή τη μέθοδο κατασκευάζεται η εικόνα 3.3 .



Εικόνα 3.3 *Εικόνα Απόχρωσης του Γκρι Ανάλογη της Υψομετρικής Κλίμακας του Χώρου Μελέτης*

Η μέθοδος και τα βήματα δημιουργίας της εικόνας αναλύονται από το ακόλουθο διάγραμμα καταστάσεων της εικόνας 3.4 .



Εικόνα 3.4 *Διαδικασία Δημιουργίας Εικόνας Ύψους*

Για να υλοποιηθεί η διαδικασία αυτή έχει δημιουργηθεί ένα πρόγραμμα το οποίο κατασκευάζει την εικόνα αυτή διαβάζοντας τα δεδομένα από το `.asc` αρχείο. Αρχικά διαβάζει τον αριθμό των σειρών και τον αριθμό των στηλών του πίνακα του αρχείου `.asc` μέσω των μετά - πληροφοριών που περιέχονται στις πρώτες γραμμές του αρχείου αυτού.

Επίσης αποθηκεύεται η τιμή στην οποία δεν υπάρχουν δεδομένα NODATA η οποία είναι η τιμή που αντιστοιχεί στο απόλυτο μαύρο της παραγόμενης εικόνας.

Στην συνέχεια κατασκευάζεται η εικόνα

```
Bitmap gridImage = new Bitmap(nCols, nRows,
PixelFormat.Format16bppArgb1555);
```

τόσων στηλών και γραμμών όσων περιγράφονται στο αρχείο αυτό.

```
gridImage.SetPixel(col, row, GetGridColor(noData, pixelValue));
```

Η συνάρτηση GetGridColor δέχεται σαν παραμέτρους την τιμή όπου δεν υπάρχουν δεδομένα και την τιμή που αναγράφεται στο αρχείο.

```
private static Color GetGridColor(double noData, double pixelValue)
        {
            double imageValue = pixelValue == noData ? 0 : ((255.0 -
Displacement) * (pixelValue < 0 ? 0 : pixelValue) / MaxHeight) +
Displacement;
            byte colorValue = (byte)Math.Round(imageValue);
            return Color.FromArgb(255, colorValue, colorValue,
colorValue);
}
```

Ύστερα από τον υπολογισμό της τιμής του χρώματος που θα πρέπει να εισαχθεί στο εκάστοτε pixel δημιουργείται το σύνολο της ασπρόμαυρης φωτογραφίας η οποία περιλαμβάνει τιμές από 0-255.

3.1.3 Παραγωγή Έγχρωμης εικόνας μέσω Δεδομένων `.asc`

Κατά τον ίδιο τρόπο όπως περιγράφτηκε στην ενότητα 3.1.2 κατασκευάζουμε και την έγχρωμη εικόνα. Η εικόνα αυτή θα επικαλύψει το τρισδιάστατο ανάγλυφο και θα δώσει μία αίσθηση ύψους ανάλογη με την αίσθηση που υπάρχει στον ανθρώπινο εγκέφαλο.

Η αίσθηση αυτή αντιστοιχεί σε μία διαβάθμιση του χρώματος πράσινου και του καφέ. Το πράσινο χρώμα αντιστοιχεί στις πεδιάδες και σε χαμηλά υψόμετρα και όσο ανεβαίνουμε σε ύψος το πράσινο χρώμα γίνεται πιο σκούρο. Αντίστοιχη λειτουργία συμβαίνει και με το χρώμα καφέ το οποίο όσο ανεβαίνουμε σε μεγαλύτερο υψόμετρο γίνεται πιο σκούρο.

Η διαδικασία αυτή περιγράφεται από την ακόλουθη συνάρτηση *GetSkinColor* η οποία δέχεται σαν παραμέτρους την τιμή όπου δεν υπάρχουν δεδομένα και την τιμή του ύψους που υπάρχει στο αρχείο `.asc`.

```
private static Color GetSkinColor(double noData, double pixelValue)
        {
            if (pixelValue == noData)
                return Color.FromArgb(255, 60, 80, 100);

            if (pixelValue <= 0)
                return Color.FromArgb(255, 139, 146, 112);
            if (pixelValue <= 250)
                return Color.FromArgb(255, 158, 159, 117);
            if (pixelValue <= 500)
                return Color.FromArgb(255, 177, 173, 123);
            if (pixelValue <= 750)
                return Color.FromArgb(255, 196, 186, 129);
            if (pixelValue <= 1000)
                return Color.FromArgb(255, 215, 200, 135);
if (pixelValue <= 1250)
                return Color.FromArgb(255, 208, 190, 128);
            if (pixelValue <= 1500)
return Color.FromArgb(255, 202, 180, 121);
            if (pixelValue <= 1750)
```

```
                return Color.FromArgb(255, 195, 170, 114);
            if (pixelValue <= 2000)
                return Color.FromArgb(255, 189, 160, 107);
            if (pixelValue <= 2250)
                return Color.FromArgb(255, 183, 150, 101);
            if (pixelValue <= 2500)
                return Color.FromArgb(255, 179, 154, 113);
if (pixelValue <= 2750)
                return Color.FromArgb(255, 175, 158, 126);
            if (pixelValue <= 3000)
                return Color.FromArgb(255, 171, 162, 138);
            if (pixelValue <= 3250)
                return Color.FromArgb(255, 167, 167, 151);

            returnColor.FromArgb(255, 192, 192, 192);
}
```

Το αποτέλεσμα αυτής της διαδικασίας είναι η εικόνα 3.5 . Αν παρατηρήσουμε την εικόνα αυτή θα δούμε ότι σε περιοχές με μεγάλο υψόμετρο, όπως για παράδειγμα ο Όλυμπος , που είναι και το υψηλότερο σημείο της Ελλάδας θα δούμε ότι σε αυτό το σημείο και στα γειτονικά του, το χρώμα που κυριαρχεί είναι το σκούρο καφέ ενώ σε αντίθεση σε περιοχές που κυριαρχεί χαμηλό υψόμετρο, για παράδειγμα η περιοχή του Κυπαρισσιακού κόλπου στην Δυτική Πελοπόννησο, κυριαρχεί το πράσινο χρώμα.

Εικόνα 3.5 *Έγχρωμη Εικόνα Ελληνικής Επικράτειας*

3.1.4 Δημιουργία Χάρτη Τριών Διαστάσεων με Χρήση του 3DStudioMax

Για την δημιουργία του Γεωγραφικού Τρισδιάστατου ανάγλυφου χρησιμοποιήθηκε το πρόγραμμα 3DStudioMax, με εμπορική άδεια που αξιοποιείται σε εταιρικό περιβάλλον.

Αρχικά δημιουργούμε ένα ορθογώνιο εισάγοντας την μέγιστη απόσταση σε μέτρα του μήκους και του πλάτους της περιοχής. Δημιουργούμε μία περιοχή όπου επιθυμούμε να μετατρέψουμε σε αντικείμενο τριών διαστάσεων σύμφωνα με το γεωγραφικό πρότυπο που έχουμε επιλέξει να εφαρμόσουμε.

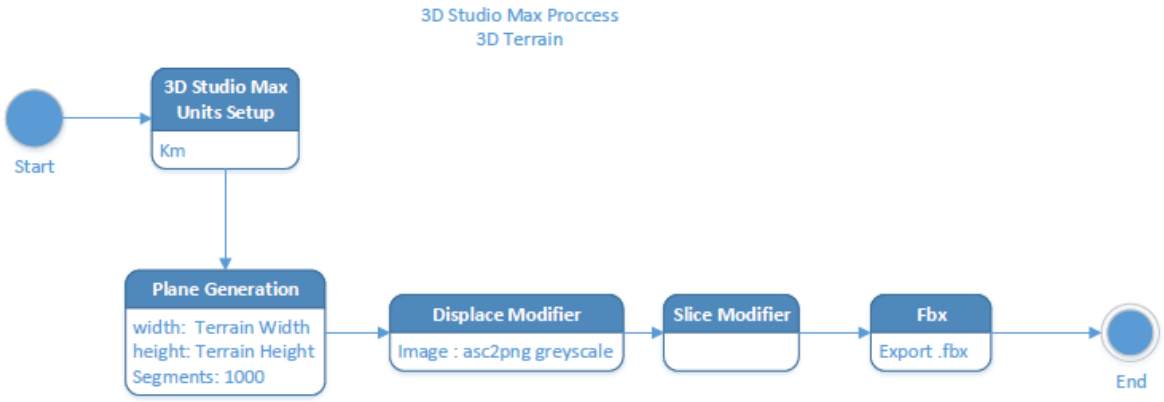
Έπειτα προσθέτουμε ένα displace modifier στην επιλεγμένη περιοχή και εισάγουμε την εικόνα height map που έχει δημιουργηθεί μέσω της μετατροπής του αρχείου `.asc` σε `grayscale.png` καθώς και σαν material την έγχρωμη εικόνα που δημιουργήθηκε μέσω της μετατροπής του αρχείου `.asc` σε `.png`.

Με αυτό τον τρόπο λαμβάνουμε λαμβάνουμε το αποτέλεσμα της εικόνας 3.6 .

Εικόνα 3.6 *Τρισδιάστατο Ανάγλυφο του Ελλαδικού Χώρου*

Τέλος εξάγουμε το αρχείο σε τέτοια μορφή που είναι επιθυμητή από το Unity3D δηλαδή σε αρχείο της μορφής `.fbx`

Η συνολική διαδικασία φαίνεται στο διάγραμμα της εικόνας 3.7 .



Εικόνα 3.7 *Διαδικασία Δημιουργίας Γεωγραφικού Ανάγλυφου Τριών Διαστάσεων*

3.1.5 Ευθυγράμμιση Χάρτη Τριών Διαστάσεων σε Περιβάλλον 3D

Ύστερα από την δημιουργία του Χάρτη Τριών Διαστάσεων υπό τη μορφή `.fbx`, τύπος που δέχεται η μηχανή Τριών Διαστάσεων για την εισαγωγή των μοντέλων, θα πρέπει να τεθεί στις σωστές συντεταγμένες προκειμένου να έχουμε ένα κοινό γεωγραφικό Υπόβαθρο.

Για την δημιουργία της Τρισδιάστατης Εφαρμογής θα χρησιμοποιηθεί το πρόγραμμα Unity3D στην ελεύθερη έκδοση του.

Σύμφωνα με τα προαναφερθέντα καταλαβαίνουμε ότι για να δημιουργήσουμε το Γεωγραφικό Υπόβαθρο Τριών Διαστάσεων με Γεωαναφορά στο προβολικό σύστημα ΕΓΣΑ’87 (Κεφάλαιο 2 Ενότητα 2.3) θα πρέπει να θέσουμε ένα σημείο αναφοράς το οποίο ισοδυναμεί με το Κέντρο του Διονύσου που είναι και το κέντρο αναφοράς αυτού του προτύπου.

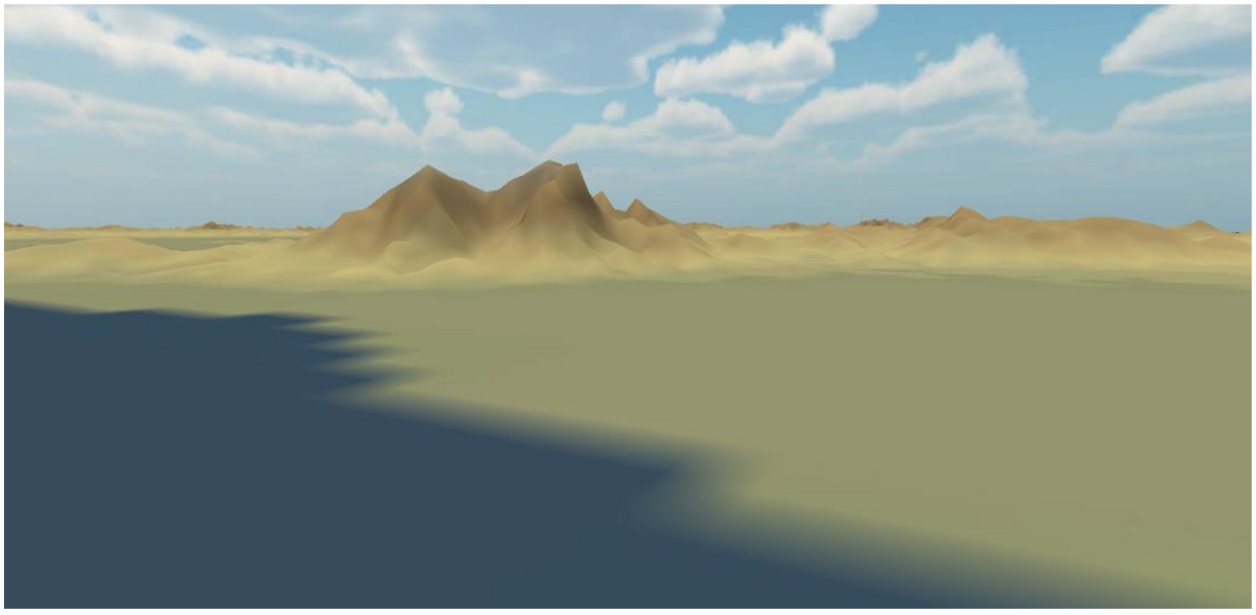
Ένα σημαντικό σημείο είναι ότι η μονάδα μέτρησης του ΕΓΣΑ ’87 όπως και του Unity3D είναι τα μέτρα, επομένως έχουμε την ίδια μονάδα μέτρησης και στα δύο συστήματα.

Έπειτα προσθέτουμε ένα αντικείμενο στο κέντρο των αξόνων της Τρισδιάστατης Μηχανής σαν σημείο αναφοράς, βάσει του οποίου γίνονται οι μετατροπές και ο υπολογισμός των συντεταγμένων.

Το κέντρο του Διονύσου αντιστοιχεί στις συντεταγμένες 500000,4200000 οπότε όλες οι συντεταγμένες υπολογίζονται βάσει αυτών των συντεταγμένων. Μεταφέρουμε το μοντέλο των τριών διαστάσεων στο σημείο 0,0,0 του Unity3D στο σημείο του Διονύσου.

Εικόνα 3.8 *Κάτοψη Χάρτη , Σημείο 0,0,0 Σε Περιοχή του Πλανητικού Δορυφορικού Σταθμού του Διονύσου*

Παράλληλα έχει κατασκευαστεί ένα πρόγραμμα δοκιμών όπου ελέγχουμε πού αντιστοιχούν οι συγκεκριμένες συντεταγμένες και ανάλογα εκτελούνται λεπτές διορθώσεις προκειμένου να υπάρχει λεπτομέρεια στις συντεταγμένες.



Εικόνα 3.9 *Όλυμπος , Το υψηλότερο βουνό στην Ελλάδα με μέγιστο υψόμετρο 2.918μέτρα*

3.2 Δημιουργία Web-Service Παροχής Υδρογεωλογικών Δεδομένων

Μία Διαδικτυακή Υπηρεσία (Web Service) είναι ένα λογισμικό επικοινωνίας μεταξύ δύο συσκευών μέσω δικτύου.

Οι υπηρεσίες Παροχής Δεδομένων είναι πολύ χρήσιμες επειδή μπορούν να έχουν αλληλεπίδραση με τη Βάση Δεδομένων και να επιστρέφουν κάθε φορά δεδομένα έπειτα από ένα αίτημα υπό την μορφή JSON ή της μορφής XML.

Στην συγκεκριμένη εφαρμογή λόγω αδυναμίας άμεσης σύνδεσης στα δεδομένα στη Βάση Δεδομένων μέσω της μηχανής τρισδιάστατων γραφικών, η μόνη δυνατότητα που παρέχεται από το Unity3D είναι η αποστολή ενός διαδικτυακού αιτήματος (Web-Request) όπου καλείται με την αναφορά `www` **[14]**.

Το `www` είναι ένα άρθρωμα (module) με το οποίο καλείται μία διεύθυνση διαδικτύου όπου συνήθως πρόκειται για ένα Web Service προκειμένου να ληφθούν δεδομένα από αυτή τη διεύθυνση.

Το Web Service υλοποιήθηκε με τεχνολογία PHP – MySQL. Στην Βάση Δεδομένων MySql έχουν εισαχθεί τα δεδομένα των γεωτρήσεων και των φίλτρων σε δύο πίνακες και η Βάση έχει εγκατασταθεί σε έναν Εξυπηρετητή Βάσης Δεδομένων (Database Server), ο οποίος είναι εγκατεστημένος στην υποδομή Okeanos του ΕΔΕΤ.

Σε κάθε εξυπηρετητή έχει εγκατασταθεί ένα Web Service το οποίο επικοινωνεί με τη μηχανή τρισδιάστατων γραφικών προκειμένου να κατασκευαστεί ο υδροφόρος ορίζοντας.

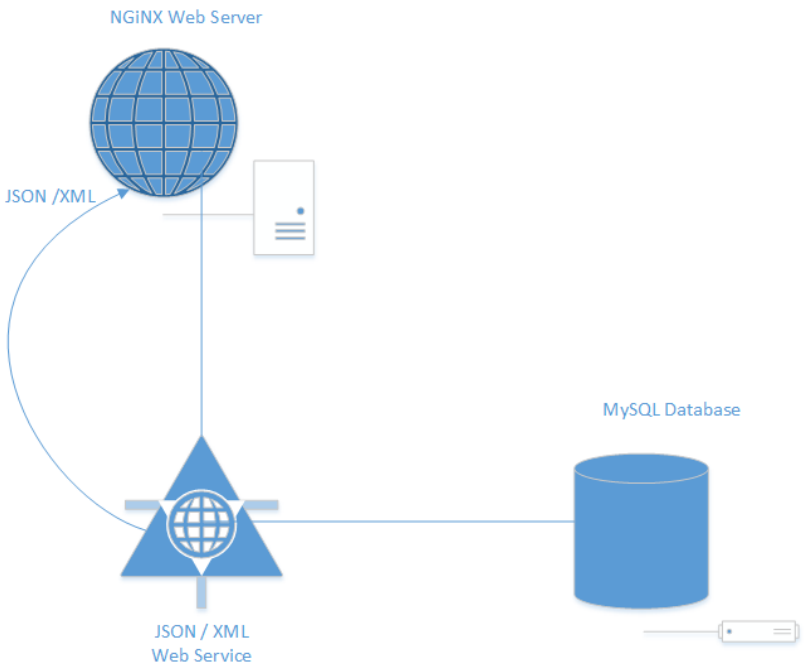
Ο κώδικας που χρησιμοποιείται για την επιστροφή των δεδομένων είναι ο ακόλουθος:

```
<?php
     $link =
     mysql_connect("83.212.108.251:3306","tomchavakis","******") or
     die('Cannot connect to the DB');
     mysql_select_db("Wells",$link) or die('Cannot select the DB');
  mysql_query("SET NAMES UTF8;");
     $query = "SELECT * FROM Well";
```

```
	$result = mysql_query($query,$link) or die('Errant query:
'.$query);
	$posts = array(); //new Array
	if(mysql_num_rows($result)) {
		while($post = mysql_fetch_assoc($result)) {
		$posts[] = $post; //Insert results from query to array
		}
	}
	  $format='json';
		if($format == 'json') {
		header('Content-type: application/json');
		echo json_encode($posts);  //encoding results to JSON
	}
	else {  // Ability to Change Returning type to XML
		header('Content-type: text/xml');
		echo '<posts>';
		foreach($posts as $index => $post) {
			if(is_array($post)) {
				foreach($post as $key => $value) {
					echo '<',$key,'>';
					if(is_array($value)) {
						foreach($value as $tag => $val) {
							echo
'<',$tag,'>',htmlentities($val),'</',$tag,'>';
						}
					}
					echo '</',$key,'>';
				}
			}
		}
		echo '</posts>';
@mysql_close($link);
}
?>
```

Ο ανωτέρω κώδικας δίνει τη δυνατότητα να επιστρέψουμε τα δεδομένα της βάσης σε αντικείμενο JSON ή σε μορφή XML.

Η διασύνδεση των συστημάτων φαίνεται στο διάγραμμα της εικόνας 3.10 .



Εικόνα 3.10 *Web Service - Διασύνδεση Συστημάτων*

Όπως φαίνεται στο διάγραμμα, σε έναν εξυπηρετητή διαδικτύου NGiNX έχει εγκατασταθεί η υπηρεσία Διαδικτύου (Web Service) η οποία διασυνδέεται με την Βάση Δεδομένων MySQL . Το Web Service στέλνει ερωτήματα στη βάση και αυτή επιστρέφει τα δεδομένα σε μορφή JSON/ XML στον Web Server και από εκεί αυτά διαχειρίζονται από την Μηχανή των Τριών Διαστάσεων.

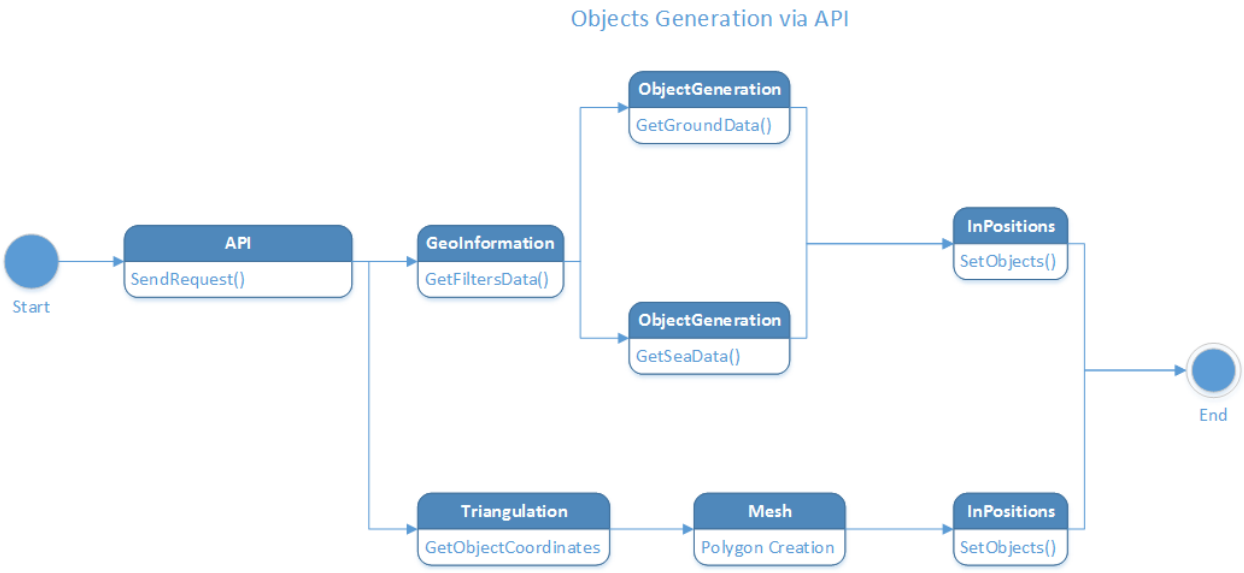
3.3 Δημιουργία Υδροφόρου Ορίζοντα Τριών Διαστάσεων

Οι διαδικασίες που λαμβάνουν μέρος σε αυτή τη λειτουργία είναι οι ακόλουθες :

- Επικοινωνία με τη Βάση Δεδομένων μέσω της Υπηρεσίας Διαδικτύου (Web Service)
- Αλγόριθμος Ταξινόμησης Γεωτρήσεων (*βλ.* Ενότητα 2.4.1)

- Αλγόριθμος Τριγωνοποίησης Ear Clipping (*βλ.* Ενότητα 2.4.2)
- Δημιουργία Αντικειμένων Τριών Διαστάσεων αποτέλεσμα της Ένωσης Γεωτρήσεων
- Δημιουργία Υποβάθρου Βάσης

Η διαδικασία περιγράφεται σχηματικά από το ακόλουθο διάγραμμα καταστάσεων της εικόνας 3.11 .



Εικόνα 3.11 *Διάγραμμα Καταστάσεων Δημιουργίας Υδροφόρου Ορίζοντα*

Όπως φαίνεται και στο διάγραμμα, αρχικά μέσω ενός Web Service που περιγράφτηκε στην ενότητα 3.2, τα δεδομένα διαχωρίζονται σε δύο βασικές κατηγορίες. Η μία κατηγορία σχετίζεται με την τριγωνοποίηση και την δημιουργία του Υδροφόρου Ορίζοντα και η άλλη με τα φίλτρα και την δημιουργία των Γεωτρήσεων σαν Αντικείμενα των Τριών Διαστάσεων, καθώς και με τα μορφολογικά χαρακτηριστικά της συγκεκριμένης γεώτρησης (νερό ή γεωλογικοί σχηματισμοί).

Κατά την διαδικασία την Δημιουργίας των Γεωτρήσεων μέσω της συνάρτησης `GetFiltersData()` διαχωρίζουμε τα δεδομένα μας σε δεδομένα Πετρωμάτων `GetGroundData()` και δεδομένα Ύδατος `GetSeaData()` και στη συνέχεια τα θέτουμε στις θέσεις τους σε ΕΓΣΑ ’87 γεωγραφικό σύστημα.

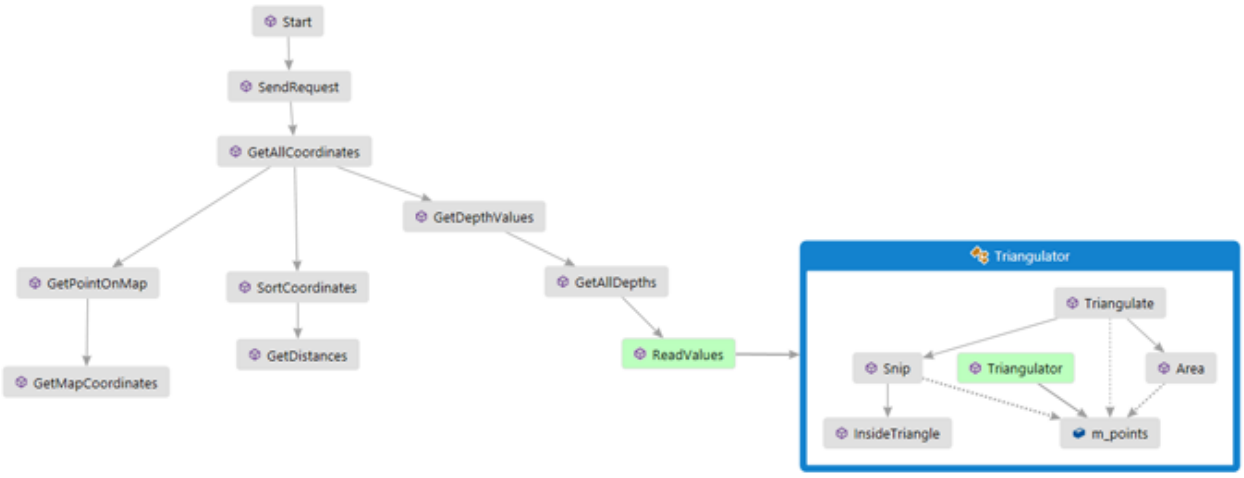
Στην συνέχεια πρέπει να κατασκευαστεί ο Υδροφόρος Ορίζοντας, ο οποίος σχηματίζεται μέσω της σύνδεσης των Γεωτρήσεων και των φίλτρων που υπάρχουν στην εκάστοτε γεώτρηση.

Για αυτό το λόγο με τη συνάρτηση `GetObjectCoordinates`() διαχωρίζουμε τις συντεταγμένες και εκτελούμε τον αλγόριθμο ταξινόμησης των Γεωτρήσεων και στην συνέχεια μέσω του `PolygonCreation` εκτελούμε τον αλγόριθμο Τριγωνοποίησης Ear Clipping ο οποίος δημιουργεί το ενιαίο πλέγμα ένωσης μεταξύ τεσσάρων σημείων.

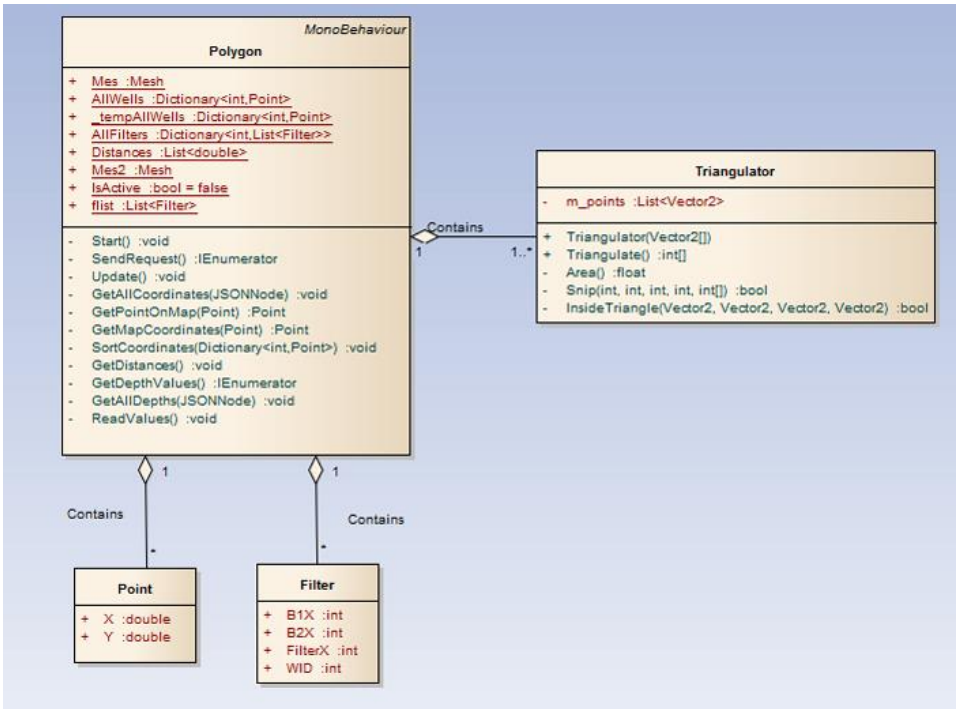
Για την δημιουργία του πλέγματος τριών διαστάσεων της ένωσης μεταξύ 2 γεωτρήσεων θα πρέπει να γνωρίζουμε 4 σημεία προκειμένου ο αλγόριθμος Ear Clipping να εκτελεστεί όπως περιγράφτηκε στην ενότητα 2.4.2 . Επειδή το σύστημα ΕΓΣΑ ’87 και το Unity3D έχει σαν μονάδα μέτρησης τα μέτρα αυτό σημαίνει ότι στο σύστημα έχουμε Εγκάρσια Μερκατορική Προβολή γεγονός που διευκολύνει τις πράξεις εύρεσης της απόστασης μεταξύ δύο σημείων.

Η συνάρτηση που δύναται να υπολογίσει την απόσταση μεταξύ των δύο γεωτρήσεων είναι η ακόλουθη :

$$d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2} \quad (1)$$



Εικόνα 3.12 *Διάγραμμα Ροής Διαδικασίας Τριγωνοποίησης*



Εικόνα 3.13 *Διάγραμμα Κλάσεων Τριγωνοποίησης*

Ο προτεινόμενος αλγόριθμος τριγωνοποίησης ο οποίος αξιοποιεί την μέθοδο Ear Clipping μεταξύ τεσσάρων σημείων $(X_1,Y_1),(X_2,Y_2)$ και $(X_1,Y_{1filteri})$, $(X_2,Y_{2\ filteri})$ είναι ο ακόλουθος :

Αλγόριθμος Τριγωνοποίησης :

input: set of Wells, Wells $\subseteq\{1,\dots,Well_i\}$at current iteration.
input: set of Filters, Filters $\subseteq\{1,\dots,Filter_i\}$at current iteration.
output: Mesh Connection after Triangulation at current iteration.
1:for $Well_i$in a set of Wells do
2: tempWells = sorting Wells by X // Sorted Wells by Coordinates.
3:end for
4:for $tempWell_k$in a set of tempWells do
5: Filters = Set of Filter k.
6: AllWells = Set of Wells with Filters .
7:end for
8:for $AllWell_i$in a set of AllWells do

9: Distance = Distance between two Points. // Find Distance by eq.1
10: Mesh = Object Generation from Triangulation 4 Vertices according
11: to Ear Clipping Algorithm.
12: Angle is the Relative Angle Between $AllWell_i$ and $AllWell_{i+1}$.
13: Connection is Mesh Connection $AllWell_i$(X,Y) with angle. // output
14:endfor

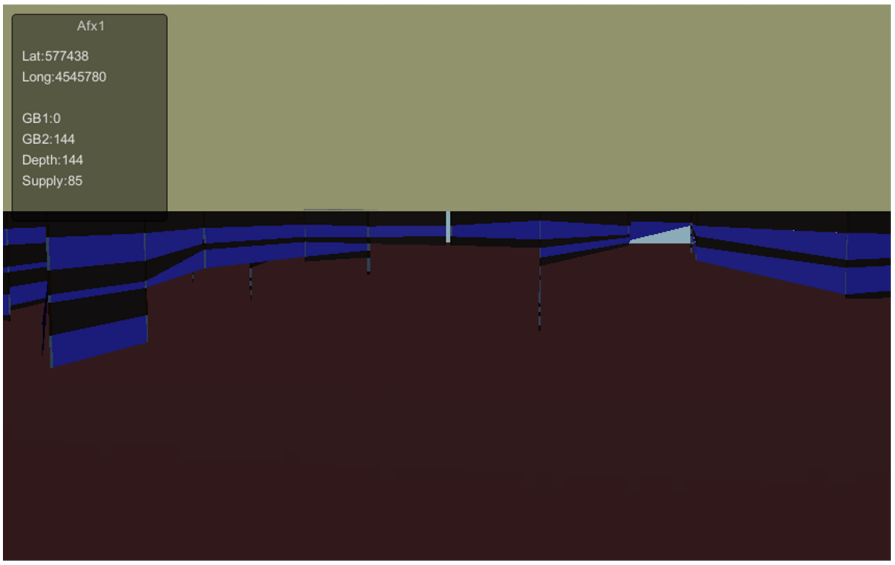
Ο παραπάνω προτεινόμενος αλγόριθμος περιγράφει τη διαδικασία τριγωνοποίησης και θα έχει σαν αποτέλεσμα την ένωση μεταξύ δύο γεωτρήσεων τεσσάρων σημείων των άκρων των γεωτρήσεων. Για να επιτευχθεί όμως η ένωση μεταξύ των σημείων αυτών θα πρέπει να υπολογιστεί και η γωνία σύνδεσης μεταξύ 2 σημείων Τριών διαστάσεων. Ο προτεινόμενος αλγόριθμος αυτός περιγράφεται ως ακολούθως :

Αλγόριθμος Υπολογισμού Γωνίας μεταξύ δύο Σημείων Τριών Διαστάσεων

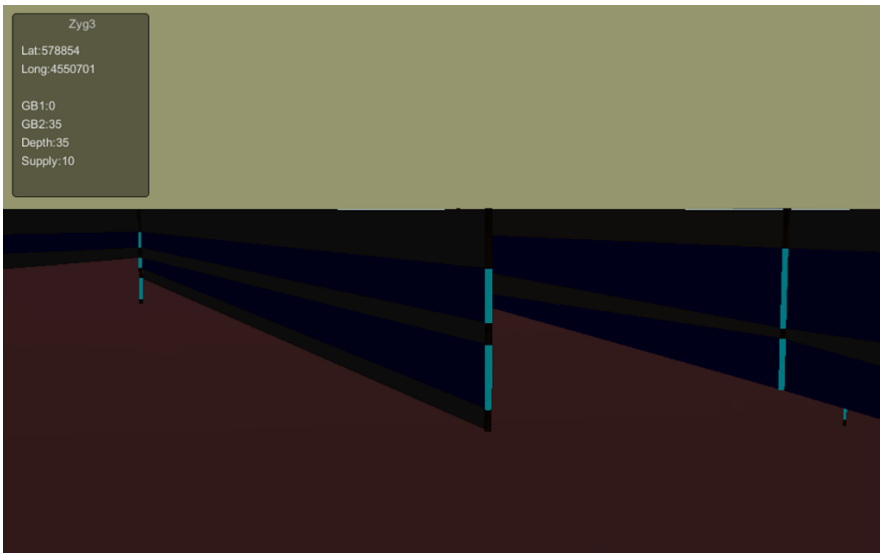
input: 3D Point From = $\{X_1, Y_1, Z_1\}$.
input: 3D Point To = $\{X_2, Y_2, Z_2\}$.
output: Angle = The relative angle between From and To 3D Points.
1: Clamp = Value Between a minimum oat and Maximum
Float value.// values are Dot(from,to) , -1 , 1
2: Dot = Float Value equals to the magnitudes of the two vectors multiplied by the
cosine of the angle between them.
3:if Points are in completely opposite directions then
4: Dot returns -1.
5:end if
6:if Points are in exactly the same direction then
7: Dot returns 1.
8:end if
9: 57.29578 = (180/PI). // convert radians to degree
10: Angle = Cos(Clamp(Dot(from,to),-1,1)*57.29578). // Angle output

Τα αποτελέσματα της ένωσης των σημείων παρουσιάζονται στις εικόνες 3.14 ή 3.15 :

Εικόνα 3.14 *Γεωτρήσεις στην περιοχή Ζυγός*



Εικόνα 3.15 *Αποτέλεσμα Τριγωνοποίησης και Διασύνδεσης Γεωτρήσεων. Νερό (Μπλε), Γεωλογικοί Σχηματισμοί (Καφέ), Βάση που αντιστοιχεί στο βαθύτερο σημείο είναι με Σκούρο Καφέ*



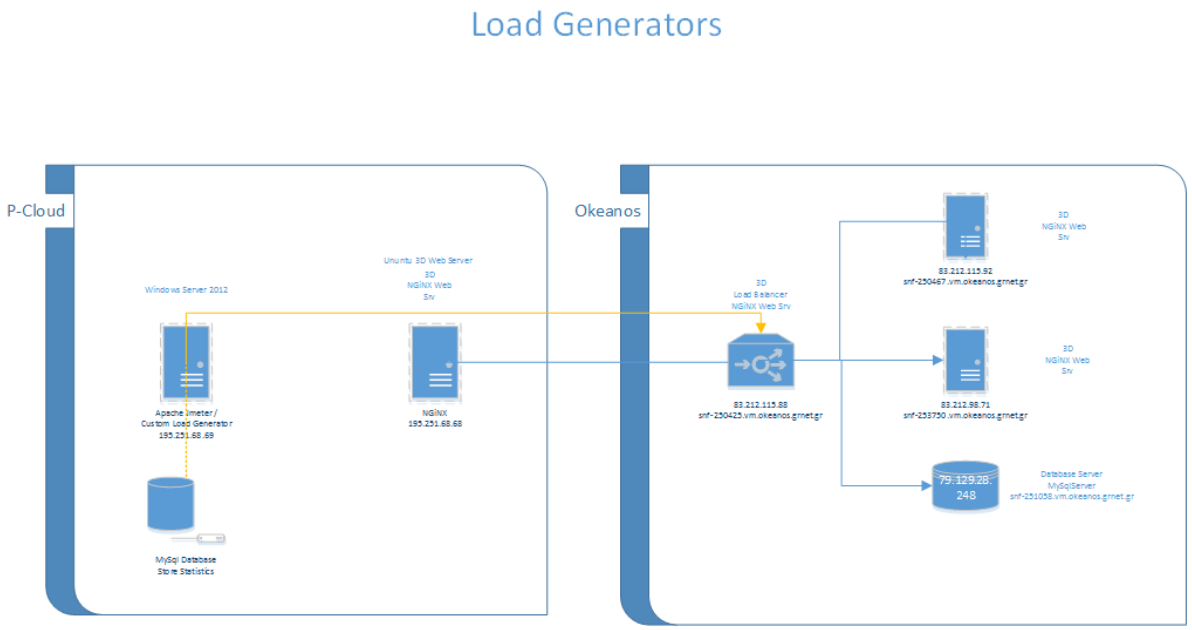
Εικόνα 3.16 *Αποτελέσματα ένωσης Αλγορίθμου Τριγωνοποίησης σε Οξείες Γωνίες*

3.4 Χρήση της Υπολογιστικής Νέφους

Όλη η υποδομή της Διατριβής έχει δοκιμαστεί και εγκατασταθεί σε υποδομή Υπολογιστικού Νέφους.

3.4.1 Διασύνδεση Συστημάτων Τεχνολογιών Σύννεφου

Η υποδομή φιλοξενείται στην υποδομή υπολογιστικού νέφους Okeanos καθώς και στην υποδομή υπολογιστικού νέφους που βρίσκεται υπό υλοποίηση για τις ανάγκες του Εργαστηρίου. Η διασύνδεση των συστημάτων παρουσιάζεται στην εικόνα 3.17:



Εικόνα 3.17 *Αρχιτεκτονική Διασύνδεσης Συστημάτων*

Όπως φαίνεται και από το ανωτέρω σχήμα, η υποδομή και η αρχιτεκτονική του συστήματος είναι χωρισμένη σε δύο μέρη, τα οποία διασυνδέονται μεταξύ τους.

Στο αριστερό μέρος της εικόνας παρουσιάζεται η υποδομή που είναι εγκατεστημένη στο p-Cloud (private cloud) και στο δεξιό μέρος βρίσκεται η υποδομή στον Okeanos.

Στο p-Cloud έχει εγκατασταθεί ένας ESXI Virtual Server σε ένα φυσικό μηχάνημα. Σε αυτό το σύστημα έχει εγκατασταθεί ένα εικονικό μηχάνημα με λειτουργικό σύστημα Ubuntu Server και λειτουργεί σαν ένας εξυπηρετητής διαδικτύου έχοντας στο λογισμικό του έναν NGiNX Web Server το οποίο έχει το τελικό προϊόν των Τριών Διαστάσεων.

Επίσης έχει εγκατασταθεί ένα έτερο εικονικό μηχάνημα με λειτουργικό Windows Server 2012 το οποίο λειτουργεί σαν Εξυπηρετητής Βάσης Δεδομένων Τεχνολογίας MySQL η οποία αποθηκεύει τα στατιστικά στοιχεία που παράγονται από τις Γεννήτριες Φορτίου.

Αυτό το μηχάνημα λειτουργεί επίσης και σαν Γεννήτρια Φορτίου (Load Generator), όπως θα αναλυθεί σε επόμενο κεφάλαιο.

Από την άλλη πλευρά υπάρχουν δύο εγκατεστημένοι εξυπηρετητές Δικτύου με λειτουργικό Ubuntu Server οι οποίοι λειτουργούν σαν εξυπηρετητές διαδικτύου τεχνολογίας NGiNX Web Server.

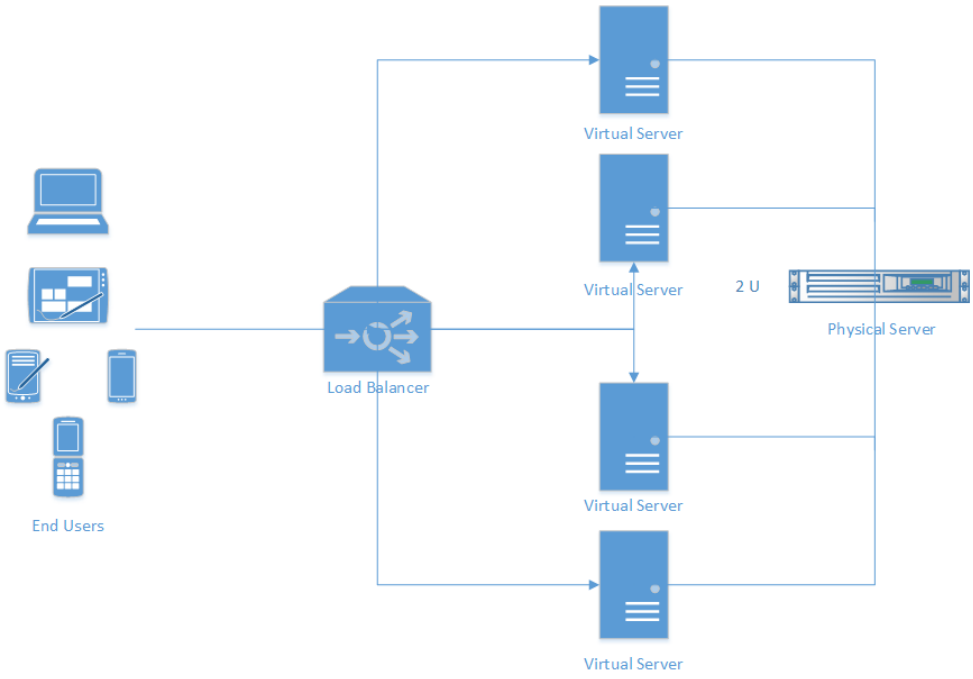
Επίσης είναι εγκατεστημένος ένας Εξυπηρετητής Βάσης Δεδομένων τεχνολογίας MySQL Server η οποία είναι εγκατεστημένη σε λειτουργικό Ubuntu Server και φέρει τα δεδομένα των γεωτρήσεων.

Τέλος, σε έτερο εικονικό μηχάνημα τεχνολογίας Ubuntu Server, είναι εγκατεστημένος ένας Εξισορροπητής Φορτίου ο οποίος είναι εγκατεστημένος πάνω σε ένα Εξυπηρετητή Διαδικτύου Τεχνολογίας NGiNX Web Server.Αυτή η εικονική μηχανή έχει σαν σκοπό να ανιχνεύει την κίνηση του δικτύου και ανάλογα με τον φόρτο των εικονικών μηχανών να διαμοιράζεται το φορτίο στους εξυπηρετητές της υποδομής.

3.4.2 Αποτελέσματα Χρήσης Εξισσοροπητών Φορτίου

Η ταχύτητα του συστήματος βελτιώθηκε σημαντικά χρησιμοποιώντας τεχνολογίες Ανοιχτού Λογισμικού σε σύγκριση με τεχνολογίες κλειστού Λογισμικού οι οποίες προκαλούσαν Διαρροές Μνήμης με αποτέλεσμα το σύστημα μετά από ένα μικρό χρονικό διάστημα να είναι μη λειτουργικό.

Όπως περιγράφτηκε στην ενότητα 2.5.2 ο εξισορροπητής φορτίου έχει σαν σκοπό την κατανομή του φορτίου στους εξυπηρετητές ανάλογα με τον αλγόριθμο που έχει τεθεί στην λειτουργικότητα του.



Εικόνα 3.18 *Λειτουργία Εξισορροπητή (LoadBalancing)*

Στην Εικόνα 3.18 βλέπουμε ότι σε ένα φυσικό μηχάνημα μπορούν να εγκατασταθούν πολλοί εικονικοί εξυπηρετητές. Το φορτίο εξαρτάται από τους χρήστες και ο εξισορροπητής φορτίου, ανάλογα με τις ρυθμίσεις που φέρει στο εσωτερικό του (π.χ Αλγόριθμος), κατανέμει το φορτίο αντίστοιχα.

Έτσι λοιπόν υποθέτοντας ότι η εφαρμογή χρησιμοποιείται από ένα μεγάλο αριθμό χρηστών, ένας εξυπηρετητής δεν είναι επαρκής για την ομαλή και αδιάλειπτη λειτουργία του συστήματος, και για αυτό το λόγο η υποδομή αποτελείται από μία φάρμα εξυπηρετητών (Web Farm), στην οποία είναι συνδεδεμένοι οι εξυπηρετητές μέσω του κατανεμητή φορτίου (Load Balancer).

Το configuration του συγκεκριμένου αρχείου είναι το κάτωθι :

```
//NGinNX Load Balancer default

upstream WellLoadBalancer{
      server 195.251.68.68;
      server 83.212.115.92;
      server 83.212.98.71;
```

```
}

server {
      listen   80; ## listen for ipv4; this line is default and
implied
      #listen   [::]:80 default ipv6only=on; ## listen for ipv6

      root /usr/share/nginx/www;
      index index.php index.html index.htm;

      # Make site accessible from http://localhost/
      server_name 83.212.124.24;

      location / {
            proxy_pass http://WellLoadBalancer;
            try_files $uri $uri/ /index.html;

      }
```

Στο πρώτο μέρος του Configuration δηλώνονται οι διευθύνσεις ip των εξυπηρετητών, δηλαδή οι ip των συνδεδεμένων μηχανημάτων που επιθυμούμε να γίνει κατανομή φορτίου. Οι ip είναι οι κάτωθι :

- 195.251.68.68-- Υποδομή p-Cloud
- 83.212.115.92-- Okeanos.grnet
- 83.212.98.71 -- Okeanos.grnet

Επομένως, ο συνδετικός κρίκος όλων των εικονικών μηχανημάτων είναι ο Load Balancer, επειδή εκείνος ανακατευθύνει το αίτημα του χρήστη στον εξυπηρετητή με τα λιγότερα αιτήματα και το λιγότερο φορτίο.

3.5 Γεννήτριες Φορτίου (Load Generators)

Κάθε σύστημα που κατασκευάζεται θα πρέπει να συνοδεύεται από μετρήσεις οι οποίες προσομοιάζουν και ελέγχουν την αποδοτικότητα της εκάστοτε εφαρμογής προκειμένου να γνωρίζουμε την αντοχή του συστήματος τόσο σε υλικό όσο και σε λειτουργικό επίπεδο.

Τα προγράμματα που εκτελούν τέτοιες μετρήσεις χωρίζονται σε δύο μέρη. Το ένα μέρος αποτελείται από μία γεννήτρια φορτίου (Load Generator) και το άλλο μέρος αφορά την καταγραφή των μετρήσεων για εξαγωγή στατιστικών μετρήσεων. Τα προγράμματα αυτά εκτελούν ελέγχους και δοκιμές που ονομάζονται stress test.

Αυτή η διαδικασία εφόσον παρέχεται και σαν υπηρεσία αποκαλείται Taas (Testing As A Service) **[15]**

Με τη διαδικασία του Ελέγχου ελέγχουμε την λειτουργικότητα, την χρηστικότητα, την σταθερότητα, την διαλειτουργικότητα, την ευκολία, το περιεχόμενο, την ευαισθησία των συστημάτων και για αυτό το λόγο είναι πάρα πολύ σημαντικός παράγοντας πριν διατεθεί το προϊόν στους χρήστες.**[15]**

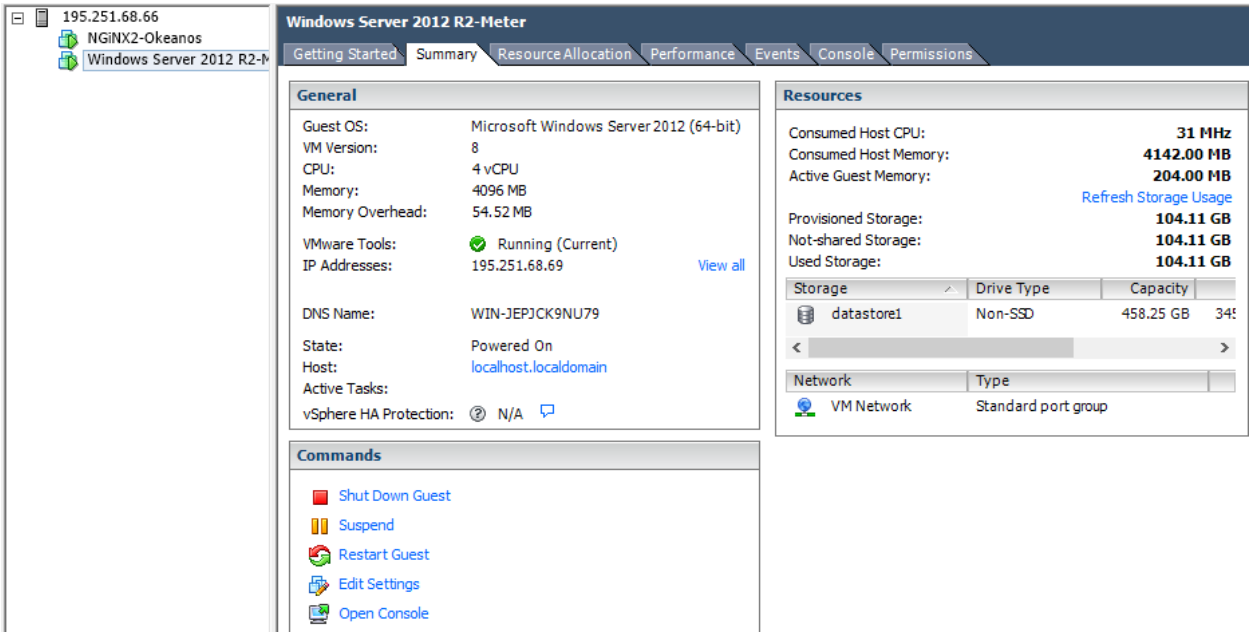
Για αυτό το λόγο στο υφιστάμενο σύστημα χρησιμοποιήθηκαν δύο εργαλεία δημιουργίας φορτίου τα οποία καταγράφουν στατιστικά κατά την εκτέλεση τους ο apache jmeter και ένα προτεινόμενο εργαλείο γεννήτριας φορτίου.

3.5.1 Διαδικασία Μετρήσεων μέσω Apache JMeter

Πριν αναλυθεί η διαδικασία των μετρήσεων μέσω του εργαλείου Apache JMeter είναι σημαντικό να αναλυθούν τα χαρακτηριστικά των εικονικών μηχανών.

Χαρακτηριστικά Μηχανής Windows Server

- OS: Windows Server 2012 R2
- CPU: 4vCPU
- RAM: 4096 MB
- HDD: 500 GB



Εικόνα 3.19 *WindowsServerSettings*

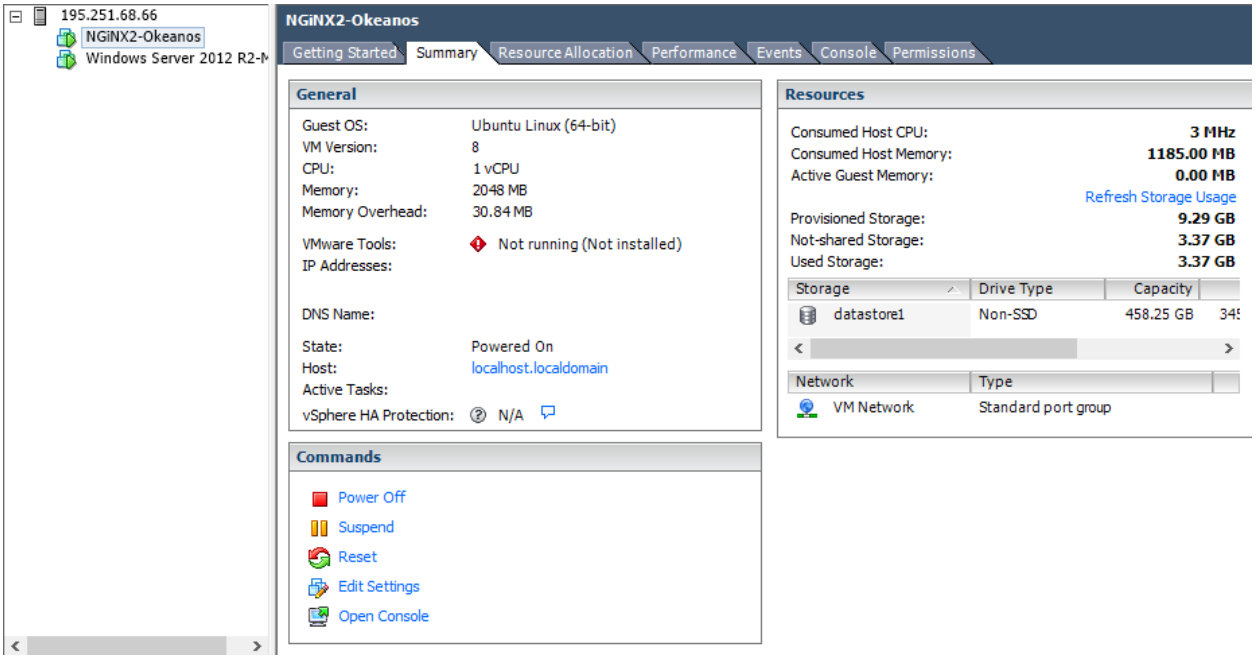
Σε αυτή την εικονική μηχανή έχουμε εγκαταστήσει τα εξής προγράμματα :

- Custom Load Generator
- Apache JMeter
- MySql Workbench
- MySql Server

Η συγκεκριμένη εικονική μηχανή προσομοιάζει μία γεννήτρια φορτίου (LoadGenerator) η οποία αποθηκεύει τις μετρήσεις της σε μία Βάση Δεδομένων για στατιστική μελέτη.

Και η δεύτερη εικονική μηχανή του συστήματος με χαρακτηριστικά :

- OS: Ubuntu Server
- CPU: 1vCPU
- RAM: 4048 MB
- HDD: 500 GB



Εικόνα 3.20 *NGiNXWebServer*

Στην υποδομή του Okeanos.grnet.gr έχουμε εγκαταστήσει 4 εικονικές μηχανές τεχνολογίας ανοιχτού λογισμικού.

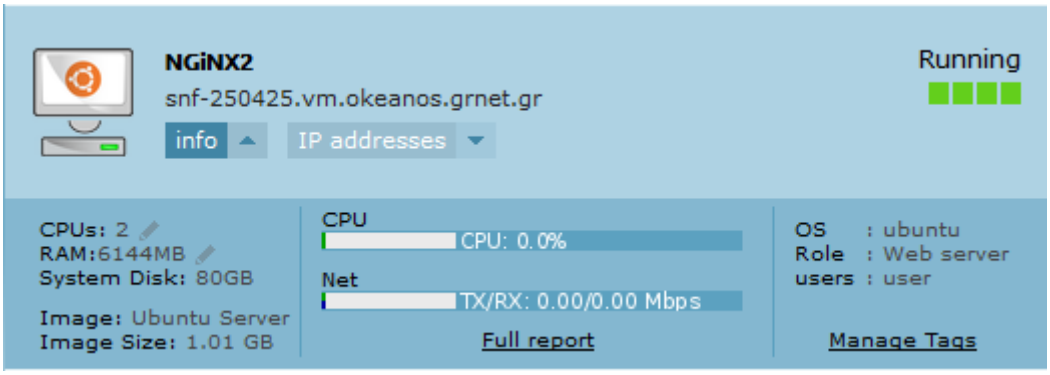
Αναλυτικότερα έχουν εγκατασταθεί 2 NGiNX Web Server οι οποίοι φέρουν την τελική εφαρμογή τριών διαστάσεων, καθώς και από ένα Web Service σε κάθε Web Server το οποίο επικοινωνεί με την Βάση Δεδομένων που βρίσκεται στην Τρίτη Εικονική Μηχανή τεχνολογίας MySql.

Η τέταρτη εικονική μηχανή είναι ένας NGiNX Web Server ο οποίος συμπεριφέρεται ως ένας κατανεμητής Φορτίου (Load Balancer), κατανέμει δηλαδή το φορτίο ισόποσα σύμφωνα με τον αλγόριθμο Round Robin σε όλους τους εξυπηρετητές (NGiNX Web Servers)

Οι Εικονικές αυτές μηχανές έχουν τα ίδια χαρακτηριστικά :

- OS: Ubuntu Server
- CPU: 2
- RAM: 6 GB

- HDD: 80 GB



Εικόνα 3.21 *NGiNX Hardware Settings*

Το εργαλείο Apache JMeter είναι ένα εξ' ολοκλήρου πρόγραμμα ανοιχτού κώδικα υλοποιημένο με τεχνολογία Java, το οποίο επικεντρώνεται στην προσομοίωση και στον έλεγχο της εφαρμογής με σενάρια τα οποία μπορούν να υλοποιηθούν και να εφαρμοστούν για ανοιχτό χρονικό διάστημα. Το εργαλείο αυτό μας δίνει τη δυνατότητα να κάνουμε test την εφαρμογή σε διάφορα επίπεδα του δικτύου καθώς και της εφαρμογής .

Είναι υλοποιημένο με πολυνηματική αρχιτεκτονική (multithreading) για την προσομοίωση πολλών χρηστών. Κάθε νήμα αντιστοιχεί σε ένα χρήστη της εφαρμογής που ελέγχεται.
Πιο συγκεκριμένα έχει τη δυνατότητα να παράγει φορτίο και να διεκπεραιώνει test σε πολλά πρωτόκολλα όπως :

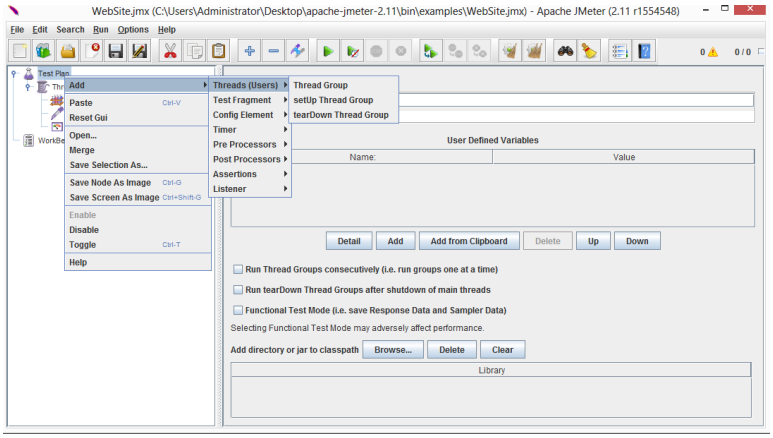
- Web - HTTP, HTTPS
- SOAP
- FTP
- Database via JDBC
- LDAP
- Message-oriented middleware (MOM) via JMS
- Mail - SMTP(S), POP3(S) and IMAP(S)
- MongoDB (NoSQL)

- Native commands or shell scripts
- TCP

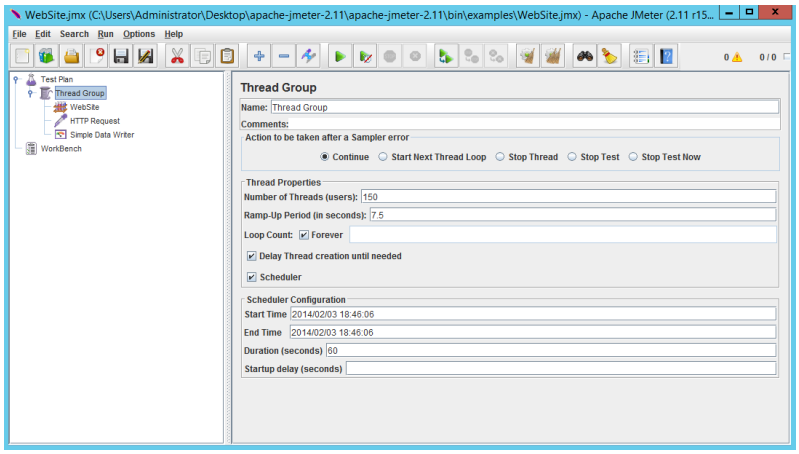
Κατά την εκτέλεση των δοκιμών και των ελέγχων μέσω του JMeter σε ιστοσελίδες , το εργαλείο αυτό δεν εκτελεί ενέργειες οι οποίες είναι γραμμένες σε γλώσσα Javascipt, εκτελεί μόνο τον κώδικα HTML. Για την καταγραφή των μετρήσεων υλοποιήθηκε ένα σενάριο μέσω του εργαλείου Ανοιχτού Λογισμικού Apache JMeter. Αυτό το πρόγραμμα μας δίνει την δυνατότητα να υλοποιήσουμε ένα σενάριο σύμφωνα με τις προδιαγραφές που θέτουμε εμείς μέσω αρθρωμάτων(modules) που υπάρχουν υλοποιημένα στο πρόγραμμα αυτό.
Το σενάριο που επιθυμούμε έχει σαν σκοπό να προσομοιάσει 150 χρήστες οι οποίοι εκτελούν αιτήματα (Web Requests) για χρονικό διάστημα μίας ημέρας (24 ωρών).

Η διαδικασία που ακολουθήθηκε έχει ως εξής :

α) Εισαγωγή ενός ThreadGroup



Εικόνα 3.22 *Προσομοίωση Χρηστών*

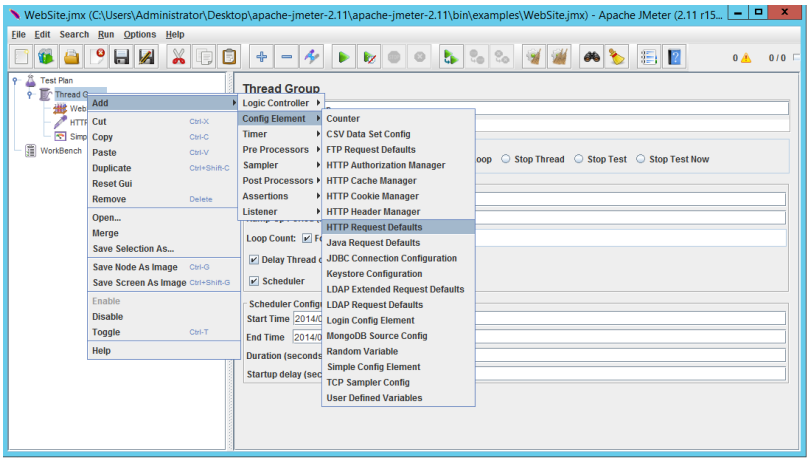


Εικόνα 3.23 *Ρυθμίσεις Apache Jmeter σε ThreadGroupProperties*

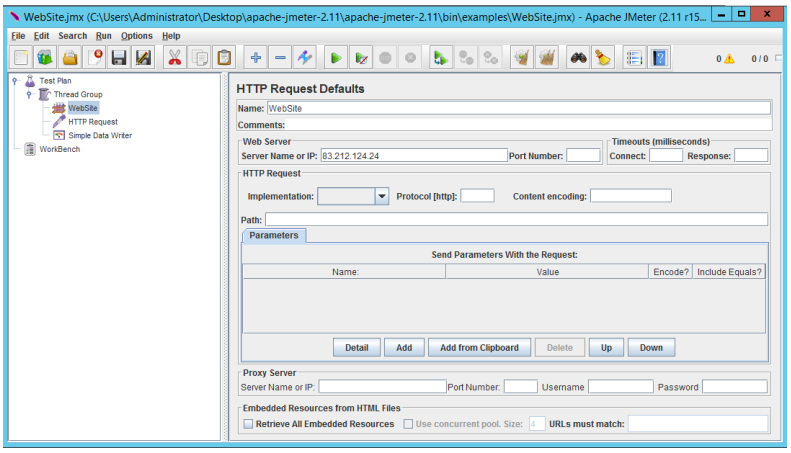
Αρχικά όπως φαίνεται στην εικόνα 3.22 εισάγουμε το άρθρωμα ThreadGroup Και στη συνέχεια εισάγουμε τις ρυθμίσεις σε αυτό όπως φαίνεται στην εικόνα 3.23.

Στο συγκεκριμένο τμήμα θα πρέπει να καθορίσουμε την πολιτική του σεναρίου που επιθυμούμε να εκτελεστεί. Η πολιτική μας είναι ότι 150 χρήστες που ισοδυναμούν με 150 Threads θα εκτελούν αιτήματα σε ένα server.

Ως Ramp-up Period ορίζεται ο χρόνος που χρειάζεται μέχρι όλοι οι χρήστες να είναι έτοιμοι να εκκινήσουν τα αιτήματα. Με LoopCount ενεργοποιώντας την αναγραφόμενη επιλογή Forever επιθυμούμε αυτή η διαδικασία να εκτελείται συνέχεια και με τον Scheduler βάζουμε την χρονική διάρκεια που θα εκτελεστεί το συγκεκριμένο σενάριο.



Εικόνα 3.24 *Εισαγωγή Αρθρώματος HttpReqest*



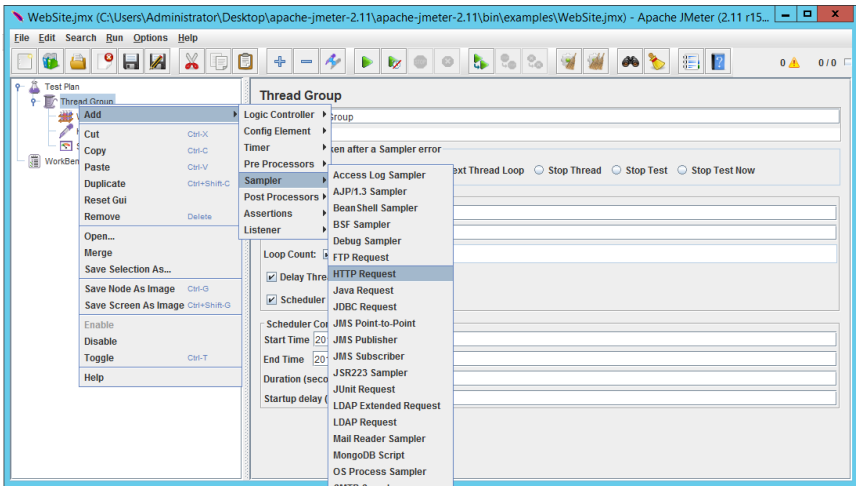
Εικόνα 3.25 *Εισαγωγή Σενάριού HTTPRequest*

Στην συνέχεια εισάγουμε το άρθρωμα HTTP Request Defaults όπως φαίνεται και στην εικόνα 3.24.

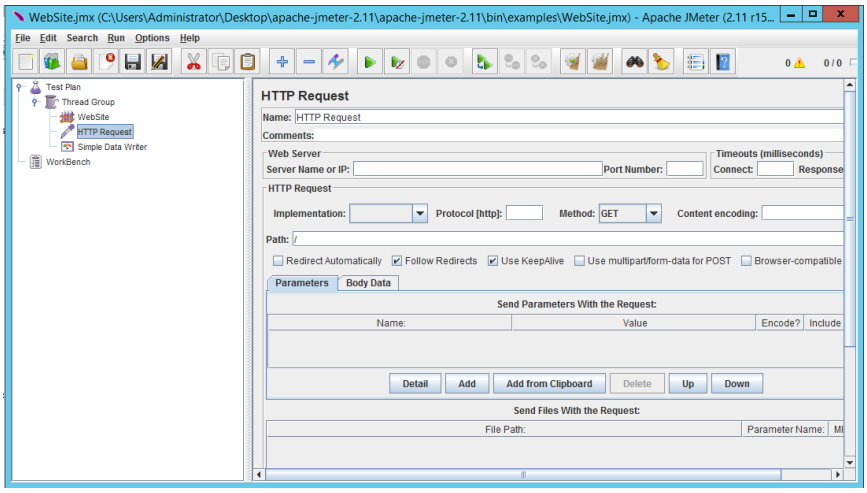
Με αυτό το άρθρωμα εισάγουμε το πλάνο που επιθυμούμε για να εκτελέσουμε τα αιτήματα.

Στην συγκεκριμένη περίπτωση εισάγουμε την διεύθυνση του Load Balancer (Εικόνα 3.25) στο πεδίο ServerName or IP, ο οποίος θα ανακατευθύνει τα αιτήματα ανάλογα με τον αλγόριθμο Round Robin και το φόρτο του συστήματος στους αντίστοιχους Web Server που φαίνονται στο σχήμα της Εικόνας 3.18.

Στην συνέχεια με το άρθρωμα HTTP Request εισάγουμε τις ρυθμίσεις εκείνες όπου θα στείλουμε το αίτημα (HTTP Request).

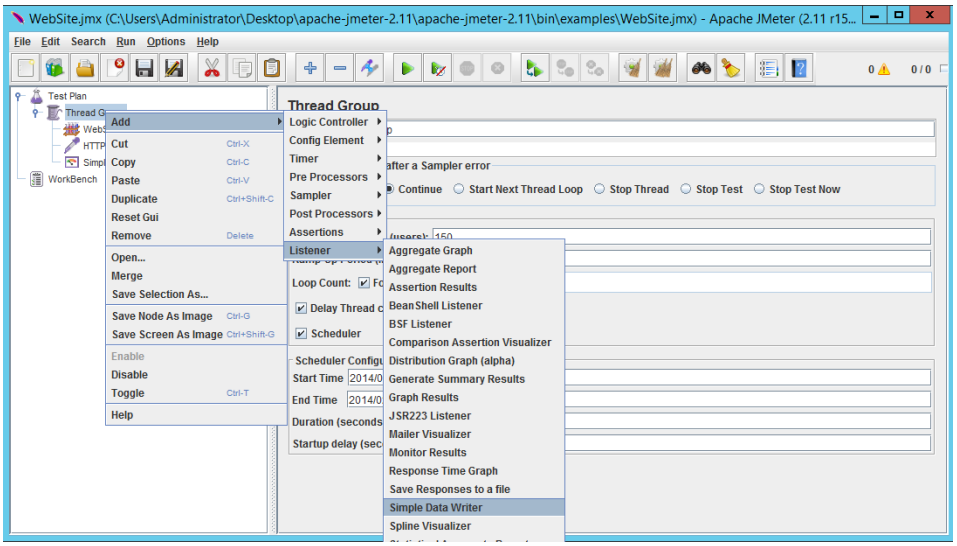


Εικόνα 3.26 *Εισαγωγή Αρθρώματος SamplerHttp Request*

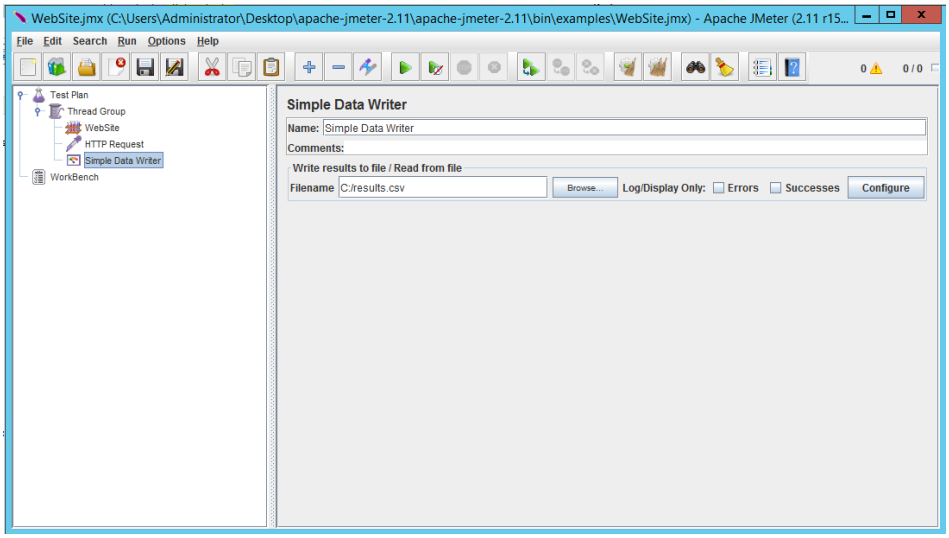


Εικόνα 3.27 *Εισαγωγή Ρυθμίσεων SamplerData*

Με την ιδιότητα Path εισάγουμε την διεύθυνση στην οποία θα κατευθύνουμε το αίτημα και εμείς στην συγκεκριμένη περίπτωση το ερώτημα το κατευθύνουμε στην σελίδα / (root path) καθώς η σελίδα μας είναι η index.html η οποία είναι προκαθορισμένη στο αρχείο ιδιοτήτων του εκάστοτε εξυπηρετητή NGiNX Web Server της υποδομής.



Εικόνα 3.28 *Εισαγωγή Listener Simple Data Writer*



Εικόνα 3.29 *Εισαγωγή Αρχείου Καταγραφών*

Στην συνέχεια εισάγουμε ένα άρθρωμα που αφορά την καταγραφή των μετρήσεων σε ένα αρχείο της μορφής .csv για εξαγωγή αποτελεσμάτων.

Είναι σημαντικό να αναφερθεί ότι αν εισάγουμε Listener αποτελεσμάτων τα οποία εξάγουν γραφικά αποτελέσματα, αυτό έχει σαν αποτέλεσμα να επηρεάζονται οι τιμές και ο τελικός αριθμός των αιτημάτων που εκτελούνται στον εξυπηρετητή, για αυτό το λόγο η καλύτερη τακτική είναι να αποθηκευτούν τα αποτελέσματα σε ένα αρχείο και να εξαχθούν εκ των υστέρων τα στατιστικά αποτελέσματα.

3.5.2 Δημιουργία Προγράμματος Γεννήτριας Φορτίου

Στα πλαίσια της διατριβής κατασκευάστηκε ένα εργαλείο παραγωγής φορτίου.
Το εργαλείο αυτό υλοποιήθηκε με τις ακόλουθες τεχνολογίες :

- Console Application
- C#
- .Net Framework 4.5
- Tasks μέσω TPL (Task Parallel Library) **[16]**

Για την κατασκευή του χρησιμοποιήθηκαν οι τελευταίες τεχνολογίες του .NET Framework4.5 σε πολυνηματική αρχιτεκτονική μέσω της βιβλιοθήκης TPL (Task Parallel Library)**[16]**.

Η δυσκολία που εμφανίζεται σε αυτά τα προγράμματα είναι ο έλεγχος και η αποδοτικότερη χρήση των πόρων του συστήματος που εκτελεί την προσομοίωση, καθώς λόγω της αυξημένης απαίτησης σε πόρους της επεξεργαστικής Ισχύος και σε μνήμη αυτό μπορεί να έχει σαν αποτέλεσμα το σύστημα να μην λειτουργεί στο βέλτιστο.
Για αυτό το λόγο ο προγραμματισμός θα πρέπει να είναι προσεκτικός, και θα πρέπει να γίνεται κατανομή των αιτημάτων (Web Requests) με τέτοιο τρόπο, έτσι ώστε το σύστημα να μην αναγκαστεί να τερματιστεί χωρίς την άδειά μας και να έχουμε απρόβλεπτα αποτελέσματα.
Για να γίνει κάτι τέτοιο εφικτό, χρησιμοποιήθηκε το .NetFramework 4.5 με χρήση της βιβλιοθήκης TPL η οποία εμφανίζει βελτιώσεις και καινούριους τρόπους διαχείρισης των διεργασιών του επεξεργαστή.
Η βιβλιοθήκη που χρησιμοποιήθηκε ονομάζεται TaskParallelism (Task Parallel Library)**[16]** και είναι η συνέχεια της δεξαμενής διεργασιών (ThreadPools) που ήταν εξέλιξη των διεργασιών (Threads).
Ο παραλληλισμός των Task αναφέρεται σε ένα ή περισσότερα μεμονωμένα Task που εκτελούνται ταυτόχρονα. Ο λόγος που χρησιμοποιήθηκε η τεχνολογία αυτή είναι επειδή έχει δύο βασικά πλεονεκτήματα.

α) Τα Task εισάγονται σε μία ουρά μίας δεξαμενής από διεργασίες (ThreadPool) και χρησιμοποιούνται στο κατώτερο επίπεδο αλγόριθμοι που κάνουν κατανομή φορτίου (Load Balancing) για την μεγιστοποίηση του Throughput.

β) Γίνεται πιο εύκολα ο έλεγχος των διεργασιών, καθώς μας δίνεται η δυνατότητα να προγραμματίσουμε μέσω της βιβλιοθήκης τις διεργασίες αυτές που περιλαμβάνουν μεγάλο έλεγχο και διαχείριση των διεργασιών αποφεύγοντας τα λάθη.

Μας δίνεται η δυνατότητα να προσομοιώσουμε τόσους χρήστες, όσα είναι και τα threads που μας επιτρέπονται.

Η λογική του συστήματος είναι να εκτελέσουμε ταυτόχρονα αιτήματα (Web Requests) σε μία συγκεκριμένη ιστοσελίδα(Web Page) και να μετρήσουμε τα αποτελέσματα από κάθε αίτημα.

Το συγκεκριμένο πρόγραμμα διασυνδέεται με μία Βάση δεδομένων τεχνολογίας MySQL όπου αποθηκεύονται τα τρέχοντα στατιστικά.

Από την διαδικασία αυτή εξάγουμε τα ακόλουθα χαρακτηριστικά :

- **Thread**: Αντιστοιχεί σε ένα αριθμό που δείχνει το πρωτεύον κλειδί της Διεργασίας (Thread) του συστήματος που εκτέλεσε το συγκεκριμένο Request
  `Thread.CurrentThread.ManagedThreadId`
- **Request Status**: Όταν εκτελούμε αίτημα σε μία σελίδα τότε ο Web Server Μας επιστρέφει ένα Νούμερο που αντιστοιχεί στο RequestStatus . Υπάρχουν διάφορες καταστάσεις που αντιστοιχούν σε κωδικούς, τα οποία δείχνουν αν ένα Request ήταν επιτυχές ή αποτυχημένο και αν ήταν αποτυχημένο σε ποιο επίπεδο δικτύου προκλήθηκε το πρόβλημα αυτό.

Πιο αναλυτικά υπάρχουν τα Web Requests που έχουν εφαρμοστεί στο πρόγραμμα όπως εμφανίζονται στο πίνακα 3.2 .

Κατάσταση	Περιγραφή	Ανάλυση
200	Statusok	Το αίτημα ήταν επιτυχημένο.
204	NoContent	Το αίτημα ήταν επιτυχημένο και η ανταπόκριση (response) ήταν κενή.
403	Forbidden	Το αίτημα μας δείχνει ότι ο εξυπηρετητής αρνήθηκε να εκτελέσει το αίτημα.

404	NotFound	Το αίτημα δεν ανταποκρίθηκε γιατί δεν υπάρχει η πηγή στον εξυπηρετητή.
500	InternalServerError	Είναι ένα γενικό πρόβλημα το οποίο οφείλεται στον εξυπηρετητή.
502	BadGateway	Ο ενδιάμεσος Proxy εξυπηρετητής έλαβε μία ανταπόκριση από έναν άλλο Proxy ή από τον πρωτεύον εξυπηρετητή.
503	ServiceAbavailable	Ο εξυπηρετητής είναι προσωρινά μη διαθέσιμος, συνήθως λόγω υψηλού φορτίου ή λόγω συντήρησης.

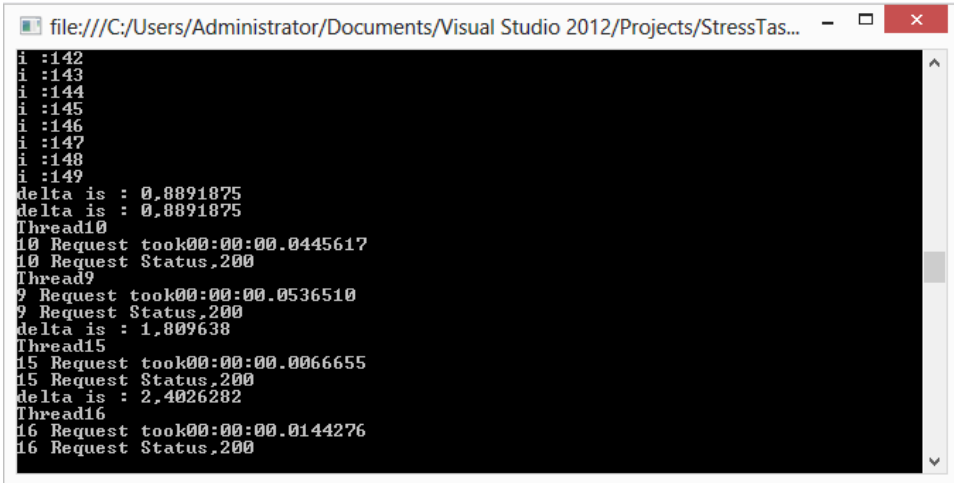
Πίνακας 3.2 *Κωδικοί - Καταστάσεις Ανταποκρίσεων Αιτημάτων (Web Response Status)*

Πηγή: http://msdn.microsoft.com/en-us/library/system.net.httpstatuscode%28v=vs.110%29.aspx

- **ResponseTime:** είναι το χρονικό διάστημα σε ms που απαιτήθηκε για να ανταποκριθεί το εκάστοτε αίτημα, από τη στιγμή που ξεκινάει το κάθε αίτημα μέχρις ότου ο εξυπηρετητής διαβάσει όλο το μέγεθος του περιεχομένου (Content Length) που επιστρέφει το σύστημα, δηλαδή το μέγεθος του κώδικα HTML.
- **ServerIP:** Λόγω του ότι το σύστημα ήταν τοποθετημένο σε υποδομή Υπολογιστικού Νέφους υλοποιημένο με κατανεμητή φορτίου πίσω από τρεις εξυπηρετητές, αποθηκεύεται στη βάση η διεύθυνση δικτύου του εκάστοτε αιτήματος.
- **Length:** Το μέγεθος του περιεχομένου της σελίδας που εκτελείται μέσω του αιτήματος.
- **TimeStarted:** Κάθε φορά που εκτελείται ένα αίτημα αποθηκεύεται στη Βάση δεδομένων η χρονική στιγμή που πραγματοποιείται το ερώτημα. Πρόκειται για μία χρονοσφραγίδα της μορφής

```
string NowTime = DateTime.Now.ToString("yyyyMMddHHmmssffff");
```

- **Throughput:** Είναι ο συντελεστής απόδοσης του συστήματος και αντιστοιχεί στο κλάσμα του μέγεθους του περιεχομένου της επιστρεφόμενης σελίδας προς τον χρόνο που απαιτήθηκε για να ολοκληρωθεί το αίτημα αφού επιστραφεί το περιεχόμενο του αιτήματος.



Εικόνα 3.30 *Stress Test Console Application*

Όπως φαίνεται και στην εικόνα 3.30 το πρόγραμμα που υλοποιήθηκε υπολογίζει τις τιμές που χρειαζόμαστε και στην συνέχεια τις αποθηκεύει σε μία βάση δεδομένων για περαιτέρω ανάλυση και εξαγωγή στατιστικών μετρήσεων και γραφημάτων.

ΚΕΦΑΛΑΙΟ 5 ΕΠΙΛΟΓΟΣ

5.1 Αξιοποίηση Διατριβής

Η εργασία αυτή υλοποιήθηκε στο πλαίσιο διατριβής, ωστόσο το τελικό προϊόν λόγω της υποδομής και της διαλειτουργικότητας, είναι αποτέλεσμα συνεργασίας και διεπιστημονικού ενδιαφέροντος. Το αρχικό πόνημα, δημιουργήθηκε για να αξιοποιηθούν τα δεδομένα του «Εθνικού Μητρώου Υδρογεωτρήσεων» (ΥΠΕΚΑ, αρ.πρωτ. 1042/12.9.2013, ΑΔΑ: ΒΛ9Κ0-ΝΝΤ). Βάσει του τελικού παραδοτέου, η εφαρμογή μπορεί να υποστηρίξει ένα μεγάλο αριθμό χρηστών μέσω διαδικτύου, και να καλύψει ένα βασικό αριθμό λειτουργιών οι οποίες δεν υπάρχουν σε άλλου είδους τέτοια εφαρμογή, με τόση ευκολία στην πρόσβαση και στην ανανέωση των δεδομένων. Με την προϋπόθεση ότι το προϊόν θα χρησιμοποιηθεί από το κοινό του και η βάση του θα ανανεώνεται από τους επιστήμονες του χώρου, τότε θα αποτελέσει το πιο ολοκληρωμένο πρόγραμμα οπτικής αναπαράστασης υδροφόρου Ορίζοντα καθώς και την πιο πλήρη Βάση Δεδομένων σε θέματα που αφορούν τις Υδρογεωτρήσεις και τα Φίλτρα αυτών.

Επίσης η εφαρμογή αυτή θα μπορούσε να αξιοποιηθεί από τα Πανεπιστήμια και ειδικά από τους Επιστήμονες της Γεωλογίας και της Υδροπληροφορικής στο πλαίσιο μελετών και εκπαίδευσης καθώς και περιήγησης στο Εικονικό Υπόγειο Υδροφόρο Ορίζοντα για έλεγχο και επιβεβαίωση των αναγραφόμενων τιμών στα έντυπα καταγραφών των Γεωτρήσεων.

5.2 Μελλοντικές Προοπτικές

Αρχικά θα μπορούσε να γίνει καλύτερη προσέγγιση στο τρισδιάστατο ανάγλυφο και να γίνει με διαφορετικό τρόπο προκειμένου να έχουμε καλύτερα αποτελέσματα.

- Ένας πιθανός τρόπος δημιουργίας του τρισδιάστατου ανάγλυφου είναι μέσω της στατιστικής ανάλυσης των δεδομένων προκειμένου να μην υπάρχει τόσο μεγάλη διαφορά στα μεσαία ύψη, π.χ στα 0-500 μέτρα.

- Επίσης ένας άλλος τρόπος είναι μέσω WMS Server να φορτωθούν υπό μορφή πυραμίδας (LOD Level Of Detail) τα δεδομένα σε ένα server και να αντλούνται από αυτόν μέσω Spatial Database και δυναμικής φόρτωσης και εκφόρτωσης Tile, δηλαδή απαίτηση δημιουργίας μηχανησμού Tile Management με φόρτωση Δορυφορικών Δεδομένων μέσω υπηρεσιών διαδικτύου της Microsoft, ή ESRI ή GoogleEarth.

- Ακόμα μέσω του εργαλείου Meshlab **[17]** θα μπορούσαμε να έχουμε καλύτερη ποιότητα αποτελέσματος στο τρισδιάστατο ανάγλυφο αλλά λόγω του μεγάλου όγκου του τελικού αρχείου δεν γίνεται να επεξεργαστεί σε μεγάλη κλίμακα και απαιτούνται υπολογιστές με υψηλές προδιαγραφές Υλικού.

- Από πλευράς τριγωνοποίησης του γεωγραφικού υπόβαθρου ο αλγόριθμος Delaunay πιθανόν να έχει καλύτερα αποτελέσματα από πλευράς ενοποίησης των γεωτρήσεων. Ο συγκεκριμένος αλγόριθμος δημιουργεί ένα ενιαίο τμήμα μέσω του τριγωνισμού όλων των σημείων που υπάρχουν στο χώρο σε αντίθεση με τον αλγόριθμο Ear Clipping ο οποίος δημιουργεί πολύγωνα χωρίς να ενώνονται τα ενδιάμεσα τμήματα.

- Η εισαγωγή των φίλτρων – πετρωμάτων που υπάρχουν στο υπόβαθρο σε τρισδιάστατη αναπαράσταση υπό μορφή τρισδιάστατων Αντικειμένων.

- Καλύτερη απεικόνιση των πηγαδιών..

- Εκμετάλλευση του Geolocation (HTML5, GPS) Χρήστη και ανάλογα με το πού βρίσκεται να εμφανίζεται το γεωγραφικό υπόβαθρο σε έκδοση για κινητά και για Tablet.

- Εργαλεία πάνω στο 3D για μέτρηση αποστάσεων μεταξύ σημείων καθώς και αντίστοιχων λειτουργιών

- Ενεργοποίηση διαφορετικών Layer (π.χ εισαγωγή Οδικού Χάρτη ή κάτι αντίστοιχου) μέσω αξιοποίησης των OSMData μέσω του OpenStreetMaps και της ESRI 3DCityEngine.

- Ενεργοποίηση καλύτερου Οπτικού Αποτελέσματος μέσω εικόνας από Δορυφορικές Φωτογραφίες.

- Διασύνδεση με το ESRI 3DCityEngine και μέσω OpenStreetMaps για εισαγωγή του Τρισδιάστατου μοντέλου δρόμων και πόλεων.

- Διασύνδεση με Web Application Site για εισαγωγή και ενημέρωση των δεδομένων της εφαρμογής .

- Real Time Χαρακτηριστικά μέσω δεδομένα ροής.

ΑΝΑΦΟΡΕΣ

[1] Syggros I. (2004). *Transformation of Coordinates for Geographical Data in Greece*, 3rd Pan-Hellenic Conference of Hellenic Corporation for GIS Systems (HellasGI), 11-12 March.

[2] Βέης Γ., (1992). *Ανώτερη Γεωδαισία, Εθνικό Μετσόβιο Πολυτεχνείο, Τμήμα Αγρονόμων και Τοπογράφων Μηχανικών*.

[3] Ran L. (2010). *A Comparison of Ear Clipping and a New Polygon Triangulation Algorithm* S-801 76 Gavle, Sweden. pp. 5-7.

[4] Fisher M., Springborn B., Schroder P., Bobenko A. (2006). *An Algorithm for the Construction of Intrinsic Delaunay Triangualtions with Applications to Digital Geometry Processing*, ACM SIGGRAPH 2006 Courses, pp.69-74, ISBN: 1-59593-364-6.

[5] Huan L. and Sewook W. (2009). *Web Server Farm in the Cloud: Performance Evaluation and Dynamic Architecture*, Springer. First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009, vol. 5931, pp. 369-380, ISSN: 0302-9743.

[6] Piorkowski A., Kempny A., Hajduk A., Strzelczyk J. (2010). *Load Balancing for Eterogenous Web Servers* , Department of Geoinformatics and Applied Computer Science AGH University of Science and Technology, Cracow, Poland CN 2010.CCIS, vol. 79, Springer, Heidelberg (2010), pp. 189-198.

[7] Meng Y. Ayanni R., (2001). *Comparison of Load Balancing Strategies on Cluster-based Web Servers*, Department of Computer Science National University of Singapore and Department of Microelectronics and Information Technology Royal Institute of Technology (KTH). *SIMULATION* November 2001 vol.77 no. 5-6 pp.185-195, DOI:10.1177/003754970107700504.

[8] Humphrey M., et al. (2013). *CloudDRN: A Lightweight, End-to-End System for Sharing Distributed Research Data in the Cloud*, IEEE 9th International Conference on e-Science, 22-25 Oct., pp.254-261.

[9] Muzafar A. et al. (2011). *Cloud Computing: A solution to Geographical Information Systems (GIS)*, International Journal on Computer Science and Engineering (IJCSE), 3 (2). ISSN: 0975-3397.

[10] Swarkar N. and Shankar R. (2013). *A Survey of Load Balancing Techniques in Cloud Computing*, International Journal of Engineering Research & Technology (IJERT), 2(8), ISSN: 2278-0181.

[11] Εθνικό Δίκτυο Έρευνας & Τεχνολογίας (ΕΔΕΤ), Grnet.gr, (2014). *Okeanos IaaS Cloud | GRNET*. Διαθέσιμο: http://okeanos.grnet.gr. Προσπελάστηκε : 15 Απριλίου 2014

[12] Εθνικό Δίκτυο Έρευνας & Τεχνολογίας (ΕΔΕΤ), Grnet.gr, (2014). *Okeanos IaaS Cloud | GRNET*. Διαθέσιμο: https://www.grnet.gr/el/node/317 . Προσπελάστηκε : 15 Απριλίου 2014.

[13] Qgis.org, (2014). Διαθέσιμο: http://www.qgis.org/en/site/index.html . Προσπελάστηκε : 15 Απριλίου 2014.

[14] Technologies, U. (2014). *Unity Script Reference:* Docs.unity3d.com. Διαθέσιμο: https://docs.unity3d.com/Documentation/ScriptReference/WWW.html . Προσπελάστηκε : 15 Απριλίου 2014 .

[15] Narasimba M.S. , Suma V. (2014) *A Study on Cloud Testing Tools*, Springer International Publishing Switzerland vol. 248, DOI:10.1007/978-3-319-03107-1_66.

[16] Msdn.microsoft.com, (2014). *Task Parallelism (Task Parallel Library)*. Διαθέσιμο: http://msdn.microsoft.com/en-us/library/dd537609%28v=vs.110%29.aspx Προσπελάστηκε : 15 Απριλίου 2014].

[17] Cignoni, P. (2014). *MeshLab*. Meshlab.sourceforge.net. Διαθέσιμο: http://meshlab.sourceforge.net/ . Προσπελάστηκε : 15 Απριλίου 2014.

[18] Allaby M. (2008). *Dictionary of Earth Sciences*, Oxford University Press, 3rd Edition, ISBN 978-0-19-921194-4.

[19] Google Cloud Platform, Google App Engine (2014). Διαθέσιμο : https://cloud.google.com/products/appengine/?gclid=CjgKEAjw2dqcBRC2qLXjpfxjnQSJAAeYF5LiKDSVO3u74c5G0wxOIPwvHiud_Wz2l3oI4dy1ZacsfD_BwE . Προσπελάστηκε 11 Ιουνίου 2014 .

[20] Microsoft Azure . (2014) Διαθέσιμο: http://azure.microsoft.com/en-us/ . Προσπελάστηκε 11 Ιουνίου 2014 .

Παράρτημα :

Όλος ο κώδικας που έχει χρησιμοποιηθεί για τους σκοπούς της διπλωματικής εργασίας υπάρχει σε CD.

Κώδικας CustomLoadGenerator

```
using MySql.Data.MySqlClient;
using System;
using System.Diagnostics;
using System.IO;
using System.Net;
using System.Threading;
using System.Threading.Tasks;

publicclassProgram
{
publicstaticDateTime start = DateTime.Now;
publicstaticstring myConnectionString;

publicstaticvoid Main()
    {
        myConnectionString = "server=###;uid=#######;" + "pwd=########;database=test;";
var tokenSource2 = newCancellationTokenSource();
CancellationToken ct = tokenSource2.Token;
constint Max_Threads = 150;     //users
Task[] tasks = newTask[Max_Threads];

for (int i = 0; i < Max_Threads; i++)
        {
Console.WriteLine("i :" + i);
            tasks[i] = newTask(() => DoComputation()); //Call DoComputation Function
        }
foreach (Task task in tasks)
            {
                task.Start(); //Begin Task for every task
            }
Console.ReadLine();
    }
privatestaticvoid DoComputation()
    {
while (true)
        {
string NowTime = DateTime.Now.ToString("yyyyMMddHHmmssffff");;
var delta = (DateTime.Now - start).TotalSeconds;  //Difference between now and time
started
Console.WriteLine("delta is : " + delta.ToString());
if (delta >= 86400) { Console.WriteLine("break" +
Thread.CurrentThread.ManagedThreadId); break; }
            System.Diagnostics.Stopwatch timer = newStopwatch();
string url = "http://83.212.124.24";   // Web Service URL
var get = (HttpWebRequest)WebRequest.Create(url);
            get.Method = "GET";
            get.Proxy = null;
            get.KeepAlive = true;
            timer.Start();
```

```
using (var response = get.GetResponse())
        {
using (var stream = response.GetResponseStream())
             {
using (var reader = newStreamReader(stream))
                  {
Console.WriteLine("Thread" + Thread.CurrentThread.ManagedThreadId);
string html = reader.ReadToEnd();
double Length = response.ContentLength;
                      timer.Stop();
int WebRequestStatus = newint();
TimeSpan timeTaken = timer.Elapsed;
Console.WriteLine(Thread.CurrentThread.ManagedThreadId + " Request took" + timeTaken);
string serverResponse = string.Empty;
if (html.Contains("195.251.68.68")) {   //Check which Web Server Response the Request
                          serverResponse = "195.251.68.68";
                      }
elseif (html.Contains("83.212.115.92")) {
                          serverResponse = "83.212.115.92";
                      }
elseif (html.Contains("83.212.98.71"))
                      {
                          serverResponse = "83.212.98.71";
                      }
                      timer.Reset();

HttpStatusCode statusCode = ((HttpWebResponse)response).StatusCode;
if (statusCode == System.Net.HttpStatusCode.OK)
                      {
                          WebRequestStatus = 200;
Console.WriteLine(Thread.CurrentThread.ManagedThreadId + " Request Status" + ",200");
                      }
elseif (statusCode == System.Net.HttpStatusCode.Forbidden)
                      {
                          WebRequestStatus = 403;
Console.WriteLine(Thread.CurrentThread.ManagedThreadId + " Request Status" + ",403");
                      }
elseif (statusCode == System.Net.HttpStatusCode.NoContent)
                      {
                          WebRequestStatus = 204;
Console.WriteLine(Thread.CurrentThread.ManagedThreadId + " Request Status" + ",204");
                      }
elseif (statusCode == System.Net.HttpStatusCode.NotFound)
                      {
                          WebRequestStatus = 404;
Console.WriteLine(Thread.CurrentThread.ManagedThreadId + " Request Status" + ",404");
                      }
elseif (statusCode == System.Net.HttpStatusCode.ServiceUnavailable)
                      {
                          WebRequestStatus = 503;
Console.WriteLine(Thread.CurrentThread.ManagedThreadId + " Request Status" + ",503");
                      }
elseif (statusCode == System.Net.HttpStatusCode.InternalServerError)
                      {
                          WebRequestStatus = 500;
Console.WriteLine(Thread.CurrentThread.ManagedThreadId + " Request Status" + ",500");
                      }
elseif (statusCode == System.Net.HttpStatusCode.BadGateway)
```

```
                    {
                        WebRequestStatus = 502;
Console.WriteLine(Thread.CurrentThread.ManagedThreadId + " Request Status" + ",502");
                    }
else
                    {
                        WebRequestStatus = 0;
Console.WriteLine(Thread.CurrentThread.ManagedThreadId + " Request Status" + ",0");
                    }



using (MySql.Data.MySqlClient.MySqlConnection conn = new
MySql.Data.MySqlClient.MySqlConnection(myConnectionString))
                    {
double RTime = timeTaken.TotalMilliseconds;   //<----------you can use any format to
get the current time
double throughput = Length / (RTime*1000);
MySqlCommand command = conn.CreateCommand();
                        command.CommandText = "INSERT INTO Stats
(Thread,RequestStatus,ResponseTime,ServerIP,Length,TimeStarted,Throughput) VALUES
(@thread,@status,@responseTime,@ip,@length,@timeStarted,@throughput)";
                        command.Parameters.AddWithValue("?thread",
Thread.CurrentThread.ManagedThreadId);
                        command.Parameters.AddWithValue("?status", WebRequestStatus);
                        command.Parameters.AddWithValue("?responseTime", RTime);
                        command.Parameters.AddWithValue("?ip", serverResponse);
                        command.Parameters.AddWithValue("?length", Length);
                        command.Parameters.AddWithValue("?timeStarted", NowTime);
                        command.Parameters.AddWithValue("?throughput", throughput);
try
                        {
                            conn.Open();
Console.WriteLine("Connected");
                            command.ExecuteNonQuery();
                        }
catch (Exception ex)
                        {
Console.ForegroundColor = ConsoleColor.Red;
Console.WriteLine(ex.ToString());
                        }
finally
                        {
conn.Close();
                        }
                    }

                }
            }

        }

    }

  }
}
```