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

Σχεδιασμός και Υλοποίηση Εφαρμογής lab-on-demand

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Παναγιώτα Ν. Βαλιαντή

Επιβλέπων : Ευστάθιος Συκάς
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2018

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

Σχεδιασμός και Υλοποίηση Εφαρμογής lab-on-demand

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Παναγιώτα Ν. Βαλιαντή

Επιβλέπων : Ευστάθιος Συκάς
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή τη 13η Ιουλίου 2018.

.....	.....	.....
Ευστάθιος Συκάς	Ιωάννα Ρουσσάκη	Γεώργιος Στασινόπουλος
Καθηγητής Ε.Μ.Π.	Επικ. Καθηγήτρια Ε.Μ.Π	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2018

.....
Παναγιώτα Ν. Βαλιαντή

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

Περίληψη

Σκοπός της παρούσας διπλωματικής άσκησης ήταν ο σχεδιασμός και η υλοποίηση μιας εφαρμογής ιστού, η οποία επιτρέπει την κατασκευή αλλά και την παραμετροποίηση μιας εικονικοποιημένης τοπολογίας δικτύου με απλό και γρήγορο τρόπο.

Στα πλαίσια της εργασίας έγινε μελέτη διαφόρων τεχνολογιών εικονικοποίησης ανοιχτού-κώδικα. Συγκεκριμένα μελετήθηκαν τα QEMU/KVM, Xen, libvirt και VirtualBox. Η εργασία αυτή περιγράφει αναλυτικά τις Front-End και Back-End τεχνολογίες ιστού ανοιχτού-κώδικα που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής, αλλά και το λειτουργικό σύστημα Linux, το οποίο απαιτείται για την εγκατάσταση της εφαρμογής.

Επιπλέον, η παρούσα εργασία αναλύει τα σημαντικότερα στάδια ανάπτυξης της εφαρμογής, όπως το στάδιο ανάλυσης, το στάδιο σχεδιασμού και το στάδιο υλοποίησης. Επίσης, δίνονται οδηγίες εγκατάστασης της εφαρμογής σε έναν Linux διακομιστή.

Η εφαρμογή που αναπτύχθηκε μπορεί να φανεί ιδιαίτερα χρήσιμη σε φοιτητές που ενδιαφέρονται να μελετήσουν τη λειτουργία δικτύων υπολογιστών και δικτυακών πρωτοκόλλων. Έκτος από την εκπαίδευση, η εφαρμογή μπορεί να βοηθήσει τη δοκιμή δικτυακών τοπολογιών.

Λέξεις κλειδιά: lab-on-demand, εικονικοποίηση, AJAX, libvirt, KVM, Java, Linux.

Abstract

The purpose of this thesis was the design and implementation of a lab-on-demand web application that would enable the creation and configuration of a virtual network topology in a simple and fast way.

This thesis investigates the most common open-source virtualization technologies available, namely QEMU/KVM, Xen, Libvirt and VirtualBox, in terms of architecture and overall usage. The operating system Linux, as well as the various Front-End and Back-End open-source technologies that were used in the development of the lab-on-demand web application are descripted in this thesis.

Furthermore, this thesis analyses the most important development stages of the lab-on-demand web application, specifically the stages of analysis, design and implementation. A guide for installing the web application on a Linux server is given.

The lab-on-demand web application can be especially useful for students who want to focus on the operation of computer networks and network protocols. However, apart from education, it can be helpful in testing network topologies.

Keywords: lab-on-demand, virtualization, AJAX, libvirt, KVM, Java, Linux.

Ευχαριστίες

Καταρχάς, θα ήθελα να ευχαριστήσω τον καθηγητή κ. Ευστάθιο Συκά, τόσο για την ανάθεση της παρούσας διπλωματικής εργασίας, όσο και για την έμπνευση που μου προσέφερε με τη διδασκαλία του, καθ' όλη τη διάρκεια της φοίτησής μου.

Στη συνέχεια, θα ήθελα να ευχαριστήσω τον υποψήφιο διδάκτορα κ. Βασίλη Ασθενόπουλο, για την αμέριστη υποστήριξη, τη βοήθεια και τον πολύτιμο χρόνο που διέθεσε καθοδηγώντας με σε όλα τα στάδια της εκπόνησης της εργασίας.

Τέλος, οι θερμότερες ευχαριστίες απευθύνονται στην οικογένειά μου αλλά και στους φίλους μου, για την υποστήριξη και συμπαράσταση καθ' όλη τη διάρκεια των σπουδών μου. Ιδιαίτερα θα ήθελα να ευχαριστήσω την αδερφή μου, Χαρά, για την ενθάρρυνση και αισιοδοξία που μου παρείχε. Χωρίς τη συμβολή τους η ολοκλήρωση της διπλωματικής αυτής δεν θα ήταν εφικτή.

Παναγιώτα Βαλιαντή

Αθήνα, Ιούλιος 2018

Αφιερωμένη στη μνήμη της μητέρας μου, Άρτεμις....

Πίνακας Περιεχομένων

Κατάλογος Σχημάτων / Εικόνων

1. Εισαγωγή

1.1 Πρόλογος

Η εικονικοποίηση έχει αλλάξει τον τρόπο που βλέπουμε την πληροφορική. Αντί να χρησιμοποιούμε πολλά διαφορετικά φυσικά μηχανήματα για διάφορες εργασίες, πλέον με την εικονικοποίηση μπορούμε να χρησιμοποιήσουμε ένα φυσικό μηχάνημα το οποίο εκτελεί ταυτόχρονα πολλές εικονικές μηχανές και να έχουμε κεντρική διαχείριση.

Στον τομέα της εκπαίδευσης, η εικονικοποίηση έχει επιλύσει πολλά προβλήματα που σχετίζονται με τη διαχείριση συστημάτων εργαστηρίων για προπτυχιακά μαθήματα πληροφορικής, όπως λειτουργικά συστήματα, δίκτυα υπολογιστών, ασφάλεια δικτύου κλπ. Αυτή η τεχνολογική πρόοδος επέτρεψε στα μαθήματα αυτά να μεταβούν από το περιγραφικό τους περιεχόμενο, στη μάθηση μέσω δραστηριοτήτων που εμπλέκουν τους μαθητές σε πρακτικά, αυθεντικά, προβλήματα. Δεδομένου ότι αυτό το είδος δραστηριότητας απαιτεί οι μαθητές να είναι διαχειριστές των δικών τους εικονικών μηχανών ή ακόμα και εικονικών δικτύων, η αποκτηθείσα εμπειρία είναι εγγενώς αυθεντική.

Στην περίπτωση που η εικονικοποίηση χρησιμοποιείται για τη δημιουργία εικονικοποιημένων δικτυακών συσκευών και εικονικοποιημένων δικτύων, η χρήση της προορίζεται όχι μόνο σε μαθητές, αλλά και σε όποιον θα ήθελε να δοκιμάσει τη λειτουργία τοπολογιών δικτύου χωρίς τη χρηματική επιβάρυνση που θα είχε η κατασκευή της φυσικής τοπολογίας δικτύου.

Ο συνδυασμός των εικονικών εργαστηρίων αυτών με τον Παγκόσμιο Ιστό, θα μπορούσε να εξυπηρετήσει εξ αποστάσεως με ευκολία και απλότητα, ένα μεγάλο μέρος του ανθρώπινου πληθυσμού.

1.2 Αντικείμενο της Διπλωματικής

Η παρούσα διπλωματική εργασία έχει ως αντικείμενο τον σχεδιασμό και υλοποίηση μιας εφαρμογής ιστού, η οποία επιτρέπει την κατασκευή αλλά και την παραμετροποίηση μιας εικονικοποιημένης τοπολογίας δικτύου με απλό και γρήγορο τρόπο.

Μέσω της εφαρμογής αυτής, δίνεται η δυνατότητα στους χρήστες να κατασκευάσουν μια τοπολογία δικτύου επιλέγοντας για κάθε επιμέρους δικτυακή συσκευή μία αρχική εικόνα και ένα σετ δικτυακών ρυθμίσεων που θα εφαρμοστεί στην αρχική εικόνα, ώστε να προσδιοριστεί ο δικτυακός της ρόλος στην τοπολογία. Η εφαρμογή επιτρέπει τη μαζική δημιουργία πανομοιότυπων εικονικών δικτυακών συσκευών αλλά και την παραμετροποίηση των εικονικών τρόπων δικτύωσης. Επίσης, η εφαρμογή παρέχει τη δυνατότητα επικοινωνίας με κάθε εικονική μηχανή.

Η εφαρμογή σχεδιάστηκε ώστε να είναι λειτουργική και φιλική προς τον χρήστη, με την επιλογή των κατάλληλων Front-End και Back-End τεχνολογιών. Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε κυρίως η γλώσσα προγραμματισμού Java στο Back-End και η βιβλιοθήκη jQuery στο Front-End.

1.3 Οργάνωση Κειμένου

Αρχικά, στο κείμενο, ο αναγνώστης αποκτά μια γενική γνώση για τα εργαλεία και τις έννοιες που χρησιμοποιούνται, και στη συνέχεια παρακολουθεί τα στάδια της ανάπτυξης της εφαρμογής και άλλα θέματα που αφορούν την εφαρμογή.

Στο **Κεφάλαιο 2**, παρουσιάζεται το θεωρητικό υπόβαθρο που χρειάζεται να έχει ο αναγνώστης για να κατανοήσει το υπόλοιπο κείμενο. Γίνεται μια μελέτη για τις τεχνολογίες εικονικοποίησης και ξεκαθαρίζεται ο όρος εφαρμογή Παγκόσμιου Ιστού.

Στο **Κεφάλαιο 3**, περιγράφονται οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής και οι λόγοι που οδήγησαν στην επιλογή τους. Χωρίζονται σε Front-End και Back-End.

Στο **Κεφάλαιο 4**, αναλύονται οι απαιτήσεις και οι προδιαγραφές της εφαρμογής, αφού διευκρινιστεί ο σκοπός της.

Στο **Κεφάλαιο 5**, παρουσιάζονται οι σχεδιαστικές επιλογές της εφαρμογής καθώς και η αρχιτεκτονική της εφαρμογής. Χρησιμοποιείται και σχηματική δομή για να διευκολυνθεί η κατανόηση της αρχιτεκτονικής της εφαρμογής.

Στο **Κεφάλαιο 6**, παραθέτουμε σημαντικά σημεία κώδικα με τα οποία υλοποιήθηκε η εφαρμογή καθώς και ένα παράδειγμα χρήσης της εφαρμογής.

Στο **Κεφάλαιο 7**, περιγράφεται η διαδικασία εγκατάστασης της εφαρμογής σε έναν Linux διακομιστή.

Τέλος, στο **Κεφάλαιο 8**, συνοψίζονται σκέψεις από τη διαδικασία υλοποίησης της εφαρμογής και προτείνονται πιθανές μελλοντικές επεκτάσεις.

2. Θεωρητικό Υπόβαθρο

Σε αυτό το κεφάλαιο παρατίθενται και αναλύονται έννοιες στις οποίες βασίστηκε η παρούσα εργασία. Οι παρακάτω έννοιες ανήκουν κυρίως σε δύο κατηγορίες: στην εικονικοποίηση και στις εφαρμογές Παγκόσμιου Ιστού. Επίσης, αυτό το κεφάλαιο θα αναφερθεί σε μερικά από τα πιο δημοφιλή μοντέρνα λογισμικά εικονικοποίησης και συγκεκριμένα στα: KVM, QEMU, Xen, libvirt και VirtualBox. Θα γίνει σύγκριση των τεχνολογιών αυτών και αναλυτική περιγραφή.

2.1 Εικονικοποίηση

Αυτό το υποκεφάλαιο θα εξηγήσει τι είναι η εικονικοποίηση. Θα αναφερθεί σε έννοιες και ορισμούς που αφορούν την τεχνολογία της εικονικοποίησης, θα αναλύσει τη θεωρία πίσω από την εικονικοποίηση καθώς και τι υλικό υπολογιστή απαιτείται για να υποστηρίξει εικονικοποίηση. Έπειτα θα εξετάσει τους διάφορους τύπους εικονικοποίησης και τα πλεονεκτήματα που έχει.

2.1.1 Έννοιες και Ορισμοί

Αρχικά, θα ήθελα να διευκρινίσω τους ακόλουθους όρους: εικονική μηχανή, ελεγκτής εικονικών μηχανών, επόπτης, οικοδεσπότης και επισκέπτης.

- **Εικονική Μηχανή (Virtual Machine, VM)** Η εικονική μηχανή είναι λογισμικό προσομοίωσης ενός φυσικού συστήματος (με πραγματικό/απτό υλικό).

- **Ελεγκτής Εικονικών Μηχανών (Virtual Machine Monitor, VMM)** Ο ελεγκτής εικονικών μηχανών είναι λογισμικό το οποίο είναι υπεύθυνο για τη δημιουργία, διαχείριση και ασφαλή εκτέλεση των εικονικών μηχανών.

- **Επόπτης (Hypervisor)** Είναι το ίδιο με τον VMM.

- **Οικοδεσπότης (Host)** Ο όρος αυτός χρησιμοποιείται για το φυσικό μηχάνημα στο οποίο τρέχουν οι εικονικές μηχανές.

- **Επισκέπτης (Guest)** Ο όρος αυτός χρησιμοποιείται για τις εικονικές μηχανές.

2.1.2 Τι Είναι η Εικονικοποίηση;

Η εικονικοποίηση (virtualization) είναι μια τεχνική λογισμικού που υπάρχει εδώ και μισό αιώνα σχεδόν τώρα, η οποία δίνει τη δυνατότητα εκτέλεσης πολλαπλών εικονικών μηχανών σε ένα φυσικό μηχάνημα. Κάθε εικονική μηχανή έχει την ψευδαίσθηση ότι έχει πλήρη και αποκλειστική πρόσβαση στο υλικό του συστήματος (εικονική CPU (VCPU), εικονική μνήμη, εικονικές συσκευές). Ο VMM εγγυάται την ασφαλή πρόσβαση των εικονικών μηχανών στους φυσικούς πόρους του συστήματος.

Αρχικά αναπτύχθηκε για καλύτερη χρήση του διαθέσιμου υλικού (hardware), το οποίο ήταν συχνά δαπανηρό και είχε αυστηρό χρονοπρογραμματισμό (scheduling). Αυτό σήμαινε ότι οι προγραμματιστές συχνά θα έπρεπε να περιμένουν αρκετές μέρες για να βρουν διαθέσιμο υπολογιστή ώστε να δοκιμάσουν και να εκτελέσουν τα προγράμματά τους, το οποίο συχνά οδηγούσε στη μη βέλτιστη χρήση του υπολογιστή. Επιπλέον, για να επιτρέψει σε πολλούς χρήστες να έχουν το δικό τους τερματικό και ως εκ τούτου να υπάρχουν πολλαπλοί χρήστες ενός μεμονωμένου υπολογιστή.

2.1.3 Χαρακτηριστικά

Η εικονικοποίηση έχει τις ρίζες της στην έννοια της εικονικής μνήμης (virtual memory) και των συστημάτων καταμερισμού χρόνου (time sharing systems). Αρχικά η φυσική μνήμη ήταν δαπανηρή γι’ αυτό ήταν πολύ αναγκαίο να βρεθεί μια λύση που θα επέτρεπε την εκτέλεση ενός προγράμματος που ήταν μεγαλύτερο από τη διαθέσιμη μνήμη. Η λύση ήταν να αναπτυχθεί η εικονική μνήμη και οι τεχνικές σελιδοποίησης (paging) που θα έκαναν εύκολη την παρουσία μεγάλων προγραμμάτων στη μνήμη. Μια άλλη τεχνολογία που βοήθησε την εικονικοποίηση είναι ο καταμερισμός χρόνου (time sharing).

Τρία από τα χαρακτηριστικά του VMM είναι [1]:

- **Ισότητα (Equivalence)** Αυτό το χαρακτηριστικό σημαίνει ότι ο VMM παρέχει ένα περιβάλλον για κάθε πρόγραμμα, το οποίο είναι στην ουσία πανομοιότυπο με την πρωτότυπη μηχανή, δίνοντας την ψευδαίσθηση στα guest VMs ότι εκτελούνται στο εγγενές υλικό.

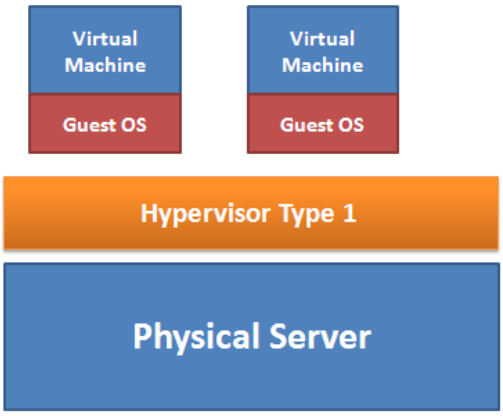
- **Απόδοση (Efficiency)** Προγράμματα που εκτελούνται μέσα σ’ αυτό το περιβάλλον έχουν μικρές μειώσεις στην απόδοσή τους.

- **Έλεγχος Πόρων (Resource Control)** Ο VMM έχει τον πλήρη έλεγχο των πόρων του συστήματος.

2.1.4 Είδη VMMs

Συνήθως ένας VMM χωρίζεται σε Τύπου 1, Τύπου 2 ή Υβριδικό (Hybrid) VMM [2].

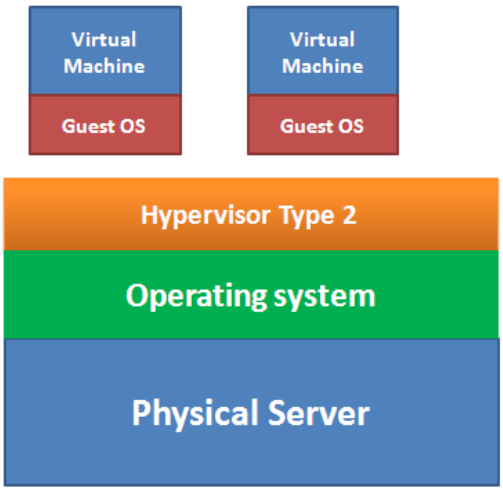
- **VMM Τύπου 1 (Type 1)** Τρέχουν απευθείας στο υλικό του οικοδεσπότη, χωρίς να υπάρχει λειτουργικό σύστημα από κάτω του, το οποίο σημαίνει ότι οι VMMs Τύπου 1 έχουν απευθείας επικοινωνία με το υλικό. Επειδή δεν υπάρχει κάποιο στρώμα που να μεσολαβή ανάμεσα στον VMM και το φυσικό υλικό αυτό αναφέρεται και σαν υλοποίηση "γυμνού υλικού" (bare-metal). Χωρίς λοιπόν να υπάρχει μεσολαβητής, ο VMM Τύπου 1 μπορεί απευθείας να επικοινωνήσει με του πόρους του υλικού στην κατώτερη στοίβα, κάνοντάς τον έτσι πιο αποτελεσματικό από τον VMM Τύπου 2. Εκτός από το ότι διαθέτει καλύτερα χαρακτηριστικά επιδόσεων, ο VMM Τύπου 1, θεωρείτε και ότι διαθέτει μεγαλύτερη ασφάλεια από τον VMM Τύπου 2. Μια εικονική μηχανή μπορεί να βλάψει μόνο την ίδια και το γεγονός αυτό δεν ξεφεύγει από τα σύνορα του περιέκτη (container) εικονικών μηχανών. Οι υπόλοιποι guests συνεχίζουν την επεξεργασία τους και ο VMM παραμένει ανεπηρέαστος. Μπορούν να τρέχουν περισσότερες εικονικές μηχανές σε κάθε host, αφού απαιτείται λιγότερη επιβάρυνση επεξεργασίας για έναν VMM Τύπου 1.



Σχήμα 1: VMM Τύπου 1[1]

[1] https://securitywing.com/types-virtualization-technology/

- **VMM Τύπου 2 (Type 2)** Οι VMMs Τύπου 2 τρέχουν ως μια εφαρμογή μέσα από το λειτουργικό σύστημα του host. Ένα πλεονέκτημα αυτού του μοντέλου είναι ότι μπορεί να υποστηρίξει ένα μεγάλο φάσμα υλικού διότι αυτό κληρονομείτε από το λειτουργικό σύστημα που χρησιμοποιεί. Οι VMMs Τύπου 2 είναι εύκολο να εγκατασταθούν και να αναπτυχθούν επειδή μεγάλο μέρος της διαμόρφωσης υλικού έχει καλυφθεί από το λειτουργικό σύστημα, όπως η δικτύωση και η αποθήκευση. Παρ' όλ' αυτά οι VMMs Τύπου 2 θεωρούνται λιγότερο αξιόπιστοι αφού έχουν περισσότερα σημεία αποτυχίας. Οτιδήποτε επηρεάζει τη διαθεσιμότητα του λειτουργικού μπορεί να έχει αντίκτυπο στον VMM και στους guests που υποστηρίζει. Όλες οι απαιτήσεις του VMM Τύπου 1 πρέπει να καλύπτονται ώστε να υποστηρίζεται και ο VMM Τύπου 2.



Σχήμα 2: VMM Τύπου 2[2]

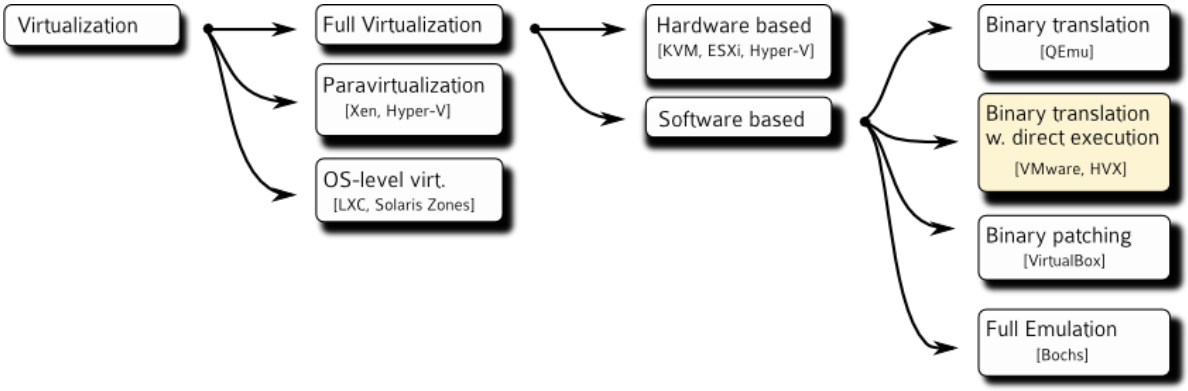
- **Υβριδικό VMM (Hybrid Virtual Machine, HVM)** Υλοποιείται συνήθως όταν ούτε ο Τύπου 1 ούτε ο Τύπου 2 VMM μπορεί να υποστηριχθεί από τον επεξεργαστή. Όλες οι προνομιούχες (privileged) εντολές ερμηνεύονται σε λογισμικό και οι ειδικοί οδηγοί (drivers) πρέπει να είναι γραμμένοι για το λειτουργικό σύστημα που τρέχει ως guest.

Παραδείγματα VMMs Τύπου 1 είναι το Xen, το VMware και οι λύσεις εικονικοποίησης που πρόσφερε η IBM, όπως το z/VM. Παραδείγματα VMMs Τύπου 2 είναι το VMware Workstation, το VirtualBox και το KVM. Παράδειγμα HVM είναι το Xen, το οποίο χρησιμοποιεί οδηγούς παραεικονικοποίησης (paravirtualized drivers).

[2] https://securitywing.com/types-virtualization-technology/

2.1.5 Μέθοδοι Εικονικοποίησης

Αυτό το υποκεφάλαιο θα περιγράψει μερικές από τις μεθόδους εικονικοποίησης: την εξομοίωση, την πλήρης εικονικοποίηση, την παραεικονικοποίηση, την εικονικοποίηση υποβοηθούμενη από το υλικό και την εικονικοποίηση περιβάλλοντος προγραμματισμού.



Σχήμα 3: Κατηγοριοποίηση εικονικοποίησης[3]

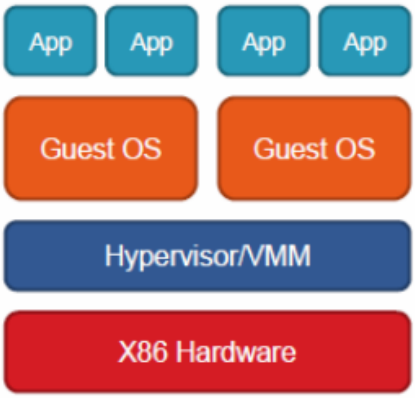
2.1.5.1 Εξομοίωση

Η εξομοίωση (emulation) είναι χρήσιμη όταν το host σύστημα έχει μια αρχιτεκτονική συστήματος και το guest σύστημα μεταγλωττίστηκε για μια διαφορετική αρχιτεκτονική. Σε αυτή την περίπτωση είναι απαραίτητο να μεταφράσουμε όλες τις εντολές της ΚΜΕ προέλευσης, έτσι ώστε να μετατραπούν στις ισοδύναμες εντολές ΚΜΕ στόχου. Η εξομοίωση μπορεί να αυξήσει τη διάρκεια ζωής προγραμμάτων και μας επιτρέπει να εξερευνήσουμε παλιές αρχιτεκτονικές χωρίς να έχουμε μια πραγματική μηχανή. Η κύρια πρόκληση της εξομοίωσης είναι η απόδοση, αφού η εξομοίωση συνόλου εντολών μπορεί να εκτελείται πολύ βραδύτερα από τις εγγενείς εντολές, επειδή μπορεί να χρειάζονται δέκα εντολές στο νέο σύστημα για να διαβαστεί, να αναλυθεί και να προσομοιωθεί μια εντολή από το παλιό σύστημα. Παράδειγμα λογισμικού το οποίο εξομοιώνει έναν τυπικό επεξεργαστή με τα περιφερειακά του είναι ο QEMU, τον οποίο θα δούμε πιο αναλυτικά σε επόμενο υποκεφάλαιο.

[3] https://blogs.oracle.com/ravello/nested-virtualization-with-binary-translation

2.1.5.2 Πλήρης Εικονικοποίηση

Η πλήρης εικονικοποίηση (full virtualization) επιτρέπει στα λειτουργικά συστήματα και στον πυρήνα τους να τρέχουν χωρίς τροποποιήσεις σε μια εικονική μηχανή. Η εικονική μηχανή προσομοιώνει επαρκές τμήμα του πραγματικού υποκείμενου υλικού ώστε να επιτρέπει την εκτέλεση επάνω της ενός μη τροποποιημένου, guest λειτουργικού συστήματος σχεδιασμένου για τον ίδιο τύπο επεξεργαστή με την πραγματική ΚΜΕ (π.χ. VirtualPC, VMware, Win4Lin κλπ). Στην πλήρη εικονικοποίηση δεν χρειάζεται εξομοίωση του συνόλου εντολών του επεξεργαστή και μάλιστα ένα τμήμα του κώδικα του guest λειτουργικού συστήματος μπορεί να εκτελείται απευθείας από το υλικό, χωρίς μεσολάβηση του VMM, αρκεί να μην επηρεάζει υποσυστήματα εκτός του άμεσου ελέγχου του VMM. Τα κρίσιμα σημεία του guest κώδικα ωστόσο, όπως αυτά που προσπαθούν να αποκτήσουν πρόσβαση στο υλικό (π.χ. κλήσεις συστήματος), συλλαμβάνονται από το λογισμικό ελέγχου και προσομοιώνονται, αφού τα αποτελέσματα κάθε λειτουργίας που επιτελείται σε μία εικονική μηχανή δεν επιτρέπεται να τροποποιούν την κατάσταση άλλων εικονικών μηχανών, του VMM ή του υλικού. Αν το πραγματικό υλικό βοηθά και επιταχύνει τη λειτουργία του λογισμικού ελέγχου τότε η πλήρης εικονικοποίηση ονομάζεται εγγενής (native). Η βοήθεια αυτή αφορά κυρίως εύκολη διάκριση μεταξύ εντολών που μπορούν να εκτελεστούν απευθείας και εντολών που πρέπει να προσομοιωθούν από το λογισμικό. Όπως και στην εξομοίωση η εικονική μηχανή παρέχει στο φιλοξενούμενο λειτουργικό σύστημα μία αφαίρεση της μνήμης, των συσκευών Εισόδου / Εξόδου κλπ, ενώ η εγγενής εκτέλεση μεγάλου μέρους του κώδικα παρέχει πολύ καλύτερες επιδόσεις σε σχέση με την εξομοίωση.

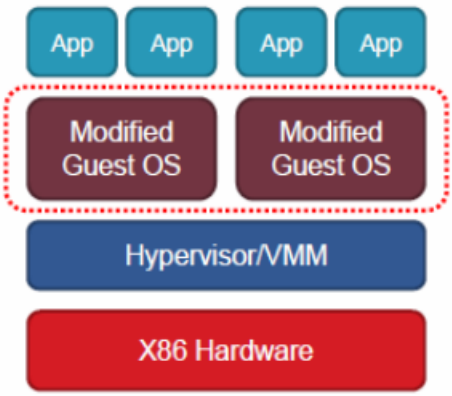


Σχήμα 4: Πλήρης εικονικοποίηση [4]

[4] https://pdfs.semanticscholar.org/presentation/bb54/3a4cb6fbd155f6a7074e1e34aee1633d8201.pdf

2.1.5.3 Παραεικονικοποίηση

Η παραεικονικοποίηση (paravirtualization) είναι μια τεχνική με βάση την οποία το guest λειτουργικό σύστημα τροποποιείται, έτσι ώστε να εργάζεται σε συνεργασία με τον VMM για να βελτιστοποιήσει την απόδοση. Η παραεικονικοποίηση παρουσιάζει στον guest ένα σύστημα το οποίο είναι παρόμοιο αλλά όχι πανομοιότυπο με το προτιμώμενο σύστημα του χρήστη. Αντί να κάνει πλήρη εικονικοποίηση του guest λειτουργικού συστήματος, η παραεικονικοποίηση παρέχει μια διεπαφή λογισμικού (software interface) ή μια Διεπαφή Προγραμματισμού Εφαρμογών (API), η οποία είναι παρόμοια με το υποκείμενο υλικό. Τμήματα του guest λειτουργικού συστήματος γνωρίζουν ότι εκτελούνται σε εικονικό περιβάλλον και ότι δεν έχουν το σύστημα δικό τους. Μέσω της διεπαφής ο guest μπορεί να κάνει άμεσες αιτήσεις, γνωστές ως υπερκλήσεις (hypercalls), στον VMM. Εάν το επίπεδο εικονικοποίησης υποστηρίζει την άμεση επικοινωνία με το υλικό μέσω διαθέσιμων εγκαταστάσεων, για παράδειγμα μέσω εικονικοποίησης υποβοηθούμενη από το υλικό, οι αιτήσεις αυτές μπορούν να φτάσουν απευθείας στο υλικό. Απαιτούνται αλλαγές στον guest πυρήνα ώστε να εκτελείται στο παραεικονικοποιημένο υλικό. Το κέρδος από αυτήν την πρόσθετη εργασία είναι ότι συγκεκριμένες λειτουργίες θα πραγματοποιούνται ταχύτερα, λόγω της αποτελεσματικότερης χρήσης των πόρων και του μικρότερου επίπεδου εικονικοποίησης. Ο κορυφαίος VMM για παραεικονικοποίηση είναι ο Xen.



Σχήμα 5: Παραεικονικοποίηση[5]

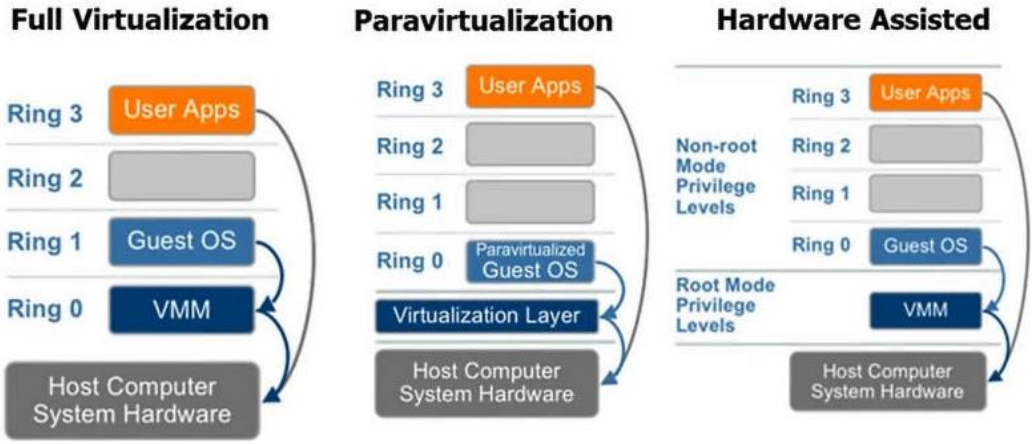
[5] https://pdfs.semanticscholar.org/presentation/bb54/3a4cb6fbd155f6a7074e1e34aee1633d8201.pdf

2.1.5.4 Εικονικοποίηση Υποβοηθούμενη Από το Υλικό

Η εικονικοποίηση με υποβοήθηση υλικού (hardware-assisted virtualization) επιτρέπει την αποτελεσματική πλήρη εικονικοποίηση πλατφόρμας, χρησιμοποιώντας τη βοήθεια από τις δυνατότητες του υλικού, ώστε να διακρίνει τη λειτουργία guest από τη λειτουργία host στον επεξεργαστή. Αυτό καθιστά δυνατή την κατασκευή VMM που χρησιμοποιούν την κλασσική τεχνική trap-and-emulate. Στην πλήρη εικονικοποίηση η εικονική μηχανή εκτελεί σε πλήρη απομόνωση ένα μη τροποποιημένο λειτουργικό σύστημα guest (χρησιμοποιώντας το ίδιο σύνολο εντολών όπως το host μηχάνημα). Η εικονικοποίηση με υποβοήθηση υλικού προστέθηκε σε επεξεργαστές x86 (Intel VT-x ή AMD-V) το 2005 και το 2006 (αντίστοιχα) και προσφέρει σαφώς βελτιωμένη επίδοση, αλλά απαιτεί επεκτάσεις υλικού (virtualization extensions).

2.1.5.5 Εικονικοποίηση Περιβάλλοντος Προγραμματισμού

Η εικονικοποίηση περιβάλλοντος προγραμματισμού (programming-environment virtualization) είναι ένα διαφορετικό μοντέλο εκτέλεσης εικονικοποίησης, με βάση το οποίο οι VMM δεν εικονικοποιούν πραγματικό υλικό, αλλά αντί αυτού δημιουργούν ένα βελτιστοποιημένο εικονικό σύστημα. Εδώ, μια γλώσσα προγραμματισμού σχεδιάζεται για να εκτελείται μέσα σε ένα προσανατολισμένο εικονικοποιημένο περιβάλλον. Για παράδειγμα, η Oracle Java έχει πολλά χαρακτηριστικά τα οποία εξαρτώνται από την εκτέλεσή της μέσα στην εικονική μηχανή της Java (Java Virtual Machine, JVM), που περιλαμβάνουν συγκεκριμένες μεθόδους για ασφάλεια και διαχείριση μνήμης. Επίσης χρησιμοποιείται από το Microsoft.NET.



Σχήμα 6: Σύγκριση της αρχιτεκτονικής διαφορετικών μεθόδων εικονικοποίησης[6]

[6] https://zhihuicao.wordpress.com/2015/06/13/para-virtualization-full-virtualization-differences/

2.1.6 Οφέλη Εικονικοποίησης

Αρκετά χαρακτηριστικά κάνουν την εικονικοποίηση ελκυστική. Τα περισσότερα από αυτά σχετίζονται με τη δυνατότητα διαμοιρασμού του ίδιου υλικού, για την εκτέλεση αρκετών και διαφορετικών περιβαλλόντων εκτέλεσης (δηλ. διαφορετικών λειτουργικών συστημάτων) ταυτόχρονα.

- **Απομόνωση** Παρόλο που οι εικονικές μηχανές μπορούν να μοιραστούν τους φυσικούς πόρους ενός ενιαίου υπολογιστή, παραμένουν εντελώς απομονωμένες η μια από την άλλη σαν να ήταν χωριστές φυσικές μηχανές. Το host σύστημα προστατεύεται από τις εικονικές μηχανές, όπως και οι εικονικές μηχανές προστατεύονται η κάθε μία από τις άλλες. Ένας ιός μέσα σ' ένα guest λειτουργικό σύστημα μπορεί να καταστρέψει αυτό το λειτουργικό σύστημα, αλλά δεν μπορεί να επηρεάσει το host σύστημα ή τους άλλους guests και άρα αυτά θα παραμείνουν διαθέσιμα. Επειδή κάθε εικονική μηχανή είναι σχεδόν πλήρως απομονωμένη από τις άλλες εικονικές μηχανές, δεν υπάρχει σχεδόν κανένα πρόβλημα προστασίας. [3]

- **Μείωση Κόστους και Ενέργειας** Σήμερα πολλές εταιρίες έχουν διάφορους servers που αφιερώνονται για να τρέχουν μόνο ένα λειτουργικό σύστημα είτε ακόμα και για να τρέχουν μόνο μία συγκεκριμένη υπηρεσία. Σε πολλές περιπτώσεις αυτές οι υπηρεσίες τρέχουν σε υλικό το οποίο είναι και δαπανηρό και δύσκολο να συντηρηθεί. Εδώ μπορεί να βοηθήσει η εικονικοποίηση, αντικαθιστώντας τους πολλούς και μικρούς servers με έναν μεγαλύτερο. Αυτός θα τρέχει λογισμικό εικονικοποίησης για να επιτρέπει σε διάφορα λειτουργικά συστήματα και υπηρεσίες να τρέχουν στο ίδιο υλικό. Για παράδειγμα, σε datacenters χρησιμοποιούνται λιγότεροι φυσικοί servers οι οποίοι φιλοξενούν εικονικές μηχανές. Έτσι πετυχαίνουν μικρότερη υποδομή και λιγότερες ανάγκες για ενέργεια λειτουργίας/ψύξης.

- **Ανάπτυξη Εφαρμογών** Η εικονικοποίηση διευκόλυνε την ανάπτυξη αλλά και τη δοκιμή εφαρμογών σε διαφορετικά λειτουργικά συστήματα. Εργαλεία εικονικοποίησης όπως το QEMU και το KVM χρησιμοποιούνται ευρέως από προγραμματιστές Linux κατά τη διάρκεια του κύκλου ανάπτυξης λογισμικού, για δοκιμές και για debugging. Τα εργαλεία αυτά παρέχουν στους προγραμματιστές ένα ευέλικτο περιβάλλον για να εργαστούν. Το περιβάλλον αυτό δημιουργείται και αναδιαμορφώνονται πιο εύκολα από το πραγματικό υλικό. Η αλλαγή μεγέθους μνήμης, για παράδειγμα, είναι πολύ πιο

εύκολη με το QEMU / KVM, απλά αυξάνοντάς την από την κονσόλα. Το ίδιο ισχύει και για άλλα εργαλεία εικονικοποίησης.

- **Εκπαίδευση** Η εικονικοποίηση έχει πολλά πλεονεκτήματα στην εκπαίδευση. Το λογισμικό εικονικοποίησης μπορεί να χρησιμοποιηθεί σε αίθουσες διδασκαλίας και σε εργαστήρια υπολογιστών, παραχωρώντας ολόκληρα εργαστήρια και δαπανηρές υποδομές δοκιμών. Η χρήση της εικονικοποίησης στην εκπαίδευση προσφέρει μεγάλη ευελιξία τόσο για τους σπουδαστές όσο και για τους εκπαιδευτικούς. Σε μαθήματα όπως τα δίκτυα υπολογιστών ή τα λειτουργικά συστήματα, η χρήση εικονικών υποδομών αντί πραγματικών, που απαιτούν μερικές φορές δεκάδες υπολογιστές, θα έκανε εξοικονόμηση στις σχολές και θα διευκόλυνε τη διαμόρφωσή τους από διαχειριστές συστημάτων. Σε μαθήματα ασφάλειας δικτύων, η εικονικοποίηση θα προσέφερε προστασία, μειώνοντας το ρίσκο κάποιος φοιτητής να προκαλέσει ζημιά στους πραγματικούς υπολογιστές.

2.2 Λογισμικό Εικονικοποίησης

Αυτό το υποκεφάλαιο θα επικεντρωθεί στις διάφορες διαθέσιμες ανοιχτού κώδικα (open-source) τεχνολογίες εικονικοποίησης. Συγκεκριμένα, θα αναλύσει τις τεχνολογίες KVM, QEMU, Xen, libvirt και VirtualBox.

2.2.1 QEMU / KVM

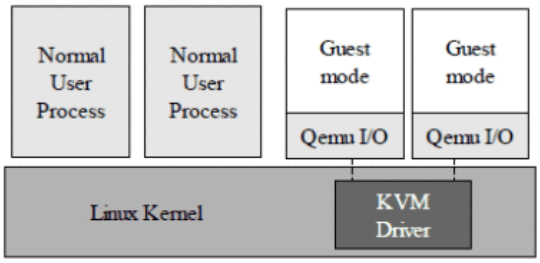
Αρχικά, θα αναφερθούμε στα QEMU και KVM. Θα παρουσιαστεί η βασική αρχιτεκτονική του KVM και έπειτα η βασική αρχιτεκτονική του QEMU. Αφού τα QEMU και KVM είναι στενά συνδεδεμένα, στη συνέχεια θα δούμε τη σουίτα QEMU/KVM.

2.2.1.1 KVM

Το Kernel-based Virtual Machine (KVM) [4] είναι ένα υποσύστημα του Linux πυρήνα το οποίο επιτρέπει την πλήρη εικονικοποίηση σε X86 αρχιτεκτονική επεξεργαστών και εκτίθεται ως συσκευή στο /dev/kvm. Αρχικά εμφανίστηκε το 2006 και ενσωματώθηκε στον πυρήνα του λειτουργικού συστήματος Linux το 2007, στην 2.6.20 έκδοσή του. Έπειτα αναπτύχθηκε από μια εταιρία, γνωστή ως Qumranet, η οποία αργότερα αποκτήθηκε από την RedHat.

Ένας hypervisor αποτελείται από διάφορα συστατικά, συνήθως πρέπει να κάνει χρονοδρομολόγηση (scheduling), να διαχειρίζεται τη μνήμη, να έχει I/O-stack και να έχει οδηγούς για την αρχιτεκτονική που προορίζεται. Αντίθετα με άλλους hypervisors όπως το Xen, το KVM επικεντρώνεται στον χειρισμό της εικονικοποίησης των guest λειτουργικών συστημάτων επιτρέποντας στον πυρήνα του λειτουργικού συστήματος Linux να λειτουργήσει ως ο hypervisor.

Δεδομένου ότι το Linux έχει εξελιχθεί σε ένα ασφαλές και σταθερό λειτουργικό σύστημα, και αφού έχει μερικά από τα πιο σημαντικά χαρακτηριστικά για έναν hypervisor, όπως έναν χρονοδρομολογητή (scheduler) και μια πληθώρα οδηγών (drivers), είναι πιο αποτελεσματική η επαναχρησιμοποίηση και η αξιοποίηση αυτού από την επανεφεύρεση νέου hypervisor.



Σχήμα 7: Η αρχιτεκτονική του KVM[7]

Όπως παρατηρούμε από το Σχήμα 7, το KVM εκτελείται πάνω σε ένα ήδη εγκατεστημένο λειτουργικό σύστημα, δηλαδή είναι ένας τύπου 2 hypervisor.

Το KVM τρέχει ως μια λειτουργική μονάδα στον πυρήνα (kernel module), η οποία χειρίζεται την όλη επικοινωνία μεταξύ των guests και του υλικού. Όλοι οι guests πρέπει να προετοιμαστούν από ένα εργαλείο χώρου χρήστη (user-space tool), το οποίο συνήθως είναι μια έκδοση του QEMU με υποστήριξη KVM. Το KVM χειρίζεται το χαμηλότερο επίπεδο των εικονικών μηχανών, όπως τον έλεγχο της εναλλαγής του guest και του host στο υλικό, των καταχωρητών του επεξεργαστή, κλπ. Το πώς θα χειριστεί η εναλλαγή του guest και του host εξαρτάται από τις επεκτάσεις υλικού (Intel, AMD) που υλοποιούνται στο KVM.

Κάθε guest επεξεργαστής τρέχει σε δικό του νήμα (thread), που προέρχεται από το user-space tool, το οποίο στη συνέχεια χρονοδρομολογείται από τον hypervisor. Κάθε νήμα guest

[7] https://nthadani.wordpress.com/2012/02/02/a-retrospective-analysis-on-the-road-to-red-hat/

διεργασίας και guest επεξεργαστή, χρονοδρομολογείται από τον πυρήνα του Linux, όπως κάθε άλλη διεργασία χρήστη (πχ. φυλλομετρητής ιστού).

Το KVM έχει έναν πρόσθετο τρόπο λειτουργίας της ΚΜΕ, σε σύγκριση με άλλα συμβατικά συστήματα όπως το Xen, που ονομάζεται guest-mode και σε αυτό τον τρόπο λειτουργίας εκτελούνται οι guest εικονικές μηχανές. Οι δύο πρώτοι τρόποι λειτουργίας είναι οι user-mode και kernel-mode.

Σχετικά με τη μνήμη, το use-space tool παρέχει τη μνήμη στους guests και κάνει τις αντιστοιχίσεις της φυσικής μνήμης των guests με την εικονική μνήμη του host. Παραδοσιακά οι αντιστοιχίσεις των εικονικών-φυσικών διευθύνσεων της μνήμης γίνονταν με τη χρήση ειδικών δομών που ονομάζονται shadow page tables, οι οποίες όμως εξομοιώνονται με λογισμικό και κάνουν την έξοδο από το guest-mode να είναι ακριβή. Με την προσθήκη νέων τεχνολογιών (EPT της Intel και RVI της AMD) το υλικό μπορεί σε κάποιο βαθμό να χειριστεί τη μνήμη, ώστε κάθε guest να διατηρεί τα δικά του page tables, ενώ ο hypervisor να χειρίζεται μόνο τα δικά του tables και τις αντιστοιχίσεις από τον hypervisor στον guest.

Άλλες εγκαταστάσεις όπως I/O και αποθήκευση χειρίζονται από τα user-space tools.

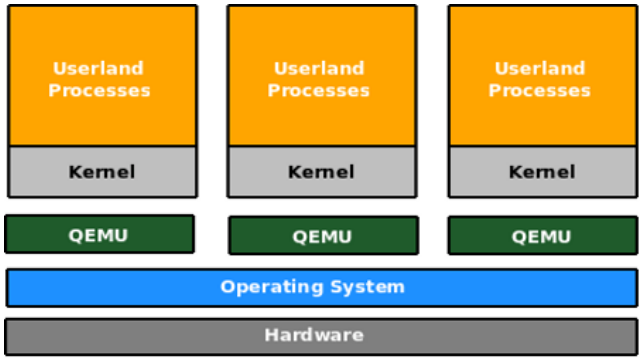
2.2.1.2 QEMU

Ο Quick Emulator (QEMU) [5] είναι ένας εξομοιωτής επεξεργαστή, ο οποίος έχει τη δυνατότητα να εξομοιώνει ένα αριθμό από αρχιτεκτονικές ΚΜΕ (x86, ARM, PowerPC και Sparc), σε διάφορους hosts (x86, PowerPC, ARM, Sparc, Alpha και MIPS). Ο QEMU περιλαμβάνει διάφορα υποσυστήματα, τον εξομοιωτή της ΚΜΕ γνωστός ως Tiny Code Generator (TCG), τις εξομοιωμένες συσκευές όπως οθόνες VGA και κάρτες δικτύου, καθώς και τη διεπαφή του χρήστη και την οθόνη (monitor) του QEMU.

Ο TCG είναι ο εξομοιωτής της ΚΜΕ, ο οποίος είναι υπεύθυνος για την εξομοίωση όλων των guest επεξεργαστών, όταν χρησιμοποιείται εικονικοποίηση υποβοηθούμενη από το υλικό ή άλλος hypervisor, όπως το KVM. Στην ουσία ο TCG είναι ένας δυαδικός μεταφραστής που εκτελεί τη μετάφραση των εντολών των guest για την ΚΜΕ προορισμού. Για την αρχιτεκτονική x86, αυτός ήταν ο μόνος τρόπος εξομοίωσης των guests, μέχρι την εμφάνιση της εικονικοποίησης υποβοηθούμενης από το υλικό και του KVM.

Η αρχιτεκτονική του QEMU είναι παρόμοια με αυτή που είδαμε στο Σχήμα 7. Το KVM χειρίζεται την εικονικοποίηση που βρίσκεται σε χαμηλότερο επίπεδο, ο QEMU κάνει την εξομοίωση και παρουσιάζει τα εξομοιωμένα μηχανήματα στους guests. Επίσης ο QEMU χειρίζεται το υλικό, τις δικτυακές διεπαφές, τις μονάδες αποθήκευσης, τα γραφικά, το I/O και τις θύρες που είναι εξομοιωμένα, το εξομοιωμένο PCI και μερικές λειτουργίες μνήμης για τους guests. Γι' αυτό δεν είναι παράξενη η σύνδεση του QEMU με το KVM. Ο επεξεργαστής του guest (VCPU) τρέχει σε ένα νήμα το οποίο ξεκινά από τον QEMU. Η μνήμη των guests παρέχεται από τον QEMU κατά την εκκίνηση και αντιστοιχείται στον χώρο διευθύνσεων της QEMU διεργασίας. Αυτό λειτουργεί ως η φυσική μνήμη του guest.

Ο QEMU υπάρχει ως εφαρμογή χρήστη και τρέχει ως QEMU διεργασία που χρονοδρομολογείται από το host λειτουργικό σύστημα. Ο QEMU επικοινωνεί με τη μονάδα του KVM μέσω της /dev/kvm διεπαφής.

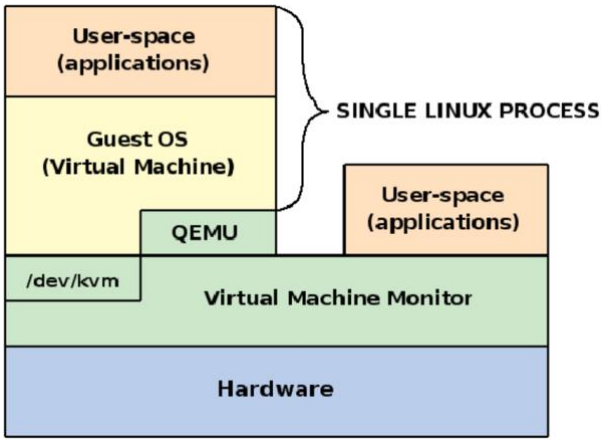


Σχήμα 8: Η αρχιτεκτονική του QEMU[8]

2.2.1.3 QEMU και KVM

Υπάρχει μία σύγχυση όσο αφορά τη χρήση των QEMU και KVM. Ο KVM είναι ο hypervisor, ενώ ο QEMU είναι το user-space tool που αλληλεπιδρά με τον hypervisor. Ο λόγος του μπερδέματος είναι κυρίως επειδή υπάρχουν δύο εκδόσεις διαθέσιμες για τον QEMU, μία για τους προγραμματιστές του QEMU, γνωστή ως Qemu και μια για τους προγραμματιστές του KVM, γνωστή ως qemu-kvm. Στο Σχήμα 9 φαίνεται πιο αναλυτικά η αρχιτεκτονική του KVM με τον QEMU.

[8] https://opensourceforu.com/2009/05/containing-linux-instances-with-openvz/

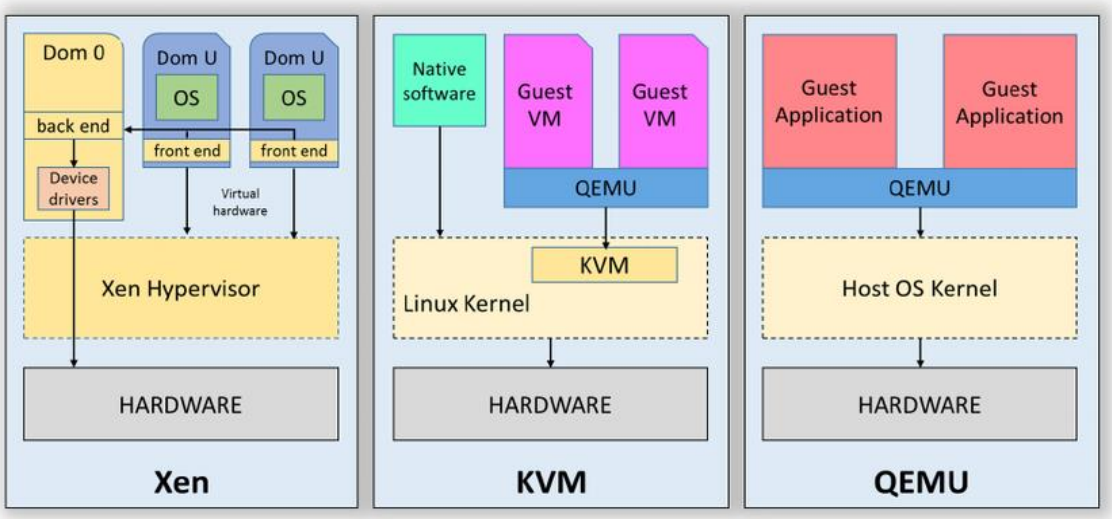


Σχήμα 9: Η αρχιτεκτονική του KVM με QEMU[9]

2.2.2 Xen

Το Xen [6] είναι ένα λογισμικό που παρέχει υπηρεσίες εικονικοποίησης. Ο hypervisor του Xen διαφέρει πολύ από τα KVM και QEMU αρχιτεκτονικά. Το KVM είναι απλά μια μονάδα πυρήνα για το Linux που χρησιμοποιεί το host Linux σύστημα ως hypervisor, ενώ το Xen είναι το ίδιο ο hypervisor.

Το Xen προήλθε από μία επιστημονική εργασία του πανεπιστημίου του Cambridge το 2003 και από τότε έχει υιοθετηθεί ευρέως ως βάση πολλών εμπορικών και ανοιχτού κώδικα εφαρμογών. Επίσης χρησιμοποιείται σήμερα από τα μεγαλύτερα περιβάλλοντα cloud.



Σχήμα 10: Δομή και σύγκριση των Xen, KVM, QEMU[10]

[9] https://www.researchgate.net/figure/Kernel-Virtual-Machine-architecture-Courtesy-of-22_fig8_27516949
[10] https://www.researchgate.net/figure/Comparison-of-Xen-KVM-and-QEMU_fig1_281177318

Όπως παρατηρούμε από το Σχήμα 10, το Xen χρησιμοποιεί έναν hypervisor ο οποίος τρέχει απευθείας πάνω από το υλικό, δηλαδή είναι ένας τύπου 1 hypervisor. Αυτός είναι υπεύθυνος για τη χρονοδρομολόγηση (scheduling) και τη διαμέριση της μνήμης στους guests. Ο hypervisor χειρίζεται όλες τις εκτελέσεις (execution) των guests, εκτός από τα I/O και δεν γνωρίζει τους φυσικούς οδηγούς, ούτε χειρίζεται τις περιφερειακές συσκευές. Όλοι οι I/O πόροι, ο αποθηκευτικός χώρος του δίσκου και άλλοι τυπικοί πόροι του λειτουργικού συστήματος χειρίζονται από μια ειδική εικονική μηχανή, γνωστή ως Domain 0 (Dom0). Ο Domain 0 είναι συνήθως ένας τροποποιημένος πυρήνας Linux ή άλλο UNIX σύστημα (πχ. NetBSD) και είναι απαραίτητος για να μπορεί το Xen να εκτελεί τους guests. Σε αντίθεση με την αρχιτεκτονική του KVM, η οποία προσθέτει μόνο υποστήριξη για εικονικοποίηση υλικού στον πυρήνα, το Xen τρέχει απευθείας στο υλικό χρησιμοποιώντας έναν Linux ή BSD διαμεσολαβητή που χειρίζεται, μεταξύ άλλων, τα I/O και την αποθήκευση.

Οι guest που τρέχουν στον hypervisor του Xen ονομάζονται Domain U (DomU) και αντιμετωπίζονται όπως οι κοινές εικονικές μηχανές. Η επικοινωνία μεταξύ των DomU και του Dom0, ο οποίος είναι συνήθως η προνομιούχα εικονική μηχανή που ονομάζεται Driver Domain, πραγματοποιείται μέσω του μοντέλου των διαχωρισμένων οδηγών (split driver model). Ο Driver Domain περιέχει στο “πίσω” μέρος του split driver (backend driver), ο οποίος επικοινωνεί άμεσα με τον πραγματικό driver, ενώ τα υπόλοιπα εικονικά μηχανήματα περιέχουν στο “μπροστά” μέρος τους split driver (frontend driver) ο οποίος στέλνει αιτήματα στον backend driver. Επειδή υπάρχει περίπτωση πολλές guest εικονικές μηχανές να ζητήσουν ταυτόχρονη πρόσβαση σε μια συσκευή του συστήματος, ο backend driver πρέπει να παρέχει δυνατότητα πολύπλεξης για τα αιτήματα.

Ιστορικά ο hypervisor του Xen χρησιμοποιούσε μόνο την παραεικονικοποίηση για να υποστηρίξει τους guests. Πλέον, μετά την προσθήκη επεκτάσεων υλικού για την υποστήριξη της εικονικοποίησης στους x86 επεξεργαστές, το Xen μπορεί να υποστηρίξει και την πλήρη εικονικοποίηση των guests χωρίς την απαίτηση ειδικών οδηγών. Αντίθετα, στην παραεικονικοποίηση οι guests γνωρίζουν ότι είναι εικονικοποιημένοι και χρειάζονται τη χρήση ειδικών οδηγών.

2.2.3 libvirt

Το libvirt [7] είναι ένα επίπεδο αφαίρεσης και μια εργαλειοθήκη (toolkit) για τη διαχείριση εικονικών μηχανών. Περιλαμβάνει Διεπαφή Προγραμματισμού Εφαρμογών (API) και έναν

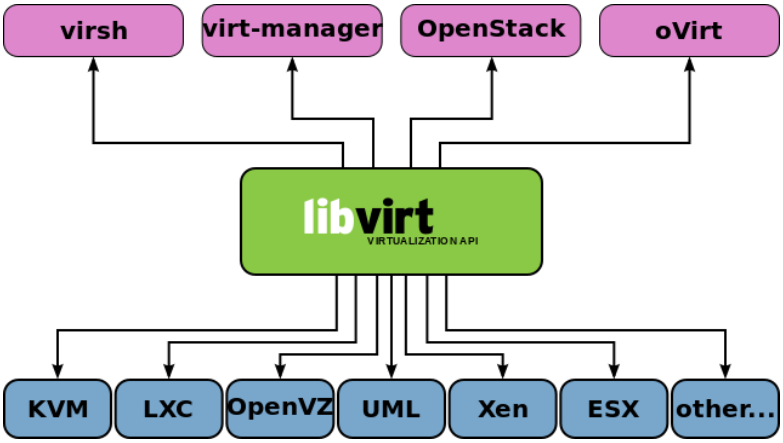
δαίμονα (daemon) γνωστό ως libvirtd, ο οποίος τρέχει στο host μηχάνημα και ακούει αιτήσεις τις οποίες αντιστοιχεί στους κατάλληλους hypervisors (VMMs).

Το libvirt API περιέχει και έχει συνδέσεις (bindings) για τις πιο δημοφιλείς γλώσσες προγραμματισμού όπως C, Python, Perl, Java, Ruby και άλλες. Η πιο χρησιμοποιημένη είναι η Python, με την οποία γράφτηκε και ο Virtual Machine Manager.

Εφαρμογές όπως το virsh και το Virtual Machine Manager, όπως θα δούμε και αργότερα, χρησιμοποιούν το API Libvirt για να επιτρέπουν στους χρήστες την επικοινωνία με τον hypervisor.

2.2.3.1 Βασική Αρχιτεκτονική

Το libvirt δουλεύει χρησιμοποιώντας μια αρχιτεκτονική διακομιστή-πελάτη (server-client) όπου ένας διακομιστής εκτελεί τον δαίμονα libvirt και οι εφαρμογές-πελάτες χρησιμοποιούν το API για να επικοινωνούν με το διακομιστή. Το Σχήμα 11 παρουσιάζει μια απλοποιημένη άποψη της συνολικής αρχιτεκτονικής, με τις εφαρμογές χρηστών που χρησιμοποιούν το libvirt API για να επικοινωνήσουν με τον δαίμονα να είναι στην κορυφή, ο οποίος στη συνέχεια αντιστοιχεί τις αιτήσεις στον κατάλληλο hypervisor (κάτω μέρος σχήματος).

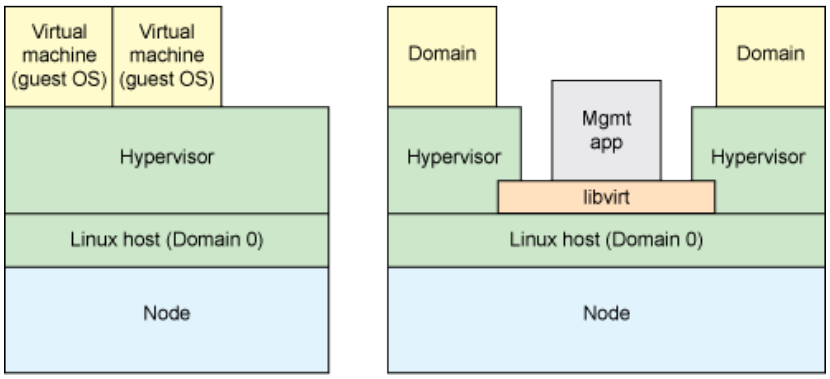


Σχήμα 11: Το libvirt υποστηρίζει διάφορους hypervisors και υποστηρίζεται από διάφορα εργαλεία χρήστη[11]

Ας δούμε πιο αναλυτικά την όψη του μοντέλου που χρησιμοποιεί το libvirt. Το libvirt API έχει σχεδιαστεί για να χρησιμοποιείται από μια εφαρμογή διαχείρισης (management application)

[11] https://commons.wikimedia.org/wiki/File:Libvirt_support.svg

όπως φαίνεται στο Σχήμα 12. Το libvirt, μέσω ενός μηχανισμού που είναι ειδικός για κάθε hypervisor, επικοινωνεί με τον κάθε διαθέσιμο hypervisor για να εκτελέσει τα αιτήματα API.

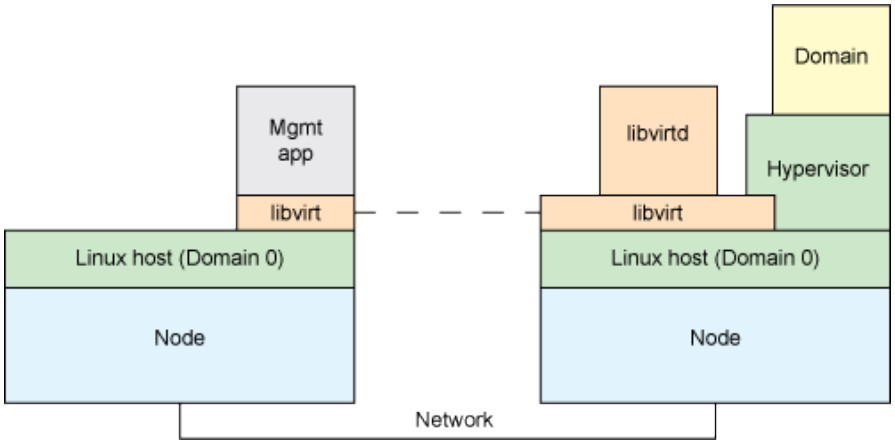


Σχήμα 12: Σύγκριση και χρησιμοποίηση του libvirt μοντέλου[12]

Στο Σχήμα 12 εμφανίζεται και μια σύγκριση της ορολογίας που χρησιμοποιεί το libvirt. Αυτή η ορολογία είναι σημαντική, καθώς αυτοί οι όροι χρησιμοποιούνται από το API. Οι δύο θεμελιώδεις διαφορές είναι ότι το libvirt καλεί τον φυσικό host, κόμβο (node) και το guest λειτουργικό σύστημα, domain. Σημειώστε ότι το libvirt (και η εφαρμογή του) εκτελείται στον domain του host λειτουργικού συστήματος Linux (Domain 0). [8]

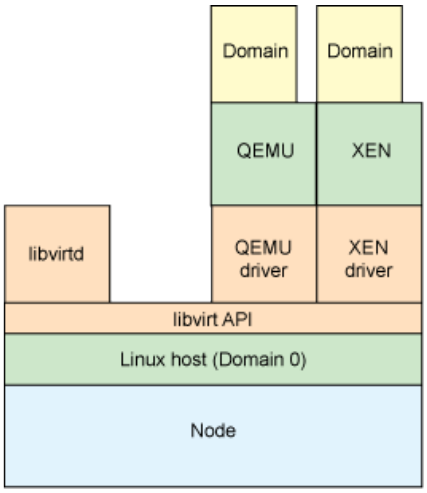
Το libvirt έχει δύο διαφορετικά μέσα ελέγχου. Το πρώτο μέσο παρουσιάζεται στο Σχήμα 12, όπου η εφαρμογή και οι domains βρίσκονται στον ίδιο κόμβο (host). Σε αυτή την περίπτωση, η εφαρμογή διαχείρισης λειτουργεί μέσω του libvirt για να ελέγχει τους τοπικούς domains. Όταν η εφαρμογή διαχείρισης και οι domains βρίσκονται σε διαφορετικούς κόμβους, χρησιμοποιείται το δεύτερο μέσο. Σε αυτή την περίπτωση απαιτείται απομακρυσμένη επικοινωνία μέσω δικτύου όπως φαίνεται στο Σχήμα 13. Αυτή η λειτουργία χρησιμοποιεί τον δαίμονα libvirtd που εκτελείται σε απομακρυσμένους κόμβους. Όταν εγκαθίσταται σε νέο κόμβο το libvirt, ο δαίμονας αυτός ξεκινάει αυτόματα και μπορεί να καθορίσει αυτόματα τους τοπικούς hypervisors και να ρυθμίσει οδηγούς (drivers) για αυτούς. Η εφαρμογή διαχείρισης επικοινωνεί με τον απομακρυσμένο libvirtd μέσω του τοπικού libvirt χρησιμοποιώντας ένα προσαρμοσμένο πρωτόκολλο.

[12] https://www.ibm.com/developerworks/library/l-libvirt/index.html



Σχήμα 13: Έλεγχος απομακρυσμένων hypervisor μέσω libvirtd[13]

Το libvirt υποστηρίζει μια μεγάλη ποικιλία από hypervisors, εφαρμόζοντας μια αρχιτεκτονική που βασίζεται σε οδηγούς (driver-based), η οποία επιτρέπει σε ένα κοινό API να εξυπηρετεί έναν μεγάλο αριθμό hypervisors με κοινό τρόπο. Αυτό σημαίνει ότι ορισμένες εξειδικευμένες λειτουργίες ορισμένων hypervisors δεν είναι ορατές μέσω του API. Επιπλέον, μερικοί hypervisors ενδέχεται να μην υλοποιούν όλες τις λειτουργίες API, οι οποίες στη συνέχεια ορίζονται ως μη υποστηριζόμενες εντός του συγκεκριμένου driver. Το Σχήμα 14 απεικονίζει τα επίπεδα του libvirt API και των σχετικών οδηγών. Σημειώστε επίσης ότι εδώ το libvirtd παρέχει τη δυνατότητα πρόσβασης σε τοπικούς domains από απομακρυσμένες εφαρμογές.



Σχήμα 14: Η driver-based αρχιτεκτονική του libvirt[14]

Για τις τους QEMU και KVM hypervisors, το libvirt χρησιμοποιεί είτε το δυαδικό (binary) qemu-system-x86_64 είτε το qemu-kvm αντίστοιχα για να χειριστεί την εικονικοποίηση και την

[13] https://www.ibm.com/developerworks/library/l-libvirt/index.html
[14] https://www.ibm.com/developerworks/library/l-libvirt/index.html

εξομοίωση. Και τα δύο μπορούν να βρεθούν στον κατάλογο /usr/bin. Για το Xen, το libvirt αναζητά μια τρέχουσα εμφάνιση του Xen δαίμονα (xend) και ελέγχει για εικονικές μηχανές στον κατάλογο /etc/xen. Επιπλέον το libvirt χρησιμοποιεί από προεπιλογή τους οδηγούς Virtio για τους εγκατεστημένους KVM guests.

Στον παρακάτω πίνακα (Σχήμα 15) απεικονίζονται οι hypervisors για τους οποίους το libvirt διαθέτει drivers. Είναι πιθανό να προστεθούν κι άλλοι drivers ώστε να υποστηριχθούν νέοι ανοιχτού κώδικα hypervisors. Οι KVM και Xen hypervisors είναι οι πιο γνωστοί καθώς υποστηρίζονται από τη γραφική διεπαφή Virtual Machine Manager.

Hypervisor	Description
Xen	Hypervisor for IA-32, IA-64, and PowerPC 970 architectures
QEMU	Platform emulator for various architectures
Kernel-based Virtual Machine (KVM)	Linux platform emulator
Linux Containers (LXC)	Linux (lightweight) containers for operating system virtualization
OpenVZ	Operating system-level virtualization based on the Linux kernel
VirtualBox	Hypervisor for x86 virtualization
User Mode Linux	Linux platform emulator for various architectures
Test	Test driver for a fake hypervisor
Storage	Storage pool drivers (local disk, network disk, iSCSI volume)

Σχήμα 15: Οι hypervisors που υποστηρίζει το libvirt[15]

2.2.3.2 Εργαλεία Χρήστη και Χρήση

Αφού αναλύσαμε την αρχιτεκτονική του libvirt, θα καλύψουμε μερικά από τα εργαλεία χρήσης του libvirt API: το virsh, το Virtual Machine Manager, Virt Install, Virt Clone και Virtual Machine Viewer.

[15] https://www.ibm.com/developerworks/library/l-libvirt/index.html

virsh

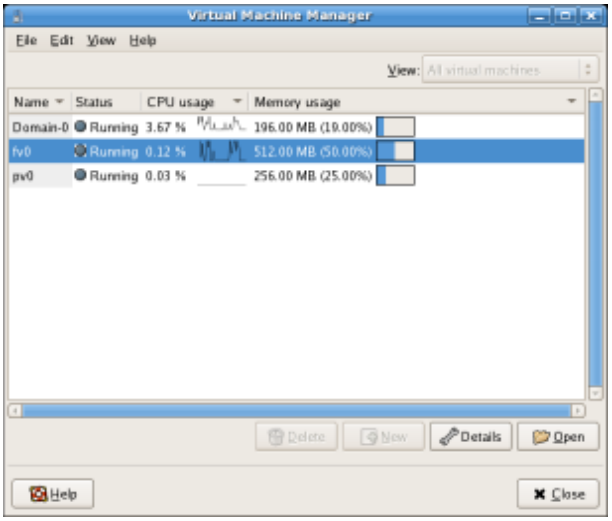
Το virsh είναι ένα εργαλείο γραμμής εντολών (command line tool) που προσφέρεται από το libvirt για τη διαχείριση των εικονικών μηχανών και των hypervisors. Παρακάτω δίνεται ένα παράδειγμα χρήσης.

```
virsh start FreeBSD-import
```

Το οποίο ξεκινά και κάνει boot έναν guest με το όνομα FreeBSD-import.

Virtual Machine Manager

Το Virtual Machine Manager ή virt-manager, είναι ένα γραφικό front-end που επιτρέπει στους χρήστες να αλληλεπιδρούν με τις εικονικές μηχανές καθώς και να εγκαθιστούν εικονικές μηχανές. Το GUI παρουσιάζει στον χρήστη πληροφορίες για τις εικονικές μηχανές που τρέχουν, γραφήματα με τη χρησιμοποίηση της ΚΜΕ, τη χρησιμοποιούμενη μνήμη και τη γενική κατάσταση των εικονικών μηχανών.



Σχήμα 16: Virtual Machine Manager[16]

Virt Install

Το Virt Install ή virt-install, είναι ένα εργαλείο γραμμής εντολών για τη δημιουργία νέων εικονικών μηχανών. Παράδειγμα χρήσης δίνεται παρακάτω.

```
virt-install \
     --connect qemu:///system \
     --virt-type kvm \
     --name demo \
     --ram 500 \
```

[16] https://people.redhat.com/berrange/virt-manager/screenshots.html#hostSummary

```
        --disk path=/var/lib/libvirt/images/demo.img,size=8 \
        --graphics vnc \
        --cdrom /dev/cdrom \
        --os-variant fedora13
```

Virt Clone

Το Virt Clone ή virt-clone είναι ένα εργαλείο γραμμής εντολών για την κλωνοποίηση εικονικών μηχανών που ήδη υπάρχουν και είναι συνδεδεμένες με το libvirt. Η εικονική μηχανή αντιγράφεται ολόκληρη, μαζί με όλες τις ρυθμίσεις υλικού που ταυτίζονται μεταξύ των κλώνων. Τα προβλήματα μοναδικότητας χειρίζονται από τη διαδικασία κλωνοποίησης, όπως οι διευθύνσεις MAC και τα UUIDs. Ακολουθεί παράδειγμα χρήσης.

```
virt-clone \
     --original demo \
     --name newdemo \
     --file /var/lib/xen/images/newdemo.img
```

Virtual Machine Viewer

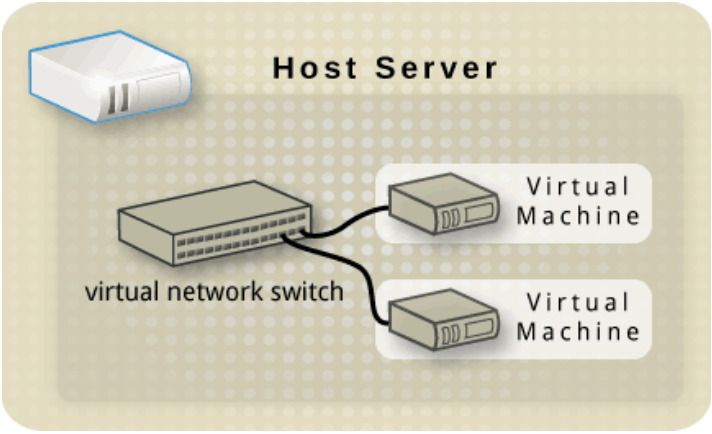
Το Virtual Machine Viewer ή virt-viewer, είναι μία ελαφριά (lightweight) διεπαφή για τη γραφική επικοινωνία με τις εικονικές μηχανές. Οι εικονικές μηχανές είναι προσβάσιμες μέσω του VNC ή του SPICE, και μπορούν να συνδεθούν με τους guests είτε μέσω του libvirt είτε μέσω SSH. Παραδείγματα χρήσης SSH για QEMU και Xen:

```
virt-viewer --connect qemu+ssh://user@host.example/system 'VM name'
virt-viewer --connect xen+ssh://user@host.example/system 'VM name'
```

2.2.3.3 Εικονικοποιημένα Δίκτυα στο libvirt

Αυτό το υποκεφάλαιο θα εξηγήσει πώς τα εικονικοποιημένα δίκτυα που χρησιμοποιούνται από τους guests δουλεύουν. Η υλοποίηση των δικτύων αυτών σε μορφή κώδικα θα αναλυθεί σε επόμενο κεφάλαιο (βλέπε Θέματα Υλοποίησης). Γενικά, η δικτύωση χρησιμοποιώντας το libvirt είναι απλή και εφαρμόζεται με τον ίδιο τρόπο σε όλους τους hypervisors, είτε είναι το KVM, το Xen ή κάποιος άλλος.

Το libvirt χρησιμοποιεί την ιδέα ενός εικονικού μεταγωγέα δικτύου (virtual network switch). Πρόκειται για μια κατασκευή λογισμικού στον host διακομιστή, στην οποία συνδέονται οι guest εικονικές μηχανές και μέσω αυτής κατευθύνουν την κίνηση δικτύου. Στον Linux host το virtual network switch εμφανίζεται ως μια δικτυακή διεπαφή (network interface). Όταν ο libvirt δαίμονας (libvirtd) εγκατασταθεί και ξεκινήσει στον host, περιλαμβάνει μία αρχική δικτυακή διεπαφή με όνομα vibr0.



Σχήμα 17: Η δικτύωση των guests στο libvirt γίνεται με virtual network switches[17]

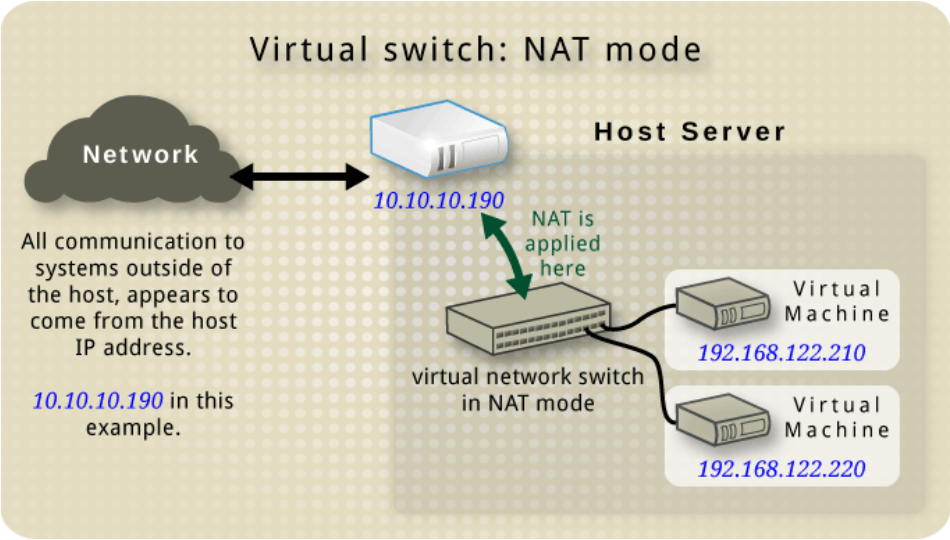
Σε κάθε virtual network switch μπορεί να δοθεί ένα εύρος από IP διευθύνσεις, για να παραχωρηθούν στους guests μέσω DHCP. Για αυτόν το σκοπό, το libvirt χρησιμοποιεί ένα πρόγραμμα που ονομάζεται dnsmasq, το οποίο ρυθμίζεται και ξεκινά αυτόματα από το libvirt για κάθε virtual network switch που το χρειάζεται. Το dnsmasq, εκτός από DHCP server περιλαμβάνει και DNS server.

Παρακάτω θα δοθούν οι βασικοί τρόποι δικτύωσης στο libvirt, αν και προχωρημένοι χρήστες μπορούν να τροποποιήσουν σημαντικά τη λειτουργία του επιπέδου δικτύωσης.

Network Address Translation (NAT)

Το virtual network switch λειτουργεί σε NAT τρόπο δικτύωσης από προεπιλογή (χρησιμοποιώντας IP masquerading). Η λειτουργία αυτή είναι κατάλληλη για απλές περιπτώσεις δικτύωσης όπου ο guest χρειάζεται να κάνει μόνο εξερχόμενες συνδέσεις (τύπου πελάτη). Για παράδειγμα, για έναν υπολογιστή που θέλει να επισκεφθεί μια σελίδα στο διαδίκτυο. Σε αυτή τη λειτουργία οι εγκατεστημένοι guests μπορούν να επικοινωνούν με το εξωτερικό δίκτυο, μέσω της φυσικής μηχανής του host. Αυτό σημαίνει ότι όσοι guests είναι συνδεδεμένοι σε αυτό το virtual network switch, χρησιμοποιούν την IP διεύθυνση του host για να στείλουν κίνηση προς τον έξω κόσμο. Η έναρξη επικοινωνίας από υπολογιστές που είναι έξω από τον host προς τους guests που είναι μέσα δεν είναι εφικτή. Το libvirt χρησιμοποιεί κανόνες iptables για να ρυθμίσει τη λειτουργία NAT.

[17] https://wiki.libvirt.org/page/VirtualNetworking#Advanced

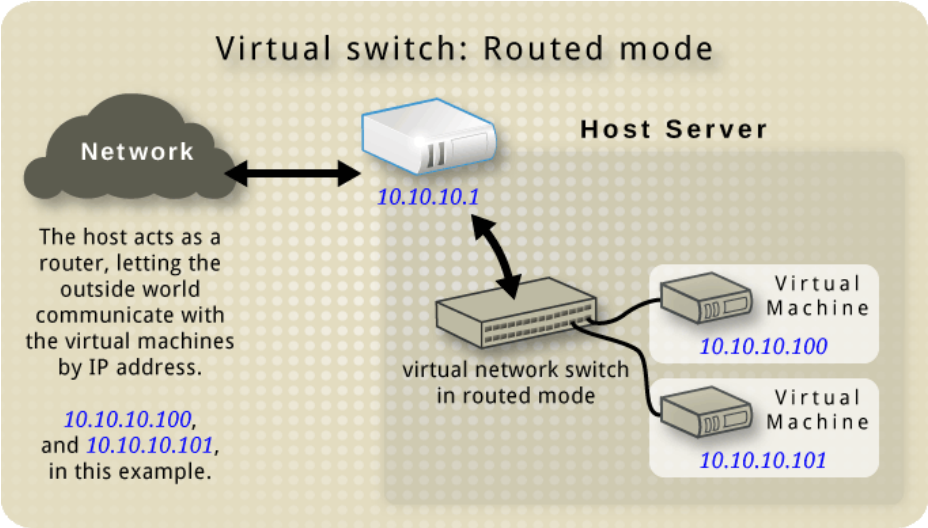


Σχήμα 18: Ο NAT τρόπος δικτύωσης στο libvirt[18]

Λειτουργία δρομολόγησης (Routed mode)

Στον routed τρόπο δικτύωσης, το virtual network switch συνδέεται στο φυσικό LAN που είναι συνδεδεμένο το φυσικό μηχάνημα του host, περνώντας guest κίνηση εμπρός και πίσω χωρίς τη χρήση του NAT. Το virtual network switch μπορεί να εξετάσει την όλη κίνηση και να χρησιμοποιήσει την πληροφορία που βρίσκεται μέσα στα πακέτα δικτύου (IP διεύθυνση) για να πάρει αποφάσεις δρομολόγησης. Σε αυτό τον τρόπο δικτύωσης, όλοι οι guests βρίσκονται σε ένα δικό τους υποδίκτυο που δρομολογείται μέσω του virtual network switch. Αυτή η κατάσταση δεν είναι πάντα ιδανική, καθώς καμία άλλη φυσική μηχανή στο φυσικό δίκτυο (εκτός από τον host) γνωρίζει αυτό το υποδίκτυο και έτσι δεν είναι δυνατή η πρόσβαση στα guest εικονικά μηχανήματα. Είναι οπόταν απαραίτητη η ρύθμιση διαδρομών στους δρομολογητές του φυσικού δικτύου (πχ. χρησιμοποιώντας στατική δρομολόγηση). Ο τρόπος δρομολόγησης λειτουργεί στο Layer 3 του μοντέλου δικτύωσης OSI.

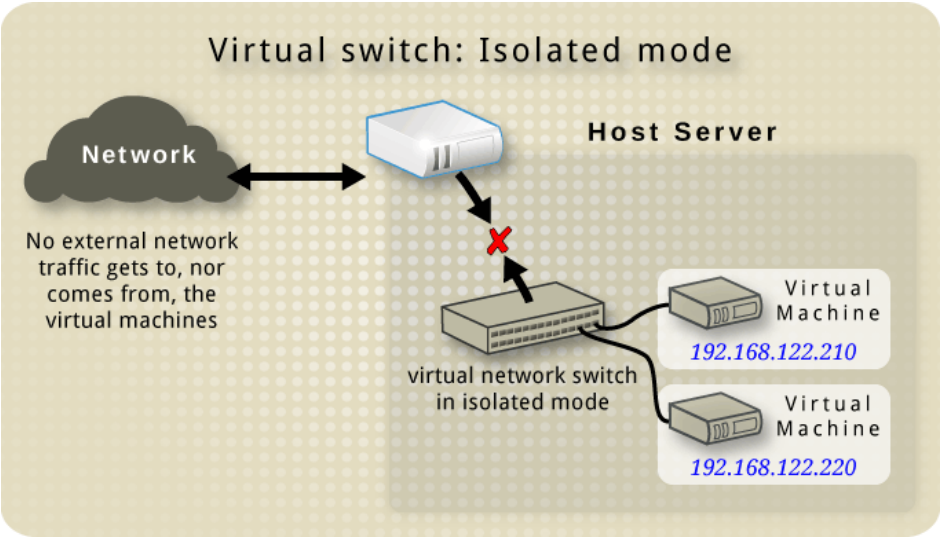
[18] https://wiki.libvirt.org/page/VirtualNetworking#Advanced



Σχήμα 19: Ο routed τρόπος δικτύωσης στο libvirt[19]

Λειτουργία απομόνωσης (Isolated mode)

Όταν χρησιμοποιείται αυτός ο τρόπος δικτύωσης, οι guests που είναι συνδεδεμένοι με το virtual network switch μπορούν να επικοινωνούν μεταξύ τους και με το φυσικό μηχάνημα του host. Παρ' όλ' αυτά, η κίνησή τους δεν θα περάσει έξω από το φυσικό μηχάνημα του host, ούτε μπορεί να εισέλθει κίνηση που προέρχεται έξω από τον host. Είναι πιθανό να χρησιμοποιείται το πρόγραμμα dnsmasq σε αυτή τη λειτουργία, για τη χρήση του DHCP server. Ωστόσο, ακόμη κι αν αυτό το δίκτυο είναι απομονωμένο από οποιοδήποτε φυσικό δίκτυο, τα ονόματα DNS εξακολουθούν να επιλύονται. Επομένως, μπορεί να προκύψει μια κατάσταση όπου τα ονόματα DNS επιλύονται, αλλά τα ICMP echo requests (ping) των guests αποτυγχάνουν.

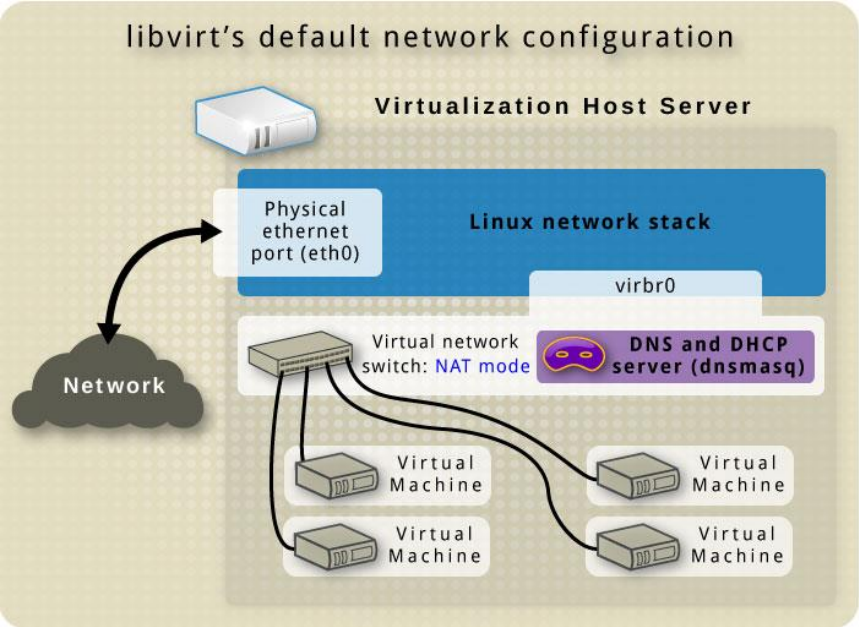


Σχήμα 20: Ο isolated τρόπος δικτύωσης στο libvirt[20]

[19] https://wiki.libvirt.org/page/VirtualNetworking#Advanced
[20] https://wiki.libvirt.org/page/VirtualNetworking#Advanced

Η προεπιλεγμένη ρύθμιση (default configuration) - NAT Network

Όταν ο libvirt δαίμονας (libvirtd) εγκατασταθεί και ξεκινήσει στον host, περιλαμβάνει ένα αρχικό virtual network switch που βρίσκεται σε NAT λειτουργία. Αυτή η λειτουργία επιτρέπει στους εγκατεστημένους guests να επικοινωνήσουν με εξωτερικά δίκτυα, ανεξάρτητα από το εάν ο host χρησιμοποιεί ethernet, wireless, dialup ή VPN δικτίωση. Εάν ο host δεν έχει δικτύωση, τουλάχιστον επιτρέπεται στους guests να επικοινωνούν απευθείας μεταξύ τους. Όπως παρατηρούμε και στο Σχήμα 21, το πρόγραμμα dnsmasq περιέχεται στην προεπιλεγμένη δικτύωση.



Σχήμα 21: Ο προεπιλεγμένος τρόπος δικτύωσης στο libvirt[20]

2.2.4 VirtualBox

Το VirtualBox [9][41] είναι μια εφαρμογή εικονικοποίησης, η οποία μπορεί να εγκατασταθεί ως εφαρμογή από τον χρήστη. Το VirtualBox διαφέρει από τα υπόλοιπα λογισμικά εικονικοποίησης που αναφέραμε αφού κυρίως στοχεύει σε desktop χρήστες. Μία άλλη διαφορά είναι ότι είναι διαθέσιμη και για τα λειτουργικά συστήματα Windows και Mac OS X εκτός από το Linux.

Αρχικά είχε αναπτυχθεί από την Innotek GmbH, μια γερμανική εταιρία, η οποία στη συνέχεια εξαγοράστηκε από την Sun Microsystems. Σήμερα, το VirtualBox είναι μέρος των προϊόντων της Oracle καθώς από το 2010, η Oracle εξαγόρασε την Sun Microsystems. Έτσι το επίσημο όνομα του προϊόντος είναι το Oracle VM Virtualbox.

Το VirtualBox επιτρέπει μόνο την πλήρη εικονικοποίηση των guests που τρέχει, όμως μπορούν να εικονικοποιηθούν είτε μέσω εικονικοποίησης λογισμικού ή είτε μέσω εικονικοποίησης υλικού. Οι guests που τρέχουν χρησιμοποιώντας την εικονικοποίηση υλικού, χρησιμοποιούν τις προσθήκες της εικονικοποίησης υποβοηθούμενης από το υλικό στον επεξεργαστή, ώστε να πετύχουν την πλήρη εικονικοποίηση.

Αρχιτεκτονικά το VirtualBox χρησιμοποιεί μια μονάδα πυρήνα που λειτουργεί ως ο hypervisor και από πάνω του υπάρχει το VirtualBox API που επικοινωνεί με τον hypervisor. Ο hypervisor του VirtualBox είναι τύπου 2. Από το API μπορούν να «χτιστούν» οι εφαρμογές του χρήστη, η πλούσια GUI διεπαφή και το εργαλείο γραμμής εντολών VBoxManage.

Το VirtualBox μπορεί να χρησιμοποιηθεί είτε από τη γραμμή εντολών, με το άνοιγμα ενός τερματικού, είτε μέσω του γραφικού περιβάλλοντος. Μια φορητή έκδοση που δεν χρειάζεται εγκατάσταση είναι επίσης διαθέσιμη με το όνομα Portable-VirtualBox.

Σχήμα 22: Το γραφικό περιβάλλον του VirtualBox σε Mac OS X[21]

[21] http://www.oracle.com/us/corporate/press/1842885

2.3 Εφαρμογή Παγκόσμιου Ιστού (Web Application)

Η εφαρμογή παγκόσμιου ιστού (web application ή web app) [10] είναι ένα πρόγραμμα υπολογιστή που έχει αρχιτεκτονική πελάτη-διακομιστή (client-server), όπου ο πελάτης, μαζί με τη διεπαφή του χρήστη, τρέχει σε έναν φυλλομετρητή ιστού (web browser). Οι πιο γνωστές εφαρμογές ιστού είναι το webmail, τα ηλεκτρονικά καταστήματα, τα wikis, οι ηλεκτρονικές πλατφόρμες διεξαγωγής δημοπρασιών και τα προγράμματα ανταλλαγής άμεσων μηνυμάτων.

Ο Παγκόσμιος Ιστός (World Wide Web, www, ή Web) απαρτίζεται από ένα σύνολο πρωτοκόλλων βασιζόμενων στο Διαδίκτυο (Internet), τα οποία επιτρέπουν τη δημιουργία και ανάγνωση εγγράφων πολυμέσων απ' οποιονδήποτε συνδεδεμένο υπολογιστή στον κόσμο. Ο Παγκόσμιος Ιστός υποστηρίζει υπερκείμενο, γραφικά, ήχους και βίντεο, ενώ για τη δόμηση και περιγραφή της πληροφορίας χρησιμοποιεί τη γλώσσα Hypertext Markup Language (HTML). Τα έγγραφα μορφής HTML διερμηνεύονται από μια ειδική κατηγορία προγραμμάτων, τις αποκαλούμενες εφαρμογές περιήγησης ή φυλλομετρητές (browser). Υπάρχουν πολλές εφαρμογές browser, οι πιο γνωστές και χρησιμοποιημένες σήμερα είναι ο Google Chrome, ο Mozilla Firefox, ο Internet Explorer και ο Safari.

Υπάρχει μια γενική σύγχυση στη διάκριση μιας δυναμικής ιστοσελίδας από μια εφαρμογή ιστού. Συνήθως χρησιμοποιούμε τον όρο «εφαρμογή ιστού» για να αναφερθούμε σε ιστοσελίδες που έχουν παρόμοια λειτουργικότητα με ένα λογισμικό εφαρμογής επιφάνειας εργασίας (desktop software application) ή με μια εφαρμογή κινητού (mobile app).

Οι τεχνολογίες που χρησιμοποιούνται συνήθως σε εφαρμογές ιστού και που χρησιμοποιήθηκαν για την εφαρμογή lab-on-demand που αναπτύξαμε, τόσο για το κομμάτι του πελάτη όσο και του διακομιστή, θα αναλυθούν στο επόμενο κεφάλαιο (βλέπε Τεχνολογίες / Εργαλεία).

3. Τεχνολογίες / Εργαλεία

Σε αυτό το κεφάλαιο αναλύονται τα κύρια χαρακτηριστικά των τεχνολογιών και εργαλείων που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής καθώς και οι λόγοι που οδήγησαν στην επιλογή τους. Εδώ χρησιμοποιούμε τον όρο «τεχνολογίες» πιο διευρυμένα, αφού αναφερόμαστε και σε γλώσσες προγραμματισμού, σε βοηθητικές βιβλιοθήκες/εφαρμογές, σε προγραμματιστικά εργαλεία αλλά και σε λειτουργικά συστήματα.

Η χρήση τους θα αναλυθεί περεταίρω σε επόμενο κεφάλαιο (βλέπε Θέματα Υλοποίησης).

3.1 Γενικά

Εφόσον η εφαρμογή μας είναι προσβάσιμη μέσω του Παγκόσμιου Ιστού, χρησιμοποιεί το πρωτόκολλο μεταφοράς υπερκειμένου (HTTP) [11] το οποίο αποτελεί τη θεμελιώδης δομή για επικοινωνία δεδομένων στον Παγκόσμιο Ιστό (World Wide Web).

Επίσης χρησιμοποιεί την τεχνολογία JavaScript Object Notation (JSON) [12], για την ασύγχρονη επικοινωνία μεταξύ του φυλλομετρητή ιστού και του διακομιστή ιστού. Είναι το πλέον διαδεδομένο πρότυπο κωδικοποίησης δεδομένων. Σε αυτή τη μορφή αναπαριστά τα δεδομένα των αντικειμένων της Java το Spring Framework. Τα δεδομένα γράφονται σε HTTP απαντήσεις ως JSON.

Είναι εκτός του σκοπού, όμως, αυτής της διπλωματικής να αναφερθούμε αναλυτικά για τα παραπάνω.

3.2 Front-End

Στο Front-End κομμάτι, δηλαδή στη διεπαφή ιστού που αλληλεπιδρά μαζί της ο πελάτης (client-side web interface) χρησιμοποιήθηκε η Hypertext Markup Language (HTML) [13] μαζί με Cascading Style Sheets (CSS) και JavaScript [14].

Η γλώσσα HTML είναι η κύρια γλώσσα σήμανσης για αναπαράσταση πληροφορίας σε ιστοσελίδες και εφαρμογές ιστού, τα στυλ μορφοποίησης CSS ορίζουν την εμφάνιση και τη

διάταξη της πληροφορίας και η JavaScript προσφέρει ασύγχρονη ανταλλαγή δεδομένων και δυναμική αλλαγή περιεχομένου του HTML εγγράφου.

Επιπλέον, χρησιμοποιήθηκαν η βιβλιοθήκη jQuery και το πλαίσιο (framework) Bootstrap για να δημιουργήσουμε δυναμικές ιστοσελίδες που είναι λειτουργικές και χρηστικές με ομοιόμορφο τρόπο (σε όλες τις συσκευές) και να πετύχουμε μια διεπαφή φιλική με τον χρήστη (user-friendly interface).

3.2.1 jQuery

Η jQuery [15] είναι μια γρήγορη, μικρή σε μέγεθος και πλούσια σε χαρακτηριστικά βιβλιοθήκη JavaScript. Ο σκοπός της είναι να διευκολύνει και να απλοποιήσει τη χρήση της της γλώσσας JavaScript, η οποία χρησιμοποιείται στο client-side της δημιουργίας δυναμικών διαδικτυακών εφαρμογών και ιστοσελίδων. Είναι λογισμικό ανοιχτού κώδικα και διατίθεται δωρεάν κάτω από τους όρους της MIT License [16]. Επίσης, προσφέρει συμβατότητα με τους περισσότερους φυλλομετρητές ιστού.

Κυκλοφόρησε τον Ιανουάριο του 2006 στο BarCamp NYC από τον John Resig και επηρεάστηκε από την προηγούμενη βιβλιοθήκη cssQuery του Dean Edwards. Η χρήση της βιβλιοθήκης είναι απλή καθώς το μόνο που χρειάζεται είναι το κατέβασμα της βιβλιοθήκης από την επίσημη ιστοσελίδα [15] και να συμπεριληφθεί στην επικεφαλίδα του αντίστοιχου αρχείου HTML.

Η σύνταξη της βιβλιοθήκης jQuery έχει σχεδιαστεί για να διευκολύνει τη διάσχιση ενός εγγράφου. Μερικά από τα σημαντικότερα χαρακτηριστικά που προσφέρει είναι η επιλογή στοιχείων Document Object Model (DOM), μέσω Selectors, βάσει διάφορων κριτηρίων όπως το όνομα της ετικέτας (tag), το όνομα της κλάσης (class) και το αναγνωριστικό (id) καθώς και ο χειρισμός στοιχείων DOM, αρχείων HTML και CSS αρχείων. Για παράδειγμα μέσω της πρόσθεσης ή αφαίρεσης στοιχείων και της τροποποίησης του HTML περιεχομένου ή της CSS κλάσης. Επίσης προσφέρει ειδικά εφέ που μπορούν να εφαρμοστούν στα DOM στοιχεία, όπως να εμφανιστούν ή αποκρυφθούν στοιχεία, να μειωθεί η ορατότητά τους, και εφέ κίνησης (animation).

Ένα άλλο χαρακτηριστικό είναι ο χειρισμός γεγονότων (events) που είναι ισοδύναμα με DOM events, όπως το κλικ του ποντικιού, το focus out από ένα πεδίο φόρμας και το πάτημα ενός πλήκτρου από το πληκτρολόγιο. Επίσης, καθιστά απλή την ανάπτυξη εφαρμογών AJAX (συντομογραφία για "Ασύγχρονη JavaScript και XML") [17], συμπεριλαμβάνοντας απλοποιημένες συναρτήσεις για ασύγχρονες κλήσεις AJAX. Με την AJAX, οι διαδικτυακές εφαρμογές μπορούν να στείλουν και να ανακτήσουν δεδομένα από τους διακομιστές (servers) ασύγχρονα (αποστολή HTTP αιτήματος σε ξεχωριστό νήμα και παροχή callback hook για το αποτέλεσμα), χωρίς να παρεμβαίνουν στην εμφάνιση και λειτουργία της υπάρχουσας σελίδας. Έτσι είναι δυνατή η ανάγνωση δεδομένων από τον διακομιστή ιστού (web server) μετά τη φόρτωση της σελίδας, η ενημέρωση της ιστοσελίδας χωρίς την ανάγκη για επαναφόρτωση ολόκληρης της ιστοσελίδας αφού μόνο ένα μικρό τμήμα της σελίδας αλλάζει, όπως και η αποστολή δεδομένων στον διακομιστή ιστού στο παρασκήνιο. Οι ιστοσελίδες που χρησιμοποιούν AJAX είναι ταχύτατες και πολύ πιο εξυπηρετικές για τον χρήστη.

Υπάρχει μεγάλη ποικιλία από plug-ins που δημιουργούνται από προγραμματιστές πάνω από τη βιβλιοθήκη JavaScript, τα οποία μπορούν να ενσωματωθούν και να χρησιμοποιηθούν αυτοτελή ώστε να μειωθεί ο χρόνος δημιουργίας του επιθυμητού εφέ.

Στην εφαρμογή μας χρησιμοποιήθηκαν τα περισσότερα από τα χαρακτηριστικά της jQuery. Η επιλογή στοιχείων DOM, ο χειρισμός τους και τα ειδικά εφέ χρησιμοποιήθηκαν κυρίως στο κομμάτι για τη δημιουργία μιας νέας εικονικοποιημένης τοπολογίας δικτύου, όπου ο χρήστης καλείται να προσδιορίσει τα χαρακτηριστικά του κάθε εικονικού μηχανήματος που δημιουργεί. Ο χειρισμός των events χρησιμοποιήθηκε κυρίως για την καταγραφή του κλικ του ποντικιού, όταν ο χρήστης θέλει να διαγράψει κάποιο εικονικό μηχάνημα ή για να αφαιρέσει κάποιο πεδίο από τα χαρακτηριστικά του domain στη διαδικασία δημιουργίας νέας τοπολογίας.

Η AJAX χρησιμοποιήθηκε σε πολλά σημεία της εφαρμογής καθώς τα δεδομένα αλλάζουν διαρκώς και απαιτείται η συνεχής ανανέωσή τους. Κάθε κάποια δευτερόλεπτα, η λίστα με τις διαθέσιμες εικονικές μηχανές καθώς και η λίστα με τα διαθέσιμα δίκτυα ενημερώνεται αποστέλλοντας ασύγχρονα HTTP GET αιτήματα στον διακομιστή ιστού. Σε κάθε δημιουργία νέας τοπολογίας αποστέλλονται HTTP POST αιτήματα για τη δημιουργία κλώνων και τη δημιουργία νέων εικονικών δικτύων. Αποστέλλονται ασύγχρονα HTTP PUT αιτήματα για αλλαγή της κατάστασης των εικονικών μηχανών και δικτύων, όπως για την εκκίνησή τους, την απενεργοποίησή τους και τη διαγραφή τους.

3.2.2 Bootstrap

Το Bootstrap [18] είναι front-end πλαίσιο (framework) ανοιχτού κώδικα για τη δημιουργία και σχεδιασμό ιστοσελίδων και διαδικτυακών εφαρμογών. Παρέχει μια συλλογή εργαλείων για ανάπτυξη που βασίζονται σε HTML, CSS και JavaScript και υποστηρίζει όλους τους γνωστούς φυλλομετρητές ιστού.

Αναπτύχθηκε από τον Mark Otto και τον Jacob Thornton για το μέσο κοινωνικής δικτύωσης, Twitter [18], ως ένα framework για την εξασφάλιση μιας ενιαίας αισθητικής στις διάφορες λειτουργίες του. Κυκλοφόρησε τον Αύγουστο του 2011 ως λογισμικό ανοιχτού κώδικα και τον Φεβρουάριο του 2011 έγινε το πιο δημοφιλές έργο ανάπτυξης στο GitHub [20]. Διατίθεται δωρεάν κάτω από τους όρους της MIT License [16].

Για να χρησιμοποιήσει κάποιος το Bootstrap αρκεί να κατεβάσει το στυλ CSS από την επίσημη ιστοσελίδα [18] και να το συμπεριλάβει στην επικεφαλίδα του αντίστοιχου αρχείου HTML. Εάν θέλει να χρησιμοποιήσει και τις ενσωματωμένες λειτουργικότητες που προσφέρει η JavaScript, τότε θα πρέπει να κατεβάσει και τη βιβλιοθήκη jQuery [15] και να τη συμπεριλάβει μαζί με το Bootstrap στην επικεφαλίδα του αντίστοιχου HTML αρχείου.

Το Bootstrap περιέχει προκατασκευασμένα στυλ και design templates που χρησιμοποιούνται με ενιαίο τρόπο για έτοιμη μορφοποίηση καθώς και ενσωματωμένες λειτουργικότητες, όπως collapse και carousel, μέσω προαιρετικών JavaScript επεκτάσεων. Περιέχει επίσης έτοιμα επαναχρησιμοποιήσιμα συστατικά, εκτός από τα βασικά HTML στοιχεία, όπως φόρμες, κουμπιά, μενού πλοήγησης, προειδοποιητικά μηνύματα, προηγμένες τυπογραφικές δυνατότητες και άλλα συστατικά διεπαφής. Έτσι ο προγραμματιστής δεν χρειάζεται να δαπανά χρόνο σε αυτά τα συνήθη κομμάτια κώδικα και μπορεί πολύ εύκολα να επεκτείνει ή να κάνει μικρές αλλαγές ώστε να παράγει το επιθυμητό οπτικό αποτέλεσμα.

Ένα από τα κύρια πλεονεκτήματα του Bootstrap είναι ο ανταποκρίσιμος σχεδιασμός (responsive design), ο οποίος καθιστά τις ιστοσελίδες λειτουργικές και χρηστικές με ομοιόμορφο τρόπο σε Desktop, Tablet και mobile συσκευές, αφού η διάταξη των ιστοσελίδων προσαρμόζεται δυναμικά ανάλογα με τα χαρακτηριστικά της συσκευής που χρησιμοποιείται.

3.3 Back-End

Το Back-End κομμάτι, δηλαδή ο κώδικας που υπάρχει στην πλευρά του διακομιστή, υλοποιήθηκε κυρίως με τη χρήση της γλώσσας προγραμματισμού Java. Παρακάτω αναλύουμε τις «τεχνολογίες» που χρησιμοποιήθηκαν σε αυτό το κομμάτι.

3.3.1 Linux

Το Linux είναι ένα σύγχρονο λειτουργικό σύστημα που βασίζεται στις βασικές σχεδιαστικές αρχές και την παράδοση του UNIX [32]. Αρχικά αναπτύχθηκε από τον Linus Torvalds το 1991 για τον επεξεργαστή της Intel 80386 αλλά έχει μεταφερθεί σήμερα σε πολλές διαφορετικές αρχιτεκτονικές (PowerPC, IA-64, MIPS, SPARC, Alpha, κ.α.). [21]

Αυτό που κάνει το Linux ελκυστικό είναι το γεγονός ότι δεν είναι ένα εμπορικό λειτουργικό σύστημα. Είναι ανοιχτού κώδικα και δωρεάν κάτω από τους όρους της GNU General Public Licence (GPL) [22]. Ο κώδικάς του είναι διαθέσιμος για όποιον θέλει να τον μελετήσει και μπορεί να τον κατεβάσει από την επίσημη ιστοσελίδα [23], είτε να τον βρει σε Linux CD. Γι' αυτό τον λόγο, το Linux χρησιμοποιείται για εκπαιδευτικούς, αλλά και για ερευνητικούς σκοπούς.

Τα κύρια συστατικά του συστήματος Linux είναι ο πυρήνας (kernel), οι βιβλιοθήκες συστήματος (system libraries) και τα βοηθήματα συστήματος (system utilities). Ο πυρήνας είναι υπεύθυνος για τη διατήρηση όλων των σημαντικών αφαιρέσεων του λειτουργικού συστήματος, που περιλαμβάνουν την εικονική μνήμη και τις διεργασίες. Οι βιβλιοθήκες του συστήματος ορίζουν ένα σύνολο συναρτήσεων, μέσω των οποίων οι εφαρμογές μπορούν να αλληλεπιδρούν με τον πυρήνα. Αυτές οι συναρτήσεις υλοποιούν το μεγαλύτερο μέρος του λειτουργικού συστήματος, που δεν χρειάζεται τα πλήρη προνόμια του κώδικα του πυρήνα. Η σημαντικότερη βιβλιοθήκη συστήματος είναι η βιβλιοθήκη C, γνωστή ως libc. Τα βοηθήματα του συστήματος είναι προγράμματα είναι προγράμματα τα οποία εκτελούν διακριτές και εξειδικευμένες εργασίες διαχείρισης. Μερικά βοηθήματα του συστήματος καλούνται μόνο μια φορά, για να εκκινήσουν και να διαμορφώσουν κάποιο χαρακτηριστικό του συστήματος. Άλλα, τα οποία είναι γνωστά στην ορολογία του UNIX ως δαίμονες (daemons), εκτελούνται μόνιμα και χειρίζονται εργασίες, όπως απόκριση σε εισερχόμενες συνδέσεις δικτύου και αποδοχή αιτημάτων σύνδεσης από τερματικά.

Όλος ο κώδικας του πυρήνα εκτελείται σε προνομιούχο τρόπο λειτουργίας του επεξεργαστή, γνωστός ως τρόπος λειτουργίας πυρήνα (kernel mode), ώστε να έχει πλήρη πρόσβαση σε όλους τους φυσικούς πόρους του υπολογιστή. Κάθε κώδικας υποστήριξης του λειτουργικού συστήματος, που δεν χρειάζεται να εκτελείται σε τρόπο λειτουργίας πυρήνα, τοποθετείται σε βιβλιοθήκες συστήματος και εκτελείται σε τρόπο λειτουργίας χρήστη (user mode). Ο τρόπος λειτουργίας χρήστη, σε αντίθεση με τον τρόπο λειτουργίας πυρήνα, έχει πρόσβαση μόνο σ' ένα ελεγχόμενο υποσύνολο πόρων του συστήματος.

Προκειμένου ο χρήστης να έχει άμεση πρόσβαση στις λειτουργίες του πυρήνα, ώστε μια διεργασία χρήστη να μπορέσει να εκτελέσει μια λειτουργία για την οποία δεν έχει τα κατάλληλα δικαιώματα (πχ πρόσβαση σε κάποια συσκευή), υπάρχει ένας μηχανισμός που λέγεται κλήση συστήματος (system call). Η κλήση συστήματος πετυχαίνει τη δυνατότητα επικοινωνίας μεταξύ χρήστη και πυρήνα, λειτουργώντας ως εξής: κατά την εκτέλεση μιας διεργασίας, η διεργασία ετοιμάζει και υποβάλλει κατάλληλη αίτηση στο λειτουργικό σύστημα. Τότε, η ΚΜΕ ξεκινά να εκτελεί κώδικα πυρήνα, ο οποίος ελέγχει την ορθότητα των δεδομένων εισόδου και αν η διεργασία έχει τα ανάλογα δικαιώματα, πραγματοποιεί τη ζητούμενη λειτουργία και επιστρέφει τα αποτελέσματά της πίσω στον χώρο χρήστη. Ο έλεγχος επανέρχεται τότε στον κώδικα της διεργασίας.

Το πακετάρισμα του Linux πυρήνα, καθώς και των βοηθητικών εφαρμογών και των προγραμμάτων που τον περιβάλλουν συνθέτουν αυτό που ονομάζουμε διανομή. Υπάρχουν πάρα πολλές διανομές Linux. Μερικές από τις πιο δημοφιλείς είναι το Ubuntu, το Fedora, το CentOS και το Red Hat Enterprise.

3.3.2 Java

Η Java [24][25] είναι μια αντικειμενοστραφής, ανεξάρτητη από την πλατφόρμα, ασφαλής γλώσσα προγραμματισμού, η οποία σχεδιάστηκε έτσι ώστε η εκμάθησή της να είναι ευκολότερη από την C++ [26], ενώ ταυτόχρονα καθιστά δυσκολότερα τα σφάλματα κατά τη χρήση της συγκριτικά με τις C και C++.

Η Java δημιουργήθηκε αρχικά από τον James Gosling για την Sun Microsystems (η οποία ανήκει στην Oracle Corporation πλέον) και πρωτοπαρουσιάστηκε το 1995.

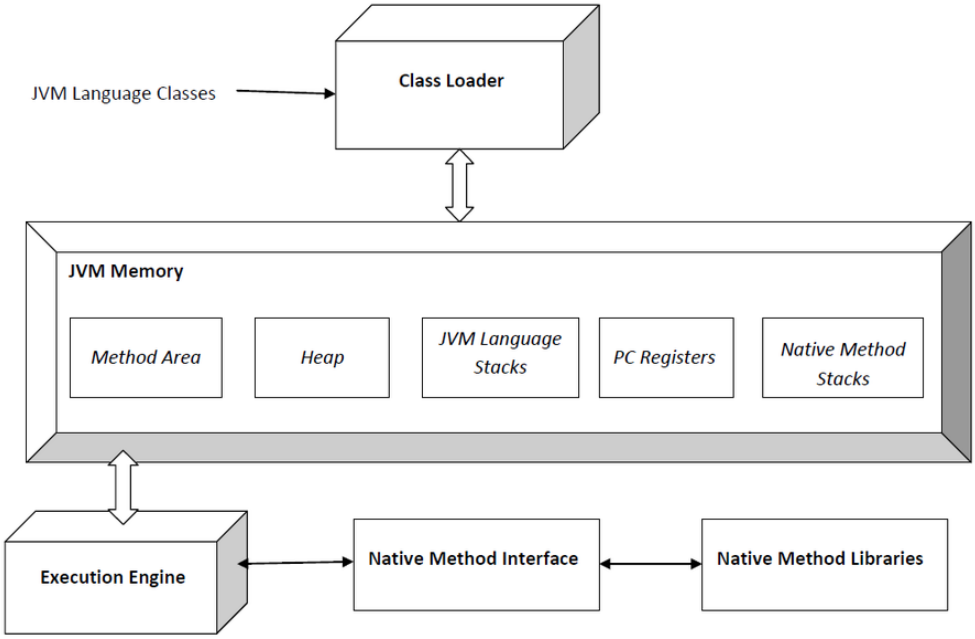
Αν και αρχικά χρησιμοποιούταν για τη δημιουργία απλών προγραμμάτων και ιστοσελίδων, σήμερα μπορεί κανείς να βρει την Java σε πολλούς τομείς, όπως web servers, σχεσιακές βάσεις δεδομένων, τηλεσκόπια που βρίσκονται σε τροχιά προσωπικούς ψηφιακούς βοηθούς (PDA), κινητά τηλέφωνα.

Παρόλο που η Java παραμένει σήμερα χρήσιμη στους δημιουργούς εφαρμογών ιστού (web), η χρήση της εκτείνεται πλέον πολύ πέρα και έξω από το web. Η Java σήμερα είναι μία δημοφιλής γλώσσα προγραμματισμού γενικού σκοπού.

Ο αντικειμενοστραφής προγραμματισμός (Object-Oriented Programming, OOP) είναι μία μεθοδολογία ανάπτυξης λογισμικού η οποία μιμείται τον τρόπο με τον οποίο «συναρμολογούνται» αντικείμενα στον φυσικό κόσμο. Ένα πρόγραμμα θεωρείται σαν μια ομάδα αντικειμένων τα οποία δουλεύουν μαζί. Τα αντικείμενα (objects) δημιουργούνται χρησιμοποιώντας πρότυπα τα οποία αποκαλούνται κλάσεις (classes), και περιέχουν δεδομένα και τις εντολές που απαιτούνται για τη χρήση αυτών των δεδομένων.

Η ανεξαρτησία από την πλατφόρμα είναι η δυνατότητα ενός προγράμματος να εκτελείται χωρίς να απαιτούνται τροποποιήσεις του για διαφορετικά περιβάλλοντα υπολογιστών (λειτουργικά συστήματα). Αφού τα προγράμματα που γράφονται σε Java μεταγλωττιστούν σε μορφή bytecode (αρχεία .class), μπορούν να εκτελεστούν σε οποιοδήποτε λειτουργικό σύστημα, αρκεί η πλατφόρμα να διαθέτει έναν διερμηνευτή Java.

Η εικονική μηχανή της Java (Java Virtual Machine, JVM) διαχειρίζεται αυτόματα τη δέσμευση και αποδέσμευση της μνήμης (garbage collection). Υποστηρίζει πολλαπλά νήματα (threads), κάνει just-in-time μεταγλώττιση και είναι ένα γρήγορο, στιβαρό, αξιόπιστο και διεθνώς διαδεδομένο περιβάλλον εκτέλεσης.



Σχήμα 23: Η δομή του JVM[22]

Η αρχιτεκτονική του JVM έχει την εξής δομή:

- Method Area: αποθήκευση των κλάσεων και των σχετικών τους πληροφοριών (μέθοδοι, πεδία, υπερ-κλάσεις, κτλ.).
- Heap Area: αποθήκευση όλων των αντικειμένων / στιγμιοτύπων.
- Language Stacks: κάθε thread έχει τη δική του στοίβα, όπου αποθηκεύονται τοπικές μεταβλητές, ενδιάμεσα αποτελέσματα, κτλ. σε μορφή stack frames.
- PC Registers: Program Counter Register που περιέχει τη διεύθυνση του υπό εκτέλεση bytecode instruction.
- Native Method Stacks: αποθηκεύονται όλες οι native μέθοδοι της εφαρμογής.

Τα βασικά στοιχεία ασφάλειας της Java είναι η έλλειψη δεικτών (pointers) και η αυτόματη διαχείριση της μνήμης, αφού έτσι παρεμποδίζεται η εσφαλμένη χρήση τους.

3.3.3 Spring Framework

Το Spring [27][28] είναι ένα ανοιχτού κώδικα πλαίσιο (framework) για εφαρμογές Java. Αναπτύχθηκε για να διευκολύνει την ανάπτυξη επαγγελματικών (enterprise) εφαρμογών και να

[22] https://commons.wikimedia.org/w/index.php?curid=35963523

αντιμετωπίσει την πολυπλοκότητά τους. Η πρώτη του έκδοση κυκλοφόρησε το 2003 από τον Rod Johnson και είναι διαθέσιμο κάτω από τους όρους της Apache License 2.0.

Το Spring μας βοηθάει να αναπτύξουμε Java (stand-alone, Web, JEE) εφαρμογές και απλοποιεί τον κώδικα, κάνει ευκολότερο και πιο αποτελεσματικό τον έλεγχο και τη χαλαρή διασύνδεση (loose coupling) κάθε Java εφαρμογής.

Είναι ένα ελαφρύ (lightweight) πλαίσιο ανάπτυξης λογισμικού. Με τον όρο αυτό αναφερόμαστε κυρίως στον επιπλέον φόρτο (overhead) εργασίας που απαιτείται ο οποίος είναι αμελητέος. Ο προγραμματιστής χρειάζεται να κάνει ελάχιστες αλλαγές στον κώδικα της εφαρμογής που αναπτύσσει, έτσι ώστε να χρησιμοποιήσει το συγκεκριμένο πλαίσιο ανάπτυξης λογισμικού. Επίσης αναφερόμαστε στο μέγεθος, αφού η διανομή του Spring πλαισίου μπορεί να συμπεριληφθεί σε ένα απλό jar αρχείο 2,5 MB.

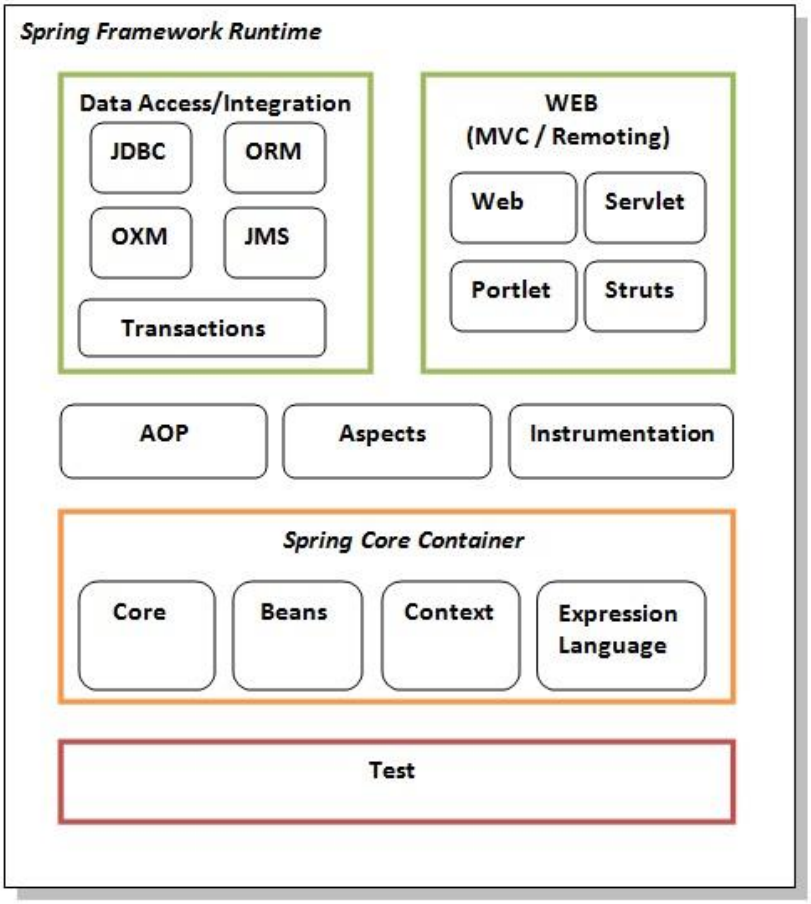
Το Spring προάγει τη χαλαρή διασύνδεση χρησιμοποιώντας μια τεχνική που είναι γνωστή ως Dependency Injection (DI). Όταν εφαρμόζεται η DI, τα αντικείμενα λαμβάνουν παθητικά τις εξαρτήσεις τους (dependencies) αντί να τις δημιουργούν ή να τις αναζητούν μόνα τους. Είναι κάτι σαν ένα αντίστροφο “Java Naming and Directory Interface” (JN-DI), αντί ένα αντικείμενο να ψάχνει μόνο του για τις εξαρτήσεις του σε έναν φορέα, ο ίδιος ο φορέας δίνει τις εξαρτήσεις στο αντικείμενο χωρίς να περιμένει πρώτα να ερωτηθεί. Το DI υλοποιείται στο Spring μέσω των JavaBeans και των interfaces.

Υποστηρίζει προσανατολισμένο προγραμματισμό, το οποίο έχει σαν αποτέλεσμα τη δημιουργία εφαρμογών με περισσότερη συνοχή ακόμη και αν είναι απαραίτητη η συνύπαρξη διαφορετικών λογικών λειτουργίας. Κάθε αντικείμενο της εφαρμογής κάνει μόνο ό,τι το αφορά και δεν είναι υπεύθυνο για λειτουργίες που έχουν να κάνουν με άλλα συστήματα.

Το Spring παρέχει ένα πλαίσιο το οποίο δίνει τη δυνατότητα να δημιουργηθούν πολύπλοκες εφαρμογές με το συνδυασμό άλλων, λιγότερο πολύπλοκων, κομματιών. Στο Spring τα αντικείμενα δημιουργούνται σε απλά XML αρχεία. Επιπλέον παρέχει πολλές δομικές λειτουργίες, επιτρέποντας στον προγραμματιστή να ασχοληθεί περισσότερο με τη λογική της εφαρμογής του.

Το περιβάλλον εργασίας του Spring αποτελείται από αρκετές ενότητες (modules), οι οποίες σαν σύνολο δίνουν στον προγραμματιστή ό,τι χρειάζεται, για να αναπτύξει Java εφαρμογές. Δεν

απαιτείται από το προγραμματιστή να βασίσει όλη την εφαρμογή του πάνω στο Spring. Αντίθετα του παρέχεται η δυνατότητα να επιλέξει όποιες από τις ενότητες πιστεύει ότι καλύπτουν τις ανάγκες του. Επιπλέον παρέχονται τρόποι σύνδεσης με άλλα πλαίσια και βιβλιοθήκες, έτσι ώστε ο προγραμματιστής να μην χρειαστεί να το αναπτύξει από την αρχή.



Σχήμα 24: Οι ενότητες του Spring Framework[23]

Όλες οι ενότητες του Spring είναι χτισμένες πάνω από τον βασικό της πυρήνα (container), όπως φαίνεται στο Σχήμα 24. Ο πυρήνας καθορίζει το πώς δημιουργούνται τα beans, πώς αρχικοποιούνται και πώς γίνεται ο χειρισμός τους — κάτι που αποτελεί τις κυριότερες λειτουργίες της Spring. Οι προγραμματιστές όμως ενδιαφέρονται περισσότερο για άλλες ενότητες, αυτές που χρησιμοποιούν τον πυρήνα και τις υπηρεσίες που αυτός παρέχει.

[23] https://www.javatpoint.com/spring-modules

3.3.3.1 Spring Boot

Το Spring Boot σχεδιάστηκε για να επιταχύνει και να απλοποιήσει τη δημιουργία εφαρμογών που βασίζονται στο Spring τις οποίες μπορείς «απλά να τρέξεις». Απαιτείται ελάχιστη αρχική ρύθμιση (configuration) του Spring.

Τα βασικά χαρακτηριστικά του Spring Boot είναι:

- Δημιουργία αυτόνομων (stand-alone) εφαρμογών Spring.
- Απευθείας ενσωμάτωση του Apache Tomcat, του Jetty, ή του Undertow (δεν χρειάζεται να γίνουν deploy WAR αρχεία).
- Παρέχει "starter" παραδείγματα για τον ορισμό εξαρτήσεων ώστε να απλοποιηθεί το build configuration (πχ. pom.xml παραδείγματα για το Maven).
- Αυτόματη ρύθμιση του Spring και τρίτου μέρους βιβλιοθηκών όταν είναι δυνατόν.
- Παρέχει λειτουργίες έτοιμες για παραγωγή, όπως μετρήσεις, έλεγχοι υγείας και εξωτερική ρύθμιση.
- Δεν απαιτείται καμία δημιουργία κώδικα και δεν χρειάζεται ρύθμιση αρχείων XML (πχ. web.xml).

Στην εφαρμογή χρησιμοποιήσαμε τον Apache Tomcat ως web server της εφαρμογής, ενσωματώνοντάς τον στην εφαρμογή μέσω του Spring Boot.

3.3.4 libvirt API

Το libvirt [7] είναι μια ανοιχτού κώδικα (open-source) εργαλειοθήκη για τη διαχείριση πλατφόρμων εικονικοποίησης, όπως είδαμε και σε προηγούμενο υποκεφάλαιο (βλέπε υποκεφάλαιο 2.2.3). Περιλαμβάνει Διεπαφή Προγραμματισμού Εφαρμογών (API), δαίμονα (daemon) και διαχειριστικά εργαλεία. Μπορεί να χρησιμοποιηθεί για τη διαχείριση των KVM, QEMU, Xen και άλλων τεχνολογιών εικονικοποίησης.

Το libvirt API είναι μια C βιβλιοθήκη που έχει συνδέσεις (bindings) για τις πιο δημοφιλείς γλώσσες προγραμματισμού όπως C, Python, Perl, Java, Ruby και άλλες. Η πιο χρησιμοποιημένη είναι η Python.

Στην εφαρμογή μας είχαμε πρόσβαση στο libvirt API μέσω Java. Χρησιμοποιήσαμε το Java binding του libvirt για να αντλούμε πληροφορίες για τις εικονικές μηχανές και τα εικονικά δίκτυα (κατάσταση, όνομα, uuid, λίστα με ενεργές/ανενεργές εικονικές μηχανές/δίκτυα, κτλ) αλλά και για να διαχειριζόμαστε τις εικονικές μηχανές (εκκίνηση, απενεργοποίηση, διαγραφή, κλπ) και τα εικονικά δίκτυα (δημιουργία, διαγραφή, σύνδεση διεπαφής κλπ).

Παρ' όλ' αυτά, το libvirt API binding για Java δεν υποστήριζε τη δημιουργία κλώνων, ούτε μπορούσε να δώσει κάποιες πληροφορίες για τις εικονικές μηχανές (πχ. για τη δικτύωση). Αυτό το πετύχαμε με τα command line εργαλεία του libvirt “virt-clone” και “virsh” αντίστοιχα, σε συνδιασμό με το ExpectIt API το οποίο θα αναλυθεί στην επόμενη παράγραφο.

3.3.5 ExpectIt API

Το Expect [29][30] είναι εργαλείο το οποίο σχεδιάστηκε για τον έλεγχο διαδραστικών εφαρμογών που εμφανίζουν κείμενο σε διεπαφή τερματικού (text terminal interface), όπως το telnet, ftp, passwd, fsck, rlogin, tip, κλπ.. Οι εφαρμογές αυτές παρακινούν και περιμένουν τον χρήστη να πληκτρολογήσει μια απάντηση. Ένας προγραμματιστής Expect μπορεί να γράψει κώδικα που περιγράφει ένα διάλογο, για να αυτοματοποιήσει τη διαδραστικότητα των εφαρμογών αυτών.

Το Expect γράφτηκε το 1990 από τον Don Libes ως επέκταση της γλώσσας προγραμματισμού σεναρίου (scripting language) Tcl [31]. Αρχικά ήταν γραμμένο μόνο για Unix συστήματα [32], όμως μετά έγινε διαθέσιμο και για άλλα λειτουργικά συστήματα όπως τα Microsoft Windows [33].

```
# Assume $remote_server, $my_user_id, $my_password, and $my_command were
read in earlier
# in the script.
# Open a telnet session to a remote server, and wait for a username
prompt.
spawn telnet $remote_server
expect "username:"
# Send the username, and then wait for a password prompt.
send "$my_user_id\r"
expect "password:"
# Send the password, and then wait for a shell prompt.
send "$my_password\r"
expect "%"
# Send the prebuilt command, and then wait for another shell prompt.
send "$my_command\r"
expect "%"
```

```
# Capture the results of the command into a variable. This can be
displayed, or written to disk.
set results $expect_out(buffer)
# Exit the telnet session, and wait for a special end-of-file character.
send "exit\r"
expect eof
```

Παράδειγμα: Χρήση Expect για σύνδεση telnet σε απομακρυσμένο server[24]

Το ExpectIt [34] είναι μια Java υλοποίηση του εργαλείου Expect. Διατίθεται δωρεάν κάτω από τους όρους της Apache License 2.0. Είναι μια Διεπαφή Προγραμματισμού Εφαρμογών (Application Programming Interface, API), εύκολη στη χρήση, απλή και επεκτάσιμη.

Υπάρχουν κι άλλες δημοφιλείς υλοποιήσεις του εργαλείου Expect για Java, όμως η βιβλιοθήκη αυτή έχει περισσότερα και μοντέρνα χαρακτηριστικά. Στην ιστοσελίδα του ExpectIt [34] μπορούν να βρεθούν πολλά παραδείγματα χρήσης, όπως διαδραστικότητα με SSH server, πελάτης telnet, διαδραστικότητα με απομακρυσμένο φλοιό, διαδραστικότητα με διαδικασίες λειτουργικού συστήματος και άλλα.

```
/**
 * A telnet client example showing weather forecast for a city.
 */

public class TelnetExample {
    public static void main(String[] args) throws IOException {
        TelnetClient telnet = new TelnetClient();
        telnet.connect("rainmaker.wunderground.com");
        StringBuilder wholeBuffer = new StringBuilder();
        Expect expect = new ExpectBuilder()
                .withOutput(telnet.getOutputStream())
                .withInputs(telnet.getInputStream())
                .withEchoOutput(wholeBuffer)
                .withEchoInput(wholeBuffer)
                .withExceptionOnFailure()
                .build();

        expect.expect(contains("Press Return to continue"));
        expect.sendLine();
        expect.expect(contains("forecast city code--"));
        expect.sendLine("SAN");
        expect.expect(contains("X to exit:"));
        expect.sendLine();
        String response = wholeBuffer.toString();
        System.out.println(response);
        expect.close();
    }
}
```

Παράδειγμα: Χρήση ExpectIt για σύνδεση telnet σε απομακρυσμένο server[25]

[24] https://en.wikipedia.org/wiki/Expect

Στην εφαρμογή μας το ExpectIt χρησιμοποιήθηκε για την εκτέλεση εντολών εντός των εικονικών μηχανών μέσω της σειριακής κονσόλας τους (στο επίπεδο του λειτουργικού συστήματος). Επίσης χρησιμοποιήθηκε για την αξιοποίηση του command line εργαλείου "virsh" (κομμάτι του libvirt) σε διάφορες περιπτώσεις, όπως για να παίρνουμε περισσότερες πληροφορίες για τη δικτύωση των εικονικών μηχανών. Ακόμα, χρησιμοποιήθηκε για την αξιοποίηση του command line εργαλείου "virt-clone", ώστε δημιουργούμε κλώνους εικονικών μηχανών και για τη διαγραφή των αρχείων των εικονικών μηχανών.

3.3.6 Apache Maven

Τα εργαλεία αυτοματισμού «χτισίματος» λογισμικού (build automation) καλύπτουν ένα κομμάτι της τεχνικής διαχείρισης έργου λογισμικού. Ο βασικός «τεχνικός» στόχος κάθε έργου λογισμικού είναι η παραγωγή ενός ή περισσότερων software artifacts (.jar, .exe, .deb, .rpm κτλ). Τα software artifacts είναι αυτοτελή αρχεία έτοιμα προς εκτέλεση ή μερικώς αυτοτελή αρχεία προς ενσωμάτωση σε άλλες εφαρμογές (βιβλιοθήκες). Η δομή/περιεχόμενά τους εξαρτάται από τη γλώσσα προγραμματισμού, το λειτουργικό σύστημα, την εφαρμογή κτλ.

Στην Java κοινότητα το software artifact συνήθως είναι ένα .jar αρχείο. Το .jar αρχείο είναι ουσιαστικά ένα zip αρχείο το οποίο περιέχει .class αρχεία (JVM κλάσεις) και ενδεχομένως metadata (manifests), resources (εικόνες), αρχεία ρυθμίσεων κ.ο.κ.

Τα ζητούμενα στην πράξη από ένα εργαλείο αυτοματισμού «χτισίματος» λογισμικού πέρα από την παραγωγή των software artifacts, είναι η αυτόματη διαχείριση εξαρτήσεων, η μεταγλώττιση κώδικα, η εκτέλεση σεναρίων ελέγχου, η συνεχής ολοκλήρωση των software artifacts και η απόθεση/δημοσίευσή τους σε κάποια αποθήκη (software release).

Το Apache Maven [35] είναι ένα εργαλείο αυτοματισμού «χτισίματος» λογισμικού κυρίως του Java οικοσυστήματος. Η πρώτη έκδοση του Apache Maven έγινε το 2004. Χρησιμοποιεί ένα XML αρχείο που ονομάζεται Project Object Model (POM) για την περιγραφή του λογισμικού που θα «χτιστεί» (build), τις εξαρτήσεις του από άλλα εξωτερικά συστατικά λογισμικού

---

[25] https://github.com/Alexey1Gavrilov/ExpectIt/blob/master/expectit-core/src/test/java/net/sf/expectit/TelnetExample.java

(επαναχρησιμοποιήσιμα τμήματα λογισμικού που διατίθενται ως ξεχωριστά software artifacts), τη σειρά «χτισίματος», τους καταλόγους (directories) και τα απαιτούμενα plugins.

Παράδειγμα pom.xml:

```
<project>
  <!-- model version is always 4.0.0 for Maven 2.x POMs -->
  <modelVersion>4.0.0</modelVersion>

  <!-- project coordinates, i.e. a group of values which
       uniquely identify this project -->

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0</version>

  <!-- library dependencies -->

  <dependencies>
    <dependency>

      <!-- coordinates of the required library -->

      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>

      <!-- this dependency is only used for running and compiling tests-->

      <scope>test</scope>

    </dependency>
  </dependencies>
</project>
```

Διαχειρίζεται αυτόματα τις εξαρτήσεις αφού κατεβάζει δυναμικά τις Java βιβλιοθήκες και τα Maven plugins από τοπικές ή δημόσιες αποθήκες. Το Apache Maven διατηρεί τη δική του δημόσια αποθήκη της Java κοινότητας στο Maven Central Repository [36].

Είναι κατασκευασμένο χρησιμοποιώντας μια αρχιτεκτονική που βασίζεται σε plugins, η οποία του επιτρέπει τον χειρισμό των εφαρμογών μέσω του standard input.

Παράδειγμα χρήσης:

```
> ls project
  src pom.xml
> mvn [plugin]:[command]
> mvn [phase]
```

Όπου τα default build lifecycle phases είναι: validate, generate-sources, process-sources, generate-resources, process-resources, compile, process-test-sources, process-test- resources, test-compile, test, package, install, deploy.

Στην εφαρμογή μας χρησιμοποιήσαμε το Apache Maven για να δηλώσουμε με εύκολο τρόπο τις εξαρτήσεις (Spring Framework, ExpectIt API, libvirt API, jQuery βιβλιοθήκη, Bootstrap βιβλιοθήκη), ώστε η εγκατάσταση του λογισμικού να γίνεται γρήγορα και αυτοματοποιημένα αλλά και για να απλοποιήσουμε το τρέξιμο της εφαρμογής αφού χρειάζεται μόνο η εκτέλεση της εντολής: mvn spring-boot:run, όπου το “spring-boot” είναι plugin και το “run” command.

3.4 Shell In A Box

Το Shell In A Box [37] είναι ένας εξομοιωτής τερματικού που βασίζεται στον ιστό (web based terminal emulator), ο οποίος χρησιμοποιεί την τεχνολογία AJAX για να προσφέρει την εμφάνιση και την αίσθηση ενός φυσικού φλοιού (shell).

Είναι δωρεάν κάτω από τους όρους της GNU General Public Licence [22], ανοιχτού κώδικα λογισμικό (open source software) και αναπτύχθηκε από τον Markus Gutschke.

Ο shellinaboxd δαίμονας (daemon) υλοποιεί έναν διακομιστή ιστού (web server) που μπορεί να εξάγει εργαλεία γραμμής εντολών (command line tools) σε έναν εξομοιωτή τερματικού που βασίζεται στον ιστό. Αυτός ο εξομοιωτής είναι προσβάσιμος από οποιοδήποτε φυλλομετρητή ιστού (web browser) που υποστηρίζει JavaScript και CSS και δεν απαιτεί πρόσθετα plugins στον φυλλομετρητή.

Ο shellinaboxd δαίμονας ακούει σε μια συγκεκριμένη θύρα (port). Ο διακομιστής ιστού δημοσιεύει μία ή περισσότερες υπηρεσίες που θα εμφανίζονται σε έναν εξομοιωτή VT100 που υλοποιείται ως εφαρμογή ιστού AJAX. Από προεπιλογή, η θύρα είναι η 4200 και το προεπιλεγμένο URL της υπηρεσίας είναι http://localhost:4200/. Η προεπιλεγμένη θύρα μπορεί να αλλαχτεί σε οποιονδήποτε τυχαίο αριθμό θύρας της επιλογής σας. Το shellinabox παρέχει τη δυνατότητα απομακρυσμένης πρόσβασης στον φλοιό ενός συστήματος μέσω τοπικού συστήματος. Μετά την εγκατάσταση του shellinabox σε όλους τους απομακρυσμένους διακομιστές που είναι επιθυμητή η απόκτηση πρόσβασης, μπορεί να ανοιχθεί ένας φυλλομετρητής ιστού και να γίνει μετάβαση στη διεύθυνση: http://IP-Address:4200/. Αφού πληκτρολογηθεί το όνομα χρήστη και ο κωδικός πρόσβασης μπορεί να αρχίσει η χρησιμοποίηση του φλοιού του απομακρυσμένου συστήματος.

Μία ή περισσότερες υπηρεσίες μπορούν να καταχωρηθούν σε διαφορετικά URL paths με την εντολή που έχει την εξής σύνταξη:

```
ΥΠΗΡΕΣΙΑ: = <url-path> ':' ΕΦΑΡΜΟΓΗ
```

Εάν δεν ζητηθεί κάποια συγκεκριμένη υπηρεσία, ο διακομιστής εκκινεί την προκαθορισμένη εφαρμογή 'LOGIN' η οποία καλεί το /bin/login ζητώντας από το χρήστη το όνομά του και τον κωδικό πρόσβασής του. Στη συνέχεια ξεκινά η σύνδεση στο shell του χρήστη. Αυτή είναι η προεπιλεγμένη επιλογή για τον χρήστη root, αφού απαιτεί δικαιώματα root.

Υπάρχει και μια άλλη προκαθορισμένη εφαρμογή, 'SSH'. Αντί να καλεί το /bin/login, καλεί το ssh. Αυτή είναι η προεπιλεγμένη επιλογή για τους μη προνομιούχους χρήστες (unprivileged users), αν δεν έχει οριστεί καμία υπηρεσία. Αυτή η λειτουργία είναι διαθέσιμη σε privileged και κανονικούς χρήστες. Εάν παραλειφθεί η προαιρετική παράμετρος host υπολογιστή, το shellinaboxd συνδέεται με το localhost.

Εναλλακτικά, μπορεί να οριστεί μια εφαρμογή ορίζοντας έναν χρήστη, έναν κατάλογο εργασίας (working directory) και μια γραμμή εντολών (command line):

```
ΕΦΑΡΜΟΓΗ: = 'LOGIN' | "SSH" [':' <host>] | ΧΡΗΣΤΗΣ ':' CWD ':' CMD
```

Στην εφαρμογή μας ορίσαμε τον shellinaboxd δαίμονα να ακούει σε διαφορετικό αριθμό θύρας για κάθε εικονική μηχανή. Ως υπηρεσία χρησιμοποιήσαμε μια νέα εφαρμογή που τρέχει τη σειριακή κονσόλα της συγκεκριμένης εικονικής μηχανής και ως url-path ορίσαμε το '/uuid', όπου uuid το μοναδικό συνθηματικό της εικονικής μηχανής στο libvirt.

Υπάρχει δυνατότητα εγκατάστασης πιστοποιητικών SSL/TLS, ώστε όλες οι επικοινωνίες πελάτη-διακομιστή να είναι κρυπτογραφημένες.

3.5 Git

Η ανάγκη για συστήματα διαχείρισης εκδόσεων λογισμικού προκύπτει όταν πρέπει να διαχειριστούμε την ανάπτυξη και τη συντήρηση μεγάλων έργων λογισμικού. Σε αυτή την περίπτωση απαιτείται η διαχείριση του source code base (της «βάσης» του κώδικα), η καταγραφή των αλλαγών (ιστορικό, ποιος έκανε τι, πώς, πότε), η συνεργατική ανάπτυξη, η

τήρηση πολλών παράλληλων καταστάσεων του κώδικα ταυτόχρονα καθώς και η ανάκτηση συγκεκριμένης παρελθούσας κατάστασης.

Το σύστημα Git [38] καλύπτει όλες τις παραπάνω απαιτήσεις καθώς είναι το πλέον διαδεδομένο κατανεμημένο σύστημα ελέγχου εκδόσεων (distributed version control system), παρέχεται δωρεάν κάτω από τους όρους της GNU General Public Licence [22], και είναι ανοιχτού κώδικα (open source). Αρχικά σχεδιάστηκε και αναπτύχθηκε το 2005 από τον Linus Torvald [39] για την ανάπτυξη του πυρήνα Linux.

Το Git υποστηρίζει τη μη γραμμική ανάπτυξη λογισμικού με χρήση διακλαδώσεων ιστορικού (branches).

Κάθε προγραμματιστής διατηρεί το δικό του τοπικό αποθετήριο λογισμικού (repository), μαζί με όλο το ιστορικό των commits (δηλαδή τις αποθηκευμένες αλλαγές του κώδικα), το οποίο σημαίνει ότι το Git είναι γρήγορο αφού δεν χρειάζεται σύνδεση δικτύου για να κάνεις commit αλλαγές, να ελέγξεις προηγούμενες εκδόσεις ενός αρχείου, να συγχωνέψεις διακλαδώσεις (merge branches), να αλλάξεις διακλάδωση (switch branch) και να δεις το ιστορικό.

Σε περίπτωση συνεργατικής ανάπτυξης, είναι εφικτή η ταυτόχρονη εργασία στο ίδιο αρχείο από πολλά άτομα. Όταν πάνε να αποθηκεύσουν τις αλλαγές τους, το Git προσπαθεί να τις συγχωνέψει στο αρχικό αρχείο και σε περίπτωση σύγκρουσης (conflict) τους καλεί να επιλέξουν το τι θα αποθηκεύσουν τελικά και ποιος είναι αυτός που έκανε την αλλαγή.

Μπορεί να στηθεί τοπικά ή σε κάποιον απομακρυσμένο διακομιστή στον οποίον θα έχουν πρόσβαση μέλη που δουλεύουν στην ίδια εφαρμογή.

Το Git μαθαίνεται εύκολα και είναι σχεδιασμένο για να μπορεί να χειριστεί από μικρά μέχρι πολύ μεγάλα έργα λογισμικού με ταχύτητα και αποδοτικότητα. Έχει πολύ καλό documentation διαθέσιμο δωρεάν σε μορφή E-Book [40].

4. Ανάλυση Εφαρμογής

Σε αυτό το κεφάλαιο δίνονται λεπτομέρειες ανάλυσης της εφαρμογής, συγκεκριμένα δίνεται ο σκοπός της εφαρμογής, αναλύοντας τις ανάγκες που οδήγησαν στη δημιουργία της, αναλύονται οι απαιτήσεις της εφαρμογής και γίνεται η καταγραφή των προδιαγραφών της εφαρμογής.

4.1 Σκοπός Εφαρμογής lab-on-demand

Σκοπός της εφαρμογής παγκόσμιου ιστού, lab-on-demand, είναι η απλή, γρήγορη και μαζική κατασκευή εικονικών μηχανών και η παραμετροποίησή τους, ώστε να δημιουργηθεί η επιθυμητή εικονικοποιημένη τοπολογία δικτύου.

Η εφαρμογή αυτή θα μπορούσε να είναι ιδιαίτερα χρήσιμη σε φοιτητές που παρακολουθούν μαθήματα εξομοίωσης συστημάτων επικοινωνιών, αντίστοιχα με αυτό που προσφέρεται από το Εθνικό Μετσόβιο Πολυτεχνείο (Εξομοίωση Συστημάτων Επικοινωνιών, 8ο Εξάμηνο, Ροή Δ, Επιλογής). Η εφαρμογή ανταποκρίνεται απόλυτα στις ανάγκες του εργαστηρίου του συγκεκριμένου μαθήματος και έχει σκοπό να αυτοματοποιήσει τη διαδικασία δημιουργίας και παραμετροποίησης εικονικοποιημένων τοπολογιών δικτύου, η οποία χρειαζόταν περίπου 4 φορές σε κάθε άσκηση (4 x τουλάχιστον 2 εικονικά μηχανήματα x 12 ασκήσεις = περισσότερες από 96 εικονικές μηχανές ανά φοιτητή έπρεπε να δημιουργηθούν και παραμετροποιηθούν). Η χρήση του VirtualBox [41] της Oracle στο εργαστήριο, για την εκτέλεση λειτουργικών συστημάτων σε εικονικό περιβάλλον, είχε ως αποτέλεσμα το χάσιμο πολύτιμου χρόνου για τη δημιουργία και παραμετροποίηση μεγάλων τοπολογιών και κυρίως για την αναμονή μέχρι να ξεκινήσουν τα εικονικά μηχανήματα.

Επίσης, η εφαρμογή αυτή εκτός από εκπαιδευτικούς σκοπούς, θα μπορούσε να χρησιμοποιηθεί για τη δοκιμή διαφόρων δικτυακών τοπολογιών, πριν την κατασκευή των φυσικών δικτύων και την αγορά φυσικών δικτυακών συσκευών. Αυτό θα μπορούσε να μειώσει έξοδα από την αγορά υλικού που τελικά μπορεί να είναι περιττό ή να μην είναι η βέλτιστη επιλογή.

Ένα άλλο πλεονέκτημα της εφαρμογής είναι ότι δεν καταναλώνονται πόροι (CPU + μνήμη) του υπολογιστή του τελικού χρήστη/πελάτη (client) για εικονικοποίηση, ούτε χρειάζεται να υποστηρίζει ο υπολογιστής του την εικονικοποίηση. Το μόνο που απαιτείται από τον πελάτη

είναι σύνδεση στο διαδίκτυο και ένας φυλλομετρητής ιστού (web browser) που να υποστηρίζει JavaScript και CSS. Ο διακομιστής (server) αναλαμβάνει την υλοποίηση των τοπολογιών, τη διαχείρισή τους και τη ρύθμισή τους. Ως αποτέλεσμα, ο πελάτης είναι συμβατός με όλα τα λειτουργικά συστήματα και δεν απαιτεί την εγκατάσταση πρόσθετου λογισμικού.

Εκτός από τον υπολογιστή, η εφαρμογή είναι εύκολα προσβάσιμη και από smartphones, tablets και γενικότερα σε όλες τις συσκευές με σύνδεση στο διαδίκτυο, αφού είναι διαθέσιμη σε μορφή ιστοσελίδας.

Η εφαρμογή θα βοηθήσει όσους θέλουν να επικεντρωθούν στη μελέτη και τη δοκιμή της λειτουργίας των δικτύων υπολογιστών και των δικτυακών πρωτοκόλλων που χρησιμοποιούνται στο διαδίκτυο, χωρίς να χάνουν χρόνο στην κατασκευή της εικονικοποιημένης τοπολογίας δικτύου.

4.2 Απαιτήσεις

Στο στάδιο καταγραφής και ανάλυσης απαιτήσεων της εφαρμογής καταλήξαμε στις παρακάτω κύριες απαιτήσεις:

- Ο χρήστης θα μπορεί να κατασκευάζει μια τοπολογία δικτύου επιλέγοντας για κάθε επιμέρους δικτυακή συσκευή: μία αρχική εικόνα (image) από ένα σύνολο «βασικών» εικόνων που καθορίζουν το λειτουργικό σύστημα και τον προσανατολισμό χρήσης της συσκευής, και ένα σετ ρυθμίσεων (σε επίπεδο hypervisor και λειτουργικού συστήματος) που θα εφαρμοστεί στην αρχική εικόνα, ώστε η εικονική μηχανή που προκύπτει να επιτυγχάνει το δικτυακό της ρόλο στην τοπολογία.
- Ο χρήστης θα μπορεί να παραμετροποιήσει τους εικονικούς τρόπους δικτύωσης.
- Ο χρήστης θα μπορεί να δημιουργεί πανομοιότυπες δικτυακές συσκευές με μαζικό τρόπο.
- Ο χρήστης θα έχει τη δυνατότητα επικοινωνίας με κάθε εικονική μηχανή μέσω γραμμής εντολών.
- Ο χρήστης θα διαχειρίζεται τις εικονικές μηχανές μέσω μιας διεπαφής ιστού (web interface).
- Η υλοποίηση και οι ρυθμίσεις της περιγραφόμενης από τον χρήστη διάταξης, θα υλοποιούνται μέσω ενός διαθέσιμου hypervisor σε έναν διακομιστή.

4.3 Προδιαγραφές

Μετά τον καθορισμό απαιτήσεων θα πρέπει να οριστούν κάποια βασικά χαρακτηριστικά και προδιαγραφές της εφαρμογής, ώστε να είναι χρήσιμη και λειτουργική προς τον χρήστη, πριν προχωρήσουμε στον σχεδιασμό και την υλοποίηση της εφαρμογής.

Αφού αναλύσαμε τις απαιτήσεις στην προηγούμενη παράγραφο, προέκυψαν οι παρακάτω προδιαγραφές που πρέπει να καλύπτει η εφαρμογή:

- Θα υπάρχει μία λίστα στην οποία θα εμφανίζονται οι διαθέσιμες εικονικές μηχανές, όπου για κάθε εικονική μηχανή θα υπάρχει μία κάρτα που θα δίνει βασικές πληροφορίες (τρέχει/απενεργοποιημένη, ρόλος μηχανής/λειτουργικό σύστημα, όνομα, δικτύωση) και βασικά κουμπιά για τον χειρισμό/χρήση της (ενεργοποίηση/απενεργοποίηση, διαγραφή, πρόσβαση στην κονσόλα). Στον τίτλο της λίστας θα εμφανίζεται και το πλήθος των διαθέσιμων εικονικών μηχανών.

- Οι εικονικές μηχανές θα διαχωρίζονται ανάλογα με την κατάστασή τους. Οι μηχανές που «τρέχουν» θα εμφανίζουν πράσινο περίγραμμα στην κάρτα τους και οι μηχανές που είναι απενεργοποιημένες κόκκινο περίγραμμα.

- Στη λίστα με της διαθέσιμες εικονικές μηχανές, οι «βασικές» εικόνες θα εμφανίζουν μόνο τον ρόλο τους (πχ. «basic» Η/Υ, «basic» Δρομολογητής), αποκρύπτοντας κουμπιά για τον χειρισμό/χρήση τους. Επίσης θα έχουν γκρίζο χρώμα στο περίγραμμά τους και στο κείμενο περιγραφής τους.

- Θα υπάρχει μία λίστα στην οποία θα εμφανίζονται τα διαθέσιμα εικονικά δίκτυα και θα χωρίζονται σε δύο κατηγορίες: Ενεργά, Ανενεργά. Για κάθε εικονικό δίκτυο θα υπάρχει κουμπί για τη διαγραφή του. Θα εμφανίζεται και το πλήθος εικονικών δικτύων σε κάθε κατηγορία.

- Θα υπάρχουν κουμπιά για την επιλογή του αριθμού, Ν «βασικών» εικόνων, που πρόκειται να δημιουργηθούν και να παραμετροποιηθούν. Όταν πατηθεί το κουμπί «Πρόσθεσε» θα εμφανίζονται στην οθόνη Ν δομές (σε μορφή accordion) στις οποίες θα μπορεί ο χρήστης να περάσει τις επιθυμητές ρυθμίσεις και ένα κουμπί «Δημιουργία».

Όταν πατηθεί το κουμπί «Ακύρωση» οι δομές αυτές θα εξαφανίζονται από την οθόνη, μαζί με το κουμπί «Δημιουργία».

- Πάνω από κάθε δομή εικονικής μηχανής που πρόκειται να δημιουργηθεί και να παραμετροποιηθεί θα αναγράφεται ο ρόλος της (πχ. Η/Υ, δρομολογητής). Ο χρήστης θα μπορεί μέσα από τη δομή αυτή να ορίσει γενικές και δικτυακές ρυθμίσεις.

- Στις γενικές ρυθμίσεις κάθε εικονικού μηχανήματος που πρόκειται να δημιουργηθεί, θα δίνεται ένα προεπιλεγμένο hostname, το οποίο θα μπορεί ο χρήστης να αλλάξει πληκτρολογώντας αυτό που επιθυμεί στο αντίστοιχο text-box.

- Στις δικτυακές ρυθμίσεις κάθε εικονικού μηχανήματος που πρόκειται να δημιουργηθεί, ο χρήστης θα μπορεί να προσθέσει/αφαιρέσει μέχρι 8 κάρτες δικτύου πατώντας τα ανάλογα κουμπιά. Θα υπάρχει drop-down λίστα για την επιλογή του τρόπου δικτύωσης και text-box για να πληκτρολογήσει το όνομα του δικτύου.

- Θα υπάρχει κουμπί «Δημιουργία» που θα ξεκινά τη δημιουργία της τοπολογίας, θα κάνει τις επιθυμητές ρυθμίσεις και έπειτα θα τρέχει την τοπολογία. Στη διεπαφή θα εμφανίζονται τα κατάλληλα μηνύματα ανάλογα με την εξέλιξη αυτής της διαδικασίας.

- Ο χρήστης θα μπορεί να έχει πρόσβαση στη γραμμή εντολών του κάθε εικονικού μηχανήματος πατώντας ένα κουμπί το οποίο θα ανοίγει αυτόματα ένα νέο tab στον φυλλομετρητή ιστού. Στη νέα σελίδα θα εμφανίζεται ένα κεντραρισμένο πλαίσιο που θα περιέχει την οθόνη με τον φλοιό της εικονικής μηχανής. Ο χρήστης θα έχει τη δυνατότητα να πληκτρολογήσει και να εκτελέσει νέες εντολές μέσα στο εικονικό μηχάνημα μέσω του πλαισίου αυτού.

- Εκτός από την αρχική σελίδα, ο χρήστης θα έχει τη δυνατότητα ενεργοποίησης και απενεργοποίησης της εικονικής μηχανής μέσω κουμπιών και μέσα από τη σελίδα πρόσβασης στη γραμμή εντολών που της αντιστοιχεί.

5. Αρχιτεκτονική και Σχεδίαση Εφαρμογής

Αυτό το κεφάλαιο θα αναφερθεί στις σχεδιαστικές αποφάσεις και επιλογές, που προέκυψαν μετά την ανάλυση και την καταγραφή των απαιτήσεων και προδιαγραφών της εφαρμογής (βλέπε Ανάλυση Εφαρμογής). Έπειτα θα περιγράψει αναλυτικά την αρχιτεκτονική της εφαρμογής.

5.1 Σχεδιαστικές Επιλογές

Κατά το στάδιο σχεδιασμού της εφαρμογής λήφθηκαν οι παρακάτω σχεδιαστικές αποφάσεις.

Αρχικά επιλέγηκε το σύνολο των «βασικών» εικόνων που θα χρησιμοποιούσε η εφαρμογή για την κατασκευή της τοπολογίας δικτύου. Αποφασίσαμε ότι για τον σκοπό αυτό θα έπρεπε να έχουμε μία «βασική» εικόνα που θα είχε τον ρόλο ενός ηλεκτρονικού υπολογιστή (Η/Υ) και μία άλλη «βασική» εικόνα που θα είχε τον ρόλο ενός δρομολογητή. Για τον Η/Υ κρίθηκε ότι το κατάλληλο λειτουργικό σύστημα θα ήταν το FreeBSD, ενώ για τον δρομολογητή το FreeBSD μαζί με την εγκατάσταση του Quagga. Περιληπτικά, το FreeBSD είναι μία έκδοση του UNIX ενώ το Quagga είναι ένα εξειδικευμένο λογισμικό δρομολόγησης ανοικτού κώδικα, για λειτουργικά συστήματα τύπου UNIX. Το Quagga περιλαμβάνει υλοποιήσεις διαφόρων δυναμικών πρωτοκόλλων (OSPF, RIP, BGP, IS-IS) καθώς και πολλές δυνατότητες στατικής και δυναμικής δρομολόγησης.

Για τη μαζική δημιουργία πανομοιότυπων δικτυακών συσκευών επιλέχθηκε να χρησιμοποιηθεί ένα εργαλείο που θα επέτρεπε τη δημιουργία κλώνων από μία αρχική εικονική μηχανή. Έτσι η διαδικασία αυτή θα μπορούσε να αυτοματοποιηθεί και δεν θα απαιτούσε πολλές γραμμές κώδικα για να υλοποιηθεί. Το εργαλείο που επιλέχθηκε ήταν το “virt-clone” και η υλοποίηση με κώδικα θα αναλυθεί σε επόμενη παράγραφο.

Εξ αρχής αποφασίσαμε ότι η επικοινωνία με τη γραμμή εντολών της κάθε εικονικής μηχανής θα γίνεται μέσω της σειριακής της κονσόλας αντί μέσω ssh για παράδειγμα, ώστε να είναι δυνατή η πρόσβαση στον φλοιό (shell) όλων των εικονικών μηχανών ακόμη και αυτών που δεν έχουν κάρτα δικτύου ή δεν τους έχει αντιστοιχηθεί IP διεύθυνση. Διαφορετικά, αν χρησιμοποιούσαμε το ssh, θα απαιτούσε σύνδεση δικτύου και άρα θα έπρεπε να δημιουργούμε

μια επιπλέον κάρτα δικτύου σε κάθε εικονική μηχανή, αποκλειστικά για την επικοινωνία με τον φλοιό της. Αυτό θα είχε ως αποτέλεσμα τη σπατάλη επιπλέον πόρων του host για εικονικοποίηση και θα καθυστερούσε τη διαδικασία δημιουργίας αλλά και έναρξης μιας τοπολογίας. Επίσης το ssh είναι πιο ακριβό σαν υλοποίηση σε σύγκριση με τη σειριακή κονσόλα, καθώς εκτός από την απαίτηση δικτυακής σύνδεσης, κάνει κρυπτογράφηση και απαιτεί να τρέχει ένας δαίμονας, ο sshd, στο παρασκήνιο που ακούει για συνδέσεις. Η σειριακή κονσόλα ήταν η βέλτιστη επιλογή αφού η πρόσβαση γίνεται τοπικά από τον host και άρα δεν θα υπήρχε νόημα να απαιτηθεί δικτυακή σύνδεση. Το εργαλείο επιλέχθηκε για την πρόσβαση στη σειριακή κονσόλα του κάθε εικονικού μηχανήματος είναι το “virsh console”.

Στη συνέχεια έπρεπε να καθορίσουμε τον τρόπο που θα μας επέτρεπε να ορίζουμε ρυθμίσεις εντός των εικονικών μηχανών, στο επίπεδο του λειτουργικού συστήματος, μέσω κώδικα. Για να το πετύχουμε αυτό αποφασίσαμε ότι πρώτα θα έπρεπε να εξασφαλίζουμε επικοινωνία με τη σειριακή κονσόλα του εικονικού μηχανήματος και έπειτα μέσω αυτής να περνούμε την επιθυμητή ρύθμιση. Η υλοποίηση σε κώδικα δοθεί σε επόμενη παράγραφο.

Έπειτα επιλέξαμε το σύνολο των τρόπων δικτύωσης των εικονικών μηχανών, στο επίπεδο του hypervisor. Αποφασίσαμε ότι οι τρόποι δικτύωσης που θα έπρεπε να υποστηρίζει η εφαρμογή είναι: το εσωτερικό δίκτυο, το NAT δίκτυο, το routed δίκτυο και το isolated δίκτυο (ίδιο με host-only), ώστε να καλυφθούν οι περισσότερες ανάγκες του χρήστη και να τον βοηθήσει να χτίσει τα δίκτυα που επιθυμεί. Η υλοποίηση σε κώδικα δοθεί σε επόμενη παράγραφο.

Μία άλλη σχεδιαστική απόφαση που πήραμε ήταν ότι η σύνδεση του host με τον δαίμονα του libvirt (libvirtd) θα ξεκινά μετά από κάθε αίτηση του χρήστη και θα κλείνει μόλις ολοκληρωθεί η επεξεργασία και ληφθούν τα δεδομένα που απαιτεί η αίτηση αυτή. Δηλαδή η σύνδεση δε θα κρατιέται ανοιχτή συνέχεια για όλη τη διάρκεια που τρέχει η εφαρμογή.

Οι λίστες με τις διαθέσιμες εικονικές μηχανές και τα διαθέσιμα εικονικά δίκτυα θα πρέπει να ανανεώνεται ασύγχρονα κάθε κάποια δευτερόλεπτα, ώστε ο χρήστης να βλέπει την τρέχουσα κατάσταση των εικονικών μηχανών και εικονικών δικτύων. Αυτό σημαίνει ότι κάθε κάποια δευτερόλεπτα το host μηχάνημα θα ανοίγει μια σύνδεση στο παρασκήνιο με τον libvirtd για να πάρει τις πληροφορίες που χρειάζεται και όταν ολοκληρωθεί η διαδικασία η σύνδεση θα κλείνει.

Η εφαρμογή θα έχει 2 εσωτερικά υποσυστήματα υπηρεσιών, ώστε να διαχωρίζονται οι λειτουργίες των εικονικών μηχανών από τις λειτουργίες δικτύωσης. Κάθε υποσύστημα θα υλοποιείται ως ένα interface που θα περιλαμβάνει τις εξής μεθόδους:

```
public interface DomainService {

    List<Domain> getAllDomains() throws LibvirtException;

    List<Domain> getActiveDomains() throws LibvirtException;

    List<Domain> getInactiveDomains() throws LibvirtException;

    List<DomainInfo> getAllDomainsInfo() throws LibvirtException;

    DomainInfo getDomainInfo(String uuid) throws LibvirtException;

    String getDomainUUID(String name) throws LibvirtException;

    List<String> getDomIfList(String uuid) throws IOException;

    Domain getDomainByName (String name) throws LibvirtException;

    Domain getDomainByUUID (String uuid) throws LibvirtException;

    boolean startDomain(String uuid) throws LibvirtException;

    boolean shutdownDomain(String uuid) throws LibvirtException;

    String cloneDomain(String name) throws IOException,
InterruptedException, LibvirtException;

    String executeCommand(String vmname, String cmd) throws JSchException,
IOException;

    String startDomainAndExecuteCommand(String vmname, String cmd) throws
JSchException, IOException, LibvirtException;

      String domainShellinaboxd(String uuid) throws LibvirtException,
IOException, InterruptedException;

      boolean deleteDomain(String uuid) throws LibvirtException,
IOException, InterruptedException;

}

public interface NetworkService {

      List<Network> getNetworks(boolean isActive) throws LibvirtException;

      List<String> getNetworkNames(boolean isActive) throws
LibvirtException;

      boolean attachInterface(String vmname, String netname) throws
LibvirtException, IOException;

      boolean defineAndStartNetwork(String name, String type) throws
LibvirtException;
```

```
    boolean destroyAndUndefineNetwork(String netname) throws
LibvirtException, IOException;

}
```

5.2 Αρχιτεκτονική Εφαρμογής

Σε αυτή το υποκεφάλαιο θα περιγραφεί ο αρχιτεκτονικός σχεδιασμός της εφαρμογής και θα δοθεί σχηματική δομή με τη συνολική αρχιτεκτονική της εφαρμογής.

Η εφαρμογή υποστηρίζει το αρχιτεκτονικό στυλ για τη λειτουργία του Παγκόσμιου Ιστού, Representational State Transfer (REST). Επίσης καλύπτει τις βασικές αρχές του RESTful, όπως την αρχή πελάτη-διακομιστή (client-sevrer), με τον πλήρη διαχωρισμό ενδιαφερόντων μεταξύ του πελάτη και του διακομιστή και τη χρησιμοποίηση μιας ομοιόμορφης διεπαφής για την επικοινωνία τους. Επίσης είναι stateless, δηλαδή ο διακομιστής δεν αποθηκεύει καμιά πληροφορία για την κατάσταση της εφαρμογής κατά την επικοινωνία του με τον πελάτη. Με απλά λόγια, ο πελάτης στέλνει αιτήματα (requests) στον διακομιστή χρησιμοποιώντας URIs για τον προσδιορισμό των πόρων και ο διακομιστής ανταποκρίνεται με απαντήσεις (response) στέλνοντας τα στοιχεία του πόρου σε μορφή JSON (οι πόροι διαχωρίζονται από τις αναπαραστάσεις). Ο πελάτης επικοινωνεί με τον διακομιστή μόνο μέσω των δυναμικά παρεχόμενων hypermedia, δηλαδή δεν χρειάζεται να γνωρίζει από πριν κάποιο διακομιστή ή κάποια διεπαφή (όπως στο SOA και στο RPC κτλ.), αρκεί μόνο να μπορεί να καταλάβει τη λειτουργία των hypermedia.

Υλοποιεί ένα RESTful Web Service (API), προσφέροντας ένα σταθερό HTTP base URL και υποστηρίζει τις HTTP μεθόδους GET, PUT, POST, DELETE. Το RESTful API της εφαρμογής σχεδιάστηκε έτσι ώστε να καλύπτει όλες τις ανάγκες της εφαρμογής και να είναι ευέλικτο, ως εξής:

- HTTP GET αίτημα στο “/api/domain/list” επιστρέφει μια λίστα με τις βασικές πληροφορίες όλων των εικονικών μηχανών.
- HTTP GET αίτημα στο “/api/domain/{uuid}/info” επιστρέφει τις βασικές πληροφορίες της εικονικής μηχανής που έχει UUID στο libvirt ίσο με την τιμή της path variable uuid.
- HTTP GET αίτημα στο “/api/domain/{name}/uuid” επιστρέφει το UUID της εικονικής μηχανής στο libvirt που έχει όνομα στο libvirt ίσο με την τιμή της path variable name.

- HTTP GET αίτημα στο “/api/domain/{uuid}/iflist” επιστρέφει μια λίστα με τις δικτυακές διεπαφές της εικονικής μηχανής που έχει UUID στο libvirt ίσο με την τιμή της path variable uuid.
- HTTP POST αίτημα στο “/api/domain/{uuid}/clone” δημιουργεί έναν κλώνο της εικονικής μηχανής που έχει UUID στο libvirt ίσο με την τιμή της path variable uuid και επιστρέφει το UUID της νέας εικονικής μηχανής που δημιουργήθηκε.
- HTTP PUT αίτημα στο “/api/domain/{uuid}/start” ξεκινά την εικονική μηχανή που έχει UUID στο libvirt ίσο με την τιμή της path variable uuid και επιστρέφει true εάν η διαδικασία ήταν επιτυχής και false εάν όχι.
- HTTP PUT αίτημα στο “/api/domain/{uuid}/shutdown” απενεργοποιεί την εικονική μηχανή που έχει UUID στο libvirt ίσο με την τιμή της path variable uuid και επιστρέφει true εάν η διαδικασία ήταν επιτυχής και false εάν όχι.
- HTTP PUT αίτημα στο “/api/domain/{uuid}/delete” διαγράφει την εικονική μηχανή που έχει UUID στο libvirt ίσο με την τιμή της path variable uuid και επιστρέφει true εάν η διαδικασία ήταν επιτυχής και false εάν όχι.
- HTTP GET αίτημα στο “/api/domain/{vmname}/{cmd}” εκτελεί την εντολή που δίνεται από την τιμή της path variable cmd μέσα στο τερματικό του εικονικού μηχανήματος που έχει όνομα στο libvirt ίσο με την τιμή της path variable vmname και επιστρέφει μια συμβολοσειρά με την ανταπόκριση του τερματικού.
- HTTP POST αίτημα στο “/api/domain/{uuid}/shellinabox” ξεκινά μία νέα σύνδεση με τον shellinabox δαίμονα (shellinaboxd), δηλώνοντας νέα μια υπηρεσία για την πρόσβαση στη σειριακή κονσόλα της εικονικής μηχανής που έχει UUID στο libvirt ίσο με την τιμή της path variable uuid και επιστρέφει μια συμβολοσειρά με τον αριθμό της πόρτας στην οποία ακούει ο shellinaboxd.
- HTTP PUT αίτημα στο “/api/domain/{vmname}/{netname}/attachInterface” δημιουργεί μία νέα εικονική δικτυακή διεπαφή στην εικονική μηχανή που έχει όνομα στο libvirt ίσο με την τιμή της path variable vmname και τη συνδέει στο εικονικό δίκτυο με όνομα στο libvirt ίσο με την τιμή της path variable netname. Επιστρέφει true εάν η διαδικασία ήταν επιτυχής και false εάν όχι.
- HTTP POST αίτημα στο “/api/network/{name}/{type}/create” δημιουργεί ένα νέο εικονικό δίκτυο με όνομα στο libvirt ίσο με την τιμή της path variable name και τρόπο δικτύωσης ανάλογο με την τιμή της path variable type και επιστρέφει true εάν η διαδικασία ήταν επιτυχής και false εάν όχι.

- HTTP GET αίτημα στο “/api/network/{isActive}/namelist” ανάλογα με την boolean τιμή της path variable isActive, επιστρέφει μια λίστα με τα ονόματα των ενεργών ή ανενεργών εικονικών δικτύων στο libvirt.
- HTTP PUT αίτημα στο “/api/network/{name}/delete” διαγράφει το εικονικό δίκτυο που έχει όνομα στο libvirt ίσο με την τιμή της path variable name και επιστρέφει true εάν η διαδικασία ήταν επιτυχής και false εάν όχι.

Ο web controller ο οποίος είναι υπεύθυνος για την HTML σελίδα που θα εμφανιστεί στον χρήστη σχεδιάστηκε ως εξής:

- Για URL ‘/’ εμφανίζεται η αρχική σελίδα της εφαρμογής.
- Για URL ‘/domain/{uuid}’ εμφανίζεται η σελίδα του τερματικού της εικονικής μηχανής με UUID στο libvirt ίσο με την τιμή της path variable uuid.
- Για URL ‘/domain/{uuid}/iflist’ εμφανίζεται η σελίδα με τις εικονικές δικτυακές διεπαφές του εικονικού μηχανήματος με UUID στο libvirt ίσο με την τιμή της path variable uuid.

Οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής, μαζί με τους λόγους που υπαγόρευσαν την επιλογή τους, περιγράφηκαν αναλυτικά σε προηγούμενο κεφάλαιο (βλέπε Τεχνολογίες / Εργαλεία) καθώς και το libvirt (βλέπε Θεωρητικό Υπόβαθρο).

Σε συντομία και συγκεντρωτικά στο **Back-End** (server-side) έχουμε:

Γλώσσα Προγραμματισμού: Java

Web Framework: Spring

Web Server: Apache Tomcat

Λειτουργικό Σύστημα: Linux

Hypervisor: KVM

Βιβλιοθήκες/API: libvirt, ExpectIt

Build automation tool: Apache Maven + spring-boot:run plugin

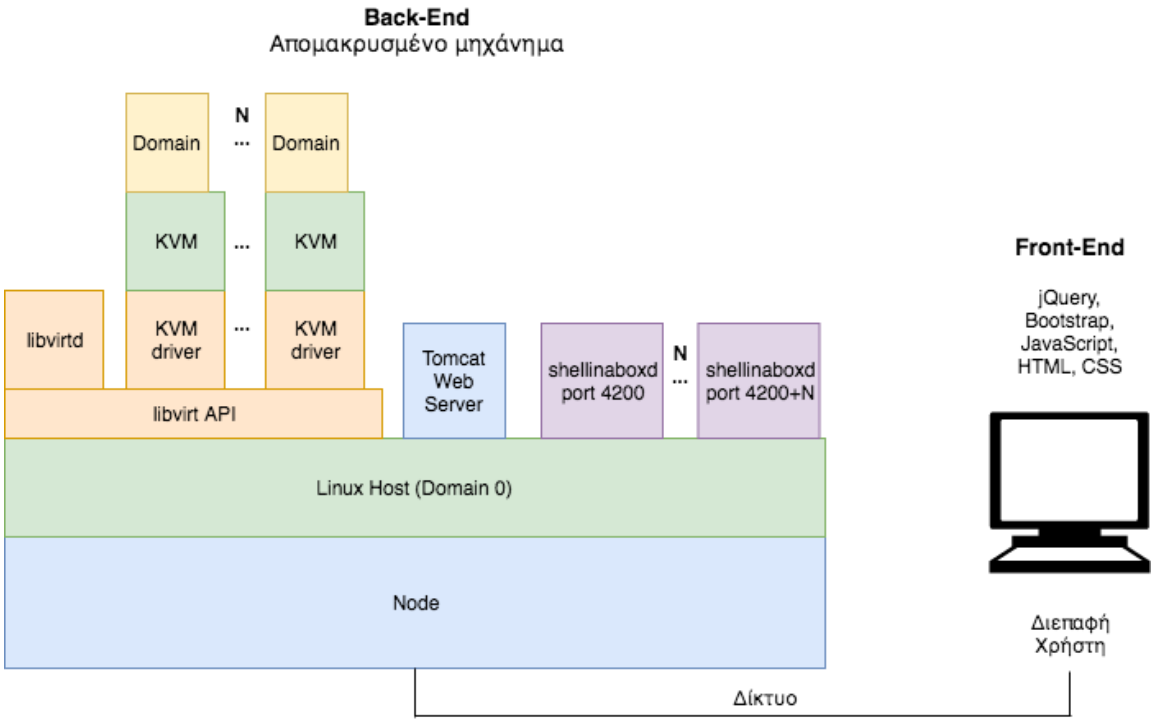
Επιπλέον στον διακομιστή είναι εγκατεστημένο το libvirt και τρέχουν οι δαίμονες του libvirt (libvirtd) και του shellinabox (shellinaboxd) όταν χρειαστεί.

Και στο **Front-End** (client-side) έχουμε:

HTML, JavaScript, CSS

Βιβλιοθήκες/API: jQuery, Bootstrap

Η συνολική αρχιτεκτονική της εφαρμογής απεικονίζεται στο Σχήμα 25.



Σχήμα 25: Η αρχιτεκτονική της εφαρμογής lab-on-demand

Όπως παρατηρούμε στο Σχήμα 25, η αρχιτεκτονική της εφαρμογής είναι client-server, όπου ο client είναι ο χρήστης και ο server το απομακρυσμένο μηχάνημα. Στο server-side, δηλαδή στο Back-End όλες οι τεχνολογίες και τα εργαλεία που χρησιμοποιήσαμε τρέχουν πάνω από το λειτουργικό σύστημα Linux. Η επικοινωνία του client με τον server γίνεται μέσω δικτύου. Στο client-side, δηλαδή στο Front-End βρίσκεται η διεπαφή του χρήστη, που είναι μια ιστοσελίδα, στην οποία έχει πρόσβαση μέσω ενός φυλλομετρητή ιστού (πχ. Google Chrome).

Ο χρήστης επικοινωνεί με το απομακρυσμένο μηχάνημα χρησιμοποιώντας την ιστοσελίδα, ανταλλάσσοντας HTTP μηνύματα. Όταν ο χρήστης μεταβεί σε μια σελίδα μέσω του φυλλομετρητή του, στέλνεται ένα HTTP GET αίτημα στον web server (Apache Tomcat) που τρέχει στον server. Τότε ο web controller του web server αποφασίζει ποια όψη, δηλαδή ποια σελίδα θα εμφανιστεί στον πελάτη ανάλογα με το URL που έστειλε μέσα στο HTTP μήνυμα. Για να ζητήσει ο χρήστης resources (πχ. πατώντας ένα κουμπί για να ξεκινήσει μια εικονική μηχανή), στέλνεται ένα ασύγχρονο HTTP αίτημα μέσω AJAX, το οποίο φτάνει στον rest controller του web server. Τότε ο rest controller με βάση το URL και την HTTP μέθοδο του αιτήματος εκτελεί τον κατάλληλο handler, δηλαδή την κατάλληλη Java μέθοδο και επιστρέφει τα resources σε μορφή JSON μέσα σε HTTP απαντήσεις. Τότε το Front-End (jQuery, Bootsrap,

JavaScript, HTML, CSS) είναι υπεύθυνο για την αναπαράσταση των resources στον φυλλομετρητή ιστού του χρήστη.

Κάθε shellinabox δαίμονας (shellinaboxd) ξεκινά να τρέχει στο απομακρυσμένο μηχάνημα σε διαφορετική πόρτα για κάθε εικονική μηχανή, όταν ζητηθεί από τον χρήστη, πατώντας τον σύνδεσμο «Πρόσβαση στην κονσόλα».

Η αρχιτεκτονική του libvirt επεξηγήθηκε αναλυτικά σε προηγούμενο υποκεφάλαιο (βλέπε 2.2.3 libvirt). Το libvirt API τρέχει πάνω από το λειτουργικό σύστημα Linux του host, όπου στην περίπτωσή μας ο host είναι το απομακρυσμένο μηχάνημα. Το libvirt API εξυπηρετεί με κοινό τρόπο όλους τους hypervisors χρησιμοποιώντας οδηγούς (drivers). Ο hypervisor που χρησιμοποιήσαμε για τη δοκιμή της εφαρμογής μας ήταν ο KVM. Παρ' όλ' αυτά, θα μπορούσε να αντικατασταθεί με οποιονδήποτε άλλο hypervisor που υποστηρίζει το libvirt και έχει όλες τις λειτουργίες API που χρησιμοποιήσαμε στην εφαρμογή, χωρίς να χρειαστεί τροποποίηση του κώδικα της εφαρμογής. Το KVM αναλαμβάνει την εικονικοποίηση των guests, οι οποίοι αποκαλούνται domains στο libvirt. Κάθε φορά που θέλουμε να πάρουμε πληροφορίες, να δημιουργήσουμε ή να διαχειριστούμε εικονικές μηχανές, ξεκινά μία τοπική σύνδεση στον host με τον libvirt δαίμονα (libvirtd), ο οποίος τρέχει πάνω από το libvirt API. Όταν τελειώσει η διαδικασία η σύνδεση κλείνει.

6. Θέματα Υλοποίησης

Σε αυτό το κεφάλαιο θα δοθούν και θα εξηγηθούν μερικά από τα σημαντικότερα κομμάτια κώδικα με τα οποία υλοποιήθηκε η εφαρμογή. Συγκεκριμένα, θα δούμε τη δημιουργία κλώνων εικονικών μηχανών, την εκτέλεση εντολών εντός της εικονικής μηχανής μέσω Java, τον ορισμό εικονικοποιημένων δικτύων μέσω libvirt και το αυτόματο login στη σελίδα της κονσόλας της εικονικής μηχανής μέσω JavaScript.

6.1 Δημιουργία Κλώνων Εικονικών Μηχανών

Για τη δημιουργία κλώνων εικονικών μηχανών χρησιμοποιήσαμε τη μέθοδο cloneDomain() της οποίας ο κώδικας θα δοθεί παρακάτω. Η συνάρτηση δέχεται ως παράμετρο το UUID (δίνεται από το libvirt) της εικονικής μηχανής που θέλουμε να κλωνοποιηθεί. Αρχικά ελέγχεται η κατάσταση της εικονικής μηχανής αυτής, εάν δεν υπάρχει και εάν τρέχει, μέσω του libvirt API. Εάν δεν ισχύει κανένα από τα δύο τότε ξεκινάμε μια νέα διεργασία μέσω Java η οποία τρέχει το εργαλείο γραμμής εντολών "virt-clone". Έπειτα μέσω του ExpectIt API παίρνουμε το αποτέλεσμα της εκτέλεσης αυτής της εντολής, ώστε να καταλάβουμε εάν ο κλώνος δημιουργήθηκε με επιτυχία ή αν εμφανίστηκε σφάλμα και το επιστρέφουμε.

```
public String cloneDomain(String uuid) throws IOException,
InterruptedException, LibvirtException{

      Domain toClone = getDomainByUUID(uuid);
      if (toClone == null) return "false"; //domain to clone does not exist

      if (toClone.isActive() == 1) return "false"; //domain to clone should
be shutdown

      Process process = Runtime.getRuntime().exec("virt-clone --original " +
      toClone.getName() + " --auto-clone");

        Expect expect = new ExpectBuilder()
                .withInputs(process.getInputStream())
                .withOutput(process.getOutputStream())
                .withTimeout(20000, TimeUnit.SECONDS)
                .withEchoInput(System.out)
                .withEchoOutput(System.out)
                .withExceptionOnFailure()
                .build();

        try {
             expect.expect(regexp("\n")).getBefore();
             expect.expect(regexp("\n")).getBefore();
             String res = expect.expect(regexp("\n")).getBefore();
```

```
            // expect the process to finish
            expect.expect(eof());
            // finally is omitted
            process.waitFor();

           if (res.contains("ERROR")) return "false";  // failed cloning
           else {
            String vmname = res.split(" ")[1].replace("'", "");
            System.out.println(vmname);
            return vmname;
           }
        } catch (InterruptedException e) {
           // Handle exception that could occur when waiting
           // for a spawned process to terminate
            System.out.println(e);

        } finally {
           expect.close();
        }

      return "false";
}
```

6.2 Εκτέλεση Εντολών Εντός της Εικονικής Μηχανής

Για την εκτέλεση εντολών εντός της εικονικής μηχανής μέσω Java, χρησιμοποιήσαμε τη μέθοδο executeCommand(), η οποία δέχεται ως παράμετρο το όνομα του εικονικού μηχανήματος και την εντολή που επιθυμούμε να εκτελεστεί μέσα στο μηχάνημα αυτό. Αρχικά, ξεκινά μία SSH σύνδεση με το host εικονικό μηχάνημα, μέσω της βιβλιοθήκης JSch. Μέσα από την SSH σύνδεση εκτελείται η εντολή “virsh console” στον host μέσω του ExpectIt API, ώστε να αποκτηθεί πρόσβαση στη σειριακή κονσόλα της εικονικής μηχανής. Για την αλληλεπίδραση με την κονσόλα χρησιμοποιείται το ExpectIt API. Στέλνετε το όνομα του χρήστη για να γίνει το login στην κονσόλα και έπειτα εκτελείται η επιθυμητή εντολή στην κονσόλα. Πριν την επιστροφή του αποτελέσματος της εντολής, γίνεται exit από την κονσόλα. Παρατηρήσαμε ότι εάν η μέθοδος αυτή διακοπτόταν για να καλεστεί ξανά από κάποια άλλη διεργασία (πχ. λόγω κάποιας ασύγχρονης κλήσης), η εκτέλεσή της δεν ολοκληρωνόταν. Για αυτό τον λόγο ορίσαμε τη μέθοδο να είναι synchronized ώστε μόνο μία διεργασία να έχει πρόσβαση κάθε φορά στη μέθοδο αυτή.

```
public String executeCommand(String vmname, String cmd) throws JSchException,
IOException {

        JSch jSch = new JSch();
        Session session = jSch.getSession("username", "IPaddress", port);
        session.setPassword("pass");

        Properties config = new Properties();
        config.put("StrictHostKeyChecking", "no");
        session.setConfig(config);
```

```
        session.connect();
        System.out.println("Session Connected!");
        Channel channel = session.openChannel("shell");

        String console_conn = "virsh console "+ vmname +" --force";
        String res = "Failed to execute command inside domain";

        Filter filter = chain(removeColors(), removeNonPrintable());
             Expect expect = new ExpectBuilder()
                 .withOutput(channel.getOutputStream())
                 .withInputs(channel.getInputStream(),
                    channel.getExtInputStream())
                 .withEchoInput(System.out)
                 .withEchoOutput(System.err)
                 .withInputFilters(filter)
                 .withTimeout(20000, TimeUnit.SECONDS)
                 .withExceptionOnFailure()
                 .build();
        channel.connect();

        try {
            expect.expect(regexp(":~\\$ "));
            expect.sendLine(console_conn);
            expect.expect(contains("\n")).getBefore();
            String domainState = expect.expect(contains("\n")).getBefore();

            if (!domainState.contains("Connected to domain")){
                    //domain shutted down
             return res;

            }

            expect.sendLine();
            String tmp = expect.expect(contains(":")).getBefore();
             if (tmp.contains("login")){
                    expect.sendLine("root");
                    expect.expect(regexp("\nroot@:")).getBefore();
               }

               expect.expect(regexp("~ \\#")).getBefore();
               expect.sendLine(cmd);
               res = expect.expect(regexp("\nroot@:~ \\#")).getBefore();

            expect.sendLine("exit");

        } finally {
            channel.disconnect();
            session.disconnect();
            expect.close();
        }
        return res;
    }
```

6.3 Εικονικοποιημένα Δίκτυα

Για να τη δημιουργία των εικονικοποιημένων δικτύων μέσω libvirt χρειάστηκε να ορίσουμε τους διάφορους τρόπους δικτύωσης σε XML μορφή. Παρακάτω δίνεται η XML μορφή των δικτυακών τρόπων που χρησιμοποιήθηκαν στην εφαρμογή.

Το element ρίζας είναι το network και απαιτείται για όλα τα εικονικά δίκτυα. Το element name ορίζει το όνομα που θα έχει το εικονικό δίκτυο στο libvirt. Το attribute stp ενεργοποιεί το Spanning Tree Protocol. Το element forward επιδεικνύει ότι το εικονικό δίκτυο θα συνδεθεί με το φυσικό τοπικό δίκτυο (LAN) και το attribute mode ορίζει τη μέθοδο προώθησης (οι τρόποι δικτύωσης στο libvirt επεξηγήθηκαν αναλυτικά στην παράγραφο 2.2.3.3 Εικονικοποιημένα Δίκτυα στο libvirt). Το attribute address του element ip δηλώνει την IP διεύθυνση (IPv4 ή IPv6) του virtual network switch που συνδέεται με το συγκεκριμένο εικονικό δίκτυο. Για τους guests αυτή θα είναι η προεπιλεγμένη τους πύλη. Το element dhcp ενεργοποιεί την υπηρεσία DHCP και το attribute range ορίζει το εύρος των IP διευθύνσεων που θα αποδοθούν στους DHCP clients.

- Εσωτερικό Δίκτυο (Internal Network)

```
<network>
   <name> internal </name>
   <bridge name='virbr0' stp='on' delay='0'/>
</network>
```

- Απομονωμένο Δίκτυο (Isolated / Host-only Network)

```
<network>
   <name> isolated </name>
   <bridge name='virbr1'/>
   <ip address="192.168.152.1" netmask="255.255.255.0">
         <dhcp>
            <range start="192.168.152.2" end="192.168.152.254"/>
         </dhcp>
   </ip>
   <ip family="ipv6" address="2001:db8:ca2:3::1" prefix="64"/>
</network>
```

- Δίκτυο NAT (NAT Network)

```
<network>
   <name> nat </name>
   <bridge name='virbr2'/>
   <forward mode='nat'/>
   <ip address="192.168.122.1" netmask="255.255.255.0">
         <dhcp>
            <range start="192.168.122.2" end="192.168.122.254"/>
         </dhcp>
   </ip>
   <ip family="ipv6" address="2001:db8:ca2:2::1" prefix="64"/>
</network>
```

- Δίκτυο Δρομολόγησης (Routed Network)

```
<network>
   <name> routed </name>
   <bridge name='virbr3'/>
```

```
    <forward mode='route'/>
    <ip address="192.168.167.1" netmask="255.255.255.0">
          <dhcp>
             <range start="192.168.167.2" end="192.168.167.254"/>
          </dhcp>
    </ip>
</network>
```

Το εσωτερικό δίκτυο είναι ένα πολύ απομονωμένο δίκτυο καθώς δεν υπάρχει τρόπος επικοινωνίας με τον host μέσω αυτού του δικτύου (δεν ορίστηκαν ip elements). Παρ' όλ' αυτά, αυτό το εικονικό δίκτυο μπορεί να χρησιμοποιηθεί για την επικοινωνία μεταξύ των guests που ανήκουν στο δίκτυο αυτό.

6.4 Αυτόματο login στη Σελίδα της Κονσόλας της Εικονικής Μηχανής

Όταν ο χρήστης άνοιγε τη σελίδα με την κονσόλα της εικονικής μηχανής, η κονσόλα περίμενε να δοθεί ένα enter και έπειτα ο χρήστης να κάνει login δίνοντας το σωστό όνομα. Αυτό δεν θα ήταν μια καλή υλοποίηση καθώς ο χρήστης θα μπορούσε να περιμένει για αρκετά λεπτά νομίζοντας ότι η κονσόλα δεν ανταποκρίνεται. Επίσης δεν γνωρίζει το όνομα για να κάνει login. Χρειάστηκε λοιπόν μία επικοινωνία με το client-side του shellinabox.

Το shellinabox δέχεται μηνύματα σε μορφή JSON που έχουν πεδία type και data. Στέλνοντας ένα μήνυμα με type 'input' στο shellinabox, γράφεται το περιεχόμενο του πεδίου data στον εξομοιωτή τερματικού. Θέλουμε δηλαδή να στείλουμε δύο JSON μηνύματα με type 'input', το ένα με data '\n' και το άλλο με data το όνομα για login.

Όμως δεν αρκεί μόνο αυτό, τα μηνύματα αυτά πρέπει να στέλνονται κάθε φορά που το τερματικό προτρέπει τον χρήστη να κάνει login. Χρειάζεται, δηλαδή, να ακούμε και τα μηνύματα που φτάνουν στο shellinabox τα οποία έχουν type 'output', ώστε τα μηνύματα αυτά να στέλνονται μόνο όταν ληφθεί το κατάλληλο μήνυμα (πχ. 'login:').

Για να το πετύχουμε αυτό χρησιμοποιήσαμε τη γλώσσα JavaScript, ενώ για να ακούμε τα JSON μηνύματα χρησιμοποιήσαμε έναν eventListener. Ένα κομμάτι του κώδικα δίνεται παρακάτω.

```
// Receive response from shellinabox
window.addEventListener("message", function(message) {

      // Allow messages only from shellinabox
      if (message.origin !== (hostname || url)) {
            return;
```

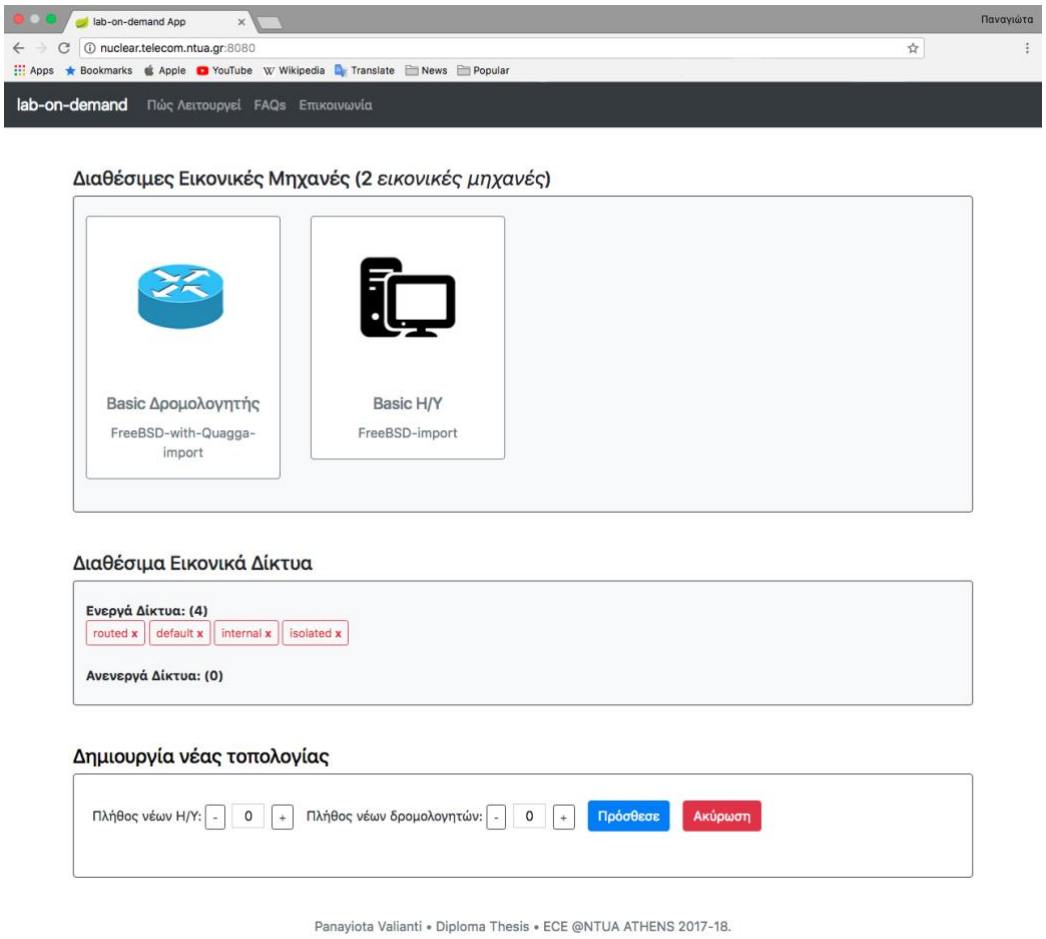
```
        }
        // Handle response according to response type
        var decoded = JSON.parse(message.data);
        switch (decoded.type) {
        case "ready":
                // Shellinabox is ready to communicate and we will enable console
output
                // by default.
                var message = JSON.stringify({
                        type : 'output',
                        data : 'enable'
                });
                iframe.contentWindow.postMessage(message, url);
                break;
        case "output":
                // Append new output
                if (decoded.data.toLowerCase().includes("\nescape character is
^]")) {
                        // Send enter to shellinabox
                        var message = JSON.stringify({
                                type : 'input',
                                data : '\n'
                        });
                        iframe.contentWindow.postMessage(message, url);
                }
                if (decoded.data.includes("\nlogin:")) {
                        // Send login to shellinabox
                        var message = JSON.stringify({
                                type : 'input',
                                data : 'root\n'
                        });
                        iframe.contentWindow.postMessage(message, url);
                }
                break;
        case "session" :
                // Reload session status
                session.innerHTML = 'Session status: ' + decoded.data;
                break;
        }

}, false);
```

6.5 Σενάριο Χρήσης Εφαρμογής

Σε αυτό το υποκεφάλαιο περιγράφεται αναλυτικά ένα ολοκληρωμένο σενάριο χρήσης της εφαρμογής lab-on-demand, μέσω του οποίου θα γίνει ευκολότερη η κατανόηση των λειτουργιών και της χρησιμότητάς της, ενώ παράλληλα παρουσιάζονται και στιγμιότυπα της εφαρμογής. Θα περιγραφεί ο τρόπος δημιουργίας νέας εικονικοποιημένης τοπολογίας δικτύου καθώς και η πρόσβαση στη γραμμή εντολών των εικονικών μηχανών.

Για να εμφανιστεί η αρχική σελίδα στη διεπαφή του χρήστη, δεδομένου ότι η εφαρμογή ήδη τρέχει σε έναν διακομιστή (ακολουθώντας τις οδηγίες εγκατάστασης που δίνονται στο κεφάλαιο Εγκατάσταση), εκτελούμε την παρακάτω διαδικασία. Ανοίγουμε έναν φυλλομετρητή

ιστού (πχ. Google Chrome) και μεταβαίνουμε στη URL διεύθυνση http://*myIPaddress*:8080/, όπου *myIPaddress* η IP διεύθυνση της διεπαφής του διακομιστή ιστού στον οποίο τρέχει η εφαρμογή. Στον φυλλομετρητή θα εμφανιστεί η αρχική σελίδα της εφαρμογής (Σχήμα 26).

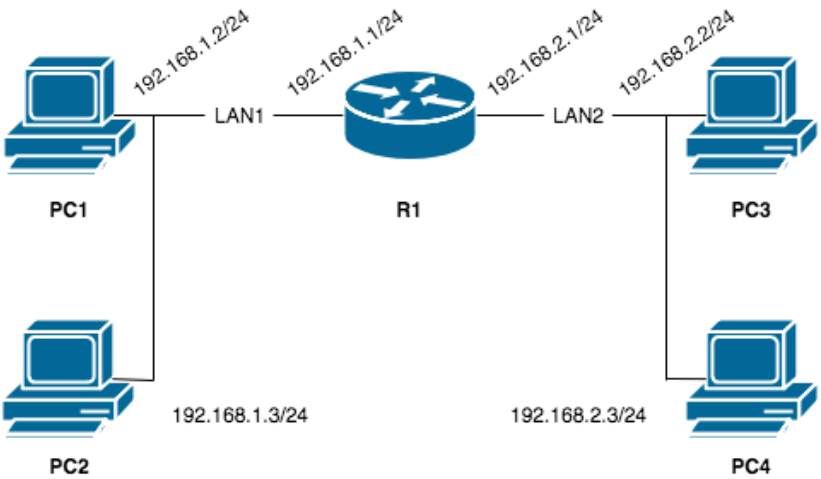


Σχήμα 26: Η αρχική σελίδα της εφαρμογής όπως εμφανίζεται στον φυλλομετρητή

Στην κορυφή της αρχικής σελίδας εμφανίζονται πληροφορίες για τις διαθέσιμες εικονικές μηχανές. Όπως παρατηρούμε αρχικά εμφανίζονται μόνο οι «βασικές» εικόνες στις οποίες δεν έχει πρόσβαση ο χρήστης ούτε μπορεί να τις διαχειριστεί. Έπειτα εμφανίζονται πληροφορίες για τα διαθέσιμα εικονικά δίκτυα, τα οποία μπορεί να διαγράψει εάν το επιθυμεί. Στο κατώτερο μέρος της σελίδας ο χρήστης μπορεί να δημιουργήσει μια νέα τοπολογία.

6.5.1 Εκτέλεση Σεναρίου

Αρχικά, θα περιγράψουμε τη διαδικασία δημιουργίας νέας εικονικοποιημένης τοπολογίας δικτύου. Υποθέτουμε ότι θέλουμε να δημιουργήσουμε ένα απλό δίκτυο με δρομολογητή, η τοπολογία του οποίου απεικονίζεται στο Σχήμα 27.



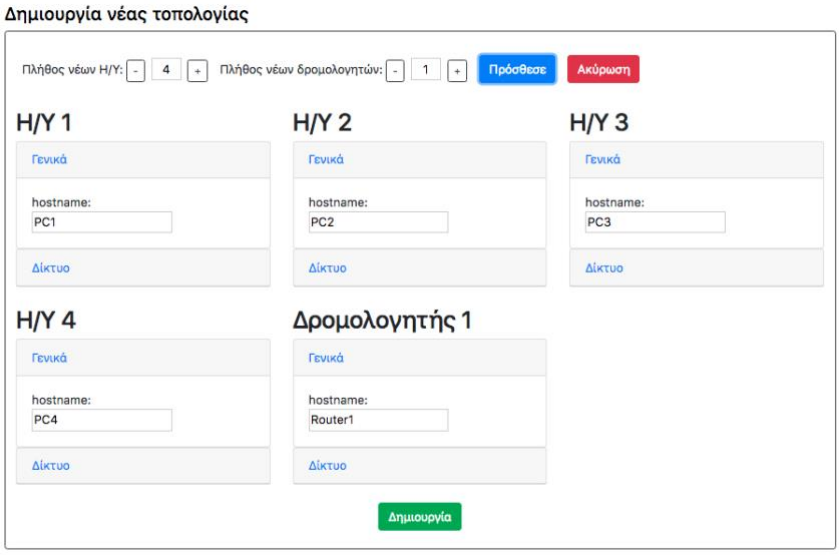
Σχήμα 27: Παράδειγμα τοπολογίας δικτύου

Θα πρέπει λοιπόν να δημιουργηθούν τέσσερις Η/Υ και ένας δρομολογητής. Ο δρομολογητής θα πρέπει να έχει δύο κάρτες δικτύου, ενώ όλοι οι Η/Υ από μία. Όλες οι κάρτες δικτύου θα πρέπει να βρίσκονται σε κατάσταση εσωτερικής (internal) δικτύωσης και θα ονομάσουμε τα δύο τοπικά δίκτυα ως “LAN1” και “LAN2”.

Αφού μεταβούμε στην αρχική σελίδα της εφαρμογής, μεταφερόμαστε στο κατώτερο μέρος της σελίδας για να ξεκινήσουμε τη δημιουργία της τοπολογίας. Ακολουθούμε τα παρακάτω απλά βήματα:

1. <u>Επιλογή «βασικών» εικόνων</u>

Επιλέγουμε το πλήθος των Η/Υ και δρομολογητών που θέλουμε να δημιουργηθούν (στο παράδειγμά μας 4 και 1 αντίστοιχα), και πατάμε το κουμπί «Πρόσθεσε».



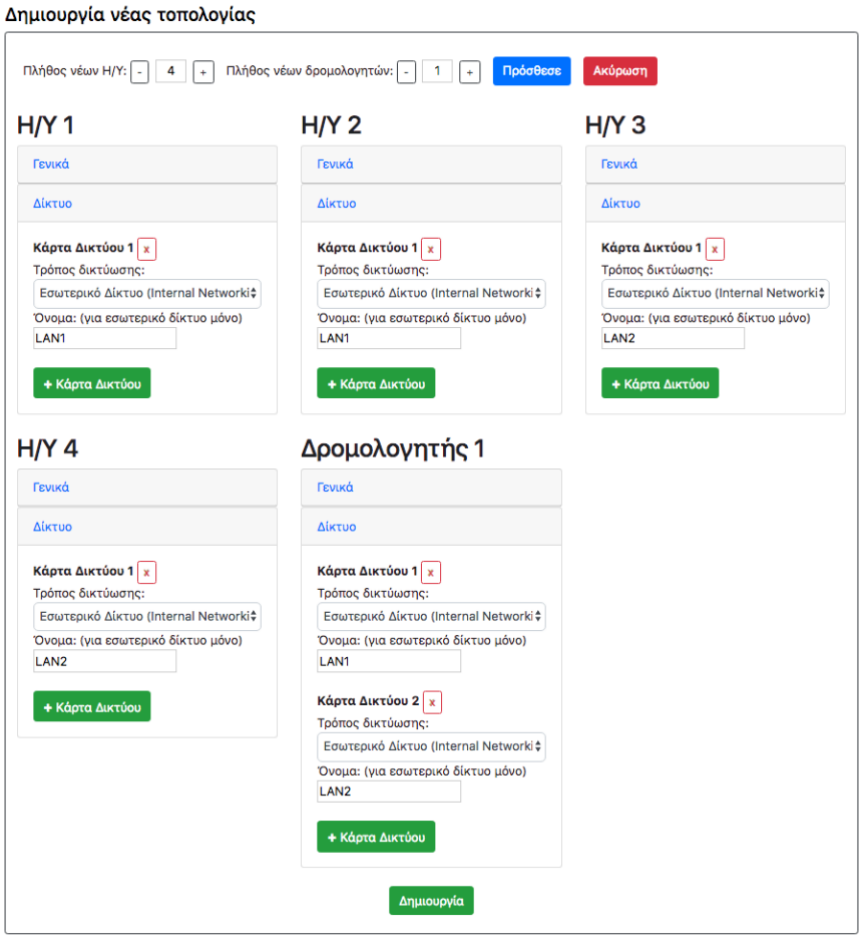
Σχήμα 28: Επιλογή πλήθους Η/Υ και δρομολογητών

Παρατηρούμε ότι στην οθόνη προστέθηκαν καρτέλες σε μορφή “accordion” για την εισαγωγή ρυθμίσεων σε κάθε εικονική μηχανή. Επίσης εμφανίζεται αυτόματα ένα προεπιλεγμένο hostname για κάθε εικονική μηχανή.

2. <u>Εισαγωγή ρυθμίσεων για κάθε εικονική μηχανή</u>

Από την καρτέλα με την ετικέτα «Γενικά», πληκτρολογούμε το hostname που θέλουμε στο text-box που υπάρχει κάτω από την ετικέτα «hostname» για κάθε εικονικό μηχάνημα, είτε αφήνουμε αυτό που επέλεξε η εφαρμογή για εμάς. Αλλάζουμε το hostname του δρομολογητή σε “R1”.

Για να εισάγουμε τις δικτυακές ρυθμίσεις κάνουμε κλικ στην καρτέλα με την ετικέτα «Δίκτυο». Έπειτα, προσθέτουμε τον αριθμό καρτών δικτύου που επιθυμούμε πατώντας το κουμπί «+ Κάρτα Δικτύου».

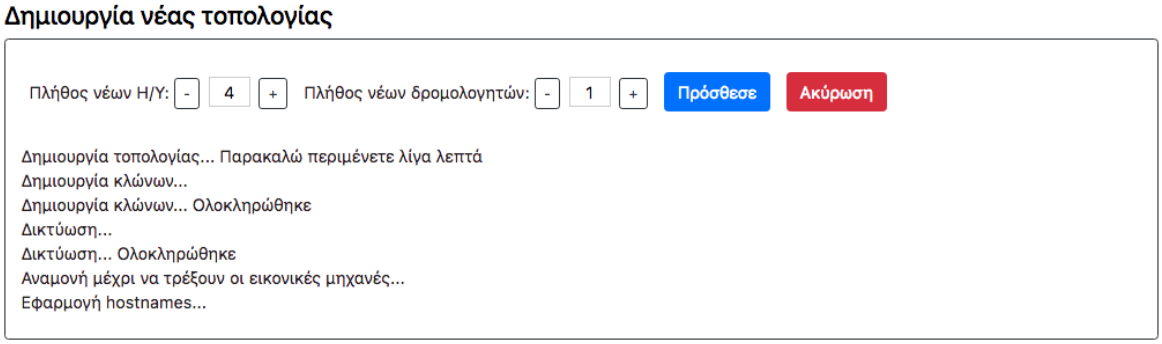


Σχήμα 29: Εισαγωγή δικτυακών ρυθμίσεων σε κάθε εικονική μηχανή

Επιλέγουμε τον επιθυμητό τρόπο δικτύωσης για κάθε κάρτα δικτύου που προσθέσαμε, μέσω της drop-down λίστας. Τέλος αν επιλέξαμε τρόπο δικτύωσης εσωτερικό δίκτυο, πληκτρολογούμε και το επιθυμητό όνομα δικτύου στο text-box κάτω από την ετικέτα «Όνομα».

3. <u>Δημιουργία τοπολογίας δικτύου</u>

Όταν ολοκληρώσουμε τις γενικές και δικτυακές ρυθμίσεις, είμαστε έτοιμοι να δημιουργήσουμε την τοπολογία πατώντας το κουμπί «Δημιουργία» που βρίσκεται στο τέλος της οθόνης.



Σχήμα 30: Εμφάνιση ενημερωτικών μηνυμάτων κατά τη δημιουργία της τοπολογίας δικτύου

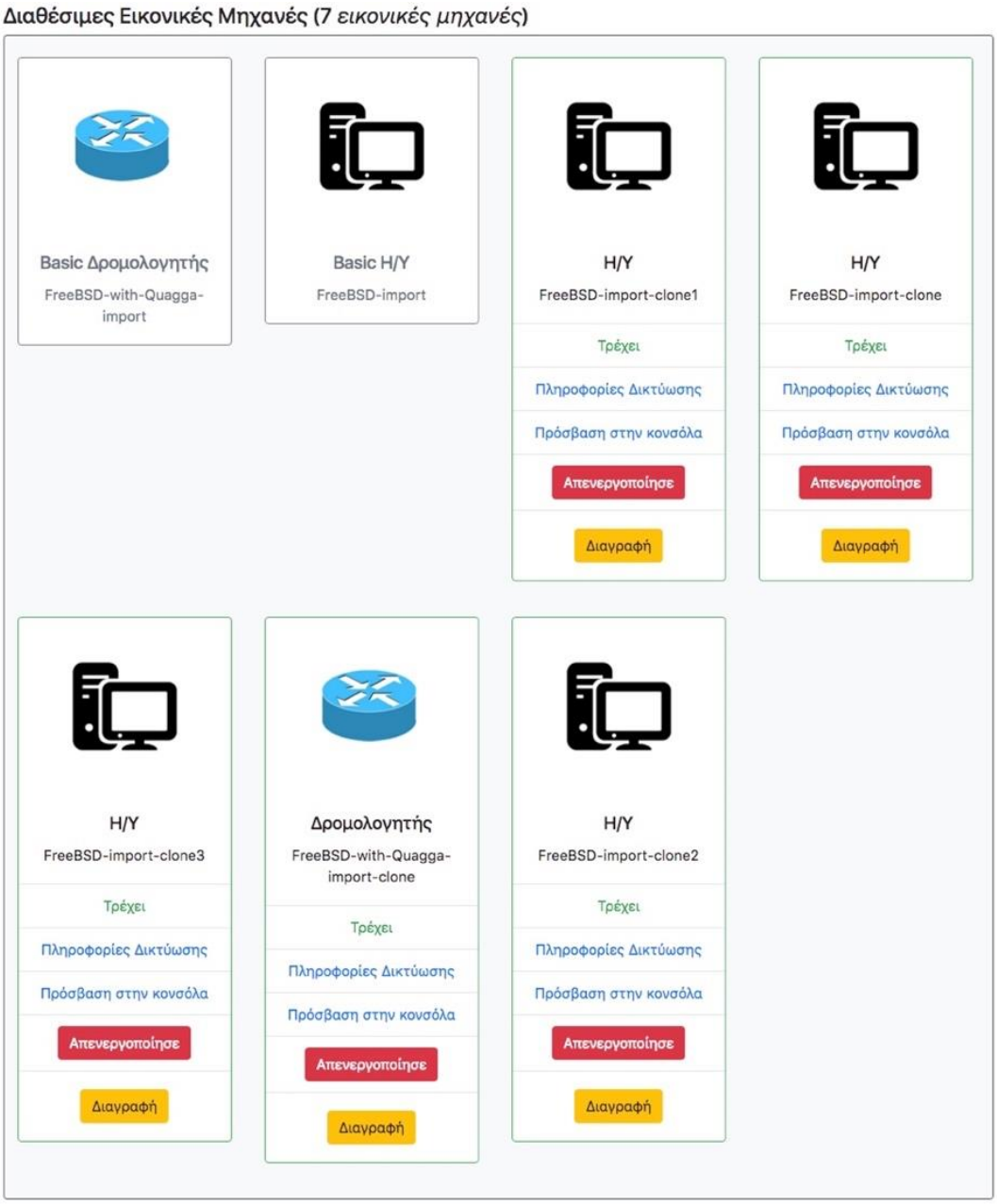
Παρατηρούμε ότι στην οθόνη εμφανίζονται μηνύματα που μας ενημερώνουν για την εξέλιξη της διαδικασίας κατασκευής αλλά και τρεξίματος της νέας τοπολογίας δικτύου.

Όταν η διαδικασία αυτή ολοκληρωθεί εμφανίζεται το ανάλογο ενημερωτικό μήνυμα στην οθόνη.

Σχήμα 31: Ενημερωτικό μήνυμα ολοκλήρωσης κατασκευής της τοπολογίας δικτύου

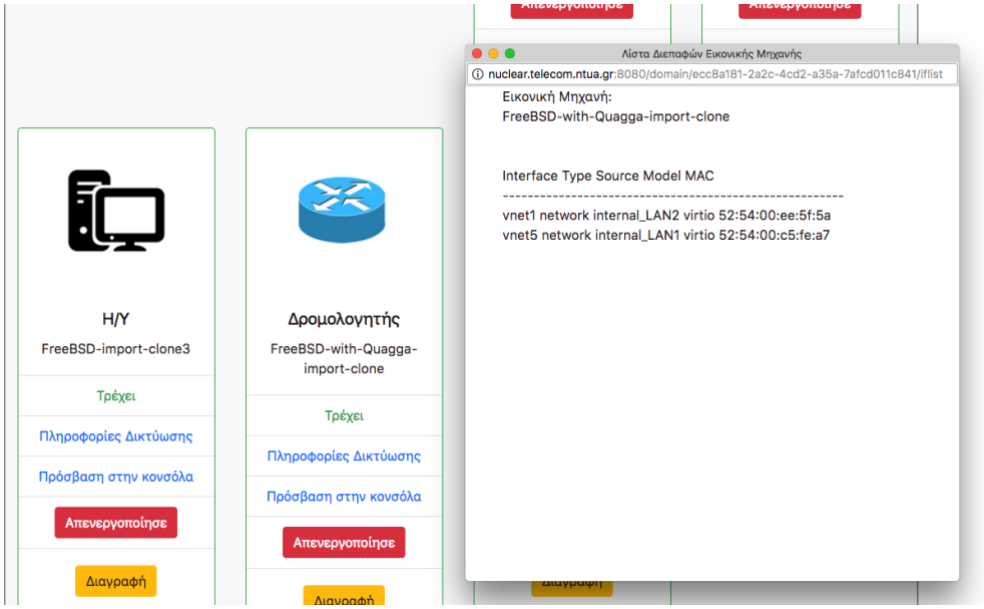
4. <u>Πληροφορίες και διαχείριση εικονικών μηχανών</u>

Παρατηρούμε ότι η λίστα με τις διαθέσιμες εικονικές μηχανές ανανεώθηκε και τώρα περιλαμβάνει και τις δικτυακές συσκευές που δημιουργήσαμε. Όλες οι εικονικές μηχανές τρέχουν και ο χρήστης έχει τη δυνατότητα να απενεργοποιήσει ή να διαγράψει την εικονική μηχανή που επιθυμεί πατώντας τα ανάλογα κουμπιά.



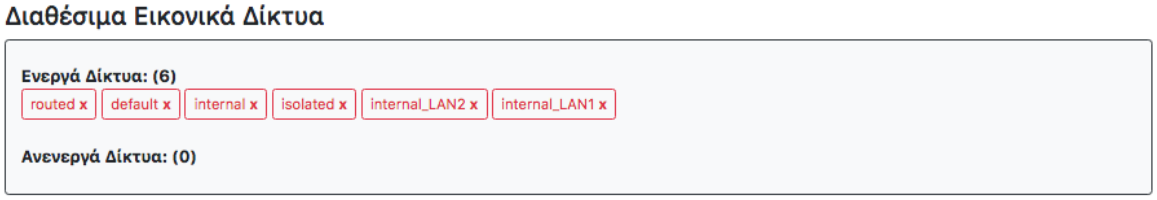
Σχήμα 32: Η λίστα με τις εικονικές μηχανές ανανεώθηκε

Επίσης ο χρήστης μπορεί να μάθει πληροφορίες για τη δικτύωση της εικονικής μηχανής που θέλει πατώντας τον σύνδεσμο «Πληροφορίες Δικτύωσης». Το συγκεκριμένο παράδειγμα απαιτεί τη γνώση της δικτύωσης του δρομολογητή. Επειδή έχει δύο κάρτες δικτύου θα πρέπει να γνωρίζουμε σε ποιο υποδίκτυο ανήκει η κάθε μια, ώστε στη συνέχεια να αντιστοιχηθεί η σωστή IP διεύθυνση στην κάθε διεπαφή. Όταν πατηθεί ο σύνδεσμος οι πληροφορίες εμφανίζονται σε νέο παράθυρο.



Σχήμα 33: Πληροφορίες δικτύωσης του δρομολογητή που δημιουργήσαμε

Παρατηρούμε ότι η πρώτη διεπαφή του δρομολογητή ανήκει στο LAN2 ενώ η δεύτερη στο LAN1.



Σχήμα 34: Η λίστα με τα διαθέσιμα εικονικά δίκτυα ανανεώθηκε

Παρατηρούμε επίσης ότι η λίστα με τα διαθέσιμα εικονικά δίκτυα ανανεώθηκε και τώρα περιλαμβάνει τα νέα δίκτυα, LAN1 και LAN2, που δημιουργήσαμε.

5. <u>Πρόσβαση στη γραμμή εντολών των εικονικών μηχανών</u>

Στη λίστα με τις διαθέσιμες εικονικές μηχανές υπάρχει ένας σύνδεσμος «Πρόσβαση στην κονσόλα» για κάθε εικονική μηχανή που τρέχει. Πατώντας τον ανοίγει ένα νέο tab στον φυλλομετρητή που περιλαμβάνει τη σελίδα της κονσόλας του εικονικού μηχανήματος που επιλέξαμε.

Κονσόλα εικονικής μηχανής: FreeBSD-with-Quagga-import-clone

Ενεργοποίησε | Απενεργοποίησε | Επανασύνδεση | Κατάσταση Σύνδεσης | Κατάσταση σύνδεσης: ???

```
root@R1:~ # ifconfig
vtnet0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=6c07bb<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,JUMBO_MTU,VLAN_HWCSUM,T
SO4,TSO6,LRO,VLAN_HWTSO,LINKSTATE,RXCSUM_IPV6,TXCSUM_IPV6>
        ether 52:54:00:ee:5f:5a
        hwaddr 52:54:00:ee:5f:5a
        inet6 fe80::5054:ff:feee:5f5a%vtnet0 prefixlen 64 scopeid 0x1
        inet 0.0.0.0 netmask 0xff000000 broadcast 255.255.255.255
        nd6 options=23<PERFORMNUD,ACCEPT_RTADV,AUTO_LINKLOCAL>
        media: Ethernet 10Gbase-T <full-duplex>
        status: active
vtnet1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=6c07bb<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,JUMBO_MTU,VLAN_HWCSUM,T
SO4,TSO6,LRO,VLAN_HWTSO,LINKSTATE,RXCSUM_IPV6,TXCSUM_IPV6>
        ether 52:54:00:c5:fe:a7
        hwaddr 52:54:00:c5:fe:a7
        inet6 fe80::5054:ff:fec5:fea7%vtnet1 prefixlen 64 scopeid 0x2
        inet 0.0.0.0 netmask 0xff000000 broadcast 255.255.255.255
        nd6 options=23<PERFORMNUD,ACCEPT_RTADV,AUTO_LINKLOCAL>
        media: Ethernet 10Gbase-T <full-duplex>
        status: active
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
        options=600003<RXCSUM,TXCSUM,RXCSUM_IPV6,TXCSUM_IPV6>
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x3
        inet 127.0.0.1 netmask 0xff000000
        nd6 options=23<PERFORMNUD,ACCEPT_RTADV,AUTO_LINKLOCAL>
```

Σχήμα 35: Εκτέλεση εντολής ifconfig εντός του δρομολογητή που δημιουργήσαμε

Παρατηρούμε ότι το hostname που ορίσαμε εμφανίζεται στη γραμμή εντολών. Επίσης υπάρχουν κουμπιά για τον έλεγχο της εικονικής μηχανής, αλλά και για τον έλεγχο της σύνδεσης με τον δαίμονα του εξομοιωτή τερματικού (shellinaboxd).

Τώρα μπορούμε να αντιστοιχίσουμε τις IP διευθύνσεις στην κάθε συσκευή μέσω της γραμμής εντολών της κάθε συσκευής και έτσι να ολοκληρώσουμε την επιθυμητή τοπολογία.

Από τη γραμμή εντολών των Η/Υ εκτελούμε τις παρακάτω εντολές.

Στο PC1: ifconfig vtnet0 192.168.1.2/24

Στο PC2: ifconfig vtnet0 192.168.1.3/24

Στο PC3: ifconfig vtnet0 192.168.2.2/24

Στο PC4: ifconfig vtnet0 192.168.2.3/24

Από τη γραμμή εντολών του δρομολογητή εκτελούμε τις εντολές:

ifconfig vtnet0 192.168.2.1/24

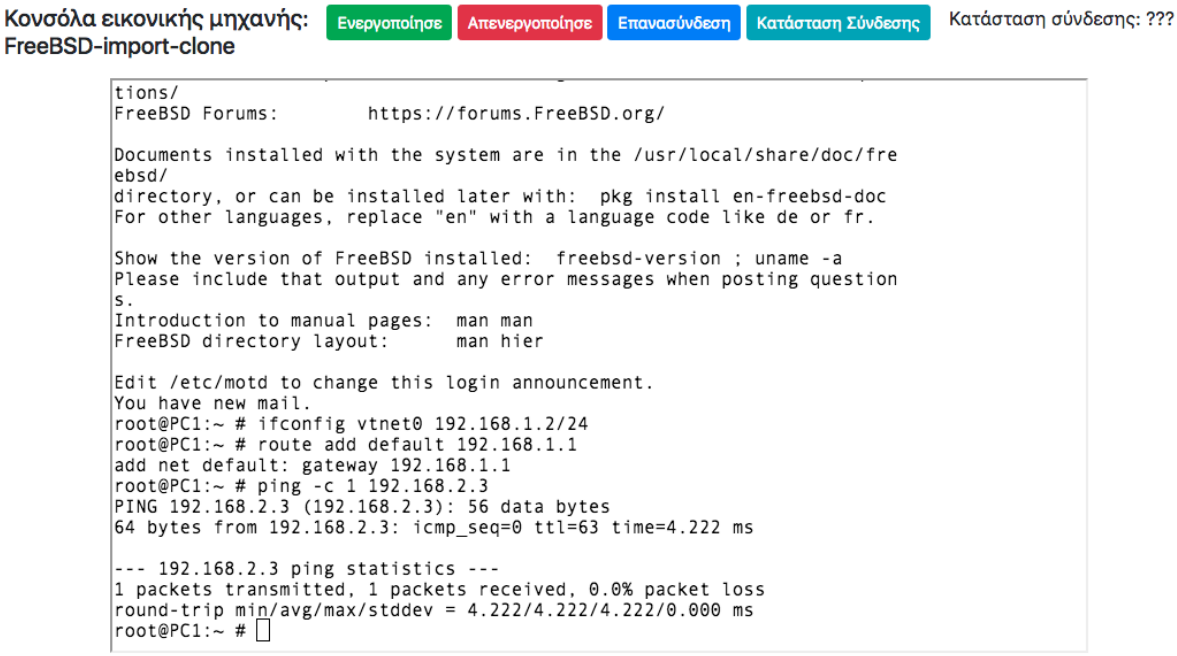
ifconfig vtnet1 192.168.1.1/24

Επίσης για να υπάρχει επικοινωνία μεταξύ LAN1 και LAN2, θέτουμε στους Η/Υ ως προεπιλεγμένη πύλη (default gateway) τον δρομολογητή. Από τη γραμμή εντολών τους εκτελούμε τις παρακάτω εντολές.

Στα PC1 και PC2: route add default 192.168.1.1

Στα PC3 και PC4: route add default 192.168.2.1

Τώρα μπορούμε να ελέγξουμε ότι η τοπολογία δικτύου που δημιουργήσαμε δουλεύει σωστά χρησιμοποιώντας την εντολή ping. Από το PC1 κάνουμε ping στο PC4, εκτελώντας την εντολή "ping -c 1 192.168.2.3" από τη γραμμή εντολών του PC1.



Σχήμα 36: Το ping από το PC1 στο PC4 έγινε με επιτυχία

7. Εγκατάσταση Εφαρμογής

Σε αυτό το κεφάλαιο περιγράφεται η διαδικασία εγκατάστασης της εφαρμογής και του προαπαιτούμενου λογισμικού/εργαλείων σε έναν διακομιστή. Αναλύεται επίσης ένας περιορισμός στο λειτουργικό σύστημα του διακομιστή.

Οι οδηγίες εγκατάστασης που ακολουθούν αφορούν Ubuntu μηχανήματα και δοκιμάστηκαν σε Ubuntu 16.04 μηχάνημα.

7.1 Περιορισμοί

Ο διακομιστής στον οποίο πρόκειται να εγκατασταθεί η εφαρμογή, θα πρέπει να έχει Linux λειτουργικό σύστημα ώστε το εργαλείο διαχείρισης εικονικών μηχανών, libvirt, να μπορέσει να δουλεύει ως διακομιστής (το libvirt δουλέυει ως πελάτης και σε άλλα λειτουργικά συστήματα όπως Microsoft Windows).

7.2 Προαπαιτούμενα

Ο διακομιστής θα πρέπει να έχει εγκατεστημένα τα ακόλουθα λογισμικά/εργαλεία πριν εγκατασταθεί η εφαρμογή.

7.2.1 Java και Apache Maven

Ανοίξτε ένα τερματικό (terminal) και ακολουθήστε τα βήματα.
Αρχικά, ελέγξτε εάν η Java είναι ήδη εγκατεστημένη στο μηχάνημα με την εντολή

```
java -version
```

Σε περίπτωση που δεν είναι εγκατεστημένη, εκτελέστε τα ακόλουθα βήματα για να την εγκαταστήσετε.

Πρώτα, προσθέστε το PPA της Oracle, μετά κάντε update το package repository.

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
```

Μετά εκτελέστε την ακόλουθη εντολή για να εγκαταστήσετε την Java (Oracle JDK 8).

```
sudo apt-get install oracle-java8-installer
```

Ελέγξτε ότι η Java εγκαταστάθηκε, χρησιμοποιώντας την εντολή που δώσαμε προηγουμένως και έπειτα προχωρήστε στην εγκατάσταση του Apache Maven.

Εκτελέστε την εντολή για να δείτε όλα τα διαθέσιμα Maven πακέτα.

```
apt-cache search maven
```

Το maven πακέτο πάντα περιλαμβάνει την τελευταία έκδοση του Apache Maven.

Εκτελέστε την εντολή για να εγκαταστήσετε την τελευταία έκδοση του Apache Maven.

```
sudo apt-get install maven
```

Επιβεβαιώστε ότι η εγκατάσταση έγινε με επιτυχία χρησιμοποιώντας την ακόλουθη εντολή.

```
mvn -version
```

7.2.2 libvirt και QEMU-KVM

Κρατήστε ανοιχτό το terminal είτε ανοίξτε ένα νέο και ακολουθήστε τα βήματα.

Η βιβλιοθήκη libvirt χρησιμοποιείται σε συνδυασμό με διάφορες τεχνολογίες εικονικοποίησης. Πριν εγκαταστήσετε το libvirt είναι καλύτερο να σιγουρευτείτε ότι το υλικό (hardware) σας υποστηρίζει τις απαραίτητες επεκτάσεις εικονικοποίησης για το KVM. Εκτελέστε την ακόλουθη εντολή.

```
kvm-ok
```

Το μήνυμα που θα τυπωθεί θα σας ενημερώνει εάν η ΜΚΕ (CPU) σας υποστηρίζει ή δεν υποστηρίζει εικονικοποίηση υλικού.

Για να εγκατασταθούν τα απαιτούμενα πακέτα εκτελέστε αυτή την εντολή.

```
sudo apt install qemu-kvm libvirt-bin virtinst
```

7.2.2.1 Εγκατάσταση Αρχικών / «Βασικών» Εικόνων

Είμαστε έτοιμοι τώρα να εγκαταστήσουμε τις αρχικές / «βασικές» εικόνες, μέσω του εργαλείου γραμμής εντολών του libvirt API, virt-install.

Θεωρούμε δεδομένο ότι στον κατάλογο /home/pval/vms υπάρχουν τα αρχεία εικόνων FreeBSD-11.1-RELEASE-i386.qcow2[26] και FreeBSD-11.1-RELEASE-i386-clone.qcow2[27] τα αποία αντιστοιχούν σε Η/Υ και δρομολογητή.

Από το τερματικό εκτελέστε τις ακόλουθες εντολές.

```
virt-install –name FreeBSD-import –memory 256 –disk /home/pval/vms/FreeBSD-11.1-RELEASE-i386.qcow2 --import
```

```
virt-install –name FreeBSD-with-Quagga-import –memory 256 –disk /home/pval/vms/FreeBSD-11.1-RELEASE-i386-clone.qcow2 --import
```

Αντικαταστήστε τον κατάλογο /home/pval/vms και τα ονόματα των αρχείων με αυτά που σας αντιστοιχούν.

7.2.3 shellinabox[28]

Αρχικά, κατεβάζουμε τις εξαρτήσεις εκτελώντας την ακόλουθη εντολή σε ένα τερματικό.

```
apt-get install git libssl-dev libpam0g-dev zlib1g-dev dh-autoreconf
```

Κατεβάζουμε τα αρχεία κώδικα και μετακινόμαστε στον κατάλογο του project με αυτή την εντολή.

```
git clone https://github.com/shellinabox/shellinabox.git && cd shellinabox
```

Τρέξτε τα autotools στον κατάλογο του project.

```
autoreconf -i
```

Τρέχτε το configure και make στον κατάλογο του project.

```
./configure && make
```

Θα χρειαστεί ένα αρχείο που ονομάζεται “certificate.pem” το οποίο περιλαμβάνει τόσο το ιδιωτικό κλειδί όσο και το δημόσιο πιστοποιητικό σε μορφή PEM (δηλαδή ASCII).

---

[26] Η εικόνα αυτή κατεβάστηκε από το Official Site του FreeBSD Project: https://www.freebsd.org/where.html
[27] Είναι κλώνος του FreeBSD-11.1-RELEASE-i386.qcow2 μαζί με εγκατάσταση του Quagga: https://www.quagga.net/
[28] https://github.com/shellinabox/shellinabox

Σημείωση: ο κώδικας του shellinabox θα ήταν καλό να βρίσκεται κάτω από τον κατάλογο /tmp ώστε να έχει τα κατάλληλα δικαιώματα και να τρέχει σωστά.

7.3 Εγκατάσταση

Η εγκατάσταση της εφαρμογής στον διακομιστή είναι πάρα πολύ απλή, αφού απαιτείται η εκτέλεση μόνο μιας εντολής στο τερματικό, δεδομένου ότι τα προαπαιτούμενα λογισμικά/εργαλεία που αναφέρθηκαν προηγουμένως είναι ήδη εγκατεστημένα.

Για να τρέξουμε την εφαρμογή στον διακομιστή, εκτελούμε την παρακάτω εντολή αφού ορίσουμε ως τρέχοντα κατάλογο εργασίας (current working directory) τον κατάλογο lab-on-demand στον οποίο βρίσκεται ο κώδικας της εφαρμογής.

```
mvn spring-boot:run
```

Για να αλλάξουμε κατάλογο εργασίας εκτελούμε αυτή την εντολή.

```
cd path-to-project/lab-on-demand
```

όπου το path-to-project είναι το μονοπάτι (path) που οδηγεί στον κατάλογο του project της εφαρμογής lab-on-demand.

8. Επίλογος

8.1 Σύνοψη και Συμπεράσματα

Στην παρούσα διπλωματική εργασία σχεδιάστηκε και υλοποιήθηκε μια εφαρμογή ιστού, η οποία επιτρέπει την κατασκευή αλλά και την παραμετροποίηση μιας εικονικοποιημένης τοπολογίας δικτύου με απλό και γρήγορο τρόπο.

Οι τεχνολογίες και τα εργαλεία που επιλέχθηκαν για την ανάπτυξη της εφαρμογής μας έδωσαν απεριόριστες δυνατότητες, απλοποίησαν και μείωσαν τη συγγραφή κώδικα αλλά και αυτοματοποίησαν τη διαδικασία εγκατάστασης και τρεξίματος της εφαρμογής. Για την εφαρμογή χρειάστηκε μια μεγάλη ποικιλία τεχνολογιών, από τεχνολογίες εικονικοποίησης και εργαλεία διαχείρισης εικονικών μηχανών, τεχνολογίες ανάπτυξης εφαρμογών Παγκόσμιου Ιστού, εργαλεία «χτισίματος» κώδικα, μέχρι συστήματα διαχείρισης εκδόσεων λογισμικού.

Η ανάλυση των απαιτήσεων και των προδιαγραφών της εφαρμογής μας βοήθησε να ξεκαθαρίσουμε τα ζητήματα που θα έπρεπε να υλοποιηθούν. Το στάδιο σχεδίασης της αρχιτεκτονικής ήταν πολύ σημαντικό για την εξασφάλιση της απλότητας, της ταχύτητας και της απόδοσης της εφαρμογής. Η υλοποίηση της εφαρμογής έγινε κυρίως με τη γλώσσα προγραμματισμού Java, τη χρήση της οποίας διευκόλυνε το web framework Spring.

Η εφαρμογή μπορεί να βοηθήσει φοιτητές προπτυχιακών μαθημάτων δικτύων υπολογιστών, καθώς και να χρησιμοποιηθεί για δοκιμή διαφόρων δικτυακών τοπολογιών, πριν την κατασκευή των φυσικών δικτύων και την αγορά φυσικών δικτυακών συσκευών.

8.2 Μελλοντικές Επεκτάσεις

Η εφαρμογή που αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας θα μπορούσε στο μέλλον να επεκταθεί ώστε να προσφέρει ακόμη περισσότερες δυνατότητες δημιουργίας αλλά και παραμετροποίησης εικονικοποιημένων δικτυακών τοπολογιών στον χρήστη.

Μία πιθανή επέκταση θα μπορούσε να είναι η προσθήκη περισσότερων δικτυακών συσκευών στο σύνολο των αρχικών «βασικών» εικόνων, όπως firewall και switch. Επίσης, θα μπορούσε οι

αρχικές εικόνες να προσφέρουν διάφορα λειτουργικά συστήματα ανά δικτυακή συσκευή, για παράδειγμα έναν ηλεκτρονικό υπολογιστή με Ubuntu και έναν ηλεκτρονικό υπολογιστή με FreeBSD. Στην ίδια λογική, θα μπορούσαν να προστεθούν κι άλλοι τρόποι δικτύωσης, όπως bridge networking. Το σετ ρυθμίσεων θα μπορούσε να εμπλουτιστεί με ακόμη περισσότερες ρυθμίσεις, όπως ρυθμίσεις για τον ορισμό DNS server, για τον ορισμό προεπιλεγμένης πύλης και για την απόδοση IP διευθύνσεων στα εικονικά μηχανήματα. Η εφαρμογή μας έχει τη δυνατότητα να υποστηρίξει επιπλέον ρυθμίσεις στο επίπεδο του λειτουργικού συστήματος. Στο Back-End κομμάτι αναπτύξαμε μια Java μέθοδο που εκτελεί ό,τι εντολή επιθυμούμε μέσα στα εικονικά μηχανήματα. Το μόνο που υπολείπεται είναι η διαμόρφωση του Front-End κώδικα. Έτσι ο χρήστης θα μπορεί να δημιουργήσει μια πιο προχωρημένη εικονικοποιημένη δικτυακή τοπολογία.

Μία άλλη λειτουργικότητα που θα μπορούσε να προστεθεί στην εφαρμογή είναι η δυνατότητα παραμετροποίησης της δικτύωσης ενός εικονικού μηχανήματος που έχει ήδη δημιουργηθεί.

Η εφαρμογή θα μπορούσε να υποστηρίζει τη διαπίστευση χρηστών (user authentication), μέσω μηχανισμού εισόδου και τη δικαιοδοσία χρηστών (user authorization), ώστε ο κάθε χρήστης να μπορεί να χρησιμοποιεί και να διαχειρίζεται μόνο τις εικονικές συσκευές που έφτιαξε ο ίδιος.

Ένα άλλο σημείο που θα μπορούσε να βελτιωθεί είναι η ασφάλεια της εφαρμογής. Θα μπορούσε να προστεθούν πιστοποιητικά ώστε να η εφαρμογή να υποστηρίζει το πρωτόκολλο HTTPS.

Τα σενάρια επέκτασης είναι απεριόριστα και μπορούν να βελτιώσουν την εμπειρία χρήσης του χρήστη. Παρ’ όλ’ αυτά, η εφαρμογή που υλοποιήθηκε καλύπτει σε μεγάλο βαθμό τις ανάγκες του χρήστη.

Βιβλιογραφία / Παραπομπές

[1] G. Popek, R. Goldberg, *Formal requirements for virtualizable third generation architectures*. Commun. ACM, pp. 412–421, 1974.

[2] R. Goldberg, "Architectural Principles for Virtual Computer Systems", Ph.D. thesis, Harvard University, Cambridge, MA, 1972.

[3] D. Mishchenko, *VMware ESXi: Planning, Implementation, and Security Cengage*, 1st ed. Clifton Park: Cengage Learning, 2010.

[4] A. Kivity, Y.Kamay, D. Laor, U. Lublin, A. Liguori, "kvm: the Linux Virtual Machine Monitor", in *Proceedings of the Linux Symposium*, 2007.

[5] F. Bellard, "QEMU, a Fast and Portable Dynamic Translator", in *Usenix annual technical conference*, 2005.

[6] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, "Xen and the Art of Virtualization", in *Proceedings of the ACM Symposium on Operating Systems Principles*, University of Cambridge, Computer Laboratory, pp. 164-177, 2003.

[7] libvirt Official Site, https://libvirt.org/, Ιούλιος 2018.

[8] T. Jones, 'Anatomy of the libvirt virtualization library', 2010, https://www.ibm.com/developerworks/library/l-libvirt/index.html, Ιούλιος 2018

[9] VirtualBox technical documentation, Oracle Corporation, https://www.virtualbox.org/wiki/VirtualBox, Ιούλιος 2018.

[10] Web application Wikipedia, https://en.wikipedia.org/wiki/Web_application, Ιούλιος 2018.

[11] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, *Hypertext transfer protocol--HTTP/1.1*. No. RFC 2616, 1999.

[12] D. Crockford, *The application/json media type for javascript object notation (json)*. No. RFC 4627, 2006.

[13] D. Raggett, A. Hors, I. Jacobs, *HTML 4.01 Specification*. W3C recommendation 24, 1999.

[14] D. Flanagan. *JavaScript: the definitive guide*. Sebastopol: O'Reilly, 2006.

[15] jQuery Library, https://jquery.com/, Ιούνιος 2018.

[16] MIT License, http://opensource.org/licenses/MIT, Ιούνιος 2018.

[17] N. Zakas, *Professional JavaScript for web developers*. Hoboken (N.J.): Wiley, 2012.

[18] Bootstrap Front-End Framework, https://getbootstrap.com/, Ιούνιος 2018.

[19] Twitter, https://twitter.com/, Ιούνιος 2018.

[20] GitHub, https://github.com/, Ιούνιος 2018.

[21] D. Bovet, M. Cesati, *Understanding the Linux Kernel: From I/O Ports to Process Management*. Sebastopol: O'Reilly: 2005.

[22] GNU General Public Licence, http://www.gnu.org/copyleft/gpl.html, Ιούνιος 2018.

[23] The Linux Kernel Archives, https://www.kernel.org/, Ιούλιος 2018.

[24] Java Official Site, https://www.oracle.com/java/index.html, Ιούλιος 2018.

[25] K. Arnold, J. Gosling, D. Holmes, *The Java Programming Language*, 4th ed. AddisonWesley Professional, 2005.

[26] B. Stroustrup, *The C++ Programming Language*, 4th ed. New Jersey: Pearson Education (US), 2013.

[27] Spring Framework Official Site, https://spring.io/, Ιούλιος 2018.

[28] Spring Framework Wikipedia, https://en.wikipedia.org/wiki/Spring_Framework, Ιούλιος 2018.

[29] Expect Official Site, https://core.tcl.tk/expect/index?name=Expect, Ιούλιος 2018.

[30] D. Libes, *Exploring Expect: A Tcl-based Toolkit for Automating Interactive Programs*. Sebastopol: O'Reilly, 1994.

[31] Tcl Official Site, http://www.tcl.tk/, Ιούλιος 2018.

[32] M. J. Bach, *The Design of the UNIX Operating System*. Upper Saddle River: Pearson Education (US), 1986.

[33] Microsoft Windows Wikipedia, https://en.wikipedia.org/wiki/Microsoft_Windows, Ιούλιος 2018.

[34] ExpectIt API, https://github.com/Alexey1Gavrilov/ExpectIt, Ιούνιος 2018.

[35] Maven, https://maven.apache.org/, Ιούνιος 2018.

[36] Maven Central Repository, https://search.maven.org/, Ιούνιος 2018.

[37] Shell In A Box, https://code.google.com/archive/p/shellinabox/, Ιούνιος 2018.

[38] Git, https://git-scm.com/, Ιούνιος 2018.

[39] Linus Torvalds, https://github.com/torvalds, Ιούνιος 2018.

[40] Git E-book, https://git-scm.com/book/en/v2, Ιούνιος 2018.

[41] VirtualBox, https://www.virtualbox.org/, Ιούνιος 2018.