ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής

Υλοποίηση υποδομής για την αυτοματοποίηση δοκιμών σε περιβάλλον NFV

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Κωνσταντίνου Π. Τσιρώνη

Επιβλέπων: **Ευστάθιος Συκάς**
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2019

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής

Υλοποίηση υποδομής για την αυτοματοποίηση δοκιμών σε περιβάλλον NFV

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Κωνσταντίνου Π. Τσιρώνη

Επιβλέπων: **Ευστάθιος Συκάς**
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 6η Μαρτίου 2019……

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

Αθήνα, Μάρτιος 2019

.....

ΚΩΝΣΤΑΝΤΙΝΟΣ ΤΣΙΡΩΝΗΣ

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

Περίληψη

Η συνεχής άνοδος των δικτύων καθοριζόμενων από λογισμικό (Software Defined Networks - SDN) καθιστούν τη σχεδίαση μιας υπηρεσίας δικτύου μια εργασία πιο δύσκολη από ποτέ. Η επιλογή των παραδοσιακών στοιχείων δικτύου που προσφέρουν την καλύτερη σχέση κόστους/ωφέλειας, δεδομένων των απαιτήσεων απόδοσης και του προϋπολογισμού που έχει διατεθεί, δεν είναι η μόνη λύση σήμερα. Η εισαγωγή τεχνολογιών εικονικοποίησης έχει στρέψει το ενδιαφέρον των τηλεπικοινωνιακών παρόχων στην προσπάθεια για ελαχιστοποίηση των χρόνων ενεργοποίησης νέων υπηρεσιών και των υποδομών τους. Με τον όρο Εικονικοποίηση Λειτουργιών Δικτύου (Network Functions Virtualization - NFV) εννοούμε την αρχιτεκτονική δικτύου στην οποία οι απαραίτητες λειτουργίες δικτύου εκτελούνται ως VNFs σε εικονικές μηχανές αντί των παραδοσιακών υλοποιήσεων σε φυσικές δικτυακές συσκευές.

Σκοπός της παρούσας διπλωματικής εργασίας είναι η παρουσίαση και μελέτη των εργαλείων ανοιχτού κώδικα που μπορούν να υποστηρίξουν αυτοματοποιημένες δοκιμές και η ανάπτυξη μιας τέτοιας δομής, χρησιμοποιώντας μερικά από τα παραπάνω εργαλεία για τη δοκιμή της τεχνολογίας εικονικοποίησης λειτουργιών δικτύου καθώς και η δημιουργία μιας διεπαφής για το χρήστη. Επίσης, θα γίνει αξιολόγηση της δομής με βάση την απόδοση, τη συμβατότητα της υποδομής καθώς και κάποιων βασικών λειτουργιών. Όλα τα μηχανήματα θα βρίσκονται στο εργαστηριακό περιβάλλον του Εργαστηρίου Δικτύου Υπολογιστών της Σχολής ΗΜΜΥ του ΕΜΠ.

Λέξεις Κλειδιά: Εικονικοποίηση Λειτουργιών Δικτύου, NFV, VNF, Elasticsearch, Logstash, Kibana, Ansible, Selenium, Virtual Machine, Raspberry Pi, ESXi Server

Abstract

The constant rise of Software Defined Networks (SDN) makes designing a network service a job more difficult than ever before. Choosing the traditional network components that offer the best cost / benefit ratio, given the performance requirements and the budget that is allocated, is not the only solution today. The introduction of virtualization technologies has turned the interest of telecommunication providers to minimizing the times of activation of new services and their infrastructures. Network Function Virtualization (NFV) means the network architecture in which the necessary network functions are executed as VNFs on virtual machines instead of traditional implementations on physical network devices.

Aim of this diploma thesis is to present and study open source tools that can support automated testing and the development of such a structure, using some of the above tools to test network virtualization technology and to create a user interface. It will also evaluate the structure based on performance, infrastructure compatibility as well as some basic functions. All the machines will be in the laboratory environment of the Computer Network Laboratory of the ECE School of the NTUA.

Keywords: Virtualization of Network Functions, NFV, VNF, Elasticsearch, Kibana, Ansible, Virtual Machine, Raspberry Pi, ESXi Server

Ευχαριστίες

Η παρούσα διπλωματική εργασία αναπτύχθηκε και ολοκληρώθηκε κατά το 10ο και 11ο εξάμηνο των σπουδών μου τα έτη 2018-19, ως αποτέλεσμα της συνεργασίας μου με τον Τομέα Επικοινωνιών, Ηλεκτρονικής & Συστημάτων Πληροφορικής της Σχολής Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου.

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της εργασίας αυτής, κ. Ευστάθιο Συκά για την άψογη συνεργασία που είχαμε κατά το διάστημα της ανάπτυξης και συγγραφής της διπλωματικής καθώς και για την ευκαιρία που μου έδωσε να εργαστώ πάνω σε ένα τόσο ενδιαφέρον θέμα. Επιπλέον, θέλω να απονείμω ιδιαίτερες ευχαριστίες στον υποψήφιο διδάκτορα, κ. Πάρι Χαραλάμπου για την υποστήριξη που μου παρείχε καθ' όλη τη διάρκεια εκπόνησης της εργασίας μου. Η ανά πάσα στιγμή διαθεσιμότητά του και η βοήθεια του ήταν καθοριστικές για την ολοκλήρωσή της και χωρίς τη βοήθειά του το έργο μου θα ήταν σαφώς δυσκολότερο.

Επίσης, θα ήθελαν να ευχαριστήσω την οικογένειά μου, τους γονείς μου και την αδερφή μου, που με στήριξαν καθ' όλη τη διάρκεια εκπόνησης της εργασίας αλλά και όλα μου τα φοιτητικά και μαθητικά ακόμη χρόνια της ζωής μου. Οι συμβουλές τους αλλά και η ενθάρρυνση που μου έδιναν όταν έβλεπαν ότι αντιμετώπιζα δυσκολίες ήταν πολύ σημαντικές τόσο για την επίτευξη των στόχων μου αλλά και για την ανάπλαση του χαρακτήρα που έχω σήμερα.

Τέλος, θέλω να ευχαριστήσω τους φίλους μου τόσο από τη γειτονιά μου όσο και αυτούς που έκανα μέσα στη σχολή για τις αξέχαστες αναμνήσεις που μου δημιούργησαν όλα αυτά τα φοιτητικά μου χρόνια. Τους εύχομαι υγεία, ευτυχία, δημιουργικότητα και πολλές επιτυχίες, τους είμαι ευγνώμων σε όλους και στον καθένα ξεχωριστά.

Πίνακας Περιεχομένων

Πίνακας Εικόνων

Πίνακας Πινάκων

1

Εισαγωγή

1.1 Οι περιορισμοί του παρελθόντος

Η σημαντική εξάρτηση των δικτύων από το υποκείμενο υλικό τους και η ύπαρξη διαφόρων εξειδικευμένων συσκευών υλικού στην υποδομή δικτύου, όπως π.χ. τείχη προστασίας, εξοπλισμό βαθιάς επιθεώρησης πακέτων (Deep Packet Inspection - DPI) και δρομολογητές, έχουν κλιμακώσει τις προκλήσεις που αντιμετωπίζουν οι πάροχοι υπηρεσιών δικτύου. Επιπλέον, οι μειωμένοι κύκλοι ζωής αυτών των τύπων υλικού εξαιτίας του γρήγορου ρυθμού καινοτομίας τείνουν να πολλαπλασιάσουν τις επενδύσεις (Capital Expenditures - CAPEX) και τις λειτουργικές δαπάνες (Operational Expenditures - OPEX). Η τεχνολογία εικονικοποίησης λειτουργίας δικτύου αναπτύχθηκε για να εκμεταλλευτεί την εξέλιξη της τεχνολογίας εικονικοποίησης. Το NFV είναι η τεχνολογία της μεταφοράς λειτουργιών δικτύου από συσκευές ειδικού εξοπλισμού σε εφαρμογές που βασίζονται σε λογισμικό που λειτουργούν με εξοπλισμό COTS (Commercial Off-The-Shelf). Αυτές οι εφαρμογές εκτελούνται και ενοποιούνται σε τυποποιημένες πλατφόρμες πληροφορικής όπως διακομιστές μεταγωγείς και συστήματα αποθήκευσης. Μέσω του NFV, οι λειτουργίες δικτύου μπορούν να εφαρμοστούν σε διάφορες τοποθεσίες, κυρίως σε κέντρα δεδομένων (Data Centers). Αναλυτικά οι περιορισμοί που αντιμετωπίζουν οι τηλεπικοινωνιακοί πάροχοι είναι:

- Η ζήτηση για **μείωση των κεφαλαιουχικών και λειτουργικών δαπανών** που ώθησε τους ειδικούς της τεχνολογίας πληροφοριών (Information Technology - IT) να εξετάσουν σχέδια για την επίτευξη αποτελεσματικότερων επενδύσεων με υψηλότερη απόδοση κεφαλαίου. Προς το σκοπό αυτό,

η τεχνολογία εικονικοποίησης εμφανίστηκε ως ένας τρόπος αποσύνδεσης των εφαρμογών λογισμικού από το υποκείμενο υλικό που επιτρέπει στο λογισμικό να τρέχει σε ένα εικονικό περιβάλλον. Σε αυτό το εικονικό περιβάλλον, το υλικό είναι εξομοιωμένο και το λειτουργικό σύστημα (Operational System - OS) τρέχει πάνω από το εξομοιούμενο υλικό σαν να τρέχει με δικούς του απομονωμένους πόρους. Χρησιμοποιώντας αυτή τη διαδικασία, πολλαπλές εικονικές μηχανές μπορούν να μοιράζονται τους διαθέσιμους πόρους και να εκτελούνται ταυτόχρονα σε μια ενιαία φυσική μηχανή. Επιπλέον, ο ρυθμός ανάπτυξης των παγκόσμιων επενδύσεων στην τεχνολογία των κέντρων δεδομένων καθιστά όλο και πιο ελκυστική την ανάπτυξη των λειτουργιών δικτύου μέσω λογισμικού που εκτελείται σε τυπικούς διακομιστές υψηλού όγκου (Standard High Volume - SHV).

- Η ζήτηση για **σύνδεση ευρυζωνικού δικτύου** με όλο και μεγαλύτερες ταχύτητες πρόσβασης την τελευταία δεκαετία. Αποκτά πρόσθετη δυναμική με την αύξηση του αριθμού των κινητών συσκευών που είναι συνδεδεμένα στο Internet, από smartphones, tablets και φορητούς υπολογιστές, μέχρι και δίκτυα αισθητήρων και συνδεσιμότητα Machine-to-Machine (M2M). Επιπλέον, το πλήθος των συνδεδεμένων συσκευών αναμένεται να αυξηθεί και από την εισαγωγή τεχνολογιών όπως το 5G και το IoT (Internet of Things). Αυτή η αυξανόμενη ζήτηση ωθεί τους παρόχους υπηρεσιών δικτύου να επενδύσουν σε υποδομές για να συμβαδίσουν με τη ζήτηση, παρόλο που οι μελέτες δείχνουν ότι η απόδοση αυτών των επενδύσεων είναι ελάχιστη. Οι δαπάνες δικτύου εξαρτώνται σε μεγάλο βαθμό από την απαραίτητη υποδομή. Το υψηλό κόστος κάθε αναβάθμισης βελτίωσης δικτύου ή νέας έκδοσης υπηρεσίας περιορίζει το περιθώριο κέρδους του παρόχου υπηρεσιών.

- Οι προκλήσεις λειτουργίας του δικτύου που δεν περιορίζονται στο κόστος δαπανηρών συσκευών υλικού, αλλά περιλαμβάνουν επίσης το **αυξανόμενο κόστος ενέργειας** και την ανταγωνιστική αγορά για προσωπικό υψηλής εξειδίκευσης με τις δεξιότητες που απαιτούνται για το σχεδιασμό, την ενοποίηση και τη λειτουργία μιας όλο και πιο πολύπλοκης υποδομής βασισμένης στο υλικό.

- Η **διαχείριση της υποδομής δικτύου** που είναι ένα επιπλέον σημαντικό μέλημα των παρόχων υπηρεσιών. Το ζήτημα αυτό δεν επηρεάζει μόνο τα έσοδα, αλλά αυξάνει και το χρόνο εισαγωγής νέων υπηρεσιών στην αγορά και περιορίζει την καινοτομία στον κλάδο των τηλεπικοινωνιών. Επομένως, οι φορείς εκμετάλλευσης δικτύων προσπαθούν να ελαχιστοποιήσουν ή και να εξαλείψουν την εξάρτησή τους από το ιδιόκτητο υλικό [3].

Το NFV παρέχει πολλά οφέλη στον κλάδο των τηλεπικοινωνιών καθώς πρόκειται για μια καινούρια και επαναστατική τεχνολογία που βασίζεται κατεξοχήν σε εικονικές δομές. Ορισμένα από αυτά τα οφέλη είναι η ανάπτυξη των πλατφορμών με πρότυπα ανοιχτού κώδικα, η δυνατότητα επεκτασιμότητας, η ευελιξία, η βελτίωση των λειτουργικών επιδόσεων, οι βραχύτεροι κύκλοι ανάπτυξης και οι μειωμένες επενδύσεις CAPEX και OPEX. Το NFV συνεπάγεται την υλοποίηση λειτουργιών δικτύου που διατίθενται σήμερα σε ιδιόκτητα μεσαία κιβώτια και υλικό εξοπλισμού δικτύου στο λογισμικό. Αυτές οι λειτουργίες εικονικού δικτύου (VNFs) μπορούν να αναπτυχθούν σε βιομηχανικούς διακομιστές βασικών προϊόντων, αποθηκευτικούς χώρους και διακόπτες. Το NFV

επιτρέπει στους CSP να αξιοποιήσουν τις τεχνολογίες εικονικοποίησης και αυτοματοποίησης σύννεφων. Όπως συμβαίνει και με το "Cloudification of IT Services", ο μετασχηματισμός NFV όχι μόνο επιτρέπει την ευέλικτη ανάπτυξη των υπηρεσιών δικτύου αλλά και βελτιώνει την ελαστική ζήτηση για την εξάπλωση και διακόπτει τον αποκλεισμό του πωλητή υλικού καθώς τα VNFs είναι φορητές σε διαφορετικές πλατφόρμες υλικού [6].

Συνεπώς, ο σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μια εφαρμογής για την εκτέλεση αυτοματοποιημένων δοκιμών για υπηρεσίες που βασίζονται στην τεχνολογία NFV. Με αυτόν τον τρόπο θα γίνει κατανοητό πώς μπορεί η τεχνολογία αυτή να αντικαταστήσει τις συσκευές που βασίζονται στο υλικό και θα αλλάξει τον τρόπο με τον οποίο λειτουργούν. Τα οφέλη από τη χρήση της συγκεκριμένης τεχνολογίας είναι πολλαπλά και στόχος της συγκεκριμένης εργασίας είναι η ανάλυση και η παρουσίαση της δυνατότητας για αυτοματοποιημένες δοκιμές σαν ένα παράδειγμα νέων δυνατοτήτων. Ουσιαστικά, αφορούν την αντιμετώπιση όλων αυτών των περιορισμών που αναφέραμε στις προηγούμενες παραγράφους και τη μετατροπή τους σε κέρδος για τους παρόχους. Επίσης με την εισαγωγή αυτοματοποιημένων δοκιμών για νέες υπηρεσίες είναι δυνατή η χρησιμοποίηση τεχνολογιών NFV για την ταχύτερη εισαγωγή νέων υπηρεσιών. Παράλληλα, στην παρούσα διπλωματική θα γίνει αναλυτική περιγραφή της διαδικασίας ανάπτυξης της εφαρμογής αυτής, καθώς και παρουσίαση των πιο σημαντικών εργαλείων ελευθέρου λογισμικού που μπορούν να χρησιμοποιηθούν στην ανάπτυξή της, ενώ στο τέλος θα γίνει μια προσπάθεια αξιολόγησής της και εξαγωγής των απαραίτητων συμπερασμάτων και προοπτικών για μέλλον.

1.2 Οργάνωση Κειμένου

Η παρούσα διπλωματική εργασία αποτελείται από έξι (6) κεφάλαια. Στο παρόν κεφάλαιο, δηλαδή στο Κεφάλαιο 1, παρουσιάζεται αναλυτικά το πρόβλημα που προσπαθεί να αντιμετωπίσει η εργασία, καθώς και οι ανάγκες που συνετέλεσαν στη δημιουργία του συγκεκριμένου θέματος. Επίσης, γίνεται και μια σύντομη παρουσίαση του αντικείμενου της.

Στο Κεφάλαιο 2 παρουσιάζονται εκτενώς οι απαραίτητες έννοιες με στόχο τη δημιουργία του κατάλληλου υπόβαθρου για την ευκολότερη κατανόηση του αντικειμένου της εργασίας από τον αναγνώστη. Συγκεκριμένα, αναλύονται οι έννοιες της εικονικοποίησης δικτυακών λειτουργιών, αλλά και των αυτοματοποιημένων τεχνικών δοκιμής.

Στο Κεφάλαιο 3 παρουσιάζεται θεωρητικά η αρχιτεκτονική του δικτύου που σχεδιάστηκε στα πλαίσια της εργασίας. Πιο συγκεκριμένα, γίνεται ανάλυση του ρόλου και της σημασίας των επιμέρους μονάδων που αποτελούν το δίκτυο. Παρουσιάζονται τα διαθέσιμα εργαλεία ανοικτού κώδικα, καθώς και εκείνα που επιλέχθηκαν για κάθε μονάδα του δικτύου. Μερικά από τα εργαλεία που αναλύονται είναι το *Selenium* για την αυτοματοποίηση της περιήγησης στο διαδίκτυο, το *Ansible* για την αυτοματοποίηση της ρύθμισης συστημάτων και το πλαίσιο web εφαρμογών Django.

Στο Κεφάλαιο 4 γίνεται διεξοδική ανάλυση της διαδικασίας που ακολουθήθηκε για την ανάπτυξη της εφαρμογής που απαιτούσε το θέμα της παρούσας διπλωματικής εργασίας, δηλαδή την web-εφαρμογή για την αυτοματοποιημένη δοκιμή της εικονικοποίησης δικτυακών λειτουργιών. Παρουσιάζεται βήμα προς βήμα η διαδικασία δημιουργίας μαζί με τις απαραίτητες επεξηγήσεις, καθώς και ο τρόπος αυτοματοποίησης των δοκιμών.

Στο Κεφάλαιο 5 επιχειρείται η αξιολόγηση της εφαρμογής που αναπτύχθηκε. Συγκεκριμένα, παρουσιάζονται αρχικά οι παράμετροι και το σύστημα αξιολόγησης και στη συνέχεια τα αποτελέσματα των πειραμάτων και τα εξαγόμενα συμπεράσματα.

Το Κεφάλαιο 6 αποτελεί τον επίλογο της εργασίας. Εδώ συνοψίζονται τα συμπεράσματα από το σύνολό της και παρουσιάζονται οι προκλήσεις που πρέπει να αντιμετωπιστούν και οι πιθανές μελλοντικές προεκτάσεις μετά την ολοκλήρωσή της.

Στο τέλος της παρούσας διπλωματικής εργασίας παρατίθεται η βιβλιογραφία που χρησιμοποιήθηκε κατά τη συγγραφή της και το παράρτημα, το οποίο περιέχει τον κώδικα που αναπτύχθηκε για το παρόν θέμα.

2

Θεωρητικό Υπόβαθρο

Σε αυτό το κεφάλαιο αναλύονται οι βασικές τεχνολογίες που χρησιμοποιήθηκαν κατά την εκπόνηση της διπλωματικής, αλλά και οι θεωρητικές έννοιες που αφορούν τη θεματολογία της. Συγκεκριμένα θα επεξηγηθεί η έννοια της εικονικοποίησης λειτουργιών δικτύων, της λειτουργίας και χρησιμότητάς της, καθώς και των πλεονεκτημάτων που προσφέρει. Επίσης, θα παρουσιαστεί η έννοια της αυτοματοποίησης των δοκιμών, οι διάφορες τεχνικές της, καθώς και τα πλεονεκτήματα που προσφέρει έναντι της παραδοσιακής χειροκίνητης δοκιμής.

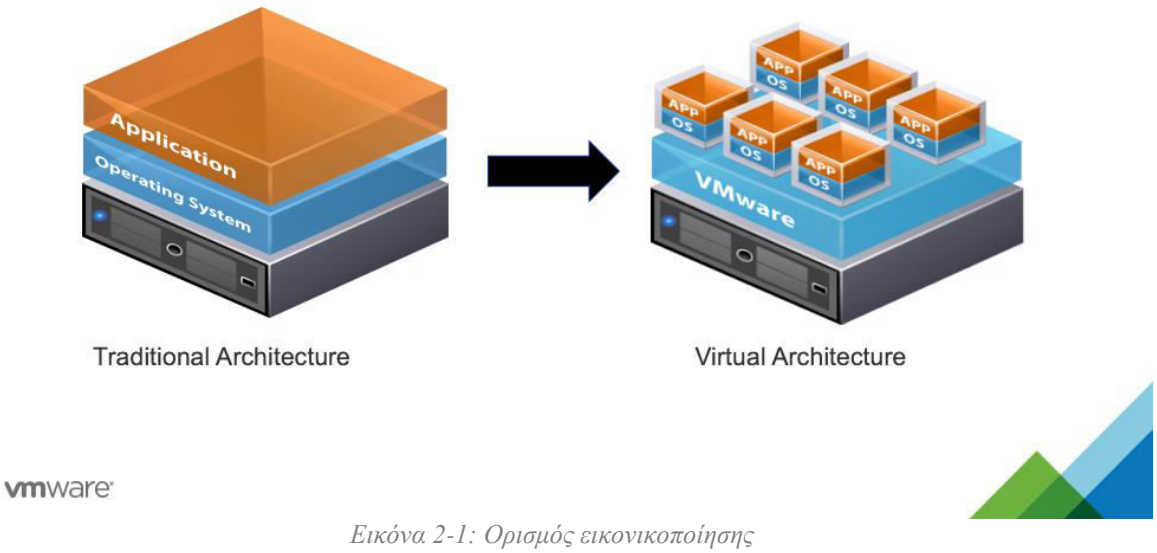
2.1 Εικονικοποίηση Δικτυακών Λειτουργιών (NFV)

2.1.1 Εισαγωγή στην Εικονικοποίηση

Ο όρος εικονικοποίηση (virtualization) είναι ένας ευρύς όρος των υπολογιστικών συστημάτων που αναφέρεται σε έναν μηχανισμό αφαίρεσης στοχευμένο στην απόκρυψη λεπτομερειών της υλοποίησης και της κατάστασης ορισμένων υπολογιστικών πόρων από πελάτες των πόρων αυτών (π.χ. εφαρμογές, άλλα συστήματα, χρήστες κλπ). Η εν λόγω αφαίρεση μπορεί είτε να αναγκάζει έναν πόρο να συμπεριφέρεται ως πλειάδα πόρων (π.χ. μία συσκευή αποθήκευσης σε διακομιστή τοπικού δικτύου), είτε πολλαπλούς πόρους να συμπεριφέρονται ως ένας (π.χ. συσκευές αποθήκευσης σε κατανεμημένα συστήματα). Η εικονικοποίηση δημιουργεί μία εξωτερική διασύνδεση η οποία αποκρύπτει την υποκείμενη υλοποίηση (π.χ. πολυπλέκοντας την πρόσβαση από διαφορετικούς χρήστες). Αυτή η προσέγγιση στην εικονικοποίηση αναφέρεται ως εικονικοποίηση πόρων. Μία άλλη

προσέγγιση, ίδιας όμως νοοτροπίας, είναι η εικονικοποίηση πλατφόρμας, όπου η αφαίρεση που επιτελείται προσομοιώνει ολόκληρους υπολογιστές.

Η πιο γνωστή μορφή εικονικοποίησης είναι η πλήρης εικονικοποίηση. Σε αυτή, η εικονική μηχανή προσομοιώνει επαρκές τμήμα του πραγματικού υποκείμενου υλικού ώστε να επιτρέπει την εκτέλεση ενός μη τροποποιημένου, φιλοξενούμενου λειτουργικού συστήματος σχεδιασμένου για τον ίδιο τύπο επεξεργαστή με την πραγματική CPU (π.χ. VirtualPC, VMware, Win4Lin κλπ). Στην πλήρη εικονικοποίηση δεν χρειάζεται εξομοίωση του συνόλου εντολών του επεξεργαστή και μάλιστα ένα τμήμα του κώδικα του φιλοξενούμενου λειτουργικού συστήματος μπορεί να εκτελείται απευθείας από το υλικό, χωρίς μεσολάβηση του επόπτη, αρκεί να μην επηρεάζει υποσυστήματα εκτός του άμεσου ελέγχου του τελευταίου. Τα κρίσιμα σημεία του φιλοξενούμενου 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 επόπτη 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 μια 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.



Εικόνα 2-1: Ορισμός εικονικοποίησης

Άλλες μορφές εικονικοποίησης είναι η εξομοίωση και η παρα-εικονικοποίηση. Στην εξομοίωση η εικονική μηχανή προσομοιώνει εξ ολοκλήρου μία αρχιτεκτονική υλικού, πιθανώς διαφορετική από το πραγματικό υποκείμενο υλικό, επιτρέποντας έτσι να εκτελεστεί σε αυτήν ένα μη τροποποιημένο, φιλοξενούμενο λειτουργικό σύστημα σχεδιασμένο για τον εξομοιούμενο επεξεργαστή (π.χ. QEMU,

έκδοση για PowerPC του VirtualPC κλπ). Η εξομοίωση είναι διερμηνεία σε χρόνο εκτέλεσης του κώδικα του φιλοξενούμενου OS (Operating System), με έναν κύκλο ανάγνωσης-αποκωδικοποίησης-εκτέλεσης όπου κάθε εντολή που ανήκει στο σύνολο εντολών του επεξεργαστή-πηγής μεταφράζεται σε μία εντολή του συνόλου εντολών του επεξεργαστή-στόχου. Παράλληλα, η εικονική μηχανή παρέχει μία αφαίρεση της μνήμης, των συσκευών Εισόδου/Εξόδου κλπ., φροντίζοντας ώστε κάθε μεταφρασμένη εντολή που απευθύνεται σε αυτά τα υποσυστήματα να τροποποιεί μόνο τις αφαιρέσεις/λογικές αναπαραστάσεις τους, οι οποίες κατευθύνονται και υλοποιούνται από το λογισμικό ελέγχου, και όχι το πραγματικό υλικό.

Στην παρα-εικονικοποίηση η εικονική μηχανή δεν προσομοιώνει επακριβώς το υλικό, αλλά παρέχει στις εικονικές μηχανές ένα API, μία προγραμματιστική διασύνδεση, που επιτρέπει την εκτέλεση ενός τροποποιημένου, φιλοξενούμενου OS σχεδιασμένου για εκτέλεση από τον συγκεκριμένο επόπτη (π.χ. Denali, XEN). Το προαναφερθέν API (Application Programming Interface) ονομάζεται διασύνδεση υπερκλήσεων. Σύμφωνα με αυτό, ένα λειτουργικό σύστημα πρέπει να μεταφερθεί ρητά σε έκδοση κατάλληλη για εκτέλεση από ένα σύστημα παρα-εικονικοποίησης, ώστε ο φιλοξενούμενος πυρήνας αντί να προσπελαύνει το υλικό άμεσα, να εκτελεί υπερκλήσεις και να αναμένει απαντήσεις ή ασύγχρονες ειδοποιήσεις από τον επόπτη. Το όφελος από τη βελτίωση των επιδόσεων και την απλοποίηση της γραφής του επόπτη είναι μεγάλο [7].

Στην παρούσα διπλωματική εργασία χρησιμοποιείται η τεχνική της πλήρους εικονικοποίησης, καθώς προσφέρει περισσότερα πλεονεκτήματα σε σχέση με τις δύο άλλες μορφές εικονικοποίησης που αναφέρθηκαν προηγουμένως. Μερικά από τα πλεονεκτήματα που προσφέρει η πλήρης εικονικοποίηση και που αποτελούν το λόγο που πολλές εταιρείες και οργανισμοί την επιλέγουν είναι τα παρακάτω:

- **Πιο εύκολη διαχείριση**: Η πλήρης εικονικοποίηση μας δίνει τη δυανατότητα για διαχείρισης τόσο των φυσικών όσο και των εικονικών πόρων ενός συστήματος. Επίσης, μας επιτρέπει την κατάνομη των φυσικών ππόρων με τέτοιο ώστε να είναι ο βέλτιστος για τις ανάγκες της εφαρμογής μας.

- **Γρήγορος χρόνος αποκατάστασης**: Σε περίπτωση βλάβης συστήματος ή καταστροφής, η εικονικοποίηση επιτρέπει την ταχύτερη ανάκτηση πόρων πληροφορικής, η οποία παρέχει βελτιωμένη επιχειρηματική συνέχεια και έσοδα. Οι παλαιότερες υποδομές δεν είναι σε θέση να ανακάμψουν μέσα σε λίγες ώρες και στις περισσότερες περιπτώσεις οι εταιρείες αντιμετωπίζουν πολύ μεγαλύτερο χρόνο διακοπής, γεγονός που έχει ως αποτέλεσμα την απώλεια εσόδων.

- **Καλύτερη δυνατότητα κλιμάκωσης**: Τα εικονικά περιβάλλοντα έχουν σχεδιαστεί ώστε να είναι κλιμακωτά, γεγονός που επιτρέπει μεγαλύτερη ευελιξία όσον αφορά την ανάπτυξη της εταιρείας. Αντί να αγοράζουν πρόσθετα στοιχεία υποδομής, οι νέες εφαρμογές και οι αναβαθμίσεις μπορούν εύκολα να εφαρμοστούν με την εικονικοποίηση.

- **Εξοικονόμηση κόστους και χώρου**: Η εξοικονόμηση στο κόστος της υποδομής πληροφορικής είναι πραγματικότητα κατά τη μετάβαση σε εικονικοποίηση. Επιπλέον επεκτείνεται και

στη μείωση τόσο της κατανάλωσης ενέργειας, όσο και του προσωπικού του τομέα πληροφορικής, ελαχιστοποιώντας ταυτόχρονα το χώρο που απαιτείται για να φιλοξενηθεί ένα ολοκληρωμένο τμήμα πληροφορικής.

- **Καλύτερη απόδοση της επένδυσης**: Εκτός από τη μείωση του κόστους συντήρησης μιας παλαιότερης υποδομής, οι εταιρείες μπορούν να αυξήσουν την απόδοση της επένδυσής τους διασφαλίζοντας τη συνέχεια λειτουργίας της επιχείρησης μετά από μια καταστροφή και αποτρέποντας την απώλεια προσόδων [9].

2.1.2 Εισαγωγή στην Εικονικοποίηση δικτυακών λειτουργιών

Όπως αναφέραμε και προηγουμένως, οι φορείς εκμετάλλευσης δικτύων παραδοσιακά έχουν δημιουργήσει υπηρεσίες δικτύου που συνίστανται σε πολλαπλές λειτουργίες δικτύου, όπως δρομολογητές, φορτιστές και θυρίδες. Αυτές οι λειτουργίες φυσικού δικτύου (Physical Network Functions - PNF) παρέχονται συνήθως ως συσκευές που αποτελούνται από ιδιόκτητο λογισμικό και ειδικό υλικό. Ενώ η εσωτερική δομή αυτών των λειτουργιών είναι πιθανόν πιο περίπλοκη από μόνη της, από την άποψη του φορέα εκμετάλλευσης, ένα PNF είναι μία ενιαία οντότητα. Με την εμφάνιση της Εικονικοποίησης Δικτύων Λειτουργιών (NFV), ο διαχωρισμός του λογισμικού και του υλικού οδηγεί στην αποσύνθεση της υλοποίησης των λειτουργιών του δικτύου σε Εικονικοποιημένες Λειτουργίες Δικτύου (VNFs) που εκτελούνται πάνω από μια Υποδομή NFV (NFV Infrastructure) [4].

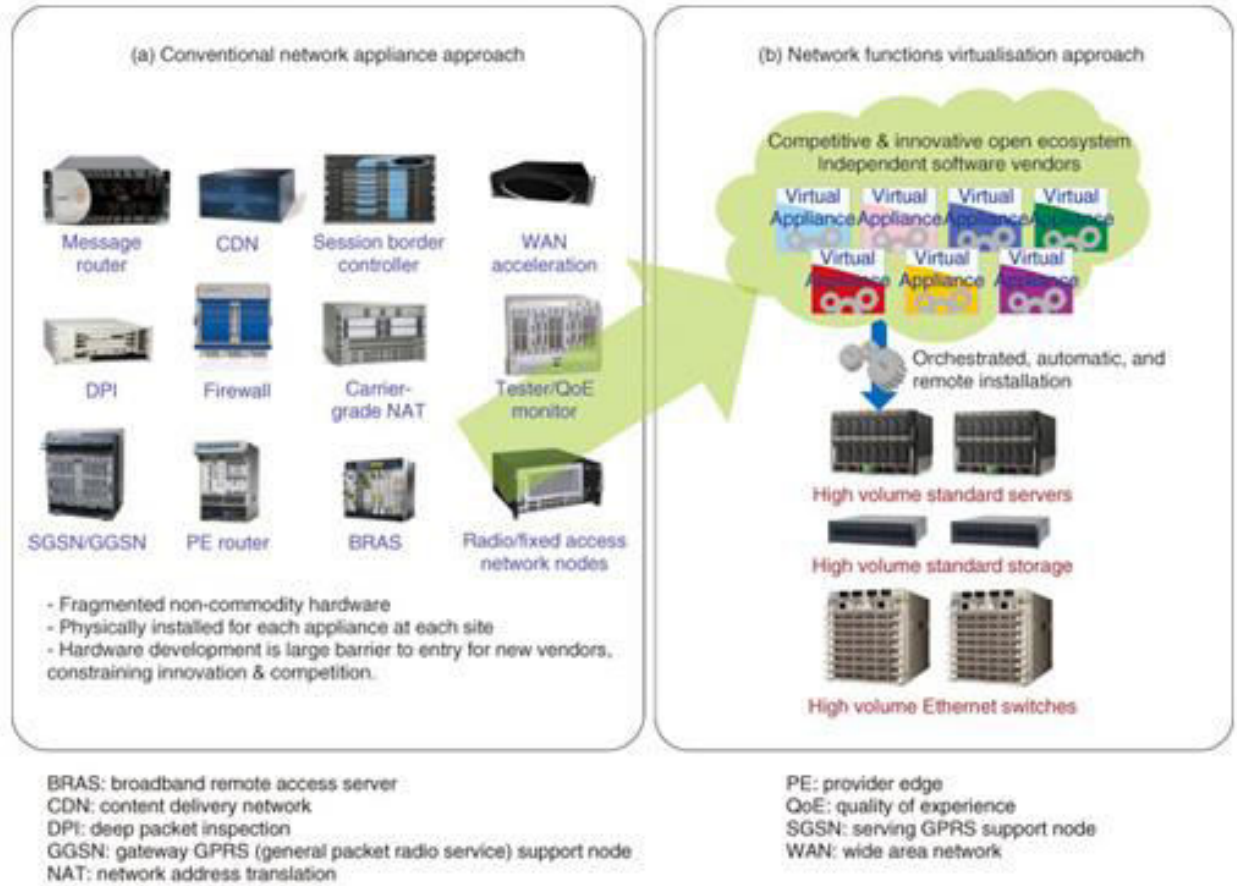
Ένα VNF είναι μια οντότητα λογισμικού που εκτελείται σε διαφορετικά περιβάλλοντα εκτέλεσης με μεταβλητές επιλογές και δυνατότητες διαμόρφωσης (π.χ. επιτάχυνση υλικού) που επηρεάζουν την απόδοση. Οι πολυπλεγμένες υποδομές NFV πόρων μπορούν να επιβάλουν σχέσεις δύσκολης πρόβλεψης μεταξύ μετρήσεων απόδοσης VNF (π.χ. καθυστέρηση, απώλεια πλαισίου), των υποκείμενων πόρων που έχουν κατανεμηθεί (π.χ. μονάδες vCPU) και του συνολικού φόρτου εργασίας του συστήματος.

Οι διεργασίες που προσανατολίζονται στο λογισμικό που εφαρμόζεται στο NFV απαιτούν αυτοματοποιημένες πρακτικές δοκιμών που καλύπτουν τη φορητότητα της πλατφόρμας, τη λειτουργική ορθότητα και τη συγκριτική αξιολόγηση επιδόσεων για κάθε υποψήφια έκδοση VNF προτού διατεθεί για ανάπτυξη. Μια ενιαία σειρά αλλαγής κώδικα που περνάει όλες τις λειτουργικές δοκιμές, θα μπορούσε επίσης να υπονομεύσει την απόδοση ενός VNF για συγκεκριμένους φόρτους εργασίας και πλατφόρμες. Αυτός ο κίνδυνος απαιτεί τυποποιημένες μεθόδους δοκιμών προς επαρκείς συγκριτικούς δείκτες VNF.

Η ετερογένεια των περιβαλλόντων υποδομής NFV (NFVI) περιλαμβάνει ποικίλες επιλογές εικονικοποίησης και δυνατότητες συστήματος (π.χ. εκφόρτωση υλικού [HW], παράκαμψη πυρήνα) για διαφορετικά φορτία εργασίας και διαφορετικές συνθήκες κοινής χρήσης πόρων.

Οι προτεινόμενες κατευθυντήριες αρχές για τον σχεδιασμό και τη δημιουργία ενός πλαισίου δοκιμών επιδόσεων μπορούν να συνδυαστούν με πολλούς πρακτικούς τρόπους για πολλαπλούς σκοπούς δοκιμών VNF:

- **Συγκρισιμότητα**: Η παραγωγή των δοκιμών πρέπει να είναι απλή στην κατανόηση και να επεξεργάζεται, σε μορφότυπο αναγνώσιμο από τον άνθρωπο, συνεκτικό και εύκολα επαναχρησιμοποιήσιμο (π.χ. εισροές για αναλυτικές εφαρμογές).
- **Επαναληψιμότητα**: Η ρύθμιση του test πρέπει να καθορίζεται ολοκληρωτικά μέσω ενός ευέλικτου σχεδίου, το οποίο μπορεί να ερμηνεύεται και να εκτελείται από την πλατφόρμα δοκιμών επανειλημμένα με την υποστήριξη της προσαρμογής.
- **Διαμορφωσιμότητα**: Θα πρέπει να διατίθενται ανοικτές διεπαφές και μοντέλα επεκτάσιμων μηνυμάτων μεταξύ των συνιστωσών για την ευέλικτη σύνθεση των περιγραφών δοκιμών και των διαμορφώσεων της πλατφόρμας.
- **Διαλειτουργικότητα**: Οι δοκιμές μεταφέρονται σε διαφορετικά περιβάλλοντα χρησιμοποιώντας ελαφριά εξαρτήματα [1].

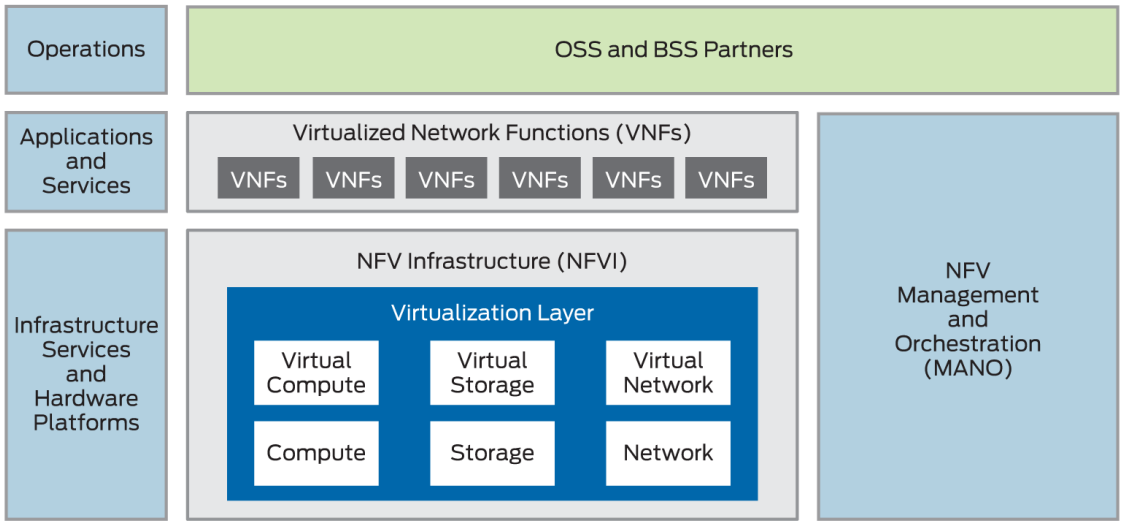


Εικόνα 2-2: Το όραμα της Εικονικοποίησης Δικτυακών Λειτουργιών

2.1.3 Το πλαίσιο αρχιτεκτονικής του NFV

Η Εικονικοποίηση Λειτουργιών Δικτύου προβλέπει την εφαρμογή των VNFs ως οντότητες μόνο λογισμικού που εκτελούνται μέσω του NFV Infrastructure (NFVI). Το πλαίσιο αρχιτεκτονικής της τεχνολογίας NFV, όπως έχει οριστεί από το Ευρωπαϊκό Ινστιτούτο Τηλεπικοινωνιακών Προτύπων (European Telecommunications Standards Institute – ETSI) τον Οκτώβριο του 2013, αποτελείται από τρεις κύριους τομείς:

- **Εικονικοποιημένες λειτουργίες δικτύου (VNFs)**, ως εφαρμογή του λογισμικού μιας λειτουργίας δικτύου που είναι ικανή να τρέχει μέσω του NFVI.
- **Υποδομή NFV (NFVI)**, συμπεριλαμβανομένης της πολυμορφίας των φυσικών πόρων και του τρόπου με τον οποίο μπορούν να εικονικοποιηθούν. Το NFVI υποστηρίζει την εκτέλεση των VNFs.
- **Διαχείριση και ενορχήστρωση NFV (Management ANd Orchestration)**, η οποία καλύπτει την ενορχήστρωση και τη διαχείριση του κύκλου ζωής των φυσικών πόρων ή/και των πόρων λογισμικού που υποστηρίζουν την εικονικοποίηση της υποδομής και τη διαχείριση του κύκλου ζωής των VNF. Η διαχείριση και ενορχήστρωση του NFV επικεντρώνεται σε όλα τα συγκεκριμένα καθήκοντα διαχείρισης που σχετίζονται με το εικονικό περιβάλλον και απαιτούνται στο πλαίσιο του NFV.



Εικόνα 2-3: Το πλαίσιο αρχιτεκτονικής ενός NFV στο υψηλό επίπεδο

Η προτεινόμενη τοποθέτηση βασίζεται στη χαρτογράφηση των οντοτήτων του πλαισίου NFV για να ταιριάζει καλύτερα στο εικονικό περιβάλλον. Ο Διαχειριστής Εικονικών Πόρων, ο Διαχειριστής VNF και ο Ενορχηστρωτής έχουν ομαδοποιηθεί σε επίπεδο προειδοποίησης. Δεδομένου ότι το εικονικό περιβάλλον δεν φιλοξενεί μόνο VNF, αλλά και άλλες εφαρμογές πληροφορικής, αυτή η ομαδοποίηση οδηγεί σε έναν κεντρικό ελεγκτή. Η υποδομή που παρέχει το NFVI ως υπηρεσία παρέχει σύγχρονες υπηρεσίες ταυτόχρονα στους ίδιους πόρους υλικού. Ουσιαστικά, ο hypervisor διαχειρίζεται και ενορχηστρώνει τους φυσικούς και λογικούς πόρους του εικονικού περιβάλλοντος. Γνωρίζει τις εικονικές μηχανές που χρησιμοποιούν το υποκείμενο υλικό και διαχειρίζεται τον προγραμματισμό πόρων και τις αποφάσεις για μετανάστευση, κλιμάκωση των πόρων και ανάκτηση

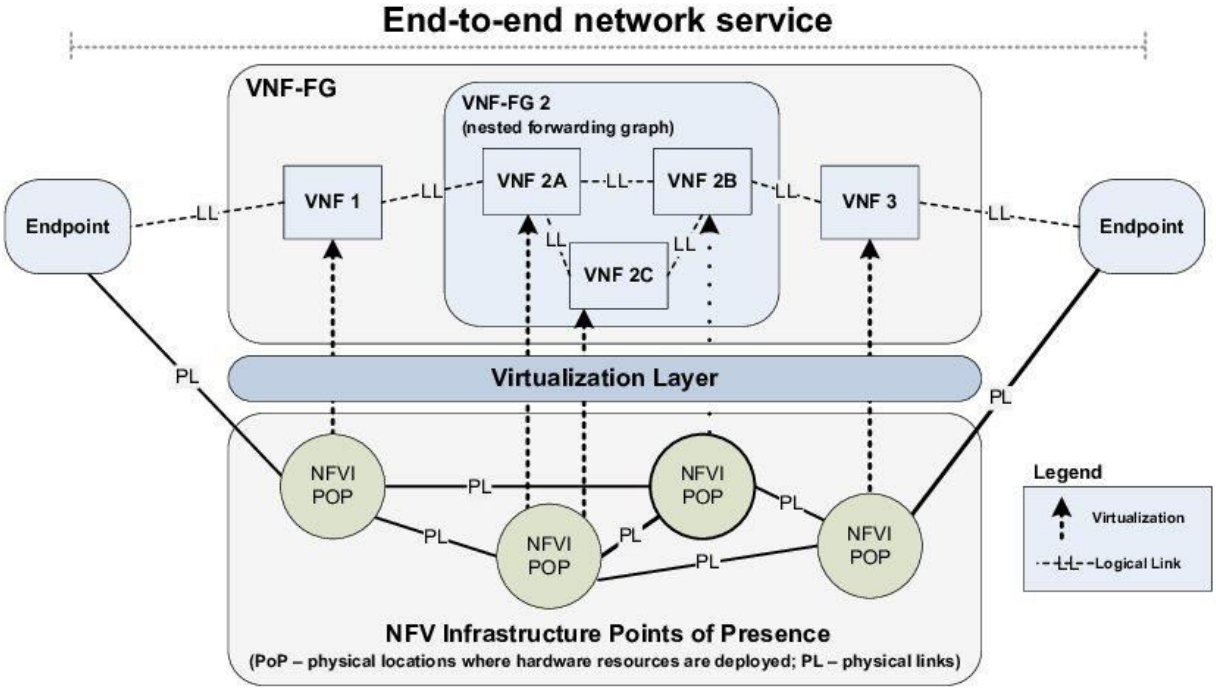
σφαλμάτων και αποτυχιών με σκοπό την αποτελεσματικότερη κάλυψη των προδιαγραφών ποιότητας VM (VNFs και APPs). Το στρώμα εικονικοποίησης αποτελείται από έναν εικονικό διαχειριστή πολλαπλών πλατφόρμων που τρέχει πάνω από τον hypervisor για να εξασφαλίσει τη φορητότητα και την ευελιξία του VNF ανεξάρτητα από τον hypervisor. Μερικά παραδείγματα πλατφορμών που φιλοξενούν εικονικές μηχανές είναι τα OpenStack, Eucalyptus, oVirt, OpenNebula και Nimbula. Η εικονική μηχανή φιλοξενεί το VNF και το σύστημα διαχείρισης στοιχείων (EMS). Κάθε παράμετρος VNF έχει το ιδιωτικό EMS της για να μειώσει την πολυπλοκότητα κατά τη μετεγκατάσταση ενός υφιστάμενου VNF ή την εκκίνηση ενός νέου [3].

Με τη χρήση της παραπάνω αρχιτεκτονικής, επιτυγχάνονται οι στόχοι που υπόσχεται το NFV, όπως η ευελιξία στην ανάθεση λειτουργιών εικονικού δικτύου (VNFs) στο υλικό, η καινοτομία ταχείας υπηρεσίας, η ενισχυμένη λειτουργική αποδοτικότητα, η μειωμένη χρήση ενέργειας και οι ανοιχτές τυποποιημένες διασυνδέσεις μεταξύ VNF. Κάθε VNF λειτουργεί σε ένα πλαίσιο που περιλαμβάνει δυναμική εκκίνηση και ενορχήστρωση των περιπτώσεων VNF από τη μονάδα Διαχείρισης και Ενορχήστρωσης (MANO). Επιπλέον, το περιβάλλον φιλοξενίας NFVI διαχειρίζεται τεχνολογίες εικονικοποίησης τεχνολογιών πληροφορικής για να καλύψει όλες τις απαιτήσεις VNF όσον αφορά τα δεδομένα, την κατανομή πόρων, τις εξαρτήσεις, τη διαθεσιμότητα και άλλα χαρακτηριστικά.

Μια υπηρεσία δικτύου μπορεί να προβληθεί αρχιτεκτονικά ως γράφημα προώθησης των λειτουργιών δικτύου (NF), διασυνδεόμενη από την υποστηριζόμενη υποδομή δικτύου. Αυτές οι λειτουργίες δικτύου μπορούν να υλοποιηθούν είτε σε ένα μόνο δίκτυο χειριστών είτε να συνεργάζονται μεταξύ διαφορετικών δικτύων χειριστών. Η υποκείμενη συμπεριφορά λειτουργίας δικτύου συμβάλλει στη συμπεριφορά της υπηρεσίας στο υψηλότερο επίπεδο. Ως εκ τούτου, η συμπεριφορά των υπηρεσιών δικτύου είναι ένας συνδυασμός της συμπεριφοράς των λειτουργικών τμημάτων της, τα οποία μπορούν να περιλαμβάνουν μεμονωμένα NFs, NF Sets, NF Forwarding Graphs ή/και το δίκτυο υποδομής.

Τα τελικά σημεία και οι λειτουργίες δικτύου της υπηρεσίας δικτύου αντιπροσωπεύονται ως κόμβοι και αντιστοιχούν σε συσκευές ή εφαρμογές. Ένα γράφημα προώθησης NF μπορεί να έχει συνδέσμους λειτουργίας δικτύου συνδεδεμένους με λογικούς συνδέσμους που μπορεί να είναι μονής κατεύθυνσης, αμφίδρομη, πολυεκπομπή ή/και μετάδοση. Ένα παράδειγμα μιας τέτοιας υπηρεσίας δικτύου από-άκρο-σε-άκρο μπορεί να περιλαμβάνει ένα smartphone, ένα ασύρματο δίκτυο, ένα τείχος προστασίας, ένα balancer φορτίου και ένα σύνολο από διακομιστές CDN (Content Distribution Network). Ο τομέας δραστηριότητας NFV βρίσκεται εντός των πόρων που ανήκουν στον φορέα εκμετάλλευσης. Επομένως, μια συσκευή που ανήκει στον πελάτη, π.χ. ένα κινητό τηλέφωνο βρίσκεται εκτός του πεδίου εφαρμογής, δεδομένου ότι ο φορέας εκμετάλλευσης δεν μπορεί να ασκήσει την εξουσία του σε αυτό. Η εικόνα 2-4 δείχνει ένα παράδειγμα μιας υπηρεσίας δικτύου από άκρο σε άκρο και τα διάφορα επίπεδα που εμπλέκονται στη διαδικασία εικονικοποίησης. Σε αυτό το παράδειγμα, μια υπηρεσία δικτύου από άκρο σε άκρο μπορεί να αποτελείται μόνο από VNF και δύο τελικά σημεία.

Η αποσύνδεση του υλικού και του λογισμικού στην εικονικοποίηση λειτουργιών δικτύου πραγματοποιείται από ένα στρώμα εικονικοποίησης. Αυτό το επίπεδο αφαιρεί τους πόρους υλικού της υποδομής NFV. Τα NFVI-PoPs περιλαμβάνουν πόρους για υπολογισμό, αποθήκευση και δικτύωση που αναπτύσσονται από έναν διαχειριστή δικτύου, όπως φαίνεται στην εικόνα 2-4. Οι εικονικοποιημένες λειτουργίες δικτύου εκτελούνται πάνω στο στρώμα εικονικοποίησης, το οποίο είναι μέρος του NFVI.



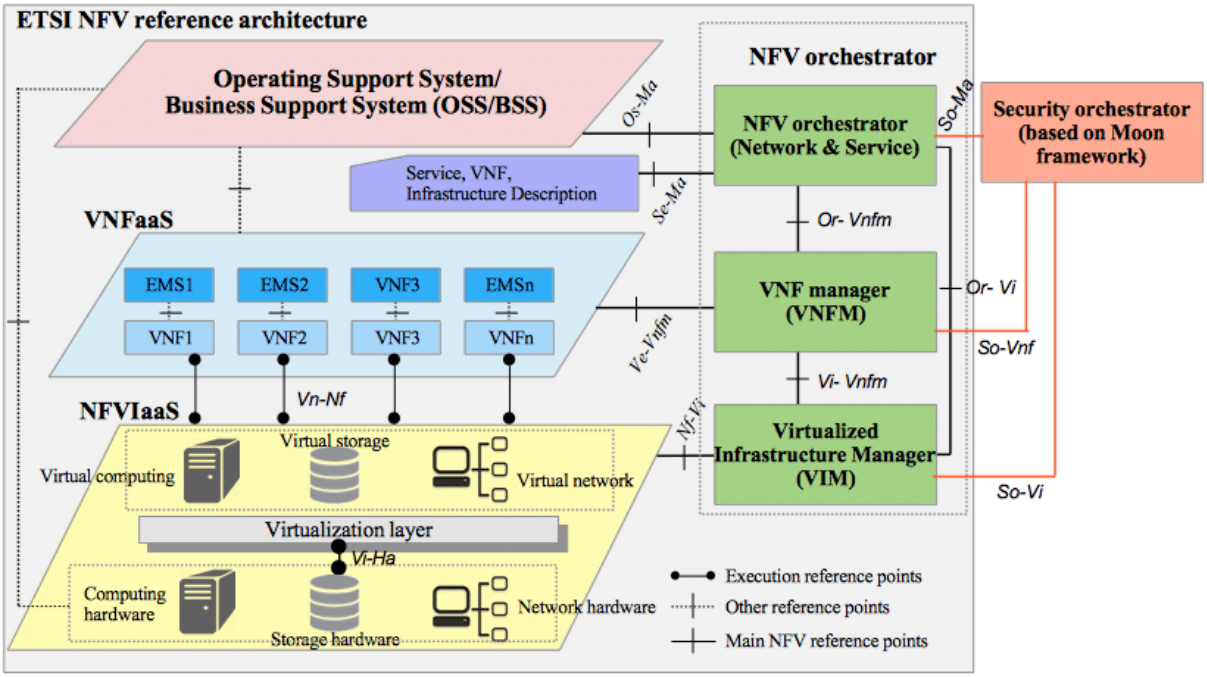
Εικόνα 2-4: Δείγμα NFV υπηρεσίας από-άκρη-σε-άκρη

Το αρχιτεκτονικό πλαίσιο NFV προσδιορίζει τα λειτουργικά τμήματα και τα κύρια σημεία αναφοράς μεταξύ αυτών των μπλοκ. Τα λειτουργικά τμήματα είναι:

- **Εικονικοποιημένη λειτουργία δικτύου (VNF)**.
- **Σύστημα Διαχείρισης Στοιχείων (EMS)**.
- **Υποδομή NFV**, συμπεριλαμβανομένων του υλικού, των εικονικών πόρων και του στρώματος εικονικοποίησης.
- **Εικονικοποιημένος διαχειριστής υποδομής**
- **Ενορχηστρωτής**
- **Διαχειριστές VNF**
- **Περιγραφή υπηρεσίας, VNF και υποδομής**.
- **Λειτουργίες και Συστήματα Υποστήριξης Επιχειρήσεων (OSS/BSS).**

Η εικόνα 2-5 δείχνει το αρχιτεκτονικό πλαίσιο NFV που απεικονίζει τα λειτουργικά τμήματα και τα σημεία αναφοράς στο πλαίσιο του NFV. Τα κύρια σημεία αναφοράς και εκτέλεσης εμφανίζονται με σταθερές γραμμές και εμπίπτουν στο πεδίο εφαρμογής του NFV. Αυτοί είναι πιθανοί στόχοι για την τυποποίηση. Το αρχιτεκτονικό πλαίσιο που παρουσιάζεται επικεντρώνεται στις

λειτουργίες που απαιτούνται για την εικονικοποίηση και την επακόλουθη λειτουργία του δικτύου του φορέα εκμετάλλευσης.



Εικόνα 2-5: Σημεία αναφοράς του πλαισίου αρχιτεκτονικής ενός NFV

Η υποδομή NFV είναι το σύνολο όλων των στοιχείων υλικού και λογισμικού που δημιουργούν το περιβάλλον στο οποίο αναπτύσσονται, εκτελούνται και γίνονται διαχειρίσιμα τα VNFs. Η υποδομή NFV μπορεί να εκτείνεται σε διάφορες τοποθεσίες, δηλαδή στις περιοχές όπου λειτουργούν τα NFVI-PoPs (Points-Of-Presence). Το δίκτυο που παρέχει συνδεσιμότητα μεταξύ αυτών των τοποθεσιών θεωρείται ότι αποτελεί μέρος της υποδομής NFV.

Στο NFV, οι φυσικοί πόροι υλικού περιλαμβάνουν υπολογιστές, αποθήκευση και δίκτυο που παρέχουν επεξεργασία, αποθήκευση και συνδεσιμότητα με VNF μέσω του στρώματος εικονικοποίησης (π.χ. Hypervisor). Το υλικό πληροφορικής θεωρείται ότι είναι COTS σε αντίθεση με το εργοστασιακό υλικό. Οι πόροι αποθήκευσης μπορούν να διαφοροποιηθούν μεταξύ του προσαρμοσμένου χώρου αποθήκευσης κοινόχρηστου δικτύου (NAS - Network-Attached Storage) και του χώρου αποθήκευσης που βρίσκεται στον ίδιο τον διακομιστή.

Το στρώμα εικονικοποίησης αφαιρεί τους πόρους υλικού και αποσυνδέει το λογισμικό VNF από το υποκείμενο υλικό, εξασφαλίζοντας έτσι έναν ανεξάρτητο από τον υλικό κύκλο ζωής για τα VNF. Με λίγα λόγια, το επίπεδο εικονικοποίησης είναι υπεύθυνο για την απελευθέρωση και λογική κατανομή φυσικών πόρων, συνήθως ως στρώμα αφαίρεσης υλικού, την ενεργοποίηση του λογισμικού που υλοποιεί το VNF για χρήση της υποκείμενης εικονικής υποδομής και την παροχή εικονικοποιημένων πόρων στο VNF, ώστε να μπορεί να εκτελεστεί το τελευταίο.

Από την άποψη του NFV, η εικονικοποιημένη διαχείριση υποδομής περιλαμβάνει τις λειτουργίες που χρησιμοποιούνται για τον έλεγχο και τη διαχείριση της αλληλεπίδρασης ενός VNF με

υπολογιστές, αποθηκευτικούς και δικτυακούς πόρους υπό την εξουσία του, καθώς και την εικονικοποίηση τους.

Σύμφωνα με τον κατάλογο των πόρων υλικού που προσδιορίζονται στην αρχιτεκτονική, ο εικονικοποιημένος διαχειριστής υποδομής εκτελεί κατ' αρχάς διαχείριση πόρων. Είναι υπεύθυνος για την απογραφή λογισμικού (π.χ. hypervisors), υπολογιστών, αποθήκευσης και πόρων δικτύου που προορίζονται για την υποδομή NFV, την κατανομή δυνατοτήτων εικονικοποίησης, π.χ. VMs σε hypervisors, υπολογιστικούς πόρους, αποθήκευση και σχετική συνδεσιμότητα δικτύου και διαχείριση πόρων και κατανομής υποδομών, π.χ. να αυξήσουν τους πόρους για τα VM, να βελτιώσουν την ενεργειακή απόδοση και την αποκατάσταση των πόρων. Επίσης, εκτελεί λειτουργίες, για ορατότητα και διαχείριση της υποδομής NFV, ανάλυση βασικών αιτιών των επιδόσεων από την άποψη της υποδομής NFV και συλλογή πληροφοριών σφαλμάτων υποδομής και πληροφοριών για σχεδιασμό χωρητικότητας, παρακολούθηση και βελτιστοποίηση.

Ο Ενορχηστρωτής είναι υπεύθυνος για την ενορχήστρωση και τη διαχείριση της υποδομής και των πόρων λογισμικού του NFV και για την υλοποίηση των υπηρεσιών δικτύου στο NFVI.

Ο διαχειριστής VNF είναι υπεύθυνος για τη διαχείριση του κύκλου ζωής του VNF (π.χ., δημιουργία στιγμιότυπων, ενημέρωση, ερώτημα, κλιμάκωση, τερματισμός). Μπορούν να αναπτυχθούν πολλοί διαχειριστές VNF. Ένας διαχειριστής VNF μπορεί να αναπτυχθεί για κάθε VNF ή ένας διαχειριστής VNF μπορεί να εξυπηρετήσει πολλαπλά VNF.

Επίσης, ορίζονται και μερικά σημεία αναφοράς, τα οποία είναι απαραίτητα για την επικοινωνία και την ανταλλαγή πληροφοριών μεταξύ των κύριων μονάδων. Μερικά από τα σημεία αναφορά ενός δικτύου NFV είναι τα εξής:

- VI-Ha: Αυτό το σημείο αναφοράς διασυνδέει το στρώμα εικονικοποίησης με τους πόρους υλικού για να δημιουργήσει ένα περιβάλλον εκτέλεσης για VNFs και συλλέγει σχετικές πληροφορίες κατάστασης πόρων υλικού για τη διαχείριση των VNF χωρίς να εξαρτάται από οποιαδήποτε πλατφόρμα υλικού.

- Vn-Nf: Αυτό το σημείο αναφοράς αντιπροσωπεύει το περιβάλλον εκτέλεσης που παρέχεται από το NFVI στο VNF. Δεν αναλαμβάνει κανένα συγκεκριμένο πρωτόκολλο ελέγχου. Είναι στο πεδίο εφαρμογής του NFV, προκειμένου να διασφαλιστεί η ανεξαρτησία του υλικού, οι απαιτήσεις απόδοσης και φορητότητας του VNF.

- Or-Vnfm: Αυτό το σημείο αναφοράς χρησιμοποιείται για αιτήματα σχετικά με πόρους, π.χ. την εξουσιοδότηση, την επικύρωση, την κράτηση και την κατανομή από τον διαχειριστή των VNF, καθώς και την αποστολή πληροφοριών διαμόρφωσης στον διαχειριστή VNF, έτσι ώστε το VNF να μπορεί να διαμορφωθεί κατάλληλα για να λειτουργεί εντός του γραφήματος προώθησης VNF στην υπηρεσία δικτύου, συλλογή πληροφοριών κατάστασης του VNF που απαιτούνται για τη διαχείριση του κύκλου ζωής των υπηρεσιών δικτύου.

- Vi-Vnfm: Αυτό το σημείο αναφοράς χρησιμοποιείται για αιτήματα κατανομής πόρων από τον διαχειριστή VNF, εικονική ρύθμιση παραμέτρων πόρου υλικού και ανταλλαγή πληροφοριών κατάστασης (π.χ. συμβάντα).

- Or-Vi: Αυτό το σημείο αναφοράς χρησιμοποιείται για παραγγελίες πόρων ή/και αιτήσεις κατανομής από τον Ενορχηστρωτή, εικονική ρύθμιση παραμέτρων πόρου υλικού και ανταλλαγή πληροφοριών κατάστασης (π.χ. συμβάντα).

- Nf-Vi: Αυτό το σημείο αναφοράς χρησιμοποιείται για ειδική ανάθεση εικονικοποιημένων πόρων ως απάντηση σε αιτήματα κατανομής πόρων, προώθηση πληροφοριών εικονικοποιημένων πόρων και ανταλλαγή πόρου υλικού και πληροφορίες κατάστασης (π.χ. συμβάντα).

- Os-Ma: Αυτό το σημείο αναφοράς χρησιμοποιείται για αιτήματα που αφορούν τη διαχείριση του κύκλου ζωής των υπηρεσιών δικτύου, αιτήματα για τη διαχείριση του κύκλου ζωής VNF, προώθηση πληροφοριών σχετικά με την κατάσταση του NFV, ανταλλαγές διαχείρισης πολιτικής, ανταλλαγές δεδομένων, προώθηση αρχείων λογιστικής και χρήσης που σχετίζονται με το NFV και ικανότητα NFVI και ανταλλαγές πληροφοριών απογραφής

- Ve-Vnfm: Αυτό το σημείο αναφοράς χρησιμοποιείται για αιτήματα σχετικά με τη διαχείριση του κύκλου ζωής VNF, ανταλλαγή πληροφοριών διαμόρφωσης και ανταλλαγή πληροφοριών κατάστασης που είναι απαραίτητες για τη διαχείριση των υπηρεσιών δικτύου.

- Se-Ma: Αυτό το σημείο αναφοράς χρησιμοποιείται για την ανάκτηση πληροφοριών σχετικά με το πρότυπο ανάπτυξης VNF, το VNF Forwarding Graph, τις πληροφορίες σχετικά με την υπηρεσία και τα μοντέλα πληροφοριών υποδομής NFV. Οι παρεχόμενες πληροφορίες χρησιμοποιούνται από τη διαχείριση και ενορχήστρωση του NFV.

2.1.4 Πλεονεκτήματα NFV

Τα πλεονεκτήματα που προσφέρει η χρήση της τεχνολογίας NFV σχετίζονται άμεσα με τους περιορισμούς που αναφέραμε στο πρώτο κεφάλαιο, αφού έρχεται για να τους αναιρέσει. Μερικά από τα πλεονεκτήματα αυτά είναι:

- **Ευελιξία υλικού**: οι φορείς εκμετάλλευσης δικτύων μπορούν να επιλέγουν και να κατασκευάζουν το υλικό με τον πιο αποτελεσματικό τρόπο ώστε να ταιριάζει στις ανάγκες και τις απαιτήσεις τους, καθώς το NFV χρησιμοποιεί κανονικό υλικό COTS. Με το NFV, οι πάροχοι μπορούν πλέον να επιλέξουν μεταξύ πολλών διαφορετικών προμηθευτών και να έχουν την ευελιξία να επιλέξουν τις δυνατότητες υλικού που είναι βέλτιστες για την αρχιτεκτονική και τον προγραμματισμό τους.

- **Ταχύτερος κύκλος ζωής υπηρεσίας**: Οι νέες υπηρεσίες ή λειτουργίες δικτύου μπορούν πλέον να αναπτυχθούν ταχύτερα, με βάση τη ζήτηση και την ανάγκη, παρέχοντας οφέλη τόσο στους τελικούς χρήστες όσο και στους παρόχους δικτύου. Σε αντίθεση με το φυσικό υλικό, τα VNF μπορούν να δημιουργηθούν και να αφαιρεθούν πολύ γρήγορα. Η δυνατότητα γρήγορης προσθήκης νέων

λειτουργιών δικτύου (ευκινησία ανάπτυξης) είναι ένα από τα μεγαλύτερα πλεονεκτήματα του NFV. Οι υπηρεσίες τώρα μπορούν επίσης να τεθούν σε λειτουργία ή να παροπλιστούν με το πάτημα ενός κουμπιού χωρίς την ανάγκη αλλαγής στο υλικό που είναι μια χρονοβόρα διαδικασία, μειώνοντας δραστικά τους χρόνους ανάπτυξης από εβδομάδες σε λεπτά.

- **Κλιμακωσιμότητα και ελαστικότητα**: Το NFV επιτρέπει αλλαγές στις δυνατότητες των νέων υπηρεσιών και εφαρμογών, προσφέροντας ένα μέσο επέκτασης και συρρίκνωσης των πόρων που χρησιμοποιούνται από τα VNF. Σε μια παραδοσιακή συσκευή δικτύου, θα ήταν απαραίτητη είτε η πλήρης αντικατάστασης συσκευής είτε η αναβάθμιση υλικού για να αλλάξει οποιαδήποτε από αυτές τις παραμέτρους. Αλλά επειδή τα VNFs δεν περιορίζονται από τους περιορισμούς του προσαρμοσμένου φυσικού υλικού, μπορούν να προσφέρουν αυτήν την ελαστικότητα. Επομένως, τα δίκτυα δεν χρειάζονται ουσιαστικά υπερβολικές προμήθειες για την προσαρμογή των απαιτήσεων μεταφορικής ικανότητας.

- **Μείωση των CAPEX και OPEX**: Η μείωση τόσο των επενδυτικών όσο και των λειτουργικών δαπανών για τους τηλεπικοινωνιακούς παρόχους λόγω της μειωμένης χρήσης εξειδικευμένου υλικού για τη λειτουργία των VNFs αλλά και λόγω της μείωσης της ενεργειακής κατανάλωσης εξαιτίας της μετάβασης από υποδομές υλικού σε εικονικές υποδομές.

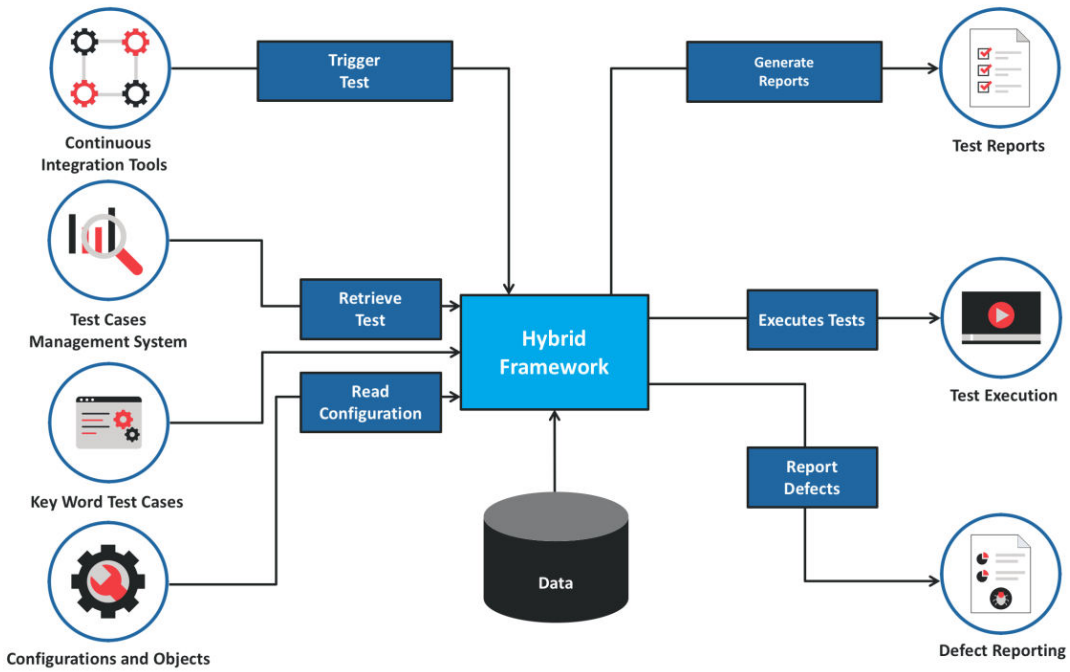
- **Γρήγορη αποκατάσταση**: Το NFV προσφέρει τη δυνατότητα γρήγορης αποκατάστασης αλλά και την επαναχρησιμοποίηση των εικονικών υποδομών σε περιπτώσεις βλάβης τους. Επίσης, μειώνονται οι απώλειες δεδομένων σε μια τέτοια περίπτωση λόγω της ύπαρξης αντιγράφων ασφαλείας για κάθε μονάδα της υποδομής του δικτύου [8].

2.2 Αυτοματοποίηση Δοκιμών (Test Automation)

Πολλές, ίσως περισσότερες, εφαρμογές λογισμικού σήμερα γράφονται ως εφαρμογές που βασίζονται στον ιστό, οι οποίες εκτελούνται σε ένα πρόγραμμα περιήγησης στο Internet. Η αποτελεσματικότητα της δοκιμής αυτών των εφαρμογών ποικίλλει ευρέως μεταξύ εταιρειών και οργανισμών. Σε μια εποχή εξαιρετικά διαδραστικών και ευαίσθητων διαδικασιών λογισμικού όπου πολλοί οργανισμοί χρησιμοποιούν κάποια μορφή μεθόδου Agile, η αυτοματοποίηση των δοκιμών γίνεται συχνά μια απαίτηση για προγράμματα λογισμικού. Η αυτοματοποίηση των δοκιμών είναι συχνά η απάντηση. Ο έλεγχος αυτοματισμού σημαίνει ότι χρησιμοποιείται ένα εργαλείο λογισμικού για την εκτέλεση επαναληπτικών δοκιμών σε σχέση με την υπό δοκιμή εφαρμογή. Για δοκιμές παλινδρόμησης παρέχει αξιόπιστη ανταπόκριση.

Η αυτοματοποίηση των δοκιμών (test automation) είναι η χρήση ειδικού λογισμικού (ξεχωριστά από το λογισμικό που δοκιμάζεται) για τον έλεγχο της εκτέλεσης των δοκιμών και της σύγκρισης των πραγματικών αποτελεσμάτων με τα προβλεπόμενα αποτελέσματα. Η αυτοματοποίηση των δοκιμών μπορεί να αυτοματοποιήσει ορισμένες επαναλαμβανόμενες αλλά απαραίτητες εργασίες σε μια επίσημη διαδικασία δοκιμών που ήδη υπάρχει ή να εκτελέσει πρόσθετες δοκιμές που θα ήταν

δύσκολο να γίνει χειροκίνητα. Η αυτοματοποίηση των δοκιμών είναι ζωτικής σημασίας για τη συνεχή δοκιμή και τη συνεχή παράδοση (CI/CD – Continuous Integration/Continuous Delivery).



Εικόνα 2-6: Δείγμα υλοποίησης αυτοματοποίησης δοκιμών

Υπάρχουν πολλές προσεγγίσεις για τον έλεγχο της αυτοματοποίησης, ωστόσο οι παρακάτω γενικές προσεγγίσεις χρησιμοποιούνται ευρέως:

Γραφική δοκιμή διεπαφής χρήστη: Ένα πλαίσιο δοκιμών που δημιουργεί συμβάντα διεπαφής χρήστη, τις πληκτρολογήσεις και τα clicks του ποντικιού, και παρατηρεί της αλλαγές που έχουν ως αποτέλεσμα τη διεπαφή χρήστη, για να επιβεβαιώσει ότι η παρατηρήσιμη συμπεριφορά του προγράμματος είναι σωστή.

Διεξαγωγή δοκιμών με API: Ένα πλαίσιο δοκιμών που χρησιμοποιεί μια διεπαφή προγραμματισμού για την εφαρμογή για την επικύρωση της υπό δοκιμή συμπεριφοράς. Τυπικά, η δοκιμή που βασίζεται σε API παρακάμπτει εντελώς τη διεπαφή χρήστη της εφαρμογής. Μπορεί της να ελέγχει δημόσιες (συνήθως) διασυνδέσεις με τάξεις, οι ενότητες ή οι βιβλιοθήκες δοκιμάζονται με μια ποικιλία από επιχειρήματα εισόδου για να επιβεβαιώσουν ότι τα αποτελέσματα που επιστρέφονται είναι σωστά.

Συνεχείς δοκιμές: Είναι η διαδικασία εκτέλεσης αυτοματοποιημένων δοκιμών ως μέρος του αγωγού παράδοσης λογισμικού για την άμεση ανατροφοδότηση των επιχειρηματικών κινδύνων που συνδέονται με έναν υποψήφιο για την απελευθέρωση λογισμικού. Για συνεχή έλεγχο, το πεδίο των δοκιμών επεκτείνεται από την επικύρωση των απαιτήσεων από τη βάση προς την κορυφή ή τις ιστορίες των χρηστών στην αξιολόγηση των απαιτήσεων του συστήματος που σχετίζονται με γενικούς επιχειρησιακούς στόχους.

Σε αυτοματοποιημένες δοκιμές, ο μηχανικός δοκιμής ή ο υπεύθυνος διασφάλισης της ποιότητας του λογισμικού πρέπει να έχουν δυνατότητα κωδικοποίησης λογισμικού, καθώς οι περιπτώσεις δοκιμής γράφονται με τη μορφή πηγαίου κώδικα ο οποίος, όταν εκτελείται, παράγει αποτελέσματα σύμφωνα με της ισχυρισμούς που αποτελούν μέρος του. Ορισμένα εργαλεία αυτοματοποίησης δοκιμής επιτρέπουν τη διεξαγωγή δοκιμαστικής σύνταξης με λέξεις-κλειδιά αντί για κωδικοποίηση, τα οποία δεν απαιτούν προγραμματισμό.

Ένας τρόπος για την αυτόματη δημιουργία δοκιμαστικών περιπτώσεων είναι η δοκιμή βάσει μοντέλου, μέσω της χρήσης του μοντέλου του συστήματος για την παραγωγή δοκιμαστικών περιπτώσεων. Σε ορισμένες περιπτώσεις, επιτρέπει σε χρήστες χωρίς τη σχετική εμπειρία να δημιουργήσουν αυτοματοποιημένες περιπτώσεις δοκιμαστικών επιχειρήσεων σε γλώσσα που είναι κατανοητή από τον άνθρωπο, ώστε να μην απαιτείται προγραμματισμός οποιουδήποτε είδους για τη διαμόρφωση πολλαπλών λειτουργικών συστημάτων, προγραμμάτων περιήγησης και έξυπνων συσκευών.

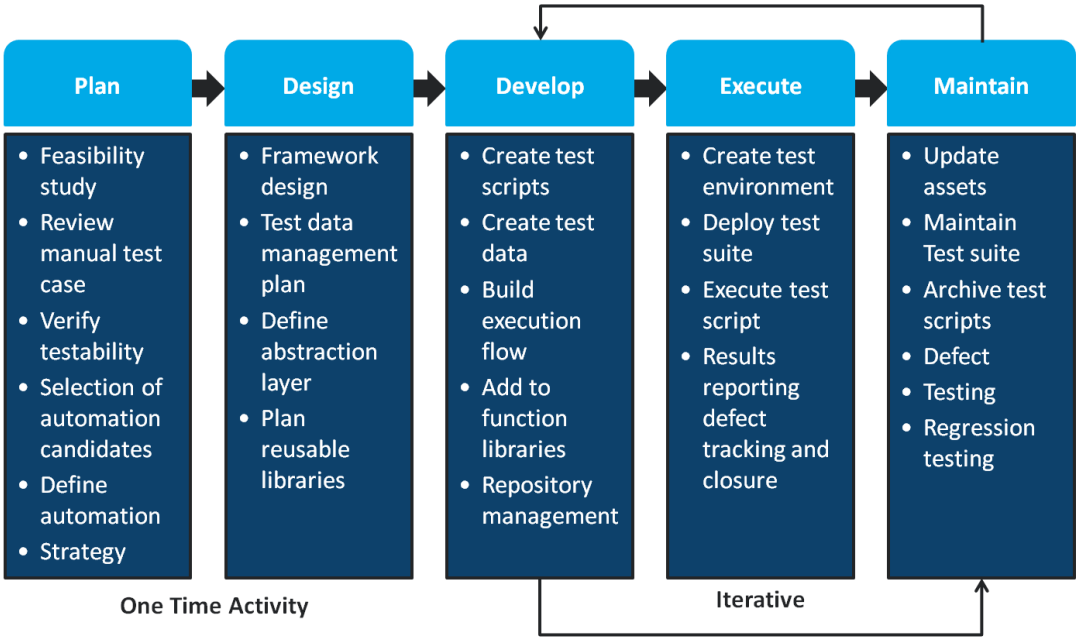
Ένα πλαίσιο αυτοματοποίησης δοκιμών είναι ένα ολοκληρωμένο σύστημα που καθορίζει τους κανόνες αυτοματοποίησης του συγκεκριμένου προϊόντος. Αυτό το σύστημα ενσωματώνει τις βιβλιοθήκες λειτουργιών, τις πηγές δεδομένων δοκιμής, τις λεπτομέρειες αντικειμένων και διάφορες επαναχρησιμοποιούμενες μονάδες. Αυτά τα στοιχεία λειτουργούν ως μικρά δομικά στοιχεία τα οποία πρέπει να συγκεντρωθούν για να αντιπροσωπεύσουν μια επιχειρηματική διαδικασία. Το πλαίσιο παρέχει τη βάση της αυτοματοποίησης των δοκιμών και απλοποιεί την προσπάθεια αυτοματισμού.

Το κύριο πλεονέκτημα του πλαισίου υποθέσεων, εννοιών και εργαλείων που παρέχουν υποστήριξη για αυτοματοποιημένο έλεγχο λογισμικού είναι το χαμηλό κόστος συντήρησης. Εάν υπάρχει αλλαγή σε οποιαδήποτε δοκιμαστική περίπτωση, τότε χρειάζεται μόνο να ενημερωθεί το αρχείο δοκιμαστικής περίπτωσης, και το σενάριο οδηγού script και εκκίνησης θα παραμείνει το ίδιο. Στην ιδανική περίπτωση, δε χρειάζεται να ενημερώνονται τα σενάρια όταν γίνουν αλλαγές στην εφαρμογή [10].

Υπάρχουν πολλά πλεονεκτήματα για τον έλεγχο της αυτοματοποίησης. Τα περισσότερα σχετίζονται με την επαναληψιμότητα των δοκιμών και την ταχύτητα με την οποία μπορούν να εκτελεστούν οι δοκιμές. Υπάρχουν διάφορα εμπορικά εργαλεία και εργαλεία ανοιχτού κώδικα που διατίθενται για να βοηθήσουν στην ανάπτυξη αυτοματοποίησης δοκιμών. Η αυτοματοποίηση των δοκιμών έχει συγκεκριμένα πλεονεκτήματα για τη βελτίωση της μακροπρόθεσμης αποδοτικότητας των διαδικασιών δοκιμών μιας ομάδας λογισμικού. Η αυτοματοποίηση των δοκιμών υποστηρίζει:

- Συχνές δοκιμές παλινδρόμησης
- Γρήγορη ανατροφοδότηση στους προγραμματιστές
- Σχεδόν απεριόριστες επαναλήψεις εκτέλεσης δοκιμαστικών περιπτώσεων
- Υποστήριξη των μεθοδολογιών Agile και της ακραίας ανάπτυξης
- Πειθαρχική τεκμηρίωση των περιπτώσεων δοκιμής
- Προσαρμοσμένες αναφορές σφαλμάτων

- Εντοπισμός σφαλμάτων που διέφυγαν με τον χειροκίνητο έλεγχο [17].



Εικόνα 2-7: Πλεονεκτήματα χρήσης

Η επιλογή της σωστού πλαισίου και των σωστών τεχνικών scripting βοηθά στη διατήρηση του χαμηλότερου κόστους. Το κόστος που σχετίζεται με τη δοκιμή scripting οφείλεται σε προσπάθειες ανάπτυξης και συντήρησης. Η προσέγγιση της δέσμης ενεργειών που χρησιμοποιείται κατά τη διάρκεια της αυτοματοποίησης δοκιμών επηρεάζει το κόστος.

Διάφορα πλαίσια και τεχνικές scripting από αυτά που χρησιμοποιούνται ευρέως είναι τα:

- **Γραμμική** (κώδικας διαδικασιών, πιθανώς παράγεται από εργαλεία όπως αυτά που χρησιμοποιούν εγγραφή και αναπαραγωγή)
- **Δομημένη** (χρησιμοποιεί δομές ελέγχου)
- **Οδηγούμενη από δεδομένα** (τα δεδομένα διατηρούνται εκτός των δοκιμών σε μια βάση δεδομένων, υπολογιστικό φύλλο ή άλλο μηχανισμό)
- **Λέξη-κλειδί**
- **Υβριδικά** (χρησιμοποιούνται δύο ή περισσότερα από τα παραπάνω πρότυπα)
- **Ευέλικτο πλαίσιο αυτοματισμού**

Ένα πλαίσιο δοκιμών είναι υπεύθυνο για:

- τον καθορισμό της μορφής με την οποία εκφράζονται οι προσδοκίες
- τη δημιουργία ενός μηχανισμού για να ενσωματωθεί ή να οδηγήσει την εφαρμογή στη διαδικασία της δοκιμής
- την εκτέλεση των δοκιμών
- την αναφορά αποτελεσμάτων.

Οι διεπαφές αυτοματισμού δοκιμών είναι πλατφόρμες που παρέχουν ένα ενιαίο χώρο εργασίας για την ενσωμάτωση πολλαπλών εργαλείων και πλαισίων δοκιμών για Συστηματικό/Ενσωματωμένο έλεγχο της υπό δοκιμή εφαρμογής. Ο στόχος της διεπαφής αυτοματοποιημένων δοκιμών είναι να απλοποιήσει τη διαδικασία της χαρτογράφησης των δοκιμών σε επιχειρηματικά κριτήρια χωρίς τη χρήση κώδικα κατά τη διάρκεια της διαδικασίας. Η διεπαφή αυτοματισμού δοκιμών αναμένεται να βελτιώσει την αποτελεσματικότητα και την ευελιξία στη διατήρηση των σεναρίων δοκιμών. Αποτελείται από τις ακόλουθες βασικές ενότητες:

- Μηχανή διεπαφής
- Περιβάλλον περιβάλλοντος
- Αποθήκη αντικειμένων [10].

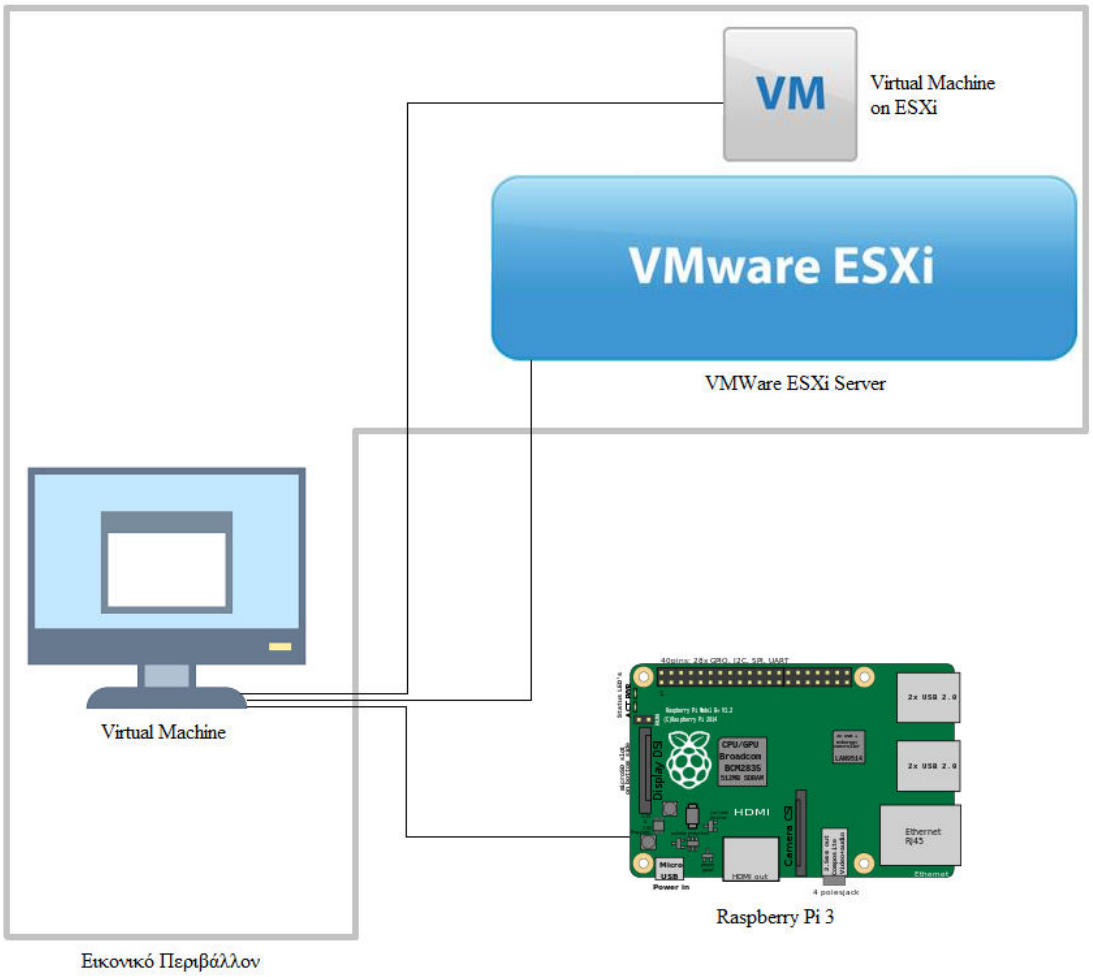
3

Αρχιτεκτονική Δικτύου

Στο κεφάλαιο αυτό θα γίνει μια πρώτη, θεωρητική παρουσίαση της αρχιτεκτονικής του δικτύου που σχεδιάστηκε για τις ανάγκες του θέματος της παρούσας διπλωματικής εργασίας. Πιο συγκεκριμένα, θα γίνει ανάλυση της κάθε μονάδας του δικτύου ξεχωριστά, με επεξήγηση των ρόλων και της λειτουργίας που επιτελεί στο δίκτυο. Στη συνέχεια, θα γίνει μια παρουσίαση των εργαλείων που επιλέχθηκαν και χρησιμοποιήθηκαν για την υλοποίηση των μονάδων του δικτύου. Μερικά από αυτά τα εργαλεία είναι το Elasticsearch, το Kibana, το Django, το Selenium και το Ansible. Θα γίνει επεξήγηση των λόγων που επιλέχθηκαν τα συγκεκριμένα εργαλεία έναντι των άλλων διαθέσιμων παραθέτοντας μερικά θετικά στοιχεία των επιλεγμένων.

3.1 Θεωρητική Περιγραφή

Ο κύριος στόχος της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μιας web-εφαρμογής για την αυτοματοποιημένη δοκιμή της τεχνολογίας εικονικοποίησης δικτυακών λειτουργιών (NFV) που αναλύθηκε στο προηγούμενο κεφάλαιο. Στην εφαρμογή αυτή βασικό ρόλο παίζουν τρείς μονάδες: ο Ενορχηστρωτής, το Raspberry Pi και ο ESXi Server. Το περιβάλλον στο οποίο έχει γίνει η υποδομή της εικονικοποίησης (NFVI) είναι το περιβάλλον του VMWare. Εκεί βρίσκονται ο Ενορχηστρωτής και ο ESXi Server, ενώ το Raspberry Pi υπάρχει ως φυσική οντότητα στις εγκαταστάσεις του Εργαστηρίου Δικτύων Υπολογιστών. Όλες οι μονάδες όμως ανήκουν εικονικά στο ίδιο LAN, αυτό του εργαστηρίου.



Εικόνα 3-1: Αρχιτεκτονική δικτύου

3.2 Ενορχηστρωτής

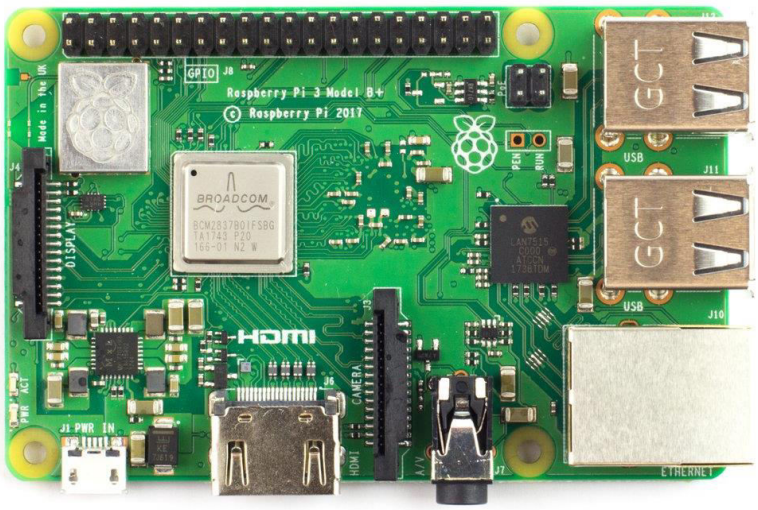
Ο ενορχηστρωτής είναι μια εικονική μηχανή που βρίσκεται στο εικονικό περιβάλλον του VMWare. Σε αυτόν βρίσκονται όλα τα απαραίτητα στοιχεία που χρειάζονται για να τεθεί η εφαρμογή σε λειτουργία. Πρόκειται για ένα VNF εφόσον βρίσκεται στο εικονικό περιβάλλον που έχουμε δημιουργήσει και είναι ικανό να εκτελέσει όλες τις δικτυακές λειτουργίες εικονικά. Επίσης, είναι η μονάδα MANO του δικτύου καθώς από αυτήν ξεκινούν όλες οι λειτουργίες και ενέργειες προς τα υπόλοιπα μέρη του δικτύου. Επίσης είναι και η μονάδα που αναλαμβάνει την αυτοματοποίηση και παραμετροποίηση των διαδικασιών για τις επιμέρους μονάδες.

3.3 ESXi Server

Ο ESXi Server είναι επίσης μια εικονική μηχανή που βρίσκεται στο εικονικό περιβάλλον του VMWare. Ο ρόλος του στη διάταξη του δικτύου μας είναι η δημιουργία εικονικών μηχανών (VM's) σε κάθε αίτηση του χρήστη προς αυτόν με σκοπό την παρατήρηση της εικονικής κίνησης δεδομένων

προς αυτές. Όντας στην υποδομή της εικονικοποίησης λειτουργιών που έχουμε θέσει, αποτελεί και αυτός ένα VNF και είναι ικανός να εκτελέσει όλες τις σχετικές λειτουργίες.

3.4 Raspberry Pi 3



Εικόνα 3-2: Raspberry Pi 3

Το Raspberry Pi είναι μια συσκευή η οποία επιλέχθηκε για την αναπαράσταση μιας φυσικής συσκευής για τη δημιουργία κίνησης δεδομένων μέσω της περιήγησης στο διαδίκτυο. Πρόκειται για έναν μικρό υπολογιστή πάνω σε μια πλακέτα που αναπτύχθηκε στο Ηνωμένο Βασίλειο από το Raspberry Pi Foundation για την προώθηση της διδασκαλίας της επιστήμης της πληροφορικής στα σχολεία και στις αναπτυσσόμενες χώρες. Το αρχικό μοντέλο έγινε πολύ πιο δημοφιλές από ό,τι αναμενόταν, με πωλήσεις ακόμα και εκτός της στοχευόμενης αγοράς, για χρήσεις όπως η ρομποτική και τα νευρωνικά δίκτυα. Δεν περιλαμβάνει περιφερειακά όπως πληκτρολόγια και ποντίκια.

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 100 Base Ethernet
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

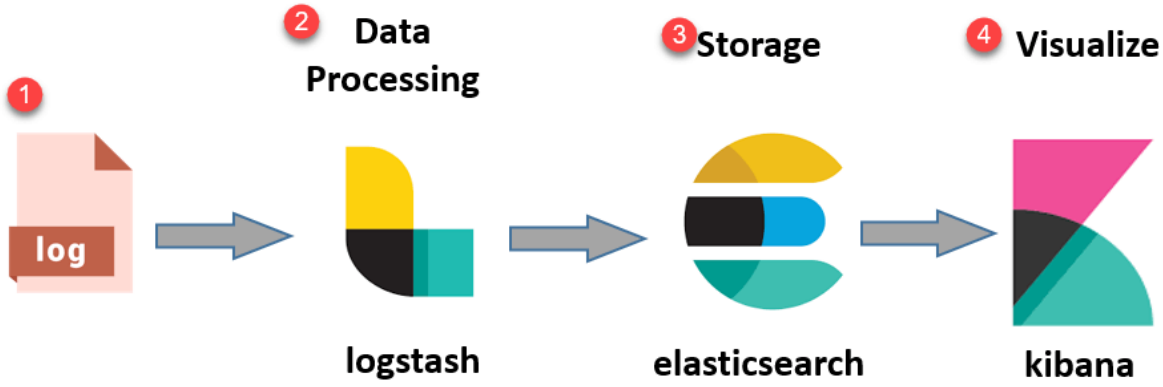
Εικόνα 3-3: Τεχνικά χαρακτηριστικά Raspberry Pi 3

Τα βασικά δομικά στοιχεία του Raspberry Pi είναι ένας επεξεργαστής τεχνολογίας ARM, μια μνήμη RAM, μια θύρα HDMI καθώς και ένας αριθμός θυρών USB για τη σύνδεση των περιφερειακών συσκευών που δεν περιλαμβάνονται στην πλακέτα. Η ταχύτητα του επεξεργαστή κυμαίνεται από 700 MHz έως 1,4 GHz, η εσωτερική μνήμη κυμαίνεται από 256 MB έως 1 GB RAM και οι θύρες USB από μία έως τέσσερις, ανάλογα με την έκδοση και το μοντέλο του Raspberry Pi. Επίσης, μπορούν να χρησιμοποιηθούν κάρτες SD για την αποθήκευση του λειτουργικού συστήματος και της μνήμης προγραμμάτων σε μεγέθη SDHC ή MicroSDHC. Τα μοντέλα B διαθέτουν θύρα Ethernet 8P8C και τα Pi 3 και Pi Zero W διαθέτουν ενσωματωμένο Wi-Fi 802.11n και Bluetooth.

Το λειτουργικό σύστημα με το οποίο λειτουργεί το Raspberry Pi είναι το Raspbian, το επίσημο λειτουργικό σύστημα που έχει αναπτύξει η Raspberry Pi Foundation και αποτελεί μια ειδική διανομή του διάσημου λειτουργικού συστήματος Linux, βασισμένο στο Debian. Παρέχει επίσης τη δυνατότητα για λειτουργικά συστήματα τρίτων κατασκευαστών όπως Ubuntu, Windows 10, IoT Core, RISC OS και εξειδικευμένες διανομές κέντρων πολυμέσων. Οι κύριες γλώσσες προγραμματισμού που υποστηρίζει είναι η Python και η Scratch.

Η επιλογή του Raspberry Pi έγινε γιατί αποτελεί έναν ισχυρό και ταυτόχρονα φθηνό υπολογιστή σε μέγεθος τσέπης, ο οποίος μπορεί να παραμετροποιηθεί και να εκτελέσει πλήρως τις λειτουργίες τις οποίες απαιτούμε. Επίσης, το Raspberry Pi μας δίνει τη δυνατότητα για γρήγορη και εύκολη σύνδεση στο Internet μέσω ενός USB dongle. Μέσω αυτού γίνεται δυνατή η προσομοίωση όλων των δυνατών ασύρματων συνδέσεων, όπως είναι το υποδίκτυο του Εργαστηρίου Δικτύων Υπολογιστών στο Ε.Μ.Π., στο οποίο πραγματοποιήθηκαν όλες οι δοκιμές τα εφαρμογής μας.

3.5 ELK Stack



Εικόνα 3-4: Αρχιτεκτονική ELK Stack

Το "ELK" είναι το ακρωνύμιο για τρία projects ανοιχτού κώδικα, το Elasticsearch, το Logstash και το Kibana. Το Elasticsearch είναι μια μηχανή αναζήτησης και ανάλυσης. Το Logstash είναι ένας αγωγός επεξεργασίας δεδομένων από την πλευρά του διακομιστή, ο οποίος καταλαμβάνει ταυτόχρονα δεδομένα από πολλαπλές πηγές, τα μετασχηματίζει και στη συνέχεια το στέλνει σε ένα "stash" όπως

το Elasticsearch. Το Kibana επιτρέπει στους χρήστες να οπτικοποιήσουν δεδομένα με διαγράμματα και γραφήματα στην Elasticsearch.

Παρακάτω παρουσιάζουμε αναλυτικά κάθε μονάδα της ELK Stack

3.5.1 Elasticsearch

Το Elasticsearch είναι ένας εξαιρετικά επεκτάσιμος μηχανισμός αναζήτησης και ανάλυσης πλήρους κειμένου ανοιχτού κώδικα. Μας επιτρέπει να αποθηκεύουμε, να αναζητούμε και να αναλύουμε μεγάλους όγκους δεδομένων γρήγορα και σε σχεδόν πραγματικό χρόνο. Χρησιμοποιείται γενικά ως υποκείμενος κινητήρας/τεχνολογία που εξουσιοδοτεί εφαρμογές που έχουν σύνθετα χαρακτηριστικά και απαιτήσεις αναζήτησης.

Υπάρχουν μερικές έννοιες που αποτελούν τον πυρήνα της Elasticsearch.

- Κοντά σε πραγματικό χρόνο (NRT): Το Elasticsearch είναι μια πλατφόρμα αναζήτησης σχεδόν πραγματικού χρόνου. Αυτό σημαίνει ότι υπάρχει μια μικρή καθυστέρηση (κανονικά ένα δευτερόλεπτο) από τη στιγμή που καταχωρείται ένα έγγραφο μέχρι να γίνει δυνατή η αναζήτηση.

- Σύμπλεγμα (Cluster): Ένα σύμπλεγμα είναι μια συλλογή από έναν ή περισσότερους κόμβους (εξυπηρετητές) που συγκρατούν μαζί ολόκληρα τα δεδομένα σας και παρέχει ομοσπονδιακές δυνατότητες ευρετηρίασης και αναζήτησης σε όλους τους κόμβους. Αναγνωρίζεται από ένα μοναδικό όνομα, το οποίο από προεπιλογή είναι "elasticsearch".

- Κόμβος (Node): Ένας κόμβος είναι ένας μόνο διακομιστής που είναι μέρος του συμπλέγματος σας, αποθηκεύει τα δεδομένα σας και συμμετέχει στις δυνατότητες ευρετηρίασης και αναζήτησης του συμπλέγματος.

- Αναγνωριστικό (Index): Ένα αναγνωριστικό είναι μια συλλογή εγγράφων που έχουν κάπως παρόμοια χαρακτηριστικά. Αναγνωρίζεται από ένα όνομα (το οποίο πρέπει να είναι με πεζά γράμματα) και αυτό το όνομα χρησιμοποιείται για την αναφορά στο ευρετήριο κατά την εκτέλεση εργασιών ευρετηρίασης, αναζήτησης, ενημέρωσης και διαγραφής έναντι των εγγράφων που περιέχονται σε αυτόν.

- Έγγραφο (Document): Ένα έγγραφο είναι μια βασική μονάδα πληροφοριών που μπορεί να ευρετηριαστεί. Εκφράζεται σε JSON (JavaScript Object Notation) η οποία είναι μια πανταχού παρούσα μορφή ανταλλαγής δεδομένων στο Διαδίκτυο.

- Θραύσματα και αντίγραφα (Shards & Replicas): Ένα αναγνωριστικό μπορεί να αποθηκεύσει ενδεχομένως ένα μεγάλο όγκο δεδομένων που μπορεί να υπερβεί τα όρια υλικού ενός μόνο κόμβου. Για την επίλυση αυτού του προβλήματος, το Elasticsearch παρέχει τη δυνατότητα υποδιαίρεσης του αναγνωριστικού σας σε πολλαπλά κομμάτια που ονομάζονται θραύσματα.

Σε ένα περιβάλλον δικτύου/cloud όπου αποτυχίες μπορεί να αναμένονται οποιαδήποτε στιγμή, είναι πολύ χρήσιμο και συνιστάται ιδιαίτερα να έχουμε έναν μηχανισμό ανακατεύθυνσης σε περίπτωση που ένα θραύσμα/κόμβος βρεθεί με κάποιο τρόπο εκτός σύνδεσης ή εξαφανιστεί για οποιοδήποτε λόγο. Για το σκοπό αυτό, το Elasticsearch μας επιτρέπει να δημιουργήσουμε ένα ή

περισσότερα αντίγραφα των τεμαχίων του αναγνωριστικού μας σε αυτά που ονομάζονται αντίγραφα θραύσματα ή απλά αντίγραφα για συντομία [14].

3.5.2 Logstash

Το Logstash είναι μια μηχανή συλλογής δεδομένων ανοιχτού κώδικα με δυνατότητες pipeline σε πραγματικό χρόνο. Το Logstash μπορεί να ενοποιήσει δυναμικά δεδομένα από διαφορετικές πηγές και να ομαλοποιήσει τα δεδομένα σε προορισμούς της επιλογής σας.

Ενώ το Logstash οδήγησε αρχικά την καινοτομία στη συλλογή αρχείων καταγραφής, οι δυνατότητές του επεκτείνονται πολύ πέρα από αυτή τη χρήση. Οποιοσδήποτε τύπος συμβάντος μπορεί να εμπλουτιστεί και να μετασχηματιστεί με ένα ευρύ φάσμα εισροών, φίλτρων και plugins εκροές, με πολλούς εγγενείς κωδικοποιητές που απλοποιούν περαιτέρω τη διαδικασία κατάποσης.

Το λογισμικό Logstash έχει τρία στάδια: εισροές → φίλτρα → εκροές. Οι εισροές παράγουν συμβάντα, τα φίλτρα τα τροποποιούν και οι εκροές τα μεταφέρουν αλλού. Οι είσοδοι και οι εξόδους υποστηρίζουν κωδικοποιητές που σας δίνουν τη δυνατότητα να κωδικοποιείτε ή να αποκωδικοποιείτε τα δεδομένα καθώς εισέρχονται ή εξέρχονται από τον αγωγό χωρίς να χρειάζεται να χρησιμοποιηθεί ξεχωριστό φίλτρο.

Οι εισροές χρησιμοποιούνται για τη λήψη δεδομένων στο Logstash. Ορισμένες από τις συχνότερα χρησιμοποιούμενες εισροές είναι:

- file: διαβάζει από ένα αρχείο στο σύστημα αρχείων, όπως και η ουρά -0F της εντολής UNIX.

- syslog: ακούει στη γνωστή θύρα 514 για syslog μηνύματα και αναλύσεις σύμφωνα με τη μορφή RFC3164.

- redis: διαβάζει από έναν redis server, χρησιμοποιώντας και τα δύο redis κανάλια και τις redis λίστες. Το Redis χρησιμοποιείται συχνά ως "μεσίτης" σε μια κεντρική εγκατάσταση Logstash, η οποία αναβάλλει τα γεγονότα Logstash από απομακρυσμένους φορτωτές Logstash.

- beats: διεκπεραιώνει συμβάντα που αποστέλλονται από τους Beats.

Τα φίλτρα το δεύτερο δομικό στοιχείο του Logstash. Πρόκειται για στάδια ενδιάμεσης επεξεργασίας στο pipeline του Logstash. Μπορούμε να συνδυάσουμε τα φίλτρα με όρους για να εκτελέσουμε μια ενέργεια σε ένα συμβάν εάν πληροί ορισμένα κριτήρια. Μερικά χρήσιμα φίλτρα περιλαμβάνουν:

- grok: ανάλυση και δομή αυθαίρετου κειμένου. Ο Grok είναι ο καλύτερος τρόπος στο Logstash να αναλύει τα μη δομημένα δεδομένα καταγραφής σε κάτι δομημένο και ερωτηματολόγιο.

- mutate: πραγματοποιεί γενικούς μετασχηματισμούς στα πεδία συμβάντων.

- drop: αποθέτει πλήρως ένα συμβάν, για παράδειγμα, εντοπίζει σφάλματα.

- clone: κάνει ένα αντίγραφο ενός συμβάντος, ενδεχομένως προσθέτοντας ή αφαιρώντας πεδία.

- geoip: προσθέτει πληροφορίες σχετικά με τη γεωγραφική θέση των διευθύνσεων IP.

Οι εκροές είναι η τελική φάση του pipeline του Logstash. Ένα συμβάν μπορεί να περάσει από πολλαπλές εξόδους, αλλά μόλις ολοκληρωθεί η επεξεργασία της κάθε εξόδου, το συμβάν έχει τελειώσει την εκτέλεση του. Ορισμένες κοινώς χρησιμοποιούμενες έξοδοι περιλαμβάνουν:

- elasticsearch: στέλνει δεδομένα συμβάντων στο Elasticsearch.

- file: γράφει δεδομένα συμβάντων σε ένα αρχείο στο δίσκο.

- graphite: στέλνει δεδομένα συμβάντων στο graphite, ένα δημοφιλές εργαλείο ανοιχτού κώδικα για την αποθήκευση και τη γραφική παράσταση μετρήσεων.

- statsd: αποστολή δεδομένων συμβάντων σε statsd, υπηρεσία που "ακούει στατιστικά, όπως μετρητές και χρονομετρητές, αποστέλλεται μέσω UDP και τα στέλνει ομαδοποιημένα σε μία ή περισσότερες plugbable back-end υπηρεσίες".

Τέλος, οι κωδικοποιητές είναι ουσιαστικά φίλτρα ρεύματος που μπορούν να λειτουργήσουν ως μέρος μιας εισροής ή εκροής. Οι κωδικοποιητές μας επιτρέπουν να διαχωρίζουμε εύκολα τη μεταφορά των μηνυμάτων μας από τη διαδικασία serialization. Οι δημοφιλείς κωδικοποιητές περιλαμβάνουν json, msgpack και plain (κείμενο).

- json: κωδικοποίηση ή αποκωδικοποίηση δεδομένων στη μορφή JSON.

- multiline: συγχώνευση συμβάντων κειμένου πολλαπλών γραμμών όπως εξαίρεση java και μηνύματα stacktrace σε ένα μόνο συμβάν.

Ο αγωγός επεξεργασίας συμβάντων Logstash συντονίζει την εκτέλεση εισροών, φίλτρων και εκροών.

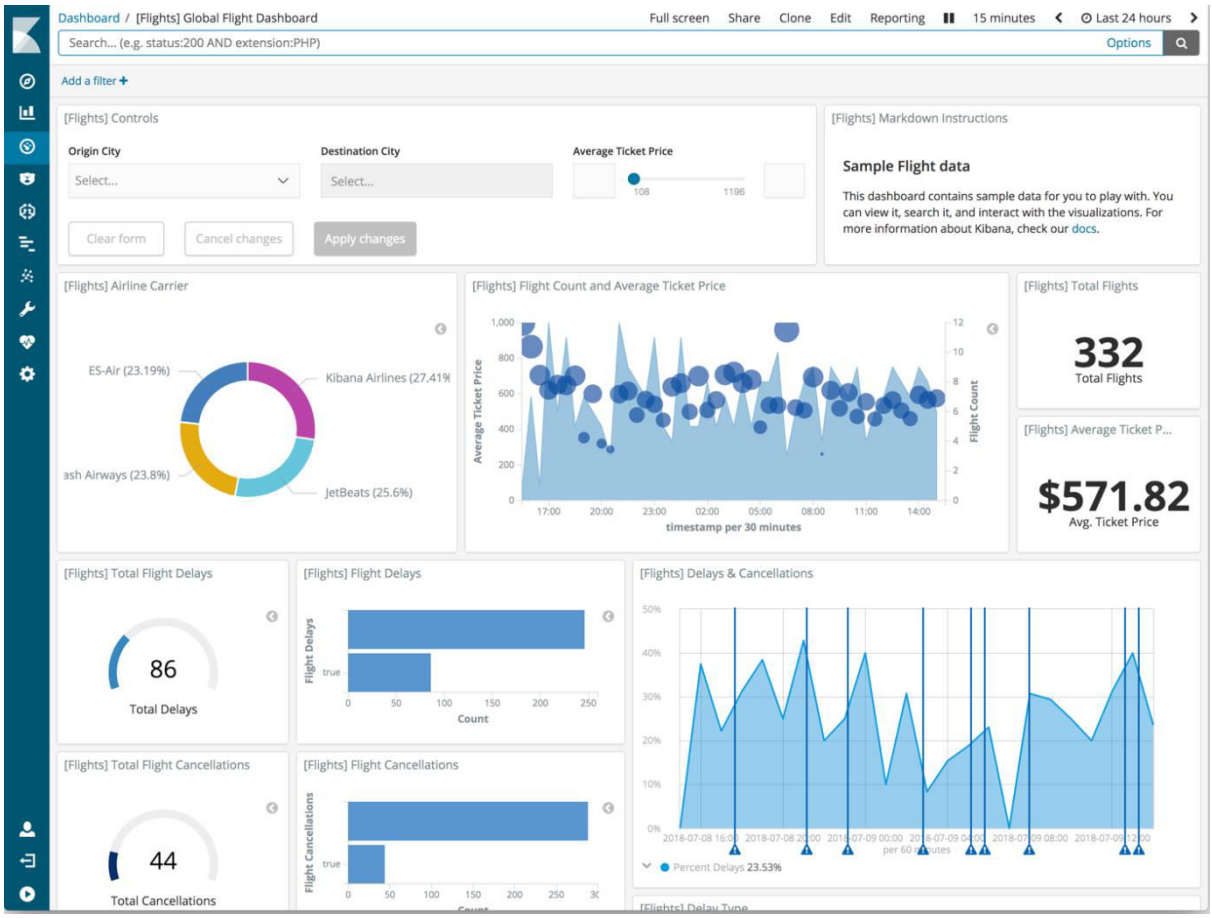
Κάθε στάδιο εισαγωγής στο pipeline του Logstash εκτελείται στο δικό του νήμα. Οι είσοδοι γράφουν συμβάντα σε μια κεντρική ουρά που είναι είτε στη μνήμη (προεπιλογή) είτε στο δίσκο. Κάθε νήμα εργαζόμενου pipeline παίρνει μια παρτίδα συμβάντων από αυτή την ουρά, εκτελεί τη δέσμη συμβάντων μέσω των διαμορφωμένων φίλτρων και στη συνέχεια εκτελεί τα φιλτραρισμένα συμβάντα μέσω οποιωνδήποτε εξόδων. Το μέγεθος της παρτίδας και ο αριθμός των θέσεων εργασίας των σωληνώσεων είναι διαμορφώσιμοι.

Από προεπιλογή, το Logstash χρησιμοποιεί ουρές που οριοθετούνται εντός μνήμης μεταξύ των σταδίων του pipeline (εισροής → φίλτρου και φίλτρου → εκροής) σε συμβάντα buffer. Αν το Logstash τερματιστεί χωρίς ασφάλεια, τα συμβάντα που είναι αποθηκευμένα στη μνήμη θα χαθούν. Για την αποφυγή της απώλειας δεδομένων, πρέπει να γίνει ενεργοποίηση στο Logstash ώστε να διατηρεί τα γεγονότα κατευθείαν στο δίσκο [15].

3.5.3 Kibana

Το Kibana είναι μια πλατφόρμα ανάλυσης και απεικόνισης ανοιχτού κώδικα σχεδιασμένη για να συνεργάζεται με το Elasticsearch και γενικά για διάφορες βάσεις δεδομένων τύπου time-series. Το Kibana χρησιμοποιείται στην αναζήτηση, προβολή και αλληλεπίδραση με τα δεδομένα που είναι αποθηκευμένα στους δείκτες Elasticsearch. Μπορεί εύκολα να εκτελέσει μια προηγμένη ανάλυση

δεδομένων και μια απεικόνιση των δεδομένων που επιθυμούμε σε μια ποικιλία διαγραμμάτων, πινάκων και χαρτών [16].



Εικόνα 3-5: Παράδειγμα Kibana dashboard

Στην παρούσα διπλωματική εργασία χρησιμοποιούμε την ELK Stack για την οπτικοποίηση στατιστικών από τις καταγραφές των πακέτων κατά τη διάρκεια περιήγησης. Συγκεκριμένα, ο φάκελος στον οποίο αποθηκεύονται οι καταγραφές από κάθε δοκιμή, παρακολουθείται από το Logstash και με τη βοήθεια του Elasticsearch, που κάνει την ευρετηρίαση των πακέτων, είναι δυνατή η οπτικοποίησή τους στο Kibana.

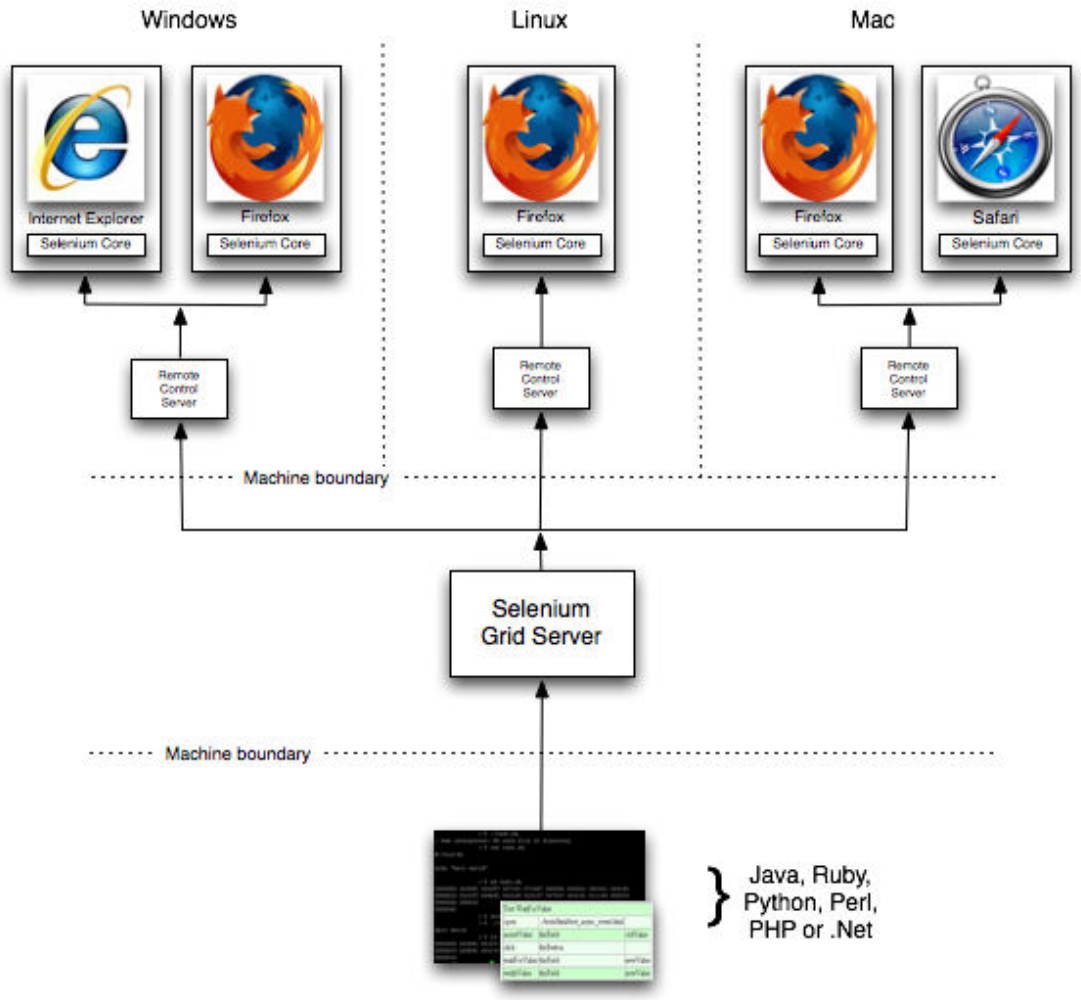
3.6 Selenium

Το Selenium είναι ένα σύνολο διαφορετικών εργαλείων λογισμικού, το καθένα με διαφορετική προσέγγιση για την υποστήριξη προς αυτοματοποίησης των δοκιμών. Ολόκληρη η σειρά εργαλείων έχει ως αποτέλεσμα ένα πλούσιο σύνολο λειτουργιών δοκιμών ειδικά προσαρμοσμένων για τις ανάγκες δοκιμών εφαρμογών ιστού όλων των τύπων. Αυτές οι λειτουργίες είναι ιδιαίτερα ευέλικτες, επιτρέποντας τις επιλογές για τον εντοπισμό στοιχείων UI και τη σύγκριση των αναμενόμενων αποτελεσμάτων των δοκιμών με την πραγματική συμπεριφορά εφαρμογής. Ένα από τα βασικά χαρακτηριστικά του

Selenium είναι η υποστήριξη της εκτέλεσης των δοκιμών σε πολλαπλές πλατφόρμες προγράμματος περιήγησης.

Το Selenium 2.0 (ή Selenium WebDriver) είναι η τελευταία κατεύθυνση του έργου και η νεότερη προσθήκη στο συνολικό εργαλείο. Αυτό το ολοκαίνουργιο εργαλείο αυτοματισμού παρέχει όλα τα είδη των εκπληκτικών δυνατοτήτων, συμπεριλαμβανομένου ενός πιο συνεκτικού και αντικειμενοστραφούς API, καθώς και μια λύση στους περιορισμούς της παλιάς εφαρμογής.

Τόσο οι κατασκευαστές του Selenium όσο και του WebDriver συμφώνησαν ότι και τα δύο εργαλεία έχουν πλεονεκτήματα και ότι η συγχώνευση των δύο έργων θα έκανε ένα πολύ πιο ισχυρό εργαλείο αυτοματοποίησης. Υποστηρίζει το API WebDriver και την υποκείμενη τεχνολογία, μαζί με την τεχνολογία Selenium 1.0 (ή Selenium RemoteControl – RC) κάτω από το API WebDriver για μέγιστη ευελιξία στη μεταφορά των δοκιμών προς. Επιπλέον, το Selenium 2.0 εξακολουθεί να χρησιμοποιεί τη διεπαφή Selenium 1.0 προς Selenium RC για συμβατότητα παλαιότερων εκδόσεων.



Εικόνα 3-5: Παράδειγμα υλοποίησης δοκιμών με χρήση του Selenium Grid

Το Selenium Grid επιτρέπει στο εργαλείο του Selenium RC να προσαρμόζεται για μεγάλες σειρές δοκιμών και για δοκιμαστικές σουίτες που πρέπει να εκτελούνται σε πολλαπλά περιβάλλοντα. Το Selenium Grid μας επιτρέπει να εκτελούμε τις δοκιμές μας παράλληλα, δηλαδή, μπορούν να

εκτελούνται ταυτόχρονα διαφορετικές δοκιμές σε διαφορετικές απομακρυσμένες μηχανές. Αυτό έχει δύο πλεονεκτήματα. Αρχικά, εάν έχουμε μια μεγάλη δοκιμαστική σουίτα ή μια δοκιμαστική σουίτα με αργούς ρυθμούς, μπορούμε να αυξήσουμε σημαντικά την απόδοσή της χρησιμοποιώντας το Selenium Grid για να διαιρέσουμε τη δοκιμαστική σουίτα, ώστε να εκτελούμε διαφορετικές δοκιμές ταυτόχρονα χρησιμοποιώντας αυτά τα διαφορετικά μηχανήματα. Επίσης, εάν πρέπει να εκτελέσουμε τη δοκιμαστική σουίτα σε πολλαπλά περιβάλλοντα, μπορούμε να έχουμε διαφορετικά απομακρυσμένα μηχανήματα που υποστηρίζουν και εκτελούν τις δοκιμές μας ταυτόχρονα. Σε κάθε περίπτωση το Selenium Grid βελτιώνει σημαντικά το χρόνο που χρειάζεται για να εκτελέσουμε τη σουίτα δοκιμών μας χρησιμοποιώντας παράλληλη επεξεργασία.

Το Selenium είναι ιδιαίτερα ευέλικτο. Υπάρχουν πολλοί τρόποι με τους οποίους μπορούμε να προσθέσουμε λειτουργικότητα σε δέσμες ενεργειών τόσο για το Selenium όσο και για το πλαίσιο του Selenium για να προσαρμόσουμε την αυτοματοποιημένη δοκιμή μας. Αυτή είναι ίσως η μεγαλύτερη δύναμη του Selenium σε σύγκριση με άλλα εργαλεία αυτοματισμού. Αυτές οι προσαρμογές περιγράφονται σε διάφορα σημεία σε όλο αυτό το έγγραφο. Επιπλέον, δεδομένου ότι το Selenium είναι λογισμικό ανοιχτού κώδικα, ο πηγαίος κώδικας μπορεί πάντα να μεταφορτωθεί και να τροποποιηθεί.

Για την παρούσα εργασία χρησιμοποιούμε τη βασική δομή του Selenium Grid με τη χρήση του Hub και των κόμβων. Ουσιαστικά, η εικονική μηχανή στην οποία βρίσκεται η εφαρμογή μας θα λειτουργεί ως Hub στο οποίο θα συνδέεται ως κόμβος το Raspberry Pi για να την εκτέλεση των δοκιμών περιήγησης. Για το Raspberry Pi επίσης, που είναι εκτελεστής των δοκιμών περιήγησης, χρησιμοποιούμε και τον WebDriver για τον περιηγητή Mozilla Firefox.

3.7 Ansible

Το Ansible είναι ένα εργαλείο αυτοματοποίησης πληροφορικής. Μπορεί να ρυθμίσει τα συστήματα, να αναπτύξει λογισμικό και να ενορχηστρώσει πιο προηγμένες λειτουργίες πληροφορικής, όπως συνεχείς αναπτύξεις ή μηδενικές αναμονής όσον αφορά το χρόνο ενημερώσεις. Οι κύριοι στόχοι του Ansible είναι η απλότητα και η ευκολία χρήσης. Δίνει επίσης μεγάλη έμφαση στην ασφάλεια και την αξιοπιστία, με ελάχιστα κινούμενα μέρη, χρήση του OpenSSH για μεταφορές (με άλλες τεχνικές και μεθόδους μεταφοράς για εναλλακτικές) και μια γλώσσα που σχεδιάζεται γύρω από τον έλεγχο από τον άνθρωπο, ακόμα κι αυτούς που δεν είναι συνηθισμένοι με το πρόγραμμα. Η πλατφόρμα είναι γραμμένη σε Python και επιτρέπει στους χρήστες να γράψουν εντολές στο YAML ως επιτακτικό προγραμματισμό.

Το Ansible διαχειρίζεται τα μηχανήματα με έναν τρόπο χωρίς την παρουσία manager. Δεν υπάρχει ποτέ θέμα αναβάθμισης απομακρυσμένων μηχανημάτων ή το πρόβλημα της μη ικανότητας

διαχείρισης των συστημάτων, επειδή τα προγράμματα έχουν απεγκατασταθεί. Επειδή το OpenSSH είναι ένα από τα πιο εξελιγμένα λογισμικά ανοιχτής πηγής, η έκθεση σε πιθανά προβλήματα ασφαλείας μειώνεται σημαντικά. Το Ansible είναι αποκεντρωμένο καθώς εξαρτάται από τα υπάρχοντα διαπιστευτήρια του λειτουργικού μας συστήματος για τον έλεγχο της πρόσβασης σε απομακρυσμένα μηχανήματα. Αν χρειαστεί, το Ansible μπορεί εύκολα να συνδεθεί με Kerberos, LDAP και άλλα κεντρικά συστήματα διαχείρισης ταυτότητας [18].

Μερικά από τα πλεονεκτήματα που προσφέρει η χρήση του Ansible είναι η καθιέρωση και διατήρηση της συνέπειας των επιδόσεων του προϊόντος, η διατήρηση των φυσικών χαρακτηριστικών με τις απαιτήσεις και το σχεδιασμό του καθώς και η διατήρηση των επιχειρησιακών πληροφοριών καθ' όλη τη διάρκεια της ζωής του.

Πέρα από το Ansible, υπάρχουν και άλλα εργαλεία για την αυτοματοποίηση και παραμετροποίηση των μηχανημάτων όπως είναι το Puppet και το Chef.

Το Puppet είναι μια πρωτοποριακή λύση ανοιχτού κώδικα για τη ρύθμιση αυτοματισμού και ανάπτυξης ενοτήτων για εφαρμογές και υποδομές. Αναπτύχθηκε αρχικά από τον Luke Kanies για να αυτοματοποιήσει τις εργασίες για sysadmins οι οποίοι θα περνούσαν πολύ καιρό στη διαμόρφωση, την παροχή, την αντιμετώπιση προβλημάτων και τη διατήρηση των λειτουργιών του διακομιστή. Είναι κατασκευασμένη με Ruby και προσφέρει προσαρμοσμένες Domain Specific Language (DSL) και Embedded Ruby (ERB) για τη δημιουργία προσαρμοσμένων αρχείων γλώσσας Puppet, καθώς και μια προσέγγιση προγραμματισμού παραδειγματικού παραδείγματος. Χρησιμοποιεί μια αρχιτεκτονική agent/master. Ο agent διαχειρίζεται κόμβους και ζητά τις σχετικές πληροφορίες από τον master που ελέγχει τις πληροφορίες διαμόρφωσης. Επίσης, πραγματοποιεί αναφορές κατάστασης και ερωτήματα σχετικά με το συσχετισμένο μηχάνημα διακομιστή του από τον κύριο διακομιστή Puppet, ο οποίος στη συνέχεια κοινοποιεί την απόκριση και τις απαιτούμενες εντολές χρησιμοποιώντας το πρωτόκολλο XML-RPC μέσω HTTPS. Αυτός ο πόρος περιγράφει λεπτομερώς την αρχιτεκτονική. Οι χρήστες μπορούν επίσης να ρυθμίσουν μια ρύθμιση χωρίς master και αποκεντρωμένη Puppet, όπως περιγράφεται εδώ.

Ένα άλλο εργαλείο αυτοματοποίησης είναι το Chef, που ξεκίνησε ως ένα εσωτερικό εργαλείο ανάπτυξης στο διακομιστή OpsCode πριν να κυκλοφορήσει ως λύση ανοιχτού κώδικα. Χρησιμοποιεί επίσης μια αρχιτεκτονική πελάτη/διακομιστή και προσφέρει διαμόρφωση σε ένα Ruby DSL χρησιμοποιώντας το επιτακτικό προγραμματιστικό πρότυπο. Το ευέλικτο πλαίσιο αυτοματοποίησης της υποδομής του cloud επιτρέπει στους χρήστες να εγκαθιστούν εφαρμογές σε απλά, γυμνά VM και cloud containers. Η αρχιτεκτονική του είναι αρκετά παρόμοια με το μοντέλο master/agent του Puppet και χρησιμοποιεί μια προσέγγιση που βασίζεται στην έλξη, εκτός από το ότι απαιτείται ένας επιπλέον λογικός σταθμός εργασίας Chef για να ελέγχει τις διαμορφώσεις από τον πλοίαρχο σε πράκτορες. Οι agents διερευνούν τις πληροφορίες από τους κεντρικούς διακομιστές που ανταποκρίνονται μέσω του SSH. Διάφορα μοντέλα SaaS και υβριδικών παραδόσεων είναι διαθέσιμα για τη διαχείριση αναλυτικών στοιχείων και αναφορών.

Τα πλεονεκτήματα που προσφέρει το Ansible σε σχέση με τα άλλα δύο εργαλεία αυτοματοποίησης ανοιχτού κώδικα είναι:

- Εύκολη απομακρυσμένη εκτέλεση και χαμηλό εμπόδιο στην είσοδο.
- Κατάλληλο για περιβάλλοντα σχεδιασμένα να κλιμακώνονται γρήγορα.
- Μοιράζεται γεγονότα μεταξύ πολλών διακομιστών, ώστε να μπορούν να αλληλεπιδρούν.
- Ισχυρή μηχανή ενορχηστρώσεων. Ισχυρή εστίαση σε περιοχές όπου δεν υπάρχουν άλλοι, όπως μηδενικές διακοπές αναμονής σε εφαρμογές πολλαπλών επιπέδων σε όλο το cloud.
- Εύκολη εγκατάσταση και αρχική ρύθμιση.
- Εύκολη εκμάθηση της σύνταξης και της ροής εργασιών για νέους χρήστες.
- Διαδοχική εντολή εκτέλεσης.
- Υποστηρίζει μοντέλα ώθησης και έλξης.
- Η έλλειψη master εξαλείφει σημεία αποτυχίας και προβλήματα επιδόσεων. Η ανάπτυξη και η επικοινωνία χωρίς master είναι ταχύτερες από το μοντέλο master/agent.
- Υψηλή ασφάλεια με SSH [11].

Για τις ανάγκες τις διπλωματικής αναπτύξαμε μια σειρά από playbooks που βοηθούν στην αυτοματοποίηση της διαδικασίας των δοκιμών. Τα playbooks αφορούν και τις τρεις μονάδες της διάταξης του δικτύου μας. Οι λειτουργίες τους αφορούν αυτοματοποίηση της διαδικασίας περιήγησης στο Raspberry Pi, καθώς την εκκίνηση μιας εικονικής μηχανής στον ESXi για την εκτέλεση δοκιμών κίνησης δεδομένων.

3.8 Django

Το Django είναι ένα ελεύθερο και ανοιχτού κώδικα, υψηλού επιπέδου πλαίσιο ιστού βασισμένο σε Python, που ακολουθεί το αρχιτεκτονικό μοτίβο Model-View-Template (MVT). Ο πρωταρχικός στόχος του Django είναι να διευκολύνει τη δημιουργία πολύπλοκων δικτυακών τόπων που βασίζονται σε βάσεις δεδομένων. Το πλαίσιο υπογραμμίζει την επαναχρησιμοποίηση και την "δυνατότητα σύνδεσης" των εξαρτημάτων, το λιγότερο κώδικα, τη χαμηλή σύζευξη, την ταχεία ανάπτυξη και την αρχή του να μην επαναλαμβάνεις τον εαυτό σου. Η Python χρησιμοποιείται σε ολόκληρο το περιβάλλον, ακόμα και για αρχεία ρυθμίσεων και μοντέλα δεδομένων. Το Django παρέχει επίσης μια προαιρετική διεπαφή δημιουργίας, ανάγνωσης, ενημέρωσης και διαγραφής που δημιουργείται δυναμικά μέσω ενδοσκόπησης και διαμορφώνεται μέσω μοντέλων admin. Πρόκειται για ένα εργαλείο που χρησιμοποιείται ευρέως από μεγάλες εταιρίες με ιστοτόπους.

Είναι εξαιρετικά γρήγορο καθώς σχεδιάστηκε για αυτό το σκοπό, δηλαδή να διευκολύνει τους προγραμματιστές να εφαρμόζουν τις ιδέες τους στο μικρότερο δυνατό χρονικό διάστημα. Περιλαμβάνει μια μεγάλη ποικιλία από πρόσθετα πακέτα και βιβλιοθήκες που μπορούν να

χρησιμοποιηθούν για το χειρισμό πιο πολύπλοκων εργασιών στη διαδικασία ανάπτυξης μιας εφαρμογής ιστού. Επίσης, μεριμνά για την ασφάλεια τόσο των δεδομένων της εφαρμογής όσο και των ίδιων των χρηστών της, μέσω του ελέγχου ταυτότητας χρηστών που παρέχει. Διαθέτει επίσης τη δυνατότητα για διαχείριση περιεχομένου, χάρτες τοποθεσιών, ροές RSS και πολλές άλλες εργασίες. Επιπλέον, είναι εξαιρετικά κλιμακώσιμο και ευέλικτο, καθώς προσαρμόζεται εύκολα με μικρές αλλαγές σε πολλών ειδών εφαρμογές, εξυπηρετώντας από τις πιο μικρές μέχρι και τις πιο βαριές απαιτήσεις.

Η εναλλακτική που υπάρχει αντί για το Django είναι η PHP, μια γλώσσα ανάπτυξης διαδικτυακών εφαρμογών. Η PHP επιτρέπει στους προγραμματιστές να δημιουργήσουν δυναμικό περιεχόμενο που θα βοηθήσει στην αλληλεπίδραση με βάσεις δεδομένων. Αν και τα δύο, το Django και η PHP σχετίζονται με τις εφαρμογές web, υπάρχουν σημαντικές διαφορές μεταξύ τους.

Η βασική διαφορά μεταξύ των δύο βρίσκεται στον ορισμό των δύο αυτών εργαλείων. Το Django είναι ένα πλαίσιο ιστού ανοιχτού κώδικα γραμμένο σε Python που βοηθά στην ταχεία ανάπτυξη και συστηματικό σχεδιασμό. Το τμήμα κώδικα είναι επίσης λιγότερο στο πλαίσιο αυτό. Αντίθετα, η PHP (Hypertext Preprocessor) είναι μια γλώσσα ανάπτυξης που αναπτύχθηκε από τον Rasmus Lerdorf το 1994. Πρόκειται για μια scripting γλώσσα που μπορεί να ενσωματωθεί σε HTML και χρησιμοποιείται για τη διαχείριση δυναμικού περιεχομένου, βάσεων δεδομένων, εντοπισμού συνεδριών κλπ. Μπορεί να ενσωματωθεί με πολλές δημοφιλείς βάσεις δεδομένων όπως MySQL, Oracle, Microsoft SQL server κ.λπ.

Άλλες διαφορές μεταξύ των δύο είναι το περιβάλλον στο οποίο χρησιμοποιούνται καθώς και ο τρόπος δημιουργίας του project. Το Django είναι ένα πακέτο για την εγκατάσταση του οποίου απαιτείται μόνο η ύπαρξη της γλώσσας προγραμματισμού Python. Αντίθετα, η PHP απαιτεί την ύπαρξη τριών βασικών στοιχείων. Αυτά είναι: ένας web server, μια βάση δεδομένων και ένας αναλυτής για την παραγωγή των σελίδων της εφαρμογής σε HTML. Επίσης, το Django επιτρέπει την εκτέλεση απλών αλλά κύριων εργασιών όπως τη δημιουργία project με την εκτέλεση μιας απλής εντολής. Αντίθετα, η δημιουργία project με την PHP απαιτεί τη γνώση γλώσσας από πιο πριν καθώς και των κατάλληλων ετικετών που χρησιμοποιούνται για αυτή τη διαδικασία.

Τέλος, διαφέρουν και στον τρόπο με τον οποίο χειρίζονται τα διάφορα στατικά αρχεία. Το Django με τη χρήση των μοντέλων που διαθέτει, επιτρέπει τον χειρισμό οποιουδήποτε είδους αρχείου με τρόπο απλό και κατανοητό, χωρίς να μπερδεύει το χρήστη. Αντίθετα, η PHP δεν διαθέτει υποδομή για τον άμεσο χειρισμό αρχείων αλλά χρησιμοποιεί μια ενδιάμεση, προσωρινή διεύθυνση και μέσω scripting ανακατευθύνει το χρήστη στη σωστή τοποθεσία.

Σε ένα ευρύτερο πλαίσιο, το Django είναι ένα πιο εύχρηστο εργαλείο για την ανάπτυξη εφαρμογών ιστού. Επιτρέπει στο χρήστη να εκτελέσει διάφορες λειτουργίες πολύ γρήγορα, επιταχύνοντας τη διαδικασία ανάπτυξης της εφαρμογής του. Για τους παραπάνω λόγους έγινε και η συγκεκριμένη επιλογή για την υλοποίηση του front-end της web εφαρμογής μας.

4

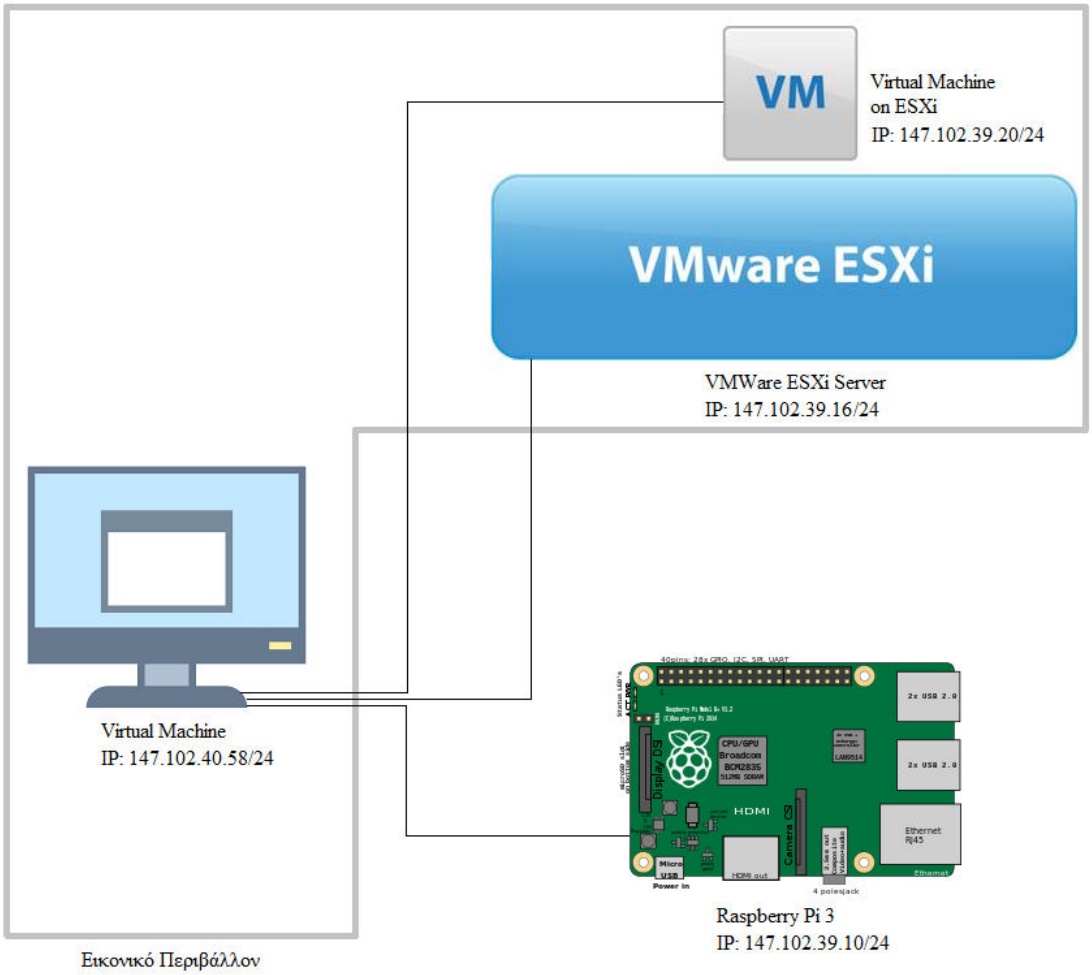
Ανάπτυξη Test Automation Framework

Στο παρόν κεφάλαιο θα γίνει ανάλυση της διαδικασίας που ακολουθήθηκε στο πειραματικό τμήμα της παρούσας διπλωματικής εργασίας. Αρχικά, θα παρουσιαστεί η διάταξη του δικτύου που αναπτύχθηκε και χρησιμοποιήθηκε για τις δοκιμές, αναλύοντας τις λειτουργίες και τα χαρακτηριστικά της κάθε μονάδας του ξεχωριστά. Στη συνέχεια, θα αναλυθεί ο τρόπος δημιουργίας της διάταξης με χρήση των τεχνολογιών και των εργαλείων που αναλύθηκαν στα προηγούμενα κεφάλαια. Τέλος, θα παρουσιαστεί η ορθή λειτουργία της εφαρμογής.

4.1 Περιγραφή Διάταξης

Για την υλοποίηση του θέματος της διπλωματικής σε εργαστηριακό περιβάλλον με στόχο την επιβεβαίωσή του, αποφασίστηκε η δημιουργία δύο εικονικών μηχανών (Virtual Machine - VM) καθώς και η χρήση ενός Raspberry Pi 3. Οι εικονικές μηχανές έχουν διαφορετική λειτουργία η καθεμιά στη διάταξη. Η πρώτη λειτουργεί ως ενορχηστρωτής της εφαρμογής, καθώς από αυτή ξεκινούν όλες οι εντολές για τις αυτοματοποιημένες δοκιμές και είναι επίσης ο οικοδεσπότης (host) ολόκληρης της web-εφαρμογής. Η δεύτερη αποτελεί έναν ESXi Server που χρησιμοποιείται σε ορισμένες δοκιμές κίνησης. Το Raspberry Pi είναι το τελευταίο άκρο της διάταξης και είναι ουσιαστικά ο αποδέκτης των αυτοματοποιημένων δοκιμών περιήγησης στο διαδίκτυο. Ο λόγος χρήσης των εικονικών μηχανών αλλά και του Raspberry Pi είναι ότι στη διάταξη αυτή μπορούν να εφαρμοστούν χωρίς πρόβλημα οι τεχνικές αυτοματοποίησης της διαδικασίας περιήγησης στο διαδίκτυο, καθώς επίσης και ότι επιτρέπει και την πραγματοποίηση δοκιμών ποικίλης κίνησης.

Επιπλέον, το γεγονός ότι όλα τα μηχανήματα βρίσκονται στο ίδιο εργαστηριακό περιβάλλον μας επιτρέπει να γνωρίζουμε από την αρχή τη διαδρομή που θα ακολουθήσει η δρομολόγηση των πακέτων κατά την εξακρίβωση της ορθής λειτουργιάς της εφαρμογής. Προφανώς, θα μπορούσε να χρησιμοποιηθεί και άλλου τύπου διάταξη όπως πχ. άλλη συσκευή αντί του Raspberry Pi για την κίνηση από το διαδίκτυο. Επίσης, η διάταξη αυτή μας δίνει τη δυνατότητα να τη μεταφέρουμε και σε οποιοδήποτε άλλο εικονικό περιβάλλον, αλλάζοντας μόνο τις διευθύνσεις IP. Στη συγκεκριμένη περίπτωση, η εικονική μηχανή που λειτουργεί ως ενορχηστρωτής έχει τη δημόσια διεύθυνση 147.102.48.54/24 και το όνομα *jenkins.cn.ece.ntua.gr*. Το Raspberry Pi έχει τη διεύθυνση 147.102.39.10/24 ενώ η εικονική μηχανή που λειτουργεί ως ESXi Server έχει τη διεύθυνση 147.102.39.16/24.



Εικόνα 4-1: Πειραματική διάταξη

Όπως γίνεται αντιληπτό, η σημαντικότερη μονάδα της όλης διάταξης είναι η εικονική μηχανή που λειτουργεί και ως ενορχηστρωτής αλλά περιλαμβάνει επίσης και όλον τον κώδικα για το user interface του site μας. Από εδώ και στο εξής θα αναφερόμαστε σε αυτή με το όνομά της, δηλαδή *jenkins.cn.ece.ntua.gr* ή απλά ως Ενορχηστρωτή. Σε αυτή την εικονική μηχανή στεγάζεται ολόκληρη η web εφαρμογή στην οποία μπορούν να συνδεθούν οι χρήστες μέσω ενός φυλλομετρητή. Όταν ένας χρήστης θελήσει να πραγματοποιήσει κάποια από της υποστηριζόμενες δοκιμές, αφού

πληκτρολογήσει τη διεύθυνση web της εφαρμογής, έχει πρόσβαση στη διεπαφή χρήστη και μπορεί να δημιουργήσει τις δικές του μοναδικές δοκιμές. Επίσης, αποτελεί το σημείο δημιουργίας και αποστολής των δοκιμών στις υπόλοιπες μονάδες της διάταξης, το Raspberry Pi και τον ESXi Server. Περιλαμβάνει όλον τον κώδικα για τη λειτουργία της εφαρμογής αλλά και αυτόν που αυτοματοποιεί τη διαδικασία των δοκιμών.

Το Raspberry Pi 3 αποτελεί τον αποδέκτη και εκτελεστή των δοκιμών περιήγησης στο διαδίκτυο με σκοπό την παρακολούθηση της κίνησης των δεδομένων. Βρίσκεται και αυτό στο περιβάλλον του εργαστηρίου, όπως μπορούμε να αποφανθούμε και από τη δομή της διεύθυνσης IP που έχει. Το Raspberry Pi αφού δεχθεί κάποια εντολή για εκτέλεση της διαδικασίας αυτοματοποιημένης περιήγησης στο διαδίκτυο, εκτελεί τη ζητούμενη διεργασία εικονικά, αφού δεν διαθέτει οπτική συσκευή. Πριν την εκτέλεση της, ξεκινάει και την απαραίτητη καταγραφή των πακέτων που διακινούνται κατά τη διάρκεια εκτέλεσης της δοκιμής.

Τέλος, ο ESXi Server χρησιμοποιείται κυρίως στις δοκιμές παρακολούθησης απλής κίνησης δεδομένων. Συγκεκριμένα, ο χρήστης μπορεί να επιλέξει από τη διεπαφή χρήστη τον αντίστοιχο τύπο δοκιμών καθώς και τις παραμέτρους της αρεσκείας του. Ο Ενορχηστρωτής δημιουργεί την ανάλογη εντολή, αφού πρώτα έχει δημιουργήσει και εκκινήσει έναν αντικειμενικό server στον ESXi και αφού έχει εκκινηθεί η αντίστοιχη εντολή στην πλευρά του server.

4.2 Django Framework

Για τη δημιουργία της διεπαφής του χρήστη με την εφαρμογή δοκιμής της τεχνολογίας NFV, επιλέχθηκε η χρησιμοποίηση του Django Framework. Οι λόγοι για τους οποίους επιλέχθηκε καθώς και τα χαρακτηριστικά του Django έχουν αναλυθεί σε προηγούμενο κεφάλαιο της παρούσας διπλωματικής. Πρόκειται για ένα framework γραμμένο στη γλώσσα προγραμματισμού Python, που έχει ως στόχο την ευκολότερη και γρηγορότερη ανάπτυξη web εφαρμογών. Επίσης προσφέρει ένα ικανοποιητικό επίπεδο ασφάλειας και είναι εξαιρετικά επεκτάσιμο. Ουσιαστικά, μέσω του Django χτίστηκε όλη η web εφαρμογή που επιτρέπει στο χρήστη την εκτέλεση δοκιμών με αυτοματοποιημένο τρόπο και την εξαγωγή συμπερασμάτων από αυτά.

Η δομή του site που χτίσαμε βασίζεται στις δοκιμές που είναι ικανός να πραγματοποιήσει ο χρήστης. Ο στόχος είναι η εκτέλεση τεσσάρων τύπων δοκιμών, μιας για δημιουργία κίνησης με χρήση της εντολής *iperf3* και τριών με τη χρήση του Selenium που θα αφορούν βασικές εργασίες περιήγησης όπως φόρτωση μιας σελίδας HTTP, αναζήτηση ενός όρου στο Google καθώς και σύνδεση στο λογαριασμό μας στο Facebook.

Τα tests αναπαριστώνται στο Django ως μοντέλα αντικειμένων, μαζί με τις παραμέτρους που χρειάζεται το καθένα για την εκτέλεσή του. Παράλληλα, υπάρχει και ίδιος αριθμός αντικειμένων για τα αποτελέσματα από τον κάθε τύπο test. Οι βασικές σελίδες που προσφέρει το site είναι τρεις:

- Μια σελίδα δημιουργίας test. Σε αυτήν ο χρήστης μπορεί να δημιουργήσει τα δικά του tests με τις παραμέτρους που ο ίδιος επιλέγει κάθε φορά.

- Μια σελίδα προβολής και επεξεργασίας όλων των δοκιμών που έχει πραγματοποιήσει ο χρήστης. Αυτή είναι και η αρχική σελίδα της εφαρμογής.

- Μια σελίδα προβολής των αποτελεσμάτων από τα διάφορα tests του χρήστη. Στη συγκεκριμένα σελίδα βρίσκονται αποτελέσματα από όλες τις εκτελέσεις της κάθε δοκιμής.

Επίσης, στο site έχει γίνει ανάπτυξη ενός συστήματος ταυτοποίησης χρηστών, ώστε ο κάθε χρήστης που συνδέεται να έχει τον πλήρη έλεγχο των δοκιμών του.

Για τη δημιουργία της εφαρμογής, το Django προσφέρει τη δυνατότητα στον προγραμματιστή με την εκτέλεση απλών εντολών στο τερματικό να δημιουργήσει γρήγορα την εφαρμογή που επιθυμεί. Στη συγκεκριμένη περίπτωση, οι εντολές που εκτελέστηκαν για τη δημιουργία της ήταν:

```
$ django-admin startproject TestAutomation
```

Με την παραπάνω εντολή δημιουργήσαμε το project στο οποίο αναπτύξαμε την εφαρμογή μας. Ουσιαστικά, δημιουργήσαμε αυτόματα ένα Django project, το οποίο δεν είναι τίποτε άλλο παρά ένας φάκελος με αρχεία που επιτελούν διάφορες ρυθμίσεις, όπως ρυθμίσεις της βάσης δεδομένων, διάφορες επιλογές του Django και ρυθμίσεις που αφορούν την εφαρμογή μας.

Ο δοκιμαστικός εξυπηρετητής (server) που προσφέρει το Django χρησιμοποιείται για καθαρά αναπτυξιακούς σκοπούς και δεν είναι ικανός να εξυπηρετήσει μεγάλες εμπορικές εφαρμογές. Σε αυτή την περίπτωση θα πρέπει να μεταφέρουμε την εφαρμογή μας σε κάποιον άλλο server όπως είναι ο Apache για παράδειγμα. Επίσης, η ανάπτυξη μιας τέτοιας εφαρμογής με PHP που αναλύθηκε σε προηγούμενο κεφάλαιο, θα είχε επιπτώσεις στην ασφάλεια των δεδομένων των χρηστών αλλά και της ίδιας της εφαρμογής. Ο Django development server είναι ένας ελαφρύς Web server γραμμένος αμιγώς σε Python. Μέσω αυτού έχουμε τη δυνατότητα να βλέπουμε και να αναπτύσσουμε το project μας γρήγορα, χωρίς να χρειάζεται να ασχολούμαστε με ρυθμίσεις του production server, όπως του Apache που αναφέραμε νωρίτερα.

Στη συνέχεια, δημιουργήσαμε την εφαρμογή μας αυτόματα με την επιλογή που μας δίνει το Django μέσω της εκτέλεσης της παρακάτω εντολής και ενώ βρισκόμαστε στο directory του project μας με τίτλο «*TestAutomation*», στο οποίο υπάρχει και το αρχείο ***manage.py***, ένα αρχείο κονσόλας που μας επιτρέπει την αλληλεπίδραση με το project για διάφορες ενέργειες:

```
$ python manage.py startapp test-automation
```

Εφόσον έχουμε αρχικοποιήσει την εφαρμογή μας, είμαστε σε θέση να την αναπτύξουμε ακόμα περισσότερο, γράφοντας τα μοντέλα, τις σελίδες και γενικά οτιδήποτε άλλο χρειάζεται η εφαρμογή μας για να είναι σύμφωνη με τις επιθυμίες μας. Η δομή της εφαρμογής βασίζεται σε μοντέλα

αντικειμένων που αναπαριστούν τις δοκιμές που είναι ικανή να εκτελέσει η εφαρμογή. Συγκεκριμένα, έχουμε ορίσει τέσσερεις τύπους δοκιμών και αντίστοιχα τέσσερεις τύπους αποτελεσμάτων. Έχουμε τρεις τύπους δοκιμών για αυτοματοποιημένη περιήγηση στο διαδίκτυο καθώς και τους αντίστοιχους τρεις τύπους αποτελεσμάτων. Επιπλέον, έχουμε και έναν τύπο δοκιμής για μεταφορά κίνησης δεδομένων. Οι ονομασίες που τους έχουμε δώσει αντικατοπτρίζουν και τη λειτουργία που επιτελεί το κάθε test. Έτσι, έχουμε τα εξής μοντέλα για τις δοκιμές: *TrafficTest, PageTest, SearchTest* και *LoginTest* καθώς και τα αντίστοιχα μοντέλα για τα αποτελέσματα: *ResultsTraffic, ResultsPage, ResultsSearch, ResultsLogin*. Η αλληλοσυσχέτιση των εκάστοτε μοντέλων μεταξύ τους γίνεται με τη χρήση του *ForeignKey*. Πρόκειται για μια συσχέτιση τύπου one-to-many που λέει στο Django ποια αποτελέσματα αντιστοιχούν σε κάθε test, χρησιμοποιώντας το ID του κάθε test. Αυτό σημαίνει επίσης ότι ένα αντικείμενο αποτελεσμάτων δε μπορεί να αντιστοιχιστεί με περισσότερα από ένα tests. Το Django υποστηρίζει όλα τα είδη των κοινών συσχετίσεων μεταξύ βάσεων δεδομένων: many-to-one, many-to-many και one-to-one. Τις αλλαγές αυτές τις κάναμε στο αρχείο ***models.py*** που βρίσκεται στο directory της εφαρμογής «*test-automation*» και στο οποίο φαίνεται ο ορισμός του κάθε μοντέλου καθώς και οι παράμετροι του καθενός που είναι και αυτές που μπορεί να επιλέξει ο χρήστης για την εκτέλεση των δοκιμών. Ο κώδικας του ***models.py*** βρίσκεται στο παράρτημα με τίτλο «**Κώδικας**».

Επόμενο βήμα ήταν η δημιουργία των σελίδων που θα φιλοξενεί την εφαρμογή μας. Όπως είπαμε και προηγουμένως, έχουμε τρεις κύριες σελίδες στο site που αναπτύξαμε. Η πρώτη περιλαμβάνει τη σελίδα στην οποία προβάλλονται συνοπτικά όλα τα tests έχει δημιουργήσει ο χρήστης. Σε αυτήν υπάρχουν οι επιλογές της *Επεξεργασίας*, *Διαγραφής* και *Εκτέλεσης* της κάθε δοκιμής. Η δεύτερη σελίδα είναι η σελίδα των αποτελεσμάτων. Σε αυτήν αποτυπώνονται τα αποτελέσματα από την κάθε εκτέλεση μιας δοκιμής. Η τρίτη σελίδα τέλος είναι αυτή της δημιουργίας ενός test, επιτρέποντας στο χρήστη να ορίσει στις παραμέτρους τις τιμές της αρεσκείας του. Για την ανάπτυξη των σελίδων, χρειάζεται αφενός ο κώδικας τους σε HTML (Hyper Text Markup Language), η κύρια γλώσσα σήμανσης για τις ιστοσελίδες και αφετέρου η επεξεργασία του αρχείου ***views.py*** που βρίσκεται και αυτό στο directory της εφαρμογής με τίτλο «*test-automation*». Τα διάφορα HTMLs καθώς και οποιοδήποτε άλλο στατικό αρχείο, όπως το stylesheet γραμμένο σε CSS βρίσκεται στο αντίστοιχο directory για «static» files. Το Django μας δίνει τη δυνατότητα να καλέσουμε αυτά τα αρχεία πολύ γρήγορα, αναφέροντας απλά το όνομά τους και το όνομα του app μας, όπως φαίνεται και στον κώδικα των σελίδων HTMLs που υπάρχει στο παράρτημα. Στο αρχείο ***views.py*** έχουν υλοποιηθεί όλες οι μέθοδοι που απαιτούνται για τις λειτουργίες που προαναφέραμε. Επίσης, όπως μπορεί να γίνει εύκολα κατανοητό, σε αυτό το αρχείο βρίσκονται και οι κλήσεις των εντολών που απαιτούνται για την εκτέλεση των δοκιμών. Παράλληλα, για το γραφικό περιβάλλον της σελίδας έγινε χρήση του Bootstrap, ενός εργαλείο ανοιχτού κώδικα για την ανάπτυξη με HTML, CSS και JavaScript. Ο κώδικας του ***views.py*** καθώς και τα αρχεία HTMLs των σελίδων βρίσκονται επίσης στο παράρτημα με τίτλο «**Κώδικας**».

Στη συνέχεια, ορίσαμε τα URLs που θα χρησιμοποιεί η εφαρμογή δημιουργώντας το αρχείο ***urls.py*** που βρίσκεται στο directory της εφαρμογής «*test-automation*». Τα URLs ουσιαστικά είναι patterns τα οποία θα εμφανίζονται στο επάνω μέρος του περιηγητή μας και θα υποδηλώνουν σε ποια από τις σελίδες του site βρισκόμαστε. Συνδέονται με τα views που έχουμε ορίσει στο αρχείο προηγουμένως καθώς κάθε URL πρέπει να είναι άμεσα συσχετισμένο με ένα view και το αντίστροφο. Ο κώδικας του ***urls.py*** βρίσκεται επίσης στο παράρτημα με τίτλο «**Κώδικας**». Για να προσθέσουμε τα URLs μας στην εφαρμογή θα χρειαστεί να προσθέσουμε την κατάλληλη εντολή στο αρχείο ***urls.py*** που βρίσκεται στο directory με το όνομα του project μας, στη δική μας περίπτωση στο «TestAutomation» και είναι διαφορετικό από το αρχείο με το ίδιο όνομα που αναφέραμε προηγουμένως. Επίσης, είναι σημαντικό να προσθέσουμε και μια ανάλογη εντολή για τα αρχεία που θα προβάλλει το site μας και προέρχονται από τον χρήστη. Τέτοια αρχεία δημιουργούνται έπειτα από την εκτέλεση των δοκιμών και αφορούν αρχεία αναφορών και καταγραφών. Οι αλλαγές που κάναμε σε αυτό το αρχείο φαίνονται παρακάτω:

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('test-automation/', include('test-automation.urls'))
]


urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Για την ανάπτυξη των σελίδων δημιουργίας νέων δοκιμών δημιουργήσαμε και το αρχείο ***forms.py*** στο ίδιο directory με τα προηγούμενα αρχεία που αναφέραμε προηγουμένως. Το συγκεκριμένο αρχείο μας επιτρέπει να δημιουργήσουμε αυτόματα τις φόρμες καθώς και τα πεδία να κατάλληλα σχεδιασμένα για την είσοδο που πρόκειται να λάβουν (για παράδειγμα οι κωδικοί να εμφανίζονται στην οθόνη με αστερίσκους).

Όσον αφορά τη βάση δεδομένων που χρησιμοποιεί το project μας στο Django και στην οποία αποθηκεύονται όλες οι αλλαγές που κάνουμε τόσο στα αρχεία κώδικα όσο και στα tests μέσω της διεπαφής χρήστη, αυτή είναι η προεπιλεγμένη SQLite. Η SQLite περιλαμβάνεται στην Python, με αποτέλεσμα να μην χρειάζεται να εγκαταστήσουμε τίποτε άλλο για να υποστηρίξουμε τη βάση δεδομένων μας. Οι ρυθμίσεις για τη βάση δεδομένων βρίσκονται στο αρχείο ***settings.py***, που βρίσκεται στο directory με το όνομα του project μας, δηλαδή στο «TestAutomation», εκεί δηλαδή που βρίσκεται και το αρχείο ***urls.py*** που αναφέραμε προηγουμένως. Το αρχείο έχει δημιουργηθεί αυτόματα από την εκτέλεση των προηγούμενων εντολών και αποτελεί την καρδιά του project μας. Αποτελεί ένα κοινό Python module στου οποίου οι μεταβλητές (variables) αναπαριστούν τις ρυθμίσεις του Django. Οι αλλαγές που έχουμε κάνει στο αρχείο των ρυθμίσεων αφορούν ρυθμίσεις για τη ζώνη ώρας στην οποία βρισκόμαστε, για την ενεργοποίηση της εφαρμογής μας αλλά και για το όνομα του

directory στο οποίο θα βρίσκει το Django τα static files μας, όπως αρχεία εικόνων, CSS stylesheets και άλλα στατικά αρχεία. Επίσης, ορίσαμε και άλλες δύο μεταβλητές που αφορούν τα αρχεία που θα επιστρέφονται από την εκτέλεση της κάθε δοκιμής και θα εμφανίζονται ως αποτελέσματα. Πρόκειται για αρχεία αναφορών είτε σε .json είτε σε HTML καθώς και τα αρχεία καταγραφών σε .pcap. Οι μεταβλητές αυτές απαιτούν την ύπαρξη μιας διεύθυνσης με το όνομα «media», στην οποία βρίσκονται όλα τα αρχεία που αναφέραμε πιο πριν. Οι αλλαγές αυτές παρουσιάζονται παρακάτω:

```
ALLOWED_HOSTS = ['jenkins.cn.ece.ntua.gr','jenkins.cn.ntua.gr', 'localhost', '127.0.0.1']
TIME_ZONE = 'Europe/Athens'
STATIC_URL = '/static/'
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, "media")
```

Για τη δημιουργία της βάσης δεδομένων απαιτείται η εκτέλεση της παρακάτω εντολής προτού χρησιμοποιήσουμε την εφαρμογή μας:

```
$ python manage.py makemigrations test-automation
```

Η εντολή *migrate* κοιτάζει στη ρύθμιση INSTALLED_APPS και δημιουργεί τυχόν απαραίτητους πίνακες, βάση της ρύθμισης DATABASES του αρχείου ***settings.py*** που αναφέραμε προηγουμένως και βάσει των database migrations τα οποία υπάρχουν σε κάθε εφαρμογή που έχουμε συμπεριλάβει. Τα migrations αναπαριστούν στο Django τις αλλαγές στα μοντέλα μας και συνεπώς, στο schema της βάσης δεδομένων μας δεν είναι τίποτε άλλο παρά αρχεία κώδικα σε Python.

Στη συνέχεια απαιτείται η εκτέλεση άλλης μιας εντολής πριν να είμαστε σε θέση να εκκινήσουμε την εφαρμογή μας. Η εντολή αυτή λέει στο Django ότι έχουμε κάνει κάποιες αλλαγές στα μοντέλα μας και ότι θα θέλαμε αυτές οι αλλαγές να αποθηκευτούν ως νέο migration:

```
$ python manage.py migrate
```

Και οι δύο αυτές εντολές για να εκτελεστούν πρέπει να βρισκόμαστε στο directory του project μας, δηλαδή στο «TestAutomation», στο οποίο βρίσκεται και το αρχείο ***manage.py***. Εδώ πρέπει να τονίσουμε ότι κάθε φορά που κάνουμε κάποια αλλαγή στα μοντέλα, στα views, στα URLs ή σε οποιοδήποτε άλλο σημείο της εφαρμογής μας, πρέπει οπωσδήποτε να εκτελούμε τις δύο παραπάνω εντολές.

Για να ξεκινήσουμε τώρα την εφαρμογή μας, εκτελούμε την παρακάτω εντολή, στην οποία έχουμε ορίσει και το URL στο οποίο θέλουμε να εμφανίζεται το site για την εφαρμογή μας:

```
$ python manage.py runserver jenkins.cn.ece.ntua.gr:8000
```

Η έξοδος που βλέπουμε τώρα στην κονσόλα μας είναι η παρακάτω και σημαίνει πως η εφαρμογή μας είναι σε λειτουργία:

```
Performing system checks...

System check identified no issues (0 silenced).
January 29, 2019 - 19:14:43
Django version 2.1.4, using settings 'testAutomation.settings'
Starting development server at http:// jenkins.cn.ece.ntua.gr:8000/
Quit the server with CTRL-BREAK.
```

4.3 Tests

Για την εξακρίβωση της σωστής λειτουργίας της εφαρμογής αλλά και την αξιολόγησής της, δημιουργήσαμε μερικούς τύπους δοκιμών (tests) τα οποία μπορεί να εκτελέσει ο χρήστης από το σημείο διεπαφής που δημιουργήσαμε και αναλύσαμε προηγουμένως. Τα tests αυτά αφορούν τόσο την αυτοματοποίηση της διαδικασίας περιήγησης στο διαδίκτυο όσο και τη δοκιμή της κίνησης δεδομένων. Συγκεκριμένα ο χρήστης, καθώς εισέρχεται στον ιστότοπο που δημιουργήσαμε και αφού επιβεβαιωθεί η ταυτότητά του, μπορεί να δημιουργήσει τα δικά του tests με τις παραμέτρους της αρεσκείας του. Ο ιστότοπός μας υποστηρίζει τέσσερις τύπους δοκιμών, τρεις που αφορούν την περιήγηση στο διαδίκτυο και έναν που αφορά την κίνηση δεδομένων.

Τα tests που αφορούν την περιήγηση στο διαδίκτυο είναι γραμμένα σε Python και χρησιμοποιούν τη βιβλιοθήκη *unittest* για να συμπεριφέρονται ως tests. Το πλαίσιο δοκιμής ανά μονάδα *unittest* αρχικά εμπνεύστηκε από το JUnit και προσφέρει παρόμοια εμπειρία με τα μεγάλα πλαίσια δοκιμών μονάδων σε άλλες γλώσσες. Υποστηρίζει την αυτοματοποίηση δοκιμών, την κοινή χρήση του κώδικα εγκατάστασης και τερματισμού λειτουργίας για δοκιμές, τη συγκέντρωση δοκιμών σε συλλογές και την ανεξαρτησία των δοκιμών από το πλαίσιο αναφοράς. Επίσης, στον κώδικα των δοκιμών χρησιμοποιείται το *Selenium* για τον ορισμό των ρυθμίσεων του περιβάλλοντος στο οποίο θα εκτελεστεί ο κώδικας, αλλά και της πορείας που θα ακολουθηθεί κατά τη διάρκεια εκτέλεσης του test στον περιηγητή. Επίσης, χρησιμοποιούμε το *HtmlTestRunner*, μια επέκταση του πακέτου *unittest* της πρότυπης βιβλιοθήκης της Python που αναλύσαμε παραπάνω και το οποίο δημιουργεί εύχρηστες αναφορές των δοκιμών σε HTML.

Το *Selenium* απαιτεί τη σύνδεση ενός προγράμματος οδήγησης με το επιλεγμένο πρόγραμμα περιήγησης. Ο Firefox, για παράδειγμα, απαιτεί geckodriver, ο οποίος πρέπει να εγκατασταθεί πριν την εκτέλεση των δοκιμών. Εφόσον οι δοκιμές θα εκτελεστούν σε διαφορετικό μηχάνημα από αυτό

στο οποίο βρίσκεται ο κώδικας, ο Firefox και ο geckodriver βρίσκονται ήδη σε αυτό, που δεν είναι άλλο από το Raspberry Pi.

Οι τρεις περιπτώσεις περιήγησης που έχουμε προσομοιώσει στις δοκιμές μας είναι η κλήση μιας απλής σελίδας ιστοτόπου, η αναζήτηση ενός όρου ή φράσης στη μηχανή αναζήτησης της Google, καθώς και η σύνδεση του χρήστη στο Facebook, πληκτρολογώντας το mail και τον κωδικό του. Και στις τρεις περιπτώσεις, ο κώδικάς μας ακολουθεί την ίδια λογική. Πρώτα υπάρχει μία μέθοδος στην οποία αρχικοποιείται η δοκιμή, σύμφωνα με τις συνθήκες του περιβάλλοντος στο οποίο θα λάβει χώρα αλλά και άλλων ειδικών συνθηκών. Στην περίπτωσή μας, ορίζεται ως περιηγητής ο Mozilla Firefox και ως λειτουργικό σύστημα το LINUX μιας και αυτές είναι οι συνθήκες που επικρατούν στο περιβάλλον του Raspberry Pi όπου θα πραγματοποιηθούν οι δοκιμές.

Στη συνέχεια, υπάρχει η μέθοδος που καθορίζει την πορεία που θα ακολουθήσει η δοκιμή. Αυτή η μέθοδος είναι διαφορετική για τις τρεις περιπτώσεις δοκιμών που έχουμε προσομοιώσει. Όμως η λογική που ακολουθείται και στις τρεις είναι αυτή που θα έκανε και ο χρήστης αν έκανε ο ίδιος την περιήγηση χειροκίνητα. Πιο συγκεκριμένα στην περίπτωση της απλής σελίδας ιστοτόπου, ο χρήστης πληκτρολογεί απλά το URL της σελίδας στην οποία θέλει να κατευθυνθεί, εφόσον έχει ανοίξει ένα παράθυρο του περιηγητή του, και επιβεβαιώνει πατώντας ENTER. Ακριβώς την ίδια διαδικασία προσομοιώνουμε και στον κώδικά μας. Παρόμοια διαδικασία ακολουθείται και για τις άλλες δύο περιπτώσεις δοκιμών για αυτοματοποιημένη διαδικασία περιήγησης.

Τέλος, υπάρχει η μέθοδος που τερματίζει τη διαδικασία περιήγησης, κλείνοντας το παράθυρο του εικονικού περιηγητή στον οποίο έχουμε τρέξει τις δοκιμές μας. Αυτή η μέθοδος είναι κοινή και για τους τρεις τύπους δοκιμών που αναλύσαμε προηγουμένως. Τα αποτελέσματα επιστρέφονται με τη βοήθεια της βιβλιοθήκης HtmlTestRunner σε μορφή HTML σελίδας για να είναι σε θέση ακόμα και ένας απλός χρήστης υπολογιστή να καταλάβει αν το test που έτρεξε ήταν επιτυχές ή όχι.

Ο τέταρτος και τελευταίος τύπος δοκιμών που δημιουργήσαμε αφορά την αποστολή κίνησης δεδομένων από τον Ενορχηστρωτή στον ESXi Server. Σε αυτόν τον τύπος δοκιμών χρησιμοποιούμε την εντολή *iperf3* για να μεταφέρουμε δεδομένα στον ESXi Server. Το iPerf3 είναι ένα εργαλείο για ενεργές μετρήσεις του μέγιστου επιτεύξιμου εύρους ζώνης σε δίκτυα IP. Υποστηρίζει τη ρύθμιση διαφόρων παραμέτρων που σχετίζονται με το χρονισμό, τα buffers και τα πρωτόκολλα (TCP, UDP) καθώς και η επιλογή για χρήση IPv4 ή IPv6. Για κάθε δοκιμή αναφέρει το εύρος ζώνης, την απώλεια και άλλες παραμέτρους. Το iPerf3 αναπτύχθηκε για πρώτη φορά από το NLANR / DAST.

Αρχικά, γίνεται εκκίνηση του iPerf3 ως Server στην πλευρά του ESXi ώστε να δέχεται την κίνηση δεδομένων και να την πετάει, επιστρέφοντάς μας τα αποτελέσματα. Στη συνέχεια εκκινείται το iPerf3 στην πλευρά του Ενορχηστρωτή με την ιδιότητα του client, δηλαδή δημιουργεί την κίνηση την οποία στέλνει στον Server που ορίζεται κάθε φορά. Η επιλογή του Server στην περίπτωσή μας είναι πάντα η ίδια, δηλαδή ο ESXi Server, ενώ οι παράμετροι έχουν οριστεί από το χρήστη όταν δημιούργησε το test. Τα αποτελέσματα από τις δοκιμές καταγράφονται σε αρχείο σε μορφή .json το οποίο γίνεται διαθέσιμο στο χρήστη μέσω του site μας μετά το πέρας της εκάστοτε δοκιμής.

4.4 Ansible Playbooks

Για την αυτοματοποίηση της διαδικασίας δοκιμών στην παρούσα διπλωματική, χρησιμοποιήσαμε το λογισμικό ανοιχτού κώδικα Ansible. Οι λόγοι που προτιμήθηκε το Ansible έναντι των υπόλοιπων λογισμικών ανοιχτού κώδικα που μας επιτρέπουν την αυτοματοποίηση και την ενορχήστρωση της διαδικασίας εξηγούνται αναλυτικά σε προηγούμενο κεφάλαιο. Στα πλαίσια του θέματος της διπλωματικής, αναπτύξαμε διάφορα playbooks τα οποία είναι γραμμένα στη γλώσσα YAML (YAML Ain't Markup Language), μια γλώσσα που μπορεί να αναγνωσθεί από ανθρώπους για τη σειριακή επεξεργασία δεδομένων και συνήθως χρησιμοποιείται για αρχεία ρυθμίσεων. Στο παράρτημα της παρούσας εργασίας με τίτλο «**Κώδικας**» βρίσκεται αναλυτικά ο κώδικας όλων των playbooks που δημιουργήσαμε για τις ανάγκες του θέματος.

Για την εκτέλεση των playbooks, απαιτείται η σύνδεση μέσω SSH στο μηχάνημα στο οποίο θα εκτελεστεί ο κώδικας που περιλαμβάνει το κάθε playbook, αν δεν πρόκειται για το ίδιο, καθώς και να έχει εγκατεστημένο το Ansible. Για το λόγο αυτό μια καλή πρακτική είναι η δημιουργία ενός κλειδιού SSH και η μεταφορά του δημόσιου κλειδιού στα μηχανήματα με τα οποία πρόκειται να γίνει η σύζευξη για την εκτέλεση των playbooks. Η εκτέλεσή τους γίνεται με την εντολή:

```
$ ansible-playbook $playbook [--extra-vars='$var1-name=$var1-value $var2-name=$var2-value ...']
```

Στην παραπάνω εντολή, η μεταβλητή *$playbook* υποδηλώνει το όνομα του playbook που θέλουμε να εκτελέσουμε κάθε φορά. Επίσης, το playbook για την εκτέλεσή του μπορεί να απαιτεί την είσοδο κάποιων επιπλέον προαιρετικών ή υποχρεωτικών μεταβλητών. Αυτές προστίθενται μετά την επιλογή *extra-vars* με το όνομα ακολουθούμενο από την τιμή τους.

Τα playbooks που δημιουργήσαμε χωρίζονται σε τρεις κατηγορίες, ανάλογα με το μηχάνημα στο οποίο πρόκειται να εκτελεστούν. Έτσι, έχουμε τα playbooks που πρόκειται να εκτελεστούν τοπικά στο μηχάνημα Ενορχηστρωτής, τα playbooks που πρόκειται να εκτελεστούν στο Raspberry Pi 3 και τέλος τα playbooks που πρόκειται να εκτελεστούν στον ESXi Server.

4.4.1 Playbooks για τον Ενορχηστρωτή

Για τον Ενορχηστρωτή δημιουργήσαμε πέντε playbooks, τα ***selenium-grid-hub.yml***, ***visualization.yml***, ***selenium-hub-close.yml***, ***start-command.yml*** και ***staert-server-url.yml***.

Το πρώτο αφορά την εκκίνηση του standalone Selenium Server μέσω του .jar αρχείου που έχουμε κατεβάσει από το site του Selenium, www.seleniumhq.org. Η εκτέλεσή του γίνεται με τη βοήθεια της Java και έχει σαν αποτέλεσμα τη δημιουργία ενός hub στο οποίο θα συνδέονται όλοι οι κόμβοι (nodes). Το hub είναι το κεντρικό σημείο που λαμβάνει όλα τα αιτήματα δοκιμής και τα διανέμει στους σωστούς κόμβους. Στη δική μας περίπτωση τα αιτήματα δοκιμής θα προέρχονται από το ίδιο μηχάνημα στο οποίο λειτουργεί το hub, τον Ενορχηστρωτή και η διαβίβασή τους γίνεται στο Raspberry Pi 3. Χρησιμοποιεί το module shell το οποίο απαιτεί τη δήλωση της εντολής όπως ακριβώς

θα την πληκτρολογούσαμε σε ένα τερματικό. Στην εντολή δηλώνουμε επίσης την πόρτα στην οποία θα «ακούει» το hub και στην οποία θα μπορούν να συνδεθούν καθώς και το ρόλο που θα έχει ο Selenium Server, δηλαδή το hub.

Το δεύτερο playbook του Ενορχηστρωτή αφορά τη μεταφορά των πακέτων .pcap από την καταγραφή στο Raspberry Pi 3 κατά τη διάρκεια εκτέλεσης μιας δοκιμής και την κατάλληλη επεξεργασία τους ώστε να μπορούν εισαχθούν στη βάση δεδομένων στο Elasticsearch, αλλά και την εξαγωγή συμπερασμάτων μέσω στατιστικών στο Kibana. Και εδώ οι εργασίες επιτελούνται με τη βοήθεια του shell module. Συγκεκριμένα οι δύο εργασίες είναι:

- Η μεταφορά του αρχείου .pcap από το Raspberry Pi 3 στον Ενορχηστρωτή μέσω SFTP. Αρχικά γίνεται η σύνδεση στο Raspberry Pi 3 μέσω SSH αυτόματα λόγω του κλειδιού που έχουμε ανταλλάξει για την εκτέλεση των playbooks. Στη συνέχεια γίνεται η μεταφορά του αρχείου από τη διεύθυνση που βρίσκεται στο Raspberry Pi 3, στην επιθυμητή διεύθυνση στον Ενορχηστρωτή που παρακολουθείται από το Logstash.

- Η μετατροπή του αρχείου .pcap σε .json με τη βοήθεια του *tshark* (μια έκδοση του Wireshark προσανατολισμένη για λειτουργία σε τερματικό). Εδώ προσθέτουμε στο όνομα του αρχείου .json που δημιουργήθηκε από την προηγούμενη εντολή και την ώρα και ημερομηνία της εκτέλεσης από το σύστημα.

Το τρίτο αφορά τον τερματισμό των διαδικασιών που εκκινήθηκαν κατά τη διάρκεια εκτέλεσης της δοκιμής και πρέπει να τερματιστούν εφόσον περατώθηκε η εκτέλεση της εκάστοτε δοκιμής. Συγκεκριμένα, τερματίζεται η διαδικασία του standalone Selenium Server που λειτουργεί ως hub στον Ενορχηστρωτή για να αποφύγουμε την άσκοπη σπατάλη πόρων.

Το τέταρτο playbook αφορά την εκτέλεση της εντολής iperf3 στην πλευρά του Ενορχηστρωτή. Για την εκτέλεσή της απαιτείται η διεύθυνση IP του μηχανήματος που δημιουργεί ο ESXi Server. Οι εργασίες που εκτελεί είναι η απόκτηση της διαύθυνσης IP του μηχανήματος στο οποίο θα εκτελεστεί η εντολή, καθώς και εκτέλεση αυτής της εντολής με χρήση του module shell και η ανακατεύθυνση της εξόδου στην αναφορά που θα δημιουργηθεί για το χρήστη.

Το πέμπτο playbook αφορά στη δοκιμή φόρτωσης του Reference Page του εικονικού μηχανήματος που δηξμιουργούμε μέσω του ESXi Server. Ο λόγος που εκτελείται σε playbook είναι για να αποκτήσουμε τη διεύθυνση IP την οποία δε γνωρίζουμε πριν την εκτέλεση της δοκιμής. Και εδώ οι εργασίες εκτελούνται μέσω του shell module. Οι εργασίες είναι η απόκτηση της διεύθυνσης IP της νέας εικονικής μηχανής καθώς και η εκτέλεση της δοκιμής σελίδας του Reference Page της μηχανής στο Raspberry Pi.

4.4.2 Playbooks για το Raspberry Pi 3

Για το Raspberry Pi 3 δημιουργήσαμε τέσσερα playbooks, τα ***selenium-grid-node.yml***, ***capture.yml***. ***selenium-node-close.yml*** και ***capture-close.yml***.

Το πρώτο αφορά την εκκίνηση του standalone Selenium Server στην πλευρά του Raspberry Pi 3 αυτή τη φορά. Και εδώ χρησιμοποιήσαμε το shell module στο playbook. Η εκτέλεση γίνεται πάλι με τη βοήθεια της Java μέσω του .jar αρχείου. Το Raspberry Pi 3, όπως έχουμε εξηγήσει και σε προηγούμενο κεφάλαιο, θα είναι ο αποδέκτης των αιτημάτων του χρήστη μέσω του Ενορχηστρωτή. Επομένως, ο ρόλος που θα επιτελεί θα είναι αυτός ενός κόμβου (node) στο hub. Επίσης, εξαιτίας της απουσίας οθόνης για το Raspberry Pi, απαιτείται η χρήση ενός επιπλέον προγράμματος που εκτελεί όλες τις γραφικές λειτουργίες σε εικονική μνήμη χωρίς να εμφανίζει καμία έξοδο οθόνης. Αυτό είναι το *Xvfb* (X virtual framebuffer) το οποίο είναι ένας διακομιστής προβολής που υλοποιεί το πρωτόκολλο διακομιστή προβολής X11. Επιπλέον για την εκτέλεση της εντολής, απαιτούνται η πόρτα στην οποία θα ακούει ο Selenium Server, ο ρόλος του που είπαμε ότι θα είναι node, η διεύθυνση του hub στο οποίο θα συνδεθεί και από το οποίο θα δέχεται τα αιτήματα για περιήγηση καθώς και διάφορες λεπτομέρειες σχετικά με τον browser τον οποίο θα τρέξει το Raspberry Pi μέσω του Xvfb.

Το δεύτερο playbook που δημιουργήσαμε για το Raspberry Pi αφορά την καταγραφή πακέτων κατά τη διάρκεια εκτέλεσης μιας δοκιμής περιήγησης στο διαδίκτυο από το χρήστη μέσω του Ενορχηστρωτή. Συγκεκριμένα, η καταγραφή γίνεται με τη βοήθεια του *tcpdump*, ένας κοινός αναλυτής πακέτων που τρέχει από τη γραμμή εντολών. Και εδώ το module που χρησιμοποιήσαμε στο playbook είναι το shell. Στην εντολή της καταγραφής δηλώνουμε επιπλέον τη διεπαφή του Raspberry Pi στην οποία θέλουμε να εκτελεστεί καθώς και το όνομα του αρχείου στο οποίο θα καταγράφεται η έξοδος μέχρις ότου τελειώσει η διαδικασία της δοκιμής.

Το τρίτο playbook που δημιουργήσαμε στο Raspberry Pi αφορά τον τερματισμό των διεργασιών του standalone Selenium Server και της εικονικής οθόνης στο. Εκτελείται στο τέλος της εκάστοτε δοκιμής τερματίζοντας τις απαραίτητες αυτές διεργασίες για να μην σπαταλάμε άσκοπα πόρους στο Raspberry Pi.

Το τέταρτο και τελευταίο playbook που δημιουργήσαμε για το Raspberry Pi αφορά τον τερματισμό της καταγραφής πακέτων με τη χρήση του *tcpdump*. Προτιμήσαμε να δημιουργήσουμε ξεχωριστό playbook για αυτή τη λειτουργία καθώς έτσι εξυπηρετείται καλύτερα η δυνατότητα εκτέλεσης περισσότερων της μιας δοκιμής από τον ίδιο κόμβο. Η εκτέλεσή του γίνεται μετά το πέρας της εκάστοτε δοκιμής.

4.4.3 Playbooks για τον ESXi Server

Για τον ESXi Server δημιουργήσαμε τρία playbooks τα οποία αφορούν την εγγραφή και εκκίνηση μιας εικονική μηχανής, την εκκίνηση του *iperf* server για την υποδοχή κίνησης δεδομένων καθώς και τον τερματισμό και την απεγγραφή της μηχανής αυτής. Συγκεκριμένα, τα playbooks είναι τα ***start-server.yml***, ***start-iperf.yml*** και ***stop-server.yml***. Εξαιτίας της αδυναμίας εγκατάστασης του Ansible στον ESXi, τα συγκεκριμένα playbooks εκτελούνται στον Ενορχηστρωτή αλλά συνδέονται στον ESXi μέσω SSH για να εκτελέσουν τα scripts με τις απαιτούμενες ενέργειες ενώ το playbook

που αφορά την εκκίνηση του iPerf3 Server εκτελεί σύνδεση μέσω SSH στην νέα εικονική μηαχνή για εκτέλεση του κατάλληλου script.

Στο πρώτο, όπως δηλώνει και το όνομά του έχουμε την εγγραφή της εικονικής μηχανής που πρόκειται να χρησιμοποιήσουμε και η εκκίνησή της. Το script που εκτελείται μέσω SSH στον ESXi είναι το ***start-server.sh*** στο οποίο υπάρχουν οι αντίστοιχες εντολές που μας επιτρέπουν την καταχώρηση μιας εικονικής μηχανής στον κατάλογο του ESXi από τη γραμμή εντολών καθώς και η εκκίνηση της σε script με τη χρήση του προγράμματος *Expect*. Για την εγγραφή απαιτείται ο προσδιορισμός ενός ID με το οποίο θα γίνεται η αναφορά σε αυτή τη μηχανή στις υπόλοιπες εντολές παρακάτω. Το ID αυτό είναι το ID που έχει ο χρήστης από το λογαριασμό του στο site μέσω του Django framework.

Στο δεύτερο γίνεται η εκκίνηση του *iperf3* server για την υποδοχή της κίνησης δεδομένων. Εδώ το script είναι το ***start-iperf.sh*** και εκτελείται στην εικονική μας μηχανή. Συνδέεται όμως μέσω SSH στην νέα εικονική μηχανή με τη χρήση του προγράμματος *Expect.* Ο iperf3 Server τρέχει στην εικονική μηχανή μέχρις ότου τον τερματίσουμε εμείς μετά το πέρας των δοκιμών μας.

Το τρίτο playbook που δημιουργήσαμε για τον ESXi Server εξυπηρετεί τον ασφαλή τερματισμό της εικονικής μηχανής που δημιουργήσαμε για την δοκιμή της κίνησης δεδομένων μέσω του *iperf*. Συγκεκριμένα στο playbook εκτελούνται δύο εργασίες Αυτές είναι:

- Ο τερματισμός του *iperf3* server. Το script που εκτελείται εδώ είναι το ***stop-iperf.sh*** το οποίο εκτελείται στην εικονική μας μηχανή που τρέχουμε το playbook και συνδέεται μέσω SSH στην νέα εικονική μηχανή που τρέχει ο *iperf3* server.

- Ο τερματισμός της εικονικής μηχανής και η απεγγραφή της. Εδώ το script που εκτελείται μέσω SSH στον ESXi είναι το ***stop-server.sh*** το οποίο μας επιτρέπει τον τερματισμό της εικονικής μηχανής από μία γραμμή εντολών και την απεγγραφή της από τον κατάλογο του ESXi Server. Το μόνο απαιτούμενο από την εντολή είναι ο προσδιορισμός της μηχανής μέσω του ID που αναφέραμε και προηγουμένως.

4.5 Αρχιτεκτονική Μηχανημάτων

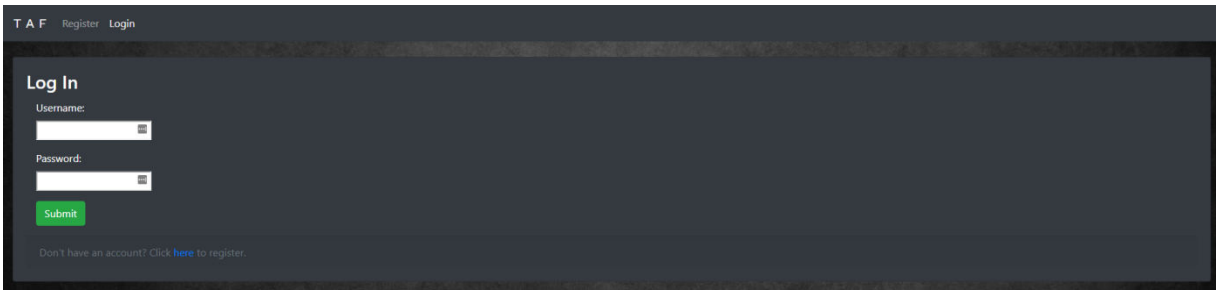
Στη συγκεκριμένη ενότητα θα αναφερθούμε στο λογισμικό που εγκαταστήσαμε στα μηχανήματα και ήταν απαραίτητο για την ανάπτυξη της εφαρμογής. Αρχικά, για το jenkins.cn.ece.ntua.gr, τον Ενορχηστρωτή μας, το λειτουργικό σύστημα που χρησιμοποιήσαμε είναι το Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-141-generic αρχιτεκτονικής 64-bit. Σε αυτόν εγκαταστήσαμε τα πακέτα των γλωσσών προγραμματισμού Java και Python. Πιο συγκεκριμένα, το πακέτο και η έκδοση της Java που εγκαταστήσαμε είναι το OpenJDK Runtime Environment (build 1.8.0_191-8u191-b12-0ubuntu0.16.04.1-b12), OpenJDK 64-Bit Server VM (build 25.191-b12, mixed mode) με έκδοση "1.8.0_191" ενώ η έκδοση της Python είναι η 3.5.2. Η έκδοση του Django στην οποία στήσαμε το site μας είναι η 2.1.3. Άλλα πακέτα Python τα οποία χρησιμοποιήσαμε είναι το HtmlTestRunner με

έκδοση 1.1.2 για τη δημιουργία μιας αναφοράς σε HTML από το αποτέλεσμα της εκτέλεσης των δοκιμών στο Raspberry Pi, το BeautifulSoup4 έκδοση 4.6.3 με για το διάβασμα των αποτελεσμάτων από το HTML αρχείο και το Selenium με έκδοση 3.141.0 για την εκτέλεση του κώδικα των δοκιμών στο Raspberry Pi. Όλα τα παραπάνω πακέτα είναι πακέτα αποκλειστικά για τη γλώσσα Python και τα εγκαταστήσαμε μέσω του pip, ένας διαχειριστής πακέτων για πακέτα Python. Για το Selenium εγκαταστήσαμε επίσης και το αρχείο .jar του standalone Selenium Server με έκδοση επίσης 3.141.0 για τη λειτουργία του Selenium Grid και την εκτέλεση των δοκιμών αυτοματοποιημένης περιήγησης στο διαδίκτυο μέσω του Raspberry Pi. Επίσης, εγκαταστήσαμε το Ansible με έκδοση 2.7.2 για την εκτέλεση των playbooks με σκοπό την αυτοματοποίηση της διαδικασίας δοκιμών, το TShark με έκδοση 2.6.6 για την ανάγνωση και μετατροπή των αρχείων καταγραφής .pcap από το Raspberry Pi στον Ενορχηστρωτή και το πακέτο για την εντολή iperf με έκδοση 2.0.5 (2 June 2018) για τη μεταφορά κίνησης δεδομένων στις δοκιμές κίνησης από τον Ενορχηστρωτή στον ESXi Server. Τέλος, εγκαταστήσαμε και τα τρία κομμάτια της ELK Stack, το Elasticsearch με έκδοση 6.5.0, το Logstash με έκδοση 6.5.4 και το Kibana με έκδοση 6.5.0 για τη ευρετηρίαση των αρχείων και εξαγωγή συμπερασμάτων με τη βοήθεια στατιστικών με οπτική απεικόνιση.

Για το Raspberry Pi 3, το λειτουργικό σύστημα που χρησιμοποιήσαμε είναι το Raspbian GNU/Linux 8 (Jessie). Στο Raspberry Pi εγκαταστήσαμε πάλι τα πακέτα των γλωσσών προγραμματισμού Java και Python, συγκεκριμένα για τη Java το Java(TM) SE Runtime Environment (build 1.8.0_191-b12), Java HotSpot(TM) Client VM (build 25.191-b12, mixed mode) με έκδοση "1.8.0_191", την ίδια με τον Ενορχηστρωτή ώστε να μην υπάρχουν ασυμβατότητες σχετικά με τη λειτουργία του Selenium Server, ενώ για την Python την έκδοση 3.4.2. Επίσης, εγκαταστήσαμε κι εδώ το Ansible με έκδοση 2.7.2 όπως και τον standalone Selenium Server με έκδοση 3.141.0 για τη λειτουργία του Selenium Grid. Για την αυτοματοποίηση της διαδικασίας περιήγησης, εγκαταστήσαμε επίσης τον Mozilla Firefox για περιηγητή για τις δοκιμές μας με έκδοση 52.9.0, τον geckodriver με έκδοση 0.17.0 για WebDriver του Firefox στο Selenium και το Xfvb με έκδοση 11 για την εικονική περιήγηση εξαιτίας της έλλειψης οπτικού μέσου στο Raspberry Pi.

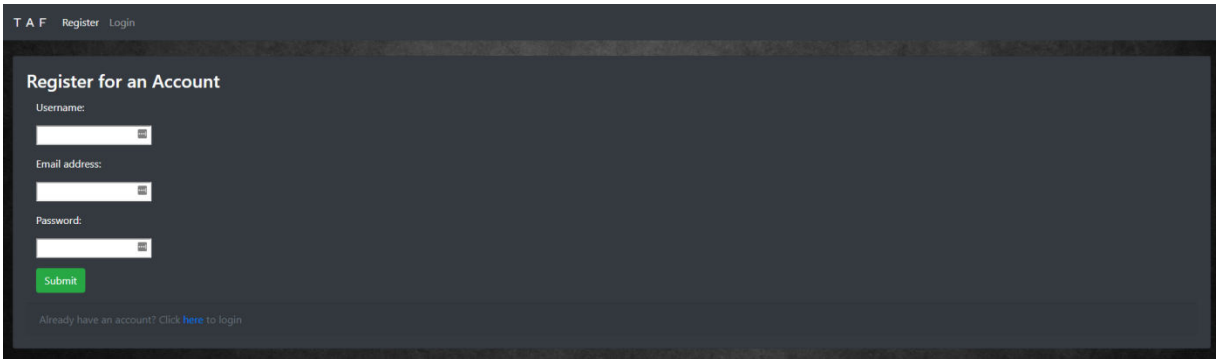
4.6 Έλεγχος Ορθής Λειτουργίας

Σε αυτή την ενότητα θα ελέξουμε τον τρόπο λειτουργίας της εφαρμογής που δημιουργήσαμε. Εφόσον έχουμε ξεκινήσει τον Developing Server του Django, μπορούμε να μεταβούμε στη σελίδα της εφαρμογής (jenkins.cn.ece.ntua.gr:8000/test-automation) μέσω ενός φυλλομετρητή. Αρχικά, η οθόνη που βλέπουμε είναι η πιστοποίηση του χρήστη που προσπαθεί να εισέλθει στην εφαρμογή. όπως φαίνεται και παρακάτω:



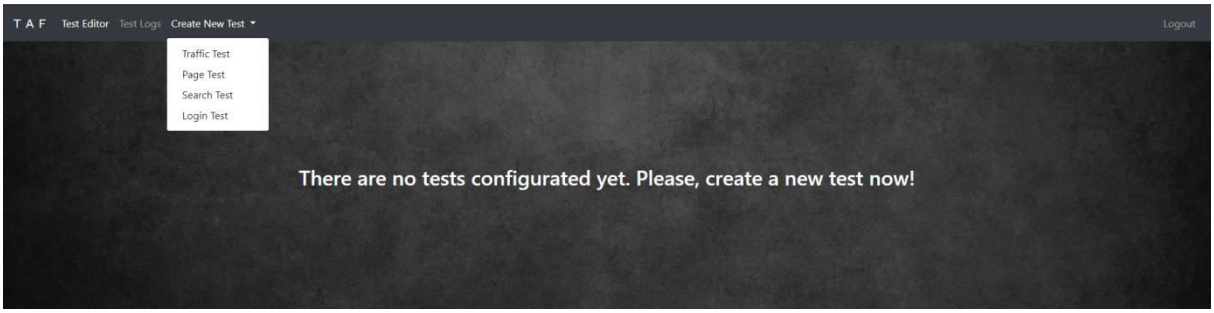
Εικόνα 4-2: Login

Αν δεν έχουμε δημιουργήσει ήδη λογαριασμό, τότε πρέπει να μεταφερθούμε στην αντίστοιχη σελίδα για δημιουργία ενός, όπως φαίνεται παρακάτω:



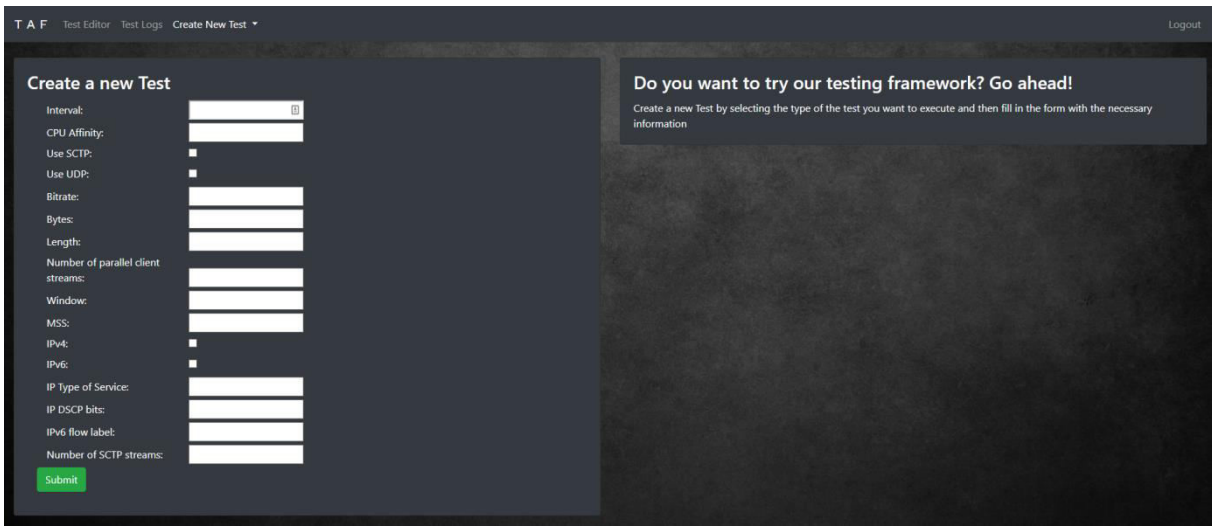
Εικόνα 4-3: Register

Εφόσον η ταυτοποίηση των στοιχείων μας γίνει με επιτυχία, μεταφερόμαστε στην αρχική σελίδα της εφαρμογής που είναι η σελίδα όπου παρουσιάζονται όλα τα tests που έχουμε δημιουργήσει. Οπότε μπορούμε να επιλέξουμε την αντίστοιχη επιλογή για δημιουργία νέων δοκιμών:

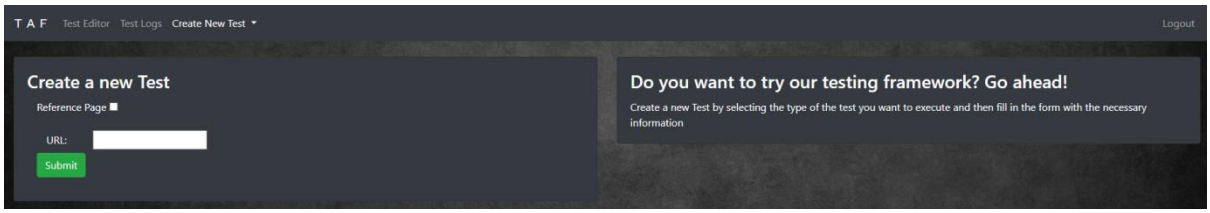


Εικόνα 4-4: Αρχική σελίδα Test Editor - Empty

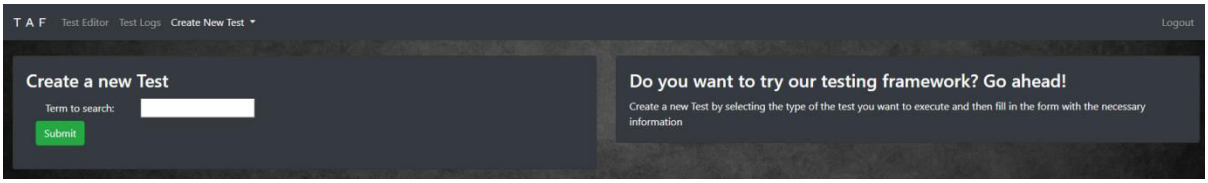
Δημιουργούμε ένα test για κάθε διαθέσιμο τύπο, από τις επιλογές που βλέπουμε στην προηγούμενη εικόνα. Στη συνέχεια παρουσιάζουμε τις φόρμες που εμφανίζονται όταν επιλέγεται καθεμιά από τις παραπάνω επιλογές.



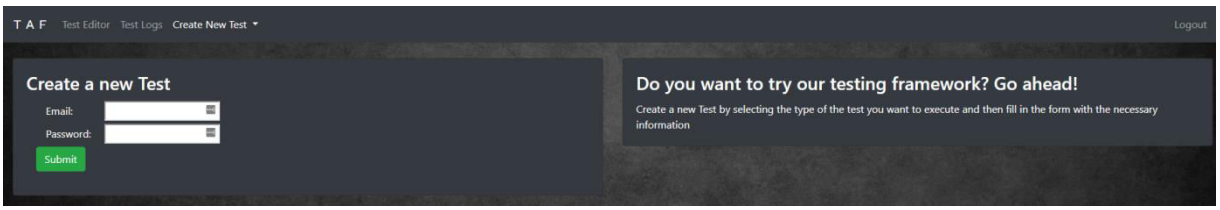
Εικόνα 4-5: Δημιουργία νέου Traffic Test



Εικόνα 4-6: Δημιουργία νέου Page Test

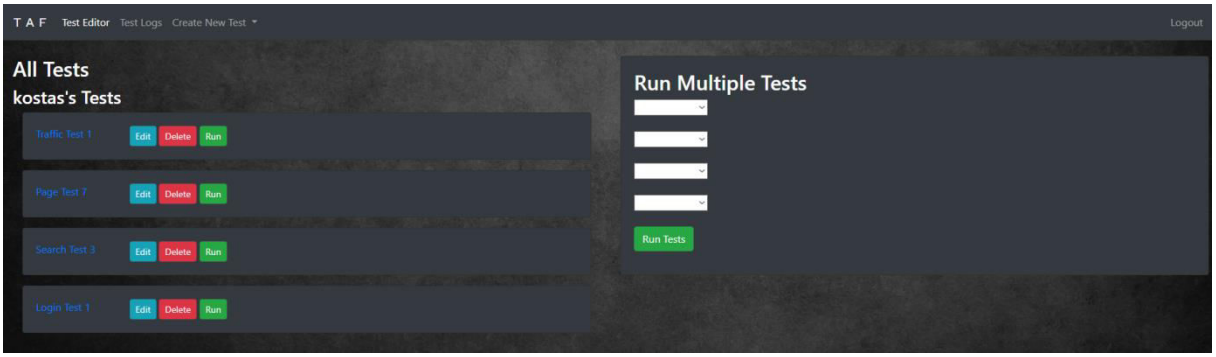


Εικόνα 4-7: Δημιουργία νέου Search Test



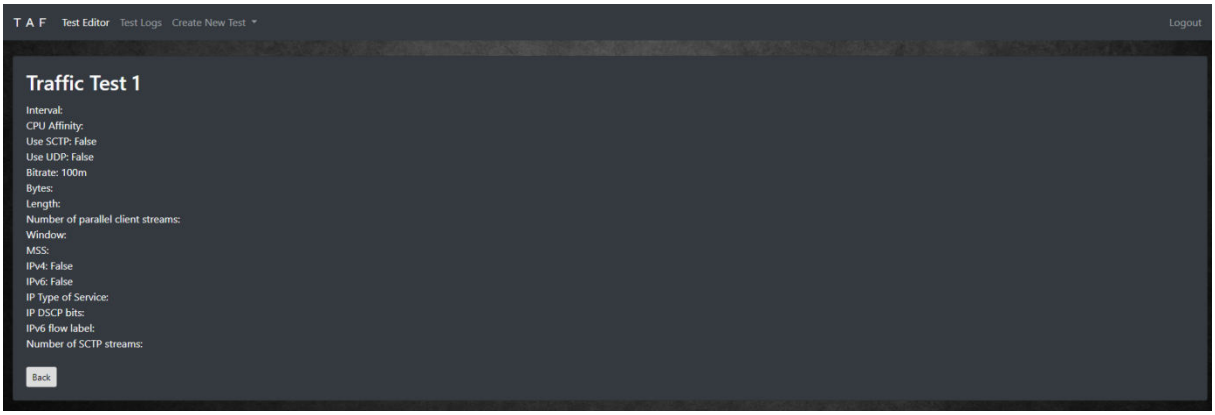
Εικόνα 4-8: Δημιουργία νέου Login Test

Η σελίδα των δοκιμών μας τώρα έχει πάρει την παρακάτω μορφή. Σε αυτή φαίνονται οι διαθέσιμες επιλογές τις οποίες μπορούμε να εφαρμόσουμε πάνω στις δοκιμές, όπως Επεξεργασία, Διαγραφή και Εκκίνηση Δοκιμής. Στη δεξιά στήλη υπάρχει και η δυνατότητα για εκκίνηση πολλαπλών δοκιμών που αφορούν την αυτοματοποιημένη περιήγηση στο Raspberry Pi 3 μέσω του Selenium:

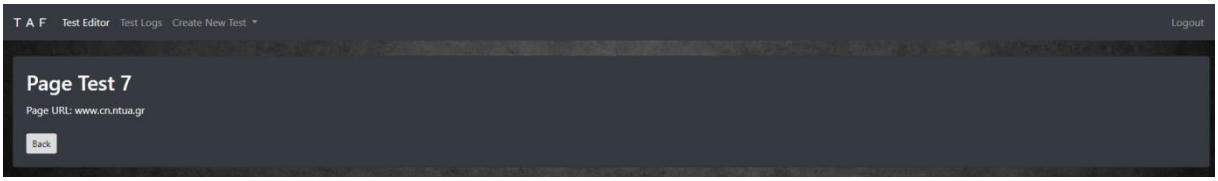


Εικόνα 4-9: Αρχική σελίδα Test Editor - Tests

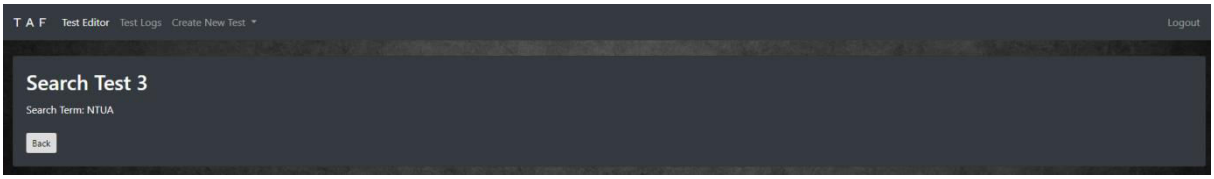
Επίσης, μπορούμε να δούμε τις λεπτομέρειες για κάθε test που δημιουργήσαμε, επιλέγοντας το σύνδεσμο πάνω στο όνομα κάθε test. Τα tests που δημιουργήσαμε έχουν τις παραμέτρους που φαίνονται παρακάτω:



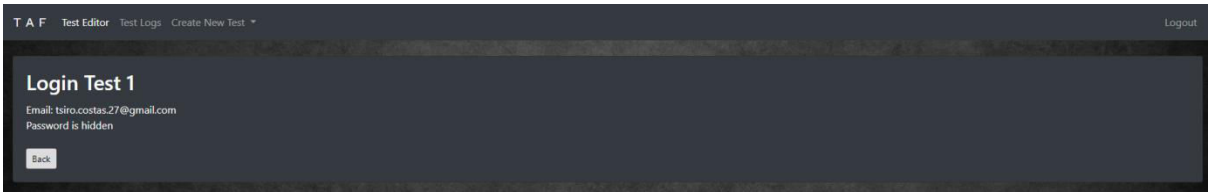
Εικόνα 4-10: Λεπτομέρειες Traffic Test



Εικόνα 4-11: Λεπτομέρειες Page Test



Εικόνα 4-12: Λεπτομέρειες Search Test



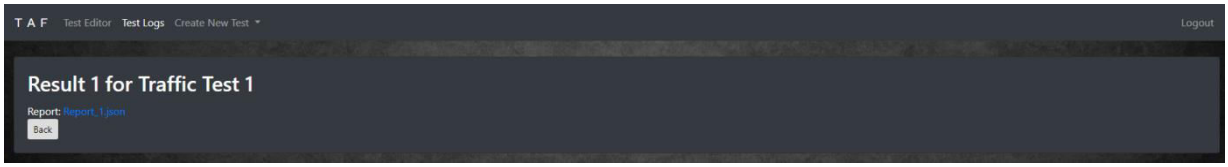
Εικόνα 4-13: Λεπτομέρειες Login Test

Στη συνέχεια, εφόσον εκτελέσουμε τις δοκιμές, είτε μεμονωμένα είτε μαζικά αν πρόκειται για δοκιμές στο Raspberry Pi, μπορούμε να δούμε τα αποτελέσματα από την εκτέλεσή τους στην καρτέλα

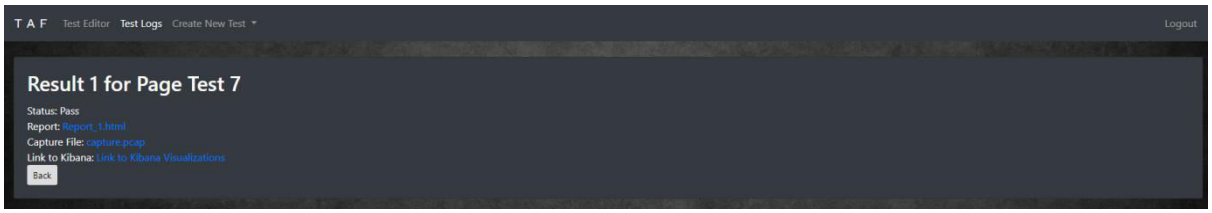
που παρουσιάζονται τα αποτελέσματα. Και εδώ βλέπουμε τις διαθέσιμες ενέργειες που μπορούμε να εκτελέσουμε στα tests που στην προκειμένη περίπτωση είναι μόνο η Διαγραφή:

Εικόνα 4-14: Σελίδα αποτελεσμάτων Test Logs

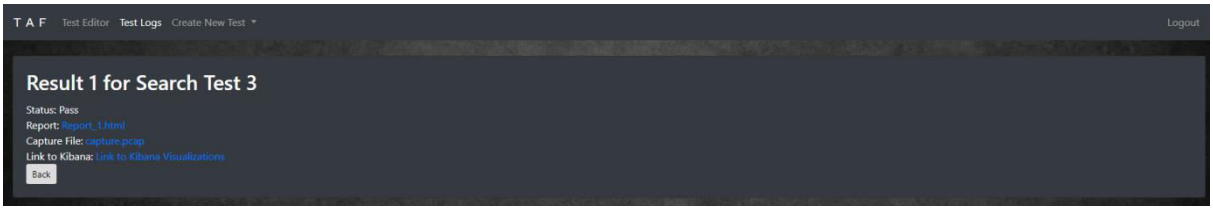
Επίσης, πατώντας στο σύνδεσμο κάθε εγγραφής αποτελεσμάτων από τις δοκιμές που εκτελέσαμε, μπορούμε να δούμε τις αναφορές των αποτελεσμάτων είτε σε μορφή .json για τις δοκιμές κίνησης δεδομένων με χρήση της iPerf3, είτε μέσω μιας αναφοράς σε HTML μαζί με την καταγραφή των πακέτων σε μορφή .pcap και την οπτικοποίηση τους στο Kibana μέσω του ELK Stack για τις δοκιμές περιήγησης στο Raspberry Pi:



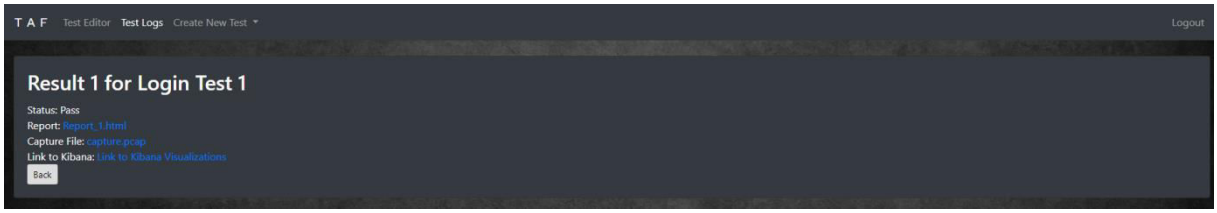
Εικόνα 4-15: Λεπτομέρειες αποτελεσμάτων Traffic Test



Εικόνα 4-16: Λεπτομέρειες αποτελεσμάτων Page Test



Εικόνα 4-17: Λεπτομέρειες αποτελεσμάτων Search Test



Εικόνα 4-18: Λεπτομέρειες αποτελεσμάτων Login Test

Οι εκτελέσεις των δοκιμών αναλύονται περισσότερο στο επόμενο κεφάλαιο της αξιολόγησης. Σαν πρώτο έλεγχο ορθής λειτουργίας παρατηρούμε ότι η εφαρμογή μας έχει την επιθυμητή λειτουργία.

5

Αξιολόγηση

Στο παρόν κεφάλαιο θα γίνει αξιολόγηση της εφαρμογής που αναπτύξαμε κατά της διάρκεια εκπόνησης της διπλωματικής και που περιγράφηκε στα προηγούμενα κεφάλαια. Αρχικά θα αναλυθούν οι παράμετροι αξιολόγησής της και η σημασία τους για την εφαρμογή. Θα παρουσιαστεί η πειραματική διαδικασία που ακολουθήθηκε καθώς και τα αποτελέσματα των πειραμάτων και τα συμπεράσματα που εξάχθηκαν από αυτά.

5.1 Παράμετροι αξιολόγησης

Οι παράμετροι αξιολόγησης της εφαρμογής που αναπτύξαμε αφορούν κυρίως χρονικά μεγέθη για την απόδοσή της. Παράλληλα, η ορθότητα λειτουργίας της εφαρμογής είναι ένας σημαντικός παράγοντας για την αξιολόγηση της διάταξής μας.

Πιο συγκεκριμένα, οι παράμετροι που επιλέχθηκαν αφορούν τις δοκιμές για την αυτοματοποίηση της διαδικασίας περιήγησης. Για τις δοκιμές περιήγησης, σημαντικές παράμετροι είναι ο συνολικός χρόνος εκτέλεσης μιας δοκιμής αλλά και ο χρόνος περιήγησης. Προφανώς, ο συνολικός χρόνος της δοκιμής θα εξαρτάται από τον χρόνο περιήγησης αλλά και από το χρόνο δημιουργίας των κατάλληλων συνθηκών για την εκτέλεση της δοκιμής, χρόνος όμως που είναι ίδιος για όλες τις δοκιμές. Σε αυτόν το χρόνο υπολογίζεται ο χρόνος δημιουργίας του Selenium Hub με την εκτέλεση του Selenium standalone Server και ο χρόνος σύνδεσης του Raspberry Pi ως κόμβου πάλι με την εκτέλεση του Selenium standalone Server. Ο χρόνος περιήγησης εξαρτάται και από τον τύπο της

δοκιμής και τις ενέργειες που έχει να εκτελέσει και από το πόσο «βαριά» είναι η σελίδα που προσπαθούμε να φορτώσουμε.

Στη σημείο αυτό θα πρέπει να αναφερθούμε και στα τεχνικά χαρακτηριστικά που έχουν τα εικονικά και φυσικά μηχανήματα που χρησιμοποιήσαμε. Ο Ενορχηστρωτής διαθέτει 2 επεξεργαστές (CPUs) χρονισμένους στα 2.4GHz, μνήμη RAM 3951MB (περίπου 4GB) και φυσική μνήμη χωρητικότητας 80GB. Το Raspberry Pi διαθέτει 4 επεξεργαστές χρονισμένους στα 1.2 GHz, μνήμη RAM 1GB και φυσική μνήμη χωρητικότητας 15GB.

5.2 Οργάνωση Πειραμάτων και Αποτελέσματα

5.2.1 Χρόνος δημιουργίας του Selenium Hub

Ο υπολογισμός του χρόνου δημιουργίας του Selenium Hub αναφέρεται στο χρονικό διάστημα εκτέλεσης της εντολής που είναι αναγραμμένη στο playbook ***selenium-grid-hub.yml*** από την αρχή εκτέλεσής της μέχρι τη στιγμή που θα είναι έτοιμο να δεχθεί αιτήματα σύνδεσης από άλλους κόμβους. Το Selenium μας εμφανίζει σε κάθε εργασία που πραγματοποιεί, τον ακριβή πραγματικό χρόνο που λαμβάνει χώρα το κάθε γεγονός, οπότε είναι πολύ εύκολο να υπολογίσουμε το διάστημα αυτό. Μετά από δέκα (10) εκτελέσεις της εντολής αυτής πήραμε τα παρακάτω αποτελέσματα.

Αριθμός Εκτέλεσης	***Διάρκεια (sec)***
1	*1.282*
2	*1.303*
3	*1.259*
4	*1.294*
5	*1.025*
6	*1.326*
7	*1.255*
8	*1.302*
9	*1.332*
10	*1.260*
Μέσος όρος	***1.264***

Πίνακας 1: Χρόνος δημιουργίας του Selenium Hub

Βλέποντας τις μετρήσεις που καταγράψαμε, παρατηρούμε ότι ο χρόνος δημιουργίας του Selenium Hub στην πλευρά του Ενορχηστρωτή είναι πάρα πολύ μικρός και μάλιστα λίγο πάνω από το ένα δευτερόλεπτο. Οι συγκεκριμένοι χρόνοι είναι αναμενόμενοι εξαιτίας των δυνατοτήτων του εικονικού μας μηχανήματος αλλά και της ζητούμενης εργασίας. Κατά την εκτέλεση των δοκιμών από την εφαρμογή ίσως προκύψει μια επιπλέον καθυστέρηση της τάξεως του ενός δευτερολέπτου λόγω

της εκτέλεσής της μέσω του Ansible playbook που αναφέραμε πιο πριν και αναλύσαμε σε προηγούμενο κεφάλαιο.

5.2.2 Χρόνος σύνδεσης κόμβου στο Selenium Hub

Για το χρόνο σύνδεσης ενός κόμβου στο Selenium Hub, θα αφήσουμε ανοιχτό το Hub στην πλευρά του Ενορχηστρωτή και θα εκτελέσουμε την εντολή που είναι αναγραμμένη στο playbook ***selenium-grid-node.yml*** στην πλευρά του Raspberry Pi, που αποτελεί και τον κόμβο της διάταξής μας. Η διαδικασία που ακολουθούμε είναι παρόμοια με αυτή την προηγούμενης ενότητας και έχει το ίδιο πλεονέκτημα όσον αφορά το χρόνο καθώς και εδώ πρόκειται για μια εντολή που αφορά το Selenium. Μετά από δέκα (10) εκτελέσεις πάλι της εντολής αυτής πήραμε τα παρακάτω αποτελέσματα.

Αριθμός Εκτέλεσης	***Διάρκεια (sec)***
1	*3.747*
2	*3.512*
3	*3.425*
4	*3.549*
5	*3.489*
6	*3.427*
7	*3.398*
8	*3.408*
9	*3.442*
10	*3.481*
Μέσος όρος	***3.488***

Πίνακας 2: Χρόνος σύνδεσης κόμβου στο Selenium Hub

Βλέποντας και εδώ τις μετρήσεις που καταγράψαμε από την επανάληψη της σύνδεσης του Raspberry Pi ως κόμβου στο Selenium Hub, παρατηρούμε ότι η διαδικασία αυτή είναι αρκετά πιο χρονοβόρα σε σχέση με την πρώτη. Αυτό είναι πάλι αναμενόμενο, καθώς η εκτέλεση γίνεται στο Raspberry Pi, ένα φυσικό μηχάνημα με πολύ μικρότερες δυνατότητες σε σχέση με την εικονική μηχανή του Ενορχηστρωτή. Επίσης, η εργασία της σύνδεσης έχει ως επιπλέον καθήκον τον έλεγχο των παραμέτρων περιβάλλοντος που ορίζονται στην εντολή με αυτές που επικρατούν στο περιβάλλον του συστήματος. Και σε αυτή την περίπτωση, υπάρχει ενδεχόμενο επιπλέον καθυστέρησης λόγω της εκτέλεσης της εντολής σε playbook και μάλιστα σε απομακρυσμένη συσκευή που απαιτεί αρχικά τη σύνδεση μέσω SSH για την εκτέλεση των εντολών.

5.2.3 Χρόνος εκτέλεσης δοκιμών περιήγησης

Η ανάπτυξη της εφαρμογής μας για την αξιολόγηση της τεχνολογίας NFV περιλαμβάνει κάποιους καθορισμένους τύπους δοκιμών. Τους τύπους αυτούς καθώς και τη διαδικασία εκτέλεσης που ακολουθείται σε κάθε δοκιμή τους αναλύσαμε στο προηγούμενο κεφάλαιο. Στη συγκεκριμένη ενότητα θα εξετάσουμε το χρόνο που χρειάζεται για την εκτέλεση μιας τέτοιας δοκιμής. Γνωρίζουμε από πριν το χρόνο που χρειάζεται για να δημιουργηθεί το Selenium Hub και τον αντίστοιχο για να συνδεθεί το Raspberry Pi σε αυτό, οπότε μπορούμε να προσθέσουμε την ανάλογη καθυστέρηση στις συναρτήσεις που έχουμε ορίσει στο αρχείο ***views.py*** ώστε να μπορέσουν να εκτελεστούν οι εντολές μας ασύγχρονα και να μην υπάρξουν προβλήματα. Οι συναρτήσεις που αναφέραμε περιλαμβάνουν όλη τη διαδικασία, από τη δημιουργία του Hub και τη σύνδεση του Raspberry Pi, μέχρι τον τερματισμό των διεργασιών αυτών και τη δημιουργία των αποτελεσμάτων.

Στην εφαρμογή μας λοιπόν έχουμε δημιουργήσει τρία tests, ένα για κάθε τύπο, όσον αφορά την αυτοματοποίηση της διαδικασίας περιήγησης. Πιο συγκεκριμένα, έχουμε επιλέξει τη σελίδα του Εργαστηρίου Δικτύων Υπολογιστών για φόρτωση σελίδας, τον όρο "NTUA" για αναζήτηση στο Google, καθώς και κάποια στοιχεία για τη σύνδεση στο Facebook. Για λόγους ασφαλείας δε χρησιμοποιήθηκαν στοιχεία που ανήκουν σε κάποιον ενεργό λογαριασμό.

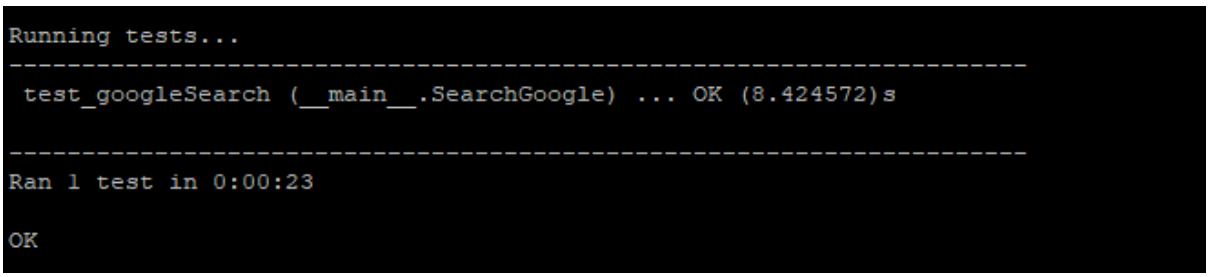
Στη συνέχεια παρουσιάζουμε τα αποτελέσματα από την εκτέλεση των παραπάνω δοκιμών που δημιουργήσαμε. Αρχικά, έχουμε την εκτέλεση της δοκιμής με τη φόρτωση της σελίδας του Εργαστηρίου Δικτύων Υπολογιστών.

```
Running tests...
----------------------------------------------------------------------
 test_pageHTTP (__main__.PageHTTP) ... OK (6.902384)s

----------------------------------------------------------------------
Ran 1 test in 0:00:20

OK
```

Εικόνα 5-1: Εκτέλεση Page Test

Παρατηρούμε ότι ο συνολικός χρόνος που απαιτείται για την εκτέλεση της δοκιμής είναι 20 δευτερόλεπτα. Μέσα σε αυτά υπολογίζεται ο συνολικός χρόνος που απαιτείται για την εκκίνηση του Firefox στο Raspberry Pi, που χρησιμοποιείται ως browser, η εικονική πληκτρολόγηση της διεύθυνσης και η φόρτωση της σελίδας, καθώς και το κλείσιμο του παραθύρου στο τέλος της εκτέλεσης της δοκιμής. Από αυτά, τα 7 περίπου σχετίζονται αποκλειστικά με την περιήγηση, ενώ τα 5 είναι η καθυστέρηση την οποία έχουμε προσθέσει για οπτικούς λόγους κυρίως.

Στη συνέχεια, παρουσιάζουμε τα αποτελέσματα από τη δεύτερη δοκιμή που δημιουργήσαμε και αφορά στην αναζήτηση του όρου «NTUA» στη μηχανή αναζήτησης της Google.



Εικόνα 5-2: Εκτέλεση Search Test

Παρατηρούμε ότι και εδώ η εκτέλεση της δοκιμής είναι πολύ γρήγορη, με συνολικό χρόνο εκτέλεσης στα 23 δευτερόλεπτα. Από αυτά, περίπου τα 9 είναι αποκλειστικά για την περιήγηση. Ουσιαστικά, διαφέρει από την προηγούμενη εκτέλεση, καθώς πέρα από τη φόρτωση της σελίδας της Google, πραγματοποιείται και πληκτρολόγηση του όρου προς αναζήτηση. Επίσης, σημαντικό ρόλο παίζει και το μέγεθος της σελίδας, μιας και η σελίδα τους Εργαστηρίου Δικτύων Υπολογιστών είναι αρκετά πιο ελαφριά από αυτή των αποτελεσμάτων της Google. Και εδώ έχουμε καθυστέρηση 5 δευτερολέπτων για οπτικούς λόγος.

Τέλος, παρουσιάζουμε τα αποτελέσματα από την εκτέλεση της τρίτης δοκιμής που αφορά τη σύνδεση στο Facebook.

Εικόνα 5-3: Εκτέλεση Login Test

Εδώ βλέπουμε μια μικρή διαφορά στο χρόνο σε σχέση με τις δύο προηγούμενες δοκιμές. Αυτό οφείλεται αφενός στις καθυστερήσεις που έχουμε προσθέσει ενδιάμεσα κατά τη διάρκεια πληκτρολόγησης των στοιχείων και αφετέρου στο ίδιο το μέγεθος της σελίδας του Facebook, που είναι μια πολύ βαριά σελίδα.

Και στις τρεις περιπτώσεις όμως παρατηρούμε ότι η διαδικασία είναι πολύ γρήγορη και η λειτουργία της εφαρμογής μπορεί να χαρακτηριστεί ως επιτυχής.

5.3 Συμπεράσματα αξιολόγησης

Με βάση τα αποτελέσματα που παρουσιάσαμε παραπάνω, ο κύριος σκοπός της διπλωματικής εργασίας που ήταν η δημιουργία μιας web εφαρμογής για την εκτέλεση αυτοματοποιημένων δοκιμών κρίνεται επιτυχής. Πιο συγκεκριμένα, η εφαρμογή που δημιουργήσαμε μας επιτρέπει να εκτελέσουμε δοκιμές με χρήση της τεχνολογίας NFV τόσο στο Raspberry Pi, που αφορούν αυτοματοποίηση της διαδικασίας περιήγησης, όσο και στον ESXi Server που αφορούν δοκιμές δημιουργίας κίνησης. Σε όλες τις περιπτώσεις παρατηρούμε ότι η εκτέλεση των δοκιμών γίνεται σε πολύ μικρό χρονικό διάστημα, μικρότερο του ενός λεπτού.

6

Επίλογος

6.1 Σύνοψη

Στόχος της παρούσας διπλωματικής ήταν η ανάπτυξη μιας web εφαρμογής για την εκτέλεση αυτοματοποιημένων δοκιμών χρησιμοποιώντας την τεχνολογία εικονικοποίησης δικτυακών λειτουργιών. Η εφαρμογή βασίζεται εξ ολοκλήρου σε εικονικές μηχανές για τη μετάδοση των απαραίτητων πληροφοριών και δεδομένων από και προς το χρήστη. Υπάρχει επίσης και η φυσική συσκευή του Raspberry Pi που αναπαριστά το τελευταίο άκρο της διάταξης που αναπτύξαμε και εκτελεί τις απαιτούμενες ενέργειες των δοκιμών στις οποίες συμμετέχει. Με τη χρήση αυτής της τεχνολογίας, επιτυγχάνεται η μείωση του κόστους λειτουργίας (OPEX) τόσο των εταιρειών, όσο και των οικιακών πελατών εξαιτίας της μειωμένης ενέργειας που καταναλώνει σε σχέση με την αντίστοιχη φυσική συσκευή που επιτελεί την ίδια εργασία. Για τις εταιρείες επίσης, μείωση εξόδων σημαίνει παράλληλα και αύξηση εσόδων, η οποία προέρχεται και από τη μείωση τόσο των επενδυτικών εξόδων (CAPEX) όσο και το χρόνο παράδοσης μιας υπηρεσίας στην αγορά. Τέλος, με τη χρήση επιτυγχάνονται αυξημένες ταχύτητες μετάδοσης της πληροφορίας στις παρεχόμενες υπηρεσίες, προσφέροντας έτσι και μια καλύτερη εμπειρία στο χρήστη.

Η εφαρμογή που αναπτύξαμε κατά τη διάρκεια της παρούσας διπλωματικής περιλαμβάνει μια ποικιλία από τύπους δοκιμών για την αξιολόγηση της τεχνολογίας NFV. Πιο συγκεκριμένα, οι δοκιμές αφορούν αυτοματοποίηση της διαδικασίας περιήγησης στο διαδίκτυο, εκτελώντας διάφορες ενέργειες, όπως και την απλή μετάδοση κίνησης δεδομένων. Αφού παρουσιάσαμε αναλυτικά τις

τεχνολογίες οι οποίες σχετίζονται με την εφαρμογή μας καθώς και τα εργαλεία ελεύθερου λογισμικού τα οποία επιλέχθηκαν για την ανάπτυξή της, στη συνέχεια παρουσιάσαμε αναλυτικά τη διαδικασία αυτή. Στο τέλος της εργασίας, αφού πραγματοποιήσαμε μια σειρά πειραμάτων, αξιολογήσαμε την απόδοση της εφαρμογής και καταλήξαμε στο συμπέρασμα ότι οι δοκιμές που εκτελέσαμε, πραγματοποιούνται σε αποδεκτά χρονικά διαστήματα. Επίσης, ο σκοπός της εργασίας που ήταν η δημιουργία μιας εφαρμογής για την αυτοματοποίηση των δοκιμών κρίνεται επιτυχής, καθώς μέσω της εφαρμογής μας είναι δυνατή η εκτέλεση τέτοιου είδους δοκιμών καθώς και η εξαγωγή συμπερασμάτων από τα αποτελέσματά τους.

6.2 Μελλοντικές επεκτάσεις

Η ανάπτυξη των VNF με αποδοτικό τρόπο στις εικονικοποιημένες υποδομές δημιουργεί σημαντικές προκλήσεις για να διασφαλιστεί ότι στον εργασιακό φόρτο παρέχονται οι απαραίτητοι υπολογιστικοί πόροι. Επιπλέον, είναι απαραίτητο να διασφαλιστεί ότι ο φόρτος εργασίας καταναλώνει τους πόρους αυτούς με αποτελεσματικό τρόπο, προκειμένου να ικανοποιηθούν οι βασικοί δείκτες επιδόσεων (Key Performance Indicators - KPI) που απαιτεί ένας διαχειριστής δικτύου. Τα τυποποιημένα εργαλεία που χρησιμοποιούνται από πολλούς φορείς, συμπεριλαμβανομένων των γεννητριών φορτίου που βασίζονται στο υλικό, μπορούν να ελέγξουν το επίπεδο απόδοσης ενός VNF. Ωστόσο, δεν παρέχουν τις απαραίτητες γνώσεις σχετικά με το πώς αλληλεπιδρά το φορτίο εργασίας VNF σε επίπεδο πόρων και διεργασίας στο VM που το φιλοξενεί. Η οργάνωση του περιβάλλοντος VM για την παροχή δεδομένων σχετικά με βασικές μετρήσεις σε επίπεδο συστήματος βοηθά να αντιμετωπιστεί το κενό πληροφόρησης μεταξύ της εξωτερικής προοπτικής του εξοπλισμού δοκιμών και της εσωτερικής άποψης μέσα από το VNF και το VM που το φιλοξενεί.

Παρόλο που υπάρχουν ήδη πολλά εργαλεία (κυρίως στην πλευρά των χρηστών) για να βοηθήσουν στη συλλογή και την παρακολούθηση των μετρητών πυρήνα στα συστήματα που βασίζονται στο Linux, παρέχουν λίγη προσαρμογή και, το πιο σημαντικό, προσφέρουν μόνο κατακερματισμένα σύνολα δεδομένων, κάνοντας δύσκολη την ενοποίηση με τα συστήματα ανάλυσης. Αυτό θέτει ένα σημαντικό κόστος στο χρήστη να επεξεργάζεται και να συναρμολογεί τα δεδομένα κατά τρόπο που να μπορεί εύκολα να εξάγει, να συνοψίζει, να αναλύει, να ερμηνεύει και να συσχετίζει τα εξαγώμενα αποτελέσματα με τον εξοπλισμό δοκιμών.

Το NFV αναμφισβήτητα θα παρουσιάσει στους φορείς εκμετάλλευσης δικτύων ορισμένες σημαντικές προκλήσεις εφαρμογής, πολλές από τις οποίες αντιμετωπίζονται ήδη μέσω συνεργατικών δραστηριοτήτων σε ολόκληρο τον κλάδο. Κατά κάποιον τρόπο, η διεξοδική επικύρωση του VNF είναι πιο απαιτητική από το αυτόνομο φυσικό υλικό, καθώς υπάρχουν πολλές περισσότερες πιθανές μεταβολές του υλικού κατασκευής και του hypervisor. Με την εκπόνηση ξεχωριστών

περιπτωσιολογικών μελετών που χρησιμοποιούν πολύ διαφορετικά VNF με μοναδικά χαρακτηριστικά που υπόκεινται σε δοκιμές, τα πλεονεκτήματα των ενσωματωμένων οργάνων έχουν επισημανθεί.

Καθώς η ανάπτυξη των VNF σε δίκτυα μεταφορέων αυξάνει δυναμική, η χρήση ενσωματωμένων οργάνων για τη συμπλήρωση των υφιστάμενων εξωτερικών προσεγγίσεων δοκιμών και των εσωτερικών δυνατοτήτων μέτρησης VNF είναι πιθανό να έχει αυξημένη σημασία και χρησιμότητα στο μέλλον. Υπάρχουν, ωστόσο, πολλά ενδιαφέροντα προβλήματα που πρέπει να διερευνηθούν σε σημαντικό βάθος. Ορισμένοι από αυτούς τους τομείς ενδιαφέροντος περιλαμβάνουν, αλλά δεν περιορίζονται σε:

• Περαιτέρω διερεύνηση υποθέσεων χρήσης που περιλαμβάνουν πολλαπλές εμφανίσεις του ίδιου VNF, καθώς και μια σειρά διαφορετικών VNF στον ίδιο φυσικό εξυπηρετητή. Το τελευταίο σενάριο εκφράζει ανησυχίες σχετικά με τα θορυβώδη γειτονικά φαινόμενα. δηλαδή, αν μπορούμε να είμαστε βέβαιοι ότι η παρουσία ενός VNF δεν επηρεάζει δυσμενώς την απόδοση ενός άλλου VNF.

• Διερεύνηση πιο σύνθετων υπηρεσιών δικτύου που περιλαμβάνουν πολλαπλά VNF που αναπτύσσονται σε ένα μόνο κέντρο δεδομένων ή σε πολλαπλά γεωγραφικά διασκορπισμένα κέντρα δεδομένων.

• Χρήση αποτελεσμάτων από τα όργανα για τον σχεδιασμό κανόνων σχεδιασμού για τους VNF που συγκατοικούν σε διακομιστές, εκθέτοντας συγκεκριμένες λειτουργίες πλατφόρμας για τη βελτίωση των αποφάσεων τοποθέτησης του φορτίου VNF, του τύπου hypervisor και της σειράς VNF που προγραμματίζεται για υποστήριξη σε συγκεκριμένο διακομιστή.

• Προσδιορισμός του τρόπου κλιμάκωσης και αποτελεσματικότητας των VNFs σε περιβάλλοντα cloud commodity και όχι σε προσαρμοσμένους κόμβους υλικού που έχουν βελτιστοποιηθεί για τη συγκεκριμένη λειτουργία [2].

• Η ασφάλεια αποτελεί σημαντική πτυχή της βιομηχανίας τηλεπικοινωνιών. Το NFV πρέπει να αποκτήσει επίπεδο ασφάλειας κοντά σε αυτό ενός ιδιόκτητου περιβάλλοντος φιλοξενίας για λειτουργίες δικτύου. Ο καλύτερος τρόπος για να επιτευχθεί αυτό το επίπεδο ασφάλειας είναι η διαίρεσή του σύμφωνα με λειτουργικούς τομείς. Οι επιθέσεις ασφαλείας αναμένεται να αυξηθούν κατά την εφαρμογή λειτουργιών δικτύου σε περιβάλλον εικονικοποίησης. Θα πρέπει να χρησιμοποιείται ένας προστατευμένος hypervisor για την αποτροπή οποιασδήποτε μη εξουσιοδοτημένης πρόσβασης ή διαρροής δεδομένων. Επιπλέον, άλλες διαδικασίες, όπως η επικοινωνία δεδομένων και η μετανάστευση VM, πρέπει να εκτελούνται σε ασφαλές περιβάλλον. Το NFV χρησιμοποιεί API για να παρέχει προγραμματισμένη ενορχήστρωση και αλληλεπίδραση με την υποδομή του. Αυτά τα API εισάγουν υψηλότερη απειλή ασφάλειας για τα VNF.

• Η ανάπτυξη λειτουργιών εικονικού δικτύου μέσω ενός εικονικού περιβάλλοντος βελτιώνει τη φορητότητα και διασφαλίζει ότι οι πόροι υλικού εξετάζονται ομοιόμορφα από το VNF. Αυτή η ανάπτυξη επιτρέπει επίσης σε κάθε VNF να εκτελεστεί στο συγκεκριμένο λειτουργικό του σύστημα παραμένοντας εν αγνοία του υποκείμενου λειτουργικού συστήματος. Επιπλέον, εξασφαλίζεται η απομόνωση των πόρων VNF επειδή οι VNFs εκτελούνται σε ανεξάρτητα VM που διαχειρίζεται το

στρώμα hypervisor, πράγμα που δεν εγγυάται μη αναμενόμενες αλληλεπιδράσεις μεταξύ τους. Η αυστηρή χαρτογράφηση των πόρων θα πρέπει να χρησιμοποιείται για την εξασφάλιση της απομόνωσης των πόρων.

• Οι λειτουργίες του εικονικού δικτύου θα πρέπει να μπορούν να συνυπάρχουν με τον εξοπλισμό δικτύου παλαιού τύπου. Αυτό σημαίνει ότι θα πρέπει να είναι σε θέση να αλληλεπιδρούν με τα παλαιά συστήματα διαχείρισης έχοντας ελάχιστες επιπτώσεις στα υπάρχοντα δίκτυα, το γράφημα της προώθησης του δικτύου δεν θα πρέπει να επηρεάζεται από την ύπαρξη ενός ή περισσοτέρων VNF και θα πρέπει να εξασφαλίζεται μια ασφαλής μετάβαση μεταξύ περιπτώσεων VNF και φυσικών λειτουργιών, χωρίς καμία διακοπή υπηρεσίας ή επιπτώσεις απόδοσης [3].

Είναι πιθανό ότι ο ρυθμός με τον οποίο αναπτύσσονται τα NFV σε ολόκληρο τον κλάδο κατά τα προσεχή έτη θα επηρεαστεί σε κάποιο βαθμό από το πόσο πληρέστερα αντιμετωπίζονται αυτές οι τεχνικές προκλήσεις.

Βιβλιογραφία

[1] Raphael Vicente Rosa, Claudio Bertoldo, and Christian Esteve Rothenberg, "Take Your VNF to the Gym: A Testing Framework for Automated NFV Performance Benchmarking", IEEE Communications Magazine, September 2017, pp. 110-17

[2] P. Veitch, M. J. McGrath, and V. Bayon, "An Instrumentation and Analytics Framework for Optimal and Robust NFV Deployment," IEEE Commun. Mag., vol. 53, no. 2, Feb. 2015, pp. 126-33

[3] R. Mijumbi et al., "Network Function Virtualization: State-of-the-Art and Research Challenges," IEEE Commun. Surveys & Tutorials, vol. 18, no. 1, 1st qtr. 2016, pp. 236–62.

[4] J. Blendin et al., "Towards a Structured Approach to Developing Benchmarks for Virtual Network Functions," Proc. 2016 5th Euro. Wksp. Software Defined Networks, Oct. 2016

[5] M. Peuster and H. Karl, "Understand Your Chains: Towards Performance Profile-Based Network Service Management," Proc. 5th Euro. Wksp. Software Defined Networks, Oct. 2016

[6] L. Cao et al., "NFV-Vital: A Framework for Characterizing the Performance Virtual Network Functions," Proc. 2015 IEEE Conf. Network Function Virtualization and Software Defined Networks, Nov. 2015, pp. 93–99

[7] Wikipedia, "Virtualization" [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Virtualization [Πρόσβαση Ιανουάριος 2019]

[8] InformIT, "Benefits of NFV | The Journey to Network Functions Virtualization (NFV) Era" [Ηλεκτρονικό]. Available: http://www.informit.com/articles/article.aspx?p=2755705&seqNum=3 [Πρόσβαση Ιανουάριος 2019]

[9] Thrive Networks, "Benefits of Virtualization" [Ηλεκτρονικό]. Available: https://www.thrivenetworks.com/blog/benefits-of-virtualization/ [Πρόσβαση Ιανουάριος 2019]

[10] Wikipedia, "Test automation" [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Test_automation [Πρόσβαση Ιανουάριος 2019]

[11] intigua, “Puppet vs. Chef vs. Ansible vs. SaltStack” [Ηλεκτρονικό]. Available: https://www.intigua.com/blog/puppet-vs.-chef-vs.-ansible-vs.-saltstack [Πρόσβαση Ιανουάριος 2019]

[12] EDUCBA, “Django vs PHP” [Ηλεκτρονικό]. Available: https://www.educba.com/django-vs-php/ [Πρόσβαση Ιανουάριος 2019]

[13] Django, “Django Overview” [Ηλεκτρονικό]. Available: https://www.djangoproject.com/start/overview/ [Πρόσβαση Ιανουάριος 2019]

[14] Elastic, “Elasticsearch Reference [6.5]” [Ηλεκτρονικό]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html [Πρόσβαση Ιανουάριος 2019

[15] Elastic, “Logstash Reference [6.5]” [Ηλεκτρονικό]. Available: https://www.elastic.co/guide/en/logstash/current/index.html [Πρόσβαση Ιανουάριος 2019]

[16] Elastic, “Kibana User Guide [6.5]” [Ηλεκτρονικό]. Available: https://www.elastic.co/guide/en/kibana/current/index.html [Πρόσβαση Ιανουάριος 2019]

[17] Selenium, “Selenium Documentation” [Ηλεκτρονικό]. Available: https://www.seleniumhq.org/docs/ [Πρόσβαση Ιανουάριος 2019]

[18] Ansible, “Ansible Documentation” [Ηλεκτρονικό]. Available: https://docs.ansible.com/ansible/latest/index.html [Πρόσβαση Ιανουάριος 2019]

Παράρτημα – Κώδικας

-*Playbooks*

capture.yml

```
---
- hosts: Raspberry-Pi
  remote_user: ktsir
  gather_facts: no

  tasks:
  - name: Start capturing packets during web browsing
    shell: /usr/sbin/tcpdump -i eth0 port not 22 -w captures/capture.pcap
```

capture-close.yml

```
---
- hosts: Raspberry-Pi
  remote_user: ktsir
  gather_facts: no

  tasks:
  - name: Kill tcpdump
    shell: kill -2 $(ps axf | grep 'tcpdump' | grep -v grep | awk '{print $1 }')
```

selenium-grid-hub.yml

```
---
- hosts: localhost
  gather_facts: no

  tasks:
  - name: Start hub with Selenium Grid
    command: java -jar /selenium/selenium-server-standalone-3.141.0.jar -port 4444 -role hub
```

selenium-grid-node.yml

```
---
- hosts: Raspberry-Pi
  remote_user: ktsir
  gather_facts: no

  tasks:
  - name: Connect as node to Selenium Grid
    shell: DISPLAY=:1 /usr/bin/xvfb-run /usr/lib/jvm/java-8-oracle/bin/java -jar
/home/ktsir/selenium/selenium-server-standalone-3.141.0.jar -port 5555 -role node -hub
http://147.102.40.58:4444/grid/register/ -browser browserName=firefox,platform=LINUX
```

selenium-hub-close.yml

```
---
- hosts: localhost
  gather_facts: no

  tasks:
  - name: Kill Selenium Grid process
    shell: kill -9 $(ps axf | grep 'selenium-server-standalone-3.141.0.jar' | grep -v grep | awk
'{print $1 }')
```

selenium-node-close.yml

```
---
- hosts: Raspberry-Pi
  remote_user: ktsir
  gather_facts: no

  tasks:
  - name: Kill Display
    shell: kill -9 $(ps axf | grep 'DISPLAY=:1' | grep -v grep | awk '{print $1 }')

  - name: Kill XVFB
    shell: kill -9 $(ps axf | grep 'Xvfb' | grep -v grep | awk '{print $1 }')
```

start-server.yml

```
---
- hosts: localhost
  gather_facts: no
  vars:
    ID: "{{ ID }}"

  tasks:
  - name: Register and Power-on ESXi Server
    shell: /home/kostas/gitlab-repository/testautomation/scripts/start-server.sh "{{ ID }}"
```

stop-server.yml

```
---
- hosts: localhost
  gather_facts: no
  vars:
    ID: "{{ ID }}"

  tasks:
  - name: Get IP to Ansible Playbook
    shell: cat /home/kostas/VMIP.txt | grep -o 'ipAddress = ".*"' | awk '{print $3}' | tr -d '"'
    register: IP

  - name: Stop iPerf server on the new VM
    shell: /home/kostas/gitlab-repository/testautomation/scripts/stop-iperf.sh "{{ IP.stdout }}"

  - name: Power-off and Unregister ESXi Server
    shell: /home/kostas/gitlab-repository/testautomation/scripts/stop-server.sh "{{ ID }}"

  - name: Delete IP file on local machine
    shell: rm /home/kostas/VMIP.txt
```

start-server-url.yml

```
---
- hosts: localhost
  gather_facts: no
  vars:
    ID: "{{ ID }}"

  tasks:
  - name: Get new VM IP to local machine
    shell: /home/kostas/gitlab-repository/testautomation/scripts/get-ip.sh "{{ ID }}" >
/home/kostas/VMIP.txt

  - name: Get IP to Ansible Playbook
    shell: cat /home/kostas/VMIP.txt | grep -o 'ipAddress = ".*"' | awk '{print $3}' | tr -d '"'
    register: IP

  - name: Run Page Test on new VM
    shell: timeout 60s python3 /home/kostas/gitlab-
repository/testautomation/website/testAutomation/Tests/test_class_pageHTTP.py "{{ IP.stdout }}"
```

start-command.yml

```
---
- hosts: localhost
  gather_facts: no
  vars:
    ID: "{{ ID }}"
    command: "{{ command }}"

  tasks:
  - name: Get IP to Ansible Playbook
    shell: cat /home/kostas/VMIP.txt | grep -o 'ipAddress = ".*"' | awk '{print $3}' | tr -d '"'
    register: IP

  - name: Start iPerf3 command
    shell: iperf3 -c {{ IP.stdout }} "{{ command }}" > /home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/TrafficReports/report_$( date '+%Y-%m-
%d_%H:%M:%S' ).txt
```

start-iperf.yml

```
---
- hosts: localhost
  gather_facts: no
  vars:
    ID: "{{ ID }}"

  tasks:
  - name: Get new VM IP to local machine
    shell: /home/kostas/gitlab-repository/testautomation/scripts/get-ip.sh "{{ ID }}" >
/home/kostas/VMIP.txt
```

visualization.yml

```
---
- hosts: localhost
  gather_facts: no

  tasks:
  - name: Get system's time
    shell: date '+%Y-%m-%d_%H:%M:%S'
    register: current_date

  - name: Get capture file from Raspberry Pi
    shell: sftp ktsir@147.102.39.10:/home/ktsir/captures/capture.pcap /home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Captures/capture_{{ current_date.stdout
}}.pcap

  - name: Convert .pcap file to .json with TShark
    shell: tshark -r /home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Captures/capture_{{ current_date.stdout
}}.pcap -T json > /home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Captures/packets_{{ current_date.stdout
}}.json
```

-*Scripts*

start-server.sh

```
#!/usr/bin/expect -f
set ID [lindex $argv 0]
set prompt "~] "
spawn ssh labuser@147.102.39.16
expect $prompt
send "vim-cmd solo/registervm /vmfs/volumes/datastore1/VM1/VM1.vmx test_server_$ID\r"
expect $prompt
send "vim-cmd vmsvc/getallvms | grep \"test_server_$ID\"\r"
expect "vim-cmd vmsvc/getallvms | grep \"test_server_$ID\"\r\n"
expect -re "(.*)test_server_"
set vmid $expect_out(0,string)
expect $prompt
send "vim-cmd vmsvc/power.on $vmid\r"
expect $prompt
send "exit\r"
```

start-iperf.sh

```
#!/usr/bin/expect -f
set IP [lindex $argv 0]
set prompt "~$ "
spawn ssh labuser@$IP
expect "yes/no" {
        send "yes\r"
}
expect "assword: "
send "D1ploma!!\r"
expect $prompt
send "iperf3 -s\r"
expect "iperf3 -s\r\n"
expect -re "(.*)---"
send "exit\r"
```

get-ip.sh

```
#!/usr/bin/expect -f
set ID [lindex $argv 0]
set prompt "~] "
spawn ssh labuser@147.102.39.16
expect $prompt
send "vim-cmd vmsvc/getallvms | grep \"test_server_$ID\"\r"
expect "vim-cmd vmsvc/getallvms | grep \"test_server_$ID\"\r\n"
expect -re "(.*)test_server_"
set vmid $expect_out(0,string)
send "vim-cmd vmsvc/get.summary $vmid | grep -E ipAddress\r"
expect "vim-cmd vmsvc/get.summary $vmid | grep -E ipAddress\r\n"
expect -re "(.*)ipAddress"
expect $prompt
send "exit\r"
```

stop-iperf.sh

```
#!/usr/bin/expect -f
set IP [lindex $argv 0]
set prompt "labuser@VM1:~$ "
spawn ssh labuser@$IP
expect "assword: " { send "D1ploma!!\r" }
expect $prompt
send "pidof iperf3\r"
expect "pidof iperf3\r\n"
expect -re "(.*)\r\n$prompt"
set vmid $expect_out(0,string)
expect $prompt
send "exit\r"
```

stop-server.sh

```
#!/usr/bin/expect -f
set ID [lindex $argv 0]
set prompt "~] "
spawn ssh labuser@147.102.39.16
expect $prompt
send "vim-cmd vmsvc/getallvms | grep \"test_server_$ID\"\r"
expect "vim-cmd vmsvc/getallvms | grep \"test_server_$ID\"\r\n"
expect -re "(.*)test_server_"
set vmid $expect_out(0,string)
expect $prompt
send "vim-cmd vmsvc/power.off $vmid\r"
expect $prompt
send "vim-cmd vmsvc/unregister $vmid\r"
expect $prompt
send "exit\r"
```

-Django Framework

admin.py

```
from django.contrib import admin
# Register your models here.
from .models import TrafficTest, ResultsTraffic, PageTest, ResultsPage, SearchTest, ResultsSearch,
LoginTest, ResultsLogin


admin.site.register(TrafficTest)
admin.site.register(ResultsTraffic)
admin.site.register(PageTest)
admin.site.register(ResultsPage)
admin.site.register(SearchTest)
admin.site.register(ResultsSearch)
admin.site.register(LoginTest)
admin.site.register(ResultsLogin)
```

apps.py

```
from django.apps import AppConfig


class TestingConfig(AppConfig):
    name = 'test-automation'
class TestingConfig(AppConfig):
    name = 'test-automation'
```

forms.py

```
from django import forms
from django.contrib.auth.models import User
from .models import PageTest, SearchTest, LoginTest, TrafficTest


class UserForm(forms.ModelForm):
    password = forms.CharField(widget=forms.PasswordInput)

    class Meta:
        model = User
        fields = ['username', 'email', 'password']


class TrafficTestForm(forms.ModelForm):
    Interval = forms.CharField(required=False)
    CPU_Affinity = forms.CharField(required=False)
    Use_SCTP = forms.CheckboxInput()
    Use_UDP = forms.CheckboxInput()
    Bandwidth = forms.CharField(required=False)
    Bytes = forms.CharField(required=False)
    Length = forms.CharField(required=False)
    Parallel = forms.CharField(required=False)
    Window = forms.CharField(required=False)
    MSS = forms.CharField(required=False)
    IPv4 = forms.CheckboxInput()
    IPv6 = forms.CheckboxInput()
    IPToS = forms.CharField(required=False)
    DSCP = forms.CharField(required=False)
    FlowLabel = forms.CharField(required=False)
    SCTP_streams = forms.CharField(required=False)
```

```
    class Meta:
        model = TrafficTest
        fields = ['Interval', 'CPU_Affinity', 'Use_SCTP', 'Use_UDP', 'Bandwidth', 'Bytes',
'Length', 'Parallel',
                  'Window', 'MSS', 'IPv4', 'IPv6', 'IPToS', 'DSCP', 'FlowLabel', 'SCTP_streams']


class PageTestForm(forms.ModelForm):
    URL = forms.CharField(required=False)

    class Meta:
        model = PageTest
        fields = ['URL']


class SearchTestForm(forms.ModelForm):
    Term_to_search = forms.CharField(required=True)

    class Meta:
        model = SearchTest
        fields = ['Term_to_search']


class LoginTestForm(forms.ModelForm):
    Email = forms.EmailField(required=True)
    Password = forms.CharField(widget=forms.PasswordInput(), required=True)

    class Meta:
        model = LoginTest
        fields = ['Email', 'Password']
```

models.py

```
from django.db import models
from django.urls import reverse
from django.contrib.auth.models import User
import os


# Create your models here.
class TrafficTest(models.Model):
    modelType = 10
    user = models.ForeignKey(User, default=1, on_delete=models.CASCADE)
    Interval = models.CharField(null=True, blank=True, max_length=250)
    CPU_Affinity = models.CharField(null=True, blank=True, max_length=250)
    Use_SCTP = models.BooleanField(default=False, null=False)
    Use_UDP = models.BooleanField(default=False, null=False)
    Bandwidth = models.CharField(null=True, blank=True, max_length=250)
    Bytes = models.CharField(null=True, blank=True, max_length=250)
    Length = models.CharField(null=True, blank=True, max_length=250)
    Parallel = models.CharField(null=True, blank=True, max_length=250)
    Window = models.CharField(null=True, blank=True, max_length=250)
    MSS = models.CharField(null=True, blank=True, max_length=250)
    IPv4 = models.BooleanField(default=False, null=False)
    IPv6 = models.BooleanField(default=False, null=False)
    IPToS = models.CharField(null=True, blank=True, max_length=250)
    DSCP = models.CharField(null=True, blank=True, max_length=250)
    FlowLabel = models.CharField(null=True, blank=True, max_length=250)
    SCTP_streams = models.CharField(null=True, blank=True, max_length=250)

    def get_absolute_url(self):
        return reverse('test-automation:detail-traffic', kwargs={'pk': self.pk})

    def __str__(self):
        return self.Bytes
```

```
class ResultsTraffic(models.Model):
    report = models.FileField(upload_to='TrafficReports/')
    resultKey = models.ForeignKey(TrafficTest, on_delete=models.CASCADE)
    modelType = 11
    user = models.ForeignKey(User, default=1, on_delete=models.CASCADE)

    def __str__(self):
        return self.status

    def get_absolute_url(self):
        return reverse('test-automation:logs-traffic', kwargs={'pk': self.pk})


class PageTest(models.Model):
    URL = models.CharField(max_length=250)
    modelType = 20
    user = models.ForeignKey(User, default=1, on_delete=models.CASCADE)

    def get_absolute_url(self):
        return reverse('test-automation:detail-page', kwargs={'pk': self.pk})

    def __str__(self):
        return self.URL


class ResultsPage(models.Model):
    status = models.CharField(max_length=250)
    report = models.FileField(upload_to='Reports/')
    capture = models.FileField(upload_to='Captures/')
    kibana = models.CharField(default="http://jenkins.cn.ece.ntua.gr:5601/app/kibana#/home?_g=()",
max_length=1000)
    resultKey = models.ForeignKey(PageTest, on_delete=models.CASCADE)
    modelType = 21
    user = models.ForeignKey(User, default=1, on_delete=models.CASCADE)

    def __str__(self):
        return self.status

    def get_absolute_url(self):
        return reverse('test-automation:logs-page', kwargs={'pk': self.pk})


class SearchTest(models.Model):
    Term_to_search = models.CharField(max_length=250)
    modelType = 30
    user = models.ForeignKey(User, default=1, on_delete=models.CASCADE)

    def get_absolute_url(self):
        return reverse('test-automation:detail-search', kwargs={'pk': self.pk})

    def __str__(self):
        return self.Term_to_search


class ResultsSearch(models.Model):
    status = models.CharField(max_length=250)
    report = models.FileField(upload_to='Reports/')
    capture = models.FileField(upload_to='Captures/')
    kibana = models.CharField(default="http://jenkins.cn.ece.ntua.gr:5601/app/kibana#/home?_g=()",
max_length=1000)
    resultKey = models.ForeignKey(SearchTest, on_delete=models.CASCADE)
    modelType = 31
    user = models.ForeignKey(User, default=1, on_delete=models.CASCADE)
```

```
    def __str__(self):
        return self.status

    def get_absolute_url(self):
        return reverse('test-automation:logs-search', kwargs={'pk': self.pk})


class LoginTest(models.Model):
    Email = models.CharField(max_length=250)
    Password = models.CharField(max_length=250, default="")
    modelType = 40
    user = models.ForeignKey(User, default=1, on_delete=models.CASCADE)

    def get_absolute_url(self):
        return reverse('test-automation:detail-login', kwargs={'pk': self.pk})

    def __str__(self):
        return self.Email


class ResultsLogin(models.Model):
    status = models.CharField(max_length=250)
    report = models.FileField(upload_to='Reports/')
    capture = models.FileField(upload_to='Captures/')
    kibana = models.CharField(default="http://jenkins.cn.ece.ntua.gr:5601/app/kibana#/home?_g=()",
max_length=1000)
    resultKey = models.ForeignKey(LoginTest, on_delete=models.CASCADE)
    modelType = 41
    user = models.ForeignKey(User, default=1, on_delete=models.CASCADE)

    def __str__(self):
        return self.status

    def get_absolute_url(self):
        return reverse('test-automation:logs-login', kwargs={'pk': self.pk})
```

urls.py

```
from django.urls import path
from . import views
from django.views.generic import RedirectView

app_name = 'test-automation'

urlpatterns = [
    path('', RedirectView.as_view(url='login/')),
    path('test-editor/', views.test_editor, name='test-editor'),
    path('new/traffic-test/', views.new_traffic, name='new-traffic'),
    path('new/page-test/', views.new_page, name='new-page'),
    path('new/search-test/', views.new_search, name='new-search'),
    path('new/login-test/', views.new_login, name='new-login'),
    path('test-editor/traffic/<int:pk>/', views.detail_traffic, name='detail-traffic'),
    path('test-editor/page/<int:pk>/', views.detail_page, name='detail-page'),
    path('test-editor/search/<int:pk>/', views.detail_search, name='detail-search'),
    path('test-editor/login/<int:pk>/', views.detail_login, name='detail-login'),
    path('test-editor/edit/traffic/<int:pk>/', views.edit_traffic, name='edit-traffic'),
    path('test-editor/edit/page/<int:pk>/', views.edit_page, name='edit-page'),
    path('test-editor/edit/search/<int:pk>/', views.edit_search, name='edit-search'),
    path('test-editor/edit/login/<int:pk>/', views.edit_login, name='edit-login'),
    path('test-editor/delete/traffic/<int:pk>/', views.delete_traffic, name='delete-traffic'),
    path('test-editor/delete/page/<int:pk>/', views.delete_page, name='delete-page'),
    path('test-editor/delete/search<int:pk>/', views.delete_search, name='delete-search'),
    path('test-editor/delete/login<int:pk>/', views.delete_login, name='delete-login'),
    path('logs/', views.logs, name='logs'),
    path('logs/traffic/<int:pk>/', views.logs_traffic, name='logs-traffic'),
```

```
    path('logs/page/<int:pk>/', views.logs_page, name='logs-page'),
    path('logs/search/<int:pk>/', views.logs_search, name='logs-search'),
    path('logs/login/<int:pk>/', views.logs_login, name='logs-login'),
    path('run-traffic/<int:pk>', views.run_traffic, name='run-traffic'),
    path('run-page/<int:pk>', views.run_page, name='run-page'),
    path('run-search/<int:pk>', views.run_search, name='run-search'),
    path('run-login/<int:pk>', views.run_login, name='run-login'),
    path('run-multiple/', views.run_multiple, name='run-multiple'),
    path('register/', views.register, name='register'),
    path('login/', views.login_user, name='login-user'),
    path('logout/', views.logout_user, name='logout-user'),
    path('logs/delete/traffic/<int:pk>/', views.delete_traffic_res, name='delete-traffic-res'),
    path('logs/delete/page/<int:pk>/', views.delete_page_res, name='delete-page-res'),
    path('logs/delete/search<int:pk>/', views.delete_search_res, name='delete-search-res'),
    path('logs/delete/login<int:pk>/', views.delete_login_res, name='delete-login-res'),
]
```

views.py

```
from django.contrib.auth import login, authenticate, logout
from django.shortcuts import render, get_object_or_404
from .models import TrafficTest, PageTest, SearchTest, LoginTest, ResultsTraffic, ResultsPage,
ResultsSearch, \
    ResultsLogin
from .forms import TrafficTestForm, PageTestForm, SearchTestForm, LoginTestForm, UserForm
from itertools import chain
from os import system, environ
from os.path import getctime, basename, splitext
from glob import glob
from bs4 import BeautifulSoup
from time import sleep
import subprocess


def return_query_set_test(request):
    qset1 = TrafficTest.objects.filter(user=request.user)
    qset2 = PageTest.objects.filter(user=request.user)
    qset3 = SearchTest.objects.filter(user=request.user)
    qset4 = LoginTest.objects.filter(user=request.user)
    return list(chain(qset1, qset2, qset3, qset4))


def return_query_set_result(request):
    qset1 = ResultsTraffic.objects.filter(user=request.user)
    qset2 = ResultsPage.objects.filter(user=request.user)
    qset3 = ResultsSearch.objects.filter(user=request.user)
    qset4 = ResultsLogin.objects.filter(user=request.user)
    return list(chain(qset1, qset2, qset3, qset4))


def register(request):
    form = UserForm(request.POST or None)
    if form.is_valid():
        user = form.save(commit=False)
        username = form.cleaned_data['username']
        password = form.cleaned_data['password']
        user.set_password(password)
        user.save()
        user = authenticate(username=username, password=password)
        if user is not None:
            if user.is_active:
                login(request, user)
                query_set = return_query_set_test(request)
                return render(request, 'test-automation/test-editor.html', {'allTests': query_set})
    return render(request, 'test-automation/register.html', {'form': form})
```

```
def login_user(request):
    if request.method == "POST":
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(username=username, password=password)
        if user is not None:
            if user.is_active:
                login(request, user)
                query_set = return_query_set_test(request)
                return render(request, 'test-automation/test-editor.html', {'allTests': query_set})
            else:
                return render(request, 'test-automation/login-user.html',
                              {'error_message': 'Your account has been disabled'})
        else:
            return render(request, 'test-automation/login-user.html', {'error_message': 'Invalid
login'})
    return render(request, 'test-automation/login-user.html')


def logout_user(request):
    logout(request)
    form = UserForm(request.POST or None)
    return render(request, 'test-automation/login-user.html', {'form': form,})


def test_editor(request):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        query_set = return_query_set_test(request)
        return render(request, 'test-automation/test-editor.html', {'allTests': query_set})


def logs(request):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        query_set = return_query_set_result(request)
        return render(request, 'test-automation/logs.html', {'allResults': query_set})


def new_traffic(request):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        form = TrafficTestForm(request.POST or None)
        if form.is_valid():
            test = form.save(commit=False)
            test.user = request.user
            test.save()
            return render(request, 'test-automation/detail-traffic.html', {'test': test})
        return render(request, 'test-automation/traffictest_form.html', {'form': form,})


def new_page(request):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        form = PageTestForm(request.POST or None)
        if form.is_valid():
            test = form.save(commit=False)
            option = request.POST.get('referencepage')
            if option == "True":
                test.URL = "Reference Page"
            test.user = request.user
            test.save()
```

```
            return render(request, 'test-automation/detail-page.html',  {'test': test})
        return render(request, 'test-automation/pagetest_form.html', {'form': form,})


def new_search(request):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        form = SearchTestForm(request.POST or None)
        if form.is_valid():
            test = form.save(commit=False)
            test.user = request.user
            test.save()
            return render(request, 'test-automation/detail-search.html',  {'test': test})
        return render(request, 'test-automation/searchtest_form.html', {'form': form,})


def new_login(request):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        form = LoginTestForm(request.POST or None)
        if form.is_valid():
            test = form.save(commit=False)
            test.user = request.user
            test.save()
            return render(request, 'test-automation/detail-login.html',  {'test': test})
        return render(request, 'test-automation/logintest_form.html', {'form': form,})


def delete_traffic(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        test = TrafficTest.objects.get(pk=test_id.get('pk'))
        test.delete()
        query_set = return_query_set_test(request)
        return render(request, 'test-automation/test-editor.html', {'allTests': query_set})


def delete_page(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        test = PageTest.objects.get(pk=test_id.get('pk'))
        test.delete()
        query_set = return_query_set_test(request)
        return render(request, 'test-automation/test-editor.html', {'allTests': query_set})


def delete_search(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        test = SearchTest.objects.get(pk=test_id.get('pk'))
        test.delete()
        query_set = return_query_set_test(request)
        return render(request, 'test-automation/test-editor.html', {'allTests': query_set})


def delete_login(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        test = LoginTest.objects.get(pk=test_id.get('pk'))
        test.delete()
```

```
        query_set = return_query_set_test(request)
        return render(request, 'test-automation/test-editor.html', {'allTests': query_set})


def delete_traffic_res(request, **result_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        result = ResultsTraffic.objects.get(pk=result_id.get('pk'))
        result.delete()
        system("rm /home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/TrafficReports/" + result.report.name)
        query_set = return_query_set_result(request)
        return render(request, 'test-automation/logs.html', {'allResults': query_set})


def delete_page_res(request, **result_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        result = ResultsPage.objects.get(pk=result_id.get('pk'))
        result.delete()
        system("rm /home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Reports/" + result.report.name)
        system("rm /home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Captures/" + result.capture.name)
        date = splitext(result.capture.name)[0]
        json = "packets_" + date.split('_')[1]
        system("rm /home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Reports/" + json + ".json")
        query_set = return_query_set_result(request)
        return render(request, 'test-automation/logs.html', {'allResults': query_set})


def delete_search_res(request, **result_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        result = ResultsSearch.objects.get(pk=result_id.get('pk'))
        result.delete()
        system("rm /home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Reports/" + result.report.name)
        system("rm /home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Captures/" + result.capture.name)
        date = splitext(result.capture.name)[0]
        json = "packets_" + date.split('_')[1]
        system("rm /home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Reports/" + json + ".json")
        query_set = return_query_set_result(request)
        return render(request, 'test-automation/logs.html', {'allResults': query_set})


def delete_login_res(request, **result_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        result = ResultsLogin.objects.get(pk=result_id.get('pk'))
        result.delete()
        system("rm /home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Reports/" + result.report.name)
        system("rm /home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Captures/" + result.capture.name)
        date = splitext(result.capture.name)[0]
        json = "packets_" + date.split('_')[1]
        system("rm /home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Reports/" + json + ".json")
        query_set = return_query_set_result(request)
```

```
        return render(request, 'test-automation/logs.html', {'allResults': query_set})


def detail_traffic(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        user = request.user
        test = get_object_or_404(TrafficTest, pk=test_id.get('pk'))
        return render(request, 'test-automation/detail-traffic.html', {'test': test, 'user': user})


def detail_page(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        user = request.user
        test = get_object_or_404(PageTest, pk=test_id.get('pk'))
        return render(request, 'test-automation/detail-page.html', {'test': test, 'user': user})


def detail_search(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        user = request.user
        test = get_object_or_404(SearchTest, pk=test_id.get('pk'))
        return render(request, 'test-automation/detail-search.html', {'test': test, 'user': user})


def detail_login(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        user = request.user
        test = get_object_or_404(LoginTest, pk=test_id.get('pk'))
        return render(request, 'test-automation/detail-login.html', {'test': test, 'user': user})


def edit_traffic(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        test = TrafficTest.objects.get(pk=test_id.get('pk'))
        if request.method == "POST":
            form = TrafficTestForm(request.POST, instance=test)
            if form.is_valid():
                test = form.save(commit=False)
                test.user = request.user
                test.save()
                return redirect('test-automation:test-editor')
        else:
            form = TrafficTestForm(instance=test)
        return render(request, 'test-automation/traffictest_form.html', {'form': form})


def edit_page(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        test = PageTest.objects.get(pk=test_id.get('pk'))
        if request.method == "POST":
            form = PageTestForm(request.POST, instance=test)
            if form.is_valid():
                test = form.save(commit=False)
                test.user = request.user
```

```
                test.save()
                return redirect('test-automation:test-editor')
        else:
            form = PageTestForm(instance=test)
        return render(request, 'test-automation/pagetest_form.html', {'form': form})


def edit_search(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        test = SearchTest.objects.get(pk=test_id.get('pk'))
        if request.method == "POST":
            form = SearchTestForm(request.POST, instance=test)
            if form.is_valid():
                test = form.save(commit=False)
                test.user = request.user
                test.save()
                return redirect('test-automation:test-editor')
        else:
            form = SearchTestForm(instance=test)
        return render(request, 'test-automation/searchtest_form.html', {'form': form})


def edit_login(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        test = LoginTest.objects.get(pk=test_id.get('pk'))
        if request.method == "POST":
            form = LoginTestForm(request.POST, instance=test)
            if form.is_valid():
                test = form.save(commit=False)
                test.user = request.user
                test.save()
                return redirect('test-automation:test-editor')
        else:
            form = LoginTestForm(instance=test)
        return render(request, 'test-automation/logintest_form.html', {'form': form})


def logs_traffic(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        user = request.user
        result = get_object_or_404(ResultsTraffic, pk=test_id.get('pk'))
        return render(request, 'test-automation/logs-traffic.html', {'result': result, 'user':
user})


def logs_page(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        user = request.user
        result = get_object_or_404(ResultsPage, pk=test_id.get('pk'))
        return render(request, 'test-automation/logs-page.html', {'result': result, 'user': user})


def logs_search(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        user = request.user
        result = get_object_or_404(ResultsSearch, pk=test_id.get('pk'))
        return render(request, 'test-automation/logs-search.html', {'result': result, 'user':
user})
```

```
def logs_login(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        user = request.user
        result = get_object_or_404(ResultsLogin, pk=test_id.get('pk'))
        return render(request, 'test-automation/logs-login.html', {'result': result, 'user': user})


def run_traffic(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        test = TrafficTest.objects.filter(user=request.user).get(pk=test_id.get('pk'))
        command = ""
        if test.Interval:
            command += " -i "
            command += test.Interval
        if test.CPU_Affinity:
            command += " -A "
            command += test.CPU_Affinity
        if test.Use_SCTP:
            command += " --sctp"
        if test.Use_UDP:
            command += " -u"
        if test.Bandwidth:
            command += " -b "
            command += test.Bandwidth
        if test.Bytes:
            command += " -n "
            command += test.Bytes
        if test.Length:
            command += " -l "
            command += test.Length
        if test.Parallel:
            command += " -P "
            command += test.Parallel
        if test.Window:
            command += " -w "
            command += test.Window
        if test.MSS:
            command += " -M "
            command += test.MSS
        if test.IPv4:
            command += " -4"
        if test.IPv6:
            command += " -6"
        if test.IPToS:
            command += " -S "
            command += test.IPToS
        if test.DSCP:
            command += " --dscp "
            command += test.DSCP
        if test.FlowLabel:
            command += " -L "
            command += test.FlowLabel
        if test.SCTP_streams:
            command += " --nstreams "
            command += test.SCTP_streams
        ID = str(request.user.id)
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/start-
server.yml --extra-vars \"ID=" + ID + "\" &")
        sleep(65)
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/start-
iperf.yml --extra-vars \"ID=" + ID + "\" &")
        sleep(5)
```

```
        IP = subprocess.check_output("cat /home/kostas/VMIP.txt | grep -o 'ipAddress = \".*\"' |
awk '{print $3}' | tr -d '\"'", shell=True)
        system("/home/kostas/gitlab-repository/testautomation/scripts/start-iperf.sh \"" +
IP.decode('utf-8') + "\" &")
        sleep(10)
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/start-
command.yml --extra-vars \"ID=" + ID + " command=\\""" + command + "\\"" &")
        sleep(20)
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/stop-
server.yml --extra-vars \"ID=" + ID + "\" &")
        sleep(10)
        list_of_files = glob("/home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/TrafficReports/*.txt")
        latest_file = max(list_of_files, key=getctime)
        result = ResultsTraffic()
        result.resultKey = test
        result.report = basename(latest_file)
        result.user = request.user
        result.save()
        query_set = return_query_set_result(request)
        return render(request, 'test-automation/logs.html', {'allResults': query_set})


def run_page(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        test = PageTest.objects.filter(user=request.user).get(pk=test_id.get('pk'))
        system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/capture.yml &")
        sleep(2)
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/selenium-
grid-hub.yml &")
        sleep(4)
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/selenium-
grid-node.yml &")
        sleep(8)
        if test.URL == "Reference Page":
            ID = str(request.user.id)
            system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/start-
server.yml --extra-vars \"ID=" + ID + "\" &")
            sleep(65)
            system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/start-
server-url.yml --extra-vars \"ID=" + ID + "\" &")
            sleep(90)
            system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/stop-
server.yml --extra-vars \"ID=" + ID + "\" &")
            sleep(20)
        else:
            system("timeout 60s python3 /home/kostas/gitlab-
repository/testautomation/website/testAutomation/Tests/test_class_pageHTTP.py " + test.URL + " &")
            sleep(60)
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/selenium-
hub-close.yml &")
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/capture-
close.yml &")
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/selenium-
node-close.yml &")
        sleep(5)
        system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/visualization.yml &")
        sleep(5)
        list_of_files = glob("/home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Reports/*.html")
        latest_file = max(list_of_files, key=getctime)
        soup = BeautifulSoup(open(latest_file), 'html.parser')
        result = ResultsPage()
```

```
        result.resultKey = test
        status = soup.findAll("span")[0].string.strip(' ')
        if status == "Error":
            p = soup.findAll("p")
            status = status + '\n' + p
        result.status = status
        result.report = basename(latest_file)
        list_of_files = glob("/home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Captures/*.pcap")
        latest_file = max(list_of_files, key=getctime)
        result.capture = basename(latest_file)
        date = splitext(result.capture.name)[0]
        json = "packets_" + date.split('_')[1]
        result.kibana =
"http://jenkins.cn.ece.ntua.gr:5601/app/kibana#/discover?_g=(refreshInterval:(pause:!t,value:0),tim
e:(from:now%2FM,mode:quick,to:now%2FM))&_a=(columns:!(_source),index:'6b602b90-3391-11e9-9589-
891a96223f22',interval:d,query:(language:kuery,query:'path%20:%20\"" + json +
"\"'),sort:!('@timestamp',desc))"
        result.user = request.user
        result.save()
        query_set = return_query_set_result(request)
        return render(request, 'test-automation/logs.html', {'allResults': query_set})


def run_search(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        test = SearchTest.objects.filter(user=request.user).get(pk=test_id.get('pk'))
        system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/capture.yml &")
        sleep(2)
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/selenium-
grid-hub.yml &")
        sleep(4)
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/selenium-
grid-node.yml &")
        sleep(8)
        system("timeout 60s python3 /home/kostas/gitlab-
repository/testautomation/website/testAutomation/Tests/test_class_searchGoogle.py " +
test.Term_to_search + " &")
        sleep(60)
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/selenium-
hub-close.yml &")
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/capture-
close.yml &")
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/selenium-
node-close.yml &")
        sleep(5)
        system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/visualization.yml &")
        sleep(5)
        list_of_files = glob("/home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Reports/*.html")
        latest_file = max(list_of_files, key=getctime)
        soup = BeautifulSoup(open(latest_file), 'html.parser')
        result = ResultsSearch()
        result.resultKey = test
        status = soup.findAll("span")[0].string.strip(' ')
        if status == "Error":
            p = soup.findAll("p")
            status = status + '\n' + p
        result.status = status
        result.report = basename(latest_file)
        list_of_files = glob("/home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Captures/*.pcap")
```

```
        latest_file = max(list_of_files, key=getctime)
        result.capture = basename(latest_file)
        date = splitext(result.capture.name)[0]
        json = "packets_" + date.split('_')[1]
        result.kibana =
"http://jenkins.cn.ece.ntua.gr:5601/app/kibana#/discover?_g=(refreshInterval:(pause:!t,value:0),tim
e:(from:now%2FM,mode:quick,to:now%2FM))&_a=(columns:!(_source),index:'6b602b90-3391-11e9-9589-
891a96223f22',interval:d,query:(language:kuery,query:'path%20:%20\"" + json +
"\"'),sort:!('@timestamp',desc))"
        result.user = request.user
        result.save()
        query_set = return_query_set_result(request)
        return render(request, 'test-automation/logs.html', {'allResults': query_set})


def run_login(request, **test_id):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        test = LoginTest.objects.filter(user=request.user).get(pk=test_id.get('pk'))
        system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/capture.yml &")
        sleep(2)
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/selenium-
grid-hub.yml &")
        sleep(4)
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/selenium-
grid-node.yml &")
        sleep(8)
        system("timeout 60s python3 /home/kostas/gitlab-
repository/testautomation/website/testAutomation/Tests/test_class_loginFacebook.py " + test.Email +
" " + test.Password + " &")
        sleep(60)
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/selenium-
hub-close.yml &")
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/capture-
close.yml &")
        system("ansible-playbook /home/kostas/gitlab-repository/testautomation/playbooks/selenium-
node-close.yml &")
        sleep(5)
        system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/visualization.yml &")
        sleep(5)
        list_of_files = glob("/home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Reports/*.html")
        latest_file = max(list_of_files, key=getctime)
        soup = BeautifulSoup(open(latest_file), 'html.parser')
        result = ResultsLogin()
        result.resultKey = test
        status = soup.findAll("span")[0].string.strip(' ')
        if status == "Error":
            p = soup.findAll("p")
            status = status + '\n' + p
        result.status = status
        result.report = basename(latest_file)
        list_of_files = glob("/home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Captures/*.pcap")
        latest_file = max(list_of_files, key=getctime)
        result.capture = basename(latest_file)
        date = splitext(result.capture.name)[0]
        json = "packets_" + date.split('_')[1]
        result.kibana =
"http://jenkins.cn.ece.ntua.gr:5601/app/kibana#/discover?_g=(refreshInterval:(pause:!t,value:0),tim
e:(from:now%2FM,mode:quick,to:now%2FM))&_a=(columns:!(_source),index:'6b602b90-3391-11e9-9589-
891a96223f22',interval:d,query:(language:kuery,query:'path%20:%20\"" + json +
"\"'),sort:!('@timestamp',desc))"
        result.user = request.user
```

```
        result.save()
        query_set = return_query_set_result(request)
        return render(request, 'test-automation/logs.html', {'allResults': query_set})


def run_multiple(request):
    if not request.user.is_authenticated:
        return render(request, 'test-automation/login-user.html')
    else:
        if request.method == "POST":
            list_of_ids = request.POST.getlist('tests')
            ids20 = []
            ids30 = []
            ids40 = []
            for i in list_of_ids:
                model_id = i.split('/')
                if model_id[0] == '20':
                    ids20.append(model_id[1])
                elif model_id[0] == '30':
                    ids30.append(model_id[1])
                elif model_id[0] == '40':
                    ids40.append(model_id[1])
            system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/capture.yml &")
            sleep(2)
            system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/selenium-grid-hub.yml &")
            sleep(4)
            system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/selenium-grid-node.yml &")
            sleep(8)
            for test_id in ids20:
                test = PageTest.objects.filter(user=request.user).get(pk=test_id)
                if test.URL == "Reference Page":
                    ID = str(request.user.id)
                    system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/start-server.yml --extra-vars \"ID=" + ID + "\" &")
                    sleep(65)
                    system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/start-server-url.yml --extra-vars \"ID=" + ID + "\" &")
                    sleep(90)
                    system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/stop-server.yml --extra-vars \"ID=" + ID + "\" &")
                    sleep(20)
                else:
                    system("timeout 60s python3 /home/kostas/gitlab-
repository/testautomation/website/testAutomation/Tests/test_class_pageHTTP.py " + test.URL + " &")
                    sleep(60)
                system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/capture-close.yml &")
                sleep(5)
                system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/visualization.yml &")
                sleep(5)
                list_of_files = glob("/home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Reports/*.html")
                latest_file = max(list_of_files, key=getctime)
                soup = BeautifulSoup(open(latest_file), 'html.parser')
                result = ResultsPage()
                result.resultKey = test
                status = soup.findAll("span")[0].string.strip(' ')
                if status == "Error":
                    p = soup.findAll("p")
                    status = status + '\n' + p
                result.status = status
                result.report = basename(latest_file)
```

```
                list_of_files = glob("/home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Captures/*.pcap")
                latest_file = max(list_of_files, key=getctime)
                result.capture = basename(latest_file)
                date = splitext(result.capture.name)[0]
                json = "packets_" + date.split('_')[1]
                result.kibana =
"http://jenkins.cn.ece.ntua.gr:5601/app/kibana#/discover?_g=(refreshInterval:(pause:!t,value:0),tim
e:(from:now%2FM,mode:quick,to:now%2FM))&_a=(columns:!(_source),index:'6b602b90-3391-11e9-9589-
891a96223f22',interval:d,query:(language:kuery,query:'path%20:%20\"" + json +
"\"'),sort:!('@timestamp',desc))"
                result.user = request.user
                result.save()
            for test_id in ids30:
                test = SearchTest.objects.filter(user=request.user).get(pk=test_id)
                system("timeout 60s python3 /home/kostas/gitlab-
repository/testautomation/website/testAutomation/Tests/test_class_searchGoogle.py " +
test.Term_to_search + " &")
                sleep(60)
                system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/capture-close.yml &")
                sleep(5)
                system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/visualization.yml &")
                sleep(5)
                list_of_files = glob("/home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Reports/*.html")
                latest_file = max(list_of_files, key=getctime)
                soup = BeautifulSoup(open(latest_file), 'html.parser')
                result = ResultsSearch()
                result.resultKey = test
                status = soup.findAll("span")[0].string.strip(' ')
                if status == "Error":
                    p = soup.findAll("p")
                    status = status + '\n' + p
                result.status = status
                result.report = basename(latest_file)
                list_of_files = glob("/home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Captures/*.pcap")
                latest_file = max(list_of_files, key=getctime)
                result.capture = basename(latest_file)
                date = splitext(result.capture.name)[0]
                json = "packets_" + date.split('_')[1]
                result.kibana =
"http://jenkins.cn.ece.ntua.gr:5601/app/kibana#/discover?_g=(refreshInterval:(pause:!t,value:0),tim
e:(from:now%2FM,mode:quick,to:now%2FM))&_a=(columns:!(_source),index:'6b602b90-3391-11e9-9589-
891a96223f22',interval:d,query:(language:kuery,query:'path%20:%20\"" + json +
"\"'),sort:!('@timestamp',desc))"
                result.user = request.user
                result.save()
            for test_id in ids40:
                test = LoginTest.objects.filter(user=request.user).get(pk=test_id)
                system("timeout 60s python3 /home/kostas/gitlab-
repository/testautomation/website/testAutomation/Tests/test_class_loginFacebook.py " + test.Email +
" " + test.Password + " &")
                sleep(60)
                system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/capture-close.yml &")
                sleep(5)
                system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/visualization.yml &")
                sleep(5)
                list_of_files = glob("/home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Reports/*.html")
                latest_file = max(list_of_files, key=getctime)
                soup = BeautifulSoup(open(latest_file), 'html.parser')
                result = ResultsLogin()
```

```
                result.resultKey = test
                status = soup.findAll("span")[0].string.strip(' ')
                if status == "Error":
                    p = soup.findAll("p")
                    status = status + '\n' + p
                result.status = status
                result.report = basename(latest_file)
                list_of_files = glob("/home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Captures/*.pcap")
                latest_file = max(list_of_files, key=getctime)
                result.capture = basename(latest_file)
                date = splitext(result.capture.name)[0]
                json = "packets_" + date.split('_')[1]
                result.kibana =
"http://jenkins.cn.ece.ntua.gr:5601/app/kibana#/discover?_g=(refreshInterval:(pause:!t,value:0),tim
e:(from:now%2FM,mode:quick,to:now%2FM))&_a=(columns:!(_source),index:'6b602b90-3391-11e9-9589-
891a96223f22',interval:d,query:(language:kuery,query:'path%20:%20\"" + json +
"\"'),sort:!('@timestamp',desc))"
                result.user = request.user
                result.save()
            system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/selenium-hub-close.yml &")
            system("ansible-playbook /home/kostas/gitlab-
repository/testautomation/playbooks/selenium-node-close.yml &")
            query_set = return_query_set_result(request)
            return render(request, 'test-automation/logs.html', {'allResults': query_set})
```

-Templates

base-template-visitor.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width = device-width, initial-scale = 1">
    <title>{% block title %} Test Automation Framework - T A F {% endblock %}</title>
    {% load staticfiles %}
    <link rel="shortcut icon" type="image/jpg" href="{%  static 'test-automation/images/icon.ico'
%}"/>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
          integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
          crossorigin="anonymous">
    <link href="https://fonts.googleapis.com/css?family=Work+Sans" rel="stylesheet">
    <link rel="stylesheet" type="text/css" href="{% static 'test-automation/styles.css' %}"/>
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
            integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
            crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"
            integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
            crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"
            integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
            crossorigin="anonymous"></script>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <!-- Header -->
    <a class="navbar-brand" href="{% url 'test-automation:test-editor' %}">T A F</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#topNavBar"
            aria-controls="topNavBar" aria-expanded="false" aria-label="Toggle navigation">>
        <span class="navbar-toggler-icon"></span>
    </button>
```

```
    <!-- Items -->
    <div class="collapse navbar-collapse" id="topNavBar">
        <ul class="navbar-nav mr-auto">
            <li class="{% block register %}{% endblock %}">
                <a class="nav-link" href="{% url 'test-automation:register' %}">
                    Register <span class="sr-only">(current)</span></a>
            </li>
            <li class="{% block login %}{% endblock %}">
                <a class="nav-link" href="{% url 'test-automation:login-user' %}">
                    Login <span class="sr-only">(current)</span></a>
            </li>
        </ul>
    </div>
</nav>
{% block body %}
{% endblock %}
</body>
</html>
```

base-template.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width = device-width, initial-scale = 1">
    <title>{% block title %} Test Automation Framework - T A F {% endblock %}</title>
    {% load staticfiles %}
    <link rel="shortcut icon" type="image/jpg" href="{%  static 'test-automation/images/icon.ico'
%}"/>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
          integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
          crossorigin="anonymous">
    <link href="https://fonts.googleapis.com/css?family=Work+Sans" rel="stylesheet">
    <link rel="stylesheet" type="text/css" href="{% static 'test-automation/styles.css' %}"/>
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
            integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
            crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"
            integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
            crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"
            integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
            crossorigin="anonymous"></script>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <!-- Header -->
    <a class="navbar-brand" href="{% url 'test-automation:test-editor' %}">T A F</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#topNavBar"
            aria-controls="topNavBar" aria-expanded="false" aria-label="Toggle navigation">>
        <span class="navbar-toggler-icon"></span>
    </button>
    <!-- Items -->
    <div class="collapse navbar-collapse" id="topNavBar">
        <ul class="navbar-nav mr-auto">
            <li class="{% block tests %}{% endblock %}">
                <a class="nav-link" href="{% url 'test-automation:test-editor' %}">
                    Test Editor <span class="sr-only">(current)</span></a>
            </li>
            <li class="{% block logs %}{% endblock %}">
                <a class="nav-link" href="{% url 'test-automation:logs' %}">
                    Test Logs </a>
            </li>
            <li class="{% block new %}{% endblock %} dropdown">
```

```
                <a class="nav-link dropdown-toggle" href="#" id="MenuCreateTest" data-
toggle="dropdown"
                   aria-haspopup="true" aria-expanded="false"> Create New Test </a>
                <div class="dropdown-menu" aria-labelledby="MenuCreateTest">
                    <a class="dropdown-item" href="{% url 'test-automation:new-traffic' %}">
Traffic Test </a>
                    <a class="dropdown-item" href="{% url 'test-automation:new-page' %}"> Page Test
</a>
                    <a class="dropdown-item" href="{% url 'test-automation:new-search' %}"> Search
Test </a>
                    <a class="dropdown-item" href="{% url 'test-automation:new-login' %}"> Login
Test </a>
                </div>
            </li>
        </ul>
        <ul class="nav navbar-nav navbar-right">
            <li class="{% block logout %}{% endblock %}">
                <a class="nav-link" href="{% url 'test-automation:logout-user' %}">
                    Logout <span class="sr-only">(current)></span></a>
            </li>
        </ul>
    </div>
</nav>
{% block body %}
{% endblock %}
</body>
</html>
```

register.html

```
{% extends 'test-automation/base-template.html' %}
{% block title %}Result {{ result.id }} for Login Test {{ result.resultKey.id }} - Result: {{
result.status }}{% endblock %}
{% block logs %}active{% endblock %}
{% block body %}
<br>
<div class="container-fluid">
    <div class="row">
        <!-- Test Parameters -->
        <div class="col-sm-4 col-md-12">
            <div class="card text-white bg-dark mb-3">
                <div class="card-body">
                    <h2 class="card-title">Result {{ result.id }} for Login Test {{
result.resultKey.id }}</h2>
                    Status: {{ result.status }}<br>
                    Report: <a
                            href="/media/Reports/{{ result.report }}"
                            download="Report{{ result.id }}-Test{{ result.resultKey.id }}">
Report_{{ result.id }}.html</a><br>
                    Capture File: <a
                            href="/media/Captures/{{ result.capture }}"
                            download="Capture{{ result.id }}-Test{{ result.resultKey.id
}}">capture.pcap</a><br>
                    Link to Kibana: <a href="{{ result.kibana }}">Link to Kibana
Visualizations</a><br>
                    <!-- Back -->
                    <form action="{% url 'test-automation:logs' %}" method="post"
                          style="display: inline;">
                        {% csrf_token %}
                        <button type="submit" class="btn btn-default btn-sm"> Back</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

login-user.html

```
{% extends 'test-automation/base-template-visitor.html' %}
{% block title %}Login{% endblock %}
{% block login %}active{% endblock %}
{% block body %}
<br>
<div class="container-fluid">
    <div class="row">
        <div class="col-sm-2 col-md-12">
            <div class="card text-white bg-dark mb-3">
                <div class="card-body">
                    <h3 class="card-title">Log In</h3>
                    {% if error_message %}
                    <p><strong>{{ error_message }}</strong></p>
                    {% endif %}
                    <form class="form-horizontal" role="form" action="{% url 'test-
automation:login-user'%}" method="post"
                          enctype="multipart/form-data">
                        {% csrf_token %}
                        <div class="form-group">
                            <label class="control-label col-sm-2" for="id_username">
                                Username:
                            </label>
                            <div class="col-sm-10">
                                <input id="id_username" maxlength="30" name="username" type="text">
                            </div>
                        </div>
                        <div class="form-group">
                            <label class="control-label col-sm-2" for="id_password">
                                Password:
                            </label>
                            <div class="col-sm-10">
                                <input id="id_password" maxlength="30" name="password"
type="password">
                            </div>
                        </div>
                        <div class="form-group">
                            <div class="col-sm-offset-2 col-sm-10">
                                <button type="submit" class="btn btn-success">Submit</button>
                            </div>
                        </div>
                    </form>
                    <div class="card-footer text-muted">
                        Don't have an account? Click <a href="{% url 'test-automation:register'
%}">here</a> to
                        register.
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

test-editor.html

```
{% extends 'test-automation/base-template.html' %}
{% block tests %}active{% endblock %}
<title>{% block title %}T A F - Test Editor{% endblock %}</title>
{% block body %}
<br>
<div class="container-fluid">
    <!-- Tests -->
    {% if allTests %}
    <div class="row">
        <div class="col-6">
            <h2>All Tests</h2>
            <h3>{{ user.username }}'s Tests</h3>
            {% for test in allTests %}
            {% if test.modelType == 10 %}
            <div class="col">
                <div class="card text-white bg-dark mb-3">
                    <div class="card-body">
                        <div class="row">
                            <div class="col">
                                <a href="{% url 'test-automation:detail-traffic' test.id %}"
class="card-title">
                                    Traffic Test {{ test.id }}</a>
                            </div>
                            <!-- Edit Test -->
                            <div class="col-10">
                                <form action="{% url 'test-automation:edit-traffic' test.id %}"
                                      style="display: inline;">
                                    {% csrf_token %}
                                    <input type="hidden" name="test_id" value="{{ test.id }}"/>
                                    <button type="submit" class="btn btn-info btn-sm">Edit</button>
                                </form>
                                <!-- Delete Test -->
                                <form action="{% url 'test-automation:delete-traffic' test.id %}"
method="post"
                                      style="display: inline;">
                                    {% csrf_token %}
                                    <input type="hidden" name="test_id" value="{{ test.id }}"/>
                                    <button type="submit" class="btn btn-danger btn-
sm">Delete</button>
                                </form>
                                <!-- Run Test -->
                                <form action="{% url 'test-automation:run-traffic' test.id %}"
method="post"
                                      style="display: inline;">
                                    {% csrf_token %}
                                    <input type="hidden" name="test_id" value="{{ test.id }}"/>
                                    <button type="submit" class="btn btn-success btn-
sm">Run</button>
                                </form>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
            {% elif test.modelType == 20 %}
            <div class="col">
                <div class="card text-white bg-dark mb-3">
                    <div class="card-body">
                        <div class="row">
                            <div class="col">
                                <a href="{% url 'test-automation:detail-page' test.id %}"
class="card-title">
                                    Page Test {{ test.id }}</a>
                            </div>
```

```
                            <!-- Edit Test -->
                            <div class="col-10">
                                <form action="{% url 'test-automation:edit-page' test.id %}"
                                      style="display: inline;">
                                    {% csrf_token %}
                                    <input type="hidden" name="test_id" value="{{ test.id }}"/>
                                    <button type="submit" class="btn btn-info btn-sm">Edit</button>
                                </form>
                                <!-- Delete Album -->
                                <form action="{% url 'test-automation:delete-page' test.id %}"
method="post"
                                      style="display: inline;">
                                    {% csrf_token %}
                                    <input type="hidden" name="test_id" value="{{ test.id }}"/>
                                    <button type="submit" class="btn btn-danger btn-
sm">Delete</button>
                                </form>
                                <!-- Run Test -->
                                <form action="{% url 'test-automation:run-page' test.id %}"
method="post"
                                      style="display: inline;">
                                    {% csrf_token %}
                                    <input type="hidden" name="test_id" value="{{ test.id }}"/>
                                    <button type="submit" class="btn btn-success btn-
sm">Run</button>
                                </form>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
            {% elif test.modelType == 30 %}
            <div class="col">
                <div class="card text-white bg-dark mb-3">
                    <div class="card-body">
                        <div class="row">
                            <div class="col">
                                <a href="{% url 'test-automation:detail-search' test.id %}"
class="card-title">
                                    Search Test {{ test.id }}</a>
                            </div>
                            <!-- Edit Test -->
                            <div class="col-10">
                                <form action="{% url 'test-automation:edit-search' test.id %}"
                                      style="display: inline;">
                                    {% csrf_token %}
                                    <input type="hidden" name="test_id" value="{{ test.id }}"/>
                                    <button type="submit" class="btn btn-info btn-sm">Edit</button>
                                </form>
                                <!-- Delete Album -->
                                <form action="{% url 'test-automation:delete-search' test.id %}"
method="post"
                                      style="display: inline;">
                                    {% csrf_token %}
                                    <input type="hidden" name="test_id" value="{{ test.id }}"/>
                                    <button type="submit" class="btn btn-danger btn-
sm">Delete</button>
                                </form>
                                <!-- Run Test -->
                                <form action="{% url 'test-automation:run-search' test.id %}"
method="post"
                                      style="display: inline;">
                                    {% csrf_token %}
                                    <input type="hidden" name="test_id" value="{{ test.id }}"/>
                                    <button type="submit" class="btn btn-success btn-
sm">Run</button>
                                </form>
```

```
                        </div>
                    </div>
                </div>
            </div>
            {% elif test.modelType == 40 %}
            <div class="col">
                <div class="card text-white bg-dark mb-3">
                    <div class="card-body">
                        <div class="row">
                            <div class="col">
                                <a href="{% url 'test-automation:detail-login' test.id %}"
class="card-title">
                                    Login Test {{ test.id }}</a>
                            </div>
                            <!-- Edit Test -->
                            <div class="col-10">
                                <form action="{% url 'test-automation:edit-login' test.id %}"
method="post"
                                      style="display: inline;">
                                    {% csrf_token %}
                                    <input type="hidden" name="test_id" value="{{ test.id }}"/>
                                    <button type="submit" class="btn btn-info btn-sm">Edit</button>
                                </form>
                                <!-- Delete Album -->
                                <form action="{% url 'test-automation:delete-login' test.id %}"
method="post"
                                      style="display: inline;">
                                    {% csrf_token %}
                                    <input type="hidden" name="test_id" value="{{ test.id }}"/>
                                    <button type="submit" class="btn btn-danger btn-
sm">Delete</button>
                                </form>
                                <!-- Run Test -->
                                <form action="{% url 'test-automation:run-login' test.id %}"
method="post"
                                      style="display: inline;">
                                    {% csrf_token %}
                                    <input type="hidden" name="test_id" value="{{ test.id }}"/>
                                    <button type="submit" class="btn btn-success btn-
sm">Run</button>
                                </form>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
            {% endif %}
            {% endfor %}
        </div>
        <div class="col-6">
            <div class="card text-white bg-dark mb-3">
                <div class="card-body"><h2>Run Multiple Tests</h2>
                    <form action="{% url 'test-automation:run-multiple' %}" method="post"
style="display: inline;">
                        {% csrf_token %}
                        {% for test in allTests %}
                        <div class="entry input-group col-xs-3" id="input">
                            <select name="tests">
                                <option value=""></option>
                                {% for test in allTests %}
                                {% if test.modelType == 20 %}
                                <option value="{{ test.modelType }}/{{ test.id }}">Page Test {{
test.id }}</option>
                                {% elif test.modelType == 30 %}
```

```
                                <option value="{{ test.modelType }}/{{ test.id }}">Search Test {{
test.id }}</option>
                                {% elif test.modelType == 40 %}
                                <option value="{{ test.modelType }}/{{ test.id }}">Login Test {{
test.id }}</option>
                                {% endif %}
                                {% endfor %}
                            </select>
                        </div>
                        <br>
                        {% endfor %}
                        <div class="form-group">
                            <input type="hidden" name="teststorun" value="tests"/>
                            <button type="submit" class="btn btn-success">Run Tests</button>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
    {% else %}
    <br><br><br><br><br><br><br>
    <center><h2>There are no tests configurated yet. Please, create a new test now!</h2></center>
    {% endif %}
</div>
{% endblock %}
```

traffictest_form.html

```
{% extends 'test-automation/base-template.html' %}
{% block title %}Create a new Traffic Test{% endblock %}
{% block new %}active{% endblock %}
{% block body %}
<br>
<div class="container-fluid">
    <div class="row">
        <div class="col col">
            <div class="card text-white bg-dark mb-3">
                <div class="card-body">
                    <h3 class="card-title">Create a new Test</h3>
                    {% if error_message %}
                    <p><strong>{{ error_message }}</strong></p>
                    {% endif %}
                    <form class="form-horizontal" role="form" action="" method="post"
enctype="multipart/form-data">
                        {% csrf_token %}
                        <div class="col-sm-10"><label class="control-label col-sm-4"
                                                      for="{{ form.Interval.id_for_label
}}">Interval:</label>
                            {{ form.Interval }}
                        </div>
                        <div class="col-sm-10"><label class="control-label col-sm-4"
                                                      for="{{ form.CPU_Affinity.id_for_label
}}">CPU Affinity:</label>
                            {{ form.CPU_Affinity }}
                        </div>
                        <div class="col-sm-10"><label class="control-label col-sm-4"
                                                      for="{{ form.Use_SCTP.id_for_label }}">Use
SCTP:</label>
                            {{ form.Use_SCTP }}
                        </div>
                        <div class="col-sm-10"><label class="control-label col-sm-4"
                                                      for="{{ form.Use_UDP.id_for_label }}">Use
UDP:</label>
                            {{ form.Use_UDP }}
```

```
                        <div class="col-sm-10"><label class="control-label col-sm-4"
                                                      for="{{ form.Length.id_for_label
}}">Length:</label>
                            {{ form.Length }}
                        </div>
                        <div class="col-sm-10"><label class="control-label col-sm-4"
                                                      for="{{ form.Parallel.id_for_label }}">Number
of parallel client
                            streams:</label>
                            {{ form.Parallel }}
                        </div>
                        <div class="col-sm-10"><label class="control-label col-sm-4"
                                                      for="{{ form.Window.id_for_label
}}">Window:</label>
                            {{ form.Window }}
                        </div>
                        <div class="col-sm-10"><label class="control-label col-sm-4"
                                                      for="{{ form.MSS.id_for_label
}}">MSS:</label>
                            {{ form.MSS }}
                        </div>
                        <div class="col-sm-10"><label class="control-label col-sm-4"
                                                      for="{{ form.IPv4.id_for_label
}}">IPv4:</label>
                            {{ form.IPv4 }}
                        </div>
                        <div class="col-sm-10"><label class="control-label col-sm-4"
                                                      for="{{ form.IPv6.id_for_label
}}">IPv6:</label>
                            {{ form.IPv6 }}
                        </div>
                        <div class="col-sm-10"><label class="control-label col-sm-4"
                                                      for="{{ form.IPToS.id_for_label }}">IP Type
of Service:</label>
                            {{ form.IPToS }}
                        </div>
                        <div class="col-sm-10"><label class="control-label col-sm-4"
                                                      for="{{ form.DSCP.id_for_label }}">IP DSCP
bits:</label>
                            {{ form.DSCP }}
                        </div>
                        <div class="col-sm-10"><label class="control-label col-sm-4"
                                                      for="{{ form.FlowLabel.id_for_label }}">IPv6
flow label:</label>
                            {{ form.FlowLabel }}
                        </div>
                        <div class="col-sm-10"><label class="control-label col-sm-4"
                                                      for="{{ form.SCTP_streams.id_for_label
}}">Number of SCTP
                            streams:</label>
                            {{ form.SCTP_streams }}
                        </div>
                        <div class="form-group">
                            <div class="col-sm-offset-2 col-sm-10">
                                <button type="submit" class="btn btn-success">Submit</button>
                            </div>
                        </div>
                    </form>
                </div>
            </div>
        </div>
        <div class="col col">
            <div class="card text-white bg-dark mb-3">
                <div class="card-body">
                    <h3 class="card-title">Do you want to try our testing framework? Go ahead!</h3>
```

```
                    <p class="card-text">Create a new Test by selecting the type of the test you
want to execute and
                        then fill in the form with the necessary information</p>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

pagetest_form.html

```
{% extends 'test-automation/base-template.html' %}
{% block title %}Create a new Page Test{% endblock %}
{% block new %}active{% endblock %}
{% block body %}
<br>
<div class="container-fluid">
    <div class="row">
        <div class="col col">
            <div class="card text-white bg-dark mb-3">
                <div class="card-body">
                    <h3 class="card-title">Create a new Test</h3>
                    {% if error_message %}
                    <p><strong>{{ error_message }}</strong></p>
                    {% endif %}
                    <form class="form-horizontal" role="form" action="" method="post"
enctype="multipart/form-data">
                        {% csrf_token %}
                        <div class="col">Reference Page
                            <input type="checkbox" name="referencepage" value="True"/>
                        </div>
                        <br>
                        <div class="col-sm-5"><label class="control-label col-sm-3"
                                                     for="{{ form.URL.id_for_label
}}">URL:</label>
                            {{ form.URL }}
                        </div>
                        <div class="form-group">
                            <div class="col-sm-offset-2 col-sm-10">
                                <button type="submit" class="btn btn-success">Submit</button>
                            </div>
                        </div>
                    </form>
                </div>
            </div>
        </div>
        <div class="col col">
            <div class="card text-white bg-dark mb-3">
                <div class="card-body">
                    <h3 class="card-title">Do you want to try our testing framework? Go ahead!</h3>
                    <p class="card-text">Create a new Test by selecting the type of the test you
want to execute and
                        then fill in the form with the necessary information</p>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

searchtest_form.html

```
{% extends 'test-automation/base-template.html' %}
{% block title %}Create a new Search Test{% endblock %}
{% block new %}active{% endblock %}
{% block body %}
<br>
<div class="container-fluid">
    <div class="row">
        <div class="col col">
            <div class="card text-white bg-dark mb-3">
                <div class="card-body">
                    <h3 class="card-title">Create a new Test</h3>
                    {% if error_message %}
                    <p><strong>{{ error_message }}</strong></p>
                    {% endif %}
                    <form class="form-horizontal" role="form" action="" method="post"
enctype="multipart/form-data">
                        {% csrf_token %}
                        <div class="col-sm-7"><label class="control-label col-sm-4"
                                                     for="{{ form.Term_to_search.id_for_label
}}">Term to search:</label>
                            {{ form.Term_to_search }}
                        </div>
                        <div class="form-group">
                            <div class="col-sm-offset-2 col-sm-10">
                                <button type="submit" class="btn btn-success">Submit</button>
                            </div>
                        </div>
                    </form>
                </div>
            </div>
        </div>
        <div class="col col">
            <div class="card text-white bg-dark mb-3">
                <div class="card-body">
                    <h3 class="card-title">Do you want to try our testing framework? Go ahead!</h3>
                    <p class="card-text">Create a new Test by selecting the type of the test you
want to execute and
                        then fill in the form with the necessary information</p>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

logintest_form.html

```
{% extends 'test-automation/base-template.html' %}
{% block title %}Create a new Login Test{% endblock %}
{% block new %}active{% endblock %}
{% block body %}
<br>
<div class="container-fluid">
    <div class="row">
        <div class="col col">
            <div class="card text-white bg-dark mb-3">
                <div class="card-body">
                    <h3 class="card-title">Create a new Test</h3>
                    {% if error_message %}
                    <p><strong>{{ error_message }}</strong></p>
                    {% endif %}
                    <form class="form-horizontal" role="form" action="" method="post"
enctype="multipart/form-data">
                        {% csrf_token %}
                        <div class="col-sm-6"><label class="control-label col-sm-3"
                                                     for="{{ form.Email.id_for_label
}}">Email:</label>
                            {{ form.Email }}
                        </div>
                        <div class="col-sm-6"><label class="control-label col-sm-3"
                                                     for="{{ form.Password.id_for_label
}}">Password:</label>
                            {{ form.Password }}
                        </div>
                        <div class="form-group">
                            <div class="col-sm-offset-2 col-sm-10">
                                <button type="submit" class="btn btn-success">Submit</button>
                            </div>
                        </div>
                    </form>
                </div>
            </div>
        </div>
        <div class="col col">
            <div class="card text-white bg-dark mb-3">
                <div class="card-body">
                    <h3 class="card-title">Do you want to try our testing framework? Go ahead!</h3>
                    <p class="card-text">Create a new Test by selecting the type of the test you
want to execute and
                        then fill in the form with the necessary information</p>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

detail-traffic.html

```
{% extends 'test-automation/base-template.html' %}
{% block title %}Traffic Test {{ test.id }}{% endblock %}
{% block tests %}active{% endblock %}
{% block body %}
<br>
<div class="container-fluid">
    <div class="row">
        <!-- Test Parameters -->
        <div class="col-sm-4 col-md-12">
            <div class="card text-white bg-dark mb-3">
                <div class="card-body">
```

```
                    <h2 class="card-title">Traffic Test {{ test.id }}</h2>
                    Interval: {{ test.Interval }}<br>
                    CPU Affinity: {{ test.CPU_Affinity }}<br>
                    Use SCTP: {{ test.Use_SCTP }}<br>
                    Use UDP: {{ test.Use_UDP }}<br>
                    Bitrate: {{ test.Bandwidth }}<br>
                    Bytes: {{ test.Bytes }}<br>
                    Length: {{ test.Length }}<br>
                    Number of parallel client streams: {{ test.Parallel }}<br>
                    Window: {{ test.Window }}<br>
                    MSS: {{ test.MSS }}<br>
                    IPv4: {{ test.IPv4 }}<br>
                    IPv6: {{ test.IPv6 }}<br>
                    IP Type of Service: {{ test.IPToS }}<br>
                    IP DSCP bits: {{ test.DSCP }}<br>
                    IPv6 flow label: {{ test.FlowLabel }}<br>
                    Number of SCTP streams: {{ test.SCTP_streams }}<br><br>
                    <!-- Back -->
                    <form action="{% url 'test-automation:test-editor' %}" method="post"
                          style="display: inline;">
                        {% csrf_token %}
                        <button type="submit" class="btn btn-default btn-sm"> Back</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

detail-page.html

```
{% extends 'test-automation/base-template.html' %}
{% block title %}Page Test {{ test.id }}{% endblock %}
{% block tests %}active{% endblock %}
{% block body %}
<br>
<div class="container-fluid">
    <div class="row">
        <!-- Test Parameters -->
        <div class="col-sm-4 col-md-12">
            <div class="card text-white bg-dark mb-3">
                <div class="card-body">
                    <h2 class="card-title">Page Test {{ test.id }}</h2>
                    Page URL: {{ test.URL }}<br><br>
                    <!-- Back -->
                    <form action="{% url 'test-automation:test-editor' %}" method="post"
                          style="display: inline;">
                        {% csrf_token %}
                        <button type="submit" class="btn btn-default btn-sm"> Back</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

detail-search.html

```
{% extends 'test-automation/base-template.html' %}
{% block title %}Search Test {{ test.id }}{% endblock %}
{% block tests %}active{% endblock %}
{% block body %}
<br>
<div class="container-fluid">
    <div class="row">
        <!-- Test Parameters -->
        <div class="col-sm-4 col-md-12">
            <div class="card text-white bg-dark mb-3">
                <div class="card-body">
                    <h2 class="card-title">Search Test {{ test.id }}</h2>
                    Search Term: {{ test.Term_to_search }}<br><br>
                    <!-- Back -->
                    <form action="{% url 'test-automation:test-editor' %}" method="post"
                          style="display: inline;">
                        {% csrf_token %}
                        <button type="submit" class="btn btn-default btn-sm"> Back</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

detail-login.html

```
{% extends 'test-automation/base-template.html' %}
{% block title %}Login Test {{ test.id }}{% endblock %}
{% block tests %}active{% endblock %}
{% block body %}
<br>
<div class="container-fluid">
    <div class="row">
        <!-- Test Parameters -->
        <div class="col-sm-4 col-md-12">
            <div class="card text-white bg-dark mb-3">
                <div class="card-body">
                    <h2 class="card-title">Login Test {{ test.id }}</h2>
                    Email: {{ test.Email }}<br>
                    Password is hidden<br><br>
                    <!-- Back -->
                    <form action="{% url 'test-automation:test-editor' %}" method="post"
                          style="display: inline;">
                        {% csrf_token %}
                        <button type="submit" class="btn btn-default btn-sm"> Back</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

logs.html

```
{% extends 'test-automation/base-template.html' %}
{% block logs %}active{% endblock %}
<title>{% block title %}T A F - Test Logs{% endblock %}</title>
{% block body %}
<br>
<div class="container-fluid">
    <!-- Results -->
    {% if allResults %}
    <div class="row">
        <div class="col">
            <h2>All Results</h2>
            <h3>{{ user.username }}'s Tests</h3>
            {% for result in allResults %}
            {% if result.modelType == 11 %}
            <div class="col">
                <div class="card text-white bg-dark mb-3">
                    <div class="card-body">
                        <div class="row">
                            <div class="col">
                                <a href="{% url 'test-automation:logs-traffic' result.id %}"
class="card-title">
                                    Result {{ result.id }} for Traffic Test {{ result.resultKey.id
}}</a>
                            </div>
                            <!-- Delete Result -->
                            <div class="col-10">
                                <form action="{% url 'test-automation:delete-traffic-res' result.id
%}" method="post"
                                      style="display: inline;">
                                    {% csrf_token %}
                                    <input type="hidden" name="result_id" value="{{ result.id }}"/>
                                    <button type="submit" class="btn btn-danger btn-sm">
Delete</button>
                                </form>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
            {% elif result.modelType == 21 %}
            <div class="col">
                <div class="card text-white bg-dark mb-3">
                    <div class="card-body">
                        <div class="row">
                            <div class="col">
                                <a href="{% url 'test-automation:logs-page' result.id %}"
class="card-title">
                                    Result {{ result.id }} for Page Test {{ result.resultKey.id
}}</a>
                            </div>
                            <!-- Delete Result -->
                            <div class="col-10">
                                <form action="{% url 'test-automation:delete-page-res' result.id
%}" method="post"
                                      style="display: inline;">
                                    {% csrf_token %}
                                    <input type="hidden" name="result_id" value="{{ result.id }}"/>
                                    <button type="submit" class="btn btn-danger btn-sm">
Delete</button>
                                </form>
                            </div>
                        </div>
```

```
                    </div>
                </div>
            </div>
            {% elif result.modelType == 31 %}
            <div class="col">
                <div class="card text-white bg-dark mb-3">
                    <div class="card-body">
                        <div class="row">
                            <div class="col">
                                <a href="{% url 'test-automation:logs-search' result.id %}"
class="card-title">
                                    Result {{ result.id }} for Search Test {{ result.resultKey.id
}}</a>
                            </div>
                            <!-- Delete Result -->
                            <div class="col-10">
                                <form action="{% url 'test-automation:delete-search-res' result.id
%}" method="post"
                                      style="display: inline;">
                                    {% csrf_token %}
                                    <input type="hidden" name="result_id" value="{{ result.id }}"/>
                                    <button type="submit" class="btn btn-danger btn-sm">
Delete</button>
                                </form>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
            {% elif result.modelType == 41 %}
            <div class="col">
                <div class="card text-white bg-dark mb-3">
                    <div class="card-body">
                        <div class="row">
                            <div class="col">
                                <a href="{% url 'test-automation:logs-login' result.id %}"
class="card-title">
                                    Result {{ result.id }} for Login Test {{ result.resultKey.id
}}</a>
                            </div>
                            <!-- Delete Result -->
                            <div class="col-10">
                                <form action="{% url 'test-automation:delete-login-res' result.id
%}" method="post"
                                      style="display: inline;">
                                    {% csrf_token %}
                                    <input type="hidden" name="result_id" value="{{ result.id }}"/>
                                    <button type="submit" class="btn btn-danger btn-sm">
Delete</button>
                                </form>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
            {% endif %}
            {% endfor %}
        </div>
    </div>
    {% else %}
    <br><br><br><br><br><br><br>
    <center><h2>There are no tests executed yet. Please, create a new test now!</h2>
    </center>
    {% endif %}
</div>
{% endblock %}
```

logs-traffic.html

```
{% extends 'test-automation/base-template.html' %}
{% block title %}Result {{ result.id }} for Traffic Test {{ result.resultKey.id }} - Results{%
endblock %}
{% block logs %}active{% endblock %}
{% block body %}
<br>
<div class="container-fluid">
    <div class="row">
        <!-- Test Parameters -->
        <div class="col-sm-4 col-md-12">
            <div class="card text-white bg-dark mb-3">
                <div class="card-body">
                    <h2 class="card-title">Result {{ result.id }} for Traffic Test {{
result.resultKey.id }}</h2>
                    Report: <a
                            href="/media/TrafficReports/{{ result.report }}"
                            download="Report{{ result.id }}-Test{{ result.resultKey.id }}">
Report_{{ result.id }}.txt</a><br>
                    <!-- Back -->
                    <form action="{% url 'test-automation:logs' %}" method="post"
                          style="display: inline;">
                        {% csrf_token %}
                        <button type="submit" class="btn btn-default btn-sm"> Back</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

logs-page.html

```
{% extends 'test-automation/base-template.html' %}
{% block title %}Result {{ result.id }} for Page Test {{ result.resultKey.id }} - Result: {{
result.status }}{% endblock %}
{% block logs %}active{% endblock %}
{% block body %}
<br>
<div class="container-fluid">
    <div class="row">
        <!-- Test Parameters -->
        <div class="col-sm-4 col-md-12">
            <div class="card text-white bg-dark mb-3">
                <div class="card-body">
                    <h2 class="card-title">Result {{ result.id }} for Page Test {{
result.resultKey.id }}</h2>
                    Status: {{ result.status }}<br>
                    Report: <a
                            href="/media/Reports/{{ result.report }}"
                            download="Report{{ result.id }}-Test{{ result.resultKey.id }}">
Report_{{ result.id }}.html</a><br>
                    Capture File: <a
                            href="/media/Captures/{{ result.capture }}"
                            download="Capture{{ result.id }}-Test{{ result.resultKey.id
}}">capture.pcap</a><br>
                    Link to Kibana: <a href="{{ result.kibana }}">Link to Kibana
Visualizations</a><br>
                    <!-- Back -->
                    <form action="{% url 'test-automation:logs' %}" method="post"
                          style="display: inline;">
                        {% csrf_token %}
```

```
                    <button type="submit" class="btn btn-default btn-sm"> Back</button>
                </form>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

search-logs.html

```
{% extends 'test-automation/base-template.html' %}
{% block title %}Result {{ result.id }} for Search Test {{ result.resultKey.id }} - Result: {{
result.status }}{% endblock %}
{% block logs %}active{% endblock %}
{% block body %}
<br>
<div class="container-fluid">
    <div class="row">
        <!-- Test Parameters -->
        <div class="col-sm-4 col-md-12">
            <div class="card text-white bg-dark mb-3">
                <div class="card-body">
                    <h2 class="card-title">Result {{ result.id }} for Search Test {{
result.resultKey.id }}</h2>
                    Status: {{ result.status }}<br>
                    Report: <a
                            href="/media/Reports/{{ result.report }}"
                            download="Report{{ result.id }}-Test{{ result.resultKey.id }}">
Report_{{ result.id }}.html</a><br>
                    Capture File: <a
                            href="/media/Captures/{{ result.capture }}"
                            download="Capture{{ result.id }}-Test{{ result.resultKey.id
}}">capture.pcap</a><br>
                    Link to Kibana: <a href="{{ result.kibana }}">Link to Kibana
Visualizations</a><br>
                    <!-- Back -->
                    <form action="{% url 'test-automation:logs' %}" method="post"
                          style="display: inline;">
                        {% csrf_token %}
                        <button type="submit" class="btn btn-default btn-sm"> Back</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

logs-login.html

```
{% extends 'test-automation/base-template.html' %}
{% block title %}Result {{ result.id }} for Login Test {{ result.resultKey.id }} - Result: {{
result.status }}{% endblock %}
{% block logs %}active{% endblock %}
{% block body %}
<br>
<div class="container-fluid">
    <div class="row">
        <!-- Test Parameters -->
        <div class="col-sm-4 col-md-12">
            <div class="card text-white bg-dark mb-3">
```

```
                <div class="card-body">
                    <h2 class="card-title">Result {{ result.id }} for Login Test {{
result.resultKey.id }}</h2>
                    Status: {{ result.status }}<br>
                    Report: <a
                            href="/media/Reports/{{ result.report }}"
                            download="Report{{ result.id }}-Test{{ result.resultKey.id }}">
Report_{{ result.id }}.html</a><br>
                    Capture File: <a
                            href="/media/Captures/{{ result.capture }}"
                            download="Capture{{ result.id }}-Test{{ result.resultKey.id
}}">capture.pcap</a><br>
                    Link to Kibana: <a href="{{ result.kibana }}">Link to Kibana
Visualizations</a><br>
                    <!-- Back -->
                    <form action="{% url 'test-automation:logs' %}" method="post"
                          style="display: inline;">
                        {% csrf_token %}
                        <button type="submit" class="btn btn-default btn-sm"> Back</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

-*Static Files*

stylesheet.css

```
body {
    background-image: url("images/dark.jpg");
    background-repeat: no-repeat;
    background-size: cover;
    color: white;
}

.navbar-brand{
    font-family: 'Work Sans', sans-serif;
}

.carousel-inner > .item > img {
  width:640px;
  height:360px;
}
```

-*Tests*

test_class_pageHTTP.py

```
import unittest
import sys
import HtmlTestRunner
from selenium import webdriver
from time import sleep
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities


class PageHTTP(unittest.TestCase):
    url = 'www.example.com'

    @classmethod
    def setUpClass(cls):
        capabilities = DesiredCapabilities.FIREFOX.copy()
        capabilities['browserName'] = "firefox"
        capabilities['platform'] = "LINUX"
        cls.driver = webdriver.Remote(
        command_executor='http://147.102.40.58:4444/wd/hub',
        desired_capabilities=capabilities)
        cls.driver.implicitly_wait(10)
        cls.driver.set_page_load_timeout(20)

    def test_pageHTTP(self):
        self.driver.get("http://" + self.url)
        sleep(5)

    @classmethod
    def tearDownClass(cls):
        cls.driver.quit()


if __name__ == '__main__':
    if len(sys.argv) > 0:
        PageHTTP.url = sys.argv.pop()
    unittest.main(testRunner=HtmlTestRunner.HTMLTestRunner(output="/home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Reports/"))
```

test_class_searchGoogle.py

```
import unittest
import sys
import HtmlTestRunner
from selenium import webdriver
from time import sleep
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities


class SearchGoogle(unittest.TestCase):
    term = 'example'

    @classmethod
    def setUpClass(cls):
        capabilities = DesiredCapabilities.FIREFOX.copy()
        capabilities['browserName'] = "firefox"
        capabilities['platform'] = "LINUX"
        cls.driver = webdriver.Remote(
        command_executor='http://147.102.40.58:4444/wd/hub',
        desired_capabilities=capabilities)
        cls.driver.implicitly_wait(10)
        cls.driver.set_page_load_timeout(20)
```

```
    def test_googleSearch(self):
        self.driver.get("http://www.google.com")
        self.driver.find_element_by_name("q").send_keys(self.term)
        self.driver.find_element_by_name("q").submit()
        sleep(5)

    @classmethod
    def tearDownClass(cls):
        cls.driver.quit()


if __name__ == '__main__':
    if len(sys.argv) > 0:
        SearchGoogle.term = sys.argv.pop()
    unittest.main(testRunner=HtmlTestRunner.HTMLTestRunner(output="/home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Reports/"))
```

test_class_loginFacebook.py

```
import unittest
import sys
import HtmlTestRunner
from selenium import webdriver
from time import sleep
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities


class LoginFacebook(unittest.TestCase):
    email = 'email'
    password = 'password'

    @classmethod
    def setUpClass(cls):
        capabilities = DesiredCapabilities.FIREFOX.copy()
        capabilities['browserName'] = "firefox"
        capabilities['platform'] = "LINUX"
        cls.driver = webdriver.Remote(
        command_executor='http://147.102.40.58:4444/wd/hub',
        desired_capabilities=capabilities)
        cls.driver.implicitly_wait(10)
        cls.driver.set_page_load_timeout(20)

    def test_facebookLoginLogout(self):
        self.driver.get("https://www.facebook.com")
        self.driver.find_element_by_name("email").clear()
        self.driver.find_element_by_name("email").send_keys(self.email)
        sleep(1)
        self.driver.find_element_by_name("pass").clear()
        self.driver.find_element_by_name("pass").send_keys(self.password)
        sleep(1)
        self.driver.find_element_by_id("loginbutton").click()
        sleep(5)

    @classmethod
    def tearDownClass(cls):
        cls.driver.quit()


if __name__ == '__main__':
    if len(sys.argv) > 1:
        LoginFacebook.password = sys.argv.pop()
        LoginFacebook.email = sys.argv.pop()
    unittest.main(testRunner=HtmlTestRunner.HTMLTestRunner(output="/home/kostas/gitlab-
repository/testautomation/website/testAutomation/media/Reports/"))
```