



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ
ΠΜΣ "ΠΛΗΡΟΦΟΡΙΚΗ"

Ανάπτυξη Εφαρμογής για παραμετροποίηση δικτύου με το Django Framework

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΤΟΥ

Ιάσωνας Σιμώτας

Επιβλέπων: Δουληγέρης Χρήστος
Καθηγητής ΠΑΠΕΙ

Αθήνα, Μήνας Έτος



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
UNIVERSITY OF PIRAEUS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ
ΠΜΣ "ΠΛΗΡΟΦΟΡΙΚΗ"
ΤΟΜΕΑΣ

Ανάπτυξη Εφαρμογής για παραμετροποίηση δικτύου με το Django Framework

ΔΙΠΛΩΜΑΤΙΚΗ

ΤΟΥ

Ιάσωνας Σιμωτας

Επιβλέπων: Δουληγέρης Χρήστος
Καθηγητής ΠΑΠΕΙ

Εγκρίθηκε από την κάτωθι τριμελή επιτροπή την 1^η Ιανουαρίου 2024.

Όνομα Επώνυμο
Καθηγητής

Όνομα Επώνυμο
Καθηγητής

Όνομα Επώνυμο
Αναπληρωτής Καθηγητής

Ιάσωνας Σιμώτας

Πτυχιούχος Μεταπτυχιακού ΠΜΣ Πληροφορικής

Copyright © Όνομα Επώνυμο, 2023

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ' ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Πανεπιστημίου Πειραιά.

Κεφάλαιο 1

Πρόλογος

1.1 Σκοπός και στόχοι της διπλωματικής εργασίας

Η πτυχιακή αυτή εργασία έχει ως σκοπό την ανάπτυξη μιας σύγχρονης εφαρμογής για την παραμετροποίηση δικτυακών συσκευών, αξιοποιώντας τις δυνατότητες που προσφέρουν τα σύγχρονα τεχνολογικά εργαλεία και πρότυπα. Η επιλογή του συγκεκριμένου θέματος βασίζεται στη διαρκώς αυξανόμενη ανάγκη για αυτοματοποίηση και ευελιξία στη διαχείριση δικτύων, ιδιαίτερα σε περιβάλλοντα που χαρακτηρίζονται από μεγάλη κλίμακα και πολυπλοκότητα.

Οι βασικοί στόχοι της εργασίας περιλαμβάνουν:

- Τη δημιουργία μιας φιλικής προς τον χρήστη εφαρμογής που θα απλοποιεί τη διαδικασία παραμετροποίησης δικτυακών συσκευών.
- Τη χρήση σύγχρονων τεχνολογιών, όπως τα *microservices* και τα *containers*, για την εξασφάλιση κλιμακωσιμότητας και φορητότητας.
- Την ενσωμάτωση εργαλείων αυτοματοποίησης και τη βελτιστοποίηση των διαδικασιών διαχείρισης.
- Την αξιολόγηση της εφαρμογής σε προσομοιωμένα περιβάλλοντα για την επιβεβαίωση της λειτουργικότητας και της απόδοσής της.

Η εργασία στοχεύει να προσφέρει μια ολοκληρωμένη λύση που θα συνδυάζει καινοτομία, πρακτικότητα και δυνατότητες για μελλοντική επέκταση.

1.2 Συνοπτική Περιγραφή της Εφαρμογής και της Υλοποίησης

Η εφαρμογή που αναπτύχθηκε βασίζεται στο Django framework της γλώσσας προγραμματισμού Python, παρέχοντας ένα ολοκληρωμένο backend σύστημα για τη διαχείριση και παραμετροποίηση δικτυακών συσκευών.

Για την υλοποίηση και τη δοκιμή της εφαρμογής:

- Χρησιμοποιήθηκαν εργαλεία όπως το Docker για την ανάπτυξη και τη διαχείριση των containers, εξασφαλίζοντας φορητότητα και σταθερότητα.
- Δοκιμάστηκε σε περιβάλλον GNS3, όπου προσομοιώθηκαν διάφορες συνθήκες δικτύου για την επιβεβαίωση της λειτουργικότητας της εφαρμογής.
- Εφαρμόστηκαν τεχνολογίες όπως τα RESTful APIs για τη διασύνδεση με τις δικτυακές συσκευές, ενώ η αρχιτεκτονική της εφαρμογής βασίζεται σε μικρο-υπηρεσίες για μεγαλύτερη ευελιξία και επεκτασιμότητα.

Η διαδικασία ανάπτυξης περιλάμβανε τη σχεδίαση ενός περιβάλλοντος φιλικού προς τον χρήστη για την εισαγωγή και επεξεργασία ρυθμίσεων, την ενσωμάτωση εργαλείων για αυτοματοποιημένη εκτέλεση εντολών και την αξιοποίηση τεχνολογιών containerization για την εύκολη ανάπτυξη της εφαρμογής σε διαφορετικά περιβάλλοντα. Η εφαρμογή φιλοδοξεί να αποτελέσει ένα χρήσιμο εργαλείο για προγραμματιστές που ενδιαφέρονται να αναπτύξουν παρόμοιες εφαρμογές καθώς και σαν γενικότερη συμβολή στο χώρο της αυτοματοποίησης και του προγραμματισμού.

Κεφάλαιο 2

Αναγνώριση και Ευχαριστίες

Η ολοκλήρωση της παρούσας διπλωματικής εργασίας αποτελεί έναν σταθμό που δεν θα ήταν εφικτός χωρίς την υποστήριξη και την ενθάρρυνση ορισμένων ανθρώπων, στους οποίους θα ήθελα να εκφράσω την βαθιά μου ευγνωμοσύνη.

Πρώτα και κύρια, θα ήθελα να ευχαριστήσω την οικογένειά μου, τους φίλους και την κοπέλα μου για τη συνεχή τους υποστήριξη. Η ενθάρρυνση και η καθοδήγησή τους ήταν αστείρευτες πηγές έμπνευσης και δύναμης. Με βοήθησαν να παραμείνω επικεντρωμένος στους στόχους μου, ακόμα και όταν οι δυσκολίες και οι προκλήσεις πολλαπλασιάζονταν. Χάρη σε αυτούς, κατάφερα να διατηρήσω την αισιοδοξία και την αντοχή που απαιτούνταν για να φέρω εις πέρας αυτή την προσπάθεια.

Παρά το γεγονός ότι οι επαγγελματικές μου υποχρεώσεις ήταν συχνά απαιτητικές και πολλές φορές δεν μου άφηναν τον χρόνο που ήθελα για την ενασχόληση με τη διπλωματική μου, κατάφερα να αντλήσω από την εργασιακή μου εμπειρία πολύτιμα εφόδια. Η επαγγελματική μου πορεία ως μηχανικός δικτύωσης και λογισμικού στον τομέα των τηλεπικοινωνιών με βοήθησε να κατανοήσω καλύτερα τις έννοιες και τις τεχνολογίες που μελετήθηκαν. Επιπλέον, αυτή η εμπειρία αποτέλεσε σημαντικό εργαλείο για τη σύνθεση, την ανάλυση και την εμβάθυνση στις τεχνικές πτυχές του έργου.

Η χρονιά που ξεκίνησα το μεταπτυχιακό μου πρόγραμμα, το 2021, συνέπεσε με μια από τις πιο γόνιμες περιόδους της ακαδημαϊκής και επαγγελματικής μου ζωής. Με δύο χρόνια εμπειρίας στον τομέα της μηχανικής δικτύων, είχα ήδη τη βάση για να διευρύνω τις γνώσεις μου και να εξελίξω την αντίληψή μου γύρω από τις τεχνολογικές εξελίξεις στις τηλεπικοινωνίες. Η αγάπη μου για τον κλάδο αυτό αποτέλεσε το κύριο κίνητρο για την απόφαση να συνεχίσω τις σπουδές μου και να ασχοληθώ με την παρούσα εργασία.

Μέσα από τη διαδικασία συγγραφής της διπλωματικής, απέκτησα όχι μόνο γνώσεις σε θεωρητικό επίπεδο αλλά και πρακτικές δεξιότητες που εμπλούτισαν την επαγγελματική μου ταυτότητα. Η εμπειρία αυτή συνδύασε την τεχνική μου κατάρτιση με τη θεωρητική ανάλυση, επιτρέποντάς μου να αναπτύξω την ικανότητα να αντιμετωπίζω περίπλοκα προβλήματα με δημιουργική και κριτική σκέψη.

Αναγνωρίζω ότι οι στιγμές αβεβαιότητας και πίεσης, που πολλές φορές συνδυάζο-

νταν με τις επαγγελματικές απαιτήσεις, υπήρξαν ιδιαίτερα δύσκολες. Ωστόσο, αυτές οι προκλήσεις με δίδαξαν την αξία της αποτελεσματικής διαχείρισης χρόνου και της υπομονής. Μέσα από αυτές τις δυσκολίες, έμαθα να προτεραιοποιώ τις υποχρεώσεις μου και να εργάζομαι με συνέπεια.

Δεν μπορώ να παραλείψω την πολύτιμη συμβολή των συναδέλφων και των φίλων μου. Η κατανόηση και η υποστήριξή τους υπήρξαν ανεκτίμητες. Η υπομονή και η ενθάρρυνσή τους, ειδικά σε περιόδους έντονης πίεσης, μου έδωσαν τη δυνατότητα να διατηρήσω την ισορροπία μου και να ολοκληρώσω αυτό το έργο με επιτυχία. Μέσα από αυτή την εμπειρία, συνειδητοποίησα τη σημασία της συνεργασίας και της αλληλοϋποστήριξης, για τις οποίες τους ευχαριστώ από καρδιάς.

Η εργασία αυτή δεν είναι απλώς μια ακαδημαϊκή ολοκλήρωση αλλά ένα προσωπικό και επαγγελματικό επίτευγμα που ελπίζω να αποτελέσει εφαλτήριο για περαιτέρω ανάπτυξη και συνεισφορά στον χώρο της τεχνολογίας και των τηλεπικοινωνιών.

Περιεχόμενα

1	Πρόλογος	5
1.1	Σκοπός και στόχοι της διπλωματικής εργασίας	5
1.2	Συνοπτική Περιγραφή της Εφαρμογής και της Υλοποίησης	5
2	Αναγνώριση και Ευχαριστίες	7
3	Εισαγωγή	13
3.1	Στόχοι του έργου	13
3.2	Περιγραφή προβλήματος και λύσης	13
3.3	Τεχνολογίες που χρησιμοποιήθηκαν και γιατί	14
4	Θεωρητικό Υπόβαθρο	17
4.1	Τεχνολογίες του software development	17
4.1.1	Django Framework,Paramiko, Netmiko και Napalm	17
4.1.2	Paramiko	17
4.1.3	Netmiko	18
4.1.4	Napalm	18
4.1.5	Version Control με Git και GitHub	19
4.1.6	Συνεχής Ενσωμάτωση και Παράδοση (CI/CD)	19
4.2	Τεχνολογίες του Software deployment	19
4.2.1	Containers και Docker	19
4.2.2	Kubernetes και Container Orchestration	20
4.3	Αυτοματοποίηση Διαχείρισης Δικτύου	20
4.3.1	Network Automation	20
4.3.2	Cisco IOS	20
4.3.3	REST APIs για Δικτυακές Συσκευές Cisco	21
4.3.4	GNS3 και Εικονικά Δίκτυα	21
4.4	Πρόγραμμα εικονοποίησης για το GNS3 VM-VirtualBox	22
5	Σχεδίαση και Ανάλυση	25
5.1	Απαιτήσεις Συστήματος	25
5.1.1	Λειτουργικές απαιτήσεις	25
5.1.2	Τοπικό περιβάλλον ανάπτυξης	25
5.2	Η επανάσταση στο web development	26
5.3	Αρχιτεκτονική της εφαρμογής	26

5.3.1	Μοντέλο MVC (Model-View-Controller)	26
5.4	Django MTV	27
5.4.1	Διαγραμματική απεικόνιση της εφαρμογής	27
5.5	Εικονικό Περιβάλλον Δικτύου GNS3 -Testbed	28
5.5.1	Σχεδίαση Τοπολογίας Δικτύου	28
5.5.2	Προσομοίωση Συσκευών Cisco	28
6	Application Demo	31
6.1	Εισαγωγή-Η λογική της λειτουργίας της εφαρμογής Δθανγο	31
6.2	Η αρχική σελίδα της εφαρμογής	32
6.3	Devices	33
6.4	Στατιστικά της συσκευής	35
6.5	Στατιστικά της διεπαφής	35
6.6	Backup της συσκευής	36
6.7	Διαμόρφωση διεύθυνσης IP	37
7	Εισαγωγή τεχνολογίας του Κυβερνήτη(Kubernetes)	39
7.1	Εισαγωγή-Η λογική της λειτουργίας του Κυβερνήτη	39
7.2	Το αποθετήριο κοντεινερ-Docker Desktop	39
7.3	Κυβερνήτης	41
8	Συμπέρασμα	45
8.1	Εισαγωγή	45
9	Βιβλιογραφία	47
9.1	Βιβλιογραφικές αναφορές	47

Κατάλογος Σχημάτων

4.1	Virtualization Γενική αρχιτεκτονική	22
4.2	Virtualization Γενική αρχιτεκτονική	23
4.3	Virtualbox	24
5.1	MTV	27
5.2	Local PC-GNS3VM-CISCO IOS Connection Architecture	28
5.3	Import appliance	29
5.4	filename configuration	29
6.1	Urls.py αρχείο	31
6.2	Django run server	32
6.3	Αρχική Σελίδα	33
6.4	Αρχικοποίηση συσκευής	34
6.5	Controller-Device Interfaces	34
6.6	Interfaces	35
6.7	Στατιστικά	36
6.8	Στατιστικά διεπαφής	37
6.9	Τρέχων Διαμόρφωση	37
6.10	IP Διαμόρφωση	38
6.11	IP Διαφορά	38
7.1	Dockerfile	40
7.2	Docker build-Δημιουργία του κοντεινερ	41
7.3	Docker push-	41
7.4	Docker image list	42
7.5	Minikube deployment	43
7.6	Django Deploy	44

Κεφάλαιο 3

Εισαγωγή

3.1 Στόχοι του έργου

Ο βασικός στόχος αυτής της διπλωματικής εργασίας είναι η ανάπτυξη μιας ολοκληρωμένης εφαρμογής που θα ενσωματώνει τις δυνατότητες της αυτοματοποίησης δικτύων, της ανάπτυξης εφαρμογών Ιστού, και των τεχνολογιών οριζόμενων από λογισμικό δικτύων (SDN). Μέσα από την πρακτική εφαρμογή, επιχειρείται η κατανόηση και αξιολόγηση της λειτουργικότητας εργαλείων όπως το Kubernetes, το Django, και η γλώσσα προγραμματισμού Python. Παράλληλα, διερευνώνται οι πρακτικές δυνατότητες της αυτοματοποίησης στη διαχείριση δικτύων, την παρακολούθηση και τη συντήρηση, καθώς και οι τρόποι με τους οποίους αυτές μπορούν να βελτιώσουν τη λειτουργικότητα και την αποτελεσματικότητα.

3.2 Περιγραφή προβλήματος και λύσης

Η διαχείριση δικτυακών υποδομών έχει γίνει εξαιρετικά περίπλοκη λόγω του μεγέθους και της πολυπλοκότητας των σύγχρονων δικτύων. Η χειροκίνητη διαχείριση αυτών των υποδομών είναι χρονοβόρα και επιρρεπής σε σφάλματα, ενώ δεν μπορεί να ανταποκριθεί επαρκώς στις αυξανόμενες απαιτήσεις για ευελιξία, ταχύτητα και αξιοπιστία. Τα παραδοσιακά μοντέλα διαχείρισης συσκευών απαιτούν εξειδικευμένες γνώσεις, καθιστώντας δύσκολη την προσαρμογή στις ταχέως μεταβαλλόμενες συνθήκες.

Η λύση που προτείνεται στην παρούσα εργασία περιλαμβάνει την υλοποίηση μιας εφαρμογής που αξιοποιεί τεχνολογίες αυτοματοποίησης για να μειώσει την ανθρώπινη παρέμβαση, να εξαλείψει επαναλαμβανόμενες εργασίες και να ενισχύσει τη δυνατότητα λήψης αποφάσεων σε πραγματικό χρόνο. Μέσα από τη χρήση βιβλιοθηκών Python, της πλατφόρμας Django, και της τεχνολογίας Kubernetes, επιτυγχάνεται η κεντριοποιημένη διαχείριση και η δυναμική προσαρμογή των δικτύων σύμφωνα με τις ανάγκες του οργανισμού.

3.3 Τεχνολογίες που χρησιμοποιήθηκαν και γιατί

Python:

Λόγοι επιλογής:

Η ευκολία στη σύνταξη και η πλούσια συλλογή βιβλιοθηκών για δικτυακές εφαρμογές και αυτοματοποίηση καθιστούν τη Python ιδανική για την ανάπτυξη της εφαρμογής. Επιπλέον, η ενσωμάτωσή της με πρωτόκολλα όπως SSH και REST επιτρέπει την αποτελεσματική διαχείριση συσκευών.

Χρήση:

Αυτοματοποιημένες διαδικασίες, ανάπτυξη scripts για παρακολούθηση συσκευών και υλοποίηση API.

Django:

Λόγοι επιλογής:

Το ισχυρό backend περιβάλλον του Django παρέχει ευελιξία, ασφάλεια, και δυνατότητα γρήγορης ανάπτυξης εφαρμογών Ιστού.

Χρήση:

Δημιουργία της κεντρικής διεπαφής διαχείρισης δικτύων, επεξεργασία δεδομένων και παροχή δυναμικών υπηρεσιών στον χρήστη.

Kubernetes:

Λόγοι επιλογής:

Η δυνατότητα κλιμάκωσης και διαχείρισης microservices μέσω containers διευκολύνει τη δυναμική ανάπτυξη και συντήρηση της εφαρμογής.

Χρήση:

Ανάπτυξη και κλιμάκωση microservices που υποστηρίζουν τις λειτουργίες του συστήματος.

Περιβάλλον ανάπτυξης (GNS3 VM, Cisco Images, VirtualBox):

Λόγοι επιλογής:

Εξομοίωση πραγματικών δικτυακών περιβαλλόντων για δοκιμή και αξιολόγηση της εφαρμογής.

Χρήση:

Δοκιμές αυτοματοποίησης και προσομοίωση διαμόρφωσης δικτύων.

Η δομή της εργασίας έχει ως εξής:

- Στο Κεφάλαιο 1, γίνεται μια εισαγωγή στο θέμα της πτυχιακής και παρουσιάζονται οι στόχοι της εργασίας.
- Στο Κεφάλαιο 2, αναλύονται οι απαιτήσεις και οι προδιαγραφές του έργου, περιλαμβάνοντας τις τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν.
- Στο Κεφάλαιο 3, περιγράφεται η μεθοδολογία ανάπτυξης της εφαρμογής.
- Στο Κεφάλαιο 4, παρουσιάζονται η υλοποίηση της εφαρμογής.
- Στο Κεφάλαιο 5, γίνεται περιγραφή των νέων τεχνολογιών(containerization)

- Στο Κεφάλαιο 6, γίνονται δοκιμές και αξιολόγηση της εφαρμογής
- Στο Κεφάλαιο 7, παρουσιάζεται το συμπέρασμα και μελλοντικές επεκτάσεις της εργασίας αυτής.
- Στο Κεφάλαιο 8, περιέχεται ένα Παράρτημα και στο Κεφάλαιο 9 η Βιβλιογραφία.

Κεφάλαιο 4

Θεωρητικό Υπόβαθρο

4.1 Τεχνολογίες του software development

4.1.1 Django Framework, Paramiko, Netmiko και Napalm

Το Django είναι ένα backend framework το οποίο βασίζεται στη γλώσσα προγραμματισμού Python. Με το Django, μπορείτε να μεταφέρετε τις εφαρμογές Ιστού από την ιδέα στην κυκλοφορία μέσα σε λίγες ώρες. Το Django βοηθάει στο να στηθεί γρήγορα μία εφαρμογή ανάπτυξης ιστού έτσι ώστε να μπορούμε να εστιάσουμε στη σύνταξη της εφαρμογής και της λειτουργικότητάς της χωρίς να χρειάζεται να ανακαλύψουμε ξανά τον τροχό. Είναι δωρεάν και ανοιχτού κώδικα. Ορισμένες από τις πιο μεγάλες εταιρίες στον πλανήτη χρησιμοποιούν την ικανότητα του να κλιμακώνεται γρήγορα και με ευελιξία για να ανταποκρίνεται στις μεγαλύτερες απαιτήσεις κίνησης. Στη δικιά μας περίπτωση χρησιμοποιήθηκε το συγκεκριμένου Φραμεворκ γιατί θα μας έδινε τη δυνατότητα να φτιάχνουμε μία εφαρμογή με μεγάλη επεκτασιμότητα και παράλληλα να μπορούσαμε να ενσωματώσουμε μέσα διαφορετικές τεχνολογίες.

Παράλληλα με το Django χρησιμοποιήθηκαν έτοιμες βιβλιοθήκες της Python προκειμένου να μπορέσουν να εκτελεστούν βασικές λειτουργίες της εφαρμογής όπως τα πρωτοκόλλα επικοινωνίας. Θεωρούμε ότι η δημιουργία τέτοιων βιβλιοθηκών ξεφεύγει από τα όρια μιας διπλωματικής εργασίας καθώς απαιτεί πολύ χρόνο και ερευνητική ενασχόληση που μόνο στα πλαίσια ενός διδακτορικού θα μπορούσε να υλοποιηθεί μία τέτοια ιδέα. Παρακάτω παρουσιάζονται οι βιβλιοθήκες που χρησιμοποιήθηκαν και κάποια βασικά χαρακτηριστικά τους.

4.1.2 Paramiko

Το Paramiko είναι μια διασύνδεση καθαρά Python που υλοποιεί το πρωτόκολλο SSH έκδοσης 2 σε Python, παρέχοντας λειτουργικότητα τόσο πελάτη όσο και διακομιστή. Το Paramiko μπορεί να επιτύχει υψηλές επιδόσεις σε χαμηλού επιπέδου κρυπτογραφικές έννοιες. Οποιαδήποτε συσκευή που μπορεί να ρυθμιστεί μέσω SSH μπορεί επίσης να ρυθμιστεί από την Python με σενάρια με τη χρήση αυτής της μονάδας.

4.1.3 Netmiko

Το Netmiko είναι μια βιβλιοθήκη ανοικτού κώδικα για πολλούς προμηθευτές, που σημαίνει ότι πολλές συσκευές μπορούν να ρυθμιστούν από την python χρησιμοποιώντας το Netmiko. Ορισμένες από τις συσκευές που υποστηρίζει το Netmiko είναι οι εξής: Cisco IOS, Juniper, Arista, HP και Linux. Μπορεί επίσης να υποστηρίζει και άλλους προμηθευτές όπως η Alcatel, η Huawei και η Ubiquity αλλά περιορισμένα δοκιμές έχουν γίνει με αυτούς τους προμηθευτές. Το Netmiko τρέχει πάνω από το Paramiko για να κάνει τη σύνδεση SSH σε συσκευές δικτύου λιγότερο περίπλοκη, πιο ευέλικτη και πιο εύκολη στη χρήση. Παρόλο που το Netmiko είναι ευκολότερο στη χρήση, όπως αναφέρθηκε παραπάνω, υποστηρίζει συγκεκριμένους προμηθευτές και μόνο έναν αριθμό συσκευών τους. Από την άλλη πλευρά, το Paramiko μπορεί να χρησιμοποιηθεί για την επικοινωνία με οποιαδήποτε συσκευή που υποστηρίζει SSH. Τόσο το Paramiko όσο και το Netmiko αποτελούν εναλλακτικές επιλογές για συσκευές που δεν υποστηρίζουν APIs.

4.1.4 Napalm

Το NAPALM (Network Automation and Programmability Abstraction Layer with Multivendor support) είναι μια βιβλιοθήκη Python που υλοποιεί ένα σύνολο λειτουργιών για την αλληλεπίδραση με διαφορετικά λειτουργικά συστήματα συσκευών δικτύου χρησιμοποιώντας ένα ενοποιημένο API. Το NAPALM υποστηρίζει διάφορες μεθόδους σύνδεσης με τις συσκευές, χειρισμού των ρυθμίσεων ή ανάκτησης δεδομένων. Το Napalm συνεπώς είναι μια βιβλιοθήκη Python που παρέχει ένα API (Application Programming Interface) για την εργασία με συσκευές δικτύου. Έχει σχεδιαστεί για να απλοποιεί την αυτοματοποίηση και τη διαχείριση του δικτύου με την αφαίρεση των υποκείμενων λεπτομερειών που σχετίζονται με τον εκάστοτε προμηθευτή και την παροχή μιας συνεπούς διεπαφής σε διαφορετικά δίκτυα. συσκευών. Το Napalm επιτρέπει στους μηχανικούς και τους διαχειριστές δικτύων να αυτοματοποιούν κοινές εργασίες διαχείρισης δικτύου, όπως η διαμόρφωση, η παροχή, η παρακολούθηση και η αντιμετώπιση προβλημάτων. Υποστηρίζει πολλούς προμηθευτές συσκευών δικτύου, συμπεριλαμβανομένων των Cisco, Juniper, Arista και Huawei. Το Napalm παρέχει ένα σύνολο κοινών λειτουργιών που μπορούν να εκτελεστούν σε συσκευές δικτύου, όπως η ανάκτηση πληροφοριών διαμόρφωσης, η εφαρμογή διαμόρφωσης αλλαγών, έλεγχος στατιστικών στοιχείων διασύνδεσης και συλλογή πληροφοριών τοπολογίας δικτύου παρέχει επίσης χαρακτηριστικά όπως υποστήριξη επαναφοράς, επικύρωση διαμόρφωσης αλλαγών διαμόρφωσης, και σύγκριση των διαφορών διαμόρφωσης μεταξύ συσκευών. ¹⁴ Το Napalm μπορεί να συνδυαστεί με βιβλιοθήκες Python όπως οι Netmiko, Paramiko και Ansible για τη δημιουργία σύνθετων ροών εργασίας αυτοματισμού δικτύου. Μπορεί επίσης να ενσωματωθεί με δημοφιλή εργαλεία παρακολούθησης δικτύου, όπως το Prometheus και το Grafana, για την παρακολούθηση της απόδοσης του δικτύου σε πραγματικό χρόνο.

4.1.5 Version Control με Git και GitHub

Το Git είναι ένα σύγχρονο σύστημα ελέγχου εκδόσεων (γνωστό και ως σύστημα διαχείρισης αναθεωρήσεων ή πηγαίου κώδικα), σχεδιασμένο με έμφαση στην ταχύτητα, την ακεραιότητα των δεδομένων και την υποστήριξη κατανεμημένων, μη γραμμικών ροών εργασίας. Στην παρούσα διπλωματική εργασία, θα αξιοποιήσουμε το Git για να διασφαλίσουμε την ορθή διαχείριση των εκδόσεων του λογισμικού, τόσο κατά τη διάρκεια της ανάπτυξης όσο και για την πιθανή μελλοντική χρήση και εξέλιξη της δουλειάς μας. Το GIT συνεπώς είναι απαραίτητο σε κάθε σοβαρό έργο ανάπτυξης, και αυτό δεν αποτελεί εξαίρεση. Παρέχει γρήγορη ανάπτυξη κώδικα, έκδοση και επιτρέπει διακλαδώσεις. Έχοντας το εγκατεστημένο στον τοπικό υπολογιστή ανάπτυξης και στο περιβάλλον παραγωγής επιτρέπει την εύκολη και γρήγορη ανάπτυξη στο περιβάλλον παραγωγής. τον ήδη δοκιμασμένο κώδικα στο τοπικό περιβάλλον ανάπτυξης. Η έκδοση παρέχει τη δυνατότητα επαναφοράς σε προηγούμενες εκδόσεις κώδικα σε περίπτωση που κάποιο άγνωστο σφάλμα εμφανιστεί σε μια νεότερη έκδοση.

4.1.6 Συνεχής Ενσωμάτωση και Παράδοση (CI/CD)

Μόλις η εφαρμογή ιστού συνδεθεί με το απομακρυσμένο αποθετήριο, η τελευταία τάση στο στον κόσμο του DevOps είναι η υλοποίηση ενός αγωγού CI/CD, ο οποίος ουσιαστικά είναι μια αυτοματοποιημένη διαδικασία που ενεργοποιείται όταν νέος κώδικας δημοσιεύεται στο απομακρυσμένο αποθετήριο. Αυτή η διαδικασία ξεκινάει τη δημιουργία κώδικα, εκτελεί κάποιες δοκιμές και τέλος, αν όλα είναι εντάξει, αναπτύσσει αυτόματα τον κώδικα στην παραγωγή περιβάλλον. Με αυτόν τον τρόπο, οι προγραμματιστές μπορούν να διασφαλίσουν ότι τίποτα δεν θα χαλάσει στην παραγωγή και οι νέες λειτουργικότητες εξυπηρετούνται το συντομότερο δυνατό στους πελάτη. Στην περίπτωσή μας τόσο η διπλωματική εργασία(latex) όσο και η εφαρμογή υλοποιήθηκαν με αυτή τη λογική.

4.2 Τεχνολογίες του Software deployment

4.2.1 Containers και Docker

Το Docker είναι μια πλατφόρμα που επιτρέπει τη δημιουργία, τη διανομή και την εκτέλεση εφαρμογών μέσα σε ελαφριά, απομονωμένα "κοντέινερ" (containers). Τα κοντέινερ περιλαμβάνουν ό,τι χρειάζεται μια εφαρμογή για να τρέξει, όπως κώδικα, βιβλιοθήκες και εξαρτήσεις, διασφαλίζοντας ότι θα λειτουργεί ομοιόμορφα ανεξάρτητα από το περιβάλλον στο οποίο εκτελείται. Με αυτόν τον τρόπο διευκολύνει τη διαχείριση και τη μεταφορά εφαρμογών από τον έναν υπολογιστή ή διακομιστή στον άλλον.

4.2.2 Kubernetes και Container Orchestration

Ο κυβερνήτης είναι ο διαχειριστής των με απλά λόγια ο διαχειριστής των containers. Είναι μια πλατφόρμα ανοικτού κώδικα για τη διαχείριση φορτίων εργασίας και υπηρεσιών που περιέχουν containers, η οποία διευκολύνει τόσο τη δηλωτική διαμόρφωση όσο και την αυτοματοποίηση. Διαθέτει ένα μεγάλο, ταχέως αναπτυσσόμενο οικοσύστημα. Οι υπηρεσίες, η υποστήριξη και τα εργαλεία του Kubernetes είναι ευρέως διαθέσιμα.

4.3 Αυτοματοποίηση Διαχείρισης Δικτύου

4.3.1 Network Automation

Η αυτοματοποίηση δικτύου δεν περιορίζεται μόνο στη διαμόρφωση συσκευών. Αντιθέτως, το πιο σημαντικό μέρος της αυτοματοποίησης δικτύου, που συμβάλλει στη μείωση των ανθρώπινων σφαλμάτων, είναι η δυνατότητα που παρέχει στους διαχειριστές να αυτοματοποιούν διαδικασίες για τη διενέργεια ελέγχων συμμόρφωσης και επικύρωσης της τρέχουσας διαμόρφωσης ή οποιασδήποτε διαμόρφωσης πρόκειται να εφαρμοστεί.

Αυτό έχει ως αποτέλεσμα τη μείωση του χρόνου υλοποίησης των αλλαγών στο δίκτυο και του κινδύνου διακοπής ή διατάραξης της υπηρεσίας, ενώ ελαχιστοποιεί την πιθανότητα ανθρώπινου λάθους και διασφαλίζει την ευθυγράμμιση με τις πολιτικές του δικτύου. Μία ακόμη διαδικασία που μπορεί να αυτοματοποιηθεί είναι η επίλυση προβλημάτων. Όταν προκύπτει κάποιο πρόβλημα στο δίκτυο, το πρώτο βήμα για την αντιμετώπισή του είναι η συλλογή πληροφοριών. Η συλλογή πληροφοριών από κάθε συσκευή μπορεί να είναι χρονοβόρα και περίπλοκη, κάτι που είναι κρίσιμο, διότι συνήθως, στο μεταξύ, το δίκτυο ή ένα μέρος του παραμένει εκτός λειτουργίας.

Με τη χρήση της αυτοματοποίησης δικτύου, μπορούμε να αυτοματοποιήσουμε τις εντολές που απαιτούνται για τη συλλογή των απαραίτητων πληροφοριών για την επίλυση προβλημάτων και να έχουμε πρόσβαση σε αυτές σε πραγματικό χρόνο. Η προγραμματική συλλογή αυτών των πληροφοριών επιτρέπει και τον έλεγχο τους σε πραγματικό χρόνο. Ο έλεγχος πληροφοριών σε πραγματικό χρόνο και η λήψη αποφάσεων για τις απαραίτητες ενέργειες, εάν, για παράδειγμα, αλλάξει η τιμή κάποιου παραμέτρου, όπως το MTU, αποτελεί μία τρίτη πτυχή της αυτοματοποίησης δικτύου, γνωστή ως αυτοματοποιημένη παρακολούθηση. Η αυτοματοποιημένη παρακολούθηση βοηθά στην πρόληψη βλαβών που προκαλούνται από αποτυχίες υλικού.

4.3.2 Cisco IOS

Το IOU σημαίνει IOS on Unix είναι μια εικονική έκδοση του λογισμικού IOS της Cisco που μπορεί να χρησιμοποιηθεί για σκοπούς προσομοίωσης και δοκιμής δι-

κτύου. Επιτρέπει στους μηχανικούς δικτύου να δημιουργούν εικονικές τοπολογίες δικτύου και να εξασκούνται σε διάφορες εργασίες δικτύου, όπως η διαμόρφωση δρομολογητών και μεταγωγέων, χωρίς να απαιτείται φυσικό υλικό. Το πλεονέκτημα του GNS3 σε σχέση με εφαρμογές άλλες όπως το Packet tracer είναι ότι το GNS3 μπορεί να σηκώσει πραγματικά images άρα πραγματικό λογισμικό συνεπώς οι λειτουργίες που μπορείς να κάνεις είναι πολύ περισσότερες.

4.3.3 REST APIs για Δικτυακές Συσκευές Cisco

Το Django, σε συνδυασμό με το Django REST Framework (DRF), είναι μια ισχυρή επιλογή για την κατασκευή backend εφαρμογών που αλληλεπιδρούν με REST APIs, συμπεριλαμβανομένων των APIs της Cisco. Μπορείς να χρησιμοποιήσεις το Django για να αυτοματοποιήσεις και να διαχειριστείς δικτυακές συσκευές της Cisco μέσω αυτών των APIs.

4.3.4 GNS3 και Εικονικά Δίκτυα

Το GNS3 Είναι ένα εργαλείο προσομοίωσης δικτύων ανοικτού κώδικα που επιτρέπει στους χρήστες να προσομοιώσουν σύνθετες τοπολογίες δικτύων στους υπολογιστές τους. Μηχανικοί δικτύων και φοιτητές το χρησιμοποιούν ευρέως για να μάθουν και να εξασκηθούν σε έννοιες δικτύωσης, να δοκιμάσουν διαμορφώσεις δικτύου και να δημιουργήσουν εικονικά περιβάλλοντα δικτύου.

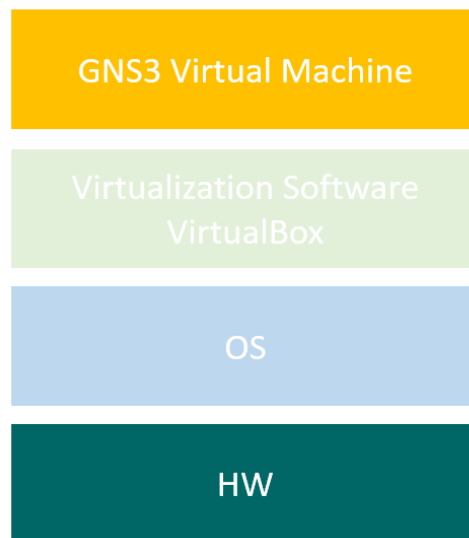
Το GNS3 υποστηρίζει διάφορες συσκευές δικτύου, όπως δρομολογητές, μεταγωγείς και τείχη προστασίας από διάφορους προμηθευτές, συμπεριλαμβανομένων των Cisco, Juniper, Nokia και άλλων. Επιτρέπει στους χρήστες να προσομοιώσουν διάφορα σενάρια και διαμορφώσεις δικτύου και να δοκιμάσουν τη συμπεριφορά των συσκευών δικτύου σε ένα ελεγχόμενο περιβάλλον.

Στην επιστήμη της πληροφορικής, η 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. Το αντίθετο της εικονικοποίησης είναι η διαφάνεια: ένας εικονικός πόρος είναι ορατός, αντιληπτός, αλλά στην πραγματικότητα ανύπαρκτος, ενώ ένας διαφανής πόρος είναι υπαρκτός αλλά αόρατος.

Θα εξηγήσουμε την εικονικοποίηση στην δικιά μας περίπτωση. Το πρώτο επίπεδο είναι αυτό του υλικού. Η εικονικοποίηση σα τεχνολογία εικονοποιεί το υλικό για να μπορέσει να δώσει πόρους στις εικονικές μηχανές. Η υλοποίηση της εικονικοποίησης γίνεται με λογισμικό hypervisor. Στη δικιά μας περίπτωση ο hypervisor είναι το Virtual Box ο οποίος είναι ένας τύπου B hypervisor. Ο hypervisor τύπου 2 είναι μια εφαρμογή εγκατεστημένη στο λειτουργικό σύστημα του κεντρικού υπολογιστή το οποίο μας δίνει τη δυνατότητα να σηκώσουμε εικονικές μηχανές άλλων λειτουργικών συστημάτων πάνω στο ήδη υπάρχον σύστημα.

Οι παρακάτω εικόνες μπορούν να εξηγήσουν σχηματικά τη γενική καθώς και την ειδική αρχιτεκτονική.

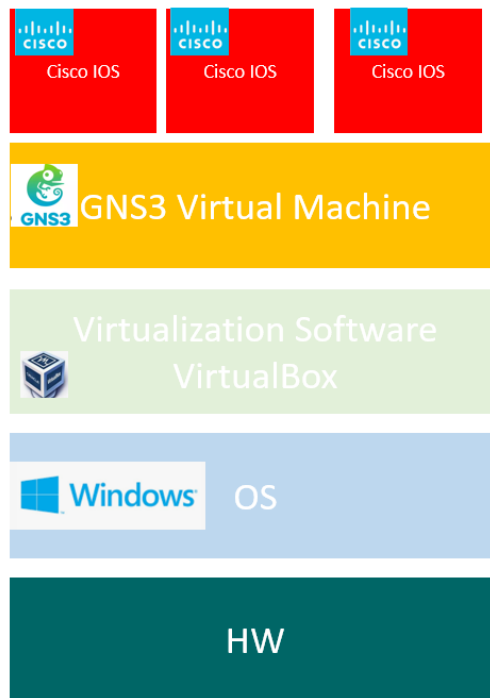


Σχήμα 4.1: Virtualization Γενική αρχιτεκτονική

4.4 Πρόγραμμα εικονοποίησης για το GNS3 VM-VirtualBox

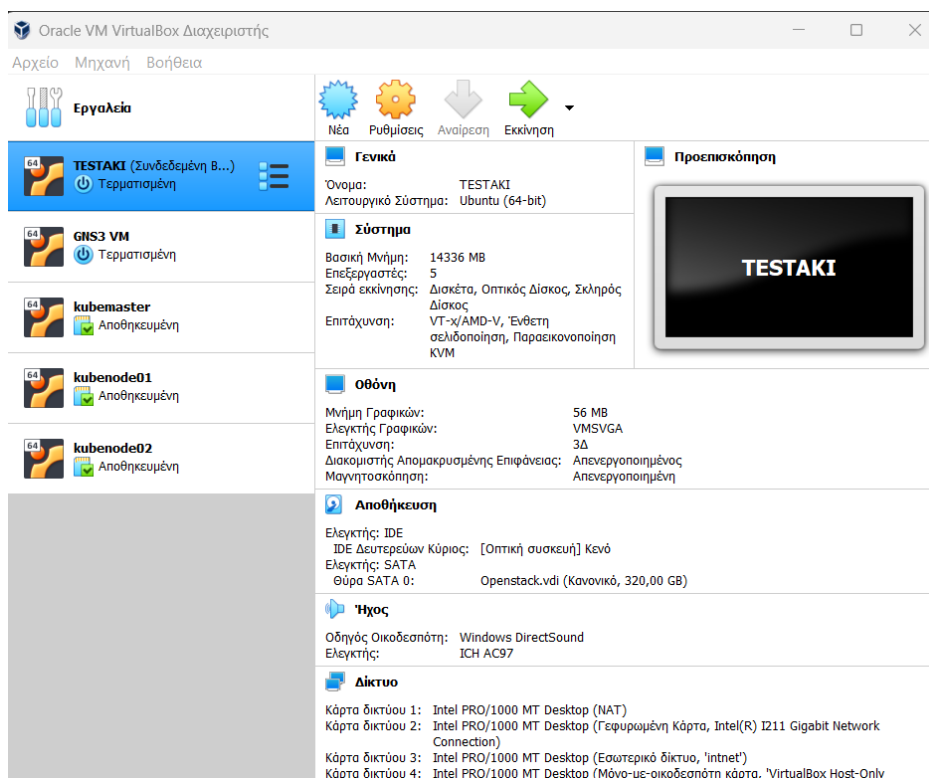
Το Oracle VM VirtualBox ή VirtualBox (πρώην Sun VirtualBox, Sun xVM VirtualBox και Innotek VirtualBox) είναι υπερεπόμενη ανοιχτού κώδικα για υπολογιστές x86 που αναπτύσσεται από την Oracle Corporation. Αναπτύχθηκε αρχικά από την Innotek GmbH και αποκτήθηκε από τη Sun Microsystems το 2008, η οποία εξαγοράστηκε από την Oracle το 2010.

Το VirtualBox μπορεί να εγκατασταθεί σε διάφορα λειτουργικά συστήματα, συμπεριλαμβανομένων των Linux, macOS, Windows, Solaris και OpenSolaris. Υπάρχουν επίσης μεταφορές για το FreeBSD και το Genode. Υποστηρίζει τη δημιουργία και τη διαχείριση εικονικών μηχανών που εκτελούν εκδόσεις και παραλλαγές των Microsoft Windows, Linux, BSD, Solaris, Haiku, OSx86 και άλλα, καθώς και περιορισμένη εικονικοποίηση macOS. Για ορισμένα λειτουργικά συστήματα είναι διαθέσιμο ένα πακέτο "Guest Additions" από μηχανές συσκευών και εφαρμογές συστήματος που συνήθως βελτιώνει την απόδοση, ειδικά των γραφικών, επίσης δίνει την



Σχήμα 4.2: Virtualization Γενική αρχιτεκτονική

δυνατότητα στον χρήστη να μεταφέρει αρχεία ή κείμενο από μία εικονική μηχανή στον υπολογιστή του χρήστη και να αυξήσει την ανάλυση του παράθυρου της μηχανή. Στην εικόνα 3.4 μπορούμε να δούμε το VirtualBox και την εικονική μηχανή GNS3 VM



Σχήμα 4.3: Virtualbox

Κεφάλαιο 5

Σχεδίαση και Ανάλυση

5.1 Απαιτήσεις Συστήματος

5.1.1 Λειτουργικές απαιτήσεις

- Η δυνατότητα παροχής στον χρήστη ενός Γραφικού Περιβάλλοντος, μέσω του οποίου μπορεί να δημιουργεί αντικείμενα που θα χρησιμεύσουν ως βασικά στοιχεία για τις επόμενες λειτουργίες.
- Η δημιουργία λειτουργίας για τη δυνατότητα επίβλεψης της κατάστασης επιλεγόμενης διεπαφής.
- Η δημιουργία λειτουργίας για τη δυνατότητα επίβλεψης στατιστικών για επιλεγόμενη δικτυακή συσκευή.
- Η δημιουργία λειτουργίας για τη δυνατότητα επίβλεψης στατιστικών επιλεγόμενης διεπαφής.
- Η δημιουργία λειτουργίας για τη δυνατότητα συλλογής ως backup τρέχοντος configuration.
- Η δημιουργία λειτουργίας για τη δυνατότητα αλλαγής IP διεύθυνσης επιλεγόμενης δικτυακής εφαρμογής.

5.1.2 Τοπικό περιβάλλον ανάπτυξης

Το λογισμικό πάνω στο οποίο δοκιμάστηκε η εφαρμογή είναι Linux, Ubuntu 22.04.2 LTS η οποία εικονοποιήθηκε πάνω σε λειτουργικό Windows ως WSL2. Το Windows Subsystem for Linux version 2 είναι μια τεχνολογία της Microsoft που επιτρέπει στους χρήστες Windows να τρέχουν Linux περιβάλλοντα απευθείας στο λειτουργικό σύστημα Windows, χωρίς την ανάγκη για εξομοιωτές ή εικονικές μηχανές. Είναι η δεύτερη έκδοση του Windows Subsystem for Linux και αποτελεί σημαντική βελτίωση σε σχέση με την πρώτη έκδοση WSL1. Το WSL 2 χρησιμοποιεί την τεχνολογία εικονικοποίησης (virtualization) για να τρέχει έναν πραγματικό πυρήνα Linux μέσα

σε μια ελαφριά εικονική μηχανή βοηθητικών λειτουργιών (utility VM). Αυτό επιτρέπει στο WSL 2 να προσφέρει καλύτερη απόδοση, πλήρη συμβατότητα με τις λειτουργίες Linux και πρόσβαση σε εργαλεία όπως το Docker.

5.2 Η επανάσταση στο web development

Στην αρχή, οι εφαρμογές ιστού δεν ήταν τίποτα περισσότερο από ένα σύνολο αρχείων HTML, CSS και javascript που ήταν συνδεδεμένα μεταξύ τους. Ένας καλός προγραμματιστής ήταν σε θέση να φτιάξει σπουδαίες εφαρμογές ιστού αν αυτός/αυτή είχε αρκετές δεξιότητες/γνώσεις.

Στην εποχή μας, εμφανίστηκαν τα frameworks και λαμβάνοντας υπόψη ότι ουσιαστικά δεν βελτιώνουν αυτό που τελικά βλέπει ο χρήστης και τις αλληλεπιδράσεις του με το frontend, τότε θα μπορούσε κανείς να αναρωτηθεί γιατί χρησιμοποιούνται ευρέως στις μέρες μας. Παρόμοιες δουλειές με την παρούσα εργασία υπάρχουν και σε άλλες διπλωματικές εργασίες καθώς και σε μη διπλωματικές εργασίες. Μηχανικοί από όλο τον κόσμο ασχολούνται με την αυτοματοποίηση συστημάτων και τη δημιουργία κώδικα που να αυτοματοποιεί συσκευές/συστήματα.

Με βάση άλλες τέτοιες προσπάθειες που έχουν γίνει στο παρελθόν εμείς συλλέξαμε την έως τώρα βιβλιογραφία και προσπαθήσαμε να φτιάξουμε μία τέτοια εφαρμογή η οποία όμως να βασίζεται στα τωρινά δεδομένα και να ενσωματώσουμε τις τελευταίες τεχνολογίες αιχμής όπως την Cloud Native αρχιτεκτονική. Στη συνέχεια γίνεται προσπάθεια να δοθεί εκτενής ανάλυση στο πως λειτουργεί η εφαρμογή καθώς και στην αλληλεπίδρασή της με τα συνεργαζόμενα συστήματα.

5.3 Αρχιτεκτονική της εφαρμογής

5.3.1 Μοντέλο MVC (Model-View-Controller)

Το μοτίβο MVC είναι ένα αρχιτεκτονικό μοτίβο ανάπτυξης λογισμικού που διαχωρίζει την παρουσίαση δεδομένων από τη λογική διαχείρισης των αλληλεπιδράσεων του χρήστη. Υπάρχει ως ιδέα εδώ και καιρό και έχει δει εκθετική αύξηση στη χρήση του από την εισαγωγή του. Επίσης, έχει χαρακτηριστεί ως ένας από τους καλύτερους τρόπους για τη δημιουργία εφαρμογών πελάτη-διακομιστή όπου ο χρήστης αντιλαμβάνεται ως γραφικό περιβάλλον. Όλα τα κορυφαία frameworks για το web είναι βασισμένα στο MVC.

Παρόλο που το Django ακολουθεί το μοτίβο MVC, προτιμά να χρησιμοποιεί τη δική του λογική στην υλοποίηση. Το framework αναλαμβάνει το Controller μέρος του MVC και αφήνει τα περισσότερα από τα "καλά" να γίνονται στο Model-Template-View (MTV).

Αυτός είναι ο λόγος που το Django συχνά αναφέρεται ως MTV framework.

5.4 Django MTV

- Model: Αντιπροσωπεύει τα δεδομένα και τη διαχείρισή τους.
- Template: Εστιάζει στο τι βλέπει ο χρήστης
- View Συνδέει το Model και το Template, διαχειριζόμενο τη λογική

Με άλλα λόγια, το Django κρατά την πολυπλοκότητα μακριά από τον προγραμματιστή, καθιστώντας την εμπειρία πιο απλή και παραγωγική.

5.4.1 Διαγραμματική απεικόνιση της εφαρμογής

Model(Μοντέλο): Το Device είναι το κύριο μοντέλο που χρησιμοποιείται εδώ. Αναπαριστά τις δικτυακές συσκευές (π.χ. δρομολογητές, switches) και τις σχετικές πληροφορίες τους:

Πεδία όπως: host, username, password, platform, και secret. Το μοντέλο αυτό συνδέεται με τη βάση δεδομένων και παρέχει API για CRUD λειτουργίες (Create, Read, Update, Delete). Το μοντέλο αυτό είναι η βάση δεδομένων μας.

Template (Πρότυπο):

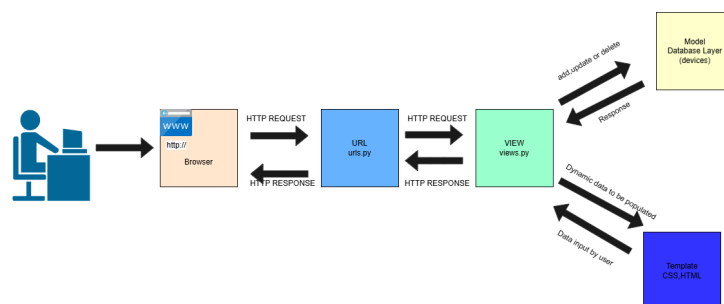
Τα αρχεία .html είναι τα πρότυπα που χρησιμοποιούνται για την παρουσίαση δεδομένων.

Προβολή λίστας συσκευών. Παρουσίαση στατιστικών. Εμφάνιση ρυθμίσεων διαμόρφωσης ή αποτελεσμάτων εκτέλεσης εντολών. Χρησιμοποιούν τη γλώσσα Django Template για δυναμικό περιεχόμενο (π.χ., λίστες συσκευών, στατιστικά).

View (Προβολή): Οι views είναι οι Python συναρτήσεις που :

Παίρνουν αιτήματα από τον χρήστη. Επικοινωνούν με το μοντέλο για δεδομένα. Επιστρέφουν απαντήσεις με τη μορφή HTML.

Στην εικόνα παρακάτω παρουσιάζεται διάγραμμα απεικόνισης της βασικής αρχιτεκτονικής της εφαρμογής.



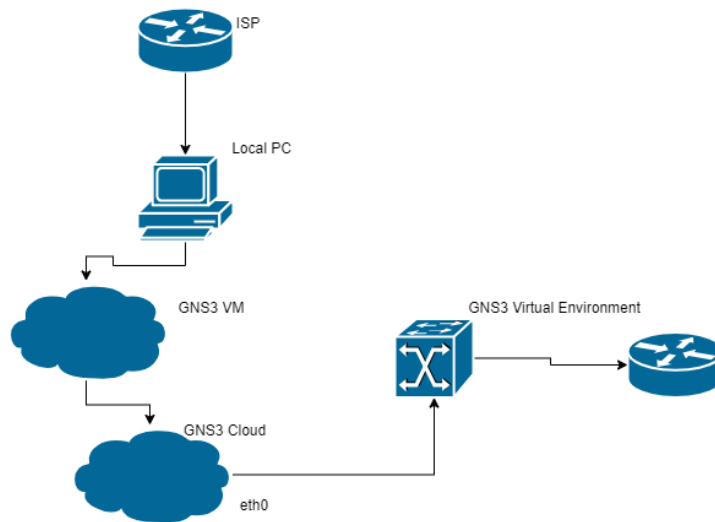
Σχήμα 5.1: MTV

5.5 Εικονικό Περιβάλλον Δικτύου GNS3 –Testbed

5.5.1 Σχεδίαση Τοπολογίας Δικτύου

Η λογική την οποία χρησιμοποιήσαμε για τη δημιουργία του Testbed είναι ότι οι συσκευές οι οποίες τρέχουν πάνω στο GNS3 VM θα μπορέσουν να αλληλεπιδράσουν με το τοπικό WSL2 περιβάλλον. Προκειμένου να επιτευχθεί αυτός ο στόχος ακολουθήθηκαν βήματα. Το πρώτο βήμα είναι η εγκατάσταση βασικών προγραμμάτων τα οποία έχουν παρουσιαστεί στο Θεωρητικό υπόβαθρο.

Σε High Level overview η εφαρμογή ακολουθεί την παρακάτω τοπολογία :



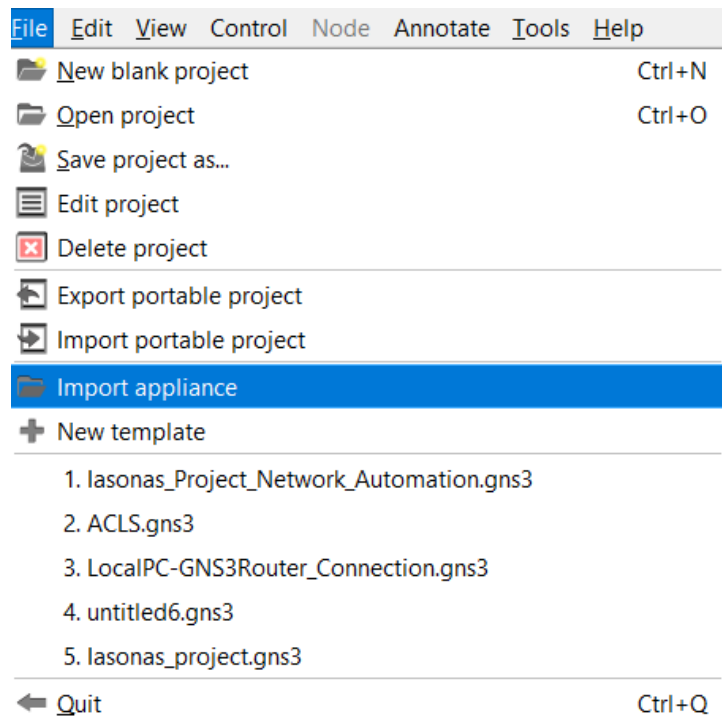
Σχήμα 5.2: Local PC-GNS3VM-CISCO IOS Connection Architecture

5.5.2 Προσομοίωση Συσκευών Cisco

Προκειμένου να προσομοιώσουμε συσκευές της Cisco το πρώτο βήμα είναι να κατεβάσουμε συγκεκριμένο appliance από το GNS3 marketplace. Αφού το κατεβάσουμε το εισάγουμε στο GNS3 με τον εξής τρόπο: (εικόνα 5.3)

Ακολουθώντας πατάμε Install appliance on the GNS3 VM και ματοσάρνοντας το filename του appliance με το image που έχουμε μας επιτρέπει να εισάγουμε τη συσκευή.

Για να δούμε το filename στο appliance ανοίγουμε το αρχείο με έναν editor και αλλάζουμε το filename αντίστοιχα όπως στην εικόνα 5.4.



Σχήμα 5.3: Import appliance

```

"images": [
  {
    "filename": "c7200-adventerprisek9-mz.153-3.XB12.image",
    "version": "153-3.XB12",
    "md5sum": "3d234a3793331c972776354531f87221",
    "filesize": 131471340
  },

```

Σχήμα 5.4: filename configuration

Μετά το τέλος της διαδικασίας η συσκευή θα έχει προστεθεί και μπορούμε να την δούμε στην επιλογή Browse all devices

Κεφάλαιο 6

Application Demo

6.1 Εισαγωγή-Η λογική της λειτουργίας της εφαρμογής Δθανγο

Όταν τρέχουμε τον διακομιστή της εφαρμογής η εφαρμογή ιστού είναι διαθέσιμη από οποιοδήποτε φυλλομετρητή. Η αρχική σελίδα στη εφαρμογή Django δηλώνεται στο πλαίσιο του Django run server ως μία συνάρτηση της Python η οποία δέχεται ένα http request και σαν απάντηση επιστρέφει την αρχική σελίδα(index.html).

Μέσα από αυτή την αρχική σελίδα μπορούμε να κατευθυνθούμε σε οποιαδήποτε επιλογή εμείς θέλουμε. Στο Django καθοριστικός παράγοντας στην ευκολία με την οποία μπορείς να χτίσεις μια εφαρμογή από την αρχή είναι ο τρόπος που χειρίζεται τη δρομολόγηση των διαφόρων σελίδων. Αυτό γίνεται μεσα απο το urls.py

Αυτή η ρύθμιση καθορίζει πώς θα δρομολογούνται οι αιτήσεις HTTP προς συγκεκριμένες λειτουργίες (views) στην εφαρμογή Django. Με τη χρήση των δυναμικών παραμέτρων, μπορούμε συνεπώς να δημιουργήσουμε πιο ευέλικτες διαδρομές URL που μπορούν να χειρίζονται ποικίλες καταστάσεις.

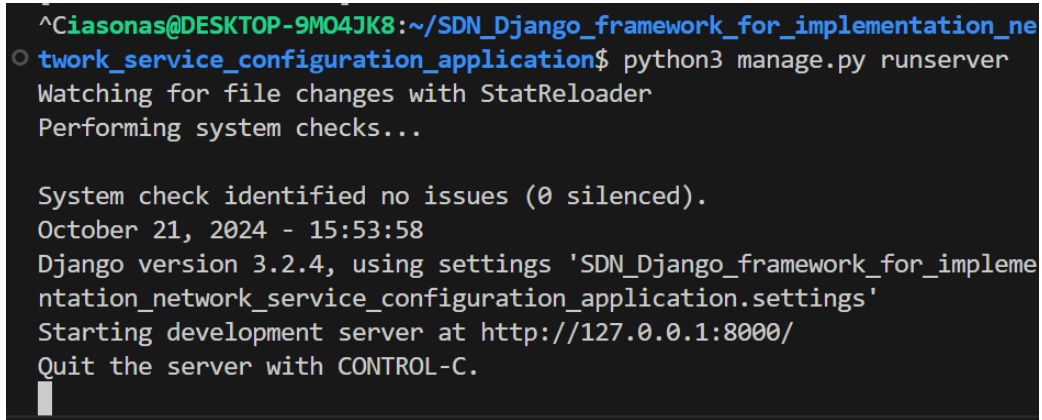
Κάθε ένα από αυτά τα paths που φαίνονται και παρακάτω στην εικόνα είναι υπεύθυνα για την ανακατεύθυνση των διευθύνσεων και τη σωστή δρομολόγησή τους ώστε να δώσουν σαν απάντηση κάθε φορά τα σωστά δεδομένα.

```
urlpatterns = [
    path('', views.firstPage),
    path('configure-ip/', views.configure_ip, name='configure_ip'),
    path('manage/', views.index, name="manage"),
    path('manage2/', views.index2, name="manage2"),
    path('manage3/', views.index3, name="manage3"),
    path('device_statistics/<int:device_id>', views.get_interface_statistics, name="device_statistics"),
    path('interface_statistics/<int:device_id>', views.get_interfaces_counters, name="interface_statistics"),
    path('device/<int:device_id>', views.get_device_stats, name="device"),
    path('execute_script/', views.execute_script_on_remote, name='execute_script'),
    path('manage4/', views.index4, name="manage4"),
    path('manage5/', views.index5, name="manage5"),
    path('running_config/<int:device_id>', views.get_running_config, name="running_config"),
    path('show_running_config/<int:device_id>', views.show_running_config, name='show_running_config'),
```

Σχήμα 6.1: Urls.py αρχείο

6.2 Η αρχική σελίδα της εφαρμογής

Προκειμένου να τρέξει ο Django server τρέχουμε την εντολή η οποία βρίσκεται στην εικόνα 5.1.

A terminal window with a dark background and light-colored text. The prompt is '^Ciasonas@DESKTOP-9MO4JK8:~/SDN_Django_framework_for_implementation_ne'. The user has entered 'work_service_configuration_application\$ python3 manage.py runserver'. The output shows 'Watching for file changes with StatReloader', 'Performing system checks...', 'System check identified no issues (0 silenced).', the date and time 'October 21, 2024 - 15:53:58', 'Django version 3.2.4, using settings 'SDN_Django_framework_for_impleme ntation_network_service_configuration_application.settings'', 'Starting development server at http://127.0.0.1:8000/', and 'Quit the server with CONTROL-C.' followed by a cursor.

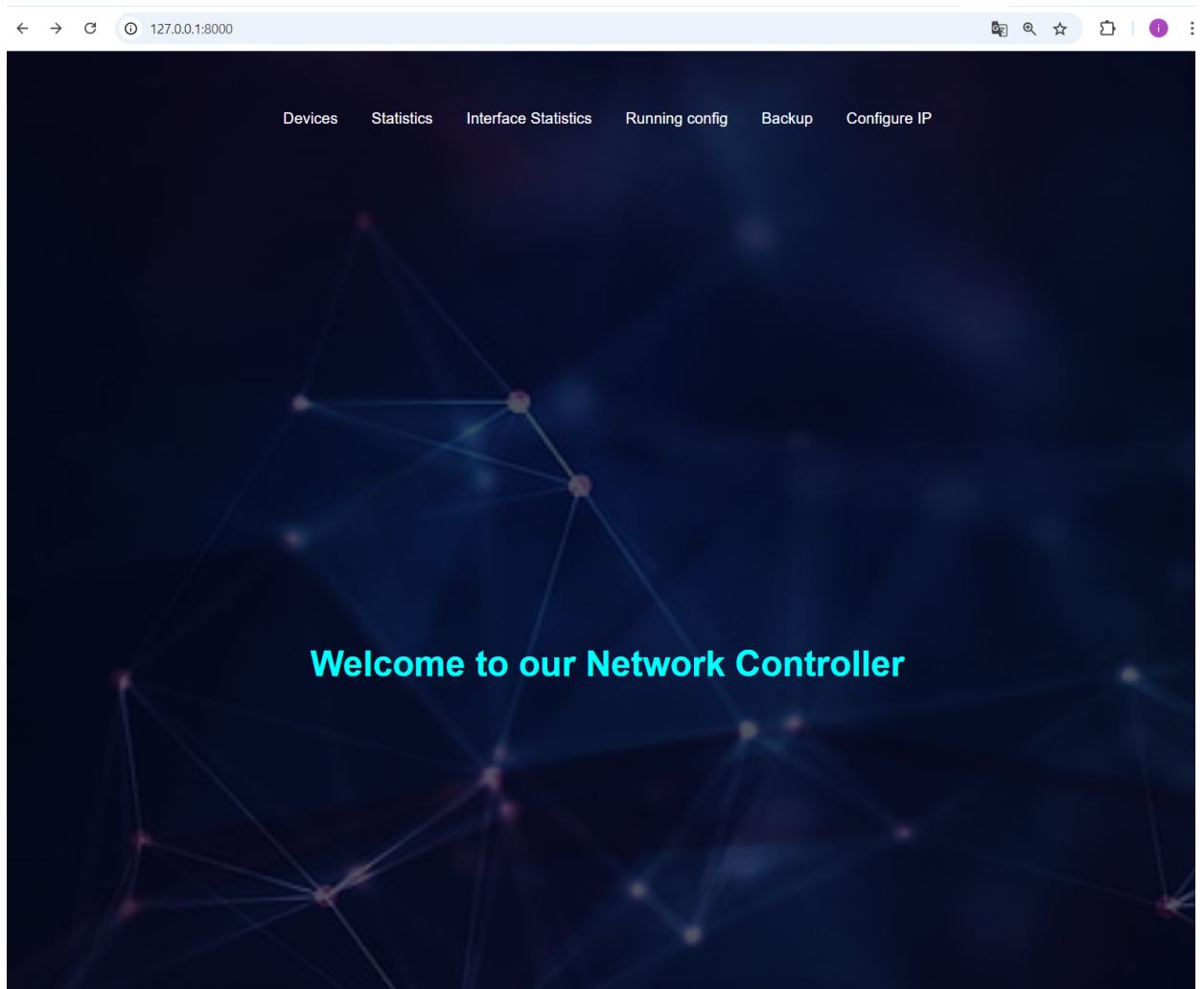
```
^Ciasonas@DESKTOP-9MO4JK8:~/SDN_Django_framework_for_implementation_ne
work_service_configuration_application$ python3 manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
October 21, 2024 - 15:53:58
Django version 3.2.4, using settings 'SDN_Django_framework_for_impleme
ntation_network_service_configuration_application.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Σχήμα 6.2: Django run server

Ο εξυπηρετητής τρέχει σαν διεργασία στο λειτουργικό και είναι διαθέσιμος στη διεύθυνση <http://127.0.0.1:8000/>

Εάν εισάγουμε αυτή τη διεύθυνση σε έναν φυλλομετρητή της επιλογής μας, θα ανακατευθυνθούμε στην αρχική σελίδα, η οποία θα παρουσιαστεί στον χρήστη όπως στην εικόνα 5.2. Το εμπρόστιο τμήμα της εφαρμογής(frontend) είναι γραμμένο με HTML,CSS ενώ το οπίσθιο τμήμα(backend) είναι γραμμένο σε Python.



Σχήμα 6.3: Αρχική Σελίδα

6.3 Devices

Προκειμένου να δημιουργήσουμε μία νέα συσκευή η παρακατω κλάση κώδικα μας βοηθάει στο να γίνει όπως στο σχήμα 5.3

```
class Device(models.Model):
    name = models.CharField(max_length=100)
    host = models.CharField(max_length=70)
    username = models.CharField(max_length=100)
    password = models.CharField(max_length=100, blank=True)
    secret = models.CharField(max_length=100, blank=True)
    device_type = models.CharField(
        max_length=30, choices=(("router", "Router"), ("switch", "Switch"), ("firewall", "Firewall")), blank=True
    )

    platform = models.CharField(
        max_length=30, choices=(("cisco_ios", "Cisco IOS"), ("cisco_iosxe", "Cisco IOS XE")), blank=True
    )
```

Σχήμα 6.4: Αρχικοποίηση συσκευής

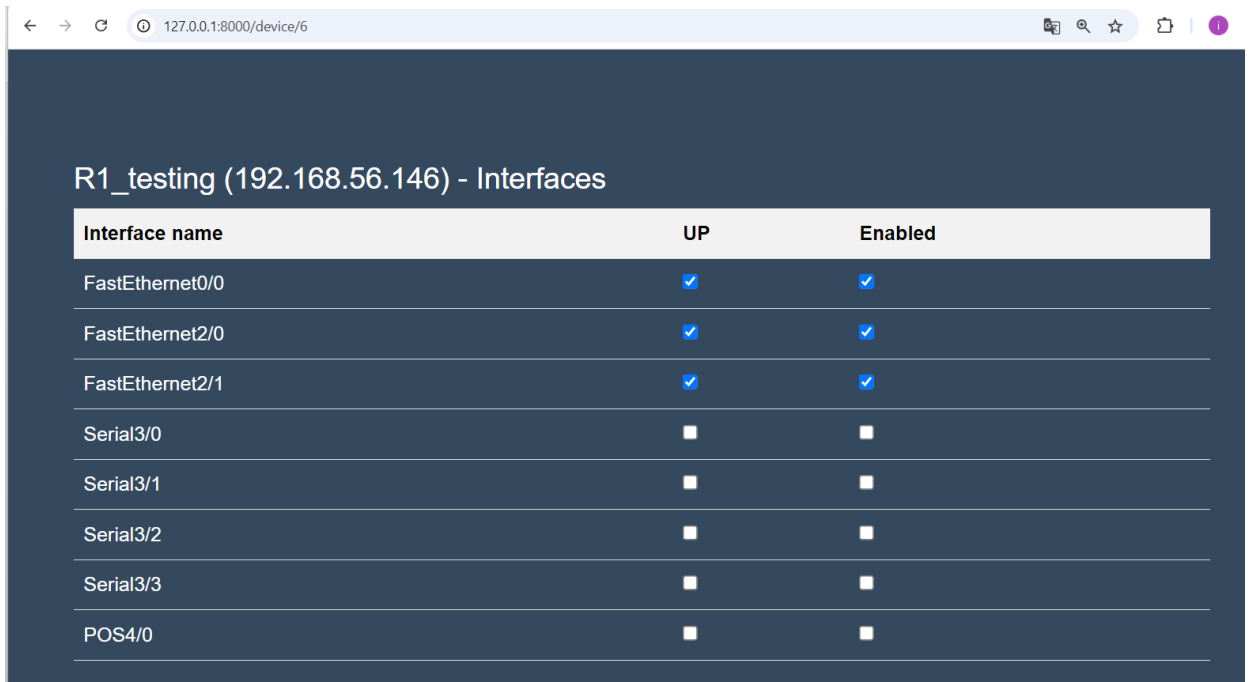
Αφού λοιπόν πατήσουμε το κουμπί Devices αυτό θα μας δρομολογήσει στο αντίστοιχο HTML link. Το οποίο θα μας εμφανίσει μια άλλη σελίδα αυτή του σχήματος 5.4.

Αποτέλεσμα του πίνακα του σχήματος 5.4 είναι όλες οι συσκευές οι οποίες είναι στη βάση δεδομένων μας όπου ο χρήστης πρόσθεσε προκειμένου να μπορεί να αναδράσει με αυτές.

CONTROLLER		
Device Interfaces		
Id	Name	Host
1	R1	192.168.2.9
2	R4	192.168.56.152
3	Router_Ericsson	192.168.56.120
4	R2_Ericsson	192.168.2.15
5	R3	192.168.56.123
6	R1_testing	192.168.56.146

Σχήμα 6.5: Controller-Device Interfaces

Πατώντας ένα από αυτά τα κουμπιά θα μπορέσουμε να πάρουμε το αποτέλεσμα που θέλουμε. Σε αυτή τη σελίδα μπορούμε να δούμε αν κάποια διεπαφή είναι κάτω ή πάνω συνεπώς αν λειτουργεί ή όχι. (Σχήμα 5.5)



Interface name	UP	Enabled
FastEthernet0/0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
FastEthernet2/0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
FastEthernet2/1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Serial3/0	<input type="checkbox"/>	<input type="checkbox"/>
Serial3/1	<input type="checkbox"/>	<input type="checkbox"/>
Serial3/2	<input type="checkbox"/>	<input type="checkbox"/>
Serial3/3	<input type="checkbox"/>	<input type="checkbox"/>
POS4/0	<input type="checkbox"/>	<input type="checkbox"/>

Σχήμα 6.6: Interfaces

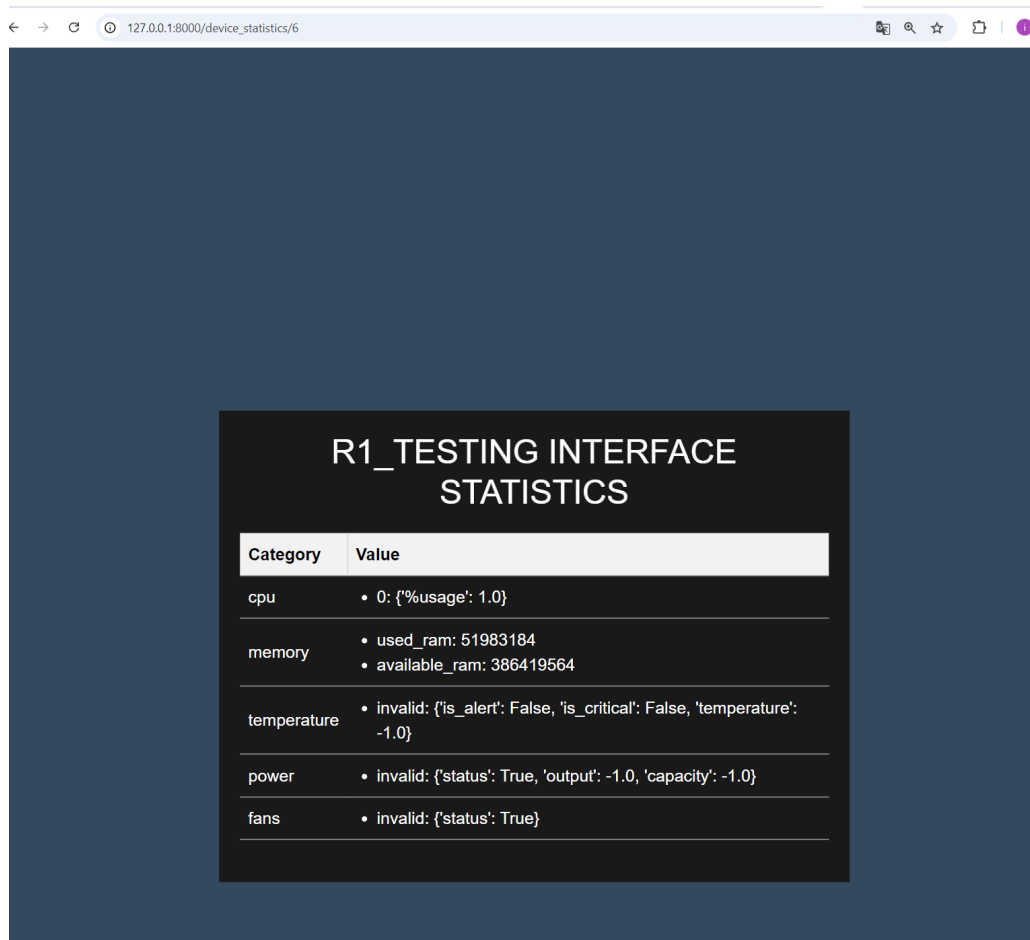
6.4 Στατιστικά της συσκευής

Η λειτουργία του κουμπιού αυτού βασίζεται στη συνάρτηση `get_interface_statistics`, η οποία είναι υπεύθυνη για τη σύνδεση σε μια δικτυακή συσκευή, την απόκτηση στατιστικών πληροφοριών σχετικά με τις διεπαφές της, και την παρουσίαση αυτών των πληροφοριών σε μια ιστοσελίδα μέσω ενός προτύπου HTML. Η συνάρτηση αυτή δέχεται δύο παραμέτρους: το αίτημα του χρήστη (`request`) και το αναγνωριστικό της συσκευής (`device_id`), το οποίο χρησιμοποιείται για να εντοπίσει τη συγκεκριμένη συσκευή από τη βάση δεδομένων.

Για να εντοπίσει τη συσκευή, παρουσιάζεται μία λίστα από το User Interface όπως και στην παραπάνω (Devices), και πατώντας πάνω της εμφανίζει τα παρακάτω στατιστικά:

6.5 Στατιστικά της διεπαφής

Η συνάρτηση επιτρέπει τη σύνδεση σε μια δικτυακή συσκευή, την ανάκτηση των στατιστικών των διεπαφών της και την παρουσίαση αυτών των δεδομένων σε μια ιστοσελίδα



Σχήμα 6.7: Στατιστικά

6.6 Backup της συσκευής

Προκειμένου να σώσουμε τη διαμόρφωση της συσκευής πατάμε το κουμπί backup και με αυτό πέρνουμε το παρακάτω αποτέλεσμα.

Ουσιαστικά είναι σε text αρχείο το λεγόμενο running config της συσκευής

← → ↻ ⓘ 127.0.0.1:8000/configure-ip/

Configure IP on Cisco Device

Device Hostname:

Username:

Password:

Interface:

IP Address:

Subnet Mask:

Σχήμα 6.10: IP Διαμόρφωση

```
R1#show ip in
R1#show ip int
R1#show ip interface br
R1#show ip interface brief
Interface IP-Address OK? Method Status Protocol
FastEthernet0/0 192.168.56.146 YES DHCP up up
FastEthernet2/0 192.168.12.2 YES NVRAM up up
FastEthernet2/1 192.168.23.2 YES NVRAM up up
Serial3/0 unassigned YES NVRAM administratively down down
Serial3/1 unassigned YES NVRAM administratively down down
Serial3/2 unassigned YES NVRAM administratively down down
Serial3/3 unassigned YES NVRAM administratively down down
POS4/0 unassigned YES NVRAM administratively down down
R1#
R1#show ip interface brief
Interface IP-Address OK? Method Status Protocol
FastEthernet0/0 192.168.56.146 YES DHCP up up
FastEthernet2/0 192.168.12.2 YES NVRAM up up
FastEthernet2/1 192.168.23.2 YES NVRAM up up
Serial3/0 unassigned YES NVRAM administratively down down
Serial3/1 unassigned YES NVRAM administratively down down
Serial3/2 unassigned YES NVRAM administratively down down
Serial3/3 unassigned YES NVRAM administratively down down
POS4/0 unassigned YES NVRAM administratively down down
R1#
*Oct 21 17:54:15.803: %SYS-5-CONFIG_I: Configured from console by iasonas on vty0 (192.168.56.1)
R1#
*Oct 21 17:54:17.395: %SYS-5-CONFIG_I: Configured from console by iasonas on vty1 (192.168.56.1)
R1#
```

Σχήμα 6.11: IP Διαφορά

Κεφάλαιο 7

Εισαγωγή τεχνολογίας του Κυβερνήτη(Kubernetes)

7.1 Εισαγωγή-Η λογική της λειτουργίας του Κυβερνήτη

Τα τελευταία χρόνια, παρατηρείται ραγδαία αύξηση στον τομέα της πληροφορικής, με την εμφάνιση και εξάπλωση νέων εννοιών, όπως ο κυβερνήτης και τα *microservices*. Ένας βασικός παράγοντας που συνέβαλε στην εισαγωγή αυτών των τεχνολογιών είναι η ικανότητα εικονικοποίησης του λειτουργικού συστήματος, καθώς και η δυνατότητα εκτέλεσης εφαρμογών ως κοντέινερ. Αυτές οι τεχνολογίες επιτρέπουν την απομόνωση και τη διαχείριση εφαρμογών με μεγαλύτερη ευελιξία και αποτελεσματικότητα, κάτι που έχει οδηγήσει σε σημαντικές αλλαγές στον τρόπο ανάπτυξης και λειτουργίας των σύγχρονων υποδομών λογισμικού

Τα κοντέινερ είναι ένας καλός τρόπος για να ομαδοποιήσουμε και να εκτελέσουμε τις εφαρμογές μας. Σε ένα περιβάλλον παραγωγής, πρέπει να διαχειριστούμε τα κοντέινερ που εκτελούν τις εφαρμογές και να διασφαλίσουμε ότι δεν υπάρχει χρόνος διακοπής λειτουργίας. Για παράδειγμα, εάν ένα κοντέινερ πέσει κάτω, ένα άλλο κοντέινερ πρέπει να ξεκινήσει.

Έτσι έρχεται να σώσει την κατάσταση ο κυβερνήτης. Το Kubernetes σάς παρέχει ένα πλαίσιο για να εκτελείτε τα κατανεμημένα συστήματα με ευελιξία. Φροντίζει για την κλιμάκωση και το *failover* για την εφαρμογή σας, παρέχει μοτίβα ανάπτυξης και πολλά άλλα. Δε θα αναλύσουμε με κάθε λεπτομέρεια τι ακριβώς είναι ο κυβερνήτης γιατί μία τέτοια προσπάθεια είναι μία διπλωματική από μόνη της αλλά θα προσπαθήσουμε να παρουσιάσουμε τα βασικά χαρακτηριστικά τα οποία χρησιμοποιήθηκαν πρακτικά στη διπλωματική.

7.2 Το αποθετήριο κοντεινερ-Docker Desktop

Προκειμένου να μπορέσουμε να χρησιμοποιήσουμε το Docker Desktop θα πρέπει να έχουμε φτιάξει την εφαρμογή μας σαν κοντέινερ.

Για να γίνει αυτό θα πρέπει να φτιάξουμε το παρακάτω Dockerfile το οποίο ουσιαστικά είναι η εφαρμογή μας αλλά σε κοντεινερ.

```
Dockerfile
1  # Use an official Python runtime as the base image
2  FROM python:3.9
3
4  # Set the working directory in the container
5  WORKDIR /app
6
7  # Copy the requirements file and install dependencies
8  COPY requirements.txt /app/
9  RUN pip install --no-cache-dir --upgrade pip
10 RUN apt-get update \
11     && apt-get install -y libyaml-dev
12
13 RUN pip install --no-cache-dir -r requirements.txt
14
15 # Copy the Django project code to the container
16 COPY . /app/
17
18 # Set the PYTHONPATH environment variable to include the Django project di
19 ENV PYTHONPATH=/app
20
21 # Expose the port on which the Django development server will run
22 EXPOSE 8000
23
24 # Run the Django development server
25 CMD ["python3", "manage.py", "runserver", "0.0.0.0:8000"]
26
```

Σχήμα 7.1: Dockerfile

Για να φτιάξουμε το image τρέχουμε τη παρακάτω εντολή

Θα πρέπει μετά να βάλουμε το image στο DockerDesktop και αφού γίνει αυτό θα μπορέσουμε να την τρέξουμε στην υποδομή του κυβερνήτη. Το ίδιο μπορεί να γίνει και χωρίς την εφαρμογή DockerDesktop αλλά τρέχοντας το Docker service απευθείας πάνω στο WSL2 όπως φαίνεται και στις παρακάτω εικόνες.


```

iasonas@DESKTOP-9M04JK8:~/SDN_Django_framework_for_implementation_network_service_configuration_application$ docker build -t django:v1 .
[+] Building 36.8s (12/12) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 743B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3.9
=> [1/7] FROM docker.io/library/python:3.9@sha256:ed8b9dd4e9f89c111f4bdb85a55f8c9f0e22796a298449380b
=> [internal] load build context
=> => transferring context: 14.1kB
=> CACHED [2/7] WORKDIR /app
=> [3/7] COPY requirements.txt /app/
=> [4/7] RUN pip install --no-cache-dir --upgrade pip
=> [5/7] RUN apt-get update && apt-get install -y libyaml-dev
=> [6/7] RUN pip install --no-cache-dir -r requirements.txt
=> [7/7] COPY . /app/
=> exporting to image
=> => exporting layers
=> => writing image sha256:67c01544160e308aea5622d3de55488251b54e023a607e2d2fffcf4886e53788
=> => naming to docker.io/library/django:v1

```

Σχήμα 7.2: Docker build-Δημιουργία του κοντεϊνερ

```

iasonas@DESKTOP-9M04JK8:~/SDN_Django_framework_for_implementation_network_service_configuration_application$ docker push s
Using default tag: latest
The push refers to repository [docker.io/iasonasi/django_thesis]
4eb48115a042: Pushed
164a177ac4f1: Pushed
da802df85c96: Pushed
63dc518f902b: Pushed
48d045e2a1f0: Pushed
dac6f4a56d09: Pushed
274fbebada99: Pushed
69470c0633ea: Pushed
249a2754c71b: Pushed
ccccc6c1c4bf: Pushed
7d98d813d54f: Pushed
ad1c7cf347f: Pushing [=====] 138.4MB/211.3MB
7aad5092c3b: Pushed
0b41a54d942d: Pushed

```

Σχήμα 7.3: Docker push-

7.3 Κυβερνήτης

Στη διπλωματική αυτή εργασία, χρησιμοποιείται ένα τοπικό περιβάλλον ανάπτυξης με Kubernetes μέσω του Minikube και του WSL2 (Windows Subsystem for Linux 2) για την ανάπτυξη και δοκιμή της εφαρμογής Django.

Το Kubernetes είναι μια δημοφιλής πλατφόρμα ενορχήστρωσης κοντεϊνερ, που επιτρέπει την αυτόματη διαχείριση και κλιμάκωση εφαρμογών σε περιβάλλοντα παραγωγής, ενώ το Minikube προσφέρει τη δυνατότητα εκκίνησης ενός τοπικού Kubernetes cluster. Με τον τρόπο αυτό, επιτυγχάνεται η δημιουργία ενός ασφαλούς, απομονωμένου περιβάλλοντος δοκιμών, το οποίο προσομοιώνει ένα πλήρες cluster, χωρίς την ανάγκη πρόσθετης υποδομής cloud.

Χάρη στο WSL2, το οποίο επιτρέπει την εκτέλεση Linux πυρήνα απευθείας στα Windows, εξασφαλίζεται ευκολία στη διαχείριση του cluster και της εφαρμογής Django, ενώ η χρήση εργαλείων όπως το kubectl καθιστά εύκολη την παρακολούθηση και τον έλεγχο των pods και υπηρεσιών. Αυτό το περιβάλλον προσφέρει μια

```

iasonas@DESKTOP-9M04JK8: ~/SDN_Django_framework_for_implementation_network_service_configuration_applica
iasonas@DESKTOP-9M04JK8: ~/SDN_Django_framework_for_implementation_network_service_configuration_applica
iasonas@DESKTOP-9M04JK8: ~/SDN_Django_framework_for_implementation_network_service_configuration_applica
iasonas@DESKTOP-9M04JK8: ~/SDN_Django_framework_for_implementation_network_service_configuration_applica
ge list
REPOSITORY    TAG        IMAGE ID      CREATED       SIZE
django         v1         67c01544160e 27 seconds ago 1.13GB
iasonas@DESKTOP-9M04JK8: ~/SDN_Django_framework_for_implementation_network_service_configuration_applica
iasonas@DESKTOP-9M04JK8: ~/SDN_Django_framework_for_implementation_network_service_configuration_applica
iasonas@DESKTOP-9M04JK8: ~/SDN_Django_framework_for_implementation_network_service_configuration_applica

```

Σχήμα 7.4: Docker image list

ολοκληρωμένη εμπειρία ανάπτυξης και δοκιμής, βοηθώντας στην κατανόηση των αρχών του Kubernetes και διευκολύνοντας τη μετάβαση της εφαρμογής σε μεγαλύτερα production περιβάλλοντα

Το Minikube είναι ένα εργαλείο που απλοποιεί την εκτέλεση και διαχείριση ενός τοπικού Kubernetes cluster στον υπολογιστή σας, ειδικά σχεδιασμένο για περιβάλλοντα ανάπτυξης και δοκιμών. Σας επιτρέπει να ξεκινήσετε ένα Kubernetes cluster με ένα μόνο κόμβο (ή ακόμα και πολλούς σε ορισμένες περιπτώσεις) χρησιμοποιώντας εικονικοποίηση μέσω WSL, Docker, ή Hypervisor. Ο κύριος σκοπός του Minikube είναι να παρέχει ένα περιβάλλον Kubernetes με όλες τις βασικές δυνατότητες του Kubernetes αλλά χωρίς την πολυπλοκότητα που θα απαιτούσε η διαχείριση ενός cluster σε παραγωγικό περιβάλλον.

Επιπλέον, το Minikube διαθέτει ενσωματωμένα εργαλεία, όπως τη δυνατότητα να παρακολουθείτε και να διαχειρίζεστε τον πίνακα ελέγχου του Kubernetes, να δημιουργείτε pods, deployments, και services, και να παρακολουθείτε τα logs των εφαρμογών σας, ενώ επιτρέπει επίσης εύκολη σύνδεση με εργαλεία όπως το kubectl για πλήρη πρόσβαση στη διαχείριση του cluster. Αυτό το καθιστά ιδανικό για προγραμματιστές που θέλουν να πειραματιστούν με Kubernetes, να κάνουν δοκιμές εφαρμογών, ή να αναπτύξουν μικροϋπηρεσίες τοπικά χωρίς να απαιτείται η πολυπλοκότητα ενός πλήρους cluster όπως το περιβάλλον της παρούσας διπλωματικής εργασίας. Παρακάτω μπορούμε να δούμε πόσο εύκολα μπορούμε να ξεκινήσουμε ένα τοπικό περιβάλλον κυβερνήτη.

Στο πλαίσιο της ανάπτυξης της εφαρμογής, χρησιμοποιήθηκε το Kubernetes για τη δημιουργία και διαχείριση ενός pod που φιλοξενεί την εφαρμογή Django. Το αρχείο διαμόρφωσης YAML (pod1.yml) που δημιουργήθηκε, ακολουθεί τη βασική δομή του Kubernetes, ορίζοντας τον τύπο πόρου ως Pod και εκχωρώντας μεταδεδομένα όπως το όνομα djangotestapp. Στην ενότητα spec, ορίζεται ένα container το οποίο χρησιμοποιεί την εικόνα iasonasi/djangotestapp:latest και ακούει στη θύρα 8000, η οποία είναι η προεπιλεγμένη θύρα της εφαρμογής Django.

Αυτό το παράδειγμα αποδεικνύει τη σημασία της χρήσης του Kubernetes YAML syntax για την αυτοματοποιημένη ανάπτυξη και διαχείριση containerized εφαρμογών. Μέσω αυτής της διαδικασίας, η εφαρμογή μπορεί να επεκταθεί εύκολα σε διάφορα περιβάλλοντα και να κλιμακωθεί ανάλογα με τις ανάγκες. Το συγκεκριμένο αρχείο YAML επιτρέπει στο Kubernetes να εκτελέσει και να διαχειριστεί το pod με

```

iasonas@DESKTOP-9M04JK8:~$ minikube start
🐳 minikube v1.34.0 on Ubuntu 22.04 (amd64)
🔔 Kubernetes 1.31.0 is now available. If you would like to upgrade, specify: --kubernetes-version=v
👉 Using the docker driver based on existing profile
👉 Starting "minikube" primary control-plane node in "minikube" cluster
📦 Pulling base image v0.0.45 ...
🔥 docker "minikube" container is missing, will recreate.
🔥 Creating docker container (CPUs=2, Memory=3900MB) ...
❗ Image was not built for the current minikube version. To resolve this you can delete and recreate the cluster using the latest images. Expected minikube version: v1.30.1 -> Actual minikube version: v1.34.0
🌐 Preparing Kubernetes v1.26.3 on Docker 23.0.2 ...
  ▪ Generating certificates and keys ...
  ▪ Booting up control plane ...
  ▪ Configuring RBAC rules ...
🔗 Configuring bridge CNI (Container Networking Interface) ...
🔍 Verifying Kubernetes components...
  ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟 Enabled addons: storage-provisioner, default-storageclass
🎉 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
iasonas@DESKTOP-9M04JK8:~$ kubectl get pods -A

```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	coredns-787d4945fb-2kwmn	1/1	Running	0	11s
kube-system	coredns-787d4945fb-htlz6	1/1	Running	0	11s
kube-system	etcd-minikube	1/1	Running	0	27s
kube-system	kube-apiserver-minikube	1/1	Running	0	26s
kube-system	kube-controller-manager-minikube	1/1	Running	0	25s
kube-system	kube-proxy-c4rzr	1/1	Running	0	11s
kube-system	kube-scheduler-minikube	1/1	Running	0	26s
kube-system	storage-provisioner	1/1	Running	0	23s

Σχήμα 7.5: Minikube deployment

τρόπο ανεξάρτητο από το υποκείμενο σύστημα, εξασφαλίζοντας επαναληψιμότητα και δυνατότητα μεταφοράς του συστήματος σε διαφορετικές υποδομές. Στην παρακάτω εικόνα φαίνεται το YAML file που χρησιμοποιήθηκε. Φυσικά ολόκληρος ο κώδικας βρίσκεται ανοιχτός και προσβάσιμος στο GITHUB

```

iasonas@DESKTOP-9M04JK8:~$
iasonas@DESKTOP-9M04JK8:~$
iasonas@DESKTOP-9M04JK8:~$ kubectl get pods -A
NAMESPACE      NAME                                     READY   STATUS    RESTARTS   AGE
default         djangotestapp                         1/1     Running   0           15m
kube-system     coredns-787d4945fb-2kwmn             1/1     Running   0           16m
kube-system     coredns-787d4945fb-htlz6             1/1     Running   0           16m
kube-system     etcd-minikube                         1/1     Running   0           16m
kube-system     kube-apiserver-minikube               1/1     Running   0           16m
kube-system     kube-controller-manager-minikube      1/1     Running   0           16m
kube-system     kube-proxy-c4rzr                     1/1     Running   0           16m
kube-system     kube-scheduler-minikube               1/1     Running   0           16m
kube-system     storage-provisioner                   1/1     Running   1 (15m ago)  16m
iasonas@DESKTOP-9M04JK8:~$
iasonas@DESKTOP-9M04JK8:~$
iasonas@DESKTOP-9M04JK8:~$
iasonas@DESKTOP-9M04JK8:~$ cat SDN_Django_framework_for_implementation_network_service_configuration_app
od1.yml
apiVersion: v1
kind: Pod
metadata:
  name: djangotestapp
spec:
  containers:
  - name: djangotestapp
    image: iasonasi/djangotestapp:latest
    ports:
    - containerPort: 8000
iasonas@DESKTOP-9M04JK8:~$

```

Σχήμα 7.6: Django Deploy

Κεφάλαιο 8

Συμπέρασμα

8.1 Εισαγωγή

Μετά από δύομιση μήνες εντατικής δουλειάς, μελέτης και προγραμματισμού για αυτή τη διπλωματική εργασία, είμαι πλέον σίγουρος ότι έχω επιτύχει τους περισσότερους από τους στόχους που έθεσα στην αρχή.

Επιπλέον, η αναζήτηση της κατάλληλης έκδοσης Cisco IOU ήταν μια χρονοβόρα διαδικασία, καθώς οι εκδόσεις αυτές δεν είναι δημόσια διαθέσιμες δωρεάν. Ακόμα και αν κάποιος πληρώσει για αυτές, η εύρεση της κατάλληλης έκδοσης K-9 που είναι συμβατή με το GNS3 που χρησιμοποιούσα, αποδείχθηκε μια πρόκληση που μου κόστισε πολύτιμο χρόνο.

Η εκμάθηση προγραμματισμού με Python είναι συχνά δύσκολη, ιδιαίτερα για όσους είναι νέοι στην πληροφορική. Παρόλο που η Python είναι δημοφιλής σε τομείς όπως η ανάπτυξη ιστοσελίδων, η επιστήμη δεδομένων και η τεχνητή νοημοσύνη, η κατανόηση της σύνταξης, των δομών δεδομένων και των ελεγκτικών δομών απαιτεί χρόνο. Το να μάθει κανείς προγραμματισμό σημαίνει επίσης ότι χρειάζεται λογική και αναλυτική σκέψη, που εμπλέκει δεξιότητες επίλυσης προβλημάτων και κριτική σκέψη.

Παρά τα εμπόδια, κατάφερα να ολοκληρώσω την εργασία μου, και διαπίστωσα ότι η υπέρβαση των προκλήσεων με γέμισε ικανοποίηση και ενθουσιασμό. Η δημιουργία της δικτυακής τοπολογίας με οδήγησε στο να μελετήσω περισσότερο για την ασφάλεια των θυρών και τα δυναμικά πρωτόκολλα δρομολόγησης, διευρύνοντας έτσι τις γνώσεις μου στα συστήματα δικτύων. Η αντιμετώπιση προβλημάτων στις διαμορφώσεις των συσκευών με βοήθησε να ανακεφαλαιώσω όσα έχω μάθει.

Η μελέτη αυτοματοποίησης δικτύου με Python μπορεί να είναι καθοριστική για άτομα στον χώρο της δικτύωσης. Καθώς πολλές εταιρείες υιοθετούν λύσεις αυτοματοποίησης και δικτύωσης οριζόμενης από λογισμικό (SDN), η ζήτηση για μηχανικούς δικτύων με δεξιότητες αυτοματοποίησης αυξάνεται. Η Python, μια γλώσσα που προσφέρει εργαλεία και βιβλιοθήκες για αυτοματοποίηση, είναι πλέον η πιο προτιμώμενη γλώσσα για τον σκοπό αυτό.

Με τη μελέτη της αυτοματοποίησης δικτύου στην Python, μπορώ να βελτιώσω

τις ικανότητές μου με πολλούς τρόπους. Πρώτον, μου επιτρέπει να αυτοματοποιώ επαναλαμβανόμενες εργασίες δικτύου, όπως η διαμόρφωση συσκευών, η διαχείριση υποδομών δικτύου και η παρακολούθηση της απόδοσης του δικτύου, εξοικονομώντας χρόνο και μειώνοντας τον κίνδυνο ανθρώπινων λαθών. Δεύτερον, με βοηθά να κατανοήσω καλύτερα τα πρωτόκολλα και τις τεχνολογίες δικτύων. Μέσω ανάπτυξης σεναρίων και εργαλείων αυτοματοποίησης, μπορώ να αποκτήσω βαθύτερη γνώση του τρόπου λειτουργίας των δικτύων και των διαφορών που υπάρχουν στα πρωτόκολλα.

Η Python παρέχει εξαιρετική ευελιξία και προσαρμοστικότητα, επιτρέποντάς μου να αναπτύξω λύσεις αυτοματοποίησης προσαρμοσμένες στις ιδιαίτερες ανάγκες του οργανισμού μου. Αυτή η προσέγγιση απαιτεί βαθιά κατανόηση της αρχιτεκτονικής του δικτύου, των πρωτοκόλλων, καθώς και των θεμελιωδών αρχών του προγραμματισμού, ώστε να διασφαλιστεί η ακρίβεια και η αποδοτικότητα των αυτοματοποιημένων διαδικασιών.

Επιπλέον, η εργασία διατίθεται ως εφαρμογή ανοιχτού κώδικα στο GitHub, γεγονός που ενθαρρύνει τη συνέχιση της ανάπτυξης και βελτίωσής της από την κοινότητα. Αυτό το χαρακτηριστικό ενισχύει τη συνεργατικότητα και την ανταλλαγή ιδεών, καθιστώντας δυνατή την εξέλιξή της μέσα από την ενεργή συμμετοχή άλλων ενδιαφερόμενων προγραμματιστών.

Όπως αναφέρεται και στην εισαγωγή της διπλωματικής, η εφαρμογή έχει δυνατότητες περαιτέρω ανάπτυξης, χωρίς περιορισμούς. Στόχος ήταν να δημιουργηθεί ένα εργαλείο που, βασισμένο σε βασικές αρχές της επιστήμης των υπολογιστών, συνδυάζει διαφορετικές διαστάσεις της επιστήμης, προωθώντας την καινοτομία στη δικτύωση και την αυτοματοποίηση. Αυτή η προσέγγιση προσδίδει στην εφαρμογή έναν δυναμικό χαρακτήρα, επιτρέποντας της να αναπτυχθεί στο μέλλον με νέες λειτουργίες και δυνατότητες.

Κεφάλαιο 9

Βιβλιογραφία

9.1 Βιβλιογραφικές αναφορές

- <https://kubernetes.io/docs/concepts/overview/>
- Vs Code development environment
- <https://el.wikipedia.org/wiki/>
- Virtual box ή οποιονδήποτε type B hypervisor
- <https://www.cloudflare.com/learning/network-layer/what-is-the-control-plane/>
- <https://el.wikipedia.org/wiki/Git>
- <https://kubernetes.io>
- <https://github.com/dmfigol/network-programmability-stream>
- <https://medium.com/@komalminhas.96/a-step-by-step-guide-to-build-and-push-your-own-docker-images-to-dockerhub-709963d4a8bc>
- <https://repository.ihu.edu.gr/>, Network automation using python George Milios
- Wikipedia, τι είναι το Windows Subsystem for Linux
- wikipedia.org/wiki/Modelviewcontroller