ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη εφαρμογής διαχείρισης και αυτοματισμού δικτύων»

Του φοιτητή
Μιχαήλ Τριχάκη
Αρ. Μητρώου: 154551

Επιβλέπων
Δρ. Αντώνιος Σαρηγιαννίδης

15 Ιουνίου 2022

Τίτλος Δ.Ε.: Ανάπτυξη εφαρμογής διαχείρισης και αυτοματισμού δικτύων
Κωδικός Δ.Ε.: 21244
Φοιτητής: Μιχαήλ Τριχάκης
Εισηγητής: Δρ. Αντώνιος Σαρηγιαννίδης
Ημερομηνία ανάληψης Δ.Ε.: 01-04-2021
Ημερομηνία περάτωσης Δ.Ε.: 16-06-2022

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή Μιχαήλ Τριχάκη που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης, ο συγγραφέας/δημιουργός εκχωρεί στο Διεθνές Πανεπιστήμιο της Ελλάδος άδεια χρήσης του δικαιώματος αναπαραγωγής, δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσης της εργασίας διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος. Η ανοικτή πρόσβαση στο πλήρες κείμενο της εργασίας, δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού, ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, πώληση, εμπορική χρήση, διανομή, έκδοση, μεταφόρτωση (downloading), ανάρτηση (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Περίληψη

Η παρούσα διπλωματική περιλαμβάνει την ανάπτυξη ενός εργαλείου διαχείρισης και αυτοματισμού δικτύων το οποίο ονομάζεται AutoNet. Το εργαλείο αυτό θα προσφέρει την δυνατότητα στον διαχειριστή του δικτύου να συνδεθεί μέσω μιας web-based εφαρμογής και να διαχειρίζεται τις δικτυακές συσκευές που επιθυμεί. Στόχος της εργασίας είναι η διευκόλυνση των διαχειριστών στη διαχείριση και στη συντήρηση μικρών και μεγάλων δικτύων. Θα παρέχεται η δυνατότητα στον διαχειριστή να παρακολουθεί σε πραγματικό χρόνο την κατάσταση των δικτυακών συσκευών Cisco και να λαμβάνει άμεσες αποφάσεις σχετικά με τυχόν προβλήματα που μπορεί να προκύψουν στο δίκτυο. Επιπλέον, ο διαχειριστής θα μπορεί να παρακολουθεί την κίνηση από και προς τις συσκευές μέσω διαγραμματικής αναπαράστασης σε πραγματικό χρόνο. Τέλος, θα παρέχεται η δυνατότητα στον διαχειριστή να παραμετροποιεί τις δικτυακές συσκευές μέσω ενός γραφικού και ευέλικτου περιβάλλοντος.

Λέξεις Κλειδιά: διαχείριση δικτύων, διαδικτυακή εφαρμογή, δικτυακές συσκευές CISCO

«Development of network management and automation application»

«Michail Trichakis»

Abstract

This thesis includes the development of a network management and automation tool called AutoNet. With this tool, the network administrator will have the ability to connect through a web-based application and manage the devices of a network. The purpose of this thesis is to facilitate the process of management and maintenance of small and large networks. Furthermore, the network administrator will be able to monitor in real time the status of Cisco network devices and make direct decisions about any issue that may arise. In addition, all the traffic from the devices will be displayed in a real-time data diagram. Finally, the administrator will be able to configure the network devices through a graphical and user-friendly environment.

Keywords: network management, web-based application, network devices CISCO

Ευχαριστίες

Με την περάτωση της παρούσας διπλωματικής εργασίας, θα ήθελα να ευχαριστήσω θερμά τον Δρ. Αντώνη Σαρηγιαννίδη, καθώς η αιτία που ασχολήθηκα και με τα δίκτυα, εκτός του προγραμματισμού, είναι το ζεύγος μαθημάτων «Ειδικά Θέματα Δικτύων Ι και ΙΙ» τα οποία παρακολούθησα πετυχημένα. Τέλος, τον ευχαριστώ θερμά για τις χρήσιμες συμβουλές δικτύων καθώς και για την καθοδήγηση πάνω στην χρήση δικτυακών πρωτοκόλλων που χρησιμοποιήθηκαν στην παρούσα εργασία.

Περιεχόμενα

Κατάλογος Σχημάτων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
CDP	Cisco Discovery Protocol
SNMP	Simple Network Management Protocol
ARP	Address Resolution Protocol
ACL	Access Control List
ACLs	Access Control Lists
IP	Internet Protocol
TCP	Transmission Control Protocol
EIGRP	Enhanced Interior Gateway Routing Protocol
OSPF	Open Shortest Path First
SSH	Secure Shell
EEM	Embedded Event Manager
JS	JavaScript
Mongo	MongoDB

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Τα τελευταία χρόνια τα δίκτυα υπολογιστών έχουν γνωρίσει μεγάλη ανάπτυξη με ανάλογη αύξηση της πολυπλοκότητας τους. Η πολυπλοκότητα συνίσταται κυρίως στην υποστήριξη περισσότερων χρηστών αλλά και περισσοτέρων και ετερογενών εφαρμογών οι οποίες δημιουργούν τόσο κίνηση δεδομένων όσο και κίνηση φωνής (Voice over IP) [1]. Επίσης, η πολυπλοκότητα των σύγχρονων δικτύων εντοπίζεται και στη χρήση πολλών δικτυακών συσκευών αλλά και στη χρήση διαφορετικών συσκευών με ανόμοιες απαιτήσεις και δυνατότητες. Ένα σύγχρονο εταιρικό δίκτυο αποτελείται από εκατοντάδες συσκευές, όπου η εξατομικευμένη διαχείριση και παρακολούθηση θεωρείται εξαιρετικά δύσκολη και χρονοβόρα. Πέρα από τον πολύ μεγάλο αριθμό των συσκευών, ένα εταιρικό δίκτυο αποτελείται από μεγάλο αριθμό διαφορετικών συσκευών όπως υπολογιστές, δρομολογητές (Routers), μεταγωγείς (Switches), περιφερειακές συσκευές, τείχη προστασίας (Firewalls), διακομιστές (Servers) κ.α. Αυτές οι συσκευές μπορεί να προέρχονται από διαφορετικούς κατασκευαστές, που σημαίνει, διαφορετικά χαρακτηριστικά και απαιτήσεις. Δεδομένη θεωρείται, σε τόσο μεγάλο αριθμό συσκευών, η αστοχία υλικού που μπορεί πλήξει σημαντικά τη διαθεσιμότητα του δικτύου. Συνεπώς, έχει αυξηθεί σημαντικά η πιθανότητα να συμβεί κάποια αστοχία και έτσι ολόκληρο το δίκτυο ή ένα μέρος του να τεθεί εκτός λειτουργίας ή να μειωθεί η αξιοπιστία και η απόδοση του. Επομένως, η παρακολούθηση, η συντήρηση και ο έλεγχος ενός τέτοιου δικτύου μπορεί να είναι μια διαδικασία ασύμφορη, επίπονη και χρονοβόρα, που απαιτεί να ασχοληθούν αρκετοί άνθρωποι.

Ένας άλλος πολύ σημαντικός παράγοντας, που ενισχύει την πολυπλοκότητα των δικτύων, είναι η δυναμική φύση των δικτύων. Οι αλλαγές τόσο στη τοπολογία όσο και στις ενεργές συσκευές αλλά και στη λειτουργία των σύγχρονων δικτύων καθιστούν απαραίτητη την αποδοτική διαχείριση τους. Σε ένα εταιρικό δίκτυο είναι πολύ συχνές οι προσθήκες, αφαιρέσεις και τροποποιήσεις συσκευών με αποτέλεσμα να αλλάξει τόσο η τοπολογία όσο και λειτουργία του δικτύου.

Με βάση τα παραπάνω, κρίνεται απαραίτητη η χρήση εξελιγμένων, ευέλικτων και αποδοτικών εργαλείων διαχείρισης, ελέγχου, παρακολούθησης και παρέμβασης σε πραγματικό χρόνο [2]. Διαχείριση δικτύου ορίζεται η διαδικασία της κεντρικής και αυτοματοποιημένης παρακολούθησης και ελέγχου ενός δικτύου υπολογιστών με σκοπό την αύξηση της διαθεσιμότητας των υπηρεσιών του δικτύου αλλά και την διατήρησης της αξιόπιστης λειτουργίας του ακόμη και κάτω από συνθήκες υπερφόρτωσης ή βλάβης. Απώτερος στόχος της διαχείρισης δικτύου είναι η αύξηση της ποιότητας υπηρεσιών που θα παρέχονται στους χρήστες του δικτύου.

Η διαχείριση των δικτύων είναι ικανή να συμβάλει σε πολλαπλούς και κρίσιμους τομείς όπως:

- Η ασφάλεια. Ίσως είναι από τους σημαντικότερους παράγοντες στα σύγχρονα δίκτυα. Ένα εργαλείο διαχείρισης δικτύων θα μπορούσε να παρέχει ενδείξεις για απρόσμενες λειτουργίες συσκευών. Για παράδειγμα, μπορεί να εντοπίσει υπερβολική, απροσδόκητη κίνηση σε μια ή περισσότερες δικτυακές συσκευές. Επίσης, θα μπορούσε να εντοπίσει αδικαιολόγητες καθυστερήσεις στο δίκτυο ή ακόμα δυσλειτουργίες πρωτοκόλλων. Σε κάθε περίπτωση, τα εργαλεία διαχείρισης προβλέπουν την έγκαιρη ειδοποίηση του διαχειριστή δικτύου [3].
- Η απομακρυσμένη πρόσβαση. Μια σημαντική λειτουργία η οποία αποτελεί ένα από τα δυνατά σημεία ενός διαχειριστικού προγράμματος δικτύων. Με δεδομένο ότι υπάρχει η ανάγκη χρήσης πολλών συσκευών σε ένα εταιρικό δίκτυο, υπάρχει και η ανάγκη συντήρησης και αποσφαλμάτωσης όλων αυτών των συσκευών. Έτσι ένα διαχειριστικό εργαλείο δικτύων θα μπορούσε να παρέχει την δυνατότητα της απομακρυσμένης πρόσβασης σε όλες τις συσκευές

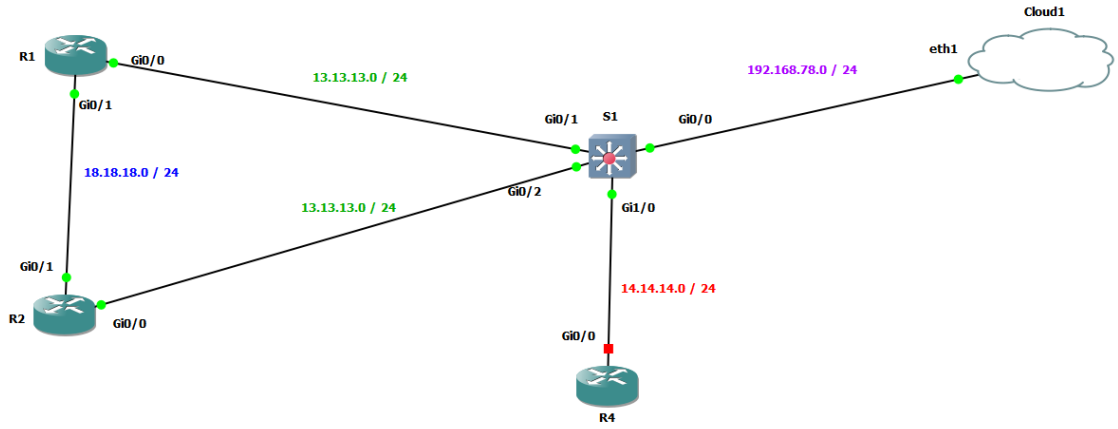
που επιθυμεί ο διαχειριστής, εξοικονομώντας χρόνο, χωρίς να χρειάζεται να συνδέεται σε κάθε μια συσκευή χειροκίνητα.

- Η απόδοση. Μια σημαντική λειτουργία των διαχειριστικών εργαλείων είναι η δυναμική συλλογή στοιχείων που αφορούν την απόδοση των συσκευών [4]. Με αυτόν τον τρόπο, ο διαχειριστής του δικτύου έχει πλήρη εικόνα για την λειτουργία των συσκευών και έχει τη δυνατότητα να παρέμβει αν διαπιστώσει ότι κάτι δεν λειτουργεί όπως θα έπρεπε. Ακόμα, έχει τη δυνατότητα να αυτοματοποιήσει κάποιες ενέργειες.
- Η διαμόρφωσης και η διαχείριση της συσκευής. Με την χρήση ενός προγράμματος διαχείρισης δικτύων ο διαχειριστής θα μπορεί να παραμετροποιήσει τις συσκευές με βάση τους στόχους που θέλει να πετύχει όπως και να βλέπει διάφορες πληροφορίες της συσκευής. Για παράδειγμα ο διαχειριστής θα μπορεί να προβάλει τον πίνακα δρομολόγησης της συσκευής, θα μπορεί να ορίσει καινούργια VLANs καθώς και να παραμετροποιήσει πρωτόκολλα δρομολόγησης.

Στα πλαίσια της παρούσας Δ.Ε. σχεδιάστηκε και υλοποιήθηκε ένα εργαλείο διαχείρισης δικτύων με πολλές και χρήσιμες λειτουργίες που ανταποκρίνονται στις ανάγκες διαχείρισης των σύγχρονων δικτύων. Η εφαρμογή ονομάζεται AutoNet και εκτός των άλλων παρέχει ένα γραφικό και φιλικό προς τον χρήστη περιβάλλον. Το AutoNet ουσιαστικά είναι μια διαδικτυακή πλατφόρμα που προσφέρει πολυεπίπεδες λειτουργίες διαχείρισης δικτύου. Συγκεκριμένα, παρέχει παρακολούθηση και έλεγχο των συσκευών του δικτύου σε πραγματικό χρόνο. Προσφέρει γραφική απεικόνιση της εισερχόμενης και εξερχόμενης κίνησης για τις συσκευές που διαχειρίζεται, καθώς επίσης και εικονικοποίηση της τοπολογίας που θα επιλέξει ο διαχειριστής του δικτύου [5]. Επιπρόσθετα, το AutoNet παρέχει στον διαχειριστή πληροφορίες για κάθε συσκευή αλλά και για όλα τα πρωτόκολλα που λειτουργούν στο δίκτυο από το φυσικό επίπεδο μέχρι το επίπεδο μεταφοράς. Τέλος, προσφέρει ένα γραφικό και εύχρηστο περιβάλλον για την απομακρυσμένη παραμετροποίηση των συσκευών.

1.2 Ανάλυση Εργασίας

Ο στόχος της διπλωματικής εργασίας είναι να δημιουργηθεί μια διαδικτυακή πλατφόρμα (web εφαρμογή) μέσα από την οποία οι διαχειριστές δικτύων θα μπορούν να διαχειριστούν τις δικτυακές τους συσκευές. Πιο συγκεκριμένα, οι συσκευές αυτές είναι της Cisco και έχουν λειτουργικό σύστημα Cisco IOS. Για την υλοποίηση αυτής της εργασίας χρειάστηκαν διάφορες δικτυακές συσκευές προκειμένου να μελετηθούν και να συμμετέχουν στα δοκιμαστικά πειράματα. Επίσης, χρησιμοποιήθηκε η πλατφόρμα GNS3. Μέσω αυτής της πλατφόρμας έγινε εφικτή η δημιουργία των εικονικών δικτύων. Στο Σχήμα 1.1 παρουσιάζονται οι δικτυακές συσκευές, καθώς και οι συνδέσεις μεταξύ τους όπως και τα δίκτυα που δημιουργούν μέσω του GNS3. Συγκεκριμένα η τοπολογία αποτελείται από μια συσκευή Cisco switch layer 2 όπου το λειτουργικό του σύστημα είναι το Cisco IOSv 15.2. Αυτή η συσκευή συνδέεται με το Cloud1, που παρέχεται από το GNS3, για να μπορεί το εικονικό δίκτυο να επικοινωνεί με τον ηλεκτρονικό υπολογιστή, ο οποίος έχει εγκατεστημένο το GNS3. Επίσης, οι δρομολογητές που χρησιμοποιήθηκαν έχουν λειτουργικό σύστημα Cisco IOSv 15.6.



Σχήμα 1.1: Αναπαράσταση του δικτύου των δοκιμών μέσα από το GNS3

Η κύρια ιδέα στην οποία βασίζεται η πλατφόρμα που δημιουργήθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας, αναλύεται ακολούθως. Αρχικά ο κάθε διαχειριστής μπορεί να έχει τον δικό του χρήστη. Ο κάθε χρήστης ανάλογα με τα δικαιώματα που έχει μπορεί να προβάλει και να τροποποιήσει τον τρόπο λειτουργίας ορισμένων συσκευών. Οι συσκευές που εισάγονται στην πλατφόρμα μπορεί να είναι οποιεσδήποτε Cisco συσκευές, πραγματικές ή εικονικές, routers ή switches. Σε αυτές τις συσκευές ορίζονται τα προαπαιτούμενα, δηλαδή, να έχουν μια έγκυρη IP διεύθυνση με την οποία μπορεί να επικοινωνήσει ο server. Επιπλέον, πρέπει να υπάρχει μια παραμετροποίηση που δέχεται συνδέσεις SSH έκδοσης 2. Τέλος το τελευταίο προαπαιτούμενο είναι να έχει δημιουργηθεί τουλάχιστον ένας χρήστης όπου έχει δικαιώματα privilege level 5.

Οι συσκευές που εισάγονται στην πλατφόρμα, εισάγονται μόνο μια φορά. Δεν υπάρχουν διπλοεγγραφές καθώς η κύρια ιδέα είναι ότι μια συσκευή μπορεί να ελεγχθεί από πολλούς χρήστες. Αυτό συνεπάγεται ότι οι συσκευές μπορεί να γίνουν κοινόχρηστες ανάλογα με τα δικαιώματα που υπάρχουν. Προς το παρόν η ύπαρξη περισσότερων χρηστών δεν υποστηρίζεται. Αναφέρεται όμως σαν κύριος στόχος στο κεφάλαιο 9 στις μελλοντικές αναβαθμίσεις.

Η πλατφόρμα ονομάστηκε AutoNet καθώς συμβολίζει τον ορισμό των αυτοματοποιημένων δικτύων. Το όνομα χρήστη και ο κωδικός του βασικού χρήστη ορίστηκαν να είναι το όνομα της πλατφόρμας δηλαδή το “autonet”. Φυσικά παρέχεται και η δυνατότητα στον χρήστη να τροποποιήσει αυτά τα διαπιστευτήρια.

Τέλος αξίζει να αναφερθεί ότι υλοποιήθηκαν δύο κύριοι στόχοι. Ο πρώτος ήταν να υλοποιηθεί μια λειτουργία ώστε να παρακολουθείται εύκολα και κατανοητά η κίνηση μιας συσκευής προκειμένου να παρθούν γρήγορες και καθοριστικές αποφάσεις. Για παράδειγμα αν παρατηρήσει ο χρήστης – διαχειριστής δικτύων, ότι η κίνηση μιας συσκευής αυξάνεται ασυνήθιστα πολύ, τότε μπορεί να παρέμβει άμεσα μέσω της πλατφόρμας. Ο δεύτερος στόχος βασίστηκε στον πρώτο, συγκεκριμένα ο διαχειριστής μπορεί γρήγορα να συνδεθεί σε όποια συσκευή κρίνει ότι χρειάζεται επέμβαση.

Καθώς η πλατφόρμα θα παρείχε στον χρήστη την δυνατότητα ελέγχου συσκευών, προέκυψε η ανάγκη να παρέχεται εικονικοποίηση των δικτύων για την πιο αποδοτική διαχείριση. Για τον λόγο αυτό δημιουργήθηκε η λειτουργία της παρουσίασης της ενεργής τοπολογίας του δικτύου, δυναμικά, σε πραγματικό χρόνο. Ο κάθε χρήστης μπορεί να επιλέξει ποιες συσκευές θα περιέχει η κάθε τοπολογία ώστε να μπορεί να κατηγοριοποιήσει τις συσκευές όπως ο ίδιος επιθυμεί. Ο χρήστης έχει την

δυνατότητα να μετακινεί, να προσθέτει και να αφαιρεί συσκευές μέσα στην τοπολογία. Μέσα από αυτή τη λειτουργικότητα είναι κατανοητό ότι η κάθε συσκευή θα μπορεί να βρίσκεται σε περισσότερες από μια τοπολογίες. Τέλος, σε κάθε τοπολογία δημιουργούνται αυτόματα οι συνδέσεις μεταξύ των συσκευών και παρουσιάζονται σε πραγματικό χρόνο οι καταστάσεις τους καθώς και η κατάσταση της κάθε συσκευής.

Ολοκληρώνοντας την ενότητα 1.2 θα αναλυθούν όλες οι δυνατότητες που παρέχει το AutoNet. Ένα από τα σημαντικότερα προτερήματα που προσφέρεται στον διαχειριστή δικτύων είναι η παρακολούθηση της κίνησης των συσκευών μέσω γραφημάτων σε πραγματικό χρόνο. Επίσης, η δυνατότητα της άμεσης πρόσβασης στο τερματικό της κάθε συσκευής, αποτελεί μέσο για την υλοποίηση δραστικών μέτρων. Πιο συγκεκριμένα, ο διαχειριστής θα είναι σε θέση να συνδέεται με SSH στα τερματικά των συσκευών που επιθυμεί χωρίς να χρειάζεται να εισάγει τα διαπιστευτήρια του. Επιπρόσθετα, παρέχεται η δυνατότητα προβολής των πινάκων δρομολόγησης, του ARP πίνακα, του πίνακα των γειτόνων CDP, του πίνακα των MAC διευθύνσεων καθώς και τα όλα τα Access Lists που περιέχει η συσκευή. Το AutoNet προσφέρει την δυνατότητα στον χρήστη να παρακολουθεί σημαντικές πληροφορίες της κάθε διεπαφής των συσκευών καθώς επίσης και να προβάλει τα αρχεία παραμετροποίησης όπως και να τα κατεβάσει στον ηλεκτρονικό του υπολογιστή. Τέλος ο κάθε χρήστης που θα εισέρχεται στην πλατφόρμα θα είναι σε θέση να προβάλει πληροφορίες για πρωτόκολλα δρομολόγησης όπως είναι το EIGRP και το OSPF, καθώς επίσης και πληροφορίες του λειτουργικού συστήματος της συσκευής όπως και διάφορες γενικές πληροφορίες (π.χ. το όνομα της συσκευής, το username και το password της συσκευής).

1.3 Οργάνωση Εργασίας

Καθώς η υλοποίηση της πλατφόρμας ήταν αρκετά περίπλοκη, η παρούσα διπλωματική εργασία χρειάστηκε να χωριστεί σε αρκετά κεφάλαια ώστε να γίνει πιο ευκολά κατανοητή. Αρχικά στο πρώτο κεφάλαιο πραγματοποιήθηκε μια εισαγωγή της εργασίας και στο δεύτερο παρουσιάζεται η πλατφόρμα AutoNet. Στο τρίτο κεφάλαιο θα αναλυθούν όλες τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν κατά την υλοποίηση της πλατφόρμας. Ακολούθως θα αφιερωθεί άλλο ένα κεφάλαιο για να αναφερθούν τα δικτυακά πρωτόκολλα και οι τεχνολογίες που χρησιμοποιήθηκαν, καθώς επίσης και για όλη την παραμετροποίηση που έγινε στις συσκευές CISCO. Τέλος, πριν ξεκινήσει η ανάλυση του κώδικα είναι αναγκαίο να αναλυθεί η βάση δεδομένων προκειμένου να κατανοηθούν τα σχήματα των δεδομένων.

Η υλοποίηση του κώδικα χωρίζεται σε τρία μέρη. Το πρώτο μέρος είναι η ανάπτυξη των Python scripts μέσα από τα οποία στέλνονται οι κατάλληλες εντολές στις δικτυακές συσκευές CISCO. Το δεύτερο μέρος αφορά την υλοποίηση του κώδικα της εφαρμογής του back-end. Αυτή η εφαρμογή είναι υπεύθυνη για την επικοινωνία με τις συσκευές, μέσα από την εκτέλεση των python scripts του πρώτου μέρους, την επικοινωνία με την front-end εφαρμογή καθώς επίσης και για την επικοινωνία με την βάση δεδομένων. Η ανάλυση του κώδικα ολοκληρώνεται με την ανάλυση του τρίτου μέρους, το οποίο αφορά την υλοποίηση του κώδικα προκειμένου να αναπτυχθεί η front-end εφαρμογή.

Τέλος εφόσον αναλυθεί ο κώδικας και των τριών κομματιών της διπλωματικής θα αναφερθούν τα συμπεράσματα που εξήχθησαν καθώς επίσης και οι μελλοντικές αναβαθμίσεις που θα πραγματοποιηθούν.

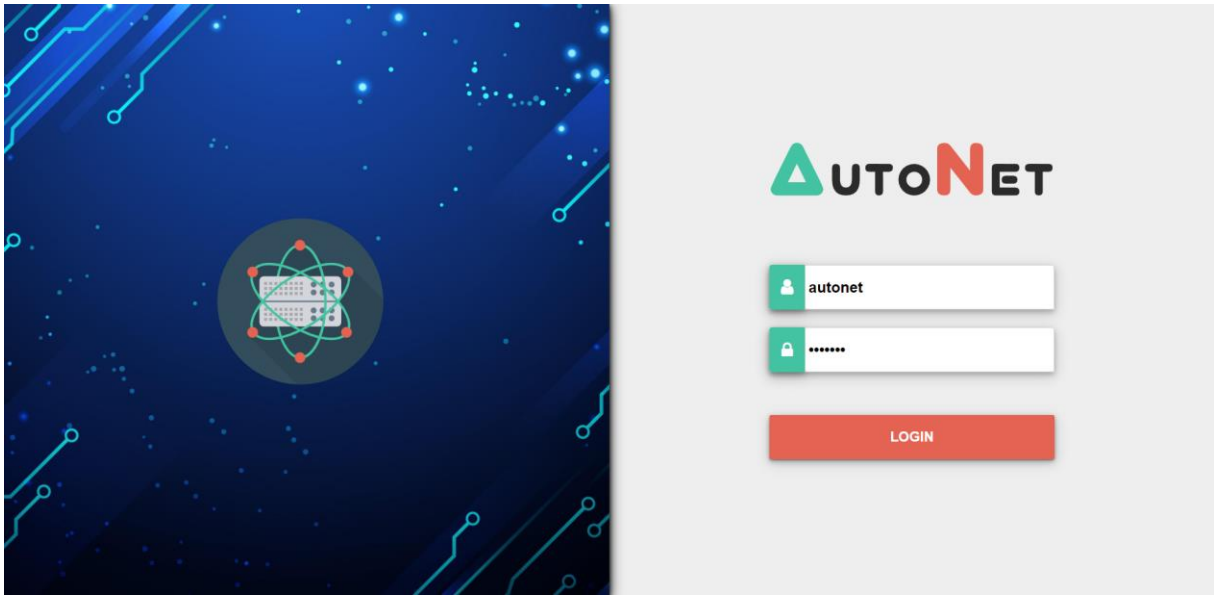
Κεφάλαιο 2ο: Παρουσίαση AutoNet

2.1 Εισαγωγή

Στο παρόν κεφάλαιο θα παρουσιαστεί η πλατφόρμα AutoNet, που αναπτύχθηκε στα πλαίσια της Δ.Ε.

2.2 Σελίδα Σύνδεσης

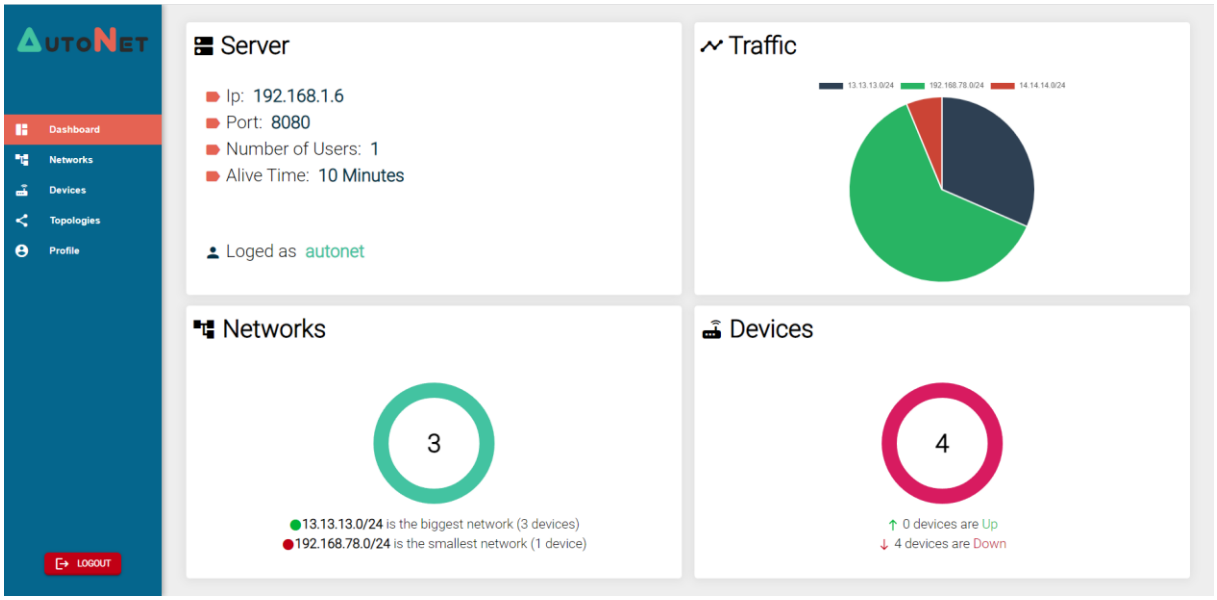
Η πρώτη σελίδα που βλέπει ο χρήστης είναι η σελίδα σύνδεσης. Στην σελίδα σύνδεσης περιέχονται 2 πεδία όπου ο χρήστης εισάγει τα διαπιστευτήριά του. Αφού τα εισάγει μπορεί να πατήσει το κουμπί Login για να συνδεθεί στην πλατφόρμα.



Σχήμα 2.1: Σελίδα σύνδεσης

2.3 Κεντρική Σελίδα

Εφόσον ο χρήστης πραγματοποίησε την σύνδεση στο σύστημα, κατευθύνεται στην κεντρική σελίδα της πλατφόρμας. Αυτή η σελίδα καλείται Dashboard και παρέχει κάποιες γενικές πληροφορίες για την πλατφόρμα καθώς και για τα δεδομένα του χρήστη. Πιο συγκεκριμένα παρέχει 4 πλαίσια με πληροφορίες όπου παρουσιάζονται στο Σχήμα 2.2. Το 1ο πλαίσιο, το οποίο βρίσκεται στην πάνω αριστερή θέση, παρέχει πληροφορίες για τον Server, όπου μεταξύ άλλων είναι ο χρόνος που είναι ενεργός ο server και το όνομα χρήστη που έχει συνδεθεί. Στο 2ο πλαίσιο το οποίο βρίσκεται στην πάνω δεξιά θέση, παρέχονται πληροφορίες για την κίνηση των δικτύων. Τέλος στα 2 τελευταία πλαίσια παρέχονται ενδεικτικές πληροφορίες για το τι συμβαίνει στα δίκτυα και στις συσκευές που επιτρέπεται να βλέπει ο συγκεκριμένος χρήστης.

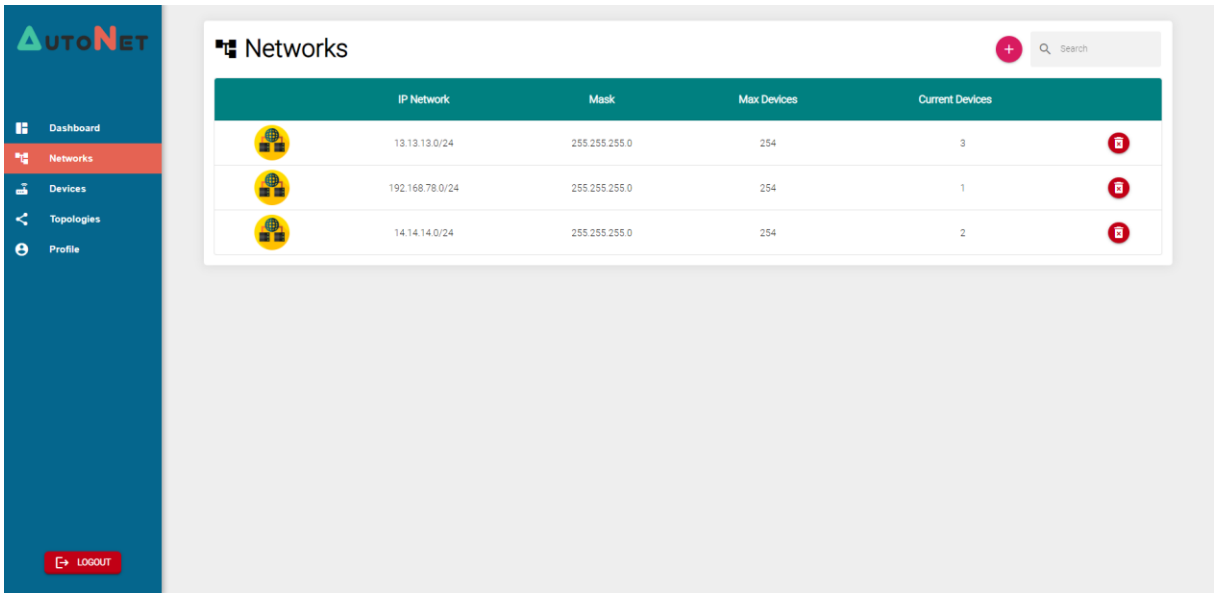


Σχήμα 2.2: Σελίδα δικτύων

Επιπρόσθετα, στο μενού στην αριστερή μεριά του Σχήματος 2.2 ο χρήστης μπορεί να πλοηγηθεί στην εφαρμογή καθώς του δίνεται η δυνατότητα να αποσυνδεθεί από την πλατφόρμα.

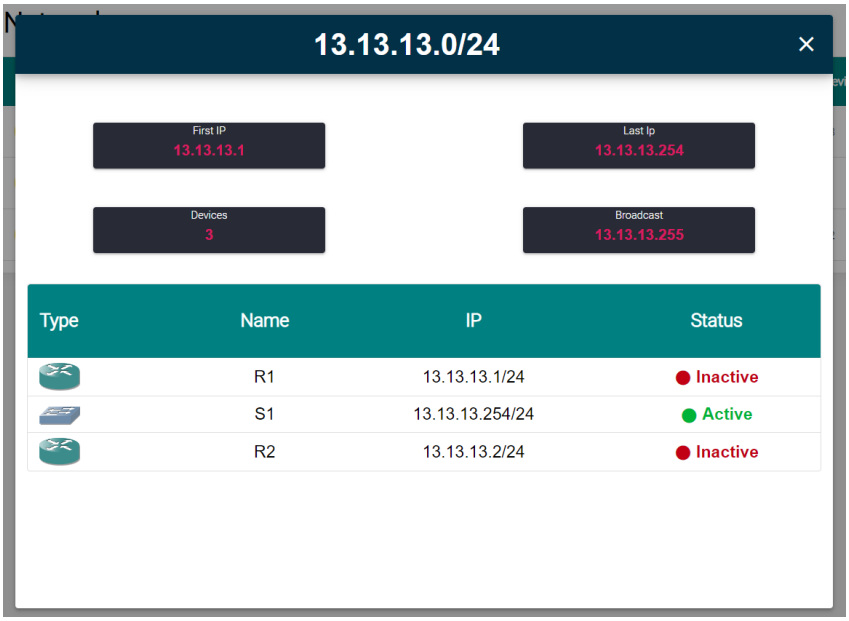
2.4 Σελίδα Δικτύων

Στο Σχήμα 2.3, η σελίδα δικτύων παρέχει έναν πίνακα με όλα τα δίκτυα που μπορεί να έχει πρόσβαση ο χρήστης. Ο χρήστης μπορεί να φιλτράρει αυτά τα δεδομένα χρησιμοποιώντας το πεδίο αναζήτησης το οποίο βρίσκεται στην πάνω δεξιά θέση του Σχήματος 2.3. Επίσης δίνεται η δυνατότητα στον χρήστη να προβάλει περισσότερες πληροφορίες πατώντας πάνω σε ένα δίκτυο καθώς και η δυνατότητα να διαγράψει αυτό το δίκτυο, πατώντας στο κουμπί διαγραφής. Αξίζει να σημειωθεί, ότι όταν διαγράφεται ένα δίκτυο δεν διαγράφονται και οι συσκευές που περιέχονται σε αυτό καθώς μπορεί να περιέχονται και σε άλλα δίκτυα. Τέλος πρέπει να αναφερθεί ότι υπάρχει ένα κουμπί στην πάνω δεξιά θέση του Σχήματος 2.3, όπου επιτρέπει την εισαγωγή νέου δικτύου.



Σχήμα 2.3: Σελίδα δικτύων

Καθώς ο χρήστης πατήσει πάνω σε ένα δίκτυο εμφανίζεται ένα παράθυρο όπου παρουσιάζεται στο Σχήμα 2.4. Σε αυτό το παράθυρο διακρίνονται πληροφορίες όπως η Broadcast IP, η πρώτη IP διεύθυνση που μπορεί να δοθεί σε ένα τερματικό host, καθώς επίσης και πληροφορίες για τις συσκευές που βρίσκονται μέσα στο εκάστοτε δίκτυο. Οι πληροφορίες για αυτές τις συσκευές είναι ο τύπος της συσκευής, το όνομά της, η διεύθυνση IP που διαθέτει για αυτό το συγκεκριμένο δίκτυο και τέλος η κατάσταση της συσκευής.

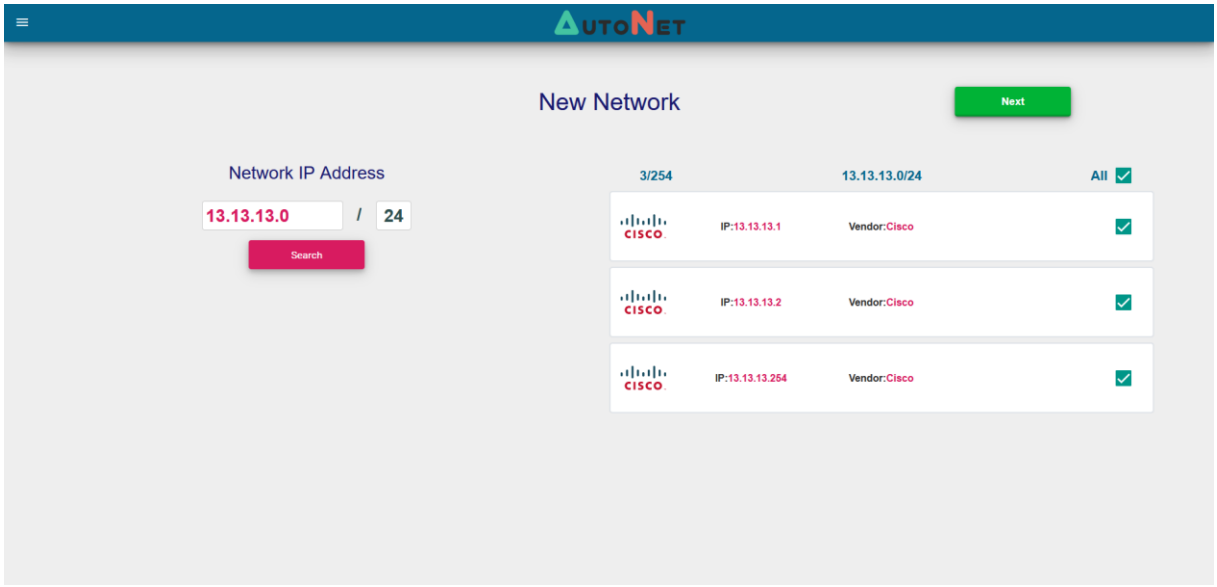


Σχήμα 2.4: Παράθυρο πληροφοριών δικτύου

2.5 Σελίδα Νέου Δικτύου

Μετά το πάτημα του κουμπιού τού νέου δικτύου που παρουσιάστηκε στο Σχήμα 2.3, η εφαρμογή πλοηγείται στην σελίδα νέου δικτύου. Σε αυτή την σελίδα ο χρήστης εισάγει μια διεύθυνση IP με την

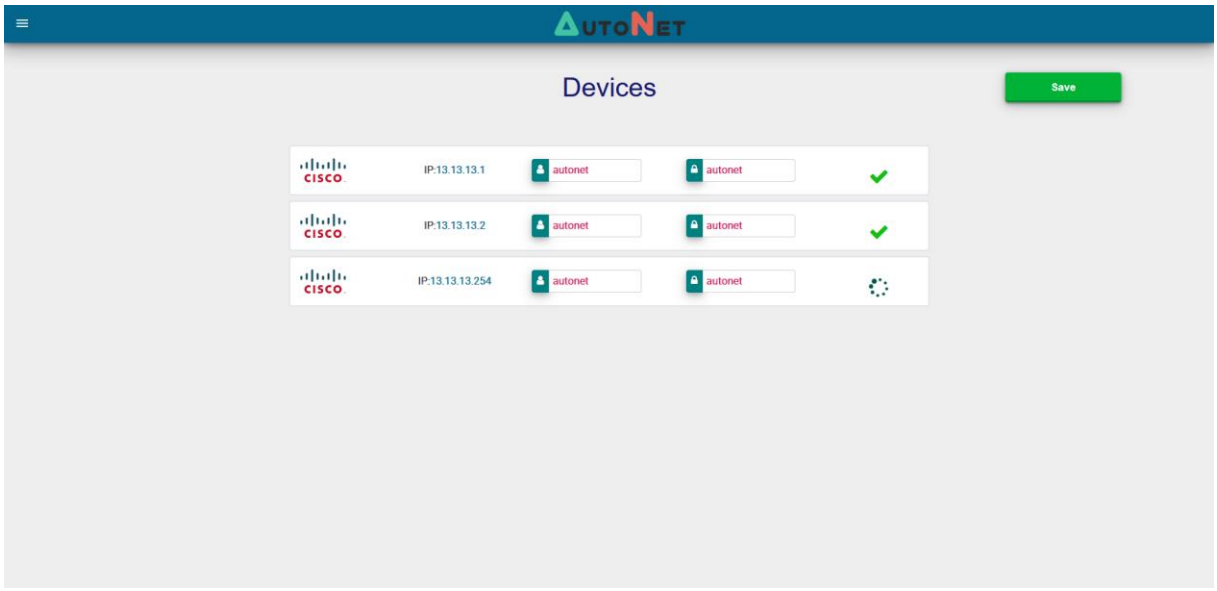
μάσκα της. Αυτή η διεύθυνση μπορεί να είναι είτε μια διεύθυνση δικτύου είτε μια διεύθυνση IP host. Το τι από τα δυο θα δώσει ο χρήστης δεν έχει σημασία για το σύστημα καθώς αν ο χρήστης δώσει μια διεύθυνση IP ενός τερματικού, το back-end application θα βρει την διεύθυνση δικτύου του. Το σύστημα θα ψάξει για όλες τις ενεργές συσκευές, Cisco routers ή Cisco switches, στο δίκτυο που δόθηκε. Ο χρήστης έχει την δυνατότητα να επιλέξει ποιες συσκευές θέλει να εισάγει στο σύστημα. Καθώς επιλέξει τις επιθυμητές συσκευές πατάει το κουμπί Next και πλοηγείται στην σελίδα αποθήκευσης των συσκευών (κεφάλαιο 2.6). Η αναπαράσταση της σελίδας της εισαγωγής νέου δικτύου παρουσιάζεται στο Σχήμα 2.5.



Σχήμα 2.5: Σελίδα νέου δικτύου

2.6 Σελίδα Αποθήκευσης Συσκευών

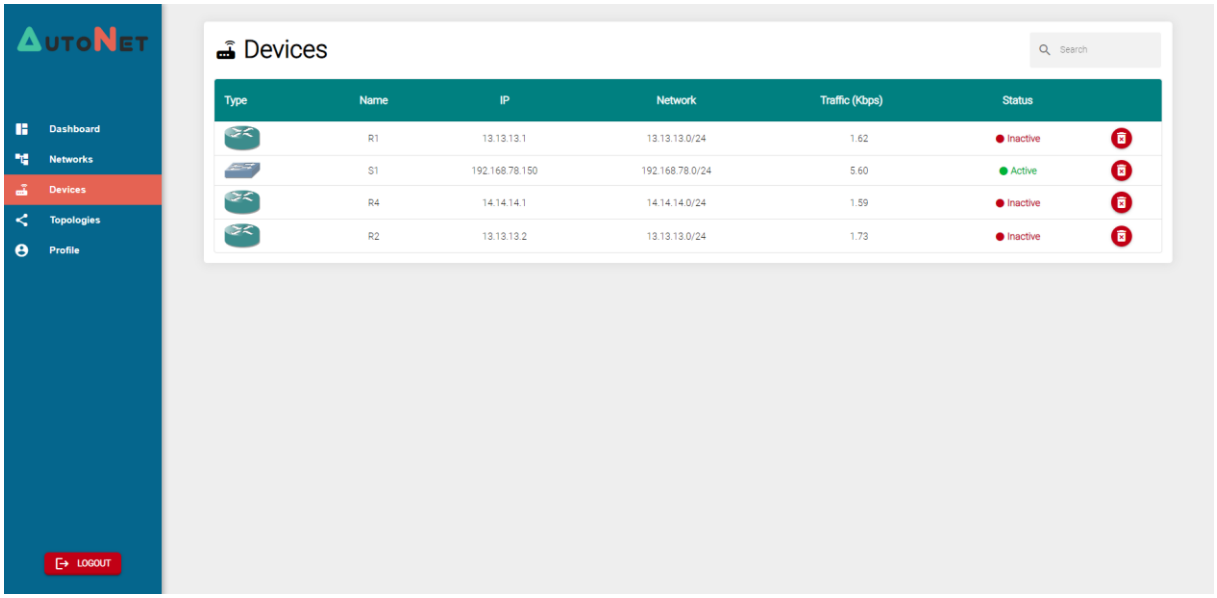
Καθώς έχουν επιλεγεί οι συσκευές που πρόκειται να αποθηκευτούν στην πλατφόρμα, το επόμενο βήμα είναι ο χρήστης να εισάγει τα διαπιστευτήρια των συσκευών. Πιο συγκεκριμένα στην σελίδα αποθήκευσης συσκευών ο χρήστης πρέπει να ορίσει τα διαπιστευτήρια των χρηστών των συσκευών οι οποίοι έχουν τιμή για το privilege level το 15. Όταν ο χρήστης εισάγει τα διαπιστευτήρια μπορεί να πατήσει το κουμπί Save και να ξεκινήσει η διαδικασία αποθήκευσης. Όταν η αποθήκευση μιας συσκευής ολοκληρώνεται με επιτυχία τότε εμφανίζεται μια ένδειξη επιτυχίας στην γραμμή της συσκευής. Σε διαφορετική περίπτωση εμφανίζεται μια ένδειξη αποτυχίας σε εκείνο το σημείο. Η αναπαράσταση της σελίδας αποθήκευσης των συσκευών φαίνεται στο Σχήμα 2.6.



Σχήμα 2.6: Σελίδα αποθήκευσης συσκευών

2.7 Σελίδα Συσκευών

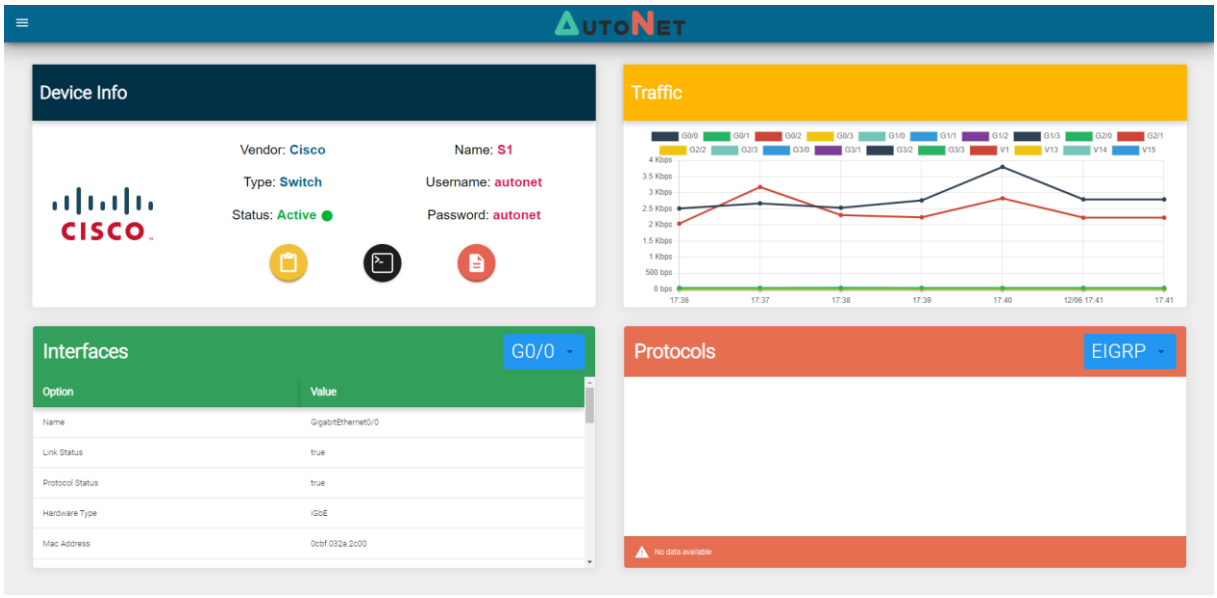
Πηγαίνοντας στην σελίδα συσκευών ο χρήστης έχει την δυνατότητα να προβάλει τις συσκευές για τις οποίες έχει τα δικαιώματα να τις παρακολουθεί. Πιο συγκεκριμένα παρουσιάζονται οι πληροφορίες της συσκευής όπως ο τύπος της, το όνομα της, η διεύθυνση IP που ο server επικοινωνεί με την συσκευή, το δίκτυο όπου ανήκει αυτή η IP, ο μέσος όρος της κίνησης που περνάει σε όλες τις διεπαφές της συσκευής αθροιστικά, για έναν αριθμό χρονικών στιγμιότυπων και τέλος η κατάσταση της συσκευής. Επίσης στο Σχήμα 2.7 υπάρχει και ένα κουμπί διαγραφής της συσκευής σε κάθε συσκευή. Τέλος δίνεται η δυνατότητα στον χρήστη να επιλέξει μια συσκευή ώστε να πλοηγηθεί στην σελίδα συσκευής (κεφάλαιο 2.8).



Σχήμα 2.7: Σελίδα συσκευών

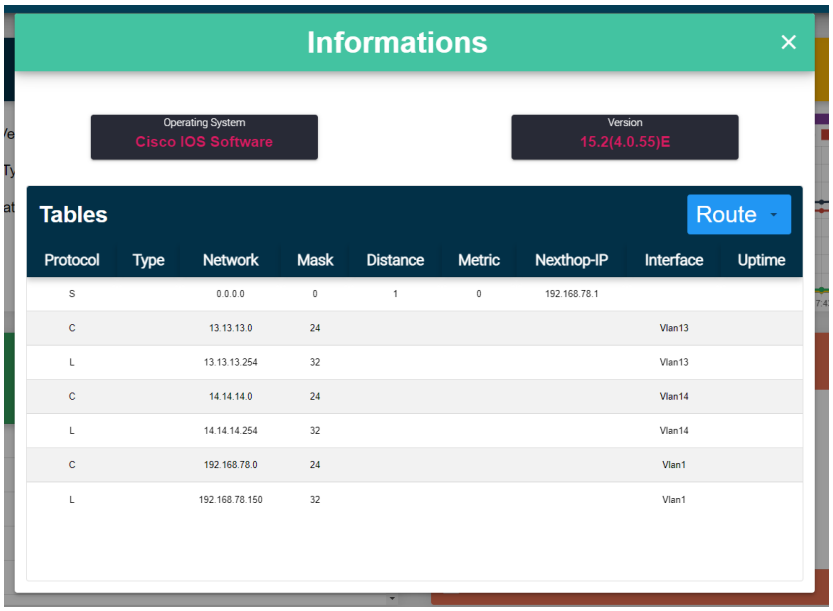
2.8 Σελίδα Συσκευής

Η σελίδα συσκευής παρουσιάζεται στο Σχήμα 2.8. Αυτή η σελίδα διαχωρίζεται σε 4 πλαίσια. Στο 1ο πλαίσιο παρέχονται πληροφορίες για την συσκευή όπως το όνομά της, τα διαπιστευτήρια που χρησιμοποιούνται για σύνδεση στην συσκευή καθώς επίσης και τρία κουμπιά τα οποία ανοίγουν ένα διαφορετικό παράθυρο το καθένα. Τα παράθυρα αυτά θα αναφερθούν παρακάτω. Στο 2ο πλαίσιο παρουσιάζεται η κίνηση όλων των διεπαφών των συσκευών, των τελευταίων χρονικών στιγμιότυπων, με την μορφή γραφημάτων ώστε να είναι πιο εύκολα κατανοητή για τον χρήστη. Επιπρόσθετα ο χρήστης μπορεί να επιλέξει ποιες διεπαφές επιθυμεί να βλέπει στο γράφημα, και να βγάλει τα δικά του συμπεράσματα με βάση το γράφημα της κίνησης. Στο 3ο πλαίσιο παρουσιάζονται όλες οι πληροφορίες της κάθε διεπαφής. Παρέχεται ένα μενού στο οποίο μπορεί ο χρήστης να επιλέξει την διεπαφή της συσκευής που επιθυμεί. Τέλος στο 4ο πλαίσιο παρουσιάζονται διάφορες πληροφορίες για τα πρωτόκολλα δρομολόγησης EIGRP και OSPF.



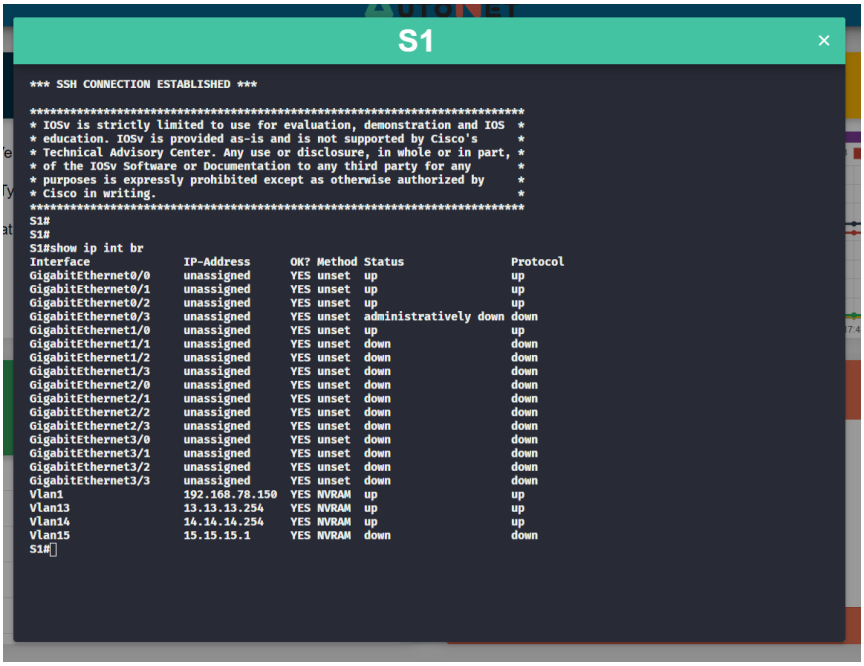
Σχήμα 2.8: Σελίδα συσκευής

Μόλις ο χρήστης πατήσει το κίτρινο κουμπί που φαίνεται στο πρώτο πλαίσιο του Σχήματος 2.8, ανοίγει ένα παράθυρο πληροφοριών. Σε αυτό το παράθυρο παρουσιάζονται πληροφορίες όπως το λειτουργικό σύστημα, η έκδοσή του καθώς και διάφοροι πίνακες όπου μεταξύ άλλων είναι ο πίνακας δρομολόγησης, ο ARP πίνακας και ο CDP πίνακας. Το παράθυρο των πληροφοριών παρουσιάζεται στο Σχήμα 2.9.



Σχήμα 2.9: Παράθυρο πληροφοριών συσκευής

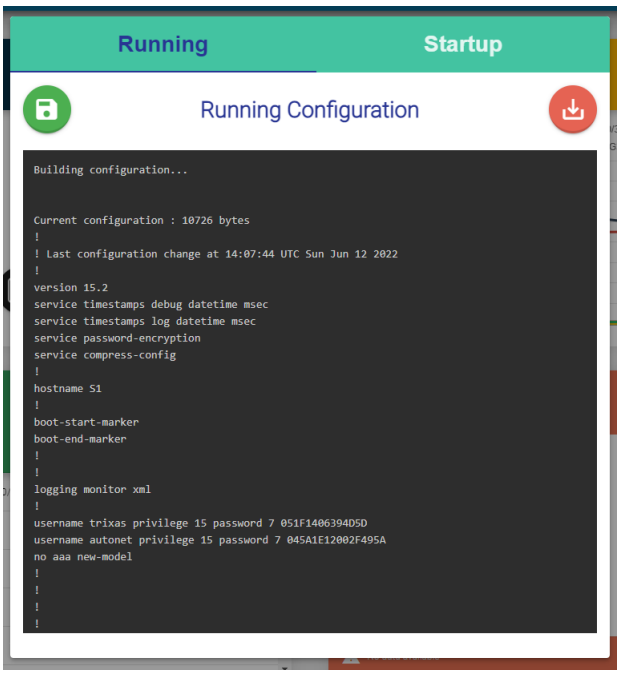
Το μαύρο κουμπί του πρώτου πλαισίου του Σχήματος 2.8, αφορά το τερματικό της συσκευής. Πιο συγκεκριμένα όταν ο χρήστης πατήσει αυτό το κουμπί τότε ανοίγει ένα παράθυρο όπου εμφανίζεται η κονσόλα της συσκευής και μπορεί να γράψει ο χρήστης εντολές σαν μια κανονική σύνδεση SSH. Δεν χρειάζεται να πληκτρολογήσει την διεύθυνση IP και τα διαπιστευτήρια της συσκευής καθώς η επίτευξη της σύνδεσης με SSH στην συσκευή είναι υπηρεσία που την παρέχει η back-end εφαρμογή. Με αυτόν τον τρόπο δίνεται στον χρήστη η δυνατότητα πρόσβασης στο τερματικό των συσκευών με πολύ άμεσο τρόπο. Το συγκεκριμένο παράθυρο παρουσιάζεται στο Σχήμα 2.10.



Σχήμα 2.10: Παράθυρο τερματικού (κονσόλας) συσκευής

Τέλος το κόκκινο κουμπί που παρουσιάζεται στο πρώτο πλαίσιο του Σχήματος 2.8, αναφέρεται στα configurations της συσκευής. Πιο συγκεκριμένα όταν ο χρήστης πατήσει αυτό το κουμπί, ανοίγει ένα παράθυρο όπου διακρίνονται τα configurations, running και startup. Ο διαχωρισμός των δύο αυτών

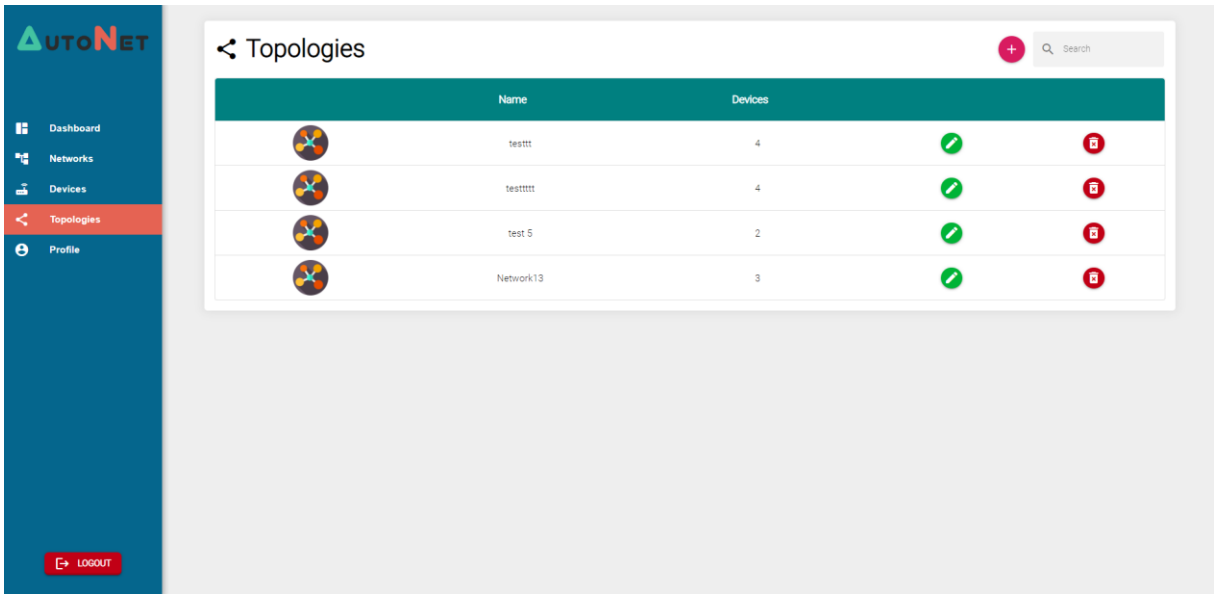
configuration γίνεται με το tab που είναι επιλεγμένο από τον χρήστη. Παρατηρείται ότι υπάρχουν δύο κουμπιά. Μόλις ο χρήστης πατήσει το πράσινο κουμπί τότε αποθηκεύεται το running configuration στο startup της συσκευής. Με άλλα λόγια εκτελείται η εντολή wr στην συσκευή. Όταν πατηθεί το κόκκινο κουμπί τότε η εφαρμογή κατεβάζει το configuration που βλέπει εκείνη την στιγμή ο χρήστης στον υπολογιστή του με την μορφή ενός text αρχείου. Τέλος αξίζει να σημειωθεί ότι το πράσινο κουμπί στο tab του startup configuration είναι απενεργοποιημένο καθώς δεν έχει κάποιο ρόλο στο startup. Το συγκεκριμένο παράθυρο παρουσιάζεται στο Σχήμα 2.11.



Σχήμα 2.11: Παράθυρο των configurations συσκευής

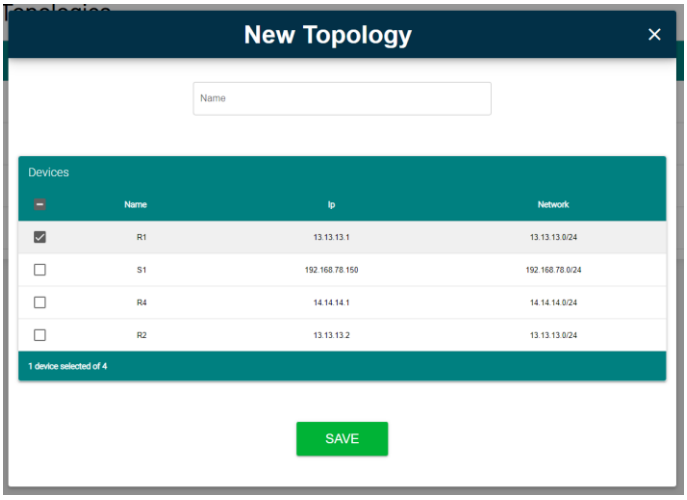
2.9 Σελίδα Τοπολογιών

Η σελίδα τοπολογιών παρουσιάζεται στο Σχήμα 2.12. Εκεί παρουσιάζονται όλες οι τοπολογίες που έχει δημιουργήσει ο χρήστης. Πιο συγκεκριμένα, για κάθε τοπολογία εκτός από το όνομα και τις συσκευές που περιέχει, παρέχονται και δύο κουμπιά, ένα για την τροποποίηση και ένα για την διαγραφή. Επίσης παρέχονται το πεδίο search, το οποίο χρησιμοποιείται για το φιλτράρισμα των δεδομένων, καθώς και ένα κουμπί με το οποίο ο χρήστης μπορεί να προσθέσει μια καινούρια τοπολογία. Παρουσιάζονται στην πάνω δεξιά θέση του Σχήματος 2.12. Το κουμπί της καινούργιας τοπολογίας και το κουμπί της επεξεργασίας της τοπολογίας ανοίγουν το ίδιο παράθυρο με διαφορετικές παραμέτρους.



Σχήμα 2.12: Σελίδα τοπολογιών

Το παράθυρο που ανοίγει όταν ο χρήστης πατήσει στο κουμπί της νέας τοπολογίας παρουσιάζεται στο Σχήμα 2.13. Σε αυτό το σχήμα εύκολα μπορεί να διακριθεί ότι ο χρήστης μπορεί να εισάγει το όνομα της τοπολογίας που επιθυμεί, καθώς επίσης δίνεται η δυνατότητα στον χρήστη να εισάγει όποιες συσκευές επιθυμεί στην τοπολογία που πρόκειται να δημιουργήσει. Στην περίπτωση της επεξεργασίας μιας τοπολογίας, οι τιμές αυτών των πεδίων είναι προκαθορισμένες με βάση τα στοιχεία της τοπολογίας.

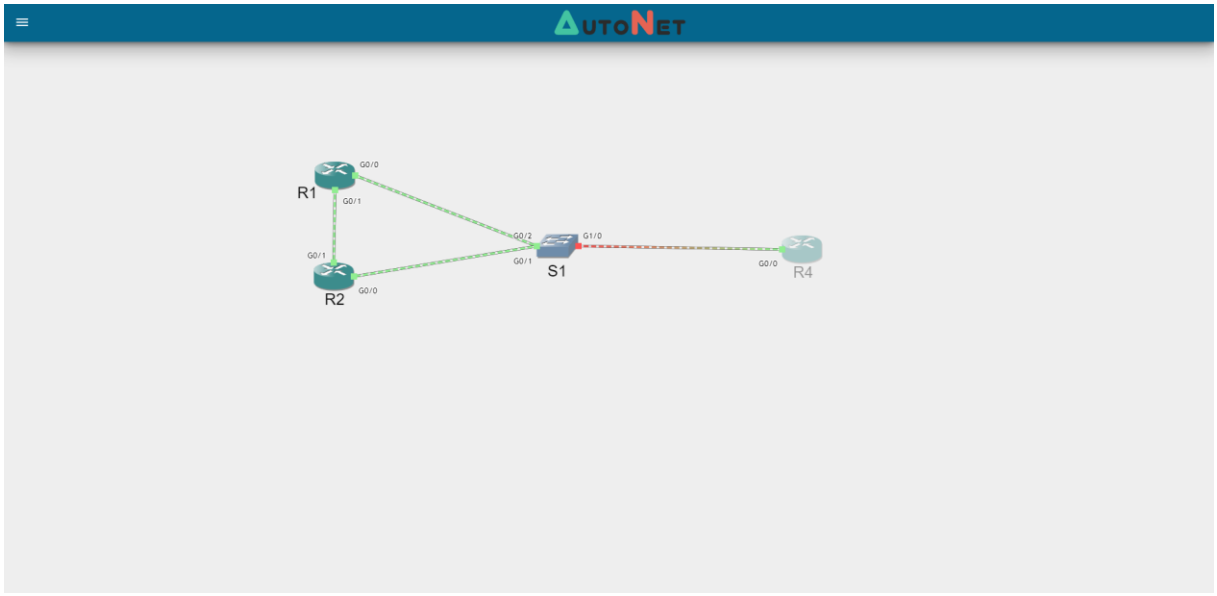


Σχήμα 2.13: Παράθυρο δημιουργία νέας τοπολογίας

2.10 Σελίδα Τοπολογίας

Στην σελίδα της τοπολογίας εμφανίζονται όλες οι συσκευές που έχει επιλέξει ο χρήστης για μια συγκεκριμένη τοπολογία, καθώς και οι συνδέσεις μεταξύ τους. Ο χρήστης μπορεί να μετακινήσει τις συσκευές προκειμένου να τις παρακολουθεί με όποιον τρόπο επιθυμεί και μπορεί να πλοηγηθεί κατευθείαν στην σελίδα της συσκευής πατώντας διπλό κλικ πάνω της. Εύκολα διακρίνεται ότι η τοπολογία είναι μια αναπαράσταση της κατάστασης των μεταξύ τους συνδέσεων τους, καθώς και της κατάστασης της συσκευής. Πιο συγκεκριμένα, όταν μια διεπαφή φαίνεται πράσινη τότε είναι ενεργή.

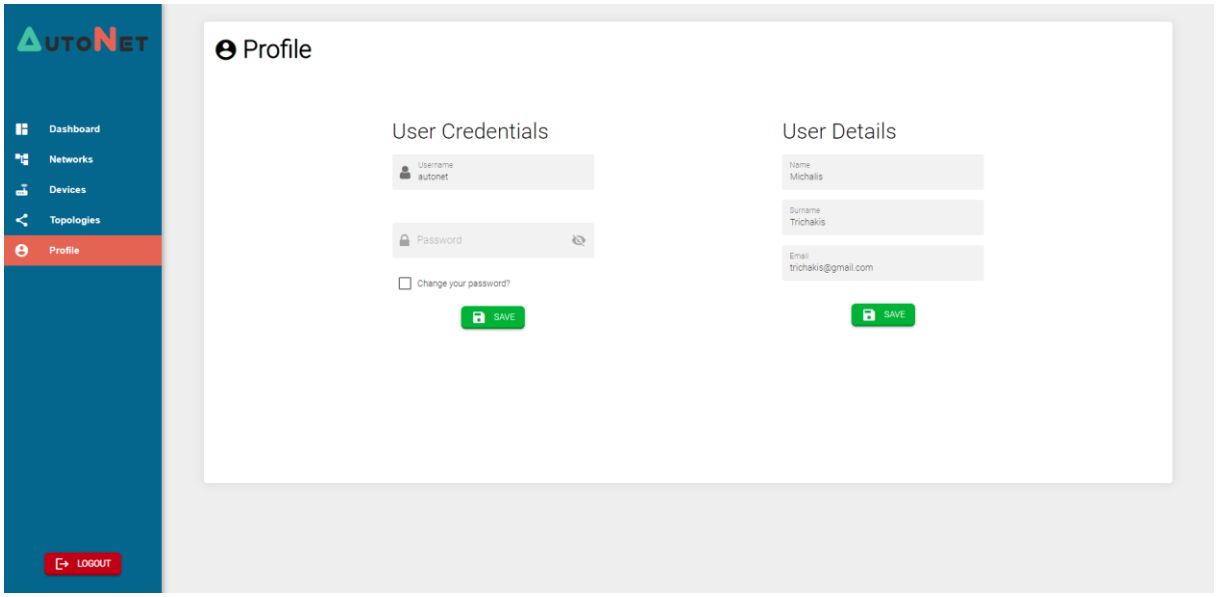
Αντιθέτως όταν μια διεπαφή είναι κόκκινη τότε σημαίνει ότι η συγκεκριμένη διεπαφή είναι “down”. Τέλος όταν η συσκευή δεν φαίνεται καθαρά όπως η R4 στο Σχήμα 2.14 σημαίνει ότι η συσκευή είναι απενεργοποιημένη ή ότι ο server δεν μπορεί να επικοινωνήσει μαζί της. Η αναπαράσταση της σελίδας της τοπολογίας φαίνεται στο Σχήμα 2.14.



Σχήμα 2.14: Σελίδα τοπολογίας

2.11 Σελίδα Προφίλ

Η σελίδα του προφίλ του χρήστη παρουσιάζεται στο Σχήμα 2.15 και αφορά τις γενικές πληροφορίες του χρήστη. Μέσα από αυτή την σελίδα ο χρήστης μπορεί να προβάλει και να τροποποιήσει τις πληροφορίες του. Εκτός από τις γενικές πληροφορίες όπως είναι το όνομα, το επίθετο και το email, δίνεται η δυνατότητα να αλλάξει τα διαπιστευτήρια της σύνδεσης. Επειδή είναι δύο διαφορετικές ενέργειες παρέχονται δύο διαφορετικά κουμπιά αποθήκευσης.



Σχήμα 2.15: Σελίδα Προφίλ

2.12 Επίλογος

Στο παρόν κεφάλαιο παρουσιάστηκε η πλατφόρμα AutoNet. Στο επόμενο κεφάλαιο θα αναλυθούν όλες οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν.

Κεφάλαιο 3ο: Τεχνολογίες που Χρησιμοποιήθηκαν

3.1 Εισαγωγή

Στο παρόν κεφάλαιο θα γίνει μια αναφορά στις τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της πλατφόρμας που αφορά την εργασία. Πιο συγκεκριμένα θα αναφερθούν οι γλώσσες προγραμματισμού που χρησιμοποιήθηκαν για την επίτευξη του web application (front-end) καθώς και του server side application (back-end) τα οποία αποτελούν και τα δυο την υλοποιημένη πλατφόρμα της εργασίας. Επιπρόσθετα θα αναφερθούν όλες οι βιβλιοθήκες που χρησιμοποιήθηκαν όπως και η βάση δεδομένων.

3.2 Βασικές Τεχνολογίες Front-End

3.2.1 HTML

Η HTML (Hyper Text Markup Language) είναι μια γλώσσα σήμανσης η οποία χρησιμοποιείται για την ανάπτυξη ιστοσελίδων. Αποτελείται από ένα σύνολο στοιχείων τα ονομαζόμενα στοιχεία HTML. Αυτά τα στοιχεία αποτελούνται από ετικέτες (tags) με το σύμβολο “μικρότερο από” (<) και τελειώνουν με το σύμβολο “μεγαλύτερο από” (>). Οι περισσότερες ετικέτες λειτουργούν ανά ζεύγη με την ετικέτα έναρξης και την ετικέτα λήξης όπως για παράδειγμα: ….. . Η ετικέτα λήξης έχει πάντα μετά το σύμβολο “μικρότερο από” (<) τον χαρακτήρα “/” με τον οποίο προσδιορίζεται ότι αυτή η ετικέτα είναι μία ετικέτα λήξης, το οποίο σημαίνει ότι εδώ κλείνει η επιρροή της ετικέτας που έχει ανοίξει προηγουμένως με την ετικέτα έναρξης. Ανάμεσα στις ετικέτες έναρξης και λήξης μπορούν οι σχεδιαστές να βάλουν ότι προτιμούν (κείμενο, πίνακες, εικόνες κτλ) ώστε να έχουν το αποτέλεσμα που επιθυμούν [6].

Ο web browser διαβάζει τα αρχεία HTML και με βάση αυτά τα αρχεία δημιουργεί τις ιστοσελίδες. Αυτό επιτυγχάνεται με την παρουσίαση των περιεχομένων των ετικετών και την μορφοποίηση τους, ανάλογα με το είδος της ετικέτας που περιέχονται. Τέλος αξίζει να σημειωθεί ότι η HTML αποτελεί και σήμερα την βασική γλώσσα σήμανσης για την υλοποίηση ιστοσελίδων [6].

```html
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Document</title>
    </head>
    <body>

    </body>
</html>
```

Σχήμα 3.1: Παράδειγμα Κώδικα HTML

3.2.2 CSS - SASS

Η CSS (Cascading Style Sheets) είναι μια γλώσσα ύφους που έχει σαν σκοπό τον έλεγχο της εμφάνισης των ετικετών ενός εγγράφου HTML. Με άλλα λόγια η CSS είναι υπεύθυνη για τον τρόπο εμφάνισης μιας ιστοσελίδας στον χρήστη.

Παρέχει στον σχεδιαστή την δυνατότητα να αλλάξει διάφορα χαρακτηριστικά ενός HTML στοιχείου όπως είναι το χρώμα του, η στοίχιση η γραμματοσειρά και άλλα πολλά. Στην ουσία η CSS παρέχει στον σχεδιαστή την δυνατότητα να ορίσει το δικό styling στον ιστότοπο.

Η CSS για την σύνταξη της παρέχει τον επιλογέα (Selector). Με τον επιλογέα δηλώνεται στην ουσία σε ποιο μέρος της σήμανσης (αρχείο HTML) θα εφαρμοστεί ένα καθορισμένο από τον σχεδιαστή στυλ. Με αυτή την λογική ο επιλογέας μπορεί να είναι το όνομα μιας ετικέτας HTML (π.χ. h2), ένα όνομα κλάσης ενός στοιχείου HTML το οποίο διακρίνεται με τον χαρακτήρα “.” μπροστά από το όνομα (π.χ. .classNameOfH3) είτε ένα id ενός στοιχείο HTML το οποίο διακρίνεται με τον χαρακτήρα “#” μπροστά από το id (π.χ. #idOfH3). Εκτός αυτών των τριών κατηγοριών επιλογέων η CSS παρέχει και την δυνατότητα εισαγωγής ψευδοκλάσεων με το χαρακτήρα “:” οι οποίες δίνουν την ικανότητα στον σχεδιαστή για μια προχωρημένη επεξεργασία εμφάνισης (π.χ. h3:hover, ο σχεδιαστής μπορεί να ορίσει ένα στυλ σε κάθε HTML στοιχείο τύπου h3 κάθε φορά που ο κέρσορας είναι πάνω από το στοιχείο) [7].

```css
.progress {
    background: linear-gradient(to right top, #65dfc9, #6cdbeb);
    width: 100%;
    height: 25%;
    border-radius: 2rem;
    position: relative;
    overflow: hidden;
}

.progress::after{
    content: "";
    width: 100%;
    height: 100%;
    background: rgb(236, 236, 236);
    position: absolute;
    left: 60%;
}
```

Σχήμα 3.2: Παράδειγμα Κώδικα CSS

Η Sass (Syntactically Awesome Style Sheets) είναι μια γλώσσα προγραμματισμού η οποία αποτελεί ένα υπερσύνολο της CSS. Κάθε γραμμή της Sass μεταφράζεται σε CSS για να μπορεί να διαβαστεί από τον web browser.

Η Sass παρέχει δύο είδη σύνταξης. Η πρώτη σύνταξη η οποία χρησιμοποιεί την εσοχή για να χωρίσει μπλοκ κώδικα. Η πιο σύγχρονη σύνταξη είναι η SCSS (Sassy CSS) η οποία ακολουθεί στην σύνταξη τα πρότυπα της CSS. Για τα αρχεία της sass με το συντακτικό της εσοχής έχει δοθεί η κατάληξη .sass ενώ για τα αρχεία της sass με το συντακτικό SCSS έχει δοθεί η κατάληξη .scss. Κατά κύριο λόγο στην παρούσα διπλωματική έχει χρησιμοποιηθεί η SCSS και ελάχιστα το συντακτικό της εσοχής [8].

Η δύναμη που προσφέρει η Sass και ο λόγος που είναι ευρέως γνωστή είναι επειδή προσθέτει στη CSS περισσότερες δυνατότητες από ότι παρέχει από μόνη της η CSS. Κάποιες από τις λειτουργίες που παρέχει είναι: η δήλωση μεταβλητών, το nesting με το οποίο ο σχεδιαστής μπορεί να ορίσει το στυλ ενός HTML στοιχείου μέσα στον γονέα του όπως είναι δηλωμένο και στο HTML αρχείο, τα mixins τα οποία είναι μπλοκς από στυλ τα οποία μπορούν να επαναχρησιμοποιηθούν σε πολλά σημεία του sass αρχείου ή ακόμα και σε άλλα sass αρχεία, οι επαναλήψεις όπως στις κλασικές γλώσσες προγραμματισμού και τέλος η δυνατότητα που δίνει στον σχεδιαστή να εισάγει και να εξάγει μεταβλητές mixins και πολλά άλλα [9].

```scss
&__side {
  height: 52rem;
  transition: all 0.8s ease;
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  backface-visibility: hidden;
  border-radius: 3px;
  overflow: hidden;
  box-shadow: 0 1.5rem 4rem rgba($color-black, 0.15);

  &--front {
    background-color: $color-white;
  }

  &--back {
    transform: rotateY(180deg);

    &-1 {
      background-image: linear-gradient(
        to right bottom,
        $color-secondary-light,
        $color-secondary-dark
      );
    }
    &-2 {
      background-image: linear-gradient(
        to right bottom,
        $color-primary-light,
        $color-primary-dark
      );
    }
    &-3 {
      background-image: linear-gradient(
        to right bottom,
        $color-tertiary-light,
        $color-tertiary-dark
      );
    }
  }
}
```

Σχήμα 3.3: Παράδειγμα Κώδικα Sass (scss)

3.2.3 JavaScript

Η JavaScript είναι μια γλώσσα προγραμματισμού που αρχικά δημιουργήθηκε για τον προγραμματισμό σεναρίων και για αυτό τον λόγο ήταν στην αρχή μια γλώσσα σεναρίων. Αυτό με την πάροδο του χρόνου άλλαξε και η JavaScript σήμερα θεωρείται μια γλώσσα προγραμματισμού γενικού σκοπού. Ο λόγος για τον οποίο την θεωρούμε γενικού σκοπού είναι επειδή με την JavaScript μπορεί ένας προγραμματιστής να γράψει εκτός από scripts που εκτελούνται στον φυλλομετρητή (client side), scripts και σε διακομιστές (server side), σε εφαρμογές για υπολογιστές (windows applications) ακόμα και σε εφαρμογές έξυπνων συσκευών (Android και iOS Applications) [10].

Η JavaScript είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού με ασθενείς τύπους. Γενικά κάθε μεταβλητή που δηλώνεται στην JavaScript μπορεί κατά την διάρκεια της εκτέλεσης να αλλάξει ο τύπος της μεταβλητής. Αυτό έρχεται να αλλάξει κυρίως η TypeScript καθώς αποτελεί ένα υπερσύνολο της JavaScript. Η TypeScript δεν έχει χρησιμοποιηθεί στην παρούσα διπλωματική και για αυτό τον λόγο δεν θα αναλυθεί περισσότερο [11].

```javascript
findInterface(name){
    return new Promise(resolve => {
        this.ifs.forEach(element => {
            if (element.interface.value === name)
                resolve(element);
        });
    })
}
```

Σχήμα 3.4: Παράδειγμα Κώδικα JavaScript

Συνοψίζοντας η JavaScript αποτελεί μια γλώσσα προγραμματισμού που μπορεί να χρησιμοποιηθεί σε διάφορες πλατφόρμες για την υλοποίηση ποικίλων τύπων εφαρμογών. Στην παρούσα διπλωματική

εργασία έχει χρησιμοποιηθεί αποκλειστικά στην front-end εφαρμογή καθώς έχει υλοποιηθεί με την Vue.js (Κεφάλαιο 3.3.1) η οποία είναι ένα javascript framework, όπως επίσης και στο back-end service καθώς έχει υλοποιηθεί με το Node.js (Κεφάλαιο 3.4.1)

3.3 Vue.js

Η Vue.js είναι ένα framework της javascript. Χρησιμοποιείται για να φτιάξει ο προγραμματιστής εύκολα user interfaces καθώς αποτελεί ένα component-based framework το οποίο σημαίνει ότι κάθε στοιχείο μιας σελίδας θα μπορούσε να είναι ένα ανεξάρτητο και επαναχρησιμοποιήσιμο component [12].

Η Vue.js χρησιμοποιεί τα πρότυπα HTML, CSS, Javascript. Ενα vue component είναι ένα αρχείο με την κατάληξη .vue και αποτελείται από 3 μέρη:

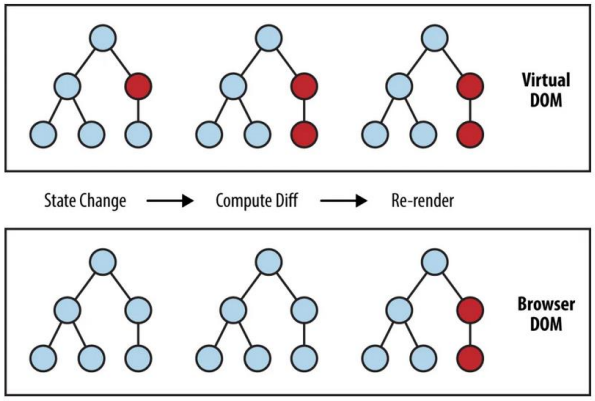
- Το template: Αποτελείται από HTML ετικέτες και στην ουσία σε αυτό το σημείο ο προγραμματιστής γράφει την HTML του εκάστοτε component (π.χ. divs, spans, p…).
- Το script: Αποτελείται από κώδικα JavaScript και εδώ μέσα ο προγραμματιστής μπορεί να ορίσει όλη την λογική και την λειτουργικότητα του εκάστοτε component
- Το style: Αποτελείται από επιλογείς CSS και σε αυτό ο προγραμματιστής μπορεί να ορίσει το στυλ του εκάστοτε component.

Με λίγα λόγια αυτό που παρέχει η vue σαν component είναι ένα αρχείο στο οποίο ο προγραμματιστής μπορεί να ορίσει το τι θα περιέχει, το πως θα λειτουργεί και το πως θα εμφανίζεται κάθε component [13].

```
<template>

</template>

<script>
export default {

}
</script>

<style>

</style>
```

Σχήμα 3.5: Παράδειγμα Vue Component

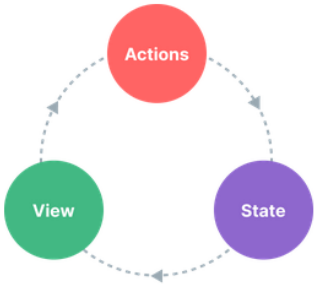
Ο κύριος λόγος όμως που η vue.js αποτελεί ένα εξαιρετικό εργαλείο δεν είναι το ότι είναι ένα component-based framework, αλλά το γεγονός ότι παρέχει το virtual Document Object Model (DOM). Το virtual DOM είναι μια αναπαράσταση JavaScript του πραγματικού DOM. Η vue.js (και το κάθε framework που χρησιμοποιεί την έννοια του virtual DOM, π.χ. η React.js) δημιουργεί το virtual DOM με βάση το πραγματικό DOM. Όταν γίνει κάποια αλλαγή στο DOM τότε συγκρίνει προηγούμενο virtual DOM με το καινούργιο και κάνει render μόνο τις αλλαγές που έγιναν στο πραγματικό DOM. Αυτή η τεχνική προσφέρει ταχύτητα και απόδοση στο web application καθώς δεν χρειάζεται να γίνει render όλο το DOM σε κάθε αλλαγή που συμβαίνει σε ένα component [14 - 16].



Σχήμα 3.6: Λειτουργία Virtual Dom [17]

Κάθε component στην vue έχει και το δικό του state. Γενικά κάθε component χαρακτηρίζεται από [18]:

- Το state: είναι τα δεδομένα τα οποία παρακολουθούνται συνεχώς για αλλαγές
- Το view: είναι η απεικόνιση του state
- Τα actions: οι ενέργειες που γίνονται όταν πρέπει να αλλάξει το state μετά την επέμβαση του χρήστη μέσω του view



Σχήμα 3.7: Λειτουργία State Management [18]

Η Vue.js παρέχει επίσης τα props. Τα props είναι μεταβλητές οποιουδήποτε τύπου οι οποίες μεταφέρονται από έναν parent component σε ένα child component. Η Vue ακολουθεί τον κανόνα του One-Way Data Flow ο οποίος σημαίνει ότι όταν αλλάξουν οι τιμές των props στο parent component τότε το parent component θα ενημερώσει τα παιδιά με τις καινούργιες τιμές αλλά δεν μπορεί να συμβεί το αντίθετο [19].

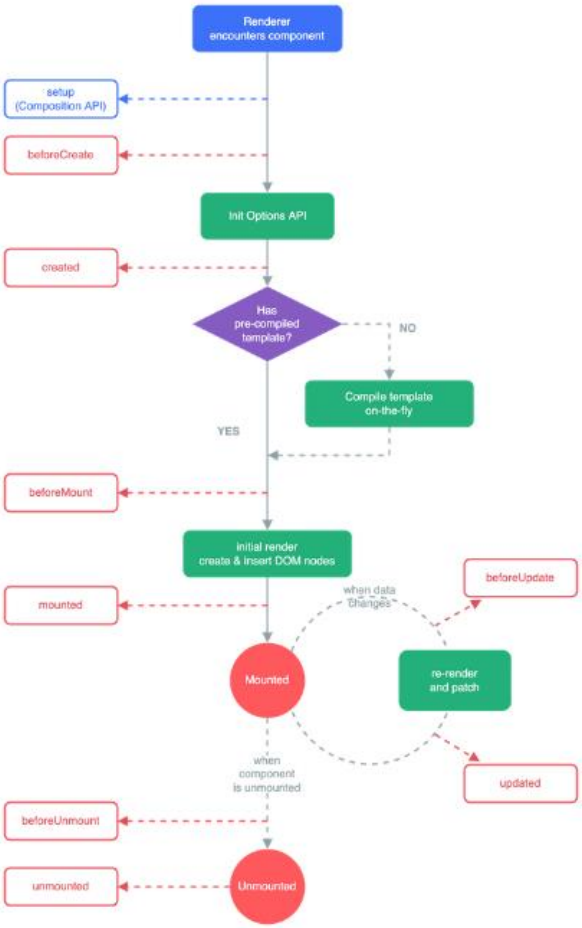
Ένα ακόμα από τα πολύ σημαντικά χαρακτηριστικά της vue είναι τα vue directives που παρέχει. Τα vue directives είναι σαν τα HTML directives με την διαφορά ότι χρησιμοποιούνται μόνο από την vue. Τα vue directives παρέχουν πολλές δυνατότητες στον προγραμματιστή όπως είναι το v-if (το οποίο κάνει render το στοιχείο αν εξυπηρετείται μια συνθήκη), το v-for (το οποίο κάνει render ένα στοιχείο πολλές φορές μέχρι να ολοκληρωθεί το loop), και άλλα πολλά [20].

Η vue.js παρέχει επίσης τα methods, τα computed properties και τα watchers. Τα methods είναι ένα αντικείμενο που εκεί μέσα δηλώνονται όλες οι μέθοδοι. Μπορούν να χρησιμοποιηθούν μετά από κάποιο action όπως για παράδειγμα το onClick [21]. Τα computed properties επιτρέπουν στον προγραμματιστή να υπολογίσει μια τιμή ή ένα σύνολο τιμών και να το επιστρέψει σε ένα σημείο. Τα computed properties ξανά υπολογίζονται κάθε φορά που αλλάζει η τιμή του state του component [22]. Τα watcher επιτρέπουν

στον προγραμματιστή να ορίσει παρατηρητές όπου η δουλειά τους είναι να παρακολουθούν ένα στοιχείο και να εκτελεί συγκεκριμένες ενέργειες όταν αλλάζει η τιμή του στοιχείου. Οι watchers γίνονται αρκετά χρήσιμοι όταν πρόκειται για ασύγχρονες ενέργειες [23].

Κάθε component μέχρι να φορτωθεί στο DOM περνάει από ένα στάδιο προετοιμασίας. Ο κύκλος ζωής λοιπόν αποτελείται από τα εξής: [24]

- **beforeCreate:** Καλείται αμέσως όταν αρχικοποιηθεί το instance, μετά την ανάλυση των props, πριν από την επεξεργασία άλλων επιλογών όπως το computed properties.
- **created:** Σε αυτό το σημείο έχουν ρυθμιστεί τα ακόλουθα τα reactive data, computed properties, τα methods και τα watchers.
- **beforeMount:** Σε αυτό το σημείο το component έχει ολοκληρώσει τη ρύθμιση του state, αλλά δεν έχει ακόμη φορτωθεί στο DOM.
- **mounted:** Σε αυτό το σημείο το component έχει φορτωθεί στο DOM.
- **beforeUpdate:** Σε αυτό το σημείο το component βρίσκεται στην φάση ακριβώς πριν ενημερώσει το DOM του λόγω μιας αλλαγής του state.
- **updated:** Σε αυτό το σημείο το component έχει ενημερώσει το DOM του λόγω αλλαγής του state του.
- **beforeUnmoun:** Σε αυτό το σημείο το component πρόκειται να βγει από το DOM αλλά όλες του οι ιδιότητες είναι ακόμα πλήρως λειτουργικές
- **unmounted:** Σε αυτό το σημείο το component έχει βγει από το DOM.



Σχήμα 3.8: Κύκλος ζωής της Vue [25]

Στην παρούσα διπλωματική έχει χρησιμοποιηθεί αποκλειστικά η Vue.js για την υλοποίηση του front-end web application και χρησιμοποιήθηκε συγκεκριμένα η έκδοση 3.

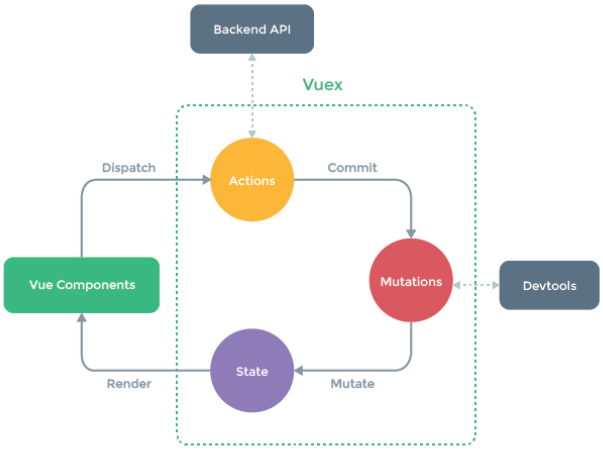
3.3.1 Vue Router

Η Vue.js παρέχει το Vue Router. Το Vue Router δίνει την δυνατότητα στον προγραμματιστή να ορίσει τα paths που θέλει για τα components που επιθυμεί. Παρέχει δύο custom components το router-link και το router-view. Με το router-link το Vue Router μπορεί να αλλάζει το url χωρίς να κάνει reload την σελίδα. Με το router-view εμφανίζει το component που αντιστοιχεί σε κάθε url. Το Vue Router αποτελεί ένα npm package και μπορεί να αποκτηθεί τρέχοντας την εντολή «npm install vue-router» στο τερματικό η οποία θα εγκαταστήσει την τελευταία έκδοση του Vue Router. Το npm θα αναλυθεί στο κεφάλαιο 3.6 [26].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την αντιστοίχιση των components με τα επιθυμητά urls και η έκδοση που χρησιμοποιήθηκε είναι η 4 (4.0.9) αφού αυτή η έκδοση είναι συμβατή με την Vue 3.

3.3.2 Vuex

Το Vuex είναι ένα state management που χρησιμοποιείται για εφαρμογές που έχουν φτιαχτεί με την Vue.js. Το πρόβλημα που έρχεται να λύσει είναι η διαχείριση του state. Το state πολλές φορές μπορεί να μεταφερθεί από component σε component μέσω των props και μπορεί να αλλάξει η τιμή του prop πολλές φορές με έναν μην επιθυμητό τρόπο ο οποίος θα οδηγήσει σε σύγχυση. Αυτό λοιπόν που προσφέρει το Vuex είναι ένα κεντρικό state το οποίο μπορούν να το χρησιμοποιήσουν όλα τα components της εφαρμογής στο οποίο εφαρμόζει κανόνες που διασφαλίζει στον προγραμματιστή ότι το state θα αλλάζει κάθε φορά μόνο με επιθυμητούς τρόπους. Αυτό το κεντροποιημένο state το Vuex το ονομάζει store. Το Vuex αποτελεί ένα npm package και μπορεί να αποκτηθεί τρέχοντας την εντολή «npm install vuex@next» στο τερματικό η οποία θα εγκαταστήσει την τελευταία έκδοση του Vuex. Το npm θα αναλυθεί στο κεφάλαιο 3.6 [27].



Σχήμα 3.9: Λειτουργία Vuex [27]

Το Vuex-persisted state χρησιμοποιήθηκε για την διατήρηση του store σαν cookie ώστε να διατηρείται μετά από ανανέωση της σελίδας ή μετά το άνοιγμα μια σελίδας του application σε νέα καρτέλα.

Αποτελεί και αυτό ένα npm package και μπορεί να αποκτηθεί τρέχοντας την εντολή «npm install vuex-persistedstate» στο τερματικό [28].

Στην παρούσα διπλωματική το Vuex χρησιμοποιήθηκε για να κεντροποιηθεί το store και να μπορούν τα δεδομένα να είναι προσβάσιμα από όλα τα components της εφαρμογής. Η έκδοση που χρησιμοποιήθηκε για το Vuex είναι η 4.0.2.

3.3.3 Quasar

Το Quasar είναι ένα open-source framework το οποίο έχει φτιαχτεί για να ενσωματωθεί σε Vue.js applications. Παρέχει μια μεγάλη γκάμα από custom components (όπως buttons, tables, cards) τα οποία προσφέρουν στον προγραμματιστή την δυνατότητα να φτιάξει σχετικά εύκολα και γρήγορα ένα όμορφο και responsive application. Αποτελεί ένα πολύ δυνατό εργαλείο καθώς παρέχει την δυνατότητα να κάνει build το app εκτός σαν ένα web app ακόμα και σε Andoid, iOS, Mac και Windows application χωρίς να χρειάζεται να γραφτεί διαφορετικός κώδικας σε διαφορετική γλώσσα προγραμματισμού για όλες αυτές τις πλατφόρμες. Επίσης αποτελεί ένα από τα πιο σημαντικά framework που μπορεί κάποιος να χρησιμοποιήσει στην Vue 3 καθώς είναι φτιαγμένο πάνω στην Vue 3 [29].

Στην παρούσα διπλωματική χρησιμοποιήθηκε αποκλειστικά για την χρήση πολλών εκ των custom components που παρέχει το Quasar καθώς σχεδόν όλο το front-end application έχει βασιστεί σε αυτό. Η έκδοση που χρησιμοποιήθηκε είναι η 2.0.0.

3.4 Βασικές Τεχνολογίες Back-End

3.4.1 Node.js

Το Node.js είναι ένα cross-platform open-source περιβάλλον το οποίο είναι χτισμένο πάνω στην JavaSctript. Επιτρέπει στον προγραμματιστή να γράψει ενα back-end service με την γλώσσα προγραμματισμού JavaScript και δίνεται η δυνατότητα να τρέξει κώδικα JavaScript εκτός του web browser. Αποτελεί ένα μέσο με το οποίο ο προγραμματιστής γράφει server-side services. Όλα τα scripts που γράφονται σε Node.js τρέχουν στον server. Αποτελεί μια πολύ καλή επιλογή για τους προγραμματιστές καθώς με μια γλώσσα προγραμματισμού την JavaScript μπορεί να γράψει και σε front-end και σε back-end [30],[31].

Το Node.js βασίζεται σε μια αρχιτεκτονική όπου τα event εκτελούνται με τρόπο ασύγχρονο. Αυτό που θέλει να πετύχει με τον τρόπο αυτό είναι η βέλτιστη απόδοση σε εφαρμογές με πολλές λειτουργίες εισόδου και εξόδου, καθώς και σε real-time εφαρμογές όπως για παράδειγμα είναι ένα chat [30],[31].

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Σχήμα 3.10: Παράδειγμα κώδικα υλοποίησης ενός Node.js server [31]

Κάθε Node.js project έχει ένα αρχείο package.json. Σε αυτό το αρχείο διακρίνονται διάφορα σημαντικά πεδία όπως: [32]

- **name:** Το όνομα του project που θέλει ο προγραμματιστής να έχει
- **version:** Η έκδοση του project το οποίο το ορίζει ο προγραμματιστής
- **main:** Το αρχείο που είναι η αρχή του project
- **scripts:** Ενα object όπου τα κλειδιά κάθε πεδίου αποτελεί το όνομα του script που θέλει ο προγραμματιστής να ορίσει και στην τιμή του πεδίου είναι η εντολή που θέλει να τρέξει. Ένα script μπορεί να τρέξει με την εντολή "npm run name_of_script"
- **repository:** Ενα object όπου έχει ένα πεδίο type για του οποίου η τιμή δείχνει τον τύπο της έκδοσης του ελέγχου (version control) για παράδειγμα το git, καθώς και ένα url στο οποίο δείχνει το url του repository του version control.
- **dependencies/devDependencies:** ενα object όπου κάθε πεδίο σαν κλειδί έχει το όνομα ενός npm package και σαν τιμή έχει την έκδοση αυτού του package. Αποτελεί το object στο οποίο βλέπουμε ποια packages έχουμε εγκαταστήσει στο Node.js project μας.

```
{
  "name": "project",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "gulp": "^3.9.0"
  }
}
```

Σχήμα 3.11: Παράδειγμα αρχείου package.json [33]

Στην παρούσα διπλωματική χρησιμοποιήθηκε αποκλειστικά για την ανάπτυξη του back-end server και αποτελεί ίσως το σημαντικότερο κομμάτι της διπλωματικής μιας και έχει διπλό ρόλο: α) να εξυπηρετεί το front-end application (Vue.js application) και β) να επικοινωνεί και να ελέγχει τις συσκευές Cisco. Η έκδοση που χρησιμοποιήθηκε είναι η 14.17.0.

3.4.2 Express.js

Η Express είναι ένα open source framework για το Node.js. Τα κύρια χαρακτηριστικά της είναι ότι μπορεί ένας προγραμματιστής να φτιάξει εύκολα ένα web application καθώς παρέχει ένα μεγάλο σύνολο δυνατοτήτων, και APIs αφού παρέχει μια μεγάλη ποικιλία από μεθόδους HTTP. Επίσης η express προτιμάται διότι διαχειρίζεται πολύ καλά την απόδοση της εφαρμογής και μπορεί να εγκατασταθεί στο Node.js project τρέχοντας την εντολή "npm install express" στο τερματικό. Το npm θα αναλυθεί στο κεφάλαιο 3.6 [34].

Στην παρούσα διπλωματική χρησιμοποιήθηκε η express για να βελτιωθεί η απόδοση, για να απλοποιηθεί το συντακτικό του κώδικα καθώς και για να δημιουργηθεί εύκολα το api για το login. Χρησιμοποιήθηκε η έκδοση 4.17.1

3.4.3 MongoDB

Η MongoDB είναι μια document-based βάση δεδομένων. Είναι αρκετά απλή και κατανοητή ενώ μπορεί να καλύψει και πιο σύνθετα ζητήματα. Η MongoDB χαρακτηρίζεται από documents και από collections. Τα documents είναι json objects τα οποία μπορούν να έχουν διαφορετική δομή μεταξύ τους. Τα collections περιέχουν τα documents. Αν μπορούσε να γίνει σύγκριση με μια sql βάση δεδομένων τότε θα λέγαμε ότι τα collections είναι τα tables και τα documents είναι οι εγγραφές κάθε πίνακα. Αποτελεί μια δωρεάν βάση δεδομένων την οποία μπορεί ο καθένας να την κατεβάσει από τον ιστότοπό της [35].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την αποθήκευση των δεδομένων των χρηστών και τις πληροφορίες κάθε συσκευής που ελέγχει η πλατφόρμα που αναπτύχθηκε. Για την απεικόνιση των δεδομένων κατά την διάρκεια της ανάπτυξης χρησιμοποιήθηκε το Mongo Compass το οποίο είναι ένα δωρεάν application που παρέχεται στο website της MongoDB.

3.5 Python

Η Python είναι μια γλώσσα προγραμματισμού γενικού σκοπού. Το συντακτικό της είναι αρκετά απλό και επιτρέπει στους προγραμματιστές να γράφουν σύνθετες έννοιες σε αρκετά λίγες γραμμές κώδικα. Η μεγάλη ποικιλία βιβλιοθηκών που παρέχονται την κάνει πολύ ισχυρή γλώσσα και είναι ευρέως διαδεδομένη. Ένα μειονέκτημα της γλώσσας είναι ότι επειδή είναι μια γλώσσα διερμηνευόμενη (interpreted) μειονεκτεί στην ταχύτητα σε σχέση με μεταγλωττιζόμενες γλώσσες (compiled languages) όπως είναι η C και η C++.

Η Python αποτελεί μια open source γλώσσα και έρχεται με προεγκατεστημένο το pip το οποίο είναι ένα σύστημα διαχείρισης πακέτων που μπορεί ο προγραμματιστής να χρησιμοποιήσει για να εγκαταστήσει εξωτερικές βιβλιοθήκες. Συγκεκριμένα από την Python 2.7.9 και μετά (σειρά Python 2) και από την Python 3.4 και μετά (χρησιμοποιείται το pip3 για την Python 3) το pip έρχεται μαζί με την εγκατάσταση της Python [36].

Στην παρούσα διπλωματική η Python χρησιμοποιήθηκε για την υλοποίηση της επικοινωνίας του service (back end) με τις συσκευές Cisco εφαρμόζοντας σύνδεση με το πρωτόκολλο SSH και για την αποστολή εντολών στις συσκευές. Οι τεχνολογίες που χρησιμοποιήθηκαν για τη συνδεσιμότητα των Cisco συσκευών με το service θα αναλυθεί παρακάτω.

3.5.1 Nmap

Το Nmap (Network Mapper) είναι ένα εργαλείο open source το οποίο χρησιμοποιείται για τον έλεγχο ασφάλειας και για την εύρεση συσκευών σε ένα δίκτυο. Ο βασικός του στόχος είναι η γρήγορη σάρωση μεγάλων δικτύων. Το Nmap χρησιμοποιεί διάφορες τεχνικές για να πάρει πληροφορίες όπως ποιες συσκευές είναι ενεργές στο εκάστοτε δίκτυο, ποιες είναι οι βασικές πληροφορίες τους όπως το όνομα εφαρμογής και η έκδοση, καθώς και ποιο λειτουργικό σύστημα και τί έκδοση του λειτουργικού συστήματος τρέχουν. Σε γενικές γραμμές το Nmap παρέχει και άλλες δυνατότητες αλλά δεν θα αναλυθούν περισσότερο [37].

Για να χρησιμοποιηθεί σε ένα python script θα πρέπει να γίνει πρώτα εγκατάσταση η βιβλιοθήκη του Nmap για την Python με την χρήση του pip τρέχοντας την εντολή "pip install python-nmap" σε ένα τερματικό. Το documentation της βιβλιοθήκης μπορεί να βρεθεί εδώ: https://pypi.org/project/python-nmap/

```
import nmap
import json
import sys

nm = nmap.PortScanner()


try:
    nm.scan(sys.argv[1], arguments='-O')
    node = { "vendor": nm[sys.argv[1]]['osmatch'][0]['osclass'][0]['vendor'],
             "ip": sys.argv[1]}

except:
    node = { "vendor": None,
             "ip": sys.argv[1] }
```

Σχήμα 3.12: Παράδειγμα χρήσης του Nmap μέσα στο python script

Στην παρούσα διπλωματική το Nmap χρησιμοποιήθηκε για την εύρεση Cisco συσκευών, Routers ή Switches, μέσα σε ένα δίκτυο που ο χρήστης έχει ορίσει. Η έκδοση που χρησιμοποιήθηκε είναι η 0.6.4.

3.5.2 Sys

Το sys (System-specific) είναι εγκατεστημένο με την python και η χρησιμότητα του είναι να παρέχει πρόσβαση σε μεταβλητές που χρησιμοποιούνται από τον interpreter. Στην παρούσα διπλωματική χρησιμοποιήθηκε η λίστα argv του sys (sys.argv) η οποία είναι η λίστα των ορισμάτων – παραμέτρων που έχουν δοθεί στην εκτέλεση του python script [38].

Στην παρούσα διπλωματική το sys.argv χρησιμοποιήθηκε για να παρέχει πρόσβαση στα scripts στις παραμέτρους που μπορούν να περαστούν κατά την εκτέλεση τους.

3.5.3 Json

Το Json (Javascript Object Notation) αποτελεί μια λειτουργία που παρέχει η python χωρίς να απαιτεί κάποια εγκατάσταση ξεχωριστά. Ένα Json object είναι μια μορφή ανταλλαγή δεδομένων καθώς και μοιάζει αρκετά με ένα Javascript object. Αποτελεί την επικρατέστερη μορφή δεδομένων για ανταλλαγή δεδομένων όπως τα APIs [39].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για να εξάγει τα δεδομένα του script σε μορφή Json.

3.5.4 Ipaddress

Το Ipaddress είναι μια βιβλιοθήκη που παρέχει την δυνατότητα διαχείρισης διευθύνσεων και δικτύων IPv4 και IPv6. Μπορεί να εγκατασταθεί με την χρήση του pip τρέχοντας την εντολή "pip install ipaddress" σε ένα τερματικό [40].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την εύρεση όλων των διαθέσιμων διευθύνσεων IP στην διεύθυνση δικτύου που θα γίνει η εύρεση των δικτυακών συσκευών. Η έκδοση είναι η 1.0.

3.5.5 Netmiko – ConnectHandler

Το Netmiko είναι μια βιβλιοθήκη της Python η οποία χρησιμοποιεί το paramiko για να χρησιμοποιεί δικτυακές συσκευές. Το Netmiko προσφέρει το ConnectHandler το οποίο δίνει την δυνατότητα μέσα στο Python script να αναπτύξει ο προγραμματιστής μια σύνδεση SSH με κάποια συσκευή. Το Netmiko αποτελεί μια βιβλιοθήκη της Python και για να εγκατασταθεί πρέπει, με την χρήση του pip, να τρέξει την εντολή "pip install netmiko" σε ένα τερματικό [41].

Στην παρούσα διπλωματική χρησιμοποιήθηκε αποκλειστικά ο ConnectHandler του Netmiko για την σύνδεση στις δικτυακές συσκευές Cisco καθώς και για την αποστολή εντολών σε αυτές. Η έκδοση που χρησιμοποιήθηκε είναι η 3.4.0.

3.5.6 Re

Το re αποτελεί μια βιβλιοθήκη που παρέχει η python από προεπιλογή και παρέχει λειτουργίες RegEx (Regular Expression). Ένα RegEx μπορεί να χρησιμοποιηθεί για να ελέγχει αν μια συγκεκριμένη σειρά από χαρακτήρες υπάρχει μέσα σε μια συμβολοσειρά [42].

Στη παρούσα διπλωματική χρησιμοποιήθηκε στην μορφοποίηση των δεδομένων που παίρνουν τα scripts από τις συσκευές.

3.5.7 Ntc Templates

Το Ntc Templates αποτελεί ένα repository του TextFSM Templates για δικτυακές συσκευές. Η χρήση αυτής της βιβλιοθήκης είναι κυρίως για την μορφοποίηση των αποτελεσμάτων συγκεκριμένων εντολών που θέλει ο προγραμματιστής να εκτελέσει μέσω ενός python script στις συσκευές [43]. Ένα παράδειγμα χρήσης του NTC Templates παρουσιάζεται στο Σχήμα 3.13, όπου στην μέθοδο send_command του net_connect, ορίζεται η παράμετρος use_textfsm σε true. Μπορεί ο κάθε προγραμματιστής να την κατεβάσει από αυτό το git repository: https://github.com/networktocode/ntc-templates

```
cisco_881 = {
    'device_type': 'cisco_ios',
    'host':   sys.argv[1],
    'username': sys.argv[2],
    'password': sys.argv[3],
    'fast_cli': False
}
net_connect = ConnectHandler(**cisco_881)

runConf = net_connect.send_command('show running-config', use_textfsm=True)
startConf = net_connect.send_command('show startup-config', use_textfsm=True)
```

Σχήμα 3.13: Παράδειγμα χρήσης του Ntc Template

3.6 Npm

Το Npm (Node Package Manager) είναι ένας διαχειριστής πακέτων για την JavaScript και αποτελεί τον προεπιλεγμένο διαχειριστή πακέτων για τα Node.js projects. Αποτελείται από δύο σκέλη: α) Τον client στη γραμμή εντολών ο οποίος ονομάζεται npm και β) μια βάση δεδομένων όπου περιέχει όλα τα δημόσια και ιδιωτικά πακέτα τα οποία είναι διαθέσιμα για τους προγραμματιστές [44]. Ο κάθε προγραμματιστής μπορεί να αναζητήσει πληροφορίες για τα πακέτα που πρόκειται να χρησιμοποιήσει μέσω της ιστοσελίδας που παρέχει: https://www.npmjs.com/

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την εγκατάσταση των packages που χρειάστηκαν για την υλοποίηση της πλατφόρμας τόσο στην front-end όσο και στην back-end εφαρμογή.

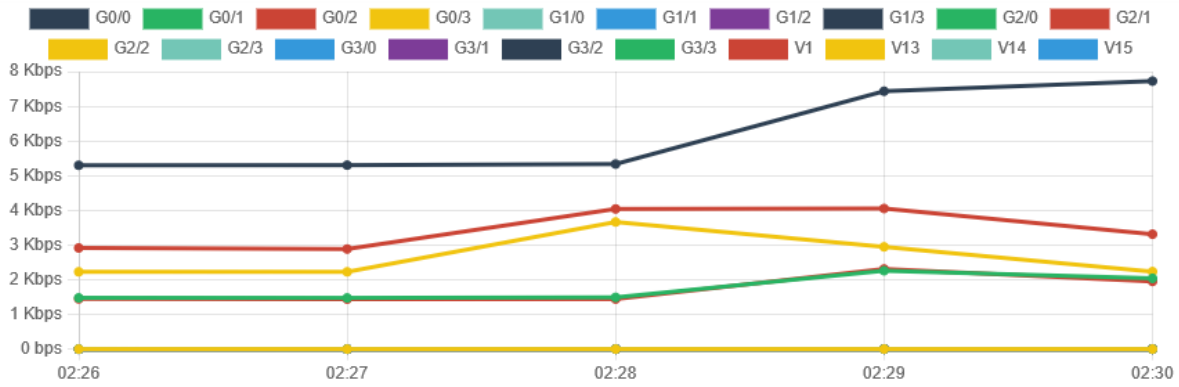
3.6.1 JavaScript Cookie (js-cookie)

Το js-cookie είναι ένα npm package. Η χρησιμότητα του είναι ότι παρέχει την δυνατότητα στον προγραμματιστή να αποθηκεύει cookies στην front-end εφαρμογή του. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή “npm install js-cookie” στη γραμμή εντολών [45].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την αποθήκευση του persisted store του vuex στα cookies. Χρησιμοποιήθηκε στην front-end εφαρμογή με την έκδοση 2.2.1.

3.6.2 Chart.js

Το Chart.js είναι μια open-source javascript βιβλιοθήκη που παρέχει την δυνατότητα της δημιουργίας ποικίλων γραφημάτων. Παρέχει πολύ καλό documentation και είναι αρκετά εύκολο να το χρησιμοποιήσει κανείς. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή “npm install chart.js ” στη γραμμή εντολών [46],[47].



Σχήμα 3.14: Γράφημα κίνησης συσκευής με την χρήση του Chart.js

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την αναπαράσταση της κίνησης των συσκευών και δικτύων με την χρήση γραφημάτων που παρέχει το Chart.js. Επομένως χρησιμοποιήθηκε στην front-end εφαρμογή με την έκδοση 3.5.0.

3.6.3 Xterm.js

Το Xterm.js είναι μια βιβλιοθήκη γραμμένη σε TypeScript. Η χρησιμότητα είναι ότι παρέχει την δυνατότητα στους προγραμματιστές να αναπτύξουν τερματικά – γραμμή εντολών στα web applications. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή " npm install xterm" στη γραμμή εντολών [48].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την απεικόνιση και την ανάπτυξη τερματικών για τις διαθέσιμες δικτυακές συσκευές που ελέγχει η πλατφόρμα με SSH. Επομένως χρησιμοποιήθηκε στην front-end εφαρμογή με την έκδοση 4.14.1.

3.6.4 Prism.js

Το Prism.js είναι μια JavaScript βιβλιοθήκη. Παρέχει στους προγραμματιστές την δυνατότητα να παρουσιάσουν highlighting συντάξεις. Μπορεί να παραμετροποιηθεί και ο προγραμματιστής μπορεί εκτός από το να επιλέξει τα χρώματα (theme), ακόμα να ορίσει τον τύπο του περιεχομένου (π.χ. απεικόνιση κώδικα JavaScript). Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή " npm install prismjs" στη γραμμή εντολών [49].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την απεικόνιση των configurations (running και startup) των δικτυακών συσκευών Cisco. Επομένως χρησιμοποιήθηκε στην front-end εφαρμογή με την έκδοση 1.25.0.

3.6.5 Body-parser

Το Body-parser είναι μια JavaScript βιβλιοθήκη. Παρέχει στους προγραμματιστές την δυνατότητα διαχειρίζονται στο Node.js project τους τα request που λαμβάνουν. Το body-parser στην ουσία βάζει όλες τις παραμέτρους ενός api request μέσα στην ιδιότητα req.body και από αυτό έχει πρόσβαση ο προγραμματιστής στις παραμέτρους του request. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή "npm i body-parser" στη γραμμή εντολών [50].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την πρόσβαση στις παραμέτρους των request που γίνονται και συγκεκριμένα στο login request. Χρησιμοποιήθηκε στην back-end εφαρμογή με την έκδοση 1.19.0.

3.6.6 Socket.IO

Το Socket.IO είναι μια JavaScript βιβλιοθήκη για real-time διαδικτυακές εφαρμογές. Παρέχει μια ζωντανή σύνδεση μεταξύ server και client και βασίζεται στα events που γίνονται μεταξύ τους. Η σύνδεση αυτή βασίζεται στην υλοποίηση των web sockets το οποίο το παρέχει η βιβλιοθήκη Socket.IO. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή "npm install socket.io" για το server-side και την εντολή "npm install socket.io-client" για το client-side στη γραμμή εντολών [51].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την υλοποίηση της αποκλειστικής επικοινωνίας του back-end application (Node.js app) και του front-end application (Vue.js) αφού ο χρήστης έχει συνδεθεί με τα σωστά διαπιστευτήρια επιτυχώς. Με την Socket.IO επιτυγχάνεται η ενημέρωση του front-end application όταν γίνεται κάποια αλλαγή στα δεδομένα για να ενημερώνεται άμεσα ο χρήστης χωρίς να χρειάζεται να γίνονται request ανά τακτά χρονικά διαστήματα από το front-end application. Επομένως χρησιμοποιήθηκε και στις δύο εφαρμογές front-end και back-end, με τις εκδόσεις 4.0.0 και 4.4.1 αντίστοιχα.

3.6.7 Axios

Το Axios είναι μια Javascript βιβλιοθήκη. Αποτελεί έναν HTTP client και χρησιμοποιείται για την επίτευξη αιτημάτων HTTP. Είναι από τις πιο δημοφιλής βιβλιοθήκες στην ανάπτυξη ενός web application. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή "npm install axios" στη γραμμή εντολών [52],[53].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την υποβολή του HTTP αιτήματος σύνδεσης του χρήστη καθώς και για την υποβολή των αιτημάτων HTTP για την εκκίνηση της εύρεσης συσκευών σέ ένα καθορισμένο δίκτυο. Χρησιμοποιήθηκε στην front-end εφαρμογή με την έκδοση 0.21.4.

3.6.8 Bcrypt.js

Το Bcrypt.js είναι μια JavaScript βιβλιοθήκη. Η χρησιμότητα του είναι ότι παρέχει την λειτουργία δημιουργίας ενός hash από ένα δοθέν αλφαριθμητικό. Το hash είναι ένα μονόδρομο αποτέλεσμα το οποίο δεν μπορεί να αναστραφεί στο αρχικό αλφαριθμητικό από το οποίο δημιουργήθηκε. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή "npm install bcryptjs" στη γραμμή εντολών [54].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την δημιουργία των hash των κωδικών πρόσβασης των χρηστών τα οποία αποθηκεύονται στην βάση δεδομένων και αποτελεί το μέσο ελέγχου για την εγκυρότητα των διαπιστευτηρίων που έδωσε ο χρήστης κατά της σύνδεσής του στην πλατφόρμα. Χρησιμοποιήθηκε στην back-end εφαρμογή με την έκδοση 2.4.3.

3.6.9 Default-gateway

Το Default-gateway είναι μια βιβλιοθήκη της JavaScript. Η χρήση της είναι να παρέχει στον προγραμματιστή την προεπιλεγμένη πύλη (default gateway) για κάθε interface που διαθέτει το μηχάνημα στο οποίο τρέχει η εφαρμογή. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή "npm install default-gateway" στη γραμμή εντολών [55].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για να αποκτηθούν τα ονόματα των interfaces του Node.js application. Επομένως χρησιμοποιήθηκε στην back-end εφαρμογή με την έκδοση 6.0.3.

3.6.10 Dotenv

Το Dotenv είναι ένα module που χρησιμοποιήθηκε σαν JavaScript βιβλιοθήκη. Η χρησιμότητα του είναι να φορτώνει μεταβλητές περιβάλλοντος από ένα αρχείο .env. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή "npm install dotenv" στη γραμμή εντολών [56].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την αποθήκευση του path για την σύνδεση με την βάση δεδομένων, την αποθήκευση του jwt token secret (θα αναφερθεί στο κεφάλαιο 3.6.13) και για την αποθήκευση της TCP port που τρέχει το front-end application. Χρησιμοποιήθηκε στην back-end εφαρμογή με την έκδοση 8.2.0.

3.6.11 Ip

Το IP είναι ένα npm package και αναφέρεται στο παγκοσμίως γνωστό πρωτόκολλο τρίτου επιπέδου IP (Internet Protocol). Χρησιμοποιείται για την εύκολη διαχείριση διευθύνσεων IP μέσα στον κώδικα JavaScript. Παρέχει την δυνατότητα στον προγραμματιστή να συγκρίνει IP διευθύνσεις, να ελέγξει για την εγκυρότητα τους και πολλές άλλες λειτουργίες. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή "npm install ip" στη γραμμή εντολών [57].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για τον έλεγχο της εγκυρότητας των διευθύνσεων δικτύων που δόθηκαν από τον χρήστη καθώς και για την εύρεση διαθέσιμων διευθύνσεων IP μέσα σε μια διεύθυνση δικτύου. Χρησιμοποιήθηκε στην back-end εφαρμογή με την έκδοση 1.1.5.

3.6.12 Joi

Το Joi είναι μια βιβλιοθήκη JavaScript. Η χρησιμότητα του είναι να περιγράφει τα δεδομένα με μια απλή γλώσσα και να ελέγχει την επικύρωση αυτών των δεδομένων. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή “npm install joi” στη γραμμή εντολών [58].

```
const loginValidation = data => {
    const schema = Joi.object({
        username: Joi.string()
            .min(6)
            .required(),
        password: Joi.string()
            .min(6)
            .required()
    });
    return schema.validate(data);
}
```

Σχήμα 3.15: Παράδειγμα κώδικα με την χρήση του Joi

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την επικύρωση των δεδομένων του χρήστη (username, password, email) στο Node.js application. Συνεπώς χρησιμοποιήθηκε στη back-end εφαρμογή με την έκδοση 17.4.0.

3.6.13 Json Web Token (JWT)

Το Json Web Token (JWT) είναι ένας μηχανισμός κρυπτογράφησης δεδομένων το οποίο χρησιμοποιήθηκε ως JavaScript βιβλιοθήκη. Η χρησιμότητα του είναι ότι παρέχει στον προγραμματιστή την δυνατότητα της ασφαλής μετάδοσης πληροφοριών σαν Json objects οι οποίες πληροφορίες είναι αξιόπιστες διότι είναι ψηφιακά υπογεγραμμένες. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή “npm install jsonwebtoken” στη γραμμή εντολών [59].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την δημιουργία του token μετά από μια επιτυχημένη σύνδεση ενός χρήστη. Χρησιμοποιήθηκε στην back-end εφαρμογή με την έκδοση 8.5.1.

3.6.14 Mongoose

Το Mongoose είναι μια βιβλιοθήκη JavaScript για την επικοινωνία του Node.js application με την MongoDB. Παρέχει την δυνατότητα μοντελοποίησης των δεδομένων που πρόκειται να αποθηκευτούν καθώς και τον έμμεσο σχηματισμό των documents που θα υπάρχουν στα collections της MongoDB. Επιπλέον το mongoose παρέχει μια ποικιλία από λειτουργίες και μεθόδους ώστε να είναι εφικτή η εκτέλεση ερωτημάτων στην βάση δεδομένων μέσω της JavaScript. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή “npm install mongoose” στη γραμμή εντολών [60].

```
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
    username: {
        type: String,
        required: true,
        min: 3,
        max: 40
    },
    password: {
        type: String,
        required: true,
        min: 6,
        max: 1024
    },
    email: {
        type: String,
        required: true,
        min: 6,
        max: 255
    },
    name: {
        type: String,
        required: false,
        min: 6,
        max: 255
    },
    surname: {
        type: String,
        required: false,
        min: 6,
        max: 255
    },
    date: {
        type: Date,
        default: Date.now
    }
});

module.exports = mongoose.model('users', userSchema);
```

Σχήμα 3.16: Παράδειγμα κώδικα μοντέλου στην MongoDB μέσω του mongoose

Στην παρούσα διπλωματική χρησιμοποιήθηκε αποκλειστικά για όλες τις ενέργειες που έγιναν από την μεριά του Node.js application με στόχο την αποθήκευση, την ενημέρωση και την ανάκτηση δεδομένων. Συνεπώς χρησιμοποιήθηκε στην back-end εφαρμογή με την έκδοση 5.12.0.

3.6.15 Ping

Το Ping είναι ένα npm package. Η χρησιμότητα του είναι ότι παρέχει στον προγραμματιστή την δυνατότητα να ελέγξει αν μια συσκευή με μια διεύθυνση IP είναι ενεργεί και είναι σε θέση να επικοινωνήσει. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή “npm install ping” στη γραμμή εντολών [61].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για τον έλεγχο όλων των IP διευθύνσεων ενός δικτύου προκειμένου να καθοριστούν οι ενεργές συσκευές του δικτύου. Χρησιμοποιήθηκε στην back-end εφαρμογή με την έκδοση 0.4.0.

3.6.16 Python-shell

Το Python-shell είναι μια JavaScript βιβλιοθήκη. Επιτρέπει στον προγραμματιστή να εκτελέσει python scripts μέσα στον κώδικα JavaScript καθώς και να πάρει αποτελέσματα αλλά και να διαχειριστεί τυχόν σφάλματα που επέστρεψε η Python. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή “npm install python-shell” στη γραμμή εντολών [62].

```
const getNodeInfo = (host) => {
    console.log("Go Python: " + host.ip);
    return new Promise(resolve => {
        let shell = new PythonShell('server/python/node_info.py', {
            mode: 'json',
            args: [host.ip, host.username, host.password]
        });
        shell.on('message', function (message) {
            resolve(message);
        });
        shell.on('error', function (message) {
            resolve({error: true, message})
        })
    })
}
```

Σχήμα 3.17: Παράδειγμα κώδικα χρήσης του python-shell

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την εκτέλεση των python scripts που αναπτύχθηκαν για την αποστολή μιας σειράς από εντολές στις Cisco συσκευές. Χρησιμοποιήθηκε στην back-end εφαρμογή με την έκδοση 2.0.3.

3.6.17 Ssh2

Το Ssh2 είναι ένα npm package το οποίο παρέχει την δυνατότητα επίτευξης σύνδεσης SSH (version 2) με μια απομακρυσμένη συσκευή. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή “npm install ssh2” στη γραμμή εντολών [63].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την επίτευξη σύνδεσης με SSH (version 2) με τις Cisco συσκευές προκειμένου να δοθεί η δυνατότητα στον χρήστη να συνδεθεί και να στείλει εντολές μέσω του front-end application. Χρησιμοποιήθηκε η έκδοση 1.4.0.

3.6.18 Syslog-server

Το Syslog-server είναι ένα npm package. Αυτό που προσφέρει στον προγραμματιστή είναι η υλοποίηση ενός Syslog Server - listener μέσα στον κώδικα JavaScript. Ο SysLog Server θα αναλυθεί στο κεφάλαιο 4. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή “npm install syslog-server” στη γραμμή εντολών [64].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την ανάπτυξη ενός SysLog Server στο Node.js application για το άκουσμα συμβάντων που γίνονται στις συσκευές και εξάγονται μέσω syslog. Συνεπώς χρησιμοποιήθηκε στη back-end εφαρμογή με την έκδοση 1.0.1.

3.6.19 Net-snmp

Το Net-snmp είναι ένα εργαλείο το οποίο χρησιμοποιήθηκε σαν npm package. Παρέχει στον προγραμματιστή την δυνατότητα της ανάπτυξης ενός SNMP server – listener μέσα στον κώδικα JavaScript. Το SNMP πρωτόκολλο θα αναλυθεί στο κεφάλαιο 4. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή “npm install net-snmp” στη γραμμή εντολών [65].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την ανάπτυξη ενός SNMP Server στο Node.js application για το άκουσμα συμβάντων που γίνονται στις συσκευές και εξάγονται μέσω SNMP. Συνεπώς χρησιμοποιήθηκε στη back-end εφαρμογή με την έκδοση 3.5.2.

3.6.20 Plain Draggable

Το Plain Draggable είναι μια JavaScript βιβλιοθήκη. Αποτελεί μια βιβλιοθήκη που προσφέρει πολύ υψηλή απόδοση κατά το drag των HTML elements και καθιστά πολύ απλή την διαχείριση του drag event των HTML elements. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή “npm install plain-draggable” στη γραμμή εντολών [66].

Στην παρούσα διπλωματική χρησιμοποιήθηκε στις συσκευές κάθε τοπολογίας ώστε να μπορεί ο χρήστης να κουνήσει με τον κέρσορα αυτές τις συσκευές για να μπορεί να προσαρμόσει αυτός τον τρόπο αναπαράστασης των συσκευών στην εκάστοτε τοπολογίας. Συνεπώς χρησιμοποιήθηκε στην front-end εφαρμογή με την έκδοση 2.5.14.

3.6.21 Leader Line

Το Leader Line είναι μια JavaScript βιβλιοθήκη. Παρέχει στον προγραμματιστή την δυνατότητα εισαγωγής διάφορων όμορφων γραμμών που καλύπτει κάθε ανάγκη. Καθώς αποτελεί ένα npm package μπορεί να εγκατασταθεί τρέχοντας την εντολή “npm install leader-line-new” στη γραμμή εντολών [67].

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την αναπαράσταση των καλωδίων (συνδέσεων) των συσκευών μεταξύ τους. Συνεπώς χρησιμοποιήθηκε στην front-end εφαρμογή με την έκδοση 1.1.9.

3.7 Εργαλεία

Παρακάτω θα γίνει μια αναφορά στα εργαλεία που χρησιμοποιήθηκαν κατά την ανάπτυξη της πλατφόρμας. Όλα τα εργαλεία που θα αναφερθούν είναι δωρεάν και μπορεί να τα χρησιμοποιήσει οποιοσδήποτε προγραμματιστής προκειμένου να πετύχει τους σκοπούς του.

3.7.1 Visual Studio Code (VS Code)

Το Visual Studio Code είναι ένας δωρεάν code editor και είναι διαθέσιμο σε Windows, Linux και macOS. Είναι από τους πιο δημοφιλής code editors καθώς παρέχει μια ποικιλία από επεκτάσεις γλωσσών προγραμματισμού που επιτρέπουν την εγγραφή κώδικα σε αυτές τις γλώσσες. Το Visual Studio Code χρησιμοποιήθηκε για την ανάπτυξη του κώδικα της πλατφόρμας [68].

3.7.2 MongoDB Compass

Το MongoDB Compass είναι μια GUI εφαρμογή με την οποία μπορεί ο χρήστης να δει τα δεδομένα της MongoDB, να τα αναλύσει, να τα επεξεργαστεί καθώς και πολλές άλλες λειτουργίες. Το MongoDB Compass είναι ένα δωρεάν application και είναι διαθέσιμο για Windows, Linux και macOS. Χρησιμοποιήθηκε κατά την διάρκεια της ανάπτυξης της πλατφόρμας σαν βοηθητικό εργαλείο για προβολή των δεδομένων της βάσης MongoDB [69].

3.7.3 GNS3

Το GNS3 (Graphical Network Simulator-3) είναι ένα open source δωρεάν λογισμικό το οποίο χρησιμοποιείται για την εξομοίωση, την διαμόρφωση, την δοκιμή και την αντιμετώπιση προβλημάτων εικονικών και πραγματικών δικτύων. Παρέχει στον χρήστη την δυνατότητα δημιουργίας τοπολογιών στις οποίες μπορεί να εισάγει δικτυακές συσκευές. Χρησιμοποιήθηκε για την δημιουργία εικονικών συσκευών με σκοπό να εισαχθούν στην πλατφόρμα που αναπτύχθηκε και να «ελέγχονται» από αυτή. Η έκδοση που χρησιμοποιήθηκε είναι η 2.2.21 [70].

3.8 Επίλογος

Στο παρόν κεφάλαιο παρουσιάστηκαν κάποιες από τις πιο σημαντικές τεχνολογίες που χρησιμοποιήθηκαν προκειμένου να υλοποιηθεί το AutoNet. Στο επόμενο κεφάλαιο θα αναλυθούν τα πρωτόκολλα και οι τεχνολογίες δικτύωσης που εφαρμόστηκαν.

Κεφάλαιο 4ο: Πρωτόκολλα και Τεχνολογίες Δικτύωσης

4.1 Εισαγωγή

Στο παρόν κεφάλαιο θα αναφερθούν τα πρωτόκολλα, οι συσκευές και οι τεχνικές που χρησιμοποιήθηκαν προκειμένου να είναι δυνατή η επικοινωνία των συσκευών με τη πλατφόρμα.

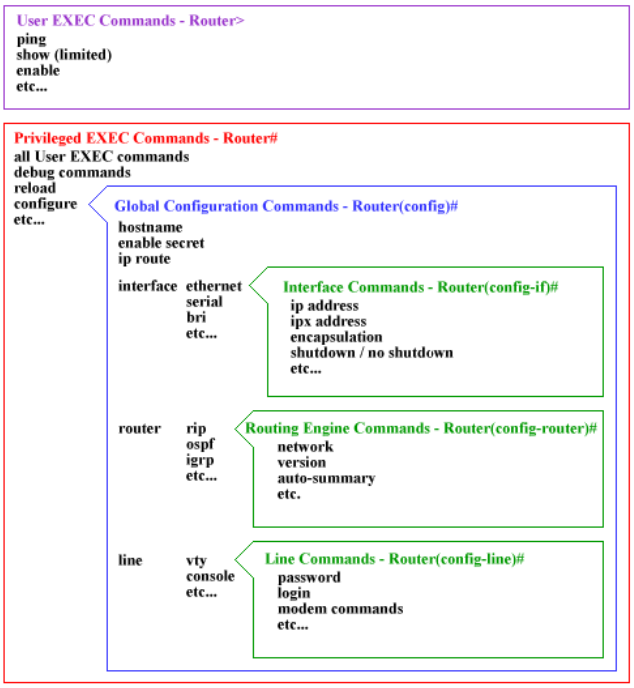
4.2 Συσκευές Cisco

Στην παρούσα διπλωματική όταν αναφέρεται μια συσκευή CISCO αφορά μια δικτυακή συσκευή που παράγει η Cisco Systems. Αυτή η συσκευή μπορεί να είναι δρομολογητής (router) ή μεταγωγέας (switch). Η εκάστοτε συσκευή τρέχει το λειτουργικό σύστημα Cisco IOS (Internetwork Operating System) του οποίου η βασική ιδιότητα είναι να καθιστά δυνατή την επικοινωνία των κόμβων των δικτύων [71].

Το Cisco IOS παρέχει μια ποικιλία από λειτουργίες στον διαχειριστή όπως η ρύθμιση της συσκευής, ο καθορισμός IP διευθύνσεων, η ενεργοποίηση πρωτοκόλλων και άλλες πολλές ενέργειες. Ο τρόπος λειτουργίας της συσκευής βασίζεται στα δύο αρχεία παραμετροποίησης το running configuration και το startup configuration. Για την εκτέλεση εντολών παρέχει τα εξής επίπεδα ασφαλείας [72]:

- **User EXEC Level:** Επιτρέπει την εκτέλεση μόνο εντολών παρακολούθησης (π.χ. ping)
- **Privileged EXEC Level:** Επιτρέπει την εκτέλεση όλων των εντολών και μπορεί να προστατευτεί αυτό το επίπεδο με κωδικούς πρόσβασης.

Το **Global Configuration** είναι ένα κομμάτι του Privileged EXEC Level από το οποίο μπορεί ο διαχειριστής να εκτελέσει εντολές ώστε να επηρεάσει την λειτουργία της εκάστοτε συσκευής [72].

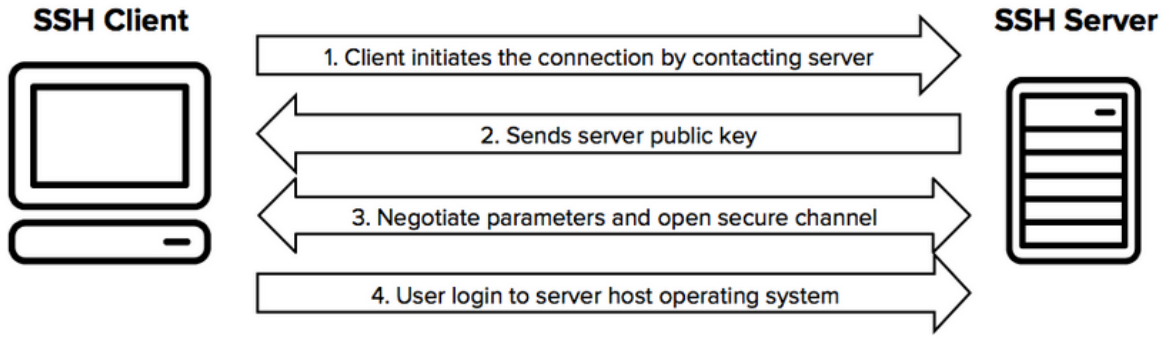


Σχήμα 4.1: Ιεραρχία εντολών Cisco IOS [72]

Δεν θα αναλυθεί περισσότερο το λειτουργικό σύστημα Cisco IOS διότι παρέχει πάρα πολλές δυνατότητες και ξεφεύγει από τους στόχους αυτής της Δ.Ε..

4.3 SSH

Το SSH (Secure Shell) είναι ένα πρωτόκολλο το οποίο προσφέρει την απομακρυσμένη σύνδεση από ένα τερματικό (host) σε ένα άλλο τερματικό (host). Το κύριο πλεονέκτημα που παρέχει το SSH είναι η ασφάλεια και η ισχυρή κρυπτογράφηση των δεδομένων που ανταλλάσσονται μεταξύ δυο σημείων. Το τερματικό (host) που πρόκειται να ξεκινήσει την απομακρυσμένη σύνδεση καλείται SSH client ενώ το τερματικό (host) που δέχεται την απομακρυσμένη σύνδεση καλείται SSH server. Για την σύνδεση του SSH client στον SSH server αρκεί η IP διεύθυνση του SSH server χρησιμοποιώντας την TCP πόρτα 22 καθώς και τα διαπιστευτήρια του εκάστοτε SSH server [73].



Σχήμα 4.2: Διαδικασία σύνδεσης SSH [74]

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την σύνδεση στις δικτυακές συσκευές Cisco από την πλατφόρμα. Επομένως στην προκειμένη περίπτωση ο SSH client είναι το Node.js application (back-end) καθώς και τα python scripts, ενώ ο SSH server είναι η κάθε δικτυακή συσκευή Cisco που ελέγχεται από την πλατφόρμα που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής εργασίας. Η έκδοση που χρησιμοποιήθηκε για το SSH είναι η 2.

4.4 CDP

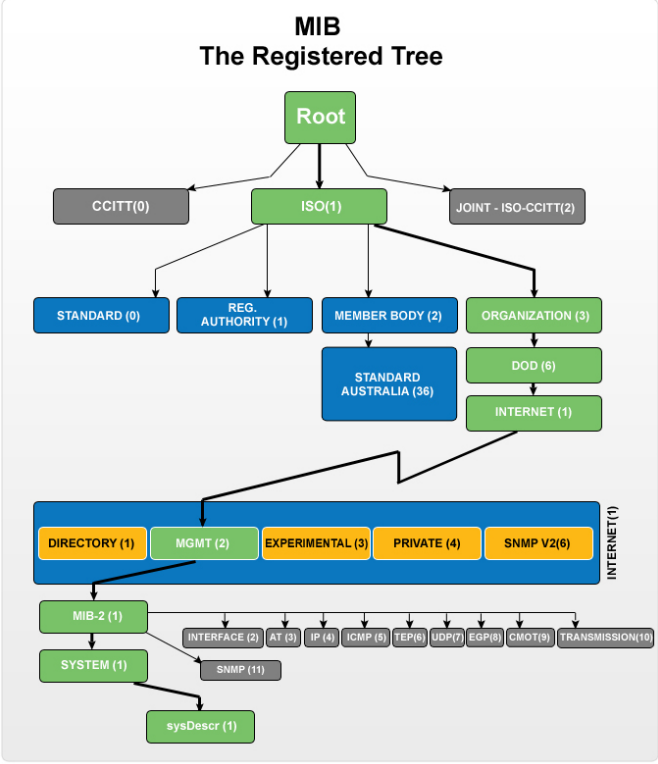
Το CDP (Cisco Discovery Protocol) είναι ένα πρωτόκολλο της Cisco και χρησιμοποιείται για την ανίχνευση των άμεσα συνδεδεμένων συσκευών οι οποίες τρέχουν και αυτές το CDP. Παρέχει πληροφορίες για το λογισμικό και το υλικό των γειτονικών συσκευών όπως είναι για παράδειγμα το όνομα της συσκευής, την διεπαφή (interface) της συσκευής με την οποία είναι συνδεδεμένη και πολλές άλλες πληροφορίες. Την εύρεση γειτόνων την επιτυγχάνει στέλνοντας πακέτα CDP περιοδικά ανά ένα προκαθορισμένο χρονικό διάστημα για το οποίο η προεπιλεγμένη τιμή είναι το 1 λεπτό. Οι πληροφορίες κάθε γείτονα αποθηκεύονται σε έναν πίνακα γειτόνων (neighbors table) και διαγράφονται αν δεν υπάρχει ένδειξη ότι η συσκευή είναι σε θέση να απαντήσει σε αυτά τα CDP πακέτα μέσα σε ένα προκαθορισμένο χρονικό διάστημα [75].

4.5 SNMP

Το SNMP (Simple Network Management Protocol) είναι ένα πρωτόκολλο για την συλλογή και ανταλλαγή πληροφοριών διαχείρισης των δικτυακών συσκευών. Αποτελείται από έναν SNMP Manager ο οποίος είναι υπεύθυνος για την συλλογή πληροφοριών από τις δικτυακές συσκευές καθώς και για την αποστολή εντολών αν χρειαστεί. Ο SNMP Manager είναι ένα application και ελέγχει τις δικτυακές συσκευές στις οποίες έχει υλοποιηθεί ο SNMP Agent. Ο SNMP Agent είναι υπεύθυνος για την συλλογή τοπικών πληροφοριών της εκάστοτε συσκευής, για την αποθήκευσή τους σε μια τοπική βάση δεδομένων και για την αποστολή τους στον SNMP Manager [76].

Το SNMP παρέχει επίσης τα SNMP Traps τα οποία υλοποιούνται στη συσκευή που έχει υλοποιημένο το SNMP Agent και επιτρέπει σε αυτό να ειδοποιεί τον SNMP Manager όταν συμβεί κάποιο συμβάν που ενεργοποιεί το SNMP Trap [76].

Το μήνυμα που φτάνει στον SNMP Manager μετά από την ενεργοποίηση ενός trap είναι ένα OID το οποίο προσδιορίζει τον τύπο της παγίδας καθώς και άλλες πληροφορίες σχετικά με αυτό. Ένα OID (Object Identifier) είναι ένα αντικείμενο που προσδιορίζει κάποια συγκεκριμένα χαρακτηριστικά μιας συσκευής και κάποιες συγκεκριμένες πληροφορίες. Κάθε OID είναι μοναδικό και οργανώνεται με ιεραρχικό τρόπο μέσα στο MIB (Management Information Base). Τέλος κάθε OID είναι μια λίστα ακεραίων οι οποίοι είναι διαχωρισμένοι με την τελεία “.”. Για παράδειγμα ένα OID για ένα interface της συσκευής είναι το “1.3.6.1.2.1.2.2.1” [77],[76].



Σχήμα 4.3: Αναπαράσταση του MIB [76]

Στην παρούσα Δ.Ε. υλοποιήθηκαν SNMP Agents στις δικτυακές συσκευές Cisco και SNMP Manager στο Node.js application (back-end). Χρησιμοποιήθηκε για την ειδοποίηση του Node.js application όταν

αλλάζει η κατάσταση του κάθε interface κάθε δικτυακής συσκευής προκειμένου να ενημερωθεί άμεσα ο χρήστης της πλατφόρμας. Η έκδοση που χρησιμοποιήθηκε είναι η 2.

4.6 SysLog

Το SysLog είναι ένα πρότυπο καταγραφής μηνυμάτων. Η χρησιμότητα του είναι κάθε συσκευή να ενημερώνει τον διαχειριστή για τυχόν συμβάντα με μηνύματα στην κονσόλα της συσκευής. Υπάρχει και η δυνατότητα αποστολής αυτών των μηνυμάτων σε κάποιον SysLog server ο οποίος μπορεί να είναι ένα application. Το SysLog διαχωρίζει τη σημαντικότητα των μηνυμάτων με τα severity levels. Διαθέτει 8 severity levels και είναι τα εξής [78]:

- 0 – Emergencies
- 1 – Alerts
- 2 – Critical
- 3 – Error
- 4 – Warning
- 5 – Notifications
- 6 – Informational
- 7 – Debugging

Στην παρούσα διπλωματική χρησιμοποιήθηκε για την αποστολή της κίνησης κάθε συσκευής στον SysLog που υλοποιήθηκε στο Node.js application.

4.7 Embedded Event Manager

Το EMM (Embedded Event Manager) είναι μια δυνατότητα του Cisco IOS η οποία παρέχει την δυνατότητα στον διαχειριστή να εφαρμόσει μεθόδους προγραμματισμού και αυτοματισμού. Είναι μια λειτουργία που βασίζεται σε στην παρακολούθηση συγκεκριμένων events τα οποία ορίζονται με το EMM και μια σειρά από actions τα οποία θα εκτελεστούν ύστερα από την πυροδότηση του εκάστοτε event [79].

Στην παρούσα διπλωματική χρησιμοποιήθηκε το event timer για την αποστολή της κίνησης ενώ το χρονικό διάστημα που ορίστηκε είναι 1 λεπτό.

4.8 Τεχνικές Επίτευξης Επικοινωνίας

4.8.1 Αναγνώριση Cisco Συσκευής

Κατά την διαδικασία αναζήτησης συσκευών Cisco μέσα σε ένα δίκτυο χρησιμοποιήθηκαν 2 τεχνικές στις οποίες γίνεται χρήση του Nmap (κεφάλαιο 3.5.1). Οι δύο τεχνικές χωρίζονται στην γρήγορη εύρεση της συσκευής και στην αργή εύρεση της συσκευής.

Στην γρήγορη εύρεση γίνεται η εκτέλεση της nmap εντολής στο τερματικό με το όρισμα “-sn”. Αυτό το όρισμα ορίζει στο nmap να μην κάνει σάρωση της θύρας (port scan) λειτουργεία που είναι πολύ αποδοτική από άποψη χρόνου. Εάν βρεθεί ο κατασκευαστής της συσκευής τότε σταματάει η διαδικασία. Εάν δεν βρεθεί ο κατασκευαστής της συσκευής τότε τρέχει η διαδικασία της αργής εύρεσης [80].

Στην αργή εύρεση της συσκευής γίνεται εκτέλεση της nmap εντολής στο τερματικό με το όρισμα “-O”. Αυτό το όρισμα ορίζει στο nmap να κάνει έναν πιο βαθύ έλεγχο και να φέρει περισσότερες πληροφορίες όπως στοιχεία του λειτουργικού συστήματος της συσκευής, τις πόρτες που έχει η συσκευή και άλλα

πολλά. Αυτό διαρκεί περισσότερο χρόνο αλλά στο τέλος μπορεί να βρεθεί αν η συσκευή που ελέγχει είναι CISCO συσκευή [81].

4.8.2 Εύρεση Γειτόνων

Στην διαδικασία της εύρεσης γειτόνων κάθε συσκευής χρησιμοποιήθηκε αποκλειστικά το πρωτόκολλο CDP (κεφάλαιο 4.4). Το γεγονός αυτό προϋποθέτει ότι όλες οι συσκευές που έχουν εισαχθεί στην πλατφόρμα θα χρησιμοποιούν το CDP.

4.8.3 Ειδοποίηση Αλλαγής Κατάστασης Διεπαφών

Στην διαδικασία της ειδοποίησης αλλαγής κατάσταση διεπαφών κάθε συσκευής χρησιμοποιήθηκε αποκλειστικά το SNMP. Χρησιμοποιώντας τα κατάλληλα SNMP traps στις συσκευές αποστέλλονται τα events της αλλαγής κατάστασης κάθε διεπαφής στον SNMP Manager ο οποίος αναπτύχθηκε με το Node.js application.

4.8.4 Κίνηση (Traffic)

Η ανάκτηση της κίνησης των συσκευών γίνεται με τον συνδυασμό της χρήση του SysLog (κεφάλαιο 4.6) και του Embedded Event Manager (κεφάλαιο 4.7) . Πιο συγκεκριμένα χρησιμοποιήθηκε ένα event timer το οποίο τρέχει κάθε 1 λεπτό. Στα actions αυτού του event γίνεται η εκτέλεση της εντολής “show interface” σε κάθε διεπαφή, για την ανάκτηση των πακέτων και των bytes που έχουν εισέλθει και έχουν εξέλθει σε κάθε interface, μέχρι το χρονικό στιγμιότυπο που εκτελείται αυτή η εντολή.

Η κίνηση που προκλήθηκε σε ένα διάστημα χρονικών στιγμιότυπων, από ένα χρονικό στιγμιότυπο “n-1” έως το αμέσως επόμενο χρονικό στιγμιότυπο “n”, μιας διεπαφής “i”, μιας συσκευής προκύπτει από τον τύπο:

$$Traffic_{[i,n]} = (bytesIn + bytesOut)_{[i,n]} - (bytesIn + bytesOut)_{[i,n-1]}$$

(όπου n είναι ένα χρονικό στιγμιότυπο και n-1 είναι το αμέσως προηγούμενο χρονικό στιγμιότυπο που έτρεξαν ξανά τα actions του event. Το αποτέλεσμα της κίνησης μετριέται σε bytes).

Αυτό προϋποθέτει τον κανόνα ότι δεν μπορεί να υπάρξει μέτρηση της κίνησης αν δεν υπάρχουν τουλάχιστον δύο χρονικά στιγμιότυπα.

4.9 Παραμετροποίηση Συσκευών CISCO

4.9.1 Εισαγωγή

Στο παρόν κεφάλαιο θα αναφερθούν όλες οι εντολές που χρειάστηκαν να χρησιμοποιηθούν για την ανάπτυξη της πλατφόρμας. Οι εντολές αυτές χωρίζονται σε εντολές ανάκτησης πληροφοριών και σε εντολές αρχικοποίησης συσκευών.

4.9.2 Εντολές Ανάκτησης Πληροφοριών

Η πληροφορίες που χρειάστηκαν να ανακτηθούν από τις συσκευές για την ανάπτυξη της πλατφόρμας που παρουσιάζονται σε αυτή την διπλωματική εργασία είναι οι εξής:

- **Running Configuration:** show running-config
- **Startup Configuration:** show startup-config

- **Πληροφορίες όλων των Interfaces:** show interfaces
- **Πληροφορίες Λειτουργικού Συστήματος και Συσκευής:** show version
- **Πίνακας δρομολόγησης:** show ip route
- **Πίνακας ARP:** show ip arp
- **Access Lists:** show access-list
- **Πληροφορίες γειτόνων CDP:** show cdp neighbors detail
- **Πληροφορίες EIGRP:** show ip eigrp topology
- **Πληροφορίες OSPF:** show ip ospf database router
- **Πληροφορίες STP:** show spanning-tree
- **Πίνακας διευθύνσεων MAC:** show mac address-table

4.9.3 Εντολές Αρχικοποίησης Συσκευών

Προκειμένου οι συσκευές να είναι έτοιμες για χρήση από την πλατφόρμα χρειάζεται πρώτα η εκτέλεση των ακόλουθων εντολών οι οποίες στέλνονται μέσω των python scripts (κεφάλαιο 6):

- Υλοποίηση του SNMP Agent:
  - snmp-server community AutoNet RW
  - snmp-server host [SNMP Manager IP] version 2c AutoNet
  - snmp-server trap link ietf
  - snmp-server enable traps snmp authentication linkdown linkup coldstart warmstart
- Υλοποίηση του SysLog Client:
  - logging host [SysLog Server IP]
  - logging monitor xml
- Υλοποίηση CDP: cdp run
- Ενεργοποίηση CDP σε κάθε Interface:
  - interface [Όνομα interface]
  - cdp enable
  - exit
- Υλοποίηση EMM και Μηχανισμού Αποστολή Κίνησης:
  - event manager applet AutonetTraffic
  - event timer watchdog time 60
  - action 100 cli command "en"
  - action 101 cli command "show interface [Όνομα interface] | in packets input"
  - action 102 set inTr "$_cli_result"
  - action 103 cli command "show interface [Όνομα interface] | in packets output"
  - action 104 set outTr "$_cli_result"
  - action 105 puts "[Όνομα interface] $inTr"
  - action 106 puts "[Όνομα interface] $outTr"

Όλα τα actions που χρειάστηκαν για την υλοποίηση του EMM εφαρμόζονται για κάθε interface της συσκευής.

4.10 Επίλογος

Στο παρόν κεφάλαιο αναλύθηκαν τα πρωτόκολλα και οι τεχνολογίες δικτύωσης που εφαρμόστηκαν. Στο επόμενο κεφάλαιο θα παρουσιαστεί η βάση δεδομένων που αναπτύχθηκε στα πλαίσια της υλοποίησης του AutoNet.

Κεφάλαιο 5ο: Ανάλυση Βάσης Δεδομένων

5.1 Εισαγωγή

Στο παρόν κεφάλαιο θα αναλυθούν όλα τα σχήματα των δεδομένων που αποθηκεύονται στην βάση δεδομένων. Η βάση δεδομένων είναι η MongoDB και τα σχήματα των δεδομένων που θα αναλυθούν είναι οι μορφές των λεγόμενων documents της βάσης. Τα documents ομαδοποιούνται σε Συλλογές (Collections).

5.2 Users

Στην συλλογή Users αποθηκεύονται τα δεδομένα των χρηστών που χρησιμοποιούν αυτή την πλατφόρμα. Κάθε χρήστης περιέχει τα εξής πεδία:

- **_id:** Αυτό το πεδίο το προσθέτει από προεπιλογή η MongoDB, είναι τύπου ObjectId (ένας τύπος στην MongoDB) και καθορίζει το αναγνωριστικό του χρήστη
- **username:** Αυτό το πεδίο είναι τύπου αλφαριθμητικό (String). Προσδιορίζει το όνομα χρήστη του χρήστη, είναι μοναδικό για κάθε χρήστη και αποτελεί ένα από τα διαπιστευτήρια του κάθε χρήστη.
- **password:** Αυτό το πεδίο είναι τύπου αλφαριθμητικό (String), προσδιορίζει τον κωδικό πρόσβασης του χρήστη και αποτελεί και αυτό αποτελεί ένα από τα διαπιστευτήρια του κάθε χρήστη.
- **email:** Είναι τύπου αλφαριθμητικό (String) και προσδιορίζει το email του χρήστη
- **name:** Είναι τύπου αλφαριθμητικό (String) και αντιπροσωπεύει το όνομα του χρήστη
- **surname:** Αυτό το πεδίο είναι τύπου αλφαριθμητικό (String) και αντιπροσωπεύει το επίθετο του χρήστη
- **date:** Είναι τύπου ημερομηνίας (Date) και προσδιορίζει την ημερομηνία που εγγράφηκε ο χρήστης.

5.3 Networks

Στην συλλογή Networks αποθηκεύονται τα δεδομένα των δικτύων που έχουν εισαχθεί από τους χρήστες σε αυτή την πλατφόρμα. Κάθε δίκτυο περιέχει τα εξής πεδία:

- **_id:** Αυτό το πεδίο το προσθέτει από προεπιλογή η MongoDB, είναι τύπου ObjectId και καθορίζει το αναγνωριστικό του δικτύου
- **user:** Αυτό το πεδίο είναι τύπου ObjectId. Προσδιορίζει το αναγνωριστικό (_id) του χρήστη που μπορεί να προβάλει πληροφορίες και να κάνει ενέργειες σε αυτό το δίκτυο.
- **ipNetwork:** Είναι ένα αντικείμενο (Object) το οποίο αντιπροσωπεύει την IP διεύθυνση του δικτύου και περιέχει τα πεδία:
    - **name:** Η λεζάντα που θα φαίνεται στον χρήστη
    - **value:** Η τιμή της διεύθυνσης δικτύου.
- **ipAddress:** Είναι ένα αντικείμενο (Object) το οποίο αντιπροσωπεύει την IP διεύθυνση του δικτύου χωρίς την μάσκα και περιέχει τα πεδία:
    - **name:** Η λεζάντα που θα φαίνεται στον χρήστη
    - **value:** Η τιμή της διεύθυνσης δικτύου χωρίς την μάσκα.
- **firstAddress:** Είναι ένα αντικείμενο (Object) το οποίο αντιπροσωπεύει την πρώτη διαθέσιμη IP διεύθυνση του δικτύου που μπορεί να δοθεί σε ένα κόμβο και περιέχει τα πεδία:
    - **name:** Η λεζάντα που θα φαίνεται στον χρήστη
    - **value:** Η τιμή της πρώτης διαθέσιμης διεύθυνσης του δικτύου

- **lastAddress:** Είναι ένα αντικείμενο (Object) το οποίο αντιπροσωπεύει την τελευταία διαθέσιμη IP διεύθυνση του δικτύου που μπορεί να δοθεί σε ένα κόμβο και περιέχει τα πεδία:
    - **name:** Η λεζάντα που θα φαίνεται στον χρήστη
    - **value:** Η τιμή της τελευταίας διαθέσιμης διεύθυνσης του δικτύου
- **broadcastAddress:** Είναι ένα αντικείμενο (Object) το οποίο αντιπροσωπεύει την broadcast IP διεύθυνση του δικτύου, δεν μπορεί να δοθεί σε ένα κόμβο και περιέχει τα πεδία:
    - **name:** Η λεζάντα που θα φαίνεται στον χρήστη
    - **value:** Η τιμή της broadcast διεύθυνσης του δικτύου
- **subnetMask:** Είναι ένα αντικείμενο (Object) το οποίο αντιπροσωπεύει την μάσκα υποδικτύου και περιέχει τα πεδία:
    - **name:** Η λεζάντα που θα φαίνεται στον χρήστη
    - **value:** Η τιμή της διεύθυνσης υποδικτύου
- **subnetMaskLength:** Είναι ένα αντικείμενο (Object) το οποίο αντιπροσωπεύει το μέγεθος της μάσκα υποδικτύου και περιέχει τα πεδία:
    - **name:** Η λεζάντα που θα φαίνεται στον χρήστη
    - **value:** Η τιμή της του μεγέθους της μάσκας υποδικτύου
- **numHosts:** Είναι ένα αντικείμενο (Object) το οποίο αντιπροσωπεύει τον μέγιστο πλήθος των κόμβων που μπορούν να πάρουν διεύθυνση IP από το εκάστοτε δίκτυο:
    - **name:** Η λεζάντα που θα φαίνεται στον χρήστη
    - **value:** Η τιμή της του μεγέθους των host που χωράνε στο δίκτυο

5.4 Nodes

Στην συλλογή Nodes αποθηκεύονται τα δεδομένα των συσκευών που έχουν εισαχθεί από τους χρήστες σε αυτή την πλατφόρμα. Κάθε συσκευή περιέχει τα εξής πεδία:

- **_id:** Αυτό το πεδίο το προσθέτει από προεπιλογή η MongoDB, είναι τύπου ObjectId και καθορίζει το αναγνωριστικό της συσκευής
- **user:** Αυτό το πεδίο είναι τύπου ObjectId. Προσδιορίζει το αναγνωριστικό (_id) του χρήστη που μπορεί να προβάλει πληροφορίες και να κάνει ενέργειες σε αυτή την συσκευή.
- **username:** Είναι ένα αντικείμενο (Object) το οποίο αντιπροσωπεύει το όνομα χρήστη της συσκευής για την σύνδεση με SSH και περιέχει τα πεδία:
    - **name:** Η λεζάντα που θα φαίνεται στον χρήστη
    - **value:** Η τιμή του ονόματος χρήστη.
    - **editable:** Είναι μια Boolean τιμή που δείχνει αν μπορεί να επεξεργαστεί από τον χρήστη ή όχι
- **password:** Είναι ένα αντικείμενο (Object) το οποίο αντιπροσωπεύει τον κωδικό πρόσβασης της συσκευής για την σύνδεση με SSH και περιέχει τα πεδία:
    - **name:** Η λεζάντα που θα φαίνεται στον χρήστη
    - **value:** Η τιμή του κωδικού πρόσβασης.
    - **editable:** Είναι μια Boolean τιμή που δείχνει αν μπορεί να επεξεργαστεί από τον χρήστη ή όχι
- **name:** Είναι ένα αντικείμενο (Object) το οποίο αντιπροσωπεύει το όνομα της συσκευής και περιέχει τα πεδία:
    - **name:** Η λεζάντα που θα φαίνεται στον χρήστη
    - **value:** Η τιμή του ονόματος της συσκευής.
    - **editable:** Είναι μια Boolean τιμή που δείχνει αν μπορεί να επεξεργαστεί από τον χρήστη ή όχι
- **vendor:** Είναι ένα αντικείμενο (Object) το οποίο αντιπροσωπεύει τον κατασκευαστή της συσκευής και περιέχει τα πεδία:
    - **name:** Η λεζάντα που θα φαίνεται στον χρήστη
    - **value:** Η τιμή του ονόματος του κατασκευαστή της συσκευής.

    - **editable:** Είναι μια Boolean τιμή που δείχνει αν μπορεί να επεξεργαστεί από τον χρήστη ή όχι
- **type:** Είναι ένα αντικείμενο (Object) το οποίο αντιπροσωπεύει τον τύπο της συσκευής και περιέχει τα πεδία:
    - **name:** Η λεζάντα που θα φαίνεται στον χρήστη
    - **value:** Η τιμή του τύπου της συσκευής.
    - **editable:** Είναι μια Boolean τιμή που δείχνει αν μπορεί να επεξεργαστεί από τον χρήστη ή όχι
- **status:** Είναι ένα αντικείμενο (Object) το οποίο αντιπροσωπεύει την κατάσταση της συσκευής και περιέχει τα πεδία:
    - **name:** Η λεζάντα που θα φαίνεται στον χρήστη
    - **value:** Η Boolean τιμή της κατάστασης της συσκευής.
    - **editable:** Είναι μια Boolean τιμή που δείχνει αν μπορεί να επεξεργαστεί από τον χρήστη ή όχι
- **model:** Είναι ένα αντικείμενο (Object) το οποίο αντιπροσωπεύει το μοντέλο της συσκευής και περιέχει τα πεδία:
    - **name:** Η λεζάντα που θα φαίνεται στον χρήστη
    - **value:** Η τιμή του μοντέλου της συσκευής.
    - **editable:** Είναι μια Boolean τιμή που δείχνει αν μπορεί να επεξεργαστεί από τον χρήστη ή όχι
- **upTime:** Είναι ένα αντικείμενο (Object) το οποίο αντιπροσωπεύει τον χρόνο που είναι ενεργή η συσκευής και περιέχει τα πεδία:
    - **name:** Η λεζάντα που θα φαίνεται στον χρήστη
    - **value:** Η τιμή του χρόνου που είναι ενεργή η συσκευής.
    - **editable:** Είναι μια Boolean τιμή που δείχνει αν μπορεί να επεξεργαστεί από τον χρήστη ή όχι
- **os:** Είναι ένα αντικείμενο (Object) το οποίο αντιπροσωπεύει την έκδοση του λειτουργικού συστήματος Cisco IOS της συσκευής και περιέχει τα πεδία:
    - **name:** Η λεζάντα που θα φαίνεται στον χρήστη
    - **version:** Η τιμή της έκδοσης του Cisco IOS.
- **runConf:** Είναι ένα αλφαριθμητικό (String) το οποίο αντιπροσωπεύει το running configuration της συσκευής.
- **startupConf:** Είναι ένα αλφαριθμητικό (String) το οποίο αντιπροσωπεύει το startup configuration της συσκευής.
- **serial:** Είναι ένα αλφαριθμητικό (String) το οποίο αντιπροσωπεύει το serial number της συσκευής.
- **interfaces:** Είναι ένας πίνακας (Array) ο οποίος αντιπροσωπεύει το σύνολο των διεπαφών που διαθέτει η συσκευής.
- **route_table:** Είναι ένας πίνακας (Array) ο οποίος αντιπροσωπεύει τον πίνακα δρομολόγησης της συσκευής
- **arp_table:** Είναι ένας πίνακας (Array) ο οποίος αντιπροσωπεύει τον πίνακα ARP της συσκευής
- **acl:** Είναι ένας πίνακας (Array) ο οποίος αντιπροσωπεύει τα access lists της συσκευής.
- **cdp:** Είναι ένας πίνακας (Array) ο οποίος αντιπροσωπεύει τον πίνακα των γειτόνων CDP της συσκευής
- **stp:** Είναι ένας πίνακας (Array) ο οποίος αντιπροσωπεύει τον πίνακα του spanning tree protocol της συσκευή
- **mac:** Είναι ένας πίνακας (Array) ο οποίος αντιπροσωπεύει τον πίνακα MAC της συσκευής.
- **eigrp:** Είναι ένας πίνακας (Array) ο οποίος αντιπροσωπεύει το eigrp topology table της συσκευής
- **ospf:** Είναι ένας πίνακας (Array) ο οποίος αντιπροσωπεύει το ospf table της συσκευής.

5.5 Topologies

Στην συλλογή Topologies αποθηκεύονται τα δεδομένα των τοπολογιών που δημιουργήθηκαν από τους χρήστες της πλατφόρμας. Κάθε τοπολογία περιέχει τα εξής πεδία:

- **_id:** Αυτό το πεδίο το προσθέτει από προεπιλογή η MongoDB, είναι τύπου ObjectId (ένας τύπος στην MongoDB) και καθορίζει το αναγνωριστικό της τοπολογίας
- **user:** Αυτό το πεδίο είναι τύπου ObjectId. Προσδιορίζει το αναγνωριστικό (_id) του χρήστη που μπορεί να προβάλει πληροφορίες και να κάνει ενέργειες σε αυτή την τοπολογία.
- **name:** Είναι τύπου αλφαριθμητικό (String) και αντιπροσωπεύει το όνομα της τοπολογίας.
- **nodes:** Αυτό το πεδίο είναι τύπου πίνακα (Array), αντιπροσωπεύει τις συσκευές που περιέχονται στην τοπολογία και περιέχει τα εξής πεδία:
    - **id:** Προσδιορίζει το id της συσκευής
    - **x:** Προσδιορίζει την θέση της συσκευής στην οθόνη ως προς τον άξονα X
    - **y:** Προσδιορίζει την θέση της συσκευής στην οθόνη ως προς τον άξονα Y
    - **label:** Περιέχει τα πεδία x και y και προσδιορίζει την θέση του ονόματος της συσκευής στην οθόνη

5.6 Servers

Στην συλλογή Servers αποθηκεύονται τα δεδομένα του server της πλατφόρμας. Ο server περιέχει τα εξής πεδία:

- **_id:** Αυτό το πεδίο το προσθέτει από προεπιλογή η MongoDB, είναι τύπου ObjectId (ένας τύπος στην MongoDB) και καθορίζει το αναγνωριστικό του server
- **timestamp:** Αυτό το πεδίο είναι τύπου Double. Προσδιορίζει το χρονικό στιγμιότυπο όπου ο server ξεκίνησε να λειτουργεί.
- **address:** Είναι ένα αντικείμενο (Object) το οποίο αντιπροσωπεύει τις πληροφορίες της διεύθυνσης του server και περιέχει τα πεδία:
    - **ip:** Η τιμή της IP διεύθυνσης του server
    - **port:** Η τιμή της TCP θύρας του backend application.
    - **frontPort:** Η τιμή της TCP θύρας του frontend application.

5.7 Επίλογος

Στο παρόν κεφάλαιο παρουσιάστηκε η βάση δεδομένων που χρησιμοποιήθηκε για την αποθήκευση των δεδομένων. Στο επόμενο κεφάλαιο θα αναλυθούν τα python scripts που υλοποιήθηκαν προκειμένου να επιτευχθεί η επικοινωνία με τις συσκευές.

Κεφάλαιο 6ο: Python Scripts

6.1 Εισαγωγή

Στο παρόν κεφάλαιο θα αναλυθεί ο κώδικας που υλοποιήθηκε προκειμένου να αναπτυχθούν τα python scripts τα οποία είναι υπεύθυνα για την αποστολή εντολών στις δικτυακές συσκευές. Για κάθε python script αποφασίστηκε ότι όλα τα αποτελέσματα θα είναι τύπου json για να μην υπάρχει σύγχυση και να υπάρχει μια ομοιομορφία. Επιπρόσθετα για την σύνδεση με SSH στις συσκευές χρησιμοποιήθηκε η μέθοδος ConnectHandler της βιβλιοθήκης netmiko ενώ για την αποστολή εντολών στο configuration level χρησιμοποιήθηκε η μέθοδος send_config_set που προέκυψε από την ConnectHandler. Τέλος αφού έχουν πραγματοποιηθεί οι ενέργειες που έπρεπε χρησιμοποιήθηκε η μέθοδος disconnect προκειμένου να τερματιστεί η σύνδεση.

```
cisco_881 = {
    'device_type': 'cisco_ios',
    'host':   sys.argv[1],
    'username': sys.argv[2],
    'password': sys.argv[3]
}
net_connect = ConnectHandler(**cisco_881)
```

Σχήμα 6.1: Κώδικας υλοποίησης σύνδεσης με την συσκευή

6.2 Εύρεση IP Διευθύνσεων Δικτύου

Προκειμένου να βρεθούν όλες οι διαθέσιμες διευθύνσεις IP μέσα σε ένα δίκτυο χρησιμοποιήθηκε η βιβλιοθήκη ipaddress. Πιο συγκεκριμένα χρησιμοποιώντας την μέθοδο hosts της μεθόδου ip_network της βιβλιοθήκης καταφέρνουμε να πάρουμε όλες τις διαθέσιμες IP διευθύνσεις που μπορούν να δοθούν σε τερματικά (hosts). Αυτό επιτυγχάνεται ύστερα από ένα όρισμα που έχει δοθεί κατά το κάλεσμα της εκτέλεσης του script το οποίο συμβολίζει μια IP διεύθυνση δικτύου.

```
x = list(ipaddress.ip_network(sys.argv[1]).hosts())

ips = []

for i in range(len(x)):
    ips.append({ "ip": str(x[i]) })

print(json.dumps(ips))
sys.stdout.flush()
```

Σχήμα 6.2: Κώδικας εύρεσης IP διευθύνσεων μέσα σε μια IP διεύθυνση δικτύου

6.3 Αναζήτηση Συσκευών Cisco

Στην αναζήτηση συσκευών χρησιμοποιήθηκε η βιβλιοθήκη nmap. Όπως έχει αναλυθεί στο κεφάλαιο 4.8.1 η αναζήτηση συσκευών γίνεται με την χρήση των ορισμάτων “-sn” και “-o” στο nmap, για την γρήγορη και την αργή αναζήτηση αντίστοιχα. Με την μέθοδο PortScanner της βιβλιοθήκης nmap εντοπίζεται το nmap στο σύστημα και αποθηκεύονται οι λειτουργίες του στην μεταβλητή nm. Εάν βρεθεί το nmap στο σύστημα τότε με την χρήση της μεθόδου scan γίνεται η αναζήτηση των συσκευών του δικτύου και σε αυτό το σημείο ορίζεται και το κατάλληλο όρισμα του nmap. Σε αυτή την ενέργεια ανακτώνται κυρίως οι αναγνωριστικές πληροφορίες της συσκευής όπως ο κατασκευαστής της συσκευής

καθώς και άλλες πληροφορίες όπως η MAC διεύθυνση, το όνομα της συσκευής και ο τύπος της συσκευής.

```
nm = nmap.PortScanner()
nm.scan(sys.argv[1], arguments='-sn')


try:
    nm.scan(sys.argv[1], arguments='-sn')
    node = { "ip" : nm[sys.argv[1]]["addresses"]["ipv4"] }
    try:
        mac = nm[sys.argv[1]]["addresses"]["mac"]
        try:
            node = { "ip" : nm[sys.argv[1]]["addresses"]["ipv4"],
                     "mac" : mac,
                     "vendor" : nm[sys.argv[1]]["vendor"][mac],
                     "name": nm[sys.argv[1]]["hostnames"][0]["name"],
                     "type" : nm[sys.argv[1]]["hostnames"][0]["type"]}
        except:
                node = { "ip" : nm[sys.argv[1]]["addresses"]["ipv4"],
                         "mac" : mac,
                         "vendor" : None,
                         "name": nm[sys.argv[1]]["hostnames"][0]["name"],
                         "type" : nm[sys.argv[1]]["hostnames"][0]["type"]}

    except:
        node = { "ip" : nm[sys.argv[1]]["addresses"]["ipv4"],
                 "mac" : None,
                 "vendor" : None,
                 "name": nm[sys.argv[1]]["hostnames"][0]["name"],
                 "type" : nm[sys.argv[1]]["hostnames"][0]["type"]}
except:
    node = { "ip" : ""}
```

Σχήμα 6.3: Κώδικας γρήγορης εύρεσης συσκευών

6.4 Αρχικοποίηση Συσκευών Cisco

Στην διεργασία της αρχικοποίησης των συσκευών προκειμένου να είναι έτοιμες για χρήση από την πλατφόρμα αποστέλλονται όλες οι εντολές που αναλύθηκαν στο κεφάλαιο 4.9.3. Για την αρχικοποίηση του EMM σε κάθε interface της συσκευής χρησιμοποιήθηκε μια λούπα. Μετά την αρχικοποίηση του EMM για κάθε διεπαφή, ορίζεται στην συσκευή να στέλνει ένα μήνυμα “Traffic_Finish” προκειμένου να γνωρίζει ο SysLog server ότι η αποστολή της κίνησης των interfaces της συσκευής τελείωσε.

```
i = 100
flag = True
for interface in interfaces:

    config_commands = []
    config_commands.append('event manager applet AutonetTraffic')
    config_commands.append('event timer watchdog time 60')

    if flag:
        config_commands.append('action '+ str(i) + ' cli command "en"')
        i += 1
        flag = False

    config_commands.append('action '+ str(i) + ' cli command "show interface ' + interface["interface"] + ' | in packets input"')
    i += 1
    config_commands.append('action '+ str(i) + ' set inTr "$_cli_result"')
    i += 1
    config_commands.append('action '+ str(i) + ' cli command "show interface ' + interface["interface"] + ' | in packets output"')
    i += 1
    config_commands.append('action '+ str(i) + ' set outTr "$_cli_result"')
    i += 1
    config_commands.append('action '+ str(i) + ' puts "' + interface["interface"] + ' $inTr"')
    i += 1
    config_commands.append('action '+ str(i) + ' puts "' + interface["interface"] + ' $outTr"')
    i += 1
    config_commands.append('exit')
    output = net_connect.send_config_set(config_commands)

config_commands = []
config_commands.append('event manager applet AutonetTraffic')
config_commands.append('event timer watchdog time 60')
config_commands.append('action '+ str(i) + ' puts Traffic_Finish')
config_commands.append('exit')
output = net_connect.send_config_set(config_commands)
```

Σχήμα 6.4: Κώδικας εντολών αρχικοποίησης συσκευής

6.5 Ανάκτηση Πληροφορίων Συσκευών Cisco

Στην διεργασία της ανάκτησης πληροφοριών από τις συσκευές με στόχο την παροχή των πληροφοριών στους χρήστες της πλατφόρμας, αποστέλλονται όλες οι εντολές που αναλύθηκαν στο κεφάλαιο 4.9.2. Η αποστολή των εντολών γίνονται με την μέθοδο send_command της μεθόδου ConnectHandler η οποία έχει εισαχθεί από την βιβλιοθήκη netmiko. Προκειμένου να γίνεται η ανάκτηση πληροφοριών σε μορφή αντικειμένου (object) ορίζεται η παράμετρος use_textfsm σε true η οποία ορίζει την μέθοδο send_command να χρησιμοποιεί τα ntc_templates (κεφάλαιο 3.5.7). Μετά από την παραλαβή των αποτελεσμάτων, εκτελείται μια σειρά από μεθόδους όπου στόχο έχουν την τροποποίηση των αποτελεσμάτων ώστε να εξυπηρετούν τους σκοπούς της εφαρμογής.

```
runConf = net_connect.send_command('show running-config', use_textfsm=True)
startConf = net_connect.send_command('show startup-config', use_textfsm=True)
interfaces = net_connect.send_command('show interfaces', use_textfsm=True)
version = net_connect.send_command('show version', use_textfsm=True)
typeNode = 'Switch'
routeTable = net_connect.send_command('show ip route', use_textfsm=True)
arpTable = net_connect.send_command('show ip arp', use_textfsm=True)
acl = net_connect.send_command('show access-list', use_textfsm=True)
cdp = net_connect.send_command('show cdp neighbors detail', use_textfsm=True)
eigrp = net_connect.send_command('show ip eigrp topology', use_textfsm=True)
ospf= net_connect.send_command('show ip ospf database router', use_textfsm=True)
stp = net_connect.send_command('show spanning-tree', use_textfsm=True)
os = net_connect.send_command('show version')
mac = net_connect.send_command('show mac address-table', use_textfsm=True)
```

Σχήμα 6.5: Κώδικας εντολών για ανάκτηση πληροφοριών από τις συσκευές

6.6 Αποστολή Εντολών Τροποποίησης

Μέσω αυτής της διεργασίας επιτρέπεται στον χρήστη της πλατφόρμας να στείλει εντολές οι οποίες αλλάζουν το configuration της συσκευής. Προς το παρόν δίνεται μόνο η δυνατότητα τροποποίησης των διαπιστευτηρίων του χρήστη της συσκευής, αλλά έχει υλοποιηθεί με τέτοιο τρόπο ώστε μελλοντικά να μπορούν εύκολα να προστεθούν και άλλες λειτουργίες. Πιο συγκεκριμένα ελέγχεται το 4ο όρισμα που έχει δοθεί, και με βάση αυτό γίνεται εφικτός ο διαχωρισμός του είδους της ενέργειας που επιθυμεί να πραγματοποιηθεί.

```
config_commands = []

if sys.argv[4] == 'credentials':
    config_commands = ["username " + sys.argv[5]  + " privilege 15 password " + sys.argv[6],]


if len(config_commands) > 0:
    output = net_connect.send_config_set(config_commands)

net_connect.send_command('wr')
net_connect.disconnect()
```

Σχήμα 6.6: Κώδικας αποστολής εντολών τροποποίησης των συσκευών

6.7 Επίλογος

Στο κεφάλαιο αυτό αναλύθηκαν όλες οι εντολές και οι τεχνικές που χρησιμοποιήθηκαν για την επικοινωνία με τις συσκευές Cisco, με χρήση της γλώσσας προγραμματισμού Python. Τα python scripts αποτελούν το μέσο επικοινωνίας του Node.js server με τις δικτυακές συσκευές. Αυτό σημαίνει ότι τώρα θα αναλυθεί η back-end εφαρμογή καθώς αποτελεί το επόμενο βήμα της διπλωματικής εργασίας.

Κεφάλαιο 7ο: Εφαρμογή Back-end

7.1 Εισαγωγή

Στο παρόν κεφάλαιο θα αναλυθεί ο κώδικας που υλοποιήθηκε προκειμένου να αναπτυχθεί η back-end εφαρμογή η οποία είναι υπεύθυνη για την επικοινωνία με τις συσκευές με την εκτέλεση των python scripts (κεφάλαιο 6) και άλλων τεχνικών, τη αποθήκευση των δεδομένων στην βάση δεδομένων και την αποστολή των δεδομένων στη Frontend εφαρμογή (κεφάλαιο 8). Η back-end εφαρμογή έχει χτιστεί πάνω στο Node.js.

7.2 Διαδικασία Σύνδεσης

Προκειμένου να μπορεί ο κάθε χρήστης να συνδεθεί στην πλατφόρμα δημιουργήθηκε ένα rest api endpoint το οποίο περιμένει αιτήματα σύνδεσης ώστε να τα εξυπηρετήσει. Πιο συγκεκριμένα περιμένει αιτήματα τύπου POST στην διαδρομή /user/login. Με την βοήθεια του Router που παρέχει η Express η υλοποίηση του ήταν σχετικά απλή.

Όταν φτάνει ένα request στον Node.js server τότε αρχικά ελέγχονται αν τα δεδομένα του request είναι έγκυρα με την βοήθεια του Joi. Αμέσως μετά γίνεται έλεγχος στην βάση δεδομένων Mongo DB, αν υπάρχει το όνομα χρήστη που πρόκειται να συνδεθεί στην πλατφόρμα. Εφόσον περάσει και αυτόν τον έλεγχο τότε δημιουργείται το hash μέσω του κωδικού πρόσβασης που δόθηκε χρησιμοποιώντας τη μέθοδο compare του bcrypt. Αυτή η μέθοδος συγκρίνει το hash που δημιουργήθηκε από τον κωδικό πρόσβασης που περιέχεται μέσα στο request, με το hash του κωδικού πρόσβασης του χρήστη που υπάρχει στην βάση δεδομένων. Αν τα δύο hash είναι ίδια τότε επιτρέπεται η σύνδεση στον χρήστη, δημιουργείται το token του και αποστέλλεται πίσω σαν απάντηση στο αίτημα. Αυτό το token επιτρέπει στον χρήστη να διατηρείται συνδεδεμένος στο σύστημα. Το token δημιουργείται από το id του χρήστη που υπάρχει στην βάση δεδομένων και από το TOKEN_SECRET το οποίο είναι ένα τυχαίο αλφαριθμητικό (string), που βρίσκεται στο .env αρχείο. Για την επίτευξη της δημιουργίας του token χρησιμοποιήθηκε η jwt βιβλιοθήκη. Τέλος στην περίπτωση που αποτύχει σε κάποιο σημείο η επαλήθευση των διαπιστευτηρίων, που περιέχονται στο αίτημα, στέλνεται μια απάντηση σφάλματος στην front-end εφαρμογή.

```
router.post('/login', async (req,res) => {
    let data = {};
    Object.keys(req.query).length === 0 ? data = req.body : data = req.query;

    // Validate Request Data
    const { error } = loginValidation(data);
    if (error) return res.status(400).send(error.details[0].message);

    // Check if request Username exists
    const user = await User.findOne({ username: data.username });
    if(!user) return res.status(400).send("Username or Password is wrong");

    // Check if request Password is correct
    const validPass = await bcrypt.compare(data.password, user.password);
    if(!validPass) return res.status(400).send("Username or Password is wrong");

    // Create user Token
    const token = jwt.sign({_id: user._id}, process.env.TOKEN_SECRET)

    res.header('auth-token', token).send(token);

});
```

Σχήμα 7.1: Κώδικας υλοποίησης api endpoint για την σύνδεση των χρηστών

Πριν περάσουμε στην επόμενη ενότητα αξίζει να σημειωθεί ότι έχει υλοποιηθεί και η διαδικασία της εγγραφής ενός καινούργιου χρήστη και είναι έτοιμη να χρησιμοποιηθεί μελλοντικά όταν θα είναι έτοιμη και η front-end εφαρμογή να υποστηρίξει αυτή την δυνατότητα.

7.3 Εύρεση και Αποθήκευση Συσκευής

Το επόμενο βήμα είναι να αναλυθεί η λογική που βασίζεται η διαδικασία της εύρεσης συσκευής. Το συγκεκριμένο κομμάτι ήταν το πιο δύσκολο και το πιο περίπλοκο να υλοποιηθεί καθώς αποφασίστηκε να χρησιμοποιηθούν διάφορες τεχνικές προκειμένου να ελαττωθεί στο ελάχιστο η πιθανότητα να μην βρεθεί μια συσκευή η οποία είναι ενεργή.

Προκειμένου να γίνεται σε πραγματικό χρόνο η ενημέρωση του χρήστη για τα αποτελέσματα κάθε συσκευής χρησιμοποιήθηκε ο συνδυασμός του Router της Express και την χρήση των web sockets της βιβλιοθήκης Socket IO. Πιο συγκεκριμένα έχει υλοποιηθεί ένα rest api endpoint στην διαδρομή /api/devices. Τα αιτήματα που αναμένονται να φτάσουν περιέχουν μια διεύθυνση IP. Με την χρήση της βιβλιοθήκης IP παίρνουμε την διεύθυνση δικτύου. Ο σκοπός είναι η εύρεση όλων των συσκευών Cisco (routers και switches) μέσα στο εκάστοτε δίκτυο.

Το πρώτο βήμα για να πραγματοποιηθεί αυτό είναι να βρεθούν όλες οι διαθέσιμες IP διευθύνσεις που μπορούν να δοθούν σε κόμβους. Αυτό πραγματοποιείται με την εκτέλεση του python script για την εύρεση των IP διευθύνσεων σε ένα δίκτυο (κεφάλαιο 6.2).

Αφού βρεθούν οι IP διευθύνσεις, με την χρήση της μεθόδου sys.probe της βιβλιοθήκης ping, ελέγχεται σε ποιες από αυτές τις IP διευθύνσεις υπάρχουν ενεργές συσκευές. Για κάθε συσκευή που βρέθηκε εκτελείται η διαδικασία της γρήγορης εύρεσης συσκευής (κεφάλαιο 6.3). Αν η απάντηση του python script που εκτελεί αυτή την διαδικασία επιστρέψει την πληροφορία ότι ο κατασκευαστής της συσκευής είναι η Cisco τότε αποστέλλεται μέσω των sockets στον χρήστη. Σε αντίθετη περίπτωση εκτελείται η αργή εύρεση συσκευής (κεφάλαιο 6.3) και ακολουθεί η ίδια διαδικασία. Στην περίπτωση που ο κατασκευαστής δεν είναι η Cisco ή δεν μπόρεσε να βρεθεί τότε ενημερώνεται και πάλι ο χρήστης για την εξέλιξη αυτή μέσω των sockets.

```
function hostsFinder(ips, socketid, completeScan)  {
    return new Promise(resolve => {
        let count = 0;
        let hosts = [];
        let deadHosts = [];
        ips.forEach(function(host){
            ping.sys.probe(host.ip, async function(isAlive){
                count++;
                if(isAlive){
                    let aliveHost = await deviceFinderFast(host.ip);
                    if(aliveHost.vendor == null){
                        completeScan.value++;
                        deviceFinderSlow(aliveHost, socketid, completeScan);
                    }else{
                        if(aliveHost.vendor == "Cisco Systems"){
                            aliveHost.vendor = "Cisco"
                        }
                    }
                    io.to(socketid).emit('net-scan', aliveHost);
                    hosts.push(aliveHost);
                } else {
                    deadHosts.push(host);
                }
                if(count == ips.length){
                    resolve(deadHosts);
                }
            });
        });
    });
}
```

Σχήμα 7.2: Κώδικας εύρεσης συσκευών μέσω της εκτέλεσης των python scripts

Η παραπάνω διαδικασία πραγματοποιείται τρεις φορές προκειμένου να ελαχιστοποιηθεί η πιθανότητα να μην βρεθεί μια ενεργή συσκευή. Μόλις ολοκληρωθεί η διαδικασία τότε αποστέλλεται η απάντηση στο αίτημα έτσι ώστε να ενημερωθεί η front-end εφαρμογή ότι η διαδικασία της εύρεσης των συσκευών στο δίκτυο που δόθηκε, ολοκληρώθηκε.

Ακολουθώντας την ίδια αρχιτεκτονική, υλοποιήθηκε ένα rest api endpoint σε συνδυασμό με τα web sockets για την αποθήκευση των συσκευών στο σύστημα. Χρησιμοποιήθηκε η διαδρομή /api/nodesSave. Το request που αναμένεται να φτάσει περιέχει μια σειρά από παραμέτρους όπου η κάθε μια παράμετρος περιέχει την IP διεύθυνση και τα διαπιστευτήρια της κάθε συσκευής. Αφού φτάσει ένα request, στέλνονται εντολές στις συσκευές προκειμένου να αρχικοποιηθούν (κεφάλαιο 6.4) ώστε να είναι έτοιμες για χρήση από την πλατφόρμα, και εντολές για να πάρει το σύστημα τις πληροφορίες που απαιτούνται από τις συσκευές (κεφάλαιο 6.5). Ύστερα από αυτές τις διαδικασίες προετοιμάζονται τα πεδία της κίνησης και του δικτύου κάθε διεπαφής της κάθε συσκευής. Στο τελευταίο βήμα βρίσκεται η αποθήκευση όλων αυτών των πληροφοριών στην βάση δεδομένων και μετά από αυτό ενημερώνεται ο χρήστης μέσω των sockets.

```
const getNodeInfo = (host) => {
    console.log("Go Python: " + host.ip);
    return new Promise(resolve => {
        let shell = new PythonShell('server/python/node_info.py', {mode: 'json', args: [host.ip, host.username, host.password]});
        shell.on('message', function (message) {
            resolve(message);
        });
        shell.on('error', function (message) {
            resolve({error: true, message})
        })
    })
}
```

Σχήμα 7.3: Κώδικας εκτέλεσης του python script για την ανάκτηση των πληροφοριών της

```
let node = {};
node = await initNode(host)
node = await getNodeInfo(host);
node.interfaces = await initTraffic(node.interfaces)
node.interfaces = await setInterfacesNetworks(node.interfaces)
user = await getUser(user)
if(node != null){
    const nodeDb = new Node({
        user: user._id,
        username: node.username,
        password: node.password,
        name: node.name,
        vendor: node.vendor,
        type: node.type,
        status: node.status,
        model: node.model,
        upTime: node.upTime,
        runConf: node.runConf,
        startConf: node.startConf,
        route_table: node.route_table,
        arp_table: node.arp_table,
        acl: node.acl,
        cdp: node.cdp,
        stp: node.stp,
        mac: node.mac,
        serial: node.serial,
        interfaces: node.interfaces,
        os: { name: node.os.name, version: node.os.version},
        eigrp: node.eigrp,
        ospf: node.ospf
    });

    try{
        const savedNode = await nodeDb.save();
        let sendNode = {ip: host.ip, messageState: 1}
        emitSavedNode(socket, [user], sendNode)

    }catch (err){
        let sendNode = {ip: host.ip, messageState: 2}
        emitSavedNode(socket, [user], sendNode)
        // res.json({ message: err});
    }
```

Σχήμα 7.4: Κώδικας αποθήκευσης της συσκευής στην βάση δεδομένων MongoDB

7.4 SNMP Manager

Σε αυτή την ενότητα θα αναλυθεί η επίτευξη του SNMP Manager στην JavaScript. Χρησιμοποιώντας την βιβλιοθήκη net-snmp κατέστη εφικτή η υλοποίηση ενός SNMP Manager. Πιο συγκεκριμένα χρησιμοποιήθηκε ένας receiver ώστε να λαμβάνει τα μηνύματα snmp που στέλνουν οι snmp agents. Αυτός ο receiver ακούει στην TCP πόρτα 162. Όταν φτάνει ένα μήνυμα snmp, ελέγχεται το oid. Οι μόνες πληροφορίες που ενδιαφέρουν την πλατφόρμα αυτήν την στιγμή είναι αν το snmp μήνυμα που έφτασε, αφορά την αλλαγή κατάστασης μιας διεπαφής της συσκευής αποστολέα. Αυτός ο έλεγχος καθώς και η ανάκτηση πληροφοριών από το μήνυμα που παρελήφθη γίνεται με τη αναγνώριση ενός συγκεκριμένου oid. Το oid που χρησιμοποιήθηκε για τα interfaces των συσκευών είναι το 1.3.6.1.2.1.2.2.1. Μόλις ανακτηθούν οι κατάλληλες πληροφορίες από το oid τότε γίνεται ενημέρωση της συσκευής στην βάση δεδομένων καθώς και αποστέλλεται στην front-end εφαρμογή.

```
module.exports = function(io) {
    const snmp = require ("net-snmp");


    let callback = async function (error, notification) {
        if ( error ) {
            console.error (error);
        } else {
            let pdu = notification.pdu;
            let info = notification.rinfo;
            let data = await modifyDataSNMP(pdu.varbinds, modifyTypeSNMP.receive, info.address);
            if(Object.keys(data).length > 1){
                updateInterfaceStatus(data)
            }
        }

    }

    let options = {
        port: 162,
        disableAuthorization: true,
        accessControlModelType: snmp.AccessControlModelType.None,
        address: null,
        transport: "udp4"
    }

   let receiver = snmp.createReceiver(options,callback);

   return receiver;
}
```

Σχήμα 7.5: Κώδικας υλοποίησης του SNMP Manager

Προς το παρόν δεν χρειάστηκε να χρησιμοποιηθούν περισσότερα oids ώστε να είναι σε θέση να ανακτά περισσότερες πληροφορίες από τα snmp μηνύματα που λαμβάνει. Μελλοντικά όμως θα μπορέσουν πολύ εύκολα να προστεθούν οποιαδήποτε oids χρειαστούν καθώς όλος ο μηχανισμός είναι υλοποιημένος.

7.5 SysLog Server

Ο SysLog Server υλοποιήθηκε για την παραλαβή της κίνησης από τις συσκευές. Χρησιμοποιώντας τη βιβλιοθήκη syslog-server επιτεύχθηκε η υλοποίηση του SysLog Server στην JavaScript. Πιο συγκεκριμένα με αυτή την βιβλιοθήκη υλοποιήθηκε ένας ακροατής συμβάντων syslog. Για κάθε μήνυμα που λαμβάνει ελέγχει αν περιέχεται μέσα σε αυτό η λέξη κλειδί AutonetTraffic. Όταν αναγνωριστεί αυτή η λέξη τότε καταλαβαίνουμε ότι το μήνυμα που ήρθε πρόκειται για την κίνηση ενός χρονικού στιγμιότυπου μιας διεπαφής της συσκευής αποστολέα. Προκειμένου να αποθηκευτούν τα δεδομένα της κίνησης με την επιθυμητή μορφή αναπτύχθηκε μια συνάρτηση τροποποίησης των δεδομένων η modifyMessage. Μόλις ολοκληρωθεί αυτή η διαδικασία, εκτελείται η μέθοδος pushTraffic ώστε να προστεθούν τα δεδομένα της κίνησης του εκάστοτε χρονικού στιγμιότυπου στην λίστα της κίνησης. Όταν φτάσει το μήνυμα που περιέχει την λέξη κλειδί AutonetTraffic και την λέξη κλειδί Finish τότε πρόκειται για την ολοκλήρωση της αποστολής της κίνησης όλων των διεπαφών της συσκευής αποστολέα και τα δεδομένα αποθηκεύονται στην βάση δεδομένων.

```
module.exports = function(io) {
    const serverSys = new SyslogServer();
    serverSys.on("message", async (value) => {

        if(value.message.includes("AutonetTraffic")){
            if(value.message.includes("Finish")){
                saveTraffic(value.host, io)
            }else{
                value = await modifyMessage(value, typeMessage.traffic)
                if(traffic[value.host]){
                    pushTraffic(value)
                }else{
                    traffic[value.host] = []
                    pushTraffic(value)
                }
            }
        }
    });

    serverSys.start();
    return serverSys
}
```

Σχήμα 7.6: Κώδικας υλοποίησης του SysLog Server

7.6 Sockets IO – Server

Η επικοινωνία της back-end εφαρμογής με την front-end εφαρμογή βασίζεται κατά κύριο λόγο στην υλοποίηση των web sockets που παρέχει η βιβλιοθήκη Sockets IO (κεφάλαιο 3.6.6). Η τεχνική που ακολουθήθηκε είναι να δημιουργούνται συγκεκριμένοι ακροατές ώστε να μπορεί να κατηγοριοποιηθεί το μήνυμα που λαμβάνει από τον Socket IO client. Ένα παράδειγμα είναι ο ακροατής που ακούει στα συμβάντα με αναγνωριστικό “topology”. Πρόκειται για τον ακροατή που ακούει όλα τα συμβάντα που αφορούν μια τοπολογία. Ένα συμβάν μπορεί να αφορά διάφορες ενέργειες για την εκάστοτε τοπολογία. Οπότε για την αποφυγή της δημιουργίας περαιτέρω ακροατών για την εξυπηρέτηση όλων των διάφορων ενεργειών, χρησιμοποιήθηκε ένα πεδίο που ονομάζεται method. Αυτό το πεδίο προσδιορίζει την ενέργεια που ζητείται να εξυπηρετηθεί. Έτσι λοιπόν όταν φτάνει ένα μήνυμα στον Socket IO server με το αναγνωριστικό “topology”, ελέγχεται το πεδίο method του μηνύματος και πραγματοποιείται με βάση αυτό η αντίστοιχη ενέργεια. Μόλις πραγματοποιηθεί η ενέργεια αυτή, ενημερώνει τον Socket IO client με ένα socket μήνυμα.

```
socket.on('topology', async (data) => {
    console.log(`Changes TOPOLOGY from ${data.user}` );
    let msg = {
        topology: data.topology,
        type: 'topology',
        method: data.method
    }

    switch(data.method){
        case 'new':
            msg.res = await newTopology(data.user, data.name, data.nodes)
            break
        case 'delete':
            msg.res = await deleteTopology(data.id)
            break
        case 'update':
            msg.res = await updateTopology(data.topology)
            break
    }
    io.emit(data.user, msg)
})
```

Σχήμα 7.7: Κώδικας υλοποίησης του Socket IO ακροατή του server

Ο Socket IO server εκτός από το είναι υπεύθυνος να εξυπηρετεί αιτήματα των χρηστών, είναι επίσης υπεύθυνος να ενημερώνει όλους τους χρήστες που είναι ενεργοί εκείνη την χρονική στιγμή σε κάθε

αλλαγή που προέκυψε στα δεδομένα της πλατφόρμας. Απευθύνεται στους χρήστες που έχουν την εξουσιοδότηση πρόσβασης στα εκάστοτε δεδομένα. Η αποστολή των δεδομένων μέσω των web sockets στον client απαιτεί στον Socket IO client να έχουν υλοποιηθεί οι κατάλληλοι ακροατές. Αυτή η διαδικασία θα αναλυθεί στο κεφάλαιο 8. Συγκεκριμένα με την χρήση της μεθόδου emit που παρέχει η Socket IO γίνεται η εκπομπή – αποστολή των δεδομένων στους χρήστες. Για λόγους επεκτασιμότητας και ευελιξίας δημιουργήθηκαν διάφορες μέθοδοι που στόχο έχουν την εκπομπή των δεδομένων όπου η κάθε μια έχει τον δικό της ρόλο. Με αυτό τον τρόπο μπορούν πολύ εύκολα να προστεθούν και άλλες τέτοιες μέθοδοι στο μέλλον, χωρίς να επηρεάζει τα υπόλοιπα. Ένα παράδειγμα είναι η μέθοδος emitTraffic, όπου όταν καλείται με τα κατάλληλα ορίσματα αποστέλλεται η κίνηση μιας συσκευής στους χρήστες που έχουν την εξουσιοδότηση για πρόσβαση στα δεδομένα της συσκευής.

```
const emitTraffic = (users, data) => {
    let msg = {
        data: data,
        type: 'traffic'
    }
    users.forEach(element => {
        mainIo.emit(element.username, msg)
    });
}
```

Σχήμα 7.8: Κώδικας υλοποίησης αποστολής της κίνησης, στους χρήστες της πλατφόρμας, μέσω των sockets

7.7 Αλληλεπίδραση με την MongoDB

Το επόμενο βήμα είναι να αναλυθεί ο τρόπος με τον οποίο ο Node.Js server επικοινωνεί με την βάση δεδομένων MongoDB. Χρησιμοποιώντας την βιβλιοθήκη mongoose, και συγκεκριμένα την μέθοδο connect ο server συνδέεται στην βάση δεδομένων. Τα στοιχεία που απαιτούνται για την πραγματοποίηση της σύνδεσης βρίσκονται στο .env αρχείο με την ονομασία DB_CONNECTION. Η αλληλεπίδραση με την MongoDB χωρίστηκε σε δυο φάσεις. Η πρώτη ήταν η υλοποίηση των μοντέλων της βάσης δεδομένων και η δεύτερη φάση είναι τα αιτήματα που γίνονται στην Mongo.

Στην διαδικασία της υλοποίησης των μοντέλων, δημιουργήθηκαν τα μοντέλα δεδομένων, των συλλογών της MongoDB. Πιο συγκεκριμένα χρησιμοποιώντας τον constructor της mongoose, που ονομάζεται Schema, δημιουργούμε τον σκελετό των δεδομένων που θα χρησιμοποιείται σε όλη την back-end εφαρμογή. Η mongoose παρέχει την δυνατότητα να ορίσουμε στα δεδομένα κάποιους κανόνες όπως για παράδειγμα αν κάποιο πεδίο είναι υποχρεωτικό. Για παράδειγμα στο μοντέλο που δημιουργήθηκε για την αποθήκευση των δεδομένων ενός δικτύου (Σχήμα 7.9), διακρίνεται ότι όλα τα πεδία είναι υποχρεωτικά.

```
const networkSchema = new mongoose.Schema({
    user:{
        type: Schema.Types.ObjectId,
        required: true,
        ref: 'users'
    },
    ipNetwork:{
        type: Object,
        required: true
    },
    ipAddress:{
        type: Object,
        required: true
    },
    firstAddress:{
        type: Object,
        required: true
    },
    lastAddress:{
        type: Object,
        required: true
    },
    broadcastAddress:{
        type: Object,
        required: true
    },
    subnetMask:{
        type: Object,
        required: true
    },
    subnetMaskLength:{
        type: Object,
        required: true
    },
    numHosts:{
        type: Object,
        required: true
    },
})
```

Σχήμα 7.9: Κώδικας υλοποίησης του μοντέλου των δικτύων στην MongoDB

Στην δεύτερη φάση, υλοποιήθηκε όλος ο κώδικας που αφορά ενέργειες όπως αποθήκευση, επεξεργασία και ανάκτηση των δεδομένων από την MongoDB. Για την ολοκλήρωση αυτών των ενεργειών χρησιμοποιήθηκαν οι διάφοροι μέθοδοι που παρέχει η mongoose στα μοντέλα που δημιουργήθηκαν στην πρώτη φάση. Για παράδειγμα για την ανάκτηση των δεδομένων των συσκευών που έχει εξουσιοδότηση για πρόσβαση στα δεδομένα, ένας χρήστης, χρησιμοποιήθηκε αρχικά το μοντέλο του User, που δημιουργήθηκε στην πρώτη φάση. Πιο συγκεκριμένα χρησιμοποιήθηκε η μέθοδος findOne του μοντέλου, με παράμετρο το όνομα χρήστη προκειμένου να ανακτηθούν οι πληροφορίες του χρήστη καθώς και το αναγνωριστικό του (id). Αμέσως μετά καλείται από το μοντέλο Node, που είναι το μοντέλο για τις συσκευές, η μέθοδος find ώστε να βρεθούν όλες οι συσκευές του εκάστοτε χρήστη (Σχήμα 7.10).

```
const getUserNodes = async (username) => {
    let user = await User.findOne({username: username})
    let nodes = await Node.find({user: user._id})
    nodes = await modifyNodesData(nodes)
    return nodes;
}
```

Σχήμα 7.10: Κώδικας ανάκτησης όλων των συσκευών ενός χρήστη από την βάση δεδομένων

7.8 Ενημέρωση Δεδομένων Συσκευών

Μέχρι στιγμής έχουν αναλυθεί οι διάφορες τεχνικές που έχουν υλοποιηθεί προκειμένου ο Node.js server να ακούει τα συμβάντα. Τα συμβάντα όμως δεν είναι αρκετά για να κρατάνε ενήμερη την εφαρμογή. Για παράδειγμα δεν μπορεί ο server να γνωρίζει αν άλλαξε το όνομα της συσκευής. Για αυτό τον λόγο δημιουργήθηκε η ανάγκη της ανάκτησης πληροφοριών από τις συσκευές κάθε 1 λεπτό.

Αυτό το τέχνασμα υλοποιείται με την εκτέλεση του python script, που αναφέρθηκε στο κεφάλαιο 6.5. Πιο συγκεκριμένα αρχικά ανακτώνται όλες οι απαραίτητες πληροφορίες από όλες τις συσκευές που υπάρχουν στην βάση δεδομένων. Αμέσως μετά για κάθε συσκευή που βρέθηκε στην βάση, διακρίνεται η IP διεύθυνση της συσκευής και εκτελείται η συνάρτηση saveUpdatedNodesData όπου εκτελούνται μια σειρά από διαδικασίες προκειμένου να ενημερωθούν τα δεδομένα που υπάρχουν στην MongoDB. Τέλος για να μπορέσει να πραγματοποιείται η παραπάνω διαδικασία κάθε 1 λεπτό ορίζεται το κάλεσμα της μεθόδου updateNodesData με την μέθοδο που παρέχει η JavaScript, η οποία ονομάζεται setInterval.

```
const updateNodesData = async () => {
    let nodes = await getAllNodes()
    nodes = nodes.map(node => {
        return { ...node, ip: node.ip.includes('/') ? node.ip.slice(0, -(node.ip.length - node.ip.indexOf('/'))) : node.ip }
    })
    nodes.forEach(node => saveUpdatedNodesData(node))
}

const saveUpdatedNodesData = async (node) => {
    let updatedNode = await getNodeInfo(node)
    if(!updatedNode.error){
        const nodeDb = await getNodeInfoByID(node._id)
        if(nodeDb.interfaces.length < updatedNode.interfaces.length){
            updatedNode = await initNode(node)
            updatedNode = await getNodeInfo(node)
        }
        updatedNode._id = node._id
        updateNodeData(updatedNode)
    }else{
        setNodeStatus(node, false)
    }
}
```

Σχήμα 7.11: Κώδικας υλοποίησης ανάκτησης δεδομένων από όλες τις συσκευές μέσω του python script

7.9 Τερματικό – SSH

Το τελευταίο κομμάτι της back-end εφαρμογής που θα αναλυθεί είναι η δυνατότητα σύνδεσης του χρήστη στο τερματικό της συσκευής. Προκειμένου να είναι ένα οικείο περιβάλλον στον χρήστη χρειαζόταν η άμεση αποστολή εντολών καθώς και η άμεση αποστολή χαρακτήρων στο τερματικό της συσκευής. Για παράδειγμα όταν ο χρήστης συνδεθεί σε ένα τερματικό και πατήσει έναν χαρακτήρα από το πληκτρολόγιό του, πρέπει να μπορεί να δει κατευθείαν την ανταπόκριση της συσκευής. Για να επιτευχθεί αυτό χρησιμοποιήθηκε η Socket IO.

Ένα ακόμα ζήτημα ήταν να δημιουργηθεί μια σύνδεση SSH με την συσκευή και να διατηρηθεί η σύνδεση ανοιχτή μέχρι ο χρήστης να την τερματίσει. Για τον σκοπό αυτό χρησιμοποιήθηκε ο constructor, που ονομάζεται Client, της βιβλιοθήκης ssh2 (κεφάλαιο 3.6.17).

Η λογική που ακολουθήθηκε για την υλοποίηση της σύνδεσης πραγματικού χρόνου είναι ο συνδυασμός των δύο παραπάνω τεχνικών όπου ο Node.js server παίρνει τον ρόλο του ενδιάμεσου κόμβου μεταξύ της front-end εφαρμογής και των συσκευών. Πιο συγκεκριμένα τα sockets χρησιμοποιούνται για την επικοινωνία με τον χρήστη και το ssh2 για την επικοινωνία με τις συσκευές. Έτσι όταν ο χρήστης εισάγει έναν χαρακτήρα στέλνεται με τα sockets στον Node.js server. Μόλις ληφθεί ο χαρακτήρας αυτός στέλνετε κατευθείαν στην συσκευή μέσω της ανοιχτής σύνδεσης SSH. Ακολούθως η συσκευή στέλνει πίσω στον Client της ssh2 την απάντηση σε αυτό που έστειλε ο χρήστης. Τέλος ο server στέλνει την απάντηση της συσκευής πίσω στον χρήστη με την χρήση των sockets. Έτσι δημιουργήθηκε η δυνατότητα σύνδεσης του χρήστη στο τερματικό των συσκευών σε πραγματικό χρόνο.

```
sshCon.on('ready', function(){
    io.to(req.socket).emit('consoleData', '\r\n*** SSH CONNECTION ESTABLISHED ***\r\n');
    sshCon.shell(function(err, stream){
        if (err){
            res.status(400).send(err.message);
            return io.to(req.socket).emit('consoleData', '\r\n SHH SHELL ERROR: ' + err.message + '***\r\n');
        }else{
            res.status(200).send('SSH CONNECTION ESTABLISHED');
        }
        streams[req.socket] = stream;
        stream.on('data', function(d){
            io.to(req.socket).emit('consoleData', d.toString('binary'))
        }).on('close', function(){
            sshCon.end();
        });
    });
}).on('close', function(){
    io.to(req.socket).emit('consoleData', '\r\n*** SSH CONNECTION CLOSED *** \r\n');
}).on('error', function(err){
    io.to(req.socket).emit('consoleData', '\r\n*** SSH CONNECTION ERROR: ' + err.message + ' *** \r\n');
```

Σχήμα 7.12: Κώδικας υλοποίησης του τερματικού στην μεριά του server

7.10 Επίλογος

Στο κεφάλαιο αυτό αναλύθηκαν όλες οι τεχνικές που χρησιμοποιήθηκαν προκειμένου να υλοποιηθεί μια back-end εφαρμογή, όπου θα μπορεί να επικοινωνεί με την front-end εφαρμογή, να ελέγχει και να επικοινωνεί με δικτυακές συσκευές Cisco καθώς και να χρησιμοποιεί την βάση δεδομένων προκειμένου να επιτευχθούν οι επιθυμητές ενέργειες. Η υλοποίηση του ήταν αρκετά περίπλοκη διότι υπήρχε η ανάγκη της χρήσης και τον συνδυασμό πολλών διαφορετικών τεχνολογιών. Αναλύοντας και την back-end εφαρμογή, το επόμενο βήμα είναι να αναλυθεί η front-end εφαρμογή.

Κεφάλαιο 8ο: Εφαρμογή Front-end

8.1 Εισαγωγή

Στο παρόν κεφάλαιο θα αναλυθεί όλος ο κώδικας που υλοποιήθηκε προκειμένου να αναπτυχθεί η web εφαρμογή που θα έρχεται σε επαφή με τον χρήστη. Η Frontend εφαρμογή έχει χτιστεί πάνω στην Vue.js 3 και χρησιμοποιήθηκε το Quasar για την εύκολη δημιουργία ευέλικτων οθονών.

8.2 Vue Store

Το store που παρέχει η Vuex (κεφάλαιο 3.3.2) χρησιμοποιήθηκε για να αποθηκεύονται τα δεδομένα του χρήστη. Προκειμένου να μπορούν να χρησιμοποιηθούν τα δεδομένα του χρήστη από οποιοδήποτε σημείο της εφαρμογής προέκυψε η ανάγκη της δημιουργίας ενός store όπου θα αποθηκεύονται μεταβλητές και θα είναι διαθέσιμες από παντού. Μοιάζει με την έννοια των global μεταβλητών με την διαφορά ότι η Vuex παρέχει περισσότερες δυνατότητες. Πιο συγκεκριμένα δημιουργήθηκαν μέθοδοι για αλλαγή των τιμών των μεταβλητών καθώς και μέθοδοι που επιστρέφουν αυτές τις τιμές.

Για παράδειγμα στα δεδομένα του χρήστη και συγκεκριμένα στις συσκευές, δημιουργήθηκαν μέθοδοι (mutations) όπως setNodes, addNode, deleteNode, updateNodeInterfaceStatus καθώς και άλλες πολλές. Για να χρησιμοποιηθούν αυτά τα mutations από οποιοδήποτε σημείο της εφαρμογής υλοποιήθηκαν τα actions τα όποια έχουν τις ίδιες ονομασίες με τα mutations. Για την ανάκτηση των δεδομένων, δημιουργήθηκαν οι getters. Η getter μέθοδος getNodes επιστρέφει όλες τις συσκευές του χρήστη, ενώ η getter μέθοδος getNetworkByIp δέχεται μια παράμετρος η οποία η καθορίζει την διεύθυνση IP δικτύου, για το δίκτυο που πρόκειται να ανακτηθεί. Τέλος οι getters μέθοδοι μπορούν να χρησιμοποιηθούν με τις δυνατότητες που προσφέρει η Vue.js όπως η watch, ώστε να παρακολουθούν τις τιμές αυτές και να γίνονται κάποιες ενέργειες όταν αλλάζουν οι τιμές τους.

```
updateNodeInterfaceStatus(state, val){
    const node = state.nodes.data.find(node ⇒ node._id === val.id)
    node.interfaces.find(inter ⇒ inter.interface.value === val.if).link_status.value = val.adminStatus
    node.interfaces.find(inter ⇒ inter.interface.value === val.if).protocol_status.value = val.operStatus
    state.nodes.changedFromUser = val.changedFromUser
},
```

Σχήμα 8.1: Κώδικας υλοποίησης του mutation για την ενημέρωση της κατάστασης μιας διεπαφής, μιας συσκευής

```
updateNodeInterfaceStatus({commit}, val){
    commit('updateNodeInterfaceStatus', val)
},
```

Σχήμα 8.2: Κώδικας υλοποίησης του action για την εκτέλεση του mutation του Σχήματος 8.1

```
getNetworkByIp: (state) ⇒ (ip) ⇒ {
    return state.networks.data.find(network ⇒ network.ipNetwork.value === ip)
},
```

Σχήμα 8.3: Κώδικας υλοποίησης της getter μεθόδου για ανάκτηση ενός δικτύου με κριτήριο την IP διεύθυνση

Στην εφαρμογή επειδή υπάρχει η περίπτωση οι αλλαγές των δεδομένων να προέρχονται από τον server μέσω των sockets ορίστηκε μια σημαία η οποία ονομάζεται changeFromUser. Αυτή η σημαία καθορίζει αν οι αλλαγές προήλθαν από ενέργεια του χρήστη (true) ή αν "ακούστηκε" από τον server (false). Όταν υπάρχει αυτή η σημαία, τα δεδομένα του χρήστη βρίσκονται στο πεδίο data της μεταβλητής των συγκεκριμένων δεδομένων στο store. Ένα παράδειγμα είναι η μεταβλητή nodes η οποία βρίσκεται μέσα στο store. Τα δεδομένα που αφορούν τις συσκευές αποθηκεύονται στο πεδίο data της μεταβλητής node ενώ η ένδειξη αν οι αλλαγές προήλθαν από τον χρήστη ή όχι, αποθηκεύεται στο πεδίο changeFromUser.

```
server: {},
userInfoCredentials: {},
userInfoDetails: {},
nodes: {data: [], changedFromUser: false},
networks: {data: [], changedFromUser: false},
topologies: {data: [], changedFromUser: false},
```

Σχήμα 8.4: Κώδικας υλοποίησης του store των δεδομένων του χρήστη

8.3 Sockets – Client

Καθώς έχει αναλυθεί η λειτουργία των sockets σαν server στην παρούσα ενότητα θα αναλυθεί πως υλοποιήθηκε ο socket client και πως συνεργάζεται με τον socket server. Αφού επιτευχθεί η σύνδεση του χρήστη στην πλατφόρμα, δημιουργείται μια σύνδεση με τον server με την χρήση των sockets (Socket IO). Αρχικά, μόλις η εφαρμογή δημιουργήσει την σύνδεση, ζητώνται όλα τα δεδομένα του χρήστη από τον server.

```
// Init data after sockets connection
socket.on('connect', () => {
    socket.emit('initUser', store.getters['User/getUsername']);
    store.dispatch('User/setSocket', socket.id);
    store.dispatch('Socket/setSocketReady', true);
});
```

Σχήμα 8.5: Κώδικας υλοποίησης του socket ακροατή όταν επιτυγχάνεται η σύνδεση του χρήστη

Η τεχνική με την οποία θα ακούει τα μηνύματα από τον server απαιτεί έναν ακροατή, ο οποίος ακούει μηνύματα που έχουν την ταμπέλα του ονόματος χρήστη (username), που συνδέθηκε στην εφαρμογή. Όταν ο ακροατής ακούει ένα μήνυμα, τότε ελέγχεται ο τύπος του μηνύματος ώστε να καθοριστούν οι ενέργειες που πρέπει να πραγματοποιηθούν. Για παράδειγμα όταν πρόκειται για την απάντηση του server, όταν ο socket client ζήτησε όλα τα δεδομένα του χρήστη προκειμένου να τα αρχικοποιήσει, ο τύπος του μηνύματος ονομάζεται userData. Όταν φτάσει ένα τέτοιο μήνυμα, τότε αποθηκεύονται όλες οι πληροφορίες στο store.

```
// Listener to receive user data (nodes, topologies ... ) changes
socket.on(store.getters['User/getUsername'], (msg) => {
    switch(msg.type){
        case 'userData':
            store.dispatch('UserData/setNodes', { data: msg.data.nodes, changedFromUser: false });
            store.dispatch('UserData/setTopologies', msg.data.topologies);
            store.dispatch('UserData/setNetworks', msg.data.networks);
            msg.data.server.users = msg.data.users
            store.dispatch('UserData/setServer', msg.data.server)
            store.dispatch('UserData/setUserInfoCredentials', {username: msg.data.userInfo.username})
            store.dispatch('UserData/setUserInfoDetails', {
                firstname: msg.data.userInfo.name,
                surname: msg.data.userInfo.surname,
                email: msg.data.userInfo.email
            })
            break;
```

Σχήμα 8.6: Κώδικας αποθήκευσης των δεδομένων του χρήστη στο store του vuex

Το δεύτερο σκέλος της χρήσης των sockets είναι η μετάδοση των αλλαγών που προκαλεί ο χρήστης στα δεδομένα, προκειμένου να αποθηκευτούν στην βάση δεδομένων. Αυτό πραγματοποιείται με τον συνδυασμό των watchers, που παρέχει η Vue, στα δεδομένα του store. Έτσι όταν γίνεται κάποια αλλαγή τότε μπορεί να μεταδοθεί στον server μέσω των sockets. Για παράδειγμα έχει υλοποιηθεί ένας watcher για να παρακολουθεί τα δεδομένα των στοιχείων σύνδεσης του χρήστη στην πλατφόρμα. Αν παρατηρηθεί κάποια αλλαγή τότε στέλνονται τα αλλαγμένα πλέον διαπιστευτήρια στον server μέσω των sockets, με την ταμπέλα save-user_credentials.

```
watch(() => store.getters['UserData/getUserInfoCredentials'], (data) => {
    if(data != null){
        socket.emit('save-user_credentials', {
            user: store.getters['User/getUsername'],
            username: data.username, password: data.password
        });
    }
})
```

Σχήμα 8.7: Κώδικας υλοποίησης watcher για τα διαπιστευτήρια του χρήστη

8.4 Κονσόλα Συσκευής

Στην παρούσα ενότητα θα αναλυθεί τη τεχνική που χρησιμοποιήθηκε για να επιτευχθεί η δυνατότητα της σύνδεσης στο τερματικό της συσκευής μέσω του web application. Προκειμένου να δημιουργηθεί ένα οικείο περιβάλλον για τον χρήστη χρησιμοποιήθηκε το Xterm (κεφάλαιο 3.6.3). Το Xterm παρέχει έναν constructor που καλείται Terminal. Με αυτό τον τρόπο δημιουργείται ένα τερματικό για τον χρήστη χωρίς να μπορεί να κάνει το οτιδήποτε για την ώρα, αφού δεν έχει υλοποιηθεί κάποια σύνδεση προς το παρόν. Αμέσως μετά στέλνεται ένα rest request με την χρήση του Axios (κεφάλαιο 3.6.7), στον server, ώστε να ενημερώσει ότι πρέπει να υλοποιηθεί μια σύνδεση SSH στην εκάστοτε συσκευή.

```
const initConsoleSSH = data => {
    return new Promise(resolve => {
        const req = {
            method: 'get',
            url: url.initConsole,
            params: {
                ip: data.ip,
                socket: data.socket,
                username: data.username,
                password: data.password,
                port: data.port
            }
        }
        axios(req)
            .then(response => { resolve(response); })
            .catch(error => { resolve(error.response) });
    })
}
```

Σχήμα 8.8: Κώδικας αρχικοποίησης της σύνδεσης με SSH στη συσκευή, για χρήση του τερματικού της συσκευής

Στην συνέχεια όταν ο χρήστης πατάει ένα πλήκτρο, ανιχνεύεται από το terminal, που δημιουργήθηκε μέσω του constructor Terminal, με την χρήση της μεθόδου, που παρέχει το Xterm, onKey. Όταν ανιχνευτεί το πλήκτρο, αποθηκεύεται ο χαρακτήρας του πλήκτρου στο store, και αμέσως μετά ανιχνεύεται από τον watcher που έχει υλοποιηθεί με τον τρόπο που αναλύθηκε στο κεφάλαιο 8.3. Ο Socket client στέλνει τον χαρακτήρα που εισήγαγε ο χρήστης στον Socket server. Η διαδικασία που πραγματοποιείται στον server αναλύθηκε στο κεφάλαιο 7.9. Καθώς φτάσει η απάντηση από τον server ο socket ακροατής αποθηκεύει την απάντηση στο store. Τέλος με την χρήση ενός watcher ανιχνεύεται η απάντηση της συσκευής που δόθηκε από τον server, και με την χρήση της μεθόδου write, που παρέχει το αντικείμενο terminal, που δημιουργήθηκε προηγουμένως, εμφανίζεται στην οθόνη του χρήστη μέσα στο τερματικό.

```
terminal.onKey(function (ev) {
    store.dispatch('Socket/setConsoleDataEmit', ev);
});
```

Σχήμα 8.9: Κώδικας αποθήκευσης του χαρακτήρα που πατήθηκε στο store

```
watch(() => store.getters['Socket/getConsoleDataListen'], (data) => {
    if(data != null){
        terminal.write(data);
        store.dispatch('Socket/setConsoleDataListen', null);
    }

})
```

Σχήμα 8.10: Κώδικας του watcher για τους χαρακτήρες που πατήθηκαν στο τερματικό και αποστολή αυτών στην back-end εφαρμογή μέσω των sockets

8.5 Γραφήματα Κίνησης

Προκειμένου να εμφανίζεται με γραφήματα η κίνηση στον χρήστη ώστε να μπορεί πιο εύκολα να κατανοηθεί, χρησιμοποιήθηκε η βιβλιοθήκη Chart.js. Με τον constructor που παρέχει η Chart.js, δημιουργείται το αντικείμενο του γραφήματος. Μόλις έρθουν τα δεδομένα του χρήστη, χρησιμοποιώντας την μέθοδο update, που παρέχει το αντικείμενο που δημιουργήθηκε προηγουμένως,

γίνεται ενημέρωση του γραφήματος της κίνησης. Επειδή όμως χρειαζόταν, για κάθε αλλαγή που γινόταν στην κίνηση των συσκευών, να ενημερώνονται τα δεδομένα του γραφήματος χρησιμοποιήθηκε ο συνδυασμός των sockets και του store.

Για να γίνει κατανοητό αυτό θα αναλύσουμε το παράδειγμα του γραφήματος της κίνησης που εμφανίζεται σε κάθε συσκευή. Αρχικά δημιουργείται το αντικείμενο του γραφήματος με τα κατάλληλα ορίσματα. Την πρώτη φορά που θα οριστούν τα δεδομένα εκτελείται η μέθοδος initData. Σε αυτή την μέθοδο για κάθε διεπαφή της συσκευής δημιουργείται μια εγγραφή στο dataset του γραφήματος. Για κάθε διεπαφή προστίθεται στο data του dataset της διεπαφής, η υπολογισμένη κίνηση κάθε χρονικού στιγμιότυπου που περιέχει. Η τρόπος υπολογισμού της κίνησης έχει αναλυθεί στο κεφάλαιο 4.8.4. Μόλις τελειώσει η παραπάνω διαδικασία και κάποιες άλλες ενέργειες, εκτελείται η μέθοδος update ώστε να ενημερωθεί το γράφημα.

```
async function initData(){
    let ifs = props.node.interfaces

    for(let j=0; j < ifs.length; j++){
        await myChart.data.datasets.push({
            label: ifs[j].interface_short.value,
            backgroundColor: colors[colorIndex],
            borderColor: colors[colorIndex],
            data: [],
        })
        let trafficIf = ifs[j].traffic.value
        let trafficIfIndex = 0;
        trafficIf.length ≥ 5 ? trafficIfIndex = trafficIf.length-5 : trafficIfIndex=0
        trafficFlag = trafficIf.length;
        for(let i=trafficIfIndex; i < trafficIf.length; i++){
            if(i ≠ 0){
                if(j == 0){
                    await pushLabelTime(trafficIf[i].date);
                }
                let rate = 0;
                if(parseInt(trafficIf[i].bytes.out) < parseInt(trafficIf[i-1].bytes.out))
                    rate = parseInt(trafficIf[i].bytes.out) + parseInt(trafficIf[i].bytes.in);
                else
                    rate =
                        (parseInt(trafficIf[i].bytes.out) +
                        parseInt(trafficIf[i].bytes.in)) -
                        (parseInt(trafficIf[i-1].bytes.out) +
                        parseInt(trafficIf[i-1].bytes.in));
                await myChart.data.datasets[j].data.push(rate/60);
            }
        }

        if(colorIndex == colors.length-1)
            colorIndex = 0
        else
            colorIndex++
    }

    if(dateFlag ≠ null && myChart.data.labels.length > 0){
        for(let k=0; k < myChart.data.labels.length; k++){
            myChart.data.labels[k] = await myChart.data.labels[k].split(' ')[1];
        }
    }
    myChart.update();

}
```

Σχήμα 8.11: Κώδικας δημιουργίας του γραφήματος της κίνησης όλων των διεπαφών μιας συσκευής

Όταν ο socket client "ακούει" καινούργια δεδομένα για τις συσκευές, ενημερώνει τα δεδομένα των συσκευών στο store. Χρησιμοποιώντας τους watchers της Vue ανιχνεύεται η αλλαγή των δεδομένων και εκτελείται η μέθοδος addData στην περίπτωση που οι αλλαγές αυτές αφορούν την κίνηση. Ο ρόλος της μεθόδου αυτής είναι να προσθέσει στο γράφημα τα δεδομένα της κίνησης για το καινούργιο χρονικό στιγμιότυπο.

```
watch(() => _.cloneDeep(props.node), () => {
    fillChart()
})

function fillChart(){
    if(!initFlag){
        initData()
        initFlag = true
    }else{
        if(updatedTraffic())
            addData()
    }
}
```

Σχήμα 8.12: Κώδικας ενημέρωσης πρόσθετης κίνησης των διεπαφών μιας συσκευής

8.6 Πίνακες Δεδομένων

Στην παρούσα ενότητα θα παρουσιαστεί ο τρόπος με τον οποίο εμφανίζονται τα δεδομένα σε πίνακες. Για το σκοπό αυτό χρησιμοποιήθηκε το component “q-table” που παρέχει το Quasar. Προκειμένου να ενημερώνονται τα δεδομένα σε περίπτωση αλλαγών, δημιουργήθηκαν κάποια computed properties που παρέχει η Vue.js. Για παράδειγμα στον πίνακα των διεπαφών της συσκευής έχει υλοποιηθεί το rows computed property που αντιστοιχεί στις γραμμές που θα έχει ο πίνακας. Μέσα από αυτό το rows, ανακτώνται όλα τα δεδομένα του interface που έχει επιλεχθεί, εφόσον η τιμή του visible είναι true (Σχήμα 8.13).

```
rows(){
  let rowsArray = [];
  let node = this.node;
  try{
    for(let k=0; k < node.interfaces.length; k++){
      if(node.interfaces[k].interface_short.value == this.model){
        Object.keys(node.interfaces[k]).forEach(key => {
          if(node.interfaces[k][key].visible)
            rowsArray.push({name: node.interfaces[k][key].name, value: node.interfaces[k][key].value})
        });
        return rowsArray
      }
    }
  }catch(error){return rowsArray}
  return rowsArray
},
```

Σχήμα 8.13: Κώδικας του rows computed property για τον πίνακα των διεπαφών μιας συσκευής

Σε ορισμένες περιπτώσεις χρειάστηκε ο συνδυασμός του q-table component με άλλα components που παρέχει το Quasar. Στην παραπάνω περίπτωση για παράδειγμα χρησιμοποιήθηκε το component q-select όπου επιτρέπει στον χρήστη να επιλέγει το interface της συσκευής για το οποίο επιθυμεί να δει πληροφορίες. Τα δεδομένα του q-select ορίζονται μέσω ενός computed property.

Ένα ακόμα αρκετά χρήσιμο component που χρησιμοποιήθηκε σε συνδυασμό με το q-table είναι το q-input. Αυτό το component χρησιμοποιήθηκε για να παρέχει την δυνατότητα στον χρήστη να φιλτράρει τα αποτελέσματα. Πιο συγκεκριμένα όταν ο χρήστης πληκτρολογεί σε αυτό το component τότε ορίζονται αυτοί οι χαρακτήρες στο πεδίο filter του q-table. Με αυτό το πεδίο ορίζονται ποιες γραμμές θα εμφανίζονται από τις υπάρχουσες.

```
<q-table
  class="dashboard-table"
  :rows="rows"
  :columns="columns"
  row-key="ip"
  flat
  hide-pagination
  :filter="filter"
  :pagination="pagination"
  bordered
>
```

Σχήμα 8.14: Κώδικας υλοποίησης ενός πίνακα με την χρήση του q-table component

Τέλος αξίζει να σημειωθεί ότι τα computed properties χρησιμοποιούνται για να ενημερώσουν τις πληροφορίες που εμφανίζονται στην οθόνη. Για την παρακολούθηση όμως των αλλαγών των δεδομένων, που είναι αποθηκευμένα στο store, χρησιμοποιούνται οι watchers.

8.7 Αναπαράσταση Configurations Συσκευών

Τα configurations των συσκευών διακρίνονται σε running και start-up. Προκειμένου να δοθεί η δυνατότητα στον χρήστη να προβάλει τα configurations αυτά χρησιμοποιήθηκε η βιβλιοθήκη Prism.js (κεφάλαιο 3.6.4). Η υλοποίησή του ήταν αρκετά απλή καθώς το Prism αναλαμβάνει την εμφάνιση. Πιο συγκεκριμένα δημιουργήθηκε ένας watcher ώστε να ακούει τις αλλαγές του της συσκευής που έχει περαστεί ως prop της Vue από το component γονέα. Σε κάθε αλλαγή των δεδομένων ορίζεται το dataConfig που παρουσιάζεται στον χρήστη ανάλογα με το ποιο από τα δύο configuration θέλει να προβάλει.

```
watch(() => _.cloneDeep(props.node), () => {
    if(props.title.includes('Running'))
        dataConfig = props.node.runConf
    else
        dataConfig = props.node.startConf
})
```

Σχήμα 8.15: Κώδικας υλοποίησης μεταβλητής δεδομένων του prism

```
<q-scroll-area class="scrollArea">
    <prism language="editorconfig">{{ dataConfig }}</prism>
</q-scroll-area>
```

Σχήμα 8.16: Κώδικας υλοποίησης αναπαράστασης ενός configuration μέσω του prism

8.8 Τοπολογία

Η επίτευξη της λειτουργίας μιας τοπολογίας ήταν ίσως από τις πιο περίπλοκες υλοποιήσεις στην front-end εφαρμογή. Πέρα από τα sockets, το store και τους watcher της Vue, η υλοποίηση της τοπολογίας βασίζεται στον συνδυασμό των βιβλιοθηκών Plain Draggable και Leader Line. Η plain-draggable χρησιμοποιήθηκε για να είναι εφικτό ο χρήστης να “σύρει” τις συσκευές, με το ποντίκι του στην επιφάνεια της τοπολογίας. Η leader-line χρησιμοποιήθηκε για την δημιουργία των συνδέσεων μεταξύ των συσκευών που περιέχει η εκάστοτε τοπολογία. Τέλος δημιουργήθηκαν δύο constructors. Ο ένας είναι για την δημιουργία των συσκευών στην τοπολογία καθώς περιέχει και το draggable στοιχείο που

δημιουργείται από το plain-draggable. Ο άλλος constructor αφορά τις συνδέσεις μεταξύ των συσκευών και περιέχει πληροφορίες για το "καλώδιο" ακόμα και το στοιχείο που δημιουργήθηκε από το leader-line. Για λόγους συντομίας οι συνδέσεις μεταξύ των συσκευών θα αναφέρονται ως καλώδια.

Αρχικά υλοποιήθηκαν οι watchers της Vue όπου παρατηρούν τα δεδομένα, και δίνουν εντολές να δημιουργηθούν ή να επεξεργαστούν τα αντικείμενα που βρίσκονται στην τοπολογία. Πιο συγκεκριμένα για να δημιουργηθούν οι συσκευές στην τοπολογία καλείται η μέθοδος initNodes. Μέσα στην μέθοδο αυτή δημιουργούνται μέσω του constructor, που αναφέρθηκε προηγουμένως, οι συσκευές, και ορίζεται η θέση τους στην επιφάνεια της τοπολογίας.

```
initNodes() {
    let nodesData = this.nodesData.value
    for(let i=0; i < nodesData.length; i++){
        let topoNode = this.topo.value.nodes.find(node => node.id == nodesData[i]._id)
        this.nodes[nodesData[i]._id] = new NetNode(
            nodesData[i]._id,
            this.topo.value._id,
            this.imgRefs[i],{
                name: this.labelRefs[i],
                x: topoNode.label.x,
                y:topoNode.label.y
            },
            [],
            nodesData[i].interfaces,
            this.$router.push
        );
        this.nodes[nodesData[i]._id].dragnode.left = topoNode.x;
        this.nodes[nodesData[i]._id].dragnode.top = topoNode.y;
        this.nodes[nodesData[i]._id].labelPosition();
    }

    this.findLinks()
},
```

Σχήμα 8.17: Κώδικας δημιουργίας των συσκευών στην τοπολογία

Ο constructor NetNode, δημιουργεί δύο αντικείμενα με την βιβλιοθήκη plain-draggable. Το ένα είναι η αναπαράσταση της συσκευής ενώ το άλλο η αναπαράσταση της ονομασίας της συσκευής. Η κύρια ιδέα είναι ότι όταν ο χρήστης κουνάει την συσκευή μέσα στην τοπολογία τότε πρέπει η ονομασία της συσκευής να ακολουθάει την συσκευή. Ταυτόχρονα πρέπει να επιτρέπεται η αλλαγή της θέσης της ονομασίας στην οθόνη. Η θέση των συσκευών και των ονομασιών, ορίζεται με τα πεδία x και y των συσκευών της τοπολογίας. Για να επιτευχθούν όλοι οι στόχοι που είχαν οριστεί για την τοπολογία, χρειάστηκε να υλοποιηθούν 3 ακροατές συμβάντων. Ο onDrag ακροατής εκτελεί την μέθοδο labelPosition όπου υπολογίζει ξανά την θέση της ονομασίας της συσκευής στην οθόνη, καθώς και μια σειρά από εντολές για την μετακίνηση των καλωδίων. Αυτό το συμβάν πραγματοποιείται καθόλη την διάρκεια όπου ο χρήστης μετακινεί την εκάστοτε συσκευή στην τοπολογία. Ο onDragEnd ακροατής αποθηκεύει τα καινούργια δεδομένα στο store, όπου στην προκειμένη περίπτωση είναι οι νέες συντεταγμένες (x και y) της συσκευής πάνω στην τοπολογία. Αυτό το συμβάν εκτελείται όταν ο χρήστης «αφήσει» την συσκευή σε ένα σημείο της τοπολογίας. Και τέλος ο dbClick ακροατής όπου κάθε φορά που ο χρήστης πατάει διπλό κλικ στην συσκευή η εφαρμογή πηγαίνει στην οθόνη της συσκευής που πατήθηκε.

```
constructor(id,topoId,node,label,links,ifs, goToNodeInfo){
    this.id = id;
    this.topoId = topoId
    this.ifs = ifs;
    this.node = node;
    this.links = links;
    this.label = {element: label.name, x: label.x, y: label.y};
    this.draglabel = new plainDraggable(this.label.element);
    this.dragnode = new plainDraggable(node);
    this.draglabel.onDragEnd = () =>{
        this.label.y = this.draglabel.top - this.dragnode.top;
        this.label.x = this.draglabel.left - this.dragnode.left
    }

    this.dragnode.onDrag = () => {
        this.labelPosition();
    }
    this.dragnode.onDragEnd = () => {
        this.labelPosition();
        store.default.dispatch('UserData/updateTopology', {
            id: this.topoId,
            node: {
                id: this.id,
                x: this.dragnode.left,
                y: this.dragnode.top
            },
            changedFromUser: true
        });
    }

    this.node.addEventListener('dblclick', function () {
        const ip = ifs.find(inter => inter.mainIf.value === true).ip_address.value
        console.log(ip)
        console.log(goToNodeInfo)
        goToNodeInfo('device?ip='+ (ip.includes('/') ? ip.slice(0, -(ip.length - ip.indexOf('/'))) : ip))
    })

}
```

Σχήμα 8.18: Κώδικας του constructor για το στοιχείο (element) της συσκευής της τοπολογίας

Εφόσον δημιουργηθούν οι συσκευές στην τοπολογία το επόμενο βήμα είναι να βρεθούν τα καλώδια. Τα καλώδια βρίσκονται από τους πίνακες CDP κάθε συσκευής όπου προσδιορίζονται οι γείτονες των interfaces των συσκευών. Όταν βρεθούν ελέγχεται για κάθε γείτονα αν ο γείτονας υπάρχει στην συγκεκριμένη τοπολογία και αν δεν έχει δημιουργηθεί ήδη το καλώδιο. Στην περίπτωση που περνάει και τους δύο ελέγχους εκτελείται η συνάρτηση addLink. Στην addLink δημιουργείται το εκάστοτε καλώδιο χρησιμοποιώντας τον δεύτερο constructor που αναφέρθηκε παραπάνω, Link. Ο constructor δημιουργεί ένα αντικείμενο όπου περιέχονται οι απαραίτητες πληροφορίες ενός καλωδίου, καθώς και οι απαραίτητες μέθοδοι που απαιτούνται.

```
constructor(start,end,ifstart,ifend){
    this.start = start;
    this.end = end;
    this.ifstart = ifstart;
    this.ifend = ifend;
    this.linkAttrs.startLabel = LeaderLine.captionLabel(this.ifstart.name, this.linkFonts);
    this.linkAttrs.endLabel = LeaderLine.captionLabel(this.ifend.name, this.linkFonts);
    this.link = new LeaderLine(start,end,this.linkAttrs);
    ifstart.state ? this.link["startPlugColor"] = this.linkColors.colorUp : this.link["startPlugColor"] = this.linkColors.colorDown;
    ifend.state ? this.link["endPlugColor"] = this.linkColors.colorUp : this.link["endPlugColor"] = this.linkColors.colorDown;
}

setDown = (node) => {
    if(node == this.start){
        this.link["startPlugColor"] = this.linkColors.colorDown;
        this.ifstart.state = false
    }else if(node == this.end){
        this.link["endPlugColor"] = this.linkColors.colorDown;
        this.ifend.state = false
    }
}

setUp = (node) => {
    if(node == this.start){
        this.link["startPlugColor"] = this.linkColors.colorUp;
        this.ifstart.state = true
    }else if(node == this.end){
        this.link["endPlugColor"] = this.linkColors.colorUp;
        this.ifend.state = true
    }
}
```

Σχήμα 8.19: Κώδικας του constructor για το στοιχείο (element) του καλωδίου της τοπολογίας

Προκειμένου να κατανοηθεί η λειτουργία των τεχνολογιών που χρησιμοποιήθηκαν για την επίτευξη της τοπολογίας θα αναλυθεί ένα σενάριο όπου ένας Α χρήστης, ο οποίος έχει συνδεθεί στην πλατφόρμα, μετακινεί μια συσκευή στην τοπολογία, καθώς και ένας χρήστης Β ο οποίος έχει συνδεθεί με τα ίδια διαπιστευτήρια στην πλατφόρμα, και έχει ανοιχτή την οθόνη της τοπολογίας.

Όταν ο Α χρήστης μετακινεί μια συσκευή τότε υπολογίζονται ξανά όλες οι θέσεις των καλωδίων, που σχετίζονται με αυτή την συσκευή, καθώς και της ετικέτας της ονομασίας της. Όταν «αφήσει» την συσκευή σε ένα σημείο της τοπολογίας τότε αυτομάτως αποθηκεύονται οι νέες συντεταγμένες της συσκευής της τοπολογίας, στο store. Αφού άλλαξαν τα δεδομένα της τοπολογίας ενεργοποιείται ο watcher και στέλνει μέσω των sockets την αλλαγή των συντεταγμένων στον server. Ο server από την μεριά του αποθηκεύει τα νέα δεδομένα στην βάση δεδομένων MongoDB και ενημερώνει όλους τους χρήστες που είναι συνδεδεμένοι με το Username του Α, για τις αλλαγές στην τοπολογία, μέσω των sockets.

Ο Β αφού είναι συνδεδεμένος με τα διαπιστευτήρια του Α θα ακούσει ο Socket IO client τις αλλαγές αυτές. Αφού τις αποθηκεύσει στο store, ο watcher που βρίσκεται μέσα στην τοπολογία θα ανιχνεύσει ότι οι πληροφορίες της τοπολογίας έχουν αλλάξει και θα ξανά ορίσει τα νέα δεδομένα. Με αυτόν τον τρόπο όταν ο χρήστης Α μετακινεί μια συσκευή, ο χρηστής Β θα το βλέπει κατευθείαν στην οθόνη του.

8.9 Επίλογος

Στο κεφάλαιο αυτό αναλύθηκαν όλες οι τεχνικές που χρησιμοποιήθηκαν προκειμένου να υλοποιηθεί η front-end εφαρμογή, όπου είναι υπεύθυνη για την αλληλεπίδραση του χρήστη με την πλατφόρμα. Χρησιμοποιήθηκαν αρκετές διαφορετικές τεχνικές και τεχνολογίες πράγμα που το έκανε αρκετά περίπλοκο στην υλοποίηση. Εφόσον αναλύθηκε και η front-end εφαρμογή και ολοκληρώθηκε με αυτό το κεφάλαιο η ανάλυση του κώδικα, το επόμενο βήμα είναι η παρουσίαση της πλατφόρμας AutoNet που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής.

Κεφάλαιο 9ο: Συμπεράσματα ή/και προτάσεις βελτίωσης

Η διαχείριση των δικτύων είναι ένας πολύ κρίσιμος τομέας στην εύρυθμη λειτουργία των δικτύων, διότι αν μη τι άλλο, μπορεί να αποτρέψει ανεπιθύμητες καταστάσεις. Στις μέρες μας καθίσταται αναγκαία η ύπαρξη ενός διαχειριστικού εργαλείου, καθώς τα δίκτυα επεκτείνονται συνεχώς με πολύ γρήγορους ρυθμούς. Επομένως, προκειμένου να εξασφαλιστεί η ασφάλεια, η αξιοπιστία και η βέλτιστη απόδοση απαιτείται από τους διαχειριστές να πάρουν σωστές αποφάσεις και να ενεργήσουν με τον σωστό τρόπο χρησιμοποιώντας την κατάλληλη διαχειριστική πλατφόρμα δικτύων που εξυπηρετεί τον σκοπό αυτό.

Στα πλαίσια της παρούσας Δ.Ε. αναπτύχθηκε ένα εργαλείο δυναμικής διαχείρισης που προσφέρει στους διαχειριστές ένα πλήθος χρήσιμων και κρίσιμων λειτουργιών διαχείρισης και παρακολούθησης της λειτουργίας του δικτύου. Η συγκεκριμένη πλατφόρμα σχεδιάστηκε με τέτοιο τρόπο ώστε να είναι αποδοτική ως προς την δέσμευση πόρων ενός ηλεκτρονικού υπολογιστή καθώς και να παρέχει εύκολες, κατανοητές και ευέλικτες λειτουργίες. Τέλος παρέχει ένα σύγχρονο γραφικό περιβάλλον για πιο αποδοτική και γρήγορη διαχείριση

Υπάρχουν αρκετοί τομείς που χρήζουν αναβάθμισης. Οι σημαντικότεροι περιγράφονται σύντομα. Αρχικά το πρώτο που πρέπει να γίνει είναι να μετατραπεί η front-end και η back-end εφαρμογή από JavaScript σε TypeScript. Αυτό θα βοηθούσε στο μέλλον να γίνεται πιο εύκολα η αποσφαλμάτωση καθώς και στο ότι ο κώδικας θα ήταν πιο οργανωμένος. Ένας ακόμα στόχος που έχει οριστεί για μελλοντική αναβάθμιση είναι η υποστήριξη περισσότερων συσκευών ακόμα και από άλλους κατασκευαστές όπως είναι η Mikrotik. Αυτός ο στόχος είναι αναγκαίο να επιτευχθεί καθώς στα σύγχρονα δίκτυα εμπλέκονται περισσότερες συσκευές από Cisco routers και switches. Επιπλέον μια ακόμα σημαντική αναβάθμιση θα είναι η δημιουργία μιας σελίδας εγγραφής όπου θα μπορεί ο κάθε διαχειριστής να δημιουργεί καινούργιους χρήστες οι οποίοι θα μπορούν να είναι είτε διαχειριστές είτε απλοί χρήστες. Επιπρόσθετα στους στόχους για την αναβάθμιση είναι και η προσθήκη πολύ περισσότερων SNMP oids ώστε η εφαρμογή να προσφέρει και άλλες λειτουργίες του πρωτοκόλλου SNMP. Τέλος θα μπορούσε να αναπτυχθεί και μια android και iOS εφαρμογή ώστε να μπορεί ο χρήστης να επεμβαίνει εύκολα και γρήγορα από την φορητή συσκευή του αλλά και να λαμβάνει σημαντικές ειδοποιήσεις για συμβάντα που μπορεί να προκύψουν στις συσκευές.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Hassan, R., Razali, R., Mohseni, S., Mohamad, O., & Ismail, Z. (2010). Architecture of network management tools for heterogeneous system. arXiv preprint arXiv:1001.1967.

[2] Kijazi, Ahmed, and Kisangiri Michael. "A Step on Developing Network Monitoring Tools." (2014).

[3] Iyamuremye, Blake, and Hisato Shima. "Network security testing tools for SMEs (small and medium enterprises)." 2018 IEEE International Conference on Applied System Invention (ICASI). IEEE, 2018.
[4] Zhou, Donghao, et al. "A survey on network data collection." Journal of Network and Computer Applications 116 (2018): 9-23.

[5] Chowdhury, NM Mosharaf Kabir, and Raouf Boutaba. "A survey of network virtualization." Computer Networks 54.5 (2010): 862-876.

[6] "HTML For Beginners The Easy Way: Start Learning HTML & CSS Today" https://html.com/ [Online; accessed 13-June-2022]

[7] "Selectors Level 4", https://www.w3.org/TR/selectors/ [Online; accessed 13-June-2022]

[8] "What is the difference between SCSS and SASS ?" https://www.geeksforgeeks.org/what-is-the-difference-between-scss-and-sass/ [Online; accessed 13-June-2022]

[9] "Sass Basics", https://sass-lang.com/guide [Online; accessed 13-June-2022]

[10] "Introduction Electron", https://www.electronjs.org/docs/latest [Online; accessed 13-June-2022]

[11] "TypeScript for the New Programmer", https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html [Online; accessed 13-June-2022]

[12] "Component-Based Development (CBD)", https://www.techopedia.com/definition/31002/component-based-development-cbd [Online; accessed 13-June-2022]

[13] "Quick Start Vue.js", https://vuejs.org/guide/quick-start.html [Online; accessed 13-June-2022]

[14] "Rendering Mechanism", https://vuejs.org/guide/extras/rendering-mechanism.html [Online; accessed 13-June-2022]

[15] "Virtual DOM and Internals", https://reactjs.org/docs/faq-internals.html [Online; accessed 13-June-2022]

[16] "The Virtual DOM", https://www.codecademy.com/article/react-virtual-dom [Online; accessed 13-June-2022]

[17] "How is Virtual DOM faster?", https://programmingwithmosh.com/react/react-virtual-dom-explained/ [Online; accessed 13-June-2022]

[18] "State Management", https://vuejs.org/guide/scaling-up/state-management.html [Online; accessed 13-June-2022]

[19] "Props", https://vuejs.org/guide/components/props.html [Online; accessed 13-June-2022]

[20] "Built-in Directives", https://vuejs.org/api/built-in-directives.html [Online; accessed 13-June-2022]

[21] “Vue.js Methods”, https://www.geeksforgeeks.org/vue-js-methods [Online; accessed 13-June-2022]

[22] “Understanding computed properties in Vue.js”, https://blog.logrocket.com/understanding-computed-properties-in-vue-js/ [Online; accessed 13-June-2022]

[23] “Vue.js Watchers”, https://www.geeksforgeeks.org/vue-js-watchers [Online; accessed 13-June-2022]

[24] “Options: Lifecycle”, https://vuejs.org/api/options-lifecycle.html [Online; accessed 13-June-2022]

[25] “Lifecycle Diagram”, https://vuejs.org/guide/essentials/lifecycle.html#lifecycle-diagram [Online; accessed 13-June-2022]

[26] “Getting Started – Vue Router”, https://router.vuejs.org/guide/#html [Online; accessed 13-June-2022]

[27] “What is Vuex?”, https://vuex.vuejs.org/ [Online; accessed 13-June-2022]

[28] “vuex-persistedstate”, https://www.npmjs.com/package/vuex-persistedstate [Online; accessed 13-June-2022]

[29] “Why Quasar?”, https://quasar.dev/introduction-to-quasar [Online; accessed 13-June-2022]

[30] “What You Need To Know About Node.js”, https://readwrite.com/what-you-need-to-know-about-nodejs [Online; accessed 13-June-2022]

[31] “About Node.js®”, https://nodejs.org/en/about/ [Online; accessed 13-June-2022]

[32] “What Is package.json?”, https://heynode.com/tutorial/what-packagejson/ [Online; accessed 13-June-2022]

[33] “Package.Json in node js”, https://riteeksrivastava.medium.com/package-json-in-node-js-c01df5eed230 [Online; accessed 13-June-2022]

[34] “Express”, https://expressjs.com/ [Online; accessed 13-June-2022]

[35] “What Is MongoDB?”, https://www.mongodb.com/what-is-mongodb [Online; accessed 13-June-2022]

[36] “How To Install PIP to Manage Python Packages On Windows”, https://phoenixnap.com/kb/install-pip-windows [Online; accessed 14-June-2022]

[37] “Chapter 15. Nmap Reference Guide”, https://nmap.org/book/man.html#man-description [Online; accessed 14-June-2022]

[38] “System-specific parameters and functions”, https://docs.python.org/3/library/sys.html [Online; accessed 14-June-2022]

[39] “JSON encoder and decoder”, https://docs.python.org/3/library/json.html [Online; accessed 14-June-2022]

[40] “IPv4/IPv6 manipulation library”, https://docs.python.org/3/library/ipaddress.html [Online; accessed 14-June-2022]

[41] “Module netmiko”, https://pyneng.readthedocs.io/en/latest/book/18_ssh_telnet/netmiko.html [Online; accessed 14-June-2022]

[42] “Python RegEx”, https://www.w3schools.com/python/python_regex.asp [Online; accessed 14-June-2022]

[43] “NTC TEMPLATES”, https://github.com/networktocode/ntc-templates [Online; accessed 14-June-2022]

[44] “Readme.md - npm”, https://github.com/npm/cli/commit/4626dfa73b7847e9c42c1f799935f8242794d020 [Online; accessed 14-June-2022]

[45] “JavaScript Cookie”, https://www.npmjs.com/package/js-cookie [Online; accessed 14-June-2022]

[46] “Chart.js”, https://www.w3schools.com/js/js_graphics_chartjs.asp [Online; accessed 14-June-2022]

[47] “Chart.js”, https://www.chartjs.org/ [Online; accessed 14-June-2022]

[48] “Xterm.js”, https://github.com/xtermjs/xterm.js/ [Online; accessed 14-June-2022]

[49] “Prism”, https://github.com/PrismJS/prism [Online; accessed 14-June-2022]

[50] “body-parser”, https://www.npmjs.com/package/body-parser [Online; accessed 14-June-2022]

[51] “Socket.IO”, https://socket.io/ [Online; accessed 14-June-2022]

[52] “What is Axios?”, https://axios-http.com/docs/intro [Online; accessed 14-June-2022]

[53] “Difference between Fetch and Axios.js for making http requests”, https://www.geeksforgeeks.org/difference-between-fetch-and-axios-js-for-making-http-requests/ [Online; accessed 14-June-2022]

[54] “Hashing passwords in NodeJS with bcrypt library tutorial”, https://sebhastian.com/bcrypt-node/ [Online; accessed 14-June-2022]

[55] “default-gateway”, https://www.npmjs.com/package/default-gateway [Online; accessed 14-June-2022]

[56] “dotenv”, https://www.npmjs.com/package/dotenv [Online; accessed 14-June-2022]

[57] “IP address utilities for node.js”, https://www.npmjs.com/package/ip [Online; accessed 14-June-2022]

[58] “Joi API”, https://joi.dev/api/?v=17.6.0 [Online; accessed 14-June-2022]

[59] “Introduction to JSON Web Tokens”, https://jwt.io/introduction [Online; accessed 14-June-2022]

[60] “Elegant MongoDB object modeling for Node.js”, https://mongoosejs.com/ [Online; accessed 14-June-2022]

[61] “A ping wrapper for nodejs”, https://www.npmjs.com/package/ping [Online; accessed 14-June-2022]

[62] “python-shell”, https://www.npmjs.com/package/python-shell [Online; accessed 14-June-2022]

[63] “ssh2”, https://www.npmjs.com/package/ssh2 [Online; accessed 14-June-2022]

[64] “NodeJS Syslog Server”, https://www.npmjs.com/package/syslog-server [Online; accessed 14-June-2022]

[65] “net-snmp”, https://www.npmjs.com/package/net-snmp [Online; accessed 14-June-2022]

[66] “PlainDraggable”, https://github.com/anseki/plain-draggable [Online; accessed 14-June-2022]

[67] “LeaderLine”, https://github.com/anseki/leader-line [Online; accessed 14-June-2022]

[68] “Getting Started – Visual Studio Code”, https://code.visualstudio.com/docs [Online; accessed 14-June-2022]

[69] “What is MongoDB Compass?”, https://www.mongodb.com/docs/compass/current/ [Online; accessed 14-June-2022]

[70] “Getting Started with GNS3”, https://docs.gns3.com/docs/ [Online; accessed 14-June-2022]

[71] “Cisco IOS (Cisco Internetwork Operating System)”, https://www.techtarget.com/searchnetworking/definition/Cisco-IOS-Cisco-Internetwork-Operating-System [Online; accessed 14-June-2022]

[72] “Cisco IOS Command Hierarchy”, https://www.cisco.com/E-Learning/bulk/public/tac/cim/cib/using_cisco_ios_software/02_cisco_ios_hierarchy.htm [Online; accessed 14-June-2022]

[73] “SSH Protocol – Secure Remote Login and File Transfer”, https://www.ssh.com/academy/ssh/protocol [Online; accessed 14-June-2022]

[74] “The SSH protocol”, https://www.ssh.com/academy/ssh#the-ssh-protocol [Online; accessed 14-June-2022]

[75] “Cisco Discovery Protocol (CDP)”, https://learningnetwork.cisco.com/s/article/cisco-discovery-protocol-cdp-x [Online; accessed 14-June-2022]

[76] “What is SNMP?”, https://www.manageengine.com/network-monitoring/what-is-snmp.html [Online; accessed 14-June-2022]

[77] “OID Repository - 1.3.6.1.2.1.2.2.1”, http://oid-info.com/get/1.3.6.1.2.1.2.2.1 [Online; accessed 14-June-2022]

[78] “Syslog: Configure syslog server logging (Cisco)”, https://www.grandmetric.com/knowledge-base/design_and_configure/syslog-configure-syslog-server-logging-cisco/ [Online; accessed 14-June-2022]

[79] “Chapter: Embedded Event Manager Overview”, https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/eem/configuration/xe-16/eem-xe-16-book/eem-overview.html [Online; accessed 14-June-2022]

[80] “Host Discovery”, https://nmap.org/book/man-host-discovery.html [Online; accessed 14-June-2022]

[81] “Usage and Examples - Nmap”, https://nmap.org/book/osdetect-usage.html [Online; accessed 14-June-2022]