ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

WEB ΕΦΑΡΜΟΓΗ ΓΙΑ ΤΗΝ ΚΑΤΑΓΡΑΦΗ ΚΑΙ ΑΠΕΙΚΟΝΙΣΗ ΤΗΣ ΕΠΙΔΟΣΗΣ ΑΠΟΜΑΚΡΥΣΜΕΝΩΝ ΣΥΣΤΗΜΑΤΩΝ

Διπλωματική Εργασία

Χρήστος Βούλγαρης

Επιβλέπων: Θάνος Γεώργιος

Φεβρουάριος 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

WEB ΕΦΑΡΜΟΓΗ ΓΙΑ ΤΗΝ ΚΑΤΑΓΡΑΦΗ ΚΑΙ ΑΠΕΙΚΟΝΙΣΗ ΤΗΣ ΕΠΙΔΟΣΗΣ ΑΠΟΜΑΚΡΥΣΜΕΝΩΝ ΣΥΣΤΗΜΑΤΩΝ

Διπλωματική Εργασία

Χρήστος Βούλγαρης

Επιβλέπων: Θάνος Γεώργιος

Φεβρουάριος 2022

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

WEB BASED APPLICATION FOR MONITORING PERFORMANCE OF REMOTE OPERATING SYSTEMS

Diploma Thesis

Christos Voulgaris

Supervisor: Thanos Georgios

February 2022

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων **Θάνος Γεώργιος**
Μέλος Ε.ΔΙ.Π., Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Αντωνόπουλος Χρήστος**
Αναπληρωτής Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Δασκαλοπούλου Ασπασία**
Επίκουρη Καθηγήτρια, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Ευχαριστίες

Αρχικά, θέλω να ευχαριστήσω την οικογένεια μου για την αγάπη και τη στήριξη που μου προσέφεραν όλα αυτά τα χρόνια, καθώς και τους φίλους μου για τη στήριξη και τις αξέχαστες στιγμές.

Επίσης, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή Θάνο Γεώργιο και τους καθηγητές Αντωνόπουλο Χρήστο και Δασκαλοπούλου Ασπασία για τη συνεργασία μας και τις πολύτιμες συμβουλές τους.

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

«Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Δηλώνω επίσης ότι τα αποτελέσματα της εργασίας δεν έχουν χρησιμοποιηθεί για την απόκτηση άλλου πτυχίου. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής».

Ο Δηλών

Χρήστος Βούλγαρης

Διπλωματική Εργασία

WEB ΕΦΑΡΜΟΓΗ ΓΙΑ ΤΗΝ ΚΑΤΑΓΡΑΦΗ ΚΑΙ ΑΠΕΙΚΟΝΙΣΗ ΤΗΣ ΕΠΙΔΟΣΗΣ ΑΠΟΜΑΚΡΥΣΜΕΝΩΝ ΣΥΣΤΗΜΑΤΩΝ

Χρήστος Βούλγαρης

Περίληψη

Ο όρος Information Technology (IT) αφορά τη χρήση υπολογιστών και γενικότερα συσκευών για τη δημιουργία, επεξεργασία, αποθήκευση, ασφάλεια και ανταλλαγή όλων των μορφών ηλεκτρονικών δεδομένων. Οι συσκευές αυτές υποστηρίζουν πληθώρα IT εφαρμογών, όπως συντήρηση δικτύων, προστασία υπολογιστικών συστημάτων κ.α. και χρησιμοποιούνται από πολλές επιχειρήσεις για την επεξεργασία δεδομένων. Οι κίνδυνοι που αφορούν το IT περιλαμβάνουν βλάβες στο υλικό, ανθρώπινα σφάλματα, ιούς και κακόβουλες επιθέσεις. Επομένως, είναι σημαντικός ο εντοπισμός αυτών των κινδύνων για την αποφυγή οικονομικών επιπτώσεων. Το workstation monitoring αποτελεί ένα από τα μέτρα πρόληψης των προαναφερθέντων κινδύνων. Με την ανάλυση της κατάστασης των workstations σε πραγματικό χρόνο, αυξάνεται η αξιοπιστία των συστημάτων αποτρέποντας βλάβες και απώλειες δεδομένων. Στόχος της διπλωματικής εργασίας είναι να δημιουργήσει μια ολοκληρωμένη λύση στο πλαίσιο του workstation monitoring, υλοποιημένη από το μηδέν, προσφέροντας στο χρήστη ευελιξία μέσω των δυνατοτήτων της. Η λύση βασίζεται στην δημιουργία ενός διακομιστή, στον οποίο αποθηκεύονται τα δεδομένα μετά τη συλλογή τους από κάθε σύστημα. Για τη παρακολούθηση και προβολή των συστημάτων μέσω των δεδομένων που λαμβάνονται, αναπτύσσεται ένας web client με γνώμονα την απλή εμπειρία του χρήστη.

Diploma Thesis

WEB BASED APPLICATION FOR MONITORING PERFORMANCE OF REMOTE OPERATING SYSTEMS

Christos Voulgaris

Abstract

The use of computers and devices in general to create, process, store, secure, and exchange all forms of electronic data is known as information technology (IT). These devices are used in a wide range of IT applications including network management, system security, data processing etc.. Hardware and software failures, human error, viruses, and malicious attacks are all examples of IT risks. As a result, it is critical for businesses to detect these risks to avoid huge financial costs. One of the risk-prevention techniques is workstation monitoring as it helps to strengthen the reliability of the entire system by preventing malfunctions and data loss through enabling real-time analysis of the state of working systems. The aim of the dissertation is to create a complete solution in the context of workstation monitoring, implemented from scratch, offering the user flexibility through its capabilities. The solution is based on the existence of a server where the data is stored after their collection by each system. To monitor the workstations, a web client is developed to provide a simple and flexible user experience.

Πίνακας περιεχομένων

Κατάλογος σχημάτων

Συντομογραφίες

κ.λπ.	και λοιπά
ΒΔ	Βάση Δεδομένων
IT	Information Technology
IoT	Internet of Things
MQTT	Message Queuing Telemetry Transport
DRF	Django Rest Framework
SQL	Structured Query Language
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
JSON	JavaScript Object Notation
DRY	Don’t Repeat Yourself
HTTP	Hypertext Transfer Protocol
API	Application Programming Interface
REST	Representational State Transfer
URL	Uniform Resource Locator
XML	Extensible Markup Language
W3C	World Wide Web Consortium
WHATWG	Web Hypertext Application Technology Working
OOP	Object Oriented Programming
JSX	Javascript Syntax Extension
CORS	Cross-Origin Resource Sharing
JWT	JSON Web Token

Κεφάλαιο 1

Εισαγωγή

1.1 Αντικείμενο της διπλωματικής

Η τεχνολογία πληροφοριών (Information Technology) αναφέρεται στη χρήση συσκευών για τη δημιουργία, επεξεργασία, αποθήκευση, ασφάλεια και ανταλλαγή όλων των μορφών ηλεκτρονικών δεδομένων. Ήδη σε όλο τον κόσμο υπάρχουν δισεκατομμύρια συσκευές συνδεδεμένες στο διαδίκτυο, ο αριθμός των οποίων αυξάνεται ραγδαία. Αυτές οι συσκευές μπορούν να χρησιμοποιηθούν σε πλήθος IT εφαρμογών, όπως συντήρηση και αναβάθμιση δικτύων, προστασία συστημάτων, ανάπτυξη διαδικτυακών εφαρμογών κ.λπ. [6].

Η άνοδος της οικονομίας και η ψηφιακή επανάσταση οδήγησαν πολλές επιχειρήσεις να εξαρτώνται όλο και περισσότερο από τα δεδομένα και την επεξεργασία τους [7]. Επομένως, διάφορες αστοχίες στο IT μπορούν να προκαλέσουν σοβαρά προβλήματα σε αρκετές επιχειρήσεις, με δυσμενείς οικονομικές επιπτώσεις, πτώσεις στην παραγωγικότητα, απώλεια δεδομένων κ.α. [8].

Η καταγραφή και η πρόβλεψη αυτών των κινδύνων είναι γνωστή ως workstation monitoring και έχει ως αποτέλεσμα τη βελτίωση της απόδοσης των συστημάτων και την έγκαιρη ενημέρωση για την αποφυγή σημαντικών προβλημάτων. Πιο συγκεκριμένα, περιλαμβάνει τον έλεγχο της κατάστασης των συστημάτων τόσο στο υλικό (hardware) όσο και στον όγκο εργασίας (workload) για τη καλύτερη διαχείρισή τους, καθώς και την άμεση αποστολή ειδοποιήσεων κατά τον εντοπισμό ενός προβλήματος [9].

Υπάρχουν ήδη πολλές εταιρείες (Netdata, Nagios, Prometheus) που προσφέρουν ολοκληρωμένες εφαρμογές-πλατφόρμες στον τομέα του workstation monitoring είτε ανοικτού περιεχομένου είτε προς αγορά. Οι εφαρμογές αυτές συνήθως χρησιμοποιούνται από μεγάλες

επιχειρήσεις που διαθέτουν πλήθος συσκευών για τη βελτιστοποίηση της διαδικασίας της παραγωγής.

1.1.1 Συνεισφορά

Σκοπός της διπλωματικής είναι η συνεισφορά στον τομέα του workstation monitoring με την παρουσίαση μιας υλοποίησης κατασκευασμένης απο το μηδέν και βασισμένης σε εργαλεία ανοιχτού περιεχομένου, παρέχοντας τη δυνατότητα ελέγχου των συστημάτων από τον χρήστη. Η υλοποίηση βασίζεται στην εξαγωγή των δεδομένων-μετρήσεων από το λειτουργικό σύστημα με τη χρήση shell scripts και την επικοινωνία μέσω του πρωτοκόλλου MQTT με τον διακομιστή για την επεξεργασία και την αποθήκευσή τους στη βάση δεδομένων. Τέλος, δημιουργείται μια εφαρμογή για την προβολή των δεδομένων από τον τελικό χρήστη.

Αναλυτικότερα οι δυνατότητες του συστήματος είναι οι εξής:

• Προβολή γενικών πληροφοριών για το χρήστη και μετρήσεων για κάθε συσκευή. Οι μετρήσεις αφορούν τον επεξεργαστή, τη μνήμη, το δίσκο και τη μπαταρία του συστήματος.

• Προβολή διαδραστικών διαγραμμάτων για κάθε συσκευή βασισμένα στο χρονικό διάστημα που επιλέγει ο χρήστης.

• Δημιουργία ορίων των μετρήσεων για κάθε συσκευή. Αν ξεπεραστεί κάποιο όριο (άνοδος της θερμοκρασίας του επεξεργαστή, εξάντληση της μπαταρίας), ο χρήστης θα λάβει ειδοποίηση τόσο στην εφαρμογή, μέσω ξεχωριστής σελίδας ειδοποιήσεων για τη διαχείρισή τους, όσο και στο email του.

• Ταυτοποίηση του χρήστη για την είσοδο στην εφαρμογή.

1.2 Οργάνωση του τόμου

Η διπλωματική εργασία έχει την ακόλουθη δομή. Αρχικά, στο Κεφάλαιο 2 αναλύεται το υπόβαθρο της διπλωματικής. Πιο συγκεκριμένα, παρουσιάζονται αναλυτικά οι σημαντικές έννοιες που πρέπει να γνωρίζει ο αναγνώστης για την κατανόηση της διπλωματικής. Οι έννοιες αυτές περιλαμβάνουν τα shell scripts, το πρωτόκολλο MQTT, καθώς και μια εισαγωγή στη δημιουργία διαδικτυακών εφαρμογών, τόσο στο επίπεδο του διακομιστή όσο και στο επίπεδο του χρήστη. Έπειτα, το Κεφάλαιο 3 περιγράφει την υλοποίηση. Αναλυτικότερα, περιγράφεται η διαδικασία εξαγωγής των δεδομένων από το λειτουργικό σύστημα με τη χρήση shell scripts και η αποστολή τους μέσω του πρωτοκόλλου MQTT για αποθήκευση στη βάση

δεδομένων. Έπειτα, προσδιορίζεται ο σχεδιασμός της ΒΔ με επεξήγηση κάθε οντότητας-πίνακα. Τέλος, αναφέρεται ο τρόπος με τον οποίο δημιουργήθηκε η διαδικτυακή εφαρμογή τόσο στην πλευρά του διακομιστή, με τη χρήση του Django Rest Framework, όσο και στην πλευρά του χρήστη χρησιμοποιώντας το React.JS. Στη συνέχεια, στο Κεφάλαιο 4, γίνεται η παρουσίαση του γραφικού περιβάλλοντος και των δυνατοτήτων της εφαρμογής στη πλευρά του χρήστη. Τέλος, στο Κεφάλαιο 5 παρατίθεται η σύνοψη της διπλωματικής με τα συμπεράσματα και τις μελλοντικές επεκτάσεις.

Κεφάλαιο 2

Υπόβαθρο

2.1 Workstation Monitoring

Η καταγραφή και η πρόβλεψη της λειτουργίας των συστημάτων είναι γνωστή ως workstation monitoring και έχει ως αποτέλεσμα τη βελτίωση της απόδοσής τους και την έγκαιρη ενημέρωση για την αποφυγή σημαντικών επιπτώσεων. Το workstation monitoring [10] υιοθετείται από όλο και περισσότερες επιχειρήσεις. Αυτό οφείλεται στο γεγονός ότι επιτρέπει την παρακολούθηση της ομαλής λειτουργίας κάθε συστήματος, καθώς και τον εντοπισμό βλαβών σε αυτά. Βασικές εφαρμογές του αποτελούν οι εξης:

• Απόδοση του συστήματος.

Χάρη στο workstation monitoring είναι δυνατή η παρακολούθηση κάθε υπολογιστή, ανεξάρτητα από το λειτουργικό σύστημα (Windows, Linux, Mac OS) που διαθέτει. Ο τακτικός έλεγχος της κατάστασης, στην οποία βρίσκονται τα συστήματα, επιτρέπει τη βέλτιστη διαχείριση τους και την προβλεψη βλαβών. Πιο συγκεκριμένα, ελέγχεται η κατάσταση του υλικού με σκοπό τη διάγνωση της αξιοπιστίας του συστήματος (πρόβλεψη σφαλμάτων στο σκληρό δίσκο, άνοδος στη θερμοκρασία του επεξεργαστή). Επίσης, εποπτεύει τον φόρτο εργασίας του workstation, μέσω της καταγραφής του αριθμού των χρηστών, των αιτημάτων και της απόδοσης του δικτύου. Τέλος, σημαντικά στοιχεία του ελέγχου αποτελούν ο χρόνος λειτουργίας των εφαρμογών (uptime) και ο χρόνος απόκρισης των προγραμμάτων (response time).

• Καταγραφή σφαλμάτων και ασφάλεια.

Μέσω της εποπτείας που παρέχεται, είναι δυνατό να εντοπιστεί η παρουσία κατεστραμμένων αρχείων χάρη στα αρχεία καταγραφής. Επιπλέον, χρησιμοποιείται για τον εντοπισμό των επιθέσεων στο τείχος προστασίας και τη λήψη αναφορών σχετικών με εκτελέσεις κακό-

βουλου λογισμικού.

• Έγκαιρη ενημέρωση.

Εξίσου σημαντικό χαρακτηριστικό αποτελεί η έγκαιρη αποστολή ειδοποιήσεων σε περίπτωση εντοπισμού οποιουδήποτε προβλήματος, προειδοποιώντας τον υπεύθυνο σε πραγματικό χρόνο, ώστε να μπορεί να επέμβει άμεσα.

2.2 Shell Script

Στην παρούσα διπλωματική, για τη συλλογή των δεδομένων από το λειτουργικό σύστημα χρησιμοποιήθηκαν shell scripts. Shell script [11] είναι μια λίστα εντολών σε ένα πρόγραμμα υπολογιστή, που εκτελείται από το κέλυφος λειτουργικού συστήματος Unix (Unix shell) και εξυπηρετεί πολλές διαφορετικές λειτουργίες, όπως η εκτέλεση ενός προγράμματος, η διαχείριση αρχείων και η εκτύπωση κειμένου. Αναλυτικότερα, οι διαφορετικές δυνατότητες που παρέχει είναι οι ακόλουθες:

• Είναι δυνατή η αυτόματη εκτέλεση εντολών, χωρίς να είναι απαραίτητη η χειροκίνητη εισαγωγή κάθε μίας ξεχωριστά στο τερματικό από το χρήστη. Η μετάβαση σε μια εντελώς διαφορετική σύνταξη δεν είναι απαραίτητη, καθώς οι εντολές που υποστηρίζει το shell script είναι ίδιες με αυτές που εισάγονται στο τερματικό.

• Τα σύγχρονα shell scripts διαθέτουν δυνατότητες που βρίσκονται μόνο σε προηγμένες γλώσσες προγραμματισμού, όπως πίνακες, μεταβλητές, σχόλια, βρόγχους κ.λπ.. Πολλές απαιτητικές εφαρμογές μπορούν να γραφτούν με χρήση shell script χρησιμοποιώντας αυτές τις δυνατότητες. Από την άλλη, ένα βασικό μειονέκτημα αποτελεί η απουσία χαρακτηριστικών, όπως κλάσεις και νήματα.

2.3 Πρωτόκολλο MQTT

Μετά τη συλλογή των δεδομένων ακολουθεί η αποστολή τους στον διακομιστή, μέσω του πρωτοκόλλου MQTT, για αποθήκευση στη βάση δεδομένων. Το MQTT (Message Queuing Telemetry Transport) [12] είναι ένα πρωτόκολλο μηνυμάτων που βασίζεται στην αρχιτεκτονική Publish/Subscribe. Το MQTT είναι κατάλληλο για αλληλεπίδραση μηχανής με μηχανή, καθώς ειδικεύεται σε περιβάλλοντα χαμηλού εύρους ζώνης (bandwidth) και υψηλού latency. Επίσης, χρησιμοποιείται σε εφαρμογές IoT και σε περιβάλλοντα cloud.

2.3.1 Publish / Subscribe

Η λογική Publish/Subscribe (γνωστή και ως pub/sub) [13] παρέχει μια εναλλακτική λύση στην κλασική αρχιτεκτονική client-server. Στο μοντέλο client-server, ένας client επικοινωνεί απευθείας με ένα τελικό σημείο (endpoint). Το μοντέλο pub/sub ξεχωρίζει τον client που στέλνει ένα μήνυμα (publisher) από τους clients που λαμβάνουν τα μηνύματα (subscribers). Οι publishers και οι subscribers δεν επικοινωνούν ποτέ απευθείας μεταξύ τους. Στην πραγματικότητα, δεν γνωρίζουν καν ότι ο άλλος υπάρχει. Η μεταξύ τους σύνδεση διεκπεραιώνεται από μια τρίτη συνιστώσα τον ”broker”. Ο ρόλος του broker είναι να φιλτράρει όλα τα εισερχόμενα μηνύματα και να τα διανέμει σωστά στους αντίστοιχους subscribers.

2.3.2 MQTT broker

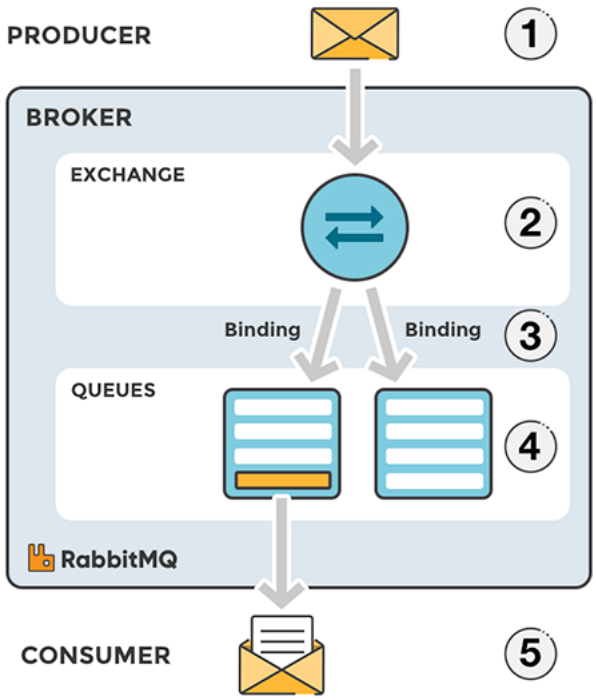
Ο MQTT broker είναι η βάση κάθε πρωτοκόλλου Publish/Subscribe. Ανάλογα με την υλοποίηση, ένας broker μπορεί να διαχειριστεί έως και χιλιάδες ταυτόχρονα συνδεδεμένους MQTT clients. Πιο συγκεκριμένα, είναι υπεύθυνος για τη λήψη και το φιλτράρισμα όλων των μηνυμάτων, την αντιστοίχισή τους στους κατάλληλους clients και την αποστολή των μηνυμάτων σε αυτούς. Σε αρκετές περιπτώσεις ο broker είναι επεκτάσιμος, γεγονός που διευκολύνει τη σύνδεση με διεργασίες στο back-end επίπεδο της εφαρμογής. Αυτό είναι ιδιαίτερα σημαντικό, επειδή εκτίθεται άμεσα στο Διαδίκτυο, εξυπηρετεί πολλούς clients και προωθεί μηνύματα σε συστήματα ανάλυσης και επεξεργασίας. Εν ολίγοις, ο broker είναι ο κεντρικός κόμβος μέσω του οποίου δρομολογείται κάθε μήνυμα. Επομένως, είναι σημαντικό να μπορεί να ενσωματωθεί σε back-end services και να είναι εύκολος στην παρακολούθηση και διαχείριση. Στην υλοποίηση που προτείνεται, επιλέχθηκε ως broker το RabbitMQ.

2.3.3 RabbitMQ

Με δεκάδες χιλιάδες χρήστες, το RabbitMQ [14] είναι ένας από τους πιο δημοφιλείς brokers ανοιχτού περιεχομένου. Το RabbitMQ είναι ελαφρύ και εύκολο στην εγκατάσταση. Υποστηρίζει πολλαπλά πρωτόκολλα ανταλλαγής μηνυμάτων, όπως το MQTT, και μπορεί να εφαρμοστεί για να καλύψει απαιτήσεις μεγάλης κλίμακας. Τέλος, είναι συμβατό με δημοφιλείς γλώσσες προγραμματισμου όπως Java, .NET, PHP, Python, JavaScript, Ruby, Go και πολλές άλλες.

Το RabbitMQ χρησιμοποιεί ειδική ορολογία. Ο όρος “producing” δεν σημαίνει τίποτα

άλλο από την αποστολή μηνυμάτων. Η ουρά (queue) είναι το όνομα ενός ”ταχυδρομικού κουτιού”, που βρίσκεται μέσα στο RabbitMQ, και τα μηνύματα αποθηκεύονται μόνο μέσα σε αυτή. Μια ουρά περιορίζεται μόνο από τα όρια μνήμης και δίσκου του κεντρικού υπολογιστή, είναι ουσιαστικά ένας μεγάλος buffer μηνυμάτων. Πολλοί producers μπορούν να στείλουν μηνύματα που πηγαίνουν σε μία ουρά και πολλοί consumers μπορούν να προσπαθήσουν να λάβουν δεδομένα από μία ουρά. Ο όρος “producing” έχει παρόμοια σημασία με την αποστολή μηνυμάτων. Ένας consumer είναι ένα πρόγραμμα που περιμένει κυρίως να λάβει μηνύματα. Η βασική ιδέα στο μοντέλο ανταλλαγής μηνυμάτων στο RabbitMQ είναι ότι ο producer δεν στέλνει ποτέ κανένα μήνυμα απευθείας σε μια ουρά. Στην πραγματικότητα, πολύ συχνά ο producer δεν γνωρίζει καν εάν ένα μήνυμα θα παραδοθεί σε κάποια ουρά. Αντίθετα, μπορεί να στείλει μηνύματα μόνο σε μια ανταλλαγή (exchange). Η ανταλλαγή από τη μια πλευρά λαμβάνει μηνύματα από producers και από την άλλη πλευρά τα προωθεί σε ουρές. Σχήμα 2.1.



Σχήμα 2.1: Ο τρόπος λειτουργίας του RabbitMQ [1].

2.4 Ανάπτυξη Διαδικτυακών εφαρμογών

Για την διαχείριση των δεδομένων που συλλέχθηκαν, δημιουργήθηκε μια διαδικτυακή εφαρμογή, ικανή για την προβολή τους από το χρήστη και την επεξεργασία και αποθήκευσή τους σε μια βάση δεδομένων. Η ανάπτυξη Διαδικτυακών εφαρμογών [15] είναι η δημιουργία προγραμμάτων-εφαρμογών που βρίσκονται σε απομακρυσμένους servers και παραδίδονται στη συσκευή του χρήστη μέσω του Διαδικτύου. Ένας τελικός χρήστης μπορεί να έχει πρόσβαση σε μια διαδικτυακή εφαρμογή μέσω ενός browser, όπως το Google Chrome, το Safari ή το Mozilla Firefox. Η πλειονότητα των Διαδικτυακών εφαρμογών μπορούν να γραφτούν σε Python, JavaScript, CSS και HTML5. Για τη δημιουργία μιας ολοκληρωμένης διαδικτυακής εφαρμογής είναι απαραίτητη η υλοποίηση τόσο του Back-end (server side) όσο και του Front-end (client side) μέρους.

2.4.1 Back-end

Το back-end αναφέρεται στην πλευρά του server μιας εφαρμογής και αποτελείται συνήθως από τρία μέρη: έναν server, μια εφαρμογή και μια βάση δεδομένων. Οι χρήστες δεν έρχονται σε άμεση επαφή με τον κώδικα που εκτελεί το back-end. Αντιθέτως, ο server μεταβιβάζει τις πληροφορίες της ΒΔ στον τελικό χρήστη. Για την ανάπτυξη back-end εφαρμογών επιλέγονται συνήθως γλώσσες, όπως Ruby, Java και Python. Δημοφιλείς ΒΔ όπως SQLite, MongoDB και PostgreSQL μπορούν να χρησιμοποιηθούν για την αποθήκευση των δεδομένων. Στην υλοποίηση που παρουσιάζεται, επιλέχθηκαν η γλώσσα Python και το Django Rest Framework για την ανάπτυξη του server και η SQLite ως ΒΔ.

2.4.1.1 Python

Η Python δημιουργήθηκε για πρώτη φορά το 1991 από τον Guido van Rossum [16]. Κύριο χαρακτηριστικό της αποτελεί ο ευανάγνωστος κώδικας, λόγω της απλής σύνταξης, γι αυτό και αναγνωρίζεται ως η δημοφιλέστερη γλώσσα για εκμάθηση προγραμματισμού. Η Python χρησιμοποιείται στην ανάπτυξη Διαδικτυακών εφαρμογών ως γλώσσα back-end και συνήθως συνδυάζεται με κάποια άλλη γλώσσα front-end (συχνά Javascript) για τη ανάπτυξη μιας ολοκληρωμένης εφαρμογής. Η Python επιτρέπει στους προγραμματιστές να δημιουργούν εφαρμογές συστήματος με γραφικά, παιχνίδια, βοηθητικά προγράμματα γραμμής εντολών, διαδικτυακές εφαρμογές και πολλές ακόμη. Υπάρχουν διάφοροι παράγοντες για τους

οποίους επιλέγεται η Python ως γλώσσα ανάπτυξης Διαδικτυακών εφαρμογών:

• **Εύκολη εκμάθηση της γλώσσας**: Η απλή σύνταξή της επιτρέπει ακόμη και στους αρχάριους προγραμματιστές να ενταχθούν πιο ομαλά και γρήγορα στην ομάδα ανάπτυξης του κώδικα ενός project.

• **Καλή αναγνωσιμότητα**: Το γεγονός ότι η σύνταξη της python είναι παρόμοια με την καθημερινή μας γλώσσα διευκολύνει τους προγραμματιστές στη συγγραφή κατανοητού και ευανάγνωστου κώδικα.

• **Μεγάλο εύρος βιβλιοθηκών**: Υπάρχουν πολλές βιβλιοθήκες της python που μπορουν να χρησιμοποιηθούν για την ταχύτερη ανάπτυξη μιας εφαρμογής. Αυτές οι βιβλιοθήκες είναι ανοιχτού περιεχομένου, επομένως δεν χρειάζεται να υλοποιηθούν απο το μηδέν.

• **Δημοφιλή πλαίσια ανάπτυξης**: Μερικά από τα πιο δημοφιλή πλαίσια ανάπτυξης Διαδικτυακών εφαρμογών για python αποτελούν τα Django, Flask, Pyramid, Web2Py και Turbogears. Αυτά στοχεύουν στην ταχύτερη ανάπτυξη μιας διαδικτυακής εφαρμογής και περιέχουν πακέτα κώδικα για διάφορους τομείς της (URL mapping, πρόσβαση στη ΒΔ, αιτήματα HTTP κ.λπ.).

Σε επόμενη ενότητα γίνεται αναφορά στο πιο δημοφιλές πλαίσιο ανάπτυξης της Python, το Django και, κατ΄επέκταση, το Django Rest Framework (DRF).

2.4.1.2 SQL

Η SQL (Structured Query Language) [17] είναι μια γλώσσα που αναπτύχθηκε για να λειτουργεί με σχεσιακές βάσεις δεδομένων. Πιο συγκεκριμένα, χρησιμοποιείται για τη διαχείριση των δεδομένων σε αυτές. Επιπλέον, αποτελεί εργαλείο αλληλεπίδρασης μιας ΒΔ με μια διαδικτυακή εφαρμογή. Οι δημοφιλέστερες ΒΔ που χρησιμοποιούν SQL είναι οι MySQL, PostgreSQL και SQLite. Η δομή στην οποία αποθηκεύονται τα δεδομένα έχει τη μορφή ενός πίνακα. Κάθε εγγραφή του πίνακα αποθηκεύεται σε μία γραμμή και κάθε στήλη αντιπροσωπεύει μια ιδιότητα αυτής της εγγραφής. Κάθε πίνακας πρέπει να έχει ένα πρωτεύον κλειδί (primary key) που είναι ένα μοναδικό αναγνωριστικό για κάθε οντότητα σε αυτόν τον πίνακα. Μια ΒΔ μπορεί να έχει περισσότερους από έναν πίνακες και η σύνδεση τους πραγματοποιείται με χρήση ξένων κλειδιών (foreign key).

2.4.1.3 Django Framework

Το Django [2] είναι ένα πλαίσιο ανάπτυξης της Python που χρησιμοποιείται για την ανάπτυξη ασφαλών Διαδικτυακών εφαρμογών. Το Django παρέχει εργαλεία για τη δημιουργία μιας διαδικτυακής εφαρμογής, ώστε να μην είναι απαραίτητο να χτιστεί από το μηδέν. Επίσης, είναι ανοιχτού περιεχομένου, έχει ενεργή κοινότητα και εξαιρετικό documentation. Το Django είναι κατάλληλο για την ανάπτυξη λογισμικού που είναι:

• Ολοκληρωμένο, καθώς παρέχει όλα τα απαραίτητα εργαλεία για τη δημιουργία Διαδικτυακών εφαρμογών και εκτενές documentation.

• Ευέλικτο, διότι χρησιμοποιείται για την κατασκευή οποιουδήποτε τύπου διαδικτυακής εφαρμογής (εφαρμογές wiki, κοινωνικής δικτύωσης και ειδησεογραφικές ιστοσελίδες). Επίσης, συνδυάζεται με πληθώρα από client frameworks (React.Js) προσφέροντας περιεχόμενο σχεδόν σε οποιαδήποτε μορφή (όπως HTML, JSON, κ.λπ.). Τέλος, είναι πλήρως συμβατό με πολλές ΒΔ.

• Ασφαλές. Αναλυτικότερα, το Django παρέχει έναν ασφαλή τρόπο διαχείρισης των δεδομένων του χρήστη, συμπεριλαμβανομένων τόσο λογαριασμών και κωδικών πρόσβασης όσο και των cookies, με τη χρήση ενός κλειδιού ως περιεχόμενό τους, αντί για το πραγματικό που καταχωρείται στη ΒΔ.

• Επεκτάσιμο, καθώς χρησιμοποιεί την αρχιτεκτονική ”shared-nothing” (κάθε κομμάτι είναι ανεξάρτητο από τα υπόλοιπα και μπορεί να αλλάξει εάν χρειάζεται). Αποτέλεσμα του διαχωρισμού των διαφορετικών τμημάτων αποτελεί η δυνατότητα κλιμάκωσης, σε περίπτωση αύξησης της επισκεψιμότητας.

• Συντηρήσιμο, διότι είναι βασισμένο σε πρότυπα επαναχρησιμοποίησης κώδικα. Ειδικότερα, χρησιμοποιεί την αρχή ”Don’t Repeat Yourself” (DRY) με σκοπό τη μείωση της περιττής αντιγραφής και της ποσότητας του κώδικα. Για αυτό το λόγο, οι εφαρμογές που αναπτύσσονται με το Django στοχεύουν στην επαναχρησιμοποίησή τους.

2.4.1.4 Ο τρόπος λειτουργίας του Django Framework

Ο βασικός τρόπος επικοινωνίας μεταξύ ενός web client ή ενος browser και μιας διαδικτυακής εφαρμογής βασίζεται στο πρωτόκολλο HTTP. Κατα τη λήψη ενός αιτήματος, η εφαρμογή επεξεργάζεται τις πληροφορίες που περιέχονται στη διεύθυνση URL και στα δεδομένα GET ή POST. Έπειτα, για την ικανοποίηση του αιτήματος, το Django αναλαμβάνει να διαβάσει πληροφορίες από μια ΒΔ, να γράψει σε αυτή ή να εκτελέσει την απαιτούμενη

ενέργεια. Στη συνέχεια, θα επιστραφεί μια απάντηση στον browser για τη προβολή των δεδομένων μέσω ενός HTML προτύπου (template).

Οι διαδικτυακές εφαρμογές Django αναλαμβάνουν την ομαδοποίηση του κώδικα που αντιπροσωπεύει τα βήματα της παραπάνω διαδικασίας σε διαφορετικά αρχεία. Σχήμα 2.2.:

• Διευθύνσεις URL: Ενώ είναι εφικτή η επεξεργασία και η απάντηση κάθε αιτήματος μέσω μιας μεθόδου-συνάρτησης από κάθε μεμονωμένη διεύθυνση URL, αποτελεί καλύτερη αρχή η δημιουργία μιας ξεχωριστής συνάρτησης-προβολής (view) για καλύτερη διαχείριση του αιτήματος. Μια διεύθυνση URL αντιστοιχίζει κάθε αίτημα HTTP στην κατάλληλη προβολή. Η αντιστοίχιση URL μπορεί, επίσης, να αντιστοιχίσει συμβολοσειρές ή ψηφία, που περιέχονται σε μια διεύθυνση URL, και να τα μεταβιβάσει ως δεδομένα για χρήση σε μια προβολή.

• Προβολή (View): Οι προβολές αναλαμβάνουν την επεξεργασία αιτημάτων GET, POST, PUT και DELETE επιστρέφοντας τις ανάλογες απαντήσεις. Για αυτό το λόγο, απαιτούν προσβάση στα μοντέλα (models) και αντιστοιχίζουν την απάντηση σε πρότυπα (templates).

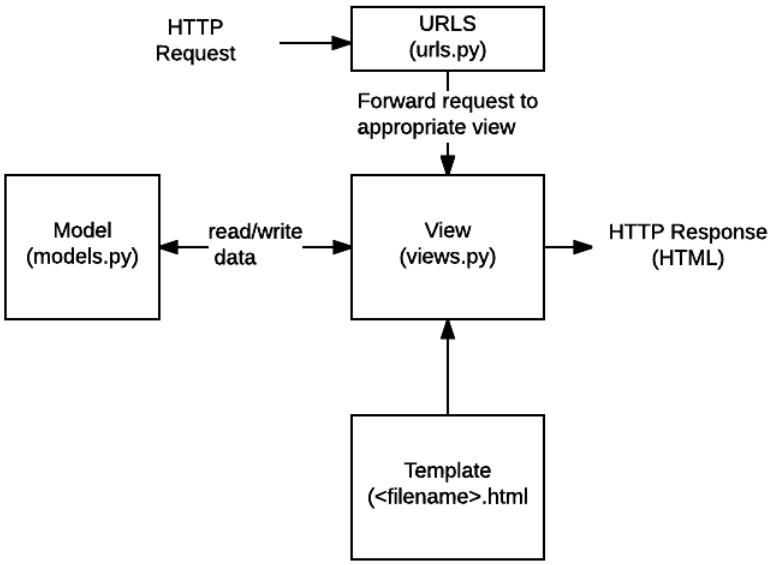
• Μοντέλο (Model): Τα μοντέλα είναι Python objects που ορίζουν τη δομή των δεδομένων μιας Django εφαρμογής, παρέχοντας εργαλεία για προσθήκη, τροποποίηση, διαγραφή και αναζήτηση εγγραφών στη ΒΔ.

• Πρότυπο (Template): Για να αναπαρασταθεί το πραγματικό περιεχόμενο ενός αρχείου/σελίδας (συνηθως HTML), χρησιμοποιούνται τα πρότυπα, δηλαδή αρχεία κειμένου για τη δομή και τη διάταξη του αρχείου. Η διαδικασία περιλαμβάνει τη συμπλήρωση ενός προτύπου HTML με τα δεδομένα ενός μοντέλου για τη δυναμική δημιουργία της σελίδας HTML. Αξίζει να σημειωθεί, ότι το πρότυπο δεν είναι απαραίτητο να είναι αρχείο HTML.

2.4.1.5 REST

Το REST (Representational State Transfer) είναι ένας όρος που παρουσιάστηκε για πρώτη φορά το 2000 από τον Roy Fielding [18] και είναι ένα σύνολο κανόνων και κατευθυντήριων γραμμών για τη δημιουργία διαδικτυακών υπηρεσιών που βασίζονται σε πόρους [19].

Όταν ένας client πραγματοποιεί ένα αίτημα μέσω ενός REST API, δημιουργείται μια αναπαράσταση του πόρου και μεταφέρεται στο endpoint. Οι πιο δημοφιλείς μορφές της αναπαράστασης είναι σε HTML και JSON. Ειδικότερα, το JSON είναι η δημοφιλέστερη μορφή αρχείου που χρησιμοποιείται, επειδή είναι ευανάγνωστη τόσο από ανθρώπους όσο και από μηχανήματα [20]. Σχήμα 2.3.



Σχήμα 2.2: Ο τρόπος λειτουργίας του Django framework [2].

2.4.1.6 Django REST Framework

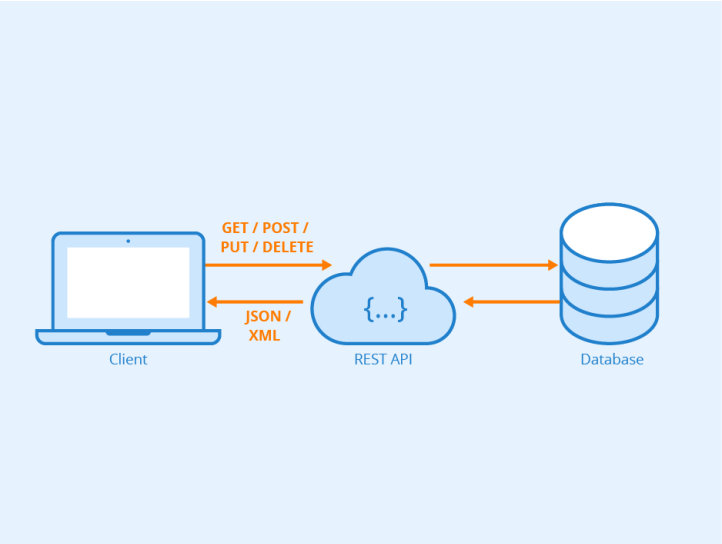
Το Django REST Framework (DRF) [21] είναι μια βιβλιοθήκη ανοιχτού περιεχομένου, βασισμένη σε Python και Django, που χρησιμεύει στη ανάπτυξη σύγχρονων Διαδικτυακών APIs. Συγκεκριμένα, είναι ένα ευέλικτο σύνολο εργαλείων με αρχιτεκτονική που καθιστά δυνατή την δημιουργία πολύπλοκων REST APIs.

Παρότι το Django REST framework περιέχει ένα ευρύ σύνολο λειτουργιών, είναι εύκολο στη χρήση. Η κύρια ιδέα πίσω από το DRF είναι η γενικευμένη αναπαράσταση των δεδομένων σε διάφορες μορφές (JSON, XML, κ.λπ.) και η ύπαρξη ένος συνόλου γενικών κλάσεων-προβολών (Class-Based-Views) για την ικανοποίηση των endpoints του API.

Ένα βασικό πλεονέκτημα του DRF αποτελεί η δημιουργία custom διευθύνσεων URL, σε αντίθεση με άλλα λιγότερο ευέλικτα πλαίσια ανάπτυξης που αυτοματοποιούν τη μετατροπή από μοντέλα Django σε endpoints. Επιπλέον, διαθέτει ενσωματωμένο πρόγραμμα περιήγησης για τη δοκιμή του API.

2.4.2 Front-End

Το front-end αναφέρεται στον προγραμματισμό της εφαρμογής από την πλευρά του client και χρησιμοποιεί συνήθως εργαλεία HTML, CSS και JavaScript. Η HTML θα δώσει οδηγίες σε ένα πρόγραμμα περιήγησης για την εμφάνιση του περιεχομένου σε μια ιστοσελίδα, ενώ η CSS διατηρεί το περιεχόμενο στη σωστή μορφή. Η JavaScript χρησιμοποιείται για τον



Σχήμα 2.3: Πραγματοποίηση ενός αιτήματος σε ενα REST API [3].

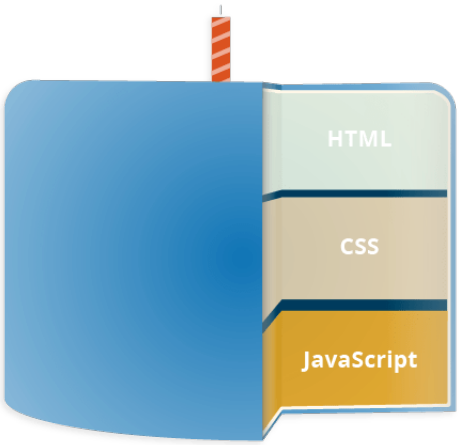
προγραμματισμό του δυναμικού μέρους μια ιστοσελίδας και την αλληλεπίδραση της με το χρήστη.

2.4.2.1 HTML5

Η HTML5 είναι μια γλώσσα για τη δομή και την παρουσίαση περιεχομένου στον Παγκόσμιο Ιστό. Είναι η τελευταία [22] σπουδαία έκδοση HTML που συστήνει το World Wide Web Consortium (W3C). Η τρέχουσα έκδοση υποστηρίζεται από την ομάδα Web Hypertext Application Technology Working Group (WHATWG). Η πρώτη δημόσια κυκλοφορία της HTML5 πραγματοποιήθηκε στις 22 Ιανουαρίου 2008 [23] και το 2014 πραγματοποιήθηκε μια σημαντική ενημέρωση.

2.4.2.2 CSS

Η Cascading Style Sheets (CSS) είναι μια γλώσσα που περιγράφει τη δομή (γραμματοσειρά, διάταξη και χρώματα) των στοιχείων ενός αρχείου HTML [24]. Με τη CSS μια ιστοσελίδα γίνεται πιο προσβάσιμη για το χρήστη. Επιπλέον, με τη χρήση ξεχωριστού αρχείου .css είναι εφικτός ο διαμοιρασμός της μορφοποίησης από διάφορες ιστοσελίδες. Ακόμη, επιτρέπει την προσωρινή αποθήκευση του αρχείου .css για τη ταχύτερη ανταπόκριση μιας σελίδας. Τελευταίο, αλλά εξίσου σημαντικό, είναι εφικτή η πρόσβαση σε μια ιστοσελίδα μέσω φορητής συσκευής, καθώς η CSS ακολουθεί κανόνες δυναμικής μορφοποίησης.



Σχήμα 2.4: Επίπεδα τεχνολογιών ιστού [4].

2.4.2.3 Javascript

Η JavaScript [4] είναι μια scripting γλώσσα προγραμματισμού, που επιτρέπει την εμφάνιση δυναμικού περιεχομένου σε ιστοσελίδες (ενημέρωση περιεχομένου, διαδραστικοί χάρτες, κινούμενα 2D/3D γραφικά, βίντεο κ.λπ). Είναι το τρίτο επίπεδο των τεχνολογιών Ιστού, δύο από τις οποίες (HTML και CSS) έχουν αναφερθεί σε προηγούμενες υποενότητες. Σχήμα 2.4.

Η JavaScript είναι μια από τις πιο δημοφιλείς γλώσσες και περιλαμβάνει πολλές δυνατότητες[25] όσον αφορά την ανάπτυξη Διαδικτυακών εφαρμογών:

• ”Dynamic typing”, που σημαίνει ότι ο τύπος μιας μεταβλητής ορίζεται με βάση την τιμή που του ανατέθηκε. Για παράδειγμα, σε μια μεταβλητή x μπορεί να αποθηκευτεί μια συμβολοσειρά ή μια τιμή τύπου αριθμού ή ένας πίνακας ή ένα αντικείμενο.

• ”Object Oriented Programming” (OOP). Δύο σημαντικές αρχές υποστήριξης OOP σε JavaScript είναι η δημιουργία αντικειμένων και η επαναχρησιμοποίηση κώδικα. Επίσης, οι συναρτήσεις στη JavaScript μπορούν να χρησιμοποιηθούν ως αντικείμενα και μπορούν να περάσουν και σε άλλες συναρτήσεις.

• “Async” λειτουργίες. Αυτές οι λειτουργίες δεν εκτελούνται σειριακά, αλλά παράλληλα, κάτι που μειώνει σημαντικά τον χρόνο επεξεργασίας.

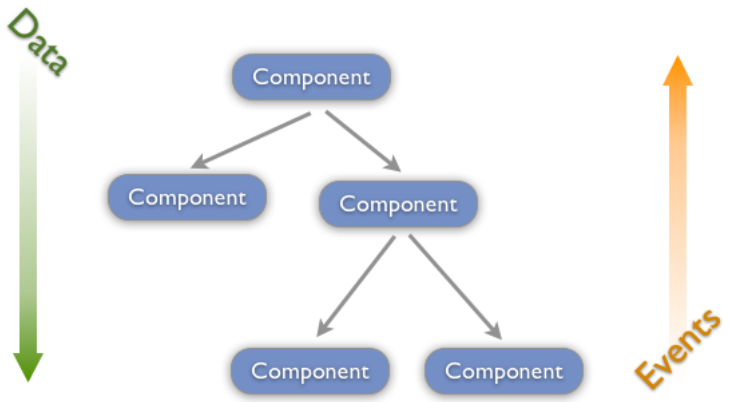
2.4.2.4 React.JS

Για τη δημιουργία της διεπαφής του χρήστη χρησιμοποιήθηκε το πλαίσιο ανάπτυξης React.JS. Το React.JS [26] είναι μια βιβλιοθήκη ανοιχτού περιεχομένου της JavaScript για τη

δημιουργία διεπαφών (interface) για τους χρήστες. Αναλυτικότερα, χρησιμοποιείται για την αναπαράσταση του περιεχομένου στο χρήστη τόσο σε διαδικτυακές εφαρμογές όσο και σε κινητές συσκευές. Το React αναπτύχθηκε για πρώτη φορά για χρήση στο Facebook το 2011 και στο Instagram το 2012. Επίσης, επιτρέπει στους προγραμματιστές να δημιουργούν διαδικτυακές εφαρμογές μεγάλης κλίμακας που μπορούν να τροποποιήσουν δεδομένα, χωρίς να απαιτείται επαναφόρτωση της σελίδας. Ο σχεδιασμός του επιτρέπει τη σύνδεση με άλλες βιβλιοθηκες ή JavaScript πλαίσια ανάπτυξης, όπως το Angular JS.

Το React, αντί να χρησιμοποιεί JavaScript για τη δημιουργία templates, χρησιμοποιεί JSX, μια προέκτασή της, που επιτρέπει τη συγγραφή κώδικα τύπου HTML. Έπειτα, η σύνταξη HTML μετατρέπεται σε κλήσεις JS. Πιο συγκεκριμένα, το JSX είναι πιο γρήγορο από την απλή JS, καθώς πραγματοποιεί βελτιστοποιήσεις κατά τη μετατροπή. Αντί να ξεχωρίζει τα κομμάτια κώδικα HTML και JS σε διαφορετικά αρχεία, το React χρησιμοποιεί τα components. Τα components είναι κομμάτια κώδικα, ανεξάρτητα και επαναχρησιμοποιήσιμα, τα οποία λειτουργούν ως συναρτήσεις JS με τη διαφορά οτι επιστρέφουν HTML.

Σε ορισμένες περιπτώσεις, τα components πρέπει να επικοινωνούν (να στέλνουν δεδομένα το ένα στο άλλο) και αυτό επιτυγχάνεται με τη χρήση των props (properties) [27]. Τα props χρησιμοποιούνται μόνο για ανάγνωση και η μετάβαση τους γίνεται προς μία κατεύθυνση (parent-component προς child-component). Ως αποτέλεσμα, τα δεδομένα που προέρχονται από τον γονέα δεν πρέπει να μεταβάλλονται από ένα παιδί. Σχήμα 2.5.



Σχήμα 2.5: Ο τρόπος επικοινωνίας των components με τη χρήση των props [5].

Το React δημιουργεί μια κρυφή μνήμη που υπολογίζει τις αλλαγές που έγιναν και, στη συνέχεια, ενημερώνει τον browser. Αυτό δίνει την ψευδαίσθηση στο χρήστη ότι ενημερώνεται ολόκληρη η σελίδα σε κάθε αλλαγή, ενώ στην πραγματικότητα ενημερώνονται μόνο τα

components που πραγματικά αλλάζουν.

Κεφάλαιο 3

Υλοποίηση

3.1 Συλλογή των Δεδομένων

Το πρωταρχικό βήμα της υλοποίησης αποτελεί η διαδικασία συλλογής των δεδομένων από το λειτουργικό. Για αυτό το σκοπό χρησιμοποιήθηκαν shell scripts με χρήση κατάλληλου parsing για την εξαγωγή όλων των επιθυμητών δεδομένων. Τα δεδομένα που συλλέχθηκαν, διακρίνονται στις παρακάτω κατηγορίες:

• Γενικές πληροφορίες για τον χρήστη. Σε αυτη την κατηγορία εντάσσονται 3 scripts. Το πρώτο συλλέγει πληροφορίες για το id του χρήστη στο σύστημα, ενώ τα επόμενα εξάγουν τις mac και ip addresses. Οι εντολές που χρησιμοποιήθηκαν για τη συλλογή των δεδομένων είναι η “whoami” και η “ip addr”, αντίστοιχα. Πίνακας 3.1. Η εντολή whoami, που χρησιμοποιείται στο λειτουργικό σύστημα Unix, εμφανίζει το username του τρέχοντος χρήστη, ενώ η ip addr εμφανίζει τις διευθύνσεις όλων των network interfaces.

User Id	whoami
Mac Address	ip addr show $(awk 'NR==3{print $1}' /proc/net/wireless | tr -d :) | awk '/ether/{print $2}'
Ip Address	ip addr | grep inet | grep wlp | awk '{print$2}' | awk -F/ '{print$1}'
Ip Address (Vm)	ip addr | grep inet | grep ens | awk '{print$2}' | awk -F/ '{print$1}'

Σχήμα 3.1: Generic scripts.

• Επεξεργαστής του συστήματος. Συγκεκριμένα, συλλέχθηκαν δεδομένα για τη χρήση και τη θερμοκρασία του επεξεργαστή, με χρήση των εντολών top και sensors. Πίνακας 3.2. Η εντολή top εμφανίζει τις συνοπτικές πληροφορίες του συστήματος και τη λίστα των διεργασιών ή των νημάτων που διαχειρίζεται αυτήν τη στιγμή ο πυρήνας του Linux. Η εντολή sensors, που ανήκει στο πακέτο lm_sensors, χρησιμοποιείται συνήθως για την παρακολού-

θηση της θερμοκρασίας του επεξεργαστή.

Usage	top -b -n 2 -d1 | grep ”Cpu(s)” | tail -n1 | awk ’{print $2}’
Temperature	sensors | grep -oP ’Package.*?\+\K[0-9.]+’

Σχήμα 3.2: Cpu scripts.

• Μνήμη και swap μνημη του συστήματος. Αναλυτικότερα, τα δεδομένα περιλαμβάνουν το μέγεθος και τη χρήση τόσο σε μνήμη όσο και σε swap μνήμη, καθώς και τη χρήση της buff/cache μνήμης. Για το σκοπό αυτό χρησιμοποιήθηκε η εντολή free, η οποία παρέχει πληροφορίες για τη RAM και τη swap μνήμη του συστήματος. Πίνακας 3.3.

Total Memory	free -m | grep ”Mem” | awk ’{print $2}’
Used Memory	free -m | grep ”Mem” | awk ’{print $3}’
Free Memory	free -m | grep ”Mem” | awk ’{print $4}’
Total Swap Memory	free -m | grep ”Swap” | awk ’{print $2}’
Used Swap Memory	free -m | grep ”Swap” | awk ’{print $3}’
Free Swap Memory	free -m | grep ”Swap” | awk ’{print $4}’
Buff/Cache Memory	free -m | grep ”Mem” | awk ’{print $6}’

Σχήμα 3.3: Memory scripts.

• Σκληρός δίσκος του συστήματος και συγκεκριμένα δεδομένα που αφορούν το μέγεθος και τη χρήση του δίσκου, χρησιμοποιώντας την εντολή df. Πίνακας 3.4.

Disk size	df -h | grep ”/dev/sda5” | awk ’{print $2}’
Disk used	df -h | grep ”/dev/sda5” | awk ’{print $3}’
Free Disk space	df -h | grep ”/dev/sda5” | awk ’{print $4}’
Disk usage	df -h | grep ”/dev/sda5” | awk ’{print $5}’

Σχήμα 3.4: Disk scripts.

• Τέλος, στην περίπτωση που είναι φορητός υπολογιστής, συλλέγονται πληροφορίες για τη μπαταρία του συστήματος (κατάσταση και ποσοστό μπαταρίας) μέσω της εντολής upower, καθώς και για το χρόνο που απομένει για την πλήρη φόρτιση, στην περίπτωση που φορτίζει, και το χρόνο που απομένει για την εξάντληση μπαταρίας, στην αντίθετη περίπτωση. Πίνακας 3.5

State	upower -i /org/freedesktop/UPower/devices/battery_BAT0 | grep "state" | awk '{print $2}'
Percentage	upower -i /org/freedesktop/UPower/devices/battery_BAT0 | grep "percentage" | awk '{print $2}' | awk -F% '{print $1}'
Time to ful	upower -i /org/freedesktop/UPower/devices/battery_BAT0 | grep "time to full" | awk '{ for (x=4; x<=5; x++) printf("%s ", $x) }'
Time to empty	upower -i /org/freedesktop/UPower/devices/battery_BAT0 | grep "time to empty" | awk '{ for (x=4; x<=5; x++) printf("%s ", $x) }'

Σχήμα 3.5: Battery scripts.

3.2 MQTT Connection

3.2.1 Producer

Το πρώτο βήμα στη κατασκευή της MQTT υποδομής είναι δημιουργία της επικοινωνίας με τον broker RabbitMQ. Για τη σύνδεση με έναν broker, απλώς προσδιορίζεται το όνομα ή η διεύθυνση IP του μαζί με την πύλη (port) και τα credentials. Μετά την πραγματοποίηση της σύνδεσης δηλώνεται η ανταλλαγή. Αυτό το βήμα είναι απαραίτητο, καθώς απαγορεύεται η δημοσίευση σε ανύπαρκτη ανταλλαγή. Τα μηνύματα θα χαθούν, εάν δεν υπάρχει ακόμη ουρά για την ανταλλαγή που δημιουργήθηκε. Εάν κανένας consumer δεν λαμβάνει μηνύματα ακόμη, μπορούν να απορριφθούν με ασφάλεια. Σχήμα 3.6

```
connection = pika.BlockingConnection(pika.ConnectionParameters(broker, port, '/', credentials))
channel = connection.channel()
channel.exchange_declare(exchange='linux_monitoring', exchange_type='fanout')
```

Σχήμα 3.6: Σύνδεση με broker και δημιουργία ανταλλαγής.

Στη συνέχεια, σε μια συνεχή επανάληψη, αρχίζει η διαδικασία συλλογής των δεδομένων, η οποία αναφέρθηκε με περισσότερες λεπτομέρειες στην προηγούμενη ενότητα. Το επόμενο βήμα είναι η αρχικοποίηση του dictionary, στο οποίο θα αποθηκευτούν και θα πακεταριστούν τα δεδομένα, ώστε να αποσταλούν στην κατάλληλη ανταλλαγή. Αναλυτικότερα, χρησιμοποιείται η συνάρτηση json.dumps(), η οποία μετατρέπει ένα αντικείμενο Python σε συμβολοσειρά json, και η συνάρτηση encode() για την κωδικοποίηση σε utf-8 πριν την αποστολή. Η αποστολή επιτυγχανεται με τη συνάρτηση "channel.basic_publish", η οποία παίρνει ως ορίσματα το όνομα της ανταλλαγής και το dictionary το οποίο δημιουργήσαμε στο προηγούμενο βήμα. Τέλος, καλείται η time.sleep με συγκεκριμένο χρόνο, που αντιπροσωπεύει τη συχνότητα της δειγματοληψίας.

3.2.2 Consumer

Για τη δημιουργία της επικοινωνίας με τον broker, επαναλαμβάνεται το πρώτο βήμα όπως αναφέρεται στον Producer. Για την επίτευξη της σύνδεσης στον broker, χρειάζεται μια νέα, άδεια ουρά. Η ουρά που δημιουργείται μπορεί να περιέχει ένα τυχαίο όνομα ή να επιλέξει ο διακομιστής ένα τυχαίο όνομα για αυτή. Αυτό πραγματοποιείται παρέχοντας κενή παράμετρο ουράς στο queue_declare. Δεύτερον, μόλις κλείσει η σύνδεση μεταξύ producer και consumer, η ουρά θα πρέπει να διαγραφεί. Υπάρχει ένα αποκλειστικό flag για αυτό το λόγο (exclusive=True). Έπειτα, πρέπει ή ανταλλαγή να στείλει μηνύματα στην ουρά. Αυτή η σχέση μεταξύ ανταλλαγής και ουράς καλείται binding. Σχήμα 3.7. Στη συνέχεια, δημιουργείται μια callback συνάρτηση που θα λαμβάνει μηνύματα από την ουρά. Αυτό επιτυγχάνεται με την μέθοδο channel.basic_consume, η οποία αντιστοιχίζει την ουρά στην callback συνάρτηση. Σχήμα 3.8. Η callback συνάρτηση, με χρήση mutex για αμοιβαίο αποκλεισμό, δημιουργεί ένα thread για την αποθήκευση των δεδομένων στη βάση (μέθοδος που θα αναλυθεί σε επόμενη ενότητα). Τελευταίο βήμα αποτελεί ο ορισμός ενός ατέρμονου βρόγχου που περιμένει για δεδομένα και εκτελεί callbacks, όποτε είναι απαραίτητο.

```
result = channel.queue_declare(queue='', exclusive=True)
queue_name = result.method.queue
channel.queue_bind(exchange='linux_monitoring', queue=result.method.queue)
```

Σχήμα 3.7: Ορισμός της ουράς και σύνδεση με τη κατάλληλη ανταλλαγή.

```
def callback(ch, method, properties, body):
    Thread(target=data_handling.save_to_db, args=(body, mutex)).start()


channel.basic_consume(queue=queue_name, on_message_callback=callback, auto_ack=True)
channel.start_consuming()
```

Σχήμα 3.8: Δημιουργία callback συνάρτησης και αντιστοίχιση στην κατάλληλη ουρά.

3.3 Σχεδιασμός της Βάσης Δεδομένων

Βασικό βήμα της υλοποίησης αποτελεί η δημιουργία της βάσης δεδομένων. Η ΒΔ παρουσιάζεται στο σχήμα 3.9. Πιο αναλυτικά, αποτελείται από τις εξής οντότητες-πίνακες:

1. User: Αντιπροσωπεύει κάθε χρήστη της εφαρμογής.

2. AppUser: Περιέχει προσωπικές πληροφορίες για κάθε χρήστη. Συμπεριλαμβάνει τα εξής πεδία:
   - account: Ξένο κλειδί στην οντότητα User.
   - first_name & last_name: Ονοματεπώνυμο χρήστη.
   - email & telephone: Email και τηλέφωνο χρήστη.

3. Cluster: Περιέχει πληροφορίες για το group που γίνεται η παρακολούθηση (π.χ data-center). Σε αυτή την οντότητα περιλαμβάνονται:
   - user: Ξένο κλειδί στην οντότητα AppUser.
   - name: Όνομα του cluster.
   - location: Τοποθεσία του cluster.
   - id: Id του cluster (αριθμός).
   - status: Boolean πεδίο για τη κατάσταση του cluster (1 - ενεργό, 0 - ανενεργό).
   - date_created: Ημερομηνία δημιουργίας του cluster.

4. Device: Αντιπροσωπεύει μία συσκευή και περιλαμβάνει τα εξής πεδία:
   - cluster: Ξένο κλειδί στην οντότητα Cluster.
   - mac_address: Διεύθυνση mac της συσκευής.
   - ip_address: Διεύθυνση ip της συσκευής.
   - name & id: Όνομα και id της συσκευής.
   - status: Boolean πεδίο για τη κατάσταση μιας συσκευής (1 - ενεργή, 0 - ανενεργή).
   - type: Τύπος συσκευής (Laptop, Desktop, Vm).
   - date_created: Ημερομηνία δημιουργίας της συσκευής.
   - updated_at: Ημερομηνία ενημέρωσης της συσκευής.

5. Option: Αντιπροσωπεύει μια επιλογή. Μία συσκευή μπορεί να έχει από μία έως τέσσερις επιλογές, ανάλογα με τις απαιτήσεις τους χρήστη και τον τύπο της συσκευής. Ένα device έχει μια επιλογή για κάθε κατηγορία (cpu, memory, disk και battery) που επιθυμεί ο χρήστης να προβάλλεται στην εφαρμογή (υπάρχει option cpu, εάν ο χρήστης επιθυμεί να λαμβάνει πληροφορίες για το cpu). Κάθε οντότητα option περιέχει τα παρακάτω πεδία:

• device: Ξένο κλειδί στην οντότητα Device

• date_created: Ημερομηνία δημιουργίας του option.

• updated_at: Ημερομηνία ενημέρωσης του option.

• type: Τύπος του option (Cpu, Memory, Disk, Battery).

• status: Περιέχει πληροφορία για το status ενός option (1 - ενεργό, 0 - ανενεργό). Πιο συγκεκριμένα, όταν ένα option με τύπο Cpu είναι ανενεργό, τότε ο χρήστης δεν δέχεται πληροφορίες για το cpu σε αυτό το device.

• notification_status: Αποτελεί ένα boolean πεδίο για την επιλογή παραλαβής ειδοποίησης για ένα option. Αναλυτικότερα, όταν ένα option με τύπο Memory έχει μηδενικό notification_status, τότε ο χρήστης δεν δέχεται ειδοποιήσεις που σχετίζονται με τη μνήμη σε αυτο το device.

• last_notification: Ημερομηνία τελευταίας ειδοποίησης για ένα option.

6. Cpu: Αποτελεί μια μέτρηση για τη cpu. Περιλαμβάνει τα πεδία:

   • option: Ξένο κλειδί στην οντότητα Option.

   • datetime: Ημερομηνία και ώρα της μέτρησης.

   • usage: Τιμή του cpu usage.

   • temperature: Τιμή της θερμοκρασίας του cpu.

7. Memory: Αποτελεί μια μέτρηση για τη μνήμη. Περιλαμβάνει τα πεδία:

   • option: Ξένο κλειδί στην οντότητα Option.

   • datetime: Ημερομηνία και ώρα της μέτρησης.

   • total: Συνολική μνήμη.

   • used: Χρήση μνήμης.

   • free: Ελεύθερη μνήμη.

   • buff_cache: Χρήσης buff/cache μνήμης.

   • total: Συνολική μνήμη swap.

   • used: Μνήμη swap που χρησιμοποιείται.

   • free: Ελεύθερη μνήμη swap.

8. Disk: Αποτελεί μια μέτρηση για το δίσκο. Περιλαμβάνει τα πεδία:

- option: Ξένο κλειδί στην οντότητα Option.
- datetime: Ημερομηνία και ώρα της μέτρησης.
- size: Μέγεθος δίσκου.
- used: Χρήση δίσκου.
- available: Ελεύθερος χώρος δίσκου.
- used_percent: Ποσοστό χρήσης δίσκου.

9. Battery: Αποτελεί μια μέτρηση για τη μπαταρία. Περιλαμβάνει τα πεδία:
   - option: Ξένο κλειδί στην οντότητα Option.
   - datetime: Ημερομηνία και ώρα της μέτρησης.
   - percentage: Ποσοστό μπαταρίας.
   - state: Κατάσταση μπαταρίας (charging, discharging, fully-charged).
   - time_to_full: Χρόνος που απομένει για πλήρη φόρτιση.
   - time_to_empty: Χρόνος που απομένει για εξάντληση μπαταρίας.

10. Notification: Αντιπροσωπεύει μια ειδοποίηση και περιλαμβάνει τα εξής πεδία:
    - app_user: Ξένο κλειδί στην οντότητα AppUser.
    - option: Ξένο κλειδί στην οντότητα Option.
    - type: Τύπος του notification (Generic για μια γενική ειδοποίηση ή Option, εάν η ειδοποίηση αναφέρεται σε κάποιο option).
    - checked: Boolean πεδίο. Η τιμή του πεδίου δείχνει αν η ειδοποίηση έχει διαβαστεί από το χρήστη(1 - ναι, 0 - οχι).
    - saved: Boolean πεδίο. Η τιμή του πεδίου δείχνει αν η ειδοποίηση έχει αποθηκευτεί από το χρήστη(1 - ναι, 0 - οχι).
    - created_at: Ημερομηνία δημιουργίας της ειδοποίησης.
    - updated_at: Ημερομηνία ενημέρωσης της ειδοποίησης.
    - body: Κείμενο της ειδοποίησης

11. Cpu Threshold: Οντότητα με τις οριακές τιμές για τη cpu.
    - option: Ξένο κλειδί στην οντότητα Option.
    - date_created: Ημερομηνία δημιουργίας του ορίου.

• usage_warning: Όριο για αποστολή warning ειδοποίησης για τη χρήση του επεξεργαστή.

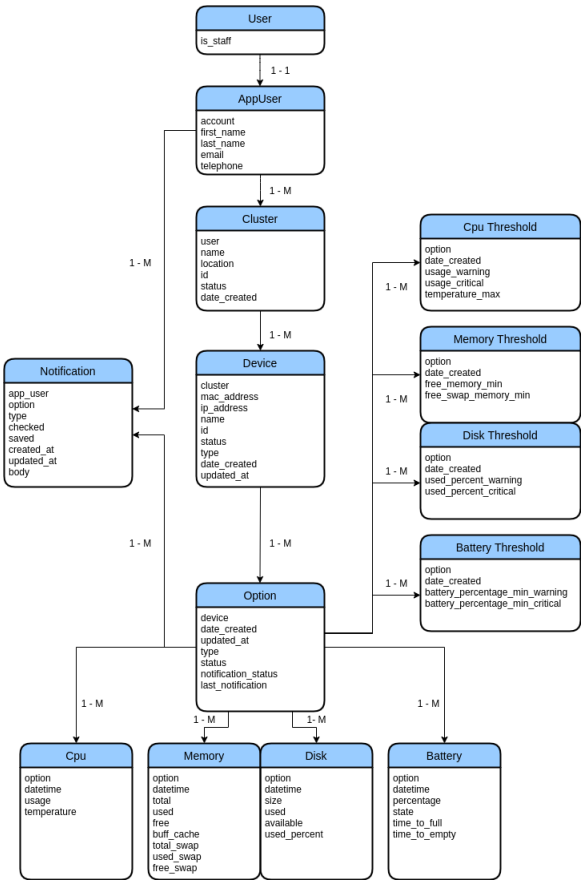
• usage_critical: Όριο για αποστολή critical ειδοποίησης για τη χρήση του επεξεργαστή.

• temperature_max: Όριο για τη θερμοκρασία του επεξεργαστή.

12. Memory Threshold: Οντότητα με τις οριακές τιμές για τη μνήμη.
    • option: Ξένο κλειδί στην οντότητα Option.
    • date_created: Ημερομηνία δημιουργίας του ορίου.
    • free_memory_min: Όριο για την ελεύθερη μνήμη.
    • free_swap_memory_min: Όριο για την ελεύθερη swap μνήμη.

13. Disk Threshold: Οντότητα με τις οριακές τιμές για το δίσκο.
    • option: Ξένο κλειδί στην οντότητα Option.
    • date_created: Ημερομηνία δημιουργίας του ορίου.
    • used_percent_warning: Όριο για αποστολή warning ειδοποίησης για τη χρήση του δίσκου.
    • used_percent_critical: Όριο για αποστολή critical ειδοποίησης για τη χρήση του δίσκου.

14. Battery Threshold: Οντότητα με τις οριακές τιμές για τη μπαταρία.
    • option: Ξένο κλειδί στην οντότητα Option.
    • date_created: Ημερομηνία δημιουργίας του ορίου.
    • battery_percentage_min_warning: Όριο για αποστολή warning ειδοποίησης για την εξάντληση της μπαταρίας.
    • battery_percentage_min_critical: Όριο για αποστολή critical ειδοποίησης για την εξάντληση της μπαταρίας.



Σχήμα 3.9: Σχεσιακό σχήμα βάσης δεδομένων.

3.4 Δημιουργία Django Rest API

Το Django Rest Framework είναι ένα διαδικτυακό πλαίσιο ανάπτυξης της Python, που παρέχει όλα τα απαραίτητα εργαλεία για τη δημιουργία σύγχρονων διαδικτυακών εφαρμογών. Το DRF χρησιμοποιήθηκε για τη δημιουργία ενός API, που μπορεί να χρησιμοποιηθεί για την προβολή δεδομένων σε οποιαδήποτε εφαρμογή από την πλευρά του client. Για το σχεδιασμό του API χρησιμοποιήθηκε μαζί με το DRF και μια SQLite βάση δεδομένων.

3.4.1 Δημιουργία και αρχικοποίηση του DRF

Μετά τη δημιουργία του Django server πραγματοποιείται η σύνδεση του με το Django Rest Framework. Το Django διαθέτει ένα αρχείο ρυθμίσεων (settings.py), όπου μπορεί να προστεθεί οποιοδήποτε ενδιάμεσο λογισμικό. Για να επιτευχθεί αυτό, αρκεί η προσθήκη του ’rest_framework’ στο κάτω μέρος του πίνακα INSTALLED_APPS στο αρχείο settings.py. Σχήμα 3.10. Επιπλέον, δημιουργήθηκε μια λίστα με web domains που επιτρέπεται να ανακτούν δεδομένα από το API. Αυτό είναι γνωστό ως Cross-Origin Resource Sharing (CORS).

Επομένως, στο πεδίο ALLOWED_HOSTS στο αρχείο settings.py προστέθηκε το κατάλληλο web domain.

```
INSTALLED_APPS=[
 ...,
 'rest_framework',
]
```

Σχήμα 3.10: Προσθήκη rest_framework στο αρχείο settings.py.

3.4.2 Δημιουργία της εφαρμογής ”api”

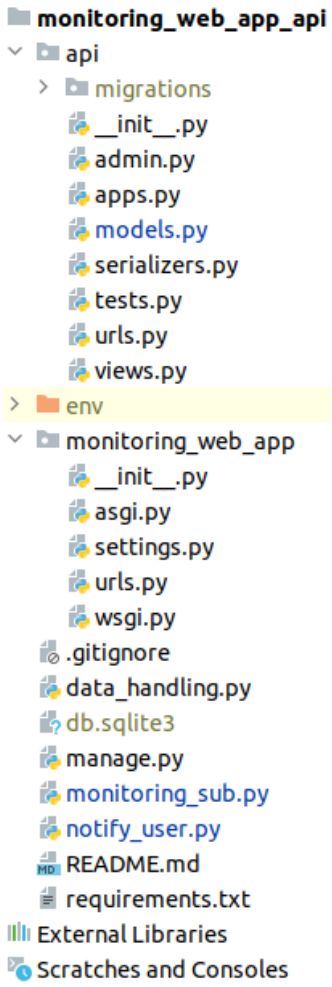
Το Django δίνει τη δυνατότητα στους προγραμματιστές για τη δημιουργία modular εφαρμογών. Οι «εφαρμογές» του Django χρησιμοποιούνται για το διαχωρισμό των λειτουργιών σε αυτόνομα πακέτα. Για το σκοπό της διπλωματικής, δημιουργήθηκε μια εφαρμογή με όνομα api. Η δομή του project, μετά τη δημιουργία της, φαίνεται στο σχήμα 3.11.

3.4.3 Models

Τα μοντέλα στο Django ορίζονται στο αρχείο models.py και είναι αυτά που επιτρέπουν τη σύνδεση με τη βάση δεδομένων. Ένα μοντέλο Django αντιπροσωπεύει τη δομή του σχήματος πίσω από τους πίνακες της ΒΔ. Στο σχήμα 3.12 φαίνεται η δημιουργία ενός μοντέλου που αντιπροσωπεύει την οντότητα συσκευής στο παραπάνω σχεσιακό σχήμα 3.9. Κατα τη διάρκεια της υλοποίησης, κάθε φορά που πραγματοποιούταν μια αλλαγή στα μοντέλα, ήταν απαραίτητη η εκτέλεση των migrations. Τα migrations είναι λειτουργίες στη βάση δεδομένων, που μετασχηματίζουν τη δομή του σχήματος και εκτελούνται μη αυτόματα. Επιπλέον, μια προσθήκη λογισμικού στο αρχείο settings.py ενδέχεται, επίσης, να χρειαστεί την εκτέλεση ενός migration βάσης δεδομένων, εάν αυτό το λογισμικό απαιτεί την αποθήκευση δεδομένων ή την ενημέρωση του σχήματος. Για την εκτέλεση ενός migration πρέπει να συνδεθεί το Django project στη ΒΔ που επιλέχθηκε (SQLite). [28].

3.4.4 Serializers

Κατά τον χειρισμό των αιτημάτων GET, χρειάστηκε ένας τρόπος μετατροπής των μοντέλων σε ένα dictionary με μορφή JSON, ώστε να σταλούν τα δεδομένα πίσω μέσω μιας απόκρισης HTTP. Ομοίως, εάν ένας χρήστης στέλνει δεδομένα στο API μέσω αιτήματος POST



Σχήμα 3.11: Δομή του Django project.

ή PUT, απαιτήθηκε ένας τρόπος επικύρωσης των δεδομένων για λόγους ακεραιότητας και ασφάλειας, προτού επιτραπεί σε αυτό το αίτημα χρήστη να προσθέσει δεδομένα στη ΒΔ. Για το σκοπό αυτό δημιουργήθηκαν οι serializers. Σχήμα 3.13. Το JSON, που δημιουργεί και στέλνει ο serializer, μπορεί να περιέχει πεδία ενός συγκεκριμένου μοντέλου (π.χ. Device), αλλά και πεδία που αντιστοιχίζονται σε συναρτήσεις που έχουν οριστεί στον serializer 3.14. Είναι σύνηθες να χρησιμοποιούνται serializers μέσω των views, τα οποία θα αναλυθούν στην επόμενη ενότητα.

3.4.5 Views

Σε αυτό το αρχείο (views.py) περιλαμβάνονται τα 'ViewSets', που χειρίζονται αιτήματα GET, POST, PUT και DELETE. Κατά τη δημιουργία ενός ViewSet ορίστηκαν οι προκαθορισμένες επιλογές για το queryset και τον serializer που θα καλέσει. Σχήμα 3.15. Σε κάθε

```python
class Device(models.Model):
    choices = (('Desktop', 'Desktop'), ('Laptop', 'Laptop'), ('Vm', 'Vm'))

    cluster = models.ForeignKey(Cluster, related_name='devices', on_delete=models.CASCADE)
    mac_address = models.CharField(max_length=20, unique=True, blank=True)
    ip_address = models.CharField(max_length=20, unique=True, blank=True)
    name = models.CharField(max_length=30, null=True)
    device_id = models.PositiveSmallIntegerField(default=0)
    status = models.BooleanField(default=True)
    type = models.CharField(max_length=10, choices=choices, null=True)
    date_created = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True, null=True)

    class Meta:
        verbose_name_plural = "Device"

    def __str__(self):
        return str(self.name) + " - " + str(self.cluster.name)

    def get_device_location(self):
        return self.cluster.location

    def get_device_owner(self):
        return str(self.cluster.user.__str__())
```

Σχήμα 3.12: Δημιουργία του Device model.

ViewSet δημιουργήθηκαν συναρτήσεις που εξυπηρετούν ένα αίτημα και εκτελούν μια συγκεκριμένη λειτουργία. Κάθε συνάρτηση μπορεί να καλέσει κάποιο διαφορετικο serializer απο τον προκαθορισμένο για το ViewSet. Στο τέλος επιστρέφει ενα Response με τα δεδομένα του serializer και ένα status επιτυχίας ή αποτυχίας του αιτήματος. Σχήμα 3.16.

3.4.6 Urls

Σημαντική είναι και η σύνδεση των views με τα τελικά σημεία διεύθυνσης URL. Σχήμα 3.20. Πρώτα δημιουργήθηκε ένα αρχείο URL μέσα στο περιβάλλον της εφαρμογής ”api” και, στη συνέχεια, συνδέθηκε το αρχείο URL της απομονωμένης εφαρμογής με το καθολικό αρχείο URL αυτού του project.

3.4.7 Ασφάλεια

Token Authentication

Token Authentication θεωρείται η μετατροπή ενός ζεύγους ονόματος χρήστη και κωδικού πρόσβασης για τη δημιουργία ενός token που θα χρησιμοποιηθεί σε όλα τα επόμενα αιτή-

```python
class DeviceDetailsSerializer(serializers.ModelSerializer):
    date_created = serializers.DateTimeField(format="%Y-%m-%d %H:%M:%S", read_only=True)
    current_cluster = serializers.SerializerMethodField(method_name='get_current_cluster')
    battery_percentage = serializers.SerializerMethodField(method_name='get_battery_percentage')
    time_to_full_or_empty = serializers.SerializerMethodField(method_name='get_time_to_full_or_empty')
    battery_state = serializers.SerializerMethodField(method_name='get_battery_state')
    options = IndexOptionSerializer(many=True, read_only=True)

    class Meta:
        model = Device
        fields = ('pk',
                  'current_cluster',
                  'name',
                  'mac_address',
                  'ip_address',
                  'device_id',
                  'status',
                  'type',
                  'date_created',
                  'updated_at',
                  'battery_percentage',
                  'time_to_full_or_empty',
                  'battery_state',
                  'options')
```

Σχήμα 3.13: Δημιουργία ενός serializer για το μοντέλο Device.

ματα, έτσι ώστε να αναγνωρίζεται ο χρήστης από την πλευρά του server. Για την ενσωμάτωση του χρειάστηκε η προσθήκη δύο πληροφοριών στο αρχείο settings.py. Πρώτα συμπεριλήφθηκε το rest_framework.authtoken στη λίστα INSTALLED_APPS και έπειτα το TokenAuthentication στο REST_FRAMEWORK. Επόμενο βήμα ήταν η προστασία κάθε τελικού σημείου του API, ώστε να εφαρμοστεί το TokenAuthentication. Αυτό έγινε προσθέτοντας σε κάθε viewset το πεδίο permission_classes = IsAuthenticated. Σχήμα 3.15

Simple JWT

To Simple JWT είναι ένα JSON Web Token authentication πρόσθετο για το Django REST Framework. Στόχος του είναι η ασφαλής χρήση των endpoints του API. Για την εφαρμογή του χρειάστηκε η προσαρμογή του Django project με την προσθήκη του rest_framework_simplejwt.authenticati στη λίστα DEFAULT_AUTHENTICATION_CLASSES, στο αρχείο settings.py. Η λογική του Simple JWT περιστρέφεται γύρω από τα refresh και access tokens, τα οποία έχουν ημερομηνίες λήξης. Το access token μπορεί να χρησιμοποιηθεί ως έλεγχος ταυτότητας για ένα protected viewset. Όταν λήξει το access token, χρησιμοποιείται το refresh token για την απόκτηση ενός νέου access token. Σχήμα 3.18.

3.4.8 Ειδοποιήσεις

Η αποστολή ειδοποιήσεων στο χρήστη είναι ένα ιδιαίτερα σημαντικό χαρακτηριστικό της εφαρμογής. Οι ειδοποιήσεις αποστέλλονται τόσο στην εφαρμογή όσο και και στο email του

```python
def get_time_to_full_or_empty(self, obj):
    try:
        if Option.objects.filter(device=obj, type='Battery').exists():
            battery_option = Option.objects.get(device=obj, type='Battery')
            if Battery.objects.filter(option=battery_option).exists():
                latest_battery_measurement = Battery.objects.filter(option=battery_option).latest('datetime')
                if latest_battery_measurement.battery_state == 'charging':
                    time_to_full_or_empty = latest_battery_measurement.battery_time_to_full
                    return time_to_full_or_empty
                elif latest_battery_measurement.battery_state == 'discharging':
                    time_to_full_or_empty = latest_battery_measurement.battery_time_to_empty
                    return time_to_full_or_empty
                else:
                    return None
            return None
        return None
    except Exception as ex:
        print(ex)
        return None
```

Σχήμα 3.14: Δημιουργία της συνάρτησης get_time_to_full_or_empty για υπολογισμό του χρόνου που απομένει για την εξάντληση ή την πλήρη φόρτιση της μπαταρίας μιας συσκευής. Το αποτέλεσμα της συνάρτησης ανατίθεται στο πεδίο time_to_full_or_empty του serializer της προηγούμενης εικόνας (Εικόνα 3.13).

```python
class AppUserViewSet(viewsets.ModelViewSet):
    queryset = AppUser.objects.all()
    serializer_class = AppUserSerializer
    permission_classes = (IsAuthenticated,)
```

Σχήμα 3.15: View για το μοντέλο AppUser. Αναλυτικότερα, φαίνονται οι προκαθορισμένες τιμές για το queryset και τον serializer.

χρήστη, για την έγκαιρη διάγνωση των πιθανών προβλημάτων. Για αυτό το λόγο, δόθηκε βάση στο απλό περιεχόμενο των ειδοποιήσεων, το οποίο περιέχει πληροφορίες για τη συσκευή, το option (Cpu, Memory, Disk, Battery), την τιμή, καθώς και τά όρια που είχε ορίσει ο χρήστης. Επίσης, θεωρήθηκε σημαντικό να μην λαμβάνει ο χρήστης συνεχώς μηνύματα. Αντιθέτως, ειδοποίηση αποστέλλεται ξανά μετά από συγκεκριμένο χρονικό διάστημα. Σχήμα 3.19.

3.5 Data Ingestion

Για την αποστολή των δεδομένων στη ΒΔ για αποθήκευση και επεξεργασία, χρησιμοποιήθηκε μια διαδικασία που ονομάζεται Data Ingestion. Πιο συγκεκριμένα, είναι μια διαδικασία με την οποία τα δεδομένα μετακινούνται από μία ή περισσότερες πηγές σε έναν προορισμό, όπου μπορούν να αποθηκευτούν και να αναλυθούν περαιτέρω. Αυτά τα δεδομένα μπορούν να προέρχονται από διάφορες πηγές, συμπεριλαμβανομένων data lakes και

```python
@action(detail=True, methods=['PUT'])
def edit_user_profile(self, request, pk=None):
    if 'first_name' in request.data or 'last_name' in request.data \
            or 'email' in request.data \
            or 'telephone' in request.data:
        user = User.objects.get(username=request.user)
        # Get Authenticated App User
        authenticated_app_user = AppUser.objects.get(account=user)
        try:
            if not int(pk):
                response = {'message': 'Wrong user pk!'}
                return Response(response, status=status.HTTP_400_BAD_REQUEST)
        except:
            response = {'message': 'Wrong user pk!'}
            return Response(response, status=status.HTTP_400_BAD_REQUEST)
        # Get url pk App User
        current_app_user = get_object_or_404(AppUser, pk=pk)
        if authenticated_app_user != current_app_user:
            response = {'message': 'No access allowed'}
            return Response(response, status=status.HTTP_400_BAD_REQUEST)
        else:
            if 'first_name' in request.data:
                authenticated_app_user.first_name = request.data['first_name']
            if 'last_name' in request.data:
                authenticated_app_user.last_name = request.data['last_name']
            if 'e_mail' in request.data:
                authenticated_app_user.email = request.data['email']
            if 'telephone' in request.data:
                authenticated_app_user.telephone = request.data['telephone']
            authenticated_app_user.save()
            serializer = AppUserSerializer(authenticated_app_user, many=False)
            response = {'message': 'User profile updated', 'result': serializer.data}
            return Response(response, status=status.HTTP_200_OK)
    else:
        response = {'message': 'You need to provide at least one field'}
        return Response(response, status=status.HTTP_400_BAD_REQUEST)
```

Σχήμα 3.16: Συνάρτηση του AppUserViewSet για την ενημέρωση του προφίλ ενός χρήστη.

συσκευών IoT, και να καταλήγουν σε διαφορετικά περιβάλλοντα, όπως αποθήκες δεδομένων cloud [29].

Η διαδικασία αυτή έχει ως σημείο εκκίνησης τον Consumer. Πιο λεπτομερώς, όπως αναφέρθηκε σε προηγούμενη ενότητα, κάθε φορά που ο Consumer λαμβάνει δεδομένα από την ουρά, καλεί μια callback συνάρτηση. Αυτή η συνάρτηση, με τη χρήση ενός mutex, δημιουργεί ένα thread για την εκτέλεση της μεθόδου save_to_db του αρχείου data_handling.py. Η μέθοδος επικοινωνεί με τον server και αναλαμβάνει την αποθήκευση των δεδομένων στη βάση. Αναλυτικότερα, η διαδικασία ξεκινάει με την αποκωδικοποίηση του json που λήφθηκε από την ουρά. Έπειτα, ελέγχεται αν στη βάση δεδομένων υπάρχει η συσκευή για την οποία προορίζονται τα δεδομένα. Στη συνέχεια, και εφόσον υπάρχουν options για τη συγκεκριμένη συσκευή, δημιουργούνται οι ανάλογες μετρήσεις για κάθε option (Cpu, Memory, Disk, Battery) με τις κατάλληλες τιμές. Τέλος, με χρήση αμοιβαίου αποκλεισμού, αρχίζει η διαδικασία notifiy_user για την ειδοποίηση του χρήστη, εφόσον ξεπεράστηκε κάποιο όριο.

```
router = routers.DefaultRouter()
router.register('users', UserViewSet)
router.register('app_users', AppUserViewSet)
router.register('clusters', ClusterViewSet)
router.register('devices', DeviceViewSet)
router.register('options', OptionViewSet)
router.register('cpu_thresholds', CpuThresholdsViewSet)
router.register('memory_thresholds', MemoryThresholdsViewSet)
router.register('disk_thresholds', DiskThresholdsViewSet)
router.register('battery_thresholds', BatteryThresholdsViewSet)
router.register('cpu_measurements', CpuViewSet)
router.register('memory_measurements', MemoryViewSet)
router.register('disk_measurements', DiskViewSet)
router.register('battery_measurements', BatteryViewSet)
router.register('notifications', NotificationViewSet)


urlpatterns = [
    path('', include(router.urls)),
    path('change/password/<int:pk>/',
         ChangePasswordView.as_view(),
         name='auth_change_password'),
]
```

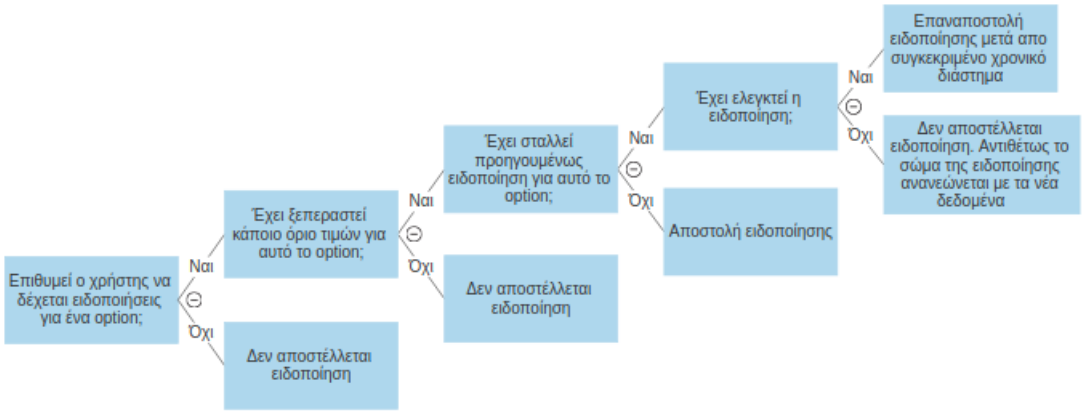
Σχήμα 3.17: Urls της εφαρμογής api.

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('api.urls')),
    path('auth/', obtain_auth_token),
    path('token/', TokenObtainPairView.as_view(), name='token_obtain_pair'),
    path('token/refresh/', TokenRefreshView.as_view(), name='token_refresh'),
]
```

Σχήμα 3.18: Urls του project.

3.6 Web Client

Για την ολοκλήρωση της υλοποίησης, δημιουργήθηκε ο web client, ένα γραφικό περιβάλλον για την απεικόνιση των δεδομένων στον χρήστη. Με αυτό τον τρόπο ο χρήστης μπορεί να παρακολουθεί όλες τις συσκευές του σε πραγματικό χρόνο, έχοντας πλήρη εποπτεία. Επιπλέον, του παρέχεται η δυνατότητα για προβολή ιστορικού μιας συσκευής και εξαγωγή των δεδομένων. Για την υλοποίηση της client εφαρμογής χρησιμοποιήθηκε το εργαλείο React, το οποίο αναφέρθηκε αναλυτικότερα στο κεφάλαιο 2, μαζί με τη βιβλιοθήκη Ant Design για την πρόσβαση σε πληθώρα βασικών components, όπως Cards, Tables και Charts.



Σχήμα 3.19: Decision tree για την αποστολή ειδοποίησης στο χρήστη.

Δυνατότητες της εφαρμογής

Οι δυνατότητες που προσφέρει η εφαρμογή του χρήστη είναι οι εξής:

• Γενικές πληροφορίες για κάθε χρήστη. Πιο συγκεκριμένα, πληροφορίες για το είδος και των αριθμό των συσκευών που είναι συνδεδεμένες.

• Πιο εξειδικευμένες πληροφορίες για κάθε ξεχωριστή συσκευή, που αφορούν τα options Cpu, Memory, Disk και Battery.

• Προβολή μετρήσεων για κάθε option.

• Προβολή διαδραστικών διαγραμμάτων για κάθε option, βασισμένα στο χρονικό διάστημα που επιλέγει ο χρήστης.

• Δημιουργία ορίων για κάθε option. Αν ξεπεραστεί κάποιο όριο, ο χρήστης θα λάβει ειδοποίηση τόσο στην εφαρμογή όσο και στο email του.

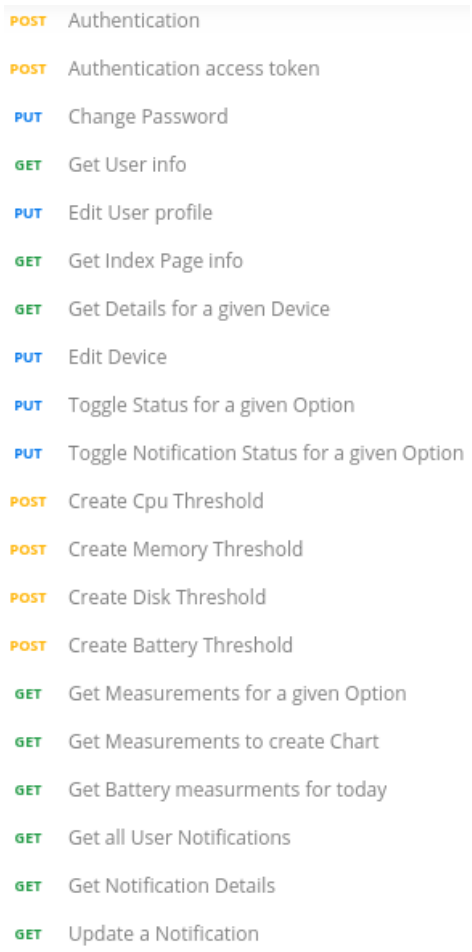
• Ξεχωριστή σελίδα ειδοποιήσεων για τη διαχείριση τους.

• Επεξεργασία του προφίλ του χρήστη.

• Σύνδεση και αποσύνδεση από την εφαρμογή με χρήση username και password.

• Αλλαγή κωδικού πρόσβασης.

• Επιλογή θέματος εμφάνισης (light, dark).



Σχήμα 3.20: Τα endpoints του API.

Κεφάλαιο 4

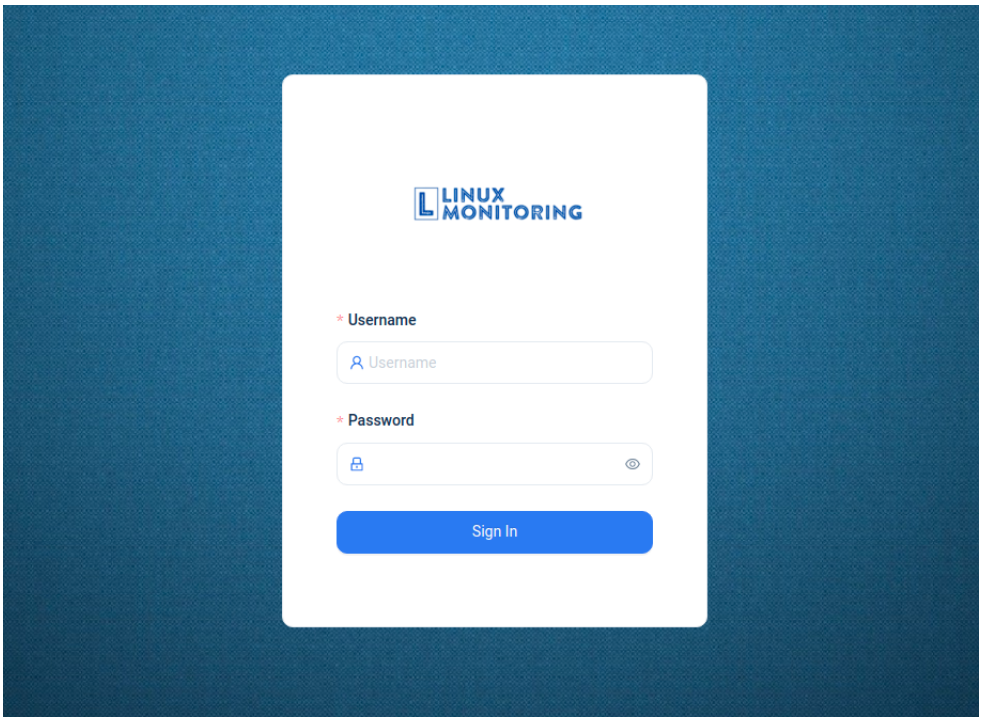
Γραφικό Περιβάλλον

Για την καλύτερη απεικόνιση των δυνατοτήτων της εφαρμογής, είναι σημαντική η παρουσίαση του γραφικού περιβάλλοντος του client, οταν ο χρήστης χρησιμοποιεί την εφαρμογή μέσω του browser. Πιο αναλυτικά, οι οθόνες που περιλαμβάνει ο client είναι οι εξής:

1. Login Page για την είσοδο του χρήστη στην εφαρμογή πληκτρολογώντας username και password. Όσο είναι συνδεδεμένος, έχει πρόσβαση από το μενού πλοήγησης στην επιλογή logout για έξοδο από την εφαρμογή. Σχήμα 4.1.

2. Μενού πλοήγησης στο πάνω-δεξιά μέρος της εφαρμογής. Το μενού πλοήγησης περιλαμβάνει επιλογές για μεταφορά στη σελίδα ειδοποιήσεων και στις ρυθμίσεις της εφαρμογής, αλλαγής θέματος από light σε dark και αποσύνδεσης από την εφαρμογή. Σχήμα 4.2.

3. Αρχική σελίδα για προβολή γενικών πληροφοριών για το χρήστη. Σχήμα 4.3. Αναλυτικότερα, το πάνω μέρος περιέχει πληροφορίες για το πλήθος των συσκευών και την κατηγορία στην οποία ανήκουν (Desktop, Laptop, Vm), καθώς και για τον αριθμό των clusters και των καθημερινών ειδοποιήσεων που δεν έχουν ελεγθεί απο τον χρήστη. Στο κάτω μέρος υπάρχουν πίνακες των Clusters με τις συσκευές που ανήκουν στο καθένα. Για κάθε συσκευή αναγράφονται πληροφορίες για το όνομά της, το είδος και το status της, καθώς και την ημερομηνία και την ώρα της τελευταίας μέτρησης. Στην τελευταία στήλη του πίνακα υπάρχει ένα κουμπί για προβολή των τελευταίων μετρήσεων μια συσκευής. Πατώντας το, ανοίγει ένα παράθυρο στο οποίο φαίνονται περιληπτικά



Σχήμα 4.1: Login Page.

τα δεδομένα. Σχήμα 4.4

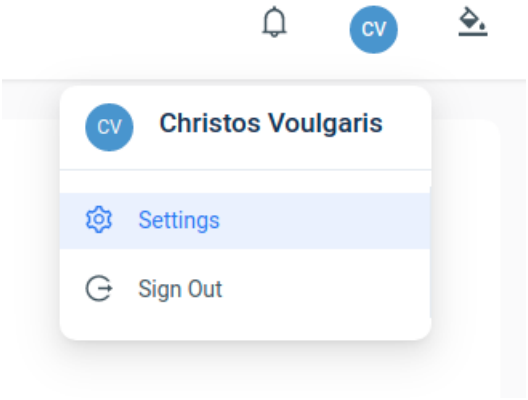
4. Σελίδα για προβολή λεπτομερειών για μια συσκευή. Σχήμα 4.5. Συγκεκριμένα, περιλαμβάνει:

   • Μία κάρτα με γενικές πληροφορίες για τη συσκευή.

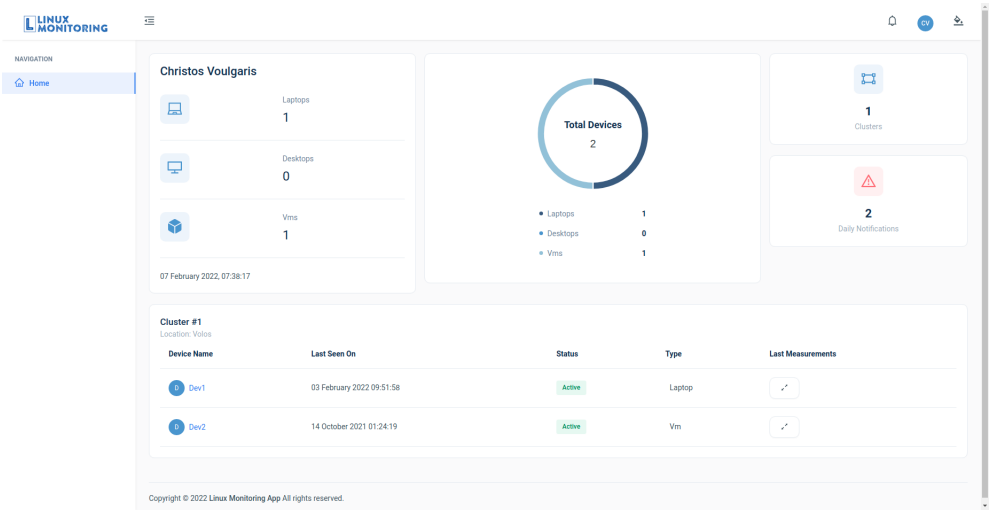
   • Έναν πίνακα για τη διαχείριση των options της συσκευής. Ο χρήστης έχει τη δυνατότητα να αλλάξει το status για παύση προβολής πληροφοριών και το notification status για παύση λήψης ειδοποιήσεων για ένα option. Στη τελευταία στήλη του πίνακα, πατώντας το κουμπί, ανοίγει ένα παράθυρο δίνοντας τη δυνατότητα δημιουργίας νέου ορίου για ένα option. Σχήμα 4.6.

   • Γραφήματα για τη κατάσταση και το ποσοστό της μπαταρίας της συσκευής.

   • Πίνακας με τα options της συσκευής. Σχήμα 4.7. Ειδικότερα, αναγράφονται ο τύπος του option, η ημερομηνία και ώρα της τελευταίας μέτρησης και ένα κουμπί για λεπτομερή προβολή των μετρήσεων ενός option. Πατώντας το κουμπί, ανοίγει ένα παράθυρο που περιέχει ένα πίνακα με όλες τις μετρήσεις για το συγκεκριμένο option. Σχήμα 4.8. Στον πίνακα δίνεται η δυνατότητα για αλλαγή του αριθμού των μετρήσεων που περιέχει κάθε σελίδα, καθώς και λήψη των μετρήσεων σε μορφή .csv.



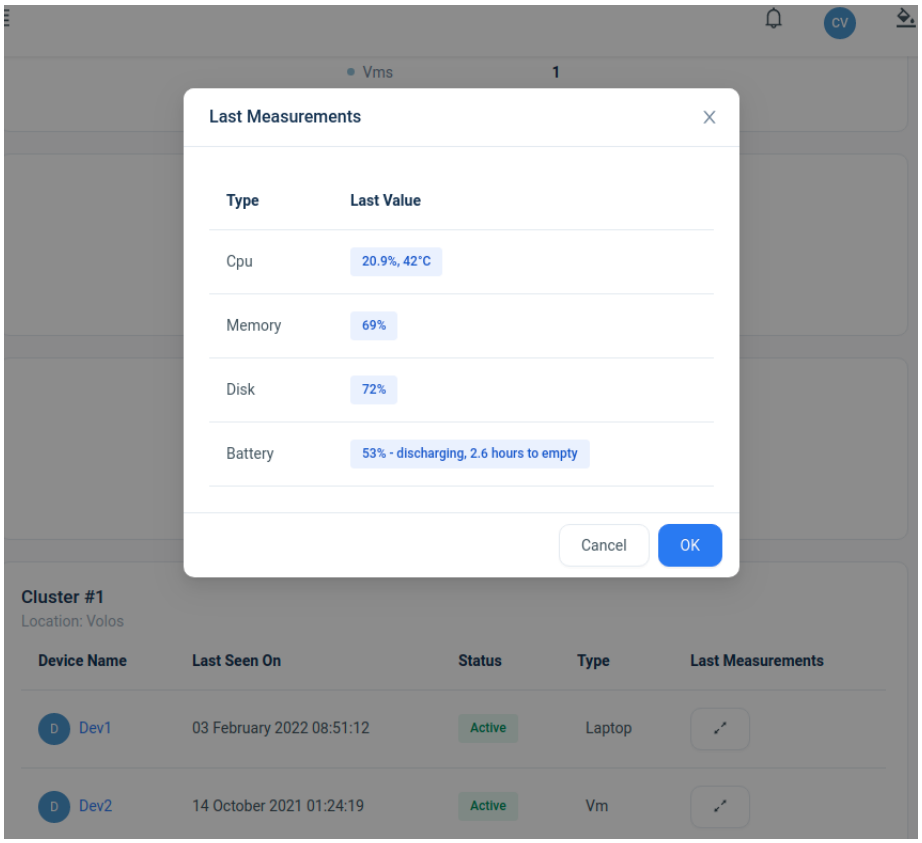
Σχήμα 4.2: Μενού πλοήγησης.



Σχήμα 4.3: Αρχική σελίδα της εφαρμογής.

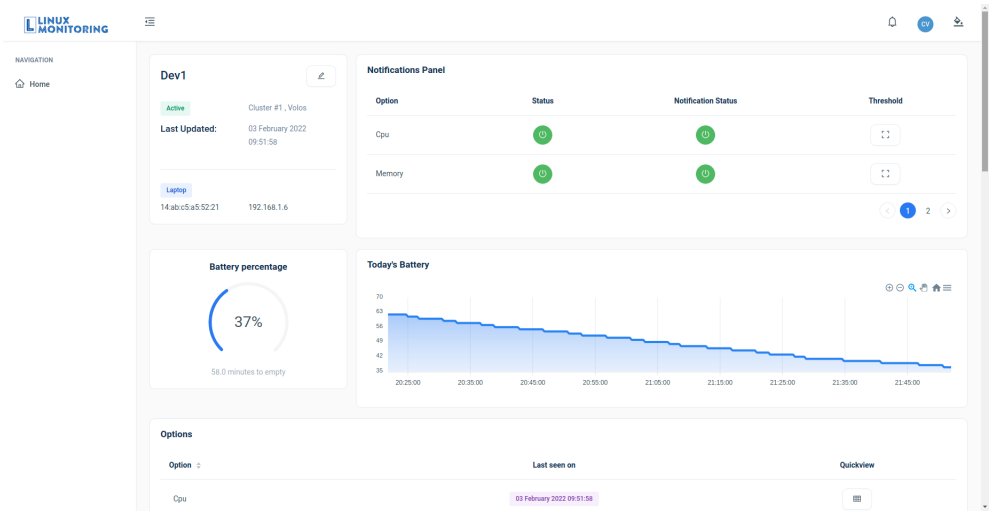
• Γραφήματα για τα υπόλοιπα options της συσκευής (Cpu, Memory, Disk) με δυνατότητα προβολής των μετρήσεων, επιλέγοντας το χρονικό διάστημα που επιθυμεί. Επίσης, ο χρήστης μπορεί ανα πάσα στιγμή να μεγεθύνει το γράφημα και να πραγματοποιήσει εξαγωγή των μετρήσεων για το χρονικό διάστημα που αποτυπώνεται στο γράφημα σε μορφή .csv ή λήψη σε εικόνα μορφής .png . Σχήμα 4.9

5. Σελίδα προβολής ειδοποιήσεων. Πατώντας στο καμπανάκι στο πάνω-δεξιά μέρος της εφαρμογής ο χρήστης μεταφέρεται στη σελίδα ειδοποιήσεων, όπου απεικονίζονται όλες οι ειδοποιήσεις που έχει λάβει. Για κάθε ειδοποίηση υπάρχουν πληροφορίες για το option, τη συσκευή και το cluster στο οποίο αναφέρεται. Επιπλέον, απεικονίζεται το περιεχόμενο και η ημερομηνία και η ώρα της ειδοποίησης. Ο χρήστης έχει τη δυνατότητα να δεί πιο λεπτομερώς μια ειδοποίηση, να την αποθηκεύσει ή να τη μετακινήσει στη λίστα των ειδοποιήσεων που έχουν ελεγχθεί.

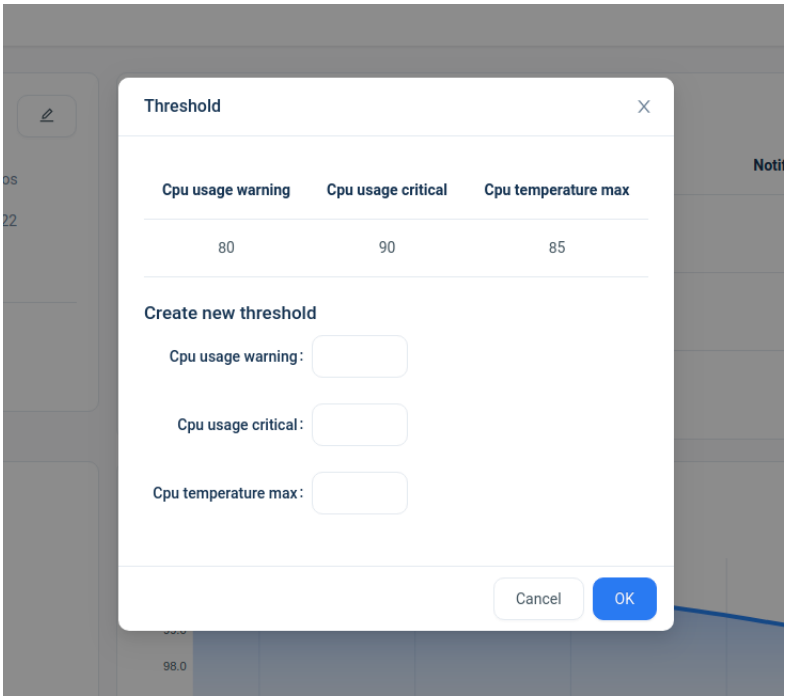


Σχήμα 4.4: Παράθυρο με τις τελευταίες μετρήσεις μιας συσκευής.

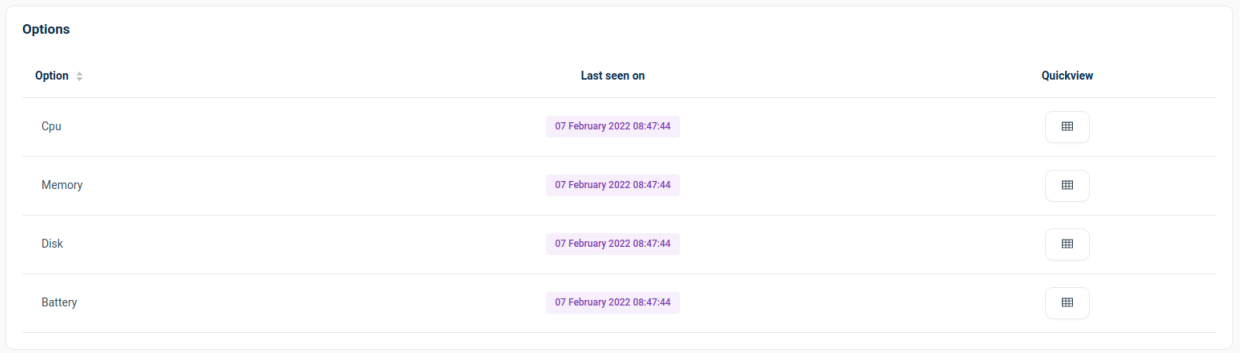
6. Ρυθμίσεις της εφαρμογής. Συγκεκριμένα, δίνεται η δυνατότητα επεξεργασίας του προφίλ του χρήστη (Σχήμα 4.12) και αλλαγής κωδικού πρόσβασης (Σχήμα 4.13).



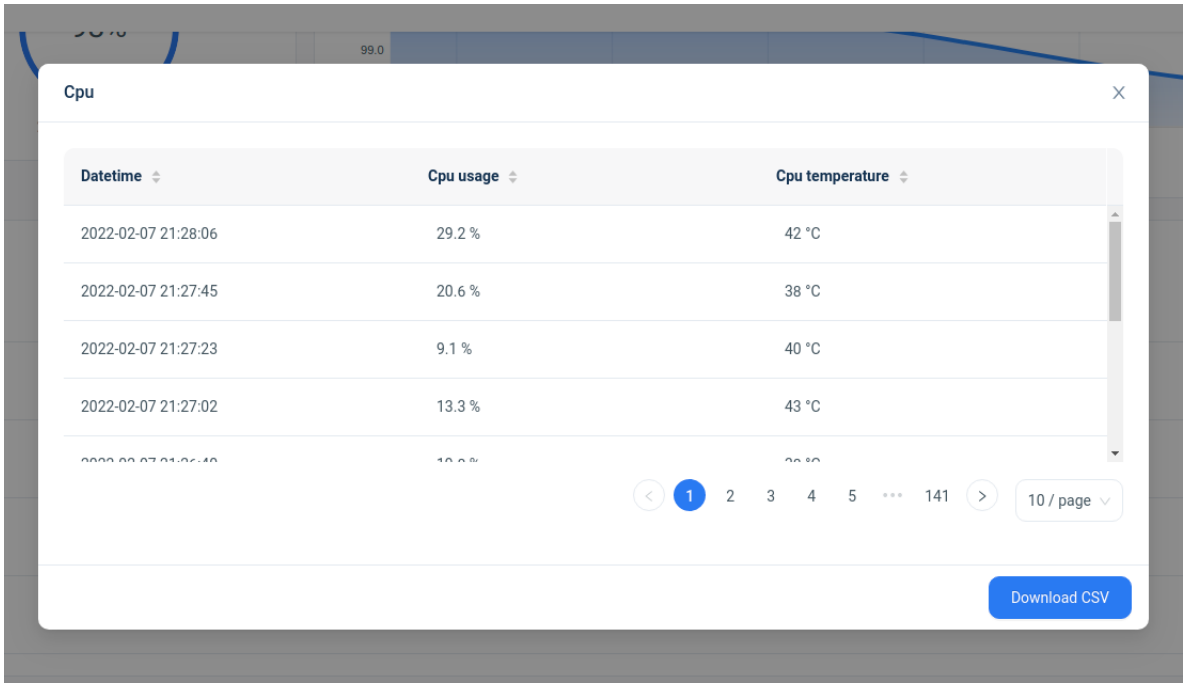
Σχήμα 4.5: Σελίδα με τις λεπτομέρειες μίας συσκευής.



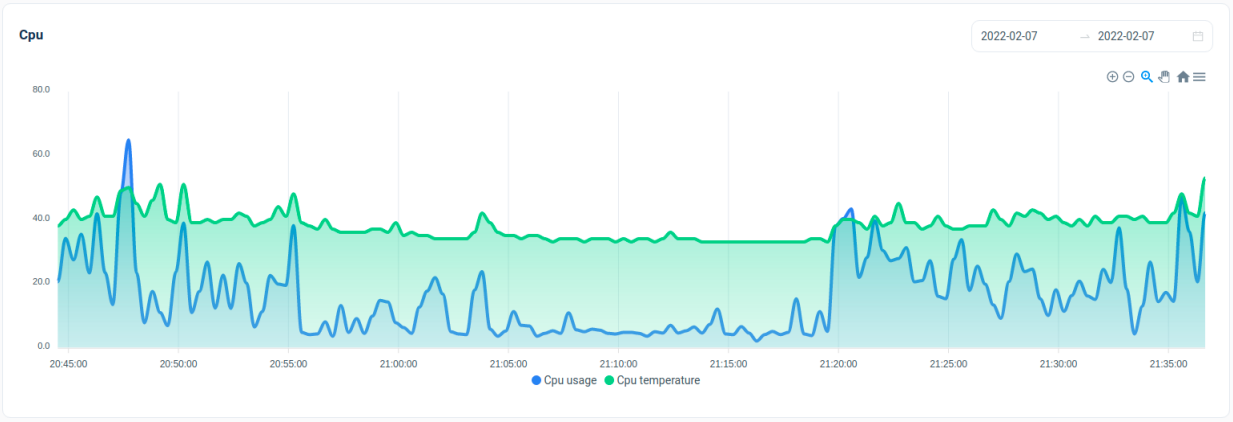
Σχήμα 4.6: Παράθυρο για τη δημιουργία νέου ορίου για το option Cpu.



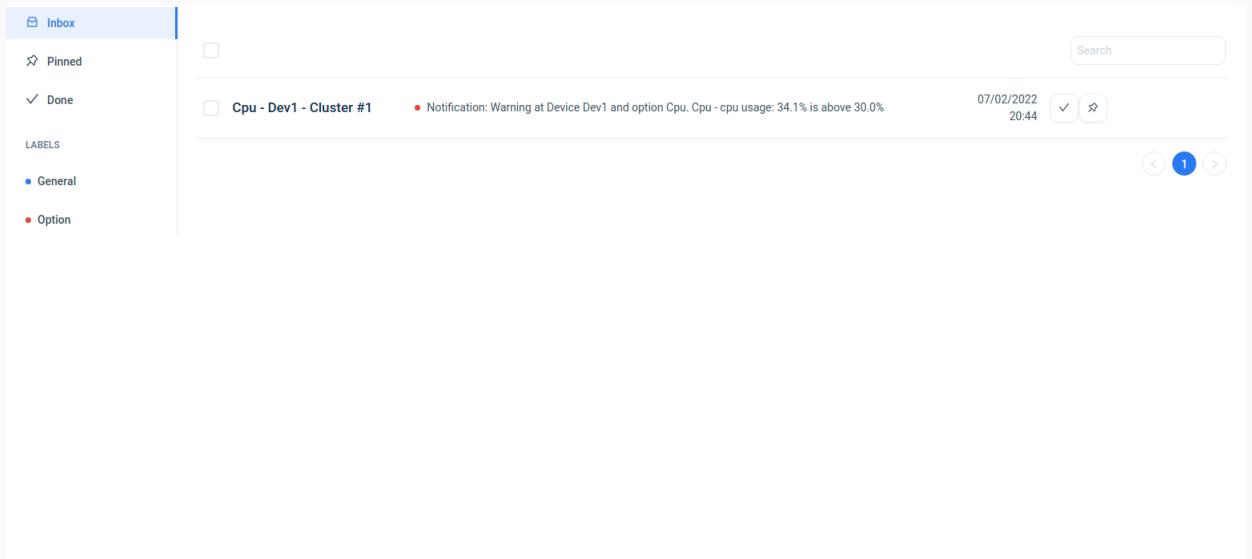
Σχήμα 4.7: Πίνακας με τα options μια συσκευής.



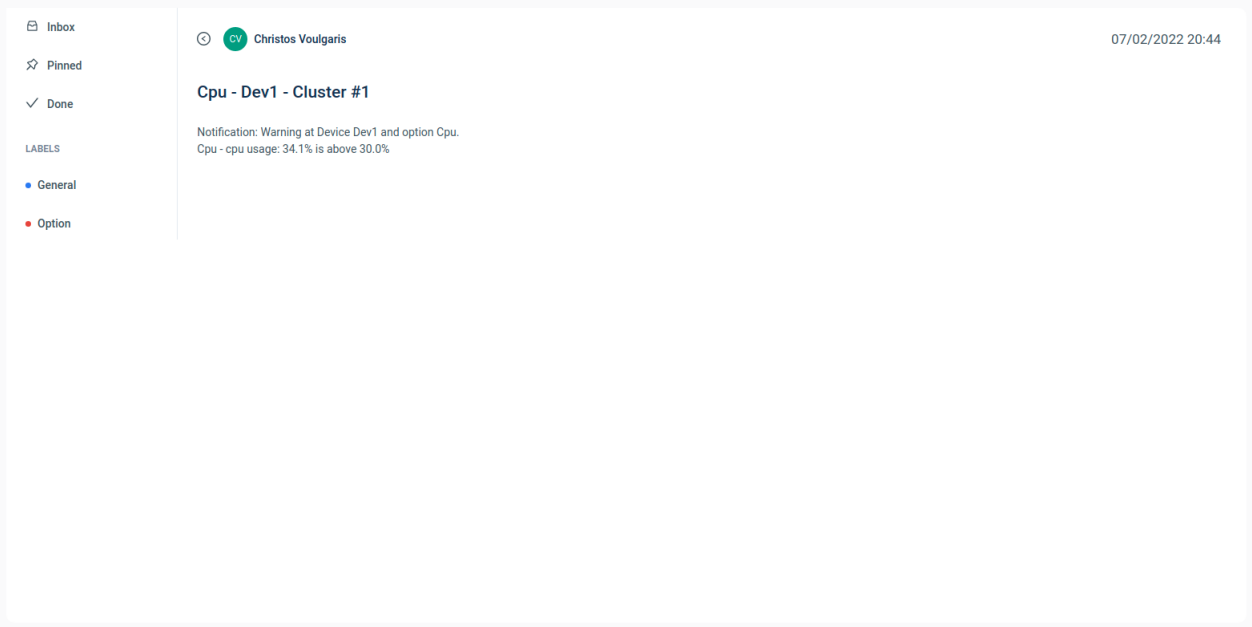
Σχήμα 4.8: Πίνακας με όλες τις μετρήσεις ενός Cpu option.



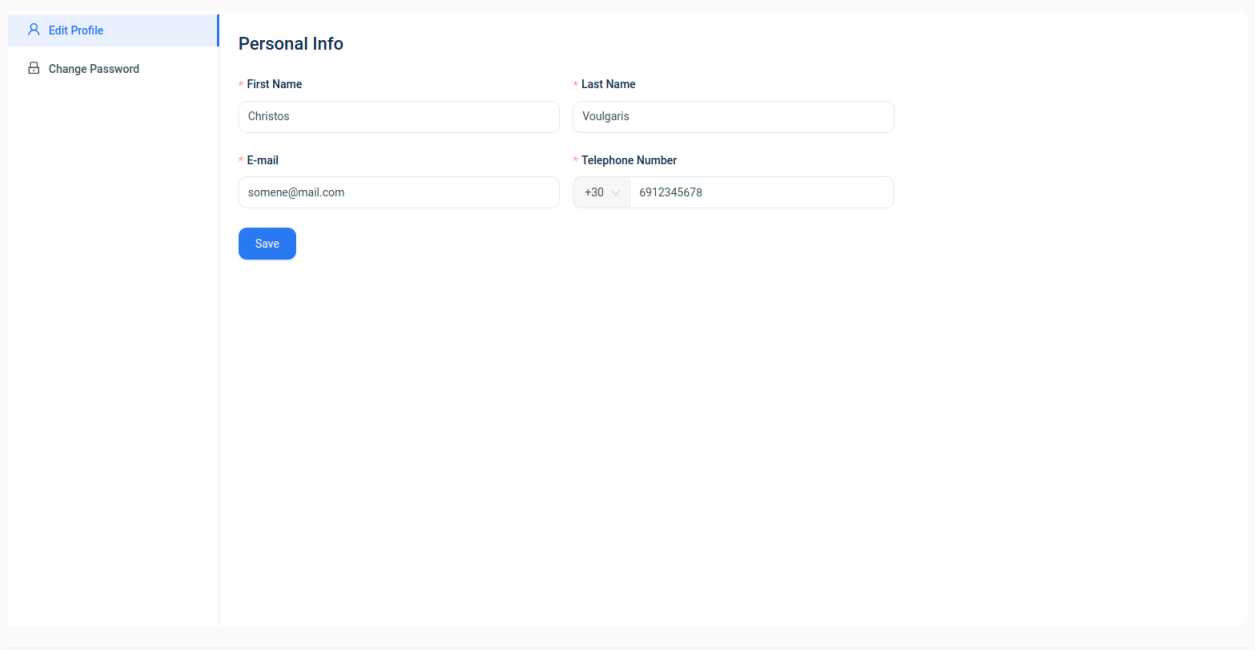
Σχήμα 4.9: Γράφημα ενός Cpu option. Στο γράφημα απεικονίζονται οι τιμές Usage και Temperature του επεξεργαστή σε σχέση με το χρόνο. Το γράφημα είναι δυναμικό, αφού ο χρήστης έχει τη δυνατότητα να επιλέξει στο κάτω μέρος του γραφήματος όποια μετρική επιθυμεί να απεικονίζεται (Cpu Usage, Cpu Temperature), καθώς και το χρονικό διάστημα.



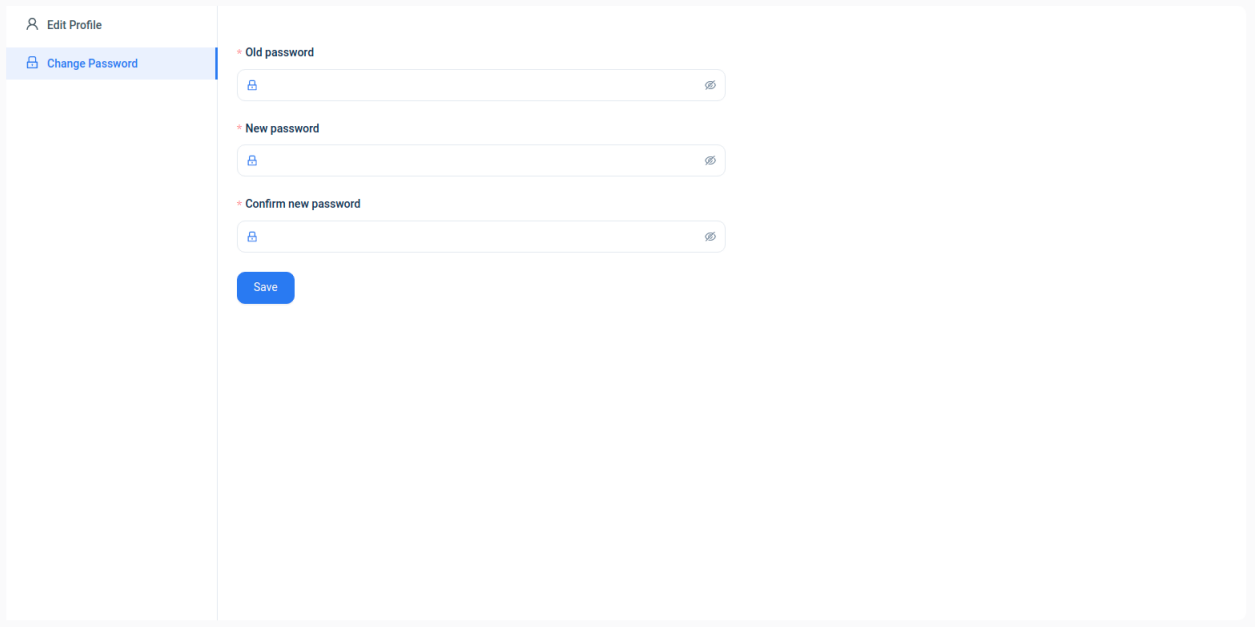
Σχήμα 4.10: Σελίδα ειδοποιήσεων.



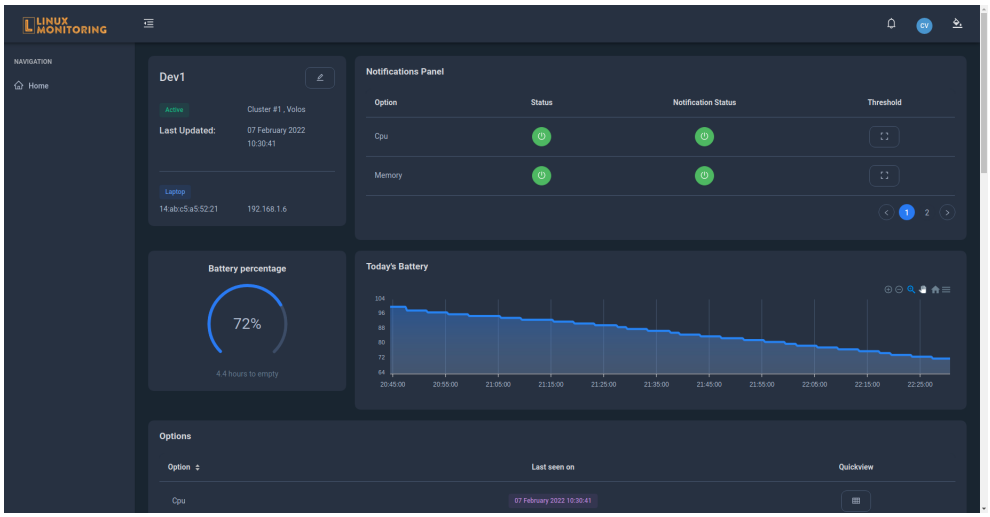
Σχήμα 4.11: Λεπτομέρειες μιας ειδοποίησης.



Σχήμα 4.12: Επεξεργασία προφίλ του χρήστη.



Σχήμα 4.13: Αλλαγή κωδικού πρόσβασης του χρήστη.



Σχήμα 4.14: Προβολή της σελίδας που περιέχει λεπτομέρειες για μία συσκευή σε σκούρο θέμα.

Κεφάλαιο 5

Συμπεράσματα

5.1 Σύνοψη και συμπεράσματα

Με την ολοκλήρωση της διπλωματικής, τέθηκε σε λειτουργία η εφαρμογή, η οποία παρέχει στο χρήστη τη δυνατότητα παρακολούθησης των υπολογιστικών συστημάτων που επιθυμεί. Λόγω της εγκυρότητας των μετρήσεων, καθώς αντιπροσωπεύουν τα πραγματικά δεδομένα ενός συστήματος, η εφαρμογή θα μπορούσε να χρησιμοποιηθεί για την προστασία του υλικού πολλών επιχειρήσεων. Όσον αφορά την υλοποίηση χρησιμοποιήθηκαν έτοιμες λύσεις που προσφέρονται στην αγορά. Αντιθέτως, η κατασκευή του συστήματος έγινε από το μηδέν και αποκλειστικά με χρήση εργαλείων ανοικτού περιεχομένου. Προϋποθέσεις για την ολοκλήρωση της υλοποίησης αποτελούν οι καλές γνώσεις προγραμματισμού και ειδικότερα των πλαισίων ανάπτυξης και των εργαλείων που χρησιμοποιήθηκαν.

Με την ολοκλήρωσή της αποδεικνύεται ότι με μηδενικό κόστος, εξαιτίας της αποκλειστικής χρήσης εργαλείων και πλαισίων ανάπτυξης ανοικτού περιεχομένου, η εφαρμογή εκπληρώνει όλους τους αρχικούς στόχους που τέθηκαν. Αναλυτικότερα, ο χρήστης έχει τη δυνατότητα της άμεσης παρακολούθησης των συστημάτων με γνώμονα την απλή εμπειρία που του προσφέρεται.

5.2 Μελλοντικές επεκτάσεις

Ως μελλοντική επέκταση, θα ήταν δυνατή η δημιουργία μιας εφαρμογής για την παρακολούθηση των συστημάτων ενός χρήστη απο μια κινητή συσκευή. Η ανάπτυξη θα πραγματοποιούνταν με τη χρήση του πλαισίου ανάπτυξης ”React Native”, το οποίο βασίζεται στο

React JS και ειδικεύεται στη δημιουργία ολοκληρωμένων και παραμετροποιήσιμων εφαρμογών για χρήση από κινητές συσκευές.

Βιβλιογραφία

[1] Rabbitmq. https://stackoverflow.com/questions/51838776/rabbitmq-is-it-possible-to-duplicate-some-messages-within-rabbitmq. Accessed: 2021-11-28.

[2] Django introduction. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction. Accessed: 2021-12-7.

[3] Rest api definition: What is a rest api integration (restful api)? https://www.astera.com/type/blog/rest-api-definition/. Accessed: 2021-12-8.

[4] What is javascript? https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. Accessed: 2021-12-14.

[5] React.js data flow. https://medium.embengineering.com/in-react-js-data-flows-in-one-direction-from-parent-to-child-841103ed3aed. Accessed: 2021-12-19.

[6] What is it (information technology). https://www.acilearning.com/blog/what-is-it-information-technology. Accessed: 2021-11-9.

[7] Information technology risk (it risk). https://cio-wiki.org/wiki/Information_Technology_Risk_(IT_Risk). Accessed: 2021-11-9.

[8] The true impact of it failures. https://www.codestone.net/our-thoughts/true-impact-of-it-failures/. Accessed: 2021-11-9.

[9] What is workstation monitoring? https://secur01.com/what-is-workstation-monitoring/. Accessed: 2021-11-20.

[10] What is workstation monitoring? https://microage.ca/nwd/en/what-is-workstation-monitoring/. Accessed: 2021-11-20.

[11] What is shell script? https://www.tutorialspoint.com/what-is-shell-script. Accessed: 2021-11-25.

[12] What is mqtt? https://www.opc-router.com/what-is-mqtt/. Accessed: 2021-11-26.

[13] Publish and subscribe - mqtt essentials. https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/. Accessed: 2021-10-27.

[14] Rabbitmq features. https://www.rabbitmq.com/#features. Accessed: 2022-01-25.

[15] Web application development. https://searchcloudcomputing.techtarget.com/definition/web-application-development. Accessed: 2021-12-1.

[16] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[17] Sql for web developers. https://medium.com/swlh/sql-for-web-developers-part-i-e8b6c32f1a04. Accessed: 2021-12-3.

[18] Roy Thomas Fielding and Richard N. Taylor. *Architectural Styles and the Design of Network-Based Software Architectures*. PhD thesis, 2000. AAI9980887.

[19] What is a rest api? https://www.contentful.com/blog/2021/10/04/what-is-a-rest-api/. Accessed: 2021-12-8.

[20] What is a rest api? https://www.redhat.com/en/topics/api/what-is-a-rest-api. Accessed: 2021-12-8.

[21] Django rest framework. https://quintagroup.com/cms/python/django-rest-framework. Accessed: 2021-12-10.

[22] Browser vendors win war with w3c over html and dom standards. https://web.archive.org/web/20190529021959/https://www.zdnet.

com/article/browser-vendors-win-war-with-w3c-over-html-and-dom-standards/. Accessed: 2021-12-12.

[23] A vocabulary and associated apis for html and xhtml. https://www.w3.org/TR/2008/WD-html5-20080122/. Accessed: 2021-12-14.

[24] What is css? https://www.w3.org/standards/webdesign/htmlcss#whatcss. Accessed: 2021-12-14.

[25] Javascript features. https://www.studytonight.com/javascript/javascript-features. Accessed: 2021-12-15.

[26] What and why react.js? https://www.c-sharpcorner.com/article/what-and-why-reactjs/. Accessed: 2021-12-17.

[27] What is "props" and how to use it in react? https://itnext.io/what-is-props-and-how-to-use-it-in-react-da307f500da0. Accessed: 2021-12-18.

[28] Migrations | django documentation. https://docs.djangoproject.com/en/4.0/topics/migrations/. Accessed: 2022-01-25.

[29] What is data ingestion and why this technology matters. https://www.striim.com/what-is-data-ingestion-and-why-this-technology-matters/. Accessed: 2022-01-27.

ΠΑΡΑΡΤΗΜΑΤΑ

Παράρτημα Α

Κώδικας

Ο κώδικας που δημιουργήθηκε στα πλαίσια της παρούσας διπλωματικής είναι διαθέσιμος στους ακόλουθους συνδέσμους:

- `https://github.com/chvoul/monitoring`
- `https://github.com/chvoul/monitoring_web_app_api`
- `https://github.com/chvoul/front_end_monitoring`