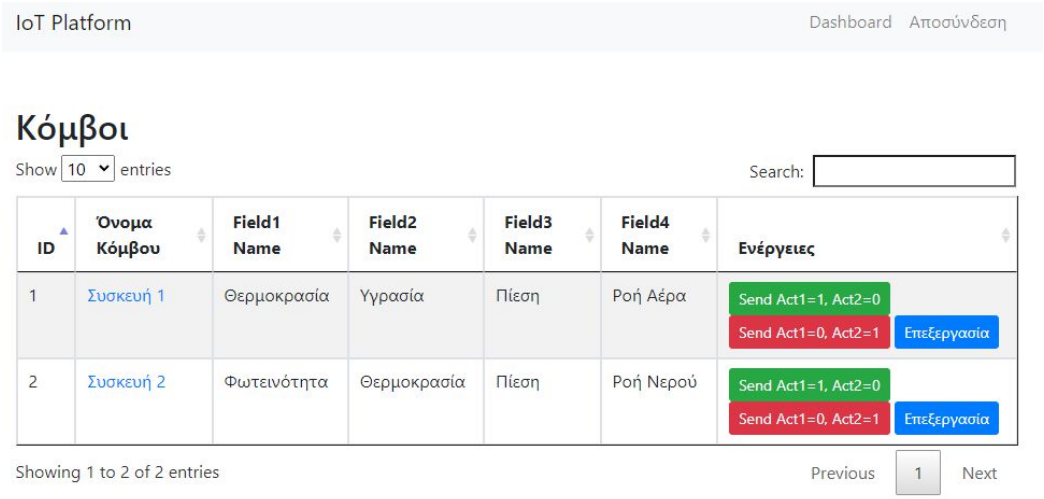
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Πλατφόρμα Internet of Things με υποστήριξη websockets»



Φοιτητής

Μυγδαλιάς Αθανάσιος

134102

Επιβλέπων

Δρ. Κυριάκος Τσιακμάκης

Σεπτέμβριος 2024

Πλατφόρμα Internet of Things με υποστήριξη websockets

Κωδικός: 24174

Φοιτητής: Μυγδαλιάς Αθανάσιος

Εισηγητής: Δρ Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 27-03-2024

Ημερομηνία περάτωσης Π.Ε. 10-09-2024

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 πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Περίληψη

Η εργασία αυτή αφορά την ανάπτυξη μιας πλατφόρμας Internet of Things (IoT) με ενσωμάτωση τεχνολογίας WebSocket, που επιτρέπει αμφίδρομη επικοινωνία σε πραγματικό χρόνο μεταξύ συσκευών IoT και διακομιστών. Ο κύριος στόχος της πλατφόρμας είναι να διευκολύνει την ανταλλαγή δεδομένων και την αποστολή εντολών ελέγχου με ταχύτητα και ασφάλεια, βελτιώνοντας την απόκριση και τη λειτουργικότητα των εφαρμογών IoT. Η υλοποίηση περιλαμβάνει τη δημιουργία ενός Python server που επικοινωνεί με τους clients μέσω WebSockets, εξασφαλίζοντας την αποστολή και λήψη δεδομένων σε πραγματικό χρόνο. Τα δεδομένα που συλλέγονται από τις συσκευές θα αποθηκεύονται σε βάση δεδομένων για ανάλυση και περαιτέρω επεξεργασία. Επιπλέον, ένας εξουσιοδοτημένος χρήστης θα έχει τη δυνατότητα πρόσβασης σε μια ειδικά σχεδιασμένη ιστοσελίδα βασισμένη στο Flask framework, όπου θα μπορεί να παρακολουθεί τα δεδομένα κάθε κόμβου, να δημιουργεί, να τροποποιεί, να διαγράφει κόμβους, καθώς και να διαχειρίζεται αισθητήρες και ενεργοποιητές. Η πλατφόρμα αυτή επιδιώκει να παρέχει μια ολοκληρωμένη λύση για τη διαχείριση συστημάτων IoT, επιτρέποντας την εύκολη και αποδοτική διασύνδεση μεταξύ των συσκευών και των εφαρμογών.

« Internet of Things Platform with WebSocket Support »

Abstract

This project involves the development of an Internet of Things (IoT) platform with integrated WebSocket technology, enabling real-time, bidirectional communication between IoT devices and servers. The primary goal of the platform is to facilitate efficient data exchange and command control with speed and security, enhancing the responsiveness and functionality of IoT applications. The implementation includes the creation of a Python server that communicates with clients via WebSockets, ensuring real-time data transmission and reception. The data collected from the devices will be stored in a database for analysis and further processing. Additionally, an authorized user will have access to a specially designed website built on the Flask framework, where they can monitor each node's data, create, modify, and delete nodes, as well as manage sensors and actuators. This platform aims to provide a comprehensive solution for managing IoT systems, allowing for seamless and efficient integration between devices and applications.

Ευχαριστίες

Θέλω να ευχαριστήσω τους γονείς μου για τη συμπαράσταση τους και τον επιβλέπων για τη συνεχή παιδαγωγική καθοδήγηση, επιστημονικές οδηγίες του και συμβολή του στον κώδικα της εφαρμογής.

Περιεχόμενα

Περίληψη iv
Abstract v
Ευχαριστίες vi
Περιεχόμενα vii
Κατάλογος Σχημάτων viii
Κεφάλαιο 1ο: Εισαγωγή 9
1.1 Εισαγωγή 9
1.2 Δομή της εργασίας 10
Κεφάλαιο 2ο: Βιβλιογραφική Ανασκόπηση – Παρόμοια συστήματα 11
2.1 Βιβλιογραφική Ανασκόπηση 11
2.2 Παρόμοια συστήματα 13
2.2.1 ThingSpeak 13
2.2.2 Thingsboard 15
2.2.3 Thinger.io 16
Κεφάλαιο 3ο: Τεχνολογία – Γλώσσες Προγραμματισμού 19
3.1 Python 19
3.1.1 Python Server 21
3.2 Websockets 25
3.3 Python+Websockets 27
3.4 Flask 28
3.5 MySQL 31
Κεφάλαιο 4ο: Το σύστημα Websocket 34
4.1 Το σύστημα 34
4.2 Websocket server 41
4.3 Websocket client 47
4.4 Flask server – Ιστοσελίδα 49
4.5 Η βάση 51
4.6 Ασφάλεια στο σύστημα και στα δεδομένα 53
Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης 54
ΒΙΒΛΙΟΓΡΑΦΙΑ 56
ΠΑΡΑΡΤΗΜΑ Α 57

Κατάλογος Σχημάτων

Εικόνα 2.1: ThingSpeak 13
Εικόνα 2.2: Thingsboard 15
Εικόνα 2.3: Thinger.io 17
Εικόνα 3.1: Χρήσεις Python 19
Εικόνα 3.2: Flask 29
Εικόνα 4.1: Το ολοκληρωμένο σύστημα 34
Εικόνα 4.2: Ο Server δέχεται τιμές από κόμβος ή στέλνει τιμές σε αυτούς 35
Εικόνα 4.3: Η αρχική της πλατφόρμας 36
Εικόνα 4.4: Σελίδα για Εγγραφή στη πλατφόρμα 36
Εικόνα 4.5: Σελίδα για Σύνδεση στη πλατφόρμα 37
Εικόνα 4.6: Σελίδα για προβολή κόμβων με όλα τα χαρακτηριστικά-πεδία και ενέργειες 37
Εικόνα 4.7: Σελίδα επεξεργασίας κόμβου 38
Εικόνα 4.8: Σελίδα προβολής των δύο πρώτων πεδίων του κόμβου σε chart 39
Εικόνα 4.9: Σελίδα προβολής των δύο τελευταίων πεδίων του κόμβου σε chart 40
Εικόνα 4.10: Ο Websocket server δέχεται ένα websocket πελάτη ενώ ταυτόχρονα δέχεται και act από τη σελίδα 40
Εικόνα 4.11: Websocket πελάτης – κόμβος (πχ esp32) που στέλνει δεδομένα στον Server 41

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Στην εποχή της ψηφιακής τεχνολογίας, το Διαδίκτυο των Πραγμάτων (Internet of Things - IoT) αναδεικνύεται ως ένας από τους πιο δυναμικούς τομείς ανάπτυξης, προσφέροντας νέες δυνατότητες συνδεσιμότητας και επικοινωνίας μεταξύ των συσκευών. Η συνεχής ανταλλαγή δεδομένων μεταξύ των συσκευών αυτών δημιουργεί ένα εκτεταμένο οικοσύστημα, όπου η αποδοτική διαχείριση των πληροφοριών και η ακριβής παρακολούθηση των διαδικασιών είναι κρίσιμη. Ωστόσο, για να επιτευχθεί αυτός ο στόχος, απαιτούνται λύσεις που να επιτρέπουν την άμεση και αξιόπιστη επικοινωνία μεταξύ των συσκευών και των κεντρικών συστημάτων, διασφαλίζοντας την ταχύτητα και την ασφάλεια των μεταδιδόμενων δεδομένων.

Σε αυτό το πλαίσιο, η παρούσα εργασία εστιάζει στην ανάπτυξη μιας πλατφόρμας IoT με υποστήριξη WebSockets, η οποία προσφέρει διπλής κατεύθυνσης επικοινωνία σε πραγματικό χρόνο μεταξύ των συσκευών και των διακομιστών. Η χρήση των WebSockets επιτρέπει τη συνεχή σύνδεση και την άμεση ανταλλαγή δεδομένων χωρίς καθυστερήσεις, βελτιώνοντας τη συνολική λειτουργικότητα των εφαρμογών IoT. Η πλατφόρμα αυτή σχεδιάστηκε για να προσφέρει μια ολοκληρωμένη λύση διαχείρισης, επιτρέποντας στους χρήστες να παρακολουθούν και να ελέγχουν εύκολα και αποτελεσματικά τις συνδεδεμένες συσκευές τους.

Η υλοποίηση της πλατφόρμας περιλαμβάνει τη δημιουργία ενός server γραμμένου σε Python, ο οποίος επικοινωνεί με τους clients μέσω WebSockets, καθώς και την ανάπτυξη μιας διαδικτυακής διεπαφής με χρήση του Flask framework. Η διεπαφή αυτή επιτρέπει στους εξουσιοδοτημένους χρήστες να διαχειρίζονται τους κόμβους του συστήματος, να προσθέτουν ή να αφαιρούν αισθητήρες και ενεργοποιητές, και να παρακολουθούν τα δεδομένα σε πραγματικό χρόνο. Η συγκεκριμένη πλατφόρμα όχι μόνο βελτιώνει την αλληλεπίδραση εντός του οικοσυστήματος του IoT, αλλά και συμβάλλει στη συνολική βελτίωση της αποδοτικότητας και της απόκρισης των εφαρμογών που βασίζονται στο Διαδίκτυο των Πραγμάτων.

Ο κύριος στόχος της εργασίας είναι η ανάπτυξη μιας ολοκληρωμένης πλατφόρμας Internet of Things (IoT) με υποστήριξη WebSockets, που θα επιτρέπει την αμφίδρομη επικοινωνία σε πραγματικό χρόνο μεταξύ των συνδεδεμένων συσκευών και του κεντρικού διακομιστή. Η πλατφόρμα αυτή στοχεύει να βελτιώσει την αποδοτικότητα και την ταχύτητα της ανταλλαγής δεδομένων, ενώ παράλληλα να παρέχει στους χρήστες ένα εύχρηστο και ασφαλές περιβάλλον διαχείρισης των IoT συστημάτων τους. Μέσω της πλατφόρμας, οι χρήστες θα μπορούν να παρακολουθούν, να ελέγχουν και να διαχειρίζονται

δυναμικά τις συνδεδεμένες συσκευές και τους αισθητήρες τους, συμβάλλοντας έτσι στην ενίσχυση της λειτουργικότητας και της απόκρισης των εφαρμογών IoT.

Ο κύριος στόχος της εργασίας είναι η ανάπτυξη μιας ολοκληρωμένης πλατφόρμας Internet of Things (IoT) με υποστήριξη WebSockets, που θα επιτρέπει την αμφίδρομη επικοινωνία σε πραγματικό χρόνο μεταξύ των συνδεδεμένων συσκευών και του κεντρικού διακομιστή. Η πλατφόρμα αυτή στοχεύει να βελτιώσει την αποδοτικότητα και την ταχύτητα της ανταλλαγής δεδομένων, ενώ παράλληλα να παρέχει στους χρήστες ένα εύχρηστο και ασφαλές περιβάλλον διαχείρισης των IoT συστημάτων τους. Μέσω της πλατφόρμας, οι χρήστες θα μπορούν να παρακολουθούν, να ελέγχουν και να διαχειρίζονται δυναμικά τις συνδεδεμένες συσκευές και τους αισθητήρες τους, συμβάλλοντας έτσι στην ενίσχυση της λειτουργικότητας και της απόκρισης των εφαρμογών IoT.

1.2 Δομή της εργασίας

Η δομή της εργασίας έχει ως εξής:

Στο πρώτο κεφάλαιο παρουσιάζεται μια εισαγωγή στην εργασία, μαζί με τους βασικούς στόχους της. Επιπλέον, γίνεται αναφορά στη συνεισφορά της και στα σημεία στα οποία εστιάζει η ανάλυση που ακολουθεί.

Στο δεύτερο κεφάλαιο, πραγματοποιείται μια βιβλιογραφική ανασκόπηση των σχετικών έργων και παρουσιάζονται παρόμοια συστήματα που έχουν αναπτυχθεί στο παρελθόν. Στη συνέχεια, γίνεται μια αναλυτική αναφορά σε συστήματα όπως το ThingSpeak, το Thingsboard και το Thinger.io.

Στο τρίτο κεφάλαιο, αναλύονται οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν στην υλοποίηση της εργασίας. Αυτά περιλαμβάνουν τη γλώσσα προγραμματισμού Python, τα WebSockets, το Flask και τη MySQL, τα οποία ενσωματώνονται στην εφαρμογή και αποτελούν τη βάση για την υλοποίησή της.

Το τέταρτο κεφάλαιο επικεντρώνεται στην παρουσίαση του συστήματος που αναπτύχθηκε. Περιλαμβάνει αναλυτικές περιγραφές με εικόνες και διαγράμματα, τα οποία εξηγούν τη λειτουργία του WebSocket server, του client, και του Flask server, καθώς και την αρχιτεκτονική της βάσης δεδομένων.

Τέλος, στο πέμπτο κεφάλαιο, παρουσιάζονται τα συμπεράσματα της εργασίας, ενώ διατυπώνονται προτάσεις για μελλοντική βελτίωση και έρευνα. Στο τέλος της εργασίας παρατίθεται η βιβλιογραφία, ενώ σε κάποια παραρτήματα μπορεί να συμπεριλαμβάνονται τμήματα κώδικα που χρησιμοποιήθηκαν.

Κεφάλαιο 2ο: Βιβλιογραφική Ανασκόπηση – Παρόμοια συστήματα

2.1 Βιβλιογραφική Ανασκόπηση

Στο έργο με τίτλο *"Study of HTML5 WebSocket for a Multimedia Communication"*, παρουσιάζεται μια αναλυτική μελέτη της τεχνολογίας WebSocket στο πλαίσιο του HTML5, εστιάζοντας στις δυνατότητες και τις εφαρμογές της για αμφίδρομη επικοινωνία σε πραγματικό χρόνο. Οι συγγραφείς διερευνούν πώς τα WebSockets μπορούν να βελτιώσουν την απόδοση των διαδικτυακών εφαρμογών, ιδιαίτερα αυτών που ασχολούνται με πολυμέσα και άλλες εφαρμογές που απαιτούν συνεχή ροή δεδομένων. Το άρθρο συγκρίνει την απόδοση των WebSockets με τις παραδοσιακές μεθόδους επικοινωνίας, όπως το polling και το long polling, τονίζοντας τα πλεονεκτήματα της χρήσης WebSockets σε ό,τι αφορά τη μείωση του φορτίου δικτύου και τη βελτίωση της ταχύτητας απόκρισης. Μέσα από πειράματα και ανάλυση δεδομένων, οι συγγραφείς καταδεικνύουν τη σημασία των WebSockets για την ανάπτυξη αποδοτικών και ασφαλών web εφαρμογών σε ένα περιβάλλον HTML5. Το έργο καταλήγει σε συμπεράσματα που αφορούν τη μελλοντική έρευνα και τις προοπτικές βελτίωσης της τεχνολογίας αυτής, ιδίως σε σχέση με τη διαλειτουργικότητα μεταξύ διαφορετικών προγραμμάτων περιήγησης και την ασφάλεια της επικοινωνίας. [1]

Στο έργο με τίτλο "WebSocket-Based IoT Temperature and Humidity Monitoring System for Underground Coal Mine," παρουσιάζεται η ανάπτυξη ενός συστήματος παρακολούθησης θερμοκρασίας και υγρασίας για υπόγειες ανθρακωρυχίες, το οποίο βασίζεται στην τεχνολογία WebSocket και το Διαδίκτυο των Πραγμάτων (IoT). Το σύστημα αυτό έχει ως στόχο να αντιμετωπίσει τις προκλήσεις ασφάλειας που αντιμετωπίζουν οι εργαζόμενοι στις ανθρακωρυχίες, ιδιαίτερα λόγω των ακραίων περιβαλλοντικών συνθηκών, όπως υψηλές θερμοκρασίες και υπερβολική υγρασία.

Οι συγγραφείς αναλύουν τη χρήση αισθητήρων DHT22/AM2302 για τη μέτρηση της θερμοκρασίας και της υγρασίας, οι οποίοι είναι συνδεδεμένοι με μια πλακέτα Arduino. Τα δεδομένα που συλλέγονται από τους αισθητήρες αποστέλλονται σε έναν διακομιστή μέσω ενός Raspberry Pi, χρησιμοποιώντας το πρωτόκολλο HTTP για την αρχική μεταφορά δεδομένων και το WebSocket API για τη συνεχή ενημέρωση των δεδομένων σε πραγματικό χρόνο στους client devices.

Η αρχιτεκτονική που προτείνεται βασίζεται στη συνδυαστική χρήση τεχνολογιών όπως το IoT, το WebSocket, και το cloud computing, προκειμένου να επιτευχθεί μια αξιόπιστη και αποδοτική λύση για την παρακολούθηση και την έγκαιρη αντίδραση σε επικίνδυνες καταστάσεις στις ανθρακωρυχίες. Το σύστημα προσφέρει επίσης δυνατότητες αποθήκευσης των δεδομένων σε cloud servers και παροχής

ειδοποιήσεων στους χρήστες όταν οι μετρήσεις υπερβαίνουν τα ασφαλή όρια, βελτιώνοντας έτσι την ασφάλεια των εργαζομένων.

Το έργο καταλήγει στο συμπέρασμα ότι η προτεινόμενη αρχιτεκτονική είναι μια αποτελεσματική λύση για την παρακολούθηση των περιβαλλοντικών συνθηκών σε υπόγειες ανθρακωρυχίες και υπογραμμίζει τη σημασία της χρήσης τεχνολογιών όπως τα WebSockets για την παροχή συνεχούς, αξιόπιστης επικοινωνίας σε πραγματικό χρόνο. Επιπλέον, προτείνονται μελλοντικές βελτιώσεις, όπως η ενσωμάτωση περισσότερων αισθητήρων και η χρήση τεχνικών μηχανικής μάθησης για την πρόβλεψη πιθανών κινδύνων με βάση τα δεδομένα που συλλέγονται.

[Prince, M. R. I., & Islam, M. R. (2021, December). WebSocket Based IoT Temperature and Humidity Monitoring System for Underground Coal Mine. In 2021 5th International Conference on Electrical Information and Communication Technology (EICT) (pp. 1-5). IEEE.]

Στο έργο με τίτλο "Introducing WebSocket-Based Real-Time Monitoring System for Remote Intelligent Buildings," που δημοσιεύτηκε στο International Journal of Distributed Sensor Networks από τους Kun Ma και Runyuan Sun, παρουσιάζεται ένα σύστημα παρακολούθησης σε πραγματικό χρόνο, το οποίο βασίζεται στην τεχνολογία WebSocket και στοχεύει στη διαχείριση και παρακολούθηση έξυπνων κτιρίων από απόσταση. Το έργο επικεντρώνεται στην ανάπτυξη μιας αρχιτεκτονικής που ενσωματώνει ασύρματα δίκτυα αισθητήρων (WSNs) για τη συλλογή δεδομένων από διάφορους αισθητήρες, όπως αυτοί που μετρούν τη θερμοκρασία, την υγρασία, και άλλες περιβαλλοντικές παραμέτρους.

Οι συγγραφείς επισημαίνουν τα πλεονεκτήματα της χρήσης του πρωτοκόλλου WebSocket σε σχέση με τις παραδοσιακές μεθόδους, όπως το HTTP polling και τα FlashSockets, υπογραμμίζοντας τη μειωμένη καθυστέρηση, τη χαμηλή κατανάλωση ενέργειας, και την ικανότητα διαχείρισης μεγάλου αριθμού ταυτόχρονων συνδέσεων. Η προτεινόμενη λύση ενσωματώνει επίσης τεχνολογίες HTML5, όπως το Canvas API και το Chart.js, για την οπτική αναπαράσταση των δεδομένων που συλλέγονται σε πραγματικό χρόνο μέσω του web browser.

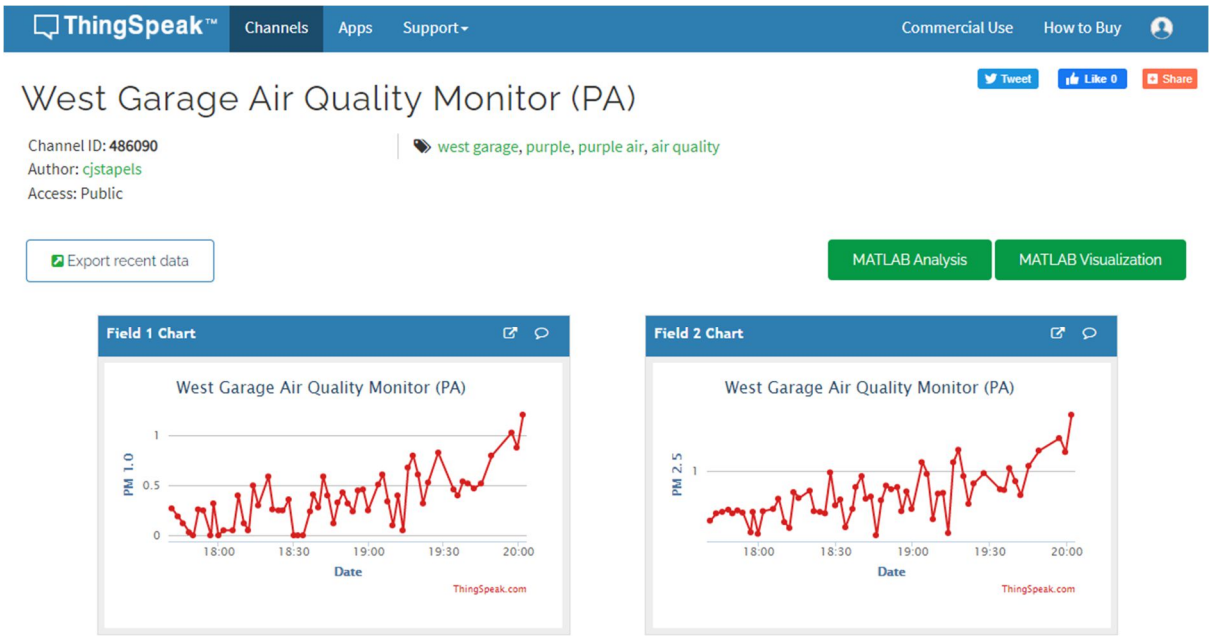
Το έργο παρουσιάζει επίσης ένα μοντέλο αποθήκευσης των ιστορικών δεδομένων παρακολούθησης χρησιμοποιώντας μια NoSQL βάση δεδομένων, όπως η MongoDB, για τη μείωση της πλεονάζουσας πληροφορίας και την αποδοτική διαχείριση μεγάλων όγκων δεδομένων. Οι πειραματικές δοκιμές δείχνουν ότι η προτεινόμενη προσέγγιση υπερέχει των παραδοσιακών μεθόδων παρακολούθησης σε ό,τι αφορά τη γρήγορη φόρτωση, τη χαμηλή καθυστέρηση, την υψηλή διαθεσιμότητα και την αποδοτικότητα της αποθήκευσης.

Το άρθρο καταλήγει ότι το WebSocket-based σύστημα παρακολούθησης μπορεί να προσφέρει σημαντικά οφέλη στη διαχείριση έξυπνων κτιρίων, επιτρέποντας την απομακρυσμένη παρακολούθηση και τον έλεγχο των συστημάτων, ενώ παράλληλα βελτιώνει την αποδοτικότητα και μειώνει το κόστος. [2]

2.2 Παρόμοια συστήματα

2.2.1 ThingSpeak

Το ThingSpeak είναι μια πλατφόρμα ανοιχτού κώδικα που σχεδιάστηκε για την αποθήκευση, ανάλυση και οπτικοποίηση δεδομένων που συλλέγονται από συσκευές Internet of Things (IoT). Αναπτύχθηκε από την MathWorks και είναι γνωστό για την ευκολία χρήσης του και τη δυνατότητα να επιτρέπει στους χρήστες να συλλέγουν, να επεξεργάζονται και να αναλύουν δεδομένα σε πραγματικό χρόνο από αισθητήρες και άλλες συσκευές IoT. Το ThingSpeak υποστηρίζει επίσης την ενσωμάτωση με άλλα εργαλεία, όπως το MATLAB, παρέχοντας ισχυρές δυνατότητες ανάλυσης δεδομένων και μοντελοποίησης.



Εικόνα 2.1: ThingSpeak

[https://blogs.mathworks.com/iot/files/2021/05/thingspeak_channel_view.png]

Χαρακτηριστικά

Συλλογή Δεδομένων σε Πραγματικό Χρόνο: Το ThingSpeak επιτρέπει στους χρήστες να συλλέγουν δεδομένα από IoT συσκευές σε πραγματικό χρόνο. Τα δεδομένα μπορούν να αποστέλλονται στην

πλατφόρμα μέσω HTTP, MQTT ή άλλων πρωτοκόλλων, καθιστώντας την ιδανική για εφαρμογές που απαιτούν συνεχή παρακολούθηση και ανάλυση.

Οπτικοποίηση Δεδομένων: Μία από τις κύριες δυνατότητες του ThingSpeak είναι η ενσωματωμένη δυνατότητα οπτικοποίησης δεδομένων. Οι χρήστες μπορούν να δημιουργούν γραφήματα και άλλα είδη οπτικών αναπαραστάσεων των δεδομένων τους απευθείας μέσα στην πλατφόρμα, διευκολύνοντας την κατανόηση και την ανάλυση των πληροφοριών.

Ανάλυση Δεδομένων με MATLAB: Το ThingSpeak παρέχει ενσωματωμένες δυνατότητες ανάλυσης δεδομένων μέσω του MATLAB. Οι χρήστες μπορούν να γράφουν scripts MATLAB απευθείας στην πλατφόρμα για να εκτελούν σύνθετες αναλύσεις, προσομοιώσεις ή μοντελοποιήσεις των δεδομένων τους.

Διαχείριση Καναλιών: Τα δεδομένα στο ThingSpeak οργανώνονται σε "κανάλια", καθένα από τα οποία μπορεί να περιέχει πολλαπλά πεδία δεδομένων. Οι χρήστες μπορούν να δημιουργούν, να διαχειρίζονται και να μοιράζονται αυτά τα κανάλια, επιτρέποντας την ευέλικτη και οργανωμένη αποθήκευση δεδομένων.

Ενσωμάτωση με το IoT: Το ThingSpeak είναι σχεδιασμένο να συνεργάζεται εύκολα με πολλές IoT συσκευές και πλατφόρμες. Υποστηρίζει διάφορα πρωτόκολλα επικοινωνίας και μπορεί να συνδεθεί με άλλες πλατφόρμες IoT για τη δημιουργία ολοκληρωμένων συστημάτων παρακολούθησης και ελέγχου.[3]

Πλεονεκτήματα

Το ThingSpeak είναι γνωστό για την απλή και κατανοητή διεπαφή του, που επιτρέπει ακόμη και σε αρχάριους χρήστες να το χρησιμοποιούν αποτελεσματικά. Η δυνατότητα ανάλυσης δεδομένων μέσω MATLAB παρέχει ισχυρά εργαλεία για την επεξεργασία και ανάλυση των δεδομένων σε βάθος. Το ThingSpeak είναι ανοιχτού κώδικα, γεγονός που επιτρέπει στους χρήστες να προσαρμόζουν την πλατφόρμα σύμφωνα με τις ανάγκες τους.

Μειονεκτήματα

Η δωρεάν έκδοση του ThingSpeak έχει περιορισμούς στον αποθηκευτικό χώρο και τη συχνότητα των ενημερώσεων δεδομένων, γεγονός που μπορεί να είναι περιοριστικό για εφαρμογές που απαιτούν συχνές ενημερώσεις σε πραγματικό χρόνο.

Για να αξιοποιηθούν πλήρως οι δυνατότητες ανάλυσης του ThingSpeak, απαιτείται γνώση και άδεια χρήσης του MATLAB, κάτι που μπορεί να είναι περιοριστικό για ορισμένους χρήστες.

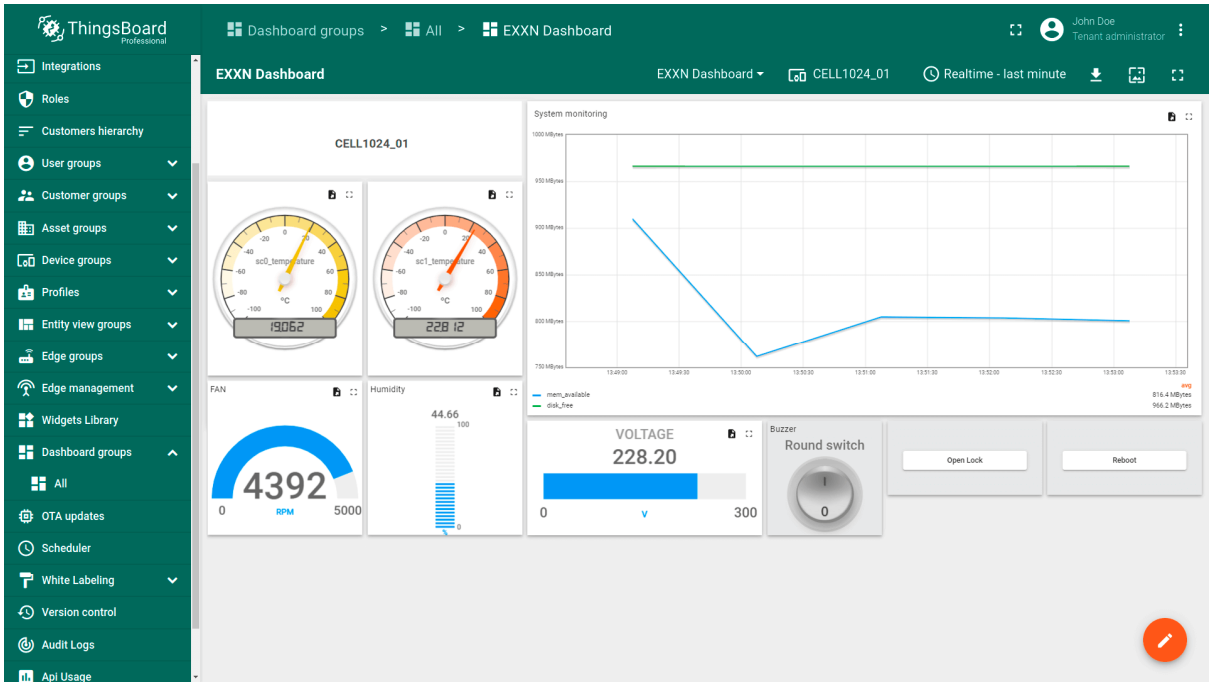
Γενικά οι χρήστες μπορούν να συλλέγουν και να παρακολουθούν δεδομένα από αισθητήρες θερμοκρασίας, υγρασίας, και ατμοσφαιρικής πίεσης σε πραγματικό χρόνο, προσφέροντας ένα εργαλείο για τη διαχείριση κλιματικών συνθηκών.

Το ThingSpeak μπορεί να χρησιμοποιηθεί για τον έλεγχο και την παρακολούθηση συσκευών σε ένα έξυπνο σπίτι, όπως θερμοστάτες, φώτα και συστήματα ασφαλείας, επιτρέποντας την απομακρυσμένη διαχείριση μέσω μιας ενιαίας πλατφόρμας.

Επιχειρήσεις μπορούν να χρησιμοποιούν το ThingSpeak για τη συλλογή δεδομένων από μηχανήματα και συστήματα, και με τη χρήση του MATLAB να αναπτύσσουν μοντέλα πρόβλεψης που βοηθούν στη συντήρηση και την αποφυγή αστοχιών. [4]

2.2.2 Thingsboard

Το ThingsBoard είναι μια ισχυρή πλατφόρμα ανοιχτού κώδικα για τη διαχείριση, τη συλλογή και την οπτικοποίηση δεδομένων από συσκευές Internet of Things (IoT). Η πλατφόρμα παρέχει μια ολοκληρωμένη λύση για τη δημιουργία εφαρμογών IoT, επιτρέποντας στους χρήστες να συνδέουν και να παρακολουθούν συσκευές, να διαχειρίζονται δεδομένα σε πραγματικό χρόνο και να αναπτύσσουν λογικές κανόνες για την αυτοματοποίηση των συστημάτων τους.



Εικόνα 2.2: Thingsboard

[https://img.thingsboard.io/samples/exxn/ennx-dashboard.png]

Ένα από τα κύρια χαρακτηριστικά του ThingsBoard είναι η ευελιξία του όσον αφορά την υποστήριξη πολλαπλών πρωτοκόλλων επικοινωνίας, όπως MQTT, CoAP, και HTTP, επιτρέποντας τη σύνδεση και την επικοινωνία με μια ευρεία ποικιλία συσκευών IoT. Επιπλέον, η πλατφόρμα υποστηρίζει προηγμένα εργαλεία ανάλυσης και οπτικοποίησης δεδομένων, επιτρέποντας στους χρήστες να δημιουργούν προσαρμοσμένους πίνακες ελέγχου (dashboards) που απεικονίζουν τα δεδομένα σε πραγματικό χρόνο, μέσω γραφημάτων, διαγραμμάτων και άλλων οπτικών αναπαραστάσεων.

Το ThingsBoard περιλαμβάνει επίσης δυνατότητες για τη δημιουργία κανόνων αυτοματοποίησης, επιτρέποντας στους χρήστες να διαμορφώνουν κανόνες που αντιδρούν σε συγκεκριμένα γεγονότα ή συνθήκες. Αυτό καθιστά την πλατφόρμα ιδανική για εφαρμογές που απαιτούν άμεση αντίδραση σε αλλαγές περιβαλλοντικών συνθηκών ή άλλων παραμέτρων. Για παράδειγμα, η πλατφόρμα μπορεί να προγραμματιστεί να στέλνει ειδοποιήσεις ή να ενεργοποιεί συσκευές με βάση προκαθορισμένα όρια που καθορίζονται από τους χρήστες.

Ένα άλλο σημαντικό πλεονέκτημα του ThingsBoard είναι η κλιμακωσιμότητα του. Η πλατφόρμα μπορεί να διαχειριστεί χιλιάδες συσκευές και εκατομμύρια μηνύματα ανά ημέρα, καθιστώντας την ιδανική για μεγάλες επιχειρήσεις και έργα που απαιτούν υψηλή απόδοση και αξιοπιστία. Η ευελιξία της πλατφόρμας επεκτείνεται επίσης και στη δυνατότητα ανάπτυξης της σε διάφορα περιβάλλοντα, όπως το cloud, on-premise εγκαταστάσεις, ή υβριδικά μοντέλα.

Ωστόσο, παρόλο που το ThingsBoard προσφέρει πολλά πλεονεκτήματα, υπάρχουν και ορισμένα μειονεκτήματα που πρέπει να ληφθούν υπόψη. Η πολυπλοκότητα της πλατφόρμας μπορεί να αποτελεί πρόκληση για τους αρχάριους χρήστες, απαιτώντας μια καμπύλη εκμάθησης για την πλήρη αξιοποίηση των δυνατοτήτων της. Επιπλέον, η εκτέλεση της πλατφόρμας σε μεγάλα περιβάλλοντα μπορεί να απαιτεί σημαντικούς υπολογιστικούς πόρους, ιδιαίτερα αν χρησιμοποιούνται προηγμένες λειτουργίες όπως η επεξεργασία μεγάλων ποσοτήτων δεδομένων σε πραγματικό χρόνο.

Παρά τα μειονεκτήματα αυτά, το ThingsBoard παραμένει μια εξαιρετική επιλογή για επιχειρήσεις και οργανισμούς που αναζητούν μια ευέλικτη και ισχυρή πλατφόρμα IoT. Η ικανότητα του ThingsBoard να προσφέρει ολοκληρωμένες λύσεις για τη διαχείριση και την ανάλυση δεδομένων από συσκευές IoT το καθιστά ιδανικό για εφαρμογές σε διάφορους τομείς, όπως η βιομηχανία, η γεωργία, και η έξυπνη πόλη. [5]

2.2.3 Thinger.io

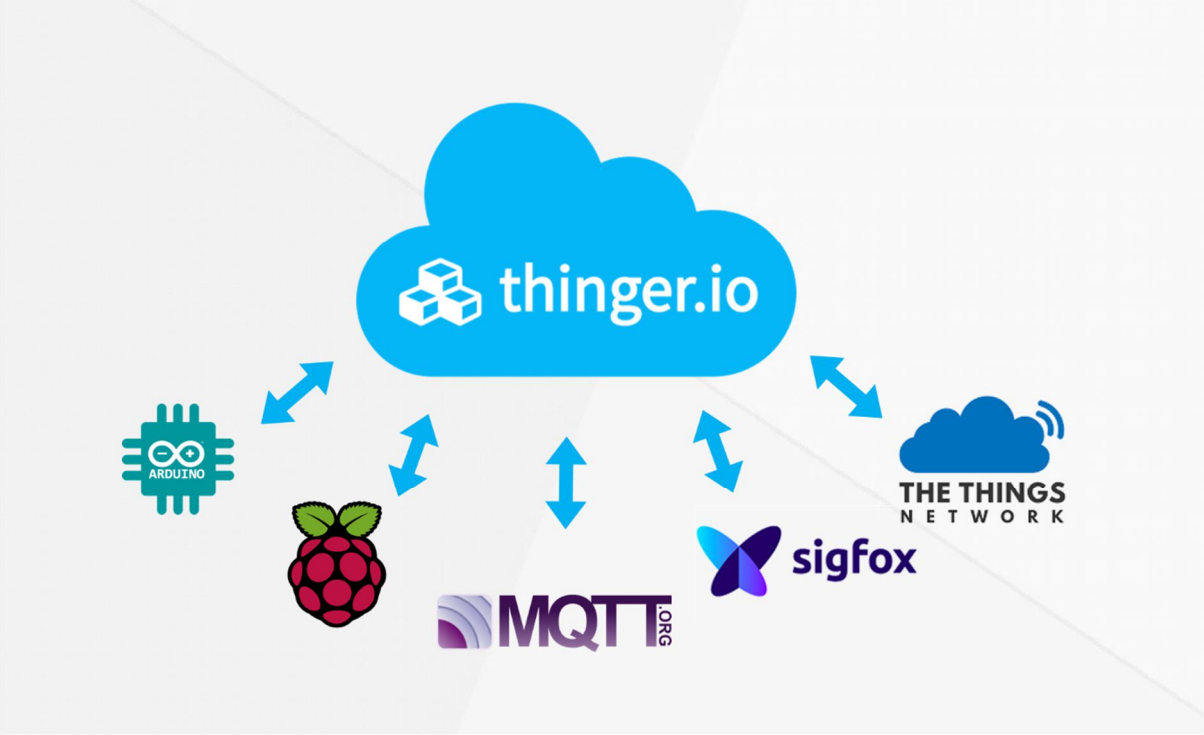
Το Thinger.io είναι μια πλατφόρμα διαχείρισης Internet of Things (IoT) που προσφέρει μια ολοκληρωμένη λύση για τη σύνδεση, την αποθήκευση και την οπτικοποίηση δεδομένων από συσκευές IoT. Η πλατφόρμα είναι σχεδιασμένη να είναι φιλική προς τον χρήστη, επιτρέποντας την εύκολη σύνδεση συσκευών και τη διαχείριση δεδομένων σε πραγματικό χρόνο. Το Thinger.io υποστηρίζει την

ανάπτυξη εφαρμογών IoT σε διάφορους τομείς, όπως η έξυπνη οικία, η βιομηχανία και η γεωργία, προσφέροντας μια ευέλικτη και επεκτάσιμη λύση για έργα κάθε μεγέθους.

Ένα από τα βασικά χαρακτηριστικά του Thinger.io είναι η δυνατότητα σύνδεσης συσκευών χρησιμοποιώντας κοινά πρωτόκολλα επικοινωνίας, όπως MQTT, HTTP, και WebSockets. Αυτή η ευελιξία επιτρέπει την ενσωμάτωση μιας ευρείας γκάμας συσκευών και αισθητήρων, διευκολύνοντας τη δημιουργία πολύπλοκων συστημάτων IoT. Η πλατφόρμα προσφέρει επίσης εργαλεία για την αποθήκευση δεδομένων στο cloud, επιτρέποντας την ασφαλή και αξιόπιστη αποθήκευση των πληροφοριών που συλλέγονται από τις συσκευές.

Το Thinger.io διαθέτει επίσης ένα ισχυρό σύστημα οπτικοποίησης δεδομένων, που επιτρέπει στους χρήστες να δημιουργούν διαδραστικούς πίνακες ελέγχου (dashboards) με γραφήματα, χάρτες και άλλες οπτικές αναπαραστάσεις. Αυτό διευκολύνει την παρακολούθηση και την ανάλυση των δεδομένων σε πραγματικό χρόνο, καθιστώντας το Thinger.io ιδανικό για εφαρμογές που απαιτούν συνεχή παρακολούθηση και άμεση αντίδραση σε αλλαγές των συνθηκών.

Επιπλέον, η πλατφόρμα προσφέρει δυνατότητες για την αυτοματοποίηση συστημάτων μέσω κανόνων και ενσωματωμένων αλγορίθμων, επιτρέποντας στους χρήστες να προγραμματίζουν ενέργειες βάσει συγκεκριμένων συνθηκών. Αυτή η λειτουργία είναι ιδιαίτερα χρήσιμη σε περιπτώσεις όπου η ταχύτητα αντίδρασης είναι κρίσιμη, όπως σε βιομηχανικές εφαρμογές ή στην παρακολούθηση κρίσιμων περιβαλλοντικών παραμέτρων.



Εικόνα 2.3: Thinger.io

[https://thinger.io/wp-content/uploads/2020/04/im-mqtt.png]

Ένα από τα σημαντικά πλεονεκτήματα του Thinger.io είναι η ευκολία χρήσης του. Η πλατφόρμα είναι σχεδιασμένη για να είναι προσιτή ακόμα και για αρχάριους χρήστες, με μια φιλική προς τον χρήστη διεπαφή που απλοποιεί τη διαδικασία σύνδεσης και διαχείρισης συσκευών. Επιπλέον, το Thinger.io προσφέρει μια δωρεάν έκδοση της πλατφόρμας, καθιστώντας το μια ελκυστική επιλογή για μικρά έργα ή δοκιμές.

Ωστόσο, υπάρχουν και ορισμένα μειονεκτήματα που πρέπει να ληφθούν υπόψη. Η δωρεάν έκδοση της πλατφόρμας έχει περιορισμούς όσον αφορά τον αριθμό των συνδεδεμένων συσκευών και τον αποθηκευτικό χώρο, γεγονός που μπορεί να αποτελέσει πρόβλημα για μεγαλύτερα έργα. Επίσης, ενώ η πλατφόρμα είναι εύκολη στη χρήση, οι προηγμένες δυνατότητες αυτοματοποίησης και ανάλυσης μπορεί να απαιτούν βαθύτερη κατανόηση του συστήματος, καθιστώντας τη λιγότερο ιδανική για ορισμένους χρήστες.

Συνολικά, το Thinger.io είναι μια ισχυρή και ευέλικτη πλατφόρμα IoT που προσφέρει μια ολοκληρωμένη λύση για τη σύνδεση, τη διαχείριση και την ανάλυση δεδομένων από συσκευές IoT. Η ευκολία χρήσης και η δυνατότητα για προσαρμογή καθιστούν την πλατφόρμα μια ελκυστική επιλογή για χρήστες που αναζητούν μια αξιόπιστη λύση για την ανάπτυξη εφαρμογών IoT. [6][7]

Κεφάλαιο 3ο: Τεχνολογία – Γλώσσες Προγραμματισμού

3.1 Python

Η Python είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού γενικού σκοπού, η οποία αναπτύχθηκε για πρώτη φορά στις αρχές της δεκαετίας του 1990 από τον Guido van Rossum. Η γλώσσα αυτή ξεχωρίζει για την απλότητά της και την καθαρότητα του συντακτικού της, γεγονός που την καθιστά ιδανική τόσο για αρχάριους όσο και για έμπειρους προγραμματιστές. Με την πάροδο των ετών, η Python έχει εξελιχθεί σε μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού παγκοσμίως, χρησιμοποιούμενη σε ένα ευρύ φάσμα εφαρμογών, από την ανάπτυξη ιστοσελίδων και τον αυτοματισμό διαδικασιών μέχρι την ανάλυση δεδομένων και την τεχνητή νοημοσύνη.

Εικόνα 3.1: Χρήσεις Python

[https://www.98thpercentile.com/hs-fs/hubfs/What%20is%20python%20used%20for-1.jpg?width=713&height=357&name=What%20is%20python%20used%20for-1.jpg]

Ένας από τους κύριους λόγους για την εκτεταμένη χρήση της Python είναι η πλούσια βιβλιοθήκη εργαλείων και πλαισίων που διαθέτει. Οι βιβλιοθήκες αυτές προσφέρουν μια πληθώρα λειτουργιών και αλγορίθμων, διευκολύνοντας τον προγραμματιστή στην υλοποίηση σύνθετων έργων με λιγότερη προσπάθεια. Επιπλέον, η Python υποστηρίζει πολλαπλά προγραμματιστικά παραδείγματα, όπως αντικειμενοστραφή, λειτουργικό και διαδικαστικό προγραμματισμό, παρέχοντας μεγάλη ευελιξία στον τρόπο που μπορεί να χρησιμοποιηθεί.

Η απλότητα και η ευελιξία της Python την καθιστούν επίσης ιδανική για ταχύτερη ανάπτυξη πρωτοτύπων και επαναληπτικών διαδικασιών ανάπτυξης λογισμικού. Αυτός είναι ο λόγος που συχνά

επιλέγεται για έργα που απαιτούν γρήγορες και ευέλικτες λύσεις, όπως η ανάπτυξη εφαρμογών IoT, η επιστημονική έρευνα, και οι αναλύσεις δεδομένων. Παράλληλα, η Python έχει μια ζωντανή κοινότητα χρηστών και προγραμματιστών που συνεχώς συνεισφέρουν στην ανάπτυξη και τη βελτίωση της γλώσσας, καθιστώντας την ένα από τα πιο δυναμικά οικοσυστήματα στον κόσμο του προγραμματισμού. [8][9]

Η Python έχει καταστεί μία από τις πιο ευέλικτες γλώσσες προγραμματισμού, με χρήση σε πληθώρα εφαρμογών και τομέων της τεχνολογίας. Ένας από τους κυριότερους τομείς όπου η Python βρίσκει ευρεία εφαρμογή είναι η ανάπτυξη web εφαρμογών. Με τη βοήθεια ισχυρών frameworks, όπως το Django και το Flask, οι προγραμματιστές μπορούν να δημιουργήσουν πλήρως λειτουργικές και ασφαλείς ιστοσελίδες με λιγότερο κώδικα και σε μικρότερο χρόνο. Η απλότητα και η καθαρότητα του συντακτικού της Python καθιστούν τη γλώσσα ιδανική για την ταχεία ανάπτυξη web εφαρμογών, επιτρέποντας την ευκολότερη συντήρηση και αναβάθμιση του κώδικα.

Ένας άλλος σημαντικός τομέας όπου η Python διαπρέπει είναι η ανάλυση δεδομένων και η επιστήμη δεδομένων (Data Science). Χάρη σε βιβλιοθήκες όπως το Pandas, το NumPy, και το Matplotlib, η Python επιτρέπει στους επιστήμονες δεδομένων να επεξεργάζονται, να αναλύουν και να οπτικοποιούν μεγάλα σύνολα δεδομένων με αποδοτικό τρόπο. Επιπλέον, η Python χρησιμοποιείται ευρέως στην ανάπτυξη μοντέλων μηχανικής μάθησης και τεχνητής νοημοσύνης, με βιβλιοθήκες όπως το TensorFlow και το Scikit-learn, καθιστώντας την απαραίτητο εργαλείο για τον τομέα της τεχνολογίας αιχμής.

Η Python έχει επίσης ισχυρή παρουσία στον αυτοματισμό και τον έλεγχο συστημάτων. Εφαρμογές αυτοματισμού, όπως τα scripts για τη διαχείριση συστημάτων και τη ρύθμιση διακομιστών, υλοποιούνται συχνά στην Python λόγω της απλότητάς της και της ισχυρής υποστήριξης από τη κοινότητα. Επιπλέον, η Python χρησιμοποιείται στην ανάπτυξη παιχνιδιών, με βιβλιοθήκες όπως το Pygame, και στη ρομποτική, όπου παρέχει εργαλεία για τον έλεγχο και τη διαχείριση φυσικών συσκευών. Αυτή η ποικιλία χρήσεων καθιστά την Python μία από τις πιο πολύπλευρες γλώσσες προγραμματισμού, ικανή να προσαρμοστεί σε διαφορετικά έργα και περιβάλλοντα. [10]

Η Python διακρίνεται για τα πολυάριθμα πλεονεκτήματα που την καθιστούν ιδιαίτερα δημοφιλή στον χώρο του προγραμματισμού. Ένα από τα βασικότερα πλεονεκτήματα της Python είναι η απλότητα και η αναγνωσιμότητα του συντακτικού της. Σε αντίθεση με άλλες γλώσσες προγραμματισμού, η Python χρησιμοποιεί μια καθαρή και ευανάγνωστη σύνταξη που μοιάζει περισσότερο με φυσική γλώσσα, γεγονός που διευκολύνει την εκμάθηση και την κατανόησή της από νέους προγραμματιστές. Αυτή η απλότητα μειώνει τον χρόνο ανάπτυξης λογισμικού, επιτρέποντας στους προγραμματιστές να επικεντρώνονται στην επίλυση προβλημάτων αντί για τις τεχνικές λεπτομέρειες του κώδικα.

Ένα ακόμη πλεονέκτημα της Python είναι η ευρεία υποστήριξη βιβλιοθηκών και πλαισίων, που επιταχύνουν την ανάπτυξη εφαρμογών σε πολλούς τομείς, όπως η ανάπτυξη ιστοσελίδων, η επιστήμη δεδομένων, η μηχανική μάθηση, και ο αυτοματισμός. Αυτές οι βιβλιοθήκες προσφέρουν έτοιμες λύσεις για κοινά προβλήματα προγραμματισμού, επιτρέποντας στους προγραμματιστές να αναπτύσσουν σύνθετες εφαρμογές με λιγότερο κώδικα και μεγαλύτερη ακρίβεια. Επιπλέον, η μεγάλη κοινότητα προγραμματιστών της Python εξασφαλίζει ότι υπάρχουν πάντα διαθέσιμοι πόροι, παραδείγματα κώδικα, και υποστήριξη για την αντιμετώπιση τυχόν προγραμματιστικών προβλημάτων.

Η Python είναι επίσης μια γλώσσα που προσφέρει εξαιρετική διαλειτουργικότητα και φορητότητα. Μπορεί να εκτελεστεί σε διαφορετικά λειτουργικά συστήματα, όπως Windows, macOS, και Linux, χωρίς να απαιτούνται σημαντικές τροποποιήσεις στον κώδικα. Επιπλέον, η Python είναι ερμηνευόμενη γλώσσα, κάτι που σημαίνει ότι οι προγραμματιστές μπορούν να εκτελούν και να δοκιμάζουν τον κώδικα άμεσα, χωρίς την ανάγκη προηγούμενης μεταγλώττισης. Αυτό διευκολύνει τον εντοπισμό και τη διόρθωση σφαλμάτων κατά την ανάπτυξη λογισμικού, μειώνοντας τον χρόνο και το κόστος παραγωγής. [11] [12]

3.1.1 Python Server

Η Python έχει καθιερωθεί ως μια από τις πιο δημοφιλείς γλώσσες για την ανάπτυξη server-side εφαρμογών, χάρη στην ευελιξία και την απλότητά της. Οι Python servers χρησιμοποιούνται ευρέως για την ανάπτυξη web εφαρμογών, την εξυπηρέτηση API, και την διαχείριση δεδομένων σε πραγματικό χρόνο. Η γλώσσα προσφέρει μια πληθώρα επιλογών για την ανάπτυξη servers, από ελαφριές λύσεις όπως το Flask, μέχρι πιο πλήρη frameworks όπως το Django, καλύπτοντας έτσι τις ανάγκες διαφορετικών ειδών εφαρμογών.

Το Flask, ένα από τα πιο δημοφιλή micro-frameworks στην Python, επιτρέπει στους προγραμματιστές να δημιουργούν γρήγορα και εύκολα ελαφριές και αποδοτικές web εφαρμογές. Παρέχει βασικές λειτουργίες, όπως η διαχείριση αιτημάτων HTTP και η δρομολόγηση URL, δίνοντας παράλληλα την ευελιξία στον προγραμματιστή να επιλέξει τα εργαλεία και τις βιβλιοθήκες που χρειάζεται για την εκάστοτε εφαρμογή. Αυτό καθιστά το Flask ιδανικό για την ανάπτυξη μικρών και μεσαίων έργων ή την υλοποίηση APIs.

Από την άλλη πλευρά, το Django είναι ένα πιο ολοκληρωμένο framework, σχεδιασμένο για την ανάπτυξη μεγάλων και πολύπλοκων εφαρμογών. Παρέχει πλήθος ενσωματωμένων εργαλείων, όπως σύστημα διαχείρισης βάσεων δεδομένων, διαχείριση χρηστών και sessions, καθώς και δυνατότητες

ασφαλείας. Το Django ενσωματώνει επίσης το πρότυπο σχεδιασμού Model-View-Controller (MVC), που διευκολύνει την ανάπτυξη εφαρμογών με ξεκάθαρη δομή και διαχωρισμό των λειτουργιών.

Οι Python servers δεν περιορίζονται μόνο σε web εφαρμογές. Η γλώσσα χρησιμοποιείται επίσης για την ανάπτυξη servers που υποστηρίζουν επικοινωνία σε πραγματικό χρόνο μέσω πρωτοκόλλων όπως τα WebSockets. Βιβλιοθήκες όπως το asyncio και το websockets επιτρέπουν την υλοποίηση ασύγχρονων servers, οι οποίοι μπορούν να διαχειρίζονται μεγάλο όγκο ταυτόχρονων συνδέσεων χωρίς σημαντική επιβάρυνση των πόρων του συστήματος. Αυτό καθιστά την Python ιδανική για την ανάπτυξη servers που εξυπηρετούν εφαρμογές όπως chat εφαρμογές, online παιχνίδια, και IoT συστήματα.

HTTP Server

Η Python είναι εξαιρετικά δημοφιλής για την υλοποίηση HTTP servers, λόγω της απλότητας και της ευελιξίας που προσφέρει. Ένας HTTP server στην Python μπορεί να αναπτυχθεί εύκολα χρησιμοποιώντας τις ενσωματωμένες βιβλιοθήκες της γλώσσας, όπως η http.server, που προσφέρει βασική λειτουργικότητα για τη διαχείριση αιτημάτων HTTP. Η βιβλιοθήκη αυτή επιτρέπει τη δημιουργία απλών HTTP servers, οι οποίοι μπορούν να εξυπηρετούν στατικές σελίδες, να διαχειρίζονται αρχεία και να απαντούν σε βασικά HTTP αιτήματα, όπως GET και POST.

Για πιο σύνθετες εφαρμογές, όπως δυναμικές ιστοσελίδες και APIs, η Python παρέχει διάφορα frameworks όπως το Flask και το Django, τα οποία επεκτείνουν τις δυνατότητες ενός HTTP server. Αυτά τα frameworks επιτρέπουν τη δημιουργία εφαρμογών με πολλές δυνατότητες, όπως διαχείριση χρηστών, σύνθετη δρομολόγηση αιτημάτων και αλληλεπίδραση με βάσεις δεδομένων. Ένα από τα βασικά πλεονεκτήματα της χρήσης της Python για την ανάπτυξη HTTP servers είναι η ταχεία ανάπτυξη και ευκολία στη συντήρηση του κώδικα, καθιστώντας την γλώσσα ιδανική για startups και επιχειρήσεις που χρειάζονται ευέλικτες λύσεις. [13][14]

Ένα απλό παράδειγμα για την υλοποίηση ενός βασικού HTTP server χρησιμοποιώντας την ενσωματωμένη βιβλιοθήκη http.server της Python:

```
from http.server import SimpleHTTPRequestHandler, HTTPServer


# Ορίζουμε τη διεύθυνση IP και τη θύρα όπου θα τρέχει ο server
host_name = "localhost"
server_port = 8080
```

```
# Δημιουργούμε μια κλάση χειρισμού αιτημάτων που κληρονομεί από την SimpleHTTPRequestHandler
class MyServer(SimpleHTTPRequestHandler):
    def do_GET(self):
        # Ορίζουμε την απάντηση για τα GET αιτήματα
        self.send_response(200)  # Κωδικός 200 σημαίνει επιτυχία
        self.send_header("Content-type", "text/html")  # Ορίζουμε το τύπο περιεχομένου της απάντησης
        self.end_headers()  # Κλείνουμε τα headers
        # Στέλνουμε το περιεχόμενο της απάντησης
        self.wfile.write(bytes("<html><body><h1>Hello, Python HTTP Server!</h1></body></html>", "utf-8"))

# Δημιουργούμε και ξεκινάμε τον server
if __name__ == "__main__":
    web_server = HTTPServer((host_name, server_port), MyServer)
    print(f"Server started http://{host_name}:{server_port}")

    try:
        web_server.serve_forever()  # Ο server θα τρέχει αδιάκοπα μέχρι να τον σταματήσουμε
    except KeyboardInterrupt:
        pass

    web_server.server_close()  # Κλείνουμε τον server όταν τερματιστεί
    print("Server stopped.")
```

Το παραπάνω παράδειγμα δημιουργεί έναν απλό HTTP server που τρέχει τοπικά στη θύρα 8080.

Όταν κάποιος επισκέπτεται τη διεύθυνση http://localhost:8080, ο server απαντά με ένα απλό HTML μήνυμα "Hello, Python HTTP Server!".

Η κλάση SimpleHTTPRequestHandler χειρίζεται τα αιτήματα HTTP GET.

Η Python παρέχει ισχυρά εργαλεία για την ανάπτυξη TCP servers, οι οποίοι είναι βασικοί για την υλοποίηση δικτυακών εφαρμογών που απαιτούν σταθερή και αξιόπιστη σύνδεση μεταξύ πελάτη και διακομιστή. Η δημιουργία ενός TCP server στην Python μπορεί να επιτευχθεί χρησιμοποιώντας την ενσωματωμένη βιβλιοθήκη socket, η οποία παρέχει ένα χαμηλού επιπέδου API για την επικοινωνία

μέσω δικτύου. Μέσω της βιβλιοθήκης αυτής, ένας TCP server μπορεί να δεχτεί συνδέσεις από πολλαπλούς clients, να διαχειριστεί την ανταλλαγή δεδομένων, και να διατηρήσει συνεχή επικοινωνία με τους πελάτες του.

Για την υλοποίηση πιο πολύπλοκων TCP servers, που πρέπει να διαχειριστούν μεγάλο όγκο ταυτόχρονων συνδέσεων, μπορούν να χρησιμοποιηθούν βιβλιοθήκες όπως το asyncio, η οποία υποστηρίζει ασύγχρονες λειτουργίες. Με τη χρήση αυτής της βιβλιοθήκης, ένας TCP server μπορεί να εξυπηρετήσει πολλές συνδέσεις χωρίς να μπλοκάρει, βελτιώνοντας την απόδοση και μειώνοντας τη χρήση πόρων. Αυτή η προσέγγιση είναι ιδιαίτερα χρήσιμη σε εφαρμογές όπως διαδικτυακά παιχνίδια, συστήματα ανταλλαγής μηνυμάτων και IoT εφαρμογές, όπου η ταχύτητα και η αποδοτικότητα της δικτυακής επικοινωνίας είναι κρίσιμες.

[Python.org. (n.d.). socket — Low-level networking interface. Retrieved from Python.org]

[Real Python. (n.d.). Asynchronous Programming with Python asyncio. Retrieved from Real Python]

Ένα παράδειγμα για την υλοποίηση ενός απλού TCP server χρησιμοποιώντας την ενσωματωμένη βιβλιοθήκη socket της Python:

```
import socket


# Ορίζουμε τη διεύθυνση IP και τη θύρα όπου θα τρέχει ο server
host = "localhost"
port = 65432


# Δημιουργούμε ένα socket και το δεσμεύουμε στη διεύθυνση και τη θύρα
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
    server_socket.bind((host, port))
    server_socket.listen()  # Ο server αρχίζει να ακούει για εισερχόμενες συνδέσεις
    print(f"Server listening on {host}:{port}")


    while True:
        client_socket, client_address = server_socket.accept()  # Αποδέχεται μια σύνδεση
        with client_socket:
            print(f"Connected by {client_address}")
            while True:
                data = client_socket.recv(1024)  # Λαμβάνει δεδομένα από τον client
                if not data:
```

```
            break  # Αν δεν υπάρχουν δεδομένα, κλείνει η σύνδεση
        client_socket.sendall(data)  # Στέλνει πίσω τα δεδομένα στον client (echo)
```

Το παράδειγμα δημιουργεί έναν απλό TCP server που ακούει στη θύρα 65432.

Όταν κάποιος client συνδεθεί, ο server δέχεται δεδομένα από τον client και τα επιστρέφει πίσω (echo server).

Ο server παραμένει ενεργός, αναμένοντας νέες συνδέσεις, μέχρι να τον σταματήσουμε χειροκίνητα.

3.2 Websockets

Τα WebSockets είναι μια σύγχρονη τεχνολογία που επιτρέπει την αμφίδρομη επικοινωνία σε πραγματικό χρόνο μεταξύ ενός προγράμματος περιήγησης (ή άλλου client) και ενός server, χρησιμοποιώντας ένα και μόνο σύνδεσμο TCP. Σε αντίθεση με τα παραδοσιακά πρωτόκολλα HTTP, τα οποία λειτουργούν με βάση το μοντέλο αιτήματος-απόκρισης, τα WebSockets επιτρέπουν τόσο στον client όσο και στον server να στέλνουν δεδομένα οποιαδήποτε στιγμή, χωρίς την ανάγκη για νέα αιτήματα από τον client.

Η τεχνολογία αυτή εισήχθη για να ξεπεραστούν οι περιορισμοί των προηγούμενων τεχνικών για την αμφίδρομη επικοινωνία, όπως το polling και το long polling, που είτε σπαταλούν πόρους είτε δεν παρέχουν πραγματικά πραγματικό χρόνο αλληλεπίδρασης. Τα WebSockets παρέχουν έναν πιο αποδοτικό τρόπο επικοινωνίας, διατηρώντας μια ανοιχτή σύνδεση μεταξύ του client και του server, μέσω της οποίας μπορούν να μεταδίδονται δεδομένα συνεχώς και χωρίς καθυστέρηση.

Ένας από τους βασικούς τομείς εφαρμογής των WebSockets είναι οι εφαρμογές όπου η ταχύτητα και η αμεσότητα της επικοινωνίας είναι κρίσιμες. Παραδείγματα περιλαμβάνουν chat εφαρμογές, διαδικτυακά παιχνίδια, συστήματα παρακολούθησης σε πραγματικό χρόνο, και το Διαδίκτυο των Πραγμάτων (IoT). Με τα WebSockets, οι εφαρμογές αυτές μπορούν να επιτύχουν μια πιο φυσική και αδιάκοπη ροή δεδομένων, βελτιώνοντας τη συνολική εμπειρία χρήστη.

Η υλοποίηση των WebSockets είναι σχετικά απλή και υποστηρίζεται από τους περισσότερους σύγχρονους web browsers και servers. Η επικοινωνία αρχίζει με ένα αρχικό αίτημα HTTP, το οποίο αναβαθμίζεται σε σύνδεση WebSocket μέσω μιας ειδικής διαδικασίας χειραψίας (handshake). Μόλις η σύνδεση είναι ανοιχτή, τα δεδομένα μπορούν να μεταδίδονται με πολύ χαμηλή καθυστέρηση, επιτρέποντας την άμεση ανταλλαγή πληροφοριών.

Τα WebSockets παρέχουν επίσης εξαιρετική επεκτασιμότητα, καθιστώντας τα κατάλληλα για εφαρμογές που απαιτούν την ταυτόχρονη σύνδεση μεγάλου αριθμού clients. Οι server μπορούν να διαχειριστούν πολλές συνδέσεις χωρίς σημαντική επιβάρυνση, κάνοντας τα WebSockets ιδανικά για σύγχρονες διαδικτυακές υπηρεσίες που απαιτούν υψηλή απόδοση και αξιοπιστία. [15]

Τα WebSockets προσφέρουν αρκετά πλεονεκτήματα σε σχέση με τα παραδοσιακά πρωτόκολλα επικοινωνίας, καθιστώντας τα ιδανικά για συγκεκριμένες εφαρμογές που απαιτούν αμφίδρομη επικοινωνία σε πραγματικό χρόνο.

1. **Αμφίδρομη Επικοινωνία σε Πραγματικό Χρόνο**: Ένα από τα μεγαλύτερα πλεονεκτήματα των WebSockets είναι η δυνατότητα για αμφίδρομη επικοινωνία. Τόσο ο client όσο και ο server μπορούν να στέλνουν και να λαμβάνουν δεδομένα οποιαδήποτε στιγμή, χωρίς να χρειάζεται να περιμένουν κάποιο αίτημα από την άλλη πλευρά.
2. **Μειωμένη Καθυστέρηση**: Οι WebSocket συνδέσεις διατηρούνται ανοιχτές, επιτρέποντας την άμεση μεταφορά δεδομένων. Αυτό μειώνει σημαντικά την καθυστέρηση σε σύγκριση με τις παραδοσιακές HTTP συνδέσεις, όπου απαιτείται να δημιουργηθεί νέα σύνδεση για κάθε αίτημα-απάντηση.
3. **Μειωμένη Χρήση Πόρων**: Επειδή τα WebSockets διατηρούν μια συνεχή σύνδεση, η ανάγκη για συνεχείς αιτήσεις HTTP εξαλείφεται, με αποτέλεσμα να μειώνεται η χρήση πόρων τόσο στον server όσο και στον client. Αυτό επιτρέπει την εξυπηρέτηση μεγαλύτερου αριθμού ταυτόχρονων συνδέσεων.
4. **Απλότητα Υλοποίησης**: Παρόλο που οι WebSockets προσφέρουν προηγμένη λειτουργικότητα, η υλοποίησή τους είναι σχετικά απλή και υποστηρίζεται από τους περισσότερους σύγχρονους web browsers και servers. Αυτό διευκολύνει την ενσωμάτωση WebSockets σε υπάρχουσες εφαρμογές.
5. **Καλύτερη Επεκτασιμότητα**: Οι WebSocket servers μπορούν να διαχειριστούν μεγάλες ποσότητες ταυτόχρονων συνδέσεων με αποδοτικό τρόπο, καθιστώντας τους κατάλληλους για εφαρμογές που απαιτούν υψηλή επεκτασιμότητα.

Χρήσεις

Τα WebSockets βρίσκουν εφαρμογή σε πολλούς τομείς όπου η γρήγορη και αμφίδρομη επικοινωνία είναι απαραίτητη.

1. **Chat Εφαρμογές**: Πλατφόρμες ανταλλαγής μηνυμάτων σε πραγματικό χρόνο, όπως το Slack και το Discord, χρησιμοποιούν WebSockets για να επιτρέπουν άμεση επικοινωνία μεταξύ των χρηστών. Η τεχνολογία επιτρέπει την αποστολή και λήψη μηνυμάτων χωρίς καθυστέρηση, προσφέροντας μια ομαλή εμπειρία χρήστη.
2. **Online Παιχνίδια**: Στα διαδικτυακά παιχνίδια, όπου οι παίκτες πρέπει να επικοινωνούν και να αλληλεπιδρούν σε πραγματικό χρόνο, τα WebSockets χρησιμοποιούνται για τη διαχείριση αυτών των συνδέσεων. Εφαρμογές όπως το Agar.io και το Fortnite χρησιμοποιούν WebSockets για να εξασφαλίσουν ότι τα δεδομένα μεταδίδονται άμεσα μεταξύ των παικτών και των servers.

3. **Συστήματα Παρακολούθησης σε Πραγματικό Χρόνο**: Τα WebSockets χρησιμοποιούνται σε εφαρμογές παρακολούθησης, όπως συστήματα παρακολούθησης αποθεμάτων ή τιμών χρηματιστηρίου, όπου τα δεδομένα πρέπει να ανανεώνονται άμεσα και συνεχώς για να αντανακλούν τις αλλαγές σε πραγματικό χρόνο.
4. **Εφαρμογές Διαδικτύου των Πραγμάτων (IoT)**: Στο πεδίο του IoT, τα WebSockets επιτρέπουν την αδιάλειπτη επικοινωνία μεταξύ των συσκευών και των κεντρικών συστημάτων ελέγχου. Αυτό επιτρέπει την άμεση ανταλλαγή δεδομένων και εντολών, βελτιώνοντας την απόδοση και την απόκριση των IoT συστημάτων.
5. **Συνεργατικές Εφαρμογές**: Εφαρμογές που επιτρέπουν την ταυτόχρονη εργασία πολλών χρηστών, όπως τα Google Docs ή άλλες πλατφόρμες online επεξεργασίας, χρησιμοποιούν WebSockets για να συγχρονίζουν τις αλλαγές σε πραγματικό χρόνο.

3.3 Python+Websockets

Η Python υποστηρίζει την υλοποίηση WebSocket servers μέσω διαφόρων βιβλιοθηκών, με την πιο διαδεδομένη να είναι η websockets βιβλιοθήκη. Αυτή η βιβλιοθήκη προσφέρει ένα απλό και αποδοτικό τρόπο για την ανάπτυξη WebSocket servers που μπορούν να εξυπηρετούν μεγάλο αριθμό ταυτόχρονων συνδέσεων.

```
import asyncio
import websockets


async def echo(websocket, path):
    async for message in websocket:
        await websocket.send(f"Received: {message}")


# Ο server τρέχει στο localhost στη θύρα 8765
start_server = websockets.serve(echo, "localhost", 8765)


asyncio.get_event_loop().run_until_complete(start_server)
asyncio.get_event_loop().run_forever()
```

Αυτό το παράδειγμα δημιουργεί έναν απλό WebSocket server που ακούει για μηνύματα από τους clients και απαντά με ένα μήνυμα επιβεβαίωσης ("Received:").

Ο server τρέχει ασύγχρονα, χρησιμοποιώντας τη βιβλιοθήκη asyncio, η οποία επιτρέπει την εξυπηρέτηση πολλών ταυτόχρονων συνδέσεων χωρίς μπλοκάρισμα των πόρων του συστήματος.

Εκτός από servers, η Python μπορεί επίσης να χρησιμοποιηθεί για την υλοποίηση WebSocket clients, οι οποίοι συνδέονται σε έναν WebSocket server και επικοινωνούν μαζί του σε πραγματικό χρόνο. Χρησιμοποιώντας και πάλι τη βιβλιοθήκη websockets, ένας client μπορεί να στείλει και να λάβει δεδομένα από έναν server.

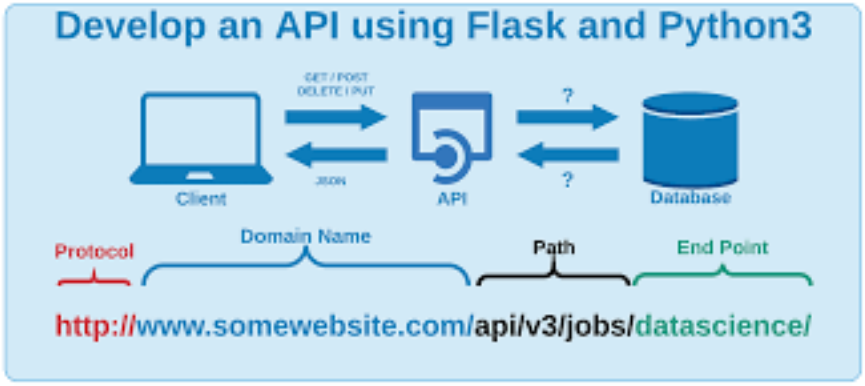
```
import asyncio
import websockets


async def hello():
    uri = "ws://localhost:8765"
    async with websockets.connect(uri) as websocket:
        await websocket.send("Hello, Server!")
        response = await websocket.recv()
        print(f"Response from server: {response}")


asyncio.get_event_loop().run_until_complete(hello())
```

Ο παραπάνω client συνδέεται σε έναν WebSocket server που τρέχει στο localhost στη θύρα 8765, στέλνει ένα μήνυμα ("Hello, Server!") και περιμένει την απάντηση του server.

Το μήνυμα που λαμβάνει από τον server εκτυπώνεται στην κονσόλα. [16]

3.4 Flask

Το Flask είναι ένα μικρο-πλαίσιο (micro-framework) για την ανάπτυξη web εφαρμογών με την Python. Δημιουργήθηκε από τον Armin Ronacher το 2010 και έκτοτε έχει γίνει ένα από τα πιο δημοφιλή εργαλεία για την ανάπτυξη web εφαρμογών λόγω της ευελιξίας και της απλότητας που προσφέρει. Το Flask ακολουθεί τη φιλοσοφία "ελάχιστου πυρήνα, με δυνατότητα επέκτασης", παρέχοντας έναν ελαφρύ και λιτό πυρήνα με βασικές λειτουργίες, ενώ επιτρέπει στους προγραμματιστές να προσθέσουν πρόσθετα χαρακτηριστικά και βιβλιοθήκες ανάλογα με τις ανάγκες του έργου.



Εικόνα 3.2: Flask

[https://miro.medium.com/v2/resize:fit:1400/1*ZU1EQ7tYeQNQlhOhyonHFA.png]

Το Flask χρησιμοποιείται ευρέως για την ανάπτυξη διαφόρων ειδών web εφαρμογών, από απλές ιστοσελίδες μέχρι πιο σύνθετα API και μικρο-υπηρεσίες (microservices).

Το Flask είναι ιδανικό για την ανάπτυξη RESTful APIs λόγω της ευελιξίας του. Με την προσθήκη επεκτάσεων, όπως το Flask-RESTful, οι προγραμματιστές μπορούν να δημιουργήσουν ισχυρά API endpoints με εύκολο και αποδοτικό τρόπο. Η απλότητα του Flask το καθιστά εξαιρετική επιλογή για την ανάπτυξη πρωτοτύπων (prototypes) και MVPs (Minimum Viable Products). Οι προγραμματιστές μπορούν γρήγορα να στήσουν μια λειτουργική εφαρμογή και να τη βελτιώσουν σταδιακά καθώς το έργο εξελίσσεται. Το Flask μπορεί να χρησιμοποιηθεί για τη δημιουργία δυναμικών ιστοσελίδων με βάση δεδομένων, αξιοποιώντας τεχνολογίες όπως το Jinja2 για την απόδοση προτύπων HTML και το SQLAlchemy για τη διαχείριση βάσεων δεδομένων. Το ελαφρύ και επεκτάσιμο μοντέλο του Flask είναι ιδανικό για την ανάπτυξη μικρο-υπηρεσιών (microservices), όπου κάθε υπηρεσία μπορεί να λειτουργεί ανεξάρτητα και να επικοινωνεί με άλλες υπηρεσίες μέσω API.

Το Flask προσφέρει πολλά πλεονεκτήματα, αλλά όπως κάθε εργαλείο, έχει και ορισμένα μειονεκτήματα που πρέπει να ληφθούν υπόψη κατά την επιλογή του για ένα έργο.

Το Flask είναι ιδιαίτερα ευέλικτο και δεν επιβάλλει αυστηρή δομή ή περιορισμούς. Αυτό επιτρέπει στους προγραμματιστές να προσαρμόσουν την εφαρμογή τους όπως ακριβώς επιθυμούν. Με έναν μικρό και εύκολα κατανοητό πυρήνα, το Flask είναι ιδανικό για όσους ξεκινούν με την ανάπτυξη web εφαρμογών, καθώς προσφέρει μια ήπια καμπύλη μάθησης.

Παρόλο που το Flask ξεκινά με έναν ελαφρύ πυρήνα, μπορεί να επεκταθεί με τη χρήση διαφόρων επεκτάσεων, καλύπτοντας έτσι τις ανάγκες για πρόσθετη λειτουργικότητα χωρίς περιττή πολυπλοκότητα. Το Flask έχει μια ενεργή και μεγάλη κοινότητα, με πληθώρα διαθέσιμων πόρων, επεκτάσεων και υποστήριξης, που διευκολύνει την ανάπτυξη και επίλυση προβλημάτων.

Η ευελιξία του Flask μπορεί να αποδειχθεί μειονέκτημα για μεγάλες ομάδες ή σύνθετα έργα, όπου η έλλειψη αυστηρής δομής μπορεί να οδηγήσει σε ασυνέπεια και δυσκολίες στη συντήρηση. Σε σύγκριση με πιο ολοκληρωμένα frameworks, όπως το Django, το Flask δεν προσφέρει έτοιμες λύσεις για όλες τις ανάγκες (π.χ. ενσωματωμένη διαχείριση χρηστών ή διοικητικό πάνελ).

Για πολλές βασικές λειτουργίες, οι προγραμματιστές πρέπει να προσθέσουν επεκτάσεις, κάτι που μπορεί να οδηγήσει σε μεγαλύτερη πολυπλοκότητα και πιθανές ασυμβατότητες μεταξύ των επεκτάσεων.

Ένα απλό παράδειγμα εφαρμογής web με το Flask:

```
from flask import Flask, jsonify, request


app = Flask(__name__)


# Διαδρομή για την αρχική σελίδα
@app.route('/')
def home():
    return "<h1>Hello, Flask!</h1>"


# Διαδρομή για ένα API endpoint που επιστρέφει δεδομένα JSON
@app.route('/api/data', methods=['GET'])
def get_data():
    data = {"message": "Hello, World!", "status": "success"}
    return jsonify(data)


# Διαδρομή για ένα API endpoint που δέχεται δεδομένα POST
@app.route('/api/data', methods=['POST'])
def post_data():
    received_data = request.get_json()
    return jsonify({"received": received_data})


# Εκκίνηση του server
if __name__ == '__main__':
    app.run(debug=True)
```

Αυτό το απλό παράδειγμα δημιουργεί μια web εφαρμογή με το Flask που περιλαμβάνει μια αρχική σελίδα και δύο API endpoints.

Το πρώτο endpoint (/api/data) επιστρέφει ένα JSON αντικείμενο όταν λαμβάνει ένα αίτημα GET.

Το δεύτερο endpoint (/api/data) δέχεται δεδομένα σε μορφή JSON μέσω POST και επιστρέφει τα δεδομένα που έλαβε.

Η εφαρμογή τρέχει σε λειτουργία "debug", που επιτρέπει την εύκολη ανάπτυξη και ανίχνευση σφαλμάτων. [17]

3.5 MySQL

Η MySQL είναι ένα από τα πιο δημοφιλή και ευρέως χρησιμοποιούμενα συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS) στον κόσμο. Αναπτύχθηκε αρχικά από τη MySQL AB, μια σουηδική εταιρεία που αργότερα εξαγοράστηκε από την Oracle Corporation. Η MySQL είναι γνωστή για την ταχύτητα, την αξιοπιστία και την ευκολία χρήσης της, καθιστώντας την την πρώτη επιλογή για πολλές εφαρμογές, από μικρές ιστοσελίδες μέχρι μεγάλες επιχειρηματικές λύσεις.

Η MySQL χρησιμοποιείται ευρέως σε διάφορους τομείς και εφαρμογές λόγω της ευελιξίας και της επεκτασιμότητάς της. Η MySQL είναι η προτιμώμενη βάση δεδομένων για πολλές πλατφόρμες ανάπτυξης ιστοσελίδων και συστημάτων διαχείρισης περιεχομένου (CMS) όπως το WordPress, το Joomla, και το Drupal. Χρησιμοποιείται για την αποθήκευση και διαχείριση περιεχομένου, χρηστών, και ρυθμίσεων εφαρμογών. Πλατφόρμες ηλεκτρονικού εμπορίου όπως το Magento και το WooCommerce βασίζονται στη MySQL για τη διαχείριση προϊόντων, παραγγελιών, πελατών και πληρωμών. Η MySQL παρέχει την ταχύτητα και την απόδοση που απαιτείται για την εξυπηρέτηση εκατοντάδων χιλιάδων συναλλαγών ημερησίως. Παρά την απλότητά της, η MySQL μπορεί να χρησιμοποιηθεί σε περιβάλλοντα Big Data σε συνδυασμό με άλλες τεχνολογίες όπως το Hadoop. Παρέχει σταθερότητα και επεκτασιμότητα για την αποθήκευση και ανάλυση μεγάλων όγκων δεδομένων. Πολλές επιχειρήσεις χρησιμοποιούν τη MySQL για τη διαχείριση των δεδομένων τους, συμπεριλαμβανομένων των πελατών, των προμηθευτών, και των προϊόντων. Η ευελιξία της επιτρέπει την προσαρμογή της βάσης δεδομένων σύμφωνα με τις ανάγκες της κάθε επιχείρησης.

Η MySQL προσφέρει πολλά πλεονεκτήματα που την καθιστούν ιδανική για πολλές εφαρμογές, αλλά έχει και ορισμένα μειονεκτήματα που πρέπει να ληφθούν υπόψη.

Η MySQL είναι γνωστή για την ταχύτητά της, ειδικά σε εφαρμογές που απαιτούν γρήγορη ανάκτηση δεδομένων. Η δυνατότητα της MySQL να διαχειρίζεται μεγάλες βάσεις δεδομένων με αποτελεσματικότητα την καθιστά δημοφιλή επιλογή για web εφαρμογές. Είναι εξαιρετικά επεκτάσιμη

και μπορεί να προσαρμοστεί για να διαχειρίζεται μικρές ή μεγάλες εφαρμογές, είτε πρόκειται για ένα μικρό blog είτε για ένα μεγάλο διαδικτυακό κατάστημα με εκατομμύρια χρήστες. Προσφέρει ισχυρά χαρακτηριστικά ασφαλείας που περιλαμβάνουν έλεγχο ταυτότητας χρηστών, δικαιώματα πρόσβασης και υποστήριξη για SSL συνδέσεις, διασφαλίζοντας έτσι την ακεραιότητα και την ασφάλεια των δεδομένων. Η MySQL είναι εύκολη στη μάθηση και χρήση, με σαφή τεκμηρίωση και μια μεγάλη κοινότητα που παρέχει υποστήριξη και παραδείγματα χρήσης. Ενσωματώνεται εύκολα με άλλες γλώσσες προγραμματισμού και τεχνολογίες, όπως PHP, Python, και Java, καθιστώντας την ιδανική επιλογή για πολλαπλά περιβάλλοντα ανάπτυξης.

Αν και η MySQL είναι εξαιρετικά αποδοτική, μπορεί να αντιμετωπίσει προβλήματα απόδοσης όταν οι βάσεις δεδομένων γίνονται εξαιρετικά μεγάλες και οι αιτήσεις πολύ περίπλοκες. Σε τέτοιες περιπτώσεις, μπορεί να απαιτείται βελτιστοποίηση ή χρήση πιο εξειδικευμένων εργαλείων. Σε σύγκριση με άλλα RDBMS όπως το PostgreSQL, η MySQL έχει περιορισμένη υποστήριξη για πολύπλοκα ερωτήματα και συναρτήσεις, κάτι που μπορεί να αποτελέσει πρόβλημα σε εφαρμογές με σύνθετες ανάγκες. Παρόλο που η MySQL είναι ανοιχτού κώδικα, ορισμένες από τις πιο προηγμένες λειτουργίες της είναι διαθέσιμες μόνο στην έκδοση Enterprise, η οποία απαιτεί άδεια χρήσης από την Oracle.

Για μια εφαρμογή Flask με μια βάση δεδομένων MySQL, μπορείτε να χρησιμοποιήσετε τη βιβλιοθήκη mysql-connector-python ή μια ORM βιβλιοθήκη όπως το SQLAlchemy.

Πρώτα, εγκαθιστούμε τη βιβλιοθήκη χρησιμοποιώντας pip:

```
pip install mysql-connector-python
```

Μετά δημιουργούμε την εφαρμογή Flask και συνδεόμαστε με τη βάση δεδομένων:

```
from flask import Flask, jsonify
import mysql.connector

app = Flask(__name__)

# Στοιχεία σύνδεσης με τη βάση δεδομένων MySQL
db_config = {
    'user': 'your_username',
    'password': 'your_password',
    'host': 'localhost',
    'database': 'your_database_name'
}
```

```
# Δημιουργία σύνδεσης με τη βάση δεδομένων
def get_db_connection():
    connection = mysql.connector.connect(**db_config)
    return connection

# Παράδειγμα διαδρομής που ανακτά δεδομένα από τη βάση δεδομένων
@app.route('/users', methods=['GET'])
def get_users():
    connection = get_db_connection()
    cursor = connection.cursor()
    cursor.execute("SELECT id, username FROM users")
    users = cursor.fetchall()
    cursor.close()
    connection.close()

    # Επιστροφή δεδομένων σε μορφή JSON
    return jsonify(users)

# Εκκίνηση του Flask server
if __name__ == '__main__':
    app.run(debug=True)
```

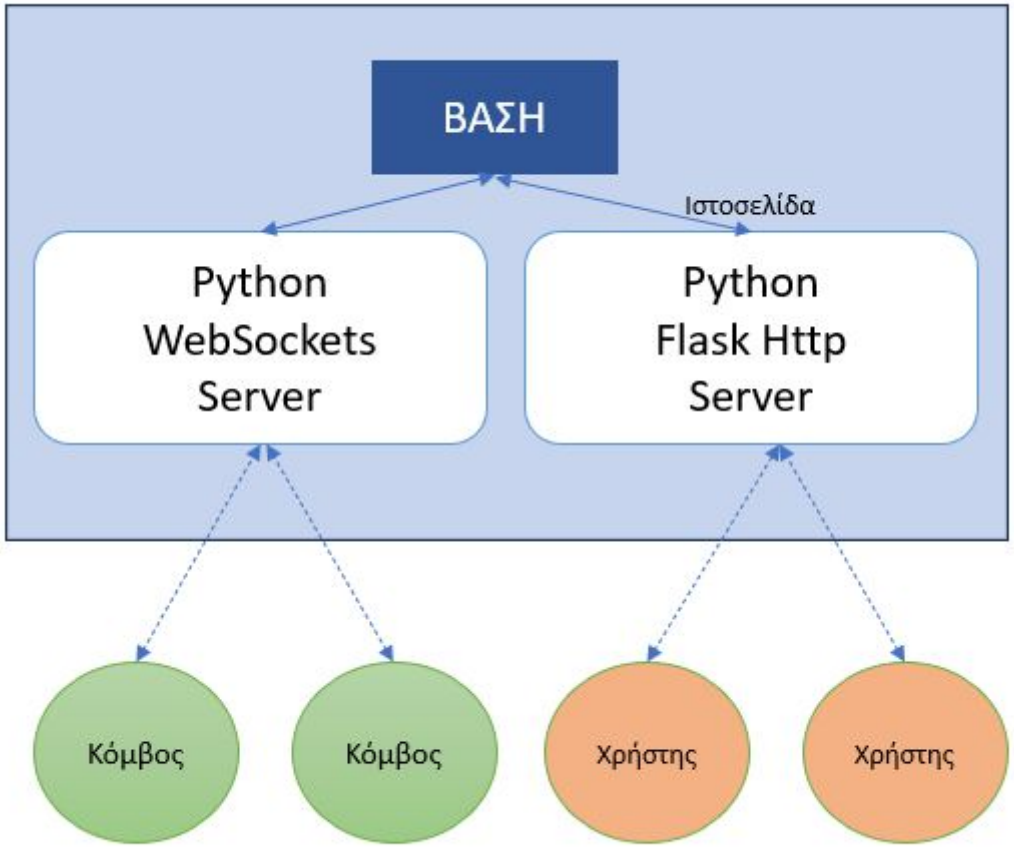
Το παράδειγμα δείχνει πώς να συνδεθείτε σε μια βάση δεδομένων MySQL από μια εφαρμογή Flask και να εκτελέσετε ένα απλό ερώτημα για την ανάκτηση δεδομένων από έναν πίνακα users.

Το get_db_connection() δημιουργεί και επιστρέφει μια σύνδεση με τη βάση δεδομένων, ενώ η διαδρομή /users εκτελεί το ερώτημα και επιστρέφει τα αποτελέσματα σε μορφή JSON.

Ο server τρέχει σε λειτουργία "debug", επιτρέποντας την εύκολη ανίχνευση σφαλμάτων. [18]

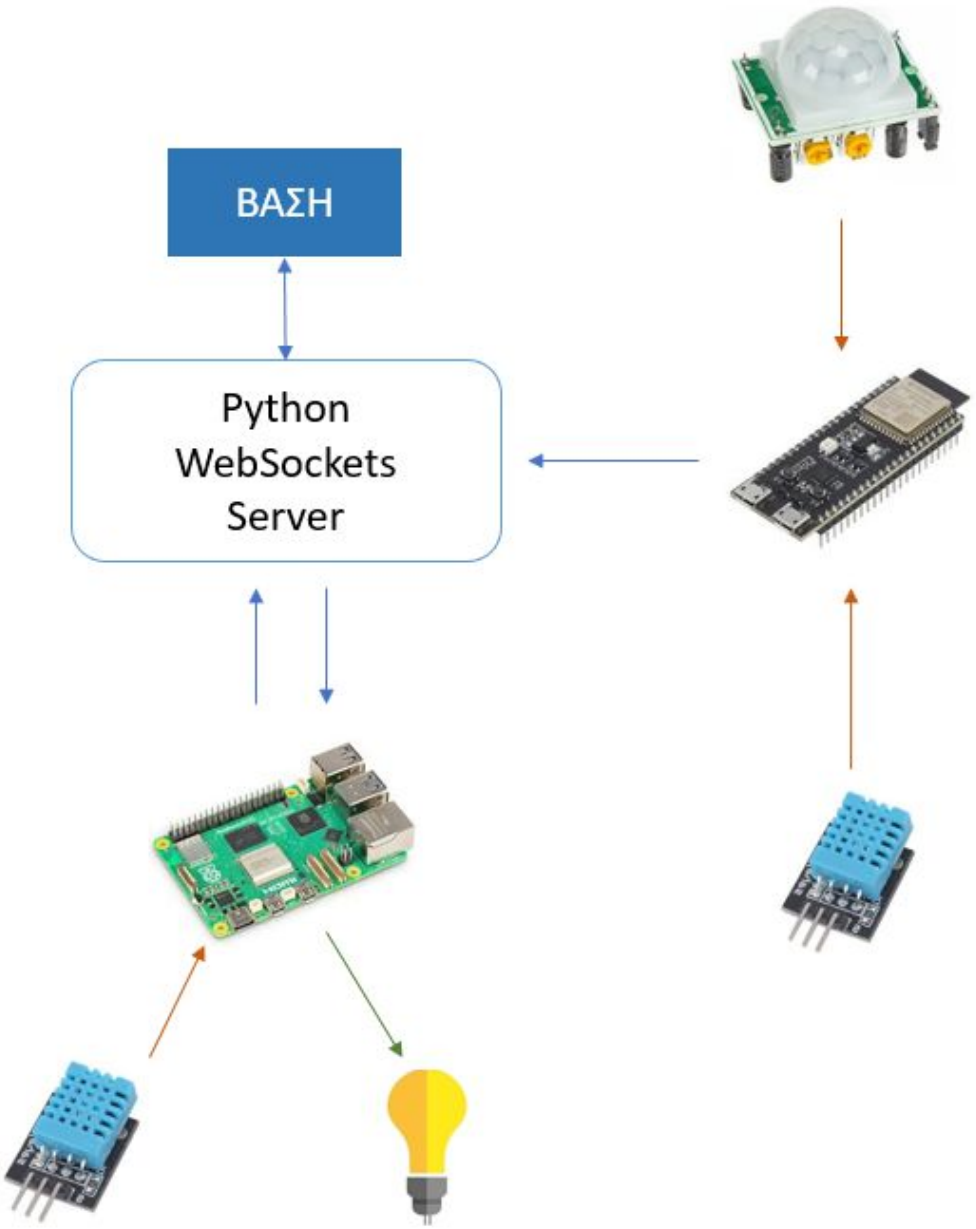
Κεφάλαιο 4ο: Το σύστημα Websocket

4.1 Το σύστημα



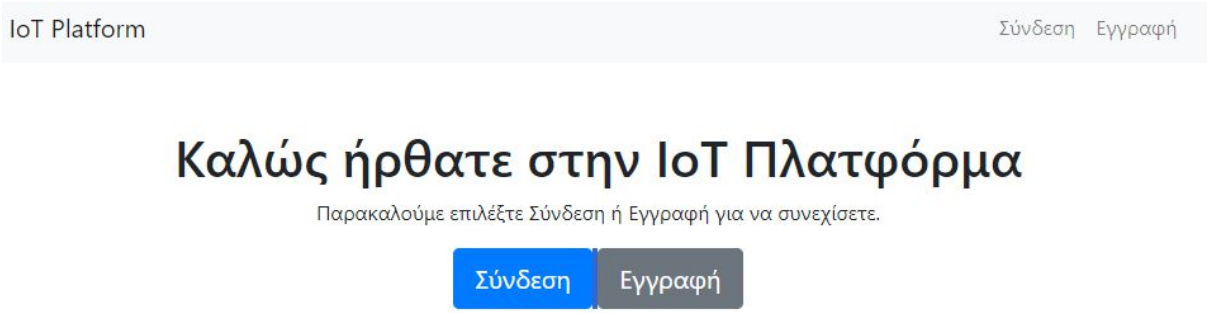
Εικόνα 4.1: Το ολοκληρωμένο σύστημα

Η Εικόνα 4.1 παρουσιάζει το ολοκληρωμένο σύστημα της πλατφόρμας, το οποίο περιλαμβάνει έναν Python WebSockets Server και έναν Python Flask Http Server που επικοινωνούν με τους κόμβους και τους χρήστες αντίστοιχα. Ο WebSockets Server λαμβάνει και στέλνει δεδομένα σε πραγματικό χρόνο από τους κόμβους, ενώ ο Flask Http Server επιτρέπει στους χρήστες να παρακολουθούν και να διαχειρίζονται τους κόμβους μέσω μιας ιστοσελίδας. Η βάση δεδομένων λειτουργεί ως κοινός πόρος για την αποθήκευση και ανάκτηση των δεδομένων.



Εικόνα 4.2: Ο Server δέχεται τιμές από κόμβος ή στέλνει τιμές σε αυτούς

Στην Εικόνα 4.2, βλέπουμε τη διασύνδεση του Python WebSockets Server με διάφορους αισθητήρες και ενεργοποιητές που συνδέονται μέσω κόμβων (π.χ. Raspberry Pi ή ESP32). Ο Server λαμβάνει δεδομένα από τους αισθητήρες (όπως θερμοκρασία ή κίνηση) και μπορεί να στείλει εντολές στους ενεργοποιητές (όπως ενεργοποίηση λαμπτήρων). Αυτό το διάγραμμα αναδεικνύει την αμφίδρομη επικοινωνία μεταξύ των κόμβων και του Server, καθώς και την αλληλεπίδραση με τη βάση δεδομένων.

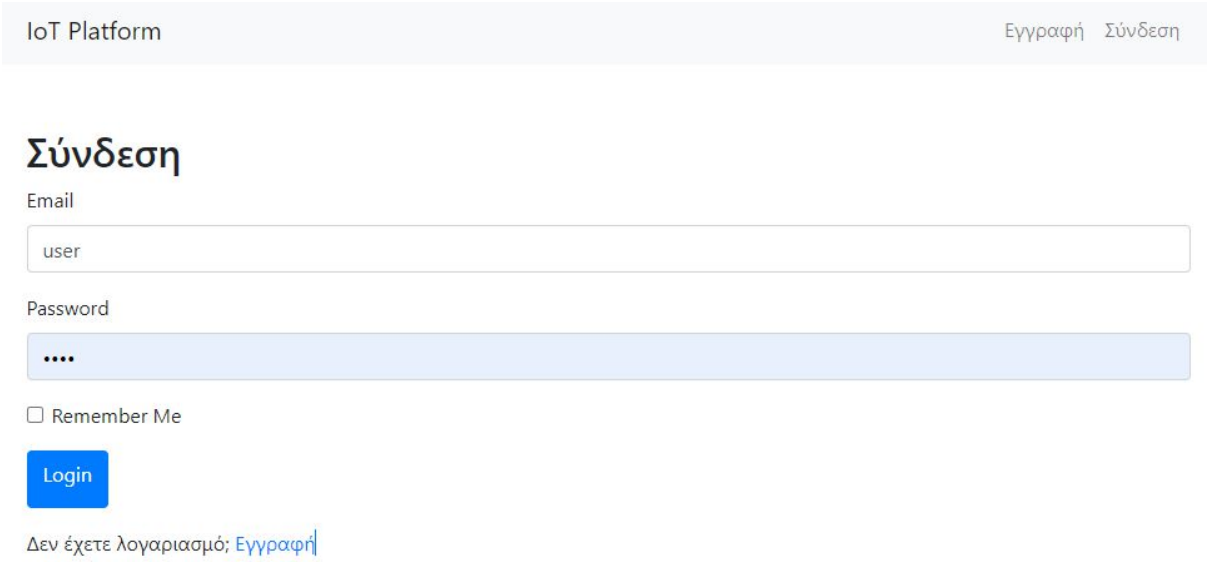


Εικόνα 4.3: Η αρχική της πλατφόρμας

Η αρχική σελίδα της πλατφόρμας αποτελεί το κεντρικό σημείο εισόδου του χρήστη, παρέχοντας επιλογές για εγγραφή νέων χρηστών ή σύνδεση υφιστάμενων χρηστών. Η σελίδα είναι σχεδιασμένη με απλό και φιλικό περιβάλλον, διασφαλίζοντας εύκολη πλοήγηση για όλους τους χρήστες.

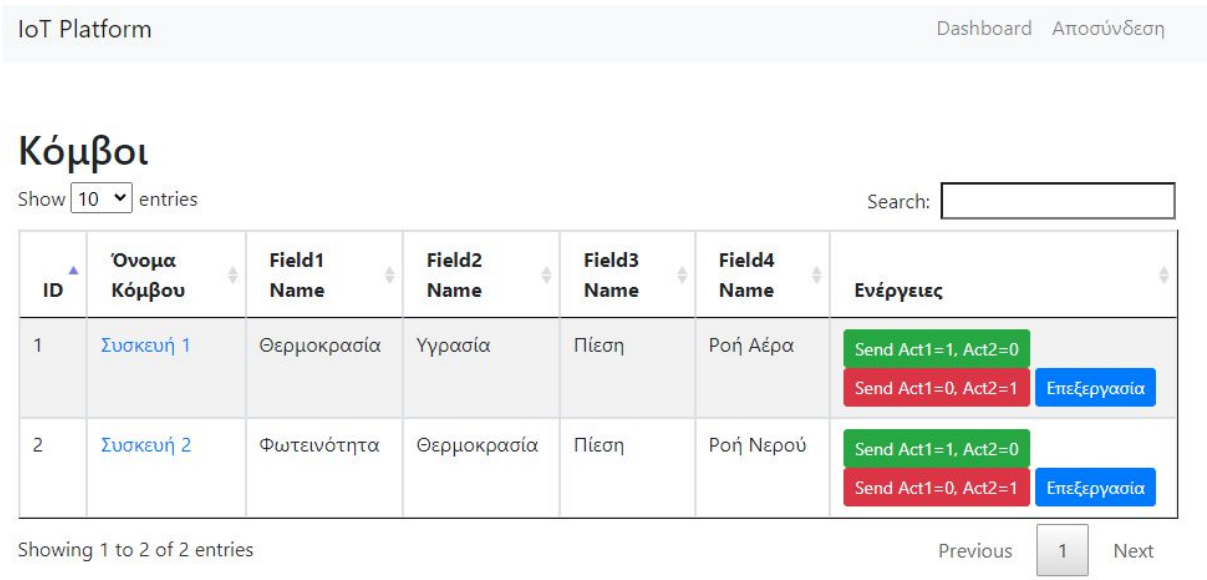
Εικόνα 4.4: Σελίδα για Εγγραφή στη πλατφόρμα

Η αρχική σελίδα της πλατφόρμας αποτελεί το κεντρικό σημείο εισόδου του χρήστη, παρέχοντας επιλογές για εγγραφή νέων χρηστών ή σύνδεση υφιστάμενων χρηστών. Η σελίδα είναι σχεδιασμένη με απλό και φιλικό περιβάλλον, διασφαλίζοντας εύκολη πλοήγηση για όλους τους χρήστες.



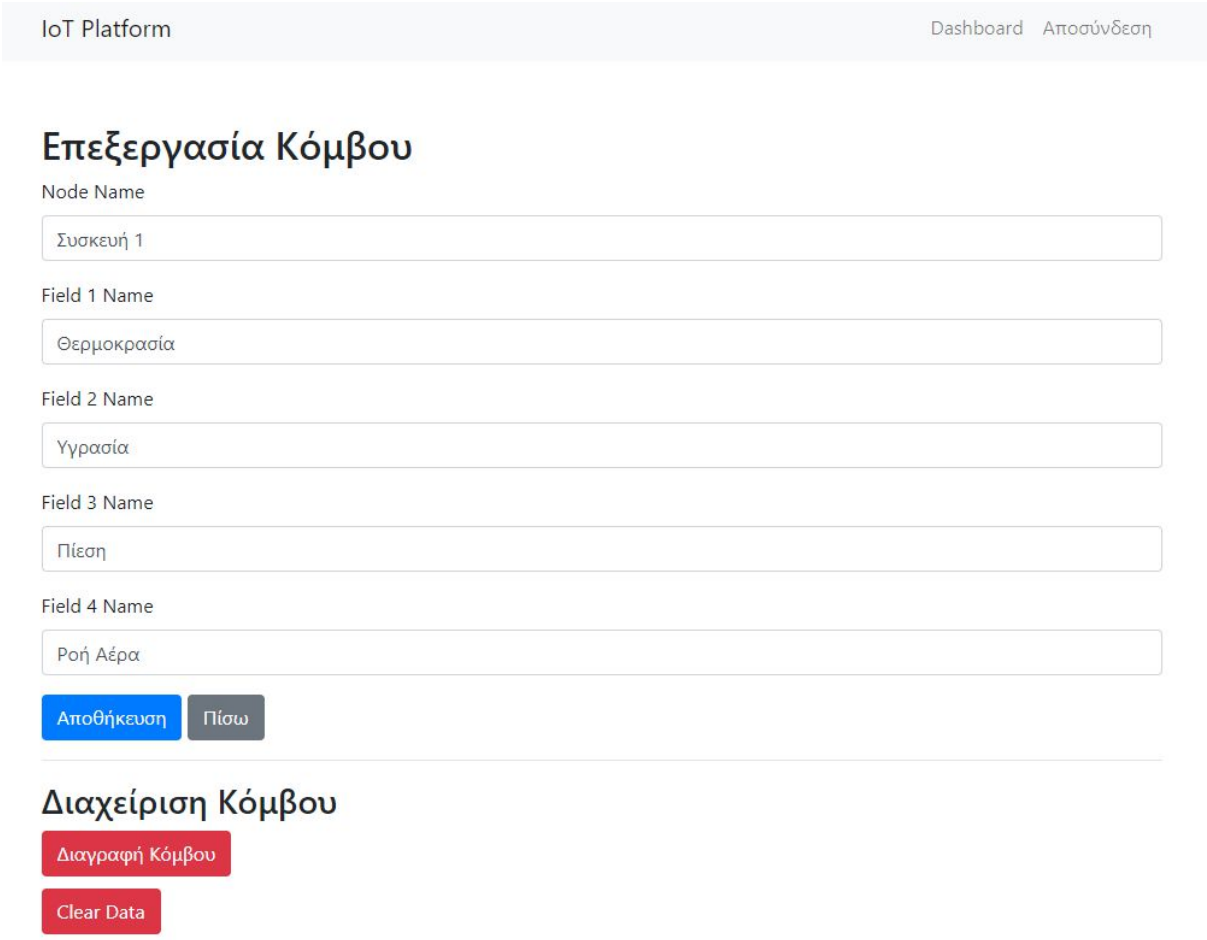
Εικόνα 4.5: Σελίδα για Σύνδεση στη πλατφόρμα

Η σελίδα σύνδεσης επιτρέπει σε υφιστάμενους χρήστες να αποκτήσουν πρόσβαση στον λογαριασμό τους, εισάγοντας τα διαπιστευτήρια τους. Παρέχεται επίσης δυνατότητα ανάκτησης κωδικού σε περίπτωση που ο χρήστης έχει ξεχάσει τον κωδικό πρόσβασης.



Εικόνα 4.6: Σελίδα για προβολή κόμβων με όλα τα χαρακτηριστικά-πεδία και ενέργειες

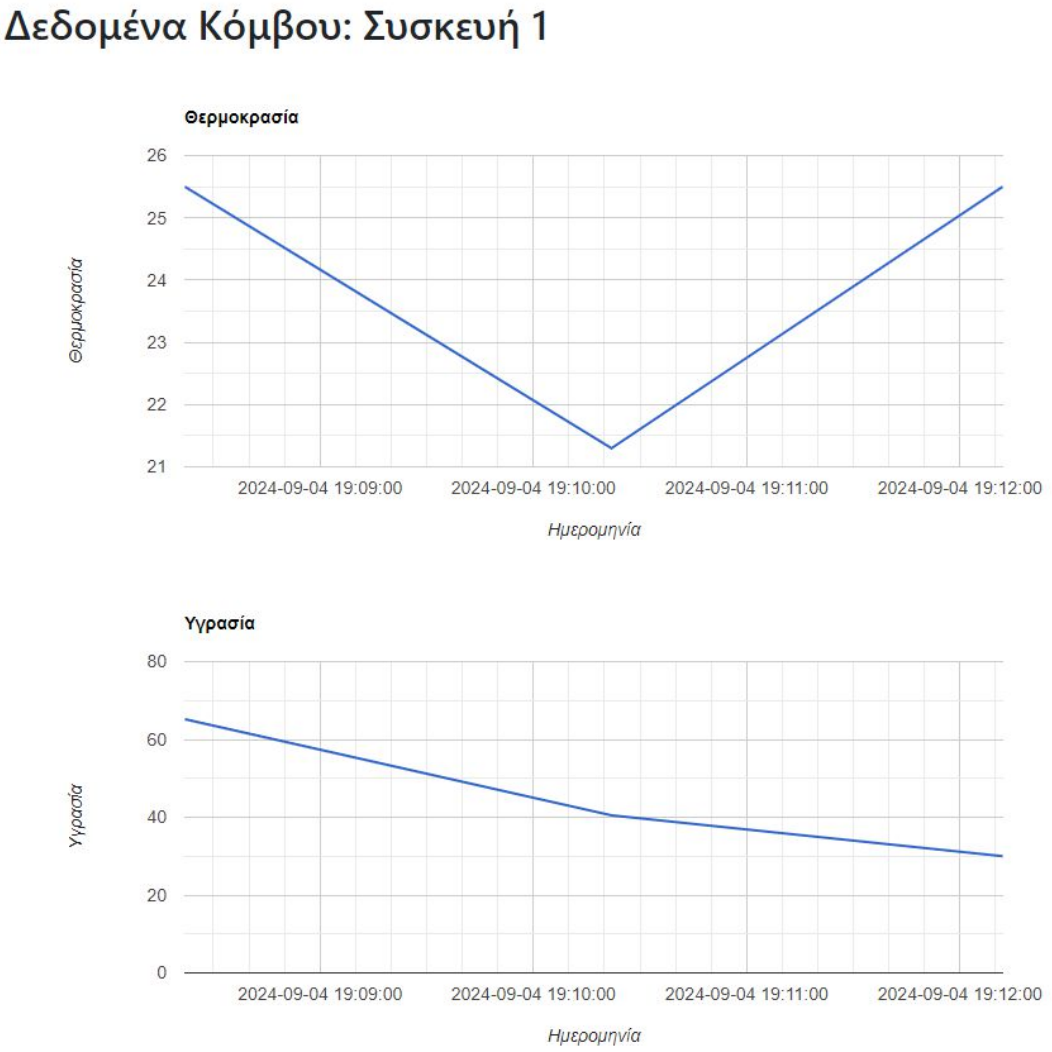
Η σελίδα σύνδεσης επιτρέπει σε υφιστάμενους χρήστες να αποκτήσουν πρόσβαση στον λογαριασμό τους, εισάγοντας τα διαπιστευτήρια τους. Παρέχεται επίσης δυνατότητα ανάκτησης κωδικού σε περίπτωση που ο χρήστης έχει ξεχάσει τον κωδικό πρόσβασης.



Εικόνα 4.7: Σελίδα επεξεργασίας κόμβου

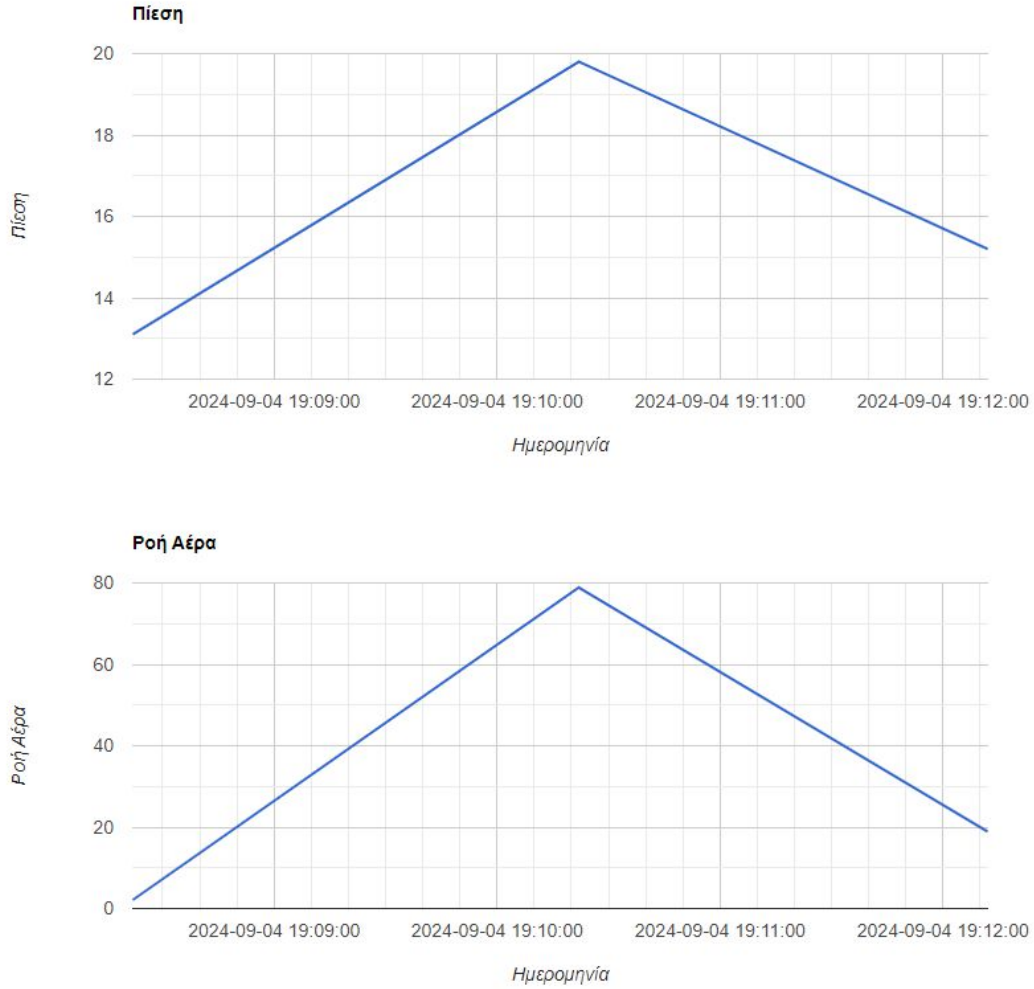
Η σελίδα επεξεργασίας κόμβου επιτρέπει στους χρήστες να τροποποιούν τα χαρακτηριστικά των κόμβων τους. Οι χρήστες μπορούν να ενημερώσουν τα πεδία, να αλλάξουν το όνομα του κόμβου και να διαχειριστούν αισθητήρες που σχετίζονται με τον κόμβο.

Επίσης μπορεί να γίνει διαγραφή κόμβου ή διαγραφή όλων των δεδομένων με επιβεβαίωση.



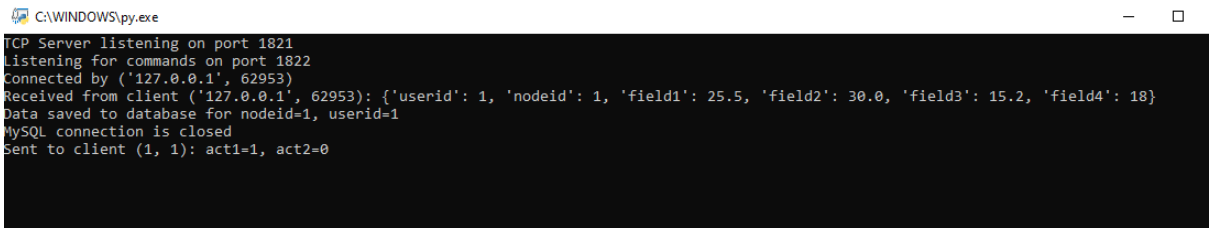
Εικόνα 4.8: Σελίδα προβολής των δύο πρώτων πεδίων του κόμβου σε chart

Αυτή η σελίδα εμφανίζει τα δεδομένα από τα δύο πρώτα πεδία του επιλεγμένου κόμβου σε μορφή γραφήματος. Το γράφημα επιτρέπει στους χρήστες να παρακολουθούν σε πραγματικό χρόνο τις μετρήσεις των αντίστοιχων αισθητήρων ή ενεργοποιητών, διευκολύνοντας την ανάλυση των δεδομένων.



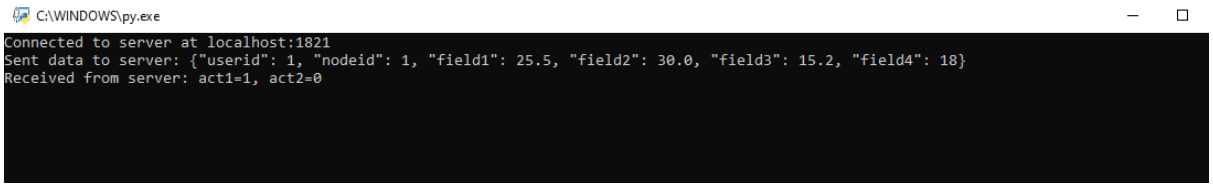
Εικόνα 4.9: Σελίδα προβολής των δύο τελευταίων πεδίων του κόμβου σε chart

Σε αυτή τη σελίδα παρουσιάζονται τα δεδομένα από τα δύο τελευταία πεδία του κόμβου, επίσης με τη μορφή γραφήματος. Όπως και στην προηγούμενη σελίδα, τα δεδομένα απεικονίζονται σε πραγματικό χρόνο, επιτρέποντας στον χρήστη να παρακολουθεί την εξέλιξη των τιμών του κόμβου.



Εικόνα 4.10: Ο Websocket server δέχεται ένα websocket πελάτη ενώ ταυτόχρονα δέχεται και act από τη σελίδα

Η εικόνα αυτή παρουσιάζει τη λειτουργία του WebSocket server, ο οποίος δέχεται δεδομένα από έναν WebSocket πελάτη και ταυτόχρονα λαμβάνει και εντολές act από τη σελίδα. Αυτή η αρχιτεκτονική επιτρέπει την ταυτόχρονη επικοινωνία και την αμφίδρομη μετάδοση δεδομένων σε πραγματικό χρόνο.



Εικόνα 4.11: Websocket πελάτης – κόμβος (πχ esp32) που στέλνει δεδομένα στον Server

Η εικόνα παρουσιάζει έναν WebSocket πελάτη, όπως ένας κόμβος ESP32, ο οποίος συνδέεται με τον server και αποστέλλει δεδομένα αισθητήρων. Η επικοινωνία μέσω WebSocket επιτρέπει τη συνεχή ενημέρωση του server με πραγματικού χρόνου δεδομένα από τον κόμβο, χωρίς καθυστέρηση.

4.2 Websocket server

Παρακάτω είναι ο κώδικας σε Python για έναν WebSocket server που μπορεί να εξυπηρετεί πολλούς χρήστες χρησιμοποιώντας threads.

Ο κώδικας αυτός υλοποιεί έναν TCP WebSocket server που διαχειρίζεται επικοινωνία με κόμβους Internet of Things (IoT) και επιτρέπει την αμφίδρομη μεταφορά δεδομένων σε πραγματικό χρόνο. Οι IoT κόμβοι, όπως οι αισθητήρες και οι ενεργοποιητές, επικοινωνούν με τον server στέλνοντας δεδομένα όπως θερμοκρασία, υγρασία ή άλλες μετρήσεις. Ο server λαμβάνει αυτά τα δεδομένα, τα επεξεργάζεται και τα αποθηκεύει σε μια MySQL βάση δεδομένων, προσφέροντας μια δομημένη και οργανωμένη προσέγγιση για τη διαχείριση των πληροφοριών των κόμβων.

Παράλληλα, ο server μπορεί να λαμβάνει εντολές από έναν Flask HTTP server, ο οποίος χρησιμοποιείται για τη διαχείριση της πλατφόρμας από τους χρήστες. Μέσω του Flask server, οι χρήστες μπορούν να στείλουν εντολές στους IoT κόμβους, όπως η ενεργοποίηση ή απενεργοποίηση μιας συσκευής, οι οποίες στη συνέχεια προωθούνται στους συνδεδεμένους κόμβους μέσω του WebSocket server.

Η αρχιτεκτονική βασίζεται στη χρήση threads, όπου κάθε client (κόμβος) εκτελείται σε ξεχωριστό thread, επιτρέποντας παράλληλη εξυπηρέτηση πολλών κόμβων ταυτόχρονα. Επίσης, χρησιμοποιείται μια δομή που διατηρεί τις συνδέσεις των clients με βάση το nodeid και το userid, εξασφαλίζοντας έτσι άμεση και στοχευμένη επικοινωνία. Αυτή η λύση είναι ιδανική για εφαρμογές IoT όπου απαιτείται επικοινωνία σε πραγματικό χρόνο και δυναμική διαχείριση συσκευών.

```
import socket
import threading
import json
```

```
import pymysql


# Δομή για αποθήκευση των συνδέσεων των clients (με nodeid, userid ως κλειδιά)
clients = {}


# Εξήγηση: Το πρόγραμμα αυτό υλοποιεί έναν TCP server ο οποίος επικοινωνεί με IoT κόμβους μέσω WebSockets.
# Οι κόμβοι στέλνουν δεδομένα στον server, ο οποίος τα αποθηκεύει σε μία MySQL βάση δεδομένων.
# Παράλληλα, ο server ακούει για εντολές από έναν Flask HTTP server, τις οποίες στη συνέχεια στέλνει στους συνδεδεμένους κόμβους.


# Λειτουργία που εξυπηρετεί κάθε client σε ξεχωριστό thread
def handle_client(conn, addr):
    try:
        with conn:
            print(f"Connected by {addr}")
            while True:
                data = conn.recv(1024)  # Λήψη δεδομένων από τον client
                if not data:
                    break


                # Ανάγνωση των δεδομένων από τον client
                message = json.loads(data.decode())
                userid = message.get('userid')
                nodeid = message.get('nodeid')
                field1 = message.get('field1')
                field2 = message.get('field2')
                field3 = message.get('field3')
                field4 = message.get('field4')


                # Αποθήκευση των πληροφοριών του client στη δομή clients
                clients[(nodeid, userid)] = conn
                print(f"Received from client {addr}: {message}")


                # Κλήση για αποθήκευση των δεδομένων στη βάση δεδομένων
                save_data_to_db(nodeid, userid, field1, field2, field3, field4)


    except (ConnectionResetError, ConnectionAbortedError, OSError) as e:
        print(f"Client {addr} disconnected unexpectedly: {e}")
```

```
    finally:
        # Αφαίρεση του client από τη λίστα όταν αποσυνδέεται
        client_key = (nodeid, userid)
        if client_key in clients:
            del clients[client_key]
            print(f"Removed client {client_key} from active clients.")

# Συνάρτηση για την αποθήκευση δεδομένων στη βάση MySQL
def save_data_to_db(nodeid, userid, field1, field2, field3, field4):
    try:
        # Σύνδεση στη βάση δεδομένων
        connection = pymysql.connect(
            host='localhost',  # Προσαρμόστε το αν ο server MySQL είναι αλλού
            user='root',       # Χρήστης MySQL
            password='',  # Κωδικός πρόσβασης MySQL
            database='wsiot'   # Όνομα βάσης δεδομένων
        )

        with connection.cursor() as cursor:
            # SQL εντολή για εισαγωγή δεδομένων στη βάση
            sql_insert_query = """
            INSERT INTO fieldvalues (nodeid, userid, field1, field2, field3, field4, created_at)
            VALUES (%s, %s, %s, %s, %s, %s, NOW())
            """
            record = (nodeid, userid, field1, field2, field3, field4)
            cursor.execute(sql_insert_query, record)
            connection.commit()  # Αποθήκευση των αλλαγών στη βάση
            print(f"Data saved to database for nodeid={nodeid}, userid={userid}")

    except pymysql.MySQLError as e:
        print(f"Error while connecting to MySQL: {e}")
    finally:
        if connection:
            connection.close()  # Κλείσιμο της σύνδεσης με τη βάση
            print("MySQL connection is closed")

# Λειτουργία που θα λάβει την εντολή από τον Flask Server και θα στείλει τις εντολές act1 και act2
```

```
def handle_command(nodeid, userid, act1, act2):
    try:
        client_key = (nodeid, userid)
        if client_key in clients:
            client_conn = clients[client_key]
            # Στέλνουμε τις εντολές act1 και act2 στον συνδεδεμένο client
            message = json.dumps({
                "act1": act1,
                "act2": act2
            })
            client_conn.sendall(message.encode())  # Αποστολή εντολής στον client
            print(f"Sent to client {client_key}: act1={act1}, act2={act2}")
        else:
            print(f"No client connected with nodeid={nodeid} and userid={userid}")
    except Exception as e:
        print(f"Error handling command: {e}")

# Λειτουργία για την εκκίνηση του TCP Server που ακούει τους clients (κόμβους)
def start_tcp_server():
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind(('localhost', 1821))  # Δέσμευση θύρας 1821
        s.listen()
        print('TCP Server listening on port 1821')

        while True:
            conn, addr = s.accept()  # Αποδοχή σύνδεσης από client
            # Δημιουργία και εκκίνηση thread για τον client
            client_thread = threading.Thread(target=handle_client, args=(conn, addr))
            client_thread.start()

# Ειδικό thread για τη λήψη εντολών από το Flask Server
def start_command_listener():
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind(('localhost', 1822))  # Άλλη θύρα για εντολές από Flask (1822)
        s.listen()
        print('Listening for commands on port 1822')

        while True:
```

```
            conn, addr = s.accept()  # Αποδοχή εντολής
            data = conn.recv(1024)
            if data:
                command = json.loads(data.decode())  # Ανάγνωση εντολής
                nodeid = command.get('nodeid')
                userid = command.get('userid')
                act1 = command.get('act1')
                act2 = command.get('act2')
                # Κλήση της συνάρτησης που στέλνει τις εντολές στον client
                handle_command(nodeid, userid, act1, act2)

if __name__ == "__main__":
    # Εκκίνηση των δύο threads: ενός για τον TCP server και ενός για τη λήψη εντολών
    server_thread = threading.Thread(target=start_tcp_server)
    command_thread = threading.Thread(target=start_command_listener)

    server_thread.start()  # Εκκίνηση TCP server
    command_thread.start()  # Εκκίνηση ακρόασης εντολών από Flask

# Εξήγηση: Αυτός ο κώδικας δημιουργεί έναν TCP WebSocket Server που διαχειρίζεται επικοινωνία με IoT κόμβους
# και παράλληλα λαμβάνει εντολές από έναν Flask HTTP Server, τις οποίες προωθεί στους συνδεδεμένους κόμβους.
# Ο server αποθηκεύει τα δεδομένα από τους κόμβους σε MySQL βάση δεδομένων και επικοινωνεί με τους κόμβους
# σε πραγματικό χρόνο μέσω WebSockets.
```

Ο κώδικας υλοποιεί έναν TCP WebSocket server που επιτρέπει την επικοινωνία σε πραγματικό χρόνο μεταξύ IoT κόμβων (όπως αισθητήρες ή ενεργοποιητές) και ενός κεντρικού διακομιστή, καθώς και την αποθήκευση αυτών των δεδομένων σε μια MySQL βάση δεδομένων. Παράλληλα, ο server λαμβάνει εντολές από έναν Flask HTTP server και τις προωθεί στους αντίστοιχους IoT κόμβους, επιτρέποντας την απομακρυσμένη διαχείριση των συσκευών.

Διαχείριση των Clients (Κόμβων)

Κάθε client (κόμβος) που συνδέεται με τον server εξυπηρετείται από ένα ξεχωριστό thread, το οποίο εκκινεί όταν λαμβάνεται σύνδεση από έναν IoT κόμβο. Η συνάρτηση handle_client() διαχειρίζεται τη σύνδεση κάθε κόμβου, λαμβάνοντας δεδομένα από τον κόμβο μέσω του πρωτοκόλλου TCP. Τα δεδομένα αυτά περιέχουν τιμές από διάφορους αισθητήρες του κόμβου, όπως field1, field2, field3, και field4, καθώς και τα αναγνωριστικά του χρήστη (userid) και του κόμβου (nodeid).

Οι πληροφορίες αυτές αποθηκεύονται σε μια δομή clients, όπου το nodeid και το userid χρησιμοποιούνται ως κλειδιά για να συνδέσουν τον κόμβο με την κατάλληλη σύνδεση TCP. Αυτό επιτρέπει στο server να εντοπίσει άμεσα ποιος client ανήκει σε κάθε κόμβο και να επικοινωνήσει με αυτόν αργότερα, εάν απαιτηθεί.

Αποθήκευση Δεδομένων στη MySQL

Τα δεδομένα που λαμβάνονται από τους IoT κόμβους αποθηκεύονται στη βάση δεδομένων MySQL. Η συνάρτηση save_data_to_db() αναλαμβάνει την εισαγωγή αυτών των δεδομένων στον πίνακα fieldvalues, αποθηκεύοντας τις τιμές από τα πεδία field1, field2, field3, και field4, μαζί με τα nodeid και userid. Επίσης, κάθε εγγραφή στη βάση συνοδεύεται από το created_at πεδίο, το οποίο αποθηκεύει την ημερομηνία και ώρα εισαγωγής.

Η σύνδεση στη MySQL πραγματοποιείται μέσω της βιβλιοθήκης pymysql, η οποία επιτρέπει την εκτέλεση SQL εντολών από την Python. Το σύστημα προσπαθεί να συνδεθεί στη βάση, εκτελεί την εισαγωγή των δεδομένων, και στη συνέχεια κλείνει τη σύνδεση, εξασφαλίζοντας αποδοτική διαχείριση των πόρων.

Εντολές από Flask Server

Ο server όχι μόνο λαμβάνει δεδομένα από τους κόμβους, αλλά και δέχεται εντολές από έναν Flask HTTP server. Η συνάρτηση start_command_listener() είναι υπεύθυνη για την ακρόαση των εντολών από το Flask server μέσω μιας ξεχωριστής θύρας TCP. Οι εντολές αυτές αφορούν την αποστολή ενεργειών, όπως act1 και act2, στους IoT κόμβους.

Όταν ληφθεί μια εντολή από τον Flask server, η συνάρτηση handle_command() αναλαμβάνει να στείλει την εντολή στον αντίστοιχο client (κόμβο), με βάση το nodeid και το userid. Εάν ο κόμβος είναι συνδεδεμένος, η εντολή στέλνεται σε πραγματικό χρόνο μέσω της ήδη ανοικτής σύνδεσης TCP, επιτρέποντας την απομακρυσμένη ενεργοποίηση ή απενεργοποίηση συσκευών, όπως λαμπτήρων ή άλλων μηχανισμών.

Πολυνημάτωση (Threading)

Ο server υποστηρίζει την ταυτόχρονη επικοινωνία με πολλούς κόμβους χρησιμοποιώντας threads. Κάθε σύνδεση client δημιουργεί ένα νέο thread, το οποίο διαχειρίζεται την επικοινωνία με τον συγκεκριμένο κόμβο. Επιπλέον, υπάρχει ξεχωριστό thread για την ακρόαση των εντολών από τον Flask server. Αυτή η προσέγγιση επιτρέπει στον server να εξυπηρετεί πολλούς κόμβους και να δέχεται εντολές από τον Flask server ταυτόχρονα, χωρίς να μπλοκάρει καμία από τις δύο λειτουργίες.

Ασφάλεια και Συντήρηση Συνδέσεων

Όταν ένας client (κόμβος) αποσυνδεθεί ξαφνικά ή τερματίσει τη σύνδεση, ο server ανιχνεύει το συμβάν και αφαιρεί τη σύνδεση από τη δομή clients. Αυτό επιτρέπει τη σωστή διαχείριση των ενεργών

συνδέσεων και την αποτροπή απώλειας μνήμης ή άλλων πόρων. Επιπλέον, όταν ληφθεί μια νέα σύνδεση για έναν ήδη υπάρχοντα κόμβο, η προηγούμενη σύνδεση αντικαθίσταται, εξασφαλίζοντας ότι οι συσκευές είναι πάντα συνδεδεμένες με τον server σε πραγματικό χρόνο.

4.3 Websocket client

Ο κώδικας αυτός υλοποιεί έναν απλό IoT client που συνδέεται σε έναν TCP WebSocket server και ανταλλάσσει δεδομένα σε πραγματικό χρόνο. Ο client αυτός αντιπροσωπεύει έναν IoT κόμβο, όπως έναν αισθητήρα ή έναν ενεργοποιητή, και επικοινωνεί με τον server στέλνοντας μετρήσεις και λαμβάνοντας εντολές για την εκτέλεση ενεργειών. Η επικοινωνία πραγματοποιείται μέσω TCP sockets και τα δεδομένα που ανταλλάσσονται είναι μορφοποιημένα σε JSON.

```
import socket
import json


# Διεύθυνση και θύρα του TCP server
SERVER_ADDRESS = 'localhost'
SERVER_PORT = 1821


# Στοιχεία του client
client_data = {
    "userid": 1,
    "nodeid": 1,
    "field1": 25.5,
    "field2": 30.0,
    "field3": 15.2,
    "field4": 18
}


def connect_to_server():
    try:
        # Δημιουργία του TCP socket
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            # Σύνδεση με τον TCP server
```

```
            s.connect((SERVER_ADDRESS, SERVER_PORT))
            print(f"Connected to server at {SERVER_ADDRESS}:{SERVER_PORT}")

            # Αποστολή δεδομένων πεδίων στον server
            message = json.dumps(client_data)
            s.sendall(message.encode())
            print(f"Sent data to server: {message}")

            # Λήψη μηνυμάτων από τον server
            while True:
                data = s.recv(1024).decode()
                if data:
                    response = json.loads(data)
                    act1 = response.get("act1")
                    act2 = response.get("act2")
                    print(f"Received from server: act1={act1}, act2={act2}")
        except Exception as e:
            print(f"Error: {e}")

if __name__ == "__main__":
    # Συνδέεται στον server και αναμένει εντολές
    connect_to_server()
```

Δομή των δεδομένων του Client

Τα δεδομένα που στέλνει ο client στον server περιλαμβάνουν το userid και το nodeid, τα οποία ταυτοποιούν τον χρήστη και τον κόμβο αντίστοιχα. Επίσης, περιλαμβάνονται τέσσερα πεδία μετρήσεων (field1, field2, field3, field4) που αντιπροσωπεύουν τις τιμές που συλλέγει ο αισθητήρας του κόμβου, όπως θερμοκρασία, υγρασία, πίεση κ.λπ. Τα δεδομένα αυτά αποθηκεύονται σε μία δομή client_data, η οποία χρησιμοποιείται στη συνέχεια για να αποσταλούν στον server.

Σύνδεση με τον TCP Server

Η συνάρτηση connect_to_server() δημιουργεί ένα TCP socket και συνδέεται με τον TCP server μέσω της θύρας 1821. Η διεύθυνση του server και η θύρα καθορίζονται από τις μεταβλητές SERVER_ADDRESS και SERVER_PORT, αντίστοιχα. Μετά τη σύνδεση, ο client στέλνει τα

δεδομένα του στον server σε μορφή JSON, χρησιμοποιώντας την εντολή s.sendall(). Τα δεδομένα μετατρέπονται σε JSON με χρήση της json.dumps() πριν την αποστολή τους.

Λήψη Εντολών από τον Server

Αφού ο client στείλει τα δεδομένα του στον server, αναμένει για πιθανές εντολές από τον server μέσω της συνάρτησης recv(). Ο server μπορεί να στείλει εντολές act1 και act2, τις οποίες ο client λαμβάνει, αποκωδικοποιεί και εκτυπώνει. Οι εντολές αυτές μπορεί να αφορούν τη λειτουργία του IoT κόμβου, όπως ενεργοποίηση ή απενεργοποίηση συσκευών ή άλλες ενέργειες που σχετίζονται με τον κόμβο.

Χειρισμός Σφαλμάτων

Σε περίπτωση που προκύψει σφάλμα κατά τη σύνδεση ή την επικοινωνία με τον server, το σφάλμα καταγράφεται και εμφανίζεται στην οθόνη με χρήση της try-except δομής. Αυτό εξασφαλίζει ότι τυχόν αποτυχίες στη σύνδεση ή την ανταλλαγή δεδομένων δεν θα καταστρέψουν το πρόγραμμα και θα δοθεί αναλυτικό μήνυμα σφάλματος.

Σενάριο Εκτέλεσης

Το πρόγραμμα ξεκινάει με την εκτέλεση της συνάρτησης connect_to_server() μέσω της κύριας δομής if __name__ == "__main__":. Κατά την εκτέλεση, ο client συνδέεται στον server, στέλνει τα δεδομένα του, και παραμένει συνδεδεμένος για να λαμβάνει τυχόν εντολές που θα του στείλει ο server.

4.4 Flask server – Ιστοσελίδα

Στο project που δημιουργήσαμε με το αρχείο app.py και τις σελίδες HTML, υλοποιούμε μια πλήρως λειτουργική πλατφόρμα για τη διαχείριση IoT κόμβων σε πραγματικό χρόνο. Το σύστημα αυτό επιτρέπει στους χρήστες να συνδεθούν, να δημιουργήσουν και να διαχειριστούν τους IoT κόμβους τους μέσω ενός φιλικού περιβάλλοντος web, το οποίο βασίζεται στο Flask, ένα ελαφρύ framework για την ανάπτυξη web εφαρμογών στην Python.

Flask Backend (app.py): Το αρχείο app.py είναι η καρδιά του backend της εφαρμογής. Είναι ένας Flask server που:

Διαχειρίζεται την εγγραφή και σύνδεση των χρηστών: Με λειτουργίες για τη δημιουργία νέων λογαριασμών και τη σύνδεση υπαρχόντων χρηστών.

Διαχειρίζεται κόμβους IoT: Οι εξουσιοδοτημένοι χρήστες μπορούν να δημιουργούν, επεξεργάζονται και διαγράφουν IoT κόμβους. Κάθε κόμβος περιέχει διάφορα πεδία που αντιπροσωπεύουν αισθητήρες ή ενεργοποιητές, όπως π.χ. θερμοκρασία, υγρασία κ.λπ.

Αλληλεπίδραση με WebSocket server: Ο Flask server λαμβάνει εντολές από τους χρήστες, όπως ενεργοποίηση ή απενεργοποίηση συσκευών, και τις προωθεί στον WebSocket server, ο οποίος με τη σειρά του επικοινωνεί με τους IoT κόμβους.

Σελίδες HTML (Frontend): Οι σελίδες HTML που δημιουργήσαμε συνδυάζονται με τον Flask server για να παρέχουν ένα εύχρηστο περιβάλλον στους χρήστες:

Αρχική σελίδα και Σελίδα Σύνδεσης/Εγγραφής: Οι χρήστες μπορούν να συνδεθούν ή να δημιουργήσουν έναν νέο λογαριασμό μέσω αυτών των σελίδων.

Dashboard: Μια κεντρική σελίδα που εμφανίζει όλους τους IoT κόμβους του χρήστη. Ο χρήστης μπορεί να δει μια λίστα με όλους τους κόμβους, να κάνει επεξεργασία των χαρακτηριστικών τους ή να τους διαγράψει.

Σελίδα Επεξεργασίας Κόμβου: Αυτή η σελίδα επιτρέπει στους χρήστες να τροποποιούν τα χαρακτηριστικά ενός κόμβου, όπως το όνομα, το είδος των αισθητήρων, και τις παραμέτρους του.

Σελίδα Προβολής Γραφημάτων: Εδώ οι χρήστες μπορούν να δουν γραφήματα σε πραγματικό χρόνο από τα δεδομένα των αισθητήρων των κόμβων τους. Χρησιμοποιούμε Google Charts για την απεικόνιση των μετρήσεων των αισθητήρων (π.χ. θερμοκρασία, υγρασία) που έχουν αποσταλεί από τους κόμβους και αποθηκευτεί στη βάση δεδομένων.

Προβολή και Αποστολή Εντολών σε IoT Κόμβους: Από το dashboard, οι χρήστες έχουν τη δυνατότητα να στέλνουν εντολές στους IoT κόμβους μέσω WebSockets. Οι εντολές αυτές, όπως ενεργοποίηση ή απενεργοποίηση μιας συσκευής (π.χ. φως, μοτέρ), στέλνονται απευθείας στους συνδεδεμένους IoT κόμβους, οι οποίοι ανταποκρίνονται σε πραγματικό χρόνο.

Βάση Δεδομένων: Η βάση δεδομένων MySQL χρησιμοποιείται για την αποθήκευση των χρηστών, των IoT κόμβων και των δεδομένων των αισθητήρων. Κάθε φορά που ένας κόμβος αποστέλλει δεδομένα, αυτά αποθηκεύονται στη βάση δεδομένων, ενώ κάθε εγγραφή ή αλλαγή που γίνεται από τον χρήστη καταγράφεται επίσης στη βάση.

Ο κώδικας βρίσκεται στο παράρτημα Α.

4.5 Η βάση

Η βάση δεδομένων του συστήματος που υλοποιήθηκε έχει ως στόχο την αποθήκευση των δεδομένων των IoT κόμβων και των χρηστών που διαχειρίζονται αυτούς τους κόμβους. Η βάση δεδομένων έχει δημιουργηθεί σε MySQL και αποτελείται από τρεις βασικούς πίνακες: users, nodes, και fieldvalues. Ο κάθε πίνακας έχει συγκεκριμένο ρόλο στη λειτουργία της πλατφόρμας, όπως περιγράφεται παρακάτω.

Πίνακας users

Ο πίνακας users αποθηκεύει τις πληροφορίες των χρηστών που έχουν πρόσβαση στην πλατφόρμα. Τα πεδία του πίνακα περιλαμβάνουν:

id: Ένας μοναδικός αύξων αριθμός που ταυτοποιεί τον κάθε χρήστη.

enable: Ένα πεδίο που δείχνει αν ο χρήστης είναι ενεργός (1) ή αν έχει απενεργοποιηθεί.

kind: Ένα πεδίο που καθορίζει τον τύπο του χρήστη (π.χ., 0 για κανονικούς χρήστες).

email: Το email του χρήστη, το οποίο χρησιμοποιείται για την είσοδο στην πλατφόρμα.

pass: Ο κωδικός πρόσβασης του χρήστη.

fname και lname: Το όνομα και το επώνυμο του χρήστη αντίστοιχα.

Αυτός ο πίνακας χρησιμεύει στη διαχείριση των χρηστών και των δικαιωμάτων τους στην πλατφόρμα, επιτρέποντας την ταυτοποίηση και αυθεντικοποίηση τους κατά τη σύνδεση.

Πίνακας nodes

Ο πίνακας nodes αποθηκεύει πληροφορίες για κάθε IoT κόμβο που ανήκει σε έναν συγκεκριμένο χρήστη. Κάθε κόμβος αντιστοιχεί σε μια φυσική συσκευή IoT και περιέχει τις εξής πληροφορίες:

id: Ένας μοναδικός αύξων αριθμός που ταυτοποιεί τον κόμβο.

enable: Ένα πεδίο που δείχνει αν ο κόμβος είναι ενεργός (1).

userid: Ο αριθμός ταυτοποίησης του χρήστη στον οποίο ανήκει ο κόμβος.

name: Το όνομα του κόμβου που καθορίζεται από τον χρήστη.

field1name, field2name, field3name, field4name: Τα ονόματα των πεδίων μετρήσεων του κόμβου (π.χ. θερμοκρασία, υγρασία, πίεση), τα οποία προσαρμόζονται ανάλογα με τις ανάγκες του χρήστη και των συσκευών.

Ο πίνακας αυτός επιτρέπει στους χρήστες να διαχειρίζονται τους κόμβους τους και τα δεδομένα που συλλέγουν, παρέχοντας τη δυνατότητα παραμετροποίησης των μετρήσεων για κάθε κόμβο.

Πίνακας fieldvalues

Ο πίνακας fieldvalues αποθηκεύει τις μετρήσεις που λαμβάνονται από τους IoT κόμβους σε πραγματικό χρόνο. Κάθε εγγραφή στον πίνακα αυτό αντιπροσωπεύει ένα σύνολο μετρήσεων που έχει καταγραφεί για έναν συγκεκριμένο κόμβο σε μια συγκεκριμένη χρονική στιγμή. Τα πεδία του πίνακα περιλαμβάνουν:

id: Ένας μοναδικός αύξων αριθμός που ταυτοποιεί κάθε μέτρηση.

enable: Ένα πεδίο που δείχνει αν η εγγραφή είναι ενεργή (1).

nodeid: Ο αριθμός ταυτοποίησης του κόμβου από τον οποίο προέρχονται οι μετρήσεις.

userid: Ο αριθμός ταυτοποίησης του χρήστη στον οποίο ανήκει ο κόμβος.

field1, field2, field3, field4: Οι τιμές των τεσσάρων πεδίων που αντιστοιχούν στις μετρήσεις του κόμβου (π.χ. θερμοκρασία, υγρασία κ.λπ.).

created_at: Η χρονική στιγμή κατά την οποία καταγράφηκε η μέτρηση, σε μορφή datetime.

Αυτός ο πίνακας καταγράφει τις μετρήσεις που λαμβάνονται από τους κόμβους σε πραγματικό χρόνο, επιτρέποντας την παρακολούθηση της λειτουργίας τους και την αποθήκευση ιστορικών δεδομένων.

Ευρετήρια και AUTO_INCREMENT

Κάθε πίνακας περιλαμβάνει ένα PRIMARY KEY ευρετήριο στο πεδίο id για να εξασφαλιστεί η μοναδικότητα των εγγραφών. Επίσης, οι πίνακες fieldvalues, nodes και users έχουν ρυθμιστεί ώστε το πεδίο id να είναι AUTO_INCREMENT, το οποίο σημαίνει ότι η τιμή του αυξάνεται αυτόματα με κάθε νέα εγγραφή. Αυτό διευκολύνει την αυτόματη διαχείριση των πρωτεύοντων κλειδιών.

Συνολική Λειτουργία της Βάσης

Η βάση δεδομένων wsiot είναι πλήρως λειτουργική και εξυπηρετεί τις ανάγκες της πλατφόρμας για τη διαχείριση χρηστών, IoT κόμβων, και δεδομένων από αισθητήρες. Η χρήση των τριών βασικών

πινάκων επιτρέπει τη συλλογή, αποθήκευση, και διαχείριση των μετρήσεων από τους κόμβους, ενώ ταυτόχρονα διασφαλίζεται η σωστή διαχείριση χρηστών και κόμβων. Η βάση δεδομένων υποστηρίζει την επεκτασιμότητα της πλατφόρμας, επιτρέποντας την προσθήκη νέων χρηστών και κόμβων, καθώς και την αποθήκευση μεγάλου όγκου δεδομένων από τους IoT κόμβους σε πραγματικό χρόνο.

4.6 Ασφάλεια στο σύστημα και στα δεδομένα

Η ασφάλεια αποτελεί έναν από τους πιο κρίσιμους παράγοντες στη διαχείριση των συστημάτων Internet of Things (IoT), δεδομένου ότι οι συσκευές αυτές συχνά συλλέγουν και διαβιβάζουν ευαίσθητα δεδομένα. Στην πλατφόρμα που αναπτύχθηκε, η ασφάλεια των δεδομένων και των επικοινωνιών διασφαλίζεται μέσω της χρήσης κρυπτογραφημένων WebSockets, τα οποία παρέχουν προστασία έναντι επιθέσεων man-in-the-middle και άλλων κακόβουλων ενεργειών. Η κρυπτογράφηση των δεδομένων τόσο κατά την αποστολή όσο και κατά την αποθήκευση είναι απαραίτητη για την προστασία της ακεραιότητας και της εμπιστευτικότητας των πληροφοριών που διαχειρίζεται η πλατφόρμα.

Επιπλέον, η αυθεντικοποίηση των χρηστών είναι ένας βασικός μηχανισμός για την πρόληψη μη εξουσιοδοτημένης πρόσβασης στο σύστημα. Η πλατφόρμα ενσωματώνει διαδικασίες ελέγχου ταυτότητας και διαχείρισης χρηστών, επιτρέποντας μόνο σε εξουσιοδοτημένα άτομα να έχουν πρόσβαση στις λειτουργίες και τα δεδομένα του συστήματος. Αυτό συμβάλλει στην προστασία των δεδομένων από μη εξουσιοδοτημένες τροποποιήσεις ή διαρροές, εξασφαλίζοντας ότι μόνο οι αρμόδιοι χρήστες μπορούν να αλληλεπιδρούν με την πλατφόρμα.

Τέλος, ένα επιπλέον επίπεδο ασφάλειας μπορεί να προστεθεί με τη χρήση τεχνολογιών όπως η ανίχνευση ανωμαλιών και η παρακολούθηση δικτύου σε πραγματικό χρόνο. Αυτές οι τεχνικές επιτρέπουν την ανίχνευση ύποπτων δραστηριοτήτων ή προσπαθειών παραβίασης της ασφάλειας, προσφέροντας τη δυνατότητα έγκαιρης αντίδρασης και αντιμετώπισης. Με τη συνεχή παρακολούθηση της πλατφόρμας και την εφαρμογή πρωτοκόλλων ασφαλείας, είναι δυνατόν να μειωθεί ο κίνδυνος επιθέσεων και να διασφαλιστεί η σταθερότητα και η αξιοπιστία του συστήματος σε μακροπρόθεσμη βάση.

Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης

Η παρούσα εργασία έδειξε τη σημασία της χρήσης της τεχνολογίας WebSocket για την υποστήριξη αμφίδρομης επικοινωνίας σε πραγματικό χρόνο σε πλατφόρμες Internet of Things (IoT). Η ανάπτυξη της πλατφόρμας επέτρεψε την άμεση και αποδοτική ανταλλαγή δεδομένων μεταξύ των συσκευών και του κεντρικού διακομιστή, γεγονός που βελτίωσε τη συνολική απόδοση και ανταπόκριση των IoT συστημάτων. Τα αποτελέσματα έδειξαν ότι η χρήση των WebSockets εξασφάλισε χαμηλότερους χρόνους καθυστέρησης και καλή αξιοπιστία στη μεταφορά δεδομένων σε σύγκριση με παραδοσιακές μεθόδους επικοινωνίας.

Η πλατφόρμα που υλοποιήθηκε προσέφερε στους χρήστες ένα ολοκληρωμένο και εύχρηστο περιβάλλον διαχείρισης των συσκευών IoT, με δυνατότητες δημιουργίας, τροποποίησης και διαγραφής κόμβων, καθώς και διαχείρισης αισθητήρων και ενεργοποιητών. Η διεπαφή χρήστη, που αναπτύχθηκε με το Flask framework, αποδείχθηκε λειτουργική και ευέλικτη, διευκολύνοντας τους εξουσιοδοτημένους χρήστες να παρακολουθούν τα δεδομένα και να λαμβάνουν άμεσα αποφάσεις για τη λειτουργία των συσκευών τους.

Ακόμη, η αποθήκευση και η επεξεργασία των δεδομένων σε βάση δεδομένων προσέφερε σημαντικά πλεονεκτήματα στην ανάλυση των πληροφοριών που συλλέχθηκαν, επιτρέποντας τη δημιουργία λεπτομερών αναφορών και την εξαγωγή χρήσιμων συμπερασμάτων για τη βελτιστοποίηση των λειτουργιών των IoT συστημάτων. Η συνολική αρχιτεκτονική της πλατφόρμας έδειξε πως η ενσωμάτωση σύγχρονων τεχνολογιών μπορεί να προσφέρει ουσιαστική αξία στην ανάπτυξη και διαχείριση IoT εφαρμογών.

Η εργασία κατέδειξε επίσης τη σημαντικότητα της ασφάλειας και της διασφάλισης της ιδιωτικότητας στη διαχείριση των IoT συστημάτων. Η πλατφόρμα παρέχει τις απαραίτητες δικλείδες ασφαλείας για την προστασία των δεδομένων και την εξασφάλιση της ασφαλούς επικοινωνίας μεταξύ των συσκευών και των διακομιστών.

Παρά τα επιτυχημένα αποτελέσματα, υπάρχουν ορισμένα πεδία όπου η πλατφόρμα μπορεί να βελτιωθεί. Πρώτον, η βελτιστοποίηση της πλατφόρμας για μεγαλύτερη κλίμακα συσκευών είναι απαραίτητη. Καθώς το IoT συνεχίζει να αναπτύσσεται, η δυνατότητα της πλατφόρμας να διαχειρίζεται εκατοντάδες ή και χιλιάδες συνδεδεμένες συσκευές θα αποτελέσει κρίσιμο παράγοντα. Για να επιτευχθεί αυτό, θα μπορούσαν να εξεταστούν τεχνικές όπως η κατανομή φόρτου και η χρήση πιο αποδοτικών αλγορίθμων για την επεξεργασία των δεδομένων.

Δεύτερον, η βελτίωση της ασφάλειας είναι ένα ακόμα σημαντικό ζήτημα. Παρόλο που η πλατφόρμα διαθέτει βασικές δικλείδες ασφαλείας, η ενσωμάτωση προηγμένων μεθόδων κρυπτογράφησης και ο έλεγχος ταυτότητας πολλαπλών παραγόντων θα μπορούσαν να ενισχύσουν περαιτέρω την προστασία των δεδομένων. Η προσθήκη επιπλέον επιπέδων ασφαλείας θα διασφαλίσει την ιδιωτικότητα και την ακεραιότητα των δεδομένων, ιδιαίτερα σε περιβάλλοντα με αυξημένες απαιτήσεις ασφάλειας.

Τέλος, η διεπαφή χρήστη μπορεί να επεκταθεί και να βελτιωθεί περαιτέρω με την ενσωμάτωση εργαλείων ανάλυσης δεδομένων και δυναμικών γραφικών απεικονίσεων. Αυτές οι προσθήκες θα επιτρέψουν στους χρήστες να αποκτούν βαθύτερη κατανόηση των δεδομένων που συλλέγονται και να λαμβάνουν πιο ενημερωμένες αποφάσεις. Η ανάπτυξη λειτουργιών που επιτρέπουν την προσαρμογή των αναφορών και την ανάλυση σε πραγματικό χρόνο θα αυξήσει σημαντικά την αξία της πλατφόρμας για τους τελικούς χρήστες.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Park, J. T., Hwang, H. S., Yun, J. S., & Moon, I. Y. (2014). Study of HTML5 WebSocket for a Multimedia Communication. *International Journal of Multimedia & Ubiquitous Engineering*, *9*(7), 61-72.

[2] Ma, K., & Sun, R. (2013). Introducing websocket-based real-time monitoring system for remote intelligent buildings. *International Journal of Distributed Sensor Networks*, *9*(12), 867693

[3] MathWorks. (n.d.). ThingSpeak™ IoT. Retrieved from ThingSpeak Documentation

[4] Martin, C. (2018). An Introduction to ThingSpeak. Retrieved from IoT

[5] ThingsBoard Documentation. (n.d.). ThingsBoard – Open-source IoT Platform. Retrieved from ThingsBoard Documentation

[6] Thinger.io Documentation. (n.d.). Thinger.io – The Open Source IoT Platform. Retrieved from Thinger.io Documentation

[7] IoT For All. (2019). A Beginner's Guide to Thinger.io. Retrieved from IoT For All

[8] Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.

[9] Lutz, M. (2013). Learning Python, 5th Edition. Sebastopol, CA: O'Reilly Media.

[10] Real Python. (n.d.). The Ultimate Guide to Web Development with Python. Retrieved from Real Python

[11] W3Schools. (n.d.). Python Advantages. Retrieved from W3Schools

[12] GeeksforGeeks. (2020). Top 10 Reasons Why You Should Learn Python. Retrieved from GeeksforGeeks

[13] Python.org. (n.d.). http.server — HTTP servers. Retrieved from Python.org

[14] Real Python. (n.d.). Python Web Development Tutorials. Retrieved from Real Python TCP Server

[15] MDN Web Docs. (n.d.). Introduction to WebSockets. Retrieved from MDN Web Docs

[16] Python Websockets. (n.d.). websockets: A library for building WebSocket servers and clients in Python. Retrieved from websockets.readthedocs.io

[17] Real Python. (n.d.). Flask by Example: A Complete Guide. Retrieved from Real Python

[18] MySQL Documentation. (n.d.). MySQL :: MySQL 8.0 Reference Manual. Retrieved from MySQL Docs

ΠΑΡΑΡΤΗΜΑ Α

Στο παράρτημα αυτό αναφέρονται τα βασικά κομμάτια του κώδικα που χρησιμοποιήθηκε.

```
from flask import Flask, render_template, redirect, url_for, flash, request, jsonify
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager, UserMixin, login_user, logout_user, login_required, current_user
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField, BooleanField
from wtforms.validators import DataRequired, Length, EqualTo
from flask_bootstrap import Bootstrap

import socket
import json

app = Flask(__name__)
app.config['SECRET_KEY'] = 'your_secret_key'
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql+pymysql://root:@localhost/wsiot'

db = SQLAlchemy(app)
bootstrap = Bootstrap(app)
login_manager = LoginManager(app)
login_manager.login_view = 'login'

class FieldValue(db.Model):
    __tablename__ = 'fieldvalues'
    id = db.Column(db.Integer, primary_key=True)
    enable = db.Column(db.Boolean, default=True)
    nodeid = db.Column(db.Integer, nullable=False)
    userid = db.Column(db.Integer, nullable=False)
    field1 = db.Column(db.Float, nullable=False)
    field2 = db.Column(db.Float, nullable=False)
    field3 = db.Column(db.Float, nullable=False)
    field4 = db.Column(db.Float, nullable=False)
    created_at = db.Column(db.DateTime, default=db.func.current_timestamp())



# Model για τον User
class User(db.Model, UserMixin):
    __tablename__ = 'users'
```

```
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(100), unique=True)
    passw = db.Column('pass', db.String(100))  # Χρησιμοποιούμε 'pass' όπως είναι στη βάση
    fname = db.Column(db.String(100))
    lname = db.Column(db.String(100))


# Model για τον Node
class Node(db.Model):
    __tablename__ = 'nodes'
    id = db.Column(db.Integer, primary_key=True)
    userid = db.Column(db.Integer, db.ForeignKey('users.id'))
    name = db.Column(db.String(100))
    field1name = db.Column(db.String(100), default='field1')
    field2name = db.Column(db.String(100), default='field2')
    field3name = db.Column(db.String(100), default='field3')
    field4name = db.Column(db.String(100), default='field4')

@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))

# Φόρμα Εγγραφής
class RegistrationForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Length(min=6, max=100)])
    password = PasswordField('Password', validators=[DataRequired(), Length(min=6)])
    confirm_password = PasswordField('Confirm Password', validators=[DataRequired(),
EqualTo('password')])
    fname = StringField('First Name', validators=[DataRequired(), Length(min=2, max=100)])
    lname = StringField('Last Name', validators=[DataRequired(), Length(min=2, max=100)])
    submit = SubmitField('Register')

# Φόρμα Login
class LoginForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Length(min=6, max=100)])
    password = PasswordField('Password', validators=[DataRequired()])
    remember = BooleanField('Remember Me')
    submit = SubmitField('Login')

# Διαχείριση κόμβων
class NodeForm(FlaskForm):
    name = StringField('Node Name', validators=[DataRequired()])
    field1name = StringField('Field 1 Name', validators=[DataRequired()])
    field2name = StringField('Field 2 Name', validators=[DataRequired()])
    field3name = StringField('Field 3 Name', validators=[DataRequired()])
    field4name = StringField('Field 4 Name', validators=[DataRequired()])
    submit = SubmitField('Save')
```

```
# Αρχική Σελίδα
@app.route('/')
def index():
    return render_template('index.html')

# Διαγραφή Κόμβου
@app.route('/delete_node/<int:node_id>', methods=['POST'])
@login_required
def delete_node(node_id):
    node = Node.query.get_or_404(node_id)
    if node.userid != current_user.id:
        flash('Δεν έχετε δικαίωμα να διαγράψετε αυτόν τον κόμβο.', 'danger')
        return redirect(url_for('dashboard'))

    db.session.delete(node)
    db.session.commit()
    flash('Ο κόμβος διαγράφηκε με επιτυχία.', 'success')
    return redirect(url_for('dashboard'))

# Καθαρισμός Δεδομένων Κόμβου
@app.route('/clear_node_data/<int:node_id>', methods=['POST'])
@login_required
def clear_node_data(node_id):
    # Εντοπίζουμε τον κόμβο
    node = Node.query.get_or_404(node_id)
    if node.userid != current_user.id:
        flash('Δεν έχετε δικαίωμα να διαγράψετε τα δεδομένα αυτού του κόμβου.', 'danger')
        return redirect(url_for('dashboard'))

    # Διαγραφή δεδομένων από τον πίνακα fieldvalues που σχετίζονται με αυτόν τον κόμβο
    db.session.execute('DELETE FROM fieldvalues WHERE nodeid = :nodeid', {'nodeid': node.id})
    db.session.commit()

    flash('Τα δεδομένα του κόμβου καθαρίστηκαν με επιτυχία.', 'success')
    return redirect(url_for('dashboard'))

# Σελίδα Εγγραφής
@app.route('/register', methods=['GET', 'POST'])
def register():
    form = RegistrationForm()
    if form.validate_on_submit():
        user = User(email=form.email.data, passw=form.password.data, fname=form.fname.data,
lname=form.lname.data)
        db.session.add(user)
        db.session.commit()
        flash('Registration successful!', 'success')
        return redirect(url_for('login'))
    return render_template('register.html', form=form)
```

```
# Σελίδα Login
@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        if user and user.passw == form.password.data:
            login_user(user, remember=form.remember.data)
            return redirect(url_for('dashboard'))
        else:
            flash('Login Unsuccessful. Please check email and password', 'danger')
    return render_template('login.html', form=form)

# Σελίδα Διαχείρισης Κόμβων
@app.route('/dashboard', methods=['GET', 'POST'])
@login_required
def dashboard():
    nodes = Node.query.filter_by(userid=current_user.id).all()
    return render_template('dashboard.html', nodes=nodes)

# Προσθήκη / Τροποποίηση Κόμβων
@app.route('/node/<int:node_id>', methods=['GET', 'POST'])
@login_required
def edit_node(node_id):
    node = Node.query.get_or_404(node_id)  # Βρίσκουμε τον κόμβο με βάση το node_id
    if node.userid != current_user.id:
        flash('Δεν έχετε δικαίωμα να επεξεργαστείτε αυτόν τον κόμβο.', 'danger')
        return redirect(url_for('dashboard'))

    form = NodeForm()
    if form.validate_on_submit():
        node.name = form.name.data
        node.field1name = form.field1name.data
        node.field2name = form.field2name.data
        node.field3name = form.field3name.data
        node.field4name = form.field4name.data
        db.session.commit()
        flash('Ο κόμβος ενημερώθηκε με επιτυχία!', 'success')
        return redirect(url_for('dashboard'))

    # Γέμισμα της φόρμας με τα υπάρχοντα δεδομένα όταν φορτώνει η σελίδα
    elif request.method == 'GET':
        form.name.data = node.name
        form.field1name.data = node.field1name
        form.field2name.data = node.field2name
        form.field3name.data = node.field3name
        form.field4name.data = node.field4name
```

```
    # Περάστε τον κόμβο στο template
    return render_template('edit_node.html', form=form, node=node)

# Στέλνει HTTP GET request στον TCP server με act1 και act2
def send_act_to_tcp_server(nodeid, userid, act1, act2):
    try:
        # Σύνδεση με τον TCP server
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.connect(('localhost', 1822))
            # Δημιουργία μηνύματος JSON με τις πράξεις act1 και act2
            message = json.dumps({
                "userid": userid,
                "nodeid": nodeid,
                "act1": act1,
                "act2": act2
            })
            # Αποστολή μηνύματος στον TCP server
            s.sendall(message.encode())
            print(f"Sent to TCP server: {message}")
    except Exception as e:
        print(f"Error: {e}")

# Σελίδα Google Charts για τον συγκεκριμένο κόμβο
# @app.route('/node/<int:node_id>/charts')
# @login_required
# def node_charts(node_id):
    # # Παράδειγμα node
    # node = {
        # 'id': node_id,
        # 'name': 'Example Node',
        # 'field1name': 'Temperature',
        # 'field2name': 'Humidity',
        # 'field3name': 'Pressure',
        # 'field4name': 'Airflow'
    # }

    # # Παράδειγμα δεδομένων (αντί για χρήση της βάσης δεδομένων)
    # field_values = [
        # {
            # 'created_at': '2024-09-07T12:34:56',
            # 'field1': 25.5,
            # 'field2': 60.2,
            # 'field3': 1013.2,
            # 'field4': 2.3
        # },
        # {
            # 'created_at': '2024-09-07T12:36:56',
```

```
            # 'field1': 26.0,
            # 'field2': 61.0,
            # 'field3': 1012.8,
            # 'field4': 2.5
        # },
        # {
            # 'created_at': '2024-09-07T12:38:56',
            # 'field1': 24.8,
            # 'field2': 59.9,
            # 'field3': 1013.0,
            # 'field4': 2.4
        # }
    # ]

    # # Εξασφαλίζουμε ότι όλες οι τιμές είναι σωστές πριν τις μετατρέψουμε σε JSON
    # chart_data_json = json.dumps(field_values)

    # # Αποστέλλουμε τα δεδομένα στο template
    # return render_template('node_charts.html', node=node, chart_data=chart_data_json)

@app.route('/node/<int:node_id>/charts')
@login_required
def node_charts(node_id):
    # Ανακτούμε τον κόμβο από τη βάση δεδομένων
    node = Node.query.get_or_404(node_id)

    # Ανακτούμε τις τιμές από τον πίνακα `FieldValue` για τον συγκεκριμένο κόμβο
    field_values = FieldValue.query.filter_by(nodeid=node.id).all()

    # Δημιουργούμε τα δεδομένα για τα γραφήματα
    chart_data = []
    for value in field_values:
        chart_data.append({
            'created_at': value.created_at.isoformat(),  # Μετατρέπουμε την ημερομηνία σε ISO format
            'field1': value.field1 if value.field1 is not None else None,
            'field2': value.field2 if value.field2 is not None else None,
            'field3': value.field3 if value.field3 is not None else None,
            'field4': value.field4 if value.field4 is not None else None
        })

    # Μετατρέπουμε τα δεδομένα σε JSON
    chart_data_json = json.dumps(chart_data)

    # Αποστέλλουμε τα δεδομένα στο template
    return render_template('node_charts.html', node=node, chart_data=chart_data_json)


# Αποστολή εντολών από το Dashboard για τον συγκεκριμένο κόμβο
```

```
@app.route('/send-command/<int:node_id>/<int:act1>/<int:act2>', methods=['POST'])
@login_required
def send_command(node_id, act1, act2):
    node = Node.query.get_or_404(node_id)
    send_act_to_tcp_server(nodeid=node.id, userid=current_user.id, act1=act1, act2=act2)
    flash(f'Command sent to node {node.name} with act1={act1} and act2={act2}.', 'success')
    return redirect(url_for('dashboard'))

# Αποσύνδεση
@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('login'))

if __name__ == '__main__':
    app.run(debug=True)
```