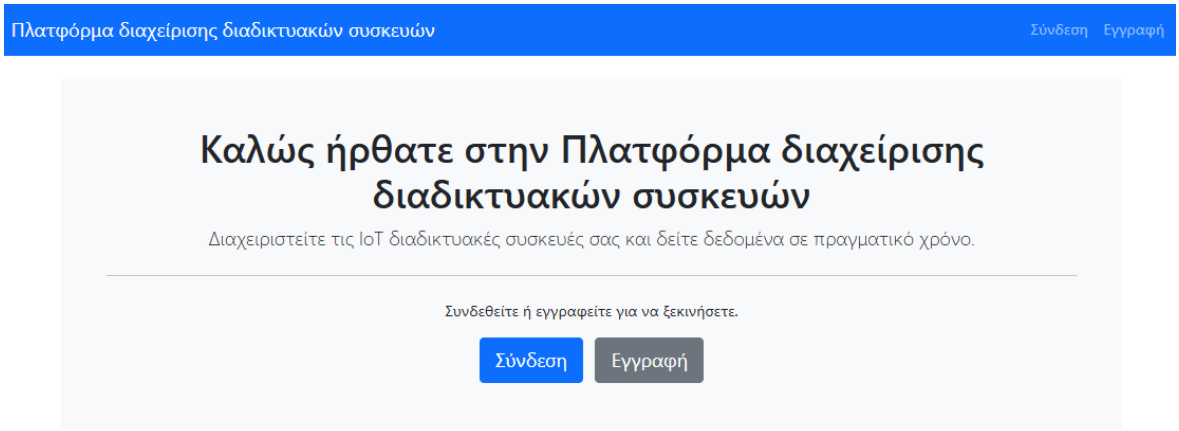
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Πλατφόρμα διαχείρισης διαδικτυακών συσκευών»



Φοιτητής

ΙΩΑΝΝΗΣ ΨΩΜΑΘΙΑΝΟΣ

515164

Επιβλέπων

Δρ. Κυριάκος Τσιακμάκης

ΦΕΒΡΟΥΑΡΙΟΣ 2025

Πλατφόρμα διαχείρισης διαδικτυακών συσκευών

Κωδικός: 24301

Φοιτητής: Ψωμαθιανός Ιωάννης

Εισηγητής: Δρ Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 31-10-2024

Ημερομηνία περάτωσης Π.Ε. 23-01-2025

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 πλατφόρμα διαχείρισης διαδικτυακών συσκευών σχεδιάστηκε για την παρακολούθηση, έλεγχο και συντονισμό IoT συσκευών. Παρέχει εργαλεία για απομακρυσμένη διαχείριση, ανάλυση δεδομένων σε πραγματικό χρόνο και έλεγχο λειτουργικότητας μέσω ασφαλών API keys. Οι χρήστες μπορούν να προσθέτουν συσκευές, να ενημερώνουν τιμές και να παρακολουθούν ιστορικά δεδομένα μέσω γραφημάτων.

Η υλοποίηση βασίζεται σε Python για το backend, MySQL για αποθήκευση δεδομένων, Bootstrap για responsive διεπαφές και ZingChart για γραφήματα. Η πλατφόρμα βελτιστοποιεί την αποδοτικότητα και προσφέρει ασφάλεια, εξασφαλίζοντας πρόσβαση μόνο σε εξουσιοδοτημένους χρήστες. Στόχος είναι η δημιουργία ενός φιλικού και λειτουργικού εργαλείου διαχείρισης συσκευών IoT, κατάλληλου για πολλές εφαρμογές.

« Online Device Management Platform »

Abstract

The web-based device management platform is designed to monitor, control and coordinate IoT devices. It provides tools for remote management, real-time data analysis and functionality control via secure API keys. Users can add devices, update values and monitor historical data via graphs.

The implementation is based on Python for the backend, MySQL for data storage, Bootstrap for responsive interfaces and ZingChart for graphs. The platform optimizes efficiency and offers security, ensuring access only to authorized users. The goal is to create a friendly and functional IoT device management tool, suitable for many applications.

Ευχαριστίες

Τις ευχαριστίες μου στον επιβλέποντα μου και στους γονείς μου.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Το Internet of Things (IoT), ή αλλιώς Διαδίκτυο των Πραγμάτων, αποτελεί μία από τις σημαντικότερες τεχνολογικές καινοτομίες του 21ου αιώνα. Ο όρος IoT αναφέρεται στη διασύνδεση φυσικών συσκευών, όπως αισθητήρες, κάμερες, μηχανές και οικιακές συσκευές, με το διαδίκτυο. Αυτή η διασύνδεση επιτρέπει την ανταλλαγή δεδομένων και την αλληλεπίδραση μεταξύ των συσκευών, ανοίγοντας νέους ορίζοντες στη διαχείριση δεδομένων, την αυτοματοποίηση και την αποδοτικότητα.

Οι εφαρμογές του IoT καλύπτουν ευρύ φάσμα τομέων, όπως η βιομηχανία (Industry 4.0), η υγειονομική περίθαλψη, η γεωργία, οι έξυπνες πόλεις, η ενέργεια και τα έξυπνα σπίτια. Η τεχνολογία IoT προσφέρει

- **Αυτοματοποιημένη Συλλογή Δεδομένων**: Η συνεχής ροή πληροφοριών από συσκευές σε πραγματικό χρόνο μειώνει την ανάγκη για ανθρώπινη παρέμβαση.
- **Βελτιστοποίηση Πόρων**: Η διαχείριση δεδομένων και η πρόβλεψη μελλοντικών αναγκών επιτρέπουν τη μείωση κόστους και τη βελτίωση της αποδοτικότητας.
- **Υπηρεσίες**: Η ανάλυση δεδομένων παρέχει τη δυνατότητα προσαρμογής υπηρεσιών στις ατομικές ανάγκες του χρήστη.

Μία από τις μεγαλύτερες προκλήσεις του IoT είναι η διαχείριση του τεράστιου όγκου δεδομένων που παράγουν οι συσκευές. Εδώ εισέρχονται οι IoT πλατφόρμες, οι οποίες αποτελούν το ενδιάμεσο επίπεδο μεταξύ των συσκευών και των εφαρμογών. Οι IoT πλατφόρμες λειτουργούν ως «γέφυρα» που επιτρέπει την επικοινωνία, την αποθήκευση, την επεξεργασία και την οπτικοποίηση των δεδομένων.

Μία τυπική IoT πλατφόρμα παρέχει:

1. **Διαχείριση Συσκευών**: Περιλαμβάνει την εγγραφή, την παρακολούθηση και τη συντήρηση των συσκευών IoT. Κάθε συσκευή αποκτά έναν μοναδικό αναγνωριστικό αριθμό και παρακολουθείται συνεχώς.
2. **Συλλογή Δεδομένων**: Οι IoT πλατφόρμες συλλέγουν δεδομένα από διάφορες συσκευές και τα αποθηκεύουν σε βάσεις δεδομένων για μελλοντική χρήση.
3. **Ανάλυση Δεδομένων**: Προσφέρουν εργαλεία επεξεργασίας και ανάλυσης δεδομένων, επιτρέποντας στους χρήστες να εξάγουν χρήσιμα συμπεράσματα.
4. **Αυτοματοποίηση**: Οι πλατφόρμες IoT υποστηρίζουν τη δημιουργία αυτοματοποιημένων διαδικασιών που ενεργοποιούνται βάσει δεδομένων και κανόνων.
5. **Ασφάλεια**: Παρέχουν πρωτόκολλα ασφαλείας για την προστασία των δεδομένων από μη εξουσιοδοτημένη πρόσβαση.

Οι IoT πλατφόρμες διατίθενται σε διάφορες μορφές, από εμπορικές λύσεις όπως οι AWS IoT, Google Cloud IoT και Microsoft Azure IoT, μέχρι ανοιχτού κώδικα λύσεις όπως το ThingsBoard. Ενσωματώνουν τεχνολογίες αιχμής όπως το machine learning, τα μεγάλα δεδομένα και την ανάλυση πραγματικού χρόνου, επιτρέποντας στους χρήστες να κατανοούν καλύτερα τα δεδομένα και να βελτιώνουν τη λήψη αποφάσεων.

Παρά τα πλεονεκτήματά τους, οι IoT πλατφόρμες αντιμετωπίζουν και προκλήσεις. Αυτές περιλαμβάνουν την πολυπλοκότητα των δικτύων, την ανάγκη για διαλειτουργικότητα μεταξύ διαφορετικών συσκευών και πρωτοκόλλων, καθώς και την προστασία των δεδομένων από κυβερνοεπιθέσεις. Για την αντιμετώπιση αυτών των προκλήσεων, οι IoT πλατφόρμες εξελίσσονται συνεχώς, ενσωματώνοντας νέες τεχνολογίες και πρακτικές ασφάλειας.

Η ανάπτυξη μίας IoT πλατφόρμας είναι ένα σύνθετο εγχείρημα που απαιτεί τη συνδυασμένη χρήση πολλών τεχνολογιών, από συστήματα cloud και βάσεις δεδομένων μέχρι εφαρμογές ανάλυσης και διεπαφές χρήστη. Το έργο αυτό παρέχει μια πρακτική εφαρμογή των παραπάνω αρχών, εστιάζοντας στη δημιουργία μιας εύχρηστης και ασφαλούς λύσης για τη διαχείριση συσκευών IoT.

Με την επέκταση των IoT τεχνολογιών, οι πλατφόρμες αυτές αποτελούν το θεμέλιο για τη μελλοντική ανάπτυξη του οικοσυστήματος IoT, προσφέροντας λύσεις που προσαρμόζονται στις ανάγκες του κάθε χρήστη και περιβάλλοντος.

Οι στόχοι της εργασίας περιλαμβάνουν:

- **Ανάπτυξη μιας Εύχρηστης Πλατφόρμας**: Δημιουργία ενός φιλικού περιβάλλοντος διαχείρισης συσκευών IoT για τελικούς χρήστες.
- **Ασφάλεια Δεδομένων**: Διασφάλιση ότι τα δεδομένα των συσκευών παραμένουν ασφαλή μέσω της χρήσης κρυπτογραφημένων επικοινωνιών και μοναδικών κλειδιών API.
- **Διαδραστική Ανάλυση Δεδομένων**: Ενσωμάτωση εργαλείων οπτικοποίησης δεδομένων για την παρακολούθηση και ανάλυση πληροφοριών σε πραγματικό χρόνο.
- **Επεκτασιμότητα**: Δημιουργία μιας πλατφόρμας που μπορεί να προσαρμοστεί σε διαφορετικές εφαρμογές και περιβάλλοντα.

1.2 Δομή της εργασίας

Στο πρώτο κεφάλαιο παρουσιάζεται μια εισαγωγή στην εργασία, όπου περιγράφονται η σημασία των πλατφορμών IoT και οι στόχοι της ανάπτυξης του συστήματος. Επίσης, εξηγείται η συμβολή της εργασίας στην κατανόηση και αξιοποίηση των συστημάτων IoT.

Στο δεύτερο κεφάλαιο πραγματοποιείται μια βιβλιογραφική ανασκόπηση, όπου αναλύονται παρόμοιες πλατφόρμες IoT, όπως το AWS IoT Core, το Google Cloud IoT, το Microsoft Azure IoT Hub, και το ThingsBoard. Δίνεται έμφαση στα χαρακτηριστικά και τις λειτουργίες που παρέχουν αυτές οι πλατφόρμες, καθώς και στις διαφορές τους με την παρούσα εργασία.

Στο τρίτο κεφάλαιο περιγράφονται οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του έργου. Αναλύεται η χρήση της γλώσσας Python, του Flask ως web framework, της MySQL για τη διαχείριση δεδομένων, καθώς και των εργαλείων Bootstrap και ZingChart για τη δημιουργία δυναμικών διεπαφών και την οπτικοποίηση δεδομένων.

Στο τέταρτο κεφάλαιο παρουσιάζεται η υλοποίηση της πλατφόρμας. Εξηγείται η δομή του κώδικα, περιγράφονται τα API endpoints και οι λειτουργίες τους, και αναλύεται η βάση δεδομένων. Επίσης, περιλαμβάνονται παραδείγματα χρήσης, διαγράμματα ροής δεδομένων, και στιγμιότυπα οθόνης από την πλατφόρμα. Επιπλέον, εξετάζεται η ασφάλεια του συστήματος, με έμφαση στην προστασία χρηστών, δεδομένων και συσκευών.

Στο πέμπτο κεφάλαιο συνοψίζονται τα αποτελέσματα της εργασίας, επισημαίνοντας την αποτελεσματικότητα της πλατφόρμας. Παρουσιάζονται, επίσης, προτάσεις για μελλοντικές βελτιώσεις και επεκτάσεις, όπως η ενσωμάτωση πιο προηγμένων χαρακτηριστικών και η βελτίωση της απόδοσης σε περιβάλλοντα μεγάλης κλίμακας.

Κεφάλαιο 2ο: Βιβλιογραφική Ανασκόπηση

2.1 Εισαγωγή

Το Internet of Things (IoT) έχει αναδειχθεί ως ένας από τους βασικούς πυλώνες της σύγχρονης τεχνολογίας, επηρεάζοντας διάφορους τομείς της καθημερινής ζωής και της βιομηχανίας. Η βασική ιδέα πίσω από το IoT είναι η διασύνδεση φυσικών συσκευών μέσω του διαδικτύου, ώστε να ανταλλάσσουν δεδομένα και να παρέχουν πληροφορίες σε πραγματικό χρόνο. Η τεχνολογία αυτή επιτρέπει την αυτοματοποίηση διεργασιών, τη βελτιστοποίηση της χρήσης πόρων και την παροχή εξατομικευμένων υπηρεσιών σε χρήστες. Η εξέλιξη του IoT βασίζεται στην πρόοδο σε τομείς όπως η ασύρματη επικοινωνία, η επεξεργασία δεδομένων και η τεχνητή νοημοσύνη.

Οι πλατφόρμες IoT είναι το κεντρικό σημείο συνάντησης αυτής της τεχνολογίας, παρέχοντας τις απαραίτητες υποδομές για τη συλλογή, αποθήκευση, ανάλυση και οπτικοποίηση δεδομένων που προέρχονται από συσκευές IoT. Μια τυπική πλατφόρμα IoT συνδέει τις συσκευές με εφαρμογές και χρήστες, επιτρέποντας την ασφαλή επικοινωνία και τη λήψη αποφάσεων βάσει δεδομένων. Οι πλατφόρμες αυτές ενσωματώνουν λειτουργίες όπως η παρακολούθηση της κατάστασης των συσκευών, η διαχείριση ενημερώσεων λογισμικού, η ανάλυση δεδομένων και η αυτοματοποίηση ενεργειών. Επιπλέον, οι πλατφόρμες IoT παρέχουν μηχανισμούς ασφάλειας για την προστασία των δεδομένων από μη εξουσιοδοτημένη πρόσβαση και κυβερνοεπιθέσεις.

Η χρησιμότητα των πλατφορμών IoT είναι προφανής σε πολλούς τομείς εφαρμογής. Στη βιομηχανία, το IoT επιτρέπει τη δημιουργία έξυπνων εργοστασίων, όπου οι μηχανές μπορούν να επικοινωνούν μεταξύ τους, να παρακολουθούνται σε πραγματικό χρόνο και να συντηρούνται προληπτικά. Στη γεωργία, οι πλατφόρμες IoT επιτρέπουν την παρακολούθηση συνθηκών όπως η υγρασία του εδάφους και οι καιρικές συνθήκες, βελτιώνοντας την αποδοτικότητα της καλλιέργειας. Στον τομέα της υγείας, το IoT υποστηρίζει την απομακρυσμένη παρακολούθηση ασθενών μέσω έξυπνων συσκευών, ενώ στις έξυπνες πόλεις χρησιμοποιείται για τη βελτίωση της κυκλοφορίας, την εξοικονόμηση ενέργειας και τη διαχείριση απορριμμάτων.

Η έρευνα στον τομέα του IoT συνεχώς διευρύνεται, εξετάζοντας ζητήματα όπως η ασφάλεια δεδομένων, η ενεργειακή απόδοση συσκευών και η βελτίωση της διαλειτουργικότητας. Παράλληλα, υπάρχει έντονο ενδιαφέρον για τη χρήση τεχνολογιών τεχνητής νοημοσύνης και μηχανικής μάθησης, οι οποίες μπορούν να ενισχύσουν την ανάλυση των δεδομένων και την αυτοματοποίηση ενεργειών στις πλατφόρμες IoT. Ειδική έμφαση δίνεται, επίσης, στη δημιουργία ανοιχτών προτύπων, που να εξασφαλίζουν τη συνεργασία συσκευών από διαφορετικούς κατασκευαστές.

Η ανάπτυξη και η διάδοση των πλατφορμών IoT αναδεικνύουν τη σημασία της συνεργασίας μεταξύ της έρευνας και της πρακτικής εφαρμογής. Από τη μία πλευρά, η έρευνα συμβάλλει στην επίλυση τεχνικών προβλημάτων και στην ανάπτυξη νέων λειτουργιών. Από την άλλη πλευρά, οι πρακτικές

εφαρμογές προσφέρουν πολύτιμη ανατροφοδότηση για τη βελτίωση των τεχνολογιών. Οι εμπορικές πλατφόρμες όπως οι AWS IoT, Google Cloud IoT και Microsoft Azure IoT, καθώς και οι λύσεις ανοιχτού κώδικα όπως το ThingsBoard, ενσωματώνουν αυτές τις εξελίξεις, παρέχοντας καινοτόμες δυνατότητες στους χρήστες τους.

Οι πλατφόρμες IoT αποτελούν το κλειδί για την πλήρη αξιοποίηση των δυνατοτήτων του Διαδικτύου των Πραγμάτων. Παρέχουν ένα ευέλικτο και ασφαλές περιβάλλον για τη διαχείριση συσκευών και την αξιοποίηση δεδομένων σε πραγματικό χρόνο. Η έρευνα στον τομέα αυτό συνεχίζει να προωθεί τη βελτίωση της τεχνολογίας, ανοίγοντας τον δρόμο για νέες εφαρμογές και καινοτομίες [1].

2.2 Πλατφόρμες διαχείρισης διαδικτυακών συσκευών

Θα εξετάσουμε τις παρακάτω τέσσερις πλατφόρμες, οι οποίες καλύπτουν διαφορετικές ανάγκες και προσφέρουν ποικίλες δυνατότητες:

1. **AWS IoT Core**: Η πλατφόρμα IoT της Amazon που προσφέρει επεκτασιμότητα, ασφάλεια και ενσωμάτωση με υπηρεσίες cloud για αποθήκευση και ανάλυση δεδομένων.
2. **Google Cloud IoT**: Η λύση της Google που επικεντρώνεται στη διαχείριση συσκευών, την ανάλυση δεδομένων και την ενσωμάτωση τεχνητής νοημοσύνης για πιο εξελιγμένες εφαρμογές.
3. **Microsoft Azure IoT Hub**: Η πλατφόρμα της Microsoft που προσφέρει ευέλικτη υποστήριξη για διαφορετικά πρωτόκολλα και δυνατότητες διαχείρισης συσκευών, ιδανική για επιχειρησιακές λύσεις.
4. **ThingsBoard**: Μια λύση ανοιχτού κώδικα που επικεντρώνεται στη διαλειτουργικότητα, την προσαρμοστικότητα και την απλότητα, κατάλληλη για μικρότερες εφαρμογές και ανεξάρτητους προγραμματιστές.

2.2.1 AWS IoT Core

Το AWS IoT Core είναι η λύση του Amazon για την υποστήριξη εφαρμογών IoT, προσφέροντας ένα επεκτάσιμο και ασφαλές περιβάλλον για τη διαχείριση συσκευών και τη συλλογή δεδομένων. Η πλατφόρμα παρέχει τη δυνατότητα να συνδέονται εκατομμύρια συσκευές, να ανταλλάσσουν δεδομένα σε πραγματικό χρόνο και να ενεργοποιούν διαδικασίες βάσει των δεδομένων αυτών. Ένα από τα βασικά χαρακτηριστικά της AWS IoT Core είναι η δυνατότητα ασφαλούς επικοινωνίας, χρησιμοποιώντας

πρωτόκολλα όπως το MQTT, HTTP και WebSocket. Επιπλέον, παρέχεται ενσωμάτωση με υπηρεσίες κρυπτογράφησης και έλεγχο ταυτότητας μέσω πιστοποιητικών, εξασφαλίζοντας την εμπιστευτικότητα και την ακεραιότητα των δεδομένων.

Μία από τις ισχυρότερες πτυχές της AWS IoT Core είναι η βαθιά ενσωμάτωσή της με το οικοσύστημα AWS, επιτρέποντας στους χρήστες να αξιοποιήσουν υπηρεσίες όπως το AWS Lambda για την επεξεργασία δεδομένων, το Amazon S3 για την αποθήκευση δεδομένων και το Amazon SageMaker για την ανάπτυξη μοντέλων τεχνητής νοημοσύνης. Αυτή η ενσωμάτωση καθιστά την πλατφόρμα ιδανική για εφαρμογές που απαιτούν σύνθετες επεξεργασίες δεδομένων και ενσωμάτωση με άλλες υπηρεσίες cloud. Οι χρήστες μπορούν εύκολα να δημιουργούν αυτοματοποιημένες διαδικασίες, όπως η ανάλυση δεδομένων από αισθητήρες και η δημιουργία αναφορών.

Ένα άλλο χαρακτηριστικό που ξεχωρίζει στην AWS IoT Core είναι η δυνατότητα οπτικοποίησης και ανάλυσης δεδομένων μέσω της υπηρεσίας AWS IoT Analytics. Η πλατφόρμα επιτρέπει τη συλλογή δεδομένων από συσκευές IoT, τον καθαρισμό και την αποθήκευσή τους σε data lakes, ενώ παράλληλα παρέχει εργαλεία για τη δημιουργία σύνθετων αναφορών και γραφημάτων. Αυτή η δυνατότητα είναι ιδιαίτερα χρήσιμη για επιχειρήσεις που θέλουν να αποκτήσουν βαθύτερη κατανόηση της απόδοσης των συσκευών τους και να βελτιστοποιήσουν τις διαδικασίες τους.

Παρόλο που η AWS IoT Core προσφέρει εξαιρετικές δυνατότητες και επεκτασιμότητα, απευθύνεται κυρίως σε μεγάλες επιχειρήσεις με υψηλές απαιτήσεις. Το κόστος χρήσης της πλατφόρμας είναι σημαντικό και εξαρτάται από τον αριθμό των συνδεδεμένων συσκευών, τον όγκο των δεδομένων που διαχειρίζονται και τις υπηρεσίες που χρησιμοποιούνται. Αυτό καθιστά την πλατφόρμα λιγότερο κατάλληλη για μικρότερες επιχειρήσεις ή ανεξάρτητους προγραμματιστές που αναζητούν οικονομικές λύσεις.

Το AWS IoT Core αποτελεί μια κορυφαία επιλογή για οργανισμούς που χρειάζονται μια επεκτάσιμη, ασφαλή και ενσωματωμένη λύση για το IoT. Με την ισχυρή υποστήριξη που παρέχει το οικοσύστημα AWS και τις προηγμένες δυνατότητες ανάλυσης, η πλατφόρμα είναι ιδανική για επιχειρήσεις που θέλουν να επενδύσουν σε λύσεις IoT υψηλής ποιότητας. Η AWS IoT Core συνεχίζει να εξελίσσεται, ενσωματώνοντας νέες τεχνολογίες και προσφέροντας περισσότερες δυνατότητες για την υποστήριξη προηγμένων εφαρμογών [2].

2.2.2 Google Cloud IoT

Το Google Cloud IoT είναι η λύση της Google για την υποστήριξη εφαρμογών IoT, παρέχοντας ένα ισχυρό και ευέλικτο περιβάλλον για τη διαχείριση συσκευών και τη συλλογή δεδομένων. Η πλατφόρμα επικεντρώνεται στη σύνδεση, την παρακολούθηση και τη διαχείριση συσκευών IoT σε μεγάλη

κλίμακα, ενώ ενσωματώνεται απρόσκοπτα με το ευρύτερο οικοσύστημα υπηρεσιών Google Cloud. Με δυνατότητες που καλύπτουν από τη συλλογή δεδομένων μέχρι την ανάλυσή τους μέσω τεχνητής νοημοσύνης, το Google Cloud IoT αποτελεί μία από τις πιο ολοκληρωμένες λύσεις για IoT εφαρμογές.

Ένα από τα βασικά χαρακτηριστικά της πλατφόρμας είναι το IoT Core, το οποίο επιτρέπει τη σύνδεση και τη διαχείριση εκατομμυρίων συσκευών. Το IoT Core υποστηρίζει διάφορα πρωτόκολλα επικοινωνίας, όπως MQTT και HTTP, και παρέχει εργαλεία για την ασφαλή ταυτοποίηση και εξουσιοδότηση συσκευών μέσω κρυπτογράφησης TLS. Παράλληλα, οι χρήστες έχουν τη δυνατότητα να παρακολουθούν την κατάσταση των συσκευών τους σε πραγματικό χρόνο, να λαμβάνουν ειδοποιήσεις για πιθανές δυσλειτουργίες και να ενημερώνουν το λογισμικό των συσκευών εξ αποστάσεως.

Μία από τις ισχυρές πτυχές του Google Cloud IoT είναι η ενσωμάτωσή του με υπηρεσίες ανάλυσης δεδομένων, όπως το BigQuery και το Dataflow, καθώς και με εργαλεία μηχανικής μάθησης, όπως το Vertex AI. Μέσω αυτών των εργαλείων, οι χρήστες μπορούν να επεξεργάζονται τεράστιες ποσότητες δεδομένων, να δημιουργούν προβλέψεις και να λαμβάνουν τεκμηριωμένες αποφάσεις. Για παράδειγμα, οι χρήστες μπορούν να αναλύσουν τις πληροφορίες που συλλέγονται από αισθητήρες, ώστε να βελτιστοποιήσουν την αποδοτικότητα μιας διαδικασίας ή να ανιχνεύσουν ανωμαλίες που μπορεί να υποδεικνύουν προβλήματα.

Η πλατφόρμα Google Cloud IoT υποστηρίζει επίσης την οπτικοποίηση δεδομένων μέσω του Google Data Studio, επιτρέποντας στους χρήστες να δημιουργούν διαδραστικούς πίνακες ελέγχου και γραφήματα. Αυτή η δυνατότητα είναι ιδιαίτερα χρήσιμη για επιχειρήσεις που χρειάζονται ένα φιλικό και ευέλικτο εργαλείο για την ανάλυση και την παρουσίαση δεδομένων. Επιπλέον, η πλατφόρμα ενσωματώνεται με υπηρεσίες όπως το Google Maps, παρέχοντας γεωγραφική απεικόνιση δεδομένων, κάτι που είναι ιδανικό για εφαρμογές logistics και παρακολούθησης περιουσιακών στοιχείων.

Ωστόσο, το Google Cloud IoT, όπως και άλλες εμπορικές πλατφόρμες, παρουσιάζει προκλήσεις, όπως η πολυπλοκότητα στην αρχική ρύθμιση και το υψηλό κόστος για επιχειρήσεις με περιορισμένους πόρους. Ενώ η πλατφόρμα προσφέρει εξαιρετικές δυνατότητες επεκτασιμότητας και ευελιξίας, οι χρήστες χρειάζονται τεχνογνωσία για να αξιοποιήσουν πλήρως τις υπηρεσίες της. Παράλληλα, το κόστος κλιμακώνεται ανάλογα με τον όγκο δεδομένων και τον αριθμό των συσκευών που συνδέονται.

Το Google Cloud IoT είναι μια ισχυρή πλατφόρμα που συνδυάζει επεκτασιμότητα, ευελιξία και ενσωμάτωση προηγμένων τεχνολογιών. Είναι ιδανική για επιχειρήσεις που επιδιώκουν να επεξεργάζονται δεδομένα σε μεγάλη κλίμακα και να ενσωματώνουν τεχνητή νοημοσύνη στις διαδικασίες τους. Με τις συνεχείς εξελίξεις και τις καινοτομίες που εισάγει η Google, η πλατφόρμα IoT παραμένει μία από τις κορυφαίες επιλογές για εφαρμογές IoT σε παγκόσμιο επίπεδο [3].

2.2.3 Microsoft Azure IoT Hub

Το Microsoft Azure IoT Hub αποτελεί την πρόταση της Microsoft για τη διαχείριση συσκευών και δεδομένων στο οικοσύστημα του Internet of Things (IoT). Πρόκειται για μια πλήρως διαχειριζόμενη υπηρεσία cloud που επιτρέπει τη σύνδεση, την παρακολούθηση και τον έλεγχο δισεκατομμυρίων συσκευών IoT. Το Azure IoT Hub διακρίνεται για την ευελιξία του στη χρήση διαφορετικών πρωτοκόλλων επικοινωνίας και τη δυνατότητα προσαρμογής στις ιδιαίτερες ανάγκες κάθε εφαρμογής. Η πλατφόρμα παρέχει στους χρήστες ισχυρά εργαλεία για την παρακολούθηση συσκευών, την ανταλλαγή μηνυμάτων σε πραγματικό χρόνο και την εφαρμογή μέτρων ασφάλειας σε μεγάλη κλίμακα.

Ένα από τα σημαντικότερα χαρακτηριστικά της πλατφόρμας είναι η υποστήριξη πολλών πρωτοκόλλων, όπως το MQTT, το HTTP και το AMQP. Αυτή η ευελιξία την καθιστά κατάλληλη για εφαρμογές με ετερογενείς συσκευές και απαιτήσεις. Επιπλέον, η πλατφόρμα παρέχει εργαλεία όπως το Device Twins, το οποίο αποθηκεύει την κατάσταση κάθε συσκευής και επιτρέπει τη διαχείρισή της ακόμη και όταν είναι προσωρινά αποσυνδεδεμένη. Το χαρακτηριστικό αυτό διευκολύνει την απομακρυσμένη συντήρηση συσκευών, προσφέροντας μια αξιόπιστη λύση για εφαρμογές που απαιτούν υψηλή διαθεσιμότητα.

Μία από τις σημαντικές δυνατότητες του Azure IoT Hub είναι η ενσωμάτωσή του με άλλες υπηρεσίες της Microsoft, όπως το Azure Stream Analytics και το Azure Machine Learning. Μέσω αυτών των εργαλείων, οι χρήστες μπορούν να αναλύουν δεδομένα σε πραγματικό χρόνο, να δημιουργούν προβλεπτικά μοντέλα και να αυτοματοποιούν διαδικασίες. Για παράδειγμα, μια εφαρμογή IoT μπορεί να χρησιμοποιήσει δεδομένα από αισθητήρες για να προβλέψει πότε μια μηχανή χρειάζεται συντήρηση, μειώνοντας έτσι το κόστος και αυξάνοντας την αποδοτικότητα.

Η πλατφόρμα δίνει επίσης έμφαση στην ασφάλεια, παρέχοντας δυνατότητες όπως η χρήση πιστοποιητικών, η κρυπτογράφηση δεδομένων και η διαχείριση πρόσβασης. Επιπλέον, το Azure IoT Hub υποστηρίζει Role-Based Access Control (RBAC), επιτρέποντας στους διαχειριστές να ορίζουν λεπτομερείς ρόλους και δικαιώματα για κάθε χρήστη ή ομάδα. Αυτή η δυνατότητα είναι ιδιαίτερα σημαντική για μεγάλους οργανισμούς που χρειάζονται έναν σαφή και ασφαλή διαχωρισμό των δικαιωμάτων πρόσβασης.

Παρά την εκτεταμένη λειτουργικότητα της πλατφόρμας, το Azure IoT Hub παρουσιάζει ορισμένες προκλήσεις, όπως η πολυπλοκότητα στη ρύθμιση και η ανάγκη για εξειδικευμένη τεχνογνωσία. Οι χρήστες που δεν είναι εξοικειωμένοι με το οικοσύστημα Azure μπορεί να χρειαστούν πρόσθετη υποστήριξη για την πλήρη αξιοποίηση των δυνατοτήτων της πλατφόρμας. Επίσης, το κόστος χρήσης

εξαρτάται από τον αριθμό των συσκευών και τον όγκο των δεδομένων, κάτι που πρέπει να λαμβάνεται υπόψη για την ανάπτυξη εφαρμογών μεγάλης κλίμακας.

Το Microsoft Azure IoT Hub αποτελεί μια ισχυρή και ευέλικτη πλατφόρμα, ιδανική για επιχειρήσεις που χρειάζονται λύσεις υψηλής επεκτασιμότητας και διαλειτουργικότητας. Η στενή ενσωμάτωση με τις υπόλοιπες υπηρεσίες Azure και οι προηγμένες δυνατότητες ανάλυσης το καθιστούν μια από τις κορυφαίες επιλογές για IoT εφαρμογές, ενώ οι συνεχιζόμενες βελτιώσεις από τη Microsoft διασφαλίζουν την αντοχή της πλατφόρμας στις μελλοντικές απαιτήσεις [4].

2.2.4 ThingsBoard

Το ThingsBoard είναι μια λύση ανοιχτού κώδικα που επικεντρώνεται στη διαχείριση και οπτικοποίηση δεδομένων IoT. Πρόκειται για μία από τις πιο δημοφιλείς πλατφόρμες στον χώρο του IoT, ιδανική για οργανισμούς που επιθυμούν να διατηρήσουν τον έλεγχο της υποδομής τους και να προσαρμόσουν τη λειτουργικότητα της πλατφόρμας στις ιδιαίτερες ανάγκες τους. Το ThingsBoard παρέχει πλούσια χαρακτηριστικά που καλύπτουν από τη διαχείριση συσκευών έως την ανάλυση δεδομένων και τη δημιουργία διαδραστικών πινάκων ελέγχου.

Η πλατφόρμα υποστηρίζει ευρέως χρησιμοποιούμενα πρωτόκολλα επικοινωνίας όπως MQTT, CoAP και HTTP, διευκολύνοντας τη σύνδεση συσκευών από διαφορετικούς κατασκευαστές. Ένα από τα βασικά πλεονεκτήματά της είναι η ευελιξία στη διαχείριση δεδομένων. Οι χρήστες μπορούν να αποθηκεύουν, να επεξεργάζονται και να οπτικοποιούν δεδομένα από συσκευές σε πραγματικό χρόνο, ενώ παράλληλα υποστηρίζονται πολύπλοκοι κανόνες και σενάρια αυτοματισμού μέσω του ενσωματωμένου Rule Engine. Το εργαλείο αυτό επιτρέπει τη δημιουργία προσαρμοσμένων ενεργειών που ενεργοποιούνται βάσει συγκεκριμένων γεγονότων ή συνθηκών.

Ένα από τα κύρια χαρακτηριστικά του ThingsBoard είναι οι διαδραστικοί πίνακες ελέγχου. Οι χρήστες μπορούν να δημιουργούν εξατομικευμένα dashboards που παρουσιάζουν δεδομένα σε γραφήματα, πίνακες ή χάρτες. Αυτή η δυνατότητα είναι ιδιαίτερα χρήσιμη για οργανισμούς που θέλουν να παρακολουθούν την απόδοση των συσκευών τους ή να αναλύουν κρίσιμες πληροφορίες σε πραγματικό χρόνο. Η δυνατότητα οπτικοποίησης δεδομένων καθιστά την πλατφόρμα ιδιαίτερα φιλική προς τον χρήστη, ακόμα και για άτομα χωρίς εξειδικευμένες τεχνικές γνώσεις.

Το ThingsBoard δίνει επίσης μεγάλη έμφαση στην επεκτασιμότητα και την προσαρμοστικότητα. Η πλατφόρμα μπορεί να εγκατασταθεί είτε σε τοπικούς διακομιστές είτε σε υποδομές cloud, επιτρέποντας στους χρήστες να επιλέξουν τη λύση που ταιριάζει καλύτερα στις ανάγκες τους. Η υποστήριξη multi-

tenant αρχιτεκτονικής διευκολύνει τη χρήση της πλατφόρμας από μεγάλους οργανισμούς που χρειάζονται διαχωρισμό δεδομένων και λειτουργιών μεταξύ διαφορετικών ομάδων ή τμημάτων.

Παρόλο που το ThingsBoard είναι μια ισχυρή λύση, απαιτεί εξειδικευμένη τεχνική γνώση για τη ρύθμιση και τη συντήρησή του. Η πολυπλοκότητα της πλατφόρμας μπορεί να αποτελέσει πρόκληση για οργανισμούς που δεν διαθέτουν εξειδικευμένο προσωπικό IT. Επίσης, αν και η ανοιχτή φύση της πλατφόρμας επιτρέπει σημαντική ευελιξία, οι χρήστες πρέπει να επενδύσουν χρόνο για να προσαρμόσουν την πλατφόρμα στις ανάγκες τους.

Το ThingsBoard είναι μια εξαιρετική επιλογή για οργανισμούς που επιθυμούν μια προσαρμόσιμη και οικονομική λύση IoT. Η ανοιχτή φύση της πλατφόρμας, η ευελιξία στη χρήση και οι προηγμένες δυνατότητες οπτικοποίησης και αυτοματισμού την καθιστούν ιδανική για μικρές και μεσαίες επιχειρήσεις, καθώς και για ερευνητικά έργα. Η συνεχιζόμενη ανάπτυξη και υποστήριξη από την κοινότητα του ThingsBoard εξασφαλίζει ότι η πλατφόρμα παραμένει επίκαιρη και συμβατή με τις σύγχρονες ανάγκες του IoT [5].

Κεφάλαιο 3ο: Τι χρησιμοποιήσαμε για την υλοποίηση του έργου

3.1 Python

Η Python είναι μία από τις πιο διαδεδομένες γλώσσες προγραμματισμού στον κόσμο, με μια πληθώρα εφαρμογών που καλύπτουν διάφορους τομείς, όπως η ανάπτυξη λογισμικού, η επιστημονική έρευνα, η ανάλυση δεδομένων, και η μηχανική μάθηση. Εισήχθη το 1991 από τον Guido van Rossum και έκτοτε έχει εξελιχθεί σε ένα εργαλείο που χαρακτηρίζεται από την απλότητα, την αναγνωσιμότητα του κώδικα, και την ισχυρή κοινότητα υποστήριξης.

Χαρακτηριστικά της Python

Η Python χρησιμοποιεί μια συντακτική δομή που προσεγγίζει την ανθρώπινη γλώσσα, γεγονός που την καθιστά εύκολη στη μάθηση και τη χρήση. Αυτό την καθιστά ιδανική για αρχάριους προγραμματιστές, ενώ παράλληλα επιτρέπει στους έμπειρους προγραμματιστές να αναπτύσσουν σύνθετα έργα γρήγορα και αποτελεσματικά.
Η Python είναι συμβατή με διάφορες πλατφόρμες, όπως Windows, macOS και Linux, και μπορεί να εκτελείται σχεδόν σε οποιοδήποτε σύστημα χωρίς τροποποιήσεις.

Με μια εκτεταμένη συλλογή από ενσωματωμένες βιβλιοθήκες και πακέτα τρίτων, η Python υποστηρίζει εφαρμογές που κυμαίνονται από την ανάπτυξη ιστοσελίδων (Django, Flask) μέχρι την τεχνητή νοημοσύνη (TensorFlow, PyTorch).

Η Python χρησιμοποιεί δυναμική τυποποίηση, επιτρέποντας τη γρήγορη ανάπτυξη κώδικα χωρίς την ανάγκη καθορισμού τύπων δεδομένων.

Η Python διαθέτει μια από τις μεγαλύτερες κοινότητες προγραμματιστών, παρέχοντας υποστήριξη, εργαλεία, και εκτενή τεκμηρίωση.

Χρήση της Python στην Πλατφόρμα μας

Η Python επιλέχθηκε για την ανάπτυξη της πλατφόρμας διαχείρισης συσκευών λόγω των εξής πλεονεκτημάτων:

Γρήγορη Ανάπτυξη: Η απλότητα της Python επέτρεψε τη γρήγορη δημιουργία λειτουργικών μονάδων.

Ενσωμάτωση με Βάση Δεδομένων: Η Python παρέχει ισχυρά εργαλεία, όπως το pymysql, για τη σύνδεση και διαχείριση δεδομένων από βάσεις δεδομένων MySQL.

Υποστήριξη API: Η βιβλιοθήκη Flask χρησιμοποιήθηκε για τη δημιουργία RESTful APIs, διευκολύνοντας την επικοινωνία με εξωτερικές εφαρμογές.

Σύνδεση με Βάση Δεδομένων

```
import pymysql
def get_db_connection():
    return pymysql.connect(
        host='localhost',
        user='root',
        password='',
        database='psomathianos',
        cursorclass=pymysql.cursors.DictCursor
    )
```

Το παραπάνω απόσπασμα κώδικα δημιουργεί σύνδεση με τη βάση δεδομένων MySQL.

Δημιουργία Endpoint για API

```
@app.route('/api/device/read', methods=['GET'])
def api_device_read():
    apiread = request.args.get('apiread')
    connection = get_db_connection()
    try:
        with connection.cursor() as cursor:
            sql = "SELECT * FROM devices WHERE apiread = %s"
            cursor.execute(sql, (apiread,))
            device = cursor.fetchone()
            return jsonify(device)
```

```
    finally:
        connection.close()
```

Αυτό το endpoint επιτρέπει την ανάγνωση δεδομένων μέσω ενός apiread key.

Η Python ξεκίνησε στις αρχές της δεκαετίας του 1990 ως ένα έργο προσωπικής πρωτοβουλίας. Από τότε:

- Το 2000 κυκλοφόρησε η Python 2.0, εισάγοντας σημαντικές βελτιώσεις.
- Το 2008, η Python 3.0 εισήγαγε νέες δυνατότητες και βελτιώσεις, αλλάζοντας παράλληλα τη συμβατότητα με παλαιότερες εκδόσεις.

Σήμερα, η Python παραμένει στην κορυφή των δημοφιλέστερων γλωσσών προγραμματισμού παγκοσμίως.

Η Python αποτελεί αναπόσπαστο μέρος της πλατφόρμας μας, επιτρέποντας τη γρήγορη ανάπτυξη και τη λειτουργική ευελιξία. Η ισχυρή κοινότητα και η διαθεσιμότητα εργαλείων την καθιστούν ιδανική για κάθε είδους έργο, από μικρές εφαρμογές έως μεγάλα συστήματα [6-7].

3.2 Flask

Το Flask είναι ένα ελαφρύ web framework για την Python, σχεδιασμένο να παρέχει απλότητα και ευελιξία. Δημιουργήθηκε από τον Armin Ronacher το 2010 και βασίζεται στη βιβλιοθήκη Werkzeug για τη διαχείριση αιτημάτων και στο Jinja2 για τη δημιουργία δυναμικών προτύπων HTML. Το Flask είναι ιδιαίτερα δημοφιλές για την ανάπτυξη RESTful APIs και εφαρμογών μικρής έως μεσαίας κλίμακας [8-9].

Χαρακτηριστικά του Flask

1. **Ελαφρύ και Επεκτάσιμο**

   Το Flask δεν επιβάλλει τη χρήση συγκεκριμένων εργαλείων ή βιβλιοθηκών, προσφέροντας στον προγραμματιστή πλήρη ελευθερία επιλογών.

2. **Υποστήριξη RESTful APIs**

   Το Flask διευκολύνει τη δημιουργία RESTful APIs, καθιστώντας το ιδανικό για συστήματα που απαιτούν επικοινωνία μεταξύ πελατών και διακομιστών.

3. **Δυναμικά HTML Πρότυπα**

   Η ενσωμάτωση του Jinja2 επιτρέπει τη δημιουργία δυναμικών HTML σελίδων που ανταποκρίνονται στα δεδομένα του backend.

4. **Εκτεταμένο Οικοσύστημα Επεκτάσεων**

   Μέσω επεκτάσεων, όπως το Flask-SQLAlchemy και το Flask-Login, το Flask μπορεί να υποστηρίξει λειτουργίες όπως η σύνδεση σε βάσεις δεδομένων και η διαχείριση συνεδριών.

Χρήση του Flask στην Πλατφόρμα μας

Η πλατφόρμα μας χρησιμοποιεί το Flask για:

- Τη δημιουργία δυναμικών ιστοσελίδων.
- Την ανάπτυξη endpoints για API.
- Τη διαχείριση συνεδριών χρηστών.

Παράδειγμα Εφαρμογής Flask

Απλό Endpoint:

```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/')

def home():

    return jsonify({'message': 'Καλωσορίσατε στην πλατφόρμα μας!'})


if __name__ == '__main__':

    app.run(debug=True)
```

Το / είναι το βασικό endpoint της εφαρμογής.

Η μέθοδος jsonify επιστρέφει δεδομένα σε μορφή JSON.

2. **Διαχείριση Αιτημάτων POST:**

```
@app.route('/add', methods=['POST'])
def add_data():
    data = request.get_json()
    return jsonify({'received': data})
```

Το endpoint /add δέχεται δεδομένα σε μορφή JSON μέσω HTTP POST.

Επιστρέφει τα δεδομένα που λήφθηκαν ως απόκριση.

Η πλατφόρμα μας χρησιμοποιεί το Flask για πιο σύνθετες λειτουργίες, όπως η διαχείριση χρηστών και δεδομένων συσκευών. Χρησιμοποιούμε το Flask για:

- **Διαχείριση Συνδέσεων Χρηστών:** Τα endpoints για εγγραφή, σύνδεση και αποσύνδεση βασίζονται στη διαχείριση συνεδριών του Flask.
- **API για Δεδομένα Συσκευών:** Τα RESTful APIs που επιτρέπουν την ανάγνωση και την ενημέρωση δεδομένων συσκευών δημιουργήθηκαν με το Flask.

3.3 MySQL

Η MySQL είναι ένα από τα πιο διαδεδομένα συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS) στον κόσμο. Χρησιμοποιείται ευρέως για την αποθήκευση και διαχείριση δεδομένων σε εφαρμογές μικρής και μεγάλης κλίμακας, χάρη στην αξιοπιστία, την ταχύτητα και την ευκολία χρήσης της. Στην πλατφόρμα μας, η MySQL επιλέχθηκε ως το κύριο εργαλείο για τη διαχείριση δεδομένων, εξασφαλίζοντας αποτελεσματική οργάνωση και ανάκτηση πληροφοριών.

Χαρακτηριστικά της MySQL

1. **Αξιοπιστία και Ταχύτητα**

   Η MySQL είναι γνωστή για την υψηλή της ταχύτητα στην επεξεργασία δεδομένων, καθώς και για την αξιοπιστία της σε εφαρμογές παραγωγής.

2. **Υποστήριξη Σχεσιακού Μοντέλου**

Η MySQL χρησιμοποιεί ένα σχεσιακό μοντέλο βάσης δεδομένων, το οποίο επιτρέπει την αποθήκευση δεδομένων σε πίνακες και τη διασύνδεσή τους μέσω σχέσεων (π.χ. primary και foreign keys).

3. **Επεκτασιμότητα**

   Αν και ελαφριά, η MySQL μπορεί να κλιμακωθεί για να υποστηρίξει βάσεις δεδομένων με εκατομμύρια εγγραφές.

4. **Υποστήριξη για SQL**

   Η MySQL χρησιμοποιεί την Structured Query Language (SQL), ένα πρότυπο για τη διαχείριση δεδομένων, διευκολύνοντας την πρόσβαση και επεξεργασία τους.

5. **Συμβατότητα με Πλατφόρμες**

   Λειτουργεί σε μια μεγάλη ποικιλία λειτουργικών συστημάτων, όπως Windows, Linux, και macOS.

Χρήση της MySQL στην Πλατφόρμα μας

Η MySQL χρησιμοποιείται για:

- **Διαχείριση Χρηστών:**

  Αποθηκεύει πληροφορίες χρηστών, όπως ονόματα, κωδικούς πρόσβασης, και στοιχεία εγγραφής.

- **Διαχείριση Συσκευών:**

  Οργανώνει δεδομένα συσκευών, όπως ονόματα, περιγραφές, και τιμές αισθητήρων.

- **API Keys:**

  Κρατά τα API keys που χρησιμοποιούνται για την ασφαλή επικοινωνία με τις συσκευές.

Πλεονεκτήματα της MySQL στην Πλατφόρμα

Η MySQL εξασφαλίζει ταχύτατη ανάκτηση και αποθήκευση δεδομένων, κάτι που είναι κρίσιμο για την απόδοση της πλατφόρμας. Συνεργάζεται άψογα με την Python μέσω βιβλιοθηκών, όπως το pymysql, για τη σύνδεση και διαχείριση δεδομένων. Προσφέρει δυνατότητες ελέγχου πρόσβασης και κρυπτογράφησης για την προστασία των δεδομένων. Επίσης μπορεί να χειριστεί αυξανόμενα δεδομένα και χρήστες χωρίς σημαντική πτώση στην απόδοση.

Η MySQL αποτελεί βασικό στοιχείο της αρχιτεκτονικής της πλατφόρμας μας, παρέχοντας μια γρήγορη, αξιόπιστη και ασφαλή λύση για τη διαχείριση δεδομένων. Η ευελιξία της και η υποστήριξή της από την κοινότητα προγραμματιστών την καθιστούν ιδανική επιλογή για συστήματα IoT και διαχείρισης δεδομένων.

3.4 Bootstrap 5

Το Bootstrap είναι ένα από τα πιο δημοφιλή εργαλεία για τη δημιουργία responsive ιστοσελίδων. Πρόκειται για ένα framework ανοιχτού κώδικα που δημιουργήθηκε από το Twitter το 2011, και περιλαμβάνει έτοιμα CSS και JavaScript στοιχεία, τα οποία διευκολύνουν τον σχεδιασμό μοντέρνων, καλαίσθητων και λειτουργικών διεπαφών χρήστη [12].

Χαρακτηριστικά του Bootstrap

1. **Responsive Σχεδιασμός**

   Το Bootstrap χρησιμοποιεί ένα σύστημα πλέγματος (grid system), επιτρέποντας τη δημιουργία ιστοσελίδων που προσαρμόζονται σε διάφορες συσκευές, όπως κινητά τηλέφωνα, tablets, και υπολογιστές.

2. **Έτοιμα Στοιχεία UI**

   Περιλαμβάνει προκαθορισμένα στοιχεία, όπως κουμπιά, φόρμες, πλοήγηση, καρτέλες, και modal windows, τα οποία μπορούν να προσαρμοστούν εύκολα.

3. **Υποστήριξη JavaScript**

   Με την ενσωμάτωση JavaScript plugins, το Bootstrap προσφέρει διαδραστικά στοιχεία, όπως dropdown menus, carousels, και tooltips.

4. **Διαλειτουργικότητα με Εξωτερικά Εργαλεία**

   Το Bootstrap μπορεί να συνδυαστεί με βιβλιοθήκες όπως το jQuery και το ZingChart, καθιστώντας το ιδανικό για πιο σύνθετες εφαρμογές.

5. **Ευκολία Προσαρμογής**

   Παρέχει τη δυνατότητα αλλαγής των προεπιλεγμένων στυλ μέσω μεταβλητών CSS ή χρήσης εργαλείων όπως το Sass.

Χρήση του Bootstrap στην Πλατφόρμα μας

Η πλατφόρμα μας χρησιμοποιεί το Bootstrap για:

- **Responsive Διάταξη:**

  Όλες οι ιστοσελίδες προσαρμόζονται αυτόματα σε διαφορετικές συσκευές.

- **Ευκολία Σχεδιασμού:**

  Ενσωματώσαμε προκαθορισμένα στυλ για κουμπιά, φόρμες, και πίνακες.

- **Δυναμικές Ενότητες:**

  Χρησιμοποιήσαμε στοιχεία όπως modals και alerts για καλύτερη εμπειρία χρήστη.

3.5 ZingChart

Το ZingChart είναι ένα ευέλικτο εργαλείο οπτικοποίησης δεδομένων που επιτρέπει τη δημιουργία διαδραστικών και προσαρμοσμένων γραφημάτων. Υποστηρίζει μια πληθώρα τύπων γραφημάτων, όπως γραμμικά, πίτας, ράβδων και θερμικών χαρτών, καθιστώντας το ιδανικό για την απεικόνιση σύνθετων δεδομένων σε πραγματικό χρόνο. Με την ενσωμάτωση JavaScript, το ZingChart προσφέρει πλούσια λειτουργικότητα και ευκολία προσαρμογής, καθιστώντας το κατάλληλο για σύγχρονες διαδικτυακές εφαρμογές[14].

Χαρακτηριστικά του ZingChart

1. **Ποικιλία Τύπων Γραφημάτων** Το ZingChart υποστηρίζει περισσότερους από 35 τύπους γραφημάτων, όπως γραμμικά, ράβδων, θερμικών χαρτών και scatter plots, καλύπτοντας τις ανάγκες διαφορετικών περιπτώσεων χρήσης.
2. **Διαδραστικότητα** Παρέχει λειτουργίες όπως hover effects, tooltips, και zoom, δίνοντας στον χρήστη τη δυνατότητα να εξερευνά τα δεδομένα πιο αποτελεσματικά.
3. **Προσαρμοστικότητα** Το ZingChart μπορεί να προσαρμοστεί πλήρως μέσω JSON configuration, επιτρέποντας στον προγραμματιστή να ελέγχει κάθε πτυχή του γραφήματος.
4. **Υποστήριξη Πραγματικού Χρόνου** Το ZingChart μπορεί να ανανεώνει δυναμικά τα δεδομένα, καθιστώντας το ιδανικό για συστήματα που απαιτούν παρακολούθηση δεδομένων σε πραγματικό χρόνο.
5. **Responsive Σχεδιασμός** Τα γραφήματα προσαρμόζονται αυτόματα σε διαφορετικές διαστάσεις οθόνης, εξασφαλίζοντας καλή εμπειρία χρήστη σε κινητά, tablets και υπολογιστές.

Χρήση του ZingChart στην Πλατφόρμα μας

Η πλατφόρμα μας χρησιμοποιεί το ZingChart για:

- **Οπτικοποίηση Δεδομένων Συσκευών**

  Τα δεδομένα των αισθητήρων κάθε συσκευής εμφανίζονται σε διαδραστικά γραφήματα, επιτρέποντας στους χρήστες να εντοπίζουν μοτίβα ή ανωμαλίες.

- **Προσαρμογή Εμφάνισης**

  Τα γραφήματα προσαρμόζονται με τίτλους, υπότιτλους, ετικέτες και γραμμές πλέγματος, διευκολύνοντας την κατανόηση των δεδομένων.

- **Σύγκριση Τιμών**

  Πολλαπλές τιμές από διαφορετικά πεδία μπορούν να εμφανίζονται ταυτόχρονα, επιτρέποντας την εύκολη σύγκριση δεδομένων.

Πλεονεκτήματα του ZingChart

1. **Πλούσια Δυνατότητα Διαδραστικότητας** Χάρη σε χαρακτηριστικά όπως tooltips, markers και δυνατότητες zoom, οι χρήστες μπορούν να εξετάζουν τα δεδομένα λεπτομερώς.
2. **Υψηλή Απόδοση** Το ZingChart είναι βελτιστοποιημένο για να διαχειρίζεται μεγάλα σύνολα δεδομένων χωρίς να θυσιάζει την απόδοση.
3. **Προσαρμοσμένα Θέματα** Προσφέρει υποστήριξη για custom themes, καθιστώντας την οπτικοποίηση δεδομένων αισθητικά ευχάριστη και προσαρμοσμένη στο ύφος της εφαρμογής.
4. **Εύκολη Ενσωμάτωση** Η ενσωμάτωση του ZingChart σε ιστοσελίδες είναι απλή, με χρήση JavaScript και JSON για τη διαμόρφωση των γραφημάτων.
5. **Συνεχής Υποστήριξη** Η πλατφόρμα ZingChart υποστηρίζεται ενεργά, με συχνές ενημερώσεις και πλούσια τεκμηρίωση.

Παραδείγματα Χρήσης στην Πλατφόρμα

- **Γραμμικά Γραφήματα**: Απεικόνιση τιμών αισθητήρων, π.χ., θερμοκρασίας ή υγρασίας, σε σχέση με τον χρόνο.
- **Σύγκριση Δεδομένων**: Παρουσίαση πολλαπλών πεδίων αισθητήρων σε ένα μόνο γράφημα για εύκολη σύγκριση.

- **Προσαρμογή Εμφάνισης**: Κάθε γράφημα περιλαμβάνει τίτλους, ετικέτες αξόνων και υποσημειώσεις για καλύτερη παρουσίαση δεδομένων.

Το ZingChart αποτελεί ένα ισχυρό εργαλείο για την οπτικοποίηση δεδομένων στην πλατφόρμα μας, συνδυάζοντας διαδραστικότητα, ευελιξία και υψηλή απόδοση. Είναι ιδανικό για εφαρμογές IoT, όπου η παρακολούθηση και ανάλυση δεδομένων σε πραγματικό χρόνο είναι κρίσιμη.

Κεφάλαιο 4ο: Το έργο

4.1 Εισαγωγή στην πλατφόρμα

Η πλατφόρμα διαχείρισης IoT συσκευών σχεδιάστηκε για να προσφέρει ευκολία στη διαχείριση, προβολή και επεξεργασία δεδομένων δικτυωμένων συσκευών. Σε αυτό το κεφάλαιο, θα γίνει αναλυτική παρουσίαση της λειτουργικότητας και της αρχιτεκτονικής του κώδικα της εφαρμογής. Θα περιγραφούν οι βασικές λειτουργίες, όπως η σύνδεση χρηστών, η διαχείριση συσκευών, η ανάγνωση/εγγραφή δεδομένων μέσω API, καθώς και η απεικόνιση δεδομένων σε γραφήματα και πίνακες.

```
/project_directory/
│
├── app.py
│
├── templates/
│   ├── base.html
│   ├── home.html
│   ├── register.html
│   ├── login.html
│   ├── dashboard.html
│   ├── device_edit.html
│   ├── device_data.html
│
├── static/
│   ├── css/
│   │   ├── custom.css
│   ├── js/
│   │   ├── custom.js
```

```
│   ├── images/
│   │   ├── logo.png
│   │
└── database/
    ├── schema.sql
```

Εικόνα 4.1: Χάρτης με τα αρχεία του συστήματος-πλατφόρμας

Εισαγωγή στο app.py

Το αρχείο app.py αποτελεί την κύρια πηγή εκκίνησης της εφαρμογής και περιλαμβάνει τη διαχείριση όλων των endpoints, των λειτουργιών σύνδεσης στη βάση δεδομένων, και της επικοινωνίας με τα API.

```
from flask import Flask, request, jsonify, render_template, redirect, url_for, session
import pymysql


app = Flask(__name__)
app.secret_key = 'psomathianos_secret_key'
```

Βιβλιοθήκες: Χρησιμοποιείται το Flask ως web framework και το PyMySQL για την επικοινωνία με τη βάση δεδομένων.

secret_key: Χρησιμοποιείται για τη διαχείριση των συνεδριών (sessions).

Λειτουργία της Αρχικής Σελίδας

Η αρχική σελίδα της πλατφόρμας καλωσορίζει τον χρήστη και τον παρακινεί να συνδεθεί ή να εγγραφεί.

Κώδικας στο app.py:

```
@app.route('/')
def home():
    return render_template('home.html')
```

Περιεχόμενο στο home.html:

```
<div class="bg-light p-5 rounded text-center">
    <h1>Καλώς ήρθατε στην Πλατφόρμα διαχείρισης διαδικτυακών συσκευών</h1>
    <p class="lead">Διαχειριστείτε τις IoT διαδικτυακές συσκευές σας και δείτε δεδομένα σε πραγματικό χρόνο.</p>
    <hr class="my-4">
    <a class="btn btn-primary btn-lg me-2" href="{{ url_for('login') }}" role="button">Σύνδεση</a>
    <a class="btn btn-secondary btn-lg" href="{{ url_for('register') }}" role="button">Εγγραφή</a>
</div>
```

Περιγραφή: Παρουσιάζεται ένα φιλικό περιβάλλον καλωσορίσματος, ενθαρρύνοντας τον χρήστη να αλληλεπιδράσει με την πλατφόρμα.

Συνδέσεις: Τα κουμπιά "Σύνδεση" και "Εγγραφή" οδηγούν στις αντίστοιχες σελίδες μέσω των αντίστοιχων endpoints (login, register).

Διαχείριση Χρηστών: Εγγραφή και Σύνδεση

Η πλατφόρμα υποστηρίζει την εγγραφή νέων χρηστών και τη σύνδεση υπαρχόντων χρηστών.

Εγγραφή Χρηστών (register):

```
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'GET':
        return render_template('register.html')

    data = request.form
    username = data.get('username')
    password = data.get('password')
    firstname = data.get('firstname')
    lastname = data.get('lastname')
```

```
    if not username or not password or not firstname or not lastname:
        return jsonify({'error': 'Missing required fields'}), 400

    connection = get_db_connection()
    try:
        with connection.cursor() as cursor:
            sql = "INSERT INTO users (username, password, firstname, lastname) VALUES (%s, %s, %s, %s)"
            cursor.execute(sql, (username, password, firstname, lastname))
            connection.commit()
        return render_template('login.html', message="Registration successful. Please login.")
    finally:
        connection.close()
```

Περιεχόμενο στο register.html:

```
<div class="row justify-content-center">
    <div class="col-md-6">
        <h2>Εγγραφή</h2>
        <form method="POST" action="{{ url_for('register') }}">
            <div class="form-group mb-3">
                <label for="username" class="form-label">Όνομα Χρήστη</label>
                <input type="text" id="username" name="username" class="form-control" required>
            </div>
            <div class="form-group mb-3">
                <label for="password" class="form-label">Κωδικός</label>
                <input type="password" id="password" name="password" class="form-control" required>
```

```
        </div>
        <button type="submit" class="btn btn-primary">Εγγραφή</button>
    </form>
  </div>
</div>
```

Περιγραφή: Οι χρήστες εισάγουν όνομα χρήστη, κωδικό και άλλα στοιχεία για να εγγραφούν.

Βάση δεδομένων: Τα στοιχεία αποθηκεύονται στον πίνακα users.

Σύνδεση Χρηστών (login):

```
@app.route('/login', methods=['GET', 'POST'])
def login():
  if request.method == 'GET':
    return render_template('login.html')

  data = request.form
  username = data.get('username')
  password = data.get('password')

  connection = get_db_connection()
  try:
    with connection.cursor() as cursor:
      sql = "SELECT * FROM users WHERE username = %s AND password = %s"
      cursor.execute(sql, (username, password))
      user = cursor.fetchone()
      if user:
        session['user_id'] = user['id']
```

```
            session['username'] = user['username']
            return redirect(url_for('list_devices'))
        else:
            return render_template('login.html', error="Invalid username or password.")
    finally:
        connection.close()
```

Λειτουργικότητα: Οι χρήστες συνδέονται αν τα στοιχεία τους είναι σωστά. Τα δεδομένα τους αποθηκεύονται στο session.

Πίνακας Ελέγχου Συσκευών (Dashboard)

Ο πίνακας ελέγχου είναι η κεντρική σελίδα της πλατφόρμας, όπου εμφανίζονται όλες οι συσκευές που έχουν καταχωριστεί από τον συνδεδεμένο χρήστη. Από εδώ, ο χρήστης μπορεί να δει, να επεξεργαστεί, ή να διαγράψει συσκευές, καθώς και να προσθέσει νέες.

Κώδικας στο app.py για τον Πίνακα Ελέγχου:

```
@app.route('/dashboard', methods=['GET'])
def list_devices():
    if 'user_id' not in session:
        return redirect(url_for('login'))

    connection = get_db_connection()
    try:
        with connection.cursor() as cursor:
            sql = "SELECT * FROM devices WHERE userid = %s"
            cursor.execute(sql, (session['user_id'],))
            devices = cursor.fetchall()
        return render_template('dashboard.html', devices=devices, username=session.get('username'))
```

```
    finally:
        connection.close()
```

Λειτουργικότητα:

- Ελέγχεται αν ο χρήστης είναι συνδεδεμένος μέσω του session['user_id'].
- Αντλούνται από τη βάση δεδομένων οι συσκευές του χρήστη (userid).
- Οι συσκευές προβάλλονται στο πρότυπο dashboard.html.

HTML για τον Πίνακα Ελέγχου (dashboard.html):

```
{% extends "base.html" %}
{% block content %}
<h1 class="text-center my-4">Πίνακας Ελέγχου Συσκευών</h1>
<table class="table table-hover table-bordered">
    <thead class="table-dark">
        <tr>
            <th>Όνομα Συσκευής</th>
            <th>Περιγραφή</th>
            <th>Ενέργειες</th>
        </tr>
    </thead>
    <tbody>
        {% for device in devices %}
        <tr>
            <td>{{ device.namedevice }}</td>
            <td>{{ device.description }}</td>
            <td>
```

```
            <a href="{{ url_for('get_device', device_id=device.id) }}" class="btn btn-info btn-sm">Προβολή</a>
            <a href="{{ url_for('edit_device', device_id=device.id) }}" class="btn btn-warning btn-sm">Επεξεργασία</a>
            <form action="{{ url_for('delete_device', device_id=device.id) }}" method="POST" style="display:inline;">
                <button type="submit" class="btn btn-danger btn-sm">Διαγραφή</button>
            </form>
          </td>
        </tr>
        {% endfor %}
    </tbody>
</table>
<a href="{{ url_for('add_device') }}" class="btn btn-success">Προσθήκη Συσκευής</a>
{% endblock %}
```

Πίνακας Συσκευών:

- Περιέχει στήλες για το όνομα, την περιγραφή, και τις διαθέσιμες ενέργειες για κάθε συσκευή.
- Δίνεται η δυνατότητα προβολής, επεξεργασίας και διαγραφής μέσω αντίστοιχων κουμπιών.

Προσθήκη Συσκευής:

- Ένα κουμπί επιτρέπει στον χρήστη να κατευθυνθεί στη σελίδα προσθήκης νέας συσκευής.

Επεξεργασία Συσκευών (edit_device)

Η σελίδα επεξεργασίας επιτρέπει στον χρήστη να τροποποιήσει τα χαρακτηριστικά μιας συσκευής, όπως το όνομα, την περιγραφή, και τα API keys. Επίσης, μπορεί να ενεργοποιήσει ή να απενεργοποιήσει πεδία δεδομένων.

Κώδικας στο app.py:

```
@app.route('/device/<int:device_id>/edit', methods=['GET', 'POST'])
def edit_device(device_id):
    if 'user_id' not in session:
        return redirect(url_for('login'))

    connection = get_db_connection()
    try:
        with connection.cursor() as cursor:
            if request.method == 'GET':
                sql = "SELECT * FROM devices WHERE id = %s AND userid = %s"
                cursor.execute(sql, (device_id, session['user_id']))
                device = cursor.fetchone()
                if device:
                    return render_template('device_edit.html', device=device)
                else:
                    return redirect(url_for('list_devices'))

            data = request.form
            namedevice = data.get('namedevice')
            description = data.get('description')
            apiwrite = data.get('apiwrite')
            apiread = data.get('apiread')

            field_titles = [data.get(f'field{i}_title') for i in range(1, 5)]
            field_enabled = [data.get(f'field{i}_enabled') == 'on' for i in range(1, 5)]

            sql = """
```

```
            UPDATE devices
            SET namedevice = %s, description = %s, apiwrite = %s, apiread = %s,
                field1_title = %s, field1_value = %s,
                field2_title = %s, field2_value = %s,
                field3_title = %s, field3_value = %s,
                field4_title = %s, field4_value = %s
            WHERE id = %s AND userid = %s
        """
        cursor.execute(sql, (
            namedevice, description, apiwrite, apiread,
            field_titles[0], 1 if field_enabled[0] else 0,
            field_titles[1], 1 if field_enabled[1] else 0,
            field_titles[2], 1 if field_enabled[2] else 0,
            field_titles[3], 1 if field_enabled[3] else 0,
            device_id, session['user_id']
        ))
        connection.commit()
        return redirect(url_for('list_devices'))
    finally:
        connection.close()
```

HTML για Επεξεργασία Συσκευών (device_edit.html):

```
<h2>{{ 'Επεξεργασία Συσκευής' }}</h2>
<form method="POST">
    <label>Όνομα Συσκευής</label>
    <input type="text" name="namedevice" value="{{ device.namedevice }}" required>
    <label>Περιγραφή</label>
```

```
    <textarea name="description">{{ device.description }}</textarea>
    <label>API Write Key</label>
    <input type="text" name="apiwrite" value="{{ device.apiwrite }}">
    <label>API Read Key</label>
    <input type="text" name="apiread" value="{{ device.apiread }}">
    <button type="submit">Αποθήκευση</button>
</form>
```

Προβολή Δεδομένων Συσκευής και API Endpoints

Η πλατφόρμα περιλαμβάνει δυνατότητες προβολής δεδομένων μιας συσκευής σε πίνακες και γραφήματα, χρησιμοποιώντας δεδομένα από τα API. Στην ενότητα αυτή, θα αναλύσουμε τα **endpoints** για ανάγνωση και εγγραφή δεδομένων, καθώς και τη σελίδα προβολής.

Προβολή Δεδομένων Συσκευής

Ο χρήστης μπορεί να επιλέξει μια συσκευή από τον πίνακα ελέγχου για να δει τα δεδομένα της.

Κώδικας στο app.py:

```
@app.route('/device/<int:device_id>', methods=['GET'])
def get_device(device_id):
    if 'user_id' not in session:
        return redirect(url_for('login'))

    connection = get_db_connection()
    try:
        with connection.cursor() as cursor:
            sql_device = "SELECT * FROM devices WHERE id = %s AND userid = %s"
            cursor.execute(sql_device, (device_id, session['user_id']))
            device = cursor.fetchone()
```

```
        if not device:
            return redirect(url_for('list_devices'))

        sql_values = """
            SELECT createdat, field1_value, field2_value, field3_value, field4_value
            FROM devices
            WHERE id = %s ORDER BY createdat DESC
        """
        cursor.execute(sql_values, (device_id,))
        valuefields = cursor.fetchall()

        fields_with_titles = {
            f'field{i}_value': device[f'field{i}_title']
            for i in range(1, 5)
            if device[f'field{i}_title']
        }
    return render_template('device_data.html', device=device, valuefields=valuefields, fields_with_titles=fields_with_titles)
  finally:
    connection.close()
```

HTML για Προβολή Δεδομένων (device_data.html):

```
<div class="container my-5">
  <h2 class="text-center">Τιμές Συσκευής: {{ device.namedevice }}</h2>
  <div id="valuesChart" style="height: 400px;"></div>
  <table class="table table-striped mt-4">
    <thead>
      <tr>
```

```
            <th>Ημερομηνία</th>
            {% for field, title in fields_with_titles.items() %}
            <th>{{ title }}</th>
            {% endfor %}
        </tr>
    </thead>
    <tbody>
        {% for value in valuefields %}
        <tr>
            <td>{{ value.createdat }}</td>
            {% for field in fields_with_titles.keys() %}
            <td>{{ value[field] }}</td>
            {% endfor %}
        </tr>
        {% endfor %}
    </tbody>
</table>
<script>
    ZC.LICENSE = ["YOUR-ZINGCHART-LICENSE"];
    const valuesConfig = {
        type: 'line',
        title: { text: 'Δεδομένα Συσκευής', fontSize: 16 },
        series: [
            {% for field, title in fields_with_titles.items() %}
            { values: {{ valuefields|map(attribute=field)|list|tojson }}, text: '{{ title }}' },
            {% endfor %}
        ],
        scaleX: {
```

```
            labels: {{ valuefields|map(attribute='createdat')|list|tojson }},
            label: { text: 'Ημερομηνία' }
        },
        scaleY: { label: { text: 'Τιμές' } },
        legend: { draggable: true }
    };
    zingchart.render({
        id: 'valuesChart',
        data: valuesConfig,
        height: '100%',
        width: '100%'
    });
  </script>
</div>
```

Γράφημα: Εμφανίζει τις τιμές κάθε ενεργοποιημένου πεδίου (fieldX_value).

Πίνακας: Παρουσιάζει τις τιμές κάθε πεδίου σε συνδυασμό με την ημερομηνία καταγραφής.

API για Ενημέρωση Δεδομένων

Οι χρήστες ή εξωτερικές εφαρμογές μπορούν να ενημερώσουν δεδομένα συσκευών μέσω του apiwrite.

Κώδικας στο app.py:

```
@app.route('/api/device/write', methods=['GET'])
def api_device_write():
  apiwrite = request.args.get('apiwrite')
  field = request.args.get('field')
  value = request.args.get('value')

  if not apiwrite or not field or not value:
    return jsonify({'error': 'Missing parameters'}), 400
```

```
    connection = get_db_connection()
    try:
        with connection.cursor() as cursor:
            sql = "SELECT * FROM devices WHERE apiwrite = %s"
            cursor.execute(sql, (apiwrite,))
            device = cursor.fetchone()
            if not device:
                return jsonify({'error': 'Invalid API key'}), 403

            sql_update = f"UPDATE devices SET {field} = %s WHERE id = %s"
            cursor.execute(sql_update, (value, device['id']))
            connection.commit()
            return jsonify({'success': 'Value updated'})
    finally:
        connection.close()
```

API για Ανάγνωση Δεδομένων

Η ανάγνωση δεδομένων μιας συσκευής γίνεται μέσω του apiread.

Κώδικας στο app.py:

```
@app.route('/api/device/read', methods=['GET'])
def api_device_read():
    apiread = request.args.get('apiread')
    limit = int(request.args.get('limit', 10))

    if not apiread:
        return jsonify({'error': 'Missing API key'}), 400
```

```
    connection = get_db_connection()
    try:
        with connection.cursor() as cursor:
            sql = "SELECT * FROM devices WHERE apiread = %s"
            cursor.execute(sql, (apiread,))
            device = cursor.fetchone()
            if not device:
                return jsonify({'error': 'Invalid API key'}), 403


            sql_values = """
                SELECT createdat, field1_value, field2_value, field3_value, field4_value
                FROM devices WHERE id = %s ORDER BY createdat DESC LIMIT %s
            """
            cursor.execute(sql_values, (device['id'], limit))
            values = cursor.fetchall()
            return jsonify({'device': device['namedevice'], 'values': values})
    finally:
        connection.close()
```

Δημιουργία Νέων Συσκευών

Η πλατφόρμα επιτρέπει την προσθήκη νέων συσκευών, με δυνατότητα ρύθμισης του ονόματος, της περιγραφής και των API keys. Η λειτουργία αυτή είναι προσβάσιμη μέσω του κουμπιού Προσθήκη Συσκευής στον πίνακα ελέγχου.

Κώδικας στο app.py για Προσθήκη Συσκευών:

```
@app.route('/device/add', methods=['GET', 'POST'])
def add_device():
```

```
if 'user_id' not in session:
    return redirect(url_for('login'))

if request.method == 'GET':
    return render_template('device_edit.html', device=None)

data = request.form
userid = session['user_id']
namedevice = data.get('namedevice')
description = data.get('description')

# Δημιουργία τυχαίων API keys
apiwrite = ''.join(random.choices(string.ascii_letters + string.digits, k=32))
apiread = ''.join(random.choices(string.ascii_letters + string.digits, k=32))

if not namedevice:
    return jsonify({'error': 'Missing required fields'}), 400

connection = get_db_connection()
try:
    with connection.cursor() as cursor:
        sql = """
            INSERT INTO devices (userid, namedevice, description, apiwrite, apiread)
            VALUES (%s, %s, %s, %s, %s)
        """
        cursor.execute(sql, (userid, namedevice, description, apiwrite, apiread))
        connection.commit()
    return redirect(url_for('list_devices'))
```

```
    finally:
        connection.close()
```

Διαδικασία:

- Ελέγχεται αν ο χρήστης είναι συνδεδεμένος.
- Αν ο χρήστης υποβάλει τη φόρμα, δημιουργείται μια νέα συσκευή και αποθηκεύεται στη βάση δεδομένων.
- Τα API keys (apiwrite και apiread) δημιουργούνται αυτόματα.

HTML για Προσθήκη Συσκευών (device_edit.html):

```
<div class="container mt-4">
    <h2>{{ 'Προσθήκη Νέας Συσκευής' }}</h2>
    <form method="POST">
        <div class="mb-3">
            <label for="namedevice" class="form-label">Όνομα Συσκευής</label>
            <input type="text" class="form-control" id="namedevice" name="namedevice" required>
        </div>
        <div class="mb-3">
            <label for="description" class="form-label">Περιγραφή</label>
            <textarea class="form-control" id="description" name="description" rows="3"></textarea>
        </div>
        <button type="submit" class="btn btn-success">Προσθήκη</button>
    </form>
</div>
```

Φόρμα:

- Παρέχει πεδία για το όνομα και την περιγραφή της συσκευής.
- Τα API keys δεν εμφανίζονται, καθώς δημιουργούνται αυτόματα στο backend.

Επεξεργασία Συσκευών: Παράδειγμα Χρήσης

Αφού δημιουργηθεί μια συσκευή, ο χρήστης μπορεί να τη διαχειριστεί μέσω της σελίδας επεξεργασίας. Εδώ, ο χρήστης μπορεί:

- Να ενημερώσει το όνομα και την περιγραφή.
- Να τροποποιήσει τα API keys (αν απαιτείται).
- Να ρυθμίσει ποια πεδία είναι ενεργά.

Παράδειγμα:

@app.route('/device/<int:device_id>/edit', methods=['GET', 'POST'])

```
def edit_device(device_id):
    if 'user_id' not in session:
        return redirect(url_for('login'))

    connection = get_db_connection()
    try:
        with connection.cursor() as cursor:
            if request.method == 'GET':
                sql = "SELECT * FROM devices WHERE id = %s AND userid = %s"
                cursor.execute(sql, (device_id, session['user_id']))
                device = cursor.fetchone()
                if device:
                    return render_template('device_edit.html', device=device)
                else:
                    return redirect(url_for('list_devices'))

            data = request.form
            namedevice = data.get('namedevice')
            description = data.get('description')
```

```
        apiwrite = data.get('apiwrite')
        apiread = data.get('apiread')

        sql = """
            UPDATE devices
            SET namedevice = %s, description = %s, apiwrite = %s, apiread = %s
            WHERE id = %s AND userid = %s
        """
        cursor.execute(sql, (namedevice, description, apiwrite, apiread, device_id, session['user_id']))
        connection.commit()
        return redirect(url_for('list_devices'))
    finally:
        connection.close()
```

Διαγραφή Συσκευών και Παρακολούθηση Δεδομένων

Η πλατφόρμα παρέχει τη δυνατότητα διαγραφής συσκευών, καθώς και προβολής των δεδομένων τους σε πραγματικό χρόνο. Σε αυτή την ενότητα, θα αναλύσουμε τη λειτουργία διαγραφής συσκευών και την παρακολούθηση δεδομένων μέσω API.

Διαγραφή Συσκευών

Η λειτουργία διαγραφής επιτρέπει στον χρήστη να αφαιρέσει συσκευές που δεν είναι πλέον απαραίτητες. Η λειτουργία αυτή διασφαλίζει ότι μόνο ο χρήστης που δημιούργησε τη συσκευή έχει δικαίωμα διαγραφής.

Κώδικας στο app.py:

```
@app.route('/device/<int:device_id>/delete', methods=['POST'])
def delete_device(device_id):
    if 'user_id' not in session:
        return redirect(url_for('login'))
```

```
    connection = get_db_connection()
    try:
        with connection.cursor() as cursor:
            sql = "DELETE FROM devices WHERE id = %s AND userid = %s"
            cursor.execute(sql, (device_id, session['user_id']))
            connection.commit()
        return redirect(url_for('list_devices'))
    finally:
        connection.close()
```

Επεξήγηση:

- Η εντολή SQL διαγράφει τη συσκευή από τη βάση δεδομένων, εφόσον το userid της συσκευής ταιριάζει με το session['user_id'].

HTML για Διαγραφή Συσκευής

Στη σελίδα του **Dashboard**, η διαγραφή συσκευής ενεργοποιείται μέσω ενός κουμπιού.

Κώδικας στο dashboard.html:

```
<tbody>
    {% for device in devices %}
    <tr>
        <td>{{ device.namedevice }}</td>
        <td>{{ device.description }}</td>
        <td>
            <a href="{{ url_for('get_device', device_id=device.id) }}" class="btn btn-info btn-sm">Προβολή</a>
            <a href="{{ url_for('edit_device', device_id=device.id) }}" class="btn btn-warning btn-sm">Επεξεργασία</a>
            <form action="{{ url_for('delete_device', device_id=device.id) }}" method="POST" style="display:inline;">
```

```
            <button type="submit" class="btn btn-danger btn-sm">Διαγραφή</button>
        </form>
    </td>
  </tr>
  {% endfor %}
</tbody>
```

Επεξήγηση:

- Το κουμπί **Διαγραφή** υποβάλλει μια POST αίτηση στο delete_device endpoint.
- Η διαγραφή προστατεύεται μέσω του ελέγχου συνεδρίας (session['user_id']).

Παρακολούθηση Δεδομένων

Η πλατφόρμα επιτρέπει στους χρήστες να παρακολουθούν δεδομένα συσκευών σε πραγματικό χρόνο, χρησιμοποιώντας API.

Κώδικας για Προβολή Δεδομένων (get_device):

```
@app.route('/device/<int:device_id>', methods=['GET'])
def get_device(device_id):
  if 'user_id' not in session:
    return redirect(url_for('login'))

  connection = get_db_connection()
  try:
    with connection.cursor() as cursor:
      sql_device = "SELECT * FROM devices WHERE id = %s AND userid = %s"
      cursor.execute(sql_device, (device_id, session['user_id']))
      device = cursor.fetchone()
      if not device:
        return redirect(url_for('list_devices'))
```

```
        sql_values = """
            SELECT createdat, field1_value, field2_value, field3_value, field4_value
            FROM devices
            WHERE id = %s ORDER BY createdat DESC
        """
        cursor.execute(sql_values, (device_id,))
        valuefields = cursor.fetchall()

        fields_with_titles = {
            f'field{i}_value': device[f'field{i}_title']
            for i in range(1, 5)
            if device[f'field{i}_title']
        }
    return render_template('device_data.html', device=device, valuefields=valuefields, fields_with_titles=fields_with_titles)
  finally:
    connection.close()
```

Επεξήγηση:

- Ο χρήστης βλέπει τα δεδομένα της συσκευής σε μορφή πίνακα και γραφήματος.
- Η βάση δεδομένων επιστρέφει τις τελευταίες εγγραφές για κάθε πεδίο.

HTML για Προβολή Δεδομένων (device_data.html):

```
<div class="container my-5">
  <h2 class="text-center">Τιμές Συσκευής: {{ device.namedevice }}</h2>
  <div id="valuesChart" style="height: 400px;"></div>
  <table class="table table-striped mt-4">
    <thead>
      <tr>
```

```
            <th>Ημερομηνία</th>
            {% for field, title in fields_with_titles.items() %}
            <th>{{ title }}</th>
            {% endfor %}
        </tr>
    </thead>
    <tbody>
        {% for value in valuefields %}
        <tr>
            <td>{{ value.createdat }}</td>
            {% for field in fields_with_titles.keys() %}
            <td>{{ value[field] }}</td>
            {% endfor %}
        </tr>
        {% endfor %}
    </tbody>
</table>
<script>
    ZC.LICENSE = ["YOUR-ZINGCHART-LICENSE"];
    const valuesConfig = {
        type: 'line',
        title: { text: 'Δεδομένα Συσκευής', fontSize: 16 },
        series: [
            {% for field, title in fields_with_titles.items() %}
            { values: {{ valuefields|map(attribute=field)|list|tojson }}, text: '{{ title }}' },
            {% endfor %}
        ],
        scaleX: {
```

```
            labels: {{ valuefields|map(attribute='createdat')|list|tojson }},
            label: { text: 'Ημερομηνία' }
        },
        scaleY: { label: { text: 'Τιμές' } },
        legend: { draggable: true }
    };
    zingchart.render({
        id: 'valuesChart',
        data: valuesConfig,
        height: '100%',
        width: '100%'
    });
  </script>
</div>
```

Πίνακας: Εμφανίζει τις ημερομηνίες και τις τιμές για κάθε ενεργοποιημένο πεδίο.

Γράφημα: Απεικονίζει τα δεδομένα σε γραμμές, με ξεχωριστή ετικέτα για κάθε πεδίο.

4.2 Χρήση των api endpoints μέσω Python

Η πλατφόρμα που υλοποιήθηκε παρέχει μια σειρά από API endpoints για την επικοινωνία με τις συσκευές IoT. Αυτά τα endpoints επιτρέπουν τη διαχείριση, ανάγνωση και ενημέρωση δεδομένων των συσκευών, χρησιμοποιώντας τα κλειδιά API (apiwrite και apiread). Στο κεφάλαιο αυτό θα περιγραφεί αναλυτικά η χρήση αυτών των endpoints μέσω Python, παρέχοντας παραδείγματα κώδικα για την εύκολη ενσωμάτωσή τους σε εξωτερικά συστήματα ή εφαρμογές.

Η πλατφόρμα παρέχει τα παρακάτω API endpoints:

1. **API Write**: Ενημερώνει τιμές για συγκεκριμένα πεδία μιας συσκευής.
2. **API Read**: Επιστρέφει δεδομένα από συγκεκριμένα πεδία μιας συσκευής με δυνατότητα περιορισμού των εγγραφών.

Χρήση του API Write μέσω Python

Το API Write επιτρέπει την ενημέρωση των τιμών των πεδίων μιας συσκευής. Η βασική δομή ενός αιτήματος είναι:

URL:

```
http://localhost:5000/api/device/write?apiwrite=<api_write_key>&field=<field_name>&value=<value>
```

Παράδειγμα Python

```
import requests


# Ρυθμίσεις
api_write_key = "your_api_write_key"
field_name = "field1_value"
value = 100
url = f"http://localhost:5000/api/device/write?apiwrite={api_write_key}&field={field_name}&value={value}"


# Αίτημα
response = requests.get(url)


# Αποτελέσματα
if response.status_code == 200:
    print("Η τιμή ενημερώθηκε επιτυχώς:", response.json())
else:
    print("Σφάλμα:", response.status_code, response.json())
```

Χρήση του API Read μέσω Python

Το API Read επιτρέπει την ανάγνωση δεδομένων από συγκεκριμένα πεδία μιας συσκευής. Η βασική δομή ενός αιτήματος είναι:

URL:

```
http://localhost:5000/api/device/read?apiread=<api_read_key>&limit=<number_of_records>
```

Παράδειγμα Python

```
import requests


# Ρυθμίσεις
api_read_key = "your_api_read_key"
limit = 5
url = f"http://localhost:5000/api/device/read?apiread={api_read_key}&limit={limit}"


# Αίτημα
response = requests.get(url)


# Αποτελέσματα
if response.status_code == 200:
    data = response.json()
    print("Δεδομένα συσκευής:", data)
else:
    print("Σφάλμα:", response.status_code, response.json())
```

Συνδυασμός των API για Αυτόματη Ενημέρωση και Ανάγνωση

Σε πραγματικά συστήματα, συχνά απαιτείται συνδυασμός της ενημέρωσης και ανάγνωσης δεδομένων. Παρακάτω παρουσιάζεται ένα σενάριο όπου ενημερώνεται μια τιμή και στη συνέχεια επιστρέφονται οι πρόσφατες τιμές.

```
import requests
import time
```

```
# Ρυθμίσεις
api_write_key = "your_api_write_key"
api_read_key = "your_api_read_key"
field_name = "field1_value"
value = 150
limit = 5

# Ενημέρωση τιμής
write_url = f"http://localhost:5000/api/device/write?apiwrite={api_write_key}&field={field_name}&value={value}"
write_response = requests.get(write_url)

if write_response.status_code == 200:
    print("Η τιμή ενημερώθηκε επιτυχώς:", write_response.json())
else:
    print("Σφάλμα στην ενημέρωση:", write_response.status_code, write_response.json())

# Αναμονή για επικαιροποίηση
time.sleep(2)

# Ανάγνωση τιμών
read_url = f"http://localhost:5000/api/device/read?apiread={api_read_key}&limit={limit}"
read_response = requests.get(read_url)

if read_response.status_code == 200:
    print("Πρόσφατα δεδομένα:")
```

```
    for record in read_response.json()['values']:
        print(record)
else:
    print("Σφάλμα στην ανάγνωση:", read_response.status_code, read_response.json())
```

Χρήση με Προγραμματιστική Διαχείριση

Η επεξεργασία πολλών συσκευών ή η χρήση αυτοματοποιημένων διαδικασιών μπορεί να γίνει μέσω αντικειμενοστραφούς προσέγγισης. Ακολουθεί ένα παράδειγμα για τη διαχείριση πολλών συσκευών:

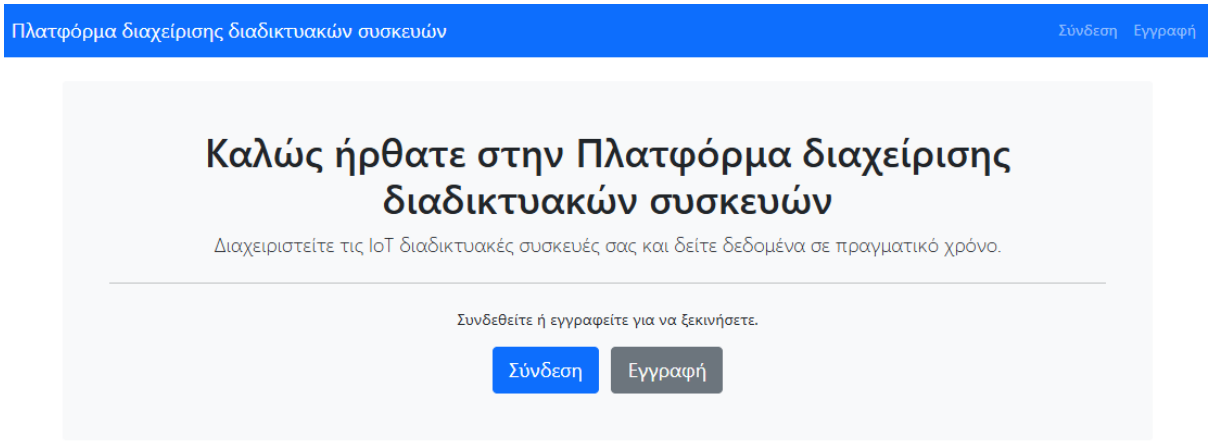
```
class IoTDevice:
    def __init__(self, api_write, api_read):
        self.api_write = api_write
        self.api_read = api_read

    def write_value(self, field, value):
        url = f"http://localhost:5000/api/device/write?apiwrite={self.api_write}&field={field}&value={value}"
        response = requests.get(url)
        return response.json() if response.status_code == 200 else response.json()

    def read_values(self, limit=10):
        url = f"http://localhost:5000/api/device/read?apiread={self.api_read}&limit={limit}"
        response = requests.get(url)
        return response.json() if response.status_code == 200 else response.json()

# Παράδειγμα χρήσης
device = IoTDevice(api_write="your_api_write_key", api_read="your_api_read_key")
print(device.write_value("field1_value", 200))
```

```
print(device.read_values(limit=5))
```

Η χρήση των API endpoints μέσω Python διευκολύνει την ενσωμάτωση των συσκευών IoT σε μεγαλύτερα συστήματα ή εφαρμογές. Η ευελιξία που παρέχεται από τις GET κλήσεις επιτρέπει την άμεση επικοινωνία με τη βάση δεδομένων, την ενημέρωση και την ανάγνωση δεδομένων σε πραγματικό χρόνο

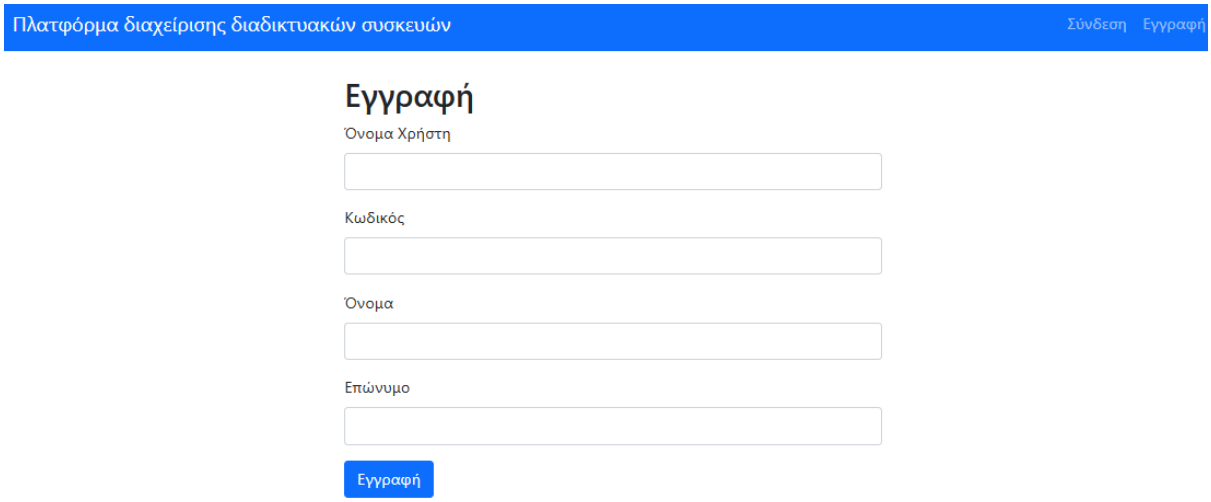
4.3 Η πλατφόρμα

Στο κεφάλαιο αυτό θα περιγραφεί η πλατφόρμα του συστήματος.

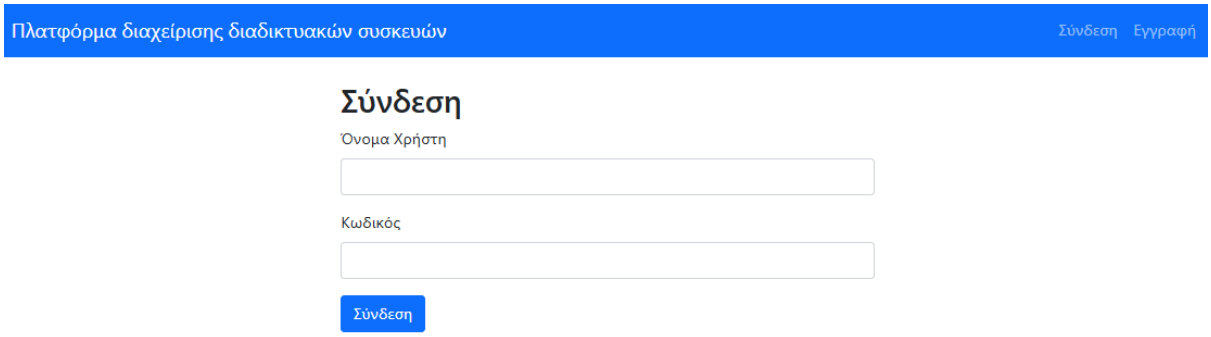


Εικόνα 4.2: Welcome σελίδα

Στην Εικόνα 4.2 παρουσιάζεται η αρχική σελίδα καλωσορίσματος της πλατφόρμας. Αυτή η σελίδα παρέχει μια συνοπτική περιγραφή του συστήματος και των δυνατοτήτων του, ενώ προσκαλεί τους χρήστες να συνδεθούν ή να εγγραφούν. Διαθέτει δύο κουμπιά που οδηγούν στις αντίστοιχες σελίδες σύνδεσης και εγγραφής, εξασφαλίζοντας εύκολη πλοήγηση για νέους και υπάρχοντες χρήστες. Η σελίδα χαρακτηρίζεται από καθαρό και λειτουργικό σχεδιασμό, ο οποίος εισάγει τον χρήστη στο περιβάλλον της πλατφόρμας.



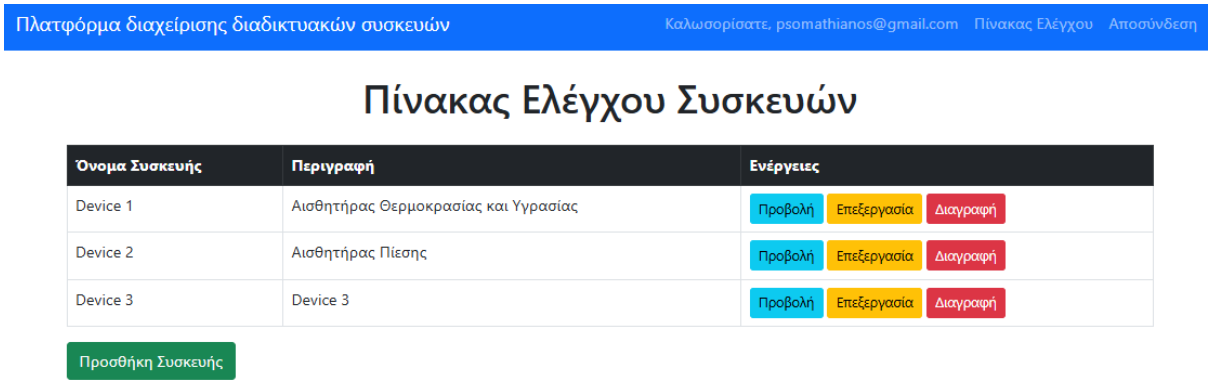
Εικόνα 4.3: Σελίδα εγγραφής ενός νέου χρήστη



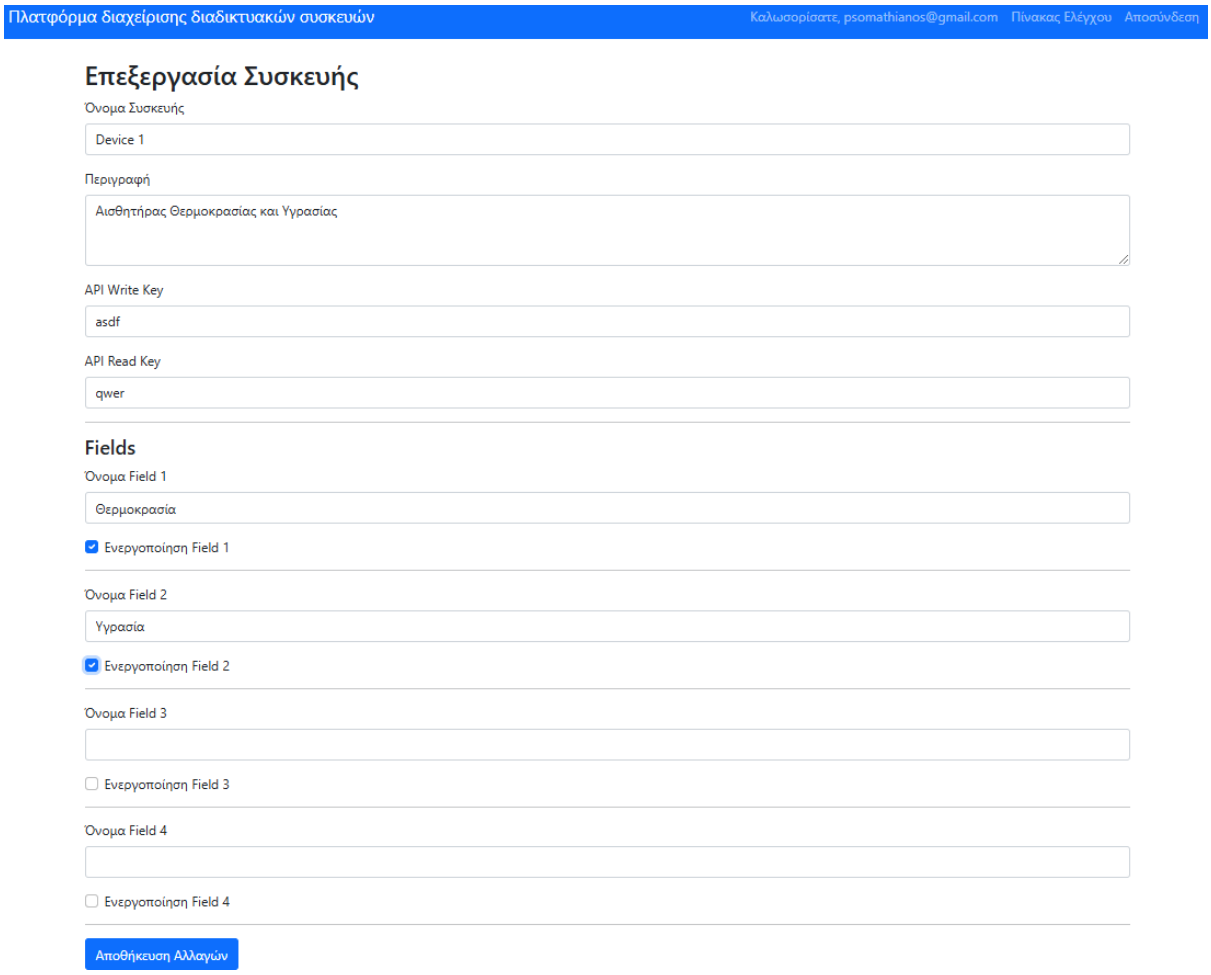
Εικόνα 4.4: Ιστοσελίδα για τη σύνδεση του χρήστη

Στην Εικόνα 4.3 απεικονίζεται η σελίδα εγγραφής νέου χρήστη. Η φόρμα περιλαμβάνει πεδία για την εισαγωγή ονόματος χρήστη, κωδικού πρόσβασης, και προσωπικών στοιχείων, όπως το όνομα και το επώνυμο. Η σελίδα επιτρέπει την εύκολη καταχώριση των απαραίτητων πληροφοριών και την ενσωμάτωσή τους στη βάση δεδομένων του συστήματος. Παρέχει επίσης μηνύματα καθοδήγησης για την ορθή συμπλήρωση της φόρμας και ενημερώνει τους χρήστες για τυχόν λάθη κατά τη διαδικασία εγγραφής.

Στην Εικόνα 4.4 παρουσιάζεται η σελίδα σύνδεσης του χρήστη. Η φόρμα περιλαμβάνει δύο πεδία: ένα για το όνομα χρήστη και ένα για τον κωδικό πρόσβασης. Σε περίπτωση που οι πληροφορίες είναι λανθασμένες, η πλατφόρμα εμφανίζει κατάλληλο μήνυμα σφάλματος. Η σελίδα αυτή είναι σχεδιασμένη να είναι απλή και γρήγορη στη χρήση, επιτρέποντας στους χρήστες να συνδεθούν άμεσα στον λογαριασμό τους και να αποκτήσουν πρόσβαση στις λειτουργίες της πλατφόρμας.



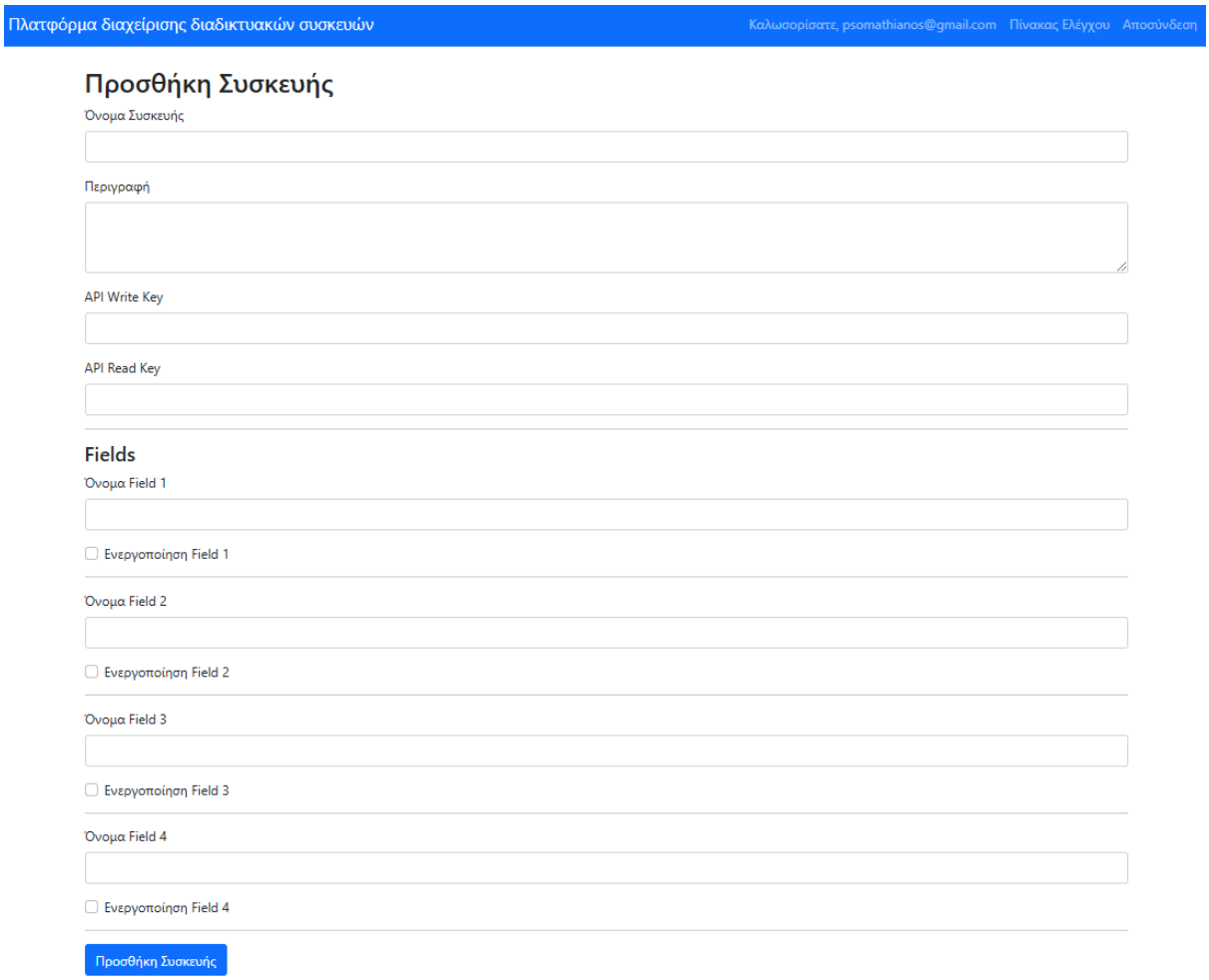
Εικόνα 4.5: Η κεντρική ιστοσελίδα με τις συσκευές με δυνατότητα προβολής, επεξεργασίας και διαγραφής κάθε συσκευής.



Εικόνα 4.6: Ιστοσελίδα επεξεργασίας ενός device και των πεδίων του και των api keys

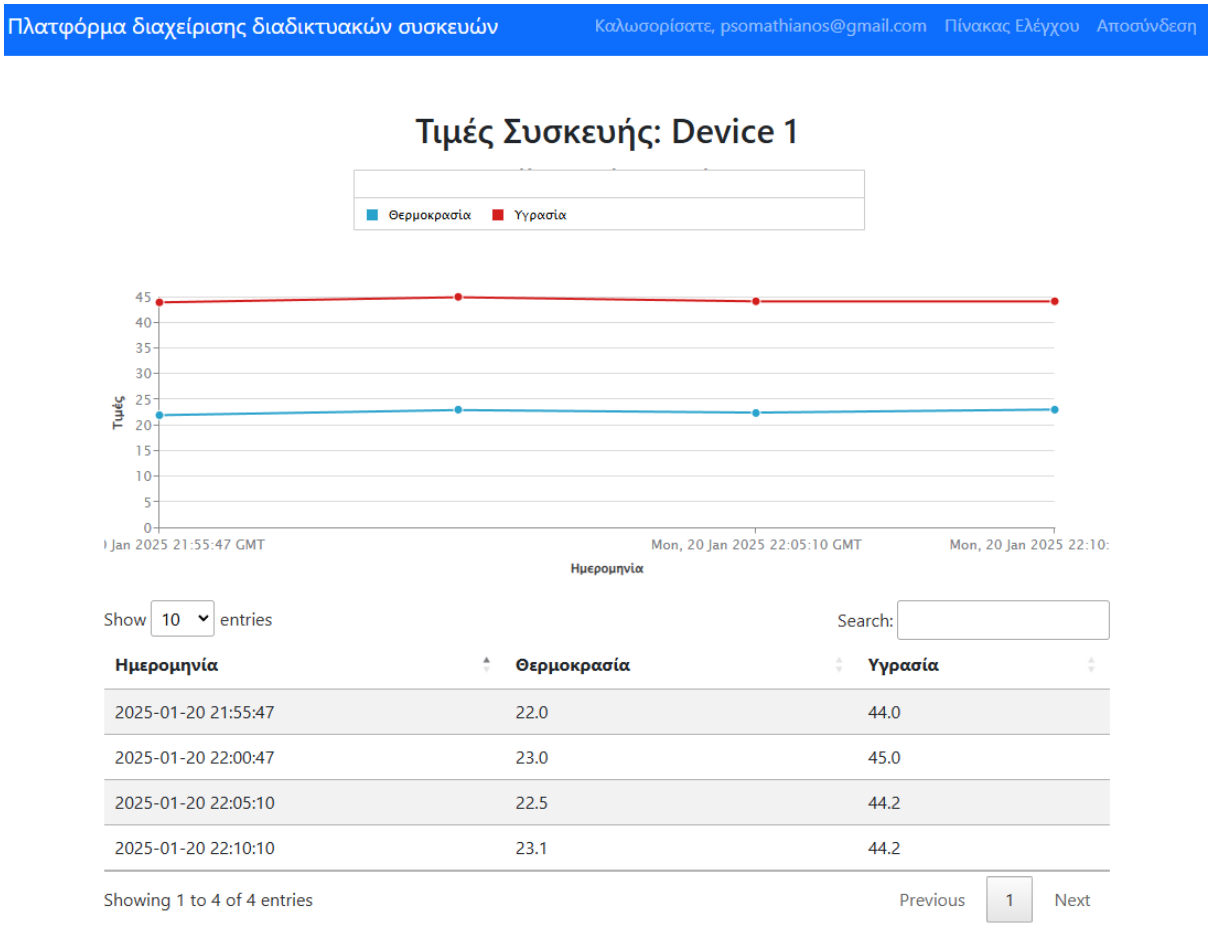
Στην Εικόνα 4.5 απεικονίζεται η κεντρική ιστοσελίδα της πλατφόρμας, όπου ο χρήστης μπορεί να δει όλες τις συσκευές που έχει δημιουργήσει. Κάθε συσκευή συνοδεύεται από επιλογές για την προβολή, την επεξεργασία ή τη διαγραφή της. Η σελίδα προσφέρει μια οργανωμένη προβολή των συσκευών, με σαφή παρουσίαση των ονομάτων και περιγραφών τους. Οι χρήστες μπορούν να εκτελέσουν γρήγορα ενέργειες στις συσκευές τους μέσω των αντίστοιχων κουμπιών, εξασφαλίζοντας αποτελεσματική διαχείριση.

Στην Εικόνα 4.6 παρουσιάζεται η σελίδα επεξεργασίας μιας συσκευής. Ο χρήστης έχει τη δυνατότητα να αλλάξει το όνομα της συσκευής, την περιγραφή της, καθώς και τα κλειδιά API (API write και API read). Επιπλέον, μπορεί να αλλάξει τα ονόματα των πεδίων της συσκευής και να ενεργοποιήσει ή να απενεργοποιήσει τη λειτουργικότητά τους. Αυτή η σελίδα είναι καίρια για τη διαμόρφωση των παραμέτρων λειτουργίας μιας συσκευής, προσφέροντας μεγάλη ευελιξία και προσαρμοστικότητα.



Εικόνα 4.7: Ιστοσελίδα δημιουργίας ενός device και των πεδίων του και των api keys.

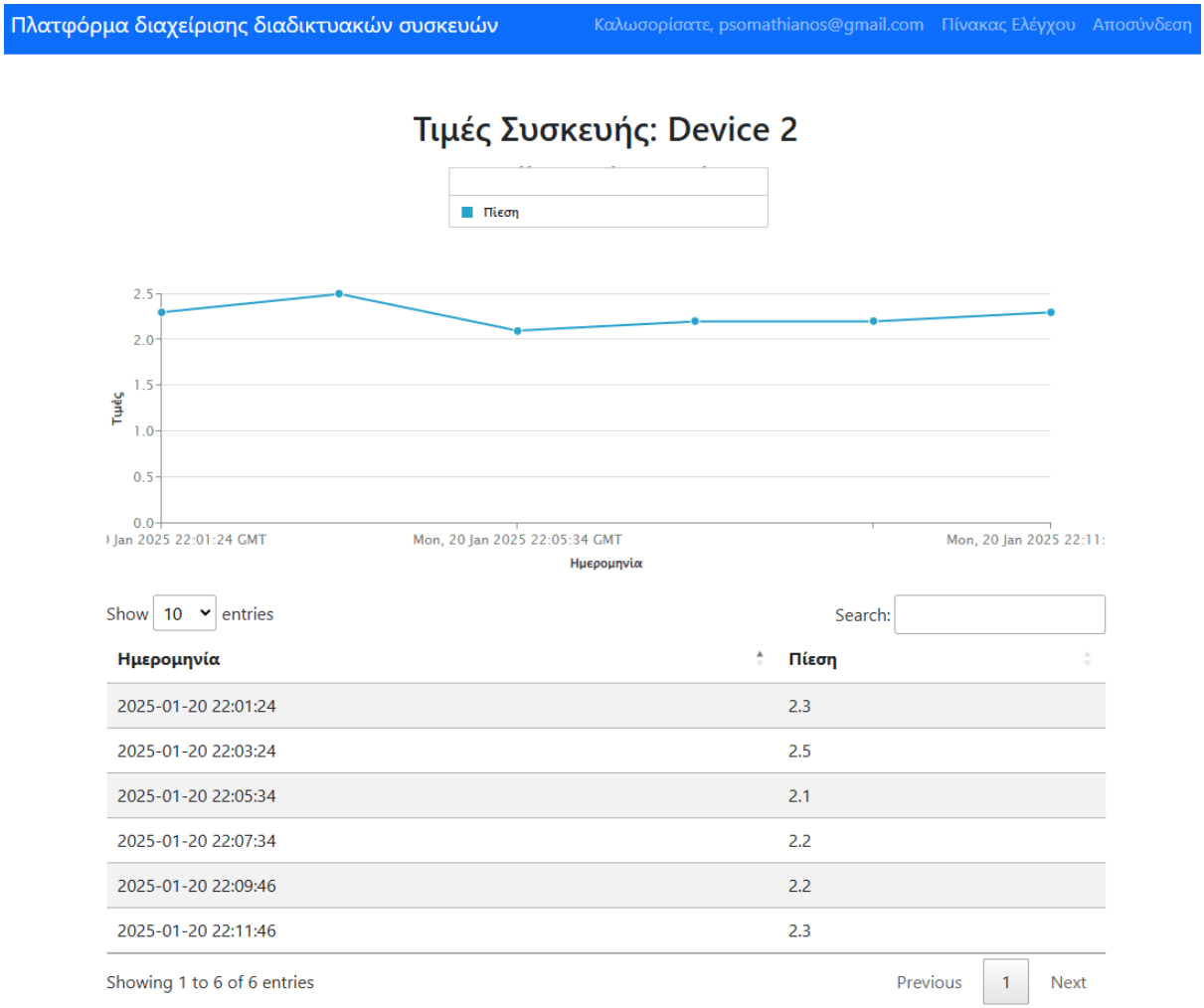
Στην Εικόνα 4.7 απεικονίζεται η σελίδα δημιουργίας μιας νέας συσκευής. Ο χρήστης μπορεί να εισάγει το όνομα της συσκευής, την περιγραφή της και να δημιουργήσει αυτόματα νέα κλειδιά API. Επιπλέον, η σελίδα δίνει τη δυνατότητα ορισμού των πεδίων της συσκευής και της ενεργοποίησής τους.



Εικόνα 4.8: Ιστοσελίδα προβολής των δεδομένων μια συσκευής – γράφημα και πίνακας τιμών – Device 1

Στην Εικόνα 4.8 φαίνεται η σελίδα προβολής δεδομένων μιας συσκευής. Η σελίδα περιλαμβάνει ένα δυναμικό γράφημα που απεικονίζει τις τιμές των πεδίων της συσκευής σε σχέση με τον χρόνο, καθώς και έναν πίνακα με τις τιμές σε μορφή λίστας. Τα δεδομένα εμφανίζονται σε πραγματικό χρόνο, παρέχοντας στον χρήστη μια πλήρη εικόνα της απόδοσης της συσκευής.

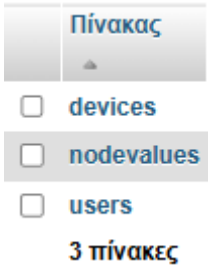
Στην Εικόνα 4.9 παρουσιάζεται μια παρόμοια σελίδα με την Εικόνα 4.8, αλλά αφορά μια διαφορετική συσκευή. Ο πίνακας και το γράφημα ενημερώνονται δυναμικά με τα δεδομένα της συγκεκριμένης συσκευής, παρέχοντας χρήσιμες πληροφορίες για την παρακολούθηση και την ανάλυση των λειτουργιών της. Η σελίδα υποστηρίζει την εύκολη πλοήγηση και την κατανόηση των δεδομένων από τον χρήστη.



Εικόνα 4.9: Ιστοσελίδα προβολής των δεδομένων μια συσκευής – γράφημα και πίνακας τιμών – Device 2

4.4 Η Βάση δεδομένων

Η βάση δεδομένων αποτελεί τη θεμελιώδη δομή για την αποθήκευση, διαχείριση και ανάκτηση των δεδομένων του συστήματος. Στην παρούσα εργασία, η βάση δεδομένων σχεδιάστηκε με γνώμονα την αποτελεσματική οργάνωση των δεδομένων που αφορούν τις συσκευές IoT, τους χρήστες και τις μετρήσεις τους. Η επιλογή της MySQL ως διαχειριστή βάσης δεδομένων έγινε λόγω της ευελιξίας, της ταχύτητας και της ευρείας αποδοχής της σε έργα IoT.



Εικόνα 4.10: Οι πίνακες της βάσης

Η βάση δεδομένων αποτελείται από τρεις κύριους πίνακες: users, devices και nodevalues. Κάθε πίνακας έχει σχεδιαστεί για να εξυπηρετεί συγκεκριμένες λειτουργίες του συστήματος, ενώ οι σχέσεις μεταξύ των πινάκων διασφαλίζουν τη συνεκτικότητα και την ακεραιότητα των δεδομένων.

Περιγραφή Πινάκων

Πίνακας users

Ο πίνακας users είναι υπεύθυνος για την αποθήκευση των δεδομένων των χρηστών που χρησιμοποιούν την πλατφόρμα. Περιλαμβάνει πληροφορίες που αφορούν την ταυτότητα και την πρόσβαση των χρηστών. Η δομή του πίνακα περιλαμβάνει τα εξής πεδία:

- **id**: Ένας μοναδικός αριθμός που αντιπροσωπεύει κάθε χρήστη.
- **username**: Το όνομα χρήστη που χρησιμοποιείται για τη σύνδεση.
- **password**: Ο κωδικός πρόσβασης του χρήστη, αποθηκευμένος σε ασφαλή μορφή.
- **firstname**: Το όνομα του χρήστη.
- **lastname**: Το επώνυμο του χρήστη.
- **createdat**: Η ημερομηνία και ώρα εγγραφής του χρήστη.
- **updatedat**: Η ημερομηνία και ώρα της τελευταίας ενημέρωσης του χρήστη.

Αυτός ο πίνακας εξασφαλίζει την ασφαλή σύνδεση κάθε χρήστη.

Πίνακας devices

Ο πίνακας devices περιέχει πληροφορίες για όλες τις συσκευές IoT που διαχειρίζεται το σύστημα. Κάθε εγγραφή αναφέρεται σε μία συσκευή που έχει δημιουργηθεί από έναν χρήστη. Τα πεδία περιλαμβάνουν:

- **id**: Ένας μοναδικός αριθμός για κάθε συσκευή.

- **active**: Δείκτης που υποδεικνύει αν η συσκευή είναι ενεργή (1) ή απενεργοποιημένη (0).
- **public**: Δείκτης που δείχνει αν η συσκευή είναι δημόσια ή ιδιωτική.
- **userid**: Ο αριθμός του χρήστη που δημιούργησε τη συσκευή (σχέση με τον πίνακα users).
- **namedevice**: Το όνομα της συσκευής.
- **description**: Μια περιγραφή της συσκευής.
- **apiwrite**: Το μοναδικό κλειδί API για την ενημέρωση δεδομένων της συσκευής.
- **apiread**: Το μοναδικό κλειδί API για την ανάγνωση δεδομένων της συσκευής.
- **fieldX_value**: Τιμές για τέσσερα διαφορετικά πεδία που μπορούν να οριστούν από τον χρήστη.
- **fieldX_title**: Τίτλοι για τα αντίστοιχα πεδία.
- **createdat**: Η ημερομηνία και ώρα δημιουργίας της συσκευής.
- **updatedat**: Η ημερομηνία και ώρα της τελευταίας ενημέρωσης της συσκευής.

Αυτός ο πίνακας παρέχει την υποδομή για τη διαχείριση συσκευών και των ιδιοτήτων τους.

Πίνακας nodevalues

Ο πίνακας nodevalues αποθηκεύει τις τιμές που προέρχονται από τις συσκευές IoT. Ο πίνακας αυτός επιτρέπει την καταγραφή μετρήσεων και δεδομένων σε πραγματικό χρόνο. Τα πεδία περιλαμβάνουν:

- **id**: Ένας μοναδικός αριθμός για κάθε εγγραφή.
- **active**: Δείκτης που δείχνει αν η εγγραφή είναι ενεργή.
- **device_id**: Ο αριθμός της συσκευής από την οποία προέρχονται τα δεδομένα (σχέση με τον πίνακα devices).
- **fieldX_value**: Τιμές για τέσσερα πεδία δεδομένων.
- **created_at**: Η ημερομηνία και ώρα καταγραφής των δεδομένων.

Ο πίνακας nodevalues διασφαλίζει την καταγραφή και οργάνωση όλων των δεδομένων που προέρχονται από τις συσκευές IoT.

Σχέσεις Πινάκων

Οι τρεις πίνακες συνδέονται μέσω πρωτευόντων και ξένων κλειδιών:

- Ο πίνακας devices σχετίζεται με τον πίνακα users μέσω του πεδίου userid.
- Ο πίνακας nodevalues σχετίζεται με τον πίνακα devices μέσω του πεδίου device_id.

Αυτές οι σχέσεις εξασφαλίζουν τη διατήρηση της συνοχής και της λογικής συνάφειας των δεδομένων στη βάση.

Η βάση δεδομένων του συστήματος είναι σχεδιασμένη για να υποστηρίζει αποτελεσματικά τις ανάγκες μιας πλατφόρμας IoT. Η δομή της είναι ευέλικτη, εξασφαλίζοντας εύκολη επέκταση για μελλοντικές λειτουργίες, ενώ οι σχέσεις μεταξύ των πινάκων διασφαλίζουν τη συνοχή των δεδομένων. Επιπλέον, η χρήση της MySQL παρέχει τη δυνατότητα γρήγορης ανάκτησης και ανάλυσης δεδομένων, υποστηρίζοντας λειτουργίες σε πραγματικό χρόνο και αυξάνοντας την απόδοση του συστήματος.

4.5 Ασφάλεια στην πλατφόρμα και στη διασύνδεση των συσκευών

Η ασφάλεια αποτελεί έναν από τους πιο κρίσιμους παράγοντες κατά τη σχεδίαση και ανάπτυξη ενός συστήματος IoT. Στην παρούσα πλατφόρμα, η ασφάλεια εξασφαλίζεται τόσο σε επίπεδο χρήστη όσο και σε επίπεδο συσκευών, αποτρέποντας μη εξουσιοδοτημένη πρόσβαση, παραβιάσεις δεδομένων και κακόβουλες επιθέσεις.

4.5.1 Ασφάλεια Χρηστών

Η προστασία των δεδομένων των χρηστών είναι ζωτικής σημασίας για τη διατήρηση της αξιοπιστίας της πλατφόρμας. Ορισμένα από τα μέτρα που υλοποιούνται περιλαμβάνουν:

1. **Ασφαλής Διαχείριση Κωδικών**

   Οι κωδικοί πρόσβασης αποθηκεύονται κρυπτογραφημένοι στη βάση δεδομένων. Έτσι, ακόμη και αν κάποιος αποκτήσει πρόσβαση στη βάση, οι κωδικοί δεν είναι αναγνώσιμοι. Χρησιμοποιείται ο αλγόριθμος bcrypt για τη δημιουργία ασφαλών hash.

2. **Σύστημα Επαλήθευσης Χρήστη**

   Η πλατφόρμα περιλαμβάνει ένα σύστημα επαλήθευσης που διασφαλίζει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να συνδεθούν και να διαχειριστούν τις συσκευές τους. Κατά τη διαδικασία σύνδεσης, η πλατφόρμα ελέγχει τα διαπιστευτήρια μέσω ασφαλών μεθόδων ελέγχου ταυτότητας.

3. **Χρήση Session Tokens**

   Οι συνδέσεις των χρηστών διαχειρίζονται μέσω ασφαλών session tokens, τα οποία αποθηκεύονται προσωρινά στον φυλλομετρητή του χρήστη. Τα tokens αυτά λήγουν μετά από συγκεκριμένο χρονικό διάστημα ή αν ο χρήστης αποσυνδεθεί, αποτρέποντας μη εξουσιοδοτημένη πρόσβαση.

4.5.2 Ασφάλεια Συσκευών

Η προστασία των δεδομένων και της λειτουργίας των συσκευών IoT είναι κρίσιμης σημασίας. Τα μέτρα που εφαρμόζονται περιλαμβάνουν:

1. **Κλειδιά API (API Keys)**

   Κάθε συσκευή διαθέτει μοναδικά κλειδιά API για την ανάγνωση (API Read) και την ενημέρωση (API Write) των δεδομένων της. Αυτά τα κλειδιά διασφαλίζουν ότι μόνο οι εξουσιοδοτημένοι χρήστες και εφαρμογές μπορούν να αλληλεπιδράσουν με τη συσκευή. Τα κλειδιά API είναι μεγάλης πολυπλοκότητας για την αποφυγή προβλέψιμων μοτίβων.

2. **Περιορισμοί Πρόσβασης**

   Οι συσκευές μπορούν να ρυθμιστούν ως δημόσιες ή ιδιωτικές. Στην περίπτωση ιδιωτικών συσκευών, μόνο ο χρήστης που τις δημιούργησε μπορεί να έχει πρόσβαση στα δεδομένα ή να τις διαχειριστεί.

3. **Κρυπτογράφηση Δεδομένων**

   Όλα τα δεδομένα που μεταφέρονται μεταξύ της πλατφόρμας και των συσκευών είναι κρυπτογραφημένα μέσω του πρωτοκόλλου HTTPS. Αυτό διασφαλίζει ότι τα δεδομένα είναι προστατευμένα κατά τη μεταφορά, αποτρέποντας την υποκλοπή ή την αλλοίωση τους.

4.5.3 Ασφάλεια Δεδομένων

Η ακεραιότητα και η προστασία των δεδομένων που αποθηκεύονται στη βάση δεδομένων αποτελούν κύρια προτεραιότητα. Τα μέτρα που εφαρμόζονται περιλαμβάνουν:

1. **Διαχωρισμός Δεδομένων Χρηστών**

   Τα δεδομένα κάθε χρήστη διατηρούνται πλήρως διαχωρισμένα. Έτσι, κάθε χρήστης έχει πρόσβαση μόνο στα δεδομένα των δικών του συσκευών.

2. **Δικαιώματα Βάσης Δεδομένων**

   Ο χρήστης της MySQL που συνδέεται με την πλατφόρμα έχει περιορισμένα δικαιώματα, ώστε να αποτρέπεται η τυχαία ή κακόβουλη τροποποίηση της δομής της βάσης δεδομένων.

3. **Δημιουργία Αντιγράφων Ασφαλείας**

   Αντίγραφα ασφαλείας της βάσης δεδομένων λαμβάνονται τακτικά και αποθηκεύονται σε ασφαλείς τοποθεσίες. Αυτό διασφαλίζει ότι τα δεδομένα μπορούν να ανακτηθούν σε περίπτωση αποτυχίας του συστήματος.

4.5.4 Προστασία από Κακόβουλες Ενέργειες

Για την αποτροπή κακόβουλων ενεργειών, η πλατφόρμα περιλαμβάνει τα εξής μέτρα:

1. **Προστασία από SQL Injection**

   Όλα τα αιτήματα προς τη βάση δεδομένων εκτελούνται με προετοιμασμένες δηλώσεις (prepared statements), εξαλείφοντας την πιθανότητα εκτέλεσης κακόβουλου SQL κώδικα.

2. **Προστασία από Brute Force Attacks**

   Η πλατφόρμα εφαρμόζει μηχανισμούς περιορισμού αιτημάτων, όπως ο καθορισμός μέγιστου αριθμού αποτυχημένων προσπαθειών σύνδεσης, για την αποτροπή επιθέσεων brute force.

3. **Καταγραφή Ενεργειών**

   Όλες οι ενέργειες των χρηστών και των συσκευών καταγράφονται για λόγους παρακολούθησης. Τα αρχεία καταγραφής αναλύονται περιοδικά για τον εντοπισμό ύποπτων δραστηριοτήτων.

Η πλατφόρμα έχει σχεδιαστεί με γνώμονα την ασφάλεια, υιοθετώντας τεχνικές και στρατηγικές που διασφαλίζουν την προστασία χρηστών, συσκευών και δεδομένων. Από την κρυπτογράφηση και τη διαχείριση κλειδιών API μέχρι την προστασία από κακόβουλες επιθέσεις, κάθε πτυχή της ασφάλειας έχει ληφθεί υπόψη. Αυτό όχι μόνο ενισχύει την εμπιστοσύνη των χρηστών, αλλά διασφαλίζει και τη βιωσιμότητα του συστήματος στο απαιτητικό περιβάλλον του IoT.

Κεφάλαιο 5ο: Συμπεράσματα και βελτιώσεις

Η παρούσα εργασία επικεντρώθηκε στον σχεδιασμό, την ανάπτυξη και την αξιολόγηση μιας πλατφόρμας διαχείρισης διαδικτυακών συσκευών στο περιβάλλον του Internet of Things (IoT). Μέσω της υλοποίησης, αναδείχθηκαν τα βασικά χαρακτηριστικά που καθιστούν μια τέτοια πλατφόρμα χρήσιμη, ασφαλή και επεκτάσιμη, ενώ παράλληλα εντοπίστηκαν σημεία που μπορούν να βελτιωθούν.

Σημειώνεται ότι κατά τη διάρκεια της εργασίας, το open ai χρησιμοποιήθηκε σε διάφορα στάδια για τη σύνταξη, τη διόρθωση και τη βελτίωση της μορφοποίησης του κειμένου.

Η ανάπτυξη της πλατφόρμας ανέδειξε τη σημασία της αποτελεσματικής διαχείρισης συσκευών IoT, καθώς και την ανάγκη για ασφάλεια και ευχρηστία.

Η πλατφόρμα παρέχει μια φιλική προς τον χρήστη διεπαφή, επιτρέποντας τη διαχείριση συσκευών, την παρακολούθηση δεδομένων και τη ρύθμιση παραμέτρων με ελάχιστες απαιτήσεις εκπαίδευσης. Οι μηχανισμοί ασφάλειας, όπως τα κλειδιά API, η κρυπτογράφηση δεδομένων και ο περιορισμός πρόσβασης, διασφαλίζουν την προστασία των πληροφοριών από κακόβουλες ενέργειες. Η αρχιτεκτονική του συστήματος επιτρέπει την εύκολη ενσωμάτωση νέων συσκευών και λειτουργιών, καθιστώντας την πλατφόρμα κατάλληλη για μελλοντικές επεκτάσεις. Η δυνατότητα προβολής δεδομένων μέσω γραφημάτων και πινάκων σε πραγματικό χρόνο βελτιώνει τη λήψη αποφάσεων και την ανάλυση των συσκευών. Η σταθερότητα της πλατφόρμας κατά τη χρήση πολλαπλών συσκευών και η ταχύτητα στην ανταπόκριση αναδεικνύουν την αξιοπιστία της.

Παρόλο που η πλατφόρμα παρέχει σημαντική λειτουργικότητα, υπάρχουν ορισμένα σημεία που μπορούν να βελτιωθούν για να αυξηθεί η απόδοσή της και να καλυφθούν πιο εξειδικευμένες ανάγκες. Παρόλο που η πλατφόρμα υποστηρίζει τα βασικά πρωτόκολλα επικοινωνίας, η ενσωμάτωση επιπλέον πρωτοκόλλων, όπως το Zigbee ή το LoRaWAN, θα αυξήσει τη συμβατότητά της με περισσότερες συσκευές. Η προσθήκη λειτουργιών ειδοποίησης, όπως αποστολή email ή SMS σε περίπτωση ανίχνευσης κρίσιμων τιμών, θα βελτιώσει την αμεσότητα στη λήψη αποφάσεων. Η επέκταση των δυνατοτήτων γραφημάτων, όπως η προσθήκη δυναμικών φίλτρων και επιλογών ανάλυσης, θα ενισχύσει την εμπειρία του χρήστη.

Η υλοποίηση επιπλέον επιπέδων ασφάλειας, όπως η πολυπαραγοντική ταυτοποίηση και η δυναμική ανανέωση κλειδιών API, θα ενισχύσει περαιτέρω την προστασία του συστήματος και η ενσωμάτωση τεχνολογιών όπως τα NoSQL συστήματα για την αποθήκευση μεγάλου όγκου δεδομένων θα αυξήσει την απόδοση της πλατφόρμας, ειδικά σε περιπτώσεις όπου διαχειρίζεται πολλές

συσκευές ταυτόχρονα. Τέλος, η ανάπτυξη API που θα επιτρέπουν την εύκολη ενσωμάτωση της πλατφόρμας με άλλες εφαρμογές και εργαλεία θα ενισχύσει τη χρησιμότητά της.

Η εργασία αυτή παρέχει τη δυνατότητα δημιουργίας μιας λειτουργικής, επεκτάσιμης και ασφαλούς πλατφόρμας IoT. Με την υλοποίηση των προτάσεων βελτίωσης, η πλατφόρμα μπορεί να εξελιχθεί περαιτέρω, παρέχοντας ακόμη μεγαλύτερη αξία στους χρήστες της και διαδραματίζοντας σημαντικό ρόλο στο σύγχρονο περιβάλλον του IoT.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Ashton, K. (2009). That ‘Internet of Things’ Thing. RFID Journal.

[2] https://aws.amazon.com/iot-core/

[3] https://cloud.google.com/architecture/connected-devices/iot-platform-product-architecture

[4] https://azure.microsoft.com/en-us/products/iot-hub

[5] https://thingsboard.io/

[6] https://www.python.org/doc/

[7] https://realpython.com/

[8] https://flask.palletsprojects.com/

[9] https://realpython.com/flask-by-example/

[10] https://flask.palletsprojects.com/

[11] https://www.w3schools.com/mysql/

[12] https://www.w3schools.com/bootstrap5/bootstrap_get_started.php

[13] https://www.zingchart.com/