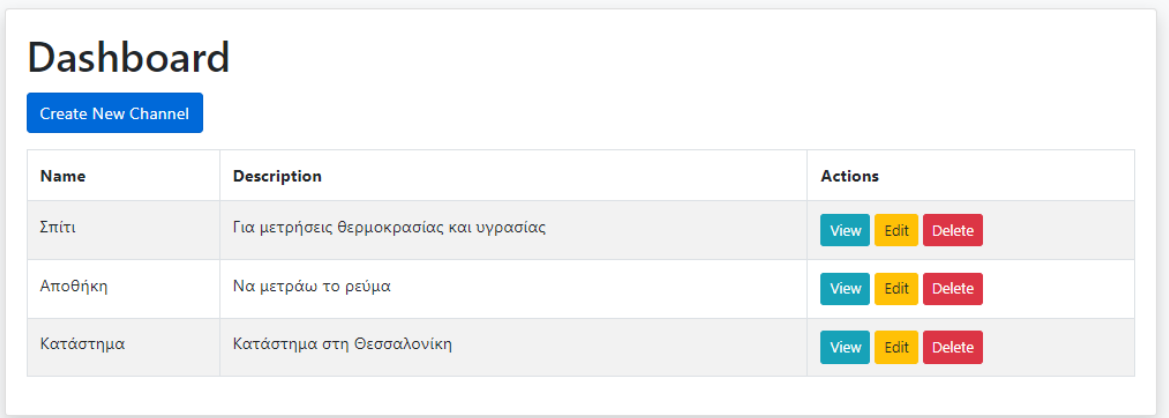
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Πλατφόρμα υποστήριξης επικοινωνίας συσκευών μέσω διαδικτύου»



Φοιτητής

ΚΩΝΣΤΑΝΤΙΝΙΔΗΣ ΑΠΟΣΤΟΛΟΣ

518200

Επιβλέπων

Δρ. Κυριάκος Τσιακμάκης

ΙΟΥΝΙΟΣ 2024

Τίτλος: Πλατφόρμα υποστήριξης επικοινωνίας συσκευών μέσω διαδικτύου

Κωδικός: 24172

Φοιτητής: Κωνσταντινίδης Απόστολος

Εισηγητής: Δρ Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 27-03-2024

Ημερομηνία περάτωσης Π.Ε. 23-05-2024

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.

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

Περίληψη

Η εργασία αυτή αφορά τη μελέτη συστημάτων και την ανάπτυξη εφαρμογής IoT χρησιμοποιώντας τεχνολογίες όπως το Laravel και η MySQL. Σκοπός της είναι να δημιουργήσει μια ολοκληρωμένη λύση για τη συλλογή, αποθήκευση, ανάλυση και οπτικοποίηση δεδομένων από διάφορες συσκευές IoT. Αφορά την ανάπτυξη της εφαρμογής χρησιμοποιώντας το Laravel για τη διαχείριση της βάσης δεδομένων MySQL, την ανάπτυξη του API και τη διαχείριση χρηστών και δεδομένων. Το Laravel επιλέχθηκε λόγω της ευελιξίας, της απλότητας και των ισχυρών εργαλείων που προσφέρει, ενώ η MySQL επιλέχθηκε για την αξιοπιστία και την ευρεία υποστήριξή της στην ανάπτυξη web εφαρμογών. Περιγράφεται η ανάπτυξη του API με την Python και η επικοινωνία του με το Laravel μέσω HTTP αιτημάτων, επιτρέποντας την εύκολη ανταλλαγή δεδομένων μεταξύ των δύο συστημάτων. Αναλύονται τα πλεονεκτήματα και τα μειονεκτήματα της χρήσης αυτών των τεχνολογιών, καθώς και οι λόγοι για τους οποίους προτιμώνται σε σύγχρονες εφαρμογές IoT.

« Platform for Supporting Device Communication over the Internet »

Abstract

This work concerns the study of systems and the development of an IoT application using technologies such as Laravel and MySQL. Its purpose is to create a comprehensive solution for collecting, storing, analyzing, and visualizing data from various IoT devices. It involves the development of the application using Laravel for managing the MySQL database, developing the API, and handling users and data. Laravel was chosen for its flexibility, simplicity, and powerful tools, while MySQL was selected for its reliability and widespread support in web application development. The development of the API using Python is described, along with its communication with Laravel via HTTP requests, allowing for easy data exchange between the two systems. The advantages and disadvantages of using these technologies are analyzed, as well as the reasons why they are preferred in modern IoT applications.

Ευχαριστίες

Θα ήθελα να εκφράσω την ευγνωμοσύνη μου στους γονείς μου για την αμέριστη στήριξή τους καθ' όλη τη διάρκεια αυτού του έργου. Επίσης, θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντά μου, κ. Τσιακμάκη, για την καθοδήγηση του, τις επιστημονικές συμβουλές του και την πολύτιμη συμβολή του στην ανάπτυξη του 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 11
Κεφάλαιο 2ο: Βιβλιογραφική Ανασκόπηση Iot συστημάτων 12
2.1 Εισαγωγή 12
2.2 ThingSpeak 14
2.3 ThingsBoard 15
2.4 AWS IoT Core 16
2.5 Συγκρίσεις 16
Κεφάλαιο 3ο: Τα τεχνολογικά εργαλεία 18
3.1 PHP 18
3.2 Laravel 19
3.3 Python 21
3.3.1 API Python 23
3.4 MySQL 26
Κεφάλαιο 4ο: Το σύστημα IoT 28
4.1 Εισαγωγή – Η διαδικασία 28
4.2 Η εφαρμογή - ιστοσελίδα 30
4.2.1 API 38
4.3 Η Βάση 42
4.4 Ασφάλεια στο σύστημα και στα δεδομένα 49
Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης 52
ΒΙΒΛΙΟΓΡΑΦΙΑ 54
ΠΑΡΑΡΤΗΜΑ Α 56

Κατάλογος Σχημάτων

Εικόνα 4.1: Η σελίδα που βλέπει ο μη συνδεδεμένος χρήστης. Έχει διαθέσιμα το Login και Register. 30
Εικόνα 4.2: Η σελίδα Register-Εγγραφή για νέο χρήστη 31
Εικόνα 4.3: Η σελίδα Login-Σύνδεση για έναν εγγεγραμμένο χρήστη 32
Εικόνα 4.4: Dashboard. Η αρχική σελίδα που βλέπει ο συνδεδεμένος χρήστης. Βλέπει όλα τα κανάλια του με επιλογές προβολής, επεξεργασίας ή διαγραφής..... 32
Εικόνα 4.5: Σελίδα για δημιουργία καναλιού – άδεια πεδία 33
Εικόνα 4.6: Σελίδα για δημιουργία καναλιού – με συμπληρωμένα πεδία 34
Εικόνα 4.7: Σελίδα για επεξεργασία των χαρακτηριστικών ενός καναλιού 35
Εικόνα 4.8: Επιλογή προβολής καναλιών 35
Εικόνα 4.9: Το chart για το κανάλι Σπίτι – δύο πεδία temp,hum 36
Εικόνα 4.10: Το chart για το κανάλι Αποθήκη – τρία πεδία Voltage,Current,Watt 36
Εικόνα 4.11: Το chart για το κανάλι Κατάστημα – τρία πεδία temp,hum,pressure..... 37
Εικόνα 4.12: Η βάση με τους πίνακες 46
Εικόνα 4.13: Η δομή του πίνακα 46
Εικόνα 4.14: Το περιεχόμενο του πίνακα 47
Εικόνα 4.15: Η δομή του πίνακα channel 47
Εικόνα 4.16: Το περιεχόμενο του πίνακα channel..... 47
Εικόνα 4.17: Η δομή του πίνακα fieldvalue 48
Εικόνα 4.18: Το περιεχόμενο του πίνακα fieldvalue..... 48

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Το σύστημα IoT που αναπτύξαμε στο πλαίσιο αυτού του έργου είναι μια ευέλικτη λύση για τη διαχείριση δεδομένων από διάφορες συσκευές IoT. Με την πρόοδο της τεχνολογίας και την αύξηση του αριθμού των συσκευών που συνδέονται στο διαδίκτυο, η ανάγκη για συστήματα που μπορούν να συλλέγουν, να αποθηκεύουν, να αναλύουν και να προβάλλουν δεδομένα σε πραγματικό χρόνο έχει γίνει επιτακτική. Το σύστημα IoT που περιγράφεται προσφέρει μια ολοκληρωμένη λύση που καλύπτει αυτές τις ανάγκες, παρέχοντας στους χρήστες ένα εύχρηστο και ασφαλές περιβάλλον για τη διαχείριση των δεδομένων τους.

Οι κύριοι στόχοι του συστήματος IoT είναι οι εξής:

Να επιτρέπει τη συλλογή δεδομένων από διάφορους αισθητήρες και συσκευές IoT μέσω ασφαλών και αξιόπιστων πρωτοκόλλων επικοινωνίας.

Να αποθηκεύει τα δεδομένα με ασφαλή τρόπο, διασφαλίζοντας την ακεραιότητα και τη διαθεσιμότητα των πληροφοριών.

Να παρέχει εργαλεία για την ανάλυση των συλλεγόμενων δεδομένων, επιτρέποντας στους χρήστες να εξάγουν πολύτιμα συμπεράσματα και να λαμβάνουν ενημερωμένες αποφάσεις.

Να προσφέρει δυναμικές και διαδραστικές απεικονίσεις των δεδομένων μέσω γραφημάτων και αναφορών, διευκολύνοντας την κατανόηση των τάσεων και των προτύπων.

Η αρχιτεκτονική του συστήματος IoT βασίζεται σε μια πολυεπίπεδη προσέγγιση.

Η διεπαφή χρήστη είναι σχεδιασμένη για να είναι φιλική και εύχρηστη, επιτρέποντας στους χρήστες να πλοηγούνται εύκολα και να εκτελούν τις απαραίτητες ενέργειες. Χρησιμοποιώντας το Laravel framework, η διεπαφή παρέχει λειτουργίες για τη δημιουργία και διαχείριση καναλιών, την προβολή δεδομένων και την επεξεργασία πληροφοριών.

Το σύστημα περιλαμβάνει ένα σύνολο από API endpoints που επιτρέπουν την επικοινωνία με τις συσκευές IoT. Τα API αυτά χρησιμοποιούν ασφαλή πρωτόκολλα και κρυπτογράφηση για την προστασία των δεδομένων κατά τη μεταφορά τους. Τα κύρια API endpoints περιλαμβάνουν την αποστολή δεδομένων (write API) και την ανάκτηση δεδομένων (read API).

Η βάση δεδομένων είναι σχεδιασμένη για να αποθηκεύει με ασφάλεια τα δεδομένα που συλλέγονται από τους αισθητήρες. Περιλαμβάνει πίνακες για τη διαχείριση χρηστών, καναλιών και δεδομένων

πεδίων. Οι σχέσεις μεταξύ των πινάκων εξασφαλίζουν την οργάνωση και την ακεραιότητα των δεδομένων.

Το σύστημα περιλαμβάνει μηχανισμούς για την επεξεργασία και ανάλυση των δεδομένων που συλλέγονται. Οι χρήστες μπορούν να προβάλουν τα δεδομένα τους σε δυναμικά γραφήματα, χρησιμοποιώντας τη βιβλιοθήκη Chart.js, η οποία προσφέρει διαδραστική και οπτική αναπαράσταση των πληροφοριών.

Το σύστημα IoT προσφέρει μια πληθώρα χαρακτηριστικών και λειτουργιών που το καθιστούν ένα ισχυρό εργαλείο για τη διαχείριση δεδομένων από συσκευές IoT.

Το σύστημα παρέχει λειτουργίες εγγραφής και σύνδεσης για τους χρήστες, επιτρέποντάς τους να δημιουργούν λογαριασμούς και να διαχειρίζονται τα κανάλια και τα δεδομένα τους. Η ασφάλεια των χρηστών εξασφαλίζεται μέσω κρυπτογράφησης κωδικών πρόσβασης και ασφαλών συνδέσεων HTTPS.

Οι χρήστες μπορούν να δημιουργούν και να διαχειρίζονται κανάλια δεδομένων. Κάθε κανάλι μπορεί να περιλαμβάνει πολλά πεδία δεδομένων, τα οποία μπορούν να ενεργοποιηθούν ή να απενεργοποιηθούν κατά βούληση. Οι χρήστες μπορούν επίσης να επεξεργάζονται τα χαρακτηριστικά των καναλιών, όπως το όνομα, την περιγραφή και τα API keys.

Το σύστημα υποστηρίζει τη συλλογή δεδομένων από αισθητήρες μέσω ασφαλών API endpoints. Τα δεδομένα αποθηκεύονται στη βάση δεδομένων με ασφαλή και οργανωμένο τρόπο, διασφαλίζοντας την ακεραιότητα και τη διαθεσιμότητά τους.

Οι χρήστες μπορούν να προβάλουν τα δεδομένα τους σε δυναμικά γραφήματα, τα οποία προσφέρουν μια οπτική αναπαράσταση των μετρήσεων. Αυτή η λειτουργία διευκολύνει την ανάλυση των τάσεων και των προτύπων, παρέχοντας πολύτιμες πληροφορίες για τη λήψη ενημερωμένων αποφάσεων.

Το σύστημα διαθέτει ισχυρά μέτρα ασφαλείας για την προστασία των δεδομένων και τη διασφάλιση της ιδιωτικότητας των χρηστών. Η χρήση API keys για την εγγραφή και ανάγνωση δεδομένων, η κρυπτογράφηση των κωδικών πρόσβασης και η επικύρωση των δεδομένων εισόδου συμβάλλουν στην αποτροπή μη εξουσιοδοτημένης πρόσβασης και επιθέσεων.

Αυτό το σύστημα IoT που περιγράφεται αποτελεί μια ισχυρή και ευέλικτη πλατφόρμα για τη διαχείριση δεδομένων από συσκευές IoT. Με την έμφαση στην ευκολία χρήσης, την ασφάλεια και την αναλυτική ικανότητα, το σύστημα αυτό παρέχει στους χρήστες τα εργαλεία που χρειάζονται για να συλλέγουν, να αποθηκεύουν και να αναλύουν δεδομένα από μια ποικιλία αισθητήρων και συσκευών. Οι δυνατότητες επέκτασης και προσαρμογής του συστήματος το καθιστούν κατάλληλο για ευρεία γκάμα εφαρμογών, από την οικιακή αυτοματοποίηση μέχρι τη βιομηχανική παρακολούθηση και συντήρηση, προσφέροντας αξιόπιστη και αποτελεσματική διαχείριση δεδομένων σε πραγματικό χρόνο.

1.2 Δομή της εργασίας

Η παρούσα εργασία είναι δομημένη σε πέντε κύρια κεφάλαια, καθένα από τα οποία συμβάλλει στην ανάπτυξη και την κατανόηση του συστήματος IoT που μελετάμε.

Στο πρώτο κεφάλαιο, παρέχεται μια εισαγωγή στο αντικείμενο της μελέτης, περιγράφοντας το σκοπό και τους στόχους της εργασίας, καθώς και τη δομή της.

Το δεύτερο κεφάλαιο περιλαμβάνει τη βιβλιογραφική ανασκόπηση των συστημάτων IoT, με εστίαση σε δημοφιλείς πλατφόρμες όπως το ThingSpeak, το ThingsBoard και το AWS IoT Core, καθώς και συγκρίσεις μεταξύ τους.

Στο τρίτο κεφάλαιο, αναλύονται τα τεχνολογικά εργαλεία που χρησιμοποιούνται, συμπεριλαμβανομένων των PHP, Laravel, Python και MySQL, με λεπτομέρειες για τις δυνατότητες και τη χρήση τους.

Το τέταρτο κεφάλαιο επικεντρώνεται στην ανάπτυξη του συστήματος IoT, περιγράφοντας τη διαδικασία, την εφαρμογή-ιστοσελίδα, το API και τη βάση δεδομένων, καθώς και τα μέτρα ασφάλειας για την προστασία του συστήματος και των δεδομένων.

Τέλος, το πέμπτο κεφάλαιο παρουσιάζει τα συμπεράσματα της μελέτης και προτείνει βελτιώσεις για το σύστημα.

Η εργασία ολοκληρώνεται με τη βιβλιογραφία και το παράρτημα, που παρέχουν επιπλέον πληροφορίες και τεκμηρίωση.

Κεφάλαιο 2ο: Βιβλιογραφική Ανασκόπηση Iot συστημάτων

2.1 Εισαγωγή

ThingSpeak

Το ThingSpeak είναι μια πλατφόρμα IoT που επιτρέπει τη συλλογή, αποθήκευση, ανάλυση και οπτικοποίηση δεδομένων από αισθητήρες και συσκευές IoT σε πραγματικό χρόνο [1][2]. Η πλατφόρμα προσφέρει εργαλεία για την εύκολη ενσωμάτωση των δεδομένων από τις συσκευές IoT και την επεξεργασία τους μέσω MATLAB. Οι χρήστες μπορούν να δημιουργούν κανάλια για την αποθήκευση δεδομένων και να χρησιμοποιούν ενσωματωμένα γραφήματα για την οπτικοποίηση των μετρήσεων. Το ThingSpeak υποστηρίζει την αποστολή δεδομένων μέσω HTTP, MQTT και άλλων πρωτοκόλλων, καθιστώντας το ιδιαίτερα ευέλικτο για διάφορες εφαρμογές. Ένα από τα κύρια πλεονεκτήματα της πλατφόρμας είναι η δυνατότητα εκτέλεσης MATLAB code απευθείας στο περιβάλλον του ThingSpeak, επιτρέποντας την ανάπτυξη σύνθετων αλγορίθμων ανάλυσης και επεξεργασίας δεδομένων [3].

ThingsBoard

Το ThingsBoard είναι μια ανοιχτή πλατφόρμα IoT που προσφέρει ισχυρές δυνατότητες διαχείρισης και ανάλυσης δεδομένων από διάφορες συσκευές IoT [4]. Η πλατφόρμα παρέχει εργαλεία για την τηλεμετρία, τον έλεγχο και την οπτικοποίηση δεδομένων μέσω ενός ευέλικτου και προσαρμόσιμου dashboard [5][6]. Οι χρήστες μπορούν να διαχειρίζονται συσκευές και αισθητήρες, να συλλέγουν δεδομένα και να δημιουργούν κανόνες για την αυτοματοποίηση των ενεργειών βάσει των δεδομένων. Το ThingsBoard υποστηρίζει πρωτόκολλα όπως MQTT, CoAP και HTTP, και ενσωματώνει δυνατότητες για την ανάλυση των δεδομένων και την ειδοποίηση σε πραγματικό χρόνο. Η πλατφόρμα μπορεί να επεκταθεί και να προσαρμοστεί σύμφωνα με τις ανάγκες του χρήστη, καθιστώντας την ιδανική για εφαρμογές σε βιομηχανικούς και εμπορικούς τομείς.

Kaa IoT Platform

Η Kaa IoT Platform είναι μια ανοιχτή πλατφόρμα που επιτρέπει τη διαχείριση συνδεδεμένων συσκευών και την ανάπτυξη εφαρμογών IoT [7][8]. Η πλατφόρμα προσφέρει ένα ευρύ φάσμα εργαλείων για τη συλλογή, την επεξεργασία και την ανάλυση δεδομένων από διάφορες συσκευές και αισθητήρες. Η αρχιτεκτονική της Kaa βασίζεται σε μικροϋπηρεσίες, επιτρέποντας την εύκολη κλιμάκωση και την προσαρμογή στις ανάγκες του χρήστη. Οι χρήστες μπορούν να ενσωματώσουν συσκευές μέσω πρωτοκόλλων όπως MQTT, CoAP και HTTP και να διαχειρίζονται τις ενημερώσεις λογισμικού των συσκευών εξ αποστάσεως. Η πλατφόρμα περιλαμβάνει επίσης εργαλεία για την οπτικοποίηση των δεδομένων και την ανάπτυξη κανόνων για την αυτοματοποίηση των ενεργειών βάσει των δεδομένων.

Losant

Το Losant είναι μια πλατφόρμα IoT που παρέχει εργαλεία για τη συλλογή, ανάλυση και οπτικοποίηση δεδομένων από συνδεδεμένες συσκευές [9]. Η πλατφόρμα υποστηρίζει την εύκολη ενσωμάτωση συσκευών μέσω πρωτοκόλλων όπως MQTT, HTTP και WebSockets, και προσφέρει δυνατότητες για τη δημιουργία προσαρμοσμένων workflows που αυτοματοποιούν τις διαδικασίες βάσει των δεδομένων [10]. Το Losant διαθέτει ένα ευέλικτο dashboard για την οπτικοποίηση των δεδομένων σε πραγματικό χρόνο, επιτρέποντας στους χρήστες να δημιουργούν γραφήματα, πίνακες και ειδοποιήσεις. Επιπλέον, η πλατφόρμα υποστηρίζει την ενσωμάτωση με τρίτες υπηρεσίες, όπως το AWS, το Azure και το Google Cloud, επιτρέποντας την ανάπτυξη σύνθετων εφαρμογών IoT σε κλίμακα.

OpenRemote

Το OpenRemote είναι μια ανοιχτή πλατφόρμα IoT που προσφέρει εργαλεία για τη διαχείριση και την οπτικοποίηση δεδομένων από συνδεδεμένες συσκευές [11]. Η πλατφόρμα επιτρέπει στους χρήστες να δημιουργούν προσαρμοσμένα dashboards, να διαχειρίζονται συσκευές και να αναπτύσσουν κανόνες για την αυτοματοποίηση των ενεργειών [12]. Το OpenRemote υποστηρίζει τη συλλογή δεδομένων μέσω διαφόρων πρωτοκόλλων επικοινωνίας και προσφέρει δυνατότητες για την επεξεργασία και ανάλυση των δεδομένων σε πραγματικό χρόνο. Οι χρήστες μπορούν επίσης να ενσωματώσουν την πλατφόρμα με άλλες υπηρεσίες και συστήματα, καθιστώντας την ιδανική για εφαρμογές σε έξυπνα κτίρια, έξυπνες πόλεις και βιομηχανικές εφαρμογές.

Cayenne

Το Cayenne είναι μια πλατφόρμα IoT που απευθύνεται κυρίως σε χρήστες που θέλουν να δημιουργήσουν γρήγορα και εύκολα εφαρμογές IoT χωρίς να απαιτούνται προχωρημένες γνώσεις προγραμματισμού [13]. Η πλατφόρμα προσφέρει ένα φιλικό προς τον χρήστη περιβάλλον για τη σύνδεση συσκευών, τη συλλογή δεδομένων και την ανάπτυξη κανόνων αυτοματισμού [14]. Οι χρήστες μπορούν να δημιουργούν προσαρμοσμένα dashboards για την οπτικοποίηση των δεδομένων, να δημιουργούν ειδοποιήσεις και να εκτελούν ενέργειες βάσει των δεδομένων. Το Cayenne υποστηρίζει πρωτόκολλα όπως MQTT, HTTP και LoRa, καθιστώντας το ιδανικό για εφαρμογές σε έξυπνα σπίτια, γεωργία και βιομηχανία.

Particle

Το Particle είναι μια πλατφόρμα IoT που προσφέρει εργαλεία για την ανάπτυξη, τη διαχείριση και την κλιμάκωση συνδεδεμένων προϊόντων [15]. Η πλατφόρμα περιλαμβάνει hardware συσκευές, λογισμικό και cloud υπηρεσίες που διευκολύνουν τη σύνδεση και τη διαχείριση συσκευών IoT [16]. Οι χρήστες μπορούν να συλλέγουν δεδομένα από τις συσκευές τους μέσω πρωτοκόλλων όπως MQTT και HTTP, να αποθηκεύουν τα δεδομένα στο cloud και να τα αναλύουν μέσω ενσωματωμένων εργαλείων. Το Particle προσφέρει επίσης δυνατότητες για την απομακρυσμένη διαχείριση και την ενημέρωση του λογισμικού των συσκευών, επιτρέποντας στους χρήστες να διατηρούν τα προϊόντα τους ενημερωμένα και ασφαλή. Η πλατφόρμα είναι ιδανική για επιχειρήσεις που θέλουν να αναπτύξουν συνδεδεμένα προϊόντα σε μεγάλη κλίμακα.

AWS IoT Core

Το AWS IoT Core είναι μια διαχειριζόμενη υπηρεσία από την Amazon Web Services που επιτρέπει τη σύνδεση συσκευών IoT στο cloud και τη διαχείριση των δεδομένων τους με ασφαλή και επεκτάσιμο τρόπο [17]. Η πλατφόρμα προσφέρει δυνατότητες για την εύκολη ενσωμάτωση συσκευών μέσω πρωτοκόλλων όπως MQTT, HTTP και WebSockets, και επιτρέπει τη συλλογή, επεξεργασία και ανάλυση δεδομένων σε πραγματικό χρόνο [18]. Το AWS IoT Core υποστηρίζει την αυτοματοποίηση και τη διαχείριση κανόνων μέσω της υπηρεσίας AWS IoT Rules Engine, ενώ ενσωματώνεται άμεσα με άλλες υπηρεσίες AWS, όπως το AWS Lambda, το Amazon S3 και το Amazon Kinesis. Αυτή η ενσωμάτωση επιτρέπει την ανάπτυξη σύνθετων και ισχυρών εφαρμογών IoT, εξασφαλίζοντας παράλληλα την ασφάλεια και την κλιμάκωση των δεδομένων και των συσκευών.

2.2 ThingSpeak

Το ThingSpeak είναι μια ισχυρή πλατφόρμα IoT που παρέχει ένα ευρύ φάσμα χαρακτηριστικών και δυνατοτήτων για τη συλλογή, αποθήκευση, ανάλυση και οπτικοποίηση δεδομένων από αισθητήρες και συσκευές IoT σε πραγματικό χρόνο. Ένα από τα κύρια χαρακτηριστικά της πλατφόρμας είναι η δυνατότητα δημιουργίας καναλιών όπου οι χρήστες μπορούν να αποθηκεύουν δεδομένα σε έως και οκτώ πεδία ανά κανάλι. Η πλατφόρμα υποστηρίζει την αποστολή δεδομένων μέσω HTTP, MQTT και άλλων πρωτοκόλλων, κάνοντάς την ευέλικτη για διάφορες εφαρμογές. Ένα από τα βασικά πλεονεκτήματα του ThingSpeak είναι η ενσωμάτωση με το MATLAB, που επιτρέπει στους χρήστες να εκτελούν σύνθετους αλγόριθμους ανάλυσης και επεξεργασίας δεδομένων απευθείας στο περιβάλλον της πλατφόρμας. Οι δυνατότητες οπτικοποίησης περιλαμβάνουν τη δημιουργία διαδραστικών γραφημάτων και πινάκων που βοηθούν τους χρήστες να αναλύουν τις τάσεις και τα μοτίβα των δεδομένων. Επιπλέον, το ThingSpeak υποστηρίζει τη δημιουργία ειδοποιήσεων και την εκτέλεση

ενεργειών βάσει των δεδομένων μέσω της υπηρεσίας ThingHTTP και των προσαρμόσιμων Webhooks. Σχετικά με τις τιμές, το ThingSpeak προσφέρει ένα δωρεάν βασικό πακέτο που επιτρέπει τη δημιουργία καναλιών και τη συλλογή δεδομένων με περιορισμούς στη συχνότητα αποστολής δεδομένων (μέχρι 3 εκατομμύρια μηνύματα ανά έτος). Για πιο απαιτητικές εφαρμογές, υπάρχουν εμπορικά πακέτα με συνδρομή που προσφέρουν υψηλότερα όρια αποστολής δεδομένων, επιπλέον δυνατότητες ασφάλειας και υποστήριξης, καθώς και πρόσβαση σε πρόσθετες λειτουργίες του MATLAB. Συνολικά, το ThingSpeak παρέχει μια ολοκληρωμένη και ευέλικτη λύση για τη διαχείριση δεδομένων IoT, προσφέροντας παράλληλα ισχυρά εργαλεία ανάλυσης και οπτικοποίησης σε ανταγωνιστικές τιμές.

2.3 ThingsBoard

Το ThingsBoard είναι μια ανοιχτή πλατφόρμα IoT που προσφέρει ένα ευρύ φάσμα χαρακτηριστικών και δυνατοτήτων για τη διαχείριση και ανάλυση δεδομένων από διάφορες συσκευές IoT. Η πλατφόρμα υποστηρίζει τη συλλογή δεδομένων από συσκευές μέσω πρωτοκόλλων όπως MQTT, CoAP και HTTP, επιτρέποντας την εύκολη ενσωμάτωση με ποικίλες συσκευές και αισθητήρες. Ένα από τα βασικά χαρακτηριστικά του ThingsBoard είναι η δυνατότητα δημιουργίας προσαρμοσμένων dashboards που επιτρέπουν την οπτικοποίηση των δεδομένων σε πραγματικό χρόνο μέσω γραφημάτων, χαρτών, πινάκων και άλλων διαδραστικών widgets. Η πλατφόρμα παρέχει επίσης εργαλεία για την τηλεμετρία και τον έλεγχο συσκευών, επιτρέποντας στους χρήστες να διαχειρίζονται τις συσκευές τους και να εκτελούν ενέργειες εξ αποστάσεως. Επιπλέον, το ThingsBoard διαθέτει έναν ισχυρό μηχανισμό κανόνων που επιτρέπει την αυτοματοποίηση των ενεργειών βάσει συγκεκριμένων συνθηκών, όπως η αποστολή ειδοποιήσεων ή η εκτέλεση ενεργειών όταν τα δεδομένα υπερβαίνουν ορισμένα όρια. Η πλατφόρμα υποστηρίζει την κλιμάκωση και μπορεί να προσαρμοστεί για να καλύψει τις ανάγκες μικρών και μεγάλων εφαρμογών IoT, από μικρές εγκαταστάσεις μέχρι βιομηχανικές εφαρμογές σε μεγάλη κλίμακα. Σχετικά με τις τιμές, το ThingsBoard προσφέρει μια δωρεάν ανοιχτού κώδικα έκδοση που είναι ιδανική για μικρότερες εφαρμογές και προγραμματιστές που θέλουν να πειραματιστούν με την πλατφόρμα. Για πιο απαιτητικές εφαρμογές, υπάρχουν εμπορικά πακέτα με συνδρομή που παρέχουν πρόσθετες λειτουργίες, όπως προηγμένη ασφάλεια, υποστήριξη πολλαπλών ενοικιαστών (multi-tenancy), και επαγγελματική υποστήριξη. Τα εμπορικά πακέτα είναι διαθέσιμα σε διάφορα επίπεδα, προσφέροντας επιλογές που μπορούν να καλύψουν τις ανάγκες τόσο μικρών επιχειρήσεων όσο και μεγάλων βιομηχανικών πελατών. Συνολικά, το ThingsBoard προσφέρει μια ολοκληρωμένη, ευέλικτη και επεκτάσιμη λύση για τη διαχείριση συσκευών IoT και την ανάλυση δεδομένων, παρέχοντας ισχυρά εργαλεία και δυνατότητες σε ανταγωνιστικές τιμές.

2.4 AWS IoT Core

Το AWS IoT Core είναι μια πλήρως διαχειριζόμενη υπηρεσία από την Amazon Web Services που επιτρέπει τη σύνδεση, τη διαχείριση και την ασφάλεια δισεκατομμυρίων συσκευών IoT. Ένα από τα κύρια χαρακτηριστικά του AWS IoT Core είναι η υποστήριξη πρωτοκόλλων επικοινωνίας όπως MQTT, HTTP και WebSockets, επιτρέποντας την εύκολη και ασφαλή επικοινωνία μεταξύ των συσκευών και του cloud. Η υπηρεσία προσφέρει δυνατότητες για την αυθεντικοποίηση και την εξουσιοδότηση των συσκευών μέσω X.509 πιστοποιητικών, Amazon Cognito και άλλων μηχανισμών ασφαλείας, διασφαλίζοντας την προστασία των δεδομένων και την αποτροπή μη εξουσιοδοτημένης πρόσβασης. Επιπλέον, το AWS IoT Core περιλαμβάνει τον AWS IoT Rules Engine, ο οποίος επιτρέπει τη δημιουργία κανόνων για την επεξεργασία των δεδομένων σε πραγματικό χρόνο και την ενεργοποίηση ενεργειών βάσει συγκεκριμένων συνθηκών, όπως η αποστολή ειδοποιήσεων, η αποθήκευση δεδομένων στο Amazon S3 ή η εκτέλεση λειτουργιών στο AWS Lambda. Η υπηρεσία προσφέρει επίσης τη δυνατότητα δημιουργίας σκιών συσκευών (device shadows), που επιτρέπουν την αποθήκευση και την ανάκτηση της κατάστασης των συσκευών, διευκολύνοντας την επικοινωνία και τον έλεγχο σε καταστάσεις όπου οι συσκευές είναι προσωρινά εκτός σύνδεσης. Το AWS IoT Core ενσωματώνεται άμεσα με άλλες υπηρεσίες AWS, επιτρέποντας την ανάπτυξη σύνθετων και ισχυρών εφαρμογών IoT που εκμεταλλεύονται τις δυνατότητες ανάλυσης, αποθήκευσης και μηχανικής μάθησης του AWS. Σχετικά με τις τιμές, το AWS IoT Core ακολουθεί μια μοντέλο πληρωμής ανά χρήση, όπου οι χρήστες χρεώνονται βάσει του αριθμού των μηνυμάτων που αποστέλλονται και λαμβάνονται, των κανόνων που εκτελούνται και των σκιών συσκευών που διατηρούνται. Αυτό το μοντέλο τιμολόγησης επιτρέπει την κλιμάκωση των δαπανών σύμφωνα με την ανάπτυξη της εφαρμογής, καθιστώντας το AWS IoT Core μια προσιτή και ευέλικτη επιλογή για επιχειρήσεις κάθε μεγέθους. Συνολικά, το AWS IoT Core προσφέρει μια ολοκληρωμένη και ισχυρή πλατφόρμα για τη διαχείριση συσκευών IoT και την επεξεργασία δεδομένων, παρέχοντας υψηλή ασφάλεια, αξιοπιστία και δυνατότητες κλιμάκωσης.

2.5 Συγκρίσεις

Το ThingSpeak, το ThingsBoard και το AWS IoT Core είναι τρεις διαφορετικές πλατφόρμες IoT, καθεμία με τα δικά της μοναδικά χαρακτηριστικά, δυνατότητες και προτάσεις αξίας. Το ThingSpeak ξεχωρίζει για την απλότητα και την ευκολία χρήσης του, ιδιαίτερα για χρήστες που αναζητούν γρήγορη και απλή ενσωμάτωση με δυνατότητες ανάλυσης δεδομένων μέσω MATLAB. Προσφέρει βασικά χαρακτηριστικά όπως η συλλογή και αποθήκευση δεδομένων σε κανάλια, η δημιουργία γραφημάτων και η εκτέλεση MATLAB code για την ανάλυση των δεδομένων. Οι τιμές του ThingSpeak είναι ανταγωνιστικές, με ένα δωρεάν βασικό πακέτο και εμπορικά πακέτα για πιο απαιτητικές εφαρμογές. Από την άλλη πλευρά, το ThingsBoard είναι μια ανοιχτή πλατφόρμα που προσφέρει ευελιξία και

επεκτασιμότητα για διαχείριση και ανάλυση δεδομένων σε βιομηχανικό επίπεδο. Υποστηρίζει πολλαπλά πρωτόκολλα επικοινωνίας, διαθέτει έναν ισχυρό μηχανισμό κανόνων για αυτοματοποίηση και προσφέρει προσαρμόσιμα dashboards για οπτικοποίηση δεδομένων σε πραγματικό χρόνο. Το ThingsBoard είναι διαθέσιμο σε δωρεάν ανοιχτού κώδικα έκδοση και εμπορικά πακέτα που παρέχουν πρόσθετες λειτουργίες και επαγγελματική υποστήριξη. Τέλος, το AWS IoT Core, ως μέρος του οικοσυστήματος της Amazon Web Services, προσφέρει απαράμιλλη κλιμάκωση και ενσωμάτωση με άλλες υπηρεσίες AWS. Υποστηρίζει πρωτόκολλα όπως MQTT, HTTP και WebSockets, και παρέχει υψηλή ασφάλεια μέσω X.509 πιστοποιητικών και άλλων μηχανισμών. Η δυνατότητα δημιουργίας σκιών συσκευών και η χρήση του AWS IoT Rules Engine για την αυτοματοποίηση ενεργειών και την επεξεργασία δεδομένων σε πραγματικό χρόνο καθιστούν το AWS IoT Core ιδανικό για μεγάλες και σύνθετες εφαρμογές. Το μοντέλο τιμολόγησης του AWS IoT Core βασίζεται στη χρήση, επιτρέποντας στις επιχειρήσεις να κλιμακώσουν τις δαπάνες τους ανάλογα με τις ανάγκες τους. Συνολικά, το ThingSpeak είναι ιδανικό για απλές και γρήγορες αναπτύξεις, το ThingsBoard προσφέρει ευελιξία και δυνατότητες για βιομηχανικές εφαρμογές, ενώ το AWS IoT Core παρέχει ολοκληρωμένες και ισχυρές λύσεις για μεγάλες κλίμακες και σύνθετες ανάγκες.

Κεφάλαιο 3ο: Τα τεχνολογικά εργαλεία

Στο κεφάλαιο αυτό θα περιγράφουν τα τεχνολογικά εργαλεία που χρησιμοποιήθηκε για την υλοποίηση του συστήματος IoT.

3.1 PHP

Η PHP (Hypertext Preprocessor) είναι μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού για την ανάπτυξη ιστοσελίδων και web εφαρμογών. Αρχικά αναπτύχθηκε από τον Rasmus Lerdorf το 1994 ως ένα απλό σύνολο από CGI scripts γραμμένα σε C για να παρακολουθεί τους επισκέπτες στην προσωπική του ιστοσελίδα. Από τότε, η PHP έχει εξελιχθεί σε μια πλήρως ανεπτυγμένη γλώσσα προγραμματισμού με πλούσιο σύνολο χαρακτηριστικών και μεγάλη κοινότητα χρηστών [19][20]. Η πρώτη έκδοση της PHP, γνωστή ως PHP/FI (Personal Home Page/Forms Interpreter), κυκλοφόρησε το 1995 και περιλάμβανε βασικές δυνατότητες όπως η διαχείριση φορμών και η επικοινωνία με βάσεις δεδομένων. Το 1997, η PHP/FI 2.0 κυκλοφόρησε, προσθέτοντας νέα χαρακτηριστικά και βελτιώσεις. Το πραγματικό άλμα για την PHP ήρθε με την έκδοση 3.0, που κυκλοφόρησε το 1998, όταν οι Zeev Suraski και Andi Gutmans ξανάγραψαν τον parser της PHP και την επαναπροσδιόρισαν ως PHP: Hypertext Preprocessor. Η PHP 4.0, που κυκλοφόρησε το 2000, βασίστηκε στον νέο Zend Engine και πρόσθεσε σημαντικές βελτιώσεις στην απόδοση και τη διαχείριση των πόρων. Η PHP 5.0 κυκλοφόρησε το 2004 και έφερε την υποστήριξη για αντικειμενοστραφή προγραμματισμό (OOP), καθιστώντας την PHP πιο ισχυρή και ευέλικτη για την ανάπτυξη σύνθετων εφαρμογών. Η πιο πρόσφατη σταθερή έκδοση, η PHP 8.0, κυκλοφόρησε το 2020 και περιλαμβάνει πολλές βελτιώσεις απόδοσης και νέες δυνατότητες όπως η Just-In-Time (JIT) compilation και οι συναρτήσεις.

Η PHP συχνά συγκρίνεται με άλλες γλώσσες προγραμματισμού για το web όπως η Python, η Ruby και η JavaScript. Ένα από τα κύρια πλεονεκτήματα της PHP είναι η ευκολία χρήσης και η γρήγορη καμπύλη μάθησης, καθιστώντας την ιδανική για αρχάριους. Σε αντίθεση με την Python, που είναι μια γενικής χρήσης γλώσσα, η PHP είναι σχεδιασμένη ειδικά για την ανάπτυξη web εφαρμογών, πράγμα που την καθιστά πιο εξειδικευμένη σε αυτόν τον τομέα. Η Ruby on Rails προσφέρει μια πιο αυστηρά καθοδηγούμενη προσέγγιση στην ανάπτυξη web εφαρμογών, ενώ η PHP παρέχει μεγαλύτερη ευελιξία και ελευθερία στον προγραμματιστή. Η JavaScript, αν και κυριαρχεί στην ανάπτυξη frontend, χρησιμοποιείται επίσης στο backend μέσω του Node.js, προσφέροντας μια ενιαία γλώσσα για όλο το stack, κάτι που η PHP δεν προσφέρει άμεσα.

Η PHP προτιμάται για πολλούς λόγους, συμπεριλαμβανομένης της ευκολίας μάθησης, της ευρείας υποστήριξης από παρόχους φιλοξενίας και της μεγάλης κοινότητας χρηστών και προγραμματιστών. Η γλώσσα παρέχει επίσης ισχυρή ενσωμάτωση με πολλές βάσεις δεδομένων, όπως MySQL, PostgreSQL

και SQLite, καθώς και υποστήριξη για διάφορα πρωτόκολλα, όπως HTTP, FTP και IMAP. Επιπλέον, η PHP έχει ένα μεγάλο οικοσύστημα από frameworks και εργαλεία, όπως το Laravel, το Symfony και το CodeIgniter, που διευκολύνουν την ανάπτυξη και τη συντήρηση εφαρμογών.

Η PHP έχει πολλά πλεονεκτήματα, όπως:

- Η σύνταξη της PHP είναι απλή και ευκολονόητη, καθιστώντας την ιδανική για αρχάριους.
- Η PHP μπορεί να ενσωματωθεί εύκολα με HTML, καθιστώντας την ιδανική για την ανάπτυξη δυναμικών ιστοσελίδων.
- Η PHP έχει μια τεράστια κοινότητα χρηστών που παρέχει υποστήριξη και πόρους.
- Υπάρχουν πολλά frameworks και εργαλεία διαθέσιμα που διευκολύνουν την ανάπτυξη.

Ωστόσο, η PHP έχει και κάποια μειονεκτήματα:

- Παρόλο που η PHP έχει βελτιωθεί σημαντικά όσον αφορά την ασφάλεια, η κακή διαχείριση κώδικα μπορεί να οδηγήσει σε ευπάθειες.
- Αν και οι πρόσφατες εκδόσεις έχουν βελτιώσει την απόδοση, η PHP μπορεί να είναι πιο αργή σε σύγκριση με άλλες γλώσσες όπως η Node.js για συγκεκριμένα είδη εφαρμογών.
- Η PHP εκτελείται στο server, έτσι δεν μπορεί να χρησιμοποιηθεί για τη δημιουργία λογικής στον client-side χωρίς τη βοήθεια άλλων τεχνολογιών όπως η JavaScript.

Η PHP παραμένει μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού για την ανάπτυξη ιστοσελίδων και web εφαρμογών. Σύμφωνα με το W3Techs, περίπου το 78% όλων των ιστοσελίδων χρησιμοποιούν PHP στην πλευρά του διακομιστή. Μερικά από τα πιο γνωστά CMS (Content Management Systems) όπως το WordPress, το Joomla και το Drupal είναι γραμμένα σε PHP, γεγονός που συμβάλλει στην ευρεία διάδοση και χρήση της γλώσσας. Η PHP χρησιμοποιείται επίσης από μεγάλες πλατφόρμες και υπηρεσίες όπως το Facebook και το Wikipedia, αποδεικνύοντας την αξιοπιστία και την κλιμακωσιμότητα της.

3.2 Laravel

Το Laravel είναι ένα από τα πιο δημοφιλή και ισχυρά PHP frameworks που χρησιμοποιούνται για την ανάπτυξη web εφαρμογών. Δημιουργήθηκε από τον Taylor Otwell το 2011 με σκοπό να προσφέρει έναν πιο εκλεπτυσμένο και ευέλικτο τρόπο για την ανάπτυξη εφαρμογών σε PHP, γεφυρώνοντας τα κενά που υπήρχαν σε άλλα frameworks εκείνης της εποχής. Το Laravel βασίζεται σε έννοιες όπως η

καθαρή αρχιτεκτονική, η modular δομή και η ευκολία στη χρήση, καθιστώντας το ιδανικό τόσο για αρχάριους όσο και για έμπειρους προγραμματιστές [21].

Το Laravel ξεκίνησε το 2011 με την έκδοση 1.0, η οποία έφερε βασικά χαρακτηριστικά όπως το routing, τη διαχείριση συνεδριών και την επικύρωση. Η έκδοση 2.0, που κυκλοφόρησε λίγο αργότερα, προσέθεσε τη δυνατότητα για bundles, κάτι σαν πακέτα που επιτρέπουν την επαναχρησιμοποίηση κώδικα. Το Laravel 3.0, κυκλοφόρησε το 2012 και εισήγαγε σημαντικά χαρακτηριστικά όπως το Artisan CLI, το Eloquent ORM και το σύστημα συσκευασίας. Το Laravel 4.0, γνωστό και ως "Illuminate", αποτέλεσε μια πλήρη ανασχεδίαση βασισμένη στο Composer, το σύστημα διαχείρισης εξαρτήσεων της PHP, προσφέροντας μεγαλύτερη ευελιξία και αρθρωτότητα. Το Laravel 5.0, που κυκλοφόρησε το 2015, έφερε πολλές βελτιώσεις όπως το Scheduler, τα Events, και το Middleware. Η πιο πρόσφατη έκδοση, το Laravel 8.0, εισήγαγε βελτιώσεις στην απόδοση, βελτιώσεις στα jobs και queues, και το Laravel Jetstream για την απλούστευση της διαχείρισης ταυτότητας και του scaffolding.

Το Laravel συχνά συγκρίνεται με άλλα δημοφιλή PHP frameworks όπως το Symfony, το CodeIgniter και το Yii. Το Laravel προσφέρει μια πιο απλή και ευέλικτη προσέγγιση σε σχέση με το Symfony, το οποίο είναι πιο περίπλοκο αλλά και πιο ισχυρό και προσαρμόσιμο για μεγάλες επιχειρηματικές εφαρμογές. Σε σύγκριση με το CodeIgniter, το Laravel παρέχει περισσότερες δυνατότητες και μια πιο μοντέρνα αρχιτεκτονική, αν και το CodeIgniter είναι γνωστό για την ταχύτητα και την ευκολία στη χρήση του. Το Yii, από την άλλη πλευρά, προσφέρει παρόμοιες δυνατότητες με το Laravel αλλά δεν έχει την ίδια ευρεία υιοθέτηση και κοινότητα. Το Laravel είναι επίσης γνωστό για την κομψότητα και την απλότητα του κώδικά του, καθιστώντας το πιο προσιτό για νέους προγραμματιστές.

Το Laravel προτιμάται από πολλούς προγραμματιστές για διάφορους λόγους. Πρώτον, προσφέρει μια καθαρή και ευανάγνωστη σύνταξη, που διευκολύνει την ανάπτυξη και τη συντήρηση κώδικα. Δεύτερον, παρέχει πολλά ενσωματωμένα εργαλεία και βιβλιοθήκες, όπως το Eloquent ORM για αλληλεπίδραση με τη βάση δεδομένων, το Artisan CLI για αυτοματοποίηση εργασιών και το Blade για δυναμική δημιουργία προβολών. Επιπλέον, το Laravel ενσωματώνει το σύστημα διαχείρισης εξαρτήσεων Composer, διευκολύνοντας την ενσωμάτωση τρίτων βιβλιοθηκών και πακέτων. Το Laravel υποστηρίζει επίσης την ανάπτυξη RESTful APIs και έχει ισχυρή υποστήριξη για δοκιμές, επιτρέποντας στους προγραμματιστές να γράφουν εύκολα και αξιόπιστα τεστ για τον κώδικά τους.

Το Laravel έχει πολλά πλεονεκτήματα:

- Διευκολύνει την ανάπτυξη και τη συντήρηση κώδικα.
- Περιλαμβάνει πολλά ενσωματωμένα εργαλεία που απλοποιούν κοινές εργασίες, όπως το Eloquent ORM και το Artisan CLI.

- Μεγάλη κοινότητα χρηστών που προσφέρει υποστήριξη, οδηγούς και πόρους.
- Υποστηρίζει τη δημιουργία τόσο μικρών όσο και μεγάλων εφαρμογών.

Ωστόσο, το Laravel έχει και κάποια μειονεκτήματα:

- Παρά την ευκολία χρήσης του, μπορεί να έχει μια αρχική καμπύλη μάθησης για αρχάριους προγραμματιστές.
- Αν και βελτιώνεται συνεχώς, η απόδοση του Laravel μπορεί να υστερεί σε σύγκριση με πιο ελαφριά frameworks.
- Οι πολλαπλές δυνατότητες και τα χαρακτηριστικά μπορεί να προκαλέσουν πολυπλοκότητα στην ανάπτυξη και τη συντήρηση μεγάλων έργων.

Το Laravel είναι ένα από τα πιο δημοφιλή PHP frameworks, με μια τεράστια κοινότητα χρηστών και προγραμματιστών. Σύμφωνα με διάφορες έρευνες και στατιστικές, το Laravel χρησιμοποιείται από χιλιάδες επιχειρήσεις και οργανισμούς παγκοσμίως. Μερικά από τα πιο γνωστά έργα που έχουν αναπτυχθεί με το Laravel περιλαμβάνουν το Laravel Forge, μια υπηρεσία διαχείρισης διακομιστών, και το Laravel Vapor, μια πλατφόρμα ανάπτυξης χωρίς διακομιστές. Η δημοφιλία του Laravel οφείλεται στην ευκολία χρήσης, την εκτεταμένη τεκμηρίωση και την ενεργή κοινότητα που προσφέρει συνεχή υποστήριξη και ανάπτυξη. Η ευελιξία του το καθιστά κατάλληλο για διάφορες εφαρμογές, από μικρά ιστολόγια μέχρι μεγάλες επιχειρηματικές εφαρμογές και πλατφόρμες.

Το Laravel είναι ένα ισχυρό και ευέλικτο PHP framework που προσφέρει ένα εκτεταμένο σύνολο εργαλείων και δυνατοτήτων για την ανάπτυξη σύγχρονων web εφαρμογών. Παρά τα μειονεκτήματά του, τα πλεονεκτήματα και η δημοφιλία του το καθιστούν μια από τις καλύτερες επιλογές για προγραμματιστές που επιθυμούν να δημιουργήσουν καθαρές, αποδοτικές και κλιμακούμενες web εφαρμογές.

3.3 Python

Η Python είναι μια από τις πιο δημοφιλείς και ευέλικτες γλώσσες προγραμματισμού στον κόσμο. Δημιουργήθηκε από τον Guido van Rossum και κυκλοφόρησε για πρώτη φορά το 1991. Σχεδιάστηκε με σκοπό να είναι εύκολη στην ανάγνωση και τη σύνταξη, καθιστώντας την ιδανική για αρχάριους προγραμματιστές, ενώ ταυτόχρονα είναι αρκετά ισχυρή για να χρησιμοποιείται σε μεγάλα και σύνθετα έργα. Η Python έχει εξελιχθεί μέσα από πολλές εκδόσεις και έχει υιοθετηθεί ευρέως σε διάφορους τομείς, όπως η επιστήμη δεδομένων, η ανάπτυξη web εφαρμογών, η αυτοματοποίηση και η τεχνητή

νοημοσύνη [22][23]. Η Python ξεκίνησε ως ένα έργο χόμπι του Guido van Rossum στα τέλη της δεκαετίας του 1980 και κυκλοφόρησε για πρώτη φορά το 1991. Η πρώτη έκδοση, Python 1.0, προσέφερε βασικά χαρακτηριστικά όπως η διαχείριση εξαιρέσεων και η υποστήριξη για modules. Η Python 2.0 κυκλοφόρησε το 2000, εισάγοντας σημαντικά νέα χαρακτηριστικά όπως η συλλογή σκουπιδιών (garbage collection) και η λίστα κατανόησης (list comprehensions). Ωστόσο, η Python 3.0, που κυκλοφόρησε το 2008, αποτέλεσε μια μεγάλη αναθεώρηση της γλώσσας με πολλές αλλαγές που έσπασαν την συμβατότητα προς τα πίσω. Η Python 3 έφερε βελτιώσεις στη συνέπεια και την απλότητα της γλώσσας, αλλάζοντας συντακτικά στοιχεία και απομακρύνοντας παλαιότερες πρακτικές. Σήμερα, η πιο πρόσφατη σταθερή έκδοση είναι η Python 3.9, η οποία συνεχίζει να εξελίσσεται με νέες δυνατότητες και βελτιώσεις στην απόδοση [24].

Η Python συγκρίνεται συχνά με άλλες δημοφιλείς γλώσσες προγραμματισμού όπως η Java, η C++ και η JavaScript. Ένα από τα κύρια πλεονεκτήματα της Python είναι η απλότητα και η καθαρή της σύνταξη, η οποία την καθιστά πιο εύκολη στη μάθηση και τη χρήση σε σύγκριση με τη Java και τη C++, που έχουν πιο περίπλοκες και αυστηρές συντακτικές δομές. Σε αντίθεση με τη JavaScript, που κυριαρχεί στον τομέα της ανάπτυξης frontend, η Python είναι πιο ισχυρή για εργασίες backend και scripting. Η Python είναι επίσης ευρέως χρησιμοποιούμενη στην επιστήμη δεδομένων και την τεχνητή νοημοσύνη, λόγω των ισχυρών βιβλιοθηκών της όπως το NumPy, το pandas και το TensorFlow, ενώ η R χρησιμοποιείται επίσης στην ανάλυση δεδομένων αλλά είναι λιγότερο ευέλικτη για γενικό προγραμματισμό.

Η Python προτιμάται για πολλούς λόγους. Πρώτον, η απλότητά της και η ευανάγνωστη σύνταξή της την καθιστούν ιδανική για αρχάριους. Δεύτερον, η ευελιξία της την καθιστά κατάλληλη για διάφορες εφαρμογές, από web ανάπτυξη και αυτοματοποίηση έως ανάλυση δεδομένων και τεχνητή νοημοσύνη. Τρίτον, η μεγάλη κοινότητα χρηστών και προγραμματιστών προσφέρει πλούσια τεκμηρίωση, παραδείγματα κώδικα και υποστήριξη. Τέλος, οι ισχυρές βιβλιοθήκες και τα frameworks της Python, όπως το Django και το Flask για web ανάπτυξη, και το NumPy και το SciPy για επιστημονικούς υπολογισμούς, διευκολύνουν την ανάπτυξη εφαρμογών και την επίλυση σύνθετων προβλημάτων.

Η Python έχει πολλά πλεονεκτήματα:

- Η καθαρή και απλή σύνταξη της Python την καθιστά ιδανική για αρχάριους.
- Μπορεί να χρησιμοποιηθεί για μια ποικιλία εφαρμογών, από web ανάπτυξη και scripting έως επιστημονικούς υπολογισμούς και τεχνητή νοημοσύνη.
- Η Python διαθέτει μια από τις μεγαλύτερες κοινότητες προγραμματιστών, παρέχοντας πλούσια τεκμηρίωση και υποστήριξη.
- Πολλές βιβλιοθήκες και frameworks διευκολύνουν την ανάπτυξη και επιταχύνουν την επίλυση προβλημάτων.

Η Python έχει και κάποια μειονεκτήματα:

- Η Python είναι πιο αργή σε σύγκριση με γλώσσες όπως η C++ και η Java, λόγω της ερμηνευμένης φύσης της.
- Οι εφαρμογές Python μπορεί να απαιτούν περισσότερους πόρους συστήματος.
- Η μετάβαση από την Python 2 στην Python 3 μπορεί να είναι δύσκολη λόγω ασυμβατότητας προς τα πίσω.

Η Python είναι μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού στον κόσμο, καταλαμβάνοντας συχνά τις πρώτες θέσεις σε διάφορους δείκτες δημοφιλίας, όπως το TIOBE Index και το PYPL Popularity of Programming Language. Η ευρεία χρήση της Python καλύπτει διάφορους τομείς, συμπεριλαμβανομένης της web ανάπτυξης, της επιστήμης δεδομένων, της μηχανικής μάθησης, της αυτοματοποίησης, των παιχνιδιών και των εφαρμογών επιφάνειας εργασίας. Μεγάλες εταιρείες και οργανισμοί όπως η Google, η NASA, το CERN και πολλές άλλες χρησιμοποιούν την Python για να αναπτύξουν και να διαχειριστούν κρίσιμες εφαρμογές και υπηρεσίες.

Η Python χρησιμοποιείται επίσης ευρέως στην εκπαίδευση, με πολλά πανεπιστήμια και σχολεία να τη διδάσκουν ως την πρώτη γλώσσα προγραμματισμού. Η ευκολία της στη μάθηση και η ευρεία εφαρμογή της την καθιστούν ιδανική για την εισαγωγή των μαθητών στον κόσμο του προγραμματισμού. Είναι μια ευέλικτη, ισχυρή και ευκολονόητη γλώσσα προγραμματισμού που έχει κερδίσει τη δημοτικότητα και την αποδοχή της παγκόσμιας κοινότητας των προγραμματιστών. Παρά τα μειονεκτήματά της, τα πλεονεκτήματα και η εκτεταμένη χρήση της την καθιστούν μια από τις καλύτερες επιλογές για την ανάπτυξη μιας ευρείας ποικιλίας

3.3.1 API Python

Η ανάπτυξη API (Application Programming Interface) με τη χρήση της Python είναι μια δημοφιλής πρακτική, κυρίως λόγω της απλότητας και της ισχύος που προσφέρει η γλώσσα [25]. Τα APIs επιτρέπουν τη διασύνδεση και την επικοινωνία μεταξύ διαφόρων εφαρμογών, διευκολύνοντας την ανταλλαγή δεδομένων και λειτουργιών. Το Flask και το Django είναι δύο από τα πιο δημοφιλή frameworks για την ανάπτυξη web εφαρμογών και APIs στην Python. Το Laravel, από την άλλη πλευρά, είναι ένα ισχυρό PHP framework που χρησιμοποιείται ευρέως για την ανάπτυξη web εφαρμογών και APIs. Σε αυτό το κεφάλαιο, θα εξετάσουμε πώς μπορούμε να αναπτύξουμε ένα API χρησιμοποιώντας Python και πώς αυτό μπορεί να επικοινωνήσει με μια εφαρμογή Laravel [26]. Για

την ανάπτυξη API στην Python, το Flask και το Django είναι οι δύο πιο κοινά χρησιμοποιούμενες επιλογές. Το Flask είναι ένα μικρό και ελαφρύ framework που προσφέρει μεγάλη ευελιξία, ενώ το Django είναι ένα πιο ολοκληρωμένο και ισχυρό framework που παρέχει πολλές ενσωματωμένες λειτουργίες.

Το Flask είναι ένα micro-framework που επιτρέπει την ανάπτυξη μικρών και ελαφριών web εφαρμογών. Είναι ιδανικό για τη δημιουργία APIs λόγω της απλότητας και της ευελιξίας του.

```
from flask import Flask, request, jsonify


app = Flask(__name__)


@app.route('/api/data', methods=['GET', 'POST'])
def data():
    if request.method == 'POST':
        data = request.json
        return jsonify({"message": "Data received", "data": data}), 201
    else:
        sample_data = {"key": "value"}
        return jsonify(sample_data)


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

Επικοινωνία με το Laravel

Η επικοινωνία μεταξύ ενός API που έχει αναπτυχθεί με Python και μιας εφαρμογής Laravel μπορεί να επιτευχθεί μέσω HTTP αιτημάτων. Το Laravel μπορεί να στείλει και να λάβει δεδομένα από το API της Python χρησιμοποιώντας το Guzzle HTTP client ή τις ενσωματωμένες λειτουργίες του Laravel για HTTP αιτήματα.

Σενάριο 1: Η εφαρμογή Laravel στέλνει δεδομένα στο API της Python

```
use Illuminate\Support\Facades\Http;


$response = Http::post('http://python-api-server/api/data', [
    'key' => 'value',
]);


$data = $response->json();
```

Σενάριο 2: Η εφαρμογή Laravel λαμβάνει δεδομένα από το API της Python

```
use Illuminate\Support\Facades\Http;


$response = Http::get('http://python-api-server/api/data');
$data = $response->json();
```

Η Python είναι γνωστή για την απλότητά της, ενώ το Laravel για την ισχύ και την ευελιξία του. Χρησιμοποιώντας και τα δύο, μπορούμε να εκμεταλλευτούμε τα πλεονεκτήματα κάθε πλατφόρμας. Η χρήση HTTP αιτημάτων επιτρέπει την εύκολη διασύνδεση μεταξύ εφαρμογών που έχουν αναπτυχθεί σε διαφορετικές γλώσσες προγραμματισμού. Τα APIs επιτρέπουν την ανάπτυξη ανεξάρτητων υπηρεσιών, διευκολύνοντας την κλιμάκωση και την ενημέρωση των εφαρμογών.

Η επικοινωνία μέσω HTTP αιτημάτων μπορεί να προσθέσει καθυστερήσεις, ειδικά αν τα αιτήματα είναι μεγάλα ή συχνά. Η διαχείριση δύο διαφορετικών τεχνολογιών μπορεί να προσθέσει πολυπλοκότητα στην ανάπτυξη και τη συντήρηση του συστήματος.

Η Python και το Laravel είναι πολύ δημοφιλή στις κοινότητες των προγραμματιστών, κυρίως λόγω της ευκολίας χρήσης, της ισχύος και της ευελιξίας τους. Η Python είναι ιδιαίτερα αγαπητή για την ανάλυση δεδομένων, την τεχνητή νοημοσύνη και την αυτοματοποίηση, ενώ το Laravel είναι γνωστό για την ανάπτυξη web εφαρμογών και APIs. Η συνδυασμένη χρήση τους επιτρέπει στους προγραμματιστές να εκμεταλλευτούν τα καλύτερα χαρακτηριστικά και των δύο πλατφορμών, δημιουργώντας ισχυρά και ευέλικτα συστήματα.

Η ανάπτυξη APIs με την Python και η επικοινωνία τους με εφαρμογές Laravel προσφέρει μια ισχυρή και ευέλικτη λύση για τη δημιουργία σύγχρονων και αποδοτικών web εφαρμογών. Παρά τις προκλήσεις που μπορεί να παρουσιαστούν, τα πλεονεκτήματα της χρήσης αυτών των δύο ισχυρών

τεχνολογιών υπερτερούν, παρέχοντας στους προγραμματιστές τα εργαλεία που χρειάζονται για να αναπτύξουν αποδοτικά και κλιμακούμενα συστήματα.

3.4 MySQL

Η MySQL είναι μία από τις πιο δημοφιλείς και ευρέως χρησιμοποιούμενες σχεσιακές βάσεις δεδομένων στον κόσμο. Αναπτύχθηκε αρχικά από τον Michael Widenius και τον David Axmark το 1995 και αργότερα αγοράστηκε από την Oracle Corporation το 2010 [27]. Η MySQL έχει γίνει η βάση δεδομένων επιλογής για πολλές εφαρμογές web λόγω της απόδοσης, της αξιοπιστίας και της ευκολίας χρήσης της [28]. Στο πλαίσιο της ανάπτυξης web εφαρμογών, το Laravel είναι ένα από τα πιο δημοφιλή PHP frameworks και προσφέρει ενσωματωμένη υποστήριξη για τη MySQL, καθιστώντας τη διαδικασία ενσωμάτωσης και διαχείρισης βάσεων δεδομένων απλή και αποδοτική. Η MySQL ξεκίνησε την ανάπτυξή της στις αρχές της δεκαετίας του 1990 και κυκλοφόρησε για πρώτη φορά το 1995. Αρχικά σχεδιάστηκε για να είναι μια ελαφριά και γρήγορη βάση δεδομένων, κατάλληλη για μικρές έως μεσαίες εφαρμογές. Κατά τη διάρκεια των ετών, η MySQL έχει εξελιχθεί σε μια ισχυρή πλατφόρμα που υποστηρίζει μεγάλες και πολύπλοκες εφαρμογές. Η εξαγορά της από την Oracle Corporation το 2010 έφερε περαιτέρω ανάπτυξη και ενίσχυση της MySQL, ενώ ταυτόχρονα προέκυψαν και διακλαδώσεις (forks) όπως το MariaDB, που συνεχίζει την ανοιχτού κώδικα παράδοση της MySQL [29].

Η ενσωμάτωση της MySQL με το Laravel είναι απλή και εύκολη χάρη στις ενσωματωμένες λειτουργίες του Laravel για τη διαχείριση βάσεων δεδομένων. Το Laravel χρησιμοποιεί το Eloquent ORM (Object-Relational Mapping) για να διευκολύνει τη διαχείριση των βάσεων δεδομένων και την εκτέλεση των ερωτημάτων.

Στο αρχείο .env του Laravel, διαμορφώστε τις ρυθμίσεις της βάσης δεδομένων:

DB_CONNECTION=mysql

DB_HOST=127.0.0.1

DB_PORT=3306

DB_DATABASE=your_database_name

DB_USERNAME=your_database_user

DB_PASSWORD=your_database_password

Η MySQL προτιμάται για διάφορους λόγους:

- Η MySQL είναι γνωστή για την υψηλή απόδοσή της και τη δυνατότητά της να χειρίζεται μεγάλα φορτία δεδομένων με αποδοτικότητα.
- Προσφέρει αξιόπιστες υπηρεσίες με χαρακτηριστικά όπως η αναπαραγωγή (replication), τα αντίγραφα ασφαλείας (backups) και η αποκατάσταση (recovery).
- Είναι εύκολη στη ρύθμιση και χρήση, με πολλές διαθέσιμες γραφικές διεπαφές (GUIs) και εργαλεία διαχείρισης.
- Διαθέτει μια μεγάλη κοινότητα χρηστών και προγραμματιστών που παρέχει υποστήριξη, τεκμηρίωση και οδηγούς.

Κάποια από τα πλεονεκτήματα είναι ότι χρησιμοποιείται ευρέως σε πολλές εφαρμογές και πλατφόρμες, από μικρές ιστοσελίδες μέχρι μεγάλες επιχειρηματικές εφαρμογές. Είναι συμβατή με πολλαπλές πλατφόρμες και γλώσσες προγραμματισμού. Προσφέρει ισχυρά χαρακτηριστικά ασφαλείας, όπως έλεγχος ταυτότητας χρηστών και κρυπτογράφηση δεδομένων. Μπορεί να κλιμακωθεί εύκολα για να υποστηρίξει αυξανόμενα φορτία και μεγάλες βάσεις δεδομένων.

Τα μειονεκτήματα της εστιάζονται στην περιορισμένη υποστήριξη για πολύπλοκα Ερωτήματα. Αν και η MySQL είναι ισχυρή, μπορεί να μην είναι τόσο αποδοτική όσο άλλες βάσεις δεδομένων στη διαχείριση πολύπλοκων ερωτημάτων και αναλύσεων. Η MySQL δεν υποστηρίζει πλήρως όλα τα χαρακτηριστικά της SQL όπως κάποιες άλλες βάσεις δεδομένων (π.χ., PostgreSQL).

Η MySQL είναι μια από τις πιο δημοφιλείς βάσεις δεδομένων παγκοσμίως και χρησιμοποιείται από πολλές γνωστές εταιρείες και πλατφόρμες, όπως το Facebook, το Twitter, το YouTube και το Airbnb. Η δημοτικότητά της οφείλεται στην υψηλή απόδοση, την ευκολία χρήσης, την αξιοπιστία και την υποστήριξη από την κοινότητα. Χρησιμοποιείται ευρέως σε διάφορους τομείς, από μικρές επιχειρήσεις και startups μέχρι μεγάλες επιχειρήσεις και οργανισμούς.

Η MySQL είναι μια ισχυρή και ευέλικτη βάση δεδομένων που προσφέρει πολλές δυνατότητες και πλεονεκτήματα για την ανάπτυξη web εφαρμογών. Η ενσωμάτωσή της με το Laravel είναι απλή και αποτελεσματική, καθιστώντας την ιδανική επιλογή για προγραμματιστές που επιθυμούν να δημιουργήσουν αποδοτικές και κλιμακούμενες εφαρμογές. Παρά τα μειονεκτήματά της, τα πλεονεκτήματα της MySQL την καθιστούν μια από τις κορυφαίες επιλογές για τη διαχείριση δεδομένων σε διάφορες εφαρμογές και τομείς.

Κεφάλαιο 4ο: Το σύστημα IoT

4.1 Εισαγωγή – Η διαδικασία

Το σύστημα IoT που έχουμε σχεδιάσει και αναπτύξει είναι μια ολοκληρωμένη λύση για την καταγραφή, διαχείριση και ανάλυση δεδομένων από συσκευές IoT, χρησιμοποιώντας το Laravel framework. Στο επίκεντρο του συστήματος βρίσκονται οι χρήστες, οι οποίοι μπορούν να εγγραφούν και να συνδεθούν για να δημιουργήσουν και να διαχειριστούν κανάλια δεδομένων. Κάθε κανάλι αντιπροσωπεύει μια συλλογή από δεδομένα αισθητήρων και περιλαμβάνει πολλαπλά πεδία που μπορούν να ενεργοποιηθούν ή να απενεργοποιηθούν κατά βούληση. Τα δεδομένα από τις συσκευές IoT, όπως θερμοκρασία, υγρασία και άλλα, αποστέλλονται στο σύστημα μέσω ασφαλών API endpoints χρησιμοποιώντας μοναδικά κλειδιά πρόσβασης (API keys). Η διαχείριση των καναλιών γίνεται μέσω ενός φιλικού προς τον χρήστη περιβάλλοντος διαχείρισης, όπου οι χρήστες μπορούν να δημιουργούν, επεξεργάζονται και να διαγράφουν κανάλια, καθώς και να προβάλλουν τα δεδομένα σε δυναμικά γραφήματα (charts) χρησιμοποιώντας τη βιβλιοθήκη Chart.js. Επιπλέον, το σύστημα παρέχει δυνατότητες API για την αποστολή και ανάκτηση δεδομένων σε πραγματικό χρόνο, επιτρέποντας την ενσωμάτωση με εξωτερικές εφαρμογές και πλατφόρμες. Η ασφάλεια και η διαχείριση πρόσβασης είναι επίσης κεντρικά στοιχεία του συστήματος, με αυστηρή επικύρωση δεδομένων και χρήση των PHP Sessions για τη διατήρηση των συνδέσεων των χρηστών. Το αποτέλεσμα είναι ένα ισχυρό και ευέλικτο σύστημα που απλοποιεί τη διαδικασία συλλογής και ανάλυσης δεδομένων IoT, κάνοντάς το ιδανικό για εφαρμογές σε διάφορους τομείς όπως οικιακή αυτοματοποίηση, παρακολούθηση περιβάλλοντος και βιομηχανική συντήρηση.

Το σύστημα αυτό επιτρέπει στους χρήστες να επικοινωνούν με συσκευές IoT (όπως ESP32 ή Raspberry Pi) μέσω διαδικτύου, καταγράφοντας και διαχειριζόμενοι δεδομένα από αισθητήρες σε πραγματικό χρόνο.

Χρήστες και Αυθεντικοποίηση

Οι χρήστες μπορούν να εγγραφούν, να συνδεθούν και να αποσυνδεθούν από το σύστημα.

Η αυθεντικοποίηση βασίζεται σε χειροκίνητο έλεγχο των διαπιστευτηρίων με χρήση των PHP Sessions για την αποθήκευση των στοιχείων του συνδεδεμένου χρήστη.

Διαχείριση Καναλιών

Οι χρήστες μπορούν να δημιουργούν, επεξεργάζονται και διαγράφουν κανάλια που αντιπροσωπεύουν ομάδες δεδομένων από αισθητήρες.

Κάθε κανάλι έχει πεδία που μπορούν να ενεργοποιηθούν ή να απενεργοποιηθούν σύμφωνα με τις ανάγκες του χρήστη.

Καταγραφή Δεδομένων

Οι συσκευές IoT στέλνουν δεδομένα στους κατάλληλους πίνακες της βάσης δεδομένων μέσω HTTP GET requests χρησιμοποιώντας API keys.

Το σύστημα επεξεργάζεται τα εισερχόμενα δεδομένα και τα αποθηκεύει στην βάση δεδομένων στον πίνακα fieldvalue.

Προβολή Δεδομένων

Οι χρήστες μπορούν να προβάλουν τα δεδομένα των καναλιών τους σε μορφή γραφημάτων (charts) χρησιμοποιώντας τη βιβλιοθήκη Chart.js.

Τα γραφήματα ενημερώνονται δυναμικά για να δείχνουν τις πιο πρόσφατες εγγραφές δεδομένων.

API Δυνατότητες

Write API: Οι συσκευές IoT μπορούν να στείλουν δεδομένα χρησιμοποιώντας ένα μοναδικό Write API key.

Read API: Οι χρήστες και άλλες εφαρμογές μπορούν να διαβάσουν δεδομένα από τα κανάλια χρησιμοποιώντας ένα μοναδικό Read API key.

Τεχνική Αρχιτεκτονική

Laravel Framework: Το σύστημα είναι κατασκευασμένο με το Laravel, ένα PHP framework που παρέχει ισχυρά εργαλεία για την ανάπτυξη web εφαρμογών.

Βάση Δεδομένων: Χρησιμοποιείται MySQL για την αποθήκευση χρηστών, καναλιών και δεδομένων αισθητήρων.

Sessions: Χρησιμοποιούνται PHP Sessions για τη διαχείριση των συνδεδεμένων χρηστών.

Blade Templates: Χρησιμοποιούνται για την απόδοση των σελίδων με δυναμικό περιεχόμενο.

DataTables: Χρησιμοποιείται για την εμφάνιση των καναλιών σε μορφή πίνακα με δυνατότητες φιλτραρίσματος και ταξινόμησης.

Chart.js: Χρησιμοποιείται για την εμφάνιση δεδομένων αισθητήρων σε μορφή γραφημάτων.

API: Παρέχονται endpoints για την καταγραφή και ανάκτηση δεδομένων από τις συσκευές IoT.

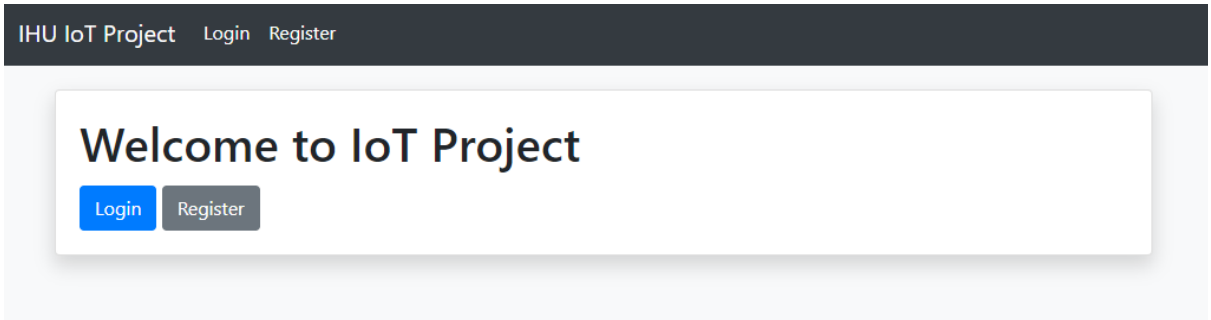
4.2 Η εφαρμογή - ιστοσελίδα

Το σύστημα IoT που έχουμε σχεδιάσει και αναπτύξει είναι μια ολοκληρωμένη λύση για την καταγραφή, διαχείριση και ανάλυση δεδομένων από συσκευές IoT, χρησιμοποιώντας το Laravel framework. Στο επίκεντρο του συστήματος βρίσκονται οι χρήστες, οι οποίοι μπορούν να εγγραφούν και να συνδεθούν για να δημιουργήσουν και να διαχειριστούν κανάλια δεδομένων. Κάθε κανάλι αντιπροσωπεύει μια συλλογή από δεδομένα αισθητήρων και περιλαμβάνει πολλαπλά πεδία που μπορούν να ενεργοποιηθούν ή να απενεργοποιηθούν κατά βούληση.

Τα δεδομένα από τις συσκευές IoT, όπως θερμοκρασία, υγρασία και άλλα, αποστέλλονται στο σύστημα μέσω ασφαλών API endpoints χρησιμοποιώντας μοναδικά κλειδιά πρόσβασης (API keys). Η διαχείριση των καναλιών γίνεται μέσω ενός φιλικού προς τον χρήστη περιβάλλοντος διαχείρισης, όπου οι χρήστες μπορούν να δημιουργούν, επεξεργάζονται και να διαγράφουν κανάλια, καθώς και να προβάλλουν τα δεδομένα σε δυναμικά γραφήματα (charts) χρησιμοποιώντας τη βιβλιοθήκη Chart.js. Επιπλέον, το σύστημα παρέχει δυνατότητες API για την αποστολή και ανάκτηση δεδομένων σε πραγματικό χρόνο, επιτρέποντας την ενσωμάτωση με εξωτερικές εφαρμογές και πλατφόρμες.

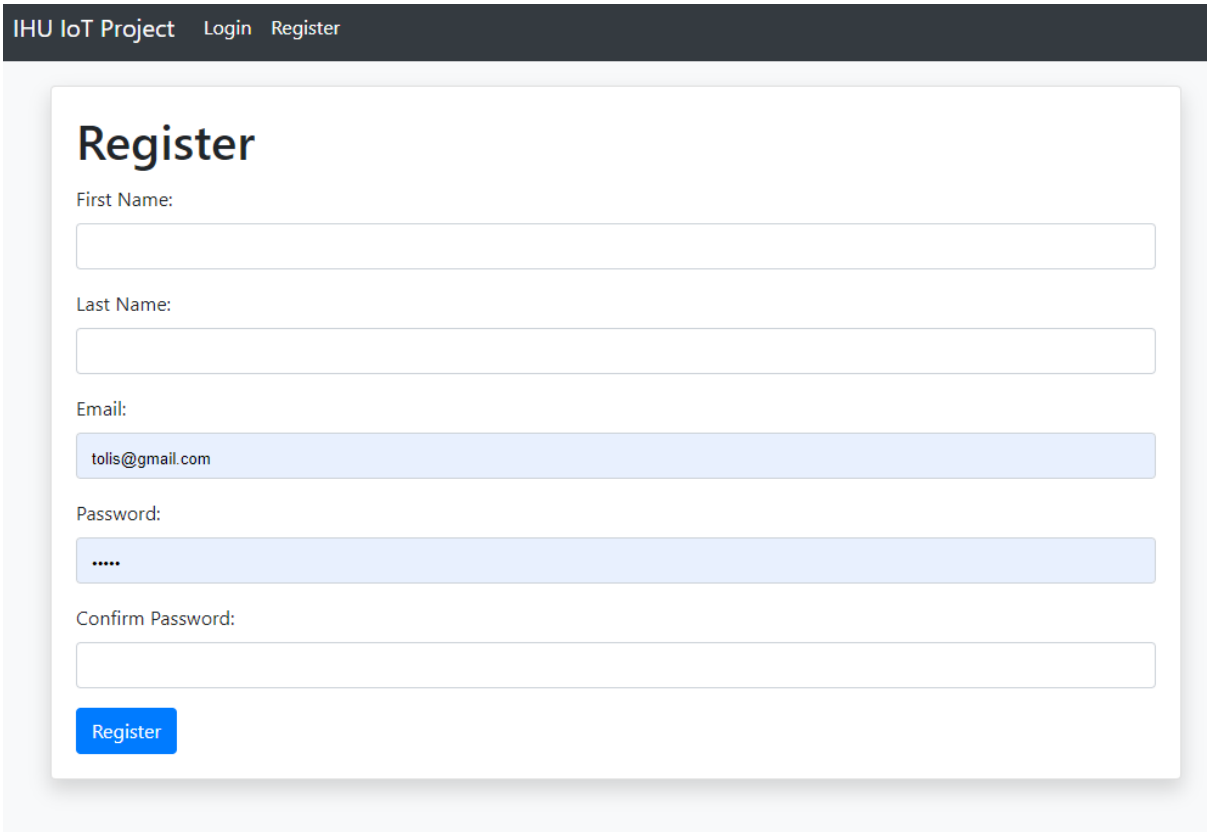
Η ασφάλεια και η διαχείριση πρόσβασης είναι επίσης κεντρικά στοιχεία του συστήματος, με αυστηρή επικύρωση δεδομένων και χρήση των PHP Sessions για τη διατήρηση των συνδέσεων των χρηστών. Το αποτέλεσμα είναι ένα ισχυρό και ευέλικτο σύστημα που απλοποιεί τη διαδικασία συλλογής και ανάλυσης δεδομένων IoT, κάνοντάς το ιδανικό για εφαρμογές σε διάφορους τομείς όπως οικιακή αυτοματοποίηση, παρακολούθηση περιβάλλοντος και βιομηχανική συντήρηση.

Στην αρχική σελίδα της εφαρμογής IoT, οι μη συνδεδεμένοι χρήστες βλέπουν τις επιλογές Login και Register. Αυτή η σελίδα προσφέρει μια φιλική προς τον χρήστη πύλη εισόδου στο σύστημα, επιτρέποντας στους νέους χρήστες να εγγραφούν και στους υπάρχοντες χρήστες να συνδεθούν για να αποκτήσουν πρόσβαση στα κανάλια και τα δεδομένα τους.



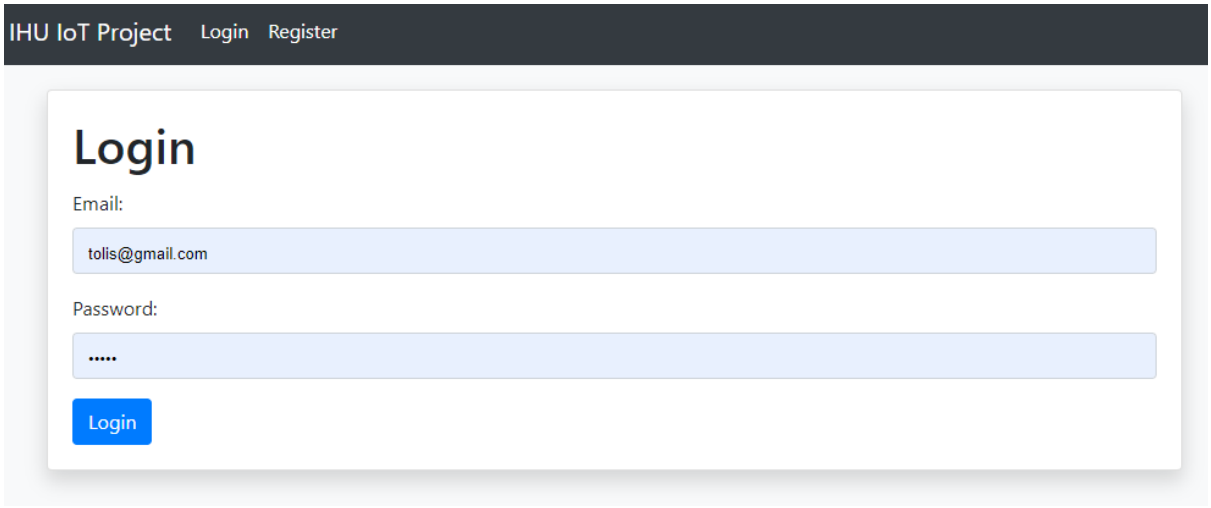
Εικόνα 4.1: Η σελίδα που βλέπει ο μη συνδεδεμένος χρήστης. Έχει διαθέσιμα το Login και Register.

Η σελίδα εγγραφής επιτρέπει στους νέους χρήστες να δημιουργήσουν έναν λογαριασμό στο σύστημα IoT. Οι χρήστες συμπληρώνουν τα πεδία με το όνομά τους, το επώνυμο, το email και τον κωδικό πρόσβασης. Αυτή η διαδικασία είναι απλή και ασφαλής, εξασφαλίζοντας την ακεραιότητα των στοιχείων των χρηστών.

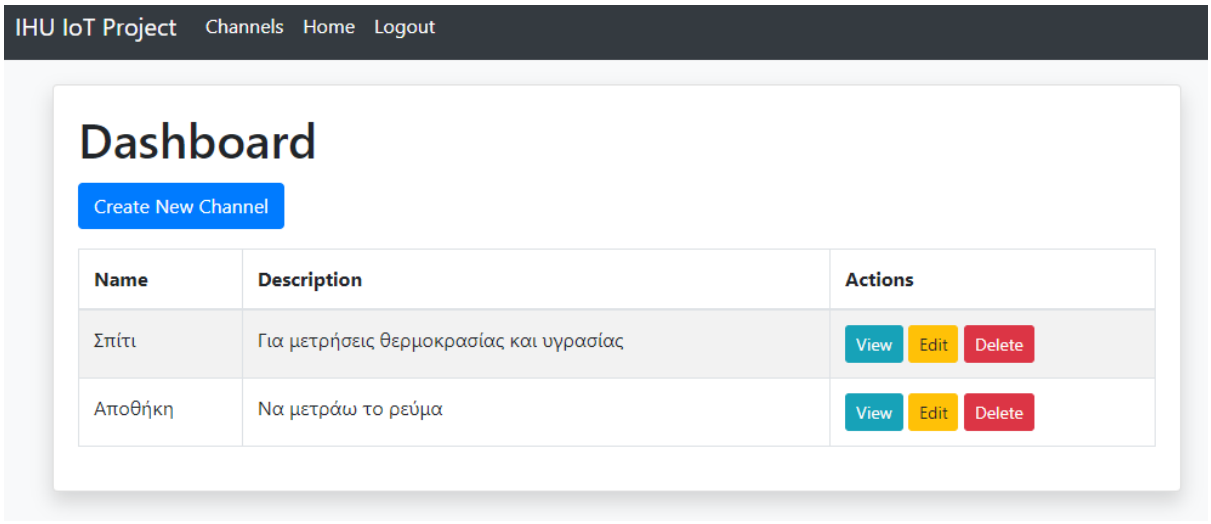


Εικόνα 4.2: Η σελίδα Register-Εγγραφή για νέο χρήστη

Η σελίδα σύνδεσης επιτρέπει στους εγγεγραμμένους χρήστες να εισέλθουν στο σύστημα IoT χρησιμοποιώντας το email και τον κωδικό πρόσβασής τους. Μετά την επιτυχή σύνδεση, οι χρήστες μεταφέρονται στο dashboard τους, όπου μπορούν να διαχειριστούν τα κανάλια και τα δεδομένα τους.

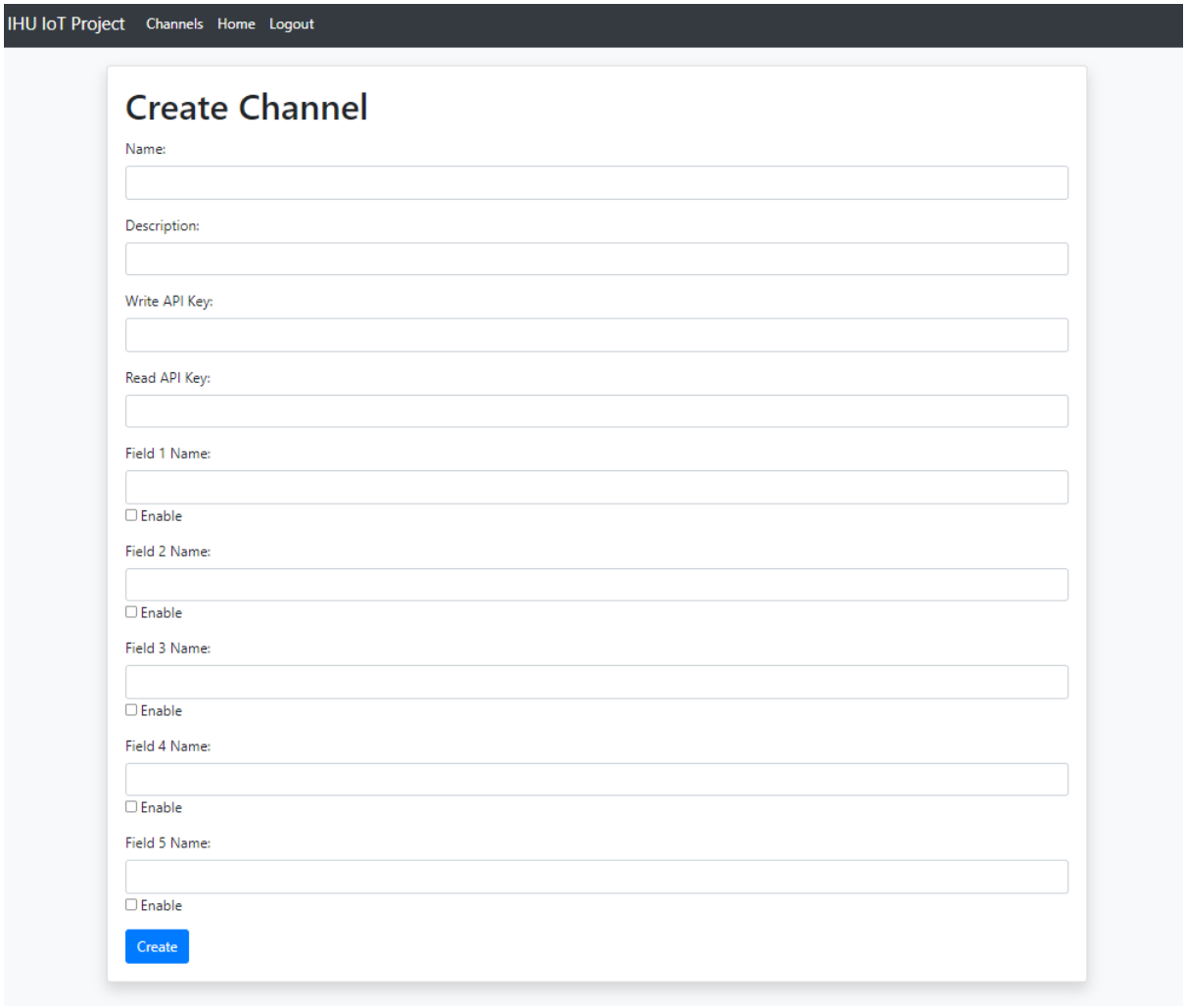


Εικόνα 4.3: Η σελίδα Login-Σύνδεση για έναν εγγεγραμμένο χρήστη



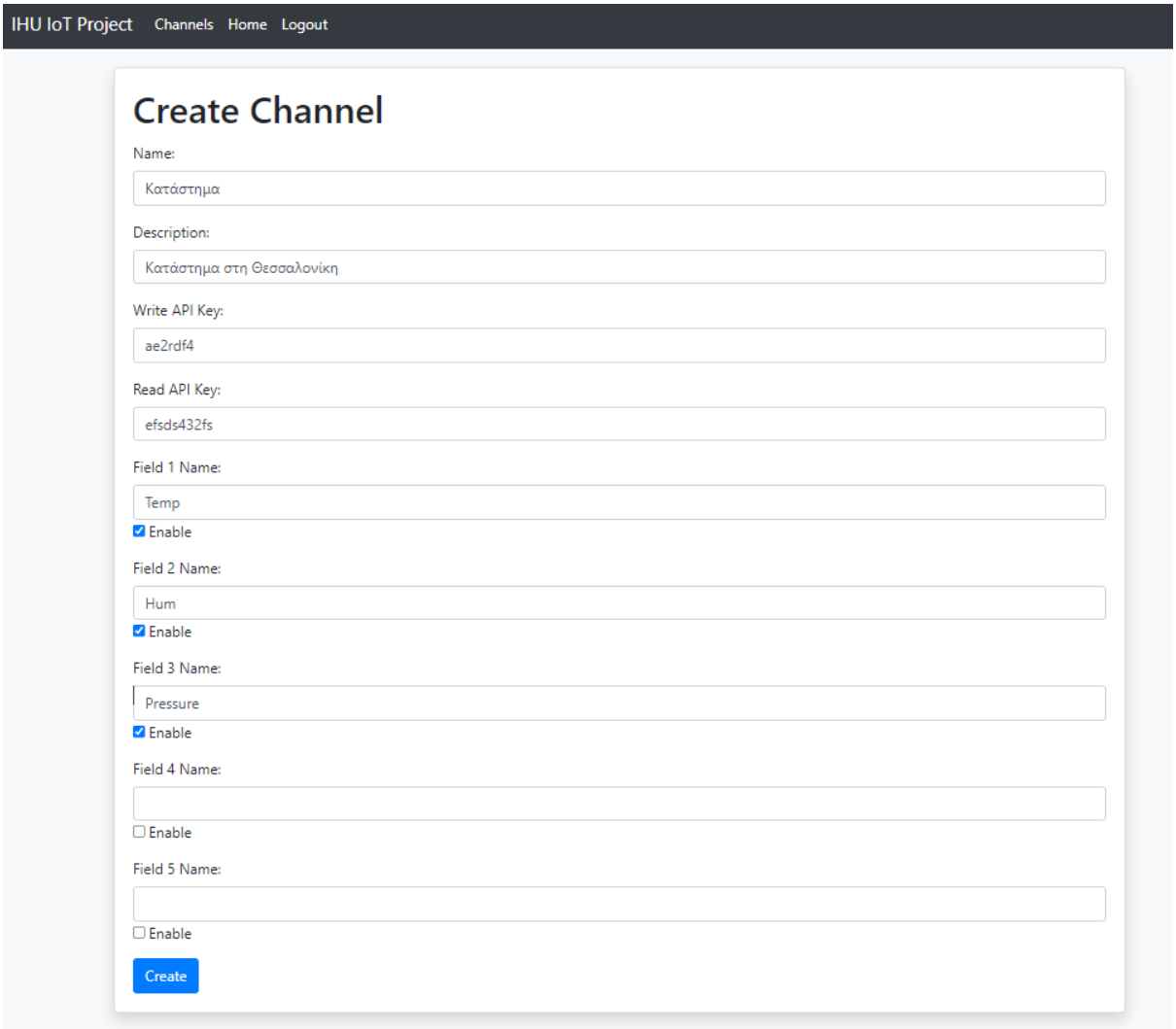
Εικόνα 4.4: Dashboard. Η αρχική σελίδα που βλέπει ο συνδεδεμένος χρήστης. Βλέπει όλα τα κανάλια του με επιλογές προβολής, επεξεργασίας ή διαγραφής

Το dashboard είναι η κεντρική σελίδα για τους συνδεδεμένους χρήστες, όπου εμφανίζονται όλα τα κανάλια δεδομένων τους. Οι χρήστες έχουν τη δυνατότητα να προβάλουν, να επεξεργάζονται ή να διαγράφουν τα κανάλια τους, καθώς και να δημιουργούν νέα κανάλια μέσω μιας απλής και διαισθητικής διεπαφής.



Εικόνα 4.5: Σελίδα για δημιουργία καναλιού – άδεια πεδία

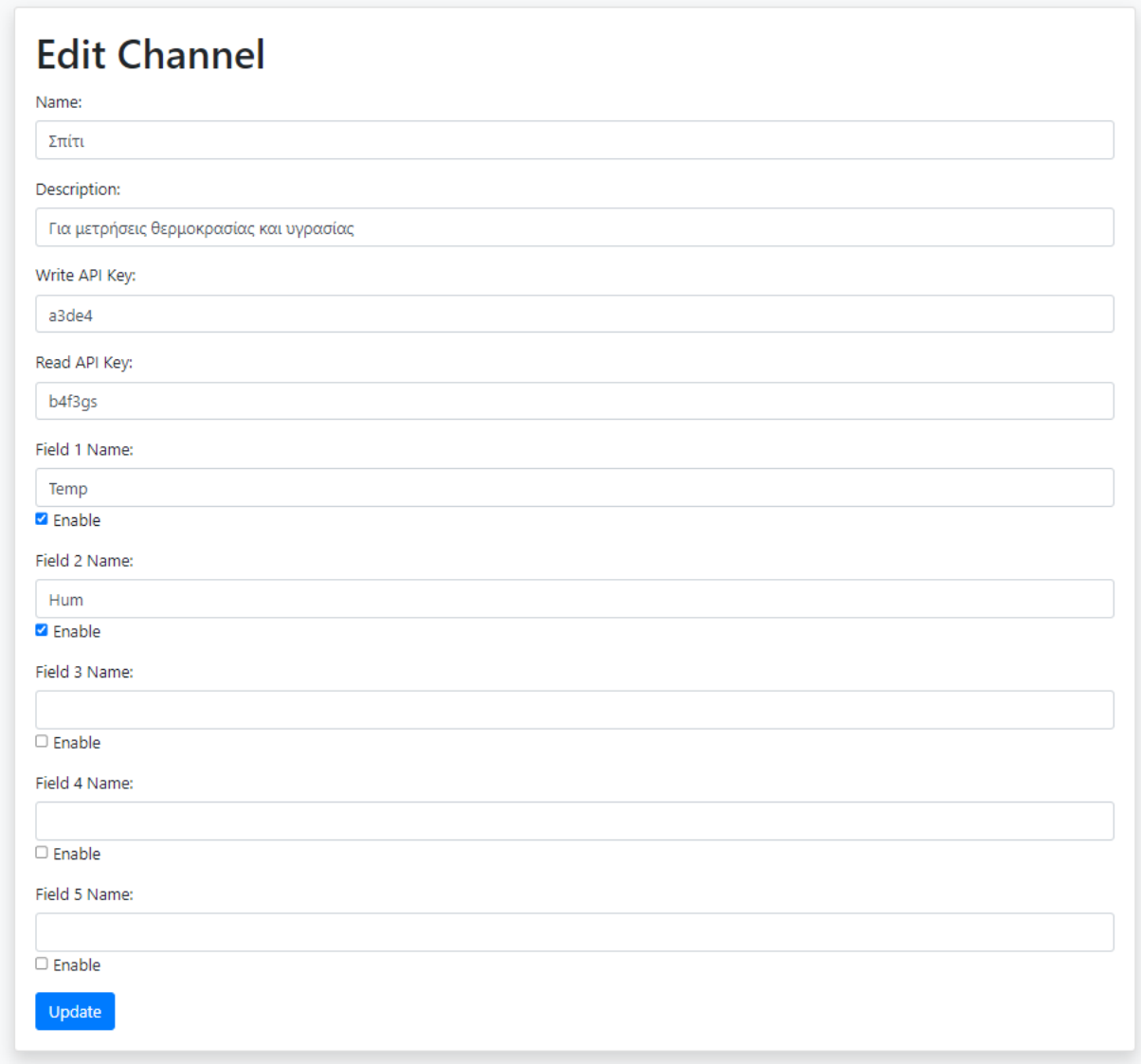
Η σελίδα δημιουργίας καναλιού επιτρέπει στους χρήστες να προσθέτουν νέα κανάλια δεδομένων. Αρχικά, τα πεδία είναι άδεια και οι χρήστες μπορούν να εισάγουν το όνομα του καναλιού, την περιγραφή και τις παραμέτρους των πεδίων που θέλουν να ενεργοποιήσουν.



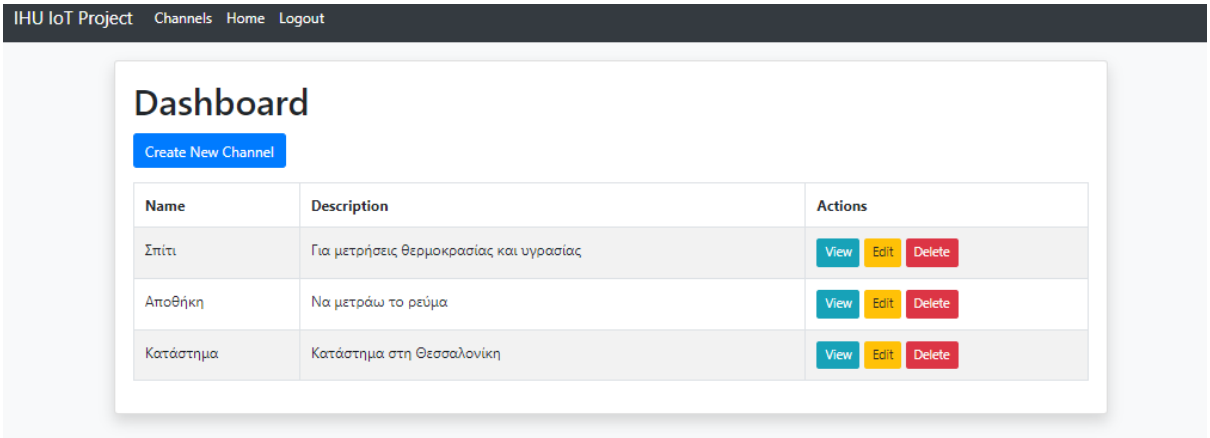
Εικόνα 4.6: Σελίδα για δημιουργία καναλιού – με συμπληρωμένα πεδία

Η ίδια σελίδα δημιουργίας καναλιού εμφανίζεται με συμπληρωμένα πεδία, επιδεικνύοντας πώς οι χρήστες μπορούν να εισάγουν συγκεκριμένες λεπτομέρειες για τα νέα τους κανάλια. Αυτή η λειτουργικότητα διευκολύνει τη ρύθμιση και την προσαρμογή των καναλιών ανάλογα με τις ανάγκες των χρηστών.

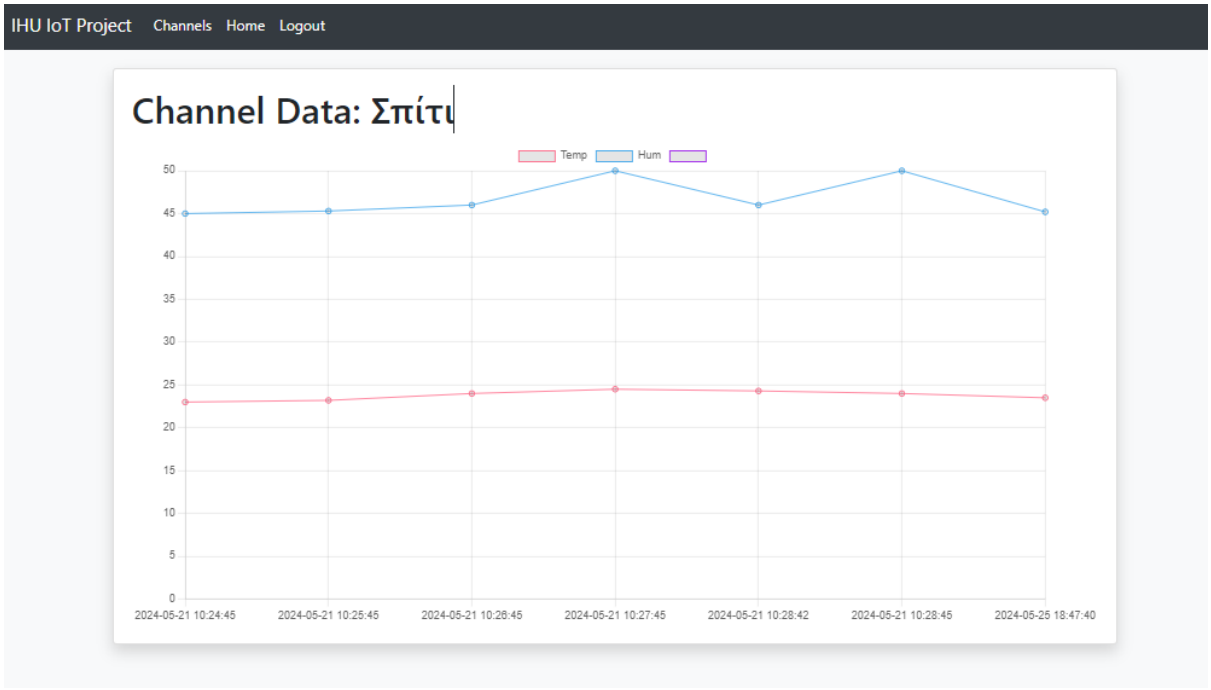
Η σελίδα επεξεργασίας καναλιού επιτρέπει στους χρήστες να τροποποιούν τα χαρακτηριστικά ενός υπάρχοντος καναλιού. Οι χρήστες μπορούν να αλλάζουν το όνομα, την περιγραφή, τα πεδία δεδομένων και τις παραμέτρους των πεδίων, διασφαλίζοντας ότι τα κανάλια τους είναι πάντα ενημερωμένα και προσαρμοσμένα στις ανάγκες τους.



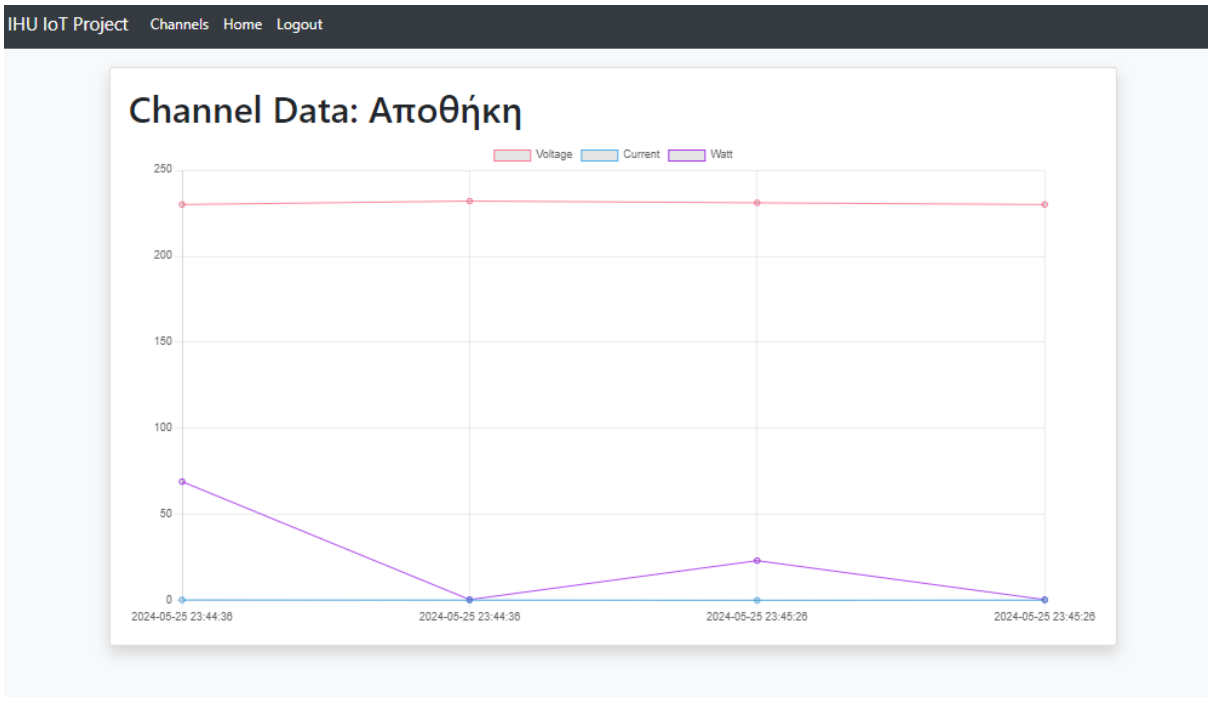
Εικόνα 4.7: Σελίδα για επεξεργασία των χαρακτηριστικών ενός καναλιού



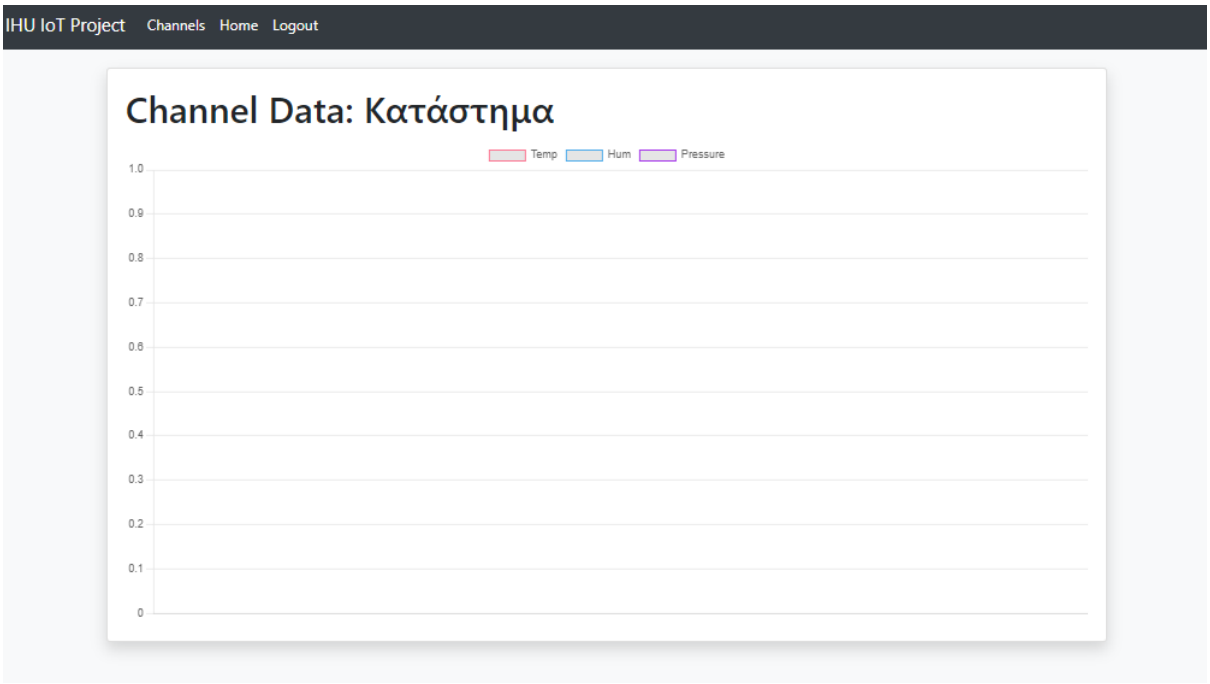
Εικόνα 4.8: Επιλογή προβολής καναλιών



Εικόνα 4.9: Το chart για το κανάλι Σπίτι – δύο πεδία temp,hum



Εικόνα 4.10: Το chart για το κανάλι Αποθήκη – τρία πεδία Voltage,Current,Watt



Εικόνα 4.11: Το chart για το κανάλι Κατάστημα – τρία πεδία temp,hum,pressure

Η επιλογή προβολής καναλιών επιτρέπει στους χρήστες να βλέπουν τα δεδομένα των καναλιών τους με τη μορφή γραφημάτων (charts). Αυτή η λειτουργία προσφέρει μια γραφική απεικόνιση των δεδομένων που συλλέγονται από τους αισθητήρες, διευκολύνοντας την ανάλυση και την παρακολούθηση των τάσεων.

Το γράφημα για το κανάλι "Σπίτι" εμφανίζει δεδομένα για δύο πεδία, τη θερμοκρασία (temp) και την υγρασία (hum). Αυτή η οπτική αναπαράσταση επιτρέπει στους χρήστες να βλέπουν τις μεταβολές της θερμοκρασίας και της υγρασίας σε πραγματικό χρόνο, παρέχοντας πολύτιμες πληροφορίες για το περιβάλλον του σπιτιού τους.

Το γράφημα για το κανάλι "Αποθήκη" εμφανίζει δεδομένα για τρία πεδία: την τάση (Voltage), το ρεύμα (Current) και την ισχύ (Watt). Αυτή η οπτική αναπαράσταση βοηθά τους χρήστες να παρακολουθούν την κατανάλωση ενέργειας στην αποθήκη τους, επιτρέποντας τη λήψη ενημερωμένων αποφάσεων για την εξοικονόμηση ενέργειας.

Το γράφημα για το κανάλι "Κατάστημα" εμφανίζει δεδομένα για τρία πεδία: τη θερμοκρασία (temp), την υγρασία (hum) και την πίεση (pressure). Αυτή η οπτική αναπαράσταση παρέχει στους χρήστες πληροφορίες για τις περιβαλλοντικές συνθήκες στο κατάστημά τους, βοηθώντας τους να διατηρούν ιδανικές συνθήκες για τους πελάτες και τα προϊόντα τους.

4.2.1 API

4.2.1.1 writeiot.py

Αυτό το script Python επιτρέπει την αποστολή δεδομένων από αισθητήρες σε ένα API του IoT συστήματος μέσω ενός HTTP GET αιτήματος. Τα δεδομένα περιλαμβάνουν έως πέντε τιμές πεδίων και ένα API κλειδί για την εξουσιοδότηση της αποστολής. Το script χρησιμοποιεί τη βιβλιοθήκη requests για να στείλει το αίτημα και να χειριστεί την απόκριση.

Για αρχή προσχωρούμε σε εγκατάσταση της βιβλιοθήκης requests:

Εάν δεν έχετε ήδη εγκαταστήσει τη βιβλιοθήκη requests, μπορείτε να την εγκαταστήσετε χρησιμοποιώντας την ακόλουθη εντολή:

```
pip install requests
```

Εκτέλεση του script:

Μπορείτε να εκτελέσετε το script εισάγοντας τον κώδικα σε ένα αρχείο Python και τρέχοντας το αρχείο. Το script περιέχει μια παράδειγμα κλήσης στη συνάρτηση send_data για να στείλει δεδομένα στο API.

```
import requests


def send_data(api_key, field1, field2=None, field3=None, field4=None, field5=None):
    """
    Αποστέλλει δεδομένα αισθητήρων στο IoT API.

    :param api_key: Το API κλειδί για εξουσιοδότηση.
    :param field1: Τιμή για το πεδίο 1 (υποχρεωτικό).
    :param field2: Τιμή για το πεδίο 2 (προαιρετικό).
    :param field3: Τιμή για το πεδίο 3 (προαιρετικό).
```

```
    :param field4: Τιμή για το πεδίο 4 (προαιρετικό).
    :param field5: Τιμή για το πεδίο 5 (προαιρετικό).
    """
    url = "http://localhost/iot/apiupdate"  # Η διεύθυνση URL του API endpoint
    payload = {
        'apikeywrite': api_key,
        'field1': field1,
        'field2': field2,
        'field3': field3,
        'field4': field4,
        'field5': field5
    }

    # Αποστολή του αιτήματος GET στο API
    response = requests.get(url, params=payload)

    # Έλεγχος της κατάστασης της απόκρισης
    if response.status_code == 200:
        print("Data sent successfully")
    else:
        print("Failed to send data")
        print(response.content)
        # print(response.json())  # Μπορεί να χρησιμοποιηθεί αν η απόκριση είναι σε μορφή JSON

# Παράδειγμα κλήσης της συνάρτησης send_data
send_data("a3de4", 23.5, 45.2, 0, 0, 0)
```

Βεβαιωθείτε ότι ο διακομιστής σας είναι σε λειτουργία και ότι το API endpoint http://localhost/iot/apiupdate είναι προσβάσιμο.

Αντικαταστήστε το api_key με το σωστό API κλειδί που έχετε για το σύστημα IoT σας.

Αποθηκεύστε τον κώδικα.

Τρέξτε το αρχείο χρησιμοποιώντας την εντολή:

python writeiot.py

Η συνάρτηση send_data θα στείλει τα δεδομένα στο API και θα εκτυπώσει μήνυμα επιτυχίας ή αποτυχίας με βάση την απόκριση του διακομιστή.

4.2.1.2 readiot.py

Αυτό το script Python επιτρέπει την ανάκτηση δεδομένων από ένα API του IoT συστήματος μέσω ενός HTTP GET αιτήματος. Το script αποστέλλει ένα αίτημα με το API κλειδί ανάγνωσης και τον αριθμό των αποτελεσμάτων που θέλετε να ανακτήσετε. Η βιβλιοθήκη requests χρησιμοποιείται για την αποστολή του αιτήματος και τη διαχείριση της απόκρισης.

Μπορείτε να εκτελέσετε το script εισάγοντας τον κώδικα σε ένα αρχείο Python και τρέχοντας το αρχείο. Το script περιέχει μια παράδειγμα κλήσης στη συνάρτηση read_data για να ανακτήσει δεδομένα από το API.

```
import requests


def read_data(api_key, results=10):
    """
    Ανακτά δεδομένα από το IoT API.


    :param api_key: Το API κλειδί για εξουσιοδότηση.
    :param results: Ο αριθμός των αποτελεσμάτων που θέλετε να ανακτήσετε (προαιρετικό, προεπιλεγμένο 10).
    """
    url = "http://localhost/iot/apiread"  # Η διεύθυνση URL του API endpoint
    payload = {
```

```
        'apikeyread': api_key,
        'results': results
    }

    # Αποστολή του αιτήματος GET στο API
    response = requests.get(url, params=payload)

    # Έλεγχος της κατάστασης της απόκρισης
    if response.status_code == 200:
        data = response.json()
        print("Data retrieved successfully")
        for entry in data:
            print(entry)
    else:
        print("Failed to retrieve data")
        # print(response.json())  # Μπορεί να χρησιμοποιηθεί αν η απόκριση είναι σε μορφή JSON

# Παράδειγμα κλήσης της συνάρτησης read_data
read_data("b4f3gs", 5)
```

Βεβαιωθείτε ότι ο διακομιστής σας είναι σε λειτουργία και ότι το API endpoint http://localhost/iot/apiread είναι προσβάσιμο.

Αντικαταστήστε το api_key με το σωστό API κλειδί που έχετε για το σύστημα IoT σας.

Εκτέλεση:

Αποθηκεύστε τον κώδικα σε ένα αρχείο, read_data.py.

Τρέξτε το αρχείο χρησιμοποιώντας την εντολή:

python read_data.py

Η συνάρτηση read_data θα ανακτήσει τα δεδομένα από το API και θα εκτυπώσει μήνυμα επιτυχίας ή αποτυχίας με βάση την απόκριση του διακομιστή. Τα δεδομένα που ανακτώνται θα εμφανιστούν στη κονσόλα.

Με αυτά τα βήματα, μπορείτε να χρησιμοποιήσετε το script για να ανακτήσετε δεδομένα από το IoT σύστημα χρησιμοποιώντας το API.

4.3 Η Βάση

Η βάση δεδομένων του συστήματος IoT, με όνομα iotdb, είναι σχεδιασμένη για την αποθήκευση και διαχείριση δεδομένων που συλλέγονται από διάφορους αισθητήρες μέσω συσκευών IoT. Η βάση δεδομένων περιλαμβάνει τρεις κύριους πίνακες: user, channel, και fieldvalue. Αυτοί οι πίνακες συνδυαστικά αποθηκεύουν πληροφορίες χρηστών, καναλιών δεδομένων και τιμές πεδίων από τους αισθητήρες.

Πίνακας user

Ο πίνακας user αποθηκεύει πληροφορίες για τους χρήστες του συστήματος. Περιλαμβάνει τα ακόλουθα πεδία:

id: Ο μοναδικός αριθμητικός αναγνωριστικός αριθμός για κάθε χρήστη (πρωτεύον κλειδί).

active: Ένα πεδίο που υποδεικνύει αν ο χρήστης είναι ενεργός (προεπιλεγμένη τιμή 1).

firstname: Το όνομα του χρήστη.

lastname: Το επώνυμο του χρήστη.

email: Η διεύθυνση ηλεκτρονικού ταχυδρομείου του χρήστη.

password: Ο κωδικός πρόσβασης του χρήστη (αποθηκευμένος σε hashed μορφή).

created_at: Η ημερομηνία και ώρα δημιουργίας της εγγραφής.

updated_at: Η ημερομηνία και ώρα της τελευταίας ενημέρωσης της εγγραφής.

```
CREATE TABLE `user` (
 `id` int(10) UNSIGNED NOT NULL,
 `active` tinyint(3) UNSIGNED DEFAULT 1,
 `firstname` varchar(255) NOT NULL,
```

```
  `lastname` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp(),
  `updated_at` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Πίνακας channel

Ο πίνακας channel αποθηκεύει πληροφορίες για τα κανάλια δεδομένων, τα οποία αντιπροσωπεύουν διαφορετικές πηγές δεδομένων αισθητήρων. Περιλαμβάνει τα ακόλουθα πεδία:

id: Ο μοναδικός αριθμητικός αναγνωριστικός αριθμός για κάθε κανάλι (πρωτεύον κλειδί).

active: Ένα πεδίο που υποδεικνύει αν το κανάλι είναι ενεργό (προεπιλεγμένη τιμή 1).

user_id: Ο αναγνωριστικός αριθμός του χρήστη που δημιουργεί το κανάλι.

name: Το όνομα του καναλιού.

description: Μια περιγραφή του καναλιού.

apikeywrite: Το API κλειδί για την εγγραφή δεδομένων στο κανάλι.

apikeyread: Το API κλειδί για την ανάγνωση δεδομένων από το κανάλι.

field1check έως field5check: Πεδία που υποδεικνύουν αν κάθε πεδίο δεδομένων είναι ενεργό.

field1name έως field5name: Ονόματα των πεδίων δεδομένων.

created_at: Η ημερομηνία και ώρα δημιουργίας της εγγραφής.

updated_at: Η ημερομηνία και ώρα της τελευταίας ενημέρωσης της εγγραφής.

```
CREATE TABLE `channel` (
  `id` int(10) UNSIGNED NOT NULL,
  `active` tinyint(3) UNSIGNED DEFAULT 1,
  `user_id` int(10) UNSIGNED NOT NULL,
```

```
  `name` varchar(50) NOT NULL,
  `description` varchar(255) DEFAULT NULL,
  `apikeywrite` varchar(255) NOT NULL,
  `apikeyread` varchar(255) NOT NULL,
  `field1check` tinyint(3) UNSIGNED DEFAULT 0,
  `field1name` varchar(50) DEFAULT NULL,
  `field2check` tinyint(3) UNSIGNED DEFAULT 0,
  `field2name` varchar(50) DEFAULT NULL,
  `field3check` tinyint(3) UNSIGNED DEFAULT 0,
  `field3name` varchar(50) DEFAULT NULL,
  `field4check` tinyint(3) UNSIGNED DEFAULT 0,
  `field4name` varchar(50) DEFAULT NULL,
  `field5check` tinyint(3) UNSIGNED DEFAULT 0,
  `field5name` varchar(50) DEFAULT NULL,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp(),
  `updated_at` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Πίνακας fieldvalue

Ο πίνακας fieldvalue αποθηκεύει τις τιμές των πεδίων που συλλέγονται από τους αισθητήρες για κάθε κανάλι. Περιλαμβάνει τα ακόλουθα πεδία:

id: Ο μοναδικός αριθμητικός αναγνωριστικός αριθμός για κάθε εγγραφή δεδομένων (πρωτεύον κλειδί).

active: Ένα πεδίο που υποδεικνύει αν η εγγραφή είναι ενεργή (προεπιλεγμένη τιμή 1).

channel_id: Ο αναγνωριστικός αριθμός του καναλιού στο οποίο ανήκει η εγγραφή.

field1value έως field5value: Τιμές για τα πεδία δεδομένων.

created_at: Η ημερομηνία και ώρα δημιουργίας της εγγραφής.

```
CREATE TABLE `fieldvalue` (
  `id` int(10) UNSIGNED NOT NULL,
  `active` tinyint(3) UNSIGNED DEFAULT 1,
  `channel_id` int(10) UNSIGNED NOT NULL,
  `field1value` double DEFAULT NULL,
  `field2value` double DEFAULT NULL,
  `field3value` double DEFAULT NULL,
  `field4value` double DEFAULT NULL,
  `field5value` double DEFAULT NULL,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Σχέσεις μεταξύ Πινάκων

Ο πίνακας user συνδέεται με τον πίνακα channel μέσω του πεδίου user_id, υποδεικνύοντας ότι κάθε χρήστης μπορεί να έχει πολλά κανάλια.

Ο πίνακας channel συνδέεται με τον πίνακα fieldvalue μέσω του πεδίου channel_id, υποδεικνύοντας ότι κάθε κανάλι μπορεί να έχει πολλές εγγραφές δεδομένων.

Ευρετήρια και Πρωτεύοντα Κλειδιά

Κάθε πίνακας έχει πρωτεύον κλειδί το πεδίο id, το οποίο είναι αυτόματης αύξησης (AUTO_INCREMENT).

Ο πίνακας channel έχει ευρετήριο στο πεδίο id.

Ο πίνακας fieldvalue έχει ευρετήριο στο πεδίο id.

Ο πίνακας user έχει ευρετήριο στο πεδίο id.

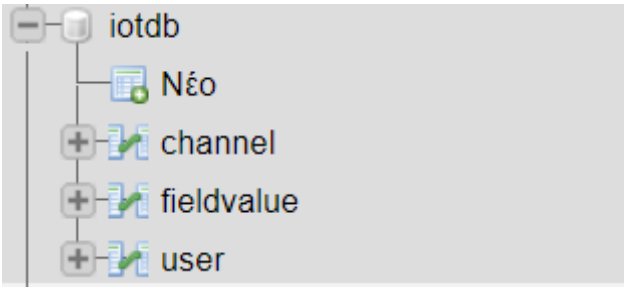
Εισαγωγή Δεδομένων

Τα δεδομένα εισάγονται στους πίνακες με SQL εντολές INSERT. Για παράδειγμα, στον πίνακα channel έχουν εισαχθεί τα εξής δεδομένα:

```
INSERT INTO `channel` (`id`, `active`, `user_id`, `name`, `description`, `apikeywrite`, `apikeyread`, `field1check`, `field1name`, `field2check`, `field2name`, `field3check`, `field3name`, `field4check`, `field4name`, `field5check`, `field5name`, `created_at`, `updated_at`) VALUES

(1, 1, 1, 'Σπίτι', 'Για μετρήσεις θερμοκρασίας και υγρασίας', 'a3de4', 'b4f3gs', 1, 'Temp', 0, 'Hum', 0, NULL, 0, NULL, 0, NULL, '2024-05-21 07:20:57', '2024-05-25 18:07:58'),

(2, 1, 1, 'Αποθήκη', 'Να μετράω το ρεύμα', 'dew342', 'dfsfd34', 1, 'Voltage', 1, 'Current', 1, 'Watt', 0, NULL, 0, NULL, '2024-05-25 16:12:19', '2024-05-25 17:43:21'),

(3, 1, 1, 'Κατάστημα', 'Κατάστημα στη Θεσσαλονίκη', 'ae2rdf4', 'efsds432fs', 1, 'Temp', 1, 'Hum', 1, 'Pressure', 0, NULL, 0, NULL, '2024-05-25 17:54:08', '2024-05-25 17:54:08');
```

Με αυτόν τον τρόπο, η βάση δεδομένων του συστήματος IoT είναι πλήρως οργανωμένη και έτοιμη να υποστηρίξει την αποθήκευση και διαχείριση δεδομένων από αισθητήρες, επιτρέποντας στους χρήστες να δημιουργούν και να διαχειρίζονται κανάλια δεδομένων και να ανακτούν πολύτιμες πληροφορίες από τις συσκευές IoT.

Ας δούμε και τους πίνακες της βάσης σε Εικόνες.



Εικόνα 4.12: Η βάση με τους πίνακες

#	Όνομα	Τύπος	Σύνθεση	Χαρακτηριστικά	Κενό	Προεπιλογή	Σχόλια	Πρόσθετα
1	**id**	int(10)		UNSIGNED	Όχι	*Καμία*		AUTO_INCREMENT
2	**active**	tinyint(3)		UNSIGNED	Ναι	1		
3	**firstname**	varchar(255)	utf8mb4_general_ci		Όχι	*Καμία*		
4	**lastname**	varchar(255)	utf8mb4_general_ci		Όχι	*Καμία*		
5	**email**	varchar(255)	utf8mb4_general_ci		Όχι	*Καμία*		
6	**password**	varchar(255)	utf8mb4_general_ci		Όχι	*Καμία*		
7	**created_at**	timestamp			Όχι	current_timestamp()		
8	**updated_at**	timestamp			Όχι	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()

Εικόνα 4.13: Η δομή του πίνακα

id	active	firstname	lastname	email	password	created_at	updated_at
1	1	Απόστολος	Κωνσταντινίδης	tolis@gmail.com	12345	2024-05-20 12:00:00	2024-05-20 12:00:00
2	1	Πέτρος	Λάμπρου	petros@gmail.com	12345	2024-05-20 12:00:00	2024-05-20 12:00:00
3	1	Ελένη	Παπαπέτρου	eleni@gmail.com	12345	2024-05-20 12:00:00	2024-05-20 12:00:00

Εικόνα 4.14: Το περιεχόμενο του πίνακα

#	Όνομα	Τύπος	Σύνθεση	Χαρακτηριστικά	Κενό	Προεπιλογή	Σχόλια	Πρόσθετα
1	id	int(10)		UNSIGNED	Όχι	*Καμία*		AUTO_INCREMENT
2	active	tinyint(3)		UNSIGNED	Ναι	1		
3	user_id	int(10)		UNSIGNED	Όχι	*Καμία*		
4	name	varchar(50)	utf8mb4_general_ci		Όχι	*Καμία*		
5	description	varchar(255)	utf8mb4_general_ci		Ναι	*NULL*		
6	apikeywrite	varchar(255)	utf8mb4_general_ci		Όχι	*Καμία*		
7	apikeyread	varchar(255)	utf8mb4_general_ci		Όχι	*Καμία*		
8	field1check	tinyint(3)		UNSIGNED	Ναι	0		
9	field1name	varchar(50)	utf8mb4_general_ci		Ναι	*NULL*		
10	field2check	tinyint(3)		UNSIGNED	Ναι	0		
11	field2name	varchar(50)	utf8mb4_general_ci		Ναι	*NULL*		
12	field3check	tinyint(3)		UNSIGNED	Ναι	0		
13	field3name	varchar(50)	utf8mb4_general_ci		Ναι	*NULL*		
14	field4check	tinyint(3)		UNSIGNED	Ναι	0		
15	field4name	varchar(50)	utf8mb4_general_ci		Ναι	*NULL*		
16	field5check	tinyint(3)		UNSIGNED	Ναι	0		
17	field5name	varchar(50)	utf8mb4_general_ci		Ναι	*NULL*		
18	created_at	timestamp			Όχι	current_timestamp()		
19	updated_at	timestamp			Όχι	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()

Εικόνα 4.15: Η δομή του πίνακα channel

id	active	user_id	name	description	apikeywrite	apikeyread	field1check	field1name	field2check	field2name	field3check	field3name	field4check	field4name	field5check	field5name	created_at	updated_at
1	1	1	Σπίτι	Για μετρήσεις θερμοκρασίας και υγρασίας	a3de4	b4f3gs	1	Temp	0	Hum	0	NULL	0	NULL	0	NULL	2024-05-21 10:20:57	2024-05-25 21:07:58
2	1	1	Αποθήκη	Να μετράω το ρεύμα	dew342	dfsfd34	1	Voltage	1	Current	1	Watt	0	NULL	0	NULL	2024-05-25 19:12:19	2024-05-25 20:43:21
3	1	1	Κατάστημα	Κατάστημα στη Θεσσαλονίκη	ae2rdf4	efsds432fs	1	Temp	1	Hum	1	Pressure	0	NULL	0	NULL	2024-05-25 20:54:08	2024-05-25 20:54:08

Εικόνα 4.16: Το περιεχόμενο του πίνακα channel

#	Όνομα	Τύπος	Σύνθεση	Χαρακτηριστικά	Κενό	Προεπιλογή	Σχόλια	Πρόσθετα
1	id	int(10)		UNSIGNED	Όχι	*Καμία*		AUTO_INCREMENT
2	active	tinyint(3)		UNSIGNED	Ναι	1		
3	channel_id	int(10)		UNSIGNED	Όχι	*Καμία*		
4	field1value	double			Ναι	*NULL*		
5	field2value	double			Ναι	*NULL*		
6	field3value	double			Ναι	*NULL*		
7	field4value	double			Ναι	*NULL*		
8	field5value	double			Ναι	*NULL*		
9	created_at	timestamp			Όχι	current_timestamp()		

Εικόνα 4.17: Η δομή του πίνακα fieldvalue

id	active	channel_id	field1value	field2value	field3value	field4value	field5value	created_at
1	1	1	23	45	*NULL*	*NULL*	*NULL*	2024-05-21 10:24:45
2	1	1	23.2	45.3	*NULL*	*NULL*	*NULL*	2024-05-21 10:25:45
3	1	1	24	46	*NULL*	*NULL*	*NULL*	2024-05-21 10:26:45
4	1	1	24.5	50	*NULL*	*NULL*	*NULL*	2024-05-21 10:27:45
5	1	1	24.3	46	*NULL*	*NULL*	*NULL*	2024-05-21 10:28:42
6	1	1	24	50	*NULL*	*NULL*	*NULL*	2024-05-21 10:28:45
7	1	1	23.5	45.2	*NULL*	*NULL*	*NULL*	2024-05-25 18:47:40
8	1	2	230	0.3	69	*NULL*	*NULL*	2024-05-25 23:44:36
9	1	2	232	0.2	0.464	*NULL*	*NULL*	2024-05-25 23:44:36
10	1	2	231	0.1	23.1	*NULL*	*NULL*	2024-05-25 23:45:26
11	1	2	230	0.2	0.46	*NULL*	*NULL*	2024-05-25 23:45:26

Εικόνα 4.18: Το περιεχόμενο του πίνακα fieldvalue

4.4 Ασφάλεια στο σύστημα και στα δεδομένα

Η ασφάλεια αποτελεί έναν από τους πιο κρίσιμους παράγοντες για τη σωστή λειτουργία και αξιοπιστία του συστήματος IoT. Η προστασία των δεδομένων, η διασφάλιση της ιδιωτικότητας των χρηστών και η αποτροπή μη εξουσιοδοτημένης πρόσβασης είναι θεμελιώδεις αρχές που πρέπει να τηρούνται.

1. Ασφάλεια Δεδομένων

Η ασφάλεια των δεδομένων στο σύστημα IoT είναι καίριας σημασίας. Τα δεδομένα που συλλέγονται από τους αισθητήρες και αποθηκεύονται στη βάση δεδομένων περιέχουν πολύτιμες πληροφορίες που πρέπει να προστατεύονται από μη εξουσιοδοτημένη πρόσβαση και αλλοίωση.

Κρυπτογράφηση Δεδομένων: Όλα τα ευαίσθητα δεδομένα, όπως τα API keys και οι κωδικοί πρόσβασης των χρηστών, αποθηκεύονται σε κρυπτογραφημένη μορφή στη βάση δεδομένων. Αυτό διασφαλίζει ότι ακόμη και αν η βάση δεδομένων παραβιαστεί, οι πληροφορίες θα παραμείνουν ασφαλείς.

Μεταφορά Δεδομένων μέσω HTTPS: Όλες οι επικοινωνίες μεταξύ των συσκευών IoT, των χρηστών και της εφαρμογής γίνονται μέσω ασφαλών συνδέσεων HTTPS. Αυτό αποτρέπει την υποκλοπή δεδομένων κατά τη μεταφορά τους μέσω του διαδικτύου.

2. Διαχείριση Πρόσβασης

Η διαχείριση πρόσβασης είναι ζωτικής σημασίας για την προστασία των δεδομένων και την εξασφάλιση ότι μόνο εξουσιοδοτημένοι χρήστες μπορούν να εκτελούν συγκεκριμένες ενέργειες στο σύστημα.

API Keys: Το σύστημα χρησιμοποιεί μοναδικά API keys για την εγγραφή και ανάγνωση δεδομένων από τα κανάλια. Κάθε κανάλι έχει ένα apikeywrite για εγγραφή δεδομένων και ένα apikeyread για ανάγνωση δεδομένων. Τα API keys παρέχουν έναν επιπλέον μηχανισμό ασφαλείας για την πρόσβαση στα δεδομένα.

Ρόλοι και Δικαιώματα: Οι χρήστες του συστήματος έχουν διαφορετικά επίπεδα πρόσβασης ανάλογα με το ρόλο τους. Οι διαχειριστές έχουν πλήρη πρόσβαση για να δημιουργούν, να επεξεργάζονται και να διαγράφουν κανάλια, ενώ οι απλοί χρήστες μπορεί να έχουν περιορισμένη πρόσβαση.

3. Επικύρωση και Αυθεντικοποίηση

Η επαλήθευση της ταυτότητας των χρηστών και η επικύρωση των δεδομένων εισόδου είναι κρίσιμα για την αποτροπή επιθέσεων και τη διασφάλιση της ακεραιότητας του συστήματος.

Επικύρωση Δεδομένων: Κάθε αίτημα που αποστέλλεται στο σύστημα ελέγχεται για να διασφαλιστεί ότι τα δεδομένα εισόδου είναι έγκυρα και εντός των αποδεκτών ορίων. Η επικύρωση των δεδομένων αποτρέπει επιθέσεις SQL injection και άλλες μορφές εκμετάλλευσης των ευπαθειών του συστήματος.

Σύστημα Αυθεντικοποίησης: Οι χρήστες πρέπει να εισάγουν το email και τον κωδικό πρόσβασής τους για να συνδεθούν στο σύστημα. Οι κωδικοί πρόσβασης αποθηκεύονται σε κρυπτογραφημένη μορφή, διασφαλίζοντας ότι ακόμη και αν οι πληροφορίες παραβιαστούν, οι κωδικοί θα παραμείνουν ασφαλείς. Επιπλέον, χρησιμοποιούνται PHP Sessions για τη διατήρηση της σύνδεσης των χρηστών και την προστασία από επιθέσεις τύπου session hijacking.

4. Παρακολούθηση και Καταγραφή

Η παρακολούθηση και καταγραφή των δραστηριοτήτων στο σύστημα βοηθά στην ανίχνευση και αντιμετώπιση πιθανών απειλών.

Καταγραφή Ενεργειών: Όλες οι σημαντικές ενέργειες στο σύστημα, όπως η δημιουργία, επεξεργασία και διαγραφή καναλιών, καταγράφονται. Αυτές οι καταγραφές βοηθούν στην ανίχνευση μη εξουσιοδοτημένων ενεργειών και στην ανάλυση των περιστατικών ασφαλείας.

Παρακολούθηση Συστήματος: Η συνεχή παρακολούθηση του συστήματος για ασυνήθιστες δραστηριότητες ή ανωμαλίες βοηθά στην έγκαιρη ανίχνευση και απόκριση σε πιθανούς κινδύνους.

5. Στρατηγικές Ανάκαμψης

Η προετοιμασία για την αντιμετώπιση περιστατικών ασφαλείας και η διασφάλιση της συνέχειας της λειτουργίας του συστήματος είναι κρίσιμη.

Αντίγραφα Ασφαλείας: Τα δεδομένα του συστήματος αποθηκεύονται τακτικά σε ασφαλή αντίγραφα ασφαλείας. Σε περίπτωση παραβίασης ή απώλειας δεδομένων, τα αντίγραφα ασφαλείας επιτρέπουν την γρήγορη αποκατάσταση των πληροφοριών.

Σχέδιο Αντιμετώπισης Έκτακτων Καταστάσεων: Ένα σχέδιο αντιμετώπισης έκτακτων καταστάσεων περιλαμβάνει τα βήματα που πρέπει να ακολουθηθούν σε περίπτωση παραβίασης ασφαλείας ή άλλου περιστατικού. Το σχέδιο αυτό βοηθά στην ελαχιστοποίηση των επιπτώσεων και στην ταχεία αποκατάσταση της λειτουργίας του συστήματος.

Με την υιοθέτηση αυτών των μέτρων ασφάλειας, το σύστημα IoT διασφαλίζει την προστασία των δεδομένων και την ακεραιότητα της λειτουργίας του, παρέχοντας μια αξιόπιστη και ασφαλή πλατφόρμα για τη συλλογή και ανάλυση δεδομένων από συσκευές IoT.

Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης

Το σύστημα IoT που αναπτύχθηκε και περιγράφηκε σε αυτό το έργο αποτελεί μια ολοκληρωμένη και ευέλικτη λύση για την καταγραφή, διαχείριση και ανάλυση δεδομένων από διάφορες συσκευές IoT. Η εφαρμογή βασίζεται στο Laravel framework και προσφέρει μια φιλική προς τον χρήστη διεπαφή για τη δημιουργία και διαχείριση καναλιών δεδομένων. Οι χρήστες μπορούν να συλλέγουν δεδομένα από αισθητήρες, να τα αποθηκεύουν σε μια ασφαλή βάση δεδομένων και να τα προβάλλουν σε δυναμικά γραφήματα για ανάλυση και παρακολούθηση.

Από την ανάπτυξη και την υλοποίηση του συστήματος προέκυψαν αρκετά σημαντικά συμπεράσματα:

Η διεπαφή χρήστη είναι σχεδιασμένη με γνώμονα την ευκολία χρήσης, επιτρέποντας στους χρήστες να δημιουργούν και να διαχειρίζονται κανάλια δεδομένων με ελάχιστη προσπάθεια. Οι επιλογές για ενεργοποίηση και απενεργοποίηση πεδίων προσφέρουν ευελιξία και προσαρμοστικότητα, καθιστώντας το σύστημα κατάλληλο για διάφορες εφαρμογές. Ιδιαίτερη έμφαση δόθηκε στην ασφάλεια των δεδομένων, με τη χρήση κρυπτογράφησης και ασφαλών συνδέσεων HTTPS. Η διαχείριση πρόσβασης μέσω μοναδικών API keys και η αυστηρή επικύρωση δεδομένων εξασφαλίζουν την προστασία των πληροφοριών και την αποτροπή μη εξουσιοδοτημένης πρόσβασης.

Η δυνατότητα προβολής των δεδομένων σε δυναμικά γραφήματα (charts) προσφέρει στους χρήστες πολύτιμα εργαλεία ανάλυσης. Η οπτικοποίηση των δεδομένων διευκολύνει την παρακολούθηση των τάσεων και την λήψη ενημερωμένων αποφάσεων. Το σύστημα έχει σχεδιαστεί για να παρέχει αξιόπιστη και συνεπή απόδοση. Οι στρατηγικές ανάκαμψης και τα αντίγραφα ασφαλείας εξασφαλίζουν τη συνέχεια της λειτουργίας και την αποκατάσταση των δεδομένων σε περίπτωση απρόοπτων περιστατικών.

Παρά την επιτυχία της υλοποίησης, υπάρχουν αρκετές περιοχές όπου μπορούν να γίνουν βελτιώσεις για να ενισχυθεί η λειτουργικότητα και η απόδοση του συστήματος.

Η προσθήκη υποστήριξης για περισσότερα είδη συσκευών και αισθητήρων θα καταστήσει το σύστημα ακόμα πιο ευέλικτο και κατάλληλο για ευρύτερη γκάμα εφαρμογών. Η ενσωμάτωση νέων πρωτοκόλλων επικοινωνίας και η υποστήριξη περισσότερων πλατφορμών IoT θα ενισχύσουν την επεκτασιμότητα του συστήματος. Η βελτιστοποίηση των ερωτημάτων προς τη βάση δεδομένων και η χρήση τεχνικών caching μπορεί να βελτιώσει την απόδοση του συστήματος, ιδιαίτερα σε περιπτώσεις με μεγάλο όγκο δεδομένων. Επιπλέον, η χρήση τεχνολογιών όπως τα WebSockets για πραγματικό χρόνο ενημερώσεις μπορεί να βελτιώσει την απόδοση και την εμπειρία χρήστη. Αν και το σύστημα διαθέτει ισχυρά μέτρα ασφαλείας, η προσθήκη επιπλέον επιπέδων ασφαλείας, όπως η πολυπαραγοντική αυθεντικοποίηση (MFA) και η ενσωμάτωση προτύπων ασφαλείας όπως το OAuth για τη διαχείριση ταυτότητας και πρόσβασης, μπορεί να βελτιώσει περαιτέρω την ασφάλεια.

Η αναβάθμιση της διεπαφής χρήστη με τη χρήση σύγχρονων τεχνολογιών frontend, όπως το Vue.js ή το React, μπορεί να βελτιώσει την αλληλεπίδραση και την εμπειρία χρήστη. Η προσθήκη περισσότερων δυνατοτήτων ανάλυσης δεδομένων και αναφορών θα προσφέρει στους χρήστες καλύτερα εργαλεία για τη διαχείριση των δεδομένων τους. Η ενσωμάτωση του συστήματος με άλλες πλατφόρμες και υπηρεσίες IoT μπορεί να προσφέρει μεγαλύτερη ευελιξία και να ενισχύσει τη συνεργασία μεταξύ διαφορετικών συστημάτων. Η χρήση προτύπων και ανοιχτών APIs θα διευκολύνει τη διαλειτουργικότητα και την ανταλλαγή δεδομένων μεταξύ διαφορετικών συστημάτων.

Το σύστημα IoT μπορεί να αναβαθμιστεί σε μια ακόμα πιο ισχυρή, ευέλικτη και ασφαλή πλατφόρμα, ικανή να καλύψει τις ανάγκες ενός ευρύτερου φάσματος εφαρμογών και χρηστών. Οι συνεχείς βελτιώσεις και η προσαρμογή στις εξελισσόμενες ανάγκες της τεχνολογίας και της αγοράς θα διασφαλίσουν ότι το σύστημα παραμένει σύγχρονο και αποδοτικό.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] ThingSpeak. (n.d.). Ανακτήθηκε από https://thingspeak.com

[2]MathWorks. (n.d.). ThingSpeak Documentation. Ανακτήθηκε από https://www.mathworks.com/help/thingspeak

[3]ThingSpeak – IoT Platform for MATLAB Users. (n.d.). Ανακτήθηκε από https://thingspeak.com/pages/learn_more

[4] ThingsBoard. (n.d.). Ανακτήθηκε από https://thingsboard.io

[5] ThingsBoard Documentation. (n.d.). Ανακτήθηκε από https://thingsboard.io/docs

[6] What is ThingsBoard? (n.d.). Ανακτήθηκε από https://www.techrepublic.com/article/what-is-thingsboard

[7] Kaa IoT Platform. (n.d.). Ανακτήθηκε από https://www.kaaproject.org

[8] Kaa Documentation. (n.d.). Ανακτήθηκε από https://docs.kaaproject.org

[9] Losant. (n.d.). Ανακτήθηκε από https://www.losant.com

[10] Losant Documentation. (n.d.). Ανακτήθηκε από https://docs.losant.com

[11] OpenRemote. (n.d.). Ανακτήθηκε από https://www.openremote.io

[12] OpenRemote Documentation. (n.d.). Ανακτήθηκε από https://github.com/openremote/openremote

[13] Cayenne. (n.d.). Ανακτήθηκε από https://cayenne.mydevices.com

[14] Cayenne Documentation. (n.d.). Ανακτήθηκε από https://developers.mydevices.com/cayenne/docs

[15] Particle. (n.d.). Ανακτήθηκε από https://www.particle.io

[16] Particle Documentation. (n.d.). Ανακτήθηκε από https://docs.particle.io

[17] AWS IoT Core. (n.d.). Ανακτήθηκε από https://aws.amazon.com/iot-core

[18] AWS IoT Core Documentation. (n.d.). Ανακτήθηκε από https://docs.aws.amazon.com/iot/latest/developerguide

[19] PHP: The Right Way. (n.d.). Ανακτήθηκε από https://phptherightway.com

[20] W3Techs. (n.d.). Usage Statistics of PHP for Websites. Ανακτήθηκε από https://w3techs.com/technologies/details/pl-php

[21] https://laravel.com/

[22] Python Software Foundation. (n.d.). Python History. Ανακτήθηκε από https://www.python.org/doc/essays/blurb

[23] Van Rossum, G. (1991). An Introduction to Python. In Python 1.0.

[24] Python Software Foundation. (n.d.). What's New In Python 3.9. Ανακτήθηκε από https://docs.python.org/3/whatsnew/3.9.html

[25] Full Stack Python. (n.d.). REST APIs with Flask and Python. Ανακτήθηκε από https://www.fullstackpython.com/flask.html

[26] Laravel Documentation. (n.d.). HTTP Client - Laravel. Ανακτήθηκε από https://laravel.com/docs/8.x/http-client

[27] MySQL. (n.d.). MySQL History. Ανακτήθηκε από https://dev.mysql.com/doc/refman/8.0/en/history.html

[28] Oracle. (n.d.). Why MySQL? Ανακτήθηκε από https://www.mysql.com/why-mysql

[29] MariaDB Foundation. (n.d.). History of MariaDB. Ανακτήθηκε από https://mariadb.org/about/history/

ΠΑΡΑΡΤΗΜΑ Α

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

DashboardController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Session;
use Illuminate\Support\Facades\Log;

class DashboardController extends Controller
{
    public function index()
    {
        if (!Session::has('user_id')) {
            return redirect('welcome');
        }

        $channels = DB::table('channel')->where('user_id', Session::get('user_id'))->get();
        return view('dashboard.index', compact('channels'));
    }

    public function createChannel()
    {
        if (!Session::has('user_id')) {
            return redirect('welcome');
        }

        return view('dashboard.create_channel');
    }

    public function storeChannel(Request $request)
    {
        if (!Session::has('user_id')) {
            return redirect('welcome');
        }

        // Επικύρωση των δεδομένων εισόδου
        $validatedData = $request->validate([
            'name' => 'required|string|max:50',
            'description' => 'nullable|string|max:255',
            'apikeywrite' => 'required|string|max:255',
```

```
            'apikeyread' => 'required|string|max:255',
            'field1name' => 'nullable|string|max:50',
            'field2name' => 'nullable|string|max:50',
            'field3name' => 'nullable|string|max:50',
            'field4name' => 'nullable|string|max:50',
            'field5name' => 'nullable|string|max:50',
        ]);

        // Εισαγωγή του νέου καναλιού στη βάση δεδομένων
        DB::table('channel')->insert([
            'user_id' => Session::get('user_id'),
            'name' => $validatedData['name'],
            'description' => $validatedData['description'],
            'apikeywrite' => $validatedData['apikeywrite'],
            'apikeyread' => $validatedData['apikeyread'],
            'field1check' => $request->has('field1check') ? 1 : 0,
            'field1name' => $validatedData['field1name'],
            'field2check' => $request->has('field2check') ? 1 : 0,
            'field2name' => $validatedData['field2name'],
            'field3check' => $request->has('field3check') ? 1 : 0,
            'field3name' => $validatedData['field3name'],
            'field4check' => $request->has('field4check') ? 1 : 0,
            'field4name' => $validatedData['field4name'],
            'field5check' => $request->has('field5check') ? 1 : 0,
            'field5name' => $validatedData['field5name'],
            'created_at' => now(),
            'updated_at' => now(),
        ]);

        // Ανακατεύθυνση στο dashboard
        return redirect()->route('dashboard');
    }

    public function editChannel($id)
    {
        if (!Session::has('user_id')) {
            return redirect('welcome');
        }

        $channel = DB::table('channel')->where('id', $id)->first();
        return view('dashboard.edit_channel', compact('channel'));
    }

    public function updateChannel(Request $request, $id)
    {
        if (!Session::has('user_id')) {
            return redirect('welcome');
        }
```

```
        DB::table('channel')->where('id', $id)->update([
            'name' => $request->name,
            'description' => $request->description,
            'apikeywrite' => $request->apikeywrite,
            'apikeyread' => $request->apikeyread,
            'field1check' => $request->has('field1check') ? 1 : 0,
            'field1name' => $request->field1name,
            'field2check' => $request->has('field2check') ? 1 : 0,
            'field2name' => $request->field2name,
            'field3check' => $request->has('field3check') ? 1 : 0,
            'field3name' => $request->field3name,
            'field4check' => $request->has('field4check') ? 1 : 0,
            'field4name' => $request->field4name,
            'field5check' => $request->has('field5check') ? 1 : 0,
            'field5name' => $request->field5name,
            'updated_at' => now(),
        ]);

        return redirect('/dashboard');
    }

    public function deleteChannel($id)
    {
        if (!Session::has('user_id')) {
            return redirect('welcome');
        }

        DB::table('channel')->where('id', $id)->delete();
        return redirect('/dashboard');
    }

    public function showChannelData($id)
    {
        if (!Session::has('user_id')) {
            return redirect('welcome');
        }

        $channel = DB::table('channel')->where('id', $id)->first();
        $fieldvalues = DB::table('fieldvalue')->where('channel_id', $id)->get();
        return view('dashboard.show_channel_data', compact('channel', 'fieldvalues'));
    }
}
```

AuthController.php

```
<?php
```

```php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Session;
use Illuminate\Support\Facades\Log;

class AuthController extends Controller
{
    public function showRegister()
    {
        return view('register');
    }

    public function register(Request $request)
    {
        // Επικύρωση των δεδομένων εισόδου
        $validatedData = $request->validate([
            'firstname' => 'required|string|max:255',
            'lastname' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:user',
            'password' => 'required|string|min:6|confirmed',
        ]);

        // Δημιουργία του νέου χρήστη στη βάση δεδομένων
        DB::table('user')->insert([
            'firstname' => $validatedData['firstname'],
            'lastname' => $validatedData['lastname'],
            'email' => $validatedData['email'],
            'password' => $validatedData['password'],//Hash::make($validatedData['password']),
            'created_at' => now(),
            'updated_at' => now(),
        ]);

        // Αυτόματη σύνδεση του χρήστη μετά την εγγραφή
        $user = DB::table('user')->where('email', $validatedData['email'])->first();
        Session::put('user_id', $user->id);
        Session::put('user_email', $user->email);

        // Ανακατεύθυνση στο dashboard
        return redirect()->route('dashboard');
    }

    public function showLogin()
    {
        return view('login');
    }
```

```
    public function login(Request $request)
    {
        $user = DB::table('user')->where('email', $request->email)->first();
        //Log::info(json_encode($user));

        if ($user && ($request->password === $user->password)) {
            Session::put('user_id', $user->id);
            Session::put('user_email', $user->email);

            //Log::info('valid Credentials');
            return redirect('dashboard');
        } else {
            //Log::info('Invalid Credentials');
            return redirect('login')->with('error', 'Invalid Credentials');
        }
    }

    public function logout()
    {
        Session::flush();
        return redirect('welcome');
    }
}
```

ApiController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Log;

class ApiController extends Controller
{
    public function update(Request $request)
    {
        Log::info('ApiController update');

        $channel = DB::table('channel')->where('apikeywrite', $request->apikeywrite)->first();
        if ($channel) {
            DB::table('fieldvalue')->insert([
                'channel_id' => $channel->id,
                'field1value' => $request->field1,
                'field2value' => $request->field2,
                'field3value' => $request->field3,
                'field4value' => $request->field4,
```

```
                'field5value' => $request->field5,
                'created_at' => now(),
            ]);
            return response()->json(['success' => true]);
        } else {
            return response()->json(['error' => 'Invalid API Key'], 400);
        }
    }

    public function read(Request $request)
    {
        Log::info('ApiController read');
        Log::info(json_encode($request));
        $channel = DB::table('channel')->where('apikeyread', $request->apikeyread)->first();
        Log::info(json_encode($channel));
        if ($channel) {
            $fieldvalues = DB::table('fieldvalue')
                ->where('channel_id', $channel->id)
                ->orderBy('created_at', 'desc')
                ->limit($request->results)
                ->get();
            return response()->json($fieldvalues);
        } else {
            return response()->json(['error' => 'Invalid API Key'], 400);
        }
    }
}
```

<!-- welcome.blade.php -->

```
@extends('header')

@section('title', 'Welcome')

@section('content')
<div class="card shadow">
    <div class="card-body">
        <h1 class="card-title">Welcome to IoT Project</h1>
        <a href="{{ route('showLogin') }}" class="btn btn-primary">Login</a>
        <a href="{{ route('register') }}" class="btn btn-secondary">Register</a>
    </div>
</div>
@endsection
```

login.blade.php

```
@extends('header')

@section('title', 'Login')

@section('content')
<div class="card shadow">
    <div class="card-body">
        <h1 class="card-title">Login</h1>
        <form action="{{ route('login') }}" method="POST">
            @csrf
            <div class="form-group">
                <label for="email">Email:</label>
                <input type="email" class="form-control" name="email" required>
            </div>
            <div class="form-group">
                <label for="password">Password:</label>
                <input type="password" class="form-control" name="password" required>
            </div>
            <button type="submit" class="btn btn-primary">Login</button>
        </form>
    </div>
</div>
@endsection
```

register.blade.php

```
@extends('header')

@section('title', 'Register')

@section('content')
<div class="card shadow">
    <div class="card-body">
        <h1 class="card-title">Register</h1>
        <form action="{{ route('register') }}" method="POST">
            @csrf
            <div class="form-group">
                <label for="firstname">First Name:</label>
                <input type="text" class="form-control" name="firstname" required>
            </div>
            <div class="form-group">
                <label for="lastname">Last Name:</label>
                <input type="text" class="form-control" name="lastname" required>
            </div>
            <div class="form-group">
                <label for="email">Email:</label>
                <input type="email" class="form-control" name="email" required>
```

```
            </div>
            <div class="form-group">
                <label for="password">Password:</label>
                <input type="password" class="form-control" name="password" required>
            </div>
            <div class="form-group">
                <label for="password_confirmation">Confirm Password:</label>
                <input type="password" class="form-control" name="password_confirmation" required>
            </div>
            <button type="submit" class="btn btn-primary">Register</button>
        </form>
    </div>
</div>
@endsection
```

index.blade.php

```
@extends('header')

@section('title', 'Dashboard')

@section('content')
<div class="card shadow">
    <div class="card-body">
        <h1 class="card-title">Dashboard</h1>
        <a href="{{ route('createChannel') }}" class="btn btn-primary mb-3">Create New Channel</a>
        <table id="channelsTable" class="table table-striped table-bordered">
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Description</th>
                    <th>Actions</th>
                </tr>
            </thead>
            <tbody>
                @foreach($channels as $channel)
                <tr>
                    <td>{{ $channel->name }}</td>
                    <td>{{ $channel->description }}</td>
                    <td>
                        <a href="{{ route('showChannelData', $channel->id) }}" class="btn btn-info
btn-sm">View</a>
                        <a href="{{ route('editChannel', $channel->id) }}" class="btn btn-warning btn-
sm">Edit</a>
                        <form action="{{ route('deleteChannel', $channel->id) }}" method="POST"
style="display:inline;">
                            @csrf
                            @method('DELETE')
```

```
                        <button type="submit" class="btn btn-danger btn-sm">Delete</button>
                    </form>
                </td>
            </tr>
            @endforeach
        </tbody>
    </table>
</div>
</div>
<script>
    $(document).ready( function () {
        $('#channelsTable').DataTable();
    });
</script>
@endsection
```