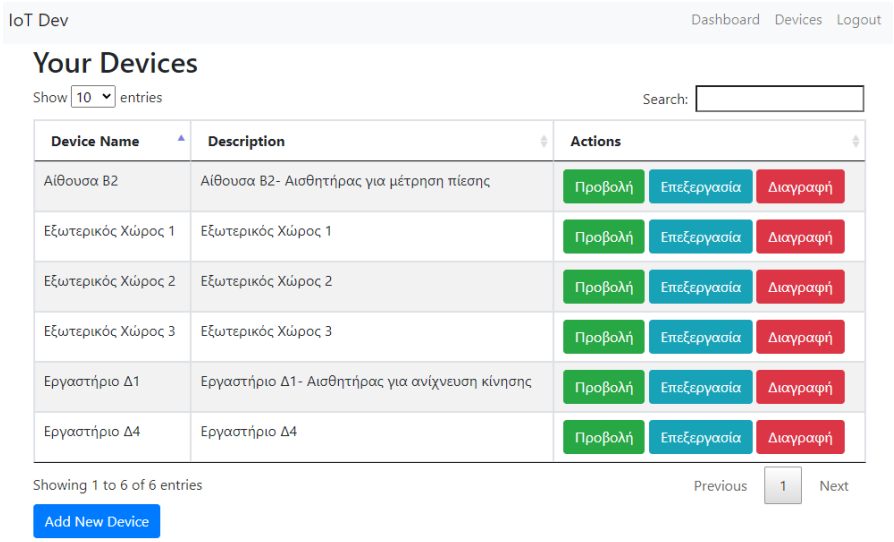
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη πλατφόρμας IoT για τη διαχείριση συσκευών και δεδομένων»



Φοιτητής

ΑΛΕΞΑΝΔΡΟΣ ΠΑΡΣΑΛΙΔΗΣ

515116

Επιβλέπων

Δρ. Κυριάκος Τσιακμάκης

ΙΟΥΝΙΟΣ 2024

Ανάπτυξη πλατφόρμας IoT για τη διαχείριση συσκευών και δεδομένων

Κωδικός: 23322

Φοιτητής: Παρσαλίδης Αλέξανδρος

Εισηγητής: Δρ Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 05-11-2023

Ημερομηνία περάτωσης Π.Ε. 26-05-2024

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.

Περίληψη

Η εργασία αυτή αφορά την ανάπτυξη της πλατφόρμας IoTDev, μιας ολοκληρωμένης εφαρμογής για τη διαχείριση IoT συσκευών, τη συλλογή δεδομένων από αισθητήρες και την προβολή αυτών των δεδομένων σε μορφή γραφημάτων. Η πλατφόρμα χρησιμοποιεί το Flask framework για το backend και το Bootstrap για το frontend, ενώ η αποθήκευση των δεδομένων πραγματοποιείται σε βάση δεδομένων MySQL. Οι χρήστες μπορούν να προσθέτουν, να επεξεργάζονται και να διαγράφουν συσκευές, να βλέπουν δεδομένα σε πραγματικό χρόνο και να διαχειρίζονται API keys για ασφαλή πρόσβαση. Η εργασία περιλαμβάνει επίσης μια βιβλιογραφική ανασκόπηση άλλων πλατφορμών IoT και προτείνει βελτιώσεις για τη μελλοντική ανάπτυξη της IoTDev.

« Development of an IoT Platform for Device and Data Management»

Abstract

This work concerns the development of the IoTDev platform, an integrated application for managing IoT devices, collecting data from sensors and displaying this data in the form of graphs. The platform uses the Flask framework for the backend and Bootstrap for the frontend, while the data is stored in a MySQL database. Users can add, edit and delete devices, view real-time data and manage API keys for secure access. The paper also includes a literature review of other IoT platforms and suggests improvements for the future development of IoTDev.

Ευχαριστίες

Εκφράζω τις ευχαριστίες μου στους γονείς μου για την αδιάλειπτη στήριξή τους, καθώς και στον επιβλέποντα μου για την αδιάκοπη καθοδήγηση, τις συμβουλές του και τη συνεισφορά του στην εφαρμογή.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Στην εποχή της ραγδαίας ανάπτυξης των τεχνολογιών πληροφορικής και επικοινωνιών, οι συσκευές IoT (Internet of Things) παίζουν έναν ολοένα και πιο σημαντικό ρόλο στην καθημερινή μας ζωή και στις επιχειρηματικές διαδικασίες. Οι συσκευές αυτές επιτρέπουν τη συλλογή, αποθήκευση και ανάλυση δεδομένων από το περιβάλλον, παρέχοντας πολύτιμες πληροφορίες και συμβάλλοντας στη λήψη τεκμηριωμένων αποφάσεων. Η ανάγκη για αποτελεσματική διαχείριση αυτών των δεδομένων και των συσκευών που τα παράγουν έχει οδηγήσει στην ανάπτυξη πλατφορμών και συστημάτων που επιτρέπουν τη διασύνδεση, τον έλεγχο και την ανάλυση των IoT συσκευών.

Η εργασία αυτή αφορά την ανάπτυξη της πλατφόρμας IoTDev, μιας ολοκληρωμένης εφαρμογής που επιτρέπει τη διαχείριση IoT συσκευών, τη συλλογή δεδομένων από αισθητήρες και την προβολή αυτών των δεδομένων σε μορφή γραφημάτων. Η IoTDev είναι σχεδιασμένη να είναι φιλική προς τον χρήστη, παρέχοντας εύκολη και ασφαλή πρόσβαση στις λειτουργίες της μέσω μιας διαδικτυακής διεπαφής.

Η εφαρμογή αναπτύχθηκε χρησιμοποιώντας το Flask framework για το backend, το Bootstrap και τη βιβλιοθήκη DataTables για το frontend, και την MySQL για την αποθήκευση των δεδομένων. Η χρήση αυτών των τεχνολογιών εξασφαλίζει την ευελιξία, την επεκτασιμότητα και την ασφάλεια της πλατφόρμας, καλύπτοντας τις ανάγκες των χρηστών για διαχείριση IoT συσκευών.

Κάποιοι από τους στόχους της εργασίας περιλαμβάνουν τη δημιουργία ενός συστήματος που επιτρέπει στους χρήστες να εγγράφονται, να συνδέονται και να διαχειρίζονται τις IoT συσκευές τους μέσω μιας εύχρηστης διεπαφής, την ενσωμάτωση λειτουργιών για την προσθήκη, επεξεργασία και διαγραφή συσκευών, καθώς και την ενεργοποίηση και απενεργοποίηση συγκεκριμένων πεδίων δεδομένων για κάθε συσκευή. Επίσης, την παροχή δυνατοτήτων για τη συλλογή δεδομένων από αισθητήρες IoT συσκευών και την αποθήκευσή τους στη βάση δεδομένων και την προβολή των δεδομένων σε μορφή γραφημάτων χρησιμοποιώντας το Google Charts, διευκολύνοντας την ανάλυση των δεδομένων σε πραγματικό χρόνο. Τέλος, υλοποιήθηκε σύστημα χρήσης API keys για την εξουσιοδότηση των αιτημάτων εγγραφής και ανάγνωσης δεδομένων από τις συσκευές, διασφαλίζοντας την προστασία των δεδομένων από μη εξουσιοδοτημένη πρόσβαση.

Η συνεισφορά της εργασίας αυτής έγκειται στην παροχή μιας ολοκληρωμένης λύσης για τη διαχείριση IoT συσκευών, η οποία μπορεί να χρησιμοποιηθεί από μεμονωμένους χρήστες και επιχειρήσεις για την παρακολούθηση και ανάλυση δεδομένων από αισθητήρες. Η πλατφόρμα IoTDev όχι μόνο βελτιώνει την αποτελεσματικότητα και την ακρίβεια της διαχείρισης δεδομένων IoT, αλλά επίσης παρέχει ένα ασφαλές και αξιόπιστο περιβάλλον για τη συλλογή και επεξεργασία αυτών των δεδομένων. Μέσω της IoTDev, οι χρήστες μπορούν να αξιοποιήσουν πλήρως τις δυνατότητες των IoT συσκευών τους,

επιτυγχάνοντας καλύτερη κατανόηση και έλεγχο των παραμέτρων που επηρεάζουν τις διαδικασίες και τις αποφάσεις τους.

1.2 Δομή της εργασίας

Η εργασία αυτή είναι δομημένη σε πέντε κύρια κεφάλαια, καθένα από τα οποία καλύπτει διαφορετικές πτυχές της ανάπτυξης και της λειτουργίας της πλατφόρμας IoTDev.

Το πρώτο κεφάλαιο παρέχει μια εισαγωγή στην εργασία και στη συνέχεια περιγράφει τη δομή της. Αναλύεται η σημασία των IoT πλατφορμών και παρουσιάζεται η γενική κατεύθυνση και οι στόχοι της ανάπτυξης της πλατφόρμας IoTDev.

Στο δεύτερο κεφάλαιο γίνεται μια βιβλιογραφική ανασκόπηση των πιο γνωστών IoT πλατφορμών, όπως το ThingSpeak, το ThinkBoard, το Grafana και το AWS IoT Core. Κάθε πλατφόρμα παρουσιάζεται ξεχωριστά, τονίζοντας τα κύρια χαρακτηριστικά και τις εφαρμογές της.

Το τρίτο κεφάλαιο εστιάζει στις τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της πλατφόρμας IoTDev. Παρουσιάζονται η PHP, η MySQL και το Laravel, καθώς και οι λόγοι για την επιλογή τους.

Το τέταρτο κεφάλαιο περιγράφει αναλυτικά το έργο IoTDev.

Το πέμπτο κεφάλαιο περιλαμβάνει τα συμπεράσματα της εργασίας και προτείνει βελτιώσεις για τη μελλοντική ανάπτυξη της πλατφόρμας IoTDev. Αναλύονται οι επιτυχίες και οι προκλήσεις που αντιμετωπίστηκαν κατά την ανάπτυξη της εφαρμογής, και παρουσιάζονται προτάσεις για την περαιτέρω βελτίωση της απόδοσης, της ασφάλειας και της λειτουργικότητας της πλατφόρμας.

Κεφάλαιο 2ο: Βιβλιογραφική Ανασκόπηση

2.1 Εισαγωγή

Η τεχνολογία του Internet of Things (IoT) έχει αλλάξει ριζικά τον τρόπο με τον οποίο αλληλεπιδρούμε με τις συσκευές και τα δεδομένα. Η βασική ιδέα πίσω από το IoT είναι η διασύνδεση φυσικών συσκευών στο Διαδίκτυο, επιτρέποντας την ανταλλαγή δεδομένων και την απομακρυσμένη διαχείριση. Οι συσκευές IoT περιλαμβάνουν αισθητήρες, ενεργοποιητές, κάμερες, και πολλές άλλες συσκευές που χρησιμοποιούνται σε διάφορους τομείς όπως η υγειονομική περίθαλψη, η γεωργία, η βιομηχανία, και τα έξυπνα σπίτια.

Μια IoT πλατφόρμα είναι μια λογισμική υποδομή που διευκολύνει την ανάπτυξη, διαχείριση και παρακολούθηση IoT συσκευών και εφαρμογών. Αυτές οι πλατφόρμες προσφέρουν μια σειρά από εργαλεία και υπηρεσίες που επιτρέπουν στους προγραμματιστές να συνδέουν, να παρακολουθούν και να ελέγχουν τις IoT συσκευές τους με αποτελεσματικό τρόπο. Οι IoT πλατφόρμες έχουν καταστεί αναγκαίες λόγω της πολυπλοκότητας και της μεγάλης κλίμακας των IoT συστημάτων.

Οι IoT πλατφόρμες παρέχουν τη δυνατότητα διασύνδεσης πολλών και διαφορετικών συσκευών σε ένα ενιαίο δίκτυο. Αυτό επιτρέπει την εύκολη επικοινωνία μεταξύ των συσκευών και τη συλλογή δεδομένων από διάφορες πηγές. Οι πλατφόρμες προσφέρουν εργαλεία για τη συλλογή, αποθήκευση και ανάλυση των δεδομένων που παράγονται από τις IoT συσκευές. Αυτό επιτρέπει στους χρήστες να αξιοποιήσουν τα δεδομένα για τη λήψη τεκμηριωμένων αποφάσεων.

Η ασφάλεια των δεδομένων και των συσκευών είναι ζωτικής σημασίας στις IoT πλατφόρμες. Παρέχουν μηχανισμούς ασφαλείας για την προστασία των δεδομένων από μη εξουσιοδοτημένη πρόσβαση και για την εξασφάλιση της ιδιωτικότητας των χρηστών. Οι IoT πλατφόρμες είναι σχεδιασμένες να κλιμακώνονται ώστε να υποστηρίζουν μεγάλο αριθμό συσκευών και μεγάλες ποσότητες δεδομένων. Αυτό επιτρέπει στις επιχειρήσεις να επεκτείνουν τα IoT συστήματά τους χωρίς να αντιμετωπίζουν προβλήματα απόδοσης. Μέσω των IoT πλατφορμών, οι χρήστες μπορούν να διαχειρίζονται και να ελέγχουν τις συσκευές τους απομακρυσμένα. Αυτό περιλαμβάνει την αναβάθμιση λογισμικού, την παρακολούθηση της κατάστασης των συσκευών και την εκτέλεση ενεργειών με βάση τα δεδομένα των αισθητήρων. Οι πλατφόρμες συχνά παρέχουν εργαλεία για την οπτικοποίηση των δεδομένων, όπως γραφήματα και πίνακες. Αυτά τα εργαλεία βοηθούν τους χρήστες να κατανοήσουν καλύτερα τις τάσεις και τα μοτίβα στα δεδομένα τους.

Οι IoT πλατφόρμες χρησιμοποιούνται σε διάφορους τομείς και εφαρμογές.

Στις έξυπνες πόλεις, οι IoT πλατφόρμες χρησιμοποιούνται για τη διαχείριση της κυκλοφορίας, τη βελτίωση της δημόσιας ασφάλειας, τη διαχείριση της ενέργειας και των υδάτινων πόρων, και την παρακολούθηση της ποιότητας του αέρα.

Στον βιομηχανικό τομέα, οι IoT πλατφόρμες επιτρέπουν την παρακολούθηση και τον έλεγχο των μηχανημάτων, τη διαχείριση της παραγωγής και τη βελτίωση της αποδοτικότητας μέσω της ανάλυσης των δεδομένων.

Στην υγειονομική περίθαλψη, οι IoT πλατφόρμες χρησιμοποιούνται για την παρακολούθηση των ασθενών, τη διαχείριση των ιατρικών συσκευών και τη συλλογή δεδομένων για τη βελτίωση της ποιότητας της φροντίδας.

Στα έξυπνα σπίτια, οι IoT πλατφόρμες επιτρέπουν τον έλεγχο των συσκευών και των συστημάτων του σπιτιού, όπως ο φωτισμός, η θέρμανση, η ασφάλεια και τα οικιακά συστήματα ψυχαγωγίας.

Η ανάπτυξη της πλατφόρμας IoTDev επωφελήθηκε σημαντικά από τις δυνατότητες και τα εργαλεία που προσφέρουν οι IoT πλατφόρμες. Η IoTDev ενσωματώνει λειτουργίες για τη διαχείριση συσκευών, τη συλλογή και ανάλυση δεδομένων, και την απομακρυσμένη παρακολούθηση, παρέχοντας στους χρήστες μια ολοκληρωμένη λύση για τη διαχείριση των IoT συσκευών τους.

Η IoTDev επιτρέπει την εύκολη διαχείριση και παρακολούθηση πολλαπλών IoT συσκευών, βελτιώνοντας την αποδοτικότητα των χρηστών και των επιχειρήσεων. Μέσω της χρήσης ασφαλών μηχανισμών για την προστασία των δεδομένων και την εξουσιοδότηση των χρηστών, η IoTDev εξασφαλίζει την αξιοπιστία και την ασφάλεια των δεδομένων.

Η IoTDev παρέχει εργαλεία οπτικοποίησης και ανάλυσης δεδομένων, διευκολύνοντας τους χρήστες να εξάγουν πολύτιμες πληροφορίες από τα δεδομένα των IoT συσκευών τους.

2.2 Πλατφόρμες IoT

Μερικές από τις πιο γνωστές πλατφόρμες IoT, συγκεκριμένα το ThingSpeak, το ThinkBoard, το Grafana και το AWS IoT Core. Αυτές οι πλατφόρμες παρέχουν ποικίλα εργαλεία και υπηρεσίες για τη διαχείριση, την ανάλυση και την οπτικοποίηση δεδομένων από IoT συσκευές, και έχουν χρησιμοποιηθεί ευρέως σε πολλές εφαρμογές.

2.2.1 ThingSpeak

Το ThingSpeak είναι μια ανοιχτή πλατφόρμα IoT που επιτρέπει στους χρήστες να συλλέγουν, αποθηκεύουν, αναλύουν και οπτικοποιούν δεδομένα από αισθητήρες σε πραγματικό χρόνο. Παρέχει ένα εύχρηστο API για τη διασύνδεση IoT συσκευών και τη μεταφορά δεδομένων προς την πλατφόρμα.

Οι χρήστες μπορούν να δημιουργήσουν κανάλια δεδομένων για κάθε συσκευή, όπου κάθε κανάλι περιλαμβάνει πολλαπλά πεδία για την αποθήκευση δεδομένων. Το ThingSpeak προσφέρει επίσης εργαλεία ανάλυσης δεδομένων και οπτικοποίησης, επιτρέποντας στους χρήστες να δημιουργούν γραφήματα και πίνακες με τα δεδομένα τους, καθώς και να εκτελούν βασικές στατιστικές αναλύσεις [1].

2.2.2 ThinkBoard

Το ThinkBoard είναι μια άλλη πλατφόρμα IoT που παρέχει δυνατότητες συλλογής και ανάλυσης δεδομένων από IoT συσκευές. Επικεντρώνεται στην απλότητα χρήσης και στην παροχή μιας εύκολης στη χρήση διεπαφής για την παρακολούθηση και τη διαχείριση των συσκευών.

Η πλατφόρμα ThinkBoard επιτρέπει στους χρήστες να δημιουργούν και να διαχειρίζονται κανάλια δεδομένων, να ρυθμίζουν ειδοποιήσεις βάσει των δεδομένων αισθητήρων και να οπτικοποιούν τα δεδομένα μέσω γραφημάτων και άλλων εργαλείων οπτικοποίησης. Η πλατφόρμα είναι σχεδιασμένη για να είναι φιλική προς τους προγραμματιστές, προσφέροντας λεπτομερή τεκμηρίωση και παραδείγματα κώδικα για τη σύνδεση IoT συσκευών [2].

2.2.3 Grafana

Το Grafana είναι μια πλατφόρμα ανοιχτού κώδικα για την οπτικοποίηση και την παρακολούθηση δεδομένων. Ενώ δεν είναι αποκλειστικά σχεδιασμένη για IoT, χρησιμοποιείται ευρέως σε IoT εφαρμογές λόγω των ισχυρών δυνατοτήτων οπτικοποίησης και της υποστήριξης πολλαπλών πηγών δεδομένων.

Το Grafana επιτρέπει στους χρήστες να δημιουργούν δυναμικούς πίνακες ελέγχου (dashboards) με γραφήματα, χάρτες και άλλα εργαλεία οπτικοποίησης, που μπορούν να αντλήσουν δεδομένα από βάσεις δεδομένων, APIs και άλλες πηγές. Η ευελιξία και η επεκτασιμότητα του Grafana το καθιστούν ένα δημοφιλές εργαλείο για την παρακολούθηση και την ανάλυση δεδομένων σε πραγματικό χρόνο [3].

2.2.4 AWS IoT Core

Το AWS IoT Core είναι μια πλατφόρμα IoT που προσφέρεται από την Amazon Web Services (AWS). Παρέχει μια ολοκληρωμένη σειρά υπηρεσιών για τη σύνδεση και τη διαχείριση δισεκατομμυρίων IoT συσκευών με ασφαλή και επεκτάσιμο τρόπο.

Το AWS IoT Core επιτρέπει τη σύνδεση συσκευών με το cloud μέσω ασφαλών επικοινωνιών και την αποστολή δεδομένων σε πραγματικό χρόνο. Προσφέρει επίσης δυνατότητες για την επεξεργασία και την ανάλυση των δεδομένων, καθώς και την ενσωμάτωση με άλλες υπηρεσίες της AWS για την αποθήκευση, την ανάλυση και την οπτικοποίηση των δεδομένων. Η ισχυρή υποδομή και οι δυνατότητες ασφαλείας της AWS καθιστούν το AWS IoT Core μια από τις πιο αξιόπιστες πλατφόρμες IoT [4].

Κεφάλαιο 3ο: Γλώσσες προγραμματισμού και εργαλεία

3.1 Python

Η Python είναι μια υψηλού επιπέδου, ερμηνευόμενη γλώσσα προγραμματισμού που είναι γνωστή για την απλότητά της και τη δύναμή της. Δημιουργήθηκε από τον Guido van Rossum και κυκλοφόρησε για πρώτη φορά το 1991. Η Python έχει σχεδιαστεί για να είναι εύκολη στην ανάγνωση και στη συγγραφή, καθιστώντας την ιδανική για αρχάριους προγραμματιστές, ενώ παράλληλα διαθέτει τις δυνατότητες που απαιτούνται για την ανάπτυξη σύνθετων εφαρμογών [5].

Η ανάπτυξη της Python ξεκίνησε στα τέλη της δεκαετίας του 1980 και η πρώτη έκδοση κυκλοφόρησε το 1991. Η γλώσσα επηρεάστηκε σε μεγάλο βαθμό από την ABC, μια άλλη γλώσσα προγραμματισμού που αναπτύχθηκε στο CWI (Centrum Wiskunde & Informatica) στην Ολλανδία. Ο Guido van Rossum σχεδίασε την Python για να είναι μια ελκυστική εναλλακτική λύση στις υπάρχουσες γλώσσες προγραμματισμού, συνδυάζοντας τη δύναμη και την ευελιξία με μια καθαρή και απλή σύνταξη.

Με την πάροδο του χρόνου, η Python γνώρισε πολλές εξελίξεις και εκδόσεις.

- Python 1.0: Κυκλοφόρησε τον Ιανουάριο του 1994, με βασικές δυνατότητες όπως τα εργαλεία εξαίρεσης, οι λειτουργίες και τα αρχεία module.
- Python 2.0: Κυκλοφόρησε τον Οκτώβριο του 2000, εισάγοντας νέα χαρακτηριστικά όπως η συλλογή απορριμμάτων (garbage collection) και η λίστα κατανόησης (list comprehensions).
- Python 3.0: Κυκλοφόρησε τον Δεκέμβριο του 2008, ήταν μια σημαντική αναβάθμιση που έφερε πολλές αλλαγές και ασυμβατότητες με τις προηγούμενες εκδόσεις, βελτιώνοντας την καθαρότητα και τη συνέπεια της γλώσσας.

Η Python έχει δύο κύριες γραμμές εκδόσεων: την Python 2.x και την Python 3.x. Η Python 3 έχει σχεδιαστεί για να αντικαταστήσει την Python 2, με πολλές βελτιώσεις και νέες δυνατότητες. Η υποστήριξη για την Python 2 σταμάτησε τον Ιανουάριο του 2020, και οι προγραμματιστές συνιστώνται να χρησιμοποιούν την Python 3 για νέα έργα.

Η Python είναι γνωστή για την ευελιξία και την ευχρηστία της. Έχει μια σύνταξη που είναι εύκολη στην ανάγνωση και στη συγγραφή, κάνοντας την ιδανική για γρήγορη ανάπτυξη και πρωτότυπα. Η Python συνοδεύεται από μια εκτεταμένη τυπική βιβλιοθήκη που περιλαμβάνει ενσωματωμένες λειτουργίες για πολλές κοινές εργασίες προγραμματισμού. Μπορεί να ενσωματωθεί με άλλες γλώσσες

προγραμματισμού και τεχνολογίες, όπως το C, το C++, η Java, και άλλες. Υποστηρίζει πλήρως τον αντικειμενοστραφή προγραμματισμό, επιτρέποντας τη δημιουργία και τη χρήση αντικειμένων και κλάσεων. Με βιβλιοθήκες όπως το asyncio, η Python επιτρέπει την ανάπτυξη ασύγχρονων εφαρμογών υψηλής απόδοσης.

Η Python χρησιμοποιείται σε ένα ευρύ φάσμα εφαρμογών και βιομηχανιών. Μερικά παραδείγματα περιλαμβάνουν

- Πλατφόρμες όπως το Django και το Flask διευκολύνουν την ανάπτυξη ισχυρών web εφαρμογών.
- Επιστήμη Δεδομένων και Μηχανική Μάθηση. Βιβλιοθήκες όπως το NumPy, το Pandas, το SciPy, το Scikit-learn και το TensorFlow καθιστούν την Python την πρώτη επιλογή για αναλύσεις δεδομένων και ανάπτυξη μοντέλων μηχανικής μάθησης.
- Η Python χρησιμοποιείται ευρέως για την αυτοματοποίηση εργασιών, συμπεριλαμβανομένης της διαχείρισης συστημάτων και της ανάλυσης δεδομένων.
- Η Python χρησιμοποιείται στην ανάπτυξη παιχνιδιών και εφαρμογών πολυμέσων μέσω βιβλιοθηκών όπως το Pygame.
- Η Python είναι ιδανική για τη διαχείριση και τον αυτοματισμό των λειτουργιών των συστημάτων και των δικτύων.

Η Python είναι μια δημοφιλής επιλογή για την ανάπτυξη εφαρμογών IoT λόγω της ευελιξίας της και της μεγάλης υποστήριξης που προσφέρει η κοινότητα. Η Python μπορεί να χρησιμοποιηθεί για προγραμματισμό μικροελεγκτών. Οι μικροελεγκτές όπως το Raspberry Pi και το ESP32 μπορούν να προγραμματιστούν χρησιμοποιώντας MicroPython, μια ελαφριά έκδοση της Python. Η Python μπορεί να συνδέεται με αισθητήρες και να συλλέγει δεδομένα από διάφορες πηγές, αποθηκεύοντας τα σε βάσεις δεδομένων ή στέλνοντάς τα σε πλατφόρμες IoT. Παρέχει ισχυρά εργαλεία για την ανάλυση των δεδομένων IoT, επιτρέποντας στους προγραμματιστές να εξάγουν πολύτιμες πληροφορίες. Χρησιμοποιώντας πλαίσια όπως το Flask ή το Django, οι προγραμματιστές μπορούν να δημιουργήσουν web interfaces για τη διαχείριση και την προβολή δεδομένων IoT.

Εγκατάσταση της Python

Επισκεφθείτε την επίσημη ιστοσελίδα της Python (https://www.python.org) και κατεβάστε την τελευταία έκδοση της Python για το λειτουργικό σας σύστημα.

Ακολουθήστε τις οδηγίες εγκατάστασης για το λειτουργικό σας σύστημα. Βεβαιωθείτε ότι έχετε επιλέξει την επιλογή "Add Python to PATH" κατά την εγκατάσταση.

Ανοίξτε ένα τερματικό (Command Prompt, PowerShell ή Terminal) και πληκτρολογήστε python --version για να επιβεβαιώσετε ότι η Python έχει εγκατασταθεί σωστά.

Υπάρχουν πολλά περιβάλλοντα επεξεργασίας κώδικα που μπορούν να χρησιμοποιηθούν για την ανάπτυξη με την Python.

- IDLE: Το ενσωματωμένο περιβάλλον ανάπτυξης της Python που περιλαμβάνεται με την εγκατάσταση της Python.
- PyCharm: Ένα ισχυρό IDE για την ανάπτυξη με την Python που προσφέρει πλούσια χαρακτηριστικά και εργαλεία.
- Visual Studio Code: Ένας δημοφιλής επεξεργαστής κώδικα με ισχυρές επεκτάσεις για την Python.

Η εκτέλεση κώδικα Python μπορεί να γίνει εύκολα μέσω τερματικού ή από το περιβάλλον επεξεργασίας:

Δημιουργήστε ένα αρχείο Python με την κατάληξη .py (π.χ., script.py). Ανοίξτε ένα τερματικό και πλοηγηθείτε στον φάκελο που περιέχει το αρχείο. Και εκτελέστε τον κώδικα πληκτρολογώντας python script.py.

3.2 Flask

Το Flask είναι ένα μικροπλαίσιο (microframework) για την ανάπτυξη web εφαρμογών γραμμένο σε Python. Δημιουργήθηκε από τον Armin Ronacher και κυκλοφόρησε το 2010. Το Flask είναι γνωστό για την απλότητά του και την ευελιξία του, επιτρέποντας στους προγραμματιστές να δημιουργούν web εφαρμογές και APIs γρήγορα και εύκολα χωρίς την ανάγκη χρήσης πολύπλοκων εργαλείων και βιβλιοθηκών. Το μινιμαλιστικό του σχέδιο επιτρέπει στους προγραμματιστές να επεκτείνουν τη λειτουργικότητά του με διάφορα plugins και επεκτάσεις ανάλογα με τις ανάγκες της εφαρμογής [6].

Το Flask χρησιμοποιείται ευρέως για την ανάπτυξη μικρών έως μεσαίου μεγέθους web εφαρμογών και APIs. Η ευκολία χρήσης και η απλότητά του το καθιστούν ιδανικό για γρήγορη ανάπτυξη εφαρμογών (rapid development), ενώ παράλληλα παρέχει τις βασικές λειτουργίες που απαιτούνται για τη δημιουργία ισχυρών και αξιόπιστων εφαρμογών.

Το Flask είναι ελαφρύ και δεν περιλαμβάνει περιττές λειτουργίες, επιτρέποντας στους προγραμματιστές να προσθέσουν μόνο ό,τι χρειάζονται. Η σύνταξη και η δομή του Flask είναι απλή και εύκολη στην κατανόηση, καθιστώντας το ιδανικό για αρχάριους και έμπειρους προγραμματιστές. Το Flask μπορεί να ενσωματωθεί εύκολα με άλλες βιβλιοθήκες και εργαλεία, όπως η SQLAlchemy για διαχείριση βάσεων δεδομένων και η Jinja2 για τη δημιουργία templates. Υπάρχει μια μεγάλη κοινότητα προγραμματιστών που χρησιμοποιεί το Flask, παρέχοντας άφθονο υλικό, οδηγούς και υποστήριξη.

Το Flask, παρά την απλότητά του, προσφέρει πολλές δυνατότητες που το καθιστούν ισχυρό και ευέλικτο για την ανάπτυξη web εφαρμογών και APIs.

Το Flask παρέχει έναν απλό και εύκολο τρόπο για τον καθορισμό ρουτών και τη διαχείριση αιτημάτων HTTP. Οι προγραμματιστές μπορούν να δημιουργήσουν ρουτες για GET, POST, PUT και DELETE αιτήματα χρησιμοποιώντας διακοσμητές (decorators).

```
from flask import Flask, request


app = Flask(__name__)


@app.route('/')
def home():
    return 'Hello, World!'


@app.route('/submit', methods=['POST'])
def submit():
    data = request.form['data']
    return f'Data received: {data}'
```

Το Flask χρησιμοποιεί τη μηχανή templates Jinja2, η οποία επιτρέπει τη δημιουργία δυναμικών HTML σελίδων. Οι προγραμματιστές μπορούν να χρησιμοποιήσουν templates για να διαχωρίσουν την λογική της εφαρμογής από την παρουσίαση.

```
<!-- templates/home.html -->
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Home</title>
</head>
<body>
    <h1>{{ title }}</h1>
    <p>Welcome to Flask!</p>
</body>
</html>
```

Το Flask παρέχει εύκολες μεθόδους για τη διαχείριση συνεδριών (sessions) και cookies, επιτρέποντας στους προγραμματιστές να αποθηκεύουν πληροφορίες χρήστη ανάμεσα στις αιτήσεις HTTP.

from flask import session

```
app.secret_key = 'supersecretkey'


@app.route('/set_session')
def set_session():
    session['username'] = 'john_doe'
    return 'Session set!'


@app.route('/get_session')
def get_session():
    username = session.get('username')
```

```
    return f'Logged in as: {username}'
```

Το Flask μπορεί να ενσωματωθεί με τη βιβλιοθήκη SQLAlchemy για την αλληλεπίδραση με βάσεις δεδομένων. Το Flask-SQLAlchemy είναι μια επέκταση που διευκολύνει τη χρήση της SQLAlchemy με το Flask.

```
from flask_sqlalchemy import SQLAlchemy


app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///app.db'
db = SQLAlchemy(app)


class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)


    def __repr__(self):
        return f'<User {self.username}>'


@app.route('/add_user')
def add_user():
    user = User(username='john_doe')
    db.session.add(user)
    db.session.commit()
    return 'User added!'
```

Το Flask παρέχει εργαλεία για την ασφάλεια των web εφαρμογών, όπως την προστασία από CSRF (Cross-Site Request Forgery) επιθέσεις και τη χρήση εργαλείων για την επαλήθευση των χρηστών.

```
from flask_wtf.csrf import CSRFProtect


```

```
csrf = CSRFProtect(app)
```

Υπάρχει πληθώρα επεκτάσεων για το Flask που προσθέτουν επιπλέον λειτουργίες και διευκολύνουν την ανάπτυξη εφαρμογών, όπως το Flask-RESTful για την ανάπτυξη APIs, το Flask-Login για τη διαχείριση συνδέσεων χρηστών και το Flask-Migrate για τις μεταναστεύσεις βάσεων δεδομένων.

Εγκατάσταση του Flask μέσω pip:

Ανοίξτε ένα τερματικό και εκτελέστε την ακόλουθη εντολή για να εγκαταστήσετε το Flask:

```
pip install Flask
```

Δημιουργήστε έναν νέο φάκελο για την εφαρμογή σας και δημιουργήστε ένα αρχείο app.py με τον παρακάτω κώδικα:

```
from flask import Flask


app = Flask(__name__)


@app.route('/')
def home():
    return 'Hello, Flask!'


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

Στο τερματικό, πλοηγηθείτε στον φάκελο της εφαρμογής και εκτελέστε την εφαρμογή με την ακόλουθη εντολή:

```
python app.py
```

3.3 Google Charts

Τα Google Charts είναι μια ισχυρή και ευέλικτη βιβλιοθήκη για τη δημιουργία διαδραστικών γραφημάτων και διαγραμμάτων που μπορούν να ενσωματωθούν εύκολα σε ιστοσελίδες και web εφαρμογές. Αναπτύχθηκε από την Google και προσφέρει μια μεγάλη ποικιλία τύπων γραφημάτων, από

απλά γραμμικά και ραβδογράμματα έως σύνθετα ιεραρχικά διαγράμματα και γεωγραφικά γραφήματα. Τα Google Charts χρησιμοποιούν HTML5/SVG για την απόδοση των γραφημάτων, εξασφαλίζοντας έτσι τη συμβατότητα με τους σύγχρονους web browsers χωρίς την ανάγκη για πρόσθετα [7].

Τα Google Charts χρησιμοποιούνται ευρέως σε διάφορες εφαρμογές και τομείς λόγω της ευκολίας χρήσης και των πλούσιων δυνατοτήτων τους. Τα γραφήματα είναι διαδραστικά, επιτρέποντας στους χρήστες να αλληλεπιδρούν με τα δεδομένα με κινήσεις όπως hover και click. Τα Google Charts είναι εύκολα στην ενσωμάτωση σε οποιαδήποτε ιστοσελίδα ή web εφαρμογή μέσω της χρήσης JavaScript.

Υπάρχει μεγάλη ποικιλία τύπων γραφημάτων, καλύπτοντας όλες τις ανάγκες οπτικοποίησης δεδομένων. Τα γραφήματα μπορούν να προσαρμοστούν πλήρως, από την εμφάνιση και τα χρώματα μέχρι τις ετικέτες και τους άξονες. Τα Google Charts μπορούν να διαχειριστούν μεγάλα σύνολα δεδομένων χωρίς προβλήματα απόδοσης.

Τα Google Charts προσφέρουν μια πλούσια συλλογή από δυνατότητες που επιτρέπουν την οπτικοποίηση δεδομένων με διάφορους τρόπους.

- Γραμμικά Γραφήματα (Line Charts): Ιδανικά για την προβολή τάσεων με την πάροδο του χρόνου.
- Ραβδογράμματα (Bar Charts): Χρησιμοποιούνται για τη σύγκριση ποσοτικών δεδομένων μεταξύ διαφορετικών κατηγοριών.
- Πίτες (Pie Charts): Καλύτερα για την προβολή μεριδίων της συνολικής ποσότητας.
- Διαγράμματα Περιοχής (Area Charts): Χρήσιμα για την προβολή της συσσώρευσης δεδομένων με την πάροδο του χρόνου.
- Διαγράμματα Σημείων (Scatter Charts): Ιδανικά για την αναγνώριση σχέσεων μεταξύ δύο συνόλων δεδομένων.

Τα γραφήματα μπορούν να ανταποκριθούν σε ενέργειες του χρήστη, όπως κλικ και hover, παρέχοντας επιπλέον πληροφορίες ή επισημάνσεις. Οι χρήστες μπορούν να προσαρμόσουν την εμφάνιση των γραφημάτων, όπως τα χρώματα, τις ετικέτες, τους άξονες και το μέγεθος, ώστε να ταιριάζουν με το στυλ της ιστοσελίδας τους. Τα δεδομένα μπορούν να προέρχονται από διάφορες πηγές, όπως JSON, CSV, και Google Sheets, κάνοντας εύκολη την ενσωμάτωση δυναμικών δεδομένων. Τα γραφήματα μπορούν να ενημερώνονται σε πραγματικό χρόνο, επιτρέποντας την παρακολούθηση δεδομένων όπως αυτά συλλέγονται και μεταδίδονται.

3.4 Datatables

Τα DataTables είναι μια ισχυρή jQuery plug-in βιβλιοθήκη που χρησιμοποιείται για την ενίσχυση των HTML πινάκων με πρόσθετες λειτουργίες και διαδραστικότητα. Αναπτύχθηκαν από τον Allan Jardine και είναι γνωστά για την ευκολία χρήσης και την ευελιξία τους. Τα DataTables παρέχουν δυνατότητες όπως ταξινόμηση, φιλτράρισμα, σελιδοποίηση και αναζήτηση σε δεδομένα πίνακα, καθιστώντας την εμφάνιση και τη διαχείριση μεγάλων συνόλων δεδομένων πολύ πιο εύκολη και αποτελεσματική.

Η χρησιμότητα των DataTables έγκειται στην ικανότητά τους να μετατρέπουν απλούς HTML πίνακες σε δυναμικά και διαδραστικά εργαλεία παρουσίασης δεδομένων. Τα DataTables μπορούν να ενσωματωθούν εύκολα σε οποιαδήποτε ιστοσελίδα που χρησιμοποιεί HTML και jQuery.

Παρέχουν δυνατότητες διαδραστικότητας, όπως ταξινόμηση, φιλτράρισμα και σελιδοποίηση δεδομένων σε πραγματικό χρόνο. Μπορούν να χειριστούν μεγάλα σύνολα δεδομένων χωρίς να επηρεάζεται η απόδοση, επιτρέποντας την ομαλή πλοήγηση και αναζήτηση στα δεδομένα. Προσφέρουν πολλές επιλογές προσαρμογής για την εμφάνιση και τη λειτουργικότητα των πινάκων, ώστε να ταιριάζουν στις ανάγκες του χρήστη. Οι προγραμματιστές μπορούν να επεκτείνουν τις λειτουργίες των DataTables με plugins και πρόσθετα [8].

Τα DataTables παρέχουν μια πλούσια συλλογή από δυνατότητες για τη διαχείριση και την παρουσίαση δεδομένων σε πίνακες.

- Ταξινόμηση Δεδομένων (Sorting)

  Τα DataTables επιτρέπουν την ταξινόμηση των δεδομένων σε έναν πίνακα κατά αύξουσα ή φθίνουσα σειρά με ένα απλό κλικ στην επικεφαλίδα της στήλης.

- Φιλτράρισμα Δεδομένων (Filtering)

  Οι χρήστες μπορούν να φιλτράρουν τα δεδομένα σε πραγματικό χρόνο χρησιμοποιώντας ένα πεδίο αναζήτησης που προστίθεται αυτόματα από τα DataTables.

3.5 Bootstrap

Το Bootstrap είναι ένα από τα πιο δημοφιλή πλαίσια ανάπτυξης (framework) για τη δημιουργία ευέλικτων και απόκριτων ιστοσελίδων και web εφαρμογών. Αναπτύχθηκε από την ομάδα του Twitter και κυκλοφόρησε ως ανοιχτό λογισμικό το 2011. Το Bootstrap παρέχει ένα ολοκληρωμένο σύνολο εργαλείων και βιβλιοθηκών CSS και JavaScript που επιτρέπουν στους προγραμματιστές να δημιουργούν κομψές και λειτουργικές ιστοσελίδες γρήγορα και εύκολα. Η ευελιξία του Bootstrap και η συμβατότητά του με όλους τους σύγχρονους browsers το καθιστούν ιδανική επιλογή για την ανάπτυξη σύγχρονων web εφαρμογών [9].

Το Bootstrap είναι ιδιαίτερα χρήσιμο για την ανάπτυξη ιστοσελίδων και web εφαρμογές. Παρέχει ένα πλέγμα (grid system) που προσαρμόζεται δυναμικά στο μέγεθος της οθόνης της συσκευής, εξασφαλίζοντας ότι η ιστοσελίδα θα φαίνεται καλά τόσο σε επιτραπέζιους υπολογιστές όσο και σε κινητές συσκευές. Το Bootstrap περιλαμβάνει ένα σύνολο έτοιμων προς χρήση στοιχείων και στυλ που μπορούν να εφαρμοστούν εύκολα, μειώνοντας τον χρόνο ανάπτυξης. Παρέχει έναν συνεπή τρόπο για το σχεδιασμό και την ανάπτυξη ιστοσελίδων, διασφαλίζοντας ότι όλα τα στοιχεία έχουν την ίδια εμφάνιση και αίσθηση. Το Bootstrap έχει εκτενή τεκμηρίωση και μια μεγάλη κοινότητα υποστήριξης που προσφέρει οδηγίες, παραδείγματα και λύσεις σε κοινά προβλήματα. Οι προγραμματιστές μπορούν να προσαρμόσουν και να επεκτείνουν το Bootstrap για να καλύψουν τις ειδικές ανάγκες της εφαρμογής τους.

3.6 MySQL

Η MySQL είναι ένα από τα πιο δημοφιλή συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS) που χρησιμοποιούνται παγκοσμίως. Αναπτύχθηκε αρχικά από τη MySQL AB, μια Σουηδική εταιρεία, και κυκλοφόρησε για πρώτη φορά το 1995. Η MySQL είναι ανοιχτού κώδικα και διατίθεται υπό την άδεια GPL (General Public License), καθιστώντας την προσιτή και ευέλικτη για πολλές εφαρμογές και περιβάλλοντα ανάπτυξης. Η ευρεία υποστήριξη και η κοινότητα της MySQL την καθιστούν μια σταθερή επιλογή για την αποθήκευση και διαχείριση δεδομένων.

Η MySQL είναι ευρέως χρήσιμη σε πολλές εφαρμογές και τομείς λόγω της ευελιξίας και των δυνατοτήτων της. Χρησιμοποιείται σε διαφόρους τομείς όπως οι διαδικτυακές εφαρμογές, τα e-commerce sites, τα συστήματα διαχείρισης περιεχομένου (CMS) και πολλά άλλα. Οι κυριότεροι λόγοι για τη δημοφιλία της MySQL περιλαμβάνουν την υψηλή απόδοση, την αξιοπιστία και τη δυνατότητα κλιμάκωσης, που την καθιστούν ιδανική για μικρές και μεγάλες επιχειρήσεις.

Η MySQL προσφέρει ένα πλούσιο σύνολο δυνατοτήτων που επιτρέπουν στους προγραμματιστές και τους διαχειριστές βάσεων δεδομένων να διαχειρίζονται και να επεξεργάζονται δεδομένα με αποτελεσματικό τρόπο. Οι δυνατότητες της MySQL περιλαμβάνουν υποστήριξη για διάφορους τύπους δεδομένων, πολλαπλά indexes, συναλλαγές (transactions) και δυνατότητες αποθήκευσης διαδικασιών (stored procedures). Επιπλέον, η MySQL υποστηρίζει επεκτάσιμα plugins που επιτρέπουν την προσθήκη νέων λειτουργιών και βελτιώσεων.

Η MySQL χρησιμοποιείται σε ένα ευρύ φάσμα εφαρμογών και βιομηχανιών. Στις διαδικτυακές εφαρμογές, η MySQL είναι η βάση δεδομένων πίσω από πολλές δημοφιλείς πλατφόρμες και ιστότοπους, όπως το Facebook, το Twitter και το YouTube. Στον τομέα του e-commerce, η MySQL χρησιμοποιείται για τη διαχείριση καταλόγων προϊόντων, παραγγελιών και συναλλαγών. Επιπλέον, πολλά συστήματα διαχείρισης περιεχομένου, όπως το WordPress, το Joomla και το Drupal, βασίζονται στη MySQL για την αποθήκευση και ανάκτηση δεδομένων. Η MySQL υποστηρίζεται από μια μεγάλη και ενεργή κοινότητα χρηστών και προγραμματιστών που συμβάλλουν στη συνεχή βελτίωση και

εξέλιξή της. Επιπλέον, η Oracle Corporation, που απέκτησε τη MySQL το 2010, παρέχει εμπορική υποστήριξη και υπηρεσίες για επιχειρηματικές εφαρμογές. Η συνεχής ανάπτυξη της MySQL εξασφαλίζει ότι παραμένει συμβατή με τις σύγχρονες τεχνολογίες και απαιτήσεις, προσφέροντας νέες δυνατότητες και βελτιώσεις με κάθε νέα έκδοση.

3.7 SqlAlchemy

Το SQLAlchemy είναι μια ισχυρή βιβλιοθήκη SQL toolkit και Object Relational Mapping (ORM) για την Python, που αναπτύχθηκε από τον Michael Bayer και κυκλοφόρησε για πρώτη φορά το 2005. Το SQLAlchemy παρέχει εργαλεία για την αλληλεπίδραση με βάσεις δεδομένων χρησιμοποιώντας αντικειμενοστραφή προσέγγιση, επιτρέποντας στους προγραμματιστές να γράφουν κώδικα που είναι ανεξάρτητος από τη συγκεκριμένη βάση δεδομένων που χρησιμοποιείται. Αυτή η βιβλιοθήκη προσφέρει έναν ευέλικτο και εκτεταμένο τρόπο για τη διαχείριση των δεδομένων, τη σύνταξη ερωτημάτων SQL και τη διαχείριση συναλλαγών, διασφαλίζοντας την καλύτερη απόδοση και κλιμακωσιμότητα της εφαρμογής.

Το SQLAlchemy είναι ιδιαίτερα χρήσιμο για προγραμματιστές που επιθυμούν να χρησιμοποιήσουν μια ORM προσέγγιση για την ανάπτυξη εφαρμογών με Python. Η βιβλιοθήκη προσφέρει πλήρη υποστήριξη για SQLAlchemy Core και ORM, επιτρέποντας την εύκολη και αποδοτική διαχείριση των σχέσεων μεταξύ των αντικειμένων και των αντίστοιχων εγγραφών της βάσης δεδομένων. Το SQLAlchemy υποστηρίζει πολλαπλές βάσεις δεδομένων, όπως MySQL, PostgreSQL, SQLite, και άλλα, προσφέροντας έναν ομοιόμορφο τρόπο για την αλληλεπίδραση με διαφορετικά συστήματα διαχείρισης βάσεων δεδομένων (DBMS). Επιπλέον, οι δυνατότητες επέκτασης και προσαρμογής του SQLAlchemy επιτρέπουν στους προγραμματιστές να δημιουργούν σύνθετες και αποδοτικές εφαρμογές, ενώ η ενεργή κοινότητα και η εκτεταμένη τεκμηρίωση προσφέρουν άφθονη υποστήριξη και πόρους για την ανάπτυξη.

Κεφάλαιο 4ο: Το έργο

4.1 Περιγραφή του έργου

Το σύστημα IoTDev είναι μια ολοκληρωμένη πλατφόρμα που επιτρέπει στους χρήστες να διαχειρίζονται συσκευές με δυνατότητα Διαδικτύου (IoT), όπως το ESP32 ή το Raspberry Pi. Μέσω του IoTDev, οι χρήστες μπορούν να στέλνουν δεδομένα από τους αισθητήρες των συσκευών τους στον server και να τα αποθηκεύουν σε μια βάση δεδομένων MySQL. Στη συνέχεια, τα δεδομένα αυτά μπορούν να προβληθούν σε μορφή γραφημάτων χρησιμοποιώντας Google Charts όταν ο χρήστης συνδεθεί στην ιστοσελίδα.

Η πλατφόρμα είναι υλοποιημένη με χρήση του Flask, ενός δημοφιλούς micro-framework για την ανάπτυξη web εφαρμογών με τη γλώσσα προγραμματισμού Python. Το frontend της εφαρμογής βασίζεται στο Bootstrap για την επίτευξη μοντέρνας και responsive σχεδίασης, ενώ η βιβλιοθήκη DataTables χρησιμοποιείται για την εύκολη και διαδραστική εμφάνιση των δεδομένων σε πίνακες.

Το IoTDev προσφέρει ένα API που διευκολύνει την πρόσβαση, την ανάκτηση και την καταγραφή δεδομένων. Οι χρήστες μπορούν να επικοινωνούν με τις συσκευές τους μέσω συγκεκριμένων endpoints του API για να γράφουν και να διαβάζουν δεδομένα από τη βάση.

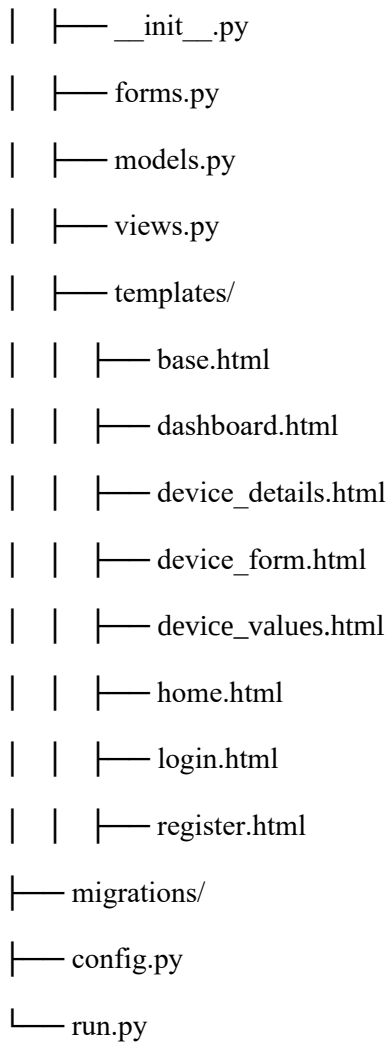
Το σύστημα περιλαμβάνει επίσης λειτουργίες όπως:

- Εγγραφή και είσοδος χρηστών.
- Προσθήκη, προβολή, επεξεργασία και διαγραφή συσκευών.
- Προβολή των τιμών των αισθητήρων των συσκευών σε πραγματικό χρόνο μέσω γραφημάτων.
- Διαχείριση API keys για την ασφάλεια των δεδομένων.
- Η πλατφόρμα IoTDev είναι ιδανική για εφαρμογές παρακολούθησης και ανάλυσης δεδομένων από IoT συσκευές, προσφέροντας μια εύχρηστη και αποτελεσματική λύση για την αποθήκευση και προβολή δεδομένων.

iotdev/

│

├── app/



Εικόνα 4.1: Δομή του έργου

Ο κύριος φάκελος του έργου περιέχει τα βασικά αρχεία και φακέλους που απαιτούνται για την ανάπτυξη και εκτέλεση της εφαρμογής IoTDev.

Φάκελος app/

Ο φάκελος app/ περιέχει τον κύριο κώδικα της εφαρμογής. Περιλαμβάνει τα αρχεία που είναι υπεύθυνα για την αρχικοποίηση της εφαρμογής, τις φόρμες, τα μοντέλα, τους ελεγκτές και τα templates.

__init__.py: Αρχικοποιεί την εφαρμογή Flask και τις επεκτάσεις της, όπως η βάση δεδομένων SQLAlchemy, το σύστημα διαχείρισης μεταναστεύσεων (migrations), το Bootstrap για το frontend και το σύστημα διαχείρισης συνδέσεων (login manager).

forms.py: Περιέχει τις φόρμες της εφαρμογής, οι οποίες χρησιμοποιούνται για τη συλλογή δεδομένων από τους χρήστες. Περιλαμβάνει τις φόρμες σύνδεσης, εγγραφής και διαχείρισης συσκευών.

models.py: Περιέχει τα μοντέλα της βάσης δεδομένων, τα οποία χρησιμοποιούνται για την αποθήκευση των δεδομένων. Περιλαμβάνει τα μοντέλα για τους χρήστες, τις συσκευές, τις τιμές των πεδίων των συσκευών και τα API keys.

views.py: Περιέχει τους ελεγκτές (controllers) που διαχειρίζονται τα αιτήματα από τους χρήστες και αποστέλλουν δεδομένα στα templates. Περιλαμβάνει τις λειτουργίες εγγραφής, σύνδεσης, αποσύνδεσης, προσθήκης, επεξεργασίας, διαγραφής συσκευών και διαχείρισης των API endpoints.

Φάκελος templates/: Περιέχει τα HTML templates που χρησιμοποιούνται για την απόδοση των σελίδων της εφαρμογής.

base.html: Το κύριο template που περιέχει τη βασική δομή της σελίδας και τα κοινά στοιχεία, όπως το μενού πλοήγησης.

dashboard.html: Το template για την κεντρική σελίδα του πίνακα ελέγχου, όπου εμφανίζονται οι συσκευές του χρήστη.

device_details.html: Το template για την προβολή των λεπτομερειών μιας συσκευής.

device_form.html: Το template για την προσθήκη και επεξεργασία συσκευών.

device_values.html: Το template για την προβολή των τιμών των πεδίων μιας συσκευής σε μορφή γραφημάτων.

home.html: Το template για την αρχική σελίδα της εφαρμογής.

login.html: Το template για τη σελίδα σύνδεσης.

register.html: Το template για τη σελίδα εγγραφής.

Φάκελος migrations/

Περιέχει τα αρχεία και τις πληροφορίες που απαιτούνται για τη διαχείριση των μεταναστεύσεων της βάσης δεδομένων (migrations) χρησιμοποιώντας το Flask-Migrate.

config.py

Περιέχει τις ρυθμίσεις της εφαρμογής, όπως η διαμόρφωση της βάσης δεδομένων και άλλες παραμέτρους που απαιτούνται για την εκτέλεση της εφαρμογής.

run.py

Το κύριο αρχείο εκκίνησης της εφαρμογής. Περιέχει τον κώδικα που απαιτείται για την εκτέλεση της εφαρμογής Flask.

Με αυτή τη δομή, η εφαρμογή IoTDev είναι οργανωμένη και έτοιμη να παρέχει όλες τις λειτουργίες που απαιτούνται για τη διαχείριση IoT συσκευών, τη συλλογή και προβολή δεδομένων και τη διαχείριση των χρηστών.

4.2 Περιγραφή των αρχείων του έργου

Ας ξεκινήσουμε από την περιγραφή των αρχείων για την υλοποίηση του έργου.

Το αρχείο __init__.py είναι υπεύθυνο για την αρχικοποίηση της εφαρμογής Flask και τη διαμόρφωση των διάφορων επεκτάσεων που χρησιμοποιούνται.

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_migrate import Migrate
from flask_bootstrap import Bootstrap
from flask_wtf.csrf import CSRFProtect
from flask_login import LoginManager

app = Flask(__name__)  # Δημιουργία της εφαρμογής Flask
app.config.from_object('config.Config')  # Φόρτωση των ρυθμίσεων από το αρχείο config

db = SQLAlchemy(app)  # Αρχικοποίηση της βάσης δεδομένων
migrate = Migrate(app, db)  # Αρχικοποίηση του Migrate για τη διαχείριση των migrations
bootstrap = Bootstrap(app)  # Αρχικοποίηση του Bootstrap για το frontend
```

```
csrf = CSRFProtect(app)  # Προστασία από CSRF επιθέσεις


login_manager = LoginManager()  # Δημιουργία του LoginManager για τη διαχείριση των συνδέσεων
login_manager.init_app(app)  # Σύνδεση του LoginManager με την εφαρμογή
login_manager.login_view = 'login'  # Ορισμός της σελίδας σύνδεσης


from app import views, models  # Εισαγωγή των views και models


@login_manager.user_loader
def load_user(user_id):
    return models.User.query.get(int(user_id))  # Φόρτωση του χρήστη με βάση το ID


if __name__ == "__main__":
    app.run(debug=True)  # Εκκίνηση της εφαρμογής σε λειτουργία ανάπτυξης
```

forms.py

Το αρχείο forms.py περιέχει τις φόρμες της εφαρμογής, που χρησιμοποιούνται για τη συλλογή δεδομένων από τους χρήστες.

```
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, BooleanField, SubmitField
from wtforms.validators import DataRequired, Email, EqualTo


class LoginForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Email()])  # Πεδίο email με validators
    password = PasswordField('Password', validators=[DataRequired()])  # Πεδίο password με validators
    submit = SubmitField('Login')  # Κουμπί υποβολής
```

```
class RegisterForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Email()])  # Πεδίο email με validators
    password = PasswordField('Password', validators=[DataRequired(), EqualTo('confirm', message='Passwords must match')])  # Πεδίο password με validators
    confirm = PasswordField('Repeat Password')  # Πεδίο επιβεβαίωσης password
    firstname = StringField('First Name', validators=[DataRequired()])  # Πεδίο πρώτου ονόματος με validator
    lastname = StringField('Last Name', validators=[DataRequired()])  # Πεδίο επωνύμου με validator
    submit = SubmitField('Register')  # Κουμπί υποβολής


class DeviceForm(FlaskForm):
    nameofdevice = StringField('Device Name', validators=[DataRequired()])  # Πεδίο ονόματος συσκευής με validator
    description = StringField('Description')  # Πεδίο περιγραφής
    field1_check = BooleanField('Field 1')  # Checkbox για το πεδίο 1
    field1_name = StringField('Field 1 Name')  # Όνομα πεδίου 1
    field2_check = BooleanField('Field 2')  # Checkbox για το πεδίο 2
    field2_name = StringField('Field 2 Name')  # Όνομα πεδίου 2
    field3_check = BooleanField('Field 3')  # Checkbox για το πεδίο 3
    field3_name = StringField('Field 3 Name')  # Όνομα πεδίου 3
    field4_check = BooleanField('Field 4')  # Checkbox για το πεδίο 4
    field4_name = StringField('Field 4 Name')  # Όνομα πεδίου 4
    field5_check = BooleanField('Field 5')  # Checkbox για το πεδίο 5
    field5_name = StringField('Field 5 Name')  # Όνομα πεδίου 5
    field6_check = BooleanField('Field 6')  # Checkbox για το πεδίο 6
    field6_name = StringField('Field 6 Name')  # Όνομα πεδίου 6
    submit = SubmitField('Save')  # Κουμπί υποβολής
```

models.py

Το αρχείο models.py περιέχει τα μοντέλα της βάσης δεδομένων, τα οποία χρησιμοποιούνται για την αποθήκευση των δεδομένων.

```
from app import db
from datetime import datetime
from werkzeug.security import generate_password_hash, check_password_hash
from flask_login import UserMixin


class User(UserMixin, db.Model):
    __tablename__ = 'user'
    id = db.Column(db.Integer, primary_key=True)  # Πρωτεύον κλειδί
    email = db.Column(db.String(255), unique=True, nullable=False)  # Μοναδικό πεδίο email
    password_hash = db.Column(db.String(255), nullable=False)  # Πεδίο password hash
    firstname = db.Column(db.String(255), nullable=False)  # Πεδίο πρώτου ονόματος
    lastname = db.Column(db.String(255), nullable=False)  # Πεδίο επωνύμου
    created_at = db.Column(db.DateTime, default=datetime.utcnow)  # Ημερομηνία δημιουργίας
    updated_at = db.Column(db.DateTime, default=datetime.utcnow, onupdate=datetime.utcnow)  # Ημερομηνία ενημέρωσης


    def set_password(self, password):
        self.password_hash = generate_password_hash(password)  # Δημιουργία του hash του password


    def check_password(self, password):
        return check_password_hash(self.password_hash, password)  # Έλεγχος του password με το hash


class Device(db.Model):
    __tablename__ = 'device'
```

```
    id = db.Column(db.Integer, primary_key=True)  # Πρωτεύον κλειδί
    userid = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)  # Ξένο κλειδί προς τον πίνακα user
    nameofdevice = db.Column(db.String(50), nullable=False)  # Πεδίο ονόματος συσκευής
    description = db.Column(db.String(255), nullable=True)  # Πεδίο περιγραφής
    field1_check = db.Column(db.Boolean, default=False)  # Checkbox για το πεδίο 1
    field1_name = db.Column(db.String(50), nullable=True)  # Όνομα πεδίου 1
    field2_check = db.Column(db.Boolean, default=False)  # Checkbox για το πεδίο 2
    field2_name = db.Column(db.String(50), nullable=True)  # Όνομα πεδίου 2
    field3_check = db.Column(db.Boolean, default=False)  # Checkbox για το πεδίο 3
    field3_name = db.Column(db.String(50), nullable=True)  # Όνομα πεδίου 3
    field4_check = db.Column(db.Boolean, default=False)  # Checkbox για το πεδίο 4
    field4_name = db.Column(db.String(50), nullable=True)  # Όνομα πεδίου 4
    field5_check = db.Column(db.Boolean, default=False)  # Checkbox για το πεδίο 5
    field5_name = db.Column(db.String(50), nullable=True)  # Όνομα πεδίου 5
    field6_check = db.Column(db.Boolean, default=False)  # Checkbox για το πεδίο 6
    field6_name = db.Column(db.String(50), nullable=True)  # Όνομα πεδίου 6
    private = db.Column(db.Boolean, default=True)  # Πεδίο για την ιδιωτικότητα της συσκευής
    created_at = db.Column(db.DateTime, default=datetime.utcnow)  # Ημερομηνία δημιουργίας
    updated_at = db.Column(db.DateTime, default=datetime.utcnow, onupdate=datetime.utcnow)  # Ημερομηνία ενημέρωσης


class ValueFields(db.Model):
    __tablename__ = 'valuefields'
    id = db.Column(db.Integer, primary_key=True)  # Πρωτεύον κλειδί
    deviceid = db.Column(db.Integer, db.ForeignKey('device.id'), nullable=False)  # Ξένο κλειδί προς τον πίνακα device
    field1_value = db.Column(db.Float, nullable=True)  # Τιμή πεδίου 1
    field2_value = db.Column(db.Float, nullable=True)  # Τιμή πεδίου 2
```

```
    field3_value = db.Column(db.Float, nullable=True)  # Τιμή πεδίου 3
    field4_value = db.Column(db.Float, nullable=True)  # Τιμή πεδίου 4
    field5_value = db.Column(db.Float, nullable=True)  # Τιμή πεδίου 5
    field6_value = db.Column(db.Float, nullable=True)  # Τιμή πεδίου 6
    created_at = db.Column(db.DateTime, default=datetime.utcnow)  # Ημερομηνία δημιουργίας


class APIKeys(db.Model):
    __tablename__ = 'api_keys'
    id = db.Column(db.Integer, primary_key=True)  # Πρωτεύον κλειδί
    deviceid = db.Column(db.Integer, db.ForeignKey('device.id'), nullable=False)  # Ξένο κλειδί προς τον πίνακα device
    write_key = db.Column(db.String(255), nullable=False)  # Κλειδί για εγγραφή
    read_key = db.Column(db.String(255), nullable=False)  # Κλειδί για ανάγνωση
    created_at = db.Column(db.DateTime, default=datetime.utcnow)  # Ημερομηνία δημιουργίας
```

views.py

Το αρχείο views.py περιέχει τους ελεγκτές (controllers) που διαχειρίζονται τα αιτήματα και αποστέλλουν δεδομένα στα templates.

```
from flask import render_template, redirect, url_for, flash, request, jsonify
from flask_login import login_user, logout_user, current_user, login_required
from app import app, db
from app.models import User, Device, ValueFields, APIKeys
from app.forms import LoginForm, RegisterForm, DeviceForm
import random
import string


def generate_key(length=16):
```

```
    return ''.join(random.choices(string.ascii_letters + string.digits, k=length))  # Δημιουργία τυχαίου κλειδιού


@app.route('/')
def home():
    return render_template('home.html')  # Επιστροφή της σελίδας home


@app.route('/register', methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('dashboard'))  # Αν ο χρήστης είναι συνδεδεμένος, ανακατεύθυνση στο dashboard
    form = RegisterForm()  # Δημιουργία του αντικειμένου φόρμας
    if form.validate_on_submit():
        user = User(email=form.email.data, firstname=form.firstname.data, lastname=form.lastname.data)  # Δημιουργία χρήστη
        user.set_password(form.password.data)  # Ορισμός του password
        db.session.add(user)  # Προσθήκη του χρήστη στη βάση δεδομένων
        db.session.commit()  # Επιβεβαίωση της αλλαγής στη βάση δεδομένων
        flash('Congratulations, you are now a registered user!')  # Εμφάνιση μηνύματος επιτυχίας
        return redirect(url_for('login'))  # Ανακατεύθυνση στη σελίδα login
    return render_template('register.html', form=form)  # Επιστροφή της σελίδας register με τη φόρμα


@app.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('dashboard'))  # Αν ο χρήστης είναι συνδεδεμένος, ανακατεύθυνση στο dashboard
    form = LoginForm()  # Δημιουργία του αντικειμένου φόρμας
```

```
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()  # Αναζήτηση του χρήστη με βάση το email
        if user is None or not user.check_password(form.password.data):
            flash('Invalid username or password')  # Εμφάνιση μηνύματος σφάλματος
            return redirect(url_for('login'))  # Ανακατεύθυνση στη σελίδα login
        login_user(user)  # Σύνδεση του χρήστη
        return redirect(url_for('dashboard'))  # Ανακατεύθυνση στο dashboard
    return render_template('login.html', form=form)  # Επιστροφή της σελίδας login με τη φόρμα

@app.route('/logout')
def logout():
    logout_user()  # Αποσύνδεση του χρήστη
    return redirect(url_for('home'))  # Ανακατεύθυνση στη σελίδα home

@app.route('/dashboard')
@login_required
def dashboard():
    devices = Device.query.filter_by(userid=current_user.id).all()  # Αναζήτηση των συσκευών του χρήστη
    return render_template('dashboard.html', devices=devices)  # Επιστροφή της σελίδας dashboard με τις συσκευές

@app.route('/device/add', methods=['GET', 'POST'])
@login_required
def add_device():
    form = DeviceForm()  # Δημιουργία του αντικειμένου φόρμας
    if form.validate_on_submit():
```

```
    device = Device(userid=current_user.id, nameofdevice=form.nameofdevice.data, description=form.description.data,
                    field1_check=form.field1_check.data, field1_name=form.field1_name.data,
                    field2_check=form.field2_check.data, field2_name=form.field2_name.data,
                    field3_check=form.field3_check.data, field3_name=form.field3_name.data,
                    field4_check=form.field4_check.data, field4_name=form.field4_name.data,
                    field5_check=form.field5_check.data, field5_name=form.field5_name.data,
                    field6_check=form.field6_check.data, field6_name=form.field6_name.data) # Δημιουργία συσκευής
        db.session.add(device)  # Προσθήκη της συσκευής στη βάση δεδομένων
        db.session.commit()  # Επιβεβαίωση της αλλαγής στη βάση δεδομένων
        apikeys = APIKeys(deviceid=device.id, write_key=generate_key(), read_key=generate_key())  # Δημιουργία API keys
        db.session.add(apikeys)  # Προσθήκη των API keys στη βάση δεδομένων
        db.session.commit()  # Επιβεβαίωση της αλλαγής στη βάση δεδομένων
        return redirect(url_for('dashboard'))  # Ανακατεύθυνση στο dashboard
    return render_template('device_form.html', form=form)  # Επιστροφή της σελίδας προσθήκης συσκευής με τη φόρμα


@app.route('/device/edit/<int:device_id>', methods=['GET', 'POST'])
@login_required
def edit_device(device_id):
    device = Device.query.get_or_404(device_id)  # Αναζήτηση της συσκευής με βάση το ID
    apikeys = APIKeys.query.filter_by(deviceid=device.id).first()  # Αναζήτηση των API keys της συσκευής
    form = DeviceForm(obj=device)  # Δημιουργία του αντικειμένου φόρμας με τα δεδομένα της συσκευής
    if form.validate_on_submit():
        device.nameofdevice = form.nameofdevice.data  # Ενημέρωση του ονόματος της συσκευής
```

```
        device.description = form.description.data  # Ενημέρωση της περιγραφής της συσκευής
        device.field1_check = form.field1_check.data  # Ενημέρωση του πεδίου 1
        device.field1_name = form.field1_name.data  # Ενημέρωση του ονόματος πεδίου 1
        device.field2_check = form.field2_check.data  # Ενημέρωση του πεδίου 2
        device.field2_name = form.field2_name.data  # Ενημέρωση του ονόματος πεδίου 2
        device.field3_check = form.field3_check.data  # Ενημέρωση του πεδίου 3
        device.field3_name = form.field3_name.data  # Ενημέρωση του ονόματος πεδίου 3
        device.field4_check = form.field4_check.data  # Ενημέρωση του πεδίου 4
        device.field4_name = form.field4_name.data  # Ενημέρωση του ονόματος πεδίου 4
        device.field5_check = form.field5_check.data  # Ενημέρωση του πεδίου 5
        device.field5_name = form.field5_name.data  # Ενημέρωση του ονόματος πεδίου 5
        device.field6_check = form.field6_check.data  # Ενημέρωση του πεδίου 6
        device.field6_name = form.field6_name.data  # Ενημέρωση του ονόματος πεδίου 6
        db.session.commit()  # Επιβεβαίωση της αλλαγής στη βάση δεδομένων
        return redirect(url_for('dashboard'))  # Ανακατεύθυνση στο dashboard
    return render_template('device_form.html', form=form, apikeys=apikeys)  # Επιστροφή της σελίδας επεξεργασίας συσκευής με τη φόρμα


@app.route('/device/delete/<int:device_id>', methods=['POST'])
@login_required
def delete_device(device_id):
    device = Device.query.get_or_404(device_id)  # Αναζήτηση της συσκευής με βάση το ID
    db.session.delete(device)  # Διαγραφή της συσκευής από τη βάση δεδομένων
    db.session.commit()  # Επιβεβαίωση της αλλαγής στη βάση δεδομένων
    return redirect(url_for('dashboard'))  # Ανακατεύθυνση στο dashboard


@app.route('/device/<int:device_id>')
@login_required
```

```
def device_details(device_id):
    device = Device.query.get_or_404(device_id)  # Αναζήτηση της συσκευής με βάση το ID
    apikeys = APIKeys.query.filter_by(deviceid=device.id).first()  # Αναζήτηση των API keys της συσκευής
    valuefields = ValueFields.query.filter_by(deviceid=device.id).all()  # Αναζήτηση των τιμών των πεδίων της συσκευής
    return render_template('device_details.html', device=device, apikeys=apikeys, valuefields=valuefields)  # Επιστροφή της σελίδας λεπτομερειών της συσκευής με τα δεδομένα


@app.route('/api/device/update', methods=['GET'])
def api_device_update():
    write_key = request.args.get('apikey')  # Λήψη του API key για εγγραφή
    apikeys = APIKeys.query.filter_by(write_key=write_key).first()  # Αναζήτηση των API keys με το write_key
    if not apikeys:
        return jsonify({'error': 'Invalid API key'}), 400  # Επιστροφή σφάλματος αν το API key είναι άκυρο


    device = Device.query.get(apikeys.deviceid)  # Αναζήτηση της συσκευής με βάση το ID των API keys
    if not device:
        return jsonify({'error': 'Device not found'}), 404  # Επιστροφή σφάλματος αν η συσκευή δεν βρεθεί


    values = ValueFields(deviceid=device.id)  # Δημιουργία νέου αντικειμένου ValueFields
    if device.field1_check:
        values.field1_value = request.args.get('field1')  # Ενημέρωση της τιμής του πεδίου 1
    if device.field2_check:
        values.field2_value = request.args.get('field2')  # Ενημέρωση της τιμής του πεδίου 2
    if device.field3_check:
        values.field3_value = request.args.get('field3')  # Ενημέρωση της τιμής του πεδίου 3
```

```
    if device.field4_check:
        values.field4_value = request.args.get('field4')  # Ενημέρωση της τιμής του πεδίου 4
    if device.field5_check:
        values.field5_value = request.args.get('field5')  # Ενημέρωση της τιμής του πεδίου 5
    if device.field6_check:
        values.field6_value = request.args.get('field6')  # Ενημέρωση της τιμής του πεδίου 6

    db.session.add(values)  # Προσθήκη των τιμών στη βάση δεδομένων
    db.session.commit()  # Επιβεβαίωση της αλλαγής στη βάση δεδομένων
    return jsonify({'success': 'Data updated successfully'})  # Επιστροφή μηνύματος επιτυχίας

@app.route('/api/device/get', methods=['GET'])
def api_device_get():
    read_key = request.args.get('apikey')  # Λήψη του API key για ανάγνωση
    results = int(request.args.get('results', 10))  # Λήψη του αριθμού των αποτελεσμάτων
    apikeys = APIKeys.query.filter_by(read_key=read_key).first()  # Αναζήτηση των API keys με το read_key
    if not apikeys:
        return jsonify({'error': 'Invalid API key'}), 400  # Επιστροφή σφάλματος αν το API key είναι άκυρο

    device = Device.query.get(apikeys.deviceid)  # Αναζήτηση της συσκευής με βάση το ID των API keys
    if not device:
        return jsonify({'error': 'Device not found'}), 404  # Επιστροφή σφάλματος αν η συσκευή δεν βρεθεί

    valuefields = ValueFields.query.filter_by(deviceid=device.id).order_by(ValueFields.created_at.desc()).limit(results).all()  # Αναζήτηση των τιμών των πεδίων της συσκευής
    data = []
```

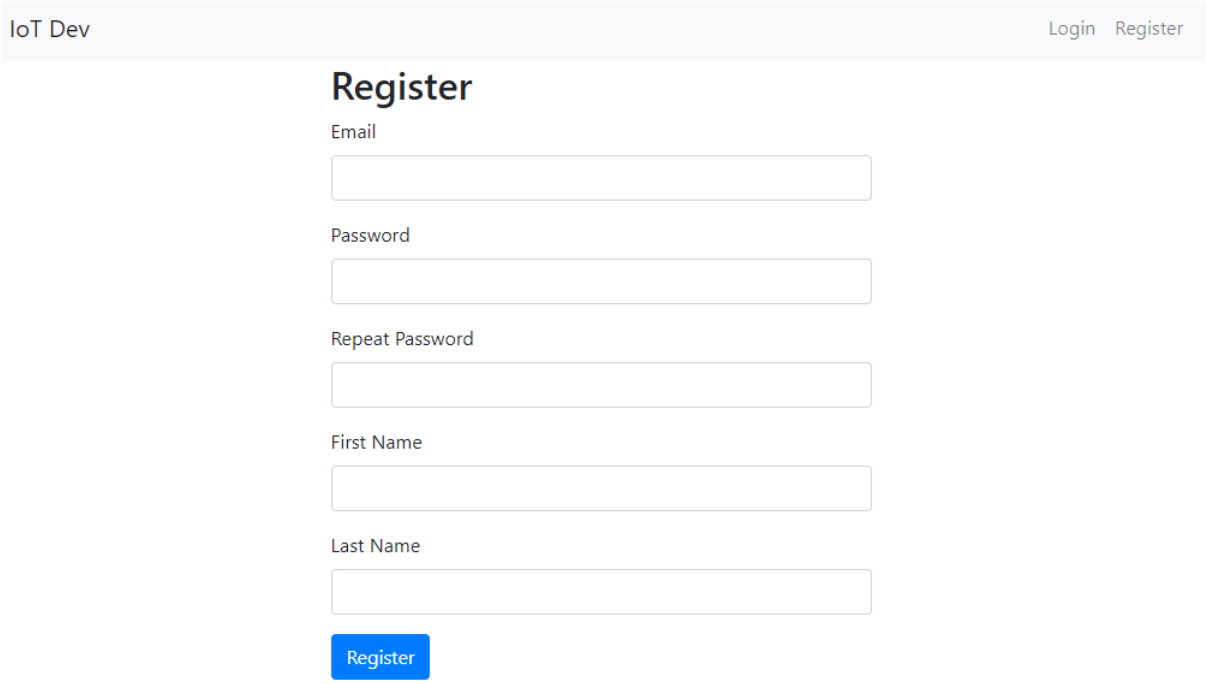
```
    for value in valuefields:
        data.append({
            'field1': value.field1_value,
            'field2': value.field2_value,
            'field3': value.field3_value,
            'field4': value.field4_value,
            'field5': value.field5_value,
            'field6': value.field6_value,
            'created_at': value.created_at
        })
    return jsonify(data)  # Επιστροφή των δεδομένων σε μορφή JSON


@app.route('/device/<int:device_id>/values')
@login_required
def device_values(device_id):
    device = Device.query.get_or_404(device_id)  # Αναζήτηση της συσκευής με βάση το ID
    valuefields = ValueFields.query.filter_by(deviceid=device.id).all()  # Αναζήτηση των τιμών των πεδίων της συσκευής
    return render_template('device_values.html', device=device, valuefields=valuefields)  # Επιστροφή της σελίδας τιμών της συσκευής με τα δεδομένα
```

Το σύστημα IoTDev περιλαμβάνει τη δυνατότητα διαχείρισης IoT συσκευών, τη λήψη και αποθήκευση δεδομένων από αισθητήρες, καθώς και την προβολή αυτών των δεδομένων σε μορφή γραφημάτων. Οι χρήστες μπορούν να εγγραφούν, να συνδεθούν, να προσθέσουν νέες συσκευές, να επεξεργαστούν υπάρχουσες και να δουν τα δεδομένα που έχουν συλλέξει οι συσκευές τους. Οι δυνατότητες αυτές παρέχονται μέσω μιας διεπαφής που έχει αναπτυχθεί με το Flask framework, χρησιμοποιώντας τις βιβλιοθήκες Bootstrap και DataTables για το frontend και MySQL για την αποθήκευση δεδομένων.

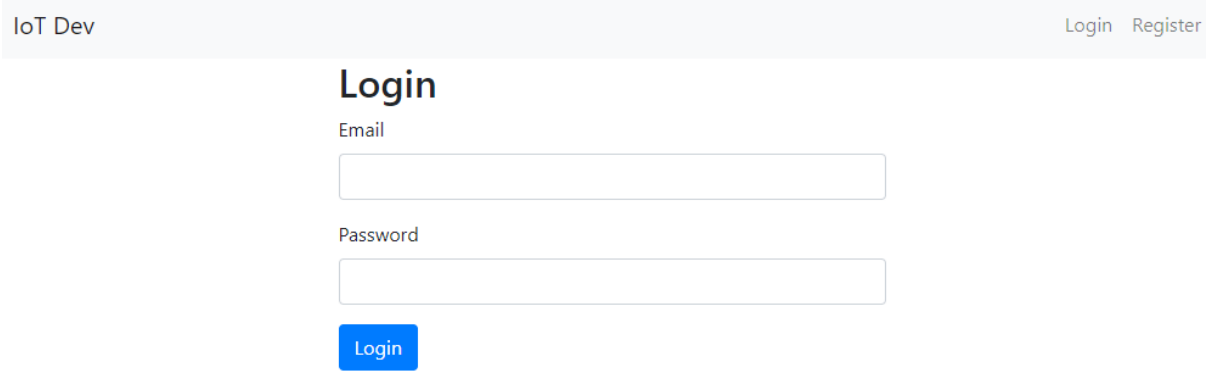
4.3 Η εφαρμογή που χρησιμοποιεί Python+Flask+MySQL+Bootstrap

Στο υποκεφάλαιο αυτό θα περιγραφεί η ιστοσελίδα του συστήματος.



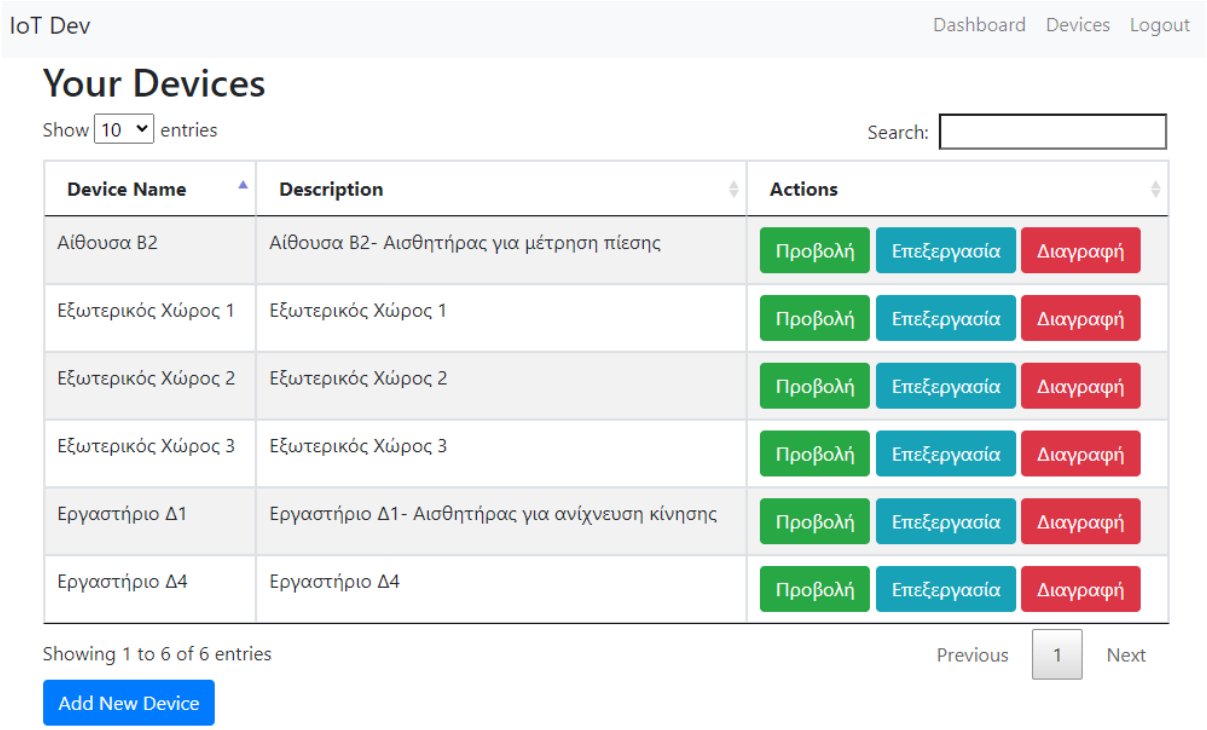
Εικόνα 4.2: Σελίδα για την εγγραφή ενός νέου Χρήστη

Η Εικόνα 4.2 παρουσιάζει τη σελίδα εγγραφής ενός νέου χρήστη στην εφαρμογή IoTDev. Αυτή η σελίδα περιέχει μια φόρμα όπου οι χρήστες μπορούν να εισάγουν τα στοιχεία τους, όπως το email, τον κωδικό πρόσβασης, το όνομα και το επώνυμό τους. Η φόρμα χρησιμοποιεί το Flask-WTF για την επικύρωση των δεδομένων, διασφαλίζοντας ότι οι πληροφορίες είναι σωστές πριν αποθηκευτούν στη βάση δεδομένων. Η σελίδα εγγραφής είναι ο πρώτος βήμα για τους νέους χρήστες ώστε να αποκτήσουν πρόσβαση στις λειτουργίες της πλατφόρμας.



Εικόνα 4.3: Σελίδα για τη σύνδεση του χρήστη

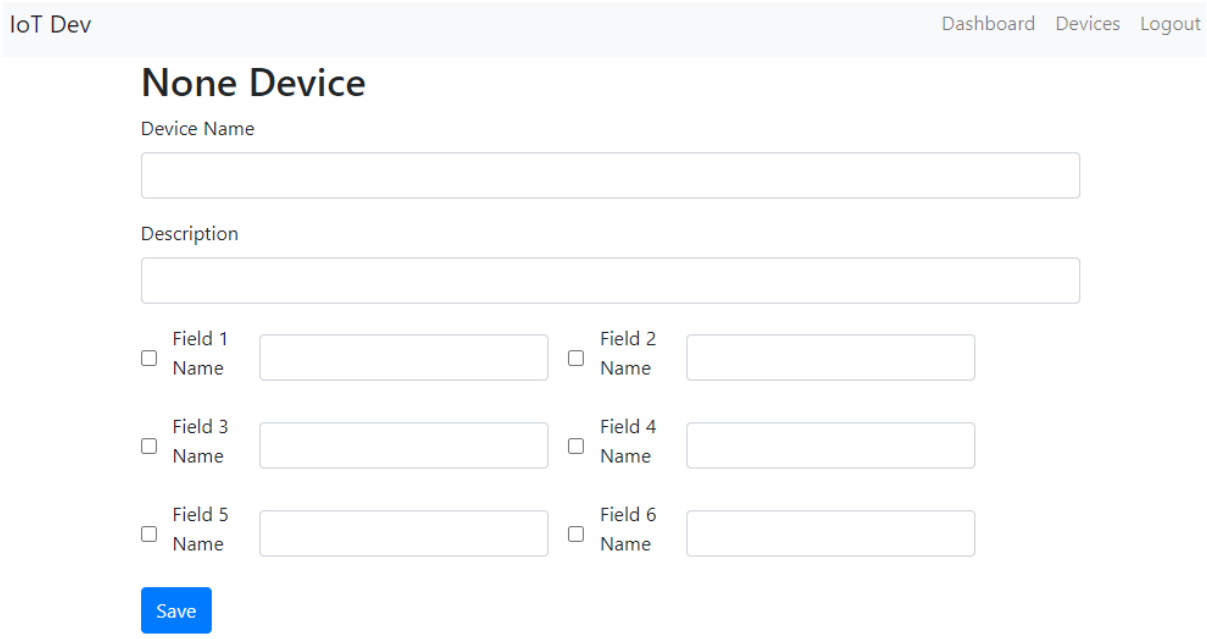
Η Εικόνα 4.3 δείχνει τη σελίδα σύνδεσης του χρήστη. Στη σελίδα αυτή, οι χρήστες μπορούν να εισάγουν το email και τον κωδικό πρόσβασής τους για να συνδεθούν στην εφαρμογή IoTDev. Η φόρμα σύνδεσης ελέγχει τα στοιχεία που έχουν εισαχθεί και, αν είναι σωστά, επιτρέπει στον χρήστη να αποκτήσει πρόσβαση στο προσωπικό του πίνακα ελέγχου. Σε περίπτωση αποτυχίας σύνδεσης, εμφανίζεται ένα μήνυμα λάθους που ενημερώνει τον χρήστη για το πρόβλημα.



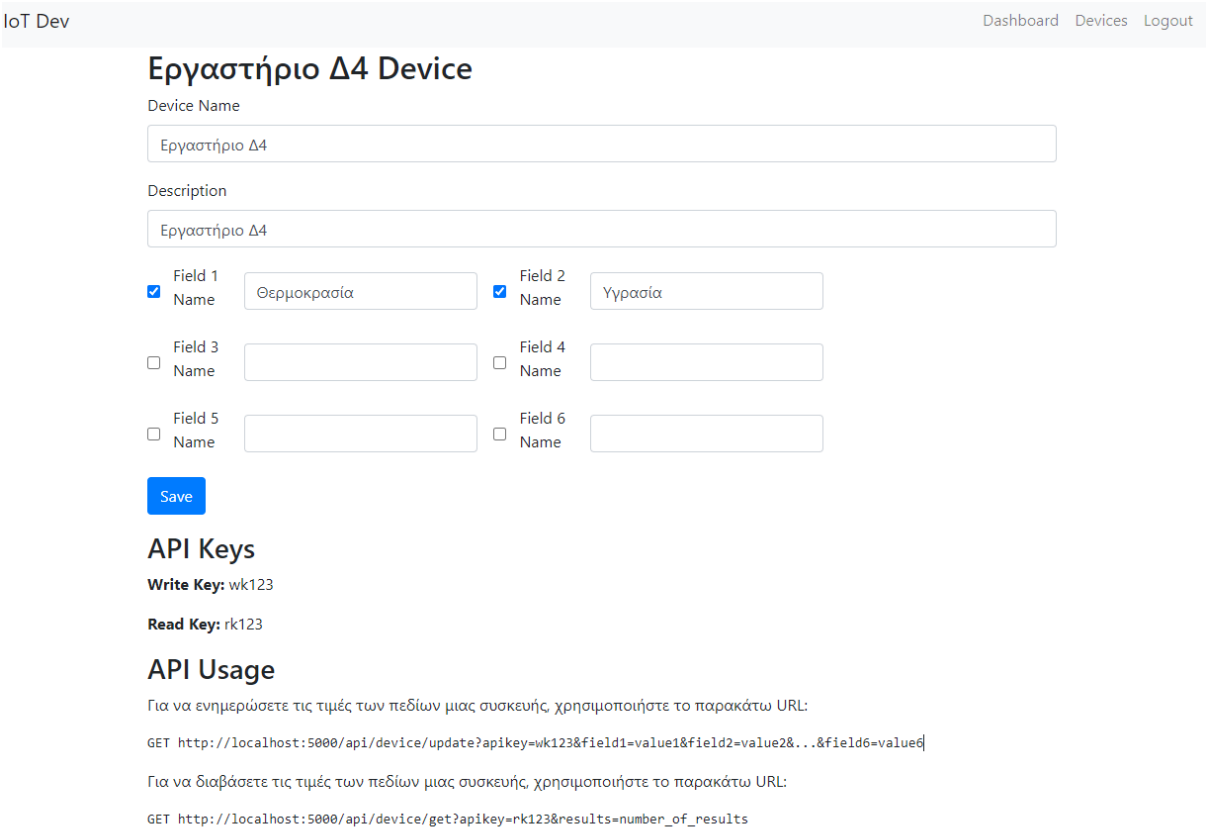
Εικόνα 4.4: Κεντρική σελίδα που φαίνονται όλα τα Devices με κουμπιά Προβολής, Επεξεργασίας ή Διαγραφής. Ο πίνακας datatable παρέχει δυνατότητες διαφορετικών προβολών.

Η Εικόνα 4.4 απεικονίζει την κεντρική σελίδα του πίνακα ελέγχου (dashboard), όπου εμφανίζονται όλες οι IoT συσκευές του χρήστη. Ο πίνακας αυτός χρησιμοποιεί τη βιβλιοθήκη DataTables για την εύκολη και διαδραστική εμφάνιση των δεδομένων, παρέχοντας δυνατότητες ταξινόμησης, αναζήτησης και σελιδοποίησης. Δίπλα από κάθε συσκευή υπάρχουν κουμπιά για προβολή, επεξεργασία και διαγραφή, επιτρέποντας στον χρήστη να διαχειριστεί τις συσκευές του εύκολα και αποτελεσματικά.

Η Εικόνα 4.5 δείχνει τη σελίδα δημιουργίας νέας συσκευής. Σε αυτή τη σελίδα, οι χρήστες μπορούν να εισάγουν τα χαρακτηριστικά μιας νέας IoT συσκευής, όπως το όνομα, την περιγραφή και τις διάφορες παραμέτρους της. Η φόρμα περιέχει επίσης επιλογές για να ενεργοποιηθούν ή να απενεργοποιηθούν συγκεκριμένα πεδία δεδομένων. Μετά την υποβολή της φόρμας, η νέα συσκευή προστίθεται στη βάση δεδομένων και εμφανίζεται στον πίνακα ελέγχου του χρήστη.

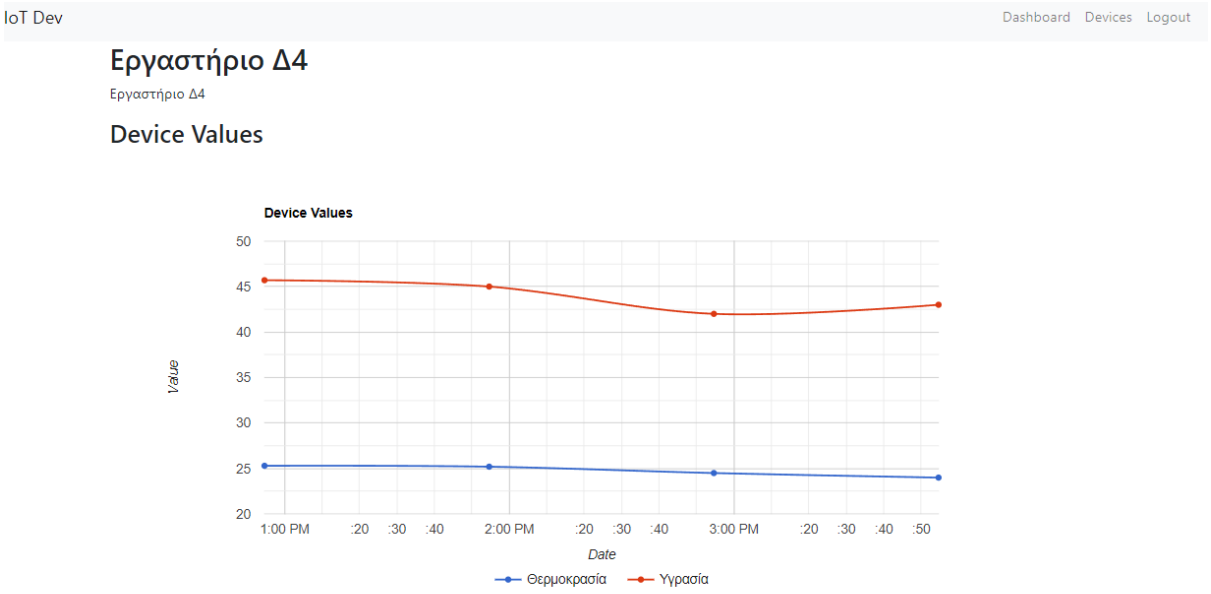


Εικόνα 4.5: Σελίδα δημιουργίας νέας συσκευής



Εικόνα 4.6: Σελίδα επεξεργασίας χαρακτηριστικών μιας συσκευής με διαθέσιμα τα κλειδιά και του κατάλληλου Link για την εκτέλεση των μεθόδων API.

Η Εικόνα 4.6 απεικονίζει τη σελίδα επεξεργασίας χαρακτηριστικών μιας συσκευής. Σε αυτή τη σελίδα, οι χρήστες μπορούν να ενημερώσουν τα στοιχεία της συσκευής τους, όπως το όνομα, την περιγραφή και τις παραμέτρους της. Επιπλέον, εμφανίζονται τα API keys (κλειδιά API) της συσκευής, καθώς και τα κατάλληλα links για την εκτέλεση των μεθόδων API. Αυτό επιτρέπει στους χρήστες να διαχειρίζονται τις συσκευές τους και να χρησιμοποιούν τις δυνατότητες του API για εγγραφή και ανάγνωση δεδομένων.

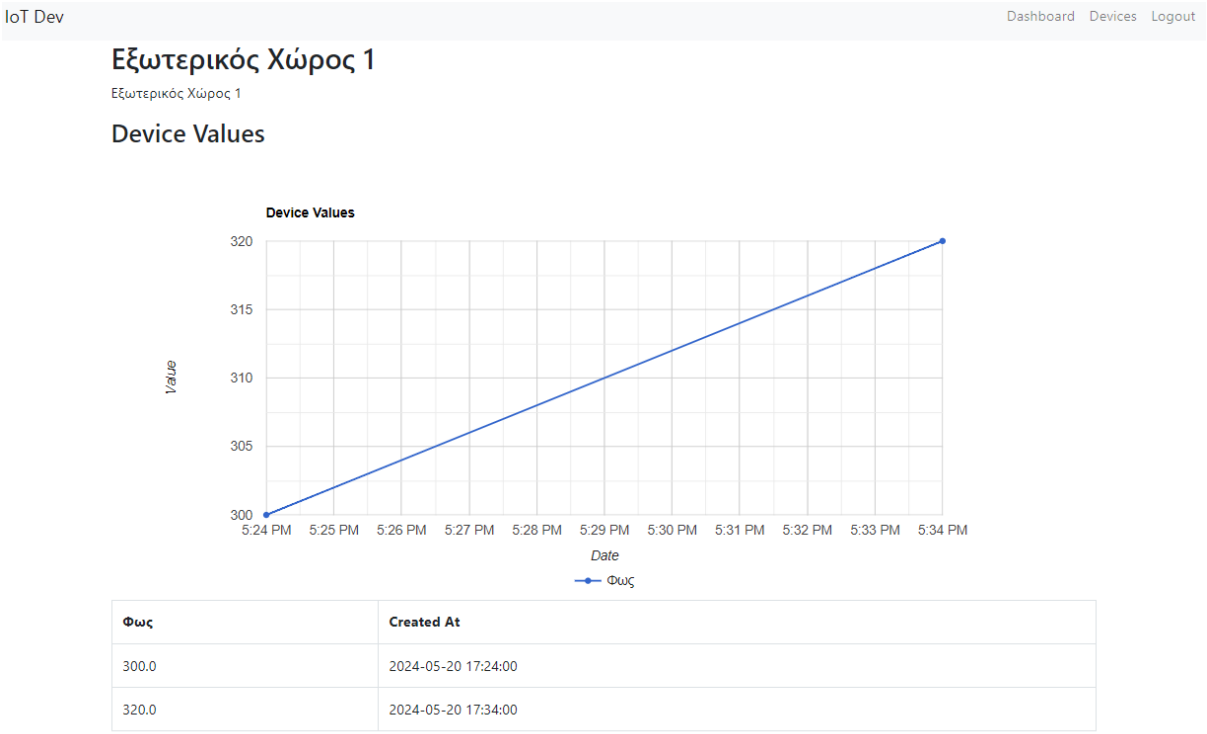


Θερμοκρασία	Υγρασία	Created At
25.3	45.7	2024-05-20 12:54:39
25.2	45.0	2024-05-20 13:54:39
24.5	42.0	2024-05-20 14:54:39
24.0	43.0	2024-05-20 15:54:39

Εικόνα 4.7: Προβολή του google chart και των δεδομένων από τα διαθέσιμα-ενεργοποιημένα πεδία της συσκευής - 1

Η Εικόνα 4.7 παρουσιάζει την προβολή του Google Chart που απεικονίζει τα δεδομένα από τα διαθέσιμα και ενεργοποιημένα πεδία μιας IoT συσκευής. Το γράφημα αυτό παρέχει μια γραφική αναπαράσταση των τιμών που έχουν συλλεχθεί από τους αισθητήρες της συσκευής, επιτρέποντας στους χρήστες να αναλύσουν τα δεδομένα εύκολα και γρήγορα. Η προβολή αυτή είναι χρήσιμη για την παρακολούθηση των αλλαγών των τιμών των πεδίων σε πραγματικό χρόνο.

Η Εικόνα 4.8 είναι μια ακόμη προβολή του Google Chart που απεικονίζει δεδομένα από τα ενεργοποιημένα πεδία μιας άλλης IoT συσκευής. Όπως και στην Εικόνα 4.7, το γράφημα προσφέρει μια οπτική αναπαράσταση των τιμών των πεδίων, βοηθώντας τους χρήστες να κατανοήσουν καλύτερα τις τάσεις και τις αλλαγές των μετρήσεων των αισθητήρων. Η προβολή αυτή μπορεί να προσαρμοστεί ανάλογα με τις ανάγκες του χρήστη και τα συγκεκριμένα δεδομένα που παρακολουθούνται.



Εικόνα 4.8: Προβολή του google chart και των δεδομένων από τα διαθέσιμα-ενεργοποιημένα πεδία της συσκευής – 2

4.4 Δημιουργία Python API clients για εισαγωγή ή ανάγνωση τιμών στην/από συσκευή

δύο Python scripts: το ένα θα εισάγει τιμές σε μια συσκευή χρησιμοποιώντας το API key για εγγραφή και το άλλο θα ανακτά τιμές από μια συσκευή χρησιμοποιώντας το API key για ανάγνωση.

Script για Εισαγωγή Τιμών (Write Values)

```
import requests


# Διεύθυνση του API για την εισαγωγή τιμών

api_url = 'http://localhost:5000/api/device/update'


# Το API key για εγγραφή (Write Key)

write_key = 'your_write_key_here'
```

```
# Οι τιμές που θέλουμε να εισάγουμε στα πεδία της συσκευής
data = {
    'apikey': write_key,  # Το API key για εγγραφή
    'field1': 25.3,       # Τιμή για το πεδίο 1
    'field2': 78.1,       # Τιμή για το πεδίο 2
    'field3': 45.2,       # Τιμή για το πεδίο 3
    'field4': 102.5,      # Τιμή για το πεδίο 4
    'field5': 36.7,       # Τιμή για το πεδίο 5
    'field6': 87.9        # Τιμή για το πεδίο 6
}

# Αποστολή του αιτήματος για εισαγωγή τιμών
response = requests.get(api_url, params=data)

# Έλεγχος της απάντησης του server
if response.status_code == 200:
    print('Επιτυχής εισαγωγή τιμών:', response.json())
else:
    print('Σφάλμα κατά την εισαγωγή τιμών:', response.status_code, response.json())
```

- Χρησιμοποιούμε το API endpoint http://localhost:5000/api/device/update για την εισαγωγή τιμών.
- Περνάμε το API key για εγγραφή (write_key) και τις τιμές για τα πεδία της συσκευής ως παράμετροι.
- Στέλνουμε ένα αίτημα GET με τις παραμέτρους και ελέγχουμε την απάντηση του server για να διαπιστώσουμε αν η εισαγωγή ήταν επιτυχής.

Script για Ανάγνωση Τιμών (Read Values)

```
import requests

# Διεύθυνση του API για την ανάγνωση τιμών
api_url = 'http://localhost:5000/api/device/get'

# Το API key για ανάγνωση (Read Key)
read_key = 'your_read_key_here'

# Ο αριθμός των αποτελεσμάτων που θέλουμε να ανακτήσουμε
results = 5

# Οι παράμετροι για το αίτημα
params = {
    'apikey': read_key,  # Το API key για ανάγνωση
    'results': results   # Ο αριθμός των αποτελεσμάτων
}

# Αποστολή του αιτήματος για ανάγνωση τιμών
response = requests.get(api_url, params=params)

# Έλεγχος της απάντησης του server
if response.status_code == 200:
    data = response.json()
    print('Τιμές:')
    for entry in data:
        print(entry)
else:
    print('Σφάλμα κατά την ανάγνωση τιμών:', response.status_code, response.json())
```

- Χρησιμοποιούμε το API endpoint http://localhost:5000/api/device/update για την εισαγωγή τιμών.
- Περνάμε το API key για εγγραφή (write_key) και τις τιμές για τα πεδία της συσκευής ως παράμετροι.
- Στέλνουμε ένα αίτημα GET με τις παραμέτρους και ελέγχουμε την απάντηση του server για να διαπιστώσουμε αν η εισαγωγή ήταν επιτυχής.

4.5 Η Βάση του συστήματος

Πρώτα θα δούμε το script δημιουργίας της βάσης και των πινάκων.

Script Δημιουργίας Βάσης Δεδομένων και Πινάκων

```
-- Δημιουργία της βάσης δεδομένων
CREATE DATABASE IF NOT EXISTS iotdev;

-- Επιλογή της βάσης δεδομένων
USE iotdev;

-- Δημιουργία του πίνακα `user`
CREATE TABLE `user` (
  `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT,  -- Πρωτεύον κλειδί με αυτόματη αύξηση
  `email` varchar(255) NOT NULL,  -- Μοναδικό πεδίο email
  `password_hash` varchar(255) NOT NULL,  -- Hash του κωδικού πρόσβασης
  `firstname` varchar(255) NOT NULL,  -- Πρώτο όνομα του χρήστη
  `lastname` varchar(255) NOT NULL,  -- Επώνυμο του χρήστη
  `created_at` timestamp NOT NULL DEFAULT current_timestamp(),  -- Ημερομηνία δημιουργίας
  `updated_at` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),  -- Ημερομηνία ενημέρωσης
  PRIMARY KEY (`id`),  -- Ορισμός του πρωτεύοντος κλειδιού
  UNIQUE KEY `email` (`email`)  -- Μοναδικός δείκτης στο πεδίο email
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

-- Δημιουργία του πίνακα `device`
CREATE TABLE `device` (
  `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT,  -- Πρωτεύον κλειδί με αυτόματη αύξηση
  `userid` int(10) UNSIGNED NOT NULL,  -- Ξένο κλειδί προς τον πίνακα `user`
```

```
  `nameofdevice` varchar(50) NOT NULL,  -- Όνομα της συσκευής
  `description` varchar(255) DEFAULT NULL,  -- Περιγραφή της συσκευής
  `field1_check` tinyint(1) DEFAULT 0,  -- Έλεγχος ενεργοποίησης για το πεδίο 1
  `field1_name` varchar(50) DEFAULT NULL,  -- Όνομα του πεδίου 1
  `field2_check` tinyint(1) DEFAULT 0,  -- Έλεγχος ενεργοποίησης για το πεδίο 2
  `field2_name` varchar(50) DEFAULT NULL,  -- Όνομα του πεδίου 2
  `field3_check` tinyint(1) DEFAULT 0,  -- Έλεγχος ενεργοποίησης για το πεδίο 3
  `field3_name` varchar(50) DEFAULT NULL,  -- Όνομα του πεδίου 3
  `field4_check` tinyint(1) DEFAULT 0,  -- Έλεγχος ενεργοποίησης για το πεδίο 4
  `field4_name` varchar(50) DEFAULT NULL,  -- Όνομα του πεδίου 4
  `field5_check` tinyint(1) DEFAULT 0,  -- Έλεγχος ενεργοποίησης για το πεδίο 5
  `field5_name` varchar(50) DEFAULT NULL,  -- Όνομα του πεδίου 5
  `field6_check` tinyint(1) DEFAULT 0,  -- Έλεγχος ενεργοποίησης για το πεδίο 6
  `field6_name` varchar(50) DEFAULT NULL,  -- Όνομα του πεδίου 6
  `private` tinyint(1) DEFAULT 1,  -- Έλεγχος ιδιωτικότητας της συσκευής
  `created_at` timestamp NOT NULL DEFAULT current_timestamp(),  -- Ημερομηνία δημιουργίας
  `updated_at` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),  -- Ημερομηνία ενημέρωσης
  PRIMARY KEY (`id`),  -- Ορισμός του πρωτεύοντος κλειδιού
  KEY `userid` (`userid`),  -- Δείκτης στο πεδίο `userid`
  CONSTRAINT `device_ibfk_1` FOREIGN KEY (`userid`) REFERENCES `user` (`id`)  -- Ορισμός ξένου κλειδιού
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

-- Δημιουργία του πίνακα `valuefields`
CREATE TABLE `valuefields` (
  `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT,  -- Πρωτεύον κλειδί με αυτόματη αύξηση
  `deviceid` int(10) UNSIGNED NOT NULL,  -- Ξένο κλειδί προς τον πίνακα `device`
  `field1_value` double DEFAULT NULL,  -- Τιμή για το πεδίο 1
  `field2_value` double DEFAULT NULL,  -- Τιμή για το πεδίο 2
  `field3_value` double DEFAULT NULL,  -- Τιμή για το πεδίο 3
  `field4_value` double DEFAULT NULL,  -- Τιμή για το πεδίο 4
  `field5_value` double DEFAULT NULL,  -- Τιμή για το πεδίο 5
  `field6_value` double DEFAULT NULL,  -- Τιμή για το πεδίο 6
  `created_at` timestamp NOT NULL DEFAULT current_timestamp(),  -- Ημερομηνία δημιουργίας
  PRIMARY KEY (`id`),  -- Ορισμός του πρωτεύοντος κλειδιού
  KEY `deviceid` (`deviceid`),  -- Δείκτης στο πεδίο `deviceid`
  CONSTRAINT `valuefields_ibfk_1` FOREIGN KEY (`deviceid`) REFERENCES `device` (`id`)  -- Ορισμός ξένου κλειδιού
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

-- Δημιουργία του πίνακα `api_keys`
CREATE TABLE `api_keys` (
  `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT,  -- Πρωτεύον κλειδί με αυτόματη αύξηση
  `deviceid` int(10) UNSIGNED NOT NULL,  -- Ξένο κλειδί προς τον πίνακα `device`
  `write_key` varchar(255) NOT NULL,  -- API key για εγγραφή
  `read_key` varchar(255) NOT NULL,  -- API key για ανάγνωση
  `created_at` timestamp NOT NULL DEFAULT current_timestamp(),  -- Ημερομηνία δημιουργίας
```

```
  PRIMARY KEY (`id`),  -- Ορισμός του πρωτεύοντος κλειδιού
  KEY `deviceid` (`deviceid`),  -- Δείκτης στο πεδίο `deviceid`
  CONSTRAINT `api_keys_ibfk_1` FOREIGN KEY (`deviceid`) REFERENCES `device` (`id`)  -- Ορισμός ξένου κλειδιού
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Το script δημιουργίας της βάσης δεδομένων και των πινάκων για την εφαρμογή IoTDev περιλαμβάνει τα εξής:

- Δημιουργία της Βάσης Δεδομένων: Το script ξεκινά με τη δημιουργία της βάσης δεδομένων iotdev αν αυτή δεν υπάρχει ήδη. Στη συνέχεια, επιλέγεται η βάση δεδομένων για να γίνουν οι επόμενες ενέργειες.

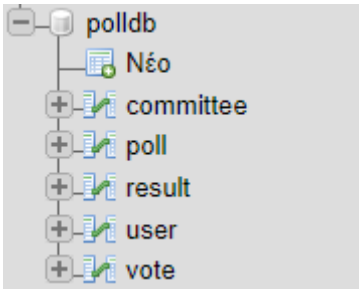
- Πίνακας user: Δημιουργείται ο πίνακας user που αποθηκεύει τις πληροφορίες των χρηστών, όπως το email, το hash του κωδικού πρόσβασης, το όνομα, το επώνυμο, και τις ημερομηνίες δημιουργίας και ενημέρωσης του λογαριασμού.

- Πίνακας device: Δημιουργείται ο πίνακας device που αποθηκεύει τις πληροφορίες για τις IoT συσκευές. Περιλαμβάνει πεδία όπως το όνομα της συσκευής, την περιγραφή, και τις τιμές ενεργοποίησης και ονομασίας για έως και έξι πεδία δεδομένων. Υπάρχει επίσης ένα πεδίο για την ιδιωτικότητα της συσκευής και ξένο κλειδί που συνδέει τη συσκευή με έναν χρήστη.

- Πίνακας valuefields: Δημιουργείται ο πίνακας valuefields που αποθηκεύει τις τιμές των πεδίων μιας συσκευής. Κάθε εγγραφή περιλαμβάνει τις τιμές για έως και έξι πεδία και την ημερομηνία δημιουργίας της εγγραφής. Ο πίνακας αυτός συνδέεται με τον πίνακα device μέσω ενός ξένου κλειδιού.

- Πίνακας api_keys: Δημιουργείται ο πίνακας api_keys που αποθηκεύει τα API keys για εγγραφή και ανάγνωση δεδομένων από τις συσκευές. Κάθε εγγραφή περιλαμβάνει το κλειδί για εγγραφή, το κλειδί για ανάγνωση και την ημερομηνία δημιουργίας. Ο πίνακας αυτός συνδέεται επίσης με τον πίνακα device μέσω ενός ξένου κλειδιού.

Στις παρακάτω Εικόνες φαίνονται η Δομή και τα περιεχόμενα της βάσης μας.



Εικόνα 4.9: Η βάση με τους πίνακες

#	Όνομα	Τύπος	Σύνθεση	Χαρακτηριστικά	Κενό	Προεπιλογή
1	**id**	int(10)		UNSIGNED	Όχι	*Καμία*
2	**enable**	tinyint(3)		UNSIGNED	Ναι	1
3	**email**	varchar(255)	utf8mb4_general_ci		Όχι	*Καμία*
4	**password_hash**	varchar(255)	utf8mb4_general_ci		Όχι	*Καμία*
5	**firstname**	varchar(255)	utf8mb4_general_ci		Όχι	*Καμία*
6	**lastname**	varchar(255)	utf8mb4_general_ci		Όχι	*Καμία*
7	**created_at**	timestamp			Όχι	current_timestamp()
8	**updated_at**	timestamp			Όχι	current_timestamp()

Εικόνα 4.10: Δομή του user

id	enable	email	password_hash	firstname	lastname
1	1	giannis@example.com	$2y$10$iJ/UUEw0dcOPilzvp0T4seWrtfNHwplZatEEedUWTg....	Γιάννης	Παπαδόπουλ
2	1	maria@example.com	$2y$10$iJ/UUEw0dcOPilzvp0T4seWrtfNHwplZatEEedUWTg....	Μαρία	Νικολάου
3	1	kostas@example.com	$2y$10$iJ/UUEw0dcOPilzvp0T4seWrtfNHwplZatEEedUWTg....	Κώστας	Γεωργίου
4	1	elena@example.com	$2y$10$iJ/UUEw0dcOPilzvp0T4seWrtfNHwplZatEEedUWTg....	Έλενα	Σωτηρίου
5	1	nikos@example.com	$2y$10$iJ/UUEw0dcOPilzvp0T4seWrtfNHwplZatEEedUWTg....	Νίκος	Δημητρίου
6	1	vicky@example.com	$2y$10$iJ/UUEw0dcOPilzvp0T4seWrtfNHwplZatEEedUWTg....	Βίκυ	Πετρίδου
7	1	dimitris@example.com	$2y$10$iJ/UUEw0dcOPilzvp0T4seWrtfNHwplZatEEedUWTg....	Δημήτρης	Κωνσταντίνου
8	1	sophia@example.com	$2y$10$iJ/UUEw0dcOPilzvp0T4seWrtfNHwplZatEEedUWTg....	Σοφία	Αναστασίου
9	1	alex@example.com	$2y$10$iJ/UUEw0dcOPilzvp0T4seWrtfNHwplZatEEedUWTg....	Αλέξανδρος	Βασιλείου
10	1	christina@example.com	$2y$10$iJ/UUEw0dcOPilzvp0T4seWrtfNHwplZatEEedUWTg....	Χριστίνα	Μιχαήλ

Εικόνα 4.11: Περιεχόμενα του user

#	Όνομα	Τύπος	Σύνθεση	Χαρακτηριστικά	Κενό	Προεπιλογή
1	id	int(10)		UNSIGNED	Όχι	*Καμία*
2	enable	tinyint(3)		UNSIGNED	Ναι	1
3	private	tinyint(3)		UNSIGNED	Ναι	1
4	userid	int(10)		UNSIGNED	Όχι	*Καμία*
5	nameofdevice	varchar(50)	utf8mb4_general_ci		Όχι	*Καμία*
6	description	varchar(255)	utf8mb4_general_ci		Ναι	*NULL*
7	field1_check	tinyint(3)		UNSIGNED	Ναι	0
8	field1_name	varchar(50)	utf8mb4_general_ci		Ναι	*NULL*
9	field2_check	tinyint(3)		UNSIGNED	Ναι	0
10	field2_name	varchar(50)	utf8mb4_general_ci		Ναι	*NULL*
11	field3_check	tinyint(3)		UNSIGNED	Ναι	0
12	field3_name	varchar(50)	utf8mb4_general_ci		Ναι	*NULL*
13	field4_check	tinyint(3)		UNSIGNED	Ναι	0
14	field4_name	varchar(50)	utf8mb4_general_ci		Ναι	*NULL*
15	field5_check	tinyint(3)		UNSIGNED	Ναι	0
16	field5_name	varchar(50)	utf8mb4_general_ci		Ναι	*NULL*
17	field6_check	tinyint(3)		UNSIGNED	Ναι	0
18	field6_name	varchar(50)	utf8mb4_general_ci		Ναι	*NULL*
19	created_at	timestamp			Όχι	current_timestamp()
20	updated_at	timestamp			Όχι	current_timestamp()

Εικόνα 4.12: Δομή του device

id	enable	private	userid	nameofdevice	description	field1_check	field1_name	field2_check	field2_name	field3_check	field3_name	field4_check	field4_name	field5_check	field5_name	field6_check	field6_name
1	1	1	1	Εργαστήριο Δ4	Εργαστήριο Δ4	1	Θερμοκρασία	1	Υγρασία	0		0		0		0	
2	1	1	1	Εργαστήριο Δ1	Εργαστήριο Δ1- Αισθητήρας για ανίχνευση κίνησης	1	Κίνηση	0	*NULL*	0	*NULL*	0	*NULL*	0	*NULL*	0	*NULL*
3	1	1	1	Αίθουσα Β2	Αίθουσα Β2- Αισθητήρας για μέτρηση πίεσης	1	Πίεση	1	Θερμοκρασία	0		0		0		0	
4	1	1	1	Εξωτερικός Χώρος 1	Εξωτερικός Χώρος 1	1	Φως	0	*NULL*	0	*NULL*	0	*NULL*	0	*NULL*	0	*NULL*
5	1	1	1	Εξωτερικός Χώρος 2	Εξωτερικός Χώρος 2	1	Ήχος	0	*NULL*	0	*NULL*	0	*NULL*	0	*NULL*	0	*NULL*
6	1	1	1	Εξωτερικός Χώρος 3	Εξωτερικός Χώρος 3	1	Υγρασία	1	Θερμοκρασία	0	*NULL*	0	*NULL*	0	*NULL*	0	*NULL*

Εικόνα 4.13: Περιεχόμενα του device

#	Όνομα	Τύπος	Σύνθεση	Χαρακτηριστικά	Κενό	Προεπιλογή
1	**id**	int(10)		UNSIGNED	Όχι	*Καμία*
2	**enable**	tinyint(3)		UNSIGNED	Ναι	1
3	**deviceid**	int(10)		UNSIGNED	Όχι	*Καμία*
4	**field1_value**	double			Ναι	*NULL*
5	**field2_value**	double			Ναι	*NULL*
6	**field3_value**	double			Ναι	*NULL*
7	**field4_value**	double			Ναι	*NULL*
8	**field5_value**	double			Ναι	*NULL*
9	**field6_value**	double			Ναι	*NULL*
10	**created_at**	timestamp			Όχι	current_timestamp()

Εικόνα 4.14: Δομή του valuefields

id	enable	deviceid	field1_value	field2_value	field3_value	field4_value	field5_value	field6_value	created_at
1	1	1	25.3	45.7	*NULL*	*NULL*	*NULL*	*NULL*	2024-05-20 12:54:39
2	1	1	25.2	45	*NULL*	*NULL*	*NULL*	*NULL*	2024-05-20 13:54:39
3	1	1	24.5	42	*NULL*	*NULL*	*NULL*	*NULL*	2024-05-20 14:54:39
4	1	1	24	43	*NULL*	*NULL*	*NULL*	*NULL*	2024-05-20 15:54:39
5	1	2	1	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	2024-05-20 13:00:00
6	1	2	1	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	2024-05-20 13:13:39
7	1	2	0	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	2024-05-20 13:15:32
8	1	3	1013.25	20.5	*NULL*	*NULL*	*NULL*	*NULL*	2024-05-20 15:12:32
9	1	3	1018	21.2	*NULL*	*NULL*	*NULL*	*NULL*	2024-05-20 17:13:39
10	1	4	300	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	2024-05-20 17:24:00
11	1	4	320	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	2024-05-20 17:34:00
12	1	5	55.3	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	2024-05-20 18:28:28
13	1	5	56.2	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	2024-05-20 18:34:00
14	1	6	60.2	22.1	*NULL*	*NULL*	*NULL*	*NULL*	2024-05-20 18:40:00
15	1	6	61.4	22.4	*NULL*	*NULL*	*NULL*	*NULL*	2024-05-20 18:44:00
16	1	6	61	22	*NULL*	*NULL*	*NULL*	*NULL*	2024-05-20 18:50:00

Εικόνα 4.15: Περιεχόμενα του valuefields

#	Όνομα	Τύπος	Σύνθεση	Χαρακτηριστικά	Κενό	Προεπιλογή
1	**id**	int(10)		UNSIGNED	Όχι	*Καμία*
2	**deviceid**	int(10)		UNSIGNED	Όχι	*Καμία*
3	**write_key**	varchar(255)	utf8mb4_general_ci		Όχι	*Καμία*
4	**read_key**	varchar(255)	utf8mb4_general_ci		Όχι	*Καμία*
5	**created_at**	timestamp			Όχι	current_timestamp()

Εικόνα 4.16: Δομή του api_keys

id	deviceid	write_key	read_key	created_at
1	1	wk123	rk123	2024-05-20 14:32:56

Εικόνα 4.17: Περιεχόμενα του api_keys

4.6 Ασφάλεια στην εφαρμογή

Η ασφάλεια είναι ένα από τα πιο κρίσιμα στοιχεία για κάθε εφαρμογή, ειδικά όταν αυτή διαχειρίζεται ευαίσθητα δεδομένα χρηστών και IoT συσκευών. Η εφαρμογή IoTDev έχει ενσωματώσει μια σειρά από πρακτικές και τεχνολογίες για να διασφαλίσει την προστασία των δεδομένων και την ασφάλεια των λειτουργιών της. Στο κεφάλαιο αυτό θα εξετάσουμε τις βασικές στρατηγικές και μέτρα που έχουν ληφθεί για την ασφάλεια της εφαρμογής.

Επικύρωση Δεδομένων και Προστασία από CSRF

Η εφαρμογή χρησιμοποιεί το Flask-WTF, το οποίο προσφέρει ισχυρές δυνατότητες επικύρωσης δεδομένων για τις φόρμες. Αυτό διασφαλίζει ότι τα δεδομένα που εισάγονται από τους χρήστες είναι σωστά και ασφαλή. Επιπλέον, το Flask-WTF παρέχει προστασία από Cross-Site Request Forgery (CSRF) επιθέσεις μέσω της χρήσης CSRF tokens. Αυτά τα tokens διασφαλίζουν ότι τα αιτήματα προέρχονται από τον εξουσιοδοτημένο χρήστη και όχι από κακόβουλες πηγές.

```
from flask_wtf.csrf import CSRFProtect

csrf = CSRFProtect(app)
```

Χρήση Hashing για Κωδικούς Πρόσβασης

Οι κωδικοί πρόσβασης των χρηστών δεν αποθηκεύονται ποτέ απευθείας στη βάση δεδομένων. Αντί αυτού, χρησιμοποιούμε το εργαλείο werkzeug.security για τη δημιουργία ασφαλών hash των κωδικών πρόσβασης. Αυτό διασφαλίζει ότι ακόμη και αν αποκτηθεί πρόσβαση στη βάση δεδομένων, οι κωδικοί πρόσβασης θα παραμείνουν προστατευμένοι.

```
from werkzeug.security import generate_password_hash, check_password_hash


class User(UserMixin, db.Model):
    ...
    def set_password(self, password):
        self.password_hash = generate_password_hash(password)


    def check_password(self, password):
        return check_password_hash(self.password_hash, password
```

Διαχείριση API Keys

Κάθε συσκευή στην εφαρμογή διαθέτει μοναδικά API keys για εγγραφή και ανάγνωση δεδομένων. Αυτά τα κλειδιά χρησιμοποιούνται για την επαλήθευση και την εξουσιοδότηση των αιτημάτων προς το API, διασφαλίζοντας ότι μόνο εξουσιοδοτημένοι χρήστες και συσκευές μπορούν να έχουν πρόσβαση ή να τροποποιούν τα δεδομένα.

```
class APIKeys(db.Model):
    ...
    write_key = db.Column(db.String(255), nullable=False)
    read_key = db.Column(db.String(255), nullable=False)
```

Σύνδεση και Διαχείριση Χρηστών

Η εφαρμογή χρησιμοποιεί το Flask-Login για τη διαχείριση των συνδέσεων και των συνεδριών των χρηστών. Αυτό περιλαμβάνει τον έλεγχο ταυτότητας των χρηστών και τη διατήρηση των συνεδριών τους με ασφαλή τρόπο. Το Flask-Login βοηθά επίσης στη διασφάλιση ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να έχουν πρόσβαση σε συγκεκριμένες σελίδες και λειτουργίες της εφαρμογής.

```
from flask_login import LoginManager


login_manager = LoginManager()
login_manager.init_app(app)
```

```
login_manager.login_view = 'login'
```

Χρήση HTTPS

Για την προστασία των δεδομένων κατά τη μεταφορά, είναι σημαντικό η εφαρμογή να λειτουργεί μέσω HTTPS. Αυτό διασφαλίζει ότι όλες οι επικοινωνίες μεταξύ του χρήστη και της εφαρμογής είναι κρυπτογραφημένες, προστατεύοντας τα δεδομένα από υποκλοπές και κακόβουλες επιθέσεις.

Περιορισμός Δικαιωμάτων Πρόσβασης

Η βάση δεδομένων και η εφαρμογή έχουν διαμορφωθεί έτσι ώστε να περιορίζονται τα δικαιώματα πρόσβασης μόνο στους εξουσιοδοτημένους χρήστες. Οι χρήστες έχουν πρόσβαση μόνο στα δεδομένα που σχετίζονται με τις συσκευές τους, διασφαλίζοντας ότι δεν μπορούν να δουν ή να τροποποιήσουν δεδομένα άλλων χρηστών.

Ενημερώσεις και Διορθώσεις Ασφαλείας

Η εφαρμογή IoTDev διατηρείται ενημερωμένη με τις τελευταίες εκδόσεις των βιβλιοθηκών και των εργαλείων που χρησιμοποιεί. Αυτό περιλαμβάνει τακτικές ενημερώσεις για την αντιμετώπιση γνωστών ευπαθειών και την ενίσχυση της συνολικής ασφάλειας του συστήματος.

Η ασφάλεια της εφαρμογής IoTDev είναι πολυεπίπεδη και περιλαμβάνει μέτρα προστασίας σε διάφορα στάδια της αρχιτεκτονικής της. Από την επικύρωση των δεδομένων και τη διαχείριση των κωδικών πρόσβασης έως τη χρήση HTTPS και τον περιορισμό δικαιωμάτων πρόσβασης, κάθε πτυχή της ασφάλειας έχει ληφθεί υπόψη για να διασφαλίσει την προστασία των δεδομένων των χρηστών και των συσκευών τους. Η συνεχής συντήρηση και ενημέρωση της εφαρμογής συμβάλλει επίσης στην αντιμετώπιση νέων προκλήσεων και απειλών στον τομέα της ασφάλειας.

Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης

Παρουσιάστηκε και αναλύθηκε η ανάπτυξη της εφαρμογής IoTDev, η οποία στοχεύει στη διαχείριση IoT συσκευών, τη συλλογή δεδομένων από αισθητήρες και την προβολή αυτών των δεδομένων σε μορφή γραφημάτων. Η ανάπτυξη της εφαρμογής περιλάμβανε τη χρήση του Flask framework για το backend, του Bootstrap και της βιβλιοθήκης DataTables για το frontend, καθώς και της MySQL για την αποθήκευση των δεδομένων. Η ανάπτυξη της εφαρμογής IoTDev απέδειξε ότι είναι δυνατή η δημιουργία μιας ολοκληρωμένης πλατφόρμας διαχείρισης IoT συσκευών χρησιμοποιώντας σύγχρονα εργαλεία και τεχνολογίες. Η εφαρμογή παρέχει μια φιλική προς τον χρήστη διεπαφή για την εγγραφή, σύνδεση και διαχείριση των IoT συσκευών, επιτρέποντας στους χρήστες να στέλνουν και να ανακτούν δεδομένα μέσω ασφαλών API endpoints.

Οι χρήστες μπορούν εύκολα να εγγραφούν και να συνδεθούν στην εφαρμογή, με ασφαλή διαχείριση των κωδικών πρόσβασης και προστασία από CSRF επιθέσεις. Μπορούν να προσθέτουν, να επεξεργάζονται και να διαγράφουν IoT συσκευές, ενώ η εφαρμογή υποστηρίζει την ενεργοποίηση και απενεργοποίηση συγκεκριμένων πεδίων δεδομένων για κάθε συσκευή.

Κάθε συσκευή διαθέτει μοναδικά API keys για εγγραφή και ανάγνωση δεδομένων, εξασφαλίζοντας την ασφάλεια και την ιδιωτικότητα των δεδομένων. Οι χρήστες μπορούν να δουν τα δεδομένα των συσκευών τους σε μορφή γραφημάτων χρησιμοποιώντας το Google Charts, διευκολύνοντας την ανάλυση των δεδομένων σε πραγματικό χρόνο. Παρά την επιτυχία της εφαρμογής, η ανάπτυξη της IoTDev δεν ήταν χωρίς προκλήσεις. Η διαχείριση των API keys, η εξασφάλιση της ασφάλειας των δεδομένων και η αντιμετώπιση της πολυπλοκότητας της διαχείρισης IoT συσκευών απαιτούσαν ιδιαίτερη προσοχή και λεπτομερή σχεδιασμό.

Για να βελτιωθεί περαιτέρω η εφαρμογή IoTDev και να αντιμετωπιστούν οι προκλήσεις που εντοπίστηκαν κατά την ανάπτυξή της, προτείνονται οι ακόλουθες βελτιώσεις. Η εισαγωγή τεχνικών caching μπορεί να βελτιώσει την απόδοση της εφαρμογής, μειώνοντας τον χρόνο απόκρισης για την ανάκτηση δεδομένων και τη φόρτωση σελίδων. Η βελτιστοποίηση των ερωτημάτων SQL και η χρήση ευρετηρίων (indexes) μπορεί να μειώσει το φορτίο στη βάση δεδομένων και να επιταχύνει την απόδοση της εφαρμογής. Η χρήση επιπλέον επιπέδων ασφαλείας, όπως ο έλεγχος ταυτότητας OAuth, μπορεί να προστατεύσει περαιτέρω τα API endpoints της εφαρμογής. Η εφαρμογή πιο λεπτομερών ελέγχων πρόσβασης (RBAC - Role-Based Access Control) μπορεί να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες έχουν πρόσβαση σε συγκεκριμένες λειτουργίες και δεδομένα. Η εισαγωγή αλγορίθμων μηχανικής μάθησης μπορεί να προσθέσει λειτουργίες ανάλυσης και πρόβλεψης στα δεδομένα των IoT συσκευών, παρέχοντας στους χρήστες πιο προχωρημένες δυνατότητες. Η προσθήκη ενός συστήματος διαχείρισης συμβάντων που ειδοποιεί τους χρήστες σε περίπτωση ασυνήθιστων δεδομένων ή σφαλμάτων στις συσκευές μπορεί να βελτιώσει την αξιοπιστία και τη διαχείριση των συσκευών. Η

προσθήκη δυνατοτήτων προσαρμογής ειδοποιήσεων μπορεί να επιτρέψει στους χρήστες να λαμβάνουν ειδοποιήσεις με βάση συγκεκριμένα κριτήρια και όρια που ορίζουν οι ίδιοι.

Η εφαρμογή IoTDev έχει καταφέρει να προσφέρει μια ολοκληρωμένη και αποτελεσματική λύση για τη διαχείριση IoT συσκευών, επιτρέποντας στους χρήστες να συλλέγουν, να αποθηκεύουν και να αναλύουν δεδομένα από τους αισθητήρες τους. Η επιτυχία της εφαρμογής βασίζεται στη χρήση σύγχρονων τεχνολογιών και στην εφαρμογή βέλτιστων πρακτικών ασφαλείας και απόδοσης.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] "ThingSpeak," MathWorks, διαθέσιμο στο: https://thingspeak.com

[2] "ThinkBoard," ThinkBoard, διαθέσιμο στο: https://thinkboard.io

[3] "Grafana," Grafana Labs, διαθέσιμο στο: https://grafana.com

[4] "AWS IoT Core," Amazon Web Services, διαθέσιμο στο: https://aws.amazon.com/iot-core

[5] Python Software Foundation, "Welcome to Python.org," διαθέσιμο στο: https://www.python.org

[6] Pallets Projects, "Flask Documentation," διαθέσιμο στο: https://flask.palletsprojects.com

[7] Google Developers, "Google Charts," διαθέσιμο στο: https://developers.google.com/chart

[8] Datatables, διαθέσιμο στο: https://datatables.net/

[9] Bootstrap, διαθέσιμο στο: https://getbootstrap.com

[10] MySQL, διαθέσιμο στο: https://www.w3schools.com/MySQL/default.asp

[11] SqlAlchemy, διαθέσιμο στο: https://www.sqlalchemy.org/