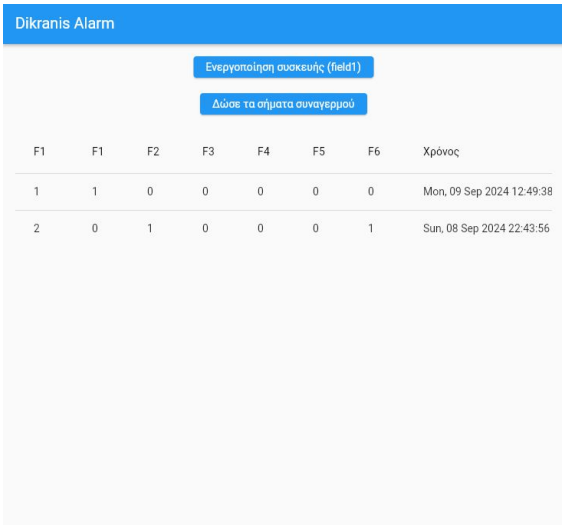
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Σύστημα ειδοποιήσεων κατάστασης εσωτερικού χώρου μέσω διαδικτύου»



Φοιτητής

Δικράνης Παναγιώτης 514306

Επιβλέπων

Δρ. Κυριάκος Τσιακμάκης

Σεπτέμβριος 2024

Σύστημα ειδοποιήσεων κατάστασης εσωτερικού χώρου μέσω διαδικτύου

Κωδικός: 23346

Φοιτητής: Δικράνης Παναγιώτης

Εισηγητής: Δρ Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 26-03-2024

Ημερομηνία περάτωσης Π.Ε. 10-09-2024

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Περίληψη

Η εργασία αφορά την ανάπτυξη ενός ολοκληρωμένου συστήματος ειδοποιήσεων κατάστασης εσωτερικού χώρου μέσω διαδικτύου, το οποίο επιτρέπει την απομακρυσμένη παρακολούθηση και διαχείριση αισθητήρων και συσκευών μέσω μιας εφαρμογής κινητού. Το σύστημα βασίζεται σε μικροελεγκτές, όπως το ESP32, οι οποίοι συνδέονται με αισθητήρες (π.χ. ανιχνευτές κίνησης, μαγνητικές παγίδες) και ρελέ. Οι αισθητήρες αυτοί συλλέγουν δεδομένα και τα αποστέλλουν σε έναν Python Server, ο οποίος επεξεργάζεται τα δεδομένα και τα αποθηκεύει σε μια βάση δεδομένων MySQL. Ο χρήστης μπορεί να έχει πρόσβαση στο σύστημα μέσω μιας εφαρμογής κινητού που αναπτύχθηκε με Flutter, επιτρέποντάς του να παρακολουθεί την κατάσταση των αισθητήρων σε πραγματικό χρόνο, να λαμβάνει ειδοποιήσεις, και να ενεργοποιεί ή να απενεργοποιεί συσκευές απομακρυσμένα.

« Indoor Status Notification System via the Internet»

Abstract

This project focuses on the development of a comprehensive indoor status notification system via the internet, which allows for the remote monitoring and management of sensors and devices through a mobile application. The system is based on microcontrollers, such as the ESP32, which are connected to sensors (e.g., motion detectors, magnetic traps) and relays. These sensors collect data and send it to a Python Server, where the data is processed and stored in a MySQL database.

Users can access the system through a mobile application developed using Flutter, enabling them to monitor the status of the sensors in real-time, receive notifications, and remotely activate or deactivate devices.

Ευχαριστίες

Θα ήθελα να εκφράσω την ευγνωμοσύνη μου στους γονείς μου για τη συνεχή υποστήριξή τους, καθώς και στον επιβλέποντά μου, για την αδιάκοπη παιδαγωγική καθοδήγηση, τις επιστημονικές συμβουλές του, και την πολύτιμη συμβολή του στον κώδικα.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Στη σύγχρονη εποχή, η χρήση συστημάτων παρακολούθησης και ελέγχου μέσω διαδικτύου είναι όλο και πιο διαδεδομένη, εξυπηρετώντας ανάγκες ασφαλείας και αυτοματισμού σε οικιακό αλλά και βιομηχανικό επίπεδο. Η δυνατότητα να έχουμε απομακρυσμένη πρόσβαση σε αισθητήρες, συσκευές και ειδοποιήσεις μέσω φορητών συσκευών, όπως τα smartphones, παρέχει σημαντική ευελιξία και ασφάλεια στον χρήστη. Η παρούσα εργασία επικεντρώνεται στην ανάπτυξη ενός τέτοιου συστήματος, το οποίο επιτρέπει την παρακολούθηση αισθητήρων και την ενεργοποίηση συσκευών μέσω ενός συνδυασμού τεχνολογιών όπως οι μικροελεγκτές (ESP32), ένας κεντρικός Python Server, μια βάση δεδομένων MySQL, και μια εφαρμογή κινητού που αναπτύχθηκε με Flutter.

Ο κύριος στόχος της μελέτης αυτής είναι η ανάπτυξη ενός ολοκληρωμένου συστήματος που μπορεί να συλλέγει δεδομένα από αισθητήρες και να επιτρέπει την αλληλεπίδραση με συσκευές σε πραγματικό χρόνο. Οι αισθητήρες είναι εγκατεστημένοι σε διάφορα σημεία και στέλνουν δεδομένα στον κεντρικό server, ο οποίος τα επεξεργάζεται και τα αποθηκεύει. Μέσω της εφαρμογής κινητού, ο χρήστης μπορεί να παρακολουθεί την κατάσταση των αισθητήρων, να λαμβάνει ειδοποιήσεις, και να ενεργοποιεί ή να απενεργοποιεί συσκευές, όπως ρελέ ή άλλους μηχανισμούς.

Η ανάπτυξη του συστήματος αυτού ενσωματώνει διάφορες τεχνολογίες, οι οποίες συνεργάζονται αρμονικά για να προσφέρουν μια ασφαλή και αποδοτική λύση παρακολούθησης. Η χρήση του Python Server για την επεξεργασία των δεδομένων και την επικοινωνία με τη βάση δεδομένων εξασφαλίζει την επεκτασιμότητα του συστήματος, επιτρέποντας την προσθήκη νέων αισθητήρων και συσκευών χωρίς να απαιτείται εκτεταμένη αναδιάρθρωση.

Το σύστημα που αναπτύχθηκε είναι σχεδιασμένο να καλύψει ένα ευρύ φάσμα εφαρμογών, όπως η παρακολούθηση εσωτερικών χώρων για λόγους ασφαλείας, η διαχείριση συσκευών απομακρυσμένα, και η παρακολούθηση περιβαλλοντικών δεδομένων, όπως η ανίχνευση κίνησης ή η ανίχνευση ανοίγματος και κλεισίματος πορτών. Κάθε αισθητήρας μπορεί να παραμετροποιηθεί από τον χρήστη μέσω της εφαρμογής κινητού, προσφέροντας ευελιξία στη χρήση του συστήματος.

Η παρούσα εργασία αποσκοπεί, επίσης, στη μελέτη των προκλήσεων που προκύπτουν κατά την υλοποίηση ενός συστήματος αυτού του είδους, κυρίως σε ζητήματα ασφάλειας, ευελιξίας και επεκτασιμότητας. Τα δεδομένα που συλλέγονται από τους αισθητήρες είναι ευαίσθητα, γεγονός που καθιστά απαραίτητη την εφαρμογή ισχυρών πρωτοκόλλων ασφαλείας, τόσο στη διαχείριση της βάσης δεδομένων όσο και στην επικοινωνία μεταξύ των συσκευών και του server.

Στα επόμενα κεφάλαια θα παρουσιαστεί λεπτομερώς η αρχιτεκτονική του συστήματος, οι τεχνολογίες που χρησιμοποιήθηκαν, καθώς και η διαδικασία υλοποίησης του κάθε τμήματος. Στη συνέχεια, θα αναλυθούν οι λειτουργίες της εφαρμογής κινητού και τα πλεονεκτήματα που προσφέρει στους χρήστες της, ενώ θα παρουσιαστούν οι δυνατότητες επέκτασης και βελτίωσης του συστήματος. Τέλος, θα δοθούν προτάσεις για μελλοντικές βελτιώσεις, με έμφαση στην ασφάλεια και την αποδοτικότητα του συστήματος.

1.2 Δομή της εργασίας

Η παρούσα εργασία αποτελείται από πέντε βασικά κεφάλαια, τα οποία αναλύουν σταδιακά το σύστημα ειδοποιήσεων κατάστασης εσωτερικού χώρου μέσω διαδικτύου, τις τεχνολογίες που χρησιμοποιήθηκαν, καθώς και παρόμοια συστήματα που υπάρχουν στην αγορά. Παρακάτω παρουσιάζεται η δομή της εργασίας:

Το πρώτο κεφάλαιο περιλαμβάνει την εισαγωγή στο αντικείμενο της εργασίας και δίνει μια γενική εικόνα του συστήματος που αναπτύχθηκε. Επίσης, περιγράφεται η δομή της εργασίας και τα κεφάλαια που ακολουθούν.

Στο δεύτερο κεφάλαιο γίνεται αναφορά σε παρόμοια συστήματα που υπάρχουν στην αγορά και χρησιμοποιούνται για την παρακολούθηση και τον έλεγχο εσωτερικών χώρων μέσω διαδικτύου.

Στο τρίτο κεφάλαιο παρουσιάζονται οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος. Γίνεται αναφορά στην Python, τη Flutter για την ανάπτυξη της εφαρμογής κινητού, τη MySQL για τη διαχείριση της βάσης δεδομένων, το ESP32 για την επικοινωνία με τους αισθητήρες, καθώς και στους αισθητήρες και ενεργοποιητές που χρησιμοποιήθηκαν.
Το τέταρτο κεφάλαιο επικεντρώνεται στην ανάλυση του συστήματος που αναπτύχθηκε. Εδώ παρουσιάζεται αναλυτικά ο κώδικας που χρησιμοποιήθηκε για το ESP32, η βάση δεδομένων, καθώς και τα μέτρα ασφαλείας που έχουν ληφθεί για την προστασία των δεδομένων.
Στο τελευταίο κεφάλαιο της εργασίας παρουσιάζονται τα συμπεράσματα από την ανάπτυξη του συστήματος, καθώς και προτάσεις για μελλοντικές βελτιώσεις και επεκτάσεις, με έμφαση στην ασφάλεια, την επεκτασιμότητα και την ευχρηστία του συστήματος.
Το Παράρτημα περιλαμβάνει επιπλέον πληροφορίες και τεχνικές λεπτομέρειες που αφορούν την υλοποίηση του συστήματος και τα εργαλεία που χρησιμοποιήθηκαν.

Κεφάλαιο 2ο: Παρόμοια Συστήματα

2.1 Εισαγωγή

Τα συστήματα παρακολούθησης και ελέγχου μέσω διαδικτύου αποτελούν έναν τομέα με συνεχώς αυξανόμενη ζήτηση. Η ανάπτυξη συστημάτων που επιτρέπουν τον έλεγχο και την παρακολούθηση συσκευών και αισθητήρων από απόσταση είναι πλέον κοινή πρακτική σε πολλές εφαρμογές, όπως η ασφάλεια, η έξυπνη διαχείριση κτιρίων και η βιομηχανική αυτοματοποίηση.

2.2 Nest Secure by Google

Το Nest Secure είναι ένα ολοκληρωμένο σύστημα ασφαλείας της Google που περιλαμβάνει αισθητήρες, κάμερες και μηχανισμούς ενεργοποίησης/απενεργοποίησης. Το σύστημα συνδέεται με το διαδίκτυο και παρέχει απομακρυσμένη παρακολούθηση και έλεγχο μέσω της εφαρμογής Google Home. Το Nest Secure χρησιμοποιείται κυρίως για την προστασία του σπιτιού, με αισθητήρες που εντοπίζουν κίνηση και παραβιάσεις σε πόρτες και παράθυρα. [1]

Εικόνα 2.1: Google Nest

[https://upload.wikimedia.org/wikipedia/commons/thumb/0/0c/GoogleNestRetailStand.jpg/220px-GoogleNestRetailStand.jpg]

Αισθητήρες Κίνησης και Παραθύρων: Οι αισθητήρες του Nest Secure παρακολουθούν την κίνηση μέσα στο σπίτι, καθώς και την κατάσταση των παραθύρων και των θυρών, ειδοποιώντας τον χρήστη για οποιαδήποτε ύποπτη δραστηριότητα.

Εφαρμογή Google Home: Μέσω της εφαρμογής, οι χρήστες μπορούν να ελέγχουν το σύστημα απομακρυσμένα, να ενεργοποιούν ή να απενεργοποιούν τον συναγερμό, και να λαμβάνουν ειδοποιήσεις σε πραγματικό χρόνο για την κατάσταση των αισθητήρων.

Κάμερες Ασφαλείας: Οι κάμερες που συνδέονται με το σύστημα παρέχουν ζωντανή ροή βίντεο από το σπίτι, επιτρέποντας την παρακολούθηση οποιαδήποτε στιγμή και από οποιοδήποτε μέρος.

Αυτόματος Έλεγχος: Το σύστημα μπορεί να προσαρμόζεται αυτόματα, ενεργοποιώντας τον συναγερμό όταν ανιχνεύσει απουσία των κατοίκων.

Πλεονεκτήματα:

Ασφάλεια μέσω της Google: Η αξιοπιστία της Google στην αποθήκευση και διαχείριση δεδομένων προσφέρει υψηλό επίπεδο ασφάλειας και εμπιστοσύνης στους χρήστες.

Ευκολία στη χρήση: Το σύστημα είναι πολύ φιλικό προς τον χρήστη, με μια εύκολη στην πλοήγηση εφαρμογή και τη δυνατότητα ελέγχου μέσω φωνητικών εντολών.

Επεκτασιμότητα: Το Nest Secure επιτρέπει την προσθήκη επιπλέον συσκευών, όπως κάμερες και αισθητήρες, για να καλύψει διαφορετικές ανάγκες ασφαλείας.

Το Nest Secure και το σύστημα που αναπτύχθηκε έχουν ορισμένες ομοιότητες στη λειτουργικότητα, κυρίως στο κομμάτι της απομακρυσμένης παρακολούθησης και ελέγχου μέσω μιας εφαρμογής κινητού. Αμφότερα τα συστήματα επιτρέπουν τη διαχείριση αισθητήρων και την αποθήκευση δεδομένων σε έναν κεντρικό server. Η βασική διαφορά έγκειται στην εμπορική εφαρμογή του Nest Secure, το οποίο είναι πλήρως ενσωματωμένο στην υποδομή της Google και υποστηρίζει ευρύτερες δυνατότητες, όπως η χρήση καμερών και φωνητικών εντολών μέσω του Google Assistant. Αντίθετα, το σύστημα που αναπτύξαμε είναι πιο προσαρμόσιμο και μπορεί να παραμετροποιηθεί σύμφωνα με τις συγκεκριμένες ανάγκες του εκάστοτε χρήστη, παρέχοντας μεγαλύτερη ευελιξία στην ενσωμάτωση διαφορετικών αισθητήρων και συσκευών.

2.3 Ring Alarm Security System by Amazon

Το Ring Alarm είναι ένα έξυπνο σύστημα ασφαλείας που έχει αναπτύξει η Amazon. Πρόκειται για ένα ολοκληρωμένο σύστημα παρακολούθησης, που παρέχει απομακρυσμένη πρόσβαση και έλεγχο μέσω της εφαρμογής Ring και υποστηρίζεται από την πλατφόρμα Amazon Alexa. Το Ring Alarm σχεδιάστηκε κυρίως για οικιακή χρήση, δίνοντας έμφαση στην προστασία του σπιτιού και την παρακολούθηση των αισθητήρων σε πραγματικό χρόνο. [2]

Εικόνα 2.2: Ring Alarm Security System by Amazon

[https://m.media-amazon.com/images/I/41HvWJtAUTL._AC_UY218_.jpg]

Αισθητήρες Κίνησης και Θυρών/Παραθύρων: Το Ring Alarm περιλαμβάνει αισθητήρες που παρακολουθούν την κίνηση μέσα στο σπίτι, καθώς και αισθητήρες που ανιχνεύουν το άνοιγμα θυρών και παραθύρων. Μόλις ανιχνευθεί κάποια δραστηριότητα, το σύστημα ειδοποιεί άμεσα τον χρήστη μέσω της εφαρμογής.

Εφαρμογή Ring: Μέσω της εφαρμογής, οι χρήστες μπορούν να ελέγχουν εξ αποστάσεως την κατάσταση του συστήματος, να ενεργοποιούν ή να απενεργοποιούν τον συναγερμό, και να λαμβάνουν ειδοποιήσεις σε πραγματικό χρόνο.

Κάμερες και Βιντεοθυροτηλέφωνα: Το Ring Alarm συνεργάζεται με μια σειρά από εξωτερικές και εσωτερικές κάμερες ασφαλείας, καθώς και με βιντεοθυροτηλέφωνα που προσφέρουν ζωντανή ροή βίντεο και δυνατότητα επικοινωνίας μέσω της εφαρμογής.

Φωνητικός Έλεγχος μέσω Alexa: Το σύστημα είναι πλήρως ενσωματωμένο με την Amazon Alexa, επιτρέποντας στους χρήστες να ελέγχουν τις συσκευές και τους αισθητήρες μέσω φωνητικών εντολών, όπως την ενεργοποίηση ή απενεργοποίηση του συναγερμού.

Επαγγελματική Παρακολούθηση: Μία από τις επιλογές του Ring είναι η δυνατότητα επαγγελματικής παρακολούθησης 24/7. Σε περίπτωση έκτακτου συμβάντος, ειδοποιούνται οι υπηρεσίες ασφαλείας.

Πλεονεκτήματα:

Ευκολία στη χρήση: Η εφαρμογή Ring προσφέρει ένα φιλικό προς τον χρήστη περιβάλλον, με άμεση πρόσβαση στα δεδομένα των αισθητήρων και την κατάσταση του σπιτιού.

Υποστήριξη μέσω Alexa: Η φωνητική εντολή προσφέρει ευκολία και ταχύτητα στη διαχείριση του συστήματος, χωρίς να χρειάζεται φυσική πρόσβαση στο κινητό τηλέφωνο.

Επαγγελματική παρακολούθηση: Το Ring Alarm προσφέρει την επιλογή για επαγγελματική παρακολούθηση, καθιστώντας το ιδανικό για όσους επιθυμούν επιπλέον ασφάλεια και άμεση απόκριση σε περίπτωση παραβίασης.

Και τα δύο συστήματα επιτρέπουν τη διαχείριση συσκευών μέσω κινητού τηλεφώνου και την αποθήκευση δεδομένων στον server. Ωστόσο, το Ring Alarm παρέχει επαγγελματική παρακολούθηση 24/7 και την υποστήριξη Alexa, κάτι που το καθιστά πιο ολοκληρωμένο ως εμπορική λύση ασφαλείας. Το σύστημα που αναπτύχθηκε, από την άλλη πλευρά, προσφέρει μεγαλύτερη ευελιξία και δυνατότητες προσαρμογής σε συγκεκριμένες ανάγκες του χρήστη, επιτρέποντας την ενσωμάτωση πολλών διαφορετικών ειδών αισθητήρων και συσκευών.

2.4 SimpliSafe Home Security System

Εικόνα 2.3: SimpliSafe Home Security System

[https://i.pcmag.com/imagery/reviews/00z89jGGstKuKqAZmXIFhw1-14..v1569478468.jpg]

Το SimpliSafe είναι ένα εύκολο στη χρήση και εγκατάσταση σύστημα ασφαλείας για το σπίτι, το οποίο προσφέρει ολοκληρωμένες λύσεις ασφαλείας μέσω απομακρυσμένης διαχείρισης και παρακολούθησης. Το σύστημα SimpliSafe είναι γνωστό για την απλότητα στην εγκατάσταση, καθώς δεν απαιτεί επαγγελματική εγκατάσταση και μπορεί να ρυθμιστεί γρήγορα από τον χρήστη. Το

σύστημα αυτό περιλαμβάνει αισθητήρες κίνησης, παραθύρων/θυρών, κάμερες και ανιχνευτές καπνού/πλημμύρας, καλύπτοντας κάθε ανάγκη ασφαλείας.

- **Αισθητήρες Θυρών/Παραθύρων και Κίνησης**: Οι αισθητήρες του SimpliSafe παρακολουθούν την κίνηση μέσα στο σπίτι και ειδοποιούν άμεσα τον χρήστη σε περίπτωση που ανιχνευτεί δραστηριότητα σε πόρτες ή παράθυρα. Οι αισθητήρες κίνησης καλύπτουν μεγάλες περιοχές και είναι ευαίσθητοι σε ανθρώπινη κίνηση.
- **Ανιχνευτές Καπνού και Πλημμύρας**: Το σύστημα διαθέτει ανιχνευτές καπνού και πλημμύρας που προσφέρουν επιπλέον επίπεδα προστασίας σε περίπτωση φυσικών καταστροφών ή απρόοπτων συμβάντων.
- **Κάμερες Ασφαλείας**: Οι κάμερες που ενσωματώνονται στο σύστημα παρέχουν συνεχή ροή βίντεο και επιτρέπουν στον χρήστη να παρακολουθεί ζωντανά τους εσωτερικούς και εξωτερικούς χώρους του σπιτιού.
- **Απομακρυσμένη Διαχείριση μέσω Εφαρμογής**: Η εφαρμογή SimpliSafe δίνει στους χρήστες τη δυνατότητα να ενεργοποιούν και να απενεργοποιούν τον συναγερμό, να παρακολουθούν ζωντανά τη ροή βίντεο από τις κάμερες, καθώς και να λαμβάνουν ειδοποιήσεις σε πραγματικό χρόνο για την κατάσταση των αισθητήρων.
- **Επαγγελματική Παρακολούθηση 24/7**: Το σύστημα υποστηρίζει επίσης επαγγελματική παρακολούθηση 24 ώρες το 24ωρο, με αυτόματες ειδοποιήσεις στις αρχές σε περίπτωση παραβίασης.

Πλεονεκτήματα:

- **Εύκολη εγκατάσταση**: Το SimpliSafe μπορεί να εγκατασταθεί από τον ίδιο τον χρήστη, χωρίς την ανάγκη για επαγγελματική βοήθεια.
- **Ευελιξία**: Το σύστημα είναι ιδιαίτερα ευέλικτο, καθώς επιτρέπει την προσθήκη επιπλέον αισθητήρων και συσκευών, ανάλογα με τις ανάγκες του χρήστη.
- **Προστασία πολλαπλών επιπέδων**: Εκτός από τους παραδοσιακούς αισθητήρες κίνησης και θυρών, το SimpliSafe παρέχει επιπλέον ανιχνευτές καπνού και πλημμύρας, προσφέροντας μια ολοκληρωμένη λύση ασφαλείας.

Το SimpliSafe παρουσιάζει ομοιότητες με το σύστημα που αναπτύχθηκε, καθώς και τα δύο συστήματα παρέχουν τη δυνατότητα παρακολούθησης αισθητήρων και απομακρυσμένου ελέγχου μέσω κινητής εφαρμογής. Και τα δύο συστήματα επιτρέπουν την άμεση ειδοποίηση του χρήστη σε περίπτωση παραβίασης ή ανίχνευσης κίνησης, και υποστηρίζουν την προσθήκη νέων αισθητήρων ανάλογα με τις ανάγκες του χρήστη. Ωστόσο, το SimpliSafe προσφέρει επιπλέον δυνατότητες, όπως οι ανιχνευτές καπνού και πλημμύρας, καθώς και η επαγγελματική παρακολούθηση 24/7.

Το SimpliSafe αποτελεί μια εξαιρετική λύση για χρήστες που επιθυμούν ένα ολοκληρωμένο και εύκολα διαχειρίσιμο σύστημα ασφαλείας, με πολλές δυνατότητες και επιπλέον προστασία από φυσικές καταστροφές.

Κεφάλαιο 3ο: Γλώσσες προγραμματισμού και εργαλεία

3.1 Python

Η Python είναι μια από τις πιο δημοφιλείς και ισχυρές γλώσσες προγραμματισμού στον κόσμο. Δημιουργήθηκε από τον Guido van Rossum το 1989, και η πρώτη της κυκλοφορία έγινε το 1991. Η αρχική πρόθεση πίσω από τη δημιουργία της Python ήταν να είναι μια γλώσσα που είναι εύκολη στην ανάγνωση και τη γραφή, επιτρέποντας την ταχεία ανάπτυξη λογισμικού και την εύκολη κατανόηση από προγραμματιστές. Το όνομά της δεν προέρχεται από το φίδι, όπως πολλοί νομίζουν, αλλά από την βρετανική κωμική σειρά "Monty Python's Flying Circus", καθώς ο δημιουργός της ήθελε να αναπτύξει μια γλώσσα που να είναι και διασκεδαστική στη χρήση. [3,4]

Χαρακτηριστικά της Python

Η Python έχει μερικά βασικά χαρακτηριστικά που την κάνουν να ξεχωρίζει:

- **Απλότητα**: Η Python δίνει έμφαση στην αναγνωσιμότητα του κώδικα. Οι εντολές της γλώσσας είναι συχνά ευανάγνωστες, καθιστώντας τον κώδικα πιο κατανοητό ακόμη και για αρχάριους.
- **Δυναμική τυποποίηση**: Σε αντίθεση με άλλες γλώσσες όπως η C ή η Java, η Python δεν απαιτεί τον προγραμματιστή να δηλώσει ρητά τον τύπο των μεταβλητών. Η Python καθορίζει τους τύπους κατά τη διάρκεια εκτέλεσης.
- **Διαλειτουργικότητα**: Η Python μπορεί να ενσωματωθεί ή να χρησιμοποιηθεί μαζί με άλλες γλώσσες, όπως η C ή η Java. Αυτό επιτρέπει στους προγραμματιστές να αξιοποιούν υπάρχουσες βιβλιοθήκες και εργαλεία από άλλες γλώσσες.
- **Αντικειμενοστραφής και Δομημένη**: Η Python υποστηρίζει και αντικειμενοστραφή και δομημένη προγραμματιστική προσέγγιση, καθιστώντας την ευέλικτη για διάφορα είδη έργων.
- **Εκτενής Βιβλιοθήκη**: Η Python έχει μια μεγάλη στάνταρτ βιβλιοθήκη που καλύπτει τα πάντα, από χειρισμό αρχείων, επεξεργασία κειμένου, μέχρι εργασία με διαδίκτυο και βάσεις δεδομένων.
- **Υποστήριξη πολλαπλών πλατφορμών**: Μπορεί να τρέξει σε μια μεγάλη ποικιλία πλατφορμών, συμπεριλαμβανομένων των Windows, macOS, Linux, και άλλων.

Χρήσεις και Εφαρμογές της Python

Η Python είναι μια γλώσσα πολλαπλών χρήσεων που χρησιμοποιείται σε πληθώρα εφαρμογών:

- **Επιστήμη Δεδομένων**: Λόγω της ύπαρξης βιβλιοθηκών όπως οι NumPy, pandas, και SciPy, η Python είναι πολύ δημοφιλής στην ανάλυση δεδομένων, στατιστική, και μηχανική μάθηση.
- **Ανάπτυξη Ιστού**: Η Python χρησιμοποιείται ευρέως για ανάπτυξη ιστοσελίδων και εφαρμογών ιστού μέσω εργαλείων όπως τα Django και Flask.
- **Αυτοματοποίηση Διεργασιών (Scripting)**: Πολλές εργασίες αυτοματοποίησης και διαχείρισης συστημάτων μπορούν να γραφτούν σε Python.
- **Τεχνητή Νοημοσύνη**: Με τη χρήση βιβλιοθηκών όπως οι TensorFlow και Keras, η Python παίζει κεντρικό ρόλο στην ανάπτυξη αλγορίθμων μηχανικής μάθησης και τεχνητής νοημοσύνης.
- **Διαχείριση Δεδομένων και Διασυνδέσεις Βάσεων**: Η Python υποστηρίζει τη σύνδεση με βάσεις δεδομένων, όπως η MySQL, SQL Server, και PostgreSQL, παρέχοντας ευέλικτους τρόπους αλληλεπίδρασης με δεδομένα.
- **Γραφικά και Παιχνίδια**: Η Pygame είναι μια βιβλιοθήκη που επιτρέπει τη δημιουργία απλών παιχνιδιών και γραφικών εφαρμογών.

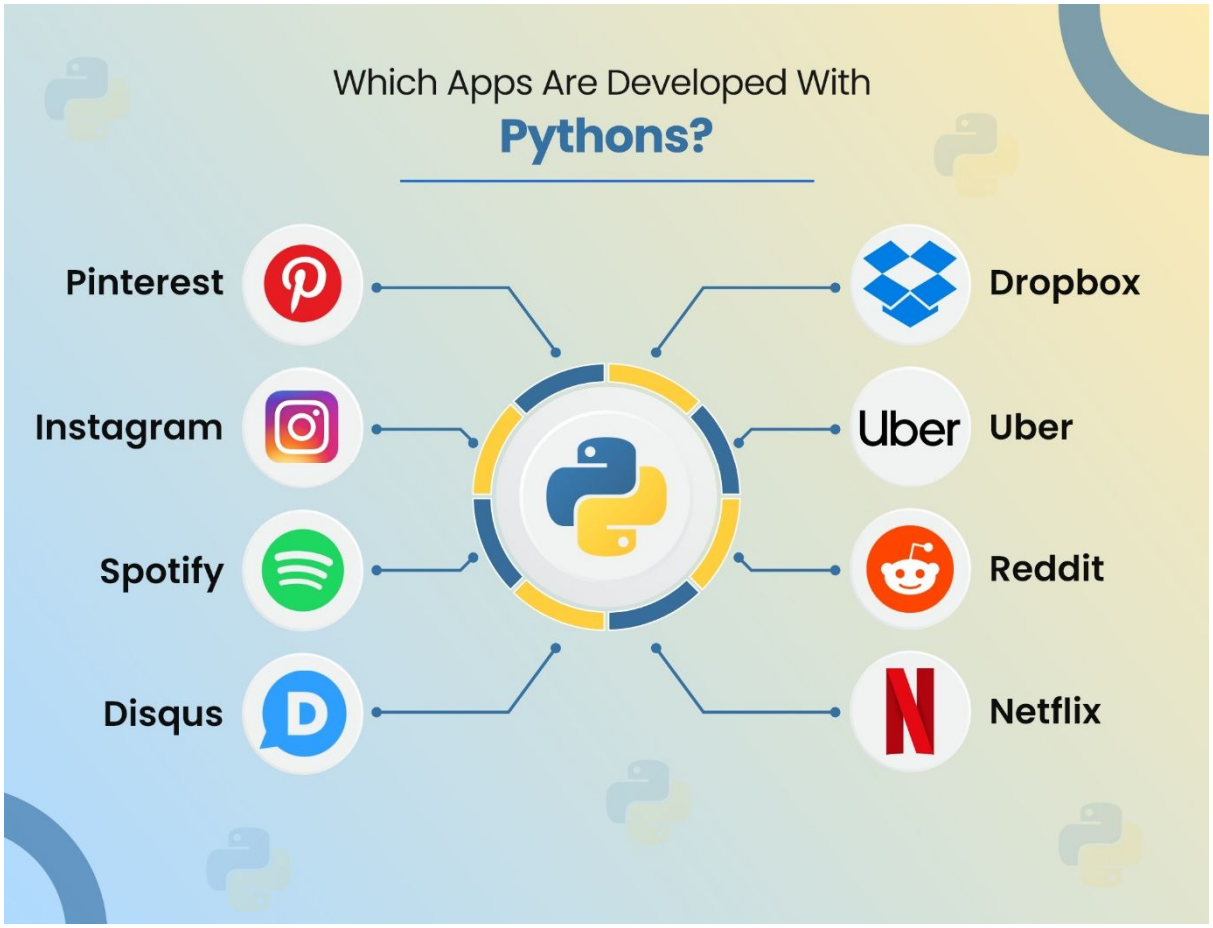
Πλεονεκτήματα της Python

- **Ευκολία Μάθησης και Χρήσης**: Η Python είναι γνωστή για την απλή σύνταξή της, που καθιστά εύκολη τη μάθηση ακόμα και για αρχάριους.
- **Εκτεταμένη Κοινότητα**: Υπάρχει μια πολύ μεγάλη και δραστήρια κοινότητα χρηστών, κάτι που εξασφαλίζει μεγάλη υποστήριξη και συνεχή βελτίωση.
- **Πολλαπλές Βιβλιοθήκες**: Υπάρχουν χιλιάδες βιβλιοθήκες για σχεδόν οποιαδήποτε εφαρμογή ή πρόβλημα που μπορεί να αντιμετωπίσει ένας προγραμματιστής.
- **Πλατφορμική Ανεξαρτησία**: Η Python μπορεί να τρέξει σχεδόν σε οποιαδήποτε πλατφόρμα, χωρίς μεγάλες τροποποιήσεις στον κώδικα.

Μειονεκτήματα της Python

- **Απόδοση**: Η Python είναι πιο αργή σε σχέση με γλώσσες όπως η C ή η Java, καθώς είναι ερμηνευόμενη γλώσσα. Αυτό μπορεί να είναι περιοριστικός παράγοντας για εφαρμογές που απαιτούν μεγάλη απόδοση.
- **Κατανάλωση Μνήμης**: Η Python μπορεί να χρησιμοποιεί περισσότερη μνήμη σε σύγκριση με άλλες γλώσσες, ειδικά σε μεγάλα και πολύπλοκα έργα.

- **Περιορισμοί Κινητών Εφαρμογών**: Η Python δεν χρησιμοποιείται ευρέως για ανάπτυξη εφαρμογών κινητών, παρά την ύπαρξη εργαλείων όπως το Kivy.



Εικόνα 3.1: Ποιοι χρησιμοποιούν Python

[https://www.apptunix.com/blog/wp-content/uploads/sites/3/2021/08/popular-apps-doveloped-with-python.jpg]

3.1.1 Σύγκριση της Python με τις γλώσσες C++ και Java

Η Python, η C++, και η Java είναι τρεις από τις πιο γνωστές γλώσσες προγραμματισμού, και η καθεμία έχει τα δικά της πλεονεκτήματα, μειονεκτήματα και χαρακτηριστικά που την καθιστούν κατάλληλη για διαφορετικά είδη εφαρμογών. [5,6]

- **Python**: Όπως αναφέρθηκε νωρίτερα, η Python είναι μια γλώσσα υψηλού επιπέδου με έμφαση στην απλότητα και την αναγνωσιμότητα του κώδικα. Είναι ερμηνευόμενη γλώσσα, κάτι που σημαίνει ότι ο κώδικας εκτελείται γραμμή προς γραμμή χωρίς να χρειάζεται μεταγλώττιση.

- **C++**: Η C++ είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού υψηλού επιπέδου που προέρχεται από τη γλώσσα C. Αναπτύχθηκε τη δεκαετία του 1980 από τον Bjarne Stroustrup και προσφέρει μεγάλη ελέγχου μνήμης και ταχύτητα εκτέλεσης. Η C++ είναι μια μεταγλωττισμένη γλώσσα (compiled language), που σημαίνει ότι ο κώδικας μεταγλωττίζεται σε δυαδική μορφή πριν την εκτέλεση.
- **Java**: Η Java δημιουργήθηκε από την Sun Microsystems το 1995 και σχεδιάστηκε για να είναι ανεξάρτητη από την πλατφόρμα μέσω της χρήσης της εικονικής μηχανής Java (JVM). Είναι επίσης μια αντικειμενοστραφής γλώσσα και χαρακτηρίζεται από τη μεταγλώττιση σε bytecode που εκτελείται σε οποιοδήποτε σύστημα έχει εγκατεστημένη τη JVM.

Ταχύτητα και Απόδοση

- **Python**: Η Python είναι γενικά πιο αργή από τις C++ και Java, καθώς είναι ερμηνευόμενη γλώσσα. Η δυναμική της φύση (dynamic typing) και η έλλειψη αυστηρής διαχείρισης της μνήμης την καθιστούν λιγότερο αποδοτική σε εφαρμογές που απαιτούν μεγάλες επιδόσεις. Παρόλα αυτά, οι βιβλιοθήκες όπως οι NumPy και Cython μπορούν να αυξήσουν σημαντικά την απόδοση.
- **C++**: Η C++ είναι γνωστή για την ταχύτητά της. Επειδή είναι μεταγλωττισμένη γλώσσα και δίνει στους προγραμματιστές άμεσο έλεγχο της διαχείρισης μνήμης (μέσω pointers και manual allocation), θεωρείται από τις πιο γρήγορες γλώσσες προγραμματισμού. Χρησιμοποιείται συχνά σε εφαρμογές που απαιτούν υψηλές επιδόσεις, όπως παιχνίδια, εφαρμογές πραγματικού χρόνου και συστήματα λειτουργίας.
- **Java**: Η Java είναι πιο αργή από την C++ αλλά πιο γρήγορη από την Python, καθώς είναι μεταγλωττισμένη σε bytecode που τρέχει σε μια εικονική μηχανή (JVM). Η Java κάνει αυτόματη διαχείριση μνήμης μέσω της garbage collection, κάτι που βοηθά στην απλότητα του προγραμματισμού, αλλά μπορεί να μειώσει την απόδοση σε κάποιες περιπτώσεις.

Εύκολη Μάθηση και Χρήση

- **Python**: Η Python είναι η πιο εύκολη γλώσσα από τις τρεις για να μάθει κανείς, κυρίως λόγω της απλής σύνταξης και της αναγνωσιμότητας του κώδικα. Οι αρχάριοι προγραμματιστές συχνά ξεκινούν με την Python, επειδή δεν απαιτεί μεγάλη τεχνική γνώση για να κατανοήσει κάποιος τις βασικές της έννοιες.
- **C++**: Η C++ θεωρείται πιο δύσκολη γλώσσα, καθώς απαιτεί από τον προγραμματιστή να κατανοήσει περίπλοκες έννοιες, όπως pointers, manual memory management, και low-level

προγραμματισμό. Ωστόσο, προσφέρει μεγάλη ισχύ και ευελιξία, κάτι που την καθιστά δημοφιλή για προχωρημένους προγραμματιστές.

- **Java**: Η Java βρίσκεται κάπου στη μέση. Έχει πιο περίπλοκη σύνταξη από την Python αλλά είναι πιο εύκολη σε σχέση με την C++, κυρίως λόγω της διαχείρισης μνήμης που γίνεται αυτόματα και της απλοποίησης του προγραμματισμού με τις βιβλιοθήκες της.

Αντικειμενοστραφής Προγραμματισμός

- **Python**: Η Python υποστηρίζει αντικειμενοστραφή προγραμματισμό, αλλά επιτρέπει επίσης την δομημένη ή λειτουργική προσέγγιση. Είναι λιγότερο αυστηρή όσον αφορά την αντικειμενοστραφή προσέγγιση σε σχέση με τη C++ και τη Java.
- **C++**: Η C++ υποστηρίζει πλήρως τον αντικειμενοστραφή προγραμματισμό και θεωρείται από τις πιο ισχυρές γλώσσες για την ανάπτυξη τέτοιων συστημάτων. Δίνει τη δυνατότητα στον προγραμματιστή να δημιουργεί πολύπλοκα συστήματα με κλάσεις, κληρονομικότητα, και πολυμορφισμό, αλλά απαιτεί μεγαλύτερη προσοχή στη διαχείριση των πόρων.
- **Java**: Η Java είναι επίσης μια πλήρως αντικειμενοστραφής γλώσσα. Σε αντίθεση με την C++, στην Java τα πάντα είναι αντικείμενα, και η διαχείριση της μνήμης γίνεται αυτόματα μέσω του garbage collector. Αυτό την καθιστά πιο εύκολη για προγραμματιστές που θέλουν να επικεντρωθούν στην επιχειρησιακή λογική χωρίς να ασχολούνται με λεπτομέρειες της διαχείρισης πόρων.

Διαλειτουργικότητα και Πλατφόρμες

- **Python**: Η Python είναι πλατφορμικά ανεξάρτητη και μπορεί να τρέξει σε οποιοδήποτε σύστημα, αρκεί να έχει εγκατασταθεί ο κατάλληλος ερμηνευτής. Η Python μπορεί να συνεργαστεί με άλλες γλώσσες μέσω βιβλιοθηκών, αλλά η ταχύτητα ενσωμάτωσης μπορεί να είναι περιορισμός.
- **C++**: Η C++ είναι επίσης ανεξάρτητη πλατφορμικά, αλλά η μεταγλώττιση σε διαφορετικά συστήματα απαιτεί συχνά προσαρμογές. Είναι η πιο συμβατή γλώσσα με άλλες χαμηλού επιπέδου γλώσσες και χρησιμοποιείται συχνά για την ανάπτυξη λογισμικού που τρέχει κοντά στο υλικό (hardware).
- **Java**: Ένα από τα κύρια πλεονεκτήματα της Java είναι η πλατφορμική ανεξαρτησία της, καθώς ο κώδικας μεταγλωττίζεται σε bytecode και τρέχει στην εικονική μηχανή (JVM), επιτρέποντάς του να εκτελείται σε οποιοδήποτε λειτουργικό σύστημα που υποστηρίζει JVM.

Χρήσεις και Εφαρμογές

- **Python**: Χρησιμοποιείται ευρέως σε εφαρμογές επιστήμης δεδομένων, μηχανικής μάθησης, ανάλυσης δεδομένων, αυτοματοποίησης, ανάπτυξης ιστού και scripting.
- **C++**: Χρησιμοποιείται για την ανάπτυξη παιχνιδιών, λογισμικού συστημάτων, εφαρμογών πραγματικού χρόνου, συστημάτων ενσωματωμένου προγραμματισμού και εφαρμογών υψηλών επιδόσεων.
- **Java**: Χρησιμοποιείται κυρίως για ανάπτυξη επιχειρησιακών εφαρμογών, εφαρμογών Android, ανάπτυξη ιστού και λογισμικού που απαιτεί ανθεκτικότητα και αξιοπιστία.

Πλεονεκτήματα και Μειονεκτήματα

Πίνακας 1

Κατηγορία	**Python**	**C++**	**Java**
Ταχύτητα Εκτέλεσης	Χαμηλή	Πολύ υψηλή	Μέση
Ευκολία Μάθησης	Πολύ εύκολη	Δύσκολη	Μέτρια
Διαχείριση Μνήμης	Αυτόματη	Χειροκίνητη	Αυτόματη
Αντικειμενοστραφής	Προαιρετική	Πολύ ισχυρή	Αποκλειστική
Διαλειτουργικότητα	Πολύ καλή με άλλες γλώσσες	Εξαιρετική με χαμηλού επιπέδου γλώσσες	Ανεξαρτησία πλατφόρμας με JVM
Χρήσεις	Ανάλυση δεδομένων, ανάπτυξη ιστού	Παιχνίδια, εφαρμογές πραγματικού χρόνου	Επιχειρησιακές εφαρμογές, Android

Η Python, η C++ και η Java έχουν διαφορετικούς σκοπούς και χρήσεις. Η Python είναι ιδανική για γρήγορη ανάπτυξη και εφαρμογές επιστήμης δεδομένων, η C++ προσφέρει υψηλή απόδοση για εφαρμογές συστημάτων και παιχνίδια, ενώ η Java παρέχει σταθερότητα και ανθεκτικότητα για επιχειρησιακές εφαρμογές και ανάπτυξη κινητών. Κάθε γλώσσα έχει τα δικά της πλεονεκτήματα και μειονεκτήματα, και η επιλογή μεταξύ αυτών εξαρτάται από τις ανάγκες του έργου.

3.1.2 API Python

Το **API** (Application Programming Interface) είναι ένα σύνολο κανόνων και πρωτοκόλλων που επιτρέπει σε διαφορετικές εφαρμογές ή υπηρεσίες να επικοινωνούν μεταξύ τους. Με άλλα λόγια, το

API λειτουργεί σαν γέφυρα μεταξύ διαφορετικών λογισμικών και συστημάτων. Μέσω του API, μια εφαρμογή μπορεί να ζητήσει δεδομένα ή υπηρεσίες από μια άλλη εφαρμογή ή υπηρεσία, χωρίς να χρειάζεται να γνωρίζει λεπτομέρειες για το εσωτερικό της.

Υπάρχουν διάφοροι τύποι API, όπως:

- **Web APIs**: Χρησιμοποιούνται για την επικοινωνία μεταξύ συστημάτων που είναι συνδεδεμένα στο διαδίκτυο, συνήθως μέσω των πρωτοκόλλων HTTP/HTTPS.
- **Library APIs**: Παρέχονται από βιβλιοθήκες λογισμικού, και επιτρέπουν στον προγραμματιστή να χρησιμοποιεί τις λειτουργίες τους μέσα από τον κώδικά του.

Συνολικά, το API επιτρέπει την ενσωμάτωση και τη διαλειτουργικότητα μεταξύ διαφορετικών συστημάτων, βελτιώνοντας τη λειτουργικότητα και την επεκτασιμότητα.

Η Python είναι εξαιρετική για την εργασία με APIs χάρη στις βιβλιοθήκες της, όπως η requests, η οποία απλοποιεί την αποστολή αιτήσεων (requests) και τη λήψη απαντήσεων (responses) από APIs. Η Python μπορεί να αλληλεπιδράσει με εξωτερικά APIs, είτε στέλνοντας δεδομένα μέσω POST requests, είτε ζητώντας δεδομένα μέσω GET requests.

Ας δούμε ένα ολοκληρωμένο παράδειγμα όπου κάνουμε μια **GET** αίτηση για να πάρουμε δεδομένα από ένα API και μια **POST** αίτηση για να στείλουμε δεδομένα σε ένα API.

Πρώτα, θα χρησιμοποιήσουμε τη βιβλιοθήκη **requests**, η οποία είναι πολύ απλή στη χρήση για αιτήσεις HTTP.

Για να χρησιμοποιήσουμε τη βιβλιοθήκη **requests**, μπορούμε να την εγκαταστήσουμε με την εντολή:

pip install requests

και ένα παράδειγμα με εξήγηση:

```
# Εισαγωγή της βιβλιοθήκης requests
import requests


# ---------------------------------------
# Παράδειγμα GET αίτησης (GET request)
# ---------------------------------------


# Καθορίζουμε τη διεύθυνση URL από όπου θα πάρουμε τα δεδομένα
url_get = "https://jsonplaceholder.typicode.com/posts"


# Κάνουμε την GET αίτηση
response_get = requests.get(url_get)
```

```
# Ελέγχουμε αν η αίτηση ήταν επιτυχής (κωδικός 200 σημαίνει επιτυχία)
if response_get.status_code == 200:
    # Εκτυπώνουμε τα δεδομένα που λάβαμε σε μορφή JSON
    print("Αποτελέσματα GET αιτήματος:")
    print(response_get.json())
else:
    # Αν η αίτηση αποτύχει, εκτυπώνουμε το σφάλμα
    print(f"Σφάλμα GET αιτήματος: {response_get.status_code}")

# --------------------------------------
# Παράδειγμα POST αίτησης (POST request)
# --------------------------------------

# Καθορίζουμε τη διεύθυνση URL όπου θα στείλουμε τα δεδομένα
url_post = "https://jsonplaceholder.typicode.com/posts"

# Καθορίζουμε τα δεδομένα που θα στείλουμε
data = {
    "title": "Πρόγραμμα σε Python",
    "body": "Παράδειγμα POST αίτησης με χρήση της Python.",
    "userId": 1
}

# Κάνουμε την POST αίτηση
response_post = requests.post(url_post, json=data)

# Ελέγχουμε αν η αίτηση ήταν επιτυχής (κωδικός 201 σημαίνει ότι δημιουργήθηκε κάτι νέο)
if response_post.status_code == 201:
    # Εκτυπώνουμε τα δεδομένα της απάντησης (response)
    print("Αποτελέσματα POST αιτήματος:")
    print(response_post.json())
else:
    # Αν η αίτηση αποτύχει, εκτυπώνουμε το σφάλμα
    print(f"Σφάλμα POST αιτήματος: {response_post.status_code}")
```

1. **GET Αίτηση**:
    - Χρησιμοποιούμε την μέθοδο requests.get() για να στείλουμε μια αίτηση στη διεύθυνση URL https://jsonplaceholder.typicode.com/posts.

- Η μέθοδος αυτή ζητά δεδομένα από τον διακομιστή και, εφόσον η αίτηση είναι επιτυχής (status code 200), επιστρέφει τα δεδομένα σε μορφή JSON.
- Αν η αίτηση αποτύχει, τυπώνουμε το σφάλμα.

2. **POST Αίτηση**:
   - Χρησιμοποιούμε την μέθοδο requests.post() για να στείλουμε δεδομένα σε έναν διακομιστή στη διεύθυνση URL https://jsonplaceholder.typicode.com/posts.
   - Τα δεδομένα αποστέλλονται σε μορφή JSON χρησιμοποιώντας το όρισμα json=data, όπου data είναι ένα dictionary με τα δεδομένα που θέλουμε να στείλουμε.
   - Εάν η αίτηση είναι επιτυχής και κάτι νέο δημιουργηθεί (status code 201), επιστρέφεται η απάντηση από τον διακομιστή, που περιέχει τα δεδομένα που αποθηκεύτηκαν.
   - Αν η αίτηση αποτύχει, εκτυπώνουμε το σφάλμα.

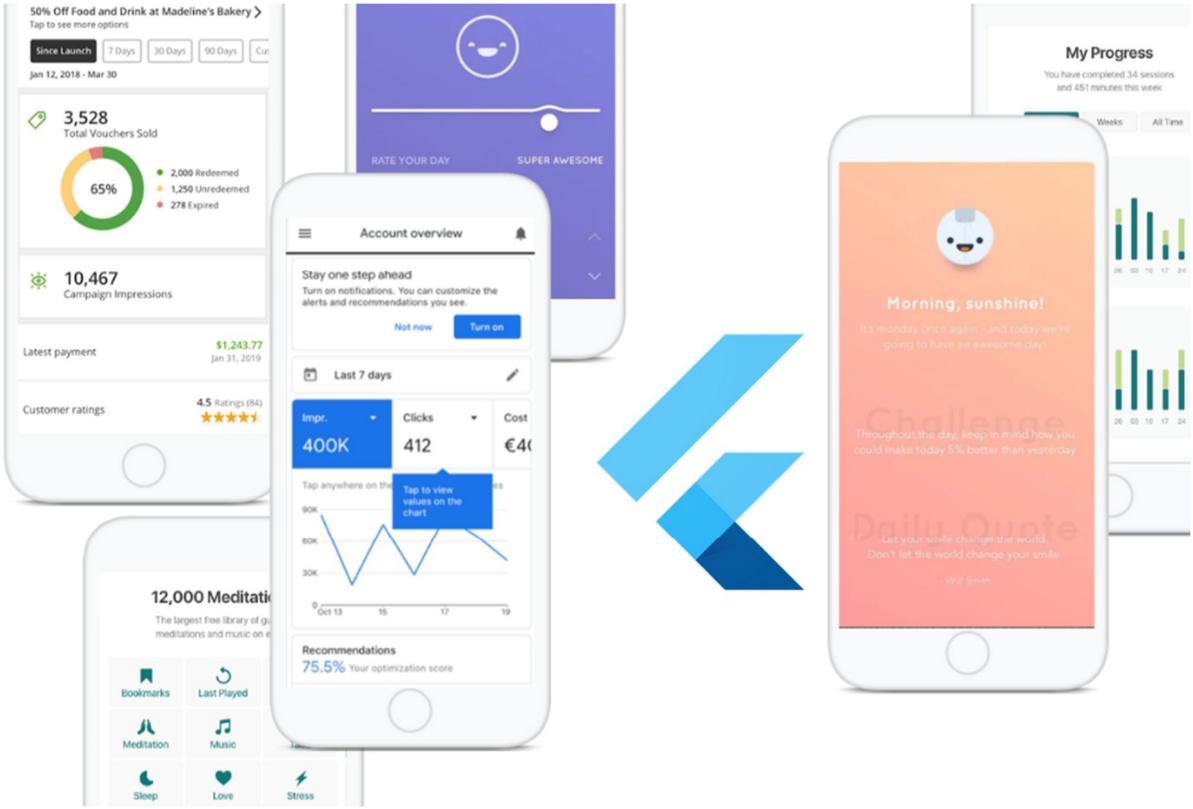
GET vs POST Αιτήσεις:

- **GET**: Χρησιμοποιείται για να **ζητήσουμε** δεδομένα από έναν διακομιστή. Συνήθως δεν περιέχει σώμα (body) και είναι ασφαλής μέθοδος (δεν αλλάζει τα δεδομένα στον διακομιστή).
- **POST**: Χρησιμοποιείται για να **στείλουμε** δεδομένα στον διακομιστή. Συνήθως χρησιμοποιείται για να δημιουργήσουμε ή να ενημερώσουμε πόρους στον διακομιστή.

Αυτό το παράδειγμα δείχνει πώς μπορούμε να χρησιμοποιήσουμε την Python για να αλληλεπιδράσουμε με ένα API, τόσο για την λήψη δεδομένων όσο και για την αποστολή δεδομένων.

3.2 Flutter

Το Flutter είναι ένα πλαίσιο λογισμικού ανοιχτού κώδικα που δημιουργήθηκε από την Google και χρησιμοποιείται για την ανάπτυξη εφαρμογών για κινητές συσκευές (Android και iOS), επιτραπέζιους υπολογιστές και τον ιστό από μια ενιαία βάση κώδικα. Η πρώτη σταθερή έκδοση του Flutter κυκλοφόρησε το Δεκέμβριο του 2018. Αρχικά αναπτύχθηκε για να αντιμετωπίσει την ανάγκη για ένα εργαλείο που θα επέτρεπε στους προγραμματιστές να δημιουργούν εφαρμογές για πολλές πλατφόρμες ταυτόχρονα, εξοικονομώντας χρόνο και πόρους. [7,8]



Εικόνα 3.2: Flutter

[https://appstronauts.co/wp-content/uploads/2020/03/flutter-app-examples.png]

Το 2021, το Flutter απέκτησε τη δυνατότητα ανάπτυξης όχι μόνο για Android και iOS, αλλά και για άλλες πλατφόρμες, όπως Windows, macOS, Linux, και εφαρμογές web, ενισχύοντας τη θέση του ως ένα πολυπλατφορμικό εργαλείο.

Χαρακτηριστικά του Flutter

Το Flutter διαθέτει μια σειρά από χαρακτηριστικά που το κάνουν να ξεχωρίζει:

- **Cross-platform ανάπτυξη**: Η δυνατότητα να γράφεις κώδικα μια φορά και να τον τρέχεις σε πολλές πλατφόρμες, όπως Android, iOS, web και desktop.
- **Dart**: Το Flutter χρησιμοποιεί τη γλώσσα προγραμματισμού Dart, η οποία είναι εύκολη στη μάθηση και ενσωματώνει στοιχεία από άλλες γλώσσες όπως η Java και η JavaScript.
- **Hot Reload**: Η λειτουργία αυτή επιτρέπει στους προγραμματιστές να βλέπουν άμεσα τις αλλαγές στον κώδικα χωρίς να χρειάζεται να κάνουν επανεκκίνηση της εφαρμογής. Είναι ιδιαίτερα χρήσιμη για γρήγορη ανάπτυξη και δοκιμές.

- **Widgets**: Όλα στο Flutter είναι widgets, και αυτό δίνει μεγάλη ευελιξία στην ανάπτυξη του UI (User Interface). Τα widgets του Flutter είναι εκτενώς προσαρμόσιμα και σχεδιασμένα για να δημιουργούν υψηλής απόδοσης και ομαλά περιβάλλοντα χρήστη.
- **Πλήρως προσαρμόσιμο UI**: Το Flutter παρέχει ένα ευρύ φάσμα εργαλείων για τη δημιουργία εντυπωσιακών, προσαρμόσιμων UI, που μπορούν να τρέχουν σε διαφορετικές πλατφόρμες με υψηλή απόδοση.
- **Εκτενής βιβλιοθήκη και κοινότητα**: Υπάρχει μια μεγάλη συλλογή από έτοιμα πακέτα (packages) και βιβλιοθήκες, που διευκολύνουν την ανάπτυξη εφαρμογών.

Χρήσεις και Εφαρμογές του Flutter

Το Flutter είναι ιδανικό για την ανάπτυξη εφαρμογών σε διάφορους τομείς:

- **Ανάπτυξη εφαρμογών κινητών συσκευών**: Το Flutter χρησιμοποιείται ευρέως για την ανάπτυξη εφαρμογών για Android και iOS, επιτρέποντας τη δημιουργία εφαρμογών με ενιαίο κώδικα.
- **Εφαρμογές για τον Ιστό**: Με το Flutter, μπορούν να δημιουργηθούν εφαρμογές για τον ιστό που είναι εξίσου αποδοτικές και μοντέρνες.
- **Εφαρμογές για επιτραπέζιους υπολογιστές**: Το Flutter υποστηρίζει την ανάπτυξη εφαρμογών για Windows, macOS και Linux.
- **Πειραματικές εφαρμογές και πρωτότυπα**: Η δυνατότητα ταχείας ανάπτυξης με το hot reload καθιστά το Flutter ιδανικό για πειραματισμούς και δημιουργία πρωτοτύπων.

Πλεονεκτήματα του Flutter

- **Ενιαία βάση κώδικα για πολλές πλατφόρμες**: Μειώνει σημαντικά το χρόνο ανάπτυξης και τις δαπάνες, καθώς οι προγραμματιστές δεν χρειάζεται να γράφουν ξεχωριστό κώδικα για κάθε πλατφόρμα.
- **Υψηλή απόδοση**: Το Flutter παρέχει σχεδόν εγγενή απόδοση, καθώς μεταγλωττίζει τον κώδικα σε εγγενές (native) κώδικα αντί να χρησιμοποιεί έναν ερμηνευτή.
- **Hot Reload**: Βοηθάει τους προγραμματιστές να βλέπουν γρήγορα τα αποτελέσματα των αλλαγών τους, βελτιώνοντας την αποδοτικότητα.
- **Ευέλικτος σχεδιασμός**: Το Flutter παρέχει μεγάλη ευελιξία στο σχεδιασμό και την προσαρμογή των εφαρμογών, δίνοντας στον προγραμματιστή τη δυνατότητα να δημιουργήσει πλούσια και μοντέρνα UI.

- **Ισχυρή κοινότητα**: Η κοινότητα του Flutter είναι ενεργή και προσφέρει συνεχώς νέες βιβλιοθήκες και εργαλεία που διευκολύνουν την ανάπτυξη εφαρμογών.

Μειονεκτήματα του Flutter

- **Μεγάλο μέγεθος εφαρμογών**: Οι εφαρμογές που αναπτύσσονται με το Flutter τείνουν να έχουν μεγαλύτερο μέγεθος σε σύγκριση με τις εφαρμογές που αναπτύσσονται με εγγενή εργαλεία (native).
- **Περιορισμένη βιβλιοθήκη για πλατφόρμες**: Αν και το Flutter έχει αρκετές βιβλιοθήκες, ορισμένες εξειδικευμένες λειτουργίες ή συσκευές ενδέχεται να μην υποστηρίζονται πλήρως και να απαιτείται εγγενής κώδικας για αυτές τις περιπτώσεις.
- **Dart**: Παρόλο που η Dart είναι μια εύκολη γλώσσα προγραμματισμού, δεν είναι τόσο δημοφιλής όσο άλλες γλώσσες, και αυτό μπορεί να αποτελέσει εμπόδιο για προγραμματιστές που δεν την γνωρίζουν.
- **Υψηλή κατανάλωση πόρων**: Οι εφαρμογές Flutter μπορεί να έχουν αυξημένες απαιτήσεις πόρων, ιδιαίτερα σε παλαιότερες συσκευές.

Το Flutter έχει καθιερωθεί ως ένα από τα κορυφαία εργαλεία για ανάπτυξη πολυπλατφορμικών εφαρμογών. Παρόλο που έχει κάποια μειονεκτήματα, τα πλεονεκτήματά του, όπως η ταχεία ανάπτυξη, η δυνατότητα δημιουργίας εφαρμογών για πολλές πλατφόρμες και η πλούσια εμπειρία χρήστη, το καθιστούν ιδανικό για πολλές επιχειρήσεις και προγραμματιστές.

Ακολουθεί ένα απλό παράδειγμα εφαρμογής Flutter που εμφανίζει ένα κουμπί και, όταν πατηθεί, αυξάνει έναν αριθμητικό μετρητή.

```
import 'package:flutter/material.dart';


void main() {
  runApp(MyApp());
}


class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
```

```
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Απλό Παράδειγμα Flutter'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
```

```
        children: <Widget>[
          Text(
            'Πατήσατε το κουμπί τόσες φορές:',
          ),
          Text(
            '$_counter',
            style: Theme.of(context).textTheme.headline4,
          ),
        ],
      ),
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: _incrementCounter,
      tooltip: 'Increment',
      child: Icon(Icons.add),
    ),
  );
 }
}
```

main(): Η συνάρτηση main() είναι το σημείο εκκίνησης της εφαρμογής Flutter. Χρησιμοποιεί τη μέθοδο runApp() για να εκκινήσει την εφαρμογή, περνώντας την κλάση MyApp ως παράμετρο.

MyApp: Αυτή η κλάση είναι η κύρια κλάση της εφαρμογής και επεκτείνει το StatelessWidget, που σημαίνει ότι δεν αλλάζει κατάσταση κατά την εκτέλεση της εφαρμογής. Επιστρέφει ένα MaterialApp, το οποίο είναι η ρίζα της εφαρμογής και ορίζει θέματα, τίτλο και την αρχική σελίδα (MyHomePage).

MyHomePage: Αυτή η κλάση είναι ένα StatefulWidget, το οποίο σημαίνει ότι έχει μια εσωτερική κατάσταση που μπορεί να αλλάξει κατά τη διάρκεια της εκτέλεσης της εφαρμογής. Η αλλαγή κατάστασης επιτρέπει στον μετρητή να ενημερώνεται όταν ο χρήστης πατάει το κουμπί.

_MyHomePageState: Αυτή η κλάση είναι η κατάσταση της MyHomePage. Εδώ βρίσκεται η βασική λογική της εφαρμογής. Ο μετρητής (_counter) αρχικοποιείται με 0 και αυξάνεται κάθε φορά που ο

χρήστης πατά το κουμπί μέσω της μεθόδου _incrementCounter(). Η μέθοδος setState() ενημερώνει την UI κάθε φορά που ο μετρητής αλλάζει.

Scaffold: Είναι ένα widget που παρέχει τη βασική δομή της οθόνης. Περιλαμβάνει το AppBar (μπάρα τίτλου), το body (κύριο περιεχόμενο) και ένα floatingActionButton (πλωτό κουμπί δράσης).

floatingActionButton: Αυτό είναι το κουμπί που εμφανίζεται στην κάτω δεξιά γωνία της οθόνης. Όταν ο χρήστης πατά το κουμπί, καλείται η συνάρτηση _incrementCounter() και αυξάνει τον αριθμητικό μετρητή.

Text: Τα widget Text εμφανίζουν μηνύματα στην οθόνη. Το πρώτο εμφανίζει σταθερό κείμενο ("Πατήσατε το κουμπί τόσες φορές:") και το δεύτερο εμφανίζει τον τρέχοντα αριθμό του μετρητή, ο οποίος ενημερώνεται δυναμικά όταν αλλάζει.

3.3 Βάση δεδομένων MySQL

Η MySQL είναι ένα από τα πιο δημοφιλή συστήματα διαχείρισης βάσεων δεδομένων (DBMS) ανοιχτού κώδικα, το οποίο χρησιμοποιείται για τη διαχείριση, αποθήκευση και ανάκτηση δεδομένων. Αναπτύχθηκε αρχικά από τον Michael Widenius το 1995 και αποτελεί ένα από τα βασικά συστήματα βάσεων δεδομένων που χρησιμοποιούνται παγκοσμίως, κυρίως λόγω της ταχύτητας, της αξιοπιστίας και της ευελιξίας της. [8,9]

Η MySQL είναι κυρίως γνωστή για την χρήση της σε εφαρμογές ιστού και αποτελεί τη βάση για πολλές διαδικτυακές πλατφόρμες. Αποτελεί βασικό στοιχείο του λεγόμενου LAMP stack (Linux, Apache, MySQL, PHP/Python/Perl), το οποίο είναι ευρέως χρησιμοποιούμενο για την ανάπτυξη ιστοσελίδων και εφαρμογών διαδικτύου.

Χαρακτηριστικά της MySQL

Ανοιχτός κώδικας: Η MySQL είναι ανοιχτού κώδικα και διατίθεται δωρεάν. Παρόλα αυτά, υπάρχει και η έκδοση Enterprise που παρέχει πρόσθετα χαρακτηριστικά και υποστήριξη για επιχειρήσεις.

Υποστήριξη πολλών χρηστών: Η MySQL υποστηρίζει πολλούς ταυτόχρονους χρήστες, καθιστώντας την κατάλληλη για μεγάλα συστήματα με πολλούς χρήστες ή εφαρμογές που διαχειρίζονται πολλά αιτήματα ταυτόχρονα.

Αρχιτεκτονική Client-Server: Η MySQL λειτουργεί σε αρχιτεκτονική client-server, όπου οι πελάτες (clients) συνδέονται με τον διακομιστή MySQL και στέλνουν αιτήματα για δεδομένα.

Υποστήριξη γλωσσών SQL: Χρησιμοποιεί την SQL (Structured Query Language) για την επικοινωνία και διαχείριση των δεδομένων. Η SQL είναι η πιο διαδεδομένη γλώσσα για βάσεις δεδομένων και επιτρέπει τη δημιουργία, διαχείριση και ανάκτηση δεδομένων με ευκολία.

Αποδοτικότητα και ταχύτητα: Η MySQL είναι γνωστή για την ταχύτητά της, ειδικά σε εφαρμογές που απαιτούν μεγάλη απόδοση και χειρισμό μεγάλου όγκου δεδομένων.

Ασφάλεια: Παρέχει δυνατότητες ελέγχου πρόσβασης και υποστήριξη κρυπτογράφησης δεδομένων για την ασφάλεια των πληροφοριών.

Πολυπλατφορμική υποστήριξη: Τρέχει σε διάφορα λειτουργικά συστήματα, όπως Linux, Windows, macOS, και Unix.

Χρήσεις και Εφαρμογές της MySQL

Ιστοσελίδες και εφαρμογές διαδικτύου: Πολλές δημοφιλείς πλατφόρμες, όπως το WordPress, το Joomla και το Drupal, χρησιμοποιούν MySQL ως την κύρια βάση δεδομένων τους για την αποθήκευση περιεχομένου, χρηστών, και ρυθμίσεων.

Ηλεκτρονικό εμπόριο: Πλατφόρμες ηλεκτρονικού εμπορίου, όπως το Magento και το WooCommerce, χρησιμοποιούν MySQL για τη διαχείριση προϊόντων, πελατών και παραγγελιών.

Αποθήκευση δεδομένων: Η MySQL χρησιμοποιείται συχνά για τη διαχείριση μεγάλων όγκων δεδομένων σε συστήματα ανάλυσης δεδομένων και επιχειρηματικής ευφυΐας.

Εφαρμογές cloud: Η MySQL χρησιμοποιείται ευρέως από υπηρεσίες cloud, όπως το Amazon RDS και το Google Cloud SQL, παρέχοντας αξιόπιστες και κλιμακούμενες λύσεις βάσεων δεδομένων.

Πλεονεκτήματα της MySQL

Ταχύτητα και Απόδοση: Η MySQL είναι γρήγορη και μπορεί να διαχειριστεί μεγάλο όγκο δεδομένων και πολλά αιτήματα ταυτόχρονα, κάνοντάς την κατάλληλη για ισχυρές εφαρμογές διαδικτύου.

Κλιμακωσιμότητα: Μπορεί να επεκταθεί για να χειριστεί βάσεις δεδομένων με μεγάλο όγκο πληροφοριών, ενώ ταυτόχρονα προσφέρει αποδοτική απόδοση για μικρότερες εφαρμογές.

Υποστήριξη από την κοινότητα: Η MySQL είναι ανοιχτού κώδικα, και διαθέτει μια μεγάλη και ενεργή κοινότητα προγραμματιστών που παρέχουν συνεχή βελτίωση και υποστήριξη.

Συμβατότητα: Είναι συμβατή με πολλές γλώσσες προγραμματισμού, όπως PHP, Python, Java, C++, καθιστώντας την ιδανική για πολλές διαφορετικές εφαρμογές.

Ασφάλεια: Η MySQL παρέχει ισχυρή υποστήριξη για τη διαχείριση χρηστών και την ασφάλεια των δεδομένων.

Μειονεκτήματα της MySQL

Περιορισμοί σε πολύπλοκες συναλλαγές: Η MySQL μπορεί να παρουσιάσει περιορισμούς σε πολύπλοκες συναλλαγές σε σύγκριση με άλλες βάσεις δεδομένων όπως η PostgreSQL.

Υποστήριξη για μεγάλα δεδομένα: Παρόλο που είναι αποτελεσματική για μεγάλες εφαρμογές, σε περιπτώσεις τεράστιων βάσεων δεδομένων (π.χ., petabytes), μπορεί να υπάρξουν θέματα απόδοσης και διαχείρισης.

Περιορισμένη υποστήριξη για εξειδικευμένες λειτουργίες: Αν και η MySQL είναι εξαιρετική για γενικές βάσεις δεδομένων, σε εξειδικευμένες περιπτώσεις μπορεί να απαιτηθεί χρήση πρόσθετων εργαλείων ή προσαρμογών.

Σύνδεση της Python με MySQL

Η Python και η MySQL συνεργάζονται εύκολα μέσω της βιβλιοθήκης MySQL Connector ή άλλων βιβλιοθηκών, όπως η SQLAlchemy. Αυτό επιτρέπει στους προγραμματιστές να αλληλεπιδρούν με βάσεις δεδομένων MySQL, εκτελώντας ερωτήματα και διαχειριζόμενοι δεδομένα απευθείας από εφαρμογές Python.

Παράδειγμα σύνδεσης Python με MySQL:

```
import mysql.connector


# Σύνδεση με τη βάση δεδομένων
mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="yourdatabase"
)
```

```
# Δημιουργία cursor για εκτέλεση SQL ερωτημάτων
mycursor = mydb.cursor()

# Εκτέλεση SQL query
mycursor.execute("SELECT * FROM yourtable")

# Ανάκτηση δεδομένων
result = mycursor.fetchall()

# Εμφάνιση των αποτελεσμάτων
for row in result:
    print(row)

# Κλείσιμο της σύνδεσης
mydb.close()
```

Η MySQL είναι ένα εξαιρετικά ισχυρό, γρήγορο και ευέλικτο σύστημα βάσεων δεδομένων που χρησιμοποιείται σε μεγάλο εύρος εφαρμογών, από μικρές ιστοσελίδες μέχρι μεγάλες επιχειρήσεις και πλατφόρμες. Παρά τους περιορισμούς της σε ορισμένες περιπτώσεις, παραμένει μία από τις κυρίαρχες λύσεις για διαχείριση δεδομένων παγκοσμίως.

3.4 Esp32

Το ESP32 είναι ένα ισχυρό και ευέλικτο μικροελεγκτή και Wi-Fi/Bluetooth chip που αναπτύχθηκε από την Espressif Systems. Κυκλοφόρησε το 2016 και σχεδιάστηκε για να αντικαταστήσει το ESP8266, προσφέροντας περισσότερη ισχύ, μεγαλύτερη ευελιξία και πρόσθετες δυνατότητες, όπως Bluetooth. Το ESP32 είναι ιδανικό για διάφορες εφαρμογές στο Internet of Things (IoT), λόγω της υποστήριξής του για Wi-Fi και Bluetooth, καθώς και του χαμηλού του κόστους.

Το ESP32 προσφέρει πολυπλοκότερες δυνατότητες από τον προκάτοχό του, κάτι που το καθιστά εξαιρετική επιλογή για έργα που απαιτούν περισσότερη επεξεργαστική ισχύ ή πρόσθετα χαρακτηριστικά συνδεσιμότητας.

Χαρακτηριστικά του ESP32

Wi-Fi και Bluetooth: Το ESP32 υποστηρίζει Wi-Fi (802.11 b/g/n) και Bluetooth (4.2 και BLE - Bluetooth Low Energy), επιτρέποντας τη δημιουργία ασύρματων συνδέσεων με εύκολο τρόπο.

Διπλός Πυρήνας (Dual-core): Διαθέτει έναν διπύρηνο επεξεργαστή Tensilica Xtensa LX6 που του επιτρέπει να εκτελεί πιο απαιτητικές εργασίες ταυτόχρονα, αυξάνοντας την απόδοση σε πιο πολύπλοκες εφαρμογές.

Μεγάλη ποσότητα μνήμης: Το ESP32 διαθέτει 520KB SRAM και μπορεί να υποστηρίξει εξωτερική μνήμη έως και 4MB, κάτι που το καθιστά ιδανικό για εφαρμογές που απαιτούν περισσότερη επεξεργαστική ισχύ και αποθήκευση δεδομένων.

Πολλαπλά I/O Pins: Διαθέτει πολλαπλές θύρες εισόδου/εξόδου (GPIO), που μπορούν να χρησιμοποιηθούν για τη σύνδεση αισθητήρων, ενεργοποιητών και άλλων συσκευών. Υποστηρίζει πρωτόκολλα όπως SPI, I2C, UART, PWM και DAC/ADC.

Λειτουργία χαμηλής κατανάλωσης ενέργειας: Το ESP32 έχει σχεδιαστεί για εφαρμογές χαμηλής ενέργειας, καθιστώντας το κατάλληλο για έργα που τροφοδοτούνται από μπαταρία και πρέπει να λειτουργούν για μεγάλες χρονικές περιόδους.

Ασφάλεια: Υποστηρίζει κρυπτογράφηση δεδομένων με δυνατότητες όπως AES, RSA, και SHA, καθιστώντας το ESP32 ασφαλές για εφαρμογές IoT που απαιτούν προστασία δεδομένων.

Χρήσεις και Εφαρμογές του ESP32

Έξυπνα Συστήματα Σπιτιού (Smart Home): Το ESP32 μπορεί να χρησιμοποιηθεί για την ανάπτυξη αυτοματοποιημένων συστημάτων φωτισμού, κλιματισμού και ασφαλείας, επιτρέποντας την απομακρυσμένη διαχείριση μέσω εφαρμογών ή φωνητικών εντολών.

Ανίχνευση Περιβάλλοντος και Καταγραφή Δεδομένων: Μπορεί να συνδεθεί με αισθητήρες για τη μέτρηση περιβαλλοντικών δεδομένων, όπως θερμοκρασία, υγρασία, πίεση ή ακόμα και ποιότητα αέρα, και να αποστείλει τα δεδομένα μέσω Wi-Fi ή Bluetooth.

Wearable συσκευές και Υγεία: Χάρη στη δυνατότητα χαμηλής κατανάλωσης ενέργειας και υποστήριξη BLE, το ESP32 είναι κατάλληλο για φορετές συσκευές που παρακολουθούν τη φυσική δραστηριότητα και την υγεία.

Βιομηχανικό IoT (Industrial IoT): Σε βιομηχανικά περιβάλλοντα, το ESP32 μπορεί να χρησιμοποιηθεί για τη σύνδεση μηχανών και αισθητήρων με το δίκτυο, επιτρέποντας την παρακολούθηση και τον έλεγχο σε πραγματικό χρόνο.

Δίκτυα αισθητήρων: Το ESP32 μπορεί να αποτελέσει κόμβο σε δίκτυα αισθητήρων που παρακολουθούν περιβαλλοντικές ή βιομηχανικές παραμέτρους και στέλνουν δεδομένα για ανάλυση.

Πλεονεκτήματα του ESP32

Ενσωματωμένες δυνατότητες ασύρματης συνδεσιμότητας: Η υποστήριξη για Wi-Fi και Bluetooth καθιστά το ESP32 ιδανικό για εφαρμογές IoT, καθώς μπορεί να συνδεθεί εύκολα με άλλες συσκευές ή το διαδίκτυο.

Υψηλή επεξεργαστική ισχύς: Η ύπαρξη διπύρηνου επεξεργαστή και αρκετής μνήμης το κάνει κατάλληλο για εφαρμογές που απαιτούν μεγάλη υπολογιστική δύναμη.

Πολλές θύρες I/O: Η πληθώρα θυρών εισόδου/εξόδου επιτρέπει τη σύνδεση ποικίλων συσκευών και αισθητήρων, καθιστώντας το ESP32 ευέλικτο για πολλαπλές εφαρμογές.

Χαμηλή κατανάλωση ενέργειας: Οι λειτουργίες εξοικονόμησης ενέργειας το καθιστούν εξαιρετική επιλογή για εφαρμογές που λειτουργούν με μπαταρία και απαιτούν συνεχή λειτουργία για μεγάλα χρονικά διαστήματα.

Ασφάλεια: Η υποστήριξη για κρυπτογράφηση και πρωτόκολλα ασφαλείας παρέχει προστασία δεδομένων, κάτι κρίσιμο για εφαρμογές IoT.

Μειονεκτήματα του ESP32

Περίπλοκη χρήση για αρχάριους: Αν και το ESP32 προσφέρει πολλές δυνατότητες, μπορεί να είναι περίπλοκο για κάποιον που μόλις ξεκινάει να εργάζεται με μικροελεγκτές. Η εγκατάσταση και παραμετροποίηση ενδέχεται να απαιτεί κάποια εξοικείωση με πλατφόρμες ανάπτυξης, όπως το Arduino IDE ή το Espressif IDF.

Συμβατότητα με ορισμένες βιβλιοθήκες: Αν και το ESP32 υποστηρίζει πολλές βιβλιοθήκες, ενδέχεται να υπάρχουν περιορισμοί με βιβλιοθήκες που έχουν αναπτυχθεί για άλλους μικροελεγκτές, όπως το Arduino.

Υψηλή κατανάλωση σε Wi-Fi mode: Παρόλο που έχει λειτουργίες χαμηλής κατανάλωσης ενέργειας, όταν λειτουργεί σε Wi-Fi mode, η κατανάλωση ενέργειας αυξάνεται σημαντικά.

3.5 Αισθητήρες και ενεργοποιητές

Το σύστημα που περιγράφεται παρακάτω αποτελείται από δύο κύριους αισθητήρες: μια μαγνητική παγίδα (magnetic door/window sensor) και έναν PIR αισθητήρα κίνησης (Passive Infrared sensor). Και οι δύο αισθητήρες είναι συνδεδεμένοι στο ESP32, το οποίο όταν ανιχνεύσει ενεργοποίηση (είτε από τη μαγνητική παγίδα είτε από τον PIR αισθητήρα), στέλνει μια ειδοποίηση στον Python server μέσω Wi-Fi. Το σύστημα έχει επίσης τη δυνατότητα να δέχεται εντολές από τον Python server για την ενεργοποίηση ενός ρελέ, που μπορεί να ελέγχει διάφορες συσκευές, όπως φώτα ή συναγερμό.

Μαγνητική Παγίδα

Η μαγνητική παγίδα αποτελείται από δύο μέρη: ένα μαγνήτη και έναν διακόπτη. Όταν ο μαγνήτης βρίσκεται κοντά στον διακόπτη (συνήθως σε μια πόρτα ή παράθυρο), ο διακόπτης παραμένει κλειστός. Όταν ο μαγνήτης απομακρυνθεί, ο διακόπτης ανοίγει, σηματοδοτώντας ότι η πόρτα ή το παράθυρο έχει ανοιχτεί.

Η παγίδα συνδέεται σε μια ψηφιακή είσοδο (GPIO) του ESP32, η οποία παρακολουθεί αν η παγίδα είναι κλειστή ή ανοιχτή. Όταν ανιχνευθεί άνοιγμα της παγίδας (δηλαδή, διακοπή του κυκλώματος), το ESP32 στέλνει μια ειδοποίηση στον Python server.

Αισθητήρας PIR

Ο PIR (Passive Infrared) αισθητήρας ανιχνεύει την κίνηση ανθρώπων ή ζώων με βάση την αλλαγή θερμοκρασίας στο περιβάλλον του. Όταν εντοπίσει κίνηση, στέλνει ένα ψηφιακό σήμα στο ESP32 μέσω μιας ψηφιακής εισόδου (GPIO).

Ο αισθητήρας συνδέεται σε μια άλλη είσοδο του ESP32, η οποία παρακολουθεί την κατάσταση του PIR. Όταν ανιχνευθεί κίνηση, το ESP32 στέλνει επίσης ειδοποίηση στον Python server, όπως γίνεται και με τη μαγνητική παγίδα.

Αποστολή Δεδομένων στον Python Server

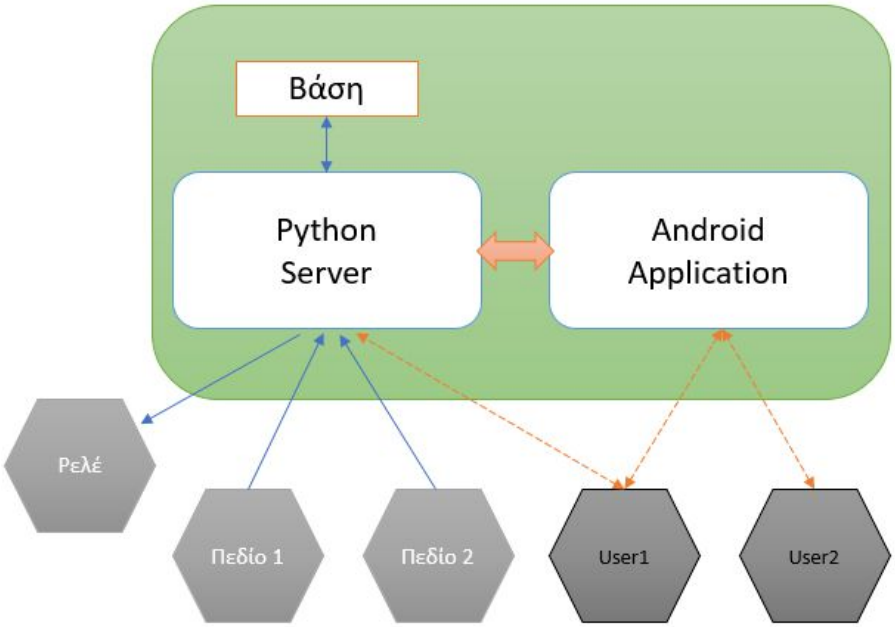
Το ESP32 επικοινωνεί με τον Python server μέσω Wi-Fi. Όταν ενεργοποιηθεί κάποιος από τους δύο αισθητήρες (μαγνητική παγίδα ή PIR), το ESP32 στέλνει ένα HTTP POST αίτημα στον server, περιλαμβάνοντας το ID του αισθητήρα που ενεργοποιήθηκε και την κατάστασή του. Ο Python server αποθηκεύει το συμβάν στη βάση δεδομένων και μπορεί να ειδοποιήσει τον χρήστη μέσω της εφαρμογής.

Έλεγχος Ρελέ από τον Python Server

Ο Python server έχει τη δυνατότητα να στείλει εντολές πίσω στο ESP32 για την ενεργοποίηση ή απενεργοποίηση του ρελέ. Το ρελέ μπορεί να χρησιμοποιηθεί για τον έλεγχο εξωτερικών συσκευών, όπως ένα σύστημα συναγερμού ή ένα φως. Το ESP32 διαβάζει αυτές τις εντολές και ελέγχει τη λειτουργία του ρελέ μέσω μιας εξόδου GPIO.

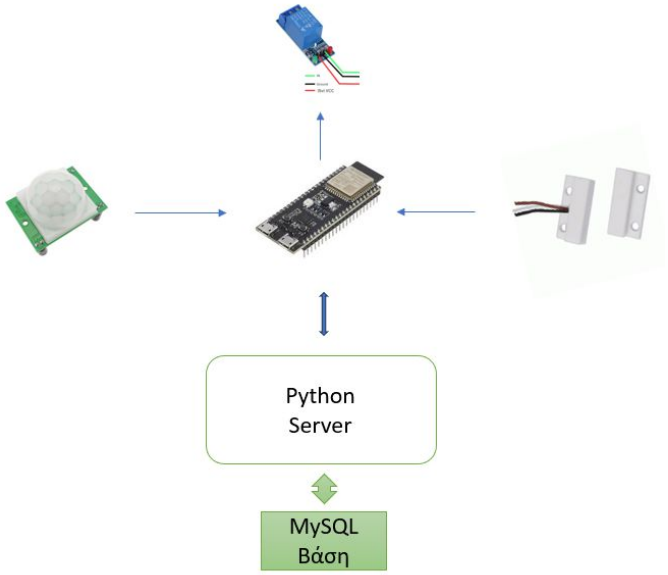
Κεφάλαιο 4ο: Το σύστημα ειδοποιήσεων κατάστασης εσωτερικού χώρου μέσω διαδικτύου

4.1 Εισαγωγή στο σύστημα



Εικόνα 4.1: Σύστημα ειδοποιήσεων κατάστασης εσωτερικού χώρου μέσω διαδικτύου

Η Εικόνα 4.1 απεικονίζει τη γενική αρχιτεκτονική στις συστήματος που αποτελείται από έναν **Python Server**, αισθητήρες και ρελέ (Relay), και μία **Android εφαρμογή** που επικοινωνεί με τον server. Ο **Python Server** επικοινωνεί άμεσα με τα πεδία αισθητήρων (Πεδίο 1, Πεδίο 2) και ένα ρελέ. Οι αισθητήρες αυτοί στέλνουν δεδομένα στον server, ο οποίος τα επεξεργάζεται και τα αποθηκεύει στη βάση δεδομένων. Οι χρήστες (User1, User2) συνδέονται με τον server μέσω στις Android εφαρμογής, που στις επιτρέπει να λαμβάνουν ενημερώσεις για την κατάσταση των αισθητήρων. Ο server επικοινωνεί με τη βάση δεδομένων, όπου αποθηκεύονται όλα τα στοιχεία που αφορούν στις ειδοποιήσεις, στις καταστάσεις των αισθητήρων, και στις ενέργειες που έχουν εκτελεστεί μέσω των ρελέ. Η αρχιτεκτονική αυτή είναι σχεδιασμένη ώστε να δίνει στους χρήστες τη δυνατότητα να παρακολουθούν την κατάσταση του συστήματος σε πραγματικό χρόνο από την κινητή στις συσκευή. Οι χρήστες μπορούν στις να ενεργοποιήσουν ή να απενεργοποιήσουν συσκευές μέσω του ρελέ, δημιουργώντας ένα ολοκληρωμένο σύστημα ελέγχου και ειδοποίησης.



Εικόνα 4.2: Σύστημα επικοινωνίας του κόμβου με τους αισθητήρες και τους ενεργοποιητές και τον server

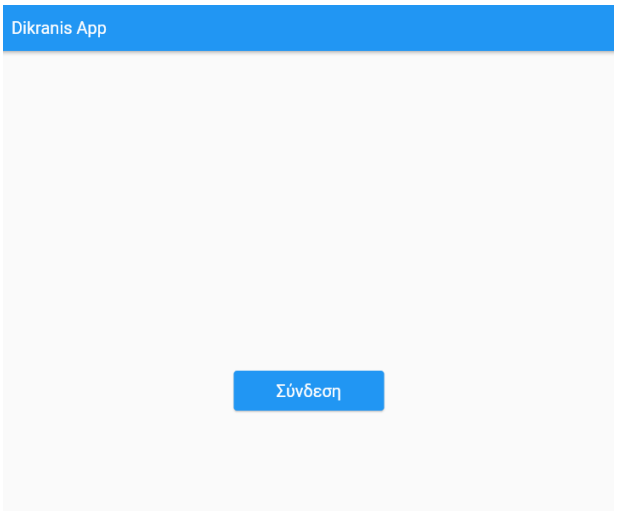
Η Εικόνα 4.2 παρουσιάζει την πιο τεχνική αρχιτεκτονική επικοινωνίας του συστήματος. Κεντρικό στοιχείο αποτελεί ο μικροελεγκτής (ESP32 ή Raspberry Pi), ο οποίος επικοινωνεί με διάφορους αισθητήρες και συσκευές, στις μαγνητικές επαφές (για ανίχνευση ανοίγματος ή κλεισίματος), ρελέ για ενεργοποίηση ή απενεργοποίηση συσκευών, και αισθητήρες κίνησης (PIR). Ο μικροελεγκτής στις συλλέγει δεδομένα από στις αισθητήρες και τα αποστέλλει στον **Python Server**, ο οποίος επεξεργάζεται στις πληροφορίες και ενημερώνει τη **MySQL βάση δεδομένων**. Η βάση δεδομένων κρατάει αρχείο των καταγεγραμμένων ενεργειών, στις συμβάντα κίνησης, ενεργοποιήσεις αισθητήρων και ρελέ. Το διάγραμμα αυτό υποδεικνύει την ροή των δεδομένων από στις αισθητήρες στις τον Python Server και την αντίστροφη ροή εντολών από τον server στις το ρελέ και στις συσκευές. Η αρχιτεκτονική είναι ιδανική για παρακολούθηση και έλεγχο μέσω διαδικτύου, προσφέροντας απομακρυσμένη πρόσβαση στις συνδεδεμένες συσκευές και αισθητήρες σε πραγματικό χρόνο.

```
127.0.0.1 - - [09/Sep/2024 12:40:01] "GET /alarms/1 HTTP/1.1" 200 -
127.0.0.1 - - [09/Sep/2024 12:49:25] "GET /alarms/1 HTTP/1.1" 200 -
127.0.0.1 - - [09/Sep/2024 12:49:38] "GET /alarm?komvosid=1&field1=1&field2=0&field3=0&field4=0&field5=0&field6=0 HTTP/1.1" 201 -
127.0.0.1 - - [09/Sep/2024 12:59:07] "GET /alarms/1 HTTP/1.1" 200 -
127.0.0.1 - - [09/Sep/2024 13:00:48] "GET /alarms/1 HTTP/1.1" 200 -
```

Εικόνα 4.3: Ο server που εξυπηρετεί την εφαρμογή στο κινητό και την επικοινωνία με τη βάση

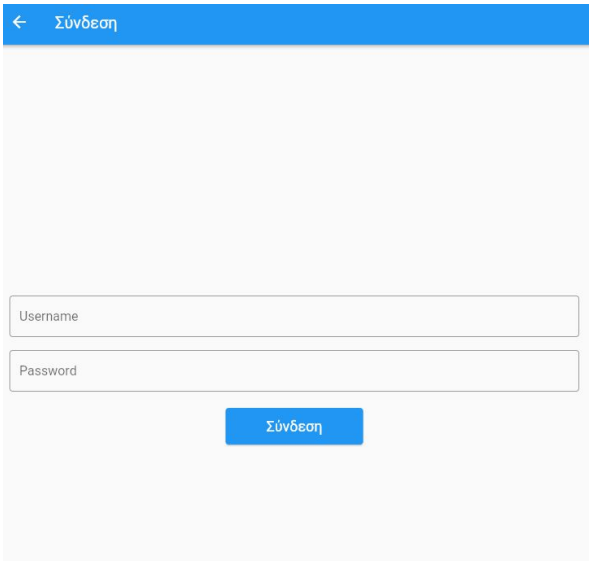
Η Εικόνα 4.3 παρουσιάζει τη λειτουργική αρχιτεκτονική του συστήματος επικοινωνίας ανάμεσα στον **Python Server**, τη **βάση δεδομένων MySQL**, και την **εφαρμογή κινητού**. Ο server είναι ο ενδιάμεσος κρίκος που δέχεται και επεξεργάζεται δεδομένα από τους αισθητήρες και τους κόμβους, όπως μαγνητικές παγίδες και ρελέ, και στη συνέχεια αποθηκεύει τα δεδομένα στη βάση. Η εφαρμογή Android επικοινωνεί με τον server μέσω αιτημάτων, συλλέγοντας δεδομένα που αφορούν την

κατάσταση των αισθητήρων και τις ειδοποιήσεις. Ο χρήστης της εφαρμογής έχει τη δυνατότητα να ελέγχει τη λειτουργία των αισθητήρων και να παρακολουθεί ειδοποιήσεις που σχετίζονται με την κατάσταση του συστήματος. Η ροή δεδομένων είναι αμφίδρομη, καθώς οι εντολές από την εφαρμογή κινητού μπορούν να ενεργοποιούν συσκευές όπως ρελέ, ενώ τα δεδομένα από τους αισθητήρες μεταφέρονται στη βάση και στη συνέχεια εμφανίζονται στον χρήστη μέσω της εφαρμογής. Η επικοινωνία αυτή καθιστά το σύστημα ιδανικό για απομακρυσμένη παρακολούθηση και έλεγχο, με έμφαση στη διαχείριση συμβάντων και την άμεση ανταπόκριση σε ειδοποιήσεις.



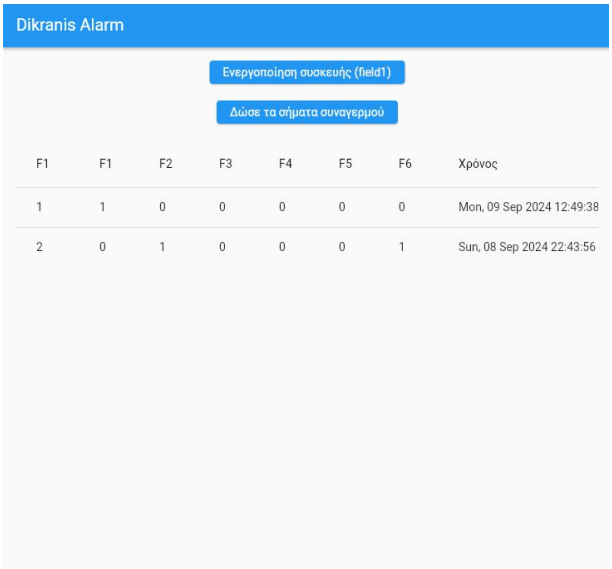
Εικόνα 4.4: Η αρχική οθόνη της εφαρμογής στο κινητό

Η Εικόνα 4.4 απεικονίζει την **αρχική οθόνη της εφαρμογής κινητού** που δημιουργήθηκε με το Flutter. Στην αρχική οθόνη εμφανίζεται ο τίτλος της εφαρμογής "Dikranis Alarm", μαζί με ένα κουμπί **"Σύνδεση"** που καλεί τον χρήστη να προχωρήσει στη διαδικασία εισόδου στο σύστημα. Η σχεδίαση είναι απλή και φιλική προς τον χρήστη, με την κεντρική ενέργεια να είναι ξεκάθαρη και εύκολη στην πρόσβαση. Η χρήση χρωμάτων, όπως το μπλε που εμφανίζεται στην επικεφαλίδα, συμβάλλει στη δημιουργία μιας ευχάριστης και επαγγελματικής αίσθησης, ενώ το κουμπί "Σύνδεση" είναι διαμορφωμένο με τέτοιο τρόπο ώστε να ξεχωρίζει. Το περιβάλλον αυτό είναι το πρώτο βήμα για τον χρήστη πριν την εισαγωγή διαπιστευτηρίων, επιτρέποντάς του να εισέλθει στο σύστημα παρακολούθησης και ελέγχου. Η απλότητα της οθόνης εξασφαλίζει μια ομαλή εμπειρία χρήσης χωρίς περιττά στοιχεία, εστιάζοντας μόνο στις βασικές λειτουργίες που χρειάζεται ο χρήστης.



Εικόνα 4.5: Η οθόνη εισαγωγής στοιχείων στην εφαρμογή στο κινητό

Η Εικόνα 4.5 απεικονίζει την **οθόνη εισαγωγής στοιχείων** (username και password) στην εφαρμογή κινητού. Ο χρήστης καλείται να εισαγάγει το **όνομα χρήστη** (username) και τον **κωδικό πρόσβασης** (password) για να συνδεθεί στο σύστημα. Αυτή η φόρμα περιλαμβάνει δύο πεδία κειμένου, το ένα για το όνομα χρήστη και το άλλο για τον κωδικό πρόσβασης, μαζί με ένα κουμπί **"Σύνδεση"**. Αφού ο χρήστης συμπληρώσει τα στοιχεία του, μπορεί να πατήσει το κουμπί "Σύνδεση", το οποίο τον μεταφέρει στην κεντρική σελίδα της εφαρμογής, ανεξάρτητα από την ορθότητα των στοιχείων. Η σχεδίαση είναι λιτή και στοχευμένη, εστιάζοντας στην εισαγωγή των βασικών διαπιστευτηρίων για πρόσβαση στο σύστημα παρακολούθησης και ελέγχου. Ο τίτλος "Σύνδεση" και τα πεδία είναι σχεδιασμένα με ευκρίνεια, εξασφαλίζοντας μια απλή και λειτουργική διεπαφή. Αυτή η οθόνη παρέχει έναν εύκολο και γρήγορο τρόπο σύνδεσης για τους χρήστες, με έμφαση στη χρηστικότητα και την ασφάλεια.



Εικόνα 4.6: Η κεντρική οθόνη της εφαρμογής

Η Εικόνα 4.6 απεικονίζει την **κεντρική οθόνη** της εφαρμογής κινητού μετά τη σύνδεση του χρήστη. Η κεντρική οθόνη εμφανίζει μια σειρά από δεδομένα που αφορούν την κατάσταση των αισθητήρων (πεδία F1 έως F6), καθώς και τη χρονική στιγμή καταγραφής του κάθε alarm. Τα δεδομένα αυτά προέρχονται από τους αισθητήρες που είναι συνδεδεμένοι με τον Python Server, ο οποίος ενημερώνει την εφαρμογή σε πραγματικό χρόνο. Η οθόνη περιλαμβάνει επίσης δύο κουμπιά: ένα για την ενεργοποίηση της συσκευής που σχετίζεται με το πεδίο F1, και ένα δεύτερο που επιτρέπει την ανάκτηση των τελευταίων σημάτων συναγερμού από τον server. Η διάταξη των δεδομένων σε πίνακα προσφέρει ευκρινή παρουσίαση, επιτρέποντας στον χρήστη να παρακολουθεί την κατάσταση των αισθητήρων και τις ειδοποιήσεις με μία ματιά. Η χρήση απλών και καθαρών γραμμών στο σχεδιασμό της διεπαφής, σε συνδυασμό με τη λειτουργικότητα, καθιστά την εμπειρία χρήσης άμεση και πρακτική, διευκολύνοντας την παρακολούθηση του συστήματος από τον χρήστη.

4.1.1 Κώδικας esp32

Ο παρακάτω κώδικας περιγράφει τη βασική λειτουργία του ESP32 για να ανιχνεύει την κατάσταση της μαγνητικής παγίδας και του PIR αισθητήρα, να στέλνει τα δεδομένα στον Python server και να δέχεται εντολές για την ενεργοποίηση του ρελέ.

```
#include <WiFi.h>
#include <HTTPClient.h>

// Wi-Fi Credentials
const char* ssid = "YourSSID";
const char* password = "YourPassword";

// Server URL
const char* serverUrl = "http://dikranis/api/sensor";

// GPIO Pins
const int magneticPin = 14;  // Pin μαγνητικής παγίδας
const int pirPin = 27;       // Pin PIR
const int relayPin = 26;     // Pin ρελέ

// Μεταβλητές κατάστασης
bool magneticState = false;
bool pirState = false;
```

```
void setup() {
  // Αρχικοποίηση Serial για εντοπισμό σφαλμάτων
  Serial.begin(115200);

  // Σύνδεση στο Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Σύνδεση με WiFi...");
  }
  Serial.println("Σύνδεση επιτυχής!");

  // Ρύθμιση GPIO pins
  pinMode(magneticPin, INPUT_PULLUP);  // Χρησιμοποιούμε pull-up για την παγίδα
  pinMode(pirPin, INPUT);
  pinMode(relayPin, OUTPUT);
}

void loop() {
  // Έλεγχος κατάστασης της μαγνητικής παγίδας
  bool newMagneticState = digitalRead(magneticPin) == LOW;  // LOW = άνοιγμα παγίδας
  if (newMagneticState != magneticState) {
    magneticState = newMagneticState;
    sendToServer("magnetic", magneticState);
  }

  // Έλεγχος κατάστασης του PIR αισθητήρα
  bool newPirState = digitalRead(pirPin) == HIGH;  // HIGH = ανίχνευση κίνησης
  if (newPirState != pirState) {
    pirState = newPirState;
    sendToServer("pir", pirState);
  }

  // Έλεγχος εντολών από τον server για τον ρελέ
  checkRelayStateFromServer();

  delay(500);  // Καθυστέρηση για σταθερό έλεγχο
}
```

```
// Αποστολή δεδομένων στον server
void sendToServer(String sensorType, bool state) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverUrl);
    http.addHeader("Content-Type", "application/json");

    String jsonData = "{\"sensor\": \"" + sensorType + "\", \"state\": " + (state ? "true" : "false") + "}";
    int httpResponseCode = http.POST(jsonData);

    if (httpResponseCode > 0) {
      String response = http.getString();
      Serial.println("Απάντηση server: " + response);
    } else {
      Serial.println("Σφάλμα κατά την αποστολή: " + String(httpResponseCode));
    }

    http.end();
  }
}

// Έλεγχος κατάστασης ρελέ από τον server
void checkRelayStateFromServer() {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin("http://dikranis/api/relay");
    int httpResponseCode = http.GET();

    if (httpResponseCode > 0) {
      String response = http.getString();
      Serial.println("Κατάσταση ρελέ από server: " + response);

      if (response == "on") {
        digitalWrite(relayPin, HIGH);  // Ενεργοποίηση ρελέ
      } else if (response == "off") {
        digitalWrite(relayPin, LOW);   // Απενεργοποίηση ρελέ
      }
```

```
  } else {
    Serial.println("Σφάλμα κατά τον έλεγχο του ρελέ: " + String(httpResponseCode));
  }

  http.end();
 }
}
```

Αισθητήρες: Η μαγνητική παγίδα και ο PIR αισθητήρας συνδέονται σε αντίστοιχα GPIO pins του ESP32. Το ESP32 παρακολουθεί συνεχώς την κατάστασή τους (ανοιχτό/κλειστό για τη μαγνητική παγίδα και κίνηση/χωρίς κίνηση για τον PIR).

Αποστολή δεδομένων στον server: Όταν αλλάζει η κατάσταση κάποιου αισθητήρα, το ESP32 στέλνει ένα HTTP POST αίτημα στον Python server με τις πληροφορίες του αισθητήρα και την κατάστασή του.

Έλεγχος ρελέ: Το ESP32 στέλνει ένα HTTP GET αίτημα στον server για να ελέγξει την κατάσταση του ρελέ (on/off). Αν ο server απαντήσει με "on", το ESP32 ενεργοποιεί το ρελέ, αλλιώς το απενεργοποιεί.

4.2 Η Βάση μας

Το σύστημα παρακολούθησης και ελέγχου που αναπτύχθηκε για τη διαχείριση των ειδοποιήσεων και των αισθητήρων βασίζεται σε μια καλά σχεδιασμένη βάση δεδομένων MySQL. Η βάση δεδομένων αυτή εξυπηρετεί όλες τις ανάγκες καταγραφής, αποθήκευσης και ανάκτησης των δεδομένων που σχετίζονται με τα συμβάντα των αισθητήρων και τους χρήστες του συστήματος. Ακολουθεί μια ανάλυση των βασικών πινάκων που χρησιμοποιούνται στη βάση, με έμφαση στη λειτουργικότητά τους και τον ρόλο τους στο σύστημα.

Πίνακας alarms

Ο πίνακας alarms είναι η καρδιά του συστήματος καταγραφής συμβάντων, καθώς αποθηκεύει όλες τις ειδοποιήσεις που λαμβάνονται από τους αισθητήρες. Κάθε εγγραφή στον πίνακα αντιπροσωπεύει ένα συμβάν, περιλαμβάνοντας:

- **ID**: Ένας μοναδικός αριθμός για κάθε συμβάν.
- **komvosid**: Η αναφορά στον κόμβο που προκάλεσε το συμβάν.
- **userid**: Ο χρήστης που σχετίζεται με τον κόμβο αυτόν.

- **field1 έως field6**: Αποτελούν τα πεδία που υποδεικνύουν την κατάσταση των αισθητήρων (ενεργοποίηση ή απενεργοποίηση) κατά τη στιγμή του συμβάντος.
- **created_at**: Η χρονική στιγμή κατά την οποία καταγράφηκε το συμβάν.

Τα δεδομένα που αποθηκεύονται στον πίνακα αυτόν επιτρέπουν στον Python server να ανακτά τις τελευταίες ειδοποιήσεις για τους χρήστες και να ενημερώνει την εφαρμογή κινητού για τις τρέχουσες καταστάσεις των αισθητήρων.

Πίνακας kovmos

Ο πίνακας kovmos αντιπροσωπεύει τους κόμβους που χρησιμοποιούνται στο σύστημα. Κάθε κόμβος συνδέεται με έναν χρήστη και περιλαμβάνει αισθητήρες ή άλλες συσκευές που παρακολουθούνται ή ελέγχονται. Ο πίνακας αυτός περιλαμβάνει τις ακόλουθες στήλες:

- **id**: Ένα μοναδικό αναγνωριστικό για κάθε κόμβο.
- **active**: Ένα πεδίο που υποδεικνύει αν ο κόμβος είναι ενεργός.
- **userid**: Το ID του χρήστη που ανήκει ο κόμβος.
- **name**: Το όνομα του κόμβου, το οποίο τον περιγράφει.
- **field1title έως field6title**: Περιγραφές για κάθε πεδίο που αντιστοιχεί σε αισθητήρα ή συσκευή.
- **field1enable έως field6enable**: Τα πεδία αυτά δείχνουν αν ο αντίστοιχος αισθητήρας ή συσκευή είναι ενεργοποιημένος και πρέπει να παρακολουθείται.

Αυτός ο πίνακας είναι σημαντικός για την παραμετροποίηση του συστήματος, καθώς επιτρέπει την ενεργοποίηση και απενεργοποίηση των αισθητήρων ανά κόμβο, δίνοντας στους χρήστες πλήρη έλεγχο των συσκευών τους.

Πίνακας users

Ο πίνακας users αποθηκεύει τα δεδομένα των χρηστών που έχουν πρόσβαση στο σύστημα. Περιλαμβάνει:

- **id**: Το μοναδικό ID του χρήστη.
- **active**: Ένα πεδίο που δείχνει αν ο χρήστης είναι ενεργός.
- **kind**: Ένα πεδίο που μπορεί να χρησιμοποιηθεί για την κατηγοριοποίηση των χρηστών (π.χ., διαχειριστές και απλοί χρήστες).
- **email, password**: Τα διαπιστευτήρια του χρήστη για την πρόσβαση στο σύστημα.
- **fname, lname**: Το όνομα και το επώνυμο του χρήστη.

Ο πίνακας αυτός χρησιμοποιείται για την αυθεντικοποίηση των χρηστών και τον έλεγχο των δικαιωμάτων πρόσβασης σε συγκεκριμένους κόμβους και δεδομένα.

Η βάση δεδομένων του συστήματος είναι σχεδιασμένη με τέτοιο τρόπο ώστε να υποστηρίζει τη διαχείριση συμβάντων από αισθητήρες και τον έλεγχο συσκευών. Κάθε πίνακας παίζει συγκεκριμένο ρόλο στην ομαλή λειτουργία του συστήματος, επιτρέποντας την καταγραφή των συμβάντων, την παρακολούθηση των αισθητήρων και την αυθεντικοποίηση των χρηστών.

4.3 Ασφάλεια

Η ασφάλεια είναι ένας από τους πιο κρίσιμους παράγοντες για την επιτυχία ενός συστήματος παρακολούθησης και ελέγχου, όπως αυτό που έχει αναπτυχθεί. Καθώς το σύστημα διαχειρίζεται δεδομένα αισθητήρων και επιτρέπει τον απομακρυσμένο έλεγχο συσκευών, η προστασία των δεδομένων και η ασφάλεια των χρηστών είναι προτεραιότητα. Παρακάτω αναλύονται τα σημαντικότερα μέτρα και στρατηγικές που πρέπει να ληφθούν υπόψη για την προστασία του συστήματος και των δεδομένων του.

Ασφάλεια Δεδομένων στη Βάση

Η βάση δεδομένων αποτελεί την κεντρική αποθήκη για όλες τις πληροφορίες σχετικά με τους αισθητήρες, τις συσκευές και τους χρήστες.

- **Κρυπτογράφηση δεδομένων**: Τα ευαίσθητα δεδομένα, όπως οι κωδικοί πρόσβασης των χρηστών, πρέπει να είναι κρυπτογραφημένα. Χρησιμοποιώντας σύγχρονες μεθόδους κρυπτογράφησης, όπως το **bcrypt** ή το **SHA-256**, τα δεδομένα παραμένουν ασφαλή ακόμη και σε περίπτωση παραβίασης.
- **Ελεγχόμενη πρόσβαση**: Η πρόσβαση στη βάση δεδομένων πρέπει να περιορίζεται μόνο στους εξουσιοδοτημένους χρήστες. Οι διαχειριστές της βάσης δεδομένων θα πρέπει να έχουν τα κατάλληλα δικαιώματα, ενώ κάθε χρήστης θα πρέπει να έχει περιορισμένη πρόσβαση στα δεδομένα που αφορούν μόνο τους κόμβους που του ανήκουν.
- **Καταγραφή συμβάντων**: Είναι σημαντικό να διατηρείται αρχείο των ενεργειών που εκτελούνται στη βάση δεδομένων. Με αυτόν τον τρόπο, οι διαχειριστές μπορούν να εντοπίσουν ύποπτες δραστηριότητες και να λάβουν άμεσα μέτρα.

Ασφάλεια Επικοινωνίας

Η επικοινωνία ανάμεσα στους αισθητήρες, τον Python Server και την εφαρμογή κινητού πρέπει να προστατεύεται από επιθέσεις υποκλοπής και αλλοίωσης δεδομένων. Μερικά από τα βασικά μέτρα είναι:

- **Κρυπτογράφηση της δικτυακής επικοινωνίας**: Όλη η επικοινωνία μεταξύ των αισθητήρων και του server, καθώς και μεταξύ της εφαρμογής και του server, πρέπει να είναι κρυπτογραφημένη αλλά λόγω δυσκολίας δεν πραγματοποιήθηκε.
- **Αυθεντικοποίηση και εξουσιοδότηση**: Κάθε συσκευή που συνδέεται με το σύστημα, είτε πρόκειται για αισθητήρα είτε για χρήστη μέσω της εφαρμογής, θα πρέπει να περνάει από διαδικασία αυθεντικοποίησης. Οι χρήστες θα πρέπει να εισάγουν τα διαπιστευτήριά τους (username και password), τα οποία θα επαληθεύονται από τον server, ενώ κάθε αισθητήρας θα πρέπει να έχει μοναδικό κωδικό ή κλειδί πρόσβασης.

Ασφάλεια Χρηστών

Η ασφάλεια των χρηστών και των διαπιστευτηρίων τους αποτελεί βασική προτεραιότητα. Τα μέτρα που πρέπει να ληφθούν περιλαμβάνουν:

- **Ισχυροί κωδικοί πρόσβασης**: Οι χρήστες θα πρέπει να ενθαρρύνονται να χρησιμοποιούν ισχυρούς κωδικούς πρόσβασης, οι οποίοι να περιλαμβάνουν κεφαλαία και πεζά γράμματα, αριθμούς και ειδικούς χαρακτήρες. Επιπλέον, το σύστημα μπορεί να εφαρμόσει ελέγχους για να διασφαλίσει ότι οι κωδικοί πρόσβασης είναι αρκετά ισχυροί.

Ασφάλεια της Εφαρμογής

Η εφαρμογή κινητού πρέπει επίσης να προστατεύεται από τρωτά σημεία που μπορεί να εκμεταλλευτούν κακόβουλοι χρήστες. Κάποια από τα βασικά μέτρα είναι:

- **Αποτροπή αποθήκευσης διαπιστευτηρίων σε μη ασφαλή μορφή**: Τα στοιχεία εισόδου των χρηστών δεν πρέπει ποτέ να αποθηκεύονται σε απλό κείμενο στη συσκευή.
- **Συνεχής ενημέρωση της εφαρμογής**: Οι ενημερώσεις του λογισμικού πρέπει να περιλαμβάνουν τα τελευταία patches ασφαλείας, έτσι ώστε να διορθώνονται τυχόν κενά ασφαλείας που ανακαλύπτονται.

Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης

Το παρόν σύστημα παρακολούθησης και ελέγχου που αναπτύχθηκε συνδυάζει ένα δίκτυο αισθητήρων και συσκευών με έναν κεντρικό Python Server και μια βάση δεδομένων MySQL. Ο σχεδιασμός του παρέχει τη δυνατότητα καταγραφής συμβάντων σε πραγματικό χρόνο και απομακρυσμένης διαχείρισης μέσω μιας εφαρμογής κινητού. Η διασύνδεση των αισθητήρων με τον server εξασφαλίζει τη συνεχή παρακολούθηση των χώρων, ενώ οι χρήστες μπορούν να λάβουν άμεσες ειδοποιήσεις και να ελέγξουν τις συσκευές μέσω του κινητού τους τηλεφώνου.

Παρά τα θετικά αποτελέσματα, αναγνωρίζονται κάποιες περιοχές όπου μπορούν να γίνουν βελτιώσεις για την ενίσχυση της απόδοσης, της ασφάλειας και της εμπειρίας χρήστη.

Αν και η χρήση της κρυπτογράφησης HTTPS για την επικοινωνία εξασφαλίζει ασφαλείς μεταφορές δεδομένων, θα μπορούσε να εφαρμοστεί περαιτέρω πολυπαραγοντική αυθεντικοποίηση (MFA) για τους χρήστες, παρέχοντας ένα δεύτερο επίπεδο προστασίας. Αυτή η μέθοδος θα εξασφάλιζε ότι ακόμα και αν κάποιος αποκτήσει πρόσβαση σε έναν κωδικό, δεν θα μπορεί να εισέλθει στο σύστημα χωρίς τον δεύτερο παράγοντα, όπως κωδικό μίας χρήσης (OTP) ή επαλήθευση μέσω email.

Μια πιθανή επέκταση του συστήματος είναι η ενσωμάτωση αλγορίθμων μηχανικής μάθησης για την ανάλυση των δεδομένων από τους αισθητήρες. Η ανάλυση αυτή θα μπορούσε να οδηγήσει σε προβλέψεις για μελλοντικά γεγονότα, όπως η πρόβλεψη κινδύνου ή η ανίχνευση ανωμαλιών στη συμπεριφορά των αισθητήρων. Για παράδειγμα, το σύστημα θα μπορούσε να μάθει τα μοτίβα κίνησης σε έναν χώρο και να ειδοποιήσει για ανωμαλίες, όπως αναπάντεχες αλλαγές στη δραστηριότητα.

Η εφαρμογή κινητού, αν και απλή και λειτουργική, μπορεί να βελτιωθεί περαιτέρω από άποψη εμπειρίας χρήστη (UX). Μπορεί να προστεθεί ένα dashboard με γραφήματα και οπτικοποιήσεις δεδομένων, που θα επιτρέπουν στον χρήστη να κατανοεί γρήγορα την κατάσταση του συστήματος και των αισθητήρων. Επίσης, η δυνατότητα προσαρμογής των ειδοποιήσεων, ώστε ο χρήστης να λαμβάνει μόνο τις ειδοποιήσεις που θεωρεί σημαντικές, θα βελτίωνε την αλληλεπίδραση με την εφαρμογή.

Μια περαιτέρω ενίσχυση της ασφάλειας της βάσης δεδομένων θα μπορούσε να περιλαμβάνει τη συνεχή παρακολούθηση των ενεργειών που εκτελούνται στη βάση, με τη δημιουργία logs που θα καταγράφουν κάθε αλλαγή στα δεδομένα. Η υλοποίηση αυτοματοποιημένων διαδικασιών ελέγχου, που θα ανιχνεύουν μη εξουσιοδοτημένες ή ύποπτες δραστηριότητες, θα μπορούσε να αποτρέψει απόπειρες παραβίασης ή κακόβουλες ενέργειες.

Η επέκταση του συστήματος για να υποστηρίξει περισσότερα είδη συσκευών και πρωτοκόλλων επικοινωνίας (π.χ. Zigbee, LoRa) θα το καταστήσει πιο ευέλικτο και ικανό να λειτουργήσει σε πιο απαιτητικά περιβάλλοντα με μεγαλύτερες ανάγκες διασύνδεσης και απομακρυσμένης διαχείρισης.

Το σύστημα παρακολούθησης που αναπτύχθηκε αποδείχθηκε αποτελεσματικό στην παρακολούθηση και καταγραφή συμβάντων από διάφορους αισθητήρες. Οι μελλοντικές βελτιώσεις θα μπορούσαν να αυξήσουν την ασφάλεια, την απόδοση και τη συνολική χρηστικότητα, ενώ η ενσωμάτωση σύγχρονων τεχνολογιών θα ενίσχυε την επεκτασιμότητα και την προσαρμοστικότητα του συστήματος σε διαφορετικές απαιτήσεις και εφαρμογές.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] https://en.wikipedia.org/wiki/Google_Nest

[2] https://www.amazon.com/ring-security-system/s?k=ring+security+system

[3] https://www.apptunix.com/blog/wp-content/uploads/sites/3/2021/08/popular-apps-doveloped-with-python.jpg

[4] Lutz, M. (2013). Learning Python. O'Reilly Media.

[5] Python vs C++ vs Java – Which One Should I Learn? (geeksforgeeks.org)

[6] Differences Between Python, Java, and C++ (javatpoint.com)

[7] https://flutter.dev/

[8] Woolery, M. (2020). Beginning Flutter: A Hands On Guide to App Development.

[9] https://dev.mysql.com/doc/

DuBois, P. (2014). MySQL Cookbook. O'Reilly Media.

ΠΑΡΑΡΤΗΜΑ Α

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

Python Server

```
from flask import Flask, request, jsonify
from flask_cors import CORS  # Ενεργοποίηση του CORS
import mysql.connector
from datetime import datetime

app = Flask(__name__)
CORS(app)  # Ενεργοποίηση του CORS για όλα τα routes

# Σύνδεση με τη βάση δεδομένων
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="dikranis"
)

# Διαδρομή για εισαγωγή δεδομένων alarm μέσω GET
@app.route('/alarm', methods=['GET'])
def add_alarm():
    komvosid = request.args.get('komvosid')
    fields = [request.args.get(f'field{i}', 0) for i in range(1, 7)]

    cursor = db.cursor(dictionary=True)
    cursor.execute("SELECT * FROM kovmos WHERE id=%s AND active=1", (komvosid,))
    komvos = cursor.fetchone()

    if not komvos:
        return jsonify({'error': 'Invalid node'}), 400

    cursor.execute("""
        INSERT INTO alarms (komvosid, userid, field1, field2, field3, field4, field5, field6,
created_at)
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
    """, (komvosid, komvos['userid'], *fields, datetime.now()))
    db.commit()

    return jsonify({'status': 'Alarm added successfully'}), 201

# Διαδρομή για επιστροφή των τελευταίων alarms για τον χρήστη
@app.route('/alarms/<int:userid>', methods=['GET'])
def get_alarms(userid):
    cursor = db.cursor(dictionary=True)
```

```
    # Βρίσκουμε όλους τους κόμβους που ανήκουν στον χρήστη
    cursor.execute("SELECT * FROM kovmos WHERE userid=%s AND active=1", (userid,))
    nodes = cursor.fetchall()

    if not nodes:
        return jsonify({'error': 'No nodes found for user'}), 404

    alarms = []

    for node in nodes:
        # Παίρνουμε το τελευταίο alarm για κάθε κόμβο
        cursor.execute("""
            SELECT * FROM alarms
            WHERE komvosid=%s
            ORDER BY created_at DESC LIMIT 1
        """, (node['id'],))
        last_alarm = cursor.fetchone()

        if last_alarm:
            # Δημιουργούμε τη δομή για το alarm
            alarm_data = {
                'komvosid': last_alarm['komvosid'],
                'alarm': {
                    'field1': last_alarm['field1'],
                    'field2': last_alarm['field2'],
                    'field3': last_alarm['field3'],
                    'field4': last_alarm['field4'],
                    'field5': last_alarm['field5'],
                    'field6': last_alarm['field6'],
                },
                'timestamp': last_alarm['created_at']
            }
            alarms.append(alarm_data)

    return jsonify({'alarms': alarms}), 200


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

Δείγμα από Flutter

```
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http; // Add http package import
import 'dart:convert';

void main() {
  runApp(MyApp());
```

```
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Dikranis App',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: HomePage(), // Αρχική σελίδα
      debugShowCheckedModeBanner: false, // Απενεργοποιεί το banner DEBUG
    );
  }
}

// Αρχική σελίδα με το κουμπί "Σύνδεση"
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Dikranis App'),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            // Πηγαίνουμε στη σελίδα σύνδεσης όταν πατήσουμε το κουμπί
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => LoginPage()),
            );
          },
          style: ElevatedButton.styleFrom(
            padding: EdgeInsets.symmetric(horizontal: 50, vertical: 20),
            textStyle: TextStyle(fontSize: 20),
          ),
          child: Text('Σύνδεση'),
        ),
      ),
```

```
    );
  }
}

// Σελίδα σύνδεσης με πεδία Username και Password
class LoginPage extends StatelessWidget {
  final TextEditingController _usernameController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Σύνδεση'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            TextField(
              controller: _usernameController,
              decoration: InputDecoration(
                labelText: 'Username',
                border: OutlineInputBorder(),
              ),
            ),
            SizedBox(height: 16),
            TextField(
              controller: _passwordController,
              decoration: InputDecoration(
                labelText: 'Password',
                border: OutlineInputBorder(),
              ),
              obscureText: true,
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: () {
                // Μόλις πατήσει το κουμπί, πηγαίνουμε στη σελίδα NodePage
```

```
                Navigator.pushReplacement(
                  context,
                  MaterialPageRoute(builder: (context) => AlarmPage()),
                );
              },
              style: ElevatedButton.styleFrom(
                padding: EdgeInsets.symmetric(horizontal: 50, vertical: 20),
                textStyle: TextStyle(fontSize: 18),
              ),
              child: Text('Σύνδεση'),
            ),
          ],
        ),
      ),
    );
  }
}

// Η σελίδα με τα alarms που είχες φτιάξει
class AlarmPage extends StatefulWidget {
  @override
  _AlarmPageState createState() => _AlarmPageState();
}

class _AlarmPageState extends State<AlarmPage> {
  final String serverUrl = 'http://localhost:5000'; // Προσθήκη της διεύθυνσης του server
  List<dynamic> alarms = [];
  bool isSending = false;

  // Λειτουργία για αποστολή alarm στον server
  Future<void> sendAlarm() async {
    setState(() {
      isSending = true;
    });

    final response = await http.get(
      Uri.parse(
        '$serverUrl/alarm?komvosid=1&field1=1&field2=0&field3=0&field4=0&field5=0&field6=0',
      ),
    );
```

```
  if (response.statusCode == 201) {
    ScaffoldMessenger.of(context)
        .showSnackBar(SnackBar(content: Text('Alarm sent successfully!')));
  } else {
    ScaffoldMessenger.of(context)
        .showSnackBar(SnackBar(content: Text('Failed to send alarm.')));
  }

  setState(() {
    isSending = false;
  });
}

// Λειτουργία για ανάκτηση των τελευταίων alarms για έναν χρήστη
Future<void> fetchAlarms(int userid) async {
  final response = await http.get(Uri.parse('$serverUrl/alarms/$userid'));

  if (response.statusCode == 200) {
    setState(() {
      alarms = jsonDecode(response.body)['alarms'];
    });
  } else {
    ScaffoldMessenger.of(context)
        .showSnackBar(SnackBar(content: Text('Failed to fetch alarms.')));
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Dikranis Alarm'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        children: [
          ElevatedButton(
            onPressed: isSending ? null : sendAlarm,
```

```
  child: isSending
      ? CircularProgressIndicator()
      : Text('Ενεργοποίηση συσκευής (field1)'),
),
SizedBox(height: 20),
ElevatedButton(
  onPressed: () => fetchAlarms(1), // Υποθέτουμε user id = 1
  child: Text('Δώσε τα σήματα συναγερμού'),
),
SizedBox(height: 20),
alarms.isEmpty
    ? Text('No alarms found')
    : Expanded(
        child: SingleChildScrollView(
          scrollDirection: Axis.horizontal,
          child: DataTable(
            columns: const [
              DataColumn(label: SizedBox(width: 20, child: Text('ID'))),
              DataColumn(label: Text('F1')),
              DataColumn(label: Text('F2')),
              DataColumn(label: Text('F3')),
              DataColumn(label: Text('F4')),
              DataColumn(label: Text('F5')),
              DataColumn(label: Text('F6')),
              DataColumn(label: Text('Χρόνος')),
            ],
            rows: alarms
                .map((alarm) => DataRow(cells: [
                      DataCell(Text(alarm['komvosid'].toString())),
                      DataCell(Text(alarm['alarm']['field1'].toString())),
                      DataCell(Text(alarm['alarm']['field2'].toString())),
                      DataCell(Text(alarm['alarm']['field3'].toString())),
                      DataCell(Text(alarm['alarm']['field4'].toString())),
                      DataCell(Text(alarm['alarm']['field5'].toString())),
                      DataCell(Text(alarm['alarm']['field6'].toString())),
                      DataCell(Text(alarm['timestamp'])),
                    ]))
                .toList(),
          ),
        ),
```

```
                ),
          ],
        ),
      ),
    );
  }
}
```