ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Μελέτη και υλοποίηση μεθόδων ασφάλειας σε συστήματα διαδικτύου των πραγμάτων»

Φοιτητής

Αμπεριάδης Δημήτριος 510134

Επιβλέπων

Δρ. Κυριάκος Τσιακμάκης

ΦΕΒΡΟΥΑΡΙΟΣ 2023

Μελέτη και υλοποίηση μεθόδων ασφάλειας σε συστήματα διαδικτύου των πραγμάτων

Κωδικός: 19107

Φοιτητής: Αμπεριάδης Δημήτριος

Εισηγητής: Δρ Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 12-03-2021

Ημερομηνία περάτωσης Π.Ε. 13-01-2023

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

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

Περίληψη

Αυτή η πτυχιακή εργασία τη μελέτη συστημάτων ασφάλειας στο χώρο των IoT συστημάτων. Αναλύονται οι κώδικες που υλοποιήσαμε για κάθε κομμάτι ασφάλειας που απαιτείται από ένα σύστημα IoT όπως το σχεδιάσαμε. Πραγματοποιήθηκε η σχεδίαση μιας απλής τοπολογία με κεντρικούς και περιφερειακούς κόμβους να συνδέονται σε αυτόν. Οι κεντρικοί κόμβοι είναι server με σύνδεση στο διαδίκτυο και είναι server βάσης δεδομένων και ιστοσελίδων.

Τα δεδομένα μπορούμε να τα βλέπουμε μέσω ιστοσελίδας που υποστηρίζεται από τον κεντρικό, αφού συνδεθούμε σε αυτή. Αν οι κόμβοι μας είναι κατοχυρωμένοι στην βάση και ανήκουν σε ένα από τους κεντρικούς κόμβους που έχουμε δηλώσει στη βάση τότε μπορούμε να δούμε τις μετρήσεις των αισθητήρων.

« Study and implementation of security methods in internet of things systems »

Abstract

This thesis studies security systems in the field of IoT systems. We develop the codes used for every piece of security required by an IoT system. A simple topology was designed with central and peripheral nodes connected to it. Host nodes are server with internet connection and use database and web page server.

The data displayed in a web page supported by the central node. If our nodes are registered in the database and belong to one of the central nodes we can see the measurements of the sensors.

Ευχαριστίες

Θέλω να ευχαριστήσω όσους με βοήθησαν να υλοποιήσω αυτό το έργο και να συντάξω το κείμενο. Να ευχαριστήσω τον επιβλέποντα μου για την συνεχή επικοινωνία και τη βοήθεια του στους κώδικες και τον προγραμματισμό.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Το Διαδίκτυο των Πραγμάτων (IoT) αναφέρεται σε ένα αναπτυσσόμενο δίκτυο συνδεδεμένων αντικειμένων ικανών να συλλέγουν και να ανταλλάσσουν δεδομένα χρησιμοποιώντας ενσωματωμένους αισθητήρες.

Η ασφάλεια στο IoT είναι η προστασία των συσκευών Διαδικτύου και των δικτύων με τα οποία είναι συνδεδεμένα από απειλές και παραβιάσεις, προστατεύοντας, εντοπίζοντας και παρακολουθώντας κινδύνους, ενώ παράλληλα βοηθά στη διόρθωση ευπαθειών από μια σειρά συσκευών που μπορεί να θέτουν κινδύνους για την ασφάλεια μιας επιχείρησής ή σπιτιού.

Μαζί με την κατανόηση της ασφάλειας του IoT, είναι σημαντικό να σημειωθούν οι μεγαλύτερες προκλήσεις που αντιμετωπίζει η ασφάλεια του IoT. Οι συσκευές IoT δεν κατασκευάστηκαν με γνώμονα την ασφάλεια, οδηγώντας σε πιθανές ευπάθειες σε ένα σύστημα πολλαπλών συσκευών. Στην πλειονότητα των περιπτώσεων, δεν υπάρχει τρόπος εγκατάστασης λογισμικού ασφαλείας στην ίδια τη συσκευή. Επιπλέον, μερικές φορές αποστέλλονται με κακόβουλο λογισμικό, το οποίο στη συνέχεια μολύνει το δίκτυο στο οποίο είναι συνδεδεμένοι.

Η ασφάλεια δικτύου δεν έχει τη δυνατότητα να ανιχνεύσει συσκευές IoT που είναι συνδεδεμένες σε αυτό ή/και την ορατότητα να γνωρίζει ποιες συσκευές επικοινωνούν μέσω του δικτύου.

Οι απαιτήσεις IoT και ασφάλειας μπορούν να επιτευχθούν μόνο με μια ολοκληρωμένη λύση που παρέχει ορατότητα, τμηματοποίηση και προστασία σε ολόκληρη την υποδομή του δικτύου.

Στην εργασία αυτή υλοποιήθηκε με κώδικα και προτείνεται η ασφάλεια:

στη μέτρηση των αισθητήρων από τον κόμβο

στη μετάδοση δεδομένων από κόμβο σε κεντρικό gateway

στη αυθεντικοποίηση του χρήστη και αντιστοίχιση κόμβων με αυτόν

στη αυθεντικοποίηση του χρήστη στην πρόσβαση του στο κεντρικό κόμβο - gateway

1.2 Δομή της εργασίας

Στο πρώτο κεφάλαιο παρουσιάζεται η εισαγωγή της εργασίας, οι στόχοι και η δομή της.

Στο δεύτερο κεφάλαιο παρουσιάζεται η εισαγωγή στα συστήματα ασφάλειας σε IoT.

Στο τρίτο κεφάλαιο περιγράφονται τεχνολογικά εργαλεία για τη μελέτη, την υλοποίηση και χρήση των τεχνικών ασφαλείας σε συστήματα IoT.

Στο τέταρτο κεφάλαιο αναλύονται οι εφαρμογές για την ασφάλεια σε συστήματα IoT

Στο τελευταίο κεφάλαιο παρουσιάζονται τα συμπεράσματα της εργασίας και θέματα για μελλοντική έρευνα.

Στο τέλος της εργασίας υπάρχει το παράρτημα με τους κώδικες.

Κεφάλαιο 2ο: Εισαγωγή στα συστήματα ασφάλειας σε IoT

2.1 Εισαγωγή στο IoT

Το Διαδίκτυο των Πραγμάτων (IoT) περιγράφει το δίκτυο φυσικών αντικειμένων — «πράγματα» — που είναι ενσωματωμένα με αισθητήρες, λογισμικό και άλλες τεχνολογίες με σκοπό τη σύνδεση και την ανταλλαγή δεδομένων με άλλες συσκευές και συστήματα μέσω του Διαδικτύου. Αυτές οι συσκευές κυμαίνονται από συνηθισμένα οικιακά αντικείμενα έως εξελιγμένα βιομηχανικά εργαλεία. Με περισσότερες από 7 δισεκατομμύρια συνδεδεμένες συσκευές IoT σήμερα, οι ειδικοί αναμένουν ότι ο αριθμός αυτός θα αυξηθεί σε 10 δισεκατομμύρια έως το 2020 και 22 δισεκατομμύρια έως το 2025.

Τα τελευταία χρόνια, το IoT έχει γίνει μια από τις πιο σημαντικές τεχνολογίες. Μπορούν να συνδεθούν αντικείμενα - συσκευές κουζίνας, αυτοκίνητα, θερμοστάτες στο διαδίκτυο μέσω ενσωματωμένων συσκευών ώστε να είναι δυνατή η απρόσκοπτη επικοινωνία μεταξύ ανθρώπων, διαδικασιών και πραγμάτων.

Μέσω υπολογιστών χαμηλού κόστους όπως το esp32, του cloud, των μεγάλων δεδομένων και τεχνολογιών κινητής τηλεφωνίας, οι συσκευές μπορούν να μοιράζονται και να συλλέγουν δεδομένα με ελάχιστη ανθρώπινη παρέμβαση. Σε αυτόν τον κόσμο, τα ψηφιακά συστήματα μπορούν να καταγράφουν, να παρακολουθούν και να προσαρμόζουν κάθε αλληλεπίδραση μεταξύ συνδεδεμένων συσκευών.

Μερικά από τα πλεονεκτήματα ενός IoT συστήματος είναι:

Πρόσβαση σε τεχνολογία αισθητήρων χαμηλού κόστους και χαμηλής ισχύος.

Οι προσιτές και αξιόπιστοι αισθητήρες καθιστούν δυνατή την τεχνολογία IoT για περισσότερους κατασκευαστές.

Συνδεσιμότητα.

Μια σειρά από πρωτόκολλα δικτύου για το Διαδίκτυο έχουν καταστήσει εύκολη τη σύνδεση αισθητήρων στο cloud και σε άλλες συσκευές για αποτελεσματική μεταφορά δεδομένων.

Πλατφόρμες υπολογιστικού νέφους - cloud.

Η αύξηση της διαθεσιμότητας πλατφορμών cloud δίνει τη δυνατότητα τόσο στις επιχειρήσεις όσο και στους καταναλωτές να έχουν πρόσβαση στην υποδομή που χρειάζονται για να αναβαθμιστούν χωρίς να χρειάζεται να τα διαχειρίζονται όλα.

Μηχανική μάθηση και ανάλυση.

Με την πρόοδο στη μηχανική μάθηση και της ανάλυσης και με την πρόσβαση σε ποικίλες και μεγάλες ποσότητες δεδομένων που είναι αποθηκευμένα στο cloud, οι επιχειρήσεις μπορούν να συλλέγουν πληροφορίες γρηγορότερα και πιο εύκολα.

Τεχνητή νοημοσύνη.

Οι πρόοδοι στα νευρωνικά δίκτυα έχουν φέρει την επεξεργασία φυσικής γλώσσας (NLP) σε συσκευές IoT (όπως Cortana ,Alexa, Siri) και τις έχουν κάνει ελκυστικές και προσιτές για οικιακή χρήση.

2.2 Ασφάλεια

Η ασφάλεια του IoT είναι το τμήμα τεχνολογίας που επικεντρώνεται στην προστασία των συνδεδεμένων συσκευών και δικτύων στο διαδίκτυο των πραγμάτων. Το IoT περιλαμβάνει την προσθήκη σύνδεσης στο Διαδίκτυο σε ένα σύστημα αλληλένδετων υπολογιστικών συσκευών, μηχανικών και ψηφιακών μηχανών, αντικειμένων και ανθρώπων. Σε κάθε συσκευή παρέχεται ένα μοναδικό αναγνωριστικό και η δυνατότητα αυτόματης μεταφοράς δεδομένων μέσω δικτύου. Το να επιτρέπεται στις συσκευές να συνδέονται στο διαδίκτυο τις ανοίγει σε μια σειρά από σοβαρά τρωτά σημεία, εάν δεν προστατεύονται σωστά.

Ορισμένα περιστατικά στα οποία χρησιμοποιήθηκε μια κοινή συσκευή IoT για διείσδυση και επίθεση στο μεγαλύτερο δίκτυο ή διαδίκτυο έχει επιστήσει την προσοχή όλων των εμπλεκόμενων στην ανάγκη για ασφάλεια ενός IoT συστήματος.

Η ασφάλεια του IoT περιλαμβάνει ένα ευρύ φάσμα τεχνικών, στρατηγικών, πρωτοκόλλων και ενεργειών που στοχεύουν στην μελέτη των αυξανόμενων τρωτών σημείων του IoT των σύγχρονων επιχειρήσεων.

Η ασφάλεια του IoT αναφέρεται στις μεθόδους προστασίας που χρησιμοποιούνται για την ασφάλεια συσκευών συνδεδεμένων στο διαδίκτυο ή δικτύου. Από ρολόγια μέχρι θερμοστάτες και κονσόλες βιντεοπαιχνιδιών, σχεδόν κάθε τεχνολογική συσκευή έχει τη δυνατότητα να αλληλεπιδρά με το Διαδίκτυο ή άλλες συσκευές.

Η ασφάλεια του IoT είναι η οικογένεια τεχνικών, στρατηγικών και εργαλείων που χρησιμοποιούνται για την προστασία αυτών των συσκευών από τον κίνδυνο παραβίασης. Κατά κάποιο τρόπο, είναι η συνδεσιμότητα που είναι εγγενής στο IoT που κάνει αυτές τις συσκευές όλο και πιο ευάλωτες σε κυβερνοεπιθέσεις.

Επειδή το IoT είναι τόσο ευρύ, η ασφάλεια του IoT είναι ακόμη ευρύτερη. Αυτό είχε ως αποτέλεσμα μια ποικιλία μεθοδολογιών να εμπίπτουν στην ομπρέλα της ασφάλειας του IoT. Η ασφάλεια της διεπαφής προγράμματος εφαρμογής (API), ο έλεγχος ταυτότητας της υποδομής δημόσιου κλειδιού και

η ασφάλεια δικτύου είναι μερικές μόνο από τις μεθόδους που μπορούν να χρησιμοποιήσουν οι προγραμματιστές για την καταπολέμηση της αυξανόμενης απειλής του εγκλήματος στον κυβερνοχώρο και της τρομοκρατίας στον κυβερνοχώρο που έχουν τις ρίζες τους σε ευάλωτες συσκευές IoT.

Όσο περισσότεροι τρόποι για να μπορούν οι συσκευές να συνδέονται μεταξύ τους, τόσο περισσότεροι είναι οι τρόποι με τους οποίους οι φορείς απειλών μπορούν να τις υποκλέψουν. Κάποια πρωτόκολλα όπως το HTTP (Hypertext Transfer Protocol) και το API είναι μερικά μόνο από τα κανάλια στα οποία βασίζονται οι συσκευές IoT και τα οποία μπορούν να υποκλαπούν από τους χάκερ.

Το IoT δεν περιλαμβάνει αυστηρά συσκευές που βασίζονται στο Διαδίκτυο. Οι συσκευές που χρησιμοποιούν τεχνολογία Bluetooth υπολογίζονται επίσης ως συσκευές IoT και, ως εκ τούτου, απαιτούν ασφάλεια IoT. Τέτοιες παραλείψεις όπως αυτή συνέβαλαν στην πρόσφατη αύξηση των παραβιάσεων δεδομένων που σχετίζονται με το IoT.

Σε αντίθεση με άλλες τεχνολογίες, οι συσκευές IoT έχουν μεγάλη πιθανότητα επίθεσης λόγω της συνδεσιμότητας που υποστηρίζεται από το Διαδίκτυο. Αν και αυτή η προσβασιμότητα είναι εξαιρετικά πολύτιμη, παρέχει επίσης στους χάκερ την ευκαιρία να αλληλεπιδρούν με συσκευές εξ αποστάσεως. Αυτός είναι ο λόγος για τον οποίο το phishing είναι ιδιαίτερα αποτελεσματικό. Η ασφάλεια του IoT, όπως και η ασφάλεια του cloud, πρέπει να προστατεύσει μεγάλο αριθμό σημείων εισόδου για την προστασία των περιουσιακών στοιχείων και δεδομένων.

Καθώς οι εταιρείες συνεχίζουν τους ψηφιακούς μετασχηματισμούς της επιχείρησής τους, το ίδιο ισχύει και για ορισμένες βιομηχανίες όπως η αυτοκινητοβιομηχανία και τομείς υγείας που έχουν πρόσφατα επεκτείνει την χρήση των συσκευών IoT για να γίνουν πιο παραγωγικές και αποδοτικές οικονομικά. Αυτή η ψηφιακή επανάσταση, ωστόσο, οδήγησε επίσης σε μεγαλύτερη τεχνολογική εξάρτηση.

Η εξάρτηση από την τεχνολογία μπορεί να ενισχύσει τις συνέπειες μιας επιτυχούς παραβίασης δεδομένων. Αυτό που το κάνει να ανησυχεί είναι ότι αυτές οι βιομηχανίες βασίζονται πλέον σε ένα κομμάτι τεχνολογίας που είναι εγγενώς πιο ευάλωτο: τις συσκευές IoT.

Πρέπει οι εταιρείες αυτές να επενδύσουν κάποιο ποσό των χρημάτων και των πόρων για την ασφάλεια αυτών των συσκευών. Αυτή η έλλειψη προοπτικής του κλάδου έχει εκθέσει άσκοπα πολλούς οργανισμούς και κατασκευαστές σε αυξημένες απειλές για την ασφάλεια στον κυβερνοχώρο.

Η έλλειψη προνοητικότητας δεν είναι το μόνο ζήτημα ασφάλειας του IoT που αντιμετωπίζουν οι πρόσφατα ψηφιοποιημένες βιομηχανίες. Μια άλλη σημαντική ανησυχία σχετικά με την ασφάλεια του IoT είναι οι περιορισμοί πόρων πολλών από αυτές τις συσκευές. Συνήθως οι συσκευές IoT δεν έχουν την υπολογιστική ισχύ για να ενσωματώσουν εξελιγμένα τείχη προστασίας ή λογισμικό προστασίας από ιούς.

Μερικοί τρόποι ενίσχυσης της ασφάλειας

Τα περισσότερα ζητήματα ασφάλειας-παραβίασης μπορούν να ξεπεραστούν με καλύτερη προετοιμασία, ιδιαίτερα κατά τη διαδικασία έρευνας και ανάπτυξης στην αρχή οποιασδήποτε ανάπτυξης συσκευών IoT. Η ενεργοποίηση της ασφάλειας από προεπιλογή είναι κρίσιμης σημασίας, καθώς και η παροχή των πιο πρόσφατων λειτουργικών συστημάτων και η χρήση ασφαλούς υλικού.

Οι προγραμματιστές IoT θα πρέπει να προσέχουν τα τρωτά σημεία της κυβερνοασφάλειας σε κάθε στάδιο ανάπτυξης.

Το ‘σύστημα δημοσίου κλειδιού’ είναι ένας εξαιρετικός τρόπος για ασφαλείς συνδέσεις πελάτη-διακομιστή μεταξύ πολλαπλών δικτυωμένων συσκευών. Χρησιμοποιώντας ένα ασύμμετρο κρυπτοσύστημα δύο κλειδιών, το ‘σύστημα δημοσίου κλειδιού’ είναι σε θέση να διευκολύνει την κρυπτογράφηση και την αποκρυπτογράφηση των ιδιωτικών μηνυμάτων και τις αλληλεπιδράσεις χρησιμοποιώντας ψηφιακά πιστοποιητικά. Αυτά τα συστήματα συμβάλλουν στην προστασία των πληροφοριών καθαρού κειμένου που εισάγουν οι χρήστες σε ιστότοπους για την ολοκλήρωση ιδιωτικών συναλλαγών.

Τα δίκτυα παρέχουν μια τεράστια ευκαιρία στους παράγοντες απειλών να ελέγχουν εξ αποστάσεως τις συσκευές IoT άλλων. Επειδή τα δίκτυα περιλαμβάνουν τόσο ψηφιακά όσο και φυσικά στοιχεία, η εσωτερική ασφάλεια IoT θα πρέπει να αφορά και τους δύο τύπους σημείων πρόσβασης. Η προστασία ενός δικτύου IoT περιλαμβάνει τη διασφάλιση της ασφάλειας της θύρας, την απενεργοποίηση της προώθησης θυρών και το ποτέ άνοιγμα των θυρών όταν δεν χρειάζεται. Επίσης, καλό θα ήταν η χρήση anti-malware, τείχη προστασίας και antivirus.

Τα API είναι η ραχοκοκαλιά των πιο εξελιγμένων ιστότοπων. Δυστυχώς, οι χάκερ μπορούν να υπονομεύσουν αυτά τα κανάλια επικοινωνίας, καθιστώντας την ασφάλεια API απαραίτητη για την προστασία της ακεραιότητας των δεδομένων που αποστέλλονται από συσκευές IoT σε συστήματα υποστήριξης και διασφαλίζοντας ότι μόνο εξουσιοδοτημένες συσκευές, προγραμματιστές και εφαρμογές επικοινωνούν με API.

Επίσης ο έλεγχος στο NAC (Network access control) μπορεί να βοηθήσει στον εντοπισμό και την απογραφή συσκευών IoT που συνδέονται σε ένα δίκτυο. Αυτό θα παρέχει μια βάση για συσκευές παρακολούθησης και παρακολούθησης.

Οι συσκευές IoT που πρέπει να συνδεθούν απευθείας στο διαδίκτυο θα πρέπει να τμηματοποιούνται στα δικά τους δίκτυα και να έχουν περιορισμένη πρόσβαση στο τοπικό-εταιρικό δίκτυο. Τα τμήματα δικτύου θα πρέπει να παρακολουθούν για παράξενη δραστηριότητα, όπου μπορούν να ληφθούν μέτρα, εάν εντοπιστεί κάποιο πρόβλημα.

Τέλος θα πρέπει οι διαχειριστές, εταιρείες και οι χρήστες να ενημερώνονται για τους κινδύνους των συστημάτων IoT και να παρέχονται βήματα για να παραμείνουν ασφαλείς, όπως η ενημέρωση των προεπιλεγμένων διαπιστευτηρίων και η εφαρμογή ενημερώσεων λογισμικού. Οι χρήστες μπορούν επίσης να διαδραματίσουν κάποιο ρόλο στο να απαιτήσουν από τους κατασκευαστές συσκευών να δημιουργήσουν ασφαλείς συσκευές και να αρνηθούν να χρησιμοποιήσουν εκείνες που δεν πληρούν τα πρότυπα υψηλής ασφάλειας.

Οι μέθοδοι ασφάλειας IoT ποικίλλουν ανάλογα με τη συγκεκριμένη εφαρμογή IoT και τη θέση στο οικοσύστημα IoT. Για παράδειγμα, οι κατασκευαστές IoT, από κατασκευαστές προϊόντων έως εταιρείες ημιαγωγών θα πρέπει να επικεντρωθούν στην οικοδόμηση ασφάλειας από την αρχή, να καταστήσουν το υλικό απαραβίαστο, να δημιουργήσουν ασφαλές υλικό, να εξασφαλίσουν ασφαλείς αναβαθμίσεις, να παρέχουν ενημερώσεις του υλικολογισμικού.

Ο προγραμματιστής πρέπει να εστιάσει είναι στην ασφαλή ανάπτυξη λογισμικού και στην ασφαλή ενσωμάτωση. Για όσους αναπτύσσουν συστήματα IoT, η ασφάλεια υλικού και ο έλεγχος ταυτότητας είναι κρίσιμα μέτρα. Ομοίως, για τους χειριστές, η διατήρηση των συστημάτων ενημερωμένα, ο μετριασμός του κακόβουλου λογισμικού, ο έλεγχος, η προστασία της υποδομής και η προστασία των διαπιστευτηρίων είναι καίριας σημασίας. Ωστόσο, με οποιαδήποτε ανάπτυξη IoT, είναι σημαντικό να σταθμίζεται το κόστος της ασφάλειας έναντι των κινδύνων πριν από την εφαρμογή.

Κεφάλαιο 3ο: Τεχνολογικά εργαλεία για τη μελέτη, την υλοποίηση και χρήση των τεχνικών ασφαλείας σε συστήματα IoT

Στο κεφάλαιο αυτό θα περιγράφουν τα τεχνολογικά εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογών και προγραμμάτων για την μελέτη, την υλοποίηση και χρήση των τεχνικών ασφαλείας σε συστήματα IoT.

3.1 Γλώσσα Python

Η Python είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που δημιουργήθηκε από τον Guido Rossum το 1989. Είναι ιδανικά σχεδιασμένη για γρήγορη δημιουργία πρωτοτύπων σύνθετων εφαρμογών. Έχει διασυνδέσεις σε πολλές κλήσεις συστήματος και βιβλιοθήκες λειτουργικού συστήματος και είναι επεκτάσιμο σε C ή C++. Πολλές μεγάλες εταιρείες χρησιμοποιούν τη γλώσσα προγραμματισμού Python, συμπεριλαμβανομένων των NASA, Google, YouTube κ.λπ.

Python: Δυναμική γλώσσα προγραμματισμού που υποστηρίζει πολλούς διαφορετικούς προγραμματισμούς όπως

- Διαδικαστικός προγραμματισμός
- Αντικειμενοστραφής προγραμματισμός
- Λειτουργικός προγραμματισμός

Ο κώδικας Python εκτελείται στον διερμηνέα Python (παρόμοιος με την Java)

→ κωδικός ανεξάρτητος πλατφόρμας

Κάποια απλά χαρακτηριστικά

- Είναι script γλώσσα
- Είναι εξαιρετικά ευέλικτη γλώσσα για ανάπτυξη ιστοσελίδων, ανάλυση δεδομένων, συντήρηση διακομιστή, αριθμητική ανάλυση.
- Η σύνταξη είναι σαφής, εύκολη στην ανάγνωση και εκμάθηση (σχεδόν ψευδοκώδικας)
- Κοινή γλώσσα
- Διαισθητικός αντικειμενοστραφής προγραμματισμός
- Πλήρης αρθρωτή, ιεραρχικά πακέτα
- Πλήρης τυπική βιβλιοθήκη για πολλές εργασίες

- Μεγάλη κοινότητα
- Απλά επεκτάσιμη μέσω C/C++, βιβλιοθηκών C/C++

Επίσης υποστηρίζει

- Καμία δήλωση μεταβλητής
- Τα ονόματα των μεταβλητών μπορούν να αντιστοιχιστούν σε διαφορετικούς τύπους δεδομένων κατά τη διάρκεια ενός προγράμματος
- Τα χαρακτηριστικά ενός αντικειμένου ελέγχονται μόνο κατά το χρόνο εκτέλεσης
- Ένα αντικείμενο ορίζεται από τις μεθόδους και τα χαρακτηριστικά του

Οι κύριοι παράγοντες που αναφέρονται από τους χρήστες της Python φαίνεται να είναι οι εξής:

Ποιότητα λογισμικού

Για πολλούς, η εστίαση της Python στην αναγνωσιμότητα, τη συνοχή και την ποιότητα του λογισμικού γενικά την ξεχωρίζει από άλλα εργαλεία στον κόσμο του script. Ο κώδικας Python έχει σχεδιαστεί για να είναι αναγνώσιμος, και ως εκ τούτου επαναχρησιμοποιήσιμος και συντηρήσιμος - πολύ περισσότερο από τις παραδοσιακές γλώσσες δέσμης ενεργειών. Η ομοιομορφία του κώδικα Python καθιστά εύκολη την κατανόηση. Επιπλέον, η Python έχει βαθιά υποστήριξη για πιο προηγμένους μηχανισμούς επαναχρησιμοποίησης λογισμικού, όπως ο αντικειμενοστραφής προγραμματισμός (OOP).

Παραγωγικότητα προγραμματιστή

Η Python ενισχύει την παραγωγικότητα των προγραμματιστών πολλές φορές πέρα από τις μεταγλωττισμένες ή στατικά πληκτρολογημένες γλώσσες όπως η C, η C++ και η Java. Ο κώδικας Python είναι συνήθως το ένα τρίτο έως το ένα πέμπτο του μεγέθους ενός ισοδύναμου κώδικα C++ ή Java. Αυτό σημαίνει ότι υπάρχουν λιγότερα για να πληκτρολογηθούν και μειώνεται ο εντοπισμός σφαλμάτων και η διατήρηση κώδικα. Τα προγράμματα Python εκτελούνται αμέσως, χωρίς τα μακρά βήματα μεταγλώττισης και σύνδεσης που απαιτούνται από ορισμένα άλλα εργαλεία, ενισχύοντας περαιτέρω την ταχύτητα του προγραμματιστή.

Φορητότητα προγράμματος

Τα περισσότερα προγράμματα Python εκτελούνται αμετάβλητα σε όλες τις μεγάλες πλατφόρμες υπολογιστών. Η μεταφορά κώδικα Python μεταξύ Linux και Windows, για παράδειγμα, είναι συνήθως απλώς θέμα αντιγραφής του κώδικα μεταξύ μηχανών. Επιπλέον, η Python προσφέρει πολλαπλές

επιλογές για την κωδικοποίηση φορητών γραφικών διεπαφών χρήστη, προγραμμάτων πρόσβασης σε βάσεις δεδομένων, συστημάτων που βασίζονται στο διαδίκτυο και πολλά άλλα.

Βιβλιοθήκες υποστήριξης

Η Python συνοδεύεται από μια μεγάλη συλλογή προκατασκευασμένων και φορητών λειτουργιών, γνωστή ως τυπική βιβλιοθήκη. Αυτή η βιβλιοθήκη υποστηρίζει μια σειρά εργασιών προγραμματισμού σε επίπεδο εφαρμογής, από αντιστοίχιση μοτίβων κειμένου έως δέσμες ενεργειών δικτύου. Επιπλέον, η Python μπορεί να επεκταθεί τόσο με εγχώριες βιβλιοθήκες όσο και με μια τεράστια συλλογή λογισμικού υποστήριξης εφαρμογών τρίτων. Η Python προσφέρει εργαλεία για την κατασκευή ιστοσελίδων, αριθμητικό προγραμματισμό, πρόσβαση σε σειριακή θύρα, ανάπτυξη παιχνιδιών και πολλά άλλα.

Η επέκταση NumPy, για παράδειγμα, έχει περιγραφεί ως δωρεάν και πιο ισχυρό ισοδύναμο με το σύστημα αριθμητικού προγραμματισμού Matlab.

Ενοποίηση στοιχείων

Οι κώδικες Python μπορούν εύκολα να επικοινωνήσουν με άλλα μέρη μιας εφαρμογής, χρησιμοποιώντας μια ποικιλία μηχανισμών ενοποίησης. Τέτοιες ενσωματώσεις επιτρέπουν στην Python να χρησιμοποιείται ως εργαλείο προσαρμογής και επέκτασης προϊόντος. Σήμερα, ο κώδικας Python μπορεί να καλέσει βιβλιοθήκες C και C++, μπορεί να κληθεί από προγράμματα C και C++, μπορεί να ενσωματωθεί με στοιχεία Java και .NET, μπορεί να επικοινωνεί μέσω πλαισίων όπως το COM, μπορεί να διασυνδέεται με συσκευές μέσω σειριακών θυρών και μπορεί να αλληλεπιδρά μέσω δικτύων με διεπαφές..

3.2 Server Python

Η τυπική βιβλιοθήκη της Python συνοδεύεται από έναν ενσωματωμένο διακομιστή ιστού που μπορεί να χρησιμοποιηθεί για απλή επικοινωνία διακομιστή web-πελάτη. Ο αριθμός θύρας μπορεί να εκχωρηθεί μέσω προγραμματισμού και η πρόσβαση στον διακομιστή ιστού γίνεται μέσω αυτής της θύρας. Αν και δεν είναι ένας πλήρης διακομιστής web που μπορεί να αναλύσει πολλά είδη αρχείων, μπορεί να αναλύσει απλά στατικά αρχεία html και να τα εξυπηρετήσει απαντώντας τους με τους απαιτούμενους κωδικούς απόκρισης. Το παρακάτω πρόγραμμα ξεκινά έναν απλό διακομιστή web και τον ανοίγει στη θύρα 4000. Η επιτυχής λειτουργία του διακομιστή υποδεικνύεται από τον κωδικό απόκρισης 200 όπως φαίνεται στην έξοδο του προγράμματος.

```
import SimpleHTTPServer
import SocketServer

PORT = 4000

Handler = SimpleHTTPServer.SimpleHTTPRequestHandler

httpd = SocketServer.TCPServer(("", PORT), Handler)

httpd.serve_forever()
```

Επειδή αυτός ο server είναι πολύπλοκος κα δύσκολος να χρησιμοποιηθεί στην εργασία αυτή χρησιμοποιήθηκε ο server που διαθέτει ο Flask και τη βιβλιοθήκη requests, που θα αναλυθεί παρακάτω και υποστηρίζει και υπηρεσίες-μεθόδους API και Rest.

3.3 API και Python

Πολλές υπηρεσίες web, όπως το YouTube και το GitHub, καθιστούν τα δεδομένα τους προσβάσιμα σε εφαρμογές τρίτων μέσω μιας διεπαφής προγραμματισμού εφαρμογών (API). Ένας από τους πιο δημοφιλείς τρόπους δημιουργίας API είναι το στυλ αρχιτεκτονικής REST. Η Python παρέχει μερικά εξαιρετικά εργαλεία όχι μόνο για τη λήψη δεδομένων από REST API αλλά και για τη δημιουργία των Python REST API.

Το REST σημαίνει μεταφορά κατάστασης αναπαράστασης και είναι ένα στυλ αρχιτεκτονικής λογισμικού που ορίζει ένα μοτίβο για τις επικοινωνίες πελάτη και διακομιστή μέσω ενός δικτύου. Το REST παρέχει ένα σύνολο περιορισμών για την αρχιτεκτονική λογισμικού για την προώθηση της απόδοσης, της επεκτασιμότητας, της απλότητας και της αξιοπιστίας στο σύστημα.

Το REST ορίζει τους ακόλουθους αρχιτεκτονικούς περιορισμούς:

Stateless: Ο διακομιστής δεν θα διατηρεί καμία κατάσταση μεταξύ των αιτημάτων από τον πελάτη.

Client-server: Ο πελάτης και ο διακομιστής πρέπει να αποσυνδεθούν μεταξύ τους, επιτρέποντας στον καθένα να αναπτυχθεί ανεξάρτητα.

Με δυνατότητα προσωρινής αποθήκευσης: Τα δεδομένα που ανακτώνται από τον διακομιστή πρέπει να είναι αποθηκευμένα είτε από τον πελάτη είτε από τον διακομιστή.

Ομοιόμορφη διεπαφή: Ο διακομιστής θα παρέχει μια ενιαία διεπαφή για την πρόσβαση σε πόρους χωρίς να ορίζει την αναπαράστασή τους.

Επίπεδο σύστημα: Ο πελάτης μπορεί να έχει πρόσβαση στους πόρους του διακομιστή έμμεσα μέσω άλλων επιπέδων, όπως διακομιστή μεσολάβησης ή εξισορρόπησης φορτίου.

Κώδικας κατ' απαίτηση: Ο διακομιστής μπορεί να μεταφέρει κώδικα στον πελάτη που μπορεί να εκτελέσει, όπως JavaScript για μια εφαρμογή μιας σελίδας.

Το REST δεν είναι μια προδιαγραφή, αλλά ένα σύνολο οδηγιών για τον τρόπο αρχιτεκτονικής ενός συστήματος λογισμικού συνδεδεμένου στο δίκτυο.

Μια υπηρεσία Ιστού REST είναι οποιαδήποτε υπηρεσία Ιστού που συμμορφώνεται με τους περιορισμούς αρχιτεκτονικής REST. Αυτές οι υπηρεσίες web διαθέτουν τα δεδομένα τους στον έξω κόσμο μέσω ενός API. Τα API REST παρέχουν πρόσβαση σε δεδομένα υπηρεσιών ιστού μέσω δημόσιων διευθύνσεων URL ιστού.

Για παράδειγμα, εδώ είναι ένα από τα URL για το REST API:

https://amperiadisiot/node/id=2

Αυτή η διεύθυνση URL επιτρέπει να έχετε πρόσβαση σε πληροφορίες σχετικά με έναν συγκεκριμένο κόμβο. Μπορείτε να αποκτήσετε πρόσβαση σε δεδομένα από ένα REST API στέλνοντας ένα αίτημα HTTP σε μια συγκεκριμένη διεύθυνση URL και επεξεργάζοντας την απάντηση.

Μέθοδοι HTTP

Τα API REST έχουν μεθόδους HTTP όπως GET, POST και DELETE για να γνωρίζουν ποιες λειτουργίες πρέπει να εκτελούνται στους πόρους της υπηρεσίας Ιστού. Ένας πόρος είναι τα δεδομένα που είναι διαθέσιμα στην υπηρεσία Ιστού στα οποία είναι δυνατή η πρόσβαση και ο χειρισμός τους με αιτήματα HTTP στο REST API. Η μέθοδος HTTP λέει στο API ποια ενέργεια να εκτελέσει στον πόρο.

Ενώ υπάρχουν πολλές μέθοδοι HTTP, οι πέντε μέθοδοι που αναφέρονται παρακάτω είναι οι πιο συχνά χρησιμοποιούμενες με τα API REST:

Μέθοδος HTTP	
GET	Ανακτήστε έναν υπάρχοντα πόρο.

POST	Δημιουργήστε έναν νέο πόρο.
PUT	Ενημερώστε έναν υπάρχοντα πόρο.
PATCH	Ενημερώστε μερικώς έναν υπάρχοντα πόρο.
DELETE	Διαγραφή πόρου.

Μια εφαρμογή πελάτη REST API μπορεί να χρησιμοποιήσει αυτές τις πέντε μεθόδους HTTP για να διαχειριστεί την κατάσταση των πόρων στην υπηρεσία Ιστού.

Κωδικοί κατάστασης

Μόλις ένα REST API λάβει και επεξεργαστεί ένα αίτημα HTTP, θα επιστρέψει μια απάντηση HTTP. Σε αυτήν την απάντηση περιλαμβάνεται ένας κωδικός κατάστασης HTTP. Αυτός ο κωδικός παρέχει πληροφορίες σχετικά με τα αποτελέσματα του αιτήματος. Μια εφαρμογή που στέλνει αιτήματα στο API μπορεί να ελέγξει τον κωδικό κατάστασης και να εκτελέσει ενέργειες με βάση το αποτέλεσμα. Αυτές οι ενέργειες θα μπορούσαν να περιλαμβάνουν τη διαχείριση σφαλμάτων ή την εμφάνιση ενός μηνύματος επιτυχίας σε έναν χρήστη.

Ακολουθεί μια μικρή λίστα με τους πιο συνηθισμένους κωδικούς κατάστασης που επιστρέφονται από REST API:

200 OK	Η ενέργεια που ζητήθηκε ήταν επιτυχής.
400 Bad Request	Το αίτημα ήταν εσφαλμένο.
401 Unauthorized	Ο πελάτης δεν είναι εξουσιοδοτημένος να εκτελέσει την ενέργεια που ζητήθηκε.
404 Not Found	Ο πόρος που ζητήθηκε δεν βρέθηκε.
500 Internal Server Error	Ο διακομιστής παρουσίασε ένα σφάλμα κατά την επεξεργασία του αιτήματος.

API Endpoints - Τελικά σημεία API

Ένα REST API εκθέτει ένα σύνολο δημόσιων διευθύνσεων URL που χρησιμοποιούν οι εφαρμογές-πελάτες για πρόσβαση στους πόρους μιας υπηρεσίας Ιστού. Αυτές οι διευθύνσεις URL, στο πλαίσιο ενός API, ονομάζονται τελικά σημεία - Endpoints .

Για να διευκρινιστεί αυτό στον παρακάτω πίνακα, θα δείτε κάποια από τα τελικά σημεία API για το σύστημα που υλοποιήθηκε στην εργασία. Αυτά τα τελικά σημεία προορίζονται για έναν πόρο πελάτη που αντιπροσωπεύει πιθανούς πελάτες στο σύστημα:

HTTP method	API endpoint	Περιγραφή
GET	/sendmessage	Λαμβάνει τιμές των αισθητήρων
GET	/node?node=2	Για τις τιμές ενός συγκεκριμένου κόμβου
POST	/login	Για σύνδεση του χρήστη
POST	/logout	Για αποσύνδεση σύνδεση του χρήστη

3.4 Βιβλιοθήκες Python

3.4.1 Request

Η βιβλιοθήκη αιτημάτων είναι ένα από τα αναπόσπαστα μέρη της Python για την υποβολή αιτημάτων HTTP σε μια καθορισμένη διεύθυνση URL. Είτε πρόκειται για REST API είτε για Web Scrapping. Όταν κάποιος κάνει ένα αίτημα σε ένα URI, επιστρέφεται μια απάντηση. Τα αιτήματα Python παρέχουν ενσωματωμένες λειτουργίες για τη διαχείριση τόσο του αιτήματος όσο και της απόκρισης.

Η requests επιτρέπει αποστολή αιτημάτων HTTP/1.1 εξαιρετικά εύκολα. Δεν χρειάζεται να προστεθεί με μη αυτόματο τρόπο συμβολοσειρών ερωτημάτων στις διευθύνσεις URL ή να κωδικοποιήθούν τα δεδομένα POST.

Η εγκατάσταση requests εξαρτάται από τον τύπο του λειτουργικού συστήματος που χρησιμοποιούμε, η βασική εντολή οπουδήποτε θα ήταν να ανοίξουμε ένα τερματικό εντολών και να εκτελέσουμε,

pip install requests

Flask framework

Χρησιμοποιούμε το πλαίσιο Flask για την ανάπτυξη εφαρμογών Web σε γλώσσα προγραμματισμού Python. Ενσωματώνεται με άλλες υπηρεσίες τρίτων και API για να παρέχει και υποστήριξη ιστοσελίδας στην υπό ανάπτυξη εφαρμογή.

Το Flask είναι ένα πλαίσιο web που παρέχει βιβλιοθήκες για τη δημιουργία ελαφρών εφαρμογών ιστού σε python. Αναπτύχθηκε από τον Armin Ronacher που ηγείται μιας διεθνούς ομάδας λάτρεις της Python (POCCO). Βασίζεται στο κιτ εργαλείων WSGI και στη μηχανή προτύπων jinja2. Το Flask θεωρείται ως framework.

3.4.2 Βιβλιοθήκη mysql.connector

Η Python μπορεί να χρησιμοποιηθεί σε εφαρμογές βάσεων δεδομένων.

Μία από τις πιο δημοφιλείς βάσεις δεδομένων είναι η MySQL.

Η Python χρειάζεται ένα πρόγραμμα οδήγησης MySQL για πρόσβαση στη βάση δεδομένων MySQL.

Το δημοφιλέστερο είναι το: MySQL Connector

και το εγκαθιστούμε με

pip install mysql-connector-python

Στην συνέχει μπορούμε να συνδεθούμε στη βάση μας μέσω του παρακάτω κώδικα:

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword"
)
```

3.4.3 Βιβλιοθήκη json

Το JSON είναι μορφοποίηση για την αποθήκευση και την ανταλλαγή δεδομένων.

Το JSON είναι κείμενο, γραμμένο με σημειογραφία αντικειμένου JavaScript.

Η Python έχει ένα ενσωματωμένο πακέτο που ονομάζεται json, το οποίο μπορεί να χρησιμοποιηθεί για εργασία με δεδομένα JSON.

Η λειτουργική μονάδα json μπορεί να εισαχθεί ως:

import json

Μετατροπή από JSON σε Python

Εάν έχετε μια συμβολοσειρά JSON, μπορείτε να την αναλύσετε χρησιμοποιώντας τη μέθοδο json.loads().

```
import json
x =  '{ "sensor1":"23", "sensor2":"44"}'
# parse x:
y = json.loads(x)
# the result is a Python dictionary:
print(y["sensor1"])
```

Αποτέλεσμα:

23

Μετατροπή από Python σε JSON

Εάν έχετε ένα αντικείμενο Python, μπορείτε να το μετατρέψετε σε συμβολοσειρά JSON χρησιμοποιώντας τη μέθοδο json.dumps().

```
import json
# a Python object :
x = {
  "sensor1": "23",
  "sensor2": "44",
}
# convert into JSON:
```

```
y = json.dumps(x)

# the result is a JSON string:

print(y)
```

Αποτέλεσμα:

{" sensor1": "23", " sensor1": 44}

3.4.4 Βιβλιοθήκη Fernet

Η κρυπτογραφία είναι η πρακτική της διασφάλισης χρήσιμων πληροφοριών κατά τη μετάδοση από τον έναν υπολογιστή στον άλλο ή την αποθήκευση δεδομένων σε έναν υπολογιστή. Η κρυπτογραφία ασχολείται με την κρυπτογράφηση του απλού κειμένου σε κρυπτογραφημένο κείμενο και την αποκρυπτογράφηση του κρυπτογραφημένου κειμένου σε απλό κείμενο.

Η Python υποστηρίζει ένα πακέτο κρυπτογραφίας που μας βοηθά να κρυπτογραφήσουμε και να αποκρυπτογραφήσουμε δεδομένα. Η μονάδα fernet του πακέτου κρυπτογραφίας έχει ενσωματωμένες λειτουργίες για τη δημιουργία του κλειδιού, κρυπτογράφηση απλού κειμένου σε κρυπτογραφημένο κείμενο και αποκρυπτογράφηση κρυπτογραφημένου κειμένου σε απλό κείμενο χρησιμοποιώντας τις μεθόδους κρυπτογράφησης και αποκρυπτογράφησης αντίστοιχα. Η μονάδα fernet εγγυάται ότι τα δεδομένα που έχουν κρυπτογραφηθεί με τη χρήση της δεν μπορούν να χειριστούν περαιτέρω ή να διαβαστούν χωρίς το κλειδί.

Μέθοδοι που χρησιμοποιούνται:

generate_key () : Αυτή η μέθοδος δημιουργεί ένα νέο κλειδί fernet. Το κλειδί πρέπει να διατηρείται ασφαλές καθώς είναι το πιο σημαντικό στοιχείο για την αποκρυπτογράφηση του κρυπτογραφημένου κειμένου. Εάν το κλειδί χαθεί, ο χρήστης δεν μπορεί πλέον να αποκρυπτογραφήσει το μήνυμα. Επίσης, εάν ένας εισβολέας ή χάκερ αποκτήσει πρόσβαση στο κλειδί, μπορεί όχι μόνο να διαβάσει τα δεδομένα αλλά και να πλαστογραφήσει τα δεδομένα.

encrypt(data): Κρυπτογραφεί τα δεδομένα που μεταβιβάζονται ως παράμετρος στη μέθοδο. Το αποτέλεσμα αυτής της κρυπτογράφησης είναι γνωστό ως "κουπόνι Fernet" που είναι βασικά το κρυπτογραφημένο κείμενο. Το κρυπτογραφημένο διακριτικό περιέχει επίσης την τρέχουσα χρονική

σήμανση όταν δημιουργήθηκε σε απλό κείμενο. Η μέθοδος κρυπτογράφησης δημιουργεί μια εξαίρεση εάν τα δεδομένα δεν είναι σε byte.

decrypt(token): Αυτή η μέθοδος αποκρυπτογραφεί το διακριτικό Fernet που διαβιβάστηκε ως παράμετρος στη μέθοδο. Κατά την επιτυχή αποκρυπτογράφηση, λαμβάνεται ως αποτέλεσμα το αρχικό απλό κείμενο, διαφορετικά δημιουργείται μια εξαίρεση.

3.5 MySQL

Η MySQL είναι ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων που βασίζεται στη δομημένη γλώσσα ερωτημάτων, η οποία είναι η δημοφιλής γλώσσα για την πρόσβαση και τη διαχείριση των εγγραφών στη βάση δεδομένων. Το MySQL είναι λογισμικό ανοιχτού κώδικα και ελεύθερο υπό την άδεια GNU. Υποστηρίζεται από την Oracle Company.

Το σεμινάριο MySQL περιλαμβάνει όλα τα θέματα της βάσης δεδομένων MySQL που παρέχει τον τρόπο διαχείρισης της βάσης δεδομένων και χειρισμού δεδομένων με τη βοήθεια διαφόρων ερωτημάτων SQL. Αυτά τα ερωτήματα είναι: εισαγωγή εγγραφών, ενημέρωση εγγραφών, διαγραφή εγγραφών, επιλογή εγγραφών, δημιουργία πινάκων, απόθεση πινάκων κ.λπ.

Μια βάση δεδομένων είναι μια εφαρμογή που αποθηκεύει την οργανωμένη συλλογή εγγραφών. Η πρόσβαση και η διαχείριση του από τον χρήστη είναι πολύ εύκολη. Μας επιτρέπει να οργανώνουμε δεδομένα σε πίνακες, σειρές, στήλες και ευρετήρια για να βρίσκουμε τις σχετικές πληροφορίες πολύ γρήγορα. Κάθε βάση δεδομένων περιέχει ξεχωριστό API για την εκτέλεση λειτουργιών βάσης δεδομένων, όπως η δημιουργία, η διαχείριση, η πρόσβαση και η αναζήτηση των δεδομένων που αποθηκεύει.

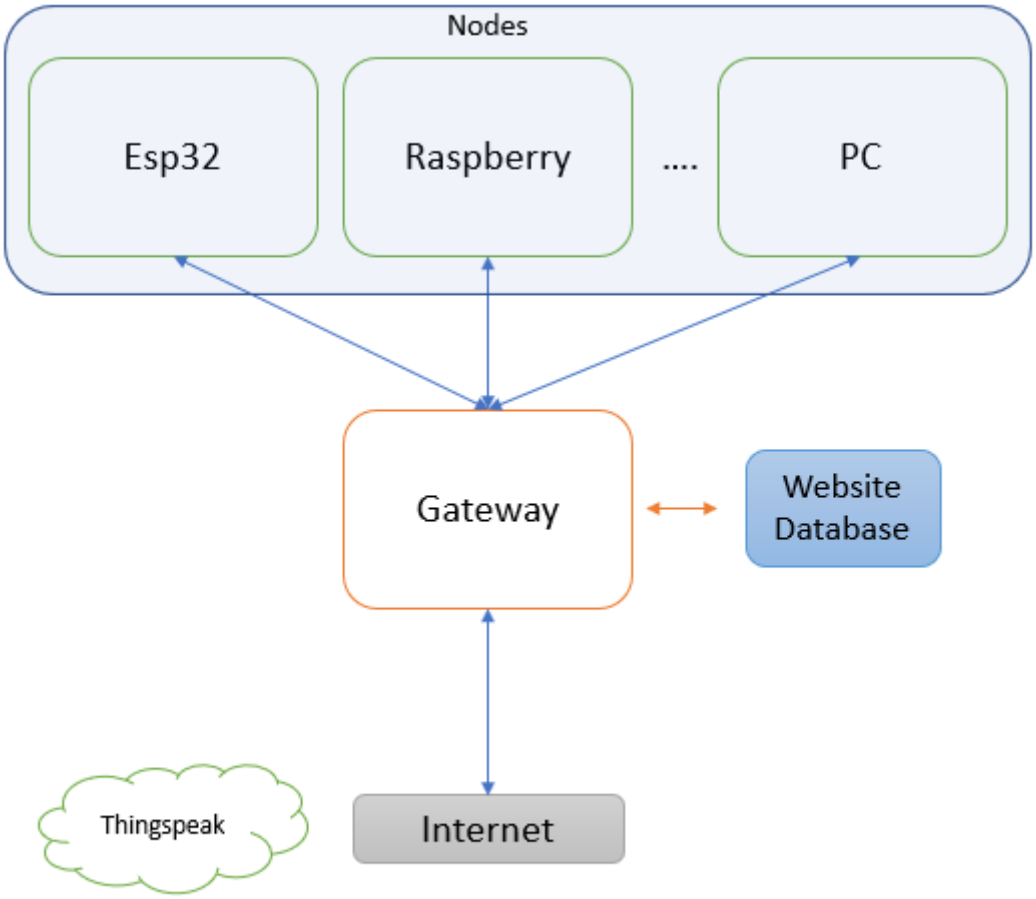
Η MySQL είναι σήμερα το πιο δημοφιλές λογισμικό συστήματος διαχείρισης βάσεων δεδομένων που χρησιμοποιείται, είναι γρήγορο, επεκτάσιμο και εύκολο στη χρήση σύστημα διαχείρισης βάσεων δεδομένων σε σύγκριση με τον Microsoft SQL Server και τη βάση δεδομένων Oracle. Χρησιμοποιείται συνήθως σε συνδυασμό με PHP ή Python για τη δημιουργία ισχυρών και δυναμικών εφαρμογών στον διακομιστή ή στο web.

Κεφάλαιο 4ο: Εφαρμογές για την ασφάλεια σε συστήματα IoT

4.1 Περιγραφή του συστήματος

Στην Εικόνα 4.1 παρουσιάζεται ένα απλό σύστημα IoT με χρήση ενός node-server που τον ονομάζουμε συχνά gateway-πύλη που εξυπηρετεί τα nodes-κόμβοι που συνδέονται μαζί του.

Οι κόμβοι μπορεί να είναι μικροελεγκτές, μικροσυστήματα ή υπολογιστές.



Εικόνα 4.1: Παράδειγμα IoT σύστημα

Η gateway συνήθως συνδέεται με το Internet για να έχει απομακρυσμένη επικοινωνία, ανταλλαγή δεδομένων και πρόσβαση.

Το gateway συνήθως είναι ένα ισχυρό υπολογιστικό σύστημα, όπως υπολογιστής ή πχ ένα raspberry.

Αυτό μπορεί να υποστηρίζει λειτουργίες server όπως εξυπηρέτηση ιστοσελίδων και βάσεων και μπορεί να επικοινωνήσει με cloud υπηρεσίες όπως Thingspeak και Amazon.

Στο κεφάλαιο αυτό θα ξεχωρίσουμε την κάθε πτυχή της ασφάλειας και θα την περιγράψουμε και μέσα από κάθε κομμάτι κώδικα με σχόλια. Ο χρήστης μπορεί να χρησιμοποιήσει το κάθε κομμάτι κώδικα στο σημείο που τον ενδιαφέρει.

4.1.1 Ασφάλεια στη μετάδοση από κόμβο σε κεντρικό-πύλη και αντίστροφα

Client 1

```
import requests


from cryptography.fernet import Fernet

import random

import string


#Fernet κλειδί

fernetkey = b'vyY29vFL_KTVWC3FpCkgdOo5G2cFiqn4_svU79oBoI8=';


#Υπάρχει η συνάρτηση της Fernet που μπορεί να παράγει ενα κλειδί για εμάς

#Fernet.generate_key()


#Το κάθε node έχει ενα μοναδικό id

myid = 'a3b4c62b'


#Κάθε server έχει ενα μοναδικό id

serverid = 'd454ba3e'


#Το μήνυμα που θα αποστείλλει περιέχει

#το id της πηγής

#το id του προορισμού

#τις τιμές των αισθητήρων μέσω ενός ειδικού format

msg = myid+'#'+serverid+'#sensor1=21.3#sensor2=47.3#sensor3=-87.3'
```

```
f = Fernet(fernetkey)


#Κωδικοποιούμε το μήνυμα με το κλειδί του Fernet
ciphertext = f.encrypt(msg.encode())
#print(ciphertext)
#Το κωδικοποιημένο μήνυμα είναι σε b" μορφή
#b'gAAAAABjwYmWYR-mtS3hPaUThwfekGtyVsygjuH6-
m1EMUpSVwZwu260xlfBcLgkI8hpOFRUgFLKFgM1Yatq8-D1sat9qc6sNuI-
D14r90jHSaBbHWMvTbbjwJLyRW8hqTZes_L553sXDeKak4NvfbxZbYNT_m4P8g=='


#Μετατρέπει το κωδικοποιημένο μήνυμα σε Text (string) μορφή
message= ciphertext.decode()
#print(message)
#gAAAAABjwYmWYR-mtS3hPaUThwfekGtyVsygjuH6-
m1EMUpSVwZwu260xlfBcLgkI8hpOFRUgFLKFgM1Yatq8-D1sat9qc6sNuI-
D14r90jHSaBbHWMvTbbjwJLyRW8hqTZes_L553sXDeKak4NvfbxZbYNT_m4P8g==


#Στέλνουμε το μήνυμα με όλες τις πληροφορίες στον Server σε συγκεκριμένο URL
result = requests.get("http://localhost:4000/ sendmessage/?mid="+myid+"&sid="+serverid+"&msg="+ message)


#Λαμβάνουμε την απάντηση και ελέγχουμε αν ο server μας απάντησε εκτελώντας σωστά την λειτουργία
if result.status_code == 200:
    print(result.text)
else:
    print(result.text)
```

Server

```
from flask import Flask
from flask import request,jsonify
```

```
import json


from cryptography.fernet import Fernet
import random
import string


#Fernet κλειδί
#Κόμβοι και Server χρησιμοποιούν το ίδιο κλειδί
key = b'vyY29vFL_KTVWC3FpCkgdOo5G2cFiqn4_svU79oBoI8=';


node_list = ['a3b4c62b','c3b6c69b','e3b2a32a']


app = Flask(__name__)


#rest endpoint (restfull api)
@app.route('/', methods=['GET', 'POST'])
def main():
    return "Main"


@app.route('/sendmessage/', methods=['GET'])
def sendmessage():
    #Αν το εισερχόμενο μήνυμα αφορά  αυτόν τον Server
    if request.args.get("sid") == "d454ba3e":


        #Αν στο εισερχόμενο μήνυμα περιέχει id του κόμβου που είναι εγγεγραμμένος για αυτόν τον Server
        if request.args.get("mid") in node_list:
            #Παίρνουμε το μήνυμα από το URL
            msg = request.args.get("msg")
            try:
                msg = msg.encode()
```

```
        f = Fernet(key)
        cleartext = f.decrypt(msg)
        #b'a3b4c62b#d454ba3e#sensor1=21.3#sensor2=47.3#sensor3=-87.3'
        #print(cleartext)
        message = cleartext.decode()
        #a3b4c62b#d454ba3e#sensor1=21.3#sensor2=47.3#sensor3=-87.3
        #print(cleartext)

        #Διαχωρίζουμε το αποκρυπτογραφημένο μήνυμα με το #
        messageArray = message.split("#")

        sid = messageArray[0]
        mid = messageArray[1]
        sensor1 = messageArray[2] #πχ θα περιέχει το sensor1=21.3
        sensor2 = messageArray[3]
        sensor3 = messageArray[4]

        sensor1Value = sensor1.split("=")[1] #πχ θα περιέχει το 21.3
        sensor2Value = sensor2.split("=")[1]
        sensor3Value = sensor3.split("=")[1]

        #print(sensor1Value)
        #print(sensor2Value)
        #print(sensor3Value)

        return "ok"
    except:
        print("Invalid decryption")
        return "Invalid decryption"
else:
```

```
            print("Wrong client")
            return "Wrong client"
    else:
        print("Wrong server")
        return "Wrong server"

    return "0"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port='4000')
```

4.1.2 Ασφάλεια στη αυθεντικοποίηση κόμβου από κεντρικό-πύλη

Είδαμε προηγουμένως ότι τα id των κόμβων είναι αποθηκευμένα στον server όπως

node_list = ['a3b4c62b','c3b6c69b','e3b2a32a']

Στην πραγματικότητα θα πρέπει οι κόμβοι να είναι αποθηκευμένοι σε μια βάση και να γίνεται έλεγχος σε αυτήν.

Αρχικά θα δούμε μέσω ενός παραδείγματος τον κώδικα που χρειάζεται ο server και ο client για την αποθήκευση τιμών στη βάση και πως μπορώ να έχω πρόσβαση σε αυτήν.

Server DB

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

app = Flask(__name__)

@app.route('/addvalue/', methods=['GET'])
```

```
def addvalue():
    #http://127.0.0.1:4000/addvalue?device=A123&field1=10&field2=96
    device = request.args.get("device")
    field1 = request.args.get("field1")
    field2 = request.args.get("field2")

    print(field1)

    mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="iot"
    )

    mycursor = mydb.cursor()

    sql = "INSERT INTO sensors (device, field1, field2) VALUES (%s, %s, %s)"
    val = (device, field1, field2)
    mycursor.execute(sql, val)
    mydb.commit()

    print(mycursor.lastrowid)

    return "ok"

@app.route('/getvalues/', methods=['GET'])
def getvalues():
```

```
    #http://127.0.0.1:4000/getvalues
    mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="iot"
    )

    mycursor = mydb.cursor()

    mycursor.execute("SELECT * FROM sensors")
    myresult = mycursor.fetchall()
    for x in myresult:
        print(x)

    return "ok"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port='4000')
```

Εδώ μπορούμε να καλέσουμε τον server με δύο τουλάχιστον τρόπους

Client DB

1.

Από ένα απλό URL στον περιηγητή

http://127.0.0.1:4000/addvalue?device=A123&field1=10&field2=96

ή

http://127.0.0.1:4000/getvalues

2.

με python

import requests

result = requests.get("http://127.0.0.1:4000/addvalue?device=A123&field1=10&field2=96")

ή

result = requests.get("http://127.0.0.1:4000/getvalues")

4.1.3 Ασφάλεια στη αυθεντικοποίηση χρήστη από κεντρικό-πύλη

Κάθε κόμβος έχει το δικό του id και ανήκει σε κάποιον χρήστη. Αυτό μπορεί να συμβαίνει και για κάποιον server-gateway.

Αυτό μπορεί να οριστεί και να ελέγχεται μέσω της βάσης.

	id	active	node	gateway	userid	kind 0:gateway,1:node
Επεξεργασία Αντιγραφή Διαγραφή	1	1	s120	NULL	1	0
Επεξεργασία Αντιγραφή Διαγραφή	2	1	b3002	s120	1	1
Επεξεργασία Αντιγραφή Διαγραφή	3	1	a2001	s120	1	1
Επεξεργασία Αντιγραφή Διαγραφή	4	1	a2002	s120	1	1
Επεξεργασία Αντιγραφή Διαγραφή	5	1	p130	NULL	2	0
Επεξεργασία Αντιγραφή Διαγραφή	6	1	b4002	p130	2	1

Εικόνα 4.2: Πίνακας στη βάση για αντιστοίχιση χρήστη και κόμβου

Αποδοχή των τιμών στον Server από τους Κόμβους

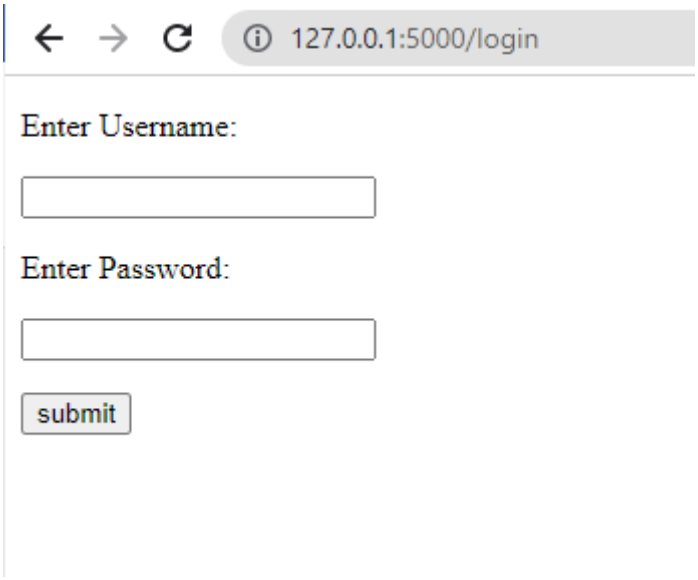
```
@app.route('/sendmessage/', methods=['GET'])
def sendmessage():
    # Στον server μου βάζω userid μου
    myuserid = 1

    gateway = request.args.get("sid")
    node = request.args.get("mid")

    #Αν το εισερχόμενο μήνυμα αφορά  αυτόν τον Server
    if gateway == "s120":
        if node <> None:
            mydb = mysql.connector.connect(
            host="localhost",
            user="root",
            password="",
            database="iotsystem"           )
            mycursor = mydb.cursor()

            #Αν στο εισερχόμενο μήνυμα περιέχει id του κόμβου που είναι εγγεγραμμένος για αυτόν τον Χρήστη
            mycursor.execute("SELECT * FROM nodes WHERE active = 1 AND userid="+myuserid+" AND node='"+node+"'")
            myresult = mycursor.fetchall()
            if len(myresult) >0:
                nodeid = myresult[0][0]

                #Παίρνουμε το μήνυμα από το URL
                msg = request.args.get("msg")
                try:
                    msg = msg.encode()
                    f = Fernet(key)
```

```
cleartext = f.decrypt(msg)                  #b'b3002#s120#sensor1=21.3#sensor2=47.3'

message = cleartext.decode()                #b3002#s120#sensor1=21.3#sensor2=47.3


#Διαχωρίζουμε το αποκρυπτογραφημένο μήνυμα με το #

messageArray = message.split("#")

sid = messageArray[0]

mid = messageArray[1]

sensor1 = messageArray[2] #πχ θα περιέχει το sensor1=21.3

sensor2 = messageArray[3]

sensor1Value = sensor1.split("=")[1] #πχ θα περιέχει το 21.3

sensor2Value = sensor2.split("=")[1]


#Αποθήκευσε τη τιμή στη βάση

#INSERT

if sensor1 <> None:

   sql = "INSERT INTO sensors (value,nodeid, field) VALUES (%s, %s, %s)"

   val = (sensor1Value, nodeid, "1")

   mycursor.execute(sql, val)

   mydb.commit()


if sensor1 <> None:

   sql = "INSERT INTO sensors (value,nodeid, field) VALUES (%s, %s, %s)"

   val = (sensor2Value, nodeid, "2")

   mycursor.execute(sql, val)

   mydb.commit()


    ….
```

4.1.4 Ασφάλεια στη πρόσβαση μέσω διαδικτύου με το κεντρικό-πύλη

Μπορούμε να υλοποιήσουμε στο server μια ιστοσελίδα που να μπορεί ο χρήστης να την επισκέπτεται τοπικά ή διαδικτυακά και να βλέπει τους κόμβους για κάθε server αλλά και τις μετρήσεις του που λαμβάνει από αυτούς.

Υπάρχει όμως μια διαδικασία ταυτοποίησης του κάθε χρήστη μέσω της βάσης.



Εικόνα 4.3: Σελίδα για την είσοδο του χρήστη στον server

```
<!DOCTYPE html>
<html>
<head>
   <title>Login Page</title>
</head>
<body>
   <form action="/login" method="POST">
      <p>Enter Username:</p>
      <p><input type="text" name="username" /></p>
      <p>Enter Password:</p>
      <p><input type="password" name="password" /></p>
```

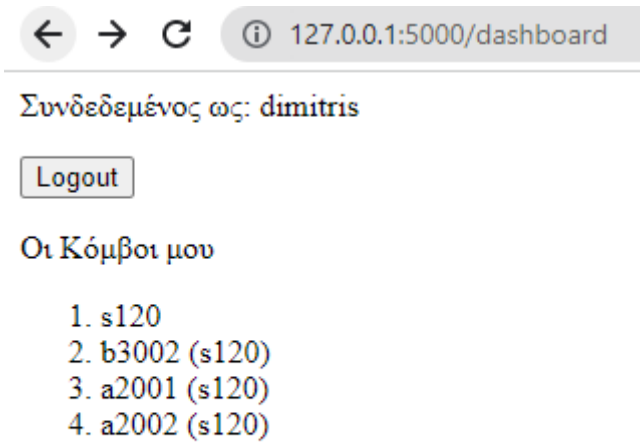
```
        <p><input type="submit" value="submit" /></p>
    </form>
</body>
</html>
```

```
@app.route('/logout', methods = ['POST', 'GET'])
def logout():
    if('username' in session):
        session.pop('username')
    if('userid' in session):
        session.pop('userid')
    return redirect('/login')
```

Από την πλευρά της Ιστοσελίδας, όταν ο χρήστης είναι συνδεδεμένος

```
@app.route('/dashboard')
def dashboard():
    if('username' in session):
        username=session['username']
        userid=session['userid']
        nodes=[]

        mydb = mysql.connector.connect(
        host="localhost",
        user="root",
        password="",
        database="iotsystem"
        )

        mycursor = mydb.cursor()
```

```
    mycursor.execute("SELECT *, CONCAT(node,'  (',gateway,')') as nodewithg FROM nodes WHERE active = 1 AND userid='"+userid+"'")
    myresult = mycursor.fetchall()

    if len(myresult) >0:
       nodes = myresult
       #print(nodes[0][2])

    return render_template("dashboard.html",username=username,nodes=nodes,lenNodes = len(nodes))

  return '<h1>You are not logged in.</h1>'
```

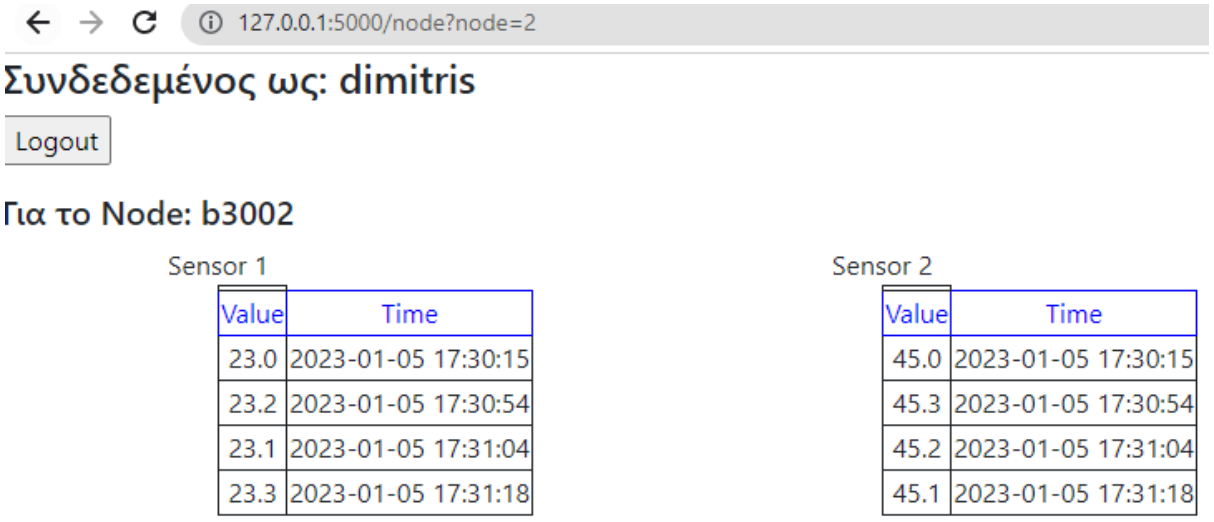


Εικόνα 4.4: Σελίδα με τους κόμβους ενός χρήστη

Στην Εικόνα 4.3 φαίνεται ποιοι κόμβοι ανήκουν στον χρήστη και ποιοι είναι οι κεντρικοί (πχ s120) και ποιοι ανήκουν σε κεντρικούς (πχ b3002).

```
@app.route('/dashboard')
def dashboard():
   if('username' in session):
```

```
        username=session['username']
        userid=session['userid']
        nodes=[]

        mydb = mysql.connector.connect(
        host="localhost",
        user="root",
        password="",
        database="iotsystem"
        )

        mycursor = mydb.cursor()

        mycursor.execute("SELECT *, CONCAT(node,'  (',gateway,')') as nodewithg FROM nodes WHERE active = 1 AND
userid='"+userid+"'")
        myresult = mycursor.fetchall()

        if len(myresult) >0:
            nodes = myresult
            #print(nodes[0][2])

        return render_template("dashboard.html",username=username,nodes=nodes,lenNodes = len(nodes))

    return '<h1>You are not logged in.</h1>'
```



Εικόνα 4.5: Σελίδα που προβάλλονται οι τιμές αισθητήρων ενός κόμβου

```
@app.route('/node')
def node():
    if('username' in session):
        username=session['username']
        userid=session['userid']

        nodeid = request.args.get("node")
        nodename=""

        field1=[]
        field2=[]
        field3=[]
        field4=[]

        lenField1=0
        lenField2=0
        lenField3=0
        lenField4=0
```

```
    mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="iotsystem"
    )

    mycursor = mydb.cursor()

    mycursor.execute("SELECT * FROM nodes WHERE active = 1 AND id="+nodeid)
    myresult = mycursor.fetchall()
    if len(myresult) >0:
       nodename = myresult[0][2]

    mycursor.execute("SELECT * FROM sensors WHERE active = 1 AND nodeid="+nodeid+" AND field=1")
    myresult1 = mycursor.fetchall()
    if len(myresult1) >0:
       field1 = myresult1

    mycursor.execute("SELECT * FROM sensors WHERE active = 1 AND nodeid="+nodeid+" AND field=2")
    myresult2 = mycursor.fetchall()
    if len(myresult2) >0:
       field2 = myresult2

    return render_template("values.html",username=username,nodename=nodename,field1=field1,lenField1=
len(field1),field2=field2,lenField2= len(field2))

    #mycursor.execute("SELECT * FROM sensors WHERE active = 1 AND nodeid='"+nodeid+"'")
    #myresult = mycursor.fetchall()
    #if len(myresult) >0:
       #values = myresult

    #return render_template("values.html",username=username,values=values,lenValues= len(values))
```

```
return '<h1>You are not logged in.</h1>'
```

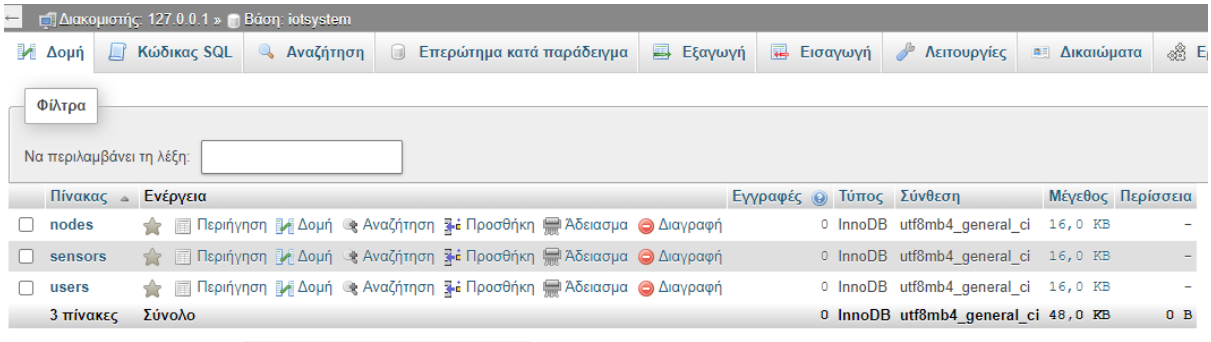
4.1.5 Αξιοπιστία στη μέτρηση των αισθητήρων

Κατά τη μέτρηση ενός αισθητήρα πρέπει να βελτιώσουμε την αξιοπιστία της μέτρησης ως προς την τιμή της. Μπορούμε να εφαρμόσουμε ειδικές τεχνικές από την μεριά του κόμβου ή από την μεριά του server.

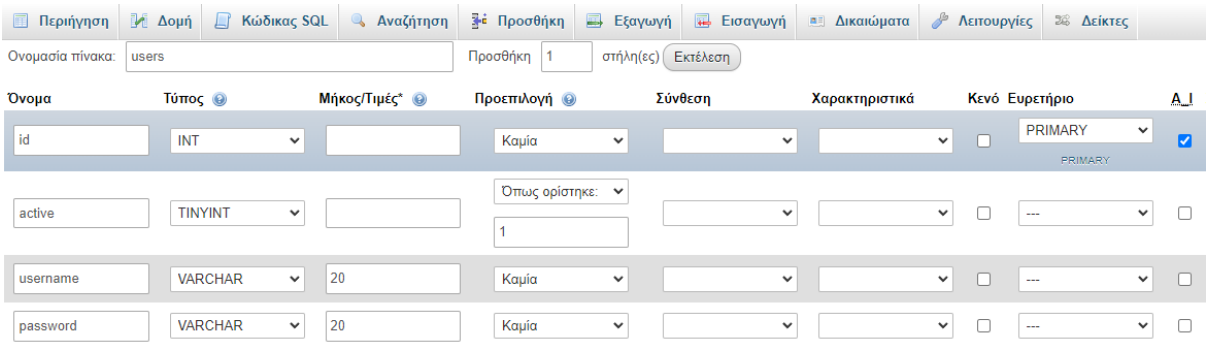
Θα επιλέξουμε να ελέγξουμε τις μετρήσεις στον server γιατί περιέχει ικανό υπολογιστικό σύστημα για έλεγχο αλλά και για να εξακριβώσει τι συμβαίνει με τον αισθητήρα.

Χρησιμοποιήθηκε μέθοδος Ελαχίστων Τετραγώνων για τον υπολογισμό των τελευταίων 5 τιμών σε μικρό χρονικό διάστημα (όταν υπάρχει η δυνατότητα) και ο έλεγχος αν η επόμενη τιμή βρίσκεται μέσα στα όρια με απόκλιση 20% της προβλεπόμενης τιμής.

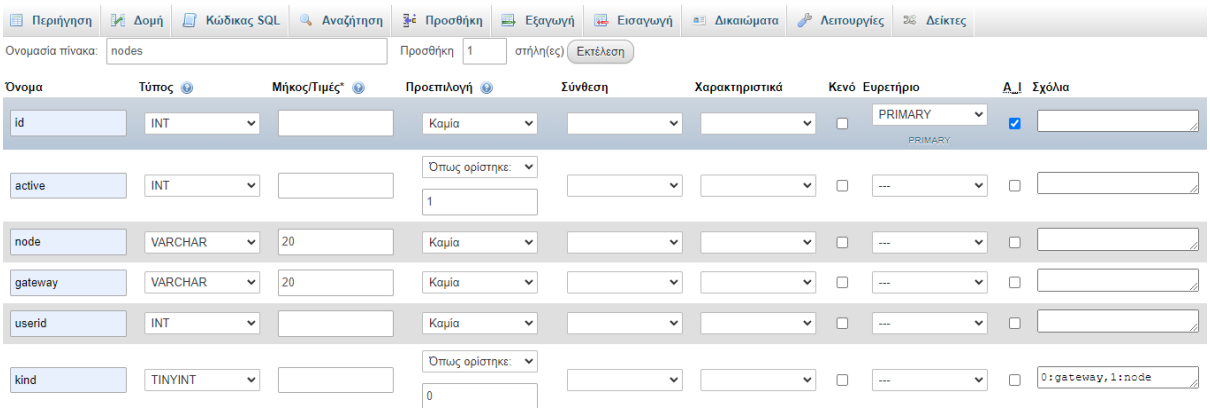
4.2 Περιγραφή της βάσης



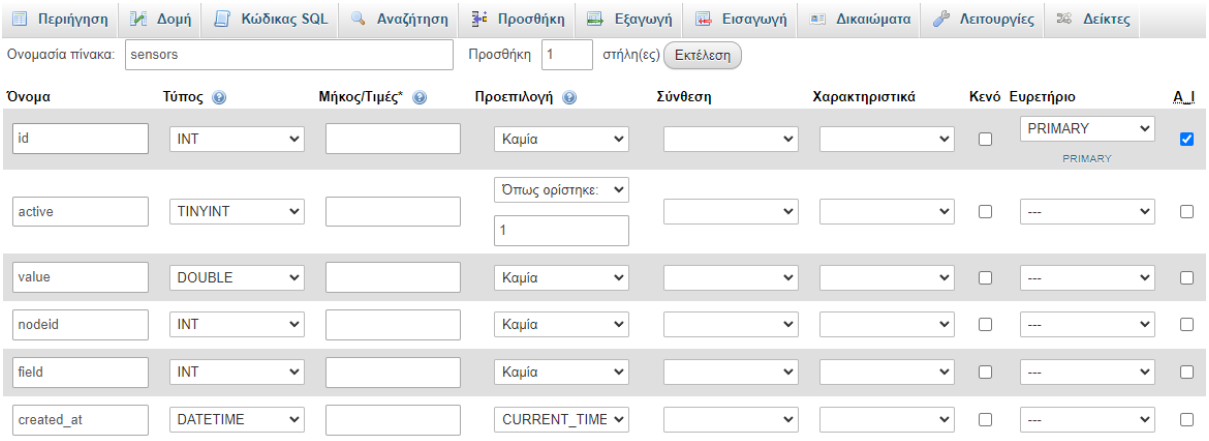
Εικόνα 4.6: Η δομή της βάσης iotsystem



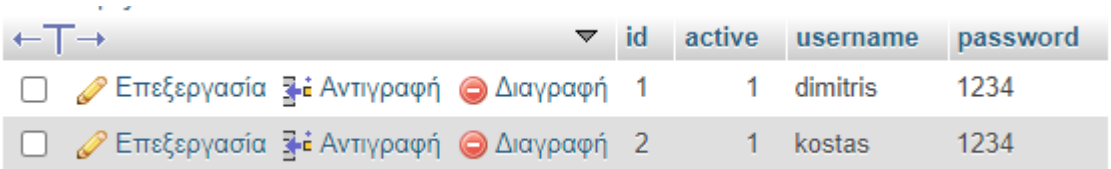
Εικόνα 4.7: Η δομή του πίνακα user



Εικόνα 4.8: Η δομή του πίνακα nodes



Εικόνα 4.9: Η δομή του πίνακα sensors



Εικόνα 4.10: Πίνακας user

←T→			id	active	node	gateway	userid	kind 0:gateway,1:node
☐	Επεξεργασία	Αντιγραφή Διαγραφή	1	1	s120	*NULL*	1	0
☐	Επεξεργασία	Αντιγραφή Διαγραφή	2	1	b3002	s120	1	1
☐	Επεξεργασία	Αντιγραφή Διαγραφή	3	1	a2001	s120	1	1
☐	Επεξεργασία	Αντιγραφή Διαγραφή	4	1	a2002	s120	1	1
☐	Επεξεργασία	Αντιγραφή Διαγραφή	5	1	p130	*NULL*	2	0
☐	Επεξεργασία	Αντιγραφή Διαγραφή	6	1	b4002	p130	2	1

Εικόνα 4.11: Πίνακας nodes

+ Επιλογές

←T→			id	active	value	nodeid	field	created_at
☐	Επεξεργασία	Αντιγραφή Διαγραφή	1	1	23	2	1	2023-01-05 17:30:15.
☐	Επεξεργασία	Αντιγραφή Διαγραφή	2	1	45	2	2	2023-01-05 17:30:15.
☐	Επεξεργασία	Αντιγραφή Διαγραφή	3	1	23.2	2	1	2023-01-05 17:30:54.
☐	Επεξεργασία	Αντιγραφή Διαγραφή	4	1	45.3	2	2	2023-01-05 17:30:54.
☐	Επεξεργασία	Αντιγραφή Διαγραφή	5	1	23.1	2	1	2023-01-05 17:31:04.
☐	Επεξεργασία	Αντιγραφή Διαγραφή	6	1	45.2	2	2	2023-01-05 17:31:04.
☐	Επεξεργασία	Αντιγραφή Διαγραφή	7	1	23.3	2	1	2023-01-05 17:31:18.
☐	Επεξεργασία	Αντιγραφή Διαγραφή	8	1	45.1	2	2	2023-01-05 17:31:18.

Εικόνα 4.12: Πίνακας sensors

Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης

Παρουσιάστηκε η μελέτη συστημάτων ασφάλειας στο χώρο των IoT συστημάτων και τι σημαίνει αυτό. Στη συνέχεια παρουσιάστηκαν οι κώδικες για κάθε κομμάτι ασφάλειας που απαιτείται από ένα σύστημα IoT όπως το σχεδιάσαμε.

Κατασκευάσαμε μια απλή τοπολογία με κεντρικούς και περιφερειακούς κόμβους να συνδέονται σε αυτόν. Οι κεντρικοί κόμβοι είναι server με σύνδεση στο διαδίκτυο και είναι server βάσης δεδομένων και ιστοσελίδων.

Μπορούμε να βλέπουμε τα δεδομένα μέσω ιστοσελίδας αφού συνδεθούμε σε αυτή και οι κόμβοι μας είναι κατοχυρωμένοι στην βάση και ανήκουν σε ένα από τους κεντρικούς κόμβους που έχουμε δηλώσει στη βάση.

Υλοποιήθηκαν με κώδικα

στη μέτρηση των αισθητήρων από τον κόμβο

στη μετάδοση δεδομένων από κόμβο σε κεντρικό gateway

στη αυθεντικοποίηση του χρήστη και αντιστοίχηση κόμβων με αυτόν

στη αυθεντικοποίηση του χρήστη στην πρόσβαση του στο κεντρικό κόμβο – gateway

Σαν βελτίωση προτείνεται η χρήση και δοκιμή των esp32 με micropython συνδεδεμένων στο δίκτυο.

Επίσης προτείνεται η παραμετροποίηση του συστήματος ή αλλιώς να γίνει πιο δυναμικό ώστε να προβάλει παραπάνω αισθητήρες ανά κόμβο και να τους εμφανίζει σε γραφήματα. Ακόμη, θα μπορούσε να γίνει ανάλυση δεδομένων.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- Khalid, L.F. and Ameen, S.Y., 2021. Secure Iot integration in daily lives: A review. *Journal of Information Technology and Informatics*, *1*(1), pp.6-12.
- Chakrabarty, S. and Engels, D.W., 2016, January. A secure IoT architecture for smart cities. In *2016 13th IEEE annual consumer communications & networking conference (CCNC)* (pp. 812-813). IEEE.
- Wang, T., Zhang, G., Liu, A., Bhuiyan, M.Z.A. and Jin, Q., 2018. A secure IoT service architecture with an efficient balance dynamics based on cloud and edge computing. *IEEE Internet of Things Journal*, *6*(3), pp.4831-4843.
- Canedo, J. and Skjellum, A., 2016, December. Using machine learning to secure IoT systems. In *2016 14th annual conference on privacy, security and trust (PST)* (pp. 219-222). IEEE.
- Dhanda, S.S., Singh, B. and Jindal, P., 2020. Lightweight cryptography: a solution to secure IoT. *Wireless Personal Communications*, *112*(3), pp.1947-1980.
- Yeh, K.H., 2016. A secure IoT-based healthcare system with body sensor networks. *IEEE access*, *4*, pp.10288-10299.
- Gope, P., Gheraibia, Y., Kabir, S. and Sikdar, B., 2020. A secure IoT-based modern healthcare system with fault-tolerant decision making process. *IEEE Journal of Biomedical and Health Informatics*, *25*(3), pp.862-873.
- Santoso, F.K. and Vun, N.C., 2015, June. Securing IoT for smart home system. In *2015 international symposium on consumer electronics (ISCE)* (pp. 1-2). IEEE.
- Haseeb, K., Ud Din, I., Almogren, A. and Islam, N., 2020. An energy efficient and secure IoT-based WSN framework: An application to smart agriculture. *Sensors*, *20*(7), p.2081.
- Pirbhulal, S., Zhang, H., E Alahi, M.E., Ghayvat, H., Mukhopadhyay, S.C., Zhang, Y.T. and Wu, W., 2016. A novel secure IoT-based smart home automation system using a wireless sensor network. *Sensors*, *17*(1), p.69.
- Pirbhulal, S., Zhang, H., E Alahi, M.E., Ghayvat, H., Mukhopadhyay, S.C., Zhang, Y.T. and Wu, W., 2016. A novel secure IoT-based smart home automation system using a wireless sensor network. *Sensors*, *17*(1), p.69.
- Kurniawan, A., 2019. *Internet of Things Projects with ESP32: Build exciting and powerful IoT projects using the all-new Espressif ESP32*. Packt Publishing Ltd.
- Babiuch, M., Foltýnek, P. and Smutný, P., 2019, May. Using the ESP32 microcontroller for data processing. In *2019 20th International Carpathian Control Conference (ICCC)* (pp. 1-6). IEEE.
- Carducci, C.G.C., Monti, A., Schraven, M.H., Schumacher, M. and Mueller, D., 2019, June. Enabling ESP32-based IoT applications in building automation systems. In *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4. 0&IoT)* (pp. 306-311). IEEE.
- Sarjerao, B.S. and Prakasarao, A., 2018, April. A low cost smart pollution measurement system using REST API and ESP32. In *2018 3rd International Conference for Convergence in Technology (I2CT)* (pp. 1-5). IEEE.

- Foltýnek, P., Babiuch, M. and Šuránek, P., 2019. Measurement and data processing from Internet of Things modules by dual-core application using ESP32 board. *Measurement and Control*, *52*(7-8), pp.970-984.
- David, V., Ragu, H., Nikhil, V. and Sasikumar, P., 2022. Analysis of Algorithms for Effective Cryptography for Enhancement of IoT Security. In *Smart Trends in Computing and Communications: Proceedings of SmartCom 2022* (pp. 91-99). Singapore: Springer Nature Singapore.
- Bray, S.W., 2020. *Implementing Cryptography Using Python*. John Wiley & Sons.
- David, V., Ragu, H., Nikhil, V. and Sasikumar, P., 2022. Analysis of Algorithms for Effective Cryptography for Enhancement of IoT Security. In *Smart Trends in Computing and Communications: Proceedings of SmartCom 2022* (pp. 91-99). Singapore: Springer Nature Singapore.
- https://cryptography.io/en/latest/fernet/
- https://www.kellton.com/kellton-tech-blog/your-complete-guide-to-iot-protocols-and-Standards-2022

ΠΑΡΑΡΤΗΜΑ Α

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

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title> IoT </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

  <style>

table {
  border-collapse: collapse;
  width: 100%;
}

td, th {
  text-align: center;
}
</style>
</head>

<body>

<div class="container">
  </br>
  <h2>Κόμβος</h2>
  <p>{{title}}</p>
  <table class="table">
    <thead>
```

```
    <tr>
      <th>Θερμοκρασία</th>
      <th>Υγρασία</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>{{field1}}</td>
      <td>{{field2}}</td>
    </tr>
  </tbody>
 </table>

</div>

</body>
</html>
```

Login.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Login Page</title>
</head>
<body>
  <form action="/login" method="POST">
    <p>Enter Username:</p>
    <p><input type="text" name="username" /></p>
    <p>Enter Password:</p>
    <p><input type="password" name="password" /></p>

    <p><input type="submit" value="submit" /></p>
  </form>
</body>
</html>
```

Dashboard.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Dash Page</title>
</head>
<body>
Συνδεδεμένος ως: {{username}}

   <form action="/logout" method="POST">
      <p><input type="submit" value="Logout" /></p>
   </form>

            <p> Οι Κόμβοι μου</p>

            <ol>
            {%for i in range(0, lenNodes)%}
            <li>{% if nodes[i][6] == None %} {{nodes[i][2]}} {% else %} {{nodes[i][6]}} {% endif %}</li>
            {%endfor%}

            </ol>

</body>
</html>
```

Values.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title> IoT Node</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

  <style>
```

```
.seira
{



margin-left: 60px;
}

table {


}

td, th {
  text-align: center;
  border: 1px solid;
}
</style>
</head>

<body>
        <h4> Συνδεδεμένος ως: {{username}} </h4>

    <form action="/logout" method="POST">
        <p><input type="submit" value="Logout" /></p>
    </form>

        <h5> Για το Node: {{nodename}}</h5>
<div style="width: 100%; overflow: hidden;">

    {% if lenField1>0 %}
        <div style="float:left; width: 400px; margin-left: 100px;">
        <span style="margin-right: 50px;">Sensor 1 </span>
        <table style="margin-left: 30px;">
            <th>
                <tr style="color: blue; ">
                <td>Value</td>
                <td>Time</td>
                </tr>
                </th>
```

```
        {%for i in range(0, lenField1)%}
        <tr>
        <td>{{field1[i][2]}}</td>
        <td>{{field1[i][5]}}</td>
        {%endfor%}
        </tr>

        </table>
        </div>
        {% endif %}

         {% if lenField2>0 %}
        <div style="float:left;  width: 400px; ">
        <span style="margin-right: 50px;">Sensor 2 </span>
        <table style="margin-left: 30px;">
                <th>
                        <tr style="color: blue; ">
                        <td>Value</td>
                        <td>Time</td>
                        </tr>
                        </th>
        {%for i in range(0, lenField2)%}
        <tr>
        <td>{{field2[i][2]}}</td>
        <td>{{field2[i][5]}}</td>
        {%endfor%}
        </tr>

        </table>
        </div>
        {% endif %}
</div>

</body>
</html>
```

Appdb.py

```
#Step – 1(import necessary library)
from flask import (Flask, render_template, request, redirect, session)
import mysql.connector

from flask import jsonify
import json

from cryptography.fernet import Fernet
import random
import string

#Fernet κλειδί
#Κόμβοι και Server χρησιμοποιούν το ίδιο κλειδί
key = b'vyY29vFL_KTVWC3FpCkgdOo5G2cFiqn4_svU79oBoI8=';



app = Flask(__name__)
app.secret_key = "iotamperiadis"

@app.route('/', methods = ['GET'])
def index():
   if('username' in session):
      username=session['username']
      return render_template("dashboard.html",username=username)
   else:
      return redirect('/login')

   return '<h1>You are not logged in.</h1>'

#Step – 4 (creating route for login)
@app.route('/login', methods = ['POST', 'GET'])
def login():
   if(request.method == 'POST'):
      username = request.form.get('username')
      password = request.form.get('password')



      mydb = mysql.connector.connect(
```

```
        host="localhost",
        user="root",
        password="",
        database="iotsystem"
        )

        mycursor = mydb.cursor()

        mycursor.execute("SELECT * FROM users WHERE active = 1 AND username='"+username+"' AND password='"+password+"'")
        myresult = mycursor.fetchall()

        if len(myresult) >0:
            print(myresult[0])
            session['username'] = username
            session['userid'] = str(myresult[0][0])
            return redirect('/dashboard')

        return "<h1>Wrong username or password</h1>"

    return render_template("login.html")

@app.route('/node')
def node():
    if('username' in session):
        username=session['username']
        userid=session['userid']

        nodeid = request.args.get("node")
        nodename=""

        field1=[]
        field2=[]
        field3=[]
        field4=[]

        lenField1=0
        lenField2=0
```

```
    lenField3=0
    lenField4=0

    mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="iotsystem"
    )

    mycursor = mydb.cursor()

    mycursor.execute("SELECT * FROM nodes WHERE active = 1 AND id="+nodeid)
    myresult = mycursor.fetchall()
    if len(myresult) >0:
       nodename = myresult[0][2]

    mycursor.execute("SELECT * FROM sensors WHERE active = 1 AND nodeid="+nodeid+" AND field=1")
    myresult1 = mycursor.fetchall()
    if len(myresult1) >0:
       field1 = myresult1

    mycursor.execute("SELECT * FROM sensors WHERE active = 1 AND nodeid="+nodeid+" AND field=2")
    myresult2 = mycursor.fetchall()
    if len(myresult2) >0:
       field2 = myresult2

    return render_template("values.html",username=username,nodename=nodename,field1=field1,lenField1=
len(field1),field2=field2,lenField2= len(field2))

    #mycursor.execute("SELECT * FROM sensors WHERE active = 1 AND nodeid='"+nodeid+"'")
    #myresult = mycursor.fetchall()
    #if len(myresult) >0:
       #values = myresult

    #return render_template("values.html",username=username,values=values,lenValues= len(values))

  return '<h1>You are not logged in.</h1>'
```

```
#Step -5(creating route for dashboard and logout)
@app.route('/dashboard')
def dashboard():
    if('username' in session):
        username=session['username']
        userid=session['userid']
        nodes=[]

        mydb = mysql.connector.connect(
        host="localhost",
        user="root",
        password="",
        database="iotsystem"
        )

        mycursor = mydb.cursor()

        mycursor.execute("SELECT *, CONCAT(node,'  (',gateway,')') as nodewithg FROM nodes WHERE active = 1 AND userid='"+userid+"'")
        myresult = mycursor.fetchall()

        if len(myresult) >0:
            nodes = myresult
            #print(nodes[0][2])

        return render_template("dashboard.html",username=username,nodes=nodes,lenNodes = len(nodes))

    return '<h1>You are not logged in.</h1>'

#Step -6(creating route for logging out)
@app.route('/logout', methods = ['POST', 'GET'])
def logout():
    if('username' in session):
        session.pop('username')
    if('userid' in session):
        session.pop('userid')
    return redirect('/login')
```

```
@app.route('/sendmessage/', methods=['GET'])
def sendmessage():

    # Στον server μου βάζω userid μου
    myuserid = 1

    gateway = request.args.get("sid")
    node = request.args.get("mid")

    #Αν το εισερχόμενο μήνυμα αφορά  αυτόν τον Server
    if gateway == "s120":


        if node <> None:

            mydb = mysql.connector.connect(
            host="localhost",
            user="root",
            password="",
            database="iotsystem"
            )

            mycursor = mydb.cursor()

            #Αν στο εισερχόμενο μήνυμα περιέχει id του κόμβου που είναι εγγεγραμμένος για αυτόν τον Χρήστη
            mycursor.execute("SELECT * FROM nodes WHERE active = 1 AND userid="+myuserid+" AND node='"+node+"'")
            myresult = mycursor.fetchall()

            if len(myresult) >0:
                nodeid = myresult[0][0]

                #Παίρνουμε το μήνυμα από το URL
                msg = request.args.get("msg")
                try:
                    msg = msg.encode()
                    f = Fernet(key)
```

```
    cleartext = f.decrypt(msg)
    #b'b3002#s120#sensor1=21.3#sensor2=47.3'
    #print(cleartext)
    message = cleartext.decode()
    #b3002#s120#sensor1=21.3#sensor2=47.3
    #print(cleartext)

    #Διαχωρίζουμε το αποκρυπτογραφημένο μήνυμα με το #
    messageArray = message.split("#")

    sid = messageArray[0]
    mid = messageArray[1]
    sensor1 = messageArray[2] #πχ θα περιέχει το sensor1=21.3
    sensor2 = messageArray[3]

    sensor1Value = sensor1.split("=")[1] #πχ θα περιέχει το 21.3
    sensor2Value = sensor2.split("=")[1]

    #Αποθήκευσε τη τιμή στη βάση

    #INSERT
    if sensor1 <> None:
        sql = "INSERT INTO sensors (value,nodeid, field) VALUES (%s, %s, %s)"
        val = (sensor1Value, nodeid, "1")
        mycursor.execute(sql, val)
        mydb.commit()

    if sensor1 <> None:
        sql = "INSERT INTO sensors (value,nodeid, field) VALUES (%s, %s, %s)"
        val = (sensor2Value, nodeid, "2")
        mycursor.execute(sql, val)
        mydb.commit()


    return "ok"
except:
    print("Invalid decryption")
    return "Invalid decryption"
```

```
        else:
            print("Wrong client")
            return "Wrong client"
    else:
        print("Wrong server")
        return "Wrong server"


    return "0"


#Step -7(run the app)
if __name__ == '__main__':
    app.run(debug=True)


#<td>{% if values[i][4]==1 %} {{values[i][2]}} {% endif %}</td>
```