ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Μελέτη και σχεδίαση συστήματος διαδικτύου των πραγμάτων με websockets»

Φοιτητής

Λουκιανός Ιωάννης 514081

Επιβλέπων

Δρ. Κυριάκος Τσιακμάκης

ΦΕΒΡΟΥΑΡΙΟΣ 2023

Μελέτη και υλοποίηση μεθόδων ασφάλειας σε συστήματα διαδικτύου των πραγμάτων

Κωδικός: 21366

Φοιτητής: Λουκιανός Ιωάννης

Εισηγητής: Δρ Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 15-10-2021

Ημερομηνία περάτωσης Π.Ε. 14-01-2023

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 αφορά τη μελέτη και σχεδίαση συστήματος διαδικτύου των πραγμάτων με websockets σε γλώσσα python. Αρχικά μελετήθηκε και υλοποιήθηκε σύστημα με χρήση της δημοφιλούς βιβλιοθήκης socket.io. Χρησιμοποιήθηκαν sessions και sockets και αναλύθηκε ο τρόπος χρήσης τους και οι δυνατότητες τους με αυτήν τη βιβλιοθήκη. Στη συνέχεια χρησιμοποιήθηκαν η ειδική βιβλιοθήκη websockets που χρησιμοποιεί τα TCP sockets και παρουσιάστηκε η σύγκριση των δύο επιλογών. Διευκρινίστηκε ότι με τη δεύτερη μπορούμε να κατασκευάσουμε συστήματα που χρησιμοποιούν sockets όταν θέλουμε να εντάξουμε esp32 και άλλα τέτοια μικρουπολογιστικά συστήματα.

« Study and design of internet of things system with websockets »

Abstract

This undergraduate thesis concerns study and design of internet of things system with websockets in Python. A system was initially studied and developed using the popular library of socket.io. Sessions and sockets are used and their capabilities are analyzed. The special webSockets library based on TCP sockets is used and the comparison of the two options is presented. It has been clarified that with the latter we can manufacture systems when we want to use ESP32 and other such micro systems.

Ευχαριστίες

Θέλω να ευχαριστήσω τους γονείς μου για τη βοήθεια τους. Ευχαριστώ και τον επιβλέπων κ. Τσιακμάκη για τη καθοδήγηση του στους κώδικες και στη διόρθωση.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Το HTTP και το WebSocket είναι και τα δύο πρωτόκολλα επικοινωνίας που χρησιμοποιούνται στην επικοινωνία πελάτη-διακομιστή. Το HTTP είναι μονής κατεύθυνσης όπου ο πελάτης στέλνει το αίτημα και ο διακομιστής στέλνει την απάντηση. Όταν ένας χρήστης στέλνει ένα αίτημα στον διακομιστή, αυτό το αίτημα έχει τη μορφή HTTP (ή HTTPS) και αφού λάβει το αίτημα-μήνυμα στέλνει την απάντηση στον πελάτη. Κάθε αίτημα σχετίζεται με μια αντίστοιχη απάντηση, μετά την αποστολή της απάντησης η σύνδεση κλείνει. Κάθε αίτημα HTTP ή HTTPS δημιουργεί τη νέα σύνδεση με τον διακομιστή κάθε φορά και μετά τη λήψη της απάντησης, η σύνδεση τερματίζεται από μόνη της.

Το HTTP είναι ένα πρωτόκολλο stateless που τρέχει πάνω από το TCP, το οποίο είναι ένα πρωτόκολλο προσανατολισμένο στη σύνδεση και εγγυάται την παράδοση της μεταφοράς πακέτων δεδομένων χρησιμοποιώντας τις μεθόδους τριπλής χειραψίας και μεταδίδει εκ νέου τα χαμένα πακέτα

Το WebSocket είναι αμφίδρομο, ένα πλήρες αμφίδρομο πρωτόκολλο που χρησιμοποιείται στο ίδιο σενάριο επικοινωνίας πελάτη-διακομιστή και σε αντίθεση με το HTTP ξεκινά από ws:// ή wss://. Είναι ένα πρωτόκολλο κατάστασης, που σημαίνει ότι η σύνδεση μεταξύ πελάτη και διακομιστή θα παραμείνει ζωντανή έως ότου τερματιστεί από οποιοδήποτε μέρος (πελάτη ή διακομιστή). Μετά το κλείσιμο της σύνδεσης από κάποιον πελάτη και διακομιστή, η σύνδεση τερματίζεται και από τα δύο άκρα.

Ας πάρουμε ένα παράδειγμα επικοινωνίας πελάτη-διακομιστή, υπάρχει ο πελάτης που είναι ένα πρόγραμμα περιήγησης ιστού και ένας διακομιστής, κάθε φορά που ξεκινάμε τη σύνδεση μεταξύ πελάτη και διακομιστή, ο πελάτης-διακομιστής αποφασίζει να δημιουργήσει μια νέα σύνδεση και αυτή η σύνδεση θα παραμείνει ζωντανή μέχρι να τερματιστεί από κάποιο από αυτούς. Όταν η σύνδεση δημιουργηθεί και είναι ζωντανή, η επικοινωνία πραγματοποιείται χρησιμοποιώντας το ίδιο κανάλι σύνδεσης μέχρι να τερματιστεί.

Με απλά λόγια σε μια HTTP (REST κτλ) σύνδεση ο client μπορεί να στείλει στον server ένα μήνυμα και ο server να ανταποκριθεί. Δεν μπορεί να συμβεί το αντίθετο. Για να υπάρχει επικοινωνία μεταξύ τους αμφίδρομη πρέπει να χρησιμοποιηθεί socket.

Σε αυτήν την εργασία χρησιμοποιήθηκε το socket.io και το WebSocket library για να μελετηθεί για χρήση σε IoT εφαρμογές.

Τα αποτελέσματα έδειξαν ότι για να χρησιμοποιηθεί σαν client ένα esp32, raspberry ή γενικά ένα μικρουπολογιστικό σύστημα που να μπορεί να επικοινωνήσει αμφίδρομα με τον server θα χρειαστεί να χρησιμοποιήσει την websocket βιβλιοθήκη,

1.2 Δομή της εργασίας

Στο πρώτο κεφάλαιο παρουσιάζεται η εισαγωγή της εργασίας και η δομή της.

Στο δεύτερο κεφάλαιο παρουσιάζεται η εισαγωγή στις βασικές τεχνικές επικοινωνίας μεταξύ server και client

Στο τρίτο κεφάλαιο περιγράφεται η γλώσσα και περιβάλλον προγραμματισμού που χρησιμοποιήθηκε.

Στο τέταρτο κεφάλαιο αναλύονται οι εφαρμογές με Websockets για IoT που χρησιμοποιήθηκαν στην εργασία

Στο τελευταίο κεφάλαιο παρουσιάζονται τα συμπεράσματα της εργασίας

Στο τέλος της εργασίας υπάρχει το παράρτημα με τους κώδικες.

Κεφάλαιο 2ο: Εισαγωγή στις βασικές τεχνικές επικοινωνίας μεταξύ server και client

2.1 API

Μια διεπαφή προγραμματισμού εφαρμογών (application programming interface - API) είναι ένας τρόπος για δύο ή περισσότερα προγράμματα υπολογιστών να επικοινωνούν μεταξύ τους. Είναι ένας τύπος διεπαφής λογισμικού, που προσφέρει υπηρεσία σε άλλα κομμάτια λογισμικού. Ένα έγγραφο ή πρότυπο που περιγράφει τον τρόπο δημιουργίας ή χρήσης μιας τέτοιας σύνδεσης ή διεπαφής ονομάζεται προδιαγραφή API. Ένα σύστημα υπολογιστή που πληροί αυτό το πρότυπο λέγεται ότι υλοποιεί ή εκθέτει ένα API. Ο όρος API μπορεί να αναφέρεται είτε στην προδιαγραφή είτε στην υλοποίηση.

Σε αντίθεση με μια διεπαφή χρήστη, η οποία συνδέει έναν υπολογιστή με ένα άτομο, μια διεπαφή προγραμματισμού εφαρμογών συνδέει υπολογιστές ή κομμάτια λογισμικού μεταξύ τους. Δεν προορίζεται να χρησιμοποιηθεί απευθείας από άτομο (τελικό χρήστη) εκτός από προγραμματιστή υπολογιστή που το ενσωματώνει στο λογισμικό.

Ένα API αποτελείται συχνά από διαφορετικά μέρη που λειτουργούν ως εργαλεία ή υπηρεσίες που είναι διαθέσιμα στον προγραμματιστή. Ένα πρόγραμμα ή ένας προγραμματιστής που χρησιμοποιεί ένα από αυτά τα μέρη λέγεται ότι καλεί αυτό το τμήμα του API. Οι κλήσεις που απαρτίζουν το API είναι επίσης γνωστές ως υπορουτίνες, μέθοδοι, αιτήματα ή τελικά σημεία. Μια προδιαγραφή API ορίζει αυτές τις κλήσεις, που σημαίνει ότι εξηγεί πώς να χρησιμοποιηθούν ή να εφαρμοστούν.

Στη δημιουργία εφαρμογών, ένα API απλοποιεί τον προγραμματισμό αφαιρώντας την υποκείμενη υλοποίηση και εκθέτοντας μόνο αντικείμενα ή ενέργειες που χρειάζεται ο προγραμματιστής. Ενώ μια γραφική διεπαφή για ένα πρόγραμμα-πελάτη ηλεκτρονικού ταχυδρομείου μπορεί να παρέχει σε έναν χρήστη ένα κουμπί που εκτελεί όλα τα βήματα για την ανάκτηση και την επισήμανση νέων μηνυμάτων ηλεκτρονικού ταχυδρομείου, ένα API για είσοδο/έξοδο αρχείου μπορεί να δώσει στον προγραμματιστή μια λειτουργία που αντιγράφει ένα αρχείο από τη μια τοποθεσία στην άλλη χωρίς απαιτώντας από τον προγραμματιστή να κατανοήσει τις λειτουργίες του συστήματος αρχείων που λαμβάνουν χώρα στο παρασκήνιο.

2.2 HTTP API

Ένα web API είναι ένα πρωτόκολλο που περιγράφει πώς οι clients μπορούν να έχουν πρόσβαση σε πόρους και ποιες μέθοδοι λειτουργούν με την αρχιτεκτονική του συστήματος. Αυτοί οι πόροι μπορεί να είναι ποικίλων τύπων μέσων όπως στοιχεία JavaScript ή HTML, μεταδεδομένα ή εικόνες.

Είναι κάτι ως οδηγός μετάφρασης από τη μια τεχνολογία στην άλλη. Ένα HTTP API είναι ένα API που χρησιμοποιεί το Πρωτόκολλο Μεταφοράς Υπερκειμένου(Hypertext Transfer Protocol) ως πρωτόκολλο επικοινωνίας μεταξύ των δύο συστημάτων. Τα API HTTP εκθέτουν τα τελικά σημεία ως πύλες API για αιτήματα HTTP για πρόσβαση σε διακομιστή.

Τα HTTP API είναι μια ευρεία κατηγορία, που σημαίνει ότι διατίθενται σε διάφορες μορφές με βάση την περίπτωση χρήσης-στόχου τους. Τα API HTTP κατηγοριοποιούνται περαιτέρω από τις αρχές αρχιτεκτονικού σχεδιασμού που χρησιμοποιούνται κατά τη δημιουργία τους. Τα περισσότερα χρησιμοποιούνται σε συστήματα πληροφοριών υπερμέσων ή ανάπτυξη ιστού, αλλά το καθένα έχει ιδιαίτερα πλεονεκτήματα και μειονεκτήματα.

Στο server φτιάχνουμε μια απλή συνάρτηση που απαντάει απλά σε ένα curl κάλεσμα. Στην πραγματικότητα το Http API δεν έχει δυνατότητες και έχει αντικατασταθεί από το REST API.

Αφού γνωρίσουμε το REST API θα αναφερθούμε στις διαφορές τους.

2.3 REST API

Το REST API σημαίνει Representational State Transfer και είναι ένα αρχιτεκτονικό μοτίβο για τη δημιουργία υπηρεσιών Ιστού. Αναπτύχθηκε από τον Roy Fielding το 2000 και οδήγησε σε μια αυξανόμενη συλλογή υπηρεσιών web RESTful που ακολουθούν τις αρχές REST. Τώρα, τα API REST έχουν ευρεία χρήση από τους προγραμματιστές εφαρμογών λόγω του πόσο απλά επικοινωνεί με άλλα μηχανήματα μέσω πολύπλοκων λειτουργιών όπως το Simple Object Access Protocol (SOAP).

Το REST είναι ένα σύνολο κανόνων που ορίζει τις βέλτιστες πρακτικές για την κοινή χρήση δεδομένων μεταξύ των πελατών και του διακομιστή. Είναι ουσιαστικά ένα στυλ σχεδίασης που χρησιμοποιείται κατά τη δημιουργία HTTP ή άλλων API που αρκεί να χρησιμοποιήσετε μόνο συναρτήσεις CRUD (create, read, update and delete), ανεξάρτητα από την πολυπλοκότητα. Οι εφαρμογές REST χρησιμοποιούν μεθόδους HTTP όπως GET, POST, DELETE και PUT. Το REST δίνει έμφαση στην επεκτασιμότητα των στοιχείων και στην απλότητα των διεπαφών.

Δεν είναι όλα τα HTTP API REST. Το API πρέπει να πληροί τις ακόλουθες αρχιτεκτονικές απαιτήσεις για να θεωρηθεί REST API:

1. Client-server: Οι εφαρμογές REST διαθέτουν διακομιστή που διαχειρίζεται τα δεδομένα και την κατάσταση της εφαρμογής. Ο διακομιστής επικοινωνεί με έναν πελάτη που χειρίζεται τις αλληλεπιδράσεις των χρηστών. Ένας σαφής διαχωρισμός χωρίζει τις δύο συνιστώσες. Αυτό σημαίνει ότι μπορείτε να τα ενημερώσετε και να τα βελτιώσετε σε ανεξάρτητα κομμάτια.

2. Stateless: Οι διακομιστές δεν διατηρούν την κατάσταση του client, οι πελάτες διαχειρίζονται τη δική τους κατάσταση εφαρμογής. Τα αιτήματα του πελάτη προς τον διακομιστή περιέχουν όλες τις πληροφορίες που απαιτούνται για την επεξεργασία τους.

3. Με δυνατότητα προσωρινής αποθήκευσης: οι διακομιστές πρέπει να επισημαίνουν τις απαντήσεις τους ως προσωρινές ή όχι. Τα συστήματα και οι πελάτες μπορούν να αποθηκεύουν τις απαντήσεις στην κρυφή μνήμη όταν είναι βολικό για να βελτιώσουν την απόδοση. Διαθέτουν επίσης πληροφορίες που δεν μπορούν να αποθηκευτούν στην προσωρινή μνήμη, επομένως κανένας πελάτης δεν χρησιμοποιεί παλιά δεδομένα.

4. Ομοιόμορφο interface: Αυτό είναι το πιο γνωστό χαρακτηριστικό ή κανόνας του REST. Το κεντρικό χαρακτηριστικό που διακρίνει το αρχιτεκτονικό στυλ REST από άλλα στυλ που βασίζονται σε δίκτυο είναι η έμφαση που δίνει σε μια ομοιόμορφη διεπαφή μεταξύ των στοιχείων. Οι υπηρεσίες REST παρέχουν δεδομένα ως πόρους, με καλό χώρο ονομάτων.

Αυτοί οι περιορισμοί συνδυάζονται για να δημιουργήσουν μια εφαρμογή με ισχυρά όρια και σαφή διαχωρισμό. Ο πελάτης λαμβάνει δεδομένα διακομιστή όταν του ζητηθεί. Ο πελάτης χειρίζεται ή εμφανίζει τα δεδομένα. Ο πελάτης ειδοποιεί τον διακομιστή για τυχόν αλλαγές κατάστασης. Τα API REST δεν αποκρύπτουν δεδομένα από τον πελάτη, παρά μόνο υλοποιήσεις.

Τα REST API είναι ιδανικά για τη δημιουργία εφαρμογών γενικής χρήσης ώστε να είναι επεκτάσιμες στο μέλλον.

Τα REST API είναι εγγενώς αποσυνδεδεμένα από την τεχνολογία πελάτη, που σημαίνει ότι η εφαρμογή σας μπορεί να λειτουργήσει καλά σε iOS και Android, πρόγραμμα περιήγησης ή μια συσκευή του μέλλοντος με ελάχιστη δυσκολία. Ως αποτέλεσμα, μπορείτε να δημιουργήσετε την εφαρμογή σας με ευκολία σχετικά με τη σύνδεση σε συγκεκριμένες τεχνολογίες από την πλευρά του πελάτη και να εστιάσετε στην ανάπτυξη της ίδιας της εφαρμογής. Τα RESTful API είναι επομένως πιο επεκτάσιμα και έχουν μεγαλύτερη διάρκεια ζωής.

Επίσης, η προσωρινή αποθήκευση επιτρέπει στα API REST να είναι πιο επεκτάσιμα, μειώνοντας τον αριθμό των αιτημάτων που χρειάζεται να επεξεργαστεί ο διακομιστής για έναν πόρο σε ζήτηση. Αποθηκεύοντας την απόκριση στην προσωρινή μνήμη και στέλνοντας την αποθηκευμένη απόκριση αντί για επανεπεξεργασία του αιτήματος, οι υπηρεσίες RESTful μπορούν να επεξεργαστούν περισσότερα αιτήματα με λιγότερους πόρους.

Η αρχιτεκτονική Stateless είναι τόσο χρήσιμη όσο και περιοριστική για το REST. Από τη μία πλευρά, οι ρυθμίσεις Stateless αυξάνουν τη διάρκεια ζωής του συστήματος επιτρέποντάς σας να αλλάζετε διακομιστές όταν τελικά καταστεί απαρχαιωμένο.

Από την άλλη, όλα τα αιτήματα πρέπει να περιλαμβάνουν όλα τα δεδομένα για να ολοκληρωθεί το αίτημα στο ωφέλιμο φορτίο μηνυμάτων. Αυτό λειτουργεί καλά όταν χρειάζεται μόνο λίγα δεδομένα, αλλά γρήγορα γίνεται μη διαχειρίσιμο για πολύπλοκα αιτήματα. Η αντιστάθμιση με το REST είναι μεταξύ του μεγέθους του ωφέλιμου φορτίου και της ευελιξίας.

Είναι επίσης σημαντικό να σημειωθεί ότι δεν χρειάζεται να τηρείται αυστηρά την αρχιτεκτονική REST σε όλες τις περιπτώσεις. Το REST είναι ένα χρήσιμο εργαλείο στη ζώνη εργαλείων του προγραμματιστή και ένας καλός γενικός κανόνας που πρέπει να ακολουθεί, αλλά δεν πρέπει να είναι το δόγμα προγραμματισμού του.

2.4 Διαφορές HTTP και REST API

Τα REST API και τα HTTP API είναι και τα δύο προϊόντα RESTful API. Τα REST API υποστηρίζουν περισσότερες δυνατότητες από τα HTTP API, ενώ τα HTTP API είναι σχεδιασμένα με ελάχιστες δυνατότητες, ώστε να μπορούν να προσφέρονται σε χαμηλότερη τιμή. Τα REST API έχουν υπηρεσίες όπως κλειδιά API, περιορισμός ανά πελάτη, επικύρωση αιτήματος, ιδιωτικά τελικά σημεία API. Τα HTTP API δεν έχουν τις δυνατότητες που περιλαμβάνονται στα REST API.

Τα βασικά δομικά στοιχεία των συστημάτων RESTful είναι οι πόροι. Ένας πόρος μπορεί να είναι μια ιστοσελίδα, μια ροή βίντεο ή μια εικόνα, για παράδειγμα. Ένας πόρος μπορεί να είναι ακόμη και μια αφηρημένη έννοια, όπως η λίστα όλων των χρηστών σε μια βάση δεδομένων ή η πρόγνωση καιρού για μια συγκεκριμένη τοποθεσία. Ο μόνος πραγματικός περιορισμός είναι ότι κάθε πόρος σε ένα σύστημα είναι μοναδικά αναγνωρίσιμος.

Επιπλέον, οι πόροι μπορεί να είναι διαθέσιμοι σε πολλαπλές αναπαραστάσεις. Σε ένα μοντέλο πελάτη-διακομιστή, ο διακομιστής είναι υπεύθυνος για τη διαχείριση της κατάστασης του πόρου, αλλά ένας πελάτης μπορεί να επιλέξει με ποια αναπαράσταση προτιμά να αλληλεπιδράσει.

Σε αντίθεση με το REST, το πρωτόκολλο μεταφοράς υπερκειμένου (HTTP) είναι ένα πρότυπο με καλά καθορισμένους περιορισμούς. Το HTTP είναι το πρωτόκολλο επικοινωνίας που τροφοδοτεί τις περισσότερες από τις καθημερινές μας αλληλεπιδράσεις στο διαδίκτυο όπως προγράμματα περιήγησης Ιστού που φορτώνουν ιστοσελίδες.

Το HTTP αν και δεν είναι το ίδιο με το REST, παρουσιάζει πολλά χαρακτηριστικά ενός συστήματος RESTful. Είναι σημαντικό ότι η χρήση HTTP δεν απαιτείται για ένα σύστημα RESTful. Τυχαίνει το HTTP να είναι μια καλή αρχή επειδή εμφανίζει πολλές ιδιότητες RESTful.

2.5 Sockets και Websockets

2.5.1 TCP Sockets

Το TCP socket είναι μια θεμελιώδης έννοια στη λειτουργία του κόσμου εφαρμογών TCP/IP.

Για παράδειγμα, εάν έχουμε έναν ιστότοπο που εκτελείται στη διεύθυνση IP 30.1.1.1, το socket που αντιστοιχεί στον διακομιστή HTTP για αυτόν τον ιστότοπο θα είναι 30.1.1.1:80.

Τα sockets ταξινομούνται σε socket ροής,

socket.SOCK_STREAM

ή socket datagram, socket.SOCK_DGRAM,

ανάλογα με το υποκείμενο πρωτόκολλο που χρησιμοποιούμε.

SOCK_STREAM: Συνδέεται με το πρωτόκολλο TCP και παρέχει ασφάλεια στη μετάδοση δεδομένων και ασφάλεια στη λήψη δεδομένων. Στην επικοινωνία προσανατολισμένη στη σύνδεση, πρέπει να υπάρχει ένα κανάλι που έχει δημιουργηθεί πριν μεταφέρουμε δεδομένα.

SOCK_DGRAM: Συνδέεται με το πρωτόκολλο UDP και υποδεικνύει ότι τα πακέτα θα ταξιδεύουν στον τύπο του datagram, το οποίο έχει ασύγχρονο στυλ επικοινωνίας. Για υποδοχές UDP, μπορούμε να στείλουμε δεδομένα χωρίς σύνδεση. Αυτό λοιπόν ονομάζεται χωρίς σύνδεση.

Το TCP socket είναι socket προσανατολισμένο στη σύνδεση που χρησιμοποιεί το πρωτόκολλο ελέγχου μετάδοσης (TCP). Απαιτούνται τρία πακέτα για τη δημιουργία μιας σύνδεσης:

το πακέτο SYN,

το πακέτο SYN-ACK

και το πακέτο ACK.

Το TCP socket ορίζεται από τη διεύθυνση IP του μηχανήματος και τη θύρα που χρησιμοποιεί. Το TCP socket εγγυάται ότι όλα τα δεδομένα λαμβάνονται και επιβεβαιώνονται.

Για παράδειγμα, στέλνουμε ένα αίτημα HTTP από τον πελάτη μας στη διεύθυνση 60.1.1.1 στον ιστότοπο στη διεύθυνση 70.1.1.1. Ο διακομιστής για αυτόν τον ιστότοπο θα χρησιμοποιεί τον γνωστό αριθμό θύρας 80, επομένως η υποδοχή του είναι 70.1.1.1:80.

Αν ο πελάτης έχει 4000 για το πρόγραμμα περιήγησης ιστού, επομένως το socket πελάτη είναι 60.1.1.1:4000. Η συνολική σύνδεση μεταξύ αυτών των συσκευών μπορεί να περιγραφεί χρησιμοποιώντας αυτό το ζεύγος υποδοχών: (70.1.1.1:80, 60.1.1.1:400).

Η socket σύνδεση είναι αξιόπιστη και τα πακέτα που χάνοντια στο δίκτυο εντοπίζονται και αναμεταδίδονται από τον αποστολέα.

Έχει παράδοση δεδομένων κατά παραγγελία δηλαδή τα δεδομένα διαβάζονται από την εφαρμογή με τη σειρά που γράφτηκαν από τον αποστολέα.

Σε αντίθεση με το TCP, το UDP είναι πρωτόκολλο χωρίς σύνδεση, επομένως προφανώς δεν χρησιμοποιεί συνδέσεις. Το ζεύγος sockets στις συσκευές αποστολής και λήψης μπορεί ακόμα να χρησιμοποιηθεί για την αναγνώριση των δύο διεργασιών που ανταλλάσσουν δεδομένα, αλλά επειδή δεν υπάρχουν συνδέσεις, το ζεύγος sockets δεν έχει τη σημασία που έχει στο TCP.

Το πακέτο αποστέλλεται με επιτυχία. Το UDP δεν ενδιαφέρεται αν το πακέτο λαμβάνεται από το άλλο τελικό σημείο ή όχι.

Στο παρακάτω παράδειγμα βλέπουμε ένα απλό TCP socket μεταξύ client και server

που δημιουργήσαμε για αυτήν την εργασία:

Server

```
import socket


HOST = 'localhost'
PORT = 7700

print("Server Starting....")
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
```

```
    print("Waiting Client")
    conn, addr = s.accept()
    with conn:
        print('Connected by', addr)
        while True:
            data = conn.recv(1024)
            if not data:
                break
            print("Receive form client: ",data)
            conn.sendall(b'return something')
```

Client

```
import socket

HOST = 'localhost'
PORT = 7700

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    x = range(1, 6, 2)
    for n in x:
        s.sendall(b'sensor value 34 ')
    data = s.recv(1024)
print('Received from server', repr(data))
```

2.5.2 WebSockets

Web Sockets ορίζονται ως μια αμφίδρομη επικοινωνία μεταξύ των διακομιστών και των πελατών, που σημαίνει ότι και τα δύο μέρη επικοινωνούν και ανταλλάσσουν δεδομένα ταυτόχρονα. Αυτό το πρωτόκολλο ορίζει μια πλήρη αμφίδρομη επικοινωνία από την αρχή. Τα Web Sockets κάνουν ένα βήμα προς τα εμπρός φέρνοντας πλούσιες λειτουργίες επιφάνειας εργασίας στα προγράμματα περιήγησης ιστού. Αντιπροσωπεύει μια εξέλιξη, η οποία αναμενόταν για μεγάλο χρονικό διάστημα στην τεχνολογία web πελάτη/διακομιστή.

Με κυριολεκτικούς όρους, η χειραψία μπορεί να οριστεί ως το πιάσιμο και το κούνημα των δεξιών χεριών από δύο άτομα, για να συμβολίσει τον χαιρετισμό, τα συγχαρητήρια, τη συμφωνία ή τον αποχαιρετισμό. Στην επιστήμη των υπολογιστών, η χειραψία είναι μια διαδικασία που διασφαλίζει ότι ο διακομιστής είναι σε συγχρονισμό με τους πελάτες του. Η χειραψία είναι η βασική ιδέα του πρωτοκόλλου Web Socket.

Web Sockets ορίζονται ως μια αμφίδρομη επικοινωνία μεταξύ των διακομιστών και των πελατών, που σημαίνει ότι και τα δύο μέρη επικοινωνούν και ανταλλάσσουν δεδομένα ταυτόχρονα.

Τα βασικά σημεία των Web Sockets είναι η πραγματική ταυτόχρονη χρήση και η βελτιστοποίηση της απόδοσης, με αποτέλεσμα πιο ανταποκρινόμενες και πλούσιες εφαρμογές web.

Τα κύρια χαρακτηριστικά των Web Sockets είναι τα εξής:

Το πρωτόκολλο Web Sockets τυποποιείται, πράγμα που σημαίνει ότι η επικοινωνία σε πραγματικό χρόνο μεταξύ των διακομιστών Ιστού και των πελατών είναι δυνατή με τη βοήθεια αυτού του πρωτοκόλλου.

Τα Web Sockets μετατρέπονται σε πρότυπο cross platform για επικοινωνία σε πραγματικό χρόνο μεταξύ ενός πελάτη και του διακομιστή.

Αυτό το πρότυπο επιτρέπει νέους τύπους εφαρμογών. Οι επιχειρήσεις για εφαρμογή web σε πραγματικό χρόνο μπορούν να επιταχύνουν με τη βοήθεια αυτής της τεχνολογίας.

Το μεγαλύτερο πλεονέκτημα του Web Socket είναι ότι παρέχει αμφίδρομη επικοινωνία (full duplex) μέσω μιας μόνο σύνδεσης TCP.

Το HTTP έχει το δικό του σύνολο σχημάτων όπως http και https. Το πρωτόκολλο Web Socket έχει επίσης παρόμοιο σχήμα που ορίζεται στο μοτίβο διεύθυνσης URL του.

Το παρακάτω δείχνει τη διεύθυνση URL του Web Socket

ws://loukianos.com:8000/sensors.php

Κάποιες από τις υπάρχουσες τεχνικές, οι οποίες χρησιμοποιούνται για την αμφίδρομη επικοινωνία μεταξύ διακομιστή και πελάτη είναι οι εξής:

- Polling
- Long Polling
- Streaming
- Postback and AJAX
- HTML5

Το Polling μπορεί να οριστεί ως μια μέθοδος, η οποία εκτελεί περιοδικές αιτήσεις ανεξάρτητα από τα δεδομένα που υπάρχουν στη μετάδοση. Τα περιοδικά αιτήματα αποστέλλονται με σύγχρονο τρόπο. Ο πελάτης κάνει μια περιοδική αίτηση σε ένα καθορισμένο χρονικό διάστημα στον Διακομιστή. Η απάντηση του διακομιστή περιλαμβάνει διαθέσιμα δεδομένα ή κάποιο προειδοποιητικό μήνυμα σε αυτήν.

Το Long polling, όπως υποδηλώνει το όνομα, περιλαμβάνει παρόμοια τεχνική όπως το polling. Ο πελάτης και ο διακομιστής διατηρούν τη σύνδεση ενεργή έως ότου ληφθούν ορισμένα δεδομένα ή λήξει το χρονικό όριο. Εάν η σύνδεση χαθεί για κάποιους λόγους, ο πελάτης μπορεί να ξεκινήσει από την αρχή και να εκτελέσει διαδοχική αίτηση.

Το Streaming θεωρείται ως η καλύτερη επιλογή για μετάδοση δεδομένων σε πραγματικό χρόνο. Ο διακομιστής διατηρεί τη σύνδεση ανοιχτή και ενεργή με τον πελάτη έως ότου ληφθούν τα απαιτούμενα δεδομένα. Σε αυτήν την περίπτωση, η σύνδεση λέγεται ότι είναι ανοιχτή επ' αόριστον. Η ροή περιλαμβάνει κεφαλίδες HTTP που αυξάνει το μέγεθος του αρχείου, αυξάνοντας την καθυστέρηση. Αυτό μπορεί να θεωρηθεί ως σημαντικό μειονέκτημα.

Το AJAX βασίζεται στο αντικείμενο XmlHttpRequest της Javascript. Είναι μια συντομευμένη μορφή Asynchronous Javascript και XML. Το αντικείμενο XmlHttpRequest επιτρέπει την εκτέλεση του Javascript χωρίς επαναφόρτωση ολόκληρης της ιστοσελίδας. Το AJAX στέλνει και λαμβάνει μόνο ένα μέρος της ιστοσελίδας.

Το HTML5 είναι ένα ισχυρό πλαίσιο για την ανάπτυξη και το σχεδιασμό εφαρμογών Ιστού. Οι κύριοι πυλώνες περιλαμβάνουν CSS και Javascript API μαζί.

2.6 REST API – προσπάθεια παρόμοιας λειτουργίας με websockets

Το REST API (γνωστό και ως RESTful API) είναι μια διεπαφή προγραμματισμού εφαρμογών (API ή web API) που συμμορφώνεται με τους περιορισμούς του αρχιτεκτονικού στυλ REST και επιτρέπει την αλληλεπίδραση με τις υπηρεσίες ιστού RESTful. Το REST σημαίνει μεταφορά κατάστασης αναπαράστασης και δημιουργήθηκε από τον επιστήμονα υπολογιστών Roy Fielding.

Ένα API είναι ένα σύνολο ορισμών και πρωτοκόλλων για τη δημιουργία και την ενοποίηση λογισμικού εφαρμογών. Μερικές φορές αναφέρεται ως σύμβαση μεταξύ ενός παρόχου πληροφοριών και ενός χρήστη πληροφοριών—καθορίζοντας το περιεχόμενο που απαιτείται από τον καταναλωτή (request) και το περιεχόμενο που απαιτείται από τον παραγωγό (response).

Με άλλα λόγια, εάν θέλετε να αλληλεπιδράσετε με έναν υπολογιστή ή ένα σύστημα για να ανακτήσετε πληροφορίες ή να εκτελέσετε μια λειτουργία, ένα API σάς βοηθά να επικοινωνήσετε αυτό που θέλετε σε αυτό το σύστημα, ώστε να μπορεί να κατανοήσει και να εκπληρώσει το αίτημα.

Μπορείτε να σκεφτείτε ένα API ως μεσολαβητή μεταξύ των χρηστών ή των πελατών και των πόρων ή των υπηρεσιών Ιστού που θέλουν να αποκτήσουν. Είναι επίσης ένας τρόπος για έναν οργανισμό να μοιράζεται πόρους και πληροφορίες διατηρώντας παράλληλα την ασφάλεια, τον έλεγχο και τον έλεγχο ταυτότητας καθορίζοντας ποιος έχει πρόσβαση σε τι.

Ένα άλλο πλεονέκτημα ενός API είναι ότι δεν χρειάζεται να γνωρίζετε τις ιδιαιτερότητες της προσωρινής αποθήκευσης δηλαδή πώς ανακτάται ο πόρος σας ή από πού προέρχεται.

Το REST είναι ένα σύνολο αρχιτεκτονικών περιορισμών, όχι ένα πρωτόκολλο ή ένα πρότυπο. Οι προγραμματιστές API μπορούν να εφαρμόσουν το REST με διάφορους τρόπους.

Όταν ένα αίτημα πελάτη γίνεται μέσω ενός RESTful API, μεταφέρει μια αναπαράσταση της κατάστασης του πόρου στον αιτούντα ή στο τελικό σημείο. Αυτές οι πληροφορίες, ή αναπαράσταση, παραδίδονται σε μία από τις διάφορες μορφές μέσω HTTP JSON (Javascript Object Notation), HTML, Python, PHP. Το JSON είναι η πιο δημοφιλής μορφή αρχείου που χρησιμοποιείται γενικά.

Οι κεφαλίδες και οι παράμετροι είναι επίσης σημαντικές στις μεθόδους HTTP ενός αιτήματος RESTful API HTTP, καθώς περιέχουν σημαντικές πληροφορίες αναγνωριστικού ως προς τα μεταδεδομένα του αιτήματος, την εξουσιοδότηση, το URI (Uniform Resource Identifier), την προσωρινή αποθήκευση, τα cookie και περισσότερο. Υπάρχουν κεφαλίδες αιτημάτων και κεφαλίδες απόκρισης, το καθένα με τις δικές του πληροφορίες σύνδεσης HTTP και κωδικούς κατάστασης.

Προκειμένου ένα API να θεωρείται RESTful, πρέπει να συμμορφώνεται με κάποια κριτήρια.

Η αρχιτεκτονική πελάτη-διακομιστή που αποτελείται από πελάτες, διακομιστές και πόρους, με αιτήματα διαχειρίζονται μέσω HTTP.

Η επικοινωνία πελάτη-διακομιστή είναι stateless, που σημαίνει ότι δεν αποθηκεύονται πληροφορίες πελάτη μεταξύ των αιτημάτων λήψης και κάθε αίτημα είναι ξεχωριστό και μη συνδεδεμένο.

Τα δεδομένα με δυνατότητα αποθήκευσης στην προσωρινή μνήμη βελτιστοποιούν τις αλληλεπιδράσεις πελάτη-διακομιστή.

Οι πόροι που ζητούνται είναι αναγνωρίσιμοι και ξεχωριστοί από τις παραστάσεις που αποστέλλονται στον πελάτη.

Οι πόροι μπορούν να διευθετηθούν από τον πελάτη μέσω της αναπαράστασης που λαμβάνει, επειδή η αναπαράσταση περιέχει αρκετές πληροφορίες για να το κάνει.

Τα μηνύματα που επιστρέφονται στον πελάτη έχουν αρκετές πληροφορίες για να περιγράψουν τον τρόπο με τον οποίο ο πελάτης πρέπει να τα επεξεργαστεί.

Το hypertext είναι διαθέσιμο, πράγμα που σημαίνει ότι μετά την πρόσβαση σε έναν πόρο, ο πελάτης θα πρέπει να μπορεί να χρησιμοποιεί υπερσυνδέσμους για να βρει όλες τις άλλες τρέχουσες διαθέσιμες ενέργειες που μπορεί να κάνει.

Απαιτείται ένα πολυεπίπεδο σύστημα που οργανώνει κάθε τύπο διακομιστή και περιλαμβάνει την ανάκτηση των ζητούμενων πληροφοριών σε ιεραρχίες, αόρατες στον πελάτη.

Το REST API εξακολουθεί να θεωρείται πιο εύκολο στη χρήση από ένα προκαθορισμένο πρωτόκολλο όπως το SOAP (Simple Object Access Protocol), το οποίο έχει συγκεκριμένες απαιτήσεις όπως η ανταλλαγή μηνυμάτων XML (XML) και η ενσωματωμένη ασφάλεια και συμμόρφωση συναλλαγών που το καθιστούν πιο αργά και πιο βαριά.

Το REST είναι ένα σύνολο κατευθυντήριων γραμμών που μπορούν να εφαρμοστούν ανάλογα με τις ανάγκες, καθιστώντας τα API REST πιο γρήγορα και πιο ελαφριά, με αυξημένη επεκτασιμότητα ιδανικά για το Internet of Things (IoT) και την ανάπτυξη εφαρμογών για κινητά.

2.7 MQTT

Το MQTT ένα πρωτόκολλο ανταλλαγής μηνυμάτων για το Internet of Things (IoT). Το MQTT σημαίνει MQ Telemetry Transport. Το πρωτόκολλο είναι ένα σύνολο κανόνων που καθορίζει τον τρόπο με τον οποίο οι συσκευές IoT μπορούν να δημοσιεύουν και να εγγραφούν σε δεδομένα μέσω του Διαδικτύου. Το MQTT χρησιμοποιείται για ανταλλαγή μηνυμάτων και δεδομένων μεταξύ IoT και βιομηχανικών συσκευών IoT, όπως ενσωματωμένες συσκευές, αισθητήρες, βιομηχανικά PLC κ.λπ. Το

πρωτόκολλο βασίζεται σε συμβάντα και συνδέει συσκευές χρησιμοποιώντας το μοτίβο δημοσίευσης /εγγραφής (Pub/Sub). Ο αποστολέας (Publisher) και ο παραλήπτης (Subscriber) επικοινωνούν μέσω Topics και αποσυνδέονται ο ένας από τον άλλο. Η μεταξύ τους σύνδεση γίνεται από τον broker MQTT. Ο broker MQTT φιλτράρει όλα τα εισερχόμενα μηνύματα και τα διανέμει σωστά στους Subscribers.

Το MQTT είναι ένα πρωτόκολλο μεταφοράς μηνυμάτων δημοσίευσης/εγγραφής διακομιστή πελάτη. Είναι ελαφρύ, ανοιχτό, απλό και σχεδιασμένο έτσι ώστε να είναι εύκολο στην εφαρμογή του. Αυτά τα χαρακτηριστικά το καθιστούν ιδανικό για χρήση σε πολλές καταστάσεις, συμπεριλαμβανομένων περιορισμένων περιβαλλόντων, όπως για επικοινωνία σε περιβάλλοντα Machine to Machine (M2M) και Internet of Things (IoT), όπου απαιτείται μικρό αποτύπωμα κώδικα ή/και το εύρος ζώνης δικτύου είναι υψηλότερο.

Είναι ένα πολύ ελαφρύ και δυαδικό πρωτόκολλο και λόγω της ελάχιστης επιβάρυνσης πακέτων, το MQTT υπερέχει κατά τη μεταφορά δεδομένων μέσω καλωδίου σε σύγκριση με πρωτόκολλα όπως το HTTP. Μια άλλη σημαντική πτυχή του πρωτοκόλλου είναι ότι το MQTT είναι εξαιρετικά εύκολο να εφαρμοστεί από την πλευρά του πελάτη. Η ευκολία χρήσης ήταν βασικό μέλημα για την ανάπτυξη του MQTT και το καθιστά ιδανικό για περιορισμένες συσκευές με περιορισμένους πόρους σήμερα.

Το πρωτόκολλο MQTT επινοήθηκε το 1999 από τους Andy Stanford-Clark (IBM) και Arlen Nipper. Χρειάζονταν ένα πρωτόκολλο για ελάχιστη απώλεια μπαταρίας και ελάχιστο εύρος ζώνης για να συνδεθούν με πετρελαιαγωγούς μέσω δορυφόρου. Οι δύο εφευρέτες καθόρισαν αρκετές απαιτήσεις για το μελλοντικό πρωτόκολλο:

Απλή υλοποίηση

Ποιότητα παροχής δεδομένων υπηρεσίας

Ελαφρύ και αποδοτικό εύρος ζώνης

Δεδομένα αγνωστικιστικά

Συνεχής επίγνωση συνεδρίας

Αυτοί οι στόχοι εξακολουθούν να βρίσκονται στον πυρήνα του MQTT. Ωστόσο, η πρωταρχική εστίαση του πρωτοκόλλου έχει αλλάξει από ιδιόκτητα ενσωματωμένα συστήματα σε ανοιχτές περιπτώσεις χρήσης Διαδικτύου των πραγμάτων (IoT). Αυτή η αλλαγή εστίασης έχει δημιουργήσει μεγάλη σύγχυση σχετικά με το τι σημαίνει το ακρωνύμιο MQTT. Η σύντομη απάντηση είναι ότι το MQTT δεν θεωρείται πλέον ακρωνύμιο. Το MQTT είναι απλώς το όνομα του πρωτοκόλλου.

Κεφάλαιο 3ο: Γλώσσα και περιβάλλον προγραμματισμού

3.1 Γλώσσα Python

Η Python έχει γίνει μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού στον κόσμο τα τελευταία χρόνια. Χρησιμοποιείται σε οτιδήποτε, από τη μηχανική εκμάθηση μέχρι τη δημιουργία ιστοτόπων και τη δοκιμή λογισμικού. Μπορεί να χρησιμοποιηθεί τόσο από προγραμματιστές και όχι μόνο.

Η Python, μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού στον κόσμο, έχει δημιουργήσει τα πάντα, από τον αλγόριθμο του Netflix μέχρι το λογισμικό που ελέγχει τα αυτόνομα αυτοκίνητα. Η Python είναι μια γλώσσα γενικής χρήσης, που σημαίνει ότι έχει σχεδιαστεί για χρήση σε μια σειρά εφαρμογών, συμπεριλαμβανομένης της επιστήμης δεδομένων, της ανάπτυξης λογισμικού και ιστού, του αυτοματισμού και γενικά της εκτέλεσης εργασιών.

Η Python είναι μια γλώσσα προγραμματισμού υπολογιστών που χρησιμοποιείται συχνά για την κατασκευή ιστοσελίδων και λογισμικού, την αυτοματοποίηση εργασιών και τη διεξαγωγή ανάλυσης δεδομένων. Η Python είναι μια γλώσσα γενικής χρήσης, που σημαίνει ότι μπορεί να χρησιμοποιηθεί για τη δημιουργία μιας ποικιλίας διαφορετικών προγραμμάτων και δεν είναι εξειδικευμένη για συγκεκριμένα προβλήματα. Αυτή η ευελιξία, μαζί με τη φιλικότητα προς τους αρχάριους, την έχει καταστήσει μία από τις πιο χρησιμοποιούμενες γλώσσες προγραμματισμού σήμερα. Μια έρευνα που διεξήχθη από την εταιρεία αναλυτών του κλάδου RedMonk διαπίστωσε ότι ήταν η δεύτερη πιο δημοφιλής γλώσσα προγραμματισμού μεταξύ των προγραμματιστών το 2022.

1 JavaScript

2 Python

3 Java

4 PHP

5 CSS

5 C#

7 C++

8 TypeScript

9 Ruby

10 C

11 Swift

12 R

13 Objective-C

14 Shell

14 Scala

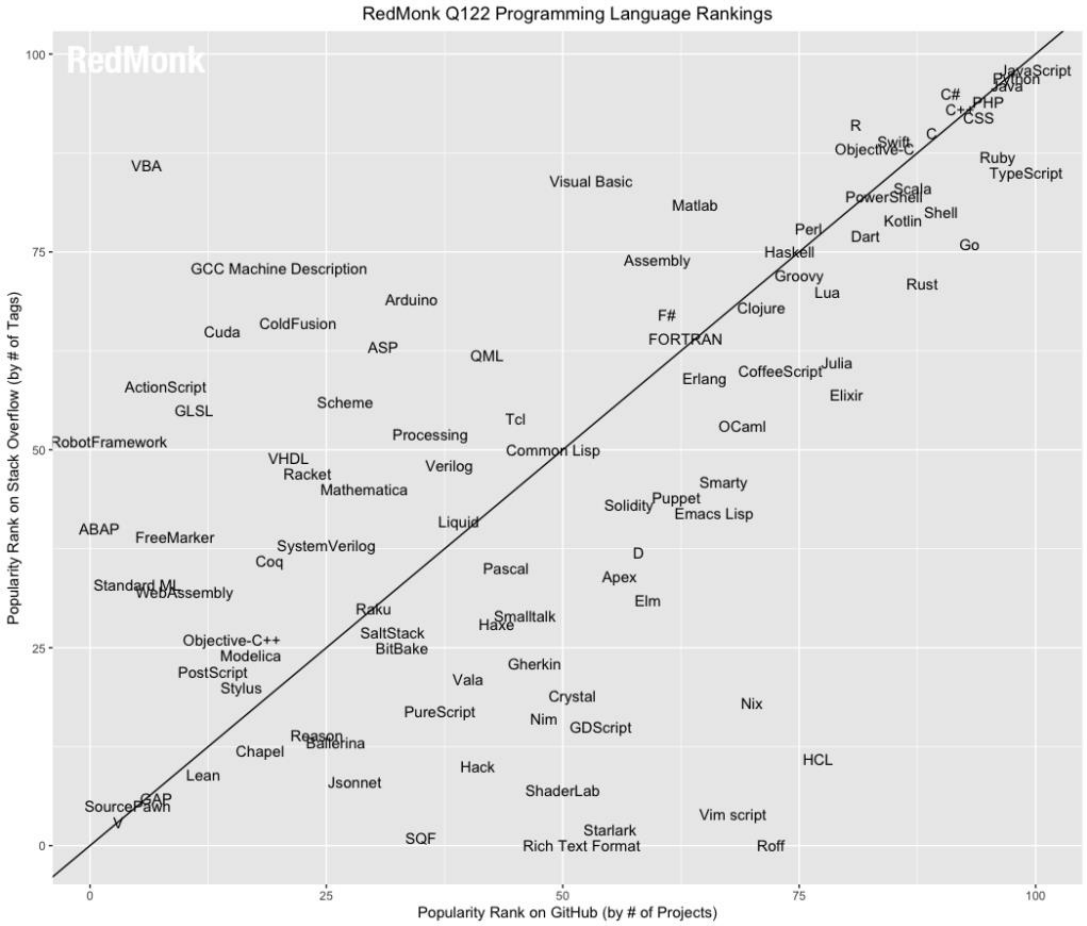
16 Go

17 PowerShell

18 Kotlin

19 Rust

19 Dart



Εικόνα 3.1: Δημοφιλείς γλώσσες

[https://redmonk.com/sogrady/2022/03/28/language-rankings-1-22/]

Η Python χρησιμοποιείται συνήθως για την ανάπτυξη ιστοσελίδων και λογισμικού, αυτοματοποίηση εργασιών, ανάλυση δεδομένων και οπτικοποίηση δεδομένων. Δεδομένου ότι είναι σχετικά εύκολο στην εκμάθηση, η Python έχει υιοθετηθεί από πολλούς μη προγραμματιστές, όπως επιστήμονες.

Μερικά πράγματα από αυτά που μπορεί να κάνει η Python είναι:

Ανάπτυξη διαδικτύου

Ανάλυση δεδομένων και μηχανική μάθηση

Αυτοματοποίηση ή σενάριο

Δοκιμή λογισμικού και δημιουργία πρωτοτύπων

Παιχνίδια ή desktop εφαρμογές

Η Python έχει γίνει βασικό στοιχείο στην επιστήμη δεδομένων, επιτρέποντας σε αναλυτές δεδομένων και άλλους επαγγελματίες να χρησιμοποιούν τη γλώσσα για να διεξάγουν σύνθετους στατιστικούς υπολογισμούς, να δημιουργούν οπτικοποιήσεις δεδομένων, να δημιουργούν αλγόριθμους μηχανικής μάθησης, να χειρίζονται και να αναλύουν δεδομένα και να ολοκληρώσουν άλλες εργασίες που σχετίζονται με δεδομένα.

Η Python μπορεί να δημιουργήσει ένα ευρύ φάσμα διαφορετικών οπτικοποιήσεων δεδομένων, όπως γραφήματα γραμμών και ράβδων, γραφήματα πίτας, ιστογράμματα και τρισδιάστατα σχέδια. Η Python διαθέτει επίσης μια σειρά από βιβλιοθήκες που επιτρέπουν στους κωδικοποιητές να γράφουν προγράμματα για ανάλυση δεδομένων και μηχανική μάθηση πιο γρήγορα και αποτελεσματικά, όπως το TensorFlow.

Η Python χρησιμοποιείται συχνά για την ανάπτυξη του backend ενός ιστότοπου ή μιας εφαρμογής - τα μέρη που ο χρήστης δεν βλέπει. Ο ρόλος της Python στην ανάπτυξη ιστού μπορεί να περιλαμβάνει την αποστολή δεδομένων από και προς τους διακομιστές, την επεξεργασία δεδομένων και την επικοινωνία με βάσεις δεδομένων, τη δρομολόγηση URL και τη διασφάλιση της ασφάλειας. Η Python προσφέρει διάφορα πλαίσια για την ανάπτυξη ιστού. Αυτά που χρησιμοποιούνται συνήθως περιλαμβάνουν το Django και το Flask.

Ορισμένες εργασίες ανάπτυξης ιστού που χρησιμοποιούν Python περιλαμβάνουν μηχανικούς back end, μηχανικούς πλήρους στοίβας, προγραμματιστές Python, μηχανικούς λογισμικού.

Διαβάστε περισσότερα: Πώς να γίνετε προγραμματιστής Ιστού

Εάν θέλουμε να εκτελείται μια εργασία επανειλημμένα, θα μπορούσαμε την αυτοματοποιήσουμε με την Python. Η εγγραφή κώδικα που χρησιμοποιείται για τη δημιουργία αυτών των αυτοματοποιημένων διαδικασιών ονομάζεται δέσμη ενεργειών. Στον κόσμο της κωδικοποίησης, η αυτοματοποίηση μπορεί να χρησιμοποιηθεί για τον έλεγχο σφαλμάτων σε πολλά αρχεία, τη μετατροπή αρχείων, την εκτέλεση απλών μαθηματικών και την αφαίρεση διπλότυπων δεδομένων.

Η Python μπορεί ακόμη και να χρησιμοποιηθεί από σχετικά αρχάριους για την αυτοματοποίηση απλών εργασιών στον υπολογιστή, όπως η μετονομασία αρχείων, η εύρεση και λήψη διαδικτυακού περιεχομένου κτλ.

Στην ανάπτυξη λογισμικού, η Python μπορεί να βοηθήσει σε εργασίες όπως έλεγχος κατασκευής, παρακολούθηση σφαλμάτων και δοκιμές. Με την Python, οι προγραμματιστές λογισμικού μπορούν να αυτοματοποιήσουν τις δοκιμές για νέα προϊόντα ή δυνατότητες. Μερικά εργαλεία Python που χρησιμοποιούνται για τη δοκιμή λογισμικού περιλαμβάνουν το Green.

Η Python δεν είναι μόνο για προγραμματιστές και επιστήμονες δεδομένων. Η εκμάθηση της Python μπορεί να ανοίξει νέες δυνατότητες για όσους ασχολούνται με επαγγέλματα με λιγότερο μεγάλο όγκο δεδομένων, όπως ιδιοκτήτες μικρών επιχειρήσεων. Η Python μπορεί επίσης να επιτρέψει σε μη προγραμματιστές να απλοποιήσουν ορισμένες καθημερινές εργασίες. Όπως η παρακολούθηση τιμών των κρυπτονομισμάτων, λίστα αγορών κτλ

Η Python είναι δημοφιλής για διάφορους λόγους. Έχει μια απλή σύνταξη που μιμείται τη φυσική γλώσσα, επομένως είναι πιο εύκολο να διαβαστεί και να κατανοηθεί. Αυτό καθιστά πιο γρήγορη την κατασκευή έργων και ταχύτερη τη βελτίωσή τους.

Είναι ευέλικτη. Η Python μπορεί να χρησιμοποιηθεί για πολλές διαφορετικές εργασίες, από την ανάπτυξη ιστού έως τη μηχανική εκμάθηση. Είναι φιλική προς τους αρχάριους, καθιστώντας τη δημοφιλές για κωδικοποιητές αρχικού επιπέδου.

Είναι ανοιχτού κώδικα, που σημαίνει ότι είναι δωρεάν για χρήση και διανομή, ακόμη και για εμπορικούς σκοπούς. Το αρχείο λειτουργιών και βιβλιοθηκών της Python δηλαδή τα πακέτα κώδικα που έχουν δημιουργήσει τρίτοι χρήστες για να επεκτείνουν τις δυνατότητες της Python είναι τεράστιο και αυξάνεται.

Η Python έχει μια μεγάλη και ενεργή κοινότητα που συνεισφέρει στη δεξαμενή λειτουργιών και βιβλιοθηκών της Python και λειτουργεί ως χρήσιμος πόρος για άλλους προγραμματιστές.

3.2 Visual Studio Code

Το εργαλείο που χρησιμοποιήσαμε για να γράψουμε python είναι το Visual Studio Code μαζί με τα plugins του.

Το Visual Studio Code είναι ένα δωρεάν, ελαφρύ αλλά ισχυρό πρόγραμμα επεξεργασίας πηγαίου κώδικα που εκτελείται στην επιφάνεια εργασίας σας και στον Ιστό και είναι διαθέσιμο για Windows, macOS, Linux, Raspberry. Έρχεται με ενσωματωμένη υποστήριξη για JavaScript και Node.js και έχει ένα πλούσιο οικοσύστημα επεκτάσεων για άλλες γλώσσες προγραμματισμού (όπως C++, C#, Java, Python, PHP), χρόνους εκτέλεσης (.NET και Unity), περιβάλλοντα (Docker και Kubernetes) και cloud (Microsoft Azure, Google Cloud Platform, Amazon Web Services).

Εκτός από την όλη ιδέα να είναι ελαφρύ και να ξεκινά γρήγορα, το Visual Studio Code έχει συμπλήρωση κώδικα IntelliSense για μεταβλητές, μεθόδους και εισαγόμενες μονάδες. γραφικό εντοπισμό σφαλμάτων, συμβουλές παραμέτρων και άλλες ισχυρές λειτουργίες επεξεργασίας. Διαθέτει έξυπνη πλοήγηση και ανακατασκευή κώδικα. και ενσωματωμένος έλεγχος πηγαίου κώδικα συμπεριλαμβανομένης της υποστήριξης Git.

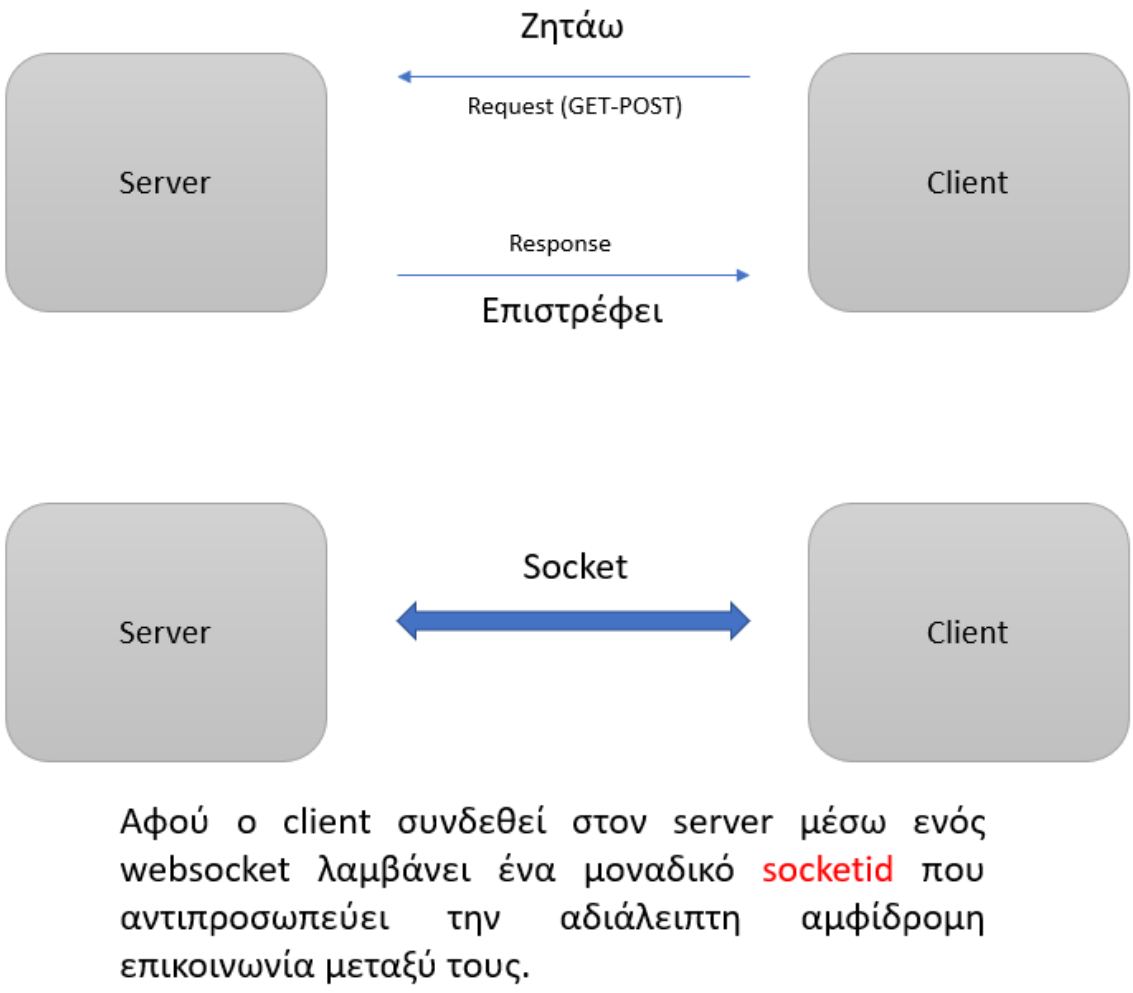
Ο κώδικας του Visual Studio δημιουργείται χρησιμοποιώντας το κέλυφος Electron, το Node.js, το TypeScript και το Πρωτόκολλο Διακομιστή Γλώσσας και ενημερώνεται σε μηνιαία βάση. Ο πλούτος της υποστήριξης ποικίλλει μεταξύ των διαφορετικών γλωσσών προγραμματισμού και των επεκτάσεών τους, από απλή επισήμανση σύνταξης και αντιστοίχιση αγκυλών έως εντοπισμό σφαλμάτων και ανακατασκευή. Μπορείτε να προσθέσετε βασική υποστήριξη για την αγαπημένη σας γλώσσα.

Ο κώδικας στο αποθετήριο του Visual Studio Code είναι ανοιχτού κώδικα υπό την άδεια MIT. Το προϊόν Visual Studio Code αποστέλλεται με μια τυπική άδεια προϊόντος της Microsoft, καθώς έχει ένα μικρό ποσοστό προσαρμογών για τη Microsoft. Είναι δωρεάν παρά την εμπορική άδεια.

Κεφάλαιο 4ο: Οι εφαρμογές με Websockets για IoT

Το REST είναι ένα αρχιτεκτονικό στυλ λογισμικού - ίσως το πιο διαδεδομένο και ευρέως χρησιμοποιούμενο. Σε αντίθεση με το WebSocket, το REST δεν είναι ένα πρωτόκολλο, αλλά ένα σύνολο αρχιτεκτονικών περιορισμών.

Το WebSocket είναι ένα πρωτόκολλο σε πραγματικό χρόνο που επιτρέπει την πλήρη αμφίδρομη, αμφίδρομη επικοινωνία μεταξύ ενός προγράμματος-πελάτη ιστού και ενός διακομιστή ιστού μέσω μιας μόνιμης, μίας υποδοχής σύνδεσης.



Εικόνα 4.1: Σύγκριση Rest και WebSocket

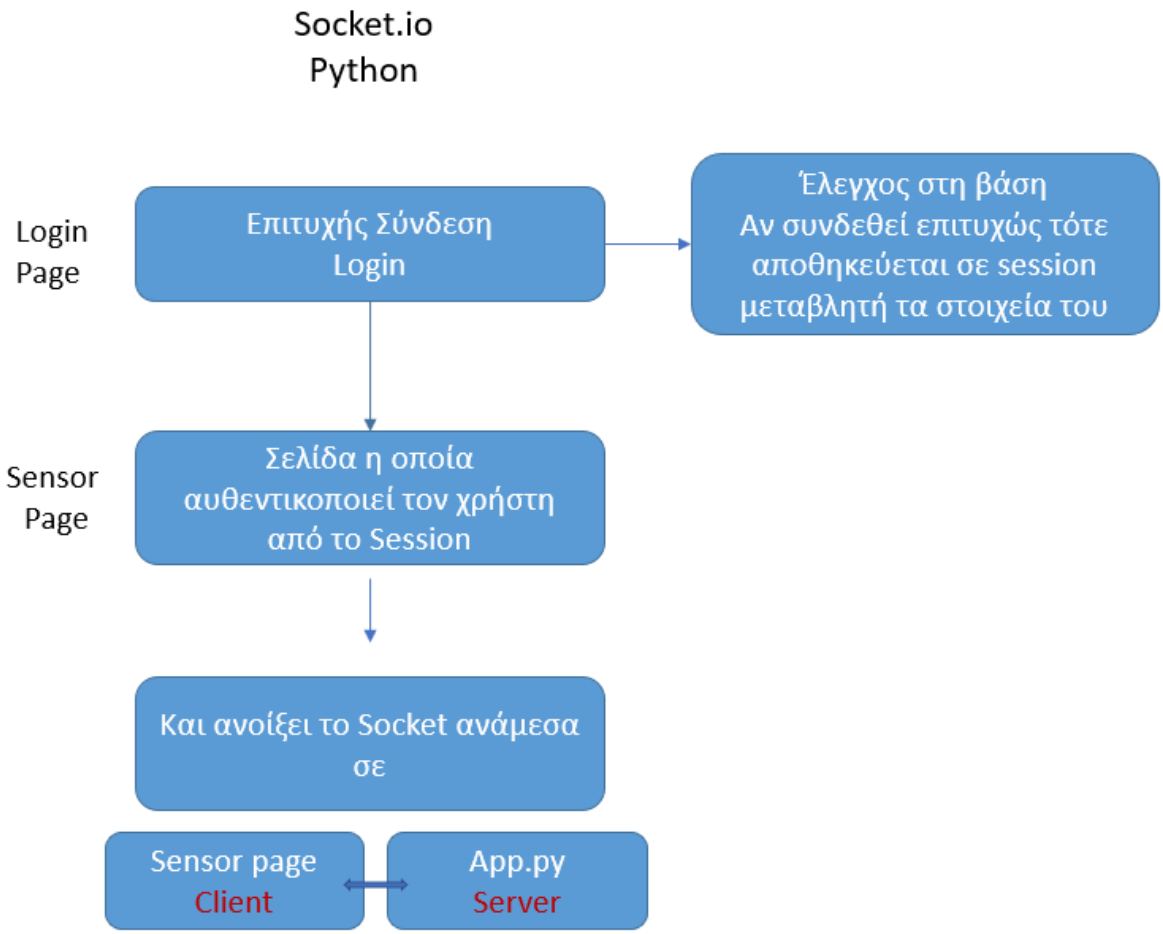
Παρόμοια με το HTTP, το WebSocket λειτουργεί πάνω από το TCP. Σε αντίθεση με το HTTP, τα WebSockets είναι stateful. Είναι κατάλληλα για εφαρμογές και υπηρεσίες που βασίζονται σε συμβάντα που απαιτούν επικοινωνία υψηλής συχνότητας και χαμηλής καθυστέρησης μεταξύ πελάτη και

διακομιστή. Οι περιπτώσεις χρήσης περιλαμβάνουν ζωντανή συνομιλία, πίνακες εργαλείων σε πραγματικό χρόνο, ενημερώσεις ζωντανών αποτελεσμάτων ροής, συνεργασία εγγράφων πολλών χρηστών, παιχνίδια για πολλούς παίκτες και πολλά άλλα.

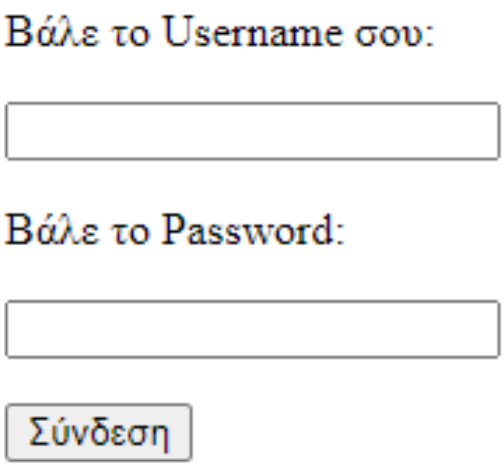
Το WebSocket API επεκτείνει το πρωτόκολλο WebSocket σε πελάτες web. Σχεδόν όλα τα προγράμματα περιήγησης υποστηρίζουν εγγενώς το WebSocket API, καθιστώντας εύκολη και εύκολη την έναρξη δημιουργίας εφαρμογών σε πραγματικό χρόνο που βασίζονται σε πρόγραμμα περιήγησης με τα WebSockets.

4.1 Με Socket.io και python

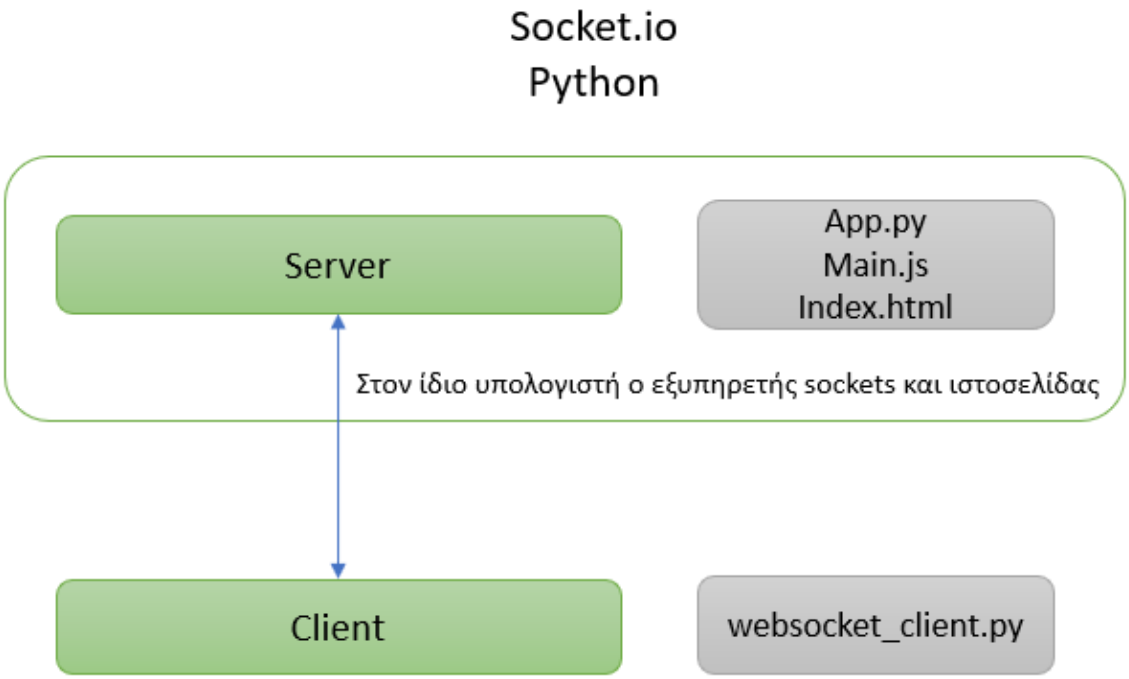
Υλοποιήσαμε με socket.io μια επικοινωνία ενός server που εξυπηρετεί και την ιστοσελίδα μας και ένα client που βρίσκεται απομακρυσμένα.



Εικόνα 4.2: WebSockets με socket.io

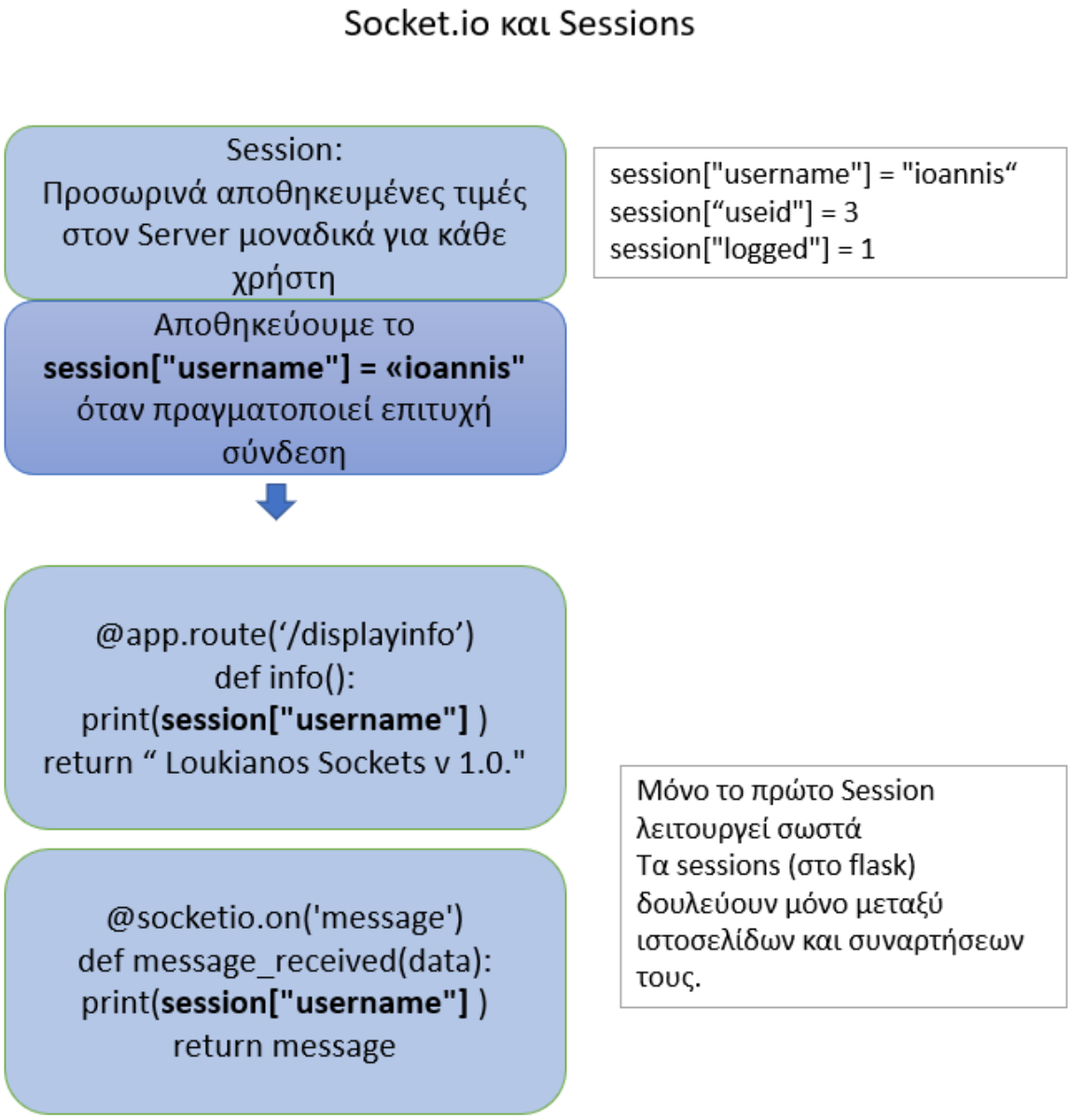


Εικόνα 4.3: Φόρμα σύνδεσης

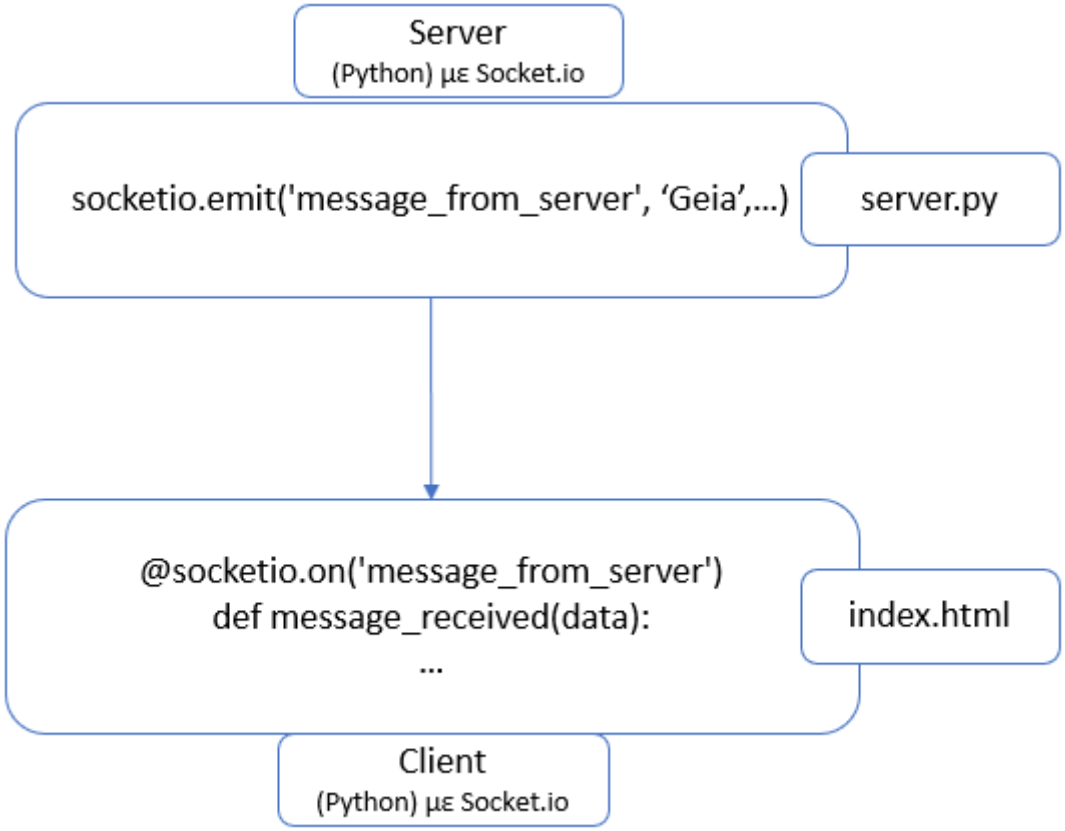


Εικόνα 4.4: socket.io επικοινωνία

Όπως περιγράφει τη διαδικασία η Εικόνα 4.2 χρησιμοποιούμε sessions και αυτά ισχύουν μόνο ανάμεσα στις μεθόδους της εφαρμογής. Στην Εικόνα 4.4. βλέπουμε η κάθε πλευρά τι χρησιμοποιεί.

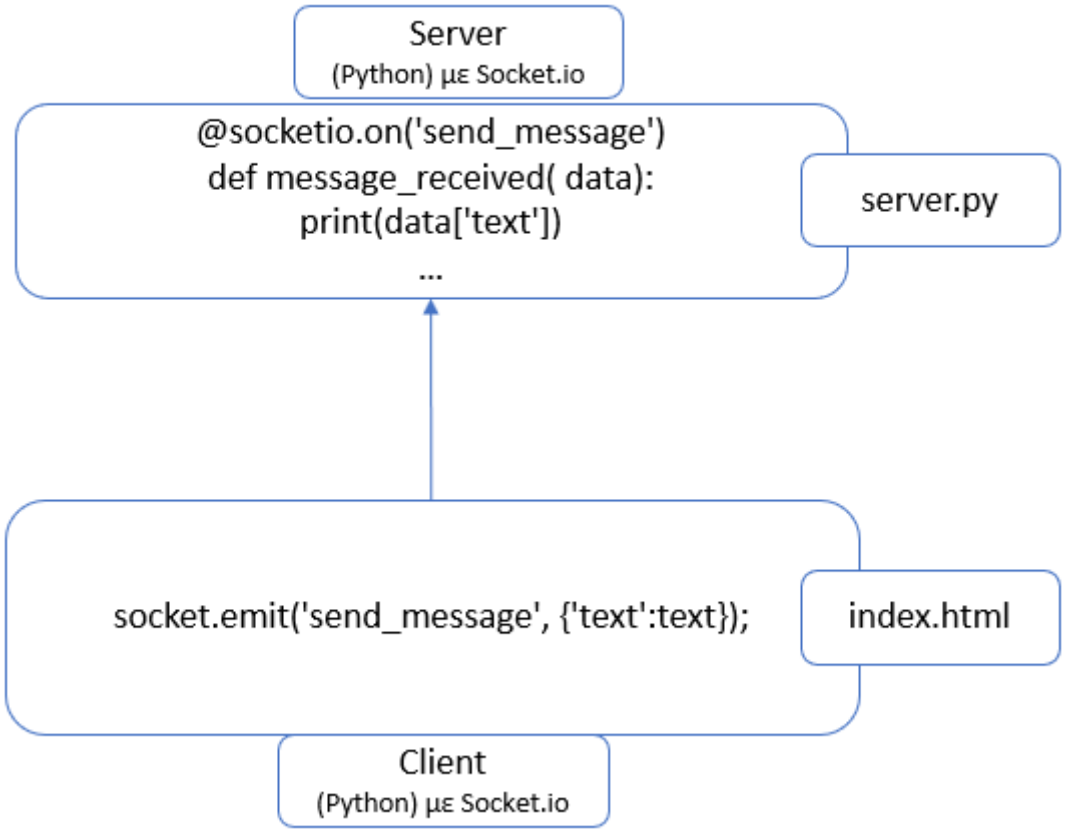


Εικόνα 4.5: Sessions και sockets στο έργο μας στο socket.io



Εικόνα 4.6: Αποστολή από Server και αποδοχή από τον client στη σελίδα του – socket.io

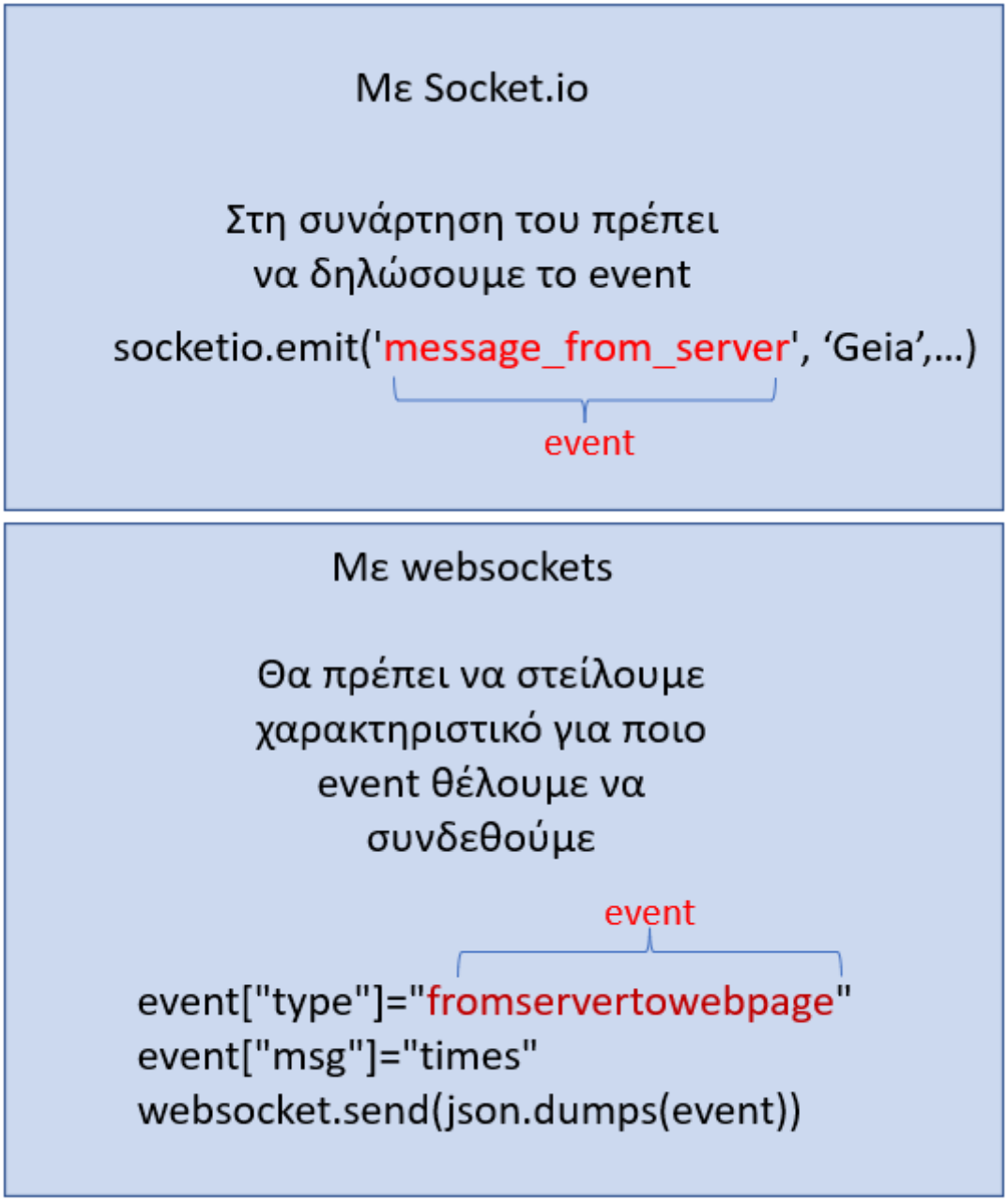
Στην Εικόνα 4.6 παρουσιάζεται η αποστολή από Server και αποδοχή από τον client στη σελίδα του.



Εικόνα 4.7: Αποστολή από client από σελίδα του και αποδοχή από τον Server – socket.io

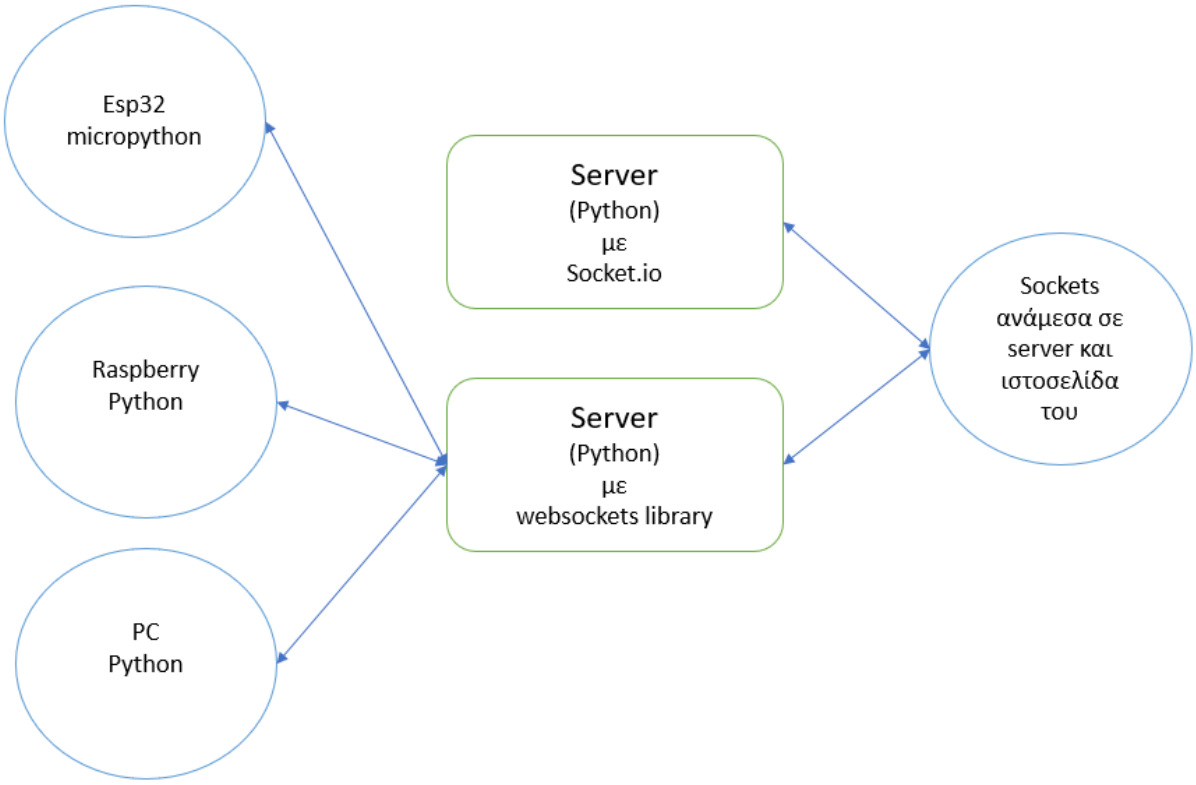
Στην Εικόνα 4.7 παρουσιάζεται η αποστολή από client από σελίδα του και αποδοχή από τον Server.

4.2 Με websockets



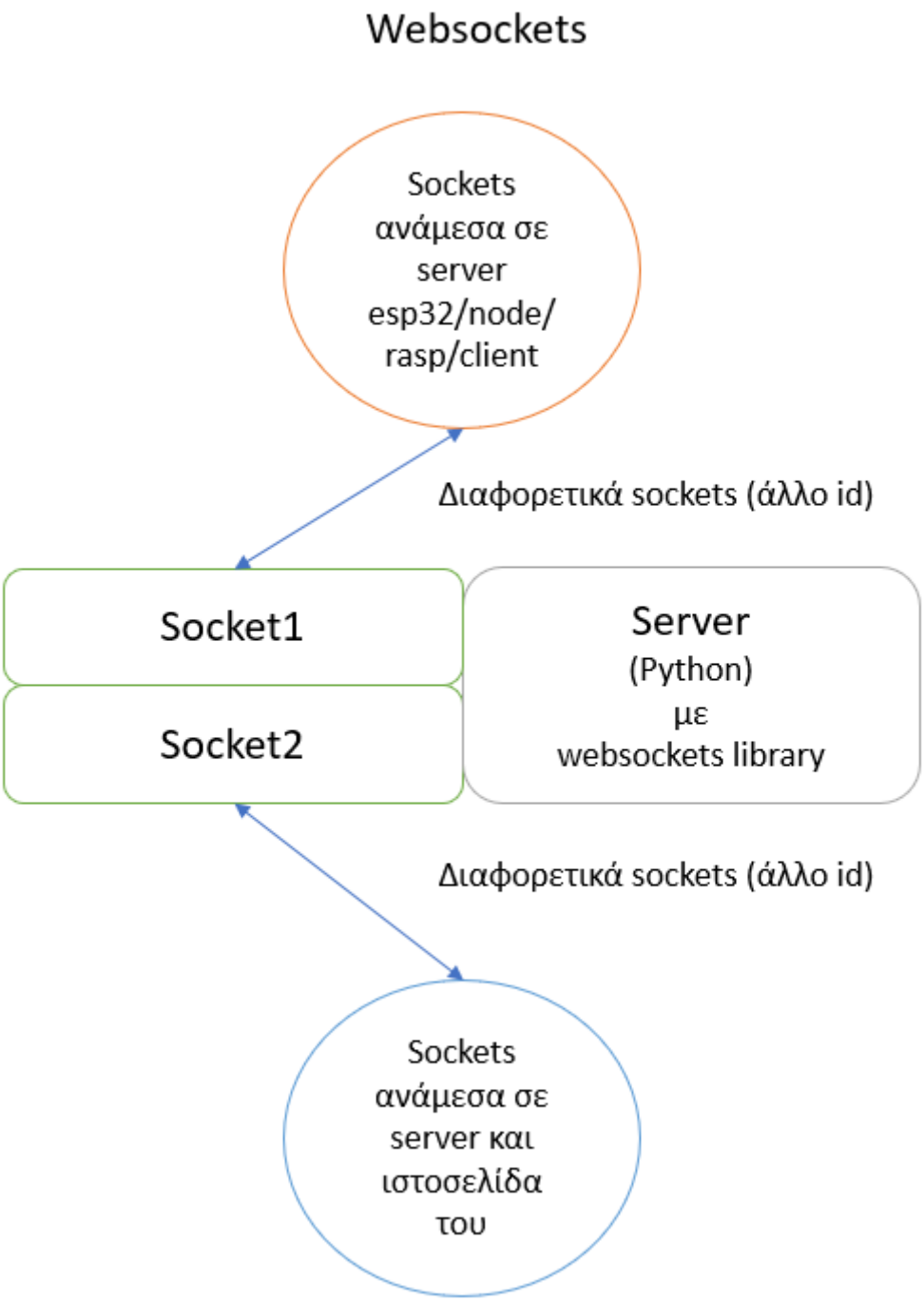
Εικόνα 4.8: Διαφορές WebSockets βιβλιοθήκη και socket.io

Στην ενότητα αυτή θα δούμε τις κύριες λειτουργίες της χρήσης με WebSockets βιβλιοθήκη. Στην Εικόνα 4.8 παρουσιάζονται Διαφορές WebSockets βιβλιοθήκη και socket.io.

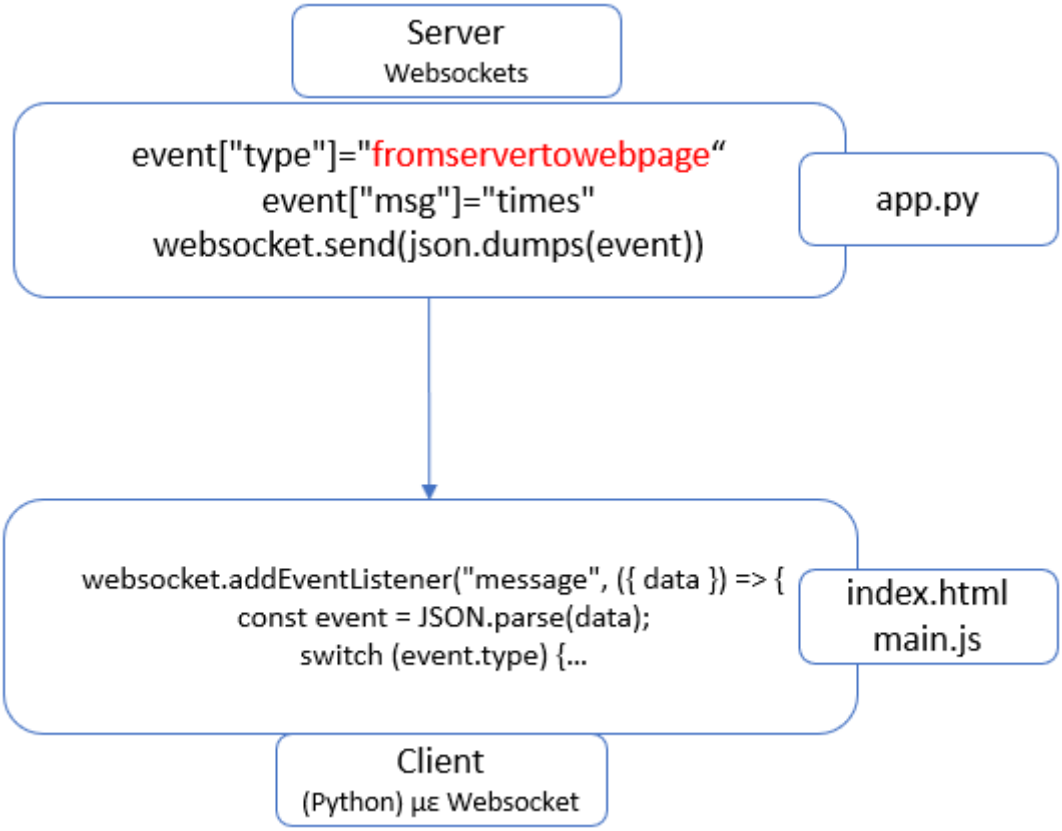


Εικόνα 4.9: Δυνατότητες με Websockets βιβλιοθήκη και socket.io

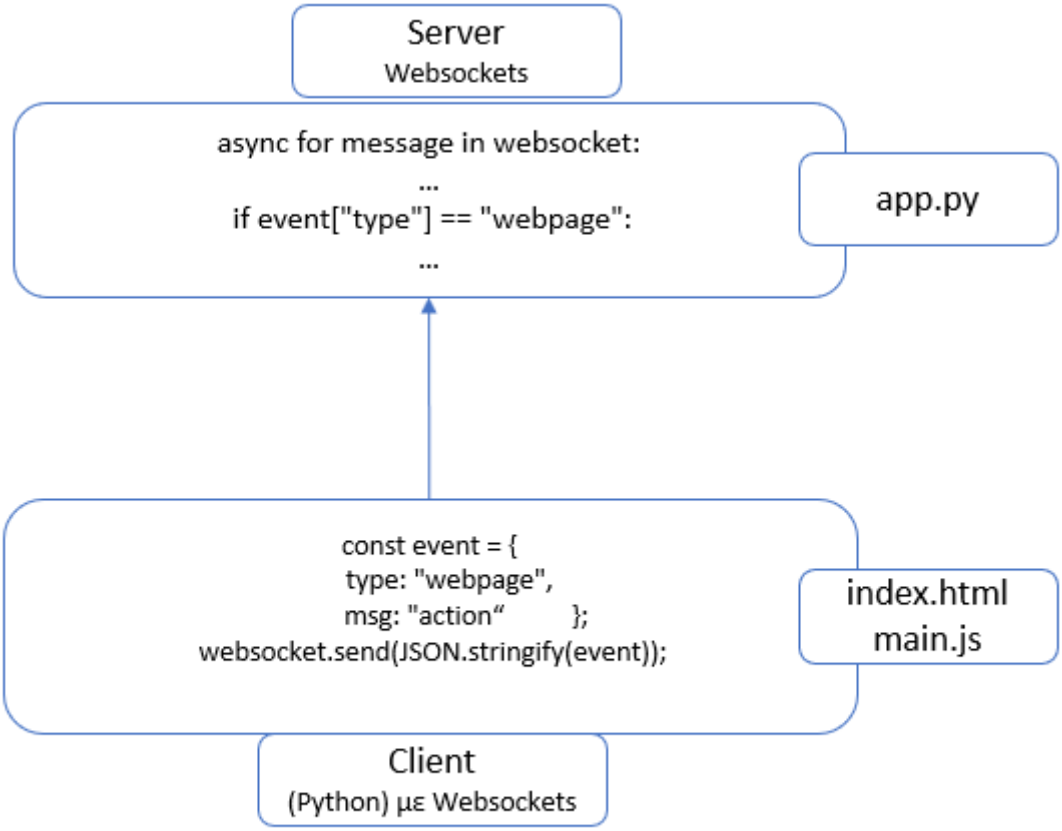
Όπως φαίνεται και στην Εικόνα 4.10 το socket.io δεν μπορεί να επικοινωνήσει εύκολα με άλλους clients Που βρίσκονται σε άλλα συστήματα που χρησιμοποιούν websockets.



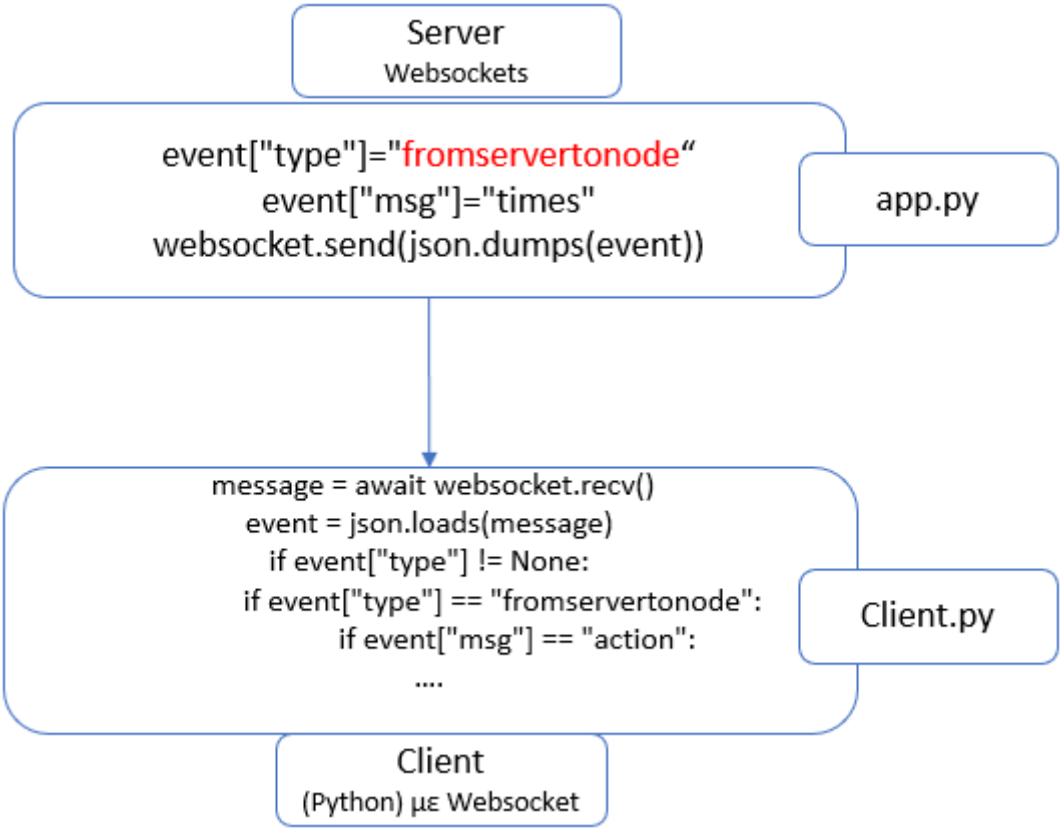
Εικόνα 4.10: Κύρια λειτουργία με Websockets βιβλιοθήκη



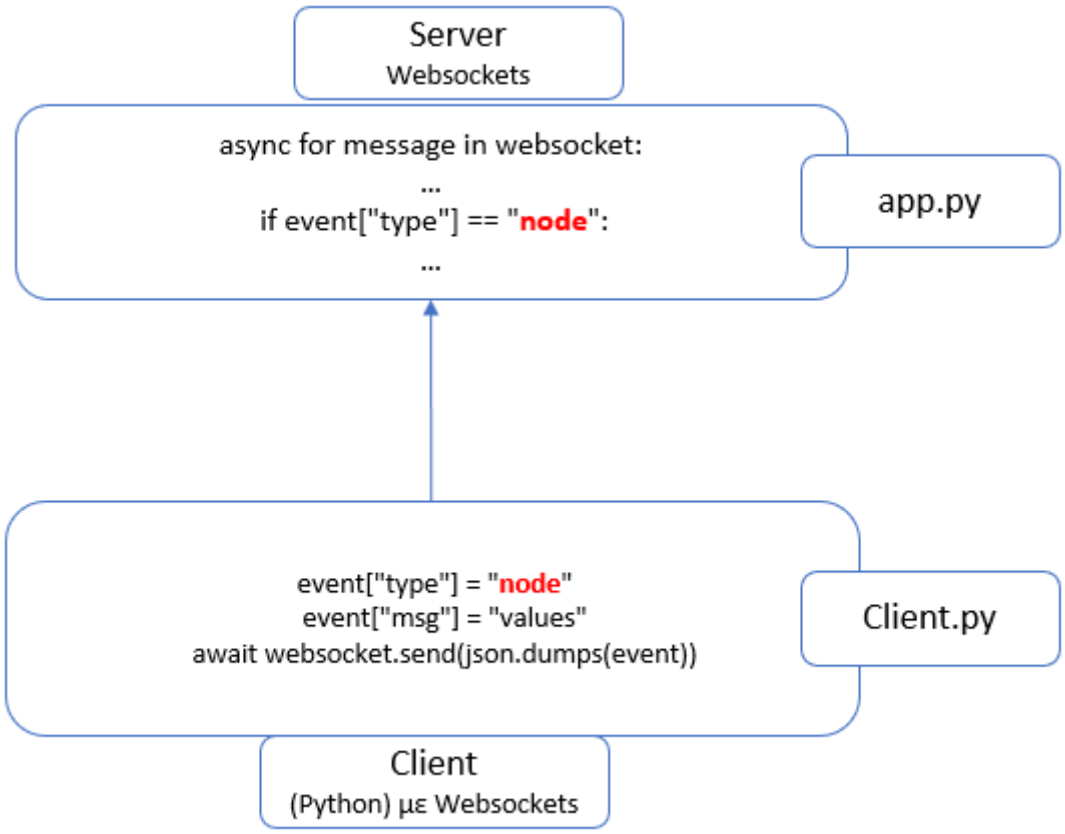
Εικόνα 4.11: Αποστολή από Server και αποδοχή από τον client στη σελίδα του - websockets



Εικόνα 4.12: Αποστολή από client από σελίδα του και αποδοχή από τον Server - websockets

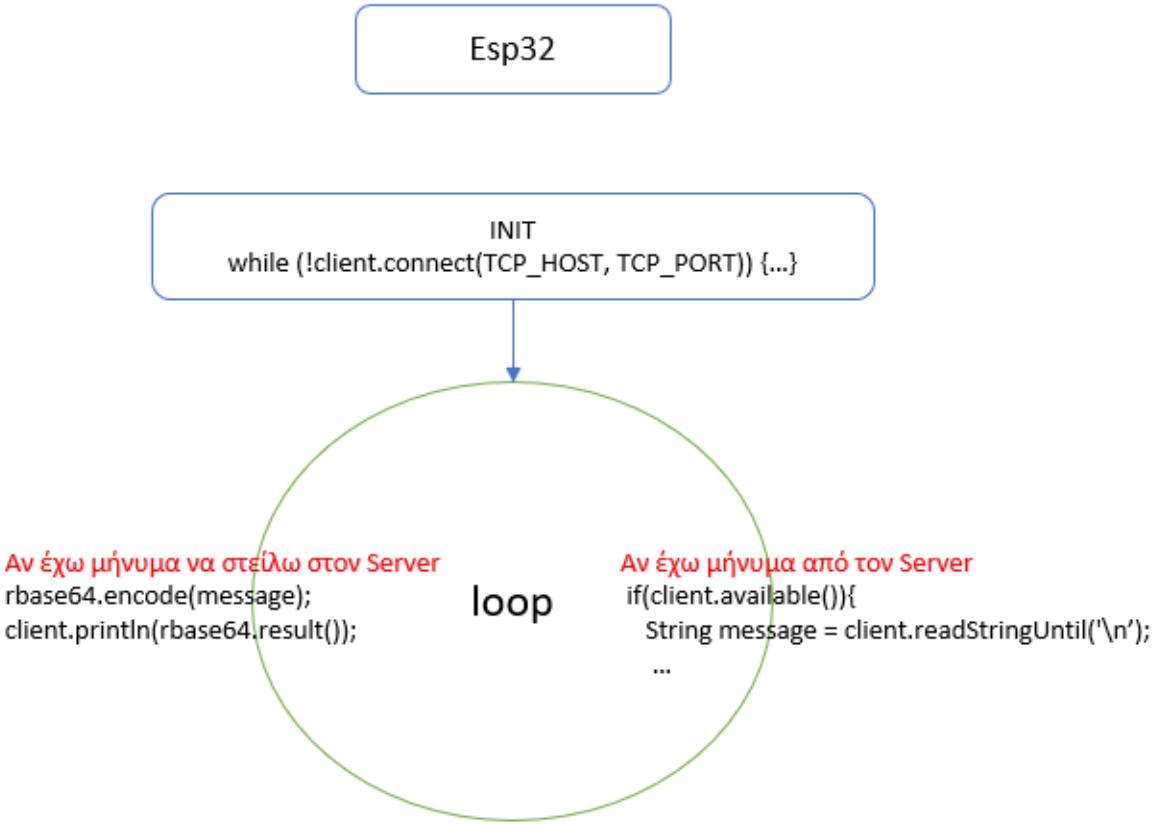


Εικόνα 4.13: Αποστολή από Server και αποδοχή από τον client - websockets



Εικόνα 4.14: Αποστολή από client του και αποδοχή από τον Server - websockets

4.3 Σε esp32



Εικόνα 4.15: Αποστολή από client esp32 μέσω sockets

Στο παρακάτω κώδικα εκτέλειται αυτό που πειγράφει η Εικόνα 4.14, την αποστολή από client esp32 μέσω sockets στον Server.

Ορίζουμε διευθύνσεις

```
WiFiClient client;

const char* TCP_HOST = "192.168.2.6";

const uint16_t TCP_PORT = 7000;
```

Πρώτη ρύθμιση

```
void setup() {
…
  startTCP();
}
```

Αυτό τρέχει συνέχεια

```
void loop() {
  handleTCP();
}
```

```
void startTCP() {
  Serial.println(TCP_HOST);
  while (!client.connect(TCP_HOST, TCP_PORT)) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("TCP connected");
  StaticJsonDocument<128> doc;
  doc["sensor"] = "gps";
  doc["time"] = 1351824120;
  char message[128];
  serializeJson(doc, message);
  sendMessageTCP(message);
}
```

```
void handleTCP(){
  if(client.available()){
    String message = client.readStringUntil('\n');
    if(message != ""){
      receiveMessageTCP(message);
    }
  }else{
    // startTCP();
  }
}
```

και οι μέθοδοι αποστολής και λήψης

```
void sendMessageTCP(String message){
    rbase64.encode(message);
    client.println(rbase64.result());
}


void receiveMessageTCP(String message){
  rbase64.decode(message);
  String jsonString = rbase64.result();
  StaticJsonDocument<128> doc;
  deserializeJson(doc, jsonString);
  Serial.println(rbase64.result());
}
```

Κεφάλαιο 5ο: Συμπεράσματα

Σε αυτήν την εργασία παρουσιάστηκε και χρησιμοποιήθηκε το socket.io και το WebSocket library για να μελετηθεί για χρήση σε IoT εφαρμογές.

Πραγματοποιήθηκε μελέτη και σχεδίαση συστήματος διαδικτύου των πραγμάτων με websockets σε γλώσσα python. Αρχικά μελετήθηκε και υλοποιήθηκε σύστημα με χρήση της δημοφιλούς βιβλιοθήκης socket.io. Χρησιμοποιήθηκαν sessions και sockets και αναλύθηκε ο τρόπος χρήσης τους και οι δυνατότητες τους με αυτήν τη βιβλιοθήκη. Στη συνέχεια χρησιμοποιήθηκαν η ειδική βιβλιοθήκη websockets που χρησιμοποιεί τα TCP sockets και παρουσιάστηκε η σύγκριση των δύο επιλογών.

Τα αποτελέσματα έδειξαν ότι για να χρησιμοποιηθεί σαν client ένα esp32, raspberry ή γενικά ένα μικρουπολογιστικό σύστημα που να μπορεί να επικοινωνήσει αμφίδρομα με τον server θα χρειαστεί να χρησιμοποιήσει την websocket βιβλιοθήκη,

ΒΙΒΛΙΟΓΡΑΦΙΑ

- Fette, I. and Melnikov, A., 2011. *The websocket protocol* (No. rfc6455).
- Wessels, A., Purvis, M., Jackson, J. and Rahman, S., 2011, April. Remote data visualization through websockets. In *2011 Eighth international conference on information technology: new generations* (pp. 1050-1051). IEEE.
- Pimentel, V. and Nickerson, B.G., 2012. Communicating and displaying real-time data with websocket. *IEEE Internet Computing*, *16*(4), pp.45-53.
- Lombardi, A., 2015. *WebSocket: lightweight client-server communications*. " O'Reilly Media, Inc.".
- Wang, V., Salim, F. and Moskovits, P., 2013. *The definitive guide to HTML5 WebSocket* (Vol. 1). New York: Apress.
- Rai, R., 2013. *Socket. IO Real-time Web Application Development*. Packt Publishing Ltd.
- Cadenhead, T., 2015. *Socket. IO Cookbook*. Packt Publishing Ltd.
- Mardan, A. and Mardan, A., 2018. Real-Time Apps with WebSocket, Socket. IO, and DerbyJS. *Practical Node. js: Building Real-World Scalable Web Apps*, pp.307-330.
- https://socket.io/
- https://python-socketio.readthedocs.io/en/latest/
- https://flask-socketio.readthedocs.io/en/latest/getting_started.html#broadcasting
- https://flask-session.readthedocs.io/en/latest/
- https://pypi.org/project/websockets/
- https://websockets.readthedocs.io/en/stable/

ΠΑΡΑΡΤΗΜΑ Α

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

Esp32

```
#include "ArduinoJson.h"
#include <rBase64.h>

WiFiClient client;

const char* WIFI_SSID = "";
const char* WIFI_PASSWORD = "";

const char* TCP_HOST = "192.168.2.6";
const uint16_t TCP_PORT = 7000;



/*_____________________________________________________SETUP_____________
_____________________________________________*/

void setup() {
  Serial.begin(115200);
  delay(10);
  Serial.println("\r\n");

  startWiFi();
  startTCP();
}

/*_____________________________________________________LOOP_____________
_____________________________________________*/

void loop() {
```

```
  Serial.println();
  handleTCP();
}

/*_____SETUP_FUNCTIONS_____
_____*/

void startWiFi() {
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(WIFI_SSID);

  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void startTCP() {
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(TCP_HOST);
```

```
  while (!client.connect(TCP_HOST, TCP_PORT)) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("TCP connected");
  StaticJsonDocument<128> doc;
  doc["sensor"] = "gps";
  doc["time"] = 1351824120;
  char message[128];
  serializeJson(doc, message);
  sendMessageTCP(message);
}

void handleTCP(){
  if(client.available()){
    String message = client.readStringUntil('\n');
    if(message != ""){
      receiveMessageTCP(message);
    }
  }else{
    // startTCP();
  }
}

void sendMessageTCP(String message){
    rbase64.encode(message);
    client.println(rbase64.result());
}

void receiveMessageTCP(String message){
```

```
  rbase64.decode(message);
  String jsonString = rbase64.result();
  StaticJsonDocument<128> doc;
  deserializeJson(doc, jsonString);
  Serial.println(rbase64.result());
}
```

Server για Websockets

```
import asyncio
import itertools
import json

import websockets

connected =[]

async def handler(websocket):

    print(websocket)
    connected.append(websocket)

    async for message in websocket:
        print("get message")
        #Παίρνω μηνύματα από τους Κόμβους-nodes-sensors
        event = json.loads(message)

        if event["type"] != None:
            if event["type"] == "node":
                print("get message from node")
                #Αποθηκεύω το μήνυμα απο τον κόμβο-node
                msg = event["msg"]
                #Ελέγχω ...
```

```
            #Αποθηκεύω στη βάση

            #Στέλνω socket μήνυμα στην ιστοσελίδα για ενημέρωση
            event["type"]="fromservertowebpage"
            event["msg"]="times ktl"
            await websocket.send(json.dumps(event))

        if event["type"] == "webpage":
            print("get message from webpage")

            #Αποθηκεύω το μήνυμα απο την Ιστοσελίδα
            msg = event["msg"]

            #Στέλνω socket μήνυμα στον κόμβο από την ιστοσελίδα
            event["type"]="fromservertonode"
            event["msg"]="action"
            event["action"]="togglerele"
            #await websocket.send(json.dumps(event))
            for connection in connected:
                #print(connection)
                try:
                    await connection.send(json.dumps(event))
                except websockets.ConnectionClosed as exc:
                    connected.remove(connection)
            print("and send message to node")

async def main():
    async with websockets.serve(handler, "", 8001):
        await asyncio.Future()  # run forever



if __name__ == "__main__":
```

```
    asyncio.run(main())
```

client.py για Websockets

```
import asyncio
import websockets
import json

#Στέλνουμε τις τιμές ενός Αισθητήρα ή κάτι άλλο
#περιοδικά ή όποτε θέλουμε εμείς καλώντας αυτήν την συνάρτηση
async def getMeasurementAndSendToServer(websocket):

    #Εδώ θα μπορούσε να μετράει απο τον ADC ή από κάποιον αισθητήρα
    print("Send measurements to Server")
    event = {}
    event["type"] = "node"
    event["msg"] = "values"
    await websocket.send(json.dumps(event))



async def handlerNode():
    async with websockets.connect("ws://localhost:8001") as websocket:
        while True:
            message = await websocket.recv()
            #Παίρνω μηνύματα από τον Server

            event = json.loads(message)
            if event["type"] != None:
                if event["type"] == "fromservertonode":
                    if event["msg"] == "action":
                        if event["action"] == "togglerele":
                            #Εκτελούμε μια ενέργεια, όπως ενεργοποίηση ενός pin
                            #για on ή off μιας συσκευής μέσω ρελέ, όπως ανεμιστήρα ή λάμπα
```

```
                    print("Toggle a Rele")

                    #Θα μπορούσαμε να στείλουμε
                    await getMeasurementAndSendToServer(websocket)
                if event["action"] == "changesetting":
                    #Να αλλάξει μια ρύθμιση στον Client (πχ Raspberry,esp32 κτλ)
                    print("changesetting")


asyncio.run(handlerNode())
```

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>iee.ihu Websockets</title>
        <style>
        .button {
          background-color: #4CAF50; /* Green */
          border: none;
          color: white;
          padding: 15px 32px;
          text-align: center;
          text-decoration: none;
          display: inline-block;
          font-size: 16px;
          margin: 4px 2px;
          cursor: pointer;
        }
```

```
        .button2 {background-color: #008CBA;} /* Blue */
        .button3 {background-color: #f44336;} /* Red */
        .button4 {background-color: #e7e7e7; color: black;} /* Gray */
        .button5 {background-color: #555555;} /* Black */
        </style>
  </head>
  <body>

    <h4> Websockets</h4>
        <button class="board button button2">Send Action to Node</button>
    <script src="main.js"></script>
  </body>
</html>
```