Ανάπτυξη Λογισμικού για Δίκτυα και Τηλεπικοινωνίες

Φεβρουάριος 2023

Δημιουργοί:

Περάκης Πολυχρόνης	1115201400336
Καπατσούλιας Θεόδωρος	1115201600056
Κολοκύθας Χρήστος	1115201700052
Ψαρούδης Γιώργος	1115201700198
Ντέμου Σταυρούλα-Ευσταθία	1115201700295

Παρουσίαση Android Εφαρμογών

---

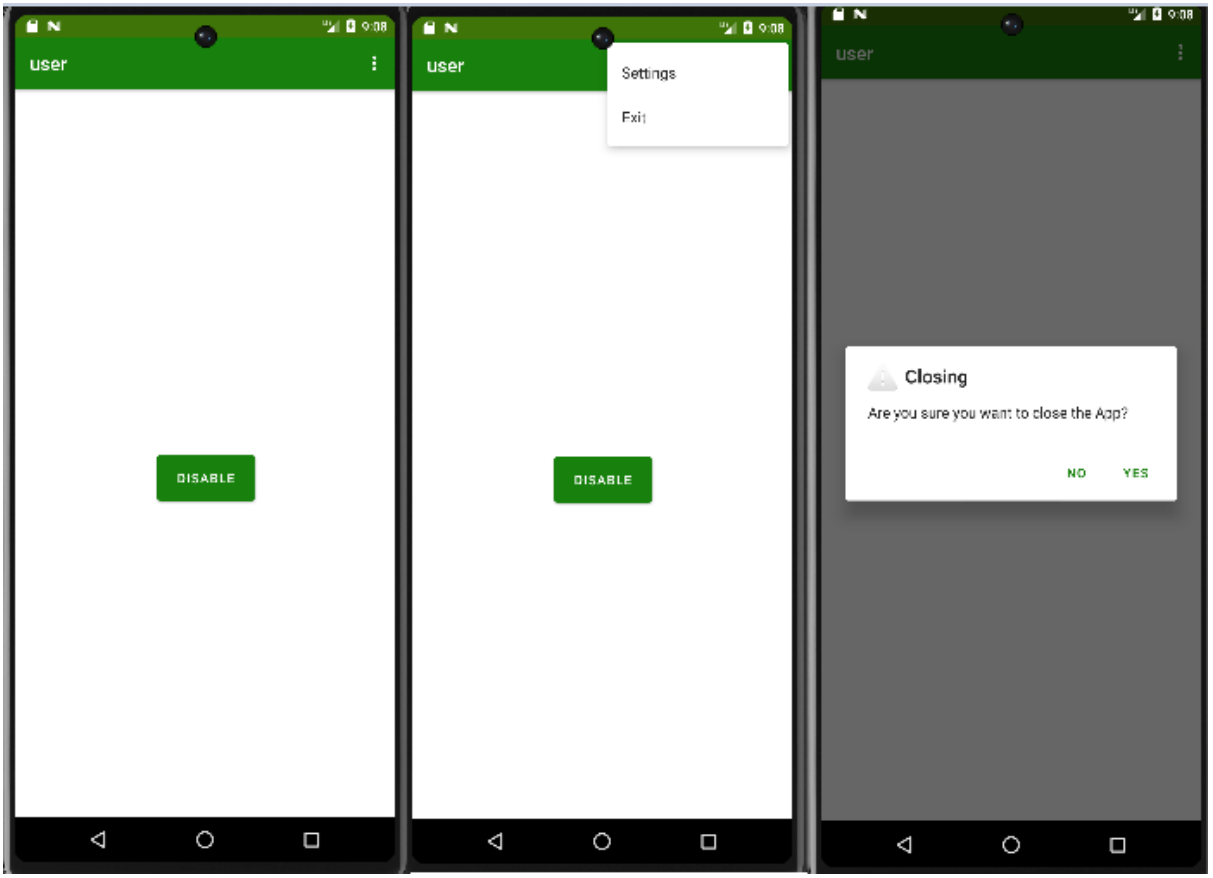
1. **Εφαρμογή χρήστη**

Πρόκειται για την εφαρμογή που θα έχει εγκατεστημένη στο κινητό του ο εκάστοτε χρήστης που θέλει να ενημερωθεί για τη περίπτωση ανίχνευσης κάποιου κινδύνου. Με το άνοιγμά της ο χρήστης συνδέεται με τον MQTT server και λαμβάνει οπτικοακουστικής μορφής ειδοποιήσεις για τυχόν κινδύνους.

Αρχική Οθόνη

Με το άνοιγμα της εφαρμογής ο χρήστης αντικρίζει την κύρια οθόνη της εφαρμογής στην οποία συναντά τις εξής επιλογές:

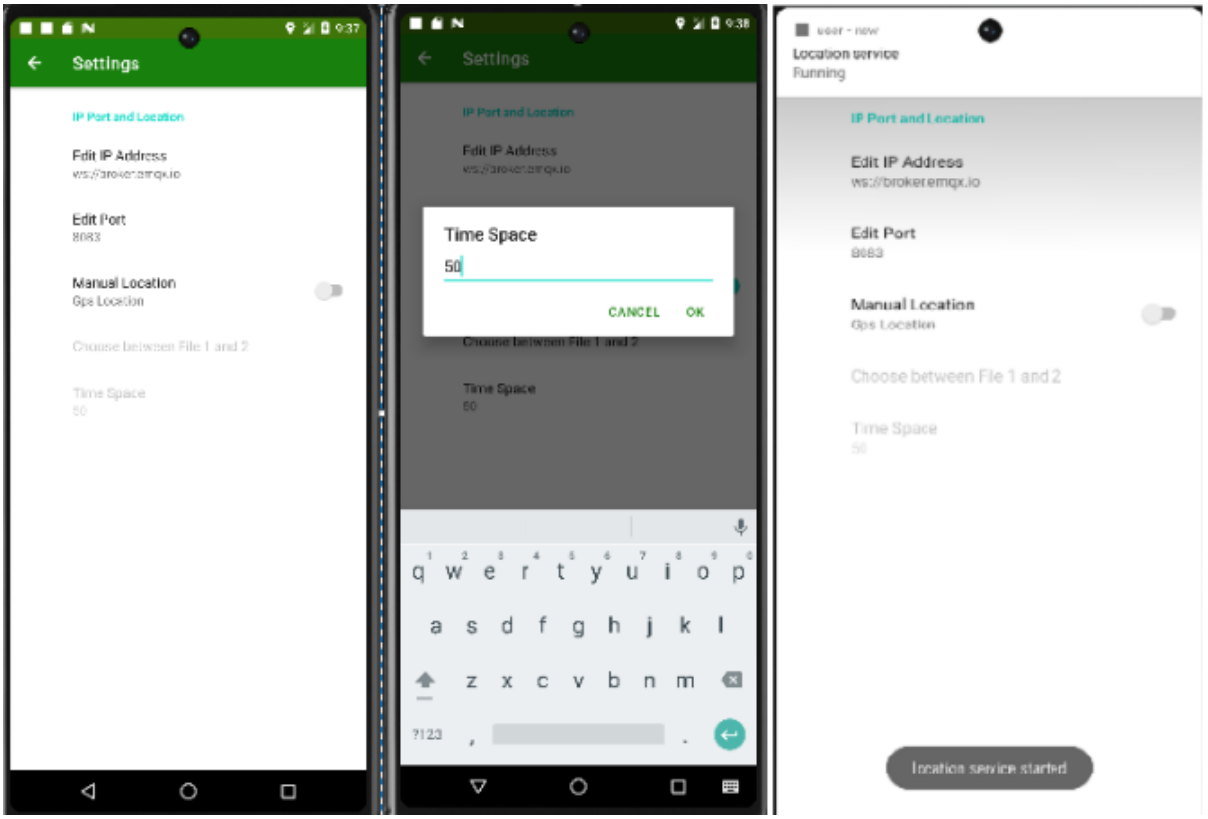
1. Ένα κουμπί **Enable/Disable** το οποίο σταματάει άμεσα την αποστολή μετρήσεων προς τον server.
2. Τις **3 τελείες** οι οποίες περιέχουν 2 επιλογές:
   I. Μετάβαση στα **Settings**.
   II. ‘Εξοδο απο την εφαρμογή. Για την έξοδο από την εφαρμογή εμφανίζεται ένα αναδυόμενο παράθυρο το οποιο ζητάει την επιβεβαιωση του χρηστη για οριστικη διακοπη της εφαρμογης.



Settings menu

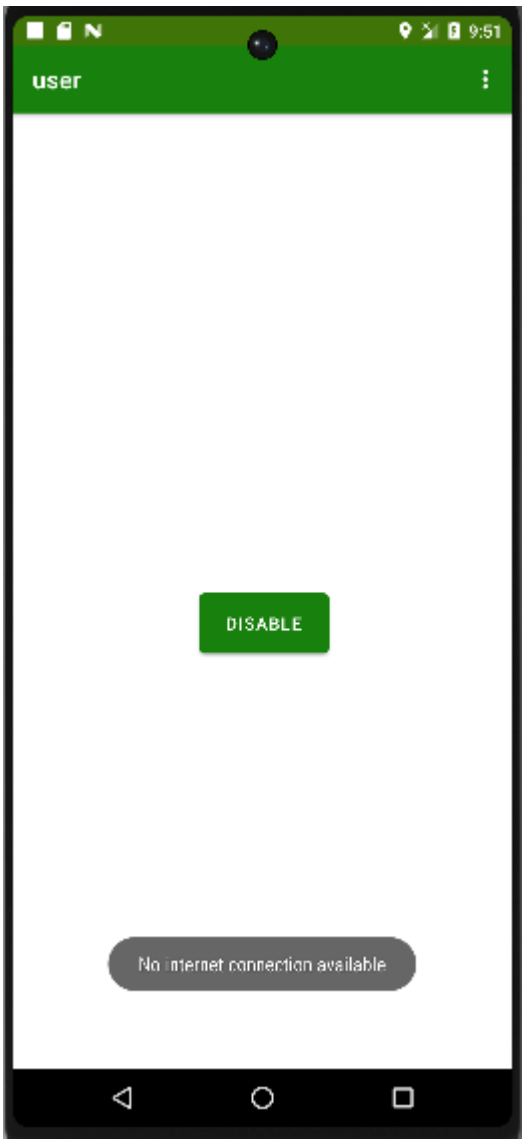
Στο **Settings menu** έχουμε τις εξής επιλογές:

1. **Edit IP Address**: Στο πεδίο αυτό εισάγουμε την IP για την επικοινωνία με τον Edge Server.
2. **Port:** Στο πεδίο αυτό εισάγουμε το Port για την επικοινωνία με τον Edge Server.
3. **Manual Location**: Το switch αυτό χρησιμοποιείται για την εναλλαγή τρόπου λήψης τοποθεσίας μεταξύ της χρήσης GPS και χειροκίνητης τοποθεσίας. Σε περίπτωση που επιλεχθεί η χειροκίνητη τοποθεσία υπάρχουν 2 ακόμα Settings.
    a. **Choose File**:Επιλέγει μεταξύ των 2 αρχείων που δίνονται απο την εκφώνηση.
    b. **TimeSpace**: Επιλέγει το χρονικό διάστημα για το οποίο γίνεται αποστολή των διανυσμάτων των μετρήσεων.



Έλεγχος σύνδεσης

Κάθε 10 δευτερόλεπτα η εφαρμογή ελέγχει μέσω του **ConnectivityManager** εάν είναι συνδεδεμένη στο internet. Σε περίπτωση που εντοπίσει απουσία σύνδεσης εμφανίζει αντίστοιχη ειδοποίηση.



Λήψη τοποθεσίας

Όπως αναφέρθηκε προηγουμένως η λήψη της τοποθεσίας γίνεται με δύο τρόπους εναν αυτοματο και εναν χειροκινητο.

Όταν έχει επιλεχθεί ο αυτόματος τρόπος χρησιμοποιείται η υπηρεσία **FusedLocationProviderClient** για τη λήψη της πραγματικής τοποθεσίας της συσκευής του χρήστη.

Σε περίπτωση επιλογης χειροκίνητης τοποθεσίας η εφαρμογή διαβάζει τα **XML αρχεία** χρησιμοποιώντας **parse** της εκφώνησης και αποθηκεύει τα διανύσματα που περιέχουν σε αντίστοιχους πίνακες,απο τους οποίους αντλεί τα δεδομένα που στέλνει στον **MQTT Server**.

Επικοινωνία με το Server

Η εφαρμογή χρήστη επικοινωνεί με τον **MQTT Server** προκειμένου να λαμβάνει ειδοποιήσεις και να στέλνει δεδομένα μέσω δύο αντίστοιχων **topics**.

- Για να λάβει ειδοποίηση κινδύνου από τον server κάνει **subscribe** στο **topic "**notifications**"**. Από το topic αυτό λαμβάνει ένα **json** αρχείο αντλεί το επίπεδο του κινδύνου καθώς και την απόσταση από το σημείο που εκδηλώθηκε ο κίνδυνος.
- Για να στείλει δεδομένα στον server η εφαρμογή κάνει **subscribe** στο **topic** "android3". Στο topic αυτό στέλνει ένα json αρχείο που περιέχει το το διάνυσμα της τοποθεσίας και το **deviceID**. (Το deviceID είναι ορισμένο ως 3 για να γνωρίζουμε ότι πρόκειται για την android συσκευή).

2. **Εφαρμογή IoT Αισθητήρων**

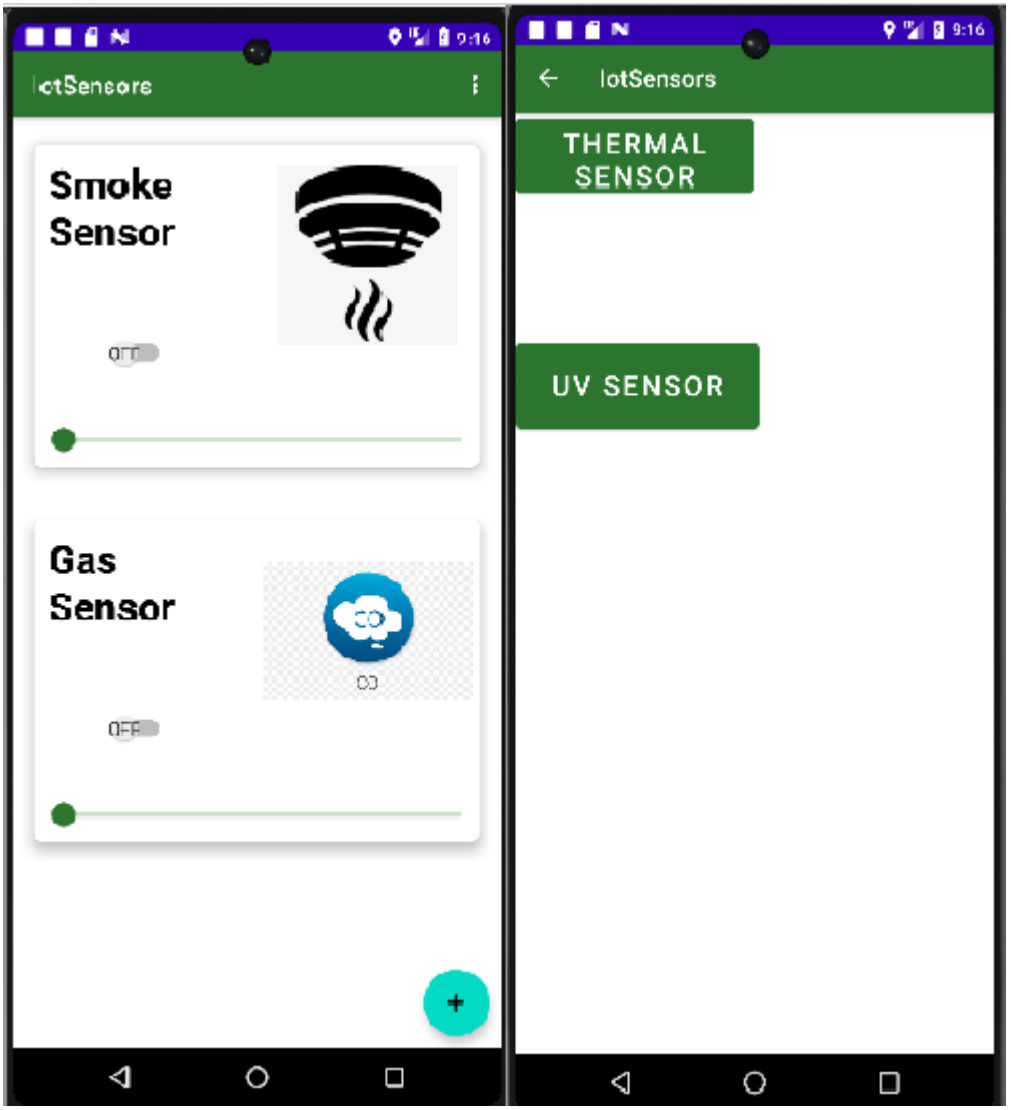
Πρόκειται για την εφαρμογή που περιέχει τις προσομοιώσεις των τεσσάρων αισθητήρων που ζητούνται από την εκφώνηση. Η εφαρμογή αυτή παρουσιάζει πολλές ομοιότητες με την εφαρμογή user στον τρόπο λειτουργίας του μενού των **settings** καθώς και τον τρόπο με τον οποίο γίνεται η λήψη της τοποθεσίας μέσω **GPS.** Γι' αυτό το λόγο παρακάτω θα αναλυθούν μόνο τα σημεία που διαφέρουν οι εφαρμογές μεταξύ τους.

Αρχική Οθόνη

Κατά το άνοιγμα της εφαρμογής συναντάμε προεγκατεστημένους τους αισθητήρες καπνού και αερίου. Στο κάτω δεξιό μέρος της οθόνης βρίσκεται ένα **floating button** το οποίο μας επιτρέπει να εγκαταστήσουμε τους υπόλοιπους αισθητήρες θερμότητας και UV ακτινοβολίας.

Οι καρτέλες των αισθητήρων προβάλλονται σαν λίστα μέσω του **RecyclerView.** Κάθε καρτέλα περιέχει το όνομα του αισθητήρα ένα αφαιρετικό εικονίδιο ένα **on/off switch** καθώς και ένα **slider.**

- **Το on/off switch** ενεργοποιεί και απενεργοποιεί τον αισθητήρα αντίστοιχα ανεξαρτήτως από την παρουσία του στη λίστα.
- **Το slider** κάθε αισθητήρα είναι ορισμένο να κινείται στα όρια των τιμών που μας δίνονται από την εκφώνηση. Με αυτόν τον τρόπο αποφεύγεται η πιθανότητα αποστολής μη έγκυρης τιμής.



Settings

Στις ρυθμίσεις της εφαρμογής **iot** συναντάμε τις εξής έξτρα λειτουργικότητες:

- Σε περίπτωση χειροκίνητης επιλογής τοποθεσίας ο χρήστης καλείται να διαλέξει μεταξύ τεσσάρων προεπιλεγμένων τοποθεσιών.
- **Device ID.** Αυτή η επιλογή δέχεται εναν ακέραιο αριθμό προκειμένου να είναι ξεκάθαρο στον server για ποια iot συσκευή πρόκειται.

Επικοινωνία με το Server

Η εφαρμογή κάνει **publish** σε ένα **topic** του οποίου το όνομα αποτελείται από το string ¨iot¨ + **device_id**(όπου device_id = 1, 2, 3 etc). Στο topic αυτό στέλνει ενα json αρχείο το οποίο περιέχει τις συντεταγμένες, τη στάθμη της μπαταρίας, το device ID καθώς και τα δεδομένα από τους αισθητήρες(όνομα, αριθμο και τιμή).

Χρήση των εφαρμογών

Και οι δύο εφαρμογές μπορούν να χρησιμοποιηθούν πολύ εύκολα είτε μέσω emulator είτε σε φυσική συσκευή με εγκατάσταση του apk αρχείου και αποδοχή των απαραίτητων αδειών. Το ελάχιστο API που προτείνεται για τις εφαρμογές ειναι API 24.

Παρουσίαση Edge Server

Για Calculator*:*

Κάνει υπολογισμό απόστασης:

- **AlertSignal**: έχει 4 παραμέτρους και 2 περιπτώσεις
    1. Το σημείο 2 να μην υπάρχει (Iot2=null)

        distance: δίνει την απόσταση του Android με το σημείο iot1 και την επιστρέφει.

    2. Υπάρχουν και τα 2 σημεία (Iot1,Iot2)

        distance: δίνει την απόσταση του Android με τα 2 σημεία (με την βοήθεια της μέσης απόστασης των σημείων) και την επιστρέφει.

- **getCentalPoint**: υπολογίζει την μέση απόσταση των 2 σημείων

    1.**lanCentral** υπολογισμός μια νέας lat μέσω των 2 lat των Iot1 και Iot2, η οποία θα αποτελεί τον μέσο όρο των 2 lat

    2.**lonCentral** υπολογισμός μια νεας lon μέσω των 2 lon των Iot1 και Iot2, η οποία θα αποτελεί τον μέσο όρο των 2 lon

    3.δίνεται ένα νέο σημείο (εστω p) με τις νέες τιμές στο lat και lon, το οποίο αποτελεί την μέση απόσταση των Iot1 Kai Iot2

- **calculateDistance**: χρησιμοποιήσαμε αυτή που έχει δοθεί στην εκφώνηση.

- -> **Point**: είναι βοηθητικό για να μην δίνει πολλές συντεταγμένες ώστε να είναι πιο εύκολο για το τι παίρνει (δηλαδή έχει x, y). Αυτό το κάνουμε για να μην έχει ο getCentalPoint 4 παραμέτρους, βάζουμε Point Iot1 στο οποίο έχει 2 παραμέτρους μέσα του (lat, lon) τα αντίστοιχα δηλαδή (χ, y).

***Για MyCallback**:*

messageArived:Αρχικα δημιουργεί ένα MqttClient,μετατρέπει το μήνυμα σε json object, το στέλνει στο REST API κάνοντας ένα PUT request και ανάλογα την τιμή της μεταβλητής id(έχουμε ορίσει σαν device id της android συσκευής το 3 και για τις iot1,iot2 το 1 και 2 αντίστοιχα) εκτελείτε διαφορετικό κομμάτι κώδικα.Στην περίπτωση που το id ειναι 3(δηλαδη λαβαμε μήνυμα από την android συσκευη)δημιουργείτε το android point με τις τρέχουσες συντεταγμένες της android συσκευής.Επίσης η μεταβλητή and_flag(αρχικοποιείται ως false ετσι ώστε στην περίπτωση που λάβει μήνυμα ο server απο καποια iot συσκευή με βαθμό κινδύνου (risk) high/medium πρωτα,ενω δεν εχει τιμες για της συντεταγμενες της android συσκευης να μην προσπαθήσει να υπολογίσει την απόσταση μεταξύ iot και android) γινεται true. Στην περίπτωση που το id ειναι 1 η 2(δηλαδη λαβαμε μηνυμα απο μια iot συσκευη)αν υπάρχουν ενεργοί ασθητήρες αποθηκεύει την τιμή τους στην αντίστοιχη μεταβλητή και υπολογίζει τον βαθμό κινδύνου (risk).Αν η risk ειναι high/medium το iot_flag της συσκευής που έστειλε το μήνυμα παίρνει την τιμή true αλλιώς παίρνει την τιμή false.Αν η risk ειναι high/medium και το and_flag ειναι true τότε ο MqttClient αφου πρώτα υπολογιστεί η απόσταση της iot συσκευής απο την android(αν το iot_flag της άλλης iot συσκευής είναι false)ή η αποσταση του κέντρου των 2 iot συσκευών απο την android(αν το iot_flag της άλλης iot συσκευής είναι true) κάνει publish στο κατάλληλο topic(που είναι subscriber η android συσκευή)μήνυμα που περιέχει τον βαθμό επικινδυνότητας και την απόσταση.

Για Main

Ορίζουμε τα topics και τον online broker που χρησιμοποιούμε για την επικοινωνία μεταξύ server,iot συσκευών και android.Δημιουργούμε ένα MqttClient ο οποίος κάνει subscribe στα παραπάνω topics και setCallback την MyCallback.

Για να τρέξει ο server σωστά πρέπει να τρέχει το REST api.

Παρουσίαση REST API

Για την σύνδεση της γραφικής διεπαφής με τον server δημιουργήθηκε ένα REST api δημιουργημένο σε node.js. Το REST api λαμβάνει ένα json από τον server, μόλις έρθει μήνυμα από κάποια συσκευή, το γράφει στο αρχείο data.json και από εκεί το παίρνει η γραφική διεπαφή για να το επεξεργαστεί.

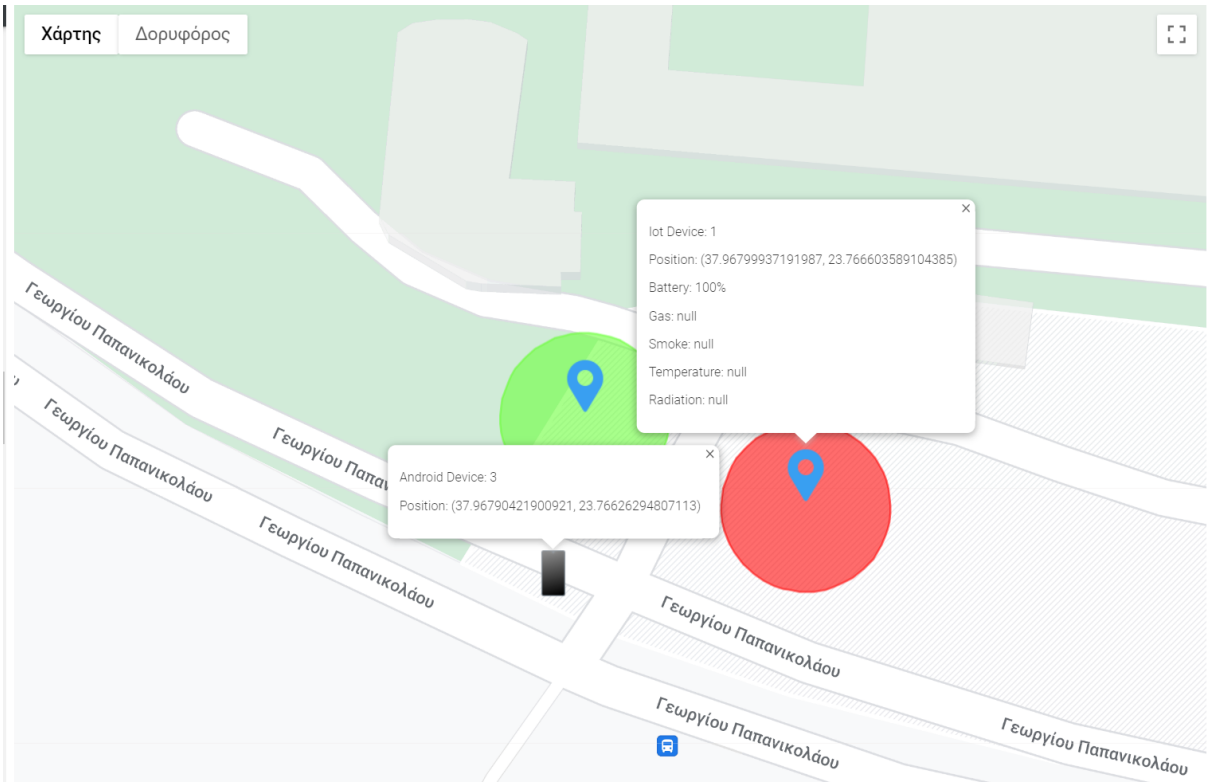
Για να τρέξουμε το REST api χρειάζονται το node.js (v18.13.0) και το αντίστοιχο npm packet manager. Στον φάκελο του REST api εκτελούμε την εντολή “**npm install**” για να εγκατασταθούν τα απαιτούμενα πακέτα. Έπειτα για να το τρέξουμε εκτελούμε στον φάκελο την εντολή “**node app.js**”. Το REST api είναι ρυθμισμένο να τρέχει στο **port:3000**.

Γραφική Διεπαφή

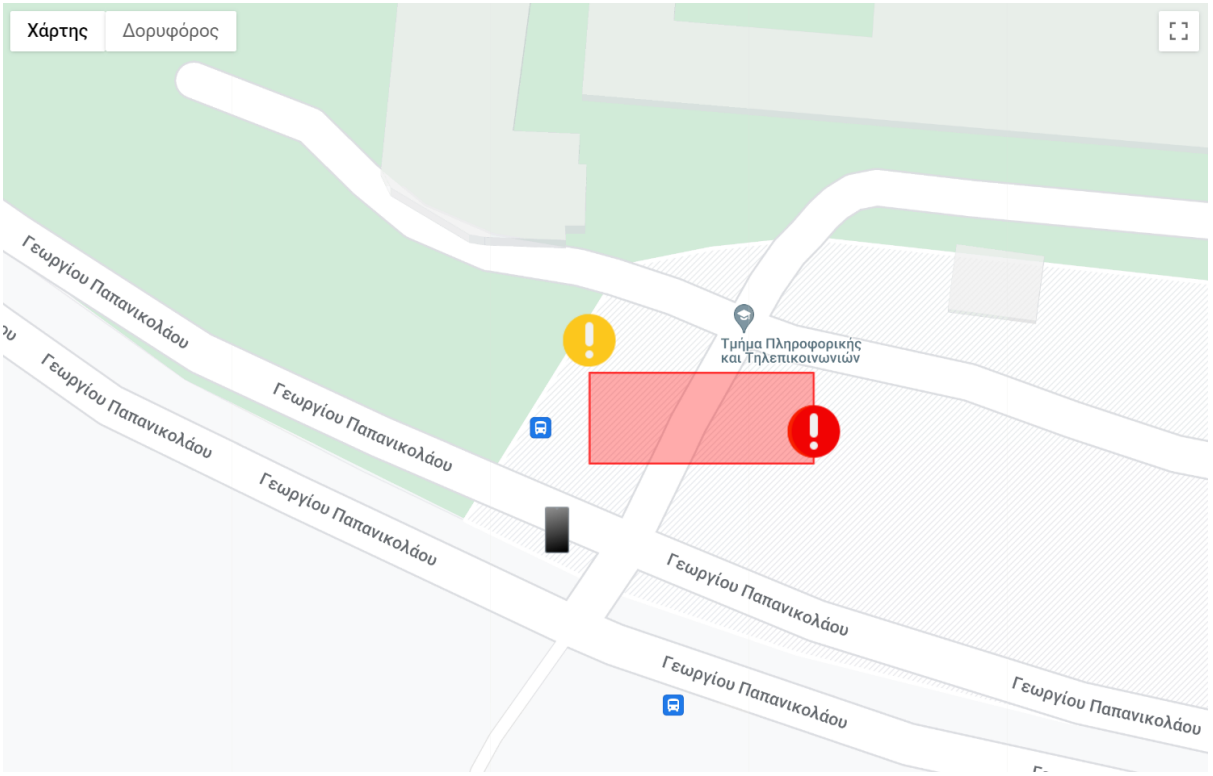
Η γραφική διεπαφή αναπτύχθηκε με javascript και συγκεκριμένα με το Svelte framework. Η γραφική διεπαφή χρησιμοποιεί το Google Maps Api για να αποτυπώσει τον χάρτη.

Η συνάρτηση fetchData χρησιμοποιείται για να “πάρει” τα δεδομένα η γραφική διεπαφή απο το REST api. Έπειτα η συνάρτηση καλή ανάλογα με το id της συσκευής (έχουμε ορίσει σαν device id της android συσκευής το 3 και για τις iot1, iot2 το 1 και 2 αντίστοιχα) καλεί την αντίστοιχη συνάρτηση (addAndroidMarker για id 3 και addIotMarkers για 2 η 3.)

Πάνω στον χάρτη εμφανίζονται τα markers της κάθε συσκευής. Υπάρχει διαφορετικό marker για την android συσκευή και για τις iot συσκευές. Το εικονίδιο της κάθε iot συσκευής αλλάζει ανάλογα με το επίπεδο κινδύνου της (risk level). Κάθε συσκευή έχει το δικό της infowindow το οποίο μπορούμε να το δούμε πατώντας πάνω στο marker της. Τα infowindows ανανεώνονται κάθε δευτερόλεπτο. Κάτω από τα markers της κάθε iot συσκευής υπάρχει ένας κύκλος που μας δείχνει αν η συσκευή είναι ενεργή.



Όταν υπάρχει event κινδύνου (high ή medium) εμφανίζεται στον χάρτη ένα κόκκινο ορθογώνιο παραλληλόγραμμο, με τις 2 iot συσκευές τοποθετημένες στις γωνίες του, το οποίο δείχνει την επικίνδυνη περιοχή.



Συναρτήσεις:

- fetchData:Καλείται κάθε ένα δευτερόλεπτο με την χρήση της setinterval().Κάνει request στο api για να λάβει το json αρχείο με τα δεδομένα που έστειλε η κάθε συσκευή. Έπειτα κάνει parse το json που λαμβάνει από το api. Ανάλογα με το id καλεί την addAndroidMarker ή την addIotMarkers.

- addAndroidMarker: Παίρνει ως ορίσματα την τοποθεσία (lat και lon) και το id της android συσκευής.Δημιουργεί ένα marker για την android συσκευή και το infowindow του αν δεν υπάρχει ήδη. Αν υπάρχει ανανεώνει την τοποθεσία του marker και τα περιεχόμενα του infowinfow.

- addIotMarkers: Παίρνει ως ορίσματα την τοποθεσία (lat και lon) της iot συσκευής το id, την τιμή της μπαταρίας και τις τιμές των αισθητήρων της(gas,smoke,temp,uv). Αν δεν υπάρχει iot marker με το ίδιο id δημιουργεί το marker, το infowindow του, τον κύκλο κάτω από το marker (αν χρειάζεται) και δημιουργεί το ορθογώνιο παραλληλόγραμμο (αν χρειάζεται). Αν υπάρχει iot marker με το ίδιο id, ανανεώνει το infowindow, αλλάζει χρώμα, ή εξαφανίζει τον κύκλο, ανάλογα με την περίπτωση, αλλάζει το εικονίδιο του marker, αν χρειάζεται και εμφανίζει ή εξαφανίζει το ορθογώνιο παραλληλόγραμμο.

- calculateRisk: Παίρνει ως ορίσματα τις τιμές των αισθητήρων και επιστρέφει την τιμή του κινδύνου (1 για low risk, 2 για medium risk και 3 για high risk) ανάλογα με αυτές.

- chooseIcon: Παίρνει ως όρισμα την τιμή του κινδύνου (risk) και διαλέγει το εικονίδιο για τα markers των iot συσκεύων.

- getCircleColor: Παίρνει ως ορίσματα τις τιμές των αισθητήρων και διαλέγει το χρώμα του κύκλου με βάση το αν είναι όλες “null”.

- drawPolygon: Δημιουργεί ένα ορθογώνιο παραλληλόγραμμο με βάση δύο σημεία στο χάρτη.

Γι να τρέξει η γραφική διεπαφή χρειάζονται το node.js (v18.13.0) και το αντίστοιχο npm packet manager. Στον φάκελο της γραφικής διεπαφής εκτελούμε την εντολή “**npm install**” για να εγκατασταθούν τα απαιτούμενα πακέτα. Έπειτα για να τρέξουμε την γραφική διεπαφή στον φάκελο της εκτελούμε την εντολή “**npm run dev**”. Η γραφική διεπαφή είναι ρυθμισμένη να τρέχει στο **port:8080** και μπορούμε να την δούμε στον browser στο:

http://localhost:8080/