

ΕΡΓΑΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ ΔΙΚΤΥΩΝ

(2024)

ΤΣΙΓΓΙΣΤΡΑΣ ΠΑΥΛΟΣ: sdi2000264

ΤΣΙΤΣΙΡΙΓΓΟΣ ΧΑΡΑΛΑΜΠΟΣ : sdi1900198

MININET:

Το Mininet είναι ένα εργαλείο προσομοίωσης δικτύων που επιτρέπει τη δημιουργία και την προσομοίωση δικτύων και χρησιμοποιείται κυρίως για την προσομοίωση δικτύων τύπου Software-Defined Networking (SDN). Στην περίπτωση αυτής της εργασίας θα χρησιμοποιήσουμε το Mininet – WIFI που είναι ένα fork του Mininet και μας δίνει περισσότερα εργαλεία για να αναδείξουμε το δίκτυο μας.

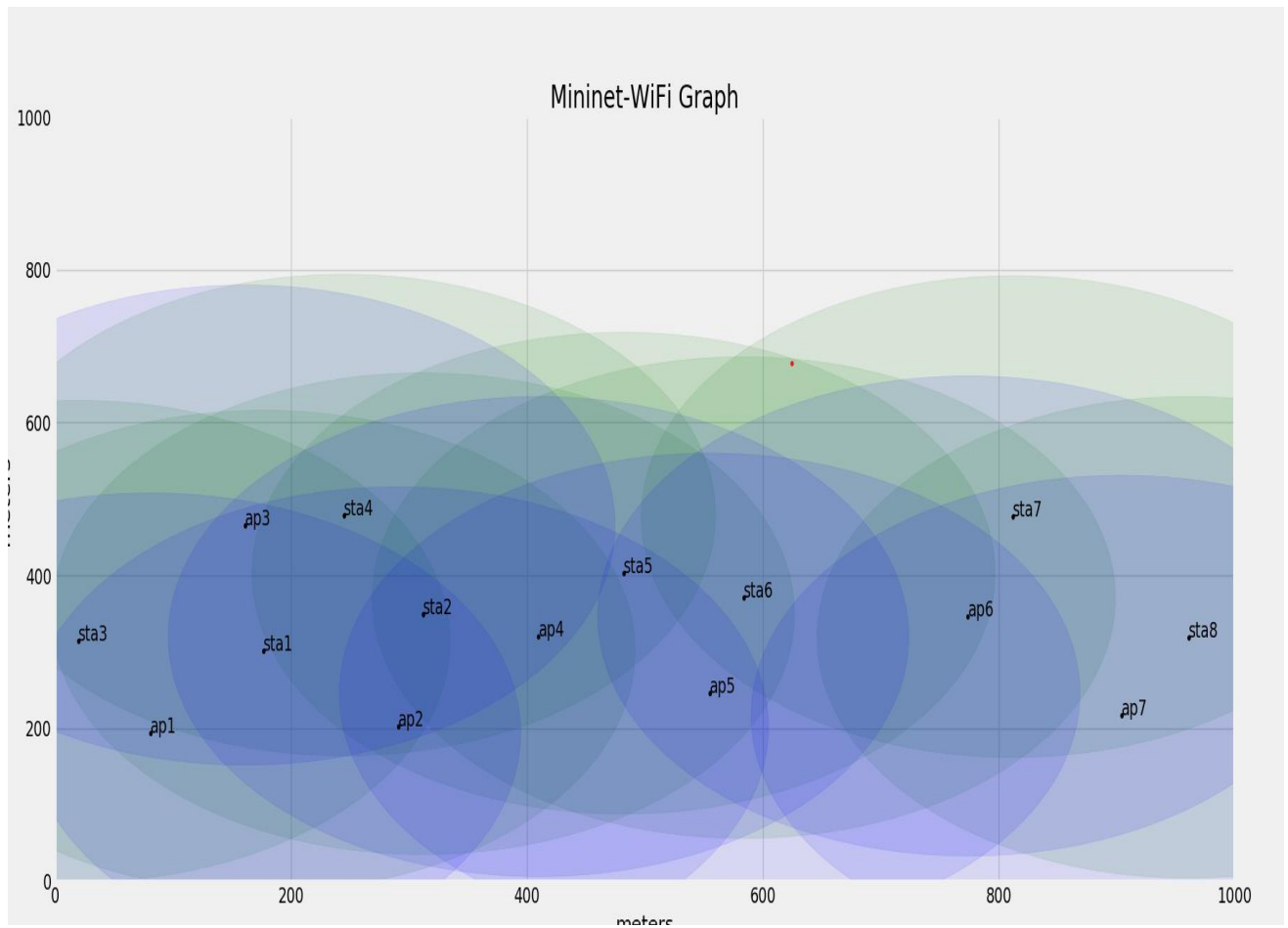
TOPOLOGIES:

Για την δημιουργία των indoor και outdoor σεναρίων βάση των διαφανειών φτιάχτηκαν δυο τοπολογίες.

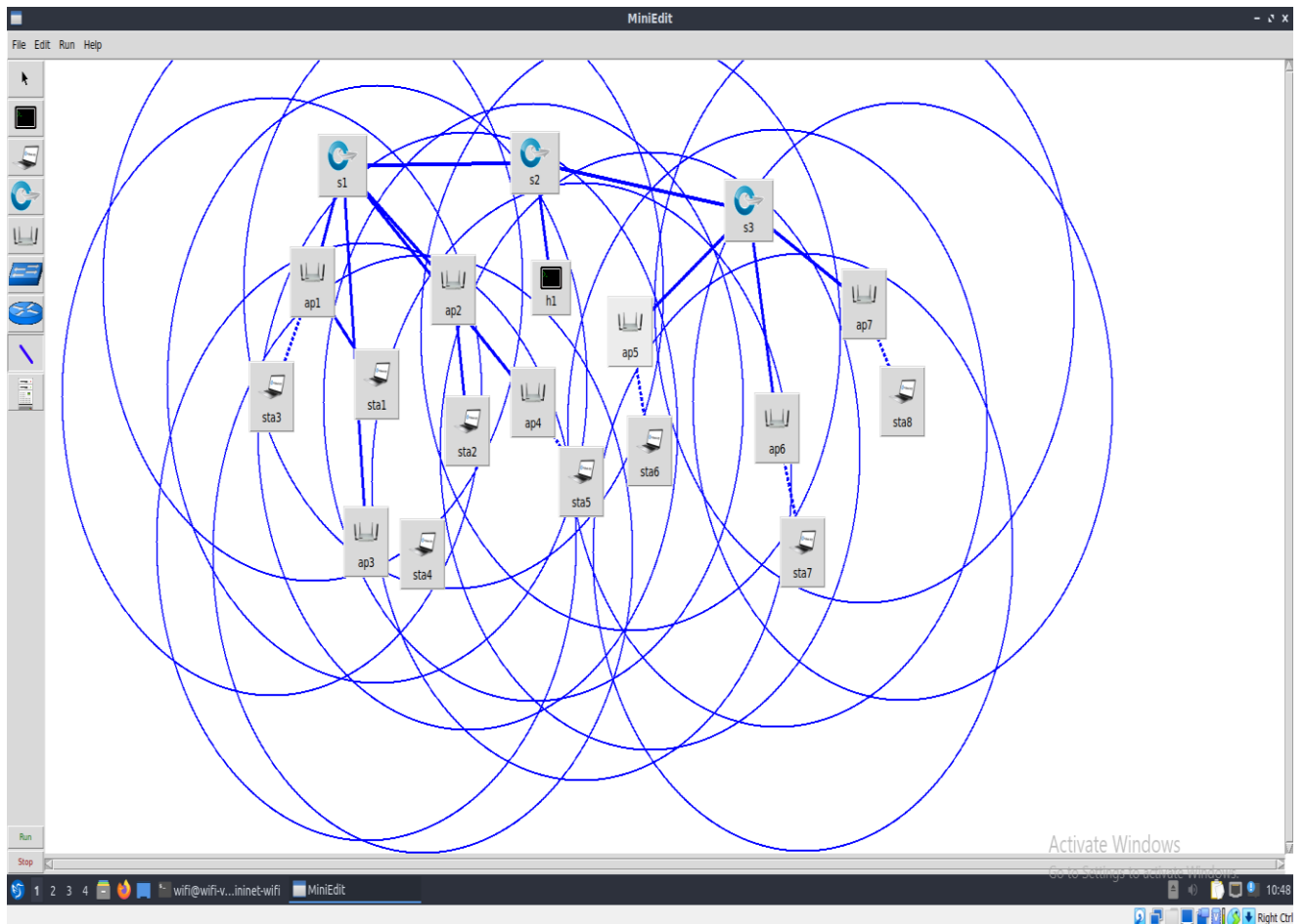
Και στις δυο περιπτώσεις έχουμε έναν host που αντιπροσωπεύει τον server του δικτύου μας, τα AP(access points) που αντιπροσωπεύουν τις κεραίες, ο host και η κεραίες συνδέονται με switches. Τα stations συνδεδεμένα στα access points αντιπροσωπεύουν τους χρήστες.

- Outdoor:

Για αυτό το σενάριο χρησιμοποιήσαμε 7 AP(κεραίες) ενδεικτικά από τις διαφάνειες τις εκφώνησης 3 switches και έναν host. Η τοπολογία περιέχει 8 stations (χρήστες) που θα μας βοηθήσουν να αναδείξουμε την λειτουργικότητα της τοπολογίας.



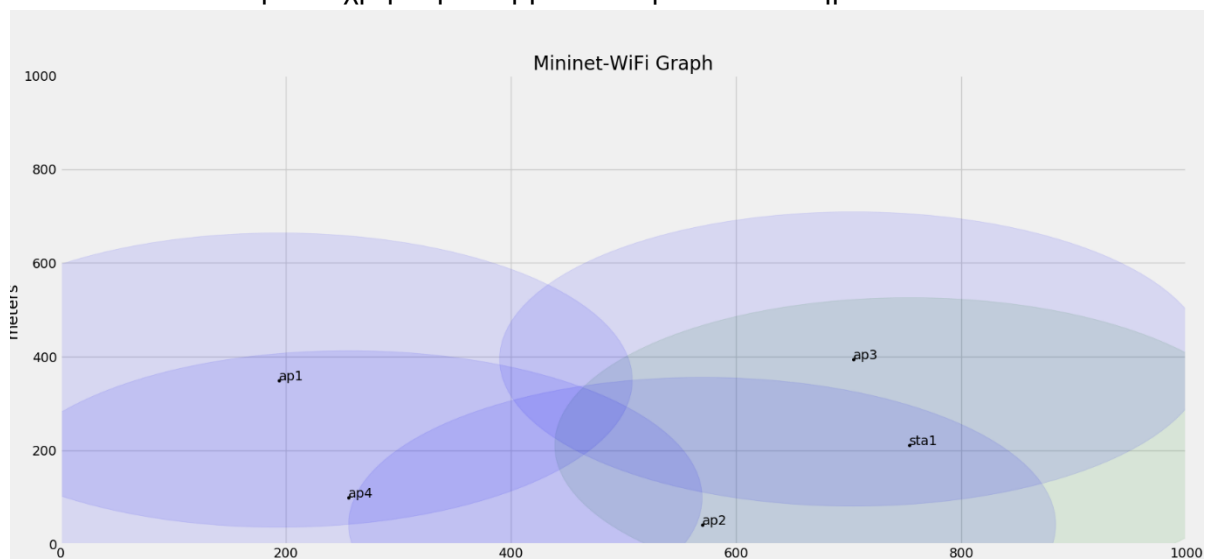
1 To outdoor δίκτυο με τα access points και stations



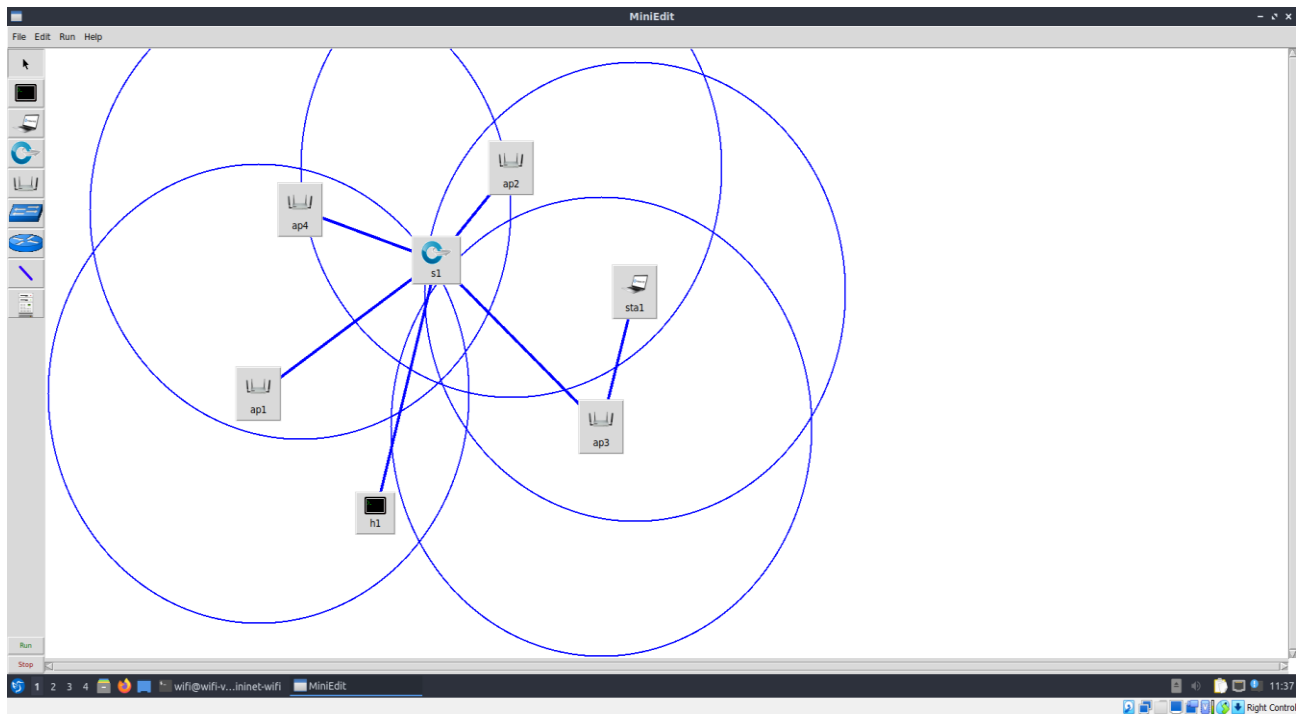
2 Το outdoor δίκτυο όπως φαίνεται στο εργαλείο Miniedit

• Indoor:

Για το indoor σενάριο χρησιμοποιήσαμε 4 AP, 1 switch και 1 host. Υπάρχει 1 station που αναπαριστά χρήστη που βρίσκεται μέσα στο κτήριο.



3 Το indoor δίκτυο με τα access points και τα stations του



4 Το indoor δίκτυο όπως φαίνεται στο εργαλείο Miniedit

FAULT MANAGEMENT

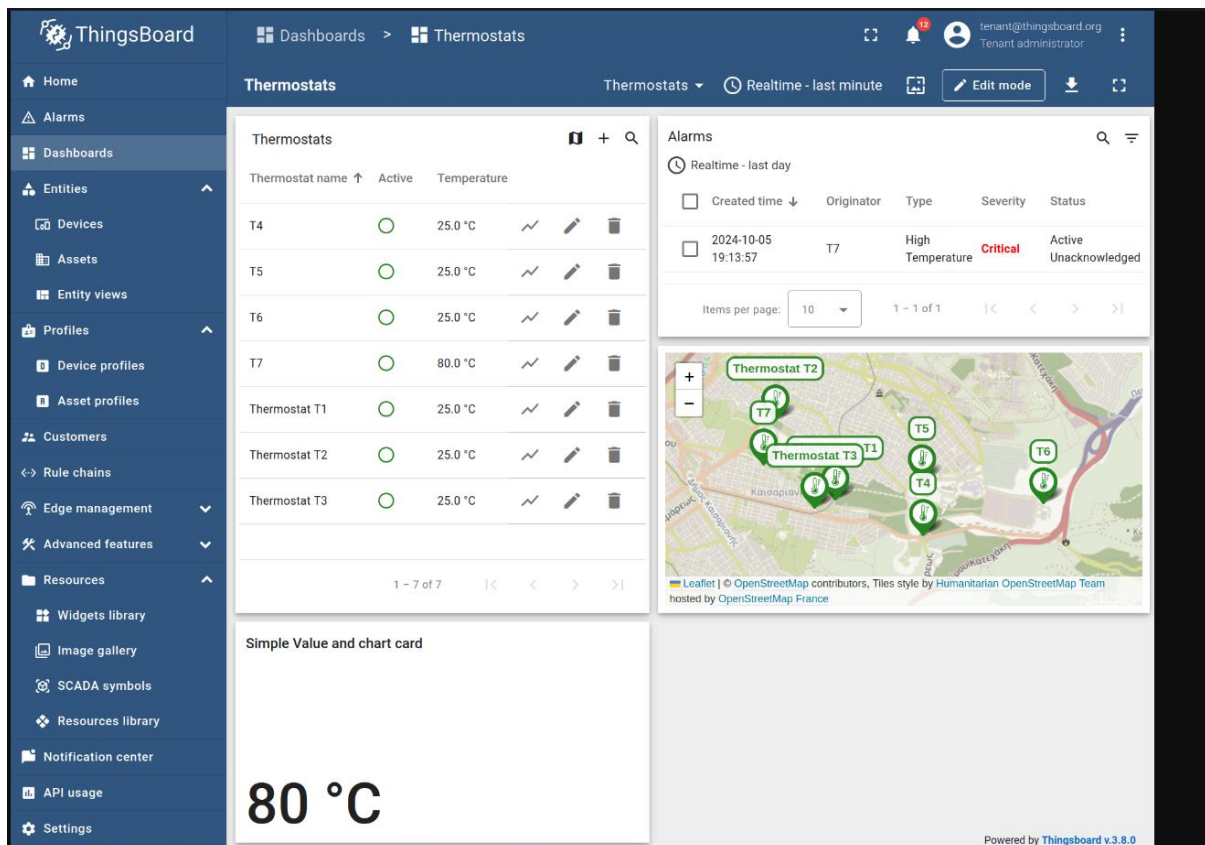
Το **fault management (διαχείριση σφάλματος)** είναι μία από τις βασικές λειτουργίες της **διαχείρισης δικτύων** και αφορά την ανίχνευση, διάγνωση, και επίλυση προβλημάτων που σχετίζονται με τις συσκευές και τις συνδέσεις σε ένα δίκτυο. Το κύριο μέλημα του fault management είναι να διατηρεί το δίκτυο σε λειτουργική κατάσταση και να μειώνει τον χρόνο διακοπών (downtime) που μπορεί να προκύψει λόγω βλαβών. Στα δίκτυα που δημιουργήσαμε θα παρομοιώσουμε την απώλεια μιας κεραίας και θα επιδιορθώσουμε το σφάλμα μέσω αναδιάταξης του δικτύου.

Η προσομοίωση της καταστροφής κεραίας θα γίνει μέσω terminal σε runtime. Θα “καταστρέψουμε” με την εντολή **py ap.stop(ap2)** την κεραία, στο outdoor σενάριο αυτή θα είναι το ap2. Στη συνέχεια με τη χρήση της **py net.addLink(sta2,ap4)** συνδέουμε τον χρήστη της κεραίας που καταστράφηκε με το access point 4 στο οποίο ο χρήστης βρίσκεται ήδη στην εμβέλεια. Στην περίπτωση που ο χρήστης βρίσκεται εκτός εμβέλειας από κάποια άλλη κεραία προσομοιώνω την μετακίνηση του με την εντολή **py sta.setPosition(xx,yy)**.

THINGSBOARD

Για κάθε σενάριο (indoor-outdoor) δημιουργήσαμε 7 devices που αναπαριστούν τους αισθητήρες θερμοκρασίας σε κάθε κεραία (για το outdoor σενάριο). Για την προσομοίωση των δεδομένων των αισθητήρων στείλαμε με rest API call χρησιμοποιώντας curl δεδομένα θερμοκρασίας πχ. (curl -v -X POST http://localhost:8080/api/v1/t7/telemetry --header Content-Type:application/json --data "{temperature:80}"). Τα devices έχουν δημιουργηθεί έτσι ώστε όταν οι αισθητήρες μας ανιχνεύσουν θερμοκρασία άνω των 70 βαθμών να στέλνουν alarm. Για το dashboard χρησιμοποιήσαμε το map widget και τοποθετήσαμε τους αισθητήρες μας στην περιοχή της πανεπιστημιούπολης σύμφωνα με το outdoor σενάριο που φαίνεται στην εκφώνηση της εργασίας.

Screenshots outdoor scenario:



Βλέπουμε πως υπάρχει alarm υψηλής θερμοκρασίας στον T7.

Στιγμιότυπο του T7 με ανίχνευση υψηλής θερμοκρασίας. Στα δεξιά φαίνονται τα API calls για να προσομοιώσουμε δεδομένα στον αισθητήρα.

The screenshot displays the ThingsBoard interface for a thermostat named T7. The left sidebar contains navigation options like Home, Alarms, Dashboards, Entities, and Settings. The main area shows the 'Thermostat settings' for T7, including a 'High temperature alarm' with a threshold of 70°C. Below this is a map showing the location of T7. To the right, a 'Temperature' graph shows a sharp spike to 70°C. Below the graph, a table lists alarms, showing a 'High Temperature' alarm for T7 that is 'Critical' and 'Active'. On the far right, a terminal window shows the API calls used to simulate data, including a POST request to the /api/v1/t7/telemetry endpoint.

Created time	Originator	Type	Severity	Status
2024-10-05 19:13:57	T7	High Temperature	Critical	Active Unacknowledged

Screenshot indoor scenario

The screenshot displays the ThingsBoard interface for an indoor scenario. The left sidebar contains navigation options like Home, Alarms, Dashboards, Entities, and Settings. The main area shows a floor plan with markers for temperature sensors. Below the floor plan, a table lists entities, including 'Indoor t1', 'it2', 'it3', and 'it4'. To the right, a terminal window shows the API calls used to simulate data, including a POST request to the /api/v1/temperature endpoint.

Name	temperature
Indoor t1	
it2	
it3	
it4	

Βλέπουμε πως υπάρχει alarm υψηλής θερμοκρασίας στον it7

Screenshot του it7 με ανίχνευση υψηλής θερμοκρασίας. Στα δεξιά φαίνονται τα API calls για να προσομοιώσουμε δεδομένα στον αισθητήρα.

The screenshot displays the ThingsBoard web interface for a 'Thermostats' dashboard. The left sidebar contains navigation links for Home, Alarms, Dashboards, Entities, Devices, Assets, Entity views, Profiles, Device profiles, Asset profiles, Customers, Rule chains, Edge management, Advanced features, Resources, Widgets library, Image gallery, SCADA symbols, Resources library, Notification center, API usage, and Settings.

The main content area shows the 'Thermostats' dashboard for device 'it7'. It includes a 'Thermostat settings' widget with a 'High temperature alarm' set at 50°C. A 'Temperature' widget shows a real-time graph of temperature over the last 10 minutes, with a red line indicating a spike to 50°C. Below the graph is an 'Alarms' table showing a critical alarm triggered at 2024-10-05 19:13:57.

On the right, a terminal window shows the following API calls and responses:

```
< Vary: Access-Control-Request-Headers
< X-Content-Type-Options: nosniff
< X-XSS-Protection: 0
< Cache-Control: no-cache, no-store, max-age=0, must-revalidate
< Pragma: no-cache
< Expires: 0
< X-Frame-Options: DENY
< Content-Length: 0
< Date: Sat, 05 Oct 2024 16:13:57 GMT
* Connection #0 to host localhost left in tact
[...]
```

```
> curl -v -X POST http://localhost:8080/api/v1/QyHxLpLJkxZjDdi3ysEL/telemetry --header Content-Type:application/json --data '{"temperature":80}'
Note: Unnecessary use of -X or --request, POST is already inferred.
* Host localhost:8080 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
* Trying [::1]:8080...
* Connected to localhost (::1) port 8080
* using HTTP/1.x
* POST /api/v1/QyHxLpLJkxZjDdi3ysEL/telemetry HTTP/1.1
* Host: localhost:8080
* User-Agent: curl/8.10.1
> Accept: */*
< Content-Type:application/json
< Content-Length: 16
>
* upload completely sent off: 16 bytes
< HTTP/1.1 200
< Vary: Origin
< Vary: Access-Control-Request-Method
< Vary: Access-Control-Request-Headers
< X-Content-Type-Options: nosniff
< X-XSS-Protection: 0
< Cache-Control: no-cache, no-store, max-age=0, must-revalidate
< Pragma: no-cache
< Expires: 0
< X-Frame-Options: DENY
< Content-Length: 0
< Date: Sat, 05 Oct 2024 17:27:20 GMT
* Connection #0 to host localhost left in tact
[...]
```

ΕΚΚΕΝΩΣΗ

Το interface της εφαρμογής μας θα προσομοιωθεί μέσω mock images και θα επιδεικνύει στον χρήστη το shortest path (συντομότερο μονοπάτι) για εκκένωση από το σημείο της πυρκαγιάς.

Παρακάτω στην εικόνα 1 βλέπουμε την κατάσταση της εφαρμογής μας πριν την ανίχνευση κάποιας πυρκαγιάς.



1 Το κόκκινο pin αναπαριστά την τοποθεσία του χρήστη της εφαρμογής.

Αν εντοπιστεί εκδήλωση φωτιάς από την εφαρμογή εμφανίζεται στον χρήστη ένα alert που του δηλώνει την παρουσία φωτιάς όπως φαίνεται στην από κάτω φωτογραφία.



2 Αναπαράσταση εκκένωσης

