



*Τμήμα Μηχανικών Η/Υ και Πληροφορικής  
Πολυτεχνική Σχολή*

**ΟΜΑΔΙΚΉ ΕΡΓΑΣΙΑ ΔΙΑΧΥΤΟΣ ΥΠΟΛΟΓΙΣΜΟΣ**

**ΟΜΑΔΑ 0**

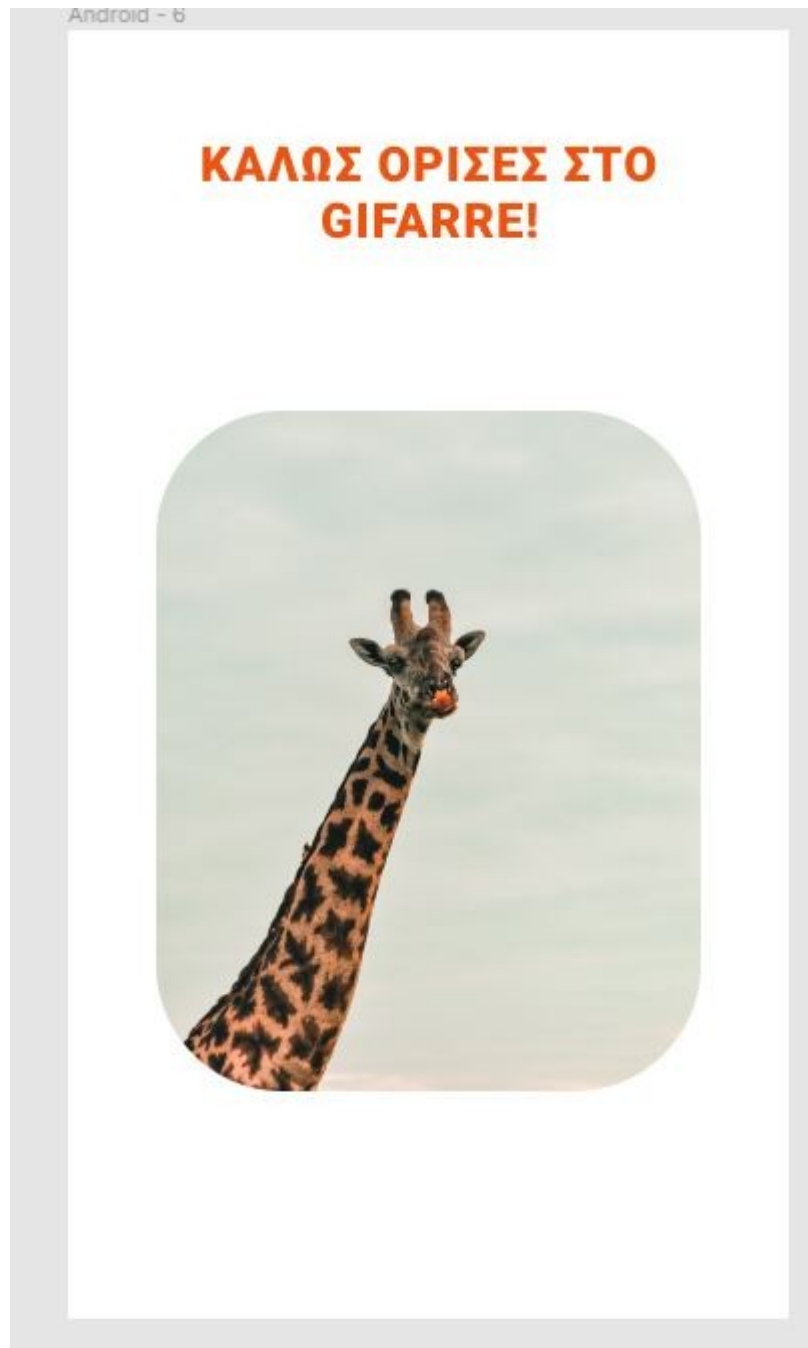
ΝΑΚΚΑΣ ΝΙΚΟΛΑΟΣ ΑΜ:1054359

ΣΥΡΟΚΩΣΤΑΣ ΚΩΝΣΤΑΝΤΙΝΟΣ ΑΜ:1054339

ΠΑΠΑΚΟΣ ΜΙΛΤΙΑΔΗΣ ΑΜ:1054360

## Μέρος 1

### Οθόνες-Mock ups



Η παραπάνω οθόνη είναι μια αρχική οθόνη γνωριμίας του χρήστη με την εφαρμογή μας (και της καμηλοπάρδαλης μασκότ μας, την Σούλα).

## ΑΣ ΓΝΩΡΙΣΤΟΥΜΕ!

ΗΛΙΚΙΑ:

ΦΥΛΟ



ΑΣ ΞΕΚΙΝΗΣΟΥΜΕ

Αν είναι η πρώτη φορά σύνδεσης του χρήστη στην εφαρμογή μας πρέπει να γνωρίσουμε τις πολύ βασικές πληροφορίες του χρήστη (όπως η Σούλα γνώρισε τον Γεράσιμο βλ. εικόνα).

## GIFARRE




Όταν συνδέεται η συσκευή του χρήστη θα του δείχνει κατόπιν της αρχικής οθόνης αυτή την οθόνη με την απεικόνιση των πληροφοριών σε Heatmap.

Ως ομάδα είχαμε αρχικά σκεφτεί να κάνει εγγραφή ο χρήστης και να μπαίνει με δικούς του κωδικούς και e-mail αλλά θεωρήσαμε ότι δεν έχει νόημα για έναν χρήστη να πρέπει να φτιάξει λογαριασμό για να βλέπω απλώς κάποια στατιστικά χρήσης της συσκευής του. Για τον λόγο αυτό επιλέξαμε απλώς η εφαρμογή να αποθηκεύει στο local storage του κινητού του χρήστη το ID της συσκευής του. Με αυτόν τον τρόπο βέβαια αν ο χρήστης διαγράψει την εφαρμογή (ή καθαρίσει τα δεδομένα της) θα χαθεί η “γνώση” για το ποια συσκευή στο ThingsBoard αντιστοιχεί σε αυτόν. Θεωρούμε όμως ότι αυτό είναι κάτι που θα συμβεί σπάνια οπότε και πάλι προτιμήσαμε να μην υπάρχει δυνατότητα σύνδεσης/εγγραφής. Σε επόμενο version της εφαρμογής θα μπορούσαμε να βάλουμε τη δυνατότητα για προαιρετική εγγραφή από τον χρήστη.

Επιπλέον στην αρχική σχεδίαση είχαμε αποφασίσει να δείχνουμε στον χρήστη κάποια γραφήματα σχετικά με την ώρα χρήσης του κινητού του. Ωστόσο στη συνέχεια σκεφτήκαμε ότι αφού ζητάμε από τον χρήστη την τοποθεσία του θα είναι καλύτερο να του δείχνουμε πληροφορίες σχετικά με αυτή. Οπότε στην τελική εφαρμογή έχουμε αντικαταστήσει τις εικόνες με τα γραφήματα με ένα heatmap που δείχνει τη συχνότητα χρήσης του κινητού του χρήστη σε διάφορες τοποθεσίες.

ΚΑΛΩΣ ΟΡΙΣΤΕΣ ΣΤΟ  
GIFARRE!



ΣΥΝΔΕΣΗ

ΕΓΓΡΑΦΗ

ΑΣ ΓΝΩΡΙΣΤΟΥΜΕ!

ΟΝΟΜΑ:


ΕΠΙΘΕΤΟ:

ΗΛΙΚΙΑ:

EMAIL:

ΚΩΔΙΚΟΣ:


ΕΠΙΛΗΘΕΥΣΗ ΚΩΔΙΚΟΥ:



ΣΥΝΔΕΣΗ!

EMAIL:

ΚΩΔΙΚΟΣ:

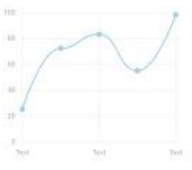


ΣΧΗΜΑΤΙΣΤΕ ΤΟΝ ΚΩΔΙΚΟ ΣΟΥ.

ΑΝΑΛΥΣΗ  
ΔΕΔΟΜΕΝΩΝ

ΕΒΔΟΜΑΔΑ

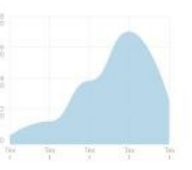
ΜΗΝΑΣ



ΑΝΑΛΥΣΗ  
ΔΕΔΟΜΕΝΩΝ

ΕΒΔΟΜΑΔΑ

ΜΗΝΑΣ



## **Τεχνολογίες ανάπτυξης εφαρμογής**

Figma: για την δημιουργία του προσχέδιου των οθονών (mock-ups)

Android Studio/ Kotlin: για την ανάπτυξη κώδικα για την εφαρμογή

ThingsBoard: για την αποθήκευση των συσκευών και των δεδομένων

Python (sklearn): για την ανάπτυξη του μοντέλου μηχανικής μάθησης

Google Cloud App Engine: για τη μεταφορά του μοντέλου στο cloud και την ενσωμάτωση του στην εφαρμογή

## Αρχιτεκτονική εφαρμογής

Γενικά η εφαρμογή ακολουθεί μία Client-Server αρχιτεκτονική όπου πολλοί clients (κινητά χρηστών) στέλνουν και λαμβάνουν δεδομένα από ένα server (ThingsBoard).

Πιο συγκεκριμένα η εφαρμογή λειτουργεί με τον εξής τρόπο.

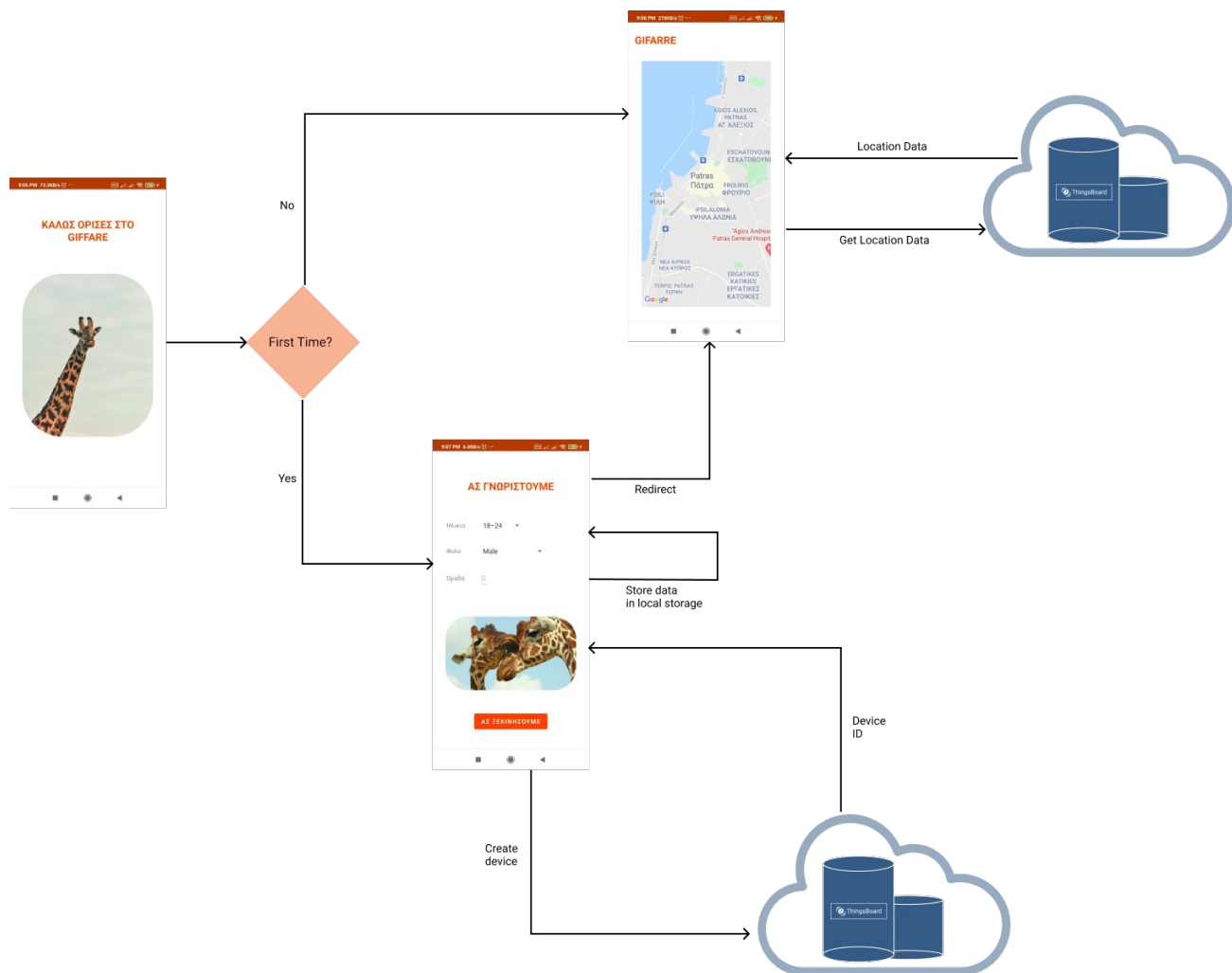
Όταν ένας χρήστης ανοίγει την εφαρμογή για πρώτη φορά εκείνη του ζητάει να συμπληρώσει ορισμένες βασικές πληροφορίες (ηλικιακή ομάδα, γένος, ομάδα που τον στρατολόγησε). Στη συνέχεια η εφαρμογή στέλνει ορισμένα requests στα endpoints του ThingsBoard ώστε να δημιουργηθεί μία νέα συσκευή για τον χρήστη. Η εφαρμογή τότε αποθηκεύει το ID αυτής της συσκευής στο Local Storage του χρήστη και ανακατευθύνει τον χρήστη στην κύρια οθόνη της εφαρμογής.

Αν ο χρήστης έχει ήδη ολοκληρώσει το βήμα καταχώρησης πληροφοριών και δημιουργίας συσκευής στο ThingsBoard η εφαρμογή τον ανακατευθύνει κατευθείαν στην κύρια οθόνη.

Από εκεί και πέρα όταν ο χρήστης ανοίγει και κλείνει το κινητό του μέσω ορισμένων background services και broadcast receivers στέλνονται δεδομένα στο ThingsBoard τα οποία αποθηκεύονται ως telemetries στη συσκευή που έχει ανατεθεί στον συγκεκριμένο χρήστη.

Όταν ο χρήστης ανοίγει την εφαρμογή και ανακατευθύνεται στην κύρια οθόνη η εφαρμογή ανακτά τα δεδομένα του χρήστη από το ThingsBoard (για τον τελευταίο μήνα) και τα παρουσιάζει στον χρήστη με μορφή Heatmap. Η εφαρμογή “ζητάει” μόνο το location id από το ThingsBoard στο request για να πάρει τα δεδομένα του χρήστη, επειδή αυτό αρκεί για τη δημιουργία του Heatmap και έτσι ελαχιστοποιούμε την κατανάλωση δεδομένων.

Τέλος αξίζει να σημειωθεί ότι έχουμε φτιάξει μία ακόμα συσκευή στο ThingsBoard την Team\_0\_Counter η οποία αποθηκεύει τον αριθμό των χρηστών της ομάδας μας. Αυτή χρησιμοποιείται για να μπορούμε να ανακτήσουμε αυτόν τον αριθμό κατά τη δημιουργία μίας νέας συσκευής από το κινητό του χρήστη ώστε να ορίσουμε το όνομα της συσκευής (Participant\_X\_Y).



Ολόκληρος ο κώδικας της εφαρμογής μπορεί να βρεθεί εδώ:  
<https://github.com/NikoNakkan/Diaxytos>



## Μέρος 2

### Συλλογή δεδομένων

Στο 2ο μέρος της εργασίας μας ασχοληθήκαμε με τη συλλογή δεδομένων από τα κινητά πραγματικών χρηστών.

Συγκεκριμένα κάναμε build την εφαρμογή που είχαμε φτιάξει στο 1ο μέρος και πήραμε το .apk αρχείο το οποίο έπειτα στείλαμε σε άλλους χρήστες.

Συνολικά από την ομάδα μας υπάρχουν 11 συσκευές στο ThingsBoard από τις οποίες 5 (0, 1, 2, 3, 5) δημιουργήθηκαν κατά το development της εφαρμογής και δεν περιέχουν δεδομένα. Από τις υπόλοιπες, οι οποίες αναπαριστούν πραγματικούς χρήστες έχουμε τα ακόλουθα δεδομένα:

Συσκευή	Αριθμός εγγραφών	Διάστημα αποστολής δεδομένων
Device_0_4	104	24/05 - 06/06
Device_0_6	26	31/05 - 06/06
Device_0_7	8	27/05 - 30/05
Device_0_8	23	27/05 - 29/05
Device_0_9	11	27/05 - 30/05
Device_0_10	46	27/05 - 30/05

Για την παρατήρηση των δεδομένων στο ThingsBoard φτιάξαμε ένα καινούριο Dashboard (Team\_0) όπου βάλαμε ένα widget πίνακα όπως φαίνεται και παρακάτω.

(Τα location\_type και location\_id είναι παντού 0 επειδή αποθηκεύονται ως strings στο ThingsBoard.)

ThingsBoard

ΑΡΧΙΚΗ

ΚΑΝΟΝΕΣ

ΠΕΛΑΤΕΣ

ASSETS

ΣΥΣΚΕΥΕΣ

ΟΡΕΙΣ ΟΝΤΟΤΗΤΩΝ

ΒΙΒΛΙΟΘΗΚΗ WIDGET

DASHBOARDS

ΗΜΕΡΟΛΟΓΙΑ ΚΑΤΑΓΡΑΦΗΣ

Dashboards > Team\_0

Team\_0Team\_0ΟντότητεςΙστορικό · από 2021-05-17 16:59:00 σε 2021-06-06 16:59:02

New Timeseries table

10PARTICIPANT\_0\_0PARTICIPANT\_0\_1PARTICIPANT\_0\_2PARTICIPANT\_0\_3PARTICIPANT\_0\_4PARTICIPANT\_0\_5

Timestamp ↓	device_interactive	display_state	location_type	location_id	location_conf	battery_level	battery_status	network_type	notifs_ac
2021-06-06 17:29:00	0	1.3333333333333333	0	0	0.12177081108093261	95	0	0	6
2021-06-05 22:29:00	0	1.25	0	0	0	67	0	0	6
2021-06-05 20:29:00	0	1.6666666666666667	0	0	0.01824376583099365	73	0	0	6
2021-06-05 18:29:00	0	1.625	0	0	0.08591509342193605	78	0	0	6
2021-06-05 16:29:00	0	1.5	0	0	0.0538411521911621	87	0	0	6
2021-06-05 14:29:00	0	1.5	0	0	0	90	0	0	6
2021-06-04 23:29:00	0	2	0	0	0	80	0	0	6
2021-06-04									

Items per page: 101 - 10 of 105

Powered by Thingsboard v3.0.1

## Μέρος 3

### Λήψη δεδομένων

Για την εκπαίδευση του μοντέλου μηχανικής μάθησης της ομάδας μας κατεβάσαμε τα δεδομένα των συσκευών της ομάδας μας καθώς και όσων άλλων είχαν αντίστοιχη δομή στα δεδομένα που ανέβαζαν. Για να το κάνουμε αυτό καθώς το ThingsBoard Community Edition δεν επιτρέπει τη δημιουργία συνολικών reports φτιάξαμε ένα widget το οποίο κατεβάζει όλα τα δεδομένα σε μορφή CSV. Τον κώδικα για το widget μπορείτε να τον βρείτε στο Widget Bundle της ομάδας μας στο ThingsBoard.

### Προεπεξεργασία δεδομένων

Τα δεδομένα που πήραμε από τους χρήστες τα επεξεργαστήκαμε με κατάλληλο τρόπο ώστε να έχουμε καλύτερα αποτελέσματα πρόβλεψης. Συγκεκριμένα κάναμε drop μη αριθμητικά δεδομένα και δεδομένα που υποδηλώνουν ταυτότητα συσκευής χρήστη. Ακόμα NaN δεδομένα αντικαταστάθηκαν με 0. Υπήρξε πείραμα να μπει αντί για 0 η μέση τιμή της εκάστοτε στήλης αλλά είχε μικρότερη απόδοση το μοντέλο μας.

### Δημιουργία και εκπαίδευση μοντέλου

Δημιουργήσαμε ένα μοντέλο χρησιμοποιώντας sklearn βιβλιοθήκη και έτοιμο μοντέλο LinearRegression . Το μοντέλο αυτό μας έδωσε σχετικά καλά αποτελέσματα με ακρίβεια 37%.

Η ακρίβεια είναι σχετικά μικρή ωστόσο μας επιτρέπει να κάνουμε integrate το μοντέλο στο σύστημα μας.

Τον κώδικα του μοντέλου μας μπορείτε να τον βρείτε σε ένα .zip αρχείο στο github όπου υπάρχει όλος ο κώδικας της ομάδας μας. Το link για το συγκεκριμένο αρχείο είναι: <https://github.com/NikoNakkan/Diaxytos/blob/master/MLModel.zip>

### Μεταφορά μοντέλου στο cloud

Σε αντίθεση με τον αρχικό σχεδιασμό μας (μέρος 1) αποφασίσαμε να μην ενσωματώσουμε το μοντέλο στην εφαρμογή μας με την χρήση του TensroFlow Lite αλλά να φτιάξουμε ένα cloud end-point με τη χρήση του Google App Engine όπου το μοντέλο θα τρέχει. Το μοντέλο εκπαιδεύτηκε τοπικά με τα δεδομένα που κατεβάσαμε από το ThingsBoard, αποθηκεύτηκε σε μορφή .pkl και ανέβηκε στο cloud.

Έτσι κάθε φορά που η εφαρμογή μας χρειάζεται να κάνει μία πρόβλεψη κάνει μία POST request στο endpoint <https://github.com/NikoNakkan/Diaxytos/blob/master/MLModel.zip> περνώντας τις παραμέτρους εισόδου στο σώμα του request και ως απάντηση παίρνει την τιμή που προβλέπει το μοντέλο.

Το Cloud Endpoint έχει γραφτεί σε Python για να είναι συμβατό με το μοντέλο και η βιβλιοθήκη Flask χρησιμοποιήθηκε για τη διαχείριση των endpoints.

Η συγκεκριμένη προσέγγιση για την ενσωμάτωση του μοντέλου στην εφαρμογή μας, μας προσφέρει τα εξής πλεονεκτήματα:

- Μείωση κατανάλωσης των πόρων του συστήματος (CPU, μνήμης) αφού η πρόβλεψη δεν γίνεται στη συσκευή.
- Μείωση μεγέθους εφαρμογής.
- Ευκολία αλλαγής του μοντέλου καθώς νέα δεδομένα έρχονται από τους χρήστες (καθώς αν αλλάξουμε το μοντέλο στον server διατηρώντας το ίδιο endpoint url η αλλαγή θα είναι εμφανής και στην εφαρμογή του χρήστη χωρίς αυτός να χρειαστεί να κάνει κάποιο update).
- Μεγαλύτερη ακρίβεια στις προβλέψεις του μοντέλου (37%), επειδή όταν υλοποιήσαμε το μοντέλο με TensorFlow και Keras για να το εκάστοτε την εφαρμογή μας (μέσω TensorFlow Lite) η ακρίβεια των προβλέψεων μας έπεσε στο 3%.

Βέβαια αυτή η προσέγγιση έχει και ορισμένα μειονεκτήματα το βασικότερο εκ των οποίων είναι η κατανάλωση δικτύου (και ειδικά δεδομένων όταν αυτά χρησιμοποιούνται από τον χρήστη) για να γίνει το request. Ωστόσο η κατανάλωση δικτύου από μία POST request είναι πολύ μικρή και γι' αυτό αποφασίσαμε να ακολουθήσουμε αυτή την προσέγγιση.

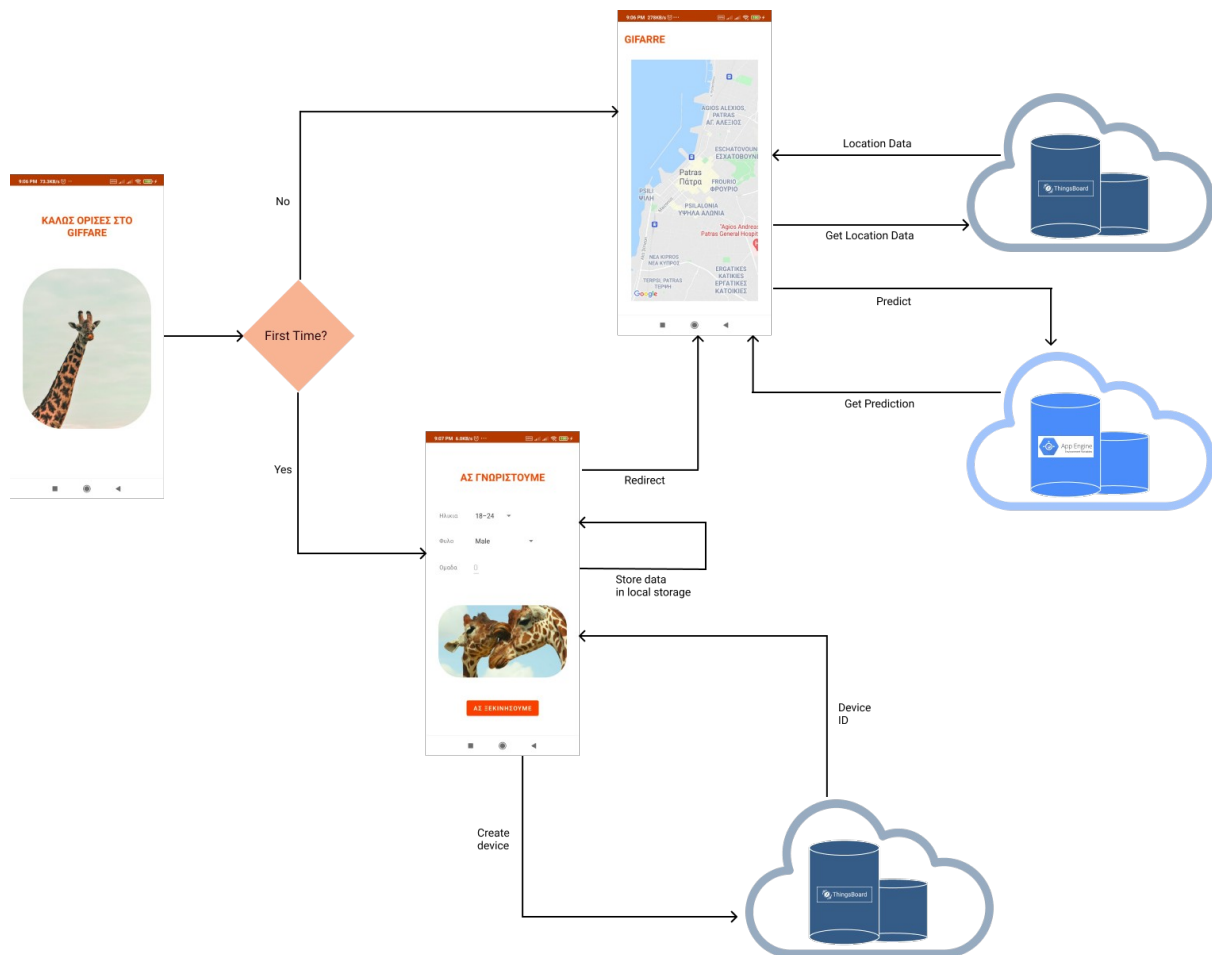
Τον κώδικα του Cloud Endpoint μας μπορείτε να τον βρείτε σε ένα .zip αρχείο στο github όπου υπάρχει όλος ο κώδικας της ομάδας μας. Το link για το συγκεκριμένο αρχείο είναι: <https://github.com/NikoNakkan/Diaxytos/blob/master/AppEngine.zip>

## Χρήση μοντέλου

Η εφαρμογή χρησιμοποιεί το μοντέλο με τον εξής τρόπο. Κάθε φορά που ο χρήστης ανοίγει το κινητό του και η εφαρμογή εντοπίζει αυτή την ενέργεια ενεργοποιείται ένα alarm\* με συχνότητα περίπου 15 λεπτά (η μέθοδος *setInexactRepeating* χρησιμοποιείται για εξοικονόμηση ενέργειας). Κάθε φορά που το alarm γίνεται triggered η εφαρμογή στέλνει ένα request στο cloud endpoint με τα δεδομένα που χρειάζεται το μοντέλο για την είσοδο. Το μοντέλο προβλέπει ένα timestamp με βάση

αυτά τα δεδομένα εισόδου και το επιστρέφει στην εφαρμογή. Το timestamp αυτό ουσιαστικά είναι η πρόβλεψη του μοντέλου σχετικά με το πότε θα “έπρεπε” να είχε γίνει ένα event (κλείσιμο οθόνης) με βάση τα συγκεκριμένα δεδομένα εισόδου. Σε περίπτωση που αυτό το timestamp είναι μικρότερο από το τωρινό (ο χρήστης θα “έπρεπε” να είχε κλείσει το κινητό του νωρίτερα) η εφαρμογή βγάζει ένα notification στον χρήστη που του προτείνει να κλείσει το κινητό. Όταν ο χρήστης κλείσει το κινητό του το συγκεκριμένο alarm ακυρώνεται και θα ενεργοποιηθεί ξανά όταν ο χρήστης ξανανοίξει το κινητό του.

Έτσι η τελική αρχιτεκτονική του συστήματος μας είναι:



Μπορείτε να βρείτε τον κώδικας ολόκληρης της εφαρμογής μας στο github, σε αυτό το link: <https://github.com/NikoNakkan/Diaxytos>

\* όταν λέμε alarm δεν αναφερόμαστε σε ένα ξυπνητήρι στο κινητό του χρήστη αλλά σε ένα επαναλαμβανόμενο event όπως ορίζεται [εδώ](#)