ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σύστημα χαρτογράφησης και απεικόνισης δεδομένων με χρήση Mobile αισθητήρων και πλατφόρμας.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Καλλίγερος Γεώργιος

Επιβλέπουσα : Θεοδώρα Βαρβαρίγου
Καθηγήτρια ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την

……. ……. …….

Περίληψη

Ο σκοπός αυτής της διπλωματικής εργασίας ήταν η ανάπτυξη μιας εφαρμογής για κινητά τηλέφωνα με λειτουργικό σύστημα Android,η οποία κάνοντας χρήση του ενσωματωμένου Δέκτη GPS του τηλεφώνου καταγράφει την διαδρομή του χρήστη πάνω στο χάρτη και άλλα δεδομένα αθλητικού ενδιαφέροντος.
Έγινε επίσης ανάπτυξη μιας web πλατφόρμας και ιστοσελίδας σε Node.js και MongoDB, που μας επιτρέπει την ταυτοποίηση κάθε χρήστη και την αποθήκευση, προβολή και διαγραφή των διαδρομών του μέσα από την εφαρμογή στο κινητό τηλέφωνο και στην ιστοσελίδα του έργου.

Λέξεις κλειδιά :κινητό τηλέφωνο, Android,GPS, Node.js, MongoDB

Abstract

The purpose of my thesis is the development of a mobile application for Android OS that uses the GPS receiver of the device to draw user's course on the map and calculate some data of athletic interest.
Also we have developed a web app and a website in Node.js and MongoDB, which allows each user of the app to save view edit and delete his/her tracks either from the app or from the website.

Keywords: mobile phone, Android, GPS, Node.js, MongoDB

Πίνακας περιεχομένων

Λίστα εικόνων

Λίστα πινάκων

Λίστα σχημάτων

1. Εισαγωγή

Στην εποχή μας τα έξυπνα κινητά τηλέφωνα έχουν τεραστία απήχηση στην αγορά και τείνουν να αντικαταστήσουν τα απλά κινητά τηλέφωνα. Μαζί με τα έξυπνα κινητά έχουν γνωρίσει ιδιαίτερη άνθηση και η εφαρμογές καταγραφής γεωγραφικών και αθλητικών δεδομένων φτάνοντας σε σημείο να αποτελούν φθηνή και αξιόλογη εναλλακτική στις εξειδικευμένες συσκευές καταγραφής.

Οι εφαρμογές καταγραφής παρά την μεγάλη επιτυχία που γνωρίζουν, εμπεριέχουν προβλήματα. Τα προβλήματα αυτά είναι τεχνικής αλλά και ηθικής-νομικής φύσεως. Ένα από τα πολλά τεχνικά θέματα που υπάρχουν στις γνωστότερες εφαρμογές του χώρου (που είναι και αυτές που δοκιμάστηκαν στα πλαίσια της εργασίας) είναι η ταχύτητά τους και γενικότερα η χρήση που κάνουν στους πόρους του συστήματος. Βέβαια αυτό είναι επακόλουθο της αθρόας προσθήκης χαρακτηριστικών. Το πρόβλημα των επιδόσεων κάνει τις περισσότερες εφαρμογές πρακτικά μη χρηστικές σε συσκευές μικρών δυνατοτήτων, κάτι που αποτελεί μείζων θέμα.

Ένα ακόμα θέμα που έχει προκαλέσει αρκετές συζητήσεις και προβληματισμό είναι το κατά πόσο οι εφαρμογές αυτές σέβονται την ιδιωτικότητα του χρήστη. Εκ των πραγμάτων για να λειτουργήσουν αυτές οι εφαρμογές απαιτείται πρόσβαση σε αισθητήρες που μπορούν να παρέχουν ευαίσθητα προσωπικά δεδομένα στο δημιουργό της εφαρμογής και τα οποία μπορούν να χρησιμοποιηθούν για ποικίλους σκοπούς εκτός της λειτουργίας της εφαρμογής. Για παράδειγμα, όλες εφαρμογές αυτού του τύπου απαιτούν πρόσβαση στο GPS, ενώ πολλές διαθέτουν μία υπηρεσία, η οποία τρέχει στο παρασκήνιο συνεχώς ακόμη και χωρίς ο χρήστης να εκκινήσει την εφαρμογή. Τεχνικά λοιπόν είναι πολύ εύκολο η εταιρεία ανάπτυξης να έχει δεδομένα για την θέση του χρήστη ανά πάσα στιγμή. Οι περισσότερες εφαρμογές διαθέτουν ένα κείμενο (που δηλώνει ρητά τι είδους δεδομένα συλλέγει η εφαρμογή και πώς μπορεί να τα

χρησιμοποιήσει), το οποίο ο χρήστης πρέπει να αποδεχτεί για τη χρήση της εφαρμογής. Βέβαια η τήρηση των συμφωνηθέντων εξαρτάται κυρίως από την εταιρεία ανάπτυξης και επαφίεται στο χρήστη η επιλογή του αν είναι έμπιστη ή όχι.

Η παρούσα διπλωματική εστιάζει στα παρακάτω θέματα που αφορούν τις εφαρμογές καταγραφής. Η δομή της είναι η εξής :

- *Κεφάλαιο 2*
  Εισαγωγή στα έξυπνα κινητά και στο οικοσύστημα των εφαρμογών τους .Εισαγωγή στις τεχνολογίες και εφαρμογές καταγραφής.
- *Κεφάλαιο 3*
  Παρουσίαση διεπαφής και λειτουργικότητας δημοφιλέστερων εφαρμογών καταγραφής.
- *Κεφάλαιο 4*
  Παρουσίαση αθλητικών δεδομένων αθλητικού ενδιαφέροντος και ανάλυση των αισθητήρων καταγραφής ,προβλήματα και λύσεις.
- *Κεφάλαιο 5*
  Παρουσίαση της αρχιτεκτονικής της εφαρμογής και της πλατφόρμας που αναπτύξαμε .
- *Κεφάλαιο 6*
  Ανάλυση κυριότερων στοιχείων κώδικα της εφαρμογής και της πλατφόρμας που αναπτύξαμε.
- *Κεφάλαιο 7*
  Παρουσίαση της διεπαφής και της λειτουργικότητας της εφαρμογής που αναπτύξαμε .
- *Κεφάλαιο 8*
  Παρουσίαση μελλοντικών επεκτάσεων σε εφαρμογή και πλατφόρμα.

2. Έξυπνα κινητά και apps

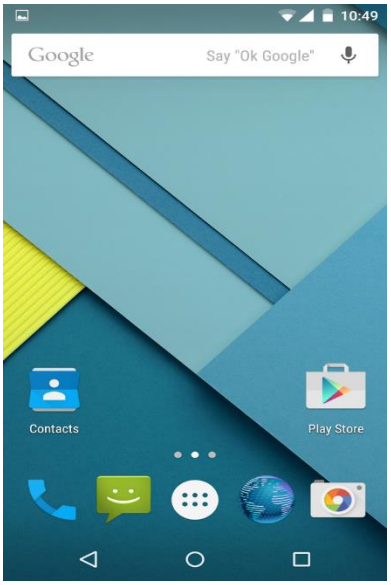
2.1 Εισαγωγή

Τα έξυπνα κινητά τηλέφωνα (Smartphones) είναι ένα είδος κινητών τηλεφώνων, τα οποία έχουν περισσότερες δυνατότητες από ένα απλό κινητό τηλέφωνο(Feature phone). Τα όρια μεταξύ απλού και έξυπνου κινητού τηλεφώνου πολλές φορές είναι αρκετά δυσδιάκριτα. Μερικά γενικά χαρακτηριστικά που διακρίνουν τα έξυπνα κινητά είναι τα εξής:

- Διαθέτουν εξελιγμένο λειτουργικό σύστημα, το οποίο μπορεί να τρέξει εφαρμογές από τρίτους κατασκευαστές.
- Διαθέτουν γκάμα εξελιγμένων αισθητήρων, όπως GPS,κάμερα, πυξίδα, επιταχυνσιόμετρο κ.α.
- Διαθέτουν σύνδεση στο διαδίκτυο.

Από τα παραπάνω θα ορίζαμε ένα smartphone ως ένα συνδυασμό υπερφορητού προσωπικού υπολογιστή και κινητού τηλεφώνου. Ένα σημαντικότατο κομμάτι ενός smartphone ως φορητού υπολογιστή είναι το λειτουργικό του σύστημα. Από τα πολλά λειτουργικά συστήματα για κινητά που κυκλοφορούν, μόνο τα παρακάτω τρία έχουν ποσοστό αγοράς άξιο αναφοράς.

- *Android*
  Ένα ανοιχτού κώδικα λειτουργικό, που βασίζεται στον πυρήνα του Linux. Αναπτύσσεται από την *Handset Alliance|Open Alliance* (παλιότερα από την Google) για την δημιουργία εφαρμογών διαθέτει ένα πλήρες εργαλείο σε γλώσσα προγραμματισμού Java. Το Android είναι διαθέσιμο σε μεγάλη γκάμα κατασκευαστών κινητών. Το Android κατέχει επίσης το μεγαλύτερο μερίδιο αγοράς, καθώς όσον αφορά τον αριθμό των συσκευών, υπολογίζεται πως το 80% των χρηστών smartphone έχει συσκευή Android.



1 Αρχική οθόνη από Android 5 (Lollip)

- *iOS*
  Το iOS είναι το λειτουργικό σύστημα που τρέχουν οι δημοφιλείς κινητές συσκευές της Apple, μεταξύ των οποίων και τα κινητά *iPhone*. Η Apple διαθέτει στους προγραμματιστές ένα πλήρες σετ εργαλείων για δημιουργία εφαρμογών σε γλώσσα Objective C. Ο περιορισμός του iOS στα τηλέφωνα της Apple είναι υπαίτιος για το σχετικά μειωμένο ποσοστό αγοράς, το οποίο υπολογίζεται στο 13% περίπου των χρηστών (και παραδόξως στο 80-90% των καθαρών κερδών της αγοράς).

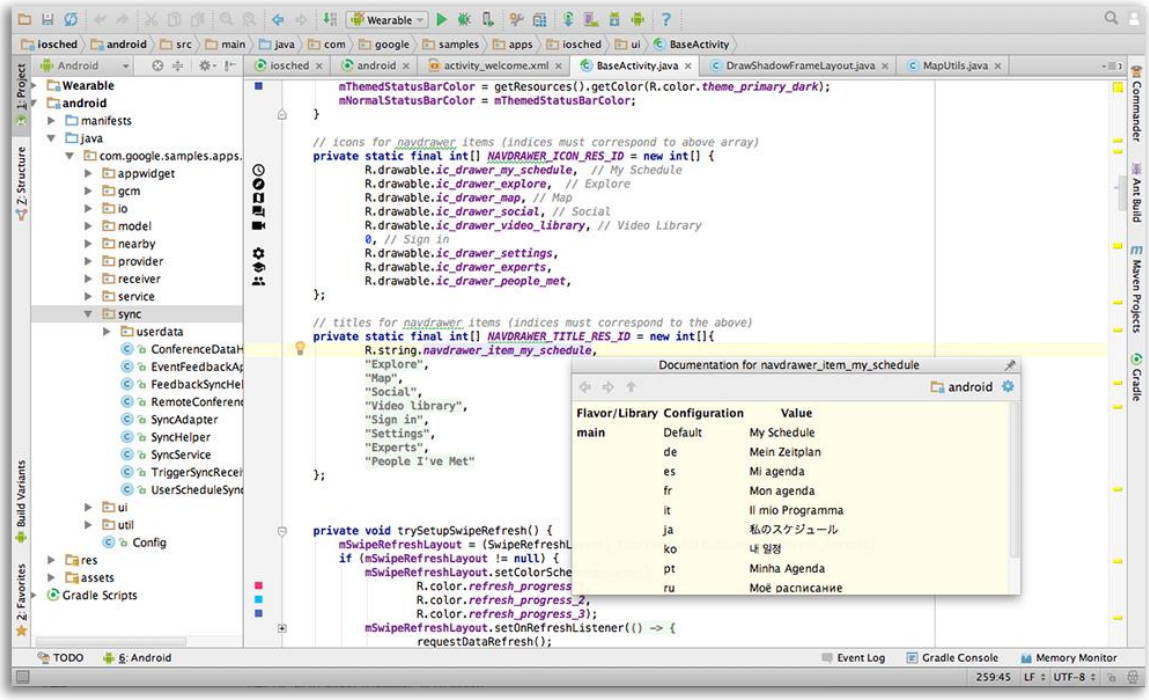
2 Αρχική οθόνη από iOS 7

- *Windows Phone*
  Τα *Windows Phone* είναι δημιούργημα της Microsoft και στην ουσία αποτελούν την επέκταση του γνωστού λειτουργικού για προσωπικούς υπολογιστές σε κινητές συσκευές. Η εταιρεία προσφέρει πλήρες περιβάλλον ανάπτυξης εφαρμογών που υποστηρίζει αρκετές γλώσσες προγραμματισμού, μεταξύ των οποίων οι C#,Visual Basic και άλλες. Το μερίδιο αγοράς είναι μικρό και υπολογίζεται περίπου στο 2-3%.

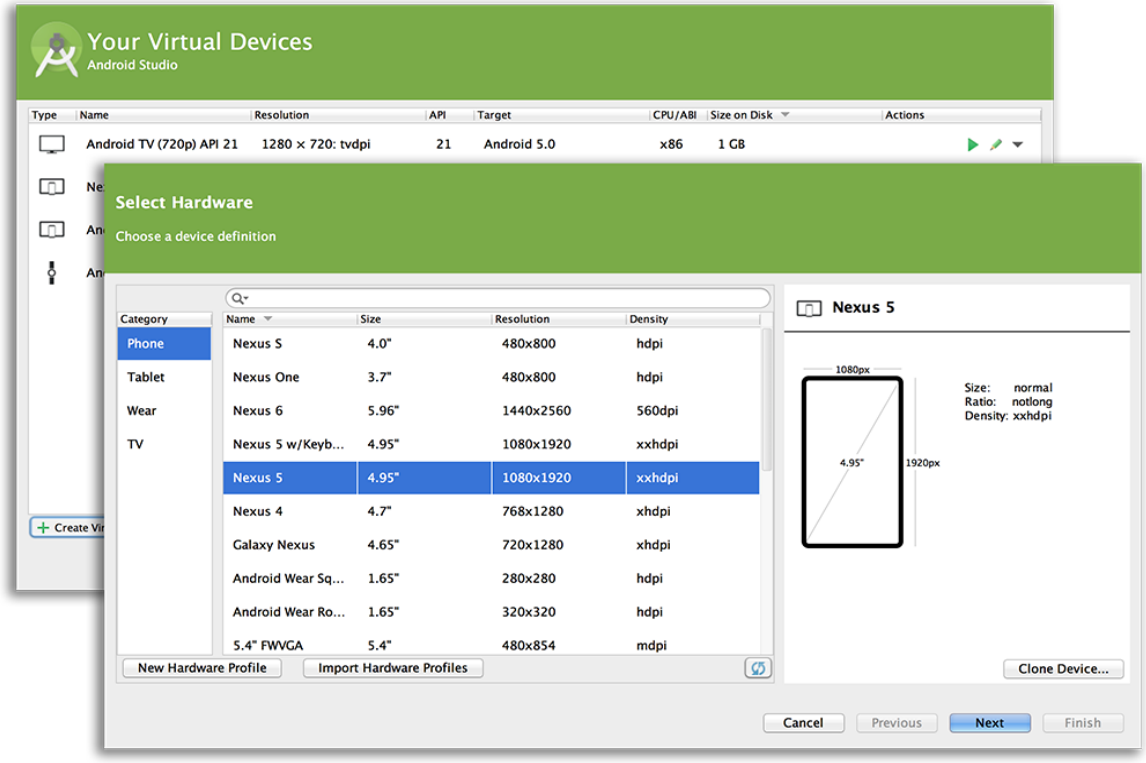
3 Αρχική οθόνη από Windows Phone 8.1

Ένα από τα ισχυρότερα χαρακτηριστικά των smartphones είναι η δυνατότητα προσθήκης εφαρμογών σε αυτά. Μέσω των εφαρμογών τα smartphones αποκτούν διευρυμένες δυνατότητες. Οι εταιρείες που αναπτύσσουν τα λειτουργικά συστήματα γνωρίζοντας ότι η πληθώρα εφαρμογών κάνει τα προϊόντα τους ελκυστικότερα ,διευκολύνουν τη δημιουργία και τη διανομή εφαρμογών γι' αυτά. Η δημιουργία των εφαρμογών διευκολύνεται συνήθως με την διάθεση στους προγραμματιστές εύχρηστων και ισχυρών εργαλείων ανάπτυξης λογισμικού. Η διάθεση των εφαρμογών στον τελικό χρήστη γίνεται ευκολότερη με τη δημιουργία ειδικών ηλεκτρονικών καταστημάτων.

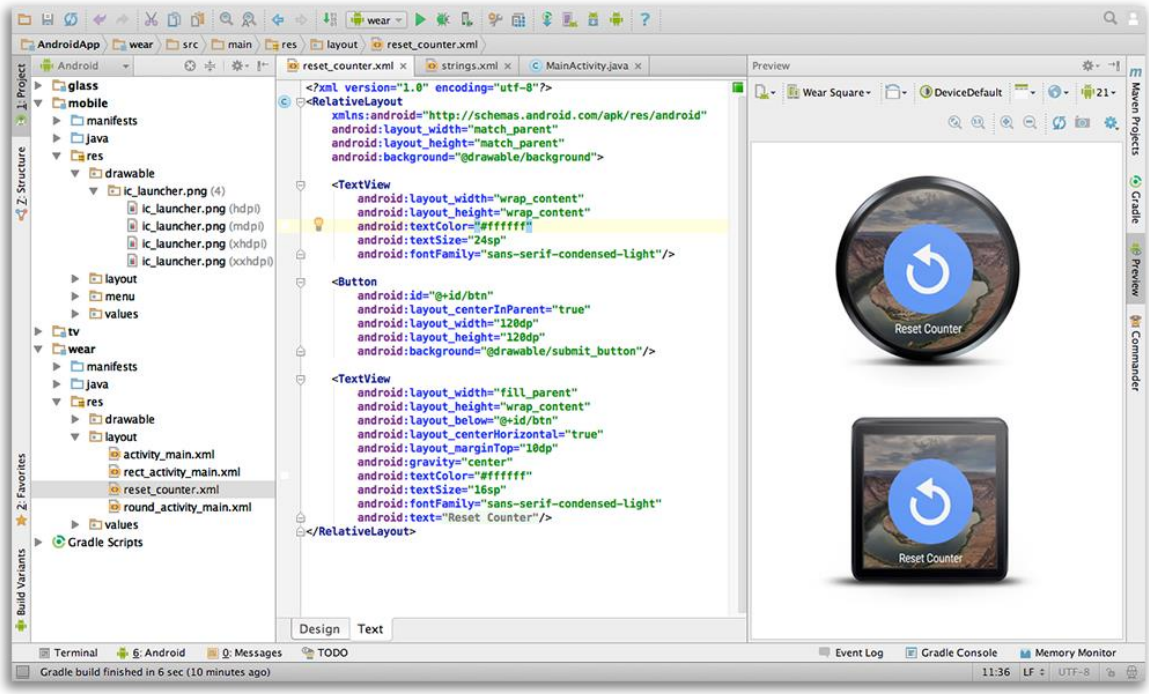
Στην περίπτωση του Android που εστιάζουμε στην παρούσα εργασία, υπάρχουν και οι δυο παραπάνω προϋποθέσεις για την δημιουργία και διάθεση των εφαρμογών. Για την δημιουργία των εφαρμογών η Google έχει κάνει διαθέσιμο ένα εργαλείο που παρέχει ό,τι απαιτεί η διαδικασία , το Android Studio. Το συγκεκριμένο εργαλείο διαθέτει εργαλεία συγγραφής και αποσφαλμάτωσης κώδικα και σχεδιασμού διεπαφών. Επίσης παρέχονται εργαλεία προσομοίωσης συσκευών Android για δοκιμή και αποσφαλμάτωση των εφαρμογών, χωρίς να χρειάζεται πραγματική συσκευή.



4 Το εργαλείο συγγραφής κώδικα του Android Studio περιέχει χρήσιμες δυνατότητες.

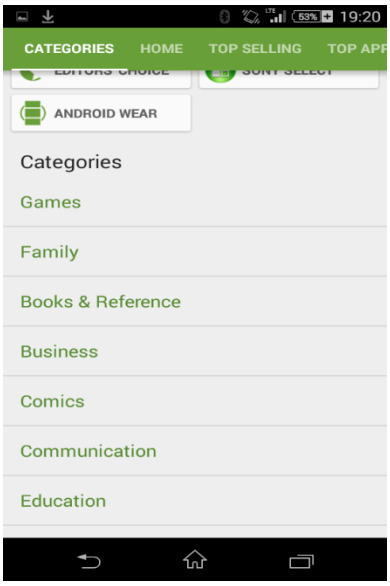


5 Το εργαλείο που εργαλείο δημιουργίας εικονικών συσκευών του Android Studio

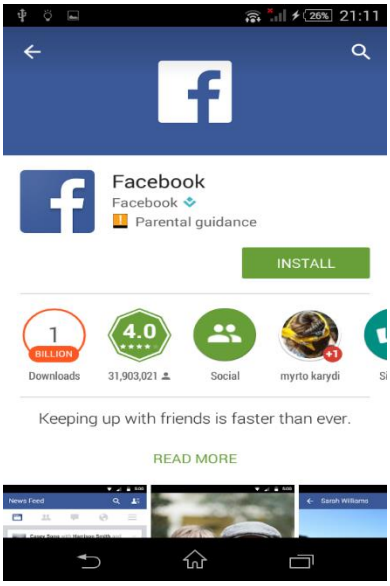


6 Το εργαλείο σχεδιασμού διεπαφών του Android Studio

Για τη διάθεση των εφαρμογών στους χρήστες υπάρχουν αρκετά καταστήματα εφαρμογών. Συνήθως κάθε μεγάλος κατασκευαστής συσκευών διαθέτει δικό του κατάστημα, το μεγαλύτερο όμως είναι αυτό που έχει δημιουργήσει η Google και ονομάζεται *Google Play*. Το συγκεκριμένο κατάστημα αποτελεί και το επίσημο κατάστημα εφαρμογών του λειτουργικού και παρέχει δωρεάν και μη εφαρμογές . Συνολικά οι εφαρμογές που διατίθενται μέσω του Google Play έχουν ξεπεράσει το 1 εκατομμύριο, ενώ έχουν γίνει πάνω από 50 δισεκατομμύρια κατεβάσματα εφαρμογών(αριθμοί του 2013). Οι εφαρμογές είναι βολικά κατηγοριοποιημένες, ανάλογα με τον τομέα στον οποίο ανήκουν ενώ και η αναζήτηση είναι πολύ εύχρηστη και αποτελεσματική. Το Google Play διαθέτει τις εφαρμογές είτε μέσω της ιστοσελίδας του είτε μέσω της ειδικής εφαρμογής για Android.Το συγκεκριμένο κατάστημα είναι δωρεάν για τους χρήστες. Οι προγραμματιστές προκειμένου να διαθέσουν τις εφαρμογές τους, πρέπει να καταβάλλουν 25 δολάρια για να ανοίξουν λογαριασμό προγραμματιστή καθώς και το 30% της αξίας από κάθε εφαρμογή που πουλάνε.



7 Οι εφαρμογές στο Google Play είναι κατηγοριοποιημένες για ευκολότερη περιήγηση.



8 Η εγκατάσταση μιας εφαρμογής στο Google Play γίνεται με το πάτημα ενός πλήκτρου.

2.2 Αθλητισμός και τεχνολογία καταγραφής δεδομένων

Ο αθλητισμός αποτελεί πλέον ένα πολύ σημαντικό κομμάτι της σύγχρονης ζωής. Αθλητικές δραστηριότητες, όπως τρέξιμο, περπάτημα ,ποδηλασία έχουν γίνει ιδιαίτερα δημοφιλείς κυρίως σε μη επαγγελματίες αθλητές. Με την άνοδο της δημοτικότητας τέτοιου τύπου δραστηριοτήτων άρχισε να γίνεται ορατή η χρησιμότητα εύρεσης ενός τρόπου καταγραφής αθλητικών και γεω-

γραφικών δεδομένων για κάθε δραστηριότητα. Τέτοιου τύπου δεδομένα είναι πολύ χρήσιμα για κάθε αθλητή, καθώς του δίνεται η δυνατότητα παρακολούθησης των επιδόσεων και της προόδου του με το χρόνο αλλά και για πολλούς ακόμα λόγους.

Τη λύση στο πρόβλημα της καταγραφής μέχρι πρόσφατα έδιναν ειδικές συσκευές που διέθεταν τους κατάλληλους αισθητήρες (GPS, επιταχυνσιόμετρο, αισθητήρας καρδιακών παλμών κ.α) για την καταγραφή των δεδομένων κι είχαν μικρό μέγεθος και βάρος. Οι συσκευές αυτές δεν είχαν ιδιαίτερα μεγάλη απήχηση σε χρήστες πέρα από τους επαγγελματίες ή τους προχωρημένους ερασιτέχνες αθλητές. Με την έλευση των έξυπνων κινητών, τα οποία διέθεταν εξελιγμένους αισθητήρες έγινε εφικτή η δημιουργία εφαρμογών καταγραφής χωρίς να χρειάζεται επιπλέον συσκευή (τουλάχιστον για την καταγραφή των περισσότερων δεδομένων).

2.2 Ιστορικά στοιχεία και τάσεις της αγοράς

Μία από τις πρώτες εφαρμογές καταγραφής ήταν η Sports Tracker, αρχικά γνωστή ως Nokia Sports Tracker. Στην αρχή δημιουργήθηκε για το λειτουργικό σύστημα Symbian της Nokia. Αν και ρηξικέλευθη για την εποχή της, η εφαρμογή δε γνώρισε μεγάλη αναγνωρισιμότητα, κυρίως λόγω της μικρής δημοτικότητας που είχαν τα κινητά τηλέφωνα που μπορούσαν να την τρέξουν αλλά και της μικρής δημοτικότητας των έξυπνων κινητών εν γένει. Η μεγάλη έκρηξη στον τομέα ήρθε τα έτη 2007-2008, μαζί με την αύξηση της δημοτικότητας των Smartphones και συγκεκριμένα των λειτουργικών iOS και Android.Με την αύξηση των χρηστών και κατά συνέπεια των κερδών, οι εφαρμογές τέτοιου τύπου άρχισαν να αυξάνονται και να βελτιώνονται σε ποιότητα λόγω του ανταγωνισμού.

Ο ανταγωνισμός που πλέον υπάρχει στον κλάδο αυτόν έχει ως αποτέλεσμα οι περισσότερες επιτυχημένες εφαρμογές να μην είναι απλοί καταγραφείς δεδομένων αλλά να προσφέρουν όλο και περισσότερα επιπλέον χαρακτηριστικά, με συνέπεια να διαφαίνονται τάσεις στην αγορά. Οι κυριότερες τάσεις που έχουν διαμορφωθεί στην αγορά είναι οι παρακάτω:

- *Gamification*
  Ο όρος *Gamification* αναφέρεται στην διαδικασία που ακολουθείται στην μετατροπή ενός συστήματος ή μιας διαδικασίας, έτσι ώστε να γίνει περισσότερο διασκεδαστική. Το παραπάνω έχει ως στόχο να γίνει η διαδικασία πιο ευχάριστη για το χρήστη και ο χρήστης να γίνει παραγωγικότερος. Στην περίπτωσή μας αυτό σημαίνει πως οι εφαρμογές εισάγουν στοιχεία ανταγωνισμού με άλλους χρήστες για να κάνουν την άσκηση παιχνίδι. Άλλες εφαρμογές πετυχαίνουν *gamification* κάνοντας το χρήστη να ασκείται ενώ παράλληλα προσπαθεί να πετύχει κάποιους στόχους στο περιβάλλον ενός ηλεκτρονικού παιχνιδιού (π.χ. να ξεφύγει από τα τέρατα που τον κυνηγούν).

- *Κοινωνικά χαρακτηριστικά*
  Η μεγάλη απήχηση που έχουν τα κοινωνικά δίκτυα στις μέρες μας δεν έχει αφήσει ανεπηρέαστες τις εφαρμογές του είδους. Πράγματι, δεν είναι λίγες οι εφαρμογές που ενσω-

ματώνουν στοιχεία, όπως λίστα φίλων, αναρτήσεις, σχόλια .Πολλές εφαρμογές έχουν εξελιχθεί τόσο πολύ σε αυτόν τον τομέα, που αποτελούν αυτοτελή κοινωνικά δίκτυα.

- *Συνδυασμός εφαρμογών με αισθητήρες που μπορούν να φορεθούν.*
  Οι αισθητήρες των κινητών τηλεφώνων εκ των πραγμάτων δεν μπορούν να μετρήσουν όλα τα δεδομένα που χρειαζόμαστε (π.χ. καρδιακοί παλμοί). Σε αυτήν την περίπτωση υπάρχουν ειδικά περιφερειακά που συνεργάζονται με εφαρμογές και μας παρέχουν πληθώρα επιπλέον δεδομένων.

3. Mobile εφαρμογές καταγραφής

Η καταγραφή των αθλητικών δραστηριοτήτων δεν αποτελεί πρόσφατη τάση, καθώς οι αθλητές ανέκαθεν κατέγραφαν δεδομένα για τις δραστηριότητες και τις επιδόσεις τους. Ακόμα και η ηλεκτρονική καταγραφή υπάρχει σε μορφή κινητών και μη συσκευών τουλάχιστον από τη δεκαετία του 90.Μέχρι πρόσφατα τέτοιες συσκευές ήταν συνήθως ακριβές και απευθύνονταν σε επαγγελματίες, στις μέρες μας όμως η μείωση των τιμών και του μεγέθους τις έχουν κάνει πιο προσιτές.

Οι κινητές εφαρμογές καταγραφής αθλητικών δραστηριοτήτων αποτελούν σχετικά καινούρια τάση. Μέχρι πρότινος ο χρήστης που ήθελε να έχει μια τέτοια δυνατότητα έπρεπε να στραφεί στη χρήση ειδικής συσκευής γι' αυτό το σκοπό. Στη συνέχεια η καθιέρωση συσκευών (Android και μη), οι οποίες διέθεταν εξελιγμένους αισθητήρες, όπως GPS και επιταχυνσιόμετρο έκανε δυνατή την ανάπτυξη εφαρμογών καταγραφής που δεν απαιτούσαν εξωτερικό περιφερειακό κάνοντάς τες έτσι πιο προσιτές στον μέσο χρήστη. Τέτοιου τύπου εφαρμογές κατέστησαν την αναλυτική καταγραφή δραστηριοτήτων απλή και φθηνή υπόθεση για κάθε χρήστη. Στις μέρες μας αυτές οι εφαρμογές έχουν γίνει πολύ δημοφιλείς, εκ των οποίων μάλιστα οι τέσσερεις γνωστότερες (σε Android)υπολογίζεται ότι έχουν πάνω από σαράντα εκατομμύρια χρήστες συνολικά.

Η μεγάλη δημοτικότητα των εφαρμογών καταγραφής έχει ως αποτέλεσμα την ύπαρξη πολλών λύσεων, συνεπώς ο τομέας αυτός αποτελεί ήδη μια πολύ ανταγωνιστική αγορά, στην οποία έχουν δραστηριοποιηθεί αρκετές εταιρείες λογισμικού ακόμα και μεγάλες πολυεθνικές. Αυτό έχει ως αποτέλεσμα οι εφαρμογές που προκύπτουν να έχουν ιδιαιτέρως επαγγελματικές δυνατότητες και ως εκ τούτου ο ανταγωνισμός να είναι μεγάλος.

Ο μεγάλος ανταγωνισμός έχει ως συνέπεια κάθε εφαρμογή να προσπαθεί να ξεχωρίσει με τα χαρακτηριστικά που προσφέρει αλλά και να αντιγράφει τα χαρακτηριστικά που προσφέρουν οι ανταγωνιστές. Όμως, εξαιτίας της συγκεκριμένης πρακτικής όλες οι εφαρμογές προσφέρουν σε γενικές γραμμές τα ίδια χαρακτηριστικά, ενώ οι συνεχείς προσθήκες που κάνουν (για να μην υστερούν σε σχέση με τους ανταγωνιστές) καθιστούν τις εφαρμογές αυτές αρκετά πολύπλοκες και σπάταλες ως προς τους πόρους του συστήματος.

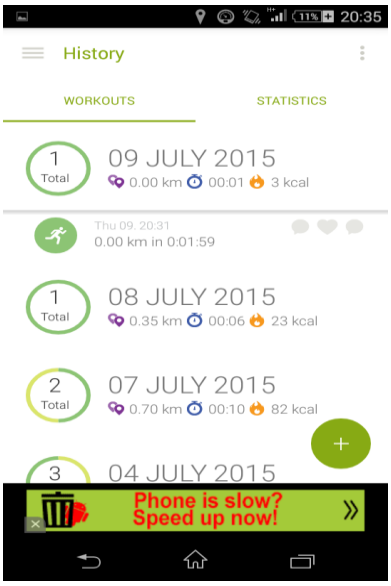
Για να κάνουμε καλύτερη χαρτογράφηση της αγοράς επιλέχθηκαν οι τέσσερις από τις δημοφιλέστερες εφαρμογές στον τομέα αυτόν, δοκιμάστηκαν ενδελεχώς και μετρήθηκε η απόδοσή τους με αρκετούς τρόπους. Η δοκιμή αφορούσε τα προσφερόμενα χαρακτηριστικά, την συνολική εμπειρία χρήσης και διάφορους τομείς των επιδόσεων κάθε εφαρμογής. Στη συνέχεια του κεφαλαίου θα καταγράψουμε τις εντυπώσεις από τις δοκιμές που κάναμε. Πιο συγκεκριμένα, θα παρουσιάσουμε αναλυτικά κάθε εφαρμογή και θα κάνουμε συγκριτικά τεστ επιδόσεων.

3.1 Endomondo

Όνομα	Endomondo
Δημιουργός	Endomondo.com
Μέγεθος	20.69 MB

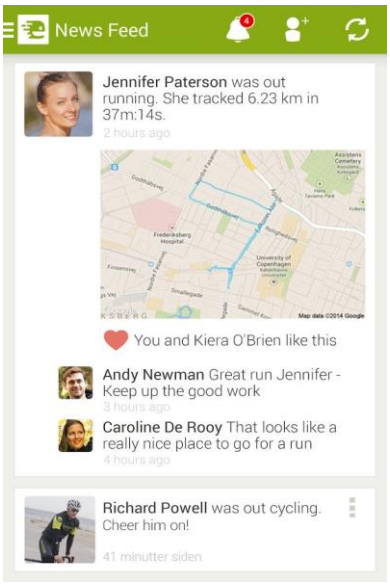
Το Endomondo αποτελεί μία από τις πιο γνωστές και πλήρεις λύσεις στο συγκεκριμένο τομέα. Μπορεί να καταγράψει πολλά είδη αθλητικών δραστηριοτήτων όπως τρέξιμο, περπάτημα, ποδηλασία και άλλα. Αξιοσημείωτη είναι η χρήση ήχου στην εφαρμογή, που πληροφορεί το χρήστη για διάφορα γεγονότα κατά τη διάρκεια της δραστηριότητας (πχ. Όταν φτάσει το 1ο χιλιόμετρο).Για κάθε δραστηριότητα η εφαρμογή καταγράφει πληθώρα αθλητικών δεδομένων.

9 Κεντρική οθόνη Edomondo κατά τη διάρκεια καταγραφής τρεξίματος



10 Οθόνη προβολής αποθηκευμένων δραστηριοτήτων του Endomondo

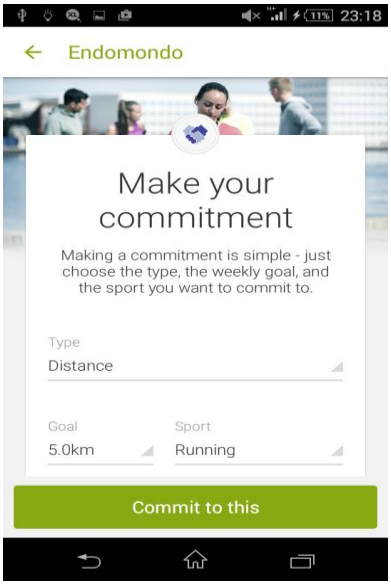
Το Endomondo είναι αρκετά ανεπτυγμένο και στο κομμάτι της κοινωνικής δικτύωσης. Συγκεκριμένα, κάθε χρήστης μπορεί να γίνει φίλος με άλλους χρήστες και να αλληλεπιδράσει με αυτούς με διάφορους τρόπους, όπως για παράδειγμα να σχολιάσει μια δραστηριότητά τους.



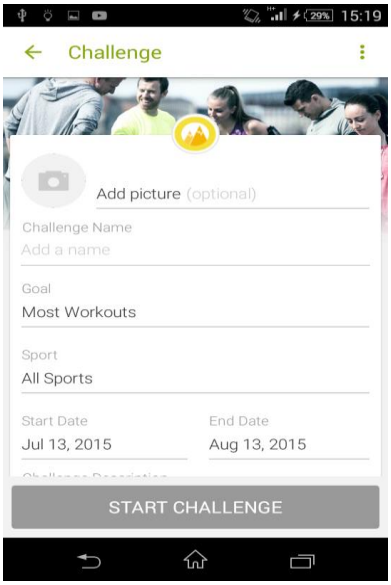
11 Τα κοινωνικά χαρακτηριστικά του Endomondoε (χρονολόγιο δραστηριοτήτων) .

Αρκετά ενδιαφέρον χαρακτηριστικό, το οποίο συμβάλλει στην παρακίνηση του χρήστη για άσκηση, είναι οι προκλήσεις (challenges).Μέσω των προκλήσεων κάθε χρήστης μπορεί να προκαλέσει τους φίλους του ή να προκληθεί από αυτούς σε ανταγωνιστικές δοκιμασίες. Για παράδειγμα, μπορεί να δημιουργηθεί μια πρόκληση στην οποία νικάει ο χρήστης που θα τρέξει τα περισσότερα χιλιόμετρα σε ένα συγκεκριμένο χρονικό διάστημα.
Ένα ακόμα χαρακτηριστικό του Endomondo που διαδραματίζει αρκετά μεγάλο ρόλο στην παρακίνηση του χρήστη είναι οι δεσμεύσεις (commitments). Χρησιμοποιώντας τες ο χρήστης

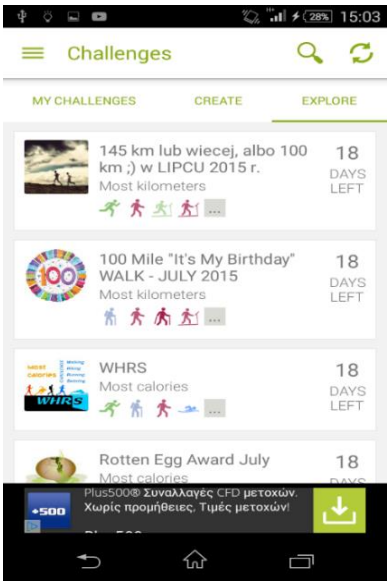
μπορεί να θέσει προσωπικούς στόχους στην άσκηση ,οι οποίοι πρέπει να εκπληρωθούν σε χρονικό διάστημα μιας εβδομάδας (πχ. να τρέχει 5 χιλιόμετρα την εβδομάδα ή να τρέχει 3 ώρες την εβδομάδα).



12 Οθόνη δημιουργίας δέσμευσης

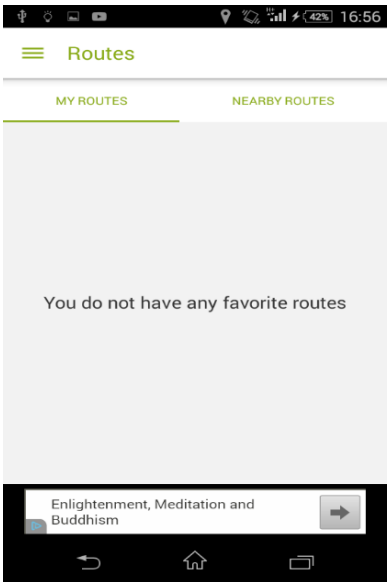


13 Οθόνη δημιουργίας πρόκλησης

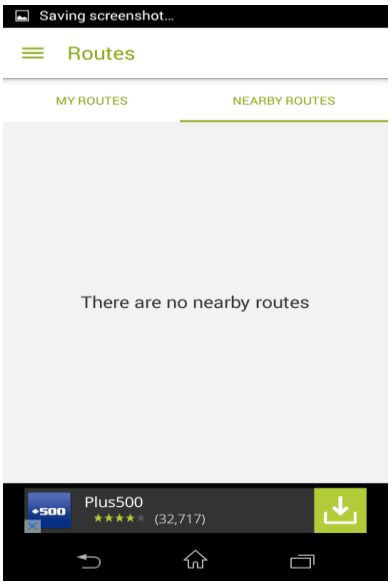


14 Ο χρήστης μπορεί να αναλάβει προκλήσεις που έχουν δημιουργηθεί από άλλους.

Στο Endomondo κάθε χρήστης μπορεί επίσης να αποθηκεύσει δικές του διαδρομές, τις οποίες μπορεί να ακολουθήσει σε μια δραστηριότητά του στο μέλλον. Ακόμη, έχει τη δυνατότητα να αναζητήσει ενδιαφέρουσες διαδρομές που έχουν δημιουργήσει άλλοι (η αναζήτηση γίνεται με βάση την εγγύτητα της διαδρομής στη θέση του χρήστη).



15 Ο χρήστης επιλέγει διαδρομές που θα προσθέσει στις αγαπημένες του.



16 Ο χρήστης μπορεί να ψάξει για διαδρομές κοντά στη θέση του (δημιουργημένες από άλλους).

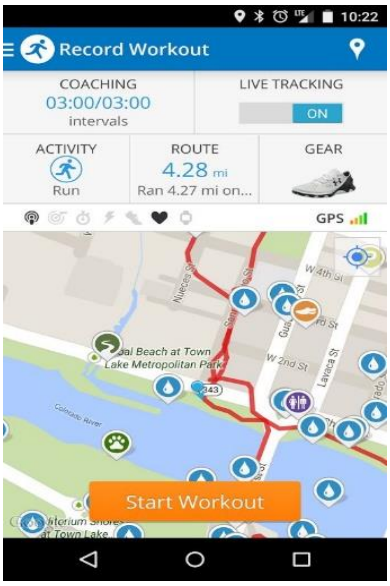
Το Endomondo προσφέρει τη βασική του έκδοση με όλα τα χαρακτηριστικά που αναλυθήκαν παραπάνω δωρεάν, υποστηριζόμενο από διαφημίσεις. Εκτός από τη δωρεάν υπάρχει και η επί πληρωμή έκδοση (Premium), η οποία προσφέρει κάποια ακόμα χαρακτηριστικά και αφαιρεί τις διαφημίσεις.
Τα επιπλέον χαρακτηριστικά της επί πληρωμή έκδοσης συνοπτικά περιλαμβάνουν:

- Αναλυτικά στατιστικά
- Πλάνα προπόνησης δημιουργημένα κατά παραγγελία
- Ζωντανή αναμετάδοση της δραστηριότητας του χρήστη στους φίλους του (live tracking)

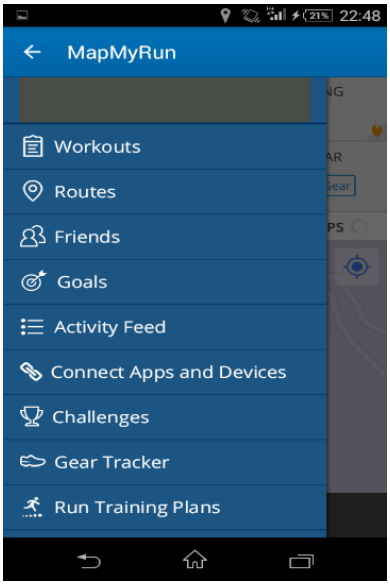
3.2 MapMyRun

Όνομα	MapMyRun
Δημιουργός	MapMyFitness, Inc
Μέγεθος	20.96 MB

Το MapMyRun είναι μια εφαρμογή, η οποία παρουσιάζει πολύ μεγάλη συνάφεια με το Endomondo, που παρουσιάστηκε στο προηγούμενο κεφάλαιο καθώς εκτός από την μεγάλη ομοιότητα στη διεπαφή έχουν και πολλά κοινά χαρακτηριστικά.
Στον τομέα της διεπαφής το MapMyRun διαθέτει μια κεντρική οθόνη για την καταγραφή της δραστηριότητας η οποία προβάλει τα αθλητικά δεδομένα σε πραγματικό χρόνο. Σημειώνεται επίσης πως είναι δυνατή η επιλογή καταγραφής πολλών ειδών δραστηριοτήτων. Η πλοήγηση στην εφαρμογή γίνεται μέσα από ένα εύχρηστο πλαϊνό μενού, με το οποίο μπορούμε να μεταβούμε σε όλα τα μέρη της εφαρμογής.



17 Η οθόνη καταγραφής του MapMyRun



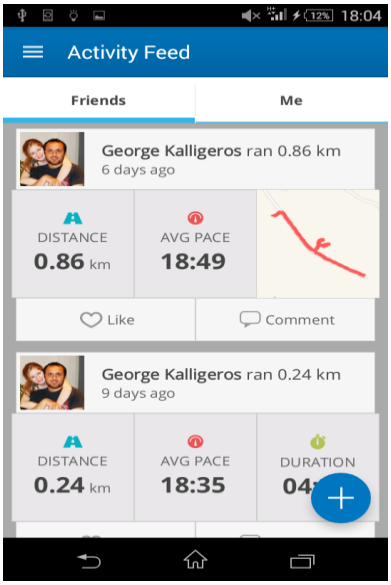
18 Το πλαϊνό μενού της εφαρμογής

19 Λίστα προβολής αποθηκευμένων δραστηριοτήτων

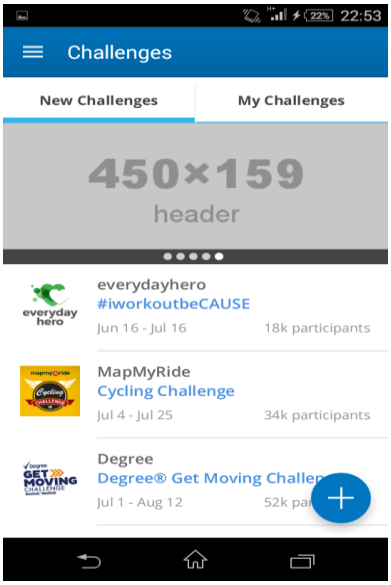
Το MapMyRun διαθέτει μια πλήρη γκάμα κοινωνικών χαρακτηριστικών, μέσω των οποίων ο χρήστης έχει τη δυνατότητα να:

- προσθέσει φίλους
- δει τις δραστηριότητές τους μέσα από ένα βολικό χρονολόγιο
- σχολιάσει και να δεχθεί σχόλια
- δημιουργήσει προκλήσεις και να προκαλέσει τους φίλους του σε αυτές

Όλα αυτά τα χαρακτηριστικά προσδίδουν στην εφαρμογή χαρακτήρα κοινωνικού δικτύου.

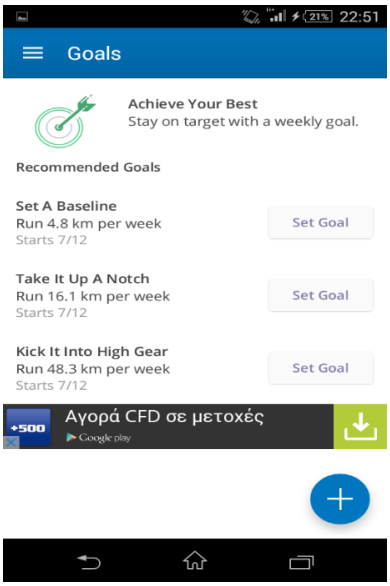


20 Χρονονολόγιο προβολής δραστηριοτήτων φίλων



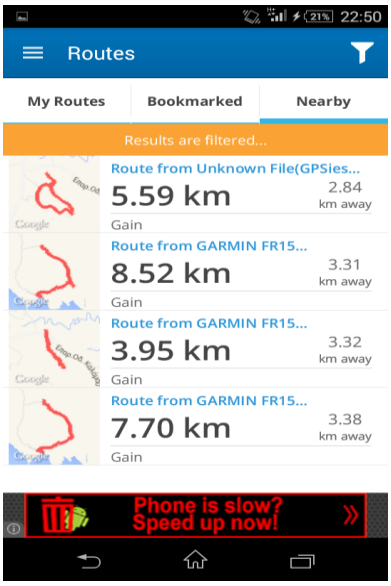
21 Οθόνη προβολής ενεργών προκλήσεων

Εκτός από τα challenges η συγκεκριμένη εφαρμογή διαθέτει ένα ακόμα χαρακτηριστικό, χρήσιμο για την παρακίνηση του χρήστη σε άσκηση, τους στόχους (Goals). Έτσι ο χρήστης μπορεί να θέτει στόχους ρυθμίζοντας αρκετές παραμέτρους (απόσταση, ταχύτητα, χρόνο κτλ) και να προσπαθεί να τους πετύχει.



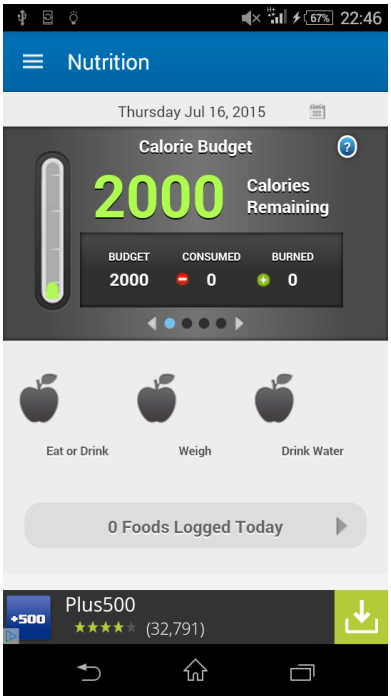
22 Ο χρήστης θέτει τους στόχους του, ενώ μπορεί να αναλάβει και κάποιους έτοιμους.

Ένα ακόμα χαρακτηριστικό που δε θα μπορούσε να λείπει είναι αυτό των προσχεδιασμένων διαδρομών. Ο χρήστης έχει τη δυνατότητα να δημιουργήσει μόνος του τις διαδρομές ή να χρησιμοποιήσει την αναζήτηση, για να βρει κάποια έτοιμη κοντά στην θέση του.



23 Ο χρήστης μπορεί να δημιουργήσει δικές του διαδρομές ή να βρει διαδρομές που είναι κοντά στη θέση του.

Ένα μοναδικό χαρακτηριστικό του MapMyRun μεταξύ των εφαρμογών που εξετάστηκαν σε αυτή την διπλωματική εργασία είναι ο καταγραφέας θερμίδων. Πιο συγκεκριμένα, δίνεται η δυνατότητα στο χρήστη να καταγράφει τις τροφές που κατανάλωσε μέσα στη μέρα. Η εφαρμογή υπολογίζει τις θερμίδες που αντιστοιχούν στις συγκεκριμένες τροφές και στη συνέχεια λαμβάνοντας υπόψη το βασικό μεταβολισμό του χρήστη και τις θερμίδες που καταναλώθηκαν με την άσκηση, του παρουσιάζει το ισοζύγιο θερμίδων του.



24 Το ισοζύγιο θερμίδων

Τα παραπάνω χαρακτηριστικά παρέχονται με την δωρεάν έκδοση του MapMyRun, η οποία στηρίζεται με διαφημίσεις μέσα στην εφαρμογή. Εκτός από την δωρεάν έκδοση υπάρχει και μια επί πληρωμή, η οποία ονομάζεται MVP και παρέχει επιπλέον αρκετά χαρακτηριστικά*, με τα κυριότερα από αυτά να είναι:

- *Live Tracking*
  Πρόκειται για ένα χαρακτηριστικό που επιτρέπει στο χρήστη να αναμεταδίδει ζωντανά την δραστηριότητά του, ώστε οι φίλοι του να μπορούν να τον βλέπουν ζωντανά μέσω της εφαρμογής.
- *Σχέδια προπόνησης (Coaching)*
  Επιτρέπει στο χρήστη να ακολουθεί διάφορα προκατασκευασμένα πλάνα προπόνησης.
- *Ανάλυση καρδιακού παλμού*
  Αν ο χρήστης διαθέτει τον κατάλληλο εξωτερικό αισθητήρα, τότε υπάρχει η δυνατότητα να του παρέχεται μια λεπτομερής ανάλυση του καρδιακού παλμού του.
- *Περισσότερα στατιστικά για την απόδοση του αθλητή*
- *Πίνακας κατάταξης μεταξύ όλων των χρηστών MVP*

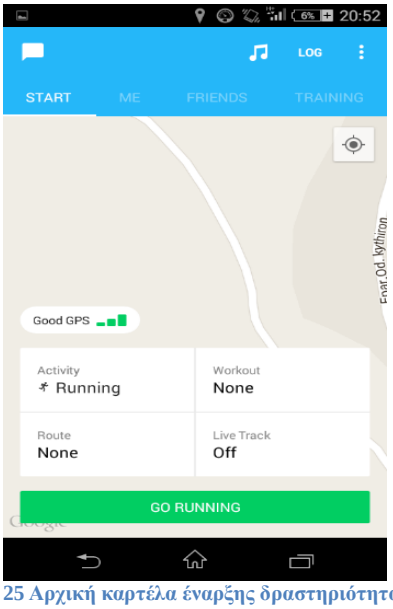
*Πλήρης λίστα με τα επιπλέον χαρακτηριστικά μπορεί να βρεθεί εδώ

3.3 Run Keeper

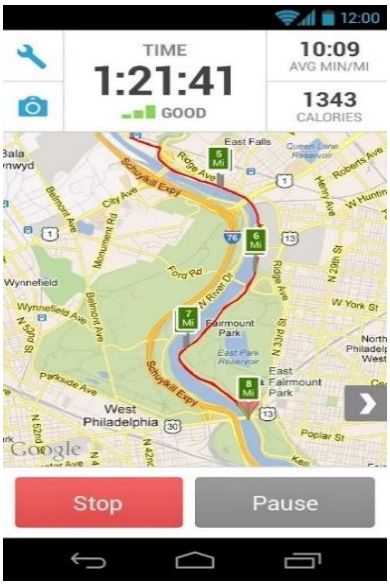
Όνομα	Run Keeper
Δημιουργός	FitnessKeeper, Inc
Μέγεθος	23,76 MB

Το RunKeeper αποτελεί σχεδιαστικά μια αρκετά διαφορετική εφαρμογή από τις υπόλοιπες που παρουσιάστηκαν, όσον αφορά τον τομέα της διεπαφής. Στον τομέα όμως των χαρακτηριστικών, δεν περιέχει κάτι παραπάνω από τις ανταγωνιστικές εφαρμογές.
Η εφαρμογή στην ουσία αποτελείται από τέσσερις καρτέλες, οι οποίες χωρίζουν όλο το περιεχόμενό της σε ενότητες. Οι καρτέλες είναι οι εξής:

- *Καρτέλα "Start"*
  Αποτελεί την κεντρική καρτέλα της εφαρμογής, μέσω της οποίας ξεκινάμε και σταματάμε την καταγραφή μιας δραστηριότητας. Τα είδη δραστηριοτήτων που μπορούν να επιλεγούν είναι πολλά (πχ τρέξιμο, περπάτημα, ποδηλασία κτλ.).

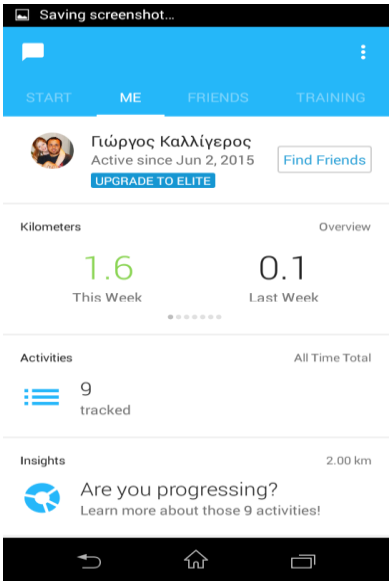


25 Αρχική καρτέλα έναρξης δραστηριότητας

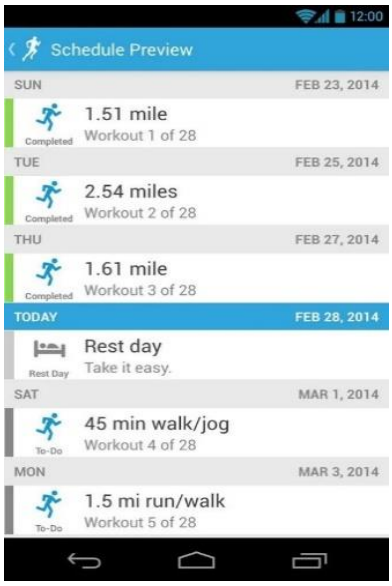


26 Οθόνη καταγραφής δραστηριότητας

- *Καρτέλα "Me"*
  Είναι η καρτέλα προβολής όλων των δεδομένων του χρήστη, όπως στοιχεία προφίλ, αποθηκευμένες διαδρομές, διάφορα στατιστικά και άλλα στοιχεία.

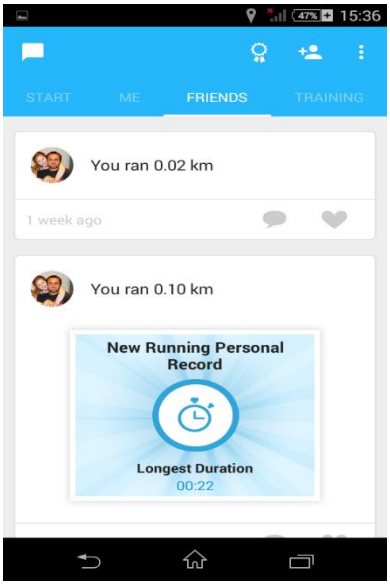


27 Αρχική οθόνη καρτέλας προφίλ. Διακρίνονται τα στοιχεία του προφίλ ,κάποια στατιστικά και παρακάτω μια λίστα με όλα τα δεδομένα που είναι διαθέσιμα.



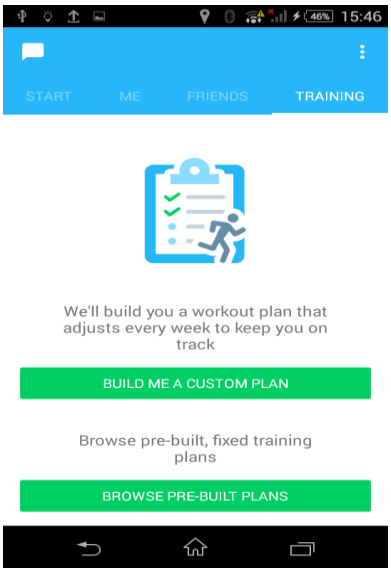
28 Μέσα από την καρτέλα του προφίλ είναι διαθέσιμη και η λίστα με όλες τις αποθηκευμένες δραστηριότητες.

- *Καρτέλα "Friends"*
  Η καρτέλα περιέχει όλα τα κοινωνικά στοιχεία της εφαρμογής. Αναλυτικότερα, διαθέτει ένα εύχρηστο χρονολόγιο, στο όποιο εμφανίζονται όλες οι δραστηριότητες των φίλων. Ο χρήστης έχει τη δυνατότητα να πατήσει «μου αρέσει» ή να γράψει κάποιο σχόλιο σε αυτές.

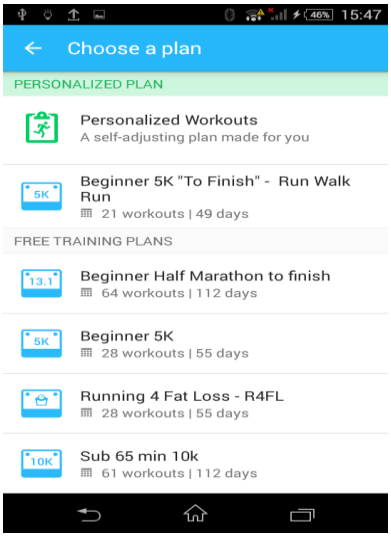


29 Καρτέλα Friends

- *Καρτέλα Training*
  Στην καρτέλα training ο χρήστης μπορεί να επιλέξει να ακολουθήσει κάποια από τα προκατασκευασμένα πλάνα προπόνησης (ελάχιστα είναι διαθέσιμα στην δωρεάν έκδοση). Στην επί πληρωμή έκδοση ο χρήστης έχει τη δυνατότητα να ζητήσει τη δημιουργία ενός προγράμματος προσαρμοσμένου κατάλληλα στα μέτρα του.



30 Στην αρχική οθόνη της καρτέλας training μπορεί να επιλεχθεί custom ή prebuild πλάνο.



31 Τα προκατασκευασμένα πλάνα (υπάρχουν δωρεάν και επί πληρωμή).

Σε αντίθεση με τις άλλες εφαρμογές, η δωρεάν έκδοση του RunKeeper **δεν** εμπεριέχει διαφημίσεις. Επιπλέον, υπάρχει και μια επί πληρωμή έκδοση που ονομάζεται RunKeeper GO και προσφέρει συνοπτικά τα εξής χαρακτηριστικά*:

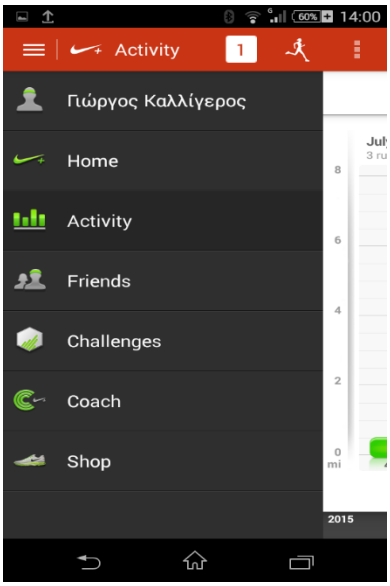
- *Live Broadcasting*
  Πρόκειται για ένα χαρακτηριστικό που επιτρέπει στο χρήστη να αναμεταδίδει ζωντανά την πορεία της δραστηριότητάς του στους φίλους του.
- *Προχωρημένα στατιστικά και αναφορές απόδοσης*
- *Προχωρημένη δημιουργία στόχων*
- *Επιπλέον πλάνα προπόνησης πέρα από τα δωρεάν που παρέχονται*
- *RunKeeper DJ*
  Ένα χαρακτηριστικό που επιτρέπει στο χρήστη να ακούει μουσική κατά τη διάρκεια του τρεξίματος, με την εφαρμογή να επιλέγει μουσική ανάλογα με το ρυθμό των βημάτων του.

*Πλήρης λίστα με τα επιπλέον χαρακτηριστικά του RunKeeper GO μπορεί να βρεθεί εδώ

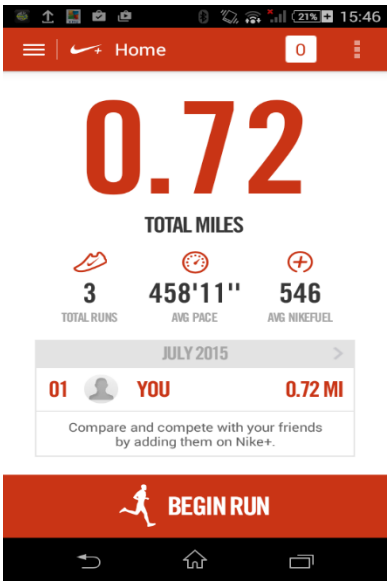
3.4 Nike running

Όνομα	Nike Running
Δημιουργός	Nike , Inc
Μέγεθος	32.22 MB

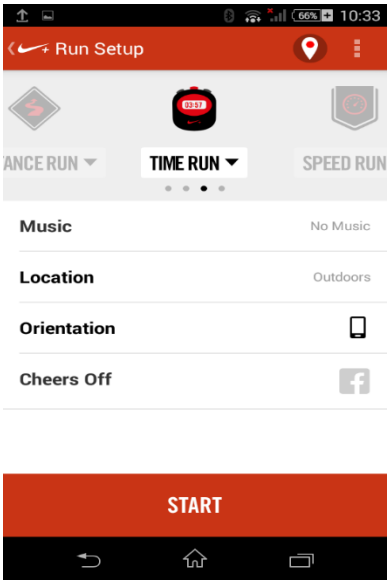
Τελευταία παρουσιάζουμε την εφαρμογή της γνωστής πολυεθνικής, η οποία είναι εντελώς δωρεάν, χωρίς μάλιστα να εμφανίζει διαφημίσεις.
Η αρχική οθόνη της εφαρμογής περιέχει κάποια βασικά στατιστικά και το κουμπί έναρξης νέας δραστηριότητας.Όταν ο χρήστης ξεκινήσει μια νέα δραστηριότητα, μεταφέρεται στην οθόνη ρύθμισης των παραμέτρων της. Εκεί μπορεί να ρυθμίσει αρκετές παραμέτρους, όπως το είδος του τρεξίματος (αντοχής,ταχύτητας κ.α), τη μουσική που θα ακούει και μια πληθώρα άλλων επιλογών.Κατά τη διάρκεια της δραστηριότητας όλα τα στοιχεία της εμφανίζονται σε πραγματικό χρόνο.



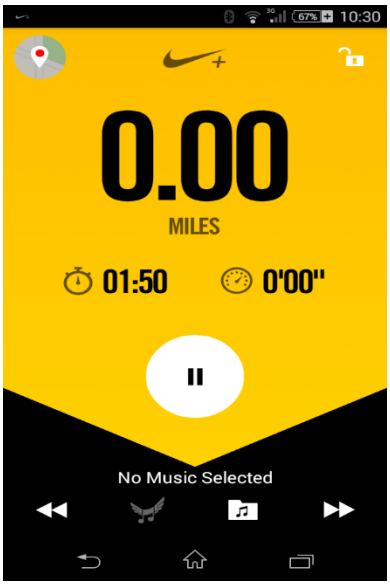
32 Το κεντρικό πλαϊνό μενού.



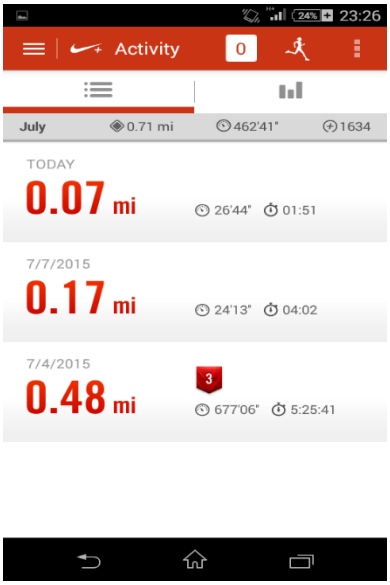
33 Αρχική οθόνη της εφαρμογής



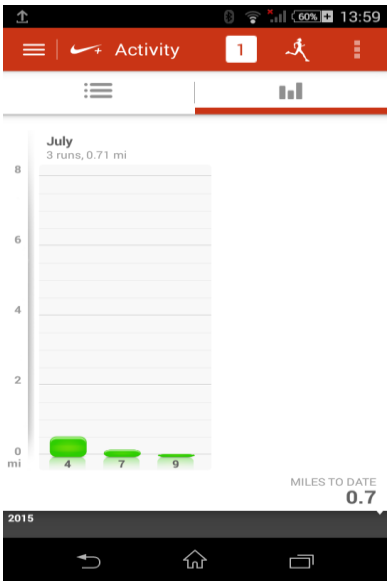
34 Οθόνη ρύθμισης παραμέτρων



35 Οθόνη απεικόνισης δεδομένων δραστηριότητας. Με πάτημα στο εικονίδιο του χάρτη πάνω αριστερά, εμφανίζεται και ο χάρτης.

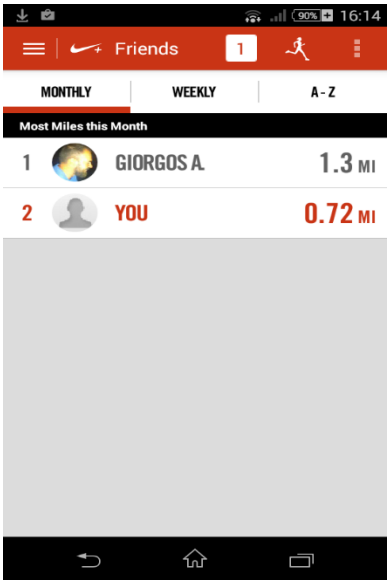


36 Λίστα με όλες τις αποθηκευμένες δραστηριότητες του χρήστη.



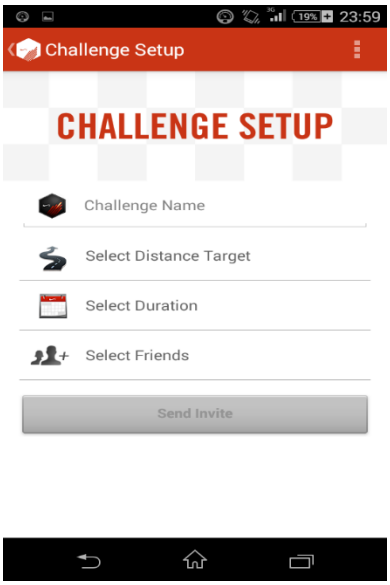
37 Μερικά συνοπτικά στατιστικά.

Τα κοινωνικά χαρακτηριστικά αν και εμφανώς πιο λιτά σε σχέση με τις άλλες εφαρμογές δε θα μπορούσαν να λείπουν. Συγκεκριμένα, ο χρήστης μπορεί να προσθέσει φίλους, να δει τα εβδομαδιαία ή τα μηνιαία στατιστικά τους σε σχέση με τα δικά του, ενώ παράλληλα δε λείπει και το χαρακτηριστικό των προκλήσεων.



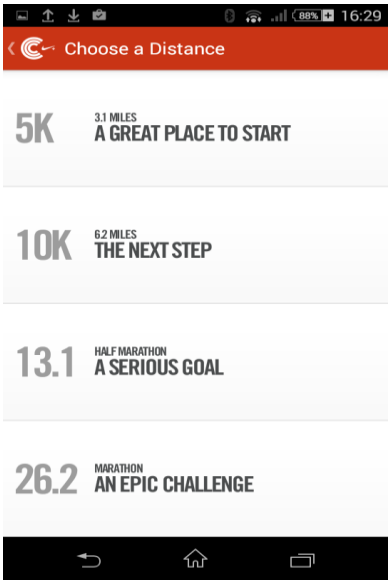
38 Τα κοινωνικά χαρακτηριστικά του Nike+ Running είναι αρκετά μινιμαλιστικά

Ο τομέας των προκλήσεων σε αυτή την εφαρμογή είναι λιγότερο πλούσιος σε χαρακτηριστικά συγκριτικά με άλλες εφαρμογές, αν και κρίνεται επαρκής για μη απαιτητικούς χρήστες. Η δημιουργία πρόκλησης είναι μία απλή διαδικασία, κατά την οποία ο χρήστης ρυθμίζει τις παραμέτρους της, προσκαλεί φίλους να λάβουν μέρος σε αυτήν και την ξεκινάει.

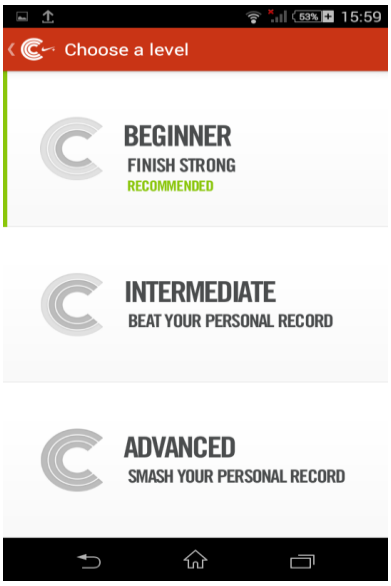


39 Οθόνη δημιουργίας πρόκλησης

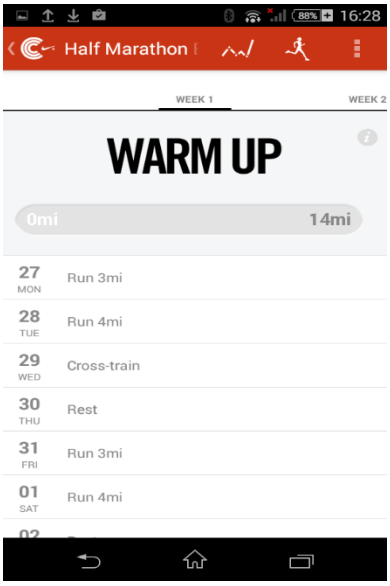
Ένα πολύ χρήσιμο κομμάτι του Nike+ Running είναι η λειτουργία προπονητή(Coach). Η λειτουργία αυτή δίνει τη δυνατότητα στο χρήστη να επιλέγει προκατασκευασμένα προγράμματα προπόνησης με βάση φυσική του κατάσταση, ενώ ταυτόχρονα η εφαρμογή τον καθοδηγεί βήμα-βήμα μέχρι την ολοκλήρωσή τους.



40 Επιλογή προγράμματος προπόνησης



41 Επιλογή επιπέδου δυσκολίας



42 Η εφαρμογή καταρτίζει αναλυτικό πρόγραμμα και οδηγίες για την επιτυχή ολοκλήρωσή του.

Αν και σημαντικά πιο απλή από τις ανταγωνιστικές, η συγκεκριμένη εφαρμογή προσφέρει πολλές χρήσιμες λειτουργίες στον χρήστη, ενώ έχει και το πλεονέκτημα του ότι είναι εντελώς δωρεάν χωρίς μάλιστα να προβάλλει διαφημίσεις.

3.5 Επιδόσεις

Για τις ανάγκες της παραπάνω παρουσίασης δοκιμάσαμε όλες τις εφαρμογές σε δύο συσκευές Android, μία μέσων επιδόσεων και μία χαμηλών επιδόσεων. Αυτό που έγινε φανερό είναι ότι όλες οι εφαρμογές αυτού του είδους είναι υπολογιστικά αρκετά απαιτητικές σε πόρους συστήματος, με συνέπεια να γίνονται αργές έως και μη ανταποκρίσιμες ειδικά σε συσκευές χαμηλών δυνατοτήτων. Το φαινόμενο αυτό οφείλεται σε ένα συνδυασμό παραγόντων, που επηρεάζουν σημαντικά την ταχύτητα και την κατανάλωση πόρων μιας εφαρμογής, οι κυριότεροι από τους οποίους είναι:

- *Σχεδιαστικές επιλογές με πολλά γραφικά –πολύπλοκες διεπαφές με πληθώρα στοιχείων σε κάθε οθόνη*.

  Ο κάθε σχεδιασμός διεπαφής από το λειτουργικό σύστημα απαιτεί το διάβασμα ενός XML αρχείου, μια αρκετά κοστοβόρα εργασία καθώς συμπεριλαμβάνει διάβασμα από την μόνιμη μνήμη του τηλεφώνου. Επίσης, η χρήση εικόνων επιβαρύνει πολύ τη μνήμη και τον επεξεργαστή
- *Σχεδιασμός online πλατφόρμας υποστήριξης εφαρμογής*.

  Ο συγκεκριμένος παράγοντας σχετίζεται και με τον ίδιο το σχεδιασμό – προγραμματισμό της πλατφόρμας , καθώς παραλήψεις (που οδηγούν σε δέσμευση των πόρων) μπορούν να οδηγήσουν σε προσωρινή αδυναμία εξυπηρέτησης των αιτημάτων των χρηστών. Επίσης συνδέεται και με την επιλογή της τεχνολογίας, πάνω στην οποία βασίζεται η πλατφόρμα. Για παράδειγμα, το REST API που δημιουργήσαμε βασίζεται σε JSON για μεταφορά πληροφορίας και Node.js +MongoDB για επεξεργασία και αποθήκευσή της αντίστοιχα. Ο συνδυασμός αυτός στην πράξη αποδεικνύεται ταχύτερος από μια παραδοσιακή προσέγγιση, που θα χρησιμοποιούσε για παράδειγμα PHP και SQL,για την συγκεκριμένη εφαρμογή .
- *Υπερπληθώρα χαρακτηριστικών που είναι χρήσιμα σε ελάχιστους χρήστες αλλά καταναλώνουν πολλούς πόρους του συστήματος*.

Κατά τη διάρκεια της ανάπτυξης της εφαρμογής που πραγματεύεται η παρούσα διπλωματική εργασία, έγινε κάθε προσπάθεια αποφυγής όλων των παραπάνω, με σκοπό η εν λόγω εφαρμογή να υπερτερεί στα συγκεκριμένα χαρακτηριστικά. Αποφύγαμε λοιπόν τη χρήση πολύπλοκων διεπαφών .Όσον αφορά την πλατφόρμα διατηρήσαμε μόνο τις απαραίτητες προσφερόμενες λειτουργίες και χρησιμοποιήσαμε Node.js, μία από τις γρηγορότερες λύσεις για την δημιουργία REST APIs. Για να ελέγξουμε πειραματικά εάν πετύχαμε τους στόχους μας πραγματοποιήσαμε μία σειρά συγκριτικών μετρήσεων μεταξύ των εφαρμογών που παρουσιάσαμε και αυτής που αναπτύξαμε εμείς.

Συγκεκριμένα, προβήκαμε σε μετρήσεις διαφόρων μεγεθών, με κάθε μέτρηση να πραγματοποιείται σε δυο συσκευές για μεγαλύτερη αξιοπιστία. Χρησιμοποιήθηκαν οι εξής συσκευές:

- *Sony Xperia E4g*

  Μία σχετικά καινούρια συσκευή, μέσων επιδόσεων με τα εξής χαρακτηριστικά:

Επεξεργαστής	Quad-core 1.5 GHz Cortex-A53
Μνήμη RAM	1 GB
Επεξεργαστής Γραφικών	Mali-T760MP2
Εσωτερική μνήμη	8GB
Συνδεσιμότητα	2G,3G,4G
Λειτουργικό	Android OS, v4.4.4 (KitKat)
GPS	Yes, with A-GPS, GLONASS

1 Χαρακτηριστικά Sony Xperia E4g

- *Samsung Galaxy Pocket Neo GT-S5310*
  Μία συσκευή μικρού μεγέθους και επιδόσεων. Λόγω του μεγέθους της είναι αρκετά βολική ώστε να χρησιμοποιηθεί για τέτοιου είδους εφαρμογές. Τα χαρακτηριστικά της είναι:

Επεξεργαστής	Single-core 850 Mhz
Μνήμη RAM	512 MB
Επεξεργαστής Γραφικών	On chip
Εσωτερική μνήμη	4GB
Συνδεσιμότητα	2G,3G
Λειτουργικό	Android OS, v4.1.2 (Jelly Bean)
GPS	Yes, with A-GPS, GLONASS

2 Χαρακτηριστικά Samsung Galaxy Pocket Neo GT-S5310

Τα μεγέθη που θα μετρήσουμε είναι τα εξής* :

- *Χρόνος εκκίνησης εφαρμογής –login*
  Η ταυτοποίηση του χρήστη σε όλες τις εφαρμογές γίνεται κατά την εκκίνηση της εφαρμογής. Επομένως, ο συνολικός χρόνος εκκίνησης εξαρτάται από την αποδοτικότητα της ίδιας της εφαρμογής, τις δυνατότητες του κινητού τηλεφώνου και την απόδοση της διαδικτυακής πλατφόρμας. Το συγκεκριμένο μέγεθος μας δίνει συνεπώς μία αρκετά ενδεικτική εικόνα της ταχύτητας όλου του συστήματος συνολικά.
- *Κατανάλωση μνήμης RAM σε λειτουργία αναμονής(χωρίς να καταγράφεται δραστηριότητα).*
  Η κατανάλωση μνήμης RAM αντικατοπτρίζει κυρίως την πολυπλοκότητα και το πλήθος των στοιχείων της διεπαφής..
- *Κατανάλωση μνήμης RAM σε λειτουργία καταγραφής δραστηριότητας*
  Η λειτουργία καταγραφής δραστηριότητας είναι η πιο βασική σε αυτού του είδους τις εφαρμογές και η απόδοσή της αποτελεί σημαντικό στοιχείο γι’ αυτές. Η κατανάλωση μνήμης στην συγκεκριμένη περίπτωση είναι λογικό να είναι μεγαλύτερη, καθώς συνήθως για την καταγραφή οι εφαρμογές εκκινούν ξεχωριστές υπηρεσίες.

*Οι μετρήσεις έγιναν με τις παρακάτω συμβάσεις:

- Μεταξύ διαδοχικών μετρήσεων γινόταν επανεκκίνηση της συσκευής, για μεγαλύτερη αξιοπιστία.
- Μετρήσαμε την μνήμη RAM με την εφαρμογή Memory Usage.
- Οι μετρήσεις που αφορούσαν την λειτουργία καταγραφής γίνονταν μετά την εκτέλεση μιας προκαθορισμένης διαδρομής και ενόσω η καταγραφή συνεχιζόταν.
- Η σύνδεση στο διαδίκτυο γινόταν μέσω πρωτοκόλλου 3G και έτεινε στο 100% του σήματος.

Τα αποτελέσματα των μετρήσεων για την συσκευή *Sony Xperia E4g* παρουσιάζονται παρακάτω.

Measure	App	Endomondo	Run Keeper	MapMyrun	Nike Running	Run Tracker	Best Result
Startup and login Time (s)		3.1	3.73	3.04	3.56	**1.73**	**1.73**
Space in RAM idle(peak)		113	101	143	87	**71**	**71**
Space in RAM (tracking)		119	119	125	130	**92**	**92**

3 Αποτελέσματα μετρήσεων στο *Sony Xperia E4g*

Παρατηρούνται εμφανώς καλύτερα μεγέθη στην εφαρμογή μας. Από τα πιο σημαντικά μεγέθη είναι ο χρόνος έναρξης, ο καλύτερος μάλιστα είναι ο μισός από τον αμέσως επόμενο. Από αυτό μπορούμε να συμπεράνουμε πως η χρήση απλής διεπαφής με μόνο τα απολύτως απαραίτητα συμβάλλει πράγματι στην επιτάχυνση της εφαρμογής. Η μείωση του χρόνου έναρξης θα μπορούσε να οφείλεται επίσης και στην ταχύτητα της πλατφόρμας και των τεχνολογιών που συνδυάστηκαν σε αυτήν (Node.js + MongoDB + μεταφορά δεδομένων σε μορφή JSON). Ο συγκεκριμένος συνδυασμός είναι ταχύτατος και ελαχιστοποιεί το μέγεθος των δεδομένων που μεταφέρονται μέσω δικτύου (γεγονός υψίστης σημασίας, καθώς στις συνδέσεις δεδομένων των κινητών συνήθως υπάρχει ογκοχρέωση).

Οι διαφορές στην κατανάλωση της μνήμης* είναι επίσης αρκετά σημαντικές διότι όσο λιγότερη μνήμη καταναλώνει μια εφαρμογή Android, τόσο περισσότερη μνήμη μένει διαθέσιμη για άλλες εφαρμογές που τρέχουν ταυτόχρονα.

*Η διαχείριση μνήμης που έχει το Android είναι πολύ ισχυρή καθώς χρησιμοποιεί εξελιγμένη τεχνολογία garbage collection για να ελευθερώνει αχρείαστη μνήμη που έχει δεσμευτεί από εφαρμογές, κι επομένως τα προβλήματα έλλειψης RAM είναι σπάνια.

Τα αποτελέσματα των μετρήσεων για την συσκευή *Samsung Galaxy Pocket Neo GT-S5310* παρουσιάζονται στον παρακάτω πίνακα.

Measure	App	Edomondo	Run Keeper	MapMyrun	Nike Running	Run Tracker	Best Result
Startup and login Time (s)		20,8	19,1	24,1	11	**8**	**8**
Space in RAM (idle) (MB)		54	64	66	56	**42**	**42**
Space in RAM (tracking) (MB)		69	66	73	58	**45**	**45**

4 Αποτελέσματα μετρήσεων στο Samsung Galaxy Pocket Neo GT-S5310

Το πρώτο σχόλιο που προκύπτει από τις παραπάνω μετρήσεις είναι πως ο χρόνος έναρξης των τριών πρώτων εφαρμογών τις κάνει πρακτικά μη χρηστικές σε τέτοιου τύπου συσκευές χαμηλών δυνατοτήτων. Ο χρήστης βέβαια είναι πολύ πιθανό να θέλει να χρησιμοποιήσει μία τέτοια συσκευή για την καταγραφή του τρεξίματός του, λόγω της χαμηλής τιμής και του μικρού μεγέθους της.

Αναφορικά με την κατανάλωση μνήμης RAM, όπως αναμενόταν η εφαρμογή μας είναι πιο αποδοτική από τις ανταγωνιστικές. Η γενικά μικρότερη κατανάλωση RAM σε αυτήν τη συσκευή οφείλεται στο ότι το Android λόγω της μικρότερης RAM της συσκευής, διαθέτει σε κάθε εφαρμογή λιγότερη μνήμη συνολικά, ενώ κάνει και πιο επιθετική χρήση του garbage collector.

Τέλος, να τονίσουμε πως η μόνη εφαρμογή που έχει συγκρίσιμα αποτελέσματα με την δική μας είναι η *Nike+Running*.

4.Αθλητικα δεδομένα

Ως αθλητικά δεδομένα ορίζουμε τα στοιχεία αυτά που αφορούν μια αθλητική δραστηριότητα και που είναι χρήσιμο ο αθλητής να γνωρίζει. Υπάρχουν πάρα πολλά τέτοια δεδομένα που θα μπορούσαν να ενδιαφέρουν τον χρήστη. Στην περίπτωση που η αθλητική δραστηριότητα περιλαμβάνει διαδρομή σε εξωτερικό χώρο(πχ περπάτημα, τρέξιμο, ποδηλασία) τα σημαντικότερα από αυτά είναι:

- *Ίχνος διαδρομής*

Η διαδρομή που ακολουθήθηκε (σχεδιασμένη πάνω στο χάρτη) είναι πολύ ουσιαστικό δεδομένο για μία αθλητική δραστηριότητα, και ο χρήστης επιθυμεί να το γνωρίζει για παρά πολλούς λόγους.

- *Απόσταση διαδρομής*

Η απόσταση είναι ένα από τα κυριότερα στοιχεία που ενδιαφέρουν το χρήστη, για προφανείς λογούς (πχ. προκειμένου να θέτει στόχους, όπως: «σήμερα θα τρέξω 1000 μέτρα»).

- *Ταχύτητα*

Η ταχύτητα, μέση ή στιγμιαία αποτελεί δεδομένο βαρύνουσας σημασίας, ειδικά σε περιπτώσεις που ο χρήστης ασχολείται με πρωταθλητισμό (πχ. μαραθώνιος ,αντοχή κτλ).

- *Χρόνος διαδρομής*

Και ο χρόνος είναι πολύ σημαντικό στοιχείο, καθώς ο χρήστης μπορεί να επιθυμεί να θέσει χρονικούς περιορισμούς ή χρονικούς στόχους στην άσκησή του.

- *Θερμίδες που καταναλωθήκαν στη διάρκεια της διαδρομής*

Οι θερμίδες αποτελούν ένα δεδομένο διαιτολογικής φύσεως και αφορούν χρήστες που έχουν στόχο κυρίως την απώλεια ή την διατήρηση βάρους μέσα από την άσκηση.

- *Καρδιακοί παλμοί κατά τη διάρκεια της διαδρομής (μέσος ορός ή ανά διαστήματα)*

Οι καρδιακοί παλμοί είναι πολύ σημαντικό δεδομένο, κυρίως για λόγους υγείας. Ο Παγκόσμιος Οργανισμός Υγείας θέτει κάποια άνω όρια για κάθε ηλικία στους καρδιακούς παλμούς, τα οποία δεν πρέπει να ξεπεραστούν κατά την διάρκεια της άσκησης (ανάλογα και με την κατάσταση της υγείας και την ηλικία του αθλητή). Συγκεκριμένα όρια στους παλμούς οριοθετούν και συγκεκριμένου τύπου άσκηση(για παράδειγμα από 110-130 παλμούς έχουμε την κατάλληλη περιοχή για κάψιμο θερμίδων).

4.1 Καταγραφή δεδομένων

4.1.1 Επιλογή δεδομένων προς καταγραφή

Αναμφισβήτητα, όλα τα προαναφερθέντα δεδομένα είναι πολύ χρήσιμα. Η συλλογή όμως μερικών από αυτά δεν είναι καθόλου εύκολη υπόθεση, δεδομένων των αισθητήρων που διαθέτει ένα κοινό έξυπνο κινητό τηλέφωνο. Ο υπολογισμός κάποιων άλλων απαιτεί εξειδικευμένες γνώσεις, εκτός των πλαισίων αυτής της διπλωματικής εργασίας, ενώ και η συλλογή ορισμένων δεδομένων καθίσταται εντελώς αδύνατη.
Καταλήγουμε επομένως στο συμπέρασμα πως θα χρειαστεί να κάνουμε αρκετούς συμβιβασμούς στα δεδομένα που επιλέξαμε να καταγράφονται, αλλά και στην ακρίβεια αυτών.

Συγκεκριμένα επιλέξαμε να μην καταγράψουμε τα ακόλουθα στοιχεία:

- *Ταχύτητα*
  Η καταγραφή της ταχύτητας δεν έγινε για λόγους πρακτικούς. Πιο αναλυτικά, το Google location service δεν μας παρέχει την ταχύτητα, οπότε έπρεπε να την υπολογίζουμε «με το χέρι» κάθε φορά δαπανώντας με τον τρόπο αυτό υπολογιστικούς πόρους.

- *Θερμίδες που καταναλωθήκαν στη διάρκεια της διαδρομής*
  Προκειμένου να υπολογιστούν οι θερμίδες με ακρίβεια απαιτείται γνώση της ταχύτητας και του είδους της αθλητικής δραστηριότητας(ποδήλατο, τρέξιμο κτλ.). Επομένως, ο υπολογισμός των θερμίδων προσδίδει μεγάλη πολυπλοκότητα, γεγονός που μας οδηγεί στην απόφαση να μην τον συμπεριλάβουμε προς το παρόν στα πλαίσια της εκπόνησης της παρούσας εργασίας.

- *Καρδιακοί παλμοί κατά τη διάρκεια της διαδρομής (μέσος ορός ή ανά διαστήματα)*
  Για να καταγράψουμε τους καρδιακούς παλμούς χρειαζόμαστε ειδικό αισθητήρα, ο όποιος υπάρχει μόνο σε μερικά ρολόγια με Android, οπότε επιλέξαμε να μην τους συμπεριλάβουμε.

Τελικά τα δεδομένα που επιλέχθηκαν προς καταγραφή είναι τα ακόλουθα:

- *Ίχνος της διαδρομής στο χάρτη*
- *Χρόνος διαδρομής*
- *Απόσταση διαδρομής*

4.1.2 Ακρίβεια μετρήσεων

Για να υπολογίσουμε την απόσταση και το ίχνος στο χάρτη είναι απαραίτητο να γνωρίζουμε την τοποθεσία του χρήστη κάθε δεδομένη στιγμή. Υπάρχουν τρείς μέθοδοι –αισθητήρες για να υπολογίσουμε την τοποθεσία, με κάθε μία να έχει τα δικά της πλεονεκτήματα και μειονεκτήματα. Οι μέθοδοι είναι οι εξής:

- *GPS (Global Positioning System)-GLONASS*
  Η μέθοδος αυτή χρησιμοποίει το κλασικό σύστημα GPS,που δημιουργήθηκε από το Υπουργείο Άμυνας των ΗΠΑ και βασίζεται σε 31 δορυφόρους ,εκ των οποίων οι 24 τουλάχιστον είναι πάντα σε λειτουργία. Για να εντοπίσει τη θέση μας ο δέκτης που βρίσκεται στο τηλέφωνο χρειάζεται σήμα από τουλάχιστον τρεις δορυφόρους, ενώ προκειμένου να εντοπίσει και το υψόμετρό μας έχει ανάγκη σήμα από έναν ακόμα. Πολλά κινητά τηλέφωνα υποστηρίζουν ταυτόχρονα και το αντίστοιχο ρωσικό σύστημα (GLONASS).
  Η μέθοδος δορυφορικού εντοπισμού διαθέτει το μεγάλο πλεονέκτημα της ακρίβειας (έως 3-5 μέτρα), αλλά χρειάζεται αρκετό χρόνο για να βρει κανείς σήμα από τους απαιτούμενους δορυφόρους. Και η λήψη όμως των δεδομένων από αυτούς απαιτεί αρκετό χρονικό διάστημα (45-60 sec συνολικά). Γι’ αυτό το λόγο έχει αναπτυχθεί το σύστημα A-GPS, που χρησιμοποιεί τη σύνδεση του κινητού τηλέφωνου στο διαδίκτυο,

με αποτέλεσμα να επιταχύνεται σημαντικά η διαδικασία εντοπισμού. Ένα ακόμα μειονέκτημα του δορυφορικού εντοπισμού είναι η εν γένει δυσκολία στον εντοπισμό σήματος από δορυφόρο κυρίως σε εσωτερικούς χώρους ή κοντά σε ψηλά κτήρια. Σε τέτοιες περιπτώσεις καταφεύγουμε στις ακόλουθες μεθόδους εντοπισμού.

- *Wi-Fi positioning*
  Η παραπάνω μέθοδος βασίζεται στα σήματα από Wi-Fi access points με γνωστή θέση, τα οποία λαμβάνει το τηλέφωνο. Μετρώντας την ισχύ κάθε σήματος μπορούμε να βρούμε τη θέση της συσκευής. Η ακρίβεια αυτής της μεθόδου είναι πολύ περιορισμένη (50-100 m).

- *Cellular network positioning*
  Η συγκεκριμένη μέθοδος είναι παρόμοια με το Wi-Fi positioning, με τη διαφορά πως χρησιμοποιεί τις κυψέλες της κινητής τηλεφωνίας για τον εντοπισμό της θέσης μας.Αποτελεί την μέθοδο με την μικρότερη ακρίβεια (της τάξεως των 500 m).

Για την δεδομένη εφαρμογή χρειαζόμαστε την μεγαλύτερη δυνατή ακρίβεια στην τοποθεσία του χρήστη. Στην ιδανική περίπτωση θα θέλαμε να έχουμε δορυφορικό εντοπισμό (GPS).Κάτι τέτοιο όμως εκτός από το ότι δεν είναι πάντα εφικτό (λόγω περιορισμών στη λήψη σήματος), μπορεί να μην είναι και επιθυμητό από τον χρήστη (εξαιτίας της υψηλής κατανάλωσης ενέργειας του δέκτη GPS).Αυτά τα ζητήματα μας οδηγούν στην σκέψη πως πρέπει να στραφούμε σε μια εξυπνότερη λύση και να κάνουμε χρήση του καλύτερου δυνατού τρόπου εντοπισμού, δεδομένων των περιορισμών που μας δίνονται.
Προς επίτευξη αυτού του σκοπού διαθέτουμε ένα σημαντικό εργαλείο, το Google Location API.Το εν λόγω API αποτελεί ένα ενοποιημένο interface για όλες τις παραπάνω μεθόδους, μπορεί να μας δώσει την θέση του χρήστη καθώς και την μεγαλύτερη δυνατή ακρίβεια βάσει των παραμέτρων που του θέτουμε και των περιορισμών υπάρχουν (πχ. αν δεν υπάρχει σήμα GPS, μας δίνει τη θέση βασιζόμενο μόνο στις άλλες δύο μεθόδους). Σε γενικές γραμμές είναι ακριβώς αυτό που χρειαζόμαστε με βάση τις δεδομένες απαιτήσεις μας.

5. Αρχιτεκτονική της εφαρμογής

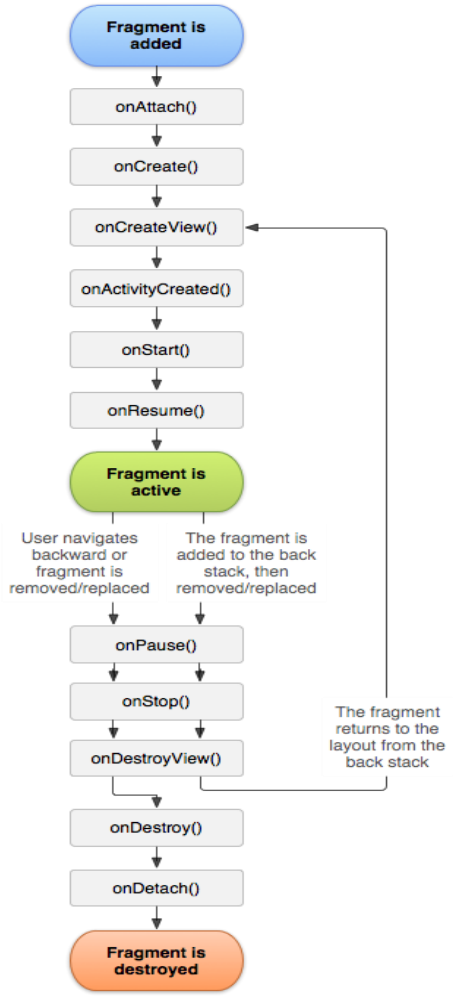
Η εφαρμογή μας αποτελείται κατά βάση από δύο κομμάτια αρκετά διαφορετικά μεταξύ τους, το mobile κομμάτι και την web πλατφόρμα. Για την λειτουργία του συστήματος είναι απαραίτητο να υπάρχει αμφίδρομη επικοινωνία μεταξύ αυτών των δύο δομικών μονάδων. Η επικοινωνία αυτή επιτυγχάνεται μέσω του HTTP πρωτόκολλου. Προκειμένου να εξασφαλίσουμε την ασφάλεια του συστήματος, χρησιμοποιούμε ένα σύστημα εξουσιοδότησης μέσω του πρωτοκόλλου HTTP .Όλα τα παραπάνω θα αναλυθούν ενδελεχώς σε αυτό το κεφάλαιο.

5.1 Mobile εφαρμογή

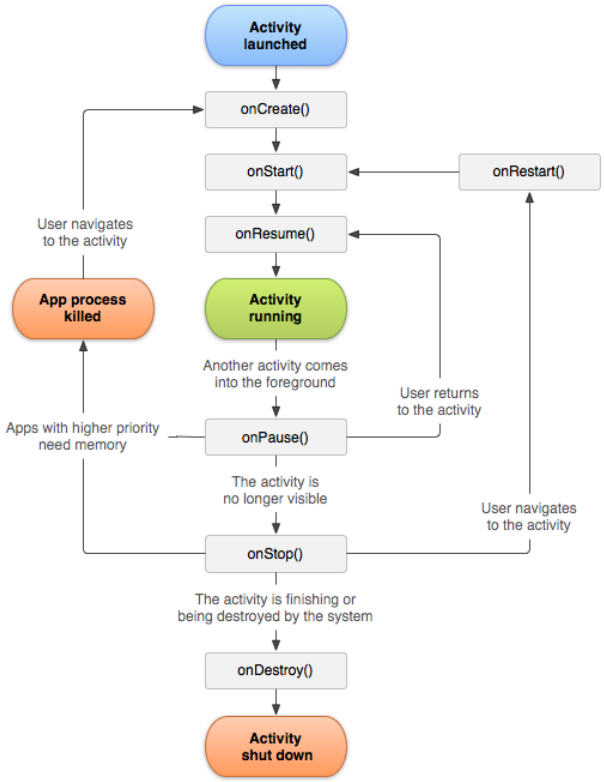
5.1.1 Εισαγωγικά

Η εφαρμογή για κινητό τηλέφωνο είναι μια native εφαρμογή για Android (γραμμένη σε γλώσσα προγραμματισμού Java).Το βασικό δομικό στοιχείο κάθε τέτοιας εφαρμογής είναι η δραστηριότητα *(Activity)*.
Η δραστηριότητα είναι το στοιχείο αυτό που προσφέρει διεπαφή του χρήστη με την εφαρμογή. Οι περισσότερες εφαρμογές αποτελούνται από την αρχική δραστηριότητα, μέσα από την όποια μπορούμε να ξεκινήσουμε και άλλες, για να εκτελέσουμε ξεχωριστές εργασίες. Κάθε δραστηριότητα έχει ένα κύκλο ζωής με διακριτά στάδια και συγκεκριμένες επιτρεπτές μεταβάσεις, όπως γίνεται φανερό στο παρακάτω σχήμα.
Αξίζει να αναφέρουμε πως όταν η δραστηριότητα αλλάζει κατάσταση, τότε το σύστημα καλεί και την ανάλογη μέθοδο, γεγονός που μας επιτρέπει να χειριστούμε αποτελεσματικά τις αλλαγές που επέρχονται στην κατάσταση της δραστηριότητας. Οι μεταβάσεις είναι απρόβλεπτες κατά το χρόνο εκτέλεσης και οφείλονται στο χρήστη ή στο λειτουργικό σύστημα (πχ. αν δούμε ότι η δραστηριότητα είναι προς καταστροφή από το λειτουργικό λόγω περιορισμένης μνήμης ή λόγω κλεισίματος από το χρήστη, έχουμε τη δυνατότητα να αποθηκεύσουμε τα απαραίτητα δεδομένα κάνοντας τις ανάλογες ενέργειες στην μέθοδο on Destroy).
Σε κάθε δραστηριότητα μπορούμε να προσθέσουμε και να αφαιρέσουμε τεμάχια (*fragments)*.Τα *fragments* αποτελούν κομμάτια κώδικα, που μπορούν να ενσωματώνουν διεπαφή χρήστη και λογική εφαρμογής. Τα *fragments* μπορούν να προσθαφαιρεθούν σε μια δραστηριότητα δυναμικά κατά τη διάρκεια εκτέλεσης της εφαρμογής. Υπάρχει επίσης η δυνατότητα να συνδυαστούν πολλά *fragments* σε μια δραστηριότητα, για την δημιουργία ενός πολύπλοκου interface. Η χρήση των fragments συνεπάγεται πολλά πλεονεκτήματα, εκ των οποίων τα κυριότερα είναι η επαναχρησιμοποίηση κώδικα, η ευελιξία, η ταχύτητα και η δυνατότητα δόμησης της εφαρμογής

μας με έναν τμηματικό τρόπο. Όπως και οι δραστηριότητες, έτσι και κάθε fragment έχει ένα κύκλο ζωής που είναι έντονα εξαρτώμενος από τον κύκλο ζωής της δραστηριότητας, μέσα στην οποία εκείνο «ζει». Σε παρακάτω σχήμα βλέπουμε όλα τα πιθανά στάδια του κύκλου ζωής ενός fragment καθώς και όλες τις πιθανές μεταβάσεις μεταξύ τους.



1 Κύκλος ζωής Fragment



2 Κύκλος ζωής δραστηριότητας

Σε περίπτωση που απαιτούμε ένα κομμάτι κώδικα να τρέχει συνεχώς στο παρασκήνιο, ανεξάρτητα από την δραστηριότητα και το fragment που είναι στο προσκήνιο εκείνη τη στιγμή (για παράδειγμα να καταγράφουμε τη θέση του χρήστη ακόμα και αν το τηλέφωνο έχει κλειδωθεί και όλες οι δραστηριότητες είναι σταματημένες), το SDK του Android μας παρέχει ένα πολύτιμο εργαλείο, τις υπηρεσίες (services).Οι υπηρεσίες αποτελούν ένα στοιχείο εφαρμογής που δεν παρέχει διεπαφή χρήστη και είναι σχεδιασμένο να τρέχει μακροπρόθεσμες εργασίες στο παρασκήνιο. Υπάρχουν πολλοί τρόποι επικοινωνίας μιας υπηρεσίας με δραστηριότητες και στην εφαρμογή μας χρησιμοποιούμε αρκετούς από αυτούς.

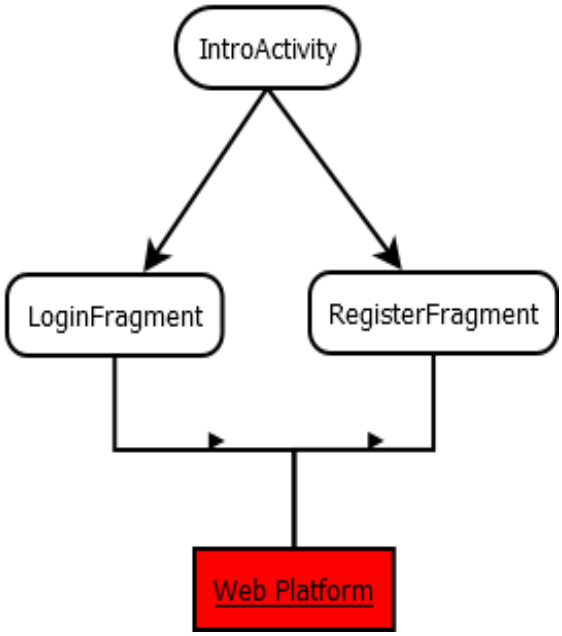
5.1.2 Η εφαρμογή

Η εφαρμογή μας κάνει χρήση όλων των δομικών μονάδων που περιγράφηκαν παραπάνω. Αποτελείται από δύο κύριες δραστηριότητες: την εισαγωγική, που χειρίζεται την είσοδο χρήστη και την εγγραφή νέου χρήστη και την κύρια, που διαχειρίζεται όλες τις βασικές λειτουργίες της εφαρμογής. Οι δύο δραστηριότητες και όλες οι υπόλοιπες μονάδες που τις πλαισιώνουν θα

παρουσιαστούν αναλυτικά παρακάτω.

ΕΙΣΑΓΩΓΙΚΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑ

Η δραστηριότητα αυτή διαχειρίζεται την εγγραφή νέου χρήστη καθώς και την είσοδο υπάρχοντος χρήστη στην εφαρμογή. Φιλοξενεί επίσης δύο fragments, τα οποία χειρίζονται τις λειτουργίες εγγραφής και σύνδεσης χρήστη καθώς και τις ξεχωριστές διεπαφές που απαιτούνται γι' αυτές. Αξίζει να σημειωθεί πως το κάθε fragment χειρίζεται μόνο του την διαδικτυακή πρόσβαση στην πλατφόρμα για επιβεβαίωση και εισαγωγή δεδομένων .



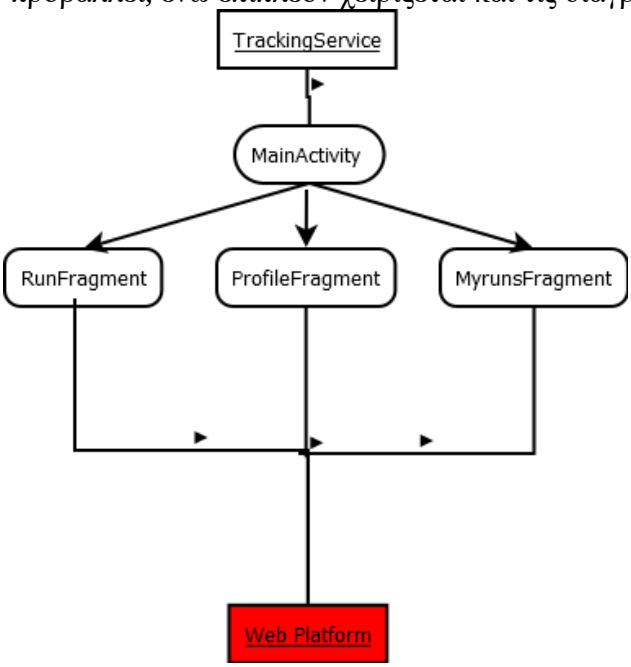
3 Δομή IntroActivity

Κεντρική εφαρμογή

Η κεντρική εφαρμογή αποτελείται από μία δραστηριότητα, η οποία εκκινεί και ελέγχει μία υπηρεσία που έχει ως ρόλο την ιχνηλάτηση του χρήστη. Η κεντρική δραστηριότητα δημιουργεί την διεπαφή με τις τρεις καρτέλες ("RUN", "MYRUNS", "PROFILE"), μέσα στις οποίες «ζουν» τα αντίστοιχα fragments.Τα fragments που ανήκουν στην κεντρική εφαρμογή είναι:

- *RunFragment*
  Αυτό το Fragment χειρίζεται την υπηρεσία ιχνηλάτησης (σταματάει ή ξεκινάει την ιχνηλάτηση κτλ.) καθώς και τη διεπαφή της καρτέλας προβολής δεδομένων δραστηριότητας. Ακόμα αναλαμβάνει την αποθήκευση νέων διαδρομών στην πλατφόρμα .
- *Profilefragment*
  Το ProfileFragment έχει ως ρόλο την προβολή και επεξεργασία των στοιχείων του προφίλ του χρήστη. Για την πραγματοποίηση των δύο αυτών δραστηριοτήτων είναι απαραίτητη η διαδικτυακή επικοινωνία με την πλατφόρμα.

- *MyRunsFragment*
  Το παραπάνω fragment ανάκτα τις διαδρομές του χρήστη από την πλατφόρμα και τις προβάλλει, ενώ επιπλέον χειρίζεται και τις διαγραφές των διαδρομών.



4 Δομή κεντρικής εφαρμογής

5.2 Web πλατφόρμα

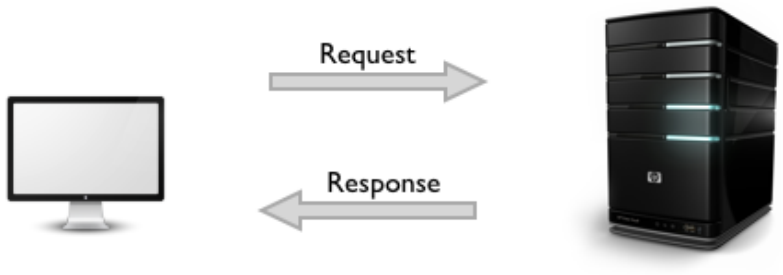
5.2.1 Εισαγωγικά

Η διαδικτυακή πλατφόρμα της εφαρμογής έχει ως ρόλο την αποθήκευση, τη διαχείριση και την ανάκτηση του περιεχομένου που δημιουργούν οι χρήστες (δηλαδή των προφίλ τους και των διαδρομών τους). Για να δομηθεί η πλατφόρμα χρησιμοποιούμε αρκετά πρωτόκολλα, τεχνικές και μεθόδους που περιγράφονται επιγραμματικά παρακάτω.

HTTP πρωτόκολλο

Το HTTP σημαίνει Hyper Text Transfer Protocol, δηλαδή πρωτόκολλο μεταφοράς υπερκειμένου κι αποτελεί ένα πρωτόκολλο επικοινωνίας μεταξύ υπολογιστικών συστημάτων. Ένα πολύ σημαντικό χαρακτηριστικό του HTTP είναι ότι δεν έχει καταστάσεις(stateless). Το πρωτόκολλο δηλαδή δεν διατηρεί κάποια κατάσταση μεταξύ των μηνυμάτων που αποστέλλονται/λαμβάνονται. Παράλληλα, κάθε νέο μήνυμα αποτελεί ανεξάρτητη οντότητα από τα προηγούμενα και δεν επηρεάζει τα επόμενα, όσον αφορά τον εξυπηρετητή.

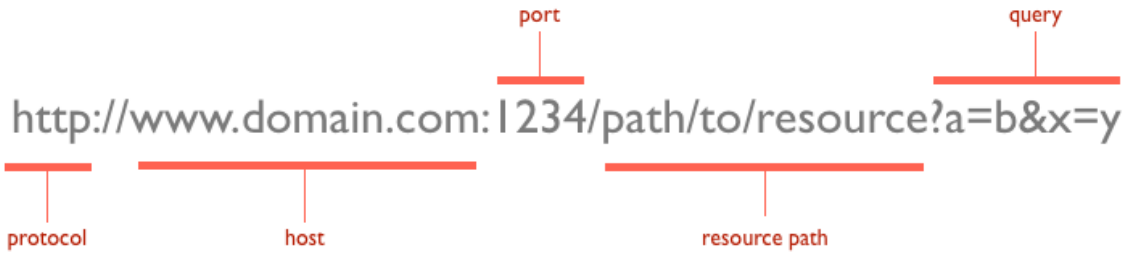
Το HTTP ακολουθεί το μοντέλο αίτηση/απόκριση (request/response), που σημαίνει ότι σε κάθε επικοινωνία υπάρχει ένας πελάτης(client) , ο οποίος στέλνει την HTTP αίτηση και ένας εξυπηρετητής, ο οποίος απαντάει με μια HTTP απόκριση. Σχηματική αναπαράσταση ενός HTTP αιτήματος/απόκρισης μπορούμε να δούμε στο παρακάτω σχήμα:



43 Βασική μορφή μιας επικοινωνίας με HTTP

Για να στείλουμε ένα HTTP αίτημα στον εξυπηρετητή χρειάζεται η διεύθυνσή του, η θύρα (port) επικοινωνίας καθώς και η συγκεκριμενοποίηση του πόρου που ζητάμε από τον εξυπηρετητή. Όλες αυτές οι πληροφορίες μας δίνονται από το URL(Uniform Resource Locator).Κάθε URL αποτελείται από τα ακόλουθα χαρακτηριστικά, με σειρά εμφάνισης:

- Το **Scheme**
  Συνήθως το scheme αποτελεί το πρωτόκολλο επικοινωνίας. Στην περίπτωση που εξετάζουμε δηλαδή είναι HTTP ή HTTPS (ασφαλής έκδοση του HTTP).
- **Οι χαρακτήρες ://**
- **Η διεύθυνση του εξυπηρετητή**, είτε με τη μορφή domain name είτε με τη μορφή IP διεύθυνσης.
- **Προαιρετικά ο χαρακτήρας «:»** Ακολουθούμενος από τον αριθμό θύρας στο HTTP, ενώ αν αυτό το κομμάτι λείπει χρησιμοποιείται η θύρα 80.
- **Όλο το μονοπάτι για τον πόρο που ζητάμε.**



44 Γενική μορφή URL

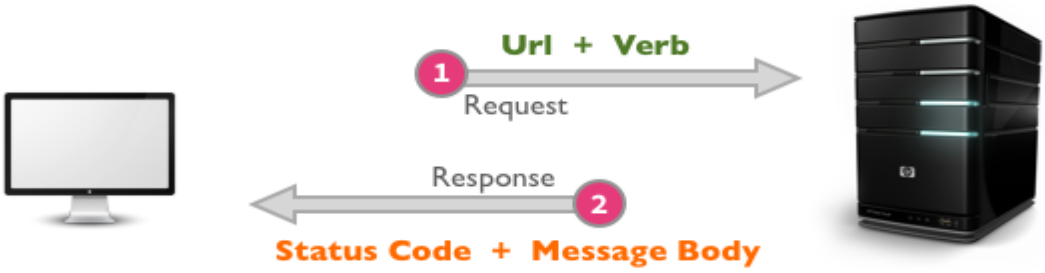
Το URL μας δείχνει τον πόρο στον οποίο θέλουμε να αποκτήσουμε πρόσβαση, όμως δεν μας δίνει καμία πληροφορία για το τι θέλουμε να κάνουμε με αυτόν (πχ. απλή προβολή, μετατροπή, διαγραφή κτλ.). Γι' αυτό το σκοπό χρησιμοποιούμε τις μεθόδους HTTP. Υπάρχουν αρκετές τέτοιες μέθοδοι και στα πλαίσια της παρούσας εργασίας παρουσιάζονται οι βασικότερες, που είναι και αυτές που χρησιμοποιούνται στον κώδικα της εφαρμογής.

- **GET**
  Η μέθοδος αυτή χρησιμοποιείται από τον πελάτη για να ζητήσει την ανάκτηση ενός πόρου από το server .

- **POST**
  Με τη μέθοδο POST κάνουμε δημιουργία ενός νέου πόρου στον εξυπηρετητή. Συνήθως η αίτηση αυτή περιέχει τα δεδομένα που θα χρησιμοποιηθούν στην δημιουργία του νέου πόρου ως παράμετροι.
- **PUT**
  Με τη συγκεκριμένη μέθοδο κάνουμε τροποποίηση ενός υπάρχοντος πόρου. Η αίτηση αυτή έχει συνήθως τα δεδομένα που θα χρησιμοποιηθούν για να αλλάξουμε τον πόρο.
- **DELETE**
  Με αυτή την μέθοδο κάνουμε διαγράφη ενός πόρου.

Ο πελάτης χρησιμοποίει τα URLs και τις μεθόδους για να στείλει ένα αίτημα στον εξυπηρετητή. Ο εξυπηρετητής στη συνέχεια απαντάει με ένα κωδικό απόκρισης και με το κύριο σώμα της απόκρισης. Ύστερα, ο πελάτης χρησιμοποιεί τον κωδικό για να ερμηνεύσει το περιεχόμενο της απόκρισης. Υπάρχουν πολλοί κωδικοί απόκρισης, γι' αυτό θα αναφέρουμε μόνο τους πιο συνηθισμένους και αυτούς που χρησιμοποιούνται στην παρούσα διπλωματική.

- **200 OK**
  Το αίτημα ήταν επιτυχές, αυτό σημαίνει π.χ. για ένα GET αίτημα πως το μήνυμα περιέχει τον πόρο που ζητήσαμε ή για ένα DELETE αίτημα πως η διαγραφή έγινε κ.ο.κ.
- **301 Moved Permanently**
  Ο πόρος που ζητείται έχει μετακινηθεί σε νέο URL.
- **304 Not Modified**
  Ο πόρος δεν έχει αλλάξει από την τελευταία του αποστολή στον πελάτη, συνεπώς ο πελάτης χρησιμοποίει το αντίγραφο του πόρου που διαθέτει στην κρυφή μνήμη του.
- **404 Not Found**
  Ο πόρος δεν υπάρχει στον εξυπηρετητή (λάθος URL συνήθως).
- **503 Service Unavailable**
  Ο κωδικός αυτός αποστέλλεται όταν υπάρχει κάποιο σφάλμα στον εξυπηρετητή (πχ. υπερφόρτωση ή εσωτερικό σφάλμα) και το αίτημα δεν μπορεί να ολοκληρωθεί.

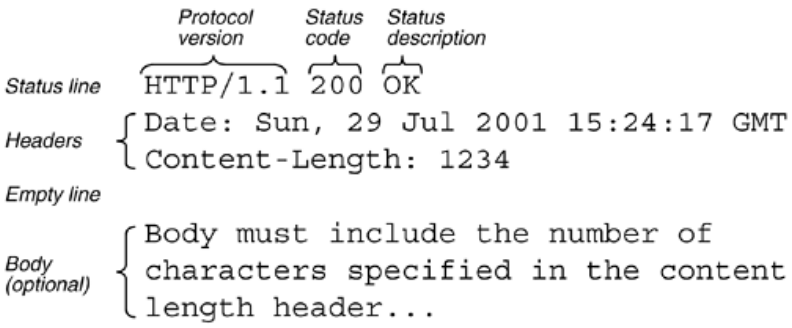


45 Σχηματική απεικόνιση όλων όσων περιέχονται σε μια HTTP επικοινωνία.

Τέλος, αξίζει να σημειωθεί πως ένα HTTP αίτημα/απάντηση εκτός από όλα τα παραπάνω μπορεί να περιέχει και διάφορες επικεφαλίδες με πληροφορίες, που είναι χρήσιμες στον πελάτη/εξυπηρετητή.

Request-line	**Get /products/dvd.htm HTTP/1.1**
General Header	**Host:www.videoequip.com** **Cache-Control:no-cache** **Connection:Keep-Alive**
Request Header	**Content-Length:133** **Accept-Language:en-us** **. . .**
Entity Header	**Content-Length:133** **Content-Language:en** **. . .**
Body	

46 Τυπική δομή GET αιτήματος



47 Τυπική δομή HTTP απάντησης με κωδικό 200

REST

Τα αρχικά REST σημαίνουν Representational State Transfer. Το REST αποτελεί μια τεχνοτροπία δόμησης λογισμικού, αποτελούμενη από ένα σύνολο οδηγιών και βέλτιστων πρακτικών για το σχεδιασμό Web εφαρμογών/APIs ,όπως της παρούσας. Το πρωτόκολλο που χρησιμοποιούμε σχεδόν πάντα σε REST WEB εφαρμογές είναι το ***HTTP.***

Όλη η φιλοσοφία της αρχιτεκτονικής REST περιστρέφεται γύρω από τους πόρους (resources), τον τρόπο πρόσβασης σε αυτούς και την αναπαράστασή τους.
Οι βασικοί περιορισμοί προκειμένου να θεωρηθεί μια πλατφόρμα **RESTful** είναι οι εξής:

- *Η εφαρμογή πρέπει να είναι μοντέλο πελάτη-εξυπηρετητή (client-server)*
  Αυτό σημαίνει ότι υπάρχει μια διεπαφή που χωρίζει τον πελάτη από τον εξυπηρετητή. Επιπλέον, ο πελάτης δεν απασχολείται με την αποθήκευση των δεδομένων, καθώς αυτό αποτελεί καθήκον του server . Αυτό συνεπάγεται πως αν και εφόσον το κομμάτι της διεπαφής πελάτη-εξυπηρετητή δεν αλλάζει, και ο πελάτης και ο εξυπηρετητής μπορούν να αλλαχθούν στην εσωτερική τους δομή κατά το δοκούν.

- *Ο εξυπηρετητής δεν αποθηκεύει στοιχεία για την κατάσταση του κάθε πελάτη (Stateless).*
  Αυτό συνεπάγεται το ότι ο πελάτης πρέπει να κρατάει όλα τα στοιχεία που του είναι απαραίτητα για να ορίζει την παρούσα κατάστασή του και να τα στέλνει στον εξυπηρετητή, μαζί με κάθε αίτημα. Επίσης, θα πρέπει να τα αλλάζει ανάλογα με κάθε απάντηση του εξυπηρετητή.

- *Θα πρέπει να γίνεται χρήση της κρυφής μνήμης (cache) για να αποθηκεύονται οι αποκρίσεις του εξυπηρετητή στον πελάτη.*
  Το παραπάνω δηλώνει πως και ο εξυπηρετητής θα πρέπει να στέλνει στις αποκρίσεις του στοιχεία για το αν αυτές θα είναι αποθηκεύσιμες στην κρυφή μνήμη καθώς και την ημερομηνία λήξης. Αυτός ο περιορισμός βοηθάει πάρα πολύ στην ταχύτητα.

- *Θα πρέπει να υπάρχει ομοιόμορφη διεπαφή για την επικοινωνία πελάτη-εξυπηρετητή.*

- *Πολυεπίπεδος σχεδιασμός*
  Κάθε αίτημα είναι πιθανό να περνάει από πολλά επίπεδα στον εξυπηρετητή για διάφορους λόγους (π.χ. στην δική μας περίπτωση έχουμε ένα επίπεδο εξουσιοδότησης πριν το κυρίως κομμάτι, ενώ σε άλλες πλατφόρμες είναι πιθανό να υπάρχει ένα επίπεδο που μοιράζει τα αιτήματα σε διαφορετικούς εξυπηρετητές για λόγους απόδοσης).

Οι παραπάνω περιορισμοί είναι αρκετά γενικοί. Στην περίπτωση μας, δηλαδή αυτή ενός Web-API, πρακτικά σημαίνουν ότι πρέπει να τηρούμε τα εξής:

- Τα URLs θα πρέπει να έχουν μορφή μονοπατιού σε φάκελο, π.χ:

HTTP://www.domain.com/api/profile/tracks

- Θα πρέπει να χρησιμοποιούμε ένα διαδικτυακό τύπο αναπαράστασης για την αναπαράσταση των πόρων μας. Ο τύπος που χρησιμοποιούμε εδώ είναι το JSON(JavaScript Object Notation), που είναι και ο πιο συνηθισμένος.
  Το JSON είναι στην ουσία ζευγάρια, όνομα μεταβλητής: τιμή, όπως φαίνεται στο παρακάτω παράδειγμα:

```
{
  "FirstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address":
        {
         "streetAddress": "21 2nd Street",
         "city": "New York",
         "state": "NY",
         "postalCode": 10021
        }
}
```

Το πλεονέκτημα του JSON είναι πως υποστηρίζεται από τις περισσότερες γλώσσες προγραμματισμού και είναι εύκολα κατανοητό και από άνθρωπο.

- Χρήση των στάνταρ HTTP μεθόδων (GET POST PUT DELETE).

Για να γίνει περισσότερο κατανοητό το παραπάνω δίνεται ένα πραγματικό παράδειγμα από την πλατφόρμα που αναπτύχθηκε γι αυτή την διπλωματική εργασία.

	HTTP μέθοδος			
URL πόρου	GET	POST	PUT	DELETE
HTTP://www.domain.com/api/profile/tracks	JSON πίνακας με όλες τις διαδρομές του χρήστη	Νέα διαδρομή για τον χρήστη (δεδομένα της διαδρομής περιλαμβάνονται στο αίτημα)	Αλλαγή μιας διαδρομής (δεν έχει υλοποιηθεί)	Διαγράφη διαδρομής (id της διαδρομής στέλνεται με το request)

5 Ενδεικτικές HTTP λειτουργίες

JSON Web Tokens (JWT)

Η ασφάλεια των πληροφοριών κάθε χρήστη αποτελεί ένα θέμα υψίστης σημασίας. Η μεγαλύτερη πρόκληση σε αυτό το θέμα είναι η αυθεντικοποίηση του χρήστη, δηλαδή η εξασφάλιση του ότι κάθε request στο API είναι από το χρήστη που έχει τα ανάλογα δικαιώματα να επεξεργαστεί-προβάλλει τις πληροφορίες που ζητάει.

Παλαιότερες λύσεις τύπου browser-cookies ή sessions κρίνονται ως ανεπαρκείς, καθώς

τα αιτήματα προς το API μπορεί να προέρχονται από μία πλατφόρμα που δεν τα υποστηρίζει. Για παράδειγμα, μια εφαρμογή κινητού τηλεφώνου ως λύση της αποθήκευσης στοιχείων στην ίδια την web εφαρμογή απορρίπτεται, καθώς δεν είναι RESTful (συμβατή με τα πρότυπα του REST). Τέλος, η λύση του να στέλνει ο χρήστης όνομα και κωδικό με κάθε αίτημα είναι μη ασφαλής.

Η λύση που θεωρήθηκε στην περίπτωσή μας ως αρκετά κομψή, ασφαλής και εύκολη να υλοποιηθεί είναι τα JSON Web Tokens (JWTs).

Ένα token θα μπορούσε να παραλληλιστεί με την κάρτα που μας δίνεται από το προσωπικό ασφαλείας στην είσοδο ενός κτιρίου, αφού δώσουμε τα στοιχεία μας και αυτά επιβεβαιωθούν. Με αυτή την κάρτα έχουμε πρόσβαση στα σημεία του κτιρίου, για τα οποία μας έχει δοθεί δικαιοδοσία. Ακριβώς ανάλογη είναι και η διαδικασία ενός Web token: Στην αρχή ο χρήστης δίνει τα στοιχεία σύνδεσής του και η εφαρμογή του απαντάει με ένα token, το οποίο αποθηκεύει και στέλνει με κάθε αίτημά του. Υπάρχουν διάφοροι τρόποι να σταλεί το token, π.χ. ως επικεφαλίδα στο HTTP request ή ως παράμετρος. Η κατοχή του token δίνει όλα τα δικαιώματα που έχει ο χρήστης (με τον ανάλογο συνδυασμό ονόματος – κωδικού) που ζήτησε την έκδοση του token.

Τα JWTs αποτελούν μία εκδοχή (σε μορφή JSON) του μηχανισμού των tokens που περιγράψαμε παραπάνω. Το JWT είναι ένα πρότυπο που βρίσκεται στη φάση του προσχέδιου. Γι’ αυτό το λόγο, θα αναφέρουμε μόνο τα βασικά του στοιχεία επιγραμματικά, προκειμένου να γίνει κατανοητή η χρήση του στο παρόν έργο.

Ένα JWT στην ουσία είναι ένα JSON αντικείμενο, που περιέχει ένα σετ αξιώσεων σε δικαιώματα πρόσβασης σε πόρους καθώς και μερικές ακόμα βοηθητικές πληροφορίες.Το JWT είναι URL – safe (αυτό πρακτικά σημαίνει πως μπορεί να μεταφερθεί μέσω HTTP). Παρακάτω βλέπουμε ένα υποδειγματικό JWT, που αποτελείται από δύο μέρη.

```
{
  "typ" : "JWT",
  "alg" : "HS256"
}
{
  "iss": "joe",
  "exp": 1300819380,
}
```

Το πρώτο μέρος περιγράφει τον τύπο του token (JWT) και τον αλγόριθμο κρυπτογράφησης που χρησιμοποιείται (HMAC SHA-256).

Το δεύτερο αποτελεί το σετ αξιώσεων. Το πεδίο iss (issuer) περιέχει το όνομα του χρήστη που ζήτησε έκδοση του JWT ,ενώ το πεδίο exp (expires) δίνει την ημερομηνία λήξης του token.

Ο αναγνώστης για περεταίρω εμβάθυνση παραπέμπεται στην τεκμηρίωση του προτύπου, όπου αναφέρονται ακόμα πολλά πεδία που προσδίδουν επιπλέον δυνατότητες.

Η διαδικασία δημιουργίας ενός JWT συνοψίζεται στα εξής βήματα:

1. Ο χρήστης στέλνει τα στοιχεία του.
2. Η web εφαρμογή επιβεβαιώνει τα στοιχεία του χρήστη.
3. Η web εφαρμογή δημιουργεί το JWT, συμπεριλαμβάνοντας όλα τα απαραίτητα στοιχεία ανάλογα με την περίπτωση.
4. Η web εφαρμογή κρυπτογραφεί το JWT με αλγόριθμο της επιλογής μας (ανάλογα πάντα με τις απαιτήσεις ασφαλείας).
5. Η web εφαρμογή απαντάει στον πελάτη με το κρυπτογραφημένο JWT.

Το βήμα 4 δεν είναι απαραίτητο στην περίπτωση που εξασφαλίζουμε ασφάλεια με κάποιον άλλον τρόπο (πχ HTTPs) .

Η διαδικασία ενός εξουσιοδοτημένου αιτήματος εμπεριέχει τα παρακάτω βήματα:

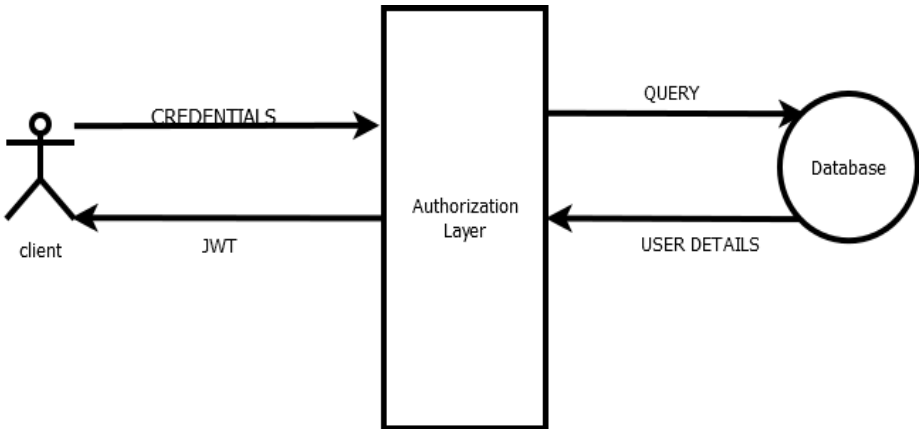
1. Ο πελάτης αποστέλλει το αίτημά του συμπεριλαμβάνοντας σε αυτό το JWT που έχει λάβει μετά την υποβολή των στοιχείων του.
2. Η Web εφαρμογή αποκρυπτογραφεί το JWT.
3. Σε περίπτωση που το Token έχει λήξει ή οι αξιώσεις που περιέχει δεν του παρέχουν πρόσβαση στο συγκεκριμένο πόρο, η εφαρμογή απαντά με το κατάλληλο μήνυμα λάθους, αλλιώς περνάμε στο βήμα 4.
4. Σε περίπτωση που στο βήμα 3 δεν έχει συμβεί κάποιο λάθος, η εφαρμογή απαντάει με τον πόρο που ζητήθηκε.

5.2.2 Η πλατφόρμα

Η Web πλατφόρμα που αναπτύχθηκε στα πλαίσια της διπλωματικής αυτής αποτελεί ένα REST API, που χρησιμοποίει JWTs για την αυθεντικοποίηση. Έχει σχεδιαστεί με βάση μια αρχιτεκτονική τριών επιπέδων (γραμμένη σε Node.js), μέσα από τα όποια περνάει (σχεδόν) κάθε αίτημα πελάτη στο API,ώστε να απαντηθεί. Αυτά τα τρία επίπεδα (με την σειρά που τα συναντάει ένα αίτημα) είναι:

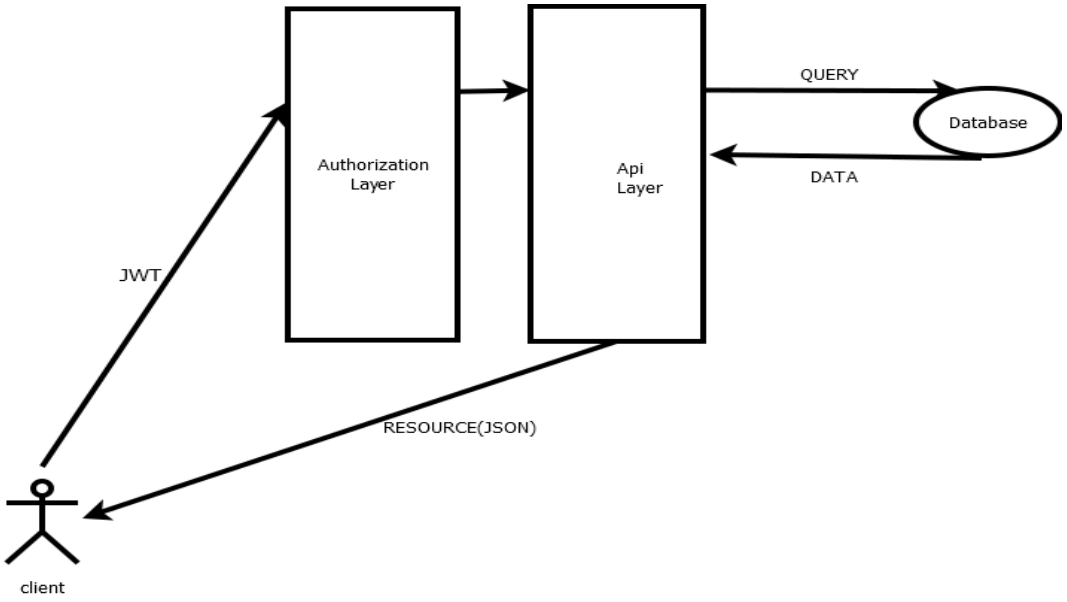
- *Επίπεδο αυθεντικοποίησης- αδειοδότησης (Authentication-Authorization Layer)*
  Αυτό το επίπεδο έχει ως ρόλο την διαχείριση των JWTs,κυρίως δηλαδή την αποκρυπτογράφηση τους και το πέρασμα των απαραίτητων πληροφοριών από αυτά στο επόμενο βήμα όπως επίσης και την έκδοση νέων JWTs.
- *Κυρίως εφαρμογή(API Layer)*
  Στο κομμάτι αυτό με βάση το αίτημα του πελάτη και τα στοιχεία που έχουν αντληθεί από το JWT στέλνεται ένα αίτημα στη βάση δεδομένων και όταν αυτή απαντήσει αποστέλλεται ο ζητούμενος πόρος στον πελάτη.
- *Βάση δεδομένων (Database)*
  Η βάση δεδομένων που χρησιμοποιήθηκε είναι η MongoDB. Σε αυτήν αποθηκεύονται όλα τα δεδομένα (δηλαδή προφίλ χρηστών και διαδρομές) και σε αυτήν τελικά καταλήγουν όλα τα αιτήματα για πόρους.

Παρακάτω βλέπουμε τη διαδικασία έκδοσης ενός νέου JWT από το επίπεδο αδειοδότησης.



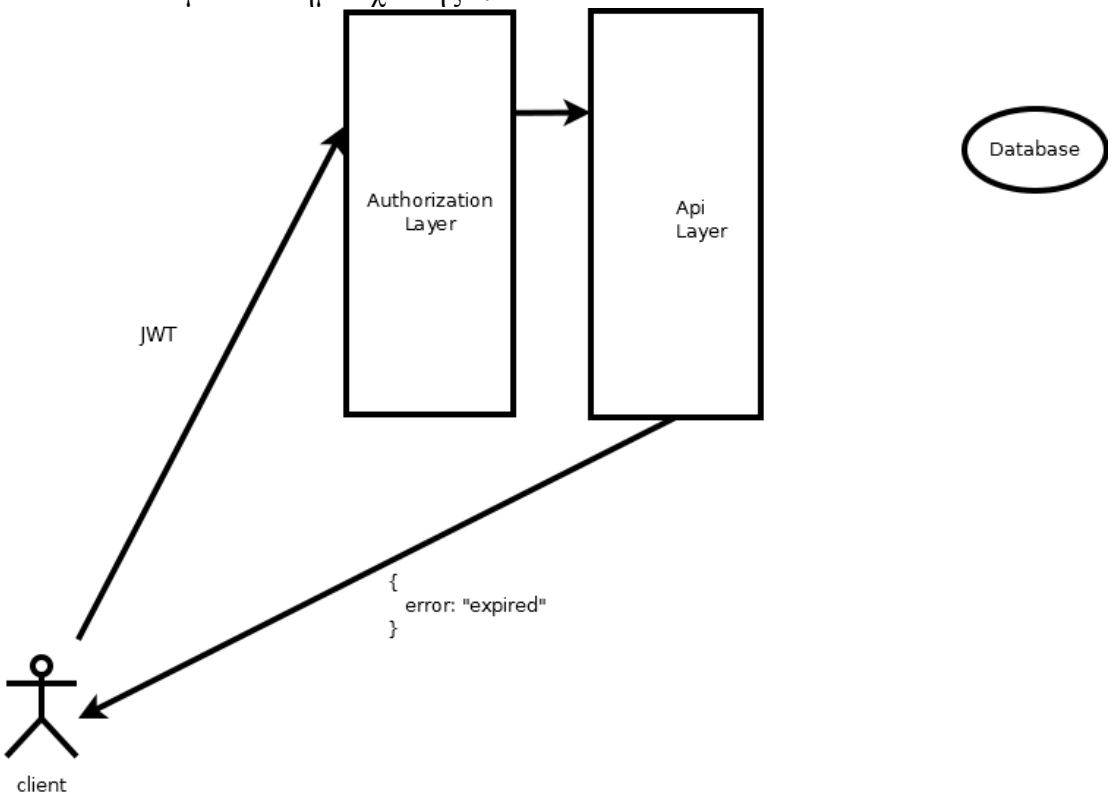
5 Διαδικασία έκδοσης JWT

Στην επόμενη εικόνα βλέπουμε τη γραφική απεικόνιση της διεκπεραίωσης ενός εξουσιοδοτημένου HTTP αιτήματος στο API. Εξουσιοδοτημένο σημαίνει πως μαζί με το αίτημα έχει σταλεί και ένα JWT, το οποίο δεν έχει λήξει και οι αξιώσεις που περιέχει (δηλαδή το όνομα χρήστη στην περίπτωση της εφαρμογής μας) είναι έγκυρες για τον πόρο που ζητάει.



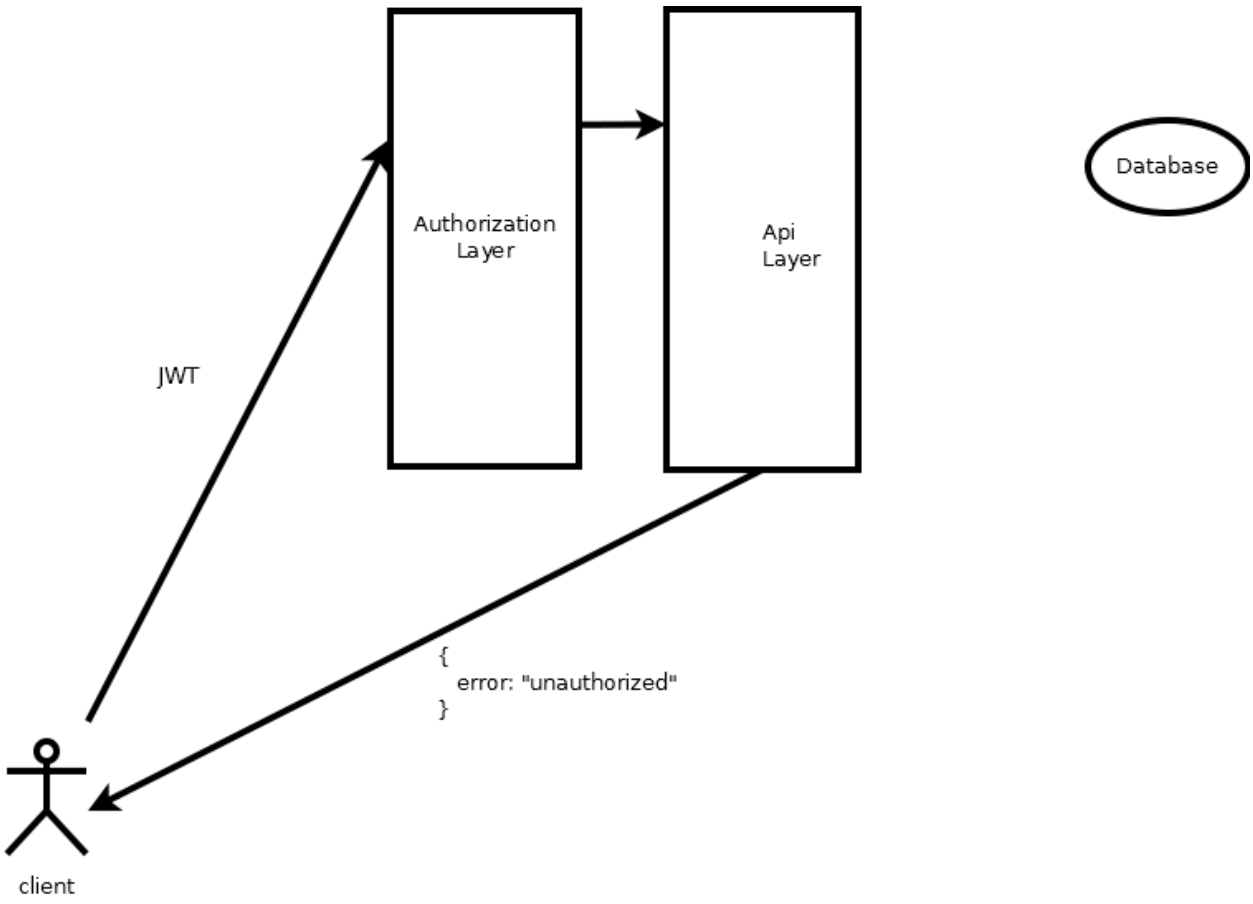
6 Τυπικό αίτημα που απαιτεί εξουσιοδότηση μέ JWT.

Στο επόμενο διάγραμμα γίνεται φανερή η διαδικασία που ακολουθείται, όταν το JWT που αποστέλλεται με το αίτημα έχει λήξει.



7 Απόκριση ληγμένου JWT

Εδώ παρατηρούμε τη διαδικασία που ακολουθείται, όταν το JWT το οποίο αποστέλλεται με το αίτημα, περιέχει όνομα χρήστη που δεν έχει δικαίωμα να εκτελέσει την συγκεκριμένη ενέργεια στον πόρο που ζητήθηκε.



8 Μη εξουσιοδοτημένη πρόσβαση

6 Παρουσίαση και ανάλυση καίριων σημείων του κώδικα

6.1 Mobile εφαρμογή

Η mobile εφαρμογή είναι μια native εφαρμογή Android. Το παραπάνω σημαίνει πως είναι γραμμένη σε γλώσσα προγραμματισμού Java. Ακόμη, χρησιμοποιεί τις υπηρεσίες της Google για προγραμματιστές, προκειμένου να κάνει χρήση των χαρτών Google αλλά και των Google Location Services. Επιπροσθέτως, χρησιμοποιεί την βιβλιοθήκη Volley της Google για τα HTTP αιτήματα στο REST API.
Η mobile εφαρμογή είναι αρκετά πολυσχιδής από άποψη κώδικα. Κι αυτό διότι εκτός από τις δύο βασικές activities, τα πέντε fragments και την υπηρεσία παρασκηνίου, που αποτελούν τις βασικές κλάσεις της εφαρμογής, υπάρχουν ακόμα αρκετές βοηθητικές κλάσεις. Παρακάτω θα κάνουμε παρουσίαση και συνοπτική ανάλυση των βασικότερων κομματιών του κώδικα.

6.1.1 IntroActivity

Η IntroActivity είναι η πρώτη δραστηριότητα που ξεκινάει όταν ο χρήστης ανοίγει την εφαρμογή. Στην οθόνη της εναλλάσσονται δύο fragments, οπότε ο κύριος ρόλος της είναι η διαχείριση αυτών των fragments.

```
1  @Override
2  protected void onCreate(Bundle savedInstanceState) {
3   super.onCreate(savedInstanceState);
4   setContentView(R.layout.activity_intro);
5   mBottomText = (TextView) findViewById(R.id.bottom_text);
6   // check if orientation changed
7   if (savedInstanceState != null) {
8       state = savedInstanceState.getBoolean("STATE");
9   } else {
10      state = false;
11  }
12  mBottomText.setOnClickListener(this);
13  if (!state) {
14      LoginFragment login = new LoginFragment();
15      FragmentTransaction ft = getFragmentManager().beginTransaction();
16      ft.replace(R.id.container, login);
17      ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN);
18      ft.commit();
19      mBottomText.setText(R.string.RegisterPrompt);
20
21  } else {
22      RegisterFragment register = new RegisterFragment();
23      FragmentTransaction ft = getFragmentManager().beginTransaction();
24      ft.replace(R.id.container, register);
25      ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN);
26      ft.commit();
27      mBottomText.setText(R.string.LoginPrompt);
28
29  }
30  mBottomText.setTextColor(Color.BLUE);
31 }
```

Μέθοδος onCreate της IntroActivity. Η μέθοδος αυτή καλείται κάθε φορά που δημιουργείται η δραστηριότητα.

Η δραστηριότητα καταστρέφεται και δημιουργείται ξανά πολλές φορές . Επομένως, η μέθοδος onCreate καλείται αρκετά συχνά. Σε αυτή την μέθοδο αυτό που κάνουμε είναι να εμφανίζουμε την διεπαφή ενός από τα δύο fragments ανάλογα με την επιλογή του χρήστη. Το if στην γραμμή 13 ελέγχει ποιο από τα δύο ήταν ενεργό πριν την καταστροφή και την αναδημιουργία της δραστηριότητας (το προεπιλεγμένο είναι το LoginFragment). Σε περίπτωση που η συσκευή άλλαξε προσανατολισμό, αποθηκεύουμε ποιο Fragment ήταν ενεργό. Γι' αυτήν τη διαδικασία υπάρχει η μέθοδος onSaveInstanceState, στην οποία αποθηκεύουμε την Boolean μεταβλητή state, που μας δείχνει ποιο από τα δύο fragments προβάλλεται την στιγμή που η δραστηριότητα «σκοτώνεται» (γραμμή 5 της μεθόδου onSaveInstanceState). Στην γραμμή 8 της onCreate ανακτούμε την τιμή της μεταβλητής αυτής (εφόσον υπάρχει, αλλιώς βάζουμε την προεπιλεγμένη).

```
1 public void onSaveInstanceState(Bundle savedInstanceState)
2   {
3   //
4   // Save the user's current fragment (login or register)
5   savedInstanceState.putBoolean("STATE", state);
6
7   // Always call the superclass so it can save the view hierarchy state
8   super.onSaveInstanceState(savedInstanceState);
9   }
```

Η μέθοδος onSaveInstenceState καλείται όταν καταστρέφεται η δραστηριότητα, με σκοπό την αποθήκευση μεταβλητών και τιμών που μας χρειάζονται για να επαναφέρουμε την εφαρμογή στην προηγούμενη κατάσταση όταν αναδημιουργηθεί.

Login Fragment

Το LoginFragment έχει ως ρόλο την δημιουργία της διεπαφής για τη σύνδεση χρήστη, καθώς και την διεκπεραίωση όλης της επικοινωνίας που απαιτείται με την Web εφαρμογή (αποστολή στοιχείων και λήψη JWT).

```
1  public View onCreateView(LayoutInflater inflater, ViewGroup container,
2  Bundle savedInstanceState) {
3   View rootView = inflater.inflate(R.layout.loginfragment, container, false);
4   mLogin = (Button) rootView.findViewById(R.id.login);
5   mUsername = (EditText) rootView.findViewById(R.id.username);
6   mPassword = (EditText) rootView.findViewById(R.id.password);
7   mRememberme = (CheckBox) rootView.findViewById(R.id.rememberme);
8   mLogin.setOnClickListener(this);
9   SharedPreferences credentials = getActivity().getPreferences(Context.MODE_PRIVATE);
10  mUsername.setText(credentials.getString("username", ""));
11  mPassword.setText(credentials.getString("password", ""));
12  return rootView;
13 }
```

Η μέθοδος onCreateView του LoginFragment. Σε αυτή τη μέθοδο δημιουργείται η διεπαφή που έχει οριστεί στο αρχείο XML.

Στην μέθοδο onCreateView δημιουργούμε την διεπαφή του fragment. Άξια αναφοράς είναι η γραμμή 9, στην οποία εάν σε προηγούμενη έναρξη της εφαρμογής ο χρήστης το έχει επιλέξει , ανακτώνται από το αντικείμενο SharedPreferences το όνομα χρήστη και ο κωδικός και εμφανίζονται στα αντίστοιχα πεδία, ώστε να μην χρειάζεται να πληκτρολογηθούν ξανά. Τέλος, στην γραμμή 8 καταχωρούμε και την ίδια την κλάση του LoginFragment,ως onClicklistener για το κουμπί login (η μέθοδος OnClick υπάρχει στον κώδικα της κλάσης, αλλά δεν παρατίθεται εδώ για οικονομία χώρου).

```
1 private void Make_login_request() {
2   pDialog = new ProgressDialog(getActivity());
3   pDialog.setMessage("Login in ");
4   showProgressDialog();
5   JSONObject credentials = null;
6   try {
7       credentials = new JSONObject();
8       credentials.put("username", mUsername.getText().toString());
9       credentials.put("password", mPassword.getText().toString());
10  } catch (JSONException e) {
11      // TODO Auto-generated catch block
12      e.printStackTrace();
13  }
14  JsonObjectRequest jsonObjReq = new JsonObjectRequest(Method.POST,
15  URLS.URL_LOGIN, credentials,
16  new Response.Listener < JSONObject > () {
17      @Override
18      public void onResponse(JSONObject response) {
19          try {
20              response.getString("token");
21              if (mRememberme.isChecked()) {
22                  // save username and password for future use in
23                  // shared preferences
24                  SharedPreferences credentials = getActivity()
25                      .getPreferences(Context.MODE_PRIVATE);
26                  SharedPreferences.Editor editor = credentials.edit();
27                  editor.putString("username", mUsername.getText().toString());
28                  editor.putString("password", mPassword.getText().toString());
29                  editor.commit();
30              }
31              Launch_app(response.getString("token"));
32          } catch (JSONException e) {
33              // TODO Auto-generated catch block
34              try {
35                  Toast.makeText(getActivity(),
36                  response.getString("error"),
37                  Toast.LENGTH_LONG).show();
38              } catch (JSONException e1) {
39                  // TODO Auto-generated catch block
40                  e1.printStackTrace();
41              }
42          }
43          hideProgressDialog();
44      }
45  }, new Response.ErrorListener() {
46      @Override
47      public void onErrorResponse(VolleyError error) {
48          VolleyLog.d(TAG, "Error: " + error.getMessage());
49          hideProgressDialog();
50      }
51  }) {
52  };
53  // Adding request to request queue
54  AppController.getInstance().addToRequestQueue(jsonObjReq, TAG);
55}
```

Η μέθοδος Make_login_request αναλαμβάνει όλη τη διαδικασία εισαγωγής του χρήστη στην εφαρμογή από το πάτημα του Login και μετά.

Η Make_login_request καλείται αφού πατηθεί το κουμπί Login. Η μέθοδος στέλνει ένα HTTP POST αίτημα με τα στοιχεία του χρήστη στο API , και εφόσον η απάντηση περιλαμβάνει ένα

JWT τότε εκκινεί την εφαρμογή. Συγκεκριμένα, από την γραμμή 14 έως την 52 δημιουργείται το αντικείμενο του αιτήματος, ενώ στην γραμμή 54 το προσθέτουμε στην γραμμή αποστολής αιτημάτων. Ιδιαίτερη προσοχή αξίζει να δώσουμε στις εξής γραμμές:

- *24-29:* Σε αυτό το κομμάτι κώδικα, εφόσον το πεδίο «Remember me» έχει τσεκαριστεί, αποθηκεύονται στο αντικείμενο SharedPreferences τα στοιχεία του χρήστη και είναι διαθέσιμα σε μελλοντικές εκτελέσεις της εφαρμογής.
- Στην γραμμή *31*, αν έχουμε λάβει απάντηση με token από το API,εκτελείται η μέθοδος Launch_app με παράμετρο το token που μόλις λάβαμε.

```
1 private void Launch_app(String token)
2     {
3         Intent intent = new Intent(getActivity(), MainActivity.class);
4         intent.putExtra("token", token);
5         intent.putExtra("Username", mUsername.getText().toString());
6         startActivity(intent);
7     }
```

Η μέθοδος Launch_app εκκινεί την κύρια εφαρμογή.

Η μέθοδος Launch_app ξεκινά την κεντρική δραστηριότητα λαμβάνοντας ως επιπλέον παραμέτρους το JWT και το όνομα χρήστη. Το JWT είναι απαραίτητο προκειμένου να διεκπεραιώνονται όλα τα HTTP αιτήματα που χρειάζονται αυθεντικοποίηση του χρήστη. Παράλληλα, το όνομα χρήστη χρειάζεται διότι εμφανίζεται μέσα στην κύρια εφαρμογή (Welcome + όνομα χρήστη).

RegisterFragment

Το RegisterFragment έχει ως κύριο σκοπό τη δημιουργία ενός νέου προφίλ. Για το λόγο αυτό έχει δική του διεπαφή για την συμπλήρωση στοιχείων, ενώ επικοινωνεί και με το REST API για την δημιουργία του καινούριου προφίλ.

```
1 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedIn-
stanceState)
2 {
3       View rootView = inflater.inflate(R.layout.registerfragment, container,false);
4       mRegister = (Button) rootView.findViewById(R.id.register);
5       mUsername = (EditText) rootView.findViewById(R.id.username);
6       mPassword1 = (EditText) rootView.findViewById(R.id.password1);
7       mPassword2 = (EditText) rootView.findViewById(R.id.password2);
8       mName = (EditText) rootView.findViewById(R.id.name);
9       mSurname = (EditText) rootView.findViewById(R.id.surname);
10      mRegister.setOnClickListener(this);
11      return rootView;
12}
```

Η μέθοδος onCreateView δημιουργεί τη διεπαφή από το αρχείο XML.

Όπως και στο LoginFragment, η onCreateView δημιουργεί τη διεπαφή από το αρχείο XML και καταχωρεί την κλάση ως onClicklistener για το κουμπί register.

```
1  private void Make_register_request()
2     {
3   pDialog = new ProgressDialog(getActivity());
4   showProgressDialog();
5   JSONObject credentials = null;
6   try
7   {
8       credentials = new JSONObject();
9       credentials.put("username", mUsername.getText().toString());
10      credentials.put("password", mPassword1.getText().toString());
11      credentials.put("name", mName.getText().toString());
12      credentials.put("surname", mSurname.getText().toString());
13
14  } catch (JSONException e)
15  {
16      e.printStackTrace();
17  }
18  JsonObjectRequest jsonObjReq = new JsonObjectRequest(Method.POST,
19      URLS.URL_PROFILE, credentials,
20      new Response.Listener<JSONObject>()
21      {
22
23          @Override
24          public void onResponse(JSONObject response)
25          {
26          hideProgressDialog();
27          try
28          {
29              response.getString("ok");
30              Toast.makeText(getActivity(),
31                  "Registration completed please login",
32                  Toast.LENGTH_LONG).show();
33          } catch (JSONException e)
34          {
35              try
36              {
37              Toast.makeText(getActivity(),
38                  response.getString("error"),
39                  Toast.LENGTH_LONG).show();
40              } catch (JSONException e1)
41              {
42              e1.printStackTrace();
43              }
44              e.printStackTrace();
45          }
46          Log.v(TAG, response.toString());
47          }
48      }, new Response.ErrorListener()
49      {
50          @Override
51          public void onErrorResponse(VolleyError error)
52          {
53          VolleyLog.d(TAG, "Error: " + error.getMessage());
54          hideProgressDialog();
55          }
56      })
57  // Adding request to request queue
58  AppController.getInstance().addToRequestQueue(jsonObjReq, TAG);
59 }
```

Η μέθοδος Make_register_request αναλαμβάνει όλες τις ενέργειες, όταν ο χρήστης πατήσει το κουμπί register.

Η μέθοδος Make_register_request καλείται (μετά από έλεγχο εγκυρότητας των στοιχείων), όταν πατηθεί το κουμπί Register. Ο κύριος ρόλος της είναι η αποστολή του HTTP αιτήματος στο REST API. Συγκεκριμένα, στις γραμμές 8-12 δημιουργείται το JSON αντικείμενο με τα στοιχεία του χρήστη ενώ στις γραμμές 18-55 δημιουργείται το αντικείμενο του HTTP αιτήματος, το οποίο στην γραμμή 58 προστίθεται στην ουρά των εκκρεμών αιτημάτων.

6.1.2 MainActivity

Η MainActivity αποτελεί την κεντρική δραστηριότητα της εφαρμογής. Σκοπός της είναι η οργάνωση των τριών θυγατρικών Fragments σε καρτέλες καθώς και η διαχείριση της υπηρεσίας παρασκηνίου(εκκίνηση, τερματισμός και πρόσβαση στην υπηρεσία γίνονται μέσω της δραστηριότητας).

```
1  protected void onCreate(Bundle savedInstanceState)
2      {
3    super.onCreate(savedInstanceState);
4    setContentView(R.layout.activity_main);
5    Bundle extras = getIntent().getExtras();
6    //get the username for display and token for authorized requests
7    if (extras != null)
8    {
9        setToken(extras.getString("token"));
10       Username = extras.getString("Username");
11       Log.v(TAG, Username);
12   }
13   viewPager = (ViewPager) findViewById(R.id.pager);
14   actionBar = getActionBar();
15   mAdapter = new TabsPagerAdapter(getSupportFragmentManager());
16   viewPager.setAdapter(mAdapter);
17   actionBar.setHomeButtonEnabled(false);
18   actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
19   actionBar.setTitle("Welcome " + Username);
20   for (String tab_name : tabs)
21   {
22       actionBar.addTab(actionBar.newTab().setText(tab_name)
23           .setTabListener(this));
24
25   }
26   viewPager.setOnPageChangeListener(new ViewPager.OnPageChangeListener()
27   {
28       @Override
29       public void onPageSelected(int position)
30       {
31       // on changing the page
32       // make respected tab selected
33       actionBar.setSelectedNavigationItem(position);
34       }
35       @Override
36       public void onPageScrolled(int arg0, float arg1, int arg2)
37       {
38       }
39       @Override
40       public void onPageScrollStateChanged(int arg0)
41       {
42       }
43   });
44     }
```

Στη μέθοδο onCreate δημιουργείται η διεπαφή με τις καρτέλες της κεντρικής εφαρμογής.

Η μέθοδος onCreate κατά τα γνωστά καλείται όταν δημιουργηθεί η δραστηριότητα. Σε αυτή τη δραστηριότητα κύριος ρόλος της είναι να δημιουργεί τη διεπαφή των καρτελών με χρήση ενός viewPager(γρ. 13).

```
1  private ServiceConnection myConnection = new ServiceConnection() {
2   // on connection return the service to activity so we can access it
3   // directly
4   public void onServiceConnected(ComponentName className, IBinder service){
5       LocalBinder binder = (LocalBinder) service;
6       myService = binder.getService();
7       isBound = true;
8   }
9   public void onServiceDisconnected(ComponentName arg0) {
10      isBound = false;
11  }
12
13 };
```

Τροποποιημένη έκδοση της ServiceConnection, στην όποια παίρνουμε αναφορά στο αντικείμενο της υπηρεσίας όταν συνδεθούμε με αυτήν.

Εκτός από τη διεπαφή, η άλλη κύρια λειτουργία της δραστηριότητας είναι η διαχείριση της υπηρεσίας παρασκηνίου που χρησιμοποιούμε. Γι' αυτό το λόγο γράφτηκε μια δική μας έκδοση της κλάσης ServiceConnection, που χρησιμοποιείται για διαχείριση υπηρεσιών. Κάναμε override τη μέθοδο onServiceConnected (γρ. 4) για να μας επιστρέφει μία αναφορά στην υπηρεσία (γρ. 6),μέσω της οποίας μπορούμε να καλέσουμε τις δημόσιες μεθόδους σε αυτήν, άρα και να την διαχειριστούμε.

```
1 protected void onResume()
2{
3   super.onResume();
4   // bind (and start the service)
5   Intent intent = new Intent(this, TrackingService.class);
6   bindService(intent, myConnection, Context.BIND_AUTO_CREATE);
7}
```

Στην onResume φροντίζουμε να συνδεθούμε με την υπηρεσία, γεγονός που συνεπάγεται και την αυτόματη εκκίνησή της.

Μέσα στην μέθοδο onResume ξεκινάμε την υπηρεσία. Η onResume τρέχει αφού έχει τρέξει η onCreate, άρα έχει δημιουργηθεί η διεπαφή.

```
1 public void onDestroy()
2{
3   super.onDestroy();
4   //unbind and destroy service
5   unbindService(myConnection);
6}
```

Στην onDestroy φροντίζουμε να αποδεσμεύσουμε την υπηρεσία, ώστε να τερματιστεί.

Όταν καταστραφεί η δραστηριότητα, αποδεσμεύουμε και την υπηρεσία με απότοκο να καταστραφεί και αυτή, εφόσον δεν είναι δεσμευμένη από άλλη δραστηριότητα.

TabsPagerAdapter

```
1 public class TabsPagerAdapter extends FragmentPagerAdapter
2 {
3     public TabsPagerAdapter(FragmentManager fm)
4     {
5   super(fm);
6     }
7     @Override
8     public Fragment getItem(int index)
9     {
10      switch (index)
11      {
12      case 0:
13         return new RunFragment();
14      case 1:
15         return new MyRunsFragment();
16      case 2:
17        return new ProfileFragment();
18       }
19      return null;
20    }
21    @Override
22    public int getCount()
23    {
24      return 3;
25    }
26}
```

Ο TabsPagerAdapter καθορίζει ποια fragments και με ποια σειρά θα εμφανίζονται στις αντίστοιχες καρτέλες.

Η κλάση TabsPagerAdapter είναι ένας εκτεταμένος FragmentPagerAdapter . Κάνει Override τη μέθοδο getItem ,ώστε να επιστρέφει ένα από τα τρία fragments ανάλογα με το δείχτη που λαμβάνεται ως παράμετρος (γραμμές 8-18).

RunFragement

Το RunFragment είναι ένα fragment επιφορτισμένο με την διαχείριση της προβολής των στοιχείων της υπό καταγραφή δραστηριότητας, την εκκίνηση- παύση της καταγραφής καθώς και την αποθήκευση των στοιχείων της στην πλατφόρμα. Όλες αυτές οι λειτουργίες προϋποθέτουν αρκετά μεγάλο όγκο κώδικα, που περιλαμβάνει ενδεικτικά:

- Δημιουργία διεπαφής που εμπεριέχει χάρτη καθώς και σχεδιασμό στο χάρτη
- Επικοινωνία με την υπηρεσία ιχνηλάτησης
- HTTP Requests στην πλατφόρμα

Κατά τα γνωστά, η onCreateView χρησιμοποιεί το XML αρχείο για να δημιουργήσει τη διεπαφή και να ορίσει onClickListeners για τα δύο πλήκτρα του fragment. Ιδιαίτερη προσοχή αξίζει να δώσουμε στις γραμμές 15 και 16, στις οποίες αντικαθιστούμε το κενό fragment που έχουμε ορίσει μέσα στο XML (εντός του RunFragment) με το χάρτη.

```
1 public View onCreateView(LayoutInflater inflater, ViewGroup container,Bundle savedInstanceS-
tate)
2     {
3
4       View rootView = inflater.inflate(R.layout.runfragment, container, false);
5       mToken = ((MainActivity) getActivity()).getToken();
6       mDistanceText = (TextView) rootView.findViewById(R.id.dst);
7       mStatus = (TextView) rootView.findViewById(R.id.status);
8       mStatus.setTextColor(Color.RED);
9       mStop = (Button) rootView.findViewById(R.id.stop);
10      Chronometer mCronometer = (Chronometer) rootView.findViewById(R.id.chronometer1);
11      mTimer.setTimer(mCronometer);
12      mStart = (Button) rootView.findViewById(R.id.start);
13      mStart.setOnClickListener(this);
14      mStop.setOnClickListener(this);
15      mMapFragment = CustomMapFragment.newInstance();
16      getChildFragmentManager().beginTransaction().replace(R.id.myfragment, mMapFrag-
ment).commit();
17      if (!mTimer.getFirstStart())
18      {
19          mTimer.startTimer();
20      }
21      return rootView;
22    }
```

Στην onCreate view δημιουργούνται όλα τα στοιχεία της διεπαφής του fragment, συμπεριλαμβανομένου και του χάρτη.

Για να σχεδιάζει το RunFragment τη διαδρομή του χρήστη στο χάρτη είναι απαραίτητη η επικοινωνία με την υπηρεσία ιχνηλάτησης. Η επικοινωνία αυτή μπορεί να γίνει με δύο τρόπους, ανάλογα με την κάθε περίπτωση.

Ο πρώτος τρόπος είναι η απευθείας πρόσβαση στην υπηρεσία μέσω της κεντρικής δραστηριότητας. Έτσι, μπορούμε να καλέσουμε οποιαδήποτε δημόσια μέθοδο της υπηρεσίας με κώδικα της μορφής:

```
((MainActivity) getActivity()).getMyService().SOME_PUBLIC_METHOD();
```

Η παραπάνω μέθοδος είναι πολύ ισχυρή, όσον αφορά το επίπεδο πρόσβασης στην υπηρεσία, αλλά έχει το μειονέκτημα ότι δεν επιτρέπει επικοινωνία από την υπηρεσία στο fragment (παρά μόνο το αντίθετο) ούτε επικοινωνία σε πραγματικό χρόνο. Γι' αυτό το λόγο χρησιμοποιούμε τον δεύτερο τρόπο, ο οποίος βασίζεται σε ένα χαρακτηριστικό του Android, τον LocalBroadcastManager. Συνοπτικά, με τον LocalBroadcastManager μπορούμε να στείλουμε μηνύματα μεταξύ διαφόρων στοιχείων της ίδιας εφαρμογής (ο γενικός BroadcastManager που επιτρέπει επικοινωνία και μεταξύ διαφορετικών εφαρμογών δε θα μας απασχολήσει εδώ). Ο τρόπος που δουλεύει το συγκεκριμένο μόρφωμα είναι ο εξής:

- Σε κάποιο (ή σε κάποια) από τα στοιχεία της εφαρμογής στέλνουμε ένα Broadcast, για να ενημερώσουμε σχετικά με κάποιο γεγονός (π.χ. αλλαγή τοποθεσίας). Αυτό γίνεται με κώδικα της παρακάτω μορφής:

```
Intent intent = new Intent("location_changed");
LocalBroadcastManager.getInstance(this).sendBroadcast(intent);
```

- Από ένα ή περισσότερα σημεία μπορούμε να λάβουμε το συγκεκριμένο Broadcast και να εκτελέσουμε τον κώδικα της επιλογής μας, όταν ληφθεί. Αυτό επιτυγχάνεται κάνοντας override τη μέθοδο onReceive της κλάσης BroadcastReceiver και καταχωρώντας το Receiver.

Χρησιμοποιώντας τα παραπάνω ορίζουμε έναν BroadcastReceiver στο RunFragment, που λαμβάνει τα BroadCasts, τα οποία στέλνει η υπηρεσία κάθε φορά που ο χρήστης αλλάζει τοποθεσία. Ύστερα, ενημερώνουμε το ίχνος κατάλληλα.

```
private BroadcastReceiver mMessageReceiver = new BroadcastReceiver()    {@Override
    // When we receive e location change from TrackingService
    public void onReceive(Context context, Intent intent) {
        Log.v(TAG, "onReceive");
        mDistanceText.setText(((MainActivity) getActivity()).getMyService()
            .getDistance() + " m");
        // update distance and map line
        if (mMap != null) {
            options.add(((MainActivity) getActivity()).getMyService()
                .getLastPoint());
            options.width(3);
            options.color(Color.GREEN);
            mMap.addPolyline(options);
            mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(
            ((MainActivity) getActivity()).getMyService()
                .getLastPoint(), ZOOM_LEVEL));
        }
    }
};
```

Στον BroadCast Receiver, μόλις λάβουμε αλλαγή τοποθεσίας, προσθέτουμε τη νέα τοποθεσία στη γραμμή του χάρτη.

```
1 public void onResume()
2     {
3   // when the user returns to RunFragment we must bring everything back
4   // from TrackingService
5   super.onResume();
6   // redraw the line by getting the list of waypoints from service
7   if (((MainActivity) getActivity()).getMyService() != null)
8   {
9       if (((MainActivity) getActivity()).getMyService().isStarted())
10      {
11      mStart.setEnabled(false);
12      // get the list of point from service and redraw the line
13      options = new PolylineOptions();
14      options.addAll(((MainActivity) getActivity()).getMyService().getCoursePoints());
15      options.width(3);
16      options.color(Color.GREEN);
17      mMap.addPolyline(options);
18      mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(
19          ((MainActivity) getActivity()).getMyService()
20              .getLastPoint(), ZOOM_LEVEL));
21      mDistanceText.setText(((MainActivity) getActivity())
22          .getMyService().getDistance() + " m");
23      }
24      if (((((MainActivity) getActivity()).getMyService().isStarted())))
25      {
26      mStatus.setText(R.string.Running);
27      mStatus.setTextColor(Color.GREEN);
28      }
29  }
30  // start listening to location changes
31  LocalBroadcastManager.getInstance(getActivity()).registerReceiver(mMessageReceiver, new
IntentFilter("location_changed"));
32    }
```

Στην onResume ανακτούμε τα δεδομένα της παρούσας δραστηριότητας από την υπηρεσία, ώστε να ξαναδημιουργήσουμε την αναπαράστασή της. Επίσης, ξεκινάμε να λαμβάνουμε τις αλλαγές τοποθεσίας που στέλνει η υπηρεσία.

Στην onResume μέθοδο διεκπεραιώνονται δυο κύριες εργασίες :Αρχικά ανασχεδιάζεται η πορεία του χρήστη σε περίπτωση που υπάρχει αθλητική δραστηριότητα σε εξέλιξη (γραμμές 7-26). Προσοχή αξίζει να δοθεί σε γραμμές όπως η 14, στην οποία υπάρχει απευθείας πρόσβαση στην υπηρεσία παρασκηνίου. Στη συνέχεια, στη γραμμή 31 καταχωρούμε τον Receiver, ώστε να λαμβάνουμε τις αλλαγές τοποθεσίας από την υπηρεσία.

```
1 public void onPause()
2     {
3       super.onPause();
4       // stop listening to TrackingService for location changes
5       LocalBroadcastManager.getInstance(getActivity()).unregisterReceiver(mMessageReceiver);
6     }
```

Στην onPause σταματάμε τη λήψη ενημερώσεων αλλαγής τοποθεσίας που στέλνει η υπηρεσία.

```
 1 private void save_track() {
 2     pDialog = new ProgressDialog(getActivity());
 3     pDialog.setMessage("Saving track ");
 4     showProgressDialog();
 5     String line = PolyUtil.encode(((MainActivity) getActivi-
ty()).getMyService().getCoursePoints());
 6     Calendar c = Calendar.getInstance();
 7     String time = (String) mTimer.getTimer().getText();
 8     String date = c.get(Calendar.DATE) + "-" + (1 + c.get(Calendar.MONTH))+"-"+
c.get(Calendar.YEAR);
 9     String distance = (String) mDistanceText.getText();
10     JSONObject new_run = new JSONObject();
11     try {
12         new_run.put("track", line);
13         new_run.put("date", date);
14         new_run.put("distance", distance);
15         new_run.put("time", time);
16     } catch (JSONException e) {
17         e.printStackTrace();
18     }
19     JsonObjectRequest save_req = new JsonObjectRequest(Method.POST,
20     URLS.URL_TRACKS, new_run, new Response.Listener < JSONObject > () {
21
22         @Override
23         public void onResponse(JSONObject response) {
24
25             hideProgressDialog();
26             reset_fragment();
27             try {
28                 if (response.get("error") != null) {
29                     Toast.makeText(getActivity(), (CharSequence) response.get("error"),
30                     Toast.LENGTH_LONG).show();
31                 }
32             } catch (JSONException e) {
33
34                 Toast.makeText(getActivity(), "Track saved",
35                 Toast.LENGTH_LONG).show();
36             }
37         }
38     }, new Response.ErrorListener() {
39
40         @Override
41         public void onErrorResponse(VolleyError arg0) {
42             hideProgressDialog();
43             Toast.makeText(getActivity(), "Error",
44             Toast.LENGTH_LONG).show();
45         }
46     }) {
47         /**
48          * Passing some request headers token and content type
49          * */@Override
50         public Map < String, String > getHeaders() throws AuthFailureError {
51             HashMap < String, String > headers = new HashMap < String, String > ();
52             headers.put("Content-Type", "application/json");
53             headers.put("x-access-token", mToken);
54             return headers;
55         }
56     };
57     AppController.getInstance().addToRequestQueue(save_req, TAG);
58}
```

Η save_track χειρίζεται τα HTTP αιτήματα του fragment προς στην πλατφόρμα.

Η save_track αποτελεί τη μέθοδο που διεκπεραιώνει την απαραίτητη επικοινωνία με την πλατφόρμα. Πιο συγκεκριμένα, αποστέλλει σε αυτήν στοιχεία της τρέχουσας δραστηριότητας προς αποθήκευση. Αξιοσημείωτες είναι οι εξής γραμμές κώδικα:

- Γραμμή 5, στην οποία το ίχνος της διαδρομής κωδικοποιείται σε μια συμβολοσειρά, για να καθίσταται δυνατή η αποθήκευσή του στη βάση δεδομένων.

- Και οι γραμμές 48-55, στις οποίες περνάμε το JWT στις επικεφαλίδες του HTTP αιτήματος ,έτσι ώστε το αίτημα να εγκριθεί από την πλατφόρμα, κάτι που κάνουμε σε όλα τα αιτήματα που απαιτούν εξουσιοδότηση.

MyRunsFragment

Το συγκεκριμένο fragment έχει ως σκοπό την ανάκτηση των δραστηριοτήτων του χρήστη από την πλατφόρμα και την προβολή τους σε λίστα. Επίσης, έχει τη δυνατότητα διαγραφής δραστηριότητας.

```
1 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedIn-
stanceState)
2     {
3       View rootView = inflater.inflate(R.layout.myrunfragment, container, false);
4       listView = (ListView) rootView.findViewById(R.id.list);
5       mRefresh = (Button) rootView.findViewById(R.id.refresh);
6       listView.setOnItemClickListener(this);
7       mMapFragment = CustomMapFragment.newInstance();
8       getChildFragmentManager().beginTransaction().replace(R.id.map_container, mMapFrag-
ment).commit();
9       mRefresh.setOnClickListener(this);
10      return rootView;
11    }
```

Στην onCreate ως συνήθως υλοποιείται η δημιουργία της διεπαφής.

```
1 public void onItemClick(AdapterView<?> arg0, View arg1, int position,long arg3)
2     {
3       PolylineOptions options = new PolylineOptions();
4       options.width(3);
5       options.color(Color.GREEN);
6       List<LatLng> track = PolyUtil.decode(mTracks.get(position).getTrack());
7       options.addAll(track);
8       LatLngBounds.Builder b = new LatLngBounds.Builder();
9       for (LatLng m : track)
10      {
11          b.include(m);
12      }
13      if (mMap != null)
14      {
15          mMap.clear();
16          LatLngBounds bounds = b.build();
17          CameraUpdate cu = CameraUpdateFactory.newLatLngBounds(bounds, 100,100, 1);
18          mMap.animateCamera(cu);
19          mMap.addPolyline(options);
20      }
21    }
```

Η onItemClick καλείται σε κάθε click αντικειμένου της λίστας.

Κάθε φορά που ο χρήστης κλικάρει ένα στοιχείο της λίστας ,εμφανίζεται το ίχνος του στο χάρτη, δίπλα. Την συγκεκριμένη ενέργεια αναλαμβάνει η παραπάνω μέθοδος. Αναλυτικότερα, στις γραμμές 9 -16 ρυθμίζουμε τα όρια και το zoom του χάρτη, ώστε να περιλαμβάνει ακριβώς αυτό το ίχνος (κι όχι άχρηστα κομμάτια) και στην γραμμή 19 το εμφανίζουμε.

```
1  private void Make_track_request()
2      {
3    pDialog = new ProgressDialog(getActivity());
4    pDialog.setMessage("Getting your tracks");
5    showProgressDialog();
6    // Creating volley request obj
7    JsonArrayRequest trackReq = new JsonArrayRequest(URLS.URL_TRACKS,
8        new Response.Listener<JSONArray>()
9        {
10           @Override
11           public void onResponse(JSONArray response)
12           {
13           // Parsing json
14           mTracks.clear();
15           for (int i = 0; i < response.length(); i++)
16           {
17               try
18               {
19               JSONObject obj = response.getJSONObject(i);
20               mTracks.add(new Track(obj.getString("date"),
21                   obj.getString("time"), obj
22                       .getString("track"), obj
23                       .getString("_id"), obj
24                       .getString("distance")));
25               Log.v(TAG,""+mTracks.get(i).getDate());
26               } catch (JSONException e)
27               {
28               e.printStackTrace();
29               }
30           }
31
32           // notifying list adapter about data changes
33           // so that it renders the list view with updated data
34           adapter.notifyDataSetChanged();
35           hideProgressDialog();
36           }
37       }, new Response.ErrorListener()
38       {
39           @Override
40           public void onErrorResponse(VolleyError error)
41           {
42           VolleyLog.d(TAG, "Error: " + error.getMessage());
43           hideProgressDialog();
44           }
45       })
46   {
47       @Override
48       public Map<String, String> getHeaders() throws AuthFailureError
49       {
50       HashMap<String, String> headers = new HashMap<String, String>();
51       headers.put("Content-Type", "application/json");
52       headers.put("x-access-token",((MainActivity) getActivity()).getToken());
53       return headers;
54       }
55   };
56   AppController.getInstance().addToRequestQueue(trackReq, TAG);
57     }
```

Η μέθοδος make_track_request ανακτεί όλες τις διαδρομές του χρήστη από την πλατφόρμα και γεμίζει τη λίστα.

Η μέθοδος make_track_request είναι πολύ ουσιώδης μέθοδος του fragment, καθώς φροντίζει να ανακτήσει τις διαδρομές του χρήστη από την πλατφόρμα. Αυτό, κατά τα γνωστά γίνεται μέσω HTTP αιτήματος που επιστρέφει πίνακα με JSON αντικείμενα, τα οποία περιέχουν τα στοιχεία των διαδρομών. Το συγκεκριμένο αίτημα απαιτεί εξουσιοδότηση, οπότε στις γραμμές 48-53 προσθέτουμε το JWT στις κεφαλίδες του αιτήματος.

Custom List Adapter

Για την εξασφάλιση όλης της λειτουργικότητας στην λίστα των διαδρομών, έχει δημιουργηθεί μια έκδοση του Interface BaseAdapter. Η συγκεκριμένη κλάση ορίζει πώς θα δομείται κάθε στοιχείο της λίστας ,σε επίπεδο εμφάνισης και λειτουργικότητας. Γενικότερα με τον BaseAdapter, μπορούμε να ορίσουμε στοιχεία λίστας με αυξημένη πολυπλοκότητα τόσο σε επίπεδο εμφανισιακής δομής, όσο και σε επίπεδο λειτουργιών.

```
1 public View getView(int position, View convertView, ViewGroup parent)
2     {
3   if (inflater == null)
4   {
5       inflater = (LayoutInflater) activi-
ty.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
6   }
7   if (convertView == null)
8   {
9       convertView = inflater.inflate(R.layout.list_row, null);
10  }
11  TextView date = (TextView) convertView.findViewById(R.id.date);
12  TextView time = (TextView) convertView.findViewById(R.id.time);
13  TextView distance = (TextView) convertView.findViewById(R.id.distance);
14  ImageButton delete = (ImageButton) convertView.findViewById(R.id.delete_btn);
15  delete.setOnClickListener(this);
16  delete.setTag(position);// set the tag for recognizing each row
17  // getting track data for the row
18  Track m = TrackItems.get(position);
19  date.setText("Date: " + m.getDate());
20  time.setText("Time: " + m.getTime() + "min");
21  distance.setText("Distance: " + m.getDistance());
22  return convertView;
23    }
```

Η μέθοδος getView ορίζει το πώς θα εμφανίζεται ένα στοιχείο της λίστας.

Κάθε έκδοση του BaseAdapter υλοποιεί την getView, η οποία ορίζει τι θα περιλαμβάνει και πώς θα εμφανίζεται ένα στοιχείο της λίστας. Για το σκοπό αυτό έχουμε ορίσει τη δομή της διεπαφής στο αρχείο list_row.xml. Στην γραμμή 9 την φορτώνουμε και στις γραμμές 14-21 ορίζουμε τα κείμενα που θα περιέχονται στο στοιχείο λίστας και θέτουμε onClickListener για το κουμπί διαγραφής, που υπάρχει στο κάθε στοιχείο. Το if στη γραμμή 7 υπάρχει για λόγους απόδοσης, καθώς το αντικείμενο convertView μπορεί να υφίσταται ήδη από τη δημιουργία προηγούμενου στοιχείου λίστας, οπότε και αποφεύγουμε το χρονοβόρο διάβασμα του XML αρχείου.

```
1  private void Make_delete_request(String id)
2  {
3
12   JsonObjectRequest jsonObjReq = new JsonObjectRequest(Method.DELETE,
13       URLS.URL_TRACKS, packet, new Response.Listener<JSONObject>()
14       {
15           @Override
16           public void onResponse(JSONObject response)
17           {
18           Log.v(TAG, "responce: " + response);
19           TrackItems.remove(hack_position);
20           notifyDataSetChanged();
21           }
22       }, new Response.ErrorListener()
23       {
24           @Override
25           public void onErrorResponse(VolleyError error)
26           {
27           VolleyLog.d(TAG, "Error: " + error.getMessage());
28           }
29       })
30   {
31       @Override
32       public Map<String, String> getHeaders() throws AuthFailureError
33       {
34       HashMap<String, String> headers = new HashMap<String, String>();
35       headers.put("Content-Type", "application/json");
36       headers.put("x-access-token", mToken);
37       headers.put("id", hack_id);
38       return headers;
39       }
40   };
41   // Adding request to request queue
42   AppController.getInstance().addToRequestQueue(jsonObjReq, TAG);
43   // Cancelling request
44   // ApplicationController.getInstance().getRequestQueue().cancelAll(tag_json_obj);
45   }
```

Η μέθοδος Make_delete_request καλείται όταν πατηθεί το κουμπί διαγραφής ενός στοιχείου και αναλαμβάνει την διαγραφή του από τη λίστα και από την πλατφόρμα.

Η Make_delete_request στέλνει το DELETE HTTP αίτημα στην πλατφόρμα. Όταν πατηθεί το κουμπί διαγραφής ενός στοιχείου της λίστας και ολοκληρωθεί με επιτυχία, διαγράφει και το στοιχείο από τη λίστα. Το αίτημα αυτό απαιτεί αυθεντικοποίηση, οπότε στις γραμμές 31-38 προσθέτουμε το JWT στην κεφαλίδα του αιτήματος. Επίσης, στην κεφαλίδα του αιτήματος προσθέτουμε το μοναδικό αναγνωριστικό της διαδρομής προς διαγραφή (δε το θέτουμε ως παράμετρο, καθώς από σχεδιαστική της επιλογή η βιβλιοθήκη Volley δεν δέχεται παραμέτρους στα DELETE αιτήματα).

ProfileFragment

Αυτό το Fragment έχει ως ρόλο την προβολή του προφίλ του χρήστη και την τροποποίησή του. Αποτελείται από μια απλή διεπαφή λίγων πεδίων κειμένου .Το μόνο κομμάτι κώδικα που θα είχε αξία να παρουσιάσουμε είναι αυτό που αφορά το PUT αίτημα στην πλατφόρμα για την αλλαγή του προφίλ.

```
1 // edit profile request
2 private void Make_edit_profile_request(String old_password,String new_password)
3 {
4   JSONObject credentials = null;
5   try
6   {
7       credentials = new JSONObject();
8       credentials.put("password", old_password);
9       credentials.put("username", mUsername.getText());
10      credentials.put("password_r", new_password);
11      credentials.put("name", mName.getText());
12      credentials.put("surname", mSurname.getText());
13
14  } catch (JSONException e)
15  {
16      e.printStackTrace();
17  }
18  JsonObjectRequest jsonObjReq = new JsonObjectRequest(Method.PUT,
19      URLS.URL_PROFILE, credentials,
20      new Response.Listener<JSONObject>()
21      {
22          @Override
23          public void onResponse(JSONObject response)
24          {
25          try
26          {
27
Toast.makeText(getActivity(),response.getString("success"),Toast.LENGTH_LONG).show();
28          } catch (JSONException e)
29          {
30              try
31              {
32
Toast.makeText(getActivity(),response.getString("error"),Toast.LENGTH_LONG).show();
33              } catch (JSONException e1)
34              {
35              // TODO Auto-generated catch block
36              e1.printStackTrace();
37              }
38              e.printStackTrace();
39          }
40          }
41      }, new Response.ErrorListener()
42      {
43          @Override
44          public void onErrorResponse(VolleyError error)
45          {
46          VolleyLog.d(TAG, "Error: " + error.getMessage());
47          }
48      })
49  {
50      @Override
51      public Map<String, String> getHeaders() throws AuthFailureError
52      {
53      HashMap<String, String> headers = new HashMap<String, String>();
54      headers.put("Content-Type", "application/json");
55      headers.put("x-access-token",((MainActivity) getActivity()).getToken());
56      return headers;
57      }
58  };
59  // Adding request to request queue
60  AppController.getInstance().addToRequestQueue(jsonObjReq, TAG);
61 }
```

Στην Make_edit_profile_request στέλνεται στην πλατφόρμα ένα PUT αίτημα για την αλλαγή των στοιχείων.

Σύμφωνα με το REST πρότυπο, τα αιτήματα PUT αφορούν την επεξεργασία υπαρχόντων πόρων. Έτσι, για την επεξεργασία του προφίλ χρήστη χρησιμοποιούμε αίτημα PUT με παραμέτρους τα νέα στοιχεία που έχει δώσει ο χρήστης (οι γραμμές 7-12 αφορούν τις παραμέτρους). Ως συνήθως, στέλνουμε το JWT με την κεφαλίδα του αιτήματος (γρ. 51-57), καθώς το αίτημα απαιτεί εξουσιοδότηση.

6.1.3 TrackingService

Η TrackingService αποτελεί ένα από τα σημαντικότερα κομμάτια της εφαρμογής, καθώς είναι μία υπηρεσία που τρέχει το παρασκήνιο και καταγράφει σε πραγματικό χρόνο όλα τα αθλητικά δεδομένα (εφόσον ο χρήστης έχει εκκινήσει καταγραφή νέας δραστηριότητας). Στα δεδομένα που έχει καταγράψει ή υπηρεσία μπορούν να έχουν πρόσβαση όλα τα fragments που βρίσκονται κάτω από την MainActivity.Η κλάση της υπηρεσίας αυτής υλοποιεί τον LocationListener των Google services. Αυτό σημαίνει πως μπορεί να παρακολουθεί τις αλλαγές τοποθεσίας μέσω της μεθόδου onLocationChanged.

```
1 public void setStarted(boolean started)
2 {
3   if (mStarted != started)
4   {
5       mStarted = started;
6       if (started)
7       {
8       // start listening to location changes
9       mCoursePoints = new ArrayList<LatLng>();
10      LocationRequest mLocationRequest = LocationRequest.create();
11      mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
12      mLocationRequest.setInterval(500);// check for location change every 0.5 sec
13      mLocationRequest.setSmallestDisplacement(5);// 5 meters min displacement
14      mLocationClient.requestLocationUpdates(mLocationRequest, this);
15      }
16  }
17}
```

Η μέθοδος setStarted θέτει τις παραμέτρους και ξεκινάει την παρακολούθηση για αλλαγές τοποθεσίας.

Η μέθοδος setStarted είναι δημόσια (αυτό σημαίνει πως μπορεί να κληθεί και εκτός της κλάσης) και συνήθως καλείται από το RunFragment. Ο ρόλος της είναι να εκκινεί την παρακολούθηση αλλαγών από την υπηρεσία, αφού πρώτα ρυθμίσει κάποιες παραμέτρους. Συγκεκριμένα:

- Στην γραμμή 9 δημιουργούμε μία λίστα με γεωγραφικά μήκη-πλάτη, που θα κρατήσει όλες τις θέσεις του χρήστη.
- Στη γραμμή 10 φτιάχνουμε το αντικείμενο LocationRequest, που κρατάει τις παραμέτρους.
- Στην γραμμή 11 θέτουμε την προτεραιότητα και την ακρίβεια στο μέγιστο.
- Στην γραμμή 12 θέτουμε το χρονικό διάστημα μεταξύ των ελέγχων αλλαγής τοποθεσίας σε 0,5 sec.
- Στη γραμμή 13 θέτουμε το όριο για να θεωρηθεί αλλαγή τοποθεσίας στα 5 μέτρα.
- Τελικά στην γραμμή 15, ξεκινάμε να παρακολουθούμε για αλλαγές τοποθεσίας με τις παραπάνω παραμέτρους και LocationListener την ίδια την κλάση.

Εφόσον έχουν ξεκινήσει οι ενημερώσεις τοποθεσίας, σε κάθε αλλαγή τοποθεσίας καλείται η μέθοδος onLocationChanged αυτόματα.

```
1  public void onLocationChanged(Location arg0)
2      {
3        // add the next point to list and broadcast the message
4        if (mStarted)
5        {
6            // add the new point to the list
7            if (!mCoursePoints.isEmpty())
8            {
9            Location prev_point = new Location("point A"); // previous location
10           prev_point.setLatitude(get_last_point(mCoursePoints).latitude);
11           prev_point.setLongitude(get_last_point(mCoursePoints).longitude);
12           // sometimes when we lose gps signal the first from second
13           // location are 3000km apart or more
14           // so we check this in order to be more precise
15           if (arg0.distanceTo(prev_point) < 15)
16           {
17               mDistance += arg0.distanceTo(prev_point);
18           }
19           } else
20           {
21               mDistance += 0;
22           }
23           mCoursePoints.add(new LatLng(arg0.getLatitude(), arg0.getLongitude()));
24           mLastPoint = new LatLng(arg0.getLatitude(), arg0.getLongitude());
25           // broadcast message
26           sendlocation();
27       }
28   }
```

Η onLocationChanged καλείται κάθε φορά που έχουμε αλλαγή θέσης με παράμετρο την τρέχουσα θέση και φροντίζει για την ενημέρωση του ίχνους του χρήστη και άλλων παραμέτρων.

Η onLocationChanged έχει ως παράμετρο την τρέχουσα τοποθεσία του χρήστη. Κάθε φορά που καλείται διεκπεραιώνει τις εξής λειτουργίες :

- Προσθήκη της τρέχουσας τοποθεσίας στην λίστα με τη διαδρομή του χρήστη (γρ. 23).
- Ενημέρωση της απόστασης που έχει διανυθεί (γρ 17) *.
- Αποστολή Broadcast ενημέρωσης αλλαγής τοποθεσίας, ώστε ο χάρτης του RunFragment να ενημερώνεται σε πραγματικό χρόνο με τη μέθοδο sendlocation.

```
1  private void sendlocation()
2{
3    // notify for location change anyone who listens
4    Intent intent = new Intent("location_changed");
5    LocalBroadcastManager.getInstance(this).sendBroadcast(intent);
6}
```

Η sendloacation στέλνει broadcast σε κάθε αλλαγή τοποθεσίας.

*Το if στην γραμμή 15 φροντίζει να μη μετράει στην απόσταση διαφορές τελευταίου- προτελευταίου σημείου μεγαλύτερες από 15 μέτρα ,καθώς τέτοιες διαφορές προκύπτουν από σφάλματα των αισθητήρων.

6.2 Web πλατφόρμα

6.2.1 Εισαγωγικά

Για την ανάπτυξη της web εφαρμογής χρησιμοποιήθηκε η πλατφόρμα Node.js σε συνδυασμό με το σύστημα βάσεων δεδομένων MongoDB.
Το Node.js είναι μία δωρεάν, ανοιχτού κώδικα πλατφόρμα για την ανάπτυξη δικτυακών εφαρμογών (σε γλώσσα JavaScript).Το Node.js τρέχει στην V8 JavaScript engine της Google, που είναι ταχύτατη.
Η πλατφόρμα διαθέτει ορισμένα χαρακτηριστικά που την διαφοροποιούν από ανταγωνιστικές λύσεις(π.χ. PHP) και την καθιστούν ελκυστική επιλογή για δημιουργία web εφαρμογών. Συγκεκριμένα:

- Το Node.js δε χρησιμοποιεί πολυνηματικότητα (multithreading) για τη διαχείριση πολλών συνδέσεων καθώς στηρίζεται σε ένα νήμα εκτέλεσης με δυνατότητα *ασύγχρονης* εισόδου/εξόδου(*non-blocking I/O*). Με χρήση ασύγχρονης εισόδου/εξόδου κάθε αίτημα πελάτη δεν μπλοκάρει το νήμα και δε είναι απαραίτητη η χρήση πολλών νημάτων για υποστήριξη πολλών χρηστών.

  Ο τρόπος που το Node.js επιτυγχάνει ασύγχρονη είσοδο/ έξοδο είναι μέσω των ***callback functions.*** *Για κάθε λειτουργία π.χ. διάβασμα/γράψιμο από αρχείο ή βάση δεδομένων ορίζουμε μια callback function, η οποία θα κληθεί όταν τελειώσει η εν λόγω λειτουργία. Έτσι το σύστημα εκκινεί τη λειτουργία και συνεχίζει στις επόμενες (που μάλιστα μπορεί να προέρχονται από άλλο χρήστη) ,όταν η διαδικασία αυτή ολοκληρωθεί καλεί την callback function.*

  Για τις βασικές λειτουργίες των εφαρμογών (π.χ. διάβασμα αρχείων, κρυπτογραφία HTTP) το Node.js διαθέτει ένα πλούσιο σετ βιβλιοθηκών με κομψή και απλή διεπαφή. Για άλλες πάλι λειτουργίες η κοινότητα του Node.js έχει δημιουργήσει ένα ολόκληρο οικοσύστημα με βιβλιοθήκες (*node package manager NPM*) ,οι οποίες καλύπτουν σχεδόν κάθε λειτουργία που μπορεί να χρειαστεί.

Στην εφαρμογή μας χρησιμοποιούμε την βιβλιοθήκη Express, που διευκολύνει πολύ την δημιουργία REST APIs σε σχέση με την ενσωματωμένη HTTP βιβλιοθήκη του Node.js.
Η MongoDB αποτελεί ένα δωρεάν, ανοιχτού κώδικα σύστημα διαχείρισης βάσεων δεδομένων, που ανήκει στις λεγόμενες noSQL βάσεις δεδομένων. Αυτό σημαίνει ότι δε χρησιμοποιεί την σχεσιακή δομή των παραδοσιακών βάσεων δεδομένων, που βασίζονται σε πίνακες και σχέσεις ανάμεσα σε αυτούς, ούτε και την γλώσσα SQL για την εκτέλεση ενεργειών στη βάση δεδομένων. Η αποθήκευση των δεδομένων χρησιμοποιεί μία αναπαράσταση που ονομάζεται BSON , η BSON (στα πλαίσια που εξετάζουμε) είναι ίδια με το γνωστό JSON.Η χρήση αυτής της αναπαράστασης BSON διευκολύνει την χρήση της MongoDB σε εφαρμογές που χρησιμοποιούν JSON, καθώς δεν απαιτούνται καθόλου μετατροπές για αποθήκευση και ανάκτηση από τη βάση.
Η MongoDB είναι η πιο διαδεδομένη noSQL βάση δεδομένων και διαθέτει βιβλιοθήκες για χρήση σε όλες τις γνωστές γλώσσες προγραμματισμού. Στο Node.js εκτός από την απλή βιβλιοθήκη που μας δίνει η εταιρία που αναπτύσσει την MongoDB και έχει τις βασικές δυνατότητες ,υπάρχουν πολύ εξελιγμένες βιβλιοθήκες με προχωρημένες δυνατότητες. Στην

εφαρμογή μας χρησιμοποιούμε την βιβλιοθήκη mongoskin, μία αρκετά ισχυρή βιβλιοθήκη που κάνει τις εργασίες στην βάση υπόθεση λίγων γραμμών κώδικα.

6.2.2 Κώδικας

App.js

Το αρχείο app.js αποτελεί το κεντρικό αρχείο της εφαρμογής. Σε αυτό γίνονται όλες οι λειτουργίες φόρτωσης βιβλιοθηκών, το άνοιγμα της σύνδεσης με τη βάση δεδομένων και ο ορισμός των συναρτήσεων χειρισμού για κάθε URL και κάθε HTTP μέθοδο.
Παρακάτω παρουσιάζουμε το αρχείο app.js κομμάτι -κομμάτι μαζί με σύντομη επεξήγηση.

```
1   //import libraries
2   var express = require('express'),
3       jwt = require('jwt-simple'),
4       moment = require('moment'),
5       routes = require('./routes'),
6       http = require('http'),
7       path = require('path'),
8       mongoskin = require('mongoskin'),
```

Οι πρώτες γραμμές αφορούν τη φόρτωση βιβλιοθηκών και κώδικα. Συγκεκριμένα, φορτώνουμε:

- Στην γραμμή 2 το Express.js,
- Στην 3 τη βιβλιοθήκη που χειρίζεται τις λειτουργίες των JWTs,
- στην 4 το moment.js ,μία βιβλιοθήκη για χειρισμό ημερομηνιών και χρονικών στιγμών γενικά
- Στην 5 το φάκελο με τα υπόλοιπα αρχεία κώδικα της εφαρμογής,
- στην 6 τη βασική βιβλιοθήκη για χειρισμό HTTP αιτημάτων (είναι απαραίτητη για τη λειτουργία του Express),
- Στην 7 το path, μία βιβλιοθήκη για λειτουργίες σε μονοπάτια φακέλων,
- Στην 8 την mongoskin, που είναι η βιβλιοθήκη που χρησιμοποιούμε για πρόσβαση στη βάση δεδομένων.

```
1    db = mongoskin.db(dbUrl,
2      {
3          safe: true
4      }),
5      collections = {
6          tracks: db.collection('tracks'),
7          users: db.collection('users')
8      };
9    // Express.js Middleware
10   var session = require('express-session'),
11     logger = require('morgan'),
12     errorHandler = require('errorhandler'),
13     cookieParser = require('cookie-parser'),
14     bodyParser = require('body-parser'),
15     methodOverride = require('method-override');
16   var app = express();
17   // Express.js configurations
18   // Express.js middleware configuration
19   // Expose collections to request handlers
20   app.use(function(req, res, next)
21   {
22       if (!collections.tracks || !collections.users) return next(new Error('No collec-
tions.'))
23       req.collections = collections;
24       return next();
25   });
26   app.use(logger('dev'));
27   app.use(bodyParser.json());
28   app.use(bodyParser.urlencoded());
29   app.use(methodOverride());
30   app.use(require('stylus').middleware(__dirname + '/public'));
31   app.use(express.static(path.join(__dirname, 'public')));
32   app.use(cookieParser('3CCC4ACD-6ED1-4844-9217-82131BDCB239'));
33   app.use(session(
34      {
35          secret: '2C44774A-D649-4D44-9535-46E296EF984F'
36      }))
37   if ('development' == app.get('env'))
38   {
39      app.use(errorHandler());
40   }
```

Οι επόμενες γραμμές αφορούν τη σύνδεση με τη βάση δεδομένων και τη ρύθμιση παραμέτρων του Express.Συγκεκριμένα, στις γραμμές 1-8 συνδεόμαστε με τη βάση και στη συνέχεια εφόσον υπάρχουν δύο συλλογές με όνομα users και tracks, τις αποθηκεύουμε στο αντικείμενο collections.Οι γραμμές 9-15 αφορούν τη φόρτωση βιβλιοθηκών, που χρησιμοποιούνται από το Express για διάφορες λειτουργίες. Στο αντικείμενο app ορίζουμε ιδιότητες που καθορίζουν τη συμπεριφορά της εφαρμογής όπως θα δούμε παρακάτω. Οι επόμενες γραμμές (20-41) χρησιμοποιούν την συνάρτηση use του αντικειμένου app. συνάρτηση αυτή χρησιμοποιείται με την εξής μορφή:

```
app.use([URL,] function [, function...])
```

Η σημασία της παραπάνω κλήσης είναι η εξής: Εφόσον ο πελάτης πραγματοποιήσει ένα αίτημα στο URL, τότε θα τρέξει η συνάρτηση αν το URL απουσιάζει, τότε η συνάρτηση θα τρέξει για όλα τα URLs (οι συναρτήσεις αυτές ονομάζονται middleware). Η εξυπηρέτηση ενός HTTP αιτήματος στο *Express* συνοψίζεται στο πέρασμά του από μία ή και περισσότερες συναρτήσεις middleware με την τελευταία να στέλνει την απόκριση στον πελάτη, αξίζει να αναφερθεί πως η εκτέλεση των middlewares γίνεται με τη σειρά που συναντώνται στον κώδικα, άρα η σειρά με την οποία γράφονται πολλές φορές είναι σημαντική.

Τα middlewares που χρησιμοποιούμε στην περίπτωση μας είναι τα εξής.

- Στις γραμμές 20-25 περνάμε ένα middleware που σε κάθε αίτημα βάζει τις δυο συλλογές της βάσεως δεδομένων στο αντικείμενο res ,ώστε να είναι ορατές από τα επόμενα middlewares που θα χειριστούν το αίτημα. Κάτι τέτοιο αποτελεί κοινή πρακτική για σε τέτοιου είδους εφαρμογές Node.js.

Στις γραμμές 26-36 έχουμε middlewares τρίτων κατασκευαστών που επιτελεί κάποιες βασικές λειτουργίες για παράδειγμα έχουμε .

- Logger middleware για την εμφάνιση στην κονσόλα μηνυμάτων αποσφαλμάτωσης
- Body parser JSON για διάβασμα JSON παραμέτρων.
- Body parser urlencoded για διάβασμα παραμέτρων κωδικοποιημένων στο url διεύθυνσης
- Session για δημιουργία sessions χρηστών κτλ.

```
1  // web guest routes
2  app.get('/logout', routes.logout);
3  app.get('/login', routes.login);
4  app.get('/register', routes.register);
5  //mobile and web login paths
6  app.post('/login', routes.auth.settoken, routes.weblogin);
7  app.post('/mlogin',routes.auth.settoken,routes.mobilelogin);
8  //api routes
9  app.get('/api/profile', routes.auth.validateuser, routes.api.get_profile); //get a profile
10 app.post('/api/profile', routes.api.new_profile); //new profile
11 app.put('/api/profile', routes.auth.validateuser, routes.api.edit_profile); //edit profile
12 app.del('/api/profile', routes.auth.validateuser, routes.api.del_profile); //delete  pro-
file
13 app.get('/api/profile/tracks', routes.auth.validateuser, routes.auth.validateuser,
routes.api.get_tracks); //get users tracks
14 app.post('/api/profile/tracks', routes.auth.validateuser, routes.api.new_track); //new
track
15 app.del('/api/profile/tracks', routes.auth.validateuser, routes.api.del_track); //delete a
track
16 app.get('/api/profile/tracks_p/:page/:size',routes.api.get_paged_tracks);
```

Οι επόμενες γραμμές ορίζουν τα τελικά middlewares χειρισμού των αιτημάτων για κάθε μέθοδο HTTP και κάθε URL . Εκτός από τα URLs του REST API (γρ. 9-16)έχουμε και κάποια URLs για την συνοδευτική ιστοσελίδα (γρ. 3-7).Ο ορισμός middleware για συγκεκριμένη μέθοδο HTTP γίνεται με τον εξής κώδικα.

```
app.HTTP_METHOD([URL,] ,middleware1,middleware2,middleware3, …)
```

Όπως βλέπουμε πως μπορούμε να ορίσουμε παραπάνω από ένα middleware για ένα URL ,τα middlewares θα εκτελεστούν με τη σειρά που δηλώνονται .Τα URLs που χρησιμοποιούμε στο REST API είναι δυο ,το */api/profile* που αφορά ενέργειες σε προφίλ χρήστη(προσθήκη, τροποποίηση, ανάκτηση) και το *api/profile/tracks* που αφορά σε ενέργειες πάνω σε διαδρομές χρήστη (προσθήκη, διαγραφή, ανάκτηση).Όλα τα URLs του API περνάνε από δυο middlewares, το πρώτο πάντα είναι αυτό που αναλαμβάνει την ταυτοποιήση του χρήστη μέσω του JWT και το δεύτερο αναλαμβάνει την απάντηση του (εκτός από το POST στο URL api/profile που δημιουργεί νέο προφίλ οπότε για προφανείς λόγους δε χρειάζεται ταυτοποίηση) . Οι περισσότερες ενέργειες στο API γίνονται μόνο από εξουσιοδοτημένο χρήστη.

Auth.js

Το αρχείο Auth περιέχει δυο middlewares που αναλαμβάνουν την αυθεντικοποίηση και εξουσιοδότηση χρήστη .

```
1 exports.settoken = function(req, res, next)
2 {
3     if (!req.body.username || !req.body.password)
4     {
5         req.error = "nopass";
6         return next();
7     }
8     req.collections.users.findOne(
9     {
10        username: req.body.username,
11    }, function(error, user)
12    {
13        if (error)
14        {
15            // send database error
16            return next(error);
17        }
18        if (user == null)
19        {
20            req.error = "No user ";
21        }
22        else
23        {
24            //check if password is corect
25            if(user.psw==crypto.createHmac('sha256', user.salt).update(req.body.pass-
word).digest('hex'))
26            {
27                console.log("authorized");
28                var expires = moment().add(600,'minutes').valueOf();
29                var token = jwt.encode(
30                {
31                    username: user.username,
32                    exp: expires
33                }, 'tsakatsoukou');
34                req.token = token;
35            }
36            else
37            {
38                req.error = "Wrong password ";
39            }
40        }
41        return next();
42    })
43};
```

Το middleware settotken αναλαμβάνει την ταυτοποίησή του χρήστη και την δημιουργία του JWT

Το middleware settoken έχει ως ρόλο τη δημιουργία του JWT αφού ταυτοποιήσει τα στοιχεία του χρήστη. Ο αλγόριθμος δημιουργίας JWT που εκτελείται είναι ο εξής:

1. Βρίσκουμε τα στοιχεία του χρήστη στις παραμέτρους του αιτήματος αν δεν υπάρχουν προσθέτουμε το ανάλογο μήνυμα λάθους στο αντικείμενο του αιτήματος.
2. Αλλιώς ψάχνουμε στη βάση δεδομένων χρήστη με αυτό το όνομα, αν δεν υπάρχει προσθέτουμε το ανάλογο μήνυμα λάθους στο αντικείμενο του αιτήματος (γρ 8-21)
3. Αν βρεθεί το όνομα χρήστη τότε ελέγχουμε αν οι κρυπτογραφημένος κωδικός στη βάση

για αυτό το χρήστη είναι ίδιος με τον κωδικό που δόθηκε (αφού τον κρυπτογραφήσουμε με κλειδί που είναι μοναδικό για κάθε χρήστη) αλλιώς προσθέτουμε το ανάλογο μήνυμα λάθους στο αντικείμενο του αιτήματος (γρ 37-39 και το if στην 25).

4. Αν οι κωδικοί είναι ίδιοι τότε δημιουργούμε το JWT με ημερομηνία λήξης μετά από 10 ώρες, το κρυπτογραφούμε και το προσθέτουμε στο αντικείμενο του αιτήματος (γρ 27-34).
5. Καλούμε την next και ο έλεγχος πάει στο επόμενο middleware χειρισμού του αιτήματος που συνήθως η δουλεία του είναι είτε να εμφανίσει το λάθος είτε να στείλει το κρυπτογραφημένο JWT στον πελάτη.

```
1 exports.validateuser = function(req, res, next)
2 {
3     var token = (req.body && req.body.token) || (req.query && req.query.token) ||
req.headers['x-access-token']||(req.session.token);
4         console.log("validateuser");
5     if (!token)
6     {
7         req.error = "Unauthorized";
8         return next();
9     }
10    else
11    {
12        //decode token
13        var decoded = jwt.decode(token, 'tsakatsoukou');
14        console.log("decoded",decoded);
15        console.log("encoded",token);
16        if (decoded.exp <= Date.now())
17        {
18            req.error = "expired";
19            return next();
20        }
21      //search database
22        else
23        {
24            req.collections.users.findOne(
25            {
26                username: decoded.username,
27            }, function(error, user)
28            {
29                if (error)
30                {
31                    return next(error);
32                }
33                if (user == null)
34                {
35                    //no user with such id (fake token)
36                    req.error = "no_user";
37                    return next();
38                }
39                else
40                {
41                    //hide password
42                    user.psw="";
43                    req.user=user;
44                    return next();
45
46                }
47            })
48        }
49    }
50}
```

Το middleware validateuser ελέγχει το JWT για την εγκυρότητά του

Το middleware validateuser χρησιμοποιεί το JWT που έχει σταλεί ως παράμετρος για να βρει το χρήστη που αντιστοιχεί σε αυτόν. Για να το κάνει αυτό ακολουθεί τον παρακάτω αλγόριθμο

1. Έλεγχος ύπαρξης του JWT σε όλα τα προβλεπόμενα σημεία (κεφαλίδα αιτήματος ,παράμετροι αιτήματος και session cookies) αν δε υπάρχει πουθενά προσθέτουμε το ανάλογο μήνυμα λάθους στο αντικείμενο του αιτήματος(γρ 3-9) .
2. Αλλιώς κάνουμε αποκρυπτογράφηση του JWT με γνωστό κλειδί (γρ.13).
3. Αν η ημερομηνία λήξης του JWT είναι πρότερη της τωρινής, προσθέτουμε το ανάλογο μήνυμα λάθους στο αντικείμενο του αιτήματος και καλούμε την next (γρ. 16-20) (ώστε ό έλεγχος να περάσει στο επόμενο middleware)
4. Αλλιώς εκτελούμε αναζήτηση στη βάση για το όνομα χρήστη JWT (γρ 24-27),σε περίπτωση που το όνομα δε βρεθεί προσθέτουμε το ανάλογο μήνυμα λάθους στο αντικείμενο του αιτήματος και καλούμε την next (γρ.39-46).
5. Αλλιώς προσθέτουμε το όνομα χρήστη το αντικείμενο του αιτήματος και καλούμε τη next (γρ. 40-45).

Api.js

Στο αρχείο api.js περιέχονται όλα τα middlewares που στέλνουν την τελική απάντηση στα αιτήματα προς το REST API,παρακάτω τα αναφέρουμε συνοπτικά . Η αναλυτική παρουσίαση του κώδικα δε κρίνεται σκόπιμη καθώς δεν περιέχει κάποιο κομμάτι άξιο αναφοράς .

Για το URL api/profile έχουμε

HTTP μέθοδος	*Ενέργεια*	*Middleware*	*Σχόλια*
GET	Ανάκτηση στοιχείων προφίλ χρήστη	get_profile	Απαιτεί αποστολή JWT με το αίτημα
POST	Δημιουργία προφίλ χρήστη	new_profile	**Δεν** απαιτεί αποστολή JWT με το αίτημα
PUT	Επεξεργασία υπάρχοντος προφίλ	edit_profile	Απαιτεί αποστολή JWT με το αίτημα .
DELETE	Διαγραφή προφίλ χρήστη	Κανένα	Δεν έχει υλοποιηθεί .

6 Middlewares για το χειρισμό προφίλ χρηστών

Για το URL api/profile/tracks έχουμε

HTTP μέθοδος	*Ενέργεια*	*Middleware*	*Σχόλια*
GET	Ανάκτηση όλων των διαδρομών του χρήστη	get_tracks	Απαιτεί αποστολή JWT με το αίτημα (στις παραμέτρους).
POST	Δημιουργία νέας διαδρομής για το χρήστη	new_track	Απαιτεί αποστολή JWT με το αίτημα
PUT	Αλλαγή υπάρχουσας διαδρομής	Κανένα	Δεν έχει υλοποιηθεί .
DELETE	Διαγραφή διαδρομής	del_track	Απαιτεί αποστολή JWT με το αίτημα .

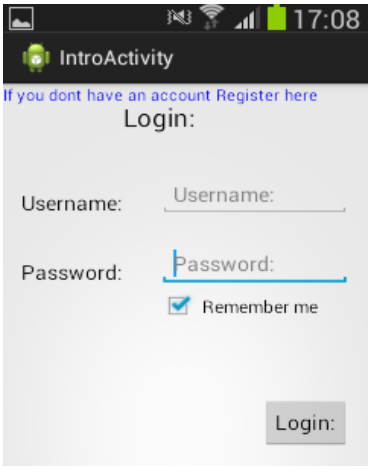
7 Middlewares χειρισμού διαδρομών (tracks)

7. Παρουσιαση της εφαρμογής από τη σκοπιά του χρήστη

7.1 Mobile εφαρμογή

Η εφαρμογή αποτελείται από δύο βασικά κομμάτια το πρώτο για είσοδο χρήστη ή εγγραφή νέου χρήστη και το κύριο κομμάτι της εφαρμογής. Αφού ο χρήστης εγγράφει μπορεί να μεταβεί στην κυρία εφαρμογή.

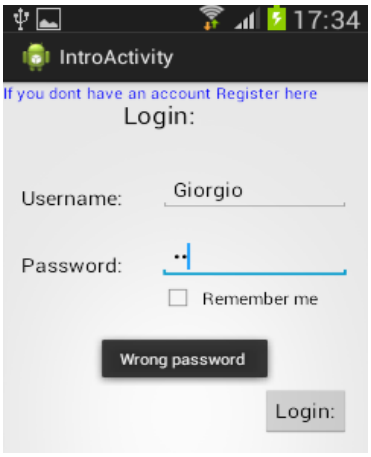
7.1.1 Οθόνη εισόδου



48 Οθόνη εισόδου χωρίς συμπληρωμένα στοιχεία

Στην οθόνη αυτή ο χρήστης εισάγει τα στοιχεία του για να εισέλθει στην υπηρεσία , αν τσεκάρει το "remember me" η εφαρμογή θυμάται τα στοιχειά και δε χρειάζεται να τα επανεισάγει όταν ξανανοίξει την εφαρμογή.
Όταν ο χρήστης πατήσει το κουμπί Login (από τη στιγμή που θα πατηθεί απαιτείται σύνδεση στο διαδίκτυο) αν τα στοιχεία του επιβεβαιωθούν ανοίγει η κυρία οθόνη της εφαρμογής, σε αντίθετη περίπτωση του εμφανίζει το ανάλογο μήνυμα λάθους. .

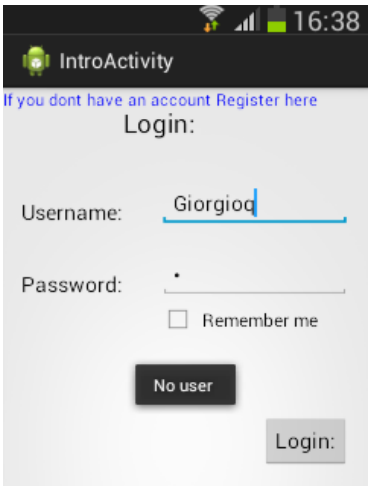
Μήνυμα λάθος κωδικού
Αυτό το μήνυμα εμφανίζεται όταν υπάρχει ο χρήστης, αλλά δόθηκε λάθος κωδικός.



49 Οθόνη εισόδου με συμπληρωμένα στοιχεία και μήνυμα λάθος κωδικού

Μήνυμα λάθος όνομα χρήστη

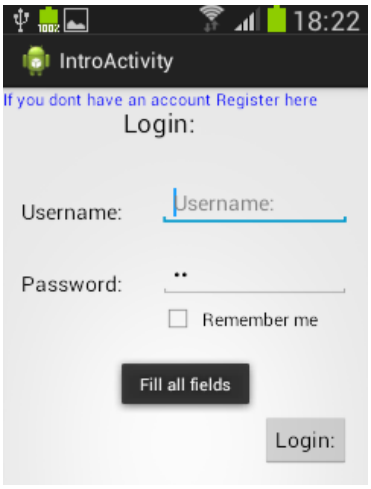
Το συγκεκριμένο όνομα χρήστη δεν υπάρχει.



50 Οθόνη εισόδου με συμπληρωμένα στοιχεία και μήνυμα λάθος όνομα χρήστη

Μήνυμα ότι δεν έχουν συμπληρωθεί όλα τα στοιχεία

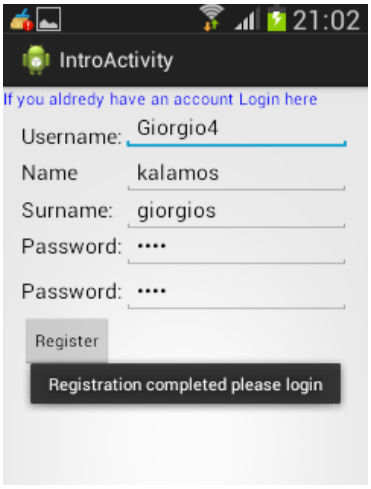
Όταν λείπει το όνομα χρήστη ή ο κωδικός τότε εμφανίζεται το παρακάτω μήνυμα:



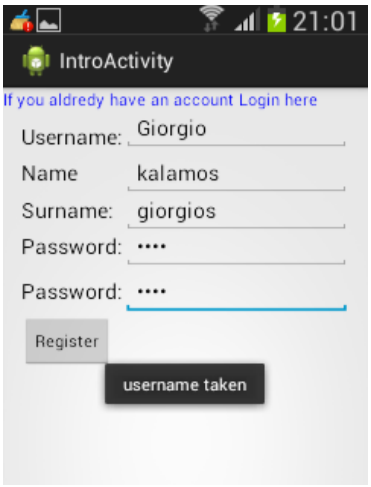
51 Οθόνη εισόδου με συμπληρωμένα στοιχεία και μήνυμα ότι λείπουν στοιχεία

7.1.2 Οθόνη εγγραφής

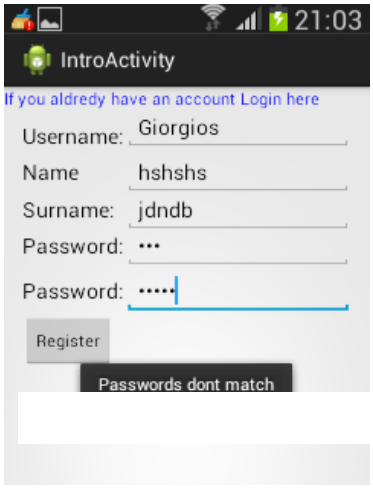
Εδώ ο χρήστης εισάγει τα δεδομένα του και κάνει εγγραφή. Εάν η εγράφη ήταν επιτυχής του εμφανίζεται μήνυμα να συνδεθεί, αλλιώς του εμφανίζεται το κατάλληλο μήνυμα λάθους.



52 Μήνυμα επιτυχούς εγράφης



53 Μήνυμα ότι το όνομα χρήστη είναι ήδη σε χρήση από άλλο χρήστη



54 Μήνυμα ότι η επιβεβαίωση κωδικού απέτυχε

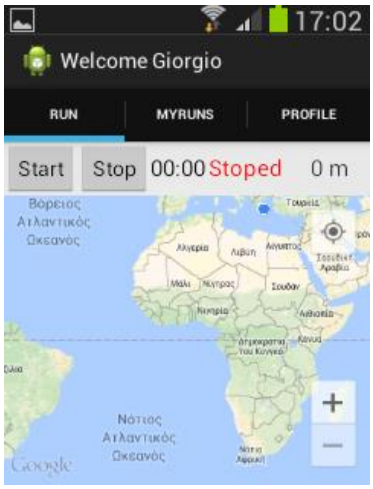
7.1.3 Κεντρική εφαρμογή

Η κεντρική εφαρμογή αποτελείται από τρεις καρτέλες, που περιγράφονται αναλυτικά παρακάτω.

7.2.1 Καρτέλα έναρξης νέας δραστηριότητας

Σε αυτήν την καρτέλα ο χρήστης μπορεί να ξεκινήσει καταγραφή νέας διαδρομής. Εφόσον ο χρήστης τελειώσει τη δραστηριότητα μπορεί να αποθηκεύσει όλα τα δεδομένα της.
Παρακάτω παρουσιάζονται αναλυτικά όλες οι δυνατές καταστάσεις αυτής της καρτέλας.

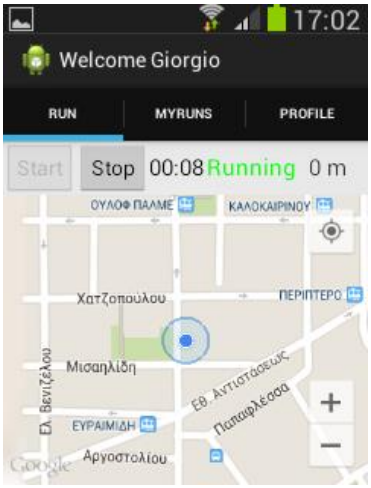
Αναμονή για έναρξη δραστηριότητας (πατώντας το κουμπί start)
Η εφαρμογή αναμένει από το χρήστη το πάτημα του κουμπιού Start για εκκίνηση καταγραφής .



55 Οθόνη έναρξης νέας δραστηριότητας σε κατάσταση αναμονής

Δραστηριότητα υπό καταγραφή

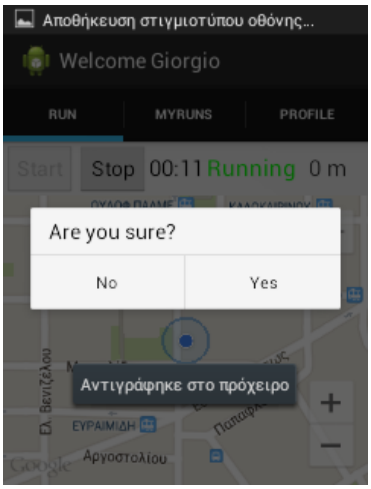
Κατά τη διάρκεια καταγραφής νέας δραστηριότητας όλα τα στοιχεία ενημερώνονται σε πραγματικό χρόνο.



56 Οθόνη έναρξης νέας δραστηριότητας σε κατάσταση αναμονής

Διάλογος επιβεβαίωσης για το σταμάτημα της δραστηριότητας

Εμφανίζεται εφόσον πατηθεί το κουμπί Stop αν ο χρήστης πατήσει ΝΟ η καταγραφή συνεχίζεται αλλιώς η δραστηριότητα αποθηκεύεται και το σύστημα μπαίνει σε αναμονή.

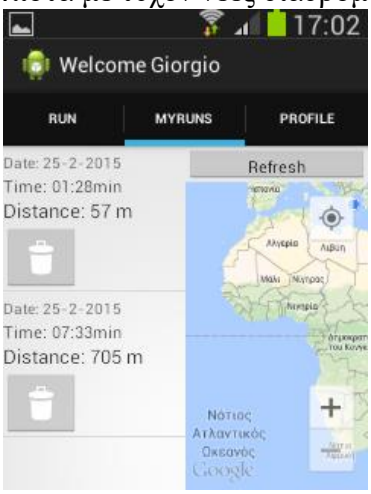


57 Οθόνη έναρξης νέας δραστηριότητας με διάλογο επιβεβαίωσης σταματήματος καταγραφής

7.2.2 Καρτέλα για την προβολή και διαγραφή νέων δραστηριοτήτων

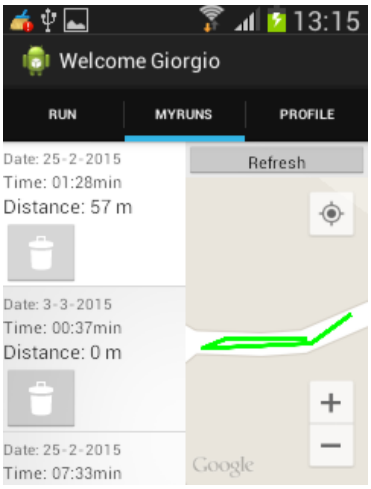
Σε αυτή την καρτέλα έχουμε μια λίστα με όλες τις δραστηριότητες που έχουν αποθηκευτεί μαζί με όλα τα σχετικά δεδομένα τους .Παρακάτω παρουσιάζονται χαρακτηριστικές οθόνες από αυτή την καρτέλα.

Η οθόνη της καρτέλας χωρίς να έχει επιλεγεί κάποια διαδρομή , στα αριστερά βλέπουμε μια λίστα με όλες τις δραστηριότητες ενώ στα δεξιά έχουμε το κουμπί refresh που ανανεώνει τη λίστα με τυχόν νέες διαδρομές και τον χάρτη που εμφανίζει το ίχνος της επιλεγμένης διαδρομής..



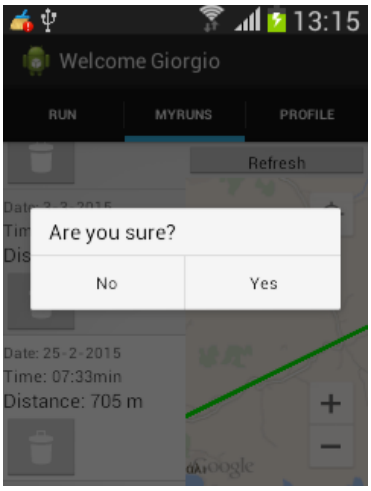
58 Οθόνη προβολής αποθηκευμένων δραστηριοτήτων χωρίς κάποια δραστηριότητα επιλεγμένη

Αν επιλεγεί ένα στοιχείο της λίστας, τότε στο χάρτη μας εμφανίζεται το ίχνος της διαδρομής που πραγματοποιήθηκε κατά τη διάρκεια της δραστηριότητας , επίσης σε κάθε στοιχείο της λίστας υπάρχει και ένα κουμπί διαγραφής που το διαγράφει.



59 Οθόνη προβολής αποθηκευμένων δραστηριοτήτων με δραστηριότητα επιλεγμένη

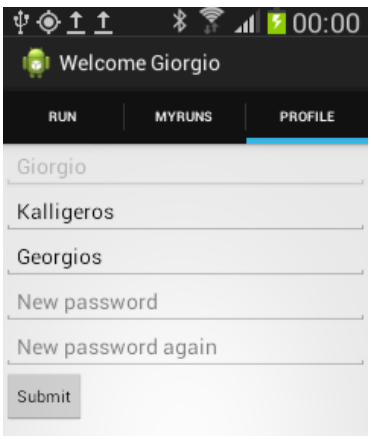
Αν πατηθεί το κουμπί διαγραφής ενός στοιχείου της λίστας, τότε εμφανίζεται ένας διάλογος επιβεβαίωσης για ασφάλεια.



60 Μήνυμα επιβεβαίωσης διαγραφής

7.2.3 Καρτέλα για προβολή και τροποποίηση στοιχείων προφίλ

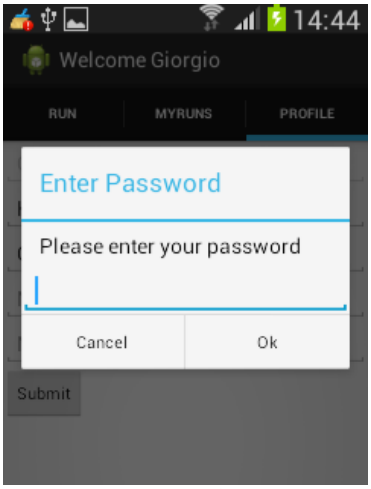
Σε αυτή την καρτέλα ο χρήστης μπορεί να δει και να αλλάξει τα στοιχεία του προφίλ του (όνομα χρήστη, όνομα ,επώνυμο).



Η παρακάτω είναι η αρχική οθόνη που βλέπει ο χρήστης καθώς μεταβαίνει στην καρτέλα profile. Η καρτέλα περιλαμβάνει τα στοιχεία του σε πεδία που επιτρέπουν στο χρήστη να τα επεξεργαστεί για να κάνει τις αλαγές που θελει

61 Οθόνη προβολής-επεξεργασίας στοιχείων προφίλ

Για να γίνει οποιαδήποτε αλλαγή εφόσον έχει πατηθεί το submit ο χρήστης πρέπει να εισάγει τον κωδικό του στο πεδίο που εμφανίζεται.



62 Οθόνη προβολής-επεξεργασίας στοιχείων προφίλ με παράθυρο εισαγωγής κωδικού

7.2 Ιστοσελίδα

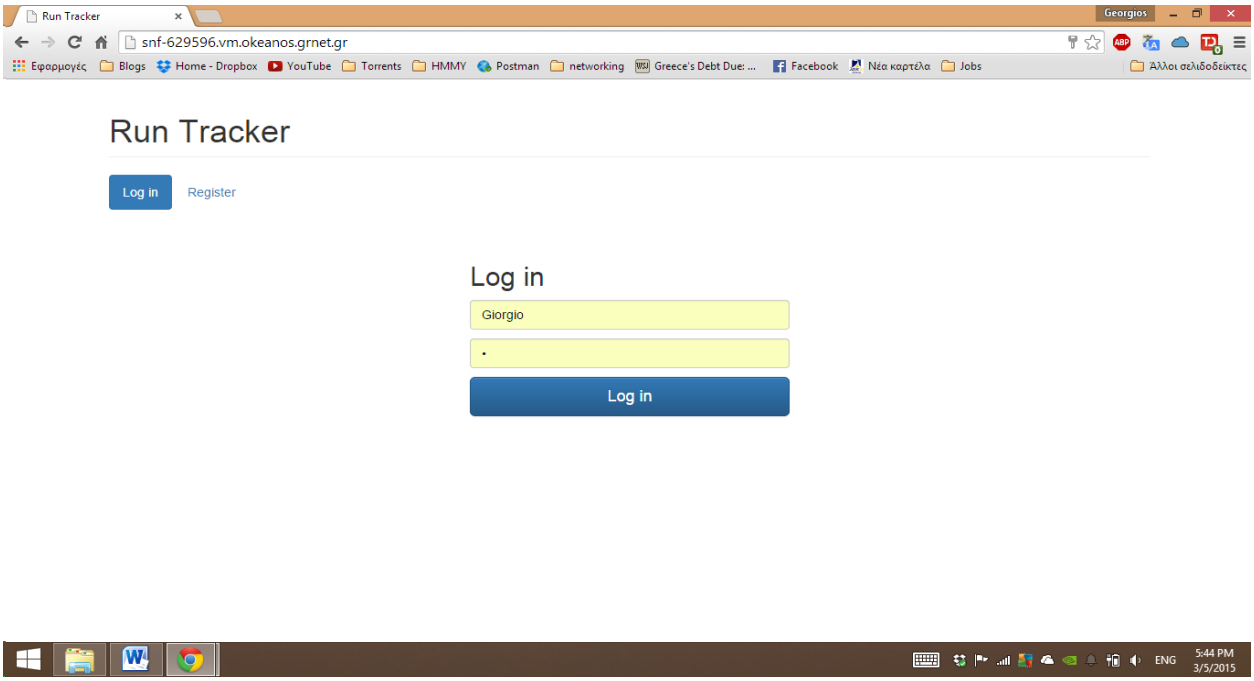
Μέσα από την ιστοσελίδα ο χρήστης μπορεί:

- Να δει όλες τις δραστηριότητες που έχει καταγράψει μέσω της εφαρμογής .
- Να διαγράψει όποια από τις δραστηριότητες επιθυμεί.
- Να δει και να επεξεργαστεί το προφίλ του .
- Να κάνει εγγραφή στο σύστημα.

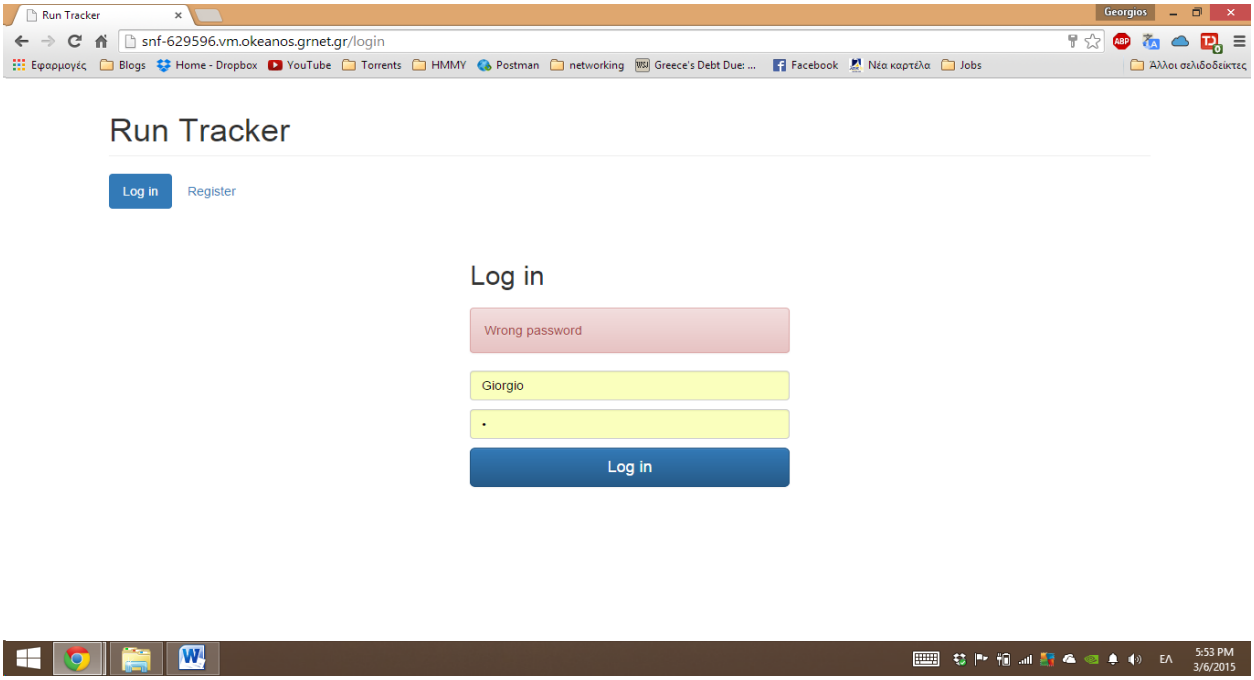
Παρακάτω θα παρουσιαστούν οι βασικές οθόνες διεπαφής της ιστοσελίδας

7.2.1 Οθόνη σύνδεσης

Σε αυτή την οθόνη ο χρήστης συμπληρώνει το όνομα χρήστη και τον κωδικό και πατάει login. Εάν ο συνδυασμός είναι σωστός η ιστοσελίδα τον κατευθύνει στην λίστα των δραστηριοτήτων του σε αντίθετη περίπτωση παίρνει ένα μήνυμα λάθους.



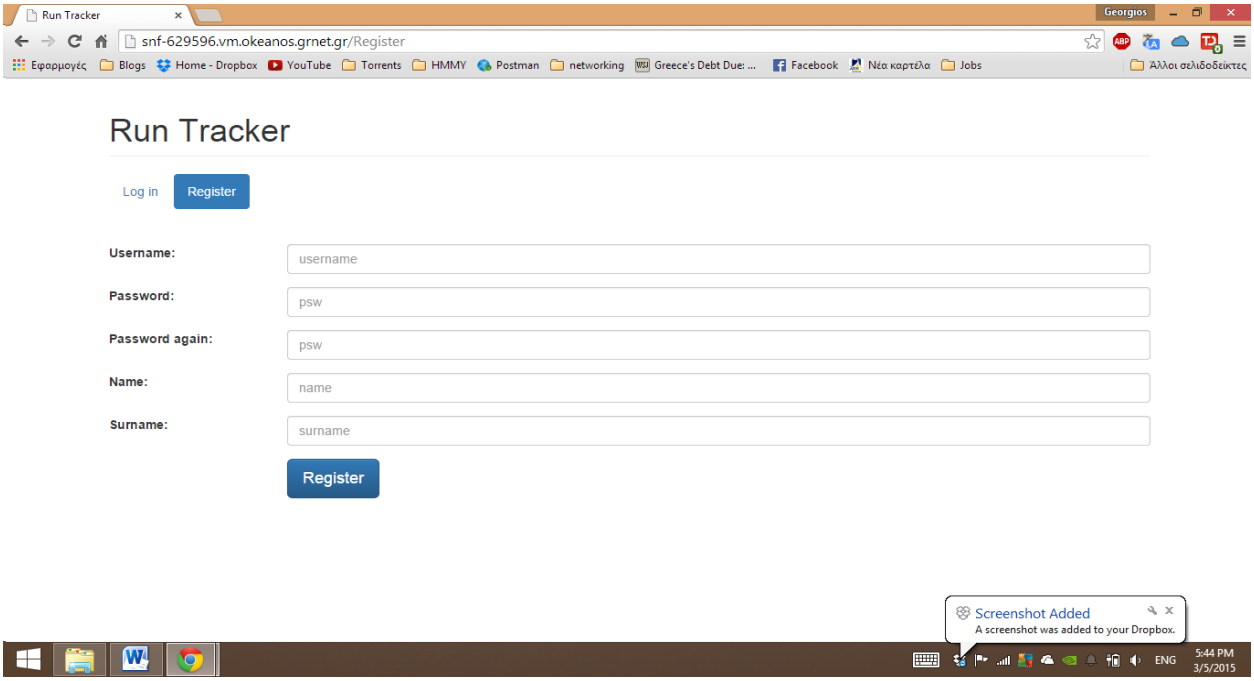
63 Αρχική οθόνη σύνδεσης



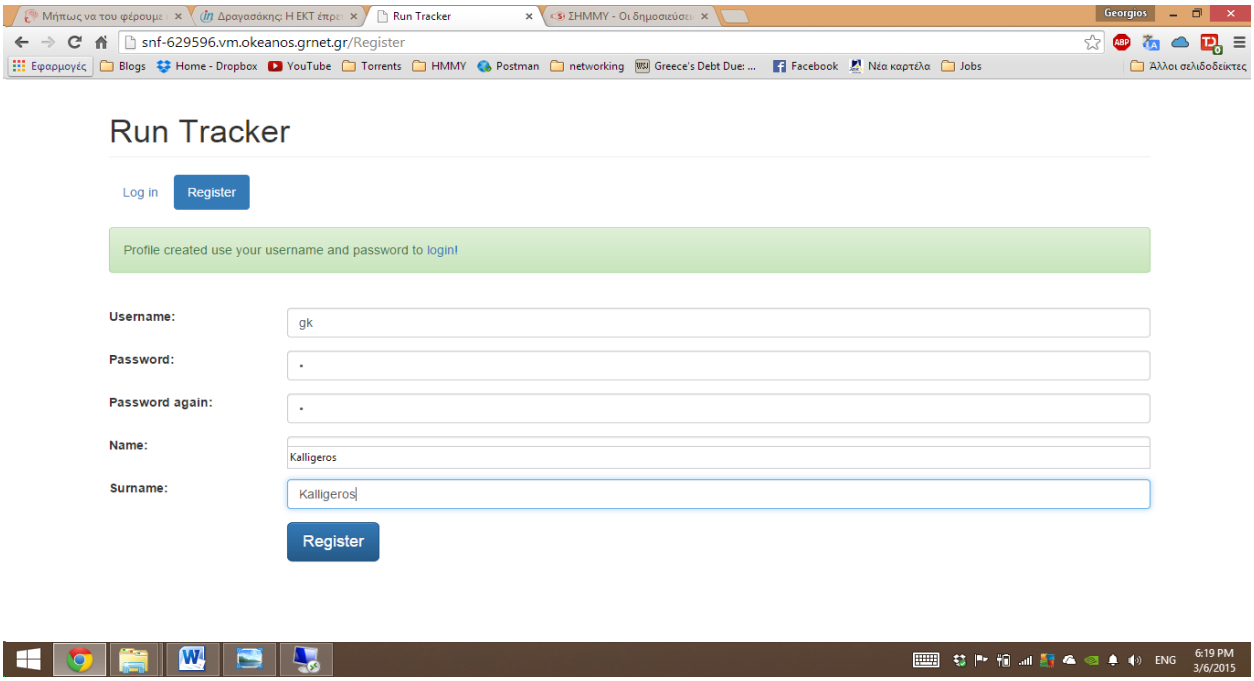
64 Οθόνη σύνδεσης με μήνυμα λάθος κωδικού

7.2.2 Οθόνη εγγραφής

Στη οθόνη αυτή ο χρήστης εισάγει τα στοιχειά του και αν αυτά περάσουν όλους τους ελέγχους, τότε εμφανίζεται ένα μήνυμα που επιβεβαιώνει τη δημιουργία λογαριασμού και προτρέπει το χρήστη να συνδεθεί, αλλιώς εμφανίζεται ένα κατάλληλο μήνυμα λάθους. Μερικές χαρακτηριστικές οθόνες παρουσιάζονται παρακάτω.



65 Αρχική οθόνη εγγραφής χωρίς καμιά είσοδο από το χρήστη



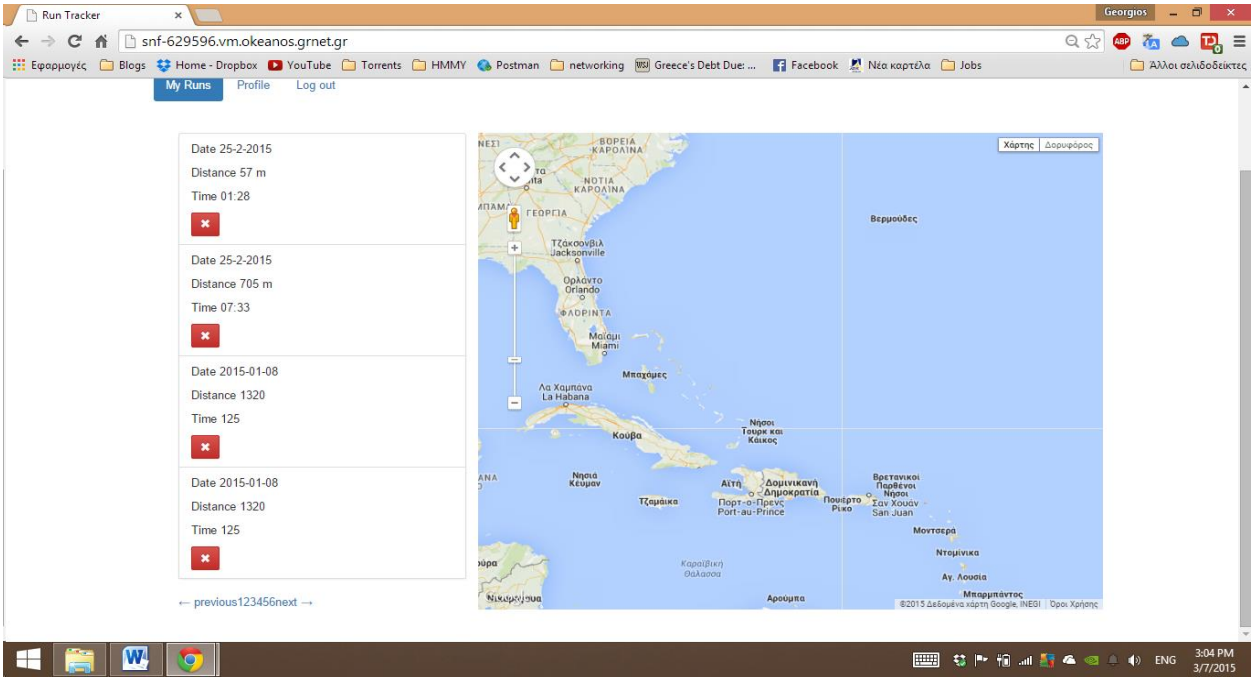
66 Μήνυμα επιτυχούς έγγραφης

7.2.3 Κεντρική ιστοσελίδα

Πρόκειται για την ιστοσελίδα που εμφανίζεται αφού ο χρήστης συνδεθεί στην υπηρεσία. Το μενού της ιστοσελίδας περιέχει τις εξής τρεις επιλογές.

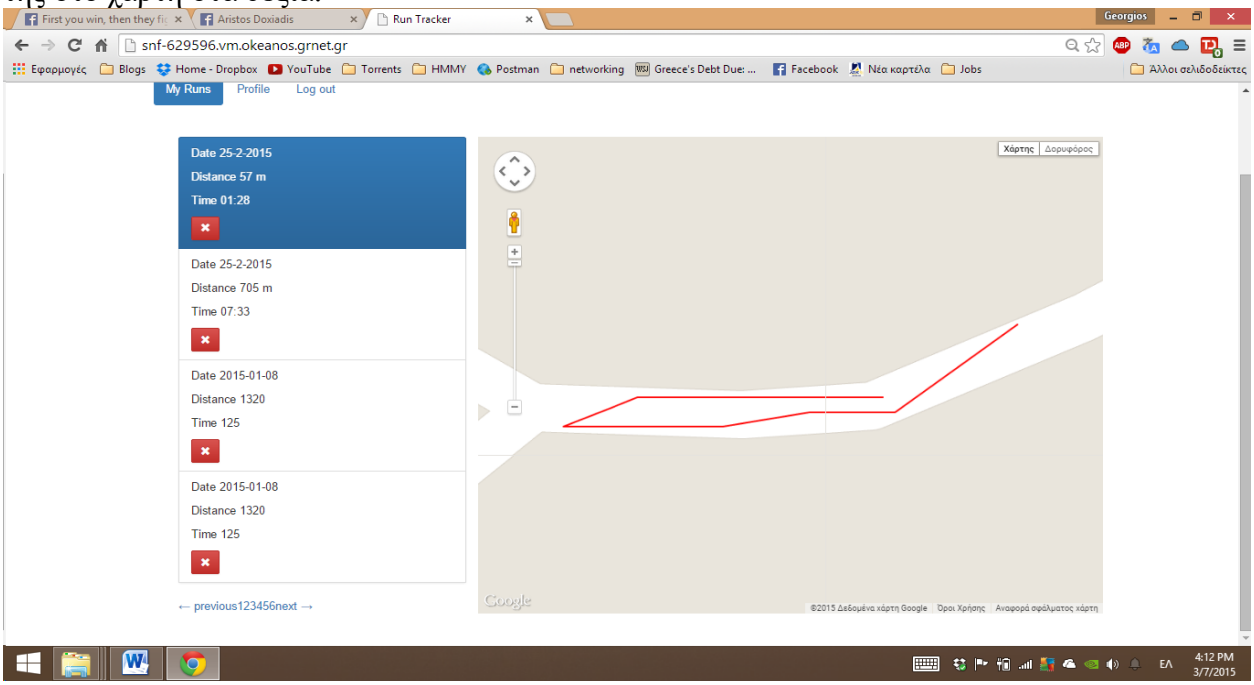
- Τη σελίδα "My Runs" στην οποία μπορεί να δει όλες του τις δραστηριότητες και να διαγράψει όποια θέλει.
- Τη σελίδα "Profile" όπου μπορεί να δει και να επεξεργαστεί τα στοιχεία του προφίλ του.
- Την επιλογή Log out για αποσύνδεση του χρήστη.

Παρακάτω παρουσιάζονται αναλυτικά όλα τα στοιχεία της κεντρικής σελίδας.

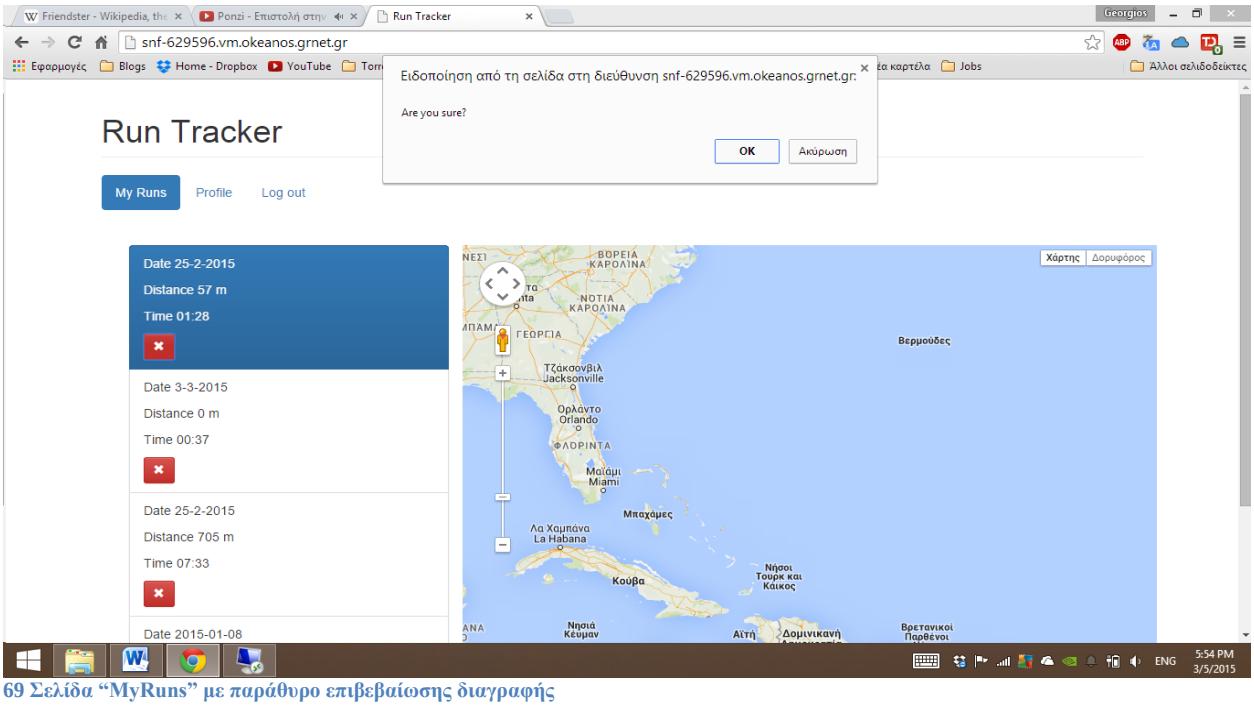


67 Σελίδα "MyRuns" χωρίς κάποια επιλεγμένη δραστηριότητα

Αριστερά βλέπουμε τη λίστα με τις δραστηριότητες ,σελιδοποιημένες ανά 4 στοιχεία ,η ημερομηνία, η απόσταση και ο χρόνος είναι τα στοιχεία κάθε διαδρομής που προβάλλονται στη λίστα, καθώς και το κουμπί διαγραφής , αν επιλεγεί μια διαδρομή στη λίστα εμφανίζεται το ίχνος της στο χάρτη στα δεξιά.



68 Σελίδα "MyRuns" με επιλεγμένη δραστηριότητα

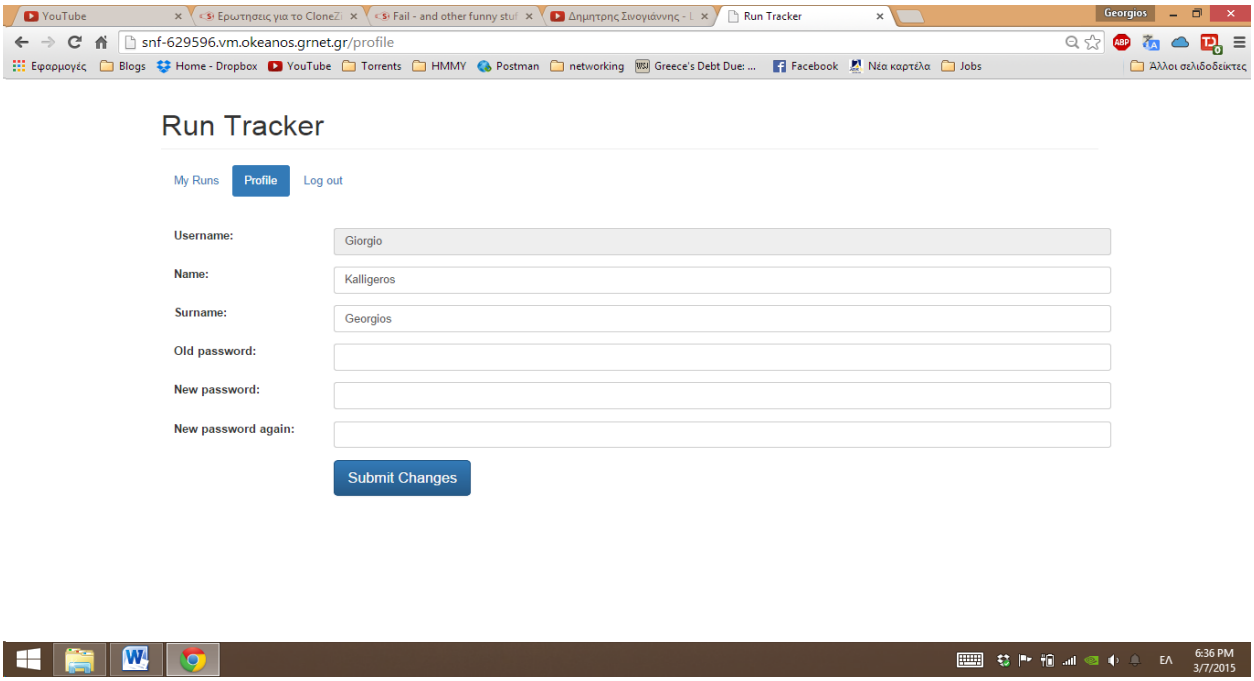


69 Σελίδα "MyRuns" με παράθυρο επιβεβαίωσης διαγραφής

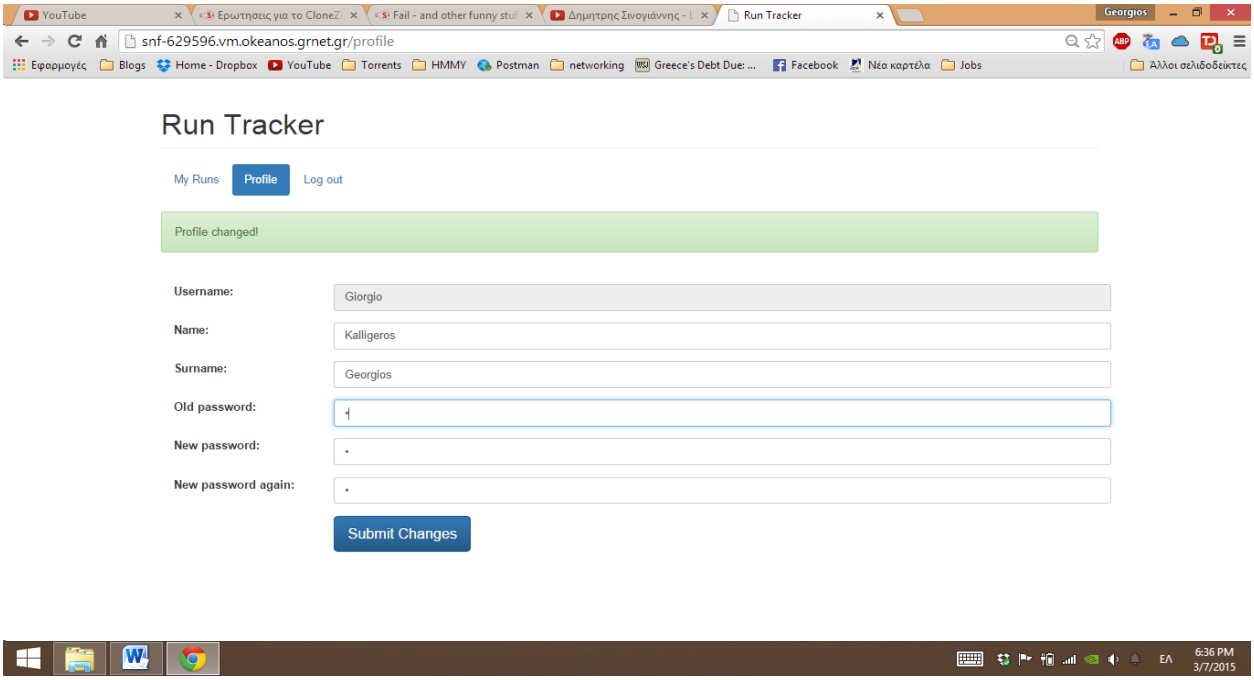
Αν πατηθεί το κουμπί διαγραφής σε κάποια δραστηριότητα της λίστας εμφανίζεται ένα παράθυρο που ζητά επιβεβαίωση της διαγραφής.

Σελίδα Profile

Η σελίδα αυτή επιτρέπει στο χρήστη να δει τα στοιχεία του προφίλ του καθώς και να τα αλλάξει (στο όνομα χρήστη δεν επιτρέπεται αλλαγή).
Για κάθε αλλαγή προφίλ είναι απαραίτητη η επανείσοδος του κωδικού χρήστη. Στην περίπτωση που ο χρήστης δε θέλει να αλλάξει τον κωδικό, απλά δε συμπληρώνει νέο κωδικό στα δυο τελευταία πεδία.



70 Αρχική σελίδα προβολής προφίλ χωρίς είσοδο από τον χρήστη



71 Επιτυχής αλλαγή προφίλ

8. Μελλοντικές επεκτάσεις- διορθώσεις

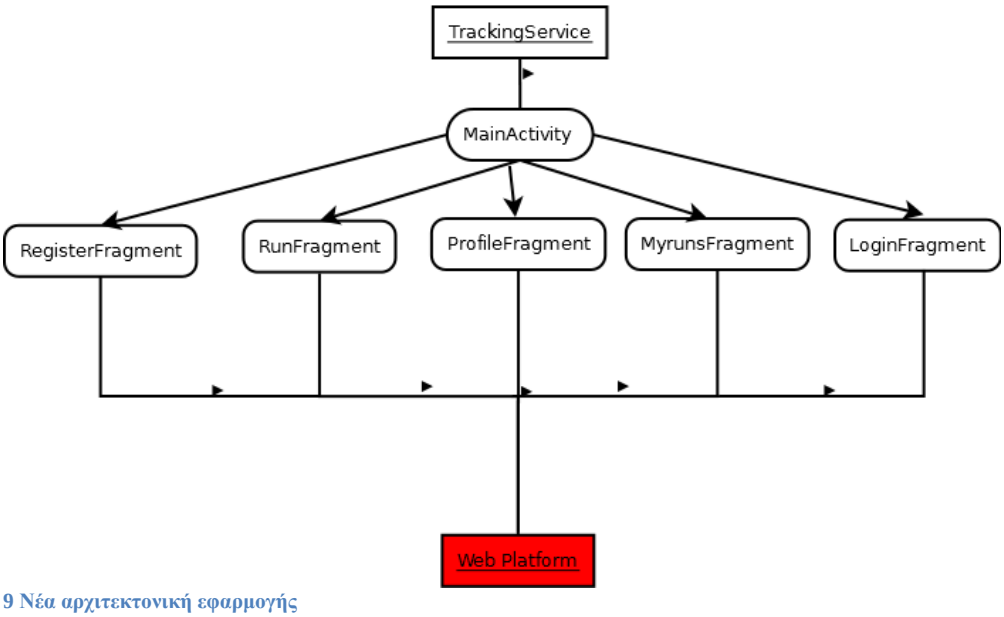
8.1 Εισαγωγή

Κατά την διάρκεια της δημιουργίας της εφαρμογής και της πλατφόρμας καταβλήθηκε η μέγιστη προσπάθεια έτσι να συμπεριληφθούν όλα τα επιθυμητά χαρακτηριστικά ,χωρίς να υπάρχουν σφάλματα στο λογισμικό και χωρίς να επιβραδύνεται σημαντικά η ταχύτητα και οι επιδόσεις. Οι εφαρμογές όμως είναι οντότητες που για να παραμείνουν χρηστικές και επίκαιρες πρέπει να έχουν συνεχή εξέλιξη και ανάπτυξη. Σε αυτό το κεφάλαιο θα παρουσιάσουμε τις αλλαγές και τις προσθήκες που σχεδιάζουμε για την επόμενη έκδοση .

8.2 Mobile εφαρμογή

Ο κύριος στόχος της εφαρμογής είναι ο συνδυασμός ταχύτητας και απλότητας , προς αυτήν τη κατεύθυνση θα κινηθούν και οι αναβαθμίσεις που θα κάνουμε σε αυτήν. Δε θα παραλείψουμε να προσθέσουμε χαρακτηριστικά που θεωρούμε χρήσιμα, έχοντας όμως πάντα στο μυαλό μας να παραμείνει η εφαρμογή απλή γρήγορη και εύχρηστη .Εκτός από τις προσθήκες χαρακτηριστικών θα προβούμε σε τεχνικές αναβαθμίσεις που θα κάνουν την εφαρμογή γρηγορότερη ελαφρύτερη και ευκολότερα συντηρήσιμη .

Στο κομμάτι των τεχνικών αναβαθμίσεων αρχικά σχεδιάζουμε μια νέα αρχιτεκτονική στην οποία όλη η εφαρμογή θα βασίζεται σε μια activity πάνω στην οποία θα έχουμε όλα τα διαφορετικά fragments για όλες τις λειτουργίες .Τα fragments θα εναλλάσσονται ανάλογα με την επιλογή του χρήση. Η μελλοντική αρχιτεκτονική φαίνεται στο παρακάτω σχήμα.



9 Νέα αρχιτεκτονική εφαρμογής

Η συγκεκριμένη αρχιτεκτονική εκτός ότι θα κάνει τον κώδικα πιο εύκολα συντηρήσιμο ,θα κάνει την εφαρμογή αρκετά γρηγορότερη καθώς οι εναλλαγές μεταξύ δραστηριοτήτων κοστίζουν περισσότερο από τις εναλλαγές μεταξύ fragments.

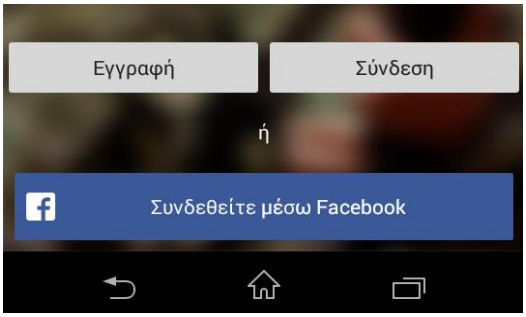
Ένα ακόμα τεχνικό ζήτημα που θα κάνει την εφαρμογή μας περισσότερο αποδοτική όσον

αφορά το μέγεθος των δεδομένων που μεταφέρονται μέσω διαδικτύου είναι η δυναμική σελιδοποίηση της λίστας διαδρομών .Με την δυναμική σελιδοποίηση η πλατφόρμα θα στέλνει τις διαδρομές σε σελίδες μόνο όταν ζητηθούν (και όχι όλες μαζί όπως τώρα), αυτή η αναβάθμιση θα επιφέρει σημαντική μείωση των δεδομένων που μεταφέρονται μέσο διαδικτύου.

Στο κομμάτι των νέων χαρακτηριστικών αρχικά σχεδιάζεται η ριζική αναμόρφωση όλης της διεπαφής καθώς κρίνεται αρκετά παρωχημένη στην κατεύθυνση αυτή θα γίνουν οι εξής ενέργειες :

- Αντικατάσταση των καρτελών με ένα μοντέρνο πλαϊνό μενού.
- Ριζικός ανασχεδιασμός όλων των οθονών με κριτήριο τη χρηστικότητα.
- Ριζικός ανασχεδιασμός της εφαρμογής και της ιστοσελίδας με κριτήριο την καλαισθησία και την ομοιομορφία

Ακολουθώντας τη τάση της εποχής μας θα προσθέσουμε αλληλεπίδραση της εφαρμογής με τα γνωστότερα κοινωνικά δίκτυα συγκεκριμένα σχεδιάζεται η προσθήκη χαρακτηριστικού που να επιτρέπει στο χρήστη να εγγραφεί και να συνδεθεί στην εφαρμογή μέσω του προφίλ του στα γνωστότερα κοινωνικά δίκτυα, ένα ακόμα χαρακτηριστικό που θα προστεθεί είναι η δυνατότητα στο χρήστη να μοιραστεί τις δραστηριότητες του με τους φίλους του στα γνωστότερα κοινωνικά δίκτυα.



72 Η εγγραφή και σύνδεση μέσω Facebook .

Ένα ακόμα χρήσιμο χαρακτηριστικό που θα προστεθεί είναι η δυνατότητα της εφαρμογής να τρέξει σε λειτουργία offline.Όταν η εφαρμογή είναι σε λειτουργία offline ο χρήστης θα μπορεί χωρίς να έχει σύνδεση στο διαδίκτυο να ανοίξει την εφαρμογή και να καταγράψει όσες δραστηριότητες θέλει ,οι οποίες θα αποθηκεύονται τοπικά και συγχρονίζονται με την πλατφόρμα όταν το επιλέξει .

Μια προσθήκη ακόμα που θα έδινε στην εφαρμογή μας μεγαλύτερη ακρίβεια και ευελιξία είναι η χρήση και άλλων αισθητήρων για τον υπολογισμό απόστασης συγκεκριμένα σχεδιάζουμε να χρησιμοποιήσουμε την πυξίδα σε συνδυασμό με το επιταχυνσιόμετρο για τον υπολογισμό των δεδομένων της διαδρομής ,αυτή η προσθήκη θα χρησιμοποιηθεί είτε μαζί με τους άλλους αισθητήρες που χρησιμοποιούμε ήδη για βελτίωση της ακρίβειας είτε και αποκλειστικά. Η ακρίβεια της πυξίδας και του επιταχυνσιόμετρου δεν επηρεάζεται από τις εξωτερικές συνθήκες άρα η εφαρμογή θα γινόταν χρηστική και σε χώρους χωρίς σήμα GPS.

Τέλος θα προστεθεί στην εφαρμογή (με ταυτόχρονη αναβάθμιση του REST API) η δυνατότητα καταγραφής επιπλέον αθλητικών δεδομένων όπως θερμίδες που καταναλώθηκαν , ταχύτητα ,τέλος θα γίνουν διαθέσιμα στατιστικά.

8.3 Web πλατφόρμα REST API /Website

Το κομμάτι του REST API θεωρείται αρκετά πλήρες και καλύπτει σε μεγάλο βαθμό τις ανάγκες της εφαρμογής, παρόλα αυτά σχεδιάζονται αρκετές αναβαθμίσεις τόσο σε τεχνικό επίπεδο αλλά και προσθήκη νέων χαρακτηριστικών και δυνατοτήτων.

Αρχικά σχεδιάζεται η αναβάθμιση των JWTs σε ένα πολυεπίδεδο σύστημα εξουσιοδότησης που θα παρέχει ελεγχόμενη πρόσβαση στις πληροφορίες ,για παράδειγμα εάν ό χρήστης Α εκδώσει ένα JWT θα μπορεί να δει τις διαδρομές και το προφίλ του χρήστη Β (εάν ο Β έχει δώσει πρόσβαση αλλά προφανώς δε θα μπορεί να τις διαγράψει ή να κάνει αλλαγές).Με το αναβαθμισμένο αυτό σύστημα εξουσιοδοτήσεων θα μπορέσουμε να δημιουργήσουμε URLs με περισσότερες λειτουργίες στο API .

Στο κομμάτι του Website που συνοδεύει την εφαρμογή θα γίνουν οι απαραίτητες αλλαγές ως προς την εμφάνιση έτσι ώστε ο σχεδιασμός του και το στυλ του να συνάδει με αυτό της εφαρμογής (στο εικαστικό κομμάτι) .. Πέρα από αυτά η ιστοσελίδα θα κρατηθεί απλό συνοδευτικό στην εφαρμογή.

Στο τεχνικό κομμάτι ιστοσελίδας σχεδιάζουμε την αλλαγή της σελίδας στη μορφή μιας Web εφαρμογής μια σελίδας (one page web app) ,αυτό θα επιτευχθεί με χρήση του ίδιου REST API που χρησιμοποιούμε και για την εφαρμογή για Android και της τεχνολογίας AJAX*,η αλλαγή αυτή θα φέρει βελτίωση της εμπειρίας χρήσης της ιστοσελίδας καθώς δεν θα έχουμε καμία επαναφόρτώση της από τη στιγμή που ο χρήστης μπαίνει σε αυτήν .

*Η τεχνολογία AJAX επιτρέπει μεταφορά δεδομένων από κάποιο server ασύγχρονα και χωρίς επαναφόρτωση της σελίδας

Παράρτημα 1 (κώδικας)

Εγκατάσταση

Για λειτουργήσει η εφαρμογή ο Server χρειάζεται να έχει εγκατεστημένο το Node.js και την MongoDB, εφόσον έχουμε εγκαταστήσει τα παραπάνω τρέχουμε την MongoDB και δημιουργούμε μια βάση δεδομένων με όνομα Run_Tracker μέσα σε αυτήν μια συλλογή με όνομα users και μια με όνομα tracks. Έχοντας κάνει την παρακάτω διαδικασία και την MongoDB να τρέχει , ξεκινάμε την γραμμή εντολών του Node.js και πηγαίνουμε στον φάκελο με τα αρχεία της εφαρμογής, τρέχουμε την εντολή "npm build" για να κατέβουν οι απαραίτητες βιβλιοθήκες και μετά την εντολή "node app.js". Με το πέρας αυτής την διαδικασίας το REST API είναι έτοιμο να δεχτεί αιτήματα στην ip του Server.

Κώδικας Web εφαρμογής

Αρχείο app.js

```
1.  var express = require('express'),
2.      jwt = require('jwt-simple'),
3.      moment = require('moment'),
4.      routes = require('./routes'),
5.      http = require('http'),
6.      path = require('path'),
7.      mongoskin = require('mongoskin'),
8.      dbUrl = process.env.MONGOHQ_URL || 'mongodb://@localhost:27017/Run_tracker';
9.  db = mongoskin.db(dbUrl, {
10.         safe: true
11.     }),
12.     collections = {
13.         tracks: db.collection('tracks'),
14.         users: db.collection('users')
15.     };
16. // Express.js Middleware
17. var session = require('express-session'),
18.     logger = require('morgan'),
19.     errorHandler = require('errorhandler'),
20.     cookieParser = require('cookie-parser'),
21.     bodyParser = require('body-parser'),
22.     methodOverride = require('method-override');
23. var app = express();
24. app.locals.appTitle = 'Run Tracker';
25. app.set('jwtTokenSecret', 'tsakatsouka');
26. // Expose collections to request handlers
27. app.use(function(req, res, next) {
28.     if (!collections.tracks || !collections.users) return next(new Error('No collections.'))
29.     req.collections = collections;
30.     return next();
31. });
32. // Express.js configurations
33. app.set('port', process.env.PORT || 80);
34. app.set('views', path.join(__dirname, 'views'));
35. app.set('view engine', 'jade');
```

```
36. // Express.js middleware configuration
37. app.use(logger('dev'));
38. app.use(bodyParser.json());
39. app.use(bodyParser.urlencoded());
40. app.use(methodOverride());
41. app.use(require('stylus').middleware(__dirname + '/public'));
42. app.use(express.static(path.join(__dirname, 'public')));
43. app.use(cookieParser('3CCC4ACD-6ED1-4844-9217-82131BDCB239'));
44. app.use(session({
45.     secret: '2C44774A-D649-4D44-9535-46E296EF984F'
46. }))
47. if ('development' == app.get('env')) {
48.     app.use(errorHandler());
49. }
50. // website authorized routes
51. app.get('/', routes.auth.validateuser, routes.index);
52. app.get('/profile', routes.auth.validateuser, routes.profile);
53. app.get('/stats', routes.auth.validateuser, routes.stats);
54. // website guest routes
55. app.get('/logout', routes.logout);
56. app.get('/login', routes.login);
57. app.get('/register', routes.register);
58. //app.post('/register',routes.webregister);
59. app.post('/login', routes.auth.settoken, routes.weblogin);
60. app.post('/mlogin', routes.auth.settoken, routes.mobilelogin);
61. //api routes
62. app.get('/api/profile', routes.auth.validateuser, routes.api.get_profile); //get a profile
63. app.post('/api/profile', routes.api.new_profile); //new profile
64. app.put('/api/profile', routes.auth.validateuser, routes.api.edit_profile); //edit profile
65. app.del('/api/profile', routes.auth.validateuser, routes.api.del_profile); //delete  profile
66. app.get('/api/profile/tracks', routes.auth.validateuser, routes.auth.validateuser, routes.api.get_
    tracks); //get a users tracks
67. app.post('/api/profile/tracks', routes.auth.validateuser, routes.api.new_track); //new track
68. app.del('/api/profile/tracks', routes.auth.validateuser, routes.api.del_track); //delete a track b
    ased on its id
69. app.get('/api/profile/tracks_p/:page/:size', routes.api.get_paged_tracks);
70. app.all('*', function(req, res) {
71.     res.send(404);
72. })
73. var server = http.createServer(app);
74. var boot = function() {
75.     server.listen(app.get('port'), function() {
76.         console.info('Express server listening on port ' + app.get('port'));
77.     });
78. }
79. var shutdown = function() {
80.     server.close();
81. }
82. if (require.main === module) {
83.     boot();
84. } else {
85.     console.info('Running app as a module')
86.     exports.boot = boot;
87.     exports.shutdown = shutdown;
88.     exports.port = app.get('port');
89. }
```

Αρχείο api.js

```
1.  moment = require('moment');
2.  uuid = require('node-uuid');
3.  crypto = require("crypto");
4.
5.  exports.get_profile = function(req, res, next) {
6.      if (req.user) {
7.          res.json(req.user);
8.      } else if (req.error) {
9.          res.json({
10.             error: req.error
11.         });
12.     } else {
13.
14.         res.json({
15.             error: "something is fucked up"
16.         });
17.     }
18. };
19. exports.new_profile = function(req, res, next) {
20.     if ((!req.body.email)(!req.body.username) || (!req.body.password) || (!req.body.name) || (!req
    .body.surname)) {
21.         res.json({
22.             error: "Fill all data"
23.         })
24.     } else {
25.         req.collections.users.findOne({
26.             username: req.body.username,
27.         }, function(error, user) {
28.             console.log("user is ", user);
29.             if (error) {
30.                 // send database error
31.                 return next(error);
32.             }
33.             if (user == null) {
34.                 var salt = uuid.v1(); //salt for encryption
35.                 var pass_enc = crypto.createHmac('sha256', salt).update(req.body.password).digest(
    'hex'); //encrypted pass
36.                 req.collections.users.insert({
37.                     "username": req.body.username,
38.                     "name": req.body.name,
39.                     "surname": req.body.surname,
40.                     "psw": pass_enc,
41.                     "salt": salt
42.                 }, function(error, userresponse) {
43.                     if (error) return next(error);
44.                     res.json({
45.                         ok: "registration completed"
46.                     });
47.                 });
48.
49.             } else {
50.                 res.json({
51.                     error: "username taken"
52.                 });
53.             }
54.
55.         })
56.
57.
```

```
58.     };
59. 
60. };
61. //
62. exports.edit_profile = function(req, res, next) {
63.     console.log("error:", req.error);
64.     if (req.error) {
65.         res.json({
66.             error: req.error
67.         });
68.     } else if (req.user.username != req.body.username) {
69.         res.json({
70.             error: "Not authorized"
71.         });
72.     } else {
73.         req.collections.users.findOne({
74.             username: req.body.username,
75.         }, function(error, user) {
76.             if (error) {
77.                 // send database error
78.                 return next(error);
79.             } else if (user == null) {
80.                 res.json({
81.                     error: "Wrong password "
82.                 });
83.             } else if (crypto.createHmac('sha256', user.salt).update(req.body.password).digest('he
    x') == user.psw) {
84.                 pass_enc = crypto.createHmac('sha256', user.salt).update(req.body.password_r).dige
    st('hex');
85.                 req.collections.users.update({
86.                         username: req.body.username
87.                     }, {
88.                         username: req.body.username,
89.                         name: req.body.name,
90.                         surname: req.body.surname,
91.                         psw: pass_enc,
92.                         salt: user.salt
93.                     }, {
94.                         strict: true
95.                     },
96.                     function(error, result) {
97.                         // console.log(result);
98.                         if (error) {
99.                             next(error)
100.                        } else {
101.                            res.json({
102.                                succes: "Profile changed!"
103.                            });
104.                        }
105. 
106.                    })
107. 
108.            } else {
109.                res.json({
110.                    error: "Wrong password "
111.                });
112.            }
113. 
114. 
115.        })
116. 
```

```
    }

};

exports.del_profile = function(req, res, next) {
    res.json({
        error: "not implemented"
    });
};


//GET tracks api

exports.get_tracks = function(req, res, next) {
    if (req.error) {
        res.json({
            error: req.error
        });
    } else {
        req.collections.tracks.find({
            username: req.user.username
        }).toArray(
            function(error, tracks) {
                if (error) {
                    // send database error
                    return next(error);
                } else {
                    // console.log("tracks are ", tracks);
                    res.json(tracks);
                }

            })
    }
};

//not used at the moment ( server pagination)
exports.get_paged_tracks = function(req, res, next) {
    var size = req.params.size;
    var page = req.params.page;
    skip = page > 0 ? ((page - 1) * size) : 0;
    //  console.log(size, page);
    req.collections.tracks.find({
        username: req.user.username
    }, null, {
        skip: skip,
        limit: size
    }, function(err, data) {
        if (err) {
            res.json(500, err);
        } else {
            res.json({
                data: data
            });
        }
    });
    next();
}
exports.new_track = function(req, res, next) {
    // console.log(req.body);
    if (req.error) {
        //not authorized
```

```
        res.json({
            error: req.error
        });
    } else {
        if ((req.user.username) && (req.body.date) && (req.body.distance) && (req.body.time) && (r
    eq.body.track)) {
            req.collections.tracks.insert({
                "username": req.user.username,
                "date": req.body.date,
                "distance": req.body.distance,
                "time": req.body.time,
                "track": req.body.track
            }, function(error, userresponse) {
                if (error) return next(error);
                res.json({
                    "ok": "ok"
                });
            });
        } else {
            res.json({
                error: "something missing"
            });
        }

    }
};
exports.del_track = function(req, res, next) {
    //  console.log( req.body);
    if (req.error) {
        res.json({
            error: req.error
        });
    } else {
        my_id = (req.body && req.body.id) || (req.query && req.query.id) || req.headers['id'];
        req.collections.tracks.findById(
            my_id,
            function(error, track) {
                if (error) {
                    // send database error
                    return next(error);
                }
                if (track == null) {
                    //no track with such id
                    res.json({
                        error: "no such track"
                    });
                } else {
                    if (req.user.username === track.username) {
                        req.collections.tracks.removeById(my_id, function(error, count) {
                            if (error) return next(error);
                            res.json({
                                affectedCount: count
                            });
                        });

                    } else {
                        res.json({
                            error: "not authorized to delete"
                        });
                    }
                }
```

```
238.            })
239.    }
240.};
```

Αρχείο auth.js

```
1.  var jwt = require('jwt-simple'),
2.      moment = require('moment'),
3.      crypto = require("crypto")
4.  exports.settoken = function(req, res, next) {
5.      if (!req.body.username || !req.body.password) {
6.          req.error = "nopass";
7.          return next();
8.      }
9.      req.collections.users.findOne({
10.         username: req.body.username,
11.     }, function(error, user) {
12.         //  console.log("user is ", user);
13.         if (error) {
14.             // send database error
15.             return next(error);
16.         }
17.         if (user == null) {
18.             // send wrong conbination response
19.             req.error = "No user ";
20.         } else {
21.             //check if password is corect
22.             if (user.psw == crypto.createHmac('sha256', user.salt).update(req.body.password).diges
    t('hex')) {
23.                 console.log("authorized");
24.                 var expires = moment().add(180, 'minutes').valueOf();
25.                 var token = jwt.encode({
26.                     username: user.username,
27.                     exp: expires
28.                 }, 'verysecretpassphrase');
29.                 req.token = token;
30.             } else {
31.                 console.log("unauthorized");
32.                 req.error = "Wrong password ";
33.             }
34.         }
35.         return next();
36.     })
37. };
38. exports.validateuser = function(req, res, next) {
39.     var token = (req.body && req.body.token) || (req.query && req.query.token) || req.headers['x-
    access-token'] || (req.session.token);
40.     console.log("validateuser");
41.     if (!token) {
42.         // no token
43.         req.error = "Unauthorized";
44.         return next();
45.     } else {
46.         //decode token
47.         var decoded = jwt.decode(token, 'tsakatsoukou');
48.         console.log("decoded", decoded);
49.         console.log("encoded", token);
50.         if (decoded.exp <= Date.now()) {
51.             req.error = "expired";
```

```
            return next();
        } else {
            req.collections.users.findOne({
                username: decoded.username,
            }, function(error, user) {
                if (error) {
                    // send database error
                    return next(error);
                }
                if (user == null) {
                    //no user with such id (fake token)
                    req.error = "no_user";
                    //     console.log("no_user");
                    return next();
                } else {
                    //hide password
                    user.psw = "";
                    req.user = user;
                    return next();
                }
            })
        }
    }
}
```

Κώδικας mobile εφαρμογής

Αρχείο IntroActivity.java

```
package com.example.run_tracker;
import android.app.Activity;
import android.app.FragmentTransaction;
import android.graphics.Color;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.TextView;
import android.widget.Toast;
public class IntroActivity extends Activity implements OnClickListener {

    private boolean state = false;
    private TextView mBottomText;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_intro);
        mBottomText = (TextView) findViewById(R.id.bottom_text);
        // check if orientation changed
        if (savedInstanceState != null) {
            state = savedInstanceState.getBoolean("STATE");
        } else {
            state = false;
```

```
        }
        mBottomText.setOnClickListener(this);
        if (!state) {
            LoginFragment login = new LoginFragment();
            FragmentTransaction ft = getFragmentManager().beginTransaction();
            ft.replace(R.id.container, login);
            ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN);
            ft.commit();
            mBottomText.setText(R.string.RegisterPrompt);

        } else {
            RegisterFragment register = new RegisterFragment();
            FragmentTransaction ft = getFragmentManager().beginTransaction();
            ft.replace(R.id.container, register);
            ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN);
            ft.commit();
            mBottomText.setText(R.string.LoginPrompt);
        }
        mBottomText.setTextColor(Color.BLUE);
    }

    public void onSaveInstanceState(Bundle savedInstanceState) {
        //
        // Save the user's current fragment (login or register)
        savedInstanceState.putBoolean("STATE", state);
        // Always call the superclass so it can save the view hierarchy state
        super.onSaveInstanceState(savedInstanceState);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.intro, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    //change between two fragments
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.bottom_text:
                {
                    //check witch fragment is currently in use
                    if (state) {
                        LoginFragment login = new LoginFragment();
                        FragmentTransaction ft = getFragmentManager().beginTransaction();
                        ft.replace(R.id.container, login);
                        ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN);
                        ft.commit();
                        mBottomText.setText(R.string.RegisterPrompt);
```

```
                        state = false;
                    } else {
                        RegisterFragment register = new RegisterFragment();
                        FragmentTransaction ft = getFragmentManager().beginTransaction();
                        ft.replace(R.id.container, register);
                        ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN);
                        ft.commit();
                        mBottomText.setText(R.string.LoginPrompt);
                        state = true;
                    }

                    break;
                }

            default:
                {
                    Toast.makeText(this, "click on something", Toast.LENGTH_LONG)
                        .show();
                }
        }
    }
}
```

Αρχείο LoginFragment.java

```
package com.example.run_tracker;

import org.json.JSONException;
import org.json.JSONObject;

import android.app.Fragment;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.Toast;

import com.android.volley.Request.Method;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.VolleyLog;
import com.android.volley.toolbox.JsonObjectRequest;

public class LoginFragment extends Fragment implements OnClickListener {
    private String TAG = "Login";
    Button mLogin;
    EditText mUsername, mPassword;
    private ProgressDialog pDialog;
    private CheckBox mRememberme;

```

```
35.     public View onCreateView(LayoutInflater inflater, ViewGroup container,
36.     Bundle savedInstanceState) {
37.         View rootView = inflater.inflate(R.layout.loginfragment, container, false);
38.         mLogin = (Button) rootView.findViewById(R.id.login);
39.         mUsername = (EditText) rootView.findViewById(R.id.username);
40.         mPassword = (EditText) rootView.findViewById(R.id.password);
41.         mRememberme = (CheckBox) rootView.findViewById(R.id.rememberme);
42.         mLogin.setOnClickListener(this);
43.         SharedPreferences credentials = getActivity().getPreferences(Context.MODE_PRIVATE);
44.         mUsername.setText(credentials.getString("username", ""));
45.         mPassword.setText(credentials.getString("password", ""));
46.         return rootView;
47. 
48.     }
49.     public void onDestroy() {
50.         super.onDestroy();
51.         Log.v(TAG, "onDestroy");
52.         AppController.getInstance().cancelPendingRequests(TAG);
53.     }
54. 
55.     @Override
56.     public void onClick(View v) {
57.         // TODO Auto-generated method stub
58.         if (isEmpty(mUsername) || isEmpty(mPassword)) {
59.             Toast.makeText(getActivity(), "Fill all fields", Toast.LENGTH_LONG)
60.                 .show();
61.         } else {
62.             // request here
63.             Make_login_request();
64. 
65.         }
66.     }
67. 
68.     private boolean isEmpty(EditText etText) {
69.         return etText.getText().toString().trim().length() == 0;
70.     }
71. 
72.     private void Make_login_request() {
73.         pDialog = new ProgressDialog(getActivity());
74.         pDialog.setMessage("Login in ");
75.         showProgressDialog();
76.         JSONObject credentials = null;
77.         try {
78.             credentials = new JSONObject();
79.             credentials.put("username", mUsername.getText().toString());
80.             credentials.put("password", mPassword.getText().toString());
81. 
82.         } catch (JSONException e) {
83.             // TODO Auto-generated catch block
84.             e.printStackTrace();
85.         }
86. 
87.         JsonObjectRequest jsonObjReq = new JsonObjectRequest(Method.POST,
88.         URLS.URL_LOGIN, credentials,
89.         new Response.Listener < JSONObject > () {
90. 
91.             @Override
92.             public void onResponse(JSONObject response) {
93.                 /*
94.                  * response.getString("token");
95.                  *
```

```
96.                  * (and save the token for future requests) else we got
97.                  * an error
98.                  */
99.                 Log.d(TAG, response.toString());
100.                try {
101.                    response.getString("token");
102.                    if (mRememberme.isChecked()) {
103.                        Log.v(TAG, "saving token");
104.                        // save username and password for future use ins
105.                        // shared preferences
106.                        SharedPreferences credentials = getActivity()
107.                            .getPreferences(Context.MODE_PRIVATE);
108.                        SharedPreferences.Editor editor = credentials.edit();
109.                        editor.putString("username", mUsername.getText().toString());
110.                        editor.putString("password", mPassword.getText().toString());
111.                        editor.commit();
112.                    }
113.                    Launch_app(response.getString("token"));
114.
115.                } catch (JSONException e) {
116.                    // TODO Auto-generated catch block
117.                    try {
118.
119.                        Toast.makeText(getActivity(),
120.                        response.getString("error"),
121.                        Toast.LENGTH_LONG).show();
122.                    } catch (JSONException e1) {
123.                        // TODO Auto-generated catch block
124.                        e1.printStackTrace();
125.                    }
126.                }
127.                hideProgressDialog();
128.            }
129.        }, new Response.ErrorListener() {
130.            @Override
131.            public void onErrorResponse(VolleyError error) {
132.                VolleyLog.d(TAG, "Error: " + error.getMessage());
133.                hideProgressDialog();
134.            }
135.        }) {
136.        };
137.        // Adding request to request queue
138.        AppController.getInstance().addToRequestQueue(jsonObjReq, TAG);
139.    }
140.    private void Launch_app(String token) {
141.        Intent intent = new Intent(getActivity(), MainActivity.class);
142.        intent.putExtra("token", token);
143.        intent.putExtra("Username", mUsername.getText().toString());
144.        startActivity(intent);
145.    }
146.    private void showProgressDialog() {
147.        if (!pDialog.isShowing()) pDialog.show();
148.    }
149.    private void hideProgressDialog() {
150.        if (pDialog.isShowing()) pDialog.hide();
151.    }
152.}
```

Αρχείο RegisterFragment.java

```
1.  package com.example.run_tracker;
2.
3.  import org.json.JSONException;
4.  import org.json.JSONObject;
5.
6.  import android.app.Dialog;
7.  import android.app.Fragment;
8.  import android.app.ProgressDialog;
9.  import android.os.Bundle;
10. import android.util.Log;
11. import android.view.LayoutInflater;
12. import android.view.View;
13. import android.view.View.OnClickListener;
14. import android.view.ViewGroup;
15. import android.widget.Button;
16. import android.widget.EditText;
17. import android.widget.Toast;
18.
19. import com.android.volley.Request.Method;
20. import com.android.volley.Response;
21. import com.android.volley.VolleyError;
22. import com.android.volley.VolleyLog;
23. import com.android.volley.toolbox.JsonObjectRequest;
24. public class RegisterFragment extends Fragment implements OnClickListener {
25.     EditText mUsername, mPassword1, mPassword2, mName, mSurname;
26.     Button mRegister;
27.     private Dialog pDialog;
28.     private String TAG = "Register";
29.
30.     public View onCreateView(LayoutInflater inflater, ViewGroup container,
31.     Bundle savedInstanceState) {
32.
33.         View rootView = inflater.inflate(R.layout.registerfragment, container,
34.         false);
35.         mRegister = (Button) rootView.findViewById(R.id.register);
36.         mUsername = (EditText) rootView.findViewById(R.id.username);
37.         mPassword1 = (EditText) rootView.findViewById(R.id.password1);
38.         mPassword2 = (EditText) rootView.findViewById(R.id.password2);
39.         mName = (EditText) rootView.findViewById(R.id.name);
40.         mSurname = (EditText) rootView.findViewById(R.id.surname);
41.         mRegister.setOnClickListener(this);
42.         return rootView;
43.     }
44.
45.     public void onDestroy() {
46.         super.onDestroy();
47.         AppController.getInstance().cancelPendingRequests(TAG);
48.     }
49.
50.     @Override
51.     public void onClick(View v) {
52.         if (!mPassword1.getText().toString()
53.             .equals(mPassword2.getText().toString())) {
54.
55.             Toast.makeText(getActivity(), "Passwords dont match",
56.             Toast.LENGTH_LONG).show();
57.         } else if (isEmpty(mPassword1) || isEmpty(mPassword2) || isEmpty(mUsername) || isEmpty(mNa
    me) || isEmpty(mSurname)) {
```

```
58.
59.             Toast.makeText(getActivity(), "Fill all fields", Toast.LENGTH_LONG)
60.                 .show();
61.         } else {
62.             // post request
63.             Make_register_request();
64.
65.         }
66.
67.     }
68.
69.     private boolean isEmpty(EditText etText) {
70.         return etText.getText().toString().trim().length() == 0;
71.     }
72.
73.     private void Make_register_request() {
74.         pDialog = new ProgressDialog(getActivity());
75.
76.         showProgressDialog();
77.         JSONObject credentials = null;
78.         try {
79.             credentials = new JSONObject();
80.             credentials.put("username", mUsername.getText().toString());
81.             credentials.put("password", mPassword1.getText().toString());
82.             credentials.put("name", mName.getText().toString());
83.             credentials.put("surname", mSurname.getText().toString());
84.
85.         } catch (JSONException e) {
86.             e.printStackTrace();
87.         }
88.
89.         JsonObjectRequest jsonObjReq = new JsonObjectRequest(Method.POST,
90.         URLS.URL_PROFILE, credentials,
91.         new Response.Listener < JSONObject > () {
92.
93.             @Override
94.             public void onResponse(JSONObject response) {
95.                 hideProgressDialog();
96.                 try {
97.                     response.getString("ok");
98.                     Toast.makeText(getActivity(),
99.                         "Registration completed please login",
100.                    Toast.LENGTH_LONG).show();
101.                } catch (JSONException e) {
102.                    try {
103.                        Toast.makeText(getActivity(),
104.                        response.getString("error"),
105.                        Toast.LENGTH_LONG).show();
106.                    } catch (JSONException e1) {
107.                        e1.printStackTrace();
108.                    }
109.                    e.printStackTrace();
110.                }
111.            }
112.        }, new Response.ErrorListener() {@Override
113.            public void onErrorResponse(VolleyError error) {
114.                VolleyLog.d(TAG, "Error: " + error.getMessage());
115.                hideProgressDialog();
116.            }
117.        }) {
118.
```

```
        };
        // Adding request to request queue
        AppController.getInstance().addToRequestQueue(jsonObjReq, TAG);


    }
    private void showProgressDialog() {
        if (!pDialog.isShowing()) pDialog.show();
    }
    private void hideProgressDialog() {
        if (pDialog.isShowing()) pDialog.hide();
    }
}
```

Αρχείο MainActivity.java

```
package com.example.run_tracker;
import android.annotation.SuppressLint;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.support.v4.app.FragmentActivity;
import android.support.v4.view.ViewPager;
import android.app.ActionBar;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import com.example.run_tracker.TrackingService.LocalBinder;
/*Main activity starts a tracking service that is running constantly and can be accessed from frag
ments*/
public class MainActivity extends FragmentActivity implements
    android.app.ActionBar.TabListener
{

    Intent serviceintent;
    private String TAG = "MTrackerr";
    private String Username;
    private ViewPager viewPager;
    private TabsPagerAdapter mAdapter;
    private android.app.ActionBar actionBar;
    private String[] tabs =
    { "run", "myRuns", "profile" };
    private String token;
    TrackingService myService;
    boolean isBound = false;

    public String getToken()
    {
    return token;
    }

    public void setToken(String token)
    {
    this.token = token;
```

```
41.     }
42.
43.     public String getUsername()
44.     {
45.     return Username;
46.     }
47.
48.     public void setUsername(String username)
49.     {
50.     Username = username;
51.     }
52.
53.     public TrackingService getMyService()
54.     {
55.     return myService;
56.     }
57.
58.     public void setMyService(TrackingService myService)
59.     {
60.     this.myService = myService;
61.     }
62.
63.     @SuppressLint("NewApi")
64.     @Override
65.     protected void onCreate(Bundle savedInstanceState)
66.     {
67.     super.onCreate(savedInstanceState);
68.     setContentView(R.layout.activity_main);
69.     Bundle extras = getIntent().getExtras();
70.     //get the username for display and token for authorized requests
71.     if (extras != null)
72.     {
73.         setToken(extras.getString("token"));
74.         Username = extras.getString("Username");
75.         Log.v(TAG, Username);
76.     }
77.     viewPager = (ViewPager) findViewById(R.id.pager);
78.     actionBar = getActionBar();
79.     mAdapter = new TabsPagerAdapter(getSupportFragmentManager());
80.
81.     viewPager.setAdapter(mAdapter);
82.     actionBar.setHomeButtonEnabled(false);
83.     actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
84.     actionBar.setTitle("Welcome " + Username);
85.     for (String tab_name : tabs)
86.     {
87.         actionBar.addTab(actionBar.newTab().setText(tab_name)
88.             .setTabListener(this));
89.
90.     }
91.     viewPager.setOnPageChangeListener(new ViewPager.OnPageChangeListener()
92.     {
93.
94.         @Override
95.         public void onPageSelected(int position)
96.         {
97.         // on changing the page
98.         // make respected tab selected
99.         actionBar.setSelectedNavigationItem(position);
100.        }
101.        @Override
```

```
        public void onPageScrolled(int arg0, float arg1, int arg2)
        {
        }
        @Override
        public void onPageScrollStateChanged(int arg0)
        {
        }
    });
    }

    public void onPause()
    {
    super.onPause();
    Log.v(TAG, "onPause");

    }

    protected void onResume()
    {
    super.onResume();
    // bind (and start the service)
    Intent intent = new Intent(this, TrackingService.class);
    bindService(intent, myConnection, Context.BIND_AUTO_CREATE);

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    Log.v(TAG, "onCreateOptionsMenu");
    return true;
    }

    public void onStart()
    {
    Log.v(TAG, "onStart");
    super.onStart();


    }

    public void onStop()
    {

    super.onStop();
    Log.v(TAG, "onStop");

    }

    public void onDestroy()
    {

    super.onDestroy();
    Log.v(TAG, "onDestroy");

    //unbind and destroy service
    unbindService(myConnection);


```

```
163.
164.    }
165.
166.    @Override
167.    public boolean onOptionsItemSelected(MenuItem item)
168.    {
169.    // Handle action bar item clicks here. The action bar will
170.    // automatically handle clicks on the Home/Up button, so long
171.    // as you specify a parent activity in AndroidManifest.xml.
172.    int id = item.getItemId();
173.    if (id == R.id.action_settings)
174.    {
175.        return true;
176.    }
177.    return super.onOptionsItemSelected(item);
178.    }
179.
180.    private ServiceConnection myConnection = new ServiceConnection()
181.    {
182.    // on connection return the service to activity so we can access it
183.    // directly
184.    public void onServiceConnected(ComponentName className, IBinder service)
185.    {
186.        LocalBinder binder = (LocalBinder) service;
187.        myService = binder.getService();
188.        isBound = true;
189.    }
190.    public void onServiceDisconnected(ComponentName arg0)
191.    {
192.        isBound = false;
193.    }
194.    };
195.
196.    @Override
197.    public void onTabReselected(android.app.ActionBar.Tab arg0,
198.        android.app.FragmentTransaction arg1)
199.    {
200.    // TODO Auto-generated method stub
201.
202.    }
203.
204.    @Override
205.    public void onTabSelected(android.app.ActionBar.Tab arg0,
206.        android.app.FragmentTransaction arg1)
207.    {
208.    // TODO Auto-generated method stub
209.    viewPager.setCurrentItem(arg0.getPosition());
210.
211.    }
212.
213.    @Override
214.    public void onTabUnselected(android.app.ActionBar.Tab arg0,
215.        android.app.FragmentTransaction arg1)
216.    {
217.    // TODO Auto-generated method stub
218.
219.    }
220.
221.}
```

Αρχείο RunFragment.java

```
1.  package com.example.run_tracker;
2.  import java.util.Calendar;
3.  import java.util.HashMap;
4.  import java.util.Map;
5.  import org.json.JSONException;
6.  import org.json.JSONObject;
7.  import android.app.AlertDialog;
8.  import android.app.ProgressDialog;
9.  import android.content.BroadcastReceiver;
10. import android.content.Context;
11. import android.content.DialogInterface;
12. import android.content.Intent;
13. import android.content.IntentFilter;
14. import android.graphics.Color;
15. import android.os.Bundle;
16. import android.support.v4.app.Fragment;
17. import android.support.v4.content.LocalBroadcastManager;
18. import android.util.Log;
19. import android.view.LayoutInflater;
20. import android.view.MenuItem;
21. import android.view.View;
22. import android.view.View.OnClickListener;
23. import android.view.ViewGroup;
24. import android.widget.Button;
25. import android.widget.Chronometer;
26. import android.widget.TextView;
27. import android.widget.Toast;
28. import com.android.volley.AuthFailureError;
29. import com.android.volley.Request.Method;
30. import com.android.volley.Response;
31. import com.android.volley.VolleyError;
32. import com.android.volley.toolbox.JsonObjectRequest;
33. import com.example.run_tracker.CustomMapFragment.OnMapReadyListener;
34. import com.google.android.gms.maps.CameraUpdateFactory;
35. import com.google.android.gms.maps.GoogleMap;
36. import com.google.android.gms.maps.model.PolylineOptions;
37. import com.google.maps.android.PolyUtil;
38.
39. public class RunFragment extends Fragment implements OnClickListener,
40. OnMapReadyListener {
41.     private Button mStart, mStop;
42.     private String TAG = "RunFragment";
43.     private float ZOOM_LEVEL = 17;
44.
45.     private GoogleMap mMap;
46.     int mDistance = 0; // distance covered
47.     private TextView mDistanceText;
48.     private TextView mStatus;
49.     AlertDialog mConfirm_stop;
50.     private Timer mTimer = new Timer();;
51.     private CustomMapFragment mMapFragment;
52.     PolylineOptions options = new PolylineOptions();;
53.     private String mToken;
54.
55.     private BroadcastReceiver mMessageReceiver = new BroadcastReceiver() {@Override
56.         // When we receive e location change from TrackingService
57.         public void onReceive(Context context, Intent intent) {
58.             Log.v(TAG, "onReceive");
59.             mDistanceText.setText(((MainActivity) getActivity()).getMyService()
```

```
                .getDistance() + " m");
            // update distance and map line
            if (mMap != null) {

                options.add(((MainActivity) getActivity()).getMyService()
                    .getLastPoint());
                options.width(3);
                options.color(Color.GREEN);

                mMap.addPolyline(options);
                mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(
                ((MainActivity) getActivity()).getMyService()
                    .getLastPoint(), ZOOM_LEVEL));

            }
        }
    };
    private ProgressDialog pDialog;@Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.runfragment, container, false);
        mToken = ((MainActivity) getActivity()).getToken();
        mDistanceText = (TextView) rootView.findViewById(R.id.dst);
        mStatus = (TextView) rootView.findViewById(R.id.status);
        mStatus.setTextColor(Color.RED);
        mStop = (Button) rootView.findViewById(R.id.stop);
        Chronometer mCronometer = (Chronometer) rootView.findViewById(R.id.chronometer1);
        mTimer.setTimer(mCronometer);
        mStart = (Button) rootView.findViewById(R.id.start);
        mStart.setOnClickListener(this);
        mStop.setOnClickListener(this);
        mMapFragment = CustomMapFragment.newInstance();
        getChildFragmentManager().beginTransaction()
            .replace(R.id.myfragment, mMapFragment).commit();
        if (!mTimer.getFirstStart()) {
            mTimer.startTimer();
        }
        return rootView;
    }
    public void onPause() {
        super.onPause();
        // stop listening to TrackingService for location changes
        LocalBroadcastManager.getInstance(getActivity()).unregisterReceiver(
        mMessageReceiver);
    }
    public void onResume() {
        // when the user returns to RunFragment we must bring everything back
        // from TrackingService
        super.onResume();
        // redraw the line by getting the list of waypoints from service
        if (((MainActivity) getActivity()).getMyService() != null)

        {
            if (((MainActivity) getActivity()).getMyService().isStarted()) {
                mStart.setEnabled(false);
                // get the list of point from service and redraw the line
                options = new PolylineOptions();
                options.addAll(((MainActivity) getActivity()).getMyService()
                    .getCoursePoints());
                options.width(3);
                options.color(Color.GREEN);
```

```
121.                 mMap.addPolyline(options);
122.                 mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(
123.                 ((MainActivity) getActivity()).getMyService()
124.                     .getLastPoint(), ZOOM_LEVEL));
125.                 mDistanceText.setText(((MainActivity) getActivity())
126.                     .getMyService().getDistance() + " m");
127.             }
128.             if (((((MainActivity) getActivity()).getMyService().isStarted()))) {
129.                 mStatus.setText(R.string.Running);
130.                 mStatus.setTextColor(Color.GREEN);
131.             }
132.
133.         }
134.         // start listening to location changes
135.         LocalBroadcastManager.getInstance(getActivity()).registerReceiver(
136.         mMessageReceiver, new IntentFilter("location_changed"));
137.     }@Override
138.     public boolean onOptionsItemSelected(MenuItem item) {
139.         // Handle action bar item clicks here. The action bar will
140.         // automatically handle clicks on the Home/Up button, so long
141.         // as you specify a parent activity in AndroidManifest.xml.
142.         int id = item.getItemId();
143.         if (id == R.id.action_settings) {
144.             return true;
145.         }
146.         return super.onOptionsItemSelected(item);
147.     }
148.
149.     @Override
150.     // button click listeners
151.     public void onClick(View v) {
152.         switch (v.getId()) {
153.             case R.id.start:
154.                 {
155.                     ((MainActivity) getActivity()).getMyService().setStarted(true);
156.                     mStatus.setText(R.string.Running);
157.                     mTimer.startTimer();
158.                     mStart.setEnabled(false);
159.                     mStop.setEnabled(true);
160.                     mStatus.setTextColor(Color.GREEN);
161.                     break;
162.                 }
163.             case R.id.stop:
164.                 show_confirm();
165.                 break;
166.         }
167.
168.     }
169.
170.     @Override
171.     // when map fragment is ready take the map object from it
172.     public void onMapReady() {
173.         mMap = mMapFragment.getMap();
174.         if (mMap != null) {
175.             mMap.setMyLocationEnabled(true);
176.         }
177.
178.     }
179.
180.     // clear everything after stop is pressed
181.     private void reset_fragment() {
```

```
182.         ((MainActivity) getActivity()).getMyService().reset_all();
183.         mMapFragment.getMap().clear();
184.         mDistanceText.setText("0 m");
185.         mStatus.setText(R.string.Stoped);
186.         mStatus.setTextColor(Color.RED);
187.         mTimer.stopTimer();
188.         mStart.setEnabled(true);
189.         mStop.setEnabled(false);
190.         options = new PolylineOptions();
191.     }
192.
193.     private void save_track() {
194.         pDialog = new ProgressDialog(getActivity());
195.         pDialog.setMessage("Saving track ");
196.         showProgressDialog();
197.         String line = PolyUtil.encode(((MainActivity) getActivity())
198.             .getMyService().getCoursePoints());
199.
200.         Calendar c = Calendar.getInstance();
201.         String time = (String) mTimer.getTimer().getText();
202.         String date = c.get(Calendar.DATE) + "-" + (1 + c.get(Calendar.MONTH)) + "-
    " + c.get(Calendar.YEAR);
203.         String distance = (String) mDistanceText.getText();
204.
205.
206.         JSONObject new_run = new JSONObject();
207.         try {
208.             new_run.put("track", line);
209.             new_run.put("date", date);
210.             new_run.put("distance", distance);
211.             new_run.put("time", time);
212.         } catch (JSONException e) {
213.             e.printStackTrace();
214.         }
215.
216.         JsonObjectRequest save_req = new JsonObjectRequest(Method.POST,
217.         URLS.URL_TRACKS, new_run, new Response.Listener < JSONObject > () {
218.
219.             @Override
220.             public void onResponse(JSONObject response) {
221.
222.                 hideProgressDialog();
223.                 reset_fragment();
224.                 try {
225.                     if (response.get("error") != null) {
226.                         Toast.makeText(getActivity(), (CharSequence) response.get("error"),
227.                         Toast.LENGTH_LONG).show();
228.                     }
229.                 } catch (JSONException e) {
230.
231.                     Toast.makeText(getActivity(), "Track saved",
232.                     Toast.LENGTH_LONG).show();
233.                 }
234.             }
235.         }, new Response.ErrorListener() {
236.
237.             @Override
238.             public void onErrorResponse(VolleyError arg0) {
239.                 hideProgressDialog();
240.                 Toast.makeText(getActivity(), "Error",
241.                 Toast.LENGTH_LONG).show();
```

```
            }

        }) {

            /**
             * Passing some request headers token and content type
             * */@Override
            public Map < String, String > getHeaders() throws AuthFailureError {
                HashMap < String, String > headers = new HashMap < String, String > ();
                headers.put("Content-Type", "application/json");
                headers.put("x-access-token", mToken);
                return headers;
            }

        };
        AppController.getInstance().addToRequestQueue(save_req, TAG);
    }

    private void showProgressDialog() {
        if (!pDialog.isShowing()) pDialog.show();
    }

    private void hideProgressDialog() {
        if (pDialog.isShowing()) pDialog.hide();
    }

    private void show_confirm() {
        DialogInterface.OnClickListener dialogClickListener = new DialogInterface.OnClickListener(
    ) {@Override
            public void onClick(DialogInterface dialog, int which) {
                switch (which) {
                    case DialogInterface.BUTTON_POSITIVE:

                        save_track();
                        // Yes button clicked save and clear
                        break;

                    case DialogInterface.BUTTON_NEGATIVE:
                        // No button clicked do nothing
                        break;
                }
            }
        };
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        builder.setMessage("Are you sure?")
            .setPositiveButton("Yes", dialogClickListener)
            .setNegativeButton("No", dialogClickListener).show();
    }
}
```

Αρχείο MyRunsFragment.java

```
1.  package com.example.run_tracker;
2.
3.  import java.util.ArrayList;
4.  import java.util.HashMap;
5.  import java.util.List;
6.  import java.util.Map;
7.
8.  import org.json.JSONArray;
9.  import org.json.JSONException;
10. import org.json.JSONObject;
11.
12. import android.app.ProgressDialog;
13. import android.graphics.Color;
14. import android.os.Bundle;
15. import android.os.Parcelable;
16. import android.support.v4.app.Fragment;
17. import android.util.Log;
18. import android.view.LayoutInflater;
19. import android.view.View;
20. import android.view.View.OnClickListener;
21. import android.view.ViewGroup;
22. import android.widget.AdapterView;
23. import android.widget.AdapterView.OnItemClickListener;
24. import android.widget.Button;
25. import android.widget.ListView;
26.
27. import com.android.volley.AuthFailureError;
28. import com.android.volley.Response;
29. import com.android.volley.VolleyError;
30. import com.android.volley.VolleyLog;
31. import com.android.volley.toolbox.JsonArrayRequest;
32. import com.example.run_tracker.CustomMapFragment.OnMapReadyListener;
33. import com.google.android.gms.maps.CameraUpdate;
34. import com.google.android.gms.maps.CameraUpdateFactory;
35. import com.google.android.gms.maps.GoogleMap;
36. import com.google.android.gms.maps.model.LatLng;
37. import com.google.android.gms.maps.model.LatLngBounds;
38. import com.google.android.gms.maps.model.PolylineOptions;
39. import com.google.maps.android.PolyUtil;
40.
41. public class MyRunsFragment extends Fragment implements OnMapReadyListener,
42. OnItemClickListener, OnClickListener {
43.     private CustomMapFragment mMapFragment;
44.     private List < Track > mTracks = new ArrayList < Track > ();
45.     private CustomListAdapter adapter;
46.     private ListView listView;
47.     private String TAG = "myruns";
48.     private ProgressDialog pDialog;
49.     private GoogleMap mMap;
50.     private Button mRefresh;@Override
51.     public void onActivityCreated(Bundle savedInstanceState) {
52.         super.onActivityCreated(savedInstanceState);
53.         if (savedInstanceState != null) {
54.             mTracks = savedInstanceState.getParcelableArrayList("tracklist");
55.         }
56.     }@Override
57.     public void onSaveInstanceState(Bundle outState) {
58.         super.onSaveInstanceState(outState);
```

```
59.         out-
    State.putParcelableArrayList("tracklist", (ArrayList <? extends Parcelable > ) mTracks);
60.         // Save the fragment's state here
61.     }
62.
63.     public View onCreateView(LayoutInflater inflater, ViewGroup container,
64.     Bundle savedInstanceState) {
65.         View rootView = inflater.inflate(R.layout.myrunfragment, container,
66.         false);
67.
68.         listView = (ListView) rootView.findViewById(R.id.list);
69.         mRefresh = (Button) rootView.findViewById(R.id.refresh);
70.         listView.setOnItemClickListener(this);
71.         mMapFragment = CustomMapFragment.newInstance();
72.         getChildFragmentManager().beginTransaction()
73.             .replace(R.id.map_container, mMapFragment).commit();
74.         mRefresh.setOnClickListener(this);
75.         return rootView;
76.
77.     }
78.     public void onDestroy() {
79.         super.onDestroy();
80.         AppController.getInstance().cancelPendingRequests(TAG);
81.     }
82.     public void onStart() {
83.         super.onStart();
84.     }
85.     public void onResume() {
86.         adapt-
    er = new CustomListAdapter(this.getActivity(), mTracks, ((MainActivity) getActivity()).getToken())
    ;
87.         adapter.notifyDataSetChanged();
88.         listView.setAdapter(adapter);
89.         if (mTracks.isEmpty()) {
90.             Make_track_request();
91.         }
92.         super.onResume();
93.     }
94.     private void Make_track_request() {
95.         pDialog = new ProgressDialog(getActivity());
96.         pDialog.setMessage("Getting your tracks");
97.         showProgressDialog();
98.         // Creating volley request obj
99.         JsonArrayRequest trackReq = new JsonArrayRequest(URLS.URL_TRACKS,
100.        new Response.Listener < JSONArray > () {@Override
101.            public void onResponse(JSONArray response) {
102.                // Parsing json
103.                mTracks.clear();
104.                for (int i = 0; i < response.length(); i++) {
105.                    try {
106.                        JSONObject obj = response.getJSONObject(i);
107.                        mTracks.add(new Track(obj.getString("date"),
108.                        obj.getString("time"), obj.getString("track"), obj.getString("_id"), obj.g
    etString("distance")));
109.                    } catch (JSONException e) {
110.                        e.printStackTrace();
111.                    }
112.                }
113.                // notifying list adapter about data changes
114.                // so that it renders the list view with updated data
115.                adapter.notifyDataSetChanged();
```

```
116.                 hideProgressDialog();
117.             }
118.         }, new Response.ErrorListener() {@Override
119.             public void onErrorResponse(VolleyError error) {
120.                 VolleyLog.d(TAG, "Error: " + error.getMessage());
121.                 hideProgressDialog();
122.             }
123.         }) {
124.             /**
125.              * Passing some request headers token and content type
126.              * */@Override
127.             public Map < String, String > getHeaders() throws AuthFailureError {
128.                 HashMap < String, String > headers = new HashMap < String, String > ();
129.                 headers.put("Content-Type", "application/json");
130.                 headers.put("x-access-token", ((MainActivity) getActivity()).getToken());
131.                 return headers;
132.             }
133.         };
134.         AppController.getInstance().addToRequestQueue(trackReq, TAG);
135. 
136.     }
137. 
138.     private void showProgressDialog() {
139.         if (!pDialog.isShowing()) pDialog.show();
140.     }
141. 
142.     private void hideProgressDialog() {
143.         if (pDialog.isShowing()) pDialog.hide();
144.     }
145. 
146. 
147.     @Override
148.     // when map fragment is ready take the map object from it
149.     public void onMapReady() {
150.         mMap = mMapFragment.getMap();
151.         if (mMap != null) {
152.             mMap.setMyLocationEnabled(true);
153.         }
154.     }
155. 
156.     @Override
157.     public void onItemClick(AdapterView <? > arg0, View arg1, int position,
158.     long arg3) {
159.         PolylineOptions options = new PolylineOptions();
160.         options.width(3);
161.         options.color(Color.GREEN);
162.         List < LatLng > track = PolyUtil.decode(mTracks.get(position).getTrack());
163.         options.addAll(track);
164.         LatLngBounds.Builder b = new LatLngBounds.Builder();
165.         for (LatLng m: track) {
166.             b.include(m);
167.         }
168.         if (mMap != null) {
169.             mMap.clear();
170.             LatLngBounds bounds = b.build();
171.             CameraUpdate cu = CameraUpdateFactory.newLatLngBounds(bounds, 100,
172.             100, 1);
173.             mMap.animateCamera(cu);
174.             mMap.addPolyline(options);
175.         }
176.     }
```

```
177.
178.    @Override
179.    public void onClick(View arg0) {
180.        Make_track_request();
181.    }
182.}
```

Αρχείο ProfileFragment.java

```
1.  package com.example.run_tracker;
2.  import java.util.HashMap;
3.  import java.util.Map;
4.  import org.json.JSONException;
5.  import org.json.JSONObject;
6.  import android.app.AlertDialog;
7.  import android.app.ProgressDialog;
8.  import android.content.DialogInterface;
9.  import android.os.Bundle;
10. import android.support.v4.app.Fragment;
11. import android.text.InputType;
12. import android.util.Log;
13. import android.view.LayoutInflater;
14. import android.view.View;
15. import android.view.View.OnClickListener;
16. import android.view.ViewGroup;
17. import android.widget.Button;
18. import android.widget.EditText;
19. import android.widget.Toast;
20.
21. import com.android.volley.Request.Method;
22. import com.android.volley.AuthFailureError;
23. import com.android.volley.Response;
24. import com.android.volley.VolleyError;
25. import com.android.volley.VolleyLog;
26. import com.android.volley.toolbox.JsonObjectRequest;
27.
28. public class ProfileFragment extends Fragment implements OnClickListener {
29.     private String TAG = "profile";
30.     private ProgressDialog pDialog;
31.     private Button mSubmit;
32.     private EditText mUsername;
33.     private EditText mPassword1;
34.     private EditText mPassword2;
35.     private EditText mName;
36.     private EditText mSurname;
37.
38.     public View onCreateView(LayoutInflater inflater, ViewGroup container,
39.     Bundle savedInstanceState) {
40.
41.         View rootView = inflater.inflate(R.layout.profilefragment, container,
42.         false);
43.
44.         mSubmit = (Button) rootView.findViewById(R.id.submit);
45.         mUsername = (EditText) rootView.findViewById(R.id.username);
46.         mPassword1 = (EditText) rootView.findViewById(R.id.newpassword1);
47.         mPassword2 = (EditText) rootView.findViewById(R.id.newpassword2);
48.         mName = (EditText) rootView.findViewById(R.id.name);
49.         mSurname = (EditText) rootView.findViewById(R.id.surname);
50.         mSubmit.setOnClickListener(this);
```

```
51.         mUsername.setEnabled(false);
52.         Make_get_profile_request();
53.
54.         return rootView;
55.
56.     }
57.
58.     private void Make_get_profile_request() {
59.
60.         JSONObject credentials = null;
61.         try {
62.             credentials = new JSONObject();
63.             credentials.put("token", ((MainActivity) getActivity()).getToken());
64.
65.         } catch (JSONException e) {
66.             e.printStackTrace();
67.         }
68.
69.         JsonObjectRequest jsonObjReq = new JsonObjectRequest(Method.GET,
70.         URLS.URL_PROFILE, credentials,
71.         new Response.Listener < JSONObject > () {
72.
73.             @Override
74.             public void onResponse(JSONObject response) {
75.                 try {
76.                     mUsername.setText(response.getString("username"));
77.                     mName.setText(response.getString("name"));
78.                     mSurname.setText(response.getString("surname"));
79.
80.                 } catch (JSONException e) {
81.
82.
83.                     try {
84.                         Toast.makeText(getActivity(),
85.                         response.getString("error"),
86.                         Toast.LENGTH_LONG).show();
87.                     } catch (JSONException e1) {
88.                         e1.printStackTrace();
89.                     }
90.                     e.printStackTrace();
91.                 }
92.
93.             }
94.         }, new Response.ErrorListener() {
95.
96.             @Override
97.             public void onErrorResponse(VolleyError error) {
98.                 VolleyLog.d(TAG, "Error: " + error.getMessage());
99.             }
100.        }) {
101.            /**
102.             * Passing some request headers token and content type
103.             * */@Override
104.            public Map < String, String > getHeaders() throws AuthFailureError {
105.                HashMap < String, String > headers = new HashMap < String, String > ();
106.                headers.put("Content-Type", "application/json");
107.                headers.put("x-access-token", ((MainActivity) getActivity()).getToken());
108.                return headers;
109.            }
110.        };
111.
```

```
        // Adding request to request queue
        AppController.getInstance().addToRequestQueue(jsonObjReq, TAG);

        // Cancelling request
        // ApplicationController.getInstance().getRequestQueue().cancelAll(tag_json_obj);
    }

    // edit profile request
    private void Make_edit_profile_request(String old_password,
    String new_password) {

        JSONObject credentials = null;
        try {
            credentials = new JSONObject();
            credentials.put("password", old_password);
            credentials.put("username", mUsername.getText());
            credentials.put("password_r", new_password);
            credentials.put("name", mName.getText());
            credentials.put("surname", mSurname.getText());

        } catch (JSONException e) {
            e.printStackTrace();
        }

        JsonObjectRequest jsonObjReq = new JsonObjectRequest(Method.PUT,
        URLS.URL_PROFILE, credentials,
        new Response.Listener < JSONObject > () {

            @Override
            public void onResponse(JSONObject response) {
                try {
                    Toast.makeText(getActivity(),
                    response.getString("success"),
                    Toast.LENGTH_LONG).show();
                } catch (JSONException e) {
                    try {
                        Toast.makeText(getActivity(),
                        response.getString("error"),
                        Toast.LENGTH_LONG).show();
                    } catch (JSONException e1) {
                        // TODO Auto-generated catch block
                        e1.printStackTrace();
                    }
                    e.printStackTrace();
                }
            }
        }, new Response.ErrorListener() {

            @Override
            public void onErrorResponse(VolleyError error) {
                VolleyLog.d(TAG, "Error: " + error.getMessage());
            }
        }) {
            /**
             * Passing some request headers token and content type
             * */@Override
            public Map < String, String > getHeaders() throws AuthFailureError {
                HashMap < String, String > headers = new HashMap < String, String > ();
                headers.put("Content-Type", "application/json");
                headers.put("x-access-token", ((MainActivity) getActivity()).getToken());
                return headers;
```

```
            }
        };

        // Adding request to request queue
        AppController.getInstance().addToRequestQueue(jsonObjReq, TAG););
}

private void showProgressDialog() {
    if (!pDialog.isShowing()) pDialog.show();
}
private void hideProgressDialog() {
    if (pDialog.isShowing()) pDialog.hide();
}@Override
public void onClick(View arg0) {
    AlertDialog.Builder alert = new AlertDialog.Builder(getActivity());

    alert.setTitle("Enter Password");
    alert.setMessage("Please enter your password");

    // Set an EditText view to get user input
    final EditText input = new EditText(getActivity());
    input.setInputType(InputType.TYPE_CLASS_TEXT | InputType.TYPE_TEXT_VARIATION_PASSWORD);
    alert.setView(input);

    alert.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
            String password = input.getText().toString();
            String new_password = null;
            if (!mPassword1.getText().toString()
                .equals(mPassword2.getText().toString())) {

                Toast.makeText(getActivity(), "Passwords dont match",
                Toast.LENGTH_LONG).show();
            } else {
                if (isEmpty(mPassword1) || (isEmpty(mPassword2))) {
                    new_password = password;
                } else {
                    new_password = mPassword1.getText().toString();
                }
                Make_edit_profile_request(password, new_password);
            }
        }
    });

    alert.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
            // Canceled.
        }
    });
    alert.show();
}
private boolean isEmpty(EditText etText) {
    return etText.getText().toString().trim().length() == 0;
}
}
```

Αρχείο TrackingService.java

```
1.  package com.example.run_tracker;
2.
3.  import java.util.ArrayList;
4.  import java.util.List;
5.
6.  import android.app.Service;
7.  import android.content.Intent;
8.  import android.location.Location;
9.  import android.os.Binder;
10. import android.os.Bundle;
11. import android.os.IBinder;
12. import android.support.v4.content.LocalBroadcastManager;
13. import android.util.Log;
14.
15. import com.google.android.gms.common.ConnectionResult;
16. import com.google.android.gms.common.GooglePlayServicesClient;
17. import com.google.android.gms.location.LocationClient;
18. import com.google.android.gms.location.LocationListener;
19. import com.google.android.gms.location.LocationRequest;
20. import com.google.android.gms.maps.model.LatLng;
21.
22. public class TrackingService extends Service implements
23.     GooglePlayServicesClient.ConnectionCallbacks,
24.     GooglePlayServicesClient.OnConnectionFailedListener, LocationListener
25. {
26.     private static final String TAG = "ServiceTrackerr";
27.     private List<LatLng> mCoursePoints;
28.     private LatLng mLastPoint;
29.     // points
30.
31.     private final IBinder mBinder = new LocalBinder();
32.     private boolean mStarted = false;
33.     private int mDistance = 0;
34.     private LocationClient mLocationClient;
35.
36.     public class LocalBinder extends Binder
37.     {
38.     TrackingService getService()
39.     {
40.         return TrackingService.this;
41.     }
42.     }
43.
44.     public void onCreate()
45.     {
46.     Log.v(TAG, "onCreate");
47.     mLocationClient = new LocationClient(this, this, this);
48.     }
49.
50.     public void onDestroy()
51.     {
52.     super.onDestroy();
53.     // unbindService(myConnection);
54.     Log.v(TAG, "onDestroy");
55.     }
56.
57.     public int onStartCommand(Intent intent, int flags, int startId)
58.     {
59.     super.onStartCommand(intent, flags, startId);
```

```
60.     Log.v(TAG, "onStartCommand");
61.     /*
62.      * Log.v("LocalService", "Received start id " + startId + ": " +
63.      * intent);
64.      *
65.      * if (!mLocationClient.isConnected() ||
66.      * !mLocationClient.isConnecting()) { mLocationClient.connect(); }
67.      */
68.     return START_STICKY;
69.
70.     }
71.
72.     @Override
73.     public IBinder onBind(Intent arg0)
74.     {
75.     Log.v(TAG, "onBind");
76.     if (!mLocationClient.isConnected() && !mLocationClient.isConnecting())
77.     {
78.         mLocationClient.connect();
79.     }
80.     return mBinder;
81.     }
82.
83.     @Override
84.     public void onLocationChanged(Location arg0)
85.     {
86.     Log.v(TAG, "onLocationChanged");
87.     // add the next point to list and broadcast the message
88.     if (mStarted)
89.     {
90.         // add the new point to the list
91.         if (!mCoursePoints.isEmpty())
92.         {
93.         Location prev_point = new Location("point A"); // previous location
94.         prev_point.setLatitude(get_last_point(mCoursePoints).latitude);
95.         prev_point.setLongitude(get_last_point(mCoursePoints).longitude);
96.         // sometimes when we lose gps signal the first from second
97.         // location are 3000km apart or more
98.         // so we check this in order to be more precise
99.         if (arg0.distanceTo(prev_point) < 15)
100.        {
101.            mDistance += arg0.distanceTo(prev_point);
102.        }
103.        } else
104.        {
105.        mDistance = 0;
106.        }
107.        mCoursePoints.add(new LatLng(arg0.getLatitude(), arg0.getLongitude()));
108.        mLastPoint = new LatLng(arg0.getLatitude(), arg0.getLongitude());
109.        // broadcast message ;
110.        sendlocation();
111.    }
112.    }
113.
114.    public List<LatLng> getCoursePoints()
115.    {
116.    return mCoursePoints;
117.    }
118.
119.    public void setCoursePoints(List<LatLng> coursePoints)
120.    {
```

```
121.    mCoursePoints = coursePoints;
122.    }
123.
124.    @Override
125.    public void onConnectionFailed(ConnectionResult arg0)
126.    {
127.
128.    }
129.
130.    @Override
131.    public void onConnected(Bundle arg0)
132.    {
133.    Log.v(TAG, "onConnected");
134.
135.    }
136.
137.    @Override
138.    public void onDisconnected()
139.    {
140.    Log.v(TAG, "onDisconnected");
141.    }
142.
143.    // this method broadcasts the location change
144.    private void sendlocation()
145.    {
146.    // notify for location change anyone who listens
147.    Log.v(TAG, "sendlocation");
148.    Intent intent = new Intent("location_changed");
149.    LocalBroadcastManager.getInstance(this).sendBroadcast(intent);
150.    }
151.
152.    // does the service listen for location changes ?
153.    public boolean isStarted()
154.    {
155.    return mStarted;
156.    }
157.
158.    // start listen for location changes
159.    public void setStarted(boolean started)
160.    {
161.    Log.v(TAG, "setStarted");
162.
163.    if (mStarted != started)
164.    {
165.        mStarted = started;
166.        if (started)
167.        {
168.        // start listening to location changes
169.        mCoursePoints = new ArrayList<LatLng>();
170.        LocationRequest mLocationRequest = LocationRequest.create();
171.        mLocationRequest
172.            .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
173.        mLocationRequest.setInterval(500);// check for location change every 0.5 sec
174.        mLocationRequest.setSmallestDisplacement(5);// 5 meters min displacement
175.        mLocationClient.requestLocationUpdates(mLocationRequest, this);
176.        Log.v(TAG, "Listening for location changes");
177.
178.        }
179.    }
180.    }
181.
```

```
182.    public int getDistance()
183.    {
184.    return mDistance;
185.    }
186.
187.    public LatLng getLastPoint()
188.    {
189.    return mLastPoint;
190.    }
191.
192.    public void setLastPoint(LatLng lastPoint)
193.    {
194.    mLastPoint = lastPoint;
195.    }
196.
197.    public void setDistance(int distance)
198.    {
199.    mDistance = distance;
200.    }
201.
202.    public void reset_all()
203.    {
204.    // stop tracking and reset service
205.    Log.v(TAG, "" + mCoursePoints);
206.    mCoursePoints.clear();
207.    mDistance = 0;
208.    mStarted = false;
209.    mLocationClient.removeLocationUpdates(this);
210.    Log.v(TAG, "reset_all");
211.    Log.v(TAG, "" + mCoursePoints);
212.    }
213.
214.    private LatLng get_last_point(List<LatLng> Points)
215.    {
216.    return Points.get(Points.size() - 1);
217.    }
218.}
```