ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη εφαρμογής εντοπισμού θέσης Μέσων Μαζικής Μεταφοράς με χρήση πληθοπορισμού (crowdsourcing)

Βαρβάρα Γιώτη

Εξεταστική επιτροπή:

Καθηγητής Διονύσιος Ν. Πνευματικάτος

Αναπληρωτής Καθηγητής Μιχαήλ Γ. Λαγουδάκης

Δρ. Δημήτριος Θεοδωρόπουλος

Χανιά, Φεβρουάριος 2016

Περίληψη

Η τεχνολογική επανάσταση που παρατηρείται στις μέρες μας έχει οδηγήσει στη δημιουργία μίας πληθώρας συσκευών στον τομέα την ενημέρωσης και της επικοινωνίας. Στην κορυφή της πυραμίδας βρίσκονται τα έξυπνα κινητά τηλέφωνα, εξαιτίας του χαμηλού κόστους τους εν συγκρίσει με την ποικιλία των δυνατοτήτων που προσφέρουν. Έτσι, εκμεταλλευόμενοι την καθημερινή χρήση των κινητών τηλεφώνων, προχωρήσαμε στη δημιουργία μίας εφαρμογής Android για τον εντοπισμό της θέσης των μέσων μαζικής μεταφοράς(λεωφορεία, τρένα, πλοία κ.α.) με χρήση πληθοπορισμού. Μέσω της εφαρμογής υλοποείται ένας ειδικός αλγόριθμος που αναγνωρίζει την επιβίβαση ή αποβίβαση κάποιου, χρησιμοποιώντας παραμέτρους όπως η ταχύτητα, η θέση του κ.α..Συγκεκριμένα, η καινοτομία βρίσκεται στο γεγονός ότι ο χρήστης δε χρειάζεται να προβεί σε κάποια επιπλέον ενέργεια κατά την επιβίβασή του σε ένα μέσο μεταφοράς, καθώς η διαδικασία αναγνώρισης είναι αυτοματοποιημένη και το μόνο που ζητείται από εκείνον είναι η δήλωση του λεωφορείου στο οποίο βρίσκεται.

Στη συνέχεια, η κινητή συσκευή επικοινωνεί με έναν απομακρυσμένο εξυπηρετητή, παίρνοντας πληροφορίες για το εν λόγω μεταφορικό μέσο – στην περίπτωσή μας λεωφορείο – και στέλνει πίσω την τοποθεσία του επιβάτη και κατ' επέκταση του λεωφορείου εκείνη τη στιγμή. Συμπερασματικά, η εξοικείωση με την τεχνολογία και τους ηλεκτρονικούς χάρτες κι η παραδοχή ότι ο πλανήτης μας είναι πλέον ένα αναπαραστατικό πεδίο εικονικής πληροφορίας, συμβάλλουν στην ευχρηστία της εφαρμογής. Με αυτόν τον τρόπο, καλλιεργείται η συλλογικότητα και ο αλτρουισμός των ατόμων μέσω του πληθοπορισμού, καλυτερεύει η εμπειρία των επιβατών κι ενισχύεται η χρήση των μέσω μαζικής μεταφοράς, που είναι σημαντική για την ανάπτυξη περιβαλλοντικής συνείδησης, καθώς η εκτεταμένη χρήση των ιδιωτικών οχημάτων οδηγεί σε ραγδαίες κλιματικές αλλαγές.

TECHNICAL UNIVERSITY OF CRETE, GREECE
SCHOOL OF ELECTRONIC AND COMPUTER ENGINEERING

Crowdsourcing Android Application for Public Transport Position Tracking

Varvara Gioti

Thesis Committee:

Professor Dionisios N. Pnevmatikatos

Associate Professor Michail G. Lagoudakis

Dr. Dimitrios Theodoropoulos

Chania, February 2016

Abstract

Nowadays, many communication and information devices have been created as a result of technological revolution. Low cost as well as wide variety of features have brought smart phones in the spotlight as one of the most widely used device of all time. Considering the daily usage of mobile phones, we developed a mobile application that can track the location of any mean of transport (such as bus, train, boat etc) using the technique of crowdsourcing.The application implements a special algorithm that automatically recognizes when a passenger gets on or off a bus (in our case), using smartphone's parameters such as GPS-based location or speed. The innovation here is that the passenger does not have to make any effort saying when he gets on or off; the whole procedure is totally automated and the only thing that he is asked to do is to give the exact number of the bus he is on.

Subsequently, the mobile device communicates with a remote server, receives information about the buses and sends back the user's location. Then, the server can distribute current location to other users of the application that might be waiting for the bus. In summary, rapid technology development and especially familiarity with online maps contributes to the usability of this application. Thus, crowdsourcing helps cultivating altruism. Moreover, passengers seem more eager to use buses when they don't have to wait for too long or in vain for them, which in turns means fewer cars in the streets and less pollution from these cars. That is what we need to strive for; make people respect and stop destroying the environment.

Στον πατέρα μου...

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον κ. Δ. Πνευματικάτο για τις πολύτιμες συμβουλές και την ευκαιρία να έχω ένα από τα πιο ενδιαφέροντα θέματα διπλωματικής εργασίας. Τον κ. Μ. Λαγουδάκη για την ανάγνωση του κειμένου, τις διορθώσεις, την αμέριστη στήριξή του όλα αυτά τα χρόνια και για τις πολύτιμες συμβουλές που μου έδωσε καθόλη τη διάρκεια της φοιτητικής μου ζωής.

Θέλω να πω ένα τεράστιο ευχαριστώ στον κ. Δημήτρη Θεοδωρόπουλο, για τη συνέπειά του στις αμέτρητες συναντήσεις όλους αυτούς τους μήνες, την υπομονή του καθώς και τον ενθουσιασμό και τη διαρκή διάθεσή του να με καθοδηγήσει και να λύσει οποιαδήποτε απορία μου.

Ακόμη, θέλω να ευχαριστήσω τη μητέρα μου που με στήριξε τόσο πολύ τα τελευταία χρόνια, την αδερφή μου Εύα που πάντα ήταν δίπλα μου, τον Γιώργο για τις ατάκες του, τον Simon για την υπομονή του αλλά και όλους μου τους φίλους στα Χανιά για τις όμορφες στιγμές που περάσαμε μαζί.

Τέλος, θέλω να ευχαριστήσω το πρόσωπο στο οποίο οφείλω την αγάπη μου για τον προγραμματισμό, Γιώργο Μπ.

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Γλωσσάρι

ADT Android Developer Tools - Εργαλεία Προγραμματιστή Android. 22

Android είναι λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας το οποίο αναπτύχθηκε από την Google και τρέχει τον πυρήνα του λειτουργικού Linux . 10, 22--25, 31, 34, 35, 37, 40, 41, 47--49, 51

API Application Programming Interface - Διεπαφή Προγραμματισμού Εφαρμογών. 51

GPS Global Positioning System - Παγκόσμιο Σύστημα Στιγματοθέτησης. 7, 16, 24, 25, 37--39, 42, 43, 48, 52--55

IDE Integrated Development Environment - Ολοκληρωμένο Περιβάλλον Ανάπτυξης. 21, 22

SDK Software Development Kit - Κιτ Ανάπτυξης Λογισμικού. 21

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρα

Η εποχή μας θα μπορούσε να χαρακτηριστεί ως η εποχή της επικοινωνίας. Η αλματώδης τεχνολογική επανάσταση που παρατηρείται στις μέρες μας, οδήγησε στην επιτυτάχυνση του ρυθμού ζωής του σύγχρονου ανθρώπου, κηρύσσοντας αναγκαία την προσπάθεια να απλοποιηθούν και να κατακερματιστούν πολλές από τις καθημερινές συνήθειες, ώστε να διευκολυνθεί κατά κόρον η ζωή του σύγχρονου ατόμου. Εξαιτίας των παραπάνω, έχουμε τη δημιουργία νέων συσκευών στον τομέα της ενημέρωσης αλλά και της επικοινωνίας. Μερικά παραδείγματα τέτοιων συσκευών είναι η τηλεόραση, το διαδίκτυο και πλέον και τα κινητά τηλέφωνα.

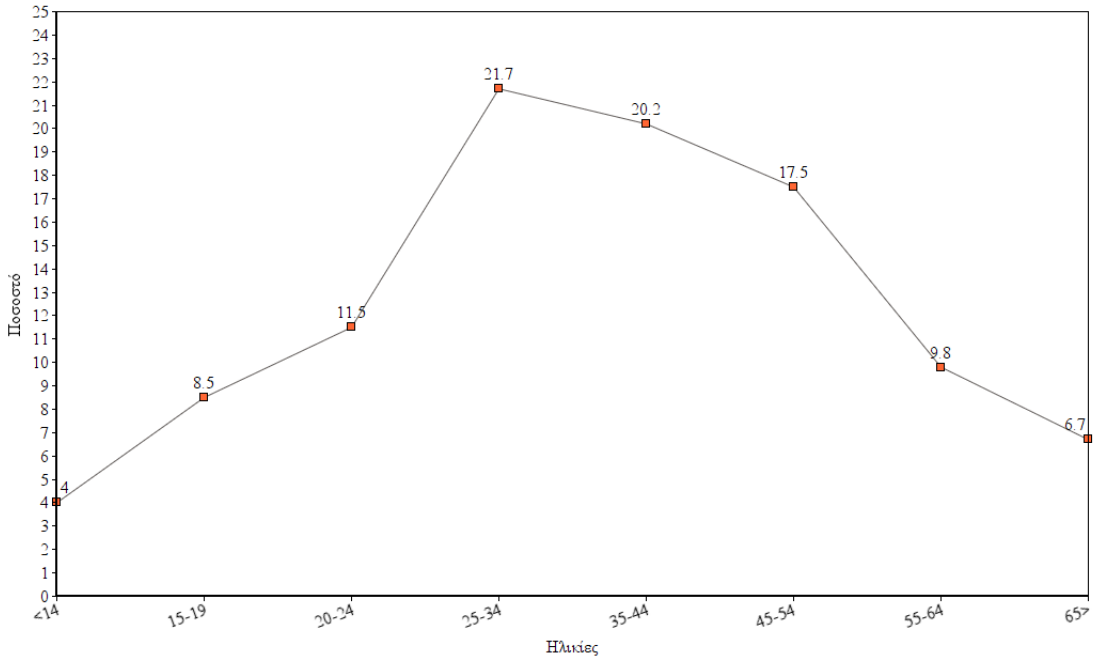
Η πληθώρα των λειτουργιών καθώς και το χαμηλό κόστος του κινητού τηλεφώνου συνέβαλαν στην καθιέρωσή του ως το πιο απαραίτητο εργαλείο της καθημερινότητας του σύγχρονου ανθρώπου. Πλέον, ο καθένας μας διαθέτει ένα "έξυπνο" - όπως αποκαλείται - κινητό τηλέφωνο, μέσω του οποίου μπορεί να κάνει σχεδόν τα πάντα δίχως κανέναν περιορισμό. Συγκεκριμένα, είναι σε θέση ανά πάσα ώρα και στιγμή να παρακολουθήσει το αγαπημένο του κανάλι, να διαβάσει την αγαπημένη του εφημερίδα, να μοιραστεί προσωπικές στιγμές με τους φίλους του, να παραγγείλει τα εβδομαδιαία του ψώνια ή να ενημερωθεί για τον καιρό στην άλλη άκρη του πλανήτη.

Βλέποντας ότι οι ανάγκες του σύγχρονου πολυπαραγωγικού ατόμου είναι πολλές και τα κινητά τηλέφωνα είναι, αν μη τι άλλο, στην καθημερινότητά μας για να λύνουν τα προβλήματά μας, αποφασίσαμε τη δημιουργία μίας εφαρμογής που θα ενθαρρύνει τη

χρήση μέσων μαζικής μεταφοράς και θα επωφελήσει τον άνθρωπο τόσο βραχυπρόθεσμα όσο και μακροπρόθεσμα. Όλοι μας γνωρίζουμε ότι οι κλιματικές αλλαγές και οι επιπτώσεις τους στο κοινωνικό σύνολο, εξαιτίας της εκτεταμένης χρήσης των αυτοκινήτων, είναι πλέον επιστημονικώς ακλόνητες.

Δυστυχώς, όμως, η ποιότητα των υπηρεσιών που προσφέρουν τα μέσα μαζικής μεταφοράς στη χώρα μας δεν είναι κι η ιδανικότερη. Όσοι στην καθημερινότητά τους χρησιμοποιούν το λεωφορείο ως μέσο μεταφοράς, τονίζουν ότι συνήθως βιάζονται να κατευθυνθούν στη στάση κι αναγκάζονται να περιμένουν πολλές φορές για διαστήματα μεγαλύτερα της μίας ώρα, ή χάνουν το λεωφορείο εξαιτίας μερικών λεπτών, καθώς δεν μπορούν να υπολογίσουν με ακρίβεια εξωτερικούς παράγοντες, όπως την κίνηση των δρόμων κ.α.. Η λύση που προσφέρεται μέσω της εφαρμογής είναι η δυνατότητα προβολής της ακριβούς θέσης του λεωφορείου στο κινητό τηλέφωνο του καθενός, δίνοντας τέλος στις ατελείωτες αναμονές στη στάση. Δε θα ήταν αυτός, λοιπόν, ένας ικανός λόγος να κάνει τον καθένα μας να αρχίσει να χρησιμοποιεί τα μέσα μετοφοράς περισσότερο, αφήνοντας το προσωπικό του όχημα στο σπίτι;

Μελετώντας τη χρήση των μέσων μαζικής μεταφοράς που γίνεται ως τώρα και βλέποντας αντίστοιχες έρευνες, στην εικόνα 1.1 βλέπουμε ότι το μεγαλύτερο ποσοστό των επιβατών είναι νέοι, ηλικίας 25-34 ετών [1]. Ξέρουμε, όμως, ότι αυτές οι ηλικίες έχουν παράλληλα και μεγάλη ευχέρεια με τη χρήση έξυπνων κινητών τηλεφώνων, όπου πλέον αποτελούν αναπόσπαστο κομμάτι της καθημερινότητάς τους και σπανίως τα αποχωρίζονται.



Σχήμα 1.1: Ηλικίες επιβατικού κοινού μέσων μαζικής μεταφοράς

1.2 Συνεισφορά

Όλα τα παραπάνω, λοιπόν, μάς οδήγησαν στη δημιουργία του BusFinder. Το BusFinder είναι μία εφαρμογή που βασίζεται στον πληθοπορισμό (crowdsourcing) [2] και δείχνει κάθε φορά που βρίσκεται το λεωφορείο σε πραγματικό χρόνο πάνω στον χάρτη [3--5]. Με την έννοια του πληθοπορισμού, εννοούμε την παροχή των απαραίτητων πληροφορίων, για την ομαλή λειτουργία την εφαρμογής, από τους ίδιους τις χρήστες της [6]. Συγκεκριμένα, δίχως να έχουμε κάποιον επιπλέον εξοπλισμό μέσα στο λεωφορείο και με μηδενικό κόστος, ο χρήστης θα μπορεί να δει που βρίσκεται το λεωφορείο, μέσω των επιβατών/ χρηστών της εφαρμογής [7].

Ο καθένας μας θα μπορεί να συνεισφέρει στους συμπολίτες του χρησιμοποιώντας την εφαρμογή όταν βρίσκεται μέσα σε κάποιο μέσο μεταφοράς ή να ανοίξει την εφαρμογή για να πληροφορηθεί για τις θέσεις των λεωφορείων που των ενδιαφέρουν. Μέσω του BusFinder, λοιπόν, προσφέρονται τα παρακάτω.

- Δωρεάν κι άμεση ενημέρωση για την ακριβή θέση ενός λεωφορείου πάνω στον χάρτη, σε πραγματικό χρόνο.

- Θα σταματήσει η άσκοπη και κουραστική αναμονή του λεωφορείου στη στάση.
- Υλοποιεί έναν ειδικό αλγόριθμο για την αναγνώριση της επιβίβασης κι αποβίβασης, χρησιμοποιώντας παραμέτρους της συσκευής όπως η ταχύτητα, η απόσταση από τη διαδρομή ενός λεωφορείου κ.λπ.
- Δε χρειάζεται πλέον να ψάχνει κάποιος τα δρομολόγια ενός λεωφορείου, να ενημερώνεται διαρκώς για αλλαγές που γίνονται στις ώρες κ.λπ.
- Θα σταματήσει κάποιος να διερωτάται διαρκώς και να ανησυχεί αν είναι αργία, αν έχει λεωφορείο τόσο αργά, κ.ο.κ.
- Ενισχύεται το αίσθημα σεβασμού προς το περιβάλλον, μέσω της ενθάρρυνσης της χρήσης των μέσων μαζικής μεταφοράς.
- Καλλιεργείται η συλλογικότητα και το αίσθημα αλληλοπροσφοράς, καθώς η εφαρμογή βασίζεται εξ' ολοκλήρου στη διάθεση των χρηστών να συμβάλλουν, μοιράζοντας τη θέση τους.
- Απευθύνεται σε χρήστες κινητών τηλεφώνων, που είναι κι η πλειοψηφία των επιβατών, σύμφωνα με έρευνες που έγιναν [1]· τονίζεται ότι τα άτομα ηλικίας 18-36 ετών είναι εκείνα που χρησιμοποιούν τα μέσα μεταφοράς περισσότερο από τις υπόλοιπες ηλικίες.

1.3 Διάρθρωση κειμένου

Σε αυτή την εργασία θα αναλύσουμε τα στάδια για τη δημιουργία της εφαρμογής, καθώς και θα ανεφερθούμε με περισσότερες λεπτομέρειες στην υλοποίησή της και στα διάφορα εργαλεία που χρειάστηκαν για τη δημιουργία της. Θα παρουσιαστούν έξι (6) ξεχωριστά Κεφάλαια, όπου στο καθένα θα αναφερθούν τα διαφορετικά στάδια ανάπτυξης κι υλοποίησης της εφαρμογής. Συγκεκριμένα, στο Κεφάλαιο 1, στο οποίο μιλήσαμε προηγουμένως, παρουσιάζεται μία μικρή εισαγωγή για τους λόγους και τις ανάγκες που οδήγησαν στη δημιουργία αυτής της εφαρμογής καθώς και τι τελικά προσφέρει η χρήση της γενικά στο άτομο. Στο δεύτερο Κεφάλαιο θα αναφερθούμε σε διάφορες άλλες επαγγελματικές και μη λύσεις που υπάρχουν αυτή τη στιγμή στην αγορά, τι προσφέρουν και

ποιες ανάγκες καλύπτουν. Στο τρίτο Κεφάλαιο θα αναφερθούμε στην αρχιτεκτονική της εφαρμογής, δηλαδή στα πρώτα βήματα που ακολουθήσαμε για να σχεδιάσουμε την εφαρμογή, τις απαιτήσεις και τη βασική της λειτουργικότητα. Στο τέταρτο Κεφάλαιο προχωρήσαμε στην ανάπτυξη της εφαρμογής. Εκεί, θα παρουσιαστεί η μέθοδος ανάπτυξης που χρησιμοποιήθηκε, ο τρόπος εργασίας για την υλοποίηση της εφαρμογής, θα παρουσιαστούν τα εργαλεία που βοήθησαν στην ανάπτυξή της καθώς και θα περιγραφεί πλήρως η δημιουργία του BusFinder. Στο πέμπτο Κεφάλαιο θα παρουσιαστεί ο τρόπος λειτουργίας της εφαρμογής δείχνοντας και κατάλληλες εικόνες και θα γίνει εκτενής αναφορά σε όλους τους ελέγχους και τις δοκιμές που έγιναν. Στο έκτο και τελευταίο Κεφάλαιο θα μιλήσουμε για τα συμπεράσματα στα οποία καταλήξαμε μετά την υλοποίηση της εν λόγω εφαρμογής και τέλος θα αναφερθούμε σε διάφορες προτάσεις για τη μελλοντική βελτίωση του BusFinder.

Κεφάλαιο 2

Σχετικές εργασίες - λύσεις

2.1 Γενικά

Με την πάροδο του χρόνου και την ανάπτυξη της τεχνολογίας, δημιουργήθηκαν νέες ανάγκες τόσο στην καθημερινότητα του καθενός όσο και στον επειχειρηματικό κλάδο. Από μεγάλες επιχειρήσεις οι οποίες έχουν ως κύριο μέλημά τους τη μεταφορά προϊόντων μεγάλης αξίας ως την άφιξη ενός λεωφορείου στην ώρα του, είναι μερικές από τις ανάγκες που οδήγησαν στη δημιουργία ενός νέου τομέα που ονομάζεται "διαχείριση στόλου" (fleet management).

Έτσι, δημιουργήθηκαν μεγάλες εταιρίες που προσφέρουν την υπηρεσία της διαχείρισης στόλου άλλων επιχειρήσεων κι υπόσχονται πλήρη και άμεση ενημέρωση του υπευθύνου για την πορεία των οχημάτων σε πραγματικό χρόνο. Ακόμη, ο εντοπισμός ενός οχήματος είναι άμεσα αναγκαίος και στις συγκοινωνίες. Εκεί, βλέπουμε και πάλι μεγάλες εταιρίες που αναλαμβάνουν εξ' ολοκλήρου τη διαχείριση των λεωφορείων και την παροχή αυτών των υπηρεσιών αλλά επίσης και μικρότερους φορείς που παρουσιάζουν λύσεις στον τομέα αυτόν [8].

Τα τελευταία χρόνια, όμως, όπου η χρήση των κινητών τηλεφώνων είναι στην καθημερινότητα όλων των ανθρώπων, δημιουργήθηκαν διάφορες εφαρμογές οι οποίες προσφέρουν αυτή τη δυνατότητα στους χρήστες, εκμεταλλευόμενοι τον δέκτη GPS [9] που βρίσκεται εγκατεστημένος στα κινητά νέας τεχνολογίας [10].

Ας δούμε παρακάτω μερικές από τις μεγαλύτερες εφαρμογές που χρησιμοποιούνται αυτή τη στιγμή για διαχείριση στόλου.

2.2 Εφαρμογές διαχείρισης στόλου

2.2.1 Track Your Truck

Σχήμα 2.1: Λογότυπο Track Your Truck

Πρόκειται για μία επαγγελματική λύση όπου με χρήση ειδικών συσκευών προσφέρει στους χρήστες έλεγχο των οχημάτων τους. Διαθέτει τρεις διαφορετικές συσκευές, αναλόγως με τις ανάγκες του κάθε χρήστη. Η εφαρμογή του κινητού τηλεφώνου διατίθεται δωρεάν. Για όλες τις άλλες υπηρεσίες ο χρήστης πρέπει να πληρώνει μηνιαία συνδρομή. [11]

2.2.2 Transit Tracking

Παρέχει πληροφορίες για τη θέση οχημάτων και μπορεί να καλύψει τις ανάγκες του στόλου μεγάλου επιχειρήσεων. Υποστηρίζει επιπλέον λειτουργίες πέραν της καταγραφής της τοποθεσίας, όπως καταγραφή ταχύτητας, αναφορά ταξιδιού, εμφάνιση όλων των οχημάτων στον χάρτη κ.α.. [12]

Σχήμα 2.2: Λογότυπο Transit Tracking

2.2.3 Live London Bus Tracker

Σχήμα 2.3: Λογότυπο εφαρμογής Live London Bus Tracker

Εφαρμογή που εξυπηρετεί μόνο την πόλη του Λονδίνου, Αγγλία, παρέχοντας πληροφορίες για αφίξεις λεωφορείων, παίρνοντας πληροφορίες από το εσωτερικό δίκτυο των λεωφορείων. Με αυτό τον τρόπο, η εφαρμογή αυτή, είναι πιο αξιόπιστη από άλλες αντίστοιχες - crowdsourcing - εφαρμογές που εκμεταλλεύονται τους επιβάτες του εκάστοτε λεωφορείου. Η χρήση του Live London Bus Tracker είναι εντελώς δωρεάν. Βέβαια, δε λαμβάνεται υπόψιν πόσο στοιχίζει ο εξοπλισμός που διαθέτουν τα ίδια τα λεωφορεία ήδη. [13]

2.2.4 Live Bus Tracker

Είναι διαθέσιμο στις Ηνωμένες Πολιτείες Αμερικής καθώς και στην Ινδία. Ενημερώνει για την τοποθεσία ενός οχήματος καθώς και παρέχει και άλλες υπηρεσίες γύρω από αυτόν τον τομέα, όπως προβολή διαθέσιμων δρομολογίων, ενημέρωση για τυχόν ακυρώσεις κ.α.. Η χρήση του δεν είναι δωρεάν· ο πελάτης πρέπει να πληρώσει για όλο το πρόγραμμα. Δεν έχουν δημιουργήσει εφαρμογή για κινητή συσκευή όπου δείχνουν στον χρήστη την τοποθεσία του λεωφορείου, μόνο μέσω browser μπορεί κάποιος να ενημερωθεί γι'αυτό. [14]

Σχήμα 2.4: Λογότυπο Live Bus Tracker

2.2.5 Vodafone M2M Control

Το πρόγραμμα Ready Business της Vodafone, παρέχει υπηρεσίες fleet control. Συγκεκριμένα, για ένα συμβόλαιο 18 μηνών με κι επιδότηση συσκευής, η Vodafone παρέχει ένα πακέτο υπηρεσιών για την καταγραφή τοποθεσίας των οχημάτων διάφορων επιχειρήσεων. Το συνολικό κόστος για 1.5 χρόνο χρήσης της υπηρεσίας ανέρχεται στα 449.82 ευρώ[1]. Ακόμη, δίνει τη δυνατότητα κάποιον επιπλέον λειτουργιών, όπως καταγραφή ποσοστού καυσίμων, καταγραφή θερμοκρασίας ψυγείων κ.α.. [15]

Σχήμα 2.5: Λογότυπο εταιρίας Vodafone

2.2.6 Wind Fleet Management

Η Wind παρέχει τη δυνατότητα διαχείρισης του στόλου μίας επιχείρησης. Μέσω ειδικού εξοπλισμού που εγκαθίσταται στα οχήματα της εταιρίας μπορεί να καταστεί διαθέσιμος ο έλεγχος της γεωγραφικής θέσης των οχημάτων και ό,τι αφορά αυτή ανά πάσα στιγμή. Το κόστος για την εγκατάσταση της προηγμένης συσκευής είναι 62 ευρώ κι η ίδια η συσκευή στοιχίζει 196 ευρώ[1]. Παρέχει τρία (3) ξεχωριστά προγράμματα για του επιχειρηματίας που ενδιαφέρονται, με τιμές από 15 ευρώ μηνιαίως

Σχήμα 2.6: Λογότυπο εταιρίας Wind

[1]Όλες οι τιμές αναφέρονται εν έτει 2016

έως 36 ευρώ. [16]

2.2.7 Moovit

Σχήμα 2.7: Λογότυπο εφαρμογής Moovit

Είναι μία εφαρμογή για έξυπνα κινητά τηλέφωνα. Υποστηρίζει όλα τα λειτουργικά κινητών συσκευών, Android, IOS και Windows Mobile και χωρίς επιπλέον εξοπλισμό, παρέχει πληροφορίες για λεωφορεία μίας πόλης καθώς και για τις πιθανές ώρες των αφίξεών τους. Διαθέσιμο σε αρκετές χώρες, μέσα σε αυτές κι η Ελλάδα (Αθήνα, Ιωάννινα, Πάτρα). Διανέμεται εντελώς δωρεάν και οι πληροφορίες δίδονται στον χρήστη μέσω της βοήθειας άλλων χρηστών (crowdsourcing). [17]

2.2.8 Link Technologies

Πρόκειται για μία εφαρμογή κινητών τηλεφώνων. Υποστηρίζει όλες τις συσκευές και όλα τα λειτουργικά συστήματα. Η εφαρμογή διατίθεται δωρεάν στο κοινό. Οι πληροφορίες, όμως, που παίρνει βασίζονται σε μία συσκευή που έχει εγκατασταθεί στο κάθε λεωφορείο. Το κόστος μίας κυμαίνεται από 50-200 ευρώ κι έχει εγκατασταθεί από μία σε κάθε λεωφορείο. Παρέχονται πληροφορίες όπως άφιξη λεωφορείου, θέση του πάνω στον χάρτη καθώς και όλες οι διαθέσιμες στάσεις. [18]

Σχήμα 2.8: Λογότυπο εταιρίας Link Technologies

2.3 Πίνακας Σύγκρισης

Μπορούμε εύκολα, λοιπόν, να διαπιστώσουμε ότι υπάρχουν πολλές διαφορετικές εφαρμογές που ασχολούνται σε αυτό το πεδίο. Πολλές από αυτές δίνουν έμφαση στην ακρίβεια των δεδομένων που παρέχουν, καθώς πρόκειται για τελείως επαγγελματικές προτάσεις και άλλες ενδιαφέρονται περισσότερο για τη δωρεάν διάθεση της εφαρμογής στο κοινό. Αναφέρθηκαν ενδεικτικά μερικές από όλες τις περιπτώσεις όπου γίνεται εκμετάλλευση της τοποθεσίας ενός οχήματος, είτε άμεσα είτε έμμεσα. Όταν λέμε άμεση

Όνομα Εφαρμογής	Κόστος	Επιπλέον Εξοπλισμός	Διαθεσιμότητα	Ακρίβεια	Γεωγραφική Κάλυψη
Track Your Truck	Υψηλό	ΝΑΙ	24/7	Υψηλή	Η.Π.Α.
Transit Tracking	Υψηλό	ΝΑΙ	24/7	Υψηλή	Η.Π.Α.
Live London Bus Tracker	Δωρεάν	ΟΧΙ	24/7	Μέτρια	Λονδίνο, Αγγλία
Live Bus Tracker	Μέτριο	ΝΑΙ	24/7	Υψηλή	Η.Π.Α. & Ινδία
Vodafone M2M Control	Μηνιαία Συνδρομή	ΝΑΙ	24/7	Υψηλή	Ελλάδα
Wind Fleet Management	Μηνιαία Συνδρομή + Εξοπλισμός	ΝΑΙ	24/7	Υψηλή	Ελλάδα
Moovit	Δωρεάν	ΟΧΙ	Μέτρια	Μέτρια	Παγκοσμίως[2]
Link Technologies	Υψηλό	ΝΑΙ	24/7	Υψηλή	Ελλάδα[3]

Πίνακας 2.1: Συγκριτικός πίνακας άλλων εφαρμογών

καταγραφή θέσης ενός οχήματος εννοούμε τον επιπλέον εξοπλισμό που εγκαθίσταται στο εν λόγω όχημα και καταγράφει διαρκώς τη θέση του. Από την άλλη, όταν λέμε εμμέσως εννοούμε την καταγραφή της θέση ενός επιβάτη του συγκεκριμένου οχήματος, επομένως κατ' επέκταση και τη θέση του οχήματος.

Στον πίνακα 2.1 βλέπουμε ότι στις εφαρμογές όπου το κόστος είναι υψηλό και παρέχεται κι επιπλέον εξολισμός, η ακρίβεια είναι υψηλή αλλά κι η παροχή υπηρεσιών είναι αδιάκοπη. Έτσι, με ένα σχετικά υψηλό κόστος επιτυγχάνεται συνεχής κάλυψη. Ενώ, αν δούμε τις εφαρμογές που διατίθονται δωρεάν, τότε θα δούμε ότι ο χρήστης μπορεί να λαμβάνει πληροφορίες για τη θέση ενός λεωφορείου μόνο όταν κάποιος άλλος χρήστης βρίσκεται μέσα στο λεωφορείο. Βλέπουμε, δηλαδή, ότι περιορίζονται οι λειτουργίες μίας εφαρμογής που παρέχεται δωρεάν.

Γίνεται σαφές, λοιπόν, ότι η αδιάκοπη διαθεσιμότητα καθ'όλη τη διάρκεια της ημέρας, όλες τις ημέρες της εβδομάδας, είναι μία υπηρεσία που ενδιαφέρει περισσότερο τους χρήστες σε επαγγελματικό επίπεδο. Όταν πρόκειται για μία εφαρμογή όπου δείχνει που βρίσκεται το λεωφορείο, με κυριότερο στόχο την εξυπηρέτηση του επιβατικού κοινού, τότε θα μπορούσαν να γίνουν κάποιες υποχωρήσεις, όσον αφορά την ακρίβεια και την διαθεσιμότητα, ώστε να δημιουργηθεί μία εφαρμογή διαθέσιμη σε όλους εντελώς δωρεάν.

Βλέποντας τα παραπάνω αλλά και τις ανάγκες της τοπικής κοινότητας των φοιτητών της πόλης των Χανίων, διαπιστώσαμε ότι ήταν αναγκαία η δημιουργία μίας εφαρμογής

[2]Εξυπηρετεί 800 πόλεις σε 60 πόλεις (2016).

[3]Συγκεκριμένες πόλεις της Ελλάδας καθώς και το Bus Rapid Transport σύστημα της Μπανγκοκαι στην Ταϋλάνδη (2016).

όπου να εξυπηρετεί σε διαρκή βάση τους φοιτητές του Πολυτεχνείου Κρήτης. Η μεγαλύτερη πρόκληση, όμως, ήταν πως θα καταφέρουμε να προσφέρουμε όσο το δυνατόν περισσότερο αξιόπιστες πληροφορίες στους χρήστς με μηδενικό κόστος. Λαμβάνοντας αυτά τα δεδομένα υπόψιν, προχωρήσαμε στην ανάπτυξη του BusFinder. Μία εφαρμογή όπου βασίζεται μόνο στην προσφορά των χρηστών της εφαρμογής και στην δική τους θέληση να βοηθήσουν όσο περισσότερο μπορούν την κοινότητά τους.

Συγκεκριμένα, το BusFinder βασίζεται στην ιδέα του πληθοπορισμού (crowdsourcing), όπου χρήσιμες πληροφορίες για την ανάπτυξη μιας εφαρμογής δίνουν οι ίδιοι οι χρήστες της. Έτσι, παίρνοντας τη θέση των χρηστών όταν βρίσκονται μέσα σε κάποιο λεωφορείο, θα είμαστε σε θέση να ξέρουμε και που βρίσκεται αυτό το λεωφορείο εκείνη τη στιγμή.

Κεφάλαιο 3

Αρχιτεκτονική - Σχεδίαση

3.1 Γενικά

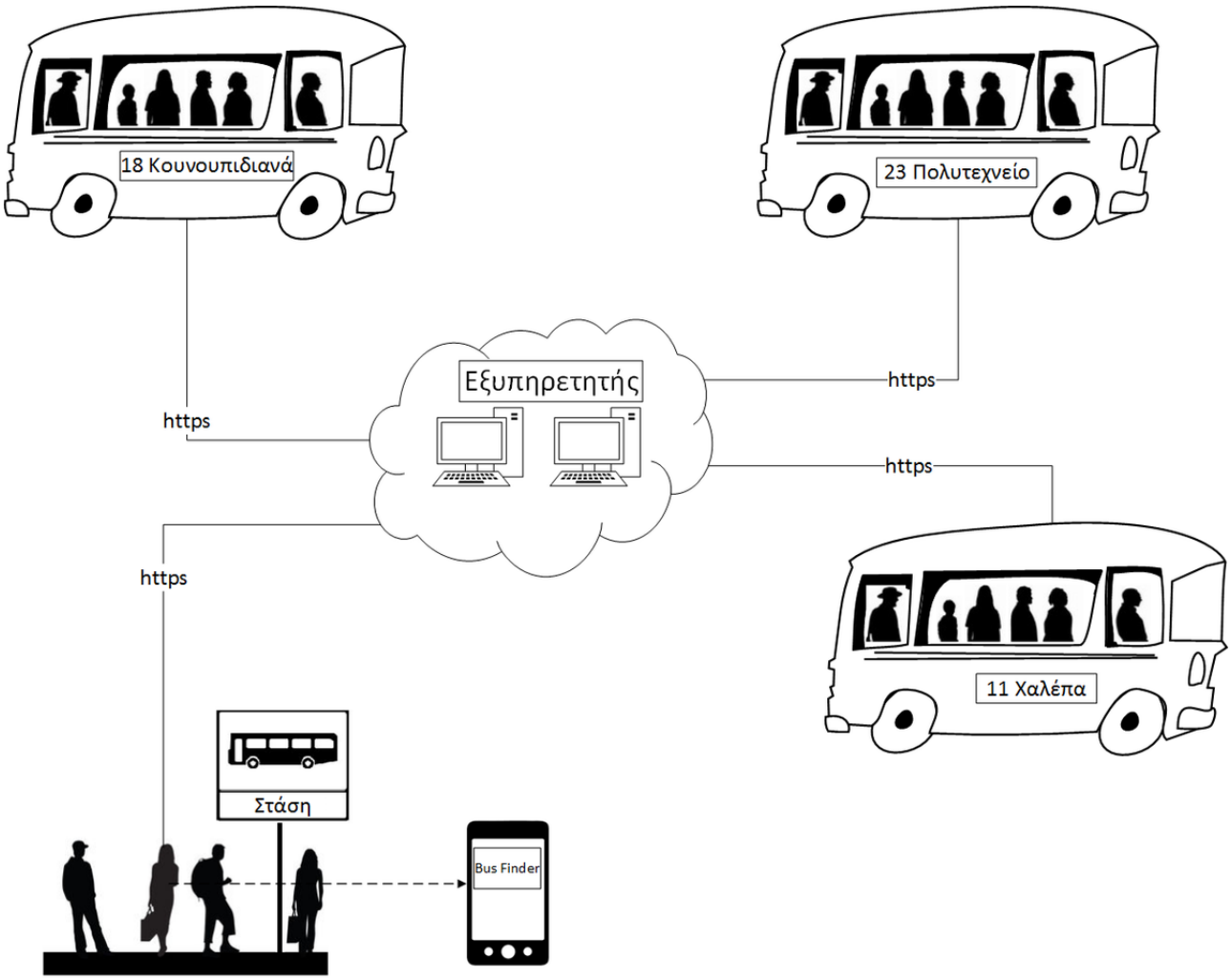
Στο κεφάλαιο αυτό θα δείξουμε τα βήματα που ακολουθήσαμε για την ανάπτυξη της εφαρμογής. Συγκεκριμένα, θα μιλήσουμε για τα πρώτα στάδια ανάπτυξής της, για τον τρόπο που έγινε ο καταμερισμός του όγκου εργασίας και τέλος για τον προσδιορισμό των απαιτήσεων, ώστε να καταλήξουμε σε μία λειτουργική εφαρμογή εν τέλει.

3.2 Σχεδιασμός

Σε πρώτο στάδιο, για την ανάπτυξη της εν λόγω εφαρμογής, χρειάζεται η δημιουργία επικοινωνίας της συσκευής μας με έναν απομακρυσμένο εξυπηρετητή, από όπου είτε θα παίρνουμε κάποια δεδομένα χρήσιμα για εμάς, είτε θα στέλνουμε κάποια αντιστοίχως χρήσιμα δεδομένα από τη συσκευή μας προς τον διακομιστή. Στην πληροφορική, αυτός ο τρόπος επικοινωνίας είναι ευρέως γνωστός ως μοντέλο πελάτη - εξυπηρετητή. Αν θέλουμε να αναπαραστήσουμε σχηματικά αυτή την επικοινωνία, τότε θα έδειχνε όπως η εικόνα του Σχήματος 3.1.

Αναλυτικότερα, στο σχήμα 3.1 βλέπουμε μία γενική εικονά για το πως δουλεύει η εφαρμογή. Παρατηρούμε ότι έχουμε μερικούς χρήστες της εφαρμογής οι οποίοι είναι επιβάτες σε διάφορα λεωφορεία. Μέσω της κινητής τους συσκευής, επικοινωνούν με τον server και στέλνουν συνεχώς την τοποθεσία τους. Από την πλευρά του server, βλέπουμε ότι πρόκειται ουσιαστικά για έναν υπολογιστή που λαμβάνει αυτά τα δεδομένα κι ανανεώνει την τοποθεσία του αντίστοιχου λεωφορείου.

Τέλος, έχουμε κάποιον χρήστη που απλώς στέκεται στη στάση και περιμένει το λεωφορείο. Ανοίγοντας της εφαρμογή, συνδέεται με τον server και μπορεί να δει που βρίσκεται το λεωφορείο σε πραγματικό χρόνο πάνω στον χάρτη. Έτσι, είναι σε θέση να ξέρει σε πόση ώρα θα βρίσκεται εκεί το λεωφορείο που τον ενδιαφέρει.



Σχήμα 3.1: Τρόπος λειτουργίας BusFinder

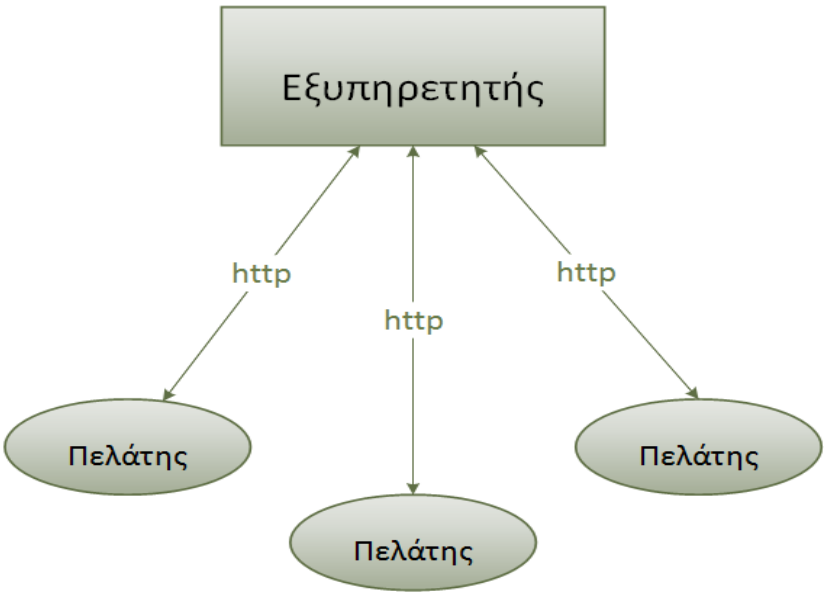
Ας δούμε αναλυτικότερα τί είναι ο εξυπηρετητής και τι ο πελάτης για εμάς.

3.2.1 Εξυπηρετητής (Server)

Ο εξυπηρετητής, είναι εκείνος ο οποίος απαντά στα αιτήματα των πελατών και βρίσκεται σε συνεχή επικοινωνία μεταξύ τους, όπως μπορούμε να δούμε στην εικόνα 3.2. Συγκεκριμένα, για εμάς θα είναι ένας server με τις αποθηκευμένες διαδρομές, στον

οποίο θα στέλνουμε συνεχώς αιτήματα για δύο πράγματα:

- Αφενός, στέλνουμε αίτημα για να μάθουμε όλες τις διαδρομές των λεωφορείων, όπου είναι αποθηκευμένες στη βάση δεδομένων.
- Αφετέρου, επικοινωνούμε σε συνεχή βάση για να παίρνουμε σε πραγματικό χρόνο κάθε στιγμή τις συντεταγμένες του λεωφορείου.



Σχήμα 3.2: Μοντέλο Εξυπηρετητή - Πελάτη

Σε αυτό το σημείο, λοιπόν, βλέπουμε ότι είναι αναγκαία η δημιουργία μίας βάσης δεδομένων που θα επικοινωνεί με τον server μας. Μέσα στη βάση δεδομένων χρειάζεται να κρατάμε πληροφορίες για αυτά τα δύο συγκεκριμένα πράγματα που αναφέρθηκαν και παραπάνω, για την τωρινή θέση των λεωφορείων αλλά και για τις διαδρομές όλων των λεωφορείων. Οπότε, η δημιουργία της βάση δεδομένων γίνεται λαμβάνοντας υπόψιν τα παραπάνω. Θα αναφερθούμε σε αυτό με περισσότερες λεπτομέρειες στο επόμενο κεφάλαιο.

3.2.2 Πελάτης (Client)

Ο πελάτης, όπως αναφέραμε και προηγουμένως, είναι εκείνος ο οποίος στέλνει τις αιτήσεις στο μοντέλο επικοινωνίας εξυπηρετητή - πελατών. Στη δική μας περίπτωση, πε-

λάτης θα είναι το κινητό ή γενικότερα οι χρήστες της εφαρμογής. Η εφαρμογή, καλείται να διαχειριστεί αυτή την επικοινωνία, μεταξύ κινητής συσκευής κι εξυπηρετητή.

Μέσω πολλαπλών διεργασιών, επιτυγχάνει διαύλους επικοινωνίας με τον εξυπηρετητή, που επιτρέπουν στον χρήστη να βλέπει ανά πάσα ώρα και στιγμή που βρίσκεται το λεωφορείο καθώς και να στέλνει την τοποθεσία του λεωφορείου όταν βρίσκεται μέσα σε αυτό.

Έτσι, μπορούμε εύκολα να αντιληφθούμε ότι ο ρόλος της εφαρμογής μπορεί να χωριστεί σε δύο βασικά μέρη:

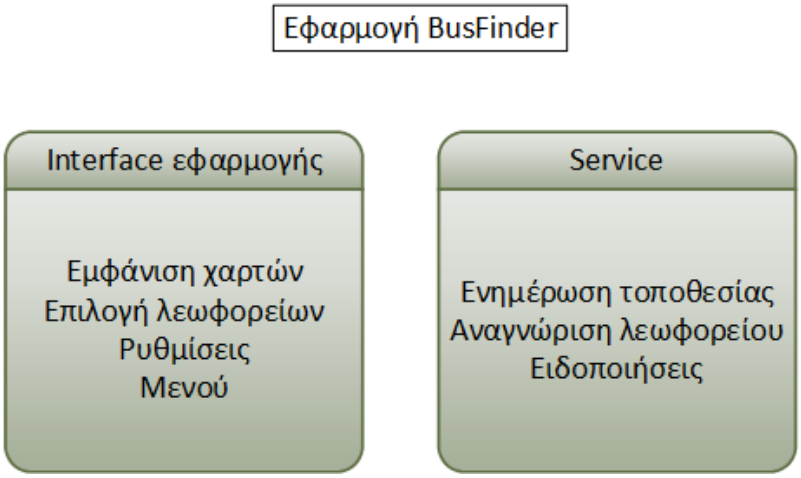
- Πρώτον, το γραφικό περιβάλλον της εφαρμογής, όπου εμφανίζουμε τα λεωφορεία που μας ενδιαφέρουν.
- Δεύτερον, η επικοινωνία που γίνεται στο "παρασκήνιο" και στέλνει στη βάση δεδομένων την τοποθεσία ενός επιβάτη, με στόχο την ανανέωση της θέσης του λεωφορείου.

Το γραφικό περιβάλλον της εφαρμογής μας θα το αποτελούν οι χάρτες, μία μπάρα εργαλείων στο πάνω μέρος κι ένα επιπλέον βοηθητικό drawer μενού. Στην μπάρα εργαλείων θα βρίσκονται επιλογές για να διευκολύνουν την πλοήγηση των χρηστών. Συγκεκριμένα, οι χρήστες θα μπορούν να διαλέγουν ποιο/α λεωφορείο/α θέλουν να εμφανίσουν και να τα εμφανίζουν πάνω στον χάρτη. Επίσης, θα μπορούν να απενεργοποιήσουν τις ειδοποιήσεις της εφαρμογής τελείως. Τέλος, όταν δεν υπάρχει σύνδεση στο διαδίκτυο ή δεν είναι ενεργοποιημένο το GPS, τότε φροντίζει να εμφανίζει κατάλληλο μήνυμα στην οθόνη.

Το service που τρέχει στο παρασκήνιο τώρα, θα φροντίζει να παίρνει διαρκώς την τοποθεσία του χρήστη και όταν υπάρχει πιθανότητα να βρίσκεται μέσα σε κάποιο λεωφορείο να τον ερωτά, εμφανίζοντας μία ειδοποίηση. Αν ο χρήστης απαντήσει θετικά ότι βρίσκεται μέσα σε κάποιο λεωφορείο, τότε ανεβάζει τις συντεταγμένες του στη βάση δεδομένων στο αντίστοιχο λεωφορείο.

Στο σχήμα 3.3 μπορούμε να δούμε τα δύο ξεχωριστά στάδια στα οποία αναπτύχθηκε η εφαρμογή. Συγκεκριμένα, το πρώτο περιλαμβάνει το κομμάτι του γραφικού περιβάλλοντος ενώ το δεύτερο το service που τρέχει στο προσκήνιο. Το πρώτο κομμάτι περιλαμβάνει τους χάρτες, το μενού για την επιλογής ενός λεωφορείου, τις ρυθμίσεις

και τις συνδέσεις στο διαδίκτυο/GPS. Ενώ το δεύτερο κομμάτι περιλαμβάνει την καταγραφή της τωρινής τοποθεσίας συνεχώς, αναγνώριση ενός λεωφορείου, το κομμάτι των ειδοποιήσεων και την ανανέωση της τοποθεσίας ενός λεωφορείου.



Σχήμα 3.3: Φάσεις ανάπτυξης του BusFinder

Περισσότερες πληροφορίες για την υλοποίηση των άνωθεν θα δοθούν στο επόμενο κεφάλαιο, όπου αφορά της υλοποίηση της εφαρμογής.

Κεφάλαιο 4

Υλοποίηση

4.1 Διαδικασία ανάπτυξης εφαρμογής

Ο αναμενόμενος χρόνος υλοποίησης της εφαρμογής είχε υπολογιστεί ότι θα ήταν οι έξι (6) μήνες. Ξεκινήσαμε την εφαρμογή τον Απρίλιο, οπότε ως χρόνος τερματισμού είχε οριστεί ο Σεπτέμβριος. Όμως, λόγω κάποιων προβλημάτων που δεν είχαν προβλεφθεί εξ αρχής, η υλοποίηση τελείωσε τέσσερις (4) μήνες αργότερα, τον Ιανουάριο. Συνολικά, λοιπόν, η εφαρμογή μας ήταν έτοιμη ύστερα από το χρονικό διάστημα των δέκα (10) μηνών, δηλαδή τον Ιανουάριο του επόμενου έτους (2016).

Αρχικά, με τον επιβλέποντα της παρούσης δπλωματικής, αποφασίσαμε πως θα εργαστούμε. Καταλήξαμε ότι η καλύτερη μέθοδος είναι οι εβδομαδιαίες συναντήσεις. Κάθε εβδομάδα λάμβανα τις κατάλληλες υποδείξεις και παρατηρήσεις για το τι πρέπει να υλοποιηθεί στη συνέχεια. Συγκεκριμένα, εξ'αρχής είχαμε μικρές, λειτουργικές εφαρμογές, όπου κάθε φορά προσθέταμε την απαραίτητη παραπάνω λειτουργικότητα.

Αυτός ο τρόπος υλοποίησης που διαλέξαμε είναι ευρέως γνωστός ως "ευέλικτη μεθοδολογία" (agile software development, ASD) [19]. Προτιμάται έναντι της μεθοδολογίας καταρράκτη, καθώς στην τελευταία τα συστήματα είναι πλήρως προσδιορίσιμα εξ αρχής. Σε αντίθεση, η ευέλικτη μεθοδολογία βασίζεται στις αρχές της συνεχούς βελτίωσης του σχεδιασμού, συχνών δοκιμών καθώς και ταχείας ανατροφοδότησης κι αλλαγής των απαραίτητων σημείων [20] [21]. Γενικότερα, στη μεθοδολογία καταρράκτη, οι απαιτήσεις είναι προσδιορισμένες εξ αρχής και σε μεγάλο βαθμό σταθερές. Ενώ στην ευέλικτη μεθοδολογία, έχουμε πολλές και άγνωστες εναλλαγές, οι οποίες αποκαλύπτονται κατά

τη διάρκεια υλοποίησης [22]. Στο διάγραμμα 4.1 μπορούμε να δούμε βασικές διαφορές ανάμεσα στις δύο αυτές μεθόδους.

Η η πιο γνωστή ευέλικτη μέθοδος είναι ο "ακραίος προγραμματισμός" (extreme programming). Ωστόσο, υπάρχουν και άλλες μέθοδοι όπως: Scrum [23], Crystal [24], Adaptive Software Development [25], DSDM [26] και Feature Driven Development [27]. Αν και κάθε ευέλικτη μέθοδος έχει τη δική της ξεχωριστή προσέγγιση, όλες μοιράζονται το ίδιο όραμα και βασικές αρχές, όπως παρουσιάζονται στο Μανιφέστο για την Ευέλικτη Ανάπτυξη Λογισμικού [28].

Εμείς χρησιμοποιήσαμε τη μέθοδο Feature Driven Developement (FSD). Η FDD είναι μια ελαφριά διαδικασία ανάπτυξης λογισμικού καθοδηγούμενη από μοντέλα, προσανατολισμένη στην παράδοση συχνών, απτών και λειτουργικών αποτελεσμάτων. Τα ελαφριά χαρακτηριστικά της μεθόδου κάνουν τις διαδικασίες της εύκολες στην παρακολούθηση κι ευκίνητες. Η FDD επικεντρώνεται στον σχεδιασμό και τον προγραμματισμό, γεγονός που τη διαφοροποιεί από άλλες προσεγγίσεις ανάπτυξης. Συνδυάζει την ευέλικτη ανάπτυξη και την ανάπτυξη με βάση τα μοντέλα, δίνοντας έμφαση στο αρχικό μοντέλο, τον καταμερισμό του έργου σε χαρακτηριστικά, και τον σχεδιασμό κάθε χαρακτηριστικού σε επαναλήψεις. Η μέθοδος παρουσιάζεται ως κατάλληλη για την ανάπτυξη κρίσιμων συστημάτων. Μια επανάληψη ενός χαρακτηριστικού αποτελείται από δύο φάσεις: το σχεδιασμό και την ανάπτυξη.

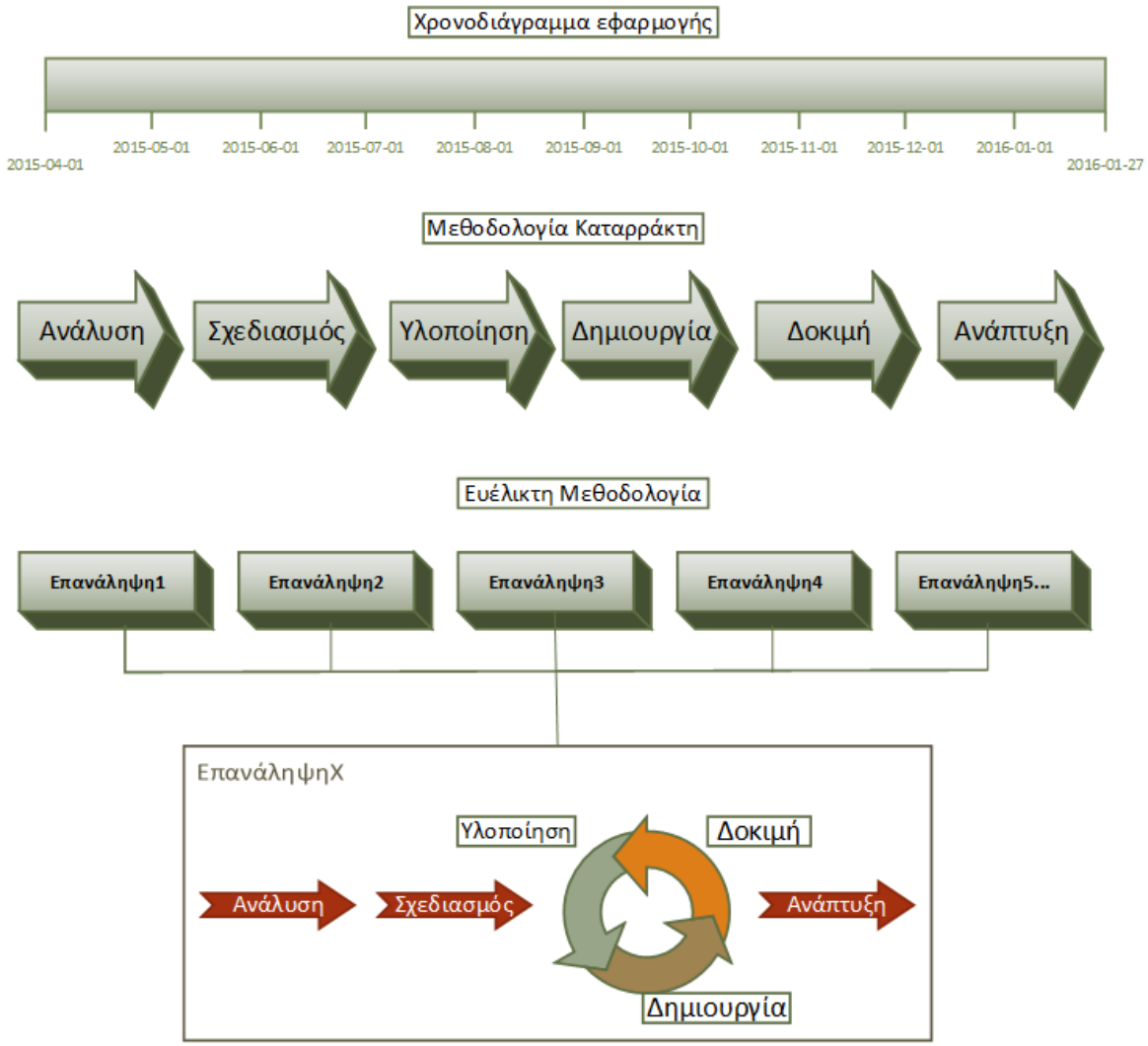
Ένα αξιοσημείωτο χαρακτηριστικό της FDD είναι ότι δίνει μια μικρή παραλλαγή στον ορισμό του “χαρακτηριστικού”, με το χαρακτηριστικό να σημαίνει μια λειτουργικότητα που έχει αξία για τον πελάτη και μπορεί να υλοποιηθεί σε δύο εβδομάδες ή λιγότερο. Η παραπάνω παραδοχή περιλαμβάνει ένα χρονοδιάγραμμα, η οποία πιστεύεται ότι συνδυάζει διακριτικά τους τεχνικούς παράγοντες (π.χ. ευελιξία) και κοινωνικό – ψυχολογικούς παράγοντες (π.χ. ενθάρρυνση των μελών της ομάδας) στη διαδικασία ανάπτυξης λογισμικού. Τα χαρακτηριστικά, ως ενότητες επανάληψης, σχεδιάζονται κι αναπτύσσονται ένα – ένα, παρέχονταςαπτά και σταθερά αποτελέσματα. Αυτό καθιστά τον έλεγχο ποιότητας πιο εύκολο. Το χαρακτηριστικό επιλέγεται προσεχτικά ώστε να υλοποιείται σε σχετικά άνετο χρόνο, δίνοντας αίσθημα εμπιστοσύνης, ενθάρρυνσης και κίνητρα στους προγραμματιστές. Γενικά, η μεθοδολογία που βασίζεται στα χαρακτηριστικά έχει αποδειχθεί αποτελεσματική στα σύγχρονα συστήματα λογισμικού.

Δουλεύοντας με την ευέλικτη μεθοδολογία, λοιπόν, καταφέραμε να υλοποιήσουμε

την εφαρμογή με τον καλύτερο δυνατό τρόπο [29]. Καθώς πρόκειται για προγραμματισμό ηλεκτρονικής συσκευής και συγκεκριμένα στην παρούσα υλοποίηση android, συνεχώς αλλάζουν οι μεθοδολογίες υλοποίησης πολλών λειτουργιών. Εξαιτίας των αλλαγών που έπρεπε να γίνονται συνεχώς πάνω στον κώδικά μας, η μεθοδολογία του καταρράκτη ήταν ακατάλληλη, καθώς με αυτήν οι απαιτήσεις προκαθορίζονται πριν την ανάπτυξη.

Συμπερασματικά, ναι μεν η μεθοδολογία των καταρρακτών αφορά γραμμικό μοντέλο, το οποίο λόγω της απλότητάς του είναι κι ευκολότερο στην υλοποίησή του, αλλά αν υπάρξει η ανάγκη να γίνουν αλλαγές στις απαιτήσεις στη φάση της ανάπτυξης της εφαρμογής, η διαδικασία είναι χρονοβόρα κι αρκετά δύσκολη σε αυτό το στάδιο. Έτσι, λοιπόν, καταλήγουμε στο γεγονός ότι, στη δική μας περίπτωση, η ευέλικτη μέθοδος είναι η καταλληλότερη για την ανάπτυξη της εφαρμογής μας.

Στο διάγραμμα 4.1 που ακολουθεί μπορούμε να δούμε αρχικά το χρονοδιάγραμμα της εφαρμογής μας, από τον Απρίλιο μέχρι τον Ιανουάριο του επόμενου έτους· διάστημα δέκα (10) μηνών συνολικά. Ακόμη, είναι εύκολο να διαπιστώσουμε τη γραμμικότητα που χαρακτηρίζει τη μεθοδολογία καταρράκτη στην εκτέλεση των βημάτων για την ανάπτυξη μίας εφαρμογής - αλλά κι ενός συστήματος γενικότερα - σε αντίθεση με την ευέλικτη μεθοδολογία, όπου διαρκώς έχουμε έτοιμες υλοποιήσεις της εφαρμογής και σε κάθε στάδιο προσθέτουμε την επιπλέον λειτουργικότητα που χρειάζεται.



Σχήμα 4.1: Ευέλικτη Μεθοδολογία και Μεθοδολογία Καταρράκτη

4.2 Εργαλεία

4.2.1 Eclipse

Σχήμα 4.2: Λογότυπο Eclipse IDE

Το Eclipse είναι ένα ολοκληρωμένο περιβάλλον εργασίας μέσα από το οποίο μπορούμε να γράψουμε και να εκτελέσουμε κώδικα. Γι' αυτό το λόγο συνήθως λέμε ότι το Eclipse είναι ένα SDK (Software Development Kit). Το περιβάλλον αυτό

είναι ελεύθερης διανομής (freeware) κι ανοικτού κώδικα (open source).

Στα πρώτα στάδια της ανάπτυξής μας εργαστήκαμε στο προγραμματιστικό περβάλλον Eclipse, κάνοντας εγκατάσταση και στα κατάλληλα εργαλεία και plug-ins που χρειαζόμαστε για την ανάπτυξη εφαρμογών Android. Ύστερα από μερικούς μήνες, όμως, ανακοινώθηκε σταθερή (stable) έκδοση για το Android Studio και όχι beta πλέον. Έτσι, αποφασίστηκε η αλλαγή γραφικού περιβάλλοντος κι από το Eclipse με το ADT Plugin, πήγαμε στο Android Studio. Από τους σημαντικότερους λόγους που αποφασίσαμε την αλλαγή κονσόλας είναι ότι το Android Studio βασίζεται στο IntelliJ το οποίο λέγεται ότι είναι γρηγορότερο κι ελαφρύτερο από το Eclipse.

4.2.2 Android Studio

Το Android Studio είναι κι αυτό με τη σειρά του ένα ολοκληρωμένο προγραμματιστικό περιβάλλον (IDE) το οποίο χρησιμοποιείται για τη δημιουργία εφαρμογών Android. Κατασκευάστρια εταίρια είναι η Google Inc, η οποία ανακοίνωσε την πρώτη δοκιμαστική του έκδοση το 2013. Τον Δεκέμβριο του 2014 ανακοινώθηκε σταθερή έκδοση της πλατφόρμας, η οποία σιγά - σιγά αντικατέστησε το Eclipse ADT.

Σχήμα 4.3: Λογότυπο Android Studio

Πρόκειται για το περιβάλλον που χρησιμοποιήσαμε κατά κόρον για την ανάπτυξη του BusFinder. Δουλεύοντας σε περιβάλλον Linux, διανομή Fedora, η σταθερή έκδοση που υπήρχε για το Android Studio ήταν η 1.2 και στη συνέχεια έγινε ανανέωση στην 1.5 RC, την οποία και χρησιμοποιήσαμε προς το τέλος μόνο. Σε γενικές γραμμές πρόκειται για ένα πολύ καλό εργαλείο, που κάνει τον προγραμματισμό διασκεδαστικό καθώς επίσης και οι διάφορες συντομεύσεις που προσφέρει διευκολύνουν κατά πολύ τη συγγραφή κώδικα.

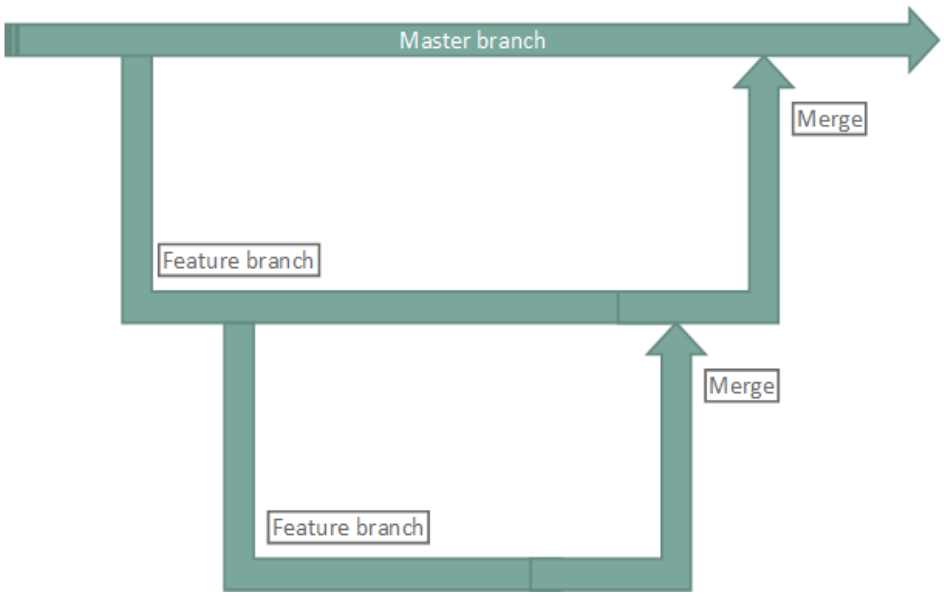
4.2.3 Git

Σχήμα 4.4: Λογότυπο Git

Το Git είναι ένα κατανεμημένο σύστημα ανάπτυξης λογισμικού που χρησιμοποιείται για τον έλεγχο εκδόσεων. Το Git αρχικά σχεδιάστηκε και δημιουργήθηκε από τον Λίνους Τορβαλντς για την ανάπτυξη του πυρήνα του Linux το 2005 [30]. Το ίδιο διαθέτει καταλόγους εργασίας, όπου κάθε κατάλογος έχει αποθηκευμένα τα πάντα με πλήρες ιστορικό και πλήρης παρακολούθηση έκδοσης, ανεξαρτήτως πρόσβασης δικτύου ή ενός κεντρικού εξυπηρετητή. Πρόκειται για ελεύθερο λογισμικό, όπως και ο kernel του Linux άλλωστε.

Η απόφαση για τη χρήση του Git πάρθηκε όταν έπρεπε να γίνει μία μεγάλη αλλαγή στον κώδικα. Συγκεκριμένα, υπήρχε η ανάγκη από τον Location Manager να γίνει αλλαγή στο FusedLocation. Έτσι, για να αποφευχθεί κάποιο ενδεχόμενο οποιασδήποτε "μοιραίας" αλλαγής, δημιουργήσαμε ένα Git repository, όπως λέγεται, στο οποίο προσθέσαμε το πρότζεκτ. Έτσι κάθε φορά που υπήρχε ανάγκη να γίνει κάποια αλλαγή, δημιουργούσαμε απλώς ένα καινούργιο "κλαδί" [31] -branch όπως κανονικά λέγεται- κι εργαζόμασταν εκεί. Αν κάτι δεν πήγαινε καλά, μπορούσαμε ανά πάσα στιγμή να επιστρέψουμε στο αρχικό branch - master branch. Έτσι, υπήρχε βεβαιότητα ότι σε κάθε βήμα θα είχαμε ΜΟΝΟ λειτουργικό κώδικα.

Ο τρόπος με τον οποίο λειτουργεί το Git, με τα διαφορετικά branches και πως, κάνοντας merge, μπορούμε βήμα - βήμα να καταλήξουμε στο master branch, φαίνεται στην εικόνα 4.5 που ακολουθεί.



Σχήμα 4.5: Τρόπος Λειτουργίας του Git

4.2.4 Android Studio Emulator

Το Android Studio προσφέρει έναν προσομοιωτή κινήτης συσκευής που τρέχει στον υπολογιστή. Πρόκειται για έναν bult-in προσομοιωτή που προσφέρει το Android Studio κι επιτρέπει τη δημιουργία, σχεδίαση και δοκιμή εφαρμογών Android σε αυτόν. Ο προσομοιωτής μιμείται όλα χαρακτηριστικά ενός απλού τηλεφώνου, με τη μόνο διαφορά βεβαίως ότι δεν μπορεί να πάρει τηλέφωνο! Γενικότερα, όμως, τρέχει λειτουργικό Android μέχρι και το επίπεδο του πυρήνα και δε διαφέρει από άποψη λειτουργικού συστήματος από κάποια συσκευή Android του εμπορίου.

Σχήμα 4.6: Android Studio Emulator

Αν και πολλά υποσχόμενο από τον τρόπο που το παρουσιάζει η Google, στην πράξη ήταν αρκετά αργός. Δυστυχώς, πέρα από αυτό, δεν καταφέραμε να εκτελέσουμε και βασικές λειτουργικότητες που θελήσαμε για τη δοκιμή της εφαρμογής μας. Συγκεκριμένα, μιας και πρόκειται για μία εφαρμογή που βασίζεται αμιγώς στην τοποθεσία, όλοι

οι έλεγχει που θέλαμε να κάνουμε αφορούσαν αυτό το κομμάτι. Όμως, το στίγμα και γενικότερα η διαχείριση του GPS της εικονικής συσκευής δεν ήταν καθόλου λειτουργική. Γι' αυτό το λόγο καταφύγαμε σε έναν άλλον emulator, όπως θα διαπιστώσετε και παρακάτω.

4.2.5 GenyMotion

Σχήμα 4.7: Λογότυπο προσομοιωτή GenyMotion

Ο GenyMotion είναι κι αυτός ένας προσομοιωτής, ο οποίος για προσωπική χρήση και μόνο δίδεται δωρεάν. Έχει δημιουργηθεί από την εταιρία GenyMobile [32], όπου κατασκευάζει γενικώς εργαλεία για Android. Η κύρια διαφορά του με τον προσομοιωτή του Android Studio είναι ότι είναι κατά πολύ πιο γρήγορος. Ακόμη, δίδεται η δυνατότητα στον χρήστη να ορίσει τις τοποθεσίες του GPS όπως εκείνος επιθυμεί, πράγμα απολύτως χρήσιμο για την εφαρμογή μας.

Όπως όλοι οι προσομοιωτές, όμως, όταν η ανάπτυξη της εφαρμογής άρχισε να γίνεται πιο πολύπλοκη, ξεκίνησε να αργεί αισθητά. Με το μεγαλύτερο πρόβλημα να έρχεται όταν για κάποιον λόγο δεν μπορούσε να εμφανίσει τους χάρτες.

Έτσι, λοιπόν, εγκαταλήφθηκε κι αυτός ο emulator. Καταλήξαμε στο γεγονός, ότι ο καλύτερος -τόσο σε απόδοση και ποιότητα όσο και σε ταχύτητα- ήταν η χρήση πραγματικής συσκευής Android. Ξεκίνησαν οι δοκιμές σε πραγματικές συσκευές Android και τότε διαπιστώθηκε το πρόβλημα ότι πρέπει με κάποιον τρόπο να επεμβαίνουμε στην τοποθεσία της συσκευής, ώστε να μπορέσουμε να προχωρήσουμε στο κομμάτι των ελέγχων.

Γι' αυτό το λόγο, δοκιμάσαμε πολλές εφαρμογές που κάνουν ακριβώς αυτή τη δουλειά: δηλώνουν ψεύτικες συντεταγμένες. Η πιο βολική, σύμφωνα με τις υπάρχουσες απαιτήσεις καταλήξαμε ότι ήταν μία. Οπότε, προχωρήσαμε σε δοκιμές με αυτήν.

4.2.6 Mock Locations

Το "Mock Locations (fake GPS path)", είναι μία εφαρμογή που επιτρέπει στο χρήστη να δηλώσει ψεύτικες συντεταγμένες και κατασκευάστηκε από τους "Dvaoru". Βρίσκεται διαθέσιμη στο Play Store. Ακόμη, μέσω της εν λόγω εφαρμογής δίδεται στον χρήστη η δυνατότητα να επιλέξει μία διαδρομή πάνω στον χάρτη, να διαλέξει την ταχύτητά του κι έτσι να το χρησιμοποιήσει για να προσομοιώσει ότι βρίσκεται σε κάποιο κινούμενο όχημα.

Σχήμα 4.8: Εφαρμογή 'Mock Loacations'

Έτσι, το χρημοποιήσαμε ευρέως για να προσομοιώσουμε τι γίνεται όταν βρισκόμαστε μέσα σε κάποιο λεωφορείο ή ακόμη και όταν βρισκόμαστε σε κάποιο μηχανοκίνητο όχημα. Γενικά, μάς βοήθησε αρκετά στους ελέγχους, ώστε στα αρχικά στάδια της ανάπτυξης να κάνουμε τους ελέγχους στο σπίτι, έχοντας πάντα συνδεδεμένο το κινητό στον υπολογιστή και παίρνοντας έτσι όλα τα μηνύματα για λάθη που τυχόν προέκυψαν.

4.3 Εξυπηρετητής (Server)

4.3.1 Μέθοδοι HTTP

Το Hypertext Transfer Protocol (HTTP) είναι ένα πρωτόκολλο που έχει σχεδιαστεί για να έχουμε επικοινωνία μεταξύ client και server. Ο client αποστέλει ένα αίτημα (request) στον server κι έπειτα περιμένει μέχρι ο server να στείλει την απάντηση του αιτήματος (response).

Το πρωτόκολλο αυτό βασίζεται στις παρακάτω δύο μεθόδους:

1. Τη μέθοδο Get, μέσω της οποίας ζητάμε κάτι από τον server. Τα δεδομένα του αιτήματος φαίνονται στη διεύθυνση της ιστοσελίδας, όπως φαίνεται και παρακάτω:

   π.χ. http://ermis.mhl.tuc.gr/ vgioti/index.php?name1=value1&name2=value2

Αυτό έχει ως αποτέλεσμα τα Get requests να μπορούν να αποθηκευτούν σε συντομεύσεις, στα αγαπημένα και στο ιστορικό. Το μέγιστο μέγεθος της διεύθυνσης μιας ιστοσελίδας είναι 2048 χαρακτήρες, συνεπώς υπάρχει και όριο στα δεδομένα που μπορούμε να στείλουμε με το Get request. Συνήθως, χρησιμοποιείται για να ζητήσουμε δεδομένα και δε χρησιμοποιείται ποτέ όταν θέλουμε να στείλουμε απόρρητα δεδομένα.

2. Τη μέθοδο Post, με την οποία ανεβάζουμε δεδομένα στον server:

   π.χ. http://ermis.mhl.tuc.gr/ vgioti/index.php

   Οι πληροφορίες του αιτήματος αποθηκεύονται στο σώμα του μηνύματος και δεν είναι ορατές στη διεύθυνση. Άρα, σε αντίθεση με τα Get requests, δεν μπορούν να αποθηκευτούν ούτε σε συντομεύσεις, ούτε στα αγαπημένα αλλά ούτε και στο ιστορικό. Φυσικά, δεν υπάρχει κανένας περιορισμός για το μέγεθος των Post requests κι είναι ιδανικά για αποστολή ευαίσθητων δεδομένων, όπως προσωπικά δεδομένα, εικόνες, κλπ.

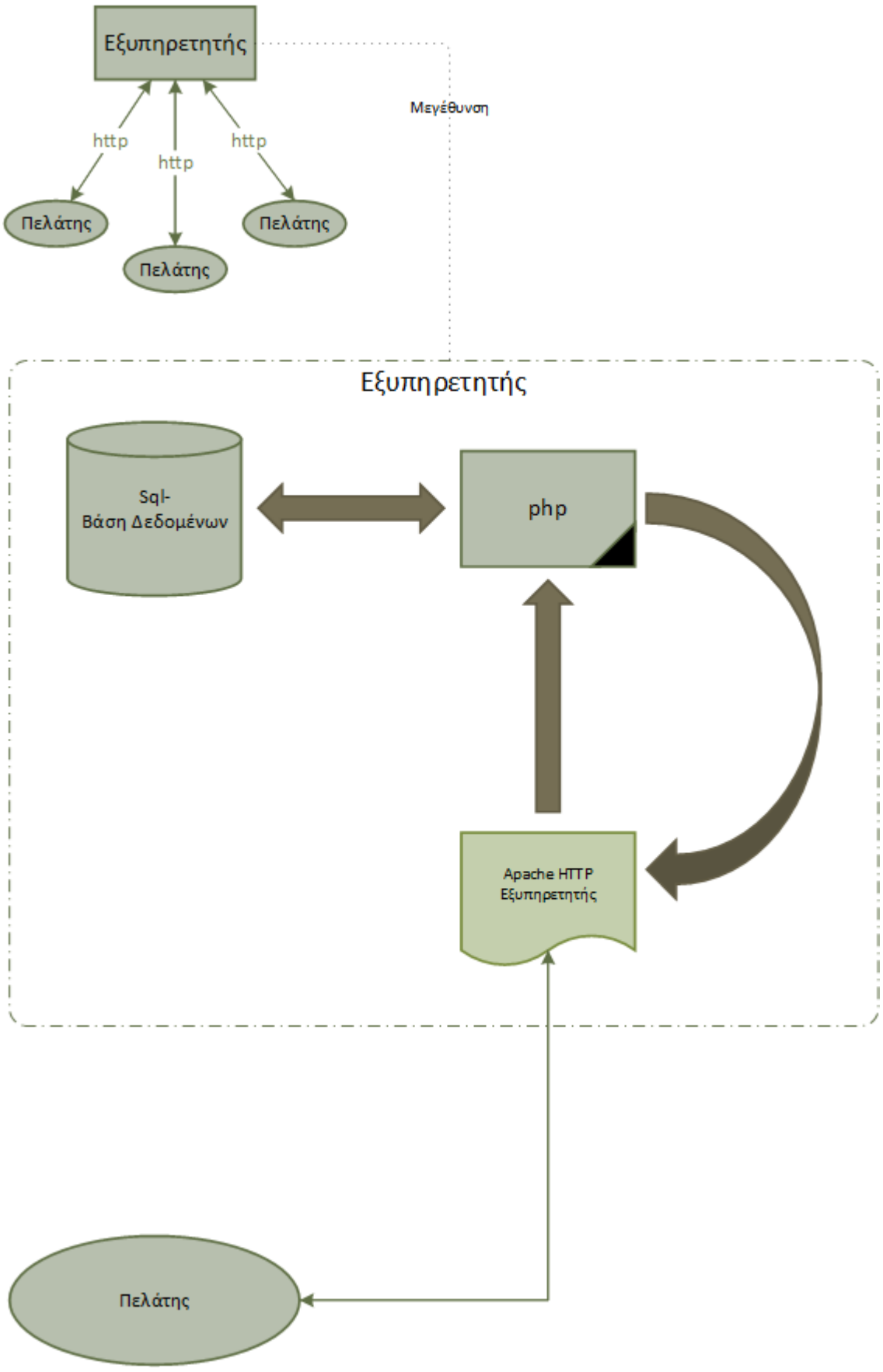
Να τονισθεί ότι στην υλοποίηση της εφαρμογής επιλέξαμε τη μέθοδο Post για λόγους ασφαλείας. Η εφαρμογή στέλνει στον server ευαίσθητα δεδομένα όπως η ακριβής τοποθεσία του χρήστη. Γι' αυτό το λόγο, δε θέλουμε να εμφανίζονται οι συντεταγμένες στο URL του request.

Για την επικοινωνία του client με τον server χρησιμοποιήσαμε τον AsyncHttpClient ο οποίος βασίζεται στις βιβλιοθήκες του Apache HttpClient και δημιουργεί ασύγχρονα HTTP αιτήματα (GET, POST, PUT, DELETE). Λέγοντας "ασύχρονα" ενοούμε εκείνα τα αιτήματα που δημιουργούνται έξω από το κυρίως UI Thread της εφαρμογής κι οποιαδήποτε επανάκλησή τους θα εκτελεστεί στο Thread που δημιουργήθηκε ο Handler. Οι δυνατότητες της κλάσης AsyncHttpClient είναι πάρα πολλές, αλλά εμείς χρησιμοποιήσαμε μόνο τον AsyncHttpResponseHandler, ο οποίος χρησιμοποιείται για να διαχειρίζεται τις απαντήσεις των αιτημάτων που δημιουργήθηκαν από τον AsyncHttpClient.

Γενικότερα, η επικοινωνία μεταξύ όλων των modules γίνεται όπως φαίνεται στο σχήμα 4.9. Συγκεκριμένα, στο αρχικό σχήμα 3.2 που παρουσίαζε το μοντέλο πελάτη - εξυπηρετητή, κάνοντας μία μεγέθυνση στον εξυπηρετητή μπορούμε να δούμε ότι εκεί έχομυε

έναν Apache http server[1], όπου έχει δημιουργηθεί από το Πολυτεχνείο Κρήτης και διατίθεται χώρος σε αυτόν ειδικά για την εφαρμογή μας. Η επικοινωνία αυτού του server με τη βάση δεδομένβν γίνεται μέσω ενός αρχείου php. Τέλος, ο πελάτης επικοινωνεί με τον server για να στείλει και να πάρει τα δεδομένα που χρειάζεται.

[1]Ερμής Apache http Server - http://ermis.mhl.tuc.gr/



Σχήμα 4.9: Επικοινωνία μεταξύ των στοιχείων του server

4.3.2 Βάση δεδομένων διακομιστή

Για την ανάπτυξης της παρούσας εφαρμογής, όπως αναφέρθηκε και νωρίτερα, είναι αναγκαία η δημιουργία μίας βάσης δεδομένων για την αποθήκευση και την αλλαγή των διαδρομών των λεωφορείων. Για να το επιτύχουμε αυτό συνδεθήκαμε στον mysql server με τη χρήση του Putty[2]. Αρχικά, αφού συνδεθήκαμε, δημιουργήσαμε τη βάση δεδομένων.

Στη συνέχεια, για την υλοποίηση της εφαρμογής δημιουργήσαμε δύο (2) πίνακες:

1. Πίνακας RouteBus (id, latitude, longitude, busName). Σε αυτόν τον πίνακα αποθηκεύουμε τα σημεία των διαδρομών των λεωφορείων, εισάγοντας για κάθε σημείο διαδρομής το μήκος, το πλάτος και το όνομα του λεωφορρείου της διαδρομής.

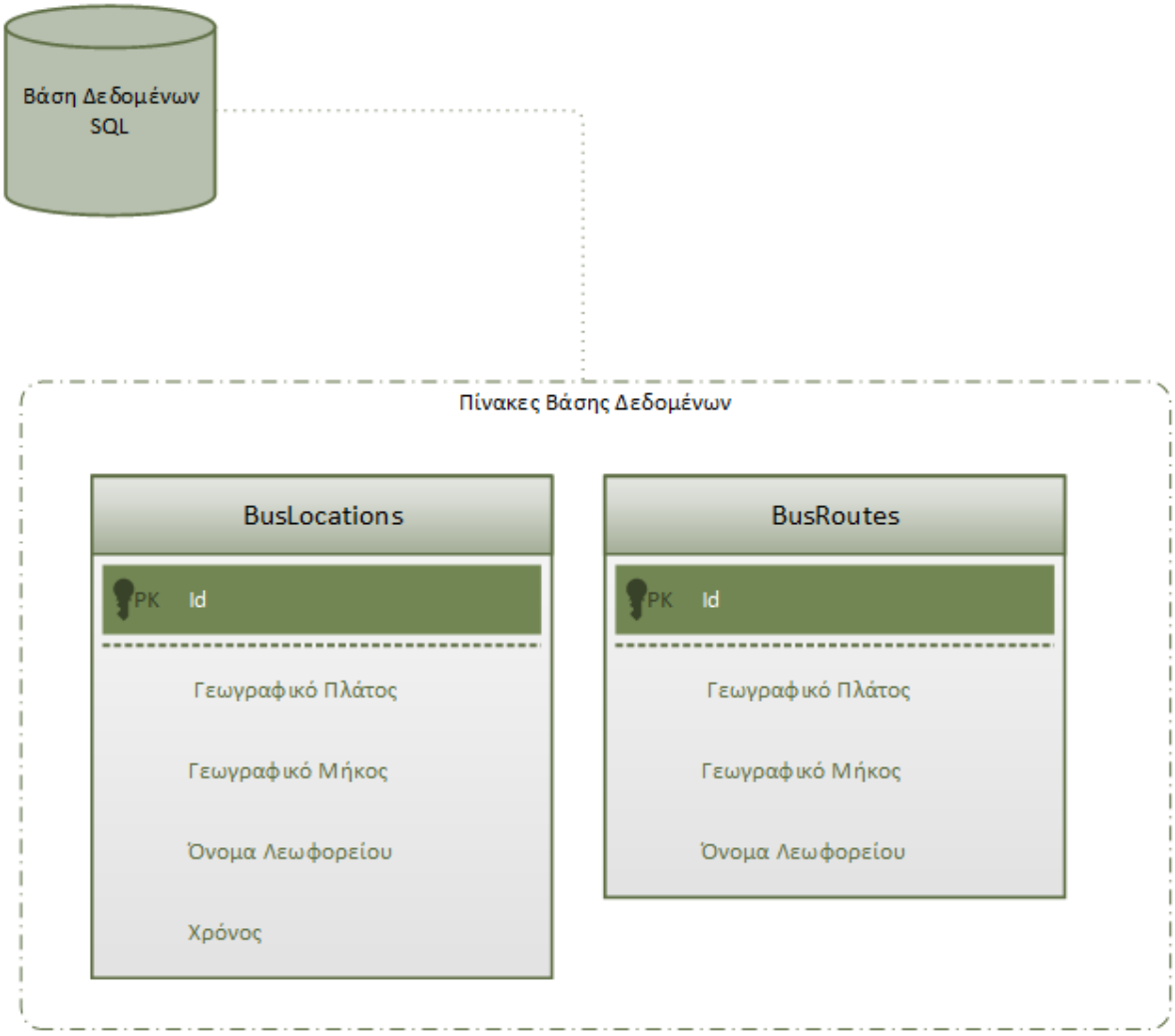
2. Πίνακας BusLocations(id, latitude, longitude, busName, lastUpdate). Οι τρέχουσες θέσεις των λεωφορείων αποθηκεύονται στον πίνακα BusLocations. Για να προσδιορίσουμε τη θέση ενός λεωφορείου χρειαζόμαστε το μήκος, το πλάτος, το όνομα του λεωφορείου και τη χρονική στιγμή που βρισκόταν σε αυτό το σημείο.

Μετά δημιουργήσαμε τους παραπάνω πίνακες με τις αντίστοιχες εντολές και γεμίσαμε τον πίνακα RouteBus με τις συντεταγμένες όλων των διαδρομών ανά λεωφορείο.

Να σημειωθεί ότι, επειδή με το terminal του putty δεν μπορούσαμε να επικολλήσουμε τις συντεταγμένες, φτιάξαμε ένα βοηθητικό αρχείο php το οποίο δημιουργεί σύνδεση με τη βάση δεδομένων κι εκτελεί τα SQL Queries που του θέτουμε.

Έτσι, λοιπόν, μεγεθύνοντας στη βάση δεδομένων βλέπουμε ότι θα έχουμε δύο πίνακες που θα έχουν τη μορφή της εικόνας 4.10. Συγκεκριμένα, ο πρώτος πίνακας με όνομα BusLocations είναι εκείνος που περιλαμβάνεις τις τωρινές τοποθεσίες των λεωφορείων. Εκεί, έχουμε τέσσερις (4) στήλες, όπου σώζουμε το γεωγραφικό πλάτος, το γεωγραφικό μήκος, το όνομα του λεωφορείου και την χρονική στιγμή που ανεβάζουμε στον server. Ο άλλος πίνκας λέγεται BusRoutes και περιλαμβάνει τις διαδρομές όλων τον λεωφορείων, ο οποίος έχει τρεις (3) διαφορετικές στήλες όπου αποθηκεύουμε το γεωγραφικό μήκος, το γεωγραφικό πλάτος και το όνομα του λεωφορείου.

[2]Δωρεάν SSH και Telnet client



Σχήμα 4.10: Πίνακες που βρίσκονται στη Βάση Δεδομένων

4.3.3 Ανταλλαγή δεδομένων μεταξύ πελάτη κι εξυπηρετητή

Χρησιμοποιήσαμε αυτή τη μορφή κειμένου στα δεδομένα που αποστέλονται μεταξύ της συσκευής Android και στον web server. Γενικά, το JSON (JavaScript Object Notation) είναι μια μορφή κειμένου αποθήκευσης δεδομένων, όπως κι η XML. Είναι φιλική για τους ανθρώπους και τους υπολογιστές, καθώς μπορούν αμφότεροι να γράψουν και να διαβάσουν πολύ εύκολα.

4.3.4 Επικοινωνία μεταξύ πελάτη κι εξυπηρετητή

Χρησιμοποιήσαμε την PHP, όπου πρόκειται για ένα δυνατό εργαλείο, για να επιτύχουμε την επικοινωνία μεταξύ Android και Database Server. Η PHP είναι μια ευρέως

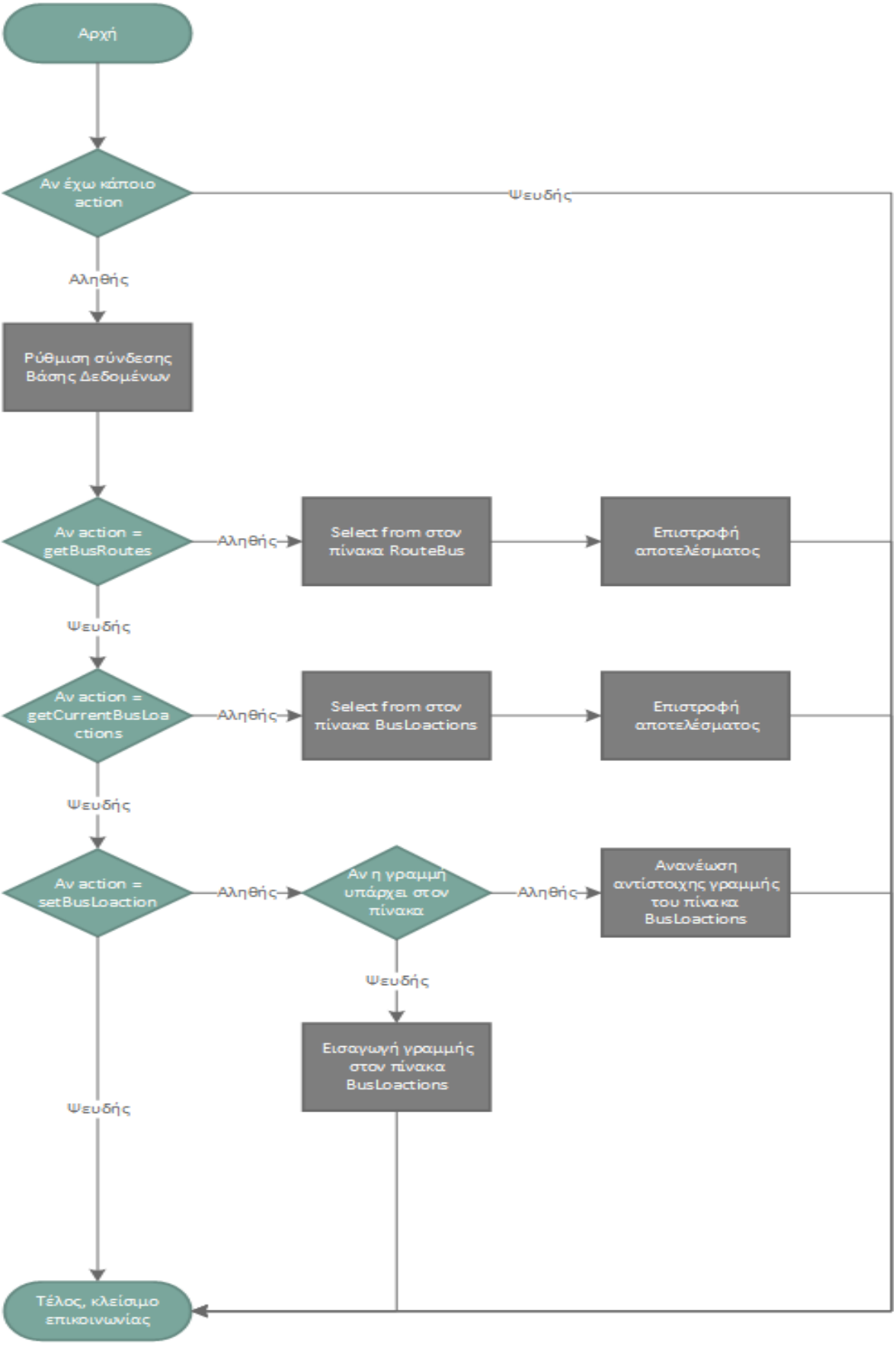
διαδεδομένη scripting γλώσσα κατασκευής ιστοσελίδων. Ο κώδικάς της αποκρύπτεται από τον client, καθώς εκτελείται εξολοκλήρου στον server. Ο client βλέπει μόνο τα αποτελέσματα της PHP.

Σε όλες τις περιπτώσεις της εφαρμογής μας πρέπει να δημιουργήσουμε μια σύνδεση μεταξύ του web Server και της βάσης δεδομένων. Επίσης, γίνεται έλεγχος για την επιτυχή ολοκλήρωση της δημιουργίας της σύνδεσης (σε περίπτωση που αποτύχει η δημιουργία, σταματάει η εκτέλεση).

Το διάγραμμα ροής 4.11 περιγράφει αναλυτικά των κώδικα που έχει υλοποιηθεί στο php αρχείο της βάσης δεδομένων. Αρχικά, ελέγχω αν έχω κάποιο action. Με τον όρο action εννοούμε αν θέλουμε να στείλουμε δεδομένα στη βάση δεδομένων ή να πάρουμε. Αν δεν έχουμε, τότε η επικοινωνία κλείνει.

Αν τώρα έχουμε κάποια ενέργεια να εκτελέσουμε, ρυθμίζουμε τη σύνδεση της βάσης δεδομένων κι ελέγχουμε αν η ενέργεια που πρέπει να κάνουμε είναι να πάρουμε τα δρομολόγια των λεωφορείων. Αν είναι αυτή τότε επιλέγουμε από τον πίνακα όλες τις γραμμές που μας δίνουν τα δρομολόγια του λεωφορείου που θέλουμε. Επιστρέφουμε το αποτέλεσμα στη συσκευή και τερματίζεται και πάλι η επικοινωνία.

Αν δε θέλουμε να πάρουμε τις διαδρομές των λεωφορείων. ελέγχουμε αν θέλουμε να πάρουμε τις συντεταγμένες του πίνακα με τις τωρινές τοποθεσίες των λεωφορείων. Σε αυτή την περίπτωση επιλέγουμε την κατάλληλη γραμμή ανάλογα με το λεωφορείο, επιστρέφουμε τα αποτελέσματα και κλείνουμε και πάλι την επικοινωνία.



Σχήμα 4.11: Διάγραμμα ροής αρχείου php

4.3.5 Βήματα επικοινωνίας κινητού με βάση δεδομένων

Γενικά, η επικοινωνία μεταξύ της κινητής συσκευής και της βάσης δεδομένων γίνεται στα εξής βήματα:

1. Android → Web Server
2. Web Server → Database Server
3. Database Server → Web Server
4. Web Server → JSON
5. JSON → Android

Αναλυτικότερα, θα έχουμε τα εξής για το κάθε κομμάτι ξεχωριστά.

1. Android → Web Server

Χρησιμοποιώντας τη μέθοδο post της κλάσης MyHttpClient στέλνουμε από το κινητό μας ένα αίτημα (http post request) στον Web Server. Συγκεκριμένα, μόλις ανοίξει η εφαρμογή θα πρέπει να ενημερωθεί για τα σημεία που σχηματίζουν τις διαδρομές των λεωφορείων. Αυτή η πληροφορία είναι καταχωρημένη στη βάση δεδομένων του server μας.

Καταχωρούμε, λοιπόν, την παράμετρο action = "getBusRoutes". Ύστερα, στέλνουμε το αίτημα στο URL uploadWebsite = "http://ermis.mhl.tuc.gr/ vgioti/index.php".

Τέλος, με τη βοήθεια της PHP ο Web Server λαμβάνει το αίτημα.

2. Web Server → Database Server

Αφού αναγνωριστεί η παράμετρος action του αιτήματος, γίνεται η σύνδεση και στέλνεται το αντίστοιχο SQL Query στον Database Server.

Στο προηγούμενο παράδειγμά μας, η παράμετρος action ισούται με getBusRoutes και συνεπώς αποστέλεται το ερώτημα:

```
"SELECT * FROM RouteBus"
```

Συγκεκριμένα, ο κώδικας θα έχει την παρακάτω μορφή:

```
$sql    = "SELECT * FROM RouteBus";
$result = mysqli_query($conn, $sql);
```

3. Database Server → Web Server

Ο Database Server επιστρέφει στον Web Server τα αποτελέσματα του Query που είχε στείλει ο δεύτερος κι αποθηκεύονται προσωρινά στη μεταβλητή $result.

4. Web Server → JSON

Εδώ, αλλάζουμε τη σύνταξη της επιστροφής του Database Server χρησιμοποιώντας τη σύνταξη του JSON, για να διευκολύνουμε την ανάγνωση από το κινητό. Έπειτα, στέλνουμε τα διαμορφωμένα δεδομένα πίσω στο κινητό.

Αναλυτικότερα, με τις ακόλουθες γραμμές κώδικα μετατρέπουμε τα αποτελέσματα σε JSON Array, τα αποθηκεύουμε στη μεταβλητή $output και τα "εκτυπώνουμε":

```
$output = "[";
// output data of each row
while($row = $result->fetch_assoc()) {
$output = $output . "{id: " . $row["id"]. ",
latitude: " . $row["latitude"]. ",
longitude: " .  $row["longitude"]. ",
busName: " . $row["busName"]. "},";
}
$output = trim($output, ",");
$output = $output . "]";
echo $output;
```

5. JSON → Android

Τέλος, το κινητό λαμβάνει τα JSON δεδομένα σαν απάντηση στο HTTP αίτημα που είχε στείλει αρχικά.

Στο παράδειγμά μας χρησιμοποιούμε τη μέθοδο parseBusRoutes για να μετατρέψουμε τα δεδομένα από JSON Array σε java.util.ArrayList<E>.

4.4 Πελάτης (Client)

Με τον όρο πελάτης εννοούμε, όπως είπαμε και προηγουμένως, ένα τμήμα λογισμικού το οποίο ζητά κάτι από ένα άλλο τμήμα λογισμικού, τον εξυπηρετητή (ή αλλιώς διακομιστή). Στη δική μας περίπτωση, λοιπόν, πελάτης είναι η εφαρμογή μας. Μέσω του BusFinder στέλνουμε αιτήματα στον server και περιμένουμε την απάντησή του, όπως περιγράφηκε κι από πάνω.

Θα μπορούσαμε να πούμε ότι τα στάδια ανάπτυξης της εφαρμογής χωρίστηκαν σε δυο μεγάλες φάσεις, όπως μπορούμε να δούμε και στην εικόνα 3.3.

1. Αυτά που δεν είναι ορατά στον χρήστη. Ό,τι έχει να κάνει με την κύρια λειτουργικότητα της εφαρμογής, δηλαδή διεργασίες και ό,τι τρέχει "από πίσω" (στο background) γενικότερα.

2. Ό,τι είναι ορατό στον χρήστη, δηλαδή η λειτουργικότητα της εφαρμογής που αντιλαμβάνεται ο χρήστης. Ενδεικτικά χαρακτηριστικά είναι το μενού εντολών, οι χάρτες κι οτιδήποτε άλλο είναι αντιληπτό στον χρήστη.

Ας ξεκινήσουμε λοιπόν, παρουσιάζοντας το πρώτο στάδιο, από το οποίο ξεκινήσαμε κιόλας.

4.4.1 Services

Σκοπος του BusFinder είναι η πληροφόρηση των χρηστών για την τοποθεσία του λεωφορείου, ώστε να διευκολυνθεί η καθημερινότητά τους όσο περισσότερο γίνεται. Για να εξασφαλίσουμε την ομαλή λειτουργικότητα της εφαρμογή, πρέπει να είμαστε βέβαιοι ότι βασικές λειτουργίες θα εκτελούνται στο background.

Συγκεκριμένα, για να είμαστε σε θέση να αντιληφθούμε που είναι το λεωφορείο, το κάνουμε έμμεσα, εκμεταλλευόμενοι τους επιβάτες του.

1. Έτσι, το BusFinder πρέπει να γνωρίζει ανά πάρα στιγμή τη θέση του πιθανού επιβάτη ή αλλιώς χρήστη της εφαρμογής.

2. Για να το κάνει αυτό, πρέπει ο χρήστης να έχει ενεργοποιήσει τις ρυθμίσεις που επιτρέπουν στην συσκευή να αναγνωρίζει την τοποθεσία στην οποία βρίσκεται. Για μεγαλύτερη ακρίβεια προτείνεται στον χρήστη κι η ενεργοποίηση του GPS της συσκευής του.

3. Ακόμη, χρειάζεται μία επικοινωνία του χρήστη με τον server, ώστε να κατεβάσει τις συντεταγμένες όλων των διαδρομών των διαθέσιμων λεωφορείων.

4. Όταν υπάρχουν ενδείξεις ότι ο χρήστης μπορεί να βρίσκεται σε ένα λεωφορείο, τότε εμφανίζεται μία ειδοποίηση, όπου τον ρωτά αν βρίσκεται μέσα και σε ποιο λεωφορείο. Ο χρήστης βλέπει μόνο τα τρία (3) πιο κοντινά λεωφορεία σε αυτόν, καθώς χρησιμοποιείται ένας αλγόριθμος που αναγνωρίζει ποια λεωφορεία είναι πιο κοντά σε εκείνον.

5. Ακόμη, όταν ο χρήστης είναι επιβάτης στο λεωφορείο, πρέπει με κάποιο τρόπο να γίνεται έλεγχος της σύνδεσής του στο διαδίκτυο, ώστε να εξασφαλίζεται η ομαλή επικοινωνία μεταξύ της συσκευής και του server.

6. Όταν όλες οι παραπάνω προϋποθέσεις πληρούνται, δηλαδή είναι κάποιος επιβάτης μέσα στο λεωφορείο αλλά κι έχει σταθερή σύνδεση με το διαδίκτυο καθώς κι ενεργοποιημένο το GPS του, τότε γίνεται ένα ανέβασμα των συντεταγμένων του και κατ' επέκταση των συντεταγμένων του λεωφορείο στον server.

7. Επίσης, η αποβίβαση του χρήστη, αναγνωρίζεται αυτόματα χρησιμοποιώντας έναν αλγόριθμο που στην πράξη δούλεψε πλήρως.

8. Τέλος, προσφέρονται δύο λειτουργίες χρήσης ανάλογα με τις απαιτήσεις της στιγμής, λειτουργία εξοικονόμησης ενέργειας αλλά κι υψηλής ακρίβειας.

1. Εύρεση τρέχουσας τοποθεσίας

Η Google, δίνει τη δυνατότητα στους προγραμματιστές Android, χρησιμοποιώντας το Google Play Services, να έχουν πρόσβαση σε μία σειρά από υπηρεσίες που παρέχει η Google μέσα από την εφαρμογή τους. Μερικά παραδείγματα των υπηρεσιών που μπορεί

ο κάθε προγραμματιστής να έχει πρόσβαση είναι το Google+, το Google Maps -όπου είναι ουσιαστικά οι χάρτες της Google - κ.α..

Με τον όρο GoogleApiClient εννοούμε ουσιαστικά το πιο απαραίτητο εργαλείο για την αξιοποίηση των υπηρεσιών που παρέχονται από το Google Play Services [33]. Το GoogleApiClient μας παρέχει πολλές μεθόδους, τις οποίες μπορούμε να χρησιμοποιήσουμε κατευθείαν στον κώδικά μας κι έτσι να κάνουμε χρήση των υπηρεσιών που προσφέρονται κι αναφέρθηκαν και παραπάνω.

Η εύρεση της τρέχουσας τοποθεσίας έγινε με τη βοήθεια του FusedLocationProviderApi, οι μέθοδοι του οποίου πρέπει να χρησιμοποιούνται σε συνδυασμένο με το GoogleApiClient κι έτσι είμαστε σε θέση να πάρουμε την τοποθεσία του χρήστη την κάθε στιγμή.

Συγκεκριμένα, δημιουργήσαμε μία καινούργια κλάση όπου την ονομάσαμε MyLocationService, όπου παίρνουμε τις συντεταγμένες όλων των λεωφορείων, δηλαδή τα δρομολόγια, που βρίσκονται στη βάση δεδομένων στον πίνακα BusRoutes στον server. Το πως γίνεται αυτή η επικοινωνία αναφέρθηκε παραπάνω. Δημιουργούμε μία καινούργια μέθοδο που να υλοποιεί τα παραπάνω, την getBusLines. Έχοντας πλέον σώσει σε ξεχωριστές ArrayList, τοπικά, τα δρομολόγια των διαθέσιμων λεωφορείων μπορούμε ανά πάρα στιγμή να επεξεργαστούμε τα δεδομένα των εν λόγω λιστών.

Έχοντας δημιουργήσει μία ακόμη λίστα, στην οποία αποθηκεύουμε όποιο λεωφορείο βρούμε και βρίσκεται σε απόσταση μικρότερη των 75 μέτρων, υποθέτουμε αυτομάτως ότι βρίσκεται και σε διαδρομή λεωφορείου. Επομένως, τα τρία (3) - το πολύ - λεωφορεία που θα βρίσκονται πιο κοντά στην τωρινή τοποθεσία του χρήστη, θα είναι κι αυτά που θα εμφανιστούν στην ειδοποίησή του. Εξασφαλίζεται, έτσι, με έναν έξυπνο τρόπο ότι, με ένα κλικ και μόνο, ο επιβάτης θα μπορεί να δίνει πληροφορίες για τη θέση του εκάστοτε λεωφορείου.

2. Ενεργοποίηση GPS

Όπως είπαμε και νωρίτερα, για να μπορέσει η συσκευή να κάνει τους απαραίτητους ελέγχους και να στείλει τις απαραίτητες πληροφορίες στον server, ώστε να διασφαλιστεί η πλήρης λειτουργικότητά της πρέπει να είναι ενεργοποιημένες οι υπηρεσίες ώστε να επιτρέπεται η πρόσβαση στην τοποθεσία του κινητού, όπως φαίνεται και στο σχήμα 4.12. Επιπλέον, για μεγαλύτερη ακρίβεια ζητείται ο χρήστης να έχει ενεργοποιημένο και

το GPS της συσκευής. Να τονισθεί, βέβαια, ότι η χρήση του γίνεται για μικρό χρονικό διάστημα και μόνο κατά τη διάρκεια που κάποιος έχει επιβιβαστεί σε κάποιο λεωφορείο, ώστε έτσι να πετύχουμε μεγαλύτερη ακρίβεια άρα και καλύτερη παροχή πληροφορίας για τους χρήστης του BusFinder.

Αν ο χρήστης δεν έχει ενεργοποιήσει το GPS κι ανοίξει την εφαρμογή για πρώτη φορά, τότε του εμφανίζεται ένα μήνυμα που τον ειδοποιεί ότι πρέπει να κάνει ενεργοποίηση και τον οδηγεί στο αντίστοιχο μενού. Αν τώρα, δεν είναι η πρώτη φορά που ανοίγει την εφαρμογή ή απενεργοποιείται παρωδικά η λειτουργία του GPS, τότε η οθόνη του θα δείχνει όπως παρακάτω. Να σημειωθεί ότι αν ο χρήστης πατήσεις στο κόκκινο πλαίσιο, τότε θα οδηγηθεί στις ρυθμίσεις, όπου θα μπορεί να το ενεργοποιήσει.

Για τον έλεγχο αν το GPS είναι ενεργοποιημένο, χρησιμοποιείται ο LocationManager και συγκεκριμένα η μέθοδός του isProviderEnabled. Η κλάση LocationManager παρέχει πρόσβαση στις υπηρεσίες που αφορούν την τοποθεσία του συστήματος. Αυτές οι υπηρεσίες επιτρέπουν γενικότερα σε κάποιον να παίρνει τη γεωγραφική θέση του κινητού. Συγκεκριμένα, η μέθοδος isProviderEnabled(String provider), είναι μία boolean μέθοδος όπου επιστρέφει την τωρινή κατάσταση του παρόχου, αν είναι ενεργοποιημένος ή απενεργοποιημένος. Έτσι, παίρνουμε πληροφορία για το αν το GPS είναι ενεργοποιημένο ή απενεργοποιημένο.

Στο σχήμα 4.12 είναι ένα διάγραμμα ροής, μέσω του οποίου μπορούμε να δούμε περισσότερες λεπτομέρειες για τον αλγόριθμο που ελέγχει την λειτουργία του GPS.

3. Επικοινωνία κινητού - βάσης δεδομένων

Οι διαδρομές όλων των λεωφορείων είναι αποθηκευμένες στη βάση δεδομένων μας, μέσα σε έναν πίνακα. Κάθε ξεχωριστό λεωφορείο έχει και ξεχωριστό όνομα. Αποφασίσαμε να αποθηκεύουμε τα δεδομένα μόνο σε μία on-line βάση δεδομένων και να μη δημιουργούμε κάποια τοπικά. Κάθε φορά που γίνεται εκκίνηση της εφαρμογής, ο client - κινητό πρέπει να επικοινωνεί με τη βάση δεδομένων και να παίρνει τα στοιχεία, αποθηκεύοντάς τα σε ένα ArrayList.

Για να το επιτύχουμε αυτό, λοιπόν, δημιουργούμε μία μέθοδο, την getBusLines(). Μέσα σε αυτήν, καταχωρούμε την παράμετρο action = "getBusRoutes", το URL αποστολής uploadWebsite = "http://ermis.mhl.tuc.gr/ vgioti/index.php" και δημιουργούμε έναν AsyncHttpResponseHandler για να διαχειριστεί την απάντηση που θα λάβει

το κινητό. Έπειτα καλούμε τη μέθοδο post της κλάσης MyHttpClient, η οποία αποστέλει το αίτημα. Αν αποτύχει η επικοινωνία μεταξύ Web Server και Android θα τρέξει η μέθοδος onFailure του Handler, η οποία μας ενημερώνει μέσω Log για το σφάλμα.

Ωστόσο, αν όλα πάνε καλά κι έχουμε μια επιτυχημένη επικοινωνία θα εκτελεστεί η onSuccess η οποία καλεί τη parseBusRoutes και σαν παράμετρο βάζει την απάντηση του αιτήματος. Στην αρχή η parseBusRoutes καθαρίζει τα δεδομένα που μπορεί να περιέχουν οι λίστες των διαδρομών των λεωφορείων και μετά καταχωρεί εκαι νέου τα δεδομένα του εκάστοτε λεωφορείου που βρίσκει στο JSON Array.

4. Ειδοποίηση (Notification)

Στην περίπτωση όπου υπάρχει ενδεχόμενο ένας χρήστης να είναι μέσα σε κάποιο λεωφορείο, τότε του εμφανίζεται ειδοποίηση στην οποία ερωτάται σε ποιο λεωφορείο βρίσκεται. Για να ερωτηθεί, όμως, πρέπει να πληρούνται κάποιες προϋποθέσεις. Συγκεκριμένα,

1. Αν η απόσταση του σημείου που βρίσκεται τώρα είναι μικρότερη των 75 μέτρων σε σχέση με την απόσταση από κάποιο σημείο, όπου είναι διαδρομή λεωφορείου.

2. Αν η ταχύτητά του είναι μεγαλύτερη των $12m/s$. Σημειώνεται ότι επιλέχθηκε αυτός ο συγκεκριμένος αριθμός, καθώς ένας πεζός δεν μπορεί να αποκτήσει ταχύτητα μεγαλύτερη των $12m/s$ ή $43.2km/h$[3].

Όπως φαίνετα και στο σχήμα 4.13 όταν πληρούνται όλες οι προϋποθέσεις, τότε ρωτάμε τον χρήστη - καλώντας τη συνάρτηση askUser(). Η ερώτηση ουσιαστικά που θέτουμε είναι η ειδοποίηση που εμφανίζεται στον χρήστη - πιθανό επιβάτη.

Έχει δημιουργηθεί μία, προσαρμοσμένη στις απαιτήσεις της εφαρμογής, ειδοποίηση. Συγκεκριμένα, ο χρήστης μπορεί να διαλέξει ανάμεσα στα τρία (3) πιθανότερα ποιο είναι το λεωφορείο στο οποίο είναι επιβάτης. Διώχνοντας ή μην κάνοντας τίποτα στην εφαρμογή, δε γίνεται απολύτως τίποτα' εκλαμβάνεται σαν ο χρήστης να μη βρίσκεται μέσα σε κάποιο λεωφορείο.

Για την εμφάνιση τριών (3) μόνο επιλογών στην ειδοποίηση, έχει δημιουργηθεί ένας ειδικός αλγόριθμος. Βρίσκουμε ποια λεωφορεία είναι πιο κοντά στην τωρινή θέση και

[3]Το ρεκόρ μεγαλύτερης ταχύτητας τρεξίματος το κατέχει ως τώρα (2016) ο Usain Bolt κι είναι $12.4m/s$

τα αποθηκεύουμε σε ένα ArrayList. Επομένως, μέσα σε αυτό το ArrayList έχουμε όλα τα λεωφορεία που η αποστασή τους με την απόσταση του χρήστη είναι μικρότερη από 75μ. Για να διαλέξουμε μόνο τα τρία από όλα αυτά, διαλέγουμε τα τρία (3) που έχουν αποθηκευτεί τις περισσότερες φορές. Να σημειωθεί ότι αν είναι λιγότερα ή και τρία, τότε απλώς τα εμφανίζουμε όλα στην ειδοποίησή μας.

Αν τώρα, η ειδοποίηση δε γίνει αντιληπτή άμεσα για τον οποιοδήποτε λόγο και ο χρήστης εσφαλμένα μετά από αρκετό χρονικό διάστημα απαντήσει ότι βρίσκεται μέσα σε κάποιο λεωφορείο, τότε και πάλι δε γίνεται τίποτα. ΔΕΝ ανανεώνεται, δηλαδή, η τοποθεσία του λεωφορείου που δήλωσε ο χρήστης. Αυτό συμβαίνει δίοτι κάθε φορά η μεταβλητή, που έχει να κάνει με το αν ο χρήστης βρίσκεται μέσα σε κάποιο λεωφορείο ή όχι, εξαρτάται από πολλές συνιστώσες και όχι μόνο από την απάντηση του χρήστη, διασφαλίζοντας έτσι την εγκυρότητα των υπηρεσιών που παρέχονται, όσο αυτό είναι εφικτό.

Όμως, το BusFinder, είναι μία εφαρμογή η οποία εξαρτάται καθαρά και μόνο από τη διάθεση των χρηστών να βοηθήσουν και να συνεισφέρουν στην κοινότητά τους. Επομένως, δε δίνεται έμφαση στο να προληφθούν παραπλανητικές πληροφορίες αλλά να δοθεί η δυνατότητα σε όλους να βοηθούσουν όσο περισσότερο μπορούν τους συνανθρώπους τους. Αυτός είναι ο σκοπός κι ενός crowdsourcing πρότζεκτ άλλωστε.

Ακολουθεί το σχήμα 4.13 που παρουσιάζει πότε εμφανίζεται η ειδοποίηση αλλά και γενικότερα τι γίνεται κάθε φορά που αλλάζει η τοποθεσία του χρήστη, δηλαδή κάθε φορά που κινείται.

Να τονισθεί ότι, αν ο χρήστης δε θέλει να στείλει την τοποθεσία για τον οποιοδήποτε λόγο, είναι απολύτως εφικτό να γίνει. Αφενός, αν αγνοήσει την ειδοποίηση στην οποία ερωτάται σε ποιο λεωφορείο βρίσκεται, τότε δεν του εμφανίζεται καινούργια για ένα ευλογο χρονικό διάστημα. Ακόμη, αν θελήσει να μην ανεβάζει για το υπόλοιπο της ημέρας ή/κι εβδομάδας, μέσως της εφαρμογής δίδεται κι αυτή η επιλογή μέσα στο μενού. Με αυτόν τον τρόπο διασφαλίζεται ότι ο χρήστης έχει πάντα τον έλεγχο.

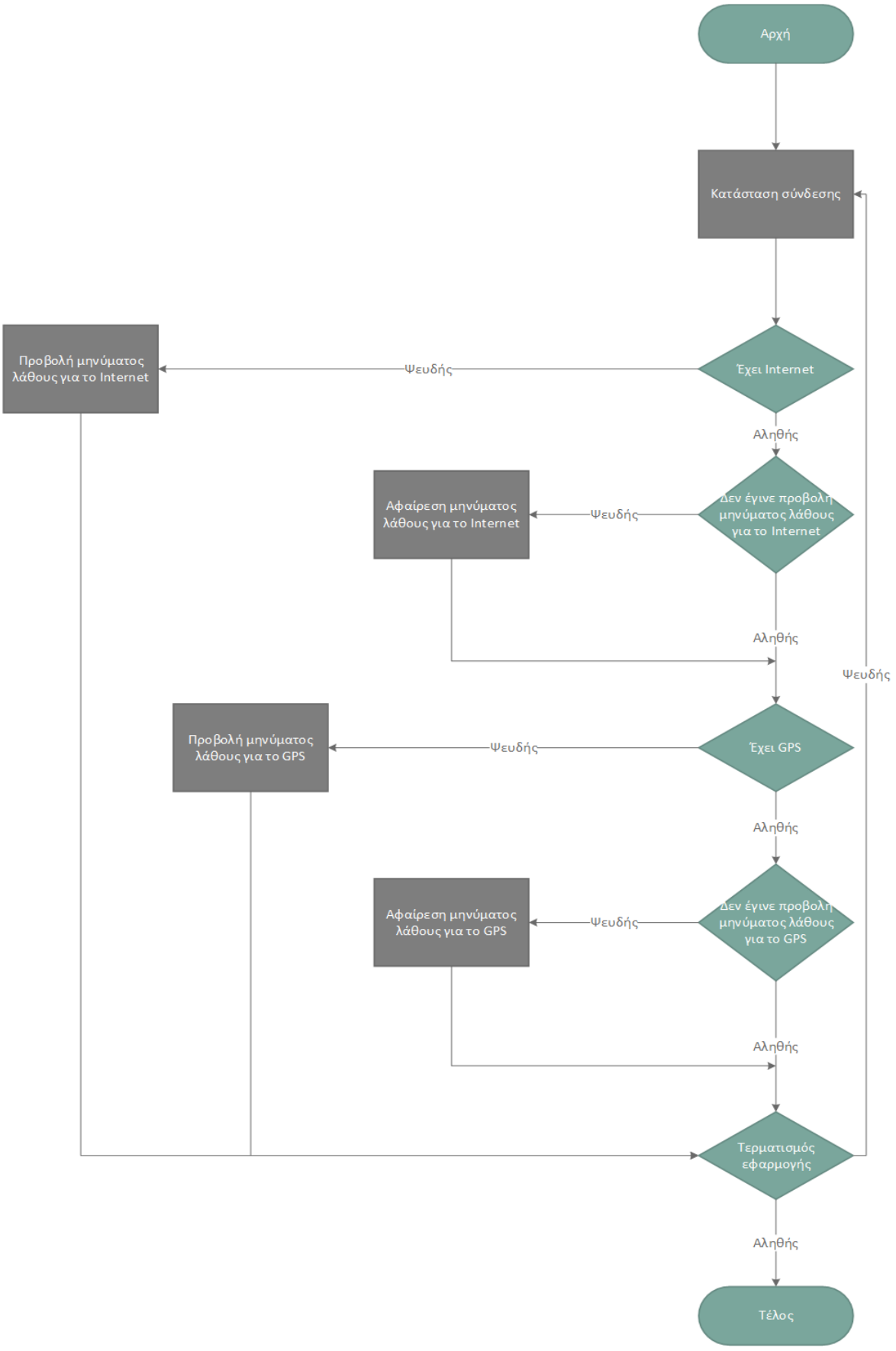
Η ειδοποίηση που διαλέγουμε να εμφανίσει δεν είναι η κλασσική ειδοποίηση που εμφανίζεται σε Android συσκευές. Στο Κεφάλαιο 5 παρουσιάζεται ενδεικτική εικόνα της ειδοποίησης που έχει δημιουργηθεί.

5. Έλεγχος σύνδεσης στο διαδίκτυο

Στο σχήμα 4.12 που ακολουθεί, βλέπουμε πως ακριβώς γίνονται οι έλεγχοι, μέσω κοινού διαγράμματος ροής που παρουσιάζει το thread, το οποίο υλοποιεί τον έλεγχο για το internet αλλά και για το GPS.

Γενικά, πλήρως λειτουργική εφαρμογή συνεπάγεται και συνδεσιμότητα στο διαδίκτυο. Για να εξασφαλιστεί αυτό ελέγχουμε συνεχώς τον χρήστη και τον πληροφορούμε αντιστοίχως. Σε περίπτωση που έχει υπάρξει κάποιο πρόβλημα στη σύνδεση του χρήστη στο διαδίκτυο, εμφανίζεται ένα πλαίσιο που τον πληροφορεί. Αν ο χρήστης πατήσει σε εκείνο το πλαίσιο, τότε τον οδηγεί στις αντίστοιχες ρυθμίσες για την ενεργοποίηση του Ίντερνετ της συσκευής του.

Για τον έλεγχο της σύνδεσης του Ίντερνετ χρησιμοποιούνται έτοιμες κλάσεις και μέθοδοι της java. Χρησιμοποιήσαμε την getActiveNetworkInfo(), η οποία είναι μέθοδος της κλάσης ConnectivityManager. Πρόκειται για μία κλάση που απαντά σε ερωτήματα που έχουν να κάνουν με τον τύπο της σύνδεσης ενός χρήστη. Ακόμη, μέσω αυτής, μπορούν να ενημερώνονται οι εφαρμογές για όταν αλλάζει η σύνδεση στο διαδίκτυο.



Σχήμα 4.12: Έλεγχος GPS και σύνδεσης στο διαδίκτυο

6. "Ανέβασμα" δεδομένων στον εξυπηρετητή (server)

Στο διάγραμμα 4.13 μπορούμε να δούμε τι γίνεται κάθε στιγμή που ο χρήστης κινείται και πότε ανεβαίνουν οι συντεταγμένες αυτού στον server. Επίσης, παρουσιάζεται και πως γίνεται η επιλογή της λειτουργίας της εφαρμογής, δηλαδή αν χρειάζεται να γίνει χρήση υψηλής ακρίβειας ή αν χρειάζεται εξοικονόμηση ενέργειας.

Όταν ο χρήστης πλέον βρίσκεται μέσα στο λεωφορείο και όλες οι προϋποθέσεις που αναφέρθηκαν και παραπάνω πληρούνται, αυτό που χρειάζεται να κάνει είναι να ανεβάσει τις συντεταγμένες του στον εξυπηρετητή. Αυτό υλοποιείται μέσω μίας διεργασίας που "τρέχει" συνεχώς στο παρασκήνιο (background) της συσκευής του, ενώ εκείνος δε χάνει καθόλου τη λειτουργικότητα του κινητού του τηλεφώνου. Επομένως, παράλληλα μπορεί να κάνει οτιδήποτε άλλο επιθυμεί καθώς και να στέλνει την τοποθεσία του στον διακομιστή.

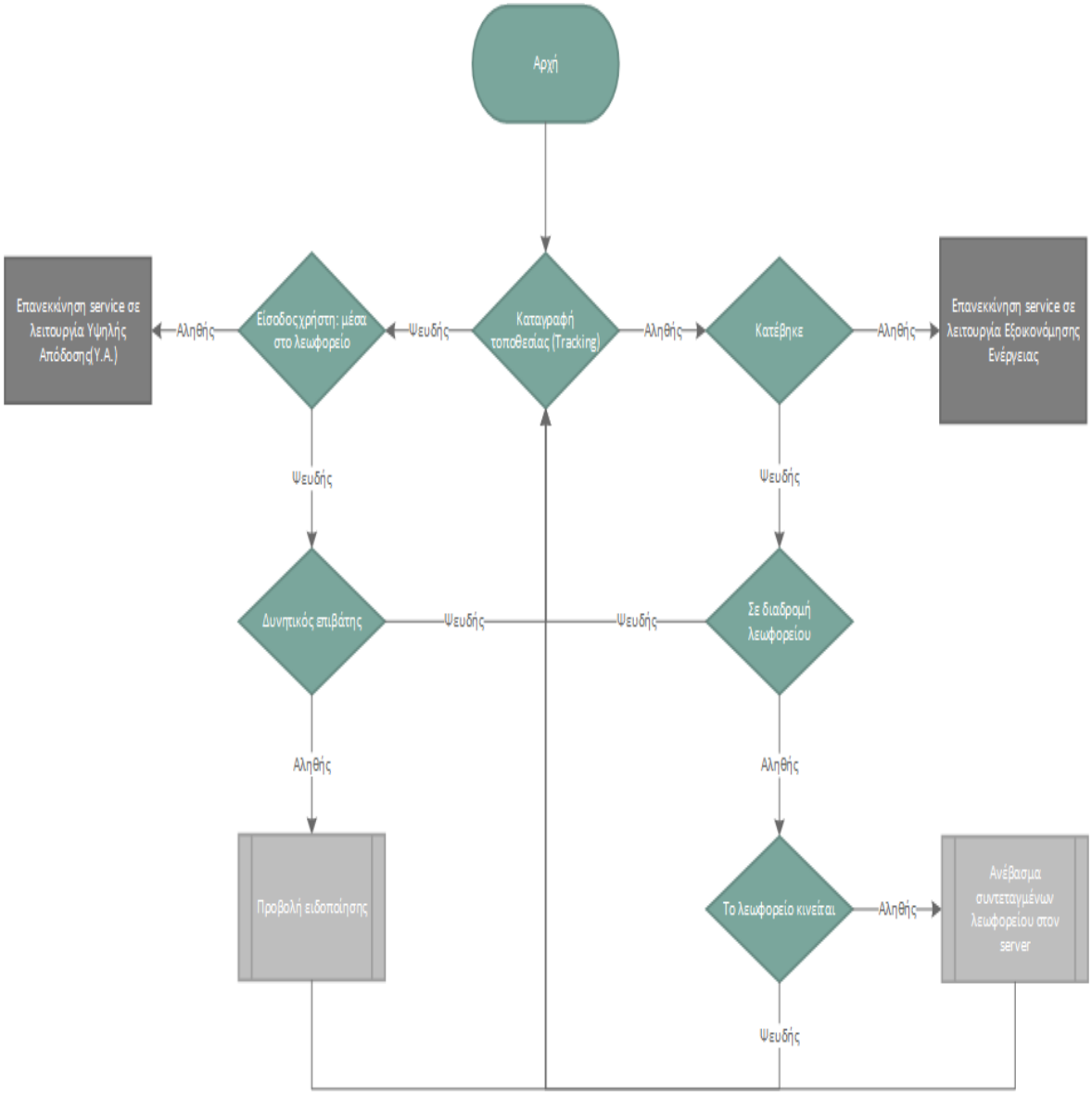
Όταν μιλάμε για "ανέβασμα" των δεδομένων στον server εννοούμε ουσιαστικά την ενημέρωση της βάσης δεδομένων με τις συντεταγμένες του επιβάτη - όπου τώρα ουσιαστικά ταυτίζονται με τη θέση του λεωφορείου - σε έναν πίνακα στη βάση δεδομένων. Αυτό ο πίνακας είναι ο BusLocations, ο οποίος αναλύθηκε και παραπάνω.

Η μεταφορά των δεδομένων στον server γίνεται μέσω της μεθόδου uploadLocationData(). Μέσω αυτής, φτιάχνουμε πάλι ένα αίτημα HTTP και το αποστέλλουμε στον Server. Αυτή τη φορά πρέπει να καταχωρήσουμε το URL αποστολής uploadWebsite = "http://ermis.mhl.tuc.gr/ vgioti/index.php" και τις εξής παραμέτρους:

- action = "setBusLocation"
- latitude
- longitude
- busName

Τέλος, καλούμε τη μέθοδο post της κλάσης MyHttpClient - που είναι μία υλοποίηση της HtttpClient, η οποία αποστέλλει το αίτημα για να ολοκληρωθεί η καταχώρηση στη βάση δεδομένων. Σε οποιαδήποτε περίπτωση σφάλματος, η εφαρμογή συνεχίζει να τρέχει και καταγράφουμε το σφάλμα στο Log.

Να τονισθεί εδώ ότι, ανεβαίνουν στον server οι τοποθεσίες τη στιγμή που το λεωφορείο βρίσκεται σε κίνηση. Όταν το λεωφορείο είναι ακίνητο, τότε δεν ανανεώνονται οι συντεταγμένες του. και στην περίπτωση που οι συντεταγμένες μείνουν ίδιες - ή το λεωφορείο μείνει ακίνητο - περισσότερο από δέκα (10) λεπτά, τότε σημαίνει ότι ο χρήστης αποβιβάστηκε από το λεωφορείο και σταματάει το "ανέβασμα" δεδομένων στον server.



Σχήμα 4.13: Ενέργειες εφαρμογής κάθε φορά που αλλάζει η τοποθεσία του χρήστη

7. Αποβίβαση χρήστη

Παρακάτω, παρουσιάζεται το διάγραμμα 4.14 όπου δείχνει ακριβώς τι έλεγχοι γίνονται για να αποφασίσουμε αν κάποιος κατέβηκε από το λεωφορείο.

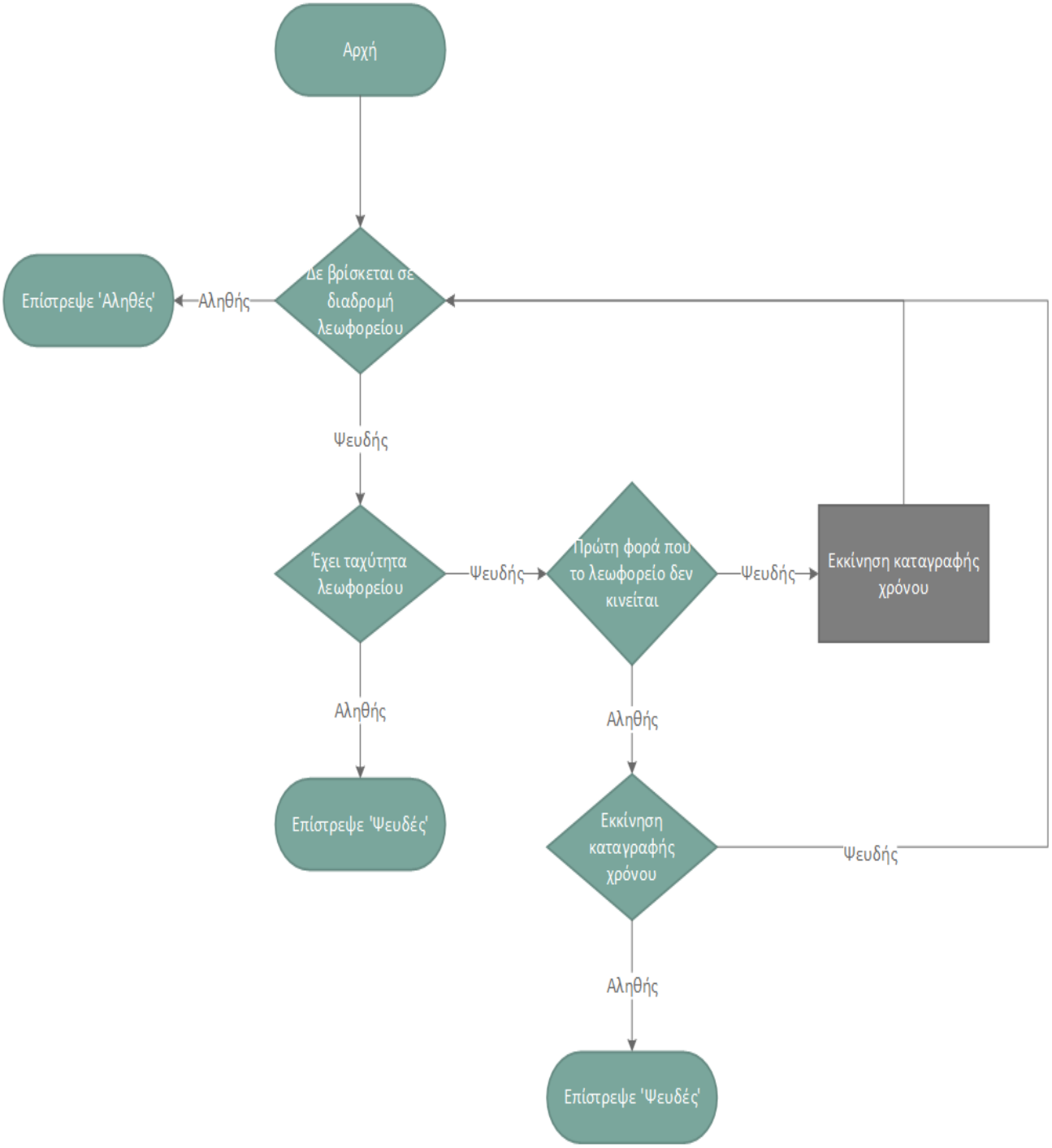
Μία από τις μεγαλύτερες προκλήσεις που κληθήκαμε να αντιμετωπίσουμε στην ανάπτυξη της παρούσας εφαρμογής είναι να μπορέσουμε να πούμε, με όσο το δυνατόν μεγαλύτερη ακρίβεια, πότε κάποιος χρήστης δεν είναι πλέον πάνω στο λεωφορείο. Αποφασίσαμε ότι η προβολή ενός καινούργιου notification ή ακόμη κι η προσμονή ότι ο ίδιος ο χρήστης θα θυμηθεί να πει πότε κατέβηκε, δεν ήταν ικανοποιητικές επιλογές. Έτσι, αποφασίστηκε να χρησιμοποιηθεί ένας τελείως αυτοματοποιημένος κι έξυπνος τρόπος, ο οποίος θα αναγνωρίζει μόνος του πότε κάποιος κατέβηκε από το λεωφορείο.

Ο αλγόριθμος ο οποίος αναφέρουμε, δεν είναι τίποτα άλλο από μία boolean μέθοδο, της οποία η επιστρεφόμενη τιμή εξαρτάται από τα εξής πράγματα:

1. Αν η απόσταση του σημείου που βρίσκεται τώρα ο χρήστης είναι μεγαλύτερη των 75 μέτρων σε σχέση με την απόσταση από κάποιο τη διαδρομή του λεωφορείου που βρισκόταν.

2. Αν η ταχύτητά του είναι μικρότερη των $12m/s$ για χρονικό διάστημα περισσότερο των 10 λεπτών.

Εάν πληρούνται οι παραπάνω δύο προϋποθέσεις, τότε μπορούμε να πούμε με βεβαιότητα ότι ο χρήστης έχει πλέον κατέβει από το λεωφορείο. Ο παραπάνω έλεγχος γίνεται μέσα στη μέθοδο gotOffBus. Δε χρειάζεται να πληρούνται και οι δύο προϋποθέσεις για να πούμε ότι κάποιος κατέβηκε. Αρκεί να μην είμαστε σε διαδρομή λεωφορείου και τότε εννοείται ότι δεν είμαστε πλέον επιβάτες λεωφορείου. Το ίδιο ισχύει και όταν οι συντεταγμένες μείνουν ίδιες για διάστημα περισσότερο των δέκα (10) λεπτών. Αυτό πάλι μάς οδηγεί στο συμπέρασμα ότι έγινε αποβίβαση από το λεωφορείο.



Σχήμα 4.14: Διάγραμμα ροής για την αποβίβαση ενός χρήστη από το λεωφορείο

8. Πολλαπλές λειτουργίες

Το πρόβλημα που μαστίζει σχεδόν όλες τις Android συσκευές είναι η ενέργεια. Συγκεκριμένα, δίχως να πέσει το επίπεδο της εφαρμογής μας, θέλουμε να κάνουμε όσο το δυνατόν μεγαλύτερη εξοικονόμηση ενέργειας. Για το λόγο αυτό, σκεφτήκαμε και

διαχωρίσαμε τις λειτουργίες της εφαρμογής μας σε δύο ξεχωριστά μέρη:

1. Λειτουργία εξοικονόμησης ενέργειας. Εδώ, βρίσκουμε τις περιπτώσεις όπου ο χρήστης δεν είναι σε λεωφορείο και δεν πρόκειται να είναι για τα επόμενα λεπτά κι έχουμε την εφαρμογή μας να "κοιμάται" καταναλώνοντας πολύ μικρά ποσοστά ενέργειας.

2. Λειτουργία υψηλής ακρίβειας. Σε αυτή την περίπτωση θέλουμε να καταγράψουμε με υψηλή ακρίβεια την τοποθεσίας μας, οπότε η συσκευή μας ξυπνάει και ως εκαι τούτου καταναλώνει και περισσότερη ενέργεια.

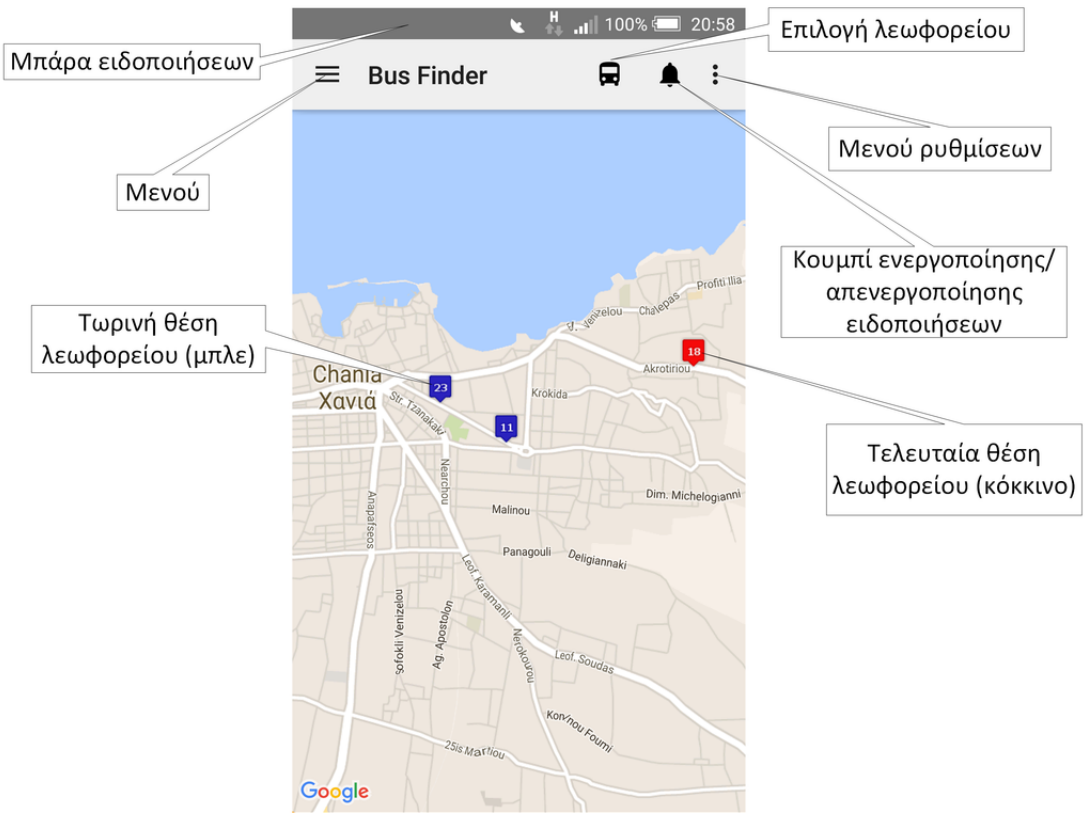
Γενικά, όμως, με αυτή τη μέθοδο καταφέρνουμε να περιορίσουμε κατά πολύ την κατανάλωση της μπαταρίας κι έτσι να μην παρατηρείται κατακόρυφη πτώση της μέσα σε λίγες ώρες, όπως συμβαίνει με άλλες εφαρμογές που χρησιμοποιούν τον δέκτη GPS.

4.4.2

Τέλος, αξίζει να αναφερθεί ότι μερικές φορές, λόγω μη επαρκής μνήμης στη συσκευή μας ή εξαιτίας και άλλων παραγόντων, το λειτουργικό σύστημα του κινητού ίσως "σκοτώσει" τη διεργασία της εφαρμογής μας. Στο BusFinder, λαμβάνουμε κι αυτό το σενάριο υπόψιν κι ακόμη κι αν το Android "σκοτώσει" την εφαρμογή μας, εκείνη θα "ξυπνήσει" ύστερα από ένα σύντομο χρονικό διάστημα. Ακόμη και όταν το κινητό κλείνει, με την επανεκκίνησή του, ξεκινά πάλι κι η εφαρμογή μας.

4.5 Γραφικό περιβάλλον εφαρμογής (GUI)

Με τον όρο γραφικό περιβάλλον εννοούμε οποιαδήποτε λειτουργικότητα έχει η εφαρμογή κι είναι ορατή στον χρήστη [34]. Στην εικόνα 4.15 βλέπουμε ένα στιγμιότυπο της εφαρμογής, όταν έχουμε επιλέξει να εμφανίζονται τρία λεωφορεία. Συγκεκριμένα, το μενού όπου είναι ουσιαστικά μία μπάρα εργαλείων. Μέσω αυτής της μπάρας ο κάθε χρήστης του BusFinder μπορεί να πλοηγηθεί στις βασικές λειτουργικότητες της εφαρμογής. Δόθηκε έμφαση στην απλότητα της εφαρμογής και κατά συνέπεια στη διευκόλυνση του χρήστη [35]. Ακόμη, υπάρχει κι ένα δευτερεύον - όπως θα μπορούσαμε να το ονομάσουμε - μενού, ένα Navigation Drawer. Τέλος, οι χάρτες που είναι κι εκείνοι που καταλαμβάνουν και το μεγαλύτερο μέρος της οθόνης.



Σχήμα 4.15: Ενδεικτικό στιγμιότυπο της εφαρμογής BusFinder

4.5.1 Μπάρα εργαλείων - Tool Bar

Στο πάνω μέρος της εφαρμογής βρίσκεται μία μπάρα εργαλείων. Εκεί είναι τοποθετημένα αρχικά ένα εικονίδιο, μέσω του οποίου γίνεται πρόσβαση στο navigation drawer. Στη συνέχεια, δίπλα ακριβώς από αυτό, βρίσκεται το όνομα της εφαρμογής μας, BusFinder. Ακολουθεί ένα εικονίδιο που μας παρουσιάζει ένα λεωφορείο. Ύστερα, έχουμε ένα εικονίδιο στο οποίο απεικονίζεται μία καμπάνα κι έτσι συμβολίζουμε τις ειδοποιήσεις. Τέλος, βρίσκεται ένα βοηθητικό μενού.

Αν κάνουμε κλίκαι στο πρώτο εικονίδιο - από αριστερά - τότε θα παρατηρήσουμε ένα navigation drawer. Με τον όρο navigation drawer δεν εννοούμε τίποτα άλλο παρά ένα πάνελ το οποίο βρίσκεται κρυμμένο συνήθως στην αριστερή πλευρά της οθόνης - στην περίπτωσή μας - και μας εξασφαλίζει πλοήγηση στις λειτουργίες της εκάστοτε εφαρμογής. Μπορούμε, λοιπόν, να το εμφανίσουμε είτε προσπαθώντας να "σύρουμε" την αριστερή άκρη της οθόνης, είτε πατώντας πάνω στο εικονίδιο που υποφηλώνει την ύπαρξη μενού σε αυτό το σημείο.

Πατώντας πάνω στο εικονίκιο που απεικονίζεται το λεωφορείο, βλέπουμε ότι δημιουργείται ένα pop-up παράθυρο. Μέσα σε αυτό εμφανίζεται μία λίστα με τα διαθέσιμα λεωφορεία που μπορούμε να εμφανίσουμε στον χάρτη. Μπορούμε να διαλέξουμε κανένα, ένα ή γενικά όσα θέλουμε. Αν το λεωφορείο έχει ανανεωθεί προσφάτως, τότε θα δούμε ένα μπλέ σημείο στον χάρτη, αλλιώς ένα κόκκινο. Κάνοντας κλικαι σε αυτό το σημείο, μπορούμε να πάρουμε ακόμη περισσότερες πληροφορίες.

Κάνοντας κλικαι πάνω στο εικονίδιο των ειδοποιήσεων μπορούμε να επιλέξουμε αν η εφαρμογή μας θέλουμε να σταματήσει να μας στέλνει ειδοποιήσεις. Αντίστοιχα, αν θέλουμε να ενεργοποιήσουμε και πάλι τη λήψη οδηγιών, τότε κάνουμε και πάλι κλικαι στο ίδιο εικονίδιο. Εμφανίζεται και το αντίστοιχο εικονίδιο για ενεργοποίηση ή απενεργοποίηση της υπηρεσίας καθώς κι ένα μήνυμα - toast - που μας ενημερώνει για τον ίδιο σκοπό.

Τέλος, το υπόλοιπο της οθόνης είναι ο χάρτης. Πάνω σε εκείνον παίρνουμε πληροφορίες για τα λεωφορεία που έχουμε επιλέξει, που βρίσκονται σε πραγματικό χρόνο, καθώς και τη διαδρομή που εκτελούν πάνω στον χάρτη.

Γενικά στο Android, ο σχεδιασμόw του layout για το interface της εφαρμογής μπορεί να γίνει με δύο τρόπους:

1. Δήλωση αντικειμένων του interface σε ένα XML αρχείο. Σε αυτή την περίπτωση δίδεται η δυνατότητα γράφοντας εντολές XML να επηρεάσουμε τις View κλάσεις, δηλαδή εκείνες που έχουν να κάνουν με την εμφάνιση της εφαρμογής. Θα μπορούσαμε να πούμε ότι αυτή είναι μία πιο έμμεση προσέγγιση.

2. Προγραμματιστικά, μέσω της java. Εδώ μπορούμε απευθείας να δημιουργήσουμε αντικείμενα των View και ViewGroup κλάσεων.

Εμείς διαλέξαμε την πρώτη μέθοδο, καθώς είναι εκείνη που προτείνεται κι επίσημα από την Google. Ενδεικτικά, μερικά από τα πλεονεκτήματα της πρώτης μεθόδου είναι:

- Διαχωρισμός του interface της εφαρμογής και του κώδικα που το διαχειρίζεται. Έτσι, μπορούν να γίνουν αλλαγές στο interface της εφαρμογής, χωρίς ωστόσο να επηρεαστεί ο κώδικας κι η λειτουργικότητα της εφαρμογής. Έτσι, μπορούν πολύ εύκολα να γίνουν τροποποιήσεις στην εφαρμογή όσων αφορά την υποστήριξη διαφορετικών τύπων οθονών, τροποποίηση όταν γίνεται στροφή 90° κ.α..

- Δηλώνοντας το interface σε XML, είναι πιο εύκολο να φανταστεί κάποιος πως θα δείχνει πραγματικά η εφαρμογή του. Έτσι, λοιπόν, το debugging γίνεται ευκολότερο, καθώς είναι πιο εύκολη η εύρεση ενός προβλήματος όταν έχεις στο μυαλό σου και τη γενική εικόνα της εφαρμογής.

Όλα τα εικονίδια που φορτώνουμε στην μπάρα, τα πήραμε από την επίσημη ιστοσελίδα της Google, όπου έχει ολόκληρη βιβλιοθήκη με εικονίδια. Άκόμη, από εκεί πληροφορηθήκαμε και τις βασικές αρχές που θα έπρεπε να ακολουθηθούν στο design της εφαρμογής.

Αν από την μπάρα πατήσουμε πάνω στο εικονίδιο με το λεωφορείο, τότε μας εμφανίζεται ένα παράθυρο για την επιλογή λεωφορείου. Πρόκειται για ένα AlertDialog.Builder, όπου δημιουργεί έναν builder για ένα alert dialog. Να σημειωθεί ότι χρησιμοποιούμε το default theme.

4.5.2 Εμφάνιση Χαρτών - Fragments

Για τη δημιουργία των χαρτών δημιουργήσαμε μία καινούργια κλάση, τη MyMapsFragment, η οποία κληρονομεί την SupportMapFragment, οποία με τη σειρά της κληρονομεί την Fragment. Γενικότερα, η χρήση ενός Fragment είναι ο πιο εύκολος τρόπος να τοποθετηθεί ένας χάρτης σε μία εφαρμογή. Τα fragments μπορούν να προστεθούν στο layout ενός activity σχετικά εύκολα.

Αρχικά, γίνεται κλήση της getMapAsync(OnMapReadyCallback), η οποία αρχικοποιεί το σύστημα των χαρτών και το view. Εδώ προσέχουμε ώστε να μην κρατάμε αντικείμενα - π.χ. markers - πέραν από τη ζωή των views. Αν κάτι τέτοιο γινόταν, υπήρχε κίνδυνος να είχαμε "διαρροές μνήμης" ή memory leaks όπως αποκαλούνται αλλιώς.

Τα Fragments πρώτη φορά εμφανίστηκαν στην έκδοση 3.0 του Android, με στόχο την υποστήριξη ενός πιο δυναμικού σχεδιασμού και καλύτερη υποστήριξη μεγαλύτερων οθονών, π.χ. tablets. Τα ίδια αντιπροσωπεύουν μία συμπεριφορά ή ένα κομμάτι του interface ενός activity. Σε ένα activity μπορούν να υπάρχουν πολλά fragments. Θα μπορούσαμε να πούμε ότι ένα fragment είναι ουσιαστικά ένα sub-activity, το οποίο μπορούμε να το χρησιμοποιήσουμε και σε άλλα activities.

Για να γίνει πιο κατανοητή η σημαντικότητα των fragments και κατά πόσο κάνουν τη σχεδίαση του UI καλύτερη σε σχέση με την εναλλαγή activities θα δοθεί ένα παράδειγμα.

Έστω ότι έχουμε μία συσκευή tablet με οθόνη 10". Ανοίγοντας ο χρήστης μία εφαρμογή που υποστηρίζει fragments, θα μπορεί να δει στην οθόνη του, στο ίδιο activity, δύο ή και τρία διαφορετικά fragments - π.χ. μία λίστα από όπου θα επιλέγει κάτι, ένα άλλο κείμενο κ.α.. Η ίδια εφαρμογή, όταν τρέχει σε κινητό, με οθόνη 4.5", λόγω έλλειψης χώρου, θα εμφανίσει μόνο ένα fragment. Έτσι, η υποστήριξη πολλών τύπων συσκευών είναι ένας από τους κυριότερους λόγους που μας οδήγησαν κι εμάς στην επιλογή χρήσης fragments.

Γενικά, υπάρχουν δύο τρόποι για την προσθήκη ενός fragment σε ένα activity.

1. Δήλωση του fragment μέσα στο layout αρχείο του activity.

2. Προγραμματιστικά, προσθήκη του fragment σε ένα υπάρχον ViewGroup.

Εμείς, επιλέξαμε να υλοποιήσουμε τον δεύτερο τρόπο. Σε αυτή την περίπτωση, για να κάνουμε "συναλλαγές" (transactions) fragment μέσα στο activity, πρέπει να χρησιμοποιηθούν API από το FragmentTransaction. Οπότε, για να ρυθμίσουμε τους χάρτες, αρχικά δημιουργούμε ένα instance και μετά ξεκινάμε τη "συναλλαγή" την επιβεβαιώνουμε με την κλήση του commit().

4.5.3 UI Threads

Δημιουργούμε και τρέχουμε παράλληλα με το κυρίως άλλα δύο thread. Ένα για τον έλεγχο των connection, εννοώντας την προσβασιμότητα στο διαδίκτυο καθώς και την κατάσταση του GPS. Επιπλέον, χρησιμοποιούμε ένα ακόμη thread για την ανανέωση των marker πάνω στον χάρτη.

Για την τοποθέτηση των markers χρησιμοποιούμε ένα νέο νήμα (thread) και όχι το πρωτεύον (main) thread. Με αυτόν τον τρόπο εξασφαλίζουμε ότι η εφαρμογή θα είναι πάντα λειτουργική, καθώς όταν ανοίγει ένα νέο thread κάνει παράλληλα τους υπολογισμούς που χρειάζονται για την τοποθέτηση των marker και όταν έχουν υπολογιστεί, τότε τοποθετούνται πάνω στους χάρτες. Βέβαια, τίποτα από αυτά δε γίνεται αντιληπτό στον χρήστη, παρά μόνο η σωστή λειτουργία των χαρτών.

Για την εμφάνιση του μηνύματος ότι δεν υπάρχει σύνδεση στο διαδίκτυο ή ότι δεν έχει ο χρήστης ενεργοποιήσει το GPS, τρέχει ένα ακόμη ξεχωριστό thread. Αυτό το thread τρέχει πάλι στο UI, καθώς τα μηνύματα λάθους θέλουμε να τα δούμε μόνο όταν έχουμε ανοιχτή την εφαρμογή, επομένως μάς ενδιαφέρει η δημιουργία και το τρέξιμο του εν λόγω thread μόνο όταν είναι ανοιχτή η εφαρμογή.

Για να γίνει περισσότερο κατανοητή η λειτουργία των threads, ας πάμε να δούμε πρώτα τον κύκλο "ζωής" ενός activity μιας εφαρμογής. Δημιουργώντας ένα activity, πρέπει οπωσδήποτε να συμπεριλάβουμε και τη μέθοδο onCreate(). Αυτή καλείται όταν δημιουργείται το activity. Μέσα σε αυτήν πρέπει να αρχικοποιήσουμε συγκεκριμένα σημαντικά στοιχεία που χρειάζεται η εφαρμογή για να τρέξει. Συγκεκριμένα, μέσα σε αυτή τη μέθοδο αρχικοποιούμε το layout, τα fragments για την προβολή των χαρτών, ξεκινάμε τον Alarm Manager, τις ειδοποιήσεις για interner και GPS και τέλος καλούμε τη μέθοδο startConnectionCheckThread(), όπου μέσω αυτής ξεκινάμε παράλληλα με το κυρίως νήμα (main thread), το thread για τον έλεγχο συνδέσεων ίντερνετ και GPS.

Ακόμη, μέσα σε ένα activity έχουμε και τη μέθοδο onPause(), η οποία καλείται όταν υπάρχει μία πρώτη ένδειξη ότι ο χρήστης εγκαταλείπει την εφαρμογή - χωρίς αυτό βέβαια να σημαίνει αναγκαστικά ότι το activity θα καταστραφεί. Εδώ είναι ουσιαστικά που πρέπει να αποθηκευτούν οποιεσδήποτε αλλαγές πριν την έξοδο του χρήστη. Εμείς εκεί αποθηκεύουμε την προηγούμενη θέση των markers, ώστε ο χρήστης αν για κάποιο λόγο έχει κλείσει την εφαρμογή κατα λάθος, να μπορείς να έχει πρόσβαση στις προηγούμενες επιλογές του. Επιπλέον, σταματούμε τα threads, ώστε να μην έχουμε κάποια διαρροή (leak) μνήμης.

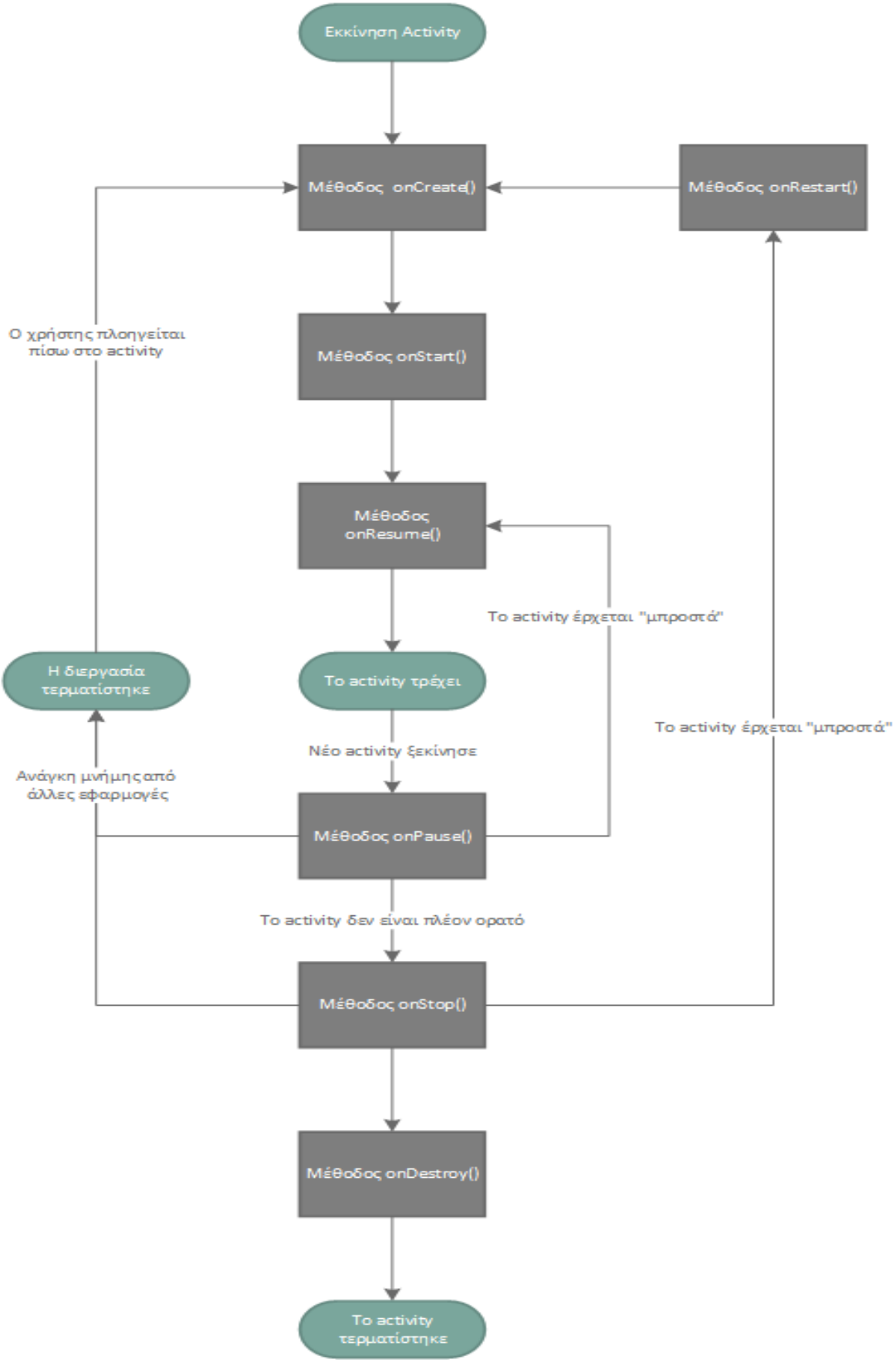
Στο διάγραμμα 4.16 που ακολουθεί μπορούμε να δούμε πότε καλούνται οι παραπάνω μέθοδοι και ποιος είναι ο τρόπος που λειτουργεί ένα activity.

Γενικά,τα νήματα τρέχουν μέσα από το κυρίως activity μιας εφαρμογής. Ελέγχονται από μία boolean μεταβλητή, όπου όταν η τιμή της είναι αληθής τότε το εν λόγω thread τρέχει ενώ όταν η τιμή της είναι ψευδής όχι. Όπως είπαμε και νωρίτερα, το πότε μπορεί να τρέχει η όχι γίνεται στις μεθόδους onCreate() και onPause().

Οι συνδέσεις ελέγχονται από ένα thread που τρέχει στο background της εφαρμογής κι ελέγχει συνεχώς αν υπάρχει σύνδεση στο ίντερνετ ή GPS συσκευή κι ενημερώνει

τον χρήστη για τυχόν έλλειψη ενός εκαι των δύο - ή και των δύο - εμφανίζοντας ένα κουμπί στην οθόνη, όπου πατώντας του οδηγείται στις ρυθμίσεις για να διορθώσει ό,τι από τα δύο δεν έχει ενεργοποιήσει. Το εν λόγω thread ξεκινάει στην onCreate() και σταματά στην onPause() ή στην onDestroy() αν τρέχει ακόμη. Ξαναρχίζει στη μέθοδο onResume() όταν ενεργοποιείται το activity και πάλι.

Σε κάθε τρέξιμο/κύκλο το thread καλεί την κλάση μας για τις συνδέσεις και λαμβάνει την τωρινή κατάσταση για το ίντερνετ και το GPS.



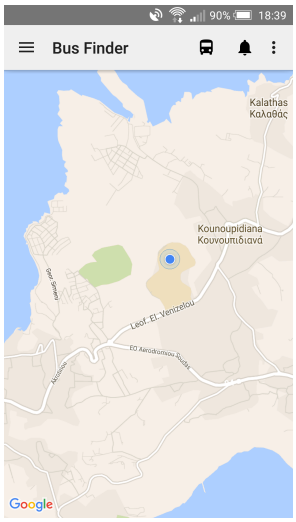
Σχήμα 4.16: Κύκλος ζωής ενός activity

Κεφάλαιο 5

Δοκιμές - Έλεγχοι

5.1 Χρήση εφαρμογής

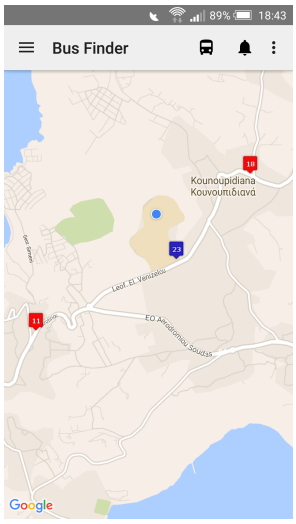
Όταν ο χρήστης ανοίξει την εφαρμογή BusFinder, αυτό που θα δει είναι οι χάρτες. Επιπλέον, στο πάνω μέρος της οθόνης θα του δίδεται ένα μενού με βασικές λειτουργίες της εφαρμογής, όπως επιλογή λεωφορείου, απενεργοποίηση ειδοποιήσεων, ρυθμίσεις και δευτερεύον μενού. Στην εικόνα 5.1 που ακολουθεί βλέπουμε την αρχική οθόνη του BusFinder, όταν κάποιος ανοίξει την εφαρμογή για πρώτη φορά.



Σχήμα 5.1: Αρχική οθόνη εφαρμογής

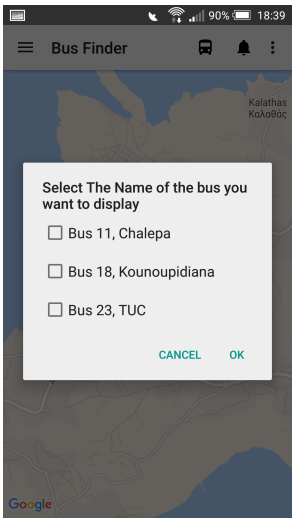
Όταν κάποιος πατήσει το εικονίδιο που απεικονίζει ένα λεωφορείο, με σκοπό την επιλογή ενός συγκεκριμένου λεωφορείου θα συνειδητοποιήσει ότι το BusFinder σχε-

διάστηκε με τέτοιο τρόπο, ώστε ο χρήστης να μπορεί να εμφανίσει όσα λεωφορεία θέλει εκείνη τη στιγμή στην οθόνη του και όχι μόνο ένα λεωφορείο, όπως συνήθως προσφέρουν άλλες εφαρμογές [17] [18]. Βλέπουμε, λοιπόν, ότι θα του εμφανιστεί ένα αναδυόμενο παράθυρο που περιλαμβάνει μία λίστα με τα διαθέσιμα λεωφορεία για να επιλέξει. Μπορεί να επιλέξει όσα λεωφορεία επιθυμεί. Στην εικόνα 5.3 φαίνεται αναλυτικά αυτή η λειτουργικότητα της εφαρμογής, που μόλις περιγράφηκε.

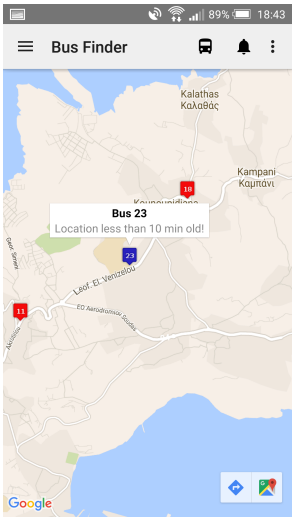


Σχήμα 5.2: Εμφάνιση τριών λεωφορείων πάνω στον χάρτη

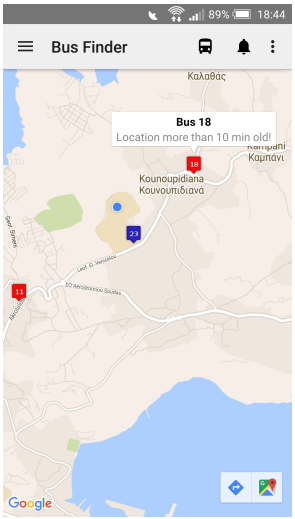
Εάν ο χρήστης επιλέξει να εμφανίσει όλα τα διαθέσιμα λεωφορεία θα δει στην οθόνη του το σχήμα 5.2. Από την εικόνα 5.2 είναι εύκολο να παρατηρήσουμε ότι γίνεται διαχωρισμός, βέβαια, της τωρινής και της παλιάς θέσης ενός λεωφορείου, χρησιμοποιώντας διαφορετικά χρώματα στα εικονίδια για τον λόγο αυτό. Με τον όρο παλιά θέση ενός λεωφορείου, εννοούμε ένα λεωφορείο όπου δεν έχει ανανεωθεί η θέση του από κάποιον επιβάτη για χρονικό διάστημα περισσότερο των δέκα (10) λεπτών. Έτσι, μπορούμε να παρατηρήσουμε στις εικόνες 5.5 και 5.4, ότι όλες οι παλιές θέσεις εμφανίζονται με κόκκινο χρώμα ενώ οι προσφάτως ανανεωμένες με μπλε.



Σχήμα 5.3: Μενού για την επιλογή λεωφορείου/λεωφορείων



Σχήμα 5.4: Μήνυμα για την μπλε ένδειξη λεωφορείου στον χάρτη



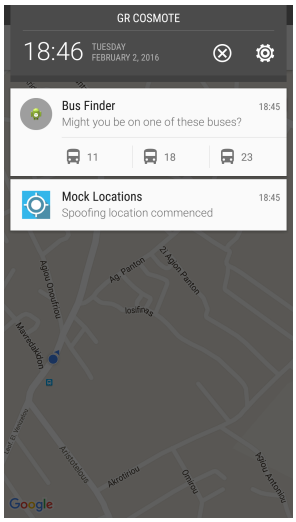
Σχήμα 5.5: Μήνυμα για την κόκκινη ένδειξη λεωφορείου

Βλέπουμε, δηλαδή, ότι δίδεται στον χρήστη η δυνατότητα να ανοίγει χάρτες στο κινητό του, όπου κάθε τρία (3) δευτερόλεπτα ανανεώνεται η θέση του λεωφορείου ή των λεωφορείων που έχει επιλέξει πάνω σε αυτούς.

Όμως, πέραν της λειτουργικότητας που είναι ορατή, υπάρχει κι αρκετή λειτουργικότητα που δεν είναι ορατή στον χρήστη καθώς γίνεται στο παρασκήνιο. Συγκεκριμένα, στην περίπτωση όπου ο χρήστης βρίσκεται μέσα σε κάποιο λεωφορείο, εκτελείται ο αλγόριθμος που αναλύθηκε στο κεφάλαιο της υλοποίησης, όπου αναγνωρίζει αυτό το

ενδεχόμενο κι ερωτάται ο χρήστης μέσω μίας ειδοποίησης. Ακόμη, για την καλύτερη διευκόλυνση του χρήστη, στο ίδιο παράθυρο ερωτάται και μέσα σε ποιο λεωφορείο είναι. Επομένως, ο χρήστης με ένα μόνο κλικαι μπορεί να δηλώσει ότι βρίσκεται μέσα στο εκάστοτε λεωφορείο.

Στην εικόνα 5.6 μπορούμε να παρατηρήσουμε ότι εμφανίζονται τα τρία πιο πιθανά λεωφορεία στα οποία μπορεί να βρίσκεται μέσα ο χρήστης. Ο τρόπος που αυτό γίνεται έχει αναλυθεί εκτενώς στο κεφάλαιο της υλοποίησης της εφαρμογής.

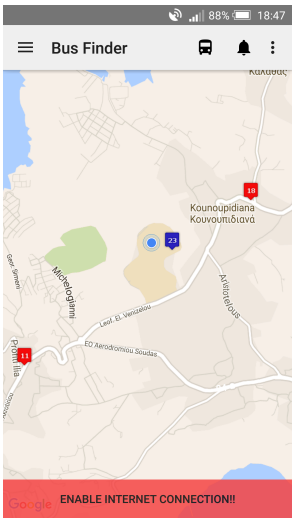


Σχήμα 5.6: Ειδικά τροποποιημένη ειδοποίηση εφαρμογής

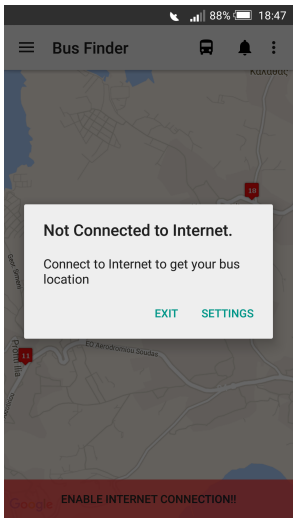
Ακόμη, για την ομαλή λειτουργία της εφαρμογής σκεφτήκαμε ότι πρέπει να γίνονται διάφοροι έλεγχοι. Συγκεκριμένα, δίχως τη χρήση διαδικτύου, δεν είναι δυνατή η επικοινωνία μεταξύ της συσκευής και του απομακρυσμένου διακοσμιτή. Έτσι, ελέγχουμε πάντα αν ο χρήστης είναι συνδεδεμένος ή όχι στο διαδίκτυο. Αν δεν είναι, εμφανίζεται στην οθόνη κατάλληλο μήνυμα που τον προτρέπει να συνδεθεί.

Στην εικόνα 5.7 βλέπουμε ότι αν ο χρήστης ανοίξει την εφαρμογή και δεν έχει ενεργοποιήσει τη χρήση ίντερνετ τότε εμφανίζεται στο κάτω μέρος της οθόνης μήνυμα που τον πληροφορεί. Πατώντας πάνω στο κόκκινο πλαίσιο ο χρήστης, πλοηγείται στο μενού ρυθμίσεων της κινητής του συσκευής ώστε να ενεργοποιήσει τη χρήστη ίντερνετ, όπως μπορούμε να δούμε και στην εικόνα 5.8.

Ένας ακόμη έλεγχος που γίνεται είναι εκείνος για την ενεργοποίηση της τοποθεσίας. Δίχως ο χρήστης να έχει ενεργοποιήσει τη ρύθμιση για την λήψη της τοποθεσίας, δεν

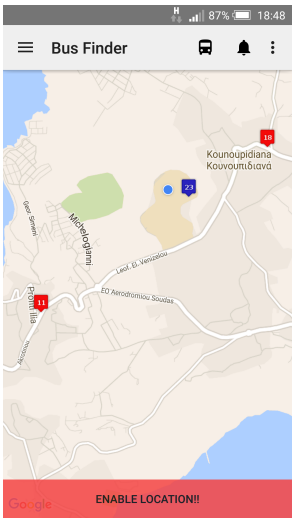


Σχήμα 5.7: Εμφάνιση προειδοποιητικού μηνύματος/ κουμπιού για την ενεργοποίηση του ίντερνετ

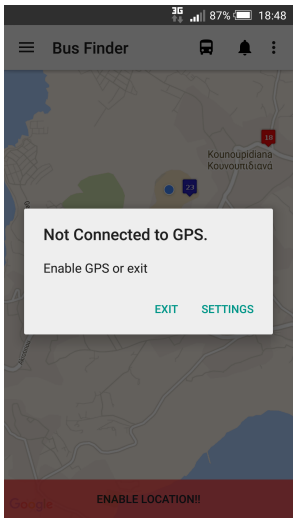


Σχήμα 5.8: Μήνυμα για τη μετάβαση στις ρυθμίσεις για την ενεργοποίηση του ίντερνετ

είναι δυνατή η συνεισφορά του χρήστη αν βρίσκεται μέσα σε κάποιο λεωφορείο. Έτσι, λοιπόν, και σε αυτή την περίπτωση θα του εμφανιστεί το κατάλληλο μήνυμα, όπου θα τον οδηγεί και στις αντίστοιχες ρυθμίσεις για ενεργοποίηση της τοποθεσίας της συσκευής. Στις εικόνες 5.9 και 5.10, ακολουθούν αντίστοιχα παραδείγματα που παρουσιάζουν αυτές τις λειτουργίες.



Σχήμα 5.9: Εμφάνιση προειδοποιητικού μηνύματος/ κουμπιού για την ενεργοποίηση του GPS



Σχήμα 5.10: Μήνυμα για τη μετάβαση στις ρυθμίσεις για την ενεργοποίηση του GPS

5.2 Πιθανά σενάρια για έλεγχο λειτουργικότητας εφαρμογής

Χρειάστηκε να γίνει μία σειρά από ελέγχους για να βεβαιωθούμε ότι η εφαρμογή μας ήταν πλήρως λειτουργική. Δουλέψαμε πάνω σε πιθανά σενάρια με αληθινές διαδρομές λεωφορείων του δήμου Χανίων [36]. Οι διαδρομές αποθηκεύτηκαν σε στον server με τις συντεταγμένες τους. Εξετάσαμε τα πιθανά προβλήματα που μπορεί να αντιμετωπίσει ο κάθε χρήστης της εφαρμογής κι αξιολογήσαμε τα αποτελέσματα που λάβαμε από το κάθε σενάριο.

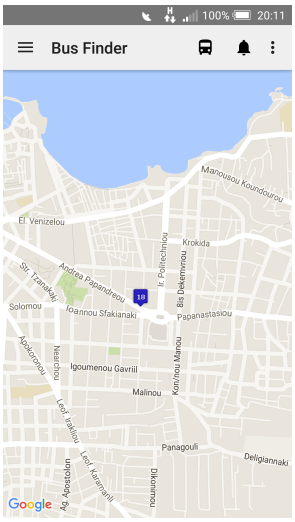
Στην εικόνα 5.11 φαίνεται μία παρουσιάση που είχε γίνει στο Πολυτεχνείο Κρήτης, παρουσιάζοντας όλα τα πιθανά σενάρια και τα αποτελέσματα που λάβαμε, χρησιμοποιώντας προσομοιωτή για την κίνηση του λεωφορείου. Συγκεκριμένα, στην εικόνα παρατηρούμε τρία (3) tablets, όπου χρησιμοποιώντας το πρόγραμμα MockLocations, θεωρήσαμε ότι είναι τα τρία διαφορετικά λεωφορεία μας. Στη συνέχεια, με τα κινητά που φαίνονται στην οθόνη, μέσω της εφαρμογής, ελέγχαμε τη θέση των λεωφορείων που μάς ενδιέφεραν. Τέλος, να αναφερθεί ότι για το συγκεκριμένο demo, δημιουργήθηκε και μία σελίδα online, στον server, όπου μπορεί ο καθένας ανοίγοντας τον περιηγητή του να δει που βρίσκονται τα λεωφορεία εκείνη τη στιγμή[1]. Παρακάτω θα αναφερθούμε στα σημαντικότερα από αυτά και στα αποτελέσματα και συμπεράσματα που εκλάβαμε από το κάθε σενάριο ξεχωριστά.

[1] http://ermis.mhl.tuc.gr/~vgioti/phpsqlajax_map_v3.html

Σχήμα 5.11: Εικόνα από την παρουσίαση που έγινε στο Πολυτεχνείο Κρήτης

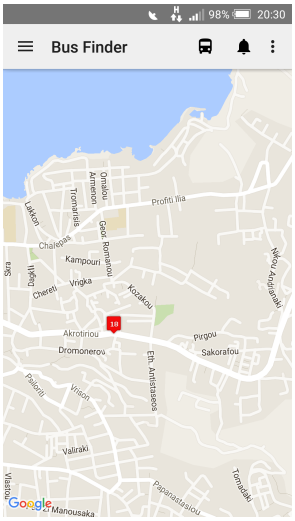
- Σενάριο 1: Ένας χρήστης στη στάση

Αρχικά, θα ελέγξουμε τη βασική λειτουργικότητα της εφαρμογής. Για να το επιτύχουμε αυτό θα υλοποιήσουμε το εξής απλό σενάριο: έστω ότι έχουμε έναν επιβάτη της εφαρμογής στον λεωφορείο "18 - Κουνουπιδιανά" κι έναν που βρίσκεται στη στάση στην πλατεία των Κουνουπιδιανών. Ο τελευταίος, θέλοντας να πληροφορηθεί για τη θέση του λεωφορείου, ανοίγει την εφαρμογή και διαπιστώνει ότι το λεωφορείο βρίσκεται ακόμη στα Χανιά, όπως μπορούμε να δούμε κι απο το στγμιότυπο 5.12. Την τοποθεσία την παίρνει από τον επιβάτη που βρίσκεται μέσα σε αυτό το λεωφρείο και κατευθύνεται προς Χαλέπα.



Σχήμα 5.12: Τωρινή θέση λεωφορείου "18 - Κουνουπιδιανά"

Στην εικόνα 5.13 βλέπουμε την περίπτωση όπου ο χρήστης που περιμένει στη στάση, ανοίγει την εφαρμογή του όταν δεν υπάρχει μέσα στο λεωφορείο κάποιος επιβάτης - χρήστης της εφαρμογή. Έτσι. τότε βλέπει με κόκκινη ένδειξη την τελευταία θέση του λεωφορείου - εκεί που κατέβηκε ο τελευταίος χρήστης της εφαρμογής - και όχι την τωρινή.

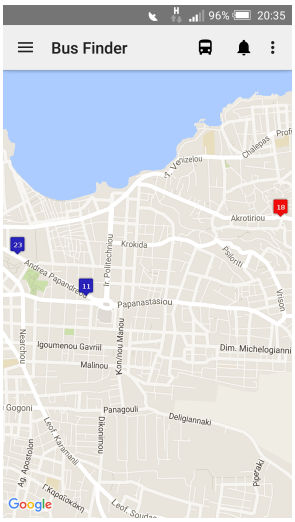


Σχήμα 5.13: Τελευταία -μη ανανεωμένη - θέση λεωφορείου "18 - Κουνουπιδιανά"

- Σενάριο 2: Πολλαπλά λεωφορεία για τον ίδιο προορισμό

Σε αυτό το σενάριο, κάποιος στη στάση των Δικαστηρίων των Χανίων θέλει να πάρει το λεωφορείο, ώστε να κατευθυνθεί προς την Ακρωτηρίου. Συνειδητοποιεί ότι για

τον συγκεκριμένο προορισμό τον εξυπηρετούν τρία λεωφορεία (11, 18, 23). Επομένως, επιλέγει να εμφανίσει στην οθόνη του πού βρίσκονται και τα 3 λεωφορεία. Όπως φαίνεται και στην εικόνα 5.14 μόνο τα δύο από αυτά είναι προσφάτως ανανεωμένα κι αυτό που έρχεται γρηγορότερα είναι το λεωφορείο "11 - Χαλέπα".



Σχήμα 5.14: Θέση λεωφορείων 11, 18 και 23

- Σενάριο 3: Ανανεώση ενός λεωφορείου συνεχώς από πολλούς επιβάτες

Εδώ εξετάσαμε το ενδεχόμενο να ανεβαίνουν επιβάτες για μικρό χρονικό διάστημα στο λεωφορείο, εξασφαλίζοντας ότι πάντα θα είναι κάποιος χρήστης της εφαρμογής μέσα στο λεωφορείο. Έτσι, όταν κάποιος βρίσκεται μέσα στο λεωφορείο, ανανεώνεται συνεχώς η θέση του. Όταν κατεβαίνει και κάποιος άλλο επιβιβάζεται, δεν υπάρχει κανένα πρόβλημα, καθώς η θέση του λεωφορείου στον χάρτη συνεχίζει να ανενεώνεται σταθερά, εξαιτίας του δεύτερου επιβάτη. Έτσι, αν κάποιος βρίσκεται στη στάση, θα παρατηρήσει απλώς τη θέση του λεωφορείου που κινείται χωρίς κανένα απολύτως πρόβλημα.

- Σενάριο 4: Πολλαπλοί επιβάτες

Σε ένα λεωφορείο έχουμε πολλούς επιβάτες - χρήστες της εφαρμογής και κάποιον στη στάση που περιμένει το λεωφορείο. Αν μερικοί από αυτούς κατέβουν, εκείνος που βρίσκεται στη στάση συνεχίζει να λαμβάνει τη θέση του λεωφορείου χωρίς να επηρεάζεται από τον αριθμό των επιβατών στο λεωφορείο. Γενικά, αρκεί έστω κι ένας επιβάτης για τον προσδιορισμό της θέσης ενός λεωφορείου.

- Σενάριο 5: Χρήση αυτοκινήτου και λεωφορείου

Έστω ότι έχουμε μία εργαζόμενη μητέρα, που δεν προλαβαίνει να πάει το παιδί στο σχολείο στα Κουνουπιδιανά, οπότε το αφήνει στην κοντινότερη στάση. Το παιδί της κι η ίδια είναι χρήστες της εφαρμογής. Οδηγώντας προς τη στάση, θα τους εμφανιστεί η ειδοποίηση που θα τους ρωτά αν βρίσκονται μέσα σε κάποιο λεωφορείο. Η μητέρα την αγνοεί, διότι οδηγεί, ενώ το παιδί απαντά αρνητικά.

Στη συνέχεια, η μητέρα αφήνει το παιδί της στη στάση και κατευθύνεται στη δουλειά της. Μετά από κάποια λεπτά το παιδί επιβιβάζεται στο λεωφορείο και του εμφανίζεται ξανά η ειδοποίηση, οπότε απαντά θετικά. Η μητέρα, μπορεί απο τη δουλειά της να δει αν τελικά πέρασε το λεωφορείο κι επίσης στο σχόλασμα να πληροφορηθεί τι ώρα θα είναι το εν λόγω λεωφορείο στη στάση, ώστε να μπορέσει παραλάβει το παιδί της.

- Σενάριο 6: Ψευδής δήλωση τοποθεσίας

Έστω ότι κάποιος ξεχνά να απαντήσει στην ειδοποίηση που έλαβα. Βλεποντάς την μετά από μεγάλο χρονικό διάστημα, όπου δε βρίσκεται μέσα σε κάποιο λεωφορείο, αποφασίζει να απαντήσει θετικά - είτε εσφαλμένα είτε εξ επίτηδες. Τότε, λαμβάνοντας υπόψιν κι ένα τέτοιο ενδεχόμενο, προφανώς και δε θα ανανεωθεί καμία θέση λεωφορείου.

Κεφάλαιο 6

Συμπεράσματα - Μελλοντικές εργασίες

6.1 Σύνοψη

Στην εποχή μας, όπου τα πάντα εκτελούνται με εξωφρενικά γοργούς ρυθμούς και το κάθε λεπτό είναι πολύτιμο, δημιουργήσαμε μία εφαρμογή που δίνει λύση στην άσκοπη αναμονή για την άφιξη των Μέσων Μαζικής Μεταφοράς. Οποιοσδήποτε θα μπορεί να δει στο κινητό του τηλέφωνο που βρίσκονται τα μέσα μεταφοράς πάνω στον χάρτη, σε πραγματικό χρόνο.

Το BusFinder, όμως, πέραν της εξοικονόμησης χρόνου, προσφέρει πολύ περισσότερα. Συγκεκριμένα, αναβαθμίζεται η εμπειρία των επιβατών των μέσων μαζικής μεταφοράς, αφού ο κυριότερος λόγος που αποφεύγεται η χρήση των τελευταίων είναι η μη συνέπεια που παρουσιάζεται στα δρομολόγιά τους. Ακόμη, με την ενθάρρυνση των μέσων μεταφοράς, συντελούμε έμπρακτα και βοηθάμε στην αποφυγή της περαιτέρω μόλυνσης του πλανήτη μας, εξαιτίας της υπερβολικής χρήσης αυτοκινήτων. Τέλος, διατίθεται εντελώς δωρεάν, καθώς στηρίζεται στην ιδέα του πληθοπορισμού. Ο πληθοπορισμός είναι μία ιδέα που προάγει την ευαισθησία προς το κοινωνικό σύνολο και τον αλτρουισμό, αφού οι ίδιοι οι χρήστες οικειοθελώς μπορούν και παρέχουν πληροφορίες για την εξυπηρέτηση των συνανθρώπων τους.

Έχοντας την αρχική ιδέα και τι θα προσφέρει, αποφασίσαμε ότι ήταν αναγκαία η υλοποίησή της. Αρχικά, για τον λόγο αυτό, μελετήσαμε τις ήδη υπάρχουσες λύσεις που προτείνονται από άλλους φορείς. Στον τομέα αυτό υπήρχαν τόσο επαγγελματικές λύσεις

με υψηλό κόστος, αλλά κι εφαρμογές με μηδενικό κόστος κατασκευής. Σχεδιάστηκε η εφαρμογή, πως θα υλοποιηθούν οι βασικές λειτουργίες της, πως θα πρέπει να εργαστούμε καθώς και άλλα πράγματα που αφορούσαν την ανάπτυξη. Στη συνέχεια, προχωρήσαμε στην ανάπτυξη της, όπου συνεχώς ελέγχαμε τη λειτουργικότητα σε κάθε κομμάτι που τελείωνε. Στο τέλος, έγιναν εκτενείς έλεγχοι μέσω πιθανών σεναρίων, με στόχο να ανακαλυφθούν σημαντικά προβλήματα - όπου διορθώθηκαν - αλλά και μερικές ελλείψεις.

6.2 Βελτιώσεις

Με το πέρας της εφαρμογής διαπιστώθηκε ότι δεν υπάρχει όρια στα πράγματα που μπορούν να γίνουν και στις λειτουργίες που μπορεί να προσφέρει μία εφαρμογή κινητού τηλεφώνου, και όπως συνηθίζεται να λένε "ο ουρανός είναι το όριο"[1]. Θα μπορούσαμε, όμως, να αναφερθούμε σε κάποιες από τις λειτουργίες που δεν συμπεριλήφθηκαν στην πρώτη έκδοση της εφαρμογής και θα ήταν θεμιτό να προσθεθούν στις επόμενες.

Αρχικά, το interface της εφαρμογής δεν μελετήθηκε εκτενώς κι έτσι είναι αρκετά φτωχό. Θα μπορούσαν να γίνουν αρκετές βελτιώσεις πάνω σε αυτό το κομμάτι, μελετώντας τους κανόνες του Nielsen για την ευχρηστία που πρέπει που πρέπει να χαρακτηρίζει το γραφικό περιβάλλον μίας εφαρμογής [37]. Ακόμη, λαμβάνοντας υπόψιν εκείνα τα οποία είχε διατυπώσει κι η D.Mayhew για τις αρχές που διέπουν τον σχεδιασμό του UI [38], θα μπορούσαν να γίνουν πιο τολμηρές επιλογές χρωμάτων.

Πέραν της προβολής των λεωφορείων που μας ενδιαφέρουν πάνω στον χάρτη, θα μπορούσε να γίνεται και μία εκτίμηση του χρόνου άφιξή τους στο σημείο που στεκόμαστε. Έχοντας το αντίστοιχο ETA, ο χρήστης είναι σε θέση ευκολότερα να προγραμματίσει διάφορες εργασίες του κ.λπ.

Η εφαρμογή προς το παρόν απευθύνεται μόνο στην φοιτητική κοινότητα του Πολυτεχνείου Κρήτης, εξυπηρετώντας μόνο τρία (3) λεωφορεία, τα οποία είναι κι αυτά που παίρνουν οι περισσότεροι φοιτητές. Θα μπορούσε να αναβαθμιστεί καλύπτοντας πολλές διαφορετικές γραμμές καθώς και πολλά λεωφορεία της ίδιας γραμμής - που δεν υλοποιείται έως τώρα.

[1]The sky is the limit. Αμερικανική έκφραση όπου θέλει να δείξει ότι δεν υπάρχουν όρια

Ακόμη, θα μπορούσε με κάποιον έξυπνο τρόπο να καλύπτει πιθανά "κενά", δίνοντας κάποιες εκτιμήσεις για την πιθανή θέση του λεωφορείου εκείνη τη στιγμή. Όταν δεν υπάρχει κάνενας μέσα στο λεωφορείο, δηλαδή, για τα πρώτα λεπτά να μπορεί να δώσει μία εκτίμηση στον χρήστη για τη θέση του εκείνη τη στιγμή.

Συμπερασματικά, όλοι μας γνωρίζουμε ότι η χρήση των μέσων μαζικής μεταφοράς στα μικρά αστικά κέντρα είναι ελάχιστη εώς μηδαμινή, με κύρια δικαιολογία όλων την ασυνέπεια που παρατηρείται στα δρομολόγια των λεωφορείων. Αντιλαμβανόμαστε, λοιπόν, ότι εγκαταλείποντας τόσοι άνθρωποι τα μέσα μεταφοράς και χρησιμοποιώντας αυτοκίνητα για τις καθημερινές τους μετακινήσεις, οι ατμοσφαιρικοί ρύποι αυξάνονται ραγδαία με αποτέλεσμα τη σταδιακή καταστροφή του περιβάλλοντος. Έτσι, η δημιουργία του BusFinder και των παρόμοιων εφαρμογών [39] στοχεύει στην αναβάθμιση της εμπειρίας των επιβατών των μέσω μαζικής μεταφοράς, προσφέροντας άμεση πληροφόρηση για τη θέση του λεωφορείου. Με αυτόν τον τρόπο συμβάλλει στον τερματισμό των ανούσιων αναμονών στις στάσεις των λεωφορείων, ενισχύοντας την καλυτέρευση της καθημερινότητας του επιβατικού κοινού αλλά και τη μείωση της εκπομπής των ρύπων από τα τα Ι.Χ..

Η προστασία του πλανήτη μας είναι ευθύνη όλων μας. Αν κάτι τέτοιο μπορεί να επιτευχθεί παράλληλα με την άμεση καλυτέρευση της καθημερινότητας των ατόμων και χρησιμοποιώντας την τεχνολογία ως σύμμαχό μας, τότε αυτή είναι κι η μεγαλύτερη επιτυχία για έναν μηχανικό.

Βιβλιογραφία

[1] John Neff. A Profile of Public Transportation Passenger Demographics and Travel Characteristics Reported in On-Board Surveys. 2007.

[2] Ross Dawson. Getting Results From Crowds: The definitive guide to using crowdsourcing to grow your business. Advanced Human Technologies, Incorporated, 2nd edition, 2012.

[3] Adam Winstanley Bashir Shalaik. Delivering Real-Time Bus Tracking Information on Mobile Devices. 2011.

[4] Daniel J. Dailey Stuart D. Maclean. Real-time Bus Information on Mobile Devices. 2012.

[5] Aaron Steinfeld John Zimmerman Anthony Tomasic Daisy Yoo Rafae Dar Aziz. Mobile Transit Rider Information Via Universal Design and Crowdsourcing. 2011.

[6] Aditi Misra Aaron Gooze Kari Watkins Mariam Asad Christopher A. Le Dantec. Crowdsourcing and Its Application to Transportation Data Collection and Management. 2014.

[7] Jeff Howe. Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business. Crown Business, 1st edition, 2009.

[8] James Biagioni Tomas Gerlich Timothy Merrifield Jakob Eriksson. EasyTracker: Automatic Transit Tracking, Mapping, and Arrival Time Prediction Using Smartphones. 2011.

[9] Vaishali Khairnar Pradip Suresh Mane. Analysis of Bus Tracking System Using Gps on Smart Phones. 2014.

[10] Michael M. Mobile Tracking: Apps, GPS, IMEI For Android & iOs Apple. Createspace, 1st edition, 2014.

[11] Track Your Truck. Gps fleet tracking. http://www.trackyourtruck.com, 2016.

[12] Transit Tracking. http://www.transittracking.com/, 2016.

[13] Live London Bus Tracker. https://play.google.com/store/apps/details?id=com.appeffectsuk.bustracker, 2016.

[14] Live Bus Tracker. http://www.livebustracker.com//, 2016.

[15] Vodafone M2M Control. Fleet control. http://www.vodafone.gr/portal/client/cms/viewCmsPage.action?pageId=6751&request_locale=en, 2016.

[16] Wind Fleet Management. https://www.wind.gr/gr/gia-tin-epiheirisi/exupnoi-sunduasmoi/wind-business-cloud-services/upiresia-diaheirisis-stolou-ohimato programmata-hrisis/, 2016.

[17] Moovit Mobile Application. http://moovitapp.com/, 2016.

[18] Link Technologies. Chania city bus. http://link-tech.gr/, 2016.

[19] "What is Agile Software Development?". https://www.agilealliance.org/agile101/what-is-agile/, 2014.

[20] Robert C. Martin. Agile Software Development, Principles, Patterns, and Practices. Pearson, 1st edition, 2002.

[21] Jonathan Rasmusson. The Agile Samurai: How Agile Masters Deliver Great Software. Pragmatic Bookshelf, 1st edition, 2010.

[22] James Shore. The Art of Agile Development. Mary O'Brien, 1st edition, 2007.

[23] Jeff Sutherland. Scrum: The Art of Doing Twice the Work in Half the Time. Crown Business, 1st edition, 2014.

[24] Alistair Cockburn. Crystal Clear: A Human-Powered Methodology for Small Teams: A Human-Powered Methodology for Small Teams. Addison-Wesley Professional, 1st edition, 2004.

[25] James Highsmith III. Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. Addison-Wesley Professional, 1st edition, 2013.

[26] DSDM Consortium. DSDM Atern The Handbook. DSDM Consortium, 1st edition, 2008.

[27] John M. Felsing Stephen R. Palmer. A Practical Guide to Feature-Driven Development. Prentice Hall, 1st edition, 2002.

[28] Agile Team. Agile Manifesto. http://www.agilemanifesto.org/, 2001.

[29] Etienne Savard. Agile Android Software Development. Symbiosoft Publishing, 1st edition, 2014.

[30] Scott Chacon and Ben Straub. Pro Git (2nd ed.). 2014.

[31] Git Branching. "Basic Branching and Merging". https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging, 2014.

[32] Genymobile. Android expertise. http://www.genymobile.com/, 2014.

[33] Google Developers. Google apis for android. https://developers.google.com/android/reference/com/google/android/gms/common/api/GoogleApiClient, 2015.

[34] Juhani Lehtimaki. Smashing Android UI: Responsive User Interfaces and Design Patterns for Android Phones and Tablets. John Wiley & Sons, 1st edition, 2012.

[35] Greg Nudelman. Android Design Patterns: Interaction Design Solutions for Developers. Wiley, 1st edition, 2013.

[36] ΑΣΤΙΚΟ ΚΤΕΛ ΧΑΝΙΩΝ Α.Ε. Δρομολόγια. http://www.chaniabus.gr/dromologia.php, 2015.

[37] Jacob Nielsen. 10 Usability Heuristics for User Interface Design. 1995.

[38] Deborah J. Mayhew. Principles and Guidelines in Software User Interface Design. 1992.

[39] Jon Froehlich Tawanna Dillahunt Predrag Klasnja Jennifer Mankoff Sunny Consolvo Beverly Harrison James A. Landay. UbiGreen: Investigating a Mobile Tool for Tracking and Supporting Green Transportation Habits. 2009.