Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή εργασία

Ανάπτυξη ιστοσελίδας και κινητής εφαρμογής για παρακολούθηση σχολικού λεωφορείου

Γεώργιος Αλεξάκης (ΑΜ: 3052)
Επιβλέπων καθηγητής: Παναγιωτάκης Σπυρίδων

Abstract

This thesis presents a complete platform designed to track a school bus by using smart devices. It is divided into two applications, which are developed in completely different technologies but having in common the web server, that intermediate link which connects them. Reference is also made to all technologies that have been used, which may be quite many, but with proper design and the use of best practices, run successfully.

The bus tracking platform combines driver's application, which is a native application and student's application, which is a hybrid application. The first application, is developed for the school bus driver and the second application is developed for the students who are going to wait for the school bus. As it mentioned earlier, the web server is the intermediate node, which links these two applications and moreover is a very important component of the whole platform.

The first application (driver application) works by sending information about the location of school bus, so in this way students know at any moment where it is and how much time the bus needs to reach them. The intermediate node (the web server) receives and saves school bus driver's application data and also sends data to the student's application. Finally, the second application (student's application) receives data from the web server, and presents them on the map, combining additional information, which is quite helpful for the user.

Σύνοψη

Στην παρούσα πτυχιακή εργασία παρουσιάζεται μια ολοκληρωμένη πλατφόρμα παρακολούθησης σχολικού λεωφορείου με τη χρήση έξυπνων συσκευών. Αποτελείται από δύο εφαρμογές, οι οποίες είναι υλοποιημένες με εντελώς διαφορετικές τεχνολογίες και τον ενδιάμεσο κρίκο, ο οποίος τις ενώνει, τον διαδικτυακό εξυπηρετητή (web server). Γίνεται αναφορά σε όλες τις τεχνολογίες που χρησιμοποιήθηκαν, οι οποίες είναι αρκετές, αλλά με τη σωστή σχεδίαση και τη χρήση καλών πρακτικών υλοποίησης, συνεργάζονται με επιτυχία.

Η πλατφόρμα παρακολούθησης λεωφορείου αποτελείται από δυο εφαρμογές, την εφαρμογή οδηγού, η οποία είναι εγγενής εφαρμογή (native application) και την εφαρμογή μαθητή, η οποία είναι υβριδική εφαρμογή (hybrid application). Η πρώτη εφαρμογή, απευθύνεται στον οδηγό του σχολικού λεωφορείο και η δεύτερη εφαρμογή απευθύνεται στους μαθητές, οι οποίοι θα περιμένουν το σχολικό λεωφορείο. Επίσης ο ενδιάμεσος κόμβος, ο οποίος θα ενώνει τις δυο αυτές εφαρμογές, ο οποίος είναι ένας web server στο web, είναι ένα πολύ βασικό σύστημα στη λειτουργία όλης της πλατφόρμας.

Η λειτουργία της πρώτης εφαρμογής (εφαρμογή οδηγού) είναι η αποστολή πληροφοριών για τη θέση του σχολικού λεωφορείου, έτσι ώστε οι μαθητές να γνωρίζουν ανά πάσα στιγμή που βρίσκεται και πόση ώρα θα κάνει να φτάσει σε αυτούς. Ο ενδιάμεσος κρίκος, ο web server αναλαμβάνει την παραλαβή και αποθήκευση των δεδομένων από τη εφαρμογή οδηγού, όπως επίσης και την αποστολή δεδομένων στην εφαρμογή μαθητή. Η λειτουργία της δεύτερης εφαρμογής (εφαρμογή μαθητή) είναι η παραλαβή των δεδομένων από το web server, και η προβολή τους στο χάρτη, συνδυάζοντας επιπλέον πληροφορίες, οι οποίες είναι αρκετά χρήσιμες για το χρήστη.

Περιεχόμενα

Πίνακας Εικόνων

1 Εισαγωγή

Όσο περνάνε τα χρόνια η τεχνολογία εξελίσσεται με μεγάλη ταχύτητα, νέες τεχνολογίες εμφανίζονται, οι οποίες κάνουν τη ζωή μας πιο εύκολη και απλή. Στις μέρες μας όλοι έχουμε πάνω μας ένα έξυπνο κινητό τηλέφωνο, το οποίο έχει πάψει εδώ και αρκετό καιρό να είναι ένα απλό κινητό τηλέφωνο αλλά είναι ένας ολόκληρος ηλεκτρονικός υπολογιστής τσέπης.

Τα έξυπνα κινητά τηλέφωνα ή smartphone έχουν αρκετές πρόσθετες δυνατότητες υπερτερώντας αρκετά από αυτές ενός σταθερού ηλεκτρονικού υπολογιστή. Διαθέτουν εξελιγμένους αισθητήρες όπως γυροσκόπιο, αισθητήρες οι οποίοι υπολογίζουν την ακριβή θέση (GPS), φωτογραφική μηχανή, θερμόμετρο, αισθητήρες υγρασίας, αισθητήρες ατμοσφαιρικής πίεσης και ασύρματες τεχνολογίες σύνδεσης με δίκτυα (WIFI, 3G Network, 4G Network), όπως επίσης και τεχνολογίες σύνδεσης με άλλες συσκευές (BLUETOOTH, NFC). Όλες αυτές οι δυνατότητες καθιστούν το smartphone ως μια συσκευή με άπειρες εφαρμογές στην καθημερινή ζωή. Οι εφαρμογές ή applications ή apps, όπως είναι αρκετά γνωστά τα τελευταία χρόνια, αξιοποιούν όλες αυτές τις δυνατότητες για να καλύψουν κάποιες ανάγκες των χρηστών τους, όπως επίσης και να διευκολύνουν τη ζωή τους με κάποιες έξυπνες υλοποιήσεις.

Η εργασία αυτή βασίστηκε στην ανάγκη των μαθητών για την ενημέρωση τους για τη θέση του σχολικού τους λεωφορείου. Αξιοποιεί τις δυνατότητες και τις τεχνολογίες ενός smartphone για να διευκολύνει την καθημερινότητα των απλών μαθητών που περιμένουν στη στάση το σχολικό λεωφορείο για να πάνε στο σχολείο τους.

1.1 Περίληψη

Σκοπός της πτυχιακής αυτής, ήταν η μελέτη, σχεδίαση και ανάπτυξη εφαρμογών για κινητά τηλέφωνα, οι οποίες θα βοηθούν τους μαθητές σχολείων να ξέρουν σε πόση ώρα θα φτάσει το λεωφορείο σε αυτούς, όπως επίσης και που ακριβώς βρίσκεται το λεωφορείο. Για την υλοποίηση της εργασίας χρησιμοποιήθηκαν αρκετές γλώσσες προγραμματισμού, οι οποίες συνεργάζονται αρκετά ομαλά, έτσι ώστε να δίνουν στο χρήστη ένα πάρα πού καλό αποτέλεσμα. Οι γλώσσες προγραμματισμού/τεχνολογίες που χρησιμοποιήθηκαν είναι η Java με το Android SDK, HTML5, CSS3, JavaScript, SQL, jQuery, jQuery Mobile, Ajax, όπως επίσης και τα προγράμματα οποία βοήθησαν στην ανάπτυξη της εργασίας είναι το Eclipse, Netbeans, Notepad++, το framework Apache Cordova για την υλοποίηση της υβριδικής εφαρμογής και το Google Maps API.

Με τη χρήση των τεχνολογιών αυτών δημιουργήθηκε ένα πλήρης σύστημα παρακολούθησης του σχολικού λεωφορείου από τους μαθητές, αρκετά εύχρηστο και απλό, δίνοντας μεγάλη έμφαση στην απόδοση. Επίσης η εφαρμογή οδηγού και μαθητή αντίστοιχα είναι σχεδιασμένες με βάση τις ανάγκες και τις δυνατότητες του κάθε χρήστη.

1.2 Κίνητρο για τη διεξαγωγή της εργασίας

Αρκετά συχνά υλοποιούνται εφαρμογές οι οποίες έχουν σαν στόχο να καλύψουν τις ανάγκες, οι οποίες δεν είναι και τόσο σημαντικές για την καθημερινότητα των χρηστών ή ακόμα στόχο έχουν την ψυχαγωγία τους. Η συγκεκριμένη εργασία στοχεύει και επικεντρώνεται στην διευκόλυνση της καθημερινότητας των μικρών μαθητών, ενημερώνοντας τους δυναμικά για την ακριβή θέση και την ώρα άφιξης του σχολικού τους λεωφορείου. Αν σκεφτούμε καλύτερα, θα καταλάβουμε ότι οι μαθητές θα κερδίζουν αρκετό χρόνο κάνοντας άλλες δραστηριότητες αν γνωρίζουν την ακριβή θέση του σχολικού τους λεωφορείου.

Η ανάπτυξη της συγκεκριμένης εργασίας βασίζεται σε μια έννοια, οποία εξελίσσεται και αναπτύσσεται με πολύ γρήγορους ρυθμούς. Το Internet of Things (IoT) είναι μια έννοια στην οποία οι συσκευές που χρησιμοποιούμε καθημερινά, από τις βιομηχανικές συσκευές έως και τα κινητά τηλέφωνα, τους ηλεκτρονικούς υπολογιστές, τις καφετιέρες, τα ψυγεία, τα αυτοκίνητα, τους ανελκυστήρες κτιρίων, τους λαμπτήρες, τα διάφορα wearable gadgets χρησιμοποιούν αισθητήρες για τη συλλογή δεδομένων και κάνουν ενέργειες με τη χρήση των δεδομένων αυτών σε ένα δίκτυο. Κάθε αντικείμενο αναγνωρίζεται μοναδικά μέσω του ενσωματωμένου υπολογιστικού συστήματος του, αλλά είναι σε θέση να λειτουργεί και μόνο του μέσα στην υπάρχουσα δικτυακή υποδομή. Πιο απλά το IoT είναι το μέλλον της τεχνολογίας, το οποίο κάνει τις ζωές μας πιο αποδοτικές.

Ειδικότερα το IoT είναι το δίκτυο από φυσικά αντικείμενα, συσκευές, οχήματα, κτίρια και άλλα αντικείμενα, τα οποία ενσωματώνουν ηλεκτρονικά κυκλώματα, αισθητήρες, λογισμικό και είναι συνδεδεμένα στο δίκτυο ώστε να συλλέγουν και να ανταλλάσσουν δεδομένα. Το IoT δίνει τη δυνατότητα στα αντικείμενα να ελέγχονται εξ αποστάσεως μέσω μιας δικτυακής δομής για την άμεση ενσωμάτωση στον φυσικό κόσμο με συστήματα που βασίζονται σε υπολογιστή. Αν και το Internet είναι βασικό κομμάτι του IoT, ένα αντικείμενο δεν είναι ανάγκη να έχει απευθείας πρόσβαση σε αυτό.

Το IoT είναι μια έννοια που ολοένα εξελίσσεται και χρησιμοποιείται όλο και περισσότερο. Μερικά παραδείγματα του IoT είναι η επικοινωνία μεταξύ των αυτοκινήτων, ώστε να κρατούν αποστάσεις ασφαλείας και να προειδοποιούν τα άλλα για πιθανά ατυχήματα. Αισθητήρες θα μετρούν τα επίπεδα των ρύπων σε μια πόλη όπως επίσης και θα λαμβάνουν μετρήσεις για την υγεία των ανθρώπων με τη χρήση wearable συσκευών. Ένα κτίριο, το οποίο χρησιμοποιεί αισθητήρες για την αυτόματη διακύμανση της θερμοκρασίας και του φωτισμού, ειδικοί αισθητήρες αναλαμβάνουν τη θέρμανση του νερού ώστε να είναι έτοιμο όταν εμείς φτάσουμε στο μπάνιο. Όπως επίσης για το βιομηχανικό εξοπλισμό, ο οποίος μπορεί να προειδοποίει για κάποια βλάβη. Συνδεδεμένα μέσα μαζικής μεταφοράς, ταξί και σχολικά λεωφορεία θα μας επιτρέπουν να γνωρίζουμε τη θέση τους, έτσι ώστε να βρισκόμαστε στη στάση. Οι εφαρμογές του IoT είναι πάρα πολλές και όσο περνάνε τα χρόνια θα γίνουν ακόμα περισσότερες.

1.3 Σκοπός και στόχοι της εργασίας

Ο σκοπός της πτυχιακής εργασίας είναι η ανάπτυξη δυο εφαρμογών για έξυπνες κινητές συσκευές. Η μια εφαρμογή θα φιλοξενείται από την κινητή συσκευή, η οποία

θα έχει ο οδηγός του σχολικού λεωφορείου προκειμένου να στέλνει σε ένα web server τη θέση του λεωφορείου. Η δεύτερη εφαρμογή θα είναι η εφαρμογή, η οποία θα έχουν οι μαθητές, έτσι ώστε να βλέπουν τη θέση του σχολικού τους λεωφορείου μέσα από ένα χάρτη, όπως επίσης και αρκετές πληροφορίες για το λεωφορείο, τον οδηγό, το μονοπάτι του, την ώρα που απομένει μέχρι να φτάσει σε αυτούς και την απόσταση από τον ίδιο το μαθητή. Η εφαρμογή που θα έχουν οι μαθητές είναι μια υβριδική εφαρμογή, δηλαδή μια εφαρμογή που θα μπορεί να τρέχει σε διαφορετικές πλατφόρμες και μια από αυτές θα είναι και ο υπολογιστής όπως επίσης θα υπάρχει και στη μορφή ιστοσελίδας.

Βασικός στόχος είναι οι δύο εφαρμογές, που θα υλοποιηθούν κατά τη διάρκεια της εργασίας, να είναι αρκετά εύχρηστες και απλές, έτσι ώστε και οι μαθητές, όπως επίσης και ο οδηγός να μπορεί με αρκετή ευκολία να χρησιμοποιεί την εκάστοτε εφαρμογή. Τέλος πάρα πολύ σημαντικός στόχος είναι να δημιουργηθούν εφαρμογές, οι οποίες θα έχουν μελετηθεί οι απροσδιόριστες καταστάσεις και θα έχουν επιλυθεί με έξυπνους τρόπους.

1.4 Δομή εργασίας

Η εργασίας χωρίστηκε σε πέντε κεφάλαια, έτσι ώστε να αναλυθούν ξεχωριστά όλες οι τεχνολογίες, οι τεχνικές υλοποίησης, οι αλγόριθμοι και ο κώδικας που χρησιμοποιήθηκε για την υλοποίηση. Το Κεφάλαιο 1 περιέχει την εισαγωγή. Στο Κεφάλαιο 2 γίνεται αναφορά σε σημαντικές θεωρίες, οι οποίες είναι η ραχοκοκαλιά όλης της εργασίας. Στο Κεφάλαιο 3 αναλύονται οι τεχνολογίες, γλώσσες προγραμματισμού, βιβλιοθήκες, προγράμματα που χρησιμοποιήθηκαν, έτσι ώστε να γίνουν πιο κατανοητά πριν ξεκινήσει η ανάλυση. Το Κεφάλαιο 4 χωρίζεται σε τρία μέρη. Στο πρώτο γίνεται εκτενή αναφορά στην εφαρμογή του οδηγού. Στο δεύτερο μέρος γίνεται εκτενή αναφορά στη υλοποίηση του web server και στο τρίτο μέρος γίνεται εκτενή αναφορά στην εφαρμογή μαθητή. Το Κεφάλαιο 5, το οποίο είναι το τελευταίο κεφάλαιο, γίνεται αναφορά στα αποτελέσματα όλης της εργασίας, όπως επίσης και στις μελλοντικές επεκτάσεις της εργασίας. Τέλος υπάρχουν οι βιβλιογραφικές αναφορές και οι πηγές που χρησιμοποιήθηκαν.

2 Τεχνολογίες ανάπτυξης κινητών εφαρμογών

2.1 Native Apps – Web Apps – Hybrid Apps

Κατά το σχεδιασμό και την υλοποίηση εφαρμογών για smartphones, ο κάθε προγραμματιστής έχει να επιλέξει αν η εφαρμογή που θα υλοποιήσει θα είναι εγγενής εφαρμογή (native application) ή διαδικτυακή εφαρμογή (web application) ή υβριδική εφαρμογή (hybrid application). Η επιλογή αυτή πρέπει να γίνει βέβαια με βάση της απαιτήσεις και το σχεδιασμό της εφαρμογής, τις προγραμματιστικές του δυνατότητες, όπως επίσης και να λάβει υπόψιν του και τα αρνητικά και θετικά κάθε υλοποίησης.

2.1.1 Native Applications – Εγγενής Εφαρμογές

Εικόνα 1: Native Applications

Οι εγγενής εφαρμογές (native applications) είναι οι εφαρμογές που απαιτούν ξεχωριστό προγραμματισμό και χρήση διαφορετικών γλωσσών προγραμματισμού για την ανάπτυξη τους ανάλογα με την πλατφόρμα για την οποία αναπτύσσεται η εφαρμογή, όπως Android, Windows, Blackberry, iOS. Εφαρμογές υλοποιημένες για συσκευές με λειτουργικό σύστημα Android δεν μπορούν να τρέξουν σε καμία άλλη πλατφόρμα όπως Windows, Blackberry, IOS, όπως επίσης και το αντίστροφο.

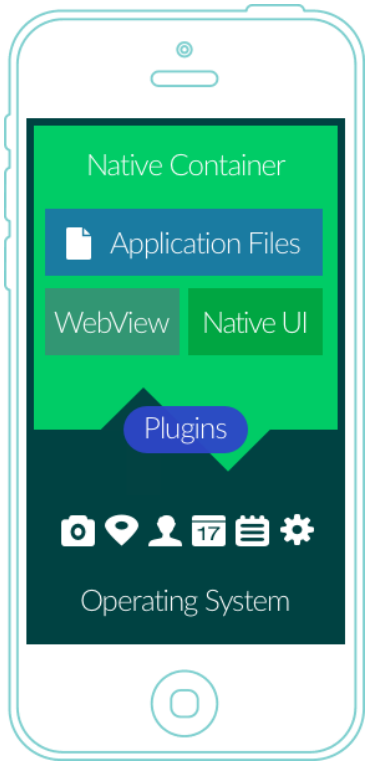
Οι native εφαρμογές είναι οι εφαρμογές οι οποίες αναπτύσσονται, έτσι ώστε να τρέχουν σε συγκεκριμένες συσκευές, για τις οποίες χρησιμοποιούνται συγκεκριμένες τεχνολογίες. Είναι γραμμένες με Objective-C ή Swift για iOS, με Java για Android και με C# για Windows Phone. Οι native εφαρμογές προσφέρουν υψηλή απόδοση και αξιοπιστία και έχουν πρόσβαση στις δυνατότητες της συσκευής, όπως η camera, το internet, το GPS και άλλα. Τα περισσότερα video games για mobile συσκευές είναι native εφαρμογές.

2.1.2 Web Applications – Διαδικτυακές Εφαρμογές

Εικόνα 2: Web Applications

Οι web εφαρμογές ή HTML5 εφαρμογές δεν είναι κανονικές εφαρμογές αλλά στην πραγματικότητα είναι ιστοσελίδες, οι οποίες μοιάζουν και έχουν κάποιες λειτουργίες όπως οι native εφαρμογές. Εκτελούνται με τη χρήση ενός web browser και ο κώδικας τους είναι γραμμένος εξολοκλήρου με τη χρήση web τεχνολογιών όπως HTML5, CSS3 και JavaScript. Οι web εφαρμογές έγιναν πολύ δημοφιλής όταν εμφανίστηκε η HTML5 και είναι κατάλληλες για εργασίες οι οποίες δεν απαιτούν πρόσβαση στις δυνατότητες ενός smartphone. Επίσης μια web εφαρμογή δεν μπορεί να διανεμηθεί μέσω του App Store της κάθε πλατφόρμας.

2.1.3 Hybrid Applications – Υβριδικές Εφαρμογές



Εικόνα 3: Hybrid Applications

Οι υβριδικές εφαρμογές (hybrid applications) είναι εφαρμογές γραμμένες για να τρέχουν σε πολλές διαφορετικές πλατφόρμες, ως μια λύση στο μειονέκτημα των native εφαρμογών. Για την υλοποίηση τους χρησιμοποιούνται οι τεχνολογίες που χρησιμοποιούνται για τις web εφαρμογές, όπως διαδικτυακές σελίδες, διαδικτυακές εφαρμογές και με την προσθήκη κάποιον επεκτάσεων (plugins), γίνονται συμβατές στις διαφορετικές πλατφόρμες. Οι υβριδικές εφαρμογές είναι όπως οι άλλες εφαρμογές που μπορεί ο καθένας να βρει στο κινητό του. Επίσης βρίσκονται και στο App Store της κάθε πλατφόρμας και με αυτές μπορούμε να παίξουμε παιχνίδια, να τραβήξουμε φωτογραφίες και πολλά άλλα.

Όπως και οι ιστοσελίδες, έτσι και οι υβριδικές εφαρμογές είναι υλοποιημένες με ένα συνδυασμό από Web τεχνολογίες όπως η HTML, η CSS, η JavaScript. Η διαφορά που έχουν με τις native εφαρμογές είναι ότι οι υβριδικές εφαρμογές ενθυλακώνονται μέσα σε μια native εφαρμογή, η οποία τρέχει ένα web view. Με αυτό τον τρόπο μια υβριδική εφαρμογή μπορεί να έχει πρόσβαση σε όλες τις δυνατότητες της συσκευής, όπως την κάμερα, το GPS, τις επαφές και άλλα. Επίσης οι υβριδικές εφαρμογές μπορούν να περιλαμβάνουν και native UI χαρακτηριστικά.

Οι περισσότερες υβριδικές εφαρμογές χρησιμοποιούν το API του Apache Cordova, το οποίο παρέχει JavaScript βιβλιοθήκες για την πρόσβαση της υβριδικής εφαρμογής στο native κώδικα με τη χρήση plug-ins.

2.2 Android

Εικόνα 4: Android

Το Android είναι ένα λειτουργικό σύστημα για κινητές συσκευές το οποίο είναι βασισμένο στον πυρήνα του λειτουργικού συστήματος Linux. Σχεδιάστηκε για συσκευές οι οποίες έχουν οθόνη αφής, όπως smartphones, tablets, ρολόγια αλλά χρησιμοποιείται και σε μια πληθώρα από άλλες συσκευές όπως τηλεοράσεις, αυτοκίνητα, κονσόλες παιχνιδιών, ψηφιακές φωτογραφικές μηχανές και άλλα. Το UI του Android βασίζεται κυρίως στη χρήση της αφής για την εκτέλεση ενεργειών όπως το σύρσιμο, το ακούμπημα της οθόνης.

Αρχικά αναπτύχθηκε από την εταιρεία Android, μια εταιρεία που η Google αγόρασε το 2005 και αργότερα από την Open Handset Alliance, μια κοινοπραξία από διάφορες εταιρείες τηλεπικοινωνιακών εταιριών, εταιριών λογισμικού καθώς και κατασκευής hardware, οι οποίες είναι αφιερωμένες στην ανάπτυξη και εξέλιξη ανοιχτών προτύπων στις συσκευές κινητής τηλεφωνίας. Για την ανάπτυξη εφαρμογών για το λειτουργικό σύστημα Android, χρησιμοποιείται η γλώσσα προγραμματισμού Java, με τη χρήση κάποιων βιβλιοθηκών Android για τον έλεγχο της συσκευής.

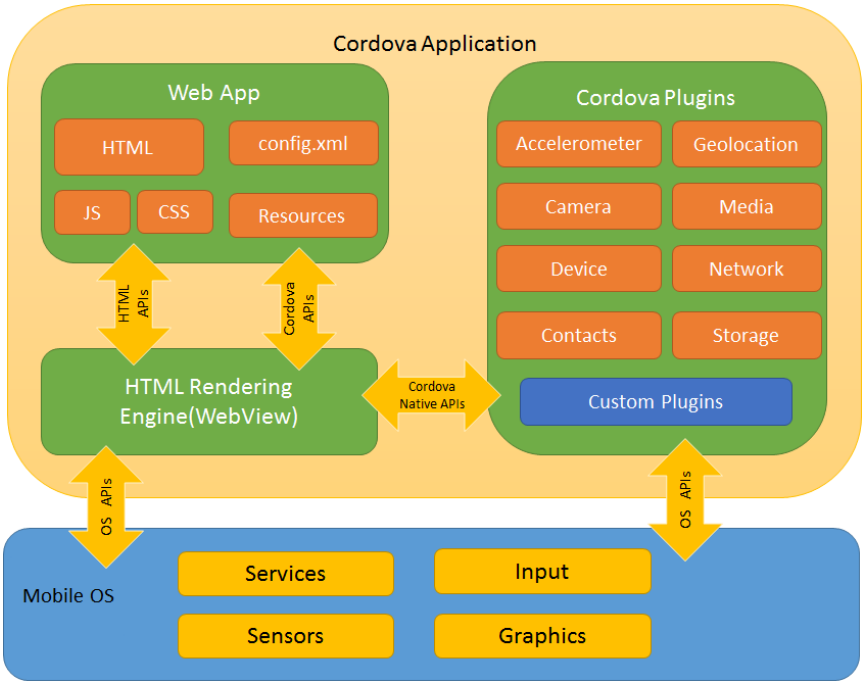
Εικόνα 5: Συσκευές Android

2.3 Apache Cordova

Εικόνα 6: Apache Cordova

Το Apache Cordova είναι ένα δημοφιλές open-source mobile framework για την ανάπτυξη mobile εφαρμογών. Επιτρέπει στους προγραμματιστές να αναπτύξουν εφαρμογές για κινητές συσκευές με τη χρήση των τεχνολογιών web, όπως το HTML5, CSS3 και JavaScript, αποφεύγοντας τη χρήση native γλωσσών ανάπτυξης για κάθε πλατφόρμα όπως Android, iOS, ή Windows.

Οι εφαρμογές γραμμένες με HTML5, CSS3 και JavaScript, εκτελούνται μέσα σε περιτυλίγματα που απευθύνονται σε κάθε πλατφόρμα βασισμένα και συμβατά με το API της κάθε πλατφόρμας, για την πρόσβαση σε αισθητήρες, δεδομένα, και την κατάσταση του δικτύου κάθε συσκευής. Το αποτέλεσμα αυτής της διαδικασίας είναι οι υβριδικές εφαρμογές, το οποίο σημαίνει ότι δεν είναι ούτε πραγματικές native εφαρμογές, γιατί όλα είναι φτιαγμένα με τις web τεχνολογίες, ούτε web εφαρμογές γιατί έχουν πρόσβαση σε όλες τις δυνατότητες μιας κινητής συσκευής, όπως η κάμερα, το GPS και άλλα.



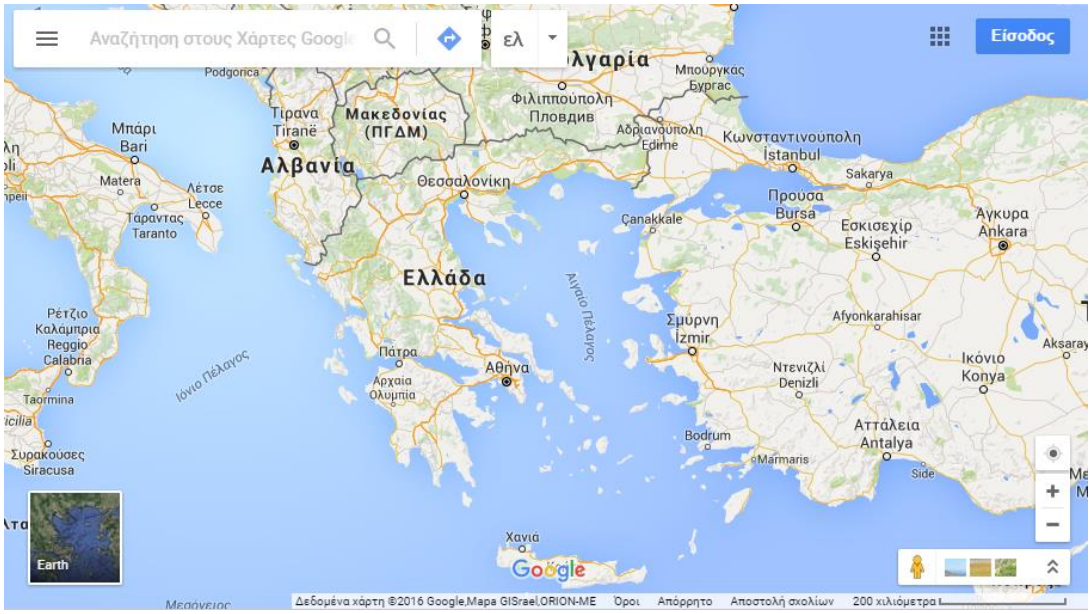
Εικόνα 7: Αρχιτεκτονική Apache Cordova

2.4 Google Maps

Εικόνα 8: Google Maps

Το Google Maps είναι μια υπηρεσία χαρτογράφησης, η οποία αναπτύχθηκε από τους Lars και Jens Eilstrup Rasmussen. Προσφέρει στους χρήστες της χάρτες δρόμων και σχεδιαστή διαδρομών για μεταφορές με τα πόδια, αυτοκίνητο, ποδήλατο ή μέσα μαζικής μεταφοράς και εντοπισμό των επιχειρήσεων, πανοραμικές εικόνες δρόμων, την κυκλοφορία στους δρόμους σε πραγματικό χρόνο.

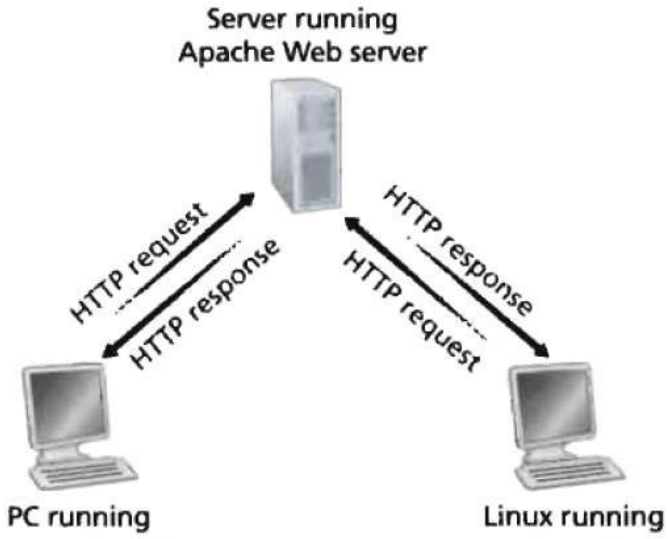
Το Google Maps αρχικά σχεδιάστηκε ως εφαρμογή για σταθερό υπολογιστή γραμμένη σε C++, αλλά στη συνέχεια μετατράπηκε σε εφαρμογή για το διαδίκτυο. Η Google προσφέρει στους προγραμματιστές το Google Maps API, το οποίο τους δίνει τη δυνατότητα να ενσωματώνουν χάρτες Google Maps στις δικές τους ιστοσελίδες και εφαρμογές. Για το σχεδιασμό ενός χάρτη σε μια web εφαρμογή ή μια ιστοσελίδα χρησιμοποιούνται η JavaScript, Ajax και η XML.



Εικόνα 9: Παράδειγμα χάρτη Google Maps

3 Τεχνολογίες και γλώσσες προγραμματισμού διαδικτύου

3.1 Πρωτόκολλο Μεταφοράς Υπερκειμένου HTTP



Εικόνα 10: HTTP

Το πρωτόκολλο μεταφοράς υπερκειμένου (HyperText Transfer Protocol, HTTP), το πρωτόκολλο επιπέδου εφαρμογής του web, είναι ο πυρήνας του web. Το HTTP υλοποιείται σε δύο προγράμματα: ένα πρόγραμμα πελάτη και ένα πρόγραμμα εξυπηρετητή, που εκτελούνται σε διαφορετικά τερματικά συστήματα, συνομιλούν μεταξύ τους ανταλλάσσοντας μηνύματα HTTP. Το HTTP ορίζει την δομή αυτών των μνημάτων και το πώς ο πελάτης και ο εξυπηρετητής ανταλλάσσουν τα μηνύματα.

Το HTTP ορίζει πώς οι πελάτες στο web ζητούν ιστοσελίδες από το web και πώς οι εξυπηρετητές μεταφέρουν ιστοσελίδες σε πελάτες. Όταν ο χρήστης ζητά μία ιστοσελίδα, το πρόγραμμα περιήγησης στέλνει μηνύματα αίτησης HTTP για τα αντικείμενα της σελίδας, στον εξυπηρετητή. Ο εξυπηρετητής δέχεται τις αιτήσεις και αποκρίνεται με μηνύματα απόκρισης HTTP, τα οποία περιέχουν αντικείμενα.

3.2 HTML

Εικόνα 11: HTML

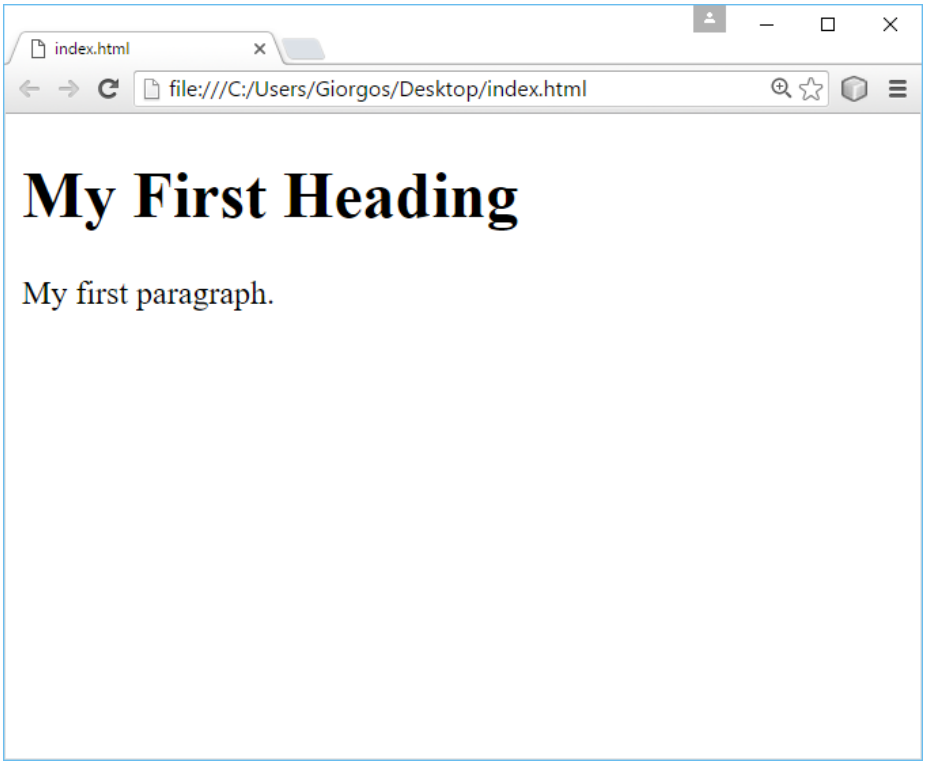
Η HTML (HyperText Markup Language) είναι η βασική γλώσσα σήμανσης για την κατασκευή και την παρουσίαση περιεχομένου στον παγκόσμιο ιστό WWW (World Wide Web). Η γλώσσα HTML είναι ένα δημιούργημα του φυσικού Tim Berners-Lee, ο οποίος ενώ εργαζόταν στο CERN επινόησε την HTML και έγραψε το λογισμικό για τον πρώτο web browser, όπως επίσης και το λογισμικό του web server. Η τελευταία έκδοση της HTML είναι η έκδοση HTML5, η οποία παρουσιάστηκε τον Οκτώβριο του 2014 από την W3C για την ανάπτυξη της γλώσσας.

Ο πηγαίος κώδικας ενός εγγράφου HTML αποτελείται από μια σειρά από ειδικά στοιχεία, τις ετικέτες, τα tags. Μια ετικέτα ξεκινάει με το γνωστό μαθηματικό σύμβολο μικρότερο από < και τελειώνει με το μαθηματικό σύμβολο μεγαλύτερο από >. Οι περισσότερες ετικέτες στην HTML λειτουργούν ανά ζεύγη, για παράδειγμα … . Η πρώτη ετικέτα επισημαίνει την αρχή και η δεύτερη ετικέτα επισημαίνει το τέλος των στοιχείων, των οποίων πρόκειται να επηρεαστούν από τις συγκεκριμένες ετικέτες. Ανάμεσα στις δύο ετικέτες μπορούν να τοποθετηθούν διάφορα δομικά στοιχεία, όπως εικόνες κείμενο, πίνακες και άλλα.

Για τη συγγραφή του πηγαίου κώδικα ενός εγγράφου HTML μπορούμε να χρησιμοποιήσουμε ένα απλό επεξεργαστή κειμένου. Για την προβολή ενός εγγράφου HTML μπορούμε να το προβάλουμε στην οθόνη με ένα περιηγητή ιστού (web browser). Ο σκοπός του είναι να ερμηνεύει τις ετικέτες ενός εγγράφου HTML και να συνθέτει σε σελίδες αυτό που διαβάζει.

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

Εικόνα 12: HTML παράδειγμα κώδικα



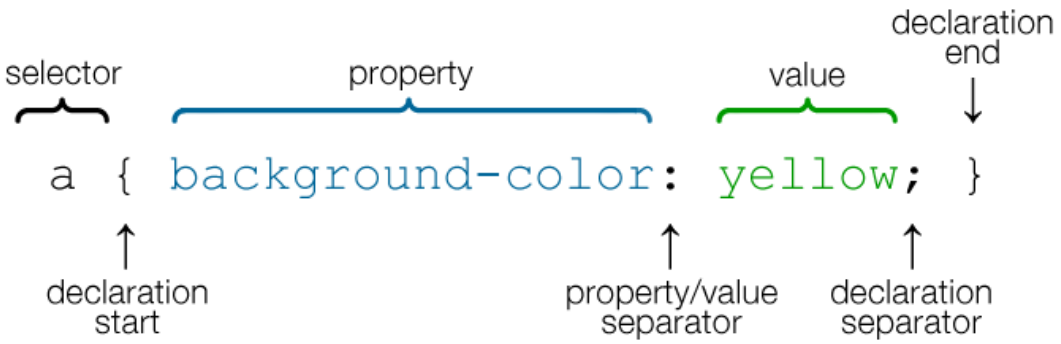
Εικόνα 13: HTML εκτέλεση κώδικα

3.3 CSS

Εικόνα 14: CSS

Η CSS είναι μια γλώσσα υπολογιστή, η οποία είναι υπεύθυνη για την εμφάνιση των ιστοσελίδων και χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης όπως η HTML. Ένα φύλλο στυλ (style sheet) γραμμένο σε CSS περιέχει κανόνες, οι οποίοι καθορίζουν τον τρόπο εφαρμογής κάποιου συγκεκριμένου τρόπου εμφάνισης σε ένα μέρος μιας ιστοσελίδας ή σε ολόκληρο το περιεχόμενο της.

Οι κανόνες στην CSS αποτελούνται από επιλογείς (selectors) και από δηλώσεις (declarations). Οι επιλογείς χρησιμοποιούνται για να δηλώσουν ποιο μέρος της ιστοσελίδας θα επηρεαστεί από τη συγκεκριμένη ιδιότητα. Οι δηλώσεις ορίζουν την ιδιότητα ενός στοιχείου στο οποίο θέλουμε να ορίσουμε κάποιο στυλ αλλά και την τιμή την οποία θα αποκτήσει το στοιχείο. Οι δηλώσεις των κανόνων περικλείονται ανάμεσα σε αγκύλες {}.



Εικόνα 15: CSS δομή

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
    color: orange;
    text-align: center;
}
p {
    font-family: "Times New Roman";
    font-size: 20px;
}
</style>
</head>
<body>

<h1>My First CSS Example</h1>
<p>This is a paragraph.</p>

</body>
</html>

```

Εικόνα 16: CSS παράδειγμα κώδικα

Εικόνα 17: CSS εκτέλεση κώδικα

3.4 JavaScript

Εικόνα 18: JavaScript

Η JavaScript είναι μια γλώσσα προγραμματισμού, που βασίζεται στα πρωτότυπα, είναι δυναμική, με ασθενείς τύπους, έχει συναρτήσεις ως αντικείμενα και αποτελεί έναν εύκολο τρόπο για το χειρισμό του κώδικα της HTML. Επίσημα η JavaScript δημιουργήθηκε το 1995 από τον Brendan Eich, ενώ δούλευε στην εταιρεία Netscape με το όνομα Mocha. Αργότερα μετονομάστηκε LiveScript και τελικά σε JavaScript. Αρχικά αποτέλεσε μέρος της υλοποίησης των web browsers, ώστε τα σενάρια από την πλευρά του πελάτη (client-side scripts) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται.

Η JavaScript ενσωματώνεται μέσα στον πηγαίο κώδικα HTML με αρμονικό τρόπο, έτσι ώστε να θεωρείται αναπόσπαστο τμήμα της HTML και δεν απαιτεί μεταγλωττιστή για να εκτελεστεί, αλλά ένα απλό web browser, δηλαδή ένα απλό ένα διερμηνευτή. Επίσης η JavaScript χρησιμοποιείται σε εφαρμογές που δεν βασίζονται στο web, όπως τα αρχεία PDF, τα desktop widgets και σε άλλα.

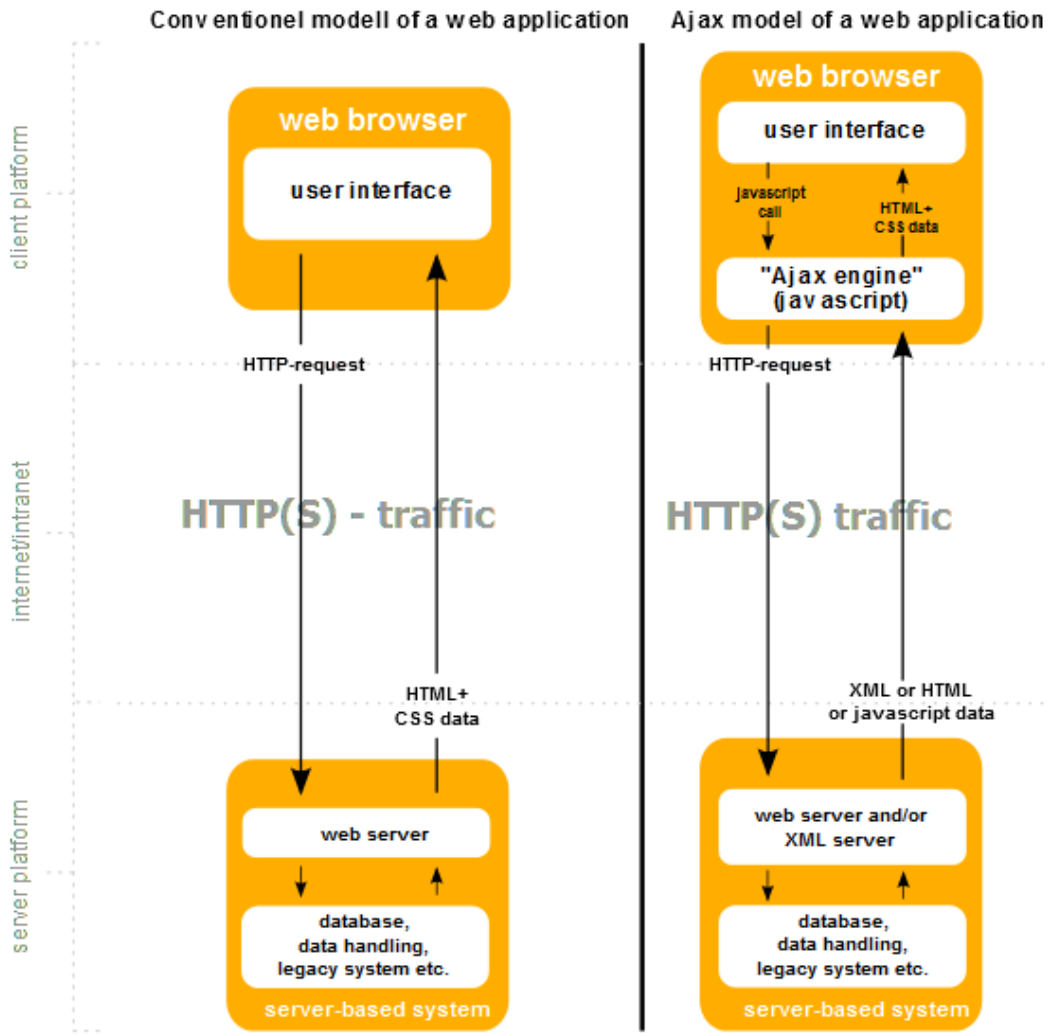
Ο πηγαίος κώδικας JavaScript μιας σελίδας περικλείεται από τις ετικέτες της HTML <script> και </script>. Η σύνταξη της JavaScript είναι επηρεασμένη από τη γλώσσα προγραμματισμού C και οι βασικές αρχές σχεδιασμού της προέρχονται από τις γλώσσες προγραμματισμού Self και Scheme.

```
<!DOCTYPE html>
<html>
<body>

<h1>JavaScript Statements</h1>
<p>Statements are separated by semicolons.</p>
<p>The variables x, y, and z are assigned
the values 5, 6, and 11:</p>
<p id="demo"></p>

<script>
var x = 5;
var y = 6;
var z = x + y;
document.getElementById("demo").innerHTML = z;
</script>

</body>
</html>
```

Εικόνα 19: JavaScript παράδειγμα κώδικα

Εικόνα 20: JavaScript εκτέλεση κώδικα

3.5 Ajax



Εικόνα 21: Ajax

Η Ajax (Asynchronous JavaScript and XML) είναι μια τεχνική, η οποία χρησιμοποιείται από τις web εφαρμογές και τις ιστοσελίδες, επιτρέποντας τους την ανανέωση ενός μέρους τους, χωρίς να χρειάζεται να γίνει ανανέωση σε ολόκληρη την εφαρμογή ή την ιστοσελίδα. Ειδικότερα με τη χρήση της Ajax, οι εφαρμογές μπορούν να στέλνουν και να λαμβάνουν δεδομένα από ένα διαδικτυακό εξυπηρετητή ασύγχρονα, δηλαδή χωρίς την ανανέωση ολόκληρης της σελίδας.

Η Ajax χρησιμοποιεί την γλώσσα προγραμματισμού JavaScript και παρά τη χρήση του XML στο όνομα της, μπορεί να χρησιμοποιηθεί και η μορφοποίηση JSON. Η τεχνική Ajax προτάθηκε το 2005 από τον Jesse James Garrett για την ανάπτυξη βέλτιστων εφαρμογών στο web. Συνδυάζει τις υπάρχουσες τεχνολογίες διαδικτυακού προγραμματισμού με στόχο τη δημιουργία διαδικτυακών εφαρμογών, οι οποίες είναι παρόμοιες με τις κλασσικές εφαρμογές.

3.6 jQuery

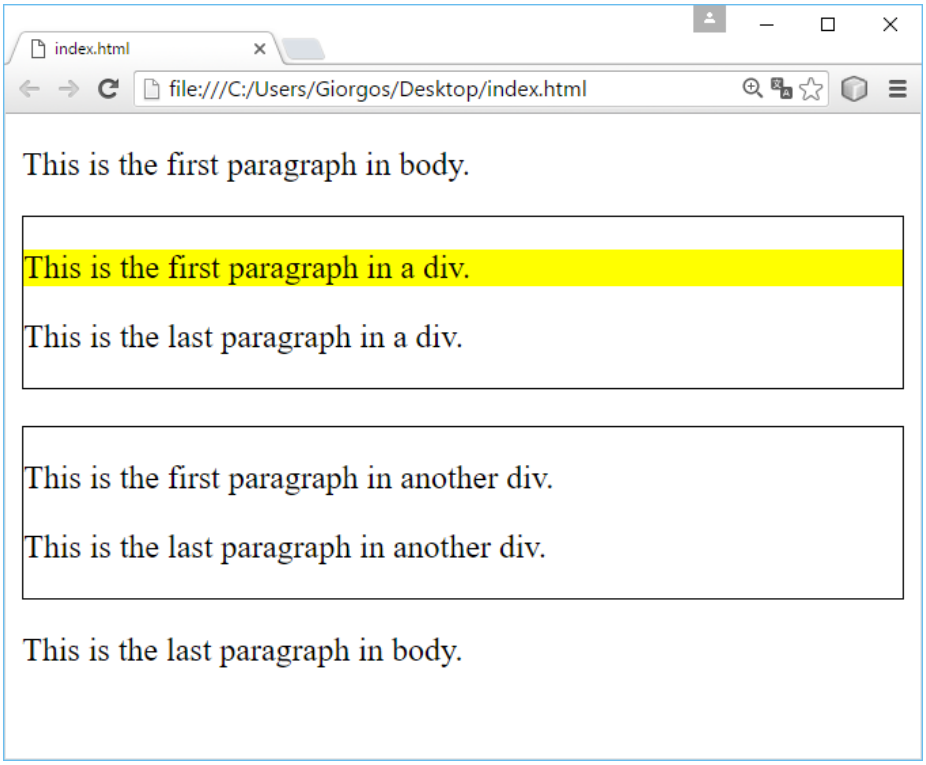
Εικόνα 22: jQuery

Η jQuery είναι μια βιβλιοθήκη της γλώσσας JavaScript και είναι σχεδιασμένη για την απλοποίηση της χρήσης ορισμένων εντολών, οι οποίες είναι υπεύθυνες για το χειρισμό ενός HTML έγγραφου από τη μεριά του πελάτη. Η jQuery δημιουργήθηκε από τον John Resig το 2006 και επηρεάστηκε από τη βιβλιοθήκη cssQuery. Υποστηρίζεται από τους περισσότερους web browsers και είναι αρκετά γρήγορη και απλή στη σύνταξη της.

Η σύνταξη της jQuery είναι σχεδιασμένη, έτσι ώστε να απλοποιεί τη δημιουργία animations, την επιλογή στοιχείων DOM, τον χειρισμό των event και την ανάπτυξη εφαρμογών οι οποίες χρησιμοποιούν AJAX τεχνολογία. Για να γίνει χρήση της βιβλιοθήκης θα πρέπει να δηλωθεί ανάμεσα στην HTML, ότι θα γίνει η χρήση της είτε παρέχοντας το αρχείο τοπικά είτε παρέχοντας ένα σύνδεσμο σε ένα από τους διακομιστές που τη φιλοξενούν.

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("div p").first().css("background-color", "yellow");
});
</script>
</head>
<body>
<p>This is the first paragraph in body.</p>
<div style="border: 1px solid black;">
  <p>This is the first paragraph in a div.</p>
  <p>This is the last paragraph in a div.</p>
</div><br>
<div style="border: 1px solid black;">
  <p>This is the first paragraph in another div.</p>
  <p>This is the last paragraph in another div.</p>
</div>
<p>This is the last paragraph in body.</p>
</body>
</html>

```

Εικόνα 23: jQuery παράδειγμα κώδικα



Εικόνα 24: jQuery εκτέλεση κώδικα

3.7 jQuery Mobile

Εικόνα 25: jQuery Mobile

Η jQuery Mobile είναι μια βιβλιοθήκη της γλώσσας JavaScript, η οποία αναπτύχθηκε και συντηρείται από την ομάδα της jQuery. Η δημιουργία της επικεντρώνεται στην δημιουργία εφαρμογών, οι οποίες είναι συμβατές με τα περισσότερα smartphones και tablets. Χρησιμοποιεί της γλώσσες HTML5, CSS3, JavaScript, Ajax και είναι σχεδιασμένη, έτσι ώστε να δημιουργεί responsive ιστοσελίδες και εφαρμογές, οι οποίες είναι συμβατές με όλα τα smartphones, tablets και σταθερούς υπολογιστές.

Για τη χρήση της βιβλιοθήκης θα πρέπει να δηλωθεί ανάμεσα στην HTML, ότι θα γίνει η χρήση της είτε παρέχοντας το αρχείο τοπικά είτε παρέχοντας ένα σύνδεσμο σε ένα από τους διακομιστές που τη φιλοξενούν, όπως και στην jQuery. Βασικό χαρακτηριστικό της jQuery είναι ότι ένα HTML5 έγγραφο μπορεί να περιέχει περισσότερες από μια σελίδες. Με αυτό τον τρόπο για την προβολή όλων των σελίδων

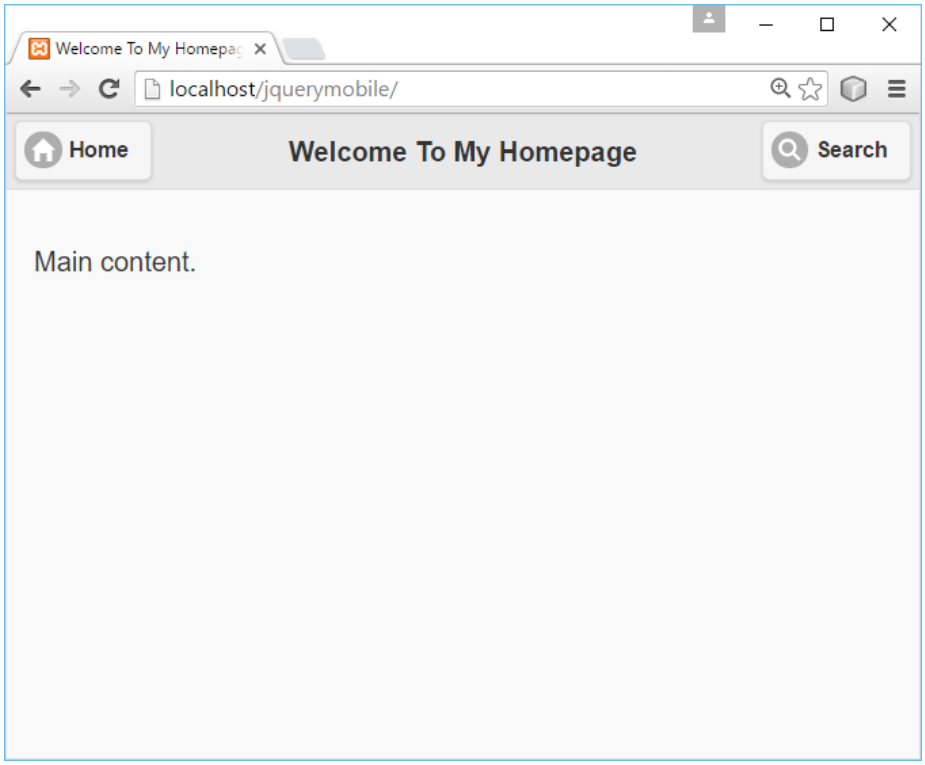
είναι αναγκαία η φόρτωση ενός μόνο εγγράφου. Η μια σελίδα συνδέεται με κάποια άλλη σελίδα με τη δήλωση του id της σελίδας με το attribute href="#id".

Βασικά attributes

- **data-role:** Ορίζει το ρόλο του στοιχείου, όπως επικεφαλίδα, περιεχόμενο, υποσέλιδο και άλλα.
- **data-theme:** Ορίζει ποιο θέμα θα χρησιμοποιηθεί για το σχεδιασμό.
- **data-position:** Ορίζει αν η θέση του στοιχείου πρέπει να είναι fixed ή inline.
- **data-transition:** Ορίζει ένα από το 10 προκατασκευασμένα animations για τη χρήση κατά το φόρτωμα σελίδων.
- **data-icon:** Ορίζει ένα από τα 50 προκατασκευασμένα εικονίδια.

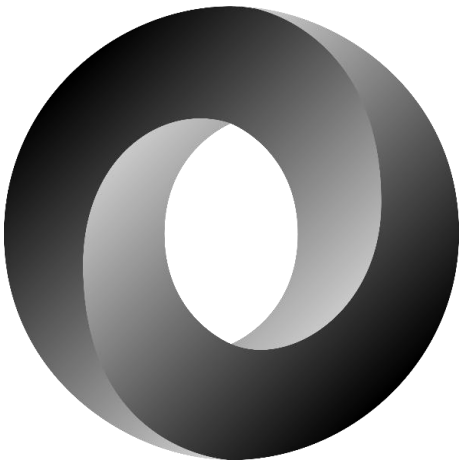
```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.css">
<script src="http://code.jquery.com/jquery-1.11.3.min.js"></script>
<script src="http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.js"></script>
</head>
<body>
<div data-role="page">
  <div data-role="header">
    <a href="#" class="ui-btn ui-corner-all ui-shadow ui-icon-home ui-btn-icon-left">Home</a>
    <h1>Welcome To My Homepage</h1>
    <a href="#" class="ui-btn ui-corner-all ui-shadow ui-icon-search ui-btn-icon-left">Search</a>
  </div>
   <div data-role="main" class="ui-content">
    <p>Main content.</p>
  </div>
</div>
</body>
</html>
```

Εικόνα 26: jQuery Mobile παράδειγμα κώδικα



Εικόνα 27: jQuery Mobile εκτέλεση κώδικα

3.8 JSON

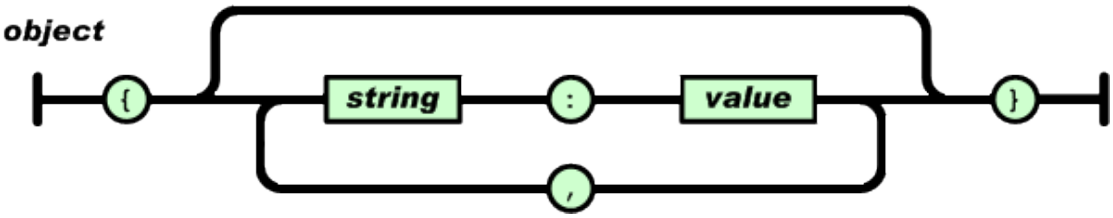


Εικόνα 28: JSON

Το JSON (JavaScript Object Notation) είναι ένα ανοικτό πρότυπο για την περιγραφή ιεραρχικών δεδομένων με τη μορφή κειμένου. Το 2006 ο Douglas Crolford δημιούργησε το πρότυπο JSON, το οποίο είναι αρκετά εύκολο να διαβαστεί και να γραφτεί από ανθρώπους και θεωρείται ως το πιο ελαφρύ σχήμα ανταλλαγής δεδομένων. Επίσης είναι εντελώς ανεξάρτητο από γλώσσες προγραμματισμού αν και μοιάζει συντακτικά με τη γλώσσα προγραμματισμού C.

Τα δεδομένα με μορφοποίηση JSON μπορούν πολύ εύκολα να αναγνωρισθούν από τη JavaScript, καθιστώντας το ιδανικό για ανταλλαγή δεδομένων με τη χρήση της τεχνολογίας Ajax. Ένα αντικείμενο JSON είναι ένα σύνολο από ζευγάρια

ονομάτων/τιμών και κάθε όνομα ακολουθείται από άνω-κάτω τελεία «:» και τα ζευγάρια ονόματος/τιμής χωρίζονται από κόμμα «,». Η δήλωση ενός αντικειμένου JSON ξεκινάει με { (αριστερό άγκιστρο) και τελειώνει με } (δεξιό άγκιστρο).



Εικόνα 29: JSON δομή

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

Εικόνα 30: JSON παράδειγμα κώδικα

3.9 PHP

Εικόνα 31: PHP

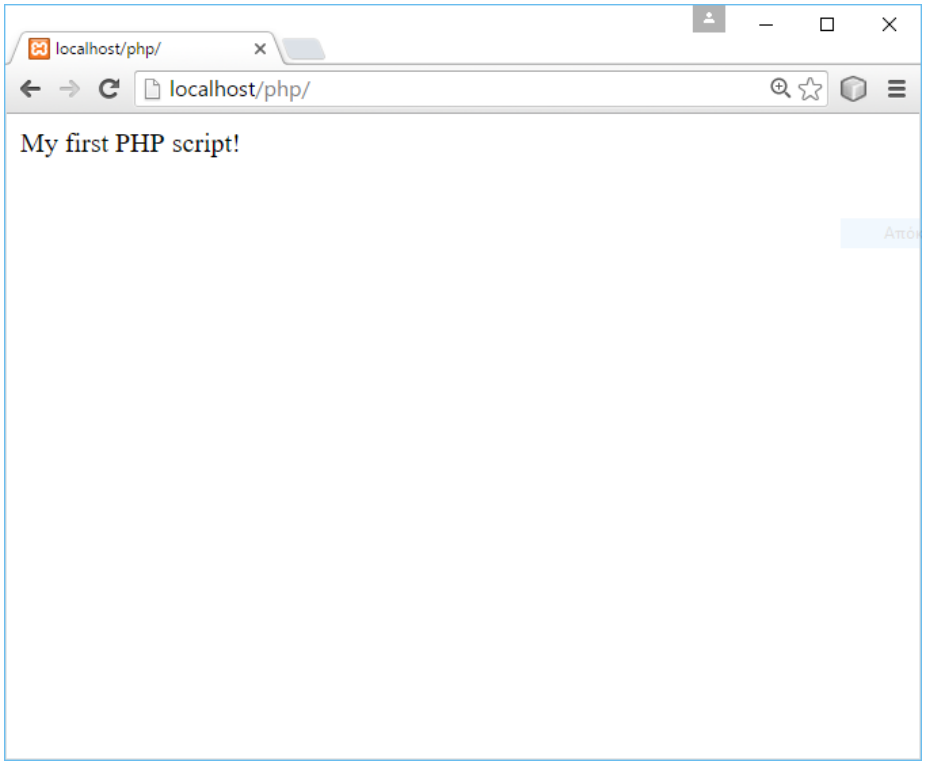
Η PHP είναι μια γλώσσα προγραμματισμού, η οποία χρησιμοποιείται για τη δημιουργία δυναμικών ιστοσελίδων, όπως επίσης είναι και μια γλώσσα γενικού σκοπού. Μια σελίδα γραμμένη σε PHP τρέχει στο web server, έτσι ώστε να παραχθεί το τελικό περιεχόμενο το οποίο θα αποσταλεί στον web browser των επισκεπτών σε μορφή κώδικα HTML. Ο κώδικας της PHP ενσωματώνεται μέσα στον HTML κώδικα ή μπορεί να συνδυαστεί με διάφορα Web Frameworks, CMS ή Web template systems.

Ο Rasmus Lerdorf δημιούργησε την PHP το 1994 χρησιμοποιώντας αρχικά ως ένα απλό script με όνομα php.cgi με τη χρήση της γλώσσας προγραμματισμού C. Ο σκοπός για τη δημιουργία του ήταν η διατήρηση στατιστικών για τα άτομα που έβλεπαν διαδικτυακά το βιογραφικό του. Αργότερα η γλώσσα ονομάζεται PHP/FI (Personal Home Page/Form Interpreter) και το 1997 έφθασε στην έκδοση 2.0. Ο κώδικας της PHP ενσωματώνεται στον κώδικα της HTML και περικλείεται ανάμενα

στα σύμβολα αρχής <?php και τα σύμβολα ?> για το κλείσιμο ενός κομματιού PHP κώδικα. Επίσης δεν είναι απαραίτητο ο κώδικας PHP να είναι ενσωματωμένος στον HTML κώδικα, αλλά μπορεί να γραφεί και ξεχωριστά.

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5  <?php
6  echo "My first PHP script!";
7  ?>
8
9  </body>
10 </html>
```

Εικόνα 32: PHP παράδειγμα κώδικα



Εικόνα 33: PHP εκτέλεση κώδικα

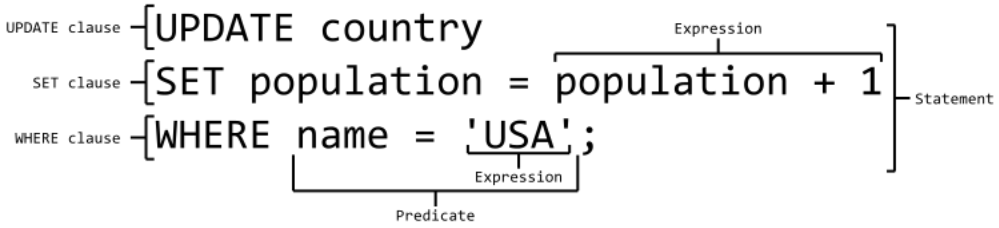
3.10 SQL

Εικόνα 34: SQL

Η SQL (Structured Query Language) είναι μια γλώσσα ερωτημάτων, η οποία σχεδιάστηκε για τη διαχείριση δεδομένων σε ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (Relational Database Management System, RDBMS). Η SQL χρησιμοποιεί ένα συνδυασμό από σχεσιακή άλγεβρα και σχεσιακό λογισμό. Η γλώσσα περιλαμβάνει δυνατότητες ανάκτησης και ενημέρωσης δεδομένων, δημιουργίας και τροποποίησης σχημάτων και σχεσιακών πινάκων, αλλά και ελέγχου πρόσβασης στα δεδομένα.

Η SQL αναπτύχθηκε στην IBM στις αρχές της δεκαετίας του 1970 και αρχικά ονομαζόταν SEQUEL ως μέρος του έργου System R. και στη συνέχεια μετονομάστηκε σε SQL. Η γλώσσα SQL περιλαμβάνει διάφορα γλωσσικά στοιχεία.

- **clauses:** Είναι τα συστατικά των δηλώσεων και ερωτήσεων και σε μερικές περιπτώσεις είναι προαιρετικά.
- **expressions:** Μπορούν να παραγάγουν είτε τις κλιμακωτές τιμές είτε πίνακες, οι οποίοι αποτελούνται από στήλες και σειρές δεδομένων.
- **predicates:** Διευκρινίζουν τους όρους που μπορούν να αξιολογηθούν σαν σωστό, λάθος ή άγνωστο.
- **queries:** Ανακτούν τα στοιχεία βασισμένα σε ειδικά κριτήρια.
- **statements:** Μπορούν να έχουν επίδραση στα σχήματα και τα δεδομένα, ή μπορούν να ελέγξουν τις συνδέσεις, τη ροή του προγράμματος και άλλα.



Εικόνα 35: SQL δομή

```
1  INSERT INTO Customers (CustomerName, Country)
2  SELECT SupplierName, Country FROM Suppliers;
```

Εικόνα 36: SQL παράδειγμα κώδικα

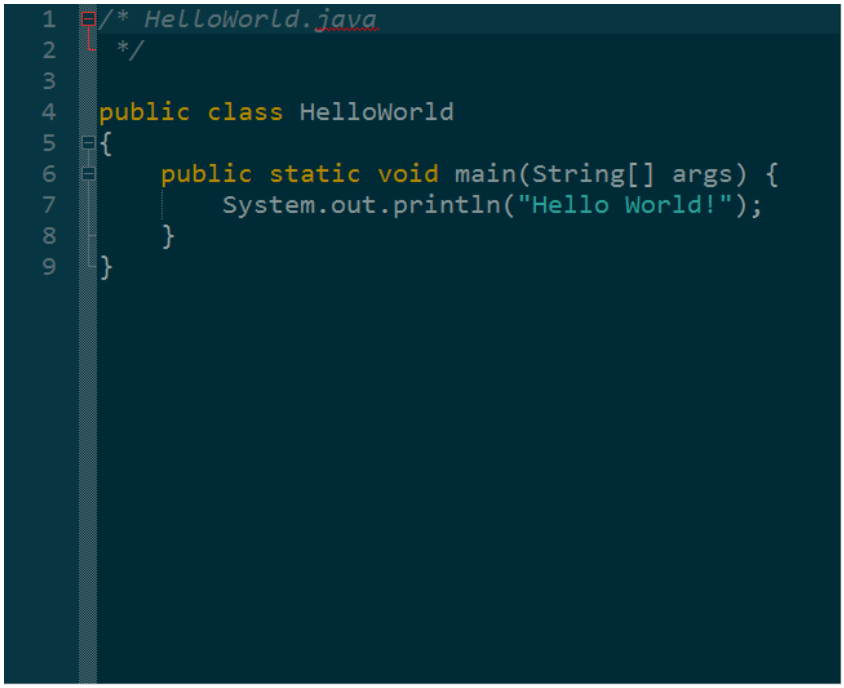
3.11 JAVA

Εικόνα 37: Java

Η Java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού, η οποία είναι σχεδιασμένη, έτσι ώστε να είναι ανεξάρτητη από την πλατφόρμα που εκτελείται. Αρχικά χρησιμοποιήθηκε για δημιουργία απλών προγραμμάτων σε ιστοσελίδες, αλλά πλέον χρησιμοποιείται και για εφαρμογές πέρα από το Web.

Η ιστορία της Java ξεκινάει στις αρχές του 1991, όταν James Gosling, ο οποίος δούλευε στην εταιρεία Sun Microsystems, αναζητούσε ένα εργαλεία για τη δημιουργία λογισμικού για μικροσυσκευές, επειδή απογοητεύτηκε από τη γλώσσα που χρησιμοποιούσαν. Τα εργαλεία της εποχής ήταν η C++ μια αντικειμενοστραφής γλώσσα προγραμματισμού αναπτυγμένη μια δεκαετία πριν ως επέκταση της C. Για να αντιμετωπίσει μερικά πράγματα που τον απογοήτευσαν στη C++ δημιούργησε μια νέα γλώσσα προγραμματισμού, τη Java, η οποία κυκλοφόρησε το 1995.

Βασικό χαρακτηριστικό της Java είναι η δυνατότητα ενός προγράμματος γραμμένο με τη Java, να εκτελείται σε διαφορετικά περιβάλλοντα χωρίς τροποποιήσεις. Ένα πρόγραμμα Java μετατρέπεται σε μορφή bytecode και μπορεί να εκτελεστεί σε οποιαδήποτε υπολογιστή, κινητό τηλέφωνο ή οποιαδήποτε συσκευή, η οποία είναι εξοπλισμένη σε μια εικονική μηχανή Java. Ένα πρόγραμμα γραμμένο με τη γλώσσα Java μπορεί να εκτελείται σε ένα υπολογιστή με λειτουργικό σύστημα Windows, Unix, Linux, BSD, MacOS, όπως επίσης και σε οποιοδήποτε σύστημα αρκεί να είναι εξοπλισμένο με μια εικονική μηχανή Java.



```java
/* HelloWorld.java
 */

public class HelloWorld
{
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Εικόνα 38: Java παράδειγμα κώδικα

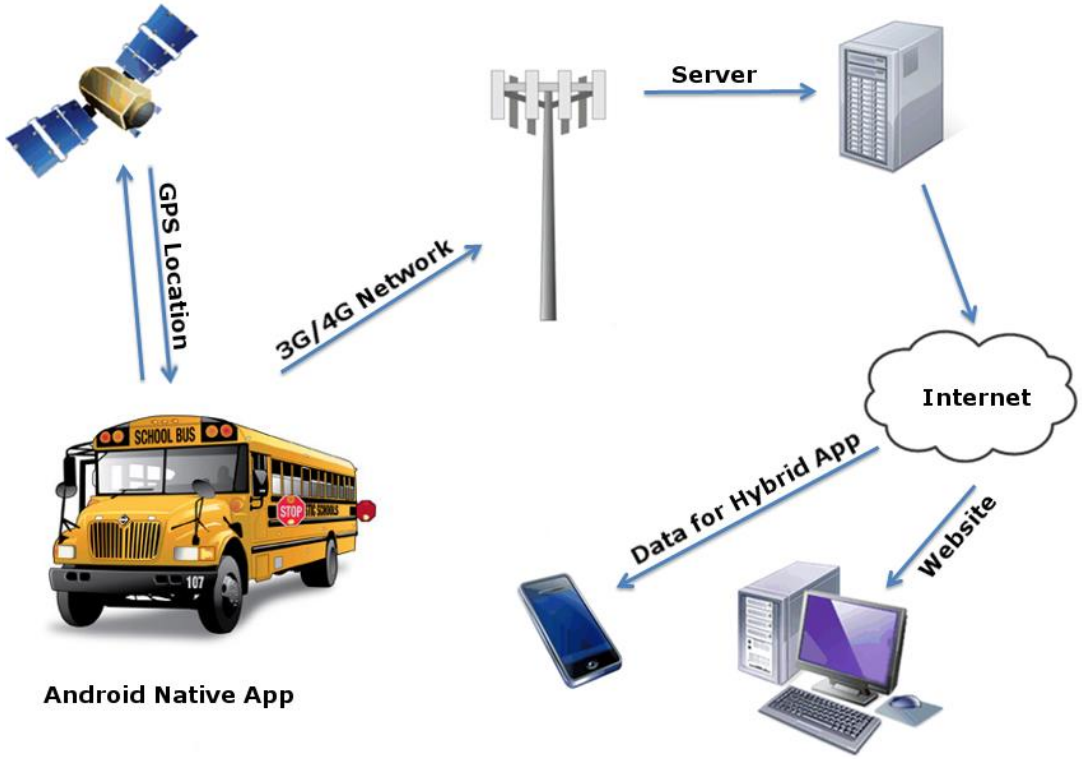
```
run:
Hello World!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Εικόνα 39: Java εκτέλεση κώδικα

4 Υλοποίηση

Σε αυτό το κεφάλαιο θα γίνει αναφορά στην υλοποίηση της πλατφόρμας παρακολούθησης σχολικού λεωφορείου. Ο σχεδιασμός της υλοποίησης χωρίζεται σε τρία κομμάτια. Το πρώτο περιλαμβάνει την εφαρμογή οδηγού, το δεύτερο περιλαμβάνει το διαδικτυακό εξυπηρετητή και το τρίτο την εφαρμογή μαθητή. Αρχικά γίνεται ο σχεδιασμός σύμφωνα με τις απαιτήσεις και στη συνέχεια γίνεται η συγγραφή του κώδικα για την τελική υλοποίηση.

Η σχεδίαση της κάθε εφαρμογής έγινε με διαφορετικό τρόπο και χρησιμοποιήθηκαν εντελώς διαφορετικές τεχνολογίες για την ανάπτυξη τους. Για την επικοινωνία των δύο εφαρμογών χρειάστηκε να μεσολαβήσει ένας διαδικτυακός εξυπηρετητής (web server). Αναφορά γίνεται στο user interface της κάθε εφαρμογής και στον κώδικα για τη δημιουργία τους. Για τη δοκιμή και παρουσίαση της πλατφόρμας δημιουργήθηκε, μια εφαρμογή προσομοίωσης διαδρομής, η οποία στέλνει δοκιμαστικές διαδρομές κάνοντας προσομοίωση μιας διαδρομής.



Εικόνα 40: Σχεδιάγραμμα υλοποίησης

4.1 Εφαρμογή οδηγού

4.1.1 Εισαγωγή εφαρμογής οδηγού

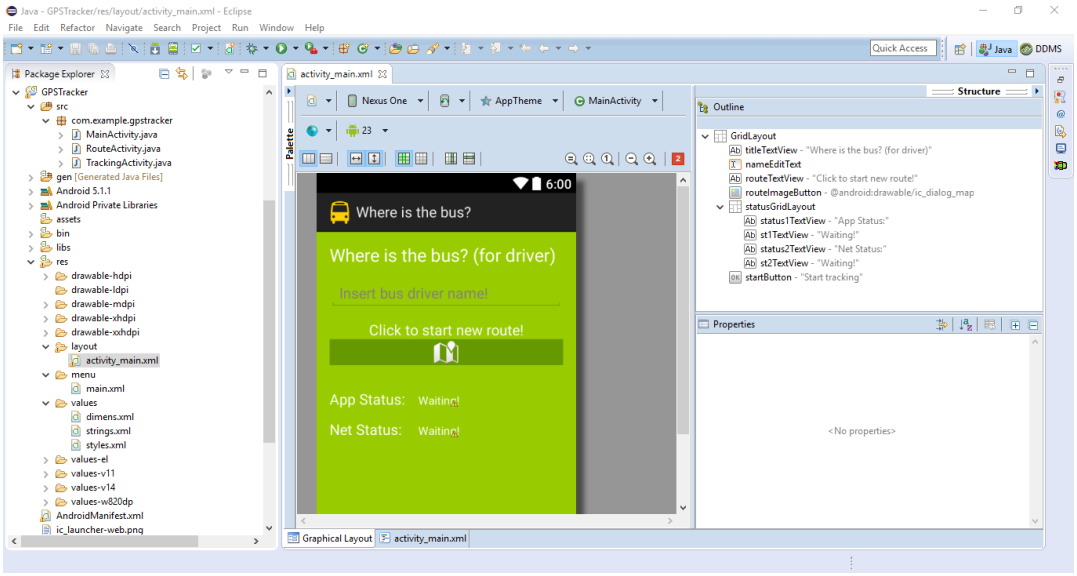
Βασικό κομμάτι της εργασίας είναι η εφαρμογή που θα πρέπει να έχει ο οδηγός του σχολικού λεωφορείου στο έξυπνο κινητό του τηλέφωνο. Προτού ξεκινήσει η συγγραφή του κώδικα για την ανάπτυξη της εφαρμογής, έγινε ανάλυση και πλήρης σχεδιασμός για τις δυνατότητες και τις απαιτήσεις που θα έχει η εφαρμογή, τις ιδιότητες και γενικότερα για το περιβάλλον του χρήστη.

Η επιτυχία μιας εφαρμογής ή και γενικότερα ενός προγράμματος κρίνεται από πολλούς παράγοντες. Ένας παράγοντας που μπορεί να ωθήσει μια εφαρμογή, έτσι ώστε να γίνει επιτυχημένη είναι η ευχρηστία, η ταχύτητα, η σταθερότητα και η χωρίς προβλήματα λειτουργία της. Το αποτέλεσμα της ανάλυσης ήταν, ότι θα χρειαστεί μια εφαρμογή, η οποία θα είναι αρκετά απλή, έτσι ώστε να μην χρειάζεται ο οδηγός του λεωφορείου αρκετές γνώσεις για να τη θέση σε λειτουργεία και να περιέχει όσο το δυνατό περισσότερα γραφικά για την πλήρη καθοδήγηση του χρήστη, όπως επίσης και να χρειάζεται λίγους πόρους από το σύστημα, έτσι ώστε να έχουμε μια καλή και ίσως πραγματική απόκριση στα κινητά των μαθητών.

Η εφαρμογή θα πρέπει να στέλνει το στίγμα της τοποθεσίας του λεωφορείου σε ένα web server, όπως και επιπλέον πληροφορίες για το λεωφορείο, όπως την ταχύτητα του λεωφορείου για τον υπολογισμό της απόστασης και του χρόνου άφιξης. Τα επιπλέον δεδομένα θα χρειαστούν για τον υπολογισμό των αποτελεσμάτων και χρησιμοποιήθηκαν αυτά που είναι απαραίτητα στη περίπτωση αυτή. Ο χρήστης πρέπει να εισάγει το όνομα του και να δημιουργεί τη νέα διαδρομή, όπως και να ξεκινάει την αποστολή δεδομένων της τοποθεσίας του. Κατά τη ανάλυση της εφαρμογής, ήταν αρκετά βοηθητικός ο χωρισμός της σε υποκατηγορίες, έτσι ώστε όταν ενωθούν να δώσουν το τελικό αποτέλεσμα. Τέλος επιλέχθηκε το όνομα «**Where is the bus?**» για το όνομα της εφαρμογής.

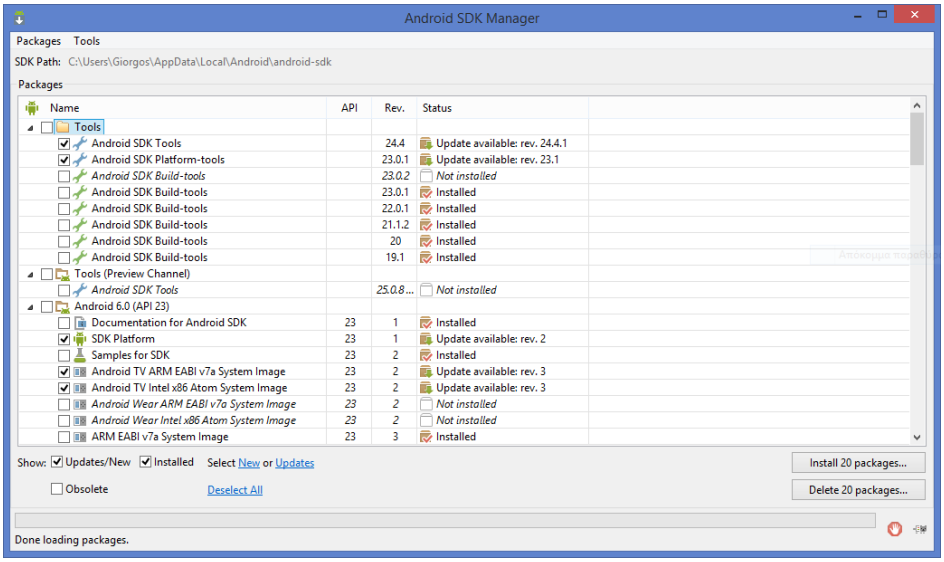
4.1.2 Eclipse & Android SDK

Προϋπόθεση για τη σχεδίαση και την υλοποίηση της συγκεκριμένης εφαρμογής ήταν να χρησιμοποιηθεί η γλώσσα προγραμματισμού Java και το Android SDK, έτσι ώστε να αναπτυχθεί μια native εφαρμογή για συσκευές με λειτουργικό σύστημα Android. Τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της εργασίας είναι το Eclipse, το οποίο είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης ιδανικό για την ανάπτυξη της συγκεκριμένης εργασίας, όπως και γενικότερα για εργασίες για Android. Έχει πάρα πολλές δυνατότητες και μπορεί να χρησιμοποιηθεί για την ανάπτυξη εργασιών σε πολλές διαφορετικές γλώσσες προγραμματισμού παρέχοντας και διαφορετικές τεχνολογίες και ταυτόχρονα παρέχει αρκετά βοηθητικά εργαλεία και επεκτάσεις στους προγραμματιστές.



Εικόνα 41: Eclipse – Παράδειγμα

Το Android SDK είναι ένα ολοκληρωμένο σύνολο εργαλείων ανάπτυξης, το οποίο περιλαμβάνει βιβλιοθήκες, τους προσομοιωτές, την τεκμηρίωση, δείγματα κώδικα και βοηθήματα για την ανάπτυξη εφαρμογών για το λειτουργικό σύστημα έξυπνων κινητών Android. Το Eclipse και το Android SDK πρέπει να εγκατασταθούν επιτυχώς στο σύστημα, πριν ξεκινήσει η συγγραφή του κώδικα της εφαρμογής.



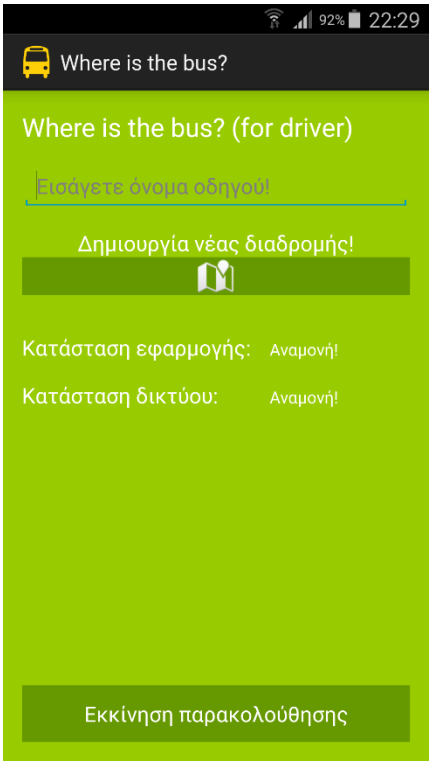
Εικόνα 42: Android SDK

4.1.3 User Interface εφαρμογής οδηγού

Αρχικά σχεδιάστηκε το περιβάλλον χρήστη (user interface) στο χαρτί και μετά στο Eclipse. Το περιβάλλον χρήστη αποτελείται από αρκετά απλά γραφικά, έτσι ώστε να είναι εύκολο στη λειτουργία του από τον οδηγό του λεωφορείου. Το περιβάλλον αποτελείται από ένα στοιχείο Edit-Text, στο οποίο ο οδηγός θα εισάγει το όνομα του και δυο στοιχεία Buttons, ένα για τη δημιουργία νέας διαδρομής και ένα για την αποστολή δεδομένων της τοποθεσίας του. Η λειτουργία του κάθε στοιχείο αναλύεται

παρακάτω. Από ότι βλέπουμε και στις εικόνες παρακάτω το UI της εφαρμογής είναι αρκετά εύχρηστο.

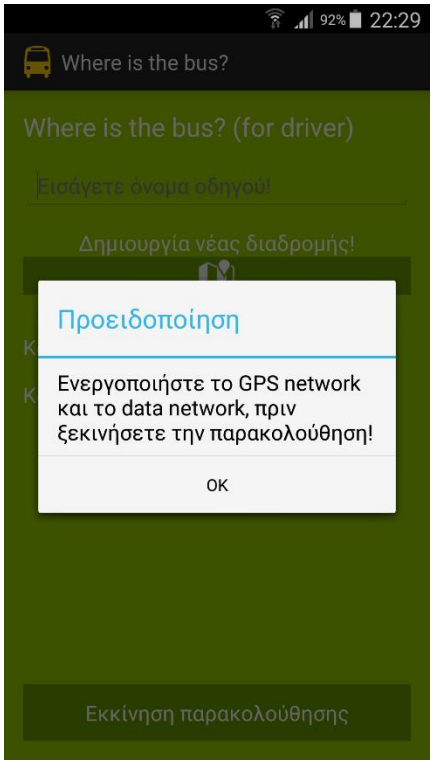
Το user interface της εφαρμογής του οδηγού είναι σχεδιασμένο, έτσι ώστε η χρήση του να είναι αρκετά εύκολη. Ξεκινώντας από το επάνω μέρος της οθόνης υπάρχει ο τίτλος, ένα περιθώριο για εισαγωγή κειμένου, ένα κουμπί για της δημιουργία νέας διαδρομής, δύο πεδία που θα εμφανίσουν την κατάσταση της σύνδεσης και της εφαρμογής και ένα κουμπί για την έναρξη της καταγραφής της διαδρομής.



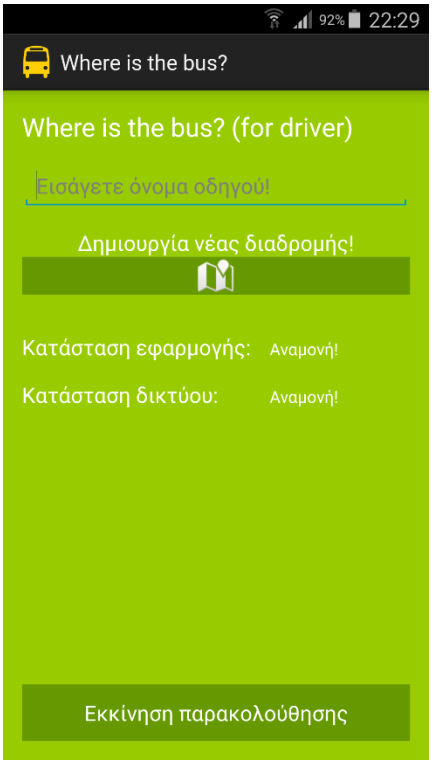
Εικόνα 43: Εφαρμογή Οδηγού – Βασική οθόνη

Ο οδηγός του λεωφορείου για να ξεκινήσει την καταγραφή της διαδρομής, πρώτα θα πρέπει να εισάγει το όνομα του και να πατήσει το κουμπί «**Δημιουργία νέας διαδρομή!**». Στη συνέχεια πρέπει να περιμένει να αλλάξουν οι καταστάσεις της εφαρμογής από «**Αναμονή!**» σε «**Route created OK**» και σε «**Code=1 – Delete**» αντίστοιχα. Αυτές οι δύο καταστάσεις είναι απαραίτητες, έτσι ώστε να ξέρει ο χρήστης, σε τη κατάσταση βρίσκεται η εφαρμογή ή εάν υπάρχει κάποιο πρόβλημα στην εφαρμογή ή στο δίκτυο κατά την επικοινωνία με τον web server. Η κατάσταση «**Κατάσταση εφαρμογής**» δείχνει στο χρήστη την κατάσταση που βρίσκεται η εφαρμογή και η κατάσταση «**Κατάσταση δικτύου**» την κατάσταση που βρίσκεται το δίκτυο.

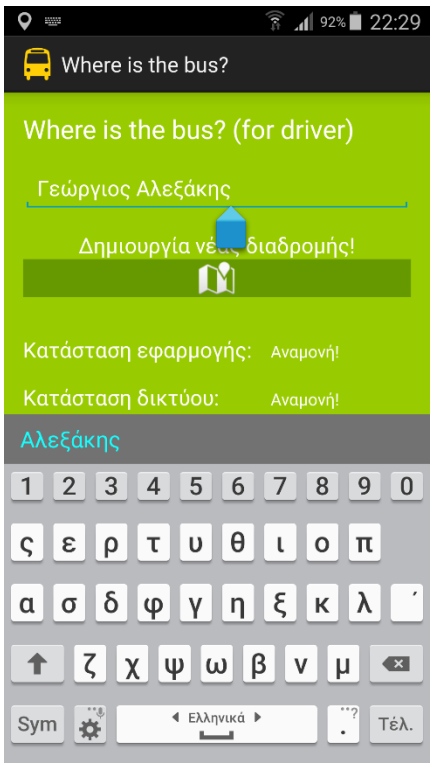
Όταν δημιουργηθεί η νέα διαδρομή μπορεί να ξεκινήσει η καταγραφή της διαδρομής. Είναι σημαντικό για λόγους ασφαλείας, ότι για να ενεργοποιηθεί το κουμπί «**Εκκίνηση παρακολούθησης**» πρέπει να έχει εισαγάγει ο χρήστης το όνομα του και να έχει δημιουργήσει μια νέα διαδρομή. Αυτή η ενέργεια έγινε, έτσι ώστε να αποφευχθούν οι απροσδιόριστες καταστάσεις, τις οποίες μπορεί να αντιμετωπίσει ο χρήστης. Για παράδειγμα εάν δεν έχει δημιουργηθεί η νέα διαδρομή δεν μπορεί να ξεκινήσει η καταγραφή.



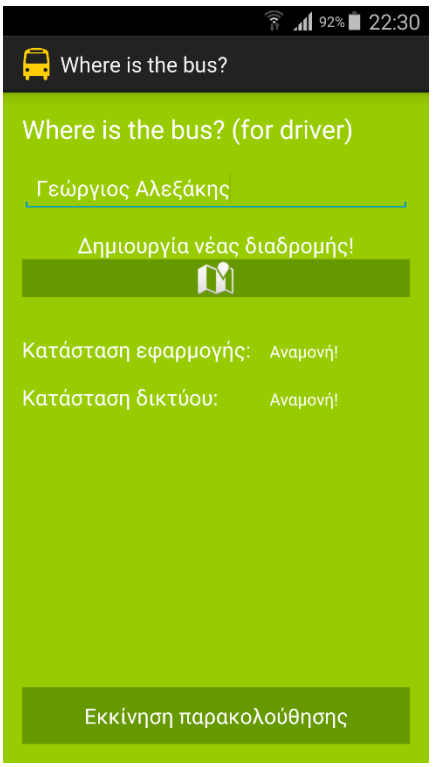
Εικόνα 44: Εφαρμογή Οδηγού – Μήνυμα αρχικής οθόνης



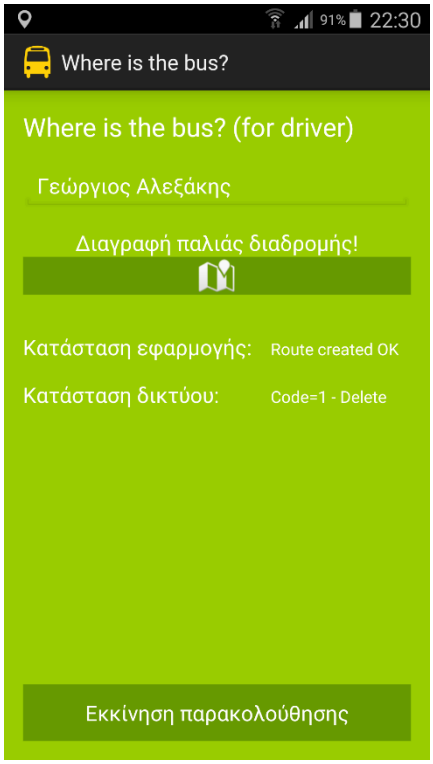
Εικόνα 45: Εφαρμογή Οδηγού – Βασική οθόνη



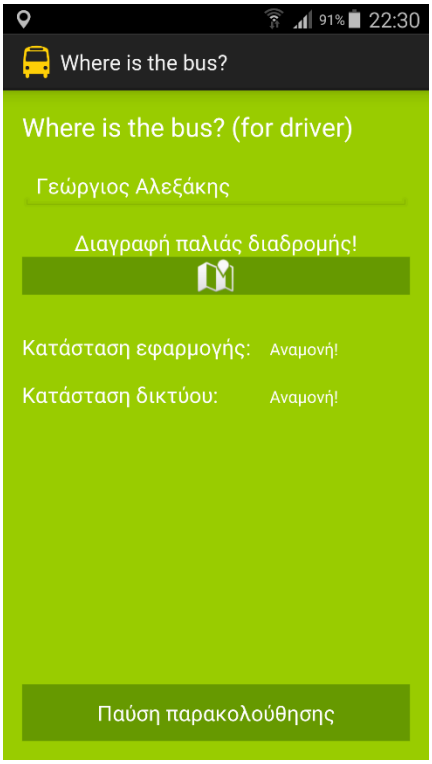
Εικόνα 46: Εφαρμογή Οδηγού – Εισαγωγή ονόματος χρήστη 1



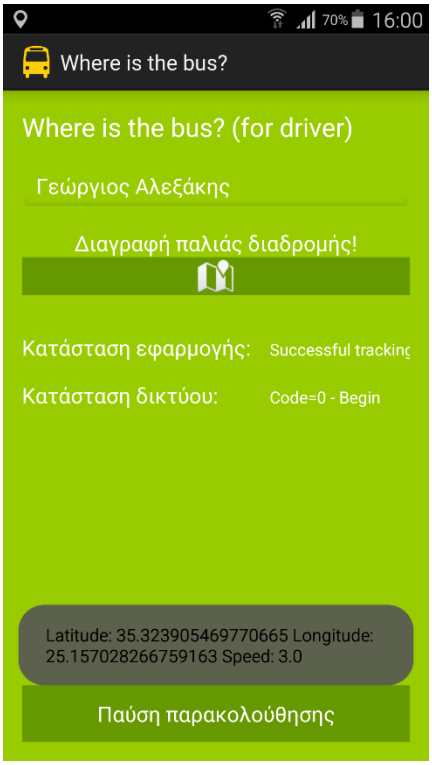
Εικόνα 47: Εφαρμογή Οδηγού – Εισαγωγή ονόματος χρήστη 2



Εικόνα 48: Εφαρμογή Οδηγού – Δημιουργία νέας διαδρομής



Εικόνα 49: Εφαρμογή Οδηγού – Εκκίνηση παρακολούθησης



Εικόνα 50: Εφαρμογή Οδηγού – Αποστολή θέσης

4.1.4 Ανάλυση κώδικα εφαρμογής οδηγού

Ο κώδικας της εφαρμογής του οδηγού είναι γραμμένος με τη γλώσσα προγραμματισμού Java και τη χρήση Android βιβλιοθηκών. Αυτή είναι η επόμενη υποκατηγορία της ανάλυσης της εφαρμογής του οδηγού. Η ανάλυση του κώδικα θα γίνει γραμμή προς γραμμή, όσο πιο αναλυτικά γίνεται.

Ο κώδικας της εφαρμογής του οδηγού είναι χωρισμένος σε τρείς κλάσεις. Η πρώτη κλάση είναι η **MainActivity**, στην οποία γίνεται αρχικοποίηση των περισσότερων μεταβλητών. Επίσης η **MainActivity** κάνει implement το **LocationListener,** έτσι ώστε να χρησιμοποιήσουμε τις υπηρεσίες τοποθεσίας του κινητού κάθε φορά που μεταβάλλεται η τοποθεσία του χρήστη. Αρχικά στις γραμμές 3 έως 17 δηλώνονται οι βιβλιοθήκες που χρησιμοποιήθηκαν. Μέσα στην κλάση γίνονται επίσης οι δηλώσεις των μεταβλητών που χρησιμοποιούνται στην κλάση. Αυτές οι μεταβλητές είναι κουμπιά, πεδία κειμένου, ακέραιοι και άλλες μεταβλητές.

Στην συνέχεια τρέχει η μέθοδος **onCreate()**, η οποία είναι βασική μέθοδος και είναι η μέθοδος που δημιουργεί το περιβάλλον της εφαρμογής. Μέσα στη μέθοδο, γίνεται αρχικοποίηση σε όλες τις μεταβλητές που χρησιμοποιούνται, συνδέονται τα γραφικά αντικείμενα με το γραφικό περιβάλλον της εφαρμογής όπως το αναλύσαμε προηγουμένως και ορίζονται οι listeners στα κουμπιά (γραμμές 29 - 65).

Για τη λήψη των συντεταγμένων θέσης του κινητού τηλεφώνου χρησιμοποιήθηκε ο location listener, ο οποίος κάθε φορά που αλλάζει η θέση του χρήστη βλέπει τη θέση του. Το πρόβλημα σε αυτή την περίπτωση ήταν, για την παραμικρή αλλαγή της θέσης του λεωφορείο, θα είχε ως αποτέλεσμα τη λήψη συντεταγμένων και την αποστολή τους στο web server, με αποτέλεσμα την υπερφόρτωση και την υπερβολική καθυστέρηση. Η λύση ήταν να γίνετε λήψη κάθε 10 δευτερόλεπτα ή κάθε 10 μέτρα διαφορά από τη προηγούμενη λήψη συντεταγμένων (γραμμές 63 - 64). Οι συγκεκριμένες ρυθμίσεις δόθηκαν στην εφαρμογή μετά από πραγματική δοκιμή στο δρόμο με όχημα και ταχύτητές που πλησιάζουν αρκετά την ταχύτητα του λεωφορείου. Για την υλοποίηση του location listener πρέπει να γίνει υλοποίηση 4 μεθόδων (**onLocationChanged**, **onStatusChanged**, **onProviderChanged**, **onProviderEnabled**) (γραμμές 119 - 159). Στη συγκεκριμένη εφαρμογή χρησιμοποιείται μόνο η πρώτη μέθοδος η **onLocationChanged()**, η οποία πυροδοτείτε όταν η θέση του κινητού αλλάζει. Η μέθοδος ξεκινάει να στέλνει δεδομένα στο web server μόνο όταν πατηθεί το κουμπί «**Εκκίνηση παρακολούθησης**», δηλαδή ότι η μεταβλητή **enableTracking** είναι ίση με true. Στην περίπτωση αυτή αποθηκεύονται οι συντεταγμένες στις μεταβλητές latitude, longitude και η ταχύτητα αφού έχει μετατραπεί από m/s σε km/h.

Για την ενημέρωση και του οδηγού ότι η εφαρμογή είναι σε πλήρη λειτουργία εμφανίζεται ένα παράθυρο στο χρήστη που του δείχνει τις τρέχουσες συντεταγμένες και την τρέχουσα ταχύτητα του με ένα toast κάθε φορά που ενημερώνεται και ο web server (γραμμές 126 - 130). Τελευταία λειτουργεία της μεθόδου είναι η δημιουργία ενός αντικειμένου της κλάσης **TrackingActivity**, η οποία είναι υπεύθυνη για την αποστολή των δεδομένων στο web server. Ο κώδικας της συγκεκριμένης κλάσης θα αναλυθεί παρακάτω (γραμμές 132 - 138).

Η λειτουργία δημιουργίας νέας διαδρομής ήταν αναγκαία να εισαχθεί στην εφαρμογή. Στην πραγματικότητα, κατά τη δημιουργία νέας διαδρομής, οι ενέργειες που γίνονται είναι η διαγραφή της παλιάς διαδρομής και η προετοιμασία της εφαρμογής να στείλει νέα δεδομένα στο web server. Για τη δημιουργία νέας διαδρομής

έχει εισαχθεί ένα κουμπί στο γραφικό περιβάλλον. Όταν πατηθεί πραγματοποιούνται όλες οι ενέργειες για τη δημιουργία νέας διαδρομής. Πιο συγκεκριμένα (γραμμές 73 - 98) για τη λειτουργία του κουμπιού πρέπει να γίνει υλοποίηση του **OnClickListener()**. Αρχικά εμφανίζονται οι δύο καταστάσεις σε «**Αναμονή!**» και στη συνέχεια τρέχει η κλάση για τη δημιουργία νέας διαδρομής, η **RouteActivity**, η οποία θα αναλυθεί εκτενέστερα παρακάτω. Στη συνέχεια απενεργοποιείται το πεδίο, στο οποίο ο χρήστης εισάγει το όνομα του και αλλάζει ο τίτλος του κουμπιού σε «**Διαγραφή παλιάς διαδρομής**». Αν πατηθεί ξανά το κουμπί γίνονται κάποιες κοινές ενέργειες, όπως η δημιουργία νέας διαδρομή.

Η εφαρμογή ξεκινάει να στέλνει δεδομένα, όταν ο χρήστης πατήσει το κουμπί «**Εκκίνηση παρακολούθησης**». Όπως και στα άλλα κουμπιά, έτσι και εδώ χρησιμοποιήθηκε ο **OnClicckListener()** για τη λειτουργία του. Υπάρχουν δυο περιπτώσεις να πατήσει ο χρήστης το κουμπί. Η μια είναι να θέλει να ξεκινήσει την παρακολουθήσει και η άλλη είναι να τη σταματήσει. Στην πρώτη περίπτωση η μεταβλητή **enableTracking** είναι ίση με false. Όταν πατηθεί το κουμπί, γίνεται ίση με true, δηλαδή όπως είδαμε στη μέθοδο **onLocationChanged()**, μόνο τότε θα τρέξει ο κώδικας για να ξεκινήσει η αποστολή δεδομένων. Στην συνέχεια αλλάζει η μεταβλητή σε false, το κείμενο του κουμπιού σε «**Παύση παρακολούθησης**» και εμφανίζονται οι καταστάσεις σε «**Αναμονή!**» μέχρι να ολοκληρωθεί η διαδικασία και να εμφανιστούν οι καταστάσεις της εφαρμογής. Στη δεύτερη περίπτωση στην οποία δηλαδή γίνεται σταμάτημα της παρακολούθησης, η μεταβλητή **enableTracking** γίνεται ίση με false, το κείμενο του κουμπιού αλλάζει σε «**Εκκίνηση παρακολούθησης**» και τέλος εμφανίζονται οι καταστάσεις σε «**Αναμονή!**» μέχρι να ολοκληρωθεί η διαδικασία και να εμφανιστούν οι καταστάσεις της εφαρμογής (γραμμές 100 - 117).

```
1.  package com.example.gpstracker;
2.
3.  import android.app.Activity;
4.  import android.app.AlertDialog;
5.  import android.content.Context;
6.  import android.content.DialogInterface;
7.  import android.location.Location;
8.  import android.location.LocationListener;
9.  import android.location.LocationManager;
10. import android.os.Bundle;
11. import android.view.View;
12. import android.view.View.OnClickListener;
13. import android.widget.Button;
14. import android.widget.EditText;
15. import android.widget.ImageButton;
16. import android.widget.TextView;
17. import android.widget.Toast;
18.
19. public class MainActivity extends Activity implements LocationListener {
20.
21.     private TextView status, role, routeTextView;
22.     private ImageButton routeImageButton;
23.     private Button startButton;
24.     private String name, speed, latitude, longitude, deletePassword;
25.     private Boolean enableTracking, enableButtons;
26.     private EditText nameEditText;
27.     private double speedConverter;
28.
29.     @Override
30.     protected void onCreate(Bundle savedInstanceState) {
31.         super.onCreate(savedInstanceState);
```

```
32.         setContentView(R.layout.activity_main);
33.
34.         AlertDialog.Builder builder = new AlertDialog.Builder(this);
35.         builder.setMessage(R.string.alert_message).setTitle(R.string.alert);

36.         builder.setPositiveButton(R.string.ok,
37.                 new DialogInterface.OnClickListener() {
38.                     public void onClick(DialogInterface dialog, int id) {
39.                         dialog.cancel();
40.                     }
41.                 });
42.         AlertDialog dialog = builder.create();
43.         dialog.show();
44.
45.         enableButtons = false;
46.         enableTracking = false;
47.         deletePassword = "1234";
48.
49.         routeTextView = (TextView) findViewById(R.id.routeTextView);
50.         status = (TextView) findViewById(R.id.st1TextView);
51.         role = (TextView) findViewById(R.id.st2TextView);
52.
53.         nameEditText = (EditText) findViewById(R.id.nameEditText);
54.
55.         routeImageButton = (ImageButton) findViewById(R.id.routeImageButton)
    ;
56.         routeImageButton.setOnClickListener(newRoute);
57.
58.         startButton = (Button) findViewById(R.id.startButton);
59.         startButton.setOnClickListener(startTracking);
60.         startButton.setEnabled(false);
61.
62.         // Update time = 10 second and update distance = 10 meters
63.         LocationManager lm = (LocationManager) getSystemService(Context.LOCA
    TION_SERVICE);
64.         lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 10000, 10, t
    his);
65.     }
66.
67.     // Set text to text-views waiting status
68.     public void showStatus() {
69.         status.setText(getResources().getString(R.string.waiting));
70.         role.setText(getResources().getString(R.string.waiting));
71.     }
72.
73.     public OnClickListener newRoute = new OnClickListener() {
74.
75.         @Override
76.         public void onClick(View v) {
77.             showStatus(); // Show waiting status
78.             if (enableButtons == false) {
79.                 new RouteActivity(status, role, startButton, enableButtons)

80.                         .execute(deletePassword);
81.                 nameEditText.setEnabled(false);
82.                 routeTextView.setText(getResources().getString(
83.                         R.string.delete_route));
84.                 enableButtons = true;
85.             } else if (enableButtons == true) {
86.                 new RouteActivity(status, role, startButton, enableButtons)

87.                         .execute(deletePassword);
88.                 nameEditText.setEnabled(true);
89.                 routeTextView.setText(getResources().getString(
90.                         R.string.new_route));
91.                 enableButtons = false;
```

```
                startButton.setText(getResources().getString(
                        R.string.start_tracking));
                startButton.setEnabled(false);
            }
        }

    };

    public OnClickListener startTracking = new OnClickListener() {

        @Override
        public void onClick(View v) {
            if (enableTracking == false) {
                enableTracking = true;
                startButton.setText(getResources().getString(
                        R.string.stop_tracking));
                showStatus(); // Show waiting status
            } else if (enableTracking == true) {
                enableTracking = false;
                startButton.setText(getResources().getString(
                        R.string.start_tracking));
                showStatus(); // Show waiting status
            }
        }

    };

    @Override
    public void onLocationChanged(Location location) {
        if (enableTracking == true) {

            // Convert speed from m/s to km/h and round it
            speedConverter = (Math.round(location.getSpeed()) * 3600)
     / 1000;

            String toastString = "Latitude: " + location.getLatitude(
    )
                    + " Longitude: " + location.getLongitude() + " Sp
    eed: "
                    + speedConverter;
            Toast.makeText(getBaseContext(), toastString, Toast.LENGT
    H_LONG)
                    .show();

            name = "" + nameEditText.getText();
            latitude = "" + location.getLatitude();
            longitude = "" + location.getLongitude();
            speed = "" + speedConverter;

            new TrackingActivity(status, role).execute(name, latitude
    ,
                    longitude, speed);
        }

    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle e
    xtras) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onProviderEnabled(String provider) {
        // TODO Auto-generated method stub
```

```
152.
153.            }
154.
155.            @Override
156.            public void onProviderDisabled(String provider) {
157.                // TODO Auto-generated method stub
158.
159.            }
160.        }
```

Η κλάση **TrackingActivity** είναι υπεύθυνη για τη σύνδεση της εφαρμογής οδηγού με το web server. Επίσης είναι υπεύθυνη για την αποστολή των δεδομένων στο web server την λήψη της απόκρισης του web server και την ενημέρωση της **MainActivity** για μηνύματα σφαλμάτων.

Αρχικά γίνεται η δήλωση των βιβλιοθηκών που χρησιμοποιήθηκαν στην κλάση (γραμμές 1 - 10). Στη συνέχεια η κλάση **TrackingActivity** δηλώνεται ως κλάση παιδί της **AsyncTask,** δηλαδή κάνει extends την **AsyncTask**, έτσι ώστε να κληρονομήσει όλες τις μεθόδους που θα χρησιμοποιηθούν παρακάτω. Γίνεται δήλωση και αρχικοποίηση όλων των στοιχείων και μεταβλητών που χρησιμοποιήθηκαν (γραμμές 12 - 18) και στη συνέχεια τρέχει η μέθοδος **doInBackground**, η οποία δέχεται για ορίσματα τις συντεταγμένες, το όνομα και την ταχύτητα του οδηγού του λεωφορείου (γραμμές 23 - 26). Το String με όνομα link περιέχει τη διεύθυνση, την οποία βρίσκεται το αρχείο στο web server, στον οποίο πρόκειται να αποσταλούν τα δεδομένα. Το String με όνομα data περιέχει τα δεδομένα τα οποία πρόκειται να αποσταλούν στο web server σε κωδικοποίηση UTF-8 (γραμμές 28 - 36). Στη συνέχεια δημιουργείται μια νέα σύνδεση και αποστέλλονται τα δεδομένα με τη μέθοδο POST (γραμμές 38 - 46). Στις γραμμές 48 έως 58 γίνεται λήψη της απόκρισης του web server.

Η μέθοδος **onPostExecute** εμφανίζει το αποτέλεσμα της απόκρισης του web server σε μια συνθήκη, η οποία διαχωρίζει τα μηνύματα σε επιτυχής και ανεπιτυχείς εκτελέσεις (γραμμές 66 - 76).

```
1.  package com.example.gpstracker;
2.
3.  import java.io.BufferedReader;
4.  import java.io.InputStreamReader;
5.  import java.io.OutputStreamWriter;
6.  import java.net.URL;
7.  import java.net.URLConnection;
8.  import java.net.URLEncoder;
9.  import android.os.AsyncTask;
10. import android.widget.TextView;
11.
12. public class TrackingActivity extends AsyncTask<String, Void, String> {
13.     private TextView statusField, roleField;
14.
15.     public TrackingActivity(TextView statusField, TextView roleField) {
16.         this.statusField = statusField;
17.         this.roleField = roleField;
18.     }
19.
20.     @Override
21.     protected String doInBackground(String... arg0) {
22.         try {
23.             String name = (String) arg0[0];
24.             String latitude = (String) arg0[1];
```

```
25.             String longitude = (String) arg0[2];
26.             String speed = (String) arg0[3];
27.
28.             String link = "http://g-alex.net78.net/gps/index.php";
29.             String data = URLEncoder.encode("name", "UTF-8") + "="
30.                     + URLEncoder.encode(name, "UTF-8");
31.             data += "&" + URLEncoder.encode("latitude", "UTF-8") + "="
32.                     + URLEncoder.encode(latitude, "UTF-8");
33.             data += "&" + URLEncoder.encode("longitude", "UTF-8") + "="
34.                     + URLEncoder.encode(longitude, "UTF-8");
35.             data += "&" + URLEncoder.encode("speed", "UTF-8") + "="
36.                     + URLEncoder.encode(speed, "UTF-8");
37.
38.             URL url = new URL(link);
39.             URLConnection conn = url.openConnection();
40.
41.             conn.setDoOutput(true);
42.             OutputStreamWriter wr = new OutputStreamWriter(
43.                     conn.getOutputStream());
44.
45.             wr.write(data);
46.             wr.flush();
47.
48.             BufferedReader reader = new BufferedReader(new InputStreamReader
    (
49.                     conn.getInputStream()));
50.
51.             StringBuilder sb = new StringBuilder();
52.             String line = null;
53.
54.             // Read Server Response
55.             while ((line = reader.readLine()) != null) {
56.                 sb.append(line);
57.                 break;
58.             }
59.
60.             return sb.toString();
61.         } catch (Exception e) {
62.             return new String("Exception: " + e.getMessage());
63.         }
64.     }
65.
66.     @Override
67.     protected void onPostExecute(String result) {
68.         if (!result.equals("Code=0 - Begin")) {
69.             this.statusField.setText("Connection error");
70.             this.roleField.setText(result);
71.         } else {
72.             this.statusField.setText("Successful tracking");
73.             this.roleField.setText(result);
74.         }
75.     }
76. }
```

Η κλάση **RouteActivity** εκτελεί ακριβώς τις ίδιες λειτουργίες με την κλάση **TrackingActivity** με τη μοναδική διαφορά ότι σε αυτή την περίπτωση αντί να γίνεται αποστολή δεδομένων προς το web server, εκτελείται η διαγραφή των δεδομένων που υπάρχουν στη βάση δεδομένων του web server, έτσι ώστε να ξεκινήσει μία νέα διαδρομή.

Η διαγραφή της προηγούμενης διαδρομής εκτελείται με την αποστολή του κωδικού διαγραφής, ο οποίος είναι «**1234**». Τις υπόλοιπες λειτουργίες τις εκτελεί ο

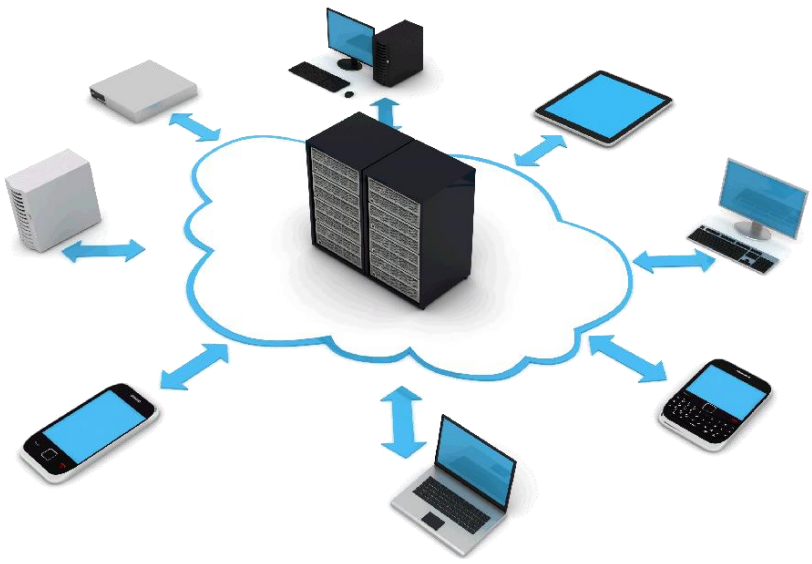
web server. Στο τέλος γίνεται λήψη της απόκρισης του web server, έτσι ώστε να ενημερωθεί ο χρήστης για την κατάσταση της εφαρμογής.

```
1.  package com.example.gpstracker;
2.
3.  import java.io.BufferedReader;
4.  import java.io.InputStreamReader;
5.  import java.io.OutputStreamWriter;
6.  import java.net.URL;
7.  import java.net.URLConnection;
8.  import java.net.URLEncoder;
9.  import android.os.AsyncTask;
10. import android.widget.Button;
11. import android.widget.TextView;
12.
13. public class RouteActivity extends AsyncTask<String, Void, String> {
14.     private TextView statusField, roleField;
15.     private Button start;
16.     private Boolean enable;
17.
18.     public RouteActivity(TextView statusField, TextView roleField,
19.             Button start, Boolean enable) {
20.         this.statusField = statusField;
21.         this.roleField = roleField;
22.         this.start = start;
23.         this.enable = enable;
24.     }
25.
26.     @Override
27.     protected String doInBackground(String... arg0) {
28.         try {
29.             String deleteCode = (String) arg0[0];
30.
31.             String link = "http://g-alex.net78.net/gps/delete.php";
32.             String data = URLEncoder.encode("deleteCode", "UTF-8") + "="
33.                     + URLEncoder.encode(deleteCode, "UTF-8");
34.
35.             URL url = new URL(link);
36.             URLConnection conn = url.openConnection();
37.
38.             conn.setDoOutput(true);
39.             OutputStreamWriter wr = new OutputStreamWriter(
40.                     conn.getOutputStream());
41.
42.             wr.write(data);
43.             wr.flush();
44.
45.             BufferedReader reader = new BufferedReader(new InputStreamReader
    (
46.                     conn.getInputStream()));
47.
48.             StringBuilder sb = new StringBuilder();
49.             String line = null;
50.
51.             // Read Server Response
52.             while ((line = reader.readLine()) != null) {
53.                 sb.append(line);
54.                 break;
55.             }
56.
57.             return sb.toString();
58.         } catch (Exception e) {
59.             return new String("Exception: " + e.getMessage());
```

```
60.         }
61.     }
62.
63.     @Override
64.     protected void onPostExecute(String result) {
65.         if (enable == false) {
66.             if (!result.equals("Code=1 - Delete")) {
67.                 this.statusField.setText("Connection error");
68.                 this.roleField.setText(result);
69.             } else {
70.                 this.statusField.setText("Route created OK");
71.                 this.roleField.setText(result);
72.                 start.setEnabled(true);
73.             }
74.         }
75.         if (enable == true) {
76.             if (!result.equals("Code=1 - Delete")) {
77.                 this.statusField.setText("Connection error");
78.                 this.roleField.setText(result);
79.             } else {
80.                 this.statusField.setText("Delete OK");
81.                 this.roleField.setText(result);
82.                 start.setEnabled(false);
83.             }
84.         }
85.     }
86. }
```

4.2 Web server

4.2.1 Εισαγωγή web server



Εικόνα 51: Web server

Το επόμενο βήμα για την υλοποίηση της εργασίας, είναι η παραλαβή των δεδομένων από τον web server. Ο web server είναι ο ενδιάμεσος κρίκος ανάμεσα στις δύο εφαρμογές. Διαχειρίζεται τα δεδομένα εισόδου από την εφαρμογή του οδηγού του

λεωφορείου και τα αποθηκεύει στη βάση δεδομένων, όπως επίσης διαχειρίζεται και τα δεδομένα εξόδου, τα οποία στέλνονται στην εφαρμογή του μαθητή.

Αυτή η κατηγορία χωρίστηκε σε δύο απλούστερες κατηγορίες για την εύκολη ανάλυση και επίλυση του προβλήματος. Η πρώτη υποκατηγορία αφορά τα δεδομένα εισόδου και την αποθήκευση τους και η δεύτερη υποκατηγορία την ανάκτηση τους και την αποστολή τους στην εφαρμογή του μαθητή.

4.2.2 Ανάλυση απαιτήσεων web server

Όλη η εργασία βασίζεται στην αποστολή δεδομένων από μια εφαρμογή σε μια άλλη. Ο ενδιάμεσος κόμβος των δύο εφαρμογών είναι πολύ σημαντικός και παίζει καθοριστικό ρόλο στην απόδοση των εφαρμογών. Για την υλοποίηση της συγκεκριμένης εργασίας χρησιμοποιήθηκε ένας online web server, ο οποίος μας παρέχει τη δυνατότητα της διαχείρισης βάσεων δεδομένων MySQL και υποστήριξης PHP. Η απόδοση του είναι αρκετά καλή και ικανοποίει πλήρως της απαιτήσεις της εργασίας, αν και το λογισμικό του δεν είναι πλήρως ενημερωμένο.

Ο λόγος για τον οποίο δεν χρησιμοποιήθηκε ένας τοπικός web server με τη βοήθεια για παράδειγμα του λογισμικού XAMPP, είναι η ανάγκη για δοκιμή της εργασίας σε πραγματικές συνθήκες, όπως για παράδειγμα η δειγματοληψία της τοποθεσίας, το οποίο δεν θα ήταν εφικτό αν γινόταν χρήση τοπικού web server. Στο διαδίκτυο υπάρχουν αρκετές εταιρείες παροχής υπηρεσιών διαδικτύου, οι οποίες παρέχουν κάποιες υπηρεσίες φιλοξενίας δωρεάν, μια από τις οποίες χρησιμοποιήθηκε για την εργασία.

3.2.3 Βάση δεδομένων web server

Η εφαρμογή του οδηγού για κάθε σημείο, το οποίο λαμβάνονται δεδομένα, στέλνει ένα HTTP μήνυμα μέσω της μεθόδου POST. Ο Apache HTTP web server λαμβάνει τα δεδομένα εισόδου, τα οποία είναι οι συντεταγμένες, το όνομα του οδηγού, η ταχύτητα του λεωφορείου και ένας κωδικός ο οποίος ασφαλίζει όλη την υπηρεσία από τυχόν προβλήματα. Στη συνέχεια αποθηκεύονται όλα τα δεδομένα στη βάση δεδομένων. Η βάση δεδομένων αποτελείται από 6 πεδία τα οποία είναι όλα εξίσου σημαντικά για την υλοποίηση της εργασίας. Τα πεδία είναι: το **gps_id**, **latitude**, **longitude**, **speed**, **name**, **reg_date**, δηλαδή το μοναδικό κλειδί, οι συντεταγμένες, η ταχύτητα του λεωφορείο, το όνομα του οδηγού και η ημερομηνία της κάθε εγγραφής.

```
1.  -- phpMyAdmin SQL Dump
2.  -- version 2.11.4
3.  -- http://www.phpmyadmin.net
4.  --
5.  -- Host: localhost
6.  -- Generation Time: Mar 01, 2016 at 02:43 PM
7.  -- Server version: 5.1.57
8.  -- PHP Version: 5.2.17
9.
10. SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
11.
12. --
13. -- Database: `a8690258_gps`
14. --
15.
```

```
-- --------------------------------------------------------

--
-- Table structure for table `gps_data`
--

CREATE TABLE `gps_data` (
  `gps_id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) COLLATE latin1_general_ci NOT NULL,
  `latitude` varchar(50) COLLATE latin1_general_ci NOT NULL,
  `longitude` varchar(50) COLLATE latin1_general_ci NOT NULL,
  `speed` varchar(50) COLLATE latin1_general_ci NOT NULL,
  `reg_date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`gps_id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci AUTO_INCREMENT=1 ;

--
-- Dumping data for table `gps_data`
--
```

Η εφαρμογή που θα έχει ο μαθητής στο κινητό του θα πρέπει να λαμβάνει τα δεδομένα της βάσης δεδομένων και να τα εμφανίζει στο χάρτη. Η εργασία αυτή υλοποιήθηκε χρησιμοποιώντας ένα αρχείο με μορφοποίηση JSON. Αρχικά γίνεται ανάκτηση των δεδομένων από την βάση δεδομένων μέσω SQL queries και στη συνέχεια τα δεδομένα στέλνονται στην εφαρμογή του μαθητή με τη JSON μορφοποίηση.

4.2.4 Ανάλυση κώδικα web server

Εικόνα 52: Παράδειγμα Web Server

Η εφαρμογή του web server διαχωρίστηκε σε 4 διαφορετικά αρχεία. Το πρώτο περιέχει τον κώδικα εγγραφής δεδομένων στη βάση δεδομένων, το δεύτερο τη διαγραφή των δεδομένων, το τρίτο την αποστολή δεδομένων με μορφοποίηση JSON στην εφαρμογή χρήστη και το τελευταίο περιέχει όλες τις ευαίσθητες πληροφορίες που χρειάζονται για τη σύνδεση του web server με τη βάση δεδομένων, όπως το username το password, το dbname και το servername.

Καθώς λαμβάνονται τα δεδομένα από το web server, ξεκινάει η εφαρμογή διαχείρισης τους. Το αρχείο το οποίο είναι υπεύθυνο για τους κωδικούς πρόσβασης,

όνομα χρήστη και όλες οι υπόλοιπες πληροφορίες βρίσκονται στο αρχείο **db_information.php** και χρησιμοποιούνται σε κάθε αρχείο που κάνει διαχείρισης της βάσης από το web server. Δημιουργείται μια νέα σύνδεση με τη βάση δεδομένων για την αποθήκευση των δεδομένων. Σε περίπτωση που υπάρξει κάποιο σφάλμα κατά τη δημιουργία σύνδεσης εμφανίζει το μήνυμα λάθους.

Στη συνέχεια αποθηκεύονται τα δεδομένα που στάλθηκαν από την εφαρμογή οδηγού, που είναι αποθηκευμένα στις **Super Global $_POST** μεταβλητές, έτσι ώστε να γίνει αποθήκευση στη βάση δεδομένων. Πριν πραγματοποιηθεί η νέα εγγραφή στη βάση ελέγχεται αν δυο μεταβλητές ($longitude, $latitude) είναι κενές, έτσι ώστε να αποφεύγονται οι κενές εγγραφές. Στη συνέχεια πραγματοποιείται η εγγραφή στη βάση δεδομένων. Στην περίπτωση που η εγγραφή πραγματοποιηθεί με επιτυχία εμφανίζεται το μήνυμα «**Code=0 – Begin**», το οποίο η εφαρμογή οδηγού, το δέχεται ως είσοδο. Σε αντίθετη περίπτωση εμφανίζεται το μήνυμα λάθους στην περίπτωση σφάλματος ή το μήνυμα «**Waiting gps data!**» στην περίπτωση που δεν έχουν αποσταλεί δεδομένα και τρέχουμε το αρχείο. Τέλος κλείνει η σύνδεση με τη βάση.

```php
<?php

// Connection information
include('db_information.php');

$name = $_POST['name'];
$latitude = $_POST['latitude'];
$longitude = $_POST['longitude'];
$speed = $_POST['speed'];

if (strlen($longitude) > 0 && strlen($latitude) > 0) {
    try {
        $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $sql = "INSERT INTO gps_data (name, latitude, longitude, speed) VALUES ('$name', '$latitude', '$longitude', '$speed')";
        $conn->exec($sql);
        $data = "Code=0 - Begin";
        echo $data;
    } catch (PDOException $e) {
        echo $sql . "<br>" . $e->getMessage();
    }
    $conn = null;
} else {
    $data = "Waiting gps data!";
    echo $data;
}
?>
```

Η επόμενη λειτουργία που πρέπει να πραγματοποιείται είναι η διαγραφή της προηγούμενης διαδρομής, δηλαδή τη διαγραφή όλων των αρχείων της βάσης δεδομένων, για τη δημιουργία νέας διαδρομής στη συνέχεια. Η λογική της υλοποίησης είναι η ίδια με την αποθήκευση των δεδομένων μόνο που αυτή τη φορά το μόνο που αποστέλλει η εφαρμογή του οδηγού είναι ο κωδικός διαγραφής.

Ο κωδικός διαγραφής είναι ένας κωδικός που έχει οριστεί και στην περίπτωση της συγκεκριμένης υλοποίησης είναι «**1234**» και χρησιμοποιήθηκε μόνο για λόγους ασφαλείας. Για τη διαγραφεί όλων των δεδομένων έγινε χρήση της εντολής

TRUNCATE TABLE gps_data, με την οποία διαγράφονται όλα τα δεδομένα του πίνακα της βάσης δεδομένων. Στην περίπτωση που η διαγραφή ολοκληρωθεί με επιτυχία αποστέλλεται στην εφαρμογή οδηγού το μήνυμα «**Code=1 – Delete**» αλλιώς αποστέλλεται το μήνυμα σφάλματος. Τέλος κλείνει η σύνδεση με τη βάση δεδομένων.

```
1.  <?php
2.
3.  // Connection information
4.  include('db_information.php');
5.
6.  $deleteCode = $_POST['deleteCode'];
7.
8.  if (strcmp($deleteCode, "1234") == 0) {
9.      try {
10.         $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
    $password);
11.         $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
12.         $sql = "TRUNCATE TABLE gps_data";
13.         $conn->exec($sql);
14.         $data = "Code=1 - Delete";
15.         echo $data;
16.     } catch (PDOException $e) {
17.         echo $sql . "<br>" . $e->getMessage();
18.     }
19.     $conn = null;
20. }
21. ?>
```

Πολύ σημαντική είναι η ανάκτηση δεδομένων από το web server προς την εφαρμογή του μαθητή. Η μορφοποίηση των δεδομένων που αποστέλλονται είναι η μορφοποίηση JSON, η οποία είναι αρκετά εύχρηστη για τη διαχείριση δεδομένων. Οι ενέργειες που γίνονται για τη αποστολή δεδομένων είναι η δημιουργία σύνδεσης με τη βάση, η επιλογή των δεδομένων για αποστολή και τέλος η μορφοποίηση JSON, η οποία θα λάβει η εφαρμογή μαθητή. Στο τέλος κλείνει η σύνδεση με τη βάση δεδομένων.

```
1.  <?php
2.
3.  header("Access-Control-Allow-Origin: *");
4.  header("Content-Type: application/json; charset=UTF-8");
5.
6.  // Connection information
7.  include('db_information.php');
8.
9.  $conn = new mysqli($servername, $username, $password, $dbname);
10. $result = $conn-
    >query("SELECT name, latitude, longitude, speed FROM gps_data ORDER BY gps_i
    d DESC");
11.
12. $outp = "[";
13. while ($rs = $result->fetch_array(MYSQLI_ASSOC)) {
14.     if ($outp != "[") {
15.         $outp .= ",";
16.     }
17.     $outp .= '{"name":"' . $rs["name"] . '",';
18.     $outp .= '"lat":"' . $rs["latitude"] . '",';
19.     $outp .= '"lng":"' . $rs["longitude"] . '",';
20.     $outp .= '"speed":"' . $rs["speed"] . '"}';
```

```
21. }
22. $outp .="]";
23.
24. $conn->close();
25.
26. echo($outp);
27. ?>
```

Το αρχείο αυτό περιέχει όλες τις πληροφορίες για τη σύνδεση με τη βάση δεδομένων.

```
1.  <?php
2.
3.  //Connection information
4.  $servername = "****";
5.  $username = "a****_gps";
6.  $password = "****";
7.  $dbname = "a****_gps";
8.  ?>
```

4.3 Εφαρμογή μαθητή

4.3.1 Εισαγωγή εφαρμογής μαθητή

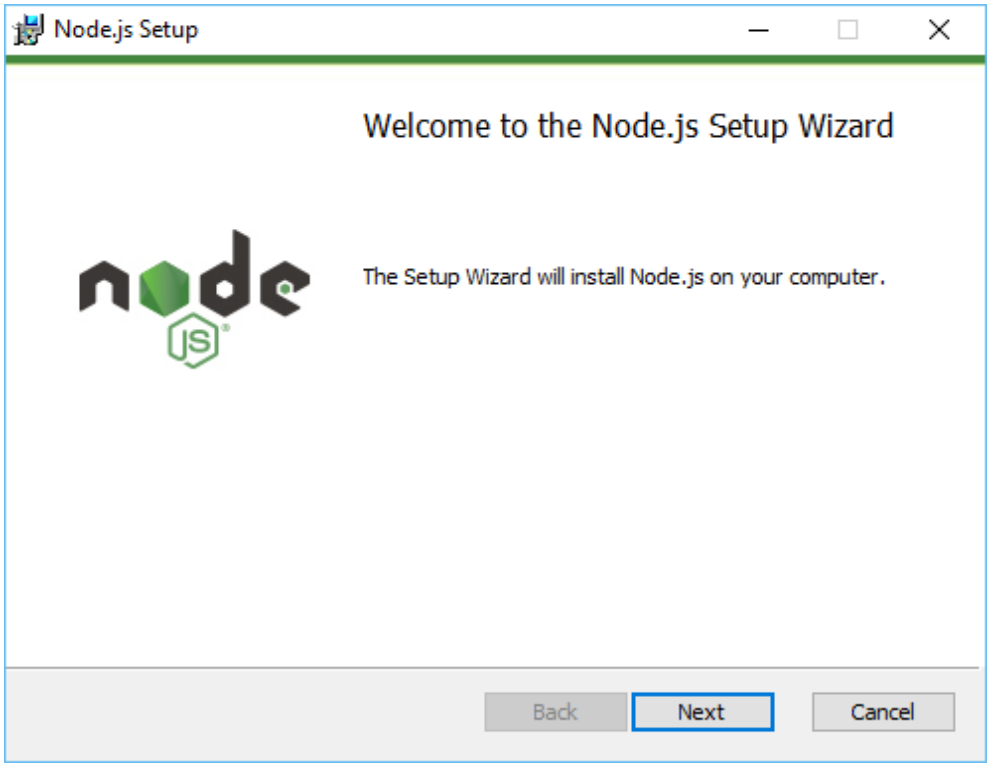
Η εφαρμογή του μαθητή είναι το τελευταίο κομμάτι της εργασίας, το οποίο ολοκληρώνει την εργασία. Αυτό το κομμάτι δέχεται όλες τις πληροφορίες, έτσι ώστε να παράγει ένα πολύ καλό αποτέλεσμα για τους χρήστες του. Αυτή τη φορά χρησιμοποιήθηκαν εντελώς διαφορετικές γλώσσες προγραμματισμού για τη συγγραφή του κώδικα, όπως επίσης και εντελώς διαφορετικές τεχνολογίες. Αυτό έγινε επειδή η εφαρμογή πρέπει να είναι συμβατή με όλα τα λειτουργικά συστήματα που υπάρχουν για έξυπνα κινητά τηλέφωνα, όπως επίσης και να είναι και διαθέσιμη και στο διαδίκτυο μέσω μιας ιστοσελίδας στο web. Αν γινόταν χρήση των τεχνολογιών και τις γλωσσών προγραμματισμού για κάθε πλατφόρμα ξεχωριστά, όλη η εργασία θα ήταν αρκετά χρονοβόρα και θα υπήρχαν αρκετά προβλήματα, όπως επίσης θα χρειαζόταν και πάρα πολύ καλές γνώσεις όλων των τεχνολογιών της κάθε πλατφόρμας ξεχωριστά, το οποίο είναι εφικτό αλλά αρκετά δύσκολο.

Η συγκεκριμένη εφαρμογή είναι μια υβριδική εφαρμογή (hybrid application). Για την ανάπτυξη της έγινε χρήση τεχνολογιών web, οι οποίες είναι οι γλώσσες προγραμματισμού HTML5, CSS3, JavaScript, jQuery, jQuery Mobile και Ajax. Επίσης αναγκαία ήταν η χρήση του framework Apache Cordova.

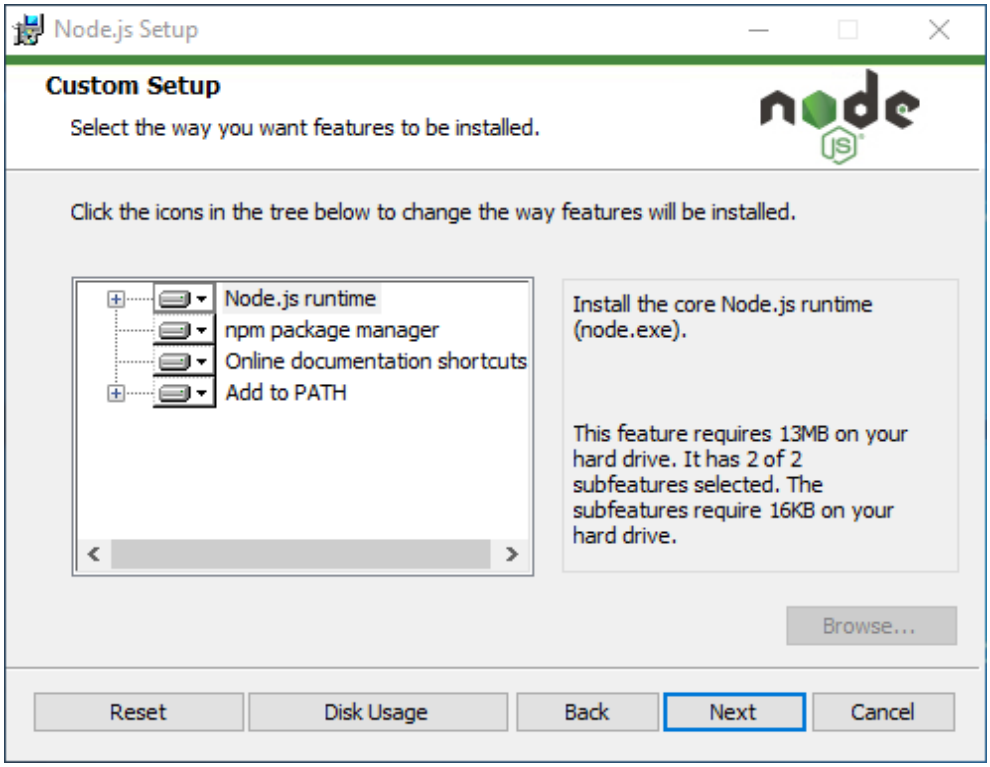
4.3.2 Netbeans και Apache Cordova

Για την υλοποίηση της συγκεκριμένης εφαρμογής χρειάστηκε να γίνει εγκατάσταση στον υπολογιστή κάποιων προγραμμάτων και μια σειρά από framework. Τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της εργασίας είναι το Netbeans, το οποίο είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης ιδανικό για την ανάπτυξη της συγκεκριμένης εργασίας αλλά και γενικότερα για εργασίες βασισμένες πάνω στις τεχνολογίες διαδικτύου. Έχει πάρα πολλές δυνατότητες και μπορεί να χρησιμοποιηθεί για την ανάπτυξη εργασιών σε πολλές διαφορετικές γλώσσες προγραμματισμού παρέχοντας ταυτόχρονα βοήθεια στους προγραμματιστές. Είναι παρόμοιο με το Eclipse που είχε χρησιμοποιηθεί στο προηγούμενο κομμάτι της εργασίας αλλά έχουν κάποιες μικρές διαφορές.

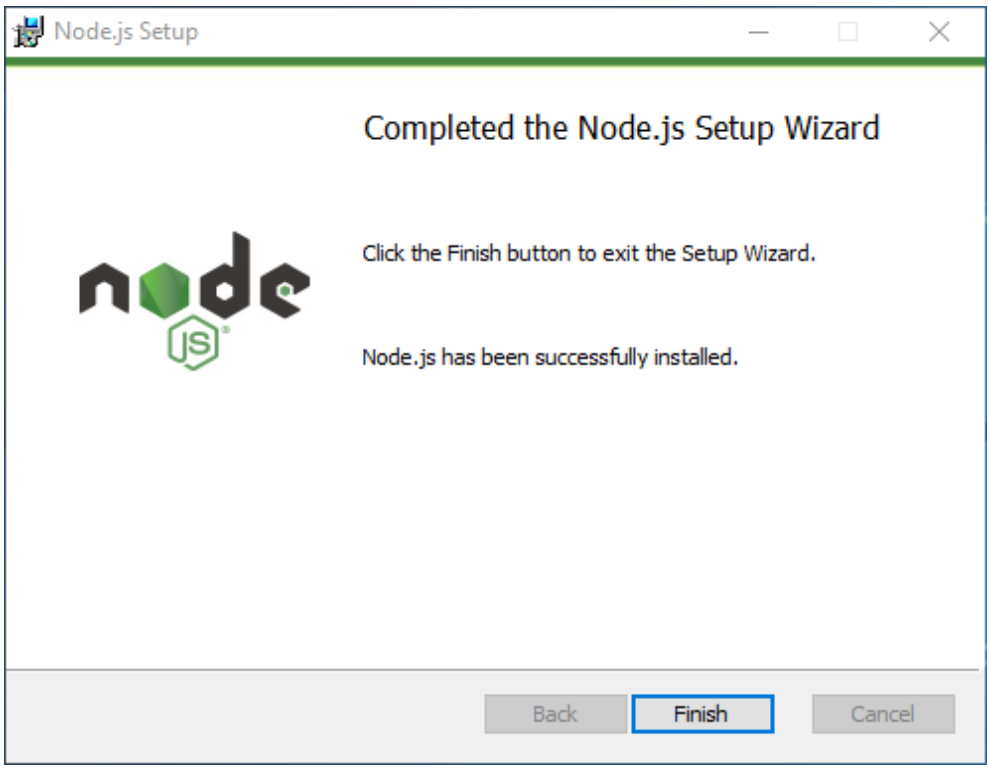
Τα framework τα οποία είναι απαραίτητα για την υλοποίηση της συγκεκριμένης εργασίας είναι το Node.js, το Git και το Apache Cordova. Αφού γίνει εγκατάσταση σωστά σε όλα αυτά τα εργαλεία, είναι όλα έτοιμα για την ανάπτυξη υβριδικών εφαρμογών.



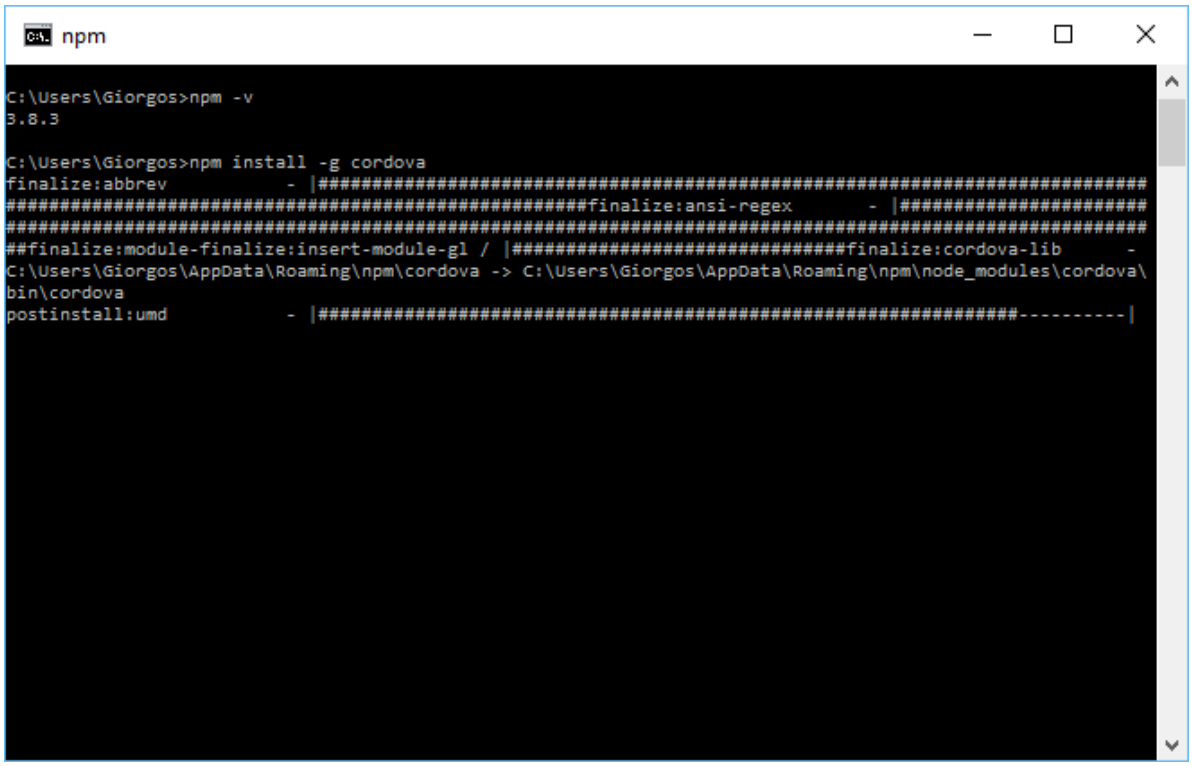
Εικόνα 53: Node.js – Εγκατάσταση 1



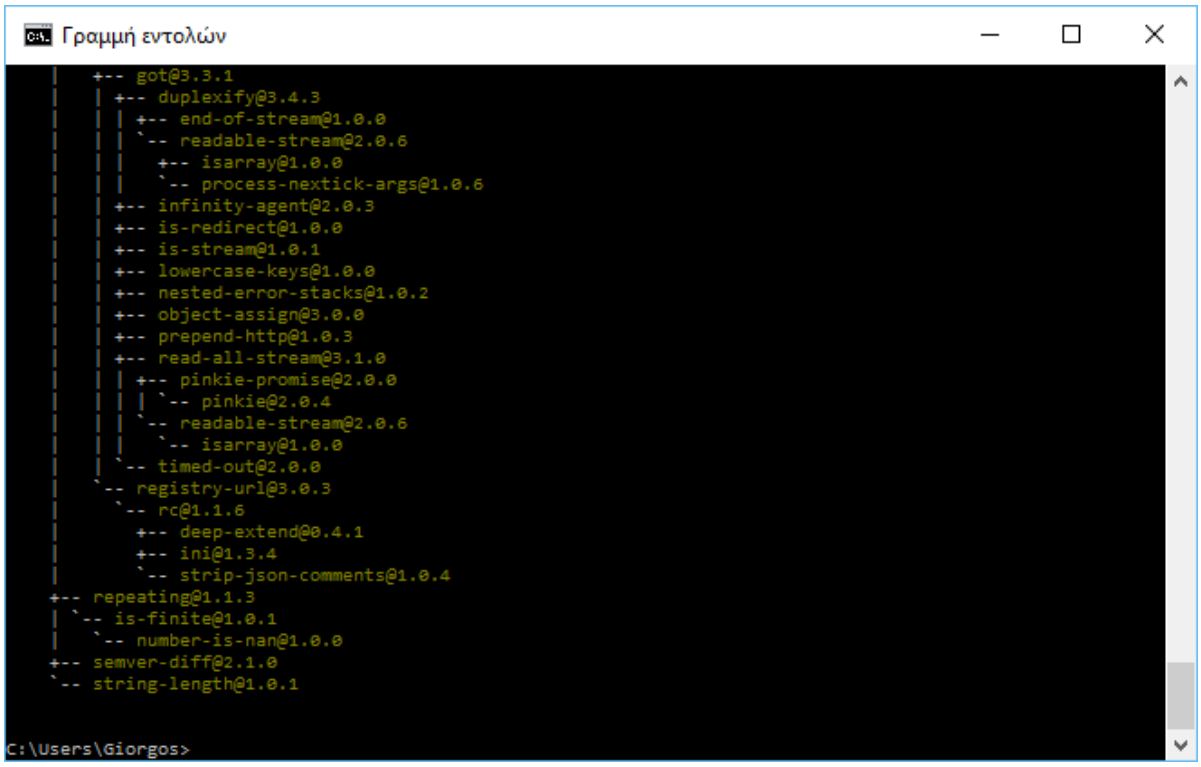
Εικόνα 54: Node.js – Εγκατάσταση 2



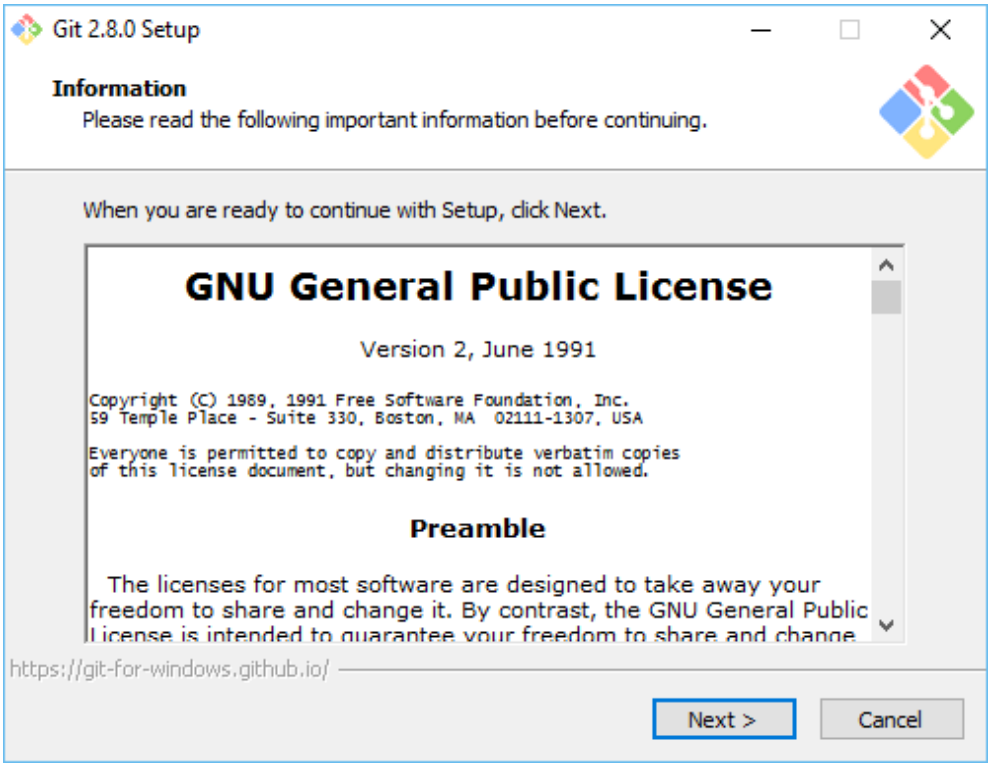
Εικόνα 55: Node.js – Εγκατάσταση 3



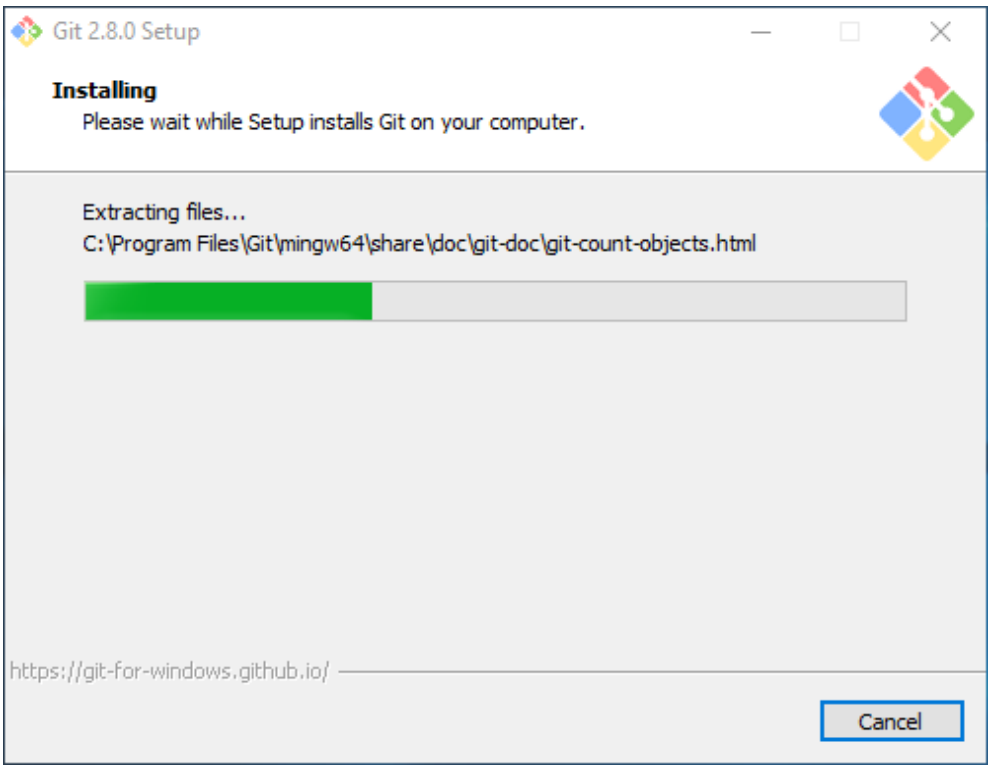
Εικόνα 56: Apache Cordova – Εγκατάσταση 1



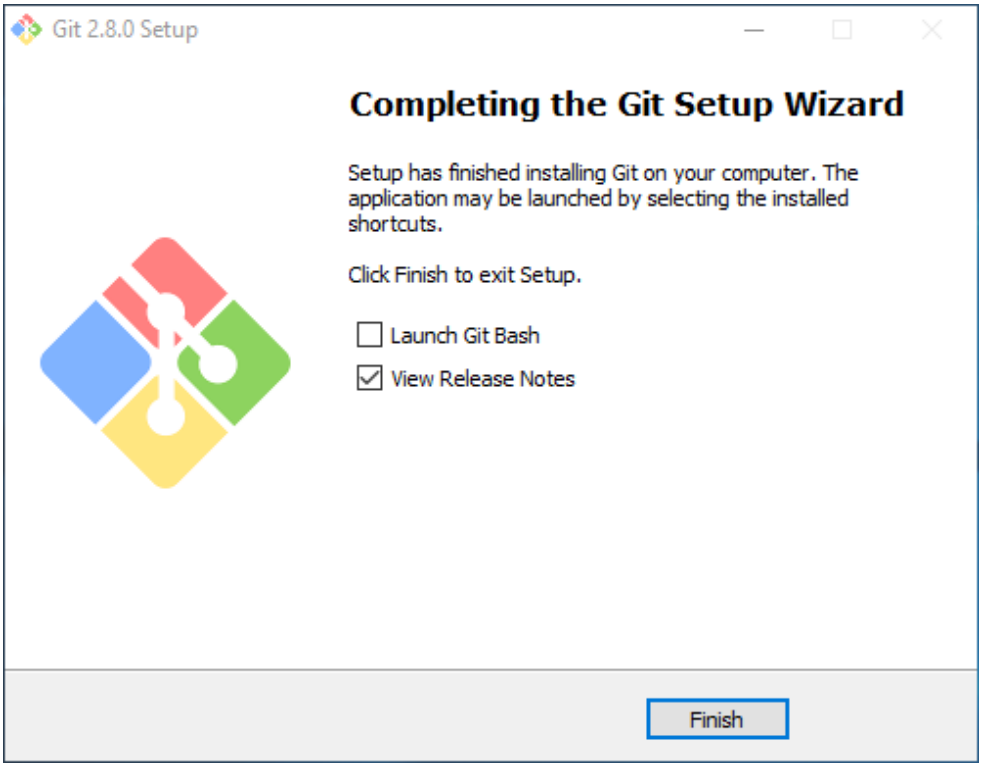
Εικόνα 57: Apache Cordova – Εγκατάσταση 2



Εικόνα 58: Git – Εγκατάσταση 1



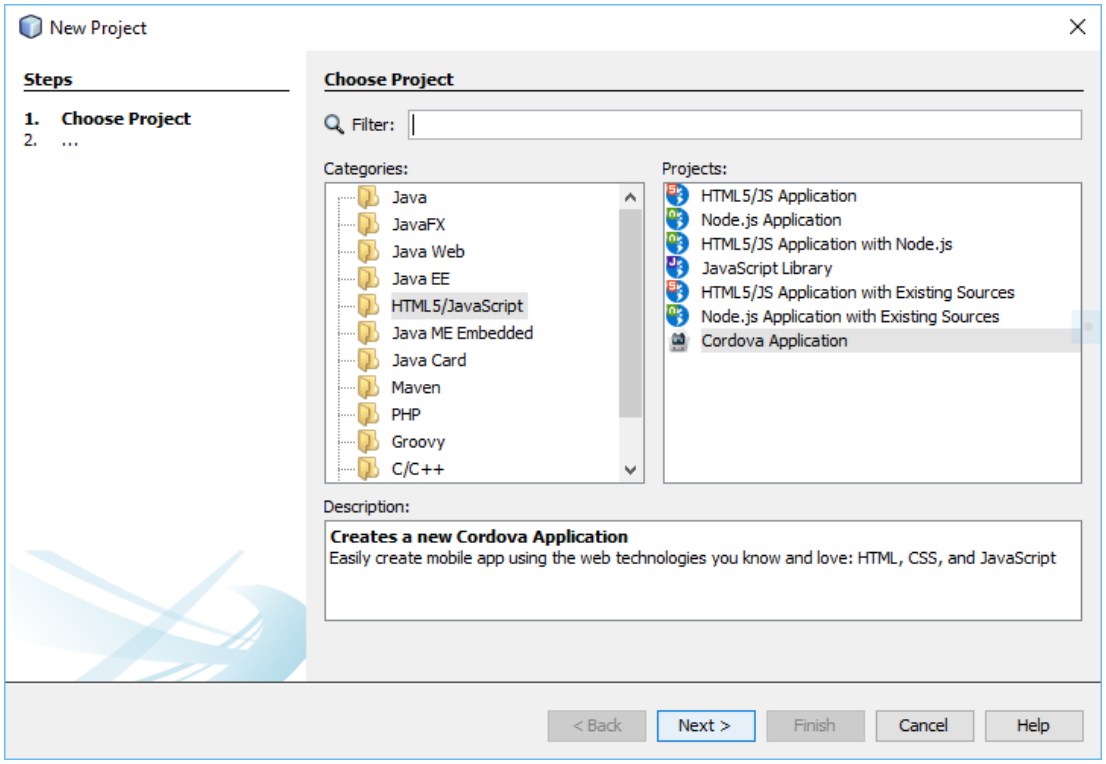
Εικόνα 59: Git – Εγκατάσταση 2



Εικόνα 60: Git – Εγκατάσταση 3

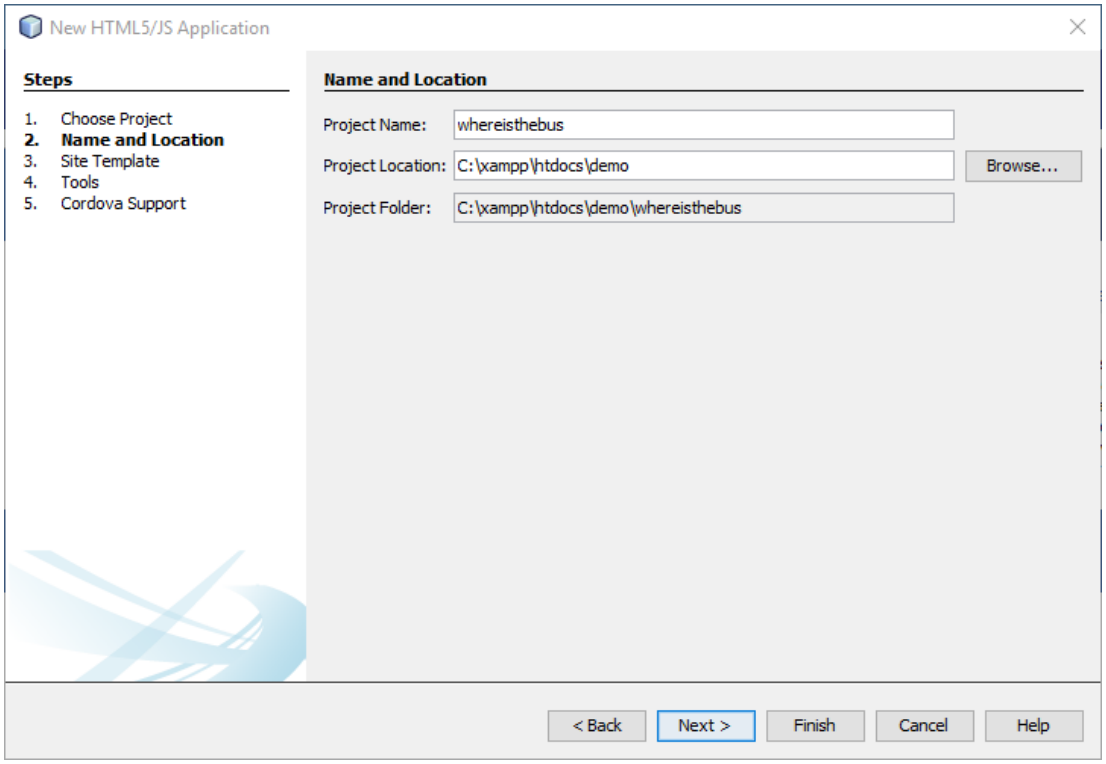
4.3.3 Δημιουργία εφαρμογής μαθητή

Μόλις ολοκληρωθεί η εγκατάσταση όλων των εργαλείων που είναι απαραίτητα για την ανάπτυξη υβριδικών εφαρμογών, η επιλογή για δημιουργία Cordova Applications από το Netbeans γίνεται διαθέσιμη.



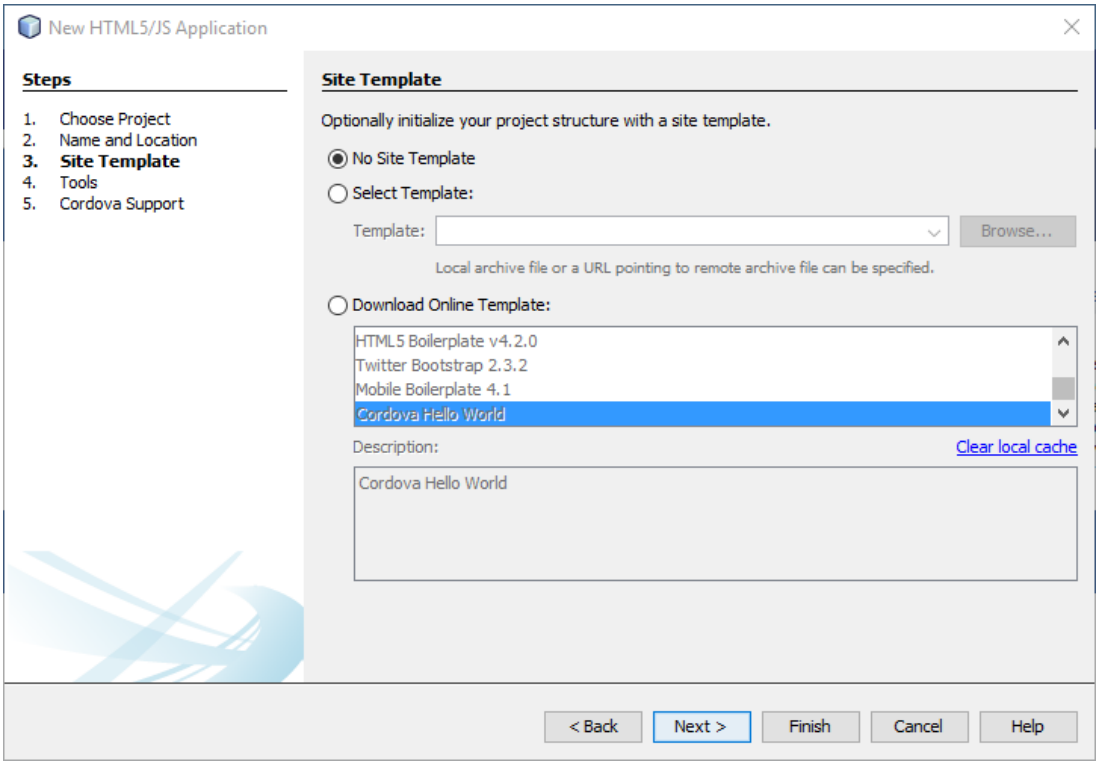
Εικόνα 61: Netbeans – Δημιουργία Cordova Application

Γίνεται επιλογή του ονόματος και της θέσης της νέας εφαρμογής.

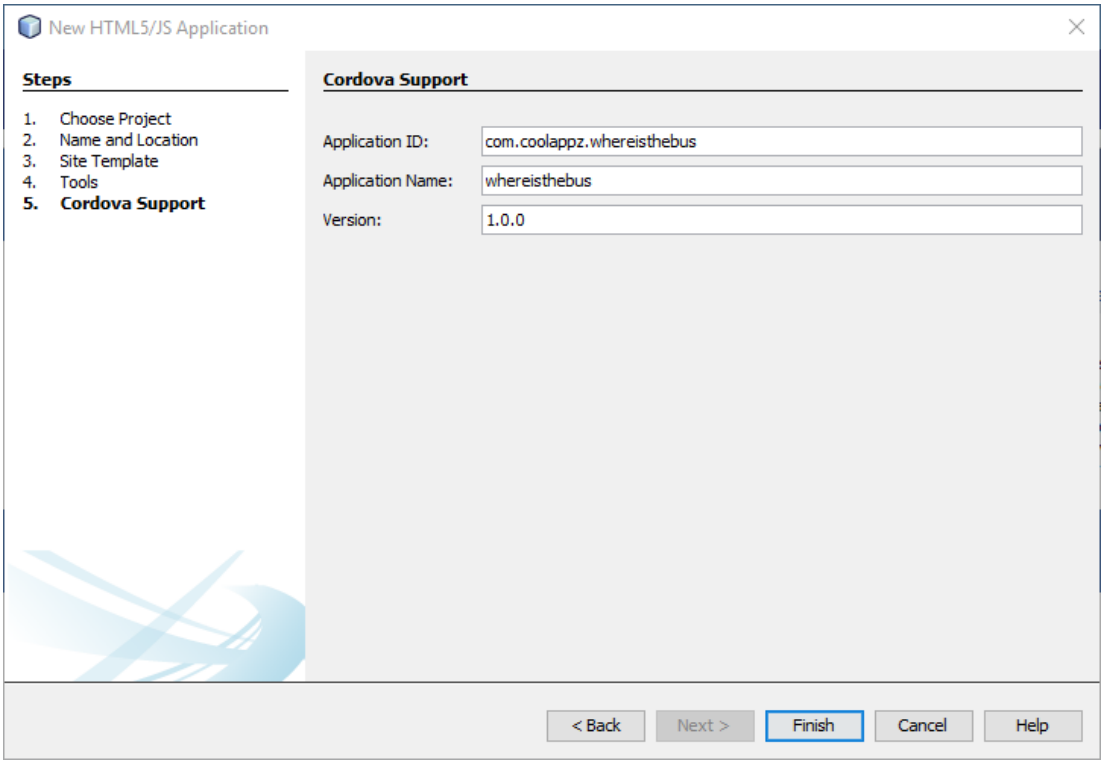


Εικόνα 62: Netbeans – Ιδιότητες εφαρμογής μαθητή

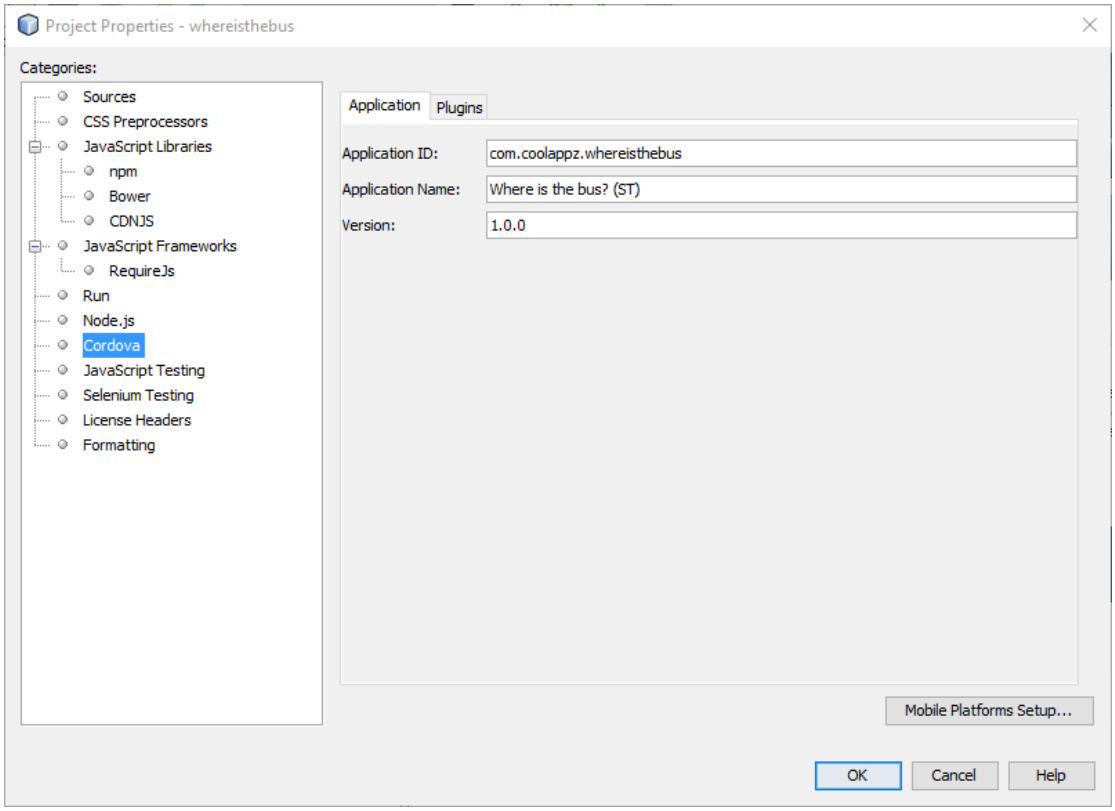
Στη συγκεκριμένη εργασία δε γίνεται χρήση κάποιου έτοιμου template.



Εικόνα 63: Netbeans – Template εφαρμογής μαθητή

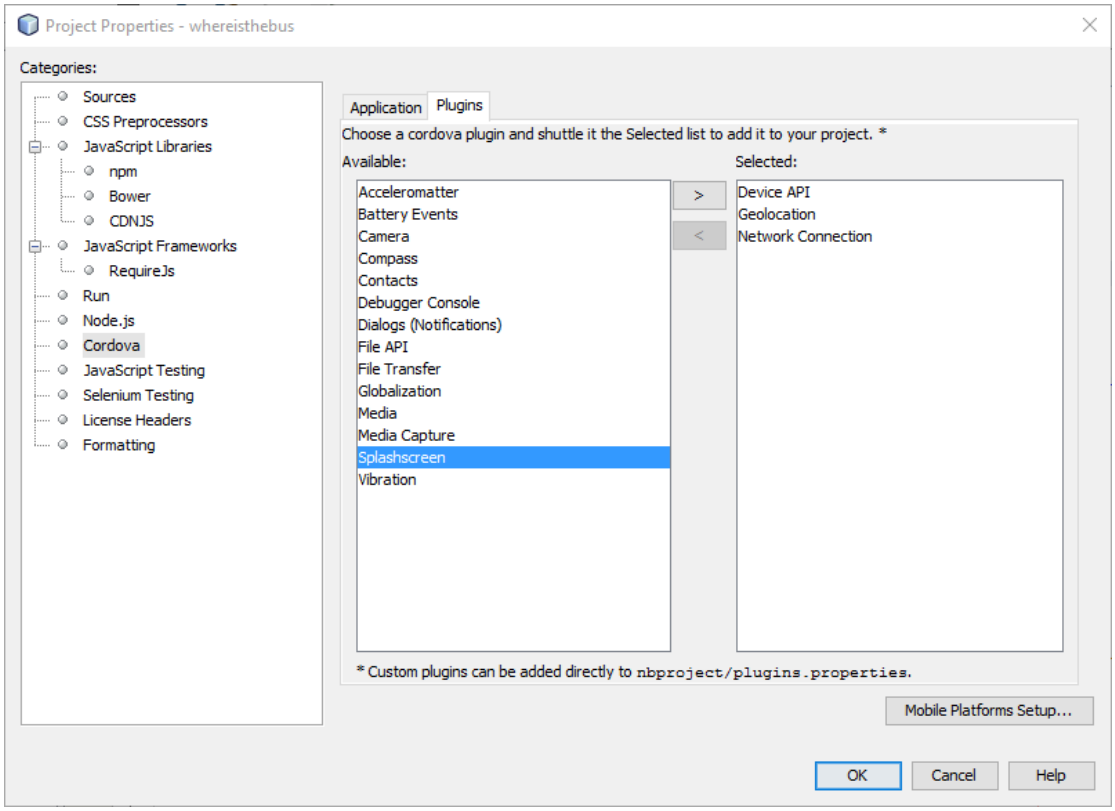


Εικόνα 64: Netbeans – Ολοκλήρωση δημιουργίας εφαρμογής μαθητή



Εικόνα 65: Netbeans – Αλλαγή ονόματος εφαρμογής μαθητή

Βασική ρύθμιση της εφαρμογής είναι η επιλογή των plugins, τα οποία θα μπορεί να χρησιμοποιήσει η υβριδική εφαρμογή. Στη συγκεκριμένη περίπτωση χρειάζονται το plugin Device API, το plugin Geolocation για τη χρήση του GPS, το plugin Network Connection για τη σύνεση της εφαρμογής με το διαδίκτυο.

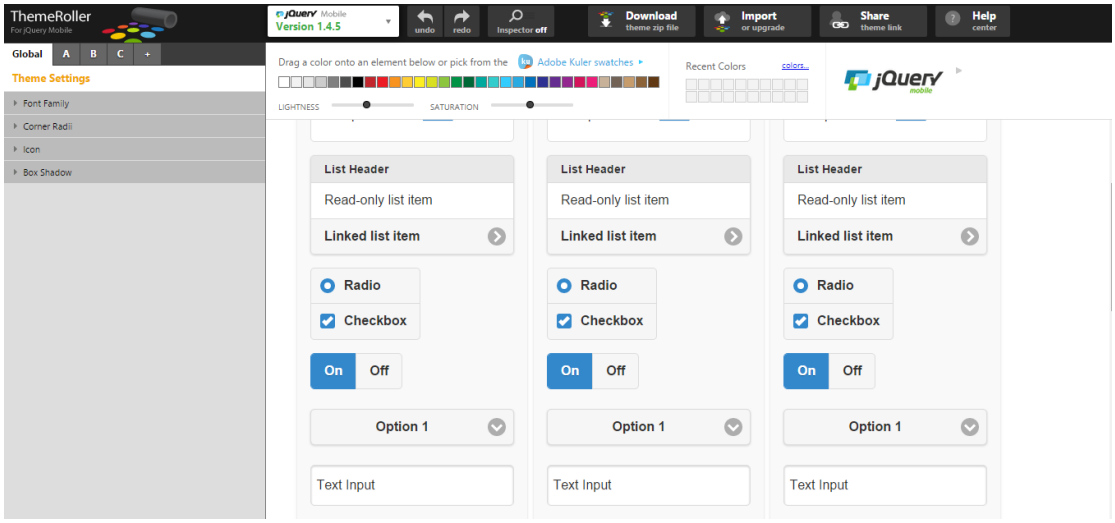


Εικόνα 66: Netbeans – Επιλογή Plugins

4.3.4 User Interface εφαρμογής μαθητή

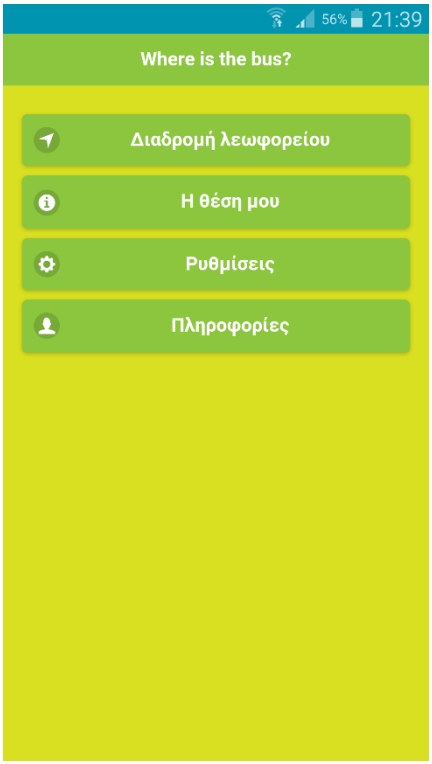
Η διαδικασία υλοποίησης της εφαρμογής που ακολουθήθηκε για αυτή την εφαρμογή είναι η ίδια με τη διαδικασία της εφαρμογής του οδηγού του σχολικού λεωφορείου. Αρχικά και εδώ σχεδιάστηκε όλο το γραφικό περιβάλλον στο χαρτί και τη συνέχεια υλοποιήθηκε. Το περιβάλλον χρήστη είναι υλοποιημένο με τη βιβλιοθήκη της γλώσσας προγραμματισμού JavaScript την jQuery Mobile.

Χρησιμοποιήθηκε σε όλη την εφαρμογή το πράσινο και το κίτρινο, το οποίο είναι αρκετά ευχάριστο για τους μαθητές και παραπέμπει στα χρώματα ενός κλασικού κίτρινου λεωφορείου. Το framework δίνει τη δυνατότητα στους προγραμματιστές που το χρησιμοποιούν να δημιουργούν θέματα με διαφορετικά χρώματα από τα ήδη προεπιλεγμένα μέσω μιας διαδικτυακής υπηρεσίας. Η επιλογή χρωμάτων και σκιάσεων έγινε με τη χρήση της υπηρεσίας https://themeroller.jquerymobile.com/. Τέλος εκτός από τα χρώματα που επιλέχθηκαν για την εφαρμογή αυτή, επιλέχθηκε και το όνομα «**Where is the bus? (ST)**».

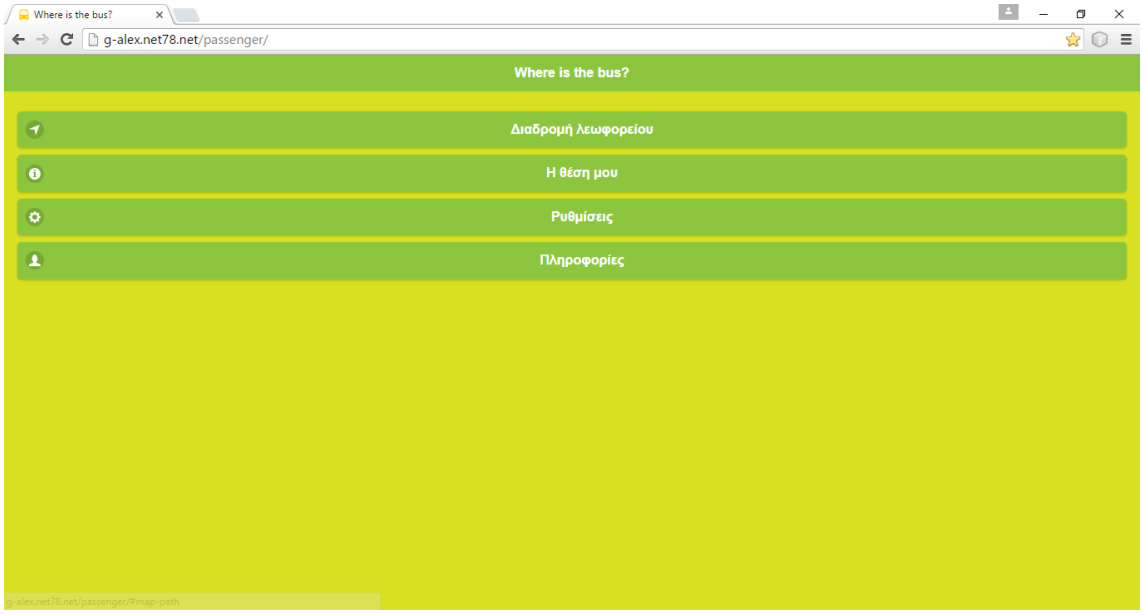


Εικόνα 67: Υπηρεσία ThemeRoller

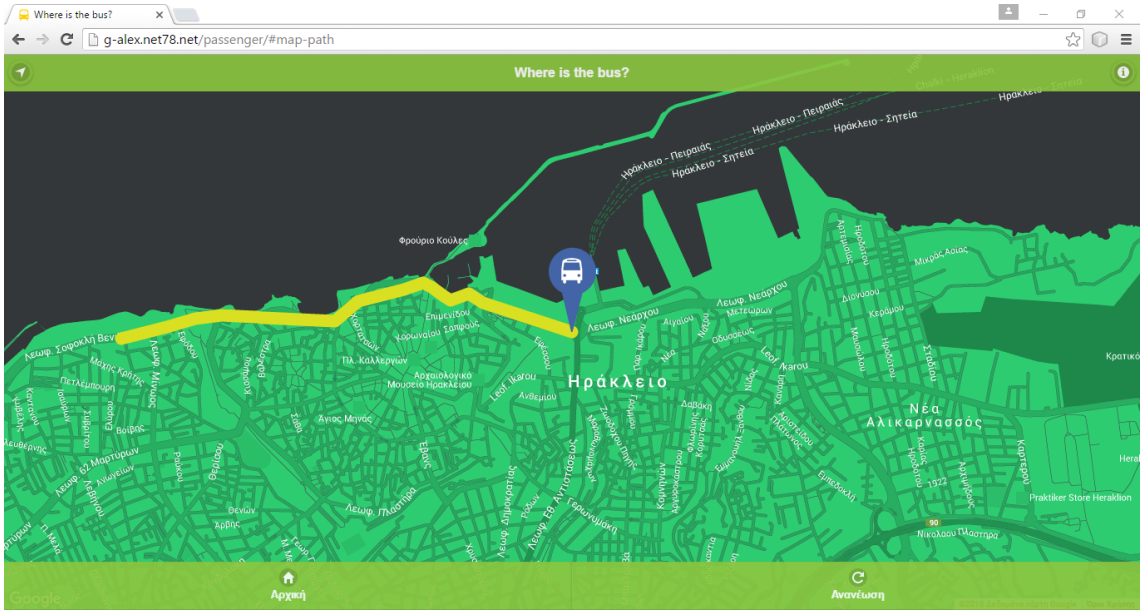
Για την αρχική οθόνη, την οθόνη που εμφανίζεται μόλις ο χρήστης τρέχει την εφαρμογή, χρησιμοποιήθηκαν τα γραφικά στοιχεία Buttons, με τα οποία ο χρήστης έχει τη δυνατότητα να επιλέξει μεταξύ διάφορων επιλογών. Παρακάτω παρουσιάζονται μια προς μια η κάθε επιλογή της εφαρμογής.



Εικόνα 68: Εφαρμογή Μαθητή – Αρχική οθόνη στο κινητό

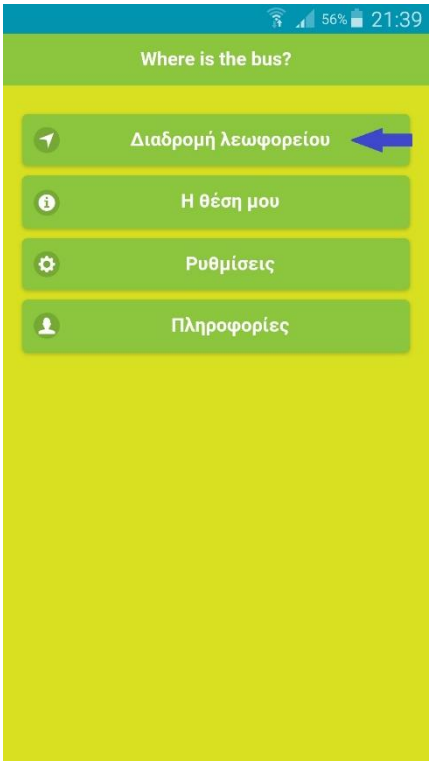


Εικόνα 69: Εφαρμογή Μαθητή – Αρχική οθόνη στον web browser



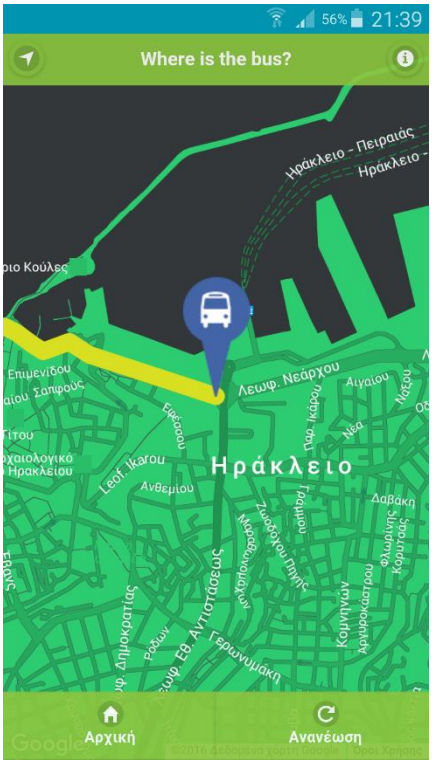
Εικόνα 70: Εφαρμογή Μαθητή – Διαδρομή λεωφορείου στον web browser

4.3.4.1 Διαδρομή λεωφορείου



Εικόνα 71: Εφαρμογή Μαθητή – Διαδρομή λεωφορείου

Η επιλογή «**Διαδρομή λεωφορείο**» είναι η πρώτη επιλογή του μενού και η πιο σημαντική, γιατί εκεί εκτελούνται όλες οι ενέργειες και οι κύριες λειτουργίες όλης της εφαρμογής. Πατώντας την πρώτη επιλογή εμφανίζεται ένας χάρτης με την τοποθεσία του σχολικού λεωφορείου, όπως επίσης και οι πρόσθετες πληροφορίες για το σχολικό λεωφορείο και τον οδηγό. Έγινε η προσθήκη των συγκεκριμένων πληροφοριών, έτσι ώστε να είναι η εφαρμογή πιο εύχρηστη και δελεαστική προς χρήση για το μαθητή.



Εικόνα 72: Εφαρμογή Μαθητή – Χάρτης διαδρομής

Στο βασικό μενού, το οποίο βρίσκεται στο πάνω μέρος της οθόνης, εμφανίζονται δύο κουμπιά. Πατώντας τα, εμφανίζονται κάποια παράθυρα τα οποία περιέχουν σημαντικές πληροφορίες. Πατώντας το δεξί κουμπί εμφανίζεται ένα πλαίσιο με πληροφορίες για τη διαδρομή που έχει διανύσει το λεωφορείο, όπως και τη μέση ταχύτητα του λεωφορείο. Πατώντας το αριστερό κουμπί εμφανίζεται ένα πλαίσιο με την οδό που βρίσκεται το λεωφορείο τη συγκεκριμένη χρονική στιγμή, την οδό που βρίσκεται ο χρήστης τη συγκεκριμένη χρονική στιγμή, την απόσταση μεταξύ λεωφορείου και μαθητή και το χρόνο που θα κάνει το λεωφορείο να φτάσει στην τοποθεσία του μαθητή.

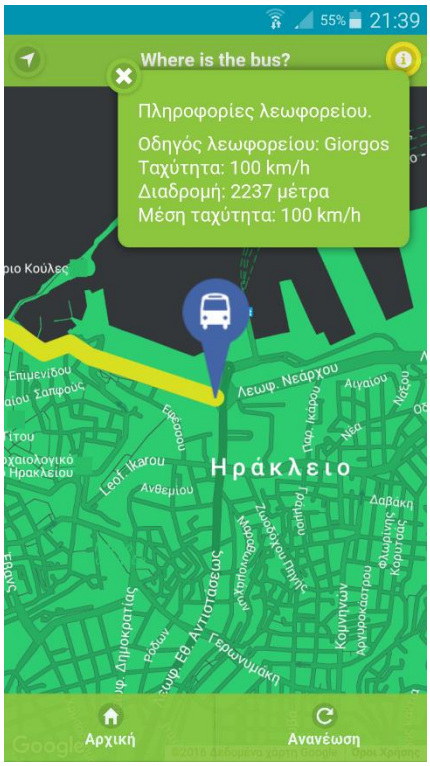
Για τον υπολογισμό της απόστασης και του χρόνου άφιξης του λεωφορείου προς το μαθητή, χρησιμοποιείται η τοποθεσία και η ταχύτητα από το κινητό τηλέφωνο το μαθητή. Ο τύπος είναι αρκετά απλός για τον υπολογισμό του χρόνου άφιξης:

$$\boldsymbol{t\ sec = \frac{x\ km}{u(t)\ km/h}}.$$

Γίνεται χρήση των συντεταγμένων του λεωφορείο και της ταχύτητας και με τη χρήση του συγκεκριμένου τύπου υπολογίζεται ο χρόνος άφιξης του. Ο υπολογισμός της διαδρομής γίνεται με τη χρήση μιας βιβλιοθήκης Google Maps API, με την οποία υπολογίζεται η κοντινότερη διαδρομή από δύο σημεία. Η λύση αυτή ίσως να μην είναι τόσο ακριβής, αλλά η απόκλιση στο χρόνο αφίξεις της εφαρμογής με τον πραγματικό χρόνο δεν έχει μεγάλη διαφορά. Σε μια μελλοντική αναβάθμιση της εφαρμογής ίσως μπορούν να χρησιμοποιηθούν προκαθορισμένες διαδρομές του λεωφορείου, πετυχαίνοντας πάρα πολύ μεγάλη ακρίβεια στον υπολογισμό του χρόνου άφιξης.

Η τοποθεσία οδηγού και χρήστη γίνεται με τη χρήση του GPS, δηλαδή των συντεταγμένων τις κάθε εφαρμογής και του Google Maps API. Το όνομα οδηγού αποστέλλεται από το web server, όπως και η ταχύτητα του λεωφορείου. Η απόσταση

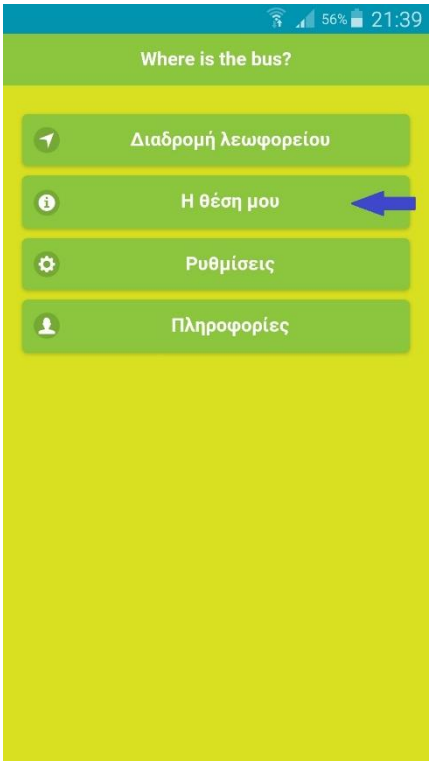
της διαδρομής του λεωφορείου υπολογίζεται με την μέτρηση των ευθυγράμμων τμημάτων που διανύει το σχολικό λεωφορείο και αυτή η ενέργεια γίνεται στην εφαρμογή του μαθητή, έτσι ώστε να μην επιβαρύνεται ο αποστολέας με περιττούς υπολογισμούς και ο web server με περιττά δεδομένα.



Εικόνα 73: Εφαρμογή Μαθητή – Πληροφορίες λεωφορείου

Εικόνα 74: Εφαρμογή Μαθητή – Απόσταση λεωφορείου από χρήστη

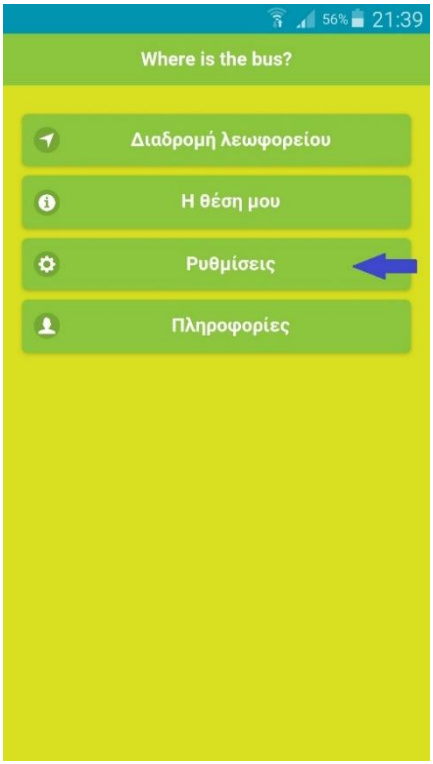
4.3.4.2 Η θέση μου



Εικόνα 75: Εφαρμογή Μαθητή – Η θέση μου

Η επιλογή «**Η θέση μου**» είναι μια βοηθητική προς τον χρήστη λειτουργία, έτσι ώστε να μπορεί να εξακριβώσει την ακριβή του θέση πάνω στο χάρτη. Θεωρήθηκε αρκετά σημαντικό ο χρήστης να μπορεί να βλέπει που βρίσκεται. Μελλοντική εξέλιξη της συγκεκριμένης ενότητας θα ήταν να επιλέγει ο χρήστης τη θέση που βρίσκεται σε περίπτωση που το κινητό του τηλέφωνο δεν διαθέτει GPS ή ακόμα και σε περίπτωση που δεν είναι διαθέσιμή η υπηρεσία.

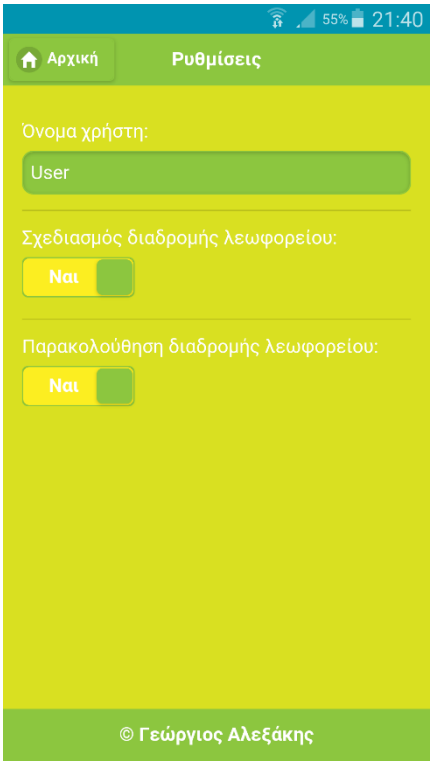
4.3.4.3 Ρυθμίσεις



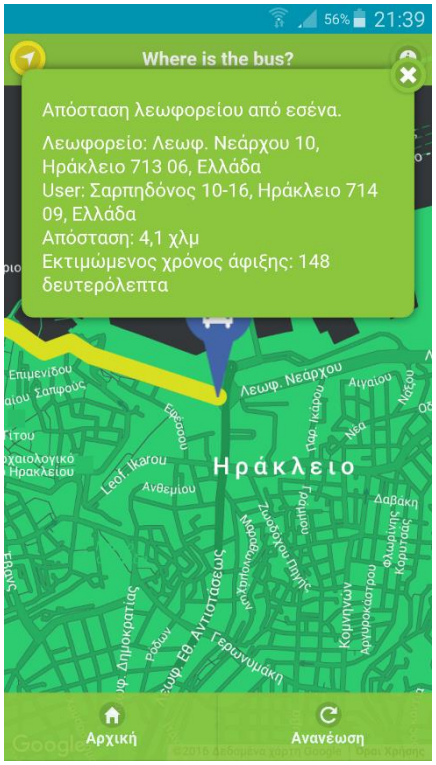
Εικόνα 76: Εφαρμογή Μαθητή – Ρυθμίσεις

Η επιλογή «**Ρυθμίσεις**» παρέχει στον χρήστη τη δυνατότητα να ρυθμίσει την εφαρμογή ανάλογα με τις δικές του προτιμήσεις. Από την επιλογή των ρυθμίσεων μπορεί να αλλαχτεί το όνομα του χρήστη της εφαρμογής, το οποίο είναι «**User**» από προεπιλογή, να γίνει ενεργοποίηση/απενεργοποίηση του σχεδιασμού της διαδρομής του λεωφορείου όπως επίσης και να γίνει ενεργοποίηση/απενεργοποίηση της παρακολούθησης της διαδρομής.

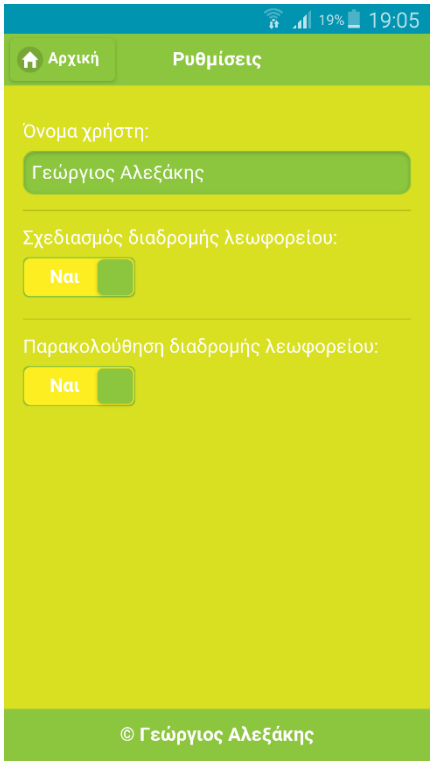
Αλλάζοντας το όνομα χρήστη στις ρυθμίσεις, αλλάζει και το όνομα που θα εμφανίζεται στην απόσταση από το λεωφορείο προς τον χρήστη. Παρακάτω παρουσιάζεται ένα παράδειγμα της συγκεκριμένης λειτουργίας.



Εικόνα 77: Εφαρμογή Μαθητή – Default όνομα χρήστη



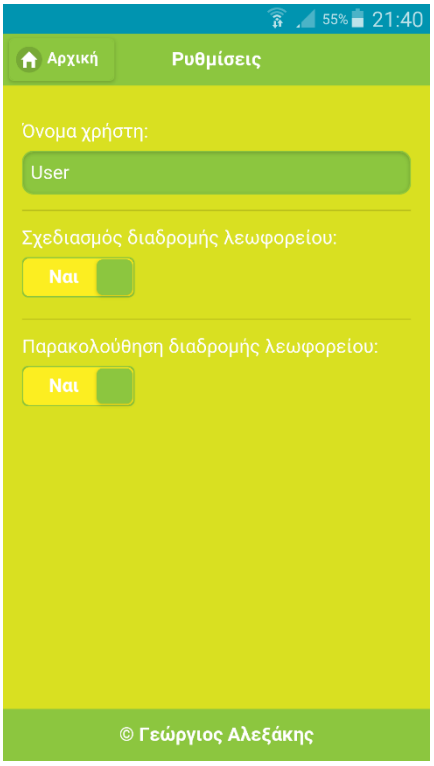
Εικόνα 78: Εφαρμογή Μαθητή – Default όνομα χρήστη



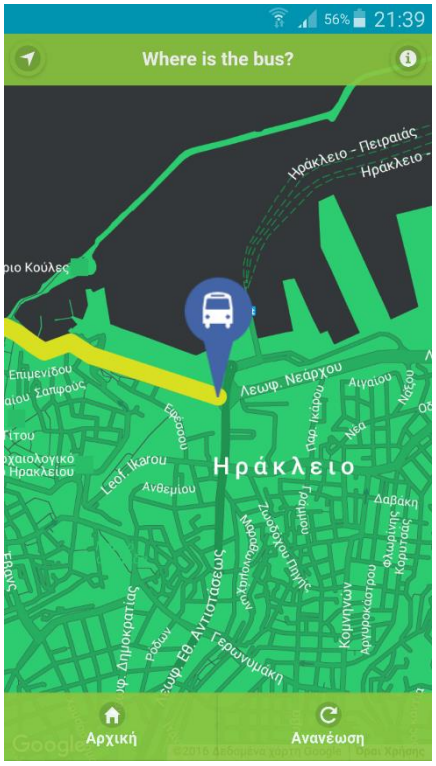
Εικόνα 79: Εφαρμογή Μαθητή – Αλλαγή ονόματος χρήστη

Εικόνα 80: Εφαρμογή Μαθητή – Αλλαγή ονόματος χρήστη

Στις αρχικές ρυθμίσεις, όπως φαίνονται στην παρακάτω εικόνα, ο σχεδιασμός διαδρομής είναι σε κατάσταση «**Ναι**». Παρατηρώντας την παρακάτω εικόνα στο χάρτη πάνω σχεδιάζεται η διαδρομή του λεωφορείου με κίτρινο χρώμα.

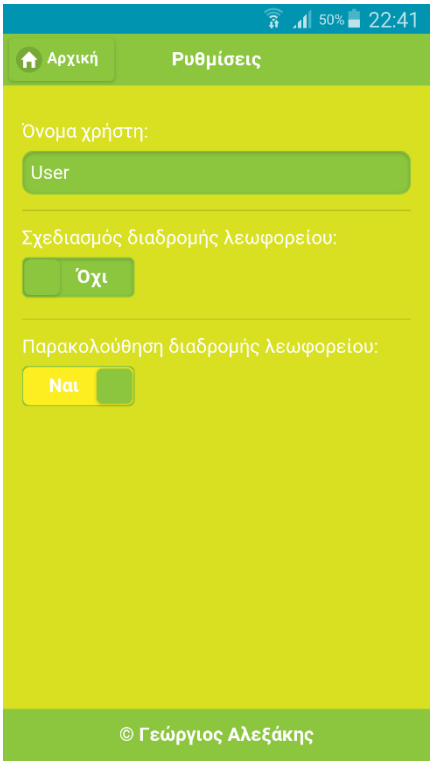


Εικόνα 81: Εφαρμογή Μαθητή – Σχεδιασμός διαδρομής «Ναι»

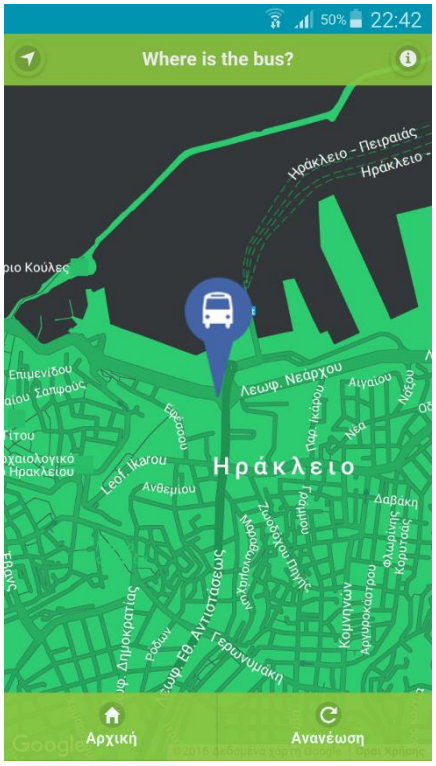


Εικόνα 82: Εφαρμογή Μαθητή – Σχεδιασμός διαδρομής «Ναι»

Στην περίπτωση που ο χρήστης επιλέξει «**Όχι**» για το σχεδιασμό διαδρομής, η διαδρομή του λεωφορείου δεν θα εμφανίζεται πια στο χάρτη, όπως φαίνεται παρακάτω.



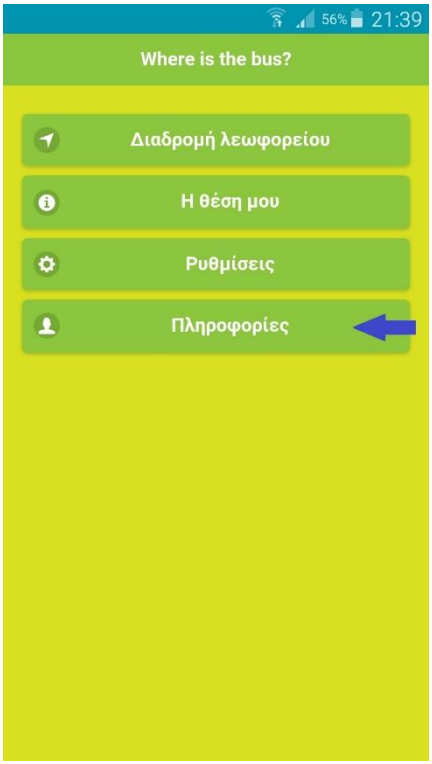
Εικόνα 83: Εφαρμογή Μαθητή – Σχεδιασμός διαδρομής «Όχι»



Εικόνα 84: Εφαρμογή Μαθητή – Σχεδιασμός διαδρομής «Όχι»

Η τελευταία ρύθμιση της εφαρμογής είναι η παρακολούθηση του λεωφορείο σε όλη τη διάρκεια της διαδρομής. Αν η επιλογή είναι στο «**Ναι**», κατά τη διάρκεια της κίνησης του λεωφορείου, ο χάρτης θα κεντράρει την οθόνη με κέντρο το λεωφορείο, έτσι ώστε να ξέρει ο χρήστης που είναι κάθε φορά. Αν επιλεχθεί το «**Όχι**» στην παρακολούθηση του λεωφορείου, η παραπάνω ενέργεια απενεργοποιείται και ο χρήστης έχει τη δυνατότητα να κεντράρει το χάρτη του σε οποιοδήποτε σημείο θέλει.

4.3.4.4 Πληροφορίες



Εικόνα 85: Εφαρμογή Μαθητή – Πληροφορίες

Η επιλογή «**Πληροφορίες**» είναι η τελευταία επιλογή του μενού, η οποία περιέχει γενικές πληροφορίες για την εφαρμογή προς τον χρήστη, όπως επίσης την έκδοση της εφαρμογής και τον προγραμματιστή της εφαρμογής. Ο χρήστης έχει την επιλογή να δει κάποιες γενικές πληροφορίες για τη εφαρμογή, την έκδοση της εφαρμογής και το όνομα του προγραμματιστή της εφαρμογής.

4.3.5 Ανάλυση κώδικα εφαρμογής μαθητή

Το βασικό κομμάτι της εργασίας είναι η επιλογή «**Διαδρομή Λεωφορείου**», η οποία περιέχει όλα τα χαρακτηριστικά της εφαρμογής, αλλά και οι επιπλέον επιλογές είναι σημαντικές για την μελλοντική εξέλιξη της εφαρμογής. Οι επιπλέον επιλογές είναι βοηθητικές και παρέχουν επιπλέον λειτουργίες. Βασικό χαρακτηριστικό της εφαρμογής είναι ο χάρτης Google Map.

Η συγγραφή του κώδικα ξεκινάει με τη συγγραφή του κώδικα HTML με τη χρήση της βιβλιοθήκης jQuery Mobile. Ο κώδικας όλων των επιλογών είναι γραμμένος σε ένα αρχείο HTML, στο οποίο χρησιμοποιήθηκαν αρκετές ιδιότητες της jQuery Mobile. Ο παρακάτω κώδικα είναι ο κώδικας της πρώτης επιλογής της «**Διαδρομή Λεωφορείου**». Χρησιμοποιούνται τα attributes και κάποιες κλάσεις, έτσι ώστε να εμφανιστούν τα κουμπιά και πάνελ και όλες οι πληροφορίες.

```
1. <!-- Map Path -->
2. <div data-dom-cache="false" data-role="page" id="map-path" data-url="map-
   path" data-title="Where is the bus?">
3. <div data-role="header" data-position="fixed" data-fullscreen="true">
```

```
<h1>Where is the bus?</h1>
<a href="#distancePopup" data-rel="popup" data-transition="slidedown" class="ui-btn ui-shadow ui-corner-all ui-icon-navigation ui-btn-icon-notext">Distance</a>
<a href="#infoPopup" data-rel="popup" data-transition="slidedown" class="ui-btn ui-shadow ui-corner-all ui-icon-info ui-btn-icon-notext">Information</a>
</div>
<div role="main" class="ui-content" id="map">
<div data-role="popup" id="distancePopup" class="ui-content">
<a href="#" data-rel="back" class="ui-btn ui-corner-all ui-shadow ui-btn ui-icon-delete ui-btn-icon-notext ui-btn-right">Close</a>
<p>…</p>
<p id="output1"></p>
</div>
<div data-role="popup" id="infoPopup" class="ui-content">
<a href="#" data-rel="back" class="ui-btn ui-corner-all ui-shadow ui-btn ui-icon-delete ui-btn-icon-notext ui-btn-left">Close</a>
<p>…</p>
<p id="output2"></p>
</div>
</div>
<div data-role="footer" data-position="fixed" data-fullscreen="true">
<div data-role="navbar" data-iconpos="top">
<ul>
<li><a href="#index" data-icon="home" class="ui-btn ui-shadow">…</a></li>
<li><a href="#" id="refresh" data-icon="refresh" class="ui-btn ui-shadow">…</a></li>
</ul>
</div>
</div>
</div>
```

Για τη συγγραφή του κώδικα JavaScript έγινε χρήση του Google Maps API με την προσθήκη των βιβλιοθηκών της Google στην εργασία μας. Η ανάκτηση των δεδομένων για την εμφάνιση των πληροφοριών στον χάρτη γίνεται με τη χρήση της τεχνολογίας Ajax. Με την τεχνολογία αυτή επιτυγχάνεται η ανανέωση των δεδομένων στο χάρτη χωρίς να γίνεται ανανέωση ολόκληρου του χάρτη και γενικότερα ολόκληρου του περιβάλλοντος της εφαρμογής. Θα αναλυθεί ο κώδικας εκτενέστερα, έτσι ώστε να γίνει κατανοητός ο τρόπος λειτουργίας του. Ο κώδικας συνοδεύεται από αρκετά σχόλια, είναι αρκετά ευανάγνωστος και έχει χωριστεί σε κομμάτια για την καλύτερη επεξήγηση.

4.3.5.1 Δήλωση global μεταβλητών

Αρχικά γίνεται δήλωση σε όλες τις Global μεταβλητές που θα χρειαστούν στην εφαρμογή αυτή. Δήλωση γίνεται επίσης στο αρχικό όνομα χρήστη, στις συντεταγμένες, στο χάρτη, στους markers του χάρτη του οδηγού και χρήστη, όπως και σε όλες τις άλλες μεταβλητές που θα χρειαστούν στην εφαρμογή. Τέλος δηλώνονται οι αρχικές συντεταγμένες και τα χρώματα που θα έχει ο χάρτης.

```
var userName = 'User';
var enablePathLines = 'on';
var initCounter = 0;
var mapCounter = 0;
var counter = 0;
```

```
6.  var myVar = 0;
7.  var averageSpeed = 0;
8.  var map;
9.  var marker;
10. var clientMarker;
11. var clientMarkerMove = 0;
12. var followBus = 'on';
13. var defaultCoords = [{name: "no coordinates", lat: "37.50148", lng: "25.2048
    42", speed: "no coordinates"}];  // Default coordinates and zoom focusing to
     Greece
14. var defaultCoordinates = new google.maps.LatLng(35.338643, 25.133018);  // D
    efault to Heraklion Crete when no geolocation support
15. var styles = [{"featureType": "water", "elementType": "geometry", "stylers":
     [{"color": "#333739"}]}, {"featureType": "landscape", "elementType": "geome
    try", "stylers": [{"color": "#2ecc71"}]}, {"featureType": "poi", "stylers":
    [{"color": "#2ecc71"}, {"lightness": -
    7}]}, {"featureType": "road.highway", "elementType": "geometry", "stylers":
    [{"color": "#2ecc71"}, {"lightness": -
    28}]}, {"featureType": "road.arterial", "elementType": "geometry", "stylers"
    : [{"color": "#2ecc71"}, {"visibility": "on"}, {"lightness": -
    15}]}, {"featureType": "road.local", "elementType": "geometry", "stylers": [
    {"color": "#2ecc71"}, {"lightness": -
    18}]}, {"elementType": "labels.text.fill", "stylers": [{"color": "#ffffff"}]
    }, {"elementType": "labels.text.stroke", "stylers": [{"visibility": "off"}]}
    , {"featureType": "transit", "elementType": "geometry", "stylers": [{"color"
    : "#2ecc71"}, {"lightness": -
    34}]}, {"featureType": "administrative", "elementType": "geometry", "stylers
    ": [{"visibility": "on"}, {"color": "#333739"}, {"weight": 0.8}]}, {"feature
    Type": "poi.park", "stylers": [{"color": "#2ecc71"}]}, {"featureType": "road
    ", "elementType": "geometry.stroke", "stylers": [{"color": "#333739"}, {"wei
    ght": 0.3}, {"lightness": 10}]}];
```

4.3.5.2 Αρχικοποίηση και ανανέωση χάρτη και ρυθμίσεων

Το deviceready event είναι ένα πάρα πολύ σημαντικό, το οποίο τρέχει όταν η εφαρμογή έχει φορτωθεί πλήρως. Κάθε υβριδική εφαρμογή πρέπει να χρησιμοποιεί το συγκεκριμένο event, γιατί από ότι έχει αναφερθεί παραπάνω, μια υβριδική εφαρμογή αποτελείται από δύο είδη κώδικα: τον native κώδικα για τα plugins και τον JavaScript κώδικα της εφαρμογής. Αυτό έχει ως αποτέλεσμα καθώς φορτώνεται η εφαρμογή, αν φορτωθεί πρώτα ο κώδικας JavaScript και χρειαστεί να καλέσει μια συνάρτηση του Cordova, για τη χρήση κάποιου native plugin, να υπάρξει σφάλμα στην εφαρμογή.

Στις γραμμές 1 έως 4 πυροδοτείτε το deviceready event. Οι γραμμές 6 έως 31 περιέχουν τα event jQuery, τα οποία δίνουν λειτουργικότητα στις ρυθμίσεις τις εφαρμογής. Οι γραμμές 64 έως 74 είναι υπεύθυνες για τη λειτουργία του κουμπιού ανανέωσης.

```
1.  $(document).ready(function () {
2.      document.addEventListener("deviceready", onDeviceReady, false);
3.      //onDeviceReady();
4.  });
5.
6.  function onDeviceReady() {
7.      console.log("Ready");
8.      // Resize page if needed
9.      $(window).bind('pageshow resize orientationchange', function (e) {
10.         maxHeight();
11.     });
```

```
12.
13.     // Settings and refresh button on map-path page
14.     $("#refresh").click(function () {
15.         refreshMap();
16.     });
17.
18.     $("select#buspath").change(function () {
19.         enablePathLines = $(this).val();
20.     });
21.
22.     $("select#busfollow").change(function () {
23.         followBus = $(this).val();
24.     });
25.
26.     $("#name").change(function () {
27.         userName = $(this).val();
28.     });
29.
30.     maxHeight();
31. }
32.
33. function maxHeight() {
34.     var w = $(window).height();
35.     var cs = $('div[data-role="ui-content"]');
36.     for (var i = 0, max = cs.length; i < max; i++) {
37.         var c = $(cs[i]);
38.         var h = $($('div[data-role="header"]')[i]).outerHeight(true);
39.         var f = $($('div[data-role="footer"]')[i]).outerHeight(true);
40.         var c_h = c.height();
41.         var c_oh = c.outerHeight(true);
42.         var c_new = w - h - f - c_oh + c_h;
43.         var total = h + f + c_oh;
44.         if (c_h < c.get(0).scrollHeight) {
45.             c.height(c.get(0).scrollHeight);
46.         } else {
47.             c.height(c_new);
48.         }
49.     }
50. }
51.
52. // Triggered on the "to" page, after the transition animation completes - ma
    p-path page
53. $(document).on("pageshow", "#map-path", function () {
54.     initCounter = 0;
55.     mapCounter = 0;
56.     counter = 0;
57.     myVar = 0;
58.     averageSpeed = 0;
59.     clientMarkerMove = 0;
60.     defaultCoordinates = new google.maps.LatLng(35.338643, 25.133018);
61.     initMap();
62. });
63.
64. // Refresh map-page
65. function refreshMap() {
66.     initCounter = 0;
67.     mapCounter = 0;
68.     counter = 0;
69.     myVar = 0;
70.     averageSpeed = 0;
71.     clientMarkerMove = 0;
72.     defaultCoordinates = new google.maps.LatLng(35.338643, 25.133018);
73.     initMap();
74. }
```

4.3.5.3 Αρχικοποίηση δεδομένων του χάρτη της Διαδρομής λεωφορείου

Η συνάρτηση **initMap()** είναι υπεύθυνη για την αρχικοποίηση όλων των μεταβλητών της επιλογής «**Διαδρομή λεωφορείου**». Χρησιμοποίει την τεχνολογία Ajax, έτσι ώστε να κάνει ανάκτηση όλων των δεδομένων που βρίσκονται στη βάση δεδομένων την πρώτη φορά που γίνεται εμφάνιση του χάρτη.

Πιο συγκεκριμένα στις γραμμές 7 έως 16 γίνεται είσοδος των δεδομένων με μορφοποίηση JSON, με μια απόκριση HTTP με τη μέθοδο POST και στη συνέχεια τρέχει μια απλή εντολή, έτσι ώστε να καθαριστούν τα δεδομένα από σκουπίδια, τα οποία τοποθετούνται αυτόματα από την εταιρεία παροχής του web server και τέλος γίνεται μετατροπή του JSON αρχείου σε Object, έτσι ώστε να γίνει χρήση τω πληροφοριών που λήφθηκαν (γραμμές 14 - 15).

Υπάρχει περίπτωση η απόκριση του web server να έχει δεδομένα αλλά υπάρχει και περίπτωση η απόκριση να μην περιέχει καθόλου δεδομένα, όταν ο χρήστης έχει διαγράψει τις παλιές διαδρομές. Σε κάθε περίπτωση γίνονται διαφορετικές ενέργειες, έτσι ώστε να επιτυγχάνεται το βέλτιστο αποτέλεσμα. Στην πρώτη περίπτωση, δηλαδή σε αύτη που η απόκριση περιέχει δεδομένα, εκτελείται η συνάρτηση **initData()** με τα δεδομένα που παρέλαβε, με εστίαση στο χάρτη ίση με 15 (γραμμές 17 - 19). Στην περίπτωση που η απόκριση δεν περιέχει καθόλου δεδομένα εμφανίζεται το μήνυμα: «**Δεν υπάρχουν συντεταγμένες στη βάση δεδομένων. Περιμένετε το λεωφορείο να ξεκινήσει.**» στα δύο πάνελ που υπάρχουν και στη συνέχεια εκτελείται η συνάρτηση **initData()** με τις default συντεταγμένες, οι οποίες κεντράρουν στην Ελλάδα με εστίαση ίση με 7 (γραμμές 20 - 26).

```
1.  // Initialize map-page data
2.  function initMap() {
3.      var xmlhttp = new XMLHttpRequest();
4.      var url = "http://g-alex.net78.net/gps/data.php";
5.      xmlhttp.onreadystatechange = function () {
6.          if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
7.              var response = xmlhttp.responseText;
8.              var i = 0;
9.
10.             // Clean JSON from rubbish
11.             while (response[i] != ']') {
12.                 i++;
13.             }
14.             var dbObject = response.substring(0, i + 1);
15.             var dbCoordinates = JSON.parse(dbObject);
16.
17.             if (dbCoordinates.length != 0) {
18.                 counter = dbCoordinates.length;
19.                 initData(dbCoordinates, 15);  // Bus coordinates with map zo
    om 15
20.             } else {
21.                 var outputDiv1 = document.getElementById('output1');
22.                 outputDiv1.innerHTML = 'message';
23.                 var outputDiv2 = document.getElementById('output2');
24.                 outputDiv2.innerHTML = 'message';
25.                 initData(defaultCoords, 7);  // Default location if there ar
    e no coordinates in db with map zoom 7
26.             }
27.         }
28.     };
29.     xmlhttp.open("POST", url, true);
30.     xmlhttp.send();
```

```
31. }
```

4.3.5.4 Δημιουργία του χάρτη και του marker

Η συνάρτηση **initData()** δέχεται στην είσοδο ένα Object με όλα τα δεδομένα που έχει για τη διαδρομή του λεωφορείου και την εστίαση που θα έχει ο χάρτης ανάλογα με τα δεδομένα που λήφθηκαν ή όχι. Δημιουργεί το χάρτη με τις συντεταγμένες και την εστίαση της εισόδου, όπως επίσης και το design (γραμμές 3 - 16).

Στη συνέχεια γίνεται έλεγχος αν τα δεδομένα είναι τα προκαθορισμένα, δηλαδή δεδομένα που δεν έχουν ληφθεί από το web server, ή δεδομένα από το web server. Στην περίπτωση που τα δεδομένα δεν είναι τα προκαθορισμένα, δημιουργείται ο marker του χάρτη και τρέχουν οι συναρτήσεις **distanceFinder()** και **distanceMeasure()**, τις οποίες θα αναλύσω παρακάτω (γραμμές 18 - 31). Τέλος ορίζεται η συνάρτηση **retrieveData()** να τρέχει κάθε 3000 milliseconds = 3 seconds, δηλαδή κάθε 3 δευτερόλεπτα η εφαρμογή θα ελέγχει για νέα δεδομένα τη βάση δεδομένων του web server (γραμμές 32 - 34).

```
1.  // Initialize map-page map
2.  function initData(dbCoordinates, myZoom) {
3.      var myLatLng = new google.maps.LatLng(dbCoordinates[0].lat, dbCoordinate
    s[0].lng);
4.      var styledMap = new google.maps.StyledMapType(styles,
5.              {name: "Styled Map"});
6.      myOptions = {
7.          zoom: myZoom,
8.          center: myLatLng,
9.          disableDefaultUI: true,
10.         mapTypeControlOptions: {
11.             mapTypeIds: [google.maps.MapTypeId.ROADMAP, 'map_style']
12.         }
13.     };
14.     map = new google.maps.Map(document.getElementById('map'), myOptions);
15.     map.mapTypes.set('map_style', styledMap);
16.     map.setMapTypeId('map_style');
17.
18.     if (dbCoordinates[0].speed.localeCompare('no coordinates') !== 0) {
19.         marker = new google.maps.Marker({
20.             position: myLatLng,
21.             map: map,
22.             icon: imageBus
23.         });
24.         marker.setMap(map);
25.         if (enablePathLines == 'on') {
26.             drawPolylines(dbCoordinates);
27.         }
28.         distanceFinder(dbCoordinates);
29.         distanceMeasure(dbCoordinates);
30.         initCounter = 1;
31.     }
32.     myVar = setInterval(function () {
33.         retrieveData();
34.     }, 3000);  // Reload retrieveData function every 3 second
35. }
```

4.3.5.5 Σχεδίαση διαδρομής λεωφορείου

Η συνάρτηση **drawPolylines()** είναι υπεύθυνη για το σχεδιασμό της διαδρομής του λεωφορείου πάνω στο χάρτη. Δέχεται για όρισμα μια μεταβλητή τύπου Object, η οποία περιέχει τις συντεταγμένες και άλλα δεδομένα και σχεδιάζει τη διαδρομή.

Στις γραμμές 3 έως 8 δημιουργείται ένα Array List και στη συνέχεια αποθηκεύονται σε αυτό οι συντεταγμένες μια προς μία από το Object της εισόδου. Στις γραμμές 10 έως 16 γίνεται δημιουργία και αρχικοποίηση του αντικειμένου της διαδρομής με τιμές: τις συντεταγμένες των σημείων, το χρώμα του μονοπατιού, τη διαφάνεια του και το πάχος της γραμμής του μονοπατιού. Τέλος γίνεται η ενσωμάτωση στον ήδη υπάρχον χάρτη.

```
1.  // Draw path lines
2.  function drawPolylines(dbCoordinates) {
3.      var path = [];
4.      var i = 0;
5.      for (i = 0; i < dbCoordinates.length; i++) {
6.          var newCoordinates = new google.maps.LatLng(dbCoordinates[i].lat, db
    Coordinates[i].lng);
7.          path.push(newCoordinates);
8.      }
9.
10.     var flightPath = new google.maps.Polyline({
11.         path: path,
12.         strokeColor: '#d9e021',
13.         strokeOpacity: 1.0,
14.         strokeWeight: 15
15.     });
16.     flightPath.setMap(map);
17. }
```

4.3.5.6 Μετακίνηση marker

Εικόνα 86: Marker χάρτη

Η συνάρτηση **moveMarker()** δέχεται για ορίσματα το αντικείμενο ενός χάρτη, το αντικείμενο ενός marker και τις συντεταγμένες ξεχωριστά. Η λειτουργία της είναι να μετακινεί το marker του σχολικού λεωφορείου πάνω στο χάρτη όταν αλλάζει θέση. Επίσης στις γραμμές 4 και 5 είναι γραμμένος ο κώδικας για τη λειτουργικότητα των ρυθμίσεων για την εστίαση του χάρτη στο λεωφορείο ή όχι.

```
1.  // Move marker to the new position
2.  function moveMarker(map, marker, latitude, longitude) {
3.      marker.setPosition(new google.maps.LatLng(latitude, longitude));
4.      if (followBus === 'on') {
5.          map.panTo(new google.maps.LatLng(latitude, longitude));
6.      }
7.  }
```

4.3.5.7 Ανάκτηση δεδομένων από τη βάση δεδομένων

Η συνάρτηση **retrieveData()** είναι η συνάρτηση που τρέχει μετά από την αρχική εκτέλεση, κατά τη διάρκεια της λειτουργίας της εφαρμογής στην ενότητα της διαδρομής λεωφορείου. Κάνει ανάκτηση δεδομένων από τη βάση δεδομένων και πραγματοποιεί παραμετροποίηση σε όλες τις μεταβλητές της εφαρμογής με στόχο την ομαλή λειτουργία της. Έχει οριστεί να ελέγχει τη βάση δεδομένων κάθε 3 δευτερόλεπτα για νέα δεδομένα.

Η συνάρτηση χρησιμοποιεί την τεχνολογία Ajax για την ανάκτηση δεδομένων χωρίς να γίνεται αντιληπτό από το χρήστη. Η συνάρτηση δε δέχεται ορίσματα αλλά χρησιμοποιεί τις Global μεταβλητές που έχουν οριστεί στην αρχή. Στις γραμμές 3 έως 7 πραγματοποιείται η σύνδεση με το web server για την ανάκτηση των δεδομένων. Αν είναι επιτυχείς η σύνδεση, δηλαδή έχουμε status ίσο με 200, τότε ξεκινάει όλη η διαδικασία. Η απόκριση που δέχεται η συνάρτηση είναι σε μορφή κειμένου και περιέχει επίσης κάποια επιπλέον πληροφορία, η οποία παραλείπεται με μια εντολή while (γραμμές 12 - 13). Στη συνέχει μετατρέπεται το JSON αρχείο της εισόδου σε Object για να μπορεί να γίνει διαχείριση όπως έγινε και στη συνάρτηση **initMap()**.

Μόλις ολοκληρωθεί το προηγούμενο βήμα, το πρώτο πράγμα που εκτελείται είναι ο έλεγχος των δεδομένων που λήφθηκαν. Αν παραληφθούν μηδενικά δεδομένα ή δεδομένα που παραλήφθηκαν προηγουμένως η συνάρτηση δεν κάνει τίποτα απλώς περιμένει δεδομένα κάνοντας ένα επαναληπτικό έλεγχο κάθε 3 δευτερόλεπτα (γραμμές 41 - 51). Στην περίπτωση που έχουμε και δεδομένα στην απόκριση και δεν έχουμε δεχθεί ίδια δεδομένα με τις προηγούμενες φορές γίνονται κάποιες ενέργειες.

Η πρώτη ενέργεια είναι η αρχικοποίηση κάποιων μεταβλητών, οι οποίες είναι υπεύθυνες για τη σωστή λειτουργεία, όπως την παύση κάποιων ενεργειών αν δεν έχει μετακινηθεί το λεωφορείο (γραμμές 20 - 22). Επόμενη ενέργεια που πρέπει να πραγματοποιηθεί είναι να δημιουργηθεί ένας marker για το λεωφορείο στην περίπτωση που δεν υπάρχει (γραμμές 23 - 29) και στη συνέχεια να τοποθετηθεί πάνω στο χάρτη με σωστή εστίαση, σωστές συντεταγμένες και σωστό κεντράρισμα (γραμμές 30 - 35). Υπάρχει και η περίπτωση να έχει δημιουργηθεί ο marker του λεωφορείου κατά την αρχικοποίηση της εφαρμογής στην οποία το μόνο που θα χρειαστεί να γίνει είναι να μετακινηθεί ο marker στις σωστές συντεταγμένες.

Στο τελευταίο κομμάτι υπάρχουν οι συναρτήσεις που δίνουν τα αποτελέσματα για την απόσταση από το χρήστη, την ταχύτητα και τη μέση ταχύτητα με την οποία πηγαίνει το λεωφορείο, την απόσταση που έχει διανύσει το λεωφορείο, όπως και το σχεδιασμό της διαδρομής του λεωφορείου. Η συνάρτηση για το σχεδιασμό της διαδρομής του λεωφορείου είναι μέσα σε μια συνθήκη, η οποία ελέγχεται από τις ρυθμίσεις σύμφωνα με τις προτιμήσεις του χρήστη (γραμμές 36 - 40).

```
1.  // Retrieve data from server
2.  function retrieveData() {
3.      var xmlhttp = new XMLHttpRequest();
4.      var url = "http://g-alex.net78.net/gps/data.php";
5.
6.      xmlhttp.onreadystatechange = function () {
7.          if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
8.              var response = xmlhttp.responseText;
9.              var i = 0;
10.
11.             // Clean JSON from rubbish
12.             while (response[i] != ']') {
13.                 i++;
14.             }
15.             var dbObject = response.substring(0, i + 1);
16.             var dbCoordinates = JSON.parse(dbObject);
17.
18.             // Check if db is empty and if previous coordinates exist
19.             if (dbCoordinates.length != 0 && counter < dbCoordinates.length)
     {
20.                 console.log("0");
21.                 counter++;
22.                 mapCounter = 0;
23.                 if (initCounter == 0) {
24.                     var myLatLng = new google.maps.LatLng(dbCoordinates[0].l
    at, dbCoordinates[0].lng);
25.                     marker = new google.maps.Marker({
26.                         position: myLatLng,
27.                         map: map,
28.                         icon: imageBus
29.                     });
30.                     marker.setMap(map);
31.                     map.setZoom(15);
32.                     map.setCenter(myLatLng);
33.                     initCounter = 2;
34.                 }
35.                 moveMarker(map, marker, dbCoordinates[0].lat, dbCoordinates[
    0].lng);
36.                 if (enablePathLines == 'on') {
37.                     drawPolylines(dbCoordinates);
38.                 }
39.                 distanceFinder(dbCoordinates);
40.                 distanceMeasure(dbCoordinates);
```

```
41.             } else if (dbCoordinates.length == 0 && counter > dbCoordinates.
    length) {
42.                 console.log("1");
43.                 mapCounter++;
44.                 if (mapCounter == 1) {
45.                     initCounter = 0;
46.                     counter = 0;
47.                     myVar = 0;
48.                     averageSpeed = 0;
49.                     initMap();
50.                 }
51.             }
52.         }
53.     };
54.     xmlhttp.open("POST", url, true);
55.     xmlhttp.send();
56. }
```

4.3.5.8 Υπολογισμός απόστασης και ταχύτητας

Η συνάρτηση **distanceMeasure()** είναι η συνάρτηση υπολογισμού ταχύτητας και απόστασης. Δέχεται για είσοδο ένα πίνακα που περιέχει τις συντεταγμένες. Στις γραμμές 3 έως 21, γίνεται ο υπολογισμός της απόστασης που έχει διανύσει το λεωφορείο μετρώντας κάθε φορά την απόσταση μεταξύ του κάθε σημείου με το επόμενο σημείο.

Για τον υπολογισμό της μέσης ταχύτητας, αθροίζονται οι συνολικές μετρήσεις και διαιρούνται με το πλήθος τους (γραμμές 23 - 29). Τελευταία ενέργεια που πραγματοποιείται είναι η εμφάνιση των δεδομένων της μέσης ταχύτητας και της απόστασης της διαδρομής στα δύο πάνελ της εφαρμογής (γραμμές 31 - 33).

```
1.  // Measure the route distance and the average speed
2.  function distanceMeasure(dbCoordinates) {
3.      var busDriverName = dbCoordinates[0].name;
4.      var busCurrentSpeed = dbCoordinates[0].speed;
5.      var distance = [];
6.      var i = 0;
7.      for (i = 0; i < dbCoordinates.length - 1; i++) {
8.          if (dbCoordinates.length > i) {
9.              var origin = new google.maps.LatLng(dbCoordinates[i + 1].lat, db
    Coordinates[i + 1].lng);
10.             var destination = new google.maps.LatLng(dbCoordinates[i].lat, d
    bCoordinates[i].lng);
11.
12.             var dis = google.maps.geometry.spherical.computeDistanceBetween(
    origin, destination);
13.             distance.push(dis);
14.         }
15.     }
16.
17.     var sumDistance = 0;
18.     var j = 0;
19.     for (j = 0; j < distance.length; j++) {
20.         sumDistance = sumDistance + distance[j];
21.     }
22.
23.     var k = 0;
24.     var sumSpeed = 0;
25.     for (k = 0; k < dbCoordinates.length; k++) {
```

```
26.         var dbSpeed = parseInt(dbCoordinates[k].speed);
27.         sumSpeed = sumSpeed + dbSpeed;
28.     }
29.     averageSpeed = sumSpeed / dbCoordinates.length;
30.
31.     var outputDiv = document.getElementById('output2');
32.     outputDiv.innerHTML = 'Driver
    name: ' + busDriverName + '<br>Speed: ' + busCurrentSpeed + ' km/h<br>Distan
    ce: ' + Math.round(sumDistance) + ' meters<br>Average
    speed: ' + Math.round(averageSpeed) + ' km/h';
33. }
```

4.3.5.9 Εύρεση θέσης χρήστη

Η εύρεση της θέσης του χρήστη χρειάζεται για τον υπολογισμό του χρόνου που θα κάνει να φτάσει το λεωφορείο σε αυτόν. Άρα χρειάζεται η ακριβή θέση του χρήστη για την παροχή έγκυρων αποτελεσμάτων στο χρόνο που θα χρειαστεί το λεωφορείο για να φτάσει στον προορισμό του, δηλαδή στο χρήστη. Η είσοδος της συνάρτησης **distanceFinder()** είναι ένας πίνακας με τις συντεταγμένες και λοιπές πληροφορίες οι οποίες είναι αποθηκευμένες στη βάση δεδομένων.

Στις γραμμές 12 έως 13 χρησιμοποιείται η συγκεκριμένη συνάρτηση για την εύρεση της τοποθεσίας του χρήστη. Στην περίπτωση επιτυχίας τρέχει η συνάρτηση **finder()** με τις συντεταγμένες του χρήστη αλλιώς σε περίπτωση αποτυχίας τρέχει η συνάρτηση με προεπιλεγμένες συντεταγμένες. Στην περίπτωση που δεν υποστηρίζεται η λειτουργία εύρεσης θέσης τρέχει η συνάρτηση με τις προεπιλεγμένες συντεταγμένες.

```
1.  // Find the client gps spot
2.  function distanceFinder(dbCoordinates) {
3.      if (navigator.geolocation) {
4.          function success(pos) {
5.              // Location found, show map with these coordinates
6.              finder(dbCoordinates, new google.maps.LatLng(pos.coords.latitude
    , pos.coords.longitude));
7.          }
8.          function fail(error) {
9.              // Failed to find location, show default map
10.             finder(dbCoordinates, defaultCoordinates);
11.         }
12.         // Find the users current position. Retrieve the real current positi
    on, timeout after 6 seconds
13.         navigator.geolocation.getCurrentPosition(success, fail, {maximumAge:
     0, enableHighAccuracy: true, timeout: 6000});
14.     } else {
15.         finder(dbCoordinates, defaultCoordinates);B  // No geolocation suppo
    rt, show default map
16.     }
17. }
```

4.3.5.10 Υπολογισμός απόστασης λεωφορείου και χρόνου άφιξης

Η συνάρτηση **finder()** δέχεται για είσοδο δύο Objects, όπου το πρώτο περιέχει τις συντεταγμένες του λεωφορείου και το δεύτερο τις συντεταγμένες του χρήστη πάνω στο χάρτη. Στις γραμμές 3 έως 15 δηλώνονται και γίνεται αρχικοποίηση των μεταβλητών που θα περιέχουν τις συντεταγμένες αφετηρία, δηλαδή του λεωφορείου και τις συντεταγμένες προορισμού, δηλαδή του χρήστη. Στη συνέχεια ελέγχεται αν είχε δημιουργηθεί προηγουμένως ο marker του χρήστη. Αν όχι τότε δημιουργείται, αλλιώς απλώς αλλάζει η θέση του.

Το Google API βοήθησε πολύ αφού χρησιμοποιήθηκε για τον υπολογισμό της απόστασης από το χρήστη. Η λογική βασίζεται στη εύρεση της κοντινότερης διαδρομής από το λεωφορείο προς το χρήστη (γραμμές 17 - 38). Στις γραμμές 18 έως 24 εισάγονται τα σημεία λεωφορείου και χρήστη και οι ιδιότητες με τις οποίες πρέπει να εκτελεστεί η συνάρτηση, όπως η διαδρομή θα υπολογιστεί σε μετρικό σύστημα μονάδων, η διαδρομή θα γίνει με κάποιο όχημα, θα συμπεριληφθούν οι εθνικές οδοί στη μέτρηση και θα υπολογιστούν και τα διόδια αν υπάρχουν. Στις γραμμές 38 έως 49 εμφανίζονται το αποτέλεσμα της συνάρτησης, δηλαδή η θέση του λεωφορείου, η θέση του χρήστη και η απόσταση μεταξύ τους και ο χρόνος που απαιτείται, έτσι ώστε το λεωφορείο να φτάσει στο χρήστη.

Όλα τα αποτελέσματα εμφανίζονται στο αριστερό πάνελ της εφαρμογής και είναι κατάλληλα διαμορφωμένα, έτσι ώστε να είναι ευανάγνωστα και κατανοητά. Για αυτό το λόγο αν το αποτέλεσμα του χρόνου είναι πολύ μικρό, δηλαδή για παράδειγμα 0.1 λεπτά, αυτό μετατρέπεται αυτόματα σε δευτερόλεπτα που είναι ίσο με 6 δευτερόλεπτα. Ο τύπος που χρησιμοποιήθηκε είναι:

$$\boldsymbol{time\ (seconds) = time\ (minutes) * 60}$$

Αντίστοιχα και για τις υπόλοιπες υποδιαιρέσεις του χρόνου χρησιμοποιήθηκαν και οι αντίστοιχοι τύποι:

$$\boldsymbol{time\ (minutes) = time\ (hours) * 60}$$

```
1.  // Measure the distance from the last GPS spot to the client
2.  function finder(dbCoordinates, geoCoordinates) {
3.      var origin = new google.maps.LatLng(dbCoordinates[0].lat, dbCoordinates[
    0].lng);
4.      var destination = geoCoordinates;
5.
6.      if (clientMarkerMove === 0) {
7.          clientMarker = new google.maps.Marker({
8.              position: geoCoordinates,
9.              map: map
10.         });
11.         clientMarker.setMap(map);
12.         clientMarkerMove = 1;
13.     } else if (clientMarkerMove === 1) {
14.         clientMarker.setPosition(geoCoordinates);
15.     }
16.
17.     var service = new google.maps.DistanceMatrixService;
18.     service.getDistanceMatrix({
19.         origins: [origin],
20.         destinations: [destination],
21.         travelMode: google.maps.TravelMode.DRIVING,
22.         unitSystem: google.maps.UnitSystem.METRIC,
```

```
23.         avoidHighways: false,
24.         avoidTolls: false
25.     }, function (response, status) {
26.         if (status !== google.maps.DistanceMatrixStatus.OK) {
27.             alert('Error was: ' + status);
28.         } else {
29.             var originList = response.originAddresses;
30.             var destinationList = response.destinationAddresses;
31.             var outputDiv = document.getElementById('output1');
32.             outputDiv.innerHTML = '';
33.
34.             for (var i = 0; i < originList.length; i++) {
35.                 var results = response.rows[i].elements;
36.                 for (var j = 0; j < results.length; j++) {
37.
38.                     var estimatedTime = '';
39.                     var tmp = (results[j].distance.value / 1000) / averageSp
    eed;
40.                     if (tmp > 1) {
41.                         estimatedTime = Math.round((results[j].distance.valu
    e / 1000) / averageSpeed) + ' hours';
42.                     } else if ((tmp * 10) > 1) {
43.                         estimatedTime = Math.round(((results[j].distance.val
    ue / 1000) / averageSpeed) * 60) + ' minutes';
44.                     } else if ((tmp * 100) > 1) {
45.                         estimatedTime = Math.round(((results[j].distance.val
    ue / 1000) / averageSpeed) * 3600) + ' seconds';
46.                     } else { estimatedTime = '0 seconds'; }
47.
48.                     outputDiv.innerHTML += 'Bus: ' + originList[i] + '<br>'
    + userName + ': ' + destinationList[j] + '<br>Distance: ' + results[j].dista
    nce.text +
49.                                 '<br>Estimated arrival
    time: ' + estimatedTime;
50.                 }
51.             }
52.         }
53.     });
54. }
```

4.3.5.11 Ανάλυση κώδικα της επιλογής η Θέση μου

Το τελευταίο κομμάτι του κώδικα στην εφαρμογή χρήστη είναι προαιρετικό και ίσως και βοηθητικό προς το χρήστη. Ίσως με μια μελλοντική αναβάθμιση της εφαρμογής, να είναι ένα πολύ σημαντικό και βοηθητικό κομμάτι της.

Ο κώδικας για τη συγκεκριμένη επιλογή τρέχει μόνο όταν εμφανίζεται η σελίδα. Στις γραμμές 3 έως 16 γίνεται εύρεση της θέσης του χρήστη με τον ίδιο ακριβώς τρόπο αναλύθηκε παραπάνω στην προηγούμενη ενότητα. Στη συνέχεια (γραμμές 18 - 38), δημιουργείται ένας χάρτης με ένα marker στη θέση του χρήστη. Αυτό το κομμάτι είναι παρόμοιο με προηγούμενα κομμάτια του κώδικα για τη δημιουργία χάρτη και marker για αυτό δεν θα αναλυθεί εκτενέστερα.

```
1.  // Triggered on the "to" page, after the transition animation completes - ma
    p-page page
2.  $(document).on("pageshow", "#map-page", function () {
3.      var defaultLatLng = new google.maps.LatLng(35.338643, 25.133018);B  // D
    efault to Heraklion Crete when no geolocation support
4.      if (navigator.geolocation) {
```

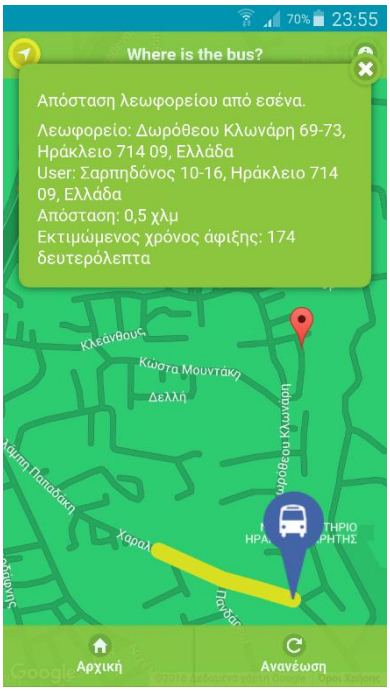
```
        function success(pos) {
            // Location found, show map with these coordinates
            drawMap(new google.maps.LatLng(pos.coords.latitude, pos.coords.l
ongitude));
        }
        function fail(error) {
            // Failed to find location, show default map
            drawMap(defaultLatLng);
        }
        // Find the users current position. Retrieve the real current positi
on, timeout after 6 seconds
        navigator.geolocation.getCurrentPosition(success, fail, {maximumAge:
 0, enableHighAccuracy: true, timeout: 6000});
    } else {
        drawMap(defaultLatLng);B  // No geolocation support, show default ma
p
    }
    function drawMap(latlng) {
        var styledMap = new google.maps.StyledMapType(styles,
                {name: "Styled Map"});
        myOptions = {
            zoom: 15,
            center: latlng,
            disableDefaultUI: true,
            mapTypeControlOptions: {
                mapTypeIds: [google.maps.MapTypeId.ROADMAP, 'map_style']
            }
        };
        var map = new google.maps.Map(document.getElementById('map-
canvas'), myOptions);
        map.mapTypes.set('map_style', styledMap);
        map.setMapTypeId('map_style');

        // Add an overlay to the map of current lat/lng
        var marker = new google.maps.Marker({
            position: latlng,
            map: map
        });
    }
});
```

5 Αποτελέσματα και μελλοντικές επεκτάσεις

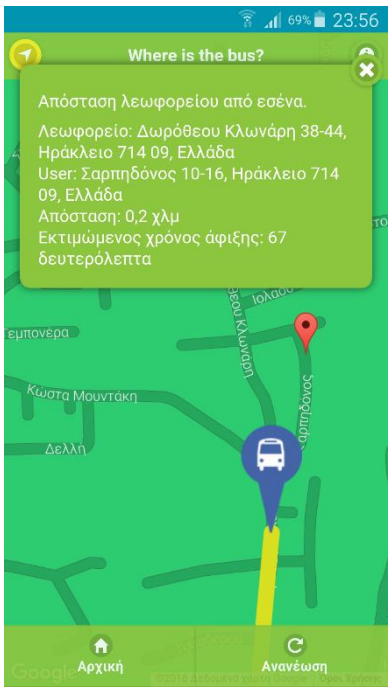
5.1 Αποτελέσματα

Το αποτέλεσμα της εργασίας είναι πάρα πολύ καλό και ικανοποιητικό ως προς τη λειτουργικότητα και την απόδοση. Παρακάτω παρουσιάζονται κάποιες περιπτώσεις χρήσης, όπου το λεωφορείο κινείται και ο χρήστης βρίσκεται κοντά στο κέντρο της πόλης του Ηρακλείου.

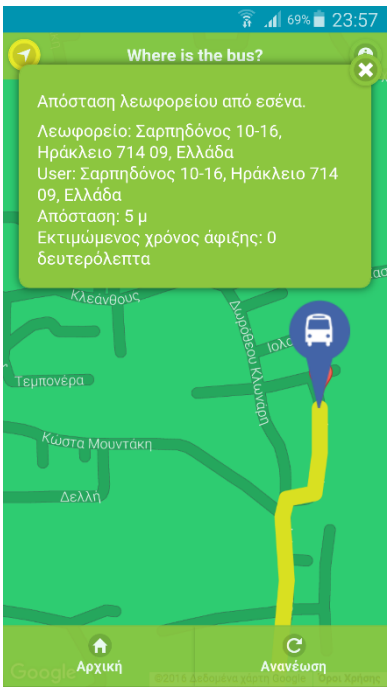
Εικόνα 87: Δεν έχει δημιουργηθεί η διαδρομή



Εικόνα 88: Διαδρομή 10%

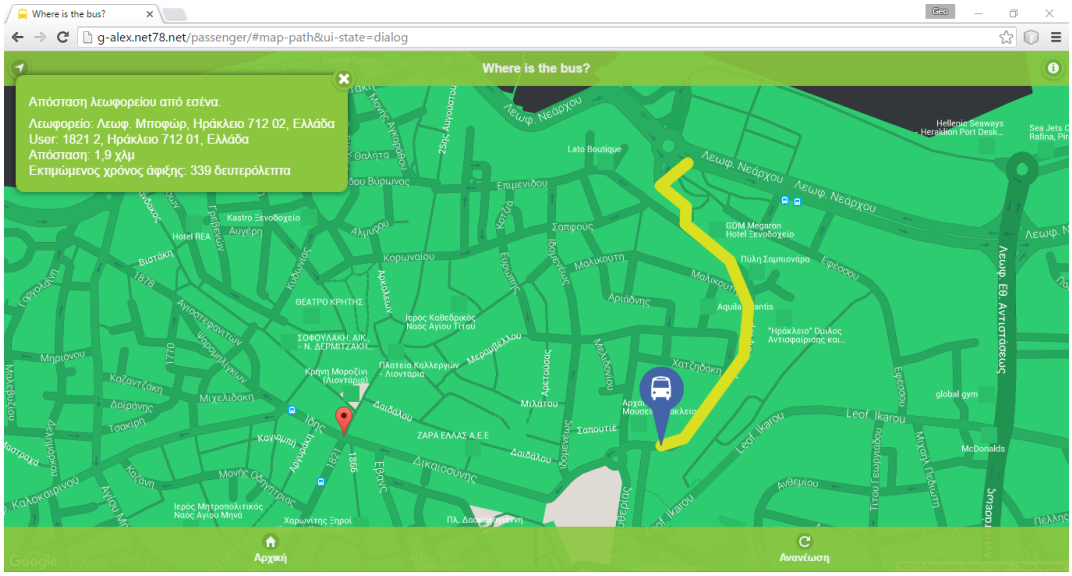


Εικόνα 89: Διαδρομή 60%

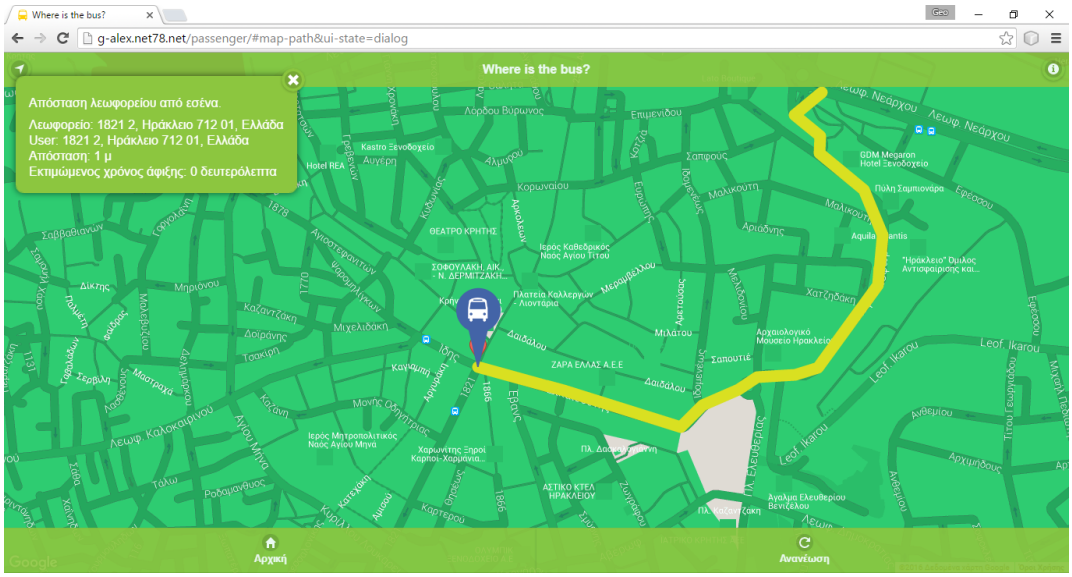


Εικόνα 90: Διαδρομή 100%

Επίσης βασικό στοιχείο της εφαρμογής είναι η διαθεσιμότητα της μέσω του διαδικτύου. Παρακάτω παρουσιάζονται κάποιες περιπτώσεις, όπου ο χρήστης βρίσκεται στο κέντρο της πόλης του Ηρακλείου.



Εικόνα 91: Διαδρομή με web browser 50%



Εικόνα 92: Διαδρομή με web browser 100%

5.2 Μελλοντικές επεκτάσεις

Στη συγκεκριμένη εργασία παρουσιάστηκε μια ολοκληρωμένη υλοποίηση μιας πλατφόρμας παρακολούθησης σχολικού λεωφορείου, η οποία ενημερώνει τους μαθητές για τη θέση του σχολικού λεωφορείο. Κάποιες μελλοντικές επεκτάσεις προκύπτουν από τους ίδιους τους μαθητές, τις ανάγκες τους και τις απαιτήσεις τους. Επίσης κάποιες μελλοντικές επεκτάσεις προκύπτουν και από τα ίδια τα σχολεία, τις ιδιαιτερότητες που έχουν, την περιοχή που βρίσκονται και τον τρόπο που λειτουργούν.

Πολλά σχολεία χρησιμοποιούν παραπάνω από ένα λεωφορείο με τον οδηγό του για την παραλαβή των μαθητών. Η εφαρμογή στη συγκεκριμένη υλοποίηση υποστηρίζει μόνο ένα οδηγό. Άρα η ενημέρωση της εφαρμογής για την υποστήριξη περισσότερων οδηγών θα καλύψει την ανάγκη περισσότερων σχολείων. Επίσης η εφαρμογή μπορεί να γίνει τόσο δημοφιλής και θα μπορούσε να χρησιμοποιηθεί και από σχολεία και μαθητές άλλων χωρών. Η προσθήκη επιπλέον γλωσσών θα ήταν πάρα πολύ σημαντική για την στόχευση ξένων χωρών.

Αν και τα περισσότερα smartphones ή tablets έχουν αισθητήρες GPS, κάποιες πιο φθηνές συσκευές δεν έχουν και αυτές που είναι ήδη εξοπλισμένες, η υπηρεσία GPS αρκετές φορές δεν είναι διαθέσιμη λόγω τοποθεσίας. Μια πολύ σημαντική επέκταση για την εφαρμογή του μαθητή θα ήταν ο ορισμός της θέσης του μαθητή χειροκίνητα με τη χρήση του χάρτη ή την προσθήκη κάποιου πεδίου στις ρυθμίσεις.

Η υλοποίηση της εφαρμογής βασίστηκε στο δεδομένο, ότι οι μαθητές γνωρίζουν την διαδρομή που πραγματοποιεί το λεωφορείο και η κάθε στάση είναι το σπίτι του κάθε μαθητή. Σε κάποιες περιπτώσεις όμως αυτό μπορεί να μην ισχύει και το σχολικό λεωφορείο να πραγματοποιεί διαφορετικές διαδρομές καθημερινά ή να κάνει διαφορετικές στάσεις. Επεκτείνοντας την πλατφόρμα, έτσι ώστε ο οδηγός να ορίζει τη διαδρομή και τις στάσεις που θα κάνει όχι μόνο θα καλύψει την επιπλέον ιδιαιτερότητα ορισμένων σχολείων, αλλά ίσως η πλατφόρμα εξελιχθεί σε μια πλατφόρμα η οποία θα μπορεί να υποστηρίξει πολλά περισσότερα μέσα μεταφοράς όπως ταξί, αστικά λεωφορεία, και πολλά μέσα μεταφοράς.

Μια τελευταία αλλά ίσως μια σημαντική μελλοντική εξέλιξη θα ήταν η χρήση μη εμπορικής υπηρεσίας χαρτών, όπως είναι το OpenStreetMap (OSM), το οποίο αναπτύσσεται μέσω του crowdsourcing, σε αντίθεση με το Google Maps. Με τη συγκεκριμένη επέκταση δε θα υπάρχουν πλέον οι περιορισμοί της χρήσης ενός εμπορικού API, όπως το Google Maps API και θα διευκολυνθεί η περεταίρω ανάπτυξη της εφαρμογής του μαθητή, όπου είναι αναγκαία η χρήση χαρτών.

Βιβλιογραφία και πηγές

[1] HTML tutorial - http://www.w3schools.com/html/
[2] JavaScript tutorial - http://www.w3schools.com/js/
[3] Ajax tutorial - http://www.w3schools.com/ajax/
[4] CSS tutorial - http://www.w3schools.com/css/
[5] jQuery tutorial - http://www.w3schools.com/jquery/
[6] jQuery Mobile tutorial - http://www.w3schools.com/jquerymobile/
[7] JSON tutorial - http://www.w3schools.com/json/
[8] PHP tutorial - http://www.w3schools.com/php/
[9] SQL tutorial - http://www.w3schools.com/sql/
[10] Google Maps tutorial - http://www.w3schools.com/googleapi/
[11] Java documentation - https://docs.oracle.com/javase/7/docs/api/
[12] JSON format - http://www.json.org/
[13] Android API - http://developer.android.com/guide/index.html
[14] Apache Cordova documentation - https://cordova.apache.org/docs/en/latest/guide/overview/
[15] HTML - https://en.wikipedia.org/wiki/HTML
[16] CSS - https://en.wikipedia.org/wiki/Cascading_Style_Sheets
[17] JavaScript - https://en.wikipedia.org/wiki/JavaScript
[18] Ajax - https://en.wikipedia.org/wiki/Ajax_(programming)
[19] jQuery - https://en.wikipedia.org/wiki/JQuery
[20] jQuery Mobile - https://en.wikipedia.org/wiki/JQuery_Mobile
[21] JSON - https://en.wikipedia.org/wiki/JSON
[22] PHP - https://en.wikipedia.org/wiki/PHP
[23] SQL - https://en.wikipedia.org/wiki/SQL
[24] Java - https://en.wikipedia.org/wiki/Java
[25] Android - https://en.wikipedia.org/wiki/Android_(operating_system)
[26] Google Maps - https://en.wikipedia.org/wiki/Google_Maps
[27] Ajax: A New Approach to Web Applications - http://adaptivepath.org/ideas/ajax-new-approach-web-applications/
[28] History of PHP - http://php.net/manual/en/history.php.php
[29] Internet of Things - https://en.wikipedia.org/wiki/Internet_of_Things
[30] Internet of Things Global Standards Initiative - http://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx
[31] Internet of Things: Science fiction or business fact? - https://hbr.org/resources/pdfs/comm/verizon/18980_HBR_Verizon_IoT_Nov_14.pdf
[32] Android Προγραμματισμός 2η Έκδοση - Paul Deitel, Harvey Deitel, Abbey Deitel
[33] Δικτύωση Υπολογιστών: Προσέγγιση από Πάνω προς τα Κάτω - James F. Kurose, Keith W. Ross
[34] Τεχνικές δημιουργίας και συντήρησης Ιστοσελίδων - Αλέξανδρος Καράκος
[35] Πλήρες εγχειρίδιο της JAVA 7 - Rogers Cadenhead
[36] Συστήματα Βάσεων Δεδομένων (4η Έκδοση) - Abraham Silberschatz, Henry F. Korth, S. Sudarshan