ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ & ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΓΕΩΓΡΑΦΙΑΣ ΚΑΙ ΠΕΡΙΦΕΡΕΙΑΚΟΥ ΣΧΕΔΙΑΣΜΟΥ

City Stroller: Σχεδιασμός και υλοποίηση πρότυπης εφαρμογής βέλτιστης δρομολόγησης και αξιοποίησης ελεύθερου χρόνου σε περιβάλλον έξυπνων κινητών

Διπλωματική εργασία

του

Κωνσταντίνου Γ. Αποστολάκη

Επιβλέπων: Φώτης Ν. Γεώργιος

Αθήνα, Μάρτιος 2019

NATIONAL & TECHNICAL UNIVERSITY
SCHOOL OF RURAL & SURVEING ENGINEERING
GEOGRAPHY & REGIONAL PLANNING

City Stroller: Design and implementation of a standard application of optimal routing and exploitation of free time in environment of smart mobiles environment

Thesis

of

Konstantinos G. Apostolakis

Supervisor: Photis N. Georgios

Athens, March 2019

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΙΝΑΚΕΣ

ΕΙΚΟΝΕΣ

ΔΙΑΓΡΑΜΜΑΤΑ

ΑΠΟΣΠΑΣΜΑΤΑ

ΠΕΡΙΛΗΨΗ

Σήμερα, με την ανάπτυξη της τεχνολογίας, τα «έξυπνα» κινητά τηλέφωνα αποτελούν πλέον ένα σημαντικό κομμάτι της καθημερινότητάς μας. Λίγοι είναι αυτοί που ακόμα χρησιμοποιούν παλιές μεθόδους για καθημερινά ζητήματα και σε λίγα χρόνια πιθανόν να μην υπάρχει άνθρωπος χωρίς «έξυπνο» κινητό. Οι εφαρμογές που αναπτύσσονται καθημερινά αποσκοπούν στην διευκόλυνση των χρηστών των συσκευών. Παράλληλα, ο τομέας του τουρισμού είναι ένας κλάδος, ο οποίος καθορίζει σε σημαντικό βαθμό την οικονομία της Ελλάδας. Ο συνδυασμός των δύο αυτών παραγόντων παρουσιάζεται στην παρούσα διπλωματική εργασία.

Με βάση σημεία ενδιαφέροντος στην Αθήνα, αναπτύσσεται μια εφαρμογή για συσκευές Android, μέσα από την οποία οι χρήστες θα μπορούν να περιηγηθούν σε αυτά τα σημεία ορίζοντας απλά την θέση, τον χρόνο, τα χρήματα και τις προτιμήσεις τους. Η εφαρμογή θα προτείνει αυτόματα στον χρήστη μια διαδρομή με βάση τα κριτήρια που έχει επιλέξει και θα δίνει οδηγίες μετάβασης από κάθε σημείο προς τα υπόλοιπα. Η εφαρμογή αυτή, αν και αποτελεί μια από τις πολλές που υπάρχουν και προσφέρουν δυνατότητες περιήγησης, στοχεύει σε άτομα με λίγη εξοικείωση της τεχνολογίας, που παράλληλα θέλουν να οργανώσουν τον χρόνο τους γρήγορα. Έχει αρκετά χαρακτηριστικά, που λίγες εφαρμογές σήμερα διαθέτουν, με βασικότερο την χρήση χωρίς την απαίτηση σύνδεσης στο διαδίκτυο.

Στην εργασία αυτή, αναλύεται όλη η διαδικασία από την επιλογή της περιοχής μελέτης, που θα είναι η πρωτεύουσα της Ελλάδας, η Αθήνα, μέχρι την συλλογή των δεδομένων και την ανάπτυξη της εφαρμογής. Περιγράφεται αναλυτικά η λειτουργία της εφαρμογής και τέλος προτείνονται κάποιες αλλαγές που θεωρούνται απαραίτητες πριν την δημοσίευσή της στο ευρύ κοινό.

ABSTRACT

Today, with the development of technology, smartphones are now an important part of our daily routine. Few people still use old methods for everyday issues and in a few years, there may not be a person without a smartphone. Applications developed on a daily basis are designed to facilitate device users. At the same time, the sector of tourism is a sector, which determines to a significant level the economy of Greece. The combination of these two factors is presented in this diploma thesis.

Based on points of interest in Athens, an application for Android devices is developed, through which users can have a tour to these points simply by setting their location, time, money and their preferences. The application will automatically suggest the user a route based on the criteria selected and give directional instructions from each point to the rest. This application, although is one of the many existing and offering browsing capabilities, targets people with little technology familiarity, who also want to organize their time quickly. It has several features that few applications today have, with the most basic the use without the need to connect to the internet.

In this paper, the whole process is analyzed from the selection of the study area, which will be the capital of Greece, Athens, to the collection of data and the development of the application. It describes the operation of the application and finally suggests some changes that are considered necessary before it is published to the general public.

ΕΙΣΑΓΩΓΗ

Η Ελλάδα αποτελεί δημοφιλή τουριστικό προορισμό κυρίως εξαιτίας της εκτενούς ακτογραμμής της, των πολυάριθμων νησιών της και της φυσικής της ομορφιάς. Ένας από τους κύριους κλάδους της οικονομίας της είναι ο τουρισμός. Για τον λόγο αυτό έχει οργανωθεί ο ελλαδικός χώρος με τέτοιο τρόπο ώστε να παρέχει τις κατάλληλες υπηρεσίες στους επισκέπτες. Έχουν δημιουργηθεί μουσεία, αξιοθέατα και άλλα σημεία ενδιαφέροντος που μπορούν να μυήσουν τον επισκέπτη στην ιστορία και την ελληνική νοοτροπία. Όλα αυτά τα σημεία ενδιαφέροντος καλύπτουν σχεδόν όλη την Ελλάδα, δίνοντας έτσι επιλογές σε όποιον ενδιαφέρεται να την γνωρίσει καλύτερα. [Πηγές: www.wwf.gr, el.wikipedia.org].

Σε μερικές περιοχές μάλιστα, η πυκνότητα αυτών των σημείων είναι τόσο μεγάλη, που ο επισκέπτης μπορεί κάλλιστα να βρεθεί σε σύγχυση από τις τόσες πληροφορίες που υπάρχουν κοντά του. Οι περιοχές αυτές είναι κυρίως τα μεγάλα αστικά κέντρα και οι περιοχές τριγύρω τους. Όσο πιο μεγάλο και τουριστικά ανεπτυγμένο είναι το κάθε μέρος, τόσο πιο πολλά σημεία ενδιαφέροντος προσφέρει. Έτσι, ο επισκέπτης – τουρίστας ή ακόμα και ένας κάτοικος που θέλει να γνωρίσει καλύτερα την ιστορία του τόπου του, μπορεί να επιλέξει ανάμεσα σε πολλές εναλλακτικές επιλογές για να διαθέσει τον χρόνο και τα χρήματά του, προσφέροντας έτσι εκτός από γνώσεις και έναν εναλλακτικό τρόπο ψυχαγωγίας σε αυτόν και πιθανόν και την οικογένειά του.

Οι πολλές επιλογές ωστόσο πολλές φορές δημιουργούν προβλήματα. Ένα από τα σημαντικότερα προβλήματα που δημιουργούνται είναι η έλλειψη πληροφοριών. Με την δημιουργία ενός σημείου ενδιαφέροντος, είτε αυτό είναι κάποιο μουσείο, κάποια ιστορική εκκλησία ακόμα και ένα εμπορικό κέντρο ή ένα στάδιο, θα πρέπει με κάποιο τρόπο να ενημερωθεί ο κόσμος για την ύπαρξή του. Θα πρέπει να γνωρίζει λοιπόν ο καθένας που επισκέπτεται αυτή την πόλη ότι σε εκείνο το σημείο υπάρχει ένα σημείο ενδιαφέροντος που αξίζει να το επισκεφθεί. Ένας τρόπος που χρησιμοποιείται είναι η σήμανση. Υπάρχει για αρκετά σημεία σήμανση, όπου καθοδηγεί τους επισκέπτες προς ένα σημείο. Προφανώς όσο αυξάνονται τα μέρη που δημιουργούνται είναι αδύνατον να αυξάνεται και η σήμανση και πόσο μάλλον να προστεθεί σήμανση σε όλη την πόλη για να κατευθυνθεί στο σημείο ο κάθε επισκέπτης. Για τον λόγο αυτό έχουν τοποθετηθεί πινακίδες μόνο για τα σημαντικά και μεγάλα αξιοθέατα, που μπορούν να οδηγήσουν τους επισκέπτες σε αυτά, αλλά και αυτές οι πινακίδες συνήθως τοποθετούνται σε κοντινή απόσταση από το κάθε σημείο. Αυτό σημαίνει ότι αν κάποιος είναι τυχερός και βρεθεί κοντά σε ένα τέτοιο σημείο και παρατηρήσει την σήμανση θα έχει την ευκαιρία να το επισκεφθεί. Βέβαια υπάρχει και η πιθανότητα να το επισκεφθεί και η τιμή του εισιτηρίου να είναι μεγαλύτερη από αυτό που ο ίδιος θέλει να διαθέσει.

Ένας άλλος τρόπος πληροφόρησης, κυρίως των επισκεπτών της πόλης που δεν μένουν σε αυτήν, είναι μέσω χαρτών που προσφέρονται δωρεάν σε κεντρικά σημεία της πόλης. Αν η πόλη διαθέτει αεροδρόμιο, τότε αυτό είναι ένα σημείο διάθεσης τέτοιων χαρτών. Στους χάρτες αυτούς μπορεί κανείς να δει πληροφορίες για το ποια σημεία ενδιαφέροντος υπάρχουν στην συγκεκριμένη περιοχή και που βρίσκονται αυτά. Συνήθως υπάρχουν και πληροφορίες για τους τρόπους μετακίνησης μέσα στην περιοχή. Ωστόσο, σε μεγάλα αστικά κέντρα αυτές οι πληροφορίες είναι πάρα πολλές και δεν μπορούν να ενσωματωθούν σε ένα τυπωμένο χάρτη, περιορίζοντας έτσι τις επιλογές ενός επισκέπτη και παράλληλα αποτρέποντάς τον να επισκεφθεί άλλα δευτερεύοντος σημασίας σημεία ενδιαφέροντος. Αυτό το πρόβλημα συμβαίνει και με την λύση της σήμανσης.

Σήμερα, με την βοήθεια του διαδικτύου έχουν δημιουργηθεί αρκετές ιστοσελίδες και πλατφόρμες που μπορούν να προσφέρουν όλες τις διαθέσιμες πληροφορίες για τα σημεία ενδιαφέροντος κάθε περιοχής της Ελλάδας. Μπορεί κανείς να αναζητήσει τα σημεία ενδιαφέροντος που επιθυμεί ανά περιοχή, να δει παραπάνω πληροφορίες που αφορούν την τοποθεσία, την τιμή του εισιτηρίου αν απαιτείται, καθώς επίσης να δει και φωτογραφίες. Με αυτόν τον τρόπο μπορεί ο καθένας από το σπίτι του να οργανώσει το ταξίδι του ίσως και σε μια άλλη χώρα επιλέγοντας από πριν που θα

πάει. Επιπλέον υπάρχει η δυνατότητα να δει ο κάθε χρήστης σχόλια από υπόλοιπους που έχουν ήδη επισκεφθεί το κάθε μέρος και να σχηματίσει μια πιο πλήρη άποψη για αυτό.

Σημαντική είναι και η συμβολή των έξυπνων κινητών (smartphones) στην λύση του προβλήματος. Όπως αναφέρθηκε πριν, ο κάθε επισκέπτης μιας πόλης θα μπορεί να προετοιμάσει το ταξίδι του από το σπίτι του μέσω των πληροφοριών που παρέχονται στο διαδίκτυο. Πλέον όμως μπορεί ο καθένας, όπου και αν βρίσκεται να έχει πρόσβαση μέσω του κινητού του σε αυτά τα δεδομένα. Επιπλέον, έχουν αναπτυχθεί αρκετές εφαρμογές που παρέχουν και κάποιες παραπάνω λειτουργίες από το να παρέχουν απλά τα δεδομένα. Για παράδειγμα το TripAdvisor, όχι μόνο παρέχει δεδομένα για τα σημεία ενδιαφέροντος μιας περιοχής, αλλά μπορεί κανείς να γνωρίζει ανά πάσα στιγμή τι υπάρχει κοντά του ανά κατηγορία. Αυτό σημαίνει ότι υπάρχει η δυνατότητα να βρει κανείς που είναι το πιο κοντινό εμπορικό κέντρο, μουσείο ή αξιοθέατο από την θέση που βρίσκεται εκείνη τη στιγμή. Επιπλέον, έχει και άλλες λειτουργίες, όπως οργάνωση ενός ταξιδιού και καταμερισμό των επιλογών στις μέρες του ταξιδιού.

Όσο όμως αυξάνονται οι επιλογές, ο προγραμματισμός του ταξιδιού γίνεται ακόμα πιο δύσκολος. Κάποια σημαντικά σημεία ενδιαφέροντος μονοπωλούν τις επιλογές των επισκεπτών και τα υπόλοιπα λίγοι τα επισκέπτονται επειδή βρίσκονται χαμηλά στις επιλογές. Η δυσκολία απόφασης αυξάνεται και ο κάθε χρήστης πρέπει να επιλέξει που θέλει να πάει με βάση τις πληροφορίες που έχει και παράλληλα να βρει τον τρόπο μετακίνησης. Αυτό γίνεται ακόμα πιο δύσκολο όταν ο χρόνος που έχει στη διάθεση του δεν είναι μέρες, αλλά ώρες. Πολλές φορές κάποιος βρίσκεται σε μια περιοχή και έχει κάποιο χρονικό διάστημα, μίας ή δύο ωρών για παράδειγμα, μέσα στο οποίο θα μπορούσε να επισκεφθεί ένα αξιοθέατο. Πιθανόν ένας τουρίστας να έχει τελειώσει πιο γρήγορα τις επισκέψεις του στα μουσεία που είχε προγραμματίσει και έχει ακόμα στη διάθεσή του 2 ώρες για να πάει σε όσα προλαβαίνει ακόμα. Αυτό είναι και το πρόβλημα, στο οποίο θα επικεντρωθεί και η παρούσα εργασία.

Αν και αυτές οι εφαρμογές που έχουν δημιουργηθεί μέχρι τώρα παρέχουν στους χρήστες ένα ευρύ φάσμα λειτουργιών, ώστε να οργανώσουν μόνοι τους τον χρόνο τους μέσα από τις επιλογές που έχουν, μετά από έρευνα δεν βρέθηκε καμία εφαρμογή που να οργανώνει αυτόματα τον χρόνο του χρήστη με περιηγήσεις σε σημεία ενδιαφέροντος. Επιπλέον σημαντικός παράγοντας είναι και τα χρήματα που διατίθεται κάποιος να ξοδέψει για αυτές τις περιηγήσεις. Επιπλέον απαραίτητος είναι και ο προσδιορισμός της θέσης του χρήστη.

Λύση σε αυτό το πρόβλημα θα μπορούσε να δοθεί με μια εφαρμογή για κινητά, η οποία θα προτείνει αυτόματα στον χρήστη σημεία ενδιαφέροντος, καθώς και την σειρά για να τα επισκεφθεί. Ο χρήστης το μόνο που θα πρέπει να κάνει είναι να καθορίσει το μέρος που βρίσκεται, τον χρόνο που έχει στη διάθεσή του και τα χρήματα που θέλει να διαθέσει για να επισκεφθεί σημεία ενδιαφέροντος. Επίσης θα μπορεί να επιλέξει τι είδους σημεία θέλει να επισκεφθεί, όπως για παράδειγμα μουσεία ή αγάλματα ή εμπορικά κέντρα. Με αυτόν τον τρόπο, ο κάθε χρήστης θα μπορεί να διαχειριστεί καλύτερα τον χρόνο και τα χρήματά του στην περιοχή που βρίσκεται ή που θα βρίσκεται, χωρίς να παίρνει ο ίδιος αυτές τις αποφάσεις και χωρίς να χρειάζεται να ψάξει πληροφορίες για κάθε αξιοθέατο.

Από την περιγραφή της εφαρμογής μπορεί να γίνει εύκολα κατανοητή η αγορά στην οποία στοχεύει αυτή η εφαρμογή. Αρχικά μπορεί να θεωρηθεί ότι θα είναι αρκετά χρήσιμη σε τουρίστες που επισκέπτονται για πρώτη φορά μια πόλη. Επιπλέον θα είναι αρκετά χρήσιμη σε ήδη κάτοικους της πόλης, οι οποίοι θέλουν να γνωρίσουν καλύτερα τη νοοτροπία της πόλης τους. Ωστόσο, η ομάδα ανθρώπων στην οποία στοχεύει αυτή η εφαρμογή είναι αυτοί που δεν έχουν την ικανότητα, την δυνατότητα ή την θέληση να ψάξουν και να οργανώσουν το ταξίδι τους. Είναι αρκετοί αυτοί, οι οποίοι δεν έχουν εξοικείωση με το διαδίκτυο και τους είναι αρκετά δύσκολο να ψάξουν για όλα τα σημεία ενδιαφέροντος μιας πόλης και να αποφασίσουν ποιο θέλουν να επισκεφθούν. Τέλος, σημαντική θα είναι και η βοήθεια προς αυτούς που θέλουν στα γρήγορα χωρίς αναζητήσεις και έρευνα να διαθέσουν λίγο χρόνο και χρήμα για να περιηγηθούν σε κάποια αξιοθέατα.

Από την περιγραφή της εφαρμογής και γνωρίζοντας πλέον που στοχεύει η εφαρμογή, μπορούν να γίνουν κατανοητά τα κύρια χαρακτηριστικά της, ώστε να είναι ανταγωνιστική προς τις υπόλοιπες. Αρχικά το πρώτο χαρακτηριστικό είναι ότι πρέπει να επιλύει ένα πρόβλημα μιας ομάδας ατόμων, να είναι δηλαδή **αποτελεσματική**. Αυτό θα επιτευχθεί μέσα από δοκιμές, ώστε να καταλήξει στην τελική της μορφή που θα μπορεί να προτείνει αυτόματα μια περιήγηση. Επιπλέον, θα πρέπει να είναι **απλή** και **εύκολη** στη χρήση της, αφού αυτός είναι και ένας λόγος που θα χρησιμοποιηθεί από ανθρώπους που δεν έχουν εξοικείωση με την τεχνολογία. Τέλος, η **ταχύτητα** θα διαδραματίσει σημαντικό ρόλο επίσης στην ανταγωνιστικότητα της εφαρμογής, αφού πρόκειται για εφαρμογή που θέλει να δώσει γρήγορα αποφάσεις.

Σύμφωνα με στοιχεία από την ιστοσελίδα www.kantarworldpanel.com, τα οποία παρουσιάζονται σε επόμενο κεφάλαιο, η αγορά των συσκευών Android υπερισχύει σε σύγκριση με τις υπόλοιπες. Όσον αφορά λοιπόν το τεχνικό κομμάτι της εφαρμογής θα πρέπει να επικεντρωθεί στην πλατφόρμα της Android, να στοχεύσει σε μια μεγαλύτερη δηλαδή αγορά. Επιπλέον θα ήταν προτιμότερο η εφαρμογή να μην απαιτεί την σύνδεση στο διαδίκτυο, καθώς κάποιος ο οποίος βρίσκεται ήδη σε ένα μέρος στην πόλη δεν έχει τη δυνατότητα συνήθως για σύνδεση στο διαδίκτυο, πόσο μάλλον όταν πρόκειται για επισκέπτη από άλλη χώρα, όπου τα δίκτυα κινητής τηλεφωνίας είναι διαφορετικά. Τέλος, αφού δεν θα απαιτείται σύνδεση στο διαδίκτυο, οι βαθμολογίες του κάθε χρήστη θα είναι ατομικές, με αποτέλεσμα να μην προτιμώνται συγκεκριμένα σημεία ενδιαφέροντος αφήνοντας τα υπόλοιπα ανεκμετάλλευτα.

Μέσα από την εργασία αυτή λοιπόν, θα αναλυθεί η διαδικασία από τον εντοπισμό του προβλήματος μέχρι το τελικό προϊόν, που θα είναι η προαναφερθείσα εφαρμογή. Αρχικά θα αναλυθούν κάποιες θεωρητικές έννοιες, που είναι απαραίτητες για την κατανόηση της όλης διαδικασίας και θα δικαιολογηθούν οι τεχνικές που θα χρησιμοποιηθούν και τα περιβάλλοντα εργασίας που χρησιμοποιήθηκαν. Έπειτα θα περιγραφεί αναλυτικά η μεθοδολογία που ακολουθήθηκε από την συλλογή των δεδομένων, την καταγραφή τους και μέχρι την επεξεργασία και οριστικοποίηση της εφαρμογής μέσα από κάποια στάδια. Θα δικαιολογηθεί η περιοχή μελέτης που επιλέχθηκε, θα κατηγοριοποιηθούν τα δεδομένα και θα αναλυθεί με λεπτομέρειες η διαδικασία ανάπτυξης της εφαρμογής με τρόπο απλό και κατανοητό ακόμα και από άτομα που δεν έχουν γνώσεις προγραμματισμού. Κλείνοντας, θα παρουσιαστεί η λειτουργία της εφαρμογής και μέσα από τα συμπεράσματα που προέκυψαν, θα προταθούν αλλαγές όχι μόνο στο τεχνικό κομμάτι της εργασίας, που αφορά την εφαρμογή αυτή καθαυτή, αλλά και στην μεθοδολογία που χρησιμοποιήθηκε.

ΘΕΩΡΗΤΙΚΕΣ ΈΝΝΟΙΕΣ

Απαραίτητο στοιχείο για την κατανόηση κάποιων διαδικασιών επομένων κεφαλαίων είναι η κατανόηση αρχικά κάποιων θεωρητικών εννοιών. Στο κεφάλαιο αυτό παρουσιάζονται κάποιες θεωρητικές έννοιες, διαδικασίες, λειτουργίες και στοιχεία της επεξεργασίας που αναλύεται στη συνέχεια, που πιθανών να είναι άγνωστες και είναι απαραίτητες για τη συνέχεια. Επιπλέον για κάποιες έννοιες, επειδή μπορούν να ερμηνευθούν με πολλούς τρόπους, γίνονται κάποιες παραδοχές ώστε να γίνει εύκολα κατανοητό το νόημα αυτών των εννοιών στην συνέχεια. Τέλος, παρουσιάζονται τα δύο περιβάλλοντα εργασίας που χρησιμοποιήθηκαν τόσο για την συλλογή και καταγραφή των δεδομένων, όσο και για την ανάπτυξη της τελικής εφαρμογής. Παρουσιάζονται όσο πιο αναλυτικά μπορούσαν τα λειτουργικά σημεία κυρίως της δεύτερης εφαρμογής που χρησιμοποιήθηκε, καθώς και η κύρια δομή και λειτουργία των εφαρμογών των κινητών.

Σαν πρώτη έννοια που έχει αναφερθεί και θα αναφερθεί πολύ στη συνέχεια είναι τα **σημεία ενδιαφέροντος**. Η εργασία αυτή έχει να κάνει με μέρη που μπορεί να επισκεφθεί ο οποιοσδήποτε στην Αθήνα. Ωστόσο αυτά τα μέρη περιλαμβάνουν πολλά είδη κατηγοριών. Τα μέρη αυτά περιλαμβάνουν μουσεία, επιχειρήσεις,

αγάλματα, πάρκα, εκπαιδευτικές σχολές, στάσεις, εμπορικά κέντρα, δημόσια κτήρια, στάδια, μνημεία και γενικά ό,τι άλλο μπορεί να προσελκύσει έναν επισκέπτη στην Αθήνα. Έπρεπε λοιπόν να βρεθεί μια ονομασία, η οποία θα περιγράφει με ακρίβεια όλα τα παραπάνω μέρη και θεωρήθηκε ως η πιο ακριβής το σημείο ενδιαφέροντος.

Σύμφωνα με την ιστοσελίδα wikipedia.org, ως **σημείο ενδιαφέροντος** (Point Of Interest ή POI), ορίζεται ένα συγκεκριμένο σημείο τοποθεσίας, το οποίο μπορεί να φανεί χρήσιμο ή ενδιαφέρον σε κάποιον. Οι περισσότεροι σήμερα χρησιμοποιούν τον όρο αυτό όταν αναφέρονται σε ξενοδοχεία, κατασκηνώσεις, βενζινάδικα ή άλλες κατηγορίες που χρησιμοποιούνται σε μοντέρνα συστήματα πλοήγησης κυρίως σε κινητές συσκευές. Τα σημεία αυτά για να οριστούν και να αναπαρασταθούν σε ένα χάρτη απαιτείται ο προσδιορισμός του γεωγραφικού πλάτους και μήκους. Πολλές φορές οι χρήστες μπορούν να δουν το όνομα του σημείου, μια μικρή περιγραφή ή ένα τηλέφωνο. Από τις προσδιορισμό των σημείων στο χάρτη μπορούν οι χρήστες να καταλάβουν ποια σημεία είναι κοντά τους εκείνη τη στιγμή. Επιπλέον παρέχονται πλέον και πληροφορίες για τις χρόνους παραμονής, ώρες λειτουργίας ή αν πρόκειται για εποχιακά σημεία ενδιαφέροντος (π.χ. θέρετρα σκι). Συνεπώς όπου στη συνέχεια αναφέρεται η έννοια σημείο ενδιαφέροντος εννοούνται όλα τα μέρη που περιγράφηκαν παραπάνω.

Ένας άλλος όρος που θα χρησιμοποιηθεί πολύ στην εργασία είναι ο URL. Σύμφωνα με την wikipedia ο όρος **URL** (Uniform Resource Locator ή Ενιαίος Εντοπιστής Πόρων) δηλώνει μια διεύθυνση ενός πόρου του Παγκόσμιου Ιστού. Είναι παρόμοιο με το όνομα ενός αρχείου, αλλά κρατάει και επιπλέον πληροφορία σχετικά με το όνομα του εξυπηρετητή, καθώς και το είδος του πρωτοκόλλου που αυτός χρησιμοποιεί. Οι ιστοσελίδες χρησιμοποιούν τα URLs για να συνδεθούν με άλλες σελίδες. Επίσης είναι πολύ συχνό να ψάχνεις ένα αρχείο στο διαδίκτυο με βάση την διεύθυνση URL. Μια συνηθισμένη διεύθυνση αναζήτησης αρχείου έχει την παρακάτω σύνταξη:

```
πρωτόκολλο://host.domain/διαδρομή/όνομα αρχείου
```

- **Πρωτόκολλο**: ορίζει το πρωτόκολλο μεταφοράς του αρχείου. Το πιο συνηθισμένο είναι το http (Hyper Text Transfer Protocol).
- **Host**: το καθορισμένο host για το πρωτόκολλο http είναι το www.
- **Domain**: ορίζει το όνομα του Domain Name, πχ. google.gr, wikipedia.org.
- **Διαδρομή**: ορίζει την διαδρομή που βρίσκεται το αρχείο στον server.
- **Όνομα αρχείου**: ορίζει το όνομα αρχείου.

Τα URL θα χρησιμοποιηθούν πολύ κατά την διάρκεια της εργασίας σε δύο κυρίως περιπτώσεις. Αρχικά στα περισσότερα σημεία ενδιαφέροντος θα προστεθούν διευθύνσεις URL, οι οποίες θα περιέχουν την σελίδα που περιέχει παραπάνω πληροφορίες για κάθε σημείο. Για παράδειγμα στο μουσείο της Ακρόπολης θα αποθηκευτεί και η ιστοσελίδα με τις πληροφορίες για το μουσείο. Επιπλέον τα URL θα χρησιμοποιηθούν και για την απόκτηση δεδομένων από την Google. Η Google παρέχει στους προγραμματιστές που επιθυμούν να δημιουργήσουν μια εφαρμογή έτοιμα εργαλεία που μπορούν να χρησιμοποιήσουν για πολλές λειτουργίες. Μερικές από αυτές είναι η εμφάνιση ενός χάρτη σε μια εφαρμογή, ο υπολογισμός ενός χρόνου μετάβασης μεταξύ δύο σημείων στο χάρτη και πολλά άλλα. Όλες αυτές τις λειτουργίες τις έχει διαθέσει η Google μέσω δικών της εφαρμογών και παράλληλα δίνει την δυνατότητα στους προγραμματιστές να χρησιμοποιήσουν τα δεδομένα της με κάποιους περιορισμούς φυσικά. Για την χρήση τέτοιων δεδομένων είναι απαραίτητη η δημιουργία URL. Στο URL ορίζεται κάθε φορά ποιο εργαλείο θα χρησιμοποιηθεί και οι παράμετροι που είναι απαραίτητοι για την απαίτηση των ζητούμενων δεδομένων. Για παράδειγμα, αν ζητηθεί από την Google η απόσταση μεταξύ δύο σημείων σε μια πόλη σε χιλιόμετρα, στο URL θα πρέπει να αναφερθεί ότι θα χρησιμοποιηθεί το εργαλείο υπολογισμού απόστασης δύο σημείων, να αναφερθούν οι συντεταγμένες των δύο σημείων ενώ μπορεί να ζητηθούν παράμετροι

όπως να υπολογιστεί η κοντινότερη απόσταση. Όλα αυτά αναφέρονται στις αντίστοιχες ιστοσελίδες που παρέχει η Google και διατυπώνονται απλά.

Όταν λοιπόν, ζητείται μια απόσταση από την Google ή οποιοδήποτε δεδομένο, ουσιαστικά ακολουθείται μια συγκεκριμένη διαδικασία. Δημιουργείται το κατάλληλο URL που περιέχει όλες τις μεταβλητές και τα στοιχεία που υποδηλώνουν τι δεδομένα ζητάει ο προγραμματιστής από την Google και εκτελεί μια εντολή λήψης δεδομένων με βάση αυτή τη διεύθυνση. Το αποτέλεσμα της λήψης συνήθως είναι ένα αρχείο μορφής json, το οποίο περιέχει όλα τα ζητούμενα δεδομένα.

Στους υπολογιστές σύμφωνα και πάλι με την wikipedia, το **JSON** (JavaScript Object Notation) είναι ένα αρχείο ανοιχτής τυποποιημένης μορφής, που χρησιμοποιεί κείμενο εύκολα αναγνώσιμο από τον άνθρωπο για να μεταβιβάσει δεδομένα. Τα αρχεία αυτά χρησιμοποιούνται πλέον ευρέως σαν απάντηση από κάποιον server σε περιπτώσεις όπου του ζητούνται δεδομένα, αντικαθιστώντας τα αρχεία xml. Τα αρχεία JSON χρησιμοποιούν σαν επέκταση το *.json. Τα δεδομένα που περιέχονται στο αρχείο αυτό αποθηκεύονται συνήθως είτε σε ζευγάρι της μορφής «ιδιότητα: τιμή» είτε σε μορφή γραμμής που περιέχει άλλα δεδομένα. Ένα παράδειγμα αρχείου JSON φαίνεται στο Απόσπασμα 1.

```
{
  "status": "OK",
  "origin_addresses": [ "Vancouver, BC, Canada" ],
  "destination_addresses": [ "San Francisco, Californie, États-Unis" ],
  "rows": [
    "elements": [
      "status": "OK",
      "duration": {
        "value": 340110,
        "text": "3 hours 22 minutes"
      },
      "distance": {
        "value": 1734542,
        "text": "1 735 km"
      }
    ]
  ]
}
```

Απόσπασμα 1: Παράδειγμα μορφής αρχείου JSON

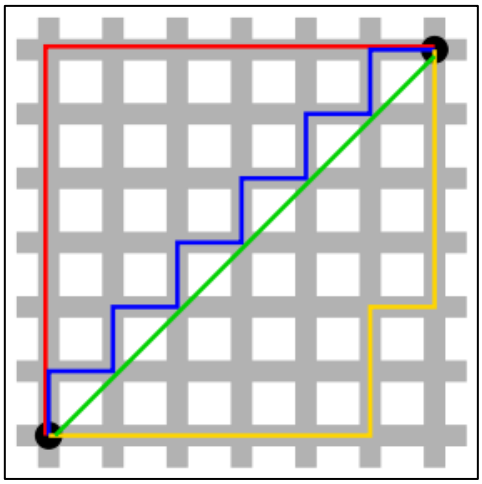
Γενικά τα αρχεία JSON είναι ανεξάρτητα από το ποια γλώσσα προγραμματισμού θα επιλεχθεί. Αρχικά χρησιμοποιήθηκαν από την JavaScript, αλλά από το 2017 πολλές γλώσσες προγραμματισμού συμπεριέλαβαν κώδικα για την δημιουργία και μεταφορά δεδομένων μέσω αρχείων JSON. Στην παρούσα εργασία χρησιμοποιήθηκε αυτός ο τύπος των αρχείων κυρίως για να ζητηθούν δεδομένα αποστάσεων από την Google. Δεν θα περιγραφεί αναλυτικά ο τρόπος εξαγωγής των δεδομένων από αυτό το αρχείο, ωστόσο υπάρχει απόσπασμα κώδικα στο Παράρτημα της εργασίας με την αναλυτική διαδικασία.

Όπως είναι προφανές μέχρι τώρα, σαν δεδομένα της εργασίας αυτής θα χρησιμοποιηθούν ήδη υπάρχοντα σημεία ενδιαφέροντος με σταθερές γεωγραφικές συντεταγμένες. Πολλές φορές θα χρειαστεί η απόσταση μεταξύ των σημείων αυτών ή γενικά η απόσταση μεταξύ δύο οποιονδήποτε σημείων πάνω στο χάρτη. Τα δεδομένα αυτά παρέχονται από την Google, ωστόσο υπάρχει κάποιος περιορισμός και επιπλέον δεν μπορούν να υπολογιστούν για μεγάλες αποστάσεις. Για τον λόγο αυτό έπρεπε να βρεθεί ένας τύπος υπολογισμού αποστάσεων δύο σημείων με γνωστές τις γεωγραφικές τους συντεταγμένες. Ο υπολογισμός της απόστασης είναι συνήθως το πιο βασικό βήμα σε πολλά χωρικά ζητήματα. Συνήθως οι κλασικές αποστάσεις δεν οδηγούν σε ακριβή αποτελέσματα. Για τον λόγο αυτό για κάθε πρόβλημα πρέπει να επιλέγεται η κατάλληλη απόσταση. Οι αποστάσεις που μπορούσαν να χρησιμοποιηθούν στην παρούσα εργασία ήταν δύο: η Ευκλείδεια απόσταση και η Manhattan απόσταση.

Όσον αφορά την Ευκλείδεια απόσταση αυτή είναι η πιο συνηθισμένη και προκύπτει από το μήκος της ευθείας γραμμής που συνδέει τα δύο σημεία. Ωστόσο,

πρέπει να θεωρηθεί δεδομένο ότι τα σημεία αυτά βρίσκονται μέσα σε πόλη, όπου το οδικό δίκτυο δεν συνδέει σε ευθεία γραμμή όλα τα πιθανά σημεία. Στην παρούσα περίπτωση η απόσταση που μας ενδιαφέρει είναι η απόσταση μέσω του οδικού δικτύου, καθώς η μετάβαση από κάθε σημείο προς τα υπόλοιπα θα γίνεται μέσω αυτού. Για τον λόγο αυτό υπάρχει και η **Manhattan απόσταση**.

Το σύστημα δόμησης της πόλης βασίζεται στην Γεωμετρία ταξί (Taxicab Geometry). Η γεωμετρία ταξί είναι μια μορφή γεωμετρίας στην οποία η συνήθης μέτρηση της Ευκλείδειας γεωμετρίας αντικαθίσταται από μια νέα μέτρηση στην οποία η απόσταση μεταξύ δύο σημείων είναι το άθροισμα των (απόλυτων) διαφορών των συντεταγμένων τους. Η Manhattan απόσταση (Manhattan Distance ή Rectilinear Distance, City Block Distance) ορίζεται ως το άθροισμα των μηκών των προβολών των τμημάτων της γραμμής μεταξύ των σημείων πάνω στους άξονες συντεταγμένων. Για την καλύτερη κατανόηση των δύο αποστάσεων φαίνεται στην Εικόνα 1 τα δύο είδη αποστάσεων. Με την πράσινη γραμμή φαίνεται η ευθεία, το μήκος της οποίας ορίζει την Ευκλείδεια απόσταση. Η κόκκινη γραμμή ορίζει την Manhattan απόσταση, ενώ οι υπόλοιπες ουσιαστικά είναι εναλλακτικές ευθείες για τον υπολογισμό της Manhattan απόστασης και ισούνται με την κόκκινη.



Εικόνα 1: Διαφορά Ευκλείδειας - Manhattan απόστασης

Η Manhattan απόσταση ουσιαστικά προκύπτει από το άθροισμα των δύο τμημάτων της κόκκινης γραμμής. Επομένως μπορεί να θεωρηθεί και το άθροισμα δύο Ευκλείδειων αποστάσεων μεταξύ τριών σημείων. Στην συγκεκριμένη εργασία χρησιμοποιήθηκε όποτε ήταν αναγκαίο η Manhattan απόσταση, αφού τα σημεία ήταν μέσα σε πόλη. Πιο αναλυτικά επομένως και όπως φαίνεται και στην παραπάνω εικόνα, γνωρίζοντας τις συντεταγμένες δύο σημείων μπορεί να υπολογιστεί η Manhattan απόσταση μεταξύ τους. Αν d είναι η απόσταση δύο σημείων 1 και 2 πάνω στο επίπεδο και (x,y) οι συντεταγμένες τους, τότε ο τύπος που ορίζει την Manhattan απόσταση μεταξύ των δύο σημείων δίνεται από τον τύπο:

$$d = |x_1 - x_2| + |y_1 - y_2|$$

Ωστόσο, επειδή τα σημεία αυτά δεν βρίσκονται πάνω σε επίπεδο, αλλά στη γη που θεωρητικά σε επίπεδο πόλης μπορεί να θεωρηθεί και σφαίρα, για τον υπολογισμό των αποστάσεων χρησιμοποιήθηκε άλλη διαδικασία υπολογισμού της Manhattan απόστασης. Χωρίστηκε η απόσταση αυτή σε δύο Ευκλείδειες αποστάσεις και το άθροισμά τους έδωσε την ζητούμενη απόσταση. Αναλυτικότερα, όπως φαίνεται και από την εικόνα υπάρχει άλλο ένα σημείο το οποίο έχει το ίδιο x με το ένα σημείο και το ίδιο y με το άλλο. Το άθροισμα των αποστάσεων των δύο σημείων προς αυτό το σημείο μας δίνει την Manhattan απόσταση. Αυτές οι αποστάσεις ξεχωριστά προκύπτουν με την παρακάτω διαδικασία για την επιφάνεια της γης. Αν (x1, y1) και (x2, y2) οι γεωγραφικές συντεταγμένες δύο σημείων σε rad και R η ακτίνα

της γης σε km τότε η διαδικασία για τον υπολογισμό της Ευκλείδειας απόστασης d αυτών των δύο σημείων είναι:

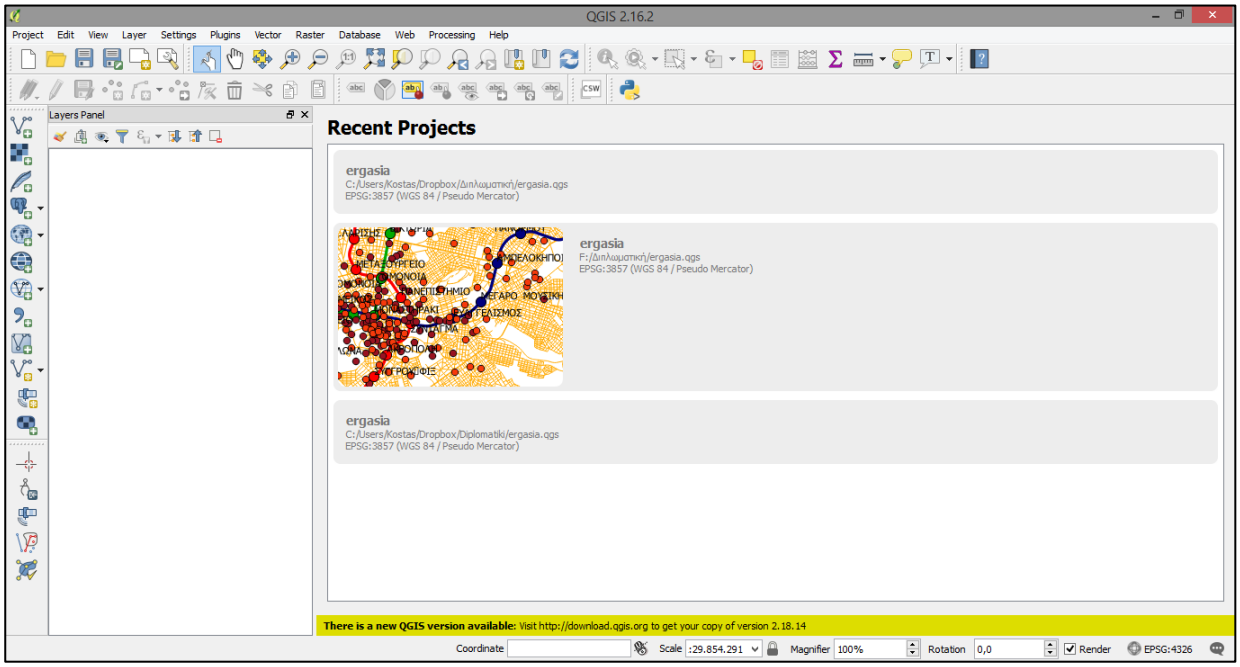
$$\boldsymbol{d_x} = x_2 - x_1$$
$$\boldsymbol{d_y} = y_2 - y_1$$
$$\boldsymbol{a} = \sin\left(\frac{d_x}{2}\right) \cdot \sin\left(\frac{d_x}{2}\right) + \cos\left(x_1\right) \cdot \cos\left(x_2\right) \cdot \sin\left(\frac{d_y}{2}\right) \cdot \sin\left(\frac{d_y}{2}\right)$$
$$\boldsymbol{c} = 2 \cdot arctan2\left(\sqrt{a}, \sqrt{(1-a)}\right)$$
$$\boldsymbol{d} = R \cdot c$$

Υπολογίζοντας με αυτόν τον τρόπο τις αποστάσεις από τα δύο ζητούμενα σημεία προς το καινούριο νοητό σημείο και προσθέτοντας αυτές τις αποστάσεις προκύπτει η ζητούμενη απόσταση. Η αναλυτική διαδικασία όπως πραγματοποιήθηκε προγραμματιστικά φαίνεται στο Παράρτημα.

Περιβάλλον εργασίας

Κατά την διάρκεια καταγραφής της εργασίας χρησιμοποιήθηκαν δύο προγράμματα. Το πρώτο ήταν το QGIS Desktop και χρησιμοποιήθηκε κατά την διάρκεια συλλογής και καταγραφής των δεδομένων και το δεύτερο ήταν το Android Studio, με το οποίο δημιουργήθηκε η εφαρμογή σε κινητά, δηλαδή το τελικό προϊόν της εργασίας.

Όσον αφορά το πρώτο, το QGIS Desktop, παλαιότερα γνωστό ως (Quantom GIS), είναι ένα δωρεάν λογισμικό ανοιχτού κώδικα και πολλαπλών πλατφορμών (open-source cross-platform application)[1] γεωγραφικών πληροφοριών για υπολογιστές, που υποστηρίζει την προβολή, επεξεργασία και ανάλυση γεωχωρικών δεδομένων (Εικόνα 2).



Εικόνα 2: Περιβάλλον εργασίας (QGIS Desktop)

Το QGIS λειτουργεί ως λογισμικό συστήματος γεωγραφικών πληροφοριών (GIS), επιτρέποντας στους χρήστες να αναλύουν και να επεξεργάζονται χωρικές

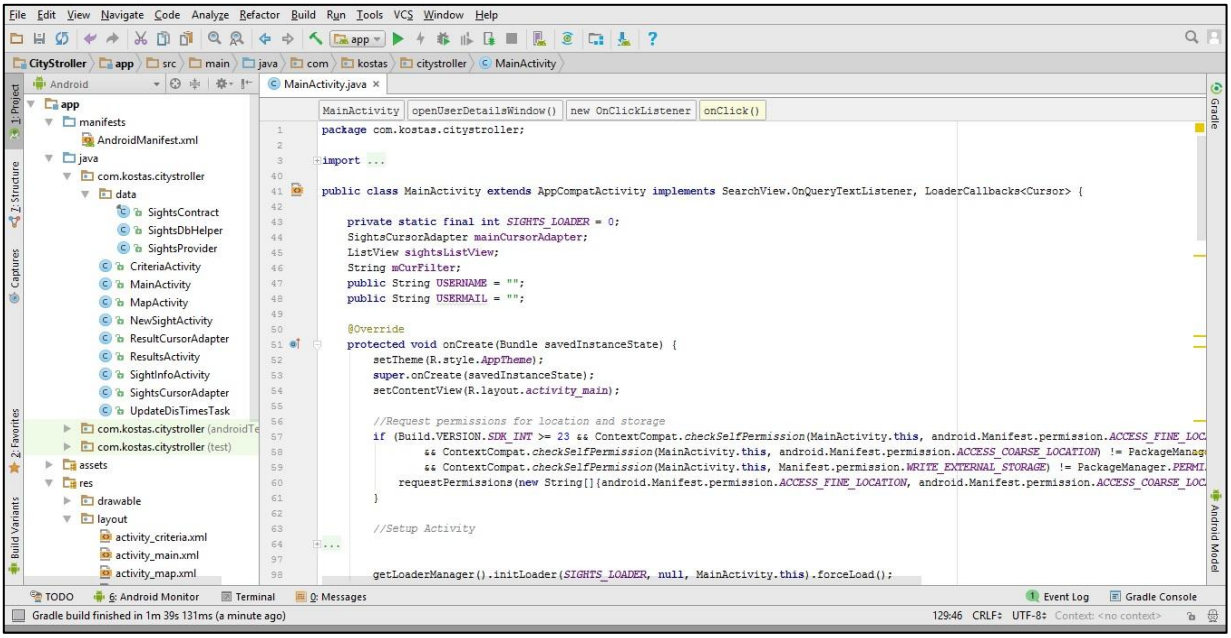
[1] Στους υπολογιστές το λογισμικό πολλαπλών πλατφορμών (cross-platform ή multi-platform ή platform-independent software) είναι λογισμικό υπολογιστή που υλοποιείται σε πολλαπλές πλατφόρμες υπολογιστών. Για παράδειγμα μπορεί να λειτουργήσει σε Microsoft Windows, Linux ή MacOS.

πληροφορίες, εκτός από τη σύνθεση και την εξαγωγή γραφικών χαρτών. Το QGIS υποστηρίζει τόσο τα στρώματα raster όσο και διανύσματα. Τα δεδομένα διανυσμάτων αποθηκεύονται ως χαρακτηριστικά σημείων, γραμμών ή πολυγώνων. Υποστηρίζονται πολλαπλές μορφές εικόνων ράστερ και το λογισμικό μπορεί να προβάλει εικόνες γεωαναφοράς.

Το QGIS υποστηρίζει shapefiles , coverages , προσωπικά geodatabases, dxf , MapInfo, PostGIS και άλλες μορφές. Υποστηρίζονται επίσης υπηρεσίες Web, συμπεριλαμβανομένης της υπηρεσίας Web Map Service και της υπηρεσίας Web Feature, για να επιτρέπεται η χρήση δεδομένων από εξωτερικές πηγές.

Το QGIS ενσωματώνεται σε άλλα πακέτα GIS ανοιχτού κώδικα, συμπεριλαμβανομένων των PostGIS , GRASS GIS και MapServer. Οι προσθήκες γραμμένες σε Python ή C ++ επεκτείνουν τις δυνατότητες του QGIS. Οι προσθήκες μπορούν να πραγματοποιήσουν γεωκωδικοποίηση χρησιμοποιώντας το API Google Geocoding, να εκτελούν λειτουργίες γεωφωτογράφησης, οι οποίες είναι παρόμοιες με τα τυπικά εργαλεία που εντοπίζονται στο ArcGIS και να αλληλεπιδρούν με τις βάσεις δεδομένων PostgreSQL,/PostGIS, SpatiaLite και MySQL.

Όσον αφορά το δεύτερο περιβάλλον εργασίας που χρησιμοποιήθηκε, στο οποίο δημιουργήθηκε η συγκεκριμένη εφαρμογή είναι το Android Studio (Εικόνα 3). Το Android Studio είναι ένα ολοκληρωμένο προγραμματιστικό περιβάλλον (IDE) για ανάπτυξη εφαρμογών στην πλατφόρμα Android και είναι διαθέσιμο ελεύθερα με την άδεια Apache License 2.0.



Εικόνα 3: Περιβάλλον εργασίας (Android Studio)

Το Android Studio ήταν διαθέσιμο σε πρώιμο στάδιο για προεπισκόπηση τον Μάιο του 2013 και ξεκίνησε το δοκιμαστικό στάδιο τον Ιούνιο του 2014. Η πρώτη σταθερή έκδοση βγήκε το Δεκέμβριο του 2014. Βασισμένο στο λογισμικό της JetBrains' IntelliJ IDEA, σχεδιάστηκε αποκλειστικά για προγραμματισμό Android. Είναι διαθέσιμο για Windows, Mac OS X και Linux και αντικατέστησε τα Eclipse Android Development Tools (ADT) ως το κύριο IDE της Google για ανάπτυξη εφαρμογών Android.

Η γλώσσα προγραμματισμού που χρησιμοποιεί είναι η Java, με εργαλεία όμως που καλύπτουν και εντολές της SQL, που χρησιμοποιήθηκαν αρκετά στην παρούσα εφαρμογή. Οι εντολές SQL έχουν να κάνουν με την δημιουργία μιας βάσης δεδομένων στο κινητό του χρήστη, τα δεδομένα της οποίας μπορεί να χρησιμοποιήσει ο χρήστης. Μέσα σε αυτή τη βάση δεδομένων μπορούν να δημιουργηθούν πολλοί πίνακες ανάλογα με τα σχέδια του προγραμματιστή. Ανάλογα με κάθε εντολή ορίζεται ο τρόπος που θα χρησιμοποιηθεί και η βάση δεδομένων (δηλαδή αν θα είναι αναγνώσιμη ή επεξεργάσιμη). Στην συγκεκριμένη περίπτωση δημιουργήθηκε μια βάση δεδομένων με δύο πίνακες, που θα περιγραφούν αναλυτικά

σε επόμενο κεφάλαιο. Μέσα σε κάθε πίνακα ορίζεται από την αρχή τι στήλες θα έχει και τι τύπους δεδομένων θα έχει η κάθε στήλη. Οι τύποι των δεδομένων που μπορούν να χρησιμοποιηθούν είναι:

- **INTEGER**, δηλαδή ακέραιοι αριθμοί.
- **REAL**, δηλαδή πραγματικοί αριθμοί.
- **TEXT**, δηλαδή κείμενο.

Οι παραπάνω τύποι δεδομένων εισάγονται στις κατάλληλες εντολές SQL. Οι κύριες εντολές SQL που χρησιμοποιούνται κατά κύριο λόγο σε μια βάση δεδομένων είναι οι ακόλουθες:

- **CREATE TABLE**: Δημιουργία πίνακα στη βάση δεδομένων. Με τη συγκεκριμένη εντολή επιλέγεται το όνομα της βάσης δεδομένων που θα δημιουργηθεί ο πίνακας και επιλέγεται αυτή η βάση δεδομένων να είναι επεξεργάσιμη. Στην συνέχεια καταγράφεται το όνομα του πίνακα, καθώς και τα ονόματα της κάθε στήλης του πίνακα μαζί με τον τύπο των δεδομένων που θα εισάγονται σε αυτή. Επιπλέον για κάθε πίνακα πρέπει να οριστεί μια στήλη η οποία θα περιέχει μοναδικές τιμές για κάθε εγγραφή και θα τις διαφοροποιούν μεταξύ τους. Συνήθως αυτή η στήλη είναι η στήλη με τους κωδικούς id και επισημαίνεται με τις λέξεις «PRIMARY KEY». Επιπλέον για μια στήλη μπορεί να οριστεί να προσθέτονται αυτόματα τα δεδομένα. Αυτό μπορεί να συμβεί αν για παράδειγμα υποτεθεί ο τύπος της στήλης «INTEGER», δηλαδή ακέραιος αριθμός και προστεθεί η λέξη «AUTOINCREMENT». Με τον τρόπο αυτό σε κάθε εγγραφή θα συμπληρώνεται αυτόματα αυτή η στήλη με τον επόμενο ακέραιο σε σειρά. Με βάση αυτές τις δύο ιδιότητες συνήθως δημιουργείται μια στήλη αυτόματης δημιουργίας μοναδικών κωδικών για κάθε εγγραφή. Επιπλέον αν κάποια στήλη πρέπει οπωσδήποτε να περιέχει δεδομένα και πρέπει να αποφευχθεί να μείνει κενή, μπορεί να χαρακτηριστεί ως «NOT NULL». Τέλος, η τιμή μιας στήλης μιας εγγραφής μπορεί να έχει μια προεπιλεγμένη τιμή. Προσθέτοντας δίπλα στον τύπο της στήλης την λέξη «DEFAULT» και την τιμή, η στήλη παίρνει πλέον την προεπιλεγμένη τιμή όταν δεν υπάρχουν δεδομένα για αυτή τη στήλη. Στη συνέχεια παρουσιάζεται ένα παράδειγμα δημιουργίας πίνακα (με όνομα persons) με τα στοιχεία ατόμων, που χρησιμοποιούνται όλα τα παραπάνω.

  ```
  CREATE TABLE persons (PersonID INTEGER PRIMARY KEY AUTOINCREMENT, Name TEXT NOT NULL, Weight REAL DEFAULT 70.0, PhoneNumber INTEGER);
  ```

- **INSERT**: Προσθήκη εγγραφής σε ένα πίνακα. Για να χρησιμοποιηθεί η συγκεκριμένη εντολή πρέπει να έχει οριστεί η βάση δεδομένων σαν επεξεργάσιμη. Για την χρήση αυτής της εντολής πρέπει να γίνει αναφορά σε ποιον πίνακα θα γίνει η προσθήκη και τι τιμές θα προστεθούν σε ποιες στήλες. Με την συγκεκριμένη εντολή προστίθεται μια εγγραφή στον αναφερόμενο πίνακα και συμπληρώνεται μια γραμμή με τα στοιχεία ενός προσώπου για την κάθε στήλη. Ένα παράδειγμα αυτής της εντολής παρουσιάζεται παρακάτω.

  ```
  INSERT INTO persons (PersonId, Name, Weight, PhoneNumber) VALUES (1, Kostas, 72.4, 6970000000);
  ```

- **SELECT** (για Android QUERY): Αναζήτηση στοιχείων σε έναν πίνακα. Ως προεπιλεγμένη εντολή για το Android Studio ορίζεται η «QUERY». Με την εντολή αυτή δημιουργείται ένας νέος υποπίνακας με βάση κάποια κριτήρια που ορίζονται στην συνθήκη. Η εντολή αυτή χρησιμοποιείται για να γίνει

αναζήτηση σε ένα πίνακα, να μπουν φίλτρα σε έναν πίνακα ώστε να εμφανίζονται συγκεκριμένες εγγραφές ή για να εμφανιστεί όλος ο πίνακας. Είναι προφανές ότι εφόσον δεν γίνονται αλλαγές στον πίνακα, αρκεί η βάση δεδομένων να είναι απλά για ανάγνωση. Η εντολή αυτή μπορεί να χρησιμοποιηθεί για να επιλέξει κανείς όλον τον πίνακα και να τον εμφανίσει στην οθόνη του χρήστη για παράδειγμα ή να επιλέξει μια συγκεκριμένη εγγραφή από έναν πίνακα. Οι δύο αυτές συνθήκες φαίνονται με παραδείγματα παρακάτω.

```
SELECT * FROM persons;
```
(Εμφανίζονται όλες οι εγγραφές του πίνακα persons)

```
SELECT Name FROM persons WHERE Weight>70;
```
(Εμφανίζεται η στήλη Name για τα άτομα που το βάρος είναι μεγαλύτερο από 70)

- **UPDATE**: Ενημερώνει μια ή πολλές εγγραφές ενός πίνακα. Η εντολή αυτή προφανώς μπορεί να χρησιμοποιηθεί για να αλλάξει τα δεδομένα είτε ολόκληρου του πίνακα είτε μιας εγγραφής ενός πίνακα. Επειδή αλλάζει τα δεδομένα, η βάση δεδομένων αρχικά πρέπει να θεωρηθεί επεξεργάσιμη. Στην εντολή αυτή ορίζεται ποιος πίνακας θα ενημερωθεί και αν πρόκειται για την ολική ενημέρωσή του ή την ενημέρωση μιας εγγραφής ή μιας στήλης με βάση κάποια κριτήρια. Παραδείγματα ενημέρωσης φαίνονται παρακάτω.

```
UPDATE persons
SET Name = 'Kostas Apostolakis', Weight = 70
```
(Ενημέρωση της στήλης Name και Weight για όλο τον πίνακα)

```
UPDATE persons
SET Name='Konstantinos'
WHERE PersonId = 1;
```
(Ενημέρωση της στήλης Name για την εγγραφή με PersonId = 1)

- **DELETE**: Διαγράφει μια ή πολλές εγγραφές από έναν πίνακα. Η εντολή αυτή μπορεί να διαγράψει μια ή παραπάνω εγγραφές επιλεγμένες με κάποια κριτήρια τα οποία ορίζονται στη συνθήκη. Για να πραγματοποιηθεί αυτή η εντολή πρέπει η βάση δεδομένων να γίνει επεξεργάσιμη. Επιπλέον μπορούν να διαγραφούν και όλες οι εγγραφές από τον πίνακα. Πρέπει ωστόσο να σημειωθεί ότι με αυτή την εντολή δεν διαγράφεται ο πίνακας από την βάση δεδομένων, αλλά μόνο οι εγγραφές από τον πίνακα και ο πίνακας μένει άδειος. Παραδείγματα διαγραφής όλων των εγγραφών του πίνακα και μιας συγκεκριμένης εγγραφής φαίνονται παρακάτω.

```
DELETE * FROM persons;
```
(Διαγραφή ολόκληρου του πίνακα persons)

```
DELETE FROM persons
WHERE PersonId = 1;
```
(Διαγραφή της εγγραφής με PersonId = 1)

- **DROP TABLE**: Διαγραφή ενός πίνακα από τη βάση δεδομένων. Με την εντολή αυτή διαγράφεται ένας ολόκληρος πίνακας από μια βάση δεδομένων. Για να εκτελεστεί η εντολή αυτή, θα πρέπει η βάση δεδομένων να είναι επεξεργάσιμη. Αρχικά υπήρχε η πρόθεση να διαγραφεί μια στήλη ενός πίνακα (με την εντολή «ALTER TABLE»), αλλά λόγω αδυναμίας

αυτής της ενέργειας από το Android Studio, είχε επιλεχθεί να διαγραφεί όλος ο πίνακας και να δημιουργηθεί ξανά. Ωστόσο, τελικά δεν χρησιμοποιήθηκε αυτή η πρακτική, καθώς δεν επηρεαζόταν το αποτέλεσμα και δεν ήταν απαραίτητη η διαγραφή και δημιουργία νέου πίνακα ξανά. Παράδειγμα χρήσης αυτής της εντολής παρουσιάζεται στη συνέχεια.

```
DROP TABLE persons;
```

Με βάση αυτές τις εντολές μπορεί ο προγραμματιστής να δημιουργήσει, να επεξεργαστεί ή να διαγράψει δεδομένα σε μια βάση δεδομένων στην συσκευή του χρήστη. Τα δεδομένα αυτά προφανώς δεν χρειάζονται Internet για να ανακτηθούν καθώς δημιουργούνται τοπικά. Όλα αυτά καθώς και οι υπόλοιπες λειτουργίες καταγράφονται σε κώδικα σε αρχεία με επέκταση *.java. Τα αρχεία αυτά αποτελούν το κυρίως μέρος της εφαρμογής και μέσω του κώδικα που είναι καταγεγραμμένος εκεί λειτουργεί η εφαρμογή.

Ξεκινώντας λοιπόν το Android Studio, αρχικά δημιουργείται ένα project στο οποίο δίνεται όνομα, διαδρομή, η έκδοση του Android που θα χρησιμοποιηθεί (συνεπώς και σε ποιες συσκευές θα λειτουργεί η εφαρμογή) και άλλα στοιχεία που το καθορίζουν μοναδικό. Μέσα σε αυτό το project δημιουργούνται **Αρχεία δραστηριοτήτων «Activities» (*.java)** και **Αρχεία «Layouts» (*.xml)**. Στα πρώτα περιέχεται το κυρίως μέρος του κώδικα όπως προαναφέρθηκε, ενώ στα δεύτερα περιέχεται το οπτικό μέρος των «δραστηριοτήτων». Αυτό σημαίνει ότι κάθε οθόνη που εμφανίζεται σε μια εφαρμογή και έχει διαφορετική μορφή αποτελεί μια διαφορετική δραστηριότητα. Όταν για παράδειγμα ανοίγει κανείς μια εφαρμογή και βλέπει μια λίστα αυτή είναι μια δραστηριότητα. Όταν πατήσει σε ένα από τα στοιχεία της λίστας και πάει σε μια άλλη οθόνη τότε αυτή είναι μια διαφορετική δραστηριότητα. Για παράδειγμα στο City Stroller, δημιουργήθηκαν 6 Δραστηριότητες (Activities):

1) Η κύρια δραστηριότητα (**Main Activity**), που περιέχει την κυρίως λίστα που εμφανίζεται στην εκκίνηση της εφαρμογής,
2) η δραστηριότητα του χάρτη (**Map Activity**), που εμφανίζει τον χάρτη με τα σημεία ενδιαφέροντος,
3) η δραστηριότητα προσθήκης νέου σημείου (**New Sight Activity**), από την οποία ο χρήστης μπορεί να προσθέσει νέο σημείο,
4) η δραστηριότητα προβολής των πληροφοριών ενός σημείου (**Sight Info Activity**),
5) η δραστηριότητα εισαγωγής των κριτηρίων (**Criteria Activity**),
6) η δραστηριότητα εμφάνισης των αποτελεσμάτων (**Results Activity**) και
7) η δραστηριότητα εμφάνισης οδηγιών (**Instructions Activity**).

Σημαντικό επίσης θεωρείται ότι από κάθε δραστηριότητα προς οποιαδήποτε άλλη υπάρχει η δυνατότητα να περάσουν κάποιες μεταβλητές. Αυτή η δυνατότητα υπάρχει με την δημιουργία ενός αντικειμένου, που λέγεται **«Bundle»**. Δημιουργώντας ένα αντικείμενο «Bundle», μπορεί ο προγραμματιστής να τοποθετήσει μέσα όσες μεταβλητές θέλει μαζί με τις τιμές τους και να «μεταβιβάσει» ουσιαστικά αυτό το αντικείμενο από μια δραστηριότητα σε μια άλλη. Έτσι μπορούν οι δραστηριότητες να συνδεθούν μεταξύ τους. Για παράδειγμα επιλέγοντας ένα στοιχείο μιας λίστας σε μια δραστηριότητας μπορεί να αποθηκεύεται το όνομα του στοιχείου σε ένα αντικείμενο και μεταβιβάζοντας αυτό το αντικείμενο στην επόμενη δραστηριότητα, να εντοπίζεται το όνομα και να προβάλλονται οι κατάλληλες πληροφορίες.

Όπως αναφέρεται μέχρι τώρα κάθε δραστηριότητα έχει μια διαφορετική δομή στην οθόνη και περιέχει διάφορες λειτουργίες, οι οποίες καθορίζονται από τα αντίστοιχα «layouts». Τα «layouts» ουσιαστικά περιέχουν το οπτικό τμήμα της κάθε δραστηριότητας. Έτσι μέσα στον κώδικα δηλώνεται ποιο «layout» θα εφαρμοστεί σε κάθε δραστηριότητα και εφαρμόζεται. Το ίδιο συμβαίνει και με τα μενού της κάθε δραστηριότητας. Τα αρχεία του κώδικα, τα «layouts» και τα αρχεία των μενού

αποθηκεύονται σε διαφορετικούς φακέλους του project και επεξεργάζονται ξεχωριστά. Με αυτόν τον τρόπο λειτουργεί το Android Studio, έτσι σχεδιάστηκε και η παρούσα εφαρμογή.

Τα «layouts» χρησιμοποιούν πολλά εργαλεία που μπορούν να προβληθούν στην οθόνη του χρήστη. Κάποια από αυτά που χρησιμοποιήθηκαν στην παρούσα εφαρμογή είναι πλαίσιο κειμένου, εικόνα, κουμπί (με κείμενο, εικονίδιο ή εικόνα), κουτί επιλογής, διακόπτης ON/OFF, αστέρια βαθμολόγησης, αναπτυσσόμενη λίστα, ένδειξη φόρτωσης της σελίδας, πεδίο εισαγωγής κειμένου ή ένας χάρτης. Όλα αυτά μπορούν να δομηθούν όπως τα ορίσει ο προγραμματιστής. Ωστόσο, ένα ακόμα εργαλείο που χρησιμοποιήθηκε πολύ στην παρούσα εφαρμογή είναι η **λίστα**.

Η χρήση της λίστας είναι σχετικά απλή. Ουσιαστικά χρησιμοποιείται μια λίστα σε ένα «layout» και ορίζεται σε διαφορετικό «layout» πως θα είναι το κάθε στοιχείο της λίστας. Για παράδειγμα μπορεί το κάθε στοιχείο να περιέχει μια εικόνα του σημείου, το όνομα και την διεύθυνση. Στη συνέχεια δημιουργείται ένα άλλο αρχείο «**ADAPTER**», στο οποίο καταγράφεται κώδικας που εντοπίζει τα δεδομένα από εκεί που του ορίζει ο προγραμματιστής και τα τοποθετεί με βάση τα δύο «layouts» σε μια λίστα στην σωστή δραστηριότητα. Ωστόσο αν τα δεδομένα είναι πολλά τότε η λίστα θα πρέπει να περιέχει πολλά στοιχεία. Αυτό συνεπάγεται και χρήση αρκετής μνήμης RAM από το κινητό, κάτι που μπορεί να προκαλέσει διακοπή της εφαρμογής. Για τον λόγο αυτό η λίστα που χρησιμοποιείται κρατάει στη μνήμη μόνο όσα στοιχεία φαίνονται στην οθόνη του χρήστη. Αυτό σημαίνει ότι όταν ο χρήστης κατεβαίνει στην λίστα για να δει τα επόμενα στοιχεία, η εφαρμογή ουσιαστικά αντικαθιστά τα στοιχεία που φαίνονται στην οθόνη με τα επόμενα. Αυτό όμως δεν είναι ορατό στον χρήστη. Σε επόμενα κεφάλαια που θα περιγραφεί η δημιουργία δραστηριοτήτων με λίστα θα είναι πλέον κατανοητή η διαδικασία δημιουργίας και προβολής της λίστας.

Όπως είναι προφανές, κατά την λειτουργία της εφαρμογής οι εργασίες που πραγματοποιούνται είναι αρκετές. Η διαδικασία εμφάνισης μιας δραστηριότητας είναι αρκετά μεγάλη. Αρχικά ξεκινάει το αρχείο με τον κώδικα της κάθε δραστηριότητας. Διαβάζεται το κατάλληλο «layout» και φορτώνεται η δομή της δραστηριότητας. Στο «layout» μπορεί να υπάρχει μια λίστα και να διαβάζει δεδομένα από μια βάση δεδομένων. Επομένως πρέπει να διαβαστούν τα δεδομένα για να παρουσιαστεί η λίστα. Διαβάζεται το αρχείο του μενού και φορτώνεται το κατάλληλο μενού για την κάθε δραστηριότητα. Έπειτα ακολουθεί το κυρίως μέρος του κώδικα που μπορεί να περιέχει επιπλέον λειτουργίες.

Όπως είναι προφανές οι λειτουργίες που πρέπει να εκτελεστούν είναι πάρα πολλές και συνήθως εκτελούνται με τη σειρά που καταγράφονται στον κώδικα. Πολλές φορές όμως είναι απαραίτητο να γίνονται κάποιες λειτουργίες στο παρασκήνιο. Η Android δίνει την δυνατότητα για τέτοιες προσθήκες, δημιουργώντας άλλες δραστηριότητες – κλάσεις που δεν είναι ορατές στον χρήστη. Η κλάσεις αυτές ονομάζονται «**AsyncTask**» ή «**Loader**». Αυτές οι δύο κλάσεις έχουν κάποιες διαφορές μεταξύ τους που δεν παίζουν ρόλο στην παρούσα εργασία, ωστόσο πετυχαίνουν και οι δύο το ίδιο αποτέλεσμα. Με τις κλάσεις αυτές, ο προγραμματιστής πετυχαίνει να κάνει παράλληλα λειτουργίες και να πετύχει μεγαλύτερες ταχύτητες στην εφαρμογή του, αλλά και παράλληλα μικρότερες πιθανότητες διακοπής της διαδικασίας. Αυτό μπορεί κανείς να το καταλάβει σε εφαρμογές, όπου εμφανίζεται η ένδειξη φόρτωσης δεδομένων. Αυτή η ένδειξη υποδηλώνει συνήθως ότι γίνονται λειτουργίες στο παρασκήνιο και ο χρήστης πρέπει να περιμένει να ολοκληρωθούν.

Τέλος, το Android Studio δίνει την δυνατότητα να συνδέσει κανείς μια πραγματική συσκευή στον υπολογιστή και να δοκιμάσει ό,τι έχει δημιουργήσει σε πραγματικό χρόνο, ελέγχοντας για τυχόν σφάλματα σε κάθε Activity. Σε περίπτωση σφάλματος παρουσιάζεται μήνυμα από το Android Studio καταδεικνύοντας που βρίσκεται το σφάλμα και παρουσιάζοντας μήνυμα για το είδος του σφάλματος. Σε κάθε στάδιο σχεδιασμού της εφαρμογής χρησιμοποιήθηκε η κινητή συσκευή Samsung Galaxy A5 (2015), ενημερωμένη κάθε φορά στην τελευταία έκδοση του Android. Ωστόσο, για την χρήση με τέτοιο τρόπο της συσκευής απαιτήθηκαν κάποιες αλλαγές στις ρυθμίσεις της συσκευής, καθώς χρησιμοποιήθηκε πλέον σαν προγραμματιστικό εργαλείο.

ΜΕΘΟΔΟΛΟΓΙΑ

Στο κεφάλαιο αυτό αρχικά θα περιγραφεί η μεθοδολογία που ακολουθήθηκε από την εύρεση του προβλήματος μέχρι την εξαγωγή του τελικού προϊόντος που αποτελεί και την λύση του προβλήματος. Η διαδικασία αυτή ακολουθείται συνήθως σε περιπτώσεις δημιουργίας εφαρμογής για κινητά. Με την ανάλυση της διαδικασίας στο κεφάλαιο αυτό θα είναι πιο κατανοητά τα στάδια που θα αναλυθούν στην συνέχεια.

Πρώτο στάδιο της όλης διαδικασίας είναι ο εντοπισμός ενός προβλήματος. Δεν γίνεται να ξεκινήσει μια έρευνα για εφαρμογή στα κινητά αν δεν λύνει κάποιο πρόβλημα ή αν δεν καλύπτει μια ανάγκη των ανθρώπων. Πριν καν ξεκινήσει η διαδικασία της έρευνας πρέπει πρώτα να εντοπιστεί ένα πρόβλημα το οποίο χρήζει λύση. Η λύση στο πρόβλημα μπορεί ήδη να υπάρχει, ωστόσο μια καινοτόμα εφαρμογή για κινητά θα μπορούσε να αποτελέσει μια καλύτερη και πιο χρήσιμη λύση του προβλήματος, πόσο μάλλον όταν αυτή η εφαρμογή προσφέρει παραπάνω λειτουργίες και να καλύπτει πολλές περιπτώσεις του προβλήματος. Παράλληλα η δημιουργία εφαρμογής μπορεί να καλύπτει ανάγκες των ανθρώπων, ανάγκες που ήδη καλύπτονται με άλλους τρόπους και απλά αυτός ο τρόπος είναι η διαφορετική προσέγγιση.

Μετά την εύρεση του προβλήματος ακολουθεί η εύρεση της λύσης. Σε αυτό το στάδιο με δεδομένο το πρόβλημα προς επίλυση, ο αναλυτής επιχειρεί να βρει την λύση μέσα από την δημιουργία μιας εφαρμογής για κινητά. Αναλύει πως θα επιτευχθεί η λύση του προβλήματος ή η κάλυψη συγκεκριμένων αναγκών. Ουσιαστικά αυτό το στάδιο αποτελεί τη σχεδίαση της εφαρμογής. Σαν σχεδίαση της εφαρμογής δεν νοείται μόνο το οπτικό κομμάτι, αυτό δηλαδή που θα βλέπει ο χρήστης στο κινητό του. Σαν σχεδίαση γενικά είναι η εύρεση εκείνων των στοιχείων που καθιστούν πιο κατανοητή τη λειτουργία της εφαρμογής. Αυτά τα στοιχεία έχουν να κάνουν με το τι θα παρέχει η εφαρμογή, σε ποιους θα απευθύνεται, ποιοι είναι οι ανταγωνιστές της, σε τι διαφέρει από αυτούς και τι προσφέρει παραπάνω από αυτούς. Αν επιπλέον υπάρχει σκοπός κέρδους από την εφαρμογή, θα πρέπει να προστεθούν σε αυτή τη διαδικασία στοιχεία όπως τρόπος προώθησης, σχέσεις με τους πελάτες, απαραίτητες ενέργειες μετά την δημιουργία, πηγές εσόδων και εξόδων. Όλα αυτά αποτελούν το «Business Model Canvas» της εφαρμογής και μέσα από αυτό μπορεί να καταλάβει κανείς ακριβώς τι είδους προϊόν ετοιμάζεται να βγει στην αγορά. Στην συγκεκριμένη περίπτωση δεν αναφέρθηκαν στοιχεία προώθησης, καθώς η εφαρμογή αυτή αποτελεί αντικείμενο διπλωματικής εργασίας. Τα δύο παραπάνω στάδια αναφέρθηκαν μέχρι τώρα σε προηγούμενο κεφάλαιο.

Με δεδομένο λοιπόν το πρόβλημα και το σχέδιο της εφαρμογής επόμενο στάδιο είναι η συλλογή των δεδομένων που θα χρησιμοποιηθούν στην εφαρμογή. Ωστόσο για την συλλογή των δεδομένων πρέπει πρώτα να ξεκαθαριστεί η περιοχή μελέτης, αφού με βάση αυτή θα συλλεχθούν τα κατάλληλα δεδομένα. Η επιλογή της περιοχής μελέτης θα γίνει από τον αναλυτή με βάση συγκεκριμένα κριτήρια, τα οποία τα ορίζει αυτός. Μπορεί να επιλέξει μια περιοχή που γνωρίζει καλά για να τον βοηθήσουν οι γνώσεις του ή μπορεί να επιλέξει μια περιοχή που δεν ξέρει για να την μάθει. Μπορεί να επιλέξει μια περιοχή που τον ενδιαφέρει άμεσα, είτε για λόγους μελλοντικής εργασίας είτε για λόγους καταγωγής και διαμονής. Στην επιλογή του είναι και το μέγεθος της περιοχής. Με βάση λοιπόν αυτή την περιοχή που επέλεξε, συλλέγει τα απαραίτητα δεδομένα.

Κάθε περιοχή παρέχει αρκετά δεδομένα για πολλές διαφορετικές περιπτώσεις. Ο αναλυτής με βάση την εφαρμογή που έχει σχεδιάσει φιλτράρει και συλλέγει τα δεδομένα που απαιτούνται για την εφαρμογή του. Η συλλογή των δεδομένων μπορεί να γίνει από πολλές πηγές. Η κυριότερη πηγή που παρέχει τα περισσότερα δεδομένα είναι το διαδίκτυο. Πλέον όλοι μπορούν να έχουν πρόσβαση σε πλήθος δεδομένων και τα πιο πολλά είναι δωρεάν. Άλλες πηγές μπορεί να είναι βιβλία, τοπικές οργανώσεις, υπουργεία και επιχειρήσεις ή μπορεί ο αναλυτής να μεταβεί κατευθείαν στο σημείο εκείνο που θέλει τα δεδομένα για να συλλέξει επί τόπου ό,τι επιθυμεί. Τα δεδομένα αυτά καταγράφονται σε βάση δεδομένων ή πρόχειρα σε χαρτί και με βάση αυτά ξεκινάει η δημιουργία της εφαρμογής.

Επόμενο στάδιο είναι το πρακτικό, η υλοποίηση δηλαδή της εφαρμογής. Στο στάδιο αυτό γίνεται σχεδιασμός της λειτουργίας της εφαρμογής. Σχεδιάζεται η διαδικασία σύμφωνα με την οποία θα καταλήξουμε στην τελική εφαρμογή. Με βάση ήδη υπάρχουσες γνώσεις ή γνώσεις που θα αποκτηθούν αποφασίζεται ποια πλατφόρμα θα χρησιμοποιηθεί για τον σχεδιασμό της εφαρμογής. Ο τρόπος αυτός μπορεί να αλλάξει καθώς μπορεί κατά την διάρκεια της δημιουργίας να βρεθεί ένα καλύτερο και πιο εύχρηστο περιβάλλον εργασίας που παρέχει πιο ακριβή και ευκολότερα εργαλεία. Όπως είναι προφανές όλα τα παραπάνω στάδια δεν απαιτούν συγκεκριμένες γνώσεις και μπορούν να πραγματοποιηθούν σχεδόν από τον καθένα. Το στάδιο αυτό απαιτεί ειδικές γνώσεις των πλατφορμών που θα χρησιμοποιηθούν και για τον λόγο αυτό θεωρείται το πιο χρονοβόρο στάδιο της διαδικασίας.

Όταν ο αναλυτής αποφασίσει πως θα δομήσει την εφαρμογή και έχει κατανοήσει πλέον τα βασικά σημεία χρήσης της πλατφόρμας που θα χρησιμοποιήσει ξεκινάει η διαδικασία δημιουργίας της εφαρμογής. Αρχικά δημιουργεί τα βασικά σημεία της εφαρμογής, τα στοιχεία που είναι απαραίτητα για να ξεκινήσει να λειτουργεί η εφαρμογή και προσθέτει τα δεδομένα που έχει συλλέξει. Κάθε φορά που γίνεται μια αλλαγή στην λειτουργία της εφαρμογής πρέπει να υπάρχει έλεγχος της λειτουργίας. Δεν είναι λίγες οι φορές που η εφαρμογή σταματάει να λειτουργεί λόγω προγραμματιστικών λαθών.

Μόλις ολοκληρωθεί το κυρίως μέρος της εφαρμογής και προστεθούν τα δεδομένα αρχίζει και γίνεται προσθήκη επιπλέον λειτουργιών. Η κύρια λειτουργία της εφαρμογής, το σημείο αυτό δηλαδή που την διαφοροποιεί από τις άλλες εφαρμογές που αποτελούν και τους ανταγωνιστές της μπορεί να προστεθεί είτε στην αρχή είτε στο τέλος. Προσθέτονται επιπλέον λειτουργίες για μια καλύτερη εμπειρία του χρήστη, ελέγχοντας πάντα μετά από κάθε αλλαγή για την σωστή λειτουργία της εφαρμογής. Πολλές φορές η προσθήκη των λειτουργιών αυτών έχει να κάνει με γνώσεις που δεν έχει ο αναλυτής. Αντιμετωπίζονται λειτουργίες που δεν έχει συναντήσει και πρέπει να χρησιμοποιήσει εργαλεία που δεν είχε χρησιμοποιήσει ξανά. Ωστόσο, υπάρχουν πάρα πολλές πληροφορίες στο διαδίκτυο από άλλους ανθρώπους που έχουν χρησιμοποιήσει τα αντίστοιχα εργαλεία και παρέχουν αναλυτική βοήθεια στους υπόλοιπους. Με δοκιμές και τις κατάλληλες προσαρμογές στην κάθε βοήθεια, επιτυγχάνεται κάθε φορά το ζητούμενο αποτέλεσμα.

Κατά την διάρκεια δημιουργίας της εφαρμογής πολλές φορές φαίνεται ότι η εφαρμογή έχει τελειώσει και είναι έτοιμη για παράδοση. Ωστόσο, κάθε φορά προέκυπτε άλλη μια βελτίωση, άλλη μια διόρθωση που καθιστούσε την εφαρμογή ακόμα καλύτερη για τον χρήστη. Η διαδικασία βελτίωσης της εφαρμογής δεν τελειώνει ποτέ. Πάντα θα υπάρχει κάτι που χρειάζεται αλλαγή και κάτι που πρέπει να προστεθεί για να μπορεί ο χρήστης να κάνει ακόμα περισσότερα. Για να καταλάβει όμως κανείς τι χρειάζεται βελτίωση έπρεπε να γίνει χρήση της εφαρμογής όχι μόνο από τον σχεδιαστή, αλλά και από άτομα που χρησιμοποιούσαν για πρώτη φορά την εφαρμογή. Ο σχεδιαστής βλέπει μεν μόνος του κάποιες αλλαγές αλλά όταν άτομα που δεν ξέρουν πως λειτουργεί και τι παρέχει η εφαρμογή την χρησιμοποιήσουν μπορεί ο σχεδιαστής να καταλάβει τι πρέπει να αλλάξει και τι χρειάζονται ακόμα.

Έτσι, με βάση την παραπάνω διαδικασία δημιουργείται η εφαρμογή. Ωστόσο, μετά την οριστική δημιουργία της πρέπει να γίνουν και πάλι έλεγχοι ώστε να επιβεβαιωθούν τα αποτελέσματα της εφαρμογής. Πολλές φορές δημιουργείται μεν η εφαρμογή, αλλά τα αποτελέσματα των λειτουργιών δεν είναι σωστά εξαιτίας μιας παράλειψης ή ενός λάθους του προγραμματιστή, χωρίς ωστόσο να προκαλείται διακοπή της λειτουργίας. Αυτά τα λάθη είναι τα πιο δύσκολα, καθώς εντοπίζονται πιο δύσκολα και χρειάζονται αρκετές δοκιμές για να καταλάβεις τι συμβαίνει. Μόλις διορθωθούν αυτά τα λάθη η εφαρμογή είναι πλέον έτοιμη. Συνοπτικά η μεθοδολογία που περιγράφηκε παραπάνω μπορεί να χωριστεί στα παρακάτω βήματα:

1. Εύρεση του προβλήματος
2. Εύρεση της λύσης (Εφαρμογή)
3. Σχεδιασμός της εφαρμογής
4. Ορισμός περιοχής μελέτης
5. Συλλογή δεδομένων
6. Επιλογή της κατάλληλης πλατφόρμας για την δημιουργία της εφαρμογής
7. Απόκτηση κατάλληλων γνώσεων
8. Δημιουργία βασικών στοιχείων της εφαρμογής
9. Έλεγχος λειτουργίας
10. Προσθήκη/Αλλαγή μιας λειτουργίας
11. Επανάληψη βημάτων 9 και 10 μέχρι την οριστικοποίηση της εφαρμογής

ΕΦΑΡΜΟΓΗ

Στο κεφάλαιο αυτό περιγράφεται αναλυτικά η διαδικασία και όλα τα πρακτικά μέρη της εργασίας αυτής. Πιο συγκεκριμένα, παρουσιάζεται η περιοχή μελέτης και ο λόγος που επιλέχθηκε, αναφέρεται ο τρόπος συλλογής και επεξεργασίας των δεδομένων και προγραμματιστικές λεπτομέρειες για την καλύτερη κατανόηση της εφαρμογής. Τέλος, παρουσιάζεται η εφαρμογή ώστε να γίνει κατανοητός ο τρόπος χρήσης και καταγράφονται συμπεράσματα για την εφαρμογή, δηλαδή τα θετικά και τα αρνητικά που διαπιστώθηκαν μετά την ολοκλήρωσή της.

Περιοχή Μελέτης

Η παρούσα εργασία έχει να κάνει με την δημιουργία μιας εφαρμογής για κινητά τηλέφωνα «Android», ώστε να διευκολύνει τους χρήστες στην καλύτερη αξιοποίηση του ελεύθερου χρόνου τους. Είναι προφανές λοιπόν, ότι σαν επιχειρηματική ιδέα, αλλά και σαν ιδέα γενικά πρέπει να επικεντρωθεί σε μια περιοχή με μεγάλη αγορά, με την έννοια του πληθυσμού. Σημαντικός παράγοντας επομένως για την δημιουργία μιας εφαρμογής είναι η αγορά που απευθύνεται ώστε τα αποτελέσματά της να είναι καλύτερα και η ζήτησή της να είναι μεγαλύτερη. Σε περιοχή με μεγάλη αγορά επιπλέον το κάθε πρόβλημα που υπάρχει μεγεθύνεται, με αποτέλεσμα να είναι ακόμα πιο αναγκαία η ύπαρξη μιας καλής εφαρμογής που θα αποτελέσει λύση στο πρόβλημα. Θα ήταν άχρηστη μια εφαρμογή αν σαν περιοχή μελέτης χρησιμοποιούσε μια πόλη ή ένα χωριό με λίγους κατοίκους και επισκέπτες, σε αντίθεση με μια μεγάλη πόλη που ο πληθυσμός είναι μεγαλύτερος και η χρήση της εφαρμογής θα είναι προφανώς περισσότερη.

Ωστόσο, η εφαρμογή έχει να κάνει με την αξιοποίηση του ελεύθερου χρόνου σε σημεία ενδιαφέροντος. Είναι προφανές λοιπόν, ότι η περιοχή που θα επιλεχθεί πρέπει να έχει πολλά σημεία ενδιαφέροντος. Αυτός ήταν και ο κυριότερος παράγοντας επιλογής της περιοχής μελέτης. Δεν έχει νόημα να δημιουργήσει κανείς μια εφαρμογή που σαν δεδομένα θα έχει 5 ή 10 σημεία ενδιαφέροντος. Αν τα σημεία αυτά είναι λίγα, ο καθένας θα μπορούσε εύκολα να πάρει μια απόφαση για το που να πάει και πώς να αξιοποιήσει το χρόνο του. Αν το πλήθος των σημείων είναι μεγάλο, αυτό σημαίνει ότι ο χρήστης θα έχει να διαλέξει ανάμεσα σε πολλές επιλογές, γεγονός που κάνει την απόφασή του ακόμα πιο δύσκολη και την χρήση της εφαρμογής ακόμα πιο αναγκαία. Άρα, ένας δεύτερος παράγοντας για την επιλογή της περιοχής μελέτης είναι το πλήθος των σημείων ενδιαφέροντος.

Τέλος, ένα τρίτο κριτήριο που χρησιμοποιήθηκε για την επιλογή της κατάλληλης περιοχής μελέτης ήταν η γνώση της περιοχής. Σήμερα το διαδίκτυο είναι σημαντικό εργαλείο ώστε να βρεις σημεία ενδιαφέροντος. Ωστόσο, όταν έχεις να κάνεις με μέρη ώστε να αξιοποιήσεις τον χρόνο σου, την καλύτερη πρόταση θα την πάρει κανείς από κάποιον που ζει σε αυτή την πόλη. Ο κάτοικος της πόλης γνωρίζει μέρη που ναι μεν υπάρχουν στο διαδίκτυο, αλλά δεν τους έχει δοθεί η κατάλληλη προσοχή ώστε να αναδειχθούν και να προσελκύσουν κόσμο.

Με γνώμονα λοιπόν αυτά τα τρία κριτήρια και δεδομένου του γεγονότος ότι οι γνώσεις μου σε σημεία ενδιαφέροντος είναι αρκετά καλές για την Αθήνα και το Ηράκλειο Κρήτης (λόγω καταγωγής), σαν περιοχή μελέτης επιλέχθηκε η Αθήνα. Η έκταση της Αθήνας είναι 2.929 km^2 και θεωρείται αρκετά πυκνοκατοικημένη, αφού ο πληθυσμός της ξεπερνάει τις 664.000, σύμφωνα με την απογραφή του 2011. Τα σημεία ενδιαφέροντος που επιλέχθηκαν λοιπόν βρίσκονται στην Αθήνα με κάποιες εξαιρέσεις που πιάνουν περιοχές λίγο έξω από αυτή.

Συλλογή Δεδομένων – Πηγές

Το επόμενο στάδιο της εργασίας ήταν η συλλογή των δεδομένων. Πριν την συλλογή των δεδομένων όμως έπρεπε να γίνει κατανοητή η λειτουργία της εφαρμογής ώστε να επικεντρωθεί η έρευνα στο τι δεδομένα απαιτούνται. Μετά λοιπόν τον σχεδιασμό της εφαρμογής και την κατανόηση της λειτουργίας της ήταν προφανές πλέον ότι τα κυρίως δεδομένα θα αφορούσαν πληροφορίες για τα σημεία ενδιαφέροντος της περιοχής μελέτης. Σ' αυτό το σημείο, πρέπει να σημειωθεί ότι η έρευνα δεν περιορίστηκε σε σημεία μόνο εντός της περιοχής μελέτης, αλλά συλλέχθηκαν και δεδομένα εκτός της περιοχής μελέτης που θεωρήθηκαν σημαντικά και έπρεπε να καταγραφούν.

Η κύρια πηγή συλλογής των δεδομένων ήταν το διαδίκτυο. Η μηχανή αναζήτησης της Google ή το TripAdvisor ή η Wikipedia είναι τρία από τα κυριότερα μέρη που παρέχονται πληροφορίες για τα σημεία ενδιαφέροντος. Επιπλέον, η ιστοσελίδα του κάθε σημείου, που αναφέρει εξειδικευμένες πληροφορίες για το αντίστοιχο μέρος ήταν σημαντική πηγή εύρεσης αληθών δεδομένων. Επιπροσθέτως έγινε προσπάθεια να μην περιοριστεί η έρευνα μόνο στα δεδομένα του διαδικτύου, γι' αυτό έγινε και έρευνα σε πληροφορίες που γνωρίζουν κάτοικοι (φίλοι και γνωστοί κυρίως) της Αθήνας και του Πειραιά. Από γνώσεις και εμπειρίες τους συλλέχθηκαν, όχι μόνο τα ονόματα ή η τοποθεσία των δεδομένων, αλλά και άλλες λεπτομέρειες, οι οποίες δεν είναι αληθείς στο διαδίκτυο, όπως ο χρόνος περιήγησης.

Όταν κάποιες πληροφορίες δεν μπόρεσαν να συλλεχθούν με κανέναν από τους παραπάνω τρόπους, η επικοινωνία με το αντίστοιχο μουσείο, για παράδειγμα, διευκόλυνε πολύ την διαδικασία. Επικοινωνία υπήρξε με αρκετά μουσεία και σημεία ενδιαφέροντος, κυρίως για την καταγραφή του χρόνο περιήγησης, αλλά παράλληλα γινόταν και μια επιβεβαίωση των στοιχείων που ήδη είχαν συλλεχθεί για το αντίστοιχο μέρος. Έτσι, είναι προφανές λοιπόν ότι αυτά τα δεδομένα θα ήταν πιο ακριβή από τα υπόλοιπα.

Η διαδικασία καταγραφής και εξαγωγής των τελικών αποτελεσμάτων ήταν απλή. Αρχικά γινόταν εύρεση του κάθε σημείου και των πληροφοριών του, με τους τρόπους που αναφέρονται παραπάνω. Στη συνέχεια, μέσω του προγράμματος προστέθηκαν όλα τα σημεία ενδιαφέροντος μαζί με τις πληροφορίες τους σε ένα χάρτη. Με αυτόν τον τρόπο όλα τα σημεία ήταν συγκεντρωμένα και γινόταν η κατάλληλη πρόσθεση και αναζήτηση σημείων σε περιοχές με λίγα σημεία ενδιαφέροντος. Η προσθήκη των σημείων έγινε χειροκίνητα ένα – ένα και αυτός είναι ο λόγος που υπάρχουν κάποιες μικρές διαφορές σε μεγάλο zoom στις συντεταγμένες των σημείων σε σύγκριση με τα αντίστοιχα σημεία στους χάρτες της Google. Τέλος, μετά την προσθήκη των σημείων έγινε εξαγωγή του πίνακα των πληροφοριών των σημείων (μαζί με τις συντεταγμένες τους) σε έναν πίνακα στο Microsoft Excel για την καλύτερη επεξεργασία των τελικών δεδομένων πλέον.

Πιο συγκεκριμένα, για κάθε σημείο μετά την εξαγωγή των αποτελεσμάτων από το QGIS Desktop και την προσθήκη μερικών πληροφοριών ακόμα που δεν μπορούσαν να προστεθούν στο QGIS Desktop, συγκεντρώθηκαν οι εξής πληροφορίες που χρησιμοποιήθηκαν στην βάση δεδομένων της εφαρμογής:

- **Γεωγραφικό μήκος:** Απαραίτητο για τον προσδιορισμό της τοποθεσίας των σημείων στο χάρτη.
- **Γεωγραφικό πλάτος:** Απαραίτητο για τον προσδιορισμό της τοποθεσίας των σημείων στο χάρτη.

- **Εικόνα:** Φωτογραφία από το σημείο ενδιαφέροντος για να γνωρίζει ο χρήστης πως είναι.
- **Όνομα:** Το όνομα του σημείου ενδιαφέροντος, ώστε ο χρήστης να γνωρίζει τι θα επισκεφθεί.
- **Διεύθυνση:** Η διεύθυνση του σημείου ενδιαφέροντος, ώστε να γνωρίζει ο χρήστης την τοποθεσία.
- **Κατηγορία:** Η κατηγορία του σημείου ενδιαφέροντος. Η τελική κατηγοριοποίηση των σημείων κατέληξε ύστερα από πολλές κατηγοριοποιήσεις. Αρχικά, κάθε σημείο που συλλέχθηκε είχε μια δική του εξειδικευμένη κατηγορία. Μετά την ολοκλήρωση όλων των σημείων, οι κατηγορίες κατέληξαν να είναι 4, με ομαδοποιήσεις των ήδη υπαρχόντων κατηγοριών. Έτσι, οι τελικές κατηγορίες σημείων ενδιαφέροντος που προέκυψαν είναι:
    - Μουσεία, που περιλαμβάνει μουσεία, γκαλερί, πινακοθήκες, ιδρύματα και άλλα στεγασμένα κτίρια ή ακίνητα που φιλοξενούν επισκέπτες.
    - Αξιοθέατα, που περιλαμβάνει αγάλματα, μνημεία, πλατείες και γενικά οτιδήποτε βρίσκεται σε ανοιχτό χώρο και αποτελεί σημείο με κάποιο πολιτιστικό ενδιαφέρον.
    - Εκκλησίες, είτε αυτές είναι μικρές, χριστιανικές ή όχι.
    - Άλλα, όπως γήπεδα, πανεπιστήμια, ζωολογικούς κήπους, εμπορικά κέντρα, λιμάνια, αεροδρόμια, στάσεις του μετρό (που έχουν κάποιο πολιτιστικό ενδιαφέρον) και οτιδήποτε άλλο δεν ανήκει στις παραπάνω κατηγορίες.
- **Τηλέφωνο:** Το τηλέφωνο από το σημείο, όταν αυτό είναι διαθέσιμο. Το τηλέφωνο συνήθως είναι διαθέσιμο στα μουσεία, στα γήπεδα, στα ιδρύματα και σε ότι άλλο είναι κλειστός οργανωμένος χώρος ή είναι γνωστό πολιτιστικό σημείο.
- **Τιμή:** Η τιμή εισόδου στο κάθε μέρος όταν αυτό είναι απαραίτητο. Σε περίπτωση που σε κάποιο σημείο δεν υπάρχει τιμή εισόδου και είναι δωρεάν η τιμή είναι 0€.
- **Δημοφιλές:** Η πληροφορία αυτή περιλαμβάνει τις τιμές ΝΑΙ / ΟΧΙ, οι οποίες υποδηλώνουν αν το αντίστοιχο σημείο επίσκεψης είναι δημοφιλές ή όχι με γνώμονα τις κριτικές και τις επισκέψεις που έχει στο διαδίκτυο.
- **Χρόνος περιήγησης:** Σαν χρόνος περιήγησης καταγράφεται ο χρόνος σε λεπτά που χρειάζεται ένας επισκέπτης να παραμείνει στο αντίστοιχο μέρος ώστε να προλάβει να δει όλο το αξιοθέατο/μουσείο/μνημείο κλπ.
- **Σύνδεσμος:** Ο σύνδεσμος στο διαδίκτυο, η διεύθυνση δηλαδή που μπορείς να βρεις παραπάνω πληροφορίες για το κάθε σημείο ενδιαφέροντος.

Όπως είναι προφανές η εξακρίβωση κάποιων πληροφοριών είναι πιο εύκολη από την εξακρίβωση άλλων. Για παράδειγμα, το όνομα, η διεύθυνση, ο σύνδεσμος, η τοποθεσία (γεωγραφικό πλάτος & μήκος) και η εικόνα του σημείου είναι τα πρώτα και τα πιο ακριβή στοιχεία που συλλέχθηκαν για κάθε σημείο. Όσον αφορά τώρα το τηλέφωνο και την τιμή εισόδου αυτά θεωρούνται μεν ακριβή στοιχεία, ωστόσο υπάρχει και η πιθανότητα να έχουν τροποποιηθεί χωρίς ταυτόχρονη ενημέρωση του διαδικτύου. Ωστόσο και σε αυτά τα στοιχεία μπορεί κανείς να πει ότι η ακρίβειά τους είναι αρκετά μεγάλη.

Αυτό όμως που δύσκολα μπορεί να συλλέξει κανείς από το διαδίκτυο είναι δεδομένα για την δημοτικότητα ή τον χρόνο περιήγησης ενός σημείου ενδιαφέροντος. Στο μεν πρώτο μπορείς να θέσεις την επισκεψιμότητα σαν κριτήριο και με βάση αυτό να ορίσεις ένα μέρος σαν δημοφιλές ή όχι. Έτσι, με κύριο γνώμονα τις ίδιες γνώσεις καθώς και τον αριθμό των κριτικών για κάθε σημείο (όπου αυτές ήταν διαθέσιμες) ορίστηκαν κάποια σημεία σαν δημοφιλή και τα υπόλοιπα όχι.

Όσον αφορά τώρα την δεύτερη πληροφορία που έχει να κάνει με τον χρόνο περιήγησης αυτή θεωρήθηκε η δυσκολότερη πληροφορία. Και αυτό συμβαίνει γιατί από άνθρωπο σε άνθρωπο τα δεδομένα αυτά μπορεί να διαφέρουν. Ο ένας μπορεί να χρειαστεί μία ώρα να δει ένα μουσείο γιατί δεν τον ενδιαφέρουν τόσο οι πληροφορίες για τα εκθέματα, ενώ ο άλλος διαβάζοντας κάθε πληροφορία για κάθε έκθεμα μπορεί να χρειαστεί δύο ώρες. Επιπλέον, κάποια πολιτιστικά σημεία διαθέτουν περιηγήσεις με ξεναγό που διαρκούν συγκεκριμένο χρονικό διάστημα. Ευτυχώς, η Google παρέχει για κάποια σημεία πληροφορίες χρόνων παραμονής των επισκεπτών στο αντίστοιχο πολιτιστικό σημείο από δεδομένα χρηστών. Έτσι, για όσα σημεία παρέχεται τέτοια πληροφορία χρησιμοποιήθηκαν αυτά τα δεδομένα ως τα πιο αξιόπιστα. Για όσα σημεία δεν παρεχόταν τέτοια πληροφορία, μια επικοινωνία με το αντίστοιχο σημείο ενδιαφέροντος ήταν πολύ βοηθητική. Τέλος, σε όσα σημεία ήταν ανέφικτη η απόκτηση τέτοιων πληροφοριών και δεν υπήρχαν ούτε ίδιες γνώσεις χρησιμοποιήθηκε μια προσεγγιστική τιμή με βάση το μέγεθος του αντίστοιχου μέρους στο χάρτη. Για παράδειγμα σε κάποιες μικρές εκκλησίες ή σε μεμονωμένα αγάλματα που δεν έχεις πολλά να δεις χρησιμοποιήθηκε ο χρόνος των 5 λεπτών σαν χρόνος περιήγησης, ενώ στην περίπτωση μιας μεγάλης πλατείας χρησιμοποιήθηκε παραπάνω χρόνος.

Μετά την συλλογή των πληροφοριών των σημείων ακολούθησε η συλλογή των χρόνων μετάβασης μεταξύ τους. Για την ανάγκη της εργασίας, όπως θα αναφερθεί σε επόμενο κεφάλαιο ήταν απαραίτητη η εύρεση του χρόνου που απαιτείται για να μεταβεί ένας πεζός (μέσω των δρόμων) από κάθε σημείο προς όλα τα άλλα. Τα δεδομένα αυτά τα παρέχει η Google μέσα από τους χάρτες της. Τα σημεία ωστόσο ήταν 267, που σημαίνει ότι χρειάζονται 267 x 267 = 71.289 χρόνοι, αριθμός πολύ μεγάλος για να αναζητηθούν ένας – ένας.

Μετά από πολλή έρευνα στο διαδίκτυο για την εύρεση του καλύτερου και οικονομικότερου τρόπου εύρεσης των δεδομένων, καθώς στις περισσότερες περιπτώσεις η παροχή τόσων δεδομένων κοστίζει, αποφασίστηκε η ανάπτυξη μιας πρόχειρης εφαρμογής, όπου θα λαμβάνει τα δεδομένα από την Google μέσω ενός έτοιμου εργαλείου που παρέχεται. Το εργαλείο αυτό ονομάζεται «Google Maps Distance Matrix API» και μέσω αυτού μπορεί κανείς να αποκτήσει δεδομένα που έχουν να κάνουν με την απόσταση μεταξύ σημείων, τον χρόνο ή τον τρόπο μετάβασης. Η διαδικασία είναι απλή. Ουσιαστικά δημιουργείται μια διεύθυνση URL, η οποία περιέχει τα απαραίτητα στοιχεία. Με βάση αυτή τη διεύθυνση, η οποία αποτελεί ένα «HTTP Request» ζητούνται δεδομένα από την Google. Η διεύθυνση αναλύεται και με βάση τα στοιχεία επιστρέφονται τα ζητούμενα δεδομένα. Η διεύθυνση αυτή είναι της μορφής:

```
https://maps.googleapis.com/maps/api/distancematrix/json?origins=37.9
727,23.7183&destinations=37.9749,23.7197&mode=walking&key=MY_API_KEY
```

Αναλυτικότερα, το πρώτο μέρος έχει να κάνει με το εργαλείο που χρησιμοποιείται. Με μαύρα γράμματα αποτυπώνεται το κυρίως μέρος της διεύθυνσης, σύμφωνα με το οποίο ζητείται να χρησιμοποιηθεί το «Distance Matrix API». Στην συνέχεια αναφέρεται ο τύπος του αρχείου που ζητείται να επιστραφεί, που στην προκειμένη περίπτωση είναι αρχείο json ακολουθούμενο με ένα ερωτηματικό (?). Έπειτα τοποθετούνται όλες οι παράμετροι με το σύμβολο «&» ενδιάμεσα. Αρχικά οι συντεταγμένες του αρχικού σημείου ή σημείων (origins), οι συντεταγμένες των προορισμών (destinations), ο τρόπος μετάβασης (mode), που στην προκειμένη περίπτωση είναι με τα πόδια (walking) και τέλος ένας μοναδικός κωδικός που σου παρέχει η Google για να αναγνωρίσει ποιος ζητάει αυτά τα δεδομένα (API key).

Ωστόσο, υπάρχει ένας περιορισμός στα δεδομένα που μπορεί να σου παρέχει η Google. Ανά πάσα στιγμή μπορεί ο προγραμματιστής να μπει στην δική του προσωπική σελίδα που έχει δημιουργηθεί στην πλατφόρμα της Android και να ελέγξει τι χρήση έγινε στην εφαρμογή, ποια εργαλεία χρησιμοποιήθηκαν και πόσα δεδομένα ζητήθηκαν. Καθημερινά μπορεί κανείς να ζητήσει μέχρι 2.500 αποστάσεις χωρίς καμία χρέωση. Από εκεί και πέρα κάθε 1.000 αποστάσεις χρεώνονται 0,50 $,

ενώ το μέγιστο όριο καθημερινών αποστάσεων είναι 10.000. Για περισσότερα αιτήματα χρειάζεται ένας Premium λογαριασμός. Για τον λόγο αυτό, έγινε προσπάθεια να ζητούνται μέχρι 2.500 αποστάσεις καθημερινά. Αυτό σημαίνει ότι κάθε μέρα ζητούνταν οι αποστάσεις από 9 σημεία προς όλα τα υπόλοιπα (9 x 266 = 2.394 αποστάσεις). Έτσι, αν και σε μεγάλο χρονικό διάστημα, συλλέχθηκαν οι απαραίτητοι χρόνοι, ώστε τα δεδομένα να είναι πιο ακριβή.

Σε περιπτώσεις όπου η Google δεν μπορούσε να παρέχει δεδομένα αποστάσεων, είτε αυτές ήταν πολύ μεγάλες, είτε δεν υπήρχε τρόπος μετάβασης με τα πόδια, χρησιμοποιήθηκε ο τύπος υπολογισμού της απόστασης Manhattan (Manhattan Distance). Υπολογίζοντας 100 τυχαίες αποστάσεις μεταξύ τυχαίων σημείων στην πλοήγηση που προσφέρει η Google, και με βάση τα αποτελέσματα που προέκυψαν, υπολογίστηκε μια μέση ταχύτητα στην περιοχή της Αθήνας (**4,7 km/h**), με βάση τον τύπο της απόστασης, του χρόνου και της ταχύτητας. Η ταχύτητα αυτή χρησιμοποιήθηκε στον τύπο υπολογισμού της Manhattan απόστασης, από τον οποίο προέκυπτε κάθε φορά ο αντίστοιχος χρόνος μετάβασης. Έτσι είχαμε δεδομένα για όλες τις αποστάσεις. Το αναλυτικό απόσπασμα του κώδικα για τον υπολογισμό των αποστάσεων φαίνεται στο Παράρτημα.

Συνοπτικά, η συλλογή των σημείων, λόγω της μη αυτοματοποίησης της διαδικασίας και της χειροκίνητης εύρεσής τους, καθώς και των χρόνων λόγω της προαναφερθείσας διαδικασίας, θεωρήθηκε πολύ χρονοβόρα. Επιπλέον, μετά την ολοκλήρωση της εφαρμογής, θεωρήθηκε αχρείαστη η χρήση του QGIS Desktop, αφού η Google παρέχει δωρεάν εργαλεία καταγραφής σημείων σε χάρτη και τα δεδομένα τελικά επεξεργάστηκαν σε Microsoft Excel. Τις συντεταγμένες των σημείων, που ήταν τα μόνα δεδομένα που εξάχθηκαν από το QGIS Desktop, μπορεί κανείς να τις βρει και από το Google Maps, με καλύτερη ακρίβεια απ' ότι να καταχωρίσεις σημεία χειροκίνητα. Το ίδιο ισχύει και για τις διευθύνσεις, οι οποίες παρέχονται αυτόματα με την εισαγωγή των συντεταγμένων. Με κάποιες προσθήκες λοιπόν από δεδομένα της Google ακολούθησε η συγγραφή του κώδικα της εφαρμογής, ώστε να καταλήξει η εργασία στο τελικό της αποτέλεσμα, το οποίο αναφέρεται σε επόμενη ενότητα.

Συγγραφή κώδικα – Δημιουργία Εφαρμογής

Μετά την συλλογή των δεδομένων ή παράλληλα (για τα δεδομένα των χρόνων) ακολούθησε το προγραμματιστικό μέρος της εργασίας. Στο κεφάλαιο αυτό θα γίνει κατανοητή η διαδικασία που ακολουθήθηκε ώστε να προκύψει το παραχθέν αποτέλεσμα. Θα περιγραφούν αναλυτικά όλα τα στάδια που πέρασαν για να φθάσει η εφαρμογή στην τελική μορφή της με τρόπο ώστε να είναι κατανοητά σε οποιονδήποτε, ακόμα και μη γνώστη προγραμματισμού. Αρχικά πρέπει να σημειωθεί ότι κύριος στόχος με την ανάπτυξη της εφαρμογής ήταν να αποφευχθεί η χρήση διαδικτύου και γενικά να μπορεί ο χρήστης όπου και να είναι να την χρησιμοποιήσει, χωρίς να εξαρτάται από το αν μπορεί ή όχι να συνδεθεί στο διαδίκτυο, καθώς απευθύνεται και σε τουρίστες που μπορεί να μην έχουν παντού πρόσβαση στο διαδίκτυο.

Όσον αφορά τις συσκευές για τις οποίες θα αναπτυχθεί η εφαρμογή αυτές είναι οι Android. Μέσα από τα δεδομένα της ιστοσελίδας www.kantarworldpanel.com, που αφορά τους χρήστες Android και άλλων συσκευών σε 5 μεγάλες χώρες της Ευρώπης (Ιταλία, Ισπανία, Γαλλία, Γερμανία και Μεγάλη Βρετανία) προκύπτει το συμπέρασμα ότι σε όλες τις χώρες υπερισχύουν οι χρήστες συσκευών Android. Για τον λόγο αυτό επιλέχθηκε η εφαρμογή να σχεδιαστεί για συσκευές Android, καθώς αποτελούν το μεγαλύτερο μέρος της αγοράς. Τα ποσοστά των χρηστών Android συσκευών φαίνονται στον παρακάτω πίνακα (Πίνακας 1).

Χώρα	Ποσοστό χρηστών (Ιανουάριος 2015)	Ποσοστό χρηστών (Δεκέμβριος 2017)	Μέγιστο ποσοστό σε αυτό το διάστημα
Ιταλία	66,6%	82,1%	84,9%
Ισπανία	86,7%	87,1%	93,9%
Γαλλία	65,3%	75,6%	80,6%
Γερμανία	71,7%	74,8%	82,8%
Μεγάλη Βρετανία	51,5%	55,3%	65,1%

Πίνακας 1: Πίνακας ποσοστών χρηστών συσκευών Android

Πριν από την ανάπτυξη ωστόσο της εφαρμογής είναι απαραίτητος ο σχεδιασμός της. Έτσι αποφασίστηκε να υπάρχει η αρχική οθόνη που θα εμφανίζονται όλα τα αξιοθέατα στο χρήστη και από εκείνη την οθόνη να πηγαίνει σε μια άλλη στην οποία θα καθορίζει τα κριτήρια που θέλει και θα του εμφανίζεται η προτεινόμενη διαδρομή. Όσον αφορά την αρχική οθόνη που ο χρήστης θα έβλεπε τα σημεία υπήρχαν δύο επιλογές: η πρώτη επιλογή ήταν να εμφανίζεται μια λίστα με τα αξιοθέατα και η δεύτερη ένας χάρτης με τα αξιοθέατα στο χάρτη. Ωστόσο, τα σημεία προέκυψαν πολλά και λόγω της μικρής τους χωρικής διασποράς επιλέχθηκε η μορφή της λίστας, καθώς σε χάρτη οι πληροφορίες θα ήταν αρκετά δυσδιάκριτες.

Ανάπτυξη των βασικών λειτουργιών της εφαρμογής

Έτσι ξεκίνησε η διαδικασία της ανάπτυξης της εφαρμογής. Για να εμφανίζονται σημεία σε λίστα υπάρχουν πολλοί τρόποι. Ο σκοπός όμως ήταν τόσο να μην χρειάζεται η χρήση διαδικτύου για την ανάγνωση των δεδομένων, όσο και να διατηρούνται οποιεσδήποτε αλλαγές κάνει ο χρήστης στις πληροφορίες των σημείων. Για τον λόγο αυτό επιλέχθηκε η δημιουργία μιας τοπικής βάσης δεδομένων στο κινητό του χρήστη. Αυτό επιτεύχθηκε με εργαλεία που παρέχει το Android Studio και κάνουν χρήση εντολών SQL. Το μόνο αρνητικό σε αυτό ήταν ότι μόνο ο χρήστης θα είχε πρόσβαση στα δεδομένα και ο κάθε χρήστης θα είχε τα δικά του δεδομένα, γεγονός που βοηθάει μεν τους χρήστες αλλά δεν υπάρχει προοπτική βελτίωσης των δεδομένων. Για τον λόγο αυτό, αποφασίστηκε να μπορεί ο χρήστης να στέλνει τα δεδομένα στον προγραμματιστή.

Το Android Studio παρέχει έτοιμα εργαλεία δημιουργίας βάσης δεδομένων στο κινητό του χρήστη. Στην συγκεκριμένη περίπτωση δημιουργείται βάση δεδομένων με το όνομα «sights.db». Κάθε φορά που ο χρήστης ανοίγει την εφαρμογή ελέγχεται αν έχει δημιουργηθεί η βάση δεδομένων με αυτό το όνομα. Σε περίπτωση που δεν έχει δημιουργηθεί, όπως γίνεται την πρώτη φορά για παράδειγμα, δημιουργείται και διαβάζεται αμέσως. Σε αντίθετη περίπτωση που εντοπίζεται η βάση με το συγκεκριμένο όνομα, δεν δημιουργείται καινούρια αλλά διαβάζονται τα δεδομένα από αυτή τη βάση δεδομένων.

Τα δεδομένα που προστέθηκαν σε αυτή τη βάση δεδομένων είναι δύο πίνακες. Ο πρώτος πίνακας είναι έχει να κάνει με την δημιουργία του πίνακα των σημείων, που περιέχει τις πληροφορίες τους. Η δημιουργία του πίνακα αυτού έγινε με εντολή SQL, όπως φαίνεται στο Απόσπασμα 2.

```
CREATE TABLE sights(
_id INTEGER PRIMARY KEY AUTOINCREMENT,
longitude REAL NOT NULL,
latitude REAL NOT NULL,
category INTEGER NOT NULL,
name TEXT NOT NULL,
picture TEXT,
address TEXT,
famous INTEGER NOT NULL DEFAULT 0,
tour_time INTEGER NOT NULL DEFAULT 5,
price INTEGER NOT NULL DEFAULT 0,
telephone TEXT,
can_delete INTEGER DEFAULT 0,
seen INTEGER DEFAULT 0,
rating REAL DEFAULT 0,
web_link TEXT,
dist_time INTEGER DEFAULT 0);
```

Απόσπασμα 2: Δημιουργία πίνακα σημείων «sights»

Με την εντολή αυτή δημιουργείται ένας πίνακας με 16 στήλες. Αναλυτικότερα όσον αφορά τις στήλες του πίνακα που συμπληρώνονται με τα δεδομένα των πληροφοριών των σημείων αυτές καταγράφονται παρακάτω (με τον τύπο δεδομένων σε παρένθεση):

- **_id (integer)**, η οποία συμπληρώνεται αυτόματα με το id της κάθε εγγραφής, δηλαδή του κάθε σημείου,
- **longitude (real)**, η οποία περιέχει το γεωγραφικό μήκος για κάθε σημείο,
- **latitude (real)**, η οποία περιέχει το γεωγραφικό πλάτος για κάθε σημείο,
- **category (integer)**, στην οποία καταγράφεται η κατηγορία του κάθε σημείου με συγκεκριμένες τιμές:
  - 0 για την κατηγορία «Μουσεία»,
  - 1 για την κατηγορία «Αξιοθέατα»,
  - 2 για την κατηγορία «Εκκλησίες»,
  - 3 για την κατηγορία «Άλλα»,
- **name (text)**, με το όνομα του σημείου ενδιαφέροντος,
- **address (text)**, με την διεύθυνση του σημείου ενδιαφέροντος,
- **famous (integer)**, με τις τιμές «0» (ΟΧΙ) αν δεν είναι δημοφιλές και «1» (ΝΑΙ) αν είναι,
- **tour_time (integer)**, με τον χρόνο που απαιτείται για να δεις γρήγορα το σημείο ενδιαφέροντος (σε λεπτά),
- **price (integer)**, με την τιμή εισιτηρίου του σημείου (σε ευρώ €),
- **telephone (text)**, με το τηλέφωνο του σημείου, όποτε αυτό είναι διαθέσιμο,
- **web_link (text)**, με το link για περισσότερες πληροφορίες για το σημείο,
- **can_delete (integer)**, με τις τιμές «1» (ΝΑΙ) αν το σημείο προστέθηκε από τον χρήστη και «0» (ΟΧΙ) για τα υπόλοιπα,
- **picture (text)**, με το όνομα του αρχείου της εικόνας, που είναι της μορφής «picID.jpg»,
- **seen (integer)**, με τις τιμές «0» (ΟΧΙ) αν ο χρήστης δεν έχει επισκεφθεί το σημείο και «1» (ΝΑΙ) αν το έχει επισκεφθεί,
- **rating (real)**, με την βαθμολογία που έχει καταχωρίσει ο χρήστης για το κάθε σημείο,
- **dist_time (integer)**, στην οποία καταχωρείται ο χρόνος που χρειάζεται σε λεπτά για να περπατήσει κανείς μια απόσταση και η στήλη αυτή βοηθάει στους υπολογισμούς που γίνονται στο τέλος για την εξαγωγή των αποτελεσμάτων.

Ο πίνακας αυτός είναι προφανές ότι με την πρώτη εκκίνηση της εφαρμογής δημιουργείται στο κινητό του χρήστη και αμέσως συμπληρώνεται με τα αντίστοιχα δεδομένα, δηλαδή 267 εγγραφές (δηλαδή 267 γραμμές), όσα δηλαδή και τα σημεία

ενδιαφέροντος που συλλέχθηκαν, όπως αναφέρεται σε προηγούμενο κεφάλαιο, με κάθε εγγραφή να περιέχει τις πληροφορίες κάθε σημείου.

Όσον αφορά τώρα τον δεύτερο πίνακα, αυτός περιέχει τον χρόνο (σε λεπτά), που χρειάζεται ο χρήστης να πάει περπατώντας από κάθε σημείο προς όλα τα άλλα. Ο πίνακας αυτός έχει μία στήλη (_id) που καταγράφεται το id κάθε σημείου και άλλες 267 (με ονόματα T1, T2, T3, … , T267). Κάθε όνομα στήλης υποδηλώνει το id του κάθε σημείου. Κάθε εγγραφή (γραμμή του πίνακα) επομένως περιέχει τον χρόνο που χρειάζεται κανείς να πάει περπατώντας από το σημείο με κωδικό που γράφεται στην στήλη _id προς όλα τα υπόλοιπα. Για την καλύτερη κατανόηση του πίνακα αυτού παρουσιάζεται παρακάτω ένα παράδειγμα.

_id	T1	T2	T3	T4	T5	…	T267
1	0	3	3	27	17	…	8
2	3	0	1	23	13	…	6
…	…	…	…	…	…	…	…
267	8	6	7	20	10	7	0

Πίνακας 2: Παράδειγμα πίνακα χρόνων «times»

Για την δημιουργία του πίνακα αυτού, που ονομάστηκε «times», δεν χρησιμοποιήθηκε ένας συγκεκριμένος κώδικας, όπως παραπάνω, αλλά διαβάζει τον αριθμό των σημείων που έχουν προστεθεί στον παραπάνω πίνακα και δημιουργεί τόσες στήλες όσα και τα σημεία. Αυτή η μέθοδος μας δίνει την δυνατότητα να αλλάζουμε τον αριθμό των σημείων στον πίνακα «sights» χωρίς να αλλάζουμε και αυτόν τον πίνακα. Ο κώδικας που χρησιμοποιήθηκε για την δημιουργία του πίνακα αυτού φαίνεται στο Απόσπασμα 3.

```
Cursor cursor = db.query(SightsEntry.TABLE_NAME, null, null, null,
null, null, null);
int numberOfSights = cursor.getCount();
cursor.close();
StringBuilder SQL_CREATE_TIMES_TABLE = new StringBuilder();
SQL_CREATE_TIMES_TABLE.append("CREATE TABLE ")
                      .append(TimesEntry.TABLE_NAME)
                      .append("(")
                      .append(TimesEntry._ID)
                      .append(" INTEGER PRIMARY KEY AUTOINCREMENT");
for (int i = 1; i <= numberOfSights; i++) {
    SQL_CREATE_TIMES_TABLE.append(", T")
                      .append(i)
                      .append(" INTEGER");}
SQL_CREATE_TIMES_TABLE.append(");");
db.execSQL(SQL_CREATE_TIMES_TABLE.toString());
```

Απόσπασμα 3: Δημιουργία πίνακα "times"

Μετά την δημιουργία λοιπόν του πίνακα με τους χρόνους, συμπληρώνεται με τα απαραίτητα δεδομένα. Η διαδικασία συμπλήρωσης των δεδομένων γίνεται ξεχωριστά από την δημιουργία σε κάθε πίνακα. Στην περίπτωση των χρόνων διαβάζεται ένα αρχείο εξαγόμενο από το Microsoft Excel σε μορφή *.csv, από το οποίο παίρνει κάθε γραμμή και την προσθέτει σαν εγγραφή στον αντίστοιχο πίνακα. Κατά την διάρκεια λειτουργίας της εφαρμογής ο χρήστης μπορεί να προσθέσει/διαγράψει εγγραφές (σημεία ενδιαφέροντος) ή να ενημερώσει κάποια στοιχεία μιας εγγραφής. Όλες αυτές οι τροποποιήσεις αποθηκεύονται στην βάση δεδομένων και είναι διαθέσιμες κάθε φορά που ανοίγει η εφαρμογή.

Στην συνέχεια δημιουργήθηκε η κλάση με όνομα «SightsProvider.java», που καθορίζει τι εντολές SQL θα χρησιμοποιηθούν και για ποιον πίνακα. Πιο συγκεκριμένα καταγράφεται αν κάποια από τις εντολές SQL που αναφέρονται σε προηγούμενο κεφάλαιο θα χρησιμοποιηθεί σε ολόκληρο τον πίνακα ή σε εγγραφή. Αναλυτικότερα, η εντολή αναζήτησης (query) χρησιμοποιείται και για τους δύο

πίνακες, είτε ολόκληρους, είτε σε μεμονωμένες εγγραφές αφού μπορεί να χρειαστεί αναζήτηση και ολόκληρου του πίνακα για προβολή των δεδομένων είτε αναζήτηση μιας εγγραφής με βάση κάποια κριτήρια. Η εντολή εισαγωγής (insert) χρησιμοποιείται και στους δύο πίνακες, καθώς θα δίνεται στον χρήστη η επιλογή προσθήκης νέου σημείου και κατ' επέκταση και η προσθήκη των αντίστοιχων χρόνων από το σημείο αυτό προς όλα τα άλλα στον πίνακα των χρόνων. Η εντολή διαγραφής (delete) και η εντολή ενημέρωσης (update) χρησιμοποιούνται και στους δύο πίνακες, αλλά για εγγραφές, καθώς δεν υπάρχει πιθανότητα επιθυμίας διαγραφής ή ενημέρωσης ολόκληρου πίνακα.

Με την ολοκλήρωση και αυτής της κλάσης ολοκληρώθηκε η διαδικασία δημιουργίας της βάσης δεδομένων στο κινητό του χρήστη. Επόμενο βήμα ήταν η προβολή των δεδομένων στην οθόνη. Όπως αναφέρεται παραπάνω χρησιμοποιήθηκε λίστα για την προβολή των σημείων. Η λίστα αυτή τοποθετήθηκε στην κύρια Δραστηριότητα, σε μια κλάση δηλαδή με όνομα «MainActivity.java». Μόλις ο χρήστης ανοίγει το κινητό του και έχει πλέον δημιουργηθεί η βάση δεδομένων, μετά από λίγα δευτερόλεπτα παρουσιάζεται η λίστα με τα σημεία στην οθόνη. Όπως περιγράφηκε σε προηγούμενο κεφάλαιο η λίστα δημιουργήθηκε με τέτοιο τρόπο ώστε να υπάρχει εξοικονόμηση μνήμης και η εφαρμογή να είναι πιο γρήγορη. Το ίδιο συνέβη και με την φόρτωση των δεδομένων. Αυτά όλα γίνονται στο παρασκήνιο και κατ' επέκταση δεν υπάρχει υπερφόρτωση της μνήμης.

Η μορφή της λίστας άλλαξε αρκετές φορές. Αρχικά η λίστα περιείχε πολλές πληροφορίες για κάθε σημείο, όπως το όνομα, την κατηγορία, την διεύθυνση και την τιμή. Όλα αυτά, σε συνδυασμό με τα μεγάλα ονόματα κάποιων σημείων ελάττωναν τον αριθμό των στοιχείων της λίστας που φαίνονταν στην οθόνη. Για τον λόγο αυτό αφαιρέθηκαν αρκετές πληροφορίες, μέχρι που κατέληξε στο τελικό αποτέλεσμα στο οποίο ο χρήστης βλέπει μόνο το όνομα και την κατηγορία του σημείου. Στην συνέχεια προστέθηκε και η εικόνα κάθε σημείου, κάτι που προκαλεί μεγαλύτερη εντύπωση στον χρήστη. Τέλος, πριν ολοκληρωθεί η κύρια Δραστηριότητα, δημιουργήθηκε το μενού πάνω δεξιά ώστε να προστεθούν οι επιπλέον λειτουργίες που θα περιγραφούν στην συνέχεια και συγχρονίστηκαν τα στοιχεία της λίστας, ώστε όταν ο χρήστης επιλέγει ένα από αυτά να κατευθύνεται σε μια καινούρια δραστηριότητα στην οποία θα μπορεί να έχει πρόσβαση σε όλες τις πληροφορίες του συγκεκριμένου σημείου.

Επόμενο βήμα ήταν η δημιουργία αυτής της κλάσης στην οποία φαίνονται όλες οι πληροφορίες του σημείου, το οποίο επέλεξε ο χρήστης από τη λίστα. Η κλάση αυτή ονομάστηκε «SightInfoActivity.java». Εδώ ο χρήστης μπορεί να δει το όνομα του σημείου, την αξιολόγηση, την κατηγορία που ανήκει, την διεύθυνση, το τηλέφωνο, την τιμή εισιτηρίου, τον χρόνο περιήγησης, καθώς και το αν είναι το σημείο δημοφιλές ή όχι. Όλα αυτά ουσιαστικά, εκτός από την αξιολόγηση, η οποία φαίνεται σε 5 αστέρια, αποτυπώνονται σε πλαίσια κειμένου. Κάθε φορά που ο χρήστης επιλέγει ένα σημείο από την λίστα, αποθηκεύεται ο κωδικός id του σημείου και στέλνεται από την MainActivity στην SightInfoActivity. Εκεί γίνεται αναζήτηση στον πίνακα των σημείων το σημείο με αυτόν τον κωδικό και προβάλλονται οι αντίστοιχες πληροφορίες στα κατάλληλα πλαίσια κειμένου.

Εκτός από τις πληροφορίες όμως του σημείου, προστέθηκε στο κάτω μέρος της οθόνης ένα κουμπί με το κείμενο «ΣΥΝΔΕΣΜΟΣ». Αυτό το κουμπί ουσιαστικά ανακατευθύνει τον χρήστη στο προεπιλεγμένο πρόγραμμα περιήγησης της συσκευής του και από εκεί ανοίγει η ιστοσελίδα του σημείου, με βάση τις πληροφορίες που υπάρχουν στην στήλη «web_link» του πίνακα των σημείων. Σε όποια σημεία δεν υπάρχει σύνδεσμος για ιστοσελίδα, απλά το κουμπί γίνεται κρυφό και δεν εμφανίζεται στον χρήστη.

Επιπλέον, στις πληροφορίες του σημείου φαίνεται και η εικόνα του. Όλες οι εικόνες έχουν καταχωρηθεί σε ένα φάκελο με όνομα «pic1.jpg», «pic2.jpg» κλπ. Οι εικόνες αυτές συλλέχθηκαν από το διαδίκτυο με αντίστοιχη αναζήτηση του κάθε σημείου. Στον πίνακα των σημείων στην στήλη «picture», περιέχεται το όνομα της εικόνας. Η εφαρμογή διαβάζει το όνομα και ψάχνει την εικόνα στον φάκελο. Αν το όνομα υπάρχει, δηλαδή έχει καταχωρηθεί όνομα εικόνας στον πίνακα, τότε παρουσιάζεται η εικόνα στις πληροφορίες του σημείου. Σε αντίθετη περίπτωση, έχει

καταχωρηθεί μια προεπιλεγμένη εικόνα, η οποία εμφανίζεται στα σημεία που δεν έχουν δεδομένα εικόνας. Και στις δύο περιπτώσεις ο χρήστης έχει τη δυνατότητα να επιλέξει την εικόνα, η οποία ανοίγει σε ένα παράθυρο στο κανονικό της μέγεθος και μπορεί να την δει σε μεγέθυνση.

Τελευταία λειτουργία που περιέχεται στις πληροφορίες του σημείου είναι ένα κουτάκι, το οποίο μπορεί ο χρήστης να το επιλέξει ή όχι και υποδηλώνει αν ο χρήστης έχει επισκεφθεί το σημείο. Στην περίπτωση το κουτί δεν είναι επιλεγμένο η τιμή στην στήλη «seen» της βάσης δεδομένων είναι 0 (ΟΧΙ). Όταν ο χρήστης επιλέξει το κουτί η τιμή ενημερώνεται σε 1 (ΝΑΙ) και αυτόματα ανοίγει ένα παράθυρο διαλόγου. Σε αυτό το παράθυρο ο χρήστης επιλέγει την αξιολόγηση του σημείου (σε αστέρια από 1 έως 5) και σημειώνει πόσο χρόνο χρειάστηκε (σε λεπτά) για να δει το αξιοθέατο. Στο κάτω μέρος του παραθύρου έχει ένα κουμπί «ΟΚ» το οποίο μόλις επιλεχθεί από τον χρήστη, λαμβάνονται οι τιμές που επέλεξε για την αξιολόγηση και τον χρόνο περιήγησης (αν επέλεξε), ενημερώνονται οι πληροφορίες του συγκεκριμένου σημείου στην βάση δεδομένων και κλείνει το παράθυρο. Αυτόματα εμφανίζονται οι ενημερωμένες πληροφορίες στην Δραστηριότητα.

Επόμενη δραστηριότητα που δημιουργήθηκε ήταν ο χάρτης με όνομα «MapActivity.java», στον οποίο φαίνονται όλα τα σημεία ενδιαφέροντος. Από την Android παρέχονται κάποια εικονίδια δωρεάν για χρήση σε εφαρμογές. Για τον λόγο αυτό χρησιμοποιήθηκε ένα εικονίδιο, που να παραπέμπει σε χάρτη, ώστε να χρησιμοποιηθεί σαν κουμπί μετάβασης από την αρχική λίστα στην δραστηριότητα του χάρτη. Το κουμπί αυτό τοποθετήθηκε στο μενού της αρχικής λίστας και με την επιλογή του ανοίγει η δραστηριότητα του χάρτη.

Η προσθήκη του χάρτη έγινε με έτοιμα εργαλεία που παρέχει η Google. Ωστόσο, για την εμφάνιση του χάρτη είναι απαραίτητη η χρήση διαδίκτυο, τουλάχιστον την πρώτη φορά. Ουσιαστικά, την πρώτη φορά που εμφανίζεται ο χάρτης λαμβάνονται τα δεδομένα από την Google και αποθηκεύονται οι πληροφορίες που προβλήθηκαν. Έτσι τις επόμενες φορές δεν χρειάζεται η χρήση διαδικτύου. Ο προγραμματιστής έχει δική του σελίδα στην οποία έχει καταχωρίσει τις βασικές πληροφορίες της εφαρμογής, όπως το όνομά της και από εκεί μπορεί να δει τι εργαλεία χρησιμοποιούνται στην εφαρμογή και πόσα δεδομένα λαμβάνονται ανά πάσα στιγμή.

Ο χάρτης ανοίγει σε συγκεκριμένη προβολή, η οποία μπορεί να καθοριστεί από τον προγραμματιστή. Σαν προβολή χάρτη νοείται το αρχικό στιγμιότυπο του χάρτη που εμφανίζεται όταν ανοίγει η αντίστοιχη δραστηριότητα. Αυτό καθορίζει πόσα δεδομένα θα ληφθούν από το διαδίκτυο και εξαρτάται από δύο κύριες μεταβλητές, το κέντρο (target) και την κλίμακα (zoom). Η πρώτη μεταβλητή έχει να κάνει με τις γεωγραφικές συντεταγμένες του κεντρικού σημείου της οθόνης και η δεύτερη με την κλίμακα του χάρτη. Όσον αφορά την κλίμακα του χάρτη, πρόκειται για ακέραιους αριθμούς που ξεκινούν από το 1. Έτσι όσο ο αριθμός μεγαλώνει, μεγαλώνει και η κλίμακα. Παρακάτω φαίνονται κάποια παραδείγματα του αριθμού της κλίμακας και στο Απόσπασμα 4 η προεπιλεγμένη προβολή του χάρτη:

- 1: Επίπεδο Πλανήτη
- 5: Επίπεδο Ηπείρου
- 10: Επίπεδο Πόλης
- 15: Επίπεδο Οδού
- 20: Επίπεδο κτιρίου

```
//Starting screen of Athens
static final CameraPosition ATHENS = CameraPosition.builder()
.target(new LatLng(37.9667, 23.7202)).zoom(13).build();
```

Απόσπασμα 4: Ορισμός προεπιλεγμένης προβολής χάρτη

Παράλληλα με την συλλογή των δεδομένων για την προβολή του χάρτη στην οθόνη, στο παρασκήνιο συλλέγονται και δεδομένα από τον πίνακα των σημείων της βάσης δεδομένων. Δημιουργείται στη δραστηριότητα αυτή ένας υποπίνακας του πίνακα σημείων, ο οποίος περιέχει μόνο τις στήλες id, όνομα και κατηγορία του

σημείου καθώς και τις γεωγραφικές τους συντεταγμένες. Όταν ο υποπίνακας δημιουργηθεί, διαβάζονται ένα – ένα τα στοιχεία και αυτόματα δημιουργούνται σημεία πάνω στο χάρτη της Google με βάση τις γεωγραφικές συντεταγμένες. Το εργαλείο αυτό παρέχεται μαζί με τον χάρτη και υπάρχει η δυνατότητα προβολής επιπλέον πληροφοριών σε ένα «συννεφάκι», όταν ο χρήστης επιλέξει ένα σημείο. Στην παρούσα περίπτωση επιλέχθηκε να εμφανίζεται το όνομα και η κατηγορία του σημείου. Παράλληλα, με την επιλογή ενός σημείου εμφανίζονται αυτόματα δύο κουμπιά στο κάτω δεξιά μέρος της οθόνης, εργαλείο που παρέχεται και αυτό μαζί με τον χάρτη. Και τα δύο κουμπιά ανοίγουν την προεπιλεγμένη εφαρμογή χαρτών της συσκευής. Το πρώτο εμφανίζει την διαδρομή (σε μορφή πλοήγησης) από την τοποθεσία που βρίσκεται εκείνη τη στιγμή ο χρήστης προς το επιλεγμένο σημείο και το δεύτερο απλά εμφανίζει το επιλεγμένο σημείο.

Ωστόσο, ο χρήστης μπορεί να θέλει να δει περισσότερες πληροφορίες για το επιλεγμένο σημείο στο χάρτη. Οπότε πρέπει να έχει τη δυνατότητα να μεταβεί στην δραστηριότητα του κάθε σημείου που περιέχει τις πληροφορίες του, χωρίς να επιστρέψει στην αρχική λίστα. Έτσι, όταν επιλεχθεί ένα σημείο και εμφανιστούν οι πληροφορίες του στο χάρτη, ο χρήστης θα μπορεί να επιλέξει το «συννεφάκι» που εμφανίζεται. Αυτόματα αποθηκεύεται ο κωδικός id του σημείου που επιλέχθηκε, ο χρήστης μεταβαίνει στην «SightInfoActivity» και με βάση το id εμφανίζονται οι κατάλληλες πληροφορίες.

Επίσης, ο χρήστης μπορεί να βρει ένα σημείο από την λίστα και βλέποντας τις πληροφορίες να θέλει να το δει στο χάρτη. Αντίστοιχα λοιπόν όπως και στην κύρια δραστηριότητα, προστέθηκε ένα κουμπί με ίδιο σύμβολο του χάρτη στο μενού της «SightInfoActivity», από το οποίο δίνεται η δυνατότητα μετάβασης από τις πληροφορίες του σημείου στον χάρτη. Επειδή όμως η μετάβαση στον χάρτη μπορεί να γίνει και από την κύρια δραστηριότητα, κάθε φορά που ανοίγει η δραστηριότητα του χάρτη, ελέγχεται αν υπάρχει κωδικός id. Σε περίπτωση που υπάρχει τότε με βάση τις συντεταγμένες του σημείου ορίζεται το κέντρο του χάρτη και με zoom ίσο με 15 ο χάρτης εστιάζει σε αυτό. Σε αντίθετη περίπτωση η μετάβαση έχει γίνει μέσω της αρχικής λίστας, οπότε εμφανίζεται η προεπιλεγμένη προβολή χάρτη με τα σημεία.

Απαραίτητη λειτουργία επίσης όταν υπάρχει χάρτης είναι και η προβολή της τοποθεσίας του χρήστη, αφού μπορεί, για παράδειγμα, να χρειαστεί να δει που βρίσκεται και τι υπάρχει κοντά του. Ωστόσο, για την χρήση των δεδομένων τοποθεσίας της συσκευής, ο χρήστης πρέπει να αποδεχθεί άδεια τοποθεσίας. Για τον λόγο αυτό, με την πρώτη εκκίνηση της εφαρμογής εμφανίζεται ένα παράθυρο στο οποίο ο χρήστης πρέπει να αποδεχθεί να χρησιμοποιηθεί η τοποθεσία του. Σε περίπτωση που αρνηθεί, συνεχίζεται η εφαρμογή κανονικά, αφού δεν χρειάζεται άμεσα η άδεια αυτή. Όμως, όταν ο χρήστης πάει να χρησιμοποιήσει την λειτουργία της τοποθεσίας στον χάρτη, αν δεν έχει αποδεχθεί την άδεια, ανοίγει πάλι το παράθυρο αυτό και δεν επιτρέπεται ο εντοπισμός της τοποθεσίας αν δεν αποδεχθεί την άδεια.

Σε περίπτωση που αποδεχθεί την άδεια, μπορεί να χρησιμοποιήσει το κλασσικό κουμπί που χρησιμοποιείται συνήθως στο κάτω δεξιά μέρος του χάρτη, το οποίο εντοπίζει την τοποθεσία του. Το κουμπί αυτό προστέθηκε χειροκίνητα για λόγους αισθητικής, καθώς το κουμπί που παρέχεται γι’ αυτή τη λειτουργία ήταν πολύ μικρό. Ο εντοπισμός γίνεται με κατάλληλες εντολές που παρέχει η Android και χρησιμοποιεί τις ρυθμίσεις της συσκευής. Πιο συγκεκριμένα, ο εντοπισμός της θέσης μπορεί να γίνει με πολλούς τρόπους. Αρχικά μπορούν να χρησιμοποιηθούν δίκτυα wi-fi που μπορεί να είναι συνδεδεμένη η συσκευή. Παράλληλα μπορούν να χρησιμοποιηθούν δεδομένα κινητής τηλεφωνίας, ενώ ο βασικός τρόπος είναι η χρήση GPS. Όσο πιο πολλές είναι οι μέθοδοι που χρησιμοποιούνται τόσο ακριβέστερος και γρηγορότερος είναι ο προσδιορισμός της θέσης. Το μόνο μειονέκτημα είναι η κατανάλωση της μπαταρίας, που όπως είναι φυσικό είναι μεγαλύτερη σε περίπτωση που χρησιμοποιούνται ταυτόχρονα πολλοί τρόποι. Στην συγκεκριμένη περίπτωση αρκεί το GPS και η σύνδεση στο διαδίκτυο είτε μέσω wifi είτε μέσω δεδομένων κινητής τηλεφωνίας.

Επομένως είναι προφανές ότι ο χρήστης για να χρησιμοποιήσει το κουμπί αυτό πρέπει να ενεργοποιήσει το εντοπισμό της τοποθεσίας (GPS), μέσω των ρυθμίσεων της συσκευής. Οπότε όταν ο χρήστης πατήσει το πλήκτρο, ελέγχεται αν ο εντοπισμός είναι ενεργός. Αν είναι τότε ο χάρτης μετακινείται στο σημείο της τοποθεσίας του. Ουσιαστικά, μετακινείται η προβολή του χάρτη με κεντρικό σημείο τις συντεταγμένες της τοποθεσίας της συσκευής και με κλίμακα ίση με 15, δηλαδή στο επίπεδο της οδού. Επιπλέον δημιουργείται και ένα σημείο, το οποίο υποδηλώνει την ακριβή τοποθεσία του και φαίνεται ένας κύκλος γύρω από αυτό υποδηλώνοντας την ακρίβεια του προσδιορισμού της θέσης. Αυτό το σημείο μετακινείται ανάλογα με τις αλλαγές στην τοποθεσία κάθε 1 δευτερόλεπτο, χρόνος που καθορίστηκε με την κατάλληλη εντολή. Σε περίπτωση τώρα που δεν είναι ενεργοποιημένο το GPS, εμφανίζεται ένα παράθυρο στο οποίο ζητείται από τον χρήστη να το ενεργοποιήσει. Αν αποδεχθεί, αμέσως ενεργοποιείται το GPS, εντοπίζεται η τοποθεσία του και ο χάρτης μετακινείται στην τοποθεσία που βρίσκεται. Αν δεν το ενεργοποιήσει, απλά κλείνει το παράθυρο χωρίς να συμβεί τίποτα.

Πιο συγκεκριμένα, μετά την αποδοχή της άδειας και την ενεργοποίηση του GPS και του διαδικτύου, ο εντοπισμός της τοποθεσίας γίνεται μέσω ενός αντικειμένου «GoogleApiClient». Αυτός είναι ο τρόπος ώστε να υπάρξει πρόσβαση σε ένα από τα εργαλεία (API) που παρέχονται στη βιβλιοθήκη Google Play Services. Αυτό το αντικείμενο διαχειρίζεται αυτόματα τη σύνδεση με τις Υπηρεσίες Google Play, βάζοντας σε σειρά κάποια αιτήματα και στέλνοντάς τα όταν συνδεθεί η συσκευή στο διαδίκτυο. Τα αντικείμενα αυτά είναι σχετικά φθηνά και εύκολα στη χρήση, γι' αυτό και χρησιμοποιούνται ευρέως. Παρακάτω στο Απόσπασμα 5 φαίνεται ο κώδικας της δραστηριότητας του χάρτη με το οποίο δημιουργείται και προστίθενται οι κατάλληλες λειτουργίες σε ένα αντικείμενο «GoogleApiClient».

```
GoogleApiClient mGoogleApiClient = new GoogleApiClient.Builder(this)
//Δημιουργία του αντικειμένου «mGoogleApiClient»
     .addApi(LocationServices.API)
//Προσθήκη του API για εύρεση τοποθεσίας
     .addConnectionCallbacks(this)
//Προσθήκη εντολών για καθορισμό της σύνδεσης στο διαδίκτυο
     .addOnConnectionFailedListener(this)
//Προσθήκη εντολής για αποτυχία σύνδεσης στο διαδίκτυο
     .build();
//Έναρξη της λειτουργίας του
```

Απόσπασμα 5: Δημιουργία "GoogleApiClient"

Για την χρήση του αντικειμένου αυτού πρέπει να καθοριστεί τι θα γίνει σε περίπτωση που δημιουργηθεί σύνδεση στο διαδίκτυο, σε περίπτωση που σταματήσει προσωρινά η σύνδεση και σε περίπτωση που διακοπεί ή αποτύχει εντελώς. Οι περιπτώσεις αυτές καθορίζονται με έτοιμες εντολές, οι οποίες παρέχονται μαζί με το «GoogleApiClient». Στις δύο περιπτώσεις διακοπής της σύνδεσης αυτόματα γίνεται προσπάθεια επανασύνδεσης με εμφάνιση αντίστοιχων μηνυμάτων στην οθόνη. Η περίπτωση επιτυχούς σύνδεσης φαίνεται παρακάτω στο Απόσπασμα 6, με πλήρεις εξηγήσεις κάθε γραμμής κώδικα.

```
mLocationRequest = LocationRequest.create();
//Δημιουργείται το αίτημα για τον προσδιορισμό της τοποθεσίας
mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
//Ορίζεται η ακρίβεια του προσδιορισμού σε μεγάλη ακρίβεια (χρήση GPS
και διαδικτύου)
mLocationRequest.setInterval(10); // Update location every second
//Προσδιορισμός του χρόνου που θα στέλνεται το αίτημα σε 1
δευτερόλεπτο
LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiCl
ient, mLocationRequest, this);
//Αποστολή του αιτήματος «mLoctionRequest» μέσω του
«mGoogleApiClient»
myLocation =
LocationServices.FusedLocationApi.getLastLocation(mGoogleApiClient);
//Παραλαβή της απάντησης του αιτήματος και αποθήκευση της τοποθεσίας
στη μεταβλητή «myLocation»
if (myLocation != null) {
    LatLng myLatLng = new LatLng(myLocation.getLatitude(),
myLocation.getLongitude());
//Αν η τοποθεσία δεν είναι κενή και ήρθε απάντηση τότε αποθηκεύονται
οι συντεταγμένες στην μεταβλητή «myLatLng»
if (mCurrentSightUri == null) {
googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(myLatLng,
15));}}
//Αν η μετάβαση έχει γίνει από την κύρια δραστηριότητα η προβολή του
χάρτη να αλλάξει με κέντρο αυτές τις συντεταγμένες και κλίμακα «15».
```

Απόσπασμα 6: Ενέργειες σε περίπτωση επιτυχούς σύνδεσης του "GoogleApiClient"

Επιπλέον, χρησιμοποιήθηκαν δύο ακόμα εργαλεία, που παρέχονται μαζί με τον χάρτη. Το πρώτο εργαλείο ήταν η μορφή του χάρτη, τι λεπτομέρειες και τι πληροφορίες μπορεί κανείς να δει μέσα στο χάρτη. Για τον λόγο αυτό προστέθηκε ένα κουμπί στο πάνω δεξιά μέρος της οθόνης. Η προεπιλεγμένη μορφή του χάρτη είναι η Google Streets, εμφανίζει δηλαδή δεδομένα οδών, κτιρίων, περιοχών και τοποθεσιών. Μόλις ο χρήστης πατήσει το κουμπί, η μορφή του χάρτη αλλάζει και φαίνεται σαν εικόνα από δορυφόρο. Ο χάρτης δηλαδή μπορεί να σου παρέχει πληροφορίες για το φυσικό περιβάλλον, τα χρώματα κλπ. Παράλληλα επιλέχθηκαν τα κατάλληλα εικονίδια για να τοποθετηθούν στο κουμπί ώστε να είναι προφανής η λειτουργία του.

Τέλος, το τελευταίο εργαλείο που χρησιμοποιήθηκε είναι η πυξίδα. Το εργαλείο αυτό παρέχεται από τους χάρτες με έτοιμη εντολή και δεν χρειάστηκε να γίνει κάτι επιπλέον προγραμματιστικά. Η πυξίδα αυτή εμφανίζεται στο πάνω αριστερό μέρος του χάρτη μόνο σε περίπτωση που αλλάξει ο προσανατολισμός του και υποδηλώνει που είναι ο βορράς. Όταν ο χρήστης κάνει κλικ στην πυξίδα, ο βορράς επιστρέφει στην προεπιλεγμένη θέση, δηλαδή ο προσανατολισμός του χάρτη επιστρέφει με τον βορρά προς τα πάνω.

Προσδιορισμός της προτεινόμενης διαδρομής

Μετά την ολοκλήρωση και αυτής της δραστηριότητας, τα βασικά σημεία της εφαρμογής είναι πλέον έτοιμα. Η βάση δεδομένων έχει δημιουργηθεί και ο χρήστης μπορεί να την προβάλει είτε σε λίστα, είτε σε χάρτη και να βλέπει ό,τι δεδομένα επιθυμεί. Ωστόσο, μέχρι τώρα όλα αυτά μπορεί να τα βρει και σε άλλες εφαρμογές πολύ πιο οργανωμένες, με μεγαλύτερη βάση δεδομένων και με πολλά χρόνια εμπειρίας. Αυτό που θα διαφοροποιήσει την συγκεκριμένη εφαρμογή είναι ο υπολογισμός μια διαδρομής με βάση κάποια κριτήρια. Επομένως μετά τον σχεδιασμό της λειτουργίας αυτής, τοποθετήθηκε ένα κουμπί στο κάτω δεξιά μέρος της κύριας λίστας, το οποίο παραπέμπει τον χρήστη σε μια νέα δραστηριότητα, που ονομάστηκε «CriteriaActivity.java».

Η δραστηριότητα αυτή ουσιαστικά παίζει τον ρόλο της μεταβατικής δραστηριότητας. Εδώ ο χρήστης θα καθορίσει τα κριτήρια σύμφωνα με τα οποία, θα

προγραμματιστεί η προτεινόμενη διαδρομή. Τα κριτήρια αυτά μετά από πολλές αλλαγές αποφασίστηκε να είναι όσο το δυνατόν λιγότερα και ο χρήστης να μην ξοδεύει πολύ χρόνο για να αποφασίσει. Έτσι, χρησιμοποιήθηκαν τα απολύτως απαραίτητα κριτήρια για τον καθορισμό της διαδρομής.

Αρχικά, επειδή πρόκειται για μια εφαρμογή που θα χρησιμοποιεί στα γρήγορα ο χρήστης, δηλαδή τη στιγμή που δεν έχει πολύ χρόνο προστέθηκε μία λίστα με συγκεκριμένους χρόνους. Στον χρήστη δίνεται η δυνατότητα να καθορίσει πόσο ελεύθερο χρόνο έχει ανάμεσα σε 3 επιλογές: 1, 2 ή 3 ώρες. Η επιλογή του αποθηκεύεται σε μια μεταβλητή που δέχεται συγκεκριμένες τιμές. Αυτό δεν συμβαίνει και με τα χρήματα που διαθέτει. Εδώ δίνεται η δυνατότητα στον χρήστη να πληκτρολογήσει ένα ποσό (σε €), που αποτελεί το ανώτερο ποσό που μπορεί να διαθέσει για την προτεινόμενη διαδρομή. Το ποσό αυτό καταχωρείται σε μια ακέραια μεταβλητή.

Στην συνέχεια, χρησιμοποιήθηκε ένας διακόπτης με ένα κείμενο δίπλα που γράφει «Ενδιαφέρομαι για τα δημοφιλή». Όταν ο χρήστης απενεργοποιεί τον διακόπτη το κείμενο αλλάζει σε «Δεν ενδιαφέρομαι για τα δημοφιλή». Ουσιαστικά αυτός ο διακόπτης καθορίζει αν η προτεινόμενη διαδρομή θα προτιμήσει τα πιο κοντινά πρώτα ή τα πιο δημοφιλή. Για παράδειγμα, αν υπάρχουν δύο κοντινά σημεία τα οποία δεν είναι δημοφιλή και ένα πιο μακρινό, αν ο χρήστης επιλέξει ότι ενδιαφέρεται για τα δημοφιλή, η προτεινόμενη διαδρομή θα αρχίσει από το δημοφιλές και αν έχει χρόνο μετά θα τον φέρει και σε ένα από τα πιο κοντινά. Σε αντίθετη περίπτωση, η προτεινόμενη διαδρομή θα αρχίσει από το πιο κοντινό και θα πηγαίνει στο πιο μακρινό. Αυτό θα γίνει πιο κατανοητό στη συνέχεια, που θα περιγραφεί αναλυτικά ο αλγόριθμος που χρησιμοποιήθηκε για την εξαγωγή της προτεινόμενης διαδρομής. Στην περίπτωση που ο χρήστης ενδιαφέρεται για τα δημοφιλή, καταχωρείται ο αριθμός «1» σε μια μεταβλητή, ενώ σε αντίθετη περίπτωση καταχωρείται ο αριθμός «0».

Στην συνέχεια, ακολουθούν οι 4 κατηγορίες σημείων, στις οποίες ο χρήστης έχει τη δυνατότητα να επιλέξει ή όχι ένα «ckeckbox» δίπλα σε αυτές. Έτσι ο χρήστης θα έχει τη δυνατότητα να επιλέξει τι κατηγορίες σημείων θέλει να περιέχει η προτεινόμενη διαδρομή. Για τον λόγο αυτό χρησιμοποιήθηκαν 4 μεταβλητές, μια για κάθε κατηγορία, με τις τιμές «1» και «0» αν επιθυμεί ή όχι αντίστοιχα να επισκεφθεί την κάθε κατηγορία.

Τέλος, ο χρήστης μπορεί να μην θέλει να ξεκινήσει την διαδρομή του από το σημείο που βρίσκεται εκείνη τη στιγμή, αλλά να την προετοιμάσει ίσως σε κάποιο άλλο μέρος που θα έχει πρόσβαση στο διαδίκτυο. Για τον λόγο αυτό, έπρεπε με κάποιο τρόπο ο χρήστης να προσδιορίσει τις συντεταγμένες του σημείου, από το οποίο επιθυμεί να ξεκινήσει την διαδρομή του. Έτσι, τοποθετήθηκε στο τέλος των κριτηρίων, ένα κουμπί το οποίο παραπέμπει τον χρήστη σε ένα χάρτη και από εκεί μπορεί να επιλέξει το σημείο έναρξης.

Ο χάρτης αυτός παρέχεται ως εργαλείο από την Google και ονομάζεται «PlacePicker». Το εργαλείο αυτό είναι σχεδιασμένο καθαρά για την επιλογή ενός σημείου και την λήψη των απαραίτητων στοιχείων που ενδιαφέρουν τον καθένα κάθε φορά. Ωστόσο, όταν ξεκινούσε ο χάρτης έδειχνε όλο τον πλανήτη. Για τον λόγο αυτό προτιμήθηκε ένα συγκεκριμένο προεπιλεγμένο στιγμιότυπο του χάρτη όπως φαίνετια στο Απόσπασμα 7, όπου εμφανίζεται ο κώδικας με τον οποίο δημιουργήθηκε ο χάρτης του «PlacePicker».

```
PlacePicker.IntentBuilder builder = new PlacePicker.IntentBuilder();
//Δημιουργία του αντικειμένου «PlacePicker»
LatLng bottomLeft = new LatLng(37.893668, 23.660787);
LatLng topRight = new LatLng(38.028669, 23.787422);
//Δημιουργία μεταβλητών για τον ορισμό του κάτω – αριστερά και του
πάνω – δεξιά σημείου του χάρτη
LatLngBounds mapBounds = new LatLngBounds(bottomLeft, topRight);
//Δημιουργία των ορίων του χάρτη με βάση τα παραπάνω σημεία
builder.setLatLngBounds(mapBounds);
//Προσθήκη αυτών των ορίων στο χάρτη του εργαλείου
Intent i = builder.build(this);
startActivityForResult(i, PLACE_PICKER_REQUEST);
//Έναρξη του χάρτη «PlacePicker»
```

Απόσπασμα 7: Δημιουργία του χάρτη του "PlacePicker"

Με τον όρο «μέρος» (Place) νοείται οποιοδήποτε σημείο ενδιαφέροντος που έχει δημιουργηθεί από οποιονδήποτε χρήστη, ιδιώτη ή εταιρεία και μπορεί κανείς να το βρει και μέσω του Google Places. Όσον αφορά το εργαλείο του «PlacePicker», ο χρήστης έχει την δυνατότητα να εντοπίσει την θέση του, να μετακινήσει τον χάρτη σε μια συγκεκριμένη θέση ή να αναζητήσει μια συγκεκριμένη διεύθυνση ή ένα συγκεκριμένο μέρος. Σε κάθε περίπτωση κοντά στο επιλεγμένο σημείο, εμφανίζεται στο κάτω μέρος μια λίστα με τα πιο κοντινά μέρη που παρέχει η Google, έτσι ώστε δίνεται η δυνατότητα να επιλεχθεί ένα ήδη υπάρχον μέρος. Αυτό θα γίνει πιο κατανοητό σε επόμενη ενότητα, όπου θα γίνει παρουσίαση της λειτουργίας της εφαρμογής. Ο χρήστης επιλέγει το σημείο που επιθυμεί και λαμβάνονται οι κατάλληλες πληροφορίες. Στην συγκεκριμένη περίπτωση, λαμβάνονται οι συντεταγμένες του σημείου, καθώς και η διεύθυνση. Οι συντεταγμένες του σημείου καταγράφονται σε δύο μεταβλητές που ορίζουν το σημείο έναρξης της προτεινόμενης διαδρομής. Όσον αφορά την διεύθυνση, δίπλα στο κουμπί «Χάρτης» τοποθετήθηκε ένα κείμενο το οποίο δεν είναι ορατό στον χρήστη, παρά μόνο όταν επιλεχθεί σημείο έναρξης. Όταν επιλεχθεί, εμφανίζεται στον χρήστη ενημερώνοντάς τον ότι επιλέχθηκε σημείο και παράλληλα αναφέρεται και η διεύθυνσή του.

Στο σημείο αυτό πρέπει να σημειωθεί ότι το εργαλείο αυτό για να λειτουργήσει σωστά απαιτείται σύνδεση στο διαδίκτυο. Αν την πρώτη φορά που θα χρησιμοποιηθεί το εργαλείο υπάρχει σύνδεση στο διαδίκτυο, αποθηκεύεται ο χάρτης. Τις επόμενες φορές, θα είναι ορατός και χωρίς σύνδεση στο διαδίκτυο, θα μπορεί να επιλεχθεί σημείο, θα μπορούν να ληφθούν οι συντεταγμένες του, αλλά δεν θα μπορεί να κάνει αναζήτηση σε μέρη, δεν θα του εμφανίζεται η λίστα με τα πιο κοντινά προτεινόμενα και δεν θα μπορούν να ληφθούν άλλες πληροφορίες του σημείου, όπως η διεύθυνσή του. Επομένως, η πρώτη φορά αρκεί να αποθηκευτεί ο χάρτης και στην συνέχεια θα μπορεί ο χρήστης να χρησιμοποιεί το εργαλείο, αφού τα μόνα απαραίτητα στοιχεία που απαιτούνται είναι οι συντεταγμένες του σημείου. Άρα, το εργαλείο αυτό ικανοποιεί τον στόχο της εφαρμογής, δηλαδή την χρήση της χωρίς την σύνδεση στο διαδίκτυο. Παρακάτω στο Απόσπασμα 8 φαίνεται ο κώδικας για την λήψη των στοιχείων του σημείου.

```
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
    if (requestCode == PLACE_PICKER_REQUEST && resultCode ==
RESULT_OK) {
       Place place = PlacePicker.getPlace(this, data);
//Λήψη του μέρους που επιλέχθηκε από τον χρήστη
       if (place == null) {
//Έλεγχος αν επιλέχθηκε μέρος
         Toast.makeText(CriteriaActivity.this,
R.string.no_place_selected, Toast.LENGTH_SHORT).show();
//Αν δεν επιλέχθηκε εμφανίζεται μήνυμα στον χρήστη
       } else {
         LatLng placeLatLng = place.getLatLng();
         startLong = placeLatLng.longitude;
         startLat = placeLatLng.latitude;
//Λήψη συντεταγμένων και καταχώρησή τους σε μεταβλητές
         String newSightAddress = String.valueOf(place.getAddress());
         String[] addressParts = newSightAddress.split(",");
//Λήψη διεύθυνσης του σημείου
         StringBuilder textForStart = new
StringBuilder(getResources().getString(R.string.start_point_selected)
);
         if (!addressParts[0].isEmpty()){
textForStart.append("\n").append(getResources().getString(R.string.ad
dress)).append(addressParts[0]);}
//Έλεγχος αν είναι διαθέσιμη η διεύθυνση και δημιουργία μεταβλητής
κειμένου
         textStartPoint.setText(textForStart);
         textStartPoint.setVisibility(View.VISIBLE);
//Προσθήκη της μεταβλητής στο κείμενο και ενεργοποίηση της ορατότητάς
του
}}}
```

Απόσπασμα 8: Λήψη στοιχείων του επιλεγμένου μέρους

Στο μενού της δραστηριότητας των κριτηρίων προστέθηκε ένα κουμπί αποθήκευσης. Αυτό αποθηκεύει τα κριτήρια και ανοίγει μια νέα δραστηριότητα, την δραστηριότητα των αποτελεσμάτων. Η δραστηριότητα αυτή ονομάστηκε «ResultsActivity.java» και περιέχει την προτεινόμενη διαδρομή. Ωστόσο, για τον υπολογισμό της, είναι απαραίτητες οι μεταβλητές που δημιουργήθηκαν στην προηγούμενη δραστηριότητα. Αυτές περνάνε από την προηγούμενη σε αυτή την δραστηριότητα και αποθηκεύονται σε νέες μεταβλητές. Αυτές είναι:

- **freeTime_Crit**: «0» για 1 ώρα, «1» για 2 ώρες και «2» για 3 ώρες, τιμές που ορίζονται αυτόματα από την αναπτυσσόμενη λίστα της δραστηριότητας των κριτηρίων. Στην συνέχεια αυτού του κεφαλαίου αναφέρεται η μετατροπή της μεταβλητής αυτής σε χρόνο (λεπτά της ώρας).
- **budget**: το διαθέσιμο ποσό (σε €).
- **popular**: «1» αν επιλέχθηκε ενδιαφέρον στα δημοφιλή και «0» αν όχι.
- **museum**: «1» αν επιλέχθηκε η κατηγορία «μουσεία» και «0» αν όχι.
- **attraction**: «1» αν επιλέχθηκε η κατηγορία «αξιοθέατα» και «0» αν όχι.
- **church**: «1» αν επιλέχθηκε η κατηγορία «εκκλησίες» και «0» αν όχι.
- **other**: «1» αν επιλέχθηκε η κατηγορία «άλλα» και «0» αν όχι.
- **startLong**: το γεωγραφικό μήκος του σημείου έναρξης της διαδρομής.
- **startLat**: το γεωγραφικό πλάτος του σημείου έναρξης της διαδρομής.

Η προτεινόμενη διαδρομή επιλέχθηκε να προβληθεί σε λίστα. Πριν από την εμφάνιση της λίστας εμφανίζεται ένα μήνυμα ενημερώνοντας τον χρήστη ότι τα αποτελέσματα δεν είναι ακριβή και μπορεί να διαφέρουν από άνθρωπο σε άνθρωπο. Η λίστα αυτή θα περιέχει τα σημεία ενδιαφέροντος που μπορεί να επισκεφθεί ο

χρήστης σε σειρά. Επιλέγοντας όποιο σημείο επιθυμεί, θα υπάρχει η δυνατότητα να μεταβεί στην δραστηριότητα των πληροφοριών του σημείου (SightInfoActivity) και να δει παραπάνω πληροφορίες. Σε περίπτωση που η λίστα είναι κενή και δεν υπάρχει προτεινόμενη διαδρομή, για παράδειγμα όταν ο χρήστης βρίσκεται πολύ μακριά απ' όλα τα σημεία, εμφανίζεται ένα κείμενο στην οθόνη που τον ενημερώνει ότι δεν υπάρχουν αποτελέσματα και ότι πρέπει να αλλάξει τα κριτήρια. Για τον λόγο αυτό προστέθηκε στο μενού της δραστηριότητας αυτής η επιλογή «Αλλαγή κριτηρίων». Έτσι ο χρήστης μπορεί ανά πάσα στιγμή να επιλέξει αυτή την επιλογή, και να επιστρέψει στα κριτήρια, ώστε να αλλάξει ότι θέλει. Όταν ο χρήστης, επιλέγει αυτή την επιλογή, οι τιμές των μεταβλητών επιστρέφουν στην δραστηριότητα των κριτηρίων και γίνονται οι κατάλληλες ρυθμίσεις ώστε να φανούν οι επιλογές στην οθόνη. Το μόνο που πρέπει να καθορίσει πάλι ο χρήστης είναι το σημείο έναρξης.

Στο σημείο αυτό πρέπει να σημειωθεί ότι αν ο χρήστης δεν επιλέξει την επιλογή της αλλαγής των κριτηρίων από το μενού της δραστηριότητας, αλλά επιλέξει να πάει πίσω ή μέσω του αντίστοιχου πλήκτρου της συσκευής είτε μέσω του βέλους στο αριστερό μέρος του μενού, η εφαρμογή εμφανίζει ένα μήνυμα που ειδοποιεί τον χρήστη για την ενέργεια που θα πραγματοποιηθεί. Αν ο χρήστης αποδεχθεί την ενέργεια αυτή τον ανακατευθύνει στην αρχική λίστα. Η διαδρομή διαγράφεται και δεν υπάρχει η δυνατότητα να μεταβεί πάλι πίσω, παρά μόνο αν δημιουργήσει πάλι την ίδια διαδρομή μέσω των κριτηρίων.

Για τον υπολογισμό τώρα της προτεινόμενης διαδρομής οι διαδικασίες είναι αρκετές, που σημαίνει ότι χρειάζεται κάποιος χρόνος. Για τον λόγο αυτό, οι διαδικασίες αυτές ήταν αναγκαίο να γίνουν στο παρασκήνιο, ώστε να μην διακοπεί η λειτουργία της εφαρμογής. Παράλληλα για να γνωρίζει ο χρήστης ότι η εφαρμογή εκτελεί κάποιες ενέργειες και πρέπει να περιμένει προστέθηκε ένας κύκλος που γυρίζει σαν ένδειξη φόρτωσης. Το εργαλείο αυτό παρέχεται από την Android, χρησιμοποιείται κυρίως για αυτό το σκοπό και είναι προτιμότερο από το να βλέπει ο χρήστης απλά μια κενή οθόνη χωρίς να γνωρίζει αν συμβαίνει κάτι στο παρασκήνιο. Όταν ολοκληρωθεί η διαδικασία και εμφανιστεί η λίστα ή το μήνυμα της κενής λίστας, ο κύκλος αυτός γίνεται αόρατος στον χρήστη.

Με την ολοκλήρωση των διαδικασιών και την εμφάνιση της λίστας, όταν αυτή δεν είναι κενή, δημιουργήθηκε και εμφανίζεται ένα στρογγυλό κουμπί στο κάτω δεξιά μέρος της οθόνης. Το κουμπί αυτό ανακατευθύνει τον χρήστη στην προεπιλεγμένη εφαρμογή χαρτών της συσκευής του, όπου προστίθενται όλα τα σημεία σε μια διαδρομή και δίνονται οδηγίες πλοήγησης μεταξύ τους. Αυτές οι λεπτομέρειες διατυπώνονται σε μια διεύθυνση που διαβάζεται από την εφαρμογή χαρτών και καταχωρούνται τα δεδομένα. Η μορφή της διεύθυνσης φαίνεται παρακάτω:

```
https://www.google.com/maps/dir/?api=1&origin=35.0000,25.0000
&destination=35.0000,25.0000&waypoints=35.0000,25.0000|35.000,25.0000
&travelmode=walking
```

Όπως φαίνεται στην παραπάνω διεύθυνση αποτελείται από το πρώτο μέρος, που διατυπώνεται η επιθυμία να χρησιμοποιηθούν οι χάρτες της συσκευής. Στην συνέχεια ορίζεται το σημείο έναρξης της διαδρομής (origin), ο προορισμός ή το τελευταίο σημείο της διαδρομής (destination) και τα ενδιάμεσα σημεία (waypoints) χωριζόμενα με το σύμβολο «|». Τέλος, καταγράφεται και ο τρόπος μετάβασης, ο οποίος είναι το περπάτημα (travelmode = walking). Η διεύθυνση αυτή αποθηκεύτηκε σε μια μεταβλητή με το όνομα «uri». Η συνολική διαδικασία που πραγματοποιείται κατά την επιλογή του πλήκτρου πλοήγησης φαίνεται παρακάτω στο Απόσπασμα 9.

```
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
//Δημιουργία του αντικειμένου που καταχωρούνται τα δεδομένα για την έναρξη της εξωτερικής εφαρμογής και καταχώρηση της διεύθυνσης ως μεταβλητή uri
intent.setPackage("com.google.android.apps.maps");
//Ορισμός της διεύθυνσης της εξωτερικής εφαρμογής
try {
    startActivity(intent);
//Έναρξη της εξωτερικής εφαρμογής
} catch (ActivityNotFoundException innerEx) {
    Toast.makeText(ResultsActivity.this, "Please install a maps application", Toast.LENGTH_LONG).show();}
//Καθορισμός ενεργειών σε περίπτωση που δεν υπάρχει εγκατεστημένη εφαρμογή χαρτών στη συσκευή
```

Απόσπασμα 9: Έναρξη εξωτερικής εφαρμογής "χαρτών"

Όπως είναι προφανές λοιπόν, για την διατύπωση της διεύθυνσης είναι η απαραίτητη η γνώση των γεωγραφικών συντεταγμένων όλων των σημείων. Για τον λόγο αυτό με την έναρξη της δραστηριότητας δημιουργούνται δύο λίστες: μια για το γεωγραφικό πλάτος και μια για το γεωγραφικό μήκος. Κατά την διάρκεια υπολογισμού της προτεινόμενης διαδρομής ουσιαστικά προστίθενται οι συντεταγμένες των σημείων με τη σειρά και τέλος χρησιμοποιούνται αυτές οι λίστες για την διατύπωση της διεύθυνσης. Ως πρώτο σημείο προστίθενται οι συντεταγμένες του σημείου έναρξης, από τις αντίστοιχες μεταβλητές, όπως φαίνεται στο Απόσπασμα 10.

```
ArrayList<Double> long_points = new ArrayList<>();
ArrayList<Double> lat_points = new ArrayList<>();
//Δημιουργία των λιστών (με τύπο δεδομένων Double)
long_points.add(startLong);
lat_points.add(startLat);
//Προσθήκη των συντεταγμένων του σημείου έναρξης στις λίστες
```

Απόσπασμα 10: Δημιουργία και συμπλήρωση λιστών συν/νων

Πριν την έναρξη της διαδικασίας της προτεινόμενης διαδρομής, θα πρέπει να υπολογιστούν οι χρόνοι μετάβασης από κάθε σημείο προς όλα τα υπόλοιπα. Ευτυχώς, ο πίνακας με τους χρόνους περιέχει ακριβή δεδομένα για τον χρόνο μετάβασης από οποιοδήποτε σημείο προς τα υπόλοιπα. Επομένως ο μόνος χρόνος που πρέπει να υπολογιστεί είναι ο χρόνος μετάβασης από το σημείο έναρξης προς όλα τα υπόλοιπα για να επιλεχθεί η καλύτερη επιλογή.

Για τον υπολογισμό αυτών των χρόνων μετάβασης από το σημείο έναρξης προς όλα τα υπόλοιπα χρησιμοποιήθηκαν δύο τρόποι. Όταν ο χρήστης έχει σύνδεση στο διαδίκτυο χρησιμοποιούνται οι χρόνοι με το «Google Api Distance Matrix». Σε αντίθετη περίπτωση χρησιμοποιείται ο τύπος της Manhattan απόστασης, όπως ακριβώς και με την συλλογή των δεδομένων του πίνακα των χρόνων. Το απόσπασμα του κώδικα που χρησιμοποιήθηκε και στις δύο περιπτώσεις με αναλυτικές περιγραφές φαίνεται στο Παράρτημα. Μετά από δοκιμές των δύο τρόπων υπολογισμού των χρόνων, παρατηρήθηκε η απαίτηση περισσότερου χρόνου με το «Distance Matrix». Αυτό είναι απόλυτα λογικό, καθώς για να προκύψει ο κάθε χρόνος πρέπει να ζητήσει κανείς δεδομένα μέσω διαδικτύου και αυτά που θα λάβει να τα αποκωδικοποιήσει, γεγονός που είναι πιο χρονοβόρο από το να χρησιμοποιήσει απλά έναν έτοιμο τύπο. Με την χρήση διαδικτύου, η διαδικασία υπολογισμού 267 χρόνων, όσο δηλαδή και τα σημεία μπορούσε να διαρκέσει μέχρι και 1 λεπτό σε αντίθεση με την μέθοδο της Manhattan απόστασης που λίγα μόνο δευτερόλεπτα αρκούσαν. Επιπλέον, όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, τα δεδομένα που παρέχονται καθημερινώς από το διαδίκτυο για τις αποστάσεις είναι συγκεκριμένα (2.500 αποστάσεις). Τα πλεονεκτήματα και τα μειονεκτήματα των δύο μεθόδων φαίνονται ξεκάθαρα στον Πίνακας 3.

	Πλεονεκτήματα	Μειονεκτήματα
Google API Distance Matrix	• Υπολογισμός αποστάσεων με βάση το οδικό δίκτυο • Μεγάλη ακρίβεια δεδομένων • Η ταχύτητα βαδίσματος και κατά συνέπεια και ο χρόνος εξαρτάται από πολλούς παράγοντες (π.χ. κλίση εδάφους) • Δυνατότητα προβολής του αριθμού των δεδομένων που λήφθηκαν	• Περιορισμός καθημερινών δεδομένων • Απαίτηση αρκετού χρόνου για λήψη των δεδομένων • Αδυναμία συλλογής δεδομένων για μεγάλες αποστάσεις • Απαραίτητη η σύνδεση στο Διαδίκτυο • Σύνθετη Διαδικασία λήψης των δεδομένων
Manhattan Distance	• Απεριόριστα καθημερινά δεδομένα • Υπολογισμός σε μικρό χρονικό διάστημα • Υπολογισμός δεδομένων ανεξάρτητα της απόστασης • Υπολογισμός χωρίς τη χρήση Διαδικτύου • Απλή Διαδικασία υπολογισμού των δεδομένων • Δεν χρειάζεται έλεγχος	• Η ταχύτητα υπολογισμού των χρόνων είναι σταθερή και ίδια για όλες τις περιοχές • Υπολογισμός αποστάσεων ανεξάρτητων του οδικού δικτύου • Μεγάλη αβεβαιότητα στα δεδομένα

Πίνακας 3: Πλεονεκτήματα/Μειονεκτήματα υπολογισμού χρόνων

Ωστόσο, τα δεδομένα από το διαδίκτυο προσφέρουν μεγαλύτερη ακρίβεια, οπότε το καλύτερο θα ήταν να χρησιμοποιηθεί το διαδίκτυο μόνο για τα απαραίτητα στοιχεία, ώστε να ελαχιστοποιηθεί ο απαραίτητος χρόνος και τα απαιτούμενα δεδομένα και να αυξηθεί η ακρίβεια της προτεινόμενης διαδρομής. Για την καταχώρηση των χρόνων μετάβασης χρησιμοποιήθηκε η στήλη «dist_time» του πίνακα των σημείων. Αρχικά όλη η στήλη συμπληρώθηκε με την τιμή «-1», ώστε όταν υπολογιστούν οι χρόνοι να γίνει αναζήτηση των σημείων με βάση τους χρόνους που είναι μεγαλύτεροι του μηδενός (0). Στην συνέχεια, υπολογίζονται και συμπληρώνονται οι χρόνοι στη στήλη αυτή συγκεκριμένων σημείων. Τα κριτήρια με βάση τα οποία επιλέγονται τα σημεία είναι:

- Η τιμή που έχει το σημείο στην στήλη «price» να είναι ίση ή μικρότερη από το διαθέσιμο ποσό, που έχει ορίσει ο χρήστης στα κριτήρια. Έτσι αποκλείονται τα σημεία που ο χρήστης πρέπει να πληρώσει παραπάνω από το ποσό που όρισε ως μέγιστο.
- Η κατηγορία του κάθε σημείου. Ελέγχεται ποιες κατηγορίες σημείων επέλεξε ο χρήστης ότι θέλει να επισκεφθεί και απορρίπτονται οι υπόλοιπες.
- Η τιμή που έχουν τα σημεία στην στήλη «seen». Όταν η στήλη αυτή περιέχει την τιμή «1», ο χρήστης έχει επισκεφθεί το συγκεκριμένο μέρος, επομένως δεν λαμβάνεται υπόψη στην προτεινόμενη διαδρομή. Επομένως, λαμβάνονται υπόψη μόνο τα σημεία με τιμή σε αυτή τη στήλη ίση με «0».

Αφού ενημερωθούν λοιπόν οι τιμές συγκεκριμένων σημείων με τους απαιτούμενους χρόνους, στους χρόνους αυτούς προστίθεται και ο χρόνος περιήγησης του αντίστοιχου σημείου. Συνοπτικά, η στήλη «dist_time» αυτή τη στιγμή περιέχει μόνο για συγκεκριμένα σημεία, που καλύπτουν τα παραπάνω κριτήρια, τον συνολικό χρόνο που χρειάζεται ο χρήστης να μεταβεί από το σημείο έναρξης προς αυτά και να τα δει. Για όλα τα υπόλοιπα σημεία η τιμή είναι «-1».

Σημαντικό στοιχείο στην πορεία της διαδικασίας είναι η μετατροπή της μεταβλητής του ελεύθερου χρόνου που λήφθηκε από τη δραστηριότητα των κριτηρίων σε

μετρήσιμη ποσότητα, δηλαδή σε λεπτά της ώρας. Όπως αναφέρεται παραπάνω η μεταβλητή αυτή (freeTime_Crit) παίρνει τις τιμές 0, 1 και 2 για την 1, 2 και 3 ώρες αντίστοιχα. Επομένως μπορεί πολύ εύκολα να αντικατασταθεί από μια άλλη μεταβλητή (freeTime), που θα περιέχει μετρήσιμες μονάδες (σε λεπτά). Η αρχικοποίηση αυτής της μεταβλητής φαίνεται στον παρακάτω τύπο:

```
int freeTime = (freeTime_Crit + 1) * 60;
```

Έτσι, πλέον είναι όλα έτοιμα να ξεκινήσει η διαδικασία δημιουργίας της προτεινόμενης διαδρομής. Το αναλυτικό απόσπασμα του κώδικα αυτής της διαδικασίας βρίσκεται στο Παράρτημα. Αρχικά διατυπώνεται η συνθήκη με βάση τα κριτήρια που ορίστηκαν από τον χρήστη, για την δημιουργία ενός υποπίνακα μέσα από τον πίνακα σημείων. Αναλυτικότερα, δημιουργείται ένας υποπίνακας στον οποίο:

- Η τιμή κάθε σημείου της στήλης «dist_time» να είναι μεγαλύτερη από 0. Με αυτόν τον τρόπο αποκλείουμε τα σημεία, τα οποία δεν έχουν ενημερωμένες αποστάσεις (δηλαδή η τιμή είναι «-1»).
- Παράλληλα θα πρέπει αυτή η τιμή να είναι μικρότερη ή ίση από τον ελεύθερο χρόνο που διαθέτει ο χρήστης. Αυτό σημαίνει ότι εμφανίζονται μόνο τα σημεία, που ο χρόνος που χρειάζεται ο χρήστης για να μεταβεί στο σημείο και να το δει είναι μικρότερος από τον ελεύθερο χρόνο που έχει.
- Επιπλέον, από αυτά τα σημεία εμφανίζονται μόνο τα σημεία που η τιμή εισιτηρίου είναι μικρότερη από το ποσό που διαθέτει ο χρήστης και καθορίστηκε στα κριτήρια.
- Ταυτόχρονα, απορρίπτονται τα σημεία, όπου η τιμή στη στήλη «seen» είναι «1», δηλαδή τα σημεία που ο χρήστης έχει επισκεφθεί και δεν χρειάζεται να ξαναπάει.
- Τέλος, επιλέγονται τα σημεία που η κατηγορία τους είναι μέσα σε αυτές που επέλεξε ο χρήστης ότι θέλει να επισκεφθεί.

Με βάση λοιπόν αυτά τα κριτήρια δημιουργείται ένας υποπίνακας, στον οποίο περιέχονται όλα τα πιθανά σημεία, που μπορεί να αποτελέσουν το πρώτο σημείο της διαδρομής. Για να επιλεχθεί όμως το πρώτο, ταξινομούνται με βάση την επιλογή του χρήστη για το ενδιαφέρον του στα δημοφιλή. Αν ο χρήστης ενδιαφέρεται για τα δημοφιλή, τότε τοποθετούνται στον υποπίνακα αυτό πρώτα τα δημοφιλή μέρη, ταξινομημένα από το μέρος εκείνο που χρειάζεται λιγότερο χρόνο μέχρι εκείνο που χρειάζεται περισσότερο (με βάση τις τιμές της στήλης «dist_time») και στη συνέχεια τα μη δημοφιλή ταξινομημένα με τον ίδιο τρόπο. Αν δεν ενδιαφέρεται τότε η ταξινόμηση γίνεται απλά με την στήλη «dist_time» από εκείνο που χρειάζεται λιγότερο χρόνο σε εκείνο που χρειάζεται περισσότερο. Έτσι, το πρώτο στοιχείο του πίνακα αυτού θα αποτελέσει και την πρώτη επιλογή της προτεινόμενης διαδρομής.

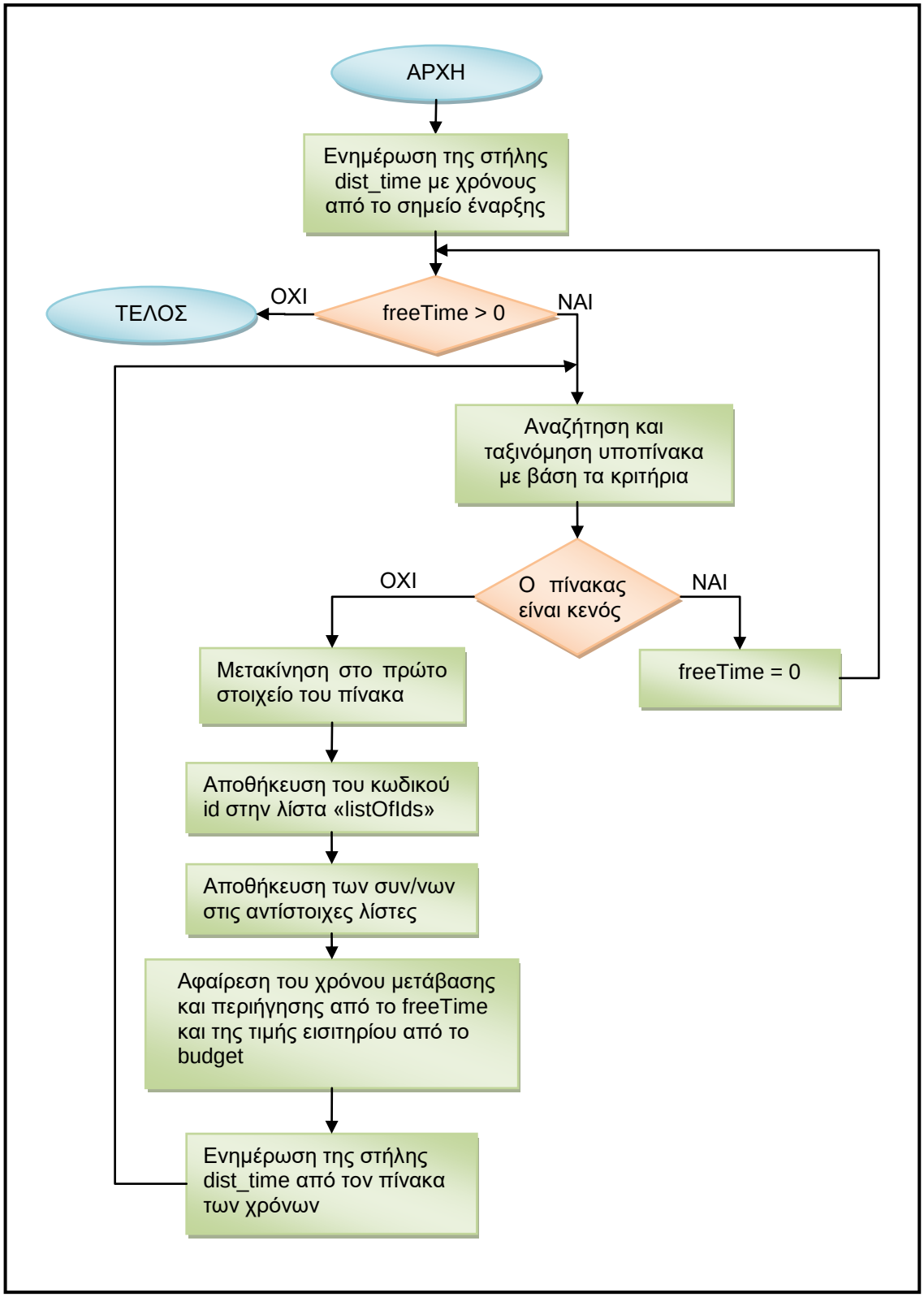
Για να καταχωρηθεί το κάθε σημείο που προτείνεται για την διαδρομή, δημιουργήθηκε μια λίστα με όνομα «listOfIds», στην οποία προστίθενται οι κωδικοί id του κάθε στοιχείου. Επομένως, από τον υποπίνακα αυτό προστίθεται ο κωδικός του πρώτου στοιχείου του πίνακα στη λίστα αυτή και παράλληλα προστίθενται και οι γεωγραφικές συντεταγμένες του στοιχείου αυτού στις αντίστοιχες λίστες που αναφέρθηκαν παραπάνω, την «long_points» και την «lat_points». Έπειτα, αφαιρείται από τον ελεύθερο χρόνο, ο χρόνος που χρειάστηκε ο χρήστης να μεταβεί και να δει το μέρος αυτό (δηλαδή η τιμή της στήλης «dist_time») και από το διαθέσιμο ποσό η τιμή εισιτηρίου του μέρους αυτού (δηλαδή η τιμή της στήλης «price»).

Τέλος, πριν την διαδικασία για την εύρεση του δεύτερου σημείου πρέπει να ενημερωθεί η στήλη «dist_time» με νέα δεδομένα. Αυτή τη στιγμή θεωρητικά ο χρήστης βρίσκεται στο πρώτο σημείο της διαδρομής. Του έχει απομείνει τόσος ελεύθερος χρόνος και τόσα χρήματα, όσα καθορίζουν οι αντίστοιχες μεταβλητές. Επομένως, η στήλη αυτή πρέπει να ενημερωθεί με τις χρόνους που χρειάζεται ο χρήστης για να μεταβεί από το πρώτο σημείο προς όλα τα άλλα και επιπλέον να τα

δει. Οι χρόνοι αυτοί βρίσκονται στον πίνακα των χρόνων. Με βάση λοιπόν τον κωδικό id του πρώτου σημείου, εντοπίζεται η εγγραφή που έχει τον ίδιο κωδικό. Τα δεδομένα αυτής της γραμμής αποτελούν τα νέα ζητούμενα δεδομένα. Ωστόσο, στις τιμές αυτές προστίθενται και οι χρόνοι περιήγησης κάθε σημείου.

Έτσι, έχει πλέον καταγραφεί το πρώτο σημείο και έχουν ενημερωθεί και οι τιμές για την έναρξη της αναζήτησης του δεύτερου. Η διαδικασία είναι η ίδια. Η μόνη διαφορά πλέον είναι ότι στην συνθήκη που διατυπώνεται για την αναζήτηση του υποπίνακα, θα προστεθεί και άλλο ένα κριτήριο. Πλέον, δεν πρέπει να υπάρχει στον πίνακα των προτεινόμενων σημείων το πρώτο σημείο. Με βάση και αυτή τη συνθήκη αναζητείται ο υποπίνακας, λαμβάνονται τα στοιχεία του πρώτου σημείου, προστίθενται στις αντίστοιχες λίστες, αφαιρούνται οι κατάλληλες τιμές από τις μεταβλητές του χρόνου και του ποσού και ενημερώνεται με τα καινούρια δεδομένα των χρόνων η στήλη «dist_time».

Η διαδικασία αυτή είναι προφανές ότι δεν γίνεται να επαναλαμβάνεται συνεχώς. Η μόνη συνθήκη που μπορεί να καθορίσει πότε θα σταματήσει η διαδικασία είναι ο ελεύθερος χρόνος. Αυτό σημαίνει ότι πριν ξεκινήσει η διαδικασία ορίζεται σαν συνθήκη επανάληψης ο ελεύθερος χρόνος να είναι μεγαλύτερος του μηδενός (0). Όσο υπάρχει δηλαδή ελεύθερος χρόνος να επαναλαμβάνεται η διαδικασία. Ωστόσο, υπάρχει η περίπτωση ο ελεύθερος χρόνος να μην αρκεί για να μεταβεί ο χρήστης σε κάποιο σημείο ή όλα τα κοντινά σημεία να έχουν τιμή εισιτηρίου μεγαλύτερη από το διαθέσιμο εκείνη τη στιγμή ποσό του χρήστη. Με βάση αυτές τις συνθήκες, ο υποπίνακας που θα προκύψει δεν θα περιέχει κανένα στοιχείο. Σε αυτήν την περίπτωση, ορίζεται η τιμή της μεταβλητής του ελεύθερου χρόνου ίση με μηδέν (0) και κατ' επέκταση η διαδικασία τερματίζεται αφού δεν ικανοποιείται η συνθήκη του ελεύθερου χρόνου. Όλη η παραπάνω διαδικασία εμφανίζεται στο επόμενο διάγραμμα ροής (Διάγραμμα 1).



Διάγραμμα 1: Διάγραμμα ροής αλγόριθμου προτεινόμενης διαδρομής

Μόλις τελειώσει λοιπόν η διαδικασία της προτεινόμενης διαδρομής, έχουν δημιουργηθεί τρεις λίστες. Οι δύο αφορούν τις γεωγραφικές συντεταγμένες και με βάση αυτές δημιουργείται η διεύθυνση που θα καθορίσει την διαδρομή στους χάρτες της συσκευής του χρήστη. Η τρίτη λίστα περιέχει σε σειρά τους κωδικούς id των σημείων της διαδρομής. Ωστόσο μόνο με αυτές τις πληροφορίες δημιουργείται μια συνθήκη, έτσι ώστε να εμφανίζονται στην λίστα της διαδρομής τα σημεία, που ο

κωδικός τους id να είναι ένας από τους καταγεγραμμένους. Αυτό όμως δεν προϋποθέτει ότι θα εμφανιστούν με τη σωστή σειρά.

Για τον λόγο αυτό και με δεδομένο ότι πλέον η στήλη «dist_time» δεν χρειάζεται, αποφασίστηκε να καταγραφεί η σειρά σε αυτή τη στήλη. Πιο συγκεκριμένα, διαβάζεται η λίστα με τους κωδικούς των προτεινόμενων σημείων, εντοπίζονται τα σημεία αυτά στον πίνακα των σημείων και ταυτόχρονα ενημερώνεται η τιμή της στήλης «dist_time» με αύξοντες αριθμούς για κάθε σημείο. Έτσι, στην λίστα του χρήστη για την προτεινόμενη διαδρομή φαίνονται τα σημεία εκείνα που:

- Ο κωδικός id τους είναι ένας από αυτούς που υπάρχουν στην λίστα «listOfIds».
- Στον υποπίνακα που προέκυψε γίνεται ταξινόμηση των σημείων με βάση τη στήλη «dist_time».

Έτσι, προκύπτει και η τελική λίστα με τα προτεινόμενα σημεία προς επίσκεψη για τον χρήστη και παρουσιάζεται στην οθόνη του. Ο κώδικας της παραπάνω διαδικασίας για την προβολή των σημείων στη λίστα φαίνεται στο Απόσπασμα 11.

```
StringBuilder selection = new StringBuilder(SightsEntry._ID + " =?");
for (int i = 0; i < listOfIds.size(); i++) {
    ContentValues timeValues = new ContentValues();
    timeValues.put(SightsEntry.COLUMN_DIST_TIME, i);
    int sightId = Integer.parseInt(listOfIds.get(i));
    Uri sightUri =
ContentUris.withAppendedId(SightsEntry.CONTENT_URI_SIGHTS, sightId);

    getContentResolver().update(sightUri, timeValues, null, null);
    selection.append(" OR " + SightsEntry._ID + " =?");
}
String[] selectionArgs = listOfIds.toArray(new
String[listOfIds.size()]);
String sortBy = SightsEntry.COLUMN_DIST_TIME + " ASC";

return new CursorLoader(ResultsActivity.this,
SightsEntry.CONTENT_URI_SIGHTS, null, selection.toString(),
selectionArgs, sortBy);
```

Απόσπασμα 11: Κατανομή των προτεινόμενων στοιχείων στη λίστα

Επιπλέον προσθήκες

Με βάση τις προσθήκες που έχουν γίνει μέχρι τώρα η εφαρμογή έχει πλέον τις βασικές λειτουργίες που είχε σχεδιαστεί να έχει. Ο χρήστης μπορεί να δει τα αξιοθέατα της Αθήνας είτε σε λίστα είτε σε χάρτη, να δει επιπλέον πληροφορίες για αυτά και με βάση κάποια κριτήρια να προκύψει η διαδρομή για την επίσκεψή του σε κάποια από αυτά. Ωστόσο, μετά από την χρήση της εφαρμογής από άλλα άτομα για την εύρεση τυχόν προσθηκών που θα βελτίωναν την εφαρμογή προέκυψαν σημαντικά σημεία, που δεν είχαν ληφθεί υπόψη. Η προσθήκη σημείων από τον χρήστη, η αποστολή των δεδομένων στον προγραμματιστή, η αναζήτηση μέσα στην λίστα για ένα συγκεκριμένο σημείο, το φιλτράρισμα της λίστας των σημείων και του χάρτη με συγκεκριμένα κριτήρια, η δημιουργία ενός πιο ελκυστικού λογότυπου και η χρήση πιο ελκυστικών χρωμάτων και γραμματοσειρών στην εφαρμογή, η προσθήκη επιπλέον γλωσσών και η προσθήκη οδηγιών για την χρήση της εφαρμογής ήταν μερικές επιπλέον προσθήκες που θα κάνουν την εμπειρία του χρήστη ακόμα καλύτερη.

Αρχικά πρώτη προσθήκη που έγινε στην εφαρμογή ήταν η δυνατότητα του χρήστη να προσθέτει ένα δικό του σημείο. Είναι προφανές ότι επειδή η συλλογή των σημείων έγινε χειροκίνητα, σίγουρα θα υπάρχουν αξιοθέατα στην Αθήνα που δεν έχουν προστεθεί στη λίστα και ο χρήστης γνωρίζει και θα ήθελε να επισκεφθεί. Για τον λόγο αυτό, δόθηκε στον χρήστη η δυνατότητα μέσα από το μενού της αρχικής

λίστας να μεταβεί σε μια νέα δραστηριότητα, που ονομάστηκε «NewSightActivity.java» και μέσα από αυτή να προσθέσει δικά του σημεία. Τα σημεία αυτά προσθέτονται στον πίνακα των σημείων της βάσης δεδομένων και επομένως εμφανίζονται και στην αρχική λίστα και στον χάρτη.

Πιο συγκεκριμένα, όταν ο χρήστης επιλέξει να προσθέσει ένα νέο σημείο ανοίγει η αντίστοιχη δραστηριότητα. Σε αυτή τη δραστηριότητα ο χρήστης έχει τη δυνατότητα να καθορίσει συγκεκριμένες πληροφορίες για το σημείο. Αρχικά πρέπει να καθορίσει την κατηγορία του σημείου μέσα από μια αναπτυσσόμενη λίστα με τις επιλογές των κατηγοριών. Σαν προεπιλογή ορίζεται το μουσείο. Στην συνέχεια έχει τη δυνατότητα να ορίσει το όνομα του σημείου, τη διεύθυνση, την τιμή του εισιτηρίου (όταν πρόκειται για σημεία που έχουν εισιτήριο εισόδου), το τηλέφωνο επικοινωνίας (αν υπάρχει) καθώς και τον χρόνο περιήγησης. Επιπλέον μετά τα στοιχεία αυτά παρέχεται και ένα κουμπί με την λέξη «ΧΑΡΤΗΣ», το οποίο ανακατευθύνει τον χρήστη στον χάρτη εύρεσης μέρους, που χρησιμοποιήθηκε και στη δραστηριότητα των κριτηρίων. Με την ίδια διαδικασία επιλέγει και πάλι ο χρήστης το σημείο στο χάρτη και αποθηκεύονται οι συντεταγμένες σε δύο μεταβλητές. Αμέσως εμφανίζεται ένα κείμενο δίπλα στο κουμπί, που υποδηλώνει ότι επιλέχθηκε τοποθεσία για το σημείο. Σε περίπτωση που παρέχονται παραπάνω πληροφορίες για αυτό το σημείο, όπως το τηλέφωνο, η διεύθυνση ή το όνομα, προσθέτονται οι αντίστοιχες πληροφορίες στα αντίστοιχα κενά. Επομένως ο χρήστης πρέπει να συμπληρώσει τα υπόλοιπα ή να αλλάξει ένα από αυτά αν δεν συμφωνεί.

Μόλις ολοκληρωθούν οι πληροφορίες ο χρήστης επιλέγει το κουμπί αποθήκευση από το μενού της δραστηριότητας. Αμέσως, λαμβάνονται οι τιμές που σημειώθηκαν, αποθηκεύονται σε μεταβλητές και προσθέτεται το νέο σημείο με τις αντίστοιχες πληροφορίες. Ωστόσο, αρκετές από τις πληροφορίες που απαιτούνται στον πίνακα των σημείων δεν έχουν καθοριστεί. Αναλυτικότερα, όσον αφορά τον κωδικό id του σημείου αυτός προσθέτεται αυτόματα με βάση το τελευταίο σημείο που προστέθηκε στον πίνακα. Το σημείο δεν θεωρείται δημοφιλές, η βαθμολογία του σε αστέρια είναι 0, όπως και 0 είναι η τιμή στην στήλη «dist_time», ενώ η στήλη της ιστοσελίδας και της εικόνας μένουν κενές με αποτέλεσμα να μην υπάρχει σύνδεσμος για περισσότερες πληροφορίες και να εμφανίζεται η προεπιλεγμένη εικόνα για το σημείο. Όσον αφορά την στήλη «seen», που καθορίζει αν ο χρήστης έχει επισκεφθεί το σημείο επιλέγεται να μην το έχει επισκεφθεί ενώ τέλος στην στήλη «can_delete» επιλέγεται θετική τιμή (η τιμή 1 καθορίζει ότι το σημείο μπορεί να διαγραφεί).

Στο σημείο αυτό πρέπει να σημειωθεί ότι δεν είναι απαραίτητη η καταγραφή όλων των πληροφοριών από τον χρήστη. Οι μόνες πληροφορίες που είναι απαραίτητες είναι η κατηγορία, το όνομα και οι συντεταγμένες. Όσον αφορά την κατηγορία, ο χρήστης δεν έχει την δυνατότητα να μην επιλέξει κατηγορία, καθώς σαν προεπιλεγμένη εμφανίζεται το «μουσείο». Αν ο χρήστης πάει να πατήσει αποθήκευση χωρίς να έχει καθορίσει καμία από τις άλλες δύο, εμφανίζεται μήνυμα στην οθόνη για να επιλέξει το όνομα του σημείου. Αν επιλέξει το όνομα και δεν έχει επιλέξει σημείο στον χάρτη, δηλαδή δεν έχει προσδιορίσει τις συντεταγμένες του, εμφανίζεται και πάλι αντίστοιχο μήνυμα για να προσδιορίσει την τοποθεσία του σημείου. Αν τα προσδιορίσει και τα δύο, αλλά δεν έχει δώσει τιμές στις υπόλοιπες πληροφορίες, έχουν καθοριστεί προεπιλεγμένες τιμές για κάθε πληροφορία και προσθέτονται αυτές για το σημείο αυτό.

Μετά την προσθήκη του νέου σημείου στον αντίστοιχο πίνακα, ο αριθμός των σημείων αυξήθηκε κατά ένα. Αυτό σημαίνει ότι αυτό το σημείο μπορεί να είναι μια πιθανή επιλογή στην λίστα των αποτελεσμάτων. Επομένως θα πρέπει να υπάρχουν τα κατάλληλα δεδομένα και στον πίνακα των χρόνων, που θα καθορίζουν τον χρόνο που χρειάζεται ο χρήστης να μεταβεί από το κάθε σημείο προς αυτό και το αντίστροφο. Αυτό σημαίνει ότι πρέπει να δημιουργηθεί μια νέα στήλη και μια νέα γραμμή στον πίνακα των χρόνων. Εκεί πρέπει να τοποθετηθούν τα δεδομένα του χρόνου μετάβασης από όλα τα σημεία προς αυτό και το αντίστροφο. Για τον υπολογισμό των χρόνων και για λόγους ταχύτητας, χρησιμοποιήθηκε ο τύπος της Manhattan απόστασης. Το απόσπασμα του κώδικα για την δημιουργία της νέας στήλης και γραμμής φαίνεται στο Απόσπασμα 12. Τέλος, μόλις προστεθεί το σημείο

στον πίνακα των σημείων και προστεθούν και τα δεδομένα των χρόνων στον αντίστοιχο πίνακα εμφανίζεται ένα μήνυμα στην οθόνη που ενημερώνει τον χρήστη για την επιτυχημένη ή αποτυχημένη προσθήκη των δεδομένων.

```
//Add new column to times table
SightsDbHelper dbHelper = new SightsDbHelper(NewSightActivity.this);
SQLiteDatabase db = dbHelper.getWritableDatabase();
String newCol = "T" + newId;
db.execSQL("ALTER TABLE " + TimesEntry.TABLE_NAME + " ADD COLUMN " +
newCol + " TEXT;");
db.close();

//Add times data to times table
ContentValues timeValues = new ContentValues();
String[] projection = {SightsEntry._ID, SightsEntry.COLUMN_LATITUDE,
SightsEntry.COLUMN_LONGITUDE};
Cursor sightsCursor =
getContentResolver().query(SightsEntry.CONTENT_URI_SIGHTS,
projection, null, null, null);

if (sightsCursor != null) {
    sightsCursor.moveToFirst();
}
do {
    int id = sightsCursor.getInt(0);
    double sightLat = sightsCursor.getDouble(1);
    double sightLong = sightsCursor.getDouble(2);
    int time = new
SightsContract().calcManhattanDistance(newSightLat, newSightLong,
sightLat, sightLong);
    String colName = "T" + id;
    timeValues.put(colName, time);

//Add times data in last column
    Uri currentTimeUri =
ContentUris.withAppendedId(TimesEntry.CONTENT_URI_TIMES, id);
    ContentValues columnValues = new ContentValues();
    columnValues.put(newCol, time);
    getContentResolver().update(currentTimeUri, columnValues, null,
null);

} while (sightsCursor.moveToNext());

timeValues.put("T" + newId, 0);
Uri timesUri =
getContentResolver().insert(TimesEntry.CONTENT_URI_TIMES,
timeValues);
sightsCursor.close();

if (newUri == null || timesUri == null) {
    Toast.makeText(this, getString(R.string.insert_sight_failed),
Toast.LENGTH_SHORT).show();
} else {
    Toast.makeText(this, getString(R.string.insert_sight_successful),
Toast.LENGTH_SHORT).show();
}
```

Απόσπασμα 12: Εισαγωγή των δεδομένων του νέου σημείου στον πίνακα των χρόνων

Όπως αναφέρθηκε και παραπάνω η στήλη «can_delete» είναι αυτή που διαφοροποιεί τα αρχικά σημεία με τα σημεία του χρήστη και είναι αυτή που επιτρέπει ή όχι την διαγραφή του σημείου από την βάση δεδομένων. Αναλυτικότερα, όταν η τιμή σε αυτή τη στήλη είναι «1», το σημείο προστέθηκε από τον χρήστη και κατ' επέκταση μπορεί να διαγραφεί. Σε όλες τις υπόλοιπες περιπτώσεις η τιμή είναι «0»

και ο χρήστης δεν μπορεί να το διαγράψει, καθώς έχει προστεθεί από τον προγραμματιστή. Η διαγραφή του σημείου γίνεται από τη δραστηριότητα των πληροφοριών του. Στο μενού αυτής της δραστηριότητας εμφανίζεται ένα εικονίδιο με ένα κάδο που υποδηλώνει την διαγραφή. Αυτό το εικονίδιο εμφανίζεται μόνο αν η τιμή στην στήλη είναι «1», ενώ στις υπόλοιπες περιπτώσεις το εικονίδιο είναι απενεργοποιημένο.

Μόλις ο χρήστης επιλέξει να διαγράψει το σημείο, εμφανίζεται ένα μήνυμα στην οθόνη για επιβεβαίωση της διαγραφής. Με την αποδοχή του μηνύματος διαγράφεται όλη η γραμμή με τις πληροφορίες του σημείου από τον πίνακα των σημείων και η γραμμή του σημείου με τους χρόνους από τον πίνακα των χρόνων. Επισημαίνεται ότι η επιπλέον στήλη με τα δεδομένα χρόνων που δημιουργήθηκε για το συγκεκριμένο σημείο δεν διαγράφεται καθώς δεν παρέχεται τέτοια λειτουργία για τους πίνακες που δημιουργήθηκαν με εντολές SQL στο Android Studio. Επιπλέον μια παραπάνω στήλη δεν επηρεάζει τα αποτελέσματα.

Ένα άλλο σημείο που πρέπει να σημειωθεί είναι ο τρόπος αντιστοίχησης κωδικού id για κάθε σημείο. Ο κωδικός id που χρησιμοποιείται κάθε φορά για το νέο σημείο, δεν είναι με βάση το τελευταίο σημείο που έχει προστεθεί εκείνη τη στιγμή στην λίστα, αλλά με βάση τα σημεία που έχουν προστεθεί γενικά. Για παράδειγμα, αν ο χρήστης προσθέσει ένα νέο σημείο στην αρχική βάση δεδομένων (που περιέχει 267 σημεία), το σημείο αυτό θα έχει κωδικό 268. Αν το διαγράψει τα σημεία θα είναι και πάλι 267. Ωστόσο, αν προσθέσει ένα νέο αυτό δεν θα έχει κωδικό 268, αλλά 269. Αυτό δημιουργεί κάποια κενά στον πίνακα, ωστόσο αυτό δεν επηρεάζει πουθενά καμία διαδικασία και δεν είναι ορατό στον χρήστη.

Τα δεδομένα αυτά βρίσκονται στη βάση δεδομένων που δημιουργείται τοπικά στην συσκευή του χρήστη. Είναι προφανές λοιπόν ότι ο προγραμματιστής δεν έχει πρόσβαση σε αυτά και κατ' επέκταση δεν μπορεί να βγάλει συμπεράσματα, όπως την βαθμολογία του κάθε μέρους ή τον χρόνο περιήγησης. Για τον λόγο αυτό στο μενού της αρχικής λίστας δόθηκε η δυνατότητα στον χρήστη να μπορεί να στείλει τον πίνακα των σημείων με mail στον προγραμματιστή σε μορφή αρχείου *.csv.

Πιο συγκεκριμένα, μόλις ο χρήστης επιλέξει αυτή την επιλογή, ανοίγει ένα παράθυρο που του ζητάει τα στοιχεία του (όνομα και mail). Αυτά τα στοιχεία θα αποτελέσουν μέρος του κειμένου που θα υπάρχει στο mail. Μόλις ο χρήστης συμπληρώσει τα στοιχεία του και πατήσει «OK», δημιουργούνται δύο αρχεία *.csv στη συσκευή του χρήστη με τους δύο πίνακες (σημείων και χρόνων). Αμέσως ανοίγει ένα παράθυρο που ζητάει από τον χρήστη να επιλέξει την εφαρμογή που θα χρησιμοποιήσει για να στείλει το mail (αν δεν έχει ορίσει προεπιλεγμένη εφαρμογή για αποστολή των mail). Όταν επιλέξει την εφαρμογή, συμπληρώνεται το mail του προγραμματιστή στο αντίστοιχο πεδίο του παραλήπτη, συμπληρώνεται το θέμα του mail, και εισάγεται το αρχείο με τα δεδομένα των σημείων σαν συνημμένο. Επιπλέον, αν η δημιουργία των αρχείων στο κινητό του χρήστη ήταν επιτυχής εμφανίζονται τα αντίστοιχα μηνύματα στο κείμενο του mail και από κάτω τα στοιχεία του χρήστη που συμπληρώθηκαν στην αρχή. Το μόνο που πρέπει να κάνει ο χρήστης είναι να πατήσει το κουμπί αποστολή.

Όπως αναφέρθηκε παραπάνω κατά την διαδικασία αποστολής των δεδομένων, δημιουργούνται δύο αρχεία στην συσκευή του χρήστη με τα αντίστοιχα δεδομένα των σημείων και των χρόνων. Για την δημιουργία αυτών των σημείων, ο χρήστης πρέπει να παρέχει την κατάλληλη άδεια στην εφαρμογή να δημιουργήσει τα αρχεία. Η άδεια αυτή επιλέχθηκε να εμφανίζεται μαζί με την άδεια για την τοποθεσία στην εκκίνηση της εφαρμογής. Σε περίπτωση που η άδεια δεν γίνει αποδεκτή, τα αρχεία δεν μπορούν να δημιουργηθούν και κατ' επέκταση δεν μπορούν να αποσταλούν στον προγραμματιστή. Σε αντίθετη περίπτωση η άδεια γίνεται αποδεκτή μια φορά στην αρχή και μετά ο χρήστης μπορεί όποτε θέλει να στείλει τα δεδομένα.

Μετά από δοκιμές της εφαρμογής από πολλά άτομα, εντοπίζοντας τις αδυναμίες της μέχρι τώρα έκδοσής της, εντοπίστηκε μια σημαντική απώλεια στην αρχική λίστα. Ο χρήστης για να αναζητήσει κάποιο σημείο της λίστας, έπρεπε να κατέβει και να τα περάσει όλα με αποτέλεσμα να είναι δύσκολη η αναζήτηση κάποιου ήδη γνωστού σημείου. Για τον λόγο αυτό, έγινε αναγκαία η προσθήκη ενός εργαλείου αναζήτησης

σημείων. Αυτό επιτεύχθηκε με την προσθήκη ενός εικονιδίου (επιλογή από τα έτοιμα που παρέχονται από την Android και συμβολίζει την αναζήτηση) στο μενού της αρχικής λίστας, το οποίο είναι πάντα ορατό και με το οποίο μπορεί να γίνει αναζήτηση στα ονόματα της λίστας. Η επιλογή της αναζήτησης παρέχεται με έτοιμες εντολές από την Android.

Πρακτικά, με την επιλογή αυτού του εικονιδίου, ανοίγει ένα πλαίσιο κειμένου στο οποίο ο χρήστης πληκτρολογεί αυτό που ψάχνει. Κάθε φορά που το κείμενο αλλάζει, αποθηκεύεται σε μια μεταβλητή. Η μεταβλητή αυτή αποτελεί το κριτήριο σύμφωνα με το οποίο γίνεται αναζήτηση στον πίνακα των σημείων. Επιλέγονται λοιπόν τα σημεία, που στο όνομά τους περιέχεται αυτή η μεταβλητή και δημιουργείται ένας υποπίνακας με την αντίστοιχη εντολή. Αυτόματα ανανεώνεται η λίστα με τον αντίστοιχο υποπίνακα που δημιουργήθηκε και ο χρήστης το καταλαβαίνει σαν αναζήτηση. Αν στο τέλος ή στην αρχή του κειμένου που πληκτρολογήθηκε, πληκτρολογηθεί και κενό, τότε αυτό απορρίπτεται και γίνεται αναζήτηση της λέξης χωρίς το κενό. Το απόσπασμα του κώδικα που φαίνεται η δημιουργία και η λειτουργία αυτού του εικονιδίου φαίνεται στον παρακάτω κώδικα.

```
//Δημιουργία του εικονιδίου της αναζήτησης στο μενού
MenuItem searchItem = menu.findItem(R.id.item_search);
SearchView searchView = (SearchView)
MenuItemCompat.getActionView(searchItem);

searchView.setOnQueryTextListener(new
SearchView.OnQueryTextListener() {
    @Override
    public boolean onQueryTextSubmit(String query) {
        return false;
    }

    @Override
    public boolean onQueryTextChange(String newText) {
//Καταχώρηση του κειμένου που πληκτρολογείται στη μεταβλητή
mCurFilter
        mCurFilter = newText.trim();
//Ανανέωση της λίστας
        getLoaderManager().restartLoader(SIGHTS_LOADER, null,
MainActivity.this);
        return true;
    }
});
searchItem.setActionView(searchView);
```

Απόσπασμα 13: Δημιουργία του εικονιδίου αναζήτησης

Ωστόσο, ήταν αρκετές οι φορές που κατά την αναζήτηση σημείων στη λίστα, δεν υπήρχε κάποιο συγκεκριμένο σημείο, αλλά απλά ο χρήστης ήθελε να δει την λίστα και να διαλέξει ποιο σημείο ήθελε να επισκεφθεί. Οι εικόνες τον βοηθούσαν από την μια αλλά μπορεί να ήθελε να διαλέξει ανάμεσα σε συγκεκριμένες κατηγορίες. Επιπλέον κάποιοι χρήστες ήθελαν να δουν τα σημεία αλφαβητικά. Όλα αυτά οδήγησαν στην προσθήκη φίλτρων ώστε να εμφανίζεται η λίστα όπως την θέλει ο χρήστης.

Έτσι, στο μενού της αρχικής λίστας προστέθηκε και η επιλογή φίλτρα. Με την επιλογή των φίλτρων ανοίγει ένα παράθυρο, δίνοντας στον χρήστη την επιλογή να τσεκάρει ή όχι ποιες κατηγορίες σημείων θα εμφανιστούν στην λίστα και ταυτόχρονα με ποιο τρόπο θα ταξινομηθούν τα σημεία. Οι επιλογές του χρήστη αποθηκεύονται σε μεταβλητές και μόλις ο χρήστης πατήσει «OK», η λίστα ανανεώνεται με τις επιλογές του. Αντίστοιχα μπορεί μετά να κάνει αναζήτηση μόνο σε αυτά τα σημεία.

Πιο συγκεκριμένα, δημιουργούνται δύο μεταβλητές στην δραστηριότητα της λίστας, οι οποίες καθορίζουν τα κριτήρια σύμφωνα με τα οποία γίνεται αναζήτηση στον πίνακα των σημείων. Στις μεταβλητές αυτές μπαίνουν και τα φίλτρα και η αναζήτηση. Η αρχικοποίηση των μεταβλητών αυτών φαίνεται στο Απόσπασμα 14.

```
String filterSelection = "(" + SightsEntry.COLUMN_CATEGORY + "=? OR "
+ SightsEntry.COLUMN_CATEGORY + "=? OR " +
SightsEntry.COLUMN_CATEGORY + "=? OR " + SightsEntry.COLUMN_CATEGORY
+ "=?) AND " + SightsEntry.COLUMN_NAME + " LIKE ?";

String[] filterSelectionArgsString =
{String.valueOf(SightsEntry.CATEGORY_MUSEUM),
String.valueOf(SightsEntry.CATEGORY_ATTRACTION),
String.valueOf(SightsEntry.CATEGORY_CHURCH),
String.valueOf(SightsEntry.CATEGORY_OTHER), "%"};
```

Απόσπασμα 14: Αρχικοποίηση μεταβλητών για ανανέωση της λίστας

Όπως φαίνεται από το παραπάνω απόσπασμα, οι δύο μεταβλητές είναι η «***filterSelection***» και η «***filterSelectionArgsString***». Σύμφωνα με την πρώτη καθορίζεται η συνθήκη με την οποία γίνεται αναζήτηση στον πίνακα των σημείων, ενώ στην δεύτερη περιέχονται οι τιμές με βάση τις οποίες ολοκληρώνονται οι συνθήκες της πρώτης. Αυτό σημαίνει ότι ο χαρακτήρας «?» της πρώτης μεταβλητής αντικαθίσταται με τις τιμές της δεύτερης με την σειρά. Αναλυτικότερα σύμφωνα με τον παραπάνω κώδικα, που αποτελεί τις αρχικές συνθήκες εμφάνισης της λίστας γίνεται αναζήτηση σε σημεία, όπου η κατηγορία τους είναι ίση ή με «μουσείο» ή με «αξιοθέατο» ή με «εκκλησία» ή με «άλλο» και ταυτόχρονα το όνομά τους να περιέχει οποιονδήποτε χαρακτήρα (με την εντολή «***LIKE %***»). Το αποτέλεσμα αυτής της αναζήτησης είναι όλα τα σημεία του πίνακα.

Οποιαδήποτε αλλαγή γίνει είτε στα φίλτρα, είτε στην αναζήτηση, ενημερώνονται αυτόματα και οι δύο μεταβλητές και γίνεται ανανέωση της λίστας με βάση αυτές τις μεταβλητές. Όσον αφορά τώρα την ταξινόμηση, ο χρήστης επιλέγει αν θα κάνει ταξινόμηση με βάση το όνομα των σημείων, οπότε ο υποπίνακας ταξινομεί τα δεδομένα από το Α ως το Ω, ή την ημερομηνία δημιουργίας των σημείων, που ουσιαστικά αντιπροσωπεύεται με τον κωδικό id των σημείων από το 1 ως το τελευταίο.

Τέλος, σε περίπτωση που ο χρήστης δεν επιλέξει καμία κατηγορία ή κατά την αναζήτηση σημείου δεν υπάρχει κανένα σημείο με αυτό το όνομα, η λίστα θα είναι κενή. Στην περίπτωση αυτή προβλέφθηκε να εμφανίζεται ένα αντίστοιχο μήνυμα στην οθόνη ενημερώνοντας τον χρήστη ότι κάτι πρέπει να αλλάξει. Επιπλέον, προστέθηκε ένα ακόμα κουμπί στο παράθυρο των φίλτρων, το οποίο δίνει την δυνατότητα στον χρήστη να κάνει επαναφορά των αρχικών επιλογών στη λίστα και να αφαιρέσει ότι φίλτρο και αναζήτηση έχει προσθέσει.

Όπως ήταν λογικό, τα φίλτρα αυτά ήταν αρκετά χρήσιμα, αλλά πιο χρήσιμα φάνηκαν στην περίπτωση του χάρτη. Η ίδια λειτουργία με μια μικρή διαφορά προστέθηκε και στο μενού του χάρτη. Κατά την μετάβαση από την δραστηριότητα της λίστας στον χάρτη, οι δύο παραπάνω μεταβλητές πέρασαν από την μια δραστηριότητα στην άλλη. Η μόνη διαφορά πλέον είναι ότι η ταξινόμηση δεν παίζει κανένα ρόλο. Το μόνο που χρειάστηκε σε αυτή τη δραστηριότητα ήταν οι κατηγορίες. Έτσι στο παράθυρο των φίλτρων αφαιρέθηκε η επιλογή της ταξινόμησης. Έτσι, ο χρήστης κάθε φορά που επιλέγει καινούρια φίλτρα, αφαιρούνται όλα τα σημεία από τον χάρτη και γίνεται ανανέωση και προσθήκη των καινούριων σημείων. Με τον τρόπο αυτό μπορεί ο χρήστης να δει οπτικά πως κατανέμεται η κάθε κατηγορία στον χάρτη και περίπου πόσα σημεία από κάθε κατηγορία υπάρχουν στην βάση δεδομένων.

Μια από τις τελευταίες προσθήκες που πρέπει να σημειωθούν και έγιναν κατά την διάρκεια όλης της διαδικασίας ήταν η επιλογή δύο γλωσσών. Σαν αρχική γλώσσα επιλέχθηκαν τα Αγγλικά. Ήταν η γλώσσα άλλωστε που χρησιμοποιήθηκε στον κώδικα και στις μεταβλητές. Αγγλικά χρησιμοποιήθηκαν επίσης και στην οθόνη και σε ότι φαίνεται στον χρήστη. Ωστόσο, το Android Studio δίνει την επιλογή όλες τις γλώσσες να τις καταχωρείς σαν μεταβλητές και μέσα από ένα πίνακα να καθορίζεις τις μεταβλητές αυτές και σε άλλες γλώσσες. Επομένως, όταν χρειάστηκε να δημιουργηθεί ένα κείμενο, ένα μήνυμα ή μια λέξη που θα φαίνεται στην οθόνη χρησιμοποιήθηκε μια μεταβλητή, η οποία μετά καθορίστηκε σε Αγγλικά και σε

Ελληνικά. Κάθε φορά που η εφαρμογή θα ξεκινάει από κάποια συσκευή ελέγχεται αν η προεπιλεγμένη γλώσσα της συσκευής είναι τα ελληνικά ή τα αγγλικά και ανάλογα εμφανίζεται η αντίστοιχη γλώσσα στην συσκευή. Αν η συσκευή είναι σε οποιαδήποτε άλλη γλώσσα, επιλέγονται τα Αγγλικά.

Τέλος, η τελευταία προσθήκη που έγινε στην εφαρμογή ήταν οι οδηγίες. Στο μενού της αρχικής λίστας προστέθηκε και η επιλογή «Οδηγίες». Αν ο χρήστης επιλέξει να δει τις οδηγίες ανοίγει μια νέα δραστηριότητα, με όνομα «InstructionsActivity.java». Στην δραστηριότητα αυτή το μόνο που εμφανίζεται είναι κείμενο με οδηγίες για την εφαρμογή και ένα κουμπί στο κάτω δεξιά μέρος της οθόνης με το κείμενο «ΕΠΟΜΕΝΟ». Πατώντας το κουμπί ο χρήστης, αλλάζει το κείμενο και ταυτόχρονα εμφανίζεται άλλο ένα κουμπί στο κάτω αριστερά μέρος της οθόνης με το κείμενο «ΠΡΟΗΓΟΥΜΕΝΟ». Αυτό γίνεται για 3 ακόμα φορές, ώσπου το κουμπί με το κείμενο «ΕΠΟΜΕΝΟ» γίνεται «ΤΕΛΟΣ» και πατώντας αυτό ο χρήστης επιστρέφει στην αρχική λίστα. Μέσα από αυτή τη διαδικασία ο χρήστης μπορεί να κατανοήσει τι μπορεί να του παρέχει η εφαρμογή και πως μπορεί να την χρησιμοποιήσει.

Η εφαρμογή είναι πλέον έτοιμη για χρήση. Τελευταία διαδικασία τελειοποίησης της εφαρμογής ήταν ο καθορισμός των χρωμάτων, που χρησιμοποιήθηκε το μπλε για το μενού, το μαύρο και το γκρι για τα γράμματα και το πορτοκαλί για τα κουμπιά και τις επιπλέον προσθήκες. Τα πρώτα ήταν προεπιλογή της Android. Το μόνο που άλλαξε είναι το πορτοκαλί, καθώς μετά από πολλές αλλαγές αποφασίστηκε ότι ένα έντονο χρώμα θα έκανε ορατά τα κουμπιά. Αρχικά είχε χρησιμοποιηθεί το γαλάζιο, αλλά φάνηκε ότι ειδικά το κουμπί στην αρχική οθόνη δεν ήταν αμέσως ορατό.

Τέλος, σχεδιάστηκε το λογότυπο της εφαρμογής σε διάφορες κλίμακες, όπως ορίζει η Android. Πιο συγκεκριμένα, λόγω της διαφορετικότητας της ανάλυσης των συσκευών που υπάρχουν η Android προτείνει όλα τα εικονίδια και το λογότυπο της εφαρμογής να δημιουργούνται σε 5 κλίμακες, όπως φαίνεται παρακάτω:

- Κλίμακα mdpi: 48 x 48 pixels
- Κλίμακα hdpi: 72 x 72 pixels
- Κλίμακα xhdpi: 96 x 96 pixels
- Κλίμακα xxhdpi: 144 x 144 pixels
- Κλίμακα xxxhdpi: 192 x 192 pixels

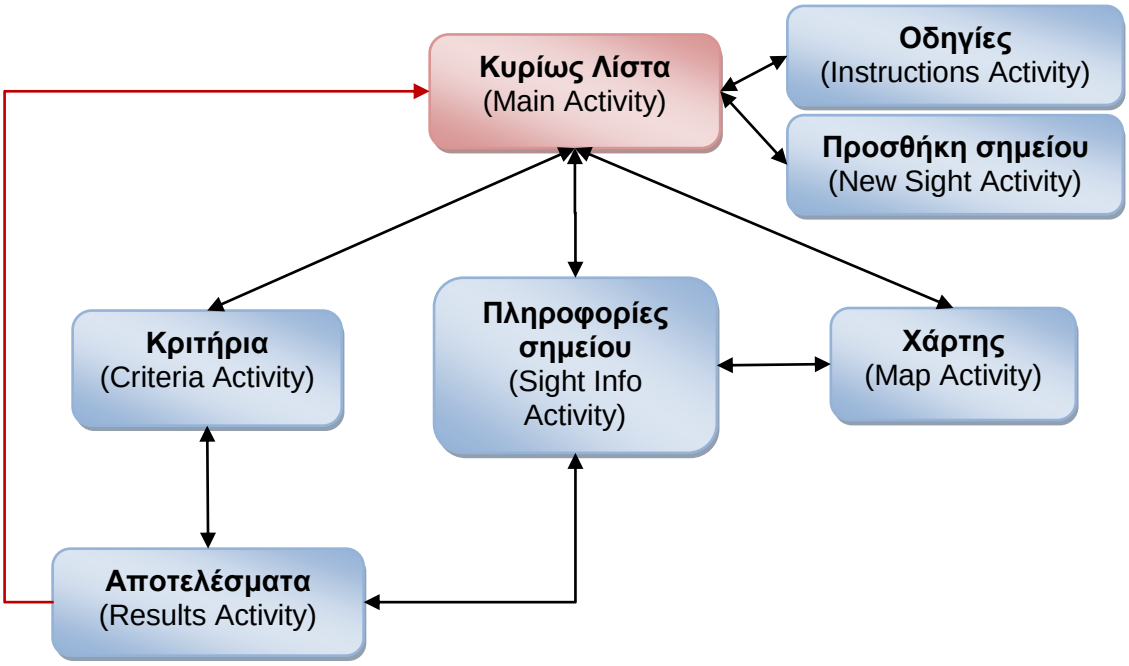
Ανάλογα με την ανάλυση της οθόνης της κάθε συσκευής, χρησιμοποιείται η αντίστοιχη κλίμακα και ευτυχώς δεν πρέπει να γίνει κάτι προγραμματιστικά για να καθοριστεί αυτό. Το λογότυπο λοιπόν της εφαρμογής, μετά από πολλές σχεδιαστικές απόπειρες και με χρήση πολλών λογισμικών κατέληξε στην μορφή που φαίνεται στην Εικόνα 4.

Εικόνα 4: Λογότυπο City Stroller

Για την παραγωγή του λογότυπου αυτού χρησιμοποιήθηκε το ελεύθερο πρόγραμμα «Jeta Logo Designer» και χρησιμοποιήθηκαν απλές εντολές παραγωγής

εικόνων. Χρησιμοποιήθηκαν κατά κύριο λόγο χρώματα που υπάρχουν και μέσα στην εφαρμογή και έχουν μεταξύ τους μια αντίθεση. Έτσι η εφαρμογή πήρε πλέον την τελική της μορφή.

Συνοπτικά, στο κεφάλαιο αυτό παρουσιάστηκε η εσωτερική λειτουργία της εφαρμογής και έγινε κατανοητό τι ακριβώς συμβαίνει πίσω από την οθόνη. Ωστόσο, επειδή αναφέρθηκαν αρκετές δραστηριότητες και λειτουργίες της εφαρμογής μπορεί να υπάρχει μια σύγχυση για τις λειτουργίες της εφαρμογής. Στο Διάγραμμα 2 παρακάτω εμφανίζεται το διάγραμμα ροής της εφαρμογής, ώστε να γίνει πιο κατανοητή η μετάβαση από κάθε δραστηριότητα προς όλες τις άλλες.



Διάγραμμα 2: Διάγραμμα ροής Δραστηριοτήτων City Stroller

Όπως παρατηρείται από την κυρίως λίστα μπορεί ο χρήστης να μεταβεί σε όλες τις άλλες δραστηριότητες και να επιστρέψει πίσω εκτός από τα αποτελέσματα. Για να μεταβεί στα αποτελέσματα, στην προτεινόμενη δηλαδή διαδρομή, πρέπει να μεταβεί πρώτα στα κριτήρια και να ορίσει τα κατάλληλα κριτήρια. Από τα αποτελέσματα μπορεί να μεταβεί στις πληροφορίες οποιουδήποτε σημείου και από εκεί και στον χάρτη έχοντας παράλληλα την δυνατότητα να επιστρέψει πάλι πίσω. Όπως μπορεί κανείς να παρατηρήσει από όλες τις δραστηριότητες μπορεί να επιστρέψει πάλι πίσω. Η μόνη διαδρομή που δεν είναι αμφίρροπη είναι από τα αποτελέσματα στην κυρίως λίστα. Αυτό σημαίνει ότι αν από τα αποτελέσματα ο χρήστης επιλέξει να γυρίσει στην αρχική λίστα δεν έχει πάλι τη δυνατότητα να δει τα αποτελέσματα ξανά. Το παραπάνω σχήμα θα γίνει πιο κατανοητό στην επόμενη ενότητα που θα γίνει παρουσίαση της εφαρμογής με εικόνες.

Παρουσίαση της Εφαρμογής

Στην ενότητα αυτή θα παρουσιαστεί η λειτουργία της εφαρμογής και θα γίνουν πιο κατανοητές οι πληροφορίες που απλά περιγράφηκαν με λόγια στην προηγούμενη ενότητα. Με την πρώτη εκκίνηση της εφαρμογής εμφανίζονται ειδοποιήσεις για αποδοχή αδειών από τον χρήστη. Αυτές οι άδειες έχουν να κάνουν με την λήψη της τοποθεσίας του χρήστη (όποτε αυτό είναι απαραίτητο κυρίως για την περίπτωση του χάρτη) και την δημιουργία αρχείων στο κινητό του χρήστη (όταν χρειαστεί να αποσταλούν τα δεδομένα στον προγραμματιστή). Στη συνέχεια εμφανίζεται η λίστα

με τα διαθέσιμα σημεία ενδιαφέροντος της Αθήνας, όπως φαίνεται στην αρχική λίστα της Εικόνα 6 του Παράρτημα.

Ο χρήστης έχει τη δυνατότητα να επιλέξει ένα σημείο από τη λίστα πατώντας πάνω σε αυτό. Αμέσως ανοίγει το σημείο με όλες τις πληροφορίες που έχει, όπως φαίνεται στο κέντρο στην Εικόνα 6 του Παράρτημα. Μπορεί να πατήσει πάνω στην εικόνα και να του εμφανιστεί σε πραγματικό μέγεθος για να έχει μια καλύτερη άποψη για το συγκεκριμένο μέρος. Με την επιλογή του κουτιού «Πήγα» ανοίγει ένα παράθυρο, όπως φαίνεται δεξιά στην Εικόνα 6 του Παράρτημα, στο οποίο ο χρήστης βαθμολογεί το σημείο ενδιαφέροντος. Επιλέγει την βαθμολογία του με βάση την κρίση του και επιπλέον μπορεί να γράψει πόση ώρα παρέμεινε εκεί. Μόλις συμπληρώσει αυτά τα δύο στοιχεία ενημερώνεται η βαθμολογία και ο χρόνος περιήγησης αντίστοιχα στις πληροφορίες του σημείου. Επιπλέον χαμηλά εμφανίζεται ένα κουμπί «Σύνδεσμος» (όταν υπάρχει η αντίστοιχη πληροφορία για το σημείο), το οποίο αυτόματα ανοίγει στον προεπιλεγμένο browser τον σύνδεσμο με τις πληροφορίες του αντίστοιχου σημείου. Τέλος, πάνω δεξιά εμφανίζεται το εικονίδιο του χάρτη με το οποίο ο χρήστης μπορεί να δει στον χάρτη το συγκεκριμένο σημείο.

Όπως φαίνεται αριστερά στην Εικόνα 6 του Παράρτημα στο μενού πάνω από τη λίστα υπάρχουν 3 κουμπιά:

- Το πρώτο έχει να κάνει με την αναζήτηση συγκεκριμένων σημείων ενδιαφέροντος μέσα από τη λίστα με βάση το όνομά τους.
- Το δεύτερο ανοίγει ένα χάρτη με όλα τα διαθέσιμα σημεία. Ο χρήστης έχει τη δυνατότητα να φιλτράρει τα σημεία του χάρτη ανά κατηγορία και να επιλέξει να εμφανίζονται μόνο κάποιες συγκεκριμένες κατηγορίες. Μέσα από αυτό το χάρτη επιπλέον μπορεί να βρει την τοποθεσία του (με το αντίστοιχο κουμπί ενεργοποιώντας το GPS της συσκευής) ή να επιλέξει ένα σημείο και να βρει οδηγίες προς αυτό (μέσω του Google Maps). Σημειώνεται ότι ο χάρτης υποστηρίζεται σε δύο μορφές: οδικό δίκτυο και δορυφόρος (αριστερά και κέντρο αντίστοιχα στην Εικόνα 7 του Παράρτημα), όπου εμφανίζονται διαφορετικές πληροφορίες κάθε φορά.
- Το τρίτο ανοίγει μια λίστα με διάφορες επιλογές:
    - Η πρώτη επιλογή είναι τα «Φίλτρα». Με την επιλογή αυτή ο χρήστης μπορεί να επιλέξει να φιλτράρει τη λίστα και να του εμφανίζονται συγκεκριμένα σημεία. Πιο συγκεκριμένα, του εμφανίζεται ένα παράθυρο στο οποίο ο χρήστης επιλέγει ποια κατηγορία σημείων θέλει να του εμφανιστεί και πως θέλει να ταξινομηθεί η λίστα (δεξιά στην Εικόνα 7 του Παράρτημα). Οι επιλογές των κατηγοριών είναι 4, ενώ η ταξινόμηση μπορεί να γίνει είτε μέσω της ημερομηνίας δημιουργίας (δηλαδή με βάση τον κωδικό id του κάθε σημείου) είτε με βάση το όνομα των σημείων αλφαβητικά.
    - Η δεύτερη επιλογή είναι η «Προσθήκη νέου σημείου». Από την λειτουργία αυτή μπορεί ο χρήστης να προσθέσει ένα σημείο ενδιαφέροντος που δεν υπάρχει στη λίστα (αριστερά στην Εικόνα 8 του Παράρτημα). Επιλέγει την κατηγορία του σημείου και πληκτρολογεί τις απαραίτητες πληροφορίες (όνομα, διεύθυνση, τιμή εισιτηρίου, τηλέφωνο και χρόνοι περιήγησης). Επιπλέον πρέπει να το βρει και στον χάρτη από το κουμπί «Χάρτης». Ο χρήστης μπορεί να κάνει αναζήτηση ενός σημείου, μιας διεύθυνσης ή να επιλέξει την τρέχουσα θέση του για την επιλογή της τοποθεσίας του σημείου. Επιπλέον του εμφανίζεται και μια λίστα με τα κοντινά μέρη με βάση δεδομένα από την Google σε περίπτωση που το σημείο που θέλει να καταχωρίσει υπάρχει ήδη σαν αξιοθέατο και δεν έχει περαστεί από τον προγραμματιστή (κέντρο στην Εικόνα 9 του Παράρτημα). Μόλις ο χρήστης συμπληρώσει όλα τα στοιχεία και πατήσει αποθήκευση το σημείο εμφανίζεται πλέον στη λίστα.

- Η τρίτη επιλογή είναι η «Αποστολή δεδομένων» και έχει να κάνει με την αποστολή των στοιχείων όλων των σημείων στον προγραμματιστή. Ο χρήστης πληκτρολογεί το όνομα και το mail του σε ένα παράθυρο που ανοίγει (κέντρο στην Εικόνα 8 του Παράρτημα) και αυτόματα κατευθύνεται στην προεπιλεγμένη εφαρμογή αποστολής mail με έτοιμα όλα τα στοιχεία προς αποστολή (mail παραλήπτη, συνημμένο, κείμενο με τα στοιχεία του χρήστη). Το μόνο που πρέπει να κάνει ο χρήστης είναι να επιλέξει να στείλει το mail. Έτσι ο προγραμματιστής θα μπορεί να ξέρει τι σημεία έχει προσθέσει ο χρήστης, σε ποια έχει πάει καθώς και τις αξιολογήσεις που έχει βάλει στο καθένα.
- Τέλος, η τέταρτη επιλογή είναι οι «Οδηγίες». Σε αυτή την επιλογή ο χρήστης έχει την δυνατότητα να γνωρίσει καλύτερα και να κατανοήσει τυχόν δυσνόητα σημεία της εφαρμογής, αφού του παρέχονται ξεκάθαρες οδηγίες για όλες τις λειτουργίες της (δεξιά στην Εικόνα 8 του Παράρτημα).

Στην αρχική λίστα μπορεί κανείς κάτω δεξιά να παρατηρήσει ένα πορτοκαλί κουμπί με έναν άνθρωπο που περπατάει. Αυτό είναι το κουμπί για την εύρεση της ιδανικής διαδρομής, η κύρια λειτουργία δηλαδή της εφαρμογής. Με το πάτημα του κουμπιού εμφανίζεται η σελίδα με τα κριτήρια για να καθορίσει ο χρήστης ανάλογα με την δική του θέληση τι ακριβώς ζητάει να δει (αριστερά στην Εικόνα 9 του Παράρτημα). Τα κριτήρια είναι συγκεκριμένα:

- Επιλογή του ελεύθερου χρόνου που διαθέτει μέσα από συγκεκριμένες επιλογές (1 ώρα, 2 ώρες ή 3 ώρες).
- Πληκτρολόγηση του διαθέσιμου ποσού που ο χρήστης μπορεί να διαθέσει για την περιήγηση αυτή. Αν το ποσό μείνει κενό, η εφαρμογή καταλαβαίνει ως 0€ και αναζητεί μόνο τα δωρεάν σημεία ενδιαφέροντος.
- Η επιλογή αν ενδιαφέρεται ο χρήστης για τα δημοφιλή, όπου θα βρεθεί διαδρομή δίνοντας βάση στα δημοφιλή πρώτα ή αν δεν τον ενδιαφέρουν, όπου θα βρεθεί διαδρομή αγνοώντας αν είναι δημοφιλή ή όχι.
- Επιλογή κατηγοριών σημείων ενδιαφέροντος. Οι κατηγορίες είναι 4: Μουσεία, Αξιοθέατα, Εκκλησίες, Άλλα. Ο χρήστης επιλέγει μόνο τις κατηγορίες που θέλει να δει.
- Τέλος, υπάρχει και το κουμπί «Χάρτης». Το κουμπί αυτό καθορίζει το σημείο από το οποίο ο χρήστης θέλει να ξεκινήσει τη διαδρομή του. Όπως και στην εύρεση της τοποθεσίας στην προσθήκη νέου σημείου έτσι και εδώ εμφανίζεται ο ίδιος χάρτης (κέντρο στην Εικόνα 9 του Παράρτημα) αυτή τη φορά όμως για να επιλέξει ο χρήστης το σημείο έναρξης της διαδρομής. Αυτό χρησιμεύει στην περίπτωση που ο χρήστης δεν θέλει να ξεκινήσει από το σημείο που βρίσκεται τώρα, αλλά θέλει να προγραμματίσει την διαδρομή του από πριν. Μόλις επιλεγεί ένα σημείο, εμφανίζεται ένα μήνυμα δίπλα στο κουμπί με την διεύθυνση του σημείου έναρξης, όπως φαίνεται στην αριστερή εικόνα της Εικόνα 9 του Παράρτημα

Μόλις επιλεχθούν τα κατάλληλα κριτήρια και ο χρήστης πατήσει αποθήκευση των κριτηρίων, η εφαρμογή υπολογίζει την κατάλληλη διαδρομή και την παρουσιάζει στον χρήστη σε μορφή λίστας. Στη λίστα φαίνονται τα σημεία που προκύπτουν με την κατάλληλη σειρά ανάλογα με τα κριτήρια που ζήτησε ο χρήστης. Αν σύμφωνα με τα κριτήρια δεν ικανοποιούνται οι συνθήκες για να προκύψει κάποιο αποτέλεσμα, η λίστα είναι κενή. Αυτό μπορεί να συμβαίνει για παράδειγμα αν το σημείο έναρξης βρίσκεται πολύ μακριά από το πιο κοντινό σημείο και ο διαθέσιμος χρόνος δεν επαρκεί.

Σε περίπτωση όμως που εμφανίζονται αποτελέσματα (δεξιά στην Εικόνα 9 του Παράρτημα), εμφανίζεται και ένα κουμπί κάτω δεξιά στην οθόνη. Αυτό το κουμπί

είναι το κουμπί της πλοήγησης. Όταν ο χρήστης το πατήσει, ανακατευθύνεται στην προεπιλεγμένη εφαρμογή που έχει ορίσει ο καθένας στο κινητό του (συνήθως το Google Maps) και του δημιουργεί την διαδρομή που έχει προκύψει, ώστε να του δώσει ακριβείς οδηγίες για την περιήγησή του. Επισημαίνεται ότι ο τρόπος κίνησης του χρήστη είναι με τα πόδια.

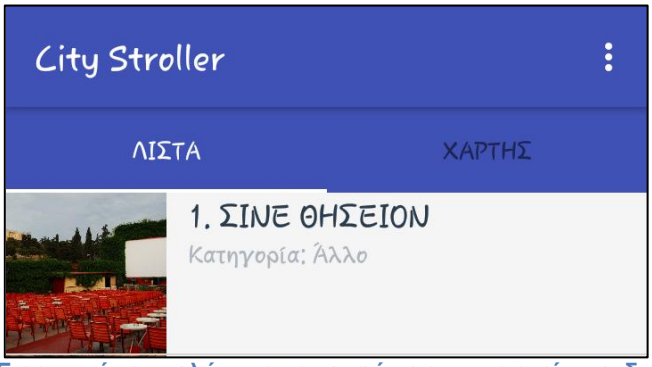
Συμπεράσματα

Η διαδικασία δημιουργίας και επεξεργασίας της εφαρμογής τελείωσε και ακόμα και τώρα η εφαρμογή δεν θεωρείται έτοιμη να ανταγωνιστεί άλλες παρόμοιες που υπάρχουν στην αγορά. Το συμπέρασμα αυτό έγινε κατανοητό κατά την διάρκεια της επεξεργασίας, όταν μετά από κάθε βελτίωση υπήρχε κάτι ακόμα που μπορούσε να βελτιωθεί. Μια νέα ιδέα, μια λειτουργία που κάνει την εφαρμογή αργή ή οτιδήποτε άλλο μπορεί να προστεθεί συνεχώς ώστε να προσφέρει στον χρήστη μια καλύτερη εμπειρία. Επειδή λοιπόν η διαδικασία βελτίωσης δεν σταματάει ποτέ, η εφαρμογή έφτασε πλέον σε ένα ικανοποιητικό στάδιο για επίπεδο διπλωματικής. Άλλωστε αυτό που έπαιζε τον σημαντικότερο ρόλο στην εφαρμογή είναι η δυνατότητα να λύνει το ζητούμενο πρόβλημα για κάθε χρήστη και αυτό επιτυγχάνεται από την προτεινόμενη διαδρομή.

Ωστόσο, για την δημοσίευση της εφαρμογής αυτής πρέπει να γίνουν κάποιες βελτιώσεις, ώστε να προσελκύσει τους χρήστες να την χρησιμοποιήσουν. Αυτές οι βελτιώσεις δεν αφορούν όμως μόνο την εμφάνιση της εφαρμογής, αυτό δηλαδή που φαίνεται προς τα έξω. Οι βελτιώσεις αυτές μπορούν να αφορούν και τον εσωτερικό κώδικα, είτε βελτιώνοντας την ταχύτητα της εφαρμογής είτε αλλάζοντας τον κώδικα με τέτοιο τρόπο που θα δεχθεί πιο εύκολα επόμενες αλλαγές. Επιπλέον, σαν στόχος πλέον δεν είναι μόνο η λύση ενός προβλήματος κάποιας ομάδας ατόμων, αλλά πλέον λαμβάνεται υπόψη και η αλληλεπίδραση με τους χρήστες αλλά και η οικονομική πλευρά της εφαρμογής.

Αναλυτικότερα, πρώτη και κύρια αλλαγή είναι στον τρόπο ανάγνωσης των δεδομένων των σημείων. Στην παρούσα εφαρμογή τα δεδομένα των σημείων εισάγονται στον πίνακα με μορφή κώδικα, που σημαίνει ότι ο κώδικας αυτός καταλαμβάνει πολλές γραμμές και κατ' επέκταση η εφαρμογή είναι πιο αργή. Ο πιο συνηθισμένος τρόπος να πραγματοποιηθεί αυτό είναι με την ανάγνωση ενός αρχείου Excel που θα περιέχει όλα τα δεδομένα των σημείων σε ένα πίνακα. Με την αλλαγή του κώδικα έτσι ώστε να διαβάζει αρχείο, θα είναι εύκολη και η προσθήκη σημείων και άλλων πόλεων και κατ' επέκταση θα απευθύνεται και σε χρήστες που θέλουν να βρουν προτεινόμενη διαδρομή και σε άλλες πόλεις. Επιπλέον κάποιες παραπάνω γλώσσες μετάφρασης θα εξυπηρετούσαν άτομα και από άλλες χώρες. Αυτές οι προσθήκες, όπως είναι προφανές, αυξάνουν τα άτομα που στοχεύει η εφαρμογή και κατ' επέκταση αυξάνονται και οι πιθανότητες μιας καλής πορείας της.

Μια ακόμα σημαντική αλλαγή θα μπορούσε να γίνει στον τρόπο εμφάνισης της αρχικής λίστας. Στην παρούσα φάση ο χρήστης μόλις ανοίξει την εφαρμογή βλέπει μια λίστα με τα διαθέσιμα σημεία ενδιαφέροντος. Στο πάνω μέρος της οθόνης υπάρχει το μενού της λίστας αυτής με κάποια κουμπιά. Ένα από αυτά σε πηγαίνει στον χάρτη της περιοχής, όπου φαίνονται τα σημεία αυτά σε χάρτη και από εκεί μπορείς να γυρίσεις πίσω στη λίστα πάλι. Ουσιαστικά ο χρήστης αλλάζει δραστηριότητα για να δει σε άλλη μορφή την αρχική οθόνη. Μια πιο σωστή λογική είναι στην αρχική οθόνη να φαίνονται ταμπέλες με τις δύο μορφές εμφάνισης των σημείων ενδιαφέροντος, όπως φαίνεται παρακάτω στην Εικόνα 5. Οι δύο αυτές μορφές, λίστα και χάρτης, θα είναι στην ίδια δραστηριότητα και ο χρήστης θα μπορεί να πηγαίνει από την μια στην άλλη χωρίς να γυρίζει πίσω. Ουσιαστικά θα αλλάζει τον τρόπο προβολής των δεδομένων. Η τεχνική των ταμπελών χρησιμοποιείται αρκετά συχνά πλέον σε πολλές εφαρμογές και αποτελεί ένα εύκολο και κατανοητό τρόπο δόμησης για πολλούς.



Εικόνα 5: Εισαγωγή ταμπελών για την εμφάνιση των σημείων ενδιαφέροντος

Σημαντικές διορθώσεις πρέπει να γίνουν επίσης και στην διαδικασία υπολογισμού της προτεινόμενης διαδρομής. Με βάση αυτή την έκδοση της εφαρμογής, οι αποστάσεις από το σημείο έναρξης προς όλα τα σημεία υπολογίζονται με χρήση του Google Distance Matrix Api. Αυτό αρχικά προϋποθέτει σύνδεση στο Διαδίκτυο και δεύτερον επειδή λαμβάνει δεδομένα από την Google, η διαδικασία γίνεται πιο αργή από το να χρησιμοποιούνταν ο τύπος της Manhattan απόστασης. Επιπλέον σε προχωρημένο στάδιο, όπου θα υπάρχουν αρκετοί χρήστες της εφαρμογής, τα δεδομένα αυτά θα χρεώνονται από την Google, γεγονός που είναι προτιμότερο να αποφευχθεί. Άλλωστε, η διαφορά μεταξύ των δεδομένων από την Goolge και των δεδομένων από τον τύπο δεν είναι τόσο μεγάλη ώστε να αλλάξει πολύ την προτεινόμενη διαδρομή, αφού όλες οι υπόλοιπες αποστάσεις είναι με ακριβή δεδομένα. Άρα με την χρήση του τύπου της Manhattan απόστασης για τις αρχικές αποστάσεις αυξάνεται η ταχύτητα, μειώνεται το κόστος και παράλληλα δεν χρειάζεται σύνδεση στο διαδίκτυο, χωρίς όμως να μειώνεται αρκετά η ακρίβεια της προτεινόμενης διαδρομής.

Μια αλλαγή που θα μπορούσε να κάνει τον υπολογισμό της προτεινόμενης απόστασης πιο γρήγορο θα μπορούσε να είναι η μείωση των δεδομένων. Όπως αναφέρθηκε και παραπάνω τα αρχικά σημεία που φαίνονται στην εφαρμογή είναι 267. Από αυτά περίπου τα μισά έχουν να κάνουν με εκκλησίες (για λόγους διπλωματικής), γεγονός που δεν υποδηλώνει μια ομοιόμορφη κατανομή των τεσσάρων κατηγοριών. Επιπλέον οι περισσότερες εκκλησίες που έχουν καταγραφεί δεν αποτελούν σημεία ενδιαφέροντος, αλλά απλές εκκλησίες της περιοχής μελέτης. Αρχικά μια αλλαγή θα μπορούσε να είναι η κατάργηση της κατηγορίας των εκκλησιών και ενσωμάτωση μόνο των αναγκαίων πιο σημαντικών εκκλησιών στην κατηγορία «Άλλα». Τα θετικά της μείωσης της ποσότητας των σημείων θα είναι πολλά:

- Μείωση του μεγέθους της βάσης δεδομένων και κατ' επέκταση και του μεγέθους της εφαρμογής στο κινητό του χρήστη.
- Αύξηση της ταχύτητας της εφαρμογής αλλά και της ταχύτητας υπολογισμού της προτεινόμενης διαδρομής, αφού θα απαιτείται ο υπολογισμός λιγότερων αποστάσεων από το σημείο έναρξης.
- Υπολογισμός προτεινόμενης διαδρομής με πιο ενδιαφέροντα αξιοθέατα και πιο κοντά στις απαιτήσεις και προτιμήσεις των χρηστών.
- Ύπαρξη 3 κατηγοριών σημείων με πιο ομοιόμορφη κατανομή αντί για 4, με αποτέλεσμα την διευκόλυνση του χρήστη και την απλότητα στην επιλογή των κριτηρίων.

Το τελευταίο χαρακτηριστικό αποτελεί μια σημαντική παρατήρηση όσον αφορά τον ορισμό των κριτηρίων. Μετά από χρήση της εφαρμογής από άλλα άτομα εξάχθηκε το συμπέρασμα ότι τα κριτήρια πρέπει να ορίζονται όσο πιο απλά και γρήγορα γίνεται. Ο κάθε χρήστης για να ορίσει τα κριτήρια πρέπει να πάρει κάποιες αποφάσεις. Πρέπει να σκεφτεί πόσο χρόνο έχει στη διάθεσή του, πόσα χρήματα, τι θέλει να δει από κατηγορίες αξιοθέατων, αν τον ενδιαφέρουν τα δημοφιλή και από πού θέλει να ξεκινήσει, κριτήρια που θέλουν κάποιο χρόνο για να τα ορίσει ο

χρήστης. Από τη μια όσο πιο λίγες είναι οι κατηγορίες των σημείων, τόσο λιγότερες είναι οι αποφάσεις του χρήστη και κατ' επέκταση τόσο πιο γρήγορη και ευχάριστη θα είναι η εμπειρία του. Από την άλλη όμως όσο μειώνονται οι κατηγορίες τόσο μειώνεται και η πιθανότητα να βρίσκεται η προτεινόμενη διαδρομή κοντά στις προτιμήσεις του χρήστη. Αυτό σημαίνει ότι στην κατηγορία «Άλλα» υπάρχουν πανεπιστήμια, πάρκα αλλά και γήπεδα. Για μια γυναίκα, που μπορεί να προτιμάει να πάει σε πάρκα και πανεπιστήμια αλλά να μην προτιμάει να επισκεφθεί γήπεδα δεν είναι εύκολο να αποφασίσει αν επιθυμεί να επιλέξει αυτή την κατηγορία. Άρα το καλύτερο σενάριο στην κατηγοριοποίηση των σημείων θα ήταν κάποιες κατηγορίες πιο επικεντρωμένες στα αξιοθέατα αλλά παράλληλα να υπάρχει καλή ταξινόμηση των σημείων στις κατηγορίες αυτές. Επομένως, ο καθορισμός των κριτηρίων πρέπει από την μια να μπορεί να καθορίσει με τον καλύτερο τρόπο τις προτιμήσεις του χρήστη, αλλά από την άλλη τα κριτήρια να είναι λίγα ώστε να μην καταναλώνεται πολύς χρόνος στην επιλογή τους.

Ένα κριτήριο που θα μπορούσε να αλλάξει και να γίνει πιο εναλλακτικό για τον χρήστη είναι ο ελεύθερος χρόνος. Αυτή τη στιγμή ο χρήστης μπορεί να επιλέξει τρεις εναλλακτικές για ελεύθερο χρόνο (1, 2 και 3 ώρες) περιορίζοντας έτσι τις εναλλακτικές του. Μια βελτίωση, λοιπόν, πέρα από την διαφοροποίηση των κατηγοριών που αναφέρθηκε παραπάνω, θα μπορούσε να είναι ο ορισμός όσου ελεύθερου χρόνου επιθυμεί ο χρήστης. Δηλαδή αντί για μια λίστα με δεδομένες επιλογές ωρών, να μπορεί ο χρήστης να πληκτρολογήσει σε ένα πλαίσιο κειμένου πόσες ώρες έχει στη διάθεσή του και να δημιουργείται με αυτό το δεδομένο προτεινόμενη διαδρομή. Άλλωστε αν θεωρήσουμε ότι κάποια μεγάλα μουσεία μπορεί να έχουν χρόνο περιήγησης για κάποιον χρήστη κοντά στις 2 ώρες αμέσως μπορεί με τις συγκεκριμένες επιλογές είτε να απορριφθούν είτε να καταναλωθεί σχεδόν όλος ο ελεύθερος χρόνος ενός χρήστη εκεί με αποτέλεσμα να μην προλάβει να δει άλλα αξιοθέατα.

Σημαντικό στοιχείο της εφαρμογής που διαφέρει από πάρα πολλές άλλες είναι η αλληλεπίδραση του χρήστη με τον προγραμματιστή. Με βάση την εφαρμογή στην παρούσα μορφή, ο χρήστης είναι ανώνυμος. Δεν γίνεται καμία εγγραφή χρήστη και ο προγραμματιστής δεν γνωρίζει ποιος χρήστης χρησιμοποιεί την εφαρμογή του, ούτε τι αλλαγές και τι ενέργειες έχει κάνει. Η μόνη επικοινωνία που έχει ο χρήστης με τον προγραμματιστή βρίσκεται στο σημείο αποστολής των δεδομένων στον προγραμματιστή και αυτό αν και όταν ο χρήστης το επιλέξει. Μια βασική λοιπόν αλλαγή θα μπορούσε να ήταν η δημιουργία λογαριασμού για κάθε χρήστη. Αρχική οθόνη της εφαρμογής πλέον δεν θα είναι η λίστα με τα αξιοθέατα, αλλά μια οθόνη που θα επιτρέπει στον κάθε χρήστη να εγγραφεί σε μια βάση χρηστών του προγραμματιστή. Τα εργαλεία αυτά είναι έτοιμα από την Android και την Google και βοηθούν να κάνουν εγγραφή οι χρήστες κάθε εφαρμογής είτε μέσω mail είτε μέσω Facebook είτε μέσω του αριθμού του κινητού τους τηλεφώνου (μέσω αποστολής μηνύματος μοναδικού κωδικού). Ανάλογα τι θα επιλέξει ο χρήστης, αποστέλλονται και οι διαθέσιμες πληροφορίες για κάθε χρήστη και δημιουργείται ένας λογαριασμός για τον καθένα.

Όλα τα παραπάνω επιτρέπουν στον προγραμματιστή να ελέγχει την λειτουργία και τις αστοχίες της εφαρμογής ώστε να την βελτιώσει και να διορθώσει τυχόν σφάλματα ή αδυναμίες της. Ωστόσο, υπάρχουν πολλά πλεονεκτήματα στην αλλαγή αυτή και πόσο μάλλον αν ο χρήστης δημιουργήσει λογαριασμό μέσω του λογαριασμού του στο Facebook. Το Facebook παρέχει μια πλατφόρμα για προγραμματιστές που έχουν χρησιμοποιήσει το εργαλείο εγγραφής χρηστών στην εφαρμογή τους. Από εκεί μπορεί να έχουν πρόσβαση σε όλα τα στοιχεία κάθε χρήστη, καθώς επίσης να εξάγουν στατιστικά για τις ηλικίες των χρηστών, την διάρκεια χρήσης της εφαρμογής και άλλα δεδομένα που μπορούν να βοηθήσουν στην βελτίωση της εφαρμογής. Επιπλέον, δημιουργώντας ο χρήστης ένα λογαριασμό, θα μπορεί να επιβεβαιωθεί ποιος στέλνει δεδομένα στον προγραμματιστή, καθώς τα δεδομένα του Facebook είναι πιο ακριβή από τα στοιχεία που καταχωρούνται από τον χρήστη πριν την αποστολή των δεδομένων. Ωστόσο το

μόνο αρνητικό σε αυτό είναι η αναγκαστική ύπαρξη Διαδίκτυο κατά την εγγραφή του χρήστη, γεγονός που δεν παίζει και πολύ σημαντικό ρόλο.

Σημαντική αλλαγή ακόμα κατά την δημιουργία ενός λογαριασμού θα μπορούσε να είναι η καταγραφή κάθε προτεινόμενης διαδρομής σε ένα ιστορικό στον λογαριασμό κάθε χρήστη. Ο χρήστης θα έχει ένα σημείο όπου θα καταγράφεται κάθε πιθανή προτεινόμενη διαδρομή που έχει δημιουργηθεί για αυτόν. Από εκεί θα μπορεί να δει αναλυτικά τις κινήσεις του. Επιπλέον, με δεδομένο ότι ο χρήστης έχει συνδέσει το λογαριασμό του Facebook του, θα μπορεί να κοινοποιήσει οποιαδήποτε προτεινόμενη διαδρομή θέλει στα μέσα μαζικής δικτύωσης διαφημίζοντας παράλληλα και την παρούσα εφαρμογή.

Μια ακόμα προσθήκη που θα μπορούσε να είναι αρκετά χρήσιμη για τους χρήστες και θα πήγαινε την εφαρμογή σε άλλο επίπεδο θα ήταν η αντίστροφη διαδικασία από αυτή της προτεινόμενης διαδρομής. Αναλυτικότερα, ο χρήστης θα μπορούσε να δηλώσει τα μέρη που θέλει να επισκεφθεί (όχι σε σειρά επίσκεψης) και η εφαρμογή θα μπορούσε να του έβγαζε την πιο σύντομη διαδρομή, καθώς επίσης και τον χρόνο και τα χρήματα που θα χρειαστεί για να επισκεφθεί αυτά τα σημεία ενδιαφέροντος. Μέρη της διαδικασίας αυτής μπορεί να προσφέρονται και από άλλες εφαρμογές, ωστόσο όλες αυτές τις πληροφορίες συγκεντρωμένες σε ένα σημείο δύσκολα μπορεί να βρει κανείς.

Τέλος, κάποιες τελευταίες διορθώσεις και προσθήκες θα μπορούσαν να γίνουν στην δραστηριότητα των πληροφοριών του κάθε σημείου. Η δόμηση όλων των στοιχείων της οθόνης στην δραστηριότητα των πληροφοριών του κάθε σημείου αποδείχθηκε μη πρακτική, καθώς από κινητό σε κινητό η δομή των στοιχείων αλλάζει. Υπάρχουν πολύ πιο εύχρηστα πρότυπα σύμφωνα και με άλλες εφαρμογές που μπορούν να παρουσιάσουν τις πληροφορίες ενός σημείου. Επιπλέον, οι εικόνες που έχουν ληφθεί προς το παρόν είναι καθαρά δοκιμαστικές για λόγους διπλωματικής, όπως επίσης και οι χρόνοι περιήγησης. Καλύτερες και περισσότερες φωτογραφίες για κάθε σημείο θα μπορούσαν να αποτελέσουν μια προφανώς πιο ελκυστική λύση. Επιπροσθέτως και η δυνατότητα προσθήκης εικόνων στα νέα σημεία που προσθέτει ο χρήστης θα ήταν μια λύση που θα έκανε την εφαρμογή πιο οργανωτική και ακριβή στα δεδομένα της. Επιπλέον ένα κουμπί κλήσης θα μπορούσε να είναι αρκετά χρήσιμο, καθώς δεν υπάρχει η δυνατότητα απευθείας κλήσης από την εφαρμογή. Έτσι, ο χρήστης δεν θα πρέπει να θυμάται τον αριθμό, για παράδειγμα ενός μουσείου, και να τον πληκτρολογήσει μόνος του. Και τέλος από τις πληροφορίες του κάθε σημείου δίνεται η δυνατότητα προβολής του σημείου στο χάρτη. Στην παρούσα φάση ο χρήστης μεταβαίνει στην δραστηριότητα του χάρτη, όπου φαίνονται όλα τα σημεία με επιλεγμένο το ζητούμενο. Μετά από δοκιμές της εφαρμογής σε άλλους χρήστες, βγήκε το συμπέρασμα ότι όταν ο χρήστης θέλει να δει το σημείο στο χάρτη, πρέπει να φαίνεται μόνο αυτό το σημείο.

Με δεδομένες λοιπόν τις παραπάνω αλλαγές, η εφαρμογή θα είναι έτοιμη πλέον να δημοσιευτεί και να χρησιμοποιηθεί από κάθε χρήστη με περισσότερο ενδιαφέρον. Προφανώς οι αλλαγές είναι πολλές, ωστόσο οι περισσότερες είναι απαραίτητες για να γίνει η εφαρμογή ανταγωνιστική και ελκυστική προς τους χρήστες. Συνοπτικά, σαν εφαρμογή καλύπτει το ζητούμενο της διπλωματικής με πολλά περιθώρια βελτίωσης. Άλλωστε, όπως αναφέρθηκε και παραπάνω συνήθως στις εφαρμογές των κινητών πάντα υπάρχουν περιθώρια βελτίωσης και προσθήκες που μπορούν να την κάνουν καλύτερη. Έτσι η διαδικασία βελτίωσης δεν σταματάει ποτέ.

ΣΥΜΠΕΡΑΣΜΑΤΑ – ΠΡΟΤΑΣΕΙΣ

Στο κεφάλαιο αυτό θα αναλυθούν τα συμπεράσματα που προέκυψαν από την διαδικασία που χρησιμοποιήθηκε μέχρι το τελικό προϊόν, την εφαρμογή που περιγράφηκε στο προηγούμενο κεφάλαιο. Θα παρουσιαστούν τα θετικά και τα αρνητικά σημεία της προσέγγισης που έγινε, ενώ παράλληλα θα προταθούν κάποιες βελτιώσεις που θα μπορούσαν να ελαττώσουν τα μειονεκτήματα της διαδικασίας.

Όπως αναφέρθηκε και στην αρχή της εργασίας, οι άνθρωποι πλέον έχουν πολλές επιλογές να εκμεταλλευτούν τον ελεύθερο χρόνο τους σε μια πόλη. Όταν μάλιστα αυτό αφορά τους τουρίστες ο χρόνος τους είναι ακόμα πιο περιορισμένος, αφού επισκέπτονται μια πόλη σε ένα περιορισμένο χρονικό διάστημα με δεδομένη την μέρα άφιξης και επιστροφής. Μέσα σε αυτές τις μέρες, που σπάνια μπορεί να είναι και ώρες, πρέπει να οργανώσουν τέλεια τον χρόνο τους ώστε να προλάβουν να επισκεφθούν όσο πιο πολλά μέρη γίνεται και να γνωρίσουν όσο το δυνατόν καλύτερα την ιστορία της πόλης μέσα από αυτά. Αυτό ισχύει και για ένα κάτοικο της πόλης, που έχει ελεύθερο χρόνο μέχρι την επόμενη δουλειά του και ψάχνει κάτι να τον αξιοποιήσει. Επιπλέον, πέρα από το χρόνο πολλοί λαμβάνουν υπόψη τους και τα χρήματα που θέλουν να διαθέσουν για να επισκεφθούν ένα σημείο ενδιαφέροντος.

Για να θεωρείται λοιπόν το αποτέλεσμα αυτής της εργασίας επιτυχές, θα πρέπει η εφαρμογή που δημιουργήθηκε να λύνει το παραπάνω πρόβλημα. Θα πρέπει να είναι σε θέση να οργανώσει τον ελεύθερο χρόνο ενός χρήστη με τέτοιο τρόπο ώστε να λαμβάνει υπόψη του κάποια κριτήρια. Σήμερα τα εργαλεία για να ψάξει κανείς τις επιλογές και να οργανώσει το χρόνο του μόνος του είναι πολλά. Αυτό που λείπει είναι να οργανώνεται ο ελεύθερος χρόνος κάθε χρήστη αυτόματα και πόσο μάλλον με βάση τα χρήματα που θέλει να διαθέσει. Αυτό μπορεί πλέον κανείς να το καταφέρει μέσα από το City Stroller. Επομένως μπορεί να θεωρηθεί ότι η εφαρμογή αυτή πέτυχε τον σκοπό της και μπορεί πλέον να χρησιμοποιηθεί.

Η διαδικασία που ακολουθήθηκε για να παραχθεί το τελικό προϊόν ήταν συγκεκριμένη και αναφέρεται και σε προηγούμενο κεφάλαιο της εργασίας. Αφού εντοπίστηκε το πρόβλημα, αρχικά επιλέχθηκε η περιοχή μελέτης και συλλέχθηκαν δεδομένα για αυτή. Στην συνέχεια δημιουργήθηκε η εφαρμογή και μέσα από προσθήκες και δοκιμές κατέληξε στο τελικό προϊόν. Ωστόσο, κατά την διάρκεια της διαδικασίας υπήρχαν κάποια λάθη, τα οποία έγιναν αντιληπτά σε μεταγενέστερο στάδιο και γι’ αυτό το λόγο παρουσιάζονται εδώ.

Το πρώτο και σημαντικότερο λάθος ήταν στο στάδιο της συλλογής των δεδομένων των σημείων. Σαν δεδομένα αρχικά ήταν όλες οι πληροφορίες για κάθε αξιοθέατο. Η διαδικασία εύρεσης και καταγραφής των δεδομένων αυτών έγινε με το QGIS και ήταν ήδη γνωστή. Για τον λόγο αυτό ξεκίνησε η διαδικασία καταγραφής δεδομένων χωρίς τις βασικές γνώσεις προγραμματισμού που αποκτήθηκαν στην πορεία. Ωστόσο, η διαδικασία αυτή έγινε χειροκίνητα με αποτέλεσμα να τοποθετούνται σημεία στον χάρτη με μια σχετικά μικρή απόκλιση από τα δεδομένα που παρέχουν οι χάρτες της Google.

Αργότερα, όταν στην εφαρμογή προστέθηκε η λειτουργία για προσθήκη νέου σημείου, όπου χρησιμοποιείται ένα έτοιμο εργαλείο και μπορεί κανείς να προσθέσει μέρη μέσω των δεδομένων της Google, παρατηρήθηκε ότι θα ήταν πιο εύκολο να απομονωθεί αυτή η λειτουργία σε μια νέα εφαρμογή και να γίνει έτσι η συλλογή των σημείων. Με αυτόν τον τρόπο, ο προγραμματιστής θα μπορούσε να συλλέξει τα δεδομένα του μέσω των δεδομένων της Google και να τα εξάγει σε ένα αρχείο Excel, το οποίο θα διαβάζει η κύρια εφαρμογή. Έτσι, μπορεί ο προγραμματιστής να έχει πιο ακριβή δεδομένα μέσα από μια πιο γρήγορη αυτόματη διαδικασία. Τα δεδομένα που μπορεί να έχει πρόσβαση είναι το όνομα του σημείου, την διεύθυνση, το τηλέφωνο επικοινωνίας, την τιμή εισόδου, όπου αυτή παρέχεται και πολλά άλλα. Τα σημεία που θα δημιουργούνται θα συμπίπτουν με τα δεδομένα που παρέχονται στους χάρτες της Google. Ο τρόπος αυτός δημιουργεί προοπτικές για προσθήκες σημείων νέων πόλεων πέρα της Αθήνας και είναι μια βελτίωση, η οποία θα πραγματοποιηθεί μετά την ολοκλήρωση της εργασίας.

Σημαντικό πρόβλημα προέκυψε επίσης με τα δεδομένα των αποστάσεων από κάθε σημείο προς όλα τα υπόλοιπα. Όπως αναφέρθηκε και παραπάνω, τα δεδομένα αυτά τελικά υπολογίστηκαν μέσα από τη δημιουργία μιας άλλης εφαρμογής, η οποία λάμβανε δεδομένα κάθε μέρα μέχρι την συγκέντρωση όλων των αποστάσεων. Με τον τρόπο αυτό χρειάστηκε περίπου ένας μήνας συλλογής δεδομένων, για να αποφευχθεί κάποια χρέωση από την Google. Μέσα από έρευνα για εύρεση καλύτερου και ακριβέστερου τρόπου, αυτός ήταν ο πιο αποτελεσματικός και οικονομικός.

Για να καταλήξει σε αυτόν τον τρόπο η έρευνα υπήρξαν πολλές επιλογές. Λόγω του πλήθους των δεδομένων αρχικά ήταν αδύνατο να εισάγουμε μια – μια τις συντεταγμένες στους χάρτες και να υπολογιστούν οι αποστάσεις. Κάποιος τύπος επίσης δεν μπορούσε να υπολογίσει με ακρίβεια τον χρόνο μετάβασης από ένα σημείο προς κάποιο άλλο. Υπήρχαν κάποια εργαλεία που παρέχονται από το ArcGis Online για την επίλυση τέτοιων προβλημάτων. Ωστόσο, η χρέωση για να χρησιμοποιηθούν ήταν πολύ μεγάλη. Αυτό ίσχυε και για την χρήση των εργαλείων μέσω της σχολής των Αγρονόμων & Τοπογράφων Μηχανικών του Ε.Μ.Π., γεγονός που περιόριζε ακόμα πιο πολύ τις επιλογές. Επομένως μετά την απόρριψη όλων αυτών των τρόπων χρησιμοποιήθηκε αρχικά ο τύπος της Ευκλείδειας απόστασης, της ευθείας απόστασης δηλαδή που ενώνει τα δύο σημεία. Στην συνέχεια έγινε σύγκριση κάποιων τιμών με τις τιμές που προκύπτουν από τους χάρτες της Google και οι τιμές ήταν αρκετά ικανοποιητικές. Μετά από έρευνα διαπιστώθηκε ότι επειδή τα σημεία βρίσκονται μέσα σε πόλη η καλύτερη απόσταση μεταξύ των σημείων ήταν η Manhattan απόσταση και αυτή χρησιμοποιήθηκε. Και πάλι οι αποκλίσεις δεν ήταν πολύ μεγάλες. Τέλος, η εφαρμογή κατέληξε στα δεδομένα που έχει τώρα, τα οποία είναι αρκετά ικανοποιητικά. Κάποιες αποστάσεις που δεν μπορούσαν να υπολογιστούν λόγω μεγάλης απόστασης κυρίως, χρησιμοποιήθηκε ο τύπος της Manhattan απόστασης. Αυτό ήταν και το κύριο πρόβλημα το οποίο αντιμετωπίστηκε από την αρχή μέχρι το τέλος.

Όσον αφορά το Android Studio, το πρόγραμμα δηλαδή που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής, αυτό ήταν πολύ εύκολο και εύχρηστο σχεδιασμένο για να μπορεί κάθε προγραμματιστής να συμπεριλάβει ό,τι λειτουργία επιθυμεί. Οι πληροφορίες ευτυχώς που παρέχονταν στο διαδίκτυο καλύπτουν όλο το φάσμα των ενεργειών που θέλει κανείς να χρησιμοποιήσει σε μια εφαρμογή και δεν υπήρχε καμία σημαντική δυσκολία στην ανάπτυξη της εφαρμογής. Όλες οι προσθήκες με δοκιμές και αναζήτηση στις κατάλληλες πηγές έφτασαν στο καλύτερο επίπεδο. Προφανώς κατά την διάρκεια της ανάπτυξης της εφαρμογής βρέθηκαν πολλά σφάλματα, τα οποία δεν ήταν από την αρχή προφανή, αλλά με αρκετές δοκιμές εντοπίστηκαν και επιλύθηκαν.

Όσον αφορά τώρα το κομμάτι της εφαρμογής, αυτή χρειάζεται αρκετές βελτιώσεις για να δημοσιευτεί στο Google Play. Οι βελτιώσεις αυτές αναλύονται στο προηγούμενο κεφάλαιο. Προφανώς, σε επίπεδο εργασίας η εφαρμογή καλύπτει αρκετά καλά το ζητούμενο πρόβλημα, ωστόσο δεν είναι φιλική προς τον χρήστη, γεγονός που απαιτεί γνώσεις σχεδιαστή. Για λόγους γνώσεων και εμπειρίας, θα δοκιμαστούν όλες οι παραπάνω βελτιώσεις και θα γίνει προσπάθεια δημοσίευσης της εφαρμογής μέσα στους επόμενους μήνες.

Βιβλιογραφία

1. http://www.tripadvisor.com.gr
2. http://www.arcgis.com/features/index.html
3. http://desktop.arcgis.com/en/
4. http://www.qgis.org/en/site/
5. http://www.esri.com/
6. http://www.openstreetmap.org/
7. https://developers.arcgis.com/
8. http://www.diva-portal.org/
9. https://www.wwf.gr/
10. https://www.kantarworldpanel.com
11. https://www.udacity.com/
12. https://stackoverflow.com/
13. https://services.github.com/
14. https://developer.android.com/index.html
15. https://developers.google.com/
16. https://www.youtube.com
17. https://material.io/
18. https://developer.xamarin.com/
19. https://android-developers.googleblog.com/
20. https://www.materialpalette.com/
21. http://abhiandroid.com/
22. https://el.wikipedia.org/
23. https://en.wiktionary.org/
24. http://www.ieee.ma
25. http://www.w3schools.com
26. https://math.stackexchange.com/

ΠΑΡΑΡΤΗΜΑ

Τύπος υπολογισμού Manhattan απόστασης

```
//Κώδικας υπολογισμού της Manhattan απόστασης
public final Integer calcManhattanDistance (double lat1, double long1,
double lat2, double long2){
    int dist1 = calcDistTimeInMins(lat1, long1, lat1, long2);
    int dist2 = calcDistTimeInMins(lat1, long2, lat2, long2);
    return dist1 + dist2;
}

private Integer calcDistTimeInMins (double lat1, double long1, double lat2,
double long2){
    int R = 6371; //in km
    double lat1_rad = deg2rad(lat1);
    double lat2_rad = deg2rad(lat2);
    double dLat = deg2rad(lat2 - lat1);
    double dLong = deg2rad(long2 - long1);

    double a = Math.sin(dLat/2) * Math.sin(dLat/2) +
            Math.cos(lat1_rad) * Math.cos(lat2_rad) * Math.sin(dLong/2) *
Math.sin(dLong/2);

    double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
    double distanceInKm = R * c;

    return (int) (distanceInKm * 60/WALKSPEED);
}

private double deg2rad (double degrees){
    return degrees * (Math.PI/180);
}
```

Κώδικας υπολογισμού χρόνων αποστάσεων

```
//Γενικός κώδικας υπολογισμού των αποστάσεων
public final Integer walkingDistanceInMins(Double lat1, Double long1, Double
lat2, Double long2) {

    String urlString =
"https://maps.googleapis.com/maps/api/distancematrix/json?origins=" + lat1 +
"," + long1
            + "&destinations=" + lat2 + "," + long2 +
"&mode=walking&key=************************************";

    URL url = null;
    try {
        url = new URL(urlString);
    } catch (MalformedURLException e) {
        Log.e(LOG_TAG, "Problem building the URL", e);
    }

    String jsonResponse = null;
    try {
        jsonResponse = makeHttpRequest(url);
    } catch (IOException e) {
        Log.e(LOG_TAG, "Problem making the HTTP request.", e);
    }

    String duration = "-1";
    if (!TextUtils.isEmpty(jsonResponse)) {
        try {
            JSONObject baseJsonResponse = new JSONObject(jsonResponse);
            JSONArray rowsArray = baseJsonResponse.getJSONArray("rows");
            JSONObject currentRow = rowsArray.getJSONObject(0);
            JSONArray elements = currentRow.getJSONArray("elements");
            JSONObject currentElement = elements.getJSONObject(0);
            String status = currentElement.getString("status");
            if (status.equals("OK")) {
                JSONObject durationObject =
currentElement.getJSONObject("duration");
                duration = durationObject.getString("text");
            }
        } catch (JSONException e) {
            Log.e("QueryUtils", "Problem parsing the distances JSON
results", e);
        }
    }
    String LOCATION_SEPARATOR1 = " hour ";
    String LOCATION_SEPARATOR2 = " hours ";

    int time;
    if (duration.equals("-1")) {
        //calculate Manhattan Distance
        time = new SightsContract().calcManhattanDistance(lat1, long1, lat2,
long2);
    } else if (duration.contains(LOCATION_SEPARATOR1)) {
        String[] parts = duration.split(LOCATION_SEPARATOR1);
        int hours = Integer.valueOf(parts[0]);
        int mins = Integer.valueOf(parts[1].substring(0, parts[1].length() -
4).trim());
        time = 60 * hours + mins;
    } else if (duration.contains(LOCATION_SEPARATOR2)) {
        String[] parts = duration.split(LOCATION_SEPARATOR2);
        int hours = Integer.valueOf(parts[0]);
        int mins = Integer.valueOf(parts[1].substring(0, parts[1].length() -
4).trim());
        time = 60 * hours + mins;
    } else {
        duration = duration.substring(0, duration.length() - 4);
        time = Integer.valueOf(duration.trim());
    }
    return time;
```

```
}

private String makeHttpRequest(URL url) throws IOException {
    String jsonResponse = "";

    if (url == null) {
        return jsonResponse;
    }

    HttpURLConnection urlConnection = null;
    InputStream inputStream = null;
    try {
        urlConnection = (HttpURLConnection) url.openConnection();
        urlConnection.setReadTimeout(10000 /* milliseconds */);
        urlConnection.setConnectTimeout(15000 /* milliseconds */);
        urlConnection.setRequestMethod("GET");
        urlConnection.connect();

        inputStream = urlConnection.getInputStream();
        jsonResponse = readFromStream(inputStream);

    } catch (IOException e) {
        Log.e(LOG_TAG, "Problem retrieving the distance JSON results.", e);
    } finally {
        if (urlConnection != null) {
            urlConnection.disconnect();
        }
        if (inputStream != null) {
            inputStream.close();
        }
    }
    return jsonResponse;
}

private String readFromStream(InputStream inputStream) throws IOException {
    StringBuilder output = new StringBuilder();
    if (inputStream != null) {
        InputStreamReader inputStreamReader = new
InputStreamReader(inputStream, Charset.forName("UTF-8"));
        BufferedReader reader = new BufferedReader(inputStreamReader);
        String line = reader.readLine();
        while (line != null) {
            output.append(line);
            line = reader.readLine();
        }
    }
    return output.toString();
}
```

Διαδικασία δημιουργίας προτεινόμενης διαδρομής

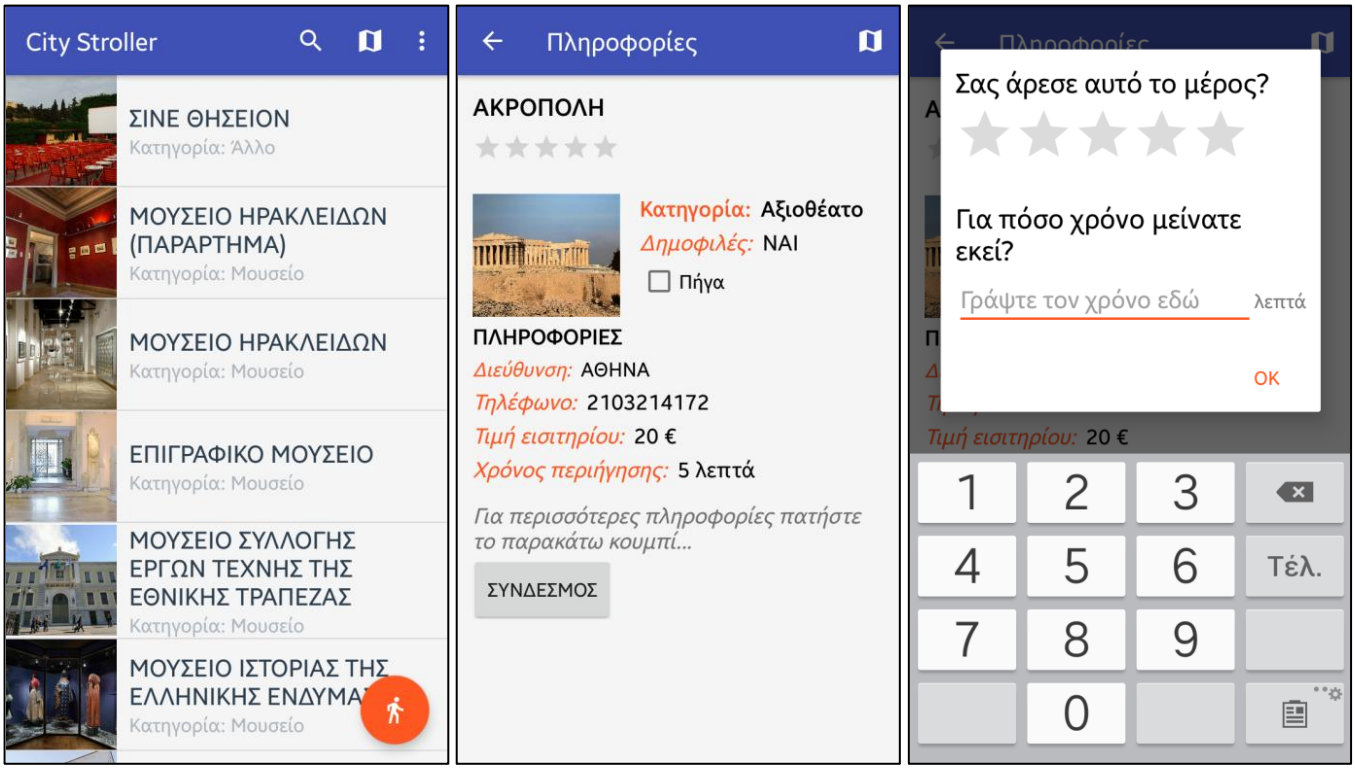
```
ProgressBar loadIndicator;
TextView emptyView;
private double startLat, startLong;
private int freeTime_Crit, budget, museum, attraction, church, other,
popular;

ArrayList<String> listOfIds = new ArrayList<>();
ArrayList<Double> long_points = new ArrayList<>();
ArrayList<Double> lat_points = new ArrayList<>();

// Update "dist_time" column with distances from starting point
private void updateDistances() {

    // Update all distances to -1
    Cursor fullCursor =
getContentResolver().query(SightsEntry.CONTENT_URI_SIGHTS, null, null, null,
null);
    if (fullCursor.getCount() > 0) {
        fullCursor.moveToFirst();
        do {
            int id =
fullCursor.getInt(fullCursor.getColumnIndex(SightsEntry._ID));
            Uri sightUri =
ContentUris.withAppendedId(SightsEntry.CONTENT_URI_SIGHTS, id);

            ContentValues timeValues = new ContentValues();
            timeValues.put(SightsEntry.COLUMN_DIST_TIME, -1);
            getContentResolver().update(sightUri, timeValues, null, null);

        } while (fullCursor.moveToNext());
        fullCursor.close();
    }

    //Update specific distances
    String selection = SightsEntry.COLUMN_PRICE + " <=" +
String.valueOf(budget) + " AND ("
            + SightsEntry.COLUMN_CATEGORY + "=? OR " +
SightsEntry.COLUMN_CATEGORY + "=? OR "
            + SightsEntry.COLUMN_CATEGORY + "=? OR " +
SightsEntry.COLUMN_CATEGORY + "=?) AND "
            + SightsEntry.COLUMN_SEEN + "=" + SightsEntry.NO;
    String[] selectionArgs = {"", "", "", ""};

    if (museum == 1) {
        selectionArgs[0] = String.valueOf(SightsEntry.CATEGORY_MUSEUM);
    }
    if (attraction == 1) {
        selectionArgs[1] = String.valueOf(SightsEntry.CATEGORY_ATTRACTION);
    }
    if (church == 1) {
        selectionArgs[2] = String.valueOf(SightsEntry.CATEGORY_CHURCH);
    }
    if (other == 1) {
        selectionArgs[3] = String.valueOf(SightsEntry.CATEGORY_OTHER);
    }

    Cursor sightsCursor =
getContentResolver().query(SightsEntry.CONTENT_URI_SIGHTS, null, selection,
selectionArgs, null);

    if (sightsCursor.getCount() > 0) {
        sightsCursor.moveToFirst();
        int rowsUpdated = 0;
        do {
            int id =
sightsCursor.getInt(sightsCursor.getColumnIndex(SightsEntry._ID));
            Uri sightUri =
ContentUris.withAppendedId(SightsEntry.CONTENT_URI_SIGHTS, id);
```

```
            double sightLat =
sightsCursor.getDouble(sightsCursor.getColumnIndex(SightsEntry.COLUMN_LATITU
DE));
            double sightLong =
sightsCursor.getDouble(sightsCursor.getColumnIndex(SightsEntry.COLUMN_LONGIT
UDE));
            int time = 0;
            try {
                time = new
CalculateDistanceTask().walkingDistanceInMins(startLat, startLong, sightLat,
sightLong);
            } catch (Exception e) {
                e.printStackTrace();
            }
            time = time +
sightsCursor.getInt(sightsCursor.getColumnIndex(SightsEntry.COLUMN_TOUR_TIME
));

            ContentValues timeValues = new ContentValues();
            timeValues.put(SightsEntry.COLUMN_DIST_TIME, time);
            rowsUpdated = rowsUpdated +
getContentResolver().update(sightUri, timeValues, null, null);

        } while (sightsCursor.moveToNext());
        sightsCursor.close();
    }
}


//Create the results in background
@Override
protected Boolean doInBackground(String... params) {

    updateDistances();

    //Find the ids, longitudes and latitudes of the correct sights
    int freeTime = (freeTime_Crit + 1) * 60;  //freeTime_Crit: 1 hour(0), 2
hours(1), 3 hours(2)
    while (freeTime > 0) {
        StringBuilder selection = new
StringBuilder(SightsEntry.COLUMN_DIST_TIME + " >= 0 AND "
                + SightsEntry.COLUMN_DIST_TIME + " <=" +
String.valueOf(freeTime) + " AND "
                + SightsEntry.COLUMN_PRICE + " <=" + String.valueOf(budget)
+ " AND ("
                + SightsEntry.COLUMN_CATEGORY + "=? OR " +
SightsEntry.COLUMN_CATEGORY + "=? OR "
                + SightsEntry.COLUMN_CATEGORY + "=? OR " +
SightsEntry.COLUMN_CATEGORY + "=?) AND "
                + SightsEntry.COLUMN_SEEN + "=" + SightsEntry.NO);
        ArrayList<String> selectionArgsList = new ArrayList<>();
        selectionArgsList.add("");
        selectionArgsList.add("");
        selectionArgsList.add("");
        selectionArgsList.add("");

        for (int i = 0; i < listOfIds.size(); i++) {
            selection.append(" AND " + SightsEntry._ID + " != ?");
            selectionArgsList.add(String.valueOf(listOfIds.get(i)));
        }

        String[] selectionArgs = selectionArgsList.toArray(new
String[selectionArgsList.size()]);

        //Museum(0), Attraction(1), Church(2), Other(3)
        if (museum == 1) {
            selectionArgs[0] = String.valueOf(SightsEntry.CATEGORY_MUSEUM);
        }
        if (attraction == 1) {
            selectionArgs[1] =
```

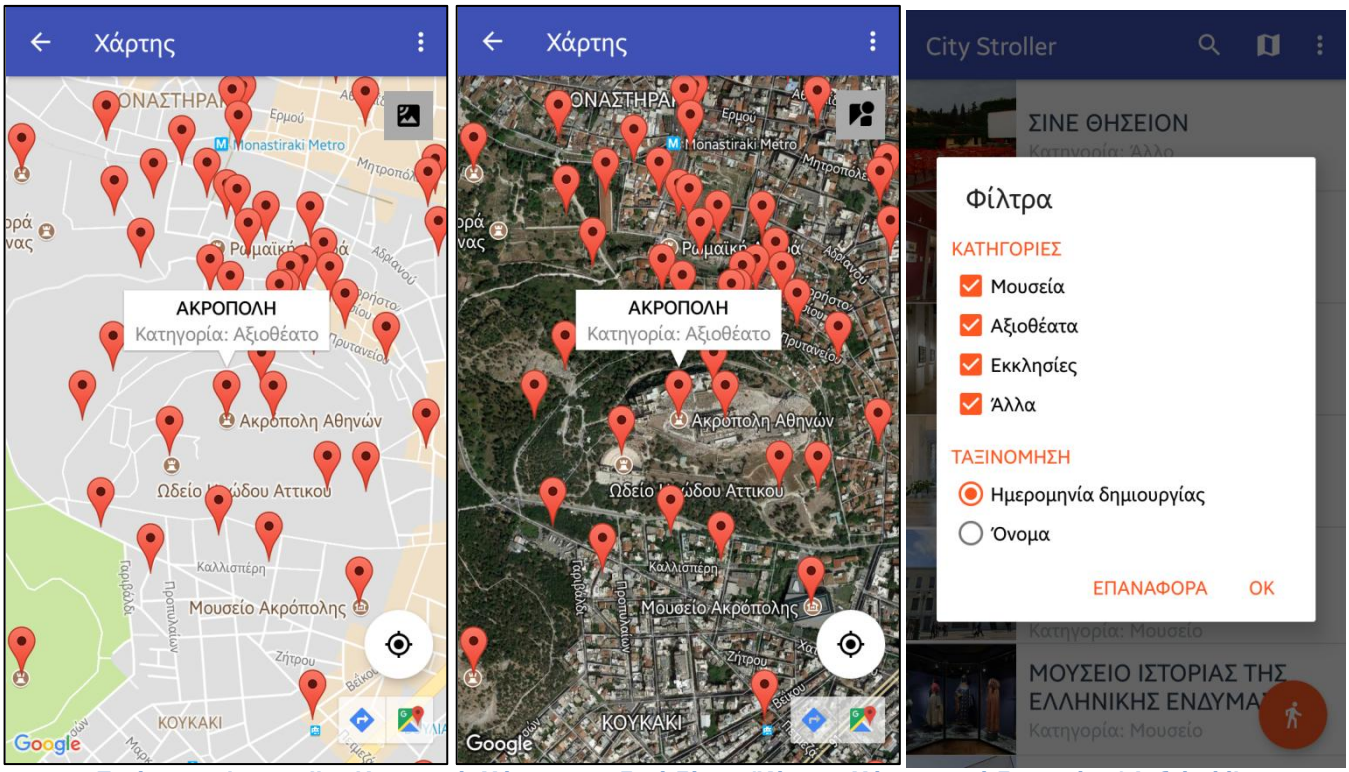
```
String.valueOf(SightsEntry.CATEGORY_ATTRACTION);
        }
        if (church == 1) {
            selectionArgs[2] = String.valueOf(SightsEntry.CATEGORY_CHURCH);
        }
        if (other == 1) {
            selectionArgs[3] = String.valueOf(SightsEntry.CATEGORY_OTHER);
        }

        String sortBy;
        if (popular == 1) {
            sortBy = SightsEntry.COLUMN_FAMOUS + " DESC, " +
SightsEntry.COLUMN_DIST_TIME + " ASC";
        } else {
            sortBy = SightsEntry.COLUMN_DIST_TIME + " ASC";
        }
        Cursor sightsCursor =
getContentResolver().query(SightsEntry.CONTENT_URI_SIGHTS, null,
selection.toString(), selectionArgs, sortBy);
        if (sightsCursor.getCount() > 0) {
            sightsCursor.moveToFirst();
            int id =
sightsCursor.getInt(sightsCursor.getColumnIndex(SightsEntry._ID));

            listOfIds.add(String.valueOf(id));

            startLong =
sightsCursor.getDouble(sightsCursor.getColumnIndex(SightsEntry.COLUMN_LONGIT
UDE));
            startLat =
sightsCursor.getDouble(sightsCursor.getColumnIndex(SightsEntry.COLUMN_LATITU
DE));

            long_points.add(startLong);
            lat_points.add(startLat);

            freeTime = freeTime -
sightsCursor.getInt(sightsCursor.getColumnIndex(SightsEntry.COLUMN_DIST_TIME
));
            budget = budget -
sightsCursor.getInt(sightsCursor.getColumnIndex(SightsEntry.COLUMN_PRICE));
            sightsCursor.close();

            //Update Dist_time column in Sights Table from the data of Times
Table
            Uri timesUri =
ContentUris.withAppendedId(SightsContract.TimesEntry.CONTENT_URI_TIMES, id);
            Cursor timesCursorForUpdate =
getContentResolver().query(timesUri, null, null, null, null);
            if (timesCursorForUpdate.getCount() > 0) {
                timesCursorForUpdate.moveToFirst();

                Cursor cursorForUpdate =
getContentResolver().query(SightsEntry.CONTENT_URI_SIGHTS, null, null, null,
null);
                if (cursorForUpdate.getCount() > 0) {
                    cursorForUpdate.moveToFirst();
                    do {
                        int currentId =
cursorForUpdate.getInt(cursorForUpdate.getColumnIndex(SightsEntry._ID));
                        Uri sightUri =
ContentUris.withAppendedId(SightsEntry.CONTENT_URI_SIGHTS, currentId);

                        int time =
timesCursorForUpdate.getInt(timesCursorForUpdate.getColumnIndex("T" +
currentId));
                        time = time +
cursorForUpdate.getInt(cursorForUpdate.getColumnIndex(SightsEntry.COLUMN_TOU
R_TIME));

                        ContentValues timeValues = new ContentValues();
```

```
                        timeValues.put(SightsEntry.COLUMN_DIST_TIME, time);
                        getContentResolver().update(sightUri, timeValues,
null, null);

                    } while (cursorForUpdate.moveToNext());
                    cursorForUpdate.close();
                }
                timesCursorForUpdate.close();
            }
        } else {
            freeTime = 0;
        }
    }
    return null;
}
```

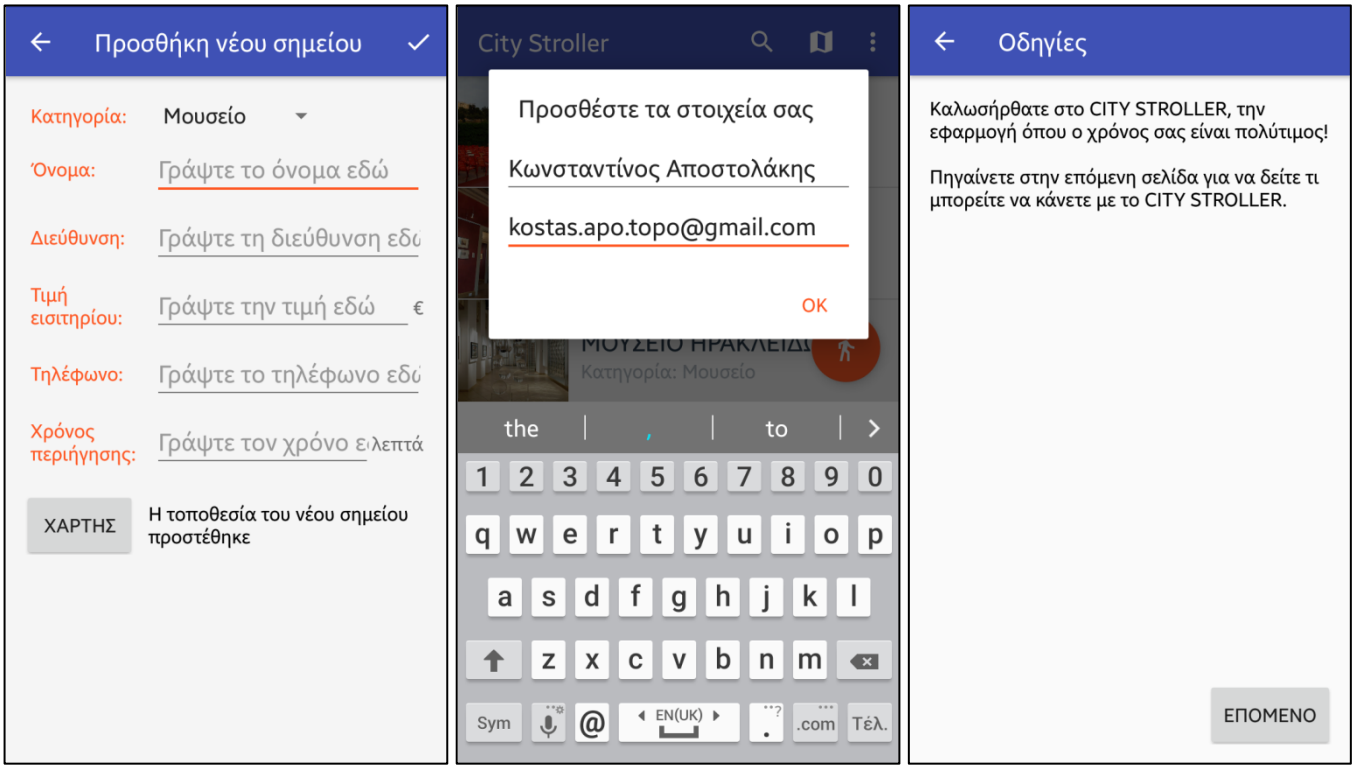
Εικόνες από την λειτουργία της εφαρμογής



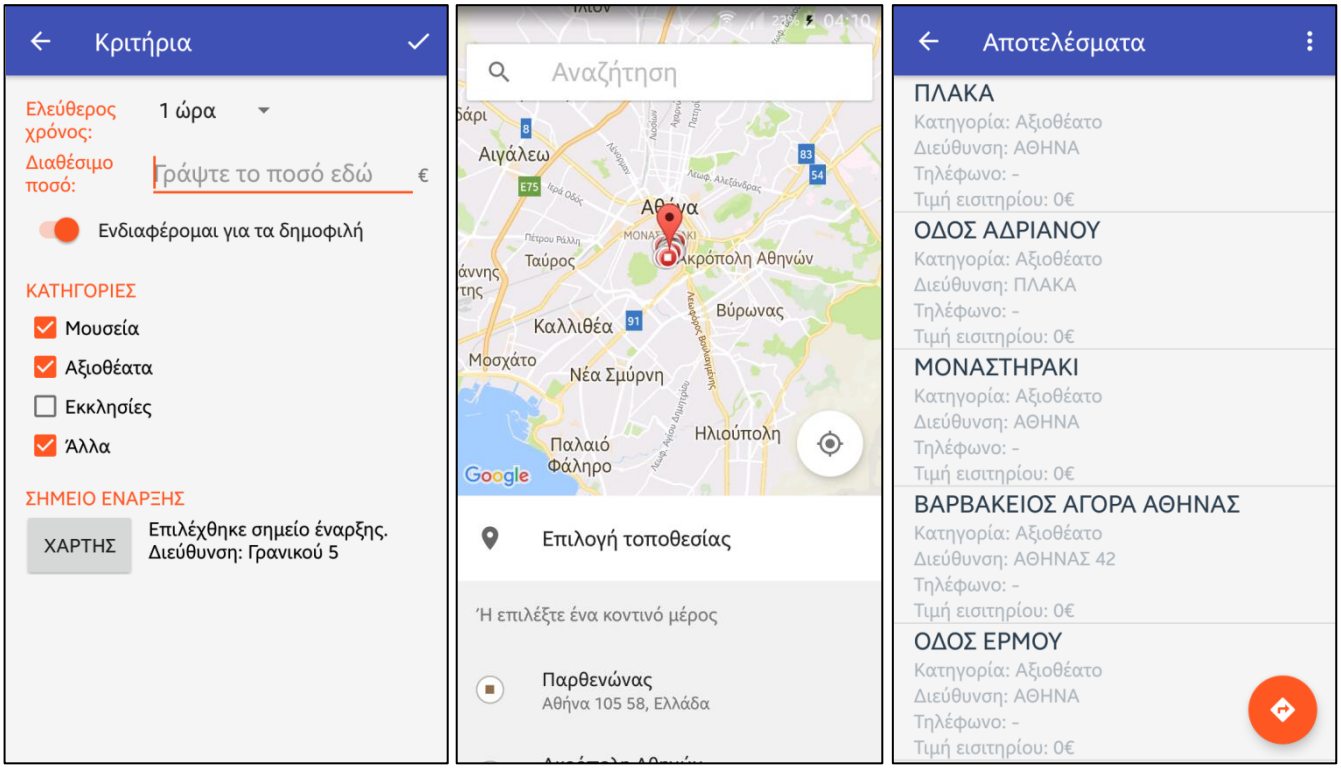
Εικόνα 6: City Stroller (Αριστερά: Αρχική λίστα/Κέντρο: Πληροφορίες σημείων/ Δεξιά: Αξιολόγηση σημείου)



Εικόνα 7: City Stroller (Αριστερά: Χάρτης με οδικό δίκτυο/Κέντρο: Χάρτης από δορυφόρο/ Δεξιά: Φίλτρα Λίστας)



Εικόνα 8: Αριστερά: Προσθήκη νέου σημείου/ Κέντρο: Μήνυμα εισαγωγής στοιχείων πριν την αποστολή των δεδομένων/ Δεξιά: Οδηγίες προς χρήστη



Εικόνα 9: Αριστερά: Κριτήρια για την δημιουργία διαδρομής/ Κέντρο: Επιλογή σημείου έναρξης και τοποθεσίας νέου σημείου σε χάρτη/ Δεξιά: Η λίστα με τα σημεία της προτεινόμενης διαδρομής