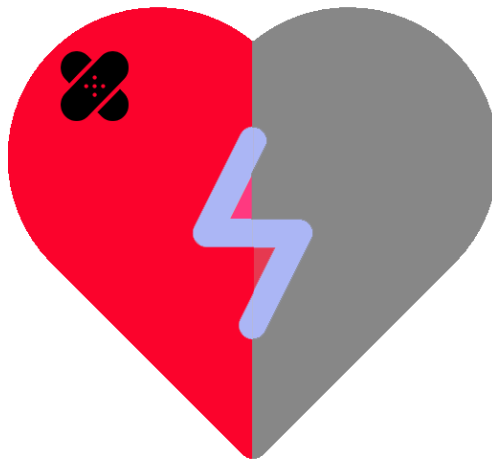


ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

Τμήμα πληροφορικής

*Ανάπτυξη αλγορίθμου και λογισμικού για βέλτιστο
χωρικό εντοπισμό*



Φαρμάκης Νίκος, Κέρκυρα Οκτώβριος 2020

Περιεχόμενα

Εισαγωγή	5
Παράρτημα σχημάτων.....	3
1. Η εξέλιξη του χωρικού εντοπισμού από την αρχαιότητα έως σήμερα.	6
2. Βέλτιστη Διαδρομή	10
2.1. Γράφοι	10
2.2. Το πρόβλημα του περιπλανώμενου πωλητή.....	14
2.3. Αλγόριθμοι συντομότερης διαδρομής.....	15
2.3.1. Ο αλγόριθμος Dijkstra.....	15
2.3.2. Ο αλγόριθμος Greedy Best First	15
2.3.3. Αλγόριθμος του Bellman Ford	16
2.3.4. Ο αλγόριθμος A* (Α άστρο).....	16
3. Ανάπτυξη εφαρμογής για τον εντοπισμό της βέλτιστης διαδρομής μεταξύ διαθέσιμων απινιδωτών	17
3.1. Επιλογή πλατφόρμας.....	17
3.2. Επιλογή minimum sdk.....	18
3.3. Επιλογή γλώσσας προγραμματισμού	19
3.4. Επιλογή χαρτογραφικού υποβάθρου.....	20
3.5. Επιλογή των κατάλληλων APIS	20
3.6. Δημιουργία εικονιδίων	21
3.7. Δεδομένα γεωγραφικής θέσης απινιδωτών	22
3.8. Λειτουργία της εφαρμογής	22
3.9. Google cloud console και δημιουργία API keys.....	25
3.10. Εισαγωγή των κλειδιών στην εφαρμογή.....	27
3.11. Άδειες εφαρμογής (permissions)	29

3.12.	Προσδιορισμός συνδεσιμότητας στο διαδίκτυο	32
3.13.	Προσδιορισμός τοποθεσίας χρήστη	33
3.14.	Προετοιμασία και εμφάνιση χάρτη	36
3.15.	Προσδιορισμός της καλύτερης επιλογής απινιδωτή.....	39
3.16.	Δημιουργία οθόνης πληροφοριών για την εφαρμογή και επικοινωνίας με τον προγραμματιστή	46
3.17.	Παρουσία στο Play store.....	50
3.18.	Βελτίωση και εξέλιξη εφαρμογής	51
4.	Συμπεράσματα	52
	Βιβλιογραφία	54

Παράρτημα σχημάτων

1.1. Εντοπισμός θέσης με χρήση τριών δορυφόρων. Πηγή: Nasa.	7
1.2: Google maps, Κέρκυρα. (Πηγή: Google maps)	9
2.1: Μη κατευθυνόμενος Γράφος. (Πηγή: Στεφανάκης, 2003)	11
2.2: Κατευθυνόμενος Γράφος (Πηγή: Στεφανάκης 2003).....	11
2.3: Γράφος με βάρη (πηγή: Στεφανάκης, 2003).....	11
2.4: Αναπαράσταση με Γράφο του προβλήματος του περιπλανώμενου πωλητή (πηγή: Στεφανάκης, 2003).....	14
3.1: Μερίδιο αγοράς λειτουργικών συστημάτων (Αύγουστος 2020) (Πηγή δεδομένων:netmarketshare.com)	18
3.2: Ενδεικτική οθόνη από android studio, ποσοστό πρόσβασης σε χρήστες με βάση το minimum SDK.....	19
3.3: εικονίδιο (marker απινιδωτών) και βασικό εικονίδιο της εφαρμογής	21
3.4:Κουμπί εύρεσης καλύτερης επιλογής.....	21
3.5: Ενδεικτικές εικόνες λειτουργίας της εφαρμογής (καλύτερη επιλογή)	23
3.6:Ενδεικτικές εικόνες λειτουργίας της εφαρμογής (επιλογή απινιδωτή από τον χρήστη) ...	23
3.7:Ενδεικτικές εικόνες λειτουργίας εφαρμογής (Σχετικά με την εφαρμογή).....	24
3.8: Επιλογή και ενεργοποίηση API απο google cloud console	25
3.9: Εμφάνιση των API keys που υπάρχουν στο συγκεκριμένο project	26
3.10: Δημιουργία signed bundle ή APK και του αντίστοιχου κλειδιού	27
3.11: δημιουργία νέου project και επιλογή Google Maps Activity.....	28
3.12: Εύρεση google_maps_api.xml (release)	29
3.13: Δήλωση αδειών στο manifest.xml.....	30
3.14: Αίτηση άδειας χρήσης τοποθεσίας κατα τη χρήση της εφαρμογής.....	30
3.15: Εμφάνιση μηνυμάτων μη εύρεσης δικτύου και απουσίας άδειας τοποθεσίας	31
3.16: Κώδικας function isOnline()	32
3.17: Function getLastLocation().....	33
3.18: Function getNewLocation()	34
3.19: locationCallback.....	35
3.20: Παύση ανανέωσης τοποθεσίας όταν το activity είναι στο παρασκήνιο	35
3.21: Επανεναρξη της ανανέωσης της τοποθεσίας όταν το activity επανέλθει στο προσκήνιο	35
3.22: Function onMapReady	37
3.23: press.xml (Διάδραση με τον χρήστη)	38
3.24: xml κώδικας του κύριου activity.....	39

3.25: Επιστροφή JSON του Distance Matrix Api	41
3.26: Κατασκευή του URL για την κλήση του Distance Matrix Api.....	42
3.27: Κώδικας στο κουμπί εύρεσης καλύτερης επιλογής.....	43
3.28: Κλάση FindBestOption.....	45
3.29: Αρχείο xml του μενού της εφαρμογής.....	47
3.30: Κουμπί μενου που ανοίγει την οθόνη πληροφοριών	47
3.31: Εισαγωγή του μενού στην ActionBar.....	47
3.32: Διαχείριση των εικονιδίων του μενού	47
3.33: Κώδικας xml αρχείου menu.....	48
3.34: xml κώδικας του activity πληροφοριών	49
3.35: Ενημέρωση για σφάλμα εφαρμογής απο το play console	50

Εισαγωγή

Η παρούσα εργασία αφορά την ανάπτυξη εφαρμογής που έχει ως στόχο την εύρεση της καλύτερης επιλογής από ένα σύνολο απινιδωτών στην πόλη της Κέρκυρας. Η εργασία προσπαθεί να απαντήσει με σύγχρονες μεθόδους στο πρόβλημα της βέλτιστης διαδρομής. Αρχικά στο πρώτο κεφάλαιο γίνεται μια ιστορική αναδρομή σχετικά με την έννοια του χωρικού εντοπισμού και την εξέλιξη της τεχνολογίας. Στο δεύτερο κεφάλαιο αναλύεται το πρόβλημα της βέλτιστης διαδρομής μέσω της θεωρίας των γράφων, γίνεται αναφορά στο πρόβλημα του πλανόδιου πωλητή και παρουσιάζονται οι σημαντικότεροι αλγόριθμοι επίλυσης του προβλήματος της βέλτιστης διαδρομής. Στο τρίτο κεφάλαιο αναλύονται τα βήματα δημιουργίας της εφαρμογής και οι τεχνικές που χρησιμοποιήθηκαν, γίνεται εκτενής αναφορά στα εργαλεία υλοποίησης και παρατίθεται και ο σχετικός κώδικας, ενώ στο τέλος του κεφαλαίου παρουσιάζονται οι προτάσεις βελτίωσης της εφαρμογής. Τέλος στο τέταρτο κεφάλαιο παρουσιάζονται τα συμπεράσματα από την παρούσα εργασία.

1. Η εξέλιξη του χωρικού εντοπισμού από την αρχαιότητα έως σήμερα.

Η έννοια του χωρικού εντοπισμού απασχολούσε την ανθρωπότητα από την αρχαιότητα. Για πολλά χρόνια ο εντοπισμός της θέσης μιας οντότητας γινόταν με την παρατήρηση του ήλιου, των αστεριών και αναγνωρίσιμων σταθερών σημείων όπως μνημεία, γέφυρες κ.α. ορόσημα. Οι παραπάνω τεχνικές ήταν προβληματικές και δεν λειτουργούσαν σωστά σε όλες τις συνθήκες, καθώς στον συννεφιασμένο ουρανό ήταν αδύνατο πολλές φορές ο εντοπισμός του ήλιου και των αστεριών.¹

Ο χωρικός εντοπισμός ήταν άμεσα συνδεδεμένος με την κατανόηση της γήινης γεωμετρίας. Μια από τις πιο ενδιαφέρουσες επιστήμες στην ελληνική εποχή ήταν η επιστήμη της γεωδαισίας. Ο Θαλής ο Μιλήσιος έφερε στην Ελλάδα την γεωμετρία από την Αίγυπτο και θεωρούσε ότι η γη ήταν ένα σώμα σαν δίσκος, που έπλεε σε έναν άπειρο ωκεανό. Η έννοια της σφαιρικότητας της γης οφείλεται στη σχολή του Πυθαγόρα, ενώ σημαντικός σταθμός για την κατανόηση της γεωμετρίας του πλανήτη ήταν ο προσδιορισμός της περιμέτρου της γης από τον Ερατοσθένη, ο οποίος πίστευε ότι όλη οι ωκεανοί συνδέονται μεταξύ τους. Η σημαντικότερη προσπάθεια αποτύπωσης της γήινης επιφάνειας έγινε από τον Πτολεμαίο και ο χάρτης που κατασκεύασε ο ίδιος χρησιμοποιούταν για πάνω από 14 αιώνες.²

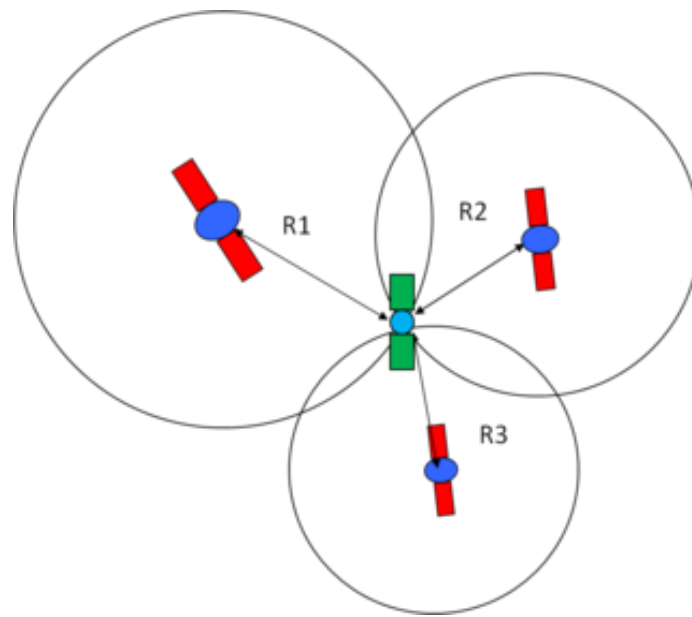
Με την εξέλιξη της επιστήμης και της τεχνολογίας όσο αφορά το ηλεκτρομαγνητικό πεδίο, δημιουργήθηκε το ραντάρ. Η βασική ιδέα της λειτουργίας του οφείλεται στα πειράματα του Γερμανού φυσικού Heinrich Hertz με την ηλεκτρομαγνητική ακτινοβολία στο τέλος της δεκαετίας του 1890, ο οποίος βασίστηκε στην έρευνα και τις εξισώσεις του Maxwell για το ηλεκτρομαγνητικό πεδίο. Η ανάπτυξη του ραντάρ ξεκίνησε τη δεκαετία του 1930 και κατά τη διάρκεια του Β' παγκοσμίου πολέμου εξελίχθηκε χάρις την κόντρα των αντίπαλων δυνάμεων για καλύτερες τεχνολογίες εντοπισμού.³

¹ National Geographic. <https://www.nationalgeographic.org/encyclopedia/navigation/> (πρόσβαση 17/9/2020)

² Δούκας Ι. Δ, Μπαντέλας Α. Γ, Σαββαΐδης Π. Δ, Υφαντής Ι. Μ, (1999). Γεωδαισία, τόμος 1, δεύτερη έκδοση. (σελ. 5 – 10)

³ Britannica. <https://www.britannica.com/technology/radar/History-of-radar> (πρόσβαση 17/9/2020)

Η εξέλιξη της διαστημικής τεχνολογίας σε συνδυασμό με την τεχνολογική εξέλιξη του ραντάρ οδήγησαν στην ανάπτυξη του GPS (Global Positioning System). Με την εκτόξευση του Sputnik 1, τον πρώτο τεχνητό δορυφόρο από τη Σοβιετική ένωση το 1957 ξεκίνησε ένας αγώνας δρόμου μεταξύ Η.Π.Α και Σοβιετικής ένωσης για την κατάκτηση και αξιοποίηση του διαστήματος.⁴ Στα μέσα της δεκαετίας του 1960 το Αμερικανικό ναυτικό ξεκίνησε πειράματα με σκοπό να μπορεί να εντοπίζει μέσω 6 δορυφόρων πολικής τροχιάς ένα υποβρύχιο με πυρηνικούς πυραύλους. Το 1978 ξεκίνησε η εκτόξευση συνολικά 24 δορυφόρων (NAVSTAR) για τον σκοπό αυτό και το σύστημα τέθηκε σε πλήρη λειτουργία το 1993. Σήμερα το GPS προσφέρει 2 επίπεδα υπηρεσιών, το SPS (Standard Positioning Service) και είναι αυτό που χρησιμοποιούμε όλοι στην καθημερινότητά μας, και το PPS (Precise Positioning Service), το οποίο χρησιμοποιείται από τον στρατό των Η.Π.Α, σε συνεργασία με τις συμμαχικές τους δυνάμεις.⁵



1.1. Εντοπισμός θέσης με χρήση τριών δορυφόρων. Πηγή: Nasa.

Για τον εντοπισμό της θέσης μια οντότητας στον πλανήτη χρησιμοποιούμε γεωγραφικές συντεταγμένες. Το σύστημα γεωγραφικών συντεταγμένων, εντοπίζεται για πρώτη φορά στον Ήππαρχο τον Ρόδιο τον 2^ο αιώνα π.Χ. Το σύστημα γεωγραφικών

⁴ NASA History Division. <https://history.nasa.gov/sputnik.html>. (πρόσβαση 17/9/2020)

⁵ Nasa. https://www.nasa.gov/directorates/heo/scan/communications/policy/GPS_History.html. (πρόσβαση 17/9/2020)

συντεταγμένων δημιουργήθηκε για τον μονοσήμαντο προσδιορισμό της θέσης ενός στοιχείου στον πλανήτη. Ο προσδιορισμός της θέσης αποτελείται από το γεωγραφικό πλάτος (γωνιακή απόσταση από τον ισημερινό στην κατεύθυνση βορρά – νότου) και το γεωγραφικό μήκος (γωνιακή απόσταση από τον 1^ο μεσημβρινό στην κατεύθυνση ανατολή δύση). Όλα τα σημεία με το ίδιο γεωγραφικό πλάτος αποτελούν έναν παράλληλο, ενώ όλα τα σημεία με το ίδιο γεωγραφικό μήκος αποτελούν έναν μεσημβρινό. Ως πρώτος μεσημβρινός ορίζεται ο μεσημβρινός που περνάει από το αστεροσκοπείο του Γκρήνουιτς κοντά στο Λονδίνο όπως έγινε παγκόσμια αποδεκτό στο το 1884 στο διεθνές συνέδριο Μεσημβρινών στην Ουάσιγκτον.⁶

Για την ακριβέστερη απεικόνιση των γεωμετριών του πλανήτη πάνω στο επίπεδο (χάρτης) χρησιμοποιούμε τα προβολικά συστήματα αναφοράς. Αντιστοίχιση των σημείων της μίας επιφάνειας (γη) σε μία άλλη (χάρτης). Υπάρχουν δυο προσεγγίσεις στην παραπάνω διαδικασία, η πρώτη προσέγγιση είναι η γεωγραφική προσέγγιση με την παραδοχή ότι η γη είναι σφαιρική, και η δεύτερη προσέγγιση είναι οι γεωδαιτική, με την παραδοχή ότι η γη είναι ένα ελλειψοειδές. Οι γεωγραφικοί χάρτες είναι συνήθως χάρτες μικρής κλίμακας και απεικονίζουν μεγάλες επιφάνειες. Οι γεωδαιτικοί χάρτες είναι χάρτες μεγάλης κλίμακας και απεικονίζουν μικρότερες περιοχές εξυπηρετώντας τις χαρτογραφικές ανάγκες. Το πρόβλημα που προκύπτει και στις δύο περιπτώσεις είναι η παραμορφώσεις των γεωμετριών καθώς η σφαίρα και το ελλειψοειδές είναι μη αναπτυκτές επιφάνειες. Το κάθε σύστημα αναφοράς χρησιμοποιεί διαφορετικές τεχνικές ανάπτυξης της σφαίρας ή του ελλειψοειδούς. Κάθε σύστημα λοιπόν καθορίζεται από την τεχνική ανάπτυξης της επιφάνειας της γης στο επίπεδο. Οι γεωδαιτικοί χάρτες υπερτερούν σε σχέση με τους γεωγραφικούς λόγο της ανωτερότητας του ελλειψοειδούς σε σχέση με την σφαίρα. Η πρώτη προσπάθεια δημιουργίας ενός παγκόσμιου γεωδαιτικού συστήματος αναφοράς έγινε το 1967 από τη διεθνή ένωση γεωδαισίας και γεωφυσικής όπου προτάθηκε το Γεωδαιτικό ελλειψοειδές αναφοράς (GRS 67) και με διάφορες τροποποιήσεις εξελίχθηκε στο Παγκόσμιο Γεωδαιτικό σύστημα 1984 (WGS 84) το οποίο χρησιμοποιούμε μέχρι και σήμερα για τις μετρήσεις του δορυφορικού συστήματος εντοπισμού (GPS). Για την εξάλειψη των αποκλίσεων από την πραγματικότητα κάθε κράτος χρησιμοποιεί το δικό του γεωδαιτικό σύστημα αναφοράς που ικανοποιεί τις χαρτογραφικές ανάγκες του.

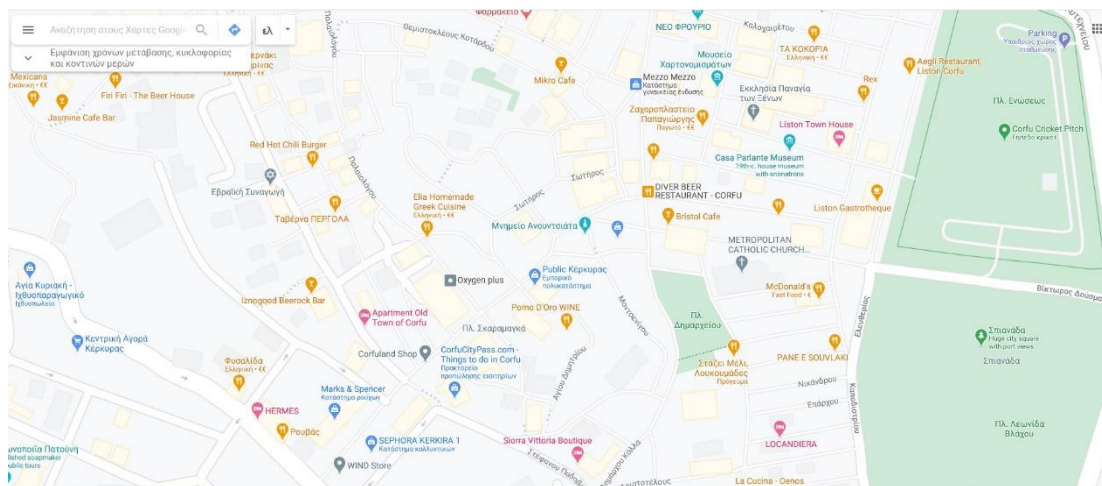
⁶ Robinson H, Arthur, Morrison L. Joel, Muehrcke C. Phillip, Kimerling A. Jon, Guptil C. Stephen, (2002). Στοιχεία χαρτογραφίας, Πανεπιστημιακές εκδόσεις Ε.Μ.Π, σελ. 57 - 58

Στην Ελλάδα χρησιμοποιείται το Ελληνικό Γεωδαιτικό Σύστημα Αναφοράς (ΕΓΣΑ 87).⁷

Στις μέρες μας έχουμε καταφέρει να χαρτογραφήσουμε το μεγαλύτερο μέρος του πλανήτη με ιδιαίτερη έμφαση στον αστικό ιστό. Με την εξέλιξη της τεχνολογίας έχουμε τη δυνατότητα πρόσβασης σε χάρτες μέσω των κινητών μας τηλεφώνων και άλλων φορητών συσκευών και μπορούμε να βρούμε και να πλοηγηθούμε σε όποιο μέρος θέλουμε στον πλανήτη με μια τυπική αναζήτηση. Μερικές από τις σημαντικότερες υπηρεσίες που προσφέρουν την παραπάνω δυνατότητα είναι:

- [Google maps](#)
- [Bing maps](#)
- [Open street map](#)

Οι παραπάνω υπηρεσίες παρέχονται δωρεάν στη βασική λειτουργία τους, δίνουν όμως τη δυνατότητα σε developers να ενσωματώσουν τις λειτουργίες αυτές σε δικιές τους εφαρμογές μέσω των APIS που παρέχουν με πληρωμή.⁸



1.2: Google maps, Κέρκυρα. (Πηγή: Google maps)

⁷ Δούκας Ι. Δ, Μπαντέλας Α. Γ, Σαββαΐδης Π. Δ, Υφαντής Ι. Μ, (1999). Γεωδαισία, τόμος 1, δεύτερη έκδοση. (σελ.324 – 327)

⁸ Groovypost. <https://www.groovypost.com/howto/seven-google-maps-alternatives-and-why-theyre-better/> (πρόσβαση 18/9/2020)

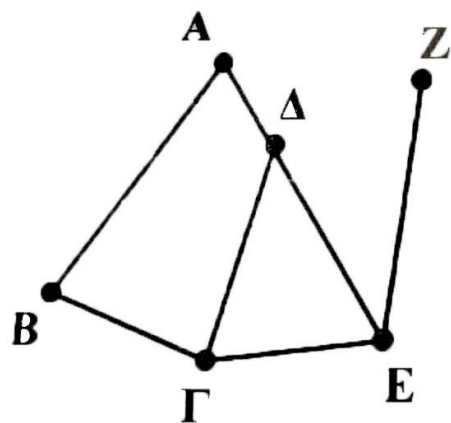
2. Βέλτιστη Διαδρομή

Με την πολυπλοκότητα του αστικού ιστού και του οδικού δικτύου δεν αρκεί να βρούμε μια διαδρομή προς ένα προορισμό, αλλά χρειαζόμαστε την βέλτιστη διαδρομή προς αυτόν. Στη σύγχρονη εποχή η έννοια της βέλτιστης διαδρομής δεν συμπίπτει πάντα και με την συντομότερη σε απόσταση διαδρομή, καθώς πχ η κίνηση στους δρόμους ή η εκτέλεση εργασιών σε κάποιον από αυτούς να κάνει την συγκεκριμένη διαδρομή εξαιρετικά χρονοβόρα. Ένα παράδειγμα εύρεσης της βέλτιστης διαδρομής είναι το πρόβλημα του περιπλανώμενου πωλητή. Για να μπορέσουμε να αναλύσουμε το συγκεκριμένο πρόβλημα πρέπει να κατανοήσουμε μερικές βασικές έννοιες που χρησιμοποιούνται για την λύση του προβλήματος.

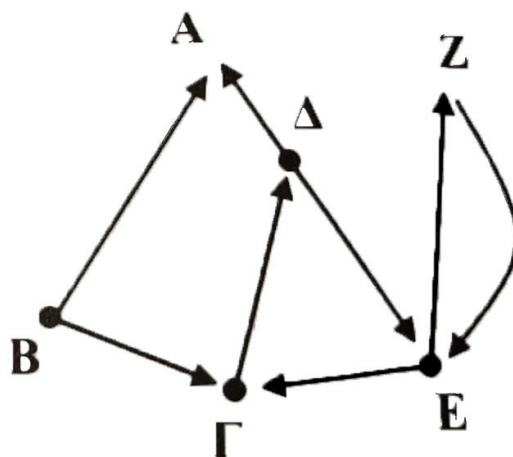
2.1. Γράφοι

Ένας Γράφος είναι ένα σύνολο κόμβων και ακμών όπου κάθε ακμή ορίζεται από δύο κόμβους στα άκρα αυτής. Οι Γράφοι χωρίζονται σε κατευθυνόμενοι, όταν στις ακμές τους ορίζονται κατευθύνσεις, και σε μη κατευθυνόμενοι όταν δεν ορίζονται κατευθύνσεις. Κάθε ακμή καταλήγει σε ένα κόμβο του γράφου, ενώ μια ακμή καλείται τυφλή όταν καταλήγει σε κόμβο ο οποίος δεν συνδέεται με άλλη ακμή. Δύο ακμές που καταλήγουν στον ίδιο κόμβο ονομάζονται γειτονικές, ενώ δυο κόμβοι ονομάζονται γειτονικοί όταν τους συνδέει άμεσα μία γραμμή. Στις ακμές ενός Γράφου μπορούν να ανατεθούν βάρη τα οποία μπορούν να αντιστοιχούν στο μήκος της ακμής, το κόστος διάσχισής της κ.α. Το οδικό δίκτυο μιας περιοχής μπορεί πολύ εύκολα να αναπαρασταθεί με ένα Γράφο όπου κάθε κόμβος αντιστοιχεί σε μια διασταύρωση του δικτύου ενώ κάθε ακμή σε έναν οδικό άξονα.⁹

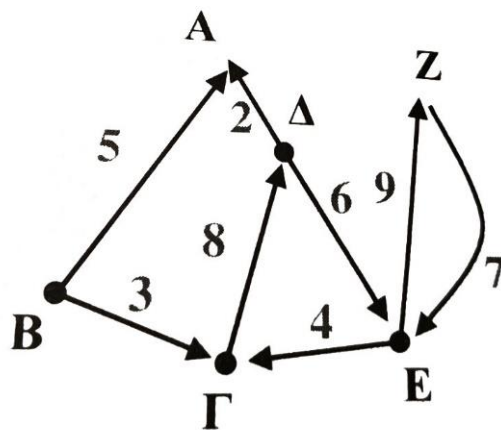
⁹ Στεφανάκης, Εμμανουήλ, (1999). Βάσεις Γεωγραφικών δεδομένων και συστήματα γεωγραφικών πληροφοριών, Εκδόσεις Παπασωτηρίου, (σελ. 195)



2.1: Μη κατευθυνόμενος Γράφος. (Πηγή: Στεφανάκης, 2003)



2.2: Κατευθυνόμενος Γράφος (Πηγή: Στεφανάκης 2003)



2.3: Γράφος με βάρη (πηγή: Στεφανάκης, 2003)

Ο μετασχηματισμός ενός δικτύου σε Γράφο μας βοηθάει να αναλύσουμε το δίκτυο με βάση συγκεκριμένες ιδιότητες ως προς τη δομή τους. Υπάρχουν 3 τύποι δικτύων με βάση τη συνδετικότητα τους.

- **Μη συνδεδεμένα δίκτυα.** Δίκτυα τα οποία έχουν κόμβους που δεν συνδέονται με κανέναν άλλον κόμβο στο δίκτυο.
- **Ελάχιστα συνδεδεμένα δίκτυα.** Κάθε κόμβος συνδέεται με τουλάχιστον άλλον έναν κόμβο και η μετακίνηση από έναν κόμβο σε έναν άλλον γίνεται μόνο από μία διαδρομή.
- **Μέγιστα συνδεδεμένα δίκτυα.** Υπάρχει μεγάλος αριθμός συνδέσεων σε σχέση με τους κόμβους του δικτύου, και η μετακίνηση από έναν κόμβο σε έναν άλλον μπορεί να γίνει με παραπάνω από μία διαδρομή.

Η συνδετικότητα ενός δικτύου περιγράφει τον βαθμό σύνδεσης μεταξύ των κόμβων του αυτού. Η θεωρία των Γράφων παρέχει δείκτες οι οποίοι μας βοηθούν να συγκρίνουμε τη δομή και την πολυπλοκότητα δύο ή περισσότερων δικτύων.

- **Δείκτης Γάμα.** Ο συγκεκριμένος δείκτης περιγράφει το βαθμό συνδετικότητας ενός δικτύου ως τον λόγο των συνδέσεων που υπάρχουν στο δίκτυο (σ) προς τον αριθμό των μέγιστων συνδέσεων που μπορούν να υπάρξουν σε αυτό ($\sigma_{\max}$).

$$\gamma = \frac{\sigma}{\sigma_{\max}}$$

Ο μέγιστος αριθμός συνδέσεων είναι πάντα $3(\kappa - 2)$ όπου κ ο αριθμός των κόμβων του δικτύου. Έτσι ο δείκτης γ με την αντικατάσταση του $\sigma_{\max}$, μπορεί να εκφραστεί ως εξής:

$$\gamma = \frac{\sigma}{3(\kappa - 2)}$$

Ο δείκτης γ παίρνει τιμές από 0 έως 1 και μπορεί να εκφραστεί και ως ποσοστό που περιγράφει τη συνδετικότητα ενός δικτύου. Παράδειγμα για την εικόνα 2.1 ο δείκτης γ είναι:

$$\gamma = \frac{7}{12} = 0,58$$

Μπορούμε λοιπόν να πούμε ότι το δίκτυο του σχήματος 2.1 είναι 58% συνδεδεμένο.

- **Δείκτης Άλφα.** Εκφράζει τον λόγο του αριθμού κυκλικών συνδέσεων που υπάρχουν σε ένα δίκτυο προς τον μέγιστο αριθμό κυκλικών συνδέσεων που μπορούν να υπάρξουν σε αυτό.

$$\alpha = \frac{\text{υπάρχοντες κυκλικές συνδέσεις}}{\text{μέγιστες κυκλικές συνδέσεις}}$$

Ο αριθμός των κυκλικών συνδέσεων σε ένα δίκτυο μπορεί να υπολογιστεί αν από τον συνολικό αριθμό συνδέσεων σε ένα δίκτυο αφαιρέσουμε τον αριθμό των συνδέσεων αν το συγκεκριμένο δίκτυο το θεωρήσουμε ως ελάχιστα συνδεδεμένο, όπου η μετακίνηση από τον έναν κόμβο στον άλλον γίνεται μόνο από μία διαδρομή. Ο αριθμός των συνολικών συνδέσεων σε ένα ελάχιστα συνδεδεμένο δίκτυο δίνεται από τη σχέση $\sigma = \kappa - 1$. Έτσι λοιπόν ο αριθμός κυκλικών συνδέσεων για ένα δίκτυο δίνεται από τη σχέση:

$$\text{Αριθμός κυκλικών συνδέσεων} = \sigma - (\kappa - 1) = \sigma - \kappa + 1$$

Με την ίδια λογική ακριβώς μπορούμε να υπολογίζουμε τον μέγιστο αριθμό κυκλικών συνδέσεων σε ένα δίκτυο αφαιρώντας των αριθμό των συνδέσεων του θεωρητικά ελάχιστα συνδεδεμένου δικτύου $(\kappa - 1)$ από των μέγιστο αριθμό συνδέσεων του δικτύου $3(\kappa - 2)$.

$$\text{Μεγ. αριθμός κυκλικών συνδέσεων} = 3(\kappa - 2) - (\kappa - 1) = 2\kappa - 5$$

Από τα παραπάνω προκύπτει:

$$\alpha = \frac{\sigma - \kappa + 1}{2\kappa - 5}$$

Για το δίκτυο της εικόνας 3 ο δείκτης α είναι:

$$\alpha = \frac{7 - 6 + 1}{2 * 6 - 5} = \frac{2}{7} = 0,28$$

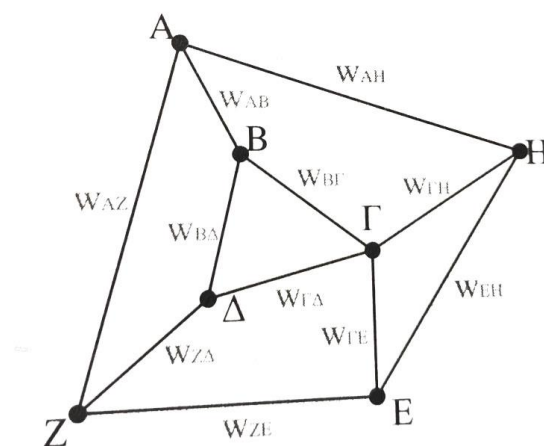
Μπορούμε λοιπόν να πούμε ότι το δίκτυο του σχήματος 2.1 έχει το 28% των κυκλικών συνδέσεων.

Η διάμετρος ενός δικτύου ορίζεται ως ο μέγιστος αριθμός συνδέσεων στην ελάχιστη από έναν κόμβο σε έναν άλλον. Στο δίκτυο πάλι του σχήματος 2.1 η διάμετρος του

δικτύου είναι 3 αφού η για τη μετακίνηση από τον κόμβο Α στον Κόμβο Ζ χρειάζονται 3 βήματα (μεσολαβούν 3 σύνδεσμοι).¹⁰

2.2. Το πρόβλημα του περιπλανώμενου πωλητή

Το συγκεκριμένο πρόβλημα αφορά την εύρεση της βέλτιστης διαδρομής που πρέπει να ακολουθήσει ένας πωλητής για να επισκεφτεί όλες τις πόλεις σε ένα δίκτυο πόλεων ώστε να ελαχιστοποιήσει το κόστος της διαδρομής. Το οδικό δίκτυο μαζί με τις πόλεις που συνδέει μπορεί να αναπαρασταθεί εύκολα με ένα Γράφο, και υπάρχουν αρκετοί αλγόριθμοι που δίνουν λύση στο συγκεκριμένο πρόβλημα.¹¹



2.4: Αναπαράσταση με Γράφο του προβλήματος του περιπλανώμενου πωλητή (πηγή: Στεφανάκης, 2003)

¹⁰ Κουτσόπουλος Κωστής, (2005), Γεωγραφικά συστήματα πληροφοριών και ανάλυση χώρου, εκδόσεις Παπασωτηρίου, (σελ 251 – 256)

¹¹ Στεφανάκης, Εμμανουήλ, (1999). Βάσεις Γεωγραφικών δεδομένων και συστήματα γεωγραφικών πληροφοριών, Εκδόσεις Παπασωτηρίου, (σελ. 205)

2.3. Αλγόριθμοι συντομότερης διαδρομής

Οι συγκεκριμένοι αλγόριθμοι υπολογίζουν όλες τις πιθανές διαδρομές μεταξύ αρχικού και τελικού κόμβου και προσπαθούν να βρουν τη διαδρομή με το μικρότερο κόστος. Παρακάτω θα αναλύσουμε μερικούς από αυτούς τους αλγόριθμους.

2.3.1. Ο αλγόριθμος Dijkstra

Ο αλγόριθμος Dijkstra προτάθηκε το 1959 από τον Edsger W. Dijkstra. Η εφαρμογή του γίνεται σε Γράφους με μη αρνητικά βάρη. Ο συγκεκριμένος αλγόριθμος αρχικά αναθέτει άπειρο κόστος μεταφοράς από τον κόμβο μετάβασης σε όλους τους υπόλοιπους κόμβους και στη συνέχεια δημιουργεί δύο σύνολα. Ένα σύνολο (ΚΕ) με τους κόμβους που έχουν επισκεφτεί κατά την επίλυση (αρχικά κενό σύνολο) και ένα (ΓΚΕ) με τους γειτονικούς κόμβους του πρώτου συνόλου. Σε κάθε βήμα ο αλγόριθμος ενημερώνει το κόστος μετάβασης προς τους κόμβους του συνόλου (ΓΚΕ) και προχωράει στον κόμβο με το μικρότερο κόστος. Η πολυπλοκότητα του αλγόριθμου Dijkstra καθιστά την εφαρμογή του αλγόριθμου ασύμφορη σε Γράφους με μεγάλο αριθμό κόμβων.¹²

2.3.2. Ο αλγόριθμος Greedy Best First

Παρόμοιος με τον αλγόριθμο Dijkstra με τη διαφορά ότι σε κάθε βήμα επιλέγει τον κόμβο που βρίσκεται πιο κοντά στον στόχο και όχι στην αφετηρία. Ο αλγόριθμος εκτελείται γρηγορότερα από τον Dijkstra, αλλά δεν βρίσκει πάντοτε την βέλτιστη διαδρομή.¹³

¹² Στεφανάκης, Εμμανουήλ, (1999). Βάσεις Γεωγραφικών δεδομένων και συστήματα γεωγραφικών πληροφοριών, Εκδόσεις Παπασωτηρίου, (σελ. 202)

¹³ Σπυρίδων Νικολαΐδης, (2005), Υλοποίηση αλγορίθμου εύρεσης βέλτιστης διαδρομής κινούμενης μονάδας, Αριστοτέλειο πανεπιστήμιο Θεσσαλονίκης, (σελ. 23)

2.3.3. Αλγόριθμος του Bellman Ford

Ο αλγόριθμος Bellman Ford μπορεί να τρέξει σε Γράφους με αρνητικά βάρη. Αν υπάρχει κύκλος αρνητικού βάρους προσβάσιμος από την αφετηρία ο αλγόριθμος ενημερώνει ότι δεν υπάρχει λύση, διαφορετικά ξεκινάει την κατασκευή του δέντρου συντομότερων μονοπατιών. Ο αλγόριθμος είναι πολύ πιο αργός σε σχέση με τον αλγόριθμο του Dijkstra καθώς κάθε ακμή εξετάζεται πάνω από μία φορά και το αποτέλεσμα της βέλτιστης διαδρομής από ένα κόμβο σε έναν άλλον δίνεται με τον τερματισμό του αλγορίθμου και όχι με τη σάρωση της διαδρομής από κόμβο σε κόμβο.¹⁴

2.3.4. Ο αλγόριθμος A* (Α άστρο)

Ο αλγόριθμος A* είναι ο δημοφιλέστερος αλγόριθμος βέλτιστης διαδρομής. Χρησιμοποιείται σε εξαιρετικά μεγάλους Γράφους που η εφαρμογή του αλγορίθμου Dijkstra είναι απαγορευτική. Αξιοποιεί τις τεχνικές της τεχνητής νοημοσύνης, προκειμένου να αυξηθεί η ταχύτητα αναζήτησης μέσα από την εφαρμογή ευρετικών κανόνων. Ο αλγόριθμος αρχικά κάνει μια εκτίμηση του κόστους μετάβασης από τον κόμβο αναφοράς στον κόμβο προορισμού και στη συνέχεια ακολουθεί τη λογική του Dijkstra για τον υπολογισμό του κόστους μετάβασης στους γειτονικούς κόμβους. Ο αλγόριθμος αποκλείει μονοπάτια που έχουν συσσωρευμένο κόστος μετάβασης μεγαλύτερο από αυτό που έχει εκτιμηθεί αρχικά για το σύνολο της διαδρομής.¹⁵

¹⁴ Χααραλαμπίδου Σοφία, (2005), Ανάπτυξη μοντέλων βελτιστοποίησης διαδρομής για route planners: η περίπτωση του i-Mobi Volos, Πανεπιστήμιο Θεσσαλίας Τμήμα Πολιτικών Μηχανικών (σελ 25)

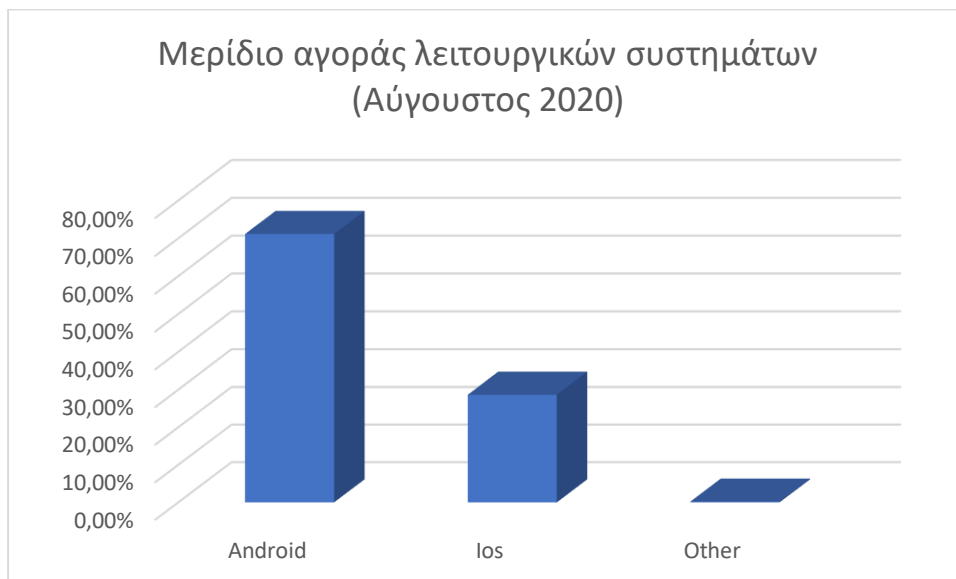
¹⁵ Στεφανάκης, Εμμανουήλ, (1999). Βάσεις Γεωγραφικών δεδομένων και συστήματα γεωγραφικών πληροφοριών, Εκδόσεις Παπασωτηρίου, (σελ. 204 - 205)

3. Ανάπτυξη εφαρμογής για τον εντοπισμό της βέλτιστης διαδρομής μεταξύ διαθέσιμων απινιδωτών

Δημιουργήθηκε εφαρμογή που βρίσκει την καλύτερη λύση μέσα από τις γνωστές θέσεις (συντεταγμένες) απινιδωτών που βρίσκονται στην πόλη της Κέρκυρας με βάση την τοποθεσία του χρήστη. Η γενική ιδέα της εφαρμογής είναι με το πάτημα ενός κουμπιού ο χρήστης να μπορεί να βρει τον κοντινότερο σε αυτόν απινιδωτή με βάση τον χρόνο που θα κάνει για να φτάσει σε αυτόν. Η εφαρμογή για την εύρεση του απινιδωτή λαμβάνει υπόψη της εκτός από την απόσταση, και την κίνηση στους δρόμους ώστε να μπορέσει να βρει την βέλτιστη λύση. Παρακάτω θα αναλυθούν τα στάδια υλοποίησης και οι τεχνικές που επιλέχθηκαν για την ολοκλήρωση τη εφαρμογής.

3.1. Επιλογή πλατφόρμας

Τα διαφορετικά λειτουργικά συστήματα που τρέχουν σε κινητές συσκευές αναγκάζουν τους developers να δημιουργήσουν αντίστοιχες εκδόσεις των εφαρμογών τους για καθένα από αυτά. Τα τελευταία χρόνια έχουν επικρατήσει στην αγορά το android της Google και το IOS της Apple. Η επιλογή της πλατφόρμας που θα ξεκινήσει η εφαρμογή εξαρτάται από το μερίδιο αγοράς που κατέχει το κάθε λειτουργικό, καθώς θέλουμε να απευθυνθούμε αρχικά σε όσους περισσότερους χρήστες μπορούμε. Στο παρακάτω διάγραμμα παρατηρούμε το μερίδιο αγοράς των λειτουργικών συστημάτων για τον Αύγουστο του 2020



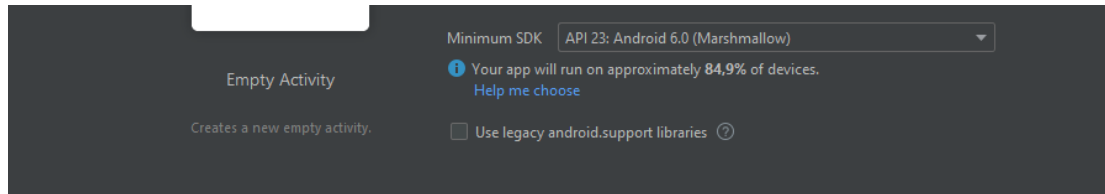
3.1: Μερίδιο αγοράς λειτουργικών συστημάτων (Αύγουστος 2020) (Πηγή δεδομένων: netmarketshare.com)

Παρατηρούμε λοιπόν ότι το Android κατέχει πάνω από το 70% της συνολικής αγοράς ενώ το λειτουργικό IOS λιγότερο από το 30%, ενώ ένα πολύ μικρό ποσοστό αφορά παλαιές εκδόσεις λειτουργικών συστημάτων όπως windows mobile, blackberry OS Symbian κ.α. που δεν υποστηρίζονται πια, και είναι εγκατεστημένα σε παλιές συσκευές. Με βάση λοιπόν τα παραπάνω είναι ξεκάθαρη η επιλογή του Android για την ανάπτυξη της εφαρμογής, αφού εξασφαλίζει πρόσβαση σε περισσότερους χρήστες σε σχέση με το IOS.

3.2. Επιλογή minimum sdk

Η επιλογή του minimum SDK (Software Development Kit) εξαρτάται αρχικά από τον αριθμό των χρηστών στους οποίους είναι διαθέσιμη η εφαρμογή, αλλά και από τις λειτουργίες και τις νέες λειτουργίες και δυνατότητες που προσφέρει το κάθε SDK. Βασικές αλλαγές έχουν γίνει στην έκδοση Marshmallow (SDK 23) κυρίως όσο αφορά τη διαχείριση αδειών. Οι άδειες της εφαρμογής πρέπει να ζητώνται πλέον από τον χρήστη κατά τη διάρκεια του τρεξίματος της εφαρμογής και όποτε αυτές χρειάζονται. Το 84,9% των συνολικών android συσκευών τρέχουν εκδόσεις ίσες ή μεγαλύτερες του marshmallow καθιστώντας τη συγκεκριμένη έκδοση ιδανική για την επιλογή ως

minimum SDK για την ανάπτυξη της εφαρμογής. Η εφαρμογή δεν θα τρέχει σε οποιαδήποτε μικρότερη έκδοση από το marshmallow.



3.2: Ενδεικτική οθόνη από android studio, ποσοστό πρόσβασης σε χρήστες με βάση το minimum SDK

3.3. Επιλογή γλώσσας προγραμματισμού

Όσο αφορά την ανάπτυξη εφαρμογών για Android υπάρχουν δύο επιλογές, Java και Kotlin. Η Java χρησιμοποιείται στις εφαρμογές από την αρχή της κυκλοφορίας του Android. Η ανάπτυξη της γλώσσας Kotlin ξεκίνησε το 2010 από την JetBrains, και η official έκδοση 1.0 δημοσιεύτηκε τον Φεβρουάριο του 2016. Το 2017 η Google ανακοίνωσε πλήρης υποστήριξη για την Kotlin και ενθαρρύνει τους χρήστες να τη χρησιμοποιούν. Και όλα τα παραδείγματα κώδικα στον ιστότοπο developer.android.com περιέχουν εκτός από Java και Kotlin. Μερικά βασικά χαρακτηριστικά της Kotlin είναι:

- Μειώνει αρκετά τον κώδικα
- Προστατεύει από το γνωστό πρόβλημα NullPointerException
- Είναι 100% λειτουργική μαζί με τη Java, και μπορεί να χρησιμοποιηθεί παράλληλα. Το android studio και πολλά άλλα IDE μετατρέπουν αυτόματα κώδικα java σε Kotlin.
- Οι διεργασίες ενημέρωσης βάσεων δεδομένων και οι κλήσεις δικτύου γίνονται πιο εύκολα και αποτελεσματικά.

Έτσι λοιπόν με βάση τα παραπάνω για την υλοποίηση της εφαρμογής επιλέχθηκε η Kotlin.

3.4. Επιλογή χαρτογραφικού υποβάθρου

Η εφαρμογή βασίζεται στους χάρτες της Google και στα APIS που παρέχει η εταιρία. Η επιλογή έγινε με βάση την ευκολότερη ανάπτυξη, καθώς υπάρχει μεγαλύτερη υποστήριξη από αρκετούς χρήστες που αναπτύσσουν εφαρμογές με αυτόν τον τρόπο. Επίσης υπάρχουν αναλυτικοί οδηγοί στο developers.google.com που καλύπτουν σχεδόν όλα τα σενάρια χρήσεις της υπηρεσίας google maps και των APIS. Η χρήση των υπηρεσιών είναι δωρεάν μέχρι έναν αριθμό κλήσεων για κάθε API και οι νέοι χρήστες ξεκινούν με 200\$ δωρεάν κλήσεις για τον πρώτο χρόνο. Η χρήση του Maps SDK for android και Maps SDK for IOS είναι δωρεάν ανεξαρτήτων κλήσεων του API, έτσι το κόστος της εφαρμογής εξαρτάται από τα υπόλοιπα API που θα χρησιμοποιηθούν και της κλήσεις σε αυτά ανά χρήστη.

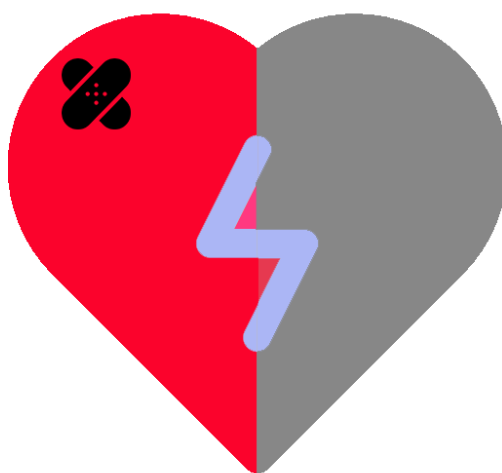
3.5. Επιλογή των κατάλληλων APIS

Ο στόχος της εφαρμογής είναι με το πάτημα ενός κουμπιού ο χρήστης να βρίσκει την καλύτερη επιλογή από το σύνολο των αντίδοτων που υπάρχουν στην περιοχή του. Η συγκεκριμένη λειτουργία επιτυγχάνεται με τη χρήση του Distance Matrix API. Το Distance Matrix API παρέχεται σε δύο εκδόσεις με διαφορετικό κόστος. Η έκδοση basic όπου κοστολογείται με 0.005 USD η κάθε κλήση (5.00 USD οι 10.000 κλήσεις) και η έκδοση advanced όπου κοστολογείται με 0.01 USD η κάθε κλήση (10.00 USD οι 10.000 κλήσεις) η διαφορά μεταξύ τους είναι ότι στην έκδοση advanced υπολογίζεται η κίνηση στους δρόμους και η απάντηση του API περιέχει και το πεδίο *duration_in_traffic*.

Η εφαρμογή χρησιμοποιεί την advanced έκδοση του Distance Matrix API, καθώς η εύρεση της διαδρομής με τον λιγότερο χρόνο είναι μείζον σημασίας σε περιπτώσεις έκτακτης ανάγκης και εξαρτάται από την κίνηση στους δρόμους.

3.6. Δημιουργία εικονιδίων

Για τις ανάγκες της εφαρμογής δημιουργήθηκαν δύο εικονίδια, ένα για τα markers των απινιδωτών το οποίο είναι και το βασικό εικονίδιο της εφαρμογής και ένα για το κουμπί της εύρεσης της καλύτερης επιλογής. Το κουμπί της εύρεσης καλύτερης επιλογής σχεδιάστηκε με τέτοιο τρόπο ώστε να υποδηλώνει ότι η πλοήγηση γίνεται με αυτοκίνητο και όχι με κάποιον άλλο τρόπο καθώς η επιλογή με αυτοκίνητο είναι προαπαιτούμενο για να βρεθεί ο χρόνος της βέλτιστης διαδρομής με βάση και την κίνηση στους δρόμους.



3.3: εικονίδιο (marker απινιδωτών) και βασικό εικονίδιο της εφαρμογής



3.4: Κουμπί εύρεσης καλύτερης επιλογής

3.7. Δεδομένα γεωγραφικής θέσης απινιδωτών

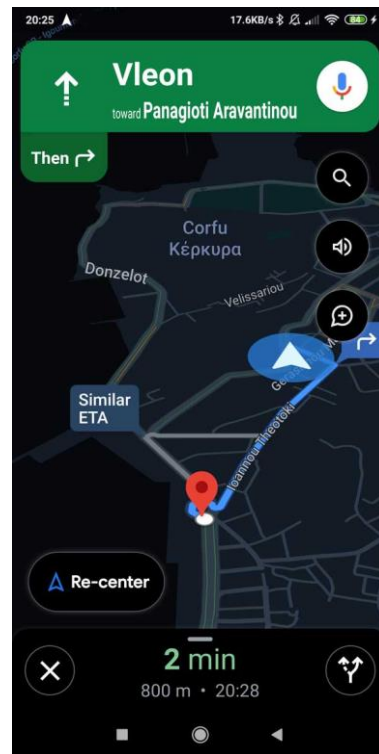
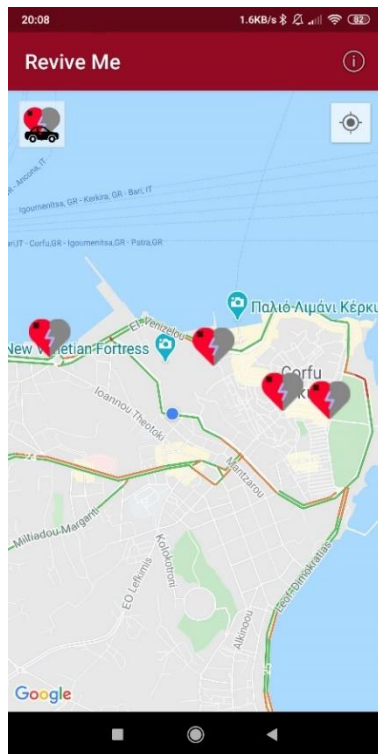
Οι θέσεις των απινιδωτών δόθηκαν από το ΕΚΑΒ Κέρκυρας σε μορφή συντεταγμένων google maps. Οι συντεταγμένες των συνολικά έξι απινιδωτών στην πόλη της Κέρκυρας είναι:

1. Sette venti café, *LatLng*(39.6258205, 19.9106273)
2. Εθνική Τράπεζα της Ελλάδος, *LatLng*(39.6238948, 19.9214520)
3. Κρατικός αερολιμένας Κέρκυρας, *LatLng*(39.6073261, 19.9140407)
4. Γήπεδο μπάσκετ, *LatLng*(39.6098500, 19.9150925)
5. Βίκτωρος Δούσμανη & Ελευθερίας, *LatLng*(39.6235919, 19.9236273)
6. Ναυτικός Σταθμός Κέρκυρας, *LatLng*(39.6255075, 19.9182310)

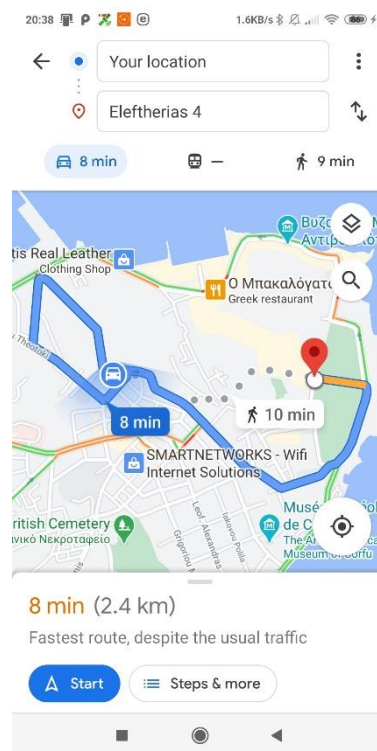
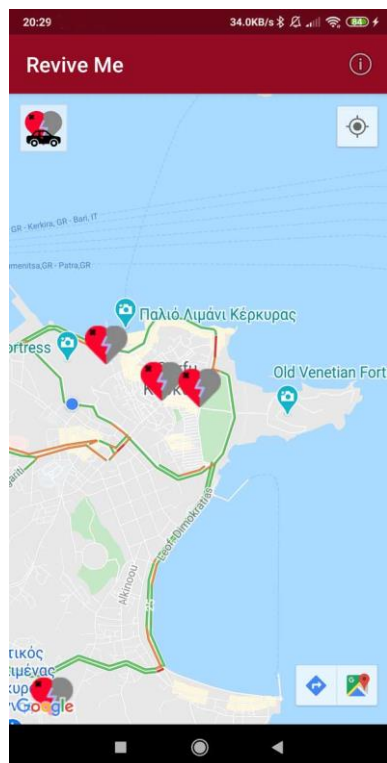
3.8. Λειτουργία της εφαρμογής

Η λειτουργία της εφαρμογής είναι εξαιρετικά απλή. Με το άνοιγμα της εφαρμογής ο χρήστης βλέπει την τοποθεσία του στο χάρτη και τις θέσεις των κοντινών στην τοποθεσία του απινιδωτών (εάν υπάρχουν). Στο πάνω μέρος της οθόνης υπάρχουν δυο κουμπιά αριστερά το κουμπί εύρεσης της καλύτερης επιλογής απινιδωτή και δεξιά το κουμπί εύρεσης της τοποθεσίας του χρηστη. Πατώντας το κουμπί εύρεσης καλύτερης επιλογής, η εφαρμογή υπολογίζει την καλύτερη επιλογή από τους διαθέσιμους απινιδωτές και ανοίγει το google maps, ώστε να ξεκινήσει η πλοήγηση προς τον απινιδωτή. Πατώντας το κουμπί εύρεσης τοποθεσίας ο χάρτης επανέρχεται στο σημείο της θέσης του χρήστη αν δεν είναι ήδη εκεί.

Ο χρήστης έχει την επιλογή να διαλέξει μόνος του τον απινιδωτή που επιθυμεί. Πατώντας πάνω σε κάποιον από τους διαθέσιμους απινιδωτές εμφανίζονται δύο κουμπιά κάτω δεξιά τα οποία τον οδηγούν στην εφαρμογή google maps για να επιλέξει να ξεκινήσει την πλοήγηση προς αυτόν επιλέγοντας και το στυλ πλοήγησης (αυτοκίνητο, πόδια μέσα μαζικής μεταφοράς).

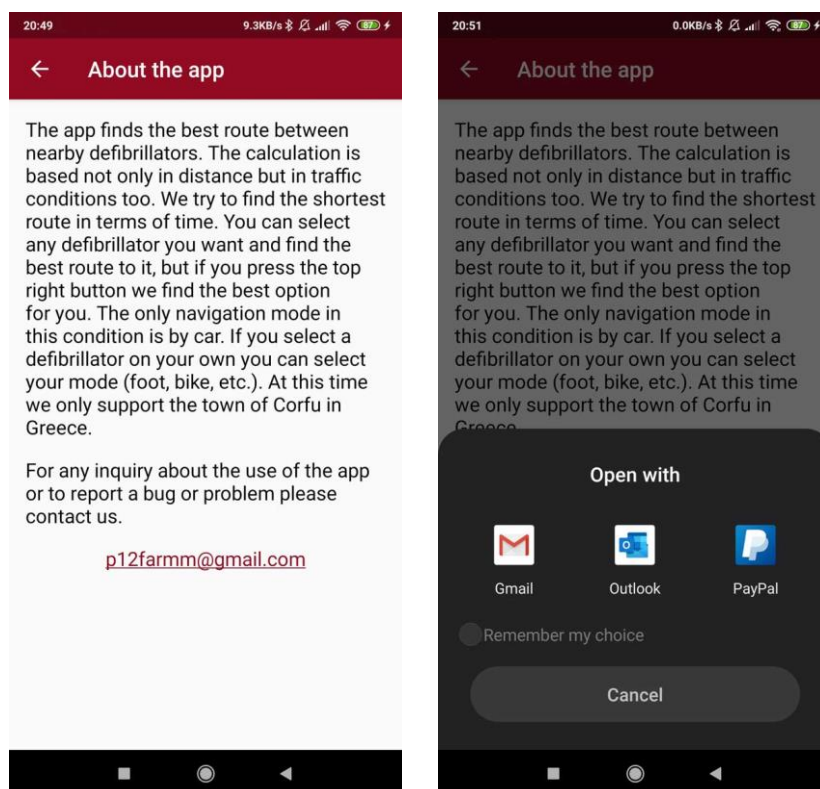


3.5: Ενδεικτικές εικόνες λειτουργίας της εφαρμογής (καλύτερη επιλογή)



3.6: Ενδεικτικές εικόνες λειτουργίας της εφαρμογής (επιλογή απινιδωτή από τον χρήστη)

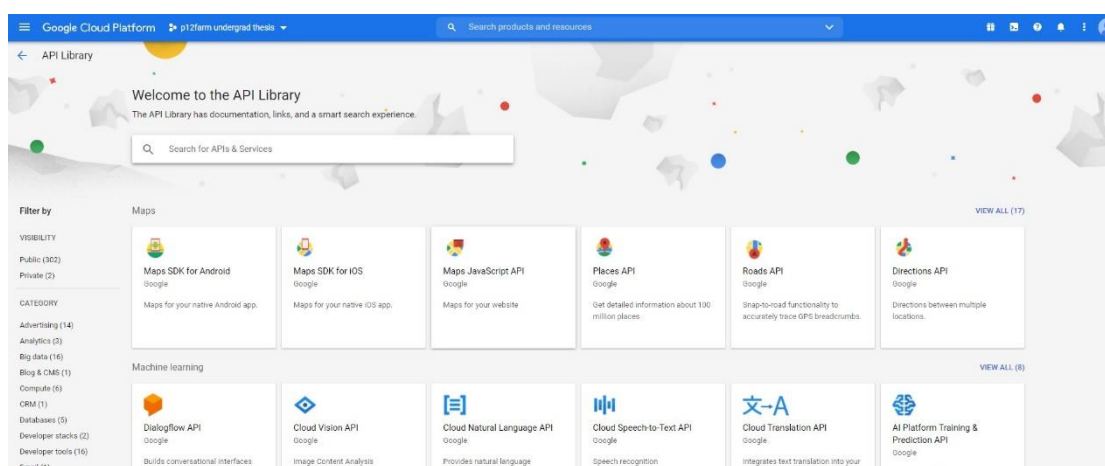
Η εφαρμογή διατίθεται σε δύο γλώσσες, Ελληνικά και Αγγλικά και η επιλογή της γλώσσας γίνεται με βάση την επιλεγμένη γλώσσα του λειτουργικού συστήματος. Πάνω δεξιά στην μπάρα της εφαρμογής (ActionBar) Υπάρχει κουμπί το οποίο ανοίγει νέα οθόνη με πληροφορίες για την εφαρμογή, ενώ υπάρχει και η επιλογή επικοινωνίας με τον προγραμματιστή (πατώντας πάνω στο mail) για οποιαδήποτε απορία ή αναφορά κάποιας δυσλειτουργίας bug.



3.7: Ενδεικτικές εικόνες λειτουργίας εφαρμογής (Σχετικά με την εφαρμογή)

3.9. Google cloud console και δημιουργία API keys

Με την είσοδο στο Google cloud console ο χρήστης καλείται να δημιουργήσει project. Αφού δημιουργηθεί και επιλεγεί το project, πρέπει να ενεργοποιηθούν τα APIS που μας χρειάζονται. Στην περίπτωση μας πρέπει να ενεργοποιηθεί το Maps SDK for android και το Distance Matrix API.



3.8: Επιλογή και ενεργοποίηση API απο google cloud console

Αφού επιλεγούν και ενεργοποιηθούν τα APIS πρέπει να δημιουργηθούν τα keys που θα χρησιμοποιηθούν από την εφαρμογή για την κλήση τους. Αυτό γίνεται με την επιλογή credentials από το μενού αριστερά.

The screenshot shows the Google Cloud Platform interface for the project 'p12farm undergrad thesis'. The left sidebar contains navigation links: Dashboard, Library, Credentials (selected), OAuth consent screen, Domain verification, and Page usage agreements. The main content area is titled 'Credentials' and includes a '+ CREATE CREDENTIALS' button and a 'DELETE' button. Below this, there is a warning message: 'Remember to configure the OAuth consent screen with information about your application.' The 'API Keys' section contains a table with the following data:

☐	Name	Creation date ↓	Restrictions
☐	✓ Maps api key (debug & release)	Sep 10, 2020	Android apps
☐	⚠ Distance matrix key	Sep 10, 2020	None

Below the API Keys section, there are two more sections: 'OAuth 2.0 Client IDs' and 'Service Accounts'. Both sections show a table with headers 'Name' and 'Creation date ↓', but they both indicate 'No OAuth clients to display' and 'No service accounts to display' respectively.

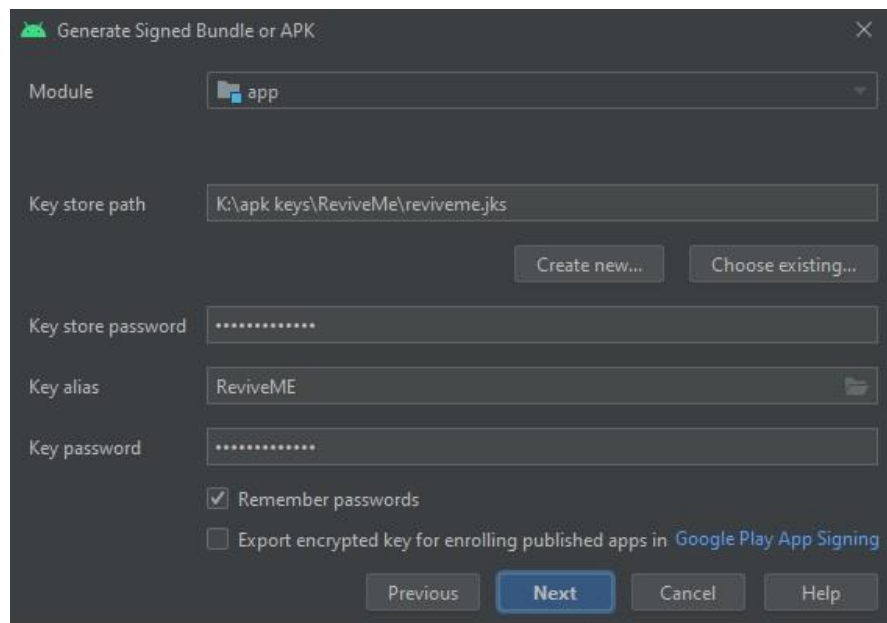
3.9: Εμφάνιση των API keys που υπάρχουν στο συγκεκριμένο project

Τα κλειδιά αφού δημιουργηθούν μπορούν να κλειδωθούν για να περιοριστεί η πρόσβαση του κάθε κλειδιού μόνο στη συγκεκριμένη εφαρμογή και το ίδιο κλειδί να μπορεί να καλέσει μόνο συγκεκριμένα API, ώστε σε περίπτωση που διαρρεύσει κάποιο κλειδί να μην μπορεί να χρησιμοποιηθεί από τρίτους.

Ενώ κανονικά η χρήση ενός και μόνο κλειδιού θα ήταν αρκετή για της ανάγκες της εφαρμογής, το κλείδωμα του κλειδιού για την συγκεκριμένη εφαρμογή, μπορεί να εφαρμοστεί μόνο στο Maps SDK for android και σε αυτή την περίπτωση η κλήση του Distance Matrix API από το ίδιο κλειδί δεν είναι δυνατή. Έτσι είναι απαραίτητη η δημιουργία δεύτερου κλειδιού για τις κλήσεις στο Distance Matrix API.

Το κλείδωμα επιτυγχάνεται μέσω του package name και του sha-1 certificate fingerprint τα οποία παρέχονται από το περιβάλλον του android studio. Το sha-1 certificate fingerprint είναι διαφορετικό για την έκδοση debug και release της εφαρμογής καθώς η εφαρμογή υπογράφεται ξανά με νέο πιστοποιητικό στην έκδοση release, επίσης κατά την δημοσίευση της εφαρμογής στο play store η google υπογράφει εκ νέου την εφαρμογή με βάση το ιδιωτικό κλειδί που δημιουργείται στην έκδοση release. Για να δουλεύει λοιπόν η εφαρμογή απροβλημάτιστα σε όλες τις περιπτώσεις το Maps SDK for android πρέπει να κλειδωθεί και με τα τρία παραπάνω sha-1 certificate fingerprint στο google cloud platform. Μια καλή πρακτική για τον

προσδιορισμό των κλήσεων και την εξαγωγή συμπερασμάτων των σφαλμάτων είναι να χρησιμοποιούνται διαφορετικά κλειδιά για κάθε έκδοση της εφαρμογής (debug - release – play store). Για να προσδιοριστεί το sha-1 certificate της έκδοσης release θα πρέπει να έχει πρώτα δημιουργηθεί signed bundle ή APK και να έχει δημιουργηθεί και αποθηκευτεί το απαραίτητο κλειδί.



3.10: Δημιουργία signed bundle ή APK και του αντίστοιχου κλειδιού

Η εύρεση του sha-1 certificate fingerprint για την έκδοση release γίνεται από το command prompt των windows με την εκτέλεση της παρακάτω εντολής:

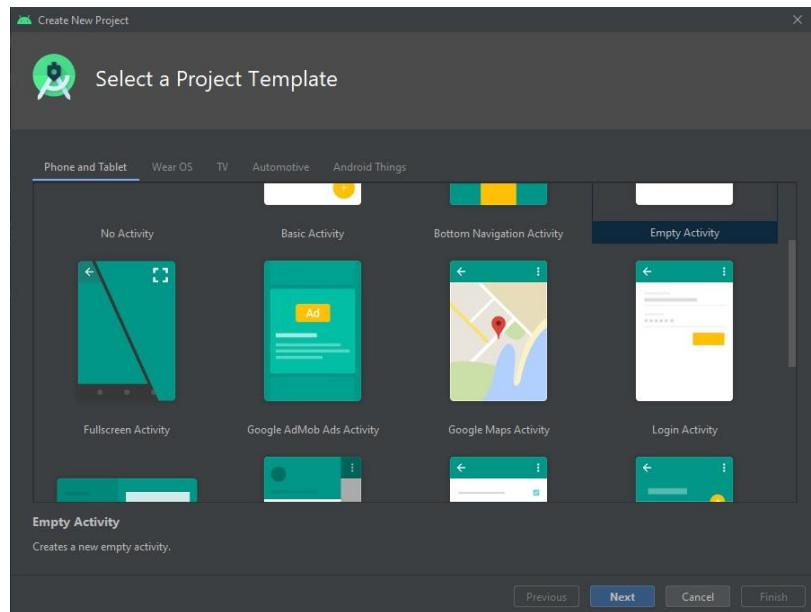
```
keytool -list -v -keystore {keystore_path} -alias {alias_name}
```

Η εντολή πρέπει να τρέξει από το path που είναι εγκατεστημένο το JDK

3.10. Εισαγωγή των κλειδιών στην εφαρμογή

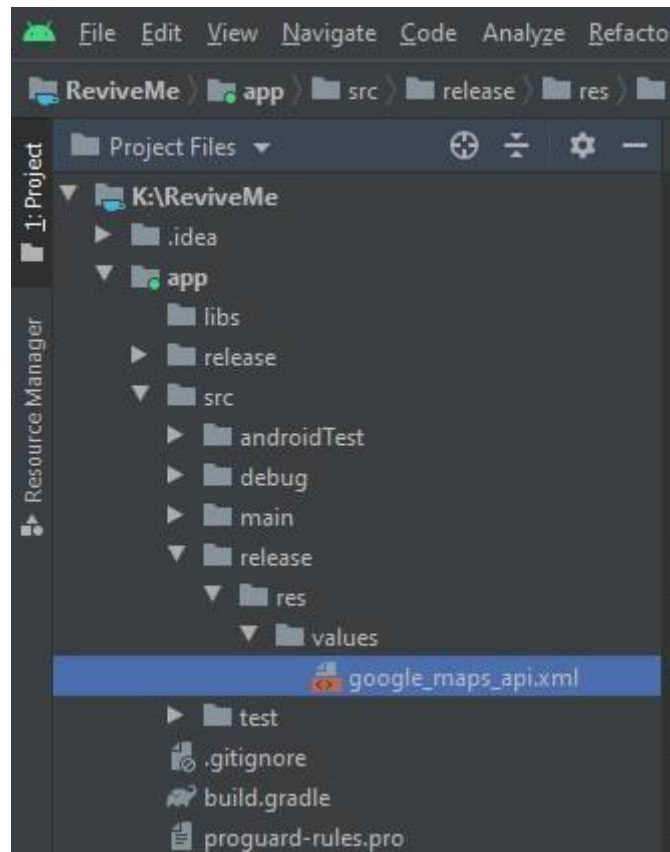
Κατά τη δημιουργία του project στο android studio με την επιλογή Google Maps Activity δημιουργούνται όλες οι απαραίτητες classes και string για την δημιουργία του χάρτη. Η λειτουργία αυτή κάνει τη δουλειά developer του αφ' ενός πολύ πιο εύκολη

καθώς εστιάζει την ενέργειά του στα βασικά σημεία της εφαρμογής και αφετέρου διατηρεί μια συνοχή στη διάταξη των αρχείων της εφαρμογής και βοηθάει στην υιοθέτηση καλύτερων πρακτικών.



3.11: δημιουργία νέου project και επιλογή Google Maps Activity

Για να εμφανιστεί ο χάρτης ο developer το μόνο που έχει να κάνει είναι να συμπληρώσει το API key στο αρχείο `google_maps_api.xml`. Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο ο developer πρέπει να συμπληρώσει το API key και για την έκδοση release της εφαρμογής στο αντίστοιχο `google_maps_api.xml`. Αυτό επιτυγχάνεται με την αλλαγή του tree view από android σε project από το android studio και το αρχείο βρίσκεται στο `/app/src/release/res/values/google_maps_api.xml`.



3.12: Εύρεση google_maps_api.xml (release)

3.11. Άδειες εφαρμογής (permissions)

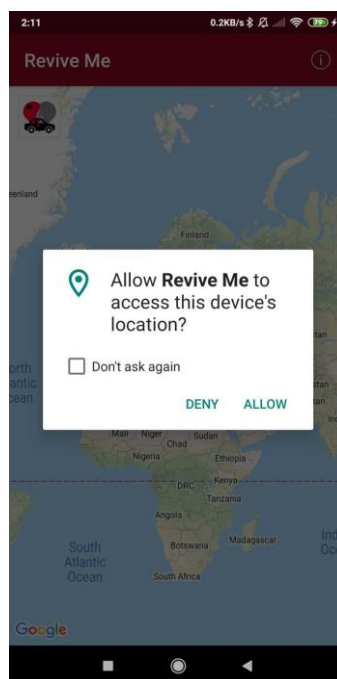
Για να λειτουργήσει σωστά η εφαρμογή, χρειάζεται πρόσβαση στα δεδομένα τοποθεσίας του χρήστη, πρόσβαση στο internet, καθώς και πρόσβαση στην κατάσταση του δικτύου, για να μπορούμε να ξέρουμε αν η συσκευή είναι συνδεδεμένη σε κάποιο δίκτυο ή όχι. Οι συγκεκριμένες άδειες πρέπει να δηλωθούν στο manifest.xml της εφαρμογής, ωστόσο στην περίπτωση της τοποθεσίας ο χρήστης θα πρέπει να δώσει έγκριση για χρήση της τοποθεσίας του κατά τη διάρκεια χρήσης της εφαρμογής.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="p12farm.reviveme">
    <!--
        The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
        Google Maps Android API v2, but you must specify either coarse or fine
        location permissions for the "MyLocation" functionality.
    -->
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />

```

3.13: Δήλωση αδειών στο manifest.xml



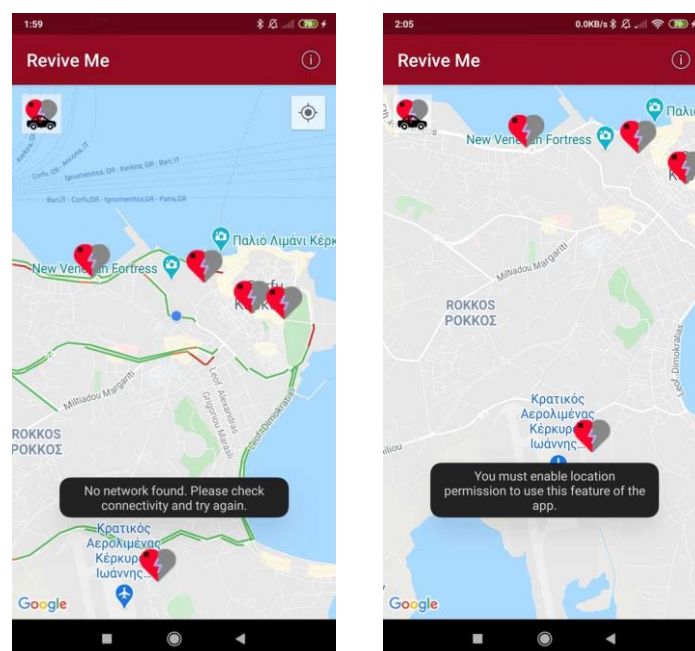
3.14: Αίτηση άδειας χρήσης τοποθεσίας κατά τη χρήση της εφαρμογής

Έτσι λοιπόν για την τοποθεσία χρειαζόμαστε το `ACCESS_COARSE_LOCATION`, και `ACCESS_FINE_LOCATION`. Η διαφορά μεταξύ των δύο είναι ότι το `ACCESS_COARSE_LOCATION` χρησιμοποιεί μόνο το δίκτυο για τον προσδιορισμό της τοποθεσίας, ενώ το `ACCESS_FINE_LOCATION`, χρησιμοποιεί το δίκτυο σε συνδυασμό με το GPS ή και ένα από αυτά τα δύο για τον προσδιορισμό της τοποθεσίας. Καταλαβαίνουμε λοιπόν ότι το `ACCESS_FINE_LOCATION`, μας δίνει μεγαλύτερη ακρίβεια από το `ACCESS_COARSE_LOCATION`, την οποία χρειαζόμαστε στην συγκεκριμένη εφαρμογή.

Για την πρόσβαση στο διαδίκτυο και την κατάσταση του δικτύου χρειαζόμαστε τις *ACCESS_NETWORK_STATE*, και *INTERNET*. Η εφαρμογή αρχικά χρειάζεται πρόσβαση στο διαδίκτυο για να φορτώσει τον χάρτη από την Google, και για να κάνει τις απαραίτητες κλήσεις στο Distance Matrix API για τον προσδιορισμό του κοντινότερου απινιδωτή.

Θέλουμε να ξέρουμε αν η συσκευή που τρέχει η εφαρμογή έχει πρόσβαση στο διαδίκτυο (*ACCESS_NETWORK_STATE*) ώστε να μπορούμε να ενημερώσουμε τον χρήστη να ενεργοποιήσει τα δεδομένα ή το WIFI σε περίπτωση που πατήσει το κουμπί εύρεσης καλύτερης επιλογής. Με αυτόν τον τρόπο γλυτώνουμε την αποτυχία κλήσης του Distance Matrix API και τον ξαφνικό τερματισμό της εφαρμογής.

Με την ίδια λογική αν ο χρήστης απενεργοποιήσει την άδεια της τοποθεσίας και στη συνέχεια πατήσει το κουμπί εύρεσης καλύτερης επιλογής τότε εμφανίζεται ανάλογο μήνυμα. Γλυτώνουμε και πάλι την αποτυχία κλήσης του Distance Matrix API λόγω μη προσδιορισμού τοποθεσίας του χρήστη.



3.15: Εμφάνιση μηνυμάτων μη εύρεσης δικτύου και απουσίας άδειας τοποθεσίας

3.12. Προσδιορισμός συνδεσιμότητας στο διαδίκτυο

Για τον προσδιορισμό της συνδεσιμότητας στο διαδίκτυο, έχει δημιουργηθεί function *isOnline* που επιστρέφει *true* αν υπάρχει σύνδεση με κάποιον από τους διαθέσιμους τρόπους, και *false* στην αντίθετη περίπτωση. Στην ουσία βλέπουμε αν ο χρήστης έχει ενεργοποιημένο το WIFI ή τα δεδομένα, υπάρχει όμως η περίπτωση να είναι ενεργοποιημένα και πάλι να μην είναι δυνατή η πρόσβαση στο διαδίκτυο (δεν υπάρχουν άλλα διαθέσιμα *mb* στο πρόγραμμα του χρήστη). Σε αυτή την περίπτωση η κλήση στο API θα αποτύχει αλλά μπορούμε να ενημερώσουμε τον χρήστη χωρίς να τερματιστεί η εφαρμογή. Ο τρόπος κλήσης του Distance Matrix API περιγράφεται σε επόμενο κεφάλαιο. Παρακάτω παρατίθεται ο κώδικας του function *isOnline*.

```
private fun isOnline(context: Context): Boolean {
    val connectivityManager =
        context.getSystemService(Context.CONNECTIVITY_SERVICE) as
ConnectivityManager
    val capabilities =
connectivityManager.getNetworkCapabilities(connectivityManager.activeNetwork)
    if (capabilities != null) {
        when {
            capabilities.hasTransport(NetworkCapabilities.TRANSPORT_CELLULAR)
-> {
                Log.i("Internet", "NetworkCapabilities.TRANSPORT_CELLULAR")
                return true
            }
            capabilities.hasTransport(NetworkCapabilities.TRANSPORT_WIFI) ->
{
                Log.i("Internet", "NetworkCapabilities.TRANSPORT_WIFI")
                return true
            }
            capabilities.hasTransport(NetworkCapabilities.TRANSPORT_ETHERNET)
-> {
                Log.i("Internet", "NetworkCapabilities.TRANSPORT_ETHERNET")
                return true
            }
        }
    }
    return false
}
```

3.16: Κώδικας function *isOnline()*

3.13. Προσδιορισμός τοποθεσίας χρήστη

Ο προσδιορισμός της τοποθεσίας του χρήστη γίνεται αρχικά με βάση την τελευταία γνωστή τοποθεσία που έχει καταγραφή στη συσκευή από κάποια εφαρμογή που χρησιμοποιεί την υπηρεσία εντοπισμού της google (google maps κ.α.). Αν η τοποθεσία αυτή είναι κενή τότε μπορούμε να ζητήσουμε την τοποθεσία real time από τη συσκευή σε διαστήματα που εμείς ορίζουμε. Και για τις δύο παραπάνω περιπτώσεις η εφαρμογή χρησιμοποιεί το Fused Location Provider API, το οποίο συνδυάζει δεδομένα τοποθεσίας από WIFI και GPS για βελτιστοποίηση της ακρίβειας σε συνδυασμό με χαμηλότερη κατανάλωση μπαταρίας στη συσκευή. Η διαδικασία γίνεται με δύο *functions*, *getLastLocation()* και *getNewLocation()*.

```
private fun getLastLocation() {
    if (checkPermission()) {
        if (isLocationEnabled()) {
            fusedLocationProviderClient.lastLocation.addOnCompleteListener
{ task->
                val location: Location? = task.result
                if (location == null) {
                    getNewLocation()
                } else {
                    //Setting the user's location in variable
                    userLocation = LatLng(location.latitude,
location.longitude)

                    //updating location on the map
                    val cameraPosition = CameraPosition.Builder()
                        .target(LatLng(location.latitude,
location.longitude))
                        .zoom(17f)
                        .build()

                    mMap.moveCamera(CameraUpdateFactory.newCameraPosition(cameraPosition))
                }
            }
        } else {
            Toast.makeText(
                this,
                getString(R.string.noLocationFound),
                Toast.LENGTH_LONG
            ).show()
        }
    } else {
        requestPermission()
    }
}
```

3.17: Function *getLastLocation()*

Το παραπάνω function αρχικά τσεκάρει αν ο χρήστης έχει δώσει στην εφαρμογή την απαραίτητη άδεια για χρήση της τοποθεσίας, αν δεν υπάρχει η απαραίτητη άδεια καλεί τη function `requestPermission()`, που ρωτάει τον χρήστη αν θέλει να εκχωρήσει το δικαίωμα στην εφαρμογή για χρήση της τοποθεσίας του. Αν η άδεια έχει δοθεί από τον χρήστη, τότε το function τσεκάρει αν είναι ενεργοποιημένο το GPS ή το δίκτυο στη συσκευή για τον προσδιορισμό της τοποθεσίας, και αν δεν είναι εμφανίζει μήνυμα ότι δεν βρέθηκε τοποθεσία. Στη διαφορετική περίπτωση αν υπάρχει τελευταία γνωστή τοποθεσία στον `fusedLocationProviderClient` η μεταβλητή `userLocation` ανανεώνεται με τις συντεταγμένες αυτής και ο χάρτης κεντράρει και ζουμάρει στην τοποθεσία του χρήστη. Αν δεν υπάρχει τελευταία γνωστή τοποθεσία, τότε καλείται η function `getNewLocation()`, που ζητάει συνεχείς ανανέωση της τοποθεσίας του χρήστη.

```
private fun getNewLocation() {
    val locationRequest = LocationRequest()
    locationRequest.priority =
LocationRequest.PRIORITY_HIGH_ACCURACY
    locationRequest.interval = 0
    locationRequest.fastestInterval = 0
    locationRequest.numUpdates = 1
    if (checkPermission()) {
        fusedLocationProviderClient =
LocationServices.getFusedLocationProviderClient(this)
        fusedLocationProviderClient.requestLocationUpdates(
            locationRequest, locationCallback, Looper.myLooper()
        )
    }
}
```

3.18: Function `getNewLocation()`

Η παραπάνω function αρχικά ορίζει τον τρόπο με τον οποίο ζητάμε την τοποθεσία το `PRIORITY_HIGH_ACCURACY` μας δίνει τον συνδυασμό της τοποθεσία από το GPS και το WIFI ή τα δεδομένα αν αυτά είναι ενεργοποιημένα, ενώ στις υπόλοιπες μεταβλητές ορίζουμε κάθε πότε ζητάμε ανανέωση τοποθεσίας. Οι τιμές αυτές καθορίζουν και τη χρήση της μπαταρίας της συσκευής από την εφαρμογή. Η function στη συνέχεια τσεκάρει αν υπάρχει και πάλι η απαραίτητη άδεια για τη χρήση της τοποθεσίας και αν υπάρχει ξεκινάει την ανανέωση της τοποθεσίας στα διαστήματα

που έχουμε ορίσει. Το αποτέλεσμα της τοποθεσίας είναι διαθέσιμο μέσω μιας μεταβλητής (*locationCallback*) που κάνει override το *onLocationResult*

```
private val locationCallback = object : LocationCallback() {  
    override fun onLocationResult(locationResult: LocationResult) {  
        val currentLLocation: Location =  
locationResult.lastLocation  
        userLocation = LatLng(currentLLocation.latitude,  
currentLLocation.longitude)  
    }  
}
```

3.19: locationCallback

Για να ελαχιστοποιήσουμε τη χρήση της μπαταρίας της συσκευής σταματάμε το ανανέωση της τοποθεσίας όταν το συγκεκριμένο activity δεν είναι στο προσκήνιο τη συσκευής, και συνεχίζουμε όταν ο χρήστης ξανανοίξει την εφαρμογή από το background. Αυτό επιτυγχάνεται αν κάνουμε override τα functions *onPause()* και *onResume()* στο activity.

```
override fun onPause() {  
    super.onPause()  
  
    fusedLocationProviderClient.removeLocationUpdates(locationCallback)  
}
```

3.20: Παύση ανανέωσης τοποθεσίας όταν το activity είναι στο παρασκήνιο

```
override fun onResume() {  
    super.onResume()  
    if(checkPermission()){  
        getNewLocation()  
    }  
}
```

3.21: Επανάραξη της ανανέωσης της τοποθεσίας όταν το activity επανέλθει στο προσκήνιο

3.14. Προετοιμασία και εμφάνιση χάρτη

Με τη δημιουργία του project το android studio δημιουργεί όλα τα απαραίτητα functions για την εμφάνιση του χάρτη στο class του activity, το μόνο που μένει είναι να τοποθετήσουμε τα δικά μας markers (απινιδωτές) και να ρυθμίσουμε πως θέλουμε να εμφανίζεται ο χάρτης και τι δυνατότητες να έχει. Ο χάρτης εμφανίζεται στην εφαρμογή μέσω fragment και η παραμετροποίησή του γίνεται μέσω του function *onMapReady()*.

```
override fun onMapReady(googleMap: GoogleMap) {
    mMap = googleMap

    //Creating custom marker
    val marker = MarkerOptions()

    marker.icon(BitmapDescriptorFactory.fromResource(R.drawable.marker)
    )

    //Creating LatLng variables
    val sete = LatLng(39.6258205, 19.9106273)
    val ethniki = LatLng(39.6238948, 19.9214520)
    val naske = LatLng(39.6255075, 19.9182310)
    val pentaфанaro = LatLng(39.6235919, 19.9236273)
    val basket = LatLng(39.6098500, 19.9150925)
    val airport = LatLng(39.6073261, 19.9140407)

    //Adding the above to the ArrayList
    dest.add(sete)
    dest.add(ethniki)
    dest.add(naske)
    dest.add(pentaфанaro)
    dest.add(basket)
    dest.add(airport)

    //Adding markers to the map
    mMap.addMarker(marker.position(sete).title(""))
    mMap.addMarker(marker.position(ethniki).title(""))
    mMap.addMarker(marker.position(naske).title(""))
    mMap.addMarker(marker.position(pentaфанaro).title(""))
    mMap.addMarker(marker.position(basket).title(""))
    mMap.addMarker(marker.position(airport).title(""))

    if (checkPermission()) {
        mMap.isMyLocationEnabled = true
        mMap.isTrafficEnabled = true
        mMap.uiSettings.isMyLocationButtonEnabled = true
    }
}
```

```
        getLastLocation()
    } else {
        requestPermission()
    }
}
```

3.22: Function onMapReady

Η παραπάνω function εισάγει αρχικά τον χάρτη σε μια μεταβλητή (mMap) που έχει αρχικοποιηθεί στην αρχή της κλάσης καθώς θέλουμε να έχουμε πρόσβαση σε αυτή και από άλλες functions κυρίως για το κεντράρισμα και ζουμάρισμα του χάρτη στην τοποθεσία του χρήστη. Στη συνέχεια δημιουργούμε τη μεταβλητή marker όπου έχει την εικόνα που θα χρησιμοποιήσουμε για να δείξουμε τη θέση των απινιδωτών στον χάρτη. Δημιουργούμε τις μεταβλητές με τις τοποθεσίες των απινιδωτών και τις εισάγουμε σε ένα ArrayList, το οποίο θα χρησιμοποιήσουμε όταν θα καλέσουμε το Distance Matrix API για τον προσδιορισμό της καλύτερης επιλογής απινιδωτή. Τέλος εισάγουμε τους απινιδωτές στον χάρτη. Αν υπάρχει η σχετική πάλι άδεια για χρήση της τοποθεσίας ενεργοποιούμε την τοποθεσία του χρήστη στον χάρτη, το κουμπί εύρεσης και εστίασης στην τοποθεσία, την οπτικοποίηση της κίνησης στους δρόμους και ζητάμε την τοποθεσία μέσω της getLastLocation() που περιγράψαμε σε προηγούμενο κεφάλαιο.

Η σχεδίαση του κουμπιού εύρεσης καλύτερης επιλογής έγινε με βάση το κουμπί εστίασης στην τοποθεσία του χρήστη από την google. Για την σχεδίαση χρησιμοποιήθηκε cardView και πάνω σε αυτό χρησιμοποιήθηκε ένα ImageButton με το εικονίδιο που δημιουργήθηκε. Οι αποστάσεις από τα άκρα τις οθόνη έγιναν και αυτές όπως το κουμπί εστίασης στην τοποθεσία του χρήστη. Για την διάδραση με τον χρήστη στο touch του κουμπιού δημιουργήθηκε αρχείο press.xml στον φάκελο drawable και περάστηκε ως background στο ImageButton. Ίσως η λογική αυτή αλλάξει σε νέα έκδοση της εφαρμογής καθώς δεν έχει τα επιθυμητά αποτελέσματα. Παρακάτω παρατίθεται ο κώδικας xml των press.xml και του activity που φιλοξενεί το fragment του χάρτη.

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_focused="true" android:state_pressed="true"
        android:drawable="@drawable/common_google_signin_btn_icon_disabled"
    />
    <item android:state_focused="false" android:state_pressed="true"
        android:drawable="@drawable/common_google_signin_btn_icon_disabled"
    />
</selector>
```

3.23: press.xml (Διάδραση με τον χρήστη)

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/background_light">

    <fragment
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:tag="mapView"
        app:layout_constraintTop_toTopOf="parent"
        tools:context=".MapsActivity"
        tools:layout_editor_absoluteX="0dp" />

    <androidx.cardview.widget.CardView
        android:layout_width="45dp"
        android:layout_height="45dp"
        android:layout_marginStart="10dp"
        android:layout_marginTop="10dp"
        app:cardBackgroundColor="#90FFFFFF"
        app:cardCornerRadius="1dp"
        app:cardElevation="0.5dp"
        app:cardMaxElevation="1dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <ImageButton
            android:id="@+id/findBestOption"
            style="@android:style/Widget.Material.Light.ImageButton"
            android:layout_width="45dp"
            android:layout_height="45dp"
            android:background="@drawable/press"
            android:contentDescription="@string/findBestRouteDesc"
            android:padding="5dp"
            android:scaleType="centerCrop"
            app:srcCompat="@drawable/best_button" />
    </ImageButton>
</CardView>
</ConstraintLayout>
```

```
</androidx.cardview.widget.CardView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

3.24: xml κώδικας του κύριου activity

3.15. Προσδιορισμός της καλύτερης επιλογής απινιδωτή

Ο προσδιορισμός της καλύτερης επιλογής όπως αναφέρθηκε και στα προηγούμενα κεφάλαια γίνεται με την κλήση του Distance Matrix API. Το Distance Matrix API, δέχεται σημεία εκκίνησης και προορισμού με συντεταγμένες και επιστρέφει σε JSON format τις ίδιες πληροφορίες γεωκωδικοποιημένες και πληροφορίες όπως την απόσταση του σημείου ή σημείων προορισμού από το σημείο εκκίνησης, του χρόνου που θα κάνει κάποιος να φτάσει σε αυτό. Αν έχει προστεθεί στην κλήση και ο χρόνος εκκίνησης το API επιστρέφει και τον χρόνο προσπέλασης της διαδρομής – διαδρομών με βάση την εκτίμηση της κίνησης στους δρόμους.

```
{
  "destination_addresses" : [
    "Eth. Antistaseos 6, Kerkira 491 00, Greece",
    "Vrachlioti 56, Kerkira 491 00, Greece",
    "Str. Xenofontos 2, Kerkira 491 00, Greece",
    "Viktoros Dousmani 70, Kerkira 491 00, Greece",
    "Anapafseos 10, Kerkira 491 00, Greece",
    "Kerkyra International Airport \"Ioannis Kapodistrias\"  
Operated by Fraport Greece, Corfu Airport, Cofu 491 00, Greece"
  ],
  "origin_addresses" : [ "Lochagou Spiridonos Vlaikou 3-9,  
Kerkira 491 00, Greece" ],
  "rows" : [
    {
      "elements" : [
        {
          "distance" : {
            "text" : "1.4 km",
            "value" : 1351
          },
          "duration" : {
            "text" : "4 mins",
            "value" : 221
          },
          "duration_in_traffic" : {
            "text" : "5 mins",
```

```

        "value" : 279
    },
    "status" : "OK"
},
{
    "distance" : {
        "text" : "1.0 km",
        "value" : 997
    },
    "duration" : {
        "text" : "4 mins",
        "value" : 220
    },
    "duration_in_traffic" : {
        "text" : "4 mins",
        "value" : 267
    },
    "status" : "OK"
},
{
    "distance" : {
        "text" : "2.0 km",
        "value" : 2029
    },
    "duration" : {
        "text" : "5 mins",
        "value" : 311
    },
    "duration_in_traffic" : {
        "text" : "6 mins",
        "value" : 351
    },
    "status" : "OK"
},
{
    "distance" : {
        "text" : "1.3 km",
        "value" : 1305
    },
    "duration" : {
        "text" : "4 mins",
        "value" : 259
    },
    "duration_in_traffic" : {
        "text" : "6 mins",
        "value" : 345
    },
    "status" : "OK"
},
{
    "distance" : {
        "text" : "3.2 km",
        "value" : 3223
    },
    "duration" : {
        "text" : "8 mins",
        "value" : 492
    }
}

```

```

    },
    "duration_in_traffic" : {
        "text" : "10 mins",
        "value" : 592
    },
    "status" : "OK"
},
{
    "distance" : {
        "text" : "2.9 km",
        "value" : 2861
    },
    "duration" : {
        "text" : "7 mins",
        "value" : 408
    },
    "duration_in_traffic" : {
        "text" : "8 mins",
        "value" : 504
    },
    "status" : "OK"
}
]
},
"status" : "OK"
}

```

3.25: Επιστροφή JSON του Distance Matrix Api

Για την κλήση του Distance Matrix API χρησιμοποιούμε τη βιβλιοθήκη okhttp3 οπότε και εισάγουμε το συγκεκριμένο dependency στο gradle. Η βιβλιοθήκη okhttp3 μας δίνει τη δυνατότητα να κάνουμε http request σε οποιονδήποτε server με εύκολο και γρήγορο τρόπο.

Η διαδικασία γίνεται σε νέο class ασύγχρονα με τη χρήση AsyncTask που τρέχει στο παρασκήνιο και μπορούμε να εμφανίσουμε το αποτέλεσμα της διαδικασίας αυτής στο προσκήνιο της εφαρμογής. Από το API level 30 και μετά η AsyncTask έχει καταργηθεί, έτσι σε μελλοντικό update η ίδια διαδικασία θα πρέπει να γίνεται με τη χρήση των coroutines της kotlin.

Πριν καλέσουμε όμως το Distance Matrix API θα πρέπει να κατασκευάσουμε το URL. Για την κατασκευή του URL κατασκευάστηκε η function `urlConstruction()`

```
private fun urlConstruction():String{
    val destString = dest.joinToString(separator = "|")
    val destinations = destString.replace("lat/lng:",
    "").replace("(", "").replace(")", "")
    return
    "https://maps.googleapis.com/maps/api/distancematrix/json?units=metric&origins=${
    userLocation.latitude},${userLocation.longitude}&destinations=${destinations}&departure_time=
    now&key=API_KEY"
```

3.26: Κατασκευή του URL για την κλήση του Distance Matrix Api

Αρχικά μετατρέπουμε την `arrayList dest` σε `string` η οποία περιέχει τις συντεταγμένες που εισάγαγε στην `function onMapReady()`. Στη συνέχεια διαμορφώνουμε το `string` σύμφωνα με τους κανόνες κλήσης του API. Τέλος επιστρέφουμε το συνολικό URL.

Οι παράμετροι που ορίζουμε στο URL είναι:

- Units. Επιλογή μετρικού συστήματος στο οποίο θέλουμε να είναι τα δεδομένα που θα επιστρέψει το json (miles για imperial, kilometers για metric)
- Origins. Σημείο εκκίνησης για τον υπολογισμό, για την εφαρμογή μας είναι η τρέχουσα τοποθεσία του χρήστη. Μπορεί να οριστεί και παραπάνω από ένα.
- Destinations. Τελικό σημείο για τον υπολογισμό, για την εφαρμογή είναι τα σημεία των απινιδωτών.
- Departure_time. Το συγκεκριμένο σημείο δεν είναι υποχρεωτικό αν όμως οριστεί σε κάποια ώρα μεγαλύτερη ή ίση της τρέχουσας τότε το API επιστρέφει και τον εκτιμώμενο χρόνο με βάση την κίνηση.
- Key. Το API key που χρησιμοποιούμε για την πρόσβαση στο API. Το API δέχεται και άλλες παραμέτρους που δεν χρησιμοποιούμε στη συγκεκριμένη εφαρμογή προς το παρόν. Για την εκτίμηση της κίνησης υπάρχει και η παράμετρος `traffic_model` και παίρνει τιμές:
- Best_guess. Είναι η default τιμή (αν δεν οριστεί καθόλου) και κάνει την καλύτερη εκτίμηση με βάση την τρέχουσα κατάσταση και τα ιστορικά δεδομένα κίνησης για τη διαδρομή.
- Pessimistic. Επιστρέφει διάρκεια διαδρομής μεγαλύτερη της πραγματικής τις περισσότερες φορές.

- Optimistic Επιστρέφει διάρκεια διαδρομής μικρότερη της πραγματικής τις περισσότερες φορές.

Η καλύτερη επιλογή λοιπόν είναι τα το αφήσουμε στο `best_guess`.

Η διαδικασία της εύρεσης της καλύτερης επιλογής γίνεται όταν ο χρήστης πατήσει το σχετικό κουμπί, στο οποίο έχουμε αναθέσει `clickListener`

```

        findBestOptionButton.setOnClickListener {
            gpsEnabled =
lm.isProviderEnabled(LocationManager.GPS_PROVIDER)
            if (checkPermission()) {
                if (!gpsEnabled) {
                    showLocationPrompt()
                } else {
                    if (isOnline(this)) {
                        getNewLocation()
                        FindBestOption(this, urlConstruction(),
dest).execute()
//Toast.makeText(this, urlConstruction(),
Toast.LENGTH_LONG).show()
                    } else {
                        Toast.makeText(
                            this,
                            getString(R.string.checkConnectivity),
                            Toast.LENGTH_LONG
                        ).show()
                    }
                }
            } else {
                Toast.makeText(
                    this,
                    getString(R.string.enableGpsPermission),
                    Toast.LENGTH_LONG
                ).show()
            }
        }
    }
}

```

3.27: Κώδικας στο κουμπί εύρεσης καλύτερης επιλογής

Πατώντας το κουμπί η εφαρμογή αρχικά τσεκάρει αν έχει δοθεί η απαραίτητη άδεια για την χρήση της τοποθεσίας και εμφανίζει σχετικό μήνυμα ενημερώνοντας το χρήστη να την ενεργοποιήσει αν δεν το έχει κάνει. Στη συνέχεια τσεκάρει αν είναι ενεργοποιημένο το GPS και αν δεν είναι εμφανίζεται παράθυρο στον χρήστη να το ενεργοποιήσει καλώντας το function `showLocationPrompt()`, ακριβώς μετά τσεκάρει αν είναι ενεργοποιημένα τα δεδομένα ή το WIFI καθώς η σύνδεση στο internet είναι απαραίτητη για την κλήση του API, και αν δεν είναι ενημερώνει τον χρήστη με σχετικό

μήνυμα. Αν όλα είναι εντάξει τότε καλείται η `getNewLocation()` για να πάρουμε την τρέχουσα τοποθεσία, και στη συνέχεια εκτελούμε την κλάση `FindBestOption()` η οποία δέχεται παραμέτρους:

- `Context`. Το χρειαζόμαστε για να ξεκινήσουμε το navigation του google maps μέσα από το `asyncTask`.
- `url`. Το string που επιστρέφει η `urlConstruction()`
- `Locations`. Η λίστα με τα locations που δημιουργήσαμε στην `onMapReady`. Τη χρειαζόμαστε για να επιστρέψουμε από αυτήν το αποτέλεσμα της καλύτερης επιλογής.

```
class FindBestOption (context: MapsActivity, private val url : String,
private val locations: ArrayList<LatLng>): AsyncTask<Void, Void, LatLng>() {
    private var timeList:ArrayList<Int> =arrayListOf()
    private var hasError = false
    private val nullLocation = LatLng(0.0, 0.0)
    private val activityReference: WeakReference<MapsActivity> =
WeakReference(context)
    override fun doInBackground(vararg params: Void?): LatLng {
        try {
            val client = OkHttpClient()
            val request = Request.Builder().url(url).build()
            val response = client.newCall(request).execute()
            val data = response.body!!.string()
            //println(data)
            val distancesObject = JSONObject(data)
            val jsonArrayRows: JSONArray =
distancesObject.getJSONArray("rows")
            val rowSize: Int = jsonArrayRows.length()

            for (i in 0 until rowSize) {
                val jsonObjectRows: JSONObject =
jsonArrayRows.getJSONObject(i)
                val jsonArrayElements: JSONArray =
jsonObjectRows.getJSONArray("elements")
                val elementSize: Int = jsonArrayElements.length()
                for (d in 0 until elementSize) {
                    val jsonObjectElements: JSONObject =
jsonArrayElements.getJSONObject(d)
                    val durationString =
jsonObjectElements.getString("duration_in_traffic")
                    val durationObject = JSONObject(durationString)

                    //Getting duration JSON Object
                    val duration = durationObject.getInt("value")

                    //Add duration record to a list
                    timeList.add(duration)
                }
            }
        } catch (e: Exception) {
            hasError = true
        }
        return nullLocation
    }
}
```

```

        }
    }
    val min = timeList.indexOf(timeList.minOrNull() ?: 0)
    return locations[min]
} catch (e: Exception) {
    hasError = true
    return nullLocation
}
}

override fun onPostExecute(location: LatLng) {
    val activity = activityReference.get()
    if (!hasError) {
        val gmmIntentUri =
Uri.parse("google.navigation:q=${location.latitude},${location.longitude}")

        val mapIntent = Intent(Intent.ACTION_VIEW, gmmIntentUri)
        mapIntent.setPackage("com.google.android.apps.maps")
        activity?.startActivity(mapIntent)
    } else {
        Toast.makeText(
            activity,
            activity!!.getString(R.string.checkConnectivity),
            Toast.LENGTH_LONG
        ).show()
    }
}
}

```

3.28: Κλάση FindBestOption

Στην παραπάνω κλάση γίνεται η κλήση του Distance Matrix API. Αρχικά αρχικοποιούμε κάποιες μεταβλητές.

- `timeList`: ArraList όπου θα αποθηκευτούν οι τιμές των πεδίων `duration_traffic`.
- `hasError`: Boolean όπου παίρνει την τιμή `true` εάν υπάρχει κάποιο σφάλμα κατά την κλήση του API.
- `nullLocation`: `LatLng(0.0, 0.0)`. Πρέπει να επιστραφεί `LatLng` ακόμα και αν υπάρχει σφάλμα.
- `activityReference`: Είναι ένα `weak context reference` το οποίο θα χρησιμοποιήσουμε για να καλέσουμε την εφαρμογή `google maps` αφού ολοκληρωθεί η διαδικασία και έχουμε αποτέλεσμα. Η χρήση του `Weak context`

reference γίνεται για να αποφύγουμε memory leaks καθώς η AsyncTask εκτελείται στο background. Είναι και ο κύριος λόγος που καταργείται από το API 30 και μετά.

Κάνουμε override την *doInBackground* και κάνουμε την κλήση στο API με τη βοήθεια της βιβλιοθήκης okhttp3. Το κύριο σφάλμα που παρατηρήθηκε κατά τη διάρκεια του ελέγχου της εφαρμογής ήταν η αδυναμία κλήσης του API λόγο *timeOutException*. Αυτό συμβαίνει στις περιπτώσεις που ενώ είναι ενεργοποιημένο το WIFI ή τα δεδομένα δεν υπάρχει αξιόπιστη σύνδεση ή δεν υπάρχουν άλλα δεδομένα στο πρόγραμμα του χρήστη με αποτέλεσμα η εφαρμογή να τερματίζει απρόοπτα. Έτσι λοιπόν χρησιμοποιούμε την *try – catch* για να τσεκάρουμε για σφάλματα κατά το Request. Στην περίπτωση που η διαδικασία ολοκληρωθεί τότε εισάγουμε όλα τις τιμές από τα πεδία του *duration_in_traffic* στην *timeList*, βρίσκουμε το *index* της ελάχιστης τιμής στην λίστα και επιστρέφουμε την τιμή της λίστας *locations* στο *index* της ελάχιστης τιμής, αφού έχουν το ίδιο μέγεθος και την ίδια σειρά.

Στη συνέχεια κάνουμε override την *onPostExecute()* η οποία δέχεται ως παράμετρο την επιστρεφόμενη μεταβλητή *LatLng* από την *doInBackground*. Αρχικά τσεκάρει αν υπήρξε error από την *doInBackground*, εμφανίζοντας μήνυμα στον χρήστη. Αν δεν υπάρχει κάποιο error τότε ανοίγει η εφαρμογή google maps για να ξεκινήσει το navigation του χρήστη στον απινιδωτή.

3.16. Δημιουργία οθόνης πληροφοριών για την εφαρμογή και επικοινωνίας με τον προγραμματιστή

Στην εφαρμογή υπάρχει οθόνη με πληροφορίες σχετικά με την εφαρμογή και δυνατότητα επικοινωνίας με τον προγραμματιστή για οποιαδήποτε απορία ή πρόβλημα. Η οθόνη αυτή είναι προσβάσιμη στον χρήστη από το menu της εφαρμογής που ανοίγει το συγκεκριμένο activity. Για το menu δημιουργήθηκε αρχείο xml με το εικονίδιο που φαίνεται στην παρακάτω εικόνα.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
  <item
```

```

        android:id="@+id/info"
        android:icon="@android:drawable/ic_menu_info_details"
        android:title="@string/info"
        app:showAsAction="always" />
    </menu>

```

3.29: Αρχείο xml του μενού της εφαρμογής



3.30: Κουμπί μενου που ανοίγει την οθόνη πληροφοριών

```

override fun onCreateOptionsMenu(menu: Menu?): Boolean {
    menuInflater.inflate(R.menu.app_menu, menu)
    return super.onCreateOptionsMenu(menu)
}

```

3.31: Εισαγωγή του μενού στην ActionBar

Η παραπάνω function εισάγει το μενού που έχουμε δημιουργήσει στην ActionBar της εφαρμογής.

```

override fun onOptionsItemSelected(item: MenuItem): Boolean {
    val id: Int = item.itemId
    if (id == R.id.info) {
        val intent = Intent(this,
        AboutActivity::class.java).apply {
            startActivityForResult(intent, 0)
        }
        startActivity(intent)
    }
    return super.onOptionsItemSelected(item)
}

```

3.32: Διαχείριση των εικονιδίων του μενού

Η function `onOptionsItemSelected` διαχειρίζεται τα click σε κάθε εικονίδιο ή item του menu. Στην συγκεκριμένη περίπτωση στο πάτημα του εικονιδίου ανοίγει ένα νέο activity με τις σχετικές για την εφαρμογή πληροφορίες βλ. [σχήμα 3.7](#). Παρακάτω παρατίθεται ο xml κώδικας του menu και του activity πληροφοριών.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/info"
        android:icon="@android:drawable/ic_menu_info_details"
        android:title="@string/info"
        app:showAsAction="always" />
</menu>
```

3.33: Κώδικας xml αρχείου menu

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    tools:layout_editor_absoluteX="1dp"
    tools:layout_editor_absoluteY="1dp">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        tools:context=".AboutActivity">

        <TextView
            android:id="@+id/textView2"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_margin="15dp"
            android:text="@string/AboutUsage"

            android:textColor="@color/common_google_signin_btn_text_dark_focused"

            android:textSize="18sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

    <TextView
```

```

        android:id="@+id/textView3"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_margin="15dp"
        android:layout_marginTop="40dp"
        android:text="@string/ContactUs"

        android:textColor="@color/common_google_signin_btn_text_dark_focused"

        android:textSize="18sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />

<TextView
    android:id="@+id/textView4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="15dp"
    android:layout_marginTop="40dp"
    android:autoLink="web|email"
    android:text="@string/Mail"
    android:textColor="@color/colorPrimaryDark"

    android:textColorHighlight="?android:attr/colorBackground"
    android:textColorHint="@color/colorPrimaryDark"
    android:textColorLink="@color/colorPrimary"
    android:textSize="18sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView3" />
</androidx.constraintlayout.widget.ConstraintLayout>
</ScrollView>

```

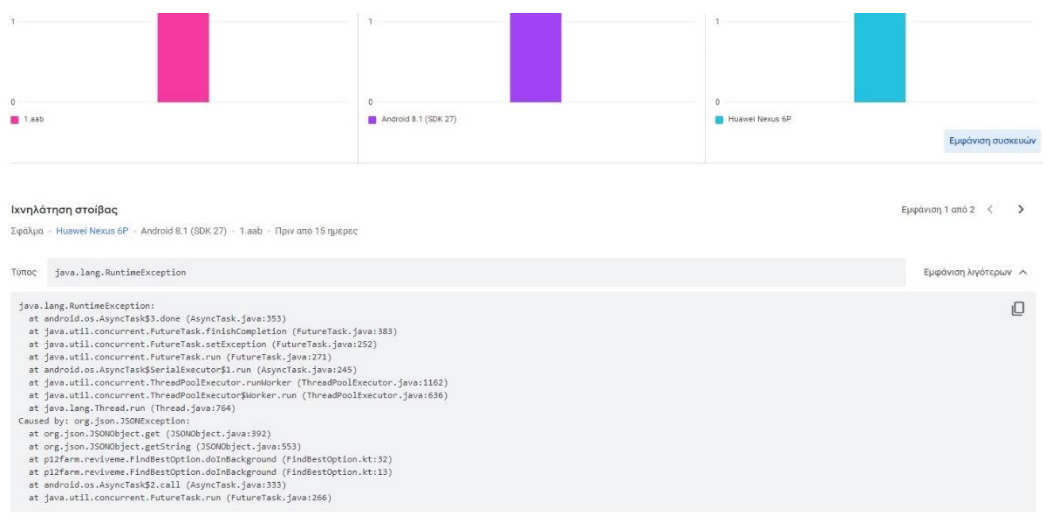
3.34: xml κώδικας του activity πληροφοριών

Για να είναι ορατή η πληροφορία στην οθόνη από όλες τις συσκευές ανεξαρτήτως προσανατολισμού (landscape – portrait) είναι αναγκαίο εισάγουμε ένα scrollView ως parent element έτσι ώστε όταν έχουμε overflow της πληροφορίας στην οθόνη να μπορούμε να κάνουμε scroll. Δεν υπάρχει κάτι άλλο σημαντικό στο συγκεκριμένο αρχείο καθώς τα περισσότερα elements είναι κυρίως textViews τα οποία δείχνουν στον χρήστη κείμενο το οποίο αντλούν από μεταβλητές string. Η λογική με τις string μεταβλητές στο κείμενο ακολουθείτε σε όλα τα σημεία του κώδικα τις εφαρμογής και εξυπηρετεί στην καλύτερη διαχείριση του περιεχομένου και τις μεταφράσεις αυτού. Επίσης το Android studio ενημερώνει ότι είναι καλό το κείμενο να εισάγεται υπό τη μορφή μεταβλητής string η οποία πρέπει να είναι δηλωμένη στο αρχείο strings.xml.

3.17. Παρουσία στο Play store

Η εφαρμογή έχει δημοσιευθεί με επιτυχία στο play store και βρίσκεται σε κατάσταση εσωτερικής δοκιμής. Αυτό σημαίνει ότι η εφαρμογή είναι διαθέσιμη για download μέσω ειδικού URL σε συγκεκριμένα άτομα που έχουν δηλωθεί από τον προγραμματιστή. Σκοπός είναι να κυκλοφορήσει κανονικά αφού γίνουν οι απαραίτητες διορθώσεις από το feedback των χρηστών. Πριν την πραγματική κυκλοφορία, η εφαρμογή θα κυκλοφορήσει διαθέσιμη ως test σε οποιονδήποτε χρήστη είναι διαθέσιμος να την τεστάρει και το έχει δηλώσει στο play store.

Η εφαρμογή βρίσκεται στην έκδοση 1.4 και έχουν διορθωθεί πολλά σφάλματα από την έκδοση 1.0. Η εκδόσεις τεστ πριν την κύρια κυκλοφορία είναι πολύ σημαντικές καθώς οι προγραμματιστές μπορούν να δουν τι σφάλματα, σε ποιες συσκευές και κάτω από ποιες συνθήκες συνέβησαν ώστε να τα διορθώσουν πριν την κανονική κυκλοφορία. Στην παρακάτω ενδεικτική οθόνη από το play console βλέπουμε ένα σφάλμα και τις λεπτομέρειες του.



3.35: Ενημέρωση για σφάλμα εφαρμογής απο το play console

3.18. Βελτίωση και εξέλιξη εφαρμογής

Η εφαρμογή πρέπει να εξελίσσεται συνεχώς ανεξάρτητα με το αν προστίθενται νέες δυνατότητες και λειτουργίες σε αυτή. Το οικοσύστημα του Android αλλάζει συνεχώς και η εφαρμογή θα πρέπει να ενημερώνεται με τις νέες οδηγίες και λειτουργίες για να συνεχίσει να διατίθεται απροβλημάτιστα στις επόμενες εκδόσεις android παράδειγμα η κατάργηση του AsyncTask που αναφέρθηκε σε προηγούμενο κεφάλαιο, θα επηρεάσει τη λειτουργία της εφαρμογής και θα πρέπει η διαδικασία να ξαναγραφτεί με όποια διαδικασία αντικατασταθεί το AsyncTask.

Το UI και αυτό θα πρέπει να συμβαδίζει με τις τάσεις της εποχής και της οδηγίες της google για καλές πρακτικές. Πρέπει να δοθεί έμφαση και στο σχεδιασμό για άτομα με αναπηρίες, καθώς είναι μια εφαρμογή ιατρικού περιεχομένου και η χρήση της γίνεται σε περιπτώσεις εξαιρετικής ανάγκης.

Κάθε τόσο βγαίνουν νέες συσκευές στην αγορά με διαφορετικές αναλύσεις οθόνης και νέες δυνατότητες που θα πρέπει να εκμεταλλευτούμε όποτε μπορούμε.

Στην περίπτωση που μεγαλώσει η εφαρμογή και εισαχθούν σε αυτή περισσότερες θέσεις απινιδωτών, ο τρόπος κλήσης του API θα πρέπει να αλλάξει, καθώς αυτή τη στιγμή η καλύτερη επιλογή γίνεται από όλους τους διαθέσιμους απινιδωτές κατι το οποίο είναι ασύμφορο, χρονοβόρο και άνευ ουσίας για δεκάδες ή ακόμα και εκατοντάδες απινιδωτές. Έτσι λοιπόν θα πρέπει να βρίσκει την καλύτερη επιλογή από τους απινιδωτές που βρίσκονται εντός μιας ακτίνας από την τοποθεσία του χρήστη.

Οι δυνατότητες όμως είναι πολλές. Θα πρέπει να δοθεί η δυνατότητα στους οργανισμούς και τους φορείς υγείας να περνάνε της θέσεις των απινιδωτών μέσω ειδικής web πλατφόρμας και η εφαρμογή να τις κατεβάζει από μια βάση δεδομένων και να τις αποθηκεύει στην τοπική SQLite βάση ώστε οι θέσεις να είναι διαθέσιμες στον χρήστη και offline.

Εκτός από την εύρεση απινιδωτών η εφαρμογή μπορεί να εξελιχθεί κάνοντας διαθέσιμες στον χρήστη και άλλες υπηρεσίες υγείας όπως εφημερεύοντα φαρμακεία, ιατρεία νοσοκομεία κ.α. Για την καλύτερη εξυπηρέτηση του χρήστη αλλά και την αποσυμφόρηση των υπηρεσιών υγείας μπορεί να δημιουργηθεί ειδική πλατφόρμα όπου οι υπηρεσίες υγείας (νοσοκομεία, ιατρεία, φαρμάκια κ.α.) θα ενημερώνουν για τον

χρόνο αναμονής σε αυτά. Η εφαρμογή θα υπολογίζει εκτός από την απόσταση και την κίνηση και τον χρόνο αναμονής στις επιλεγμένες υπηρεσίες και θα κατευθύνει τον χρήστη στην υπηρεσία με το μικρότερο άθροισμα χρόνου (χρόνος πρόσβασης + χρόνος αναμονής).

4. Συμπεράσματα

Από την αρχαιότητα έως σήμερα η έννοια του χωρικού εντοπισμού παίζει σημαντικό ρόλο σε όλους τους τομείς της ανθρώπινης δραστηριότητας. Η εξέλιξη της τεχνολογίας μα έδωσε τα μέσα να μπορούμε να εντοπίσουμε τόσο τη θέση μας όσο και τη θέση ενός άλλου αντικειμένου η γεωγραφικής οντότητας στον χώρο εύκολα και γρήγορα, το πρόβλημα όμως της βέλτιστης διαδρομής θα μας απασχολεί για καιρό ακόμα. Παρ όλες τις μαθηματικές λύσεις που προτάθηκαν κατά καιρούς για το πρόβλημα της βέλτιστης διαδρομής που αποτυπώθηκε ξεκάθαρα στο πρόβλημα του πλανόδιου πωλητή το πρόβλημα παραμένει.

Οι λύσεις και οι αλγόριθμοι που προτάθηκαν για το πρόβλημα της βέλτιστης διαδρομής θεωρούν το πρόβλημα ως κάτι στατικό και με αυτή την έννοια οι αλγόριθμοι κάνουν και θα κάνουν στο μέλλον ακόμα καλύτερη δουλειά, όμως το πρόβλημα της βέλτιστης διαδρομής δεν είναι στατικό, αλλά μεταβαλλόμενο με βάση τις νέες ανάγκες που προκύπτουν από την εξέλιξη της κοινωνίας. Όταν διατυπώθηκε το πρόβλημα για πρώτη φορά δεν υπήρχε η έννοια της κίνησης και του μοτιλιαρίσματος πάρα μόνο η χλιομετρική απόσταση μεταξύ των κόμβων ενός δικτύου. Ακόμα και στις μέρες μας δεν είμαστε σε θέση να υπολογίσουμε με ακρίβεια τη κίνηση στο οδικό δίκτυο.

Η έννοια όμως της βέλτιστης διαδρομής περιπλέκεται περισσότερο αν λάβουμε υπόψη μας μια κατάσταση ανάγκης. Για παράδειγμα σε ένα έκτακτο περιστατικό είναι σημαντική η μεταφορά ενός ασθενή η τραυματία σε νοσοκομείο. Αν οι διαθέσιμες επιλογές είναι πχ 3 (Νοσοκομείο Α, Νοσοκομείο Β, Νοσοκομείο Γ) θα πρέπει να βρούμε την πιο γρήγορη επιλογή από την τοποθεσία του ασθενή. Έστω λοιπόν ότι το νοσοκομείο Α απέχει 2χμ από την τοποθεσία του ασθενή, το νοσοκομείο Β 2.5χμ και το νοσοκομείο Γ 3χμ η βέλτιστη επιλογή είναι ξεκάθαρα το Α. Αν όμως ο χρόνος που χρειαζόμαστε να φτάσουμε στο Α είναι 20 λεπτά λόγω αυξημένης κίνησης, στο Β 10

λεπτά και στο Γ 15 λεπτά, τότε βέλτιστη επιλογή είναι το Β. Για να περιπλέξουμε ακόμα πιο πολύ τα πράγματα αν το νοσοκομείο Α δεν έχει καθόλου αναμονή, το νοσοκομείο Β έχει 20 λεπτά αναμονή ενώ το νοσοκομείο Γ 10 λεπτά τότε η βέλτιστη επιλογή είναι και πάλι το Α αφού αυτό που μας ενδιαφέρει δεν είναι απλά να φτάσουμε γρήγορα, αλλά να νοσηλευτεί ο ασθενής όσο το δυνατόν γρηγορότερα ειδικά σε περιπτώσεις έκτακτης ανάγκης (τροχαία, σοβαρές παθήσεις).

Καταλαβαίνουμε λοιπόν ότι το πρόβλημα της βέλτιστης διαδρομής θα πρέπει να λάβει το δίκτυο σαν μια συνεχώς μεταβαλλόμενη οντότητα και όχι σαν κάτι στατικό που μπορεί να επιλυθεί με τη χρήση απλών αλγορίθμων. Τι θα πρέπει να συμβεί αν η συνθήκες και οι χρόνοι αλλάζουν πριν ο χρήστης φτάσει στον προορισμό του; Αν υπάρχει εκείνη τη στιγμή διαθέσιμη διαδρομή ή προορισμός γρηγορότερα προσπελάσιμος από την νέα τοποθεσία του χρήστη, ο χρήστης θα πρέπει να οδηγηθεί σε αυτή.

Η τεχνολογία μας δίνει τη δυνατότητα να κατασκευάσουμε συστήματα που μπορούν και δίνουν ολοένα και καλύτερες λύσεις στο παραπάνω πρόβλημα. Το μεγαλύτερο μέρος του οδικού δικτύου του πλανήτη έχει χαρτογραφηθεί με μεγάλη ακρίβεια και είναι διαθέσιμο σε οποιονδήποτε διαθέτει μια κινητή συσκευή (Τηλέφωνο – tablet κ.α.) Η ακρίβεια του εντοπισμού της θέσης έχει και αυτή εξελιχθεί όπως και οι αλγόριθμοι πλοήγησης των μεγάλων εταιριών όπως η Google. Αυτό που πρέπει να γίνει είναι να κατηγοριοποιηθούν και αναλυθούν οι σύγχρονες ανάγκες που ορίζουν μια βέλτιστη διαδρομή εκτός από τον χρόνο μετάβασης στον προορισμό και να εισαχθούν ως πιθανά βάρη που θα αλλάζουν συνεχώς δυναμικά στους τερματικούς κόμβους των δικτύων.

Οι εφαρμογές των κινητών τηλεφώνων θα πρέπει να εξελιχθούν και αυτές ώστε να λαμβάνουν υπόψη τους τις επιθυμίες του χρήστη πριν εισάγουν τα βάρη στους κόμβους, καθώς η κάθε περίπτωση είναι ιδιαίτερη και η βέλτιστη διαδρομή μπορεί να είναι διαφορετική για κάθε προορισμό και για κάθε χρήστη.

Βιβλιογραφία

Βιβλία

Robinson H, Arthur, Morrison L. Joel, Muehrcke C. Phillip, Kimerling A. Jon, Guptil C. Stephen, (2002). Στοιχεία χαρτογραφίας, Πανεπιστημιακές εκδόσεις Ε.Μ.Π.

Δούκας Ι. Δ, Μπαντέλας Α. Γ, Σαββαΐδης Π. Δ, Υφαντής Ι. Μ, (1999). Γεωδαισία, τόμος 1, δεύτερη έκδοση.

Κουτσόπουλος Κωστής, (2005), Γεωγραφικά συστήματα πληροφοριών και ανάλυση χώρου, εκδόσεις Παπασωτηρίου

Στεφανάκης, Εμμανουήλ, (1999). Βάσεις Γεωγραφικών δεδομένων και συστήματα γεωγραφικών πληροφοριών, Εκδόσεις Παπασωτηρίου.

Εργασίες

Σπυρίδων Νικολαΐδης, (2005), Υλοποίηση αλγορίθμου εύρεσης βέλτιστης διαδρομής κινούμενης μονάδας, Αριστοτέλειο πανεπιστήμιο Θεσσαλονίκης.

Χαραλαμπίδου Σοφία, (2005), Ανάπτυξη μοντέλων βελτιστοποίησης διαδρομής για route planners: η περίπτωση του i-Mobi Volos, Πανεπιστήμιο Θεσσαλίας Τμήμα Πολιτικών Μηχανικών.

Ιστοτόποι

Developers.android.com

- <https://developer.android.com/kotlin>, (22/09/2020)
- <https://developer.android.com/topic/libraries/architecture/coroutines>, (24/09/2020)

Developers.google.com

- <https://developers.google.com/location-context/fused-location-provider>, (24/09/2020)
- <https://developers.google.com/maps/documentation/distance-matrix/usage-and-billing>, (22/09/2020)

Doctorcodetutorial, <https://doctorcodetutorial.blogspot.com/2020/05/android-simplest-way-to-get-user.html> (15/9/2020)

GitHub, <https://square.github.io/okhttp/>, (24/09/2020)

Google cloud platform, <https://cloud.google.com/maps-platform/pricing>, (22/09/2020)

Groovypost, <https://www.groovypost.com/howto/seven-google-maps-alternatives-and-why-theyre-better/> (πρόσβαση 18/9/2020)

Jetbrains, <https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/> (22/09/2020)

Kotlinlang.com, <https://kotlinlang.org/docs/reference/faq.html>, (22/09/2020)

NASA History Division, <https://history.nasa.gov/sputnik.html>. (17/9/2020)

Nasa, https://www.nasa.gov/directorates/heo/scan/communications/policy/GPS_History.html, (17/9/2020)

National Geographic, <https://www.nationalgeographic.org/encyclopedia/navigation/> (17/9/2020)

Netmarketshare, <https://www.netmarketshare.com/operating-system-market-share.aspx?options=%7B%22filter%22%3A%7B%22%24and%22%3A%5B%7B%22deviceType%22%3A%7B%22%24in%22%3A%5B%22Mobile%22%5D%7D%7D%5D%7D%2C%22dateLabel%22%3A%22Custom%22%2C%22attributes%22%3A%22share%22%2C%22group%22%3A%22platform%22%2C%22sort%22%3A%7B%22share%22%3A%22id%22%3A%22platformsMobile%22%2C%22dateInterval%22%3A%22Monthly%22%2C%22dateStart%22%3A%222020-08%22%2C%22dateEnd%22%3A%222020-08%22%2C%22segments%22%3A%22-1000%22%7D>, (21/09/2020)

Raywenderlich.com, <https://www.raywenderlich.com/230-introduction-to-google-maps-api-for-android-with-kotlin> 15/9/2020)

Stack Overflow, <https://stackoverflow.com/questions/15727912/sha-1-fingerprint-of-keystore-certificate/15727931>, (23/09/2020)

Britannica, <https://www.britannica.com/technology/radar/History-of-radar>
(17/9/2020)