

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής



ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

***Εξατομικευμένος Ηλεκτρονικός Οδηγός με βάση επιλογές και
ενδιαφέροντα***

Ευάγγελος Κορώνης του Δημητρίου, Π17050

Επιβλέπων: Επίκουρος Καθηγητής Ε. Σακκόπουλος

Πειραιάς, Ιούλιος 2021





Επιτελική Σύνοψη

Η Αθήνα είναι μια μητρόπολη με πλούσια ιστορία και πολιτισμό που χρονολογείται χιλιάδες χρόνια πίσω, αλλά συγχρόνως μια σύγχρονη πρωτεύουσα γεμάτη ζωή και δρώμενα. Για αυτό τον λόγο, προσφέρει σε οποιονδήποτε την επισκέπτεται ή αποτελεί κάτοικος της πόλης απεριόριστα σημεία ενδιαφέροντος. Λόγω, όμως, της μεγάλης ποικιλίας δραστηριοτήτων και αξιοθέατων που διαθέτει, καθώς και της αχανής έκτασής της, καθίσταται δύσκολη η διαδικασία της επιλογής και της εύρεσης προορισμών, οι οποίοι συμπίπτουν με τα ενδιαφέροντα του περιηγητή. Βέβαια, υπάρχουν πολλοί ιστότοποι και εφαρμογές που προτείνουν διάφορα σημεία ενδιαφέροντος στην Αθήνα, ή και κάποιες που προσφέρουν έτοιμα προγράμματα ξενάγησης. Αλλά ακόμα και σε αυτές τις περιπτώσεις, ο χρήστης θα πρέπει να σπαταλήσει ένα σημαντικό χρόνο σε έρευνα και αναζήτηση στο διαδίκτυο, κάτι το οποίο πολλές φορές δεν είναι εύκολο. Ακόμα χειρότερα ο περιηγητής μπορεί να καταλήξει μπερδεμένος από τον μεγάλο όγκο πληροφοριών που υπάρχει διαθέσιμος. Όλα αυτά έχουν σημαντικές επιπτώσεις στην εμπειρία που αφήνει η παραμονή του στην πόλη. Ως λύση στα προαναφερθέντα προβλήματα σχεδιάσαμε μια εφαρμογή για android κινητό, εύχρηστη, γρήγορη και ασφαλής, η οποία θα βοηθήσει τον χρήστη στην εύρεση σημείων ενδιαφέροντος στην Αθήνα, προσαρμοσμένα στα δικά του ενδιαφέροντα και ανάγκες. Μάλιστα επειδή στην Αθήνα οι αποστάσεις είναι αρκετά μεγάλες, πρέπει τα σημεία να είναι όσο το δυνατόν κοντά μεταξύ τους ώστε να μην αναγκάζεται ο περιηγητής να προβαίνει σε μεγάλες μετακινήσεις. Επιπλέον, η εφαρμογή πρέπει να μπορεί να εξυπηρετεί τόσο τους Έλληνες όσο και τους ξένους επισκέπτες, όποτε σημαντικό ρόλο παίζει και η προσαρμογή της γλώσσας. Με βάση αυτές τις απαιτήσεις, καταλήξαμε στην δημιουργία της εφαρμογής My Tour Guide, η οποία θα δώσει τέλος σε όλη αυτή την μάταιη και επίπονη διαδικασία της αναζήτησης του περιηγητή για προορισμούς εντός της πόλης που να ταιριάζουν στις προτιμήσεις του. Το μόνο που απαιτείται από τον χρήστη είναι να συμπληρώσει ένα σύντομο προφίλ με κάποια προσωπικά χαρακτηριστικά. Στην συνέχεια η εφαρμογή, με την βοήθεια ενός αλγορίθμου σύγκρισης ομοιότητας, θα υπολογίσει μια διαδρομή εξατομικευμένη στις προτιμήσεις και τις ανάγκες του.



Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επίκουρο καθηγητή κ. Ευάγγελο Σακκόπουλο, για την βοήθεια ,την καθοδήγηση και τις γνώσεις που μου πρόσφερε τόσο κατά την διάρκεια της εκπόνησης της παρούσας πτυχιακής εργασίας, όσο κατά διάρκεια των υπολοίπων μαθημάτων στα οποία συμμετείχα. Επίσης θα ήθελα να ευχαριστήσω όλους τους διδάσκοντες του τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς για τις γνώσεις που αποκόμισα από αυτούς κατά την διάρκεια της φοίτησής μου.

Ιούλιος 2021
Κορώνης Ευάγγελος



Περιεχόμενα

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ.....	7
1 Εισαγωγή.....	8
1.1 Περιγραφή του υπό μελέτη προβλήματος.....	8
1.2 Ανάλυση σχετικών εφαρμογών.....	8
1.3 Σκοπός και στόχοι της εργασίας.....	9
1.4 Βασικοί ορισμοί.....	9
1.4.1 Android OS.....	9
1.4.2 Αντικειμενοστραφής προγραμματισμός.....	9
1.4.3 Γλώσσα προγραμματισμού Java.....	10
1.4.4 Android Studio.....	10
1.4.5 Firebase.....	10
1.4.6 API.....	10
1.4.7 XML.....	10
1.5 Παραδοτέα της εργασίας.....	11
2 Ανάλυση και σχεδίαση της εφαρμογής.....	13
2.1 Διαγράμματα περίπτωσης χρήσης (Use Case Diagrams).....	13
2.1.1 Σύνδεση χρήστη στην εφαρμογή.....	13
2.1.2 Διαδικασία συμπλήρωσης προφίλ και εμφάνιση προτάσεων.....	14
2.2 Διαγράμματα ακολουθίας (Sequence Diagrams).....	15
2.2.1 Διάγραμμα ακολουθίας με τις βασικές λειτουργίες της εφαρμογής.....	15
3 Αναλυτική περιγραφή υλοποίησης.....	18
3.1 Προγραμματισμός στο Android Studio.....	18
3.1.1 Activities.....	18
3.1.2 Fragments.....	19
3.1.3 Σχεδιασμός UI στο Android Studio.....	20
3.2 Σύνδεση και ταυτοποίηση χρήστη.....	20
3.3 Η κύρια οθόνη.....	26
3.3.1 Διαμοιρασμός λειτουργιών σε Fragments.....	26
3.3.2 Αρχική οθόνη.....	31
3.3.3 Συμπλήρωση προφίλ.....	33
3.3.4 Προβολή προτεινομένων σημείων ενδιαφέροντος.....	35
3.3.5 Χάρτης.....	37
3.4 Περιοχή με γενικές πληροφορίες για την Αθήνα.....	41
3.5 Περιοχή με τις διαθέσιμες τοποθεσίες της εφαρμογής.....	42



3.6	Περιοχή με τα αγαπημένα	43
3.7	Δεδομένα και βάση δεδομένων	44
3.7.1	Firebase Realtime Database	44
3.7.2	Models	46
3.7.3	Viewmodels	47
3.8	Υπολογισμός συντελεστή ομοιότητας	50
4	Εγχειρίδιο Χρήστη	54
4.1	Σύνδεση στην εφαρμογή	54
4.2	Άδεια χρήση της τοποθεσίας	55
4.3	Αρχική οθόνη	56
4.3.1	Γενικές πληροφορίες	56
4.3.2	Διαθέσιμα μέρη της εφαρμογής	57
4.3.3	Αγαπημένα	59
4.4	Δημιουργία προφίλ	60
4.5	Προτάσεις	61
4.6	Χάρτης	62
5	Συμπεράσματα	65
6	Βιβλιογραφικές Πηγές	66



ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Εικόνα 1. Διαδικασία σύνδεσης στην εφαρμογή.	14
Εικόνα 2. Διαδικασία συμπλήρωσης προφίλ και εμφάνιση προτάσεων.....	15
Εικόνα 3. Διάγραμμα ακολουθίας με βασικές λειτουργίες της εφαρμογής	16
Εικόνα 4. Κύκλος ζωής ενός Activity	19
Εικόνα 5. Οθόνη σύνδεσης.....	21
Εικόνα 6. Οθόνη HomeFragment	32
Εικόνα 7. Προφίλ χρήστη.....	34
Εικόνα 8. Οθόνη προτάσεων	36
Εικόνα 9. Χάρτης	38
Εικόνα 10. Γενικές πληροφορίες	41
Εικόνα 11. Τοποθεσίες ανά κατηγορία.	42
Εικόνα 12. Αγαπημένες τοποθεσίες.	43
Εικόνα 13. Realtime Database profiles	44
Εικόνα 14. Realtime Database τοποθεσίες.....	45
Εικόνα 15. Οθόνη σύνδεσης.....	54
Εικόνα 16. Μήνυμα για χρήση GPS.....	55
Εικόνα 17. Αρχική οθόνη	56
Εικόνα 18. Γενικές πληροφορίες για την Αθήνα.....	57
Εικόνα 19. Διαθέσιμα μέρη της εφαρμογής.....	58
Εικόνα 20. Πληροφορίες για τον προορισμό	59
Εικόνα 21. Αγαπημένα	60
Εικόνα 22. Προφίλ	61
Εικόνα 23. Προτάσεις.....	62
Εικόνα 24. Χάρτης	63

ΚΑΤΑΛΟΓΟΣ ΕΞΙΣΩΣΕΩΝ

Εξίσωση 1. Όριο ομοιότητας προφίλ χρήστη με τοποθεσία	50
Εξίσωση 2. Jaccard Coefficient.....	50



Κεφάλαιο 1^ο

1 Εισαγωγή

1.1 Περιγραφή του υπό μελέτη προβλήματος

Ένα από τα συνηθέστερα προβλήματα που αντιμετωπίζει ένας τουρίστας επισκεπτόμενος μια μεγαλούπολη, όπως η Αθήνα, είναι η δημιουργία ενός ημερήσιου προγράμματος επίσκεψης διάφορων σημείων ενδιαφέροντος. Παρόλο που στο διαδίκτυο υπάρχει διαθέσιμος μεγάλος όγκος πληροφορίας, ο ενδιαφερόμενος πρέπει να σπαταλήσει αρκετό χρόνο σε αναζήτηση. Μάλιστα, το γεγονός ότι η Αθήνα προσφέρει αμέτρητες επιλογές για κάθε είδους ενδιαφέροντος, δημιουργεί επιπλέον προβλήματα στην προσπάθειά του περιηγητή να επιλέξει μέρη που ταιριάζουν περισσότερο με την προσωπικότητά του. Άλλο ένα σημαντικό εμπόδιο είναι η αχανής έκταση της Αθήνας. Σε πολλές περιπτώσεις η απόσταση ανάμεσα σε δύο σημεία είναι μεγάλη και αναγκάζει τον επισκέπτη να ταξιδεύει για αρκετή ώρα. Για αυτό τον λόγο είναι προτιμότερο να επιλέγονται σημεία για επίσκεψη τα οποία είναι σχετικά κοντά μεταξύ τους. Άλλη μια παράμετρος που πρέπει να λάβει υπόψη του οποιοσδήποτε θέλει να οργανώσει μια περιήγηση στην Αθήνα είναι ο τρόπος μετακίνησης. Η Αθήνα είναι μια πόλη μεγάλη σε έκταση, αλλά συγχρόνως πυκνοκατοικημένη και με έντονο συγκοινωνιακό πρόβλημα. Αυτό επηρεάζει τον τρόπο με τον οποίο κάποιος μπορεί να μετακινηθεί σε ένα μέρος. Σε κάποια μέρη είναι προτιμότερο να μετακινηθείς με τα ΜΜΜ και σε άλλα με ιδιωτικό όχημα. Αναλογιζόμενοι τα προαναφερθέντα προβλήματα, αποφασίσαμε να δημιουργήσουμε μια εφαρμογή για το κινητό, μέσω της οποίας ο χρήστης θα μπορεί γρήγορα και εύκολα να βρίσκει μέρη τα οποία ταιριάζουν στα ενδιαφέροντα του και στις ανάγκες του απλά συμπληρώνοντας ένα σύντομο προφίλ.

1.2 Ανάλυση σχετικών εφαρμογών

Στο play store υπάρχουν διαθέσιμες μερικές εφαρμογές που είναι σχετικές με το θέμα της περιήγησης. Μια από αυτές είναι η Athens Guide by Civitatis, η οποία προσφέρει διάφορα πακέτα ξεναγήσεων για την Αθήνα. Παρόλα αυτά, οι επιλογές που παρέχει δεν είναι εξατομικευμένες ως προς τον χρήστη. Μπορείτε να κατεβάσετε την συγκεκριμένη εφαρμογή στον παρακάτω σύνδεσμο:

<https://play.google.com/store/apps/details?id=com.civitatis.atenas&hl=el&gl=US>

Το Athens Best Travel Tour Guide είναι άλλη μία σχετική εφαρμογή, η οποία μας παρέχει πληροφορίες για τα πιο γνωστά σημεία ενδιαφέροντος στην Αθήνα, για διαθέσιμα ξενοδοχεία και γενικές πληροφορίες για την πόλη. Όπως και η πρώτη η εφαρμογή, έτσι και αυτή δεν προσφέρει εξατομικευμένες προτάσεις. Μπορείτε να βρείτε την συγκεκριμένη εφαρμογή στο παρακάτω σύνδεσμο:

<https://play.google.com/store/apps/details?id=com.moolennyapp.athenstraveltourguide&hl=el&gl=US>



1.3 Σκοπός και στόχοι της εργασίας

Σκοπός της εργασίας είναι να αξιοποιήσουμε τις δυνατότητες που μας παρέχει το Android studio για να δημιουργήσουμε μια σύγχρονη εφαρμογή για android κινητά. Μέσω αυτού του εργαλείου προσπαθήσαμε να γράψουμε κώδικα σε γλώσσα Java, που να στηρίζεται στους κανόνες του αντικειμενοστραφούς προγραμματισμού και των προτύπων σχεδιασμού προγραμμάτων, ώστε να είναι όσο το δυνατόν πιο εύκολα συντηρήσιμος και προσιτός στις επεκτάσεις .

Όσον αφορά το τελικό προϊόν, πρωταρχικός σκοπός είναι η εφαρμογή μας να είναι εύχρηστη και απλή στην χρήση. Στόχος μας είναι η να χρησιμοποιείται από όλες τις ηλικιακές ομάδες και από χρήστες που μπορεί να μην είναι εξοικειωμένοι με την χρήση του κινητού. Ο χρήστης πρέπει να συμπληρώνει ένα σύντομο προφίλ, το οποίο πρέπει να ζητάει τα απαιτούμενα στοιχεία ώστε η εφαρμογή να εμφανίζει αποτελέσματα που τον ικανοποιούν. Επίσης, πολύ σημαντικό είναι να εντοπίζονται ορισμένες δυσκολίες που μπορεί να αναδεικνύονται από το προφίλ του χρήστη ώστε οι προορισμοί που του προτείνονται να ανταποκρίνονται στις προσωπικές του ανάγκες. Για παράδειγμα, ένας χρήστης που έχει μικρά παιδιά και κουβαλάει καροτσάκι, χρειάζεται μέρη που είναι ευκόλως προσβάσιμα. Για κάθε σημείο ενδιαφέροντος ,πρέπει να παρέχονται κάποιες βασικές πληροφορίες ώστε ο χρήστης να διαμορφώνει μια βασική εικόνα για αυτό. Επιπλέον πρέπει η εφαρμογή να διαθέτει χάρτη, ώστε να βοηθήσει τον χρήστη να πλοηγηθεί μέσα στην πόλη και να φτάσει στους προορισμούς που του προτείνονται. Τέλος, ο χρήστης πρέπει να μπορεί να συνδεθεί εύκολα και γρήγορα χρησιμοποιώντας τους λογαριασμούς του από το Facebook και το Google+,ή ακόμα και χωρίς λογαριασμό ως απλός επισκέπτης.

1.4 Βασικοί ορισμοί

Παρακάτω θα αναλυθούν κάποιο βασικοί ορισμοί, που είναι αναγκαίοι για την κατανόηση της της παρούσας εργασίας.

1.4.1 Android OS

Το Android είναι ένα λειτουργικό σύστημα που χρησιμοποιεί τον πυρήνα του Linux σχεδιασμένο για κινητές συσκευές με οθόνες αφής όπως έξυπνα κινητά (smartphones) και tablets.Ακριβώς ,δηλαδή, ότι είναι τα Windows, τα Macintosh και τα Linux για τους ηλεκτρονικούς υπολογιστές. Το όνομα του είναι ελληνικής προέλευσης και σημαίνει ανδροειδές, δηλαδή ανθρωπόμορφο ρομπότ. Επειδή αρχικά σχεδιάστηκε για να χρησιμοποιηθεί σε συσκευές με οθόνες αφής, περιλαμβάνει ένα μεγάλο αριθμό αντίστοιχων λειτουργιών συμπεριλαμβανομένου και εικονικού πληκτρολογίου. Επίσης υποστηρίζει υπηρεσίες φωνής, δηλαδή τηλεφωνικές κλήσεις, υπηρεσίες σύντομων μηνυμάτων SMS και αποστολή μηνυμάτων πολυμέσων MMS .Πλέον χρησιμοποιείται και σε πλήθος άλλων συσκευών όπως έξυπνες τηλεοράσεις, κονσόλες και ψηφιακές μηχανές.

1.4.2 Αντικειμενοστραφής προγραμματισμός.

Ο αντικειμενοστραφής προγραμματισμός εμφανίστηκε κατά την διάρκεια της δεκαετίας του 1960 ως τρόπο για την επίλυση των προβλημάτων που προκύπταν από τον δομημένο προγραμματισμό. Συγκεκριμένα «ο αντικειμενοστρεφής προγραμματισμός είναι ένας τρόπος οργάνωσης προγραμμάτων. Ο αντικειμενοστρεφής προγραμματισμός έχει να κάνει με το πώς



τα προγράμματα σχεδιάζονται, όχι με τις λεπτομέρειες των μεμονωμένων δηλώσεων προγράμματος. Ειδικότερα, τα αντικειμενοστρεφή προγράμματα οργανώνονται γύρω από τα αντικείμενα, τα οποία περιέχουν και τα στοιχεία και τις λειτουργίες. Μια κλάση είναι η προδιαγραφή για ένα αριθμό παρόμοιων αντικειμένων»¹. Με άλλα λόγια, στον αντικειμενοστρεφή προγραμματισμό, τα προγράμματα οργανώνονται σε κλάσεις, οι οποίες αντιπροσωπεύουν διάφορα αντικείμενα, όπως ένα αυτοκίνητο ή έναν μαθητή. Με την σειρά τους οι κλάσεις αποτελούνται από μεταβλητές (attributes), οι οποίες περιγράφουν τα χαρακτηριστικά ενός αντικειμένου (όπως η μάζα του), και από τις μεθόδους (ή και συναρτήσεις), η οποίες εκτελούν τις λειτουργίες που θέλουμε να εκτελέσει αυτό το αντικείμενο.

1.4.3 Γλώσσα προγραμματισμού Java

Η Java είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού. Είναι μια από τις γλώσσες που χρησιμοποιεί το android studio και αυτή με την οποία αναπτύξαμε την εφαρμογή μας.

1.4.4 Android Studio

Το android studio είναι ένα σύγχρονο προγραμματιστικό περιβάλλον για την ανάπτυξη εφαρμογών για λειτουργικό σύστημα Android. Οι πιο δημοφιλείς γλώσσες που χρησιμοποιούνται για την ανάπτυξη μια εφαρμογής είναι η Java και η Kotlin.

1.4.5 Firebase

Το Firebase είναι ένα σύνολο εργαλείων που αναπτύχθηκε από την Google για την κατασκευή, βελτίωση και ανάπτυξη της εφαρμογών. Τα εργαλεία που παρέχει καλύπτουν ένα μεγάλο μέρος των υπηρεσιών που κανονικά οι προγραμματιστές θα έπρεπε να φτιάξουν μόνοι τους. Αυτό περιλαμβάνει πράγματα όπως στατιστικά δεδομένα της εφαρμογής, ταυτοποίηση, βάσεις δεδομένων και αποθήκευση αρχείων. Οι υπηρεσίες φιλοξενούνται στο cloud και είναι αρκετά εύκολο για τον προγραμματιστή να τις ενσωματώσει στην εφαρμογή του.

1.4.6 API

Το API είναι η συντομογραφία του Application Programming Interface, και αποτελείται από μια σειρά κανόνων που καθορίζουν τον τρόπο λειτουργίας του. Με άλλα λόγια, είναι ένα μεσάζων εργαλείο πληροφόρησης. Τα APIs επιτρέπουν σε μια εφαρμογή να εξάγει πληροφορίες από ένα κομμάτι λογισμικού και να τις χρησιμοποιήσει αυτούσιες ή να τις αναλύσει.

1.4.7 XML

Τα έγγραφα XML περιέχουν δεδομένα που περικλείονται από ετικέτες, οι οποίες ορίζουν τη σημασία των δεδομένων. Με την χρήση ετικετών η XML μας δίνει την δυνατότητα επαναχρησιμοποίησης των δεδομένων με διάφορους τρόπους. Η XML μας επιτρέπει να δημιουργήσουμε οποιαδήποτε ετικέτα χρειαζόμαστε για να περιγράψουμε τα δεδομένα μας και τη δομή αυτών των δεδομένων ώστε να αναγνωρίζουμε το είδος των δεδομένων πολύ εύκολα. Η XML δεν εξαρτάται επίσης από την εκάστοτε πλατφόρμα, που σημαίνει ότι οποιοδήποτε πρόγραμμα που έχει σχεδιαστεί για να χρησιμοποιεί την XML μπορεί να διαβάσει και να επεξεργαστεί τα δεδομένα XML, ανεξάρτητα από το υλικό ή το λειτουργικό σύστημα. Στο

¹ Παναγιωτόπουλος, Ι.-Χ., Αποστόλου, Δ.(2012), Αρχές Προγραμματισμού με C/C++. Πειραιάς: Αυτοέκδοση, σ. 62



android studio τα αρχεία xml χρησιμοποιούνται για την περιγραφή των γραφικών της εφαρμογής.

1.5 Παραδοτέα της εργασίας

Στα παραδοτέα της εργασίας περιλαμβάνονται τα αρχεία πηγαίου κώδικα της εφαρμογής , το έντυπο κείμενο της πτυχιακής εργασίας, το οποίο περιλαμβάνει την επισκόπηση της σχετικής με το χώρο βιβλιογραφίας, ένα βίντεο με τις λειτουργίες της εφαρμογής καθώς και ένα αρχείο APK για την εγκατάσταση της εφαρμογής σε κινητή συσκευή. Ο πηγαίος κώδικας μπορεί να βρεθεί και στο GitHub στον παρακάτω σύνδεσμο:

<https://github.com/vaggos99/MyTourGuide.git>





Κεφάλαιο 2^ο

2 Ανάλυση και σχεδίαση της εφαρμογής

Σε αυτό το κεφάλαιο θα παρουσιαστεί το στάδιο της ανάλυσης και της σχεδίασης της εφαρμογής με την βοήθεια δύο ειδών διαγραμμάτων UML. Συγκεκριμένα θα παρουσιαστούν τα διαγράμματα περίπτωσης χρήσης (Use Case) και ακολουθίας (Sequence).

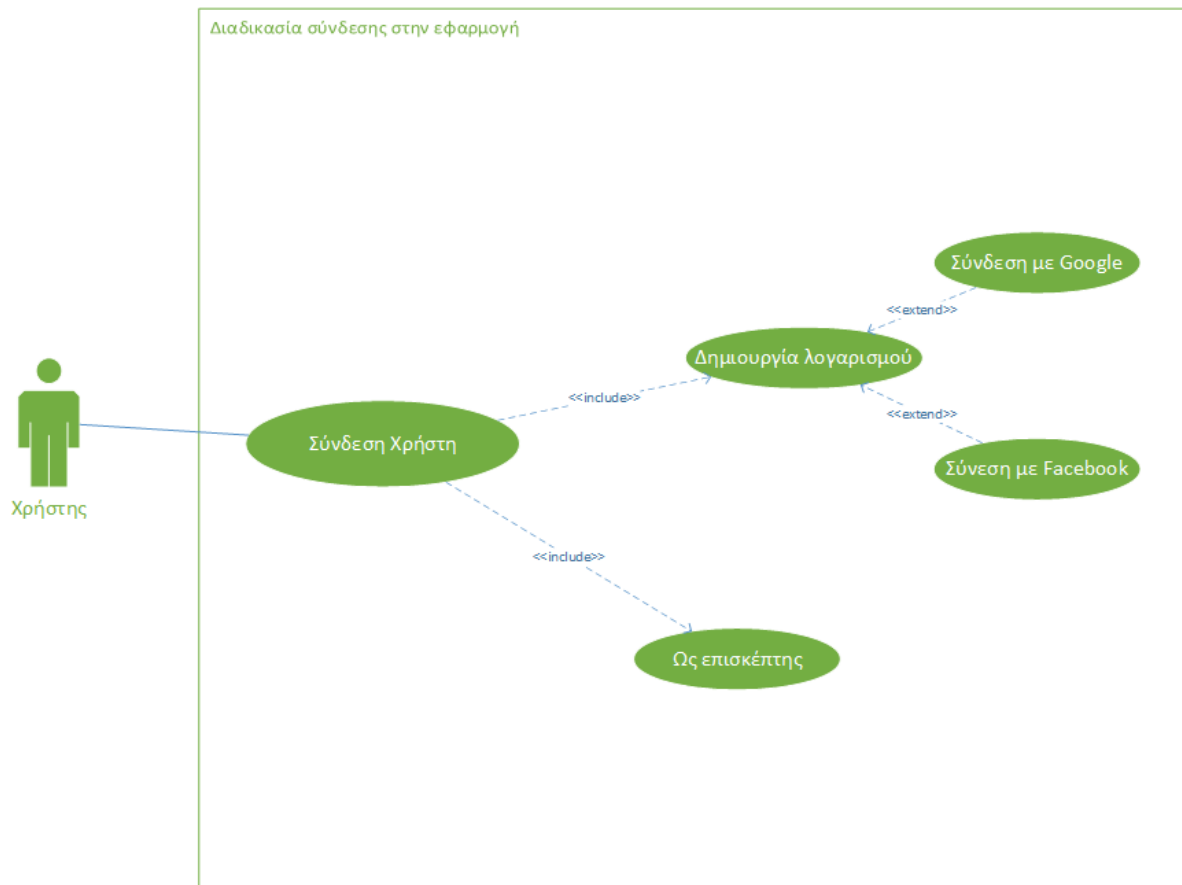
2.1 Διαγράμματα περίπτωσης χρήσης (Use Case Diagrams)

Τα διαγράμματα περίπτωσης χρήσης (Use Case) περιγράφουν τον τρόπο που αλληλοεπιδρά ο χρήστης με το σύστημα. Τα βασικά στοιχεία ενός use case διαγράμματος είναι το σύστημα, οι χρήστες (actors), και οι περιπτώσεις χρήσης. Συνδέοντας αυτές τις οντότητες στο διάγραμμα, βγάζουμε βασικά συμπεράσματα σχετικά με το σύστημα που θέλουμε να αναπτύξουμε όπως, τι είδους σύστημα φτιάχνουμε, ποιοι χρησιμοποιούν το σύστημα και τι θέλουν να πετύχουν μέσα από την χρήση του (ποιες είναι οι περιπτώσεις χρήσης). Σε ένα διάγραμμα περιπτώσεων χρήσης παρουσιάζονται οι σχέσεις τόσο μεταξύ του χρήστη και των περιπτώσεων χρήσης, όσο και μεταξύ των περιπτώσεων χρήσης. Υπάρχουν τρία είδη συσχέτισης:

- Το «include» υποδηλώνει ότι μια λειτουργία εκτελείται κάθε φορά που εκτελείται μια άλλη.
- Το «extend» υποδηλώνει ότι μια λειτουργία μπορεί να εκτελεσθεί ως επακόλουθο μιας άλλης
- Η γενική συσχέτιση που αντιπροσωπεύεται από μια συνεχόμενη γραμμή.

2.1.1 Σύνδεση χρήστη στην εφαρμογή

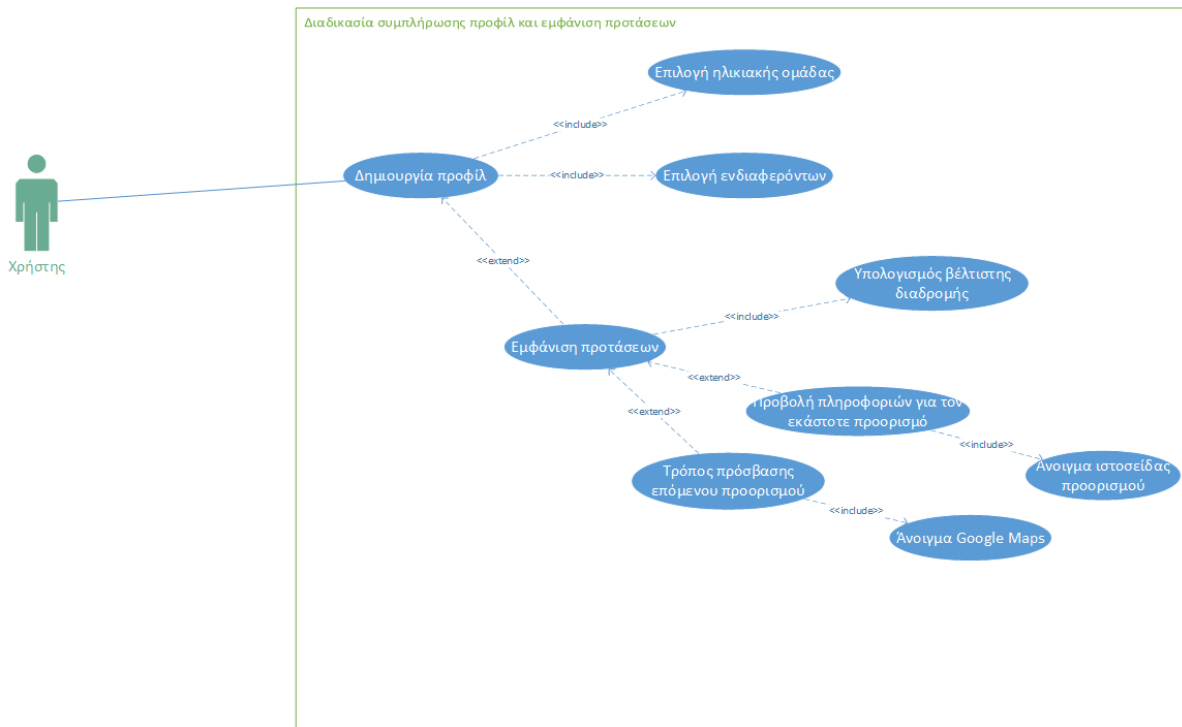
Το παρακάτω διάγραμμα μας παρουσιάζει την διαδικασία σύνδεσης του χρήστη στην εφαρμογή. Όπως βλέπουμε ο χρήστης μπορεί είτε να δημιουργήσει λογαριασμό και κατ' επέκταση να συνδεθεί με Facebook ή Google, είτε να συνδεθεί σαν επισκέπτης.



Εικόνα 1. Διαδικασία σύνδεσης στην εφαρμογή.

2.1.2 Διαδικασία συμπλήρωσης προφίλ και εμφάνιση προτάσεων

Το παρακάτω διάγραμμα μας παρουσιάζει την διαδικασία δημιουργίας προφίλ και την εμφάνιση των προτάσεων από την εφαρμογή στον χρήστη. Ο χρήστης για να δημιουργήσει το προφίλ του, πρέπει να επιλέξει την ηλικιακή του ομάδα και τα ενδιαφέροντα του. Αφού συμπληρώσει το προφίλ του μπορεί να δει τι προορισμούς του προτείνει η εφαρμογή. Η διαδικασία εμφάνισης των προτάσεων περιλαμβάνει και τον υπολογισμό των προορισμών που ταιριάζουν στο προφίλ του χρήστη. Επιπρόσθετα ο χρήστης μπορεί να δει πληροφορίες σχετικά με τον προορισμό που του προτείνεται καθώς και να βρει οδηγίες για το πώς να πάει εκεί μέσω του Google Maps.



Εικόνα 2. Διαδικασία συμπλήρωσης προφίλ και εμφάνιση προτάσεων

2.2 Διαγράμματα ακολουθίας (Sequence Diagrams)

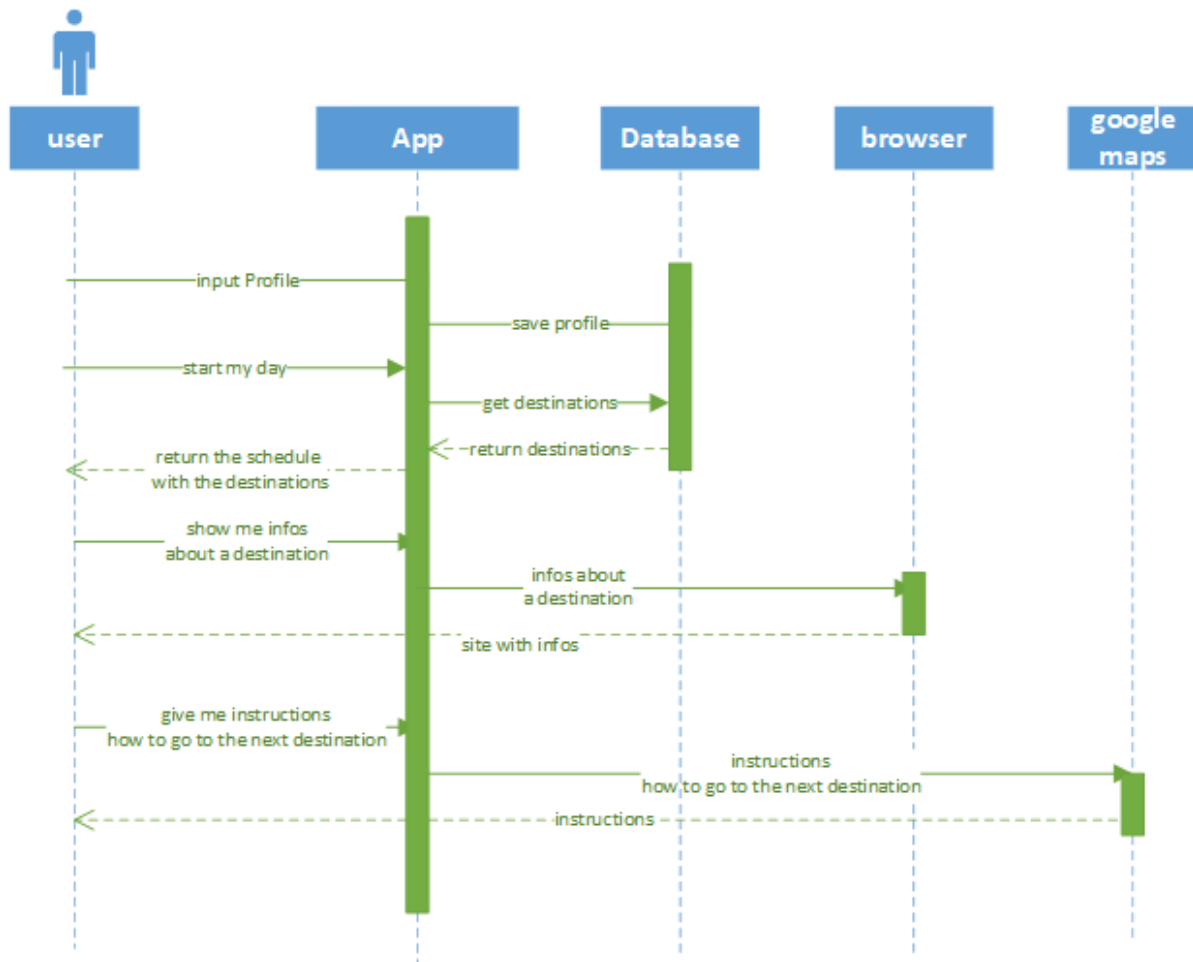
Τα διαγράμματα ακολουθίας (Sequence Diagrams) είναι διαγράμματα αλληλεπίδρασης που περιγράφουν λεπτομερώς τον τρόπο διεξαγωγής των λειτουργιών. Καταγράφουν την αλληλεπίδραση μεταξύ αντικειμένων στο πλαίσιο μιας συνεργασίας. Μέσα από αυτά τα διαγράμματα μπορούμε να δούμε τη σειρά της αλληλεπίδρασης οπτικά χρησιμοποιώντας τον κατακόρυφο άξονα του διαγράμματος, για να απεικονίσουμε τον χρόνο, ποια μηνύματα αποστέλλονται, από ποιον και πότε. Πιο συγκεκριμένα, στον κάθετο άξονα αναπαρίσταται η διάρκεια ζωής του κάθε αντικειμένου, ενώ τα αντικείμενα επικοινωνούν μεταξύ τους με οριζόντια βέλη που συμβολίζουν τα μηνύματα που ανταλλάσσουν. Τα βέλη αυτά είναι ταξινομημένα χρονολογικά ώστε να ξεχωρίζεται η σειρά με την οποία εμφανίζονται. Τα μηνύματα που αποτελούν κάποιο αίτημα αναπαρίστανται με βέλη με συνεχή γραμμές, ενώ τα μηνύματα που αποτελούν απαντήσεις αναπαρίστανται με βέλη με διακεκομμένες γραμμές.

2.2.1 Διάγραμμα ακολουθίας με τις βασικές λειτουργίες της εφαρμογής

Στο παρακάτω διάγραμμα απεικονίζεται ο τρόπος αλληλεπίδρασης του χρήστη με το σύστημα της εφαρμογής και της εφαρμογής με άλλα αντικείμενα. Όταν ο χρήστης δώσει στην εφαρμογή το προφίλ του, αυτή με την σειρά της επικοινωνεί με την βάση δεδομένων και αποθηκεύει τα στοιχεία του

Όταν ο χρήστης θέλει να ξεκινήσει τον βόλτα του, στέλνει το αντίστοιχο μήνυμα στην εφαρμογή. Η εφαρμογή στέλνει αίτημα στην βάση δεδομένων για να λάβει τους διαθέσιμους προορισμούς και αυτή ανταποκρίνεται στέλνοντάς τους. Αφού η εφαρμογή φιλτράρει την λίστα με τους προορισμούς βάση το προφίλ του χρήστη, επιστρέφει ένα πρόγραμμα διαδρομής στον χρήστη.

Αν ο χρήστης θέλει να δει πληροφορίες για έναν προορισμό τότε η εφαρμογή ανοίγει τον browser της συσκευής και παρουσιάζει την κατάλληλη ιστοσελίδα. Αν ο χρήστης θέλει να δει τον τρόπο πρόσβασης σε έναν προορισμό, η εφαρμογή τον ανακατευθύνει στο Google Maps που με την σειρά του του δίνει οδηγίες.



Εικόνα 3. Διάγραμμα ακολουθίας με βασικές λειτουργίες της εφαρμογής





Κεφάλαιο 3^ο

3 Αναλυτική περιγραφή υλοποίησης

Σε αυτό το κεφάλαιο θα αναλυθεί ο τρόπος υλοποίησης των λειτουργιών τις εφαρμογής. Πρώτα όμως θα μιλήσουμε λίγο περισσότερο για κάποιες βασικές έννοιες που κάποιος συναντά όταν ασχολείται με προγραμματισμό στο Android Studio.

3.1 Προγραμματισμός στο Android Studio

Το Android Studio είναι ένα σύγχρονο περιβάλλον ανάπτυξης εφαρμογών για android συσκευές. Σε αυτή την ενότητα θα δούμε τα βασικά στοιχεία από τα οποία αποτελείται κάθε εφαρμογή που αναπτύσσεται σε περιβάλλον Android Studio και που είναι απαραίτητα για την κατανόησή των παρακάτω εννοιών.

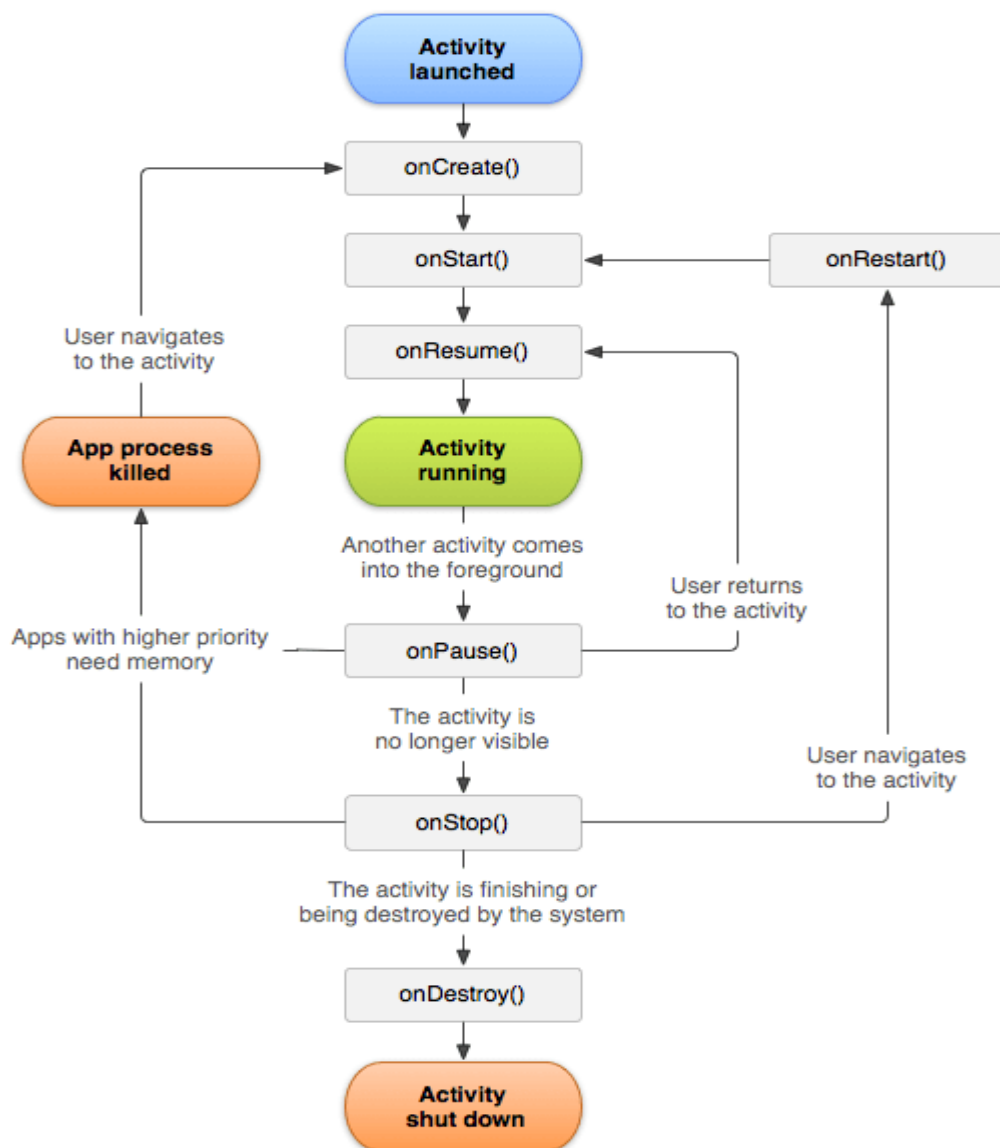
3.1.1 Activities

Τα activities είναι το πιο σημαντικό κομμάτι μιας εφαρμογής android. Αν θέλαμε να είμαστε αρκετά απλοϊκοί θα μπορούσαμε να παρουσιάσουμε ένα activity με την μέθοδο main , η οποία ξεκινά και τρέχει τις λειτουργίες ενός προγράμματος. Με παρόμοια τρόπο, σε ένα σύστημα Android μια εφαρμογή ξεκινάει μέσα από ένα activity, το οποίο ενεργοποιείται μέσω της μεθόδου onCreate .

Ένα activity παρέχει το παράθυρο στο οποίο η εφαρμογή σχεδιάζει το γραφικό περιβάλλον της (User Interface). Αυτό το παράθυρο συνήθως γεμίζει την οθόνη, αλλά μπορεί να είναι και μικρότερο και να επιπλέει πάνω από άλλα παράθυρα. Γενικά, ένα activity υλοποιεί μία οθόνη σε μια εφαρμογή. Για παράδειγμα, ένα από τα activity μιας εφαρμογής μπορεί να υλοποιεί μια λίστα με email του χρήστη, ενώ μια άλλη δραστηριότητα υλοποιεί ένα UI για την αναπαραγωγή τραγουδιών.

Οι περισσότερες εφαρμογές περιέχουν πολλαπλές οθόνες, πράγμα που σημαίνει ότι περιλαμβάνουν πολλαπλά activity. Συνήθως ένα από τα activity ορίζεται ως το κύριο activity (main activity), δηλαδή αυτό που εμφανίζεται όταν ο χρήστης εκκινεί την εφαρμογή. Κάθε activity μπορεί στη συνέχεια να ξεκινήσει ένα άλλο για να εκτελέσει διαφορετικές ενέργειες. Για παράδειγμα, το main activity σε μια απλή εφαρμογή ηλεκτρονικού ταχυδρομείου μπορεί να παρέχει το γραφικό περιβάλλον που εμφανίζει τα εισερχόμενα μηνύματα. Από εκεί, μπορεί να ξεκινήσει άλλα activities που παρέχουν άλλες λειτουργίες για εργασίες όπως η σύνταξη μηνυμάτων ηλεκτρονικού ταχυδρομείου και το άνοιγμα μεμονωμένων μηνυμάτων ηλεκτρονικού ταχυδρομείου.

Το κάθε activity έχει τον δικό του κύκλο ζωής, του οποίου τα στάδια αντιπροσωπεύονται και από μια μέθοδο. Αυτό φαίνεται και στην επόμενη εικόνα.



Εικόνα 4. Κύκλος ζωής ενός Activity

3.1.2 Fragments

Τα fragments είναι μια πιο μικρή έκδοση των activities. Ομοίως με τα activities, τα fragments παρέχουν το UI για διάφορες λειτουργίες της εφαρμογής. Ένα σημαντικό πλεονέκτημα των fragments είναι ότι μπορούν να επαναχρησιμοποιηθούν σε πολλά σημεία μέσα στην εφαρμογή. Παρόλο που ένα fragment έχει τον δικό του κύκλο ζωής παρόμοιο με τα activities, πρέπει αναγκαστικά να ανήκει σε κάποιο activity ή σε ένα άλλο fragment. Κάθε fragment περιέχει την μέθοδο `onCreateView` η οποία εκτελείται όταν δημιουργείται.



3.1.3 Σχεδιασμός UI στο Android Studio

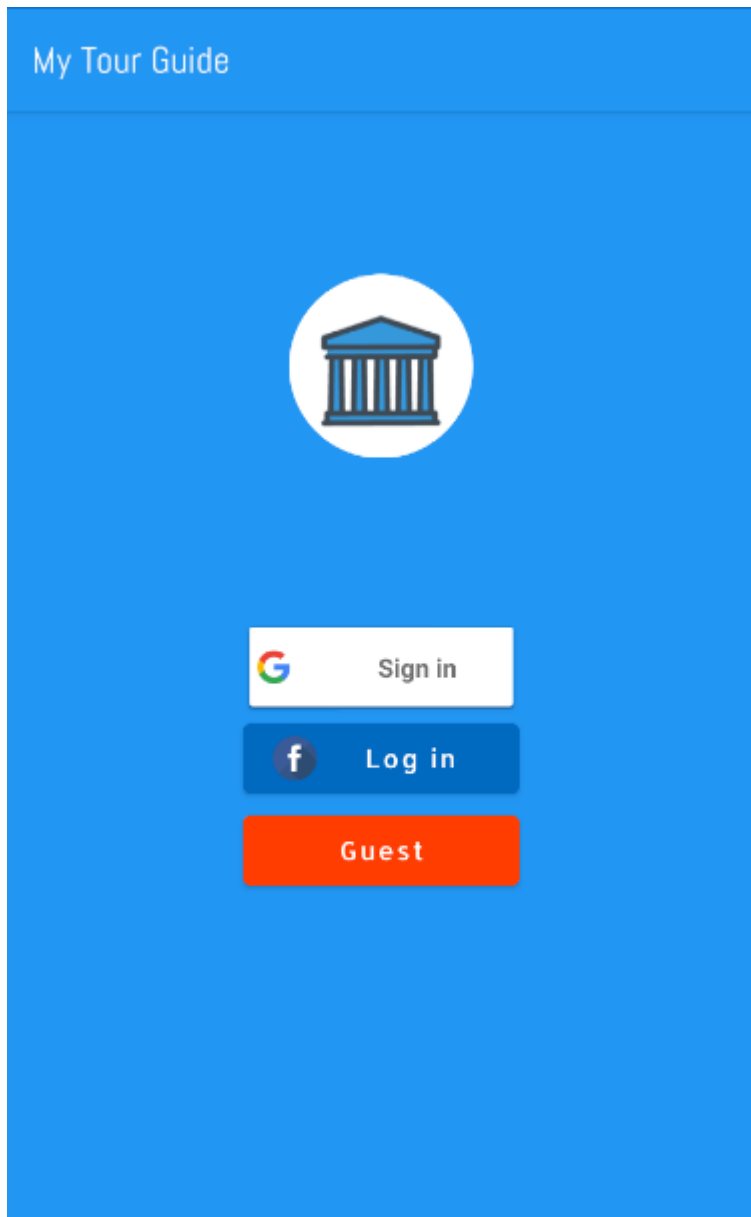
Στο android studio το user interface σχεδιάζεται με xml αρχεία. Κάθε activity και fragment αντιστοιχίζεται σε ένα xml αρχείο. Μέσα σε αυτό με την μορφή ετικετών xml περιγράφονται τα διάφορα συστατικά (views) που απαρτίζουν το γραφικό περιβάλλον της εφαρμογής, όπως κουμπιά (buttons), πλαίσια κειμένου (textViews) ή και ολόκληρα άλλα αρχεία UI όπως ένα fragment. Φυσικά μπορούμε να έχουμε πρόσβαση σε όλα αυτά τα αντικείμενα και μέσα από τα activities με την χρήση των αντίστοιχων κλάσεων

3.2 Σύνδεση και ταυτοποίηση χρήστη

Για την σύνδεση και ταυτοποίηση (authentication) του χρήστη χρησιμοποιήσαμε το σύστημα αυθεντικοποίησης που μας παρέχει το εργαλείο Firebase. Από τους πολλούς τρόπους αυθεντικοποίησης που υπάρχουν διαθέσιμοι στο Firebase εμείς διαλέξαμε τους εξής:

- **Google:** Οι περισσότεροι χρήστες Android συσκευών έχουν λογαριασμό Google, με τον οποίο μάλιστα συνδέουν τις περισσότερες λειτουργίες και εφαρμογές του κινητού τους. Η ευρεία χρήση καθιστά την Google ως τον καταλληλότερο πάροχο για android εφαρμογές.
- **Facebook:** Το Facebook είναι το γνωστότερο μέσο κοινωνικής δικτύωσης στο διαδίκτυο.
- **Επισκέπτης:** Το Firebase μας δίνει την δυνατότητα να φτιάχνουμε προσωρινούς λογαριασμούς μιας χρήσης. Η σύνδεση είναι ανώνυμη χωρίς να απαιτείται καμία ταυτοποίηση από τον χρήστη.

Με τις δύο πρώτους μεθόδους ο χρήστης δημιουργεί έναν μόνιμο λογαριασμό και τα δεδομένα του από την χρήση της εφαρμογής παραμένουν αποθηκευμένα στην Realtime Database της firebase. Αν συνδεθεί ως επισκέπτης ο χρήστης δημιουργεί έναν προσωρινό λογαριασμό ο οποίος διαγράφεται (όπως και ότι δεδομένα έχουν προκύψει από την χρήση της εφαρμογής) όταν αποσυνδεθεί.



Εικόνα 5. Οθόνη σύνδεσης

Η οθόνη σύνδεσης υλοποιείται από το LoginActivity. Παρακάτω παρατίθεται ο κώδικας της συνάρτησης onCreate η οποία καλείται όταν δημιουργείται το activity.



```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);
    // Initialize Google Login button
    GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestIdToken("785495160839-145qu8hafgrp7ald1q3qggbb1b319tqnq.apps.goog...")
        .requestEmail()
        .build();
    mGoogleSignInClient = GoogleSignIn.getClient( activity: this, gso);
    mAuth=FirebaseAuth.getInstance();

    findViewById(R.id.sign_in_button).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { signIn(); }
    });

    // Initialize Facebook Login button

    mCallbackManager = CallbackManager.Factory.create();

    LoginManager.getInstance().registerCallback(mCallbackManager,
        new FacebookCallback<LoginResult>() {
            @Override
            public void onSuccess(LoginResult loginResult) {
                Log.d( tag: TAG+"Success", msg: "Login");
                handleFacebookAccessToken(loginResult.getAccessToken());
            }

            @Override
            public void onCancel() { Log.d(TAG, msg: "onCancel"); }

            @Override
            public void onError(FacebookException exception) { Log.d(TAG, msg: "onError"); }
        });
};
```



```
findViewById(R.id.login_button_fb).setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        List<String> permissions=new ArrayList<>();  
        permissions.add("email");  
        permissions.add("public_profile");  
        LoginManager.getInstance().loginWithReadPermissions( activity, LoginActivity.this,permissions);  
    }  
});  
  
findViewById(R.id.guest_button).setOnClickListener(v->{  
    mAuth.signInAnonymously()  
        .addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {  
            @Override  
            public void onComplete(@NonNull Task<AuthResult> task) {  
                if (task.isSuccessful()) {  
                    // Sign in success, update UI with the signed-in user's information  
                    Log.d(TAG, msg: "signInAnonymously:success");  
  
                    updateUI();  
                } else {  
                    // If sign in fails, display a message to the user.  
                    Log.w(TAG, msg: "signInAnonymously:failure", task.getException());  
                    Toast.makeText( context: LoginActivity.this, text: "Authentication failed.",  
                                Toast.LENGTH_SHORT).show();  
                }  
            }  
        });  
});  
}
```

Όπως βλέπουμε πρώτα αρχικοποιούμε την σύνδεση με το API της Google για την σύνδεση με τον λογαριασμό της και καθορίζουμε τις λειτουργίες που πρέπει να ακολουθηθούν όταν πατηθεί το αντίστοιχο κουμπί. Το ίδιο γίνεται και για την σύνδεση με Facebook και για την σύνδεση ως επισκέπτης. Για την σύνδεση ως επισκέπτης δεν χρειάζεται η επικοινωνία με API κάποιου



εξωτερικού παρόχου, όπως δηλαδή γίνεται με την σύνδεση Google και Facebook. Η σύνδεση γίνεται κατευθείαν μέσω firebase.

Η μέθοδος `signIn` καλείται όταν το κουμπί για την σύνδεση με Google πατηθεί.

```
private void signIn() {  
    Intent signInIntent = mGoogleSignInClient.getSignInIntent();  
    startActivityForResult(signInIntent, RC_GOOGLE_SIGN_IN);  
}
```

Από αυτή την μέθοδο καλείται η μέθοδος `onActivityResult`, η οποία επεξεργάζεται τα αποτελέσματα από την αίτηση για σύνδεση με Google. Ο κώδικας της είναι ο εξής:

```
@Override  
public void onActivityResult(int requestCode, int resultCode, Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
  
    // Result returned from launching the Intent from GoogleSignInApi.getSignInIntent(...);  
    if (requestCode == RC_GOOGLE_SIGN_IN) {  
        Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);  
        Exception exception=task.getException();  
        if(task.isSuccessful()){  
            try {  
                // Google Sign In was successful, authenticate with Firebase  
                GoogleSignInAccount account = task.getResult(ApiException.class);  
                Log.d( tag: "LoginActivity", msg: "firebaseAuthWithGoogle:" + account.getId());  
                firebaseAuthWithGoogle(account.getIdToken());  
            } catch (ApiException e) {  
                // Google Sign In failed, update UI appropriately  
                Log.w( tag: "LoginActivity", msg: "Google sign in failed", e);  
                // ...  
            }  
        }  
        else  
            Log.w( tag: "LoginActivity", exception.toString());  
    }  
    else mCallbackManager.onActivityResult(requestCode, resultCode, data);  
}
```

Σε περίπτωση που το αίτημα σύνδεσης γίνει αποδεκτό, καλείται η μέθοδος `firebaseAuthWithGoogle`, η οποία μας συνδέει με την υπηρεσία authentication της firebase και ολοκληρώνει την σύνδεση μέσω Google. Ο κώδικας της παρατίθεται παρακάτω.



```
private void firebaseAuthWithGoogle(String idToken) {
    AuthCredential credential = GoogleAuthProvider.getCredential(idToken, accessToken: null);
    mAuth.signInWithCredential(credential)
        .addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    // Sign in success, update UI with the signed-in user's information
                    Log.d( tag: "LoginActivity", msg: "signInWithCredential:success");

                    updateUI();

                } else {
                    // If sign in fails, display a message to the user.
                    Log.w( tag: "LoginActivity", msg: "signInWithCredential:failure", task.getException());
                }
            }
        });
}
```

Η σύνδεση μέσω Facebook γίνεται μέσω της μεθόδου `handleFacebookAccessToken` η οποία αφού συνδεθεί με το API της Facebook, επικοινωνεί με την Firebase και ολοκληρώνει την σύνδεση. Ο κώδικας της παρατίθεται παρακάτω.

```
private void handleFacebookAccessToken(AccessToken token) {
    Log.d( tag: "LoginActivity", msg: "handleFacebookAccessToken:" + token);

    AuthCredential credential = FacebookAuthProvider.getCredential(token.getToken());
    mAuth.signInWithCredential(credential)
        .addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    // Sign in success, update UI with the signed-in user's information
                    Log.d( tag: "LoginActivity", msg: "signInWithCredential:success");

                    updateUI();

                } else {
                    // If sign in fails, display a message to the user.
                    Log.w( tag: "LoginActivity", msg: "signInWithCredential:failure", task.getException());
                }
            }
        });
}
```

Τέλος με την μέθοδο `updateUI` μεταφερόμαστε την κύρια οθόνη της εφαρμογής.

```
private void updateUI(){
    Intent intent = new Intent(getApplicationContext(), MainActivity.class);
    startActivity(intent);
    finish();
}
```

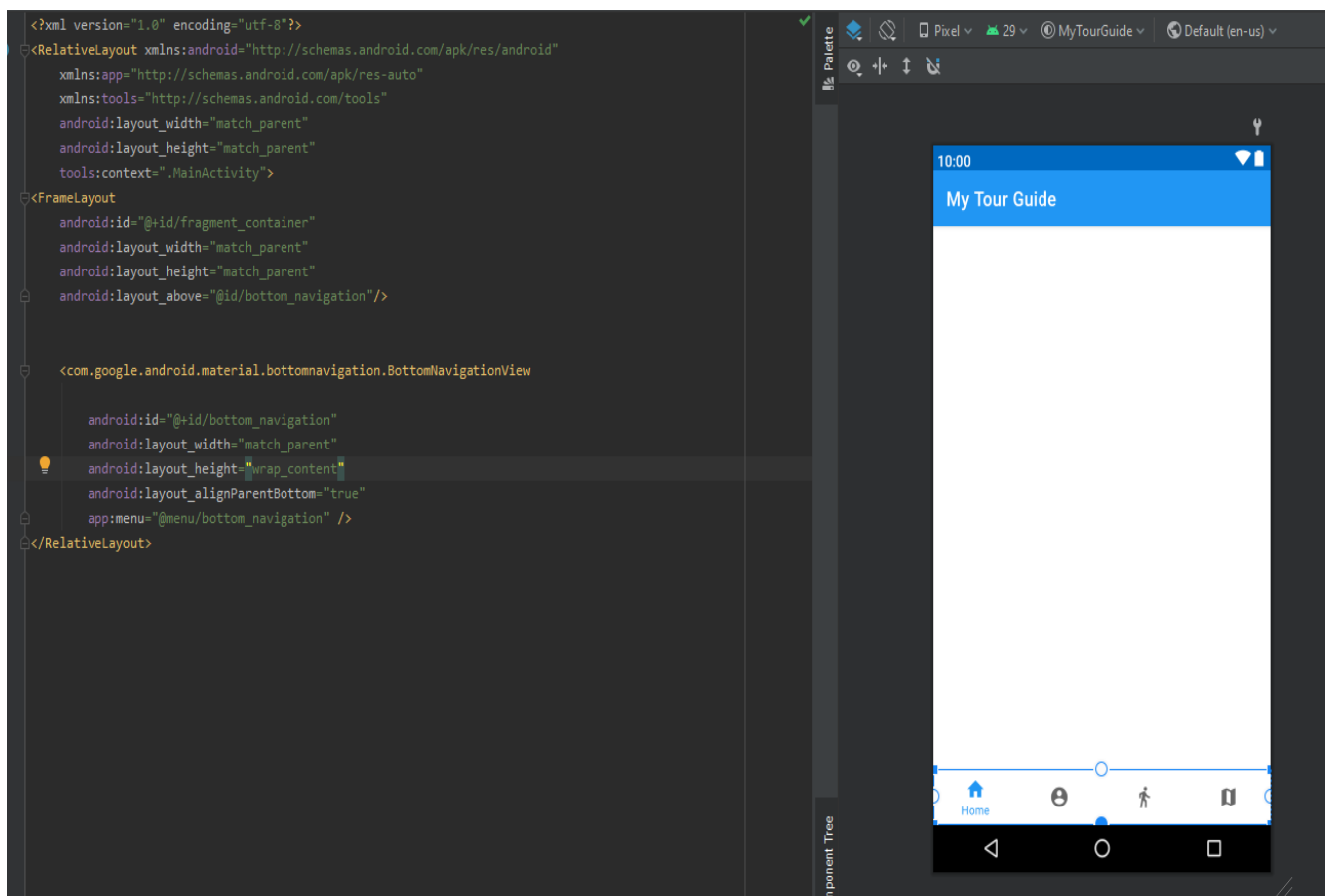
3.3 Η κύρια οθόνη

Η κύρια οθόνη της εφαρμογής υλοποιείται από το MainActivity.

3.3.1 Διαμοιρασμός λειτουργιών σε Fragments

Επειδή οι λειτουργίες που χρειάζεται να υλοποιήσει είναι πάρα πολλές για να υλοποιηθούν μόνο από μία κλάση, αλλά και το γραφικό περιβάλλον που απαιτείται δεν γίνεται να χωρέσει μέσα σε μία μόνο οθόνη, οι λειτουργίες του activity μοιράστηκαν σε τέσσερα fragments.

Για να καταλάβουμε καλύτερα τι εννοούμε με τον διαμερισμό λειτουργιών, ας ρίξουμε μια ματιά στο αρχείο activity_main.xml, το οποίο περιέχει το γραφικό περιβάλλον του MainActivity.



Στο αριστερό μέρος φαίνεται ο κώδικας και στο δεξιό πως μεταφράζεται γραφικά. Όπως βλέπουμε δεν υπάρχουν και πολλά στοιχεία. Για την ακρίβεια υπάρχουν τρία:



- **το RelativeLayout:** το οποίο δηλώνει τον τρόπο οργάνωσης των άλλων 2 στοιχείων μέσα στην οθόνη. Μέσα σε αυτή την ετικέτα δηλώνονται τα στοιχεία που απαρτίζουν την οθόνη. Τέτοιες είδους ετικέτες υπάρχουν σε κάθε αρχείο xml στο android studio.
- **το framelayout:** μέσα το οποίο θα εμφανίζονται τα τέσσερα fragments στα οποία αναφερθήκαμε προηγουμένως.
- **το BottomNavigationView:** που όπως δηλώνει η ονομασία του χρησιμοποιείται για την πλοήγηση σε αυτά τα fragments.

Τώρα ας δούμε την κλάση του MainActivity. Πρώτα ξεκινάμε από την μέθοδο onCreate, ο κώδικας της οποίας φαίνεται στην επόμενο πλαίσιο.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    if (user == null)
        logout();
    mFusedClient = LocationServices.getFusedLocationProviderClient( activity.this);
    viewModel = new ViewModelProvider( owner: this).get(ProfilesViewModel.class);
    LocationManager manager = (LocationManager) this. getSystemService(Context. LOCATION_SERVICE);
    if(getLocationPermission()){
        if(savedInstanceState==null && !manager.isProviderEnabled( LocationManager.GPS_PROVIDER ) &&viewModel.getLocation().getValue()==null){
            buildAlertMessageNoGps("Your GPS seems to be disabled, do you want to enable it?");
        }
        else if(manager.isProviderEnabled( LocationManager.GPS_PROVIDER ) && viewModel.getLocation().getValue()==null){
            getLocation();
        }
    }
}

DatabaseReference mDatabase = FirebaseDatabase.getInstance().getReference().child("destinations");
mDatabase.keepSynced(true);
getLocationPermission();
BottomNavigationView bnv = findViewById(R.id.bottom_navigation);
bnv.setOnNavigationItemSelectedListener(navListener);

if (user != null)
    viewModel.getProfile();
if (savedInstanceState == null) {
    selectedFragment = new HomeFragment();
    TAG_MY_FRAGMENT = "home";
    getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, selectedFragment, TAG_MY_FRAGMENT).commit();
} else {
    TAG_MY_FRAGMENT = savedInstanceState.getString( key: "TAG_MY_FRAGMENT", defaultValue: "home");
    selectedFragment = getSupportFragmentManager().findFragmentByTag(TAG_MY_FRAGMENT);
}
}
```

Η μέθοδος αυτή ελέγχει αν ο χρήστης είναι συνδεδεμένος. Σε περίπτωση που δεν είναι τον οδηγεί στην οθόνη σύνδεσης. Αν είναι συνδεδεμένος του ζητάει την άδεια για πρόσβαση στις



υπηρεσίες εντοπισμού τοποθεσίας του κινητού του και αρχικοποιεί στην οθόνη ένα από τα τέσσερα fragments. Τα fragment αυτά είναι τα εξής:

- **To HomeFragment:** το οποίο περιέχει κάποιες γενικές λειτουργίες και εμφανίζεται όταν ανοίγουμε την εφαρμογή.
- **To ProfileFragment:** στο οποίο ο χρήστης συμπληρώνει το προφίλ του.
- **To GuideFragment:** στο οποίο ο χρήστης βλέπει τις εξατομικευμένες, με βάση το προφίλ του, προτάσεις
- **To MapFragment:** το οποίο περιέχει ένα χάρτη.

Αναλυτικά τα fragment αυτά θα αναλυθούν σε επόμενες ενότητες.

Οι υπόλοιπες συναρτήσεις που περιέχονται στο MainActivity είναι οι παρακάτω:

- **getLocationPermission:** Αυτή η μέθοδος ελέγχει αν ο χρήστης έχει δώσει δικαίωμα στην εφαρμογή για πρόσβαση στις υπηρεσίες εντοπισμού τοποθεσίας του κινητού. Σε περίπτωση που δεν έχει δώσει τον ρωτάει αν θέλει να δώσει αυτό το δικαίωμα στην



εφαρμογή.

```
private boolean getLocationPermission() {
    Log.d(TAG, msg: "getLocationPermission: getting permissions");
    String[] permissions = {Manifest.permission.ACCESS_FINE_LOCATION};
    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
        return true;
    } else {
        ActivityCompat.requestPermissions( activity: this, permissions, LOCATION_PERMISSION_REQUEST_CODE);
        return false;
    }
}
```

- **onRequestPermissionsResult:** Αυτή η μέθοδος ζητάει από τον χρήστη να ανοίξει το GPS.

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    Log.d(TAG, msg: "checking location permissions");

    switch (requestCode) {
        case LOCATION_PERMISSION_REQUEST_CODE:
            if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                buildAlertMessageNoGps("Your GPS seems to be disabled, do you want to enable it?");
                Log.d(TAG, msg: "permission given");
            }
    }
}
```

- **onCreateOptionsMenu:** Αυτή η μέθοδος ενσωματώνει το μενού που βρίσκεται στο πάνω μέρος της οθόνης.
- **getLocation:** Αυτή η μέθοδος βρίσκει την τοποθεσία του χρήστη. Η τοποθεσία αποθηκεύεται σε ένα ViewModel ώστε να είναι διαθέσιμη και στα υπόλοιπα fragments.



Περισσότερα σχετικά για τα Viemodells θα δούμε σε επόμενες ενότητες.

```
@SuppressWarnings("MissingPermission")
public void getLocation() {

    mFusedClient.getLastLocation()
        .addOnSuccessListener( activity: this, new OnSuccessListener<Location>() {
            @Override
            public void onSuccess(Location location) {
                // Got last known location. In some rare situations this can be null.
                if (location != null) {
                    // Logic to handle location object
                    My_Location loc = new My_Location(location.getLatitude(),location.getLongitude());
                    viewModel.setLocation(loc);
                } else {
                    final LocationRequest locationRequest = LocationRequest.create();
                    locationRequest.setExpirationDuration(30000);
                    locationRequest.setInterval(10000);
                    locationRequest.setFastestInterval(5000);
                    locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
                    mLocationCallback = (LocationCallback) onLocationResult(locationResult) -> {
                        super.onLocationResult(locationResult);
                        if (locationResult == null) {
                            return;
                        }
                        Location mLastKnownLocation = locationResult.getLastLocation();
                        My_Location loc = new My_Location(mLastKnownLocation.getLatitude(),mLastKnownLocation.getLongitude());
                        viewModel.setLocation(loc);
                        Log.d(TAG, msg: "Location found remove updates");
                        mFusedClient.removeLocationUpdates(mLocationCallback);
                    };
                    Log.d(TAG, msg: "Location requesting updates ");
                    mFusedClient.requestLocationUpdates(locationRequest, mLocationCallback, LocationSettingsCompat.DEFAULT, null);
                }
            }
        });
}
```

- **onOptionsItemSelected:** Αυτή η μέθοδος υλοποιεί την λειτουργία αποσύνδεσης.

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    switch (item.getItemId()) {
        case R.id.log_out:
            FirebaseAuth user = FirebaseAuth.getInstance().getCurrentUser();
            if ( user.isAnonymous() ) {
                Log.d(TAG, msg: "Anonymous user");
                DatabaseReference database = FirebaseDatabase.getInstance().getReference().child("Profiles").child(user.getUid());
                database.removeValue();
                user.delete();
                Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
                startActivity(intent);
                finish();
            } else
                logout();
        }
    return super.onOptionsItemSelected(item);
}
```



- **logout:** Καλείται από την προηγούμενη μέθοδο για να αποσυνδεθεί ο χρήστης.

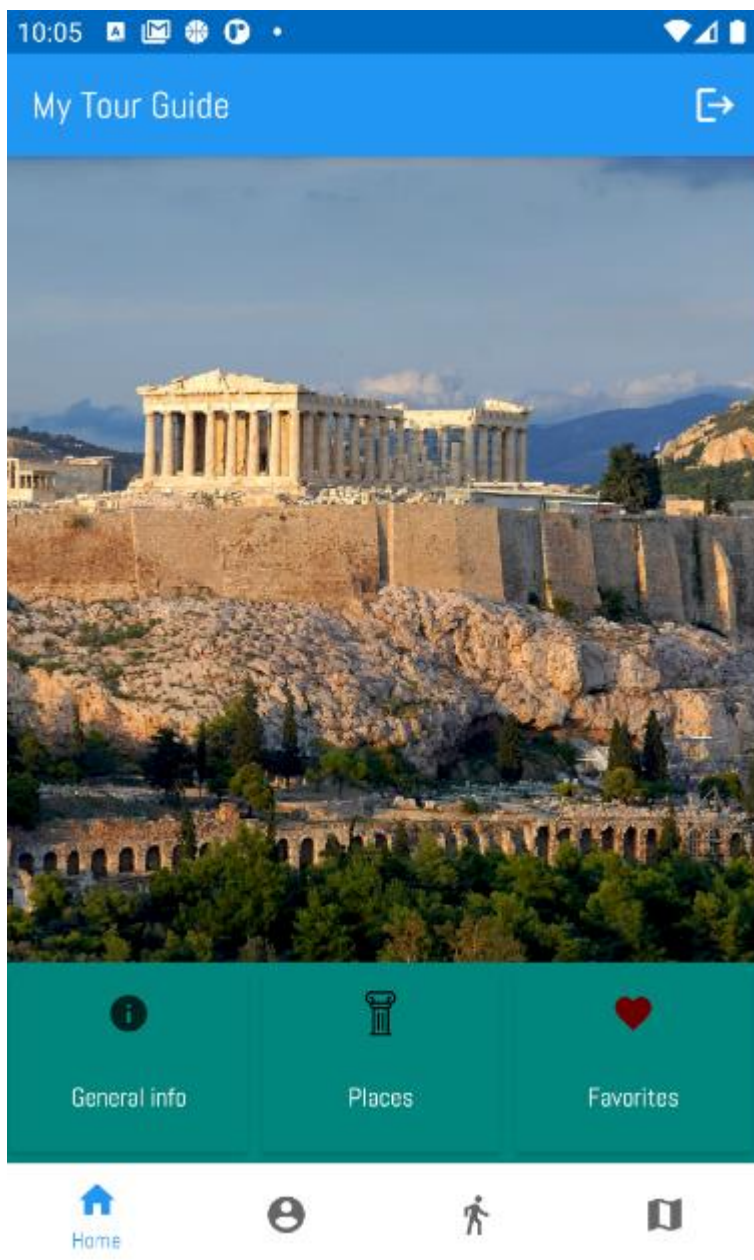
```
private void logout() {  
  
    FirebaseAuth.getInstance().signOut();  
    Intent intent = new Intent(getApplicationContext(), LoginActivity.class);  
    startActivity(intent);  
    finish();  
}
```

- **isServiceOk:** Αυτή η μέθοδος ελέγχει αν η εφαρμογή μπορεί να φορτώσει τον χάρτη στο MapFragment.
- **setFragment:** Αυτή η μέθοδος φορτώνει το fragment που αντιστοιχεί στο κουμπί που πατήθηκε από το κάτω μενού του activity.

```
private void setFragment(int id) {  
  
    switch (id) {  
        case R.id.home:  
            TAG_MY_FRAGMENT = "home";  
            selectedFragment = new HomeFragment();  
            break;  
        case R.id.profile:  
            TAG_MY_FRAGMENT = "profile";  
            selectedFragment = new ProfileFragment();  
            break;  
        case R.id.start:  
            TAG_MY_FRAGMENT = "guide";  
            selectedFragment = new GuideFragment();  
            break;  
        case R.id.map:  
            if (isServicesOk()) {  
                TAG_MY_FRAGMENT = "map";  
                selectedFragment = new MapFragment();  
            }  
            break;  
    }  
  
    getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, selectedFragment, TAG_MY_FRAGMENT).commit();  
}
```

3.3.2 Αρχική οθόνη

Η αρχική οθόνη από το HomeFragment και εμφανίζεται μόλις ο χρήστης ανοίξει την εφαρμογή ή μόλις συνδεθεί.



Εικόνα 6.Οθόνη HomeFragment

Μέσω του HomeFragment ο χρήστης μπορεί να περιηγηθεί σε άλλες περιοχές της εφαρμογής. Συγκεκριμένα μπορεί να δει γενικές πληροφορίες για την Αθήνα, να δει τα μέρη που υπάρχουν διαθέσιμα στην εφαρμογή και να δει την λίστα των τοποθεσιών που έχει επιλέξει ως αγαπημένα του.

Το HomeFragment αποτελείται από μόλις δύο μεθόδους, την onCreateView και την startActivity.

Η onCreateView είναι η συνάρτηση που περιέχεται σε κάθε fragment και η πρώτη που εκτελείται όταν ένα fragment ξεκινάει. Στο συγκεκριμένο fragment αυτό που κάνει είναι να



καθορίζει τις λειτουργίες που πρέπει να κάνει κάθε κουμπί όταν πατηθεί.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                          Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    view= inflater.inflate(R.layout.fragment_home, container, attachToRoot: false);
    view.findViewById(R.id.athensInfoButton).setOnClickListener(v -> {
        startActivity(AthensInfoActivity.class);
    });
    view.findViewById(R.id.dest_Button).setOnClickListener(v -> {
        startActivity(DestinationsActivity.class);
    });
    view.findViewById(R.id.myFavorites).setOnClickListener(v -> {
        startActivity(FavoritesActivity.class);
    });

    return view;
}
```

Η startActivity δέχεται ως όρισμα ένα activity και το ανοίγει.

```
private void startActivity(Class activity){
    Intent intent = new Intent(getActivity(), activity);
    startActivity(intent);
}
```

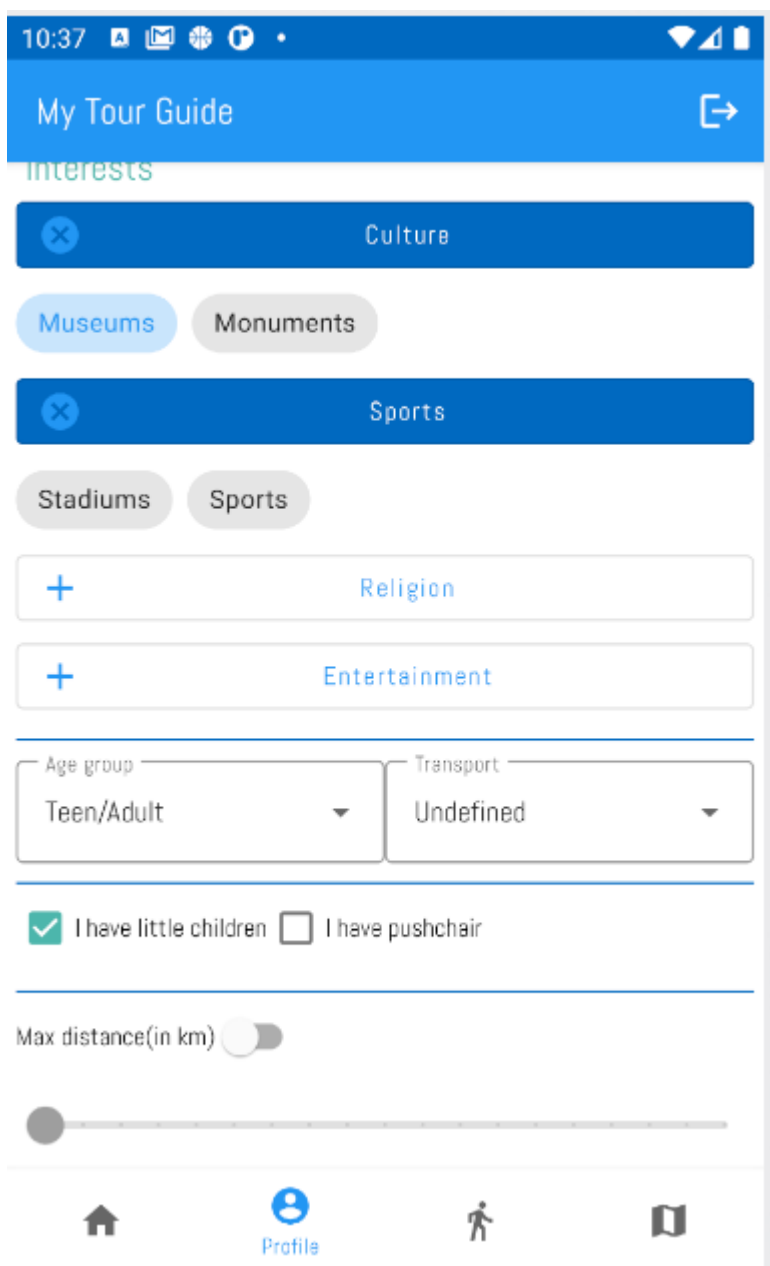
3.3.3 Συμπλήρωση προφίλ

3.3.3.1 Προφίλ χρήστη

Η οθόνη στην οποία ο χρήστης μπορεί να συμπληρώσει το προφίλ του υλοποιείται από το ProfileFragment. Όπως φαίνεται στην εικόνα 7 ο χρήστης μπορεί να επιλέξει ως ενδιαφέροντά του μια ή περισσότερες επιλογές που υπάρχουν διαθέσιμες σε κάθε κατηγορία . Συγκεκριμένα κάθε κατηγορία έχει τις εξής επιλογές:

- **Πολιτισμός:** μουσεία και μνημεία
- **Αθλητισμός:** στάδια και άθληση
- **Θρησκεία:** εκκλησίες και μοναστήρια
- **Διασκέδαση:** περιήγηση, shopping, θέατρο, καφές/φαγητό

Επιπλέον ο χρήστης μπορεί να δηλώσει σε ποια ηλικιακή ομάδα ανήκει, με τι μέσο έχει σκοπό να μετακινηθεί, αν έχει παιδιά και σε περίπτωση που έχει αν κουβαλάει καροτσάκι. Στο κάτω μέρος της οθόνης ο χρήστης μπορεί ενεργοποιώντας τον διακόπτη να επιλέξει την ακτίνα μέσα στην οποία θέλει να μετακινηθεί. Για την τελευταία λειτουργία είναι αναγκαία η χρήση GPS.



Εικόνα 7. Προφίλ χρήστη

Σκοπός μας ήταν να δημιουργήσουμε ένα προφίλ που μπορεί να συμπληρωθεί γρήγορα και συγχρόνως να αποσπά από τον χρήστη όλες τις απαραίτητες πληροφορίες που χρειαζόμαστε για να του βρούμε τα μέρη που του ταιριάζουν. Γνωρίζοντας τα ενδιαφέροντα του μπορούμε να προσδιορίσουμε με μεγάλη ακρίβεια τις τοποθεσίες που τον ενδιαφέρουν. Ταυτόχρονα η ηλικιακή του ομάδα και η οικογενειακή του κατάσταση μας υποδηλώνουν κάποιες ειδικές ανάγκες που πρέπει να ληφθούν υπόψιν όπως η ευκολία πρόσβασης των προορισμών και το πόσο μεγάλες αποστάσεις είναι ικανός ο χρήστης να διανύσει.



Για παράδειγμα, ένας ηλικιωμένος θα προτιμά να επισκεφθεί μέρη κοντά σε απόσταση και με εύκολη πρόσβαση. Μια οικογένεια με παιδιά θα θέλει να επισκεφθεί που είναι κατάλληλα για παιδιά.

Επίσης, επειδή η Αθήνα είναι μια πόλη με μεγάλο κυκλοφοριακό πρόβλημα κάποια μέρη είναι καλύτερο να τα επισκεφθείς με ιδιωτικό όχημα και άλλα με μέσα μαζικής μεταφοράς.

3.3.3.2 ProfileFragment

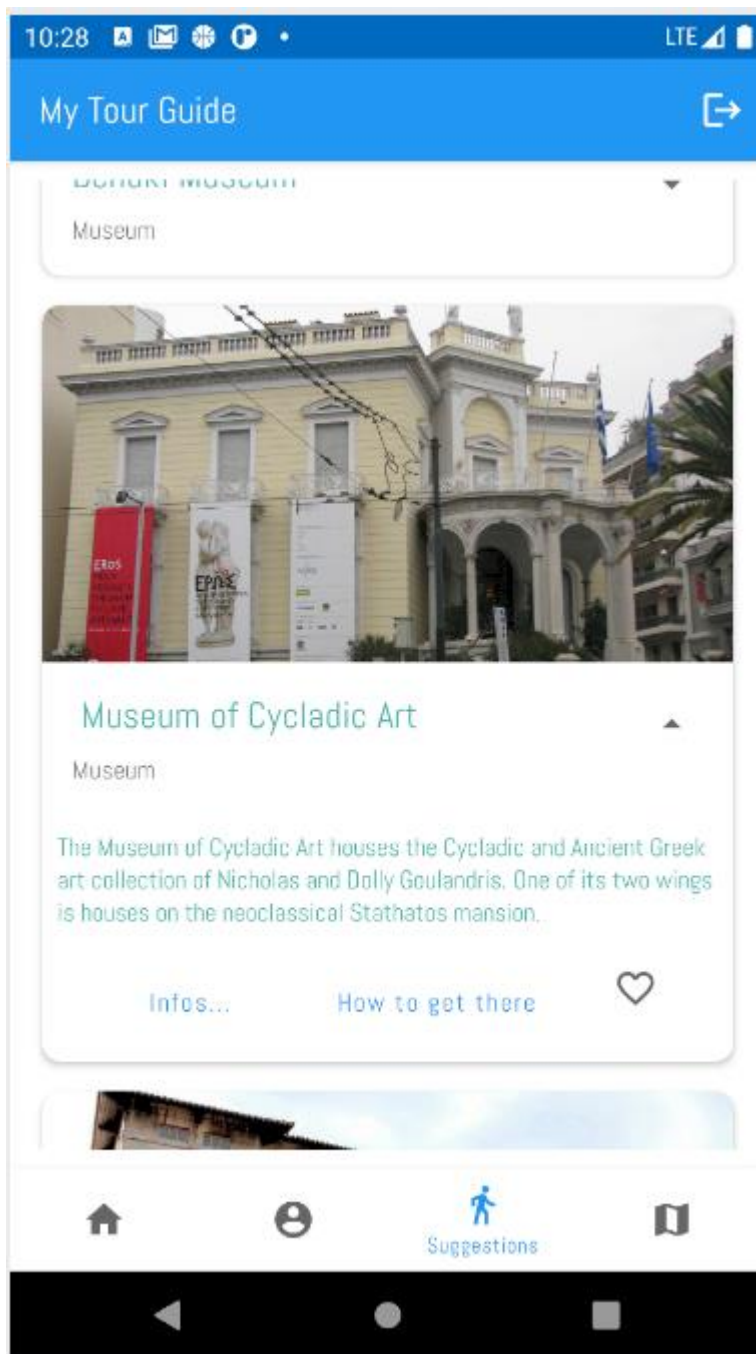
Έχοντας αναλύσει την λογική πίσω από τον σχεδιασμό του προφίλ ας δούμε τον κώδικα που υλοποιεί τις απαραίτητες λειτουργίες. Το ProfileFragment αποτελείται από τις εξής μεθόδους:

- **onCreateView:** Σε αυτή την μέθοδο φορτώνεται το προφίλ του χρήστη από την Realtime Database και ανανεώνεται το UI.
- **chipClicked:** Κάθε φορά που ο χρήστης πατάει σε ένα από τα διαθέσιμα ενδιαφέροντα, αυτή τη μέθοδος ανανεώνει το προφίλ του.
- **buttonClicked:** Πατώντας ο χρήστης πάνω σε μια κατηγορία αυτή διαστέλλεται και συστέλλεται.
- Οι μέθοδοι **initializeChips**, **initializeExpendables**, **initializeInterestButtons** αρχικοποιούν διάφορα στοιχεία του UI και καθορίζουν τις ενέργειες που πρέπει να κάνουν όταν ο χρήστης αλληλοεπιδράσει μαζί τους
- **isLocationPermissionGranded:** Ελέγχει αν ο χρήστης έχει δώσει δικαιώματα χρήσης GPS στην εφαρμογή.
- **onResume:** Καλείται όποτε η εφαρμογή επανέρχεται από την αναμονή.
- **onSaveInstanceState:** Καλείται πριν γίνει κάποια αλλαγή στις παραμέτρους της εφαρμογής, όπως η περιστροφή οθόνης.
- **buildAlertMessageNoGps:** Εμφανίζει ένα μήνυμα ζητώντας από τον χρήστη να ανοίξει το GPS.

3.3.4 Προβολή προτεινομένων σημείων ενδιαφέροντος

Ο χρήστης πατώντας το κουμπί «Προτάσεις» στο κάτω μέρος της οθόνης μπορεί να δει τα προτεινόμενα μέρη. Τα μέρη αυτά είναι αποτέλεσμα της ανάλυσης των χαρακτηριστικών του προφίλ του χρήστη. Ο τρόπος εύρεσης αυτών των τοποθεσιών θα αναλυθεί σε παρακάτω ενότητα.

Όπως φαίνεται ο χρήστης μπορεί να δει βασικές πληροφορίες για τον κάθε προορισμό, να επισκεφθεί την ιστοσελίδα της τοποθεσίας και να βρει οδηγίες για το πώς να πάει σε αυτόν. Επίσης πατώντας στην καρδιά μπορεί να προσθέσει τον προορισμό στα αγαπημένα του.



Εικόνα 8. Οθόνη προτάσεων

Οι λειτουργίες που μόλις αναφέρθηκαν υλοποιούνται από το GuideFragment. Οι μέθοδοι που το αποτελούν είναι οι εξής:

- **onCreateView:** Παίρνει το προφίλ του χρήστη από το ProfilesViewModel και το παρέχει στην κλάση MyDestinationViewModel για να του επιστρέψει τα προτεινόμενα μέρη.



```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_guide, container, attachToRoot: false);
    Log.i(TAG, msg: "Guide start");
    destinationsRV = view.findViewById(R.id.destinationsRV);

    ProfilesViewModel viewModel = new ViewModelProvider(requireActivity()).get(ProfilesViewModel.class);
    MyDestinationsViewModel destviewModel = new ViewModelProvider(requireActivity()).get(MyDestinationsViewModel.class);
    float distance = viewModel.getDistance().getValue();
    My_Location my_location=viewModel.getLocation().getValue();

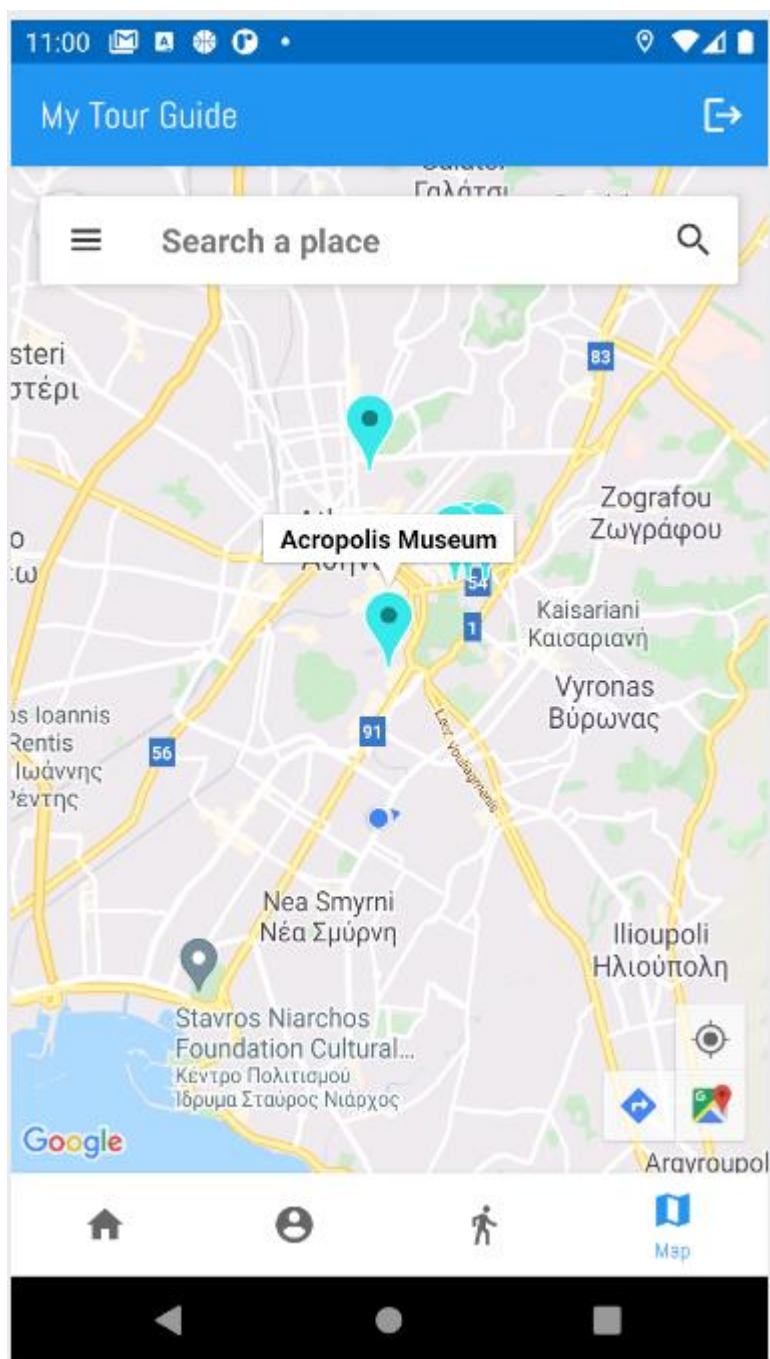
    profile =viewModel.getProfile().getValue();
    destviewModel.setDestinations(profile,distance,my_location);

    destviewModel.getDestinations().observe(getViewLifecycleOwner(), new Observer<List<Destination>>() {
        @Override
        public void onChanged(List<Destination> dest) {
            Log.i(TAG, msg: "data changed");
            for(Destination destination:dest)
                Log.i(TAG, msg: destination.getName()+" "+destination.getLocation().getLatitude()+" "+destination.getLocation().getLongitude());
            Log.i(TAG,String.valueOf(dest.size()));
            destinations = (ArrayList<Destination>) dest;
            if(destinations==null)
                Toast.makeText(getContext(), text: "You have to fill your profile",Toast.LENGTH_SHORT).show();
            else
                setUpAdapter();
        }
    });
    return view;
}
```

- **setUpAdapter:** που εμφανίζει την λίστα με τις προτάσεις.

3.3.5 Χάρτης

Σε εφαρμογές όπου ο χρήστης αναζητά διάφορες τοποθεσίες σε άγνωστα μέρη είναι απαραίτητη η παροχή ενός χάρτη. Στην δική μας εφαρμογή ο χάρτης γίνεται διαθέσιμος πατώντας το τελευταίο κουμπί στο μενού που βρίσκεται στο κάτω μέρος της οθόνης. Με την βοήθεια του χάρτη ο χρήστης μπορεί να προσανατολιστεί, να βρει τα μέρη που του προτείνει η εφαρμογή (εμφανίζονται στον χάρτη) και να ψάξει άλλες τοποθεσίες.



Εικόνα 9. Χάρτης



Ο χάρτης υλοποιείται από το MapFragment, το οποίο αποτελείται από τις παρακάτω μεθόδους:

- **onCreateView:** Αρχικοποιεί τα γραφικά.
- **onMapReady:** Αρχικοποιεί τις παραμέτρους του χάρτη. Βρίσκει την τοποθεσία του χρήστη αν είναι ανοιχτό το GPS και υπάρχουν . Καλείται όταν αρχικοποιηθεί ο χάρτης.
- **isLocationPermissionGranded:** Ελέγχει αν ο χρήστης έχει δώσει στην εφαρμογή δικαιώματα χρήσης της τοποθεσίας του.
- **initMap:** Αρχικοποιεί τον χάρτη.
- **getDeviceLocation:** Βρίσκει την τοποθεσία του χρήστη.

```
@SuppressWarnings("MissingPermission")
private void getDeviceLocation() {
    mFusedclient.getLastLocation()
        .addOnCompleteListener(new OnCompleteListener<Location>() {
            @Override
            public void onComplete(@NonNull Task<Location> task) {
                if (task.isSuccessful()) {
                    mLastKnownLocation = task.getResult();
                    if (mLastKnownLocation != null) {
                        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(new LatLng(mLastKnownLocation.getLatitude(), mLastKnownLocation.getLongitude()), DEFAULT_ZOOM));
                    } else {
                        final LocationRequest locationRequest = LocationRequest.create();
                        locationRequest.setInterval(10000);
                        locationRequest.setFastestInterval(5000);
                        locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
                        mLocationCallback = (LocationCallback) onLocationResult(locationResult) -> {
                            super.onLocationResult(locationResult);
                            if (locationResult == null) {
                                return;
                            }
                            mLastKnownLocation = locationResult.getLastLocation();
                            mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(new LatLng(mLastKnownLocation.getLatitude(), mLastKnownLocation.getLongitude()), DEFAULT_ZOOM));
                            mFusedclient.removeLocationUpdates(mLocationCallback);
                        };
                        mFusedclient.requestLocationUpdates(locationRequest, mLocationCallback, looper: null);
                    }
                } else {
                    Toast.makeText(getActivity(), "unable to get last location", Toast.LENGTH_SHORT).show();
                }
            }
        });
}
```

- **getLocations:** Παίρνει τις τοποθεσίες που προτείνονται στον χρήστη και τις φορτώνει στον χάρτη.



```
private void getLocations(){
    destViewModel.getDestinations().observe(getViewLifecycleOwner(), new Observer<List<Destination>>() {
        @Override
        public void onChanged(List<Destination> dest) {
            for(Marker m : blue_markers){
                m.remove();
            }
            blue_markers.clear();
            if(dest!=null&&dest.size()>0){
                for(Destination destination:dest){
                    String title;
                    if(Locale.getDefault().getDisplayLanguage().contains("English"))
                        title=destination.getName();
                    else
                        title=destination.getName_gr();
                    LatLng latlng = new LatLng(destination.getLocation().getLatitude(), destination.getLocation().getLongitude());
                    blue_markers.add(mMap.addMarker(new MarkerOptions().position(latlng).title(title).icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_CYAN))));
                }
            }
        }
    });
}
```

- **search:** Εκτελεί την λειτουργία της αναζήτησης τοποθεσιών.

```
private void search() {
    materialSearchBar.setOnSearchActionListener(new MaterialSearchBar.OnSearchActionListener() {
        @Override
        public void onSearchStateChanged(boolean enabled) {

        }

        @Override
        public void onSearchConfirmed(CharSequence text) {
            String location = text.toString();
            List<Address> addressList = null;
            if (location != null || !location.equals("")) {
                Geocoder geocoder = new Geocoder(getContext());
                try {
                    addressList = geocoder.getFromLocationName(location, maxResults: 1);
                    Address address = addressList.get(0);
                    LatLng latlng = new LatLng(address.getLatitude(), address.getLongitude());
                    mMap.addMarker(new MarkerOptions().position(latlng).title(location));
                    mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latlng, DEFAULT_ZOOM));
                } catch (IOException e) {
                    e.printStackTrace();
                } catch (IndexOutOfBoundsException e) {
                    Toast.makeText(getContext(), text: "Zero results", Toast.LENGTH_SHORT).show();
                }
            }
        }
    });
}
```

- **onButtonClicked:** Καθορίζει τις λειτουργίες του χάρτη.
- **onNavigationItemSelected:** Αλλάζει τον τύπο του χάρτη.

3.4 Περιοχή με γενικές πληροφορίες για την Αθήνα

Πατώντας το κουμπί «Γενικές πληροφορίες» (ή General Info στο αγγλικό UI), ο χρήστης μπορεί να διαβάσει μερικές πληροφορίες για την Αθήνα.



About Athens

Athens is the capital of Greece since 1834 and the largest and most populous city in the country. It is located in Attica, in eastern central Greece, and is one of the oldest cities in the world, with its recorded history reaching up to 3,200 BC.

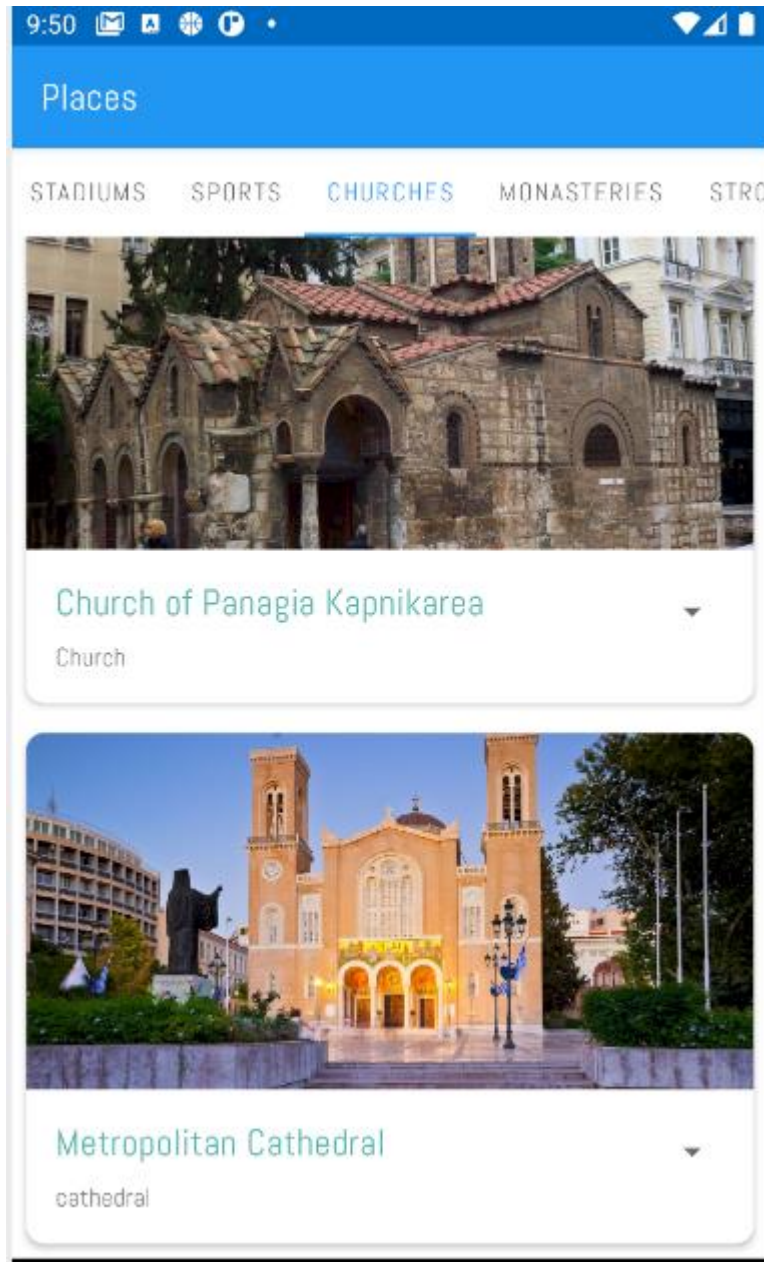
One of the most ancient cities of the world, Athens Greece is famous as the birthplace of democracy. With a history of over 3,000 years, Athens is the best town for sightseeing. According to the myth, the city took its name after Athena, the Goddess of Wisdom and daughter of Zeus. She was the protector of the city and in fact, the temple of Parthenon on the Acropolis is dedicated to her.

The first traces of Athens date from the Mycenaean period, while it reached its peak after at the 5th century BC, known as the Golden Century of Pericles, the Athenian statesman who has inspired so many innovations for his homeland and led it to its glory. From as early as the 8th century BC, Athens was gradually developing into an important city-state for Greece, giving emphasis on culture and its naval power. But it was in the 5th century BC when great political formations were made, new buildings were constructed, including the Acropolis, the Temple of

Εικόνα 10. Γενικές πληροφορίες.

3.5 Περιοχή με τις διαθέσιμες τοποθεσίες της εφαρμογής

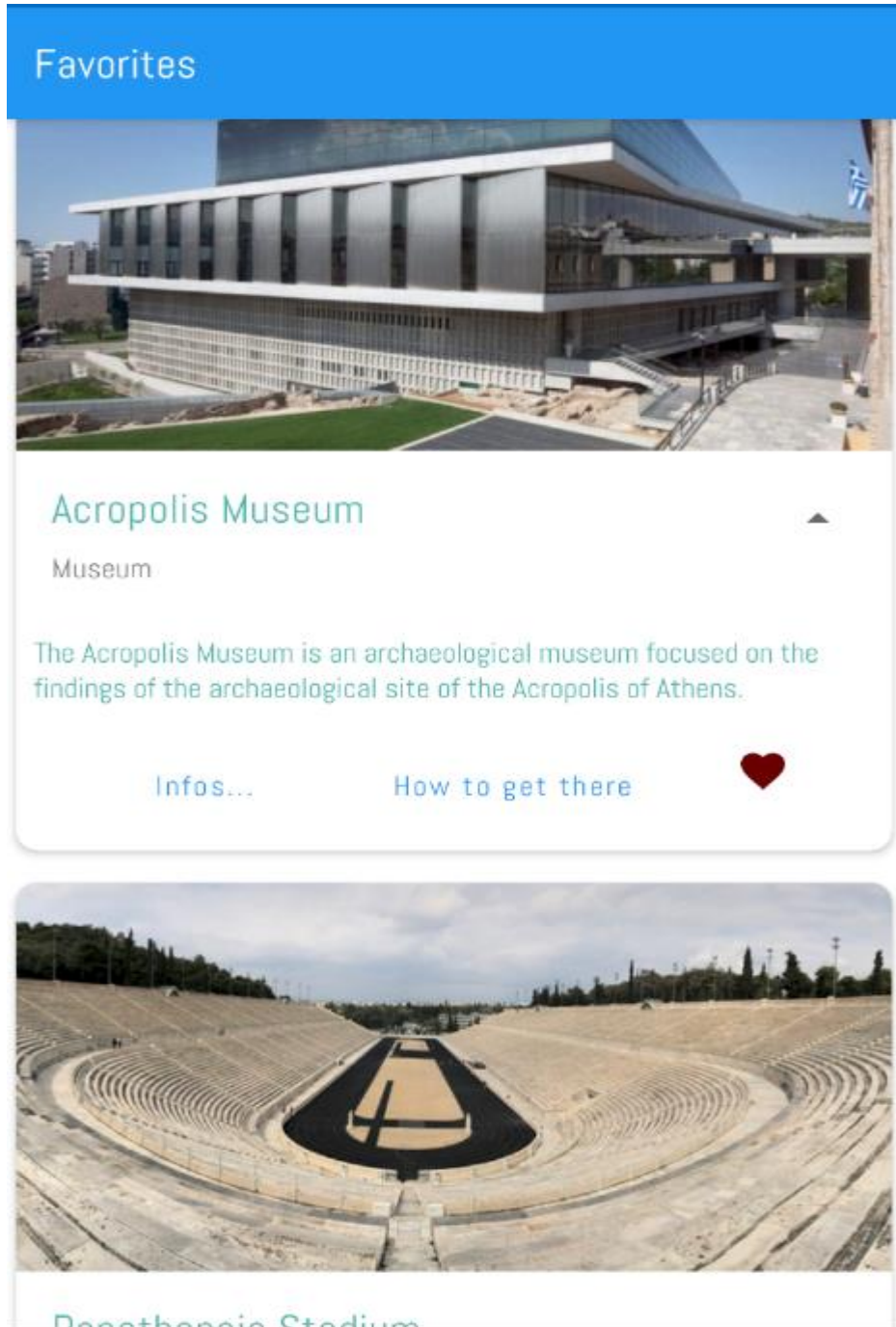
Πατώντας του κουμπι «Μέρη» (ή Places στο αγγλικό UI), ο χρήστης μπορεί να δει όλα τα διαθέσιμα μέρη, που περιέχονται στην βάση δεδομένων, ταξινομημένα ανά κατηγορία.



Εικόνα 11.Τοποθεσίες ανά κατηγορία.

3.6 Περιοχή με τα αγαπημένα

Πατώντας το κουμπί «Αγαπημένα» (Favorites στο αγγλικό UI), ο χρήστης μπορεί να δει τις τοποθεσίες που έχει αποθηκεύσει ως αγαπημένες του.



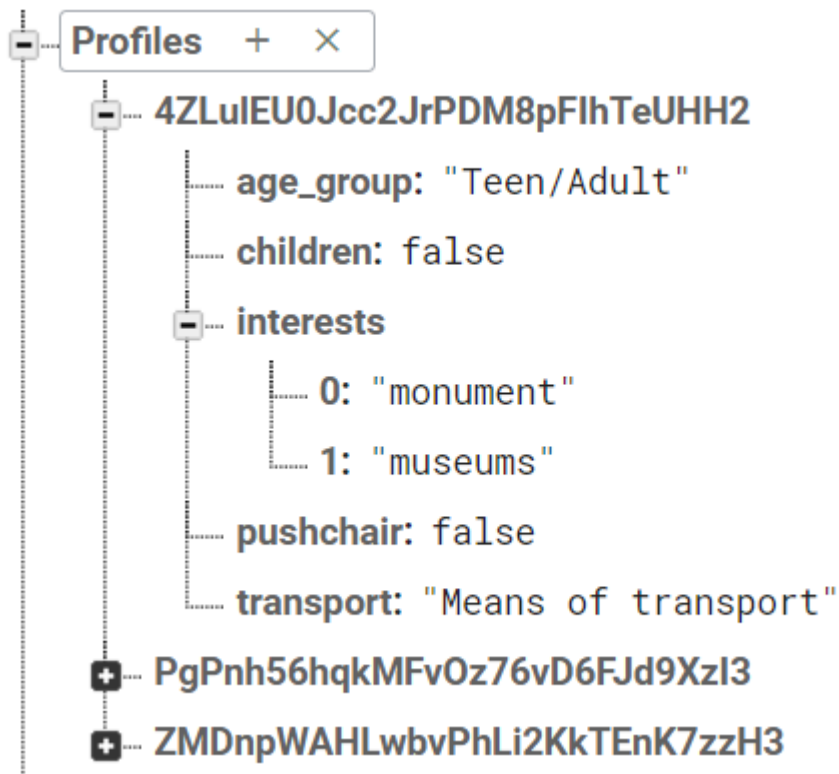
Εικόνα 12. Αγαπημένες τοποθεσίες.

3.7 Δεδομένα και βάση δεδομένων

3.7.1 Firebase Realtime Database

Για την αποθήκευση των δεδομένων της εφαρμογής χρησιμοποιήσαμε την Realtime Database της Firebase, η οποία αποθηκεύει τα δεδομένα σε μορφή json. Εκεί αποθηκεύονται όλες οι τοποθεσίες που παρουσιάζονται στην εφαρμογή και τα προφίλ των χρηστών.

Στην παρακάτω βλέπουμε τον τρόπο με το οποίο αποθηκεύεται το προφίλ του χρήστη.



Εικόνα 13. Realtime Database profiles

Τα προφίλ αποθηκεύονται κάτω από το node «Profiles». Κάθε προφίλ ξεχωρίζεται με βάση το id του λογαριασμού του χρήστη. Κάτω από το id υπάρχουν οι επιλογές που έχει επιλέξει ο χρήστης στην εφαρμογή.

Στην επόμενη εικόνα βλέπουμε τον τρόπο με τον οποίο είναι αποθηκευμένες οι τοποθεσίες.



Εικόνα 14. Realtime Database τοποθεσίες

Οι τοποθεσίες είναι αποθηκευμένες κάτω από το κόμβο «destinations». Για κάθε μέρος έχουμε αποθηκεύσει βασικές πληροφορίες. Συγκεκριμένα:

- σε τι κατηγορία ανήκει
- αν είναι ευκόλως προσβάσιμο
- μια εικόνα του
- πληροφορίες και στα αγγλικά και στα ελληνικά
- την τοποθεσία του
- το όνομα του στα αγγλικά και στα ελληνικά
- ιστοσελίδα με πληροφορίες
- τον προτεινόμενο τρόπο για μεταφορά προς αυτό το μέρος(ΜΜΜ ή/και αυτοκίνητο)
- αν είναι κατάλληλο για παιδιά



- αν είναι φυσικό τοπίο

3.7.2 Models

Για να μπορέσουμε να λάβουμε όλα αυτά τα δεδομένα φτιάξαμε κλάσεις-μοντέλα που αποτυπώνουν τα πεδία της βάσης. Έτσι για τις τοποθεσίες έχουμε το μοντέλο Destination και για το προφίλ το μοντέλο Profile.

```
public class Destination {  
    private String name;  
    private String site;  
    private String name_gr;  
    private boolean children;  
    private String image;  
    private String info;  
    private String info_gr;  
    private boolean easy_access;  
    private boolean nature;  
    private String type;  
    private ArrayList<String> category;  
    private ArrayList<String> transport;  
    private My_Location location;  
    private String type_gr;  
  
    public boolean isNature() { return nature; }  
  
    public void setNature(boolean nature) { this.nature = nature; }  
  
    public String getSite() { return site; }  
  
    public void setSite(String site) { this.site = site; }  
  
    public boolean isChildren() { return children; }  
  
    public void setChildren(boolean children) { this.children = children; }
```



```
public class Profile {
    private ArrayList<String> interests = new ArrayList<>();
    private ArrayList<String> favorites = new ArrayList<>();
    private String transport = "Undefined";
    private String age_group = "Undefined";
    private boolean pushchair;
    private boolean children;

    public ArrayList<String> getFavorites() { return favorites; }

    public void setFavorites(ArrayList<String> favorites) { this.favorites = favorites; }

    public boolean isPushchair() { return pushchair; }

    public void setPushchair(boolean pushchair) { this.pushchair = pushchair; }

    public boolean isChildren() { return children; }

    public void setChildren(boolean children) { this.children = children; }

    public String getTransport() {
        return transport;
    }

    public void setTransport(String transport) { this.transport = transport; }

    public String getAge_group() { return age_group; }

    public void setAge_group(String age_group) { this.age_group = age_group; }

    public ArrayList<String> getInterests() {
        return interests;
    }
}
```

3.7.3 Viewmodels

Η λήψη των δεδομένων από την βάση γίνεται μέσα σε ViewModels. Τα ViewModels χρησιμοποιούνται για να κρατούν τα δεδομένα της εφαρμογής κατά την διάρκεια της εκτέλεσής της. Έχουμε δημιουργήσει δύο ViewModels που αντιστοιχούν στα μοντέλα που περιγράφηκαν προηγουμένως; το ProfilesViewModel και το MyDestinationsViewModel.



3.7.3.1 ProfilesViewModel

Στο ProfilesViewModel λαμβάνουμε το προφίλ του χρήστη από την βάση και γράφουμε προς αυτήν τις διάφορες αλλαγές που κάνει ο χρήστης στο προφίλ του. Επιπλέον αποθηκεύουμε την τοποθεσία του χρήστη και την ακτίνα μέσα στην οποία επιθυμεί να βρίσκονται οι τοποθεσίες, ώστε να είναι διαθέσιμη σε όλα τα fragments.

Η λήψη του προφίλ από την βάση γίνεται από την μέθοδο `getProfile`.

```
public MutableLiveData<Profile> getProfile() {
    Log.i(TAG, msg: "Get profile");
    if (profile == null) {
        profile = new MutableLiveData<>();
        FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
        DatabaseReference mDatabase = FirebaseDatabase.getInstance().getReference().child("Profiles");
        mDatabase.child(user.getId()).addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if (snapshot.exists())
                    profile.postValue(snapshot.getValue(Profile.class));
                else profile.postValue(new Profile());
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
            }
        });
    }
    if(profile.getValue()==null)
        profile.postValue(new Profile());
    return profile;
}
```

Η εγγραφή του προφίλ στην βάση γίνεται από την μέθοδο `setProfile`.

```
public void setProfile(Profile profile) {
    Log.i(TAG, msg: "update profile");
    this.profile.setValue(profile);
    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    DatabaseReference mDatabase = FirebaseDatabase.getInstance().getReference();
    assert user != null;
    mDatabase.child("Profiles").child(user.getId()).setValue(profile);
}
```

Η λήψη και η αποθήκευση τις τοποθεσίας γίνεται από τις μεθόδους `getLocation` και `setLocation` αντίστοιχα.

Η λήψη και η αποθήκευση τις ακτίνας γίνεται από τις μεθόδους `getDistance` και `setDistance` αντίστοιχα.

3.7.3.2 MyDestinationsViewModel

Οι λειτουργίες στο MyDestinationsViewModel δεν περιορίζονται στην επιστροφή των δεδομένων της βάσης αφιλτράριστων. Οι τοποθεσίες που λαμβάνονται από την βάση φιλτράρονται με βάση το προφίλ του χρήστη και επιστρέφεται μια λίστα με τις καλύτερες επιλογές. Αυτή η διαδικασία γίνεται από την μέθοδο setDestinations().

```
mDatabase.addListenerForSingleValueEvent(new ValueEventListener() {  
  
    @Override  
    public void onDataChange(@NonNull DataSnapshot snapshot) {  
        Log.d(TAG, "msg: \"Fetching destinations\"");  
        ArrayList<Double> scores = new ArrayList<>();  
        ArrayList<Destination> dest = new ArrayList<>();  
  
        for (DataSnapshot dataSnapshot : snapshot.getChildren()) {  
            Destination destination = dataSnapshot.getValue(Destination.class);  
            Log.d(TAG, destination.getName());  
            if (distance >= 1 && my_location != null) {  
                float[] resultArray = new float[99];  
                Location.distanceBetween(my_location.getLatitude(), my_location.getLongitude(), destination.getLocation().getLatitude(), destination.getLocation().getLongitude(), resultArray);  
                if (resultArray[0] / 1000 > distance)  
                    continue;  
            }  
            double score = Jaccard.calculate(profile, destination);  
            if (score >= 1.0 - (float) profile.getInterests().size() / 8.0 || score >= 0.8) {  
                scores.add(score);  
                dest.add(destination);  
            }  
        }  
    }  
});
```

```
if (dest.size() > 0) {  
    if (my_location == null) {  
        int max = scores.indexOf(Collections.max(scores));  
        Collections.swap(dest, 0, max);  
        SortDestinations.sortByDistance(dest, dest.get(0).getLocation(), index: 0);  
    } else {  
        SortDestinations.sortByDistance(dest, my_location, index: -1);  
    }  
    if ((profile.isChildren() || profile.getAge_group().equals("elder"))) {  
        ArrayList<Destination> dest_temp = new ArrayList<>();  
        dest_temp.add(dest.get(0));  
        for (int i = 1; i < scores.size(); i++) {  
            float[] resultArray = new float[99];  
            Location.distanceBetween(dest.get(i).getLocation().getLatitude(), dest.get(i).getLocation().getLongitude(), dest.get(0).getLocation().getLatitude(),  
                scores.set(i, scores.get(i) * 3000 / resultArray[0]);  
            if (scores.get(i) >= (1.0 - (float) profile.getInterests().size() / 5.0) || scores.get(i) >= 0.8) {  
                dest_temp.add(dest.get(i));  
            }  
        }  
        if (dest_temp.size() >= 5) {  
            dest_temp.subList(5, dest_temp.size()).clear();  
            destinations.postValue(dest_temp);  
        } else  
            destinations.postValue(dest_temp);  
    } else {  
        if (dest.size() >= 7) {  
            dest.subList(7, dest.size()).clear();  
            destinations.postValue(dest);  
        } else  
            destinations.postValue(dest);  
    }  
} else  
    destinations.postValue(dest);  
Log.d(TAG, String.valueOf(scores));
```



Οι λειτουργίες αυτής της μεθόδου χωρίζονται σε δύο φάσεις. Στην πρώτη φάση κάθε τοποθεσία στην βάση δεδομένων βαθμολογείται με βάση την ομοιότητα της με το προφίλ του χρήστη. Για την μέτρηση της βαθμολογίας χρησιμοποιούμε μια μετρική που βασίζεται στην μετρική Jaccard και θα την αναλύσουμε στην επόμενη ενότητα. Όσες τοποθεσίες παίρνανε ένα συγκεκριμένο όριο παίρνανε στην επόμενη φάση. Η εξίσωση του ορίου παραδίδεται παρακάτω.

$$\text{όριο} = \frac{\text{αριθμός ενδιαφερόντων χρήστη}}{8}$$

Εξίσωση 1. Όριο ομοιότητας προφίλ χρήστη με τοποθεσία

Στην δεύτερη φάση οι τοποθεσίες ταξινομούνται σε αύξοντα σειρά είτε με βάση την τοποθεσία του χρήστη (αν είναι διαθέσιμη), είτε με βάση την τοποθεσία της τοποθεσίας με το μεγαλύτερο σκορ. Επίσης η απόσταση ενός μέρους από το επόμενο έχει αρνητική βαρύτητα σε περίπτωση που ο χρήστης είναι ηλικιωμένος ή έχει παιδί και ελέγχεται ξανά αν περνάει το όριο.

Αν ο χρήστης είναι ο ηλικιωμένος ή έχει παιδιά του επιστρέφεται μια διαδρομή μέχρι 5 τοποθεσιών, αλλιώς του επιστρέφονται 7.

3.8 Υπολογισμός συντελεστή ομοιότητας

Ο υπολογισμός του συντελεστή ομοιότητας μεταξύ του προφίλ του χρήστη και ενός προορισμού γίνεται με βάση την μετρική Jaccard. Ο συντελεστής ομοιότητας Jaccard υπολογίζει την ομοιότητα μεταξύ πεπερασμένων συνόλων και ορίζεται ως το πηλίκο του μέτρου της τομής των συνόλων προς το μέτρο της ένωσης.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Εξίσωση 2. Jaccard Coefficient

Στην εφαρμογή μας ο υπολογισμός της ομοιότητας γίνεται από την κλάση Jaccard και συγκεκριμένα από την μέθοδο calculate. Η μέθοδος δέχεται ως είσοδο το προφίλ του χρήστη και έναν προορισμό και δίνει ως έξοδο την ομοιότητα τους.



```
public static double calculate(Profile profile, Destination destination) {  
  
    ArrayList profile_interests = new ArrayList(profile.getInterests());  
    ArrayList destination_interests = new ArrayList(destination.getCategory());  
    ArrayList transport_list = new ArrayList(destination.getTransport());  
    double union = getUnionOfLists(profile_interests, destination_interests);  
    double intersect = 1.5*getIntersectOfLists(profile_interests, destination_interests) ;  
  
    if (transport_list.contains(profile.getTransport()) && !profile.getTransport().equals("Undefined")) {  
        intersect++;  
        union ++;  
    } else if (!profile.getTransport().equals("Undefined"))  
        union ++;  
    else{  
        union+=transport_list.size();  
        intersect+=transport_list.size();  
    }  
  
    if ((profile.getAge_group().equals("Elder") || profile.isPushchair())) {  
        if (destination.isEasy_access())  
            intersect++;  
        union++;  
    }  
  
    if (profile.isChildren()) {  
        if (destination.isChildren())  
            intersect++;  
        union++;  
    }  
  
    if(profile.getAge_group().equals("Teen/Adult") && !profile.isChildren()){  
        if(destination.isNature()) {  
            intersect++;  
            union++;  
        }  
    }  
  
    Log.d( tag: "Jaccard intersect",String.valueOf(intersect) );  
    Log.d( tag: "Jaccard union",String.valueOf(union) );  
    Log.d( tag: "Jaccard score",String.valueOf(intersect / union));  
    return intersect / union;  
}
```

Όμως όπως είδαμε και στην ενότητα που αναλύσαμε την βάση δεδομένων δεν υπάρχει ιδιαίτερη ομοιότητα μεταξύ των χαρακτηριστικών του προφίλ του χρήστη και των χαρακτηριστικών των προορισμών. Στην πραγματικότητα, όμως, κάθε χαρακτηριστικό του προφίλ του χρήστη αντιστοιχίζεται σε ένα ή περισσότερα χαρακτηριστικά ενός προορισμού. Συγκεκριμένα :

- Τα ενδιαφέροντα του χρήστη συγκρίνονται με την κατηγορία που ανήκει ο προορισμός.
- Για έναν χρήστη που είναι έφηβος/ενήλικας και δεν έχει παιδιά κοιτάμε αν ο προορισμός έχει φυσικό περιβάλλον ή όχι.



- Για έναν χρήστη που είναι ηλικιωμένος κοιτάμε αν ο προορισμός είναι ευκόλως προσβάσιμος ή όχι.
- Για έναν χρήστη που έχει παιδιά κοιτάμε αν ο προορισμός είναι κατάλληλος για παιδιά.
- Για έναν χρήστη που κουβαλάει καροτσάκι κοιτάμε αν ο προορισμός είναι ευκόλως προσβάσιμος.

Τέλος επειδή θέλουμε να επιβραβεύσουμε την ομοιότητα ενός προορισμού με τα ενδιαφέροντα του χρήστη, το μέτρο της τομής των ενδιαφερόντων του χρήστη με τις κατηγορίες που ανήκει ο προορισμός πολλαπλασιάζεται επί 1,5.



Κεφάλαιο 4^ο

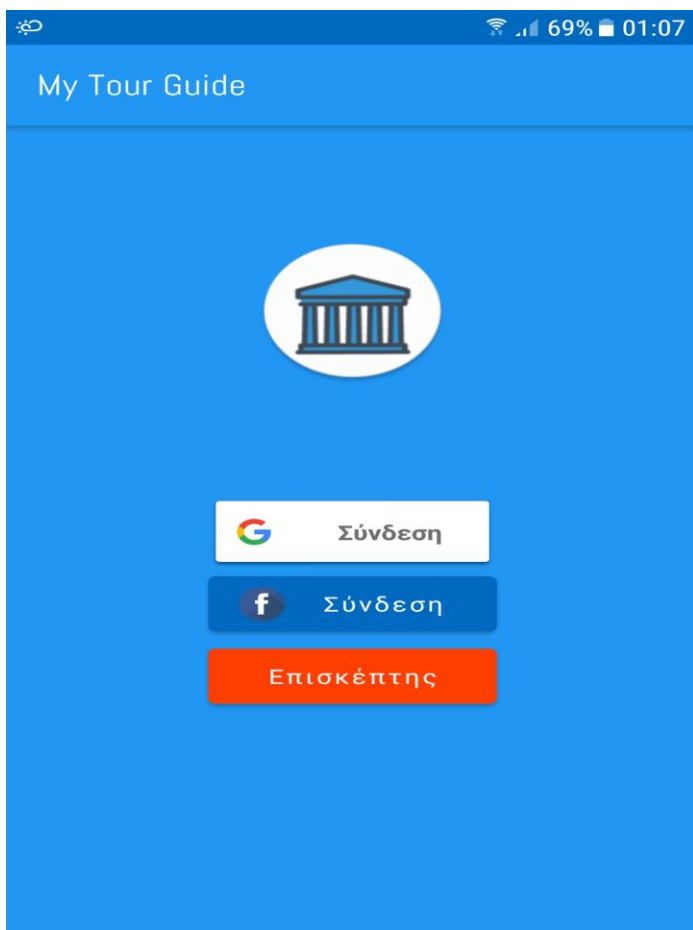
4 Εγχειρίδιο Χρήστη

4.1 Σύνδεση στην εφαρμογή

Μπορείτε να συνδεθείτε στην εφαρμογή με τρεις τρόπους:

- Με λογαριασμό Google.
- Με λογαριασμό Facebook
- Ως επισκέπτης

Αν συνδεθείτε μέσω Google ή Facebook, δημιουργείτε μόνιμο προφίλ. Δηλαδή το προφίλ σας θα είναι αποθηκευμένο και για τις επόμενες χρήσεις σας. Αν συνδεθείτε ως επισκέπτης δημιουργείτε ένα προσωρινό προφίλ το οποίο διαγράφεται μόλις αποσυνδεθείτε.

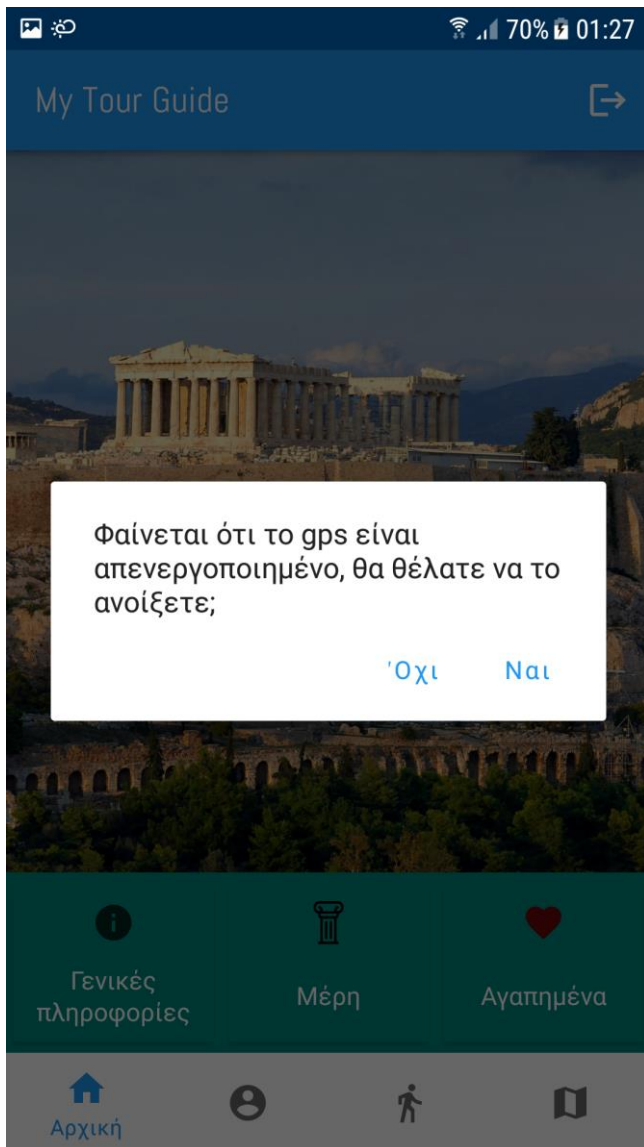


Εικόνα 15. Οθόνη σύνδεσης

Για να συνδεθείτε πατήστε το κουμπί που αντιστοιχεί στον τρόπο σύνδεσης που επιθυμείτε.

4.2 Άδεια χρήση της τοποθεσίας

Με το που συνδεθείτε θα σας ζητηθεί να δώσετε δικαιώματα στην εφαρμογή να έχει πρόσβαση στην τοποθεσία της συσκευής σας. Την τοποθεσία σας την χρησιμοποιούμε για να βρούμε προορισμούς κοντά σας και η χρήση της είναι προαιρετική. Σε περίπτωση που δώσετε το δικαίωμα για χρήση της τοποθεσίας και δεν έχετε ανοιχτό το GPS θα ερωτηθείτε αν θέλετε να το ανοίξετε.



Εικόνα 16. Μήνυμα για χρήση GPS

4.3 Αρχική οθόνη

Όταν συνδεθείτε θα ανακατευθυνθείτε στην αρχική οθόνη. Εκεί έχετε τρεις επιλογές:

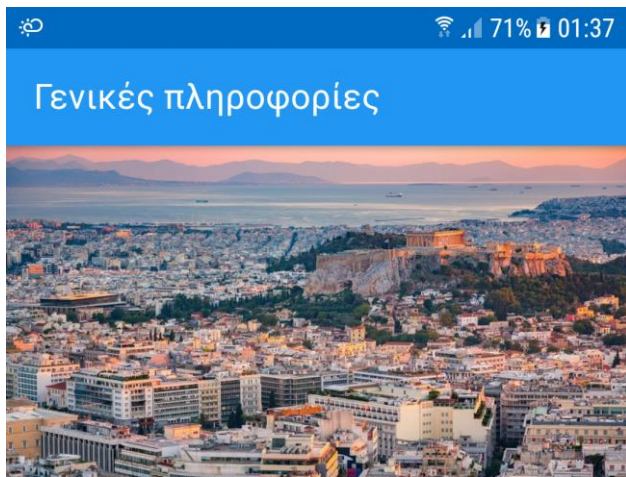
- Να μάθετε βασικές πληροφορίες για την Αθήνα πατώντας το κουμπί «Γενικές πληροφορίες».
- Να δείτε όλα τα μέρη που είναι διαθέσιμα στην εφαρμογή πατώντας το κουμπί «Μέρη».
- Να δείτε τα μέρη που έχετε αποθηκεύσει ως αγαπημένα σας, πατώντας το κουμπί «Αγαπημένα».



Εικόνα 17. Αρχική οθόνη

4.3.1 Γενικές πληροφορίες

Σε αυτή την περιοχή μπορείτε να διαβάσετε βασικές πληροφορίες για την Αθήνα.



Σχετικά με την Αθήνα

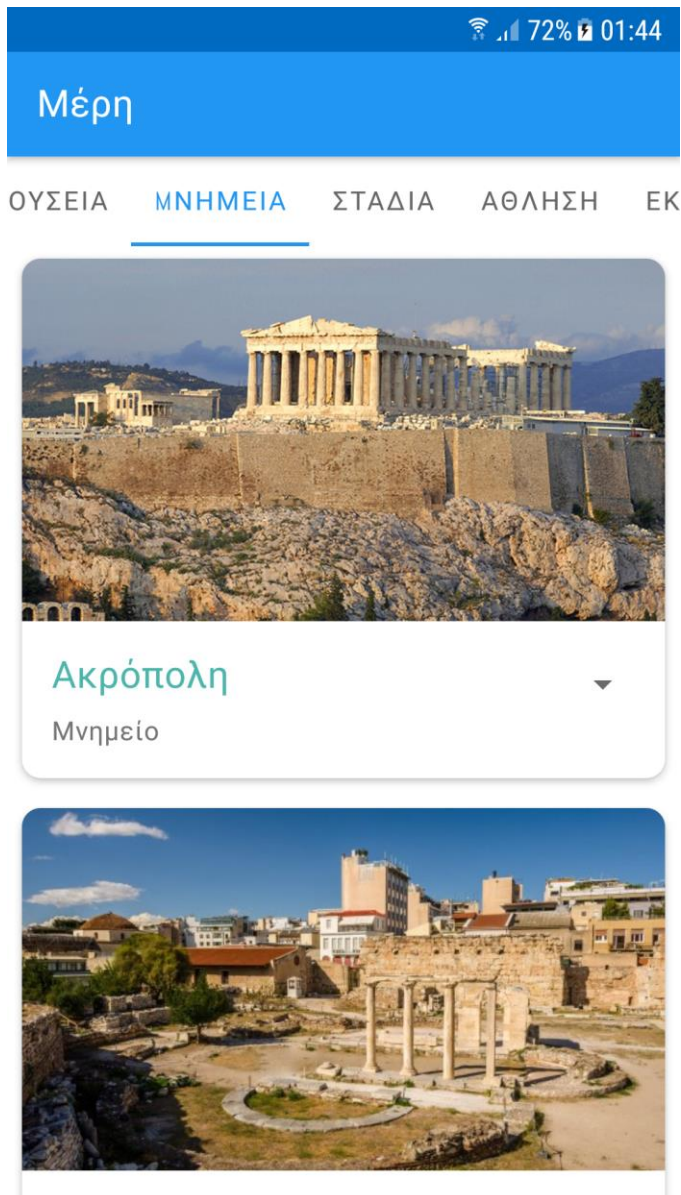
Η Αθήνα είναι η πρωτεύουσα της Ελλάδας από το 1834 και η μεγαλύτερη και πιο πυκνοκατοικημένη πόλη της χώρας. Βρίσκεται στην Αττική, στην ανατολική Στερεά Ελλάδα, και είναι από τις αρχαιότερες πόλεις του κόσμου, με την καταγεγραμμένη ιστορία της να φθάνει ως το 3.200 π.Χ.

Μια από τις αρχαιότερες πόλεις του κόσμου, η Αθήνα είναι γνωστή ως η γενέτειρα της Δημοκρατίας. Με ιστορία πάνω από 3.000 χρόνια, η Αθήνα είναι η καλύτερη πόλη για αξιοθέατα. Σύμφωνα με το μύθο, η πόλη πήρε το όνομά της από την Αθηνά, τη θεά της Σοφίας και κόρη του Δία. Ήταν η προστάτιδα της πόλης και στην πραγματικότητα, ο ναός του Παρθενώνα στην Ακρόπολη είναι αφιερωμένος σε αυτήν.


Εικόνα 18. Γενικές πληροφορίες για την Αθήνα

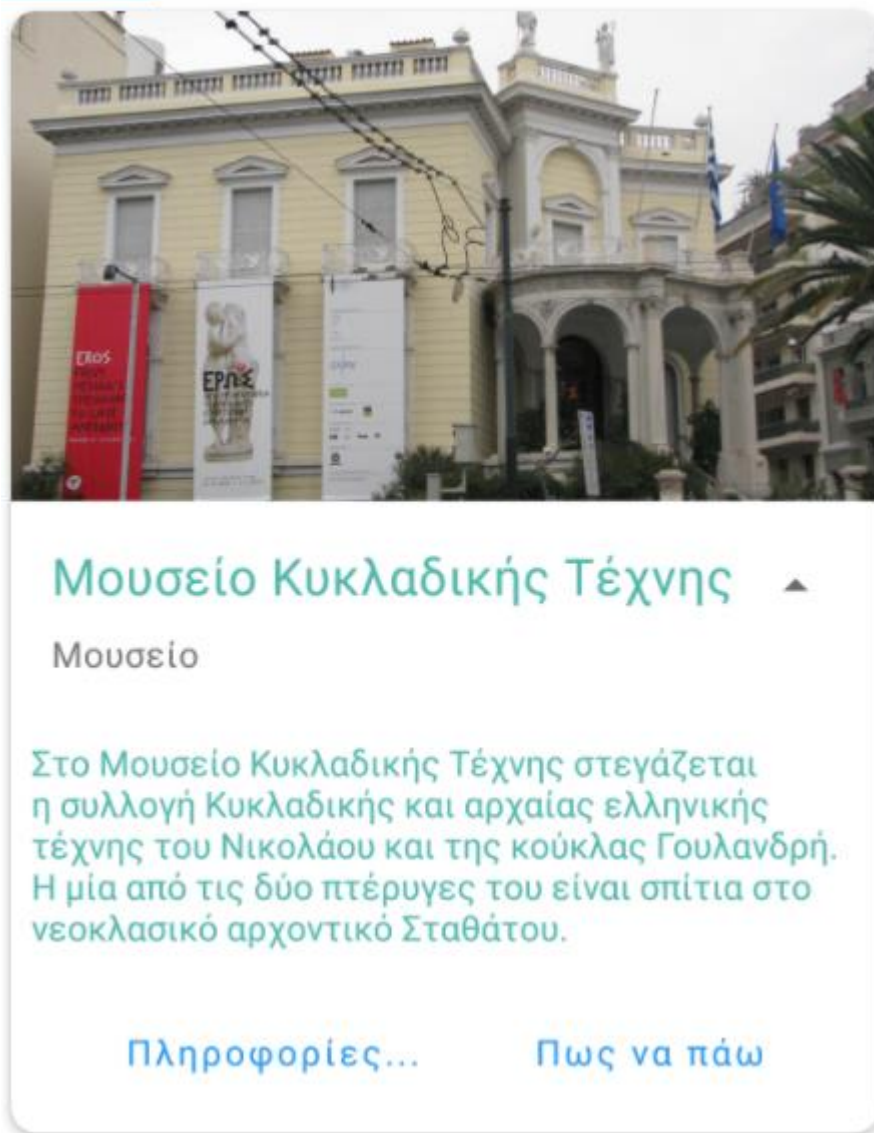
4.3.2 Διαθέσιμα μέρη της εφαρμογής

Σε αυτή την περιοχή μπορείτε να δείτε όλα τα διαθέσιμα μέρη της εφαρμογής ανά κατηγορία. Για να δείτε την κατηγορία που θέλετε πατήστε πάνω της, στο μενού που βρίσκεται στο πάνω μέρος της οθόνης.



Εικόνα 19. Διαθέσιμα μέρη της εφαρμογής

Πατώντας το κουμπί  μπορείτε να δείτε βασικές πληροφορίες για τον προορισμό, να επισκεφθείτε την ιστοσελίδα του πατώντας του κουμπί «Πληροφορίες...» και να δείτε οδηγίες για το πώς να πάτε εκεί πατώντας το κουμπί «Πώς να πάω».



Εικόνα 20. Πληροφορίες για τον προορισμό


4.3.3 Αγαπημένα

Σε αυτή την περιοχή μπορείτε να δείτε τα αγαπημένα σας μέρη. Για να τα διαγράψετε πατήστε πάνω στην καρδιά.



Εικόνα 21. Αγαπημένα

4.4 Δημιουργία προφίλ

Πατώντας το κουμπί  πάτε στην φόρμα συμπλήρωσης του προφίλ σας.



My Tour Guide

Ενδιαφέροντα

✕ Πολιτισμός

Μουσεία

Μνημεία

+ Αθλητισμός

+ Θρησκεία

+ Διασκέδαση

Ηλικιακή ομάδα

Εφηβος/
Ενήλικας

Μεταφορά

Απροσδιόριστ
ο

☒ Εχω μικρά παιδιά

☐ Εχω καροτσάκι

Μέγιστη απόσταση(σε χλμ) ☐

🏠

👤

Προφίλ


🚶

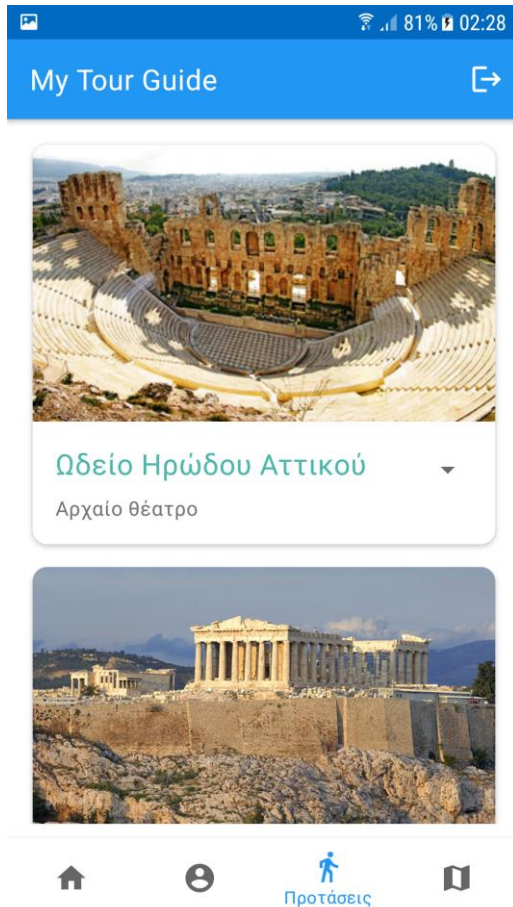
📖

Εικόνα 22. Προφίλ

Πατώντας πάνω στις κατηγορίες των ενδιαφερόντων, εμφανίζονται οι επιλογές ανά κατηγορία. Επίσης μπορείτε να συμπληρώσετε την ηλικιακή σας ομάδα, τον τρόπο μετακίνησης που επιθυμείτε, αν έχετε παιδιά και αν κουβαλάτε καροτσάκι καθώς και την μέγιστη απόσταση που θέλετε να διανύσετε.


4.5 Προτάσεις

Πατώντας το κουμπί  μπορείτε να δείτε την διαδρομή που σας προτείνουμε να ακολουθήσετε.



Εικόνα 23. Προτάσεις

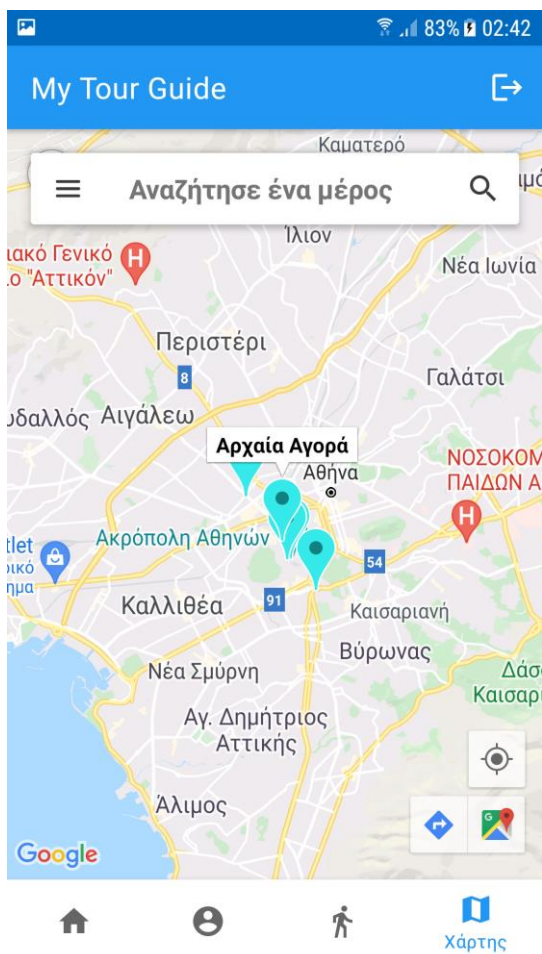
4.6 Χάρτης

Πατώντας το κουμπί  πάτε στον χάρτη. Με κυανά σημάδια μπορείτε να δείτε τα προτεινόμενα σημεία. Επίσης στο άσπρο πλαίσιο, στο πάνω μέρος του χάρτη μπορείτε να

αναζητήσετε όποιο μέρος θέλετε. Πατώντας το



πάτε στο μενού όπου μπορείτε να



Εικόνα 24. Χάρτης





Κεφάλαιο 5^ο

5 Συμπεράσματα

Μέσα από την διαδικασία της ανάλυσης και της σχεδίασης της εφαρμογής βγάλαμε κάποια συμπεράσματα σχετικά με την υλοποίηση τόσο εφαρμογών με θέμα την εξατομικευμένη ηλεκτρονική περιήγηση, όσο και για την ανάπτυξη εφαρμογών για android κινητά γενικότερα. Όσο αφορά την εξατομικευμένη περιήγηση, κάποιος που έχει ως σκοπό να υλοποιήσει μια εφαρμογή που θα βοηθήσει στον μέγιστο βαθμό τον χρήστη, οφείλει να αναλογιστεί όλες τις ανάγκες του χρήστη και να βρει τον σωστό τρόπο να εξάγει τις πληροφορίες που χρειάζεται η εφαρμογή για να του προτείνει να καλύτερα μέρη. Εμείς δημιουργήσαμε μια φόρμα προφίλ εύχρηστη, η οποία μπορεί να συμπληρωθεί γρήγορα και ζητάει βασικές πληροφορίες από τον χρήστη. Παρόλο που οι επιλογές που προσφέρει η φόρμα του προφίλ είναι αρκετά στατικές, οι πληροφορίες που εξάγουμε είναι αρκετές, ώστε ο αλγόριθμος ομοιότητας να προτείνει μέρη που σίγουρα θα προσφέρουν στον εκάστοτε χρήστη μια ευχάριστη εμπειρία. Επιπλέον προσφέρουμε στον χρήστη και δευτερεύουσες λειτουργίες, όπως πρόσβαση σε χάρτη και σε βασικές πληροφορίες, για να μπορεί ο χρήστης να περιηγηθεί πιο εύκολα. Τέλος δώσαμε ιδιαίτερη βαρύτητα στον σχεδιασμό του γραφικού περιβάλλοντος της εφαρμογής, με αποτέλεσμα να είναι αρκετά εύχρηστη και γρήγορη στην χρήση.

Φυσικά, υπάρχει περιθώριο για αρκετές βελτιώσεις και επεκτάσεις. Μια σημαντική λειτουργία που θα συνέβαλε στην εύρεση ταιριαστών προορισμών για τον χρήστη, είναι η βαθμολόγηση από τον ίδιο τον χρήστη της κάθε πρότασης που εμφανίζει η εφαρμογή. Έτσι εφαρμόζοντας έναν αλγόριθμο *deep learning* και συγκρίνοντας τις προτιμήσεις των χρηστών με παρόμοια προφίλ, θα μπορούσαμε να βελτιστοποιήσουμε τις προτάσεις μας. Επίσης, σκοπεύουμε να δώσουμε την δυνατότητα στους χρήστες να επικοινωνούν μεταξύ τους μέσω κάποιου φόρουμ, που θα υπάρχει εντός της εφαρμογής, ώστε να ανταλλάσσουν εμπειρίες. Τέλος στην λίστα μας βρίσκεται και η επέκταση της εφαρμογής μας και σε άλλες συσκευές όπως σε σταθερούς υπολογιστές, σε iOS λειτουργικό καθώς και στο web.



6 Βιβλιογραφικές Πηγές

1. Θέατρο Badminton ,
https://upload.wikimedia.org/wikipedia/commons/thumb/b/b7/Badminton_Theater_aud,
https://el.wikipedia.org/wiki/%CE%98%CE%AD%CE%B1%CF%84%CF%81%CE%BF_Badminton
2. Ακρόπολη. <https://i2.wp.com/www.greecetaxitours.gr/wp-content/uploads/2017/11/01-Acropolis.jpg?resize=1000%2C700> ,
<https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
3. Ναός Αγίου Διονυσίου
Αεροπαγήτου. https://lh3.googleusercontent.com/proxy/yX_DWUH0-S2miK-AhGpStWAAybqXTzQ7GhOzt-a_H1BL2G_zkz-O1nHmownXtuFGs2nhLi4oLgVsrszXxRPhx1DCDPK_bYxbdZjLXuAcZwICMZ40ze-gHKQ-ApV8QKu1sUO9IrvT4WKx1UDC6AoW-MOFfQ ,
<http://www.athinodromio.gr/%CE%BF%CE%B9-%CF%80%CE%BF%CE%BB%CF%8D-%CE%BC%CE%B5%CE%B3%CE%AC%CE%BB%CE%B5%CF%82-%CE%B5%CE%BA%CE%BA%CE%BB%CE%B7%CF%83%CE%AF%CE%B5%CF%82-%CF%84%CE%B7%CF%82-%CE%B1%CE%B8%CE%AE%CE%BD%CE%B1%CF%82/#.YOL1KegzaUk>
4. Αγία Ειρήνη. http://www.religiousgreece.gr/athens-attica/-/asset_publisher/lpcrESIL5iOO/content/agia-eirene ,
[https://el.wikipedia.org/wiki/%CE%95%CE%BA%CE%BA%CE%BB%CE%B7%CF%83%CE%AF%CE%B1_%CE%91%CE%B3%CE%AF%CE%B1%CF%82_%CE%95%CE%B9%CF%81%CE%AE%CE%BD%CE%B7%CF%82_\(%CE%9F%CE%B4%CF%8C%CF%82_%CE%91%CE%B9%CF%8C%CE%BB%CE%BF%CF%85\)](https://el.wikipedia.org/wiki/%CE%95%CE%BA%CE%BA%CE%BB%CE%B7%CF%83%CE%AF%CE%B1_%CE%91%CE%B3%CE%AF%CE%B1%CF%82_%CE%95%CE%B9%CF%81%CE%AE%CE%BD%CE%B7%CF%82_(%CE%9F%CE%B4%CF%8C%CF%82_%CE%91%CE%B9%CF%8C%CE%BB%CE%BF%CF%85))
5. Ναός Αγίας Τριάδος Πειραιώς. https://www.orthodoxianewsagency.gr/wp-content/uploads/2020/03/fd8795150c84c082f779efb1fd13a720_XL.jpg ,
<http://www.athinodromio.gr/%CE%BF%CE%B9-%CF%80%CE%BF%CE%BB%CF%8D-%CE%BC%CE%B5%CE%B3%CE%AC%CE%BB%CE%B5%CF%82-%CE%B5%CE%BA%CE%BA%CE%BB%CE%B7%CF%83%CE%AF%CE%B5%CF%82-%CF%84%CE%B7%CF%82-%CE%B1%CE%B8%CE%AE%CE%BD%CE%B1%CF%82/#.YOL6eOgzaUm>
6. Ναός Αγίου Παντελεήμονα.
<https://tetysolou.wordpress.com/2018/07/27/%CE%BF-%CE%AC%CE%B3%CE%B9%CE%BF%CF%82-%CF%80%CE%B1%CE%BD%CF%84%CE%B5%CE%BB%CE%B5%CE%AE%CE%BC%CE%BF%CE%BD%CE%B1%CF%82-%CF%83%CF%84%CE%B7%CE%BD-%CE%B1%CF%87%CE%B1%CF%81%CE%BD%CF%8E%CE%BD/> ,
http://www.religiousgreece.gr/athens-attica/-/asset_publisher/lpcrESIL5iOO/content/agios-panteleemon-acharnon-
7. Μουσείο Ακρόπολης. https://ellaktor.com/wp-content/uploads/2019/06/pegasus_LARGE_t_200_11150132_type13021-720x450.jpg ,
https://el.wikipedia.org/wiki/%CE%9C%CE%BF%CF%85%CF%83%CE%B5%CE%AF%CE%BF_%CE%91%CE%BA%CF%81%CF%8C%CF%80%CE%BF%CE%BB%CE%B7%CF%82



8. Πλάκα. <https://www.athensphotonews.gr/wp-content/uploads/2019/05/anafiotika-akropolis-plaka.jpg> , <https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
9. Αρχαία Αγορά. <https://media-cdn.tripadvisor.com/media/photo-s/10/fb/f9/80/ancient-agera-of-athens.jpg> , <https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
10. Λόφος Άρειου Πάγου.
https://www.trip2athens.com/userfiles/photos/attractions/A/A1/areios_pagos_01.jpg ,
<https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
11. Athens Metro Mall.
https://www.athensmetromall.gr/photos/c_574px_558px/pages/201911/athensmetro_mall_stores.jpg , <https://www.google.com/>
12. Μητροπολιτικός Ναός. <https://athensattica.com/wp-content/uploads/2017/10/Athens-Metropolitan-Cathedral.jpg> , <https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
13. Άλσος Βεΐκου. <https://www.veikoutrail.gr/images/slides/slide5.jpg> ,
https://el.wikipedia.org/wiki/%CE%86%CE%BB%CF%83%CE%BF%CF%82_%CE%92%CE%B5%CE%90%CE%BA%CE%BF%CF%85
14. Μουσείο Μπενάκη.
[https://www.gtp.gr/showphoto.asp?FN=/MGfiles/travel/image41734\[6985\].jpg&w=650&H=370](https://www.gtp.gr/showphoto.asp?FN=/MGfiles/travel/image41734[6985].jpg&w=650&H=370) , <https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
15. Βυζαντινό Μουσείο. <https://museumfinder.gr/wp-content/uploads/-%CE%BA%CE%B1%CE%B9-%CE%A7%CF%81%CE%B9%CF%83%CF%84%CE%B9%CE%B1%CE%BD%CE%B9%CE%BA%CF%8C-%CE%9C%CE%BF%CF%85%CF%83%CE%B5%CE%AF%CE%BF-e1515067910557.jpg> ,
<https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
16. Ερμού. https://www.hotelstanley.gr/wp-content/uploads/2019/04/Ermou_shopping_district.jpg
17. Φιλαδέλφεια. <https://www.kosmosnf.gr/wp-content/uploads/2018/12/sardeon.jpg> ,
https://el.wikipedia.org/wiki/%CE%9D%CE%AD%CE%B1_%CE%A6%CE%B9%CE%BB%CE%B1%CE%B4%CE%AD%CE%BB%CF%86%CE%B5%CE%B9%CE%B1
18. Φλοίσβος Μαρίνα. https://www.nou-pou.gr/wp-content/uploads/2018/06/floisvos_740x450-1-970x590.jpg ,
<https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
19. Γκάζι.
https://www.athensmagazine.gr/photos/w_800px/articles/201006a/d3d28820e9f1b99



- [960efeab593ea9bff.jpg](#) , [https://el.wikipedia.org/wiki/%CE%93%CE%BA%CE%AC%CE%B6%CE%B9_\(%CF%83%CF%85%CE%BD%CE%BF%CE%B9%CE%BA%CE%AF%CE%B1\)](https://el.wikipedia.org/wiki/%CE%93%CE%BA%CE%AC%CE%B6%CE%B9_(%CF%83%CF%85%CE%BD%CE%BF%CE%B9%CE%BA%CE%AF%CE%B1))
20. Golden Hall. <https://www.skai.gr/sites/default/files/styles/large/public/2020-03/golden-hall-pigi-fb.jpg?itok=aicqqcKo> , <https://www.google.com/>
21. Μουσείο Κυκλαδίτικης Τέχνης. <https://www.athenstransport.com/wp-content/uploads/2017/07/Goulandris-Museum-of-Cycladic-Art-Athens.jpg> , <https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
22. Αθλητικό Κέντρο Γκράβας. <https://www.news247.gr/img/5899/7645318/469000/o/660/0/grava.jpg>
23. Θέατρο Κούκλας της Ιρίνα Μπόικο. <https://www.kathemeragoneis.com/wp-content/uploads/2020/10/Facebook-Post-940x788-px-5-2.jpeg>
24. Ωδείο Ηρώδου Αττικού. http://www.wondergreece.gr/v1/el/Perioxes/N_Attikis/Politismos/Mnimeia_Aksiothea/ta/7812-Wdeio_Irwdoj_Attikoy , https://el.wikipedia.org/wiki/%CE%A9%CE%B4%CE%B5%CE%AF%CE%BF_%CE%97%CF%81%CF%8E%CE%B4%CE%BF%CF%85_%CE%84%CE%BF%CF%85_%CE%91%CF%84%CF%84%CE%B9%CE%BA%CE%BF%CF%8D
25. Ιερός ναός Αγίων Ισιδώρων. <https://choratouaxoritou.gr/wp-content/uploads/2018/05/%CE%B9%CF%83%CE%B9%CE%B4%CF%89%CF%81%CE%BF%CE%B9.jpg> , [https://el.wikipedia.org/wiki/%CE%95%CE%BA%CE%BA%CE%BB%CE%B7%CF%83%CE%AF%CE%B1_%CE%91%CE%B3%CE%AF%CF%89%CE%BD_%CE%99%CF%83%CE%B9%CE%B4%CF%8E%CF%81%CF%89%CE%BD_\(%CE%9B%CF%85%CE%BA%CE%B1%CE%B2%CE%B7%CF%84%CF%84%CF%8C%CF%82\)](https://el.wikipedia.org/wiki/%CE%95%CE%BA%CE%BA%CE%BB%CE%B7%CF%83%CE%AF%CE%B1_%CE%91%CE%B3%CE%AF%CF%89%CE%BD_%CE%99%CF%83%CE%B9%CE%B4%CF%8E%CF%81%CF%89%CE%BD_(%CE%9B%CF%85%CE%BA%CE%B1%CE%B2%CE%B7%CF%84%CF%84%CF%8C%CF%82))
26. Εκκλησία της Παναγίας Καπνικαρέας. <https://1.bp.blogspot.com/-avH1s6woeJ4/XwbPbIOKE9I/AAAAAAAAo1A/T4enN1pGj3ICAc7KkcEdNpz4Y3tIPZ7QgClcBGAsYHQ/s1600/Kapnikarea.jpg> , <http://www.byzantineathens.com/pialphanualphagammaiotaalpha-kappaalphapinuiotakappaalpharhoepsilonalpha.html>
27. Στάδιο Καραϊσκάκης. <http://www.stadia.gr/karaiskaki/karaiskaki-gr.html> , https://el.wikipedia.org/wiki/%CE%A3%CF%84%CE%AC%CE%B4%CE%B9%CE%BF_%CE%93%CE%B5%CF%8E%CF%81%CE%B3%CE%B9%CE%BF%CF%82_%CE%9A%CE%B1%CF%81%CE%B1%CF%8A%CF%83%CE%BA%CE%AC%CE%BA%CE%B7%CF%82
28. Κεραμεικός. <https://media-cdn.tripadvisor.com/media/photo-s/0e/8e/88/ab/keremeikos-cemetery.jpg> , <https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
29. Ιερός Ναός Κωνσταντίνου και Ελένης. <http://www.athinodromio.gr/%CE%BF%CE%B9-%CF%80%CE%BF%CE%BB%CF%8D-%CE%BC%CE%B5%CE%B3%CE%AC%CE%BB%CE%B5%CF%82-%CE%B5%CE%BA%CE%BA%CE>



[%BB%CE%B7%CF%83%CE%AF%CE%B5%CF%82-%CF%84%CE%B7%CF%82-%CE%B1%CE%B8%CE%AE%CE%BD%CE%B1%CF%82/#.YOMMbegzaUn](#)

30. Γήπεδο Απόστολος Νικολαΐδης. <https://newspao.gr/wp-content/uploads/2017/06/leoforos-panathinaikos-newspao.jpg>,
<https://el.wikipedia.org/wiki/%CE%93%CE%AE%CF%80%CE%B5%CE%B4%CE%BF%CE%91%CF%80%CF%8C%CF%83%CF%84%CE%BF%CE%BB%CE%BF%CF%82%CE%9D%CE%B9%CE%BA%CE%BF%CE%BB%CE%B1%CE%90%CE%B4%CE%B7%CF%82>
31. Λόφος Λυκαβηττού. <https://www.skai.gr/sites/default/files/styles/large/public/2019-06/lycabettus20190620.jpg?itok=PGuunX2c>,
<https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
32. McArthurGlen. https://el.aegeanair.com/milesandbonus/-/media/milesandbonus/partnesimages_withoutlogo/mcarthurglen_designer_outlets.jpg
33. Μοναστηράκι. <https://www.athenstransport.com/wp-content/uploads/2017/07/Monastiraki-Athens-768x512.jpg>,
<https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
34. Αγία Ειρήνη Χρυσοβαλάντου, <https://cdn.enimerotiko.gr/media/2019/07/to-epivlitiko-monastiri-tis-agias-eirinis-chrysovalantoy-sti-lykovrysi.jpeg>,
<https://www.ekklisiaonline.gr/nea/agia-irini-chrysovalantou-to-paleoimerologitiko-monastiri-sti-lykovrysi/>
35. Μονή Ασωμάτων.
http://www.wondergreece.gr/v1/el/Perioxes/N_Attikis/Politismos/Ekklesies_Monastiria/7882-Iera_Moni_Aswma_twn_Petra_ki,
<https://el.wikipedia.org/wiki/%CE%9C%CE%BF%CE%BD%CE%AE%CE%A0%CE%B5%CF%84%CF%81%CE%AC%CE%BA%CE%B7>
36. Μονή Καισαριανής.
https://www.google.com/url?sa=i&url=http%3A%2F%2Fimkby.gr%2Fnews%2F%3Fp%3D904&psig=AOvVaw2g0uVuW8DPt5_GJuqRBObq&ust=1625580108541000&source=images&cd=vfe&ved=0CAoQjRxqFwoTCLiYpf-LzPECFQAAAAAdAAAAABAF,
<https://el.wikipedia.org/wiki/%CE%9C%CE%BF%CE%BD%CE%AE%CE%9A%CE%B1%CE%B9%CF%83%CE%B1%CF%81%CE%B9%CE%B1%CE%BD%CE%AE%CF%82>
37. Μονή Ιωάννη Προδρόμου Καισαριανής. http://www.religiousgreece.gr/athens-attica/-/asset_publisher/lpcrESIL5iOO/content/iera-mone-agiou-ioannou-prodrom-1
38. Ιερά Μονή Αγίου Ιωάννη Θεολόγου. http://www.religiousgreece.gr/athens-attica/-/asset_publisher/lpcrESIL5iOO/content/iera-mone-agiou-ioannou-theologou,
<http://www.byzantineathens.com/alphagammaiotaomicronsigma-iotaomegaalphanunuetasigma-thetaepsilonomiconlambdaomicrongammaomicronsigma.html>



39. Αρχαιολογικό Μουσείο. https://upload.wikimedia.org/wikipedia/commons/thumb/8/83/National_Archaeological_Museum_Athens_09.jpg/300px-National_Archaeological_Museum_Athens_09.jpg , <https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
40. Εθνικό Θέατρο. https://www.n-t.gr/pictures/b/b_21146_main.jpg , https://el.wikipedia.org/wiki/%CE%95%CE%B8%CE%BD%CE%B9%CE%BA%CF%8C_%CE%98%CE%AD%CE%B1%CF%84%CF%81%CE%BF_%CE%95%CE%BB%CE%BB%CE%AC%CE%B4%CE%B1%CF%82
41. Εθνικός Κήπος. <https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
42. Αθηναϊκή Τριλογία. <https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
43. Στάδιο Νέας Σμύρνης. <https://neasmyrni.gr/wp-content/uploads/2018/12/neasmyrni-stadio-01.jpg> , https://el.wikipedia.org/wiki/%CE%A3%CF%84%CE%AC%CE%B4%CE%B9%CE%BF_%CE%9D%CE%AD%CE%B1%CF%82_%CE%A3%CE%BC%CF%8D%CF%81%CE%BD%CE%B7%CF%82
44. Ολυμπιακό Στάδιο Αθηνών , <https://athensattica.com/wp-content/uploads/2018/03/stadium.jpg> , https://el.wikipedia.org/wiki/%CE%9F%CE%BB%CF%85%CE%BC%CF%80%CE%B9%CE%B1%CE%BA%CF%8C_%CE%91%CE%B8%CE%BB%CE%B7%CF%84%CE%B9%CE%BA%CF%8C_%CE%9A%CE%AD%CE%BD%CF%84%CF%81%CE%BF_%CE%91%CE%B8%CE%B7%CE%BD%CF%8E%CE%BD_%C2%AB%CE%A3%CF%80%CF%8D%CF%81%CE%BF%CF%82_%CE%9B%CE%BF%CF%8D%CE%B7%CF%82%CE%B5
45. Στάδιο Νίκος Γκάλης. https://el.wikipedia.org/wiki/%CE%9F%CE%BB%CF%85%CE%BC%CF%80%CE%B9%CE%B1%CE%BA%CF%8C_%CE%91%CE%B8%CE%BB%CE%B7%CF%84%CE%B9%CE%BA%CF%8C_%CE%9A%CE%AD%CE%BD%CF%84%CF%81%CE%BF_%CE%91%CE%B8%CE%B7%CE%BD%CF%8E%CE%BD_%C2%AB%CE%A3%CF%80%CF%8D%CF%81%CE%BF%CF%82_%CE%9B%CE%BF%CF%8D%CE%B7%CF%82%CE%B5 , https://el.wikipedia.org/wiki/%CE%9A%CE%BB%CE%B5%CE%B9%CF%83%CF%84%CF%8C_%CE%93%CE%AE%CF%80%CE%B5%CE%B4%CE%BF_%CE%9C%CF%80%CE%AC%CF%83%CE%BA%CE%B5%CF%84_%CE%9D%CE%AF%CE%BA%CE%BF%CF%82_%CE%93%CE%BA%CE%AC%CE%BB%CE%B7%CF%82_%CF%84%CE%BF%CF%85_%CE%9F%CE%91%CE%9A%CE%91
46. Παναθηναϊκό Στάδιο. <https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
47. Παραμυθοχώρα. <https://www.culturenow.gr/wp-content/uploads/2017/09/paramythoxwra-2017-2018-e1504692979559.jpg>



48. Ιερά Μονή Προφήτη Ηλία. https://www.panelinios.gr/images/photo_big/149110.jpg , <https://vresonline.gr/iero-isixastirio-profitou-ilia-ekklisia-galatsi-18295.html>
49. River West. <https://www.riverwest.gr/wp-content/uploads/2020/05/unnamed-1.jpg> , https://www.google.com/search?q=river+west&rlz=1C1GCEA_enGR837GR837&oq=river+w&aqs=chrome.1.69i57j35i39j0i131i433j46i175i199j46j69i60l3.5318j0j7&sourceid=chrome&ie=UTF-8
50. Στάδιο Ειρήνης και Φιλίας . <http://www.stadia.gr/sef/sef-gr.html> , https://el.wikipedia.org/wiki/%CE%A3%CF%84%CE%AC%CE%B4%CE%B9%CE%BF_%CE%95%CE%B9%CF%81%CE%AE%CE%BD%CE%B7%CF%82_%CE%BA%CE%B1%CE%B9_%CE%A6%CE%B9%CE%BB%CE%AF%CE%B1%CF%82
51. Σταύρος Νιάρχος.
https://www.ktirio.gr/media/k2/items/cache/5c44a7ad31cda12127dfd90a7d4feb9e_X_L.jpg , <https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
52. Πλατεία Συντάγματος.
https://upload.wikimedia.org/wikipedia/commons/thumb/2/28/Syntagma_Square_%282015%29.jpg/320px-Syntagma_Square_%282015%29.jpg , <https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
53. Ναός του Ηφαίστου. <https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
54. Ναός του Ολυμπίου Διός. <https://www.athenstransport.com/2017/07/places-to-visit-in-athens-greece-public-transport/>
55. The Mall Athens.
https://www.trip2athens.com/userfiles/photos/attractions/L/L8/the_mall_athens_02.jpg?w=619&h=348&mode=crop&scale=both&quality=80 , https://el.wikipedia.org/wiki/The_Mall_Athens
56. Θέατρο Παλλάς. https://www.viva.gr/tickets/getattachment/e9e8c85a-1c39-4136-a329-1f6ae217331a/venue_banner.png , <https://www.mytheatro.gr/theatro-pallas/>
57. Πάρκο Τρίτση.
https://www.naturagraeca.com/ws/upload/lib/Oikotopoi/mikroi_oikotopoi/parko_tritsi/arhikitopio_ao3.jpg , https://www.wikiwand.com/el/%CE%9C%CE%B7%CF%84%CF%81%CE%BF%CF%80%CE%BF%CE%BB%CE%B9%CF%84%CE%B9%CE%BA%CF%8C_%CE%A0%CE%AC%CF%81%CE%BA%CE%BF_%C2%AB%CE%91%CE%BD%CF%84%CF%8E%CE%BD%CE%B7%CF%82_%CE%A4%CF%81%CE%AF%CF%84%CF%83%CE%B7%CF%82%C2%BB
58. Παναγία των Βρουόλων. http://photos.wikimapia.org/p/00/04/47/40/00_big.jpg
59. Παρθενώνας.
https://www.history.com/image/ar_1:1%2Cc_fill%2Ccs_srgb%2Cfl_progressive%2Cq_a



uto:good%2Cw_1200/MTU3ODc5MDg2NzAwNTcwMzM1/greece-attica-athens-acropolis-listed-as-world-heritage-by-unesco.jpg

60. Αθήνα. <https://ychef.files.bbci.co.uk/976x549/p07qsmysr.jpg>
61. Use Case Diagram. <https://online.visual-paradigm.com/diagrams/tutorials/use-case-diagram-tutorial/>
62. Sequence Diagram. <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/>
63. Android OS. <https://coolweb.gr/android-ti-einai/>
64. Παναγιωτόπουλος, Ι.-Χ., Αποστόλου, Δ.(2012), Αρχές Προγραμματισμού με C/C++. Πειραιάς: Αυτοέκδοση
65. Firebase. <https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0>
66. API. <https://www.grow-digital.gr/ti-einai-ena-api/>
67. XML. <https://support.microsoft.com/el-gr/topic/%CF%84%CE%BF-xml-%CE%B3%CE%B9%CE%B1-%CF%84%CE%BF%CF%85%CF%82-%CE%B1%CE%BC%CF%8D%CE%B7%CF%84%CE%BF%CF%85%CF%82-a87d234d-4c2e-4409-9cbc-45e4eb857d44>
68. Activities. <https://developer.android.com/guide/components/activities/intro-activities>
69. Jaccard Index. https://en.wikipedia.org/wiki/Jaccard_index