

ΣΧΟΛΗ ΕΠΙΣΤΗΜΗΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

AUGMENTED REALITY MOBILE APPLICATION FOR THE DEVELOPMENT OF LOCATION-BASED INTERACTIVE ROUTES IN THE OPEN LANDSCAPE



ΕΛΕΥΘΕΡΙΟΣ ΑΡΙΣΤΕΙΔΗΣ ΑΛΕΞΙΟΥ

A.M.: Π2017143

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:

ΤΡΙΜΕΛΗΣ ΕΠΙΤΡΟΠΗ:

ΔΗΜΗΤΡΙΟΣ ΡΙΓΓΑΣ

ΦΟΙΒΟΣ ΜΥΛΩΝΑΣ

ΕΛΕΝΗ ΧΡΙΣΤΟΠΟΥΛΟΥ

ΚΕΡΚΥΡΑ

ΣΕΠΤΕΜΒΡΙΟΣ 2022

Περιεχόμενα

Εισαγωγή.....	5
Βιβλιογραφική Επισκόπηση.....	7
1. Εφαρμογές Augmented Reality.....	8
2. Εφαρμογές πλοήγησης.....	10
Ορισμοί.....	22
Android.....	22
1. Gradle.....	22
2. View.....	22
3. ViewGroup.....	23
4. Activity.....	24
5. Activity Lifecycle.....	25
6. Fragments.....	27
7. Adapters.....	27
8. RecyclerView.....	27
9. FirestoreRecyclerViewAdapter.....	28
10. Bottom Navigation View.....	29
11. Menus.....	29
12. Menu Item.....	29
13. Menu Group.....	30
ArCore.....	30
1. Motion Tracking.....	30
2. Planes.....	30
3. Anchors.....	31
4. World Space & Pose.....	31
Flutter.....	31
1. Το Framework και τα Widgets.....	31
2. State.....	32
3. Stateless & Statefull.....	32

4. Build.....	33
5. Material App/Design.....	33
6. Navigator	33
7. Βασικά Material Widget.....	34
8. Δημοφιλή Widget Γενικής Χρήσης.....	35
9. Provider	36
10. Stream	36
11. Stream Provider	36
12. Stream Builder	36
13. Inkwell.....	37
14. Card Widget.....	37
15. Dialog Widget.....	37
16. Controllers.....	38
17. Form Widget	38
18. Checkbox.....	38
19. Dropdown.....	39
Σχεδίαση.....	40
Επιλογή Framework	40
Σχεδίαση Android	41
1. AR βιβλιοθήκες	41
2. GPS - Χάρτες.....	41
3. Cloud Server.....	42
4. Σελίδα για Admin.....	43
Περιορισμοί και Δυσκολίες	44
Κατάργηση της Sceneform	44
Προϋποθέσεις ARCore.....	46
Μεθοδολογία.....	47
Εφαρμογή Android	47
1. Αρχικοποίηση περιβάλλοντος	47

2. Βασικές σελίδες	48
3. Thread για συνεχή εντοπισμό του χρήστη.....	48
4. Ανταλλαγή δεδομένων μεταξύ Fragment.....	49
5. Δομή των Αντικειμένων	50
Admin Ιστοσελίδα	52
1. Η main	53
2. Αυτόματη Σύνδεση Χρήστη	54
3. Σελίδα Αυθεντικοποίησης χρήστη	54
4. Home Page.....	60
5. Φόρμα για πρόσθεση νέου αντικειμένου	66
Αποτελέσματα.....	71
Βιβλιογραφία.....	78
Διαγράμματα.....	80
Πίνακες	80
Εικόνες	80
Παράρτημα	82

Εισαγωγή

Η παρούσα πτυχιακή λαμβάνει χώρα, σε μια περίοδο ριζικών αλλαγών στις δυνατότητες και τα όρια, στις τεχνολογίες ψηφιακής, μικτής και επαυξημένης πραγματικότητας, ιδίως στις κινητές συσκευές. Ακόμη περισσότερο αναγκαίες κρίνονται όλες αυτές οι δυνατότητες, οι οποίες έρχονται σε μια ευάλωτη εποχή. Μέσα στην πανδημία αλλά και τώρα που φαίνεται να φτάνει προς το τέλος της, η ανάπτυξη νέων ψηφιακών εργαλείων και η ζήτηση ψηφιακών λύσεων εκτοξεύτηκε. Αυτές οι νέες τεχνολογίες είναι απαραίτητο να μελετηθούν. Υπάρχει μια πληθώρα πεδίων της τεχνολογίας, στις οποίες ο άνθρωπος μπορεί να δραστηριοποιηθεί και μπορεί να του φανούν χρήσιμες, βελτιώνοντας έτσι την καθημερινότητα και τον τρόπο με τον οποίο έρχεται σε διάδραση με τον κόσμο γύρω του.

Η ιδέα του Augmented Reality και γενικότερα της μίξης εικονικών κόσμων με τον πραγματικό, δεν είναι καινούργια. Ο συγγραφέας L. Frank Baum ήδη από τις αρχές του εικοστού αιώνα αναφέρει μελλοντικές συσκευές σαν γυαλιά οράσεως τα οποία θα μπορούν να δείχνουν ψηφιακούς ανθρώπους στο πεδίο οράσεως του χρήστη. Ωστόσο, δεν υπήρξε κάποια σημαντική τεχνολογική εξέλιξη στον τομέα μέχρι την δεκαετία του 60. Τότε, ο Morton Heilig έφτιαξε το Sensorama (Media Art Net, n.d.), μια συσκευή που φέρνει τον χρήστη κοντά σε μια οθόνη, η οποία καλύπτει όλο το πρόσωπό του και ακόμα περιέχει στερεοσκοπικά ηχεία, ανεμιστήρες και μια κινητή καρέκλα για να αυξήσει την εμπύθιση του χρήστη στον κόσμο που του παρουσιάζεται. Στις ημέρες μας, το Sensorama θεωρείται μια από τις πρώτες προσπάθειες δημιουργίας μιας Virtual Reality (VR) συσκευής. Το 1990 ο Thomas P. Caudell θέτει για πρώτη φορά τον όρο Επαυξημένη Πραγματικότητα (Augmented Reality ή AR). Δυο χρόνια αργότερα, ο Louis Rosenberg φτιάχνει την πρώτη συσκευή επαυξημένης πραγματικότητας, εν ονόματι Virtual Fixtures. Σκοπός της συσκευής είναι να παρέχει πληροφορίες από σένσορες που αντανακλούν πάνω στον πραγματικό κόσμο, δίνοντας στο χρήστη μια φυσική διάδραση με τις πληροφορίες και κατά συνέπεια την αύξηση της αποδοτικότητας του χρήστη σε διάφορες εργασίες. Από εκεί και πέρα, η τεχνολογία συνέχισε να αναπτύσσεται και να βελτιώνεται. Περίπου πριν 15 χρόνια μια νέα τεχνολογική επανάσταση έφτασε στα χέρια του κοινού. Τα έξυπνα τηλέφωνα άλλαξαν την καθημερινότητα του ανθρώπου αλλά και τον τρόπο που γίνεται η αλληλεπίδραση ανθρώπου και υπολογιστή. Τα smartphones σε συνδυασμό με το διαδίκτυο εκδημοκράτισαν τις πληροφορίες που μπορεί να παρέχει ένας υπολογιστής, χάρη στην απλότητα χρήσης, τη φορητότητα και το αναλογικά μικρότερο

κόστος τους. Χάρη στην δημοτικότητα, τη φορητότητα, την υπολογιστική ισχύ και το μεγάλο αριθμό αισθητήρων που βρίσκονται στα σύγχρονα έξυπνα τηλέφωνα, μια τεράστια ποικιλία νέων εφαρμογών έχει παραχθεί, που προηγουμένως φάνταζε αδύνατη ή ασύμφορη.

Μεταξύ όλων αυτών των εφαρμογών που αναπτύσσονται βρίσκεται και η επauξημένη πραγματικότητα. Μέχρι τώρα, οι συσκευές επauξημένης πραγματικότητας περιορίζονταν σε μεγάλα headset που περιείχαν μια οθόνη για κάθε μάτι και διάφορους αισθητήρες. Αυτά τα headset μπορούσε να τα φορέσει ο χρήστης και συνδέοντας τη συσκευή με καλώδια σε ένα ισχυρό υπολογιστή μπορούσε πλέον ο χρήστης να αλληλεπιδράσει με ψηφιακά αντικείμενα στο φυσικό κόσμο. Άλλα όπως αναφέρθηκε και προηγουμένως, εφόσον τα κινητά σήμερα έχουν όλους τους αισθητήρες που χρειάζονται και επαρκή επεξεργαστική ισχύ, τέθηκε το ερώτημα “γιατί να μην φτιαχτούν κάποιες εφαρμογές AR και εδώ;”.

Η χώρα μας επιδεικνύει μεγάλη τουριστική προσέλευση και ιδιαίτερα τα τελευταία χρόνια έχει επιδείξει μια ραγδαία αύξηση του αριθμού προσελεύσεων τουριστών με αποτέλεσμα νέα εργαλεία που μπορούν να διευκολύνουν την εξuπηρέτηση των τουριστών να βρίσκονται σε μεγάλη ζήτηση. Ήδη έχουν προηγηθεί έρευνες που προσπαθούν να βρουν διαδραστικούς τρόπους με τη χρήση της επauξημένης πραγματικότητας, για να μεταφέρουν πληροφορίες στους επισκέπτες με ένα πιο διασκεδαστικό τρόπο, σαν παιχνίδι, χρησιμοποιώντας μια μεγάλη ποικιλία πολυμέσων, όπου οι παραδοσιακές αντίστοιχες μέθοδοι (π.χ. Οι ξεναγήσεις με ξεναγούς ή τα audio guides) δεν μπορούν να προσφέρουν.

Λαμβάνοντας υπόψιν την εμπορική παράμετρο, έχουν υπάρξει εφαρμογές στον τομέα που χρησιμοποιούν την επauξημένη πραγματικότητα άλλα στη μεγαλύτερη πλειοψηφία, οι εμπειρίες AR που προσφέρονται είναι σύντομες και μικρότερης κλίμακας. Στην πραγματικότητα, δεν υπάρχει κάποια εφαρμογή που μπορούν να αξιοποιήσουν οι επισκέπτες σε μεγάλο βαθμό, για την πλοήγηση και την ξενάγησή τους.

Βιβλιογραφική Επισκόπηση

Τις τελευταίες δεκαετίες υπάρχει μια κλιμάκωση της χρήσης νέων τεχνολογιών στην καθημερινότητα του ανθρώπου. Ειδικά την τελευταία δεκαετία παρατηρείται μια συνεχής χρήση αυτών, σε όλους τους τομείς που αφορούν τον άνθρωπο, από την χρήση ενός smartphone μέχρι μια εγχείρηση εξ αποστάσεως με την βοήθεια της τεχνητής νοημοσύνης(A.I.). Όπως αναφέρθηκε και σε προηγούμενο κεφάλαιο, σε αυτή την εργασία θα γίνει εφαρμογή της επαυξημένης πραγματικότητας για την ανάπτυξη μιας εφαρμογής για κινητές συσκευές, η οποία θα επιτρέπει στους χρήστες να αλληλεπιδρούν με την εφαρμογή και να μαθαίνουν με διαδραστικό τρόπο πληροφορίες για το σημείο που βρίσκονται.

Η επαυξημένη πραγματικότητα έχει χρησιμοποιηθεί αρκετές φορές στο παρελθόν, καθώς έχουν κυκλοφορήσει διάφορες εφαρμογές, οι οποίες παρέχουν οδηγίες και πληροφορίες στο χρήστη, για μουσεία, πανεπιστήμια, εκδηλώσεις και άλλα. Πρωτοπόρα εφαρμογή στον χώρο αποτελεί το Archeoguide (Vlahakis, et al., 2002). Το Archeoguide έκανε χρήση ενός head mounted display (HMD) για να αναπαραστήσει τα κτήρια και τους ανθρώπους στην Αρχαία Ολυμπία. Τελικά, η εφαρμογή δεν προχώρησε λόγω διάφορων τεχνολογικών και πρακτικών περιορισμών της εποχής.

Ωστόσο πιο σύγχρονες έρευνες και εφαρμογές έχουν αποδείξει την βιωσιμότητα AR εφαρμογών στον τομέα του τουρισμού και των ξεναγήσεων. Για παράδειγμα, σε πανεπιστήμια (Andri , Alkawaz, & Sallow, 2018) έχουν υπάρξει πολλές διαφορετικές εφαρμογές που χρησιμοποιούν το AR για ξεναγήσεις αλλά με διαφορετικές τεχνικές υλοποίησης. Κάποιες αποτελούν μέρος μιας εκδήλωσης, άλλες είναι φτιαγμένες σαν παιχνίδια και άλλες είναι πιο γενικής μορφής. Πολλές περιέχουν λειτουργίες για την επιλογή και αναζήτηση των σημείων ενδιαφέροντος ή των πληροφοριών. Ωστόσο, παρά τις διαφορές τους σε κάποιες “δευτερεύοντες” λειτουργίες, σχεδόν όλες παρουσιάζονται να παρέχουν με τον ένα ή τον άλλο τρόπο πληροφορίες για το σημείο ενδιαφέροντος και οδηγίες για την καθοδήγηση του χρήστη. Επίσης, σαν εφαρμογές A.R. παρέχουν στον χρήστη την προβολή δισδιάστατων ή τρισδιάστατων αντικειμένων.

Από την άλλη πλευρά, σε μια εφαρμογή AR, για την ξενάγηση των τουριστών στο εθνικό μουσείο του παλατιού της Κορέας (Park & Kim, 2011), εξαιτίας του μη ακριβούς εντοπισμού του σημείου του χρήστη για μικρές αποστάσεις, χρησιμοποιούν μια αρχιτεκτονική στην οποία το τηλέφωνο στέλνει την εικόνα από το σημείο στο οποίο βρίσκεται ο χρήστης σε ένα σέρβερ, ο οποίος με την σειρά του τρέχει ένα αλγόριθμο

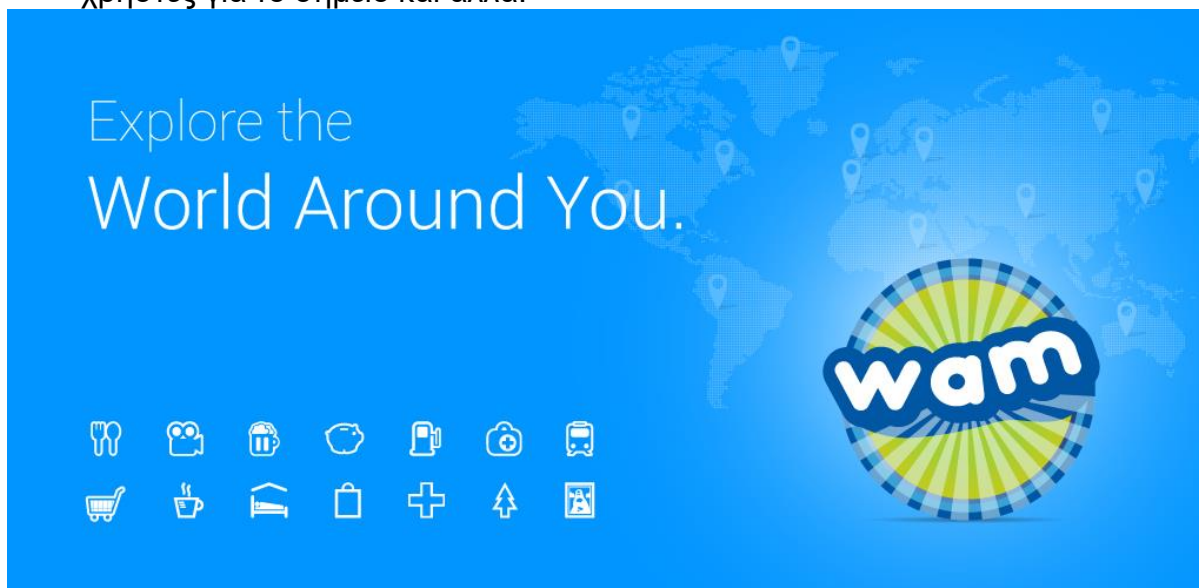
αναγνώρισης για να προσδιορίσει το έκθεμα στο οποίο εστιάζει ο χρήστης, για να του δείξει το σχετικό περιεχόμενο.

Κρίσιμα συστατικά μιας τέτοιας εφαρμογής (Koo, Kim, Kim, Kim, & Cha, 2020) AR φαίνεται να είναι η οπτική αρέσκεια από τον χρήστη και η εύκολη χρήση της τεχνολογίας. Οπτικά, η εφαρμογή πρέπει να τραβά το ενδιαφέρον του χρήστη χωρίς να τον κουράζει ή να τον μπερδεύει. Από την άλλη, η διάδραση πρέπει να γίνεται με μέσα κατανοητά και οικεία προς κάθε χρήστη, ακόμα και αν αυτός έχει μικρή εξοικείωση με την τεχνολογία. Όσο πιο απλή και κατανοητή είναι η εφαρμογή τόσο αυξάνονται και οι πιθανότητες του χρήστη να τη χρησιμοποιήσει αυτοβούλως.

1. Εφαρμογές Augmented Reality

Σήμερα, υπάρχουν πολλές εφαρμογές AR που χρησιμοποιούνται για ξεναγήσεις ή ως διαδραστικά παιχνίδια. Κάποια από τα σημαντικότερα, τα οποία αποτέλεσαν πηγή έμπνευσης και για αυτή την εργασία είναι τα εξής:

- Το WAM (World around me) είναι μια ταξιδιωτική εφαρμογή που χρησιμοποιεί την κάμερα για να δείξει κάποια σημαντικά μέρη ως “Σημεία Ενδιαφέροντος”(Points of Interest). Τα μέρη αυτά χωρίζονται σε 31 κατηγορίες και μπορεί να αποτελούνται από εστιατόρια, ATM, μαγαζιά, σταθμοί λεωφορείων / τρένων κτλ. Ο Χρήστης βλέπει την κατεύθυνση στην οποία βρίσκεται το σημείο και αν πατήσει πάνω του, παρέχονται διάφορες πληροφορίες όπως το τηλέφωνο κρατήσεων ενός εστιατορίου, οι τιμές των δωματίων σε ένα ξενοδοχείο, οι κριτικές από τους χρήστες για το σημείο και άλλα.



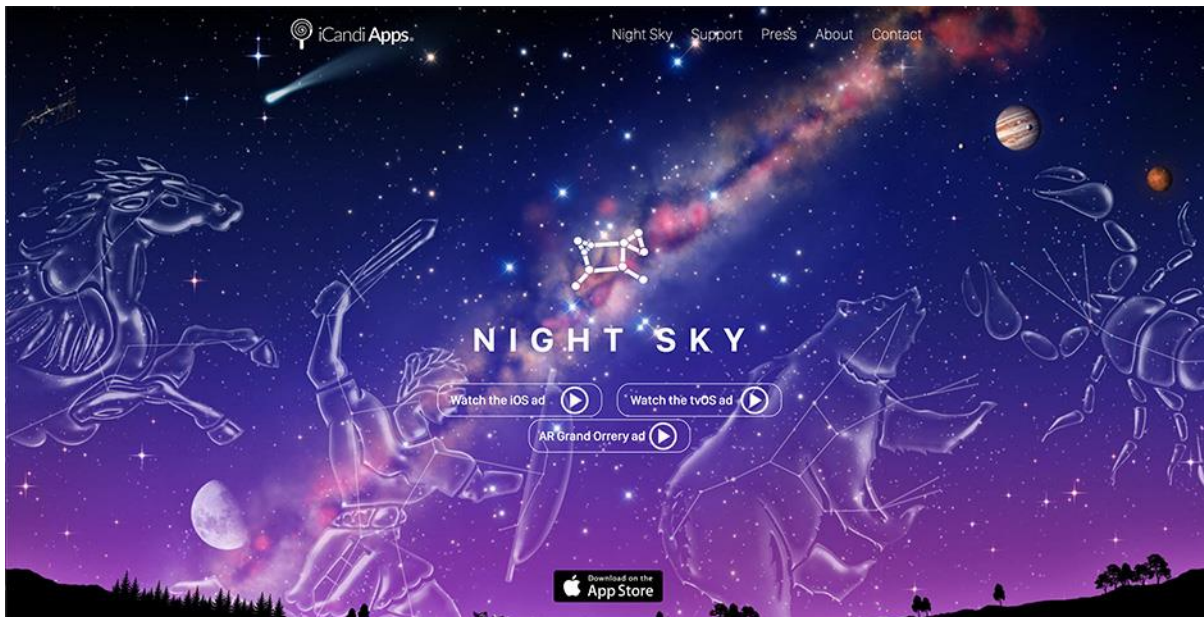
Εικόνα 1: Wam

- Το “Senditur “ σχεδιάστηκε με σκοπό να αναδείξει σημαντικά αξιοθέατα στην Ισπανία. Η εφαρμογή σχεδιάστηκε για πεζοπορία στην εξοχή αλλά περιέχει και σημεία εντός των πόλεων. Εκτός από αξιοθέατα η εφαρμογή δείχνει τα βουνά που υπάρχουν τριγύρω, τις σημαντικές διαδρομές για πεζοπορία, κάποια χρήσιμα σημεία (καταφύγια, βενζινάδικα, φαρμακεία κ.λπ.). Το εικονίδιο του σημείου ενδιαφέροντος είναι διαφορετικό, ανάλογα σε ποια κατηγορία ανήκει το συγκεκριμένο σημείο. Πατώντας πάνω σε ένα σημείο θα εμφανιστούν κάποιες χρήσιμες πληροφορίες για αυτό.



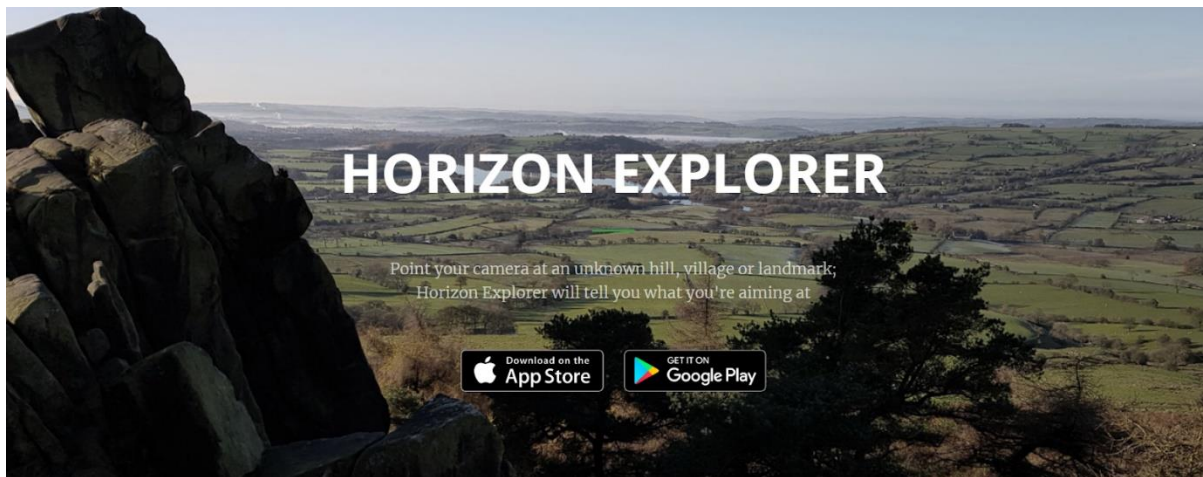
Εικόνα 2: Senditur

- Το “Night Sky” είναι λίγο διαφορετικό, αυτό δεν χρησιμοποιεί AR για να δείξει στον χρήστη σημεία ενδιαφέροντος. Αυτή η εφαρμογή θα μπορούσε κανείς να πει ότι λειτουργεί ως ιδιωτικό πλανητάριο, δείχνει στον χρήστη τους αστερισμούς, άλλα ουράνια σώματα καθώς και διάφορους δορυφόρους, σαν τρισδιάστατα μοντέλα που αν θέλει μπορεί να τα προβάλει στο δωμάτιο που βρίσκεται εκείνη την στιγμή, μέσω της AR λειτουργίας του.



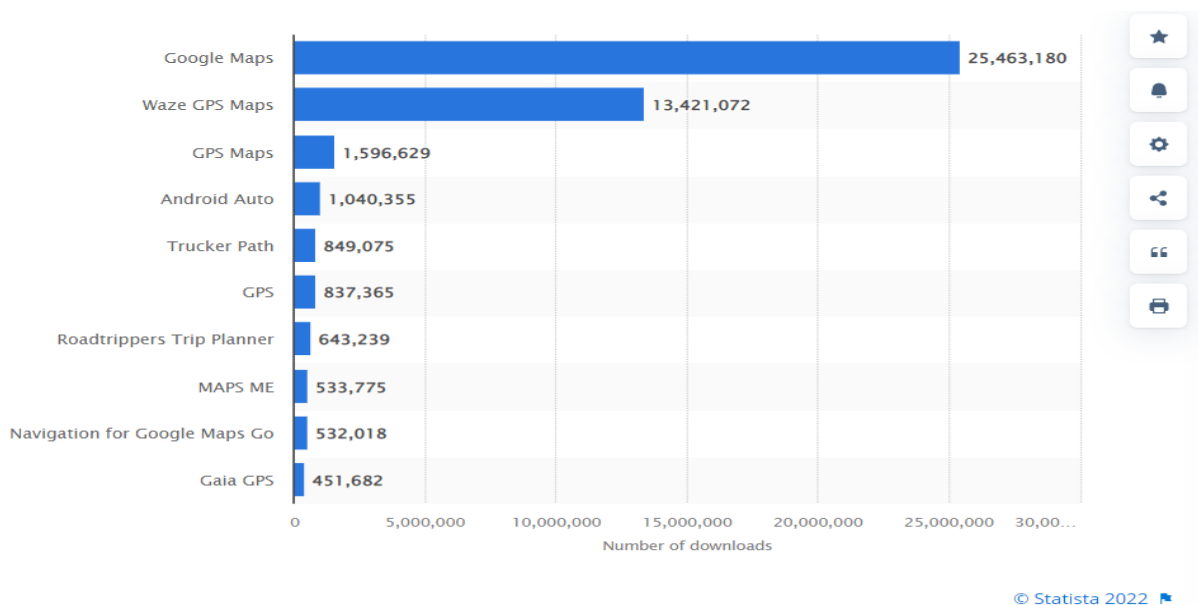
Εικόνα 3: Night Sky

- Μια ακόμα εφαρμογή που επηρέασε την εργασία αποτελεί το “Horizon Explorer AR”. Αυτή, απεικονίζει με ένα βέλος το σημείο ενδιαφέροντος και από πάνω υπάρχει το όνομα του σημείου μαζί με την απόσταση από τον χρήστη και το υψόμετρο στο οποίο βρίσκεται . Κεντράροντάς το, εμφανίζονται στο χρήστη πληροφορίες σχετικά με αυτό. Ακόμη υπάρχει η λειτουργία 3D Map, η οποία επιτρέπει στο χρήστη να δει τα σημεία στο χάρτη. Σημεία μπορούν να αποτελούν κοντινές πόλεις ή χωριά, δρόμοι, βουνά, λίμνες, ακόμη και αεροπλάνα που περνούν κοντά από το χρήστη και για τα οποία παρέχονται και πληροφορίες όπως ο προορισμός, το σημείο εκκίνησης και άλλα.



Εικόνα 4: Horizon Explorer

2. Εφαρμογές πλοήγησης



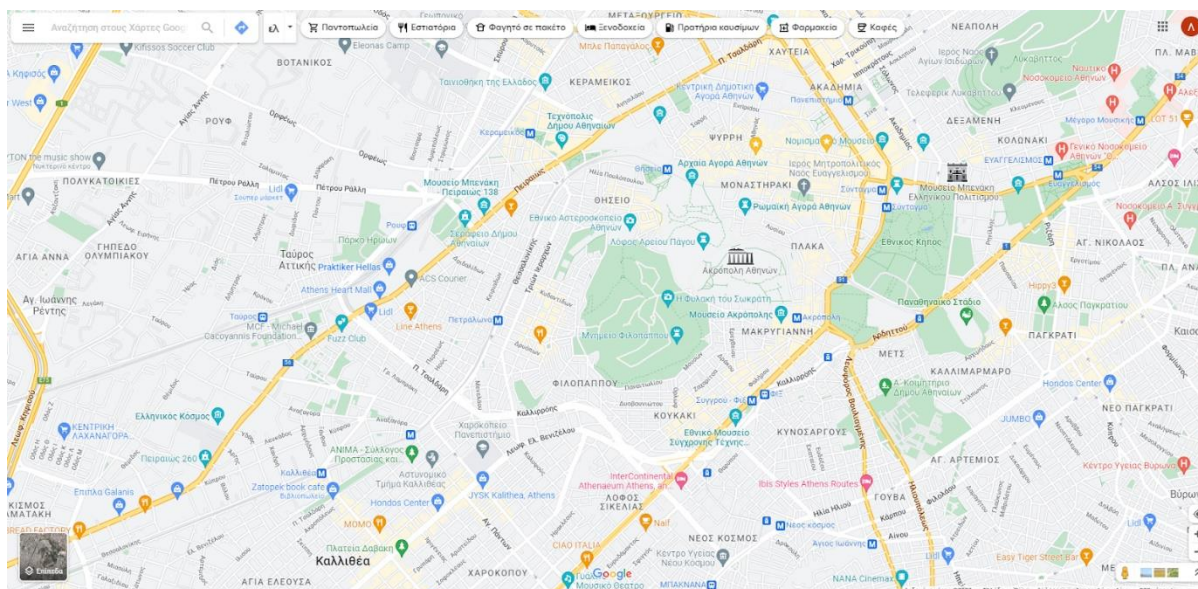
Εικόνα 5: Leading mapping apps in the United States in 2021, by downloads

Πέρα από τις λειτουργίες ενισχυμένης πραγματικότητας, το άλλο σημαντικό κομμάτι που χρειάστηκε να γίνει ανάλυση ήταν η πλοήγηση του χρήστη. Σε αυτόν τον τομέα υπάρχουν πολλές εφαρμογές. Μερικές από τις εμπορικά, πιο επιτυχημένες εφαρμογές αποτελούν οι Google Maps, Apple Maps, Here We Go και Waze.

Τα Google Maps, είναι η πιο γνωστή και πολυχρησιμοποιημένη εφαρμογή πλοήγησης στον κόσμο. Το 2021, σύμφωνα με το Statista, το Google Maps ήταν η πιο επιτυχημένη εφαρμογή για πλοήγηση ενός χρήστη, καθώς είχε τα περισσότερα download στις Ηνωμένες Πολιτείες της Αμερικής με δεύτερη εφαρμογή την Waze. Το ενδιαφέρον σε αυτή την μέτρηση είναι ότι δεν έχει σημασία που το Google Maps είναι προεγκατεστημένο στα κινητά Android. (Ceci, statista, 2022)

Η εφαρμογή είναι διαθέσιμη σε συσκευές Android μέσω του playstore, IOS μέσω του appstore και Web. Η εφαρμογή, αποτελεί το στάνταρ στο χώρο της πλοήγησης και οι περισσότερες άλλες εφαρμογές πλοήγησης έχουν βασίσει τον τρόπο διάδρασης του χρήστη με την εφαρμογή τους, με βάση το πώς είναι υλοποιημένες οι αντίστοιχες λειτουργίες στο Google Maps. Οποιαδήποτε εφαρμογή περιέχει κάποιο είδος χάρτη κοιτάει και συγκρίνει την υλοποίησή της με αυτή των Google Maps. Οπότε για να γίνει κατανοητό το πως λειτουργεί αυτή η εφαρμογή πρέπει να υλοποιηθεί μια πλήρη επισκόπηση των δυνατοτήτων της.

Ξεκινώντας λοιπόν, όταν την ανοίξεις την εφαρμογή ,σου δείχνει τον χάρτη της περιοχής σου με όλους τους δρόμους και τα ονόματά τους, που αυτή περιέχει. Ακόμη, είναι καρφίτσωμένα διάφορα σημεία ενδιαφέροντος συνοδευόμενα από το όνομα του σημείου. Τα σημεία αυτά ποικίλουν, μπορεί να είναι απλά εμπορικά καταστήματα, καταστήματα εστίασης, δημόσιες υπηρεσίες, πανεπιστήμια, σχολεία, σταθμοί μέσων μαζικής μεταφοράς (π.χ. Λεωφορεία, τρένα , μετρό), κοινόχρηστοι δημόσιοι χώροι (όπως πλατείες, πάρκα), άλση, λόφοι, ποτάμια, λιμάνια, αεροδρόμια και άλλα.



Εικόνα 6: Google Maps



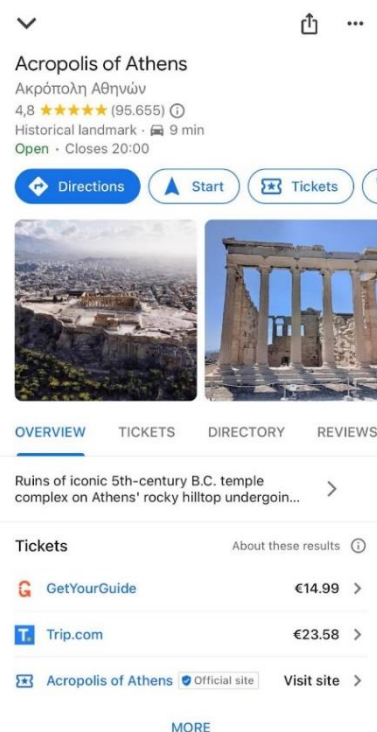
Εικόνα 7: Google Maps 1

Συμπεριλαμβανομένων των σημείων σε αυτή τη λίστα αποτελούν και οι αρχαιολογικοί χώροι, τα μουσεία και τα αξιοθέατα τα οποία μερικές φορές έχουν και μια μοναδική αναγνωριστική “καρφίτσα” για να ξεχωρίζουν από όλα τα υπόλοιπα σημεία. Η κάθε καρφίτσα στο κέντρο της έχει ένα εικονίδιο το οποίο υποδηλώνει την κατηγορία του σημείου, δηλαδή για παράδειγμα αν πρόκειται για εστιατόριο περιέχει ένα εικονίδιο με μαχαιροπήρουνα, αν είναι σταθμός μετρό έχει ένα κεφαλαίο γράμμα Μ, αν είναι κάποιος αρχαιολογικός χώρος περιέχει ένα πύργο και πολλά άλλα. Το χρώμα της καρφίτσας και του ονόματος του σημείου είναι επίσης διαφορετικό για κάθε σημείο υποδηλώνοντας την κατηγορία στην οποία ανήκει το σημείο. Όταν πατήσεις πάνω σε ένα σημείο εμφανίζεται ένα μενού με στοιχεία σχετικά με το μέρος ενδιαφέροντος. Πιο συγκεκριμένα, δείχνει στο χρήστη μια καρτέλα με μερικές φωτογραφίες του σημείου, αν είναι βέβαια διαθέσιμες, από κάτω αναγράφει το όνομα του σημείου και πιο κάτω με μικρότερη γραμματοσειρά το είδος-κατηγορία του σημείου. Ανάμεσα σε αυτά βρίσκεται και η βαθμολογία του, που βασίζεται με

άριστα τα 5 αστέρια και στην καρτέλα φαίνεται ως ο μέσος όρος όλων των βαθμολογιών.

Μετά, δεξιά από τον μέσο όρο, μέσα σε παρένθεση αναγράφονται και ο αριθμός των κριτικών που έχουν πραγματοποιηθεί για να βγει αυτός ο μέσος όρος. Τέλος, στα δεξιά του ονόματος υπάρχουν 2 κουμπιά. Το ένα είναι για να δείξει οδηγίες στο χρήστη για τη διαδρομή που πρέπει να ακολουθήσει για να πάει στο σημείο αυτό και το άλλο κουμπί είναι για να βάλει “σελιδοδείκτη” στο σημείο και να μπορέσει να το ξαναβρεί αργότερα από την προσωπική του σελίδα με όλους τους σελιδοδείκτες του.

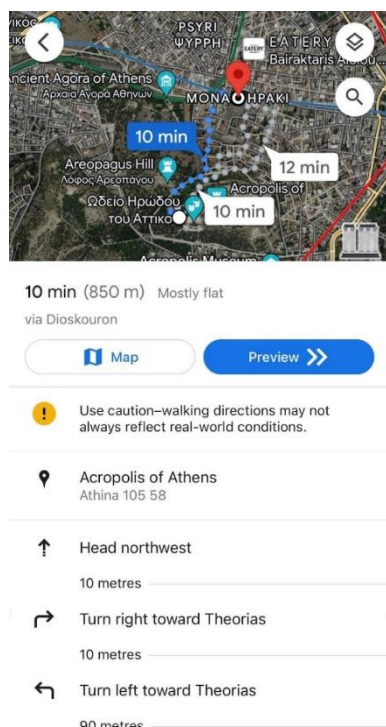
Στο κάτω δεξιά μέρος της οθόνης υπάρχουν τρία κουμπιά. Το ένα κεντράρει στη γεωγραφική θέση του χρήστη. Τα άλλα δύο είναι για να μπορεί να ελέγχει ο χρήστης το ζουμ της περιοχής και να μπορεί αντίστοιχα να την διευρύνει ή να την συρρικνώνει. Το Google Maps προσφέρει ένα παγκόσμιο χάρτη του πλανήτη οπότε δεν υπάρχει κάποιο όριο για τους τόπους στους οποίους μπορεί να περιηγηθεί ο χρήστης. “Σέρνοντας” το κέρσορα ή το δάχτυλό του στο χάρτη, ο χρήστης μπορεί να μετακινήσει το χάρτη που εμφανίζεται στην οθόνη του.



Εικόνα 8: Google Maps 2

Στο πάνω αριστερά μέρος της οθόνης, υπάρχει ένα πεδίο κειμένου για να μπορεί ο χρήστης να αναζητήσει σημεία και διευθύνσεις πάνω στο χάρτη. Για να πραγματοποιηθεί η αναζήτηση, πρέπει ο χρήστης να πατήσει πάνω στο πεδίο αναζήτησης και να αρχίσει να πληκτρολογεί το όνομα του σημείου ή την διεύθυνση την οποία επιθυμεί να εντοπίσει. Καθώς πληκτρολογεί, κάτω από το πεδίο θα εμφανιστούν σχετικές προτάσεις σημείων και διευθύνσεων με βάση το κείμενο που έχει πληκτρολογήσει ήδη στο πεδίο αναζήτησης.

Μόλις ο χρήστης διαλέξει ένα σημείο, τότε ο χάρτης κεντράρει στο σημείο αυτό και ανοίγει ένα πάνελ στην οθόνη που πάνω δείχνει μια φωτογραφία του σημείου, αν είναι διαθέσιμη, από κάτω πάλι το όνομα του σημείου ακολουθούμενο από τη βαθμολογία και τον αριθμό των αξιολογήσεων και από κάτω το είδος-κατηγορία του. Κάτω από όλα αυτά, υπάρχουν πέντε κουμπιά. Το πρώτο είναι για να δώσει οδηγίες για το πώς να κατευθυνθεί προς το σημείο αυτό, ο χρήστης. Το δεύτερο, ανοίγει τον ιστότοπο του σημείου αν είναι διαθέσιμος. Το επόμενο, αποθηκεύει το σημείο στους σελιδοδείκτες του χρήστη. Τα δύο τελευταία, βρίσκουν άλλα κοντινά σημεία με το σημείο αυτό, που η εφαρμογή προτείνει να επισκεφτεί ο χρήστης και η τελευταία κοινοποιεί το σημείο σε κάποιο μέσω κοινωνικής δικτύωσης. Κάτω από αυτά, προαιρετικά φαίνεται μια σύντομη περιγραφή του σημείου μαζί με στοιχεία όπως την πλήρη διεύθυνση, το τηλέφωνο, το email και άλλα. Για χώρους εστίασης, μαγαζιά και υπηρεσίες παρέχονται συχνά και τα ωράρια λειτουργίας.



Πατώντας ένα από τα κουμπιά πλοήγησης που αναφέρθηκαν παραπάνω, τότε ανοίγει ένα πάνελ πλοήγησης το οποίο πάνω έχει δύο πεδία με τη διεύθυνση από όπου ξεκινάει ο χρήστης και ένα με τη διεύθυνση του τελικού του προορισμού. Ακριβώς πάνω από αυτά, υπάρχουν κουμπιά για να επιλέξει με τι μέσο θα μετακινηθεί ο χρήστης. Για παράδειγμα, μπορεί να επιλέξει να μετακινηθεί με αυτοκίνητο, με τα πόδια, με ποδήλατο, με μέσα μαζικής μεταφοράς και άλλα. Μόλις έχουν συμπληρωθεί τα παραπάνω, τότε η εφαρμογή προτείνει όλες τις πιθανές διαδρομές που μπορεί να ακολουθήσει ο χρήστης για να φτάσει στον προορισμό του, ταξινομημένα με βάση τις πιο γρήγορες διαδρομές. Μόλις επιλεγθεί μία διαδρομή, τότε αυτή εμφανίζεται πάνω στο χάρτη ως ένα σύνολο μπλε γραμμών που περνούν πάνω από τους

Εικόνα 9: Google Maps 3

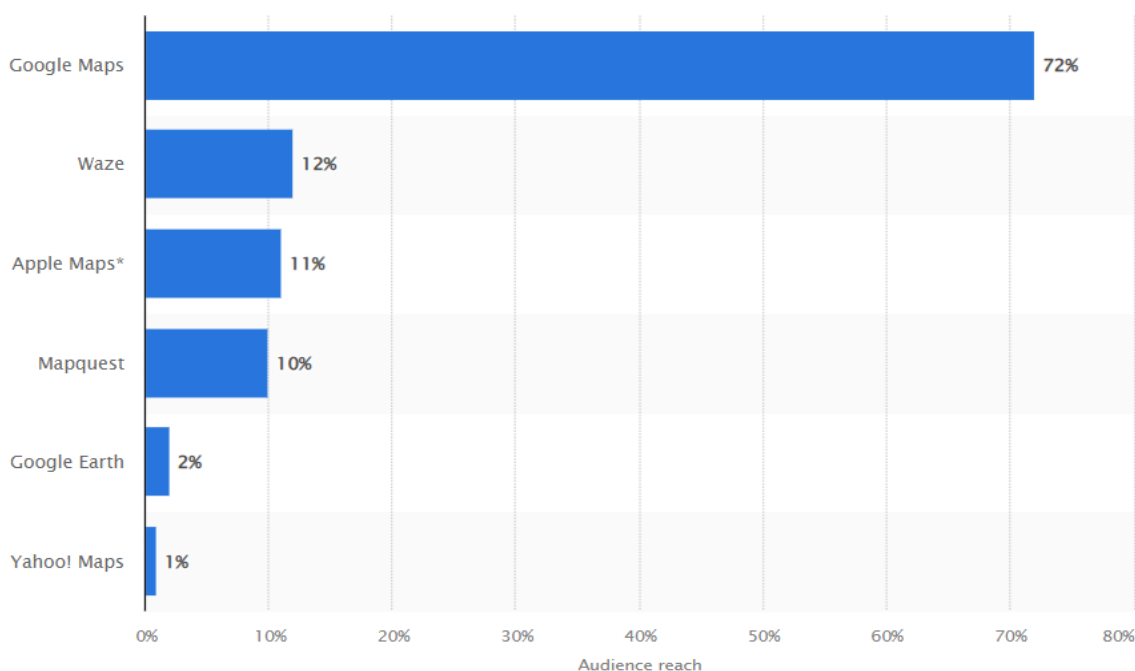
δρόμους της επιλεγμένης διαδρομής. Οι εναλλακτικές διαδρομές επίσης, εμφανίζονται πάνω στο χάρτη ως γκρι γραμμές.



Better ways ahead.

Εικόνα 10: Apple Maps

Σύμφωνα με μια έρευνα που έχει γίνει στις αρχές του 2021, οι χρήστες iPhone ανέρχονται στο 1 δισεκατομμύριο, με ποσοστό αύξησης 5,48% από το 2020. Η Apple όταν λάνσαρε την πρώτη γενιά iPhone χρησιμοποιούσε σαν καθιερωμένη εφαρμογή πλοήγησης το Google Maps. Η δημιουργία του Apple Maps ήταν αποτέλεσμα μιας διαφωνίας ανάμεσα στους δύο κολοσσούς το 2009, αφού μετά από αναβάθμιση της εφαρμογής από την Google, η οποία εισήγαγε καινούργια χαρακτηριστικά, οι χρήστες iOS δεν είχαν πρόσβαση σε αυτά. Έτσι, το 2012, η Apple λάνσαρε την καινούργια προεγκαταστημένη εφαρμογή στα iPhone, το Maps. (Wikipedia, 2022)



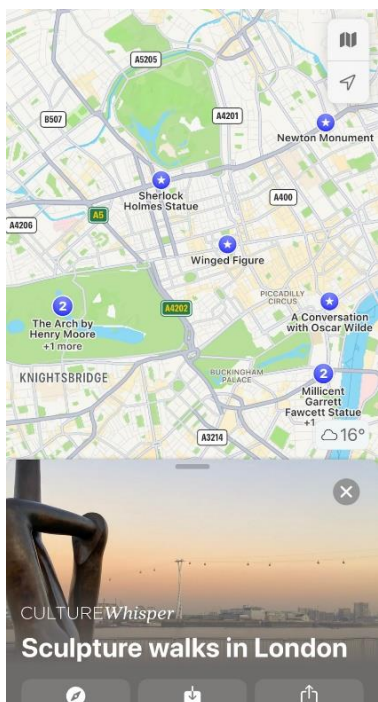
Εικόνα 11: Most popular mapping apps in the United States as of April 2018, by reach

Η εφαρμογή χρησιμοποιείται αποκλειστικά από χρήστες συσκευών της Apple και σύμφωνα με έρευνα του 2018, είναι η τρίτη δημοφιλέστερη εφαρμογή για χάρτες στην Αμερική. (Ceci, statista, 2022)

Όπως έχει αναφερθεί και πιο πάνω, οι περισσότερες εφαρμογές πλοήγησης έχουν στηρίξει την ανάπτυξη τους στην εφαρμογή Google Maps. Όταν μπει κάποιος στην εφαρμογή, το πρώτο πράγμα που θα δει είναι ένας χάρτης με σημειωμένο το μέρος που βρίσκεται η συσκευή αυτή τη στιγμή, και ένα πανοραμικό χάρτη κλίμακας 375μ, στον οποίο εμφανίζονται οι μεγαλύτεροι δρόμοι και τα ονόματά τους. Κάνοντας μεγέθυνση της εικόνας εμφανίζονται πιο πολλά ονόματα δρόμων. Η εφαρμογή μπορεί να εστιάσει από 1000χμ έως 12μ. Υπάρχουν τέσσερις επιλογές χάρτη, για οδήγηση, για εξερεύνηση, για τα μέσα μαζικής μεταφοράς και ένας χάρτης με την ακριβή εμφάνιση μέσω δορυφόρου. Η επιλογή χάρτη οδήγησης εμφανίζει την κίνηση των δρόμων, τους κλειστούς δρόμους και τα πιθανά ατυχήματα που μπορεί να έχουν συμβεί. Ο χάρτης με τα μέσα μαζικής μεταφοράς δείχνει τις διαδρομές του Ηλεκτρικού, του μετρό, του τραμ, των λεωφορείων και των τρόλεϊ. Στην Ελλάδα, αυτό το χαρακτηριστικό δεν είναι διαθέσιμο ακόμα. Στον τρίτο χάρτη, μπορεί κάποιος να δει τι υπάρχει γύρω του(καταστήματα, μουσεία, πλατείες, κ.λπ.). Τέλος, ο χάρτης από τον δορυφόρο απεικονίζει μια ακριβή εικόνα της γύρω περιοχής.



Εικόνα 12: Maps 1

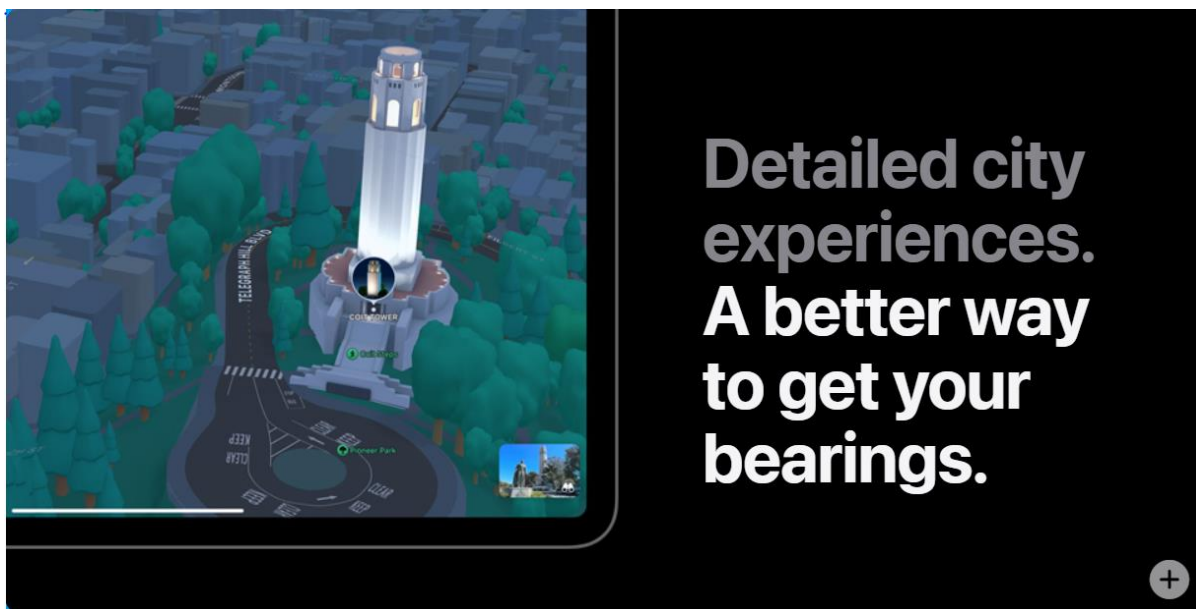


Εικόνα 13: Maps 2

Σε όλους τους χάρτες υπάρχουν καρφισωμένα σημεία, τα οποία χωρίζονται σε κατηγορίες με διαφορετικά χρώματα, συνοδευόμενα πάντα από το όνομα του χώρου. Οι πινέζες μπορεί να είναι εστιατόρια ή καφετέριες(πορτοκαλί), μουσεία ή αρχαιολογικοί χώροι(μπλε), μαγαζιά(κίτρινο), κοινόχρηστοι δημόσιοι χώροι(πράσινο), ξενοδοχεία(μοβ), χώροι τέχνης(ροζ) κ.λπ.

Στο κάτω μέρος της εφαρμογής, μπορεί να ψάξει κάποιος μια τοποθεσία. Πατώντας στο πλαίσιο αναζήτησης, το Maps εμφανίζει κάποιες κατηγορίες που μπορεί ο χρήστης να τις βρει χρήσιμες, όπως είναι τα κοντινότερα φαρμακεία ή βενζινάδικα ή σουπερμάρκετ, καθώς και κάποιους οδηγούς, οι οποίοι έχουν δημιουργηθεί από ξένα περιοδικά, μπλογκ ή ιστοσελίδες. Για παράδειγμα στη διπλανή εικόνα, η ιστοσελίδα CULTUREWhisper έχει δημιουργήσει έναν οδηγό με μέρη στα οποία μπορεί ο χρήστης να δει γλυπτά στο

Λονδίνο. Αυτή η λειτουργία δεν είναι ακόμα διαθέσιμη στην Ελλάδα. Όπως και το παρακάτω καινούργιο χαρακτηριστικό, το οποίο είναι διαθέσιμο σε μόνο 4 πόλεις: Λονδίνο, Σαν Φρανσίσκο, Νέα Υόρκη και Λος Άντζελες. Αυτό το καινούργιο χαρακτηριστικό επιτρέπει στο χρήστη να περιηγηθεί σε έναν χάρτη, με μια άνευ προηγουμένου λεπτομέρεια για το υψόμετρο, τις γειτονιές, τα κτίρια, τα δέντρα και χαρακτηριστικά των δρόμων όπως είναι οι διαβάσεις πεζών. Επίσης ο χρήστης μπορεί “να επισκεφθεί” διάσημα μνημεία τα οποία έχουν αναπαρασταθεί σε 3D μοντέλα και υπάρχει η δυνατότητα να εμφανίζονται με δύο διαφορετικούς τρόπους, είτε με πρωινό είτε βραδινό φόντο. (Apple, 2022)



Εικόνα 14: CULTUREWhisper



Εικόνα 15: Maps 3

Ακόμα ο χρήστης μπορεί να αποθηκεύσει τις αγαπημένες του τοποθεσίες ώστε να έχει εύκολη πρόσβαση και στο μέλλον αλλά και να φτιάξει δικούς του χάρτες- διαδρομές που να έχει μέρη που αυτός θέλει να εμφανίζονται όλα μαζί σε έναν χάρτη. Επίσης, ο χρήστης μπορεί να έχει πρόσβαση σε όλες τις πρόσφατες αναζητήσεις που έχει κάνει.

Επιλέγοντας ένα καρφισωμένο σημείο, πρώτα εμφανίζεται το όνομα του σημείου, σε ποια κατηγορία ανήκει, ακολουθούν δύο “κουμπιά”, το ένα δίνει οδηγίες για το πως ο χρήστης θα κατευθυνθεί σε αυτό το σημείο και το δεύτερο, αν το σημείο που έχει επιλεγεί είναι μουσείο ή κάποιος χώρος στον οποίο για να περιηγηθείς πρέπει να έχεις εισιτήριο, η εφαρμογή έχει το κουμπί “εισιτήριο” ώστε να μπορεί κάποιος να κλείσει εύκολα εισιτήριο μέσω της εφαρμογής. Από κάτω φαίνεται αν ο χώρος που θέλει να επισκεφτεί ο χρήστης είναι ανοιχτός ή κλειστός,

τη βαθμολογία του στο TripAdvisor και την απόσταση μεταξύ των δύο σημείων. Στη συνέχεια ακολουθούν οι φωτογραφίες από το σημείο και τέσσερα κουμπιά τα οποία είναι το τηλέφωνο επικοινωνίας, η ιστοσελίδα του, ένα κουμπί που σου επιτρέπει να σώσεις το σημείο, ώστε να μπορείς να το ξαναβρείς εύκολα και τέλος ένα κουμπί που σου επιτρέπει να μοιραστείς την τοποθεσία σου μέσω κάποιων μέσων κοινωνικής δικτύωσης. Τέλος, έχει ένα μικρό κείμενο με πληροφορίες για το σημείο, που έχουν αντληθεί από την Βικιπαίδεια. Όλα όσα περιεγράφηκαν παραπάνω γίνονται εμφανή και από την εικόνα 2.

Πατώντας το κουμπί της πλοήγησης, τότε ανοίγει ένα καινούργιο πάνελ στο οποίο ο χρήστης καλείται να επιλέξει από που θα ξεκινήσει η διαδρομή και ποια ώρα θέλει να φύγει από το σημείο που βρίσκεται, ώστε να δώσει μια εκτίμηση για το πόση ώρα θα πάρει να φτάσει στον προορισμό. Ακολουθούν 5 επιλογές για το πως να φθάσει ο χρήστης στον προορισμό του. Η πρώτη είναι το αυτοκίνητο, στο οποίο δίνει την χιλιομετρική απόσταση, πόση ώρα χρειάζεται κάποιος για να φθάσει στον προορισμό του, καθώς και επιλογές για τον ποιο δρόμο να ακολουθήσει(π.χ. τον πιο σύντομο) και την επιλογή να μην συμπεριλάβει διαδρομές που να έχουν διόδια ή αυτοκινητόδρομους. Η δεύτερη επιλογή είναι να πάει κάποιος με τα πόδια στον προορισμό του. Σε αυτή την περίπτωση το Maps δίνει 2-3 επιλογές για το πως να φτάσει κανείς γρηγορότερα στον προορισμό του. Οι επόμενες δύο επιλογές είναι η μετακίνηση με τα μέσα μαζικής μεταφοράς ή με ποδήλατο αλλά αυτές οι δύο κατηγορίες δεν είναι διαθέσιμες στην Ελλάδα. Η τελευταία επιλογή είναι να καλέσεις ταξί, στο οποίο συνδέει το χρήστη με τις δύο εφαρμογές(Uber, Beat) που χρησιμοποιούνται στην Ελλάδα για ταξί.



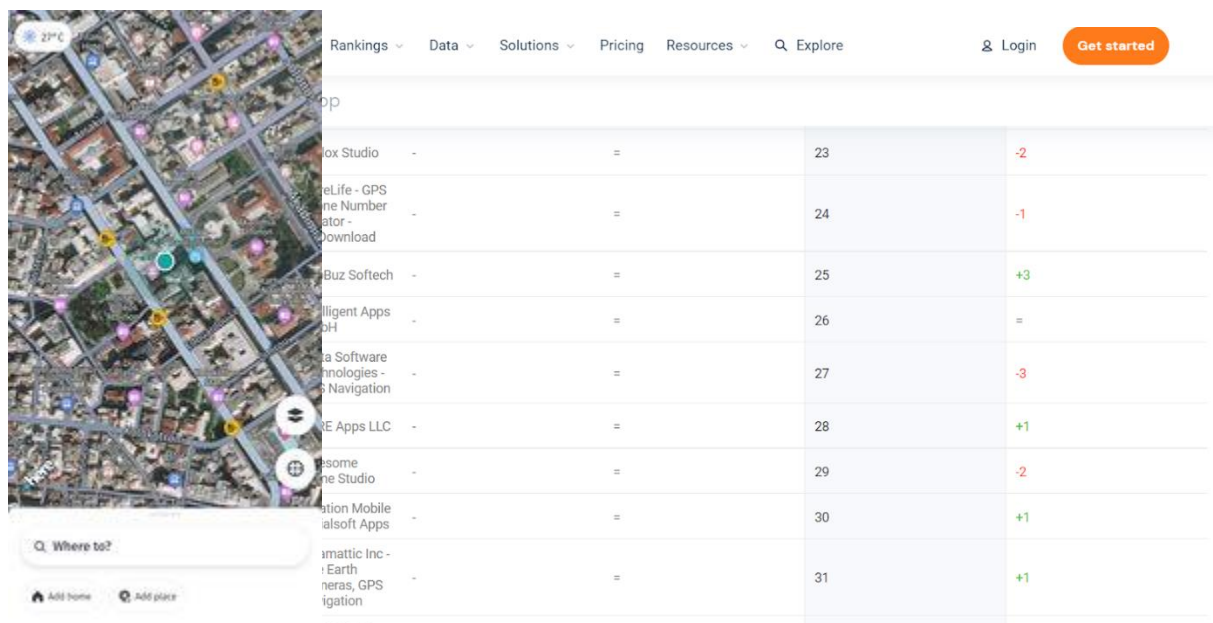
Εικόνα 16: Maps 4

Το Here WeGo είναι η επόμενη εφαρμογή πλοήγησης για την οποία θα γίνει ανάλυση. Η εφαρμογή πριν φτάσει στην τελική της ονομασία πέρασε από διάφορα στάδια. Η εκκίνηση έγινε το 2006, όταν η Nokia εξαγόρασε την Gate5, η οποία είχε αναπτύξει το λογισμικό smart2go, το οποίο χρησιμοποιήθηκε σε κινητές συσκευές με Windows 5.0. Το 2007 η εφαρμογή μετονομάστηκε σε Oni Maps μέχρι το 2011 που μετονομάστηκε σε Nokia Maps, προσφέροντας δωρεάν πλοήγηση και πληροφορίες για την κίνηση. Το 2012, το Nokia Maps μετονομάστηκε σε Here Maps και επιτέλους μπορούσαν να το χρησιμοποιήσουν χρήστες συσκευών iOS και Windows. Τρία χρόνια μετά έγινε και η επέκταση σε συσκευές Android. Το 2016 γίνεται ξανά μετονομασία της εφαρμογής σε Here WeGo, η ονομασία που ισχύει μέχρι σήμερα.



Εικόνα 17: HEREWeGo

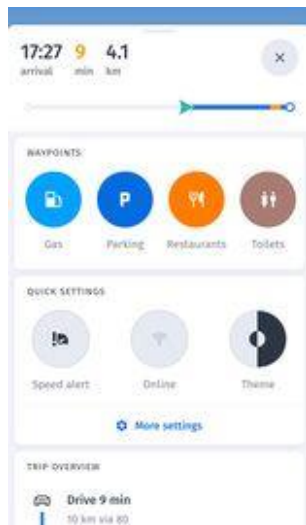
Σύμφωνα με το similarweb, η εφαρμογή είναι 28η σε δημοφιλή στην κατηγορία Πλοήγησης-Χαρτών στην Ελλάδα. (similarweb, 2022)



Rank	App Name	Change	Score	Delta
23	Box Studio	=	23	-2
24	eLife - GPS Phone Number Locator - Download	=	24	-1
25	Buz Softech	=	25	+3
26	Intelligent Apps 24h	=	26	=
27	Mapa Software Technologies - Navigation	=	27	-3
28	Mapa Apps LLC	=	28	+1
29	Mapa Studio	=	29	-2
30	Navigation Mobile alsoft Apps	=	30	+1
31	Mapmatic Inc - Earth, GPS, Navigation	=	31	+1

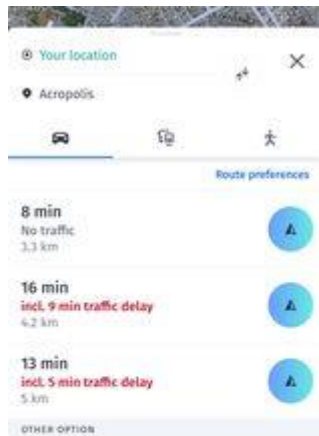
Εικόνα 18: Top Apps Ranking

Όταν μπει κάποιος στην εφαρμογή, το πρώτο πράγμα που θα δει είναι ένας χάρτης με σημειωμένο το μέρος που βρίσκεται η συσκευή αυτή τη στιγμή με μια πράσινη κουκίδα. Στο χάρτη εμφανίζονται οι μεγαλύτεροι δρόμοι και τα ονόματά τους. Κάνοντας μεγέθυνση της εικόνας εμφανίζονται ακόμα περισσότεροι δρόμοι με τα ονόματά τους. Υπάρχουν δυο επιλογές χάρτη, ο προκαθορισμένος και ένας χάρτης με την ακριβή εμφάνιση μέσω δορυφόρου. Και στους δύο χάρτες δίνεται η επιλογή εμφάνισης της κίνησης στους δρόμους. Αυτή η επιλογή, εμφανίζει την κίνηση των δρόμων, τους κλειστούς δρόμους



Εικόνα 20: HereWeGo 1

σημείο ανοίγει μια δεύτερη σελίδα στην οποία, βλέπουμε καταρχάς το σημείο να είναι καρφισωμένο με μία πράσινη πινέζα. Στη συνέχεια εμφανίζεται το όνομα της τοποθεσίας, η διεύθυνση και ένα σύστημα βαθμολόγησης(rating system), το οποίο το έχουν αντλήσει από τον ιστότοπο του TripAdvisor. Ακολουθούν τρία κουμπιά, το πρώτο σε ανακατευθύνει σε καινούργια σελίδα που δίνει οδηγίες για τη διαδρομή, το δεύτερο επιτρέπει στο χρήστη να μοιραστεί την τοποθεσία του με κάποιον άλλο και το τελευταίο αποθηκεύει την τοποθεσία στα “Αγαπημένα” ώστε ο χρήστης να μπορεί να το ξαναβρεί εύκολα. Λίγο πιο κάτω εμφανίζεται μια εικόνα του μέρους και κάποιες πληροφορίες επικοινωνίας.



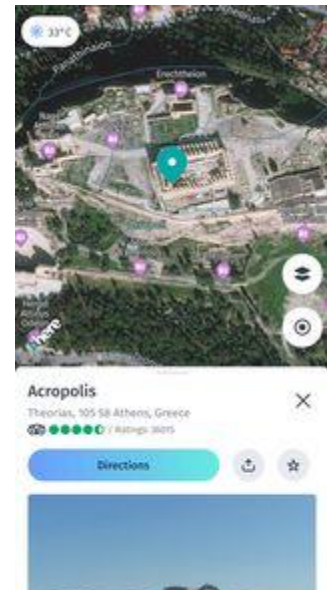
Εικόνα 21: HereWeGo 3

Εικόνα 22: HereWeGo 4

και τα πιθανά ατυχήματα που μπορεί να έχουν συμβεί. Στον προκαθορισμένο χάρτη μπορεί κάποιος να δει τι υπάρχει γύρω του(καταστήματα, μουσεία, πλατείες ,κ.λπ.). Τέλος, ο χάρτης από το δορυφόρο απεικονίζει μια ακριβή εικόνα της γύρω περιοχής.

Σε όλους τους χάρτες υπάρχουν καρφισωμένα σημεία, τα οποία χωρίζονται σε κατηγορίες με διαφορετικά χρώματα, συνοδευόμενα πάντα από το όνομα του χώρου. Οι πινέζες μπορεί να είναι εστιατόρια ή καφετέριες(πορτοκαλί), μουσεία ή αρχαιολογικοί χώροι(μοβ), μαγαζιά(φούξια), κοινόχρηστοι δημόσιοι χώροι(πράσινο), ξενοδοχεία(μπλε σκούρο), χώροι τέχνης(ροζ), μέσα μαζικής μεταφοράς/βενζινάδικα(γαλάζιο) κ.λπ.

Στο κάτω μέρος της οθόνης εμφανίζεται μια μηχανή αναζήτησης στην οποία μπορεί κάποιος να βάλει την τοποθεσία στην οποία θέλει να πάει. Μόλις βρει το



Εικόνα 19: HereWeGo 2

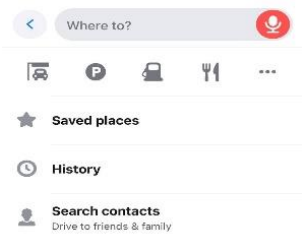
Πατώντας το κουμπί της πλοήγησης, ο χρήστης ανακατευθύνεται σε μια καινούργια σελίδα στην οποία του δίνονται κάποιες επιλογές. Πρώτα πρέπει να επιλέξει το σημείο εκκίνησης της διαδρομής και το σημείο τερματισμού. Με βάση αυτό, εμφανίζονται στο χάρτη κάποιες διαδρομές με προεπιλεγμένη διαδρομή την πιο γρήγορη. Ο χρήστης μπορεί ακόμα να επιλέξει αν θα μετακινηθεί με αυτοκίνητο, με τα μέσα μαζικής μεταφοράς ή με τα πόδια. Ακόμα, η εφαρμογή δίνει την δυνατότητα να επιλέξει μεταξύ της πιο γρήγορης διαδρομής και της πιο μικρής(χιλιομετρικά) διαδρομής. Ο χρήστης μπορεί να βάλει και κάποιους περιορισμούς για τη

διαδρομή που θέλει να κάνει, μπορεί να επιλέξει να αποφύγει διόδους, τούνελ, αυτοκινητόδρομους, μη ασφαλτοστρωμένους δρόμους και πλοία.

Επιλέγοντας μια διαδρομή, εμφανίζεται το σημείο εκκίνησης, αν η διαδρομή είναι με αμάξι, τότε κάτω αριστερά εμφανίζεται το όριο ταχύτητας του δρόμου και η ταχύτητα του οχήματος. Λίγο πιο κάτω, η εφαρμογή δίνει κατά προσέγγιση μια ώρα άφιξης στον προορισμό, τη μέση ώρα διαδρομής και τη χιλιομετρική απόσταση. Οδηγίες δίνονται και με φωνητική πλοήγηση και με οδηγίες στην οθόνη της συσκευής. Αν κατεβάσουμε λίγο ακόμα την σελίδα, τότε ο χρήστης μπορεί να βρει χρήσιμα μέρη τα οποία ίσως να χρειάζεται, (π.χ. βενζινάδικο, παρκινγκ, εστιατόριο, τουαλέτες) αλλά και να απενεργοποιήσει το κουμπί “Ειδοποίηση Ταχύτητας” και να αλλάξει φόντο από άσπρο σε μαύρο.

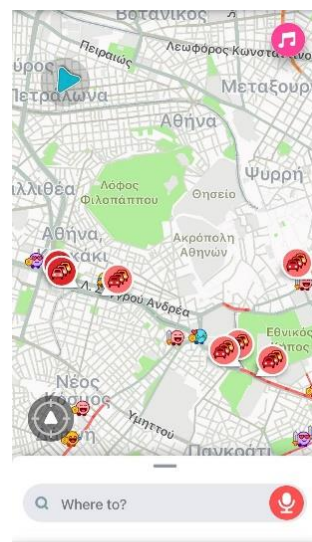
Η τελευταία εφαρμογή πλοήγησης για την οποία θα γίνει ανάλυση είναι η Waze. Όπως αναφέρθηκε και πιο πάνω, είναι η δεύτερη εφαρμογή με τα περισσότερα downloads στις Ηνωμένες Πολιτείες της Αμερικής για το 2021. Η FreeMap Israel είναι μια εταιρία που αρχικά δραστηριοποιήθηκε στο Ισραήλ. Η εταιρία ανέπτυξε ένα λογισμικό που παρέχει πλοήγηση μέσω δορυφόρου. Το 2009, μετονομάστηκε σε Waze Mobile Ltd. Το 2013 εξαγοράστηκε από την Google και σήμερα είναι θυγατρική της.

Με την είσοδο στην εφαρμογή, ο χρήστης το πρώτο πράγμα που βλέπει είναι ένας χάρτης με σημειωμένο το μέρος που βρίσκεται η συσκευή αυτή τη στιγμή με ένα μπλε βελάκι. Στο χάρτη εμφανίζονται οι μεγαλύτεροι δρόμοι και τα ονόματά τους καθώς αν γίνει σμίκρυνση εμφανίζονται τα ονόματα των



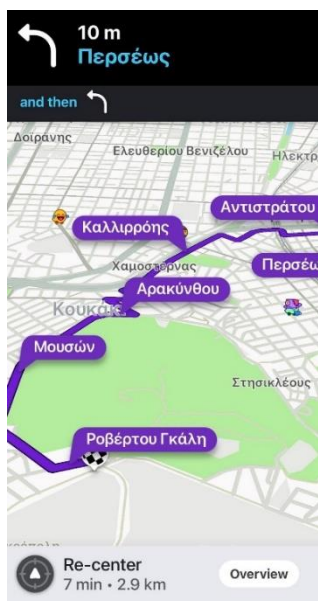
περιοχών. Υπάρχει μια επιλογή χάρτη. Η εφαρμογή δίνει μια λεπτομερή απεικόνιση της κίνησης της περιοχής, των ατυχημάτων, και άλλα. Σε αντίθεση με τις παραπάνω εφαρμογές το Waze δεν έχει καρφίτσωμένα σημεία (π.χ. εστιατόρια, μουσεία, μαγαζιά) στο χάρτη.

Στο κάτω μέρος της εφαρμογής, μπορεί να ψάξει ο χρήστης την τοποθεσία στην οποία θέλει να πάει. Πατώντας το πλαίσιο



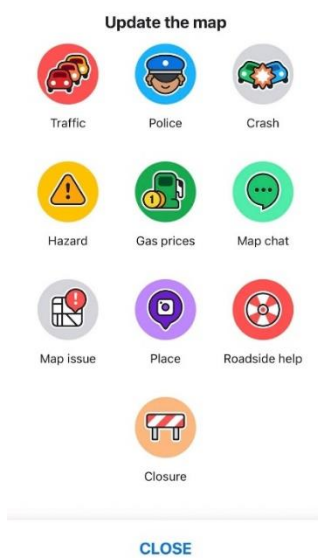
Εικόνα 23: Waze 1

Εικόνα 24: Waze 2



Εικόνα 25: Waze 3

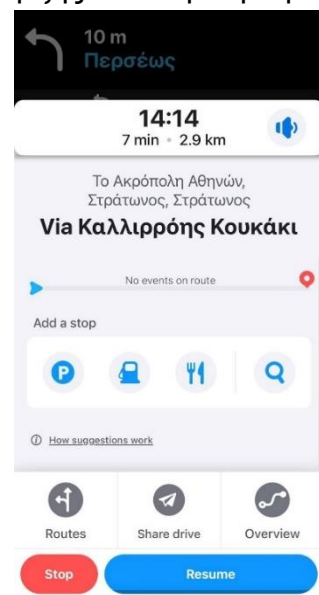
την χιλιομετρική απόσταση, εμφανίζει αν υπάρχουν πιθανές καθυστερήσεις στο δρόμο, ενώ προτείνει και τα κοντινότερα πρατήρια βενζίνης, πάρκινγκ και εστιατόρια. Ακριβώς από κάτω, έχει τρία κουμπιά. Χρησιμοποιώντας το πρώτο ο χρήστης μπορεί να αλλάξει το δρόμο για να φτάσει στο σημείο που έχει επιλέξει, το δεύτερο δίνει την επιλογή κοινοποίησης της διαδρομής σε κάποιο άλλο πρόσωπο και το τρίτο δίνει μια επισκόπηση της διαδρομής. Κλείνοντας το πάνελ, στα αριστερά υπάρχει ένα κουμπί το οποίο μετράει την ταχύτητα



Εικόνα 27: Waze 5

αναζήτησης, το Waze εμφανίζει κάποιες κατηγορίες που μπορεί ο χρήστης να τις βρει χρήσιμες, όπως είναι τα πρατήρια βενζίνης, drive-thru, παρκινγκ, σουπερμάρκετ, φαρμακεία, κ.α. Ακριβώς από κάτω εμφανίζονται τα μέρη που έχει σώσει ο χρήστης, τις τελευταίες του διαδρομές, και επιτρέπει να γίνει σύνδεση της εφαρμογής με τις επαφές του κινητού, ώστε να αντλήσει πληροφορίες για τις διαδρομές προς σπίτια φίλων και γνωστών. Ακόμα ο χρήστης μπορεί να αποθηκεύσει τις αγαπημένες του τοποθεσίες ώστε να έχει εύκολη πρόσβαση και στο μέλλον.

Επιλέγοντας τη διεύθυνση, για την οποία έγινε η αναζήτηση, η εφαρμογή δίνει στο χρήστη τρεις διαφορετικές διαδρομές για να επιλέξει. Επιλέγοντας τη διαδρομή, στο πάνω μέρος της οθόνης δίνει τις πρώτες δυο οδηγίες για την διαδρομή που ακολουθεί ο χρήστης. Κάνοντας κλικ στο κάτω μέρος της οθόνης, ανοίγει ένα καινούργιο πάνελ το οποίο δίνει την ώρα άφιξης στον προορισμό,



Εικόνα 26: Waze 4

του αυτοκινήτου και στα δεξιά υπάρχει ένα κουμπί στο οποίο ο χρήστης μπορεί να δηλώσει περιστατικά που συμβαίνουν στο δρόμο. Ξεχωριστό χαρακτηριστικό της εφαρμογής είναι ότι στο δρόμο εμφανίζονται και άλλοι χρήστες του Waze, οι οποίοι λέγονται

Wazers, για τους οποίους ο χρήστης μπορεί να μάθει πότε γράφτηκαν στην εφαρμογή και με πόσα χιλιόμετρα κινούνται την ώρα. Με κάποιους από τους άλλους χρήστες μπορεί να γίνει και αλληλεπίδραση μέσω της εφαρμογής.

Ορισμοί

Android

1. Gradle

Το Gradle είναι ένα εργαλείο αυτοματοποίησης το οποίο χρησιμοποιείται, για την αυτοματοποίηση των διαδικασιών του compile, ενός προγράμματος, καθώς και για να πακετάρει και να προετοιμάσει το πρόγραμμα για deployment και publishing. Το εργαλείο δεν χρησιμοποιείται αποκλειστικά για εφαρμογές Android, καθώς μπορεί να χρησιμοποιηθεί γενικά για την ανάπτυξη λογισμικού Java. Ωστόσο, έρχεται ως το προεπιλεγμένο σύστημα αυτοματοποίησης, αυτών των λειτουργιών όταν δημιουργείται ένα νέο Android Project.

2. View

Το View είναι ο δομικός λίθος των User Interface(U.I.) αντικειμένων μιας εφαρμογής android. Το View είναι υπεύθυνο για την σχεδίαση(rendering) του αντικειμένου στην οθόνη καθώς και για τον χειρισμό όποιου συμβάντος(event) προκύψει. (developers, n.d.) Ένα view μπορεί να έχει ένα παιδί(child) (ή περισσότερα στην περίπτωση των view group που θα αναλυθεί αργότερα), δημιουργώντας έτσι μια δενδρική κατάταξη τους. Υπάρχουν ακόμη και πολλές υποκατηγορίες views οι οποίες μπορούν και λειτουργούν με μερικές εξειδικευμένες δυνατότητες όπως το να εμφανίζουν κείμενο, φωτογραφίες από αρχεία και άλλα. Επίσης, μια ιδιαίτερη δυνατότητα που χρησιμοποιείται λιγότερο συχνά είναι η χρήση τους σαν controllers, δηλαδή πραγματοποιούν τον έλεγχο των λειτουργιών άλλων views.

Ανάλογα με το είδος του view, υπάρχουν κάποιες ιδιότητες τις οποίες μπορεί να χρησιμοποιήσει. Αυτές οι ιδιότητες ονομάζονται properties και συνήθως είναι της μορφής key - value. Δηλαδή το view μπορεί να έχει μια ιδιότητα όπου το όνομα της είναι το key (μέσω του οποίου ο προγραμματιστής έχει πρόσβαση στην τιμή του property) και από την άλλη είναι η τιμή (value) που έχει το property και μπορεί να διαφέρει σε κάθε instance του view. Μερικά παραδείγματα properties θα μπορούσαν να είναι: σε ένα textView να

υπήρχε ένα `property text` το οποίο αποθηκεύει το κείμενο που πρέπει να εμφανιστεί στην οθόνη, ή θα μπορούσε να ήταν `properties` που κρατάν α) τις διαστάσεις του `View`, β) ένα `link` για να οδηγήσει το χρήστη κάπου ή να τραβήξει δεδομένα, γ) ένα `reference` με κάποια μέθοδο που πρέπει να εκτελεστεί όταν συμβεί κάτι στο `view` και άλλα. Γενικά υπάρχει μεγάλη ποικιλία σε `properties` και στις λειτουργίες τους. Κάποια πιο ιδιαίτερα `properties` μπορούν να κάνουν κάποιες πιο κρίσιμες λειτουργίες, όπως να ελέγχουν αν ένα `view` είναι ορατό ή αν δεν εμφανίζεται στην οθόνη. Μια ακόμη ιδιότητα που έχουν είναι να μπορούν να ελέγχουν το `focus`. Κάθε στιγμή το λειτουργικό μπορεί να κάνει `focus` σε ένα `view`. Συνήθως, το `focus` παρέχεται και μεταβάλλεται ανάλογα με το `input` του χρήστη, αν και μπορεί ο προγραμματιστής να ζητήσει το `focus` σε κάποιες περιπτώσεις, χρησιμοποιώντας την συνάρτηση `requestFocus`. Ένα παράδειγμα για την χρήση του `focus` θα μπορούσε να είναι `EditText View`, το οποίο όταν πατήσει πάνω του ο χρήστης και κατά συνέπεια το λειτουργικό του δώσει το `focus`, τότε αυτό καλεί το πληκτρολόγιο να εμφανιστεί για να δεχτεί κείμενο από τον χρήστη.

Τα `views` έχουν ένα ιδιαίτερο τρόπο για να λαμβάνουν το `input` του χρήστη. Χρησιμοποιούν μια σειρά από `callbacks` που στην συνέχεια γίνονται διαθέσιμα κάνοντας `override` κάποια `callbacks` ορισμένων `interfaces`. Αυτή η αρχιτεκτονική είναι πιο απλή για τον προγραμματιστή αφού δεν χρειάζεται να κάνει `extend` τα αρχικά `callback` σε κάθε `View` ξεχωριστά. Αυτά τα συγκεκριμένα `interfaces` τα ονομάζουμε `event listeners`, αφού πρωτίστως η δουλειά τους είναι να μεταφέρουν τις εντολές που θέλει να εκτελέσει ο προγραμματιστής στην περίπτωση μιας αλληλεπίδρασης του `View` με τον χρήστη. Όλη αυτή η διαδικασία είναι απαραίτητη αφού ο έλεγχος των `input` για λόγους απόδοσης και ευκολίας γίνεται από το λειτουργικό σύστημα, οπότε σε αυτή την περίπτωση ο προγραμματιστής αντιμετωπίζει μια ασύγχρονη λειτουργία που δε γνωρίζει το χρονικό διάστημα που θα εκτελεστεί η μέθοδος.

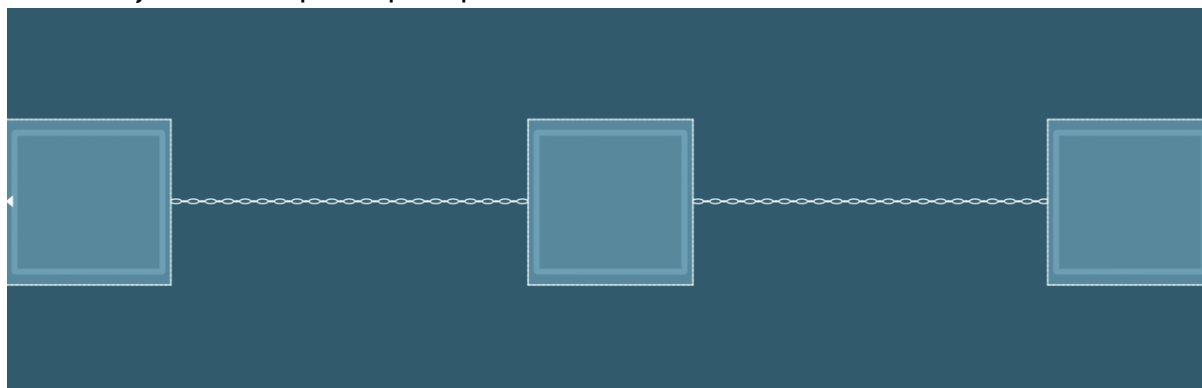
3. ViewGroup

Το `ViewGroup` είναι μια ειδική κατηγορία `View` που έχει την ικανότητα να δέχεται πολλαπλά παιδιά. Πάνω σε αυτό το `view` βασίζονται τα `layout views` τα οποία πιάνουν ένα μέρος ή ολόκληρη την οθόνη δίνοντας τη δυνατότητα στους προγραμματιστές να φτιάξουν νέα πιο περίπλοκα `Views` και `layout` που χρησιμοποιούν ως βάση, πιο απλά `Views`.

Μερικά από τα πιο συνηθισμένα `ViewGroup` που χρησιμοποιούνται για την δημιουργία `layout` είναι τα `RelativeLayout`, `LinearLayout`, `FrameLayout`, `ConstraintLayout` και άλλα. Ξεκινώντας με τον `LinearLayout` (developers, n.d.), αυτό τοποθετεί όλα τα παιδιά του στην οθόνη το ένα μετά το άλλο σε μια γραμμή είτε οριζόντια είτε κάθετα. Το `RelativeLayout` από την άλλη (developers, n.d.), όπως λέει και το όνομα, τοποθετεί τα `Views` του σε σύγκριση του ενός με τη θέση κάποιου άλλου `View` ή σε σύγκριση με το πλαίσιο χώρου

που πιάνει το RelativeLayout. Με άλλα λόγια, στο κάθε View για μια πλευρά του, επιλέγεται μια πλευρά ενός άλλου View ή του RelativeLayout για να πάει να τοποθετηθεί η μια πλευρά δίπλα στην άλλη.

Ένα άλλο layout view που μοιάζει η λειτουργία του με το RelativeLayout είναι το ConstraintLayout (developers, n.d.). Το ConstraintLayout παίρνει τη λογική του RelativeLayout και τη πάει ένα βήμα παραπέρα. Εδώ, αντί να δηλώνεις μόνο μια πλευρά, που μπορεί να πάει και να “κολλήσει” ένα View, μπορείς να δηλώσεις πολλές πλευρές. Ας πούμε τώρα ότι θέλουμε ένα View να είναι από τα αριστερά ενός άλλου View και από τα δεξιά του να έχει ένα ConstraintLayout, αλλά αυτή η απόσταση είναι πολύ μεγάλη και είναι αδύνατον και οι δυο πλευρές να κολλήσουν μεταξύ τους. Σε τέτοια περίπτωση το constraint πάει και τοποθετεί το View στην μέση της απόστασης της πλευράς του άλλου View και του ConstraintLayout που επιλέχθηκαν, οπότε και οι δύο πλευρές του View απέχουν το ίδιο από της πλευρές των άλλων αντικειμένων που έχει γίνει constraint πάνω. Απεικονίζεται καλύτερα στην παρακάτω εικόνα.



Εικόνα 28: Designing complex UI using Android Constraint Layout | Code magic Blog

4. Activity

Στις εφαρμογές και στις ιστοσελίδες πάντα υπάρχει μια σελίδα-οθόνη που πιάνει όλο το διαθέσιμο χώρο και πρακτικά εντός της υπάρχει η εφαρμογή. Ο προγραμματιστής μπορεί να διαλέξει να αλλάξει την σελίδα που βλέπει ο χρήστης, αλλάζοντας τη διαθέσιμη πληροφορία και λειτουργικότητα, που είναι διαθέσιμη στα χέρια του χρήστη. Στο android αυτή η σελίδα ονομάζεται Activity. Το Activity αποτελείται από δύο αρχεία, ένα σε Java με όλη την λογική και τη λειτουργικότητα της σελίδας και ένα σε XML για τη δημιουργία των UI αντικειμένων. Αυτά τα δύο αρχεία κώδικα μπορούν να “επικοινωνούν” μεταξύ τους αφού από την Java μπορεί ο προγραμματιστής να αλλάξει, να προσθέσει ή να αφαιρέσει XML αντικείμενα από το αντίστοιχο αρχείο. Αυτό είναι δυνατόν αφού στην αρχικοποίηση του activity στην java δημιουργείται ένα reference που δείχνει με ποιο XML αρχείο αντιστοιχεί, μέσω της εντολής setContentView. Επίσης, αφού έχει φτιαχτεί ένα reference στο XML layout τότε μπορούν να δημιουργηθούν και references προς συγκεκριμένα

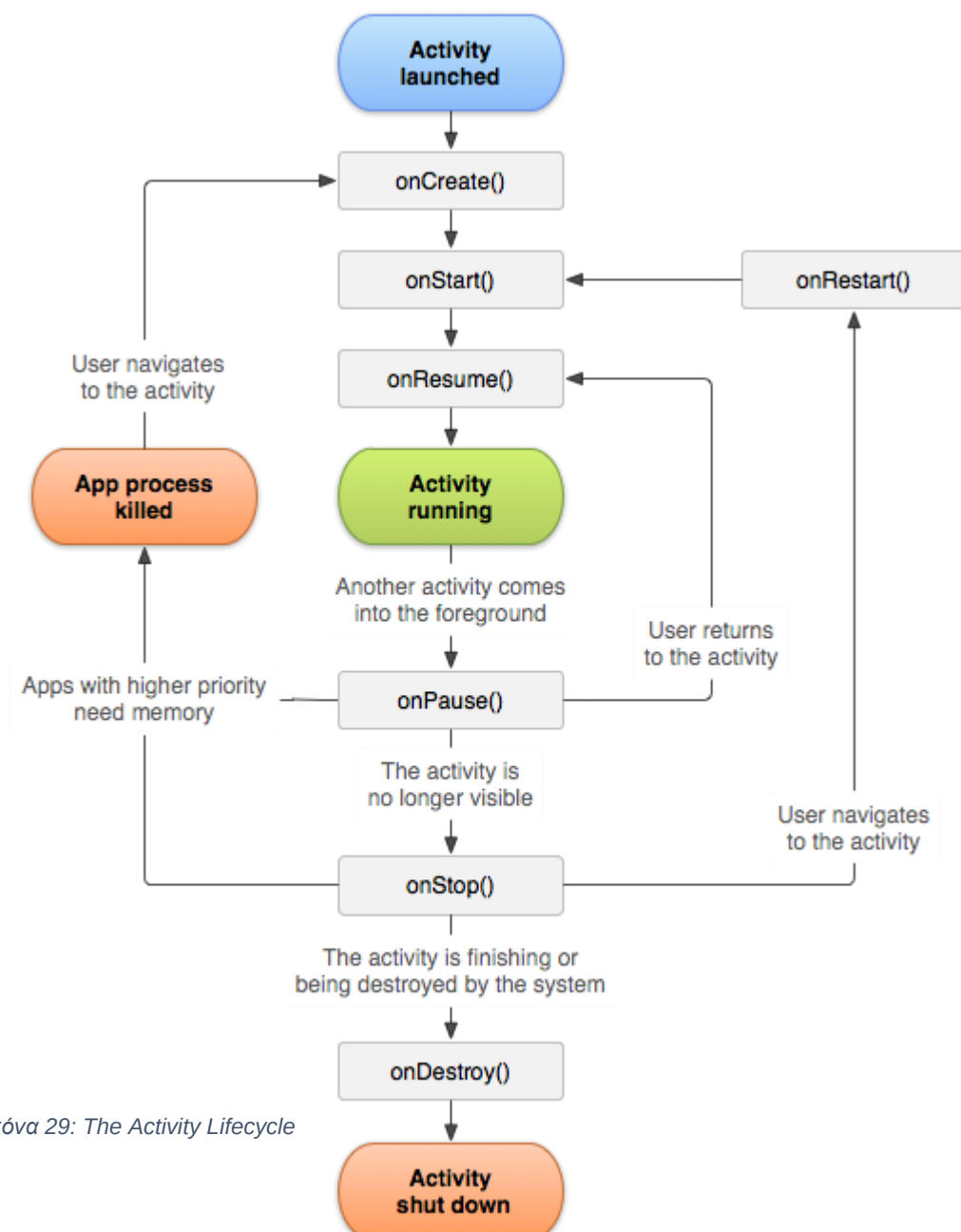
Views χρησιμοποιώντας την `findViewById`. Αυτή χρησιμοποιεί μια τιμή “id” του View στην XML για να μπορέσει να το εντοπίσει (προφανώς το id είναι μοναδικό αναγνωριστικό).

Πολλές φορές όμως τα Views δεν υπάρχουν εξ αρχής στο αρχείο και πρέπει να δημιουργούνται δυναμικά κατά την εκτέλεση. Σε τέτοιες περιπτώσεις χρησιμοποιείται ο `LayoutInflater`. Αυτός κάνει “inflate” το view, δηλαδή με άλλα λόγια “φουσκώνει” την οθόνη προσθέτοντας της καινούργια αντικείμενα.

5. Activity Lifecycle

Από την στιγμή που ξεκινάει ένα Activity μέχρι και την στιγμή που καταστρέφεται περνάει αυτόματα μέσα από μια σειρά προκαθορισμένων καταστάσεων που επιτρέπει στον προγραμματιστή να ελέγχει την εφαρμογή και τις αλλαγές που προκύπτουν από αυτή την αλλαγή καταστάσεων. Η κάθε μια από αυτές τις καταστάσεις περιέχει μια συνάρτηση η οποία εκτελείται αυτόματα από την εφαρμογή όταν εισέλθει σε αυτή την κατάσταση. Όταν η εφαρμογή δίνει εντολή σε ένα Activity να ξεκινήσει, αυτό πρώτα πάει και εκτελεί την `onCreate`. Εκεί ο προγραμματιστής μπορεί να αρχικοποιήσει μεταβλητές, να αρχίσει session με βάσεις δεδομένων, να συνδεθεί με Application Programming Interfaces (API's) και άλλα. Στην συνέχεια καλείται η `onStart`, η οποία ξεκινά την εφαρμογή με βάση τις αρχικές αρχικοποιήσεις. Παρόμοια δουλειά κάνει και η `onResume` που έρχεται μετά. Η διαφορά των δύο είναι ότι αργότερα, στην `onPause` αποθηκεύεται προσωρινά η τρέχουσα κατάσταση και με την `onResume` συνεχίζεται η λειτουργία από εκεί που είχε μείνει ο χρήστης, ενώ με την `onStart` κάνει πάντα μια “καθαρή” εκκίνηση με τις αρχικές τιμές (εκτός και αν ο προγραμματιστής χειροκίνητα αποθηκεύσει μερικώς ή ολικώς τις μεταβλητές του Activity). Στην συνέχεια από την `onResume`, το Activity είναι ενεργό και έτοιμο να αλληλοεπιδράσει με τον χρήστη. Αν τώρα ένα νέο Activity έρθει στο μπροστά μέρος της οθόνης χωρίς όμως να πιάνει ολόκληρη την οθόνη (π.χ. Εξαιτίας διαφάνειας ή ενός Dialog), άρα το προηγούμενο activity είναι έστω και ελάχιστα εμφανίσιμο στην οθόνη, τότε καλείται η `onPause` που αποθηκεύει προσωρινά την κατάσταση της εφαρμογής, δεν επιτρέπει στο χρήστη να διαδράσει με την από πίσω activity και όταν κλείσει αργότερα η μπροστινή activity και επανέρθει μπροστά η προηγούμενη, καλείται στην προηγούμενη η `onResume` όπου θα επαναφέρει την εφαρμογή στην τελευταία της κατάσταση, σαν ο χρήστης να μην “έφυγε” ποτέ. Αν το activity εξαφανιστεί τελείως από την οθόνη ή ο χρήστης αλλάξει εφαρμογή, κλείσει την οθόνη κτλ., τότε καλείται η `onStop` μετά την `onPause`. Εκεί μερικές φορές ο προγραμματιστής μπορεί να θέλει να αποθηκεύσει κάποιες από τις μεταβλητές για να μην γίνει ολική επανεκκίνηση αν επιστρέψει ο χρήστης. Αν έχει καλεστεί η `onClose` και ο χρήστης επιστρέψει στο Activity τότε καλείται η `onRestart` και στην συνέχεια η `onStart`. Δουλειά του `onRestart`, παρομοίως με της `onCreate`, είναι να αρχικοποιήσει μεταβλητές, κατά κύριο λόγο. Ωστόσο, σε αντίθεση με την `onCreate` που κάνει μια “καθαρή”

αρχικοποίηση, στην `onRestart` συνηθίζεται να χρησιμοποιούνται ότα δεδομένα αποθηκεύτηκαν στην `onStop` από τον προγραμματιστή. Από εκεί και πέρα, η εφαρμογή συνεχίζει, χρησιμοποιώντας αυτήν την κυκλική ροή. Κάποια στιγμή όμως η εφαρμογή κλείνει είτε από το χρήστη είτε από το σύστημα. Σε αυτή την περίπτωση πρώτα καλείται η `onPause` με την `onStop` και στο τέλος η `onDestroy`. Η `onDestroy` θα απελευθερώσει την μνήμη από τις μεταβλητές, και ότα άλλα resources χρησιμοποιεί η εφαρμογή, για να μπορέσει τελικά να τερματιστεί.



Εικόνα 29: The Activity Lifecycle

6. Fragments

Μερικές φορές θέλουμε απλώς να βάλουμε ένα ολόκληρο layout να καλύπτει ένα μέρος της οθόνης και να λειτουργεί ανεξάρτητα ή τουλάχιστον ημιανεξάρτητα από το υπόλοιπο UI (developers, n.d.). Κατά κάποιο τρόπο θα μπορούσε να πει κανείς, ότι θέλει να βάλει ένα Activity ως child ενός άλλου Activity. Αυτή η λειτουργικότητα μπορεί να επιτευχθεί με τη χρήση των Fragment. Τα Fragment έχουν την ιδιότητα ότι μπορούν να επαναχρησιμοποιηθούν σε διάφορα layout και Activities. Ακριβώς και όπως ένα Activity, το Fragment μπορεί να έχει και να ελέγχει το δικό του layout, έχει το δικό του lifecycle όπου λειτουργεί παρόμοια με αυτό ενός Activity και τέλος μπορεί να ελέγχει τα δικά του event input. Τα Fragments είναι δυνατόν να βρίσκονται εντός ενός Activity ή κάποιου άλλου Fragment.

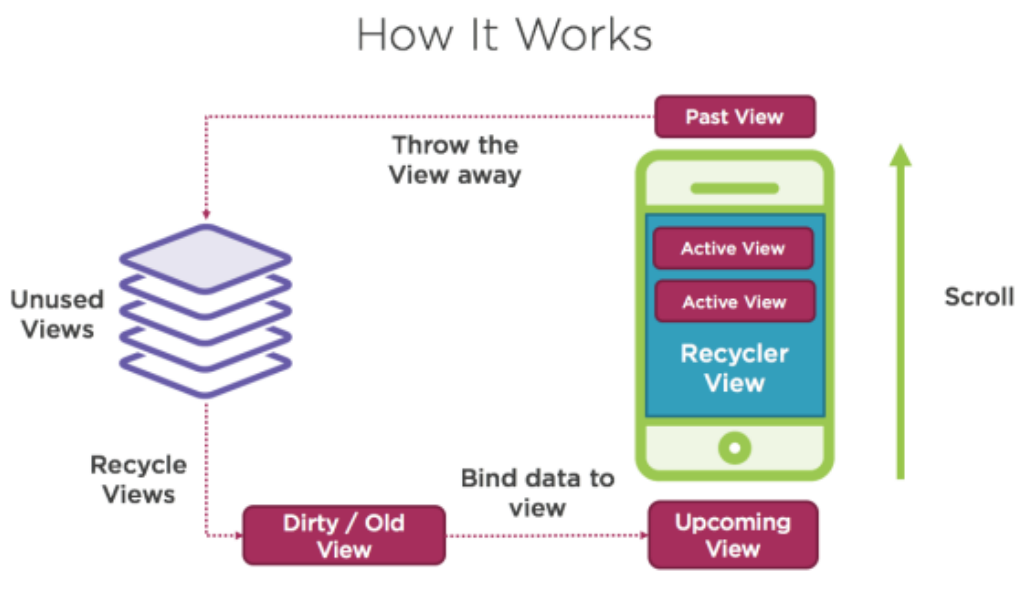
7. Adapters

Ορισμένες φορές ο προγραμματιστής χρειάζεται να φτιάξει δομές παρόμοιων View με χρήση ωστόσο, κάποιων διαφορετικών δεδομένων. Εδώ εισέρχονται στο παιχνίδι οι Adapters. Ένας αντάπτορας χρησιμοποιείται ως μεσάζων μεταξύ του View και των δεδομένων. Δεν μπορεί οποιοδήποτε View να χρησιμοποιήσει αντάπτορες. Μόνο Views της κλάσης AdapterView και μόνο οι απόγονοι αυτής της κλάσης μπορούν να τους χρησιμοποιήσουν. Οι αντάπτορες προσφέρουν πρόσβαση στα δεδομένα και για αυτό το λόγο μια συχνή λειτουργία που μπορούν να κάνουν είναι, να δημιουργούν δυναμικά ένα View για κάθε εγγραφή που υπάρχει στα δεδομένα (στην περίπτωση που γίνεται επεξεργασία δεδομένων με την μορφή μιας λίστας ή ενός map και άλλων όμοιων δομών δεδομένων).

8. Recycler View

Ένα ιδιαίτερα συνηθισμένο View με την μορφή λίστας αποτελεί το RecyclerView. Το Recycler View παίρνει κάποια δεδομένα και με βάση αυτά κατασκευάζει μια λίστα από δυναμικά views. Η λίστα μπορεί να είναι γραμμική με οριζόντια ή κάθετη κατεύθυνση, ή μπορεί ακόμα να έχει και πιο περίπλοκες δομές όπως ένα Grid, για παράδειγμα. Ωστόσο, το όνομά του το πήρε από την ικανότητά του να “ανακυκλώνει” τα views της λίστας. Η ανακύκλωση λειτουργεί με τον εξής τρόπο. Όταν ένα view της λίστας βγει εκτός των ορίων της οθόνης αντί να καταστρέφεται ή να αποθηκεύεται στη μνήμη, σπαταλώνοντας μνήμη και υπολογιστική ισχύ, πηγαίνει στο άλλο άκρο τη λίστας (δηλαδή αν όταν βγήκε από την λίστα ήταν στο τέλος της τότε θα πάει στην αρχή της λίστας, αντίστοιχα αν ήταν στην αρχή της λίστας θα πάει στο τέλος της). Από εκεί ανανεώνει τα δεδομένα του με το επόμενο

αντικείμενο που είναι να εμφανιστεί στην οθόνη αν ο χρήστης συνεχίσει το scroll προς την ίδια κατεύθυνση, εξοικονομώντας τη διαδικασία της συνεχούς καταστροφής και δημιουργίας νέων view από την αρχή.



Εικόνα 30: Android Kotlin Basics- RecyclerView and Adapters- Alex Dunn

Κάθε αντικείμενο της λίστας ορίζεται από ένα αντικείμενο κλάσης ViewHolder. Το RecyclerView χρησιμοποιεί ένα Adapter για να θέσει τα δεδομένα σέ ένα αντικείμενο ViewHolder. Τέλος, ο layout manager θέτει τον τρόπο που θα εμφανίζονται αυτά τα View Holder αντικείμενα στην οθόνη. Η βιβλιοθήκη περιέχει ήδη έτοιμες υλοποιήσεις για γραμμικές λίστες και λίστες σε μορφή πλέγματος, ωστόσο είναι εφικτό ο προγραμματιστής να φτιάξει το δικό του layout και τη λογική του πως θα εμφανίζονται - εξαφανίζονται τα views από την οθόνη.

9. FirestoreRecyclerViewAdapter

Το Firestore RecyclerView Adapter είναι ένα View που χρησιμοποιεί το απλό RecyclerView και το κάνει extend προσθέτοντάς του και την λειτουργικότητα του αντάπτορα στο ίδιο View. Ωστόσο, το query για την βάση δεδομένων δίνεται ως παράμετρος κατά την δημιουργία του αντικειμένου και το view αυτόματα τραβά τα δεδομένα(και τα ανανεώνει αυτόματα όταν υπάρξει κάποια αλλαγή στην βάση) και τα παρέχει στον προγραμματιστή κατά τη διαδικασία της δημιουργίας του ViewHolder αντικειμένου για να μπορέσει να τα χρησιμοποιήσει.

10. Bottom Navigation View

Το View αυτό δημιουργεί ένα bottom navigation bar με βάση την αρχιτεκτονική του Material Design (developers, n.d.). Στόχος του είναι να προσφέρει στον χρήστη ένα κοινό τρόπο σε όλη την εφαρμογή για να πλοηγείται στις βασικές οθόνες της. Τα κουμπιά που μπορεί να επιλέξει ο χρήστης για να μεταφερθεί σε μια νέα οθόνη βρίσκονται σαν menu items μέσα σε ένα menu view.

11. Menus

Τα Menus αποτελούν views τα οποία προσφέρουν ένα ιδικό API για να ικανοποιήσουν τις βασικές και δημοφιλείς λειτουργικότητες που μπορούν να έχουν τα διάφορα μενού (αφού τα μενού είναι ένα από τα πιο συνηθισμένα είδη λίστας στις κινητές εφαρμογές). Οι βασικοί λόγοι που προτείνεται η χρήση των menus είναι γιατί προσφέρουν μια οργανωμένη δομή στο κώδικα καθώς και η δημιουργία διαφορετικών menu από τον προγραμματιστή σε κάποιες περιπτώσεις, ανάλογα με την έκδοση του λογισμικού ή των τεχνικών προδιαγραφών της συσκευής, που τρέχει η εφαρμογή. Υπάρχουν τρία είδη μενού (developers, n.d.), το option menu, το context menu και το popup menu. Το πρώτο από αυτά, το option, χρησιμοποιείται για πιο πρωτεύουσες λειτουργίες που τείνουν να έχουν πιο γενική επίδραση πάνω στην εφαρμογή ή στη συγκεκριμένη οθόνη. Η δεύτερη, το context menu εμφανίζεται στην οθόνη μετά από τη διάδραση του χρήστη (συνήθως πατώντας ένα view συνεχόμενα για ένα σχετικά μεγάλο χρονικό διάστημα ώστε να γίνει trigger από το λειτουργικό ως ένα long press). Αυτό το μενού χρησιμοποιείται συνήθως για να παρέχει μια λίστα από δευτερεύουσες λειτουργίες που μπορεί να επιλέξει ο χρήστης για ένα συγκεκριμένο αντικείμενο. Η τελευταία κατηγορία, τα popup menus χρησιμοποιούνται για να παρέχουν επιπλέον λειτουργίες ή για να κεντρίσουν την προσοχή του χρήστη μια συγκεκριμένη στιγμή. Σε αντίθεση με τα context menus, τα popup menus δεν πρέπει σε καμία περίπτωση να μεταβάλλουν το περιεχόμενο του αντικειμένου - View απ' όπου φτιάχτηκαν.

12. Menu Item

Το menu item είναι ένα ιδικό view που δέχεται σαν παιδί του το menu και το καθένα αναπαριστά μια επιλογή του menu. Οι βασικές ιδιότητες που μπορεί να έχει (όλες προαιρετικές) είναι ένα εικονίδιο για να τραβήξει την προσοχή του χρήστη και να προσφέρει ένα πιο γρήγορο τρόπο να αναδείξει τη λειτουργία της εκάστοτε επιλογής. Ακόμη, περιέχει ένα τίτλο ο οποίος είναι και ο βασικότερος και πιο δημοφιλής τρόπος να εξηγήσει τη λειτουργία στο χρήστη. Τέλος, περιέχει μια ιδιότητα για να ενεργοποιείται και

απενεργοποιείται το ίδιο, όταν ο προγραμματιστής θέλει να παρέχει την υπηρεσία ή αντίστοιχα όταν θέλει να την αποτρέψει.

13. Menu Group

Λίστα με menu items με σκοπό την κατηγοριοποίησή τους και παροχή δυνατότητας για μαζικές αλλαγές ιδιοτήτων σε ολόκληρα σύνολα από menu items.

ArCore

Το ARCore είναι μια βιβλιοθήκη για συσκευές Android που αναλαμβάνει να “σκανάρει” το περιβάλλον που φαίνεται από την κάμερα δημιουργώντας ένα τρισδιάστατο χάρτη συντεταγμένων, τον οποίο οι προγραμματιστές μπορούν να χρησιμοποιήσουν για να τοποθετήσουν τα τρισδιάστατα αντικείμενά σε σταθερά σημεία στον περιβάλλον.

1. Motion Tracking

Το ARCore χρησιμοποιεί την κάμερα, με την κίνηση του χρήστη, η κάμερα τραβάει εικόνες από διαφορετικά σημεία και γωνίες, και έτσι μπορεί να εντοπίζει κάποια σημεία στην εικόνα, που θεωρεί ότι μπορεί να υπολογίσει και να δημιουργήσει ένα τρισδιάστατο χάρτη συντεταγμένων, μέσω μιας διαδικασίας που ονομάζει “simultaneous localization and mapping” (SLAM). Αυτά τα σημεία, του φυσικού κόσμου, ονομάζονται feature points και εφόσον μπορεί να συνεχίσει να τα εντοπίζει μέσω της κάμερας, μπορεί να βρίσκει ακριβώς σε ποιο σημείο και με ποια γωνία πρέπει να εμφανίζεται το τρισδιάστατο αντικείμενο.

2. Planes

Όταν βρίσκει ένα feature point, προσπαθεί να υπολογίσει, χρησιμοποιώντας αυτό και άλλα feature points, αν τα σημεία αυτά είναι μέρος μιας επιφάνειας. Αν κριθεί επιτυχής η διαδικασία, τότε δημιουργεί στην επιφάνεια ένα Plane, δηλαδή ένα χώρο (την επιφάνεια) από τον οποίο μπορεί ο προγραμματιστής να τοποθετήσει αντικείμενα σε οποιοδήποτε σημείο του.

Τα feature points για να εντοπιστούν χρησιμοποιούν το contrast μεταξύ γειτονικών pixel, έτσι αντικείμενα του περιβάλλοντος με πιο “τραχύ” texture και γωνίες (λόγω των σκιών) και χρώματα που έχουν μεγάλο contrast μεταξύ τους είναι ευκολότερο να παράξουν ένα σημείο ενώ σε πιο επίπεδα και μονόχρωμα γειτονικά αντικείμενα είναι δυσκολότερο να βρεθεί σημείο.

3. Anchors

Πάνω στα feature points και στην επιφάνεια του Plane, ο προγραμματιστής μπορεί να επιλέξει ένα σημείο για να εμφανίσει το τρισδιάστατο αντικείμενο που επιθυμεί. Αυτό το σημείο ονομάζεται Anchor. Αφού, κατά κάποιο τρόπο, θα μπορούσε να πει κάποιος ότι το αντικείμενο αγκυλώνεται σε ένα συγκεκριμένο σημείο του φυσικού κόσμου ανεξάρτητα με τη θέση και τη γωνία της συσκευής. Τα feature points και τα Planes, εξαιτίας της ιδιότητας τους να είναι τα σημεία εντοπισμού που χρησιμοποιεί η συσκευή για να αναλύσει το περιβάλλον, λέγονται ακόμη Trackables. Ο λόγος που απαιτούνται τα Anchors είναι ότι η κινητή συσκευή βρίσκεται συνεχώς σε κίνηση (έστω και μικρή). Ο χρήστης μπορεί να έρθει πιο κοντά ή να απομακρυνθεί από το σημείο που βρίσκεται το τρισδιάστατο αντικείμενο ή να αλλάξει την γωνία από την οποία το βλέπει. Για αυτό απαιτείται ένα σύστημα για να μπορούν να εντοπίζονται τέτοιες αλλαγές, όπως και το μέγεθος, η γωνία και το σημείο στην οθόνη που εμφανίζεται το αντικείμενο, τα οποία θα μεταβάλλονται ανάλογα.

4. World Space & Pose

Το World Space είναι ο τρισδιάστατος ψηφιακός χάρτης συντεταγμένων που δημιουργεί το ARCore για να κάνει ανάλυση του φυσικού περιβάλλοντος του χρήστη. Το Pose δείχνει τις συντεταγμένες και την γωνία που βρίσκεται το ψηφιακό αντικείμενο στο World Space.

Flutter

1. Το Framework και τα Widgets

Το Flutter είναι ένα open source framework που δημιουργήθηκε από την Google και χρησιμοποιεί τη γλώσσα προγραμματισμού Dart. Ο προγραμματιστής με βάση ένα κώδικα προσπαθεί να παράγει μια client εφαρμογή για οποιοδήποτε (δημοφιλές)

λειτουργικό σύστημα επιθυμεί. Μέχρι σήμερα, τα λειτουργικά συστήματα που υποστηρίζει / μπορεί να κάνει export το Flutter είναι τα Android, IOS, Mac OS, Windows και σε διαδικασία beta βρίσκεται το export, για λειτουργικά Linux. Ακόμη, σε πιο αρχικό στάδιο βρίσκεται και η ανάπτυξη embedded εφαρμογών. Το Flutter βασίζεται σε μια δενδροειδή αρχιτεκτονική όπως για παράδειγμα ισχύει και στην HTML, στην οποία το ένα αντικείμενο περιέχεται μέσα στο άλλο σαν παράμετρος. Στην ορολογία του Flutter αυτά τα “αντικείμενα” αποκαλούνται Widgets. Η λογική είναι ότι κάθε Widget κάνει inherit ένα άλλο Widget μέχρι τελικά να φτάσει να κάνει inherit την abstract κλάση Widget. Με αυτήν την λογική “τα πάντα είναι Widgets” όπως είναι συνηθισμένο να λένε και εντός του Community του Flutter.

2. State

Στην πραγματικότητα, στόχος των Widget είναι να περιγράψουν το πως πρέπει να μοιάζει το UI. Για να επιτευχθεί αυτό, τα Widget πρέπει να παρέχουν πληροφορίες για την τρέχουσα διαμόρφωση του UI καθώς και την κατάσταση (state) του. Όταν η κατάσταση αλλάζει τότε το framework αναδημιουργεί το UI με βάση τα νέα δεδομένα του widget συγκρίνοντάς τα μάλιστα με τα παλαιότερα για να μπορέσει το framework να βρει όσο το δυνατόν λιγότερα στοιχεία του UI που χρειάζονται αλλαγή. (Flutter, 2022) Με την εκκίνηση της εφαρμογής, η πρώτη μέθοδος που καλεί είναι η runApp() η οποία παίρνει σαν παράμετρο ένα Widget και το τοποθετεί σαν root στο δέντρο των widget. Όταν το framework πάει να εμφανίσει το widget θα του δώσει αυτομάτως διαστάσεις ίσες με τις διαστάσεις ολόκληρης της οθόνης.

3. Stateless & Statefull

Όπως αναφέρθηκε και προηγουμένως ένα widget χρειάζεται να έχει ένα state για να μπορεί το framework να το εμφανίσει στην οθόνη. Αυτό επιτυγχάνεται αν όλα τα widget έμμεσα ή άμεσα πρέπει να κάνουν inherit τα StatelessWidget ή Statefullwidget. Η διαφορά μεταξύ των δύο είναι ότι στα Stateless υπάρχει ένα αρχικό state το οποίο δεν μπορεί να αλλάξει ενώ στο Statefull ο προγραμματιστής μπορεί να αλλάξει ή να ανανεώσει το state με σκοπό, μέσω της αλλαγής των δεδομένων να αλλάξουν το UI component. Ο λόγος που υπάρχει αυτή η διαφοροποίηση είναι γιατί στα Stateless Widgets το framework δεν χρειάζεται να ελέγχει για αλλαγές, αυξάνοντας με αυτόν τον τρόπο το performance. Ενώ, από την άλλη τα Statefull Widgets παρέχουν στον προγραμματιστή τη δυνατότητα να ανανεώνει το UI γρήγορα και εύκολα χρησιμοποιώντας πάντα μία παρόμοια λογική-αρχιτεκτονική που ταυτόχρονα είναι και αποδοτική.

4. Build

Η λογική για όλα τα Widget βρίσκεται μέσα στην μέθοδο build. Αυτή καλείται αυτόματα από widget και σκοπό έχει να εμφανίσει το UI στην οθόνη. Για να το πετύχει αυτό η build χρησιμοποιεί άλλα, ήδη διαθέσιμα, Widget που και αυτά με την σειρά τους μπορούν να καλέσουν άλλα Widget με τελικό σκοπό στο βάθος αυτής της στοίβας να βρεθεί ένα render object το οποίο περιγράφει το πώς θα εμφανιστεί το αντικείμενο στην οθόνη και που θα εμφανιστεί στην οθόνη, με βάση τις παραμέτρους που του δόθηκαν. Στην περίπτωση των Statefull Widgets η μέθοδος build είναι αυτή της οποίας όταν τα δεδομένα της αλλάξουν, τότε το framework ξεκινάει τη διαδικασία του rebuild δηλαδή να ξανατρέξει την build χρησιμοποιώντας τα νέα δεδομένα αλλά ελέγχοντας ταυτοχρόνως τι έχει παραμείνει το ίδιο και δεν χρειάζεται αλλαγή και το τι άλλαξε και απαιτεί re-render.

5. Material App/Design

Ένα από τα πρώτα Widget που χρησιμοποιούνται συνήθως στην αρχή της εφαρμογής είναι το MaterialApp ή το CupertinoApp. Η διαφορά τους είναι καθαρά αισθητική καθώς τα παιδιά του ενός ακολουθούν την αρχιτεκτονική του Material Design, ενός συνόλου από guidelines δηλαδή, για το πώς να σχεδιάσεις τα component της εφαρμογής με βάση ότι όλα τα component πρέπει να εμφανίζονται ως φυσικά αντικείμενα ακολουθώντας τους κανόνες των φυσικών αντικειμένων αλλά στον δισδιάστατο κόσμο. Από την άλλη το CupertinoApp χρησιμοποιεί την αρχιτεκτονική του Cupertino Design το οποίο αντί να προσπαθεί να εμφανίσει κάτι που φαίνεται φυσικό χρησιμοποιεί μια πιο μινιμαλιστική προσέγγιση χρησιμοποιώντας αρκετά συχνά και την ημιδιαφάνεια κάτι που απαγορεύεται στο Material Design. Αυτές οι δύο αρχιτεκτονικές είναι οι πιο δημοφιλείς στον κόσμο των κινητών συσκευών αφού η μια έχει δημιουργηθεί από την Google για το Android ενώ η άλλη φτιάχτηκε από την Apple για το IOS. Και τα δύο αυτά Widget όμως έχουν παρόμοιες λειτουργίες. Ο βασικός τους σκοπός είναι να δημιουργήσουν ένα σύνολο από Widget που είναι τελείως απαραίτητα για να φτιαχτεί μια εφαρμογή όπως για παράδειγμα το theme που περιέχει κάποιες default τιμές για τις χρωματικές ιδιότητες κάθε ίδιου τύπου widget στην εφαρμογή καθώς και άλλες στιλιστικές επιλογές. Αυτές τις τιμές έχει την δυνατότητα ο προγραμματιστής να τις πειράξει, αλλάζοντας έτσι και το style παρόμοιων widget από ένα σημείο χωρίς την ανάγκη να επαναλαμβάνει τον ίδιο κώδικα.

6. Navigator

Ένα ιδιαίτερα σημαντικό Widget που δημιουργούν τα παραπάνω Widget είναι ο Navigator. Ο Navigator μπορεί και φτιάχνει μια στοίβα με widget(που ονομάζονται routes

και μπορεί να τα σκεφτεί κάποιος ως μια σελίδα της εφαρμογής) που δηλώνει την ιεραρχία των routes της εφαρμογής. Σε αυτήν την στοίβα αν θέλω να προσθέσω κάτι μπορώ να κάνω push για να βάλω ένα route στην αρχή της λίστας ή μπορώ να κάνω pop για αφαιρεθεί το πιο πρόσφατο route. Οπότε για παράδειγμα, αν είμαι σε μια σελίδα και θέλω να πάω στην προηγούμενη, απλώς ζητάω από τον Navigator να κάνει pop στην στοίβα του και γυρνάω πίσω. Ο Navigator δίνει την επιλογή στον προγραμματιστή να δώσει “ονόματα” στις σελίδες-widgets του και από εκεί να τις καλεί και να τις βάζει στην αρχή της στοίβας ή του δίνει και κάποιες άλλες επιλογές όπως, το να αδειάσει την στοίβα και μετά να βάλει το route ή να κάνει pop στην στοίβα μέχρι ενός συγκεκριμένου route και μετά να κάνει push το νέο route κτλ.

Για να επιλεγεί ένα route το οποίο θα είναι το προεπιλεγμένο που θα εμφανίζεται όταν το MaterialApp τρέχει, υπάρχει η παράμετρος Home. Αυτή δέχεται σαν τιμή ένα Widget το οποίο γίνεται από το framework το αρχικό route. Στην συνέχεια, το πιο συνηθισμένο Widget που χρησιμοποιείται σε αυτό το σημείο, πριν ο προγραμματιστής αρχίσει να προσθέτει τα εξειδικευμένα widget, είναι το Scaffold. Το Scaffold δημιουργεί τα βασικά UI στοιχεία που απαιτεί μια σύγχρονη εφαρμογή, όπως είναι το AppBar, τα drawers, τα bottom sheets, τα FOB (Floating Action Button) και άλλα.

7. Βασικά Material Widget

Υπάρχουν κάποια βασικά Widget που εφαρμόζουν τις αρχιτεκτονικές των Material και Cupertino Design. Εδώ θα περιγραφθούν τα σημαντικότερα με βάση το Material Design. Συγκεκριμένα, το AppBar είναι μια μπάρα στο πάνω μέρος της οθόνης που μπορεί να περιέχει τον τίτλο της εφαρμογής ή της τρέχουσας σελίδας, κουμπιά για την επιστροφή του χρήστη στην προηγούμενη σελίδα και κάποια άλλα κουμπιά γενικού σκοπού, τα οποία είναι οι ρυθμίσεις, η βοήθεια, η αναζήτηση κτλ. Το bottom navigation bar είναι ένα widget στο κάτω μέρος του Scaffold το οποίο περιέχει συνήθως μεταξύ τριών και πέντε επιλογών, οι οποίες η κάθε μία επιλέγοντάς την μπορεί να μεταφέρει το χρήστη σε μια συγκεκριμένη σελίδα. Η κάθε επιλογή συνηθίζεται να αποτελείται από ένα εικονίδιο και ορισμένες φορές και από ένα σύντομο τίτλο της σελίδας που γίνεται αναδρομολόγηση. Το bottom sheet εμφανίζεται στο κάτω μέρος της οθόνης σαν ένα popup και ο χρήστης κάνοντας κάποιες συγκεκριμένες ενέργειες προγραμματισμένες από τους δημιουργούς της εφαρμογής, μπορεί να εμφανίσει ή να εξαφανίσει το sheet. Υπάρχουν δύο είδη bottom sheet: το persistent, που συνήθως παρέχει κάποιες πληροφορίες και επιτρέπει ταυτόχρονα στο χρήστη να διαδρά με το υπόλοιπο UI και το modal που χρησιμοποιείται όπως και ένα menu, το υπόλοιπο UI “απενεργοποιείται” και για να διαδράξει ξανά μαζί του ο χρήστης, πρέπει να κλείσει το sheet. Ο drawer που είναι ένα menu στα πλάγια της οθόνης. Στις κινητές συσκευές συνηθίζεται, από προεπιλογή, να μην είναι εμφανίσιμος

και ο χρήστης να πρέπει να πατήσει κάποιο κουμπί ή να κάνει κάποιο gesture για να εμφανιστεί.

8. Δημοφιλή Widget Γενικής Χρήσης

Από εδώ και πέρα το Scaffold μπορεί να πάρει στην παράμετρο body ένα widget, οποιοδήποτε θέλει ο χρήστης, για να μπορέσει να φτιάξει το κυρίως σώμα της σελίδας. Υπάρχουν ήδη χιλιάδες Widget που μπορεί να χρησιμοποιήσει κάποιος που προέρχονται από το ίδιο το framework αλλά και από εξωτερικές βιβλιοθήκες. Αλλά εδώ θα γίνει ανάλυση μόνο ορισμένων, οι οποίες χρησιμοποιήθηκαν στην τρέχουσα εργασία. Ένα από τα πιο πολυχρησιμοποιημένα Widget είναι το text που απλώς εμφανίζει ένα κείμενο στην οθόνη μαζί με κάποιες ρυθμίσεις σχετικά με την μορφοποίηση του κειμένου, το χώρο που επιτρέπεται να πιάνει και άλλα. Άλλα, εξίσου συνηθισμένα Widget είναι τα Row και Column, τα οποία τοποθετούν μια λίστα από Widget στην οθόνη με την μορφή μιας γραμμής ή μίας στήλης. Το Container θέτει ένα Widget εντός ενός τετράγωνου διαφανούς αντικειμένου που μπορεί όμως και να αλλάξει το στυλ του, από τον προγραμματιστή. Το TextStyle και γενικότερα τα proprietary widget που ονομάζονται style μπορούν να θέσουν παραμέτρους μορφοποίησης στα από πάνω(στον “πατέρα”) widget.

Στο dart όταν πρέπει να κάνεις μια διεργασία ασύγχρονα, όπου περιλαμβάνει ένα API call ή ένα request σε μια βάση δεδομένων, τότε ο προτεινόμενος τρόπος υλοποίησης του, αποτελεί το Future. Το Future πρακτικά είναι η αντίστοιχη λειτουργία της dart με τα Promises της javascript. Πρακτικά όταν θέλω να κάνω μια από τις παραπάνω διεργασίες δηλώνω με την λέξη async, μετά από τις παραμέτρους της μεθόδου, ότι η μέθοδος είναι ασύγχρονη. Στη συνέχεια, στην αρχή των εντολών, που επιθυμώ ο κώδικας να σταματάει την εκτέλεση του (ενώ βρίσκεται σε ένα νέο thread) για να περιμένει να εκτελεστεί μια διεργασία, προσθέτω την λέξη await. Πάντα μια τέτοια ασύγχρονη μέθοδος επιστρέφει ένα future, ένα αντικείμενο δηλαδή που περιμένει μια απάντηση για να έχει δεδομένα. Για να μπορέσει κάποιος να πάρει τα δεδομένα, θα πρέπει να “περιμένει” να ολοκληρωθεί η εκτέλεση της μεθόδου. Ο τρόπος που θα φανεί αυτό μέσα στην εργασία είναι με τη χρήση του Widget FutureBuilder το οποίο αναλαμβάνει να περιμένει τα δεδομένα και να τα παρέχει στον χρήστη μαζί με πληροφορίες, σχετικά με την κατάσταση της όλης διαδικασίας. Πιο αναλυτικά, ο FutureBuilder περιέχει δύο απαραίτητα attributes. Το πρώτο είναι το “future”, που δέχεται το future που θα πρέπει να περιμένει να εκτελεστεί. Το δεύτερο είναι το “builder”, που εκεί εντός ενός anonymous function πρέπει να επιστραφεί το Widget που θα εμφανίζει τελικώς ο FutureBuilder. Το σημαντικό εδώ είναι ότι σαν παράμετρος του anonymous function παρέχεται στον χρήστη η μεταβλητή snapshot από την οποία μπορεί να πάρει την τρέχουσα κατάσταση του Future ανανεώνοντας και μεταβάλλοντας το Widget που εμφανίζεται ανάλογα, ενώ ταυτόχρονα μέσα στο snapshot θα συμπεριληφθούν και τα δεδομένα που θα γυρίσει το Future. Μια

τρίτη σημαντική προαιρετική παράμετρος είναι το `initialData`. Όπως υπονοεί και το όνομα εκεί γίνεται αποδεκτό ένα αντικείμενο το οποίο θα χρησιμοποιηθεί στο `snapshot` του `builder` ως `placeholder` μέχρι να ολοκληρωθεί το `Future` και τα δεδομένα του να αντικαταστήσουν τα `initialData`.

9. Provider

Ένα ακόμη πολύ συνηθισμένο `Widget` είναι ο `Provider`. Δουλειά του `Provider` είναι να πάρει ένα αντικείμενο ως παράμετρο και ένα `Widget` ως παιδί και να παρέχει σε όλο το υπο-δέντρο, με τα `Widget` του παιδιού, το αντικείμενο αυτό, μεταφέροντας έτσι δεδομένα εύκολα, μεταξύ ενός συμπλέγματος από προγόνους και απογόνους.

10. Stream

Από την πλευρά της `Dart` τώρα υπάρχει μια λειτουργία, η κλάση `Stream` που παρέχει στον προγραμματιστή τη δυνατότητα να επιστρέψει μια σειρά από ασύγχρονα `Event`. Τα `event` αυτά μπορούν να περιέχουν δεδομένα ή ένα `error` το οποίο να ενημερώνει ότι υπήρξε κάποιο πρόβλημα. Όταν τελειώσει ο ερχομός των δεδομένων και το `session` τερματιστεί, τότε έρχεται ένα `event` το οποίο ενημερώνει σχετικά με αυτήν την εξέλιξη, τερματίζοντας πρακτικά έτσι τη λειτουργία του `Stream`.

11. Stream Provider

Οπότε, με βάση τα δύο παραπάνω, αν τα συνδυάσουμε σε ένα `Widget` έχουμε το `StreamProvider`. Ο `StreamProvider` δέχεται ένα `stream` ως `value`, το οποίο παρέχει πρόσβαση στα δεδομένα αυτά, στο `child` του και στους απογόνους αυτού.

12. Stream Builder

Με την ίδια λογική όπως και με τον `StreamProvider`, ο οποίος συνδυάζει τις λειτουργίες ενός `Stream` με τον `Provider`, αντίστοιχα υπάρχει και ο `StreamBuilder` ο οποίος συνδυάζει τις λειτουργίες ενός `Stream` με έναν `Builder`. Συγκεκριμένα, η πρώτη υποχρεωτική παράμετρος είναι ένα `Stream` στο οποίο θα ανοίξει ένα `session` και τα δεδομένα που θα του έρχονται θα παρέχονται στον `Builder`. Στη συνέχεια, η δεύτερη υποχρεωτική παράμετρος είναι ο `Builder`. Ο `Builder` παίρνει σαν τιμή ένα `anonymous function` και έχει σαν παράμετρο ένα αντικείμενο `Snapshot` από το οποίο ο προγραμματιστής μπορεί να ανακτήσει τα δεδομένα και την κατάσταση του `Stream`. Το `function` πρέπει υποχρεωτικά

στο τέλος, να επιστρέφει ένα Widget, το widget που θα εμφανίζει στην οθόνη ο StreamBuilder.

13. Inkwell

Αν θέλει ο προγραμματιστής να προσθέσει τη δυνατότητα σε ένα Widget να γίνει διαδραστικό μέσο αφής ή κλικ, τότε το διαθέσιμο Widget για αυτήν τη δουλειά είναι το Inkwell. Αυτό παρέχει διάφορες μεθόδους, ανάλογα με το μέσο διάδρασης όπως είναι η παράμετρος onTap όπου εκτελεί μια συνάρτηση, όταν ο χρήστης “πατήσει” στο Inkwell. Ακόμη, η παράμετρος customBorder περιγράφει το σχήμα του Widget. Δίνει ακόμη στο προγραμματιστή να προσθέσει ripple effects στο Widget ως feedback προς το χρήστη με ένα απλό effect να έρχεται ως default.

14. Card Widget

Στο Material Design όταν θέλει ο προγραμματιστής-σχεδιαστής να ξεχωρίσει τις πληροφορίες που αφορούν ένα αντικείμενο με ένα άλλο ενώ πολλά τέτοια διαφορετικά αντικείμενα υπάρχουν στην οθόνη, η προτεινόμενη λύση είναι η χρήση των καρτών (Card). Η κάρτα είναι μια τετραγωνισμένη περιοχή με διαφορετικό Background απ’ ότι αυτό της σελίδας και εντός της, περιέχει όλες τις πληροφορίες-Widget του αντικειμένου. Στο Flutter όλη αυτή η λειτουργία πραγματοποιείται με τη χρήση του Widget Card.

15. Dialog Widget

Πάλι στο Material Design, αν ο προγραμματιστής επιθυμεί να τραβήξει την προσοχή του χρήστη σε μια συγκεκριμένη πληροφορία, η μέθοδος που προτείνεται είναι η χρήση των Dialog. Τα dialog είναι μια popup κάρτα στη μέση της οθόνης ή μπορεί και να καλύπτει όλη τη διαφάνειά της. Η υπόλοιπη σελίδα γύρω από το Dialog, σκοτεινιάζει για να αναδειχθεί το Dialog. Υπάρχουν πολλά είδη Dialog αλλά η περισσότερο συνηθισμένη μορφή έχει ως εξής: στο πάνω μέρος υπάρχει ένα τίτλος με μεγαλύτερη γραμματοσειρά και ακριβώς από κάτω του, υπάρχει το κυρίως σώμα που περιέχει την πληροφορία που θέλει να τονίσει ο προγραμματιστής. Προαιρετικά, στις κάτω γωνίες υπάρχουν κουμπιά για την λήψη μιας απόφασης, από το χρήστη, με βάση τη γνώση από την παραπάνω πληροφορία. Στο Flutter, το πιο κλασικό παράδειγμα Dialog είναι το SimpleDialog, το οποίο παρέχει μόνο τον τίτλο και το κυρίως σώμα, μέσω των παραμέτρων title και children αντίστοιχα.

16. Controllers

Όταν πρέπει να ανακτήσει ο προγραμματιστής δεδομένα, από input του χρήστη σε ένα Widget, τότε πρέπει να χρησιμοποιήσει έναν controller. (stack overflow, 2018) Ο controller είναι ένα αντικείμενο που δουλειά του είναι ελέγχει δεδομένα-τιμές στο Widget που προσδένεται. Για παράδειγμα μέσω ενός TextEditingController ο προγραμματιστής μπορεί να ανακτήσει πληροφορίες, όπως κείμενο που περιέχει ένα TextField (widget που δέχεται κείμενο από τον χρήστη) ή σε ένα άλλο Widget που υποστηρίζει scroll μπορεί να βάλει ένα ScrollController και να αλλάζει την τρέχουσα θέση του Scroll προγραμματιστικά ή να μπορεί να διαβάσει την τρέχουσα θέση scroll του χρήστη κτλ..

17. Form Widget

Πολλές φορές ο προγραμματιστής πρέπει να φτιάξει μια φόρμα που θα δέχεται πολλαπλά δεδομένα από το χρήστη. Ένα βολικό Widget στην προκειμένη περίπτωση είναι το Form. (Flutter, 2022) Το Form το οποίο παίρνει σαν child ένα widget ή ένα σύνολο από widget από τα οποία θα παίρνει τα δεδομένα του χρήστη. Ένα παράδειγμα τέτοιου Widget είναι το TextFormField το οποίο δέχεται input κειμένου από το χρήστη, αλλά το ιδιαίτερο χαρακτηριστικό του, όντας ένα Widget για Form, είναι ότι δέχεται ένα anonymous function στην παράμετρο validator. Δουλειά του validator είναι να περνά από ελέγχους την τιμή που έδωσε ο χρήστης και να εμφανίζει μήνυμα λάθους, αν έχει προκύψει. Τώρα, ο λόγος που χρειάζεται το Form Widget είναι, επειδή μέσω ενός GlobalKey που είναι attach στο Form ο προγραμματιστής μπορεί να εκτελέσει όλα τα validator των απογόνων του, περνώντας τους ελέγχους μαζικά, την κατάλληλη στιγμή. Ένα GlobalKey αποθηκεύει πληροφορίες για το Widget στο οποίο γίνεται attach.

18. Checkbox

Μερικές φορές ο προγραμματιστής θέλει να δώσει την επιλογή στο χρήστη, να μπορεί να διαλέγει ένα ή πολλαπλά αντικείμενα για να πραγματοποιήσει μια ενέργεια. Σε μια τέτοια περίπτωση η λύση, σύμφωνα με το Material Design, είναι το Checkbox. Αυτό, αποτελείται από ένα τετράγωνο που αν πατήσει πάνω του ο χρήστης και το “διαλέξει” εμφανίζεται ένα “τικ” στη μέση του για να δείξει ακριβώς το αποτέλεσμα αυτής της ενέργειας. Αν ο χρήστης το ξαναπατήσει το “τικ” εξαφανίζεται γυρνώντας στην αρχική του κατάσταση. Το Widget Checkbox έχει την παράμετρο value στην οποία παίρνει την κατάσταση του checkbox ως ένα boolean. Αν η τιμή είναι true, τότε το Checkbox είναι επιλεγμένο ενώ αν η τιμή είναι false το Checkbox δεν έχει επιλεγεί. Προαιρετικά, υπάρχει και η παράμετρος tristate η οποία από default είναι false και σε αυτή την περίπτωση δεν

έχει διαφορά, αλλά αν τεθεί ως true τότε το checkbox μπορεί να δεχτεί και μια τρίτη τιμή, την null. Σε αυτή την περίπτωση εμφανίζεται μια οριζόντια γραμμή (μείον) στο κέντρο του checkbox. Τέλος, υπάρχει και η onChange η οποία δέχεται μια anonymous function με παράμετρο την τρέχουσα τιμή του checkbox και η οποία εκτελείται μετά από μια αλλαγή στην τιμή της παραμέτρου value.

Το Checkbox Widget σχεδιάστηκε έτσι ώστε να μπορούν οι προγραμματιστές να δημιουργήσουν πρωτότυπα νέα widget με αυτό, αλλά σπάνια χρησιμοποιείται με αυτόν τον τρόπο μόνο του, καθώς είναι δύσκολο για τον χρήστη να αντιληφθεί το τι ακριβώς διαλέγει. Ένα έτοιμο Widget που έρχεται με το Flutter είναι το CheckboxListTile το οποίο συνδυάζει το checkbox με το ListTile. Εν συντομία, το ListTile είναι ένα Widget που επεκτείνεται οριζόντια και σκοπό έχει να παρέχει πληροφορίες στο χρήστη από τρία διαφορετικά σημεία. Το title, που βρίσκεται προς την αριστερή πλευρά του Widget και περιέχει την κυρίως πληροφορία. Ακριβώς κολλητά, στα αριστερά του, βρίσκεται το leading το οποίο συνήθως περιέχει μια πληροφορία κατηγοριοποίησης ή απαρίθμησης. Τέλος, στα δεξιά υπάρχει το trail το οποίο περιέχει μια δευτερεύουσα ή συμπληρωματική πληροφορία. Στην περίπτωση του CheckboxListTile τώρα, στο trail προστίθεται και ένα checkbox αυτόματα.

19. Dropdown

Ένα ακόμα Widget, σχετικό με input του χρήστη, είναι το DropdownButton. Αυτό, στην αρχική μορφή του, δείχνει τον τίτλο του επιλεγμένου αντικειμένου από ένα σύνολο αντικειμένων μαζί με ένα εικονίδιο, το οποίο παραπέμπει το χρήστη να πατήσει επάνω. Μόλις πατήσει το εικονίδιο, εμφανίζεται μια popup λίστα με όλα τα αντικείμενα που μπορεί να επιλέξει ο χρήστης και πατώντας σε κάποιο, κλείνει η λίστα και ο τίτλος του Dropdown ανανεώνεται με το επιλεγμένο αντικείμενο. Υποχρεωτικά, το Widget περιέχει μια λίστα με Widget τύπου DropdownMenuItem. Αυτά με την σειρά τους, περιέχουν ένα widget ως παιδί, το οποίο θα εμφανίζεται στη λίστα και στο dropdown ως “τίτλος”, καθώς και μια τιμή value, όπου είναι η τιμή αυτού του αντικειμένου για να μπορεί να ξεχωρίσει γρήγορα το επιλεγμένο αντικείμενο. Πίσω το DropdownButton, περιέχει ακόμα μια παράμετρο value, όπου είναι το value του τρέχοντος επιλεγμένου DropdownMenuItem. Τέλος, υπάρχει η παράμετρος με την ανώνυμη συνάρτηση onChange η οποία καλείται στην αλλαγή της τιμής του value.

Σχεδίαση

Η γενική ιδέα της εργασίας είναι να δημιουργηθεί μια εφαρμογή όπου θα μπορεί, με διαδραστικό τρόπο να δείχνει στο χρήστη πληροφορίες για το σημείο στο οποίο βρίσκεται. Το βασικό target group, στο οποίο θα στοχεύσει η εφαρμογή, είναι οι τουρίστες που επισκέπτονται την Ελλάδα και θέλουν να ξεναγηθούν μέσα στην πόλη της Αθήνας. Οπότε, μέσα από την εφαρμογή θα πρέπει να παρέχονται και οδηγίες, ώστε οι χρήστες να μπορέσουν εύκολα να προσανατολιστούν, μέσα στην πόλη, προς τα κοντινά σημεία ενδιαφέροντος. Λαμβάνοντας υπόψη τα παραπάνω, επιλέχθηκαν οι κάτωθι τεχνολογίες, που θεωρήθηκαν οι πιο κατάλληλες για την ανάπτυξη της συγκεκριμένης εφαρμογής.

Επιλογή Framework

Η εφαρμογή, αναπτύχθηκε στο λειτουργικό σύστημα android, καθώς για τη χρήση, που έγινε αναφορά και πιο πάνω, απαιτείται μια φορητή συσκευή όπου ο χρήστης είναι εύκολο να την κουβαλάει πάνω του κατά την διάρκεια ταξιδιού. Η καταλληλότερη συσκευή για να αναπτυχθεί μια εφαρμογή πλοήγησης είναι τα smartphones, όχι μόνο επειδή καλύπτουν όλες τις προϋποθέσεις που έχουν τεθεί στην προηγούμενη παράγραφο αλλά και λόγω του ότι είναι η δημοφιλέστερη συσκευή, και η πλειοψηφία των ανθρώπων, σε παγκόσμιο επίπεδο, τη χρησιμοποιούν. Στη συνέχεια, ο προγραμματιστής καλείται να επιλέξει ανάμεσα σε δύο λειτουργικά συστήματα, το Android και το iOS. Σε αυτή την περίπτωση, επιλέχθηκε η ανάπτυξη της εφαρμογής σε ένα περιβάλλον android, για δύο πολύ σημαντικούς λόγους. Πρώτον, οι συσκευές με λειτουργικό android είναι περισσότερες από αυτές με iOS και δεύτερον τα development environments στο android είναι διαθέσιμα σε windows και linux, σε αντίθεση με το ios που απαιτείται υπολογιστής mac για την ανάπτυξη μιας εφαρμογής. Υπάρχουν διάφοροι τρόποι και frameworks για τη δημιουργία μιας εφαρμογής android. Από την μια υπάρχουν frameworks όπως το Flutter και το React Native, όπου μπορεί κάποιος γράφοντας μια φορά κώδικα να φτιάξει την ίδια εφαρμογή και για android και για ios. Ωστόσο, κανένα από αυτά δεν είναι τόσο αποδοτικό όσο ο native προγραμματισμός, και επειδή το AR απαιτεί αρκετούς πόρους από το σύστημα, τα παραπάνω frameworks αποκλείστηκαν για να μπορεί η εφαρμογή να είναι διαθέσιμη σε όσες περισσότερες συσκευές γίνεται (ακόμα και σε πιο παλιές ή με μικρότερη επεξεργαστική ισχύ). Οι “native” επιλογές, κατά τις οποίες γίνεται απευθείας χρήση του android framework κάνουν χρήση των γλωσσών Java ή Kotlin. Και οι δύο επιλογές

προσφέρουν λίγο-πολύ τις ίδιες δυνατότητες, με τις μόνες σημαντικές διαφορές να είναι ότι, στην Kotlin υπάρχουν κάποιες πολύ καινούργιες βιβλιοθήκες, που κάνουν πιο τυπικές κάποιες μεθόδους ανάπτυξης της εφαρμογής, που στο παρελθόν υπήρξαν πολλοί τρόποι για να επιτευχθούν αλλά τώρα με τη χρήση αυτών των βιβλιοθηκών μπορούν ευκολότερα και γρηγορότερα να γραφτούν οι προτεινόμενες λύσεις από τους δημιουργούς του λειτουργικού συστήματος. Επίσης, η Kotlin τείνει να απαιτεί από τον προγραμματιστή να γράψει λιγότερες γραμμές κώδικα από την Java για να επιτύχει το ίδιο πράγμα.

Σχεδίαση Android

Αφού επιλέχθηκε η συσκευή στην οποία θα τρέχει η εφαρμογή και η γλώσσα στην οποία θα γραφτεί, ήρθε η στιγμή να αποφασιστεί μια γενική σχεδίαση της εφαρμογής με βάση τις ανάγκες του χρήστη, στην προκειμένη περίπτωση, καθώς και οι τεχνολογίες που θα χρειαστούν για την ανάπτυξη κώδικα που θα ικανοποιεί αυτές τις ανάγκες.

1. AR βιβλιοθήκες

Στην αρχή, ο χρήστης θα κάνει διάδραση με την εφαρμογή, η οποία θα του δείχνει τρισδιάστατα ψηφιακά αντικείμενα στον χώρο που βρίσκεται με την βοήθεια της κάμερας του κινητού που χρησιμοποιεί. Αυτή η λειτουργία στο android επιτυγχάνεται με τη χρήση δύο βιβλιοθηκών. Η πρώτη είναι η AR Core και η δεύτερη η Sceneform. Η AR Core αφού ο προγραμματιστής, την αρχικοποιήσει και την “καλέσει” στον κώδικά του, αυτόματα θα ανοίξει την κάμερα και θα σκανάρει τον περιβάλλοντα χώρο, με στόχο να εντοπίσει τις επιφάνειες που υπάρχουν. Στη συνέχεια, υποδεικνύει στον χρήστη τις επιφάνειες που εντόπισε, προσθέτοντας στη ζωντανή μετάδοση της εικόνας, που βλέπει ο χρήστης, κουκίδες σε όλη την επιφάνεια. Αν πατήσει πάνω σε μια από αυτές, τότε δημιουργείται μια άγκυρα, στο σημείο αυτό. Μετά, πάνω σε αυτή την άγκυρα μπορεί να κάνει κανείς attach τρισδιάστατα αντικείμενα. Κανονικά αυτά τα αντικείμενα πρέπει να δημιουργηθούν με τη χρήση του OpenGL, αλλά επειδή αυτό είναι περιττό και χρονοβόρο, σε πιο μικρές και απλές εφαρμογές υπάρχει μια βιβλιοθήκη, η Sceneform, η οποία αναλαμβάνει να κάνει το rendering του αντικειμένου, από ένα αρχείο στο οποίο είναι αποθηκευμένο, και ακόμη επιτρέπει ορισμένα “βασικά” animations.

2. GPS - Χάρτες

Για να μπορεί η εφαρμογή να αναγνωρίσει το μέρος στο οποίο βρίσκεται ο χρήστης, ώστε να γίνεται χρήση των αντικειμένων που αντιστοιχούν σε αυτό το μέρος, γίνεται χρήση του GPS του κινητού, το οποίο παρέχει τις συντεταγμένες του χρήστη.

Καθώς η εφαρμογή προορίζεται για τουρίστες, οι οποίοι το πιθανότερο είναι να μην ξέρουν να κατευθυνθούν προς τα σημεία ενδιαφέροντος, πρέπει να παρέχεται από την εφαρμογή η ικανότητα πλοήγησης του χρήστη. Αυτό υλοποιείται με τη χρήση του Google Maps API. Αυτή η βιβλιοθήκη επιτρέπει στους προγραμματιστές να φτιάξουν ένα χάρτη μέσα στην εφαρμογή τους με μεγάλη ευκολία. Απλώς, ο προγραμματιστής φτιάχνει τον χάρτη στον XML καλώντας το Map Fragment που παρέχεται από την βιβλιοθήκη και το παραμετροποιεί από τον Java κώδικα για να βάλει τα στοιχεία των σημείων ενδιαφέροντος. Επιπλέον, πρέπει να μπορεί ο χρήστης να βλέπει το δρόμο τον οποίο πρέπει να ακολουθήσει για να φτάσει στο επιλεγμένο σημείο ενδιαφέροντος. Για το λόγο αυτό, χρησιμοποιείται το Directions API, μια επιπλέον βιβλιοθήκη από αυτή των χαρτών, η οποία δίνοντάς της τις συντεταγμένες, πού αρχίζει και τελειώνει η διαδρομή, αντιστοίχα τις ανεβάζει στο cloud και επιστρέφει στο χρήστη μια λίστα με πιθανές διαδρομές, τις οποίες μπορεί να ακολουθήσει. Τέλος, ο χρήστης θα πρέπει να μπορεί να επιλέξει τη διαδρομή που θα ακολουθήσει, έτσι θα παρέχεται μια λίστα την οποία διάλεξε ο ίδιος.

3. Cloud Server

Οι διαδρομές αυτές για να είναι πιο δυναμικές και να μην χρειάζεται αναβάθμιση κάθε φορά που ο διαχειριστής της εφαρμογής θέλει να προσθέσει, αφαιρέσει ή επεξεργαστεί μια από αυτές, το πρόβλημα επιλύθηκε με τη χρήση ενός cloud service, το οποίο θα χρησιμοποιείται σαν βάση δεδομένων για τις απαραίτητες πληροφορίες. Τέτοιες υπηρεσίες προσφέρουν πολλές βάσεις δεδομένων, μερικές από τις πιο δημοφιλείς είναι οι AWS, Azure και αυτή που θα χρησιμοποιηθεί στην παρούσα εργασία είναι το Google Cloud. Πιο συγκεκριμένα θα χρησιμοποιηθεί το Google Firebase, το οποίο βασίζει τις υπηρεσίες του πάνω στο Google Cloud. Οποιαδήποτε από τις προαναφερόμενες υπηρεσίες θα μπορούσε να ικανοποιήσει πλήρως τις ανάγκες της εφαρμογής, αλλά το Google Firebase, το οποίο είναι σχεδιασμένο για να λειτουργεί με client εφαρμογές σε κινητά και σε ιστοσελίδες είναι πιο απλό στη χρήση του και ο προγραμματιστής μπορεί να το ενσωματώσει γρηγορότερα στον κώδικά του.

Πιο αναλυτικά, οι υπηρεσίες που θα χρησιμοποιηθούν είναι οι Firestore Database, Storage, Hosting και Authentication. Η Firestore Database είναι μια NoSQL βάση δεδομένων που αποθηκεύει τα δεδομένα σαν dictionaries. Αυτή θα χρησιμοποιηθεί για την αποθήκευση των δεδομένων που χρειάζονται οι διαδρομές και τα σημεία ενδιαφέροντος. Το Storage είναι μια cloud βάση για αποθήκευση αρχείων, πρακτικά αντικαθιστά ένα FTP server. Αυτό χρειάζεται για την αποθήκευση των αρχείων που περιέχουν τα τρισδιάστατα αντικείμενα, δηλαδή από εδώ η εφαρμογή θα κατεβάζει τα αρχεία για να τα “δείξει” στον χρήστη. Ακόμη, τα Hosting και Authentication θα χρησιμοποιηθούν για την ιστοσελίδα που θα αναφερθεί στην συνέχεια. Το Hosting θα κάνει διαθέσιμη την σελίδα στο διαδίκτυο ενώ ταυτόχρονα, επειδή βρίσκεται στο cloud θα

μπορεί η ιστοσελίδα να είναι συνέχεια διαθέσιμη χωρίς να απαιτείται ιδιαίτερη συντήρηση από τον διαχειριστή. Τέλος, το Authentication, χρησιμοποιήθηκε για την αυθεντικοποίηση του διαχειριστή της ιστοσελίδας καθώς όπως θα γίνει αναφορά και στην συνέχεια, μέσω αυτής, θα μπορεί ο διαχειριστής να επεξεργαστεί τα δεδομένα στη βάση δεδομένων.

4. Σελίδα για Admin

Οπότε, όπως προαναφέρθηκε και παραπάνω, η εφαρμογή θα χρησιμοποιήσει μια ιστοσελίδα με σκοπό να μπορεί να συνδεθεί ο διαχειριστής της εφαρμογής και να αλλάξει, προσθέσει ή αφαιρέσει από το περιεχόμενό της. Υπάρχουν πολλοί τρόποι και frameworks για να φτιαχτεί μια ιστοσελίδα. React, Angular, Vue είναι μερικές από τις πολλές επιλογές. Ακόμη, μπορεί κάποιος να το φτιάξει με τον “παραδοσιακό τρόπο” χρησιμοποιώντας javascript, html και css. Στην προκειμένη περίπτωση ωστόσο, επιλέχθηκε το Flutter αφού παρέχει την δυνατότητα της πολύ γρήγορης δημιουργίας της ιστοσελίδας, σε σύγκριση με τους παραπάνω τρόπους. Η επεξεργασία αυτή των δεδομένων θα γίνεται με την βιβλιοθήκη που παρέχεται από το Firebase για την πρόσβαση στις προαναφερθείσες υπηρεσίες της, από μια εφαρμογή που χρησιμοποιεί το Flutter και το Dart. Το Dart είναι η γλώσσα προγραμματισμού που χρησιμοποιεί το Flutter και είναι σχεδιασμένη για τη δημιουργία client εφαρμογών στο διαδίκτυο και σε κινητές συσκευές. Σαν γλώσσα μοιάζει αρκετά με την javascript αλλά έχει και τις δικές της ιδιαιτερότητες και λειτουργίες.

Περιορισμοί και Δυσκολίες

Στην συγκεκριμένη εργασία τα περισσότερα προβλήματα είχαν να κάνουν με τεχνικούς περιορισμούς στον κώδικα και στα εργαλεία που χρησιμοποιήθηκαν, όπως και συνηθίζεται αλώςτε και με την ανάπτυξη του λογισμικού. Άλλα, ίσως το πιο σημαντικό και χρονοβόρο πρόβλημα ήταν η έλλειψη παρόμοιων λογισμικών, ιδίως σε ερευνητικό επίπεδο αφού τα περισσότερα συστήματα αφορούσαν εμπορικές εφαρμογές. Ενώ σαν ξεχωριστά features (AR, χάρτες κτλ.) υπάρχει μια μεγάλη πληθώρα με πληροφορίες και με παραδείγματα τόσο σε εμπορικό όσο και σε ερευνητικό επίπεδο, ο συνδυασμός τους βρίσκεται σε σχετικά μικρό αριθμό πόρων. Ακόμα και στις περιπτώσεις που εμφανίζονται όλα ή τα περισσότερα από αυτά, συνήθως αποτελούν μέρος μιας πολύ εξατομικευμένης εφαρμογής που μπορεί να χρησιμοποιηθεί για ελάχιστα προκαθορισμένα πράγματα.

Κατάργηση της Sceneform

Από προγραμματιστική σκοπιά το μεγαλύτερο πρόβλημα που έπρεπε να αντιμετωπιστεί, έχει να κάνει με τη βιβλιοθήκη του Sceneform. Λίγο καιρό μετά την ολοκλήρωση την πρώτης έκδοσης της εφαρμογής Android, η ομάδα που συντηρούσε το ARCore και το Sceneform άλλαξε, με αποτέλεσμα η νέα ομάδα που ανέλαβε τα project να σταματήσει την ανάπτυξη και την συντήρηση του Sceneform. Ο λόγος ήταν ότι τα τελευταία χρόνια η ανάπτυξη λογισμικού AR είχε αρχίσει να γίνεται με τη χρήση Game Engines όπως το Unity3D και το Unreal Engine, με αποτέλεσμα η “native” ανάπτυξη AR λογισμικού να γίνει λιγότερο δημοφιλής και κατά συνέπεια να θεωρηθεί ότι πλέον οι βιβλιοθήκες που χρησιμοποιούνται αποκλειστικά στις native εφαρμογές να μην “αξίζουν” τα έξοδα συντήρησης τους. Αυτό δημιούργησε μια σειρά προβλημάτων που έπρεπε να επιλυθούν για να μπορέσει να λειτουργήσει η εφαρμογή, χωρίς να χρειαστεί όμως να ξαναγραφτεί σε κάποιο άλλο framework. Να αναφερθεί ακόμη, ότι η χρήση κάποιας άλλης βιβλιοθήκης για την αντικατάσταση της Sceneform ήταν αδύνατη αφού δεν υπήρχε κάποια άλλη διαθέσιμη την περίοδο που φτιάχτηκε η εφαρμογή.

Το πρώτο πρόβλημα που έπρεπε να επιλυθεί ήταν τα dependencies της εφαρμογής. Συγκεκριμένα, έπρεπε να δοκιμαστούν ποιες εκδόσεις των υπόλοιπων βιβλιοθηκών μπορούν να χρησιμοποιηθούν. Αυτό ήταν απαραίτητο αφού οι περισσότερες βιβλιοθήκες εξαρτώνται από την χρήση άλλων βιβλιοθηκών και τελικώς όλες πρέπει να υποστηρίζονται από την έκδοση του Gradle και από την έκδοση του Android την οποία

χρησιμοποιούμε ως Target (δηλαδή τη βασική έκδοση που έχει επιλεχτεί, για να μπορεί να την υποστηρίξει η εφαρμογή). Αυτό λύθηκε χρησιμοποιώντας αρχικά, όλες τις βιβλιοθήκες, στην πιο πρόσφατη έκδοση τους και αν κάποια, για οποιοδήποτε λόγο επέστρεφε error, χρησιμοποιούταν μια παλαιότερη έκδοση της, μέχρι να βρεθεί κάποια, που υποστηρίζεται χωρίς errors. Οι περισσότερες βιβλιοθήκες μπορούσαν να χρησιμοποιήσουν την τότε, πιο πρόσφατη έκδοση τους αλλά ήταν σημαντικό να βρεθούν, ποιες βιβλιοθήκες επηρεάζονται περισσότερο, από το πρόβλημα αυτό. Εξάλλου, η αναβάθμιση των περισσότερων βιβλιοθηκών, αργότερα, σε αυτό το σημείο έμοιαζε δύσκολη έως αδύνατη. Οπότε έπρεπε οι εκδόσεις που επιλέχθηκαν να είναι αξιόπιστες, για να διασφαλίσουν την μακροβιότητα της εφαρμογής.

Το δεύτερο μεγάλο πρόβλημα που έπρεπε να αντιμετωπιστεί, είχε να κάνει με τον μετατροπέα της Sceneform. Πιο αναλυτικά, η Sceneform χρησιμοποιούσε έναν plugin του Android Studio για να μετατρέψει αρχεία με τρισδιάστατα αντικείμενα, στην μορφή που υποστηρίζει. Συγκεκριμένα, το εργαλείο μπορούσε να πάρει αρχεία τύπου *.fbx, *.obj, *.gltf και *.glb και να τα μετατρέψει σε *.sfa (Sceneform Asset Definition) και *.sfb (Sceneform Binary asset) αρχεία τα οποία μπορούσε να τα διαβάσει μετά η βιβλιοθήκη για να τα κάνει Render στην οθόνη του χρήστη. Από τα αρχεία εισόδου που αναφέρθηκαν, αυτή την εφαρμογή την ενδιαφέρει μόνο το obj format αφού είναι το πιο δημοφιλές format αποθήκευσης τρισδιάστατων αντικειμένων και υπάρχει μια μεγάλη πληθώρα τέτοιων αντικειμένων στο διαδίκτυο, αλλά και τα περισσότερα προγράμματα Sculpting όπως το Blender και το Autodesk 3DS Max μπορούν να παράξουν αντικείμενα χρησιμοποιώντας αυτό το format. Τα sfa και sfb αρχεία είναι proprietary formats του Sceneform και για αυτό είναι δύσκολο να βρεθούν διαθέσιμα ή να δημιουργηθούν απευθείας από κάποιο πρόγραμμα. Θα πρέπει να μετατραπούν από κάποιο άλλο format, για αυτό άλλωστε και απαιτείται και ο μετατροπέας - plugin του Android Studio. Δύο προβλήματα προέκυψαν με τη χρήση του μετατροπέα απευθείας από το Android Studio. Το πρώτο, ήταν ότι αφού σταμάτησε η συντήρηση της βιβλιοθήκης, ταυτόχρονα σταμάτησε και η συντήρηση του plugin. Το αποτέλεσμα ήταν ότι μετά από μερικές ενημερώσεις του Android Studio κάποια από τα dependencies του plugin από το Android Studio άλλαξαν με αποτέλεσμα κατά την μετατροπή ενός αρχείου να επιστρέφεται ένα error και στη συνέχεια η μετατροπή να αποτυγχάνει. Αυτό, σε πρώτο στάδιο επιλύθηκε με τη χρήση μιας έκδοσης του Android Studio, η οποία ακόμη μπορούσε να χρησιμοποιήσει το plugin χωρίς προβλήματα για να μπορούν τουλάχιστον να συνεχίσουν να πραγματοποιούνται οι μετατροπές. Στην αρχή αυτή η λύση μπόρεσε να συνεχίσει την λειτουργία του workflow της εφαρμογής. Σύντομα όμως, ολοένα και περισσότερα αρχεία άρχισαν να πετάνε error κατά την μετατροπή. Σε αυτό το σημείο ήταν εμφανές ότι κάποια άλλη λύση ήταν απαραίτητη. Η αντικατάσταση του plugin αποδείχθηκε αρκετά δύσκολη. Καθώς η βιβλιοθήκη θεωρούταν πλέον deprecated και δεν υπήρχε και από πριν, κάποιο άλλο εργαλείο για αυτήν τη δουλειά, ήταν δύσκολο να βρεθεί κάποια εναλλακτική, ειδικά αν προστεθεί στην εξίσωση το ότι το plugin και το format αποτελούσαν και proprietary λογισμικό, οπότε δεν υπήρχαν πολλές

πληροφορίες διαθέσιμες για το πώς ακριβώς λειτουργούν (τί μετατροπές κάνουν από πίσω). Οπότε, εναλλακτικό εργαλείο δεν βρισκόταν και να δημιουργηθεί νέο ήταν πολύ δύσκολο και απαιτούσε τεράστιο χρονικό διάστημα. Τελικά, μετά από πολύ μεγάλη έρευνα σε διάφορες πηγές βρέθηκε μια λύση. Χάρη στον κύριο Guilherme D'Amoreira βρέθηκε ένα command line εργαλείο στην προσωπική του σελίδα στο Github το οποίο έχει αναπτύξει ο ίδιος και από πίσω χρησιμοποιεί τα binaries της τελευταίας έκδοσης του plugin για να κάνει τις μετατροπές. Το εργαλείο λειτουργεί τελείως αυτόνομα από το Android Studio. Το εργαλείο παρέχει τρία διαφορετικά scripts ανάλογα με το λειτουργικό σύστημα (Windows, Linux, MacOS) που χρησιμοποιεί ο προγραμματιστής-χρήστης του. Το εργαλείο χρησιμοποιεί τρία arguments. Το πρώτο το ονομάζει "material" και στην πραγματικότητα αποτελεί την εντολή όπου ο χρήστης διαλέγει το είδος του format αρχείου που θα είναι το αρχείο εισόδου της μετατροπής (στην περίπτωση αυτής της εφαρμογής αποτελεί το obj format όπως προαναφέρθηκε). Το δεύτερο είναι ο φάκελος στον οποίο αποθηκεύονται τα αρχεία εξόδου της μετατροπής (τα sfa και sfb αρχεία δηλαδή). Και το τελευταίο argument, αποτελεί το path για το αρχείο εισόδου που θα μετατραπεί. Αυτό το εργαλείο εμφάνισε πολύ μεγάλο ποσοστό επιτυχίας στις μετατροπές. Οπότε πλέον σε αυτό το σημείο η έρευνα είχε τελειώσει και το εργαλείο που έλειπε και απαιτούνταν για την διασφάλιση της μακρόβιας λειτουργίας της εφαρμογής είχε πλέον βρεθεί.

Προϋποθέσεις ARCore

Ένας ακόμη περιορισμός που υπήρχε, είχε να κάνει με τις προϋποθέσεις του συστήματος που απαιτεί το ARCore για να τρέξει σε μια συσκευή. Το ARCore μπορεί να τρέξει μόνο σε επιλεγμένα μέρη υλισμικού. Γενικά, το SoC (System on Chip), η οθόνη, και η έκδοση του OpenGL της εφαρμογής, είναι μερικές από τις προϋποθέσεις που πρέπει να υποστηρίζονται για να λειτουργήσει. Το ARCore απαιτεί ακόμη, η συσκευή να βρίσκεται στην έκδοση Android 7.0 (API 24) ή άνω. Αν η συσκευή υποστηρίζεται, τότε βρίσκεται εντός μιας [λίστας](#) που αναγράφει όλες τις συσκευές που μπορούν να το χρησιμοποιήσουν μαζί με τις λειτουργίες που μπορούν να υποστηρίξουν. Τέλος, απαιτείται να είναι εγκατεστημένη η εφαρμογή "Google Play Services for AR" η οποία περιέχει όλα τα εργαλεία που χρησιμοποιεί το ARCore.

Μεθοδολογία

Η παρούσα εργασία αποτελείται από τρία διαφορετικά μέρη. Πιο συγκεκριμένα, το πρώτο μέρος απαρτίζεται από την εφαρμογή android που είναι γραμμένη σε java, το δεύτερο από το backend server στον οποίο χρησιμοποιείται το Firebase σαν Cloud Server και τέλος από μια ιστοσελίδα στην οποία ο διαχειριστής μπορεί να ελέγχει τις διαδρομές.

Εφαρμογή Android

Έχοντας ως αφετηρία την ανάπτυξη της εφαρμογής, χρησιμοποιήθηκε ένας συνδυασμός java με XML για την ανάπτυξη τόσο της λογικής όσο και της διεπαφής με τον χρήστη αντίστοιχα. Το Firebase SDK χρησιμοποιήθηκε για να παρέχεται δυναμική πρόσβαση στα δεδομένα που θα βλέπει ο τελικός χρήστης. Ακόμα, χρησιμοποιείται το Maps API το οποίο μπορεί να παρουσιάσει στο χρήστη ένα χάρτη με τα σημεία ενδιαφέροντος. Τέλος, γίνεται χρήση των βιβλιοθηκών AR Core και Sceneform που είναι υπεύθυνες για την δημιουργία αγκυρών στο χώρο πάνω στα οποία θα γίνονται render τα τρισδιάστατα αντικείμενα.

Τελικός, η εφαρμογή θα φαίνεται στον χρήστη όπως στις παρακάτω εικόνες.

1. Αρχικοποίηση περιβάλλοντος

Πιο αναλυτικά, ξεκινώντας η εφαρμογή ζητά από τον χρήστη permissions, σε περίπτωση που ο χρήστης δεν τα έχει αποδεχτεί προηγουμένως. Στη συνέχεια, γίνεται αρχικοποίηση των διάφορων υπηρεσιών του Firebase που χρησιμοποιούνται, καθώς γίνεται και σύνδεση του χρήστη στην υπηρεσία Authentication σαν "ανώνυμος χρήστης". Μετά από αυτά τα βήματα, δημιουργείται ο αντάπτορας της αρχικής σελίδας, ο οποίος είναι υπεύθυνος για να εμφανίζει το σωστό tab, και ένας listener, που αλλάζει τα tabs όταν ο χρήστης επιλέξει κάποιο άλλο. Βασική προεπιλογή της εφαρμογής είναι ο χρήστης να βλέπει το LiveView, το οποίο με την χρήση της κάμερας δείχνει σε αυτόν που την χρησιμοποιεί, το περιβάλλοντα χώρο, επαυξημένο με τα αντίστοιχα ψηφιακά αντικείμενα, στην περίπτωση που βρίσκεται σε κοντινή απόσταση από ένα σημείο ενδιαφέροντος.

2. Βασικές σελίδες

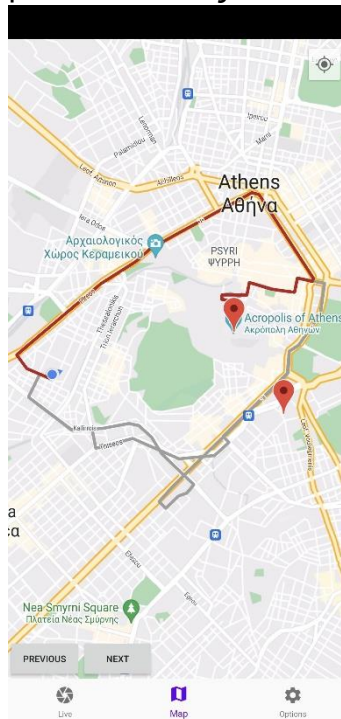
Η εφαρμογή χρησιμοποιεί τρία tabs για να δείξει στο χρήστη το περιεχόμενο της. Το πρώτο που αναφέρθηκε και παραπάνω είναι το LiveView. Σαν διεπαφή, αποτελείται από ένα fragment του sceneform, το οποίο δείχνει την κάμερα με τα ψηφιακά αντικείμενα και από ένα TextView που αναγράφει αν βρίσκεται ο χρήστης κοντά σε κάποιο σημείο ενδιαφέροντος και αν ναι, σε ποιο. Με την επιστροφή στο java κώδικα, γίνεται μια αρχικοποίηση για να μπορεί ο χρήστης να πάρει από το firebase τα δεδομένα από την βάση δεδομένων και τα αρχεία των τρισδιάστατων αντικειμένων από το Storage.

Το δεύτερο tab, είναι το Map. Εκεί, εμφανίζεται ένας χάρτης με τα σημεία ενδιαφέροντος της επιλεγμένης διαδρομής. Η διεπαφή αποτελείται από ένα fragment του google maps sdk το οποίο θα εμφανίσει το χάρτη στο χρήστη. Επίσης, στην κάτω αριστερή γωνία υπάρχουν δύο κουμπιά, τα οποία μπορεί να χρησιμοποιήσει κάποιος για να αλλάξει το σημείο ενδιαφέροντος που είναι επιλεγμένο.

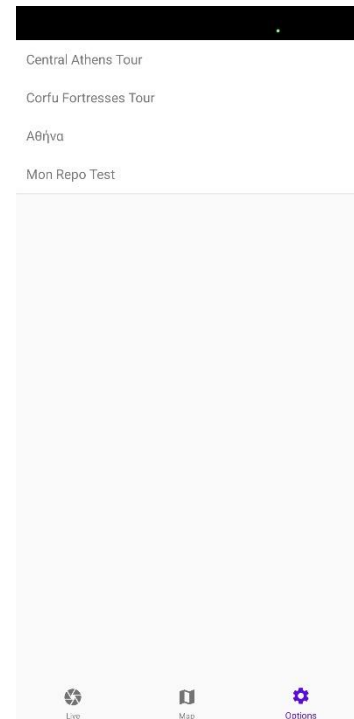
Τα τρία tab φαίνονται στις παρακάτω εικόνες.



Εικόνα 321: Live View



Εικόνα 312: Maps



Εικόνα 33: Menu

3. Thread για συνεχή εντοπισμό του χρήστη

Στην Java, αρχικά υπάρχει ένα Runnable αντικείμενο το οποίο προορίζεται για να τρέχει σε διαφορετικό thread και να ελέγχει αν ο χρήστης βρίσκεται κοντά σε κάποιο σημείο ενδιαφέροντος. Πιο συγκεκριμένα, αρχικά χρησιμοποιείται η μέθοδος

getLocation η οποία βρίσκει και αποθηκεύει την τρέχουσα τοποθεσία του χρήστη σε συντεταγμένες. Στη συνέχεια, υπολογίζεται η απόσταση κάθε σημείου στην επιλεγμένη διαδρομή από τις συντεταγμένες του χρήστη. Ο υπολογισμός γίνεται χρησιμοποιώντας το Πυθαγόρειο θεώρημα. Αν η απόσταση είναι μικρότερη από το AOE (Area of Effect) του σημείου ενδιαφέροντος, τότε ο κώδικας κατεβάζει από την βάση δεδομένων τα δεδομένα που χρειάζονται για τα τρισδιάστατα αντικείμενα και τα στέλνει στο LiveView μέσω ενός custom listener.

Πίσω στο Map fragment, αρχικά κατά την αρχικοποίηση θα ξεκινήσει ένα νέο thread στο οποίο θα τρέχει κάθε δέκα δευτερόλεπτα το παραπάνω runnable και θα καλεί την initMap. Η initMap πάει στο fragment με τα google maps το αρχικοποιεί και το συνδέει με το Google Cloud για να μπορεί να τραβάει από εκεί τα δεδομένα του χάρτη (τον ίδιο το χάρτη, τις διαδρομές και άλλα). Μόλις τελειώσει η αρχικοποίηση, καλείται η συνάρτηση onMapReady. Εκεί αποθηκεύεται το instance του αντικείμενου του χάρτη, μετά αποθηκεύονται οι συντεταγμένες του χρήστη και η κάμερα μετακινείται για να έχει στο επίκεντρο της την τοποθεσία του χρήστη. Στην συνέχεια καλείται η συνάρτηση askData, η οποία αποθηκεύει σε ένα ArrayList όλα τα δεδομένα για τα σημεία ενδιαφέροντος στην επιλεγμένη διαδρομή και σε αυτή την λίστα καλείται η addMarkers η οποία προσθέτει τα σημεία στο χάρτη.

Τελευταίο tab είναι το Menu στο οποίο φαίνεται μια λίστα με τις διαθέσιμες διαδρομές που μπορεί να επιλέξει ο χρήστης. Η διεπαφή χρησιμοποιεί ένα RecyclerView το οποίο περιέχει τη λίστα. Στην αρχικοποίηση, γίνεται ένα query στη βάση στο Firebase από την οποία επιστρέφονται όλες οι διαδρομές και αποθηκεύονται τοπικά. Στη συνέχεια, φτιάχνεται ένας νέος αντάπτορας RoutesRecyclerViewAdapter ο οποίος γίνεται attach στο RecyclerView της διεπαφής. Αυτός ο αντάπτορας είναι υπεύθυνος για να φτιάξει το UI για κάθε αντικείμενο της λίστας ανάλογα με τα δεδομένα του αντικείμενου, καθώς και να αλλάζει το χρώμα στο επιλεγμένο αντικείμενο για την υπόδειξη του προς τον χρήστη. Αφού ο αντάπτορας αρχικοποιηθεί και γίνει attach, τότε, ακολουθεί η onItemClick η οποία εκτελείται από τον αντάπτορα κάθε φορά που πατιέται ένα αντικείμενο της λίστας. Μέσα στην onItemClick φτιάχνεται ένα νέο αντικείμενο Route από τα δεδομένα που διάβασε από τη βάση δεδομένων και στέλνει το αντικείμενο στο Map Fragment για να δείξει αυτό με την σειρά του την διαδρομή στον χάρτη.

4. Ανταλλαγή δεδομένων μεταξύ Fragment

Η επικοινωνία μεταξύ των fragments γίνεται με τον ακόλουθο τρόπο: πρώτα, στο fragment- αποστολέα φτιάχνεται ένα interface με τις μεθόδους που θα χρησιμοποιηθούν για τη μεταφορά των δεδομένων. Στην main, ο προγραμματιστής κάνει override την συνάρτηση του interface και μέσα στη συνάρτηση παίρνει το instance του fragment- αποδέκτη από τον αντάπτορα, που είχε αναφερθεί αρχικά, και από εκεί καλεί μια συνάρτηση του fragment- αποδέκτη η οποία ανανεώνει τα δεδομένα της, με αυτά που

περνάει στις συναρτήσεις σαν παραμέτρους. Για παράδειγμα: ο προγραμματιστής θέλει να στείλει την επιλεγμένη διαδρομή από τη λίστα από το Menu fragment στο Map fragment για να μπορέσει εκεί να τραβήξει τα δεδομένα των σημείων ενδιαφέροντος της διαδρομής αυτής. Για να το πετύχει αυτό πρέπει: αρχικά, στο Menu Fragment να φτιάξει ένα interface MenuFragmentListener που περιέχει μία μέθοδο, την onMenuDataTransfer που παίρνει σαν παράμετρο ένα αντικείμενο Route. Στην onItemClick που είχε αναφερθεί προηγουμένως, ο προγραμματιστής παίρνει ένα instance του MenuFragmentListener και καλεί την onMenuDataTransfer και κατόπιν περνάει σαν παράμετρο τη διαδρομή του αντικειμένου της λίστας που επιλέχθηκε. Στη συνέχεια, στην Main κάνει override την onMenuDataTransfer(Route) και εδώ γράφει τον κώδικα που θα εκτελεί η συνάρτηση. Πιο συγκεκριμένα, θα παίρνει το αποθηκευμένο instance του Map Fragment μέσα από το instance του Main Adapter και εκεί θα καλεί την συνάρτηση updatePlacesData.

mainPagerAdapter.getMapFragmentInstance().updatePlacesData(route);

Στο Map Fragment φτιάχνει την updatePlacesData, που παίρνει σαν παράμετρο ένα αντικείμενο Route και αυτή αλλάζει το αντικείμενο Route και έπειτα καλεί την askData για να πάρει από την βάση δεδομένων τη λίστα με τα σημεία ενδιαφέροντος για να τα δείξει στο χάρτη.

Με παρόμοιο τρόπο μεταφέρεται και το επιλεγμένο σημείο ενδιαφέροντος από το Map Fragment στο Live Fragment μέσω της συνάρτησης onMapDataTransfer καθώς γίνεται και η μεταφορά, πάλι από το Map Fragment στο Live Fragment, μέσω της συνάρτησης gettingAwayFromPlaces που καλείται όταν ο χρήστης που βρίσκεται εντός εμβέλειας ενός σημείου ενδιαφέροντος, απομακρυνθεί εκτός εμβέλειας του.

5. Δομή των Αντικειμένων

Σε αυτό το σημείο, είναι αναγκαίο να γίνει εξήγηση του τρόπου με τον οποίο είναι δομημένα τα δεδομένα που διαβάζει η εφαρμογή από τη βάση δεδομένων, καθώς πολλά από αυτά έχουν ήδη αναφερθεί. Αρχικά, υπάρχει η κλάση Route με δύο attributes, τα title και places. Το title είναι το αναγνωριστικό όνομα που έχει δώσει ο διαχειριστής στην διαδρομή ενώ η λίστα places περιέχει τα ονόματα (titles) των σημείων ενδιαφέροντος που σχετίζονται με αυτή τη διαδρομή.

Στη συνέχεια, υπάρχει το σημείο ενδιαφέροντος (Place). Αυτό περιέχει τις συντεταγμένες του σημείου, το αναγνωριστικό όνομα-τίτλο του σημείου, το AOE (Area of Effect), το οποίο είναι η μέγιστη απόσταση που πρέπει να απέχει ο χρήστης από το σημείο ενδιαφέροντος για να μπορεί να δει τα ψηφιακά αντικείμενα που αντιστοιχούν στο σημείο αυτό. Τελευταίο στοιχείο, αποτελεί το hologramReference που είναι ένας δείκτης που δείχνει το σημείο που βρίσκεται το αντίστοιχο αντικείμενο τύπου Hologram στη βάση δεδομένων.

Hologram είναι τα τρισδιάστατα αντικείμενα που βλέπει ο χρήστης. Υπάρχουν δύο είδη αντικειμένων που μπορεί κανείς να δει. Πρώτον, υπάρχει ένας διαδραστικός

τρισδιάστατος πίνακας και δεύτερον, υπάρχουν τρισδιάστατα αντικείμενα που μπορούν να φορτωθούν από αρχεία. Για τον πίνακα, υπάρχουν τα εξής χαρακτηριστικά που μπορούν να εμφανιστούν:

1. Title: ο τίτλος που εμφανίζεται στο πάνω μέρος του πίνακα.
2. imageUrl: το url μιας προαιρετικής φωτογραφίας που εμφανίζεται κάτω από τον τίτλο στο πίνακα.
3. Description: το “σώμα” του πίνακα το οποίο δίνει αναλυτικές πληροφορίες στο χρήστη για το σημείο στο οποίο βρίσκεται.
4. Question: μια προαιρετική ερώτηση που μπορεί να γίνει προς το χρήστη σχετική με το σημείο στο οποίο βρίσκεται.
5. Answers: οι πιθανές απαντήσεις που μπορεί να επιλέξει ο χρήστης για την παραπάνω ερώτηση μέσω μιας λίστας με RadioButtons.
6. webUrl: μια προαιρετική ιστοσελίδα που μπορεί να φανεί στο τέλος του πίνακα. Είναι διαδραστική και ο χρήστης μπορεί να χρησιμοποιήσει αρκετές από τις συνηθισμένες λειτουργίες μιας ιστοσελίδας.

Δίπλα στον πίνακα, παρέχεται η δυνατότητα να εμφανιστεί και ένα τρισδιάστατο μοντέλο. Στο αντικείμενο Hologram, αυτό συνδέεται μέσω μεταβλητής arModelReference η οποία αποτελεί ένα δείκτη, ο οποίος συνδέεται με ένα αντικείμενο τύπου ArModel. Αυτή η δυνατότητα είναι προαιρετική και δε χρειάζεται ο διαχειριστής να την προσθέσει αν δεν το επιθυμεί.

Οπότε, για το τέλος, ο διαχειριστής έρχεται αντιμέτωπος με το αντικείμενο ArModel. Αυτό περιέχει όλα τα χαρακτηριστικά που χρειάζονται για να εμφανίζεται όπως το επιθυμεί ο διαχειριστής το αντικείμενο. Χαρακτηριστικά του αποτελούν τα:

1. Title: όνομα- αναγνωριστικό του αντικειμένου.
2. modelUrl: URL που επιστρέφει από τη βάση δεδομένων το αρχείο, που περιέχει το αντικείμενο που θα εμφανιστεί..
3. Scale: κλίμακα του μεγέθους του αντικειμένου ώστε να μπορεί ο διαχειριστής να το ρυθμίζει στο επιθυμητό μέγεθος.
4. distFromAnchor: η απόσταση του αντικειμένου από τον πίνακα. Όσο το νούμερο μεγαλώνει, το αντικείμενο θα εμφανίζεται όλο και πιο μακριά από τα αριστερά του πίνακα.
5. animationSpeed: η ταχύτητα περιστροφής του αντικειμένου γύρω από τον άξονά του. Στο μηδέν το αντικείμενο μένει ακίνητο.

Όλη η παραπάνω διαδικασία, του πως λειτουργεί η εφαρμογή, περιγράφεται συνοπτικά μέσα από το παρακάτω διάγραμμα ροής δεδομένων.

αυθεντικοποιεί τον διαχειριστή και να παρέχει πρόσβαση μόνο σε λογαριασμούς διαχειριστών. Επίσης, γίνεται και χρήση του Firebase Host (Serverless hosting υπηρεσία) από όπου η ιστοσελίδα γίνεται διαθέσιμη μέσω του διαδικτύου.

1. Η main

Πιο αναλυτικά, η σελίδα για τους διαχειριστές της πλατφόρμας ξεκινάει με το αρχείο `main.dart`. Εκεί αρχικοποιείται η εφαρμογή “τρέχοντας” το `MyApp Widget`. Το `Widget` αυτό κληρονομεί το `Stateless Widget` καθώς δεν θα γίνονται μαζικές αλλαγές σε αυτό σε δευτερεύοντα χρόνο. Μέσα στο `widget`, αρχικοποιείται ένα `session` με τον `firebase server` και στη συνέχεια, μέσα στην `build method` επιστρέφεται έναν `FutureBuilder`. Δουλειά του `future builder` είναι να περιμένει να ολοκληρωθεί η αρχικοποίηση του `firebase session`. Μέχρι να γίνει αυτό, το `MyApp` δείχνει στον χρήστη ένα μήνυμα “Loading...”. Αν η σύνδεση αποτύχει θα εμφανιστεί ένα μήνυμα στο χρήστη που θα δείχνει ότι η σύνδεση απέτυχε.

```
// Create the initialization Future outside of `build`:
final Future<FirebaseApp> _initialization = Firebase.initializeApp();

@override
Widget build(BuildContext context) {
  return FutureBuilder(
    future: _initialization,
    builder: (context, snapshot) {
      // check for errors
      if (snapshot.hasError) {
        // return a widget with an error message.
        return somethingWentWrong();
      }

      // Once complete init the firebase connection starts the app.
      if (snapshot.connectionState == ConnectionState.done) {
        // code that starts the app.
      }

      // show something whilst waiting for initialization to complete.
      return loading();
    },
  );
}
```

Πίνακας 1: Future Builder

2. Αυτόματη Σύνδεση Χρήστη

Αν από την άλλη όλα είναι εντάξει και το session δημιουργηθεί κανονικά, τότε η build θα επιστρέψει ένα StreamProvider. Το stream που ο διαχειριστής επιθυμεί να περάσει στο υπόλοιπο subtree, περιέχει τη κατάσταση της αυθεντικοποίησης του χρήστη. Συγκεκριμένα, σαν value στο provider θα περάσει το AuthService().user. Η κλάση AuthService θα αναλυθεί πλήρως αργότερα, αλλά προς το παρόν σημαντικό είναι να γνωρίζει κανείς, ότι το user variable γυρνάει πίσω ένα Stream με την τρέχουσα κατάσταση του χρήστη. Συνεχίζοντας, ο provider προχωρά στη δημιουργία ενός νέου MaterialApp βάζοντας τα θεμέλια του UI της εφαρμογής. Το MaterialApp θα θέσει σαν home page της εφαρμογής το Widget Wrapper. Ο Wrapper έχει μια πολύ σημαντική δουλειά, διαβάζει το Stream από τον Provider και αν δεν υπάρχει κάποιος ήδη συνδεδεμένος χρήστης στην εφαρμογή τότε τον μεταβιβάζει στην AuthPage για να συνδεθεί, αλλιώς τον οδηγεί στην HomePage.

```
final user = Provider.of<MyUser>(context);
if (user == null) {
  return AuthPage();
}else{
  return HomePage(title: "GeoAR app - Web UI",);
}
```

Πίνακας 2: Using provider value to choose a page

3. Σελίδα Αυθεντικοποίησης χρήστη

Αν δεν είναι κάποιος χρήστης αυθεντικοποιημένος(authenticated), θα φορτωθεί το Widget AuthPage. Η AuthPage είναι μια από τις δύο σελίδες της web εφαρμογής. Εδώ ο χρήστης μπορεί να συνδεθεί στην σελίδα. Η σελίδα αυτή έχει δύο πεδία με τα οποία μπορεί να διαδράσει ο χρήστης. Το πρώτο είναι το πεδίο για το email και το δεύτερο για τον κωδικό του λογαριασμού. Με το που ξεκινάει το Widget, αρχικοποιείται το AuthService.

```
// auth change user stream
Stream<MyUser> get user {
  return _auth.authStateChanges().map(_user);
}
```

Πίνακας 3: Get the logged in user

a. Κλάση AuthService

Το AuthService είναι μια custom κλάση η οποία ελέγχει την κατάσταση της αυθεντικοποίησης του χρήστη. Συγκεκριμένα, όπως αναφέρθηκε και προηγουμένως, υπάρχει μια get μέθοδος η οποία επιστρέφει ένα stream με την τρέχουσα κατάσταση του χρήστη. Δηλαδή, γυρνάει ένα αντικείμενο MyUser που περιέχει το μοναδικό αναγνωριστικό id του χρήστη. Αν δεν υπάρχει κάποιο αναγνωριστικό αποθηκευμένο τοπικά, τότε επιστρέφει null (δηλαδή δεν είναι κάποιος χρήστης συνδεδεμένος αυτή την στιγμή από αυτήν τη συσκευή).

```
Future signIn(String email, String password, var context) async {
  try {
    UserCredential userCredential = await
    _auth.signInWithEmailAndPassword(
      email: email,
      password: password,
    );
    _validationResult = null;
    return _user(userCredential.user);
  } on FirebaseAuthException catch (e) {
    if (e.code == 'user-not-found') {
      print('No user found for that email.');
```

```
      _validationResult = "No user found for that email.";
    } else if (e.code == 'wrong-password') {
      print('Wrong password provided for that user.');
```

```
      _validationResult = "Wrong password provided for that user.";
    } else if (e.code == "invalid-email") {
      print(e.message);
      _validationResult = "The email address is badly formatted.";
    } else {
      print(e.code);
      _validationResult = null;
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text(e.message),
        ),
      );
    }
    return null;
  }
}
```

Πίνακας 4: SignIn

Πέρα από αυτήν την getter, παρέχονται και οι μέθοδοι signIn και signOut. Η signIn παίρνει σαν παραμέτρους το email, τον κωδικό του χρήστη και το τρέχον context του Widget. Σε μια try, μέσα στη μέθοδο, γίνεται προσπάθεια μέσω του api του Firebase να

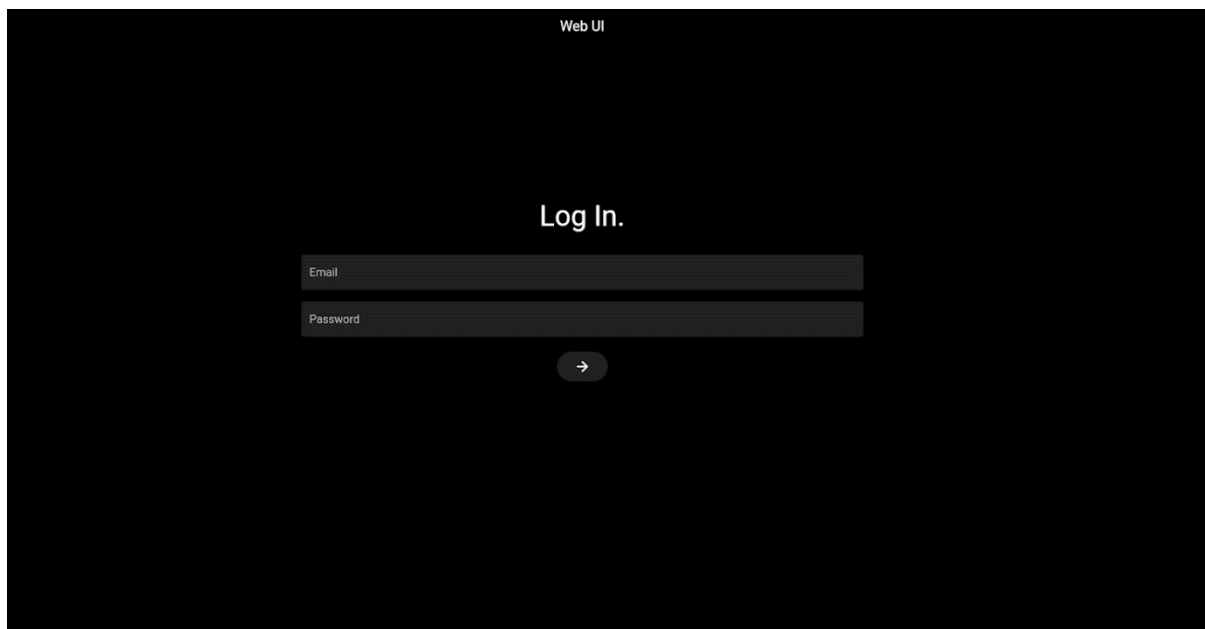
ανακτήσει τα credentials του χρήστη. Αν η διαδικασία στεφθεί επιτυχώς, τότε η signIn θα βάλει το id που επιστράφηκε από το Firebase σε ένα αντικείμενο MyUser και στη συνέχεια θα το επιστρέψει στον χρήστη. Αν από την άλλη, η διαδικασία αποτύχει, θα επιστραφεί ένα FirebaseAuthException. Ανάλογα με τον κωδικό του error που θα επιστραφεί θα αποθηκευτεί ένα αντίστοιχο μήνυμα στη μεταβλητή validationResult. Τα πιθανά error που ελέγχει η εφαρμογή είναι τα εξής:

- Δεν βρέθηκε χρήστης με αυτό το email.
- Λανθασμένος κωδικός πρόσβασης.
- Το email έχει λανθασμένο format.

Αν το error δεν αντιστοιχεί σε κάποια από αυτές τις κατηγορίες τότε το validationResult παραμένει null όπως είχε αρχικοποιηθεί και εμφανίζεται ένα Snackbar στην οθόνη με το μήνυμα που επιστρέφει το Exception. Η signInOut προσπαθεί μέσα σε μια try να αποσυνδέσει τον χρήστη με την χρήση του Firebase API.

```
// sign out the user by calling the signOut method from the FirebaseAuth //  
object  
await _auth.signOut();
```

Πίνακας 5: Sign Out



Εικόνα 35: Log In Page

b. UI της σελίδας

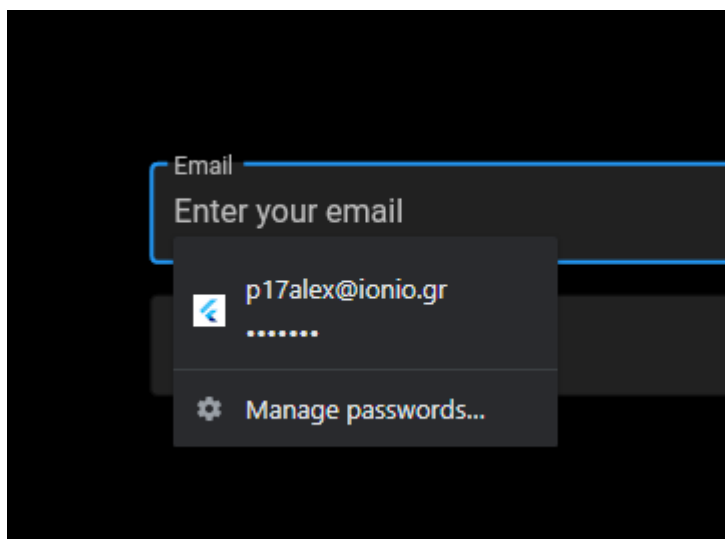
Αφού αρχικοποιηθεί η AuthService, δημιουργούνται δύο TextEditingControllers για να ελέγχουν το κείμενο που εισάγει ο χρήστης στα δύο TextFields. Εφόσον, τελειώσουν οι αρχικοποιήσεις, επιστρέφεται από την build η ίδια η σελίδα. Στο πάνω μέρος θα

δημιουργηθεί ένα AppBar με τον τίτλο “Web UI”, δηλώνοντας έτσι ότι πρόκειται για την web έκδοση διαχείρισης της βάσης δεδομένων. Στο κέντρο του σώματος της σελίδας, δημιουργείται μια φόρμα. Στο επάνω μέρος της φόρμας, υπάρχει ένα κείμενο που αναγράφει με μεγάλη γραμματοσειρά “Log In”, υποδεικνύοντας στον χρήστη να βάλει τους κωδικούς του λογαριασμού του. Ακριβώς από κάτω, όπως είναι αναμενόμενο, υπάρχει ένα πεδίο κειμένου με τον τίτλο “email”, ώστε να συμπληρώσει ο χρήστης το email του λογαριασμού του. Ακολουθώντας, βρίσκεται το πεδίο κειμένου με τίτλο “password”, για να συμπληρώσει ο χρήστης τον αντίστοιχο κωδικό πρόσβασης του λογαριασμού. Τέλος, υπάρχει ένα κουμπί για να συνδεθεί ο χρήστης με τα στοιχεία που συμπλήρωσε στη φόρμα.

```
// form's global key  
final _formKey = GlobalKey<FormState>();
```

Πίνακας 6: Form's global key

Πιο αναλυτικά, η φόρμα χρησιμοποιεί ένα global key για να μπορεί να γίνει reference και εκτός του block κώδικα της. Μέσα στην φόρμα τώρα, τα παιδιά της “τυλίγονται” όλα μέσα σε ένα AutofillGroupWidget. Αυτό θα διασφαλίσει ότι ο φυλλομετρητής θα μπορεί να συμπληρώσει αυτόματα τα αποθηκευμένα στοιχεία, καθώς και να αποθηκεύσει κάποια νέα στοιχεία που συμπλήρωσε ο χρήστης μέσω της εκάστοτε υπηρεσίας “autofill”.



Εικόνα 36: Autofill functionality

Στη συνέχεια, εντός του autofillgroup υπάρχει μια στήλη που περιέχει τα δύο textFormFields. Το κάθε πεδίο έχει ένα hint, το οποίο εμφανίζεται όταν ο χρήστης πατήσει πάνω στο πεδίο και παραμένει για όσο χρόνο δεν έχουν συμπληρωθεί χαρακτήρες εντός του πεδίου. Σκοπός του είναι, να προϊδεάσει τον χρήστη στο τί πρέπει να συμπληρώσει

εντός του πεδίου. Ακόμη, τα πεδία έχουν και ένα label το οποίο λειτουργεί σαν τον τίτλο του πεδίου. Όταν δεν είναι επιλεγμένο το πεδίο, το label εμφανίζεται στη μέση του πεδίου, αλλιώς εμφανίζεται πάνω από το πεδίο. Αυτή η μορφή του UI δημιουργείται αυτόματα από την στιγμή που έχουμε θέσει εντός του decoration attribute των πεδίων το Widget OutlineInputBorder σαν border του.

```
TextFormField(  
  // other text field attributes...  
  controller: emailController,  
  autofillHints: [AutofillHints.email],  
  decoration: InputDecoration(  
    // other decoration attributes...  
    border: OutlineInputBorder(),  
  ),  
  labelText: 'Email',  
  hintText: 'Enter your email',  
)
```

Πίνακας 7: Text Form Field

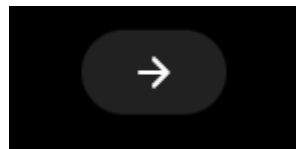
Μετά, στο attribute controller των πεδίων ο προγραμματιστής κάνει attach το αντίστοιχο TextEditingController, τα οποία αναφέρθηκαν προηγουμένως, για να μπορεί να κάνει reference στην τιμή του πεδίου. Επίσης, στο χαρακτηριστικό autofillHints δηλώνει τον τύπο του πεδίου, όπως για παράδειγμα στο πεδίο με το email το βάζει ως (AutofillHints.email) για να μπορεί η εκάστοτε υπηρεσία autofill να αντιληφθεί τον τύπο πληροφορίας που συμπληρώνεται εντός του πεδίου.

```
validator: (value) {  
  if (value.isEmpty) {  
    return "Please enter your email.";  
  } else {  
    String tmp = as.validationResult;  
    if (tmp == "No user found for that email." ||  
        tmp == "The email address is badly formatted.") {  
      return tmp;  
    }  
  }  
  return null;  
},
```

Πίνακας 8: Making Validation using the validators

Τελευταίο χαρακτηριστικό των πεδίων, αποτελεί ο validator. Η συνάρτηση που δέχεται σαν τιμή ο validator θα εκτελεστεί αργότερα, όταν πραγματοποιηθεί η αυθεντικοποίηση. Η συνάρτηση αυτή πρώτα θα κοιτάξει την τρέχουσα τιμή του πεδίου. Αν είναι null, η συνάρτηση θα επιστρέψει το μήνυμα "Please enter your email" για το email και "Please

enter your password” για τον κωδικό. Αν δεν είναι null, τότε θα διαβάσει το μήνυμα validationResult από το αντικείμενο του authService. Αν το μήνυμα σχετίζεται με το email στο πεδίο του email θα επιστρέψει το μήνυμα validationResult ενώ στο πεδίο του κωδικού θα επιστρέψει null. Αντίθετα, αν το μήνυμα σχετίζεται με τον κωδικό στο πεδίο του email θα επιστρέψει null ενώ στο πεδίο του κωδικού θα επιστρέψει το μήνυμα από το validationResult. Αυτή η τιμή που επιστρέφεται είναι σημαντική επειδή, αν είναι null το πεδίο είναι αποδεκτό (ο χρήστης έβαλε σωστά στοιχεία στα πεδία). Αν όμως επιστραφεί κάποιο κείμενο τότε αυτόματα το widget του TextFormField θα εμφανίσει το μήνυμα λάθους, που επιστράφηκε στον χρήστη.



Εικόνα 37: Log In button

```
// Login Button
MaterialButton(
  child: Icon(
    Icons.arrow_forward_rounded,
    color: Colors.white,
  ),
  // more button customisation attributes...
  onPressed: () async {
    if (_formKey.currentState.validate()) {
      await as.signIn(
        emailController.text,
        passController.text,
        context,
      );
    }
  },
),
```

Πίνακας 9: Login Button

Τέλος, στο κάτω μέρος της φόρμας υπάρχει ένα κουμπί που χρησιμοποιείται για την αυθεντικοποίηση των στοιχείων και για την είσοδο του χρήστη στην σελίδα. Πιο αναλυτικά, το κουμπί περιέχει ένα εικονίδιο με ένα βέλος προς τα δεξιά για να δείξει στον χρήστη ότι αυτό πρέπει να πατηθεί μετά την συμπλήρωση της φόρμας για να αποκτήσει πρόσβαση στο σύστημα. Τώρα, όταν το κουμπί πατηθεί, τότε θα εκτελεστεί μια συνάρτηση η οποία χρησιμοποιεί το global key της φόρμας, για να την κάνει validate, δηλαδή να εκτελέσει όλες της validate μεθόδους σε όλα τα πεδία εντός φόρμας. Αν το validation πετύχει, τότε η συνάρτηση θα καλέσει την signIn του authService για να

αυθεντικοποιήσει τον χρήστη. Στην συνέχεια η οθόνη θα γίνει rebuild και αν η αυθεντικοποίηση πετύχει τότε ο Wrapper που θα γίνει και αυτός rebuild θα μεταφέρει τον χρήστη στην homePage. Αλλιώς, αν η αυθεντικοποίηση αποτύχει τότε η signIn θα αποθηκεύσει το μήνυμα του error όπως έχει περιγραφεί προηγουμένως και η οθόνη θα γίνει rebuild έχοντας ως αποτέλεσμα το λανθασμένο πεδίο ή ένα Snackbar που θα δείξει το μήνυμα λάθους όπως έχει ήδη περιγραφεί. Μια επιπλέον πληροφορία, είναι ότι όταν ο χρήστης αυθεντικοποιηθεί, το api του firebase αυτόματα θα αποθηκεύσει τα credentials του χρήστη στο local storage και αν ο χρήστης κλείσει την καρτέλα ή τον φυλλομετρητή και την ξανανοίξει αργότερα θα συνδεθεί κατευθείαν, χωρίς να χρειάζεται να περάσει από το authPage.

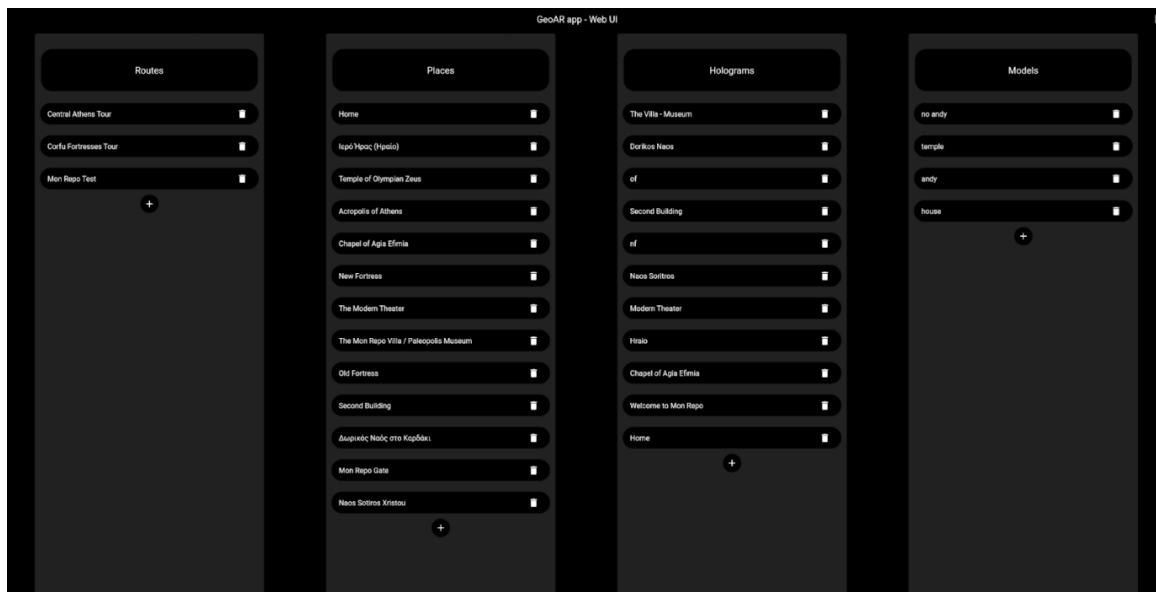
4. Home Page

a. Αρχικοποίηση

Μέχρι τώρα, έγινε γνωστός ο τρόπος με τον οποίον η σελίδα μπορεί να αναγνωρίσει αν ο χρήστης είναι ήδη συνδεδεμένος, και αν όχι πώς να τον αυθεντικοποιήσει. Οπότε στην συνέχεια θα εξηγηθεί η βασική σελίδα της ιστοσελίδας που επιτρέπει στον χρήστη να ελέγχει τις διαδρομές που είναι αποθηκευμένες στην βάση δεδομένων. Αρχίζοντας, όπως περιγράφηκε και προηγουμένως, όταν ο χρήστης αυθεντικοποιηθεί, ο Wrapper θα εμφανίσει στην οθόνη το Widget HomePage. Το widget αυτό θα αρχικοποιήσει τέσσερις λίστες που η κάθε μια από αυτές αποθηκεύει ένα αντικείμενο που χρησιμοποιείται και στην εφαρμογή. Συγκεκριμένα, η πρώτη λίστα θα αποθηκεύει τα routes που υπάρχουν στην βάση, η δεύτερη όλα τα places αυτών των routes, η τρίτη τα διάφορα holograms που έχουν τα places και τέλος τα arModels που αντιστοιχούν στα holograms αυτά.

```
List<MyRoute> routes = [];  
List<Place> places = [];  
List<Hologram> holograms = [];  
List<ArModel> arModels = [];
```

Πίνακας 10: Lists containing the data model



Εικόνα 38: Main page

b. Δομή UI

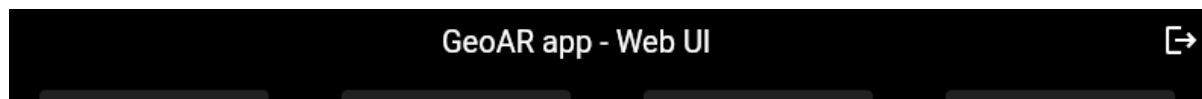
Μόλις αρχικοποιηθούν αυτές οι λίστες, ξεκινάει η build για να δημιουργήσει το UI. Μέσα στην build πρώτα αρχικοποιείται ένα αντικείμενο του AuthService. Στην συνέχεια, η build επιστρέφει το UI ξεκινώντας με ένα Scaffold widget, το οποίο περιέχει ένα appBar με τον τίτλο της σελίδας και ένα κουμπί-εικονίδιο, το οποίο για να κάνει logout τον χρήστη, χρησιμοποιεί το AuthService αντικείμενο που δημιουργήθηκε προηγουμένως.

```
AppBar(
  elevation: 0.0,
  // the title in the center of the screen.
  title: Center(child: Text(widget.title)),
  actions: <Widget>[
    IconButton(
      icon: Icon(Icons.logout), //the icon of the button.
      onPressed: () async {
        // sign out the user when the button is clicked.
        await _auth.signOut();
      },
    ),
  ],
  backgroundColor: Colors.black,
),
```

Πίνακας 11: AppBar

Όπως φαίνεται και στον κώδικα από επάνω το κουμπί-εικονίδιο βρίσκεται μέσα σε μια λίστα από Widget που χρησιμοποιείται σαν value στην παράμετρο actions. Το widget

AppBar τοποθετεί όλα τα widget του actions στην δεξιά γωνία του appBar σε αντίθεση με το value από την παράμετρο title, που τοποθετείται κατά μήκος όλης του υπόλοιπου appBar.



Εικόνα 39: AppBar

γ. Λίστες - Το βασικό περιεχόμενο

Στο επίκεντρο της σελίδας βρίσκονται τέσσερις λίστες. Η κάθε μια από αυτές περιέχει τα δεδομένα των τεσσάρων βασικών τύπων αντικειμένων της εφαρμογής. Τα αντικείμενα αυτά συνδέονται μεταξύ τους, ώστε στο τέλος να μπορούν να παρέχουν στην εφαρμογή όλα τα δεδομένα που απαιτεί, για να την ανάδειξη ενός σημείου ενδιαφέροντος στον χρήστη.

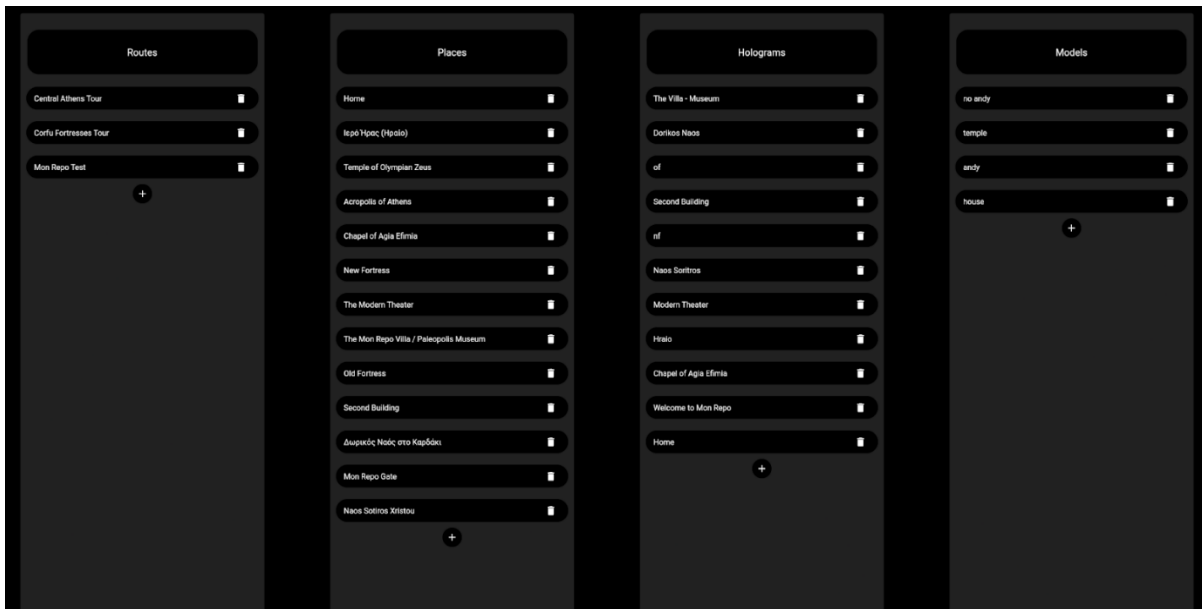
Η κάθε λίστα είναι και ένα widget που ανοίγει ένα streaming session με την βάση δεδομένων, για να μπορεί να λαμβάνει και να αλλάζει το περιεχόμενο της λίστας δυναμικά κατά την διάρκεια χρήσης της σελίδας. Η μεταβλητή collectionName καθορίζει το collection από το οποίο η λίστα θα πάει και θα τραβήξει τα δεδομένα, που θα εμφανίσει στον χρήστη. Όπως αναφέρθηκε και προηγουμένως, επειδή τα δεδομένα γίνονται stream αυτό σημαίνει ότι ο κώδικας κάθε φορά που γίνεται ένα event OnChange, δηλαδή κάθε φορά που υπάρχει/γίνεται μια αλλαγή στα δεδομένα της βάσης, η βάση “ενημερώνει” την λίστα για την αλλαγή. Εκεί ο κώδικας “κοιτάει” και αν έχει προστεθεί κάτι καινούριο τότε το προσθέτει στο τέλος της λίστας. Αντίστοιχα, αν έχει διαγραφεί κάτι τότε αφαιρείται από την λίστα.

```

snapshot.data.docChanges().forEach((docChange) {
// add new item when an object gets added and remove an item when a object
get deleted.
    if (docChange.type == DocumentChangeType.added) {
        if (collectionName == "routes") {
            // put data to the "routes" list
            ...
        } else if (collectionName == "marker") {
            // put data to the "markers" list
        }
        ...
    } else if (docChange.type == DocumentChangeType.removed) {
        // remove data from the appropriate list.
        ...
    }
}
}

```

Πίνακας 12: Example of applying changes when data are changed



Εικόνα 40: Main Page

ι.Λίστα με ArModel

Ξεκινώντας από τα δεξιά προς τα αριστερά, η πρώτη εκ των τεσσάρων λιστών είναι τα μοντέλα. Σαν έννοια το μοντέλο είναι το τρισδιάστατο αντικείμενο που θα διαδρά ο χρήστης όταν βρίσκεται σε κάποιο σημείο που χρησιμοποιείται αυτό το μοντέλο. Στην σελίδα αποθηκεύεται σαν ένα αντικείμενο ArModel. Αυτό το αντικείμενο έχει τα εξής χαρακτηριστικά: τον τίτλο και ένα μοναδικό χαρακτηριστικό του κάθε αντικειμένου, που το προσδιορίζει και το ξεχωρίζει από τα υπόλοιπα αντικείμενα ArModel. Το ModelURL που περιέχει το path στον FTP σέρβερ (δηλαδή την υπηρεσία Storage του Firebase), από όπου θα μπορεί να πάρει η εφαρμογή το αρχείο με το τρισδιάστατο μοντέλο που θα εμφανιστεί στον χρήστη. Ακόμα, υπάρχει η μεταβλητή reference η οποία αποτελεί ένα

αντικείμενο τύπου `DocumentReference` και χρησιμοποιείται σαν pointer ως προς το αντίστοιχο έγγραφο που έχει αποθηκευτεί στην βάση. Η μεταβλητή `scale` παρέχεται για να μπορεί να αυξομειώνεται το μέγεθος του τρισδιάστατου αντικειμένου όταν εμφανίζεται στην οθόνη. Το `distFromAnchor` αντισταθμίζει την απόσταση που θα εμφανιστεί το αντικείμενο ως προς τον άξονα X. Αυτό είναι χρήσιμο, αφού με βάση το ίδιο anchor θα εμφανιστούν δύο αντικείμενα στην οθόνη. Το πρώτο θα είναι ένας/μία τρισδιάστατη οθόνη/πίνακας που παρέχει πληροφορίες στον χρήστη και δεύτερο θα είναι ένα τρισδιάστατο αντικείμενο που βρίσκεται αποθηκευμένο και ανακτάται από το FTP σέρβερ. Το `offset` αυτό χρησιμοποιείται για να μην πέσει το ένα αντικείμενο πάνω στο άλλο και δεν παρέχουν τελικός τις πληροφορίες που θα έπρεπε. Τέλος, έχουμε το `animationSpeed` το οποίο υποδεικνύει την ταχύτητα περιστροφής του αντικειμένου που παίρνει η εφαρμογή από τον FTP σέρβερ.

ii.Λίστα με Holograms

Στη συνέχεια, η δεύτερη λίστα περιέχει τα Holograms. Το κάθε αντικείμενο hologram της λίστας, παρέχει τα δεδομένα που θα εμφανίζονται στην τρισδιάστατη οθόνη που με την σειρά τους θα εμφανίζονται στην οθόνη του χρήστη. Τα δεδομένα που διαβάζονται από την βάση για να δημιουργηθούν αυτά τα αντικείμενα είναι τα παρακάτω. Ένας τίτλος, σαν μοναδικό αναγνωριστικό του αντικειμένου το οποίο επίσης εμφανίζεται και στο πάνω μέρος της ψηφιακής οθόνης. Κάτω από τον τίτλο στην οθόνη εμφανίζεται η περιγραφή, ένα κείμενο που περιγράφει ή δίνει πληροφορίες σχετικά με το σημείο ενδιαφέροντος. Όπως και προηγουμένως, ένα reference που δείχνει στο αντίστοιχο Document του αντικειμένου μέσα στην βάση. Το `question` και το `answerArray`, δηλαδή μια σύντομη ερώτηση σχετική με το σημείο και τις πιθανές απαντήσεις από τις οποίες μπορεί να διαλέξει ο χρήστης αντίστοιχα. Ακόμη, περιέχει το `imageUrl` και το `webURL`. Το πρώτο περιέχει ένα string με το url για μια φωτογραφία που μπορεί να εμφανιστεί στην οθόνη, ενώ το δεύτερο περιέχει το url για μια ιστοσελίδα που μπορεί να “φορτωθεί” μέσα στην οθόνη. Τέλος, παίρνει από την βάση το `arModelReference` δηλαδή ένα δείκτη που δείχνει στο αντικείμενο `ArModel`, το οποίο θα εμφανίζεται δίπλα από την τρισδιάστατη οθόνη.

iii.Λίστα με τα Places

Η τρίτη λίστα περιέχει τα αντικείμενα places. Όπως και τα προηγούμενα, τα αντικείμενα αυτά περιέχουν κάποιες μεταβλητές που τα περιγράφουν, εμφανίζονται στις λίστες και αργότερα θα χρησιμοποιηθούν τα δεδομένα αυτά και στην εφαρμογή. Σε αυτά τα αντικείμενα, οι μεταβλητές αυτές είναι ο τίτλος, δηλαδή το όνομα του σημείου ενδιαφέροντος, ένα map, που περιέχει τις συντεταγμένες του σημείου, η μεταβλητή AOE, που δείχνει την απόσταση στην οποία εντός της ο χρήστης μπορεί να δει τα αντίστοιχα τρισδιάστατα μοντέλα, το `hologramReference`, που είναι ένας δείκτης που δείχνει στο αντικείμενο hologram, που αντιστοιχεί σε αυτό το σημείο ενδιαφέροντος.

Τέλος, υπάρχει πάλι ένα reference, που δείχνει στο Document του αντίστοιχου αντικειμένου Place, μέσα στην βάση.

iv.Λίστα με τα Routes

Η τελευταία λίστα αποτελείται από τα αντικείμενα τύπου MyRoute. Σκοπός αυτών των αντικειμένων είναι να παρέχουν τα Places τα οποία θα περιλαμβάνει κάθε διαδρομή. Το αντικείμενο αυτό είναι πιο απλό από τα προηγούμενα και περιλαμβάνει μόνο τρία χαρακτηριστικά. Όπως και όλα τα προηγούμενα έχει τον τίτλο της διαδρομής που χρησιμοποιείται ως ένα μοναδικό χαρακτηριστικό του αντικειμένου. Επίσης, περιλαμβάνεται μια λίστα με τους τίτλους των αντικειμένων Places σαν Reference σε αυτά. Και τέλος, έχουμε ένα reference που δείχνει στο Document, του αντίστοιχου αντικειμένου Route, στην βάση δεδομένων. Αν παρατηρήσει κανείς, θα δει ότι η αντίστοιχη λίστα από τα αριστερά έχει ένα Reference σε ένα αντικείμενο από την δεξιά του λίστα. Αυτό συμβαίνει για να ξεχωρίσει την κάθε λειτουργία, πιο συγκεκριμένα πως λειτουργεί το σύστημα με τις διαδρομές και άρα γίνεται πιο εύκολα διαχειρίσιμο.

d. Αντικείμενο λίστας

Όταν δημιουργείται ένα νέο αντικείμενο πρέπει να δημιουργηθεί ένα widget που θα δείχνει στον χρήστη τις πληροφορίες που περιέχει το αντικείμενο. Αυτό επιτυγχάνεται με τον εξής τρόπο. Αμέσως μετά, αφού έχει δημιουργηθεί ένα από τα προηγούμενα αντικείμενα, καλούμε μια συνάρτηση, η `makeItem` με παραμέτρους το αντικείμενο που μόλις φτιάχτηκε και το `context` της σελίδας. Η `makeItem` τελικά, θα φτιάξει ένα widget το οποίο θα μπει στην λίστα `textViewList`. Το widget που θα παράγεται θα έχει την δομή που θα περιγραφεί τώρα. Πρώτα, χρησιμοποιείται ένα `container widget` για να θέσει τις διαστάσεις συγκριτικά με τις διαστάσεις της οθόνης. Σαν `child` του `container` είναι ένα `widget card`, και σαν `child` αυτού ένα `widget inkwell` αντίστοιχα. Αυτά τα δύο χρησιμοποιούνται για αισθητικούς λόγους. Δηλαδή, μορφοποιούν το τελικό widget εμφανισιακά καθώς προσθέτουν και ένα `animation` όταν πατάει ο χρήστης το αντικείμενο της λίστας, με αυτό τον τρόπο παρέχεται `feedback` στον χρήστη για το τι συμβαίνει. Σαν παιδί του `inkwell` υπάρχει ένα `listTile` που χρησιμοποιεί σαν τίτλο, τον τίτλο του αντικειμένου και σαν `trailing`, ένα εικονίδιο `delete` το οποίο όταν πατηθεί διαγράφει το αντίστοιχο αντικείμενο από την βάση δεδομένων και στην συνέχεια, εξαιτίας αυτού του `live update` που έχει φτιαχτεί για τις λίστες θα αφαιρεθεί και από την λίστα αφού η βάση ενημερώθηκε ότι υπήρξε αλλαγή στα δεδομένα. Πηγαίνοντας λίγο πιο πίσω, αν πατήσει κάποιος στην επιφάνεια του `inkwell` τότε καλείται η συνάρτηση `execShowItemInfoDialog`. Εκεί, δημιουργείται και εμφανίζεται ένα νέο `simple dialog` με τίτλο τον τίτλο του αντίστοιχου αντικειμένου, ενώ σαν `body` έχει όλα τα δεδομένα του αντικειμένου όπως αυτά επιστρέφονται από την `toString`. Είναι σημαντικό να αναφερθεί ότι στο τέλος της κάθε λίστας προστίθεται ένα widget μέσω της συνάρτησης `makeAddButton`. Αυτή η συνάρτηση

δημιουργεί ένα Circle Avatar με παιδί ένα εικονίδιο με σύμβολο το “συν». Όταν πατηθεί το κουμπί αυτό καλείται η `execShowAddItemDialog`. Η συνάρτηση, με την σειρά της, θα προσθέσει στο Navigator το custom page `OnAddButtonClick` με παραμέτρους τα αντικείμενα της δεξιάς λίστας και τον τίτλο της ίδιας λίστας.

5. Φόρμα για πρόσθεση νέου αντικειμένου

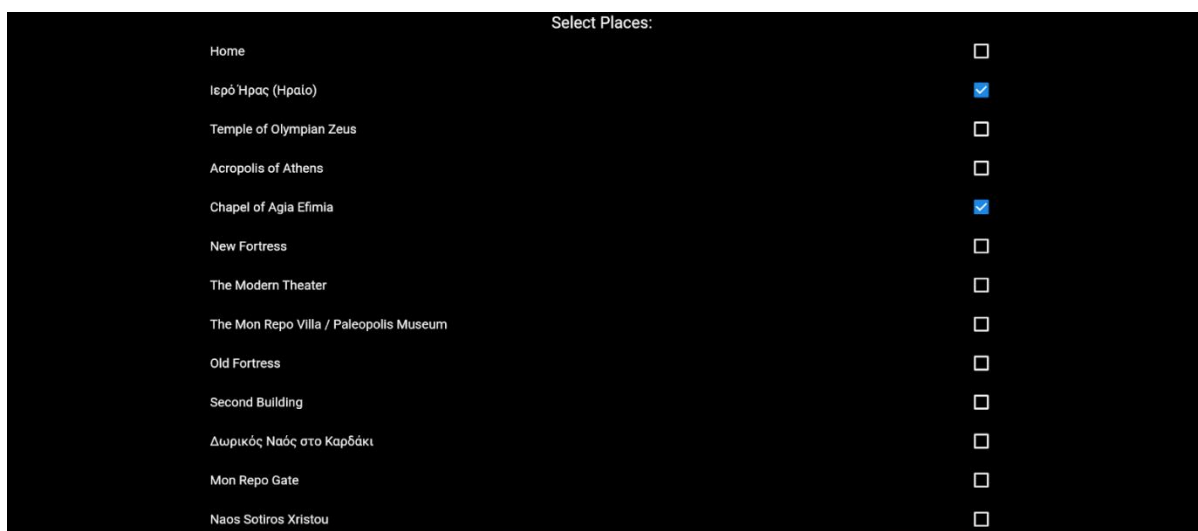
Το Page `OnAddButtonClick` θα προσφέρει στον χρήστη μια φόρμα για να μπορεί να βάζει νέα αντικείμενα στις βάσεις δεδομένων και κατά συνέπεια στις λίστες της ιστοσελίδας και στην εφαρμογή. Όπως αναφέρθηκε και προηγουμένως σαν παραμέτρους παίρνει μια λίστα με τα αντικείμενα από την “δεξιά” λίστα. Έχει ξανααναφερθεί ότι οι λίστες από τα αριστερά κρατάνε references για αντικείμενα από τις δεξιές τους λίστες. Άρα, στην φόρμα αυτή απαιτείται ο προγραμματιστής να έχει την λίστα με τα αντικείμενα από την δεξιά πλευρά για να μπορεί να επιλέξει ο χρήστης σε ποια από αυτά να κάνει refer το νέο αντικείμενο που θα φτιαχτεί. Επίσης, απαιτείται και το όνομα της λίστας στην οποία θα δημιουργηθεί το αντικείμενο.

Εφόσον, ο constructor θέσει αυτές τις μεταβλητές, τότε το flutter θα καλέσει την συνάρτηση `createState` την οποία έχουμε κάνει `override`. Σε αυτή την συνάρτηση θα γίνουν όλες οι υπόλοιπες αρχικοποιήσεις που χρειάζονται. Συγκεκριμένα, πρώτα θα αρχικοποιηθεί μια λίστα με `booleans` που θα κρατά ποια αντικείμενα έχουν επιλεγθεί για `reference`. Στην συνέχεια, αρχικοποιείται μια λίστα με τα `controllers` των `input fields` που θα χρησιμοποιηθούν στην φόρμα. Τέλος, αρχικοποιείται η μεταβλητή `dropDownValue` που περιέχει το αντικείμενο που θα εμφανιστεί σαν `default` στο `dropdown widget`. Όλες αυτές οι μεταβλητές θα μπουν σε ένα νέο `instance` της `_OnAddButtonClickState`, σαν παράμετροι και το `instance` θα επιστραφεί από την `createState`, ώστε το flutter στην συνέχεια, με την σειρά του, να το “εκτελέσει”.

Τώρα, στην `OnAddButtonClickState` αφού αρχικοποιηθούν οι μεταβλητές που αναφέρθηκαν προηγουμένως στην μέθοδο `build` δημιουργείται η φόρμα αναλόγως με τον τύπο του αντικειμένου που ο προγραμματιστής θέλει να φτιάξει. Ο τύπος του αντικειμένου φαίνεται από την μεταβλητή `categoryTitle`. Εντός της `build`, υπάρχει μια λίστα για να αποθηκευτούν όλα τα `widget` της φόρμας. Σε όλες τις περιπτώσεις, το πρώτο `widget` της λίστας είναι πάντα ένα `textField` με τον τίτλο του επικείμενου αντικειμένου. Στην συνέχεια, τα υπόλοιπα αντικείμενα αλλάζουν δυναμικά ανάλογα με τον τύπο του δημιουργηθέντος αντικειμένου, όπως αναφέρθηκε και νωρίτερα. Αν η φόρμα είναι για `route` αντικείμενο τότε περιέχει μια λίστα με `CheckBoxListTile widgets` που παρουσιάζουν τα `places` που θα περιέχει το `route`. Εκεί στην αριστερή πλευρά του `widget` βρίσκεται ο τίτλος του

αντικειμένου, ενώ στην δεξιά ένα checkbox που δείχνει αν το συγκεκριμένο αντικείμενο πρόκειται να προστεθεί στο route ή όχι.

Αν από την άλλη, το αντικείμενο που δημιουργείται είναι place, τότε στην λίστα προστίθενται δύο παρόμοια textFields για τις συντεταγμένες. Και τα δύο αυτά πεδία ελέγχουν τις τιμές που τους παρέχει ο χρήστης για να σιγουρευτούν ότι είναι αριθμός (άρα δεν περιέχει γράμματα ή ειδικούς χαρακτήρες) καθώς και ότι η τιμή είναι εντός του αποδεκτού εύρους τιμών για την εκάστοτε συντεταγμένη. Στην συνέχεια, στην λίστα



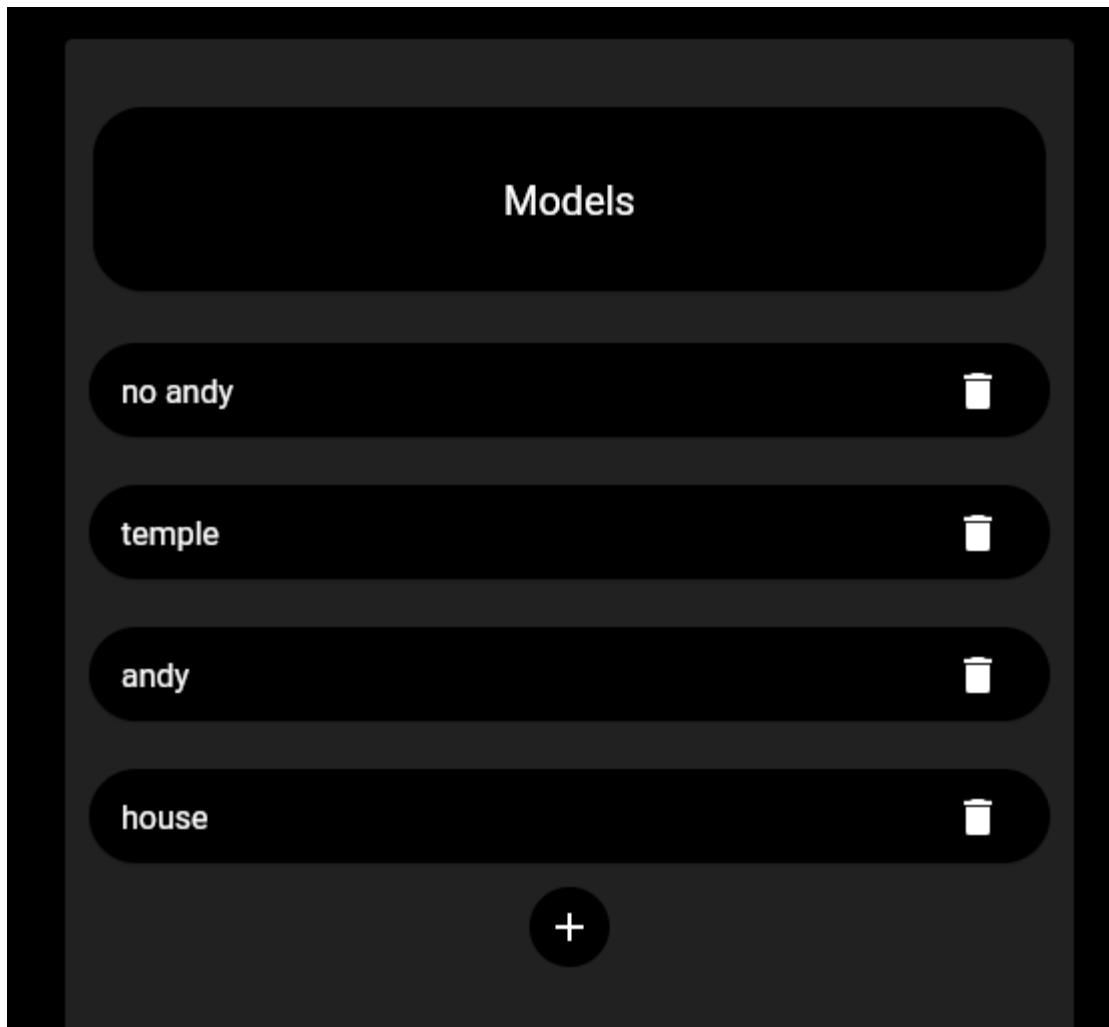
Εικόνα 41: "Add place" page

προστίθεται ένα textField το οποίο παίρνει σαν τιμή ένα θετικό αριθμό για να χρησιμοποιηθεί ως το AOE(Area Of Effect) του αντικειμένου. Τελικώς, άλλα δύο widget προστίθενται στο τέλος της λίστας. Το πρώτο είναι ένα απλό text widget που χρησιμοποιείται σαν τίτλος για το επόμενο widget, δηλαδή για το dropdown με τα holograms. Το dropdown δημιουργείται από την συνάρτηση getDropDown. Αυτή με την σειρά της επιστρέφει ένα widget τύπου DropdownButton που παίρνει σαν αρχική τιμή την _dropDownValue που αρχικοποιήθηκε στον constructor της OnAddButtonClickState. Η λίστα με τις επιλογές φτιάχνεται μέσω της μετατροπής των τιμών της λίστας _items σε μια λίστα με widget DropdownMenuItem τα οποία το καθένα παίρνει από μια τιμή από την λίστα _items και χρησιμοποιεί σαν child ένα text widget που παρουσιάζει αυτή την τιμή.

Στην περίπτωση που το αντικείμενο που δημιουργείται πρόκειται να είναι hologram τότε μετά τον τίτλο, τα επόμενα έξι widget που θα προστεθούν στην λίστα χρησιμοποιούν την συνάρτηση getEditText για να φτιάξουν ένα TextFormField αντικείμενο. Συγκεκριμένα, η συνάρτηση δέχεται σαν παραμέτρους ένα τίτλο, για να μπει σαν labelText στο widget, μια συνάρτηση validator, που θα ελέγχει την τιμή που έχει βάλει ο χρήστης και στην συνέχεια είτε θα "πετάει" error ή αν είναι null θα συνεχίσει την εκτέλεση. Ακόμη, η συνάρτηση δέχεται ως παραμέτρους ένα TextInputType για να υποδεικνύει στον χρήστη, τί τιμές είναι αποδεκτές στο συγκεκριμένο πεδίο και τον controller που θα κρατά

την τιμή, που θα αναγράφεται στο πεδίο και για να εκτελεί την συνάρτηση του validator. Τα έξι αυτά widget θα δείχνουν το Image URL, το description, μια ερώτηση με δύο απαντήσεις και ένα web URL. Όλα αυτά είναι προαιρετικά εκτός από τον τίτλο και το description. Αν υπάρχει ερώτηση, δηλαδή αν ο controller του έχει τιμή, τότε ο validator ελέγχει ότι υπάρχει και απάντηση, για να μπορεί να διαλέξει ο χρήστης ενώ αντίστοιχα στο πεδίο της απάντησης γίνεται έλεγχος από τον validator ότι υπάρχει ερώτηση. Στον τίτλο και στην περιγραφή ο validator ελέγχει ότι ο χρήστης έχει βάλει τιμή στο πεδίο. Τα τελευταία widget που προστίθενται σε αυτήν την περίπτωση είναι ένα dropdown, για να διαλέξει ο χρήστης το μοντέλο του αντικειμένου και ο τίτλος του dropdown, για να αναγνωρίζει ο χρήστης τι πρόκειται να διαλέξει. Το dropdown δημιουργείται από την getDropDown και λειτουργεί ακριβώς με τον ίδιο τρόπο όπως είχε αναφερθεί και προηγουμένως.

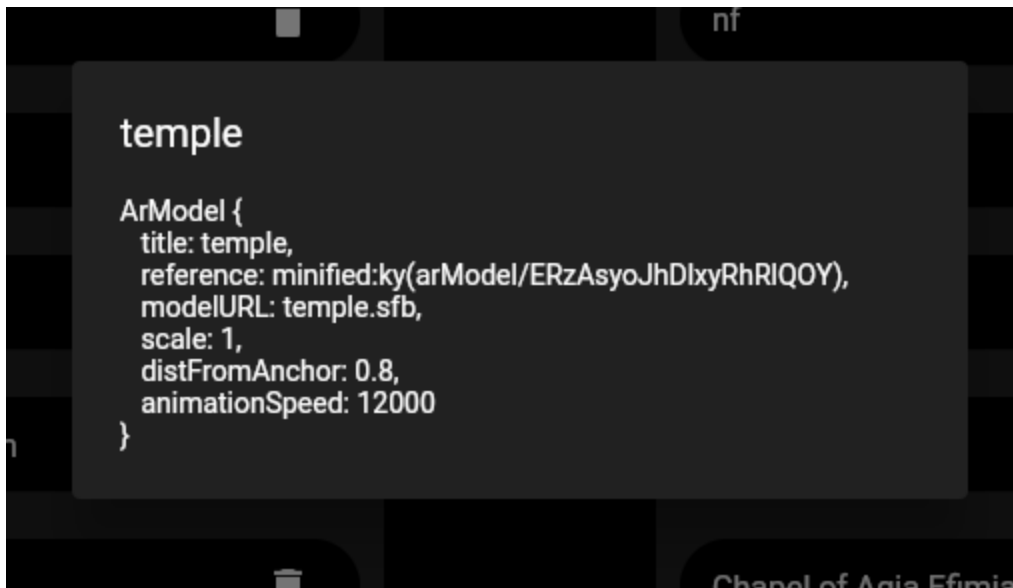
Η τελευταία φόρμα που μπορεί να δημιουργηθεί βασίζεται στα αντικείμενα τύπου Models. Η φόρμα προσθέτει στην λίστα με τα widget, τρία widget που χρησιμοποιούν την getEditText. Στην πρώτη από τις τρεις, ο χρήστης πρέπει να βάλει το scale value του αντικειμένου. Η τιμή αυτή πρέπει να είναι αναγκαστικά αριθμός, πράγμα που ελέγχει ο validator. Το δεύτερο πεδίο που προστίθεται δείχνει την απόσταση του μοντέλου από την άγκυρα, πράγμα που ο validator επίσης ελέγχει ότι είναι αριθμός. Το τρίτο πεδίο δέχεται την ταχύτητα του animation της περιστροφής του μοντέλου που επίσης είναι αριθμός και ελέγχεται. Στην συνέχεια, ακολουθούν τέσσερα widget για να μπορεί ο χρήστης να επιλέξει το αρχείο του μοντέλου ή για προσθέσει ένα καινούργιο στην βάση. Από αυτά τα τέσσερα, το πρώτο είναι ένα text widget με τίτλο “select file” για να αντιλαμβάνεται ο χρήστης τι πρέπει να κάνει σε αυτό το βήμα. Το δεύτερο, χρησιμοποιεί το getDropDown ώστε όπως και έχει περιγραφεί και προηγουμένως να δείξει στον χρήστη τις πιθανές επιλογές. Το επόμενο widget είναι ένα widget με τον τίτλο “OR” για να καταλάβει ο χρήστης ότι μπορεί ή να επιλέξει ένα ήδη υπάρχον αρχείο ή να προσθέσει ένα δικό του. Το τελευταίο widget είναι ένα MaterialButton που περιέχει ένα text widget με τίτλο “Upload”. Όταν ο χρήστης πατήσει το κουμπί και καλέσει την onPressed, τότε χρησιμοποιείται η κλάση DatabaseService για να διαλέξει ο χρήστης ένα αρχείο και στην συνέχεια ανανεώνει τις επιλογές του drop down που βρίσκονται από πάνω προσθέτοντάς το νέο αρχείο σαν επιλογή και ταυτόχρονα διαλέγοντας το αυτομάτως, ως το τρέχων επιλεγμένο.



Εικόνα 42: Models list

a. Ενέργειες χρήση στη λίστα

Στη λίστα ο διαχειριστής των διαδρομών θα μπορεί να κάνει τις εξής τρεις ενέργειες. Πρώτον να βλέπει τα υπάρχοντα μοντέλα που υπάρχουν καθώς και τα αντίστοιχα χαρακτηριστικά τους. Δεύτερον, να προσθέτει νέα αντικείμενα στην λίστα και τρίτον να διαγράφει αντικείμενα που δεν χρειάζεται πλέον. Παραπάνω είναι εμφανές το πώς βλέπει ο χρήστης την λίστα με τα υπάρχοντα μοντέλα. Πατώντας πάνω στο όνομα του μοντέλου εμφανίζεται ένας διάλογος που περιέχει τα χαρακτηριστικά του μοντέλου σε μορφή JSON.



Εικόνα 43: Showing items data

b. Εμφάνιση Δεδομένων

Εδώ στην κορυφή, φαίνεται το όνομα του μοντέλου και από κάτω βρίσκεται το JSON. Η JSON απεικόνιση περιέχει τα εξής έξι χαρακτηριστικά. Το τίτλο του μοντέλου (το ίδιο με αυτό που εμφανίζεται από πάνω), ένα reference που δείχνει στο document του μοντέλου μέσα στη βάση. Ένα URI με το όνομα του αρχείου που περιέχει το τρισδιάστατο αντικείμενο στο storage της βάσης. Ακόμα, περιέχονται και άλλα τρία χαρακτηριστικά που σχετίζονται με το πώς θα εμφανιστεί το τρισδιάστατο μοντέλο στον χρήστη. Πιο συγκεκριμένα, το scale αλλάζει το μέγεθος του αντικειμένου όπως θα φαίνεται μέσα από την οθόνη. Μαζί με το τρισδιάστατο μοντέλο εμφανίζεται και μια δισδιάστατη οθόνη με πληροφορίες σχετικές με το σημείο ενδιαφέροντος, η μετρική dist from anchor δείχνει την απόσταση μεταξύ του κέντρου του μοντέλου και του κέντρου της οθόνης στον x άξονα. Αν η τιμή είναι θετική τότε το μοντέλο εμφανίζεται αριστερά. Αν η τιμή είναι αρνητική, τότε το μοντέλο εμφανίζεται στα δεξιά της οθόνης. Τέλος, το animation speed ρυθμίζει την ταχύτητα περιστροφής του μοντέλου γύρω από τον κεντρικό του άξονα με σκοπό να δώσει στον χρήστη να έρθει σε επαφή με κάτι πιο “ζωντανό”.

Αποτελέσματα

Όλα τα σύγχρονα λογισμικά εξαρτώνται από την εμπειρία που θα έχει ο χρήστης κατά τη διάρκεια της χρήσης τους. Ένα κακά σχεδιασμένο σύστημα, ο χρήστης θα προσπαθήσει να το αποφύγει. Ιδίως σε εμπορικές εφαρμογές, η εμπειρία του χρήστη μπορεί να αποτελέσει τον κύριο παράγοντα, για τον αν θα πετύχει ή αποτύχει μια εφαρμογή.

Ακόμα και στην εφαρμογή που αναπτύχθηκε στα πλαίσια αυτής της έρευνας, που πρωτίστως έχει σαν κύριο σκοπό την απόδειξη ότι ένα τέτοιο σύστημα επαυξημένης πραγματικότητας είναι εφικτό να πραγματοποιηθεί, είναι σημαντικό να μελετηθεί και η άποψη των χρηστών σχετικά με την εμπειρία τους κατά την χρήση της εφαρμογής, με σκοπό τη βελτίωση μελλοντικών, παρόμοιων, εφαρμογών και να κριθεί τελικά, αν μια τέτοια εφαρμογή θα ήταν εφικτό να κυκλοφορήσει εμπορικά.

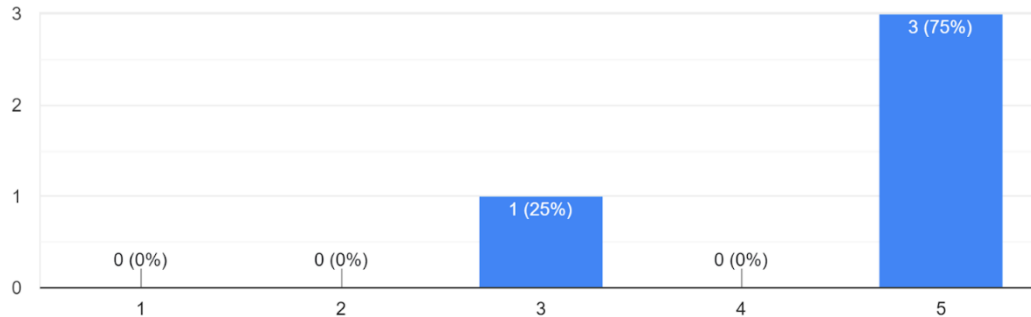
Η ευχρηστία μιας εφαρμογής είναι κάτι το υποκειμενικό. Ο καθένας το βιώνει διαφορετικά. Δεν υπάρχουν ξεκάθαροι τρόποι μέτρησης της. Ωστόσο είναι σημαντικό να μπορεί να γίνει σύγκριση διαφορετικών συστημάτων σε σύγκριση με την ευχρηστία τους. Για αυτό τον λόγο έχουν δημιουργηθεί κάποια γενικά συστήματα που προσπαθούν να δώσουν μια γενική εικόνα περί του θέματος. Ένα από αυτά τα συστήματα είναι και το “System Usability Scale” (Brooke, 1995). Το σύστημα αυτό αξιολογεί την ευχρηστία που βίωσε ο χρήστης με βάση δέκα μετρήσεις, με την κάθε μια να έχουν εύρος από το ένα έως το πέντε. Το ένα είναι, πολύ λίγο και οι πέντε, παρά πολύ.

Το δείγμα των χρηστών το οποίο χρησιμοποιήθηκε είναι 4 άτομα, μη σχετικά με τον προγραμματισμό και την ανάπτυξη συστημάτων και εφαρμογών σε κινητές συσκευές. Οι χρήστες αυτοί βρίσκονται εντός του ηλικιακού εύρους 20 με 25 χρονών. Η εξοικείωση τους με εφαρμογές android ποικίλη από ελάχιστη μέχρι άνω του μετρίου. Αντίστοιχο είναι και το επίπεδο εξοικείωσης τους με εφαρμογές επαυξημένης πραγματικότητας. Η δειγματοληψία πραγματοποιήθηκε με τη χρήση δύο διαφορετικών διαδρομών εντός της Αθήνας. Από τους τέσσερις χρήστες οι 3 μπόρεσαν να τρέξουν την εφαρμογή στο κινητό τους ενώ ο τέταρτος χρησιμοποίησε την εφαρμογή από συσκευή τρίτου προσώπου.

1. Νομίζω ότι θα ήθελα να χρησιμοποιώ το σύστημα συχνά;

I think that I would like to use the system frequently.

4 responses



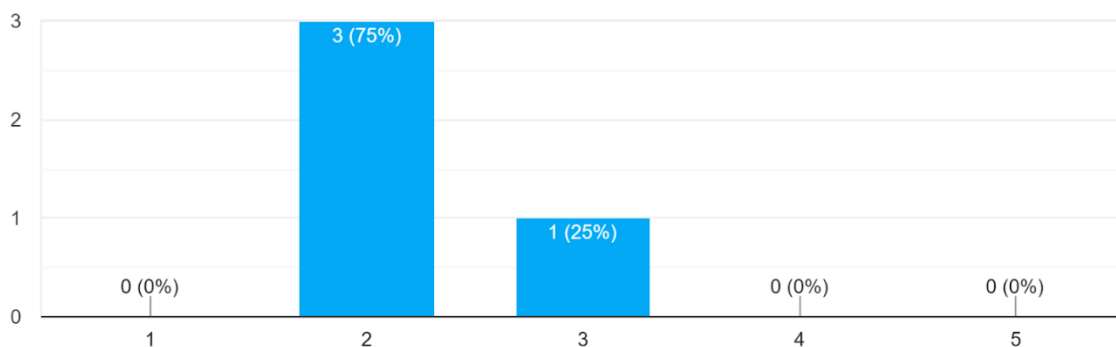
Διάγραμμα 1: Νομίζω ότι θα ήθελα να χρησιμοποιώ το σύστημα συχνά;

Σε αυτή την πρώτη ερώτηση, ο χρήστης καλείται να απαντήσει αν θα χρησιμοποιούσε συχνά την εφαρμογή. Παρατηρείται ότι το 75% θα το χρησιμοποιούσε πολύ συχνά ενώ το 25% θα το χρησιμοποιούσε, έτσι και έτσι.

2. Βρήκατε το σύστημα πολύπλοκο χωρίς αιτία;

I found the system unnecessarily complex.

4 responses



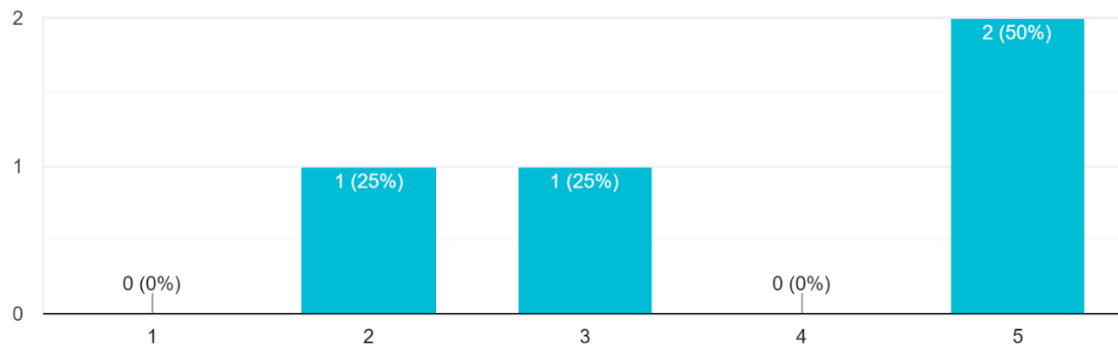
Διάγραμμα 2: Βρήκατε το σύστημα πολύπλοκο χωρίς αιτία;

Σε αυτή την ερώτηση οι γνώμες είναι κυρίως θετικές, καθώς βρέθηκε ότι το 75% πιστεύει ότι δεν είναι πολύπλοκο το σύστημα.

3. Νομίζετε ότι το σύστημα είναι εύκολο προς την χρήση του;

I though the system was easy to use.

4 responses



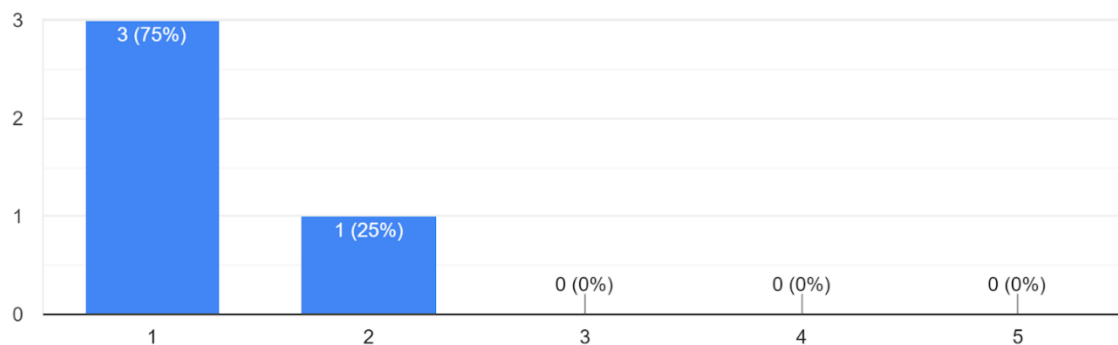
Διάγραμμα 3: Νομίζετε ότι το σύστημα είναι εύκολο προς την χρήση του;

Όπως φαίνεται σε αυτήν την ερώτηση, οι μισοί χρήστες βρήκαν πολύ εύκολη την εφαρμογή ως προς την χρήση της, ενώ οι υπόλοιποι την βρήκαν από έτσι και έτσι έως λίγο δύσκολη.

4. Νομίζετε ότι θα χρειαζόσασταν την υποστήριξη κάποιου τεχνικού για να χρησιμοποιήσετε την εφαρμογή.

I think that I would need the support of a technical person to be able to use the system.

4 responses



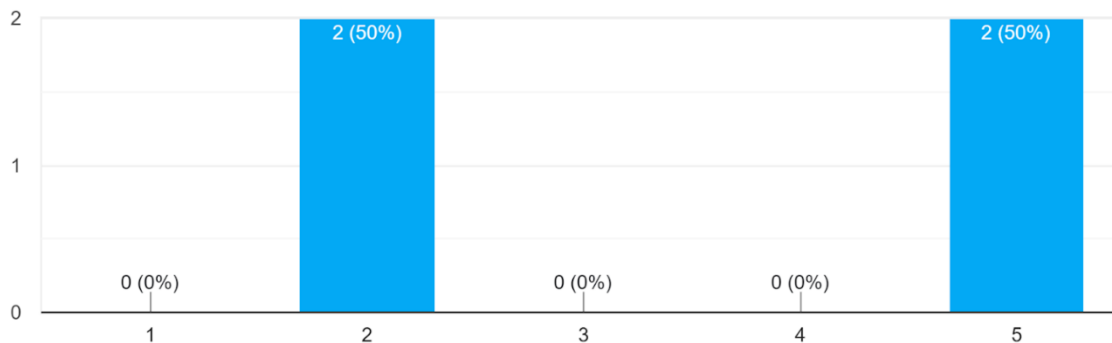
Διάγραμμα 4: Νομίζετε ότι θα χρειαζόσασταν την υποστήριξη κάποιου τεχνικού για να χρησιμοποιήσετε την εφαρμογή.

Σε αυτή την ερώτηση, οι χρήστες απάντησαν ότι θα χρειαζόντουσαν την βοήθεια τεχνικού από λίγο έως καθόλου.

5. Βρήκατε τις διάφορες λειτουργίες του συστήματος, καλά ενσωματωμένες;

I found the various functions in this system were well integrated.

4 responses



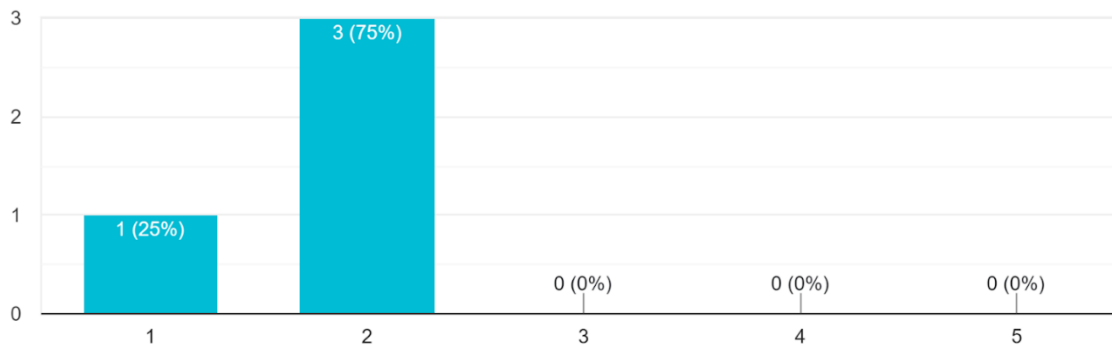
Διάγραμμα 5: Βρήκατε τις διάφορες λειτουργίες του συστήματος, καλά ενσωματωμένες;

Εδώ υπάρχει μια διαφωνία ανάμεσα στους χρήστες, οι μισοί πιστεύουν ότι οι λειτουργίες είναι πολύ καλά ενσωματωμένες ενώ οι άλλοι μισοί, πιστεύουν ότι δεν είναι καλά ενσωματωμένες.

6. Νομίζεται ότι υπήρχε υπερβολική ασυνέπεια σε αυτό το σύστημα;

I though there was too much inconsistency in this system.

4 responses



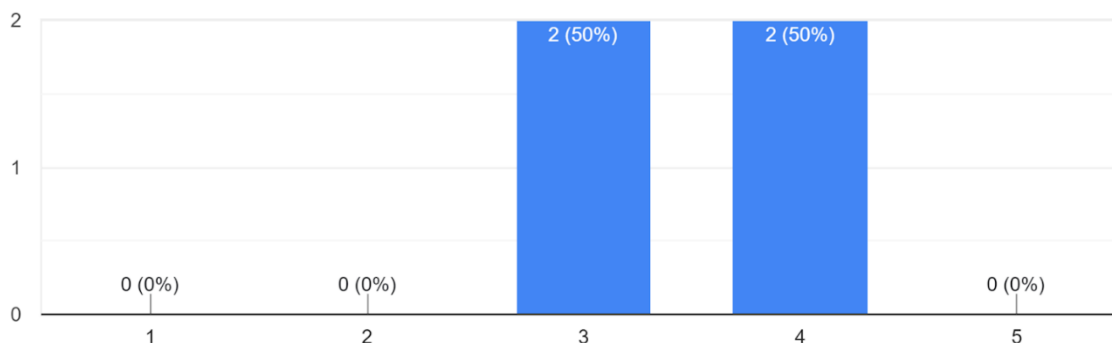
Διάγραμμα 6: Νομίζεται ότι υπήρχε υπερβολική ασυνέπεια σε αυτό το σύστημα;

Οι απαντήσεις σε αυτή την περίπτωση είναι θετικές, οπότε οι χρήστες συμφωνούν ότι δεν υπήρχαν ασυνέπειες στο σύστημα.

7. Πιστεύεται ότι οι περισσότεροι άνθρωποι θα μάθαιναν να χρησιμοποιούν εύκολα αυτό το σύστημα;

I would imagine that most people would learn to use this system very quickly.

4 responses



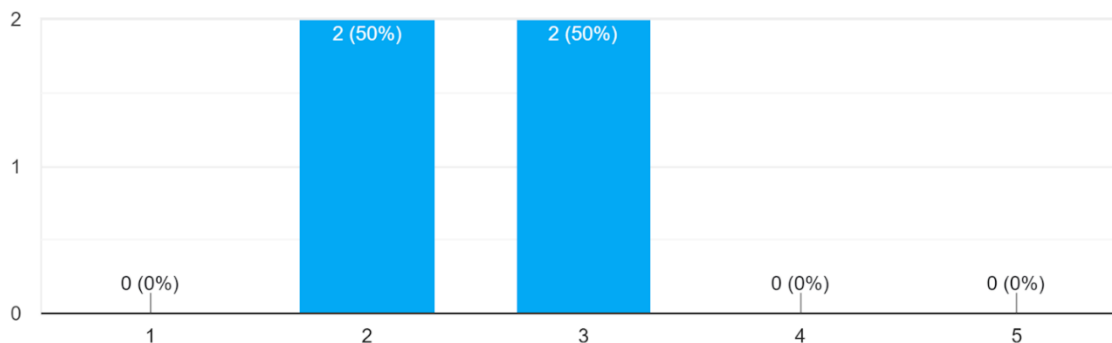
Διάγραμμα 7: Πιστεύεται ότι οι περισσότεροι άνθρωποι θα μάθαιναν να χρησιμοποιούν εύκολα αυτό το σύστημα;

Οι μισοί χρήστες πιστεύουν ότι θα ήταν εύκολο να μάθει κάποιος το σύστημα και οι υπόλοιποι ότι δεν θα ήταν ούτε εύκολο ούτε δύσκολο.

8. Βρήκατε το σύστημα πολύ δυσκίνητο στην χρήση;

I found the system very cumbersome to use.

4 responses



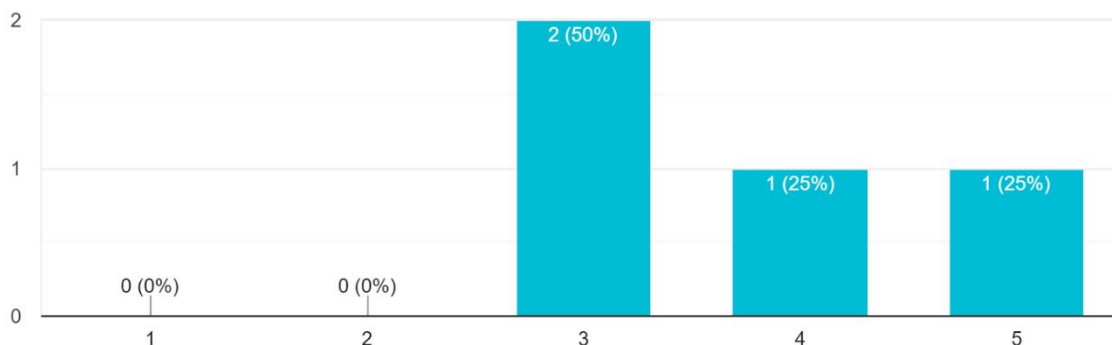
Διάγραμμα 8: Βρήκατε το σύστημα πολύ δυσκίνητο στην χρήση;

Το σύστημα δεν ήταν πολύ δυσκίνητο αφού πολλοί είπαν ότι διαφωνούν ή ότι ούτε διαφωνούν ούτε συμφωνούν.

9. Νιώσατε πολύ σίγουρος όταν χρησιμοποιήσατε το σύστημα;

I felt very confident using the system.

4 responses



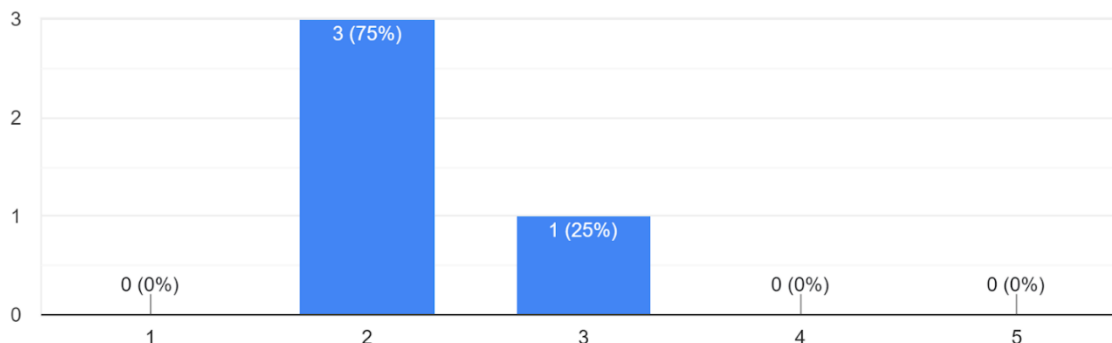
Διάγραμμα 9: Νιώσατε πολύ σίγουρος όταν χρησιμοποιήσατε το σύστημα;

Το 50% του δείγματος, ούτε συμφώνησαν ούτε διαφώνησαν με την ερώτηση που τους τέθηκε ενώ οι άλλοι μισοί ένιωσαν σίγουροι.

10. Χρειαστήκατε να μάθετε πάρα πολλά για να μπορέσετε να συνεχίσετε με αυτό το σύστημα;

I needed to learn a lot of things before I could get going with this system.

4 responses



Διάγραμμα 10: Χρειαστήκατε να μάθετε πάρα πολλά για να μπορέσετε να συνεχίσετε με αυτό το σύστημα;

Το 75% των απαντήσεων δείχνει ότι δεν χρειάστηκε να μάθουν πολλά για να συνεχίσουν να χρησιμοποιούν το σύστημα.

Συμπερασματικά, αποδείχτηκε ότι ακόμη και σε μια εφαρμογή, όπου πρωτίστως προσπαθεί να αποδείξει ότι ένα τέτοιο σύστημα επαυξημένης πραγματικότητας για τουριστική χρήση είναι δυνατό να υλοποιηθεί και δεν έχει δώσει ιδιαίτερη προσοχή στην ευχρηστία του χρήστη, ακόμη και έτσι είναι αρκετά εύκολο για τον μέσο χρήστη να αλληλεπιδράσει μέσω αυτής.

Βιβλιογραφία

- Andri , C., Alkawaz, M. H., & Sallow, A. B. (2018). Adoption of Mobile Augmented Reality as a Campus Tour. *International Journal of Engineering & Technology*, 64-69. Ανάκτηση από International Journal of Engineering & Technology.
- Apple. (2022). *Apple*. Ανάκτηση από Apple: <https://www.apple.com/maps/>
- Brooke, J. (1995, November). SUS: A quick and dirty usability scale. *ResearchGate*.
- Ceci, L. (2022, July 25). *statista*. Ανάκτηση από statista: <https://www.statista.com/statistics/865413/most-popular-us-mapping-apps-ranked-by-audience/>
- Ceci, L. (2022, January 18). *statista*. Ανάκτηση από statista: <https://www.statista.com/statistics/865419/most-popular-us-mapping-apps-ranked-by-reach/>
- developers. (χ.χ.). *developers*. Ανάκτηση από developers: <https://developer.android.com/reference/android/view/View>
- developers. (χ.χ.). *developers*. Ανάκτηση από developers: <https://developer.android.com/reference/android/widget/LinearLayout>
- developers. (χ.χ.). *developers*. Ανάκτηση από developers: <https://developer.android.com/develop/ui/views/layout/relative>
- developers. (χ.χ.). *developers*. Ανάκτηση από developers: <https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout>
- developers. (χ.χ.). *developers*. Ανάκτηση από developers: <https://developer.android.com/guide/fragments>
- developers. (χ.χ.). *developers*. Ανάκτηση από developers: <https://developer.android.com/develop/ui/views/components/menus>

- developers. (χ.χ.). *developers*. Ανάκτηση από developers:
<https://developer.android.com/reference/com/google/android/material/bottomnavigation/BottomNavigationView>
- Flutter. (2022, July). *Flutter*. Ανάκτηση από Flutter:
<https://docs.flutter.dev/development/ui/widgets-intro>
- Koo, S., Kim, J., Kim, C., Kim, J., & Cha, H. (2020). Development of an Augmented Reality Tour Guide for a Cultural Heritage Site. *Journal on Computing and Cultural Heritage*, 1-24.
- Media Art Net. (χ.χ.). *Media Art Net*. Ανάκτηση από Media Art Net:
<http://www.medienkunstnetz.de/works/sensorama/>
- Park, C., & Kim, J.-B. (2011). Development of Mobile AR Tour Application for the National Palace Museum of Korea. *Virtual and Mixed Reality-New Trends*, 55-60.
- similarweb. (2022, September 17). *similarweb*. Ανάκτηση από similarweb:
<https://www.similarweb.com/apps/top/google/store-rank/gr/maps-navigation/top-free/>
- stack overflow. (2018). *stack overflow*. Ανάκτηση από stack overflow:
<https://stackoverflow.com/questions/53667793/what-are-controllers-in-flutter#:~:text=In%20flutter%2C%20controllers%20are%20a,pattern%20stuff%20and%20increase%20performances>.
- Vlahakis, V., Ioannidis, M., Tsotros, M., Gounaris, M., Stricker, D., Gleue, T., . . . Almeida, L. (2002, November 07). *IEEE Xplore*. Ανάκτηση από IEEE Xplore: <https://ieeexplore.ieee.org/document/1028726>
- Wikipedia. (2022, September 15). *Wikipedia*. Ανάκτηση από Wikipedia:
https://en.wikipedia.org/wiki/Apple_Maps#:~:text=The%20mapping%20service%20was%20released,see%20also%20%C2%A7Early%20i
 naccuracy).

Διαγράμματα

Διάγραμμα 1: Νομίζω ότι θα ήθελα να χρησιμοποιώ το σύστημα συχνά;	72
Διάγραμμα 2: Βρήκατε το σύστημα πολύπλοκο χωρίς αιτία;	72
Διάγραμμα 3: Νομίζετε ότι το σύστημα είναι εύκολο προς την χρήση του;	73
Διάγραμμα 4: Νομίζετε ότι θα χρειαζόσασταν την υποστήριξη κάποιου τεχνικού για να χρησιμοποιήσετε την εφαρμογή.	73
Διάγραμμα 5: Βρήκατε τις διάφορες λειτουργίες του συστήματος, καλά ενσωματωμένες;	74
Διάγραμμα 6: Νομίζεται ότι υπήρχε υπερβολική ασυνέπεια σε αυτό το σύστημα;	74
Διάγραμμα 7: Πιστεύεται ότι οι περισσότεροι άνθρωποι θα μάθαιναν να χρησιμοποιούν εύκολα αυτό το σύστημα;	75
Διάγραμμα 8: Βρήκατε το σύστημα πολύ δυσκίνητο στην χρήση;	75
Διάγραμμα 9: Νιώσατε πολύ σίγουρος όταν χρησιμοποιήσατε το σύστημα;	76
Διάγραμμα 10: Χρειαστήκατε να μάθετε πάρα πολλά για να μπορέσετε να συνεχίσετε με αυτό το σύστημα;	76

Πίνακες

Πίνακας 1: Future Builder	53
Πίνακας 2: Using provider value to choose a page	54
Πίνακας 3: Get the logged in user	54
Πίνακας 4: SignIn	55
Πίνακας 5: Sign Out	56
Πίνακας 6: Form's global key	57
Πίνακας 7: Text Form Field	58
Πίνακας 8: Making Validation using the validators	58
Πίνακας 9: Login Button	59
Πίνακας 10: Lists containing the data model	60
Πίνακας 11: AppBar	61
Πίνακας 12: Example of applying changes when data are changed	63

Εικόνες

Εικόνα 1: Wam	8
Εικόνα 2: Senditur	9
Εικόνα 3: Night Sky	9

Εικόνα 4: Horizon Explorer.....	10
Εικόνα 5: Leading mapping apps in the United States in 2021, by downloads	10
Εικόνα 6: Google Maps	11
Εικόνα 7: Google Maps 1	12
Εικόνα 8: Google Maps 2	12
Εικόνα 9: Google Maps 3	13
Εικόνα 10: Apple Maps	14
Εικόνα 11: Most popular mapping apps in the United States as of April 2018, by reach	14
Εικόνα 12: Maps 1.....	15
Εικόνα 13: Maps 2.....	15
Εικόνα 14: CULTUREWhisper	16
Εικόνα 15: Maps 3.....	16
Εικόνα 16: Maps 4.....	17
Εικόνα 17: HEREWeGo	18
Εικόνα 18: Top Apps Ranking	18
Εικόνα 19: HereWeGo 2.....	19
Εικόνα 20: HereWeGo 1.....	19
Εικόνα 21: HereWeGo 3.....	19
Εικόνα 22: HereWeGo 4.....	19
Εικόνα 23: Waze 1	20
Εικόνα 24: Waze 2	20
Εικόνα 25: Waze 3	21
Εικόνα 26: Waze 4	21
Εικόνα 27: Waze 5	21
Εικόνα 28: Designing complex UI using Android Constraint Layout Code magic Blog	24
Εικόνα 29: The Activity Lifecycle	26
Εικόνα 30: Android Kotlin Basics- RecyclerView and Adapters- Alex Dunn	28
Εικόνα 32: Maps.....	48
Εικόνα 31: Live View	48
Εικόνα 33: Menu	48
Εικόνα 34: Διάγραμμα Ροής Δεδομένων	52
Εικόνα 35: Log In Page	56
Εικόνα 36: Autofill functionality.....	57
Εικόνα 37: Log In button	59
Εικόνα 38: Main page.....	61
Εικόνα 39: AppBar	62
Εικόνα 40: Main Page	63
Εικόνα 41: "Add place" page	67
Εικόνα 42: Models list	69
Εικόνα 43: Showing items data	70

Παράρτημα

GeoAr App Questionnaire

Feedback for the user's experience using the "System Usability Scale" template.

 antoniaaalexiou@gmail.com (not shared) [Switch accounts](#)



*Required

I think that I would like to use the system frequently. *

	1	2	3	4	5	
Strongly disagree	☐	☐	☐	☐	☐	Strongly agree

I found the system unnecessarily complex. *

	1	2	3	4	5	
Strongly disagree	☐	☐	☐	☐	☐	Strongly agree

I thought the system was easy to use. *

	1	2	3	4	5	
Strongly disagree	☐	☐	☐	☐	☐	Strongly agree

I think that I would need the support of a technical person to be able to use the system. *

	1	2	3	4	5	
Strongly disagree	☐	☐	☐	☐	☐	Strongly agree

I found the various functions in this system were well integrated. *

	1	2	3	4	5	
Strongly disagree	☐	☐	☐	☐	☐	Strongly agree

I though there was too much inconsistency in this system. *

	1	2	3	4	5	
Strongly disagree	☐	☐	☐	☐	☐	Strongly agree

I would imagine that most people would learn to use this system very quickly. *

	1	2	3	4	5	
Strongly disagree	☐	☐	☐	☐	☐	Strongly agree

I found the system very cumbersome to use. *

	1	2	3	4	5	
Strongly disagree	☐	☐	☐	☐	☐	Strongly agree

I felt very confident using the system. *

	1	2	3	4	5	
Strongly disagree	☐	☐	☐	☐	☐	Strongly agree

I needed to learn a lot of things before I could get going with this system. *

	1	2	3	4	5	
Strongly disagree	☐	☐	☐	☐	☐	Strongly agree

Submit

Page 1 of 1

[Clear form](#)