ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών
«Προηγμένα Συστήματα Πληροφορικής – Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Εξατομικευμένη Ηλεκτρονική Περιήγηση με φορητές συσκευές.** **Personalized Virtual Tour using mobile devices (AR & VR)**
Ονοματεπώνυμο Φοιτητή	**Γεώργιος Σίμος**
Πατρώνυμο	**Ιωάννης Σίμος**
Αριθμός Μητρώου	**ΜΠΣΠ 18021**
Επιβλέπων	**Ευάγγελος Σακκόπουλος, Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης **Νοέμβριος 2022**

Τριμελής Εξεταστική Επιτροπή

Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα
Ευάγγελος Σακκόπουλος Αναπληρωτής Καθηγητής	Ευθύμιος Αλέπης Αναπληρωτής Καθηγητής	Διονύσιος Σωτηρόπουλος Επίκουρος Καθηγητής

Ευχαριστίες

Η παρούσα διπλωματική εργασία σηματοδοτεί το κλείσιμο ενός καθοριστικού κυκλου για μένα, στα πλαίσια της οποίας είχα την χαρά και την τιμήν να γνωρίσω και να συνεργαστώ με αρκετα σημαντικά και ευχάριστα ατομα.

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου Ευάγγελο Σακκόπουλο, για την καθοδήγηση που μου προσέφερε και το χρόνο που Διέθεσε για τις πολύ χρήσιμες συμβουλές και οδηγίες που μου έδωσε για την ολοκλήρωση της
πτυχιακής μου εργασίας. Στο ίδιο πλαίσιο ευγνωμοσύνης, θα ήθελα να ευχαριστήσω όλους τους καθηγητές του τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς για τη συμβολή τους στην επιστημονική και τεχνολογική μου συγκρότηση στα χρόνια της φοίτησής μου στο Τμήμα.

Τέλος, ένα μεγάλο ευχαριστώ στην οικογένεια και τους φίλους μου για την βοήθεια και την ηθική υποστήριξη σε όλο το διάστημα των σπουδών μου.

Περίληψη

Εδώ και αρκετες δεκαετιες επιστήμονες του τομέα της τεχνολογίας προσπαθουν να υλοποιήσουν την ιδέα του να συνδιαζει κανεις στην καθημερινότητα του τον πραγματικό με έναν εικονικό κόσμο.
Τεχνολογίες οπως η εικονική (VR) και επαυξημένη (AR) πραγματικότητα αποτελούν βασικό κομμάτι του Tech Industry, προσφατα μια απο τις μεγαλύτερες εταιρείες τεχνολογίας ξεκίνησε έναν αγώνα δρόμου δισεκατομμυρίων πάνω στην επένδυση αυτον τον τεχνολογιών.

Σκοπός της παρούσας διπλωματικής εργασίας είναι η έρευνα πάνω σε διαθέσιμα εργαλεία και συστήματα ανάπτυξης εφαρμογών Επαυξημένης Πραγματικότητας (Augmented Reality, AR), καθώς και η ανάπτυξη μιας πλήρως λειτουργικής εφαρμογής για κινητές συσκευές με χρήση της τεχνολογίας αυτής. Η εφαρμογή αποσκοπεί στο να βελτιώσει την εμπειρία ηλεκτρονικής περιήγησης σε σημεία ενδιαφέροντος του χρήστη με φορητές συσκευές.

Η υλοποίηση της εφαρμογής έγινε με React Native, ενα JavaScript framework που μας δίνει την δυνατότητα να μετατρέψουμε τον αρχικό κωδικα σε γλώσσα προγραμματισμού που υποστηρίζουν τα SDKs του Android και του iOS.

Τέλος, για την αποθήκευση και τον συγχρονισμό δεδομένων σε πραγματικό χρόνο χρησιμοποιούμε το Firebase, μια πλατφόρμα που αναπτύχθηκε από την Google για τη δημιουργία εφαρμογών για κινητές συσκευές.

Λέξεις κλειδιά: επαυξημένη πραγματικότητα, ηλεκτρονική περιήγηση, διεπαφή χρήστη, εγγενείς κινητές εφαρμογές, υβριδικές κινητές εφαρμογές

Abstract

For several decades, scientists in the field of technology have been trying to implement the idea of combining the real world with a virtual world in one's daily life. Virtual (VR) and augmented (AR) reality are a vital part of the Tech Industry; recently, one of the largest technology companies has started a race of billions of investments in this technology.

The purpose of this thesis is to research available tools and systems for the development of Augmented Reality (AR) applications, as well as the result of a fully functional application for mobile devices using this technology. The application aims to improve the user's online point of interest browsing experience with mobile devices. The implementation of the application was done with React Native, a JavaScript framework that enables us to convert the original code into a programming language that supports the Android and iOS SDKs.

Finally, for real-time data storage and synchronization, we use Firebase, a platform developed by Google for creating mobile applications.

Keywords: augmented reality, virtual tour, user interface, native mobile development, hybrid or cross - platform development

Πίνακας περιεχομένων

Πίνακας εικόνων

Κεφάλαιο 1 – Εισαγωγή

1.1 Γενικά

Τα τελευταία χρόνια παρατηρείται μια ραγδαία τεχνολογική ανάπτυξη σε όλους του τεχνολογικούς τομείς. Η τεχνολογία πλέον βρίσκεται παντού γύρω μας και συνήθως έχει την μορφή ενός υπερσύγχρονου κινητού τηλεφώνου ή ενός υπολογιστή που έχει μία οθόνη λεπτή σαν ένα φύλλο χαρτί. Αυτή η εξαιρετικά ταχεία ανάπτυξης έχει ως αποτέλεσμα σχεδόν ο κάθε άνθρωπος του πλανήτη να έχει πρόσβαση στο ίντερνετ και στις πληροφορίες που το περιβάλλουν.

Κατά τη διάρκεια της πανδημίας Covid-19, χάρη στην τεχνολογία, οι περισσότερες εταιρείες και οργανισμοί κατάφεραν να παραμείνουν εν λειτουργία. Εταιρίες που δραστηριοποιούνται στην υγεία, εκπαίδευση, τουρισμό, ηλεκτρονικό εμπόριο και όχι μόνο, χρησιμοποιούν πλέον τεχνολογίες όπως η επαυξημένη πραγματικότητα.

Η επαυξημένη πραγματικότητα είναι εδώ και είναι αυτή που θα αλλάξει τον τρόπο που μαθαίνουμε, εργαζόμαστε, παίζουμε και επικοινωνούμε με τον κόσμο. Είναι ένας ιδανικός τρόπος για να οπτικοποιήσει πράγματα που θα ήταν αδύνατο ή ανεφάρμοστο να δεις διαφορετικά.

1.2 Σκοπός και στόχος

Η παρούσα διπλωματική εργασία εξετάζει την Επαυξημένη πραγματικότητα ως πεδίο τεχνολογίας, καθώς και την εφαρμογή που έχει στο σήμερα με την χρήση κινητών συσκευών. Θα γίνει μελέτη, ανάλυση, σχεδιασμός και υλοποίηση μιας cross platform εφαρμογή, η οποία θα λειτουργεί ως ηλεκτρονικός ξεναγός πλοήγησης και περιήγησης σε ανοιχτούς χώρους ενδιαφέροντος, παρέχοντας ψηφιακές πληροφορίες όπως, εικόνες και χάρτες, αξιοποιώντας στο έπακρο όλες τις προ υπάρχουσες λειτουργίες των ανωτέρω συσκευών όπως GPS, οθόνη αφής και κάμερα. Θα γίνει χρήση τεχνολογιών αιχμής για ανάπτυξη εφαρμογών κινητών συσκευών καθώς και χρήση υπηρεσιών Backend as a service, έτσι ώστε η υλοποίηση της εφαρμογής να γίνει το συντομότερο δυνατόν.

Πιο αναλυτικά οι στόχοι της διπλωματικής είναι:

- Η μελέτη χρήσης εφαρμογών AR
- Η μελέτη συστημάτων κινητών συσκευών για ανάπτυξη εφαρμογών AR
- Η σχεδίαση και ανάπτυξη εφαρμογής ξενάγησης και πλοήγησης με χρήση AR για κινητές συσκευές

1.3 Δομή της εργασίας

Η παρούσα Διπλωματική Εργασία διαιρείται σε έξι κεφάλαια, καθένα από τα οποία αποτελείται από ενότητες και υποενότητες. Η δομή της παρουσιάζεται ως ακολούθως:

Στο κεφάλαιο 1, περιγράφεται το θέμα της διπλωματικής εργασίας και γίνεται μια εισαγωγή στο αντικείμενο της. Ορίζονται ο σκοπός, οι στόχοι και τα προσδοκώμενα αποτελέσματα της καθώς και με ποιο τρόπο οργανώνονται τα κεφάλια της.

Στο κεφάλαιο 2, περιγράφεται ο ορισμός της επαυξημένης πραγματικότητας, γίνεται μια εκτενής ιστορική αναδρομή και μία παρουσίαση στους τομείς χρήσης αυτής της τεχνολογίας.

Στο κεφάλαιο 3, έχουμε το θεωρητικό τεχνολογικό υπόβαθρο της διπλωματικής, στον οποίο ερμηνεύονται κάποιες τεχνολογικές έννοιες και ορισμοί που θα χρησιμοποιηθούν στην συνέχεια.

Στο κεφάλαιο 4, έχουμε την την ανάλυση και σχεδίαση του συστήματος, στην οποία αναλύονται εκτενέστερα οι λειτουργικές και μη απαιτήσεις του συστήματος, παρουσιάζονται τα διαγράμματα περιπτώσεων χρήσης και αλληλουχίας, και τέλος γίνεται μια εκτενής ανάλυση στις περιπτώσεις χρήσης του συστήματος.

Στο κεφάλαιο 5, γίνεται η ανάλυση υλοποίησης της εφαρμογής, στην οποία αναλύονται εκτενέστερα οι τεχνολογίες, οι υπηρεσίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της. Επίσης περιγράφεται και αναλύεται η διεπαφή χρήστη της εφαρμογής, παρουσιάζοντας και εικόνες του γραφικού της περιβάλλοντος.

Τέλος, στο κεφάλαιο 6 έχουμε την περίληψη και τα συμπεράσματα της διπλωματικής εργασίας καθώς και τυχόν επεκτάσεις που θα μπορούσαν να υλοποιηθούν στο μέλλον.

Κεφάλαιο 2 - Επαυξημένη πραγματικότητα (Augmented Reality)

2.1 Η έννοια της Επαυξημένης Πραγματικότητας

Η επαυξημένη πραγματικότητα[1] (αγγλικά: augmented reality) είναι η σε πραγματικό χρόνο άμεση ή έμμεση θέαση ενός πραγματικού περιβάλλοντος(φυσικού), του οποίου τα στοιχεία επαυξάνονται από στοιχεία αναπαραγώμενα από συσκευές υπολογιστών, όπως ήχος, βίντεο, γραφικά ή δεδομένα τοποθεσίας. Ο όρος εισήχθη το 1992 από τον Τομ Κάουντελ. (“Σ. Ε. Δ. Δ. Δ.: Σεπτεμβρίου 2018 - Blogger”)

Η τεχνολογία AR χρησιμοποιείται επί το πλείστων στα έξυπνα κινητά τηλέφωνα (smartphones) αλλά και σε Infokiosk πληροφόρησης, παρουσιάσεις και events. Η χρήση γίνεται μέσω της κάμερας και του συστήματος GPS ενός smartphone όπου παρέχονται πληροφορίες (κείμενα, εικόνες, ήχοι, video) για σημεία ενδιαφέροντος (Points of Interest - POI) στη γεωγραφική θέση που βρίσκεται ο χρήστης και στοχεύει με την κάμερα του. Η θέαση των εικονικών γραφικών ή δεδομένων είναι δυνατή είτε από τις οθόνες κινητών είτε από ειδικά γυαλιά προβολής.

2.2 Διαφορά Επαυξημένης και Εικονικής Πραγματικότητας

Η επαυξημένη πραγματικότητα (AR) και η εικονική πραγματικότητα[2] (VR) γεφυρώνουν τον ψηφιακό και φυσικό κόσμο. Αυτές οι καινοτόμες τεχνολογίες αλλάζουν τον τρόπο που εργαζόμαστε, μαθαίνουμε, και επικοινωνούμε στον κόσμο. Μας βοηθούν σε καθημερινές μας δραστηριότητες , όπως αναζήτηση πληροφοριών και αγορών. Οπτικοποιούν πράγματα που θα ήταν αδύνατον να δούμε διαφορετικά.

Αν και πολύ συχνά δημιουργείται σύγχυση ανάμεσα στους όρους Επαυξημένη Πραγματικότητα (AR) και Εικονική Πραγματικότητα (VR). Η διαφορά είναι ότι στην Εικονική Πραγματικότητα γίνεται προβολή κατασκευασμένου ψηφιακού περιεχομένου δημιουργώντας αίσθηση του χώρου ενώ στην Επαυξημένη Πραγματικότητα γίνεται προβολή ψηφιακού περιεχομένου στον φυσικό κόσμο.

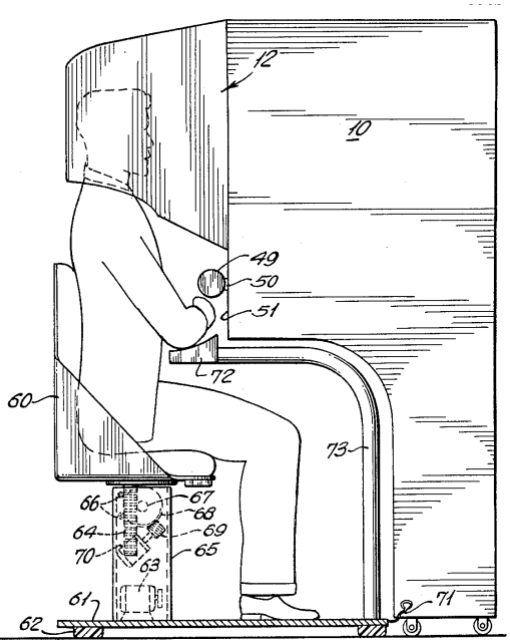
2.3 Ιστορική Αναδρομή Επαυξημένης και Εικονικής Πραγματικότητας

Όσο και να μας κάνει εντύπωση η έννοια της Επαυξημένης Πραγματικότητας (AR) δεν είναι τόσο νέα όσο νομίζουμε, οι αρχικές αναφορές πρώτο εμφανίστηκαν στις αρχές του 20ου αιώνα[3][4].

Αρχικά, εμφανίστηκε ως μια αναφορά το 1901 από τον L Frank Baum συγγραφέας του Wizard of Oz στο μυθιστόρημα The Master Key. Ο συγγραφέας περιγράφει ένα δώρο που λαμβάνει ο κεντρικός χαρακτήρας από έναν δαίμονα. Αυτό το δώρο ονομάστηκε

Character Marker (δείκτης χαρακτήρων), το οποίο ήταν ένα ζευγάρι γυαλιά τα οποία όταν τα φορούσε έβλεπε σημάδια στα μέτωπα των ανθρώπων όπως ένα γράμμα το οποίο έδειχνε τον χαρακτήρα τους (ένα Κ εάν ήταν κακός).Ο Baum κατάφερε να περάσει στο κοινό του την ιδέα των Google Glasses περίπου πριν από 100 χρόνια δηλαδή μια ηλεκτρονική οθόνη - γυαλιά που προσαυξάνουν αντικείμενα στον πραγματικό κόσμο.

Το 1955 ο κινηματογραφιστής Morton Heilig έγραψε ένα άρθρο με ονομασία 'The Cinema of the Future' στο οποίο περιέγραψε ότι κινηματογράφος είναι μια δραστηριότητά που θα μπορούσε να προσέλκυση τον θεατή με έναν διαδραστικό τρόπο συμμετοχής σε μια ταινία, λαμβάνοντας με αποτελεσματικό τρόπο αισθήσεις όπως οπτικά, ήχους, κραδασμούς, μυρωδιές, οπτικά και διάφορους ήχους.
Το 1962 δημιούργησε ένα πρωτότυπο με το όνομα Sensorama[5] (Εικόνα 2), το οποίο φυσικά δεν ήταν ελεγχόμενο από κάποιον υπολογιστή αλλά τουλάχιστον ήταν το πρώτο παράδειγμα μιας προσπάθειας προσθήκης επιπλέον δεδομένων σε μια εικονική εμπειρία.

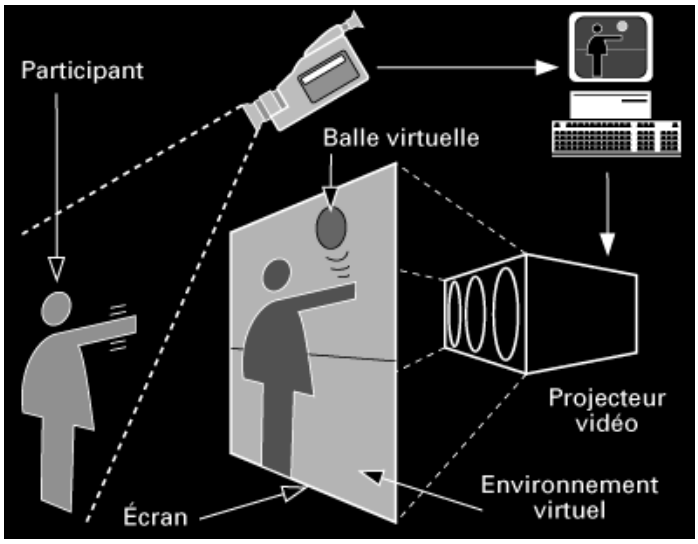


Εικόνα 1: Το Sensorama από τον Morton Heilig (Heilig, 1962)

Στην συνέχεια το 1968, ο Ivan Sutherland ένας Αμερικανός επιστήμονας πληροφορικής εφηύρε μια συσκευή με ονομασία Head Mounted Display (HMD)[6]. Ήταν ουσιαστικά μια τρισδιάστατη οθόνη που προσαρμοζεται στο κεφάλι και παρουσιάζει στον χρήστη μία εικόνα ενός πραγματικού αντικειμένου η οποία άλλαζε καθώς αυτός έκανε διάφορες κινήσεις του κεφαλιού του. Η τεχνολογία που χρησιμοποιήθηκε τότε έκανε την εφεύρεση μη πρακτική για μαζική χρήση.

Εικόνα 2: Το Head Mounted Display από τον Ivan Sutherland

Το 1975 ο Myron Krueger ένας Αμερικάνος καλλιτέχνης υπολογιστών ίδρυσε ένα εργαστήριο τεχνητής πραγματικότητας και το ονόμασε Videoplace[7]. Η ιδέα του με το videoplace επέτρεψε στους χρήστες της να χειραγωγούνται και να αλληλεπιδρούν με εικονικά αντικείμενα σε πραγματικό χρόνο χωρίς να επιβαρύνονται με τη χρήση γυαλιών και γαντιών. Το Videoplace χρησιμοποιούσε προβολείς, βιντεοκάμερες και μια μεγάλη οθόνη έτσι ώστε να τοποθετήσει τους χρήστες σε ένα διαδραστικό περιβάλλον. Αποτελούταν τουλάχιστον από δύο αίθουσες που θα μπορούσαν να βρίσκονται είτε στο ίδιο κτίριο είτε σε μια άλλη χώρα. Αυτό έκανε εφικτό διάφορους χρήστες να αλληλεπιδράσουν μεταξύ τους μέσω αυτής της τεχνολογίας. Οι συμμετέχοντες είχαν την δυνατότητα να μετακινήσουν διάφορες ψηφιακές εικόνες γύρω από την οθόνη μετακινώντας τον εαυτό τους και μπορούσαν επίσης να αλληλεπιδρούν με την εικόνα των άλλων συμμετεχόντων. Το Videoplace βρίσκεται εκτεθείτε στο Κρατικό Μουσείο Φυσικής Ιστορίας του Πανεπιστημίου Κονέκτικατ.



Εικόνα 3: Το Videoplace από τον Myron Krueger

Το 1980 Ο William Stephen George Mann ένας Καναδός μηχανικός και ερευνητής υπολογιστικής φωτογραφίας, δημιουργεί τον πρώτο φορητό υπολογιστή (wearable), ένα σύστημα όρασης υπολογιστή με κείμενο και γραφικές επικαλύψεις σε μια σκηνή με μεσολάβηση φωτογραφίας.

Το 1992 αποτέλεσε μια χρονιά ορόσημο για την ανάπτυξη και την εξέλιξη της επαυξημένης πραγματικότητας. Ο Thomas P. Caudell, πρώην ερευνητή της αεροπορική εταιρεία Boeing επινόησε τον όρο επαυξημένη πραγματικότητα. Αυτό συνέβη όταν ο ίδιος και ο συνάδελφος του David Mizell κλήθηκαν να βοηθήσουν σε ένα project[8] που θα έπρεπε να καθοδηγήσουν τους εργαζόμενους της εταιρείας να συναρμολογήσετε καλώδια για κάθε αεροσκάφος. Δημιούργησαν μια εφαρμογή επαυξημένης πραγματικότητας με συσκευή θέασης που προσαρτάται στο κεφάλι των εργατών (Head Mounted Display, HMD) που τοποθετούσε εικονικά διαγράμματα στον πίνακα τον οποίο δουλεύουν έτσι ώστε να συνδέσουν τις καλωδιώσεις στις σωστές θέσεις. Αυτό είχε ως αποτέλεσμα να υπάρχει μείωση του κόστους και βελτίωση της αποτελεσματικότητας σε πολλές από τις δραστηριότητες που εμπλέκονται χειρωνακτικές εργασίες.

Την ίδια χρονιά αναπτύχθηκε το πρώτο σωστά λειτουργικό σύστημα AR στο ερευνητικό εργαστήριο της της Πολεμική αεροπορία Αμερικής (Air Force Research Laboratory — Armstrong)[9] από τον Louis Rosenberg το 1992. Αυτό ονομάστηκε Virtual Fixtures » (Εικονικά Εξαρτήματα) . Τα εικονικά εξαρτήματα είχαν ως στόχος την επέκταση των ευκολιών ενός χειριστή και των ικανοτήτων του στη λύση προβλημάτων σε ένα μακρινό περιβάλλον, στο οποίο αυτός αισθάνεται παρόν. Στην πραγματικότητα ήταν ένα σύνθετο ρομποτικό σύστημα που επιτρέπει την επικάλυψη των αισθητηριακών πληροφοριών (αισθήσεις παραγόμενες από υπολογιστή) σε έναν μακρινό χώρο εργασίας, στον οποίο ο χειριστής αισθάνεται ότι βρίσκεται προκειμένου να βελτιωθεί η ανθρώπινη απόδοση σε άμεσες και εξ αποστάσεως

χειρισμένες εργασίες. Στη δημοσίευσή του, ο Rosenberg αναφέρει τη χρήση μόνο οπτικών και ακουστικών εικονικών εξαρτημάτων.

Στο μεταξύ, μια επιπλέον ομάδα ερευνητών από τον Steven Feiner, Doree Seligmann και Blair Macintyre παρουσιάζουν σε συνέδριο το πρώτο κύριο άρθρο πάνω σε ένα πρωτότυπο σύστημα επαυξημένης πραγματικότητας. Το σύστημα αυτό, με την ονομασία "KARMA"[12] (Knowledge-based-Augmented Reality for Maintenance assistance). χρησιμοποιεί ένα σύστημα HMD για την υποβοήθηση του τελικού χρήστη κατά τη συντήρηση ενός εκτυπωτή laser.

Υπήρχαν ακόμα πολλές σημαντικές ανακαλύψεις για την επαυξημένη πραγματικότητα από την αρχή της δεκαετιας του 90 μέχρι και σήμερα. Θα περιγράψουμε τωρα πιο συνοπτικά τα πλέον αξιοσημειωτα στιγμιότητα αυτης της επαναστατικής τεχνολογίας που πήραν μέρος κατα την διάρκεια του 20ού αιώνα μέχρι και σήμερα.

Μέχρι το 1991 δεν υπήρχε κάποιο λογισμικό ανοικτού κώδικα AR στην αγορά. Αυτό ήρθε και το κατέρριψε ο Hirokazu kato όταν κυκλοφόρησε το ARToolKit[13], μια βιβλιοθήκη που βοηθάει άλλους προγραμματιστές να δημιουργήσουν προγράμματα επαυξημένης πραγματικότητας. Η βιβλιοθήκη χρησιμοποιεί παρακολούθηση βίντεο για την επικάλυψη εικονικών γραφικών πάνω από τον πραγματικό κόσμο.
Μιας και τα κινητά τηλέφωνα και οι φορητοί υπολογιστές άρχισαν να εξελίσσονται πιο γρήγορα.

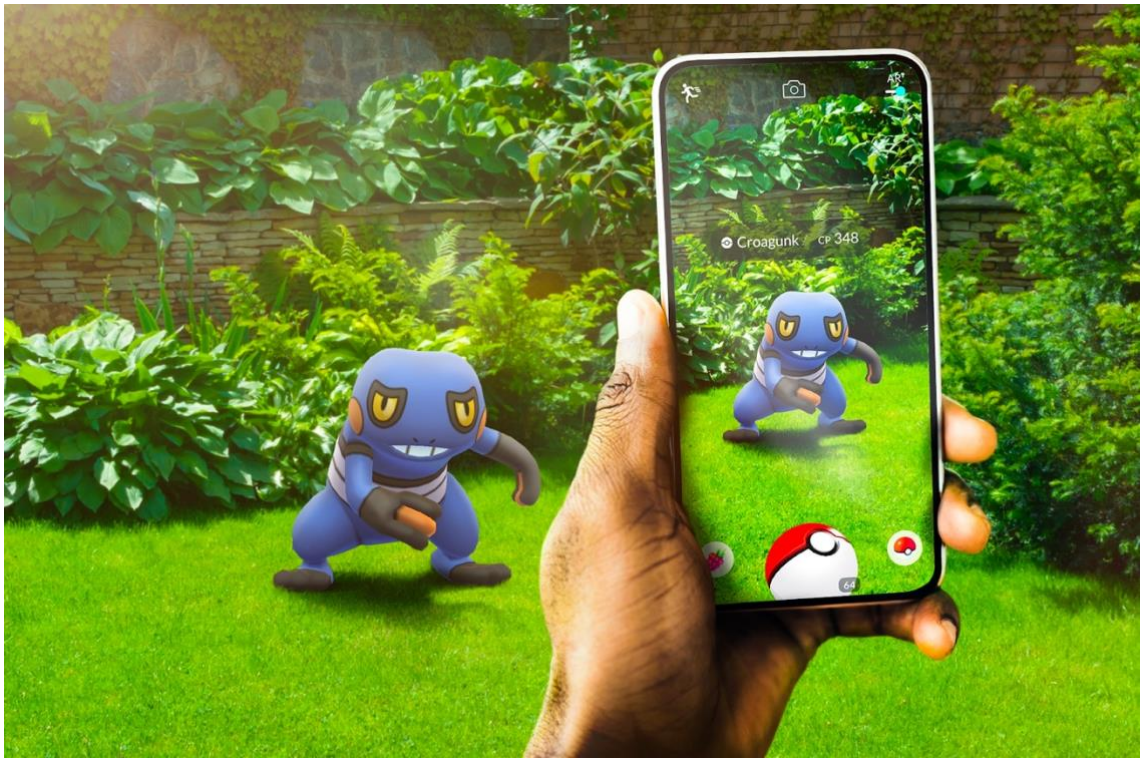
Το 2003 η Sony έβγαλε στο κοινό της μια εγχρωμη κάμερα για το playstation με την ονομασία EyeToy[14]. Η συγκεκριμένη κάμερα μπορούσε να ανιχνεύει πρόσωπα και να παρακολουθεί την κίνησή τους στον χώρο. Το πρόσωπο ενός ατόμου μπορούσε να αντικατασταθεί με ένα ψηφιακό πρόσωπο επαυξημένης πραγματικότητας όπως πρόσωπο καρτούν και να κινείται στον χώρο ή να κάνει εκφράσεις όπως χαμόγελο. ("Επαυξημένη πραγματικότητα")

Το 2008 η Wikitude AR Travel Guide[15] δημιουργήθηκε και με την χρήση του G1 Android phone πρόσφεραν στον χρήστη μια ξεχωριστή AR εμπειρία, παρόμοια σαν αυτή που δημιουργήσαμε στην παρούσα πτυχιακή, Το Wikitude είναι ένας ταξιδιωτικός οδηγός για κινητά που βασίζεται στη Wikipedia και το Panoramio.Ο χρήστης αναζητούσε με την χρήση της κάμερας διάφορα σημεία ενδιαφέροντος όπως βουνά και με την χρήση της επαυξημένης πραγματικότητας μπορούσε να δει αυτά τα τοπία.

Η Volkswagen έκανε το ντεμπούτο της με την εφαρμογή MARTA (Mobile Augmented Reality Technical Assistance)[16] η οποία έδινε κυρίως στους τεχνικούς οδηγίες επισκευής βήμα προς βήμα μέσα στο εγχειρίδιο σέρβις. Αυτή η προσαρμογή της τεχνολογίας AR ήταν πρωτοποριακή, καθώς θα μπορούσε και να εφαρμοστεί σε πολλές διαφορετικές βιομηχανίες για την ευθυγράμμιση και τον εξορθολογισμό των διαδικασιών.

Εδώ και αρκετά χρόνια η Microsoft πειραματίζεται με ολογράμματα και τεχνολογίες όπως η εικονικής πραγματικότητας. Το 2015 ανακοινώνει τα Windows Holographic μια ολοκληρωμένη πλατφόρμα Augmented Reality που συνοδεύεται από headset εικονικής πραγματικότητας. Με το Microsoft HoloLens[17] φέρνει ουσιαστικά έναν ασύρματο υπολογιστή Windows 10 στο κεφάλι του χρήστη, χάριν στον οποίο μπορεί να βλέπει εικονικά ολογράμματα σε τρεις διαστάσεις στον περιβάλλοντα χώρο. Η συσκευή τοποθετεί ένα στρώμα επαυξημένης πραγματικότητας που είναι εφικτό να το δούμε με την χρήση ειδικών γυαλιών, ενώ ταυτόχρονα επιτρέπει την αλληλεπίδραση του χρήστη με τα εικονικά αντικείμενα της πλατφόρμας Windows Holographic.

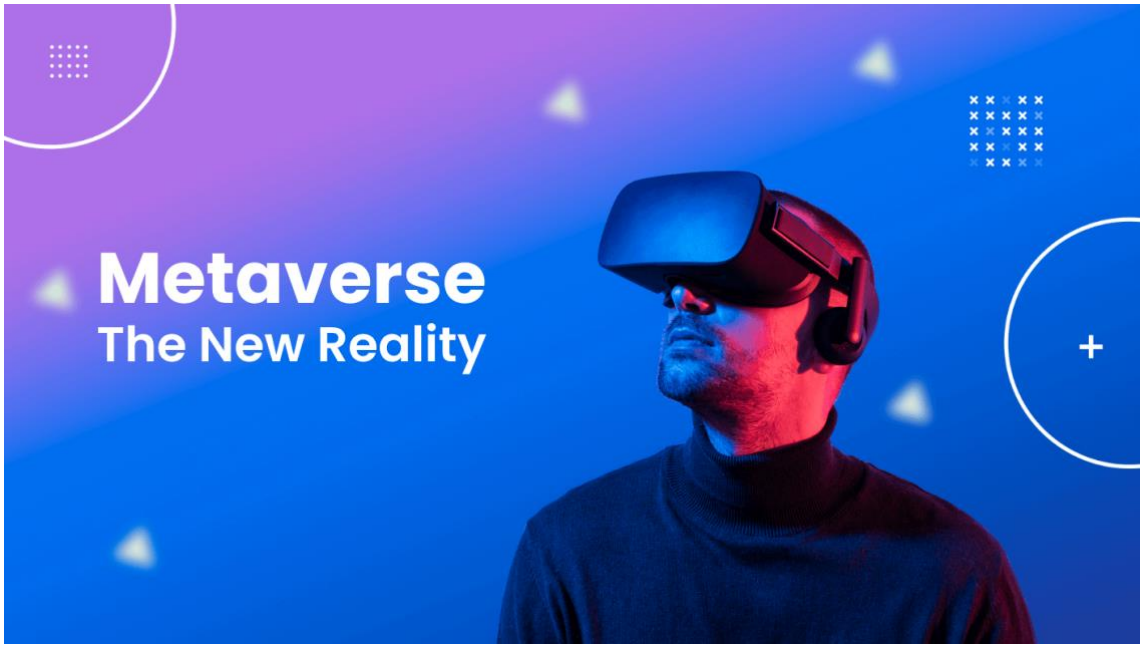
Το 2016, η Niantic παρουσίαση μια δωρεάν εφαρμογή επαυξημένης πραγματικότητας για έξυπνες συσκευές, το Pokémon Go[18]. Το παιχνίδι δίνει τη δυνατότητα στους παίκτες να μονομαχούν, να πιάνουν, και να προπονούν εικονικά πλάσματα, τα οποία ονομάζονται Pokémon, και τα οποία εμφανίζονται μέσω ενός ειδικού λογισμικού στην οθόνη, η οποία αποτυπώνει μέσω κάμερας τον πραγματικό κόσμο. Η εφαρμογή χρησιμοποιεί το GPS και την κάμερα της συμβατής συσκευής. (“Δύναμη Σκέψης: Ιουλίου 2016”) Το παιχνίδι αμέσως μετά την κυκλοφορία του έγινε μία από τις πιο διάσημες εφαρμογές για κινητές συσκευές με πολλά εκατομμύρια κατεβάσματα. (“Pokémon Go - Βικιπαίδεια”)



Εικόνα 4: Pokemon Go

Το 2019 η Microsoft ανακοινωσε μια πιο εξελιγμένη και εργονομικά καλύτερη έκδοση το HoloLens. Οi βασικές βελτιώσεις που έγιναν στη συσκευή ήταν ως προς την εργονομία και φιλικότητα προς τις επιχειρήσεις. Η Microsoft έχει επικεντρωθεί στην πώληση ακουστικών σε επιχειρήσεις που μπορούν να αντέξουν την τιμή των 3.500 δολαρίων και θέλουν να δουν αν η τεχνολογία κάνει τους εργαζόμενούς τους πιο παραγωγικούς.

Το Facebook έχει μπει δυναμικά στις τεχνολογίες του AR. Το 2021, άλλαξε το όνομά του σε Meta Platforms για να αντικατοπτρίζει το νέο focus το οποίο έθεσε, το metaverse. Το Metaverse[19] δεν έχει πλήκτρο on/off, είναι ένας κόσμος που συνεχίζει να υπάρχει και να λειτουργεί συνεχώς, ακόμη κι όταν εμείς δεν είμαστε συνδεδεμένοι σε αυτόν. (“Τι είναι το Metaverse και γιατί όλοι μιλούν γι’ αυτό τελευταία;”) Με λίγα λόγια είναι ένα σύμπαν μετά το πραγματικό.Τα σημεία κλειδιά για το metaverse περιλαμβάνουν την χρήση smartphone, επαυξημένη πραγματικότητα, μικτή πραγματικότητα και εικονική πραγματικότητας.



Εικόνα 5: Metaverse

2.4 Τύποι Εφαρμογών AR

Το καλύτερο μέρος της επαυξημένης πραγματικότητας είναι ότι είναι προσβάσιμη στον απλό χρήστη και σε αυτό έχουν βοηθήσει αρκετα οι μεγάλες εταιρίες που δραστηριοποιούνται σε αυτόν τον τομέα. Αυτή η εξέλιξη αυτής της τεχνολογίας έχει παίξει μεγάλο ρόλο στην κατηγοριοποίηση εφαρμογών επαυξημένης πραγματικότητας:

- Εφαρμογές βασισμένες στη χρήση ενός φυσικού δείκτη (Marker-Based AR)

- Εφαρμογές χωρίς χρήση φυσικού δείκτη (Markerless AR)
- Εφαρμογές με χρήση τοποθεσίας (Location-based AR)
- Εφαρμογές βασισμένες στην προβολή (Projection-based AR)

2.4.1 Marker-Based AR

Αυτός ο τύπος είναι γνωστός και ως αναγνώριση εικόνας (image recognition). Βασίζεται στην αναγνωρισει διαφορών δεικτών όπως μία εικόνα, για να μπορέσει να λειτουργήσει. Η κάμερα του κινητού μέσω της σχετικής εφαρμογής εστιάζει στον εκτυπωμένο δείκτη που υπάρχει στον πραγματικό κόσμο και το εμπλουτίζει με ψηφιακές πληροφορίες. (“Επαυξημένη Πραγματικότητα στην εκπαίδευση — Educraft”) Επομένως, ο χρήστης μπορεί να δει το αντικείμενο με περισσότερες λεπτομέρειες και από διάφορες οπτικές γωνίες. Οι πιο συνηθισμένοι δείκτες που χρησιμοποιούνται είναι οι δισδιάστατοι κωδικοί QR.

2.4.2 Markerless AR

Η επαυξημένη πραγματικότητα χωρίς δείκτη είναι μια από τις πιο ευρέως γνωστές εφαρμογές στον κλάδο. Αρχικά προσφέρουν μεγαλύτερο έλεγχο στον χρήστη, καθώς του επιτρέπουν να επιλέξει που θα ήθελε να τοποθετήσει το αντικείμενο ή περιεχόμενο. Επίσης επιτρέπει την τοποθέτηση επαυξημένων αντικειμένων σε πραγματικό μέγεθος. Αυτου του είδους οι εφαρμογές βασίζονται σε μεγάλο βαθμό σε χαρακτηριστικά έξυπνων τηλεφώνων, όπως αισθητήρες, κάμερα και επεξεργαστές. Οι εφαρμογές χωρίς χρήση φυσικού δείκτη χωρίζονται στις παρακάτω κατηγορίες: Location-based AR και Projection-based AR

2.4.3 Location-based AR

Λειτουργούν χωρίς δείκτες και χρησιμοποιούν GPS, επιταχυνσιόμετρο ή ψηφιακή πυξίδα για τον εντοπισμό της θέσης/θέσης του χρήστη και στη συνέχεια επικαλύπτουν ψηφιακά δεδομένα σε πραγματικές φυσικές τοποθεσίες. Περιέχουν πρόσθετες λειτουργίες, που τους επιτρέπουν να στέλνουν ειδοποίηση στον χρήστη σχετικά με πρόσφατα διαθέσιμο περιεχόμενο AR με βάση την τοποθεσία τους. Το Pokémon GO είναι το πιο γνωστό παράδειγμα επαυξημένης πραγματικότητας βάσει τοποθεσίας.

2.4.4 Projection-based AR

Αυτός είναι ένας από τους απλούστερους τύπους AR που είναι η προβολή φωτός σε μια επιφάνεια. Το AR που βασίζεται στην προβολή είναι ελκυστικό και διαδραστικό όταν το φως προσκρούει σε μια επιφάνεια και η αλληλεπίδραση γίνεται αγγίζοντας την προβαλλόμενη επιφάνεια με το χέρι. Οι πιο διαδεδομένες χρήσεις αυτής της τεχνικής AR είναι η δημιουργία εξαπάτησης σχετικά με τη θέση, τον προσανατολισμό και το βάθος ενός αντικειμένου.

2.5 Τομείς Χρήσης AR Εφαρμογών

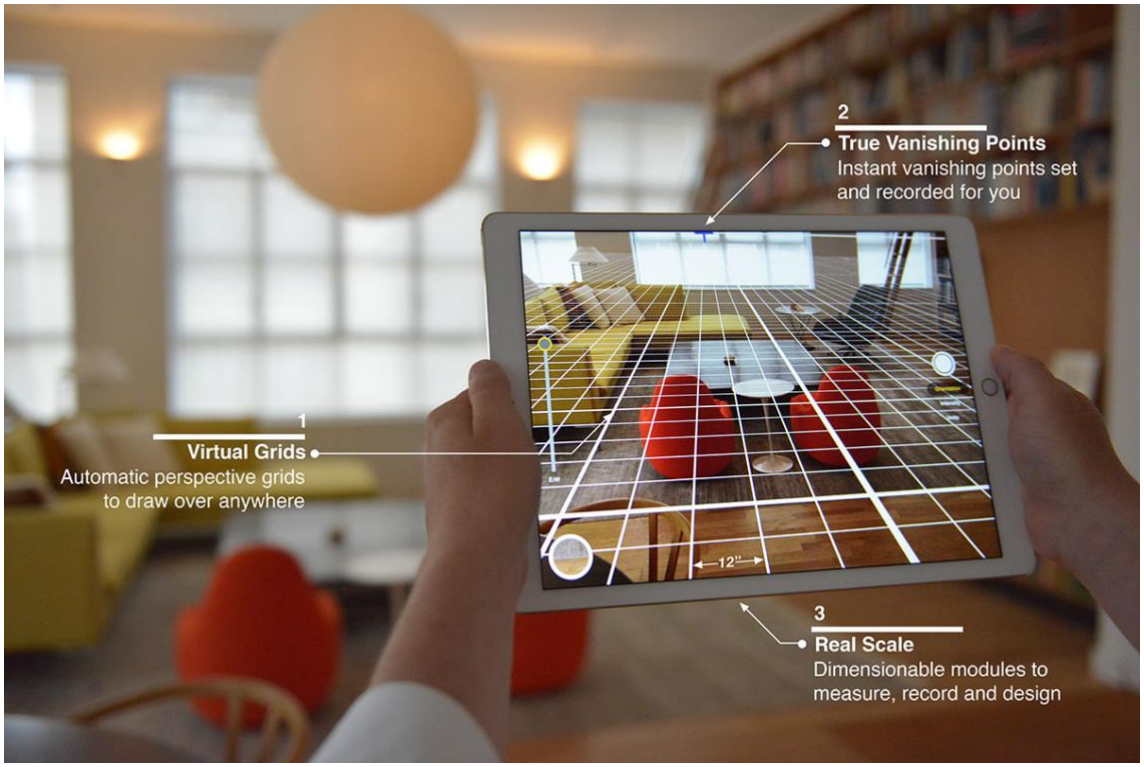
Η τεχνολογία της επαυξημένης πραγματικότητας βρίσκει εφαρμογές σε πολλούς τομείς της καθημερινής ζωής. Υπάρχουν πολλές εφαρμογές που μπορούν να τις χρησιμοποιήσουν η καθημερινοί χρήστες κινητών συσκευών. Παιχνίδια και εφαρμογές πλοηγησης ειναι απο τα πιο διαδεδομένα είδη των εφαρμογών AR που κυκλοφορούν είτε επι πλήρωμή ειτε ελεύθερα. Οι δυνατότητες που μας προσφέρει αυτός ο συνδυασμός ψηφιακών στοιχείων με τον πραγματικό κόσμο μπορεί να οδηγήσει σε σημαντικές εφαρμογές που θα βελτιώνουν τον τρόπο ζωή μας ή και ακόμα τον τρόπο που αντιλαμβανόμαστε το περιβάλλον Παρακάτω θα παρουσιάσουμε μερικούς τομείς που αυτή η τεχνολογία ήδη εφαρμόζεται.

2.5.1 Αρχιτεκτονική

Αναγνωρίζουμε ότι έχουν επέλθει μετασχηματισμοί στο χώρο της αρχιτεκτονικής κυρίως λόγω των τεχνολογιών και συγκεκριμένα των τεχνολογικών εφαρμογών επαυξημένης πραγματικότητας. Πλέον οι Αρχιτέκτονες έχουν την δυνατότητα να παρουσιάζουν μέσω αυτής της (AR) εφαρμογής μια πραγματική και ρεαλιστική ή και 3D αρχιτεκτονική οπτική μόνο από ένα σχέδιο, μακέτα ή φωτογραφία[21]. (“AR - Η Εφαρμογή για Αρχιτέκτονες & Διακόσμηση | Think AR”)

Επίσης ένας διακοσμητής μπορεί να διακοσμήσει ψηφιακά με πραγματικά ή και με 3D-Μοντέλα αντικειμένων και οικιακού εξοπλισμού και να κάνει μία δια δραστική παρουσίαση σε έναν εσωτερικό ή εξωτερικό χώρο για έναν πελάτη.

Το Morpholio AR Sketchwalk[22] είναι μια εφαρμογή AR που επιτρέπει στους σχεδιαστές ή αρχιτέκτονες να χρησιμοποιούν την επαυξημένη πραγματικότητα και με την δημιουργία διαφόρων σκίτσων να δώσουν τόσο στους πελάτες τους όσο και στους εαυτούς τους μια πιο αληθινή αίσθηση του χώρου. Χρησιμοποιώντας ένα iPad, τοποθετείτε το σκίτσο σας στο σχέδιο (ή στην τοποθεσία) και μπορείτε να το παρουσιάσετε, μεγαλώνοντας τις διαστάσεις του. Αυτό κάνει την εμπειρία της παρουσίασης ενός έργου πολύ πιο διαδραστική και ξεκάθαρη στους πελάτες.



Εικόνα 6: Morpholio AR Sketchwalk

2.5.2 Εμπόριο

Το AR επιτρέπει στους πελάτες ηλεκτρονικού εμπορίου να κάνουν προεπισκόπηση προϊόντων ή να έχουν εμπειρία υπηρεσιών στο δικό τους περιβάλλον και στον δικό τους χρόνο, προτού επιλέξουν να κάνουν μια αγορά. Χρησιμοποιώντας το AR, οι πελάτες μπορούν να κάνουν προεπισκόπηση των προϊόντων και να είναι πιο πιθανό να επιλέξουν το σωστό προϊόν την πρώτη φορά.

Το AR είναι χρήσιμο για εργασίες όπως η μεταφορά πρόσθετων πληροφοριών σχετικά με ένα προϊόν, η προβολή και η τοποθέτηση αντικειμένων σε όποιο δήποτε μεγέθους σε ένα χώρο ή η παρουσίαση του τρόπου λειτουργίας των πολλών κινούμενων μερών ενός σύνθετου μηχανήματος. Επιπλέον, με την ευρύτερη χρήση εφαρμογών AR για αγορές τόσο στο κατάστημα όσο και στο διαδίκτυο, το AR γίνεται όλο και πιο οικείο στους σημερινούς καταναλωτές και είναι πιο πιθανό να γίνει σημαντικός οδηγός πωλήσεων για το ηλεκτρονικό λιανικό εμπόριο και το ηλεκτρονικό εμπόριο. (“Επαυξημένη πραγματικότητα για το ηλεκτρονικό λιανικό εμπόριο και το ...”)

Το IKEA Place[23] είναι μια εφαρμογή AR που επιτρέπει στους χρήστες να δοκιμάζουν τα προϊόντα της IKEA σε πραγματικό χρόνο μέσω της τεχνολογίας ARKit της Apple iOS 11. Οι χρήστες της εφαρμογής απλά θα στοχεύουν την κάμερα του iPhone τους στον άδειο χώρο που θέλουν να επιπλώσουν για να δουν τι καναπές μπορεί να τους

ταιριάζει. Η εφαρμογή στη συνέχεια θα εμφανίζει μια προσαρμοσμένη στις διαστάσεις της εικόνα του επίπλου ή του αντικειμένου που ψάχνουν και θέλουν να αγοράσουν. (“IKEA Place: το νέο app που σας επιτρέπει να δοκιμάζετε τα έπιπλα”)

Εικόνα 7: IKEA Place

2.5.3 Εκπαίδευση

Το AR δημιουργεί ευκαιρίες για τους δασκάλους να βοηθήσουν τους μαθητές να κατανοήσουν αφηρημένες έννοιες. Χρησιμοποιώντας την αλληλεπίδραση και τον πειραματισμό που προσφέρουν οι τεχνολογίες AR[24], οι δάσκαλοι μπορούν να βελτιώσουν τις εμπειρίες στην τάξη, να διδάξουν νέες δεξιότητες, να εμπνεύσουν τα μυαλά των μαθητών και να ενθουσιάσουν τους μαθητές να εξερευνήσουν νέα ακαδημαϊκά ενδιαφέροντα. Το AR μπορεί να έχει σημαντικό αντίκτυπο στα μαθησιακά περιβάλλοντα όπως:

- Η ενασχόληση και το ενδιαφέρον των μαθητών: Το ενδιαφέρον των μαθητών εκτοξεύεται με την ευκαιρία να ασχοληθούν με τη δημιουργία εκπαιδευτικού περιεχομένου. Οι τεχνολογίες AR μπορούν να τους επιτρέπουν να προσθέσουν περιεχόμενο στο πρόγραμμα σπουδών, να δημιουργούν εικονικούς κόσμους και να εξερευνούν νέα ενδιαφέροντα.

- Κατανόηση περιεχομένου: Η υπάρχουσα τεχνολογία AR επιτρέπει στους εκπαιδευτικούς να δημιουργούν καθηλωτικές εκπαιδευτικές εμπειρίες από μόνοι τους για να διασφαλίσουν ότι οι μαθητές τους κατανοούν το περιεχόμενο του προγράμματος σπουδών.

- Περιβάλλον εκμάθησης: Τα μαθήματα που ενσωματώνουν AR μπορούν να βοηθήσουν τους μαθητές να εμπλακούν περισσότερο. Ένα διαδραστικό περιβάλλον μάθησης παρέχει ευκαιρίες για εφαρμογή πρακτικών προσεγγίσεων μάθησης που μπορούν να αυξήσουν τη δέσμευση, να βελτιώσουν τη μαθησιακή εμπειρία και να κάνουν τους μαθητές να μάθουν και να εξασκήσουν νέες δεξιότητες.

- Συνεργασία: Καθώς το περιεχόμενο AR είναι ψηφιακό, μοιράζεται εύκολα. Για παράδειγμα, μια ομάδα καθηγητών μπορεί να συνεργαστεί με τους μαθητές τους για να βελτιώνει συνεχώς το περιεχόμενο. Ένα συνεργατικό περιβάλλον μάθησης παρέχει στους μαθητές αυξημένα κίνητρα για μάθηση επειδή συμμετέχουν ενεργά στη διαδικασία δημιουργίας εκπαιδευτικού περιεχομένου.

- Μνήμη: Το AR είναι ένα εξαιρετικό εργαλείο για να ζωντανέψετε τα μαθήματα και να βοηθήσετε τους μαθητές να θυμούνται βασικές λεπτομέρειες. Για παράδειγμα, αντί να παρουσιάζει απλώς φωτογραφίες σε έναν προβολέα που παρουσιάζει τη ζωή στην Αρχαία Ελλάδα, ένας δάσκαλος μπορεί να χρησιμοποιήσει την τεχνολογία AR για να δημιουργήσει αξέχαστες διαδραστικές ιστορίες.

- Ανάπτυξη αισθήσεων: Η τεχνολογία AR μπορεί να βοηθήσει τους δασκάλους να δημιουργήσουν σχέδια μαθημάτων με πολυαισθητηριακές εμπειρίες. Οι μαθητές επωφελούνται από το καθηλωτικό εικονικό περιεχόμενο που ενσωματώνει ένα βιωματικό στυλ μάθησης στο οποίο οι μαθητές πραγματοποιούν σωματικές δραστηριότητες αντί να παρακολουθούν μια επίδειξη. Αυτή η προσέγγιση μπορεί να βοηθήσει στην αισθητηριακή ανάπτυξη.

- Κόστος και αποτελεσματικότητα: Το κόστος του εξοπλισμού AR αναφέρεται συχνά ως εμπόδιο στην αγορά. Ωστόσο, καθώς η χρήση κινητών συσκευών συνεχίζει να αυξάνεται μεταξύ των νεων και δεδομένου ότι τα smartphone είναι ήδη εξοπλισμένα με το υλικό που απαιτείται για την εκτέλεση εφαρμογών AR, η εφαρμογή της επαυξημένης πραγματικότητας στην εκπαίδευση είναι ολοένα και πιο οικονομική. Επιπλέον, η AR μπορεί να μειώσει το εκπαιδευτικό κόστος αντικαθιστώντας ακριβά σχολικά βιβλία.

Το Quiver[25] προσφέρει την ίδια μαγική εμπειρία που αποκομίζουν οι άνθρωποι από τις εφαρμογές AR, μόνο που αυτή τη φορά δίνει έμφαση στο εκπαιδευτικό περιεχόμενο. Η εφαρμογή παρέχει μια καθηλωτική εμπειρία εκμάθησης με σελίδες χρωματισμού που γίνονται κινούμενες. Οι μαθητές μπορούν να χρωματίσουν ένα αεροπλάνο και να το παρακολουθήσουν να πετά σε μια χώρα που εξερευνούν ή σπουδάζουν. Μπορούν επίσης να απολαύσουν διαδραστικά μαθήματα κυτταρικής

βιολογίας που τους επιτρέπουν να χρωματίζουν και να επισημαίνουν ένα κινούμενο φυτικό κύτταρο.

Εικόνα 8: Quiver

2.5.4 Παιχνίδια

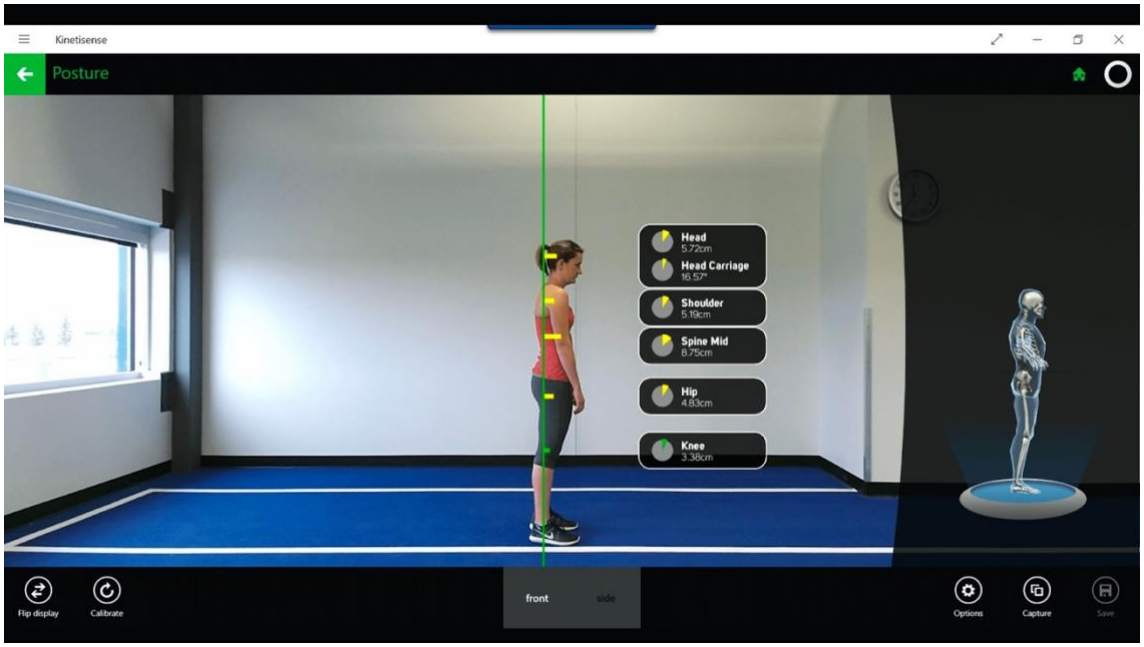
Η επαυξημένη πραγματικότητα επιτρέπει στους παίκτες να βιώσουν ένα ψηφιακό παιχνίδι σε ένα πραγματικό περιβάλλον. Είναι η ενσωμάτωση οπτικού και ακουστικού περιεχομένου παιχνιδιού με το περιβάλλον του χρήστη σε πραγματικό χρόνο. Σε αντίθεση με τα παιχνίδια εικονικής πραγματικότητας, τα οποία συχνά απαιτούν ξεχωριστό δωμάτιο ή περιορισμένο χώρο, τα παιχνίδια επαυξημένης πραγματικότητας χρησιμοποιούν το υπάρχον περιβάλλον και δημιουργούν ένα πεδίο παιχνιδιού μέσα σε αυτό.Ενώ τα παιχνίδια εικονικής πραγματικότητας απαιτούν εξειδικευμένα ακουστικά VR, μόνο ορισμένα συστήματα επαυξημένης

πραγματικότητας τα χρησιμοποιούν.Τα παιχνίδια AR παίζονται συνήθως σε συσκευές όπως smartphone, tablet και φορητά συστήματα παιχνιδιών.
Το Ingress χρησιμοποιεί την κινητή συσκευή GPS για να εντοπίσει και να αλληλεπιδράσει με πύλες που βρίσκονται κοντά στην πραγματική τοποθεσία του παίκτη. Οι πύλες είναι φυσικά σημεία ενδιαφέροντος όπου «εκφράζεται η ανθρώπινη δημιουργικότητα και εφευρετικοτητα» που συχνά εκδηλώνεται ως δημόσια τέχνη, όπως αγάλματα και μνημεία, μοναδική αρχιτεκτονική, υπαίθριες τοιχογραφίες, ιστορικά κτίρια, κόμβοι τοπικής κοινότητας και άλλες εκδηλώσεις ανθρώπινων επιτευγμάτων. Το παιχνίδι χρησιμοποιεί τις πύλες ως στοιχεία μιας ιστορίας επιστημονικής φαντασίας μαζί με μια συνεχή ανοιχτή αφήγηση που παρέχεται μέσω διαφόρων μορφών μέσων. Το Ingress Prime[26] χωρίζει τους παίκτες σε δύο ομάδες, Resistance και Enlightened, και ο σκοπός είναι να πάρεις στον έλεγχο σου της πήλες που εμφανίζονται στον χάρτη καθώς περιηγείσαι στον πραγματικό κόσμο. (“Ingress Prime: Διαθέσιμο το νέο AR game για Android και iOS”)

Εικόνα 9: Ingress Prime

2.5.5 Ιατρική

Η τεχνολογία AR μπορεί να παρέχει στο γιατρό πληροφορίες, οι οποίες κατά τα άλλα δεν είναι εμφανές με μια απλή εξέταση, όπως τον ρυθμό της καρδιάς, την αρτηριακή πίεση, την κατάσταση των οργάνων του ασθενούς, κλπ. Το AR μπορεί να χρησιμοποιηθεί για να αφήσει ένα γιατρό να κοιτάξει μέσα σε ασθενή χρησιμοποιώντας μία πηγή πληροφορίας, όπως μια ακτινογραφία ή μία μαγνητική τομογραφία. Με την χρήση αυτής της τεχνολογίας η θεραπεία ασθενειών μπορεί να μετατραπεί σε διασκέδαση. Χαρακτηριστικό παράδειγμα είναι το Kinetisense[27] για το Microsoft Kinect, που συνδυάζει τις τεχνολογίες VR και AR και μεταφέρει σε πραγματικό χρόνο κινήσεις και δεδομένα για την κατάσταση του σώματος του ασθενούς. (“Η εικονική πραγματικότητα αλλάζει την ιατρική | Fortunegreece.com”)



Εικόνα 10: Kinetisense

2.5.6 Πλοήγηση

Η τεχνολογία AR μπορεί να αυξήσει την αποτελεσματικότητα των συσκευών πλοήγησης. Οι πληροφορίες μπορούν να εμφανίζονται στο παρμπρίζ ενός αυτοκινήτου που δείχνει την αλλαγή κατεύθυνσης στον προορισμό και την υπολειπόμενη απόσταση, τον καιρό, το έδαφος, τις οδικές συνθήκες και πληροφορίες για την κυκλοφορία, καθώς και προειδοποιήσεις για πιθανούς κινδύνους στην πορεία του[28].

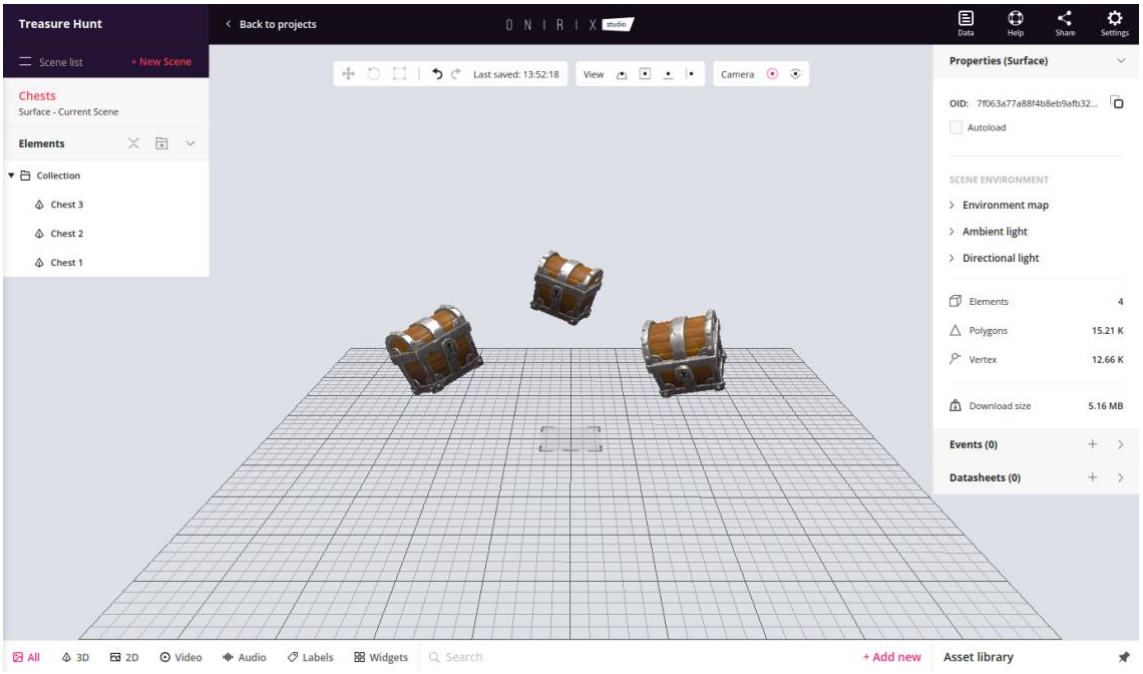
Εικόνα 11: παρμπρίζ με AR

2.5.7 Τουρισμός

Η εμπειρία της επαυξημένης πραγματικότητας στον κλάδο του τουρισμού ευνοήθηκε, κυρίως, από τον ψηφιακό μετασχηματισμό του κλάδου και, γενικότερα, της κοινωνίας. Τα smartphone, η μαζική και εύκολη πρόσβαση στο διαδίκτυο και η γεωγραφική τοποθεσία είναι οι κύριοι παράγοντες που έχουν ενισχύσει την AR στα ταξίδια. Δεν μπορούμε να ξεχάσουμε, επιπλέον, τις αλλαγές στη νοοτροπία και τη συμπεριφορά, καθώς και τις νέες τάσεις στις καταναλωτικές συνήθειες των χρηστών. Εν ολίγοις, τα ταξίδια επαυξημένης πραγματικότητας ενισχύουν την εμπειρία του ταξιδιώτη, καθιστώντας το πιο αληθινό και καθηλωτικό. Αυτό δημιουργεί έναν νέο και πολύ βελτιωμένο τρόπο ταξιδιού και τουρισμού. Οι AR εφαρμογές επιτρέπουν στους τουρίστες να γνωρίσουν προσομοιώσεις των ιστορικών γεγονότων, τόπων και αντικειμένων, καθιστώντας τες ορατές σε ένα πραγματικό τοπίο. Παρακάτω θα δούμε μερικά παραδείγματα.

2.5.8 Τουρισμός - AR Tour

Οι ξεναγήσεις είναι μια δραστηριότητα μονής κατεύθυνσης στην οποία ένας ξεναγός λέει στους τουρίστες τις πληροφορίες ενώ αυτοί απλώς στοχάζονται το περιβάλλον. Αλλά, όπως έχουμε πει, οι σημερινοί ταξιδιώτες απαιτούν περισσότερες διαδραστικές εμπειρίες που τους δίνουν περισσότερο πρωταγωνιστικό ρόλο. Πολλές τουριστικές τοποθεσίες προσφέρουν στους ταξιδιώτες τη δυνατότητα να κάνουν μια περιήγηση AR, ώστε να μπορούν να κάνουν μια πιο εις βάθος και βιωματική περιήγηση. Είναι μια αμφίδρομη δραστηριότητα, καθώς ο χρήστης αλληλεπιδρά με το περιβάλλον και τα στοιχεία του, είναι επίσης πραγματική και καθηλωτική. Μια περιήγηση AR, επομένως, μπορεί να προσφέρει διαφορετικές επιλογές. Από τη μεταφορά του χρήστη σε ένα εντελώς πραγματικό περιβάλλον και την αναδημιουργία ιστορικών ή παλαιότερων γεγονότων, έως την αλληλεπίδραση με διαφορετικά στοιχεία, την παροχή πληροφοριών και ακόμη και την αναπαραγωγή παιχνιδιών ενώ γνωρίζει το περιβάλλον. Για παράδειγμα, ένα κυνήγι θησαυρού για να διασκεδάσει ο ταξιδιώτης ανακαλύπτοντας το μέρος.



Εικόνα 12:Treasure hunt

2.5.9 Τουρισμός - AR εμπειρία μουσείων

Με την χρήση της επαυξημένης πραγματικότητας, η επίσκεψη σε ένα μουσείο μπορεί να είναι πιο δυναμική και, πάνω απ 'όλα, πιο πραγματική και ολοκληρωμένη. Ένα τέτοιο μουσείο επιτρέπει στους τουρίστες να απολαμβάνουν μια καθοδηγούμενη και διαδραστική περιήγηση ανά πάσα στιγμή χωρίς να χρειάζεται να κλείσουν ένα συγκεκριμένο πρόγραμμα και χωρίς να εξαρτώνται από έναν οδηγό ή μια ομάδα.Επίσης προσφέρεται η δυνατότητα πρόσβασης σε πιο ολοκληρωμένες και διασκεδαστικές πληροφορίες για κάθε κομμάτι του μουσείου, καθώς και την αναδημιουργία της ιστορίας καθενός από αυτά.

Ένα παράδειγμα υπερσύγχρονου, μινιμαλιστικού και διαδραστικού μουσείου ειναι το Athens Olympic Museum[29] που βρίσκεται στο Golden Hall, κοντά στο Ολυμπιακό Κέντρο Αθηνών «Σπύρος Λούης» (Ο.Α.Κ.Α.)

Εικόνα 13: Athens Olympic Museum

Κεφάλαιο 3 - Θεωρητικό τεχνολογικό υπόβαθρο

3.1 Ορισμός Λειτουργικού Συστήματος

Λειτουργικό Σύστημα (Λ.Σ.) (Operating System)[30] είναι ένα σύνολο από προγραμμάτα ενός υπολογιστικού συστήματος το οποίο λειτουργεί ως σύνδεσμος ανάμεσα στα προγράμματα του χρήστη αλλά και του υλικού. Το Λ.Σ. είναι υπεύθυνο για τη δημιουργία ενός περιβάλλοντος επικοινωνίας ανάμεσα σε έναν χρήστη και ένα σύστημα, τη διαχείριση και το συντονισμό των εργασιών του συστήματος, καθώς και για την κατανομή των διαθέσιμων πόρων. Με λίγα λόγια ένα τέτοιο λογισμικό επιτρέπει στα έξυπνα τηλέφωνα, Tablet, έξυπνα ρολόγια και διάφορες επιπλέον συσκευές να τρέχουν προγράμματα και εφαρμογές.

Το λειτουργικό σύστημα ενός κινητού ξεκινάει μόλις ενεργοποιηθεί μια συσκευή. Παρουσιάζοντας στην οθόνη κάποιο λογότυπο ή άλλες πληροφορίες που παρέχουν πρόσβαση στην συσκευή σε ασύρματα δίκτυα. Για παράδειγμα, τα δύο πιο γνωστά λειτουργικά συστήματα είναι τα εξής:Apple IOS και Google Android.

3.1.1 Apple IOS

Το iOS[31] είναι ένα ειδικό λειτουργικό σύστημα για κινητά που αναπτύχθηκε από την Apple. Είναι μέχρι στιγμής το πιο προηγμένο λειτουργικό σύστημα κινητής τηλεφωνίας στον κόσμο και σήμερα είναι ο ισχυρότερος αντίπαλος του Android στο τμήμα του λειτουργικού συστήματος για κινητά. Κωδικοποιημένο στο Objective-C, αυτό το φορητό λειτουργικό σύστημα είναι εντελώς κλειστού κώδικα και έχει σχεδιαστεί αποκλειστικά για συσκευές Apple, όπως iPhone, iPad και iPod Touch κ.λπ.

Το IOS διαθέτει iTunes το οποίο είναι για την μουσική. Από αυτήν ο χρήστης μπορεί να αναζήτηση εκατομμύρια τραγούδια, να τα αναπαράγει και έχει πολλές επιπλέον επιλογές. Επίσης, διαθέτει το δικό του Apple Store όπου από εκεί ο χρήστης μπορεί να αναζητήσει και να εγκαταστήσει όποια εφαρμογή θέλει, μελετώντας και τις αξιολογήσεις άλλων χρηστών των εφαρμογών. Ένα από τα πλεονεκτήματα του λειτουργικού συστήματος της Apple τα ενσωματωμένα βίντεο ομιλίες (video chatting) και δυνατότητες Apple Music.

3.1.2 Android OS

Το Android OS[32] είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα για κινητές συσκευές που υποστηρίζεται από Linux Kernel που αναπτύχθηκε από την Google. Προγραμματίζεται κυρίως από C, C ++, Java και kotlin και έχει σχεδιαστεί κυρίως και ειδικά για κινητές συσκευές με οθόνη αφής, όπως smartphone και tablet. Αυτό επιτρέπει στους προγραμματιστές λογισμικού να αναπτύξουν εφαρμογές και προγράμματα που ένα ποσοστό τους θα είναι διαθέσιμο στους χρήστες του δωρεάν.

3.2 Τύποι εφαρμογών για φορητές συσκευές

Αν και οι περισσότερες πλατφόρμες προσφέρουν τα δικά τους εργαλεία για την ανάπτυξη εφαρμογών. Αυτές οι εφαρμογές χωρίζονται σε τρεις μεγάλες κατηγορίες περιεχομένου για κινητές συσκευές, ως προς τον τρόπο ανάπτυξής του. Ανάλογα με τη χρήση που του δίνεται και τη συμβατότητα με τις συσκευές όπου θα εκτελεστεί, θα μπορούσαν να χρησιμοποιηθούν για διαφορετικές περιπτώσεις.

3.2.1 Εγγενείς κινητές εφαρμογές (native mobile development)

Η ανάπτυξη του κώδικα βασίζεται στο συγκεκριμένο λειτουργικό σύστημα που διαθέτει η κινητή συσκευή, οπότε απαιτούν την αποκλειστική γλώσσα κάθε λειτουργικού συστήματος για να λειτουργήσουν. Επομένως πρέπει να εκτελούνται σύμφωνα με τις προδιαγραφές κάθε συστήματος ξεχωριστά.

Για παράδειγμα, για να λειτουργεί μια εγγενής εφαρμογή που έχει σχεδιαστεί για iOS σε Android, είναι απαραίτητο να δημιουργήσετε μια πρόσθετη, εντελώς από την αρχή και στη γλώσσα του λειτουργικού συστήματος Android, πράγμα που σημαίνει

διαχωρισμός της υποστήριξης λειτουργιών μεμονωμένα. Οι εφαρμογές iOS αναπτύσσονται με τη γλώσσα Objective-C, ενώ το Android λειτουργεί με Java.

Σύμφωνα με τις λειτουργίες αυτών των εφαρμογών, μπορούμε να προσδιορίσουμε μια σειρά από πλεονεκτήματα που λαμβάνονται υπόψη κατά την επιλογή του τύπου που χρειαζόμαστε για την εταιρεία μας ή ανάλογα με το τι θέλουμε να κάνουμε. Αυτά είναι:

- Καλύτερη πρόσβαση σε σύγκριση με τις άλλες κατηγορίες.
- Καλύτερο αισθητικά προϊόν (τα χρώματα, σχεδίαση κουμπιών κλπ.) και καλύτερη αξιοποιεί σε όλες τις δυνατότητες της συσκευής.
- Είναι πολύ γρήγορες και έχουν πολύ καλή απόκριση μιας και είναι κατασκευασμένες για τη συγκεκριμένη πλατφόρμα.
- Μεγάλη αξιοπιστία και ασφάλεια
- Ενημερώνονται συνεχώς
- Μπορούν να λειτουργήσουν χωρίς σύνδεση στο Διαδίκτυο
- Διανέμονται σε ηλεκτρονικά καταστήματα εφαρμογών.

Ενώ τα μειονεκτήματα ειναι:

- Είναι πιο ακριβά στην ανάπτυξη λόγω τον συνεχών ενημερώσεων
- Ο κώδικας δε μπορεί να επαναχρησιμοποιηθεί για άλλο λειτουργικό σύστημα
- Η εφαρμογή πρέπει να περάσει από κάποιες διαδικασίες ελέγχου

3.2.2 Ιστοσελίδες κινητού Ιστού (mobile Web development)

Οι ιστοσελίδες κινητού Ιστού χρησιμοποιούν τις κυρίαρχες τεχνολογίες ανάπτυξης περιεχομένου για τον παγκόσμιο Ιστό οπως τις τεχνολογίες HTML5, CSS3 και JavaScript,. Αυτες οι εφαρμογές προσφέρονται στον κινητό χρήστη μέσω του προγράμματος περιήγησης που παρέχει η κινητή συσκευή.
Το πιο σημαντικό χαρακτηριστικό μιας εφαρμογής Ιστού είναι ότι μπορεί να χρησιμοποιηθεί σε οποιαδήποτε πλατφόρμα για φορητές συσκευές που διαθέτει πρόγραμμα περιήγησης στο Διαδίκτυο, αλλά συνήθως λειτουργούν και σε υπολογιστές, έξυπνες τηλεοράσεις και tablet.
Μια εφαρμογή web διαφέρει από την εγγενή γιατί δεν είναι εγκατεστημένη, δεν βρίσκονται ούτε σε επίσημα καταστήματα και χρειάζεται απαραίτητα σύνδεση στο διαδίκτυο για να λειτουργήσει. Αν και τα τρέχοντα προγράμματα περιήγησης σάς επιτρέπουν να προσθέσετε μια άμεση πρόσβαση στην οθόνη του κινητού τηλεφώνου.
Πλεονεκτήματα:

- Χαμηλό κόστος υλοποίησης με σχέση μια εγγενή εφαρμογή
- Μπορεί να χρησιμοποιηθεί σε οποιαδήποτε πλατφόρμα για κινητές συσκευες
- Μπορεί να δημοσιευθεί χωρίς κάποιον απαιτητικό ελεγχο η εξουσιοδότηση
- Δεν χρειάζεται να εγκατασταθεί στην συσκευή του χρήστη

μειονεκτήματα:

- Ως τελικό προϊόν δεν θα έχει τα native χαρακτηριστικά οπως μια εγγενή εφαρμογή
- Δεν μπορούν να εκμεταλλευτούν όλους τους διαθέσιμου πόρους μιας συσκευής
- Απαιτούν κυρίως σύνδεση στο διαδίκτυο
- Δεν δημοσιεύονται σε καταστήματα εφαρμογών

3.2.3 Υβριδικές κινητές εφαρμογές (hybrid ή cross - platform development)

Οι υβριδικές εφαρμογές για κινητά είναι εφαρμογές που εγκαθίστανται σε μια συσκευή, όπως και κάθε άλλη εγγενή εφαρμογή. Αυτό που τα διαφοροποιεί είναι το γεγονός ότι διαθέτουν στοιχεία από εγγενείς εφαρμογές, εφαρμογές που έχουν αναπτυχθεί για μια συγκεκριμένη πλατφόρμα όπως το iOS ή το Android, με στοιχεία από εφαρμογές ιστού, ιστότοπους που λειτουργούν σαν εφαρμογές αλλά δεν είναι εγκατεστημένοι σε μια συσκευή.
Αυτές οι εφαρμογές αναπτύσσονται σε γλώσσες Ιστού όπως CSS, HTML ή JavaScript, αλλά αυτά τα πακέτα μπορούν να ομαδοποιηθούν σε κώδικες iOS ή Android για να λειτουργήσουν και στα δύο λειτουργικά συστήματα. Για αυτό, χρησιμοποιούνται ειδικά frameworks οπως το React native και Cordova.

Η υβριδικές εφαρμογές μπορούν να εγκατασταθούν σε κινητά, επειδή οι κωδικοί προσαρμόζονται σε κάθε λειτουργικό σύστημα και ανεβαίνουν στα επίσημα καταστήματα.
Προκειμένου να αποκτήσουν πρόσβαση στις λειτουργίες υλικού μιας συσκευής (επιταχυνσιόμετρο, κάμερα, επαφές…), είναι δυνατό να συμπεριληφθούν εγγενή στοιχεία των διεπαφών χρήστη κάθε πλατφόρμας (iOS, Android). Ο εγγενής κώδικας θα χρησιμοποιηθεί για πρόσβαση στο συγκεκριμένα χαρακτηριστικά για να δημιουργήσετε μια απρόσκοπτη εμπειρία χρήστη. Οι υβριδικές εφαρμογές μπορούν επίσης να βασίζονται σε πλατφόρμες που προσφέρουν API JavaScript, εάν αυτές οι λειτουργίες καλούνται σε μια προβολή Web η αλλιώς WebView.
Πλεονεκτήματα:

- Χαμηλό κόστος υλοποίησης με σχέση μια εγγενή εφαρμογή
- Ένας πηγαίος κώδικας για περισσότερες από μία πλατφόρμες
- Παρόμοια διεπαφή με μια εγγενή εφαρμογή
- Μπορούν να δημοσιευθούν σε καταστήματα εφαρμογών

Μειονεκτήματα:

- Η εφαρμογή πρέπει να περάσει από κάποιες διαδικασίες ελέγχου
- Απαιτούν βιβλιοθήκες τρίτων για να αναπτυχθούν

3.3 Application Programming Interface

Η Διεπαφή Προγραμματισμού Εφαρμογών (αγγλ. API, από το Application Programming Interface)[36], είναι η διεπαφή των προγραμματιστικών διαδικασιών που παρέχει ένα λειτουργικό σύστημα, βιβλιοθήκη ή εφαρμογή προκειμένου να επιτρέπει να γίνονται προς αυτά αιτήσεις από άλλα προγράμματα ή/και ανταλλαγή δεδομένων.

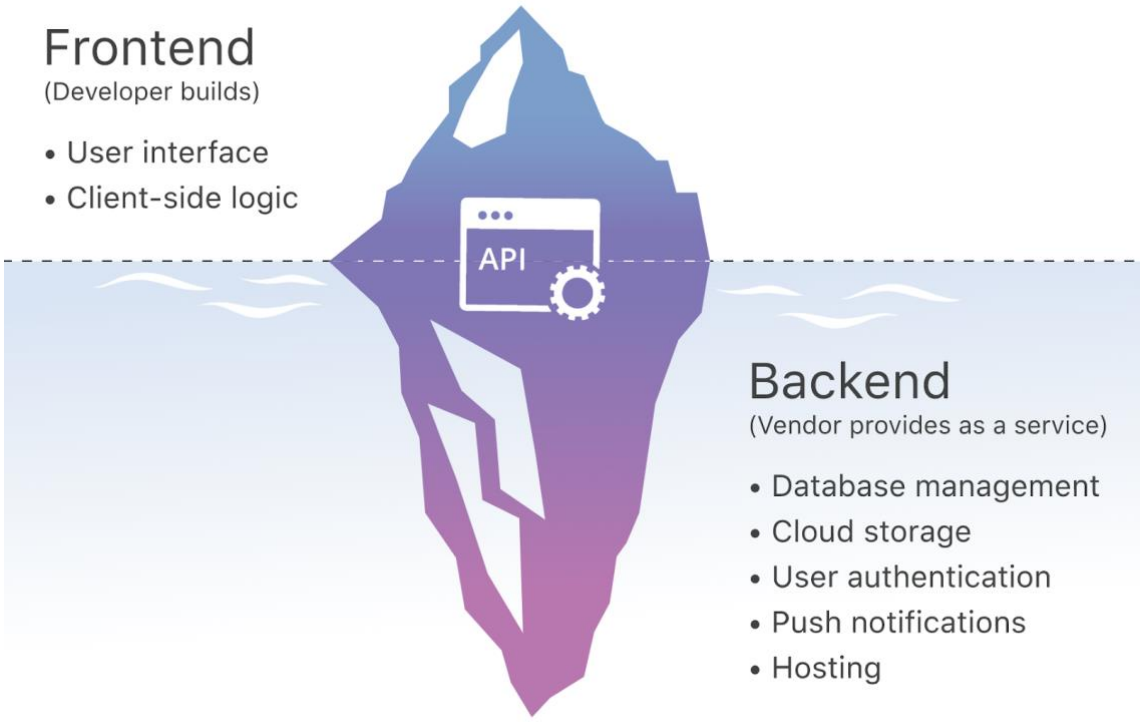
Ο βασικότερος σκοπούς μιας διεπαφής είναι ο ορισμός και η διατύπωση του σύνολου των λειτουργιών-υπηρεσιών που μπορεί να παρέχει μια βιβλιοθήκη ή ένα λειτουργικό σύστημα σε άλλα προγράμματα, χωρίς να επιτρέπει πρόσβαση στον κώδικα που υλοποιεί αυτές τις υπηρεσίες. Η διεπαφή είναι ένα συμβόλαιο κλήσης μεταξύ καλούντος και καλούμενου, η οποία διαχωρίζει την προγραμματιστική υλοποίηση κάποιων υπηρεσιών από τη χρήση τους.

Πολλές υπηρεσίες προσφέρουν δημόσια API που επιτρέπουν σε οποιονδήποτε να στέλνει και να λαμβάνει περιεχόμενο από την υπηρεσία. Τα API που λειτουργούν μέσω Διαδικτύου χρησιμοποιώντας URL http:// αναφέρονται ως API Ιστού. Στον Web το έτοιμα για να λάβουμε μια πληροφορία χαρακτηρίζεται ως GET ενώ στο να δημοσιεύσουμε POST.

3.4 Backend as a Service

Το Backend-as-a-Service (BaaS)[38] είναι ένα μοντέλο υπηρεσίας cloud στο οποίο οι προγραμματιστές αναθέτουν σε εξωτερικούς συνεργάτες όλες τις παρασκηνιακές πτυχές μιας εφαρμογής ιστού ή για κινητά, έτσι ώστε να χρειάζεται μόνο να γράφουν και να διατηρούν το frontend. Οι προμηθευτές BaaS παρέχουν προ-γραμμένο λογισμικό για δραστηριότητες που λαμβάνουν χώρα σε διακομιστές, όπως έλεγχος ταυτότητας χρήστη, διαχείριση βάσης δεδομένων, απομακρυσμένη ενημέρωση και ειδοποιήσεις push (για εφαρμογές για κινητά), καθώς και αποθήκευση και φιλοξενία στο

cloud.



Εικόνα 14: Backend as a Service

3.5 Διεπαφή Χρήστη – User Interface

Η διεπαφή χρήστη (UI)[39] είναι το μέρος του συστήματος που δρα ως ενδιάμεσο μεταξύ του χρήστη και του συστήματος διευκολύνοντας τον χρήστη να αλληλεπιδράσει με το σύστημα με αποτελεσματικό τρόπο. Η διεπαφή χρήστη είναι όλα όσα έρχεται σε επαφή ο τελικός χρήστης όταν χρησιμοποιεί το σύστημα.

Για τον τελικό χρήστη, η διεπαφή χρήστη είναι το ίδιο το σύστημα. Ως εκ τούτου, η χρηστικότητα ενός συστήματος παραμένει ένα από τα πιο σημαντικά χαρακτηριστικά ποιότητας για τον προσδιορισμό της συνολικής ποιότητας οποιουδήποτε συστήματος λογισμικού.

Η πρόκληση του σχεδιασμού της διεπαφής χρήστη[40] είναι η κατασκευή μιας ακολουθίας φυσικού διαλόγου που επιτρέπει στον χρήστη και τον υπολογιστή να ανταλλάσσουν τα μηνύματα που απαιτούνται για την εκτέλεση μιας συγκεκριμένης εργασίας. Οι διεπαφές χρήστη διαφέρουν από σύστημα σε σύστημα και χρήστη σε χρήστη.

3.6 Βιβλιοθήκες AR για Κινητές Συσκευές

Η εφαρμογή που θα υλοποιήσουμε θα είναι βασισμένη σε κάποια ήδη υπάρχουσα βιβλιοθήκη AR καθώς το υλοποιηθει απο το μηδέν μια τέτοια εφαρμογή δεν είναι εφικτό στα πλαίσια της παρούσας εργασίας. Παρακάτω θα αναλύσουμε μερικές από αυτές τις βιβλιοθήκες.

3.6.1 IOS ArKit

Το ARKit είναι ένα framework AR που δημιουργήθηκε από την Apple. Αναπτύχθηκε με την γλώσσα προγραμματισμού Objective-C και ενσωματώθηκε στα συστήματα iOS. Το ARKit[41] συνδυάζει την παρακολούθηση κίνησης της συσκευής, τη λήψη σκηνών με κάμερα, την προηγμένη επεξεργασία σκηνής και τις ανέσεις οθόνης για να απλοποιήσει το έργο της δημιουργίας μιας εμπειρίας AR. Μπορείτε να δημιουργήσετε πολλά είδη εμπειριών AR με αυτές τις τεχνολογίες χρησιμοποιώντας την μπροστινή ή την πίσω κάμερα μιας συσκευής iOS.

Η τελευταία έκδοση του ARKit παρέχει πολυ εξελιγμένες τεχνικές, ώστε ο χρήστης να μπορεί να τραβήξει εκπληκτικά βίντεο υψηλής ανάλυσης με εμπειρίες AR. Αποτελεί ιδανικό εργαλείο για επαγγελματίες που ασχολούνται με την επεξεργασία βίντεο, παραγωγή ταινιών, εφαρμογές κοινωνικών μέσων και πολλά άλλα. Οι δυνατότητες βίντεο και λήψης επεκτείνονται με την υποστήριξη βίντεο HDR και λήψης εικόνας φόντου υψηλής ανάλυσης. Το ARKit 6 φέρνει επίσης το Location Anchors σε νέες πόλεις, όπως το Μόντρεαλ, το Σίδνεϊ, τη Σιγκαπούρη και το Τόκιο, και διαθέτει βελτιώσεις στο Motion Capture.

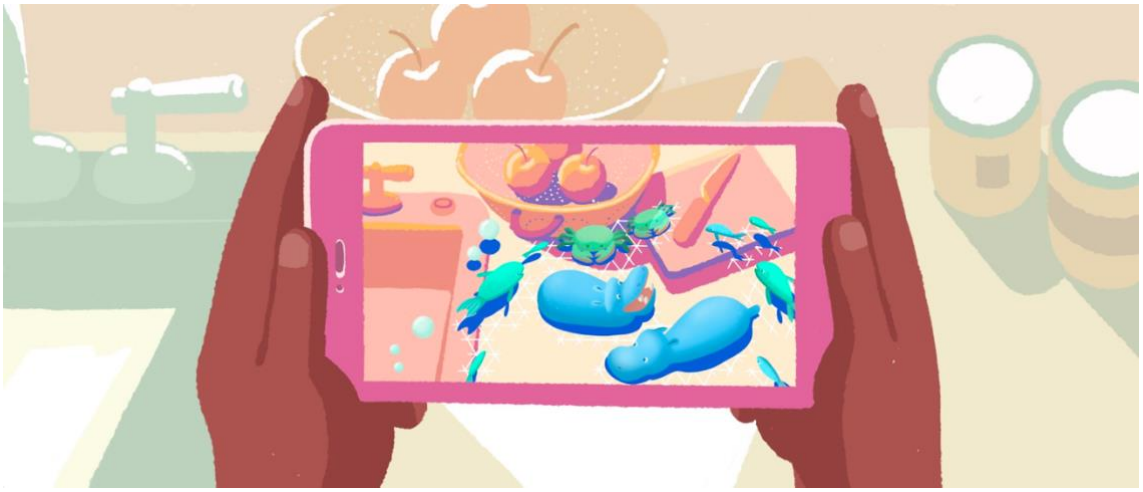
Εικόνα 15: IOS ArKit

3.6.2 ARCore

Το ARCore[42] είναι η πλατφόρμα της Google για τη δημιουργία εμπειριών επαυξημένης πραγματικότητας. Χρησιμοποιώντας διαφορετικά API, το ARCore επιτρέπει στο τηλέφωνό σας να αντιλαμβάνεται το περιβάλλον του, να κατανοεί τον κόσμο και να αλληλεπιδρά με πληροφορίες. Ορισμένα από τα API είναι διαθέσιμα σε Android και iOS για την ενεργοποίηση κοινών εμπειριών AR.

Το ARCore χρησιμοποιεί τρεις βασικές δυνατότητες για την ενσωμάτωση εικονικού περιεχομένου με τον πραγματικό κόσμο, όπως φαίνεται από την κάμερα του τηλεφώνου σας:

- Παρακολούθηση κίνησης που επιτρέπει στο τηλέφωνο να κατανοεί και να παρακολουθεί τη θέση του σε σχέση με τον κόσμο.
- Περιβαλλοντική κατανόηση που επιτρέπει στο τηλέφωνο να ανιχνεύει το μέγεθος και τη θέση όλων των τύπων επιφανειών: οριζόντιες, κάθετες και υπό γωνία επιφάνειες όπως το έδαφος, ένα τραπεζάκι του καφέ ή οι τοίχοι.
- Εκτίμηση φωτός που επιτρέπει στο τηλέφωνο να εκτιμήσει τις τρέχουσες συνθήκες φωτισμού του περιβάλλοντος.



Εικόνα 16: ARCore

3.6.3 Vuforia

Το Vuforia είναι ένα κιτ ανάπτυξης λογισμικού επαυξημένης πραγματικότητας (SDK) για κινητές συσκευές που επιτρέπει τη δημιουργία εφαρμογών επαυξημένης πραγματικότητας.

Χρησιμοποιεί τεχνολογία όρασης υπολογιστή για να αναγνωρίζει και να παρακολουθεί επίπεδες εικόνες και τρισδιάστατα αντικείμενα σε πραγματικό χρόνο. Αυτή η δυνατότητα καταχώρισης εικόνας επιτρέπει στους προγραμματιστές να τοποθετούν και να προσανατολίζουν εικονικά αντικείμενα, όπως μοντέλα 3D και άλλα μέσα, σε σχέση με αντικείμενα του πραγματικού κόσμου όταν αυτά προβάλλονται μέσω της κάμερας μιας κινητής συσκευής. Στη συνέχεια, το εικονικό αντικείμενο παρακολουθεί τη θέση και τον προσανατολισμό της εικόνας σε πραγματικό χρόνο, έτσι ώστε η προοπτική του θεατή στο αντικείμενο να αντιστοιχεί με την προοπτική του στόχου. Φαίνεται λοιπόν ότι το εικονικό αντικείμενο είναι μέρος της σκηνής του πραγματικού κόσμου.

Το Vuforia SDK[43] υποστηρίζει μια ποικιλία τύπων στόχων 2D και 3D, συμπεριλαμβανομένων των στόχων εικόνας "χωρίς δείκτη", του στόχου 3D μοντέλου και μια μορφή διευθυνσιοδοτούμενου Fiducial Marker, γνωστό ως VuMark. Οι πρόσθετες δυνατότητες του SDK περιλαμβάνουν εντοπισμό συσκευής 6 βαθμών ελευθερίας στο χώρο, εντοπισμό εντοπισμού απόφραξης τοπικά χρησιμοποιώντας «Εικονικά κουμπιά», επιλογή στόχου εικόνας χρόνου εκτέλεσης και δυνατότητα δημιουργίας και αναδιαμόρφωσης συνόλων στόχων μέσω προγραμματισμού κατά τη διάρκεια εκτέλεσης.

Η Vuforia[44] παρέχει διεπαφές προγραμματισμού εφαρμογών (API) σε C++, Java, Objective-C++ και τις γλώσσες .NET μέσω μιας επέκτασης στη μηχανή παιχνιδιών Unity. Με αυτόν τον τρόπο, το SDK υποστηρίζει τόσο την εγγενή ανάπτυξη για iOS,

Android και UWP, ενώ επιτρέπει επίσης την ανάπτυξη εφαρμογών AR στο Unity που είναι εύκολα φορητές και στις δύο πλατφόρμες.

Εικόνα 17: Vuforia

3.6.4 Viro React

Το Viro React[45] είναι μια πλατφόρμα για προγραμματιστές που μπορούν να δημιουργήσουν γρήγορα εμπειρίες επαυξημένης πραγματικότητας (AR) και εικονικής πραγματικότητας (VR). Οι προγραμματιστές γράφουν τον κώδικα τους στο React Native και το Viro εκτελεί τον κώδικά τους εγγενώς σε όλες τις πλατφόρμες εικονικής πραγματικότητας για κινητά.

Το Viro React υποστηρίζει για εφαρμογές AR τα framework ARKit και ARCore ενώ για εφαρμογές VR τα framework Cardboard (iOS and Android), Daydream and Gear VR. Ενώ αποτελείται από δύο κύρια μέρη:

- Το Viro React περιλαμβάνει μια σειρά από εξειδικευμένα στοιχεία React που σας επιτρέπουν να αποδώσετε διάφορα σενάρια, αντικείμενα και στοιχεία ελέγχου σε ένα τρισδιάστατο περιβάλλον.
- Το Viro React είναι μια εγγενής μηχανή απόδοσης 3D υψηλής απόδοσης που χρησιμοποιεί την επιτάχυνση υλικού εγγενούς συσκευής για την απόδοση όλων των αντικειμένων.

Τα πλεονεκτήματα του Viro React είναι:

- Εύκολο στην μάθηση
- Είναι δωρεάν
- Μπορούμε να χρησιμοποιήσουμε ταυτοχρονα και τις δύο τεχνολογίες AR και VR
- Ένας πηγαίος κώδικας για περισσότερες από μία πλατφόρμες

Κεφάλαιο 4 - Ανάλυση και Σχεδίαση Συστήματος

4.1 Καθορισμός Λειτουργικών - μη λειτουργικών απαιτήσεων

Οι απαιτήσεις του λογισμικού καταγράφονται σε δυο κατηγορίες, στις λειτουργικές και τις μη λειτουργικές:

- Λειτουργικές απαιτήσεις είναι η περιγραφή των λειτουργιών που θα πρέπει να κάνει το λογισμικό (π.χ. ως συναρτήσεις που λαμβάνουν είσοδο και δίνουν έξοδο).

- Μη λειτουργικές απαιτήσεις είναι η περιγραφή των χαρακτηριστικών του λογισμικού όπως η απόδοση, χρηστικότητα, ασφάλεια, νομιμότητα και ιδιωτικότητα. Με άλλα λόγια, περιγράφουν το πώς (ή το πόσο καλά) το λογισμικό θα υποστηρίξει τις λειτουργικές απαιτήσεις.

4.1.1 Λειτουργικές απαιτήσεις

Ο χρήστης της εφαρμογής θα μπορεί:

- Να δημιουργήσει έναν καινούριο λογαριασμό, να συνδεθεί και να αποσυνδεθεί από την εφαρμογή.

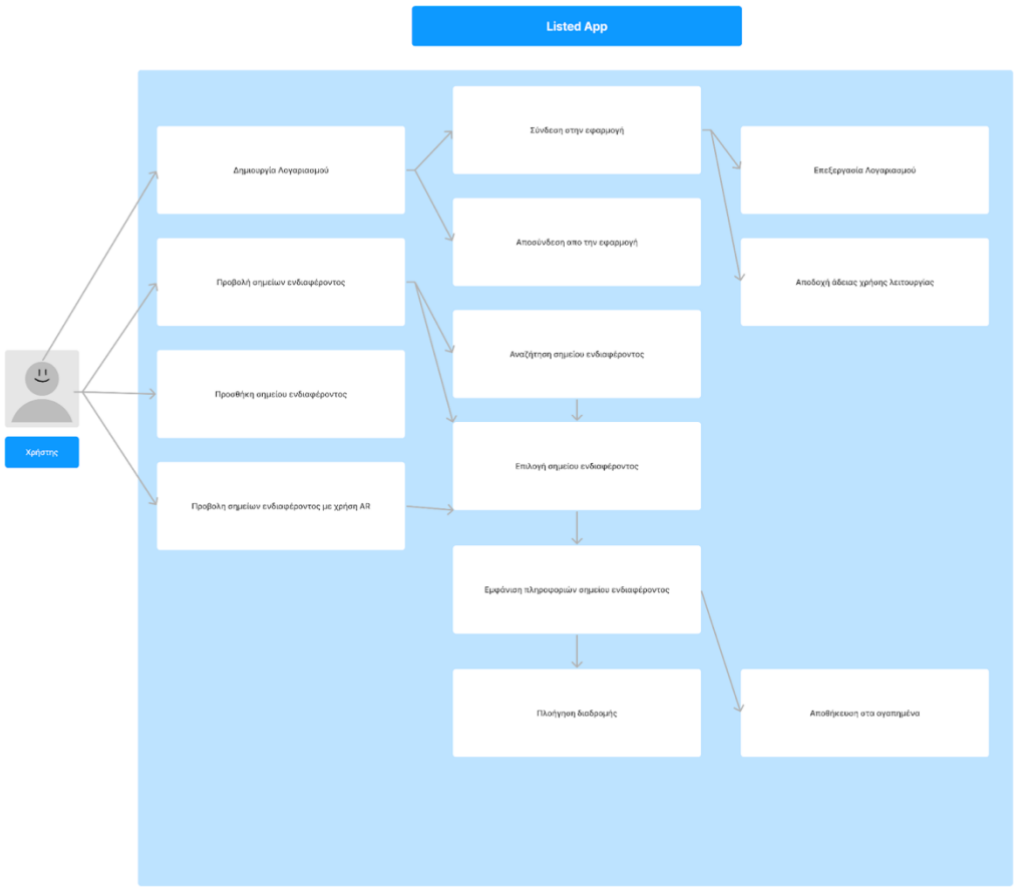
- Να επιτρέπει στην εφαρμογή να κάνει χρήση της κάμερας και άλλων λειτουργιών για κινητές συσκευές.
- Να επεξεργαστεί τα πρωσωπικά του δεδομένα όπως το όνομα, email και ηλικία.
- Να δημιουργήσει ένα νέο σημείο ενδιαφέροντος.
- Να δημιουργήσει ένα νέο σημείο ενδιαφέροντος.
- Να έχει την δυνατότητα προβολής των σημείων ενδιαφέροντος
- Να αποθηκευσει ως αγαπημένο ένα σημείο ενδιαφέροντος.
- Να αναζητήσει ένα σημείο με το όνομα του.
- Να έχει την δυνατότητα πρόσβασης σε πολυμεσικό περιεχόμενο (φωτογραφίες, βίντεο, κείμενο) για κάποιο σημείο ενδιαφέροντος που έχει επιλέξει.
- Να βλέπει σε χάρτη της εφαρμογής τα σημείων ενδιαφέροντος.
- Να επιλέξει πλοήγηση από το σημείο που βρίσκεται σε κάποιο από τα σημεία ενδιαφέροντος.
- Να επιλέξει συνδέσμους σε εφαρμογές τρίτων που αφορούν τα εμφανιζόμενα σημεία ενδιαφέροντος.
- Να βλέπει το σημείο να επαυξάνεται στο φυσικό κόσμο με χρήση της κάμερας.

4.1.2 Μη Λειτουργικές απαιτήσεις

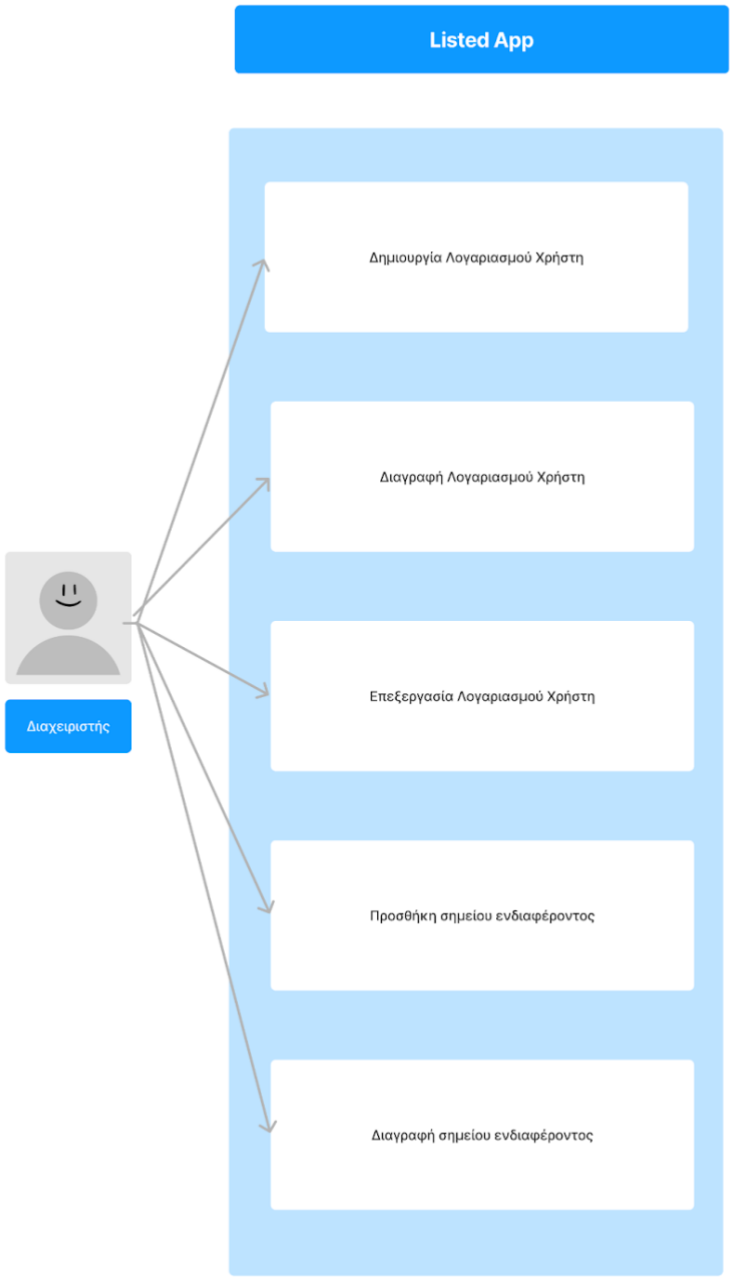
- Η εφαρμογή θα πρέπει να μπορεί να εκτελεστεί σε φορητές συσκευές με λειτουργικό σύστημα IOS 13 και πάνω.
- Απαιτήσεις επικοινωνίας με άλλα συστήματα όπως firebase.
- Η εφαρμογή θα πρέπει να χειρίζεται με τη χρήση της οθόνης αφής της φορητής συσκευής.
- Η εφαρμογή θα μπορεί να χειριστεί σε οποιονδήποτε προσανατολισμό οθόνης της φορητής συσκευής (κατακόρυφο – οριζόντιο) καθώς και σε οποιοδήποτε μέγεθος οθόνης.
- Η εφαρμογή θα πρέπει να δίνει την αίσθηση native εφαρμογής στον χρήστη.

4.2 Διάγραμμα περιπτώσεων χρήσης

Παρακάτω παρουσιάζεται το διάγραμμα περιπτώσεων χρήσης του χρήστη:



Παρακάτω παρουσιάζεται το διάγραμμα περιπτώσεων χρήσης του Διαχειριστή:



4.3 Ανάλυση Περιπτώσεων χρήσης συστήματος

Παρακάτω θα αναλύσουμε τις περιπτώσεις χρήσης του χρήστη:

4.3.1 Δημιουργια Λογαριασμού

Βασική Ροή

1. Ο χρήστης ανοίγει την εφαρμογή
2. Επιλέγει την δημιουργία λογαριασμού
3. Συμπληρώνει τα απαραίτητα πεδία
4. Συνδέεται στην εφαρμογή αυτόματα
5. Επιλέγει το εικονίδιο χρήστη και μεταφέρεται στο οθόνη του προφίλ
6. Επιλέγει επεξεργασία προφίλ
7. Κάνει τις απαραίτητες αλλαγές και επιλέγει αποθήκευση

Εναλλακτική ροή

6. Επιλέγει την άδεια χρήσης λειτουργίας
7. Μεταφέρεται στην οθόνη ρυθμίσεων

4.3.2 Προβολή Σημείων Ενδιαφέροντος

Βασική Ροή

1. Ο χρήστης μεταφέρεται στην αρχική οθόνη. Σε αυτό το βήμα ο χρήστης μπορεί να χρησιμοποιήσει την δυνατότητα αναζήτησης.
2. Επιλέγει το επιθυμητό σημείο ενδιαφέροντος.
3. Μεταφέρεται στην οθόνη του σημείου και προβάλλονται η απαραίτητες πληροφορίες
4. Επιλέγει αποθήκευση στα αγαπημένα

Εναλλακτική ροή

4. Επιλέγει πλοήγηση διαδρομής

4.3.3 Προσθήκη Σημείου Ενδιαφέροντος

Βασική Ροή

1. Επιλέγει το εικονίδιο χρήστη και μεταφέρεται στο οθόνη του προφίλ
2. Επιλέγει προσθήκη σημείου ενδιαφέροντος
3. Συμπληρώνει την φόρμα
4. Επιλέγει αποθήκευση
5. Μεταφέρεται αυτόμα στην αρχική σελίδα

4.3.4 Προβολή Σημείων Ενδιαφέροντος με χρήση AR

Βασική Ροή

1. Ο χρήστης μεταφέρεται στην οθόνη AR

2. Μετακινεί το κινητό του και με την αποδοχή πρόσβασης στην κάμερα βλέπει διάφορα σημεία ενδιαφέροντος στην οθόνη του
3. Επιλέγει το επιθυμητό σημείο
4. Μεταφέρεται στην οθόνη του σημείου και προβάλλονται η απαραίτητες πληροφορίες
5. Επιλέγει αποθήκευση στα αγαπημένα

Εναλλακτική ροή

5. Επιλέγει πλοήγηση διαδρομής

4.3.5 Επεξεργασία Προφίλ χρήστη

Βασική Ροή

1. Επιλέγει το εικονίδιο χρήστη και μεταφέρεται στο οθόνη του προφίλ
2. Επιλέγει επεξεργασία προφίλ
3. Τροποποιεί τα στοιχεία της φόρμα
4. Επιλέγει αποθήκευση
5. Μεταφέρεται αυτόμα στην σελίδα ρυθμίσεων

Παρακάτω θα αναλύσουμε τις περιπτώσεις χρήσης του διαχειριστή:

4.3.6 Δημιουργια Λογαριασμού Χρήστη

Βασική Ροή

1. Ο διαχειριστης συνδεεται στην πλατφόρμα Firebase
2. Επιλέγει το Project που τον ενδιαφέρει
3. Μεταφέρεται στην οθόνη Authentication
4. Προσθέτει τον νέο χρήστη

4.3.7 Διαγραφή Λογαριασμού Χρήστη

Βασική Ροή

1. Ο διαχειριστης συνδεεται στην πλατφόρμα Firebase
2. Επιλέγει το Project που τον ενδιαφέρει
3. Μεταφέρεται στην οθόνη Authentication
4. Διαγράφει τον χρήστη

4.3.8 Επεξεργασία Λογαριασμού Χρήστη

Βασική Ροή

1. Ο διαχειριστης συνδεεται στην πλατφόρμα Firebase
2. Επιλέγει το Project που τον ενδιαφέρει
3. Μεταφέρεται στην οθόνη Realtime Database
4. Αναζητεί τον χρήστη
5. Επεξεργάζεται τις πληροφορίες όπως ηλικία, χώρα και φύλλο

4.3.9 Προσθήκη σημείου ενδιαφέροντος

Βασική Ροή

1. Ο διαχειριστης συνδεεται στην πλατφόρμα Firebase
2. Επιλέγει το Project που τον ενδιαφέρει
3. Μεταφέρεται στην οθόνη Realtime Database
4. Κάνει προσθήκη του σημείου ενδιαφέροντος

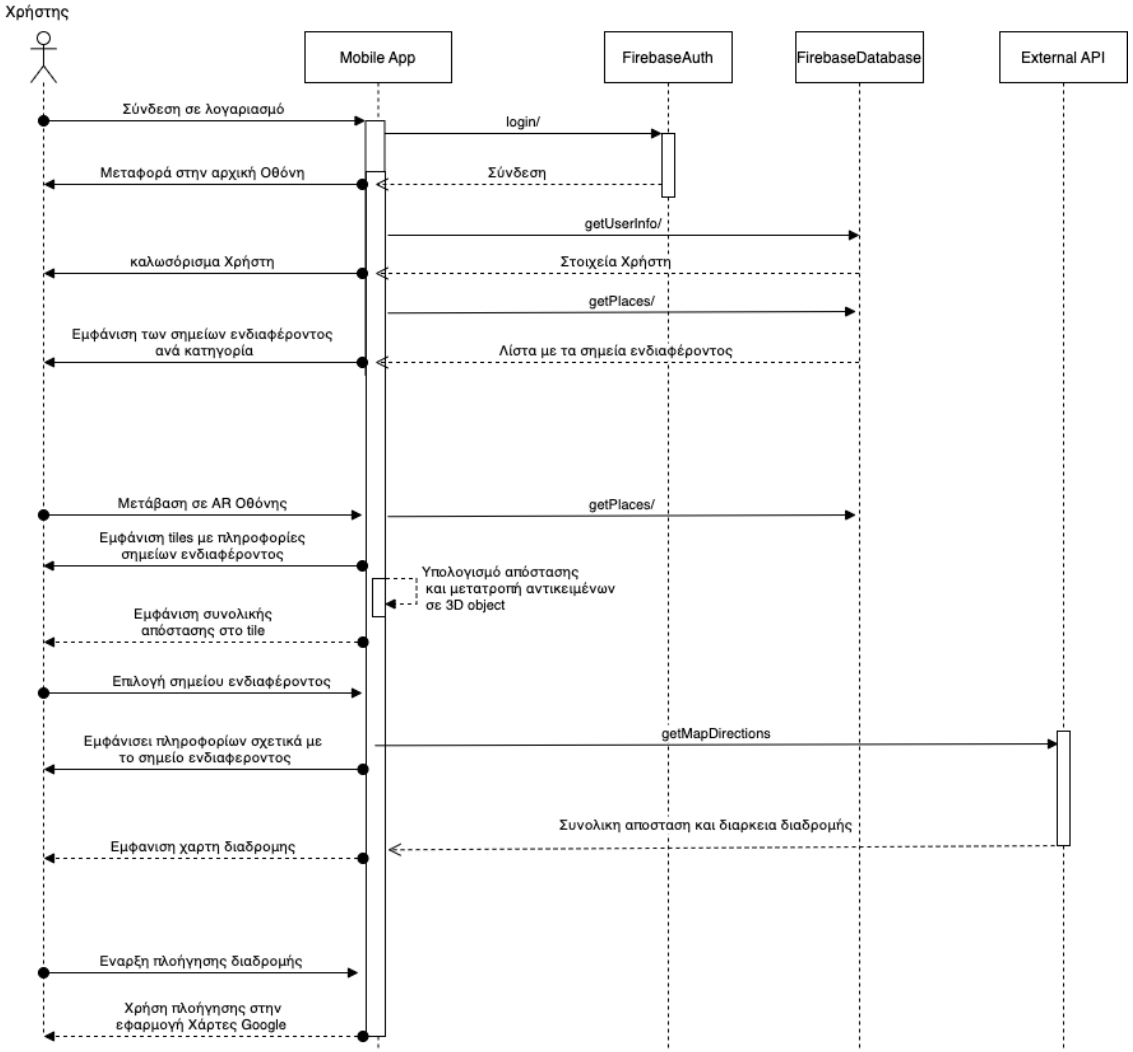
4.3.10 Διαγραφή σημείου ενδιαφέροντος

Βασική Ροή

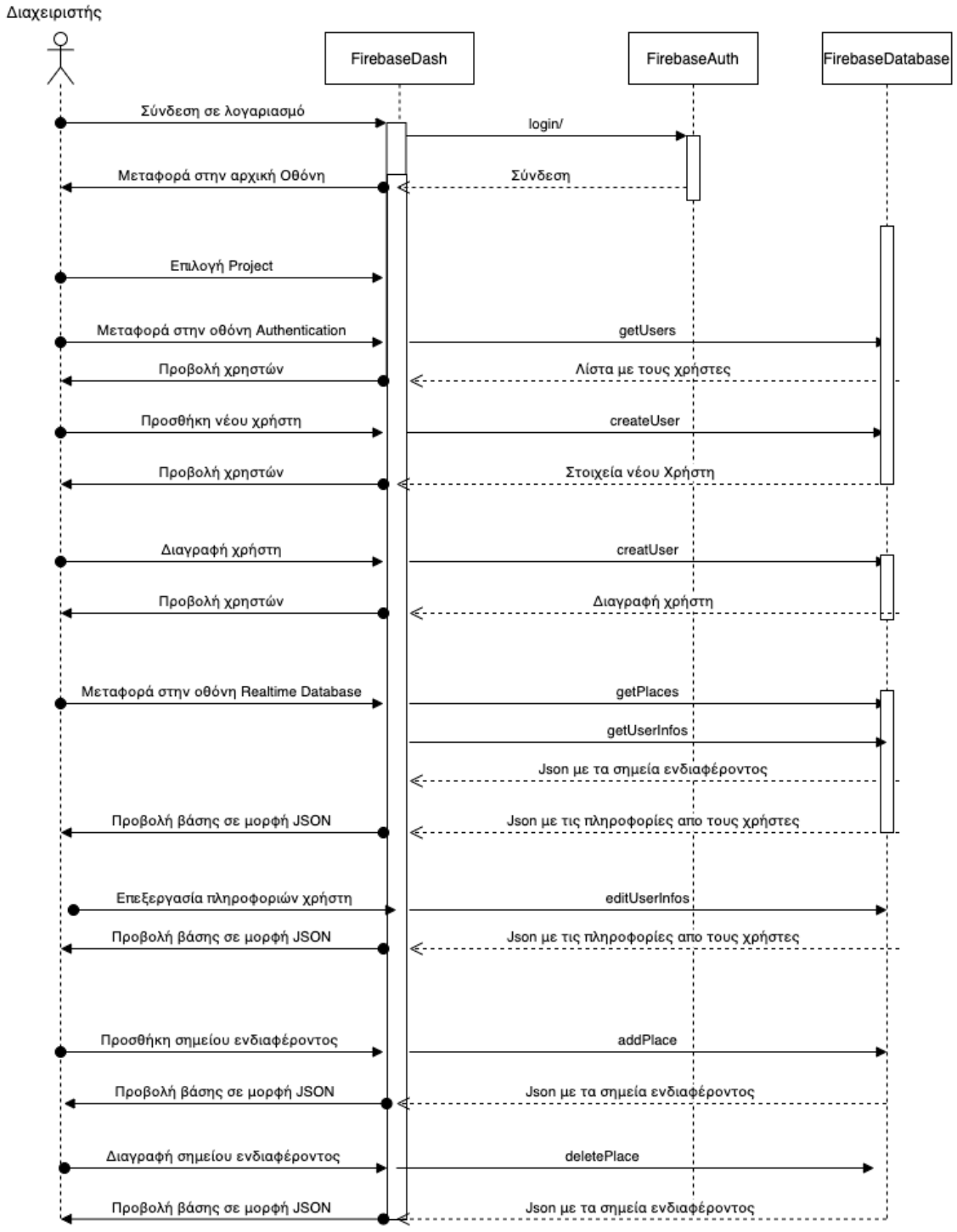
1. Ο διαχειριστης συνδεεται στην πλατφόρμα Firebase
2. Επιλέγει το Project που τον ενδιαφέρει
3. Μεταφέρεται στην οθόνη Realtime Database
4. Αναζητεί το σημείου ενδιαφέροντος
5. Κάνει διαγραφή του σημείου ενδιαφέροντος

4.4 Διαγράμματα αλληλουχίας

4.4.1 Διάγραμμα αλληλουχίας χρήστη



4.4.2 Διάγραμμα αλληλουχίας Διαχειριστή



Κεφάλαιο 5 - Ανάλυση υλοποίησης εφαρμογής

5.1 Δημιουργία εφαρμογής

Για την υλοποιηθηκε της εφαρμογής χρησιμοποιήθηκαν οι βιβλιοθήκες React Native, React Viro και Firebase. Το λειτουργικό σύστημα είναι το MacOS Monterey Version 12.6. Ως ελάχιστη έκδοση Ios για την εφαρμογή επιλέχθηκε η IOS 13.

Η τεχνολογίες React native και React Viro χρησιμοποιήθηκαν για την υλοποίηση της διεπαφή χρήστη (UI). Ως backend χρησιμοποιήθηκε η Firebase. Παρακάτω θα αναλύσουμε περισσότερες τις κύριες τεχνολογίες που χρησιμοποιήθηκαν για την αναπτυξη της εφαρμογής.

5.1.1 React

Το React [46] είναι μια βιβλιοθήκη ανοικτού κώδικα που με την χρήση της γλώσσας προγραμματισμού javascript συμβάλλει στην δημιουργία διεπαφών χρήστη ή αλλιώς το μπροστινό κομμάτι αλληλεπίδρασης (frontend).

Δημιουργήθηκε απο τον προγραμματιστη Jordan Walke τον Μάιο του 2013. Το React παρουσιάστηκε ως έργο ανοιχτού κώδικα και συντηρείται μέχρι και σήμερα, ως επί το πλείστον, από την ομάδα προγραμματιστών του Meta Platforms, Inc. Πρόκειται για ένα από τα πιο σύγχρονα και τεχνολογικά εξελιγμένα εργαλεία για δημιουργία σύνθετων εφαρμογών, που δεν περιορίζονται μόνο στο διαδικτυακό κομμάτι.

Το κύριο χαρακτηριστικό του React είναι ότι έχει τη δυνατότητα να παράγει το DOM δυναμικά, να εκτελεί τις τροποποιήσεις ενός αντιγράφου στη μνήμη και στη συνέχεια να κάνει τη διαδικασία σύγκρισης με την ενημερωμένη έκδοση του DOM, με αποτέλεσμα να προσφέρει μια πολύ γρήγορη εμπειρια στον χρήστη.

5.1.2 React Native

Το React Native[47] είναι ένα framework της JavaScript για την δημιουργία εφαρμογών για κινητά τηλέφωνα (iOS και Android). Βασίζεται στο React, τη βιβλιοθήκη JavaScript του Facebook για τη δημιουργία διεπαφών χρήστη, αλλά αντί να στοχεύει μόνο στην δημιουργία προγραμμάτων περιήγησης, στοχεύει πλατφόρμες για κινητές συσκευές.

Με άλλα λόγια: οι προγραμματιστές ιστού μπορούν τώρα να γράφουν εφαρμογές για κινητά που φαίνονται και αισθάνονται όπως οι εγγενεί. Επιπλέον, επειδή το μεγαλύτερο μέρος του κώδικα που γράφετε μπορεί να μοιραστεί μεταξύ διαφόρων πλατφορμών, το React Native διευκολύνει την ταυτόχρονη ανάπτυξη τόσο για Android όσο και για iOS.

Η μόνη διαφορά με την βιβλιοθήκη React είναι ότι δεν χειρίζεται το DOM αλλά εκτελείται απευθείας από την συσκευή και επικοινωνεί μέσω μιας σειριακής και ασύγχρονης γέφυρας με την εγγενή πλατφόρμα.

5.1.3 Firebase

ο Firebase[48] είναι μια Backend-as-a-Service (Baas) πλατφόρμα που αναπτύχθηκε από την Google. Παρέχει στους προγραμματιστές μια ποικιλία εργαλείων και υπηρεσιών για να τους βοηθήσει να αναπτύξουν ποιοτικές εφαρμογές, να αναπτύξουν τη βάση χρηστών τους και να κερδίσουν κέρδη.

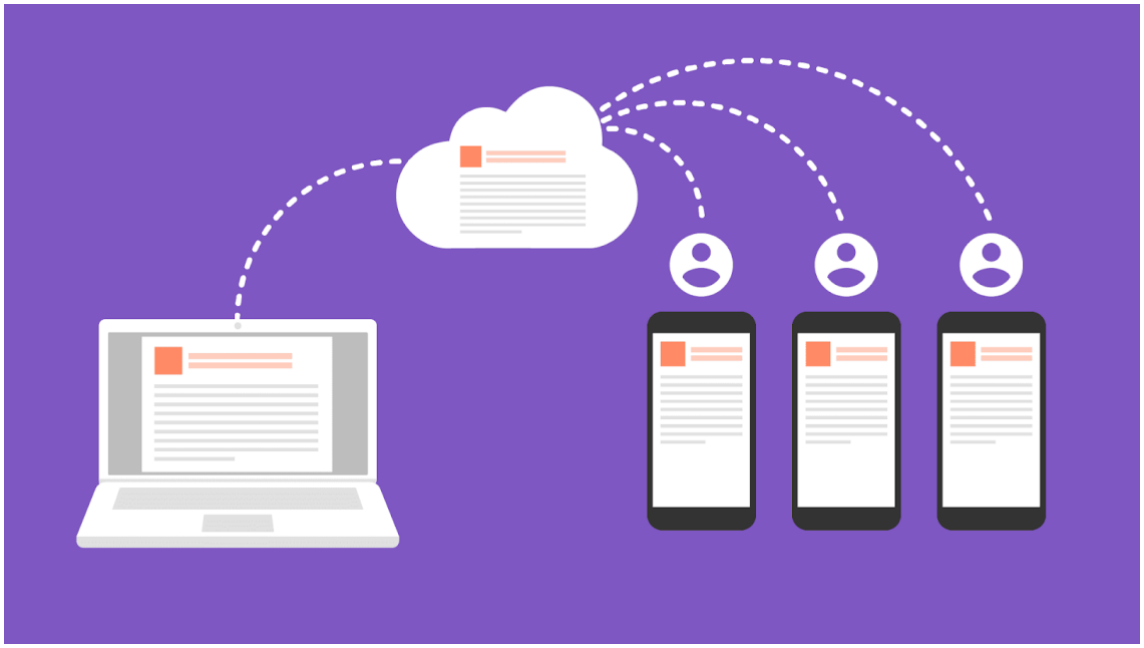
Παρέχει λειτουργίες όπως εξουσιοδότηση χρηστών στην εφαρμογή, βάσεις δεδομένων πραγματικού χρόνου, συναρτήσεις Cloud για την δημιουργία της λογικής στην πλευρά του server, χωρίς να χρειάζεται ο προγραμματιστής να διαχειρίζεται τον server, καθώς και πολλές άλλες λειτουργίες που στοχεύουν στην serverless ανάπτυξη της εφαρμογής.

Το Firebase κατηγοριοποιείται ως πρόγραμμα βάσης δεδομένων NoSQL, το οποίο αποθηκεύει δεδομένα σε έγγραφα που μοιάζουν με JSON.

5.1.4 Firebase Realtime database

Η Firebase Realtime database [49] είναι μία NoSQL βάση δεδομένων στο υπολογιστικό νέφος (Cloud). Τα δεδομένα συγχρονίζονται σε όλους τους πελάτες σε πραγματικό χρόνο και παραμένουν διαθέσιμα ακόμα και όταν μια εφαρμογή είναι εκτός σύνδεσης.

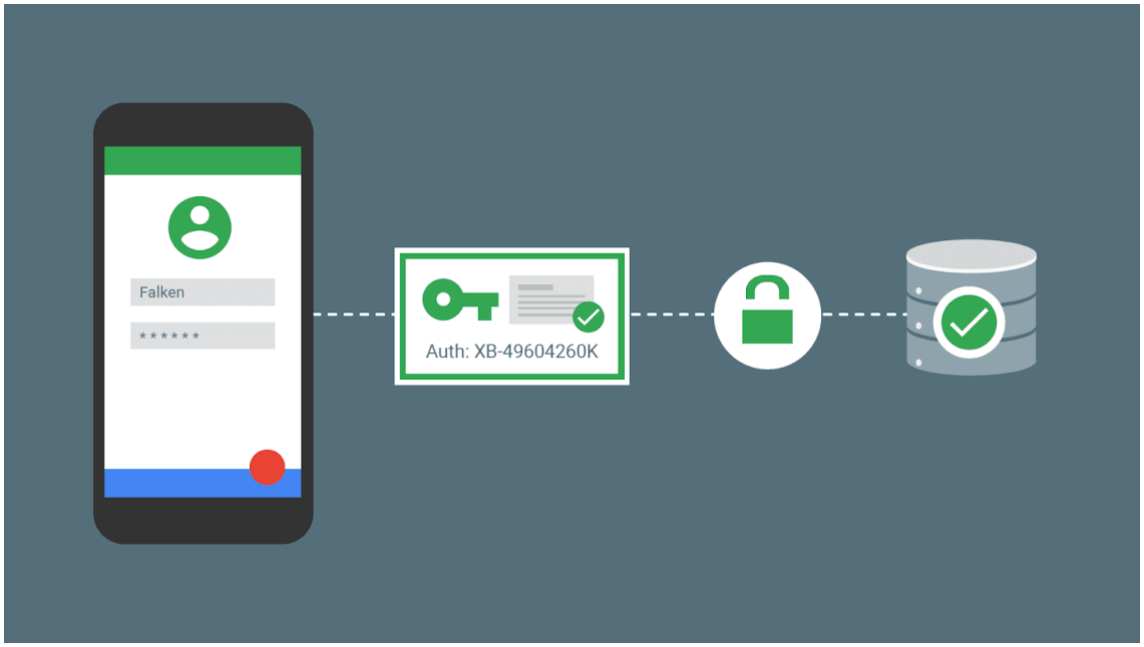
Η βάση αυτή συνοδεύεται από SDK για κινητά και web, ώστε να μπορείτε να δημιουργείτε εφαρμογές χωρίς την ανάγκη διακομιστών, χρησιμοποιώντας το Serverless μοντέλο. Η ασφάλεια και η επικύρωση των δεδομένων γίνεται μέσω των κανόνων ασφάλειας της Realtime Database, που μπορούν να γραφτούν στον πίνακα ελέγχου της βάσης στην πλατφόρμα Firebase.



Εικόνα 18: Firebase Realtime database

5.1.5 Firebase Authentication

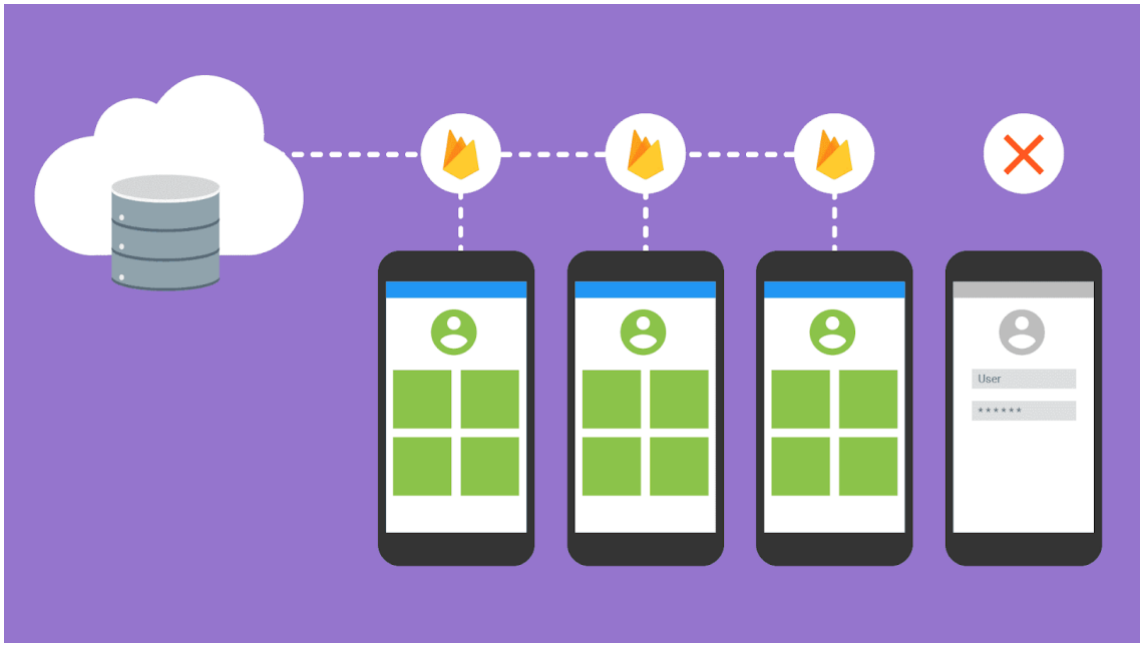
Ο Έλεγχος ταυτότητας Firebase[50] στοχεύει να διευκολύνει τη δημιουργία ασφαλών συστημάτων ελέγχου ταυτότητας, βελτιώνοντας παράλληλα την εμπειρία σύνδεσης και ενσωμάτωσης για τους τελικούς χρήστες. Παρέχει μια λύση ταυτότητας από άκρο σε άκρο, υποστηρίζοντας λογαριασμούς email και κωδικών πρόσβασης, έλεγχο ταυτότητας τηλεφώνου και σύνδεση Google, Twitter, Facebook και GitHub και πολλά άλλα.



Εικόνα 19: Firebase Authentication

5.1.6 Firebase Cloud Storage

Το Cloud Storage[51] έχει σχεδιαστεί για γρήγορη και εύκολη αποθηκεύσει περιεχόμενων που δημιουργείται από τους χρήστες, όπως φωτογραφίες και βίντεο. Το Firebase SDK για Cloud Storage ενσωματώνεται με τον έλεγχο ταυτότητας Firebase για να παρέχει απλό και διαισθητικό έλεγχο πρόσβασης. Μπορείτε να χρησιμοποιήσετε το δηλωτικό μοντέλο ασφαλείας μας για να επιτρέψετε την πρόσβαση με βάση την ταυτότητα χρήστη ή τις ιδιότητες ενός αρχείου, όπως το όνομα, το μέγεθος, τον τύπο περιεχομένου και άλλα μεταδεδομένα.



Εικόνα 20: Firebase Cloud Storage

5.1.7 Εγκατάσταση εργαλείων ανάπτυξης

Αρχικά για να γίνει η ανάπτυξη της εφαρμογής, θα πρέπει να γίνει εγκατάσταση των Node, Watchman, Ruby, Xcode, CocoaPods και react native CLI εργαλείων.
Τα εργαλεία Ruby και Xcode είναι προεγκατεστημένα σε υπολογιστές λειτουργικού συστήματος MacOS.

Η εγκατάσταση του Node και Watchman έγινε με την χρήση του homebrew. Μετά την εγκατάσταση του homebrew εκτελέσαμε την παρακάτω εντολή στο terminal μας:

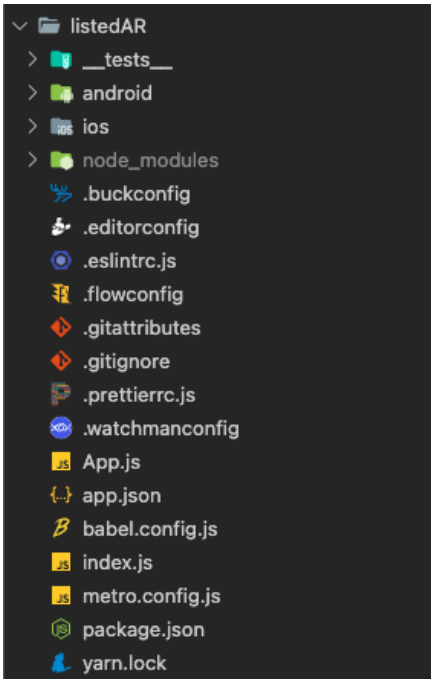
```
brew install node
brew install watchman
```

5.1.8 Δημιουργία Project Listed

Για την δημιουργία της εφαρμογής, το πρώτο βήμα ήταν η επιλογή ονομασίας του Project και η εγκατάσταση των απαραίτητων εργαλείων ανάπτυξης στην επιλεγμένη τοποθεσία. ‘Επειτα εκτελέσαμε την παρακάτω εντολή στο terminal μας:

npx react-native init ListedAR

Ως αποτέλεσμα δημιουργήθηκε η παρακάτω δομή του Project:



Εικόνα 21: Δομή Project

Τέλος για να μπορούμε να τρέξουμε και να κάνουμε ανάπτυξη της εφαρμογής σε πραγματικό χρόνο εκτελέσαμε την εντολή, η οποία ξεκίνησε την εφαρμογή σε έναν προσομοιωτή:

npx react-native run-ios

5.1.9 Εγκατάσταση βιβλιοθηκών

Για την ανάπτυξη της εφαρμογής εκτός από τις προαναφερθείσες τεχνολογίες χρειάστηκαν επιπλέον κάποιες απαραίτητες βιβλιοθήκες που συνδιαστικαν με τις κύριες τεχνολογίες. Παρακάτω θα αναφερθούν μερικές απο αυτες ονομαστικά:

- React Navigation
- fbjs
- React Native Elements
- React Native Geolocation Service

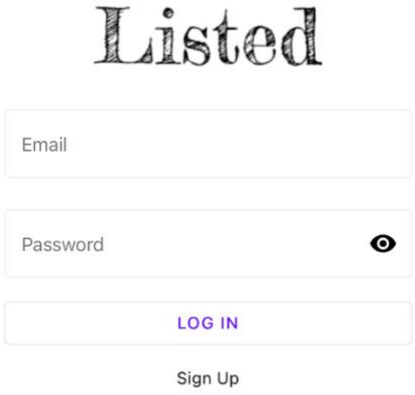
- React Native Google Maps Directions
- React Native Maps
- React Native Permissions

Για την εγκατάσταση αυτών των βιβλιοθηκών εκτελέσαμε την εντολή npm install με πρόθεμα τις ονομασίες των πακέτων.

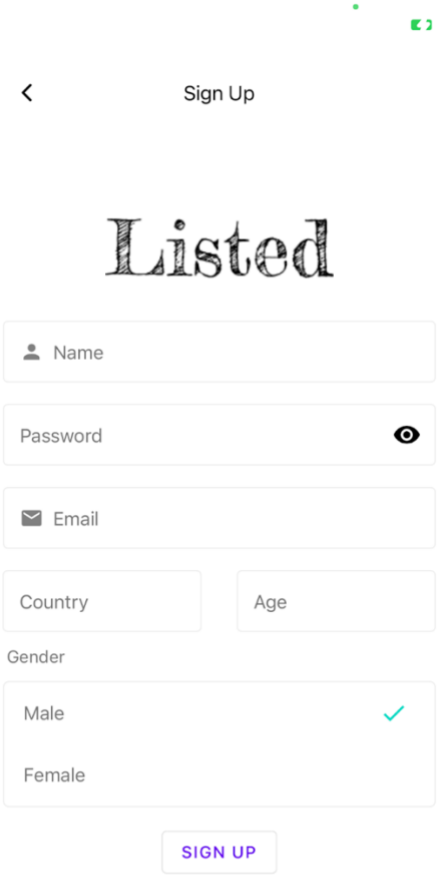
5.2 Διεπαφή χρήστη

5.2.1 Οθόνη σύνδεσης

Αρχικά δημιουργήσαμε την πρώτη οθόνη όπου ο χρήστης θα αντικρίσει μόλις ανοίξει την εφαρμογή. Σε αυτήν την οθόνη ο χρήστης θα μπορεί να δημιουργήσει νέο λογαριασμό ή να συνδεθεί σε έναν λογαριασμό που έχει ήδη δημιουργήσει.
Για να συνδεθεί στον λογαριασμό που έχει δημιουργήσει, θα πρέπει να συμπληρώσει το email και τον κωδικό πρόσβασης.



Εικόνα 22: Οθόνη Login



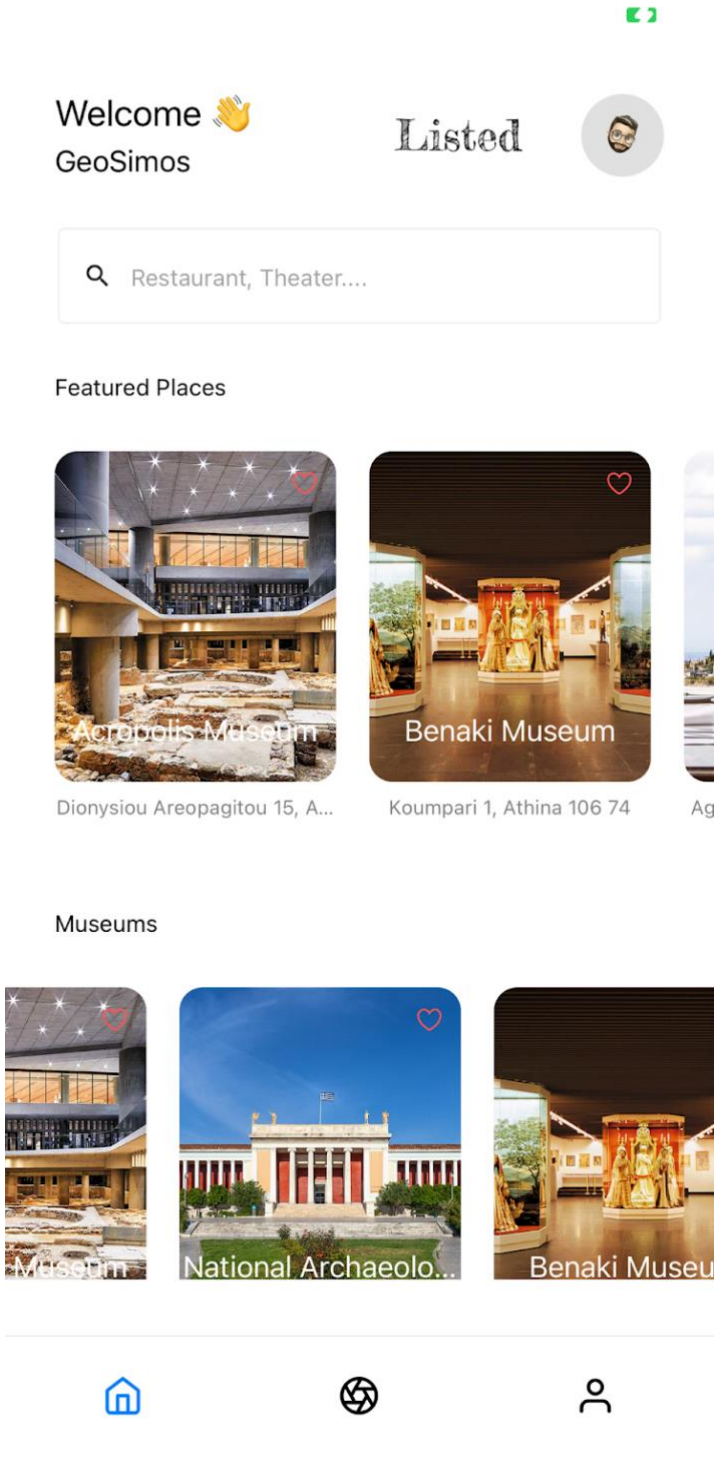
Εικόνα 23: Οθόνη Sign Up

5.2.2 Αρχική οθόνη

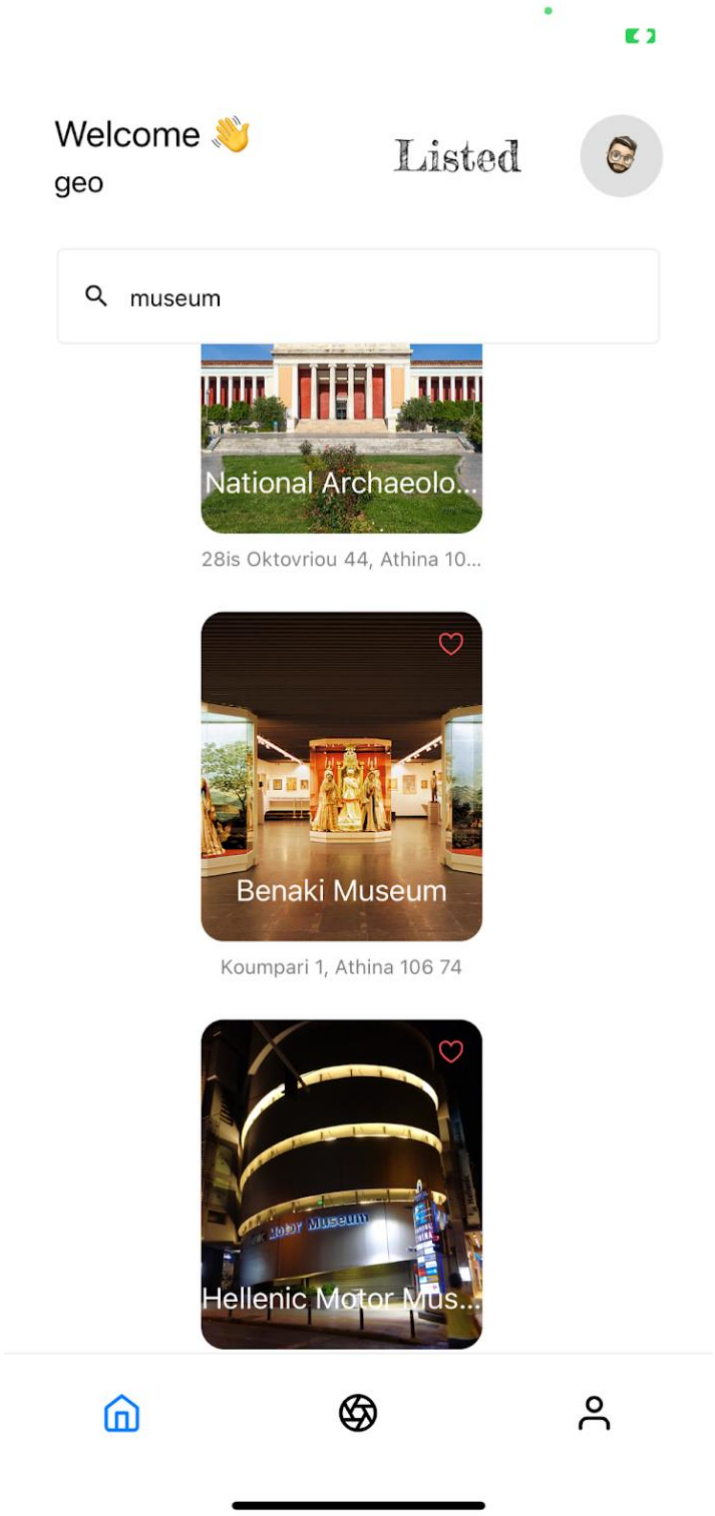
Στην συνέχεια δημιουργήσαμε την αρχική μας οθόνη όπου ο χρήστης θα αντικρίσει μόλις συνδεθεί στην εφαρμογή.

Η αρχική μας οθόνη θα επιτρέπει στον χρήστη να πλοηγηθεί στα ήδη υπάρχων σημεία ενδιαφέροντος. Τα σημεία εμφανίζονται ανα κατηγορία, η πρώτη κατηγορία αποτελείται από τα μέρη τα οποία έχει ο χρήστης τοποθετήσει στα αγαπημένα του. Όλες οι υπόλοιπες κατηγορίες δημιουργούνται δυναμικά κατα την δημιουργία ενός σημείου π.χ Μουσεία.

Ο χρήστης με το πεδίο αναζήτησης μπορεί να αναζητήσει το σημείο που το ενδιαφέρει, το μόνο που χρειάζεται είναι να αρχίσει να πληκτρολογει τον όρο που θα σχετίζεται με το σημειο.



Εικόνα 24: Αρχική Οθόνη

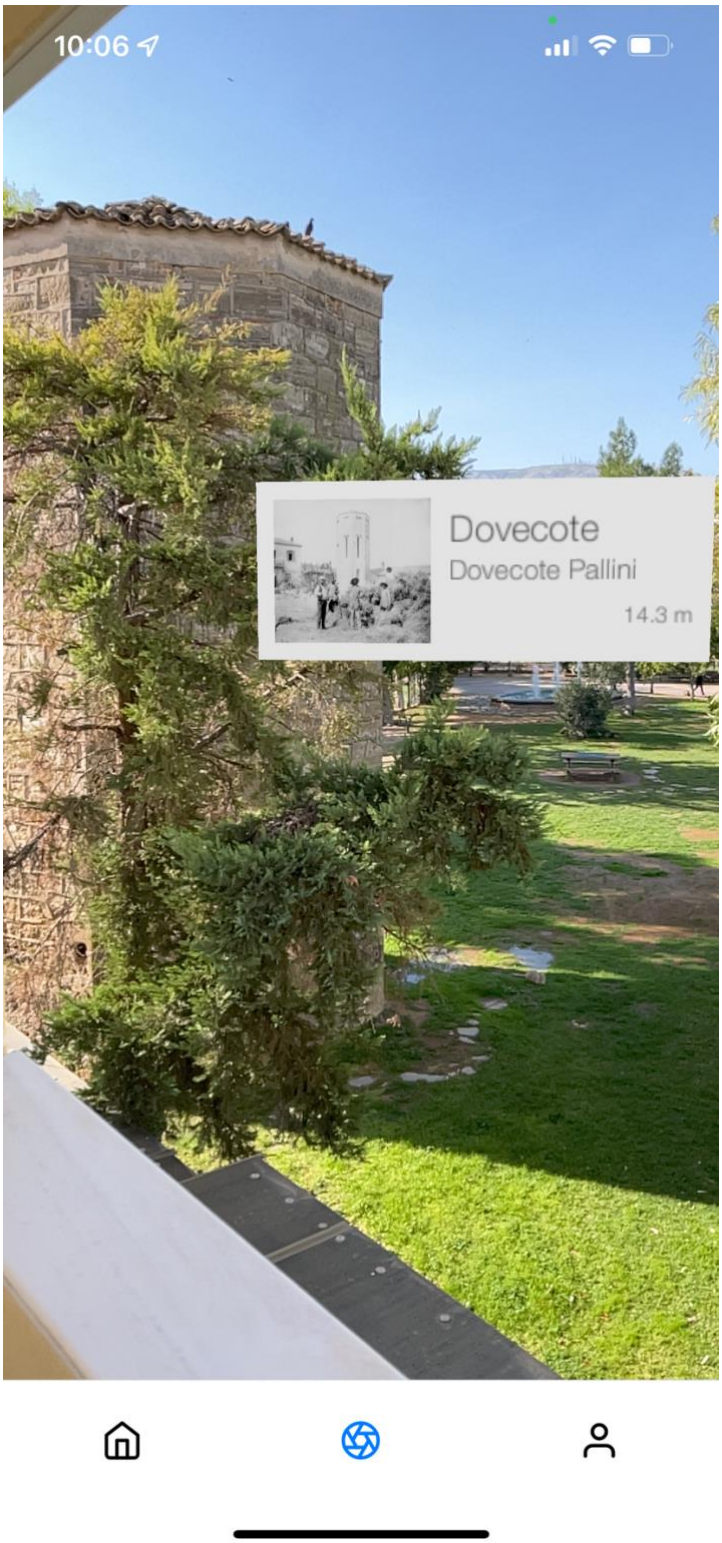


Εικόνα 25: Αρχική Οθόνη με χρήση αναζήτησης

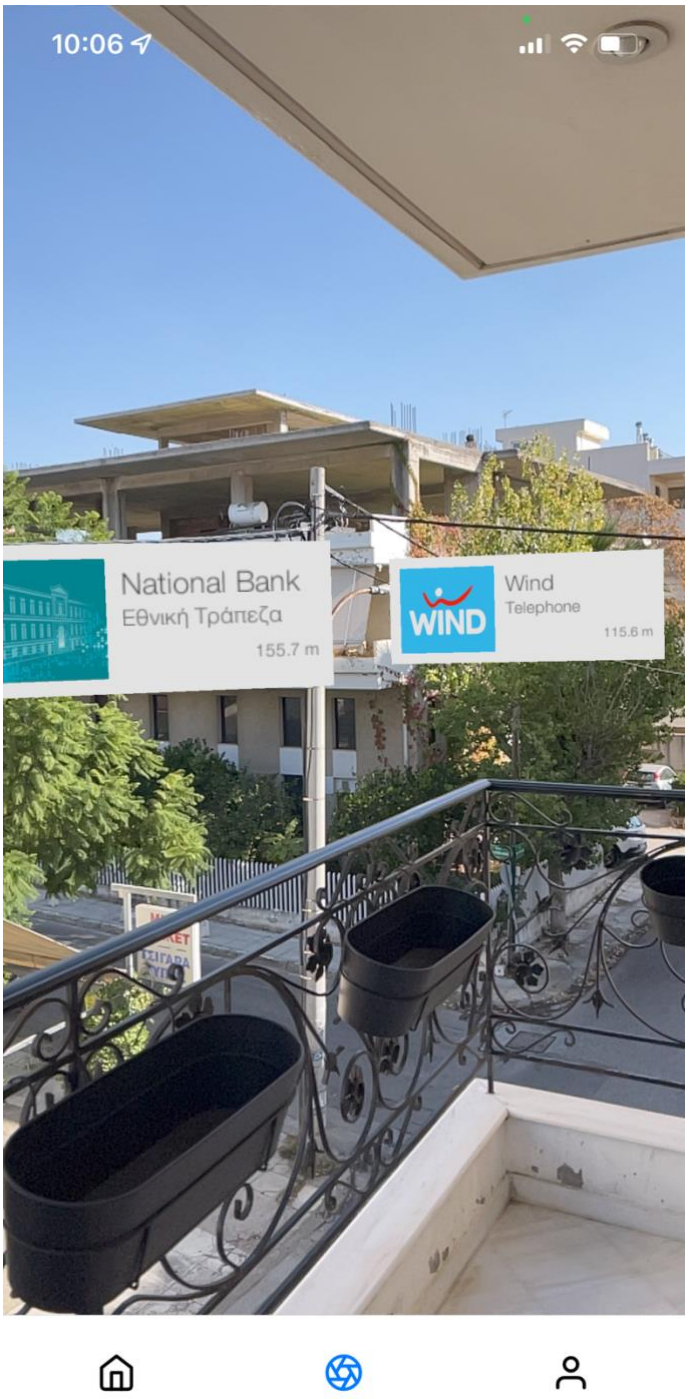
5.2.3 AR οθόνη

Με την χρήση του Tab Menu ο χρήστης μεταφέρεται στην πιο διαδραστική οθόνη της εφαρμογή μας στην οποία έχει γίνει χρήση της τεχνολογία AR. Εφόσον ο χρήστης έχει επιτρέψει την χρήση κάμερας τότε μπορεί να μετακινήσει το κινητό του σε διάφορα σημεία του χώρου και να εξερευνήσει τα διάφορα σημεία τα οποία βρίσκονται κοντά σου σε ακτίνα ενός χιλιομέτρου. Στην συγκεκριμένη σελίδα αναπτύχθηκε αλγόριθμος για να μετατρέψει τις συντεταγμένες του σημείου έτσι ώστε να τοποθετούνται σωστά στο χώρο.

Εικόνα 26: Οθόνη AR 1



Εικόνα 27: Οθόνη AR 2



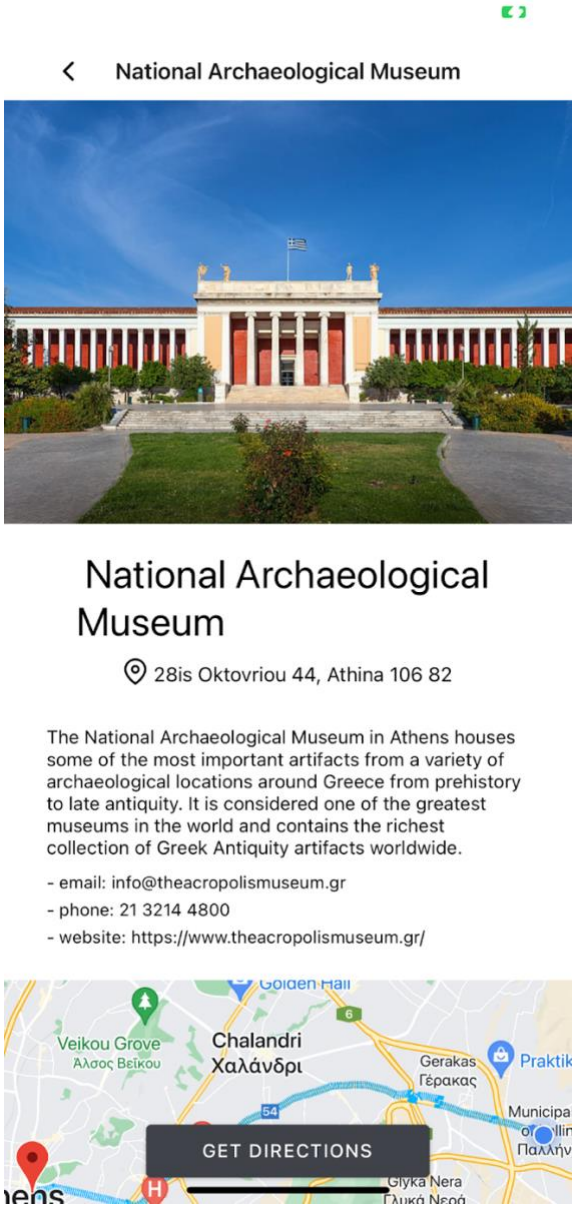
Εικόνα 28: Οθόνη AR 3

Εικόνα 29: Οθόνη AR 4

5.2.4 Οθόνη σημείου ενδιαφέροντος

Η συγκεκριμένη σελίδα εμφανίζει όλες τις απαραίτητες πληροφορίες σχετικά με το σημείο ενδιαφεροντος όπως ονομασία, διεύθυνση, φωτογραφία, περιγραφή, email, τηλέφωνο, ιστοσελίδα καθώς και ο διαδραστικός χάρτης τοποθεσίας.

Ο διαδραστικός χάρτης δείχνει την διαδρομή που ο χρήστης πρέπει να ακολουθήσει έτσι ώστε να φτάσει στον τελικό προορισμό. Πατόντας το κουμπί Get Directions ο χρήστης μεταφέρεται στην εφαρμογή google maps για την πλοηγηση του.

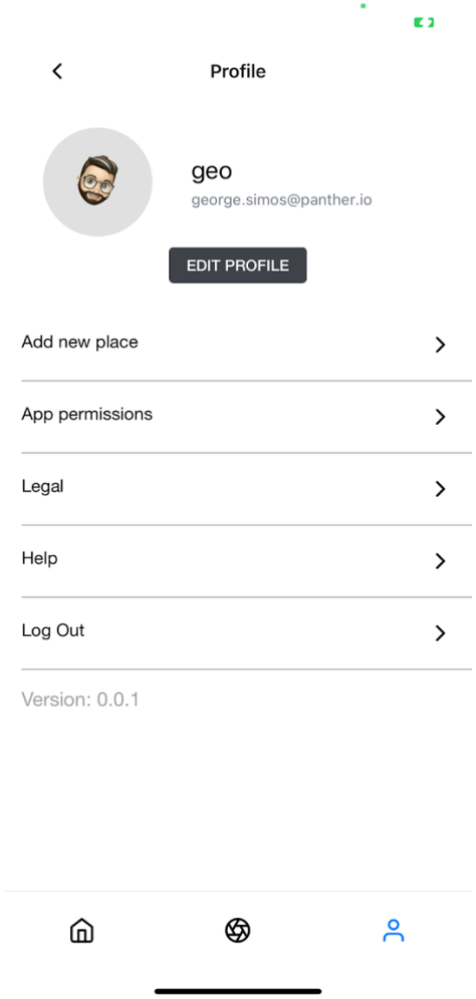


Εικόνα 30: Οθόνη σημείου ενδιαφέροντος

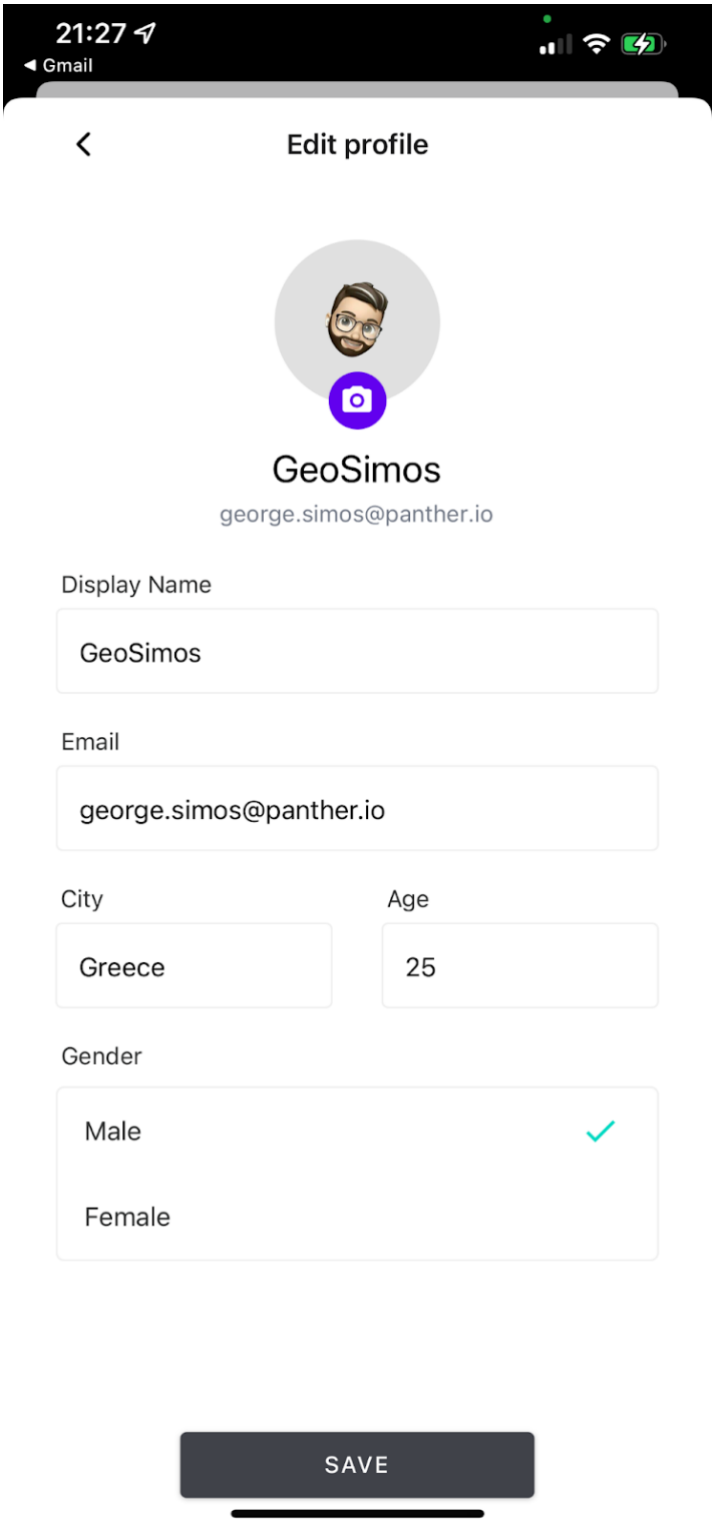
5.2.5 Οθόνης προφίλ η ρυθμίσεων

Στην οθόνη προφίλ ο χρήστης έχει τις παρακάτω δυνατότητες:

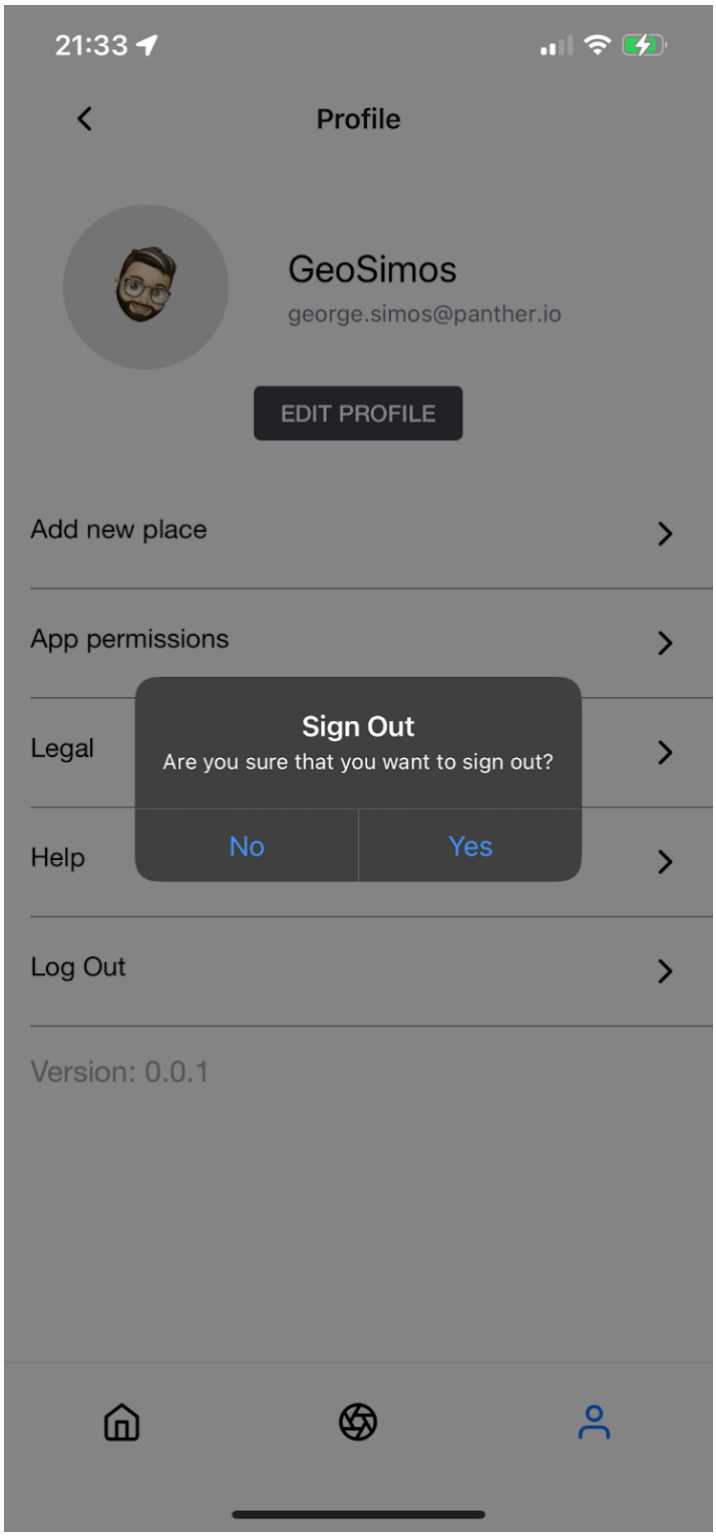
- Να τροποποιήσει τα προσωπηκά του στοιχεία οπως όνομα, email, τοποθεσία, ηλικία και φύλο.
- Να δώσει τα απαραιτητα permissions στην εφαρμογή όπως GPS και κάμερα.
- Να δημιουργήσει ένα καινούριο σημείο ενδιαφέροντος με τις ακόλουθες πληροφορίες:
    - Όνομα
    - Φωτογραφία
    - Περιγραφή
    - Email
    - Τηλέφωνο
    - Ιστοσελίδα
    - Κατηγορία
    - Τοποθεσία
- Να κάνει αποσύνδεση από την εφαρμογή.



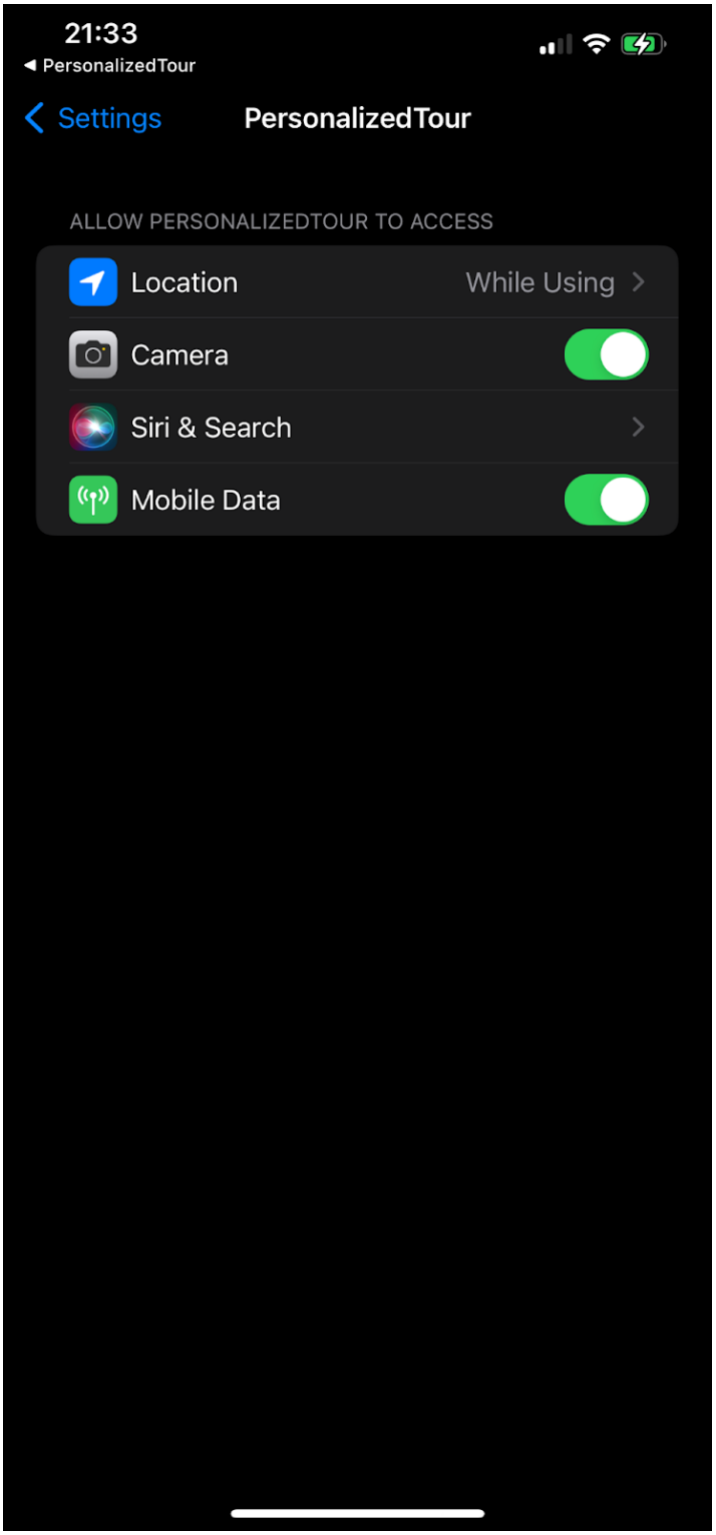
Εικόνα 31: Οθόνης προφίλ η ρυθμίσεων



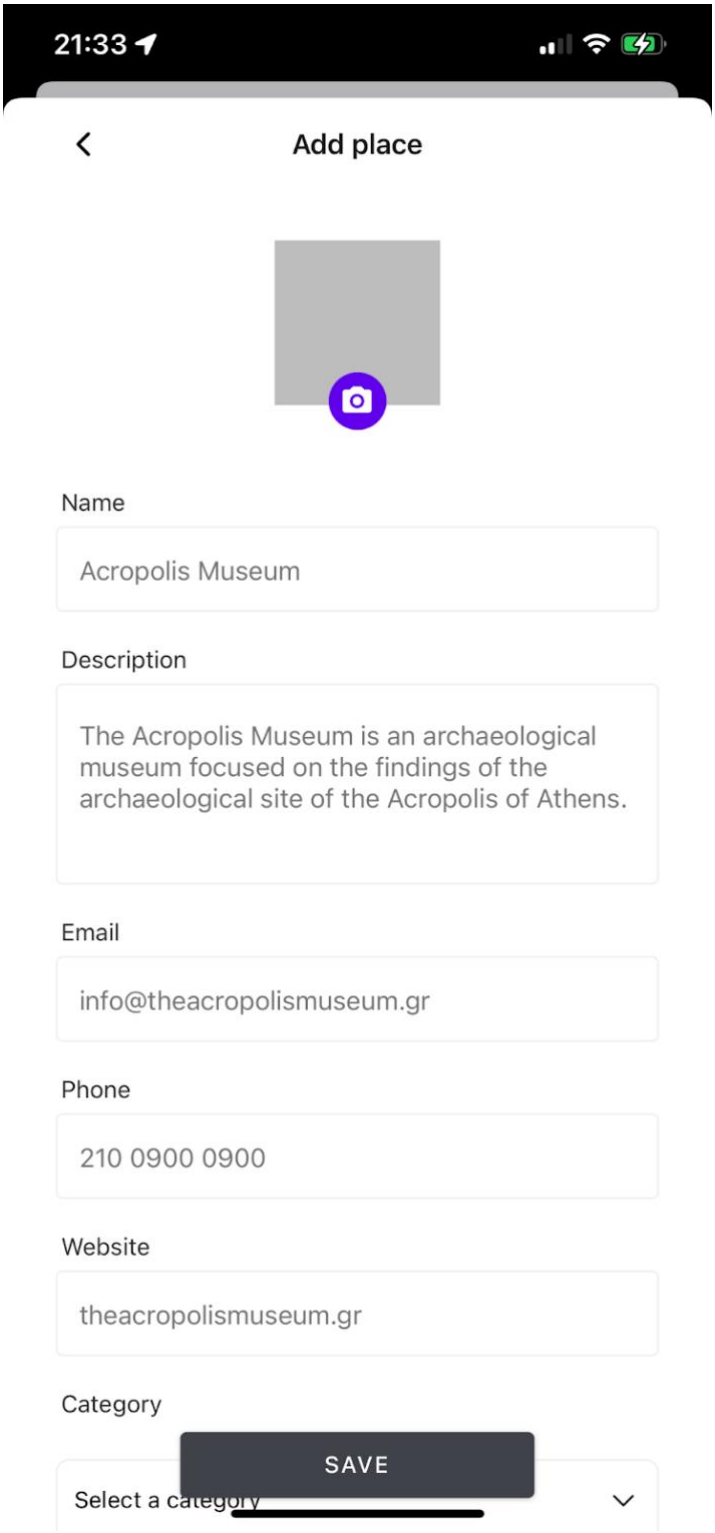
Εικόνα 32: Οθόνη επεξεργασίας προφίλ



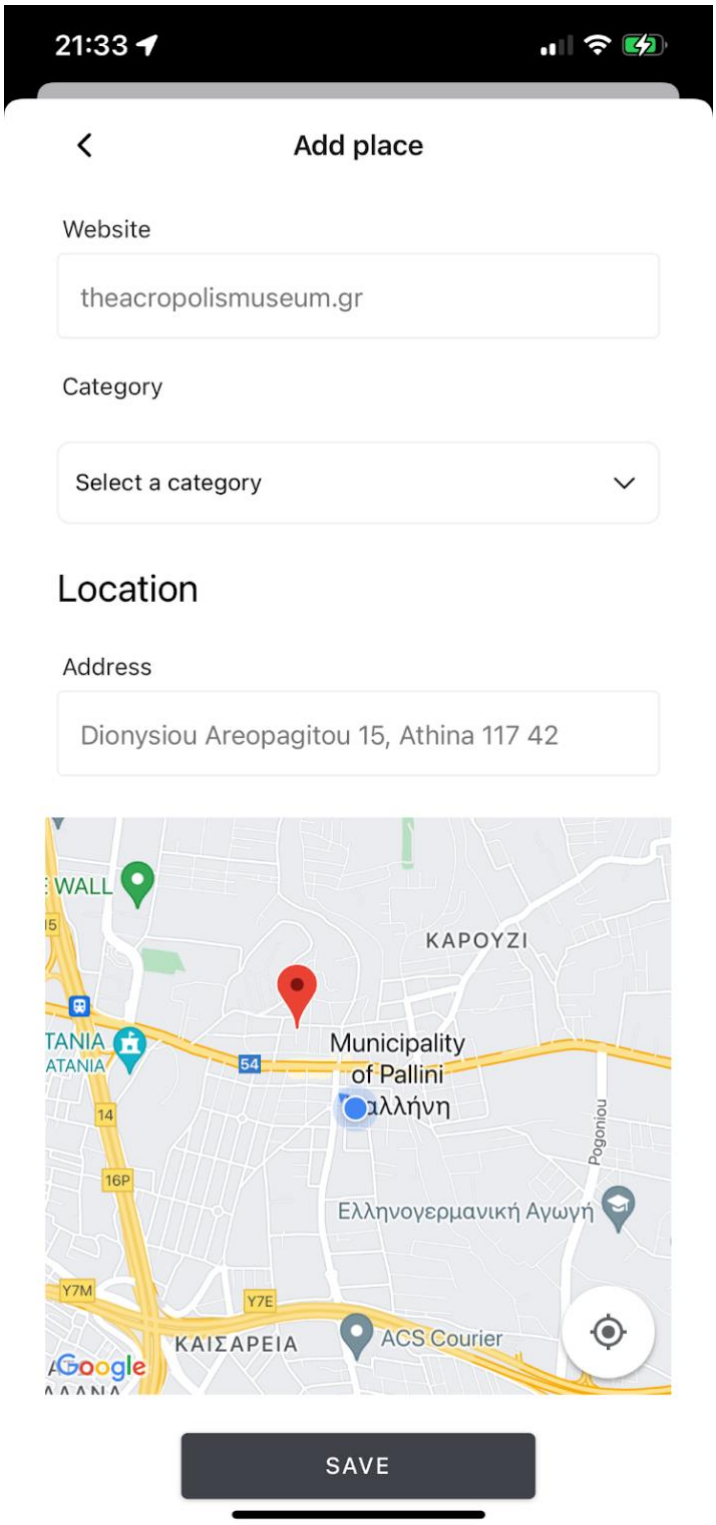
Εικόνα 33: Modal αποσύνδεσης



Εικόνα 34: Οθόνη ρυθμίσεων για Permissions



Εικόνα 35: Προσθήκη σημείου ενδιαφέροντος



Εικόνα 36: Προσθήκη σημείου ενδιαφέροντος

Κεφάλαιο 6 - Περίληψη και Συμπεράσματα

6.1 Περίληψη

Στην συγκεκριμένη διπλωματική εργασία δημιουργήθηκε μια εφαρμογή για την περιήγηση του χρήστη σε σημεία ενδιαφέροντος αλλά και την πλήρη ενημέρωση του με σχετικές πληροφορίες.

Η εφαρμογή αναπτύχθηκε με τέτοιο τρόπο ώστε να είναι εξατομικευμένη στις αναγκες του κάθε χρήστη και επίσης εύκολη στην χρήση. Ο χρήστης έχει την δυνατότητα να ομαδοποίηση σημεία ενδιαφέροντος στα οποία είναι ο μόνος που θα έχει πρόσβαση.

Η υλοποίηση της συγκεκριμένης εφαρμογής έγινε με την αξιοποίηση των πιο σύγχρονων τεχνολογιών με αποτέλεσμα να κάνει την διαδικασία ανάπτυξης πιο ευκολη και ευχαριστη.

Για την επιλογή των τεχνολογιών εγινε σχολαστική έρευνα έτσι ώστε να επιτευχθεί το καλύτερο αποτέλεσμα στο μικρότερο χρόνο. Ετσι επιλέξαμε την βιβλιοθήκη React Native με την δυνατότητα εγγραφης του κώδικα μια φορά έτσι ώστε να μπορούμε να τον μεταγλωττίσουμε σε περισσότερα λειτουργικά συστήματα κινητων.

Η χρήση μια έτοιμης πλατφόρμας όπως η Firebase, ελλάτωσε καταπόλυ τον απαιτούμενο αριθμό resources που θα έπρεπε να αποδοθούν στην υλοποιηση του Project. Το να υπάρχει μία βαλίτσα με τόσο χρήσιμα εργαλεία έτοιμα για χρήση οπως τα Realtime database και authorization έπαιξε μεγάλο ρόλο στην ταχύτητα και ποιότητα του τελικού εργου. Επίσης ο διαχειριστής μέσα από την πλατφόρμα Firebase μπορούσε πάρα πολύ εύκολα να προσθέσει , να τροποποιήσει, να διαγράψει, σημεία ενδιαφέροντος, πολυμεσικό περιεχόμενο, υπερσυνδέσμους, εικονίδια ή ότι άλλο αυτός επιθυμεί.

Παρότι το πιο δυνατό σημείο της εφαρμογής μας ειναι η οθόνη με την χρήση AR, στην οποία παρουσιάζονται όλα τα σημεία ενδιαφέροντος σε μια ακτινα ενος χιλιομέτρου. Παρουσιάστηκε μία δυσκολία στην εύρεση της κατάλληλης AR βιβλιοθηκης. Σχεδόν όλες αυτές οι βιβλιοθήκες που ήταν συμβατες με το React Native δεν είχαν καμία πρόσφατη ενημέρωση με αποτέλεσμα να μην μπορούν να χρησιμοποιηθούν με τις τελευταιες εκδοσεις του React.

Για την περαιτέρω ανάπτυξη της εφαρμογής θα μπορούσαν να προστεθούν περισσότερες οθόνες όπως ένας χάρτης με όλα τα σημεια ενδιαφεροντος με φίλτρο ένα συγκεκριμένο ευρος απόστασης το οποίο θα μπορούσε να το θέσει ο χρήστης. Δυνατοτητα διαμοιρασμου σημείων ενδιαφερόντων με άλλους χρήστες. Προσθήκη

δυνατότητας αναγνωρησεις διαφορών δεικτών όπως μία εικόνα η ένα QR code, με αποτέλεσμα τον εμπλουτισμό με ψηφιακές πληροφορίες του πραγματικού κόσμου.

6.2 Συμπεράσματα

Τα συμπεράσματα που προκύπτουν στο τέλος είναι ότι τα τελευταία χρόνια μέσω τον έξυπνων κινητών συσκευών η Επαυξημένη Πραγματικότητα έχει διείσδυση για τα καλά στην καθημερινότητα των ανθρώπων.

Το επόμενο βήμα είναι αυτές οι εφαρμογές να αρχίζουν να πολλαπλασιάζονται αφού υπάρχει μεγαλύτερη ζήτηση σε όλους τους τομείς όπως της Εκπαίδευσης, του Τουρισμού και άλλους. Έτσι ώστε να έχουμε μια πιο ρεαλιστική αίσθηση του ψηφιακού κόσμου γύρω μας εταιρείες αναπτύσσουν διαφορά σχετικά αξεσουάρ. Σε κάθε περίπτωση αυτές οι τεχνολογίες είναι εδώ και δεν είναι ένα σενάριο επιστημονικής φαντασίας.

Βιβλιογραφία

- [1] Επαυξημένη πραγματικότητα, διαθέσιμο στο https://el.wikipedia.org/wiki/Επαυξημένη_πραγματικότητα - ημερ. ανάκ. 9-8-22.
- [2] Επαυξημένη & εικονικής πραγματικότητα , διαθέσιμο στο https://dynamics.microsoft.com/el-gr/mixed-reality/guides/what-is-augmented-reality-ar/ - ημερ. ανάκ. 9-8-22.
- [3] Augmented reality , διαθέσιμο στο https://en.wikipedia.org/wiki/Augmented_reality - ημερ. ανάκ. 9-8-22.
- [4] A Brief History of Augmented Reality , διαθέσιμο στο https://www.g2.com/articles/history-of-augmented-reality - ημερ. ανάκ. 9-8-22.
- [5] Presence: Teleoperators and Virtual Environments (1992) 1 (3): 279–294. , διαθέσιμο στο https://direct.mit.edu/pvar/article-abstract/1/3/279/58782/EL-Cine-del-Futuro-The-Cinema-of-the-Future?redirectedFrom=fulltext - ημερ. ανάκ. 9-8-22.
- [6] AFIPS '68 (Fall, part I): Proceedings of the December 9-11, 1968, fall joint computer conference, part I.
- [7] Videoplace , διαθέσιμο στο https://en.wikipedia.org/wiki/Videoplace - ημερ. ανάκ. 9-8-22.
- [8] An application of heads-up display technology to manual manufacturing processes , διαθέσιμο στο https://www.researchgate.net/publication/3510119_Augmented_reality_An_application_of_heads-up_display_technology_to_manual_manufacturing_processes - ημερ. ανάκ. 9-8-22.

- [9] Armstrong Laboratory , διαθέσιμο στο https://en.wikipedia.org/wiki/Armstrong_Laboratory - ημερ. ανάκ. 9-8-22.
- [10] The History of Augmented Reality , διαθέσιμο στο https://www.colocationamerica.com/blog/history-of-augmented-reality - ημερ. ανάκ. 9-8-22.
- [11] The Mainstreaming of Augmented Reality: A Brief History , διαθέσιμο στο https://hbr.org/2016/10/the-mainstreaming-of-augmented-reality-a-brief-history - ημερ. ανάκ. 9-8-22.
- [12] Knowledge-Based Augmented Reality , διαθέσιμο στο https://www.researchgate.net/publication/220420450_Knowledge-Based_Augmented_Reality - ημερ. ανάκ. 9-8-22.
- [13] ARToolKit , διαθέσιμο στο https://en.wikipedia.org/wiki/ARToolKit - ημερ. ανάκ. 9-8-22.
- [14] EyeToy , διαθέσιμο στο https://en.wikipedia.org/wiki/EyeToy - ημερ. ανάκ. 9-8-22.
- [15] Wikitude , διαθέσιμο στο https://en.wikipedia.org/wiki/Wikitude - ημερ. ανάκ. 9-8-22.
- [16] Volkswagen develops augmented reality service manual for the XL1 , διαθέσιμο στο https://www.engadget.com/2013-10-01-volkswagen-augmented-reality-ipad-manual-xl1.html - ημερ. ανάκ. 9-8-22.
- [17] Microsoft HoloLens , διαθέσιμο στο https://www.microsoft.com/en-us/hololens - ημερ. ανάκ. 9-8-22.
- [18] Pokemon GO , διαθέσιμο στο https://pokemongolive.com/en/ - ημερ. ανάκ. 9-8-22.
- [19] Metaverse , διαθέσιμο στο https://en.wikipedia.org/wiki/Metaverse - ημερ. ανάκ. 9-8-22.
- [20] Types of AR , διαθέσιμο στο https://digitalpromise.org/initiative/360-story-lab/360-production-guide/investigate/augmented-reality/getting-started-with-ar/types-of-ar - ημερ. ανάκ. 9-8-22.
- [21] AR in Architecture , διαθέσιμο στο https://redshift.autodesk.com/articles/what-is-augmented-reality - ημερ. ανάκ. 15-8-22.
- [22] Morpholio , διαθέσιμο στο https://www.morpholioapps.com/ - ημερ. ανάκ. 15-8-22.
- [23] IKEA Place , διαθέσιμο στο https://www.ikea.com - ημερ. ανάκ. 15-8-22.
- [24] Augmented Reality in Education , διαθέσιμο στο https://online.maryville.edu/blog/augmented-reality-in-education - ημερ. ανάκ. 15-8-22.
- [25] Quiver , διαθέσιμο στο https://quivervision.com/products - ημερ. ανάκ. 15-8-22.
- [26] ingress , διαθέσιμο στο https://www.ingress.com/ - ημερ. ανάκ. 15-8-22.
- [27] kinetisense , διαθέσιμο στο https://apps.microsoft.com/store/detail/kinetisense- ημερ. ανάκ. 15-8-22.

- [28] How augmented reality creates a seamless driving experience , διαθέσιμο στο https://www.tomtom.com/blog/navigation/augmented-reality-creates-seamless-driving-experience/ - ημερ. ανάκ. 15-8-22.
- [29] Athens Olympic Museum , διαθέσιμο στο https://www.greece-is.com/exploring-athens-olympic-museum/ - ημερ. ανάκ. 15-8-22.
- [30] Operating system , διαθέσιμο στο https://en.wikipedia.org/wiki/Operating_system - ημερ. ανάκ. 15-8-22.
- [31] Apple IOS , διαθέσιμο στο https://www.apple.com/ios/ios-16/ - ημερ. ανάκ. 15-8-22.
- [32] Android , διαθέσιμο στο https://www.android.com/ - ημερ. ανάκ. 15-8-22.
- [33] Native Mobile App Development , διαθέσιμο στο https://mdevelopers.com/blog/what-is-a-native-mobile-app-development - ημερ. ανάκ. 16-8-22.
- [34] mobile Web development , διαθέσιμο στο https://developer.mozilla.org/en-US/docs/Web/Guide/Mobile - ημερ. ανάκ. 18-8-22.
- [35] Native vs Hybrid , διαθέσιμο στο https://dzone.com/articles/native-vs-hybrid-vs-cross-platform-how-and-what-to - ημερ. ανάκ. 19-8-22.
- [36] API , διαθέσιμο στο https://en.wikipedia.org/wiki/API - ημερ. ανάκ. 20-8-22.
- [37] API , διαθέσιμο στο https://support.apple.com/el-gr/guide/shortcuts-mac/apd2e30c9d45/mac - ημερ. ανάκ. 20-8-22.
- [38] Backend as a Service , διαθέσιμο στο https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/ - ημερ. ανάκ. 20-8-22.
- [39] User interface , διαθέσιμο στο https://en.wikipedia.org/wiki/User_interface - ημερ. ανάκ. 20-8-22.
- [40] User Interface Design Issues for Easy and Efficient Human Computer Interaction: An Explanatory Approach , διαθέσιμο στο https://www.researchgate.net/publication/294428623_User_Interface_Design_Issues_for_Easy_and_Efficient_Human_Computer_Interaction_An_Explanatory_Approach - ημερ. ανάκ. 20-8-22.
- [41] arkit , διαθέσιμο στο https://developer.apple.com/augmented-reality/arkit/ - ημερ. ανάκ. 20-8-22.
- [42] ARCore , διαθέσιμο στο https://developers.google.com/ar/develop - ημερ. ανάκ. 20-8-22.
- [43] Vuforia , διαθέσιμο στο https://www.ptc.com/en/products/vuforia - ημερ. ανάκ. 20-8-22.
- [44] Vuforia , διαθέσιμο στο https://en.wikipedia.org/wiki/Vuforia_Augmented_Reality_SDK - ημερ. ανάκ. 20-8-22.
- [45] Viro React , διαθέσιμο στο https://viro-community.readme.io/docs/overview - ημερ. ανάκ. 20-8-22.
- [46] React , διαθέσιμο στο https://reactjs.org/ - ημερ. ανάκ. 20-9-22.

- [47] React Native , διαθέσιμο στο https://reactnative.dev/ - ημερ. ανάκ. 20-9-22.
- [48] Firebase , διαθέσιμο στο https://firebase.google.com/ - ημερ. ανάκ. 20-9-22.
- [49] Firebase Realtime Database , διαθέσιμο στο https://firebase.google.com/products/realtime-database - ημερ. ανάκ. 20-9-22.
- [50] Firebase Authentication , διαθέσιμο στο https://firebase.google.com/products/auth - ημερ. ανάκ. 20-9-22.
- [51] Firebase Cloud Storage , διαθέσιμο στο https://firebase.google.com/products/storage - ημερ. ανάκ. 20-9-22.

Παράρτημα Α (Κώδικας)

Κώδικας εφαρμογής React Native

Ο κώδικας εφαρμογής : https://github.com/georgesimos/ListedAR

app/PersonalizedTour/package.json

```
{
 "name": "listedAR",
 "version": "0.0.1",
 "private": true,
 "scripts": {
   "android": "react-native run-android",
   "ios": "react-native run-ios",
   "start": "react-native start",
   "test": "jest",
   "lint": "eslint ."
 },
 "dependencies": {
   "@react-native-async-storage/async-storage": "^1.17.10",
   "@react-native-community/geolocation": "^2.1.0",
   "@react-native-masked-view/masked-view": "^0.2.7",
   "@react-navigation/bottom-tabs": "^6.3.3",
   "@react-navigation/native": "^6.0.12",
   "@react-navigation/native-stack": "^6.8.0",
   "@react-navigation/stack": "^6.2.3",
   "@viro-community/react-viro": "^2.23.0",
   "fbjs": "^3.0.4",
   "firebase": "^9.9.4",
   "proj4": "^2.7.0",
   "react": "17.0.2",
   "react-native": "0.65.1",
   "react-native-dropdown-picker": "^5.4.2",
   "react-native-elements": "^3.1.0",
   "react-native-geolocation-service": "^5.1.1",
   "react-native-gesture-handler": "^2.6.0",
   "react-native-google-maps-directions": "^2.1.1",
   "react-native-google-places-autocomplete": "^2.4.1",
   "react-native-image-picker": "^4.10.0",
   "react-native-maps": ">=1.0.0",
   "react-native-maps-directions": "^1.9.0",
```

```
    "react-native-paper": "^4.12.4",
    "react-native-permissions": "^3.0.3",
    "react-native-reanimated": "^2.10.0",
    "react-native-safe-area-context": "^4.3.3",
    "react-native-screens": "^3.17.0",
    "react-native-toast-message": "^2.1.5",
    "react-native-vector-icons": "^9.2.0"
  },
  "devDependencies": {
    "@babel/core": "7.15.0",
    "@babel/runtime": "7.15.3",
    "@react-native-community/eslint-config": "3.0.0",
    "babel-jest": "27.0.6",
    "eslint": "7.32.0",
    "jest": "27.0.6",
    "metro-react-native-babel-preset": "0.66.2",
    "react-native-codegen": "0.0.7",
    "react-test-renderer": "17.0.2"
  },
  "jest": {
    "preset": "react-native"
  }
}
```

app/PersonalizedTour/app.json

```
{
  "name": "listedAR",
  "displayName": "listedAR"
}
```

app/PersonalizedTour/src/App.js

```
import 'react-native-gesture-handler';
import React, { useContext, useEffect } from 'react';
import { NavigationContainer } from '@react-navigation/native';
import {
  Context as LocationContext,
  Provider as LocationProvider,
} from './context/LocationContext';
import useLocation from './hooks/useLocation';
import RootStack from './navigation/RootStack';

const locationOptions = {
  watch: true,
  accuracy: {
    ios: 'bestForNavigation',
  },
  enableHighAccuracy: true,
  distanceFilter: 10,
  forceRequestLocation: true,
  showsBackgroundLocationIndicator: true,
};

const App = () => {
  const { addCurrentLocation } = useContext(LocationContext);
  const [location] = useLocation(locationOptions);

  useEffect(() => {
```

```
    if (location) {
      addCurrentLocation(location);
    }
  }, [location]);

  return (
    <NavigationContainer>
      <RootStack />
    </NavigationContainer>
  );
};

const Provider = () => (
  <LocationProvider>
    <App />
  </LocationProvider>
);

export default Provider;
```

app/PersonalizedTour/src/components/Map.js

```
import React, { useState, useContext } from 'react';
import { Text, View } from 'react-native';
import { Dimensions, Image, StyleSheet } from 'react-native';
import MapView, { PROVIDER_GOOGLE, Marker } from 'react-native-maps';
import MapViewDirections from 'react-native-maps-directions';
import { GOOGLE_MAPS_API_KEY } from '../configs';
import { Context as LocationContext } from '../context/LocationContext';
import useGooglePlaces from '../hooks/useGooglePlaces';
import carImage from '../assets/car.png';

const { width, height } = Dimensions.get('window');
const ASPECT_RATIO = width / height;
const LATITUDE = 38.00423250286659;
const LONGITUDE = 23.882492126034105;
const LATITUDE_DELTA = 0.02;
const LONGITUDE_DELTA = 0.02 * ASPECT_RATIO;

const styles = StyleSheet.create({
  stretch: {
    width: 30,
    height: 30,
    resizeMode: 'stretch',
  },
});

export default ({ style, options = {} }) => {
  const {
    state: { currentLocation, destination, destinationLatLong },
    addDestinationLatLong,
    addDirections,
  } = useContext(LocationContext);

  const [newPoi, setNewPoi] = useState();

  const [places] = useGooglePlaces(
    currentLocation?.coords?.latitude,
    currentLocation?.coords?.longitude,
```

```
);

const onPressMap = e => {
  setNewPoi({ coordinate: e.nativeEvent.coordinate });
};

const onPoiClick = e => {
  const poi = e.nativeEvent;
  console.log(poi);
};

if (!currentLocation) {
  return null;
}

console.log({ currentLocation, destination, destinationLatLong, places });

return (
  <MapView
    style={style}
    provider={PROVIDER_GOOGLE}
    initialRegion={{
      latitude: LATITUDE,
      longitude: LONGITUDE,
      latitudeDelta: LATITUDE_DELTA,
      longitudeDelta: LONGITUDE_DELTA,
    }}
    showsUserLocation
    showsMyLocationButton
    onPress={e => onPressMap(e)}
    onPoiClick={onPoiClick}
    {...options}>
    {places.length > 0 &&
      places.map(place => (
        <Marker
          key={place.id}
          icon={place.icon}
          coordinate={{ latitude: place.lat, longitude: place.lng }}
          onPress={e => {
            const {
              nativeEvent: { coordinate },
            } = e;
            console.log(coordinate);
          }}>
          <View>
            <Image
              source={{
                uri: place.icon,
              }}
              style={styles.stretch}
            />
            <Text>{place.title}</Text>
          </View>
        </Marker>
      ))}
    {newPoi && (
      <Marker
        coordinate={newPoi.coordinate}
        anchor={{ x: 0.5, y: 0.5 }}
        image={carImage}
      />
    )}
```

```
      {destinationLatLong && (
        <Marker
          coordinate={destinationLatLong}
          anchor={{ x: 0.5, y: 0.5 }}
          image={carImage}
        />
      )}
      {destination && (
        <MapViewDirections
          origin={{
            latitude: currentLocation.coords.latitude,
            longitude: currentLocation.coords.longitude,
          }}
          destination={destination.description}
          apikey={GOOGLE_MAPS_API_KEY}
          strokeWidth={6}
          strokeColor="#00b2ff"
          lineDashPattern={[30, 20]}
          geodesic={true}
          optimizeWaypoints={true}
          precision="high"
          mode="WALKING"
          onStart={params => {
            console.log(JSON.stringify(params));
          }}
          onReady={result => {
            console.log({ result });
            const { coordinates } = result;
            addDirections(result);
            addDestinationLatLong(coordinates[coordinates.length - 1]);
          }}
          onError={errorMessage => {
            console.log('GOT AN ERROR', errorMessage);
          }}
        />
      )}
    </MapView>
  );
};
```

app/PersonalizedTour/src/components/PlaceCard.js

```
import React, { useState } from 'react';
import { View, Text, Image, StyleSheet, TouchableOpacity } from 'react-
native';
import Icon from 'react-native-vector-icons/Ionicons';
import { auth, database } from '../firebase/config';
import { ref, push, remove, onValue } from 'firebase/database';
import { useNavigation } from '@react-navigation/native';

const PlaceCard = ({ data }) => {
  const [isFavorite, setIsFavorite] = useState(false);
  const [favoriteKey, setFavoriteKey] = useState();
  const navigation = useNavigation();
  const user = auth.currentUser;

  const addFavorites = () => {
    const reference = ref(database, 'users/' + user.uid + '/favorites');
    push(reference, {
```

```
    ...data,
  }).then(response => {
    console.log({ response });
  });
};

const removeFavorites = async key => {
  remove(ref(database, 'users/' + user.uid + '/favorites/' + key));
};

const getUserFavorites = React.useCallback(() => {
  onValue(
    ref(database, 'users/' + user.uid + '/favorites'),
    querySnapShot => {
      const result = querySnapShot.val() || {};

      for (let key in result) {
        if (result[key].id === data.id) {
          setFavoriteKey(key);
          setIsFavorite(true);
        }
      }
    },
  );
}, [data.id, user.uid]);

const clickLike = () => {
  getUserFavorites();

  if (isFavorite) {
    removeFavorites(favoriteKey);
    setIsFavorite(false);
  } else {
    addFavorites();
  }
};

const onPlaceClick = () => {
  navigation.navigate('PlaceDetailsScreen', data);
};

React.useEffect(() => {
  getUserFavorites();
}, [getUserFavorites]);

return (
  <TouchableOpacity onPress={onPlaceClick}>
    <View style={styles.mainCard}>
      <View style={styles.cardInfosFrame}>
        <Image
          resizeMode="cover"
          source={{ uri: data.image }}
          style={styles.listingCoverImage}
        />
        <View style={styles.placeTitleFrame}>
          <Text numberOfLines={1} style={styles.placeTitleText}>
            {data.name}
          </Text>
        </View>
        <View style={styles.heartIconFrame}>
          <TouchableOpacity onPress={clickLike}>
```

```
            <Icon
              size={18}
              style={styles.heartIcon}
              name={isFavorite ? 'heart' : 'heart-outline'}
              color="#ff5357"
            />
          </TouchableOpacity>
        </View>
      </View>

      <View style={styles.cardBottomLocationFrame}>
        <Text numberOfLines={1} style={styles.cardBottomLocationText}>
          {data?.address}
        </Text>
      </View>
    </View>
  </TouchableOpacity>
 );
};

const styles = StyleSheet.create({
 mainCard: {
   width: 170,
   marginBottom: 25,
   marginRight: 20,
 },
 cardInfosFrame: {
   width: 170,
   height: 200,
   alignItems: 'center',
   justifyContent: 'center',
   marginRight: 20,
   borderRadius: 14,
   overflow: 'hidden',
 },
 listingCoverImage: {
   width: '100%',
   height: '100%',
 },
 placeTitleFrame: {
   position: 'absolute',
   width: '100%',
   height: '100%',
   justifyContent: 'flex-end',
   alignItems: 'center',
 },
 placeTitleText: {
   marginBottom: 20,
   color: 'white',
   fontSize: 18,
   // fontFamily: 'MonumentExtended-Regular',
 },
 heartIconFrame: {
   zIndex: 2,
   position: 'absolute',
   width: '100%',
   height: '100%',
   justifyContent: 'flex-start',
   alignItems: 'flex-end',
 },
 heartIcon: {
   marginRight: 10,
   marginTop: 10,
```

```
  },
  cardBottomLocationFrame: {
    width: 170,
    justifyContent: 'center',
    alignItems: 'center',
  },
  cardBottomLocationText: {
    fontSize: 12,
    textAlign: 'center',
    marginTop: 8,
    color: 'gray',
    // fontFamily: 'Roboto-Light',
  },
});

export default PlaceCard;
```

app/PersonalizedTour/src/context/LocationContext.js

```
import createDataContext from './createDataContext';

const INITIAL_STATE = {
  initialLocation: null,
  currentLocation: null,
  destination: null,
  destinationLatLong: null,
  directions: null,
};

const locationReducer = (state, action) => {
  switch (action.type) {
    case 'ADD_CURRENT_LOCATION':
      return {
        ...state,
        currentLocation: action.payload,
        initialLocation:
          ? state.initialLocation
          : action.payload,
      };
    case 'ADD_DESTINATION':
      return { ...state, destination: action.payload };
    case 'ADD_DESTINATION_LAT_LONG':
      return { ...state, destinationLatLong: action.payload };
    case 'ADD_DIRECTIONS':
      return { ...state, directions: action.payload };

    case 'RESET':
      return {
        ...state,
        ...INITIAL_STATE,
      };
    default:
      return state;
  }
};

const addCurrentLocation = dispatch => payload =>
  dispatch({ type: 'ADD_CURRENT_LOCATION', payload });
```

```
const addDestination = dispatch => payload =>
 dispatch({ type: 'ADD_DESTINATION', payload });

const addDestinationLatLong = dispatch => payload =>
 dispatch({ type: 'ADD_DESTINATION_LAT_LONG', payload });

const addDirections = dispatch => payload =>
 dispatch({ type: 'ADD_DIRECTIONS', payload });

const reset = dispatch => () => {
 dispatch({ type: 'RESET' });
};

export const { Context, Provider } = createDataContext(
 locationReducer,
 {
   addCurrentLocation,
   addDestination,
   addDestinationLatLong,
   addDirections,
   reset,
 },
 { INITIAL_STATE },
);
```

app/PersonalizedTour/src/firebase/auth.js

```
import {
 createUserWithEmailAndPassword,
 getAuth,
 signInWithEmailAndPassword,
 signOut,
 updateProfile,
} from 'firebase/auth';
import { app } from './config';

const auth = getAuth(app);

export const signup = (email, password) =>
 createUserWithEmailAndPassword(auth, email, password);

export const login = (email, password) =>
 signInWithEmailAndPassword(auth, email, password);

export const logout = () => signOut(auth);

export const updateUserProfile = (user, { displayName, photoURL }) =>
 updateProfile(user, { displayName, photoURL });
```

app/PersonalizedTour/src/firebase/config.js

```
import { initializeApp } from 'firebase/app';
import { getAuth } from 'firebase/auth';
import { getDatabase } from 'firebase/database';
```

```
const firebaseConfig = {
 apiKey: 'api_key',
 authDomain: 'domain.firebaseapp.com',
 projectId: 'personalizedtours',
 storageBucket: 'bucket',
 messagingSenderId: '104209557237',
 appId: '1:104209560232:web:ad205dd4705fa03ee08435',
 measurementId: 'G-FDMV3KC195',
 databaseURL: 'https://personalizedtours-Papeisd-rtdb.firebaseio.com',
};

const app = initializeApp(firebaseConfig);
const auth = getAuth(app);
const database = getDatabase(app);

export { app, auth, database };
```

app/PersonalizedTour/src/firebase/database.js

```
import { getDatabase, ref, set } from 'firebase/database';
import { app } from './config';

const database = getDatabase(app);

export const setUserInfo = (userId, data) => {
 const reference = ref(database, 'users/' + userId + '/info');
 set(reference, data);
};
```

app/PersonalizedTour/src/firebase/storage.js

```
import { getDownloadURL, getStorage, ref, uploadBytes } from
'firebase/storage';
import { updateUserProfile } from './auth';
import { app } from './config';

const storage = getStorage(app);

export const uploadProfilePhoto = async (file, currentUser, setLoading) => {
 const fileRef = ref(storage, currentUser.uid + '.png');

 setLoading(true);

 await uploadBytes(fileRef, file);
 const photoURL = await getDownloadURL(fileRef);

 updateUserProfile(currentUser, { photoURL });

 setLoading(false);
};

export const uploadPlacePhoto = async (file, fileName) => {
 const fileRef = ref(storage, 'images/' + fileName);

 const metadata = {
```

```
    contentType: 'image/jpeg',
  };

  const snapshot = await uploadBytes(fileRef, file, metadata);
  console.log({ snapshot });
  return getDownloadURL(fileRef);
};
```

app/PersonalizedTour/src/hooks/useAuth.js

```
import { getAuth, onAuthStateChanged } from 'firebase/auth';
import { useState, useEffect } from 'react';
import { app } from '../firebase/config';

const auth = getAuth(app);

export default () => {
  const [currentUser, setCurrentUser] = useState();

  useEffect(() => {
    const unsub = onAuthStateChanged(auth, user => setCurrentUser(user));
    return unsub;
  }, []);

  return currentUser;
};
```

app/PersonalizedTour/src/hooks/useLocation.js

```
import { useState, useEffect, useRef } from 'react';
import Geolocation from 'react-native-geolocation-service';

const useLocation = (props = {}) => {
  const [location, setLocation] = useState();
  const [error, setError] = useState();
  const locationWatchId = useRef(null);

  const clearLocationWatch = () =>
    locationWatchId.current && Geolocation.clearWatch(locationWatchId.current);

  useEffect(() => {
    const { watch, ...options } = props;

    if (watch) {
      locationWatchId.current = Geolocation.watchPosition(
        setLocation,
        setError,
        options,
      );
    } else {
      Geolocation.getCurrentPosition(setLocation, setError, options);
    }
    return clearLocationWatch;
  }, [props]);
```

```
 return Object.assign([location, error, clearLocationWatch], {
   location,
   error,
   clearLocationWatch,
 });
};

export default useLocation;
```

app/PersonalizedTour/src/hooks/usePermissions.js

```
import { useState, useEffect } from 'react';
import { check, PERMISSIONS, request, RESULTS } from 'react-native-
permissions';

export default () => {
 const [error, setError] = useState(null);

 useEffect(() => {
   check(PERMISSIONS.IOS.CAMERA)
     .then(result => {
       switch (result) {
         case RESULTS.UNAVAILABLE:
           console.log(
             'This feature is not available (on this device / in this
context)',
           );
           break;
         case RESULTS.DENIED:
           request(PERMISSIONS.IOS.CAMERA).then(result => {
             // …
           });
           break;
         case RESULTS.LIMITED:
           console.log('The permission is limited: some actions are
possible');
           break;
         case RESULTS.GRANTED:
           console.log('The permission is granted');
           break;
         case RESULTS.BLOCKED:
           console.log('The permission is denied and not requestable
anymore');
           break;
       }
     })
     .catch(err => setError(err));
 }, []);

 return Object.assign([error], { error });
};
```

app/PersonalizedTour/src/hooks/usePlaces.js

```
import React from 'react';
import { onValue, ref } from 'firebase/database';
```

```
import { database } from '../firebase/config';

const usePlaces = () => {
 const [places, setPlaces] = React.useState([]);
 const [placesByCategory, setPlacesByCategory] = React.useState();

 React.useEffect(() => {
   if (places.length) {
     return;
   }

   const reference = ref(database, 'places');
   onValue(reference, snapshot => {
     const placesArr = [];
     snapshot.forEach(childSnapshot => {
       const childKey = childSnapshot.key;
       const childData = childSnapshot.val();
       placesArr.push({ ...childData, id: childKey });
     });

     const grouped = placesArr.reduce((catetories, item) => {
       const category = catetories[item.category] || [];
       return {
         ...catetories,
         [item.category]: [...category, item],
       };
     }, {});

     setPlacesByCategory(grouped);
     setPlaces(placesArr);
   });
 }, [places]);

 return Object.assign([places, placesByCategory], {
   places,
   placesByCategory,
 });
};

export default usePlaces;
```

app/PersonalizedTour/src/navigation/HomeStack.js

```
import React from 'react';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import HomeScreen from '../screens/HomeScreen';

const Stack = createNativeStackNavigator();

export default () => {
 return (
   <Stack.Navigator
     screenOptions={{
       headerShown: false,
     }}>
     <Stack.Screen name="Home" component={HomeScreen} />
   </Stack.Navigator>
 );
```

```
};
```

app/PersonalizedTour/src/navigation/ProfileStack.js

```
import React from 'react';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import ProfileScreen from '../screens/ProfileScreen';
import NavigationBack from '../components/NavigationBack';

const Stack = createNativeStackNavigator();

export default () => {
 return (
   <Stack.Navigator
     screenOptions={{
       headerLeft: () => <NavigationBack />,
     }}>
     <Stack.Screen name="Profile" component={ProfileScreen} />
   </Stack.Navigator>
 );
};
```

app/PersonalizedTour/src/navigation/RootStack.js

```
import React from 'react';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import TabNavigation from './TabsNavgation';
import LoginScreen from '../screens/LoginScreen';
import SignUpScreen from '../screens/SignUpScreen';
import AuthControl from '../screens/AuthControl';
import ModalEditProfile from '../screens/ModalEditProfile';
import ModalAddPlace from '../screens/ModalAddPlace';
import PlaceDetailsScreen from '../screens/PlaceDetailsScreen';
import NavigationBack from '../components/NavigationBack';

export const Stack = createNativeStackNavigator();

export default () => (
 <Stack.Navigator>
   <Stack.Group
     screenOptions={{
       headerShown: false,
     }}>
     <Stack.Screen name="AuthControl" component={AuthControl} />
     <Stack.Screen name="Login" component={LoginScreen} />
     <Stack.Screen name="SignUp" component={SignUpScreen} />
     <Stack.Screen name="TabNavigation" component={TabNavigation} />
   </Stack.Group>
   <Stack.Group
     screenOptions={{
       headerLeft: () => <NavigationBack />,
     }}>
     <Stack.Screen name="PlaceDetailsScreen" component={PlaceDetailsScreen} />
   </Stack.Group>
   <Stack.Group
     screenOptions={{
       presentation: 'modal',
```

```
        headerLeft: () => <NavigationBack />,
      }}>
      <Stack.Screen
        options={{ title: 'Edit profile' }}
        name="ModalEditProfile"
        component={ModalEditProfile}
      />
      <Stack.Screen
        options={{ title: 'Add place' }}
        name="ModalAddPlace"
        component={ModalAddPlace}
      />
    </Stack.Group>
  </Stack.Navigator>
);
```

app/PersonalizedTour/src/navigation/TabsNavgation.js

```
import React from 'react';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import Icon from 'react-native-vector-icons/Feather';
import ARScreen from '../screens/ARScreen';
import { Dimensions } from 'react-native';
import HomeStack from './HomeStack';
import ProfileStack from './ProfileStack';
// import GoogleMapScreen from '../screens/GoogleMapScreen';

const Tab = createBottomTabNavigator();

const TabsNavgation = () => (
  <Tab.Navigator
    screenOptions={{
      headerShown: false,
      tabBarShowLabel: false,
      tabBarStyle: {
        height: Dimensions.get('window').height / 9,
      },
    }}>
    <Tab.Screen
      name="HomeStack"
      component={HomeStack}
      options={{
        tabBarIcon: ({ focused, color, size }) => (
          <Icon name="home" color={focused ? color : 'black'} size={size} />
        ),
      }}
    />
    {/* <Tab.Screen
      name="Map"
      component={GoogleMapScreen}
      options={{
        tabBarIcon: ({ focused, color, size }) => (
          <Icon name="map" color={focused ? color : 'black'} size={size} />
        ),
      }}
    /> */}
    <Tab.Screen
      name="ArMap"
      component={ARScreen}
      options={{
        tabBarIcon: ({ color, focused, size }) => (
```

```
          <Icon name="aperture" color={focused ? color : 'black'} size={size}
/>
        ),
      }}
    />
    <Tab.Screen
      name="ProfileStack"
      component={ProfileStack}
      options={{
        tabBarIcon: ({ color, focused, size }) => (
          <Icon name="user" color={focused ? color : 'black'} size={size} />
        ),
      }}
    />
  </Tab.Navigator>
);

export default TabsNavgation;
```

app/PersonalizedTour/src/scenes/ArScene/ArScene.js

```
import React, { useContext, useState } from 'react';
import {
 ViroAmbientLight,
 ViroARScene,
 ViroTrackingStateConstants,
 ViroSpotLight,
} from '@viro-community/react-viro';
import { Context as LocationContext } from '../../context/LocationContext';
import PlaceNodes from './PlaceNodes';
import usePlaces from '../../hooks/usePlaces';

const ArScene = () => {
 const [initialized, setInitialized] = useState(false);

 const {
   state: { initialLocation },
 } = useContext(LocationContext);

 const [places] = usePlaces();

 const onInitialized = state => {
   if (state === ViroTrackingStateConstants.TRACKING_NORMAL) {
     setInitialized(true);
   } else if (state === ViroTrackingStateConstants.TRACKING_NONE) {
     setInitialized(false);
   }
 };

 return (
   <ViroARScene onTrackingUpdated={onInitialized}>
     {initialized && (
       <>
         <PlaceNodes
           places={places}
           currentLocation={initialLocation}
           initialized={initialized}
         />
         <ViroAmbientLight color={'#aaaaaa'} />
```

```
          <ViroSpotLight
            innerAngle={5}
            outerAngle={90}
            direction={[0, -1, -0.2]}
            position={[0, 3, 1]}
            color="#ffffff"
            castsShadow={true}
          />
        </>
      )}
    </ViroARScene>
  );
};

export default ArScene;
```

app/PersonalizedTour/src/scenes/ArScene/PlaceNodes.js

```
import React from 'react';
import { Platform } from 'react-native';
import { ViroFlexView, ViroImage, ViroText } from '@viro-community/react-
viro';
import { calculateDistance, latLongToMerc } from '../../utils';
import { useNavigation } from '@react-navigation/native';

const PlaceNodes = ({ places, currentLocation, initialized }) => {
  const navigation = useNavigation();
  if (!places || !currentLocation || !initialized) {
    return null;
  }

  const transformGpsToAR = (lat, lng) => {
    const isAndroid = Platform.OS === 'android';
    const latObj = lat;
    const longObj = lng;
    const latMobile = currentLocation.coords.latitude;
    const longMobile = currentLocation.coords.longitude;

    const deviceObjPoint = latLongToMerc(latObj, longObj);
    const mobilePoint = latLongToMerc(latMobile, longMobile);
    const objDeltaY = deviceObjPoint.y - mobilePoint.y;
    const objDeltaX = deviceObjPoint.x - mobilePoint.x;

    if (isAndroid) {
      let degree = currentLocation.coords.heading;
      let angleRadian = (degree * Math.PI) / 180;
      let newObjX =
        objDeltaX * Math.cos(angleRadian) - objDeltaY * Math.sin(angleRadian);
      let newObjY =
        objDeltaX * Math.sin(angleRadian) + objDeltaY * Math.cos(angleRadian);
      return { x: newObjX, z: -newObjY };
    }

    return { x: objDeltaX, z: -objDeltaY };
  };

  const onPlaceClick = place => {
    navigation.navigate('PlaceDetailsScreen', place);
  };
```

```
 return places.map((item, index) => {
   const coords = transformGpsToAR(
     item.coordinate.latitude,
     item.coordinate.longitude,
   );

   const distance = calculateDistance(item.coordinate,
currentLocation.coords);

   const scale = Math.abs(Math.round(coords.z / 2));

   const distanceFormatted =
     distance > 1000
       ? (distance / 1000).toFixed(1) + ' km'
       : distance.toFixed(1) + ' m';

   return (
     <ViroFlexView
       key={`place-card-${index}-${item.id}`}
       style={{
         flexDirection: 'row',
         padding: 0.1,
         backgroundColor: '#ffff',
       }}
       width={2.5}
       height={1.0}
       position={[coords.x, 0, coords.z]}
       scale={[scale, scale, scale]}
       transformBehaviors="billboard"
       onClick={() => onPlaceClick(item)}
       visible={distance <= 1500}>
       <ViroImage source={{ uri: item.image }} style={{ flex: 0.4 }} />
       <ViroFlexView
         style={{ flex: 0.6, flexDirection: 'column', paddingLeft: 0.1 }}>
         <ViroText
           style={{
             fontSize: 20,
             color: '#000',
           }}
           text={item.name}
         />
         <ViroText
           style={{
             fontSize: 15,
             color: '#222222',
           }}
           text={item.description}
         />
         <ViroText
           textAlign="right"
           style={{
             fontSize: 12,
             color: '#222222',
           }}
           text={distanceFormatted}
         />
       </ViroFlexView>
     </ViroFlexView>
   );
 });
};
```

```
export default PlaceNodes;
```

app/PersonalizedTour/src/screens/ARScreen.js

```
import React from 'react';
import { StyleSheet, View } from 'react-native';
import { ViroARSceneNavigator } from '@viro-community/react-viro';
import ArScene from '../scenes/ArScene/ArScene';

const ARScreen = () => {
 return (
   <View style={styles.root}>
     <View style={styles.viroContainer}>
       <ViroARSceneNavigator
         worldAlignment="GravityAndHeading"
         initialScene={{ scene: ArScene }}
         viroAppProps={{ myProps: {} }}
       />
     </View>
   </View>
 );
};

const styles = StyleSheet.create({
 root: {
   flex: 1,
 },
 viroContainer: { ...StyleSheet.absoluteFillObject },
});

export default ARScreen;
```

app/PersonalizedTour/src/screens/AuthControl.js

```
import React from 'react';
import { View } from 'react-native';
import { auth } from '../firebase/config';

const AuthControl = ({ navigation }) => {
 React.useEffect(() => {
   auth.onAuthStateChanged(user => {
     user
       ? navigation.navigate('TabNavigation')
       : navigation.navigate('Login');
   });
 }, [navigation]);

 // eslint-disable-next-line react-native/no-inline-styles
 return <View style={{ flex: 1, backgroundColor: 'white' }} />;
};
export default AuthControl;
```

app/PersonalizedTour/src/screens/GoogleMapScreen.js

```
import React from 'react';
import { StyleSheet, View } from 'react-native';
import GooglePlacesInput from '../components/GooglePlacesInput';
import Map from '../components/Map';

const GoogleMapScreen = () => {
 return (
   <View style={styles.root}>
     <View style={styles.mapContainer}>
       <Map style={styles.map} options={{ showsMyLocationButton: true }} />
     </View>
     <GooglePlacesInput />
   </View>
 );
};

const styles = StyleSheet.create({
 root: {
   flex: 1,
 },
 mapContainer: {
   ...StyleSheet.absoluteFillObject,
   justifyContent: 'flex-end',
   alignItems: 'center',
 },
 map: {
   ...StyleSheet.absoluteFillObject,
 },
});

export default GoogleMapScreen;
```

app/PersonalizedTour/src/screens/HomeScreen.js

```
import React from 'react';
import {
 View,
 Text,
 FlatList,
 StyleSheet,
 TouchableOpacity,
 ScrollView,
} from 'react-native';
import { Icon, Image } from 'react-native-elements';
import { Avatar, TextInput } from 'react-native-paper';
import { SafeAreaView } from 'react-native-safe-area-context';
import PlaceCard from '../components/PlaceCard';
import { auth } from '../firebase/config';
import usePlaces from '../hooks/usePlaces';
import { match } from '../utils';

const HomeHeader = ({ navigation }) => {
 const user = auth.currentUser;

 return (
   <View style={styles.header}>
     <View style={styles.headerTopFrame}>
```

```
        <View style={styles.headerTopFrameLeft}>
          <Text style={styles.welcomeText}>Welcome 👋</Text>
          <Text numberOfLines={1} style={styles.userNameText}>
            {user?.displayName}
          </Text>
        </View>

        <View style={styles.headerTopFrameMiddle}>
          <Image
            style={styles.logoImage}
            resizeMode="contain"
            source={require('../assets/listed.png')}
          />
        </View>
      </View>

      <View style={styles.headerTopFrameRight}>
        <TouchableOpacity onPress={() => navigation.navigate('ProfileStack')}>
          <Avatar.Image
            source={{ uri: user?.photoURL }}
            style={styles.avatarImage}
            size={50}
          />
        </TouchableOpacity>
      </View>
    </View>
  );
};

const HomeScreen = ({ navigation }) => {
  const [places, placesByCategory] = usePlaces();
  const [searchInput, setSearchInput] = React.useState('');

  const filteredPlaces = match(searchInput, places, 'name');

  return (
    <SafeAreaView style={styles.root}>
      <HomeHeader navigation={navigation} />
      <View style={styles.searchFrame}>
        <View style={styles.searchView}>
          <TextInput
            onChangeText={setSearchInput}
            outlineColor="#efefef"
            placeholderTextColor={'#a3a3a3'}
            style={styles.searchInput}
            autoComplete="none"
            autoCapitalize="none"
            autoFocus="none"
            left={
              <TextInput.Icon name={() => <Icon name="search" size={18} />} />
            }
            mode="outlined"
            placeholder="Restaurant, Theater...."
          />
        </View>
      </View>

      <ScrollView style={{ display: searchInput.length > 1 ? 'none' : '' }}>
        <View style={styles.popularEventsFrame}>
          <View style={styles.popularEventsFrame_left}>
            <Text style={styles.popularEventsText}>Featured Places</Text>
          </View>
```

```
      </View>

      <View style={styles.populerEventsListFrame}>
        <FlatList
          onScroll={false}
          showsHorizontalScrollIndicator={false}
          contentContainerStyle={{ paddingLeft: 30 }}
          horizontal={true}
          data={places}
          renderItem={({ item }) =>
            item.featured ? <PlaceCard page="Home" data={item} /> : null
          }
        />
      </View>

      {placesByCategory &&
        Object.keys(placesByCategory).map(cat => {
          return (
            <View key={cat}>
              <View style={styles.popularEventsFrame}>
                <View style={styles.popularEventsFrame_left}>
                  <Text style={styles.popularEventsText}>{cat}</Text>
                </View>
              </View>
              <View style={styles.eventsListTicketView}>
                <FlatList
                  onScroll={false}
                  showsHorizontalScrollIndicator={false}
                  contentContainerStyle={{ paddingLeft: 30 }}
                  horizontal={true}
                  data={placesByCategory[cat]}
                  renderItem={({ item }) => (
                    <PlaceCard page="Home" data={item} />
                  )}
                />
              </View>
            </View>
          );
        })}
    </ScrollView>

    <View
      style={[
        styles.searchResultsFrame,
        {
          display: searchInput.length > 1 ? '' : 'none',
        },
      ]}>
      <FlatList
        onScroll={false}
        contentContainerStyle={styles.searchResultFlatList}
        data={filteredPlaces}
        renderItem={({ item }) => <PlaceCard page="Home" data={item} />}
      />
    </View>
  </SafeAreaView>
 );
};

const styles = StyleSheet.create({
 root: {
   flex: 1,
   backgroundColor: 'white',
```

```
},
header: {
  width: '100%',
  height: 100,
  flexDirection: 'row',
  justifyContent: 'flex-end',
},
headerTopFrame: {
  width: '80%',
  height: '100%',
  justifyContent: 'center',
  flexDirection: 'row',
},
headerTopFrameLeft: {
  width: '60%',
  marginRight: -30,
  justifyContent: 'center',
  marginLeft: 30,
},
welcomeText: {
  fontSize: 22,
  // fontFamily: 'Roboto-Medium',
  marginBottom: 5,
},
userNameText: {
  fontSize: 18,
  // fontFamily: 'Roboto-Light',
},
headerTopFrameMiddle: {
  justifyContent: 'center',
  width: '40%',
  height: '100%',
},
logoImage: {
  width: '100%',
  height: 20,
},
headerTopFrameRight: {
  width: '20%',
  height: '100%',
  justifyContent: 'center',
  alignItems: 'flex-end',
},
avatarImage: {
  marginRight: 30,
  backgroundColor: '#e0e0e0',
},
searchFrame: {
  width: '100%',
  alignItems: 'center',
},
searchView: {
  width: '85%',
},
searchInput: {
  fontSize: 14,
  backgroundColor: 'white',
},
popularEventsFrame: {
  flexDirection: 'row',
  width: '100%',
  margin: 30,
},
popularEventsFrame_left: {
```

```
    width: '60%',
    height: '100%',
  },
  popularEventsText: {
    // fontFamily: 'Roboto-Medium',
  },
  popularEventsFrame_right: {
    width: '40%',
    height: '100%',
  },
  seeMoreText: {
    color: '#4256a1',
    // fontFamily: 'Roboto-Regular',
  },
  populerEventsListFrame: {
    flexDirection: 'row',
    width: '100%',
  },
  eventsListTicketView: {
    flexDirection: 'row',
    width: '100%',
    margin: 0,
    paddingBottom: 50,
  },
  searchResultsFrame: {
    width: '100%',
    alignItems: 'center',
  },
  searchResultFlatList: {
    paddingTop: 40,
    paddingBottom: 180,
  },
});

export default HomeScreen;
```

app/PersonalizedTour/src/screens/LoginScreen.js

```
import React from 'react';
import {
  View,
  SafeAreaView,
  Text,
  Image,
  StyleSheet,
  TouchableOpacity,
} from 'react-native';
import { TextInput, Button } from 'react-native-paper';
import { login } from '../firebase/auth';
import Toast from 'react-native-toast-message';

const LoginScreen = ({ navigation }) => {
  const [securePassword, setSecurePassword] = React.useState(true);
  const [isLoading, setIsLoading] = React.useState(false);
  const [email, setEmail] = React.useState('');
  const [password, setPassword] = React.useState('');

  const handleLogin = async () => {
    setIsLoading(true);
    try {
```

```
      await login(email, password);
    } catch (error) {
      Toast.show({
        type: 'error',
        text1: error.message,
        text2: error.code,
      });
    }
    setIsLoading(false);
  };

  return (
    <SafeAreaView style={styles.mainFrame}>
      <Toast />
      <View style={styles.mainFrameCentered}>
        <View style={styles.logo}>
          <Image
            style={{ width: '60%', height: 50 }}
            resizeMode="contain"
            source={require('../assets/listed.png')}
          />
        </View>

        <TextInput
          autoCapitalize="none"
          placeholderTextColor={'#2f3030'}
          mode="outlined"
          label="Email"
          style={styles.input}
          onChangeText={setEmail}
          outlineColor="#efefef"
        />
        <TextInput
          secureTextEntry={securePassword}
          placeholderTextColor={'#2f3030'}
          right={
            <TextInput.Icon
              onPressIn={() => setSecurePassword(false)}
              onPressOut={() => setSecurePassword(true)}
              name="eye"
            />
          }
          autoCapitalize="none"
          outlineColor="#efefef"
          mode="outlined"
          label="Password"
          style={styles.input}
          onChangeText={setPassword}
        />

        <Button onPress={handleLogin} loading={isLoading} mode="outlined">
          Log In
        </Button>

        <View style={styles.bottomTextFrame}>
          <TouchableOpacity onPress={() => navigation.navigate('SignUp')}>
            <Text style={styles.bottomText}> Sign Up </Text>
          </TouchableOpacity>
        </View>
      </View>
    </SafeAreaView>
  );
};
```

```
const styles = StyleSheet.create({
 mainFrame: {
   flex: 1,
   backgroundColor: 'white',
   justifyContent: 'center',
   alignItems: 'center',
 },
 mainFrameCentered: {
   width: '80%',
 },
 logo: {
   alignItems: 'center',
   marginBottom: 30,
 },
 input: {
   marginBottom: 20,
   backgroundColor: 'white',
 },
 bottomTextFrame: {
   marginTop: 20,
   width: '100%',
   justifyContent: 'center',
   alignItems: 'center',
 },
 bottomText: {
   // fontFamily: 'Roboto-Light',
   letterSpacing: 0.3,
 },
});

export default LoginScreen;
```

app/PersonalizedTour/src/screens/ModalAddPlace.js

```
import React, { useContext } from 'react';
import { push, ref } from 'firebase/database';
import { database } from '../firebase/config';
import { Button, TextInput } from 'react-native-paper';
import {
 SafeAreaView,
 Text,
 View,
 TouchableNativeFeedback,
 Dimensions,
 StyleSheet,
 ScrollView,
 LogBox,
} from 'react-native';
import { Avatar } from 'react-native-paper';
import * as ImagePicker from 'react-native-image-picker';
import { uploadPlacePhoto } from '../firebase/storage';
import { Image } from 'react-native-elements';
import MapView, { Marker, PROVIDER_GOOGLE } from 'react-native-maps';
import { Context as LocationContext } from '../context/LocationContext';
import { colors } from '../constants';
import DropDownPicker from 'react-native-dropdown-picker';

async function getFileFromUrl(url) {
 const response = await fetch(url);
```

```
 return response.blob();
}

const ModalAddPlace = ({ navigation }) => {
 const [open, setOpen] = React.useState(false);
 const [category, setCategory] = React.useState(null);
 const [items, setItems] = React.useState([
   { label: 'Museums', value: 'Museums' },
   { label: 'Sports Center', value: 'Sports Center' },
   { label: 'Stadium', value: 'Stadium' },
   { label: 'Restaurant', value: 'Restaurant' },
   { label: 'Churches', value: 'Churches' },
   { label: 'Hospital', value: 'Hospital' },
   { label: 'Historical landmark', value: 'Historical landmark' },
   { label: 'Distinctive buildings', value: 'Distinctive buildings' },
   { label: 'Tourist attractions', value: 'Tourist attractions' },
 ]);
 const [image, setImage] = React.useState('');
 const [name, setName] = React.useState('');
 const [description, setDescription] = React.useState('');
 const [website, setWebsite] = React.useState('');
 const [phone, setPhone] = React.useState('');
 const [email, setEmail] = React.useState('');
 const [address, setAddress] = React.useState('');
 const [newPoi, setNewPoi] = React.useState();

 const {
   state: { currentLocation },
 } = useContext(LocationContext);

 const lat = currentLocation?.coords?.latitude;
 const long = currentLocation?.coords?.longitude;

 const onSetNewPoiAddress = e => {
   setNewPoi({ coordinate: e.nativeEvent.coordinate });
 };

 const handleSave = async () => {
   let url = '';
   if (image && image.base64) {
     url = await uploadPlacePhoto(image.blob, image.fileName);
   }

   const extra_data = {
     featured: false,
     name,
     description,
     phone,
     email,
     website,
     image: url,
     coordinate: newPoi
       ? { ...newPoi.coordinate }
       : { latitude: lat, longitude: long },
     address,
     category: category?.value || 'Museums',
   };
   const reference = ref(database, 'places/');
   push(reference, extra_data).then(() => {
     navigation.navigate('Home');
   });
 };
```

```
const handleChoosePhoto = React.useCallback(async () => {
  const options = {
    selectionLimit: 1,
    mediaType: 'photo',
    includeBase64: true,
  };
  const result = await ImagePicker.launchImageLibrary(options);
  const fileName = result?.assets && result.assets[0].fileName;
  const uri = result?.assets && result.assets[0].uri;
  const base64 = result?.assets && result.assets[0].base64;
  const blob = await getFileFromUrl(uri);
  console.log({ blob });
  setImage({ uri, base64, fileName, blob });
}, []);

React.useEffect(() => {
  LogBox.ignoreLogs(['VirtualizedLists should never be nested']);
}, []);

return (
  <View style={styles.mainFrame}>
    <ScrollView>
      <SafeAreaView style={styles.pageContentFrame}>
        <View style={styles.headerUserInfo}>
          <View style={styles.headerUserInfo_avatarFrame}>
            <Image
              source={{ uri: image.uri }}
              style={{ width: 100, height: 100 }}
            />
            <TouchableNativeFeedback onPress={handleChoosePhoto}>
              <Avatar.Icon
                size={35}
                style={styles.headerUserInfo_profileChangeAvatar}
                icon="camera"
              />
            </TouchableNativeFeedback>
          </View>
        </View>

        <View style={[styles.editTextFrame, { marginTop: 100 }]}>
          <View style={styles.frameCentered}>
            <Text style={styles.inputText}>Name</Text>
          </View>
        </View>

        <View style={styles.editInputFrame}>
          <View style={styles.frameCentered}>
            <TextInput
              placeholder="Acropolis Museum"
              outlineColor="#efefef"
              style={styles.input}
              mode="outlined"
              onChangeText={setName}
              value={name}
            />
          </View>
        </View>

        <View style={[styles.editTextFrame, { marginTop: 20 }]}>
          <View style={styles.frameCentered}>
            <Text style={styles.inputText}>Description</Text>
```

```
          </View>
        </View>

        <View style={styles.editInputFrame}>
          <View style={styles.frameCentered}>
            <TextInput
              placeholder="The Acropolis… "
              outlineColor="#efefef"
              multiline
              style={styles.inputMultiline}
              mode="outlined"
              onChangeText={setDescription}
              value={description}
            />
          </View>
        </View>

        <View style={[styles.editTextFrame, { marginTop: 20 }]}>
          <View style={styles.frameCentered}>
            <Text style={styles.inputText}>Email</Text>
          </View>
        </View>

        <View style={styles.editInputFrame}>
          <View style={styles.frameCentered}>
            <TextInput
              placeholder="info@theacropolismuseum.gr"
              outlineColor="#efefef"
              style={styles.input}
              mode="outlined"
              onChangeText={setEmail}
              value={email}
            />
          </View>
        </View>

        <View style={[styles.editTextFrame, { marginTop: 20 }]}>
          <View style={styles.frameCentered}>
            <Text style={styles.inputText}>Phone</Text>
          </View>
        </View>

        <View style={styles.editInputFrame}>
          <View style={styles.frameCentered}>
            <TextInput
              placeholder="210 0900 0900"
              outlineColor="#efefef"
              style={styles.input}
              mode="outlined"
              onChangeText={setPhone}
              value={phone}
            />
          </View>
        </View>

        <View style={[styles.editTextFrame, { marginTop: 20 }]}>
          <View style={styles.frameCentered}>
            <Text style={styles.inputText}>Website</Text>
          </View>
        </View>

        <View style={styles.editInputFrame}>
```

```
            <View style={styles.frameCentered}>
              <TextInput
                placeholder="theacropolismuseum.gr"
                outlineColor="#efefef"
                style={styles.input}
                mode="outlined"
                onChangeText={setWebsite}
                value={website}
              />
            </View>
          </View>

          <View
            style={[styles.editTextFrame, { marginTop: 20, marginBottom: 5 }]}>
            <View style={styles.frameCentered}>
              <Text style={styles.inputText}>Category</Text>
            </View>
          </View>

          <View
            style={[styles.editInputFrame, { marginTop: 20, marginBottom: 5
}]}>
            <View style={styles.frameCentered}>
              <DropDownPicker
                placeholder="Select a category"
                open={open}
                value={category}
                items={items}
                setOpen={setOpen}
                setValue={setCategory}
                setItems={setItems}
                style={{ borderColor: '#efefef' }}
                dropDownContainerStyle={{ borderColor: '#efefef' }}
                searchTextInputStyle={{ borderColor: '#efefef' }}
              />
            </View>
          </View>

          <View style={[styles.editTextFrame, { marginTop: open ? 220 : 20 }]}>
            <View style={styles.frameCentered}>
              <Text style={styles.headerUserInfo_nameText}>Location</Text>
            </View>
          </View>

          <View style={[styles.editTextFrame, { marginTop: 20 }]}>
            <View style={styles.frameCentered}>
              <Text style={styles.inputText}>Address</Text>
            </View>
          </View>

          <View style={styles.editInputFrame}>
            <View style={styles.frameCentered}>
              <TextInput
                placeholder="Dionysiou Areopagitou 15, Athina 117 42"
                outlineColor="#efefef"
                style={styles.input}
                mode="outlined"
                onChangeText={setAddress}
                value={address}
              />
            </View>
          </View>
```

```
        <View style={styles.mapContainer}>
          <MapView
            provider={PROVIDER_GOOGLE}
            style={styles.map}
            initialRegion={{
              latitude: lat,
              longitude: long,
              latitudeDelta: 0.02,
              longitudeDelta: 0.02,
            }}
            showsUserLocation
            showsMyLocationButton
            onPress={e => onSetNewPoiAddress(e)}>
            {newPoi && (
              <Marker
                coordinate={newPoi.coordinate}
                onDrag={e => onSetNewPoiAddress(e)}
                draggable
              />
            )}
          </MapView>
        </View>
      </SafeAreaView>
    </ScrollView>

    <View style={styles.bottomButtonFrame}>
      <Button
        onPress={handleSave}
        style={styles.bottomButton}
        mode="contained">
        Save
      </Button>
    </View>
  </View>
 );
};

const styles = StyleSheet.create({
 mainFrame: {
   flex: 1,
   backgroundColor: 'white',
 },
 headerBar: {
   justifyContent: 'center',
   flexDirection: 'row',
   alignItems: 'center',
   width: '100%',
   height: Dimensions.get('window').height / 8.5,
   borderBottomWidth: 1,
   borderBottomColor: '#eeee',
 },
 headerBarLeft: {
   width: '15%',
   height: '100%',
   justifyContent: 'flex-end',
   alignItems: 'flex-end',
 },
 headerBarMiddle: {
   width: '70%',
   height: '100%',
   justifyContent: 'flex-end',
   alignItems: 'center',
```

```
},
headerBarText: {
  paddingBottom: 17,
  color: 'black',
  fontSize: 17,
  // fontFamily: 'Roboto-Regular',
},
headerBarRight: {
  width: '15%',
  height: '100%',
  justifyContent: 'flex-end',
  alignItems: 'flex-start',
},
pageContentFrame: {
  backgroundColor: 'white',
  marginTop: 30,
},
headerUserInfo: {
  width: '100%',
  height: 100,
  justifyContent: 'center',
  alignItems: 'center',
},
headerUserInfo_avatarFrame: {
  justifyContent: 'center',
  marginTop: 100,
  width: '100%',
  height: '100%',
  alignItems: 'center',
},
headerUserInfo_avatar: {
  backgroundColor: '#e0e0e0',
},
headerUserInfo_profileChangeAvatar: {
  top: 80,
  position: 'absolute',
},
headerUserInfo_texts: {
  alignItems: 'center',
  justifyContent: 'center',
  width: '100%',
  height: '100%',
},
headerUserInfo_nameText: {
  fontSize: 23,
  // fontFamily: 'Roboto-Regular',
  letterSpacing: 0,
  marginBottom: 5,
},
headerUserInfo_emailText: {
  color: '#727987',
  fontSize: 14,
  // fontFamily: 'Roboto-Regular',
  letterSpacing: 0,
},
editTextFrame: {
  width: '100%',
  alignItems: 'center',
},
frameCentered: {
  width: '85%',
},
inputText: {
  marginLeft: 3,
```

```
    fontSize: 14.5,
    color: '#2f3030',
  },
  editInputFrame: {
    width: '100%',
    justifyContent: 'center',
    alignItems: 'center',
  },
  input: {
    backgroundColor: 'white',
    height: 50,
  },
  inputMultiline: {
    backgroundColor: 'white',
    height: 120,
  },
  editRowInputFrame: {
    width: '92%',
    flexDirection: 'row',
    justifyContent: 'center',
  },
  editRowInputFrameCentered: {
    width: '50%',
  },
  editRowInputView: {
    marginTop: 20,
    width: '100%',
    alignItems: 'center',
  },
  editRowWidth: {
    width: '50%',
  },
  genderInputFrame: {
    marginTop: 10,
    paddingBottom: 100,
    width: '100%',
    alignItems: 'center',
  },
  genderInputView: {
    borderRadius: 5,
    width: '85%',
    borderWidth: 1,
    borderColor: '#efefef',
  },
  genderText: {
    color: '#2f3030',
  },

  bottomButtonFrame: {
    zIndex: 2,
    bottom: 20,
    position: 'absolute',
    width: '100%',
    height: 40,
    justifyContent: 'center',
    alignItems: 'center',
  },
  bottomButton: {
    width: '50%',
    height: '100%',
    justifyContent: 'center',
    backgroundColor: colors.profileEditBackground,
  },
  mapContainer: {
```

```
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
    width: Dimensions.get('window').width,
    height: 400,
    padding: 25,
    marginBottom: 25,
    overflow: 'hidden',
  },
  map: {
    width: '100%',
    height: '100%',
  },
});

export default ModalAddPlace;
```

app/PersonalizedTour/src/screens/ModalEditProfile.js

```
import React, { useEffect, useState } from 'react';
import { onValue, ref, set } from 'firebase/database';
import { database, auth } from '../firebase/config';
import { Button, TextInput } from 'react-native-paper';
import {
  SafeAreaView,
  Text,
  View,
  TouchableNativeFeedback,
  Dimensions,
  StyleSheet,
  ScrollView,
} from 'react-native';
import { Avatar, RadioButton } from 'react-native-paper';
import { updateProfile } from 'firebase/auth';
import { Toast } from 'react-native-toast-message/lib/src/Toast';
import * as ImagePicker from 'react-native-image-picker';
import { uploadProfilePhoto } from '../firebase/storage';
import { colors } from '../constants';

const ModalEditProfile = () => {
  const user = auth.currentUser;
  const [profilePhoto, setProfilePhoto] = useState(user.photoURL);
  const [displayName, setDisplayName] = useState(user.displayName);
  const [email, setEmail] = useState(user.email);
  const [country, setCountry] = useState('');
  const [age, setAge] = useState('');
  const [gender, setGender] = useState('');

  useEffect(() => {
    const reference = ref(database, 'users/' + user.uid + '/info');
    onValue(reference, snapshot => {
      const data = snapshot.val();
      setEmail(data.email);
      setCountry(data.country);
      setAge(data.age);
      setGender(data.gender);
    });
  }, [user.uid]);
```

```
const save = () => {
  var extra_data = {
    age: age,
    country: country,
    gender: gender,
  };

  auth.onAuthStateChanged(user => {
    updateProfile(user, {
      displayName: displayName,
      email: email,
    })
      .then(() => {
        const reference = ref(database, 'users/' + user.uid + '/info');
        set(reference, extra_data);

        Toast.show({
          type: 'success',
          text1: 'Profile updated!',
          text2: '',
        });
      })
      .catch(error => {
        Toast.show({
          type: 'error',
          text1: error.message,
          text2: '',
        });
      });
  });
};

const ProfilePhotoChangedButton = React.useCallback(async () => {
  const options = {
    selectionLimit: 1,
    mediaType: 'photo',
    includeBase64: true,
  };
  const result = await ImagePicker.launchImageLibrary(options);
  const uri = result?.assets && result.assets[0].uri;
  const base64 = result?.assets && result.assets[0].base64;
  setProfilePhoto(uri);
  await uploadProfilePhoto(base64, user, console.log);
}, [user]);

return (
  <View style={styles.mainFrame}>
    <ScrollView>
      <SafeAreaView style={styles.pageContentFrame}>
        <View style={styles.headerUserInfo}>
          <View style={styles.headerUserInfo_avatarFrame}>
            <Avatar.Image
              style={styles.headerUserInfo_avatar}
              source={{ uri: profilePhoto }}
              size={100}
            />
            <TouchableNativeFeedback
              onPress={() => ProfilePhotoChangedButton()}>
              <Avatar.Icon
                size={35}
                style={styles.headerUserInfo_profileChangeAvatar}
                icon="camera"
              />
```

```
            </TouchableNativeFeedback>
          </View>

          <View style={styles.headerUserInfo_texts}>
            <Text style={styles.headerUserInfo_nameText}>{displayName}</Text>
            <Text style={styles.headerUserInfo_emailText}>{email}</Text>
          </View>
        </View>

        <View style={[styles.editTextFrame, { marginTop: 100 }]}>
          <View style={styles.frameCentered}>
            <Text style={styles.inputText}>Display Name</Text>
          </View>
        </View>

        <View style={styles.editInputFrame}>
          <View style={styles.frameCentered}>
            <TextInput
              outlineColor="#efefef"
              placeholderTextColor={'#2f3030'}
              placeholder={displayName}
              style={styles.input}
              mode="outlined"
              onChangeText={setDisplayName}
            />
          </View>
        </View>

        <View style={[styles.editTextFrame, { marginTop: 20 }]}>
          <View style={styles.frameCentered}>
            <Text style={styles.inputText}>Email</Text>
          </View>
        </View>

        <View style={styles.editInputFrame}>
          <View style={styles.frameCentered}>
            <TextInput
              outlineColor="#efefef"
              placeholderTextColor={'#2f3030'}
              placeholder={email}
              style={styles.input}
              mode="outlined"
              onChangeText={setEmail}
              value={email}
            />
          </View>
        </View>

        <View style={styles.editTextFrame}>
          <View style={styles.editRowInputFrame}>
            <View style={styles.editRowInputFrameCentered}>
              <View style={styles.editRowInputView}>
                <View style={styles.frameCentered}>
                  <Text style={styles.inputText}>City</Text>
                </View>
              </View>

              <View style={styles.editInputFrame}>
                <View style={styles.frameCentered}>
                  <TextInput
                    outlineColor="#efefef"
                    placeholderTextColor={'#2f3030'}
```

```
                placeholder={country}
                style={styles.input}
                mode="outlined"
                onChangeText={setCountry}
              />
            </View>
          </View>
        </View>

        <View style={styles.editRowWidth}>
          <View style={styles.editRowInputView}>
            <View style={styles.frameCentered}>
              <Text style={styles.inputText}>Age</Text>
            </View>
          </View>

          <View style={styles.editInputFrame}>
            <View style={styles.frameCentered}>
              <TextInput
                outlineColor="#efefef"
                keyboardType="numeric"
                maxLength={2}
                placeholderTextColor={'#2f3030'}
                placeholder={age}
                onChangeText={setAge}
                style={styles.input}
                mode="outlined"
              />
            </View>
          </View>
        </View>
      </View>
    </View>

    <View style={[styles.editTextFrame, { marginTop: 20 }]}>
      <View style={styles.frameCentered}>
        <Text style={styles.inputText}>Gender</Text>
      </View>
    </View>

    <View style={styles.genderInputFrame}>
      <View style={styles.genderInputView}>
        <RadioButton.Group
          onValueChange={gender => setGender(gender)}
          value={gender}>
          <RadioButton.Item
            labelStyle={styles.genderText}
            label="Male"
            value="Male"
          />
          <RadioButton.Item
            labelStyle={styles.genderText}
            color="pink"
            label="Female"
            value="Female"
          />
        </RadioButton.Group>
      </View>
    </View>
  </SafeAreaView>
</ScrollView>
```

```
      <View style={styles.bottomButtonFrame}>
        <Button onPress={save} style={styles.bottomButton} mode="contained">
          Save
        </Button>
      </View>

      <Toast />
    </View>
  );
};

const styles = StyleSheet.create({
  mainFrame: {
    flex: 1,
    backgroundColor: 'white',
  },
  headerBar: {
    justifyContent: 'center',
    flexDirection: 'row',
    alignItems: 'center',
    width: '100%',
    height: Dimensions.get('window').height / 8.5,
    borderBottomWidth: 1,
    borderBottomColor: '#eeee',
  },
  headerBarLeft: {
    width: '15%',
    height: '100%',
    justifyContent: 'flex-end',
    alignItems: 'flex-end',
  },
  headerBarMiddle: {
    width: '70%',
    height: '100%',
    justifyContent: 'flex-end',
    alignItems: 'center',
  },
  headerBarText: {
    paddingBottom: 17,
    color: 'black',
    fontSize: 17,
    // fontFamily: 'Roboto-Regular',
  },
  headerBarRight: {
    width: '15%',
    height: '100%',
    justifyContent: 'flex-end',
    alignItems: 'flex-start',
  },
  pageContentFrame: {
    backgroundColor: 'white',
    marginTop: 30,
  },
  headerUserInfo: {
    width: '100%',
    height: 100,
    justifyContent: 'center',
    alignItems: 'center',
  },
  headerUserInfo_avatarFrame: {
    justifyContent: 'center',
    marginTop: 100,
    width: '100%',
```

```
    height: '100%',
    alignItems: 'center',
  },
  headerUserInfo_avatar: {
    backgroundColor: '#e0e0e0',
  },
  headerUserInfo_profileChangeAvatar: {
    top: 80,
    position: 'absolute',
  },
  headerUserInfo_texts: {
    alignItems: 'center',
    justifyContent: 'center',
    width: '100%',
    height: '100%',
  },
  headerUserInfo_nameText: {
    fontSize: 23,
    // fontFamily: 'Roboto-Regular',
    letterSpacing: 0,
    marginBottom: 5,
  },
  headerUserInfo_emailText: {
    color: '#727987',
    fontSize: 14,
    // fontFamily: 'Roboto-Regular',
    letterSpacing: 0,
  },
  editTextFrame: {
    width: '100%',
    alignItems: 'center',
  },
  frameCentered: {
    width: '85%',
  },
  inputText: {
    marginLeft: 3,
    fontSize: 14.5,
    // fontFamily: 'Roboto-Regular',
    color: '#2f3030',
  },
  editInputFrame: {
    width: '100%',
    justifyContent: 'center',
    alignItems: 'center',
  },
  input: {
    backgroundColor: 'white',
    height: 50,
  },
  editRowInputFrame: {
    width: '92%',
    flexDirection: 'row',
    justifyContent: 'center',
  },
  editRowInputFrameCentered: {
    width: '50%',
  },
  editRowInputView: {
    marginTop: 20,
    width: '100%',
    alignItems: 'center',
  },
  editRowWidth: {
```

```
    width: '50%',
  },
  genderInputFrame: {
    marginTop: 10,
    paddingBottom: 100,
    width: '100%',
    alignItems: 'center',
  },
  genderInputView: {
    borderRadius: 5,
    width: '85%',
    borderWidth: 1,
    borderColor: '#efefef',
  },
  genderText: {
    color: '#2f3030',
  },
  bottomButtonFrame: {
    zIndex: 2,
    bottom: 20,
    position: 'absolute',
    width: '100%',
    height: 40,
    justifyContent: 'center',
    alignItems: 'center',
  },
  bottomButton: {
    width: '50%',
    height: '100%',
    justifyContent: 'center',
    backgroundColor: colors.profileEditBackground,
  },
});

export default ModalEditProfile;
```

app/PersonalizedTour/src/screens/PlaceDetailsScreen.js

```
import React from 'react';
import {
  Dimensions,
  Image,
  ScrollView,
  StatusBar,
  StyleSheet,
  Text,
  View,
} from 'react-native';
import { Button } from 'react-native-paper';
import { Context as LocationContext } from '../context/LocationContext';
import Icon from 'react-native-vector-icons/Feather';
import PlaceDetailMap from '../components/PlaceDetailMap';
import { colors } from '../constants';
import getDirections from 'react-native-google-maps-directions';

const PlaceDetailsScreen = ({ route, navigation }) => {
  const {
    state: { currentLocation },
  } = React.useContext(LocationContext);
```

```
const place = route.params;
const [metaArr, setMetaArr] = React.useState([]);
React.useLayoutEffect(() => {
  navigation.setOptions({
    title: place.name === '' ? 'No title' : place.name,
  });
}, [navigation, place]);

React.useEffect(() => {
  const tempMetaArr = [];
  ['email', 'phone', 'website'].forEach(m => {
    if (place[m]) {
      tempMetaArr.push({ key: m, value: place[m] });
    }
  });
  setMetaArr(tempMetaArr);
}, [place]);

const handleGetDirections = () => {
  const data = {
    source: {
      latitude: currentLocation.coords.latitude,
      longitude: currentLocation.coords.longitude,
    },
    destination: place.coordinate,
    params: [
      {
        key: 'travelmode',
        value: 'walking',
      },
      {
        key: 'dir_action',
        value: 'navigate',
      },
    ],
  };

  getDirections(data);
};

if (!place) {
  return null;
}

return (
  <View style={styles.mainFrame}>
    <StatusBar style="light" />

    <View style={styles.placeCoverFrame}>
      <Image
        style={styles.placeCoverImage}
        resizeMode="cover"
        source={{ uri: place.image }}
      />
    </View>

    <View style={styles.placeCoverHeight} />

    <View style={styles.placeInfoFrame}>
      <View style={styles.placeInfoView}>
        <Text style={styles.placeTitleText}> {place.name} </Text>
```

```
          <Text style={styles.placeLocationNameText}>
            <Icon name="map-pin" size={20} /> {place.address}
          </Text>

          <ScrollView style={styles.scroolViewFrame}>
            <View style={styles.placeDetailsInfoFrame}>
              <View style={styles.placeDetailsInfoView}>
                <Text style={styles.placeDescriptionText}>
                  {place.description}
                </Text>
                {metaArr
                  ? metaArr.map(meta => (
                      <Text key={meta.key} style={styles.placeMetaText}>
                        - {`${meta.key}: ${meta.value}`}
                      </Text>
                    ))
                  : null}
              </View>

              <View>
                <PlaceDetailMap
                  destination={place}
                  style={styles.mapView}
                  options={{ showsMyLocationButton: false }}
                />
              </View>
            </View>
          </ScrollView>
        </View>
      </View>

      <View style={styles.bottomButtonFrame}>
        <Button
          onPress={handleGetDirections}
          style={styles.bottomButton}
          mode="contained">
          Get Directions
        </Button>
      </View>
    </View>
  );
};

const styles = StyleSheet.create({
  sectionContainer: {
    marginTop: 32,
    paddingHorizontal: 24,
  },
  sectionTitle: {
    fontSize: 24,
    fontWeight: '600',
  },
  sectionDescription: {
    marginTop: 8,
    fontSize: 18,
    fontWeight: '400',
  },
  highlight: {
    fontWeight: '700',
  },

  mapContainer: {
```

```
    position: 'absolute',
    height: '40%',
    bottom: 0,
    width: '100%',
    justifyContent: 'flex-end',
    alignItems: 'center',
    overflow: 'hidden',
  },
  map: {
    ...StyleSheet.absoluteFillObject,
  },
  mainFrame: {
    flex: 1,
  },

  headerTopFrame: {
    position: 'absolute',
    zIndex: 2,
    justifyContent: 'center',
    flexDirection: 'row',
    alignItems: 'center',
    width: '100%',
    height: Dimensions.get('window').height / 8.5,
  },
  headerLeftFrame: {
    width: '15%',
    height: '100%',
    justifyContent: 'flex-end',
    alignItems: 'flex-end',
  },
  headerLeftIcon: {
    paddingBottom: 15,
  },
  headerMiddleFrame: {
    width: '70%',
    height: '100%',
    justifyContent: 'flex-end',
    alignItems: 'center',
  },
  headerRightFrame: {
    width: '15%',
    height: '100%',
    justifyContent: 'flex-end',
    alignItems: 'flex-start',
  },

  placeCoverFrame: {
    width: '100%',
    height: 330,
    position: 'absolute',
  },
  placeCoverImage: {
    width: '100%',
    height: '100%',
  },
  placeCoverHeight: {
    flex: 0.7,
  },

  placeInfoFrame: {
    shadowColor: 'black',
    shadowOffset: { width: 0, height: 0 },
    shadowRadius: 50,
```

```
    shadowOpacity: 1,
    marginTop: -40,
    flex: 1,
    backgroundColor: 'white',
    // borderTopEndRadius: 40,
    // borderTopLeftRadius: 40,
  },
  placeInfoView: {
    width: '100%',
    marginTop: 20,
    alignItems: 'center',
  },
  placeTitleText: {
    color: 'black',
    fontSize: 30,
  },

  placeLocationNameText: {
    fontSize: 15,
    marginTop: 10,
    marginBottom: 10,
  },
  scroolViewFrame: {
    width: '100%',
  },
  placeDetailsInfoFrame: {
    paddingBottom: 210,
    width: '100%',
    justifyContent: 'center',
    alignItems: 'center',
  },
  placeDetailsInfoView: {
    width: '85%',
    marginTop: 20,
    marginBottom: 20,
  },
  placeDescriptionText: {
    marginBottom: 10,
    fontSize: 14,
  },
  placeMetaText: {
    marginBottom: 5,
    fontSize: 13,
  },

  mapView: {
    width: Dimensions.get('window').width,
    height: 250,
    // borderRadius: 30,
  },

  bottomButtonFrame: {
    zIndex: 2,
    bottom: 20,
    position: 'absolute',
    width: '100%',
    height: 40,
    justifyContent: 'center',
    alignItems: 'center',
  },
  bottomButton: {
    width: '50%',
    height: '100%',
```

```
    justifyContent: 'center',
    backgroundColor: colors.profileEditBackground,
  },
});

export default PlaceDetailsScreen;
```

app/PersonalizedTour/src/screens/ProfileScreen.js

```
import React from 'react';
import { Alert } from 'react-native';
import {
 SafeAreaView,
 Text,
 View,
 Dimensions,
 TouchableOpacity,
 StyleSheet,
} from 'react-native';
import { ScrollView } from 'react-native-gesture-handler';
import { Avatar } from 'react-native-paper';
import { openSettings } from 'react-native-permissions';
import { auth } from '../firebase/config';
import TouchLineItem from '../components/TouchLineItem';
import { colors, fonts } from '../constants';

const logout = async () => {
 await auth.signOut();
};

const ProfileScreen = ({ navigation }) => {
 const user = auth.currentUser;

 const alertSignOut = () => {
   Alert.alert(
     'Sign Out',
     'Are you sure that you want to sign out?',
     [{ text: 'No' }, { text: 'Yes', onPress: () => logout() }],
     { cancelable: false },
   );
 };

 return (
   <View style={styles.mainFrame}>
     <SafeAreaView style={styles.profileContentFrame}>
       <View style={styles.profileContentFrame_ProfileFrame}>
         <View style={styles.profileContentFrame_avatarView}>
           <Avatar.Image
             source={{ uri: user?.photoURL }}
             style={styles.avatarImage}
             size={100}
           />
         </View>

         <View style={styles.profileContentFrame_userInfos}>
           <Text style={styles.userNameText}>{user.displayName}</Text>
           <Text style={styles.userEmailText}>{user.email}</Text>
         </View>
       </View>
```

```
        <View style={styles.manageProfileContainer}>
          <TouchableOpacity
            activeOpacity={0.7}
            onPress={() => navigation.navigate('ModalEditProfile')}
            style={styles.containerEditProfile}>
            <Text style={styles.editProfileText}>Edit Profile</Text>
          </TouchableOpacity>
        </View>

        <ScrollView>
          <TouchLineItem
            onPress={() => navigation.navigate('ModalAddPlace')}
            text="Add new place"
          />

          <TouchLineItem
            onPress={() => openSettings()}
            text="App permissions"
          />

          <TouchLineItem onPress={() => null} text="Legal" />
          <TouchLineItem onPress={() => null} text="Help" />
          <TouchLineItem onPress={alertSignOut} text="Log Out" />

          <Text style={styles.versionText}>{`Version: 0.0.1`}</Text>
        </ScrollView>
      </SafeAreaView>
    </View>
  );
};

const styles = StyleSheet.create({
  mainFrame: {
    flex: 1,
    backgroundColor: 'white',
  },
  headerFrame: {
    justifyContent: 'center',
    flexDirection: 'row',
    alignItems: 'center',
    width: '100%',
    height: Dimensions.get('window').height / 8.5,
    borderBottomWidth: 1,
    borderBottomColor: '#eeee',
  },
  headerFrame_left: {
    width: '15%',
    height: '100%',
    justifyContent: 'flex-end',
    alignItems: 'flex-end',
  },
  headerFrame_middle: {
    width: '70%',
    height: '100%',
    justifyContent: 'flex-end',
    alignItems: 'center',
  },
  headerText: {
    paddingBottom: 17,
    color: 'black',
    fontSize: 17,
```

```
    // fontFamily: 'Roboto-Regular',
  },
  headerFrame_right: {
    width: '15%',
    height: '100%',
    justifyContent: 'flex-end',
    alignItems: 'flex-start',
  },
  profileContentFrame: {
    marginTop: 30,
  },
  profileContentFrame_ProfileFrame: {
    width: '100%',
    height: 100,
    flexDirection: 'row',
  },
  profileContentFrame_avatarView: {
    justifyContent: 'center',
    width: '40%',
    height: '100%',
    alignItems: 'center',
  },
  avatarImage: {
    backgroundColor: '#e0e0e0',
  },
  profileContentFrame_userInfos: {
    justifyContent: 'center',
    width: '60%',
    height: '100%',
  },
  userNameText: {
    fontSize: 23,
    // fontFamily: 'Roboto-Regular',
    letterSpacing: 0,
    marginBottom: 5,
  },
  userEmailText: {
    color: '#727987',
    fontSize: 14,
    // fontFamily: 'Roboto-Regular',
    letterSpacing: 0,
  },
  manageProfileContainer: {
    marginTop: 10,
    justifyContent: 'center',
    alignItems: 'center',
  },
  containerEditProfile: {
    alignItems: 'center',
    backgroundColor: colors.profileEditBackground,
    borderRadius: 4,
    flexDirection: 'row',
    justifyContent: 'center',
    marginBottom: 24,
  },
  editProfileText: {
    color: colors.white,
    fontFamily: fonts.medium,
    paddingHorizontal: 16,
    paddingVertical: 8,
    textTransform: 'uppercase',
  },
  menuTextFrame: {
    marginLeft: 35,
```

```
    marginTop: 40,
  },
  menuText: {
    color: '#727987',
    marginBottom: 20,
    // fontFamily: 'Roboto-Light',
  },
  menuButtonFrame: {
    width: '100%',
    flexDirection: 'row',
    alignItems: 'center',
  },
  menuButtonTextFrame: {
    width: '85%',
    flexDirection: 'row',
    alignItems: 'center',
  },
  menuButtonView: {
    justifyContent: 'center',
    alignItems: 'center',
    width: 53,
    height: 53,

    // borderRadius: '50%',
  },
  menuButtonText: {
    marginLeft: 15,
    fontSize: 17,
  },
  menuButtonArrowFrame: {
    width: '15%',
  },
  ticketIconImage: {
    tintColor: 'white',
    width: '75%',
    height: '75%',
  },
  menuButtonTouchFrame: {
    marginBottom: 20,
    borderBottomColor: colors.inactiveGrey,
    borderBottomWidth: 1,
    flexDirection: 'row',
    justifyContent: 'space-between',
    paddingRight: 16,
    paddingVertical: 20,
  },
  logoutText: {
    color: '#ff6e3f',
    marginBottom: 20,
    // fontFamily: 'Roboto-Medium',
    fontSize: 18,
  },
  versionText: {
    color: colors.inactiveGrey,
    fontFamily: fonts.regular,
    fontSize: 18,
    marginLeft: 16,
    paddingVertical: 16,
  },
});

export default ProfileScreen;
```

app/PersonalizedTour/src/screens/SignUpScreen.js

```
import React from 'react';
import {
 View,
 Image,
 Text,
 Dimensions,
 TouchableOpacity,
 StyleSheet,
} from 'react-native';
import { TextInput, Button } from 'react-native-paper';
import { Toast } from 'react-native-toast-message';
import { RadioButton } from 'react-native-paper';
import Icon from 'react-native-vector-icons/Feather';
import { signup } from '../firebase/auth';
import { setUserInfo } from '../firebase/database';
import { updateProfile } from 'firebase/auth';

const SignUpScreen = ({ navigation }) => {
 const [name, setName] = React.useState('');
 const [password, setPassword] = React.useState('');
 const [email, setEmail] = React.useState('');
 const [country, setCountry] = React.useState('');
 const [age, setAge] = React.useState();
 const [gender, setGender] = React.useState('Male');

 const [securePassword, setSecurePassword] = React.useState(true);
 const [isLoading, setIsLoading] = React.useState(false);

 const handleSignup = async () => {
   setIsLoading(true);
   try {
     const userCredential = await signup(email, password);
     const user = userCredential.user;
     await updateProfile(user, {
       displayName: name,
       photoURL:
         gender === 'Male'
           ? 'https://i.ibb.co/dL6cvCB/male.png'
           : 'https://i.ibb.co/27QxHX2/female.png',
     });

     await setUserInfo(user.uid, {
       age: age,
       gender: gender,
       country: country,
     });

     Toast.show({
       type: 'success',
       text1: `Welcome ${name} 👏`,
     });
   } catch (error) {
     Toast.show({
       type: 'error',
       text1: error.message,
       text2: error.code,
     });
   }
```

```
  setIsLoading(false);
};

return (
  <>
    <View style={styles.header}>
      <View style={styles.headerLeft}>
        <TouchableOpacity onPress={() => navigation.goBack()}>
          <Icon
            style={styles.headerLeftIcon}
            color="black"
            size={25}
            name="chevron-left"
          />
        </TouchableOpacity>
      </View>
      <View style={styles.headerMiddle}>
        <Text style={styles.headerMiddleText}> Sign Up </Text>
      </View>
      <View style={styles.headerMiddleRight} />
    </View>

    <View style={styles.pageContent}>
      <Image
        style={styles.logoImage}
        resizeMode="contain"
        source={require('../assets/listed.png')}
      />

      <View style={styles.inputFrame}>
        <View style={styles.frameCentered}>
          <TextInput
            outlineColor="#efefef"
            left={
              <TextInput.Icon
                style={styles.textInputIcon}
                size={20}
                color="gray"
                name="account"
              />
            }
            placeholderTextColor={'#2f3030'}
            label="Name"
            style={styles.textInput}
            mode="outlined"
            onChangeText={setName}
            autoCapitalize="none"
            autoComplete="none"
            autoCorrect="none"
          />
        </View>
      </View>

      <View style={[styles.inputFrame, { marginTop: 12 }]}>
        <View style={styles.frameCentered}>
          <TextInput
            secureTextEntry={securePassword}
            right={
              <TextInput.Icon
                style={styles.textInputIcon}
                onPressIn={() => setSecurePassword(false)}
                onPressOut={() => setSecurePassword(true)}
                name="eye"
```

```
            />
          }
          outlineColor="#efefef"
          placeholderTextColor={'#2f3030'}
          label="Password"
          keyboardType="default"
          autoCapitalize="none"
          autoComplete="none"
          autoCorrect="none"
          style={styles.textInput}
          mode="outlined"
          onChangeText={setPassword}
        />
      </View>
    </View>

    <View style={[styles.inputFrame, { marginTop: 12 }]}>
      <View style={styles.frameCentered}>
        <TextInput
          left={
            <TextInput.Icon
              style={styles.textInputIcon}
              size={20}
              color="gray"
              name="email"
            />
          }
          outlineColor="#efefef"
          autoCapitalize="none"
          autoComplete="none"
          autoCorrect="none"
          placeholderTextColor={'#2f3030'}
          keyboardType="email-address"
          label="Email"
          style={styles.textInput}
          mode="outlined"
          onChangeText={setEmail}
        />
      </View>
    </View>

    <View style={[styles.inputFrame, { marginTop: 12 }]}>
      <View style={styles.rowInputFrame}>
        <View style={styles.rowInputFrameWidth}>
          <View style={styles.rowInputWidth}>
            <View style={styles.frameCentered}>
              <TextInput
                outlineColor="#efefef"
                placeholderTextColor={'#2f3030'}
                label="Country"
                style={styles.input}
                mode="outlined"
                onChangeText={setCountry}
              />
            </View>
          </View>
        </View>

        <View style={styles.rowInputFrameWidth}>
          <View style={styles.rowInputWidth}>
            <View style={styles.frameCentered}>
              <TextInput
                outlineColor="#efefef"
```

```
                  keyboardType="numeric"
                  maxLength={2}
                  placeholderTextColor={'#2f3030'}
                  label="Age"
                  style={styles.input}
                  mode="outlined"
                  onChangeText={setAge}
                />
              </View>
            </View>
          </View>
        </View>

        <View style={[styles.inputFrame, { marginTop: 12 }]}>
          <View style={styles.frameCentered}>
            <Text style={styles.text}>Gender</Text>
          </View>
        </View>

        <View style={[styles.inputFrame, { marginTop: 12 }]}>
          <View style={styles.genderFrame}>
            <RadioButton.Group
              onValueChange={val => setGender(val)}
              value={gender}>
              <RadioButton.Item
                labelStyle={styles.genderText}
                label="Male"
                value="Male"
              />
              <RadioButton.Item
                labelStyle={styles.genderText}
                color="pink"
                label="Female"
                value="Female"
              />
            </RadioButton.Group>
          </View>
        </View>

        <Button
          style={styles.button}
          onPress={handleSignup}
          loading={isLoading}
          mode="outlined">
          Sign Up
        </Button>
      </View>
    </>
  );
};

const styles = StyleSheet.create({
  header: {
    position: 'absolute',
    zIndex: 1,
    justifyContent: 'center',
    flexDirection: 'row',
    alignItems: 'center',
    width: '100%',
    height: Dimensions.get('window').height / 8.5,
  },
  headerLeft: {
```

```
    width: '15%',
    height: '100%',
    justifyContent: 'flex-end',
    alignItems: 'flex-end',
  },
  headerLeftIcon: {
    paddingBottom: 15,
  },
  headerMiddle: {
    width: '70%',
    height: '100%',
    justifyContent: 'flex-end',
    alignItems: 'center',
  },
  headerMiddleText: {
    paddingBottom: 17,
    color: 'black',
    fontSize: 17,
    // fontFamily: 'Roboto-Regular',
  },
  headerMiddleRight: {
    width: '15%',
    height: '100%',
    justifyContent: 'flex-end',
    alignItems: 'flex-start',
  },
  pageContent: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: 'white',
  },
  logoImage: {
    width: '60%',
    height: 50,
    marginBottom: 30,
  },
  inputFrame: {
    width: '100%',
    justifyContent: 'center',
    alignItems: 'center',
  },
  frameCentered: {
    width: '85%',
  },
  textInputIcon: {
    marginTop: 14,
  },
  textInput: {
    backgroundColor: 'white',
    height: 50,
  },
  rowInputFrame: {
    width: '92%',
    flexDirection: 'row',
    justifyContent: 'center',
  },
  rowInputFrameWidth: {
    width: '50%',
  },
  rowInputWidth: {
    width: '100%',
    justifyContent: 'center',
    alignItems: 'center',
```

```
  },
  rowInputFrameCentered: {
    width: '100%',
    justifyContent: 'center',
    alignItems: 'center',
  },
  text: {
    marginLeft: 3,
    fontSize: 14.5,
    // fontFamily: 'Roboto-Regular',
    color: '#757575',
  },
  genderFrame: {
    borderRadius: 5,
    width: '85%',
    borderWidth: 1,
    borderColor: '#efefef',
  },
  genderText: {
    color: '#757575',
  },
  button: {
    marginTop: 20,
  },
  input: {
    backgroundColor: 'white',
    height: 50,
  },
});

export default SignUpScreen;
```