**Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης**

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

# Πτυχιακή Εργασία

Υβριδικές τεχνολογίες λογισμικού και εφαρμογές cloud.

**ΣΑΡΙΔΑΚΗΣ ΖΑΧΑΡΙΑΣ ΑΜ: 3277**

# Πίνακας περιεχομένων

Πίνακας περιεχομένων ..... 2
Πίνακας εικόνων ..... 3
Abstract ..... 4
Σύνοψη ..... 5
Κεφάλαιο 1 ..... 6
1.1 Εισαγωγή ..... 6
1.2 Περί εφαρμογών ..... 7
1.3 Τεχνολογίες που θα χρησιμοποιήσω ..... 8
1.4 Δομή κειμένου ..... 9
Κεφάλαιο 2 ..... 11
2.1 Οδηγός Χρήστη ..... 11
2.1.1 Χρήστης ..... 11
2.1.2 Admin ..... 15
Κεφάλαιο 3 ..... 21
3.1 Firebase οδηγός ..... 21
3.2 Ionic Οδηγός ..... 22
3. 3 Οδηγός για Developers ..... 24
3.3.1 Εφαρμογή Admin ..... 28
3.3.2 Εφαρμογή Χρήστη ..... 53
Κεφάλαιο 4 ..... 69
Αξιολόγηση ..... 69
Βιβλιογραφία ..... 71

# Πίνακας εικόνων

Figure 1: Βασική Αρχιτεκτονική του WebView ..... 7
Figure2: : Βασική Αρχιτεκτονική του Apache Cordova ..... 9
Figure3: Πρώτες οθόνες της εφαρμογής χρήστη ..... 11
Figure 4 Λίστα σεναρίων και παράθυρό για την εισαγωγή κωδικού ..... 12
Figure5: Τύποι σεναρίων και ρυθμίσεις ..... 12
Figure6: Λήψη δεδομένων από τα σημεία ενδιαφέροντος ..... 13
Figure7: Markers και loader ..... 14
Figure8: Τροποποίηση σεναρίου ..... 15
Figure9: Σενάριο και τροποποίηση δεδομένων ..... 16
Figure 10 : Τροποποίηση Αύξων αριθμού και συντεταγμένων ..... 17
Figure 11: Ερωτήματα πολλαπλής επιλογής και δημιουργία σεναρίου ..... 18
Figure12: Προσθήκη Marker ..... 19
Figure 13: Spinner, δημιουργία ερωτημάτων πολλαπλής επιλογής και upload ..... 20
Figure 14: Παράδειγμα δομής σημείων ενδιαφέροντος και σεναρίου στην Firebase ..... 25
Figure15: Παράδειγμα δομής δεδομένων χρήστη ..... 26
Figure16 : Παράδειγμα δομής δεδομένων admin ..... 26
Figure 17 :Asynchronous programming landscape ..... 27

# Abstract

Smartphones are something more than a cell phone. Smartphones integrate many technologies as Bluetooth, geolocation, Wi-Fi accessibility and decent quality of camera. All those features can substitute a map and a common camera. The market for smartphones/tablets are growing fast and so the number of devices per user. Statistics shows billions of profits with many more to come the following years and its saturation point is not yet visible.

That has as a result the growth of application market as well. Additionally, in recent past many application that needs user to be outside to set their scenarios in action released and was massively accepted by the users. Inspired from all the above and especially success of that application. I decide to develop a hybrid application that will be compatible with most platforms and will be used for many separate occasions. The concept of application will be an admin that creates scenarios and a user that participates on it. Application technically will be two applications one for the user that participates and one for the admin to manage scenarios. Application will be developed with the use of ionic framework and Google technologies

# Σύνοψη

Τα smartphones είναι κάτι παραπάνω από κινητά τηλέφωνα. Ενσωματώνουν μεταξύ άλλων τεχνολογίες όπως το Bluetooth, geolocation, δυνατότητα σύνδεσης σε ασύρματο δίκτυο και αξιοπρεπή ποιότητα κάμερας. Όλα αυτά τα χαρακτηριστικά μπορούν να αντικαταστήσουν έναν χάρτη και μια φωτογραφική μηχανή. Αυτό έχει σαν αποτέλεσμα να αυξάνεται ο αριθμός των χρηστών αλλά και των συσκευών ανά χρήστη. Τα στατιστικά για την αγορά smartphone/tablets δείχνουν δισεκατομμύρια κέρδους με δυνατότητα για περισσότερα τα χρόνια που ακολουθούν.

Φυσικό επακολουθώ αυτής της μεγεθύνσεις είναι και η μεγέθυνση της αγοράς των εφαρμογών. Επιπροσθέτως πρόσφατα πολλές εφαρμογές που χρειάζονται ο χρήστης να είναι σε εξωτερικό χώρο για να ξεκινήσουν τα σενάρια τους και έτυχαν ευρείας αποδοχής από τους χρήστες. Εμπνευσμένος από όλα τα παραπάνω αποφάσισα να αναπτύξω μια υβριδική εφαρμογή που θα είναι συμβατή με τις περισσότερες πλατφόρμες και θα έχει δυνατότητα για πολλές χρήσης. Η ιδέα γύρω από την εφαρμογή θα είναι ένας admin που θα δημιουργεί σενάρια και ένας χρήστης που θα συμμετέχει σε αυτά. Ουσιαστικά η εφαρμογή θα αποτελείται από δύο εφαρμογές μια για τον χρήστη και μια για τον admin για να διαχειρίζεται τα σενάρια. Η εφαρμογή θα αναπτυχθεί με την χρήση του Ionic framework και των τεχνολογιών τις Google.

# Κεφάλαιο 1

## 1.1 Εισαγωγή

Τα τελευταία 10 χρόνια έχει αυξηθεί ο αριθμός των ασύρματων συσκευών όπως τα smartphones και tablets. Ο μέσος άνθρωπος έχει στην κατοχή του τουλάχιστον μία συσκευή. Η χρήση τους ποικίλει από την επικοινωνία και την διασκέδαση μέχρι και την χρήση για εκπαιδευτικούς σκοπούς. Αυτό έχει σαν αποτέλεσμα την δημιουργία μιας αγοράς εφαρμογών για αυτές τις συσκευές που μεγαλώνει με γρήγορους ρυθμούς. Στα πρώτα χρόνια της ύπαρξης τους οι εφαρμογές αναπτύσσονταν αποκλειστικά native και για μια πλατφόρμα. Τα τελευταία χρόνια ήρθαν στο προσκήνιο οι υβριδικές εφαρμογές οι οποίες έχουν την δυνατότητα να υποστηρίζουν πολλές πλατφόρμες.

Η πτυχιακή μου εργασία αναφέρεται σε μία υβριδική εφαρμογή. Η ιδέα της πτυχιακής έχει να κάνει με σενάρια τα οποία λαμβάνουν χώρα σε εξωτερικό/εσωτερικό χώρο. Τα σενάρια δημιουργούνται από τον admin με την χρήση κάποιων σημείων ενδιαφέροντος. Τα σημεία ενδιαφέροντας είναι τοποθεσίες πάνω στον χάρτη όπου ο admin έχει την δυνατότητα να επισυνάψει κείμενο, εικόνα,video,ήχο και ερωτήσεις πολλαπλής επιλογής σε αυτό. Ο χρήστης θα διαλέξει ένα από τα διαθέσιμα σενάρια και όταν θα φτάσει στη περιοχή όπου ο admin έχει ορίσει ένα σημείο ενδιαφέροντος θα ειδοποιηθεί και τα δεδομένα θα κατεβούν στην συσκευή του. Η εφαρμογή αποτελείται από δύο εφαρμογές μία για τον admin και μία για τον χρήστη. Η εφαρμογή θα υλοποιηθεί με την χρήση του ionic framework και των τεχνολογιών τις Google όπως η firebase, τα Google maps όπως επίσης και beacons σε μικρή κλίμακα.

Στο πρώτο κεφάλαιο θα εξηγήσω εν συντομία τις τεχνολογίες που ανέφερα στην παραπάνω παράγραφο και κάποια γενικά για εφαρμογές. Στο δεύτερο κεφάλαιο θα υπάρχει ένα user guide όπου θα εξηγώ την χρήση της εφαρμογής έτσι ώστε να είναι δυνατή η χρήση της εφαρμογής από τον κάθε ένα. Στο τρίτο κεφάλαιο θα υπάρχει ένα developer's guide όπου θα εξηγώ των κώδικα της εφαρμογής ώστε ο καθένας που ενδιαφέρεται να μπορεί να τον μεταβάλει.

## 1.2 Περί εφαρμογών

Οι native εφαρμογές αναπτύσσονται αποκλειστικά για μια πλατφόρμα. Συνήθως χρησιμοποιούνται γλώσσες προγραμματισμού όπως η Java για Android και η objective -c για iOS. Η υποστήριξη για αυτές τις εφαρμογές είναι πιο ακριβές σε σχέση με τις υβριδικές εφαρμογές για τον λόγο ότι πρέπει να υποστηρίζεται διαφορετική εφαρμογή για κάθε πλατφόρμα. Οι native εφαρμογές μπορούν να αξιοποιήσουν πλήρως τις δυνατότητες μιας συσκευής και αυτός είναι ο λόγος που είναι μια καλή επιλογή για παιχνίδια με απαιτητικά γραφικά.

Οι υβριδικές εφαρμογές όπως προ είπα μπορούν να λειτουργήσουν σε πολλές συσκευές και πλατφόρμες. Για την ανάπτυξη τους χρησιμοποιούνται HTML, CSS και Javascript. Μια καλά ανεπτυγμένη εφαρμογή συμπεριφέρεται σαν μια native σε βαθμό που χρήστης δεν μπορεί να προσδιορίσει αν εάν είναι υβριδική ή όχι. Το κόστος για την υποστήριξη της είναι λιγότερο γιατί πρέπει να υποστηριχθεί μια εφαρμογή για πολλές πλατφόρμες αντί για μια για κάθε πλατφόρμα. Αυτό έχει σαν αποτέλεσμα η εταιρεία να χρειάζεται να προσλάβει έναν προγραμματιστή αντί για πολλούς. Επιπροσθέτως είναι πιθανό η εφαρμογή να δουλεύει χωρίς προβλήματα ή με μερικά updates όταν κυκλοφορήσει νέα έκδοση του λειτουργικού συστήματος. Η κύρια διαφορά μεταξύ μιας native και μιας υβριδικής εφαρμογής είναι ότι οι υβριδικές εφαρμογές κάνουν χρήση του WebView της πλατφόρμας μέσα από ένα native container.

Τα WebViews χρησιμοποιούν την αρχή γράψε μια φορά, τρέξε παντού είναι σαν μια καρτέλα web browser η οποία δίνει την δυνατότητα της ανάπτυξης σε Html, CSS και JavaScript. Ο developer μπορεί να χρησιμοποίηση ένα framework όπως το Apache Cordova ο οποίος είναι και ο πιο γνωστός για την χρήση των native δυνατοτήτων μιας συσκευής. Υπάρχουν πολλά διαφορετικά WebViews ακόμα και για desktops. Γίνεται μια προσπάθεια τα WebViews να αντικαταστήσουν τα in-app browser με τεχνολογίες όπως ο safari view controller για iOS και τον Chrome custom tabs για Android. Στην παρακάτω εικόνα μπορείτε να δείτε την αρχιτεκτονική τουWebView.

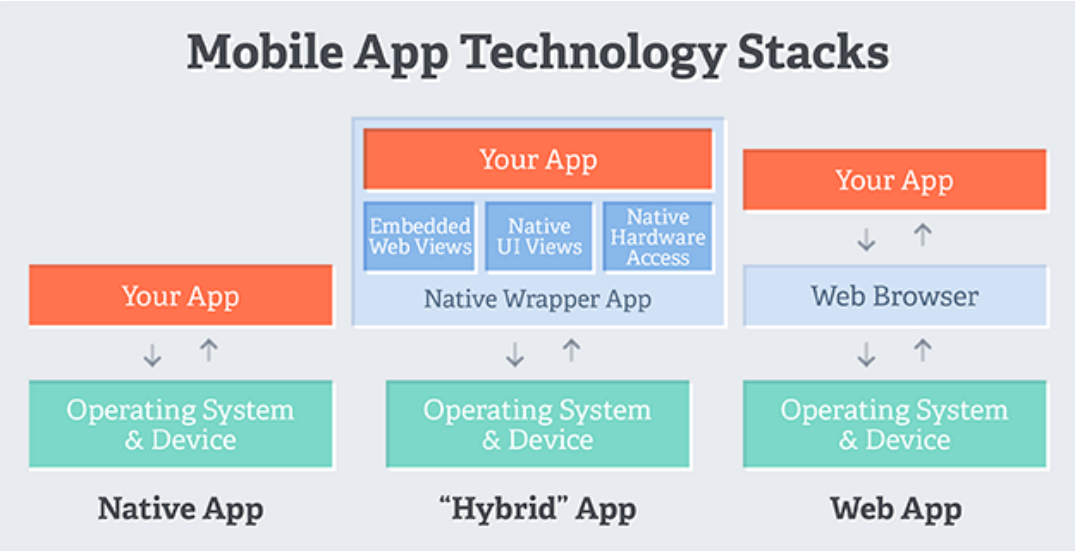


Figure 1: Βασική Αρχιτεκτονική του WebView

## 1.3 Τεχνολογίες που θα χρησιμοποιήσω

Για την πτυχιακή μου εργασία έκανα χρήση του ionic 2 framework. Το ionic framework είναι ένα ανοιχτού κώδικα SDK για εφαρμογές που τρέχουν στο web, ασύρματες συσκευές ή ακόμα και desktop. Το Ionic κάνει χρήση των CSS, HTML5 και Sass. Επικεντρώνεται περισσότερο σε UI και χρειάζεται την Angular για να δουλέψει με πλήρης δυνατότητες. Η Angular είναι μια δωρεάν front-end web πλατφόρμα ανεπτυγμένη από την Google. Κυκλοφόρησε τον Σεπτέμβριο του 2016 και διαδέχτηκε την προηγούμενη έκδοση με όνομα AngularJS με την οποία έχει σημαντικές διαφορές. Οι Angular developers μπορούν να κάνουν χρήση JavaScript ή TypeScript η οποία γίνεται compile σε JavaScript.

Η firebase είναι ένα μια web πλατφόρμα ανάπτυξης η οποία ανήκει στην Google. Η firebase αποτελείται από τα Analytics, Cloud Messaging, πιστοποίηση, βάση δεδομένων πραγματικού χρόνου, Storage, Hosting και Crash Reporting. Όλες οι υπηρεσίες της firebase παρέχονται δωρεάν για χρήση μικρής κλίμακας. Στην πτυχιακή μου εργασία χρησιμοποιώ τα πιστοποίηση, βάση δεδομένων και storage. Η πιστοποίηση παρέχει στον developer έναν αρκετά εύκολο τρόπο να χρησιμοποιήσει SDK και προ-υπάρχουσες βιβλιοθήκες. Η βάση πραγματικού χρόνου της firebase είναι μια NoSQL cloud βάση δεδομένων. Τα δεδομένα αποθηκεύονται σαν JSON και συγχρονίζονται ανεξάρτητα της πλατφόρμας που την χρησιμοποιεί. Το storage είναι μία υπηρεσία αποθήκευσης object. Ο χρήστης μπορεί να ανεβάσει δεδομένα και περιεχόμενο δικής του δημιουργίας.

Ένα beacon ή BLE (Bluetooth Low Energy) είναι ένας μικρός επαναλήπτης Bluetooth σήματος που λειτουργεί με μπαταρία. Το μέγεθος του είναι αρκετά μικρό και μεταδίδει ένα σήμα το οποίο μπορούν να λάβουν συσκευές με ικανότητα λήψεις Bluetooth σήματος όταν βρίσκονται στην περιοχή καλύψεις του. Τα Beacons εκπέμπουν μοναδικά id τα οποία είναι μία μίξη γραμμάτων και αριθμών. Μια εφαρμογή μπορεί για παράδειγμα να αναγνωρίσει το id σαν ένα κουπόνι προσφοράς ή κάποιες πληροφορίες σε ένα μουσείο. Τα μοντέρνα Beacon ξεκίνησαν με το iBeacon της Apple τα οποία μπορούν να στείλουν μερικά bits δεδομένων. Η Google κυκλοφόρησε το δικό της beacon με όνομα Eddystone.

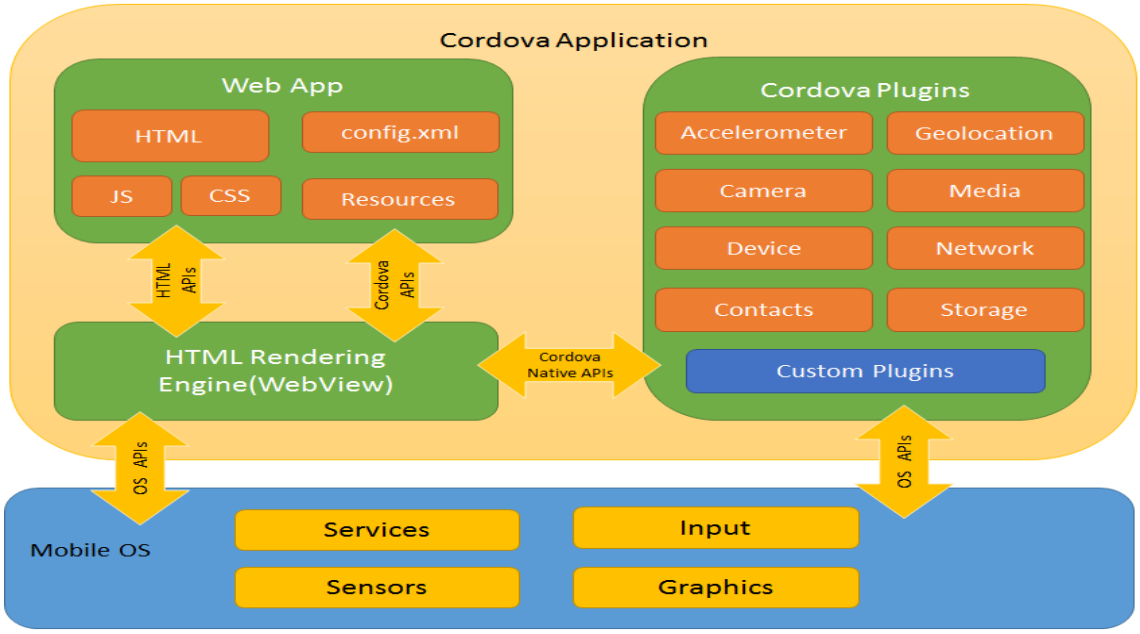


Figure2: : Βασική Αρχιτεκτονική του Apache Cordova

Το Apache Cordova είναι ένα ανοικτού κώδικα development framework. Παρέχει μια διεπαφή foreign function για να σύνδεση native τεχνολογίες σε μία εφαρμογή. Τα plugins είναι ένα απαραίτητο μέρος για την επικοινωνία του Cordova με τα native μέρη του API της συσκευής και επιτρέπει στον developer να καλέσει native κώδικα από Javascript. Η Cordova διατηρεί μερικά core plugins για την πρόσβαση σε native τεχνολογίες. Επίσης υπάρχουν third party plugins όπου κάθε ένας μπορεί να αναπτύξει. Τα συγκεκριμένα όμως μπορεί να μην είναι διαθέσιμα για όλες τις εφαρμογές. Η Cordova με την χρήση του WebView παρέχει στην εφαρμογή μία διεπαφή χρήστη. Η εφαρμογή εκτελείται σαν μια web σελίδα η οποία αποτελείται από ένα HTML file που καλεί ένα CSS, HTML, JavaScript και όλα τα αρχεία που χρειάζονται για να τρέξει η εφαρμογή. Στην παραπάνω εικόνα μπορεί να δει κάποιος την Cordova αρχιτεκτονική.

Το Apache Cordova είναι συμβατό με τις εξής πλατφόρμες Android, Blackberry 10, iOS, OSX, Ubuntu, Windows, WP8. Υπάρχουν δύο τρόποι ανάπτυξης εφαρμογών η Cross-platform (CLI) και η Platform-centered . Η Cross-platform (CLI) επικεντρώνεται στο Cordova CLI. Το CLI δίνει την δυνατότητα να γίνει ένα project build μία φορά και να τρέξει σε πολλές πλατφόρμες. Απομακρύνοντας αρκετοί από την λειτουργικότητα του lower-level shell scripts. Η Platform-centered προσεγγίσει είναι χρήσιμη στην περίπτωση που ο developer θέλει να κάνει build μία εφαρμογή για μια συγκεκριμένη πλατφόρμα και να την αλλάξει σε χαμηλότερο επίπεδο. Η συγκεκριμένη προσέγγιση βασίζεται σε ένα σύνολο από χαμηλού επιπέδου scripts τα οποία συνδυάζονται για κάθε πλατφόρμα τα οποία υποστηρίζουν.

## 1.4 Δομή κειμένου

Το συγκεκριμένο κείμενο αποτελείται από τρία κεφάλαια και αρκετά υποκεφάλαια. Στο πρώτο κεφάλαιο υπάρχει μία εισαγωγή και μια σύντομη περιγραφή που θα χρησιμοποιήσω για την

ανάπτυξη της εφαρμογής. Στο δεύτερο κεφάλαιο υπάρχει οδηγός χρήσης για την εφαρμογή του χρήστη και του admin. Το κεφάλαιο χωρίζεται σε δύο υποκεφάλαια το πρώτο αφορά την εφαρμογή του admin και το δεύτερο του χρήστη. Για την καλύτερη κατανόηση γίνεται χρήση screenshots.Το τρίτο κεφάλαιο ασχολείται με μία εισαγωγή στην δομή που χρησιμοποίησα στην βάσης δεδομένων της firebase και μία επεξήγηση για τα observables και τα promises. Επιπροσθέτως στο τρίτο κεφάλαιο υπάρχουν οδηγοί χρήσης για developers τόσο για την εφαρμογή του admin όσο και του χρήστη. Σε αμφότερους τους οδηγούς υπάρχει κώδικας επισυνημμένος για την διευκόλυνση της κατανόησης του.

# Κεφάλαιο 2

## 2.1 Οδηγός  Χρήστη

Αυτό το κεφάλαιο περιέχει δύο οδηγούς χρήσης για την εφαρμογή. Ο πρώτος είναι ένας οδηγός χρήσης για τον απλό χρήστη και ο δεύτερος για τον admin. Θα υπάρχουν κάποια screenshots έτσι ώστε να είναι πιο κατανοητή η χρήση της εφαρμογής. Τα screenshot αντιστοιχούν στην παράγραφο ή στις παραγράφους κάτω από αυτά.

### 2.1.1 Χρήστης

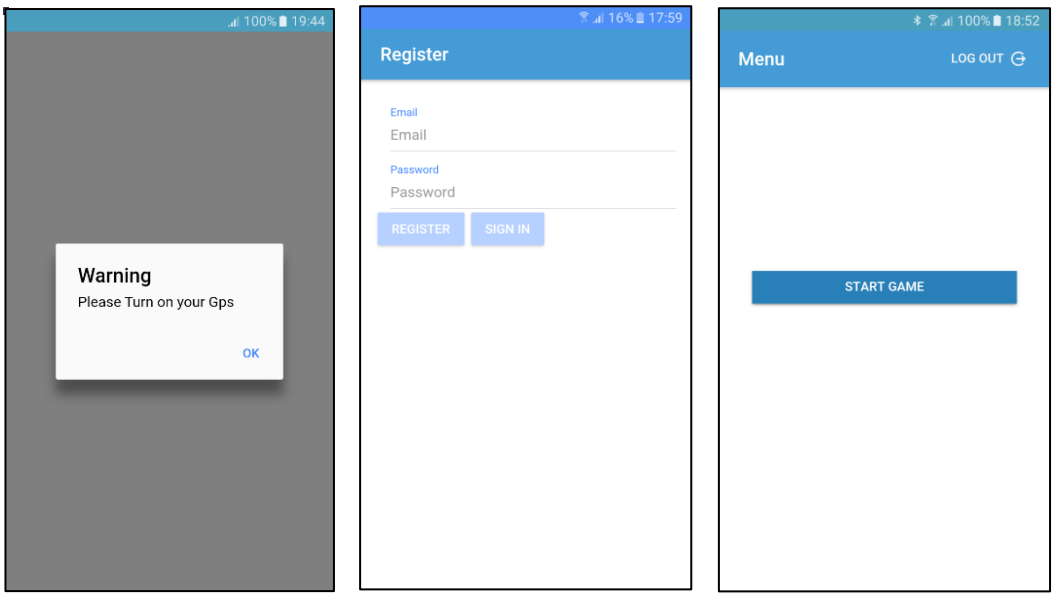


Figure3:  Πρώτες οθόνες της εφαρμογής χρήστη

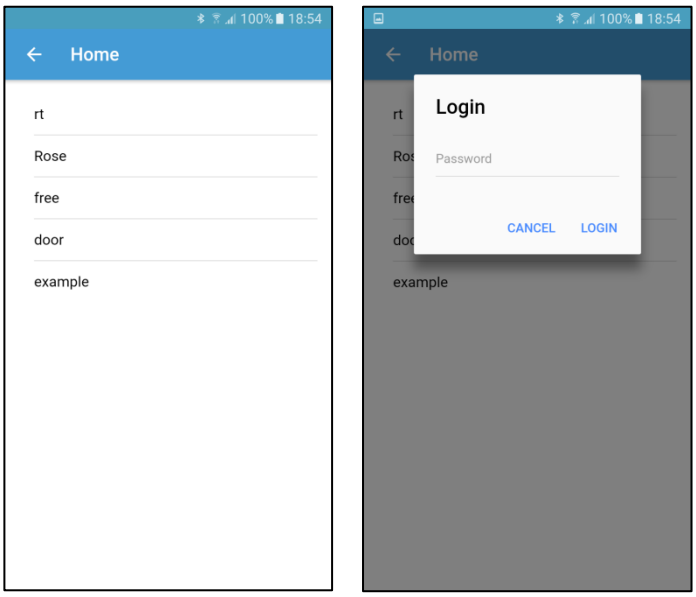


Figure 4 Λίστα σεναρίων και παράθυρό για την εισαγωγή κωδικού

Αρχικά ο χρήστης θα πρέπει να ενεργοποιήσει τα Bluetooth, GPS, Wi-Fi ή τα δεδομένα της συσκευής. Σε αντίθετη περίπτωση θα εμφανιστεί μια κενή άσπρη οθόνη με τα κατάλληλα μηνύματα. Εάν είναι ενεργοποιημένα θα εμφανιστεί στην οθόνη μια φόρμα με δύο πεδία για email και κωδικό. Όπως και δύο κουμπιά για registration και sign in αντίστοιχα. Μετά την εγγραφή ή σύνδεση του χρήστη θα εμφανιστεί ένα menu με ένα κουμπί με την ένδειξη start game στο κέντρο της οθόνης και ένα δεύτερο log out κουμπί στην πάνω δεξιά πλευρά της. Με το πάτημα του StartGame κουμπιού μια λίστα με τα διαθέσιμα σενάρια εμφανίζεται. Με την επιλογή ενός σεναρίου ένα παράθυρο εμφανίζεται στο προσκήνιο και ζητάει από τον χρήστη των κωδικό του σεναρίου. Εάν ο κωδικός είναι λάθος θα εμφανιστή κατάλληλο μήνυμα.

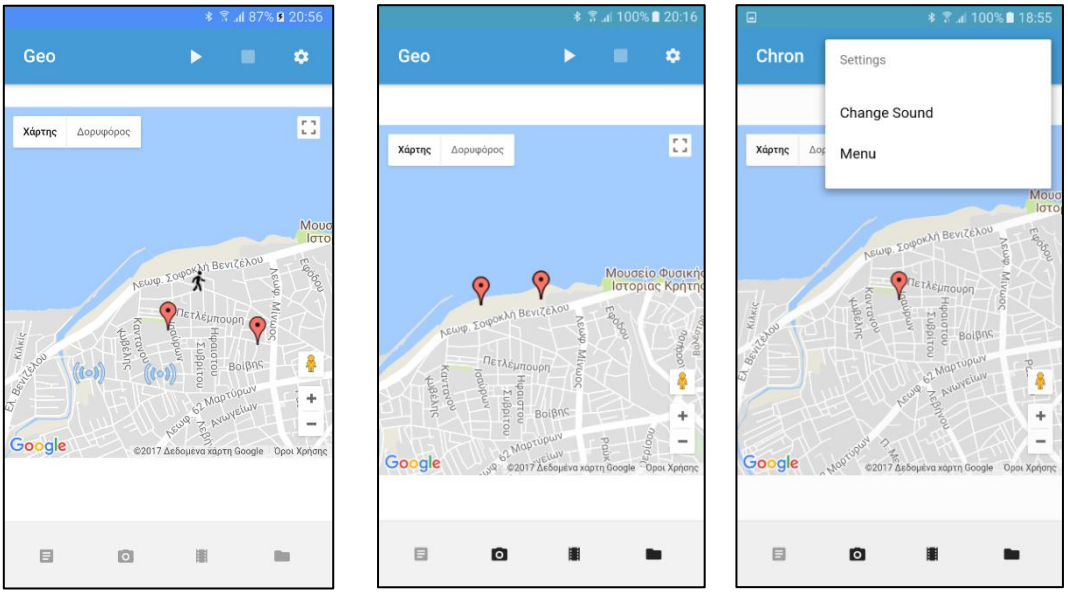


Figure5: Τύποι σεναρίων και ρυθμίσεις

Στην επόμενη οθόνη στο κάτω μέρος υπάρχουν τέσσερα κουμπιά, όλα είναι απενεργοποιημένα. Τα τρία από αυτά είναι για το ανέβασμα δεδομένων από τον χρήστη σαν διά δράση στο σενάριο. Τα δεδομένα αυτά μπορεί να είναι video, φωτογραφία από την κάμερα και δεδομένων από την συσκευή. Τα συγκεκριμένα κουμπιά ενεργοποιούνται εάν βρεθεί κάποιο σημείο ενδιαφέροντος. Το πρώτο από αριστερά κουμπί θα παραμείνει απενεργοποιημένο έως ότου υπάρξει ένα σημείο ενδιαφέροντος με ερωτήσεις πολλαπλής επιλογής η οποίες έχουν προστεθεί από τον admin. Στο επάνω μέρος της οθόνης υπάρχουν τρία κουμπιά με τα σύμβολα play, stop και ρυθμίσεις. Το κουμπί play ξεκινάει την αναζήτηση για σημεία ενδιαφέροντος, το stop σταματάει την αναζήτηση και το κουμπί ρυθμίσεις ανοίγει το αντίστοιχο menu. Ο τύπος του σεναρίου εμφανίζεται στην αριστερή γωνία. Οι τύποι σεναρίου είναι δύο Geographically με συντομογραφία Geo και chronologically με συντομογραφία Chron. Στο menu των ρυθμίσεων υπάρχουν δύο επιλογές μια για την αλλαγή του ήχου ειδοποίησης και ένα για την επιστροφή στο menu.

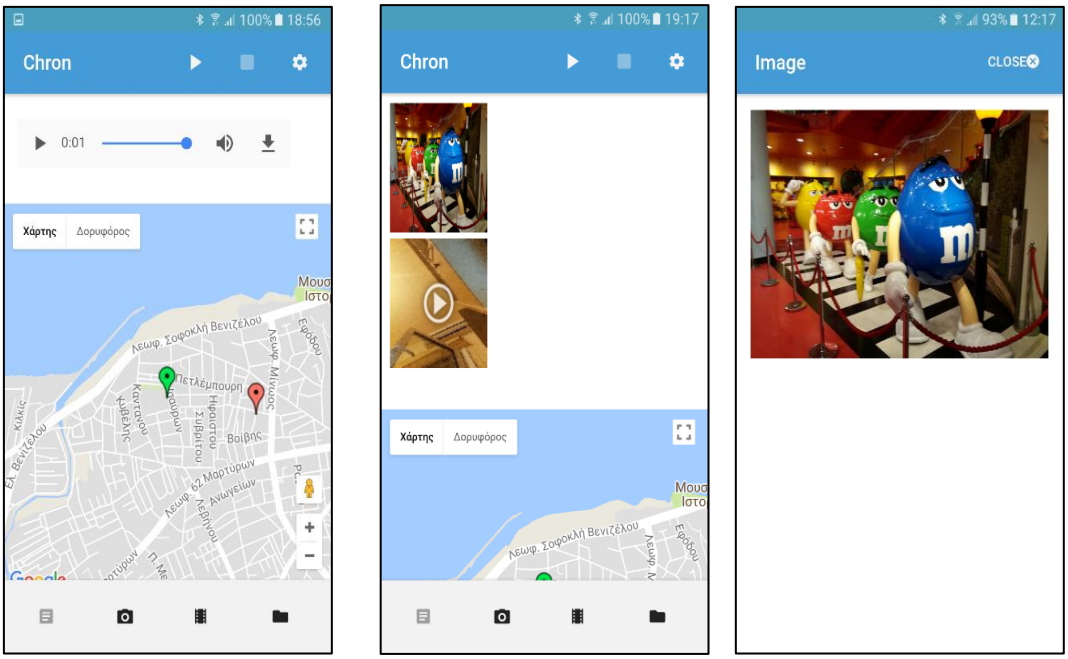


Figure6: Λήψη δεδομένων από τα σημεία ενδιαφέροντος

Επίσης ένας χάρτης εμφανίζεται στην οθόνη με τα διαθέσιμα σημεία ενδιαφέροντος. Υποθέτοντας ότι το σενάριο είναι τύπου geographically όλα τα σημεία εμφανίζονται στον χάρτη έτσι ώστε ο χρήστης να μπορεί να επιλέξει πιο θα επισκεφθεί. Στην περίπτωση που το σενάριο είναι τύπου chronologically μόνο το επόμενο σημείο θα είναι διαθέσιμο κάθε φορά. Όταν ο χρήστης βρεθεί στο πεδίο ενός σημείου ενδιαφέροντος και για τους δύο τύπους σεναρίου ο marker θα αλλάξει χρώμα από κόκκινο σε πράσινο. Ένας ήχος ειδοποίησης θα ακουστεί η αναζήτηση θα σταματήσει και τα δεδομένα του σεναρίου θα εμφανιστούν στην οθόνη μόλις κατεβούν από το cloud. Εάν τα δεδομένα είναι video ή εικόνα θα εμφανιστεί στην οθόνη ένα thumbnail. Μόλις ο χρήστης πατήσει πάνω του θα εμφανιστή ένα παράθυρο modal δείχνοντας την εικόνα σε μεγαλύτερο μέγεθος ή τον video player. Στην περίπτωση του ήχου ένας player εμφανίζεται. Το ηχητικό κομμάτι μπορεί να ακούγεται αυτόματα ή με το πάτημα του κουμπιού play. Το κείμενο απλά εμφανίζεται στην οθόνη.

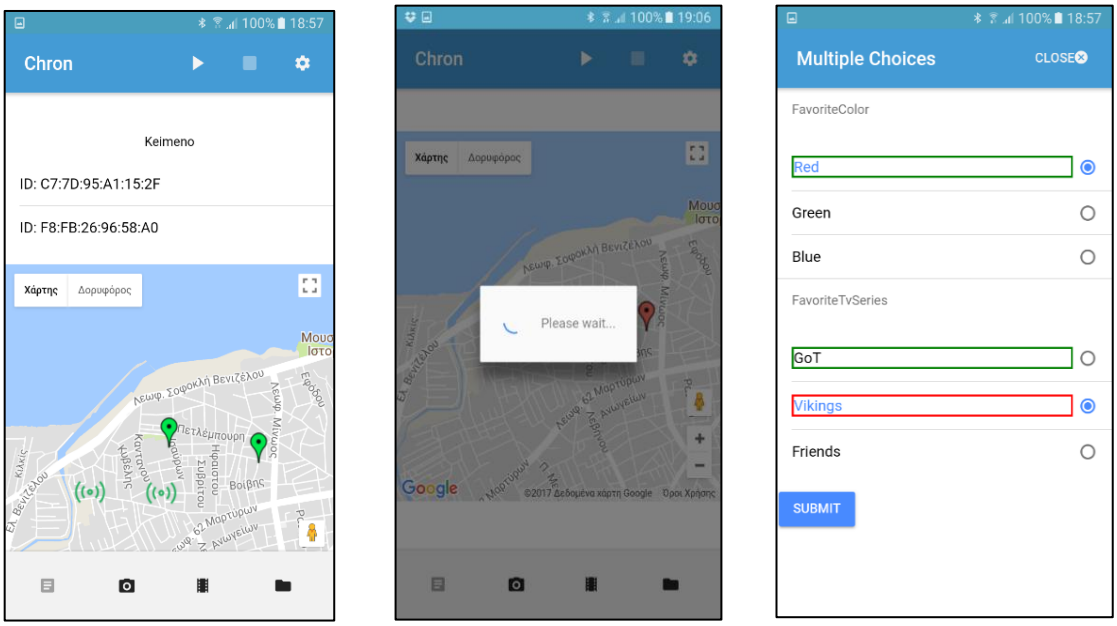


Figure7: Markers και loader

Για τα beacons σημεία ενδιαφέροντος υπάρχει ένα διαφορετικός marker. Πολλαπλά beacons μπορούν να υπάρξουν στο ίδιο γεωγραφικό μήκος και πλάτος. Σε αυτήν την περίπτωση επάνω στον χάρτη υπάρχει ένας marker για όλα και τα id τους εμφανίζονται σε μία λίστα επάνω από τον χάρτη. Είναι δυνατόν ο χρήστης να δει τα δεδομένα ενός σημείου ενδιαφέροντος πατώντας πάνω σε ένα marker κάνοντας ένα διπλό κλικ πάνω σε ένα απλό marker και ένα κλικ πάνω σε κάποιο από τα id της λίστας. Κατά την διάρκεια του upload της δια δράσης του χρήστη στο σενάριο ένας loader εμφανίζεται στην οθόνη. Οι επιτρεπτοί τύποι δεδομένων είναι mp4, mp3 και jpeg. Για οποιονδήποτε άλλο τύπο αρχείου εμφανίζεται στην οθόνη κατάλληλο μήνυμα. Στην περίπτωση που υπάρχουν ερωτήματα πολλαπλής επιλογής το αντίστοιχο κουμπί πάει να είναι απενεργοποιημένο και με το πάτημα του μία νέα οθόνη εμφανίζεται. Σε αυτήν την οθόνη υπάρχει μια λίστα με τις πιθανές απαντήσεις. Μετά το submit στην σωστά επιλεγμένη απάντηση θα εμφανιστεί ένα πλαίσιο χρώματος πράσινου και στην λάθος ένα κόκκινο.

### 2.1.2 Admin

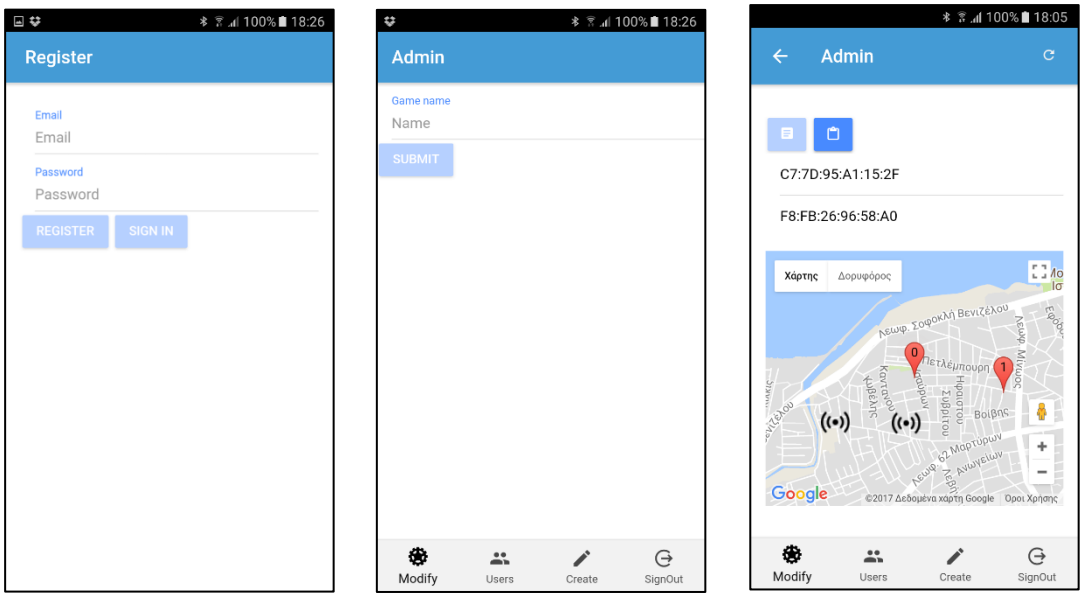


Figure8: Τροποποίηση σεναρίου

Η εφαρμογή για τον admin έχει την δυνατότητα για εγγραφή ή sign in. Απαιτείται διαφορετικός λογαριασμός από αυτόν για την εφαρμογή του χρήστη. Σε κάθε οθόνη τις εφαρμογής υπάρχουν τέσσερα κουμπιά με όνομα modify, users, create και sign out. Με το πάτημα του κουμπιού users εμφανίζεται μία λίστα με τα email των χρηστών. Με το πάτημα ενός email εμφανίζονται όλα τα σενάρια τα οποία έχει συμμετάσχει ο χρήστης. Εάν ο admin επιλέξει ένα από αυτά τα σενάρια εμφανίζεται η διάδραση του χρήστη στην οθόνη. Στις περιπτώσεις εικόνας ή video εμφανίζεται ένα modal παράθυρο με την εικόνα σε μεγαλύτερο μέγεθος ή το video player.

Η πρώτη οθόνη στην εφαρμογή είναι το modify. Αρχικά εμφανίζεται ένα input όπου ο admin πληκτρολογεί το όνομα του σεναρίου που θέλει να τροποποιήσει. Μετά το submit εμφανίζεται μια καινούργια οθόνη με έναν χάρτη με όλα τα σημεία ενδιαφέροντος επάνω καθώς και μια λίστα με τα id των beacons. Δύο κουμπιά υπάρχουν επάνω από τον χάρτη το αριστερό κουμπί είναι απενεργοποιημένο. Ενεργοποιείται εάν υπάρχουν ερωτήσεις πολλαπλής επιλογής. Με ένα κλικ πάνω στα απλά σημεία ενδιαφέροντος ή στην λίστα με τα beacon ids εμφανίζονται τα συνημμένα δεδομένα τους.

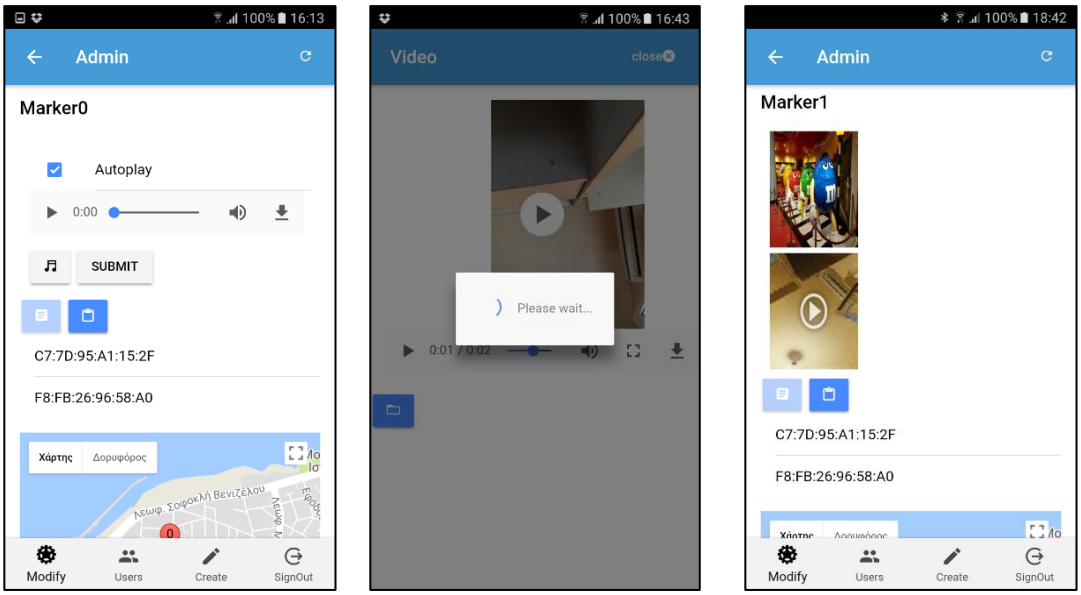


Figure9: Σενάριο και τροποποίηση δεδομένων

Με ένα κλικ πάνω στο thumbnail εμφανίζεται ένα modal με ένα κουμπί κάτω από την εικόνα ή το video. Το κουμπί αυτό ανοίγει το filesystem της συσκευής για να επιλέξει ο admin μια νέα εικόνα ή video για να αντικαταστήσει το ήδη υπάρχων. Καθόλη την διάρκεια του upload ένας loader εμφανίζεται στην οθόνη. Μετά το κλείσιμο του παραθύρου τα δεδομένα του σημείου ενδιαφέροντος ανανεώνονται. Στην περίπτωση του κειμένου εμφανίζεται ένα textbox ή μια λίστα με κείμενο και ένα κουμπί submit για κάθε ένα από αυτά. Στην περίπτωση του textbox το κείμενο μπορεί να είναι μια παράγραφος. Για την περίπτωση του ηχητικού κομματιού υπάρχει ένα checkbox με την ένδειξη autoplay πάνω από τον player και δύο κουμπιά κάτω από αυτόν. Ο admin έχει την δυνατότητα να αλλάξει το autoplay επιλέγοντας ή αποεπιλέγοντας το checkbox, να αλλάξει το ηχητικό κομμάτι επιλέγοντας ένα άλλο από το filesystem πατώντας πάνω στο κουμπί με την νότα. Με το submit και για όση ώρα διαρκεί το upload εμφανίζεται στην οθόνη ένας loader. Για να εμφανισθούν οι αλλαγές θα πρέπει ο admin να  κάνει refresh την σελίδα πατώντας στο κουμπί που βρίσκεται στην πάνω δεξιά γωνία.

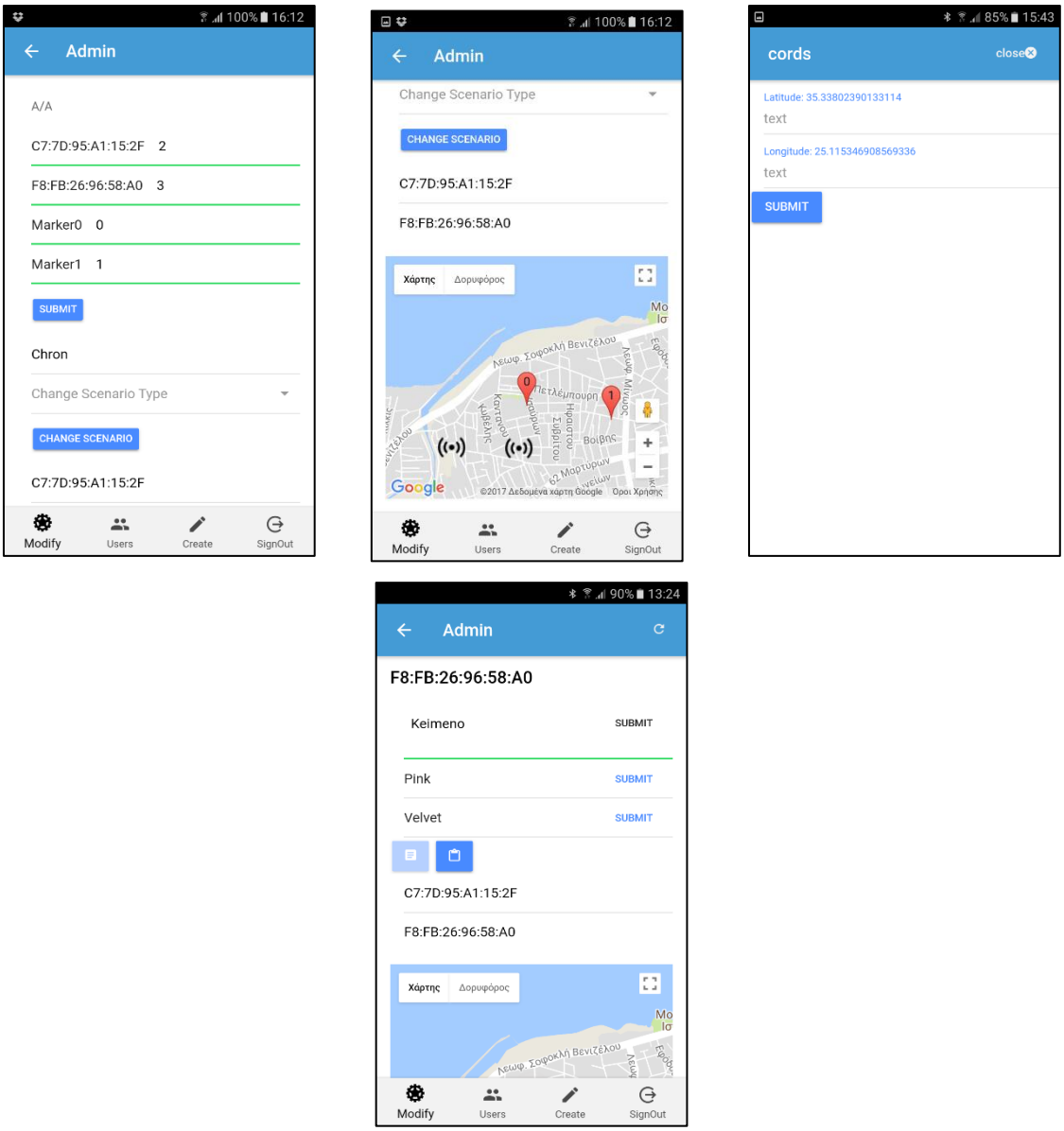


Figure *10* : Τροποποίηση Αύξων αριθμού και συντεταγμένων

Πατώντας στο κουμπί που βρίσκεται στα δεξιά επάνω από τον χάρτη. Εμφανίζεται μία οθόνη για την αλλαγής του τύπου του σεναρίου, του αύξοντα αριθμού του σημείου ενδιαφέροντος και του γεωγραφικού μήκους και πλάτους. Για την αλλαγή του αύξοντα αριθμού εμφανίζεται μια λίστα με τα ονόματα των σημείων ενδιαφέροντος. Ο admin μπορεί να μεταβάλει τη σειρά τους αλλάζοντας των αύξοντα αριθμό και πατώντας στο κουμπί submit. Εάν υπάρξουν τυχόν διπλότυπα εμφανίζεται κατάλληλο μήνυμα λάθους στην οθόνη. Στην περίπτωση της αλλαγής του τύπου του σεναρίου εμφανίζεται στο προσκήνιο ένα menu επιλογής με τους διαθέσιμους τύπους σεναρίου. Ο admin επιλέγει κάποιον από τους διαθέσιμους, πατάει το κουμπί με την ένδειξη οκ και κατόπιν το κουμπί με την ένδειξη change scenario. Στην περίπτωση αλλαγής του γεωγραφικού πλάτους και μήκους, με ένα

κλικ πάνω στον κόκκινο marker ή στο beacon id εμφανίζεται στο προσκήνιο ένα παράθυρο με τις ήδη υπάρχουσες τιμές και δύο textbox για τις νέες.

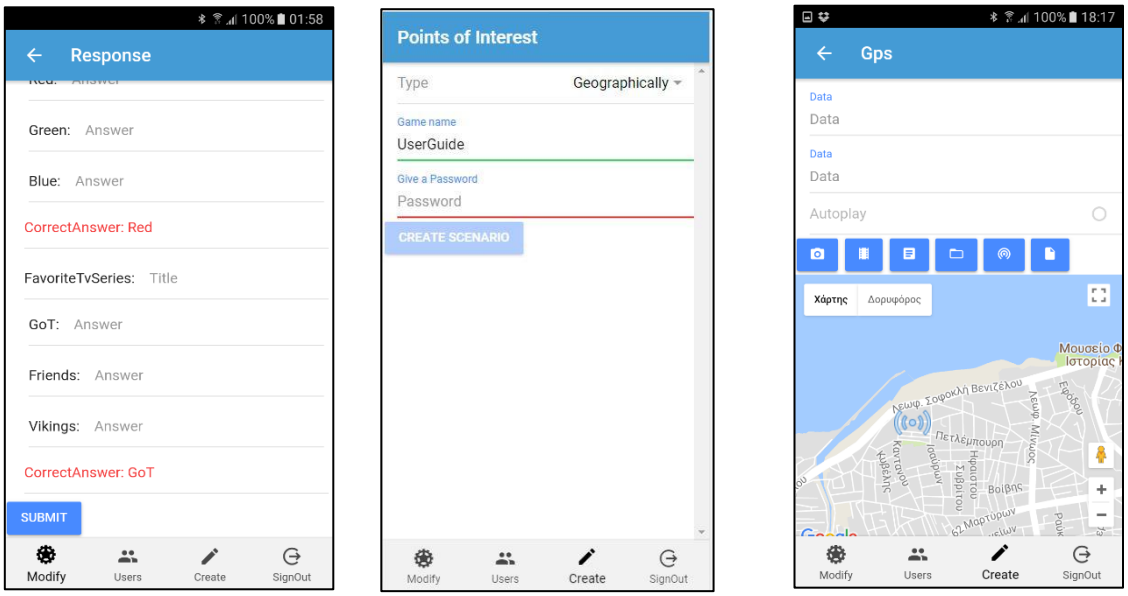


Figure *11*: Ερωτήματα πολλαπλής επιλογής και δημιουργία σεναρίου

Για την τροποποίηση των ερωτήσεων πολλαπλής επιλογής υπάρχει ένα κουμπί στα αριστερά επάνω από τον χάρτη. Με το πάτημα του μια νέα οθόνη εμφανίζεται με όλα τα ερωτήματα και τις σωστές τους απαντήσεις σε κόκκινο χρώμα. Αλλαγές μπορούν να γίνουν είτε σε ολόκληρο το ερώτημα, σε κάποια απάντηση ή στον τίτλο του. Μία άλλη επιλογή είναι η δημιουργία ενός καινούργιου σεναρίου. Με το πάτημα του κουμπιού create εμφανίζεται μια οθόνη με δύο textbox ένα για το όνομα του σεναρίου, ένα για τον κωδικό και ένα menu για τον τύπο ου σεναρίου. Ο κωδικός δεν είναι κάποιο μέτρο ασφαλείας αλλά ένας τρόπος να εμποδίσει κάποιον από το να συμμετάσχει. Όλα τα προαναφερθέντα πεδία είναι υποχρεωτικά για την ενεργοποιήσει του κουμπιού create scenario. Στην επόμενη οθόνη υπάρχει μια φόρμα με δύο πεδία για κείμενο και ένα radio button με την ένδειξη Autoplay το οποίο ενεργοποιείται μετά την επιλογή ενός ηχητικού κομματιού. Επιπροσθέτως έξι κουμπιά βρίσκονται επάνω από τον χάρτη και ένα κάτω από αυτόν με την ένδειξη submit.

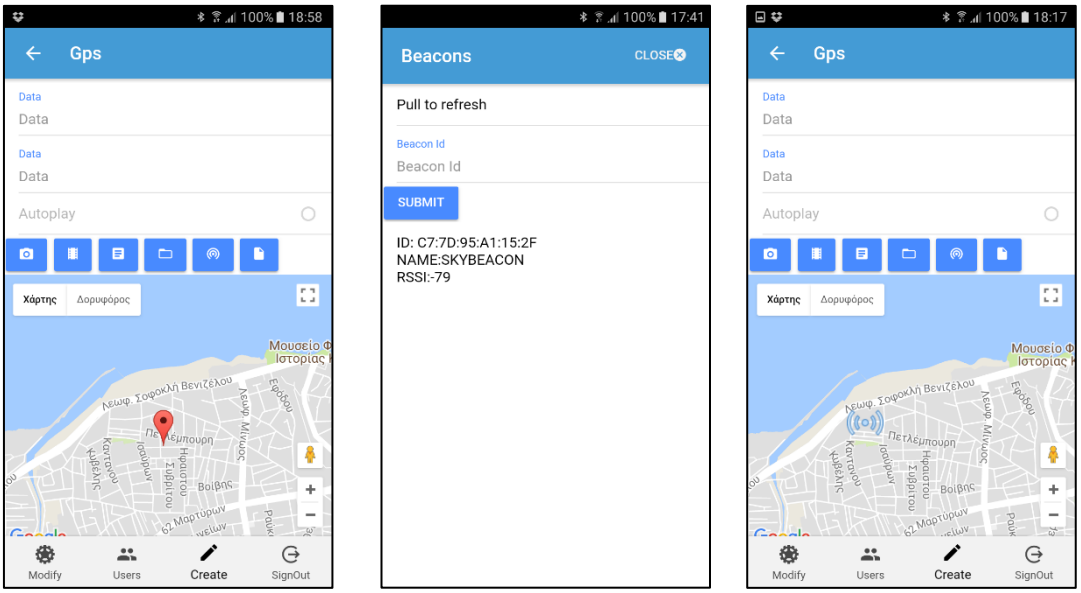


Figure12: Προσθήκη Marker

Η διαδικασία για την εισαγωγή σημείων ενδιαφέροντος είναι ο παρακάτω. Ο admin πρέπει να αγγίξει το επιθυμητό σημείο στον χάρτη που αντίστοιχοί σε κάποιο γεωγραφικό πλάτος και μήκος για να εμφανισθεί ένας marker. Για την εισαγωγή ενός beacon σημείου ενδιαφέροντος ο admin πρέπει να επιλέξει το δεύτερο από τα δεξιά κουμπί. Πατώντας το μια καινούργια οθόνη εμφανίζεται. Ο admin μπορεί να εισάγει το id ενός beacon από το υπάρχων πεδίο ή να επιλέξει ένα από τα διαθέσιμα αφότου ψάξει για αυτά τραβώντας την οθόνη προς τα κάτω. Μετά το κλείσιμο η επιλογή σημείου ενδιαφέροντος στον χάρτη αναφέρεται σε beacon. Αυτό φαίνεται από το εικονίδιο του marker που θα έχει αλλάξει όταν ο admin επιλέξει κάποιο σημείο πάνω στον χάρτη. Για την εισαγωγή beacon στην ίδια τοποθεσία μετά την επανάληψη τις διαδικασίας αρκεί ο admin να πατήσει επάνω στον ήδη υπάρχων beacon marker παρατεταμένα.

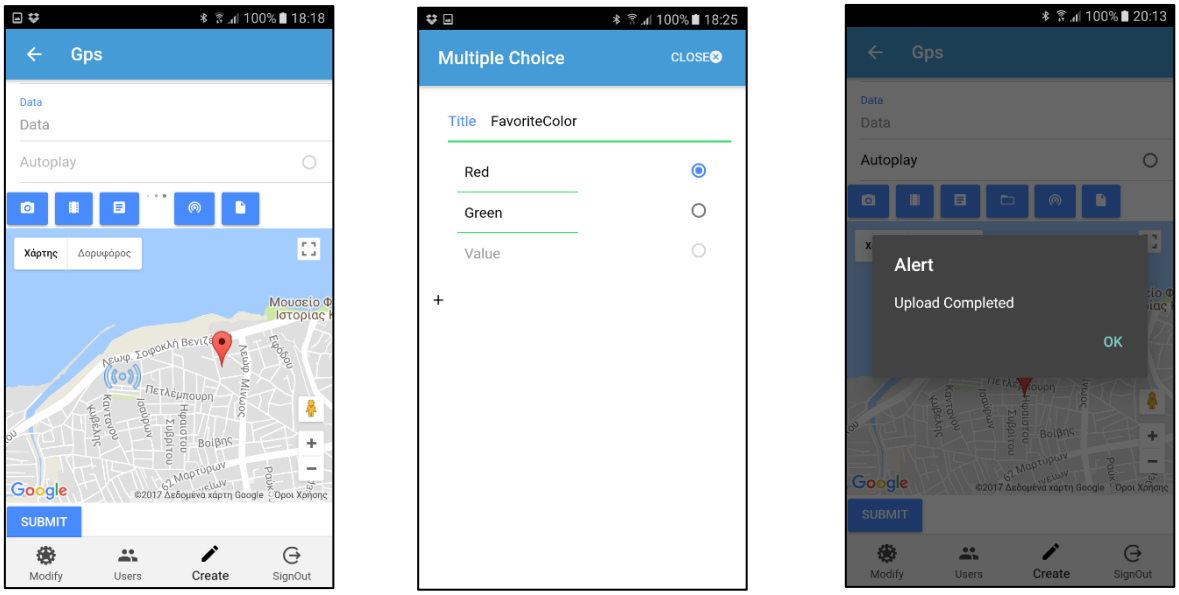


Figure *13*: Spinner, δημιουργία ερωτημάτων πολλαπλής επιλογής και upload

Για την εισαγωγή αρχείων υπάρχει η επιλογή ο admin να διαλέξει αρχεία τύπου εικόνας, video ή ήχου από το σύστημα αρχείων ή από την κάμερα. Στην περίπτωση εισαγωγής από την κάμερα υπάρχουν δύο κουμπιά με τα εικονίδια της φωτογραφίας και του video αντίστοιχα. Για την εισαγωγή αρχείου από το σύστημα αρχείων υπάρχει ένα κουμπί με το εικονίδιο του φακέλου. Για όλη την διάρκεια που χρειάζεται η εφαρμογή για να διαβάσει τα αρχεία υπάρχει ένας spinner στην θέση του κουμπιού. Στην περίπτωση που το αρχείο που επιλέχθηκε είναι ένα ηχητικό κομμάτι το πεδίο Autoplay ενεργοποιείται στην φόρμα. Για την εισαγωγή ερωτήσεων πολλαπλής επιλογής υπάρχει ένα κουμπί το τρίτο από τα δεξιά. Με το πάτημα του κουμπιού μια νέα οθόνη εμφανίζεται με μια λίστα από τρία text πεδία με radio button για την επισήμανση της σωστής απάντησης, ένα field για τον τίτλο και ένα κουμπί με το εικονίδιο συν. Για την εισαγωγή ακόμα μιας ερώτησης. Όλα τα αρχεία ανεβαίνουν στο cloud με το πάτημα του κουμπιού submit. Στην περίπτωση που κάποιο από τα αρχεία δεν είναι επιτρεπτού τύπου κατάλληλο μήνυμα εμφανίζεται στην οθόνη. Επιτρεπτοί τύποι αρχείων είναι τα mp4, mp3 και jpeg.

# Κεφάλαιο 3

## 3.1 Firebase οδηγός

Σε αυτόν τον οδηγό θα δείξω την χρήση της firebase βάσης δεδομένων, πιστοποίησης, storage σε μια εφαρμογή ionic και το setup ενός firebase project. Μετά την δημιουργία ενός ionic project το μόνο που χρειάζεται είναι η αντιγραφή του κώδικα πιστοποίησης που παρέχει η firebase και επικόλληση του στο αρχείο app.module.ts του ionic project. Όπως και την αρχικοποίησή του. Για την αρχικοποίηση μπορεί να γίνει χρήση του Cordova plugin ή του name space και να το εισάγει στα imports. Για την υπηρεσία της πιστοποίησης μπορεί να γίνει χρήση του ίδιου plugin και να προστεθεί το AngularFireAuthModule στα imports.

```
var config = {
    apiKey: "<API_KEY>",
    authDomain: "<PROJECT_ID>.firebaseapp.com",
    databaseURL: "https://<DATABASE_NAME>.firebaseio.com",
    storageBucket: "<BUCKET>.appspot.com",
    messagingSenderId: "<SENDER_ID>",
  };
```

Η πιστοποίηση είναι δυνατή με την χρήση των λογαριασμών Google, Twitter, GitHub, Facebook μέσω ανακατεύθυνσης στον πάροχο του λογαριασμού. Επίσης είναι δυνατή η πιστοποίηση σύνδεσης μέσω του τηλεφωνικού αριθμού, κωδικού και ανώνυμα. Η διαχείριση των χρηστών είναι δυνατή μέσω της firebase και δίνει την δυνατότητα αναβαθμίσεις των πληροφοριών του προφίλ χρήστη, επαναφοράς του κωδικού χρήστη μέσω email και διαγραφής ενός χρήστη. Μια κονσόλα για την πιστοποίηση των χρηστών υπάρχει στη σελίδα της firebase όπου κάποιος μπορεί να δει όλους τους χρήστες να ενεργοποιήσει ή να απενεργοποιήσει μεθόδους σύνδεσης, template για πιστοποίηση email και να κάνει χρήση της πιστοποίησης μέσω τηλεφωνικού αριθμού. Επιπροσθέτως υπάρχει υπηρεσία που ελέγχει εάν ο χρήστης είναι συνδεδεμένος ή όχι.

```
this.afAuth.auth.createUserWithEmailAndPassword(Email,Password).
then(()=>{})
```

Η firebase βάση δεδομένων μπορεί να προσπελασθεί μέσα από αντικείμενα και λίστες. Και οι δύο αυτές μέθοδοι απαιτούν ένα instance πριν το διάβασμα ή γράψιμο δεδομένων. Οι λίστες είναι χρήσιμές γιατί τα δεδομένα αποθηκεύονται κάτω από ένα μοναδικό κλειδί στην βάση δεδομένων και έχουν τη δυνατότητα προσαρτήσεις δεδομένων στα ήδη υπάρχοντα με την μέθοδο push. Τα αντικείμενα κάνουν χρήση της μεθόδου set για να εισάγουν δεδομένα. Η συγκεκριμένη μέθοδος είναι καταστροφική γιατί διαγράφει οτιδήποτε άλλο υπάρχει κάτω από το ορισμένο μονοπάτι. Για περιπτώσεις αλλαγής δεδομένων γίνεται χρήση της μεθόδου update η οποία υπάρχει και στις λίστες. Η μέθοδος remove διαγράφει όλα τα δεδομένα στο υπάρχων αντικείμενο ή λίστα.

Το ανέβασμα αρχείων στο storage είναι δυνατό μέσω bytearray, base64, base64url, dataURL και blob. Αρχικά πρέπει να δημιουργηθεί ένα reference στα δεδομένα που θα ανεβούν. Η διαχείριση του upload είναι δυνατή μέσα από τις μεθόδους pause, resume και cancel. Το κατέβασμα είναι δυνατό μέσω του URL που μπορεί να ανακτηθεί από την μέθοδο getDownloadURL. Ένα άλλο χαρακτηριστικό είναι η προθήκη metadata στο αρχείο.

```
varnewPostKey = firebase.database().ref().child('posts').push().key;
varstorageRef = firebase.storage().ref();
varmountainsRef = storageRef.child('mountains.jpg');
varmountainImagesRef = storageRef.child('images/mountains.jpg');
```

Η ανάκτηση αρχείων είναι δυνατή μέσω data snapshots. Τα δεδομένα γίνονται unwrap εξορισμού αλλά υπάρχει η επιλογή preserveSnapshot για την λήψη του snapshot όπως είναι. Επίσης υπάρχει μέθοδος για ανάκτηση των δεδομένων μία φορά με την μέθοδο once('value') ή τις μεθόδου .on('value') που ενεργοποιείται κάθε φορά που γίνεται κάποια αλλαγή στο συγκεκριμένο μονοπάτι το οποίο έχει ορισθεί. Τα snapshots έχουν την δυνατότητα εκτέλεσης τις μεθόδου forEach() για την προσπέλαση του βρόγχου και τις .key και .val() ιδιότητες. Η ιδιότητα key θεωρείτε το τελευταίο μέρος του DataSnapshot μονοπατιού. Η μέθοδος .val() επιστρέφει όλα τα υπόλοιπα δεδομένα. Τα δεδομένα μπορεί να είναι ένα array, αντικείμενο ή τύπου scalar. Επίσης είναι δυνατή η προσπέλαση του data snapshot μέσω dot notation.

```
this.itemsRef=db.list('items');
this.itemsRef.snapshotChanges(['child_added'])
.subscribe(actions=> {
actions.forEach(action=> {
console.log(action.type);
console.log(action.key);
console.log(action.payload.val());
});
});
```

## 3.2 Ionic Οδηγός

Η εγκατάσταση του ionic απαιτεί την προ-εγκατάσταση των cordova, nodejs και NPM. Μετά την εγκατάσταση των προ-απαιτήσεων η εντολή *npm install -g ionic@latest* στο cmd εγκαθιστά το framework και η εντολή *ionic start myNewProject* ξεκινάει ένα νέο project. Το ionic έχει πολλά templates τα οποία μπορούν να χρησιμοποιηθούν σαν παραδείγματα. Το ionic τρέχει στον browser με την εντολή *ionic serve* και γίνεται build με την εντολή *ionic cordova run.*

Για το build μίας εφαρμογής σε συγκεκριμένη πλατφόρμα χρειάζεται η εγκατάσταση του Android tools ή του Android studio και ένα java sdk. Στην περίπτωση του Android tools απαιτείται η εγκατάσταση του Gradle ξεχωριστά. Τα builds σε Android τρέχουν μέσω του cordova σε

προσομοίωση ή σε φυσική συσκευή. Το Ionic τρέχει σωστά σε εκδόσεις Android 5.x και πάνω αλλά παρουσιάζει bugs για την έκδοση 4.4.x. Για build σε iOS, ένα mac υπολογιστής χρειάζεται.

Μέσα σε ένα Ionic project υπάρχουν αρκετοί υποφάκελοι. Στον φάκελο με όνομα src βρίσκονται όλα τα αρχεία uncompiled. Στο αρχείο src/app/module.ts βρίσκονται όλα τα access σημεία τις εφαρμογής. Κάτω από το @NgModule υπάρχουν όλες οι δηλώσεις σελίδων, τα import modules, οι πάροχοι των services και το bootstrap. Στο αρχείο src/app/component.ts ορίζονται το template, η root σελίδα, και το promise platform.Ready() το οποίο γίνεται resolve μόλις όλα τα plugin είναι διαθέσιμα. Στο src/app/main.ts αρχείο το AppModule γίνεται bootstrap. Μέσα σε κάθε .ts αρχείο υπάρχει ο @component decorator για την εισαγωγή metadata. Τα πιο συχνά ορίσματα είναι τα Selector το οποίο ορίζει το CSS που χρησιμοποιείται στην σελίδα και το templateUrl στο οποίο ορίζεται το html template.

```
@NgModule({
  declarations: [MyApp,HelloIonicPage, ItemDetailsPage, ListPage],
  imports: [ BrowserModule, IonicModule.forRoot(MyApp)],
  bootstrap: [IonicApp],
entryComponents: [MyApp,HelloIonicPage,ItemDetailsPage,ListPage],
  providers: []
})
export class AppModule {}
```

```
@Component({
  selector: 'page-hello-ionic',
templateUrl: 'hello-ionic.html'
})
```

Ακόμα ένα σημαντικό μέρος στο Ionic είναι το navigation. Το navigation δουλεύει σαν μια συνηθισμένη στοίβα οπού σελίδες γίνονται push στην κορυφή της με τη χρήση της ομώνυμης μεθόδου. Η μέθοδος pop() είναι αυτή που αφαιρεί μια σελίδα από την στοίβα. Υπάρχει η επιλογή να περαστούν δεδομένα σαν αντικείμενα μέσω τις μεθόδου push() και κατόπιν να ανακτηθούν στην σελίδα που έγινε push. Τόσο η push όσο και η pop μέθοδοι κάνουν χρήση της NavController κλάσης. Η ανάκτηση των δεδομένων είναι δυνατή μέσω του αντικειμένου NavParams και τις get μεθόδου του. Η συγκεκριμένη μέθοδος παίρνει σαν όρισμα το όνομα της παραμέτρου.

```
itemTapped(event,item){
this.navCtrl.push(ItemDetailsPage,{
item: item
    });
}

this.navParams.get('userParams');
```

Το Ionic έχει το δικό του html στυλ. Μπορεί να γίνει χρήση του Sass και του CSS για την εισαγωγή ενός καινούργιου στυλ. Το Sass δίνει την δυνατότητα να εισαχθεί σε όλη την εφαρμογή ή σε μία κλάση. Το Sass παρέχει επίσης την δυνατότητα στους developers να εισάγουν τις δικές τους μεταβλητές. Οι μεταβλητές μπορεί να είναι τύπου Numbers, String, Color, Boolean και μοιάζουν κάπως έτσι: $primary-color:#3bbfce. Επιπροσθέτως είναι δυνατόν ο developer να δημιουργήσει το δικό του layout με την χρήση του responsive grid. Το responsive grid αποτελείται από οποιονδήποτε αριθμό από γραμμές και στήλες. Το μέγεθος του grid είναι μεταβλητό. Εάν δεν δοθούν περεταίρω

μεταβλητές το grid καταλαμβάνει όλο το διαθέσιμο ύψος και πλάτος. Για κάθε πλατφόρμα υπάρχει συγκεκριμένο στυλ εικονιδίων τα οποία μπορούν να μεταβληθούν κατά τη βούληση του developer.

## 3. 3 Οδηγός για Developers

Στο υποκεφάλαιο αυτό. Θα εξηγήσω των κώδικα της εφαρμογής έτσι ώστε οποιοσδήποτε που ενδιαφέρεται στο να τις τροποποίηση να έχει διαθέσιμο έναν οδηγό. Κάθε .ts αρχείο αποτελείται από μια κλάση. Τα plugins του ionic εκτελούνται ασύγχρονα. Αυτός είναι ο λόγος που γίνεται χρήση promises. Για τις εφαρμογές του user και admin όλα τα δεδομένα γίνονται upload στην cloud βάση δεδομένων τις firebase ή στο storage.

Στο project κάνω χρήση μερικών συγκεκριμένων ονομάτων μεταβλητών τα οποία επαναλαμβάνονται αρκετά μέσα στο project. Αυτά είναι τα GamePath το οποίο αναφέρεται στο μονοπάτι του σεναρίου στην βάση δεδομένων. MarkerName αναφέρεται στο όνομα του σημείου ενδιαφέροντος, η μεταβλητή CordsArray στο snapshot που επιστράφηκε με όλα τα δεδομένα του σεναρίου και τις συντεταγμένες. GameName για το όνομα του σεναρίου. Επίσης υπάρχουν κάποιες συγκεκριμένες κλάσεις για εγγραφή και διάβασμα δεδομένων στην firebase. Η κλάση FinderPage χρησιμοποιείται για την αναζήτηση αρχείων από την βάση δεδομένων. FirePage για ανέβασμα δεδομένων και η ReplacementPage για αντικατάσταση.

Στην βάση δεδομένων της firebase κάνω χρήση μιας συγκεκριμένης δομής. Κάθε σενάριο έχει το δικό του μοναδικό κλειδί όπου από κάτω σαν παιδί υπάρχει το GameName το οποίο με την σειρά του έχει σαν παιδί το όνομα του σεναρίου. Κάτω από το όνομα του σεναρίου υπάρχουν όλα τα σημεία ενδιαφέροντος. Τα ονόματα των σημείων ενδιαφέροντος είναι μοναδικά σε κάθε σενάριο. Στο ίδιο επίπεδο με τα σημεία ενδιαφέροντος βρίσκονται ο κωδικός και ο τύπος του σεναρίου. Τα ονόματα των απλών σημείων ενδιαφέροντος είναι τύπου Marker συν ένας αριθμός. Για τα beacon σημεία ενδιαφέροντος το όνομα τους είναι το id τους. Στο ίδιο επίπεδο με το όνομα των σημείων ενδιαφέροντος βρίσκονται οι συντεταγμένες και ο αύξων αριθμός του. Τα δεδομένα των σημείων βρίσκονται κάτω από το όνομα Data. Ένα παιδί του Data είναι τα MultipleChoices το οποίο με την σειρά του έχει σαν παιδία όλα τα δεδομένα για τις ερωτήσεις πολλαπλής επιλογής.

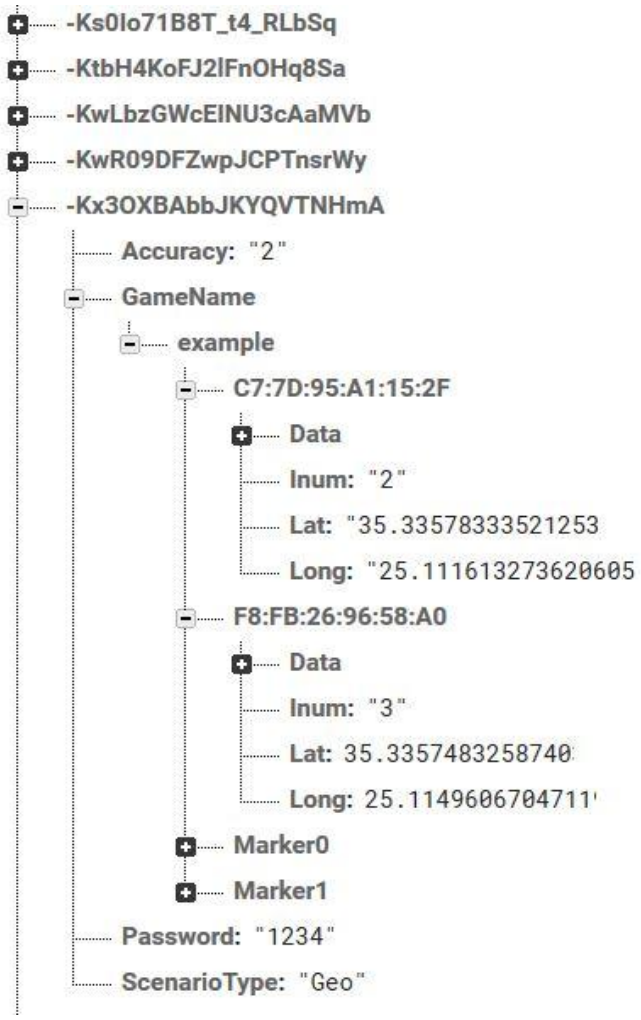


Figure *14*: Παράδειγμα δομής σημείων ενδιαφέροντος και σεναρίου στην Firebase

Επιπροσθέτως στην βάση δεδομένων βρίσκονται τα στοιχεία του χρήστη. Όλα τα στοιχεία του αποθηκεύονται κάτω από το μοναδικό id που δημιουργείται κατά την εγγραφή του. Κάτω από το id βρίσκονται το email του χρήστη και τα δεδομένα διαδρασης που έχει ανεβάσει ο χρήστης κάτω από ένα μοναδικό id που δημιουργείται με την χρήση της firebase list και τις push μεθόδου της. Κάτω από αυτό το μοναδικό id υπάρχει το όνομα του σημείου ενδιαφέροντος το οποίο έχει σαν παιδί του το μονοπάτι που βρίσκονται στο storage τα δεδομένα που ανέβασε ο χρήστης. Ακόμα υπάρχει ένα παιδί με όνομα MultipleChoices το οποίο έχει σαν παιδί του το πεδίο Answers όπου υπάρχουν οι απαντήσεις του χρήστη στο ερώτημα πολλαπλής επιλογής και το μονοπάτι που βρίσκονται οι πιθανές απαντήσεις. Κάτω από το Answers υπάρχει ένα πεδίο με όνομα τον τίτλο του ερωτήματος και την απάντηση του χρήστη σαν τιμή του. Για τον admin υπάρχει μόνο μία εγγραφή κάτω από το όνομα Admin. Η συγκεκριμένη εγγραφή έχει σαν παιδί της το μοναδικό id της πιστοποίησης και το email της εγγραφής στην εφαρμογή του admin.

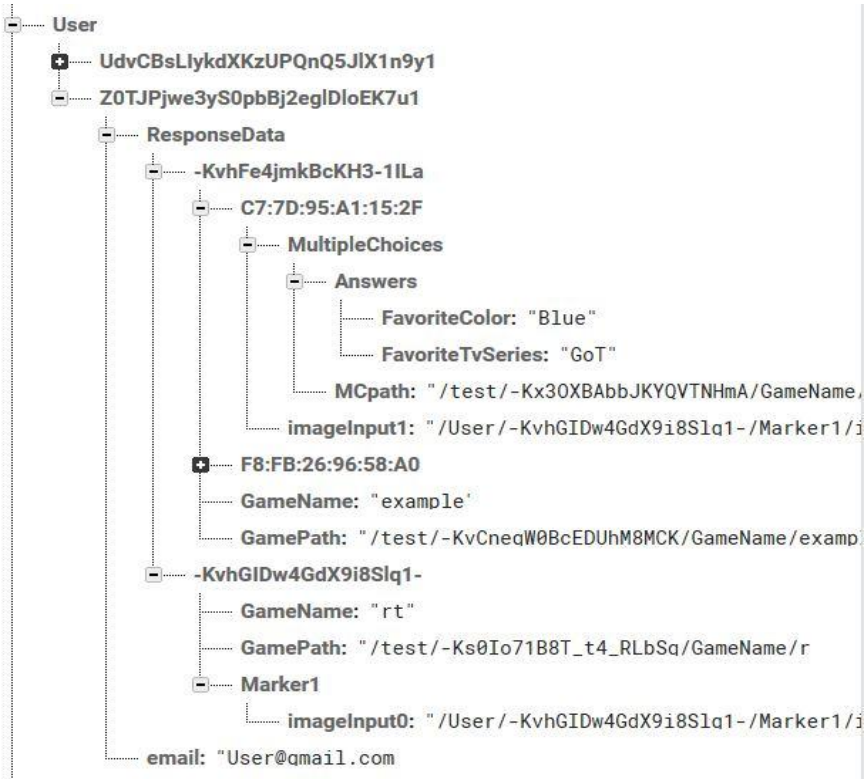


Figure*15: Παράδειγμα δομής δεδομένων χρήστη*

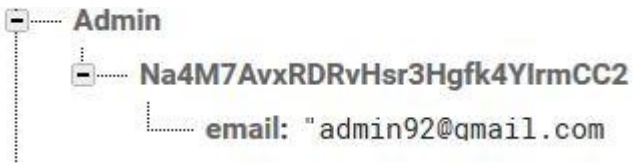


Figure16 : Παράδειγμα δομής δεδομένων admin

Τα observables χρησιμοποιούνται για την διαχείριση ασύγχρονων δεδομένων. Αυτό σημαίνει ότι τα δεδομένα μπορεί να εκτελεστούν χωρίς συγκεκριμένη σειρά στο χρόνο. Η angular κάνει χρήση μιας third party βιβλιοθήκης για την χρήση τους με όνομα RxJS. Τα observables χρησιμοποιούνται με το να γίνονται subscribe σε αυτά με την χρήση της μεθόδου subscribe(). Η μέθοδος παρέχει την χρήση τριών functions για την διαχείριση τους. Αυτές οι functions είναι η complete η οποία καλείται με την ολοκλήρωση του subscription. Η error για την διαχείριση των exception και η τρίτη για την διαχείριση των δεδομένων που έρχονται. Μία καλή τακτική είναι να εκχωρούνται τα δεδομένα σε μια εξωτερική ή public μεταβλητή για να είναι προσβάσιμα και εκτός του observable. Είναι σημαντικό το observable να γίνει unsubscribe όταν δεν χρειάζεται πλέον. Εκτός από την μέθοδο subscribe υπάρχει

και η map. Η συγκεκριμένη μέθοδος επιστρέφει ένα observable που μπορεί να γίνει με την σειρά του επίσης subscribe.

Ένα promise είναι το αποτέλεσμα μιας ασύγχρονης δραστηριότητας αλλά χειρίζεται μόνο ένα γεγονός και είτε ολοκληρώνεται ή αποτυγχάνει. Τα promises αντικαθιστούν τα callbacks γιατί σε μερικές περιπτώσεις όταν περιπλέκονται δίνουν αρκετά λάθη τα οποία είναι δύσκολο να χειριστή κάποιος. Τα promises έχουν τρείς καταστάσεις. Pending είναι η κατάσταση όπου το promise δεν έχει γίνει resolve ούτε έχει αποτύχει. Fullfilled όταν έχει γίνει επιτυχώς resolved και rejected όταν έχει αποτύχει. Υπάρχουν μέθοδοι για να γίνει ένα promise άμεσα reject ή resolve και να εκτελεσθούν όλα τα promises. Για την επιστροφή μιας promise γίνεται χρήση της μεθόδου then(). Επιπροσθέτως υπάρχει και η μέθοδος race η οποία κάνει resolve μια promise όταν κάποια άλλη γίνεται resolve ή reject. Οι κυριότερες διαφορές μεταξύ των observables και των promises είναι ότι τα observables μπορούν να ακυρωθούν, να χειριστούν πολλαπλά γεγονότα. Επίσης διατηρούν operators όπως map και foreach. Για το unsubscribe από τα observables χρησιμοποιώ την βιβλιοθήκη RxJS( Reactive-Extensions) και τους operators takeUntil και ngUnsubscribe

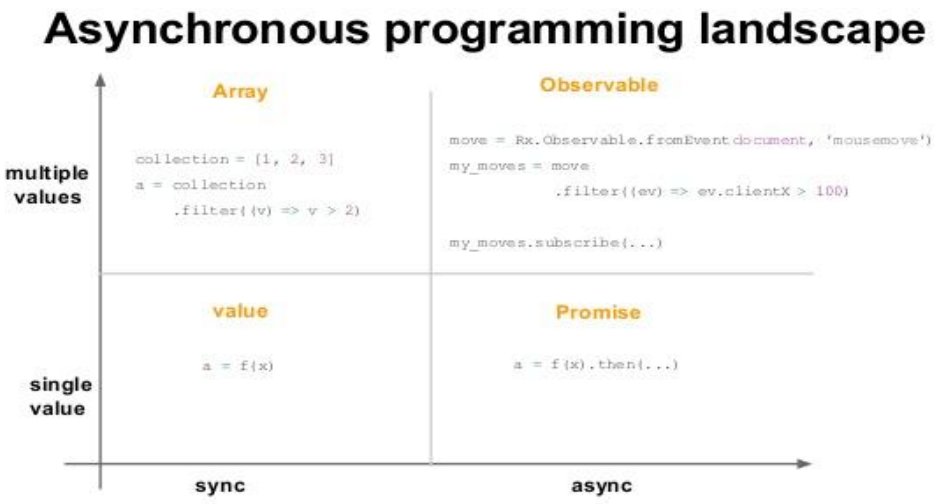


Figure 17 :Asynchronous programming landscape

### 3.3.1 Εφαρμογή Admin

Εγγραφή

Για την εγγραφή και το sign in υπάρχουν τρείς κλάσεις όλες κάνουν χρήση της βιβλιοθήκης angularfire2 και του navigation controller. Το όνομα της πρώτης από τις τρείς κλάσεις ονομάζεται CheckPage. Η συγκεκριμένη κλάση είναι η root σελίδα του project, κάνει χρήση του navigation controller για να πρόσθεση μία σελίδα στη στοίβα του navigation. Η πιστοποίηση του χρήστη για την εφαρμογή του admin γίνεται σε διαφορετικό firebase project από αυτό που χρησιμοποιείται για τον user. Έτσι ώστε να μην είναι δυνατόν να μπορεί κάποιος να συνδεθεί και στις δύο εφαρμογές με τα ίδια στοιχεία. Για όλα τα άλλα χαρακτηριστικά που προσφέρει η firebase όπως η βάση δεδομένων και το storage χρησιμοποιείται το ίδιο project και για τις δύο εφαρμογές. Το δεύτερο firebase project αρχικοποιείται στην κλάση Check σε αντίθεση με το πρώτο που αρχικοποιείται στο module.ts.

Η κλάση CheckPage ελέγχει για όλες τις αλλαγές κατάστασης όσο αναφορά την πιστοποίηση με την χρήση του onAuthStateChanged μεθόδου του auth service της firebase. Εάν ο χρήστης είναι ήδη συνδεδεμένος ή εγγεγραμμένος θα ανακατευθυνθεί στην κλάση TabsPage με το να γίνει η σελίδα push στην κορυφή της στοίβας. Σε αντίθετη περίπτωση ο admin θα ανακατευθυνθεί στην RegistrationPage κλάση. Για την εγγραφή γίνεται χρήση της μεθόδου εγγραφής χρήστη με email και κωδικό που παρέχει η firebase. Αυτή η μέθοδος δέχεται σαν ορίσματα το email και των κωδικό που δόθηκε από τον χρήστη. Με κάθε εγγραφή εισάγεται μια καινούρια εγγραφή στην βάση δεδομένων της firebase κάτω από το όνομα Admin με την χρήση της μεθόδου .set του firebase αντικειμένου. Όταν ο admin κάνει εγγραφή ή συνδεθεί ανακατευθύνεται από την κλάση Check παρότι ουσιαστικά βρίσκεται σε διαφορετική κλάση λόγω του ότι το observable του onAuthStateChanged τρέχει ακόμα.

Στην περίπτωση τις αποσύνδεσης η τρέχουσα view αφαιρείται από την στοίβα και το δεύτερο firebase project αρχικοποιείται ξανά αλλά αποτυγχάνει γιατί έχει ήδη αρχικοποιηθει. Για την αποφυγή αυτής της κατάστασης ένας έλεγχος πραγματοποιείται στο ngOnInit συνάρτηση. Ένα if statement ελέγχει εάν ο αριθμός των firebase εφαρμογών είναι μεγαλύτερος από 1. Εάν είναι, η δεύτερη διαγράφεται και επαναρχικοποιείται καλώντας την συνάρτηση SecondFirebase. Σε αντίθετη περίπτωση αρχικοποιείται με την κλήση της ίδιας συνάρτησης. Για κάθε σφάλμα εμφανίζεται κατάλληλο μήνυμα με την χρήση του alert controller του Ionic. Για την περίπτωση σύνδεσης χρήστη χρησιμοποιείται η ίδια μέθοδο με τα ίδια ορίσματα όπως στην εγγραφή. Για την αποσύνδεση η τρέχουσα view αφαιρείται από την στοίβα

```
ngOnInit(){

 console.log("Active",this.navCtrl.getActive().index);
if(firebase.apps.length>1){
            firebase.app("secondary").delete().then(()=>{
console.log("deleted");
this.SecondFirebase();
});
}
else{
this.SecondFirebase();
}
}
SecondFirebase(){


var config ={
apiKey:"AIzaSyCY-NlFrO9RY6QYbVPPDP-CdPWbvAxA91Q",
authDomain:"fapp-179c7.firebaseapp.com",
databaseURL:"https://fapp-179c7.firebaseio.com",
projectId:"fapp-179c7",
storageBucket:"fapp-179c7.appspot.com",
messagingSenderId:"866746169794"
};

var secondary=firebase.initializeApp(config,secondary");
varsecondaryDatabase=secondary.auth();
this.check(secondaryDatabase);
```

Code1: Εγγραφή Χρήστη

```
Register(){


this.secondaryDatabase.createUserWithEmailAndPassword(this.Rform.Email,thi
s.Rform.Password).then(()=>{

var user =this.secondaryDatabase.currentUser;

vardbPath="/Admin/";

newFirePage(this.af,this.loadingCtrl,this.alertCtrl).
AddUser(dbPath,user,this.token);

},(error)=>{

varerrorMessage=error.message;
    console.log("error",errorMessage);

this.AlreadyExist(errorMessage);
});
```

Code2: Αρχικοποίηση δεύτερης firebase

Τροποποίηση Σεναρίου

Μετά την εγγραφή η πρώτη οθόνη που βλέπει ο admin είναι μία σελίδα με τέσσερα κουμπιά στο κάτω μέρος της για τροποποιήσει, δημιουργία σεναρίου, τα δεδομένα του απλού χρήστη και ένα κουμπί για αποσύνδεση. Για την τροποποιήσει ενός υπάρχον σεναρίου ο admin εισάγει το όνομα του σεναρίου που θέλει να τροποποιήσει και πατάει το submit. Το όνομα που έδωσε ο admin γίνεται push εισάγεται σαν παράμετρος όταν η σελίδα γίνεται push στην κλάση HomePage .Εν συνεχεία γίνεται retrieve στην ngOnInit συνάρτηση τις κλάση. Αμέσως μετά καλείται η συνάρτηση CallToFind για να βρει τα σημεία ενδιαφέροντος του σεναρίου. Για να γίνει αυτό καλούνται επιπλέον τρείς συναρτήσεις οι οποίες βρίσκονται στην κλάση FinderPage κάθε μια από αυτές είναι εμφωλευμένη στην προηγούμενη και κάνει χρήση subscriptions. Στην FinderPage κλάση οι κληθέντες συναρτήσεις ανακτούν Data snapshots από την firebase.

Πιο αναλυτικά η πρώτη συνάρτηση με όνομα RetrieveKey δέχεται σαν όρισμα το μονοπάτι όπου βρίσκονται όλα τα σενάρια στην βάση δεδομένων της firebase. Η συνάρτηση επιστρέφει ένα array με όνομα keys το οποίο περιέχει όλα τα κλειδιά τα οποία εξήγαγε από το snapshot και αντιστοιχούν στο μοναδικά id των σεναρίων. Μέσα στο subscription της συνάρτησης μια for επανάληψη διατρέχει όλο το array με τα κλειδιά και για κάθε στοιχείο καλείται η συνάρτηση RetrievePath. Σαν ορίσματα δέχεται το ίδιο μονοπάτι το κλειδί και το όνομα του σεναρίου. Στην FinderPage ανακτάται το snapshot από το νέο μονοπάτι που αντιστοιχεί στα δεδομένα ενός σεναρίου. Ένας βρόγχος for διατρέχει το snapshot και ελέγχει εάν το δοθέν όνομα είναι το ίδιο με το όνομα του εκάστοτε σεναρίου. Εάν ναι το μονοπάτι του δημιουργείται και επιστρέφεται στην αρχική συνάρτηση. Σε αντίθετη περίπτωση επιστρέφεται undefined και προχωράει στο επόμενο κλειδί. Η τρίτη συνάρτηση με όνομα RetrivePosition δέχεται σαν όρισμα το μονοπάτι που βρίσκεται το σενάριο που ζήτησε ο admin και επιστρέφει το snapshot που αντιστοιχεί στα σημεία ενδιαφέροντος του σεναρίου.

```
new FinderPage(this.af).RetrieveKey(path).takeUntil(this.ngUnsubscribe).
subscribe(keys =>
             {

                              for(let key of keys){

newFinderPage(this.af).RetrievePath(path,key,this.GameName).
takeUntil(this.ngUnsubscribe).subscribe(GPath=>{


                     if(GPath!= undefined){
                              this.key=key;
                                     this.GamePath=GPath+"/GameName/"+this.GameName;


newFinderPage(this.af).RetrivePosition(this.GamePath).
takeUntil(this.ngUnsubscribe).subscribe(cordinates=>{

                              this.CordsArray=cordinates;
                              this.LatlngOnMap(this.CordsArray);
```

Code 3: Subscriptions

```
RetrieveKey(path){


this.lst=this.af.list(path,{preserveSnapshot:true});

returnthis.lst.map(snapshots =>{

snapshots.forEach(snapshot =>{

this.keys.push(snapshot.key);


});
returnthis.keys;

})
              }
```

Code 4: Συνάρτηση Retrieve Key

Στην τρίτη συνάρτηση καλείται μια  ακόμα με όνομα LatlngOnMap. Αρχικά η συγκεκριμένη συνάρτηση κάνει unsubscribe τις προηγούμενες και ελέγχει με την χρήση ενός if statement εάν τα σημεία ενδιαφέροντος αντιστοιχούν σε beacon ή όχι. Για αμφότερες τις δύο περιπτώσεις τοποθετεί σε ένα object ως στοιχεία του τις συντεταγμένες και το όνομα του σημείου ενδιαφέροντος. Εν συνεχεία τοποθετεί τα αντικείμενα σε ένα array με όνομα onMap εάν είναι τύπου marker και BeaconPoi εάν είναι τύπου beacon. Στο τέλος της συνάρτησης καλείται μια νέα συνάρτηση με όνομα loadMap και σαν ορίσματα τα object arrays με τα σημεία ενδιαφέροντος.

Η συνάρτηση loadMap εισάγει στην εφαρμογή έναν χάρτη και τους markers του με την χρήση του ionic native plugin για τα google maps. Αρχικά στο αντικείμενο για τις ρυθμίσεις του χάρτη με όνομα mapOptions ορίζονται οι συντεταγμένες που θα είναι το κέντρο του χάρτη, το zoom και τον τύπο του χάρτη. Το zoom και οι συντεταγμένες για το κέντρο απαιτούνται. *Στη συνέχεια το αντικείμενο του χάρτη δέχεται σαν όρισμα το element του χάρτη από το html και το αντικείμενο mapOptions*. Στη συνέχεια ένας βρόγχος for διατρέχει το array με τα markers για τα beacon σημεία και τα τοποθετεί στον χάρτη ορίζοντας την τοποθεσία, τον χάρτη και το εικονίδιο. Για την περίπτωση των απλών markers γίνεται χρήση  τις Javascript Array prototype.map μεθόδου λόγω της χρήσης clusterer για την ομαδοποίηση τους. Μέσα σε αυτή την επανάληψη ορίζεται μόνο η θέση του marker πάνω στον χάρτη. Στη συνέχεια προστίθεται ο MarkerClusterer ο οποίος σαν ορίσματα δέχεται το array με τους markers, τον χάρτη που θα τοποθετηθούν και το μονοπάτι που βρίσκονται οι εικόνες για την ομαδοποίηση. Επίσης προστίθεται ένας listener  σε όλους τους απλούς markers  με την μέθοδο κλικ η οποία καλεί την συνάρτηση FindData για την ανάκτηση των δεδομένων του σημείου ενδιαφέροντος.

```
loadMap(onMap,BeaconPoi){


varBimg="https://png.icons8.com/online-filled/ios7/32";

      let options ={timeout:10000,enableHighAccuracy:true};


  let mapOptions={
center:onMap[0].latlng,
      zoom:15,
mapTypeId:google.maps.MapTypeId.ROADMAP,

}

this.map=newgoogle.maps.Map(this.mapElement.nativeElement,mapOptions);


for(let Bpoi of BeaconPoi){
var marker=newgoogle.maps.Marker({});

marker.setPosition(Bpoi.latlng);
marker.setMap(this.map);
marker.setIcon(Bimg);
}

var map=this.map;
             let markers =onMap.map((visit,i)=>{
returnnewgoogle.maps.Marker({
position:visit.latlng,
label:i.toString()
```

Code 5: load map

```
varmarkerCluster=newMarkerClusterer(this.map, markers,
{imagePath:'https://developers.google.com/maps/documentation/javascript/examples/markerclusterer/m'});

vari=0;
for(let visit of onMap){

       markers[i].addListener('click',()=>this.FindData(visit.visited));
i++;
}


}
```

Code 6: Marker clusterer και event listener

Όλα τα παραπάνω συμβαίνουν κατά την αρχικοποίηση της σελίδας όλες οι υπόλοιπες συναρτήσεις καλούνται μετά από κάποιο click event. Η FindData συνάρτηση δέχεται σαν όρισμα το όνομα του σημείου ενδιαφέροντος. Μέσα στην συνάρτηση μία επανάληψη for διατρέχει το array CordsArray.Μέσα στην επανάληψη ένα if statement ελέγχει εάν το όνομα του σημείου ενδιαφέροντος του οποίου τα δεδομένα θα ανακτηθούν, είναι ίδιο με το όνομα που βρίσκεται στο array. Εάν ναι, τα δεδομένα του εκχωρούνται σε μία μεταβλητή ονόματος data και ορίζονται σαν όρισμα στην συνάρτηση DisplayFile. Σε αυτήν την συνάρτηση ένα for βρόγχος διατρέχει το data object που δέχεται σαν όρισμα. Με την χρήση της substring μεθόδου εξάγεται το πρώτο γράμμα από το κλειδί του object και ένα if statement ελέγχει ένα είναι ίδιο με το πρώτο γράμμα του εκάστοτε δεδομένου. ‘A’ για Audio, ‘T’ για text, ‘M’ για Multiples που αντιστοιχεί στα ερωτήματα πολλαπλής επιλογής,

‘G’ για Gform που αντιστοιχεί στα δεδομένα φόρμας, ‘v’ για video και ‘i’ για image. Εάν ναι γίνεται κλήση της συνάρτησης GetData ή GetVidData στην περίπτωση video. Και οι δύο συναρτήσεις βρίσκονται στην κλάση FinderPage.

Αμφότερες οι συναρτήσεις κάνουν χρήση promises, καλούν την μέθοδο getDownloadURL για την ανάκτηση του URL από το storage της firebase και επιστρέφουν ένα object. Σαν ορίσματα δέχονται το κλειδί και το object array με τα δεδομένα. Η GetData χρησιμοποιείται για εικόνα, ήχο και ή GetVidData για video. Στην GetData συνάρτηση έλεγχεται εάν το όνομα του αρχείου στην βάση δεδομένων της firebase εμπεριέχει τη λέξη Aplay (συντομογραφία του autoplay). Εάν ναι η μεταβλητή Autoplay δέχεται την τιμή true. Σε αντίθετη περίπτωση δέχεται την τιμή false. Η Boolean αυτή τιμή χρησιμεύη στα αρχεία ήχου. Το αντικείμενο που επιστρέφεται έχει σαν στοιχεία το download URL, την τιμή autoplay και το όνομα του αρχείου στην βάση δεδομένων. Η GetVidData συνάρτηση δεν ελέγχει για το autoplay και κάνει μία επιπλέον κλήση για να βρει το download URL του thumbnail του video το οποίο επιστρέφεται σαν στοιχείο του object αντί της Boolean τιμής. Στην περίπτωση του κειμένου απλά εκχωρείται σε μία μεταβλητή.

```
if(key.substr(0,1)=="A"){



new FinderPage(this.af).GetData(data,key).
then(audio =>{

this.Sounds.push(audio);


})
}

if(key.substr(0,1)=="T"){

          console.log("Text",data[key]);
this.txt=data[key];
}
```

Code 7: Ανάκτηση ήχου και κειμένου

Η αντικατάσταση ήχου και κειμένου γίνεται στην ίδια κλάση. Εικόνα και video αντικαθίστανται σε διαφορετική. Στην περίπτωση του ήχου για την επιλογή ενός νέου ηχητικού κομματιού γίνεται χρήση τριών cordova plugins. Όλα χρησιμοποιούν promises. Τα plugin αυτά είναι τα Filechooser για την χρήση ενός menu για πρόσβαση στο filesystem της συσκευής και την επιλογή του αρχείου. Το plugin επιστρέφει το URI του. Filepath για την μετατροπή του native URI σε content (ένα Content URl έχει την παρακάτω μορφή file:///android_asset/www/audio/ring.mp3). Το τρίτο plugin ονομάζεται File plugin. Το συγκεκριμένο μετατρέπει και επιστρέφει το αρχείο σε data URL. Πριν τη χρήση του ένα if statement ελέγχει εάν το αρχείο είναι τύπου mp3. Εάν ναι, το File plugin καλείται η συνάρτηση dataUrl η οποία βρίσκεται σε διαφορετική κλάση με όνομα CamPage. Δέχεται σαν όρισμα το μονοπάτι του φακέλου που βρίσκεται το αρχείο, το όνομα του αρχείου και τον τύπο του. Εάν το αρχείο είναι οποιοδήποτε άλλου τύπου εμφανίζεται στην οθόνη κατάλληλο μήνυμα λάθους. Για την εξαγωγή του ονόματος του φακέλου και του αρχείου γίνεται χρήση των μεθόδων substring και lastIndexOf. Για την εξαγωγή τους καλείται η συνάρτηση CheckType στην κλάση StringManipulationPage.

Πιο αναλυτικά η μέθοδος lastIndexOf βρίσκει την θέση της τελευταίας εμφανίσεις της τελείας και του slash χαρακτήρα. Κάνει substring το αρχικό μονοπάτι από αυτές τις δύο θέσεις και εκχωρεί το αποτέλεσμα που αντιστοιχεί στο όνομα του αρχείου σε νέα μεταβλητή. Για τον τύπο του αρχείου γίνεται substring το μήκος του μονοπατιού έως την τελευταία εμφανίσει της τέλειας στο μονοπάτι και εκχωρείται σε νέα μεταβλητή. Για το μονοπάτι του φακέλου αντικαθιστά από το υπάρχων μονοπάτι το όνομα και τον τύπο του με κενά string. Η συνάρτηση επιστρέφει ένα αντικείμενο με τον τύπο, το όνομα και το όνομα του φακέλου σαν στοιχεία του στην καλούσα συνάρτηση.

```
     Up(name,Autoplay){

this.ngUnsubscribe.next();
this.ngUnsubscribe.complete();

this.fileChooser.open()
.then(uri=>{ console.log(uri);


this.filePath.resolveNativePath(uri).then((filePath)=>{

varDataInfo=newStringManipulationPage().CheckType(filePath);

if(DataInfo.Type=="mp3"){

newCamPage(this.file,this.mediaCapture).
dataUrl(DataInfo.DirPath,DataInfo.Name+"."+DataInfo.Type).
then(FileEntry=>
{
this.ReplaceSound=FileEntry;
})

}else{
var subtitle="Insert an .mp3 sound file"
this.DataType(subtitle);
}

})
```

*Code8: Άνοιγμα συστήματος αρχείων*

Για το ανέβασμα του ηχητικού κομματιού καλείται η συνάρτηση RSound με το πάτημα του κουμπιού submit. Η συγκεκριμένη συνάρτηση δέχεται σαν όρισμα το όνομα που εξήχθη από την βάση δεδομένων και την Autoplay Boolean τιμή που προέρχεται από την επιλογή του admin στην φόρμα κατά την δημιουργία του σεναρίου. Η συνάρτηση ελέγχει με την χρήση ενός if statement εάν η τιμή Autoplay είναι true. Ένα δεύτερο if statement εμφωλευμένο στο πρώτο ελέγχει εάν το όνομα περιέχει την τιμή Aplay με την χρήση της μεθόδου includes. Εάν το αποτέλεσμα της μεθόδου είναι false σημαίνει ότι ο admin θέλει να προσθέσει την ιδιότητα αυτή και η λέξη προστίθεται στο όνομα. Εάν το Autoplay είναι false και το Aplay υπάρχει στο όνομα, αφαιρείται από αυτό. Για την αντικατάσταση του αρχείου στην βάση δεδομένων γίνεται κλήση της συνάρτησης με όνομα ChaneOnlyPlayOption στην κλάση ReplacementPage. Σαν ορίσματα δέχεται το όνομα το οποίο έχει

μετασχηματισθεί, το αρχικό όνομα, το μονοπάτι του σεναρίου, το όνομα του σημείου ενδιαφέροντος, το μοναδικό κλειδί του σεναρίου και το ηχητικό κομμάτι.

Αρχικά στην συνάρτηση διαμορφώνονται το StoragePath και το GamePath. Το reference ενός firebase αντικείμενου ορίζεται στο μονοπάτι που βρίσκεται το αρχείο που θα αντικατασταθεί. Ένα if statement ελέγχει εάν το αρχείο ήχου είναι undefined. Εάν ναι, σημαίνει ότι ο admin θέλει απλά να αλλάξει την τιμή του Autoplay χωρίς να αντικαταστήσει το αρχείο. Σε αντίθετη περίπτωση το καινούργιο αρχείο γίνεται upload στην βάση δεδομένων και μία νέα εγγραφή προστίθεται σε αυτήν. Για το upload γίνεται χρήση της συνάρτησης StoreInputData σαν ορίσματα δέχεται το StoragePath,GamePath, Angular Fire Database injection με όνομα af, το όνομα που θα αντικατασταθεί, το task που χρησιμοποιεί το storage reference στο μονοπάτι και στην μέθοδο putString, και τον loader ο οποίος θα εμφανιστεί στην οθόνη για όσο χρόνο χρειάζεται για να ανεβεί το αρχείο. Η μέθοδος task.on γίνεται wrap σε ένα promise και ακούει για τυχόν αλλαγή κατάστασης, λάθη και την ολοκληρώσει του. Μόλις ολοκληρωθεί το αρχείο προστίθεται στη βάση δεδομένων.

```
RSound(name,Autoplay){

this.InitilaName=name;

if(Autoplay==true){
if(name.includes("Aplay")==false)
                name="Aplay"+name;


}else
if(name.includes("Aplay")==true)
                 name =name.substring(5,name.length);

newReplacementPage(this.af,this.loadingCtrl).
ChaneOnlyPlayOption(name,this.GamePath,this.MarkerName,this.key,
this.InitilaName,this.ReplaceSound);

this.ReplaceSound=undefined;
this.CallToFind();
this.FindData(this.MarkerName);

}
```

Code 9: Προσθήκη και αφαίρεση autoplay

```
dataUrl(DirPath,Name){



returnthis.file.readAsDataURL(DirPath,Name).
then(function(FileEntry){
returnFileEntry;

},function(error){
console.log("FileEntryerror",error);
});
```

Code10: συνάρτηση dataurl

```
storeInputData(StoragePath,GamePath,af,name,task,loader){


returnnewPromise(function(resolve,reject){
loader.present();


task.on('state_changed',
function progress(snapshot){

this.percentage=(task.snapshot.bytesTransferred/task.snapshot.totalBytes)*100;
console.log(this.percentage,"% of completed ");

},

function error(err){
console.log("error is:",err);
},
functioncomplete(){
if(name.includes("Audio")==true){
this.obj=af.object(GamePath+"/Data"+"/"+name);
this.obj.set(StoragePath+"/"+name);
}
loader.dismiss()

});
});
```

Code 11: Αποθήκευση δεδομένων στο firebase storage

Στην περίπτωση αντικατάστασης εικόνας/video όταν ο admin πατήσει στο thumbnail καλείται η συνάρτηση ClickPhoto με παραμέτρους το URL τις εικόνας, τον τύπο του αρχείου εικόνα/video και το όνομα του αρχείου. Αρχικά ένα νέο modal δημιουργείται και εμφανίζεται στην οθόνη με το αρχείο που θα αντικατασταθεί. Modal είναι ένα content panel που εμφανίζεται πάνω από την εκάστοτε οθόνη. Κατά την δημιουργία του μερικές παράμετροι μεταφέρονται στη σελίδα του modal με τον ίδιο τρόπο όπως και στην μέθοδο push του navigation. Αυτές οι παράμετροι είναι το όνομα του marker, το μοναδικό κλειδί του σεναρίου, το download URL, ο τύπος, το όνομα του αρχείου και το μονοπάτι του σεναρίου. Ολα αυτά ανακτώνται στην κλάση ModalPopPage. Ένα if statement ελέγχει εάν το αρχείο είναι εικόνα ή video. Και στις δύο περίπτωσης ανακτάται το αρχείο που πρόκειται να αντικατασταθεί. Για την αντικατάσταση του αρχείου γίνεται χρήση μια παρόμοιας συνάρτησης με αυτήν για τον ήχο. Η μόνη διαφορά τους είναι ότι για τα video αρχεία δημιουργείται ένα thumbnail.

Για την δημιουργία του thumbnail καλείται η συνάρτηση MakeVideoThumnail όπου κάνει χρήση της μεθόδου createThumbnail του videoEditor cordova plugin. Η συνάρτηση δέχεται σαν όρισμα το native URI του αρχείου, το όνομα του και επιστρέφει ένα thumbnail. Στην συνέχεια καλείται η συνάρτηση AddPlayIcon η οποία δέχεται σαν παράμετρο το thumbnail και προσθέτει το εικονίδιο του play με την χρήση του HTML5 καμβά. Πιο αναλυτικά η συνάρτηση επιστρέφει μια promise όπου μέσα σε αυτήν δημιουργείται ένα html element με όνομα canvas. Ορίζονται οι διαστάσεις του (width, height) και δημιουργείται ένας δισδιάστατου περιεχομένου καμβάς. Δύο

image αντικείμενα ορίζονται. Στο πρώτο εκχωρείται το thumbnail και στο δεύτερο το play εικονίδιο που θα τοποθετηθεί επάνω στο thumbnail. Στον onload event handler η μέθοδος drawImage σχεδιάζει μία εικόνα με τις x και y διαστάσεις να είναι ίδιες με του καμβά. Ο onload event handler τις δεύτερης εικόνας είναι εμφωλευμένος στην πρώτη. Μία εικόνα σχεδιάζεται με διαστάσεις x,y ίδιες με αυτές του εικονιδίου τοποθετημένη στο κέντρο της προηγούμενης εικόνας. Αμέσως μετά ο καμβάς μετατρέπεται σε data URL και γίνεται resolve στην αρχική συνάρτηση. Όταν το modal κλείσει εάν έχει συμβεί κάποια αντικατάσταση τα δεδομένα στην οθόνη γίνονται reload.

```
AddPlayIcon(thumbnail){

returnnewPromise(resolve =>{

varc=(<HTMLCanvasElement>document.createElement("canvas"));

c.width=128;
c.height=128;
varctx=c.getContext("2d");
var imageObj2 =newImage();
var imageObj1 =newImage();
          imageObj1.src = thumbnail;
    imageObj2.src ="file:///android_asset/www/icon.png";
 imageObj1.onload =function(){
ctx.drawImage(imageObj1,0,0,128,128);
console.log("image onload");
   imageObj2.onload =function(){
ctx.drawImage(imageObj2,39,39,50,50,);
var image =c.toDataURL("image/jpeg");
resolve(image);
```

Code12: Συνάρτηση AddPlayIcaon

Στην περίπτωση αντικατάστασης των ερωτήσεων πολλαπλής επιλογής η συνάρτηση Response καλείται και ResponsePage γίνεται push στην στοίβα. Σαν δεδομένα παίρνονται το όνομα του marker και το GamePath. Στην ngOnInit συνάρτηση της κλάσης εκτός από τα δεδομένα που ανακτώνται δημιουργείται το μονοπάτι όπου βρίσκονται τα δεδομένα των ερωτήσεων. Στη συνέχεια καλείται η Mchoices συνάρτηση η οποία δέχεται σαν ορίσματα το μονοπάτι που δημιουργήθηκε και δύο κενά arrays. Μέσα στην συνάρτηση καλείται η RetrieveKey συνάρτηση. Η RetrieveKey επιστρέφει ένα array με τους τίτλους των ερωτήσεων. Μία επανάληψη for διατρέχει το εν λόγω array. Μέσα στο σώμα της επανάληψης σε ένα αντικείμενο ορίζονται σαν στοιχεία ο τίτλος και ένα πεδίο Answer σαν κενό stirng. Το αντικείμενο εκχωρείται σε ένα array με όνομα MultipleChoicesAnswers.

Μία δεύτερη συνάρτηση με όνομα RetrivePosition η οποία βρίσκεται στην FinderPage κλάση και είναι εμφωλευμένοι στην πρώτη καλείται. Σαν όρισμα δέχεται το μονοπάτι όπου είναι οι απαντήσεις στην βάση δεδομένων και τις επιστρέφει σε ένα array. Ένας for βρόγχος διατρέχει το array. Μέσα στο σώμα της επανάληψης ένα if statement ελέγχει εάν το key της απάντησης δεν είναι η σωστή απάντηση. Εάν δεν είναι, η απάντηση εκχωρείται σε ένα array ονόματος answers και σε ένα array με όνομα newAnswers εκχωρείται η τιμή null. Σε αντίθετη περίπτωση η σωστή απάντηση εκχωρείται στην μεταβλητή CorrectAnswer. Εκτός της for επανάληψης δύο αντικείμενα δημιουργούνται. Το ένα δέχεται σαν στοιχεία τον τίτλο, τις απαντήσεις και την σωστή απάντηση του ερωτήματος και στη συνέχει εκχωρείται σε ένα array ονόματος MultipleChoices. Το δεύτερο αντικείμενο δέχεται ορίσματα με το ίδιο όνομα με την διαφορά ότι όλα έχουν τιμές null. Επίσης το αντικείμενο εκχωρείται σε ένα array ονόματος newMultipleChoices. Όλα αυτά συμβαίνουν μέσα στο

σώμα τις πρώτης for. Ο λόγος υπάρξεις του δεύτερου array είναι ότι εκεί θα εκχωρηθούν οι νέες τιμές που θα δώσει ο admin και θα αντικαταστήσουν τις ήδη υπάρχουσες.

Για την αντικατάσταση των ερωτήσεων πολλαπλής επιλογής καλείται η συνάρτηση upload. Αρχικά όλα τα subscription γίνονται unsubscribe και καλείται η συνάρτηση FirebaseMultipleChoiceReplace η οποία βρίσκεται στη κλάση ReplacementPage. Σαν ορίσματα δέχεται τα δύο array ReplacementPage, newMultipleChoices και το μονοπάτι που βρίσκονται τα ερωτήματα με όνομα path. Μέσα στο σώμα της συνάρτησης το ερώτημα που θα αντικατασταθεί αφαιρείται ολόκληρο. Ένας for βρόγχος διατρέχει το newMultipleChoices array και ένα if statement ελέγχει εάν ο τίτλος του ερωτήματός είναι null η όχι. Εάν είναι null σημαίνει ότι ο admin δεν έχει δώσει κάποιο νέο τίτλο, εάν δεν είναι null σημαίνει το αντίθετο. Και για τις δύο περιπτώσεις καλείται η συνάρτηση ReplaceLoop. Σαν παραμέτρους δέχεται το μονοπάτι όπου θα εισαχθούν στην βάση δεδομένων. Τις τρέχουσες τιμές των newMultipleChoices, MultipleChoices και τον τρέχον τίτλο του MultipleChoices σε περίπτωση που ο τίτλος είναι null. Εάν δεν είναι null δέχεται τον τίτλο του newMultipleChoices.

Στην συνάρτηση ReplaceLoop ένα reference ενός firebase αντικείμενου ορίζεται στο μονοπάτι που θα εισαχθεί η σωστή απάντηση των ερωτημάτων πολλαπλής επιλογής. Ένα if statement ελέγχει εάν το στοιχείο CorrectAnswer του newMultipleChoices array είναι null. Εάν ναι, σημαίνει ότι ο admin δεν έχει δώσει κάποια σωστή απάντηση και η παλιά καταχωρείται στην βάση δεδομένων. Σε αντίθετη περίπτωση καταχωρείται η καινούρια. Στη συνέχεια ένας for βρόγχος κάνει τρείς επανάληψης. Μέσα στο σώμα του βρόγχου ο μετρητής αυξάνεται κατά ένα και εκχωρείται σε μία μεταβλητή με όνομα num.΄Ενα object ορίζεται στην θέση που θα εισαχθεί η απάντηση. Γίνεται έλεγχος με ένα if statement εάν ο admin έχει δώσει κάποια νέα απάντηση. Εάν ναι, εισάγεται η καινούρια απάντηση αλλιώς ξανά εισάγεται αυτή που υπήρχε πριν.

```
FirebaseMultipleChoiceReplace(MultipleChoices,newMultipleChoices,path){

this.obj=this.af.object(path);
this.obj.remove();

for(leti=0;i<newMultipleChoices.length;i++){

if(newMultipleChoices[i].Title==null){
this.ReplaceLoop(path,newMultipleChoices[i],MultipleChoices[i],
             MultipleChoices[i].Title);

}
else{



this.ReplaceLoop(path,newMultipleChoices[i],MultipleChoices[i],
             newMultipleChoices[i].Title);



}
```

*Code13: Συνάρτηση FirebaseMultipleChoices*

```
ReplaceLoop(path,newMultipleChoices,MultipleChoices,Title){


this.obj=this.af.object(path +"/"+Title+"/CorrectAnswer");

if(newMultipleChoices.CorrectAnswer==null)
this.obj.set(MultipleChoices.CorrectAnswer);
else
this.obj.set(newMultipleChoices.CorrectAnswer);


for(let k=0; k<3; k++){
varnum=k+1;


this.obj=this.af.object(path +"/"+Title+"/Answer"+num);

if(newMultipleChoices.Answers[k]==null)

this.obj.set(MultipleChoices.Answers[k]);
else

this.obj.set(newMultipleChoices.Answers[k]);

}
}
```

Code14: Συνάρτηση ReplaceLoop

Η συνάρτηση για την αντικατάσταση του κειμένου ονομάζεται text. Σαν παραμέτρους δέχεται το κείμενο που θα αντικαταστήσει το ήδη υπάρχων και το όνομα του υπάρχων κειμένου στην βάση δεδομένων. Αρχικά μέσα στην συνάρτηση όλα τα subscriptions γίνονται unsubscribe και η συνάρτηση ReplaceText στην κλάση ReplacementPage καλείται. Σαν ορίσματα δέχεται τις δύο παραμέτρους τις text συνάρτησης το GamePath, MarkerName και key. Μέσα στο σώμα της συνάρτησης δημιουργείται το μονοπάτι που θα εισαχθεί το κείμενο. Ένα reference ενός firebase αντικείμενου ορίζεται στο μονοπάτι που θα εισαχθεί το κείμενο και με την χρήση τις μεθόδου set αντικαθιστά το ήδη υπάρχων.

Η αντικατάσταση των συντεταγμένων των markers, των αυξόντων αριθμών και του τύπου του σεναρίου γίνεται από την κλάση PoiPage. Η κλάση PoiPage γίνεται push στην στοίβα. Σαν δεδομένα δέχεται τα δύο arrays με τα markers με όνομα onMap και BeaconPoi. Επίσης δέχεται τα GamePath, CordsArray και key. Ο χάρτης εμφανίζεται με την χρήση του ίδιου κώδικα όπως στην κλάση HomePage.Για την ανάκτηση του τύπου σεναρίου καλείται η συνάρτηση GetScenarioType με όρισμα το μονοπάτι που βρίσκεται το σενάριο στην βάση δεδομένων. Μέσα στο σώμα της συνάρτησης γίνεται κλήση της RetrieveOnce στην κλάση FinderPage. Η συνάρτηση ανακτά τα δεδομένα από το μονοπάτι που δέχεται σαν όρισμα με την χρήση της .once('value') μεθόδου και επιστρέφει τα .val() δεδομένα του snapshot. Για την αντικατάσταση του σεναρίου η διαδικασία είναι η ίδια με αυτή του κειμένου με την διαφορά ότι γίνεται χρήση της μεθόδου .update αντί της .set.

```
RetrieveScenarioType(GPath){

returnfirebase.database().ref(GPath).once('value').then((snapshot)
=>{
        console.log("type",snapshot.val());
var type =snapshot.val().ScenarioType;

returntype;


});
}
```

Code 15: Ανάκτηση τιμής με την μέθοδο once('value')

Στην περίπτωση του αύξοντα αριθμού στην συνάρτηση ngOnInit καλείται η GetInum. Μέσα στο σώμα της συγκεκριμένης συνάρτησης μια επανάληψη for διατρέχει το array CordsArray. Ένα if statement ελέγχει εάν τα έξι πρώτα γράμματα του ονόματος σχηματίζουν την λέξη Marker. Εάν ναι, σημαίνει ότι πρόκειται για ένα beacon marker και το key που αντιστοιχεί στο id του εκχωρείται σε ένα array με όνομα Names. Ανεξάρτητα του τύπου marker εκχωρείται σε ένα αντικείμενο σαν στοιχεία ο αύξων αριθμός και το όνομα του marker. Το αντικείμενο εκχωρείται σε ένα array ονόματος nums. Για την αντικατάσταση του αύξων αριθμού γίνεται χρήση της συνάρτησης IncreasingNumber. Μέσα στην συνάρτηση γίνεται έλεγχος για τυχόν διπλότυπα με την χρήση δύο for επαναλήψεων. Η πρώτη επανάληψη διατρέχει το object array *nums* και εκχωρεί τον τρέχον αριθμό στην μεταβλητή CheckNum. Η δεύτερη επανάληψη διατρέχει το ίδιο array και ελέγχει με ένα if statement εάν ο αριθμός που ελέγχεται είναι ίδιος με τον εκάστοτε αριθμό του array. Εάν ναι αυξάνει έναν μετρητή. Ένα δεύτερο if statement ελέγχει εάν ο μετρητής είναι μεγαλύτερος του ένα. Σε περίπτωση που είναι σημαίνει ότι υπάρχουν διπλότυπα, η πρώτη επανάληψη γίνεται break και στον μετρητή εκχωρείται το string 'not'.

Εκτός των επαναλήψεων ένα if statement ελέγχει εάν ο μετρητής έχει την τιμή 'not'. Εάν ναι, γίνεται κλήση της συνάρτησης ReplaceInum στην κλάση ReplacementPage. Η συνάρτηση δέχεται σαν παραμέτρους το object array με τους αύξοντες αριθμούς, το GamePath και CordsArray. Μέσα στην συνάρτηση ένας for βρόγχος διατρέχει το object array. Για κάθε αντικείμενο που αντιστοιχεί σε ένα σημείο ενδιαφέροντος δημιουργεί το κατάλληλο μονοπάτι και με την χρήση του firebase object αντικαθιστά το ήδη υπάρχων αριθμό με την μέθοδο .set. Τέλος επαναφέρει το μονοπάτι στην αρχική τιμή.

Η αντικατάσταση των συντεταγμένων γίνεται στην ίδια κλάση με την εικόνα/video. Στην ngOnInit συνάρτηση της κλάσης ελέγχει εάν η μεταβλητή type είναι 'cords' με την χρήση ενός if statement. Εάν είναι, κάνει κλήση της συνάρτησης FindCords η οποία δέχεται σαν όρισμα το CordsArray. Ένας βρόγχος for διατρέχει το array και ένα if statement ελέγχει εάν το όνομα του marker είναι ίδιο με αυτό που έδωσε ο admin. Εάν ναι, το γεωγραφικό πλάτος και ύψος εκχωρούνται σε public μεταβλητές. Για την αντικατάσταση των συντεταγμένων καλείται η συνάρτηση

SubmitCordinates η οποία με την σειρά της καλεί την συνάρτηση ReplaceCordinates στην κλάση ReplacementPage. Η συνάρτηση δέχεται σαν όρισμα το όνομα του σημείου ενδιαφέροντος, το GamePath, το γεωγραφικό μήκος και πλάτος. Μέσα στην συνάρτηση γίνεται έλεγχος εάν το γεωγραφικό μήκος και πλάτος είναι κενά string. Εάν όχι, με την χρήση του firebase object ορίζεται ένα reference στο μονοπάτι που θα γίνει η αντικατάσταση της υπάρχουσας τιμής. Για την αντικατάσταση γίνεται χρήση της μεθόδου set.

```
IncreasingNumber(num){

var i;

 loop1:
for(let CheckNum of this.nums){
CheckNum=CheckNum.Inum;
console.log("CheckNum is",CheckNum);

for(let num of this.nums){
num=num.Inum;

if(CheckNum==num){
i++;
if(i>1){
this.numConflict();
i="not";
break loop1;
}
}
}

i=0;
}
if(i!="not"){

newReplacementPage(this.af,this.loadingCtrl).
ReplaceInum(this.nums,this.GamePath,this.CordsArray);
}
```

Code 16: Έλεγχος για διπλότυπα

Δημιουργία Σεναρίου

Η κλάση GpsPage είναι η κύρια κλάση που χρησιμοποιείται για την δημιουργία ενός νέου σεναρίου. Αρχικά η κλάση MakePoiPage είναι αυτή που δέχεται την εισαγωγή του admin για το όνομα, κωδικό και τύπο για το σενάριο που θα δημιουργηθεί. Όλα αυτά εκχωρούνται σαν παράμετροι στην κλάση GpsPage όταν αυτή γίνεται push στην στοίβα. Στον constructor της GpsPage ένα ξεχωριστό id γίνεται push στην firebase με την χρήση λίστας. Στην συνάρτηση ngOnInit γίνεται έλεγχος εάν το Gps της συσκευής είναι ενεργοποιημένο με την χρήση του cordova plugin Diagnostic ένα δεν είναι κατάλληλο μήνυμα εμφανίζεται στην οθόνη. Στην συνέχεια καλείται η συνάρτηση loadMap. Η συγκεκριμένη συνάρτηση χρησιμοποιεί το geolocation cordova plugin για την ανάκτηση της τρέχουσας τοποθεσίας. Οι συντεταγμένες τις τοποθεσίας εκχωρούνται σε μια μεταβλητή με την χρήση της google.maps.latlng class. Στο mapOptions αντικείμενο τοποθετούνται σαν στοιχεία οι συντεταγμένες που θα βρίσκονται στο κέντρο του χάρτη, το zoom και τον τύπο του χάρτη. Στη συνέχεια ο χάρτης δημιουργείται και δέχεται σαν ορίσματα το element του χάρτη από το html, το αντικείμενο mapOptions και καλείται η συνάρτηση MarkerStaff. Η συγκεκριμένη συνάρτηση προσθέτει ένα click listener σε όλο τον χάρτη όπου σε κάθε click καλείται η συνάρτηση AddMarker.

Η συνάρτηση AddMarker δέχεται σαν παραμέτρους ένα constructor google.maps.Marker και το argument του event. Στην συγκεκριμένη συνάρτηση το link μίας εικόνας που θα χρησιμοποιηθεί σαν marker εκχωρείται σε μία μεταβλητή ονόματος img. Ένας marker τοποθετείται στις συντεταγμένες που αντιστοιχούν στο σημείο που σημειώθηκε το click event και τοποθετείται στον χάρτη. Το γεωγραφικό πλάτος και μήκος εκχωρούνται σε public μεταβλητές. ΄Ενα if statement ελέγχει εάν το όνομα του σημείου ενδιαφέροντος είναι undefined. ΄Ενα δεύτερο if statement ελέγχει εάν τα πρώτα έξι γράμματα του ονόματος δεν σχηματίζουν την λέξη ‘Marker’. Εάν ισχύουν τα δύο παραπάνω statement ο listener mousedown και mouseup προστίθενται στον marker. Κατά το mouseup καλείται ξανά η AddMarker συνάρτηση. Οι δύο αυτοί listeners είναι ένας τρόπος τεχνικού push στην οθόνη της συσκευής για την προσθήκη ενός δεύτερου marker στην ίδια τοποθεσία. Εάν κανένα από τα δύο statement δεν ισχύουν το όνομα του marker ορίζεται ως ‘Marker’ συν την τιμή του μετρητή (π.χ Marker1) . Ο μετρητής αρχικοποιείται κατά την δήλωση του και αυξάνεται όταν θα ανεβεί στην βάση δεδομένων.

Για την επιλογή ενός beacon marker δημιουργείται ένα νέο modal στην κλάση BeaconPage. Στην ngAfterViewInit συνάρτηση της κλάσης. Μέσα στο σώμα της συγκεκριμένης συνάρτησης καλείται η συνάρτηση bluetooth η οποία κάνει χρήση του ble cordova plugin για να ελέγξει εάν το Bluetooth της συσκευής είναι ενεργοποιημένο. Εάν δεν είναι ζητάει το δικαίωμα να το ενεργοποιήσει. Η συνάρτηση startScanning κάνει χρήση του plugin ble για την αναζήτηση των πλησιέστερων beacon με τη χρήση της συνάρτησης scan του plugin. Η εν λόγω συνάρτηση δέχεται σαν παραμέτρους ένα κενό array και τα δευτερόλεπτα που θα διαρκέσει η αναζήτηση. Η συνάρτηση επιστρέφει τα στοιχεία των συσκευών που βρίσκει και τα τοποθετεί σε ένα array. Η συνάρτηση list καλείται κάθε φορά που ο admin επιλέγει ένα beacon από την λίστα. Μέσα στο σώμα της συνάρτησης το id του beacon εκχωρείται σε μια public και καλείται η συνάρτηση dismiss που επιστρέφει το id στην καλούσα συνάρτηση.

```
startScanning(){

this.ble.scan([],5).subscribe(device =>{
varadData=new Uint8Array(device.advertising)



this.devices.push(device);







}, error =>{
console.log("scan error");
console.log(error);
});
}
```

Code 17: Συνάρτηση startScanning

Για την εισαγωγή φωτογραφίας από την κάμερα της συσκευής γίνεται κλήση της συνάρτησης OpenCam. Αρχικά στην συγκεκριμένη συνάρτηση η μεταβλητή Ispin γίνεται true. Η μεταβλητή αυτή ξεκινάει ένα spinner στην οθόνη στην θέση του κουμπιού. Στην συνέχεια καλείται η συνάρτηση Cam στην κλάση CamPage με την χρήση promise. Η συνάρτηση Cam χρησιμοποιεί το cordova plugin media capture και της capture image συνάρτησης του για να αποκτήσει πρόσβαση στην κάμερα. Η εν λόγω συνάρτηση δέχεται σαν όρισμα ένα αντικείμενό με τον αριθμό των αριθμό των φωτογραφείων που θα τραβήξει ο χρήστης και επιστρέφει πληροφορίες για την φωτογραφία που τράβηξε στην καλούσα συνάρτηση. Στην περίπτωση σφάλματος επιστρέφει false. Ένα if statement ελέγχει εάν η τιμή που επιστράφηκε είναι false. Εάν είναι, εκχωρείται στην μεταβλητή Ispin η τιμή false. Στην συνέχεια το path γίνεται resolve με τη χρήση του filepath cordova plugin και τα στοιχεία του (όνομα φακέλου, αρχείου, τύπος) εξάγονται και όπως στην κλάση HomePage. Η φωτογραφία μετατρέπεται σε data URL με τον ίδιο τρόπο όπως στην κλάση HomePage.

```
OpenCam(){

this.Ispin=true;

newCamPage(this.file,this.mediaCapture).Cam().
then(imgData=>
{

if(imgData==false)
this.Ispin=false;

this.filePath.resolveNativePath(imgData[0].fullPath).
then((filePath)=>{
varDataInfo=newStringManipulationPage().CheckType(filePath);

newCamPage(this.file,this.mediaCapture).
dataUrl(DataInfo.DirPath,DataInfo.Name+".jpg").then(FileEntry=>
{

this.Images.push(FileEntry);

this.Ispin=false;
```

Code 18: camera opening

```
Cam(){



letoptions:CaptureImageOptions={ limit:1};
returnthis.mediaCapture.captureImage(options)
.then(
function(data:MediaFile[]){


return data;
},(err:CaptureError)=>{
console.error("capture error",err);
returnfalse;
});



}
```

Code 19: Συνάρτηση Cam

Για την εισαγωγή video γίνεται χρήση του ίδιου cordova plugin όπως με την φωτογραφεία. Η διαδικασία είναι η ίδια για το άνοιγμα της κάμερας και τις επιστροφής των στοιχείων από το plugin. Η μόνη διαφορά τους είναι ότι η συγκεκριμένη συνάρτηση δημιουργεί ένα thumbnail για το video. Για την δημιουργία του κάνει κλήση της συνάρτησης MakeVideoThumnail. Η συνάρτηση δέχεται σαν όρισμα το URI του video και το επιθυμητό όνομα για το thumbnail. Μέσα στο σώμα της συνάρτησης καλείται η συνάρτηση VideoThumbnail στην κλάση VidPage. Η συγκεκριμένη συνάρτηση δέχεται σαν όρισμα τις ίδιες παραμέτρους με την MakeVideoThumnail και κάνει χρήση του videoEditor cordova plugin και της createThumbnail συνάρτησης του. Η συνάρτηση επιστρέφει το thumbnail στην καλούσα. Στην συνέχεια καλείται η συνάρτηση ThumbdataUrl στην CamPage κλάση. Αυτή η συνάρτηση δέχεται σαν παραμέτρους το όνομα του thumbnail και το μονοπάτι που βρίσκεται το thumbnail στην συσκευή.

Μέσα στο σώμα της  το μονοπάτι μετατρέπεται σε content με την προσθήκη του string ‘file://’ στην αρχή. Για την εξαγωγή του μονοπατιού του φακέλου χρησιμοποιείται της μεθόδου replace εξάγεται το όνομα του και ο τύπος του αρχείου. Στη συνέχεια με την χρήση του file cordova plugin και της readAsDataURL συνάρτηση το αρχείο μετατρέπεται σε data URL και επιστρέφει στην καλούσα συνάρτηση. Κατόπιν γίνεται κλήση της AddPlayIcon συνάρτησης όπως και στην Homepage κλάση

```
MakeVideoThumnail(vidUri,thumbName){



return
newVidPage(this.mediaCapture,this.videoEditor).
VideoThumbnail(vidUri,thumbName).then(result=>{


returnnewCamPage(this.file,this.mediaCapture).
ThumbdataUrl(result,thumbName).then(FileEntry=>
{

returnthis.AddPlayIcon(FileEntry).then(thumbnail =>{

returnthumbnail;
```

Code 20: Συνάρτηση MakeVideoThumbnail

Στην περίπτωση εισαγωγής αρχείων από το σύστημα αρχείων της συσκευής γίνεται χρήση της συνάρτησης Open. Ο τρόπος με τον οποίο *ανοίγει το σύστημά* αρχείων, μετατρέπει το αρχείο σε data URL, δημιουργίας του thumbnail είναι ίδιος με αυτόν στην κλάση HomePage. Η μόνη διαφορά τους είναι ότι στις περιπτώσεις εικόνας/ήχου όταν η dataUrl συνάρτηση επιστρέφει το αρχείο σε μορφή data URL το αρχείο εκχωρείται σαν στοιχείο σε ένα αντικείμενο μαζί με το thumbnail στη περίπτωση του video και του στοιχείου Autoplay με τιμή 'No'. Στην συνέχεια το αντικείμενο εκχωρείται σε ένα array και η τιμή Fspin δέχεται την τιμή false. Η Fspin ξεκινάει ένα spinner στην οθόνη στην θέση του κουμπιού για όση ώρα χρειάζεται η συσκευή για να διαβάσει το αρχείο.Στην περίπτωση του ηχητικού κομματιού εκχωρείται στην μεταβλητή chk η τιμή false. Εάν το αρχείο που επιλέχθηκε δεν είναι τις τύπου mp3, mp4 ή jpg. Εμφανίζεται κατάλληλο μήνυμα στην οθόνη. Η συνάρτηση Aplay αλλάζει την τιμή του Autplay στοιχείου της τελευταίας εκχώρησης στο array Audio από 'No' σε 'Auto'. Για την εύρεση της τελευταίας εκχωρήσεις γίνεται χρήση του Property accessor με το μήκος του array μείον 1. Για την εισαγωγή κειμένου ένα νέο modal δημιουργείται στην κλάση TextPage. Η κλάση δέχεται το κείμενο που εισάγει ο admin. Στο dismiss το κείμενο επιστρέφεται στην καλούσα συνάρτηση.

Για την εισαγωγή ερωτήσεων πολλαπλής επιλογής ένα νέο modal δημιουργείται στην κλάση MultipleChoicePage σαν παραμέτρους δέχεται ένα κενό array ονόματος Name. Στην συνάρτηση ngOnInit της κλάσης ανακτώνται τα δεδομένα και μία φόρμα δημιουργείται με την χρήση του angular FormBuildler. Αρχικά ένα form group εκχωρείται σε ένα parent FormGroup με όνομα myForm. Το formGroup είναι ένα directive που συσχετίζει την φόρμα με ένα HTML element. Μέσα στο form group ένα κενό form array ορίζεται. Μέσα στο array ορίζεται ένα child form group. Η μεταβλητή myForm εκχωρείται σαν ένα FormArray στην μεταβλητή formcrt. Το array Names αντιγράφεται σε ένα δεύτερο ονόματος temp και στη συνέχεια αδειάζει.Μία επανάληψη for διατρέχει το array temp μέσα στο σώμα του βρόγχου καλείται η συνάρτηση addFields η οποία δέχεται σαν όρισμα το εκάστοτε αντικείμενο του temp. Ένα if statement ελέγχει εάν ο τίτλος είναι undefined ή κενός. Εάν δεν είναι, εκχωρείται στο array Name και ένα καινούριο form group εκχωρείται στο formcrt array.

Η συγκεκριμένη επανάληψη θα εκτελεσθεί την δεύτερη φορά που ο admin θα καλέσει την συγκεκριμένη κλάση για να αλλάξει ή να προσθέσει ένα ερώτημα επειδή το array δεν θα είναι κενό. Για να ξανά απεικονισθούν στην οθόνη. Ο έλεγχος αποκλείει τα ερωτήματα με κενό τίτλο. Στη συνέχεια η συνάρτηση DataObject καλείται η οποία εκχωρεί ένα object με στοιχεία ένα string με όνομα Title, ένα enum με όνομα formData και ένα string με όνομα CorrectAnswer στο array Names. Σε κάθε αντικείμενο αντιστοιχεί εκχωρούνται τα στοιχεία που εισήγαγε ο χρήστης στη φόρμα. Η συνάρτηση removeData αφαιρεί ένα στοιχείο/φόρμα από το formcrt array.

```
      this.myForm = this.fb.group({

            inputData: this.fb.array([

                        this.fb.group({

        })
            ])
        });
       this.formcrt = <FormArray>this.myForm.controls['inputData'];

          var temp=this.Name
             this.Name=[];
       for (let Name of temp){

                  this.addFields(Name);
  }
               this.DataObject();
```

Code 21: Reactive φόρμα

```
addFields(Name){

if(!(Name.Title==undefined||Name.Title="")){

this.Name.push(Name);
this.formcrt.push(this.fb.group({multiple:['']}));
}
}

addData(){
this.DataObject();

this.formcrt.push(this.fb.group({multiple:['']}));
        console.log("Name",this.Name);
}

removeData(i: number){
this.formcrt.removeAt(i);
}
```

Code 22: Προσθήκη και αφαίρεση από το FormArray

Κατά το dissmis του modal επιστρέφεται το array Name στην καλούσα συνάρτηση. Στην συνέχεια ένας βρόγχος for διατρέχει το array Names και ελέγχει εάν το πεδίο τίτλος του εκάστοτε ερωτήματος δεν είναι κενός ή undefined. Εάν δεν είναι ένας ακόμα for βρόγχος κάνει τρείς επανάληψης. Ένα if statement ελέγχει εάν κάποιες από τις πιθανές απαντήσεις είναι κενή ή undefined. Εάν είναι ένας μετρητής αυξάνει. Εκτός της δεύτερης for ένα if statement ελέγχει εάν η τιμή του μετρητή είναι μεγαλύτερη του δύο. Στην περίπτωση που είναι, εκχωρείται στο τίτλο η τιμή undefined έτσι ώστε να απορριφθεί στο έλεγχο όταν το modal ξανά κληθεί ή από την firebase. Το array Name εκχωρείται δε μία public μεταβλητή ίδιου ονόματος.

```
for(let k=0; k<Name.length; k++){
var c=0;
console.log("K is",k);
if(!(Name[k].Title ==undefined|| Name[k].Title ==""){


for(leti=0;i<3;i++){

if(Name[k].Fdata[i]==undefined){
c++;
}
if(Name[k].Fdata[i]==""){
c++;
}
}
console.log("iis",c);
if(c >=2){


     Name[k].Title=undefined;
}
this.Name=Name;
```

Code 23: Έλεγχος για κενά πεδία

Όλα τα δεδομένα γίνονται upload με την κλήση της submitData συνάρτησης και αντιστοιχεί στο κουμπί submit στο HTML. Στις πρώτες γραμμές της συνάρτησης ανακτώνται τα δεδομένα από την κλάση MakePoiPage.Τα δεδομένα αυτά είναι τα GameName, ScenarioType και Password για των κωδικό του σημείου ενδιαφέροντος. Τα StoragePath, ScenarioPath και το μονοπάτι που θα εισαχθούν τα δεδομένα στην βάση δεδομένων με όνομα dbPath δημιουργούνται. Ένα if statement ελέγχει εάν το όνομα του σημείου ενδιαφέροντος είναι κενό. ¨Εάν δεν είναι, καλείται η συνάρτηση FirebaseSet στην κλάση FirePage η συνάρτηση δέχεται σαν όρισμα τα dbPath, τα latit και longit που αντιστοιχούν στα γεωγραφικο μήκος και πλάτος. Μέσα στο σώμα της συγκεκριμένης συνάρτησης ένα firebase object reference ορίζεται στο μονοπάτι που θα εισαχθούν το γεωγραφικό πλάτος και μήκος. Η εισαγωγή τους γίνεται με την χρήση της set μεθόδου.

Κατόπιν καλείται η συνάρτηση με όνομα FirebaseUpload στην FirePage κλάση. Η συνάρτηση αυτή δέχεται σαν παραμέτρους το κείμενο, τον κωδικό, τα StoragePath, dbPath, τα array με τα video, εικόνες, ήχο, το κείμενο από την φόρμα, τον μετρητή, το ScenarioPath, τύπο του σεναρίου, το array Names με τα ερωτήματα πολλαπλής επιλογής. Μέσα στο σώμα της συνάρτησης μία for επανάληψη διατρέχει το array Name. Μια δεύτερη επανάληψη εμφωλευμένη στην πρώτη

τρέχει τρείς φορές και εισάγει τις πιθανές ερωτήσεις των ερωτημάτων στην βάση δεδομένων. Σε κάθε επανάληψη η τιμή του μετρητή αυξημένη κατά ένα εκχωρείται στην μεταβλητή num και χρησιμοποιείται ως αύξων αριθμός στο όνομα της εισαγωγής (Answer1). Εκτός του σώματος της επανάληψης εισάγεται η σωστή απάντηση. Για τον εισαγωγή του τύπο σεναρίου και του κωδικού ένα if statement ελέγχει εάν ο μετρητής είναι ίσος με μηδέν . Εάν είναι, τα δεδομένα εισάγονται με την χρήση της μεθόδου update του firebase object. Στη συνέχεια εισάγεται ο αύξων αριθμός και το κείμενο με την ίδια μέθοδο όπως και ο τύπος σεναρίου. Στην περίπτωση του κειμένου ένα if statement ελέγχει εάν το κείμενο είναι undefined.

Για την εισαγωγή ήχου, εικόνας και video ένας βρόγχος for διατρέχει κάθε ένα από τα arrays που βρίσκονται τα δεδομένα. Μέσα στο σώμα κάθε επανάληψης καλείται η συνάρτηση SetParams. Η εν λόγω συνάρτηση δέχεται σαν παραμέτρους το storagePath, dbPath, το αρχείο που θα γίνει upload, τον μετρητή και το storage reference. Η συνάρτηση μεταβάλει το μονοπάτι dbPath και storagePath για το κάθε αρχείο ξεχωριστά. Επίσης δημιουργεί την αναφορά στο storagePath του storage της firebase και της μεθόδου putString που θα χρησιμοποιηθεί για το upload του αρχείου. Κάθε αρχείο έχει το δικό του μοναδικό όνομα στο σημείο ενδιαφέροντος όπου επισυνάπτετε. Τα αρχεία ήχου, εικόνας και video έχουν ονόματα του τύπου imageInput συν τον αύξοντα αριθμό τους στο array επαυξημένο κατά ένα. Στην θέση του image υπάρχει video ή audo ανάλογα τον τύπο του αρχείου. Για τα κομμάτια ήχου υπάρχει το όνομα Aplay εάν το ηχητικό κομμάτι έχει ορισθεί σαν autolay.

Τα αρχεία γίνονται upload με την χρήση της μεθόδου storeInputData η συγκεκριμένη μέθοδος δέχεται σαν ορίσματα την μεταβλητή task, το storagePath, dbpath, Angular Fire Database injection με όνομα af. Η συγκεκριμένη συνάρτηση επιστρέφει ένα promise. Μέσα στο promise η μέθοδος task.on κάνει χρήση τριών συναρτήσεων για την πρόοδο του upload, για τυχόν λάθη και για την ολοκλήρωση του. Στην πρώτη συνάρτηση ένα if statement ελέγχει εάν υπάρχει η λέξη snapshot στο μονοπάτι. Εάν όχι εισάγεται μια εγγραφή με το μονοπάτι του αρχείου στο storage. Στην ολοκλήρωση γίνεται resolve και ένας μετρητής αυξάνεται. ΄Ένα if statement ελέγχει εάν ο μετρητής είναι ίσος με το μέγεθος του array εάν ναι, σημαίνει ότι έχουν ανεβεί όλα τα αρχεία και κατάλληλο μήνυμα εμφανίζεται στην οθόνη με την χρήση του alert μεθόδου της javascript. Σε αντίθετη περίπτωση αυξάνεται ο μετρητής κατά ένα.

Στην περίπτωση του video καλούνται δύο φορές οι συναρτήσεις SetParams και storeInputData. Η πρώτη αφορά το video και η δεύτερη το thumbnail του, η storeInputData για το upload του thumbnail δεν κάνει χρήση promise γιατί το upload του είναι αόρατο ως προς τον χρήστη και ως εκ τούτου δεν χρειάζεται να ενημερωθεί για την ολοκλήρωση του. Τέλος στην περίπτωση του ηχητικού κομματιού ένα if statement ελέγχει εάν το κομμάτι έχει ορισθεί ως Autoplay ή όχι. Για αμφότερες τις περιπτώσεις καλείται η συνάρτηση SetParams με διαφορετικό όρισμα για το όνομα του τύπου. Για Autoplay το όρισμα είναι ‘AplayAudio’ για απλό ηχητικό κομμάτι το όρισμα είναι ‘Audio’. Η FirebaseUpload επιστρέφει τον μετρητή στην καλούσα αφού αυξήσει την τιμή του κατά ένα.

```
SetParams(StoragePath,dbPath,file,count,type,storageRef){

this.dbpath=dbPath+"/Data/"+type+"Input"+count;
this.storagePath=StoragePath+"/"+type+"Input"+count;
this.dataRef=storageRef.child(this.storagePath);
this.task=this.dataRef.putString(file,'data_url');
}
```

Code 24: Συνάρτηση SetParams

```
storeInputData(task,StoragePath,dbPath,af){


returnnewPromise( resolve=>{

task.on('state_changed',
function progress(snapshot){
var percentage =snapshot.bytesTransferred/snapshot.totalBytes*100;

console.log(percentage,"%  completed ");
if(dbPath.includes("thumbnail")==false){
                console.log(StoragePath,"StoragePath");
this.obj=af.object(dbPath);
this.obj.set(StoragePath);
}


},
function error(err){
console.log("error is:",err);
},
functioncomplete(){

resolve("");
```

Code25: Upload στο firebase storage

```
var v=0;
for(leti=0;i<Videos.length;i++){


this.Loading(this.Vloader);this.SetParams(StoragePath,dbPath,Videos[i].video,
i,"video",storageRef);
this.storeInputData(this.task,this.storagePath,this.dbpath,this.af).then(func
tion(){
if(v == Videos.length-1){

alert("Upload of Video Complete");


}
                              v++;
})
this.Loading(this.Vloader);
this.SetParams(StoragePath,dbPath,Videos[i].thumbnail,i+"thumbnail","video",
storageRef);
this.storeInputData(this.task,this.storagePath,this.dbpath,this.af,);

}
```

Code 26: Βρόγχος στο video array για τον ορισμό παραμέτρων και upload

## Users

Για τα σενάρια που συμμετείχε και την διάδραση του απλού χρήστη σε αυτά γίνεται χρήση της κλάσης StartPage. Στην ngOnInit συνάρτηση της γίνεται κλήση της ShowUsers. Η συγκεκριμένη συνάρτηση καλεί την συνάρτηση RetrieveSnapshot στην κλάση FinderPage. Η RetrieveSnapshot δέχεται σαν όρισμα το μονοπάτι στην βάση δεδομένων από όπου θα ανακτήσει τα υπάρχοντα δεδομένα σαν snapshot και θα τα επιστρέψει. Στην καλούσα συνάρτηση η snapshot εκχωρείται σε μια public μεταβλητή. Η συνάρτηση ShowDetails καλείται για να ανακτήσει τα αρχεία διάδρασης του χρήστη. Η εν λόγω συνάρτηση δέχεται σαν όρισμα το μοναδικό key όπου από κάτω του βρίσκονται σαν παιδία του όλα τα δεδομένα του χρήστη. Συνθέτη το μονοπάτι και καλή την συνάρτηση RetrieveSnapshot και εκχωρεί το snapshot που επέστρεψε σε μια public μεταβλητή.

Η συνάρτηση UserResponses ανακτά τα δεδομένα διάδρασης του χρήστη. Σαν όρισμα δέχεται το μοναδικό id του σεναρίου. Η εν λόγω συνάρτηση καλεί την RetrieveSnapshot δύο φορές την πρώτη φορά δέχεται σαν όρισμα το μονοπάτι όπου βρίσκονται όλα τα δεδομένα του χρήστη επιστρέφει ένα snapshot. Μία foreach επανάληψη διατρέχει το snapshot το οποίο επέστρεψε. Κάθε key μπορεί να αντιστοιχεί στο όνομα του σημείου ενδιαφέροντος στο οποίο έγινε η διάδραση, στο

όνομα του σενάριο ή το μονοπάτι του σεναρίου. ΄Ενα if statement ελέγχει εάν τα πρώτα τέσσερά γράμματα του key δεν σχηματίζουν την λέξη ‘Game’. Εάν ναι, σημαίνει ότι πρόκειται για κάποιο σημείο ενδιαφέροντος και γίνεται κλήση της συνάρτησης showData η οποία δέχεται σαν όρισμα το val του snapshot. Η εν λόγω συνάρτηση είναι ίδια με την DisplayFile της HomePage. Στη συνέχεια το key της snapshot εκχωρείται σε ένα array ονόματος PoiName.

Στην κλήση της δεύτερης RetrieveSnapshot δέχεται σαν όρισμα το μονοπάτι του σημείου ενδιαφέροντος όπου βρίσκονται τα δεδομένα της διαδρασης του. Μια foreach επανάληψη διατρέχει το snapshot που επιστράφηκε. Μέσα στην επανάληψη ένα if statement ελέγχει εάν η τιμή key είναι MultipleChoices. Εάν ναι, η val τιμή εκχωρείται στην μεταβλητή UserAnswers και καλείται η συνάρτηση Mchoices. Η συγκεκριμένη συνάρτηση δέχεται σαν παραμέτρους το μονοπάτι που βρίσκεται το ερώτημα πολλαπλής επιλογής και το UserAnswers. Στο σώμα της συγκεκριμένης συνάρτησης καλείται η συνάρτηση RetrieveKey της κλάσης FinderPage. Η συνάρτηση αυτή επιστρέφει στην καλούσα ένα array με τα key του snapshot. Κάθε key στο array αντίστοιχή στο τίτλο του ερωτήματος πολλαπλής επιλογής. Μια for επανάληψη διατρέχει το array. Ένα if statement ελέγχει εάν η απάντηση του χρήστη δεν είναι undefined. Εάν δεν είναι, εκχωρούνται σαν παράμετροι μέσα σε ένα object ο τίτλος, ένα κενό string με όνομα Answer και η απάντηση του χρήστη. Το αντικείμενο εκχωρείται μέσα σε ένα array με όνομα MultipleChoicesAnswers.

Στη συνέχεια καλείται για δεύτερη φορά η RetrivePosition με όρισμα το μονοπάτι που βρίσκεται η ερώτηση πολλαπλής απάντησης στο σημείο ενδιαφέροντος. ΄Ενας δεύτερος βρόγχος for εμφωλευμένος στον πρώτο διατρέχει το array που επέστρεψε η συνάρτηση και δύο if statement ελέγχουν εάν το όνομα της εισαγωγής στην βάση δεδομένων είναι ‘CorrectAnswer’. Εάν είναι, εκχωρείται σε μία μεταβλητή ίδιου ονόματος. Σε αντίθετη περίπτωση σημαίνει ότι η εισαγωγή αντιστοιχεί σε πιθανή απάντηση και εκχωρείται σε ένα array. Εκτός της δεύτερης for επανάληψης ένα if statement ελέγχει εάν η απάντηση του χρήστη δεν είναι undefined και ένα δεύτερο ελέγχει εάν είναι ίδια με την σωστή. Εάν ναι, εκχωρεί σε ένα αντικείμενο τον τίτλο, το array με τις απαντήσεις και την σωστή απάντηση. Σε αντίθετη περίπτωση εκχωρούνται σε ένα αντικείμενο όλα τα προηγούμενα συν την απάντηση του χρήστη.

```
Mchoices(path,UserAnswers){

newFinderPage(this.af).RetrieveKey(path+"/Data/MultipleChoices").
subscribe(MultipleChoicesKeys=>{


for(let key ofMultipleChoicesKeys){
if(UserAnswers[key]!=undefined){
vartmp={ Title: key,
                                    Answer:"",
UserAnswers:UserAnswers[key]
}
this.MultipleChoicesAnswers.push(tmp);
}


newFinderPage(this.af).
RetrivePosition(path+"/Data/MultipleChoices/"+key).
subscribe(MultipleChoices=>{

varCorrectAnswer;
var answers=[];
for(let answer ofMultipleChoices){

if(answer.key=="CorrectAnswer")
CorrectAnswer=answer.val();
else
answers.push(answer.val());
console.log(answer.key,answer.val());
}
vartmp;
if(UserAnswers[key]!=undefined){
if(UserAnswers[key]==CorrectAnswer){
tmp={ Title: key,
                                    Answers: answers,
CorrectAnswer:CorrectAnswer,

}
}
else{
tmp={ Title: key,
                                    Answers: answers,
CorrectAnswer:CorrectAnswer,
UserAnswers:UserAnswers[key]
}
}

                            answers=[];
this.MultipleChoices.push(tmp);
}
})

}
```

Code 27: Ανάκτηση ερωτημάτων πολλαπλής επιλογής και ορισμός τους

### 3.3.2 Εφαρμογή Χρήστη

Για την εγγραφή ή την σύνδεση του χρήστη υπάρχουν τρείς κλάσεις όπως στην εφαρμογή για τον admin. Οι κυριότερες διαφορές είναι ότι στην συνάρτηση ngOnInit ελέγχεται εάν είναι ενεργοποιημένα τα GPS, Bluetooth και Wi-Fi με τη χρήση του cordova diagnostic plugin. Γίνεται χρήση ενός firebase project. Επίσης μετά την εγγραφή/σύνδεση του χρήστη ανακατευθύνεται στην κλάση MenuPage. Κατά την αποσύνδεση μία σελίδα αφαιρείται από την navigation στοίβα.

```
this.diagnostic.isGpsLocationEnabled()

.then((state)=>{

if(state ==false){
var subtitle="Please Turn on your Gps";

this.LocationAlert(subtitle);
}

}).catch(e =>console.error("error is:",e));
```

Code28: Gps Diagnostic

Στην κλάση ScenarioPage η συνάρτηση AvailableGames καλείται μέσω της ngOnInit όταν η κλάση αρχικοποιείται. Στην AvailableGames γίνεται κλήση της συνάρτησης RetrieveKey η οποία βρίσκεται στην κλάση FinderPage και δέχεται σαν παράμετρο το μονοπάτι που βρίσκονται όλα τα σενάρια στην βάση δεδομένων. Η συγκεκριμένη συνάρτηση επιστρέφει στην καλούσα ένα array ονόματος keys με τα μοναδικά id του κάθε σεναρίου . Μια επανάληψη for διατρέχει το array και για κάθε id. Μέσα στο σώμα της επανάληψης καλείται η συνάρτηση RetrieveGameName τις κλάσης FinderPage. Στην εν λόγω κλάση ανακτάται τα δεδομένα από την βάση δεδομένων από το μονοπάτι που έχει σαν όρισμα. Ένας foreach βρόγχος διατρέχει το snapshot και για κάθε στοιχείο του εκχωρεί σαν παραμέτρους σε ένα αντικείμενο το key που αντιστοιχεί στο όνομα του σεναρίου και το μονοπάτι του σεναρίου. Το αντικείμενο επιστρέφεται στην καλούσα συνάρτηση. Ένα if statement ελέγχει εάν το αντικείμενο δεν είναι undefined. Εάν δεν είναι, εκχωρείται σε ένα array με όνομα GamesArray.

Η συνάρτηση PresentPromt δέχεται σαν όρισμα τα GamePath και GameName. Στο σώμα της συνάρτησης δημιουργείται ένα alert σαν ιδιότητες δέχεται τον τίτλο του alert ένα array με όνομα input με στοιχεία όνομα, placeholder και τύπος. Ένα array με όνομα buttons δύο αντικείμενα. Κάθε αντικείμενο αντιστοιχεί σε ένα κουμπί και έχει σαν στοιχεία το όνομα του και έναν handler. Στον handler του δεύτερου κουμπιού καλείται η συνάρτηση RetrieveOnce της κλάσης FinderPage. Η συνάρτηση δέχεται σαν όρισμα το μονοπάτι όπου βρίσκεται ο κωδικός του σεναρίου στην βάση δεδομένων τον ανακτά και τον επιστρέφει στην καλούσα συνάρτηση με όνομα μεταβλητής GamePassword. Ένα if statement συγκρίνει τον κωδικό του σεναρίου με αυτόν που εισήγαγε ο χρήστης. Εάν ο κωδικός είναι σωστός καλείται η συνάρτηση StartGame η οποία κάνει push στην στοίβα την κλάση StartPage με δεδομένα το GameName. Σε αντίθετη περίπτωση καλείται η συνάρτηση WpresentPrompt. Η συγκεκριμένη συνάρτηση είναι ίδια με την προηγούμενη με την μόνη διαφορά ότι έχει μια παράμετρο περισσότερη την subTitle με το μήνυμα 'wrong password'.

```
presentPrompt(GamePath,GameName){
this.GameName=GameName;
letalert=this.alertCtrl.create({
    title:'Login',

inputs:[

{
        name:'password',
        placeholder:'Password',
        type:'password'
}
],
buttons:[
{
text:'Cancel',

        handler: data =>{
console.log('Cancel clicked');
}
},
{
text:'Login',
        handler: data =>{

           console.log("GamePassword",this.Retrieve(GamePath));
new FinderPage(this.af).RetrievePassword(GamePath+"/Password").then(GamePassword
=>
{

if(data.password==GamePassword){

this.StartGame();
}
else{


this.WpresentPrompt(GamePath);

}
})

}
}
],
enableBackdropDismiss:false
});
alert.present();

}
```

Code29: promt alert

Στην κλάση StartPage η συνάρτηση ngOnInit κάνει χρήση του cordova plugin nativeStorage για να ανακτήσει items που τυχόν έχουν εκχωρηθεί με την χρήση της μεθόδου keys. Επίσης γίνεται χρήση του plugin BackroundMode και των μεθόδων του enable, disableWebViewOptimizations το οποίο επιτρέπει την εφαρμογή και το Gps να τρέχουν στο background. Στη συνέχεια η συνάρτηση FindMyPosition καλείται. Η εν λόγω συνάρτηση καλεί την συνάρτηση RetrieveKey στην κλάση FinderPage η συγκεκριμένη συνάρτηση επιστρέφει ένα array με τα μοναδικά id των σεναρίων. Ένας

for βρόγχος διατρέχει το array και για κάθε id γίνεται κλήση RetrievePath. Η συγκεκριμένη συνάρτηση επιστρέφει το path του σεναρίου στην καλούσα συνάρτηση με όνομα GPath. 'Ενα if statement ελέγχει εάν το path δεν είναι undefined. Εάν δεν είναι το id του σεναρίου εκχωρείται σε μία public μεταβλητή ονόματος id. Το GamePath δημιουργείται και οι συναρτήσεις GetAccuracy και GetScenarioType καλούνται. Η πρώτη δέχεται σαν όρισμα το μονοπάτι του σεναρίου και η δεύτερη το μονοπάτι του σεναρίου και το GameName.

Και οι δύο αυτές συναρτήσεις καλούν την RetrieveOnce συνάρτηση στην σελίδα FinderPage. Η GetAccuracy ανακτά την ακρίβεια του γεωγραφικού πλάτους και μήκους από την βάση δεδομένων και το εκχωρεί σε μια public μεταβλητή. Η GetScenarioType ανακτά τον τύπο του σεναρίου το εκχωρεί σε μια public μεταβλητή και καλεί την συνάρτηση IamIn. Η εν λόγω συνάρτηση εισάγει στην βάση δεδομένων τα στοιχεία του σεναρίου που συμμετέχει ο χρήστης. Κάνει χρήση του μοναδικού id του χρήστη που εξάγει με την χρήση του AngularFireAuth plugin και της μεθόδου currentUser. Όπως επίσης της μεθόδου push του firebase list για την εισαγωγή ενός μοναδικού id και της set για την εισαγωγή των GamePath, GameName. Μετά την ολοκλήρωση της η συνάρτηση επιστρέφει στην καλούσα και καλείται η Position.

Η συνάρτηση Position κάνει κλήση της RetrivePosition στην κλάση FinderPage. Η συνάρτηση επιστρέφει ένα data snapshot με τα δεδομένα του σεναρίου. Ένας βρόγχος for διατρέχει το snapshot. Μέσα στο σώμα του βρόγχου εκχωρείται σε ένα array ο αύξων αριθμός του σημείου ενδιαφέροντος. Το array ορίζεται σαν παράμετρος στην συνάρτηση LowNum τις κλάσης ChronPage. Η εν λόγω συνάρτηση ταξινομεί τα στοιχεία τις λίστα με αύξουσα σειρά. Στη συνέχεια σε ένα object εκχωρούνται σαν παράμετροι το όνομα του σημείου ενδιαφέροντος με όνομα PoiName, το γεωγραφικό μήκος και πλάτος με όνομα latlng και ο αύξων αριθμός με όνομα Inum. Το αντικείμενο εκχωρείται στο array MapCords. Στη συνέχεια εκτός του βρόχου καλείται η συνάρτηση LowestNum στην κλάση ChronPage. Η συνάρτηση δέχεται σαν όρισμα το object array MapCords και ταξινόμοι τα object κατά αύξουσα σειρά. Στη συνέχεια καλείται η συνάρτηση loadMap.

Στην συγκεκριμένη συνάρτηση όλα τα subscription εκτός του geolocation γίνονται unsubscribe. Όλες οι ρυθμίσεις του χάρτη είναι ίδιες με αυτές τις κλάσης loadMap. Στο τέλος της συνάρτησης η MarkersSet καλείται. Σαν όρισμα δέχεται τα arrays MapCords και visitedInfo. Κατά την πρώτη κλήση το array visitedInfo είναι κενό. Μέσα στο σώμα τις συνάρτησης εκχωρούνται τα link κάποιον εικόνων σε μεταβλητές για να χρησιμοποιηθούν σαν markers. Τα ονόματα αυτά είναι Bimg,Mimg για τα beacon markers και markers αντίστοιχα. VBimg και VMimg για τα beacon markers και markers που ο χρήστης έχει ήδη επισκεφθεί. Επίσης υπάρχει η μεταβλητή Myimg με την εικόνα της θέσης του χρήστη. Στην μεταβλητή με όνομα MyMarker εκχωρείται ο constructor του google maps Marker και στην συνέχεια ορίζονται ο χάρτης που θα βρίσκεται και η εικόνα που θα χρησιμοποιεί. Ένα if statement ελέγχει εάν ο τύπος του σεναρίου είναι 'Chron' και ένα δεύτερο εάν το μέγεθος του MapCords είναι μικρότερο από το visitedInfo. Εάν ναι, σημαίνει ότι ο χρήστης δεν τους έχει επισκεφθεί όλα τα σημεία ενδιαφέροντος.

Μέσα στο if statement ένα ακόμα if ελέγχει εάν τα έξι πρώτα γράμματα όνομα του σημείου ενδιαφέροντος του τελευταία στοιχείο του MapCords σχηματίζουν την λέξη Marker. Εάν σχηματίζεται εκχωρείται στην μεταβλητή img η Bimg. Σε διαφορετική περίπτωση εκχωρείται η τιμή Mimg. Στην συνέχεια με την χρήση του constructor marker του google maps ορίζεται η θέση, ο χάρτης και η εικόνα του marker. Στην περίπτωση που το σενάριο δεν είναι του τύπου 'Chron' ένας βρόγχος for διατρέχει το array MapCords και με την χρήση του constructor marker του google maps ορίζεται το γεωγραφικό μήκος ,πλάτος και ο χάρτης που θα είναι. Με ένα ίδιο for statement ελέγχεται τον τύπο του marker και ορίζεται η κατάλληλη εικόνα.

Για τον marker που ο χρήστης έχει ήδη επισκεφθεί ένα if statement ελέγχει εάν το μέγεθος του visitedInfo είναι μεγαλύτερο του μηδενός. Εάν είναι, μια επανάληψη for διατρέχει το συγκεκριμένο array. Μέσα στο σώμα τις for ως κέντρο του χάρτη ορίζονται οι συντεταγμένες του marker που μόλις επισκέφθηκε και είναι ο τελευταίος στο array.Ο marker τοποθετείται στον χάρτη με τον ίδιο τρόπο όπως και οι υπόλοιπη markers. Επιπροσθέτως προστίθεται ένα double click event listener. Το listener καλείται η συνάρτηση FindData. Τα κατάλληλα εικονίδια προστίθενται όπως και στις προηγούμενες δύο περιπτώσεις. Η συνάρτηση FindData δέχεται σαν όρισμα το όνομα του marker στον οποίο κάνει double click ο χρήστης. Στην εν λόγω συνάρτηση το όνομα του marker εκχωρείται σε μία public μεταβλητή. Η τιμή stop στην μεταβλητή data και οι τιμές false στις τιμές Multiples και search. Ένας βρόγχος for διατρέχει το CordsArray και ένα if statement ελέγχει εάν το τρέχων όνομα του σημείου ενδιαφέροντος είναι ίδιο με αυτό που ψάχνει ο χρήστης. Εάν ναι, τα δεδομένα του σημείου ενδιαφέροντος εκχωρούνται στην μεταβλητή data και καλούνται οι συναρτήσεις setToNull και DisplayFile. Η συνάρτηση setToNull αδειάζει όλα τα array και μεταβλητές που χρησιμοποιούνται από τα σημεία ενδιαφέροντος.

```
this.MyMarker=newgoogle.maps.Marker({});
this.MyMarker.setMap(map);
this.MyMarker.setIcon(Myimg);



var markers;
varnum:number;
if(this.Scenario=="Chron"){
                    console.log("mphka",this.Scenario);
if(visitedInfo.length<MapCords.length){
num=visitedInfo.length;
                         console.log("num",num);
if(MapCords[num].PoiName.substring(0,6)!="Marker")
varimg=Bimg;
else
varimg=Mimg;
var marker=newgoogle.maps.Marker({

position:MapCords[num].latlng,
                        map: map,
                        icon:img


});
}
}
else{
for(leti=0;i<MapCords.length;i++){
          console.log("mphka",MapCords[i]);

          markers=newgoogle.maps.Marker({});

markers.setPosition(MapCords[i].latlng);
markers.setMap(map);
if(MapCords[i].PoiName.substring(0,6)!="Marker")
markers.setIcon(Bimg);
else
markers.setIcon(Mimg);


}
}


if(visitedInfo.length>0){
for(leti=0;i<visitedInfo.length;i++){
map.setCenter(visitedInfo[i].latlng);
console.log("mphkavisitedInfo ",visitedInfo[i]);
          markers=newgoogle.maps.Marker({});

markers.setPosition(visitedInfo[i].latlng);
markers.setMap(map);
markers.addListener('dblclick',()=>this.FindData(visitedInfo[i].visited));


if(visitedInfo[i].visited.substring(0,6)!="Marker")
markers.setIcon(VBimg);


else
markers.setIcon(VMimg);
}
}
```

Code30: Καθορισμό Markers

Πίσω στην συνάρτηση Position καλείται η συνάρτηση watch. Η συγκεκριμένη συνάρτηση κάνει χρήση του cordova plugin geolocation και της μεθόδου του watchPosition για να εντοπίζει συνεχώς τη θέση του χρήστη. Στη συνέχεια το γεωγραφικό μήκος και πλάτος εκχωρούνται σε μια μεταβλητή και η τοποθεσία του marker που δείχνει τη θέση του χρήστη στο χάρτη αναβαθμίζεται. Στην συνέχεια το καλούνται οι συναρτήσεις ScenarioType και startScanning στην κλάση BeaconPage. Η startScanning ψάχνει για τα πλησιέστερα beacons και επιστρέφει ένα object array με όνομα devices το οποίο περιέχει τα στοιχεία τους.

```
if(this.data!="stop"){
this.Multiples=false;
this.visited=undefined;


if(this.Scenario=="Chron"){

Data=newChronPage(this.navCtrl,this.navParams,this.af,this.ble).
startchron(this.accuracy,this.InumArray,this.CordsArray,
this.Currentlatlng,this.visitedInfo,this.devices,this.BIDs);

this.backgroundMode.enable();
this.backgroundMode.disableWebViewOptimizations();
if(Data !=undefined)
{
this.visited=this.visitedInfo[this.visitedInfo.length-1].visited;
this.data="stop";
this.search=false;
this.RingMyBell();
this.MarkersSet(this.visitedInfo,this.MapCords);
this.DisplayFile(Data);
}
```

Code31: εύρεση σημείων ενδιαφέροντος

Μέσα στο σώμα της συνάρτησης ScenarioType ένα if statement ελέγχει εάν η public μεταβλητή δεν έχει την τιμή stop. Σε περίπτωση που ισχύει το if statement εκχωρείται στην μεταβλητή Multiples εκχωρείται η τιμή false και στην visited η τιμή undefined. Ένα if statement ελέγχει εάν το σενάριο είναι τύπου 'Chron'. Εάν είναι καλείται η συνάρτηση startchron στην σελίδα ChronPage.

Η συνάρτηση startchron δέχεται σαν όρισμα το accuracy, InumArray με τους αύξοντες αριθμούς των σημείων ενδιαφέροντος του σεναρίου, την εκάστοτε θέση του χρήστη με όνομα Currentlatlng, το visitedInfo, devices και το array Bids. Στο σώμα της εν λόγω συνάρτησης μια επανάληψη for διατρέχει το CordsArray. Μέσα στην επανάληψη καλείται η συνάρτηση Nearest σαν ορίσματα δέχεται τα accuracy, InumArray, το εκάστοτε σημείο ενδιαφέροντος από το CordsArray με όνομα cord, Currentlatlng, visitedInfo, devices, BIDs. Η εν λόγω συνάρτηση με την χρήση της μεθόδου substr μετασχηματίζει κατάλληλα τις συντεταγμένες. Στη συνέχεια οι συντεταγμένες εκχωρούνται σε μια μεταβλητή ονόματος latlng. Ένα if statement ελέγχει εάν το πρώτο στοιχείο του InumArray είναι ίσο με τον αύξων αριθμό του στοιχείου του σημείου ενδιαφέροντος. Εάν είναι, δύο if statement ελέγχουν εάν οι τρέχοντες συντεταγμένες είναι ίδιες με αυτές του σημείου ενδιαφέροντος. Εάν είναι ίδιες, ένα ακόμα if statement ελέγχει εάν τα πρώτα έξι γράμματα σχηματίζουν την λέξη 'Marker'. Εάν ναι, καλείται η συνάρτηση CheckExistence με ορίσματα το όνομα του σημείου ενδιαφέροντος και το visitedInfo array.

Μέσα στο σώμα της συνάρτησης ένας βρόγχος for διατρέχει το array visitedInfo. Ένα if statement ελέγχει εάν το όνομα υπάρχει στο array με τα σημεία ενδιαφέροντος που ο χρήστης έχει ήδη επισκεφθεί. Εάν υπάρχει επιστρέφει το όνομα. Σε αντίθετη περίπτωση επιστρέφει την τιμή -1. Πίσω στην καλούσα συνάρτηση ένα if statement ελέγχει εάν η τιμή που επιστράφηκε είναι -1. Στην περίπτωση που είναι, καλείται η συνάρτηση MakeData. Η εν λόγω συνάρτηση δέχεται σαν όρισμα τα InumArray, visitedInfo, cord, latlng. Μέσα στο σώμα της η InumArray γίνεται shift, εκχωρείται σε ένα object τα στοιχεία latlng, το όνομα του στοιχείου ενδιαφέροντος με όνομα visited και εκχωρείται στο array visitedInfo. Πίσω στην συνάρτηση Nearest τα δεδομένα του σημείου ενδιαφέροντος εκχωρούνται σε μια μεταβλητή data και επιστρέφεται στην συνάρτηση startchron.

Εάν το σημείο ενδιαφέροντος δεν είναι του τύπου 'Marker'. Μια επανάληψη for διατρέχει το array devices. Ένα if statement ελέγχει εάν το id του τρέχοντος beacon υπάρχει στο visitedInfo array ή εάν όλο το array είναι undefined. Εάν ισχύει μία από τις δύο προϋποθέσεις ένα ακόμα if statement ελέγχει εάν το beacon id είναι ίδιο με το id του σημείου ενδιαφέροντος. Εάν είναι ίδιο καλείται η συνάρτηση MakeData. Το beacon id του σημείου ενδιαφέροντος εκχωρείται στο array BIDs και τα αρχεία του στην μεταβλητή data.

```
     Nearest(accuracy,InumArray,cord,Currentlatlng,visitedInfo,devices,BIDs){

var data;

varClat=Currentlatlng.lat().toString().substr(0,accuracy);
varClng=Currentlatlng.lng().toString().substr(0,accuracy);


varlat=cord.val().Lat.toString().substr(0,accuracy);
varlng=cord.val().Long.toString().substr(0,accuracy);
varlatlng=newgoogle.maps.LatLng(cord.val().Lat,cord.val().Long);

if(InumArray[0]==cord.val().Inum)
if(Clat==lat)
if(Clng==lng)

if(cord.key.substr(0,6)=="Marker"){
var value=this.CheckExistence(cord.key,visitedInfo);

if(value =="-1"){
                          console.log("visited:",value);
this.MakeData(InumArray,visitedInfo,cord,latlng);
var data=cord.val().Data;

return data;
}
}
```

Code 32: Αναζήτηση σημείων ενδιαφέροντος με χρονολογική σειρά

```
for(let device ofdevices){

if(visitedInfo.indexOf(device.id)===-1||visitedInfo==undefined)
if(cord.key== device.id)

{

this.MakeData(InumArray,visitedInfo,cord,latlng);

BIDs.push(cord.key);
var data=cord.val().Data;

returndata;

}
}
```

Code 33: Αναζήτηση για beacon devices σε σενάριο χρονολογικού τύπου

Στην συνάρτηση ScenarioType εάν ο τύπος του σεναρίου είναι ΄Geo' καλείται η συνάρτηση startgeo στην κλάση GeoPage η συγκεκριμένη συνάρτηση είναι ίδια με την startchron με την μόνη διαφορά ότι δεν έχει το όρισμα InumArray όπως επίσης δεν το έχει σαν όρισμα στην κλήση της συνάρτησης Nearest η οποία καλείται στο σώμα της. Στην συνάρτηση Nearest εκχωρείται η τιμή -350 στην μεταβλητή minRssi. Οι συντεταγμένες μετασχηματίζονται κατάλληλα και εκχωρούνται στην μεταβλητή latlng. Ο έλεγχος για το εάν οι τρέχοντες συντεταγμένες είναι ίδιες με του σημείου ενδιαφέροντος και εάν είναι τύπου marker γίνονται με τον ίδιο τρόπο όπως στην συνάρτηση Nearest της κλάσης ChronPage. Στην περίπτωση που το σημείο ενδιαφέροντος δεν είναι τύπου 'Marker' μια επανάληψη for διατρέχει το array devices. Για κάθε ένα από τα beacon καλείται η συνάρτηση CheckExistence. ΄΄Ενα if statement ελέγχει εάν το id του τρέχοντος beacon υπάρχει στο visitedInfo array ή εάν όλο το array είναι undefined . 'Εάν κάποιο από τα δύο ισχύει ένα if statement ελέγχει εάν το όνομα του σημείου ενδιαφέροντος είναι ίδιο με το id του beacon. Εάν είναι ένα εμφωλευμένο if statement ελέγχει εάν το minRssi είναι μικρότερο του Rssi του beacon

Εάν είναι σημαίνει ότι το συγκεκριμένο beacon είναι το κοντινότερο από όλα και το Rssi του εκχωρείται στην μεταβλητή minRssi. Στη συνέχεια τα δεδομένα του σημείου ενδιαφέροντος εκχωρούνται στην μεταβλητή data. Σε ένα object εκχωρούνται σαν στοιχεία το latlng και το όνομα του σημείου ενδιαφέροντος. Εκτός του βρόγχου ένα if statement ελέγχει εάν το object που δημιουργήθηκε προηγουμένως δεν είναι undefined. Εάν δεν είναι εκχωρείται σε ένα array με όνομα visitedInfo και το όνομα του σημείου ενδιαφέροντος σε ένα array με όνομα BIDs. Ο λόγος που το object εκχωρείται στο array εκτός του βρόγχου for είναι γιατί μπορεί να υπάρξουν πολλά beacons με πιο μικρό Rssi από το προηγούμενο στο object array που θα μπουν στο σώμα της if. Στην περίπτωση που το σημείο ενδιαφέροντος είναι τύπου 'Marker. Καλούνται οι συναρτήσεις CheckExistence και MakeData οι οποίες είναι ίδιες με αυτές στην κλάση ChronPage και τα δεδομένα του σημείου ενδιαφέροντος εκχωρούνται στην μεταβλητή data. Η συνάρτηση Nearest επιστρέφει την μεταβλητή data στην συνάρτηση startgeo.

```
RingMyBell(){

this.nativeStorage.getItem('ring')
.then(
    data =>{console.log("data",data);
this.play(data);

},
    error =>{console.error("error",error);

vardt="file:///android_asset/www/audio/ring.mp3";
this.play(dt);
});
}
```

*Code34:* Αναπαραγωγή ήχου ειδοποίησης

Στις συναρτήσεις startchron και startgeo επιστρέφεται η μεταβλητή data από την συνάρτηση Nearest. Ένα if statement ελέγχει εάν η μεταβλητή δεν είναι undefined. Εάν δεν είναι επιστρέφεται στην συνάρτηση ScenarioType. Στην συγκεκριμένη συνάρτηση και για τις δύο περίπτωσης τύπου σεναρίου ένα if statement ελέγχει εάν η τιμή που επιστράφηκε δεν είναι undefined. Εάν δεν είναι εκχωρείται σε μία μεταβλητή ονόματος visited το τελευταίο ψηφίο του array visitedInfo. Στην μεταβλητή search εκχωρείται η τιμή false και στην public μεταβλητή data το string ‘stop’.Στη συνέχεια καλείται η συνάρτηση RingMyBell. Η εν λόγω συνάρτηση κάνει χρήση του cordova plugin nativeStorage και της μεθόδου getItem για να ανακτήσει το αρχείο ήχου ειδοποιήσεις. Ανεξαρτήτου περίπτωσης error ή επιτυχής ανακτήσεις θα γίνει κλήση της συναρτήσεις play. Στην περίπτωση error θα γίνει χρήση του εξ ορισμού ήχου ειδοποίησης από το μονοπάτι του στην συσκευή. Η συνάρτηση play κάνει χρήση του cordova plugin media και της μεθόδου create με όρισμα τον ήχο ειδοποιήσεις για την δημιουργία ενός ηχητικού κομματιού. Επίσης κάνει χρήση τριών observables για τις περιπτώσεις error, επιτυχίας και αλλαγής της κατάστασης του ηχητικού κομματιού. Τέλος με την χρήση της μεθόδου play το αρχείο αναπαράγεται. Πίσω στην συνάρτηση καλούνται οι συναρτήσεις MarkersSet και DisplayFile.

```
        Nearest(accuracy,cord,Currentlatlng,visitedInfo,devices,BIDs){

varminRssi=-350;
if(accuracy ==undefined)
          accuracy=6;

varClat=Currentlatlng.lat().toString().substr(0,accuracy);
        console.log("Clat",Clat);
varClng=Currentlatlng.lng().toString().substr(0,accuracy);
        console.log("Clng",Clng);

varlat=cord.val().Lat.toString().substr(0,accuracy);
varlng=cord.val().Long.toString().substr(0,accuracy);

varlatlng=newgoogle.maps.LatLng(cord.val().Lat,cord.val().Long);
if(Clat==lat)
if(Clng==lng)

if(cord.key.substr(0,6)!="Marker"){

for(let device ofdevices){

if(newChronPage(this.navCtrl,this.navParams,this.af,this.ble).
CheckExistence(device.id,visitedInfo)==="-1"||visitedInfo==undefined)

if(cord.key== device.id)
if(minRssi<device.rssi){
minRssi=device.rssi;
var data=cord.val().Data;

var temp ={latlng:latlng,
visited:cord.key
};

}
}
```

Code 35: Αναζήτηση για σημεία ενδιαφέροντος σε σενάριο χρονολογικού τύπου

```
if(temp !=undefined)
visitedInfo.push(temp);
BIDs.push(cord.key);

}
else{// if(cord.key.substr(0,6) != "Marker"

var value=newChronPage(this.navCtrl,this.navParams,this.af,this.ble).

CheckExistence(cord.key,visitedInfo);

if(value =="-1"){

this.MakeData(visitedInfo,cord,latlng);
var data=cord.val().Data;


}
}

returndata;
```

Code 36: Αναζήτηση για Beacon σημεία ενδιαφέροντος σε σενάριο γεωγραφικού τύπου

Η συνάρτηση StopSearch καλείται για να σταματήσει την αναζήτηση σημείων ενδιαφέροντος. Μέσα στο σώμα της εκχωρείται στην μεταβλητή data το string 'stop'. Καλείται η συνάρτηση setToNull και εκχωρείται η τιμή false στην μεταβλητή search. Η συνάρτηση FauxRefresh ξεκινάει την αναζήτηση σημείων ενδιαφέροντος. Εκχωρεί τις τιμές false, true και 'cnt' στις μεταβλητές Mulitples, search και data αντίστοιχα. Επίσης καλεί τις συναρτήσεις setToNull και ScenarioType. Για τα ερωτήματα πολλαπλής καλείται η συνάρτηση Response. Στο σώμα της συνάρτησης δημιουργείται ένα modal στην συνάρτηση ResponsePage με παραμέτρους το μοναδικό id του σεναρίου, GamePath, το μονοπάτι στο σημείο ενδιαφέροντος με όνομα μεταβλητής dbPath, το array MCvisited και το όνομα του τελευταίου σημείου ενδιαφέροντος που επισκέφθηκε ο χρήστης με όνομα μεταβλητής CurrentPoi.

Στην συνάρτηση ngOnInit της ResponsePage κλάσης ένα if statement ελέγχει εάν το array MCvisited που κατά την ανάκτηση πήρε το όνομα visited είναι διαφορετικό του μηδενός. Εάν είναι ένας βρόγχος for διατρέχει το array visited. Μέσα στο σώμα της for ένα if statement ελέγχει εάν το όνομα του τρέχοντος σημείου ενδιαφέροντος είναι ίσο με το όνομα κάποιου σημείου ενδιαφέροντος που ο χρήστης έχει ήδη επισκεφθεί. Τα στοιχείο MultipleChoicesAnswers και MultipleChoices του array εκχωρούνται στις μεταβλητές MultipleChoicesAnswers και MultipleChoices αντίστοιχα. Η τιμή true εκχωρείται στην μεταβλητή isAlreadyAnsweres και καλείται η συνάρτηση CheckAnswers. Ένα if statement ελέγχει εάν η μεταβλητή isAlreadyAnsweres είναι false. Το συγκεκριμένο κομμάτι κώδικα ελέγχει εάν το ερώτημα πολλαπλής επιλογής έχει ελεγχθεί ήδη. Εάν είναι καλείται η συνάρτηση MCstaff.

Η συγκεκριμένη συνάρτηση κάνει κλήση της συνάρτησης RetrieveKey στην κλάση FinderPage η συνάρτηση επιστρέφει ένα array με τους τίτλους των ερωτήσεων πολλαπλής επιλογής. Μία επανάληψη for διατρέχει το array και για κάθε στοιχείο/τίτλο καλεί την συνάρτηση

RetrivePosition της FinderPage. Η συνάρτηση επιστρέφει τις πιθανές απαντήσεις και την σωστή απάντηση από την βάση δεδομένων. Ένας βρόγχος for διατρέχει το array MultipleChoices που επέστρεψε η συνάρτηση. Μέσα στο σώμα της επανάληψης ένα if statement ελέγχει εάν το πρώτο γράμμα του key είναι το 'C'. Εάν είναι, εκχωρεί την τιμή val του answer που αντιστοιχεί σε μια απάντηση στην μεταβλητή CorrectAnswer. Σε αντίθετη περίπτωση η τιμή val εκχωρείται σε ένα array ονόματος answers και η τιμή false στα array isCorrectAnswer, isCorrectAnswer. Στη συνέχεια εκτός του βρόγχου δημιουργείται ένα object με παραμέτρους τα στοιχεία Answer σαν κενό string, τίτλο και CorrectAnswer του ερωτήματος πολλαπλής επιλογής. Το αντικείμενο εκχωρείται στο array MultipleChoicesAnswers. Ακόμα ένα αντικείμενο δημιουργείται με παράμερους τα στοιχεία τον τίτλο και τις απαντήσεις του ερωτήματος πολλαπλής επιλογής και τα arrays isCorrectAnswer, isYourAnswer. Το αντικείμενο επίσης εκχωρείται σε ένα array με όνομα MultipleChoices.

```
varisAlreadyAnsweres;
if(visited.length!=0){

for(let key of visited){
if(key.Name==CurrentPoi){
this.MultipleChoicesAnswers=key.MultipleChoicesAnswers;
this.MultipleChoices=key.MultipleChoices;
this.isAlreadyAnsweres=true;
this.CheckAnswers();
}

}
}
if(this.isAlreadyAnsweres==false){

this.MCstaff();

}
```

*Code 37 Έλεγχος εάν το ερώτημα πολλαπλής επιλογής έχει ελεγχθεί ήδη*

Η συνάρτηση CheckAnswers καλείται για τον έλεγχο της απαντήσεις του χρήστη. Αρχικά ένας βρόγχος for διατρέχει το array MultipleChoicesAnswers. Στο σώμα της επανάληψης ένα if statement ελέγχει εάν η απάντηση του χρήστη είναι ίδια με την σωστή. Εάν είναι ένας βρόγχος for εμφωλευμένος στον πρώτο διατρέχει το array MultipleChoices. Μέσα στον συγκεκριμένο βρόγχο ένας τρίτος βρόγχος διατρέχει το array Answers του object array MultipleChoices. Ένα if statement ελέγχει εάν κάποια από τις απαντήσεις του ερωτήματος είναι η σωστή. Όταν την βρει εκχωρεί την τιμή true στην αντιστοιχεί θέση του array isCorrectAnswer. Στην περίπτωση που η απάντηση του χρήστη δεν είναι σωστή δύο βρόγχοι for ο πρώτος εμφωλευμένος στον δεύτερο διατρέχουν τα array MultipleChoices και Answers του MultipleChoices object array αντίστοιχα. Ένα if statement ελέγχει εάν η τρέχουσα απάντηση είναι ίδια με την σωστή. Εάν είναι, εκχωρεί την τιμή true στην ίδια θέση array isCorrectAnswer. Τέλος ένα δεύτερο if statement ελέγχει εάν η τρέχουσα απάντηση είναι ίδια με του χρήστη. Σε περίπτωση που είναι επισημαίνετε επίσης η θέση της εκχωρώντας την τιμή true στην αντιστοιχεί θέση του array isYourAnswer.

```
CheckAnswers(){



for(let key ofthis.MultipleChoicesAnswers){

if(key.Answer==key.CorrectAnswer){

for(let answer ofthis.MultipleChoices){

for(leti=0;i<answer.Answers.length;i++){
if(answer.Answers[i]==key.CorrectAnswer){

answer.isCorrectAnswer[i]=true;

}

}
}


}else{
for(let answer ofthis.MultipleChoices){

for(leti=0;i<answer.Answers.length;i++){

if(answer.Answers[i]==key.CorrectAnswer){
answer.isCorrectAnswer[i]=true;
}
if(key.Answer==answer.Answers[i]){
answer.isYourAnswer[i]=true;

}

}

}
```

Code38: Έλεγχος απαντήσεων σε ερωτήματα πολλαπλής επιλογής

Κατά το dismiss του modal τα arrays MultipleChoices και MultipleChoicesAnswers εκχωρούνται σαν παράμετροι στο object result και επιστρέφονται μαζί με την μεταβλητή MultipleChoices στην συνάρτηση Response. Μέσα στο onDidDismiss του modal ένα if statement ελέγχει εάν η τιμή MultipleChoices που επιστράφηκε σαν check είναι ίση με 'false'. Εάν είναι σημαίνει ότι το result array επιστράφηκε για πρώτη φόρα. Σε ένα array ονόματος MCvisited εκχωρούνται σαν στοιχεία το όνομα του τρέχοντος σημείου ενδιαφέροντος με όνομα Name, τα arrays MultipleChoices και MultipleChoicesAnswers. Στη συνέχεια καλείται η συνάρτηση FirebaseMultipleChoiceUpload στην κλάση FirePage. Σαν παραμέτρους δέχεται το MultipleChoicesAnswers array, το μονοπάτι οπού βρίσκονται τα δεδομένα του χρήστη στην βάση δεδομένων και το μονοπάτι στα δεδομένα του σημείου ενδιαφέροντος. Στην συνάρτηση FirebaseMultipleChoiceUpload μια for επανάληψη διατρέχει το MultipleChoicesAnswers array το οποίο ανακτήθηκε σαν clicked. Για κάθε απάντηση του χρήστη προστίθεται μια εγγραφή στην βάση δεδομένων με την χρήση του firebase object και της set μεθόδου του. Εκτός της επανάληψης επίσης με την χρήση του firebase object προστίθεται το μονοπάτι του σημείου ενδιαφέροντος.

Ο κώδικας για την διάδραση του χρήστη όσα αφορά φωτογραφία και video είναι παρόμοιος με αυτόν για την εφαρμογή του admin. Στην περίπτωση φωτογραφίας γίνεται κλήση της συνάρτησης GetPhoto. Στην συγκεκριμένη συνάρτηση της OpenCam στην κλάση OpenCamPage. Στην OpenCam καλείται η συνάρτηση Cam στην CamPage η οποία είναι ίδια με την αντίστοιχη συνάρτηση στην εφαρμογή του admin. Στη συνέχεια καλείται η συνάρτηση CameraData στην κλάση StringManipulationPage. Η συγκεκριμένη κλάση δέχεται σαν όρισμα τα στοιχεία της φωτογραφίας και με την χρήση της replace αντικαθιστά το όνομα του αρχείου από το path με ένα κενό string. Η συνάρτηση επιστρέφει ένα αντικείμενο ονόματος CameraDataInfo με στοιχεία το όνομα του αρχείου και το μονοπάτι του φακέλου που βρίσκεται το αρχείο, στην συνάρτηση OpenCam. Στη συνέχεια γίνεται κλήση της συνάρτηση dataUrl που βρίσκεται επίσης στην κλάση CamPage οποία επιστρέφει το αρχείο σε μορφή data URL και το επιστρέφει στην συνάρτηση GetPhoto όπου εκχωρείται σε ένα array ονόματος Images.

Στην περίπτωση του video γίνεται κλήση της συνάρτησης GetVideo. Στην συγκεκριμένη συνάρτηση καλείται η συνάρτηση OpenVid η οποία βρίσκεται στην κλάσης OpenCamPage. Η εν λόγω συνάρτηση καλεί την συνάρτηση vid στην κλάση VidPage η οποία συνάρτηση επιστρέφει τα στοιχεία του video. Στην συνέχεια εξάγεται το όνομα του thumbnail με την χρήση του substring για την αφαίρεση της κατάληξης. Για την δημιουργία του thumbnail καλείται η συνάρτηση MakeVideoThumnail η οποία καλεί τις συναρτήσεις VideoThumbnail στην κλάση VidPage, ThumbdataUrl στην CamPage , AddPlayIcon και επιστρέφει το thummbail στην OpenVid. Οι τρείς αυτές συναρτήσεις είναι ίδιες με αυτές που χρησιμοποιούνται για την δημιουργία του σεναρίου. Στη συνέχεια καλούνται η CameraData και dataUrl. Η OpenVid επιστρέφει στην καλούσα ένα object με το video και το thumbnail του. Το οποίο object εκχωρείται σε ένα array ονόματος Videos.

Στην περίπτωση των δεδομένων από το σύστημα αρχείων της συσκευής καλείται η συνάρτηση GetData. Η συγκεκριμένη συνάρτηση καλεί την συνάρτηση Open στην σελίδα OpenCamPage. Η συνάρτηση Open κάνει χρήση των cordova plugin fileChooser για να απόκτηση ο χρήστης πρόσβαση στο σύστημα αρχείων της συσκευής και filePath για το resolve του content URI σε native. Η συνάρτηση επιστρέφει το URI στην καλούσα. Το εν λόγω URI δίνεται ως όρισμα στην κλάση CheckType της κλάσης StringManipulationPage για την εξαγωγή του ονόματος του αρχείου, του μονοπατιού του φακέλου που βρίσκεται το αρχείο και την κατάληξη του. Στη συνέχεια καλείται η συνάρτηση dataURL και η DataType. Η συνάρτηση DataType δέχεται σαν όρισμα το αντικείμενο που επέστρεψε η συνάρτηση CheckType με όνομα DataInfo, το data URL αρχείο με όνομα FileEntry

και το native URL filePath. Η λειτουργία όλων των συναρτήσεων είναι ίδια με τις αντίστοιχες στην τροποποίηση σεναρίου.

Η συνάρτηση DataType με την χρήση ενός switch statement ελέγχει την κατάληξη του αρχείου. Στις περιπτώσεις που το αρχείο είναι τύπου εικόνας (jpg) ή ήχου(mp3) εκχωρείται στα array Images και Audio. Στην περίπτωση video (mp4) γίνεται δημιουργείται ένα object με στοιχεία το αρχείο video και το thumbnail του. Το object εκχωρείται σε ένα array ονόματος video. Στην περίπτωση που το αρχείο είναι οποιοδήποτε άλλου τύπου εκτός των τριών επιτρεπτών (mp4, mp3, jpg) εμφανίζεται κατάλληλο μήνυμα λάθους. Σε όλες τις περιπτώσεις διάδρασης του χρήστη είτε για φωτογραφία/video ή αρχείο από το σύστημα αρχείων καλείται η συνάρτηση upload. Η συνάρτηση upload δημιουργεί κατάλληλα τα StoragePath και το μονοπάτι του σημείου ενδιαφέροντος για τα δεδομένα διάδρασης του χρήστη με όνομα dbPath. Στη συνέχεια καλείται η συνάρτηση FirebaseResponseUpload. Σαν παραμέτρους δέχεται τα dbPath, τα arrays Videos, Images, Audio, το id, το StoragePath καθώς και τις public μεταβλητές Icnt, Vcnt, Acnt. Η FirebaseResponseUpload συνάρτηση είναι παρόμοια με την FirebaseUpload. Η διαφορά τους είναι ότι δεν υπάρχουν δεδομένα όπως ο τύπος του σεναρίου, ο αύξων αριθμός και accuracy για να γίνουν upload. Καθώς και ότι επιστρέφουν ένα objet ονόματος counters για τους μετρητές Icnt, Vcnt, Acnt οι οποίοι αντιστοιχούν σε εικόνα, video και ήχο αντίστοιχα. Οι μετρητές αυξάνονται μετά από κάθε κλήση της SetResponseParams και εκχωρούνται στις public μεταβλητές.

```
DataType(DataInfo,FileEntry,filePath){

switch(DataInfo.Type){

case"mp4":

newOpenCamPage(this.filePath,this.fileChooser,this.navParams,this.video
Editor,this.mediaCapture,this.file,this.modalCtrl).MakeVideoThumnail(fi
lePath,DataInfo.Name).then(thumbnail =>{
vartmp={ video:FileEntry,
thumbnail : thumbnail};
this.Videos.push(tmp);
})

break;
case"jpg":
this.Images.push(FileEntry);
break;
case"mp3":
this.Audio.push(FileEntry);
        console.log("Audio",this.Audio);
break;
default:
var subtitle="Only mp4, jpeg and mp3 are allowed";
this.CheckAlert(subtitle);

}
```

Code39: switch statement για τον έλεγχο κατάληξης των αρχείων

Για τις ρυθμίσεις της εφαρμογής χρησιμοποιείται η κλάση SettingsPage. Η κλάση Open καλείται για να επιλέξει ο χρήστης ένα νέο ήχο ειδοποιήσεις από το σύστημα αρχείων. Η συνάρτηση κάνει χρήση των cordova plugins fileChooser, filePath και nativeStorage. Στο σώμα της συνάρτησης γίνεται χρήση του Instance clear του plugin nativeStorage για να καθαρίσει ότι είχε αποθηκεύσει. Τα fileChooser και resolveNativePath για την πρόσβαση στο σύστημα αρχείων και του resolve του content URI. Επίσης χρησιμοποιεί το setItem instance του nativeStorage για να αποθηκεύσει το ηχητικό κομμάτι. Για την έξοδο από την εφαρμογή γίνεται χρήση της συνάρτησης exit της κλάσης SettingsPage. Η συνάρτηση αυτή κάνει unsubscribe το watchPosition του geolocation το οποίο εκχωρήθηκε σαν δεδομένο στην κλάση. Σε μία μεταβλητή εκχωρείται η active σελίδα από το navigation controller και με την χρήση της remove μεθόδου και ορίζει την CheckPage σαν root σελίδα.

# Κεφάλαιο 4

# Αξιολόγηση

Για την αξιολόγηση της εφαρμογής πραγματοποιήθηκαν αρκετές δοκιμές σε εξωτερικό χώρο. Όλες οι δοκιμές έγιναν από εμένα τον developer. Δυστυχώς δεν ήταν δυνατών να γίνουν δοκιμές με πολλά άτομα, διαφορετικές συσκευές και εκδόσεις λειτουργικού Android. Οπότε η αξιολόγηση δεν μπορεί να θεωρηθεί ολοκληρωμένη. Δοκιμές έγιναν για όλα τα είδη σεναρίου, markers και συνημμένων δεδομένων. Σενάρια με μόνο απλά markers ή beacon markers.

Τόσο η εφαρμογή του χρήστη όσο και η εφαρμογή του admin είναι εύκολο να γίνουν build σε μία συσκευή μέσω του Ionic framework ή να γίνουν install σε μια συσκευή μέσα από το .apk αρχείο των εφαρμογών. Υπάρχει στα κεφάλαια δύο και τρία οδηγοί για την κατανόηση του κώδικα και της χρήσης των εφαρμογών. Η δομή του κώδικα είναι κατά την γνώμη μου κάπως περίπλοκη αλλά με την χρήση του οδηγού αποσαφηνίζεται η λειτουργία του. Οι δύο αυτές εφαρμογές μπορούν να γίνουν build σε διαφορετικές πλατφόρμες λόγω του ότι είναι υβριδικές. Όμως τα κομμάτια κώδικα που γίνεται χρήση του συστήματος αρχείων θα χρειαστούν κάποια αλλαγή για να είναι πλήρως συμβατά με την εκάστοτε πλατφόρμα.

Η εφαρμογή του χρήστη με ενεργοποιημένο το high accuracy για το GPS δίνει μια καλή απόκριση όσο αναφορά τη θέση του marker στον χάρτη και την πραγματική θέση της συσκευής. Σε αντίθετη περίπτωση υπάρχει απόκλιση αρκετών μέτρων μεταξύ αυτών των δύο. Η εφαρμογή υστερεί σε γραφικό περιβάλλον όσο αναφορά την απεικόνιση των συνημμένων δεδομένων των σημείων ενδιαφέροντος. Επιπροσθέτως υπάρχει απόκλιση μεταξύ του χρόνου του οποίου χρειάζεται η εφαρμογή για να εντοπίσει ένα beacon marker από ότι έναν απλό marker. Ο χρόνος για έναν απλό marker είναι μικρός και μπορώ να πω αποδεκτός. Για ένα beacon marker μπορεί να υπάρξει καθυστέρηση από μικρή (ένα δευτερόλεπτο) έως μερικά δευτερόλεπτα. Η διαφορά αυτή εξαρτάται από το εάν η εφαρμογή έχει ήδη εντοπίσει το beacon πριν ο χρήστης εισέλθει στο γεωγραφικό χώρο που έχει ορισθεί το συγκεκριμένο σημείο ενδιαφέροντος.

Υπάρχει καθυστέρηση στην περίπτωση που δεν είναι διαθέσιμο κάποιο ασύρματο δίκτυο και ο χρήστης χρησιμοποιεί τα δεδομένα της συσκευής. Η συγκεκριμένη καθυστερήσει αφορά το χρόνο που χρειάζεται η εφαρμογή να ξεκινήσει, του χρόνου για την εμφάνιση των συνημμένων δεδομένων του εκάστοτε σημείου ενδιαφέροντος στην συσκευή και της διάδρασης του χρήστη. Επιπροσθέτως η χρήση των δεδομένων σε συνδυασμό με την high accuracy λειτουργία του GPS αλλά και του Bluetooth έχουν σαν αποτέλεσμα την γρήγορη κατανάλωση της μπαταρίας της συσκευής. Επίσης παρατηρήθηκε ότι όταν ο marker της θέσης του χρήστη διασταυρωθεί με κάποιο marker επάνω στον χάρτη μένει εκεί με αποτέλεσμα να υπάρχουν δύο markers θέσης πάνω στον χάρτη. Πολλές φορές δεν αλλάζει το χρώμα του marker όταν ο χρήστης επισκεφθεί το σημείο ενδιαφέροντος που αντιπροσωπεύει. Η εφαρμογή του admin λειτουργεί χωρίς κανένα πρόβλημα εκτός των καθυστερήσεων από την χρήση των δεδομένων της συσκευής.

# Συμπεράσματα

Μέσα από τις υβριδικές εφαρμογές είναι δυνατών μια εφαρμογή να είναι διαθέσιμη σε όλες τις μεγάλες πλατφόρμες με μικρό κόστος αυξάνοντας τον πιθανό αριθμό χρηστών της. Από την πλευρά του developing είναι ευκολότερο λόγω του ότι γίνεται χρήση HTML, CSS και Javascript οι οποίες είναι ευρέως γνωστές. Επίσης το Ionic framework και τα εργαλεία της Google είναι πολύ straight forward κάνοντας  εύκολο το να φτιάξει κάποιος μια απλή υβριδική εφαρμογή. Η εφαρμογή μπορεί να έχει μια πληθώρα από δυνατές χρήσης λόγω της όλης ιδέας με τα σενάρια. Ο καθένας μπορεί να δημιουργήσει ένα δικό του με διαφορετική θεματολογία. Η θεματολογία μπορεί να ποικίλει από ψυχαγωγικό, ενημερωτικό μέχρι και εκπαιδευτικό χαρακτήρα.

# Βιβλιογραφία


1. https://developer.telerik.com/featured/what-is-a-hybrid-mobile-app
2. https://www.tdktech.com/tech-talks/mobile-development-web-vs-hybrid-vs-native
3. https://developer.telerik.com/featured/what-is-a-webview/
4. https://cordova.apache.org/docs/en/latest/guide/overview/index.html
5. https://www.brightcove.com/en/blog/2011/11/html5-and-rise-hybrid-apps
6. https://angular.io/guide/architecture
7. https://kontakt.io/beacon-basics/what-is-a-beacon/
8. https://myshadesofgray.wordpress.com/2014/04/15/hybrid-applications-and-android-native-browser/
9. https://firebase.google.com/docs/web/setup?authuser=0
10. https://firebase.google.com/docs/database/web/read-and-write?authuser=0
11. https://github.com/angular/angularfire2
12. https://firebase.google.com/docs/storage/web/download-files
13. https://github.com/don/cordova-plugin-ble-central#advertising-data
14. https://angular.io/guide/reactive-forms
15. https://firebase.google.com/docs/storage/web/create-reference
16. https://github.com/katzer/cordova-plugin-background-mode
17. https://developers.google.com/maps/documentation/javascript/reference#LatLng
18. https://hashnode.com/post/what-are-the-core-differences-between-promises-and-observables-ciqqyzak806fahw53lwkumqac
19. https://stackoverflow.com/questions/37364973/angular-promise-vs-observable
20. https://angular-2-training-book.rangle.io/handout/observables/observables_vs_promises.html
21. https://hackernoon.com/understanding-creating-and-subscribing-to-observables-in-angular-426dbf0b04a3
22. https://developer.telerik.com/topics/web-development/introduction-observables-angular-developers/
23. https://ionicframework.com/docs//intro/tutorial/navigation/
24. Nikos Pinikas, Spyros Panagiotakis, Despina Athanasaki, Athanasios G. Malamos, A Device Independent Platform for Synchronous Internet of Things Collaboration and Mobile Devices Screen Casting, International Journal of Information and Communication Sciences. Vol. 2, No. 5, 2017, pp. 59-67. doi: 10.11648/j.ijics.20170205.12
25. Panagiotakis, K. Kapetanakis, A. G. Malamos, "Architecture for Real Time Communications over the Web," International Journal of Web Engineering, 2(1): 1-8, doi:10.5923/j.web.20130201.01, 2013.