Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη Native Android εφαρμογής με Image Labelling** **Development of a Native Android application for Image Labelling**
Ονοματεπώνυμο Φοιτητή	**Χρήστος Μπάκας**
Πατρώνυμο	**Ανδρέας**
Αριθμός Μητρώου	**ΜΠΣΠ 17051**
Επιβλέπων	**Αλέπης Ευθύμιος Αναπληρωτής καθηγητής**

Ημερομηνία Παράδοσης **Οκτώβριος 2020**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Αλέπης Ευθύμιος	Βίρβου Μαρία	Πατσάκης Κωνσταντίνος
Αναπληρωτής Καθηγητής	Καθηγήτρια	Επίκουρος Καθηγητής

Ευχαριστίες

Θα ήθελα αρχικά να ευχαριστήσω τον επιβλέπων καθηγητή κ. Ευθύμιο Αλέπη για την υποστήριξη και την καθοδήγηση καθ' όλη τη διάρκεια αυτής της διπλωματικής εργασίας. Επίσης θα ήθελα να ευχαριστήσω όσους συνέβαλαν στη δοκιμή της εφαρμογής για τη στήριξη και την άμεση ανταπόκρισή τους.

Περιεχόμενα

Περίληψη

Η μεταπτυχιακή εργασία αναπτύχθηκε στα πλαίσια του μεταπτυχιακού προγράμματος «Προηγμένα Συστήματα Πληροφορικής» του τμήματος Πληροφορικής του Πανεπιστημείου Πειραιώς. Ο στόχος της διατριβής είναι να δημιουργήσει μια εφαρμογή native android η οποία θα αξιοποιεί το ML Kit της Firebase και θα κάνει image labelling αναλύοντας τα αντικείμενα που υπάρχουν μέσα στην εικόνα.

Ο χρήστης εγγράφεται στην εφαρμογή δίνοντας mail και κωδικό. Θα μπορεί να βγάλει φωτογραφίες μέσω της εφαρμογής αλλά και να χρησιμοποιήσει εικόνες που εχει τραβήξει στο παρελθόν και είναι αποθηκευμένες στη συσκευη του. Στη συνέχεια αποθηκεύεται online ένα μοντέλο για τον χρήστη με τα labels που εμφανίζονται με μεγαλύτερη συχνότητα. Ταυτόχρονα αποθηκεύεται και το geolocation, το οποίο σε συνδυασμό με τα labels και τις εικόνες αποθηκεύονται σε cloud. Χρήσιμοποιώντας όλα αυτά τα στοιχεία δημιουργούμε markers στο χάρτη με τα πιο «διάσημα» labels ανά περιοχή.

Η εφαρμογή αναπτύχθηκε σε Android Studio με χρήση της Firebase (Google, Firebase, n.d.).

Abstract

This project is a post graduate dissertation of the MSc Advanced Information Systems of the department of Informatics of the University of Piraeus. The purpose of this project is to create a native android application that will use Firebase’s ML Kit for image labelling on pictures.

The user registers on the application giving email and password. After that he will be able to take pictures through the application or use pictures he has already stored on his device. An online model is saved online for each user with the labels that appear more often. On the same time the geolocation, among with the labels and the pictures, are all stored on cloud. Using this data, we create markers on the map with the most famous labels of the area.

The project was developed with the use of Android Studio and Firebase.

Εισαγωγή

Η εξέλιξη της τεχνολογίας γίνεται με πολύ γοργούς ρυθμούς τα τελευταία χρόνια. Ζούμε στην εποχή της πληροφορίας και της ταχύτητας όπου πράγματα τα οποία δε φανταζόμασταν ότι μπορεί να χρησιμοποιούμε μας είναι πια απαραίτητα.

Η ανάπτυξη των "έξυπνων" τηλεφώνων (smartphones), τα οποία χρησιμοποιούμε όλοι μας, έφερε στην τσέπη του κάθε ανθρώπου ένα φορητό υπολογιστή αρκετά γρήγορο και πιο "ικανό" από ένα κλασσικό υπολογιστή. Εκτός από τη χρήση ως τηλέφωνο, τα smartphones μπορούν να συνδέονται στο διαδίκτυο και από εκεί να έχουν πρόσβαση σε αμέτρητες πληροφορίες αλλά και να αλληλεπιδρούν με άλλες συσκευές που βρίσκονται στο διαδίκτυο.

Καθώς η τεχνολογία εξελίσεται φαίνεται πως υπάρχει ανάγκη για πιο εξατομικευμένες υπηρεσίες οι οποίες θα βοηθούν το χρήστη να βρίσκει πιο εύκολα αυτό που ζητά και του ταιριάζει. Ο στόχος της παρούσας διπλωματικής εργασίας είναι να δημιουργήσει την εφαρμογή PlexCam η οποία θα χρησιμοποιεί φωτογραφίες από αντικείμενα που έχουν τραβήξει το ενδιαφέρον του χρήστη να τα αναγνωρίζει και στη συνέχεια να τα τοποθετεί πάνω σε ένα χάρτη. Ταυτόχρονα θα δημιουργείται και ένα προφίλ χρήστη το οποίο θα εμπλουτίζεται δυναμικά από τις φωτογραφίες που θα προστίθενται στη βάση.

1. Κεφάλαιο: Κινητό τηλέφωνο – Λειτουργικό σύστημα Android

1.1Κινητό τηλέφωνο

Το κινητό τηλέφωνο ήταν αποτέλεσμα της ανάγκης για καλύτερη επικοινωνία. Μετά τον Β' Παγκόσμιο Πόλεμο πολλοί προσπάθησαν να το κατασκευάσουν ανεπιτυχώς. Το 1973 ο δόκτωρ Μάρτιν Κούπερ της Motorola χρησιμοποίησε πρώτος το MotorolaDynaTAC. Με βάρος 900 γραμμάρια και ύψος 25 εκατοστά, το Dynatac ήταν το πρώτο κινητό τηλέφωνο που κατασκευάστηκε ποτέ και χρειάστηκαν 10 χρόνια για να βγει στην παραγωγή.

Με την κατασκευή του πρώτου αυτοματοποιημένου δικτύου κινητής τηλεφωνίας στις αρχές της δεκαετίας του '80 στη Σκανδιναβία άνοιξε ο δρόμος για την ανάπτυξη των κινητών τηλεφώνων μαζικά. Οι συσκευές άρχισαν να γίνονται μικρότερες για να μπορούν να μεταφέρονται ευκολότερα ενώ παράλληλα άρχισαν να υποστηρίζουν και άλλες υπηρεσίες όπως την αποστολή γραπτων μηνυμάτων (SMS) και αργότερα τη λήψη φωτογραφιών. Στη συνέχεια, με τη σύνδεσή τους με το διαδύκτιο και την ενσωμάτωση πιο ολοκληρωμένων λειτουργικών συστημάτων, μετατράπηκαν σε ¨έξυπνα¨ τηλέφωνα.

Σήμερα τα κινητά μπορούν να κάνουν σχεδόν ό,τι κάνει ενας ηλεκτρονικός υπολογιστής ενώ το πλεονέκτημα τους είναι ότι μπορούμε να τα χρησιμοποιήσουμε οπουδήποτε. Έχουμε

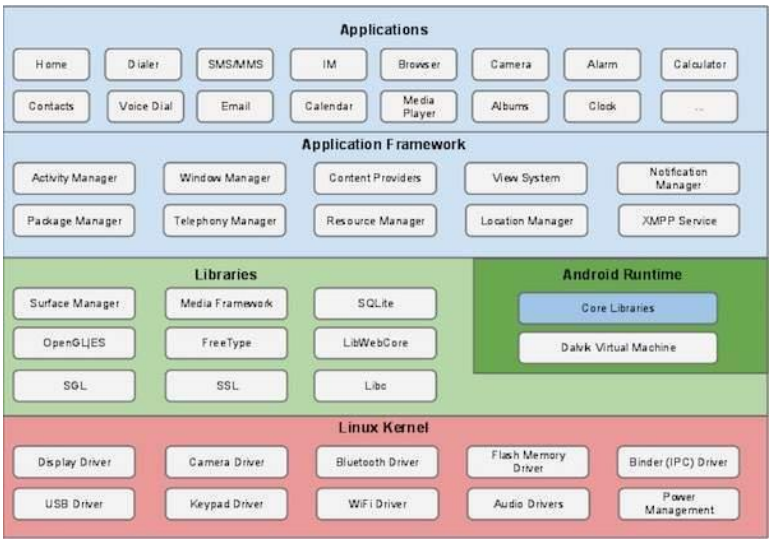
πρόσβαση στα mail μας, μπορούμε να πλοηγηθούμε παντού με τη χρήση gps, να αναζητήσουμε την απάντηση στην όποια απορία μας, να αλληλεπιδράσουμε με άλλες συσκευές που είναι συνδεδεμένες στο διαδίκτυο ρυθμίζοντας ακόμα και τη θερμοκρασία του σπιτιού μας.

1.2 Λειτουργικό συστήμα Android

Εικόνα 1 Android

Το Android, παρουσιάστηκε το Νοέμβριο του 2007 και είναι το πιο διαδεδομένο λειτουργικό σύστημα για έξυπνα τηλέφωνα. Αναπτύχθηκε από τη Google και την Open Handset Alliance. Είναι ενα ¨ανοιχτό¨ λειτουργικό σύστημα (open source) που επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με τη χρήση της γλώσσας προγραμματισμού Java.

Χρησιμοποιεί μια τροποποιημένη έκδοση του πυρήνα του λειτουργικού Linux ο οποίος είναι συμβατός με μια μεγάλη ποικιλία αρχιτεκτονικών επεξεργαστών. Πάνω από τον πυρήνα Linux έχουμε τις βασικές βιβλιοθήκες του συστήματος, βιβλιοθήκες γραφικών, συστήματα βάσεων και άλλα. Οι βιβλιοθήκες αυτές δουλεύουν στον πυρήνα του Linux για να είναι δυνατή η υλοποίηση του Adroid Runtime το οποίο αποτελεί το βασικό μηχανισμό για την εκτέλεση εφαρμογών που αναπτύσονται για το περιβάλλον του Android. Περιλαμβάνει την εικονική μηχανή Dalvik Virtual Machine που διερμηνεύει τις εφαρμογές Android αλλά και τις βασικές βιβλιοθήκες της Java για την ανάπτυξη εφαρμογών. Στη συνέχεια με βάση το Android Runtime το επόμενο επίπεδο είναι το Application Framework το οποίο προσφέρει στον προγραμματιστή δυνατότητες σχετικές με το σύστημα και τη συσκευή. Το τελευταίο επίπεδο είναι αυτό που βρίσκονται οι εφαρμογές του Android.



Εικόνα 2 Android layers

Πέρα από τα smartphones το Android χρησιμοποιείται σε tablet, τηλεοράσεις (Android TV), ρολόγια χειρός (Android Wear) και αυτοκίνητα (Android Auto).

2. Κεφάλαιο: Σύλληψη απαιτήσεων, Εργαλεία, Πρωτόκολλα Ανάπτυξης της εφαρμογής

2.1 Σύλληψη απαιτήσεων

Σκοπός της εφαρμογής είναι να κάνει αναγνώριση εικόνας και image labelling (Google, ML Kit Image Labelling, 2020) στις φωτογραφίες του χρήστη, είτε σε νέες φωτογραφίες είτε σε αποθηκευμένες στη συσκευή, και να αποθηκεύει το geolocation. Η χρήση φωτογραφιών και η προσθήκη ετικετών σε αυτές είναι μια σημαντική δραστηριότητα για μια πληθώρα εφαρμογών, όπως τα κοινωνικά δικτυα, εφαρμογές πιστοποίησης ταυτότητας και ασφάλειας χώρων. (Polonio, Tavella, Zanella, & Bujari, 2018) Ως εκτούτου, το image labelling έχει κεντρίσει το ενδιαφέρων των ερευνητών παγκοσμίως. (Kontogianni & Alepis, 2019) (Li, 2020) Στη συνέχεια θα αποθηκεύεται online ένα μοντέλο ξεχωριστά για τον κάθε χρήστη με τα labels που εμφανίζονται συχνότερα μαζί με ένα ποσοστό συχνότητας. Όλα αυτά θα αποθηκεύονται στο cloud της Firebase. Με βάση το geolocation θα γίνεται ομαδοποίηση των ίδιων labels ώστε να εμφανίζονται σε ένα χάρτη markers (Google, Developers, n.d.) με τα πιο ¨διάσημα¨ labels της περιοχής.

Η χρήση φωτογραφιών και η προσθήκη ετικετώνσε αυτές είναι μια σημαντική δραστηριότητα για μια πληθώρα εφαρμογών, όπως τα κοινωνικά δικτυα

Συνοψίζοντας θέλουμε τα εξής:

- Ο χρήστης να κάνει login/register στην εφαρμογή
- Δημιουργία authentication system
- Πρόσβαση στα αρχεία του χρήστη με το κατάλληλο permission από αυτόν ώστε να κάνουμε populate όλες τις φωτογραφίες του και να τραβήξουμε τις τελευταίες βάσει ημερομηνίας με σκοπό να μας δωθούν τα αντικείμενα που βρέθηκαν σε αυτές.
- Με κατάλληλες προγραμματιστικές μεθόδους θα πρέπει να εμπλουτίζονται δυναμικά πολλά πεδία για τον κάθε χρήστη.
- Θα πρέπει να γίνει κατάλληλη υλοποίηση κάποιας έτοιμης βάσης και cloud ώστε να καταστεί δυνατή η σωστά ομαδοποιημένη αποθήκευση του εκάστοτε χρήστη βάσει του user id. Πχ.Θα πρέπει να αποθηκεύεται συνέχεια το location του χρήστη διότι θα χρειαστεί σε πολλές μεθόδους μέσα στην εφαρμογή όπως

να βάλουμε markers στο χάρτη και να φέρουμε την εκάστοτε τοποθεσία του χρήστη.

- Θα πρέπει να δημιουργήσουμε κατάλληλο intent με τα απαιτούμενα permissions ώστε να ανοίγουμε την κάμερα του τηλεφώνου, να τραβάμε φωτογραφία και να την αποθηκεύουμε στη συσκευή και κατ' επέκταση στο cloud
- Μέσα στην εφαρμογή πρέπει να υπάρχει fragment (Google, Map Fragment, n.d.) με το χάρτη για να εμφανίζονται τα markers ανά περιοχή.

Για να λειτουργήσουν όλα αυτά ο χρήστης πρέπει να έχει συσκευή Android με Google services με σύνδεση στο διαδίκτυο και να δώσει τα κατάλληλα permissions.

3. Εργαλείο ανάπτυξης της εφαρμογής και βιβλιοθήκες

3.1 Android Studio

Εικόνα 2 Android

Το Android Studio είναι ένα ολοκληρωμένο προγραμματιστικό περιβάλλον (IDE) για ανάπτυξη εφαρμογών στην πλατφόρμα Android. Ανακοινώθηκε το Μάιο του 2013 από τη Google. Είναι βασισμένο στο λογισμικό της *JetBrains' IntelliJ IDEA* και αντικατέστησε το Eclipse Android Development Tools (ADT) ως το κύριο IDE της Google για την ανάπτυξη εφαρμογών.

Παρέχει έναν editor ο οποίος εμφανίζει το περιεχόμενο που δημιουργεί ο προγραμματιστής για την εφαρμογή του όπως θα φαινόταν και στη συσκευή. Έχει editor για τις γλώσσες Java και XML, έτοιμες βιβλιοθήκες κώδικα, το σύστημα Gradle της Google για χρήση εξωτερικών βιβλιοθηκών και ένα προσομοιωτή συσκευής για δοκιμές της εφαρμογής.

3.2Firebase

Εικόνα 3 Firebase

Το Firebase είναι ένα κιτ ανάπτυξης λογισμικού (SDK) που ξεκίνησε το 2012 και αποκτήθηκε το 2014 ως λύση για τον προγραμματισμό back-end (Tram, 2019). Οι υπηρεσίες που χρησιμοποιήθηκαν στην εφαρμογή μας είναι οι κάτωθι:

- Authentication (Google, Firebase Authentication, n.d.)

Οι περισσότερες εφαρμογές πρέπει να γνωρίζουν και να επιβεβαιώνουν την ταυτότητα του χρήστη. Γνωρίζοντας την ταυτότητα αυτή, καθίσταται δυνατή η με ασφάλεια αποθήκευση των δεδομένων του χρήστη στο cloud έτσι ώστε να παρέχεται η ίδια εξατομικευμένη εμπειρία στον χρήστη από οποιαδήποτε συσκευή και αν συνδεθεί.

- Realtime Database

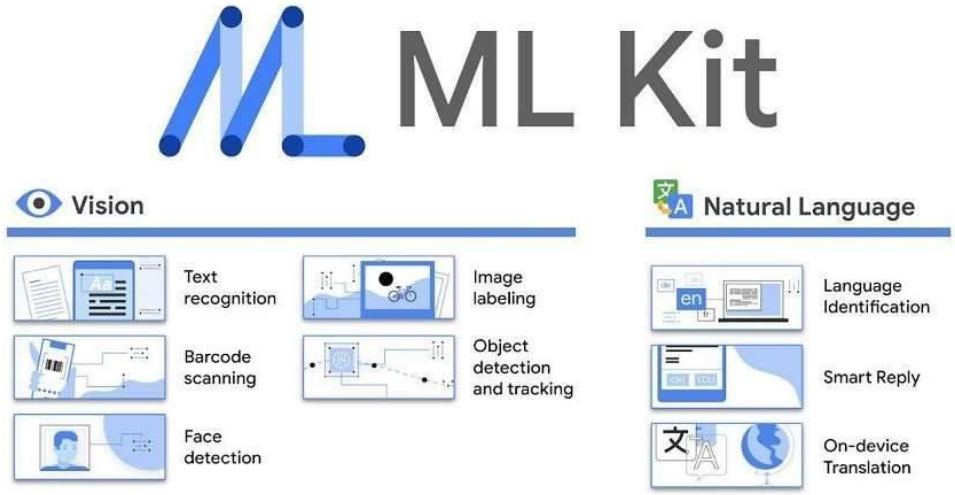
  Το Firebase Realtime Database αποτελεί μια αποθηκευμένη στο cloud βάση δεδομένων. Τα δεδομένα αποθηκεύονται σε μορφή JSON και συγχρονίζονται σε πραγματικό χρόνο για τον κάθε συνδεδεμένο χρήστη. Αυτό έχει ως αποτέλεσμα ο κάθε χρήστης να λαμβάνει αυτόματα ενημερώσεις με τα νεότερα δεδομένα.

- Storage

  Το Firebase Storage υποστηρίζει την αποθήκευση υλικού, όπως βίντεο, ήχος και εικόνες, που έχει δημιουργηθεί από τον χρήστη της εφαρμογής. Το «ανέβασμα» και το «κατέβασμα» των αρχείων γίνεται με απόλυτη ασφάλεια, ασχέτως με την ποιότητα σύνδεσης στο διαδίκτυο.

- ML Kit

  Το ML Kit είναι ένα σύστημα μηχανικής εκμάθησης για προγραμματιστές της Google. Το ML Kit διαθέτει μια ποικιλία δυνατοτήτων, όπως αναγνώριση κειμένου, η ανίχνευση προσώπων, η αναγνώριση αντικειμένων και η παραγωγή ετικετών για αυτα. Αυτή τη στιγμή είναι διαθέσιμη για προγραμματιστές iOS ή Android. Το API μπορεί να χρησιμοποιηθεί τοσό on-device όσο και στο cloud. (Jethwa, 2018)



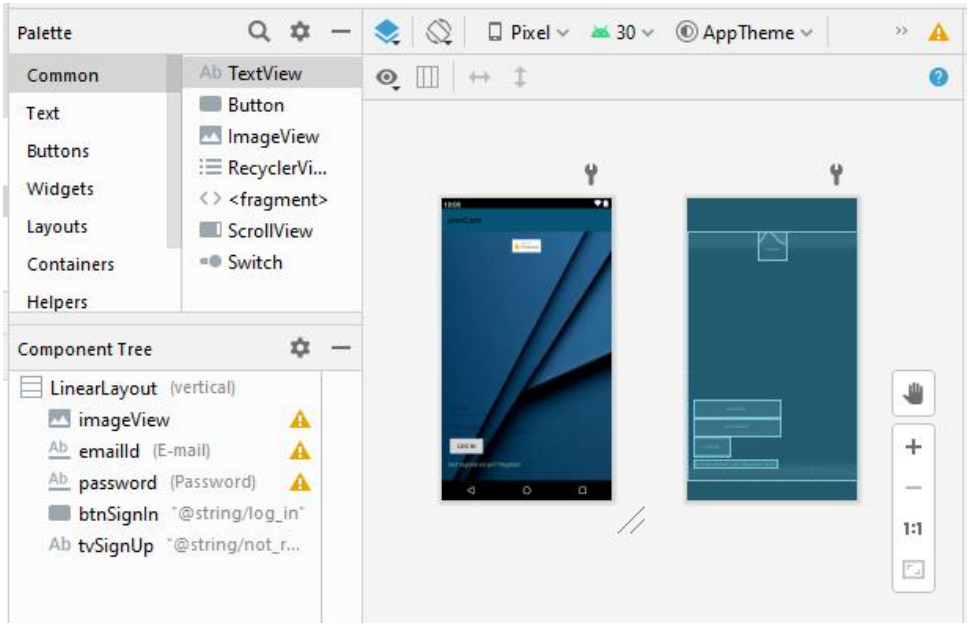
Εικόνα 4 Λειτουργίες του ML Kit (ML Kit, 2020)

4. Κεφάλαιο: Η εφαρμογή

Με τη χρήση των προαναφερόμενων τεχνολογιών, αναπτύχθηκε μια εφαρμογή μέσω της οποίας ο εκάστοτε χρήστης μπορεί να κάνει εγγραφή , να εισέλθει στην εφαρμογή , να βγάλει φωτογραφίες αποθηκεύοντάς τις παράλληλα και αυτομάτως του δημιουργείτε ένα προφίλ που προβάλλει αγαπημένα αντικείμενα και τοποθεσίες σε έναν πραγματικό χάρτη.

4.1 Διεπαφή χρήστη (User Interface)

Για τη δημιουργία διεπαφών χρήστη χρησιμοποιούνται αρχεία γλώσσας xml, τα οποία βρίσκονται μέσα στο φάκελο res/layout. Το Android Studio δίνει την δυνατότητα δημιουργίας του user interface οπτικά. Παρότι η γνώση xml(Extensive Markup language) δεν είναι απαραίτητη καθώς τα στοιχεία μπορούν να εισαχθούν με drag and drop τρόπο , σίγουρα η γνώση της καθιστά την ανάπτυξη πιο εύκολη για τον χρήστη. Ένα στιγμιότυπο της διεπαφής της δικής μας εφαρμογής φαίνεται παρακάτω.



Εικόνα 5 Interface layout

Στα αρχεία java και συγκεκριμένα στο override της μεθόδου OnCreate γίνεται σύνδεση μεταξύ των οπτικών elements(buttons κλπ) με τον κώδικα μας .
Για παράδειγμα εντός της OnCreate μπορεί να εισαχθεί το εξής κομμάτι κώδικα

```
UserEmail=findViewById(R.id.userEmail);
imageview = findViewById(R.id.imageView);
```

Έτσι έγινε απλά η σύνδεση για να δεί το πρόγραμμα μας την σύνδεση του κώδικα με το συγκεκριμένο οπτικό στοιχείο
Για να δώσουμε λειτουργικότητα στο οπτικό μας στοιχείο μπορούμε να χρησιμοποιήσουμε κάποιο listener
Όπως button.onSetClickListener().

4.2 Σχεδίαση εφαρμογής

Όταν σχεδιάζουμε μια εφαρμογή έχουμε υπόψη μας έναν απτους θεμελιώδης κανόνες της ανάπτυξης λογισμικού που είναι το να κάνουμε το πρόγραμμα όσο πιο απλό αλλά και εύχρηστο παράλληλα για τον απλό καθημερινό χρήστη.

Πρέπει να μην ξεχνάμε ότι πιθανότατα το πρόγραμμα μας δεν θα χρησιμοποιηθέι από ειδικούς του κλάδου της Πληροφορικής αλλά απο καθημερινούς ανθρώπους. Ακόμα και άν

έχουμε δημιουργήσει κώδικα υψηλής ποιότητας,δυσκολίας αλλά διατηρήσιμο αν αποτύχουμε στο κομμάτι της εξωτερικής σχεδίασης , πιθανότατα ο κόπος της υπόλοιπης δουλειάς μας θα επισκιαστεί απο αυτό μας το λάθος.

Ένας άλλος κρίσιμος παράγοντας της σχεδίασης μας είναι η συνέπεια της λειτουργικότητας των οπτικών μας στοιχείων όπως και της λειτουργικότητας τους.

Τα στοιχεία μας θα πρέπει να έχουν και να οδηγούν σε κάποια λειτουργία που είναι εντός της κεντρικής ιδέας της εφαρμογής και του σκοπού δημιουργίας της.

Για αυτούς τους λόγους η σχεδίαση είνα ένας κρίσιμος παράγοντας για την υλοποίηση ενός χρήσιμου εργαλείου.

4.3Manifest file

Το androidManifest.xml αποτελεί το κύριο building block της κάθε android εφαρμογής. Θα μπορουσαμε να παραλληλίσουμε την σχέση του αρχειου αυτού με το πρόγραμμα με τη σχέση τους bios με τον ηλεκτρονικό μας υπολογιστή. Είναι το πρώτο αρχείο που διαβάζει το λειτουργικό σύστημα ώστε να καταλάβει το πρόγραμμα μας και τα στοιχεία του.

Μέσα σε αυτό υπάρχουν τα πακέτα που έχουν χρησιμοποιηθεί,τα activities που έχουν δημιουργηθεί ,το activity που ξεκινάει η εφαρμογή , το api level της εφαρμογής καθώς και διάφορες βιβλιοθήκες , services ,permissions (Google, Permissions on Android, n.d.) και implementations (Google, Camera API, n.d.) (Google, Accessing the camera and stored media, n.d.).Κάποια ενδεικτικά παραδείγματα :

```
<uses-feature
   android:name="android.hardware.Camera"
   android:required="true" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

```
<activity android:name=".MainActivity" />
<activity android:name=".RegisterActivity" />
<activity
   android:name=".LoginActivity"
   android:label="plexCam">
<meta-data
   android:name="com.google.android.geo.API_KEY"
   android:value="@string/google_maps_key" />
```

4.4Gradle Files

Το gradle είναι ένα πολύ ισχυρό εργαλείο κατασκευής λογισμικού που δύναται να κατασκευάσει σχεδόν κάθε είδους λογισμικό σύστημα.Το gradle ουσιαστικά κάνει build την εφαρμογή μας μέσω ειδικών αυτοματοποιημένων μεθόδων. Μια εφαρμογή android έχει 2 gradle files. Το top level(build) gradle file που που έχει γενικές ρυθμίσεις οι οποίες αφορούν σε όλα τα sub-modules της εφαρμογής μας και συνήθως δεν κάνουμε ιδιαίτερα πολλές αλλαγές σε αυτό.

Το module-level(app) gradle file το οποίο αφορά στο συγκεκριμένο Module στο οποίο βρίσκεται και γενικά εκεί βάζουμε τα Implementations kai services που θα χρειαστούμε.Ένα μικρό απόσμασμα απτο module level gradle είναι απο κάτω.

```
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services'

dependencies {
   implementation 'com.google.firebase:firebase-ml-vision:24.0.3'
   implementation 'com.google.firebase:firebase-ml-vision-image-label-model:20.0.1'
   implementation 'com.google.firebase:firebase-analytics:17.2.2'
```

4.5Main Activity

Το main activity όπως προδίδει και η ονομασία του αποτελεί την καρδιά της εσφαρμογής μας και ορίζεται μέσω του setContentView. Για την εφαρμογή μας το Main Activity έχει μέσα όλη την απαραίτητη λογική αλλά και τον απαραίτητο κώδικα ώστε να μπορέσει η εφαρμογή να συναντήσει τις αρχικές απαιτήσεις. Στην αρχή η Main Activity έχει όλα τα imports για να αναγνωριστούν διάφρορα objects και external libraries που βρίσκονται εκτός του εύρους του πακέτου που χρησιμοποιούμε. (Google, Developers, n.d.)

Ένα ενδεικτικό παράδειγμα για κάποια πράγματα που χρειαστήκαμε στην εφαρμογή μας

```
import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;

import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
```

```
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.ml.vision.FirebaseVision;
import com.google.firebase.ml.vision.common.FirebaseVisionImage;
import com.google.firebase.ml.vision.label.FirebaseVisionImageLabel;
import com.google.firebase.ml.vision.label.FirebaseVisionImageLabeler;

import android.location.Location;

import com.google.firebase.ml.vision.label.FirebaseVisionOnDeviceImageLabelerOptions;
import com.google.firebase.storage.FirebaseStorage;

import com.google.firebase.storage.StorageMetadata;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;


import java.io.ByteArrayOutputStream;
```

Μετά απο τα imports ακολουθεί η δημιουργία της κλάσης της Main Activity καθώς και οι μεταβλητές σε class level scope που θα χρειαστούνε καθ όλη τη διάρκεια της ζωής του activity και θα χρησιμοποιηθούν απο διάφορες μεθόδους εντός αυτής.

```
public class MainActivity extends AppCompatActivity implements OnMapReadyCallback {
 // We are implementing multiple Firebase services , Authentication System,Firestore ,Firebase
Storage
   //and Firebase Realtime Database , each one for different purpose.

   FirebaseAuth fAuth; //firebase authentication service
   FirebaseFirestore fStore; //firebase firestore object
   StorageReference mStorageRef; //firebase Storage object
   FirebaseDatabase database=FirebaseDatabase.getInstance(); //Firebase RealTime Database
object

   //we need this as global variable
   LatLng latLong;

   String userID;
   StringBuilder userEmail;
```

Έπειτα συνεχίζουμε με την overridden method του Lifecycle μιας android εφαρμογής ονόματι OnCreate. Εδώ γίνονται initialize κάποιες μεταβλητές μας καθώς και γίνεται binding των οπτικών στοιχείων με τα αντίστοιχα code blocks καθώς και το setContentView που αναφέραμε προηγουμένως. Επίσης η μέθοδος αυτη μας παρέχει ένα bundle για την επαναφορά του activity μας απο ένα προηγούμενο frozen state.Απόσπασμα αυτής ακολουθεί παρακάτω:

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //Initializing some textviews and imageview
    UserEmail=findViewById(R.id.userEmail);
    imageview = findViewById(R.id.imageView);
    getLocation = findViewById(R.id.getLocation);
    results = findViewById(R.id.results);
    fStore=FirebaseFirestore.getInstance();


    //Initialize our google API for finding our current location.
    fusedLocationProviderClient = LocationServices.getFusedLocationProviderClient(this);



    results.setText(""); //Resetting our textview on each creation

   //Creating a SupportMapFragment Object
    //basically a fragment for our map.
    SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);
```

Μέσα στην onCreate μας επίσης έχουμε υλοποιήση και μια κομβική μέθοδο για την εφαρμογή αυτη,
Είναι η εφαρμογή που με το πάτημα ενός κουμπιού τραβάμε τις τελευταίες χρονολογικά φωτογραφίες και τις κάνουμε αναγνώρισει για τα στοιχεία που υπάρχουν σε αυτές με σκοπό να κάνουμε ένα σύντομο profiling απτην συλλογή του χρήστη.Κατάλληλα σχολιασμένο στιγμιοτυπο ακολουθει παρακάτω.

```
Button btn = findViewById(R.id.galleryImport);
btn.setOnClickListener(view -> {

        getLocation.setText("Not available for this function");//We return this message cause
```

```
location is not important for this method
        results.setText("");

        FirebaseVisionImage image;
        FirebaseVisionImageLabeler labeler = FirebaseVision.getInstance()
                .getOnDeviceImageLabeler(options);

        String path;
        results.append("Objects found in the 5 latest pictures");
    try {

        for (int i = 0; i < 5; i++) {                       //i just want to get the 5 latest images
            path = fetchGalleryImages(MainActivity.this).get(i); //getting the path from
fetchGalleryImages arrayList method
            Bitmap bm = BitmapFactory.decodeFile(path);              //creating bitmap from
filepath
            image = FirebaseVisionImage.fromBitmap(bm);              //FirebaseVisionImage from
bitmap
            imageview.setImageBitmap(bm);

            labeler.processImage(image)
                    .addOnSuccessListener(new
OnSuccessListener<List<FirebaseVisionImageLabel>>() {
                        @Override
                        public void onSuccess(List<FirebaseVisionImageLabel> labels) {
                            for (int j = 0; j < labels.size(); j++) {
                                String text = labels.get(j).getText();


                                texts.add(text);

                            }
                            results.setText("Objects found in your latest pictures taken");
                            results.append(texts.toString());
                        }


                    });
        }

    }
    catch (Exception e) {
        e.printStackTrace();
```

Αυτός ήτας ένας τρόπος υλοποίησης της εν λόγω μεθόδου καθώς θα μπορούσε να υλοποιηθεί και με άλλους τρόπους όπως πχ με onActivityresult. Με αυτή τη μέθοδο κλείνει η

overridden Μεθοδος του συστήματος onCreate και περνάμε στις δικές μας μεθόδους για την υλοποίηση των απαιτήσεων της εφαρμογής. Ακολουθουν αρκετές μέθοδοι που αρμονικά συβάλλουν στην επιτυχή περάτωση του έργου μας. Για αρχή θα χρειαστεί να στήσουμε τη βάση μας και το cloud.Τα δηλώσαμε στο OnCreate Και κάθε φορα που τα χρειαζόμαστε φτιάχνουμε ένα instance με το path και αναλόγως τραβάμε η στέλνουμε δεδομένα. Έχουμε δημιουργήσει και instance της Realtime Database για να παίρνουμε τα στοιχεία αμέσως μετά απο κάθε μεταβολή στη βάση αλλα και Instance του firebase Storage .

```
mStorageRef = FirebaseStorage.getInstance().getReference();
final StorageReference imageRef = mStorageRef.child("images/" + userID + "/" +
uri.getLastPathSegment());
UploadTask uploadTask = imageRef.putFile(uri, metadata); /

DatabaseReference myRef=database.getReference("Location/"+userID+"/");
DatabaseReference pushref=myRef.push(); //Creating push reference
pushref.setValue(latLong); //pushing the value
```

Μετά υλοποιούμε τη μέθοδο η οποία διαβάζει τις φωτογραφίες απο την συλλογή μας και χρησιμοποιείται απο την προηγούμενη μέθοδο που είδαμε για να κάνουμε το labeling.

```
public ArrayList<String> fetchGalleryImages(Activity context) {

   ArrayList<String> galleryImageUrls;
   final String[] columns = {MediaStore.Images.Media.DATA,
MediaStore.Images.Media._ID};//get all columns of type images
   final String orderBy = MediaStore.Images.Media.DATE_TAKEN;//order data by date

   Cursor imagecursor = getContentResolver().query(
         MediaStore.Images.Media.EXTERNAL_CONTENT_URI, columns, null,
         null, orderBy + " DESC");         //get all data in Cursor by sorting in DESC order

   galleryImageUrls = new ArrayList<String>();

   for (int i = 0; i < imagecursor.getCount(); i++) {
      imagecursor.moveToPosition(i);
      int dataColumnIndex =
imagecursor.getColumnIndex(MediaStore.Images.Media.DATA);//get column index
      galleryImageUrls.add(imagecursor.getString(dataColumnIndex));//get Image from column
index

   }
```

```
    Log.e("getting'", "images");
    return galleryImageUrls;
}
```

Ακολουθει η μέθοδος αλλά και οι βοηθητικές μέθοδοι αυτής που υλοποιούν τη δεύτερη βασική λειτουργία της εφαρμογής , να τραβάει φωτογραφία ,να την σώζει στη συσκευή και στο cloud , να κανει Labeling και να αναγνωρίζει την τοποθεσία επιστρέφοντας τη μαζί με τα αντικείμενα που βρέθηκαν στην αρχική οθόνη του χρήστη.

```
public void getlastlocation() {
    if (ActivityCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_DENIED && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) ==
PackageManager.PERMISSION_DENIED) {
        Toast.makeText(this, "Permission is denied", Toast.LENGTH_SHORT).show();
        ActivityCompat.requestPermissions(MainActivity.this, new
String[]{ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION},
REQUEST_CODE_PERMISSIONS);
    }


    Task<Location> locationTask = fusedLocationProviderClient.getLastLocation();
    locationTask.addOnSuccessListener(location -> {
        if (location != null) {
            longi = location.getLongitude();
            lati = location.getLatitude();
          latLong=new LatLng(lati,longi); //adding our lat,long in a LatLong object
            //We also need to initiate a geoCoder which converts our coordinates to actual readable
Location.
            Geocoder geocoder = new Geocoder(getApplicationContext(), Locale.getDefault());
            try {
                List<Address> addresses = geocoder.getFromLocation(lati, longi, 1);
                getLocation.setText("Location : " + addresses.get(0).getCountryCode() + ", " +
addresses.get(0).getLocality() + ", " + addresses.get(0).getAdminArea());

                myLocation = addresses.get(0).getCountryName() + " , " +
addresses.get(0).getAdminArea();  // this passes the correct value to be used for the uploading
process
                //Here we are saving the coordinates in our Firebase Realtime Database
                DatabaseReference myRef=database.getReference("Location/"+userID+"/"); //creating
```

```
path reference
            DatabaseReference pushref=myRef.push(); //Creating push reference
            pushref.setValue(latLong); //pushing the value
        } catch (IOException e) {
            e.printStackTrace();
        }


    }
});
}
```

Εδώ χρησιμοποιήθηκε το fuseLocationProvider high level API της google για να επιστρέψει τις συντεταγμένες της τοποθεσίας του χρήστη σε συνδυασμό με το geocoder (Google, Geocoder, n.d.) για να μετατρέψει τις συντεταγμένες σε ευανάγνωστη ακριβή τοποθεσία στον χρήστη.

```
private static final SparseIntArray ORIENTATIONS = new SparseIntArray();

static {
    ORIENTATIONS.append(Surface.ROTATION_0, 90);
    ORIENTATIONS.append(Surface.ROTATION_90, 0);
    ORIENTATIONS.append(Surface.ROTATION_180, 270);
    ORIENTATIONS.append(Surface.ROTATION_270, 180);
}

//method one : taking shot from in-app camera and recognize objects automatically
public void ShootAndDetect(View view) {
    Intent imageTakeIntent = new Intent(ACTION_IMAGE_CAPTURE); //This is the intent to
open the camera.
    if (imageTakeIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(imageTakeIntent, REQUEST_IMAGE_CAPTURE);
        ActivityCompat.requestPermissions(this, REQUIRED_PERMISSIONS,
REQUEST_CODE_PERMISSIONS);
    }
}



protected void onActivityResult(int requestCode, int resultCode, Intent imageReturned) {


    super.onActivityResult(requestCode, resultCode, imageReturned);

    if (resultCode != RESULT_CANCELED) {
        if (requestCode == 101) {
```

```
        //We are gonna use Bitmap as image type for increased performance.
        Bundle extras = imageReturned.getExtras();
        Bitmap imageBitmap = (Bitmap) extras.get("data");
        imageview.setImageBitmap(imageBitmap);
        results.setText("");
        Images.Media.insertImage(getContentResolver(), imageBitmap, "shot" + counter,
"description");   //Saving Image to Gallery
        getlastlocation();  //Right after the pic is taken we need to get our location

          //Standard implementation of the Firebase on Device Labeler
        FirebaseVisionImage image = FirebaseVisionImage.fromBitmap(imageBitmap);
        FirebaseVisionImageLabeler labeler = FirebaseVision.getInstance()
                .getOnDeviceImageLabeler(options);
        labeler.processImage(image)  //passing as parameter to processImage method our image
                .addOnSuccessListener(labels -> {
                    //for every label detected in our picture we append it to our text view.
                    for (FirebaseVisionImageLabel label : labels) {
                        String text = label.getText();
                        float confidence = label.getConfidence();
                        results.append("\n" + text + " : " + confidence); //Adding label followed by
confidence .
                        fbLabels.append(text + ",");                    //Also adding our label in fbLabels
list
                        DatabaseReference myRef2=database.getReference("Labels/"+userID+"/");
//to be saved as metadata .
                        DatabaseReference pushref=myRef2.push(); //Creating push reference
                        pushref.setValue(label.getText()); //pushing the values
                    }


                    uploadToFirebase(getImageUri(MainActivity.this, imageBitmap)); //Sending our
objects to our CloudStore

                })
                .addOnFailureListener(e -> Toast.makeText(MainActivity.this, "Task Failed
Successfully", Toast.LENGTH_SHORT).show());


    }
```

Παράλληλα βλέπουμε ότι εστάλησαν στοιχεία Metadata μαζί με την φωτογραφία στο cloud τις firebase καθώς και ξεχωριστά location και labels στο realtime database για περαιτέρω χρήση που αφορά στην μοντελοποίηση του χρήστη καθώς και εισαγωγή markers στον χάρτη της κεντρικής οθόνης.

Για να σταλεί η φωτογραφία χρειάστηκε να δοθεί το ακριβές Path το οποίο καταφέραμε να παρουμε μέσω της κάτωθι μεθόδου

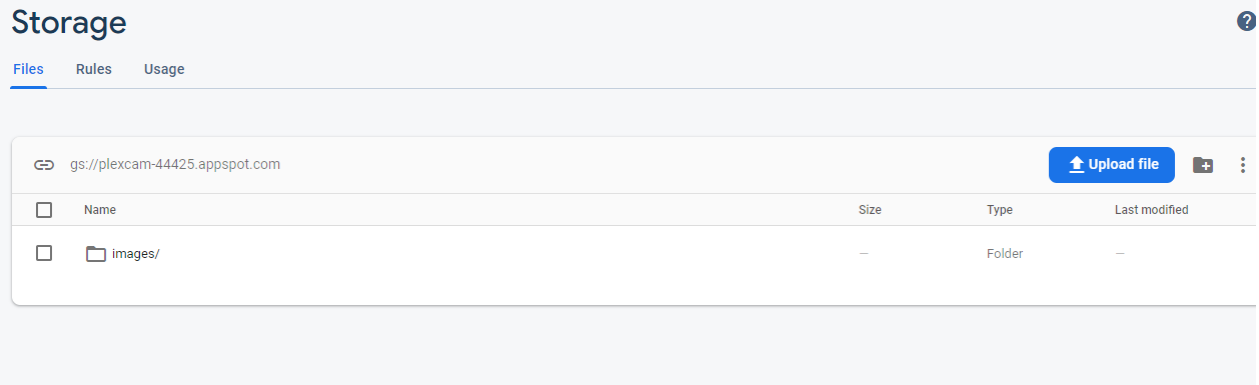
```
public Uri getImageUri(MainActivity inContext, Bitmap inImage) {
   ByteArrayOutputStream bytes = new ByteArrayOutputStream();
   inImage.compress(Bitmap.CompressFormat.JPEG, 100, bytes);

   String path = MediaStore.Images.Media.insertImage(inContext.getContentResolver(),
inImage, "Title", null);
   return Uri.parse(path);
}
```

Αυτή η μέθοδος χρησιμοποιήθηκε απο τη μέθοδο upload ώστε να πάρει το path και να το στείλει βάζοντας σε αυτή τα metadata στο cloud.

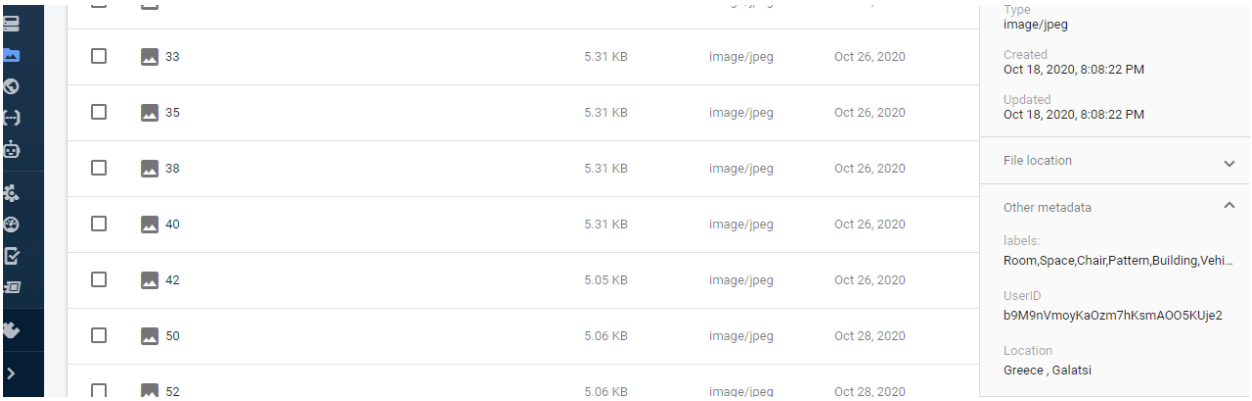
```
public void uploadToFirebase(Uri uri) {
   if (uri != null) {
      final StorageReference imageRef = mStorageRef.child("images/" + userID + "/" +
uri.getLastPathSegment());


      //Adding metadata to our pictures
      StorageMetadata metadata = new StorageMetadata.Builder()
            .setCustomMetadata("UserID", userID)       //the unique user ID
            .setCustomMetadata("Location", myLocation)  //location of the shot
            .setCustomMetadata("labels:", String.valueOf(fbLabels)) //labels that have been
detected
            .build();



      UploadTask uploadTask = imageRef.putFile(uri, metadata); //The upload task
      uploadTask.addOnFailureListener((e) -> {
         Toast.makeText(this, "Uploading task failed", Toast.LENGTH_SHORT).show();
      });
```

Ακολουθούν στιγμιότυπα απο την κονσόλα της firebase

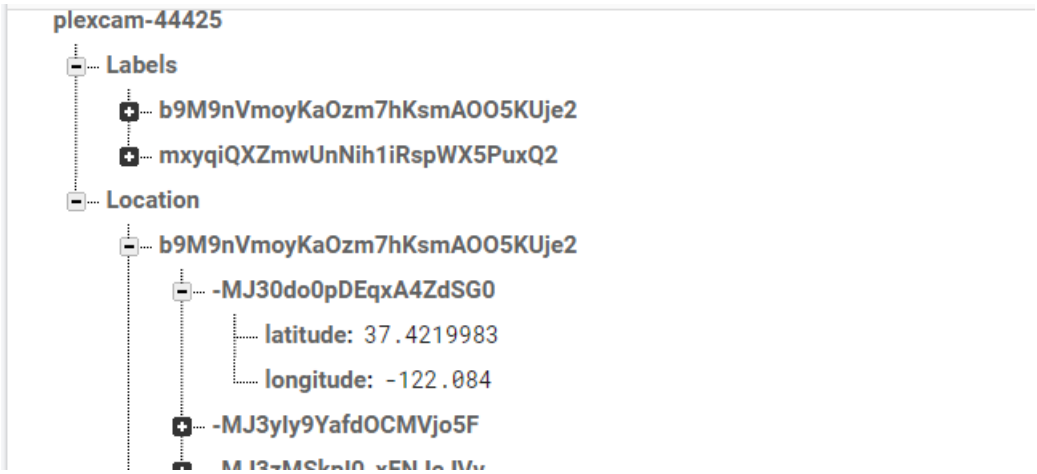
Firebase Storage



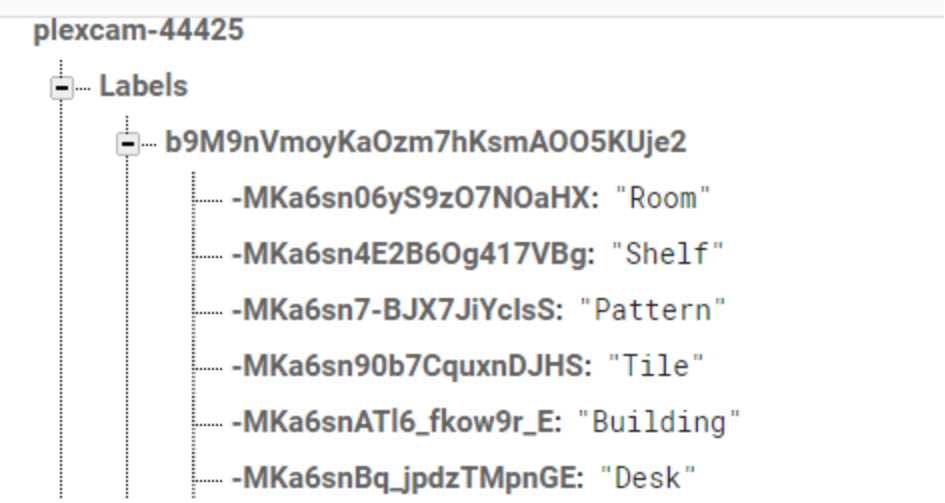
Firebase Storage



Realtime database



Realtime Database



Τα στοιχεία Location τα τραβάει η μέθοδος για το google map μας για να βάλει markers Σε όλα τα σημεία που τράβηξε ο χρήστης (για κάθε ξεχωριστό χρήστη) αλλά και για πιθανή επέκταση της λειτουργικότητας της εφαρμογής.

```
 public void onMapReady(GoogleMap googleMap) {
        //Checking for permissions is Mandatory,else we get an error.
      if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
            ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED)
      {
         return;
      }
MarkerOptions marker=new MarkerOptions(); //Creating a MarkerOptions Object in which we
later pass our coordinates.

      googleMap.setMyLocationEnabled(true);

      DatabaseReference myRef=database.getReference().child("Location").child(userID);
//creating reference
          myRef.addListenerForSingleValueEvent(new ValueEventListener() {
            //Firebase services are all about EventListeners
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
               //we are using the onDataChange to retrieve all the LatLongs(Locations)
               //that user saved in firebase realtime Database
               Iterable<DataSnapshot> snapshotIterable=snapshot.getChildren();
               //we iterate the collection of LatLongs
               //and each time we pass the values to a new LatLng object
```

```
                //which we put as parameter to our marker
                //this way we get all the user's marks and we put them to our google map.
                for (DataSnapshot dataSnapshot: snapshotIterable) {
                 double lat= (double) dataSnapshot.child("latitude").getValue();
                    double lng= (double) dataSnapshot.child("longitude").getValue();
                LatLng ltlng=new LatLng(lat,lng);
               googleMap.addMarker(new MarkerOptions().position(ltlng));
```

Τα στοιχεία απτον φάκελο Labels στο realtime database χρησιμοποιούνται για να επεξεργαστούνε καταλλήλως και να εξαχθούν τα αγαπημένα αντικείμενα μαζι με τον αριθμό εμφανίσεων τους στην αρχική οθόνη του χρηστη.

```
for (DataSnapshot dataSnapshot: snapshotIterable) {

   snaplist.add(dataSnapshot.getValue());



}
Map<String, Long> map = (Map<String, Long>) snaplist.stream()
       .collect(Collectors.groupingBy(w -> w, Collectors.counting()));
List<Map.Entry<String, Long>> result = map.entrySet().stream()
       .sorted(Map.Entry.comparingByValue(Comparator.reverseOrder()))
       .limit(3)  //to return the 3 most favourite items'

       .collect(Collectors.toList());


UserEmail.setText(userEmail+"\n"+"Favorite labels"+"\n"+result.toString());
```

Αυτό κατέστη δυνατό μέσω της χρήσης object datasnapshot και Java Stream.
Ουσιαστικά δημιουργήθηκε ενα map το οποίο πήρε το snapshot(αντικειμενο) και δίπλα τον αριθμό που εμφανίστηκε και επέστρεψε τα αντικείμενα με τις περισσότερες εμφανίσεις απο όλες τις φωτογραφίες του χρήστη.

4.6 Login and register activities

Εδώ έγινε ένα απλό Implementation του έτοιμου authentication service της firebase. Ένα στιγμιότυπο κώδικα:

```
 @Override
   public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
      FirebaseUser mFirebaseUser=mFirebaseAuth.getCurrentUser();
      if(mFirebaseUser!=null) {
         Toast.makeText(LoginActivity.this, "Log in Successful", Toast.LENGTH_SHORT);
         Intent i=new Intent(LoginActivity.this,MainActivity.class);
```

```
            startActivity(i);
        }
        else{
            Toast.makeText(LoginActivity.this, "Please try again", Toast.LENGTH_SHORT);
        }
    }
};
btnSignIn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String email=emailid.getText().toString();
        String pwd=password.getText().toString();
        if(email.isEmpty()){
            emailid.setError("please enter email");
            emailid.requestFocus();
        }

       else if (pwd.isEmpty()){
            password.setError("please enter password");
            password.requestFocus();

        }
        else if (email.isEmpty() && pwd.isEmpty()) {
            Toast.makeText(LoginActivity.this, "Fields are empty ",
Toast.LENGTH_SHORT).show();
        }
        else if (!(email.isEmpty() && pwd.isEmpty())) {

mFirebaseAuth.signInWithEmailAndPassword(email,pwd).addOnCompleteListener(LoginActiv
ity.this, new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (!task.isSuccessful()) {
                        Toast.makeText(LoginActivity.this, "Login Unsuccessful",
Toast.LENGTH_SHORT).show();
                    } else {
                        Intent loginToMain = new Intent(LoginActivity.this, MainActivity.class);
                        startActivity(loginToMain);
                    }
                }
            });
        }
            else{
            Toast.makeText(LoginActivity.this, "An error Occurred ",
Toast.LENGTH_SHORT).show();
                    }
```

```
            }
        });
tvSignUp.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent i=new Intent(LoginActivity.this,RegisterActivity.class);
        startActivity(i);
    }
```

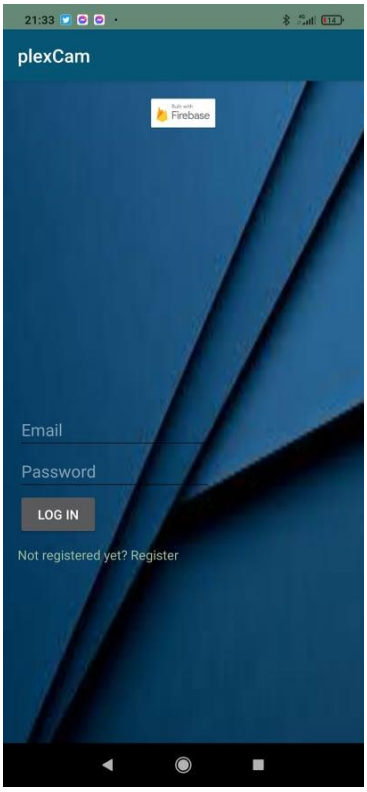
```
  @Override
        public void onClick(View view) {
            String email=emailId.getText().toString().trim();
            String pwd=password.getText().toString().trim();
            if(email.isEmpty()){
                emailId.setError("please enter email");
                emailId.requestFocus();
            }
            else if (pwd.isEmpty()){
                password.setError("please enter password");
                password.requestFocus();

            }
            else if (email.isEmpty() && pwd.isEmpty()) {
                Toast.makeText(RegisterActivity.this, "Fields are empty ",
Toast.LENGTH_SHORT).show();
            }
            else if (!(email.isEmpty() && pwd.isEmpty())) {

mFirebaseAuth.createUserWithEmailAndPassword(email,pwd).addOnCompleteListener(Registe
rActivity.this, new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (!task.isSuccessful() ) {
                            Toast.makeText(RegisterActivity.this, "Signup Unsuccessful",
Toast.LENGTH_SHORT).show();
                        }
                        else{
                            Toast.makeText(RegisterActivity.this, "User created successfully",
Toast.LENGTH_SHORT).show();
                            userID = mFirebaseAuth.getCurrentUser().getUid();
                            DocumentReference documentReference=
fStore.collection("users").document(userID);
                            Map<String,Object> user=new HashMap<>();
                            user.put("email",email);
                            documentReference.set(user).addOnSuccessListener(new
OnSuccessListener<Void>() {
```

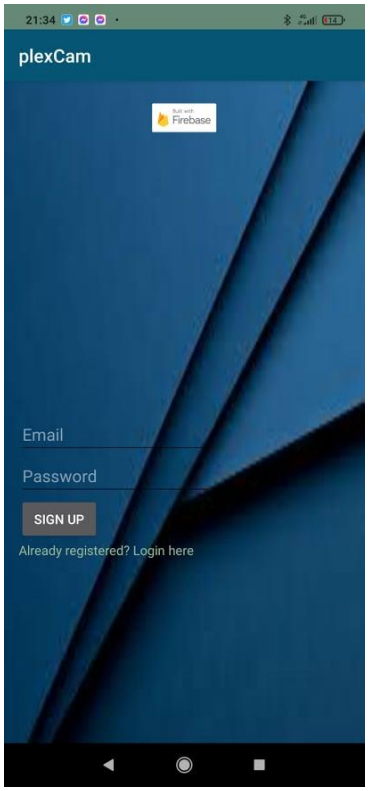
```
                        @Override
                        public void onSuccess(Void aVoid) {
                            Log.d("TAG","user profile is created"+userID);
                        }
                    });
                    startActivity( new Intent(RegisterActivity.this,MainActivity.class));

                }
            }
        });
    }
    else{
        Toast.makeText(RegisterActivity.this, "An error Occurred , please try again ",
Toast.LENGTH_SHORT).show();
    }
  }
});

tvSignIn.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {
    Intent regtolog=new Intent(RegisterActivity.this,LoginActivity.class);
    startActivity(regtolog);
```

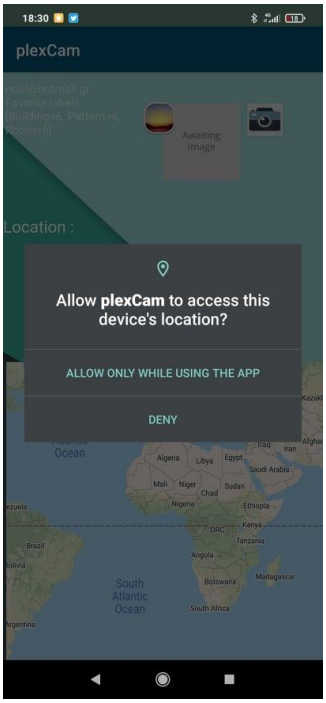
Τελευταίο απόσπασμα κώδικα java απο τη μέθοδο για το google map fragment,επισυνάπτονται μόνο τα κρίσιμα-χρήσιμα σημεία του κώδικα που βρίσκεται η κυρίως λογική.

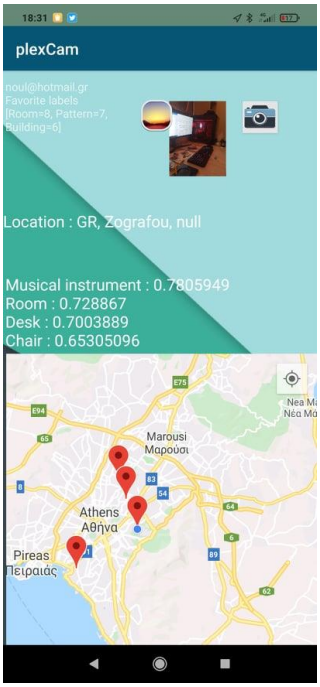
```
@Override
public View onCreateView(@NonNull LayoutInflater inflater,
                         @Nullable ViewGroup container,
                         @Nullable Bundle savedInstanceState) {
  return inflater.inflate(R.layout.fragment_maps, container, false);
}

@Override
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
  super.onViewCreated(view, savedInstanceState);
  SupportMapFragment mapFragment =
        (SupportMapFragment) getChildFragmentManager().findFragmentById(R.id.map);
  if (mapFragment != null) {
    mapFragment.getMapAsync(callback);
  }
```

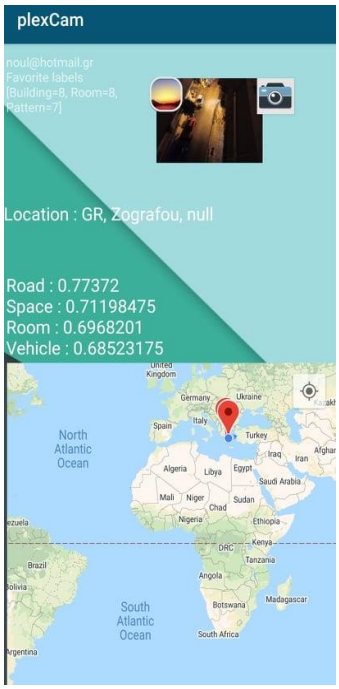
Ενδεικτικά , κάποια στιγμιότυπα οθόνης απο κινητό Xiaomi redmi note 8 με λειτουργικό σύστημα
Android version 10.











5. Συμπεράσματα – Μελλοντικές επεκτάσεις

Μέσα από αυτή τη διπλωματική διατριβή έγινε μια παρουσίαση της λειτουργίας της εφαρμογής αλλά και της διαδικασίας που ακολουθήθηκε για την υλοποίηση της με τα συγκεκριμένα εργαλεία. Είναι γεγονός ότι στις μέρες μας η χρήση smartphones είναι μέσα στην καθημερινότητά μας. Επιπλέον, σχεδόν σε καθημερινή βάση οι χρήστες των έξυπνων αυτών συσκευών λαμβάνουν και αποθηκεύουν ένα πλήθος φωτογραφιών. Η αναγνώριση των αντικειμένων σε αυτές μπορεί εν τέλη να οδηγήσει στη κατανόηση των ενδιαφερόντων του χρήστη και να βοηθήσει την ανάπτυξη εξατομικευμένων εφαρμογών που βασίζονται απόλυτα σε «έμμεσα» δεδομένα. Η εφαρμογή *PlexCam* σε αυτή της τη μορφή επιτυγχάνει τον εντοπισμό τον αντικειμένων που κατά κύριο λόγο φωτογραφίζει ο χρήστης ή αποθηκεύει στη συσκευή του και την επισύμανση αυτών που τον ενδιαφέρουν περισσότερο.

Στα μελλοντικά μας σχέδια είναι να προχωρήσουμε σε μια ευρεία αξιολόγηση του προτεινόμενου συστήματος συλλέγοντας μεγάλο αριθμό εικόνων που προέρχεται από επίσης μεγάλο αριθμό χρηστών. Στόχος μας είναι τελικά η αξιοποίηση και η ενσωμάτωση της εν λόγω τεχνολογίας σε εφαρμογές που θα προσφέρουν εξατομικευμένες υπηρεσίες στους χρήστες τους. Όπως για παράδειγμα μια εφαρμογή εξατομικευμένων συστάσεων στον τομέα του τουρισμου. Αυτό που πρέπει να τονιστεί είναι οτι τέτοιου είδους εφαρμογές θα πρέπει πάντα να έχουν προστασία στη συλλογή και αποθήκευση των δεδομένων τους καθώς θα πρέπει να σέβονται την ιδιοτηκότητα των χρηστών και είναι συμβατά με το γενικό κανονισμό για την προστασία προσωπικών δεδομένων GDPR.

Βιβλιογραφία

Google. (2020). *ML Kit Image Labelling*. Ανάκτηση από https://developers.google.com/ml-kit/vision/image-labeling

Google. (χ.χ.). *Accessing the camera and stored media*. Ανάκτηση από Codepath: https://guides.codepath.com/android/Accessing-the-Camera-and-Stored-Media

Google. (χ.χ.). *Camera API*. Ανάκτηση από Developers: https://developer.android.com/guide/topics/media/camera

Google. (χ.χ.). *Developers*. Ανάκτηση από Fused Location Provider API: https://developers.google.com/location-context/fused-location-provider

Google. (χ.χ.). *Developers*. Ανάκτηση από Maps SDK for Android: https://developers.google.com/maps/documentation/android-sdk/map-with-marker

Google. (χ.χ.). *Firebase*. Ανάκτηση από Firebase Realtime Database: https://firebase.google.com/docs/database/android/start

Google. (χ.χ.). *Firebase Authentication*. Ανάκτηση από Firebase: https://firebase.google.com/docs/auth

Google. (χ.χ.). *Geocoder*. Ανάκτηση από Developer: https://developer.android.com/reference/android/location/Geocoder

Google. (χ.χ.). *Map Fragment*. Ανάκτηση από Developers: https://developers.google.com/android/reference/com/google/android/gms/maps/MapFragment

Google. (χ.χ.). *Permissions on Android*. Ανάκτηση από Developer: https://developer.android.com/guide/topics/permissions/overview

Jethwa, D. (2018). GU-EYE-DE. Faculty of California State Polytechnic University, Pomona.

Kontogianni, A., & Alepis, E. (2019). Moments of Interest: A novel cloud-based crowdsourcing application enhancing smart tourism recommendations. *Computer Science and Electronic Engineering Conference (CEEC).*

Li, J. (2020). Real-time image recognition algorithm and system design of Android mobile devices. *International Journal of Computers and Applications*.

ML Kit. (2020, June). Ανάκτηση από TechGig: https://content.techgig.com/googles-ml-kit-ditches-firebase-now-available-as-simplified-sdk/articleshow/76632790.cms

Polonio, D., Tavella, F., Zanella, M., & Bujari, A. (2018). GHio-Ca: An Android Application for Automatic Image Classification. *International Conference on Smart Objects and Technologies for Social Good*, (σσ. 248-257).

Tram, M. (2019, July). Firebase. CENTRIA UNIVERSITY OF APPLIED SCIENCES.