

**Ανάκτηση Πληροφορίας με Χρήση Οπτικής
Αναγνώρισης Χαρακτήρων και εμφάνιση της
με χρήση Επαυξημένης Πραγματικότητας.**

Νικόλαος Σπυρόπουλος

AM: 3077

Διπλωματική Εργασία

Επιβλέπων: Ν. Μαμουλής

Ιωάννινα, Οκτώβριος 2021

**ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA**



Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα μου κο. Μαμουλή και ιδιαίτερα την οικογένεια μου για την στήριξή τους.

18/10/2021

Νικόλαος Σπυρόπουλος

Περίληψη στα ελληνικά

Ο άνθρωπος συνεχώς αναζήτα πληροφορίες με τον τάχιστο δυνατό τρόπο. Πληροφορίες που αναγράφονται σε ταμπέλες, επιγραφές ή πινακίδες μπορούν να φανούν χρήσιμες αλλά όχι αρκετές. Σε συνδυασμό των συγκεκριμένων πληροφοριών την βοήθεια σύγχρονων τεχνολογιών όπως είναι η Οπτική Αναγνώριση Χαρακτήρων και η Επαυξημένη Πραγματικότητα ερχόμαστε να λύσουμε το παραπάνω ζήτημα και να προσφέρουμε στον χρήστη μια διαδραστική εφαρμογή εύρεσης πληροφορίων και χρήσης αυτών σε πραγματικό χρόνο. Πιο συγκεκριμένα, στην παρούσα διπλωματική εργασία αναπτύχθηκε μια εφαρμογή στο Android Studio για έξυπνα κινητά που δίνει την δυνατότητα στον χρήστη να αναζητήσει πληροφορίες για επαγγελματίες, που βρίσκονται σε περιοχές της Αθήνας, σαρώνοντας λέξεις κλειδιά από σχετικές ταμπέλες. Τα συγκεκριμένα δεδομένα έχουν βρεθεί από το vrisko.gr μέσω τεχνικών scrapping, με την βοήθεια του Scrappy Framework, και έχουν αποθηκευτεί σε μια τοπική βάση δεδομένων σε SQLite διαχείριση της οποίας γίνεται μέσω την βιβλιοθήκης Room. Ο χρήστης κινείται στην πόλη και μόλις εντοπίσει κάποια ταμπέλα επαγγελματία που τον ενδιαφέρει μπορεί να σκανάρει το αναγραφόμενο λογότυπο με ένα πάτημα στην οθόνη και σε πραγματικό χρόνο να αποκτήσει πρόσβαση στην ανακτώμενη επιθυμητή πληροφορία μέσω μιας διαδραστικής εικονικής πινακίδας επαυξημένης πραγματικότητας. Η όλη διαδικασία πραγματοποιείται μέσω ενός OnTapArPlaneListener που χρησιμοποιείται κάθε φορά που ο χρήστης πατάει πάνω στην οθόνη. Για την λειτουργία της οπτικής αναγνώρισης χαρακτήρων χρησιμοποιήθηκε η ευρέως γνωστή μηχανή Tesseract OCR καθώς και κάποιοι αλγόριθμοι επεξεργασίας εικόνας με την βοήθεια της βιβλιοθήκης OpenCV έτσι ώστε να έχουμε τα καλύτερα δυνατά αποτελέσματα. Όσον αφορά την ορθή ανάκτηση πληροφορίας πραγματοποιείται αναζήτηση με το επώνυμο του επιχειρηματία και την πιο πρόσφατη τοποθεσία του χρήστη. Η τοποθεσία είναι αναγκαία για την αποφυγή λαθών λόγω διπλότυπων δεδομένων, στην περίπτωσή μας επαγγελματίες με ίδια επώνυμα. Για το διαδραστικό κομμάτι της επαυξημένης πραγματικότητας έγινε χρήση του ARCore και του Sceneform που μας επιτρέπει την τοποθέτηση 3d μοντέλων χωρίς την χρήση καθαρής OpenGL. Συγκεκριμένα χρησιμοποιήθηκε ένα Viewrendable που τοποθετείται με την χρήση ενός anchor στο σημείο που πάτησε ο χρήστης. Για τον έλεγχο την ορθής λειτουργίας της εφαρμογής έγιναν πειράματα όπως «προσομοίωση κτηρίου επαγγελματιών». Η εφαρμογή έχει ως στόχο να προσφέρει στον χρήστη την απλότητα και την ευχρηστία της σε συνδυασμό με την πλήρη αποτελεσματικότητα που παρέχει μια σειρά περίπλοκων

διαδικασιών στο backend. Το λιτό UI και το διαδραστικό περιβάλλον της εφαρμογής προσφέρει στον χρήστη την δυνατότητα κάποιων χρήσιμων ενεργειών με την ελάχιστη συμμετοχή του. Ο χρήστης πλέον μπορεί να καλέσει, στείλει email, να εισέλθει στο website του εκάστοτε επαγγελματία και να λάβει περισσότερες πληροφορίες μέσω ενός πατήματος σε μια εικονική πινακίδα. Η συγκεκριμένη διαδικασία μπορεί να γίνει επαναληπτικά σε διαφορετικές ταμπλέες χωρίς κανένα πρόβλημα. Πλέον η συγκεκριμένη εφαρμογή μπορεί να χρησιμοποιηθεί ή και να επεκταθεί για να προσφέρει στο κοινό της ένα έξυπνο και τάχιστο εργαλείο που μπορεί να χρησιμοποιήσει για την διευκόλυνση της καθημερινότητάς του.

Λέξεις Κλειδιά: κινητή εφαρμογή, επαυξημένη πραγματικότητα, οπτική αναγνώριση χαρακτήρων, γεωγραφική θέση, επεξεργασία εικόνας, Android Studio, Tesseract OCR, OpenCV, ARCore, Sceneform, Scrappy

Abstract

People are always looking to obtain information most efficiently and effectively. Information on street signs or buildings may seem useful but most of the time, are not sufficient for the users. In addition to this limited information, the help of modern technologies such as Optical Character Recognition and Augmented Reality are able to solve the previously mentioned problems. Such advanced technologies provide the users an interactive application for retrieving information and using it in real time. Thus, allowing users to retrieve further information easily and quickly. This thesis hence aims to overcome the information issues faced by users by developing an application in Android Studio for smart phones users. The application enables users to search for information about professionals such as doctors or lawyers that are located in Athens. Users are simply required to scan keywords from relevant signages. The specific data will be produced by vrisko.gr through scrapping techniques using Scrappy Framework. The data is then stored in a local database, SQLite, and is managed using the Room library. When a user is in the city and find a signage of a professional that he is interested in, he can scan the displayed signage with one touch on the screen and in real time access the desired information through an interactive virtual augmented reality signage. The whole process is performed using an OnTapArPlaneListener that is activated every time the user taps on the screen. The well-known Tesseract OCR engine is used for the optical character recognition function as well as some image processing algorithms with the help of the OpenCV library so that the best possible results are obtained. In regard to the accuracy of the information retrieved, a search in the database is performed with the surname of the professional and the most recent location of the user. The location is necessary to avoid errors due to possible duplicate data such as professionals with the same surnames. For the interactive part of augmented reality, ARCore and Sceneform were used, which allow the installation of 3D models without the use of pure OpenGL. Specifically, a Viewrendable is used as an anchor and is located at the point where users click on their smartphones. Moreover, experiments such as a "professional building simulation" were performed to test the proper operation of the application and to ensure its accuracy. The application aims to offer the user its simplicity and ease of use combined with the full efficiency provided by a series of complex backend processes. The simple UI and the interactive interface of the application offer the user the possibility to call, send an email, access the website of each professional and receive more information by clicking on a virtual signage. This process can be done repeatedly on different signs easily and without any

inconvenience. In addition, this application can also be further developed or extended to offer its audience a smart and fast tool that can be used to facilitate their daily lives for other forms of information retrieval.

Keywords: mobile app, Android, Augmented Reality, Optical Character Recognition, Geographical Location, Android Studio, Tesseract OCR, OpenCV, ARCore, Sceneform, Scrappy

Πίνακας περιεχομένων

Κεφάλαιο 1. Εισαγωγή	1
1.1 Αντικείμενο της διπλωματικής.....	1
1.2 Αναγκαιότητα της εφαρμογής	3
1.3 Οργάνωση του τόμου	4
Κεφάλαιο 2. Περιγραφή Θέματος.....	7
2.1 Στόχος της εργασίας.....	7
2.2 Σχετικές εργασίες και τεχνολογίες	8
2.2.1 Android Studio	8
2.2.2 Οπτική Αναγνώριση Χαρακτήρων.....	9
2.2.2.1 Επισκόπηση OCR	9
2.2.2.2 Εφαρμογές στην καθημερινότητα	10
2.2.2.3 Αναπτυξιακά Εργαλεία OCR	11
2.2.2.4 Tesseract	12
2.2.3 OpenCV	15
2.2.4 Επαυξημένη Πραγματικότητα (AR).....	16
2.2.4.1 Επισκόπηση AR.....	16
2.2.4.2 Εφαρμογές στην Καθημερινότητα.....	17
2.2.4.3 Αναπτυξιακά εργαλεία	19
2.2.5 Scrappy Framework.....	22
2.2.6 SQLite - SQLiteStudio	23
2.2.7 RoomDatabase	23
2.3 Σύγκριση εφαρμογής με ήδη υπάρχουσες.....	25
2.4 Ανάλυση απαιτήσεων.....	26
2.4.1 Use Cases.....	26
2.4.2 Sequence Diagram.....	30
2.4.3 Activity Diagram.....	31

Κεφάλαιο 3. Σχεδίαση & Υλοποίηση.....	32
3.1 Σχεδίαση, αρχιτεκτονική λογισμικού και υλοποίηση.....	32
3.1.1 Grandle.....	32
3.1.2 Manifest.....	33
3.1.3 Activities.....	33
3.1.3.1 Τοποθεσία Χρήστη.....	36
3.1.3.2 Πάτημα στην οθόνη.....	36
3.1.3.3 Εμφάνιση Εικονικής πινακίδας.....	38
3.1.3.4 Design κειμένου εικονική πινακίδας.....	38
3.1.3.5 Επανάληψη διαδικασίας.....	38
3.1.4 Συλλογή Πληροφοριών.....	39
3.1.5 Βάση Δεδομένων.....	41
3.1.5.1 SQLiteStudio.....	41
3.1.5.2 RoomDatabase.....	42
3.1.6 Καταγραφή στιγμάτου & Επεξεργασία του.....	46
3.1.6.1 Καταγραφή.....	46
3.1.6.2 Επεξεργασία εικόνας.....	47
3.1.7 Αναγνώριση χαρακτήρων.....	48
3.1.8 Αντιστοίχιση διευθύνσεων.....	49
3.2 Επεκτασιμότητα του λογισμικού.....	50
Κεφάλαιο 4. Πειραματική Αξιολόγηση.....	54
4.1 Ανάλυση πειράματος.....	53
4.1.1 Άδειες εφαρμογής.....	53
4.1.2 Εντοπισμός κάθετης επιφάνειας	54
4.1.3 Εντοπισμός πιο πρόσφατης τοποθεσίας χρήστη	55
4.1.4 Επεξεργασία εικόνας.....	56
4.1.5 Αναγνώριση Χαρακτήρων	58
4.1.6 Αναζήτηση στην Βάση δεδομένων	60
4.1.7 Εμφάνιση εικονικής πινακίδας	64

Κεφάλαιο 5. Επίλογος.....	73
5.1 Σύνοψη και συμπεράσματα.....	73
5.2 Μελλοντικές επεκτάσεις.....	74

Κεφάλαιο 1. Εισαγωγή

Το κεφάλαιο που ακολουθεί περιγράφει πληροφορίες για το πρόβλημα που προσπαθεί να επιλυθεί και την σημαντικότητα αυτής της εφαρμογής καθώς και την αναγκαιότητα της εφαρμογής.

1.1 Αντικείμενο της διπλωματικής

Στις μέρες μας, οι γρήγοροι ρυθμοί της καθημερινότητας ωθούν τον άνθρωπο στην αναζήτηση βέλτιστων τρόπων πραγματοποίησης των αναγκών του. Η τεχνολογία έχει συμβάλει και συνεχίζει να συμβάλει πλήρως στην ζωή του παρέχοντας του ανέσεις που χωρίς αυτήν δεν θα μπορούσε να έχει. Ιδιαίτερα τα έξυπνα κινητά συμβάλουν καθοριστικά στην διεκπεραίωση σύγχρονων καθημερινών εργασιών και αναγκών του ανθρώπου. Για αυτό τον λόγο παρατηρείται η συχνή εμφάνιση νέων καινοτόμων εφαρμογών για έξυπνα κινητά που τείνουν να αποτελούν αναπόσπαστο κομμάτι της ρουτίνας του ανθρώπου. Τα δύο βασικά λογισμικά, για έξυπνα κινητά, που κυριαρχούν στην εποχή μας είναι το Ios και το Android. Η διαφορά των δύο λογισμικών είναι ότι το Ios χρησιμοποιείται μόνο σε κινητά iPhone ενώ το Android μπορεί να βρεθεί σε ένα μεγάλο φάσμα διαφορετικών brands.

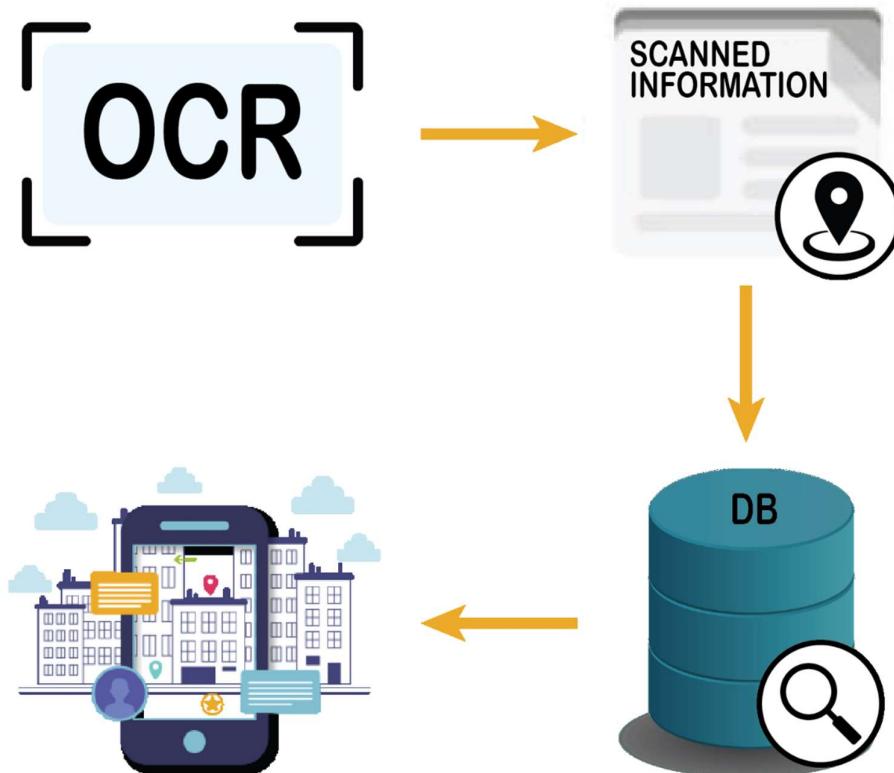
Η ανάγκη του ανθρώπου για εύρεση πληροφορίας που σχετίζεται με καταστάσεις ή οντότητες που τον ενδιαφέρουν αποτελεί καθοριστικό παράγοντα της δημιουργίας πολλών γνωστών εφαρμογών. Με γνώμονα την ανάγκη αυτή και παρατηρώντας την ευχρηστία και αποδοτικότητα των εφαρμογών που ήδη υπάρχουν στο προσκήνιο αποφασίσαμε να αναπτύξουμε μια καινοτόμα διαδραστική android εφαρμογή αναζήτησης πληροφοριών για τον χρήστη. Για την ακρίβεια, η παρούσα εφαρμογή κάνει χρήση της τεχνολογίας Οπτικής Αναγνώριση Χαρακτήρων (OCR) μέσω της κάμερας του κινητού και σε πραγματικό χρόνο ανακτά πληροφορία την οποία εμφανίζει με χρήση της τεχνολογίας Επαυξημένης Πραγματικότητας (AR). Ο χρήστης είναι σε θέση να σκανάρει λογότυπα που μπορεί να βρει σε ταμπέλες μέσα στην πόλη και να ανακτήσει πληροφορία

σχετιζόμενη με αυτά. Η ανακτώμενη πληροφορία εμφανίζεται σε πραγματικό χρόνο σε μια εικονική πινακίδα επαυξημένης πραγματικότητας η οποία δίνει την δυνατότητα στο χρήστη να προβεί σε περαιτέρω ενέργειες πατώντας πάνω στις αναγραφόμενες πληροφορίες. Πιο συγκεκριμένα, στην δική μας περίπτωση έχουμε επικεντρώσει την εφαρμογή σε αναζήτηση πληροφοριών για επαγγελματίες που βρίσκονται σε περιοχές της Αθήνας. Ο χρήστης ενώ κινείται στην πόλη μπορεί να βρει πληροφορίες για κάποιον επαγγελματία σε πραγματικό χρόνο σκανάροντας το λογότυπο που αναγράφεται στην ταμπέλα του έξω από το γραφείο του. Μέσα σε λίγα κλάσματα του δευτερολέπτου μπορεί να αποκτήσει τις απαραίτητες πληροφορίες όπως το τηλέφωνο του, το website του, το email του και να έχει άμεση πρόσβαση σε αυτά, π.χ να πάρει τηλέφωνο, με μόνο ένα απλό πάτημα πάνω στην επιθυμητή πληροφορία. Επίσης, στην εικονική πινακίδα παρέχεται μια συνοπτική περιγραφή για το αντικείμενο του εκάστοτε επαγγελματία.

Η υλοποίηση της συγκεκριμένης εφαρμογής πραγματοποιήθηκε στο Android Studio που αποτελεί το πιο γνωστό περιβάλλον για ανάπτυξη Android εφαρμογών. Η συγκέντρωση της πληροφορίας των επαγγελματιών έγινε με την βοήθεια του Scrappy Framework από το vrisko.gr και τα δεδομένα αποθηκεύτηκαν σε μια βάση δεδομένων SQLite, η διαχείριση της οποίας γίνεται μέσω της βιβλιοθήκης Room. Όσον αφορά την οπτική αναγνώριση χαρακτήρων χρησιμοποιήσαμε την ευρέως γνωστή μηχανή Tesseract OCR και συγκεκριμένα το Tesseract4Android που είναι ειδικά υλοποιημένο για χρήση σε Android development. Η οπτική αναγνώριση χαρακτήρων αποτελεί μια πολύ περίπλοκη λειτουργία καθώς πολλές φορές μπορεί να υπάρχουν εικόνες που περιέχουν μεγάλα ποσοστά θορύβου, κάτι που στέκεται εμπόδιο στην ακριβή αναγνώριση. Για αυτό τον λόγο χρησιμοποιήσαμε την γνωστή βιβλιοθήκη επεξεργασίας εικόνας OpenCV για να παρέχουμε όσο το δυνατόν καλύτερες εικόνες για αναγνώριση. Σχετικά με την ανάκτηση της πληροφορίας πραγματοποιείται με αναζήτηση στην βάση δεδομένων με βάση το επώνυμο του επαγγελματία και την πιο πρόσφατη τοποθεσία του χρήστη για αποφυγή λανθασμένων πληροφοριών λόγω ατόμων με ίδιο επώνυμο. Η εύρεση της τοποθεσίας έγινε μέσω του fused location provider που παρέχεται από το Google Play Services. Τέλος, για το κομμάτι της επαυξημένης πραγματικότητας χρησιμοποιήσαμε το ARcore και το Sceneform το οποίο αποτελεί ένα API υψηλού επιπέδου για γραφικά βασισμένο σε OpenGL το οποίο δίνει την δυνατότητα στον προγραμματιστή να το χρησιμοποιήσει έναντι καθαρής OpenGL.

Αρχικά ο έλεγχος της συγκεκριμένης εφαρμογής πραγματοποιήθηκε πειραματικά στο κτίριο που διαμένω ως μια προσομοίωση πολυκατοικίας πολλών γραφείων επαγγελματιών. Εν συνεχείᾳ, έγινε σε περιοχές των Αθηνών και σε κτίρια που στεγάζουν διάφορα γραφεία επαγγελματιών. Η συγκεκριμένη εφαρμογή είναι πλήρως λειτουργική

και επεκτάσιμη και στοχεύει στην ελάχιστη συμμετοχή του χρήστη για την ανάκτηση των επιθυμητών πληροφοριών. Ο χρήστης κυριολεκτικά μόνο με ένα πάτημα πάνω στην οθόνη εστιάζοντας στο λογότυπο που τον ενδιαφέρει μπορεί να αποκτήσει πληροφορίες σε άμεσο χρόνο με τις οποίες μπορεί να ενεργήσει ανάλογα μέσα από μια διαδραστική εμπειρία επαυξημένης πραγματικότητας. Τέλος η παρούσα εφαρμογή μπορεί να επεκταθεί πολύ εύκολα αλλάζοντας απλά την βάση δεδομένων της και κάνοντας κάποιες μικρές αλλαγές στον τρόπο αναζήτησης πληροφορίας δηλαδή την αλλαγή της λέξης κλειδί.



Εικόνα 1.1 Σχεδιάγραμμα λειτουργίας εφαρμογής

1.2 Αναγκαιότητα της εφαρμογής

Η ιδέα της ανάπτυξης της συγκεκριμένης εφαρμογής προήλθε από έναν συνδυασμό δυο πολύ διάσημων εφαρμογών της Google. Η πρώτη εφαρμογή είναι το γνωστό Google Lens που δίνει την δυνατότητα στον χρήστη να αναζήτηση ότι θέλει μέσω οπτικής αναγνώρισης χαρακτήρων που αναγράφονται πάνω στο ενδιαφερόμενο αντικείμενο ή μη αλλά και μέσω οπτικής αναγνώρισης αντικειμένου. Η δεύτερη εφαρμογή αποτελεί το Google Translate Scan που προσφέρει στον χρήστη μια εμπειρία

επαυξημένης πραγματικότητας εμφανίζοντας μεταφρασμένο κείμενο πάνω από αυθεντικό. Η πρώτη εφαρμογή μας έδωσε την ιδέα της ανάκτησης πληροφορίας μέσω Οπτικής αναγνώρισης χαρακτήρων και η δεύτερη χρήση Επαυξημένης πραγματικότητας σε συνδυασμό με Οπτική Αναγνώριση Χαρακτήρων.

Οι δυο συγκεκριμένες εφαρμογές έχουν ένα πολύ μεγάλο κοινό καθώς προσφέρουν τις υπηρεσίες τους σε εκατομμύρια χρήστες παγκοσμίως. Σύμφωνα με την πηγή το Google Lens έφτασε τα 50 εκατομμύρια downloads μέσα σε μόνο 16 μήνες από την εμφάνιση του. Όσον αφορά το Google Translate Scan όχι μόνο αποτελεί βρίσκεται στην κορυφή των εφαρμογών με αυτόματη μετάφραση αλλά αποτελεί πλέον και λειτουργία του Google Lens που όπως είπαμε είναι μια από τις δημοφιλέστερες εφαρμογές σήμερα. [1][2][3]

Αυτές οι δυο εφαρμογές λοιπόν και η αγάπη και αφοσίωση του κόσμου προς αυτές μας έκανε να κατανοήσουμε την μεγάλη αναγκαιότητα που υπάρχει για ανάπτυξη τέτοιων εφαρμογών. Για αυτό τον λόγο ερχόμαστε εμείς με την σειρά μας να δημιουργήσουμε μια εφαρμογή ανάκτησης πληροφορίας μέσω αναγνώρισης χαρακτήρων και εμφάνισης της με επαυξημένη πραγματικότητα.

1.3 Οργάνωση του τόμου

Μετά από μια σύντομη εισαγωγή του θέματος και της αναγκαιότητας της εφαρμογής στο Κεφάλαιο 1, ακολουθεί το Κεφάλαιο 2 με αναφορά σχετικών τεχνολογιών, λειτουργιών και των απαιτήσεων της εφαρμογής.

Στο Κεφάλαιο 3 παρουσιάζεται εκτενέστερα η σχεδίαση και η υλοποίηση της εφαρμογής, και μια μικρή ανάλυση στους τρόπους επεκτασιμότητας της εφαρμογής. Γίνεται χρήση διαγραμμάτων και απαραίτητων στιγμιότυπων από την υλοποίηση

Στο Κεφάλαιο 4 αναλύεται ένα πείραμα ελέγχου της εφαρμογής σε βάθος με όλες τις απαραίτητες τεχνικές λειτουργίες. Επίσης αναφέρονται σχετικές δυσκολίες που συναντήσαμε και μπορέσαμε να αντιμετωπίσουμε ή και όχι.

Το Κεφάλαιο 5 συνοψίζει τα συμπεράσματα μας μετά την ολοκλήρωση διπλωματικής εργασίας και παρουσιάζει ένα πλήθος από πιθανές επεκτατικές ιδέες που μπορεί να υλοποιήσουμε στο μέλλον.

Κεφάλαιο 2. Περιγραφή Θέματος

2.1 Στόχος της εργασίας

Στόχος της παρούσας διπλωματικής είναι η δημιουργία μιας πλήρως λειτουργικής και επεκτάσιμης εφαρμογής που θα παρέχει στον χρήστη πληροφορίες, που τον ενδιαφέρουν, σε πραγματικό χρόνο και με την ελάχιστη συμμετοχή του. Δημιουργήσαμε την συγκεκριμένη εφαρμογή με γνώμονα την τάχιστη ανάκτηση πληροφορίας από ταμπέλες ή πινακίδες που μπορεί να συναντήσει ο χρήστης στην πόλη και θέλει να γνωρίζει περισσότερα για το αντικείμενο που αναφέρεται η πινακίδα. Η συγκεκριμένη εφαρμογή αποτελεί ένα διαδραστικό εργαλείο ειδικά ανεπτυγμένο με βάση τους γρήγορους ρυθμούς της σημερινής εποχής. Δίνει την δυνατότητα στο χρήστη να βρει πληροφορίες για κάποιον επαγγελματία που ενδιαφέρεται με σχεδόν την μηδενική του συμμετοχή. Ένα παράδειγμα χρήσης της εφαρμογής θα μπορούσε να είναι ένας χρήστης που βρίσκεται έξω από ένα κτίριο που στεγάζει το γραφείο ενός δικηγόρου που τον ενδιαφέρει. Σε περίπτωση που ο δικηγόρος είναι απόν ο χρήστης μπορεί να σκανάρει την ταμπέλα του δικηγόρου με ένα απλό πάτημα της οθόνης και σε πραγματικό χρόνο του παρέχεται μια εικονική πινακίδα με πληροφορίες. Μέσω αυτής της πινακίδας ο χρήστης μπορεί να κάνει κάποιες πολύ χρήσιμες λειτουργίες όπως για παράδειγμα πατώντας πάνω στο αναγραφόμενο email να μεταφερθεί σε μια email εφαρμογή και να στείλει κάποιο ηλεκτρονικό μήνυμα στην διεύθυνση του δικηγόρου. Τέλος ο χρήστης μπορεί να μεταφερθεί πίσω στην εφαρμογή και να ενεργήσει με βάση άλλες πληροφορίες ή να επαναλάβει την συγκεκριμένη διαδικασία για κάποιον άλλο επαγγελματία που βρίσκεται στο κτίριο. Όλες αυτές οι λειτουργίες θα περιγράφουν εκτενέστερα στο κεφάλαιο 3. Βασικές τεχνολογίες για να επιτευχθεί η σωστή λειτουργία της παρούσας εφαρμογής είναι η οπτική αναγνώριση χαρακτήρων, η επαυξημένη πραγματικότητα κ.α., περισσότερες πληροφορίες για τις οποίες θα δούμε στο επόμενο κεφάλαιο.

2.2 Σχετικές εργασίες και τεχνολογίες

2.2.1 Android Studio

Για την υλοποίηση της συγκεκριμένης εφαρμογής χρησιμοποιήσαμε το περιβάλλον Android Studio που αποτελεί ένα ολοκληρωμένο προγραμματιστικό περιβάλλον για την ανάπτυξη android εφαρμογών σε γλώσσα Java ή Kotlin καθώς υποστηρίζει και C++. Έχει δημιουργηθεί από την Google και η πρώτη του εμφάνιση πραγματοποιήθηκε το 2003. Η ανάπτυξη του έχει βάση το JetBrains-IntelliJ IDEA και είναι υλοποιημένο στην γλώσσα προγραμματισμού Java. Δεν θα μπορούσαμε να παραλείψουμε την ύπαρξη του Android SDK το οποίο περιέχει ένα πλήθος βιβλιοθηκών και άλλων εργαλείων που συχνά είναι απαραίτητα για την ανάπτυξη διαφόρων εφαρμογών. Το Android Studio προσφέρει ανάπτυξη εφαρμογών android προσβάσιμες σε όλες τις Android συσκευές όπως smartphones, smart TV's, smart Watches, Tablets κ.α.

Για την δημιουργία μιας εφαρμογής στο συγκεκριμένο περιβάλλον χρησιμοποιείται ένα σύστημα κατασκευής(build) που είναι βασισμένο στο **Gradle**. Το Gradle αποτελεί ένα build system που χρησιμοποιείται για κατασκευή, testing κ.α. Τα αρχεία build.gradle που υπάρχουν σε ένα project Android Studio αποτελούν scripts τα οποία μπορούν να χρησιμοποιηθούν από τον προγραμματιστή για να αυτοματοποιήσει κάποιες λειτουργίες. Το gradle είναι απαραίτητο για ένα Android project επειδή παίρνει όλα τα source αρχεία τύπου java και XML και εφαρμόζει σε αυτά κατάλληλα εργαλεία.

Σε ένα android project υπάρχουν δύο τύποι αρχείων gradle.build.

- **Top-level build.gradle:** Το οποίο βρίσκεται στον αρχικό φάκελο του project και έχει ως βασική λειτουργία τον καθορισμό των διαμορφώσεων κατασκευής που θα εφαρμοστούν σε όλα τα modules του project.
- **Module-level build.gradle:** Το συγκεκριμένο αρχείο περιέχει τα dependencies του project που αφορούν όλες τις βασικές και επιπλέον βιβλιοθήκες και εργαλεία που χρησιμοποιούνται.

Ένα άλλο κύριο μέρος ενός Android project είναι τα **Activities**. Κάθε Activity της εφαρμογής αποτελεί μια διαφορετική οθόνη για τον χρήστη. Συνήθως χρησιμοποιείται για διαφορετικές λειτουργίες της εφαρμογής. Τα διάφορα activities της εφαρμογής μπορούν να επικοινωνούν μεταξύ τους και ανταλλάσσουν δεδομένα. Για το design των activities, αλλά και όχι μόνο, χρησιμοποιούνται τα λεγόμενα **Layout XML** αρχεία μέσα από τα οποία δίνεται η δυνατότητα στον προγραμματιστή να σχεδιάσει είτε χειροκίνητα μέσα από ένα βοηθητικό

UI του Android Studio είτε γράφοντας κώδικα XML με κουμπιά, TextViews κ.α. Αυτά τα αρχεία συνδέονται με τον πηγαίο κώδικα ενός Activity ή και όχι για την υλοποίηση λειτουργιών πάνω στα στοιχεία τους.

Υπάρχουν δύο σημαντικοί φάκελοι σε ένα project Android Studio, συγκεκριμένα ο **assets** και ο **res** (resources). Ο φάκελος assets είναι ένας πολύ σημαντικός φάκελος καθώς μπορούν να αποθηκευτούν σε αυτόν αρχεία διαφόρων τύπων που είναι χρήσιμα για την εφαρμογή όπως αρχεία κειμένου, ήχου, βάσεων κ.λπ.. Ο προγραμματιστής με την προσθήκη των assets μπορεί να χρησιμοποιήσει όποτε θέλει τα περιεχόμενα αρχεία στο κώδικα του. Όσον αφορά τον φάκελο res περιέχει τους πόρους της εφαρμογής που είναι drawables, layouts, mipmaps και values.

Τέλος, δεν θα μπορούσαμε να μην αναφέρουμε το αρχείο `AndroidManifest.xml`. Κάθε android εφαρμογή χρειάζεται να έχει αυτό το αρχείο καθώς αυτό περιγράφει όλες τις απαραίτητες πληροφορίες για την εφαρμογή. Συγκεκριμένα, μέσω αυτού καθορίζονται τα permissions που πρέπει να έχει η εφαρμογή κ.α.

2.2.2 Οπτική Αναγνώριση Χαρακτήρων

2.2.2.1 Επισκόπηση OCR

Η Οπτική Αναγνώριση Χαρακτήρων ή αλλιώς Αυτόματη Αναγνώριση Χαρακτήρων Κειμένου αποτελεί τη διαδικασία εξαγωγής αναγνώσιμου κειμένου, για τον ηλεκτρονικό υπολογιστή, από σαρωμένες εικόνες με χειρόγραφο ή τυποποιημένο περιεχόμενο. Η συγκεκριμένη τεχνολογία μπορεί να φανεί ιδιαίτερα χρήσιμη καθώς καθιστά εφικτή την εκ νέου επεξεργασία του κειμένου χωρίς να χρειάζεται η ηλεκτρονική συγγραφή του κειμένου από την αρχή.

Η εφαρμογή της Οπτικής Αναγνώρισης μπορεί να πραγματοποιηθεί με δύο τρόπους, την Αντιστοίχιση με Πρότυπα και την Εξαγωγή Χαρακτηριστικών. Παρόλο που η πρώτη μέθοδος είναι πιο διαδεδομένη, και κοινή, έχει αρκετούς περιορισμούς σε σχέση με την δεύτερη. Για αυτό τον λόγο σήμερα και κυρίως σε χειρόγραφα έγγραφα πραγματοποιείται συνδυασμός των δύο τεχνολογιών για όσο το δυνατόν καλύτερα αποτελέσματα.

Η αντιστοίχιση με πρότυπα βασίζεται σε έτοιμα πρότυπα ή περιγράμματα χαρακτήρων. Αρχικά ο εκάστοτε σαρωτής μετατρέπει το πραγματικό έγγραφο σε ψηφιοποιημένη μορφή και το λογισμικό Οπτικής Αναγνώρισης προσπαθεί να επιτύχει την βέλτιστη αντιστοίχιση των ψηφιοποιημένων πλέον χαρακτήρων με τα ήδη αποθηκευμένα

πρότυπα. Με την προϋπόθεση ότι το παραπάνω ταίριασμα έχει γίνει επιτυχές αντιστοιχίζεται με τον ανάλογο χαρακτήρα κειμένου για τον ηλεκτρονικό υπολογιστή.

Η εξαγωγή χαρακτηριστικών είναι επίσης γνωστή ως Ευφυής Αναγνώριση Χαρακτήρων (Αγγλ. Intelligent Character Recognition – ICR), ή τοπολογική ανάλυση χαρακτηριστικών. Η μέθοδος αυτή είναι ένα είδος οπτικής αναγνώρισης όπου η αντιστοιχίση δεν βασίζεται σε πρότυπα. Εδώ το λογισμικό είναι βασισμένο στην αναγνώριση επιμέρους συστατικών στοιχείων ενός χαρακτήρα, όπως γωνίες, γραμμές, ενώσεις κτλ. Σε αυτό το λογισμικό δημιουργούνται κανόνες για την επίτευξη των αντιστοιχίσεων.

Αξιόλογο θα ήταν να αναφέρουμε ότι η διαδικασία της αναγνώρισης χειρόγραφων χαρακτήρων είναι πιο πολύπλοκη και για αυτό απαιτείται ο συνδυασμός των δύο παραπάνω μεθόδων σε συνδυασμό με άλλους παράγοντες όπως τις γνώσεις για τον συγγραφέα και το περιεχόμενο του κειμένου. Τέλος, η ύπαρξη της καλλιγραφίας στα χειρόγραφα έγγραφα μπορεί να επηρεάσει καταλυτικά μια επιτυχημένη αναγνώριση των χαρακτήρων. Αυτό οφείλεται στην συνεχόμενη γραφή των γραμμάτων που εμποδίζει τον υπολογιστή να αναγνωρίσει πότε αρχίζει και πότε τελειώνει ένα γράμμα. [4]

2.2.2.2 Εφαρμογές στην καθημερινότητα

Στις μέρες μας, μετά από πολυετείς έρευνες από εταιρίες κολοσσούς και αξιόλογους επιστήμονες η Οπτική Αναγνώριση Χαρακτήρων έχει καταπολεμήσει τα περισσότερα εμπόδια της. Πλέον οι σχετικές τεχνολογίες παρέχονται σε κάθε είδους προγραμματιστή και πολίτη για χρήση τους με δυνατότητες αναγνώρισης κειμένου σε όλες τις γλώσσες και σε κάθε είδος κειμένου.

Ένα σημερινό παράδειγμα που πολλοί κάτοχοι έξυπνων κινητών χρησιμοποιούν είναι το **Google Translate** μια εφαρμογή που δίνει την δυνατότητα σε έναν απλό χρήστη να μεταφράσει κείμενο χρησιμοποιώντας απλά την κάμερα του κινητού του. Ο χρήστης σαρώνει το κείμενο που θέλει να μεταφράσει και σε πραγματικό χρόνο με τεχνικές επαυξημένης πραγματικότητας το κείμενο εμφανίζεται στην οθόνη αφού πρώτα έχει αναγνωριστεί μέσω OCR και μεταφραστεί.

Ένα άλλο παράδειγμα που μπορούμε να συναντήσουμε εφαρμογή της Οπτικής Αναγνώρισης Χαρακτήρων είναι τα **συστήματα Αυτόματης Αναγνώρισης πινακίδων κυκλοφορίας (ALPR)** τα οποία χρησιμοποιούνται σε κεντρικές οδούς για αναγνώριση πινακίδων αυτοκινήτων που έχουν ιλιγγιώδη ταχύτητα αλλά και σε ιδιωτικά parking για να γίνεται έλεγχος του αμαξιού που σπεύδει να εισέλθει.

2.2.2.3 Αναπτυξιακά Εργαλεία OCR

Για την υλοποίηση εφαρμογών Οπτικής Αναγνώρισης Χαρακτήρων έχουν δημιουργηθεί διάφορες βιβλιοθήκες και API δωρεάν ή επί πληρωμή που θα δούμε παρακάτω. Τα συγκεκριμένα εργαλεία έχουν υψηλή αποδοτικότητα με κάποια από αυτά βέβαια να χάνουν την αποδοτικότητα τους λόγω του θορύβου της εκάστοτε εικόνας αλλά και της πολυπλοκότητας της γλώσσας που πρέπει να αναγνωριστεί.

Google Vision Api

Τα τελευταία χρόνια η Google έχει δημιουργήσει ένα API ονόματι **Google Vision API** το οποίο χρησιμοποιείται ευρέως σε κινητές αλλά και άλλες εφαρμογές. Το συγκεκριμένο API δίνει την δυνατότητα στον χρήστη να μετατρέπει χειρόγραφο ή τυποποιημένο κείμενο σε κείμενο αναγνώσιμο για τον ηλεκτρονικό υπολογιστή με αρκετά υψηλή αποδοτικότητα.

Το **Google Cloud Vision OCR** είναι μέρος του **Google cloud API** για την εξαγωγή κειμένου από εικόνες. Συγκεκριμένα, υπάρχουν δύο **annotations** που βοηθούν στην αναγνώριση χαρακτήρων. Αρχικά το **Text_Anotation** εξάγει κείμενα που έχουν κωδικοποιηθεί από μηχανή υπολογιστή από οποιαδήποτε εικόνα (π.χ. φωτογραφίες από θέα στο δρόμο ή τοπία). Δεδομένου ότι αρχικά σχεδιάστηκε για να μπορεί να χρησιμοποιηθεί σε διαφορετικές συνθήκες φωτισμού, το μοντέλο είναι κατά κάποιο τρόπο πιο ισχυρό στην ανάγνωση λέξεων διαφορετικών στυλ. Το δεύτερο annotation ονόματι **Document_Text_Anotation** είναι ειδικά σχεδιασμένο για πυκνά έγγραφα κειμένου (π.χ. σαρωμένα βιβλία). Για αυτό τον λόγο, παρόλο που υποστηρίζει την ανάγνωση μικρότερων και πιο συγκεντρωμένων κειμένων, είναι λιγότερο αποδοτική σε εξωτερικές εικόνες. [17]

Tesseract

Η ανάπτυξη του συγκεκριμένου εργαλείου αναπτύχθηκε κατά τα μέσα της δεκαετίας του 80' και 90' από τα Hewlett-Packard Laboratories στο Bristol και τη Hewlett-Packard Co στο Greeley Colorado με ορισμένες αλλαγές να πραγματοποιούνται έως τα τέλη της δεκαετίας του 90'. Από το 2005 η Tesseract αποτελεί μια μηχανή ανοιχτού κώδικα και από το 2006 έως και σήμερα είναι στην κατοχή της Google η οποία έχει συμβάλει σε μεγάλο βαθμό στην αποδοτικότητα που έχει σήμερα.

Η **Tesseract** σήμερα χρησιμοποιείται σε διαφόρων ειδών εφαρμογές και είναι προβιβάσιμη σχεδόν από όλες τις γλώσσες προγραμματισμού. Γνωστός είναι ο

συνδυασμός της **OpenCV** με την **Tesseract** για προγράμματα βασισμένα στην επεξεργασία εικόνας.

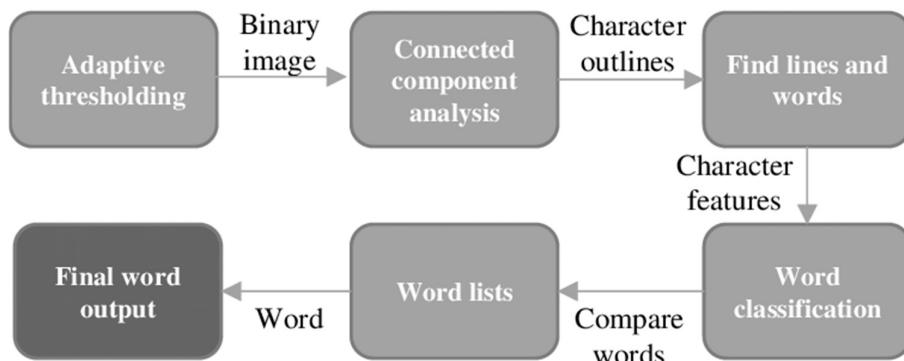
Για την χρήση της **Tesseract** σε εφαρμογές σε περιβάλλοντα **Android** χρησιμοποιείται η βιβλιοθήκη **Tess-two**. Η Tess-two παρέχει ένα Java API για πρόσβαση σε εγγενώς μεταγλωττισμένα Tesseract APIs.

2.2.2.4 Tesseract

Στην παρούσα διπλωματική εργασία για την οπτική αναγνώριση χαρακτήρων επιλέξαμε την μηχανή Tesseract OCR και συγκεκριμένα το Tesseract4Android. Το Tesseract4Android αποτελεί ένα ανοιχτού κώδικα project που μπορεί να χρησιμοποιηθεί για την υλοποίηση εφαρμογών OCR σε Android Studio. Αποτελεί ένα fork της Tess-two η οποία με την σειρά της αποτελεί ένα fork της Tesseract υλοποιημένο ειδικά για ανάπτυξη εφαρμογών σε Android Studio. Το **Tesseract4Android** <https://github.com/adaptech-cz/Tesseract4Android> πριν την αναγνώριση χαρακτήρων εκτελεί μια επεξεργασία στην εικόνα που του δίνει ως είσοδο. Για την επεξεργασία της εικόνας γίνεται χρήση των βιβλιοθηκών Leptonica 1.81.1, libjpeg v9d, libpng 1.6.37.

2.2.2.4.1 Αρχιτεκτονική tesseract [12]

Η αρχιτεκτονική της μηχανής Tesseract μπορεί να γίνει αντιληπτή από την παρακάτω εικόνα.



Εικόνα 2.1 Αρχιτεκτονική μηχανής Tesseract [12]

Αρχικά, η μηχανή Tesseract θεωρεί ότι η εικόνα που λαμβάνει ως είσοδο είναι μια πολυγωνική δυαδική περιοχή(polygonal binary area). Στο βήμα Connected Component Analysis που βλέπουμε παρακάτω αποθηκεύονται τα περιγράμματα του στοιχείου τα οποία συγκεντρώνονται σε Blobs που αποθηκεύονται σε γραμμές κειμένου. Η Tesseract βρίσκει τα μπλοκ κειμένου μέσω του αλγορίθμου εύρεσης γραμμής.

Αλγόριθμος εύρεσης γραμμής

Ο συγκεκριμένος αλγόριθμος είναι έτσι διαμορφωμένος έτσι ώστε εικόνες που είναι ελαφρά περιστραμένες να μην χρειαστεί να πρέπει να περιστραφούν για να γίνει ορθή αναγνώριση. Το φιλτράρισμα των Blobs και η κατασκευή γραμμής είναι τα δυο χαρακτηριστικά της διαδικασίας αυτής. Με την υπόθεση ότι η πλειοψηφία των blobs έχουν ομοιόμορφο μέγεθος κειμένου ένα απλό φίλτρο percentile height μπορεί να αφαιρέσει drop-cups και περίπλοκων χαρακτήρων και το μέσο ύψος προσεγγίζει το μέγεθος του κειμένου στο μπλοκ. Τα μικρά blobs που είναι μικρότερα από ένα κλάσμα του μέσου ύψους φιλτράρονται επειδή συνήθως αποτελούν τόνους, αποστρόφους και γενικά θόρυβο. Εν συνέχεια, η Tesseract πραγματοποιεί επεξεργασία των φιλτραρισμένων blobs σε οριζόντια θέση και τα εκχωρεί σε μια γραμμή κειμένου. Με την εικώρηση των blobs πρέπει να τοποθετηθούν και πάλι στην κατάλληλη θέση.

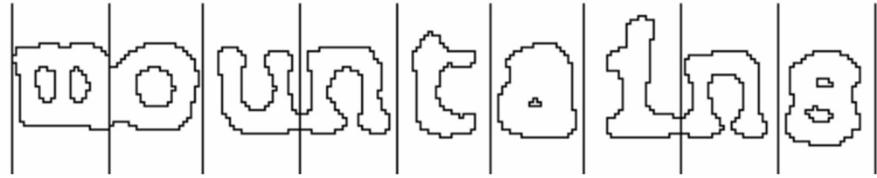
Μετά τον εντοπισμό των γραμμών κειμένου οι βασικότερες τοποθετούνται με μεγαλύτερη ακρίβεια κάνοντας χρήση μιας quadratic spline. Οι γραμμές βάσης τοποθετούνται με βάση την ομαδοποίηση των blobs με λογική συνεχή μετατόπιση για την αρχική ευθεία γραμμή βάσης όπως φαίνεται παρακάτω.

Ένα σημαντικό μειονέκτημα που μπορεί να συναντηθεί είναι ότι η ύπαρξη πολλών αυλακώσεων μπορεί να επιφέρει ασυνέχειες. Στην παρακάτω εικόνα οι 3 χρωματιστές γραμμές αναπαριστούν τον τρόπο εντοπισμού της γραμμής κειμένου. Αναλυτικότερα, η πράσινη γραμμή αποτελεί τη γραμμή βάσης, η μπλε τη γραμμή κάτω αρχής, η ροζ τη μέση γραμμή και το γαλάζιο τη γραμμή ανάβασης.

Volume 69, pages 872–879.

Εικόνα 2.2 Ένα παράδειγμα όπου η βασική γραμμή καμπυλώνει από Ray [12]

Μετά τον εντοπισμό της περιοχής κειμένου η Tesseract έρχεται να τμηματοποιήσει το κείμενο χαρακτήρα-χαρακτήρα με σκοπό να βρει σταθερό πλάτος στους χαρακτήρες.

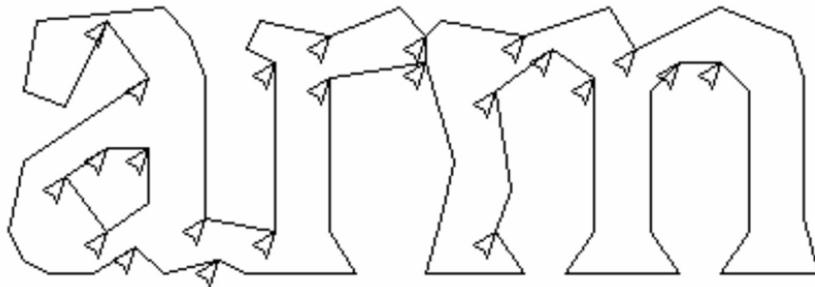


Εικόνα 2.2 Μια τεμαχισμένη λέξη σταθερού πλάτους από Ray [12]

Εφόσον βρεθεί το κείμενο το επιθυμητό σταθερό πλάτος ξεκινά το επόμενο βήμα δηλαδή η διαδικασία αναγνώρισης των λέξεων.

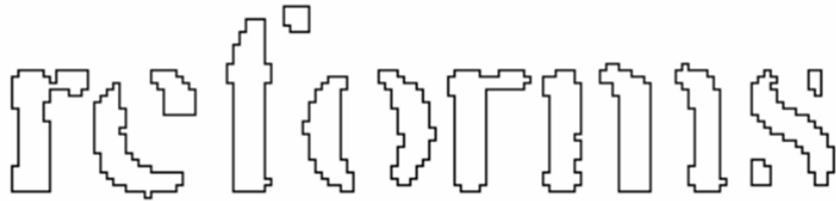
ΑΝΑΓΝΩΡΙΣΗ ΛΕΞΗΣ

Αρχικά, γίνεται βελτίωση του αποτελέσματος εάν δεν είναι το ικανοποιητικό με τεμαχισμό των blobs. Έχει ως στόχο την εύρεση σημείων στην λέξη που μπορούν να κοπούν, δηλαδή σημεία με κοίλες κορυφές μιας πολυγωνικής προσέγγισης του περιγράμματος. Με αυτό τον τρόπο καταλαβαίνει ότι δυο χαρακτήρες μπορεί να είναι ενωμένοι και ότι δεν αποτελούν έναν. Για παράδειγμα στην παρακάτω εικόνα η Tesseract καταλαβαίνει ότι το γράμμα "r" και το γράμμα "m" αποτελούν διαφορετικούς χαρακτήρες.



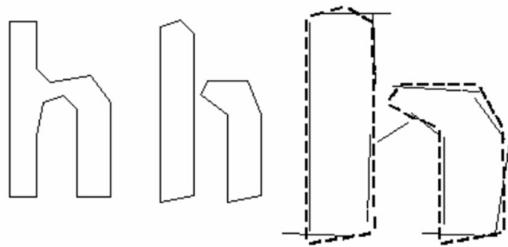
Εικόνα 2.4 Σημεία υποψήφια κοπής [12]

Στην περίπτωση που το αποτέλεσμα δεν αποτελεί το ικανοποιητικό ακόμα και μετά της εύρεση των σημείων κοπή τότε ο έρχεται στο προσκήνιο ο Associator. Ο associator χωρίζει το γράφημα σε υποψήφιους χαρακτήρες προς αναγνώριση. Στην παρακάτω εικόνα μπορούμε να δούμε μια λέξη που έχει χωριστεί στους υποψήφιους χαρακτήρες.



Εικόνα 2.5 Χωρισμένη λέξη έτοιμη να αναγνωριστεί [12]

Επίσης τα χαρακτηριστικά ενός τεμαχισμένου χαρακτήρα εξάγονται από το τεμαχισμό του περιγράμματος από τον στατικό αταξινόμητή.



Εικόνα 2.6 Χαρακτηριστικά γνωρίσματα και αντιστοίχιση ταίριασμα Ray [12]

Τέλος, θα ήταν εύλογο να αναφερθεί ότι μετά την τημηματοποίηση του χαρακτήρα, πραγματοποιείται μια διαδικασία δύο φάσεων. Συγκεκριμένα, στην πρώτη φάση, η Tesseract στοχεύει στην αναγνώριση λέξη-λέξη και ο επιθυμητός χαρακτήρας περνάει στον δυναμικό ταξινομητή ως δεδομένο εκπαίδευσης. Εν συνεχείᾳ στην δεύτερη φάση λέξεις που θεώρησε η Tesseract πως δεν είχαν αναγνωριστεί σε ικανοποιητικό βαθμό ακολουθούν επαναληπτική διαδικασία αναγνώρισης.

2.2.3 OpenCV

Η OpenCV(Open-source Computer Vision) αποτελεί μια βιβλιοθήκη υπολογιστικής όρασης και περιέχει συναρτήσεις επεξεργασίας εικόνας σε πραγματικό χρόνο. Πρωτοεμφανίστηκε το 1999 από την Intel αλλά στην συνέχεια υποστηρίχθηκε από την Willow Garage. Η OpenCV είναι μια βιβλιοθήκη ανοιχτού κώδικα, γραμμένη σε C++, και είναι η πιο ευρέως γνωστή βιβλιοθήκη σε θέματα υπολογιστικής όρασης. Σημαντικό είναι να αναφέρουμε ότι υποστηρίζει συστήματα βαθιάς μάθησης, π.χ. Tensorflow.

Η συγκεκριμένη βιβλιοθήκη σήμερα έχει πάνω από 500 συναρτήσεις χρήσιμες για διάφορες εφαρμογές στον τομέα την υπολογιστική όρασης. Σχεδιάστηκε με γνώμονα την εφαρμογή επεξεργασίας εικόνας σε πραγματικό χρόνο.

Σχεδόν σε κάθε λογισμικό ή εφαρμογή που χρειάστηκε την εφαρμογή επεξεργασίας εικόνας έχει χρησιμοποιηθεί η OpenCV. Κάποια τέτοια παραδείγματα εφαρμογής της βιβλιοθήκης είναι:

- Αναγνώριση προσώπων/αντικειμένων
- Αναγνώριση συναισθημάτων
- Αυτόματα συστήματα παρακολούθησης
- Συστήματα ελέγχου παραγωγής
- Συρραφή εικόνων από δορυφόρους και διαδικτυακούς χάρτες
- Μείωση του θορύβου σε ιατρικές εικόνες
- Αντίληψη βάθους (με 2 κάμερες)

2.2.4 Επαυξημένη Πραγματικότητα (AR)

2.2.4.1 Επισκόπηση AR

Η Επαυξημένη Πραγματικότητα κάνει εφικτή την συνύπαρξη ενός εικονικού κόσμου με τον πραγματικό αποσκοπώντας την πραγματική εμπειρία του χρήστη. Η παρούσα τεχνολογία δίνει την δυνατότητα στον προγραμματιστή και στον χρήστη να εισάγει ψηφιακά αντικείμενα στο φυσικό περιβάλλον σε πραγματικό χρόνο. Αποτελεί ένα είδος διαδραστικού περιβάλλοντος που βασίζεται σε επαυξημένα στοιχεία του αληθινού κόσμου τα οποία προέρχονται από συσκευές υπολογιστών, όπως ήχος, βίντεο, γραφικά ή δεδομένα τοποθεσίας.

Ο χρήστης μπορεί να συναντήσει την Επαυξημένη Πραγματικότητα μέσα από τις ποικίλες εφαρμογές της. Η χρήση της επαυξημένης πραγματικότητας θα μπορούσε να γίνει εμφανής σε τρεις κύριες περιπτώσεις, την Τρισδιάστατη θέαση επαυξημένης πραγματικότητας, τα Προγράμματα περιήγησης επαυξημένης πραγματικότητας και τα Παιχνίδια.

Στην Τρισδιάστατη θέαση επαυξημένης πραγματικότητας δίνεται στον χρήστη η δυνατότητα τοποθέτησης μοντέλων τριών διαστάσεων πραγματικού μεγέθους στο φυσικό του περιβάλλον. Όπως για παράδειγμα η γνωστή εφαρμογή γνωστής αλυσίδας επίπλων μέσω της οποίας ο χρήστης μπορεί να τοποθετήσει εικονικά έπιπλα μέσω ενός έξυπνου κινητού στο πραγματικό του περιβάλλον.

Η βασική ιδέα των Προγραμμάτων περιήγησης επαυξημένης πραγματικότητας είναι η εισαγωγή πληροφοριών σε μορφή επαυξημένης πραγματικότητας στην οθόνη της κάμερας. Για παράδειγμα, η εφαρμογή που έχει υλοποιηθεί στην παρούσα

διπλωματική αποτελεί ένα τέτοιο πρόγραμμα. Αναλυτικότερα, ο χρήστης χρησιμοποιώντας την μέσω ενός έξυπνου κινητού είναι σε θέση να δει πληροφορίες, σχετικές με τα λογότυπα που έχει σαρώσει, στην οθόνη του μέσω την τεχνολογία της επαυξημένης πραγματικότητας.

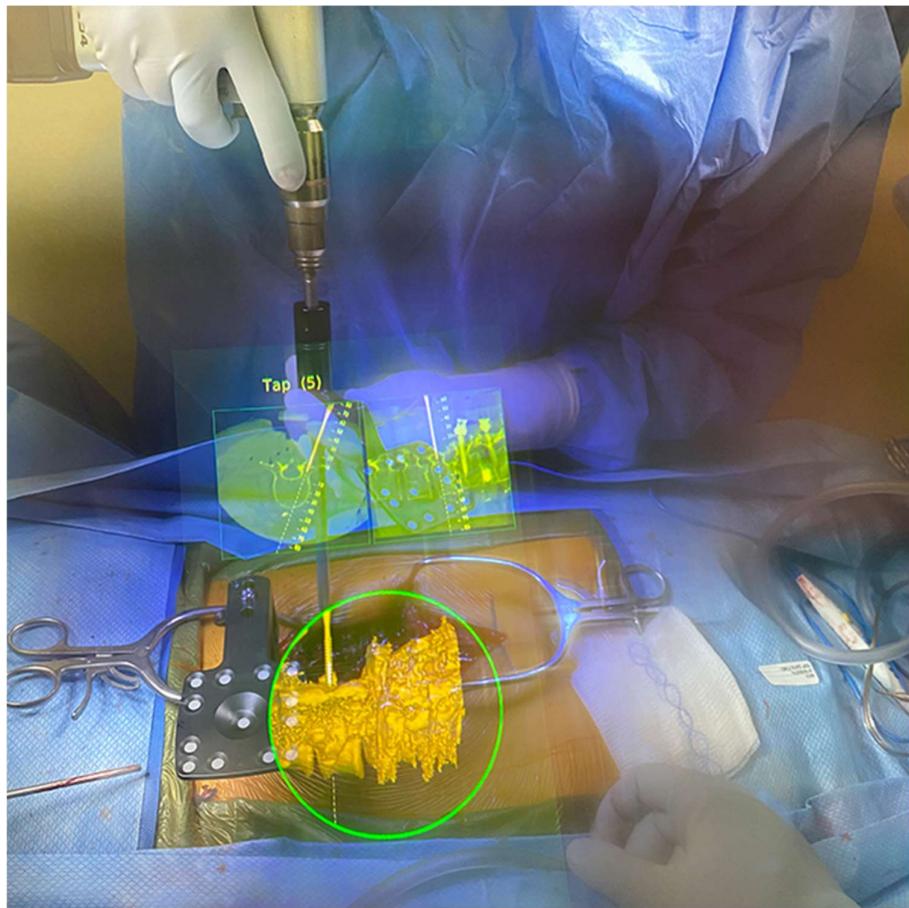
Η τελευταία κατηγορία αφορά τα παιχνίδια επαυξημένης πραγματικότητας τα οποία παρέχουν στον παίκτη μια εμπειρία μεταξύ του πραγματικού και του ψηφιακού κόσμου. Ευρέως γνωστό παράδειγμα είναι η εφαρμογή Pokémon Go που διαδόθηκε παγκοσμίως ιδιαίτερα στους λάτρεις των Pokémon. Βασική ιδέα της οποίας ήταν να μεταφέρει στον χρήστη την εμπειρία ενός κυνηγού Pokémon, αναζητώντας κρυμμένα Pokémon στον χάρτη του πραγματικού κόσμου.

2.2.4.2 Εφαρμογές στην Καθημερινότητα

Η επαυξημένη πραγματικότητα έχει αρχίσει να γίνεται κομμάτι τόσο τις καθημερινότητας των απλών πολιτών αλλά όσο και των επαγγελματιών και των επιστημών. Μπορεί να συναντηθεί σε εφαρμογές ψυχαγωγίες αλλά και σε εφαρμογές υλοποιημένες για κλάδους υγείας ή άλλους επαγγελματικούς κλάδους.

Στην **Επιστήμη της Υγείας** έχουν δημιουργηθεί εφαρμογές με εκπαιδευτικό χαρακτήρα που αφορούν τους φοιτητές ιατρικής αλλά και άλλες που είναι ειδικά σχεδιασμένες για χρήση κατά την διάρκεια χειρουργείων και άλλων επεμβάσεων. Σε στιγμές κρίσιμων καταστάσεων οι εφαρμογές επαυξημένης πραγματικότητας μπορούν να παρέχουν πληροφορίες σε πραγματικό χρόνο στην περιοχή θεραπείας για να υποστηρίξουν τα διαγνωστικά, χειρουργικά και θεραπευτικά πλάνα των γιατρών.

Η επαυξημένη πραγματικότητα μπορεί να παίξει σημαντικό ρόλο στην **σύγχρονη χειρουργική**. Καθώς οι χειρουργοί έχουν την δυνατότητα να προγραμματίσουν με ακρίβεια την διαδικασία πριν γίνει η πρώτη τομή στον ασθενή. Η πρώτη επιτυχής χειρουργική επέμβαση στην οποία χρησιμοποιήθηκε επαυξημένη πραγματικότητα έγινε από την ομάδα νευροχειρουργών του Johns Hopkins τον Φεβρουάριο του 2021.



Εικόνα 2.7 εγχείρηση με χρήση επαυξημένης πραγματικότητας

Ένας άλλος τομέας που λαμβάνει μέρος η Επαυξημένη Πραγματικότητα είναι ο χώρος των τεχνών και της θέασης. Στις μέρες μας υπάρχουν πολλά μουσεία και εκθέσεις που έχουν εκθέματα βασισμένα σε επαυξημένη πραγματικότητα. Γνωστό παράδειγμα είναι το **Muséum national d'Histoire naturelle** στο Παρίσι, που τον Ιούνιο του 2021 χρησιμοποιώντας το **HoloLens** της **Microsoft** δημιούργησε το Project “REVIVRE” (“Ζήσε ξανά”) που δίνει την δυνατότητα στους επισκέπτες να έρχονται σε επαφή με εικονικά ζώα.



Eικόνα 2.8 Muséum national d'Histoire naturelle

2.2.4.3 Αναπτυξιακά εργαλεία

Για την υλοποίηση εφαρμογών Επαυξημένης Πραγματικότητας έχουν δημιουργηθεί πολλά αναπτυξιακά εργαλεία τόσο για εφαρμογές σε περιβάλλοντα Android όσο και για εφαρμογές σε περιβάλλοντα iOS. Με τα εύχρηστα και υψηλής αποδοτικότητας εργαλεία που έχουν υλοποιήσει οι δύο ναυαρχίδες Apple και Google οι προγραμματιστές είναι σε θέση να δημιουργήσουν ψυχαγωγικές και καινοτόμες ιδέες πάνω στην επαυξημένη πραγματικότητα. Τα δύο πιο ευρέως γνωστά εργαλεία είναι το ARCore από την Google και το ARKit από την Apple που είναι διαθέσιμα στο κοινό από το 2018.

ARCore

Το Arcore γνωστό και ως **Google Play Services for AR** είναι ένα εργαλείο ανάπτυξης λογισμικού που δημιουργήθηκε από την Google για την ανάπτυξη εφαρμογών επαυξημένης πραγματικότητας. Το ARCore επικεντρώνεται σε τρεις τομείς για να φέρει την επαυξημένη πραγματικότητα σε κινητά τηλέφωνα Android την παρακολούθηση της κίνησης, την κατανόηση του περιβάλλοντος και την εκτίμηση του φωτός.

Η παρακολούθηση της κίνησης γίνεται επιτυχής με τη χρήση της κάμερας του έξυπνου κινητού για την παρακολούθηση σημείων στον χώρο και μετρήσεων της αδράνειας από τον αισθητήρα IMU της συσκευής. Η συγκεκριμένη διαδικασία είναι γνωστή ως αδρανειακή οδομετρία (visual inertial odometry ή VIO) και με βάση αυτήν το ARCore προσδιορίζει με ακρίβεια την θέση και τον προσανατολισμό του έξυπνου κινητού σε σχέση με τον χώρο.

Ο όρος της **Περιβαλλοντικής κατανόησης** σημαίνει ότι υπάρχει δυνατότητα τοποθέτηση αντικείμενών και μοντέλων επαυξημένης πραγματικότητας σε επιφάνειες, όπως ένα δάπεδο ή ένα τραπέζι. Το ARCore μπορεί να εντοπίσει τις οριζόντιες επιφάνειες κάνοντας χρήση των ίδιων χαρακτηριστικών τους που χρησιμοποιεί για την παρακολούθηση της κίνησης.

Η **εκτίμηση του φωτός** παρέχει την δυνατότητα στο ARCore να παρακολουθεί τον φωτισμό στο πραγματικό περιβάλλον και στους προγραμματιστές να δίνουν το κατάλληλο ποσοστό φωτός στα ψηφιακά αντικείμενα έτσι ώστε να φαίνονται πιο ρεαλιστικά σε σχέση με το πραγματικό περιβάλλον.

ARKit

Το ARKit είναι ένα εργαλείο ανάπτυξης λογισμικού που δημιουργήθηκε από την Apple για την ανάπτυξη εφαρμογών επαυξημένης πραγματικότητας σε iOS περιβάλλοντα. Το ARKit επιτρέπει στους προγραμματιστές να αναπτύξουν εφαρμογές AR υψηλής ευκρίνειας συμβατές σε συσκευές iPhone και Ipad. Αντικείμενα τριών διαστάσεων και άλλα μοντέλα που έχουν προστεθεί σε κάποιο περιβάλλον μπορούν να είναι ορατά και από κάποια άλλη συσκευή.

Το ARKit είναι συμβατό σε κινητές συσκευές που έχουν iOS 11 και τις επόμενες versions. Για αυτό τον λόγο η χρήση του Core A9 δίνει την δυνατότητα μεγάλης ακρίβειας, λεπτομέρειας και ρεαλισμού των αντικειμένων τριών διαστάσεων που χρησιμοποιούνται. Με την συσκευή iPhone X το ARKit μπορεί να πραγματοποιήσει αναγνώριση προσώπου σε πραγματικό χρόνο και χρησιμοποιώντας τα συγκεκριμένα δεδομένα να ελέγχει τις εκφράσεις ενός εικονικού χαρακτήρα τριών διαστάσεων.

Χρησιμοποιώντας την **κάμερα της συσκευής iOS, τα επιταχυνσιόμετρα, το γυροσκόπιο και την ευαισθητοποίηση περιβάλλοντος**, το ARKit εκτελεί την χαρτογράφηση του περιβάλλοντος κατά την μετακίνηση της συσκευής. Τα δεδομένα αδρανειακού αισθητήρα με τα δεδομένα από την κάμερα επιτρέπουν τη χαρτογράφηση υψηλής ακρίβειας. Το λογισμικό εντοπίζει χαρακτηριστικά στο περιβάλλον, όπως είναι τα επίπεδα και παρακολουθεί την κίνηση σε συνδυασμό με πληροφορίες από τους

αδρανειακούς αισθητήρες. Η κάμερα χρησιμοποιείται επίσης για τον προσδιορισμό πηγών φωτός με τις οποίες φωτίζονται αντικείμενα επαυξημένης πραγματικότητας.

Η λύση της Apple στην **αυξημένη λεπτομέρεια** και συνεπώς στη **χρήση μνήμης** είναι ένας συρόμενος χάρτης. Πιο συγκεκριμένα, τα παλιά δεδομένα εξαφανίζονται για να εμφανιστούν τα νέα. Οι χρήστες μπορούν να τοποθετήσουν archors σε συγκεκριμένες τοποθεσίες για να επισημάνουν τα αντικείμενα που θέλουν να αποθηκεύσουν.

ARCore

Για την υλοποίηση της παρούσας εφαρμογής χρησιμοποιήθηκε το ARCore καθώς αποτελεί το πιο αξιόλογο εργαλείο για την ανάπτυξη εφαρμογών επαυξημένης πραγματικότητας σε περιβάλλοντα Android. Βασικό ρόλο στην επιλογή μας έπαιξε η ακρίβεια που παρέχεται όσον αφορά την τοποθεσία των αντικειμένων στο περιβάλλον και η λεπτομέρεια των αντικειμένων που κάνει τα αντικείμενα ιδιαίτερα ρεαλιστικά.

Sceneform

Το Sceneform αποτελεί ένα SDK ανοιχτού κώδικα του Android. Το συγκεκριμένο SDK χρησιμοποιείται σε συνδυασμό με το ARCore και καθιστά απλή την απόδοση ρεαλιστικών σκηνών τριών διαστάσεων σε εφαρμογές επαυξημένης πραγματικότητας χωρίς την χρήση καθαρής OpenGL από τον προγραμματιστή. Με το Sceneform ο προγραμματιστής έχει την δυνατότητα να εισάγει αντικείμενα τριών διαστάσεων στην σκηνή.

Δυστυχώς στις νεότερες εκδόσεις του Android Studio 4+ το Sceneform πλέον έχει καταργηθεί και μπορεί να χρησιμοποιηθεί μόνο σε projects υλοποιημένα στο περιβάλλον Unity. Για αυτό τον λόγο χρησιμοποιήσαμε το "Sceneform Maintained SDK for Android" [ThomasGorissey/SceneformMaintained: Sceneform Maintained SDK for Android \(github.com\)](https://ThomasGorissey/SceneformMaintained: Sceneform Maintained SDK for Android (github.com)) που έχει δημιουργήσει ο Thomas Gorisse που αποτελεί ενα fork του Sceneform και είναι συμβατό σε εκδόσεις του Android Studio 4+. Το "Sceneform Maintained SDK for Android" υποστηρίζει αντικείμενα τύπου glTF.

2.2.5 Scrappy Framework

Η αυτόματη συλλογή δεδομένων από το διαδίκτυο αποτελεί μια πολύ σημαντική λειτουργία για την απόκτηση πολλών δεδομένων που ενδιαφέρουν έναν προγραμματιστή μέσω κάποιων απλών αλλά και κάποιων περίπλοκων βημάτων. Συχνά πραγματοποιείται αυτόματη συλλογή δεδομένων που είναι απαραίτητα για την αρχή κάποιας έρευνας ή ακόμα και την μιας απλής βάσης δεδομένων.

Για την επίτευξη αυτού έχουν δημιουργηθεί διάφορα frameworks και βιβλιοθήκες που μπορούν να μας βοηθήσουν. Ένα από αυτά είναι το Scrappy framework

το οποίο είναι ανοιχτού κώδικα και γραμμένο σε Python. To scrappy χρησιμοποίει έναν ενσωματωμένο μηχανισμό ονόματι Selectors για την εξαγωγή δεδομένων από ιστοσελίδες. Επίσης παρέχει την δυνατότητα στον προγραμματιστή να εξάγει τα επιθυμητά δεδομένα σε μορφή JSON, CSV, και XML. Διαθέτει ενσωματωμένη υποστήριξη για την επιλογή και την εξαγωγή δεδομένων από πηγές είτε με εκφράσεις XPath είτε με CSS. Για την εξαγωγή δεδομένων ο προγραμματιστής χρησιμοποιεί κάποιες κλάσεις ονόματι Spiders οι οποίες είναι υπεύθυνες για τον τρόπο παρακολούθησης του πηγαίου κώδικα ενός ιστότοπου για την εξαγωγή των επιθυμητών πληροφοριών. Τέλος, το Scrapy αποτελεί το πιο γνωστό framework για εξαγωγή δεδομένων από περίπλοκες ιστοσελίδες.

2.2.6 SQLite - SQLiteStudio

Το SQLite αποτελεί ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων. Αντιθέτως με άλλα τέτοια συστήματα αλλά μορφής Πελάτη-Διακοσμητή η συγκεκριμένη μηχανή δεν έχει αυτόματες διεργασίες για την επικοινωνία του προγράμματος της εφαρμογής. Το SQLite αποτελεί ένα κομμάτι του προγράμματος, δηλαδή ενσωματωμένη στον κώδικα. Ο προγραμματιστής μπορεί να χρησιμοποιήσει την SQLite με την βοήθεια απλών κλήσεων συναρτήσεων οι οποίες επιτυγχάνουν ταχύτερη πρόσβαση στην βάση δεδομένων. Η αποθήκευση της βάσης δεδομένων πραγματοποιείται στην μηχανή ενός host. Σημαντικό είναι να σημειωθεί ότι οι εγγραφές δεν μπορούν να γίνουν ανακατεμένα παρά μόνο με τη σειρά.

Για την δημιουργία μιας βάσης δεδομένων με το SQLite μας παρέχε το SQLiteStudio το οποίο αποτελεί μια desktop εφαρμογή ειδικά υλοποιημένη για περιήγηση και επεξεργασία αρχείων βάσεων δεδομένων SQLite. Έχει δημιουργηθεί από τον Paweł Salawa και είναι γραμμένο σε C++.

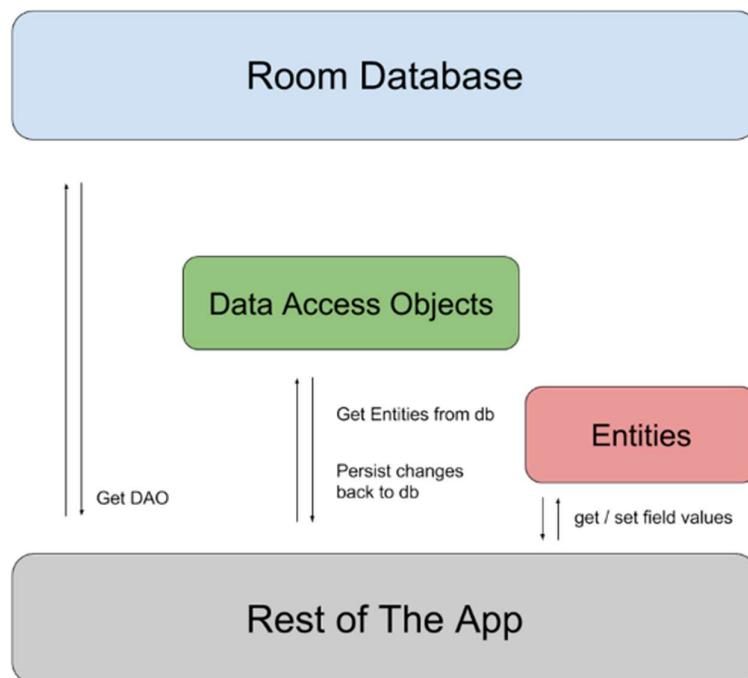
2.2.7 RoomDatabase

Η Room αποτελεί μια βιβλιοθήκη σχεδιασμένη για δημιουργία και διαχείριση βάσεων δεδομένων SQLite σε android projects. Η συγκεκριμένη βιβλιοθήκη παρέχει πολλά οφέλη και για αυτό τον λόγο είναι και ευρέως διαδεδομένη. Ένα βασικό πλεονέκτημα της Room είναι η εύκολη διαχείριση της βάσης που προσφέρει στον προγραμματιστή χωρίς να χρειάζεται να γράψει πολλές και περίπλοκες γραμμές κώδικα. Επίσης προσφέρει γρήγορες και εύκολες τεχνικές μετεγκατάστασης βάσεων μετά από αλλαγές. Τέλος, παρέχει επικύρωση χρόνου μεταγλωττισης των ερωτημάτων SQL. Αυτό σημαίνει ότι μια εφαρμογή δεν θα μεταγλωττιστεί εάν υπάρχει σφάλμα σε κάποιο ερώτημα.

Πρωτογενή συστατικά της Room είναι τρία και συγκεκριμένα τρεις κλάσεις:

- Η κλάση database που περιέχει την βάση δεδομένων του συστήματος και αποτελεί το κύριο σημείο πρόσβασης για την υποκείμενη σύνδεση με τα δεδομένα της εφαρμογής.
- Τις κλάσεις Entities ή αλλιώς οντότητες δεδομένων που αντιπροσωπεύουν τους πίνακες της βάσης δεδομένων της εφαρμογής.
- Η κλάση Dao της οποίας τα αντικείμενα παρέχουν πρόσβαση στην βάση, δηλαδή περιέχει τα queries της εφαρμογής.

Το παρακάτω σχήμα απεικονίζει τη σχέση μεταξύ των διαφορετικών στοιχείων της Room.



Εικόνα 2.8 Σχέση μεταξύ πρωτογενών συστατικών της Room Library

LiveData

Η Room χρησιμοποιεί δεδομένα τύπου **LiveData**. Η κλάση LiveData είναι μια κλάση Data holder που μπορεί να παρατηρείται. Το πλεονέκτημα των livedata είναι ότι είναι ενήμερα για τον κύκλο ζωής της εφαρμογής που σημαίνει ότι σέβονται τον κύκλο ζωής άλλων στοιχείων της εφαρμογής όπως είναι τα activities, fragments κ.α. Αυτό μας διασφαλίζει ότι τα LiveData ενημερώνουν observers (αντικείμενα της κλάσης Observer) που βρίσκονται σε κάποια κατάσταση που έχει ενεργό κύκλο ζωής την εκάστοτε στιγμή.

Παρακάτω βλέπουμε δύο από τα πολλά **οφέλη** που μας παρέχουν τα LiveData:

- Εξασφαλίζει ότι το περιβάλλον χρήστη UI ταιριάζει με την κατάσταση των δεδομένων. Ακολουθούν το observer pattern
- Οι observers συνδέονται με αντικείμενα lifecycle και καθαρίζονται κάθε φορά που τελειώνει ο σχετικός κύκλος ζωής τους. Αυτό βοηθάει στην αποφυγή υπερφόρτωσης της μνήμης.

2.3 Σύγκριση εφαρμογής με ήδη υπάρχουσες

Στις μέρες μας, με την ταχεία ανάπτυξη της τεχνολογίας προσπαθούμε συνεχώς να βρούμε τρόπους να εξαλείψουμε παλιές χρονοβόρες διαδικασίες και να κάνουμε την ζωή του χρήστη πιο εύκολη. Η ανάκτηση πληροφορίας πρέπει να πραγματοποιείται πιο γρήγορα από τεχνικές παραδοσιακών μηχανών αναζήτησης.

Η οπτική αναγνώριση χαρακτήρων έρχεται να μας βοηθήσει σε αυτό καθώς μέσω του εντοπισμού κειμένου με την βοήθεια κάμερας ή φωτογραφιών ο χρήστης πλέον δεν χρειάζεται να πληκτρολογήσει το θέμα αναζήτησης του καθώς υπάρχει ήδη έτοιμο. Μια τέτοια εφαρμογή αποτελεί το **Google Lens**, δημιουργημα της Google με διάφορες λειτουργίες μια από τις οποίες έχει ως στόχο την ανάκτηση πληροφορίας από τον ιστό σχετική με κείμενο που έχει σκαναριστεί. Η κύρια διαφορά της παρούσας εφαρμογής σε σχέση με την συγκεκριμένη λειτουργία του Google Lens είναι ότι η αναγνώριση χαρακτήρων γίνεται σε πραγματικό χρόνο χωρίς να κλείσει καθόλου η κάμερα της εφαρμογής. Επίσης όσον αφορά την εμφάνιση της πληροφορίας, στην περίπτωση μας χρησιμοποιείται μια εικονική πινακίδα επαυξημένης πραγματικότητας με πληροφορίες από μια τοπική βάση ενώ το google lens κάνει κάποια σχετική αναζήτηση στο google και εμφανίζει τα όποια αποτελέσματα.

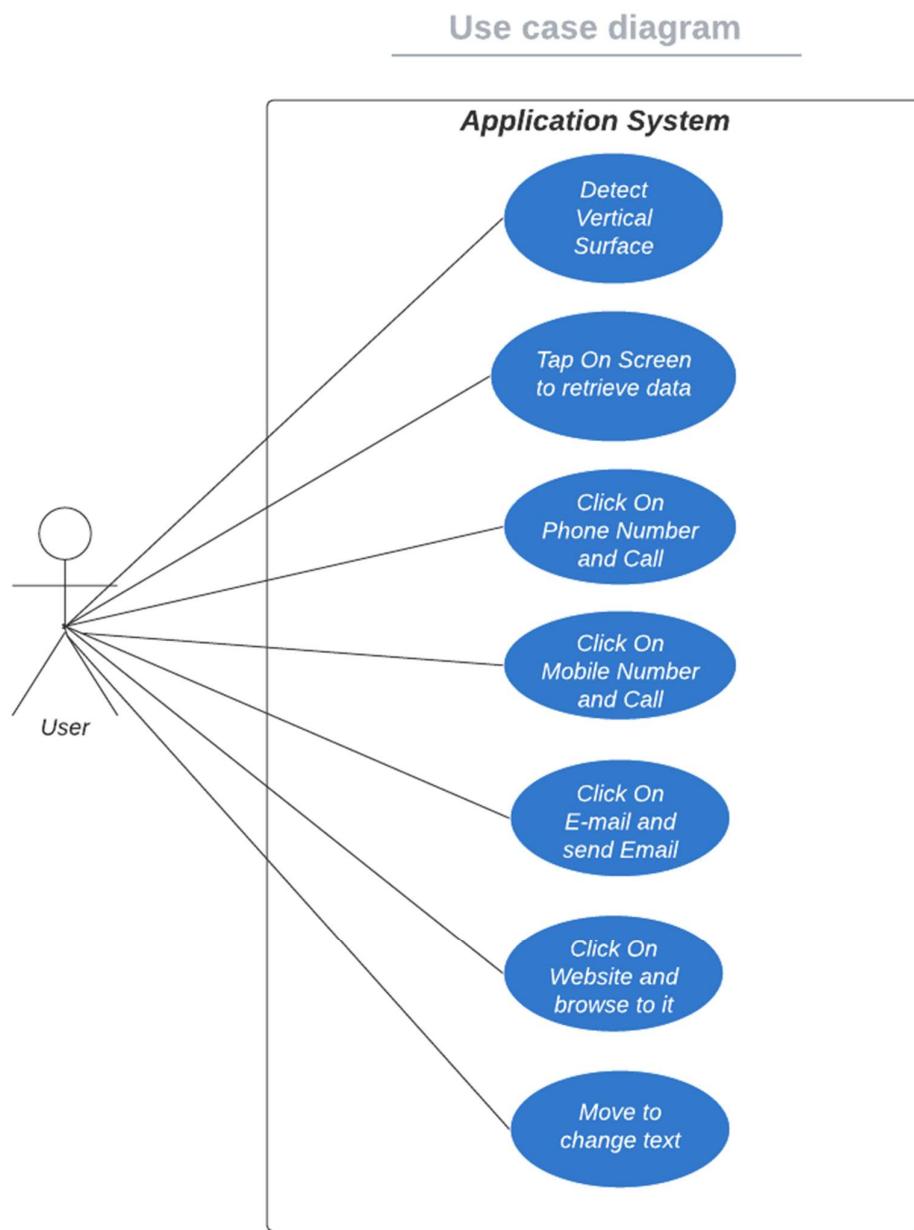
Η δική μας ιδέα στοχεύει στην ελάχιστη ως μηδενική συμμετοχή του χρήστη στην αναζήτηση συγκεκριμένης πληροφορίας χωρίς περιθώριο λαθών αναζήτησης. Για αυτό τον λόγο δημιουργήσαμε μια εφαρμογή βασισμένη στην απλότητα και στην διευκόλυνση του χρήστη εύρεσης και χρησιμοποίησης των απαραίτητων επιθυμητών πληροφοριών σε ελάχιστο χρόνο. Ένα αξιόλογο παράδειγμα είναι η εμφάνιση του email ενός επαγγελματία στην εικονική πινακίδα που δίνει την δυνατότητα στον χρήστη με ένα πάτημα να μεταφερθεί στο email app του κινητού του και να αποστείλει το επιθυμητό email στην παραπάνω διεύθυνση που έχει συμπληρωθεί αυτόματα. Φυσικά δεν θα έπρεπε να παραλείψουμε ότι η συγκεκριμένη εφαρμογή είναι απολύτως επεκτάσιμη και

με κάποιες αλλαγές στην βάση και ίσως στον τρόπο αναζήτησης μπορεί να επεκταθεί για εμφάνιση άλλων πληροφοριών π.χ. Ταμπέλες από μαγαζιά ρούχων.

Το κομμάτι της επαυξημένης πραγματικότητας στην εφαρμογή μας θα μπορούσε να συγκριθεί με το **Google Translate Scan** που αποτελεί και λειτουργία του google lens. Στην συγκεκριμένη εφαρμογή ο χρήστης σαρώνει κείμενο μέσω της κάμερας και σε πραγματικό χρόνο το κείμενο εμφανίζεται μεταφρασμένο πάνω από το σκαναρισμένο με τεχνικές επαυξημένης πραγματικότητας. Όσο ο χρήστης κινεί την κάμερα το κείμενο αλλάζει συνεχώς γίνεται μετάφραση σε κάθε frame. Η διαφορά με την δική μας εφαρμογή είναι ότι ο χρήστης πρέπει να πατήσει μια φορά πάνω στην οθόνη στο λογότυπο που τον ενδιαφέρει και μετά ακολουθεί η διαδικασία ανάκτησης πληροφορίας. Αυτό γίνεται επειδή δεν θέλουμε να αναγνωρίζονται συνεχώς χαρακτήρες σε κάθε frame διότι θέλουμε η εικονική πινακίδα να είναι σταθερή στην οθόνη για όση ώρα ο χρήστης θελήσει να χρησιμοποιεί τις περιγραφόμενες πληροφορίες.

2.4 Ανάλυση απαιτήσεων

2.4.1 Use Cases



Εικόνα 2.9 Use Case Diagram

Ακολουθούν οι απαιτήσεις της εφαρμογής υπό τη μορφή σεναρίων χρήσης (use cases).

Σενάριο Χρήσης: Detect Vertical Surface

| Αναγνωριστικό: ΣΧ01 |

Ηθοποιοί: Χρήστης
Προϋποθέσεις:
1) Ο χρήστης έχει επιστρέψει στην εφαρμογή να χρησιμοποιήσει την κάμερα και την τοποθεσία του
2) Η κάμερα έχει ανοίξει
Ροή γεγονότων:
1) Το σενάριο χρήσης ξεκινάει όταν ο χρήστης κινεί την κάμερα όπως του δείχνει το εικονικό χεράκι του ARCore
2) Ο χρήστης μετακινεί την κάμερα σε κάποια κάθετη επιφάνεια π.χ τοίχος και γίνεται εντοπισμός
Συνθήκες τέλους:
1) Ο πελάτης μπορεί να δει ένα εικονικό πλέγμα που συμβολίζει τον εντοπισμό της κάθετης επιφάνειας

Σενάριο Χρήσης: Tap on Screen to retrieve data
Αναγνωριστικό: ΣΧ02
Ηθοποιοί: Χρήστης
Προϋποθέσεις:
1) Ο πελάτης μπορεί να δει το εικονικό πλέγμα που συμβολίζει τον εντοπισμό της κάθετης επιφάνειας
Ροή γεγονότων:
1) Το σενάριο χρήσης ξεκινάει όταν ο χρήστης εντοπίζει με την κάμερα κάποιο λογότυπο που τον ενδιαφέρει
2) Ο χρήστης πλησιάζει στο λογότυπο όσο το δυνατόν πιο κοντά γίνεται
3) Ο χρήστης πατάει μια φορά πάνω στην οθόνη και η διαδικασία ανάκτησης δεδομένων ξεκινά
Συνθήκες τέλους:
1) Ο πελάτης μετά την τάχιστη ανάκτηση δεδομένων μπορεί να δει μια εικονική πινακίδα στην οθόνη του με όλες τις επιθυμητές πληροφορίες

Σενάριο Χρήσης: Click on Phone Number and call
Αναγνωριστικό: ΣΧ03
Ηθοποιοί: Χρήστης
Προϋποθέσεις:
1) Ο πελάτης μπορεί να δει την εικονική πινακίδα στην οθόνη του
Ροή γεγονότων:
1) Το σενάριο χρήσης ξεκινάει όταν ο χρήστης πατάει πάνω στο τηλέφωνο του γραφείου που είναι αναγραφόμενο στην εικονική ταμπέλα.
2) Ο χρήστης μεταφέρεται σε κάποια εφαρμογή που μπορεί να καλέσει κάποιον αριθμό τηλεφώνου
Συνθήκες τέλους:

1) Ο πελάτης μπορεί να καλέσει τον αριθμό και μετά να γυρίσει στην εφαρμογή και να προχωρήσει σε άλλες ενέργειες με την εικονική πινακίδα ή να ξαναεκτελέσει την use case ΣΧ02.

Σενάριο Χρήσης: Click on Mobile Number and call

Αναγνωριστικό: ΣΧ04

Ηθοποιοί: Χρήστης

Προϋποθέσεις:

1) Ο πελάτης μπορεί να δει την εικονική πινακίδα στην οθόνη του

Ροή γεγονότων:

1) Το σενάριο χρήσης ξεκινάει όταν ο χρήστης πατάει πάνω στο τηλέφωνο του γραφείου που είναι αναγραφόμενο στην εικονική ταμπέλα.

2) Ο χρήστης μεταφέρεται σε κάποια εφαρμογή που μπορεί να καλέσει κάποιον αριθμό τηλεφώνου

Συνθήκες τέλους:

1) Ο πελάτης μπορεί να καλέσει τον αριθμό και μετά να γυρίσει στην εφαρμογή και να προχωρήσει σε άλλες ενέργειες με την εικονική πινακίδα ή να ξαναεκτελέσει την use case ΣΧ02.

Σενάριο Χρήσης: Click on E-mail and send email

Αναγνωριστικό: ΣΧ05

Ηθοποιοί: Χρήστης

Προϋποθέσεις:

1) Ο πελάτης μπορεί να δει την εικονική πινακίδα στην οθόνη του

Ροή γεγονότων:

1) Το σενάριο χρήσης ξεκινάει όταν ο χρήστης πατάει πάνω στο email του γραφείου που είναι αναγραφόμενο στην εικονική ταμπέλα.

2) Ο χρήστης μεταφέρεται σε κάποια εφαρμογή ηλεκτρονικού ταχυδρομείου.

Συνθήκες τέλους:

1) Ο πελάτης μπορεί να στείλει email στην συγκεκριμένη διεύθυνση και μετά να γυρίσει στην εφαρμογή και να προχωρήσει σε άλλες ενέργειες με την εικονική πινακίδα ή να ξαναεκτελέσει την use case ΣΧ02.

Σενάριο Χρήσης: Click on Website and browse to it

Αναγνωριστικό: ΣΧ06

Ηθοποιοί: Χρήστης

Προϋποθέσεις:

1) Ο πελάτης μπορεί να δει την εικονική πινακίδα στην οθόνη του

Ροή γεγονότων:

1) Το σενάριο χρήσης ξεκινάει όταν ο χρήστης πατάει πάνω στην ιστοσελίδα του γραφείου που είναι αναγραφόμενη στην εικονική ταμπέλα.

2) Ο χρήστης μεταφέρεται στην ιστοσελίδα μέσω κάποιου browser του κινητού

Συνθήκες τέλους:

1) Ο πελάτης μπορεί περιηγηθεί στην ιστοσελίδα και μετά να γυρίσει στην εφαρμογή και να προχωρήσει σε άλλες ενέργειες με την εικονική πινακίδα ή να ξαναεκτελέσει την use case ΣΧ02.

Σενάριο Χρήστης : Move to change Text

Αναγνωριστικό: ΣΧ07

Ηθοποιοί: Χρήστης

Προϋποθέσεις:

1) Ο πελάτης μπορεί να δει την εικονική πινακίδα στην οθόνη του

Ροή γεγονότων:

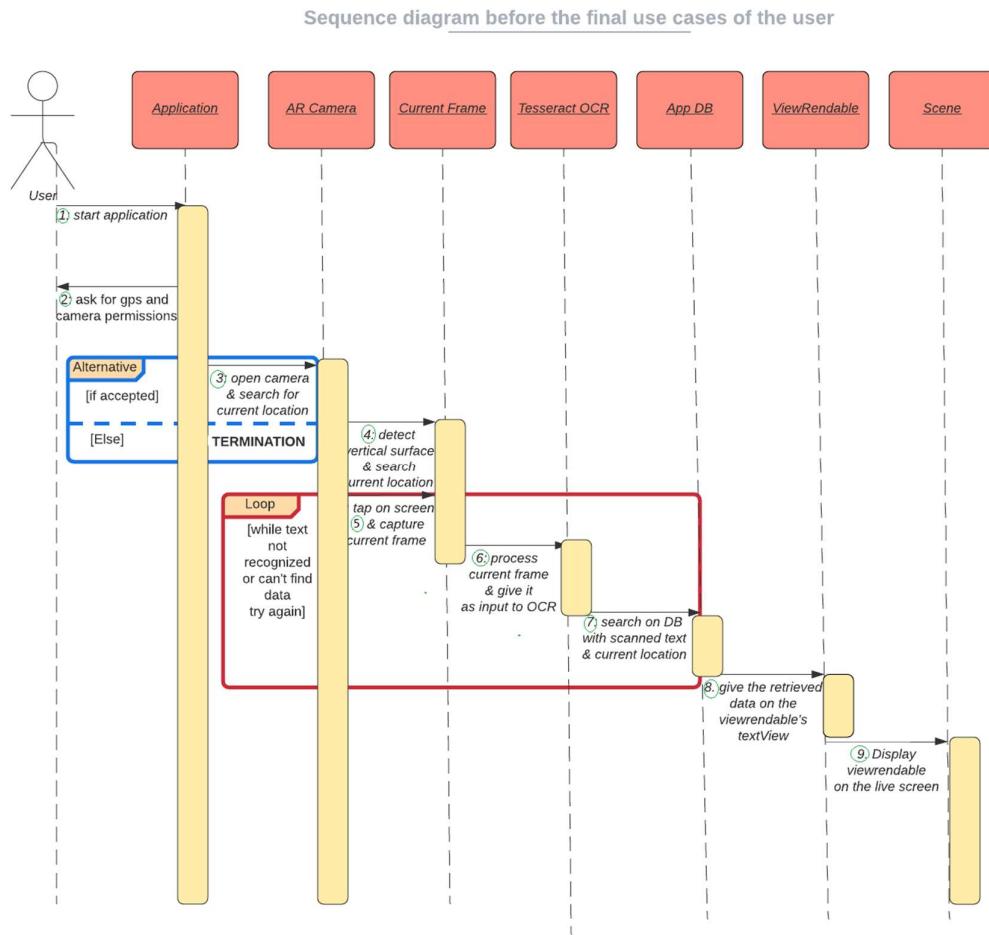
1) Το σενάριο χρήστης ξεκινάει όταν ο χρήστης μετακινείται μακριά από την εικονική πινακίδα και το κείμενο αλλάζει.

Συνθήκες τέλους:

1) Ο πελάτης μπορεί να δει μόνο το ονοματεπώνυμο του επαγγελματία αν βρίσκεται πάνω από 1 μέτρο μακριά από την πινακίδα και άμα πλησιάσει μπορεί να δει όλες τις επιθυμητές πληροφορίες και να ενεργήσει ανάλογα.

2.4.2 Sequence Diagram

Όπως αναφέρθηκε στην ΣΧ02 μόλις ο χρήστης πατήσει πάνω στην οθόνη ξεκινά μια ροή διαδικασιών για την ανάκτηση πληροφοριών προτού εμφανιστεί η εικονική πινακίδα. Παρακάτω μπορούμε να δούμε σε ένα sequence diagram την συγκεκριμένη διαδικασία.



Εικόνα 2.10 Sequence Diagram

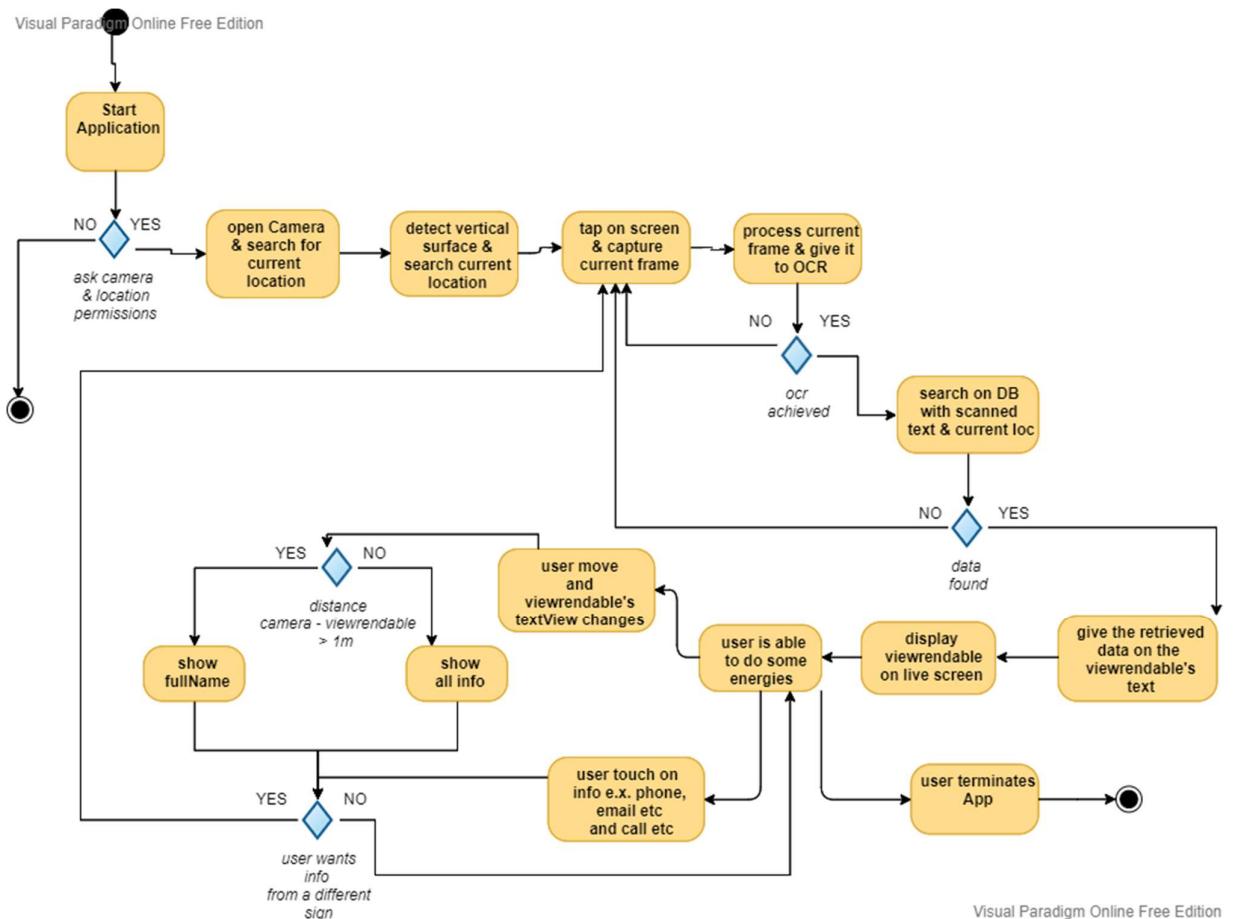
Αναλυτικότερα και σχετικά με το sequence diagram που βλέπουμε παραπάνω η διαδικασία έχει ως εξής:

1. Ο χρήστης ανοίγει την εφαρμογή.
2. Η εφαρμογή ζητάει από τον χρήστη να επιτρέψει χρήση της κάμερας και της τοποθεσίας. Ο χρήστης επιτρέπει τα παραπάνω αν όχι τότε η εφαρμογή τερματίζει.
3. Εν συνεχείᾳ ανοίγει η κάμερα και γίνεται αναζήτηση της τελευταίας τοποθεσίας.
4. Ο χρήστης έρχεται να εντοπίσει την κάθετη επιφάνεια που βρίσκεται το λογότυπο που θέλει σκανάρει ακολουθώντας την κίνηση ενός εικονικού χεριού

- που δίνεται από το ArCore. Ταυτόχρονα γίνεται αναζήτηση και ενημέρωση της τωρινής τοποθεσίας σε περίπτωση που άλλαξε.
5. Ο χρήστης πλησιάζει με την κάμερα το λογότυπο που τον ενδιαφέρει και πατάει πάνω στην οθόνη και πραγματοποιείται καταγραφή του συγκεκριμένου frame.
 6. Το συγκεκριμένο frame δίνεται για επεξεργασία και το επεξεργασμένο πλέον frame δίνεται στην μηχανή Tesseract. Αν δεν αναγνωριστεί κείμενο τότε γυρνάει στο 5.
 7. Μετά την ορθή αναγνώριση χαρακτήρων πραγματοποιείται μια αναζήτηση στην βάση δεδομένων με βάση το σαρωμένο κείμενο και την τελευταία καταγεγραμμένη τοποθεσία του χρήστη. Αν δεν γίνει επιτυχημένη αναζήτηση τότε γυρνάει στο 5.
 8. Με την επιτυχημένη αναζήτηση οι ανακτημένες πληροφορίες δίνονται στην εικονική πινακίδα.
 9. Τέλος η εικονική πινακίδα είναι πλέον ορατή στον χρήστη και μπορεί να πραγματοποιήσει τα ΣΧ03, ΣΧ04, ΣΧ05, ΣΧ06, ΣΧ07

2.4.3 Activity Diagram

Παρακάτω παρουσιάζεται και ένα activity diagram της εφαρμογής που περιλαμβάνει τα πάντα.



Εικόνα 2.11 Activity Diagram όλης της εφαρμογής

Visual Paradigm Online Free Edition

Κεφάλαιο 3. Σχεδίαση & Υλοποίηση

3.1 Σχεδίαση και αρχιτεκτονική λογισμικού

Στο συγκεκριμένο κεφάλαιο θα αναλύσουμε τον τρόπο υλοποίησης της εφαρμογής σε βάθος. Θα μιλήσουμε εκτενώς για τις πιο περίπλοκες και σημαντικές λειτουργίες της εφαρμογής με συχνή αναπαράσταση διαγραμμάτων και απαραίτητων αναφορών στον πηγαίο κώδικα.

3.1.1 Grandle

Στο grandle της εφαρμογής έχουμε κάνει όλα τα απαραίτητα import όπως φαίνεται παρακάτω. Συγκεκριμένα για την RoomDatabase, την OpenCV, το Sceneform, το Play-Services-location και η Tesseract4Android.

```
// Room components
implementation "androidx.room:room-runtime:$rootProject.roomVersion"
implementation 'androidx.wear:wear:1.1.0'
annotationProcessor "androidx.room:room-compiler:$rootProject.roomVersion"
androidTestImplementation "androidx.room:room-testing:$rootProject.roomVersion"

// Lifecycle components
implementation "androidx.lifecycle:lifecycle-viewmodel:$rootProject.lifecycleVersion"
implementation "androidx.lifecycle:lifecycle-livedata:$rootProject.lifecycleVersion"
implementation "androidx.lifecycle:lifecycle-common-java8:$rootProject.lifecycleVersion"

// opencv
implementation project(path: ':openCVLibrary3410')
implementation 'com.opencsv:opencsv:4.6'
// sceneform
implementation("com.gorisse.thomas.sceneform:sceneform:1.19.6")

// location play services
implementation 'com.google.android.gms:play-services-location:18.0.0'
implementation 'android.arch.lifecycle:extensions:1.1.1'

// tesseract
implementation 'cz.adaptech:tesseract4android:3.0.0'
```

Εικόνα 3.1 Στιγμιότυπο από το Grandle της εφαρμογής

3.1.2 Manifest

Στο manifest της εφαρμογής μπορούμε να δούμε τα permission που χρειάζεται η εφαρμογή μας για να λειτουργήσει. Συγκεκριμένα, στο manifest θα βρεθούν permissions για την τοποθεσία, την κάμερα, για κλήσεις τηλεφώνου, read-write external storage (που αφορά την Tesseract) και τέλος επειδή η εφαρμογή αποτελεί εφαρμογή επαυξημένης πραγματικότητας πρέπει να είναι συμβατή με το ARCore.

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"
    tools:ignore="ProtectedPermissions" />
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

Εικόνα 3.2 Στιγμιότυπο από το Manifest της εφαρμογής

3.1.3 Activities

Η εφαρμογή μας χρειάστηκε ένα μόνο Activity ονόματι MainActivity καθώς στοχεύουμε στην ελάχιστη συμμετοχή του χρήστη. Θέλουμε ο χρήστης να ακολουθεί την ροή των ενεργειών που αναφέρθηκε στο sequence diagram του Κεφαλαίου 2 χωρίς να διακοπεί από άλλα activities. Η ροή των λειτουργειών, έχει βάση στην κύρια κλάση μας MainActivity.

Το layout της αποτελείται από ένα εξωτερικό FrameLayout και ένα fragment, όπως μπορούμε να δούμε και στο παρακάτω στιγμιότυπο.

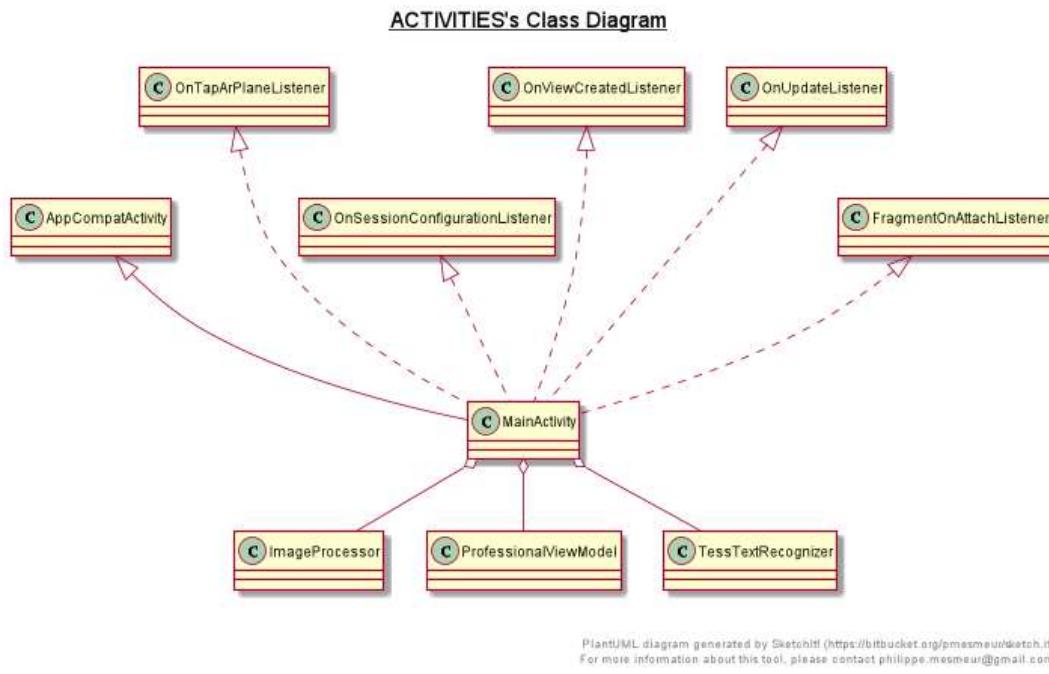
```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".activities.MainActivity">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/arFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</FrameLayout>
```

Εικόνα 3.4 Στιγμιότυπο από το Layout της MainActivity

Η εφαρμογή που υλοποιούμε αποτελεί μια εφαρμογή επαυξημένης πραγματικότητας που χρησιμοποιεί το ARCore και το Sceneform όπως έχουμε αναφέρει στο κεφάλαιο 2. Στο παρακάτω UML μπορεί να γίνει αντιληπτό αυτό από τις κλάσεις που κάνει implement η MainActivity.

Επίσης παρατηρούμε ότι έχει επικοινωνία με τρεις ακόμη κλάσεις ImageProcessor, ProfessionalViewModel, TessTextRecognizer που αφορούν τρεις βασικές λειτουργίες της εφαρμογής που θα αναλυθούν αργότερα (επεξεργασία εικόνας, πρόσβαση στην βάση δεδομένων και οπτική αναγνώριση χαρακτήρων).



Εικόνα 3.5 UML diagram MainActivity με όλες τις υπόλοιπες κλάσεις

Δίνεται η παρακάτω κάρτα CRC για μια περιληπτική ανάλυση των λειτουργιών της MainActivity.

Class Name: Professional	
Responsibilities	Collaborations
<ul style="list-style-type: none"> Υποχρέωση της κλάσης είναι να ελέγχει ότι ο χρήστης επιτρέπει την χρήση κάμερας και τοποθεσίας. Να φορτώνει τα μοντέλα που θέλουμε να αναπαραστήσουμε στην 3d σκηνή 	<ul style="list-style-type: none"> ImageProcessor ProfessionalViewModel TessTextRecognizer LocationMatcher

<p>μας και να κάνει όλες τις σχετικές αρχικοποιήσεις.</p> <ul style="list-style-type: none"> • Να βρίσκει την πιο πρόσφατη τοποθεσία του χρήστη. • Να εκτελέσει τις λειτουργίας της καταγραφής, επεξεργασίας στιγμιότυπου, και αναγνώρισης • Να κάνει αναζήτηση στην βάση με γνώμονα την τωρινή τοποθεσία και το αναγνωρισμένο κείμενο. • Να εμφανίζει την επιθυμητή εικονική πινακίδα με πληροφορίες που ο χρήστης μπορεί να παίρνει τηλέφωνο, στέλνει email, πλοηγείτε στην ιστοσελίδα • Να αλλάζει κείμενο με βάση την απόσταση του χρήστη από αυτήν. 	
---	--

Παρακάτω αναλύονται κάποια από τα responsibilities της MainActivity εκτενέστερα.

3.1.3.1 Τοποθεσία Χρήστη

Για την εύρεση της πιο πρόσφατης τοποθεσίας χρήστη χρησιμοποιήσαμε το fused location provider και την αποθηκεύουμε στο πεδίο current_location της MainActivity. Η τοποθεσία του χρήστη υπάρχει περίπτωση να μην εντοπιστεί κατευθείαν, για αυτό τον λόγο χρησιμοποιήσαμε το current_location για έλεγχο not null τοποθεσίας που θα μας οδηγούσε σε λάθη στις επόμενες ενέργειες που θα χρειαζόταν.

3.1.3.2 Πάτημα στην οθόνη

Για την συγκεκριμένη εφαρμογή χρησιμοποιούμε ένα Viewrendable το οποίο αναπαριστά την Εικονική Πινακίδα που αναφέραμε προηγουμένως.

Όταν ο χρήστης βρει κάποιο ενδιαφέρον λογότυπο εστιάζει πάνω του και πατάει πάνω στην οθόνη. Μόλις αυτό το πάτημα λάβει μέρος καλείται η μέθοδος onTapPlane η οποία μας παρέχεται από το Sceneform και συγκεκριμένα από τον BaseArFragment.OnTapArPlaneListener.

Εν συνεχεία λαμβάνουν χώρα οι εξής λειτουργίες, τοποθέτηση anchor για την μετέπειτα τοποθέτηση του viewrendable και μια σειρά από λειτουργίες για την ανάκτηση πληροφορίας.

Τοποθέτηση anchor

Με το πάτημα στην οθόνη τοποθετούμε έναν anchor ακριβώς στο σημείο που πάτησε ο χρήστης και δημιουργούμε έναν anchornode και ένα node για χρήση του στην συνέχεια για την τοποθέτηση της εικονικής πινακίδας.

Ροή λειτουργιών για την ανάκτηση πληροφορίας

Γίνεται καταγραφή στιγμιότυπου και επεξεργασία του μέσω της κλάσης ImageProcessor και με το επεξεργασμένο αποτέλεσμα γίνεται αναγνώριση χαρακτήρων μέσω της κλάσης TessTextRecognizer. Περισσότερα για τις συγκεκριμένες κλάσεις θα δούμε στις επόμενες υποενότητες.

Επειδή οι λειτουργίες της επεξεργασίας εικόνας και της αναγνώρισης χαρακτήρων μπορεί να είναι χρονοβόρες δεν είναι σωστό να τρέχουν στο κύριο νήμα της εφαρμογής και για αυτό τον λόγο έχουμε χρησιμοποιήσει έναν Executor και έναν Handler για να τρέχουν δηλαδή οι συγκεκριμένες λειτουργίες σε ένα άλλο νήμα.

Ουτρυτ οπτικής αναγνώρισης χαρακτήρων

Κάνοντας καταγραφή του στιγμιότυπου μπορεί να υπάρχουν διάφοροι χαρακτήρες και λέξεις που δεν μας ενδιαφέρουν για την αναζήτηση στην βάση. Όπως έχουμε προαναφέρει η αναζήτηση γίνεται με βάση το επώνυμο του επαγγελματία και την πιο πρόσφατη θέση του χρήστη. Για αυτό τον λόγο αρχικά κάνουμε κάποια σχετική επεξεργασία του αλφαριθμητικού που μας γυρνάει η μηχανή Tesseract, όπως αφαίρεση τόνων και μετατροπή σε όλα κεφαλαία έτσι ώστε να υπάρχει ακριβής αντιστοίχιση με τα πιθανά δεδομένα της βάσης. Τέλος εκτελούμε την λειτουργία αναζήτησης για κάθε πιθανή λέξη που αναγνώρισε το ουτρυτ ξεχωριστά.

Αναζήτηση στην βάση

Για την αναζήτηση της βάσης χρησιμοποιούμε ένα αντικείμενο της ProfessionalViewModel που όπως θα πούμε παρακάτω είναι ο συνδετικός κρίκος της MainActivity με την βάση μας. Οι δυο γνώμονες για την αναζήτηση είναι το επώνυμο του επαγγελματία και η τωρινή τοποθεσία χρήστη. Η τοποθεσία έχει αναγκαίο ρόλο εδώ επειδή βοηθάει στην αποφυγή λαθών λόγω διπλότυπων επαγγελματιών δηλαδή άτομα με ίδιο επώνυμο. Αρχικά γίνεται μια ανεύρεση όλων των διευθύνσεων που ανήκουν σε άτομα με ίδιο επώνυμο και εν συνεχεία πραγματοποιείται μια σύγκριση μεταξύ της πιο

πρόσφατης τοποθεσίας με κάθε διεύθυνση και η κοντινότερη σε αυτή σημαίνει ότι είναι και η σωστή. Τέλος γίνεται μια ανάκτηση όλων των επιθυμητών πληροφοριών π.χ. περίληψη, τηλέφωνο κ.λπ. και αποθηκεύονται σε ένα αλφαριθμητικό πεδίο της κλάσης MainActivity, καθώς και η αποθήκευση του ονοματεπωνύμου του επαγγελματία σε ένα άλλο αλφαριθμητικό. Τα δυο αυτά αλφαριθμητικά θα χρησιμοποιηθούν παρακάτω για την συμπλήρωση της εικονικής πινακίδας.

3.1.3.3 Εμφάνιση Εικονικής πινακίδας

Το ViewRenderable αποτελεί μια φλατ κάρτα που μπορεί να τοποθετηθεί στο περιβάλλον πάνω σε έναν anchornode ή σε ένα 3d model. Πάνω στο ViewRenderable μπορούν να τοποθετηθούν φωτογραφίες, κουμπιά, κείμενα κλπ. Για την δημιουργία του χρησιμοποιήθηκε ένα layout το οποίο περιέχει μόνο ένα TextView με background από ένα αποθηκευμένο drawable με άσπρο χρώμα και κυκλικές γωνίες.

Η εμφάνιση της εικονικής πινακίδας λαμβάνει μέρος στην onUpdate. Εφόσον έχει τοποθετηθεί ο anchorNode που αναφέραμε προηγουμένως και τα αλφαριθμητικά με τις πληροφορίες είναι γεμάτα τότε αρχίζει η διαδικασία απεικόνισης. Αρχικά υπολογίζουμε την ευκλείδεια απόσταση μεταξύ της κάμερας και του anchorNode και εάν είναι μικρότερη από 1 μέτρο τότε εμφανίζουμε την εικονική πινακίδα με όλες τις πληροφορίες όπως αποθηκεύτηκαν στο πρώτο αλφαριθμητικό. Δεύτερον, για απόσταση μεγαλύτερη του 1 μέτρου εμφανίζεται η μόνο το Ονοματεπώνυμο του επαγγελματία. Επιλέξαμε την αλλαγή του κειμένου έτσι ώστε να υπάρχει ορατή πληροφορία στον χρήστη από μακρινή απόσταση.

3.1.3.4 Design κειμένου εικονική πινακίδας

Για την δώσουμε την δυνατότητα στον χρήστη να μπορεί να πάρει τηλέφωνο, να στείλει email, να πλοηγηθεί στην ιστοσελίδα του επαγγελματία απλά προσθέσαμε info_text.setAutoLinkMask(Linkify.ALL); αυτή την γραμμή κώδικα που μας δίνει την δυνατότητα να κάνουμε link τα τηλέφωνα, emails, websites σε ένα TextView.

Τέλος για το design του TextView χρησιμοποιήσαμε την fromHtml μέθοδο για την πρόσθεση των bolds και τις αλλαγές γραμμών.

3.1.3.5 Επανάληψη διαδικασίας

Αν ο χρήστης θέλει να επαναλάβει την διαδικασία για κάποια άλλη ταμπέλα τότε απλά πατάει ξανά πάνω στην οθόνη και το viewrendable αντικαθίσταται από το καινούργιο. Τέλος, δεν βρεθεί κάποια πληροφορία στην βάση τότε εμφανίζεται ένα

μήνυμα στον χρήστη που λέει “Προσπαθήστε ξανά” διότι μπορεί να υπάρξει περίπτωση μικρών λαθών στην αναγνώριση.

3.1.4 Συλλογή Πληροφοριών

Όπως έχει προαναφερθεί, η παρούσα εφαρμογή βασίζεται στην ανάκτηση επιθυμητών πληροφοριών από μια τοπική βάση δεδομένων. Τα συγκεκριμένα δεδομένα συλλέχθηκαν από την γνωστή ιστοσελίδα vrisko.gr μέσω τεχνικών scrapping. Συγκεκριμένα χρησιμοποιήθηκε το ανοιχτού κώδικα framework scrappy.

Επιλέξαμε το vrisko.gr διότι είναι μια ιστοσελίδα που περιέχει πλήθος επαγγελματιών ιδιαίτερα στις περιοχές που πραγματοποιήθηκε ο έλεγχος της εφαρμογής. Επίσης η συγκεκριμένη σελίδα μας παρέχει πολλές χρήσιμες πληροφορίες για κάθε επαγγελματία, τις οποίες κατατάξαμε στην βάση μας. Ένα παράδειγμα ενός επαγγελματία που βρίσκεται στην ιστοσελίδα μπορούμε να δούμε στο παρακάτω στιγμιότυπο οθόνης.

ΑΛΑΤΖΙΔΟΥ ΖΗΝΟΒΙΑ

Χειρουργός Ωτορινολαρυγγολόγος

Σπόλωνος 34, Αθήνα - Κολωνάκι, 10673, ΑΤΤΙΚΗΣ

Ωτορινολαρυγγολόγοι

Παθολογία φωνής και κατάποσης - Διαταραχές φωνής και κατάποσης - Έλεγχος ακοής, ιλίγου, υπνικής άπνοιας, εμβοών και ροχαλητού - Ενδοσκόπηση ρινός, φάρυγγα, λάρυγγα, ώτων

Τηλέφωνο : 2109244040

Κινητό : 6944759665

Email: novialat@hotmail.com

Website: http://www.alatzidou.gr

Στοιχεία αναζήτησης: Ενδοσκοπηση Ρινος, Ελεγχος Ακοης, Ελεγχος Ιλιγου, Διαταραχες Φωνης Φωνησης, Διαταραχες Καταποσης

Κλείσε Ραντεβού

ΠΕΡΙΣΣΟΤΕΡΑ ►

Εικόνα 3.6 Παράδειγμα πίνακα πληροφοριών επαγγελματία από το vrisko.gr

Με βάση αυτό το πρότυπο κινηθήκαμε και αναπτύξαμε ένα script σε γλώσσα Python με την βοήθεια του Scrappy Framework στο περιβάλλον PyCharm. Για την εξαγωγή των συγκεκριμένων πληροφοριών χρειαστήκαμε μια κλάση spider την οποία ονομάσαμε ProfessionalDataSpider. Η βασική εργασία της συγκεκριμένης βάσης είναι να συλλέξει τα επιθυμητά δεδομένα χρησιμοποιώντας στοιχεία του πηγαίου κώδικα της σελίδας. Για αυτό τον λόγο αρχικά μελετήσαμε τον πηγαίο κώδικα του vrisko.gr έτσι ώστε να κατανοήσουμε πως συντάσσονται τα επιθυμητά πεδία, που θέλουμε να εξάγουμε, στην γλώσσα HTML της σελίδας. Στο παραπάνω στιγμιότυπο μπορούμε να δούμε στα κόκκινα πλαίσια ποια ακριβώς είναι τα πεδία που επιλέξαμε να χρησιμοποιήσουμε για την βάση μας. Τέλος, εξαγάγαμε τα συγκεκριμένα δεδομένα σε ένα αρχείο μορφής .csv για χρήση του στην δημιουργία της βάσης αργότερα.

Παρακάτω μπορούμε να δούμε το script που χρησιμοποιήσαμε για την συλλογή δεδομένων. Όπως παρατηρείτε συλλέξαμε στην βάση μας στοιχεία από γιατρούς της Αθήνας χωρίς αυτό να σημαίνει ότι δεν μπορούν να υπάρχουν άλλοι επαγγελματίες στην βάση. Η επιλογή των επαγγελματιών έγινε καθαρά με βάση το πλήθος των πληροφοριών που παρείχαν.

```
class ProfessionalsDataSpider(scrapy.Spider):
    page_number = 1
    name = 'professionals'

    start_urls = [
        'https://www.vrisko.gr/search/giatroi/athens/'
    ]

    def parse(self, response):
        blank_line = 0
        global page_number

        for prof in response.xpath("//div[@class = 'LightAdvAreaLeft']/div"):
            if blank_line % 2 != 0:
                blank_line = blank_line + 1
                continue
            yield {
                'OfficeName': prof.xpath("div/div/h2/a/meta/@content").extract_first(),
                'Description': prof.xpath("div/div/h2/meta/@content").extract_first(),
                'Category': prof.xpath("div[2]/div/div/div[1]/Label/span/text()").extract_first(),
                'Address': prof.xpath("div[2]/div/div/div[2]/text()").extract_first(),
                'Office_Phone': prof.xpath("div[2]/div[2]/div[3]/div[2]/text()").extract_first(),
                'Mobile_Phone': prof.xpath("div[2]/div[2]/div[3]/div[5]/text()").extract_first(),
                'Email': prof.xpath("div[2]/div[2]/div[3]/div[8]/a[contains(@href, 'mailto')]/@href").extract_first(),
                'Website': prof.xpath("div[2]/div[2]/div[3]/div[11]/a/text()").extract_first()
            }
            blank_line = blank_line + 1
```

Εικόνα 3.7 Script μέσω Scrapy Framework για συλλογή των δεδομένων.

Αξιόλογο θα ήταν να αναφέρουμε ότι κατά την διάρκεια πειραμάτων στις αρχικές υλοποιήσεις της εφαρμογής παρατηρήθηκε ότι το OfficeName δεν είναι ένα βολικό πεδίο για αναζήτηση διότι στην πραγματικότητα το ονοματεπώνυμο μπορεί να αναγράφεται αντίστροφα από ότι στην βάση δεδομένων, π.χ “Αλατζίδου Ζηνοβία” και “Ζηνοβία Αλατζίδου”. Για αυτό ανατρέξαμε ξανά στο vrisko.gr και ήρθαμε στο συμπέρασμα ότι το label που περιέχει το ονοματεπώνυμο κάθε επαγγελματία ακολουθεί ένα συγκεκριμένο μοτίβο δηλαδή πρώτα το επώνυμο και μετά όνομα. Για αυτό το λόγο το σπάσαμε σε δύο διαφορετικά labels όνομα και επώνυμο επειδή μας ήταν απαραίτητο το επώνυμο μεμονωμένα για την ανάκτηση δεδομένων που θα περιγράψουμε αργότερα. Τα πεδία που χρησιμοποιήσαμε λοιπόν έχουν χωριστεί όπως φαίνεται παρακάτω.

ΑΛΑΤΖΙΔΟΥ ΖΗΝΟΒΙΑ		surname - profName	
Zhnobia Alatzidou	Χειρουργός Οπορινολαρυγγολόγος	address	ΤΗΛΕΦΩΝΟ
	Σόλωνος 34, Αθήνα - Κολωνάκι, 10673, ΑΤΤΙΚΗΣ	category	ΧΑΡΤΗΣ
Oπορινολαρυγγολόγοι			WEBSITE
			EMAIL
	Παθολογία φωνής και κατάποσης - Διαταραχές φωνής και κατάποσης - Έλεγχος ακοής, ιλίγγου, υπνικής άπνοιας, εμβούων και ροχαλητού - Ενδοσκόπηση ρινός, φάρυγγα, λάρυγγα, ώτων	description	
	Τηλέφωνο : 2109244040	office phone	
	Κινητό : 6944759665	mobile phone	
	Email: novialat@hotmail.com	email	
	Website: http://www.alatzidou.gr	website	
	Στοιχεία αναζήτησης: Ενδοσκοπηση Ρινος, Ελεγχος Ακοης, Ελεγχος Ιλιγγου, Διαταραχες Φωνης Φωνησης, Διαταραχες Καταποσης		
 Κλείσε Ραντεβού			ΠΕΡΙΣΣΟΤΕΡΑ ►

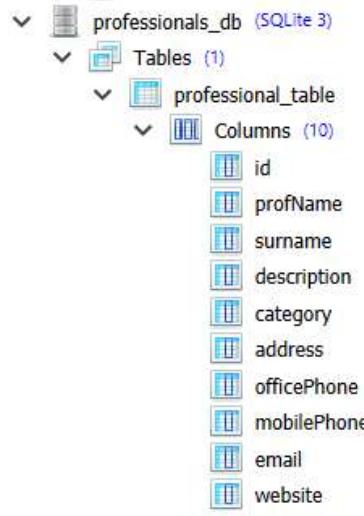
Εικόνα 3.8 Αντιστοίχιση ονομάτων με labels.

3.1.5 Βάση Δεδομένων

3.1.5.1 SQLiteStudio

Για την δημιουργία της βάσης δεδομένων μας χρησιμοποιήσαμε το SQLiteStudio. Η υλοποίηση της βάσης έγινε σε SQLite 3 διότι στην android εφαρμογή μας χρησιμοποιούμε για την διαχείριση και την ενημέρωση της βάσης την βιβλιοθήκη Room, που βασίζεται σε SQLlite.

Πιο συγκεκριμένα, δημιουργήσαμε μια βάση ονόματι professionals η οποία έχει ένα table professional_table με 10 στήλες όπως μπορούμε να δούμε στο επόμενο στιγμιότυπο από το δέντρο της βάσης στο SQLiteStudio. Όπως είναι προφανές οι 9 από τις 10 στήλες αφορούν τα πεδία που έχει το .csv αρχείο που εξήγαμε από το vrisko.gr.



Εικόνα 3.9 Βάση δεδομένων από SQLiteStudio.

Αναλυτικότερα στο ακολουθούμενο πίνακα μπορούμε να δούμε περισσότερες πληροφορίες για στήλες του **professionals_table**

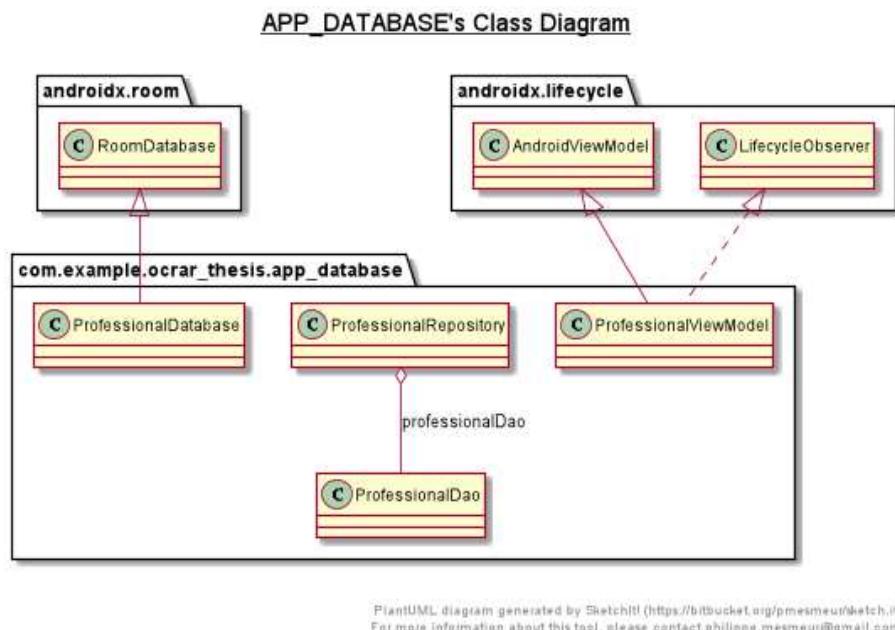
Όνομα	Τύπος δεδομένων	Περιγραφή
id	INTEGER	Το μοναδικό αναγνωριστικό κάθε επαγγελματία - Πρωτεύον κλειδί - NOT NULL
profName	TEXT	Το όνομα του επαγγελματία - NOT NULL
surname	TEXT	Το επώνυμο του επαγγελματία. NOT NULL
description	TEXT	Μια περιγραφή σχετικά με τον επαγγελματία.
category	TEXT	Η κατηγορία που ανήκει ο επαγγελματίας - τι επαγγέλλεται
address	TEXT	Η διεύθυνση του γραφείου - NOT NULL
officePhone	TEXT	Το σταθερό τηλέφωνο του γραφείου
mobilePhone	TEXT	Το κινητό τηλέφωνο του επαγγελματία
email	TEXT	Η διεύθυνση ηλεκτρονικού ταχυδρομείου του επαγγελματία.
website	TEXT	Η ιστοσελίδα του επαγγελματία.

Τέλος, εισαγάγαμε στο professionals_table τα δεδομένα από το .csv αρχείο που δημιουργήσαμε όπως εξηγήσαμε προηγουμένως.

3.1.5.2 RoomDatabase

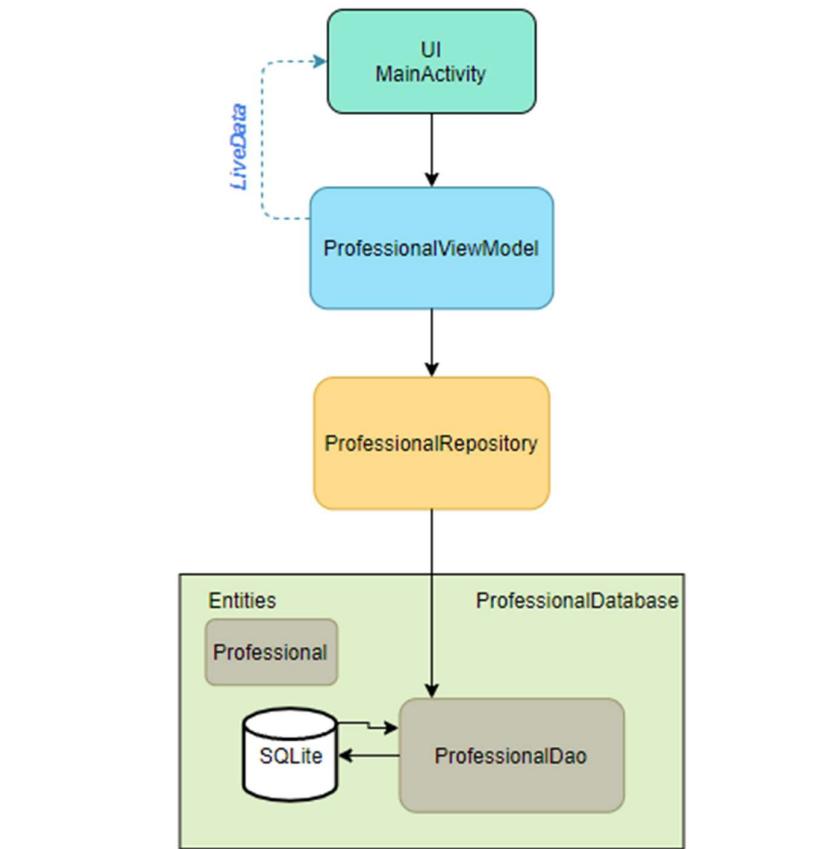
Μετά την συλλογή των επιθυμητών δεδομένων και την δημιουργία της βάσης professionals χρησιμοποιήσαμε την βιβλιοθήκη Room για εύκολη διαχείριση και ενημέρωση της βάσης. Η επιλογή της συγκεκριμένης βιβλιοθήκης αναλύεται στο κεφάλαιο 2. Αναλυτικότερα, δημιουργήσαμε την τοπική βάση δεδομένων της εφαρμογής μας και την κάναμε repopulate με την βάση που ήδη είχαμε φτιάξει στο SQLiteStudio. Ο συγκεκριμένος τρόπος προ πληθυσμού της Room Database προτάθηκε από την ιστοσελίδα [Prepopulate your Room database | Android Developers](#).

Πιο συγκεκριμένα, οι κλάσεις που αφορούν την Room Database της εφαρμογής είναι οι πέντε κλάσεις που φαίνονται στο παρακάτω UML diagram.



Εικόνα 3.10 Uml κλάσεων Βάσης Δεδομένων.

Παρακάτω αναπαριστούμε σε ένα λιτό διάγραμμα την λειτουργία της δημιουργίας και διαχείρισης της βάσης δεδομένων της εφαρμογής. Εν συνεχεία, θα εξηγήσουμε τις συγκεκριμένες κλάσεις μέσω CRC καρτών που θα μας προσφέρουν αναλυτικότερες πληροφορίες σχετικά με το παρακάτω διάγραμμα.



Εικόνα 3.11 Επεξηγηματικό διάγραμμα λειτουργιών Room Database.

Class Name: Professional	
Responsibilities	Collaborations
<ul style="list-style-type: none"> Η συγκεκριμένη κλάση αποτελεί ένα entity της Room βάσης μας. Υποχρέωση της είναι η δημιουργία ενός table ονόματι professional_table με στήλες τις στήλες που είχε το ανάλογο table που δημιουργήσαμε στο SQLiteStudio. 	<ul style="list-style-type: none"> ProfessionalDao ProfessionalDatabase ProfessionalRepository ProfessionalViewModel

Class Name: ProfessionalDao	
Responsibilities	Collaborations
<ul style="list-style-type: none"> • Η συγκεκριμένη κλάση αποτελεί το dao της Room βάσης μας. • Η συγκεκριμένη κλάση έχει ως υποχρέωση την δημιουργία των queries που θα χρησιμοποιηθούν αργότερα. 	<ul style="list-style-type: none"> • ProfessionalDatabase • ProfessionalRepository

Class Name: ProfessionalDatabase	
Responsibilities	Collaborations
<ul style="list-style-type: none"> • Η συγκεκριμένη κλάση αποτελεί μια abstract κλάση της RoomDatabase και είναι η βασική κλάση για την δημιουργία της βάσης όπως παρατηρείται και από το παραπάνω διάγραμμα. • Υποχρέωση της είναι να δημιουργήσει και να αρχικοποιήσει την βάση της εφαρμογής με την professional.db βάση που φτιάξαμε μέσω το SQLiteStudio. Η professional.db είναι αποθηκευμένη στα assets της εφαρμογής. 	<ul style="list-style-type: none"> • ProfessionalRepository

Class Name: ProfessionalRepository	
Responsibilities	Collaborations
<ul style="list-style-type: none"> Η συγκεκριμένη κλάση χρησιμοποιείται ως ένας buffer που συνδέεται άμεσα με τις βασικές κλάσεις της βάσης δηλ. Entities, dao, roomdatabase. 	<ul style="list-style-type: none"> ProfessionalViewModel

Class Name: ProfessionalViewModel	
Responsibilities	Collaborations
<ul style="list-style-type: none"> Η συγκεκριμένη κλάση αποτελεί τον συνδετικό κρίκο μεταξύ του UserInterface και της βάσης δεδομένων. Κύρια σύνδεση της είναι το ProfessionalRepository από το οποίο λαμβάνει τα LiveData χωρίς να συνδέεται απευθείας με το dao της room database. Υποχρέωσή της να μεταφέρει τα LiveData από την βάση στο MainActivity μέσω observers. 	<ul style="list-style-type: none"> MainActivity

Με την μεταγλώττιση της εφαρμογής μπορούμε να δούμε live την βάση δεδομένων της εφαρμογής από το Database Inspector και να τρέξουμε queries σε πραγματικό χρόνο.

3.1.6 Καταγραφή στιγμιότυπου & Επεξεργασία του

Για τις δυο αυτές λειτουργίες έχουμε υλοποιήσει μια κλάση ονόματι ImageProcessor.

3.1.6.1 Καταγραφή

Όπως αναφέραμε παραπάνω με την κλήση της **onTapPlane** γίνεται καταγραφή του συγκεκριμένου frame και επεξεργασία του. Πιο συγκεκριμένα, δημιουργείται ένα αντικείμενο ImageProcessor που δίνει στην αρχικοποίηση του δίνει στον constructor το **arFragment** το οποίο έχει αρχικοποιηθεί στην **onAttachFragment** και μέσω αυτού μπορούμε να κάνουμε το capture του frame. Για την ακρίβεια καλείται **arFragment.getArSceneView().getArFrame().acquireCameraImage()** μέσω μιας μεθόδου που γίνεται κλήση της στον constructor. Η παραπάνω γραμμή κώδικα μας επιστρέφει το frame σε τύπο Image και format **YUV420**. Για αυτό τον λόγο πρέπει να την μετατρέψουμε σε Bitmap και επίσης να την κάνουμε περιστροφή 90 μοιρών έτσι ώστε να μπορέσουμε να την επεξεργαστούμε στο επόμενο βήμα.

YUV420 σε Bitmap

Το YUV420 είναι μια από τις πιο γνωστές μορφές εικόνας σε Android. Συγκεκριμένα αποτελείται από 3 διαφορετικά επίπεδα (YCbCr) και η σειρά των planes είναι [0]: Y plane (Luma) [1]: U plan (Cb) [2]: V plane (Cr) όπως μπορεί να διαπιστωθεί και από το παρακάτω πινακάκι που πήραμε από το [How to use YUV \(YUV 420 888\) Image in Android | Minhaz's Blog \(minhazav.dev\)](#).

Single Frame YUV420:



Εικόνα 3.12 Δομή YUV420 εικόνας.

Επειδή το BitmapFactory δεν μπορεί να διαβάσει τέτοιου είδους δεδομένα χρειάστηκε να αλλάξουμε το YUV Image σε Bitmap όπως είπαμε παραπάνω. Για την συγκεκριμένη μετατροπή συμβουλευτήκαμε την συγκεκριμένη πηγή [java - How to get Bitmap from session.update\(\) in ARCore \[Android Studio\] - Stack Overflow](#)

Αρχικά μετατρέψαμε την YUV σε JPG και μόλις πήραμε το bytearray από την συγκεκριμένη μετατροπή δημιουργήσαμε το επιθυμητό Bitmap

Περιστροφή εικόνας

Τέλος, παρατηρήθηκε ότι το καταγεγραμμένο στιγμιότυπο ήταν πάντα γυρισμένο 90 μοίρες και για αυτό τον λόγο πριν δώσουμε το Bitmap στην επόμενη λειτουργία το περιστρέφουμε. Η εφαρμογή έτρεξε σε διάφορα κινητά τηλέφωνα έτσι ώστε να είναι σίγουρο ότι η συγκεκριμένη περιστροφή πραγματοποιείται πάντα και παντού.

3.1.6.2 Επεξεργασία εικόνας

Παρά την χρήση του **Tesseract4Android** η εικόνα μας συνέχισε να χρήζει επεξεργασίας λόγω παραγόντων που προκαλούν θόρυβο στην εικόνα όπως για παράδειγμα η φωτεινότητα στον χώρο. Για αυτό τον λόγο αποφασίσαμε να χρησιμοποιήσουμε την γνωστή βιβλιοθήκη **OpenCV**. Για την σωστή επιλογή μεθόδων συμβουλευτήκαμε την ιστοσελίδα [Improving the quality of the output | tessdoc \(tesseract-ocr.github.io\)](#) καθώς και την πηγή [How to use image preprocessing to improve the accuracy of Tesseract \(freecodecamp.org\)](#) που μας έδωσε κάποιες παραπάνω ιδέες.

Πλέον για την επεξεργασία της εικόνας μας ακολουθούμε τα παρακάτω βήματα

1. Μετατροπή της εικόνας σε **κλίμακα του γκρι**.
 - Η μετατροπή της εικόνας σε κλίμακα του γκρι είναι ιδιαίτερα σημαντικοί καθώς διευκολύνει τον αλγόριθμο εξαγωγής πληροφορίας από την εικόνα και μειώνει τις σχετικές υπολογιστικές απαιτήσεις.
2. **Blurring** ή αλλιώς θόλωση της εικόνας
 - Το θόλωμα της εικόνας είναι ένα πολύ σημαντικό βήμα για την εξομάλυνση της εικόνας και αποφυγής θορύβου και λεπτομερειών της εικόνας που δεν μας ενδιαφέρουν.
 - Χρησιμοποιήσαμε θόλωμα με την μέθοδο **Imgproc.GaussianBlur** που μας προτάθηκε από την παραπάνω πηγή ειδικά για θορύβους Gauss
 - Εν συνεχείᾳ χρησιμοποιήθηκε και η μέθοδος **Imgproc.medianBlur** που μας βοηθάει στην αποφύγει salt-and-pepper θόρυβους.
3. **Threshold - χρήση κατωφλίου**
 - Με την χρήση κατωφλίου μετατρέπουμε την εικόνα σε δυαδική εικόνα δηλαδή ασπρόμαυρη, μας βοηθάει ώστε να επιλέξουμε περιοχές στην εικόνα που μας ενδιαφέρουν (στην περίπτωση μας γραμματοσειρές).
 - Χρησιμοποιήθηκε η μέθοδος **Imgproc.threshold** και συγκεκριμένα η μέθοδος κατωφλίου **Otsu** όπως προτάθηκε από την πηγή.
4. **Dilation - διαστολή**
 - Με την χρήση του dilation επειδή δίνει έμφαση στα γράμματα και τα κάνει πιο καθαρά για την tesseract.
 - Χρησιμοποιήθηκε η **Imgproc.dilate** και με το αποτέλεσμα αυτής είμαστε πλέον έτοιμοι να δώσουμε την επεξεργασμένη εικόνα στην Tesseract για αναγνώριση.

3.1.7 Αναγνώριση χαρακτήρων

Σε αυτή την ενότητα θα αναλύσουμε τον τρόπο που υλοποιήσαμε την λειτουργία οπτικής αναγνώρισης χαρακτήρων. Όπως αναφέραμε στο κεφάλαιο 2 χρησιμοποιήσαμε την μηχανή Tesseract OCR και συγκεκριμένα το fork της Tess-two [adaptech-cz/Tesseract4Android: Fork of tess-two rewritten from scratch to support latest version of Tesseract OCR. \(github.com\)](https://github.com/adaptech-cz/Tesseract4Android: Fork of tess-two rewritten from scratch to support latest version of Tesseract OCR. (github.com)) που είναι ειδικά σχεδιασμένο για ανάπτυξη εφαρμογών σε Android Studio και προσφέρει τεχνικές επεξεργασίας εικόνας προτού δοθεί η εικόνα στην Tesseract. Για την χρήση του συγκεκριμένου project έγινε import μέσω του gradle της εφαρμογής μας.

Αναλυτικότερα, για την υλοποίηση της συγκεκριμένης λειτουργίας χρειαστήκαμε ένα αρχείο που περιέχει προ εκπαιδευμένα δεδομένα στην Ελληνική γλώσσα ονόματι “ell.traineddata” που βρίσκεται στο φάκελο assets. Χρησιμοποιήσαμε τα συγκεκριμένα δεδομένα επειδή στην εφαρμογή χρειάζεται να κάνουμε αναγνώριση ελληνικών γραμματοσειρών.

Η βασική λειτουργία της βρίσκεται στην κλάση **TessTextRecognizer** περισσότερες πληροφορίες για την οποία μπορούμε να δούμε στην παρακάτω CRC κάρτα.

Class Name: TessTextRecognizer	
Responsibilities	Collaborations
<ul style="list-style-type: none">• Η συγκεκριμένη κλάση έχει δυο υποχρεώσεις:<ul style="list-style-type: none">◦ Η εγκατάσταση της tesseract στην εφαρμογή το οποίο γίνεται με την δημιουργία ενός αντικειμένου της κλάσης. Πρέπει να δοθεί ιδιαίτερη προσοχή στα paths των trainned data και του tessdata για να γίνει σωστά η εγκατάσταση.◦ Η δεύτερη εργασία της κλάσης είναι να γυρίσει το αποτέλεσμα της αναγνώρισης του αλγορίθμου σε μορφή αλφαριθμητικού.	<ul style="list-style-type: none">• MainActivity

Θα ήταν αξιόλογο να αναφερθεί ότι για την καλύτερη και ακριβέστερη αναγνώριση χαρακτήρων έχουμε βάλει σε προσδιορίσει με ακρίβεια την WhiteList (χαρακτήρες που μας ενδιαφέρουν) και την BlackList (χαρακτήρες που δεν μας ενδιαφέρουν) των χαρακτήρων που αναγνωρίζονται. Όπως μπορούμε να δούμε και παρακάτω.

```
mTess.setVariable(TessBaseAPI.VAR_CHAR_WHITELIST, value: "ΑΒΓΔΕΖΗΙΚΑΜΝΕΩΠΡΣΤΥΦΨωθεζηθικλμνξορπαυφχψως" +  
    "ιάσσεώινήλαστηγί");  
mTess.setVariable(TessBaseAPI.VAR_CHAR_BLACKLIST, value: "!@#$%^&()_-+=[]{};:\\\"\\~,.//><?0123456789");
```

Εικόνα 3.13 WhiteList και BlackList Tesseract αλγόριθμου.

3.1.8 Αντιστοίχιση διευθύνσεων

Όπως έχουμε αναφέρει ξανά για την αντιστοίχιση της σωστής διεύθυνσης υπολογίζουμε αν η πιο πρόσφατη διεύθυνση του χρήστη είναι το πολύ 10 μέτρα μακριά από την αποθηκευμένη στην βάση. Αν είναι τότε γίνεται αντιστοιχία των διευθύνσεων. Χρησιμοποιήσαμε απόσταση το πολύ 10 μέτρα διότι το gps δεν έχει την τέλεια ακρίβεια σε μέτρα.

Την λειτουργία της αντιστοίχισης την υλοποιήσαμε στην κλάση **LocationMatcher**. Ακολουθεί η CRC κάρτα της.

Class Name: LocationMatcher	
Responsibilities	Collaborations
<ul style="list-style-type: none"> Η συγκειριμένη κλάση έχει δύο υποχρεώσεις <ul style="list-style-type: none"> Να βρει το γεωγραφικό πλάτος και μήκος από τις αποθηκευμένες διευθύνσεις στην βάση. Να βρει την απόσταση μεταξύ της διεύθυνσης βάσης και της πιο πρόσφατης διεύθυνσης του χρίστη. 	<ul style="list-style-type: none"> MainActivity

Εύρεση γεωγραφικού πλάτους και μήκους

Πιο συγκεκριμένα, για την **εύρεση του γεωγραφικού πλάτους και μήκους** από μια διεύθυνση αποθηκευμένη σε ένα String αυτό που κάναμε ήταν το εξής. Δημιουργήσαμε ένα αντικείμενο τύπου **Geocoder** με **Locale("el_GR")** επειδή οι διευθύνσεις είναι αποθηκευμένες στα ελληνικά. Τελικά με την βοήθεια της μεθόδου **getFromLocationName** μπορέσαμε να βρούμε το string της διεύθυνσης σε αντικείμενο της κλάσης **Address** και έτσι να πάρουμε το επιθυμητό γεωγραφικό πλάτος και μήκος.

Εύρεση απόστασης μεταξύ διευθύνσεων

Για την εύρεση της απόστασης μεταξύ της διεύθυνσης βάσης και της πιο πρόσφατης διεύθυνσης χρησιμοποιούμε την μέθοδο **Location.distanceBetween** η οποία κάνει σύγκριση των γεωγραφικών πλατών και μηκών των παραπάνω τοποθεσιών και γυρνάει την μεταξύ τους απόσταση.

3.2 Επεκτασιμότητα του λογισμικού

Όπως έχουμε προαναφέρει η παρούσα εφαρμογή μπορεί να επεκταθεί εύκολα χωρίς πολύ κόπο. Πιο συγκεκριμένα, όπως γνωρίζουμε στην περίπτωση μας έχουμε συμπεριλάβει δεδομένα επαγγελματιών από περιοχές της Αθήνας. Η εφαρμογή δεν είναι απαραίτητα υλοποιημένη για αναζήτηση επαγγελματιών, αλλά μπορεί με κάποιες αλλαγές να πραγματοποιεί ανάκτηση δεδομένων που δεν αφορούν επαγγελματίες. Για να γίνει όμως αυτό χρειάζεται αλλαγή της βάσεως δεδομένων και δημιουργία των αντίστοιχων Entities, Dao στην υλοποίηση της Room καθώς και αλλαγή στον τρόπο αναζήτησης στην βάση. Πλέον, η αναζήτηση στην βάση πραγματοποιείται με το επώνυμο του επαγγελματία και την πιο πρόσφατη τοποθεσία του χρήστη. Με την αλλαγή των δεδομένων θα πρέπει να οριστεί κάποιο άλλο keyword έναντι του επώνυμου του επαγγελματία και να παραμείνει η πιο πρόσφατη τοποθεσία του χρήστη καθώς αποτελεί έναν εύλογο τρόπο ελέγχου διπλότυπων δεδομένων.

Φυσικά μπορεί να υπάρξουν περιπτώσεις που η τοποθεσία δεν είναι χρήσιμη. Για παράδειγμα εάν η συγκεκριμένη εφαρμογή είχε μια άλλη χρήση όπως αναγνώριση χαρακτήρων από ταμπέλες εκθεμάτων σε ένα μουσείο και εμφάνισης πληροφοριών ή ακόμα και 3d αντικειμένων που σχετίζονται με το έκθεμα τότε η τοποθεσία χρήστη δεν θα μας ήταν χρήσιμη. Στην ιδέα που αναφέρθηκε θα έπρεπε να γίνει άλλη μια πιθανή επέκταση όσον αφορά την εμφάνιση 3d αντικειμένων που θα αντικαθιστούσαν την τωρινή εικονική πινακίδα. Κάτι που μπορεί να υλοποιηθεί πολύ εύκολα με την βοήθεια της Sceneform.

Η εφαρμογή επίσης θα μπορούσε να επεκταθεί έτσι ώστε να παρέχει πληροφορίες από ένα πλήθος διαφορετικών δεδομένων. Για αυτή την υλοποίηση θα έπρεπε να δημιουργηθούν πολλαπλά Entities και ο τρόπος ανάκτησης πληροφορίας να γίνεται με διαφορετικό τρόπο για κάθε είδος δεδομένου. Τέλος θα έπρεπε με βάση κάποια λέξη κλειδί από το αναγνωρισμένο κείμενο να κατατάσσουμε την συγκεκριμένη ταμπέλα στο σωστό Entity.

Όπως παρατηρούμε λοιπόν η εφαρμογή μας έχει έναν επεκτάσιμο χαρακτήρα και στοχεύει στην κάλυψη εύρεσης πληροφορίας διαφορετικών τύπων δεδομένων.

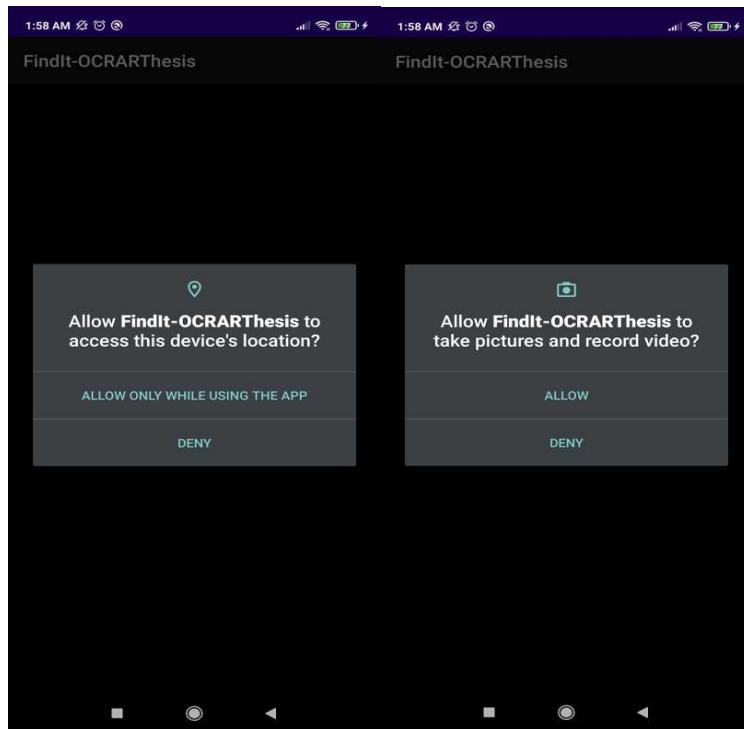
Κεφάλαιο 4. Πειραματική Αξιολόγηση

Στην ενότητα αυτή παρουσιάζουμε αναλυτικά την πειραματική αξιολόγηση της εφαρμογής μας. Ο έλεγχος της εφαρμογής έγινε με ένα πείραμα «προσημείωσης κτιρίου επαγγελματιών» καθώς και κάποιων άλλων πειραμάτων που αφορούν συγκεκριμένες λειτουργίες. Με την βοήθεια του debugger του Android Studio μπορέσαμε να ελέγξουμε όλες τις λειτουργίες που λαμβάνουν μέρος στο παρασκήνιο της εφαρμογής.

4.1 Ανάλυση πειράματος

4.1.1 Άδειες εφαρμογής

Όπως έχουμε αναφέρει στα προηγούμενα κεφάλαια με το άνοιγμα της εφαρμογής ζητείται από τον χρήστη η άδεια για χρήση της κάμερας και χρήση της τοποθεσίας. Σε περίπτωση που ο χρήστης δεν αποδεχτεί τις συγκεκριμένες άδειες η εφαρμογή τερματίζει. Παρακάτω μπορούμε να δούμε δυο στιγμιότυπα από την έναρξη της εφαρμογής με τα δυο μηνύματα που εμφανίζονται στον χρήστη.



Εικόνα 4.1 Άδεια για χρήση κάμερας Εικόνα 4.2 Άδεια για χρήση τοποθεσίας

4.1.2 Εντοπισμός κάθετης επιφάνειας

Με την αποδοχή των παραπάνω αδειών ανοίγει η εφαρμογή και εμφανίζεται η κάμερα του κινητού. Ο χρήστης βλέπει ένα εικονικό «χέρι» που του δείχνει την κίνηση που πρέπει να εκτελέσει για να εντοπιστεί η επιφάνεια. Όπως μπορούμε να δούμε παρακάτω.



Εικόνα 4.3 Εικονικό «χέρι»

Στην περίπτωση της δικής μας εφαρμογής χρειάζεται να εντοπίζουμε μόνο κάθετες επιφάνειες διότι στοχεύουμε στον εντοπισμό πινακίδων και ταμπελών που έχουν κάθετη μορφή. Για αυτό τον λόγο στην στη συνάρτηση `onSessionConfiguration` έχουμε ορίσει ότι η συγκεκριμένη εφαρμογή απευθύνεται μόνο σε κάθετες επιφάνειες όπως τοίχους.

```
@Override  
public void onSessionConfiguration(Session session, Config config) {  
    if (session.isDepthModeSupported(Config.DepthMode.AUTOMATIC)) {  
        config.setDepthMode(Config.DepthMode.AUTOMATIC);  
    }  
    config.setPlaneFindingMode(Config.PlaneFindingMode.VERTICAL);  
}
```

Εικόνα 4.4 Καθορισμός εντοπισμού μόνο κάθετων επιφανειών

Για τον επιτυχή εντοπισμό της επιφάνειας πρέπει να επιλέξουμε κάποια επιφάνεια η οποία δεν είναι μονόχρωμη και έχει αντικείμενα πάνω σε αυτήν. Για παράδειγμα ένας μονόχρωμος λευκός τοίχος είναι πιο δύσκολο να γίνει γρήγορα αντιληπτός από το ArCore σε σχέση με έναν τοίχο που έχει διάφορες κορνίζες πάνω του κορνίζες.

Με την εύρεση της επιφάνειας εμφανίζεται ένα πλέγμα στην οθόνη του χρήστη που καθορίζει την περιοχή που καλύπτει. Σε αυτήν την περιοχή θα τοποθετηθεί η εικονική πινακίδα μας. Όπως μπορεί να παρατηρηθεί με την κίνηση της κάμερας μπορεί να κινηθεί και το πλέγμα έτσι ώστε να εκτελέσει ενέργειες σε ένα μεγαλύτερο εύρος της επιφάνειας. Παρακάτω μπορούμε να δούμε το πλέγμα και τον τοίχο πάνω στον τοίχο που χρησιμοποιήθηκε για το πείραμα μας.



Εικόνα 4.5 Προσομοίωση κτιρίου επαγγελματιών

4.1.3 Εντοπισμός πιο πρόσφατης τοποθεσίας χρήστη

Όπως αναφέρθηκε στο κεφάλαιο 3 η εύρεση της πιο πρόσφατης τοποθεσίας χρήστη γίνεται μέσω του **fused location provider** που μας παρέχεται από το Google Play Services. Κάτι ευρέως γνωστό που αφορά τις Android εφαρμογές είναι ότι η τοποθεσία του χρήστη μπορεί να μην βρεθεί απευθείας και να η εφαρμογή να προχωρήσει σε

επόμενες λειτουργίες της. Για αυτό τον λόγο ελέγχεται πάντα η τοποθεσία αν είναι not null έτσι ώστε να ακολουθήσουν οι επόμενες διαδικασίες της εφαρμογής μας όπως αναγνώριση χαρακτήρων, ανάκτηση πληροφοριών κλπ.

4.1.4 Επεξεργασία εικόνας

Σύμφωνα με την εφαρμογή και όπως μπορούμε να δούμε στο Activity Diagram του κεφαλαίου 2 ο χρήστης με το που πατάει πάνω στην οθόνη πραγματοποιείται καταγραφή του στιγμιότυπου και ξεκινά μια σειρά από περίπλοκες διαδικασίες στο παρασκήνιο.

Πιο συγκεκριμένα, μια πολύ σημαντική διαδικασία είναι η διαδικασία της επεξεργασίας εικόνας. Παρακάτω χρησιμοποιήσει μια άλλη προσομοίωση ταμπέλας για ελέγχουμε σε αναλυτικότερα τις λειτουργίες της επεξεργασίας εικόνας και της αναγνώρισης χαρακτήρων μέσω του debugger του Android Studio.



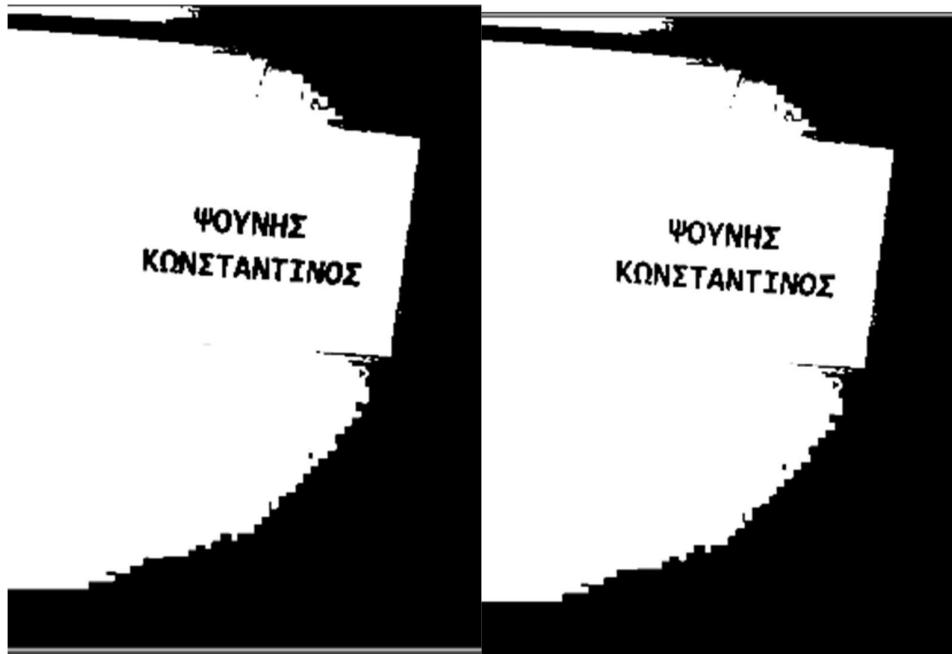
Εικόνα 4.6 Πριν την επεξεργασία

Με την καταγραφή της εικόνας πραγματοποιείται όπως είπαμε μια μετατροπή της εικόνας σε κλίμακα του γκρι και φιλτράρισμα με Gaussian kernel και median φιλτράρισμα. Παρακάτω μπορούμε να δούμε το στιγμιότυπο μετά από αυτή την επεξεργασία.



Εικόνα 4.7 Μετατροπή σε κλίμακα του γκρι και φιλτράρισμα

Εν συνεχείᾳ πραγματοποιείται thresholding Otsu και dilation. Πλέον το στιγμιότυπο έχει την κατάλληλη μορφή για να δοθεί ως είσοδος στην μηχανή Tesseract.

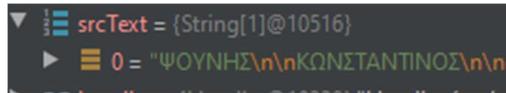


Εικόνα 4.8 Μετά το Otsu's thresholding

Εικόνα 4.9 Μετά το dilation

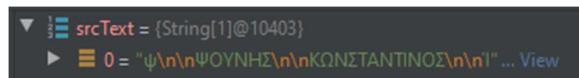
4.1.5 Αναγνώριση Χαρακτήρων

Συνεχίζουμε με το προηγούμενο παράδειγμα της υποενότητας 4.1.5 . Δίνοντας το επεξεργασμένο στιγμιότυπο στην Tesseract4Android λαμβάνουμε το επιθυμητό αποτέλεσμα που μπορεί να φανεί παρακάτω μέσα από ένα στιγμιότυπο του debugger.



Εικόνα 4.10 Αναγνωρισμένο κείμενο από το επεξεργασμένο στιγμιότυπο

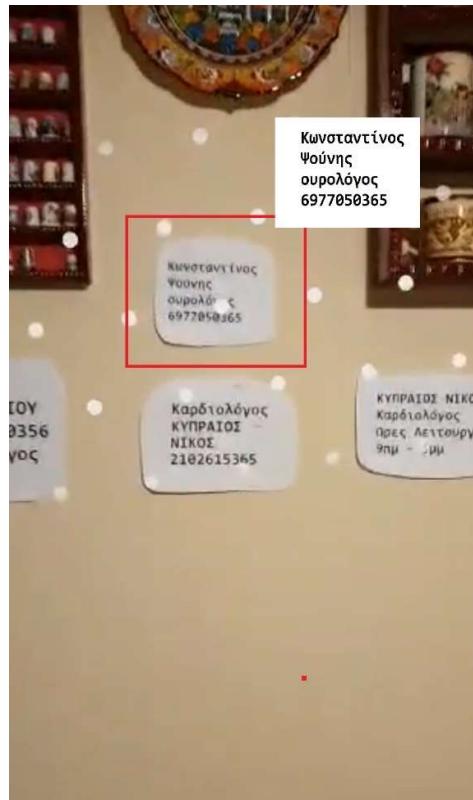
Για να καταλάβουμε την σημαντικότητα της επεξεργασίας εικόνας παρακάτω μπορούμε να δούμε ένα στιγμιότυπο από τον debugger στην περίπτωση που δεν είχε γίνει επεξεργασία εικόνας.



Εικόνα 4.11 Αναγνωρισμένο κείμενο χωρίς επεξεργασία

Στο συγκεκριμένο παράδειγμα τα λάθη της αναγνώρισης δεν είναι καταλυτικά για την συνέχιση των διαδικασιών. Σε παραδείγματα όμως που η φωτογραφία μπορεί να περιέχει περισσότερες πληροφορίες όπως είναι στο παράδειγμα μας «προσομοίωση κτιρίου επαγγελματιών» τότε θα βρίσκαμε περισσότερα λάθη που θα έπαιζαν σημαντικό ρόλο.

Παρακάτω θα δούμε ένα πιο περίτλοκο παράδειγμα που περιέχει περισσότερες πληροφορίες και επίσης περιέχει μια μίξη κεφαλαίων και μη γραμμάτων που μπορεί να περιέχουν τόνους. Για την ορθή αναζήτηση της επιθυμητής αναζήτησης ζητάμε αλφαριθμητικά με κεφαλαία γραμματά και χωρίς τόνους επειδή έτσι είναι αποθηκευμένα στην βάση μας. Επομένως κάθε φορά το αναγνωρισμένο αλφαριθμητικό ακολουθεί μια διαδικασία «καθαρισμού» πριν δοθεί στην Tesseract.



Εικόνα 4.12 Η «ταμπέλα» που αναλύεται

Πριν όμως αναλύσουμε τον «καθαρισμό» του αλφαριθμητικού ας δούμε πως είναι το αποτέλεσμα της συγκεκριμένης σάρωσης μετά από επεξεργασία, χωρίς επεξεργασία και τέλος χωρίς να χρησιμοποιήσουμε το Tesseract4Android αλλά το <https://github.com/rmtheis/tess-two> που αποτελεί την αρχική έκδοση Tesseract που είχαμε χρησιμοποιήσει.

```
▼ □ srcText[0] = "Κωνσταντίνος\nΨυώνης\nουρολόγος\nθΓΓ" ... View
  f count = 73
```

Εικόνα 4.13 Αναγνωρισμένο κείμενο με επεξεργασία εικόνας

```
▼ □ srcText = {String[1]@10659}
  ► □ 0 = "ωωωωωωω \n\nΚωνσταντίνος\nΨυώνης\nουρολόγος\nθΓΓ" ... View
```

Εικόνα 4.14 Αναγνωρισμένο κείμενο χωρίς επεξεργασία εικόνας

```

▼ srcText = {String[1]@11428}
► 0 = "ωωωωωωωω\η\ηΚωνσταντίνος\η\ηΨιώνης\η\ηουρολόνος\η\ηεγννθθθεβ\η\ηι" ... View

```

Eikόνα 4.15 Αναγνωρισμένο κείμενο όχι με Tesseract4Android

Όπως παρατηρείται η επεξεργασία εικόνας είναι ιδιαίτερα σημαντική καθώς μπορεί να μας βοηθήσει να αποφύγουμε πιθανά «σκουπίδια» στο αναγνωρισμένο κείμενο μας.

Αν προσέξουμε και στις τρεις φωτογραφίες δεν έχουν αναγνωριστεί καθόλου αριθμοί καθώς παρόλο που υπάρχουν στην ταμπέλα. Αυτό συμβαίνει επειδή όπως αναφέραμε και στο κεφάλαιο 3 έχουμε βάλει τους αριθμούς σε BlackList όπως φαίνεται παρακάτω επειδή δεν μας είναι χρήσιμοι πουθενά παρά μόνο γεμίζουν το αναγνωρισμένο κείμενο με περισσότερους χαρακτήρες. Παρόλα αυτά όπως είναι εμφανές και στα τρία παραπάνω στιγμότυπα του debugger έχει προσπαθήσει να αναγνωριστεί κάτι μετά το επώνυμο που υπάρχει το τηλέφωνο. Τέτοια παρόμοια λάθη μπορούν να γίνουν συχνά, καθώς η tesseract δεν μας δίνει την καλύτερη δυνατή ακρίβεια που θα θέλαμε να έχουμε, πόσο μάλλον χωρίς την χρήση της Tesseract4Android και ης OpenCV.

```

mTess.setVariable(TessBaseAPI.VAR_CHAR_WHITELIST, value: "ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΡΣΤΥΦΧΨΩαβγδεζηθικλμνξορστυφχψως" +
    "ιάόέώώέώέώ");
mTess.setVariable(TessBaseAPI.VAR_CHAR_BLACKLIST, value: "!@#$%^&*()_+=-[{}]{:;:\\"\\|~,./<>?0123456789");

```

Eikόνα 4.16 Blacklist και Whitelist της Tesseract στην εφαρμογή

4.1.6 Αναζήτηση στην Βάση δεδομένων

Αρχικά, για την διαδικασία της βάσης δεδομένων πρέπει να επεξεργαστούμε ανάλογα αναγνωρισμένο αλφαριθμητικό εφόσον χρειάζεται. Όπως είδαμε παραπάνω το αλφαριθμητικό περιέχει χαρακτήρες αλλαγής γραμμής /n και επίσης πολλές από τις αναγνωρισμένες λέξεις τονίζονται και είναι σε μικρά γράμματα. Οι πληροφορίες της βάσης δεδομένων που θα χρειαστεί να αναζητήσουμε περιέχουν κεφαλαία γράμματα μόνο. Επομένως, πριν την αναζήτηση καθαρίζουμε το αλφαριθμητικό από τόνους, το μετατρέπουμε σε κεφαλαία γράμματα και τέλος σπάμε της λέξεις σε έναν πίνακα αλφαριθμητικών. Χρειάζεται να σπάσουμε το αλφαριθμητικό επειδή στην συνέχεια θα γίνει αναζήτηση για κάθε λέξη ξεχωριστά.

```

srcText[0] = srcText[0].replaceAll( regex: "[\\t\\n\\r]+", replacement: " " );
srcText[0] = stripAccents(srcText[0]);

```

▼ **oo** srcText[0] = "Κωνσταντίνος Ψούνης ουρολόγος ΘΓΓ"
f count = 67

Εικόνα 4.17 Μετατροπή χαρακτήρων αλλαγής γραμμής σε κενά

```

private String stripAccents(String s)
{
    s = Normalizer.normalize(s, Normalizer.Form.NFD);
    s = s.replaceAll( regex: "[\\p{InCombiningDiacriticalMarks}]", replacement: "" );
    return s;
}

```

▼ **oo** srcText[0] = "Κωνσταντίνος Ψουνης ουρολογος ΘΓΓ"
f count = 67

Εικόνα 4.18 Αφαίρεση τόνων

▼ **p** text_array = {String[4]}@10450
▶ 0 = "ΚΩΝΣΤΑΝΤΙΝΟΣ"
▶ 1 = "ΨΟΥΝΗΣ"
▶ 2 = "ΟΥΡΟΛΟΓΟΣ"
▶ 3 = "ΘΓΓ"

Εικόνα 4.19 Χωρισμός αλφαριθμητικού σε λέξεις

Με τον χωρισμό του αναγνωρισμένου αλφαριθμητικού σε λέξεις μπορούμε να αρχίσουμε την αναζήτηση στην βάση.

Όπως έχουμε αναφέρει η αναζήτηση γίνεται με βάση το επώνυμο και την πιο πρόσφατη τοποθεσία του χρήστη. Ακολουθεί μια διαδοχική αναζήτηση με όλα τις λέξεις που αναγνωρίστηκαν μέχρι να βρεθεί το επιθυμητό κλειδί «επώνυμο».

Όπως ειπώθηκε στο κεφάλαιο 3 μόλις βρεθεί το επώνυμο γίνεται μια αναζήτηση όλων των διευθύνσεων που αντιστοιχούν στο συγκεκριμένο επώνυμο (σε περίπτωση διπλοτύπων) και γίνεται η σύγκριση με βάση την απόσταση από την πιο πρόσφατη τοποθεσία του χρήστη.

```

@Query("select address from professional_table where surname like :name")
LiveData<List<String>> getAllAddresses(String name);

```

Εικόνα 4.20 Query για την εύρεση όλων των διευθύνσεων των διπλότυπων επωνύμων

Αναλυτικότερα, στα παρακάτω στιγμιότυπα έχουμε χρησιμοποιήσει τον Database Inspector της εφαρμογής για να δείξουμε ότι στο δικό μας παράδειγμα η ταμπέλα μας αποτελεί διπλότυπο.

	id	profName	surname	description	category	address	officePhone	mobilePhone	email	website
1	25	ΚΩΝΣΤΑΝΤΙΝΟΣ	ΨΩΥΝΗΣ	Νεφρολόγος-Πα Νεφρολόγος		Κηφισίας 8, Αθήνα 2107488199	6947221102	mailto:konpsounis	http://www.nefrol...	
2	36	ΚΩΝΣΤΑΝΤΙΝΟΣ	ΨΩΥΝΗΣ	Ουρολογικό Προβ Ουρολόγος		Κωνσταντινούπόλεως 2105489653	6977052635	mailto:nikspuropo	https://www.cs.u...	

Εικόνα 4.21 Database Inspector – query για «Ψωύνης»

address
Κηφισίας 8, Αθήνα - Αμπελόκηποι, 11526, ΑΤΤΙΚΗΣ
Κωνσταντινούπόλεως 400, Αγιοι Ανάργυροι, 13562, ΑΤΤΙΚΗΣ

Εικόνα 4.22 Η διεύθυνση που ανήκει η ταμπέλα του παραδείγματος

Παρακάτω βλέπουμε την σύγκριση που γίνεται για την εύρεση του σωστού επαγγελματία. Εάν η διεύθυνση είναι το πολύ 20 μέτρα μακριά από την πιο πρόσφατη τοποθεσία του χρήστη τότε αυτή είναι η διεύθυνση του επαγγελματία που αναζητούμε. Επιλέξαμε 20 μέτρα διότι το κτίριο που έγινε η συγκεκριμένη προσομοίωση είναι μεγάλο και στον χώρο που έλαβε μέρος το πείραμα υπήρχε κάποια μικρή απόκλιση της διεύθυνσης.

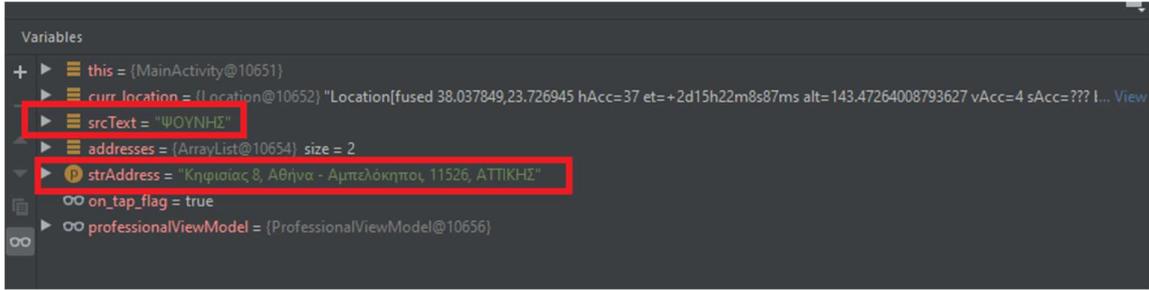
```

if (locationMatcher.getDistanceFrom(curr_location) < 20.0){
    Toast.makeText( context: this, text: "Αντιστοίχην Διεύθυνσεων ολοκληρώθηκε!", Toast.LENGTH_SHORT).show();
    getProfInfo(strAddress, srcText);
}

```

Εικόνα 4.23 Έλεγχος απόστασης διεύθυνσης βάσης – πιο πρόσφατης τοποθεσίας χρήστη

Στα επόμενα στιγμιότυπα μπορούμε να δούμε τις διευθύνσεις των επαγγελματιών που βρέθηκαν από την αναζήτηση στην βάση και την σχετική απόσταση που υπολογίστηκε.



Εικόνα 4.24 Διπλότυπο – Επαγγελματίας με ίδιο επώνυμο

```

public float getDistanceFrom(Location current_location) { current_location: "Location[fused 38.037849,23.726945 hAcc=37 et=+2d15h22m8s87ms alt=143.47264008793627 vAcc=4 sAcc=??? l... View
    float[] distance = new float[1]; distance: {6443.6714}
    Location.distanceBetween(p1.latitude, p1.longitude, current_location.getLatitude(), current_location.getLongitude(), distance);

    return distance[0]; distance: {6443.6714}
}

```

Εικόνα 4.25 Απόσταση χρήστη από την συγκεκριμένη διεύθυνση (περίπου 6,5 km)



Εικόνα 4.26 Επαγγελματίας που αναζητούμε

```

public float getDistanceFrom(Location current_location) { current_location: "Location[fused 38.037849,23.726945 hAcc=37 et=+2d15h2,
    float[] distance = new float[1]; distance: [18.032534]
    Location.distanceBetween(p1.getLatitude(), p1.getLongitude(), current_location.getLatitude(), current_location.getLongitude(), distance);

    return distance[0]; distance: 18.032534
}

```

Εικόνα 4.27 Απόσταση χρήστη από την συγκεκριμένη διεύθυνση (περίπου 18 m)

Μετά την ορθή αντιστοίχιση των διευθύνσεων προχωράμε στο στάδιο όπου πρέπει να ανακτήσουμε την επιθυμητή πληροφορία και να την εμφανίσουμε στην εικονική πινακίδα.

```

@Query("select * from professional_table WHERE address LIKE :office_address AND surname LIKE :name")
LiveData<Professional> findAll(String office_address, String name);

```

Εικόνα 4.28 Query για την ανάκτηση όλων των πληροφοριών του επαγγελματία

```

▼ └─ info = {Professional@11368}
  └─ f address = "Κωνσταντινουπόλεως 400, Άγιοι Ανάργυροι, 13562, ΑΤΤΙΚΗΣ"
  └─ f category = "Ουρολόγος"
  └─ f description = "Ουρολογικα Προβλήματα"
  └─ f email = "mailto:nikspyrropoulos@gmail.com"
  └─ f id = 36
  └─ f mobilePhone = "6977052635"
  └─ f officePhone = "2105489653"
  └─ f profName = "ΚΩΝΣΤΑΝΤΙΝΟΣ"
  └─ f surname = "ΨΟΥΝΗΣ"
  └─ f website = "https://www.cs.uoi.gr"

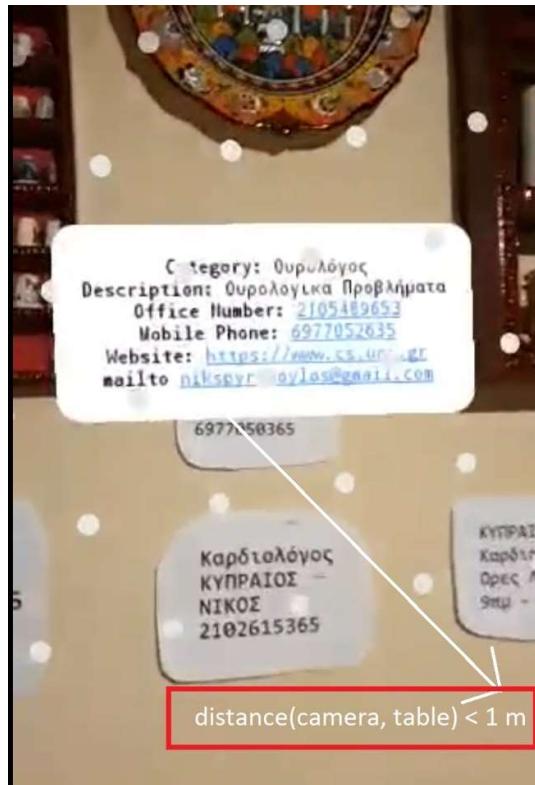
```

Εικόνα 4.29 Ανάκτηση πληροφορίας μετά την κλήση του Query

4.1.7 Εμφάνιση εικονικής πινακίδας

Όπως έχει αναφερθεί στο κεφάλαιο 3 μόλις γίνει το πάτημα από τον χρήστη τότε δημιουργείται ένα anchor και εν συνεχεία ένα node ακριβώς εκεί που πάτησε έτσι ώστε να τοποθετηθεί αργότερα η εικονική πινακίδα. Επίσης για την συμπλήρωση της εικονικής πινακίδας υπάρχουν δυο αλφαριθμητικά που είναι πεδία της MainActivity και χρησιμοποιούνται για το γέμισμα του TextView της εικονικής πινακίδας ανάλογα με την απόσταση του χρήστη. Ο node που μόλις αναφέρθηκε γίνεται ενεργοποιείται την στιγμή που τα δύο αλφαριθμητικά γεμίσουν δηλαδή μόλις έχει γίνει η σωστή αντιστοίχιση των διευθύνσεων.

Τέλος για τα κείμενα που είναι linkable όπως τηλέφωνο, ηλεκτρονική διεύθυνση κλπ χρησιμοποιήσαμε το **info_text.setAutoLinkMask(Linkify.ALL)**; για να τους δώσουμε χρήστη όπως φαίνεται παρακάτω.



Εικόνα 4.30 Απόσταση χρήστη από την εικονική πινακίδα μικρότερη του 1m

```
▼ oo info_text.getText() = {SpannableString@12921} "Category: Θυρυλόγος\nDescription: Ουρολογικά Προβλήματα\nOffice Number: 2105489653\n\noo distanceCamera(currentAnchor.getPose(), frame.getCamera().getPose()); = 0.6744598
```

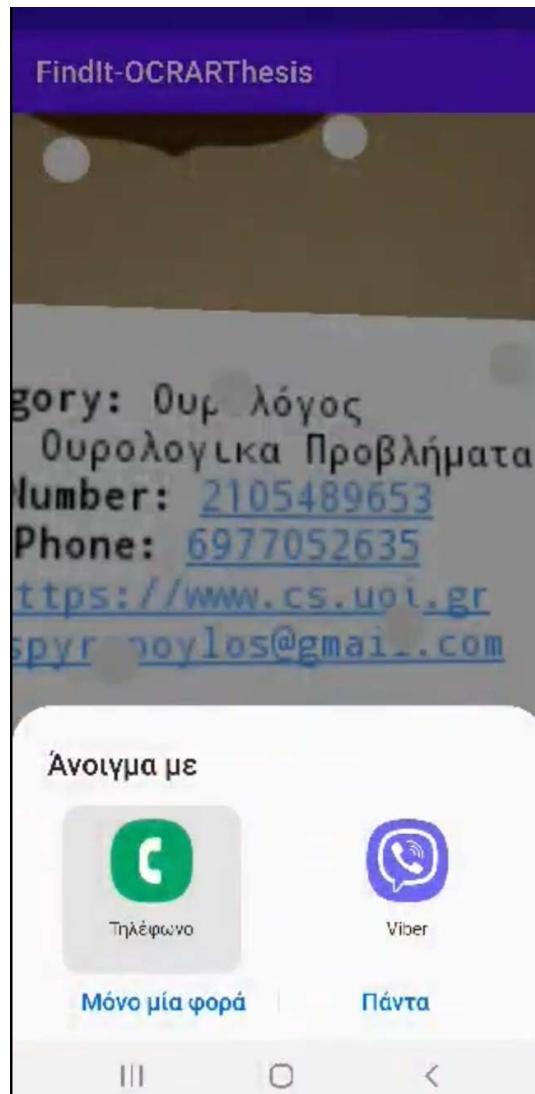
Εικόνα 4.30a Απόσταση χρήστη από την εικονική πινακίδα μικρότερη του 1m



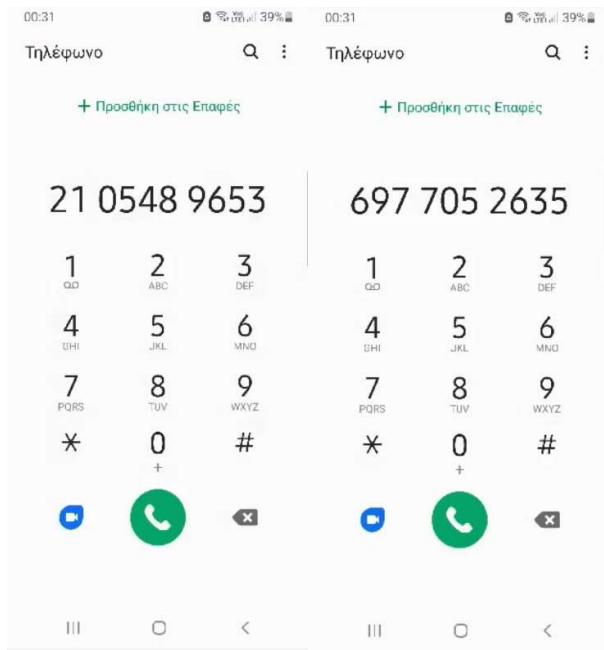
Εικόνα 4.31 Απόσταση χρήστη από την εικονική πινακίδα μεγαλύτερη του 1m

```
▼ oo info_text.getText() = (SpannableString@12959) "ΨΟΥΝΗΣ\nΚΩΝΣΤΑΝΤΙΝΟΣ" ... View
  oo distanceCamera(currentAnchor.getPose(), frame.getCamera().getPose()) = 1.0012224
```

Εικόνα 4.31β Απόσταση χρήστη από την εικονική πινακίδα μεγαλύτερη του 1m



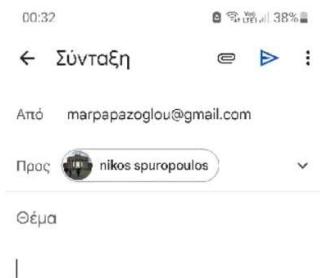
Εικόνα 4.32 Μεταφορά σε εφαρμογή κλήσης τηλεφώνου λόγω πατήματος του σταθερού (παρομοίως και για κινητό).



Εικόνα 4.33 Δυνατότητα κλήσης

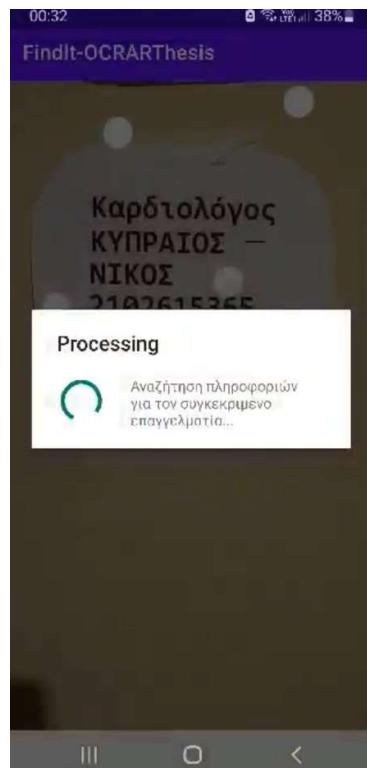


Εικόνα 4.34 Πλοιόγηση στο website με πάτημα αυτού (στο πείραμα μας έχει χρησιμοποιηθεί το website της σχολής)

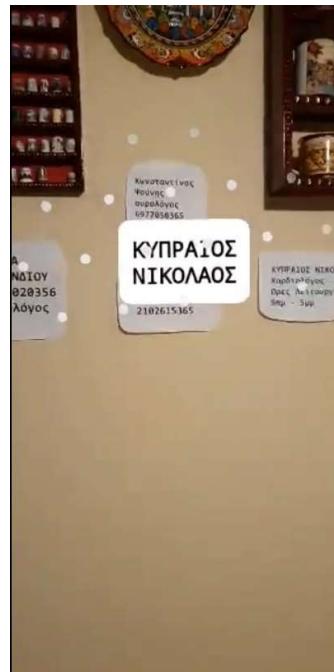


Εικόνα 4.35 Μεταφορά στην e-mail εφαρμογή του κινητού με πάτημα στο email (δυνατότητα επιλογής εφαρμογής σε περίπτωση διάφορων)

Τέλος όπως έχουμε αναφέρει η διαδικασία αυτή μπορεί να επαναληφθεί πολλές φορές χωρίς κανένα απολύτως πρόβλημα. Η εικονική πινακίδα πλέον εξαφανίζεται και στην θέση της απλά αντικαθιστάτε η καινούργια.



Εικόνα 4.36 Ο χρήστης πάτησε σε νέα πινακίδα του κτιρίου



Εικόνα 4.37 Εξαφάνιση της προηγούμενης πινακίδας και εμφάνιση της νέας (το συγκεκριμένο στιγμιότυπο είναι πάνω από 1 m μακριά από την πινακίδα για αυτό και βλέπουμε μόνο το ονοματεπώνυμο)

4.2 Δυσκολίες που εντοπίσαμε

Όπως όλες οι εφαρμογές που ασχολούνται με πραγματικά δεδομένα όπως φωτογραφίες έτσι και η δικιά μας αντιμετώπισε κάποιες δυσκολίες. Συγκεκριμένα σε περιπτώσεις χαμηλού φωτισμού ή περιπτώσεις που κάποια ταμπέλα δεν είναι τόσο καθαρή πολλές φορές το κείμενο που αναγνωρίζεται δεν είναι το επιθυμητό και τότε εμφανίζεται στην οθόνη ένα μήνυμα που λέει στον Χρήστη να προσπαθήσει ξανά. Σε μελλοντικές εργασίες πάνω στην συγκεκριμένη εφαρμογή θα μπορούσαμε να επικεντρωθούμε πιο πολύ στην επεξεργασία της εικόνας για να μπορέσουμε να έχουμε τα καλύτερα δυνατά αποτελέσματα αικόμη σε συνθήκες που κάνουν το έργο μας δύσκολο.

Επιπρόσθετα, μια δεύτερη δυσκολία που συναντήσαμε ήταν η εξής. Σκεφτήκαμε ότι μπορεί να υπάρχουν κτήρια που έχουν γραφεία επαγγελματιών με ίδια επώνυμα. Με άλλα λόγια, διπλότυπα δεδομένα στην ίδια διεύθυνση. Σε αυτή την περίπτωση λοιπόν δεν μας είναι αρκετό να αναζητήσουμε δεδομένα στην βάση μας με βάση το επώνυμο και την διεύθυνση καθώς και τα δυο αυτά θα είναι ίδια και για τους όσους επαγγελματίες είναι συνονόματοι στο εκάστοτε κτίριο. Μια πρώτη σκέψη μας ήταν να ελέγχουμε το όνομα του επαγγελματία αφού δειχτεί ότι μόνο από την διεύθυνση και το επώνυμο δεν μπορεί να βρεθεί ορθό αποτέλεσμα. Αυτό όμως αποδείχτηκε πως δεν είναι μια καλή υλοποίηση διότι όπως συναντήσαμε πολλές πινακίδες έχουν διαφορετικό όνομα απ' ότι η βάση μας, για παράδειγμα «Νικόλαος» στην βάση και «Νίκος» πινακίδα που αποτελούν δυο διαφορετικά αλφαριθμητικά. Μια δεύτερη σκέψη ήταν να γίνεται αναζήτηση με τον αριθμό τηλεφώνου αλλά δυστυχώς αποδείχτηκε ότι και αυτό δεν θα μπορούσε να λειτουργήσει επιτυχώς καθώς σε πολλές ταμπέλες δεν αναγράφεται αριθμός τηλεφώνου. Για αυτούς τους δύο λόγους βρεθήκαμε σε αδιέξοδο και δεν μπορέσαμε να βρούμε μια εφικτή λύση για αυτή την περίπτωση προς το παρόν.

Κεφάλαιο 5. Επίλογος

Στην ενότητα αυτή συνοψίζουμε τη συνεισφορά και τα αποτελέσματα της εργασίας και παραθέτουμε σκέψεις για μελλοντικές επεκτάσεις της.

5.1 Σύνοψη και συμπεράσματα

Ο σκοπός της παρούσας διπλωματικής εργασίας ήταν η κατασκευή μιας καινοτόμας και επεκτάσιμης εφαρμογής ανάκτησης πληροφοριών με χρήση της τεχνολογίας οπτικής αναγνώρισης χαρακτήρων και της επαυξημένης πραγματικότητας.

Υλοποιήσαμε μια εφαρμογή που μπορεί να προσφέρει στον χρήστη πληροφορίες για κάποιον επαγγελματία που τον ενδιαφέρει πατώντας απλά μια φορά πάνω στην οθόνη του στοχεύοντας με την κάμερα σε μια ταμπέλα γραφείου. Δώσαμε την δυνατότητα στον χρήστη να μπορεί να έχει πρόσβαση μέσω μιας εικονικής πινακίδας στο κινητό ή σταθερό τηλέφωνο, την ηλεκτρονική διεύθυνση ή/ και την ιστοσελίδα του επαγγελματία σαρώνοντας απλά την ταμπέλα του που είναι τοποθετημένη έξω από το γραφείο του. Ο χρήστης μπορεί να ενεργεί με βάση στοιχεία που μπορούσε να δει στην πραγματική ταμπέλα αλλά φυσικά δεν ήταν διαχειρίσιμα.

Οι μεγαλύτερες δύο δυσκολίες της εφαρμογής ήταν η σωστή αναγνώριση κειμένου λόγω κακής ποιότητας στιγμότυπου και ο σωστός τρόπος αναζήτησης στην βάση. Αρχικά, όπως αναφέρθηκε και στο κεφάλαιο 4 είχαμε χρησιμοποιήσει την Tess-Two που αποτελεί την Tesseract μηχανή για android αλλά τα αποτελέσματα που λαμβάναμε ήταν σπανίως ορθά. Μετά από αρκετή έρευνα καταλήξαμε στην χρησιμοποίηση της Tesseract4Android που αποτελεί ένα fork της Tess-Two και διαφέρει επειδή πραγματοποιεί κάποια επεξεργασία εικόνας προτού δώσει την εικόνα για αναγνώριση. Επίσης, χρησιμοποιήσαμε κάποιους αλγορίθμους μέσω της βιβλιοθήκης OpenCV που μας βοήθησαν ιδιαίτερα στην μακράν καλύτερη αναγνώριση χαρακτήρων.

Η δεύτερη δυσκολία που είχαμε αφορά τον τρόπο αναζήτησης στην βάση. Η πρώτη σκέψη μας ήταν να γίνεται αναζήτηση με βάση ολόκληρο το ονοματεπώνυμο του επαγγελματία. Κάτι που αποδείχθηκε ως μια κακή πρακτική καθώς το μοτίβο του ονοματεπώνυμού καθώς και το όνομα μπορεί να διαφέρει από την ταμπέλα στην βάση. Για παράδειγμα, σε κάποια πινακίδα μπορεί να αναγράφεται «Νικόλαος Σπυρόπουλος» και εμείς να έχουμε στην βάση αποθηκευμένο τον επαγγελματία ως «Σπυρόπουλος Νίκος». Εν συνεχείᾳ, παρατηρήθηκε ότι η στήλη ονοματεπώνυμο στην βάση μας από τα

στοιχεία που είχαμε συλλέξει από το vrisko.gr ακολουθούσε ένα συγκεκριμένο μοτίβο. Αναλυτικότερα, το μοτίβο αυτό είχε πρώτα το επώνυμο και μετά το όνομα και έτσι αποφασίσαμε να σπάσουμε την συγκεκριμένη στήλη σε δυο (όνομα και επώνυμο) έτσι ώστε να μπορέσουμε να κάνουμε την αναζήτηση μόνο με βάση το επώνυμο. Η συγκεκριμένη σκέψη λειτούργησε επαρκώς και πλέον η εφαρμογή πραγματοποιεί την διαδικασία αναζήτησης στην βάση με αυτό τον τρόπο.

Συμπερασματικά, η πλήρης κατασκευή της συγκεκριμένης εφαρμογής με τις διάφορες περίπλοκες λειτουργίες της που διέφεραν η μια από την άλλη αποδείχθηκε ιδιαίτερα ενδιαφέρουσα και χρήσιμη για την ανάπτυξη των γνώσεων μου σε νέους τομείς για εμένα όπως είναι η επαυξημένη πραγματικότητα.

5.2 Μελλοντικές επεκτάσεις

Υπάρχουν κάποια σχέδια επέκτασης της συγκεκριμένης εφαρμογής. Συγκεκριμένα:

- Προσθήκη περισσότερων πινάκων στην βάση. Ανάκτηση δεδομένων για διάφορες κατηγορίες όπως για παράδειγμα εκθέματα.
- Χρήση της εφαρμογής σε εσωτερικούς χώρους. Αναζήτηση με βάση την τοποθεσία χρήστη στο εσωτερικό κτιρίου. Όπως για παράδειγμα χρήση της στο κτίριο του πανεπιστημίου για εύρεση πληροφοριών για καθηγητές που μπορεί να απουσιάζουν.
- Αποθήκευση πληροφοριών που επιθυμεί ο χρήστης να έχει μόλις φύγει από την τοποθεσία που βρίσκεται η πραγματική πινακίδα. Δημιουργία μιας δεύτερης τοπικής βάσης μέσω της οποίας θα μπορεί να δει ο χρήστης αποθηκευμένες πληροφορίες στο μέλλον.
- Αναγνώριση και αγγλικών χαρακτήρων για χρήση της εφαρμογής όχι μόνο σε περιοχές της Ελλάδος.
- Δυνατότητα του χρήστη να μπορέσει να αλλάξει την γλώσσα εμφάνισης της εικονικής πινακίδας με αυτόματη μετάφραση.
- Ανάπτυξη της εφαρμογής και σε iOS συστήματα.

Βιβλιογραφία

- [1] <https://www.androidpolice.com/2019/10/11/google-lens-50-million-downloads/>
- [2] <https://thenextweb.com/news/google-translates-camera-detect-88-languages>
- [3] <https://www.geckoandfly.com/11934/google-language-translate-offline/>
- [4] https://el.wikipedia.org/wiki/%CE%9F%CF%80%CF%84%CE%B9%CE%BA%CE%AE_%CE%91%CE%BD%CE%B1%CE%B3%CE%BD%CF%8E%CF%81%CE%B9%CF%83%CE%B7_%CE%A7%CE%B1%CF%81%CE%B1%CE%BA%CF%84%CE%AE%CF%81%CF%89%CE%BD?fbclid=IwAR2nsBkn8xK8AdwgVfGmgO9fAE971ph0qpXstXV8JX1i8uDunyM_SNOZdhA
- [5] https://www.secnews.gr/160447/arcore-sdk-from-google/?fbclid=IwAR2PrOKheYQIimoYqDuYhAZrcCorT1Zlrvz8o4vNEV34p_OWLqxju4qls8
- [6] https://www.you.gr/discover/fevrouarios-2018/arkit-epafximeni-pragmatikotita-me-ti-mia?fbclid=IwAR1Xrew5Rl2PO4pmL2nHM4iXphjGelPMzVflkjFihXK6tpTDK_Rz6KrVy0
- [7] https://developer.android.com/studio?gclid=Cj0KCQjwtrSLBhCLARIsACh6RmhLB65FEiCu1MO6p_5aCCj8XsU7X8cCMdahPQ77atc-so_cnMLPvrcaAlhtEALw_wcB&gclsrc=aw.ds
- [8] <https://www.you.gr/discover/fevrouarios-2018/arkit-epafximeni-pragmatikotita-me-ti->

[mia?fbclid=IwAR1Xrew5Rl2PO4pmL2nHM4iXphJGelPMzVflkujFihXK6tpTDK_Rz6KrVy0](#)

- [9] <https://scrapy.org/>
- [10] <https://sqlitestudio.pl/>
- [11] <https://developer.android.com/training/data-storage/room>
- [12] [https://www.researchgate.net/publication/265087843 Optical character recognition applied on receipts printed in macedonian language/_figures?lo=1](https://www.researchgate.net/publication/265087843_Optical_character_recognition_applied_on_receipts_printed_in_macedonian_language/_figures?lo=1)
- [13] https://github.com/adaptech-cz/Tesseract4Android?fbclid=IwAR10DYppXyhXM5MQ8aEGFNH7H7nobxmGoD_pHW-vdRt6sWJw4stTie1XG4
- [14] <https://developers.google.com/ar>
- [15] <https://developers.google.com/sceneform/develop>
- [16] <https://github.com/ThomasGorissee>
- [17] <https://cloud.google.com/vision>
- [18] <https://tesseract-ocr.github.io/tessdoc/ImproveQuality?fbclid=IwAR0diKbDTPf3CwE-g3mN8h54BLFtTs77WuuVVG3i6R3aliENsksJmAagpk#rescaling>
- [19] Vrisko.gr