

**ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ**

**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

πτυχιακη εργασια

|  |  |
| --- | --- |
| Τίτλος πτυχιακής εργασίας | Εφαρμογή android |
| Ονοματεπώνυμο φοιτητή/τριας | Πέττα Αρετή |
| Αριθμός Μητρώου φοιτητή/τριας | Π18231 |
| Email επικοινωνίας | aretpett@gmail.com |
| Επιβλέπων καθηγητής | Αλέπης Ευθύμιος |
| Ημερομηνία παράδοσης | 01/10/2022 |
| Κατεύθυνση | Τεχνολογίες Λογισμικού και Ευφυή Συστήματα |

Περιεχόμενα

[ΕΥΧΑΡΙΣΤΙΕΣ 4](#_Toc115694098)

[ΠΡΟΛΟΓΟΣ 5](#_Toc115694099)

[ΠΑΡΟΥΣΙΑΣΗ ΘΕΜΑΤΟΣ - ΕΙΣΑΓΩΓΗ 6](#_Toc115694100)

[ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ 7](#_Toc115694101)

[ΚΩΔΙΚΑΣ 8](#_Toc115694102)

[alertDialog package 9](#_Toc115694103)

[login package 10](#_Toc115694104)

[navmenu package 11](#_Toc115694105)

[navmenu/account package 12](#_Toc115694106)

[navmenu/detection package 13](#_Toc115694107)

[navmenu/translation package 15](#_Toc115694108)

[ΧΡΗΣΗ ΤΟΥ ML KIT ΤΗΣ GOOGLE-FIREBASE 16](#_Toc115694109)

[IMPORT OPENCV LIBRARY 18](#_Toc115694110)

[ΣΥΝΔΕΣΗ ΜΕ ΒΑΣΗ – REALTIME DATABASE 20](#_Toc115694111)

[STORAGE 24](#_Toc115694112)

[USERS’ AUTHENTICATION 25](#_Toc115694113)

[ΥΠΟΣΤΗΡΙΖΟΜΕΝΕΣ ΓΛΩΣΣΕΣ 26](#_Toc115694114)

[CLASS DIAGRAM 27](#_Toc115694115)

[API ΓΙΑ ΕΞΑΓΩΓΗ ΣΤΑΤΙΣΤΙΚΩΝ ΔΕΔΟΜΕΝΩΝ 28](#_Toc115694116)

[ΠΑΡΟΥΣΙΑΣΗ ΙΔΕΑΣ 28](#_Toc115694117)

[REQUESTS ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ 29](#_Toc115694118)

[ΥΛΟΠΟΙΗΣΗ API 30](#_Toc115694119)

[ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ PROJECT 30](#_Toc115694120)

[ΣΥΝΔΕΣΗ ΜΕ FIREBASE 31](#_Toc115694121)

[ΚΩΔΙΚΑΣ 32](#_Toc115694122)

[ΠΑΡΑΓΕΙΓΜΑΤΑ ΕΚΤΕΛΕΣΗΣ APP 36](#_Toc115694123)

[ΠΑΡΑΔΕΙΓΜΑΤΑ ΕΚΤΕΛΕΣΗΣ API 37](#_Toc115694124)

[ΠΗΓΕΣ 45](#_Toc115694125)

# ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον καθηγητή μου, Ευθύμιο Αλέπη, που με εμπιστεύτηκε για την ανάπτυξη της πτυχιακής μου εργασίας και μου έδωσε την ευκαιρία να δουλέψουμε μαζί σε αυτό το project.

# ΠΡΟΛΟΓΟΣ

Το θέμα της πτυχιακής εργασίας αφορά software engineering σε συνδυασμό με machine learning. Είναι γνωστό άλλωστε ότι οι δύο αυτές τεχνολογίες συνδέονται πολύ στενά και συχνά μεταξύ τους και τα αποτελέσματά τους είναι σπουδαία και σημαντικά σε πολλούς τομείς. Οι εφαρμογές τους συναντώνται τόσο στην καθημερινότητά μας όσο και σε πιο ειδικές καταστάσεις. Από απλές εφαρμογές που μπορούν να βοηθήσουν έναν άνθρωπο στη ρουτίνα του μέχρι το επιστημονικό επίπεδο όπου βοηθάνε σε έρευνες και άλλα επιστημονικά ευρήματα.

Οι δυσκολίες που κρίνουν επιτακτική ανάγκη την δημιουργία εφαρμογών, οι οποίες θα εξυπηρετήσουν τους ανθρώπους είναι πολλές. Ένα από τα προβλήματα που προκύπτουν στα ταξίδια σε ξένες χώρες είναι η δυσκολία στην επικοινωνία με τους κατοίκους αυτών των χωρών, όταν τουλάχιστον μία από τις δύο μεριές (ο ταξιδιώτης ή οι κάτοικοι της ξένης χώρας) δεν γνωρίζει μια γλώσσα ευρέως γνωστή, όπως τα αγγλικά. Επομένως, κάπως έτσι “γεννήθηκε” η ιδέα για την ανάπτυξη μιας εφαρμογής που θα εξυπηρετεί αυτούς τους ανθρώπους που ταξιδεύουν και δυσκολεύονται να επικοινωνήσουν ή να αναγνωρίσουν ένα κείμενο γραμμένο σε μια άγνωστη – για εκείνους – γλώσσα.

# ΠΑΡΟΥΣΙΑΣΗ ΘΕΜΑΤΟΣ - ΕΙΣΑΓΩΓΗ

Το θέμα της πτυχιακής, το οποίο επιλέχθηκε να υλοποιηθεί, αφορά εφαρμογή android η οποία απευθύνεται σε ταξιδιώτες/τουρίστες σε άλλες χώρες. Την εφαρμογή μπορούν να χρησιμοποιούν οι τουρίστες για να κατανοήσουν τη γλώσσα της χώρας στην οποία βρίσκονται και κατ’ επέκταση για να επικοινωνούν σε ένα βαθμό.

Κύριος στόχος της εφαρμογής είναι η εξαγωγή του κειμένου που περιέχεται σε μία εικόνα. Επομένως, η εφαρμογή πρέπει να είναι ικανή να αναγνωρίζει το κείμενο που περιέχει μία εικόνα, να αναγνωρίζει την γλώσσα του κειμένου και τέλος να μεταφράζει το κείμενο που εντόπισε από την γλώσσα την οποία βρήκε. Για την υλοποίηση όλων αυτών των διαδικασιών που πρέπει να εκτελεί η εφαρμογή θα αξιοποιηθεί το ml kit που προσφέρεται από την Firebase.

Η εφαρμογή απευθύνεται σε authenticated users, επομένως ένας user πρέπει να δημιουργήσει ένα account με το οποίο θα μπορεί να συνδέεται στην εφαρμογή. Εφόσον, έχει δημιουργήσει το account του, μπορεί να χρησιμοποιήσει όλες τις λειτουργίες που προσφέρονται από την συγκεκριμένη εφαρμογή.

Απαραίτητη κρίνεται η χρήση μιας database για την εφαρμογή. Αρκετές από τις πληροφορίες που χρησιμοποιεί ή παράγει η εφαρμογή πρέπει να αποθηκεύονται σε ένα database ώστε να μπορούν να ανακτηθούν όποτε είναι απαραίτητο.

Παρότι η εφαρμογή απευθύνεται στο κοινό που αναφέρθηκε νωρίτερα, από τα δεδομένα που συλλέγονται μέσω της χρήσης της εφαρμογής προκύπτουν μερικά στατιστικά τα οποία μπορούν να αξιοποιηθούν από άλλους φορείς. Για την παρουσίαση αυτών των δεδομένων θα δημιουργηθεί ένα API το οποίο θα υλοποιεί συγκεκριμένα endpoints.

# ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ

Η υλοποίηση όλης της ιδέας για την εργασία αυτή χωρίζεται σε δύο μέρη. Το πρώτο μέρος αφορά την ίδια την εφαρμογή που απευθύνεται στους τουρίστες. Το δεύτερο μέρος, το οποίο θα αναλυθεί αργότερα, αφορά την αποθήκευση και την διαχείριση των στατιστικών δεδομένων. Για αρχή θα αναλυθεί το πρώτο μέρος.

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκαν οι εξής τεχνολογίες:

* Η γλώσσα στην οποία γράφτηκε είναι η Java.
* Το ide που χρησιμοποιήθηκε είναι το Android Studio.
* Η database που χρησιμοποιήθηκε είναι η realtime database της firebase.
* Το authentication των users υλοποιήθηκε με το Authentication που προσφέρει η firebase.
* Για το text recognition, το language identification και το translation χρησιμοποιήθηκε το ml kit που προσφέρει και πάλι η firebase.
* Το OpenCV library (version 4.6.0) για το live capture του text πάνω σε image.

Μερικά από τα χαρακτηριστικά της εφαρμογής είναι τα εξής:

* Compile sdk: 31
* Target sdk: 31

## ΚΩΔΙΚΑΣ

Ο κώδικας της εφαρμογής είναι χωρισμένος σε packages για να είναι ευανάγνωστος, αλλά και για να μπορεί να γίνει εύκολα η συντήρησή του σε μελλοντικό χρόνο.

* com.apetta.detext\_app (main package)
  + alertDialog
  + login
  + navmenu
    - account
    - detection
    - translation

### alertDialog package

Το συγκεκριμένο package περιέχει όλα τα custom AlertDialogs. Αυτά τα custom είναι τα εξής:

* **ProgressAlertDialog**: Aφορά το progress μιας ενέργειας που εκτελεί η εφαρμογή και η οποία χρειάζεται κάποιο χρόνο ώσπου να ολοκληρωθεί. Στην αναμονή αυτής της ολοκλήρωσης εμφανίζεται το συγκεκριμένο AlertDialog και ενημερώνει τον χρήστη για την κατάσταση της ενέργειας. Το συγκεκριμένο κλείνει αυτόματα μόλις ολοκληρωθεί η ενέργεια. Η κλάση για αυτό είναι η ProgressAlertDialog.java και το αντίστοιχο layout είναι το progress\_alert\_dialog.xml.
* **TemperatureAlertDialog**: Αφορά την εμφάνιση της θερμοκρασίας και της τοποθεσίας, οι οποίες συλλέγονται από τους αισθητήρες της συσκευής. Στην περίπτωση αυτή, το επικαλυπτόμενο παράθυρο που εμφανίζεται κλείνει μόλις ο user πατήσει το κατάλληλο κουμπί. Η κλάση για αυτό είναι η TemperatureAlertDialog.java και το αντίστοιχο layout είναι το temperature\_alert\_dialog.xml.

### login package

Το συγκεκριμένο package περιέχει όλες τις κλάσεις που αφορούν το login του user στην εφαρμογή.

* **SignInActivity.java**: Η κλάση αυτή αφορά το activity για το sign in του user. Είναι το 1ο activity που εμφανίζεται στον user μετά την εγκατάσταση του app. Από αυτό το activity μπορεί είτε να κάνει login με τα email/password του είτε να μεταβεί στο activity του sign up και να πραγματοποιήσει εγγραφή. Αν ο user παραμένει συνδεδεμένος στο app τότε δεν εμφανίζεται το συγκεκριμένο activity και μεταβαίνει κατευθείαν στο main activity με τις δυνατότητες που προσφέρονται. Το αντίστοιχο layout για αυτό το activity είναι το activity\_sign\_in.xml.
* **SignUpActivity.java**: Η κλάση αυτή αφορά το activity για το sign up του user. Ο user μεταβαίνει σε αυτό μέσω του SignInActivity, όταν δεν έχει κάποιο account για να κάνει login. Τα στοιχεία που απαιτούνται για την δημιουργία ενός account είναι τα email/password με τα οποία θα μπορεί να κάνει login οποιαδήποτε άλλη στιγμή. Το αντίστοιχο layout για αυτό το activity είναι το activity\_sign\_up.xml.
* **User.java**: Η κλάση αυτή αφορά τους users ως entity. Οι users αποθηκεύονται στην database ως objects αυτής της κλάσης κι έτσι είναι εύκολο να ανακτηθούν και οι πληροφορίες για αυτούς, όταν κρίνεται απαραίτητο. Η αποθήκευση ενός user στην database γίνεται κατά τη δημιουργία ενός νέου account, δηλαδή στο Sign Up.

### navmenu package

Το συγκεκριμένο package ομαδοποιεί άλλα packages που αφορούν το bottom navigation menu του app. Το main activity του app έχει ένα bottom navigation bar, από το οποίο ο user μπορεί να επιλέξει κάθε φορά σε ποια δραστηριότητα θέλει να μεταβεί. Οι δραστηριότητες αυτές είναι για το image detection, για τον translator και την προβολή του account. Η κλάση που διαχειρίζεται τις επιλογές αυτές είναι η **NavMenu.java** και το αντίστοιχο layout είναι το activity\_nav\_menu.xml. Σε αυτό περιέχεται ένα BottomNavigationView στο οποίο φορτώνεται ένα άλλο .xml file, το bottom\_nav.xml, το οποίο είναι το bottom navigation bar.

### navmenu/account package

Το συγκεκριμένο package, account, αφορά όλες τις κλάσεις που χρησιμοποιούνται στην ενότητα account της εφαρμογής. Πιο συγκεκριμένα, η ενότητα account παρέχει πληροφορίες για τον ίδιο τον user, αλλά και τις αναζητήσεις που έχει κάνει με εικόνες που περιέχουν κείμενο. Περιέχει, δηλαδή, το ιστορικό που έχει αποθηκεύσει ο user. Το ιστορικό αναφέρεται στις εικόνες που εισήγαγε ο user για να εξεταστούν από την εφαρμογή.

* **AccountFragment.java**: Η κλάση αυτή αφορά το activity με το account του user. Σε αυτό περιέχονται οι βασικές πληροφορίες του user. Επιπλέον, μέσα από αυτό το activity ο user μπορεί να δει την θερμοκρασία στην περιοχή στην οποία βρίσκεται, χρησιμοποιώντας τους αντίστοιχους αισθητήρες, καθώς και να δει το ιστορικό από αυτά που έχει αποθηκεύσει. Το ιστορικό αφορά εικόνες τις οποίες εισήγαγε ο user με σκοπό την εξαγωγή του κειμένου από αυτές. Τις εικόνες αυτές επιλέγει ο user αν θα τις προσθέσει στο ιστορικό του ή όχι. Στην περίπτωση που τις αποθηκεύσει, μπορεί να τις ανακτήσει και πάλι από το ιστορικό που περιεγράφηκε παραπάνω. Το αντίστοιχο layout του συγκεκριμένου activity είναι το fragment\_account.xml.
* **DetailsOfCardHistoryActivity.java**: Η κλάση αυτή αφορά το activity στο οποίο εμφανίζονται όλα τα δεδομένα που είχαν προκύψει από το text recognition και τα οποία είχαν αποθηκευτεί στο history του user. Αυτά που εμφανίζονται στο activity είναι η εικόνα εισαγωγής, η γλώσσα στην οποία εντοπίστηκε το κείμενο, το original κείμενο της εικόνας και το μεταφρασμένο κείμενο της εικόνας. Επιπλέον λειτουργίες που μπορούν να πραγματοποιηθούν από αυτό το activity είναι οι εξής: αντιγραφή του original/translated κειμένου, άνοιγμα μιας άλλης εφαρμογής (Instagram, facebook, twitter) στην οποία ο user μπορεί να διαμοιραστεί την εικόνα και τα δεδομένα που εξήγαγε η εφαρμογή. Το αντίστοιχο layout του συγκεκριμένου activity είναι το activity\_details\_of\_card\_history.xml.
* **HistoryActivity.java**: Η κλάση αυτή αφορά το history του user, αλλά είναι βοηθητική. Πιο συγκεκριμένα, η κλάση αυτή περιέχει ένα RecyclerView στο οποίο φορτώνεται ένας adapter, ο οποίος θα παρουσιαστεί παρακάτω, και μέσα σε αυτόν περιέχονται όλα τα records του history. Το layout αυτής της κλάσης είναι το activity\_history.xml.
* **HistoryAdapter.java** και **HistoryViewHolder**: Η κλάση HistoryAdapter είναι ο adapter ο οποίος φορτώνεται στο RecyclerView της κλάσης HistoryActivity. Η HistoryViewHolder είναι nested class της HistoryAdapter και είναι βοηθητική για την παρουσίαση των records του history. Τα attributes της κλάσης αυτής είναι τα controls του layout history\_row.xml. Το συγκεκριμένο layout αφορά τις βασικές πληροφορίες για κάθε record που εμφανίζεται στη λίστα του history. Επομένως, μέσω της HistoryViewHolder, η HistoryAdapter φορτώνει στα controls του history\_row.xml τα δεδομένα που έρχονται από τη βάση και αφορούν το history του user.

### navmenu/detection package

Το συγκεκριμένο package αφορά το text recognition και όλη τη διαδικασία μέχρι το extraction του text από τις εικόνες. Η εισαγωγή εικόνας για το text recognition γίνεται με 3 διαφορετικούς τρόπους: εισαγωγή εικόνας από τη συσκευή, εισαγωγή εικόνας από την κάμερα της συσκευής και τέλος με live caption.

* **CaptureVideoActivity.java**: Η κλάση αυτή αφορά το live caption για το text recognition. Η κλάση κάνει extend την CameraActivity έτσι ώστε να μπορούν να αξιοποιηθούν οι μέθοδοι που είναι απαραίτητες για τη διαδικασία του text recognition. Επιπλέον, η κλάση χρησιμοποιεί το CameraBrigdeViewBase.CvCameraViewListener2, ο οποίος παρέχεται από το OpenCV library και μέσω των μεθόδων που κάνει override βοηθάει στην καταγραφή των frames σε live χρόνο. Η μέθοδος η οποία καταγράφει τα frames είναι η onCameraFrame και επιστρέφει ένα Mat object. Επομένως, ο έλεγχος στα frames για το text recognition γίνεται σε αυτήν. Ωστόσο, ο έλεγχος δεν πραγματοποιείται σε όλα τα frames, αλλά γίνεται κάθε 10 frames. Το Mat object μετατρέπεται σε Bitmap και με αυτό γίνεται ο έλεγχος. Αν ο έλεγχος φέρει αποτέλεσμα τότε αυτομάτως μεταβαίνει σε νέο activity όπου παρουσιάζονται αναλυτικά τα data τα οποία προέκυψαν από την ανάλυση. Το αντίστοιχο layout αυτής της κλάσης είναι το activity\_capture\_video.xml.
* **ImageResultsActivity.java**: Η κλάση αυτή αφορά τα αποτελέσματα του text recognition, και του translation. Από την εισαγωγή της εικόνας (με όποιον από τους 3 τρόπους) γίνεται μετάβαση στο συγκεκριμένο activity, όπου σε πρώτη φάση αναγνωρίζεται το κείμενο της εικόνας και μάλιστα το χωρίζει σε blocks. Στη συνέχεια, για κάθε block εντοπίζεται η γλώσσα του κειμένου και τέλος μεταφράζεται. Επομένως, αυτά που παρουσιάζονται στον user είναι η εικόνα εισαγωγής και για κάθε block το original κείμενο, η γλώσσα στην οποία εντοπίστηκε και το translated κείμενο. Επιπλέον, σε αυτό το activity υπάρχουν μερικές extra δυνατότητες. Αυτές οι δυνατότητες είναι οι εξής: η αποθήκευση της εικόνας και των δεδομένων που εξήγαγε το app στο history του user, το sharing σε άλλα apps (πχ Instagram, facebook και twitter) καθώς και η αντιγραφή του original/translated κειμένου. Το αντίστοιχο layout του συγκεκριμένου activity είναι το activity\_image\_results.xml.
* **MainDetectionFragment.java**: Η κλάση αυτή αφορά το main activity του image detection. Από αυτό ο user μπορεί να επιλέξει έναν από τους τρεις τρόπους για την εισαγωγή εικόνας για το image detection. Κατά τους τρόπους εισαγωγή από τη συσκευή και εισαγωγή από κάμερα, όποια εικόνα επιλεχθεί, εμφανίζεται σε ένα preview και στη συνέχεια ο user μπορεί να προχωρήσει στο text recognition πατώντας το κατάλληλο button. Αν ο τρόπος που επιλέξει ο user είναι το live caption τότε ανοίγει η κάμερα, αλλά αυτή τη φορά ο έλεγχος για το text recognition γίνεται αυτόματα. Το αντίστοιχο layout της συγκεκριμένης κλάσης είναι το fragment\_main\_detection.xml.
* **SavedImage.java**: Η κλάση αυτή αφορά ένα entity για τις εικόνες τις οποίες ο user επιλέγει να αποθηκεύσει στο history. Για το history, στην database αποθηκεύονται objects αυτού του entity. Τα attributes της κλάσης αυτής είναι τα εξής:
  + **storagePath**: το path στο οποίο πρόκειται να αποθηκευτεί η εικόνα στο storage της firebase.
  + **date**: η ημερομηνία κατά την οποία έγινε το image detection και κατ’ επέκταση η αποθήκευση της εικόνας στο history.
  + **sourceBlocks**: μια λίστα από string τα οποία είναι τα original κείμενα που εντοπίστηκαν στα blocks.
  + **translatedBlocks**: μια λίστα από string τα οποία είναι τα μεταφρασμένα κείμενα των blocks.
  + **languageOfBlocks**: μια λίστα από string τα οποία είναι η γλώσσα στην οποία εντοπίστηκε το original κείμενο των blocks.

### navmenu/translation package

Το package αυτό αφορά τον translator που παρέχεται από το app. Πέρα από το image detection για το text recognition, η εφαρμογή προσφέρει έναν translator με τον οποίο μπορεί να γίνει η αναζήτηση της μετάφρασης μιας λέξης από μια διαθέσιμη γλώσσα σε μια άλλη διαθέσιμη γλώσσα.

* **TranslationObject.java**: Η κλάση αυτή αφορά ένα entity σχετικά με την μετάφραση μιας λέξης. Για την αποθήκευση των δεδομένων και – στη συνέχεια – για την εξαγωγή στατιστικών από τα δεδομένα αυτά δημιουργήθηκε η κλάση αυτή. Τα attributes της κλάσης είναι τα εξής:
  + **country**: η χώρα στην οποία βρίσκεται ο user
  + **locality**: η περιοχή της χώρας στην οποία βρίσκεται ο user
  + **month**: ο μήνας κατά τον οποίο έγινε η μετάφραση
  + **year**: ο χρόνος κατά τον οποίο έγινε η μετάφραση
  + **sourceWord**: η λέξη η οποία ζητήθηκε να μεταφραστεί
  + **translatedWord**: η μετάφραση της λέξης
  + **sourceLang**: η γλώσσα από την οποία ζητήθηκε να μεταφραστεί η λέξη
  + **translateLang**: η γλώσσα στην οποία ζητήθηκε να μεταφραστεί η λέξη
* **TranslatorFragment.java**: Η κλάση αυτή αφορά το activity του translator. Ο user εισάγει μια λέξη, προσδιορίζει από ποια γλώσσα προέρχεται, και στη συνέχεια προσδιορίζει και την γλώσσα στην οποία θέλει να μεταφραστεί η λέξη. Μόλις συμπληρώσει όλα αυτά τα πεδία πατάει το κουμπί της μετάφρασης. Η κλάση αυτή παίρνει την λέξη που έδωσε ο user και προσπαθεί να την μεταφράσει. Αν όλα πάνε καλά, το αποτέλεσμα θα εμφανιστεί στον user και ταυτόχρονα θα αποθηκευτεί στην database ένα object της κλάσης TranslationObject με όλες τις πληροφορίες που χρειάζονται για την μετέπειτα εξαγωγή στατιστικών δεδομένων. Το αντίστοιχο layout της συγκεκριμένης κλάσης είναι το fragment\_translator.xml.

## ΧΡΗΣΗ ΤΟΥ ML KIT ΤΗΣ GOOGLE-FIREBASE

Για πολλές από τις λειτουργίες τις οποίες προσφέρει το συγκεκριμένο app, έχει χρησιμοποιηθεί το Machine Learning Kit της Google. Ενώ το προαναφερθέν kit προσφέρει πάρα πολλές δυνατότητες, για την υλοποίηση των λειτουργιών χρησιμοποιήθηκαν τα εξής tools:

* **Text recognition**: Το συγκεκριμένο tool είναι αυτό που αναγνωρίζει το κείμενο σε μια εικόνα, αν υπάρχει. Με την βοήθεια ενός TextRecognizer, σε κάθε εικόνα, ανιχνεύονται τα blocks κειμένων, όταν αυτά υπάρχουν. Τα κείμενα των blocks αποθηκεύονται σε ένα List ώστε στη συνέχεια να γίνει η μετάφραση αυτών. Το dependency που χρειάζεται το application για να μπορεί να χρησιμοποιήσει αυτό το tool είναι το ακόλουθο

« implementation 'com.google.android.gms:play-services-mlkit-text-recognition:18.0.0'».

Πρέπει να σημειωθεί ότι στις υποστηριζόμενες γλώσσες του text recognition δεν συμπεριλαμβάνονται όλες. Σε αυτές που δεν συμπεριλαμβάνονται είναι και η ελληνική γλώσσα. Ωστόσο, στο link παρακάτω παρουσιάζονται εκείνες οι γλώσσες που υποστηρίζονται και οι οποίες μπορούν να αναγνωριστούν στις εικόνες.

Text recognition - supported languages:

* <https://developers.google.com/ml-kit/vision/text-recognition/languages>

* **Language Identification**: Το συγκεκριμένο tool είναι αυτό που βοηθάει στην αναγνώριση της γλώσσας στην οποία εντοπίζεται ένα κείμενο ή μια λέξη. Με την βοήθεια ενός LanguageIdentifier γίνεται η ταυτοποίηση με την πιο πιθανή γλώσσα προέλευσης του κειμένου. Για κάθε κείμενο καταγράφεται σε ένα List η γλώσσα από την οποία είναι πιο πιθανό να προέρχεται το κείμενο αυτό. Όπως αναφέρθηκε νωρίτερα, δεν υποστηρίζονται όλες οι γλώσσες, άρα η ταυτοποίηση γίνεται μόνο σε αυτές που περιέχονται στο link <https://developers.google.com/ml-kit/vision/text-recognition/languages>. Αν και η γλώσσα που επιστρέφεται είναι σε μορφή tag/code, στο list αποθηκεύεται όλο το display name της. Το dependency για αυτό το tool είναι το ακόλουθο

« implementation 'com.google.mlkit:language-id:17.0.3' »

* **Translate text**: Το συγκεκριμένο tool είναι αυτό που βοηθάει στην μετάφραση του κειμένου, όταν έχουν γίνει με επιτυχία το text recognition και το language identification. Με την βοήθεια ενός Translator, για κάθε κείμενο που βρέθηκε από το TextRecognizer, γίνεται η μετάφρασή του, από την γλώσσα που αναγνωρίστηκε με τον LanguageIdentifier, στην ελληνική γλώσσα. Για κάθε κείμενο που είναι αποθηκευμένο στο υπάρχον List, η μετάφραση που προκύπτει από τον Translator αποθηκεύεται σε ένα άλλο List. Το dependency για να μπορεί να αξιοποιηθεί αυτό το tool είναι το ακόλουθο

« implementation 'com.google.mlkit:translate:17.0.0' »

Αξίζει να σημειωθεί ότι η χρήση του Machine Learning Kit της Google έχει αλλάξει το τελευταίο διάστημα. Μέχρι πριν κάποιο διάστημα, το text recognition API ήταν όλο μέρος του ML Kit της Firebase. Πλέον έχει χωριστεί σε 2 ξεχωριστά APIs. Το ένα, το «on-device text recognition», είναι μέρος του standalone ML Kit SDK, το οποίο μπορεί να χρησιμοποιηθεί και ανεξάρτητα από την firebase. Το δεύτερο, το «Cloud text recognition», είναι μέρος του ML της firebase. Υπάρχουν αρκετές διαφορές στα 2 αυτά νέα features και μπορεί κανείς να βρει τις βασικές διαφορές στον ακόλουθο σύνδεσμο <https://firebase.google.com/docs/ml-kit/recognize-text>.

## IMPORT OPENCV LIBRARY

Κύριος σκοπός του app είναι το text recognition. Για το text detection αρκεί να δοθεί μια εικόνα από τον user και στη συνέχεια, με την βοήθεια του ML Kit που παρουσιάστηκε σε προηγούμενη ενότητα, προκύπτουν τα αποτελέσματα.

Για την εισαγωγή της εικόνας στην οποία θα γίνει το detection, το app προσφέρει τρεις δυνατούς τρόπους. Ο πρώτος τρόπος είναι να γίνει η εισαγωγή της εικόνας από τα αρχεία της συσκευής. Επομένως, όταν επιλεχθεί αυτός ο τρόπος, ο user μπορεί να ψάξει στα αρχεία που έχει αποθηκευμένα στη συσκευή του και να κάνει import την εικόνα που θέλει. Ο δεύτερος τρόπος για να κάνει import μια εικόνα είναι να τραβήξει επιτόπου μια φωτογραφία με τη χρήση της κάμερας που διαθέτει η συσκευή, να αποθηκευτεί η φωτογραφία στα αρχεία της συσκευής, και τέλος να γίνει το detection πάνω σε αυτήν. Οι δύο τρόποι που προαναφέρθηκαν έχουν ένα βασικό μειονέκτημα. Απαιτούν και οι δύο την δέσμευση του χώρου της συσκευής. Αυτό είναι σημαντικό πρόβλημα και όταν συμβαίνει κάτι τέτοιο από τις εφαρμογές δυσανασχετεί τους users.

Η λύση στο πρόβλημα “ήρθε” άμεσα. Από τη στιγμή που δεν είναι επιθυμητή η αποθήκευση των εικόνων στην συσκευή αρκεί να γίνεται ο έλεγχος για text recognition σε live χρόνο. Με άλλα λόγια, επιθυμητό είναι να μπορεί ο user να ανοίξει την κάμερα της συσκευής του και χωρίς να αποθηκεύσει κάποια εικόνα, να μπορεί να γίνει ο έλεγχος στα frames σε live χρόνο.

Μετά από αναζήτηση, φάνηκε ότι αυτό θα μπορούσε να υλοποιηθεί χρησιμοποιώντας το OpenCV Library. Το συγκεκριμένο library είναι open source computer vision and machine learning software library και προσφέρει πολλούς αλγορίθμους για διάφορα θέματα. Περισσότερες πληροφορίες για αυτό το library παρουσιάζονται στο ακόλουθο link <https://opencv.org/about/>. Ωστόσο, στο συγκεκριμένο project η βοήθεια που παρέχει το OpenCV Library αφορά την χρήση της κάμερας και πιο συγκεκριμένα τις λειτουργίες που χρειάζεται να γίνουν στην καταγραφή των frames.

Στην υλοποίηση του live caption, η κλάση η οποία είναι υπεύθυνη για αυτή τη λειτουργία κάνει extend την CameraActivity, από την οποία κληρονομεί εκείνα που θα δώσουν τη λύση στο πρόβλημα. Με τη βοήθεια ενός CameraBridgeViewBase, το οποίο αφορά ένα JavaCameraView που υπάρχει στο layout, και των βασικών listeners BaseLoaderCallBack και CameraBridgeViewBase.CvCameraViewListener2 επιτυγχάνεται το live caption.

Ο CameraBridgeViewBase.CvCameraViewListener2 listener κάνει override τη βασική μέθοδο που δίνει τα frames, τα οποία λαμβάνει από την κάμερα. Το κάθε frame επιστρέφεται ως Mat object, το οποίο δεν είναι άλλο παρά ένα matrix που αναπαριστά την εικόνα. Αν και με αυτόν τον τρόπο λαμβάνονται όλα τα frames, στην συγκεκριμένη διαδικασία δεν χρειάζεται να γίνεται έλεγχος για text recognition σε όλα. Έχει επιλεχθεί να γίνεται ο έλεγχος κάθε 10 frames. Άρα, κάθε 10 frames το Mat object που επιστρέφεται μετατρέπεται σε Bitmap και με βάση αυτό γίνεται ο έλεγχος. Ξανά, με τον TextRecognizer, ο οποίος αναλύθηκε πιο πάνω σε άλλη ενότητα, γίνεται έλεγχος για text recognition κι αν υπάρχει αποτέλεσμα από αυτόν τότε γίνεται όλη η διαδικασία για τον διαχωρισμό του κειμένου σε blocks και για την μετάφραση.

Το version του OpenCV Library που χρησιμοποιήθηκε στο project της εφαρμογής αυτής είναι η 4.6.0 (<https://opencv.org/releases/>). Μετά την ολοκλήρωση του installation του OpenCV Library, μέσα από το IDE έγινε και το import και πλέον μπορεί να αξιοποιηθεί.

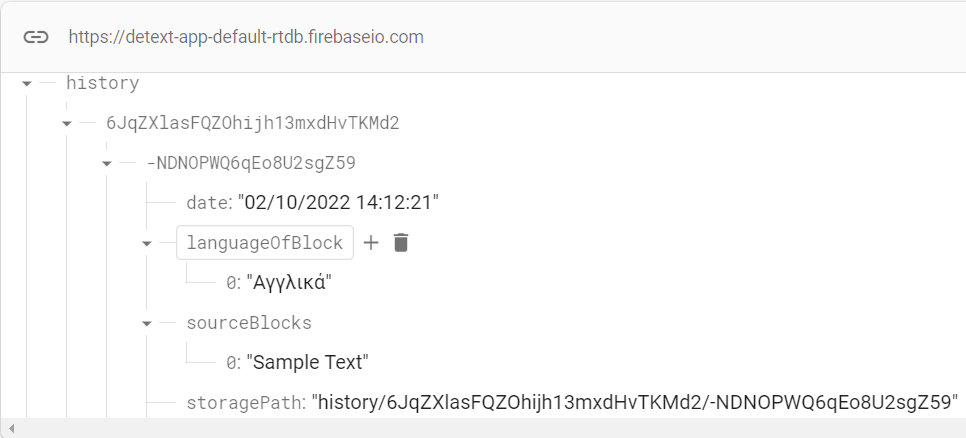
## ΣΥΝΔΕΣΗ ΜΕ ΒΑΣΗ – REALTIME DATABASE

Είναι απαραίτητη η χρήση μιας database για την αποθήκευση ορισμένων δεδομένων που χρησιμοποιεί η εφαρμογή. Η database που επιλέχθηκε να χρησιμοποιηθεί στην προκειμένη περίπτωση είναι η Realtime Database της Firebase. Στην database έχουν πρόσβαση μόνο οι authenticated users του app. Επομένως, έχουν προσαρμοστεί κατάλληλα τα rules της database.

Από το root της database υπάρχουν τρία αρχικά children τα οποία αφορούν κάτι διαφορετικό. Τα children αυτά είναι τα εξής και έχουν την ακόλουθη μορφή:

* history 🡪 node\_with\_userId\_as\_key 🡪 node\_with\_random\_key όπου στο τελευταίο είναι αποθηκευμένο ένα object της κλάσης SavedImage. Αυτή η δομή δηλώνει ότι “κάτω” από τον κόμβο history υπάρχει ένας άλλος κόμβος για κάθε user. Στη συνέχεια, “κάτω” από τον κόμβο που αναφέρεται σε κάποιον user υπάρχουν άλλοι κόμβοι οι οποίοι αναφέρονται στα records του history που έχει επιλέξει να αποθηκεύσει. Δηλαδή, για κάθε text recognition που ο user αποφασίζει να αποθηκεύσει, δημιουργείται ένα object της κλάσης SavedImage και αποθηκεύεται στην database “κάτω” από τον κόμβο που αφορά τον ίδιο τον user.

Ακολουθεί ένα παράδειγμα της δομής της database.



Saved “SavedImage” object

Random generated reference key

userId

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

* stats 🡪 fromTranslation 🡪 node\_with\_random\_key όπου στο τελευταίο είναι αποθηκευμένο ένα object της κλάσης TranslationObject. Αυτή η δομή δηλώνει ότι “κάτω” από τον κόμβο stats/fromTranslation αποθηκεύονται όλα τα objects που αφορούν τις μεταφράσεις των λέξεων που εισαγάγουν οι users. Για κάθε μετάφραση που πραγματοποιείται χρειάζεται να καταγράφονται μερικά σχετικά δεδομένα για την – μετέπειτα – εξαγωγή στατιστικών. Τα βασικά που πρέπει να αποθηκεύονται είναι η λέξη την οποία ο user εισάγει για μετάφραση, και η γλώσσα στην οποία ανήκει, η μετάφραση της λέξης αυτής, καθώς και η γλώσσα στην οποία μεταφράστηκε. Τα δευτερεύοντα, που όμως αξίζει να καταγραφούν, αφορούν την τοποθεσία (country & locality) και την χρονολογία μετάφρασης (month & year). Στη συνέχεια, σε άλλη ενότητα, θα εξηγηθεί ο λόγος που αποθηκεύονται τα συγκεκριμένα δεδομένα για την εξαγωγή στατιστικών.

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

Saved “TranslationObject”

Random generated key

* users 🡪 node\_with\_userId\_as\_key 🡪 node\_with\_user\_email\_as\_key όπου στο τελευταίο είναι αποθηκευμένο το email ενός user. Κάτω από το node\_with\_userId\_as\_key αποθηκεύεται ένα object της κλάσης User, το οποίο περιέχει την βασική πληροφορία για αυτόν, το email.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

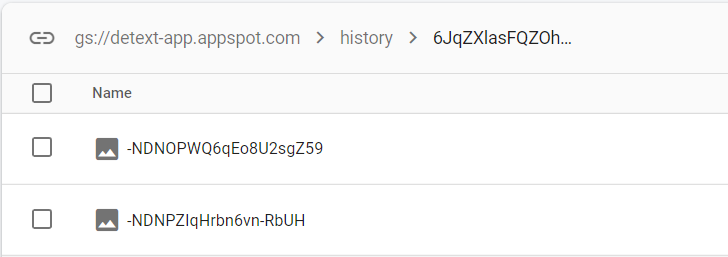
userId

## STORAGE

Το app χρησιμοποιεί το storage της firebase για την αποθήκευση εικόνων. Οι εικόνες που χρειάζεται να αποθηκευτούν είναι αυτές που εισάγουν οι users ώστε να γίνει το text recognition. Μόλις ολοκληρωθεί το image detection και το text recognition, και δεδομένου ότι υπάρχει αποτέλεσμα από τον τελευταίο έλεγχο, ο user έχει την επιλογή να αποθηκεύσει το αποτέλεσμα στο δικό του history. Επομένως, ήταν αναγκαίο να αποθηκεύεται και η εικόνα σε κάποιο μέσο για να μπορεί να ανακτηθεί από τον user όταν εκείνος επιθυμεί. Για την αποθήκευση της εικόνας, η εικόνα μετατρέπεται σε bitmap, γίνεται compressed και αυτό με τη σειρά του μετατρέπεται σε ένα array από bytes. Με αυτή τη διαδικασία το upload της εικόνας γίνεται πιο εύκολα.

Το root του storage είναι το «history» και από αυτό προκύπτουν διαφορετικά paths από τις εισαγωγές που γίνονται. Μετά από το «history» υπάρχει ένα node για κάθε user (με όνομα το userId) και αυτό δημιουργείται την πρώτη φορά που ο user κάνει save το result της εικόνας. Οπότε για κάθε νέο insert μιας εικόνας δημιουργείται ένα νέο path μετά το userId του user. Το όνομα που παίρνει το νέο node που δημιουργείται δεν είναι τυχαίο. Το node παίρνει το όνομα από το key του database reference, το οποίο χρησιμοποιείται για το insert του history στην realtime database. Αυτό γίνεται για να υπάρχει το σωστό relation μεταξύ realtime database και storage. Με άλλα λόγια, αυτό εξυπηρετεί στην άμεση σύνδεση ενός record που βρίσκεται στην realtime database και στην αντίστοιχη εικόνα που έχει αποθηκευτεί στο storage.

Ένα παράδειγμα της δομής του storage παρουσιάζεται παρακάτω.



The generated key referred to a history record

userId

## USERS’ AUTHENTICATION

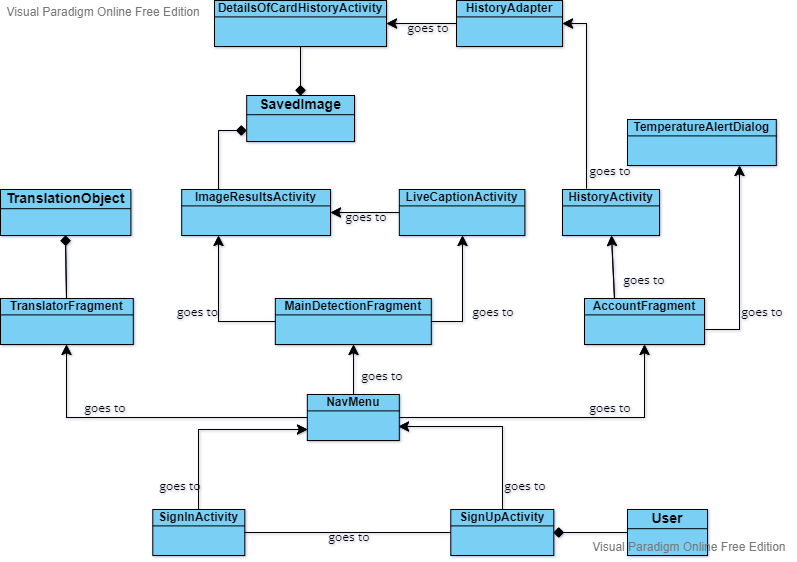
Το authentication των users γίνεται μέσω του Authentication που παρέχει η Firebase. Το registration των users γίνεται μέσω email/password. Το συγκεκριμένο app δεν απαιτεί κάτι περισσότερο από τους users και τα μόνα που χρειάζονται είναι αυτά τα δύο που αναφέρθηκαν νωρίτερα για να κάνουν login.

## ΥΠΟΣΤΗΡΙΖΟΜΕΝΕΣ ΓΛΩΣΣΕΣ

Το app, αν και αναφέρεται κυρίως σε Έλληνες τουρίστες, υποστηρίζει 2 γλώσσες. Η μία γλώσσα είναι η ελληνική και η άλλη η αγγλική. Η γλώσσα επιλέγεται αυτόματα από το σύστημα της ίδιας της συσκευής. Στον φάκελο res/values/strings υπάρχουν 2 .xml αρχεία τα οποία αφορούν τις δύο υποστηριζόμενες γλώσσες της εφαρμογής. Τα δύο αυτά αρχεία έχουν για κάθε string του app την αγγλική και την ελληνική γραπτή αναπαράσταση.

## CLASS DIAGRAM

Το class diagram της εφαρμογής παρουσιάζεται στην παρακάτω εικόνα.



*Class Diagram*

# API ΓΙΑ ΕΞΑΓΩΓΗ ΣΤΑΤΙΣΤΙΚΩΝ ΔΕΔΟΜΕΝΩΝ

## ΠΑΡΟΥΣΙΑΣΗ ΙΔΕΑΣ

Από την χρήση της εφαρμογής προκύπτουν πολλά δεδομένα από τα οποία μπορούν να βγουν πολλά συμπεράσματα. Για παράδειγμα, μπορεί να γίνει αναζήτηση στα δεδομένα που έχουν καταγραφεί σχετικά με την λέξη που αναζητήθηκε περισσότερο έναν συγκεκριμένο μήνα ή σε μια συγκεκριμένη περιοχή. Γενικά μπορούν να γίνουν αναζητήσεις με διαφορετικά φίλτρα κάθε φορά από τις οποίες προκύπτουν ενδιαφέροντα συμπεράσματα για τον τουρισμό.

Μια τέτοια εφαρμογή θα μπορούσε να βοηθήσει τον κλάδο του τουρισμού δεδομένου ότι γνωρίζουν τί έχουν αναζητήσει οι τουρίστες. Τα δεδομένα αυτά είναι πάντα ανώνυμα εφόσον σε καμία περίπτωση δεν χρειάζονται προσωπικά στοιχεία των users. Τα μόνο που χρειάζονται για τα στατιστικά είναι οι πληροφορίες σχετικά με τις αναζητήσεις των users.

Με αυτόν τον τρόπο θα μπορούν οι άνθρωποι που εργάζονται στον κλάδο του τουρισμού να γνωρίζουν ποια είναι εκείνα που ενδιαφέρουν τους τουρίστες μια συγκεκριμένη χρονική περίοδο. Γνωρίζοντας τις προτιμήσεις των τουριστών, οι καταστηματάρχες/έμποροι μπορούν να προσφέρουν όσα οι πρώτοι επιθυμούν. Κατ’ επέκταση με αυτόν τον τρόπο μπορεί να ενισχυθεί και η οικονομία της περιοχής.

Γενικά μια τέτοια εφαρμογή εξαγωγής στατιστικών δεδομένων μοιάζει χρήσιμη σε διάφορους τομείς και έχει πολλά οφέλη. Όπως αναφέρθηκε και νωρίτερα μπορεί να ενισχύσει μέχρι και την οικονομία μιας περιοχής.

## REQUESTS ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ

Η εξαγωγή των στατιστικών δεδομένων από τις καταγραφές των αναζητήσεων των users επιλέχθηκε να γίνει μέσω ενός API (Application Programming Interface). Ένας από τους λόγους που επιλέχθηκε να δημιουργηθεί API είναι πως τα APIs έχουν σημαντική θέση στις περισσότερες εφαρμογές και είναι ανεξάρτητα από το frontend. Ωστόσο μπορούν εύκολα να συνδεθούν με κάποιο UI. Επιπλέον, χρησιμοποιώντας αυτήν την τεχνολογία, αλλά και αυτές με τις οποίες μπορεί να “χτιστεί” ένα REST API, είναι πιο εύκολο να γίνουν οι αναζητήσεις στην προκειμένη περίπτωση.

Δεδομένου ότι χτίζεται ένα API, τα αιτήματα για τις αναζητήσεις που θα εκτελεί η εφαρμογή δεν θα είναι άλλα από μερικά απλά http requests μέσω localhost. Τα αποτελέσματα μπορεί κάποιος να τα δει είτε μέσα από έναν browser, είτε μέσα από ένα tool, όπως το POSTMAN ή ακόμα και από command line μέσω curl commands. Στη συνέχεια θα παρουσιαστούν μερικά παραδείγματα από τα αποτελέσματα και από τα requests που μπορούν να γίνουν. Σε κάθε περίπτωση τα αποτελέσματα είναι σε μορφή json οπότε μπορούν να παρουσιαστούν με τον πιο απλό τρόπο. Το επίσης σημαντικό με αυτό είναι πως είναι εύκολο από την μορφή στην οποία επιστρέφονται να ενσωματωθούν στο frontend.

## ΥΛΟΠΟΙΗΣΗ API

### ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ PROJECT

Για την υλοποίηση του REST API χρησιμοποιήθηκαν νέες τεχνολογίες και έγιναν διάφοροι συνδυασμοί. Επιλέχθηκε να χρησιμοποιηθεί το Spring Boot framework για την δημιουργία του API μιας και προσφέρει πολλές δυνατότητες.

Αρχικά, με την βοήθεια του Spring Initializr (<https://start.spring.io/>) δημιουργήθηκε το maven project το οποίο συνδυάζεται με το Spring Boot Framework. Τα κύρια χαρακτηριστικά του project είναι τα εξής:

* **Maven project**: αυτό σημαίνει ότι όλα τα dependencies βρίσκονται στο παραγόμενο – από το spring initializr – αρχείο pom.xml
* **Java** **11** η γλώσσα στην οποία γράφεται το project
* **Spring boot 2.7.4** το version της spring boot
* **Dependencies** που καθορίστηκαν εξ αρχής είναι το **Spring Web** το οποίο βοηθάει να γίνει η εφαρμογή web
* **Packaging** ως **Jar**

### ΣΥΝΔΕΣΗ ΜΕ FIREBASE

Σε αυτό το – δεύτερο – κομμάτι της εργασίας, πρέπει να γίνει η σύνδεση της Realtime Database με το Spring Boot Framework που χρησιμοποιείται για την υλοποίηση του API. Επομένως, για την σύνδεση αυτή απαιτεί ένα συγκεκριμένο dependency. Το dependency αυτό προστίθενται στο pom.xml file ( προέρχεται από το ακόλουθο link: [*https://mvnrepository.com/artifact/com.google.firebase/firebase-admin*](https://mvnrepository.com/artifact/com.google.firebase/firebase-admin)) και είναι το εξής:  
***<dependency>  
 <groupId>com.google.firebase</groupId>  
 <artifactId>firebase-admin</artifactId>  
 <version>9.0.0</version>  
</dependency>***

Το σημαντικό με το Framework που χρησιμοποιείται είναι πως με ένα απλό insert στο pom.xml file γίνεται όλη η σύνδεση με το database και δεν χρειάζεται καν εγκατάσταση server για να τρέξει. Όλα γίνονται πιο εύκολα και απομακρυσμένα.

Χρησιμοποιώντας το παραπάνω dependency – σε μετέπειτα χρόνο – μπορούν να γίνουν κι άλλες λειτουργίες που απαιτούν σύνδεση με firebase και όχι μόνο με την Realtime Database, αλλά και με άλλα tools που προσφέρει η Firebase. Ωστόσο, αν είναι επιθυμητή μόνο η σύνδεση με την Realtime Database της Firebase, το αντίστοιχο dependency είναι το ακόλουθο:

*<!-- https://mvnrepository.com/artifact/com.google.firebase/firebase-database -->****<dependency>  
 <groupId>com.google.firebase</groupId>  
 <artifactId>firebase-database</artifactId>  
 <version>11.0.1</version>  
</dependency>***

### ΚΩΔΙΚΑΣ

Το API το οποίο αναπτύχθηκε, υλοποιεί μόνο τα endpoints που κρίνονται απαραίτητα για τις λειτουργίες που προσφέρει, δηλαδή για τις αναζητήσεις. Οι λειτουργίες αυτές δεν είναι πολλές, αλλά είναι οι βασικές. Στις λειτουργίες του συγκεκριμένου API δεν είναι η εισαγωγή ή η διαγραφή των data. Οι λειτουργίες αφορούν μόνο επιλογή δεδομένων από την database. Επομένως, το API κάνει implement μόνο get requests, τα οποία όμως αφορούν κάτι διαφορετικό κάθε φορά. Τα get requests αναφέρονται στα εξής:

* Προβολή όλων των εγγραφών ως στατιστικά δεδομένα που έχουν αποθηκευτεί στην Realtime Database. Στην περίπτωση αυτή το API φέρνει κυριολεκτικά όλες τις αναζητήσεις που έχουν καταγραφεί. Το αποτέλεσμα για κάθε εγγραφή είναι ένα TranslationObject οπότε εμφανίζονται όλες οι πληροφορίες για αυτήν.
* Προβολή όλων των διακριτών λέξεων που έχουν αναζητηθεί. Από όλες τις εγγραφές που υπάρχουν στην database βρίσκονται οι διακριτές λέξεις και το αποτέλεσμα είναι αυτές οι διακριτές λέξεις και πόσες φορές αναζητήθηκε η καθεμία.
* Προβολή εγγραφών βάσει φίλτρων. Η αναζήτηση σε αυτήν την περίπτωση γίνεται με φίλτρα. Με άλλα λόγια μπορούν να χρησιμοποιηθούν φίλτρα για το πεδίο με το οποίο μπορεί να γίνει η αναζήτηση. Για παράδειγμα, η αναζήτηση μπορεί να γίνει βάσει του πεδίου country ή year κ.ο.κ. Αυτό αναφέρεται ως key για την αναζήτηση. Ωστόσο, πρέπει να δοθεί και το value με το οποίο πρέπει να κάνει match. Επομένως, μια αναζήτηση μπορεί να αφορά τις αναζητήσεις που έγιναν τον μήνα Σεπτέμβριο. Στο παράδειγμα αυτό το “month” είναι το key και το “ΣΕΠΤΕΜΒΡΙΟΣ” είναι το value. Οι εγγραφές που θα επιστραφούν θα αφορούν αναζητήσεις που έγιναν τον μήνα Σεπτέμβριο. Πρέπει να σημειωθεί εδώ, όμως, ότι στο αποτέλεσμα αναφέρονται και τα φίλτρα και το μέγεθος του αποτελέσματος και φυσικά τα ίδια τα αποτελέσματα, τα οποία είναι οι εγγραφές από την database.
* Προβολή της πιο searched λέξης ανά χώρα. Για κάθε χώρα, στην οποία έχει γίνει έστω και μία αναζήτηση, βρίσκεται η λέξη που έχει αναζητηθεί τις περισσότερες φορές. Το αποτέλεσμα είναι για κάθε χώρα η λέξη που βρέθηκε και πόσες φορές έχει αναζητηθεί.
* Προβολή μιας συγκεκριμένης λέξης. Για μια συγκεκριμένη λέξη εμφανίζεται ο αριθμός αναζητήσεων αυτής χωρίς φίλτρα.
* Προβολή της πιο searched λέξης από όλες τις εγγραφές που υπάρχουν στην database και πόσες φορές έχει αναζητηθεί. Δεν αφορά συγκεκριμένη χρονική περίοδο ή συγκεκριμένο τόπο, αλλά αφορά όλες τις αναζητήσεις που έχουν καταγραφεί. Το αποτέλεσμα είναι η λέξη που έχει αναζητηθεί τις περισσότερες φορές και πόσες είναι αυτές οι φορές που αναζητήθηκε.

Το application ξεκινάει από την κλάση DetextapiApplication, η οποία είναι annotated με το annotation @SpringBootApplication του Spring Boot Framework. Από αυτό ξεκινάνε όλα τα configurations του application.

#### PACKAGES

Ο κώδικας της εφαρμογής αυτής, η οποία αφορά το δεύτερο σκέλος της ολικής εργασίας, είναι και πάλι χωρισμένος σε packages για την καλύτερη οργάνωση, ανάγνωση και συντήρηση.

Η κλάση που κάνει το configuration και τρέχει όλο το application είναι η DetextapiApplication.java.

* **controller package**: το συγκεκριμένο package αφορά τον controller που έχει δημιουργηθεί για το project και δεν είναι άλλος από την κλάση TranslationController.

Η κλάση με όνομα TranslationController.java, είναι annotated με @RestController και αποτελεί τον Controller της web εφαρμογής. Το annotation αυτό προέρχεται από το Spring Boot Framework και δηλώνει ότι η κλάση - controller είναι υπεύθυνη για την λήψη των http requests και για την επιστροφή των αποτελεσμάτων από την επεξεργασία που γίνεται στο service layer για καθένα από αυτά τα requests. Ένας controller θα μπορούσε να είναι annotated απλά με @Controller, αλλά κάθε method που θα γινόταν implemented θα έπρεπε να είναι annotated και με @ResponseBody. Επομένως, μια και καλή γίνεται annotated η κλάση με @RestController. Επίσης, η κλάση είναι annotated και με @RequestMapping(“/detext/translations”). Αυτό σημαίνει ότι το mapping για τα requests θα γίνεται βάσει του συγκεκριμένου path «<http://localhost:8080/detext/translations>».

Οι μέθοδοι του controller αντιστοιχούν σε κάποιο endpoint, δηλαδή σε κάποιο http request και το mapping αυτών γίνεται με το @GetMapping. Το annotation αυτό σημαίνει ότι το method του request είναι GET το οποίο αντιστοιχεί σε “select” στην SQL γλώσσα. Σε αυτό το annotation δίνεται ένα extra path σε συνέχεια του προηγούμενου που ορίζεται στο @RequestMapping. Με αυτόν τον τρόπο αναγνωρίζεται ποια μέθοδος πρέπει να εκτελεστεί σε κάθε request. Όλες οι μέθοδοι επιστρέφουν ResponseEntity<T> και αυτό πάλι προέρχεται από το Spring Boot Framework.

Ο controller έχει ένα TranslationService attribute, το οποίο αρχικοποιείται με constructor injection. Αυτό είναι ένα object του Service της εφαρμογής (το Service θα αναλυθεί στη συνέχεια). Με την βοήθεια αυτού του Service γίνεται όλη η διαδικασία που φέρει αποτελέσματα στα requests.

* **entity package**: το συγκεκριμένο package αφορά τα entities και στην προκειμένη περίπτωση μόνο ένα, το TranslationObject. Τα objects που διαβάζονται από την Realtime Database της Firebase γίνονται serialized σε objects αυτού του entity.

Η κλάση με όνομα TranslationObject.java είναι το entity, όπως αναφέρθηκε και πιο πάνω, και είναι ακριβώς ίδια με την κλάση TranslationObject του android app που παρουσιάστηκε στο κομμάτι που αφορά το πρώτο μέρος της εργασίας. Τα attributes αυτής της κλάσης είναι τα εξής:

* + **country**: η χώρα στην οποία βρίσκεται ο user
  + **locality**: η περιοχή της χώρας στην οποία βρίσκεται ο user
  + **month**: ο μήνας κατά τον οποίο έγινε η μετάφραση
  + **year**: ο χρόνος κατά τον οποίο έγινε η μετάφραση
  + **sourceWord**: η λέξη η οποία ζητήθηκε να μεταφραστεί
  + **translatedWord**: η μετάφραση της λέξης
  + **sourceLang**: η γλώσσα από την οποία ζητήθηκε να μεταφραστεί η λέξη
  + **translateLang**: η γλώσσα στην οποία ζητήθηκε να μεταφραστεί η λέξη
* **service package**: το συγκεκριμένο package αφορά το service που έχει δημιουργηθεί και το οποίο κάνει όλο το business logic. Το service αυτό δεν είναι άλλο από την κλάση TranslationService. Η κλάση αυτή είναι annotated με @Service. Το @Service είναι ειδική κατηγορία του @Component annotation, προέρχεται από το Spring Boot Framework και δηλώνει πως η κλάση αυτή είναι ένας Service Provider.

Οι μέθοδοι του Service καλούνται από τον Controller και σε αυτές γίνεται η επικοινωνία με την Realtime Database. Για κάθε request που λαμβάνει ο Controller καλείται το Service και η αντίστοιχη μέθοδος που υλοποιεί την λειτουργία. Η σύνδεση με την Realtime Database επιτυγχάνεται με ένα DatabaseReference.

Μια άλλη κλάση που περιέχεται στο package αυτό είναι ένα Enum, το FindBy και αφορά την αναζήτηση με φίλτρα. Περιέχει τα τέσσερα keys με τα οποία μπορεί να γίνει μια αναζήτηση. Τα keys αυτά είναι τα εξής: COUNTRY, LOCALITY, MONTH, YEAR.

# ΠΑΡΑΓΕΙΓΜΑΤΑ ΕΚΤΕΛΕΣΗΣ APP

Edw…

# ΠΑΡΑΔΕΙΓΜΑΤΑ ΕΚΤΕΛΕΣΗΣ API

Μερικά από τα αποτελέσματα του API παρουσιάζονται στις εικόνες παρακάτω. Τα παραδείγματα αφορούν αποτελέσματα μέσα από το POSTMAN, αλλά και από browser.

* Παραδείγματα από POSTMAN

Στο πρώτο παράδειγμα, γίνεται αναζήτηση όλων των αναζητήσεων που έχουν καταγραφεί στην database. Στέλνοντας ένα get request με το url <http://localhost:8080/detext/translations> καλείται το αντίστοιχο method του controller, το οποίο με τη σειρά του καλεί το service για να κάνει την αναζήτηση. Το αποτέλεσμα είναι σε μορφή json και περιέχει τις εξής πληροφορίες:

* + Το μέγεθος του αποτελέσματος, δηλαδή πόσες εγγραφές βρέθηκαν
  + Τις εγγραφές οι οποίες βρέθηκαν από την αναζήτηση

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

<http://localhost:8080/detext/translations>

*Get all records from database*

Το δεύτερο παράδειγμα αφορά την αναζήτηση των διακριτών λέξεων που έχουν αναζητηθεί και έχουν καταγραφεί στην database. Το αποτέλεσμα είναι σε μορφή json και είναι μια λίστα από τις διακριτές λέξεις που βρέθηκαν μαζί με τον αριθμό των αναζητήσεων για κάθε λέξη από αυτές.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Το ResponseBody που επιστρέφεται είναι σε μορφή json.

Το αποτέλεσμα είναι μια λίστα από όλες τις διακριτές λέξεις που έχουν αναζητηθεί κατά καιρούς και τόπους μαζί με τις φορές που έχει αναζητηθεί η καθεμία.

<http://localhost:8080/detext/translations/distinct>

*Get all distinct words that have been searched*

Στο παράδειγμα παρακάτω φαίνεται μια αναζήτηση με filters. Ουσιαστικά το filtering γίνεται από τα params τα οποία στέλνονται μέσω του request. Στο συγκεκριμένο παράδειγμα η αναζήτηση είναι επιθυμητό να γίνει βάσει του μήνα και συγκεκριμένα του μήνα Οκτώβρη. Επομένως δίνεται πρώτα το key για την αναζήτηση, δηλαδή το “month”, και στην συνέχεια δίνεται το value για την αναζήτηση, δηλαδή το “OCTOBER”. Το αποτέλεσμα είναι σε μορφή json και περιέχει τις εξής πληροφορίες:

* + Το key με το οποίο έγινε η αναζήτηση (month/year/country/locality), όπου στην προκειμένη περίπτωση είναι το “month”
  + Το value με το οποίο έγινε το mapping στην αναζήτηση, όπου στην προκειμένη περίπτωση είναι το “OCTOBER”
  + Το μέγεθος του αποτελέσματος σχετικά με το πλήθος των εγγραφών που βρέθηκαν με αυτά τα filters. Στο παράδειγμα αυτό βρέθηκαν 24 εγγραφές που αφορούν στον μήνα Οκτώβρη
  + Τα records που προέκυψαν από την αναζήτηση με το filtering

Σημείωση: Το url για αυτήν την αναζήτηση είναι το <http://localhost:8080/detext/translations/params>, ωστόσο μετά από αυτό, όπως φαίνεται και στην εικόνα πιο κάτω, προστίθενται και οι παράμετροι που εισάγονται στο Params του POSTMAN. Αυτά αλλάζουν στο url ανάλογα με τα params που δίνονται από κάτω.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

<http://localhost:8080/detext/translations/params>

*Get filtered records*

Το επόμενο παράδειγμα αφορά μια αναζήτηση για την πιο searched λέξη ανά χώρα. Το αποτέλεσμα είναι σε μορφή json και περιέχει για κάθε χώρα, στην οποία έχει καταγραφεί έστω και μία αναζήτηση, την πιο searched λέξη που βρέθηκε.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

<http://localhost:8080/detext/translations/most-searched-word-per-country>

*Get the most searched word per country*

Μια άλλη αναζήτηση αφορά συγκεκριμένη λέξη. Με άλλα λόγια, σε αυτήν την περίπτωση γίνεται αναζήτηση βάσει συγκεκριμένης λέξης. Με get request και το url ως [http://localhost:8080/detext/translations/{word}](http://localhost:8080/detext/translations/%7bword%7d), όπου {word} η λέξη με την οποία είναι επιθυμητή η αναζήτηση, επιστρέφεται – αν υπάρχει – η λέξη αυτή και το πλήθος των αναζητήσεων για αυτήν. Στα συγκεκριμένα παραδείγματα πιο κάτω φαίνονται δύο αναζητήσεις για τις λέξεις “αεροδρόμιο” και “face”.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

*Get the number of searches for the word “αεροδρόμιο”*

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

*Get the number of searches for the word “face”*

Τελευταία αναζήτηση που γίνεται μέσα από το API είναι αυτή για την πιο searched λέξη χωρίς άλλα φίλτρα. Το αποτέλεσμα είναι σε μορφή json και πάλι και περιέχει την πιο searched λέξη και πόσες φορές αναζητήθηκε συνολικά. Ενδεχομένως το αποτέλεσμα να μην είναι ένα, αλλά στην προκειμένη περίπτωση επιστρέφεται μόνο ένα.

Εικόνα που περιέχει κείμενο

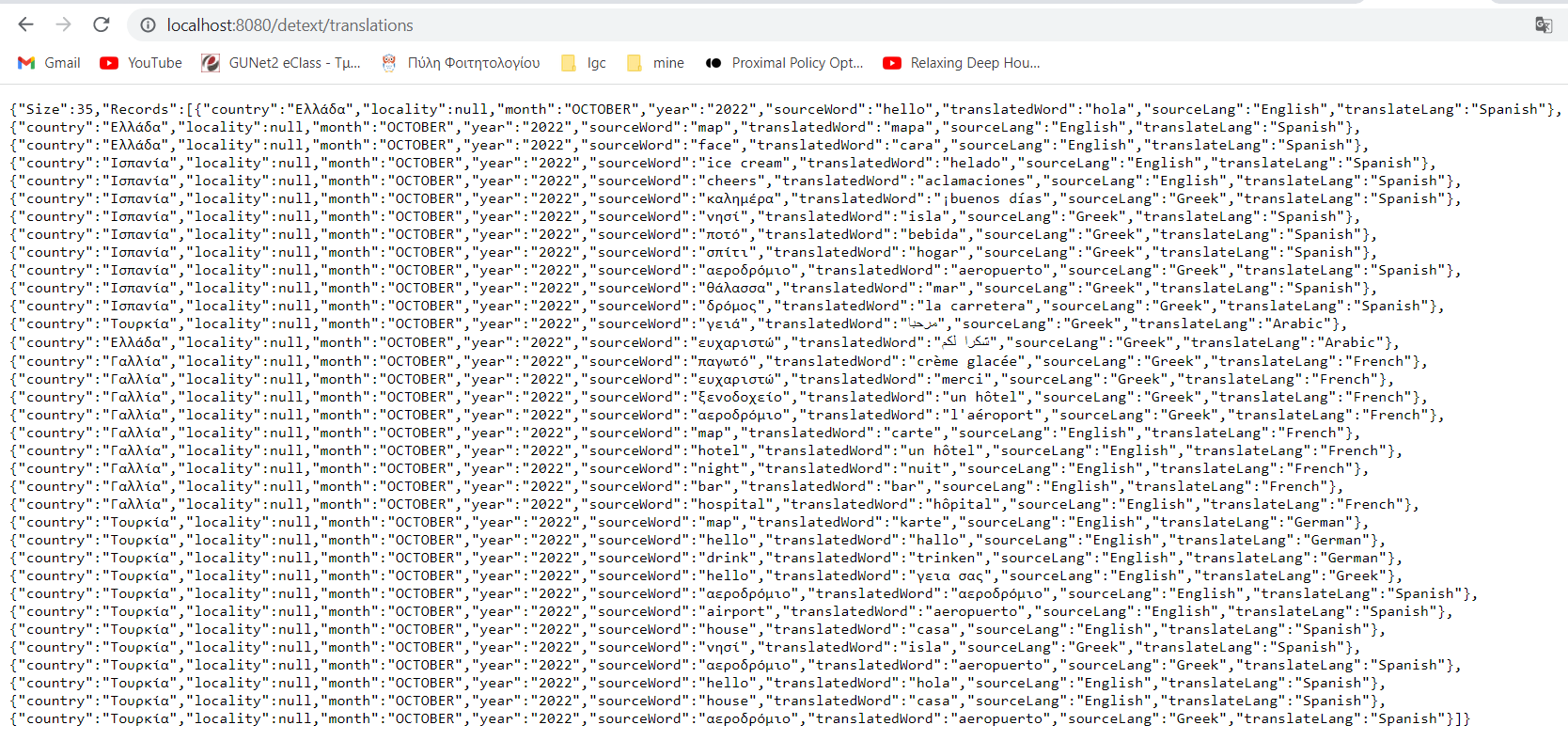
Περιγραφή που δημιουργήθηκε αυτόματα

*Get the most searched word without filtering*

* Παραδείγματα των requests μέσω browser

Όπως διακρίνεται και στην εικόνα παρακάτω, όπου εμφανίζονται όλα τα records από την database, δεν παρουσιάζονται με τον καλύτερο φιλικό τρόπο στον χρήστη κι αυτό γιατί δεν γίνεται κάποιο handling στο response ως προς την εμφάνισή τους.

<http://localhost:8080/detext/translations> για την αναζήτηση όλων των αναζητήσεων που έχουν καταγραφεί στην database.



Στην εικόνα παρακάτω διακρίνεται καλύτερα μια εγγραφή από αυτές που έχουν επιστραφεί, καθώς και το μέγεθος του αποτελέσματος.

Εικόνα που περιέχει κείμενο

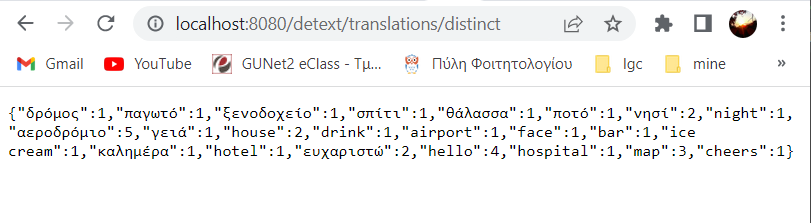
Περιγραφή που δημιουργήθηκε αυτόματα

Στη συνέχεια, παρουσιάζεται το response στο request που αφορά την πιο searched λέξη ανά χώρα.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Το επόμενο παράδειγμα αφορά τις διακριτές λέξεις που έχουν αναζητηθεί κατά καιρούς και τόπους, καθώς και το πλήθος των αναζητήσεων που έχουν γίνει για καθεμία από αυτές.



# ΠΗΓΕΣ

* Πηγές για σύνδεση με Firebase και το ML Kit
  + Text Recognition:
    - <https://developers.google.com/ml-kit/vision/text-recognition/android#java>
    - <https://developers.google.com/ml-kit/vision/text-recognition/languages>
    - <https://firebase.google.com/docs/ml-kit/recognize-text>
    - <https://developers.google.com/android/reference/com/google/mlkit/vision/common/InputImage>
  + Language Identification: <https://developers.google.com/ml-kit/language/identification/android>
  + Translation: <https://developers.google.com/ml-kit/language/translation/android>
* OpenCV Library
  + <https://opencv.org/about/>
  + Download: <https://opencv.org/releases/> και
  + <https://sourceforge.net/projects/opencvlibrary/>
  + <https://www.youtube.com/watch?v=imTTaZTSVQk>
  + <https://medium.com/android-news/a-beginners-guide-to-setting-up-opencv-android-library-on-android-studio-19794e220f3c>
* Για το API
  + <https://firebase.google.com/docs/reference/rest/database>
  + <https://start.spring.io/>
  + [*https://mvnrepository.com/artifact/com.google.firebase/firebase-admin*](https://mvnrepository.com/artifact/com.google.firebase/firebase-admin)
  + Set up: <https://firebase.google.com/docs/admin/setup>
  + <https://firebase.google.com/docs/database/admin/start>
  + <https://firebase.google.com/docs/database/rest/start>