Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη εφαρμογής Android για άτομα με προβλήματα όρασης με χρήση του Firebase ML Kit** **Development of an Android application for people with vision problems utilizing Firebase ML Kit**
Ονοματεπώνυμο Φοιτητή	**Σίμος Απέργης**
Πατρώνυμο	**Αντώνιος**
Αριθμός Μητρώου	**ΜΠΣΠ16001**
Επιβλέπων	**Αλέπης Ευθύμιος Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης **Μάρτιος 2021**

Τριμελής Εξεταστική Επιτροπή

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

1. Περίληψη

Η διπλωματική εργασία αναπτύχθηκε στα πλαίσια του μεταπτυχιακού προγράμματος «Προηγμένα Συστήματα Πληροφορικής» του τμήματος Πληροφορικής Πανεπιστημίου Πειραιώς.

Ο στόχος της διατριβής είναι να αναπτυχθεί μια εφαρμογή Android για άτομα με προβλήματα όρασης, όπου θα ανιχνεύει κείμενα, θα τα μεταφράζει και στη συνέχεια θα αναπαράγει ηχητικό αρχείο (Text-To-Speech) διαβάζοντας το μεταφρασμένο κείμενο.

Είναι ένα πρωτότυπο εργαλείο μετάφρασης που στοχεύει συγκεκριμένους χρήστες και αξιοποιεί σε ικανοποιητικό βαθμό ορισμένα από τα εργαλεία μηχανικής μάθησης που διατίθενται από την Google.

Για την επίτευξη του στόχου της εργασίας, η εφαρμογή αξιοποιεί τις δυνατότητες του Machine Learning (ML) Kit της Firebase, χρησιμοποιώντας

1. Ανίχνευση κειμένου
2. Αναγνώριση γλώσσας κειμένου
3. Μετάφραση κειμένου

Για την λειτουργία Text-To-Speech, αξιοποιεί τα εργαλεία που παρέχονται από το Android SDK, ενώ για τις γλώσσες που δεν υποστηρίζονται από αυτό, όπως για παράδειγμα τα Ελληνικά, η εφαρμογή αξιοποιεί το Google API TTS.

Abstract

This project is a post graduate dissertation of the MSc Advanced Information Systems of the department of Informatics of the University of Piraeus.

Purpose of this project is the development of an Android application for blind people which will identify, translate and read the translated text, through Text-To-Speech process.

It is a prototype translation tool, which targets specific audience and utilizes in a satisfying way some of the machine learning tools offered by Google.

Specifically, the following tools from Firebase Machine Learning (ML) Kit were utilized

1. Text identification
2. Language identification
3. Text translation

Additionally, the transformation from text to speech, will be conducted using internal tools from Android SDK, whereas for the languages that are not supported by itself, Google API TTS is being used.

2. Εισαγωγή

Ο ανθρώπινος νους επιχειρεί να κατανοήσει το περιβάλλον του μέσω της παρατήρησης και δημιουργώντας μία απλοποιημένη εκδοχή που ονομάζεται μοντέλο. Η δημιουργία ενός τέτοιου μοντέλου ονομάζεται επαγωγική μάθηση, ενώ γενικότερα η διαδικασία ονομάζεται επαγωγή. Επιπλέον, ο άνθρωπος χαρακτηρίζεται από την ικανότητα που έχει να οργανώνει και να συσχετίζει τις εμπειρίες και τις παραστάσεις που αποκομίζει, δημιουργώντας νέες δομές που αποκαλούνται πρότυπα. Κατά συνέπεια, η δημιουργία μοντέλων ή προτύπων ενός συνόλου δεδομένων από ένα υπολογιστικό σύστημα ονομάζεται Μηχανική Μάθηση (Machine Learning).

Το Machine Learning Kit SDK είναι ένα προϊόν από την Google που παρουσιάστηκε το 2018 και είναι ένα σύνολο εργαλείων ανάπτυξης λογισμικού που επιτρέπει στους προγραμματιστές να απλοποιήσουν την ενσωμάτωση μοντέλων μηχανικής μάθησης στις εφαρμογές τους για κινητές συσκευές. Κατά την χρήση του, δίνεται η δυνατότητα χρήσης κάποιων έτοιμων μοντέλων της Google καθώς και η δημιουργία νέων, ανάλογα με τις απαιτήσεις της εφαρμογής.

Στόχος της παρούσας εργασίας είναι η ανάπτυξη της Android εφαρμογής **ViDi**, η οποία θα εξυπηρετεί τις ανάγκες μετάφρασης κειμένου για άτομα με προβλήματα όρασης. Η εφαρμογή από την εκκίνηση της λειτουργεί αυτόνομα και αναζητά συνεχώς κείμενα, με τη βοήθεια της κάμερας. Κατά την επιτυχημένη αναγνώριση ενός κειμένου ακολουθεί η μετάφρασή του, έπειτα η μετατροπή αυτού σε ηχητικό αρχείο και κατά συνέπεια η αναπαραγωγή του. Η κινητή συσκευή, για πρακτικούς λόγους, ενδείκνυται να χρησιμοποιηθεί ως περιδέραιο, ώστε η κάμερα να εστιάζει προς την κατεύθυνση που κινείται ο χρήστης.

Από πλευράς παραμετροποίησης της εφαρμογής, ο χρήστης έχει τη δυνατότητα να ρυθμίσει την γλώσσα ανίχνευσης κειμένου, την γλώσσα στην οποία θα πραγματοποιηθεί η μετάφραση καθώς και τη συχνότητα που η κάμερα θα λαμβάνει εικόνες.

3. Ανασκόπηση πεδίου

Κατά την πάροδο των τελευταίων ετών, ολοένα και περισσότερες εφαρμογές μετάφρασης εισάγονταν στο Play Store, αφενός λόγω τις αναγκαιότητας για ένα άμεσο μεταφραστικό εργαλείο (για παράδειγμα στον τουρισμό) και αφετέρου λόγω εξέλιξης της μηχανικής μάθησης.

Οι κινητές συσκευές συνετέλεσαν στην εξέλιξη των μεταφραστικών εφαρμογών όχι μόνο λόγω φορητότητας αλλά και με την αξιοποίηση των αισθητήρων τους. Έτσι, ενώ αρχικά η μετάφραση περιορίζονταν στον παραδοσιακό τρόπο, από κείμενο σε κείμενο, αργότερα αξιοποιήθηκε ο αισθητήρας ήχου όπου δόθηκε η δυνατότητα μετάφρασης από ένα φωνητικό αρχείο (Speech-to-Text αλλά και Speech-to-Speech). Κατ' επέκταση, αξιοποιήθηκε και η κάμερα της συσκευής με αποτέλεσμα να υπάρχει η δυνατότητα μετάφρασης ενός κειμένου που περιέχεται σε μια εικόνα. Σαφώς, πρωταρχικό ρόλο στις παραπάνω υπηρεσίες έχει η μηχανική μάθηση.

Πρωτοπόρος εφαρμογή μετάφρασης εδώ και πολλά χρόνια είναι όπως γνωρίζουμε το **Google Translate** play.google.com/store/google_translate. Είναι το πιο διαδεδομένο εργαλείο μετάφρασης όσον αφορά web αλλά και mobile εφαρμογές. Υποστηρίζει περισσότερες από εκατό (100) γλώσσες. Πέρα από της βασικές λειτουργίες μετάφρασης που αναφέρθηκαν παραπάνω, η Google έχει προσθέσει ένα επιπλέον χαρακτηριστικό όπου ο χρήστης μπορεί με τη βοήθεια της αφής να σχεδιάσει την λέξη που επιθυμεί να μεταφράσει, αντί να χρησιμοποιήσει το πληκτρολόγιο της συσκευής.

Επίσης ένα πολύ γνωστό εργαλείο μετάφρασης είναι το **iTranslate** play.google.com/store/apps/details?id=at.nk.tools.iTranslate. Η συγκεκριμένη εφαρμογή διαθέτει δυνατότητες εξατομίκευσης, χρήση εναλλακτικών διαλέκτων καθώς και παραγωγή λεξιλογίων με διακριτές έννοιες. Επιπλέον, πολύ ενδιαφέρον είναι το γεγονός ότι η εφαρμογή διαθέτει μεταφρασμένες φράσεις που χρησιμοποιούνται πολύ συχνά, χαρακτηριστικό ιδιαίτερα χρήσιμο για άτομα που ταξιδεύουν σε ξένες χώρες.

Ακόμα μια εφαρμογή μετάφρασης που ξεχωρίζει είναι το **Microsoft Translator** play.google.com/store/apps/details?id=com.microsoft.translator**.** Ιδιαίτερο χαρακτηριστικό είναι η μετάφραση συζήτησης σε πραγματικό χρόνο όπως επίσης και η αποστολή μεταφράσεων σε άλλες εφαρμογές.

Η εφαρμογή **ViDi** ενσωματώνει όλα τα παραπάνω χαρακτηριστικά, με τη διαφορά ότι η εισαγωγή ενός κειμένου προς μετάφραση ξεκινάει πάντα από λήψη εικόνας και το μεταφρασμένο κείμενο αναπαράγεται ηχητικά, ώστε να εξυπηρετήσει τον σκοπό της.

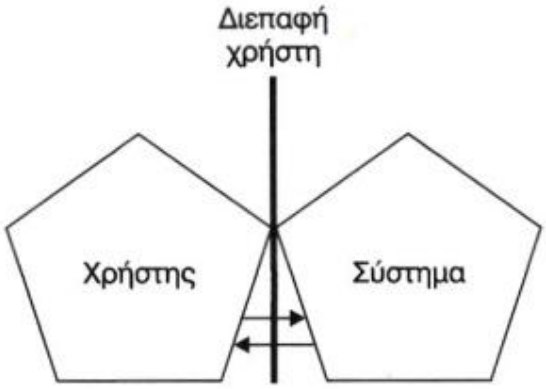
4. Παρουσίαση και χρήση της εφαρμογής

4.1 ViDi

Η εφαρμογή ViDi έχει υλοποιηθεί για κινητές συσκευές Android. Στόχος της εφαρμογής είναι η ανάπτυξη ενός εργαλείου μετάφρασης για άτομα με προβλήματα όρασης. Αυτό επιτυγχάνεται με την περιοδική ανίχνευση κειμένου, τη μετάφραση του και κατ' επέκταση την ηχητική αναπαραγωγή αυτού.

4.2 Διεπαφή χρήστη

Ο όρος διεπαφή χρήστη (user interface) είναι το σύνολο των συστατικών ενός συστήματος το οποίο επιτρέπει αμφίδρομη επικοινωνία μεταξύ συστήματος και χρήστη. Η διεπαφή χρήστη ενός συστήματος έχει σχέση με το ίδιο το σύστημα, το χρήστη του συστήματος και τον τρόπο που αλληλεπιδρούν μεταξύ τους (εικόνα 1). Ο όρος θέλει να δείξει το σημείο επαφής χρήστη και συσκευής, δηλαδή την γραμμή επαφής πίσω από την μια μεριά της οποίας βρίσκεται η συσκευή και πίσω από την άλλη μεριά ο άνθρωπος. Έτσι λοιπόν η διεπαφή χρήστη περιέχει στοιχεία που είναι τμήματα τόσο του υλικού του συστήματος, όσο και του λογισμικού που «τρέχει» σε αυτό.



Εικόνα 1

Ως στοιχεία του υλικού του συστήματος που περιλαμβάνονται στη διεπαφή χρήστη μπορούν να αναφερθούν μια οθόνη επαφής, μια φωτογραφίδα (lightpen) ή τα φυσικά πλήκτρα πάνω στις κινητές συσκευές. Μέρη του λογισμικού της διεπαφής χρήστη είναι, για παράδειγμα, τα μηνύματα λάθους, τα ηχητικά μηνύματα, τα εργαλεία πλοήγησης, εικόνες, σύμβολα και αντικείμενα πάνω στην οθόνη, κάθε τι που διαθέτει το λογισμικό σαν στοιχείο αλληλεπίδρασης του συστήματος με το χρήστη. Με άλλα λόγια, ο όρος σε ότι αφορά τη λογισμική του υπόσταση, σημαίνει ένα σύνολο από οπτικές και ακουστικές παραμέτρους, που παρέχει η συσκευή προς το χρήστη, μέσω του εκάστοτε εκτελούμενου προγράμματος, με σκοπό την καλύτερη επικοινωνία και συνεργασία μεταξύ ανθρώπου και μηχανής.

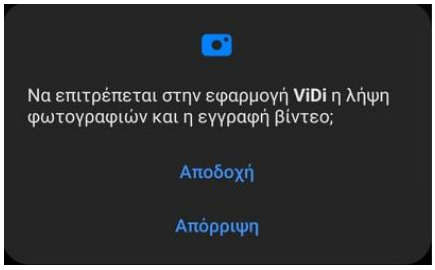
Με λίγα λόγια μπορούμε να παρομοιάσουμε τη διεπαφή χρήστη σαν κανάλι επικοινωνίας μεταξύ του χρήστη και της συσκευής Android.

Το βασικότερο στοιχείο διεπαφής χρήστη λογισμικού για την εφαρμογή ViDi είναι τα ηχητικά μηνύματα, τα οποία είναι και το κύριο μέσο αλληλεπίδρασης για τους χρήστες με προβλήματα όρασης. Παρ' όλα αυτά για λόγους παρακολούθησης της λειτουργίας της εφαρμογής, όπως θα δούμε παρακάτω, έχουν αναπτυχθεί και γραφικά στοιχεία με τα οποία ένας χρήστης δοκιμών μπορεί να αλληλεπιδράσετι με την εφαρμογή.

4.3 Αρχική οθόνη

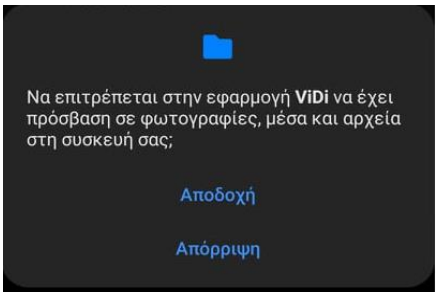
Κατά την εκκίνηση της εφαρμογής για πρώτη φορά, ζητείται η άδεια του χρήστη ώστε η εφαρμογή να έχει πρόσβαση

1. Στην κάμερα της συσκευής



Εικόνα 3

2. Στα αρχεία της συσκευής, για ανάγνωση και εγγραφή



Εικόνα 4

Αυτό συμβαίνει διότι από το λειτουργικό Android 6.0 και μετά, οι χρήστες επιτρέπουν δικαιώματα σε εφαρμογές ενώ εκτελούνται και όχι κατά την εγκατάσταση. Αυτή η προσέγγιση βελτιστοποιεί τη διαδικασία εγκατάστασης της εφαρμογής, καθώς ο χρήστης δεν απαιτεί δικαιώματα παραχώρησης κατά την εγκατάσταση ή κατά την ενημέρωση της εφαρμογής. Η πρώτη άδεια παραχωρείται ώστε η εφαρμογή να έχει τη δυνατότητα να χρησιμοποιεί την κάμερα της συσκευής για τον εντοπισμό κειμένων.

Η δεύτερη άδεια παραχωρείται ώστε η εικόνα να αποθηκεύεται προσωρινά στον αποθηκευτικό χώρο της συσκευής, έως ότου να επεξεργαστεί από το εργαλείο ανίχνευσης κειμένου. Μετά το πέρας της επεξεργασίας, η εικόνα διαγράφεται από τη συσκευή.

Εφόσον παραχωρήσει ο χρήστης τις παραπάνω άδειες, εμφανίζεται η αρχική οθόνη της εφαρμογής. Σε αυτό το σημείο να τονίσουμε ότι η αρχική οθόνη είναι και η κεντρική οθόνη της εφαρμογής, η οποία στην ονοματολογία του Android λέγεται **δραστηριότητα** (**Activity**). Μια δραστηριότητα είναι μια ενιαία οθόνη του Android. Είναι σαν ένα παράθυρο ή πλαίσιο της Java (Java Frame). Με τη βοήθεια της δραστηριότητας, μπορούμε να τοποθετήσουμε όλα τα στοιχεία ή τα γραφικά στοιχεία της διεπαφής χρήστη σε μία οθόνη.

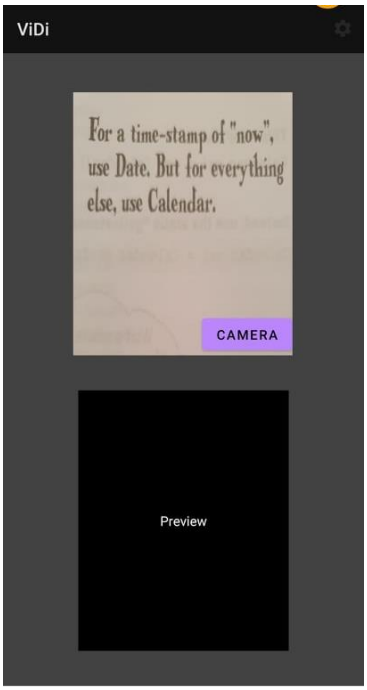
Η δραστηριότητα της αρχικής οθόνης περιέχει δύο στοιχεία. Αυτά τα στοιχεία είναι ουσιαστικά δύο μικρότερες οθόνες και ονομάζονται **Fragments**. Το fragment αντιπροσωπεύει ένα επαναχρησιμοποιήσιμο τμήμα της διεπαφής χρήστη της εφαρμογής. Ένα fragment καθορίζει και διαχειρίζεται τη δική του διάταξη, έχει τον δικό του κύκλο ζωής και μπορεί να χειριστεί τα δικά του συμβάντα εισόδου. Επίσης δεν είναι ανεξάρτητα - πρέπει να φιλοξενούνται από μια δραστηριότητα.

Τα fragments επιτρέπουν την επαναχρησιμοποίηση τους στο περιβάλλον εργασίας της δραστηριότητάς, επιτρέποντάς να διαιρούμε το περιβάλλον εργασίας χρήστη σε διακριτά κομμάτια. Οι δραστηριότητες είναι ένα ιδανικό μέρος για να τοποθετηθούν καθολικά στοιχεία στη διεπαφή χρήστη της εφαρμογής, όπως ένα εργαλείο πλοήγησης. Τα fragments χρησιμοποιούνται για τον καθορισμό και τη διαχείριση της διεπαφής χρήστη μιας μεμονωμένης οθόνης ή ενός τμήματος μιας οθόνης. Είναι ιδανικά για μεγάλες οθόνες, όπως για παράδειγμα στις ταμπλέτες, διότι επιτρέπουν την μέγιστη αξιοποίηση της ευρείας οθόνης καθότι μπορούμε να συνθέσουμε πολλά μικρά στοιχεία διεπαφής χρήστη τα οποία είναι εξίσου χρήσιμα και για μικρότερες οθόνες όπως τα κινητά τηλέφωνα, λόγω της επαναχρησιμοποίησης τους.

Τα fragments που περιλαμβάνονται στην αρχική οθόνη είναι τα εξής:

i. Η οθόνη που απεικονίζει την κάμερα. Ουσιαστικά, η εφαρμογή της κάμερας είναι ενσωματωμένη στο πλαίσιο του άνω fragment της κεντρικής οθόνης.

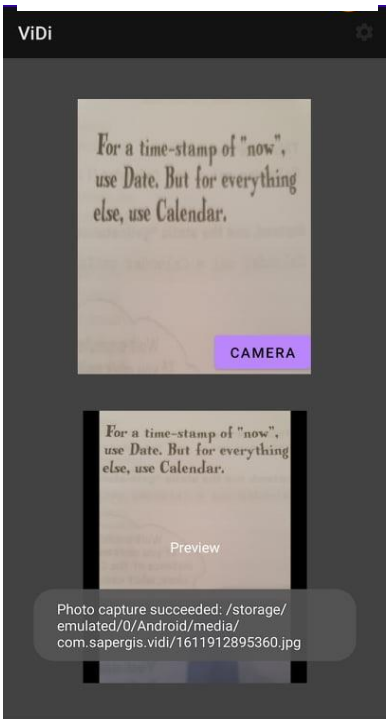
ii. Η οθόνη που προβάλλει την τελευταία λήψη που πραγματοποίησε η κάμερα. Εδώ έχουμε ενσωματώσει εργαλείο που προβάλλει την τελευταία φωτογραφία που τράβηξε η κάμερα, στην οποία θα πραγματοποιηθεί αναγνώριση κειμένου.



Εικόνα 5

Στην διπλανή εικόνα, το άνω fragment εκμεταλλεύεται την κάμερα της συσκευής. Μπορούμε να δούμε σε πραγματικό χρόνο την εικόνα στην οποία εστιάζει η κάμερα, όπως και σε μια τυπική εφαρμογή κάμερας. Εφόσον πραγματοποιηθεί λήψη εικόνας στον προκαθορισμένο χρόνο, η άνω οθόνη διατηρεί την εικόνα που έχει ληφθεί για λίγα δευτερόλεπτα και έπειτα συνεχίζει να απεικονίζει σε πραγματικό χρόνο.

Η κάτω οθόνη, κατά την εκκίνηση της εφαρμογής και προτού πραγματοποιηθεί η πρώτη λήψη εικόνας, παρουσιάζεται με σκούρο φόντο και με την ένδειξη "Preview" στο κέντρο ούτως ώστε να γίνεται αντιληπτό ότι χρησιμοποιείται για να παρουσιάζει την τελευταία λήψη της κάμερας.

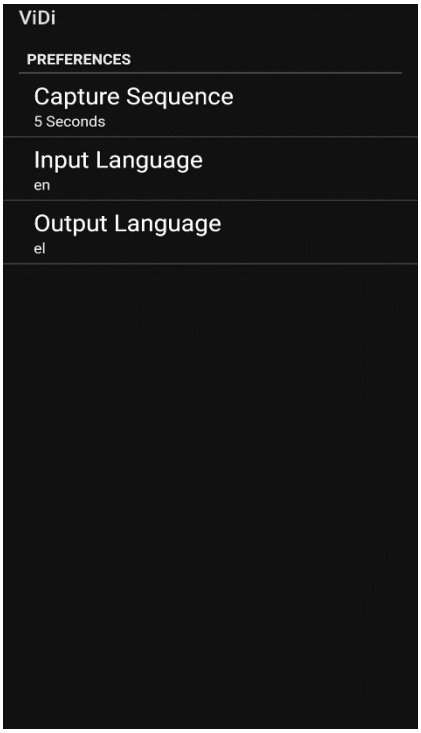


Εικόνα 6

Μόλις πραγματοποιηθεί λήψη, αυτόματα η φωτογραφία απεικονίζεται στην κάτω οθόνη – την οθόνη προεπισκόπησης (Preview). Παράλληλα, εμφανίζεται και ένα σύντομο μήνυμα το οποίο επιβεβαιώνει την επιτυχή λήψη της εικόνας καθώς σηματοδοτεί και την έναρξη διαδικασίας αναγνώρισης κειμένου. Τέλος, το σύντομο μήνυμα προβάλλει και το μονοπάτι στο οποίο αποθηκεύτηκε προσωρινά μέχρι να ολοκληρωθεί η επεξεργασία. Η εικόνα παραμένει στην οθόνη προεπισκόπησης έως ότου να ολοκληρωθεί η ροή και να ληφθεί νέα εικόνα.

4.4 Μενού επιλογών

Η δεύτερη και τελευταία οθόνη της εφαρμογής, είναι το μενού επιλογών και εξατομίκευσης. Ο χρήστης μπορεί να μεταφερθεί σε αυτό το μενού πατώντας το εικονίδιο με το γρανάζι, που βρίσκεται πάνω και δεξιά στην μπάρα πλοήγησης της αρχικής οθόνης.



Εικόνα 7

Σε αυτή την οθόνη υπάρχουν οι εξής επιλογές

Capture Sequence: Συχνότητα αυτόματης λήψης της κάμερας. Μπορούμε να ρυθμίσουμε ανά πόσα δευτερόλεπτα η κάμερα θα πραγματοποιεί λήψεις. Το διαθέσιμο εύρος επιλογών είναι 5-10 δευτερόλεπτα και η προκαθορισμένη τιμή είναι 5 δευτερόλεπτα.

Input Language*:* Γλώσσα εισόδου. Αναφέρεται στη γλώσσα προς μετάφραση ή αλλιώς στην γλώσσα αναγνώρισης κειμένων. Προκαθορισμένη γλώσσα είναι τα Αγγλικά, που σημαίνει ότι μόνο τα κείμενα αγγλικής γλώσσας θα μεταφράζονται.

Output Language*:* Γλώσσα εξόδου. Είναι η γλώσσα στην οποία θα μεταφραστεί το κείμενο. Προκαθορισμένη γλώσσα μετάφρασης της εφαρμογής είναι τα Ελληνικά.

5. Αρχιτεκτονική συστήματος

5.1 Εργαλεία προγραμματισμού

5.1.1 Android Studio IDE

Η εφαρμογή ViDi αναπτύχθηκε με τη βοήθεια του λογισμικού Android Studio 3.0, το οποίο είναι το επίσημο IDE της Google (Integrated Development Environment) για ανάπτυξη native εφαρμογών Android. Συγκεκριμένα, το Android Studio είναι μια σουίτα που περιλαμβάνει ένα σύνολο εργαλείων ανάπτυξης mobile εφαρμογών, ό-πως για παράδειγμα

- **Device Emulator**: Εικονική κινητή συσκευή με την οποία μπορούμε να δοκι-μάζουμε τον κώδικα που αναπτύσσουμε. Είναι ιδιαίτερα χρήσιμο εργαλείο διότι δίνεται η δυνατότητα να διεξάγουμε δοκιμές σε διαφορετικές εκδόσεις Android χωρίς να έχουμε στη διάθεση μας εναλλακτικές πραγματικές συ-σκευές.

- **Debugger**: Από τα βασικότερα εργαλεία στην ανάπτυξη εφαρμογών, καθώς συμβάλλει στον εντοπισμό και την επίλυση λαθών (bugs).
- **Profiler:** Το εργαλείο Android Profiler παρέχει δεδομένα σε πραγματικό χρόνο για να βοηθήσει τον προγραμματιστή να κατανοήσει και να βελτιστοποιήσει την χρήση της CPU, της μνήμης, του δικτύου και των πόρων μπαταρίας, όσον αφορά την εφαρμογή που αναπτύσσει.
- **Logcat:** Είναι μια κονσόλα στην οποία εμφανίζονται βοηθητικά μηνύματα κατά τη χρήση της εφαρμογής. Τα μηνύματα αυτά προστίθενται στον κώδικα από τον προγραμματιστή και είναι ιδιαίτερα χρήσιμα για την κατανόηση της λειτουργίας καθώς και την απασφαλμάτωση της εφαρμογής.
- **Layout Inspector:** Το Layout Inspector επιτρέπει τη σύγκριση της διάταξης της εφαρμογής με πρότυπα σχεδίασης. Επίσης, παρέχει τη δυνατότητα να εμφανίσουμε μια μεγεθυμένη ή τρισδιάστατη προβολή της εφαρμογής και να εξετάσουμε λεπτομέρειες σχετικά με τη διάταξή της κατά το χρόνο εκτέλεσης.

5.1.2 Γλώσσα ανάπτυξης

Η εφαρμογή ViDi αναπτύχθηκε χρησιμοποιώντας την γλώσσα προγραμματισμού **Java** σε συνδυασμό με το Android Software Development Kit 29.0 (**SDK**), το οποίο αποτελείται από μια ομάδα βιβλιοθηκών και εργαλείων που επιτρέπουν τον προγραμματισμό εφαρμογών για κινητές συσκευές.

Το Android SDK είναι μια συλλογή από εργαλεία ανάπτυξης λογισμικού και βιβλιοθήκες που απαιτούνται για την ανάπτυξη εφαρμογών Android. Κάθε φορά που η Google κυκλοφορεί μια νέα έκδοση Android ή μια ενημέρωση, κυκλοφορεί επίσης ένα αντίστοιχο SDK, το οποίο πρέπει να κατεβάσουν και να εγκαταστήσουν οι προγραμματιστές. Συνήθως αναπτύσσουμε εφαρμογές χρησιμοποιώντας το Android Studio. Αξίζει να σημειωθεί επίσης ότι μπορούμε να κατεβάσουμε και να χρησιμοποιήσετε το Android SDK ανεξάρτητα από το Android Studio.

Το Android SDK περιλαμβάνει όλα τα απαραίτητα εργαλεία για την ανάπτυξη προγραμμάτων από το μηδέν, ακόμη και για τη δοκιμή τους. Αυτά τα εργαλεία παρέχουν μια ομαλή ροή της διαδικασίας από την ανάπτυξη και τον εντοπισμό σφαλμάτων, έως τη διανομή στους χρήστες.

Το Android SDK είναι συμβατό με Windows, macOS και Linux, ώστε να μπορούν να το αναπτυχθούν εφαρμογές σε οποιαδήποτε από αυτές τις πλατφόρμες.

5.2 Τεχνολογίες και μέθοδοι που χρησιμοποιήθηκαν

5.2.1 Firebase ML Kit

Θεμελιώδες εργαλείο για την υλοποίηση της εφαρμογής ViDi είναι το **Firebase Machine Learning Kit**. Το ML Kit είναι ένα SDK που φέρνει την τεχνογνωσία της μηχανικής μάθησης της Google σε εφαρμογές Android. Διευκολύνει την εφαρμογή τεχνικών μηχανικής μάθησης στις εφαρμογές παρέχοντας τεχνολογίες της Google, όπως το Google Cloud Vision API, το TensorFlow Lite και το Android Neural Networks API, σε ένα SDK. Παρακάτω παρουσιάζονται και αναλύονται τα εργαλεία που επιλέχθηκαν από το ML Kit για την υλοποίηση της εφαρμογής.

Text Recognition: Το εργαλείο αυτό χρησιμοποιείται για την αναγνώριση κειμένου σε εικόνες. Η διεπαφή (API) αναγνώρισης κειμένου χρησιμοποιεί ένα αδεσμοποίητο μοντέλο που πρέπει να ληφθεί πριν τη χρήση του. Υπάρχει επιλογή να εκτελεστεί λήψη κατά την εγκατάσταση της εφαρμογής ή κατά την πρώτη εκκίνηση. Για να πραγματοποιηθεί αναγνώριση κειμένου σε μια εικόνα, χρησιμοποιούμε τον ανιχνευτή κειμένου (text recognizer) του ML kit όπως φαίνεται παρακάτω στην εικόνα 8.

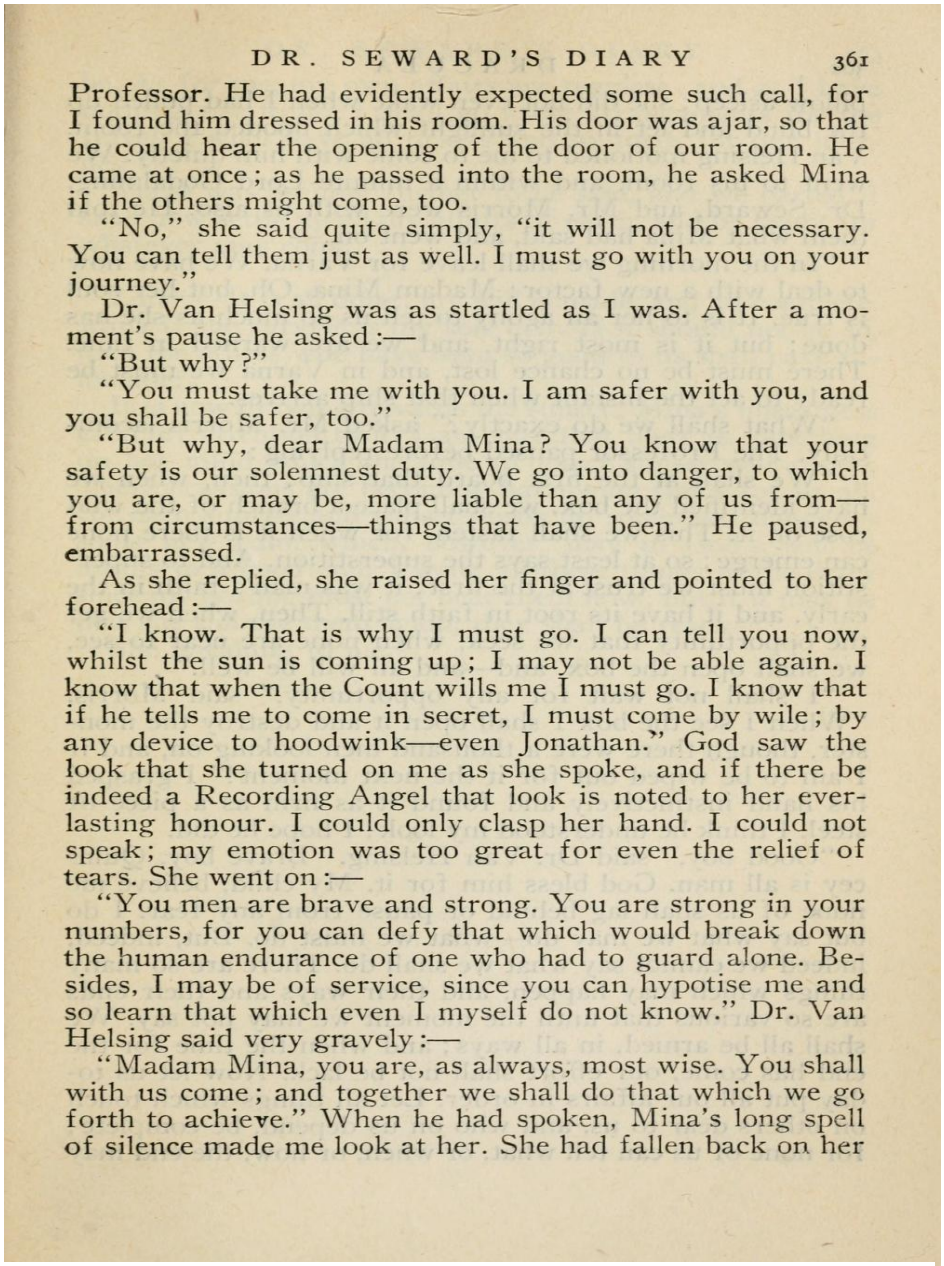
```
InputImage inputImage = InputImage.fromBitmap(bitmap, 90);
TextRecognizer recognizer = TextRecognition.getClient();
Task<Text> result = recognizer.process(inputImage);
result.addOnSuccessListener( visionText -> {
    // Task completed successfully
    VDHelper.debugLog( className: "VDTextRecognizer", message: "TEXT FOUND [DEVICE]=>"
            +visionText.getText());
    iVDTextOperations.onTextRecognized(visionText.getText());
    recognizer.close();
```

Εικόνα 8

Η εικόνα που έχει ληφθεί από την κάμερα μετατρέπεται σε αντικείμενο τύπου InputImage το οποίο παίρνει την απαραίτητη μορφή ώστε να επεξεργαστεί από το ML Kit. Έπειτα δημιουργείται το στιγμιότυπο του TextRecognizer το οποίο αναλαμβάνει την επεξεργασία για την αναγνώριση του κειμένου. Εάν η λειτουργία αναγνώρισης κειμένου πετύχει, τότε επιστρέφεται ένα αντικείμενο κειμένου από τον μηχανισμό επεξεργασίας της εικόνας. Ένα αντικείμενο κειμένου περιέχει το πλήρες κείμενο που αναγνωρίζεται στην εικόνα και ένα ή περισσότερα αντικείμενα TextBlock (ενδεχομένως και κανένα).

Κάθε TextBlock αντιπροσωπεύει ένα ορθογώνιο μπλοκ κειμένου, το οποίο περιέχει μηδέν ή περισσότερα αντικείμενα γραμμής. Κάθε αντικείμενο γραμμής περιέχει μηδέν ή περισσότερα αντικείμενα στοιχείων, τα οποία αντιπροσωπεύουν λέξεις και οντότητες που μοιάζουν με λέξεις, όπως ημερομηνίες και αριθμοί. Για κάθε αντικείμενο TextBlock, Line και Element, μπορούμε να αναγνωρίσουμε το κείμενο στην περιοχή και τις συντεταγμένες οριοθέτησης της περιοχής.

Παρακάτω ακολουθεί ανάλυση του αποτελέσματος αναγνώρισης κειμένου που προκύπτει από την επεξεργασία της εικόνας 9.



DR. SEWARD'S DIARY 361

Professor. He had evidently expected some such call, for I found him dressed in his room. His door was ajar, so that he could hear the opening of the door of our room. He came at once; as he passed into the room, he asked Mina if the others might come, too.

"No," she said quite simply, "it will not be necessary. You can tell them just as well. I must go with you on your journey."

Dr. Van Helsing was as startled as I was. After a moment's pause he asked:—

"But why?"

"You must take me with you. I am safer with you, and you shall be safer, too."

"But why, dear Madam Mina? You know that your safety is our solemnest duty. We go into danger, to which you are, or may be, more liable than any of us from—from circumstances—things that have been." He paused, embarrassed.

As she replied, she raised her finger and pointed to her forehead:—

"I know. That is why I must go. I can tell you now, whilst the sun is coming up; I may not be able again. I know that when the Count wills me I must go. I know that if he tells me to come in secret, I must come by wile; by any device to hoodwink—even Jonathan." God saw the look that she turned on me as she spoke, and if there be indeed a Recording Angel that look is noted to her everlasting honour. I could only clasp her hand. I could not speak; my emotion was too great for even the relief of tears. She went on:—

"You men are brave and strong. You are strong in your numbers, for you can defy that which would break down the human endurance of one who had to guard alone. Besides, I may be of service, since you can hypotise me and so learn that which even I myself do not know." Dr. Van Helsing said very gravely:—

"Madam Mina, you are, as always, most wise. You shall with us come; and together we shall do that which we go forth to achieve." When he had spoken, Mina's long spell of silence made me look at her. She had fallen back on her

Εικόνα 9

Έπειτα από την επεξεργασία που υφίσταται η εικόνα 9, από τους αλγόριθμους του ανιχνευτή κειμένου, προκύπτουν οι παρακάτω πληροφορίες

Recognized Text	
Text	DR. SEWARD'S DIARY 361 Professor. He had evidently expected some such call, for I found him dressed in his room. His door was ajar, so that he could hear the opening of the door of our room. He came at once; as he passed into the room, he asked Mina if the others might come, too. "No," she said quite simply, "it will not be necessary. You can tell them just as well. I must go with you on your journey." Dr. Van Helsing was as startled as I was. After a mo- ment's pause he asked: "But why?" ...
Blocks	(1 block)

Block 0	
Text	DR . SEWARD ' S DIARY 361 Professor . He had evidently expected some such call , for I found him dressed in his room . His door was ajar , so that he could hear the opening of the door of our room . He came at once ; as he passed into the room , he asked Mina if the others might come , too . " No , " she said quite simply , " it will not be necessary . You can tell them just as well . I must go with you on your journey . " Dr . Van Helsing was as startled as I was . After a mo ment ' s pause he asked : ...
Confidence	0.98

Frame	(25.0, 21.0, 359.0, 583.0)
Recognized Language Code	en
Paragraphs	(10 paragraphs)

Paragraph 1	
Text	" No , " she said quite simply , " it will not be necessary . You can tell them just as well . I must go with you on your journey . "
Confidence	0.98
Frame	(29.0, 110.0, 355.0, 44.0)
Recognized Language Code	en
Words	(34 words)

Word 7	
Text	simply
Confidence	0.99
Frame	(179.0, 110.0, 37.0, 15.0)
Recognized Language Code	en
Symbols	(6 symbols)

Symbol 0	
Text	s
Confidence	1.00
Frame	(179.0, 110.0, 3.0, 15.0)
Recognized Language Code	en

Language Identification: Χρησιμοποιείται για τον προσδιορισμό της γλώσσας μιας συμβολοσειράς κειμένου. Κατά την επεξεργασία μπορούμε να λάβουμε την

πιο πιθανή γλώσσα της συμβολοσειράς ή να λάβουμε βαθμολογίες εμπιστοσύνης για όλες τις πιθανές γλώσσες της συμβολοσειράς. Το ML Kit έχει την δυνατότητα να αναγνωρίσει κείμενο σε 103 διαφορετικές γλώσσες στα εγγενή σενάρια τους.

Παρακάτω στην εικόνα 10, βλέπουμε το κομμάτι του κώδικα που επεξεργάζεται το κείμενο ώστε να αναγνωρίσει την γλώσσα

```
LanguageIdentifier languageIdentifier = LanguageIdentification.getClient();
Task<String> result = languageIdentifier.identifyLanguage(vdText.getRawText());
result.addOnSuccessListener(languageCode ->{
                        String message;
                        //Checking if language was recognized and also if it is equals to
                        //the input language we set in preferences
                        if ( !UND.equals(languageCode) && languageCode.equals(inputLanguage)) {
                            iVDTextOperations.onLanguageIdentified(languageCode, vdText);
                        }else if (UND.equals(languageCode)){
                            message = "Can't identify language.";
                            terminate(iVDTextOperations, message);
                        }
                        else {
                            message = "Language not supported";
                            terminate(iVDTextOperations, message);
                        }

                })
```

Εικόνα 10

Για να προσδιορίσουμε τη γλώσσα ενός κειμένου, πρέπει να δημιουργηθεί ένα στιγμιότυπο της κλάσης LanguageIdentifier και στη συνέχεια να περάσουμε την συμβολοσειρά του κειμένου ως παράμετρο στη μέθοδο identifyLanguage η οποία μας επιστρέφει τον κωδικό της γλώσσας (en, el, und κ.ο.κ).

Text translation: Εργαλείο μετάφρασης κειμένου το οποίο υποστηρίζει 59 γλώσσες. Ο τρόπος με τον οποίο υλοποιήσαμε την μετάφραση κειμένου απεικονίζεται στην εικόνα 11.

```
TranslatorOptions options = new TranslatorOptions.Builder()
        .setSourceLanguage(TranslateLanguage.ENGLISH)
        .setTargetLanguage(TranslateLanguage.GREEK)
        .build();
final Translator vdTranslator = Translation.getClient(options);
DownloadConditions conditions = new DownloadConditions.Builder()
        .requireWifi()
        .build();
Task<Void> modelDownload = vdTranslator.downloadModelIfNeeded(conditions);
modelDownload.addOnSuccessListener(aVoid ->
        translate(vdText, vdTranslator, ivdTextOperations));
modelDownload.addOnFailureListener(e -> {
    ivdTextOperations.onOperationTerminated(e.getMessage());
    vdTranslator.close();
});
```

Εικόνα 11

Αρχικά, για να ξεκινήσει η διαδικασία μετάφρασης, θα πρέπει πρώτα να γίνει έλεγχος αν στη συσκευή υπάρχει εγκατεστημένο το μοντέλο μετάφρασης της συγκεκριμένης γλώσσας. Το μοντέλο περιέχει όλες τις απαραίτητες πληροφορίες για την διαδικασία μετάφρασης. Εάν δεν υπάρχει , τότε πραγματοποιείται λήψη του μοντέλου πριν συνεχίσει η διαδικασία.

```
Task<String> translation = vdTranslator.translate(vdText.getRawText());
translation.addOnSuccessListener(new OnSuccessListener<String>() {
    @Override
    public void onSuccess(String s) {
        vdText.setTranslatedText(translation.getResult());
        VDHelper.debugLog(getClass().getSimpleName(),
                message: "Translating to => ["+vdText.getTraslateTo()+"]");
        VDHelper.debugLog(getClass().getSimpleName(),
                message: "Translated text is => ["+translation.getResult()+"]");
        ivdTextOperations.onTextTranslated(vdText);
        vdTranslator.close();
    }
```

Εικόνα 12

Για να μεταφράσουμε το κείμενο, χρειαζόμαστε ένα στιγμιότυπο της κλάσης Translator. Έπειτα περνάμε την συμβολοσειρά κειμένου στην μέθοδο translate του translator και μας επιστρέφει το μεταφρασμένο κείμενο.

5.2.2 Text-To-Speech Google API

Για τη μετατροπή της μεταφρασμένης συμβολοσειράς κειμένου σε ηχητικό αρχείο, χρησιμοποιείται το TTS Google API. Επιλέχθηκε η συγκεκριμένη υπηρεσία επειδή διαθέτει Text-To-Speech σε Ελληνική γλώσσα, σε αντίθεση με την εσωτερική βιβλιοθήκη του Android.

Το Text-to-Speech επιτρέπει στους προγραμματιστές να δημιουργήσουν φυσικό ήχο, συνθετικό ανθρώπινο λόγο ως ήχο που μπορεί να αναπαραχθεί. Μας δίνει τη δυνατότητα να χρησιμοποιήσουμε τα αρχεία δεδομένων ήχου που δημιουργούμε χρησιμοποιώντας το Text-to-Speech για να τροφοδοτήσουμε μια εφαρμογή με περαιτέρω δυνατότητες.

Το Text-to-Speech μετατρέπει την εισαγωγή κειμένου ή Speech Synthesis Markup Language (SSML) σε δεδομένα ήχου, όπως MP3 ή LINEAR16 (η κωδικοποίηση που χρησιμοποιείται σε αρχεία WAV). Επιτρέπει στους προγραμματιστές να συνθέσουν ομιλία φυσικής ακρόασης με περισσότερες από 100 φωνές, διαθέσιμες σε πολλές γλώσσες και παραλλαγές. Εφαρμόζει την πρωτοποριακή έρευνα της DeepMind στο WaveNet και στα ισχυρά νευρωνικά δίκτυα της Google για να προσφέρει την υψηλότερη δυνατή πιστότητα. Είναι ένα εύχρηστο API που μπορεί να χρησιμοποιηθεί από οποιαδήποτε εφαρμογή η οποία έχει πρόσβαση στο διαδίκτυο. Απευθύνεται σε εφαρμογές όπως το ViDi, που αναπαράγει ήχο ανθρώπινης ομιλίας στους χρήστες.

Παρακάτω βλέπουμε ένα παράδειγμα αιτήματος προς το TTS API για την μετατροπή μιας συμβολοσειράς σε ηχητικό αρχείο. **Σημείωση**: Το ακόλουθο δείγμα χρησιμοποιεί την εντολή gcloud auth-default print-access-token για να ανακτήσει ένα authorization token για το αίτημα και πρέπει να υπάρχει εγκατεστημένο το gcloud για να εκτελεστεί.

```
curl -H "Authorization: Bearer "
$(gcloud auth application-default print-access-token) -H "Content-Type: application/json; charset=utf-8" --data "{
  'input':{
    'text':'I\'ve added the event to your calendar.'
  },
  'voice':{
    'languageCode':'en-gb',
    'name':'en-GB-Standard-A',
    'ssmlGender':'FEMALE'
  },
  'audioConfig':{
    'audioEncoding':'MP3'
  }
}" "https://texttospeech.googleapis.com/v1/text:synthesize"
```

Όπως παρατηρούμε στο παραπάνω αίτημα, το TTS API δέχεται αιτήματα με παραμέτρους τις παρακάτω πληροφορίες

I. Επιθυμητή γλώσσα
II. Τύπο φωνής
III. Κωδικοποίηση ηχητικού αρχείου
IV. Ταχύτητα αναπαραγωγής
V. Το Κείμενο προς μετατροπή

Στην εικόνα 13 παρουσιάζεται το τμήμα κώδικα με την σύνθεση των παραμέτρων ενός αιτήματος προς το API

```
textToSpeechSettings = TextToSpeechSettings.newBuilder().setCredentialsProvider(
        FixedCredentialsProvider.create(ServiceAccountCredentials.fromStream(
                new FileInputStream(VDHelper.GOOGLE_SERVICE_PROP_PATH))
        ))
        .build();
voice = VoiceSelectionParams.newBuilder()
        .setLanguageCode(VDHelper.GoogleTTSLanguages.GREEK)
        .setSsmlGender(SsmlVoiceGender.FEMALE)
        .build();
audioConfig = AudioConfig.newBuilder()
        .setAudioEncoding(AudioEncoding.MP3)
        .build();
```

Εικόνα 13

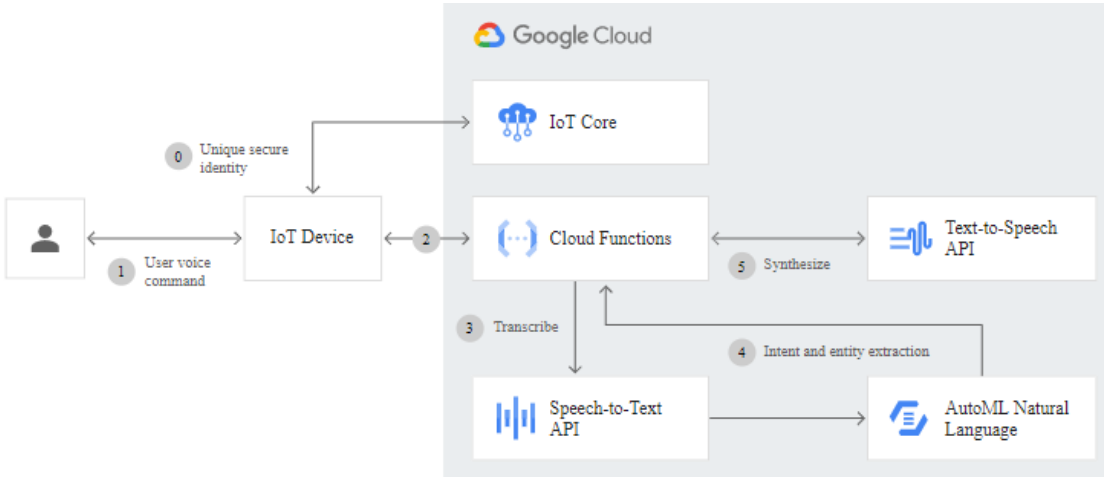
Έπειτα δημιουργούμε ένα στιγμιότυπο της κλάσης TextToSpeechClient η οποία αναλαμβάνει εσωτερικά να αποστείλει το αίτημα μέσω της μεθόδου synthesizeSpeech.

```
try (TextToSpeechClient textToSpeechClient = TextToSpeechClient.create(textToSpeechSettings)) {
    VDHelper.debugLog(className, context.getString(R.string.send_google_cloud_tts_req));
    SynthesisInput input = SynthesisInput.newBuilder().setText(text).build();
    response = textToSpeechClient
            .synthesizeSpeech(input, voice, audioConfig);
    terminated = textToSpeechClient.awaitTermination( duration: 1, TimeUnit.SECONDS);
}
```

Εικόνα 14

Εφόσον η υπηρεσία TTS λάβει το αίτημα στο cloud, επεξεργάζεται τα δεδομένα και παράγει το αντίστοιχο ηχητικό αρχείο, το οποίο αποστέλλει ως απάντηση. Τέλος, το ηχητικό αρχείο αναπαράγεται από την εφαρμογή πολυμέσων της συσκευής Android, και ο χρήστης w το μεταφρασμένο κείμενο.

Η εικόνα 15 που ακολουθεί απεικονίζει τον κύκλο ζωής ενός αιτήματος Text-To-Speech.



Εικόνα 15

5.2.3 ViewModel

Η κλάση ViewModel έχει σχεδιαστεί για να αποθηκεύει και να διαχειρίζεται δεδομένα που σχετίζονται με το περιβάλλον εργασίας χρήστη με τρόπο συνειδητό από τον κύκλο ζωής. Η κλάση ViewModel επιτρέπει στα δεδομένα να διατηρούνται από αλλαγές διαμόρφωσης, όπως περιστροφές οθόνης.

Το Android διαχειρίζεται τους κύκλους ζωής των δραστηριοτήτων (Activities) και των fragments. Συνεπώς, ενδέχεται να αποφασίσει να καταστρέψει ή να δημιουργήσει ξανά έναν ελεγκτή διεπαφής χρήστη (UI Controller) ως απάντηση σε συγκεκριμένες ενέργειες χρήστη ή συμβάντα συσκευών που είναι εκτός ελέγχου.

Εάν το σύστημα καταστρέψει ή δημιουργήσει ξανά έναν ελεγκτή UI, τυχόν προσωρινά δεδομένα που σχετίζονται με το περιβάλλον εργασίας χρήστη που αποθηκεύετε σε αυτά θα χαθούν. Για παράδειγμα, η εφαρμογή μπορεί να περιλαμβάνει την πληροφορία ενός μεταφρασμένου κειμένου σε μία από τις δραστηριότητές της. Όταν η δραστηριότητα ξαναδημιουργηθεί ύστερα από περιστροφή της συσκευής για παράδειγμα, η νέα δραστηριότητα πρέπει να επαναφέρει τις πληροφορίες κειμένου. Για απλά δεδομένα, η δραστηριότητα μπορεί να χρησιμοποιήσει τη μέθοδο onSaveInstanceState() και να επαναφέρει τα δεδομένα της από το πακέτο στην μέ-

θοδο onCreate(), αλλά αυτή η προσέγγιση είναι κατάλληλη μόνο για μικρές ποσότητες δεδομένων που μπορούν να σειριοποιηθούν και όχι για δυνητικά μεγάλες ποσότητες δεδομένων όπως ένα μεγάλο κείμενο ή μια εικόνα bitmap.

Ένα άλλο πρόβλημα είναι ότι οι ελεγκτές διεπαφής χρήστη πρέπει συχνά να πραγματοποιούν ασύγχρονες κλήσεις που ενδέχεται να χρειαστούν λίγο χρόνο για να επιστρέψουν. Ο ελεγκτής διεπαφής χρήστη πρέπει να διαχειριστεί αυτές τις κλήσεις και να διασφαλίσει ότι το σύστημα θα τις καθαρίσει μετά την καταστροφή του για να αποφευχθούν πιθανές διαρροές μνήμης. Αυτή η διαχείριση απαιτεί πολλή συντήρηση και σε περίπτωση που το αντικείμενο ξαναδημιουργηθεί για αλλαγή διαμόρφωσης, είναι σπατάλη πόρων, καθώς το αντικείμενο ενδέχεται να χρειαστεί να επανεκδώσει κλήσεις που έχει ήδη κάνει.

Οι ελεγκτές διεπαφής χρήστη, όπως δραστηριότητες και τα fragments, προορίζονται κυρίως για την εμφάνιση δεδομένων διεπαφής χρήστη, την αντίδραση σε ενέργειες χρήστη ή τη διαχείριση της επικοινωνίας του λειτουργικού συστήματος. Η απαίτηση των ελεγκτών διεπαφής χρήστη, να είναι επίσης υπεύθυνοι για τη φόρτωση δεδομένων από μια βάση δεδομένων ή δίκτυο, προσθέτει επιβάρυνση στην κλάση. Η ανάθεση υπερβολικής ευθύνης στους ελεγκτές διεπαφής χρήστη μπορεί να οδηγήσει σε μία κλάση που προσπαθεί να χειριστεί όλες τις εργασίες μιας εφαρμογής από μόνη της, αντί να μεταβιβάσει την εργασία σε άλλες κλάσεις Η εκχώρηση υπερβολικής ευθύνης στους ελεγκτές διεπαφής χρήστη με αυτόν τον τρόπο καθιστά επίσης πιο δύσκολη τη δοκιμή. Είναι πιο εύκολο και πιο αποτελεσματικό να διαχωρίζονται τα δεδομένα προβολής από τη λογική του ελεγκτή διεπαφής χρήστη.

Το Android παρέχει λύση στο παραπάνω πρόβλημα και ονομάζεται **ViewModel**, η οποία είναι μια βοηθητική κλάση για τον ελεγκτή διεπαφής χρήστη που είναι υπεύθυνος για την προετοιμασία δεδομένων. Τα αντικείμενα ViewModel διατηρούνται αυτόματα κατά τη διάρκεια αλλαγών διαμόρφωσης, έτσι ώστε τα δεδομένα που κατέχουν να είναι άμεσα διαθέσιμα στην επόμενη παρουσία δραστηριότητας ή fragment. Για παράδειγμα, όταν πρέπει να εμφανίσουμε μια εικόνα λήψης στην οθόνη του ViDi, η ευθύνη για την παρουσίαση και την διατήρηση της εικόνας έχει ανατεθεί σε ένα ViewModel, αντί για το ίδιο το fragment, όπως φαίνεται στο παρακάτω απόσπασμα κώδικα:

```
public class SharedViewModel extends AndroidViewModel implements IVDTextOperations{
private final MutableLiveData<VDBitmap> captured = new MutableLiveData<>();
public void setBitmap (VDBitmap vdBitmap){
    setOperationFinished(false);
    captured.setValue(vdBitmap);
    textRecognitionOn(vdBitmap.getBitmapImage());
  }
```

Έπειτα μπορούμε να αποκτήσουμε πρόσβαση στην πληροφορία από το fragment, χρησιμοποιώντας το παρακάτω τμήμα κώδικα

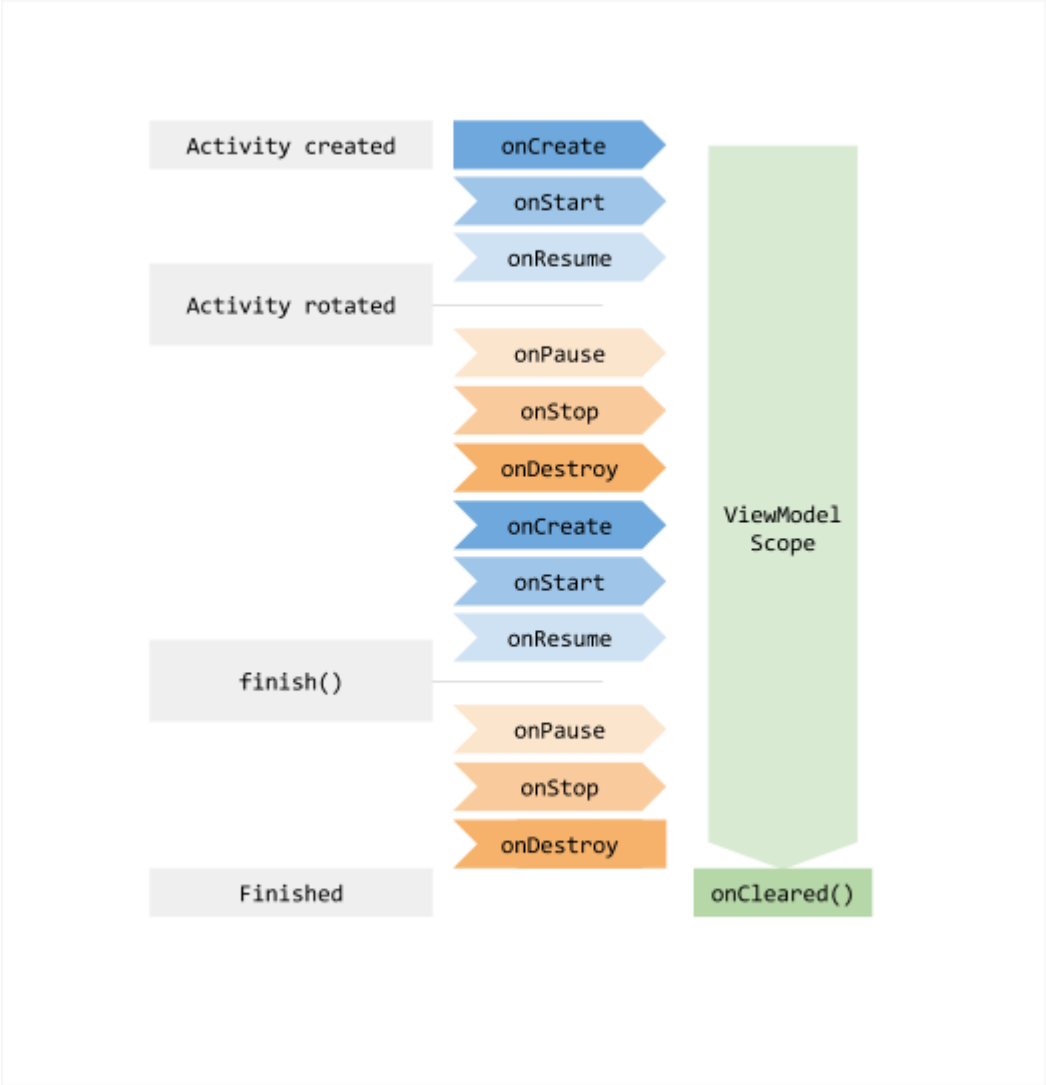
```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    sharedViewModel = new ViewModelProvider(requireActivity()).get(SharedView-
Model.class);
    sharedViewModel.getCaptured().observe(this, vdBitmap ->
        previewImage.setImageBitmap(vdBitmap.rotateBitmap())
    );
```

Εάν η δραστηριότητα ξαναδημιουργηθεί, λαμβάνει την ίδια παρουσία sharedViewModel που δημιουργήθηκε από το πρώτο fragment. Όταν ολοκληρωθεί η δραστηριότητα και κατ' επέκταση το fragment, το πλαίσιο καλεί τη μέθοδο onCleared () των αντικειμένων ViewModel έτσι ώστε να μπορεί να αποδεσμεύσει τους πόρους.

Τα αντικείμενα ViewModel έχουν σχεδιαστεί για να επιβιώνουν ύστερα από συγκεκριμένα γεγονότα. Τα αντικείμενα ViewModel μπορούν να περιέχουν LifecycleObservers, όπως αντικείμενα LiveData. Ωστόσο, δεν πρέπει ποτέ να παρατηρούν αλλαγές σε αντικείμενα που αφορούν τον κύκλο ζωής, όπως τα αντικείμενα LiveData. Εάν το ViewModel χρειάζεται το context της εφαρμογής, για παράδειγμα για να βρει μια υπηρεσία συστήματος, μπορεί να επεκτείνει την κλάση AndroidViewModel το οποίο παρέχει το context της εφαρμογής.

Τα αντικείμενα ViewModel εκτείνονται στον κύκλο ζωής που περνά στο ViewModelProvider κατά τη λήψη του ViewModel. Έτσι παραμένει στη μνήμη έως ότου ο κύκλος ζωής, στα πλαίσια του οποίου εκτείνεται, τερματίζεται ως εξής: στην περίπτωση μιας δραστηριότητας, όταν τελειώσει, ενώ στην περίπτωση ενός fragment, όταν αποσυνδέεται από την δραστηριότητα.

Το Σχήμα 16 απεικονίζει τις διάφορες καταστάσεις κύκλου ζωής μιας δραστηριότητας όταν υποβάλλεται σε περιστροφή και κατά συνέπεια καταστρέφεται. Η εικόνα δείχνει επίσης τη διάρκεια ζωής του ViewModel δίπλα στον κύκλο ζωής της σχετικής δραστηριότητας. Αυτό το συγκεκριμένο διάγραμμα απεικονίζει τις καταστάσεις μιας δραστηριότητας. Οι ίδιες βασικές καταστάσεις ισχύουν και για τον κύκλο ζωής ενός fragment.

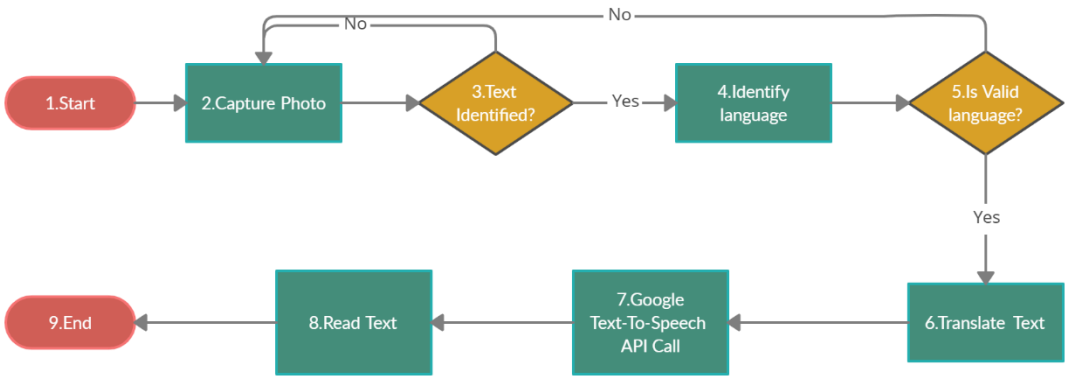


Εικόνα 16

5.3 Λειτουργία της εφαρμογής

Σε αυτό το σημείο θα αναλύσουμε την λειτουργικότητα του ViDi βήμα προς βήμα. Όπως έχουμε αναφέρει παραπάνω, η εφαρμογή λειτουργεί αυτοματοποιημένα. Αυτό σημαίνει ότι από την εκκίνηση της, χωρίς καμία περαιτέρω ενέργεια του χρήστη, ξεκινά να ανιχνεύει κείμενα μέσω της κάμερας με σκοπό να τα μεταφράσει και να τα αναπαράγει ηχητικά.

Το παρακάτω διάγραμμα ροής περιγράφει τα στάδια επαναλαμβανόμενων ενεργειών που πραγματοποιούνται κατά τη λειτουργία της εφαρμογής



Εικόνα 17

1.**Εκκίνηση της ακολουθίας.** Σε αυτό το σημείο ανοίγει η κάμερα της συσκευής.

2.**Λήψη εικόνας.** Η κάμερα πραγματοποιεί λήψη σε x δευτερόλεπτα έπειτα από την εκκίνηση της εφαρμογής.

3. **Αναγνώριση κειμένου.** Σε αυτό το στάδιο εκτελείται ο αλγόριθμος αναγνώρισης κειμένου του Firebase ML Kit (Text Identification) στην εικόνα που έχει ληφθεί. Εάν αναγνωριστούν χαρακτήρες στην εικόνα, τότε η εφαρμογή μεταβαίνει στο επόμενο βήμα. Σε αντίθετη περίπτωση, σταματάει η παρούσα ροή και ξεκινάει από την αρχή με νέα λήψη εικόνας.

4. **Αναγνώριση γλώσσας.** Εκτελείται ο αλγόριθμος αναγνώρισης γλώσσας της Firebase (Language Identification), στο κείμενο που έχει αναγνωριστεί από το προηγούμενο βήμα. Αν για οποιοδήποτε λόγο δεν αναγνωριστεί η γλώσσα κειμένου, ο αλγόριθμος επιστρέφει τον κωδικό “und” (undetermined) και η ροή ξεκινά από την αρχή. Στην περίπτωση που έχουμε κάποιο έγκυρο αποτέλεσμα, ο αλγόριθμος επιστρέφει τον κωδικό της γλώσσας, για παράδειγμα “el” για τα Ελληνικά ή “en” για τα Αγγλικά και η ροή συνεχίζεται.

5.**Έλεγχος εγκυρότητας γλώσσας εισόδου.** Εφόσον έχει αναγνωριστεί η γλώσσα κειμένου, πραγματοποιείται έλεγχος εγκυρότητας. Κατά τον έλεγχο αυτό, εάν η γλώσσα κειμένου είναι ίδια με την επιθυμητή γλώσσα εισόδου που έχει οριστεί στο μενού επιλογών, τότε η ροή συνεχίζεται. Σε διαφορετική περίπτωση η ροή επανεκκινεί.

6. **Μετάφραση κειμένου.** Χρησιμοποιώντας το εργαλείο Μετάφρασης Κειμένου της Firebase (Text Translation), το κείμενο μεταφράζεται στην γλώσσα εξόδου που έχει ρυθμιστεί στο μενού επιλογών και στη συνέχεια επιστρέφει το μεταφρασμένο κείμενο.

7. **Google Text-To-Speech API Call**. Το κείμενο που έχει μεταφραστεί αποστέλλεται σε μορφή αλφαριθμητικού (String) στο Text-To-Speech (TTS) API της Google έτσι ώστε να μετατραπεί σε ηχητικό αρχείο. Εδώ να τονίσουμε ότι το Google TTS API χρησιμοποιήθηκε για τις γλώσσες (πχ Ελληνικά) που δεν υποστηρίζονται από το αντίστοιχο

API του Android. Στη συγκεκριμένη περίπτωση, λαμβάνουμε υπόψη ότι μεταφράζουμε από τα Αγγλικά στα Ελληνικά. Το API της Google μας αποστέλλει απάντηση περιλαμβάνοντας ένα ηχητικό αρχείο που αντιπροσωπεύει το μεταφρασμένο αλφαριθμητικό.

8. **Ηχητική αναπαραγωγή.** Χρησιμοποιώντας το εργαλείο αναπαραγωγής ηχητικών αρχείων της συσκευής, το ViDi διαβάζει το μεταφρασμένο περιεχόμενο του κειμένου. Ουσιαστικά, η εφαρμογή πολυμέσων αναπαράγει το αρχείο MP3 που έχει αποσταλεί από το API της Google.

9. **Ολοκλήρωση της ακολουθίας.** Σε αυτό το σημείο ολοκληρώνεται η ακολουθία ενεργειών και έπειτα η διαδικασία ξεκινάει από την αρχή.

6. Συμπεράσματα και μελλοντικές επεκτάσεις

6.1 Συμπεράσματα

Κατά την εκτεταμένη χρήση της εφαρμογής μπορούμε να διαπιστώσουμε ότι οι αλγόριθμοι του ML Kit έχουν αρκετά μεγάλη ακρίβεια και επιτυγχάνουν να αποφέρουν τα επιθυμητά αποτελέσματα, σε υψηλό βαθμό. Παράλληλα υπάρχει ένα μικρό ποσοστό απόκλισης σε κάποιες περιπτώσεις όπου οι αλγόριθμοι αποτυγχάνουν να επιστρέψουν τα σωστά αποτελέσματα, κυρίως για τα εργαλεία αναγνώρισης κειμένου και αναγνώρισης γλώσσας. Αυτό μπορεί να οφείλεται σε διάφορες παραμέτρους που αναφέρονται παρακάτω.

Προκειμένου το ML Kit να αναγνωρίσει με ακρίβεια το κείμενο, οι εικόνες εισαγωγής πρέπει να περιέχουν κείμενο που αντιπροσωπεύεται από επαρκή δεδομένα pixel. Στην ιδανική περίπτωση, κάθε χαρακτήρας πρέπει να είναι τουλάχιστον 16x16 pixel. Γενικά, δεν υπάρχει όφελος ακρίβειας για χαρακτήρες μεγαλύτερους από 24x24 pixel. Έτσι, για παράδειγμα, μια εικόνα 640x480 μπορεί να λειτουργήσει καλά για τη σάρωση μιας επαγγελματικής κάρτας που καταλαμβάνει το πλήρες πλάτος της εικόνας. Για να σαρώσουμε ένα έγγραφο που εκτυπώνεται σε χαρτί μεγέθους γραμμάτων, ενδέχεται να απαιτείται εικόνα 720x1280 pixel. Η κακή εστίαση της εικόνας μπορεί να επίσης να επηρεάσει την ακρίβεια της αναγνώρισης κειμένου.

Παρατηρήθηκε επίσης ότι ο αλγόριθμος αναγνώρισης γλώσσας μπορεί να επιστρέψει λάθος αποτελέσματα στις περιπτώσεις που το κείμενο είναι γραμμένο με κάποια ιδιαίτερη γραμματοσειρά. Όσο πιο ευδιάκριτοι είναι οι χαρακτήρες της συμβολοσειράς τόσο μεγαλύτερη είναι και η ακρίβεια.

Όσον αφορά το Google TTS API, συμπεραίνουμε πως είναι μια υπηρεσία με πάρα πολλές δυνατότητες, αν αναλογιστούμε την ταχύτητα επεξεργασίας και μετατροπής του κειμένου σε ήχο και γενικά τον χρόνο που χρειάζεται ώστε να απαντήσει στα ερωτήματα της εφαρμογής. Επίσης υπάρχει αρκετά μεγάλη ακρίβεια στα αποτελέσματα που επιστρέφει. Αξίζει να τονίσουμε ότι υποστηρίζει πάνω από 40 γλώσσες και προσφέρει δυνατότητες εξατομίκευσης οι οποίες περιλαμβάνονται ως παράμετροι στις κλήσεις προς το API με αποτέλεσμα να μπορούν να αλλάζουν από κλήση σε κλήση αν το απαιτεί η εφαρμογή. Ορισμένες από αυτές τις παραμέτρους είναι η ταχύτητα αναπαραγωγής του ηχητικού αρχείου, το γένος της φωνής που διαβάζεται το κείμενο, η κωδικοποίηση και άλλα.

6.2 Επεκτάσεις

Όσον αφορά τις μελλοντικές επεκτάσεις του ViDi, υπάρχουν κάποια χαρακτηριστικά τα όποια μπορούν να προστεθούν στην εφαρμογή ώστε να αναβαθμιστεί σε επίπεδο χρηστικότητας.

Πρώτη προσθήκη αφορά την δυνατότητα αλληλεπίδρασης του χρήστη με την εφαρμογή. Συγκεκριμένα, η προσθήκη φωνητικών εντολών οι οποίες θα μπορούν να ελέγξουν πλήρως την συμπεριφορά του ViDi. Αυτό μπορεί να επιτευχθεί με την χρήση Speech-To-Text APIs ώστε ο χρήστης να μπορεί για παράδειγμα να παρεμβάλλει την λειτουργία της, να πραγματοποιεί ρυθμίσεις σε πραγματικό χρόνο καθώς και να απενεργοποιεί την εφαρμογή. Αντίστοιχα για μια ολοκληρωμένη εμπειρία αλληλεπίδρασης, θα είναι εύλογο να ενταχθούν φωνητικές απαντήσεις από πλευράς του ViDi έτσι ώστε να υπάρχει καλύτερη εμπειρία χρήσης της εφαρμογής (User Experience - UX).

Μια ακόμη μελλοντική επέκταση είναι η δημιουργία ενός αλγορίθμου που θα αποφασίζει εάν η εικόνα που σαρώνεται περιέχει πολύτιμα κείμενα και όχι μεμονωμένους χαρακτήρες. Με την παρούσα υλοποίηση, ο αλγόριθμος αναγνώρισης κειμένου λαμβάνει συμβολοσειρές χωρίς να υπάρχει μια προ-επεξεργασία. Αυτό σημαίνει ότι ο αλγόριθμος μπορεί και αναγνωρίζει λέξεις οι οποίες πληρούν όλες τις προϋποθέσεις και είναι μεν έγκυρες αλλά μπορεί να μην χρήζουν μετάφρασης. Για παράδειγμα ένα όνομα χρήστη ή ένας κωδικός χρήστη ή ακόμα η σάρωση χαρακτήρων από ένα πληκτρολόγιο. Γι’ αυτό το λόγο κρίνεται απαραίτητη η ανάπτυξη ενός αλγορίθμου προ-επεξεργασίας που σκοπός του είναι να αποτρέψει τις περιπτώσεις που αναφέρθηκαν παραπάνω. Αυτό το χαρακτηριστικό θα επιφέρει ως αποτέλεσμα την μείωση της άσκοπης εκμετάλλευσης πόρων καθώς και τις περιττές κλήσεις στο Google TTS API.

Ένα ακόμα χρήσιμο χαρακτηριστικό είναι η υποστήριξη μετάφρασης σε περισσότερες γλώσσες. Ο χρήστης θα μπορεί να επιλέξει ανάμεσα σε ένα ικανοποιητικό εύρος γλωσσών τόσο εισόδου όσο και εξόδου. Το χαρακτηριστικό αυτό θα δώσει τη δυνατότητα σε περισσότερους χρήστες να χρησιμοποιήσουν την εφαρμογή ακόμα και από διαφορετικές χώρες. Ένα επιπλέον χαρακτηριστικό θα είναι η δυνατότητα ρύθμισης γλώσσας εισόδου ως «αυτόματη αναγνώριση», όπου το ViDi θα πραγματοποιεί αυτόματη αναγνώρισης της γλώσσας χωρίς να υπάρχει περιορισμός της ρύθμισης σε μία μόνο γλώσσα.

Τέλος, ενδιαφέρουσα προσθήκη θα ήταν η δυνατότητα αποθήκευσης πληροφοριών σχετικά με κάποιο μεταφρασμένο κείμενο. Εφόσον το επιθυμεί ο χρήστης, με την χρήση προκαθορισμένης φωνητικής εντολής, θα έχει τη δυνατότητα να επισημαίνει το κείμενο ως «σημαντικό» και να αποθηκεύει σε μια εσωτερική βάση δεδομένων πληροφορίες που αφορούν

i. αρχικό κείμενο,
ii. μεταφρασμένο κείμενο
iii. ημερομηνία μετάφρασης
iv. τοποθεσία

Κατ' επέκταση, θα δημιουργηθεί μια διεργασία η οποία θα αναλαμβάνει να δέχεται φωνητικές εντολές και να διαβάζει στον χρήστη τις πληροφορίες σχετικά με τις μεταφράσεις που θα έχει αποθηκεύσει/επισημάνει ως σημαντικές. Αυτό το χαρακτηριστικό θα επεκτείνει τις δυνατότητες του ViDi και θα προσθέσει λειτουργίες που θα διευκολύνουν τον χρήστη σε περίπτωση που θελήσει να ανατρέξει στο ιστορικό σημαντικών μεταφράσεων με σκοπό τη συλλογή πληροφοριών. Η ανάπτυξη αυτού του χαρακτηριστικού θα πραγματοποιηθεί με την βοήθεια του Google API Speech-To-Text, όπου μετατρέπει με ακρίβεια την ομιλία σε κείμενο χρησιμοποιώντας ένα API που υποστηρίζεται από τις τεχνολογίες AI της Google, προσφέροντας τα παρακάτω προνόμια

- **Πολύ υψηλή ακρίβεια.** Εφαρμόζει τους πιο προηγμένους αλγόριθμους νευρωνικών δικτύων βαθιάς μάθησης της Google για αυτόματη αναγνώριση ομιλίας (ASR).
- **Παγκόσμια εμβέλεια.** Αναγνώριση φωνής των χρηστών παγκοσμίως, με αναγνώριση φωνής που υποστηρίζει περισσότερες από 125 γλώσσες και παραλλαγές.
- **Ευέλικτη ανάπτυξη.** Ανάπτυξη αναγνώρισης ομιλίας είτε στο cloud με το API είτε στο εσωτερικά, με το Speech-to-Text On-Prem.

7. Βιβλιογραφία

Τεχνητή Νοημοσύνη - Β' Έκδοση
Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Φ. Κόκκορας, Η. Σακελλαρίου

Σχεδίαση Διεπαφής Χρήστη
ebooks.edu.gr/ebooks/

Android Lifecycle
https://developer.android.com/guide/components/activities/activity-lifecycle

Firebase ML Kit Capabilities
https://towardsdatascience.com/ml-kit-for-firebase-features-capabilities-pros-and-cons-a182b4299cc

Machine Learning Kit for Firebase
https://firebase.google.com/docs/ml-kit/

Text-To-Speech: Lifelike Speech Synthesis | Google Cloud
https://cloud.google.com/text-to-speech

Google Services for Android
https://developers.google.com/android/guides/setup

Services in Android
https://www.geeksforgeeks.org/services-in-android

Inter Process Communication (IPC)
https://www.geeksforgeeks.org/inter-process-communication-ipc/

The Top Translation Apps for Android
https://www.ubuntupit.com/best-translator-apps-for-android/

SDK Platform Notes
https://developer.android.com/studio/releases/platforms

Java Mobile
https://www.java.com/en/download/help/java_mobile.html

Android Studio Capabilities
https://developer.android.com/studio

Google Play
https://play.google.com/store

8. Παραρτήματα

8.1 Προβλήματα που παρουσιάστηκαν

8.1.1 Απουσία Ελληνικής Γλώσσας στο Android TTS

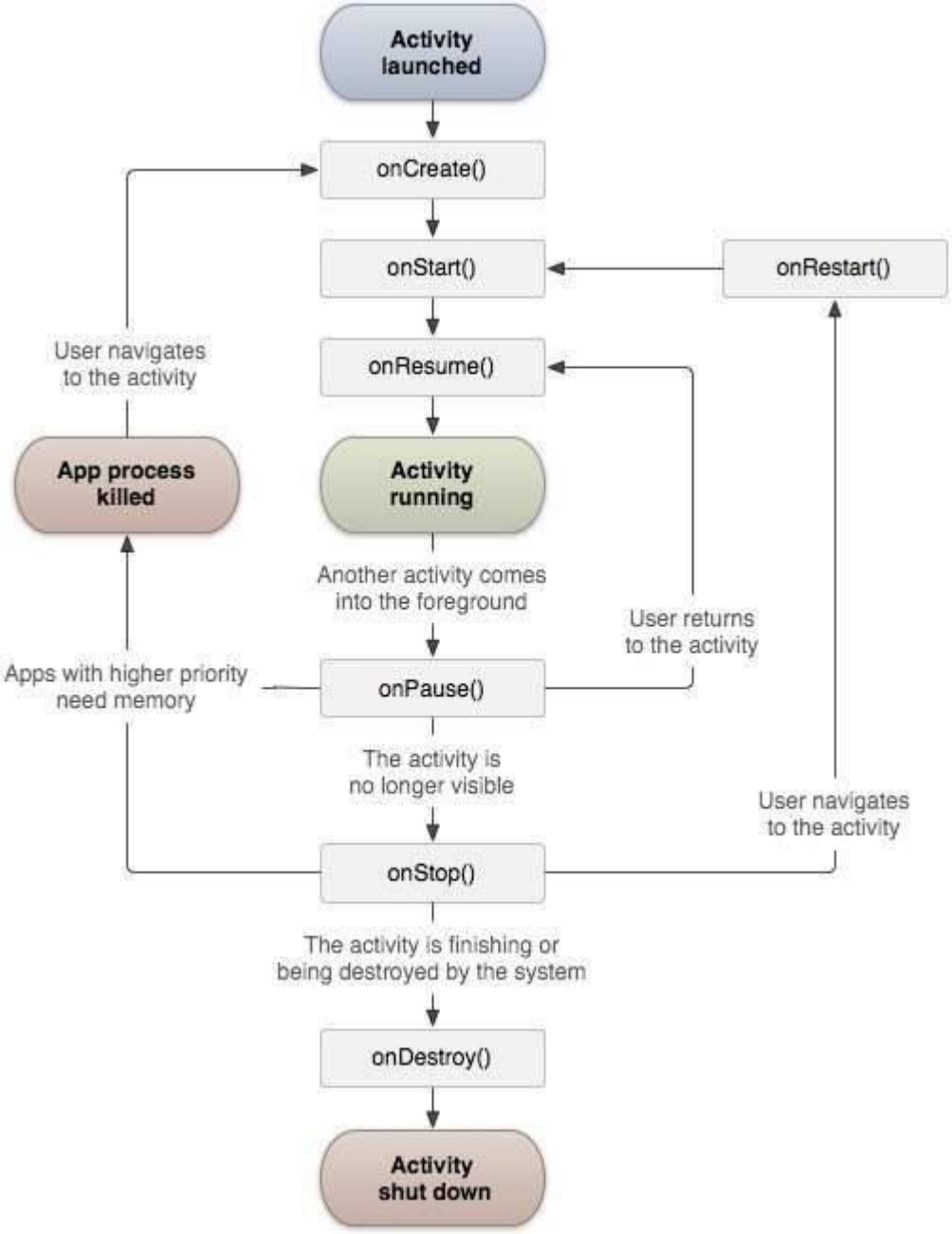
Η ιδέα αυτής της εργασίας ήταν η δημιουργία ενός εργαλείου μετάφρασης για ανθρώπους με προβλήματα όρασης, που αρχικά αφορούσε κυρίως το Ελληνικό κοινό. Από την πρώτη στιγμή ήταν γνωστό ότι το Text-To-Speech εργαλείο που διαθέτει το Android SDK, δεν υποστήριζε την Ελληνική γλώσσα. Επομένως έπρεπε να απευθυνθούμε σε μια υπηρεσία που θα μας παρείχε λύση, ανεξάρτητη από την πλατφόρμα του Android. Η καλύτερη λύση σε αυτό το πρόβλημα ήταν να χρησιμοποιήσουμε το Google TTS API το οποίο ανταποκρίνεται εξαιρετικά στις ανάγκες της εφαρμογής.

8.1.2 Διακοπή ροής κατά την αλλαγή προσανατολισμού

Ένα ακόμη πρόβλημα που παρουσιάστηκε κατά την ανάπτυξη της εφαρμογής ήταν η διακοπή της ροής των ενεργειών κατά την αλλαγή προσανατολισμού / περιστροφή της συσκευής. Για να κατανοήσουμε καλύτερα το πρόβλημα θα πρέπει να αναλύσουμε τις έννοια του κύκλου ζωής της δραστηριότητας (Activity Lifecycle).

Το παρακάτω σχήμα (εικόνα 18) απεικονίζει τον κύκλο ζωής. Το σχήμα καταγράφει τις καταστάσεις και τις μεθόδους επανάκλησης που καλούνται ως μεταβάσεις της εφαρμογής.

Το σύστημα Android ξεκινά το πρόγραμμά του με μια δραστηριότητα ξεκινώντας με μια μέθοδο κλήσης on onCreate (). Υπάρχει μια ακολουθία μεθόδων επανάκλησης που ξεκινούν μια δραστηριότητα και μια ακολουθία μεθόδων επανάκλησης που καταστρέφουν μια δραστηριότητα, όπως φαίνεται στο παρακάτω διάγραμμα κύκλου ζωής δραστηριότητας:



Εικόνα 18

Κατά την αλλαγή προσανατολισμού της συσκευής, όταν για παράδειγμα γυρίσουμε την συσκευή από κάθετα σε οριζόντια ή και το αντίθετο, το λειτουργικό σύστημα του Android καταστρέφει την τρέχουσα δραστηριότητα και την ξαναδημιουργεί.

Σύμφωνα με το παραπάνω σχήμα, μόλις αλλάξει ο προσανατολισμός της συσκευής, το λειτουργικό ξεκινά την ροή διαδικασίας καταστροφής της δραστηριότητας, ξεκινώντας από την κλήση της μεθόδου onPause() και σταδιακά καταλήγει στην μέθοδο onDestroy(). Αυτό για την εφαρμογή ViDi είχε ως αποτέλεσμα να διακόπτεται η διαδικασία μετάφρασης και η διαδικασία αναπαραγωγής του ηχητικού αρχείου, με αποτέλεσμα την απώλεια πληροφορίας.

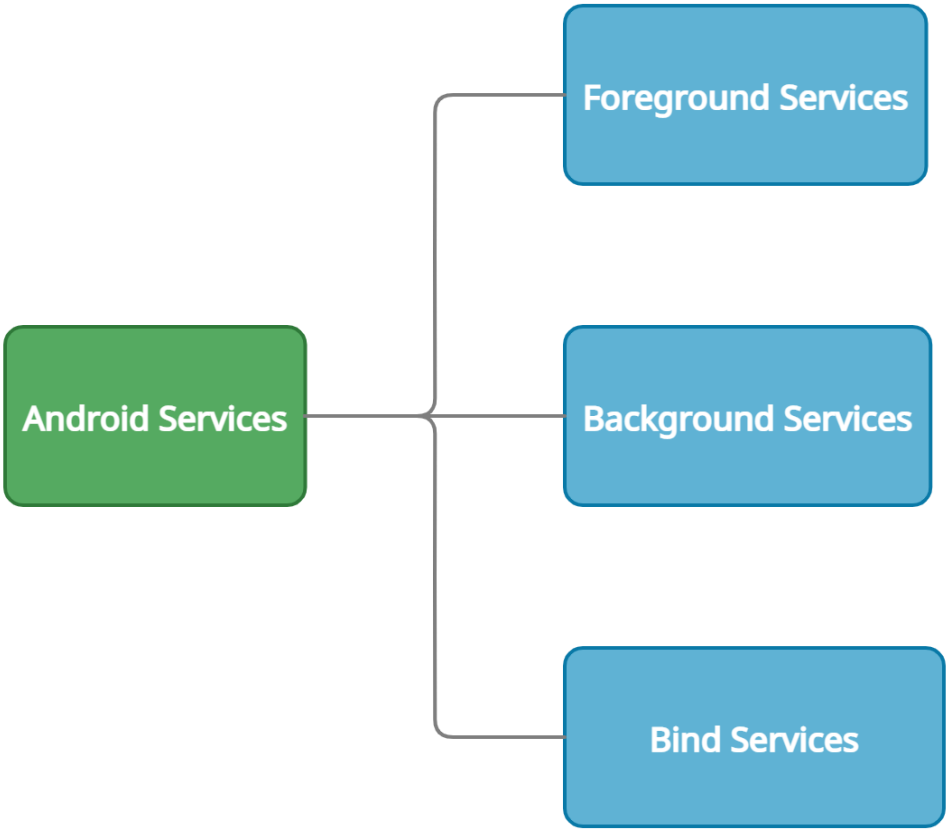
Λύση σε αυτό το πρόβλημα δόθηκε με την χρήση του **Android Service**. Ως εκ τούτου, ένα μέρος της λογικής του ViDi υλοποιήθηκε με χρήση των Services με αποτέλεσμα να λειτουργεί απρόσκοπτα και χωρίς διακοπές σε περιπτώσεις όπως η περιστροφή της συσκευής.

Το Service ή αλλιώς υπηρεσία, είναι ένα εργαλείο του Android API και είναι ένα στοιχείο που εκτελείται στο παρασκήνιο χωρίς άμεση αλληλεπίδραση με τον χρήστη. Δεν έχει διεπαφή χρήστη και δεν δεσμεύεται από τον κύκλο ζωής μιας δραστηριότητας.

Οι υπηρεσίες χρησιμοποιούνται για επαναλαμβανόμενες και δυνητικά μακροχρόνιες λειτουργίες, δηλαδή, λήψεις μέσω Διαδικτύου, ελέγχους για νέα δεδομένα, επεξεργασία δεδομένων, ενημέρωση περιεχομένου, αναπαραγωγή ηχητικών αρχείων και άλλα. Εκτελούνται με υψηλότερη προτεραιότητα από τις ανενεργές ή αόρατες δραστηριότητες και, ως εκ τούτου, το σύστημα Android δεν τις τερματίζει.

Ο πρωταρχικός στόχος μιας υπηρεσίας είναι να διασφαλίσει ότι η εφαρμογή παραμένει ενεργή στο παρασκήνιο, έτσι ώστε ο χρήστης να μπορεί να χειρίζεται πολλές εφαρμογές ταυτόχρονα. Μια υπηρεσία μπορεί να εκτελείται συνεχώς στο παρασκήνιο ακόμα και αν η εφαρμογή είναι κλειστή ή ο χρήστης μεταβαίνει σε άλλη εφαρμογή. Υπάρχει μια μεγάλη διαφορά μεταξύ των **υπηρεσιών** Android και των **νημάτων** και δεν πρέπει κανείς να τις συγχέει. Το νήμα είναι ένα χαρακτηριστικό που παρέχεται από το λειτουργικό σύστημα που επιτρέπει στον χρήστη να εκτελεί λειτουργίες στο παρασκήνιο, ενώ η υπηρεσία (service) είναι ένα στοιχείο του Android που εκτελεί μια μακροχρόνια λειτουργία για την οποία ο χρήστης ενδέχεται να μην γνωρίζει, καθώς δεν διαθέτει διεπαφή χρήστη.

Υπάρχουν τρείς κατηγορίες υπηρεσιών όπως φαίνεται στην εικόνα 19:



Εικόνα 19

Η κατηγορία υπηρεσίας που υλοποιήθηκε για το ViDi είναι τα **Bind Services**. Αυτός ο τύπος υπηρεσίας επιτρέπει στα στοιχεία της εφαρμογής όπως η δραστηριότητα (Activity) να δεσμευτούν με αυτήν. Οι δεσμευμένες υπηρεσίες εκτελούν την αποστολή τους, αρκεί να συνδέεται με οποιοδήποτε στοιχείο της εφαρμογής. Περισσότερα από ένα στοιχεία επιτρέπεται να δεσμεύονται με μια υπηρεσία κάθε φορά.

Για να υλοποιηθεί το Bind Service, απαιτεί την δημιουργία μιας ύπο-κλάσης αυτού ή να χρησιμοποιηθεί μια από της ήδη υπάρχουσες ύπο-κλάσεις. Κατά την υλοποίηση πρέπει να παρακάμψουμε ορισμένες μεθόδους που χειρίζονται βασικές πτυχές του κύκλου ζωής της υπηρεσίας και παρέχουν έναν μηχανισμό που επιτρέπει στα στοιχεία να συνδέονται με την υπηρεσία.

I. **onBind().** Το σύστημα καλεί αυτήν τη μέθοδο καλώντας το bindService () όταν ένα άλλο στοιχείο θέλει να δεσμευτεί με την υπηρεσία (όπως για να εκτελέσει RPC). Κατά την χρήση αυτής της μεθόδου, πρέπει να παρέχουμε μια διεπαφή (Interface) ώστε να χρησιμοποιούν οι clients για να επικοινωνούν με την υπηρεσία επιστρέφοντας ένα IBinder. Στην προκειμένη περίπτωση, θέλαμε να δημιουργήσουμε μια δέσμευση (binding) μεταξύ της διαδικασίας Text-To-

Speech και της υπηρεσίας, έτσι ώστε να μην διακόπτεται η αναπαραγωγή του ηχητικού αρχείου κατά την αλλαγή προσανατολισμού.

```
@Nullable
@Override
public IBinder onBind(Intent intent) {
    sendMessage(intent);
    return vdBinder;
}
```

Εικόνα 20

```
private void sendMessage(Intent intent){
    Message message = serviceHandler.obtainMessage();
    message.setData(intent.getExtras());
    serviceHandler.sendMessage(message);
}
```

Εικόνα 21

II. **onCreate().** Το σύστημα επικαλείται αυτή τη μέθοδο για να εκτελέσει διαδικασίες εφάπαξ ρύθμισης κατά την αρχική δημιουργία της υπηρεσίας (προτού κληθεί η onBind(). Εάν η υπηρεσία εκτελείται ήδη, αυτή η μέθοδος δεν καλείται.

```
@Override
public void onCreate() {
    HandlerThread handlerThread = new HandlerThread("VDServiceStartArguments");
    handlerThread.setPriority(Process.THREAD_PRIORITY_BACKGROUND);
    handlerThread.start();
    Looper serviceLooper = handlerThread.getLooper();
    serviceHandler = new ServiceHandler(serviceLooper);
}
```

Εικόνα 22

III. **onDestroy().**Το σύστημα επικαλείται αυτήν τη μέθοδο όταν η υπηρεσία δεν χρησιμοποιείται πλέον και καταστρέφεται. Η υπηρεσία θα πρέπει να χρησιμοποιήσει αυτή τη μέθοδο για να καθαρίσει τυχόν πόρους όπως νήματα, εγγεγραμμένους ακροατές ή δέκτες. Αυτή είναι η τελευταία κλήση που λαμβάνει η υπηρεσία.

Περαιτέρω, τα στοιχεία της εφαρμογής (Activities, Fragments) μπορούν να αποκτήσουν δέσμευση με την υπηρεσία για τη διεξαγωγή επικοινωνίας μεταξύ διεργασιών (Inter-Process Communication IPC).

Μία διαδικασία διαθέτει δύο τύπους

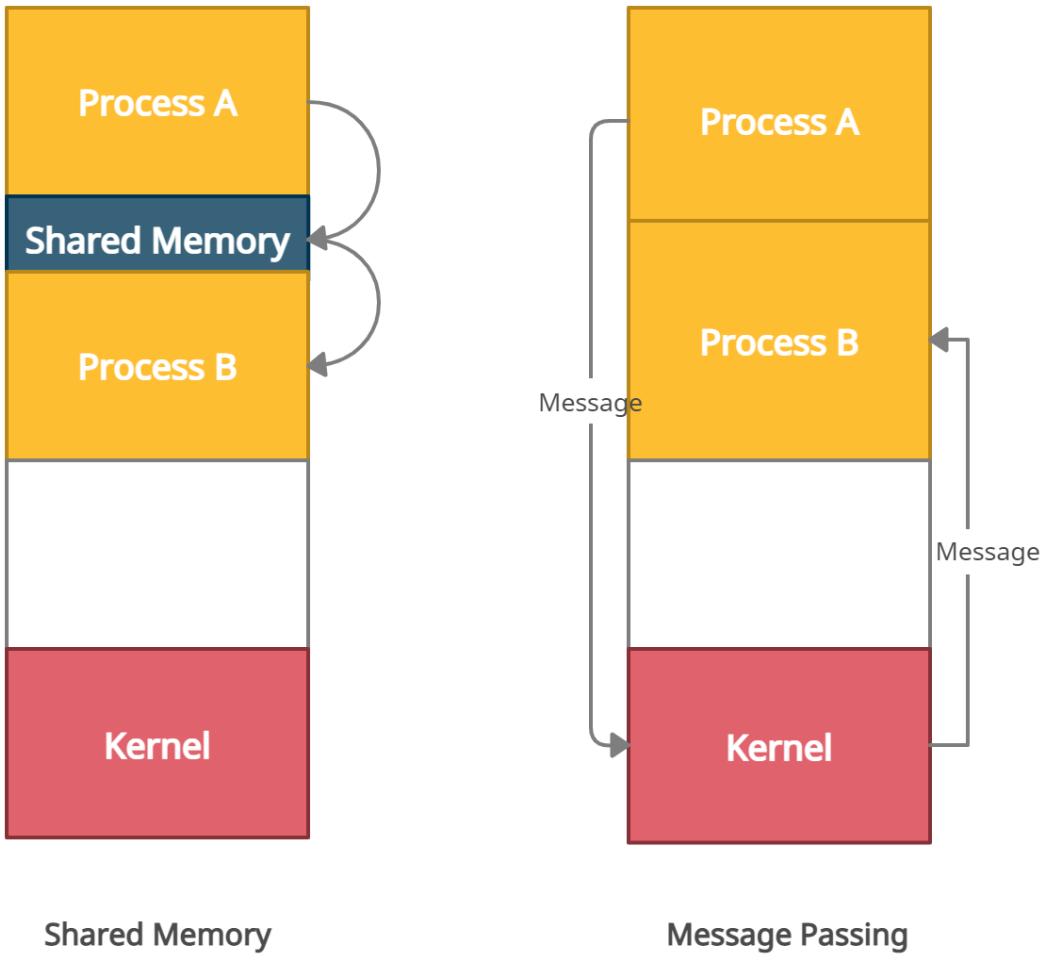
a. Ανεξάρτητη (Independent process)
b. Συνεργασίας (Co-operating process)

Η ανεξάρτητη διαδικασία δεν επηρεάζεται από την εκτέλεση άλλων διαδικασιών, ενώ μια διαδικασία συνεργασίας μπορεί να επηρεαστεί από άλλες διαδικασίες εκτέλεσης. Αν και μπορεί κανείς να σκεφτεί ότι αυτές οι διαδικασίες, οι οποίες εκτελούνται ανεξάρτητα, θα εκτελούνται πολύ αποτελεσματικά, στην πραγματικότητα, υπάρχουν πολλές καταστάσεις όπου η συνεργατική φύση μπορεί να χρησιμοποιηθεί για την αύξηση της υπολογιστικής ταχύτητας. Η επικοινωνία μεταξύ διεργασιών (IPC) είναι ένας μηχανισμός που επιτρέπει στις διαδικασίες να επικοινωνούν μεταξύ τους και να συγχρονίζουν τις ενέργειές τους. Η επικοινωνία μεταξύ αυτών των διαδικασιών μπορεί να θεωρηθεί ως μέθοδος συνεργασίας μεταξύ τους. Οι διαδικασίες μπορούν να επικοινωνούν μεταξύ τους και μέσω:

1. Κοινόχρηστης μνήμης (Shared Memory)
2. Μεταφοράς μηνύματος (Message Passing)

Ένα λειτουργικό σύστημα μπορεί να εφαρμόσει και τις δύο μεθόδους επικοινωνίας. Πρώτον, θα αναφέρουμε τις μεθόδους επικοινωνίας της κοινής μνήμης και μετά τη μετάδοση μηνυμάτων. Η επικοινωνία μεταξύ διεργασιών που χρησιμοποιούν κοινόχρηστη μνήμη απαιτεί διαδικασίες για κοινή χρήση κάποιας μεταβλητής και εξαρτάται πλήρως από τον τρόπο με τον οποίο ο προγραμματιστής θα την εφαρμόσει. Ένας τρόπος επικοινωνίας που χρησιμοποιεί κοινόχρηστη μνήμη μπορεί να περιγραφθεί ως εξής: Ας υποθέσουμε ότι η διαδικασία1 και η διαδικασία2 εκτελούνται ταυτόχρονα και μοιράζονται ορισμένους πόρους ή χρησιμοποιούν κάποιες πληροφορίες από άλλη διαδικασία. Η διαδικασία1 δημιουργεί πληροφορίες σχετικά με ορισμένους υπολογισμούς ή πόρους που χρησιμοποιούνται και τις διατηρεί ως αρχείο σε κοινόχρηστη μνήμη. Όταν η διαδικασία2 πρέπει να χρησιμοποιήσει τις κοινόχρηστες πληροφορίες, θα ελέγξει την εγγραφή που είναι αποθηκευμένη στην κοινόχρηστη μνήμη και θα λάβει υπόψη τις πληροφορίες που δημιουργούνται από τη διαδικασία1 και θα ενεργήσει ανάλογα. Οι διαδικασίες μπορούν να χρησιμοποιούν κοινόχρηστη μνήμη για την εξαγωγή πληροφοριών ως εγγραφή από άλλη διαδικασία καθώς και για την παράδοση συγκεκριμένων πληροφοριών σε άλλες διαδικασίες.

Στην εικόνα 23 παρατηρούμε την επικοινωνία μεταξύ διεργασιών χρησιμοποιώντας και τις δυο μεθόδους επικοινωνίας.



Εικόνα 23

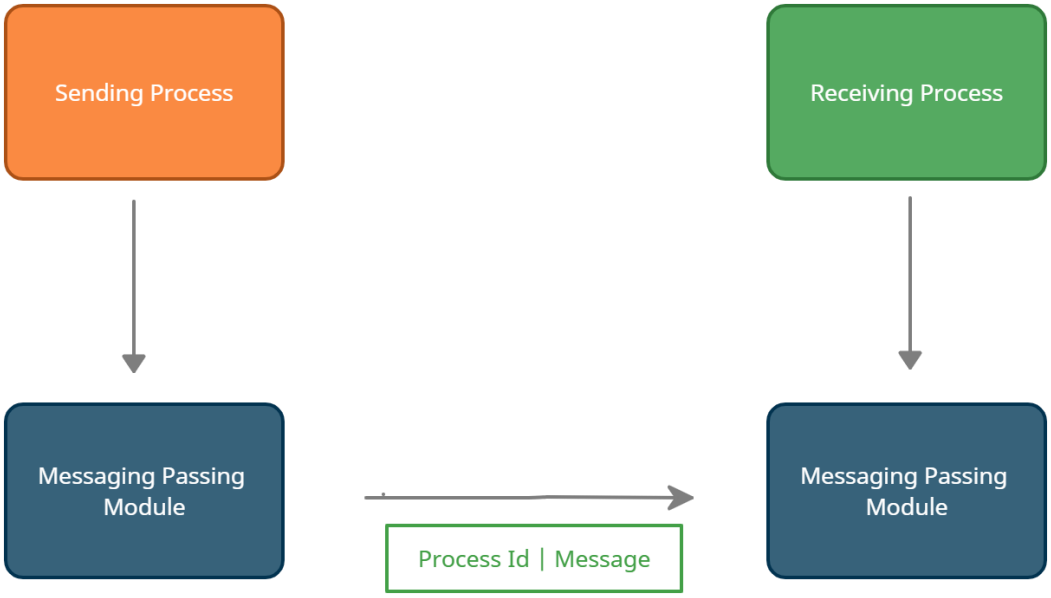
Μέθοδος Κοινόχρηστης Μνήμης - Shared Memory Method

Υπάρχουν δύο διαδικασίες: Παραγωγός και Καταναλωτής. Ο παραγωγός παράγει κάποιο είδος και ο καταναλωτής καταναλώνει αυτό το είδος. Οι δύο διαδικασίες μοιράζονται έναν κοινό χώρο ή μια θέση μνήμης γνωστή ως buffer όπου αποθηκεύεται το αντικείμενο που παράγεται από τον παραγωγό και από το οποίο ο καταναλωτής καταναλώνει το στοιχείο, εάν χρειαστεί. Υπάρχουν δύο εκδόσεις αυτού του προβλήματος: η πρώτη είναι γνωστή ως πρόβλημα περιορισμένου buffer στο οποίο ο Παραγωγός μπορεί να συνεχίσει να παράγει αντικείμενα και δεν υπάρχει όριο στο μέγεθος του buffer, η δεύτερη είναι γνωστή ως το πρόβλημα οριοθετημένου buffer στο οποίο ο Παραγωγός μπορεί να παράγει έως ένα συγκεκριμένο αριθμό αντικειμένων πριν αρχίσει να περιμένει τον καταναλωτή να το καταναλώσει.

Μέθοδος Μεταφοράς Μηνύματος – Message Passing Method

Η συγκεκριμένη μέθοδος υλοποιήθηκε για την ενσωμάτωση των services στην εφαρμογή ViDi (Εικόνες 20 , 21). Εδώ οι διεργασίες επικοινωνούν μεταξύ τους χωρίς να χρησιμοποιούν οποιοδήποτε είδος κοινόχρηστης μνήμης. Εάν δύο διαδικασίες p1 και p2 θέλουν να επικοινωνήσουν μεταξύ τους, προχωρούν ως εξής (Εικόνα 22):

- Δημιουργία συνδέσμου επικοινωνίας (εάν υπάρχει ήδη σύνδεσμος, δεν χρειάζεται να τον δημιουργήσουμε ξανά.)
- Ξεκινά η ανταλλαγή μηνυμάτων χρησιμοποιώντας βασικούς πρωτόγονους τύπους.



Εικόνα 24

Το μέγεθος του μηνύματος μπορεί να είναι σταθερού μεγέθους ή μεταβλητού μεγέθους. Ένα τυπικό μήνυμα μπορεί να έχει δύο μέρη: κεφαλίδα και σώμα.

Το τμήμα κεφαλίδας χρησιμοποιείται για την αποθήκευση τύπου μηνύματος, αναγνωριστικού προορισμού, αναγνωριστικού πηγής, μήκους μηνύματος και πληροφοριών ελέγχου. Οι πληροφορίες ελέγχου περιέχουν πληροφορίες όπως τι πρέπει να κάνετε εάν εξαντληθεί ο χώρος αποθήκευσης, ο αριθμός ακολουθίας, η προτεραιότητα. Γενικά, το μήνυμα αποστέλλεται χρησιμοποιώντας στυλ FIFO.

8.1.3 Μέγεθος μοντέλου μετάφρασης

Όπως έχουμε αναφέρει, το εργαλείο μετάφρασης του ML Kit λειτουργεί με την βοήθεια κάποιων μοντέλων, τα οποία διαθέτουν όλη την πληροφορία για την διαδικασία μετάφρασης από γλώσσα σε γλώσσα. Σύμφωνα με την εικόνα 8, πριν τη μετάφραση, πραγματοποιείται έλεγχος διαθεσιμότητας μοντέλου και αν αυτό απουσιάζει από την συσκευή, γίνεται λήψη του. Η διαδικασία αυτή λαμβάνει χώρα κυρίως κατά την πρώτη φορά που πραγματοποιείται μετάφραση στη συγκεκριμένη γλώσσα, αλλά επειδή τα μοντέλα καταλαμβάνουν μεγάλη χωρητικότητα (35MB για Ελληνικά), έχει ως αποτέλεσμα την μικρή διακοπή της ροής διαδικασίας μετάφρασης, η οποία δίνει την αίσθηση στον χρήστη ότι η υπηρεσία δεν αποκρίνεται.

Λύση σε αυτό το πρόβλημα δόθηκε με την λήψη του απαραίτητου μοντέλου μετάφρασης κατά την πρώτη εκκίνηση της εφαρμογής. Επίσης, όταν επιλέγεται διαφορετικό μοντέλο από τον χρήστη στο μενού επιλογών, η εφαρμογή πραγματοποιεί άμεσα την λήψη του μοντέλου που αντιστοιχεί στην γλώσσα που επιλέχθηκε.

```
FirebaseModelManager modelManager = FirebaseModelManager.getInstance();

// Get translation models stored on the device.
modelManager.getDownloadedModels(FirebaseTranslateRemoteModel.class)
        .addOnSuccessListener(new OnSuccessListener<Set<FirebaseTranslateRemoteModel>>() {
            @Override
            public void onSuccess(Set<FirebaseTranslateRemoteModel> models) {
                // ...
            }
        })
        .addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                // Error.
            }
        });

// Delete the German model if it's on the device.
FirebaseTranslateRemoteModel deModel =
        new FirebaseTranslateRemoteModel.Builder(FirebaseTranslateLanguage.DE).build();
modelManager.deleteDownloadedModel(deModel)
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void v) {
                // Model deleted.
            }
        })
        .addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                // Error.
            }
        });
```

Εικόνα 25

Η υλοποίηση αυτής της λύσης πραγματοποιήθηκε χρησιμοποιώντας το API διαχείρισης μοντέλων μετάφρασης του ML Kit, όπου μας δίνει την δυνατότητα να διαχειριστούμε ρητά τα μοντέλα μετάφρασης που θέλουμε να είναι διαθέσιμα στην συσκευή. Το συγκεκριμένο API χρησιμοποιείται στις περιπτώσεις που θέλουμε να κάνουμε λήψη μοντέλων εκ των προτέρων ή να διαγράψουμε μη απαραίτητα μοντέλα από την συσκευή.

Η εικόνα 23 μας παρουσιάζει τον τρόπο με τον οποίο γίνεται η υλοποίηση του API διαχείρισης των μοντέλων μετάφρασης.

8.1.4 Χρήση της συσκευής Android

Ένα πρόβλημα που προέκυψε κατά τη σύλληψη της ιδέας, ήταν η τοποθέτηση της συσκευής κατά τη χρήση της εφαρμογής. Όπως καταλαβαίνουμε, δεν είναι πρακτικό ο χρήστης να κρατάει την συσκευή στο χέρι. Η βέλτιστη λύση στο πρόβλημα ήταν να χρησιμοποιηθεί η συσκευή ως περιδέραιο. Με λίγα λόγια, ο χρήστης φοράει στο λαιμό του μια αλυσίδα από την οποία κρέμεται το κινητό. Με αυτό τον τρόπο η κάμερα εστιάζει σε κείμενα που ενδεχομένως να βρίσκονται στην ίδια κατεύθυνση κατά την οποία κινείται ο χρήστης. Συνεπώς απλοποιείται η εμπειρία χρήσης της εφαρμογής καθώς η συσκευή βρίσκεται πάνω στον χρήστη και η εφαρμογή λειτουργεί αυτοματοποιημένα.

8.2 Περιορισμοί της εφαρμογής ViDi

8.2.1 Διακοπή υπηρεσιών TTS κατά την απώλεια σήματος

Θεμελιώδες χαρακτηριστικό της εφαρμογής είναι η λειτουργία Text-To-Speech. Όπως αναφέραμε, η εφαρμογή χρησιμοποιεί το API της Google για τις γλώσσες που δεν υποστηρίζονται απο την ενσωματωμένη βιβλιοθήκη του Android. Για το λόγο αυτό, η σύνδεση στο διαδίκτυο είναι απαραίτητη. Στις περιπτώσεις που το σήμα δικτύου είναι ασθενές, η ροή της εφαρμογής διακόπτεται και η εφαρμογή θεωρητικά αποτυγχάνει να ολοκληρώσει την διαδικασία μετάφρασης, για παράδειγμα στην Ελληνική γλώσσα.

Μία εναλλακτική λύση που εφαρμόστηκε για αυτές τις περιπτώσεις, είναι ο έλεγχος δικτύου πριν τη χρήση της υπηρεσίας της Google. Στην περίπτωση που η συνδεσιμότητα έχει διακοπεί ή το σήμα είναι ασθενές, η εφαρμογή ViDi χρησιμοποιεί το Text-To-Speech εργαλείο που παρέχεται από το λειτουργικό και ενημερώνει τον χρήστη ότι υπάρχει πρόβλημα δικτύου.