ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ, ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΑΠΤΥΞΗ ΒΙΒΛΙΟΘΗΚΗΣ ΠΟΥ ΕΠΙΤΡΕΠΕΙ ΤΟΝ ΕΛΕΓΧΟ ΕΦΑΡΜΟΓΩΝ ΜΕΣΩ ΦΩΝΗΤΙΚΩΝ ΕΝΤΟΛΩΝ

Πτυχιακή εργασία του

Πετρόπουλος Ευάγγελος (3785)

Επιβλέπων: Ν. Πεταλίδης

ΣΕΡΡΕΣ, ΟΚΤΩΒΡΙΟΣ 2020

Υπεύθυνη δήλωση

Υπεύθυνη Δήλωση: Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής, Υπολογιστών και Τηλεπικοινωνιών του Διεθνούς Πανεπιστημίου της Ελλάδος

Σύνοψη

Οι μελισσοκόμοι καταγράφουν σημειώσεις για τα μελίσσια που ελέγχουν αλλά πολλές φορές δεν προλαβαίνουν επειδή ο αριθμός των μελισσιών είναι μεγάλος ή δουλεύουν μόνοι τους και πρέπει να σταματάνε την εργασία για να καταγράψουν τις σημειώσεις.Η εφαρμογή που αναπτύξαμε και ονομάζεται ConApi (Contemporary Apiculture) υλοποιεί ένα σύστημα φωνητικών εντολών που βοηθάνε τον μελισσοκόμο στην καταγραφή των σημειώσεων χωρίς την χρήση της συσκευής με τα χέρια παρά μόνο της φωνής.

Περιεχόμενα

Κατάλογος πινάκων

Κατάλογος διαγραμμάτων

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την οικογένεια μου, συγκεκριμένα τους γονείς μου Κωνσταντίνο και Μαρία, τον αδερφό μου Σπύρο και την αδερφή μου Ειρήνη για την στήριξη όλων αυτών των χρόνων. Επίσης ευχαριστώ θερμά τον επιβλέποντα καθηγητή μου, κύριο Νίκο Πεταλίδη, για όλες τις συμβουλές και την καθοδήγηση που μου παρείχε, όχι μόνο κατά την διάρκεια εκπόνησης της εργασίας αλλά και πριν. Τέλος, ένα μεγάλο ευχαριστώ στην κοπέλα μου Λία που με στήριζε από την αρχή των σπουδών μας και στην εκπλήρωση της εργασίας.

Κεφάλαιο 1

Εισαγωγή

Η μελισσοκομία είναι μια επιστήμη που οι άνθρωποι ασχολούνται από τα αρχαία χρόνια και μία απο τις δουλειές είναι ο τακτικός έλεγχος για την πρόοδο των μελισσιών. Σε κάθε έλεγχο του μελισσιού καταγράφουν χειρόγραφες σημειώσεις για την πρόοδο του. Οι σημειώσεις είτε γράφονται παράλληλα με τον έλεγχο των μελλισιών είτε μέτα το πέρας του έλεγχου.

Δυσκολίες συναντιούνται με την καταγραφή των χειρόγραφων σημειώσεων, όπως στο μεγάλο αριθμό μελισσιών ο μελισσοκόμος να μην προλαβαίνει να καταγράφει τις παρατηρήσεις του απο τον έλεγχο, να ξεχνάει τι είχε παρατηρήσει στα μελίσσια στην αρχή του ελέγχου. Επιπλέον ο μελισσοκόμος φοράει ειδική στολή (μάσκα, γάντια) που καθιστούν δύσκολη την καταγραφή των παρατηρήσεων όπως με την μάσκα δεν θα βλέπει καλά λόγο της σίτας που έχει, επίσης με τα γάντια υπάρχει δυσκόλια στο να κρατήσει το μολύβι, το τετράδιο για να καταγράψει τις σημειώσεις.

Για τους παραπάνω λόγους είναι ενδιαφέρουσα η ανάπτυξη ενός εργαλείου το οποίο θα λύνει τα χέρια του μελισσοκόμου χρησιμοποιώντας μόνο την φωνή του για την καταγραφή των παρατηρήσεων.

Σκοπός αυτής της εργασίας είναι η ανάπτυξη ένος τέτοιου εργαλείου που θα υποστηρίζει φωνητικές εντολές για την ευκολία καταγραφής των παρατηρήσεων.

1.1 Δομή της εργασίας

Κεφάλαιο 2 Παρουσίαση έλεγχου μελισσιών και χειρόγραφες σημειώσεις.

Κεφάλαιο 3 Εφαρμογή και φωνητικές εντολές.

Κεφάλαιο 2

Έλεγχος μελισσιών και χειρόγραφες σημειώσεις

2.1 Έλεγχος

Ο έλεγχος των μελισσιών πραγματοποιείται όλες τις εποχές του χρόνου. Ο μελισσοκόμος μετά απο τον Χειμώνα, στην αρχή της Άνοιξης θα κάνει τον πρώτο έλεγχο των μελισσιών που θα κοιτάξει αν η βασίλισσα του μελισσιού είναι ζωντανή και αν γεννάει γόνο. Στα μέσα της Άνοιξης θα ελέγξει αν ο γόνος αυξήθηκε, αν είναι συμπαγής και πόσα πλαίσια γόνου έχει το μελίσσι.

Την επόμενη εποχή, το καλοκαίρι με τους ελέγχους που θα κάνει θα παρατηρήσει αν τα μελίσσια συλλέγουν μέλι και αν η βασίλισσα σταμάτησε να γεννάει γόνο. Προς το τέλος του καλοκαιριού θέλει να δεί αν έχουν συλλέξει αρκετό μέλι για να γίνει ο τρύγος.

Το Φθινόπωρο γίνεται έλεγχος για να διακρίνει αν η βασίλισσα γεννάει συμπαγή γόνο και το Χειμώνα αν τα μελίσσια έχουν ασθένειες.

2.2 Χειρόγραφες σημειώσεις

Οι χειρόγραφες σημειώσεις του μελισσοκόμου αποτελούνται από τον αριθμό της κυψέλης και την ηλικία της βασίλισσας. Επιπλέον, σε κάθε έλεγχο καταγράφεται η ημερομηνία, πόσα πλαίσια έχει η κυψέλη και από τα οποία πόσα έχουν πληθυσμό. Επίσης, καταγράφεται απο τα πλαίσια πόσα έχουν γόνο, μέλι και γύρη. Τέλος, σημειώνονται

γενικές παρατηρήσεις/υπενθυμίσεις της κυψέλης.

Πίνακας 2.1: Ημερολόγιο Μελισσοκόμου.

Αριθμός Κυψέλης: 6			Ηλικία Βασίλισσας: 05-2019			
Ημερομηνία	Πλαίσια	Πληθυσμός	Γόνος	Μέλι	Γύρη	Παρατηρήσεις
09-04-2019	10	7	4			ΟΚ
18-04-2019	10	10	7			Θέλει όροφο
24-04-2019	15	10	8			2 πλαίσια έχουν βασιλικά κελιά, μπήκε όροφος
03-05-2019	15	10	8			Έκοψα παραφιάδα Ν10

2.3 Δυσκολίες

Οι μελισσοκόμοι συναντάνε δυσκολίες στο να κρατάνε τις σημειώσεις τους. Μια απο αυτές τις δυσκολίες είναι η στολή που χρησιμοποιούν, η οποία αποτελείται απο την μάσκα, την φόρμα και τα γάντια. Η μάσκα έχει τούλι στο να επιτρέπει τον μελισσοκόμο να βλέπει και να εμποδίζει τις μέλισσες να εισέλθουν μέσα στην μάσκα. Το τούλι της μάσκας δυσκολεύει την διαδικάσια καταγραφείς των σημειώσεων διότι εμποδίζει στο να βλέπεις καθαρά το τετράδιο. Τα μελισσοκομικά γάντια αποτρέπουν τις μέλισσες από τσιμπήματα στα χέρια αλλά δυσκολεύουν στο κράτημα του μολυβιού ώστε να γραφτούν οι σημειώσεις στο τετράδιο.

Κεφάλαιο 3

Εφαρμογή και φωνητικές εντολές

3.1 Εφαρμογή

Η εφαρμογή που υλοποιείται σε αυτή την εργασία δύναται να λύσει τις δυσκολίες που αντιμετωπίζει ο μελισσοκόμος.Όπως αναφέραμε παραπάνω η δυσκολία που συναντάει ο μελισσοκόμος είναι η καταγραφή των σημειώσεων που είναι σημαντικές για την πρόοδο των μελισσιών. Η εφαρμογή κρατάει τα δεδομένα όπως παρουσιάζονται στο παραπάνω πίνακα 2.1. Συγκεκριμένα, οι εγγραφές για όλα τα μελίσσια που δουλεύει ο μελισσοκόμος.
Για κάθε μελίσσι τα δεδομένα είναι:

- Αριθμός Κυψέλης
- Ηλικία Βασίλισσας
- Ημερομηνία (κάθε ελέγχου)
- Πλαίσια (αριθμός πλαισιων κυψέλης)
- Πληθυσμός κυψέλης (αριθμός πλαίσιων με μέλισσες)
- Γόνος (αριθμός πλαίσιων με γόνο)
- Μέλι (αριθμός πλαίσιων με μέλι)
- Γύρη (αριθμός πλαίσιων με γύρη)
- Παρατηρήσεις

Η εφαρμογή με την υλοποιήση ενός γραφικού περιβάλλοντος για την προβολή και επεξεργασία των μελισσοκομικών δεδομένων και την υλοποιήση μιας διεπαφής χρήστη φωνής με φωνητικές εντολές για την καταχώρηση των μελισσοκομικών δεδομένων λύνοντας τις δυσκολίες που αντιμετωπίζει ο μελισσοκόμος κατά τη διάρκεια της εργασίας.

3.2 GUI-Γραφικό περιβάλλον διεπαφής χρήστη

Το γραφικό περιβάλλον της εφαρμογής θα παρουσίαζει με ωραία γραφικά τα μελίσσια και κάθε πληροφορία του μελισσιού και θα δίνει δυνατότητα στον χρήστη να επεξεργαστεί αυτή την πληροφορία.

3.3 Φωνητικές Εντολές

Δυο κατηγορίες φωνητικών εντολών υποστηρίζονται, οι οποίες είναι:

1. Μελισσιού
2. Έλεγχος μελισσιού

Οι φωνητικές εντολές κατά τον έλεγχο του μελισσιού θα ενεργοποιούνται όταν επιλεχθεί το μελίσσι για έλεγχο. Οι φωνητικές εντολές της εφαρμογής είναι:

Πίνακας 3.1: Φωνητικές εντολές

Φωνητικές Εντολές	Περιγραφή	Παράδειγμα
Μελίσσι <αριθμός>	Στο πεδίο <αριθμός> θα λέγεται ο αριθμός του μελισσιού	Μελίσσι 6
Πλαίσια <αριθμός>	Στο πεδίο <αριθμός> θα λέγεται ο αριθμός των πλαισιων του μελισσιού	Πλαίσια 10
Πληθυσμός <αριθμός>	Στο πεδίο <αριθμός> θα λέγεται ο αριθμός των πλαισιων με πληθυσμό του μελισσιού	Πληθυσμός 7
Γόνος <αριθμός>	Στο πεδίο <αριθμός> θα λέγεται ο αριθμός των πλαισιων με γόνο του μελισσιού	Γόνος 4
Μέλι <αριθμός>	Στο πεδίο <αριθμός> θα λέγεται ο αριθμός των πλαισιων με μέλι του μελισσιού	Μέλι 3
Γύρη <αριθμός>	Στο πεδίο <αριθμός> θα λέγεται ο αριθμός των πλαισιων με γύρη του μελισσιού	Γύρη 3
Παρατήρηση <κείμενο>	Στο πεδίο <κείμενο> θα λέγεται η παρατήρηση για το μελίσσι	Παρατήρηση Θέλει όροφο

Κεφάλαιο 4

Ανασκόπηση Speech-to-Text APIs

4.1 APIs

4.1.1 Google Could Speech-to-Text API

4.1.1.1 Περιγραφή

Είναι ένα API μεταγραφής ομιλίας σε κείμενο υποστηριζοντάς πολλές γλώσσες προγραμματισμού, γεγονός που το καθιστά ανεξάρτητο πλατφόρμας. Το API που υποστηρίζεται από τις τεχνολογίες AI της Google και έχει την δυνατότητα χρήσης διαφορετικών μοντέλων μηχανικής εκμάθησης για αιτήματα μεταγραφής ήχου σε Speech-to-Text.

4.1.1.2 Δυνατότητες

- **Παγκόσμιο λεξιλόγιο** - Εκτεταμένη υποστήριξη γλώσσας ομιλίας σε κείμενο σε περισσότερες από 125 γλώσσες και παραλλαγές.
- **Ροή αναγνώριση ομιλίας** - Αποτελέσματα αναγνώρισης ομιλίας σε πραγματικό χρόνο καθώς το API επεξεργάζεται την είσοδο ήχου που μεταδίδεται σε ροή από το μικρόφωνο της εφαρμογής σας ή αποστέλλεται από ένα προκαθορισμένο αρχείο ήχου (inline ή μέσω Cloud Storage).
- **Προσαρμογή ομιλίας** - Προσαρμογή της αναγνώρισης ομιλίας για την μετατροπή ορών για συγκεκριμένους τομείς και σπάνιες λέξεις παρέχοντας συμβουλές και ενισχύση της ακριβής μεταγραφής συγκεκριμένων λέξεων ή φράσεων.

Αυτόματη μετατροπή προφορικών αριθμών σε διευθύνσεις, έτη, νομίσματα και άλλα χρησιμοποιώντας τάξεις.

- **Πολυκαναλική αναγνώριση** - Το Speech-to-Text μπορεί να αναγνωρίσει διαφορετικά κανάλια σε καταστάσεις πολλαπλών καναλιών (π.χ., τηλεδιάσκεψη) και να σχολιάσει τις μεταγραφές για τη διατήρηση της τάξης.

- **Ανθεκτικότητα θορύβου** - Το Speech-to-Text μπορεί να χειριστεί θορυβώδη ήχο από πολλά περιβάλλοντα χωρίς να απαιτείται επιπλέον ακύρωση θορύβου.

- **Μοντέλα ειδικά για τομέα** - Επιλέξτε από μια επιλογή εκπαιδευμένων μοντέλων για φωνητικό έλεγχο και τηλεφωνική κλήση και μεταγραφή βίντεο βελτιστοποιημένη για απαιτήσεις ποιότητας για συγκεκριμένο τομέα. Για παράδειγμα, το βελτιωμένο μοντέλο τηλεφωνικών κλήσεων είναι συντονισμένο για ήχο που προέρχεται από την τηλεφωνία, όπως οι τηλεφωνικές κλήσεις που έχουν εγγραφεί σε ρυθμό δειγματοληψίας 8khz.

- Υποστηρίζει γλώσσες προγραμματισμού όπως Protocol, C#, Go, java, Node.js, PHP, Python και Ruby.

Λαμβάνοντας υπόψη ότι η Google είναι ουσιαστικά το νευρικό σύστημα του Διαδικτύου σε αυτό το σημείο, δεν αποτελεί έκπληξη ότι το API ομιλίας σε κείμενο είναι ένα από τα πιο δημοφιλή - και πιο ισχυρά - API που είναι διαθέσιμα στους προγραμματιστές.

Το Google Speech-To-Text παρουσιάστηκε το 2018, μόλις μία εβδομάδα μετά την ενημέρωση κειμένου σε ομιλία. Το API Speech-To-Text της Google προβάλλει ορισμένες τολμηρές αξιώσεις, μειώνοντας τα λάθη λέξεων κατά 54% σε δοκιμή μετά τη δοκιμή. Σε ορισμένους τομείς, τα αποτελέσματα είναι ακόμη πιο ενθαρρυντικά.

Ένας από τους λόγους για την εντυπωσιακή ακρίβεια των API είναι η δυνατότητα επιλογής μεταξύ διαφορετικών μοντέλων μηχανικής εκμάθησης, ανάλογα με το τι χρησιμοποιείται η εφαρμογή σας. Αυτό καθιστά επίσης το Google Speech-To-Text μια κατάλληλη λύση για εφαρμογές εκτός από τις σύντομες αναζητήσεις ιστού. Μπορεί επίσης να ρυθμιστεί για ήχο από τηλεφωνικές κλήσεις ή βίντεο. Υπάρχει επίσης μια τέταρτη ρύθμιση, την οποία συνιστά η Google να χρησιμοποιείται ως προεπιλογή.

Το API ομιλίας σε κείμενο διαθέτει επίσης μια εντυπωσιακή ενημέρωση για επιλογές εκτεταμένων σημείων στίξης. Αυτό έχει σχεδιαστεί για να κάνει πιο χρήσιμες μεταγραφές, με λιγότερες τρέχουσες προτάσεις ή σφάλματα στίξης.

Η πιο πρόσφατη ενημέρωση επιτρέπει επίσης στους προγραμματιστές να προσθέσουν ετικέτες στον ήχο ή το βίντεο που έχουν μεταγραφεί με βασικά μεταδεδομένα. Αυτό είναι περισσότερο προς όφελος της εταιρείας παρά για τους προγραμματιστές, ωστόσο, καθώς θα επιτρέψει στην Google να αποφασίσει ποιες λειτουργίες είναι πιο χρήσιμες για τους προγραμματιστές.

Το Google Speech-To-Text API δεν είναι, ωστόσο, δωρεάν. Είναι δωρεάν για αναγνώριση ομιλίας για ήχο λιγότερο από 60 λεπτά. Για μεταγραφές ήχου περισσότερο από αυτό, κοστίζει 0,006 ανά 15 δευτερόλεπτα.

Πλεονεκτήματα

- Αναγνωρίζει πάνω από 120 γλώσσες
- Πολλαπλά μοντέλα μηχανικής εκμάθησης για αυξημένη ακρίβεια
- Αυτόματη αναγνώριση γλώσσας
- Μεταγραφή κειμένου
- Σωστή αναγνώριση ουσιαστικών
- Ιδιωτικότητα δεδομένων
- Ακύρωση θορύβου για ήχο από τηλεφωνικές κλήσεις και βίντεο

Μειονεκτήματα

- Κοστίζει χρήματα
- Περιορισμένο πρόγραμμα δημιουργίας λεξιλογίου

4.1.2 Microsoft Cognitive Services Speech-to-Text API

4.1.2.1 Περιγραφή

Microsoft Cognitive Services παρέχουν την μεταγραφή ομιλίας σε κείμενο από την υπηρεσία ομιλίας, γνωστή και ως αναγνώριση ομιλίας, επιτρέπει τη μεταγραφή

ροών ήχου σε κείμενο σε πραγματικό χρόνο. Είναι επίσης ένα μέρος των υπηρεσιών Microsoft Trust που προσφέρουν ασύγκριτες επιλογές ασφάλειας για προγραμματιστές που αναζητούν τα πιο ασφαλή δεδομένα για τις εφαρμογές τους. Το κύριο πράγμα που διαχωρίζει το Microsoft Cognitive Services 'Speech to Text API είναι η λειτουργία Αναγνώρισης Ομιλητή. Μπορεί να πραγματοποιήσει μεταγραφή σε πραγματικό χρόνο.

4.1.2.2 Δυνατότητες

Πλεονεκτήματα

- Βελτιωμένη ασφάλεια δεδομένων μέσω αλγορίθμων αναγνώρισης φωνής
- Μεταγραφή σε πραγματικό χρόνο
- Μετάφραση σε πραγματικό χρόνο
- Προσαρμόσιμο λεξιλόγιο
- Δυνατότητες κειμένου σε ομιλία για φυσικά πρότυπα ομιλίας

Μειονεκτήματα

- Ενσωματωμένοι περιορισμοί λόγω της δημιουργίας του API για γενικούς σκοπούς
- Χρησιμοποιεί μικροσυσκευές, οι οποίες μπορεί να είναι χρήσιμες για την επίλυση μεμονωμένων προβλημάτων, αλλά δεν ανταποκρίνονται σε μεγαλύτερα προβλήματα

4.1.3 IBM Watson Speech-to-Text API

4.1.3.1 Περιγραφή

Το Watson Speech to Text είναι μια λύση εγγενής στο cloud που χρησιμοποιεί αλγόριθμους AI βαθιάς μάθησης για την εφαρμογή γνώσεων σχετικά με τη γραμματική, τη δομή της γλώσσας και τη σύνθεση ήχου / φωνητικού σήματος για τη δημιουργία προσαρμόσιμης αναγνώρισης ομιλίας για βέλτιστη μεταγραφή κειμένου.

4.1.3.2 Δυνατότητες

Πλεονεκτήματα

- Επεξεργάζεται μη δομημένα δεδομένα
- Βοηθά τους ανθρώπους αντί να τους αντικαθιστούν
- Βοηθά να ξεπεραστούν οι ανθρώπινοι περιορισμοί
- Βελτιώνει την παραγωγικότητα παρέχοντας σχετικά δεδομένα
- Βελτιώνει την εμπειρία χρήστη
- Μπορεί να επεξεργαστεί μεγάλες ποσότητες δεδομένων
- Εύκολη εγκατάσταση και έναρξη

Μειονεκτήματα

- Δεν υποστηρίζει άμεσα δομημένα δεδομένα
- Ακριβές για μετάβαση σε
- Απαιτείται συντήρηση
- Υποστηρίζει μόνο περιορισμένο αριθμό γλωσσών
- Χρειάζεται χρόνος για πλήρη εφαρμογή
- Απαιτεί εκπαίδευση και κατάρτιση για να αξιοποιήσει πλήρως τους πόρους της

Κεφάλαιο 5

Επιλογή API

Στην προηγούμενη ενότητα περιγράψαμε τρία APIs. Το κάθε ένα είναι ικανό για μετραγραφή ομιλίας σε κείμενο. Όλα τα APIs εφαρμόζουν Machine learning και AI τεχνολογίες. Το καθένα API έχει τα διαχωριστικά σημεία.

Το κύριο πράγμα που διαχωρίζει το Microsoft Cognitive Services 'Speech to Text API είναι η λειτουργία Αναγνώρισης Ομιλητή. Αυτή είναι η ακουστική έκδοση του λογισμικού ασφαλείας, όπως η αναγνώριση προσώπου. Σκεφτείτε το ως σάρωση αμφιβληστροειδούς για τον ήχο της φωνής του χρήστη. Το καθιστά απίστευτα εύκολο για διαφορετικά επίπεδα χρηστών. Αυτή η ίδια δυνατότητα αναγνώρισης φωνής επιτρέπει στο λογισμικό να προσαρμόζεται στα συγκεκριμένα στυλ και μοτίβα ομιλίας του χρήστη. Προσφέρει επίσης περισσότερες προσαρμοσμένες επιλογές λεξιλογίου από το Google, ως επιπλέον πλεονέκτημα.

Το IBM Watson Speech to Text API είναι ιδιαίτερα ανθεκτικό στην κατανόηση τα συμφραζόμενα, στηρίζεται στη δημιουργία υποθέσεων και στην αξιολόγηση στη διαμόρφωση της απόκρισης. Είναι επίσης σε θέση να κάνει διάκριση μεταξύ πολλαπλών ομιλητών, γεγονός που το καθιστά κατάλληλο για τις περισσότερες εργασίες μεταγραφής. Μπορείτε ακόμη και να ορίσετε έναν αριθμό φίλτρων, εξαλείφοντας βωμολοχίες, προσθέτοντας εμπιστοσύνη λέξεων και επιλογές μορφοποίησης για εφαρμογές ομιλίας σε κείμενο.

Ένας από τους λόγους για την εντυπωσιακή ακρίβεια των API είναι η δυνατότητα επιλογής μεταξύ διαφορετικών μοντέλων μηχανικής μάθησης, ανάλογα με το τι χρησιμοποιεί η εφαρμογή. Αυτό καθιστά επίσης το Google Speech-To-Text μια κατάλληλη λύση για εφαρμογές εκτός από τις σύντομες αναζητήσεις ιστού. Μπορεί επί-

σης να ρυθμιστεί για ήχο από τηλεφωνικές κλήσεις ή βίντεο. Υπάρχει επίσης μια τέταρτη ρύθμιση, την οποία συνιστά η Google να χρησιμοποιείται ως προεπιλογή. Το API Speech-To-Text διαθέτει επίσης μια εντυπωσιακή ενημέρωση για εκτεταμένες επιλογές στίξης. Αυτό έχει σχεδιαστεί για να κάνει πιο χρήσιμες μεταγραφές, με λιγότερες τρέχουσες προτάσεις ή σφάλματα στίξης. Επίσης το Google cloud Speech-to-Text API αναγνωρίζει πάνω από 120 γλώσσες συμπεριλαμβάνοντας και την Ελληνική γλώσσα που χρησιμοποιείτε για την ανάπτυξη της βιβλιοθήκης. Το Google Speech-to-Text API προσφέρει τους εξής τρεις τρόπους χρήσης: Τις βιβλιοθήκες πελατών Google Cloud χρησιμοποιώντας γλώσσες προγραμματισμού, Το gcloud εργαλείο χρησιμοποιώντας το τερματικό και το τερματικό με την εντολή curl και την REST διεπαφή.

Κεφάλαιο 6

Γενική ιδέα API

Αυτό το έγγραφο είναι ένας οδηγός για τα βασικά στοιχεία της χρήσης Google Speech-to-Text. Αυτός ο εννοιολογικός οδηγός καλύπτει τους τύπους αιτημάτων που μπορείτε να υποβάλετε σε Speech-to-Text, πώς να δημιουργήσετε αυτά τα αιτήματα και πώς να χειριστείτε τις απαντήσεις τους.

6.1 Αιτήματα ομιλίας

Αιτήματα ομιλίας Το Speech-to-Text έχει τρεις κύριες μεθόδους για την εκτέλεση αναγνώρισης ομιλίας. Παρατίθενται παρακάτω:

- **Synchronous Recognition** (REST και gRPC) στέλνει δεδομένα ήχου στο API ομιλίας σε κείμενο, εκτελεί αναγνώριση σε αυτά τα δεδομένα και επιστρέφει αποτελέσματα μετά την επεξεργασία όλων των ήχων. Τα αιτήματα σύγχρονης αναγνώρισης περιορίζονται σε δεδομένα ήχου διάρκειας 1 λεπτού ή λιγότερο.

- **Asynchronous Recognition** (REST και gRPC) στέλνει δεδομένα ήχου στο Speech-to-Text API και ξεκινά μια μακροχρόνια λειτουργία. Χρησιμοποιώντας αυτήν τη λειτουργία, μπορείτε περιοδικά να κάνετε δημοσκοπήσεις για αποτελέσματα αναγνώρισης. Χρησιμοποιήστε ασύγχρονα αιτήματα για δεδομένα ήχου οποιασδήποτε διάρκειας έως 480 λεπτά.

- **Streaming Recognition** (μόνο gRPC) εκτελεί αναγνώριση σε δεδομένα ήχου που παρέχονται σε μια αμφίδρομη ροή gRPC. Τα αιτήματα ροής έχουν σχεδιαστεί για σκοπούς αναγνώρισης σε πραγματικό χρόνο, όπως η λήψη ζωντα-

νών ήχων από ένα μικρόφωνο. Η αναγνώριση ροής παρέχει προσωρινά αποτελέσματα κατά τη λήψη ήχου, επιτρέποντας την εμφάνιση αποτελεσμάτων, για παράδειγμα, ενώ ένας χρήστης εξακολουθεί να μιλά.

Τα αιτήματα περιέχουν παραμέτρους διαμόρφωσης καθώς και δεδομένα ήχου. Οι ακόλουθες ενότητες περιγράφουν αυτόν τον τύπο αιτημάτων αναγνώρισης, οι αποκρίσεις που δημιουργούν και τον τρόπο χειρισμού αυτών των αποκρίσεων με περισσότερες λεπτομέρειες.

6.1.1 Speech-to-Text API recognition

Ένα σύγχρονο αίτημα ομιλίας Speech-to-Text API είναι η απλούστερη μέθοδος για την αναγνώριση δεδομένων ήχου ομιλίας. Το Speech-to-Text μπορεί να επεξεργαστεί έως και 1 λεπτό δεδομένων ήχου ομιλίας που αποστέλλονται σε ένα σύγχρονο αίτημα. Μετά την επεξεργασία ομιλίας σε κείμενο και αναγνωρίζει όλο τον ήχο, επιστρέφει μια απάντηση. Ένα σύγχρονο αίτημα αποκλείει, που σημαίνει ότι το Speech-to-Text πρέπει να επιστρέψει μια απάντηση πριν από την επεξεργασία του επόμενου αιτήματος. Το Speech-to-Text συνήθως επεξεργάζεται τον ήχο ταχύτερα από τον πραγματικό χρόνο, ενώ επεξεργάζεται 30 δευτερόλεπτα ήχου σε 15 δευτερόλεπτα κατά μέσο όρο. Σε περιπτώσεις κακής ποιότητας ήχου, το αίτημά σας αναγνώρισης μπορεί να διαρκέσει σημαντικά περισσότερο. Το Speech-to-Text έχει μεθόδους REST και gRPC για την κλήση του Speech-to-Text API συγχρονισμένων και ασύγχρονων αιτημάτων. Αυτό το άρθρο δείχνει το REST API επειδή είναι πιο απλό να εμφανιστεί και να εξηγηθεί η βασική χρήση του API. Ωστόσο, η βασική σύνθεση ενός αιτήματος REST ή gRPC είναι αρκετά παρόμοια. Τα αιτήματα αναγνώρισης ροής υποστηρίζονται μόνο από το gRPC.

6.1.1.1 Αιτήματα αναγνώρισης σύγχρονης ομιλίας

Ένα σύγχρονο Speech-to-Text API αίτημα αποτελείται από μια διαμόρφωση αναγνώρισης ομιλίας και δεδομένα ήχου. Ένα δείγμα αίτησης εμφανίζεται παρακάτω:

```
{
  ”config”: {
  ”encoding”: ”LINEAR16”,
  ”sampleRateHertz”: 16000,
  ”languageCode”: ”en-US”,
  },
```

```
  "audio": {
    "uri": "gs://bucket-name/path_to_audio_file"
  }
}
```

Όλα τα σύγχρονα αιτήματα αναγνώρισης Speech-to-Text API πρέπει να περιλαμβάνουν ένα πεδίο διαμόρφωσης αναγνώρισης ομιλίας (τύπου RecognitionConfig). Το RecognitionConfig περιέχει τα ακόλουθα δευτερεύοντα πεδία:

- **encoding** - (απαιτείται) καθορίζει το σχήμα κωδικοποίησης του παρεχόμενου ήχου (τύπου AudioEncoding). Εάν έχετε την επιλογή στον κωδικοποιητή, προτιμήστε μια κωδικοποίηση χωρίς απώλειες όπως το FLAC ή το LINEAR16 για καλύτερη απόδοση. Το πεδίο κωδικοποίησης είναι προαιρετικό για αρχεία FLAC και WAV όπου η κωδικοποίηση περιλαμβάνεται στην κεφαλίδα του αρχείου.

- **sampleRateHertz** - (απαιτείται) καθορίζει την ταχύτητα δείγματος (σε Hertz) του παρεχόμενου ήχου. Το πεδίο sampleRateHertz είναι προαιρετικό για αρχεία FLAC και WAV όπου ο ρυθμός δείγματος περιλαμβάνεται στην κεφαλίδα του αρχείου.

- **languageCode** - (απαιτείται) περιέχει τη γλώσσα + περιοχή / τοπικές ρυθμίσεις για χρήση για αναγνώριση ομιλίας του παρεχόμενου ήχου. Ο κωδικός γλώσσας πρέπει να είναι αναγνωριστικό BCP-47. Σημειώστε ότι οι κωδικοί γλώσσας αποτελούνται συνήθως από ετικέτες πρωτογενούς γλώσσας και δευτερεύουσες ετικέτες δευτερεύουσας περιοχής για να υποδείξουν διαλέκτους (για παράδειγμα, «en» για Αγγλικά και «US» για τις Ηνωμένες Πολιτείες στο παραπάνω παράδειγμα.)

- **maxAlternatives** - (προαιρετικά, προεπιλογή σε 1) υποδεικνύει τον αριθμό των εναλλακτικών μεταγραφών που πρέπει να παρέχονται στην απόκριση. Από προεπιλογή, το API ομιλίας σε κείμενο παρέχει μία κύρια μεταγραφή. Εάν θέλετε να αξιολογήσετε διαφορετικές εναλλακτικές, ορίστε το maxAlternatives σε υψηλότερη τιμή. Σημειώστε ότι το Speech-to-Text θα επιστρέψει εναλλακτικές λύσεις μόνο εάν ο αναγνωριστής καθορίσει εναλλακτικές λύσεις επαρκούς ποιότητας. Γενικά, οι εναλλακτικές είναι πιο κατάλληλες για αιτήματα σε πραγματικό χρόνο

που απαιτούν σχόλια από τον χρήστη (για παράδειγμα, φωνητικές εντολές) και επομένως είναι πιο κατάλληλες για αιτήματα αναγνώρισης ροής.

- **speechContext** - (προαιρετικά) περιέχει πρόσθετες πληροφορίες με βάση τα συμφραζόμενα για την επεξεργασία αυτού του ήχου. Ένα πλαίσιο περιέχει το ακόλουθο υπο-πεδίο:
  - **phrases** - περιέχει μια λίστα λέξεων και φράσεων που παρέχουν συμβουλές για την εργασία αναγνώρισης ομιλίας.

Ο ήχος παρέχεται στο Speech-to-Text μέσω της παραμέτρου ήχου του τύπου RecognitionAudio. Το πεδίο ήχου περιέχει ένα από τα ακόλουθα υπο-πεδία:

- **content** - περιέχει τον ήχο για αξιολόγηση, ενσωματωμένο στο αίτημα. Ο ήχος που μεταδίδεται απευθείας σε αυτό το πεδίο περιορίζεται σε 1 λεπτό σε διάρκεια.
- **uri** - περιέχει ένα URI που δείχνει το περιεχόμενο ήχου. Το αρχείο δεν πρέπει να συμπιεστεί (για παράδειγμα, gzip). Προς το παρόν, αυτό το πεδίο πρέπει να περιέχει URI Google Cloud Storage (με μορφή `gs://bucket-name/path_to_audio_file`).

Περισσότερες πληροφορίες σχετικά με αυτές τις παραμέτρους αιτήματος και απόκρισης εμφανίζονται παρακάτω.

6.1.1.2 Ποσοστά δειγμάτων

Μπορείτε να καθορίσετε την ταχύτητα δειγματοληψίας του ήχου σας στο πεδίο sampleRateHertz της διαμόρφωσης αιτήματος και πρέπει να ταιριάζει με την ταχύτητα δείγματος του σχετικού περιεχομένου ή ροής ήχου. Τα ποσοστά δειγμάτων μεταξύ 8000 Hz και 48000 Hz υποστηρίζονται στο Speech-to-Text. Ο ρυθμός δείγματος για ένα αρχείο FLAC ή WAV μπορεί να προσδιοριστεί από την κεφαλίδα του αρχείου αντί από το πεδίο sampleRateHertz.

Εάν έχετε την επιλογή κατά την κωδικοποίηση του αρχικού υλικού, τραβήξτε ήχο χρησιμοποιώντας ρυθμό δείγματος 16000 Hz. Οι τιμές χαμηλότερες από αυτήν ενδέχεται να επηρεάσουν την ακρίβεια της αναγνώρισης ομιλίας και τα υψηλότερα επίπεδα δεν έχουν σημαντική επίδραση στην ποιότητα αναγνώρισης ομιλίας.

Ωστόσο, εάν τα δεδομένα ήχου σας έχουν ήδη εγγραφεί με υπάρχον ρυθμό δειγματοληψίας διαφορετικό από 16000 Hz, μην επαναλάβετε τη λήψη του ήχου σε 16000 Hz. Για παράδειγμα, οι περισσότεροι ήχοι παλαιάς τηλεφωνίας χρησιμοποιούν ρυθμούς δειγμάτων 8000 Hz, κάτι που μπορεί να δώσει λιγότερο ακριβή αποτελέσματα. Εάν πρέπει να χρησιμοποιήσετε τέτοιο ήχο, δώστε τον ήχο στο API ομιλίας με το εγγενές ρυθμό δειγματοληψίας του.

6.1.1.3 Γλώσσες

Η μηχανή αναγνώρισης ομιλίας σε κείμενο υποστηρίζει μια ποικιλία γλωσσών και διαλέκτων. Καθορίζετε τη γλώσσα (και την εθνική ή περιφερειακή διάλεκτο) του ήχου σας στο πεδίο LanguageCode της διαμόρφωσης του αιτήματος, χρησιμοποιώντας ένα αναγνωριστικό BCP-47.

6.1.1.4 Χρονικές αντισταθμίσεις (χρονικές σημάνσεις)

Η ομιλία σε κείμενο μπορεί να περιλαμβάνει τιμές μετατόπισης χρόνου (χρονικές σημάνσεις) για την αρχή και το τέλος κάθε προφορικής λέξης που αναγνωρίζεται στον παρεχόμενο ήχο. Η τιμή μετατόπισης χρόνου αντιπροσωπεύει το χρονικό διάστημα που έχει παρέλθει από την αρχή του ήχου, σε βήματα των 100ms.

Οι αντισταθμίσεις χρόνου είναι ιδιαίτερα χρήσιμες για την ανάλυση μεγαλύτερων αρχείων ήχου, όπου ίσως χρειαστεί να αναζητήσετε μια συγκεκριμένη λέξη στο αναγνωρισμένο κείμενο και να την εντοπίσετε (αναζήτηση) στον αρχικό ήχο. Οι αντισταθμίσεις ώρας υποστηρίζονται για όλες τις μεθόδους αναγνώρισής: αναγνώριση, αναγνώριση ροής και αναγνώριση μεγάλης διάρκειας.

Οι τιμές μετατόπισης χρόνου περιλαμβάνονται μόνο για την πρώτη εναλλακτική που παρέχεται στην απόκριση αναγνώρισης.

Για να συμπεριλάβετε αντισταθμίσεις χρόνου στα αποτελέσματα του αιτήματός σας, ορίστε την παράμετρο allowWordTimeOffsets σε πραγματική τιμή στη διαμόρφωση του αιτήματός σας. Για παραδείγματα που χρησιμοποιούν το REST API ή τις Βιβλιοθήκες πελατών. Για παράδειγμα, μπορείτε να συμπεριλάβετε την παράμετρο allowWordTimeOffsets στη διαμόρφωση του αιτήματος όπως φαίνεται εδώ:

```
1 {
2   "config": {
```

```
  "languageCode": "en-US",
  "enableWordTimeOffsets": true
  },
  "audio":{
    "uri":"gs://gcs-test-data/gettysburg.flac"
  }
}
```

Το αποτέλεσμα που επιστρέφεται από το Speech-to-Text API θα περιέχει τιμές μετατόπισης χρόνου για κάθε αναγνωρισμένη λέξη όπως φαίνεται παρακάτω:

```
{
  "name": "6212202767953098955",
  "metadata": {
  "@type": "type.googleapis.com/google.cloud.speech.v1.
      ↪ LongRunningRecognizeMetadata",
  "progressPercent": 100,
  "startTime": "2017-07-24T10:21:22.013650Z",
  "lastUpdateTime": "2017-07-24T10:21:45.278630Z"
  },
  "done": true,
  "response": {
    "@type": "type.googleapis.com/google.cloud.speech.v1.
        ↪ LongRunningRecognizeResponse",
    "results": [
      {
        "alternatives": [
        {
          "transcript": "Four score and twenty...(etc)...",
        "confidence": 0.97186122,
        "words": [
          {
            "startTime": "1.300s",
            "endTime": "1.400s",
            "word": "Four"
          },
          {
            "startTime": "1.400s",
            "endTime": "1.600s",
            "word": "score"
          },
          {
            "startTime": "1.600s",
            "endTime": "1.600s",
            "word": "and"
          },
```

```
        {
          "startTime": "1.600s",
          "endTime": "1.900s",
          "word": "twenty"
        },
        ...
      ]
      }
    ]
    },
    {
      "alternatives": [
      {
        "transcript": "for score and plenty...(etc)...",
      "confidence": 0.9041967,
      }
    ]
    }
  ]
 }
}
```

6.1.1.5 Επιλογή μοντέλων

Το Speech-to-Text μπορεί να χρησιμοποιήσει ένα από τα πολλά μοντέλα μηχανικής μάθησης για να μεταγράψει το αρχείο ήχου σας. Η Google έχει εκπαιδεύσει αυτά τα μοντέλα αναγνώρισης ομιλίας για συγκεκριμένους τύπους ήχου και πηγές.

Όταν στέλνετε ένα αίτημα μεταγραφής ήχου στο Speech-to-Text, μπορείτε να βελτιώσετε τα αποτελέσματα που λαμβάνετε καθορίζοντας την πηγή του αρχικού ήχου. Αυτό επιτρέπει στο Speech-to-Text API να επεξεργάζεται τα αρχεία ήχου σας χρησιμοποιώντας ένα μοντέλο μηχανικής εκμάθησης που έχει εκπαιδευτεί να αναγνωρίζει ήχο ομιλίας από τον συγκεκριμένο τύπο πηγής.

Για να καθορίσετε ένα μοντέλο αναγνώρισης ομιλίας, συμπεριλάβετε το πεδίο μοντέλου στο αντικείμενο RecognitionConfig για το αίτημά σας, καθορίζοντας το μοντέλο που θέλετε να χρησιμοποιήσετε.

Το Speech-to-Text μπορεί να χρησιμοποιήσει τους ακόλουθους τύπους μοντέλων μηχανικής μάθησης για τη μεταγραφή των αρχείων ήχου.

Πίνακας 6.1: Μοντέλα μηχανικής μάθησης

Τύπος	Enum Constant	Περιγραφή
Βίντεο	video	Χρησιμοποιήστε αυτό το μοντέλο για μεταγραφή ήχου σε βίντεο κλιπ ή που περιλαμβάνει πολλά ηχεία. Για καλύτερα αποτελέσματα, παρέχετε ήχο εγγεγραμμένο στα 16.000Hz ή μεγαλύτερο ρυθμό δειγματοληψίας.
Τηλεφωνική κλήση	phone_call	Χρησιμοποιήστε αυτό το μοντέλο για μεταγραφή ήχου από μια τηλεφωνική κλήση. Συνήθως, ο ήχος του τηλεφώνου καταγράφεται σε ρυθμό δειγματοληψίας 8.000Hz.
ASR: Εντολή και Αναζήτηση	command_and_search	Χρησιμοποιήστε αυτό το μοντέλο για να μεταγράψετε μικρότερα κλιπ ήχου. Μερικά παραδείγματα περιλαμβάνουν φωνητικές εντολές ή φωνητική αναζήτηση.
ASR: Προκαθορισμένο	default	Χρησιμοποιήστε αυτό το μοντέλο εάν ο ήχος σας δεν ταιριάζει σε ένα από τα μοντέλα που περιγράφηκαν προηγουμένως. Για παράδειγμα, μπορείτε να το χρησιμοποιήσετε για εγγραφές ήχου μεγάλης διάρκειας που διαθέτουν μόνο ένα ηχείο. Στην ιδανική περίπτωση, ο ήχος είναι υψηλής πιστότητας, καταγράφεται στα 16.000Hz ή υψηλότερος ρυθμός δειγματοληψίας.

6.1.1.6 Μεταβίβαση ήχου που αναφέρεται από ένα URI

Συνήθως, θα μεταβιβάσετε μια παράμετρο uri στο πεδίο ήχου του αιτήματος ομιλίας, δείχνοντας ένα αρχείο ήχου (σε δυαδική μορφή, όχι base64) που βρίσκεται στο Google Cloud Storage της ακόλουθης φόρμας:

```
\path{gs://bucket-name/path_to_audio_file}
```

Για παράδειγμα, το ακόλουθο μέρος ενός αιτήματος ομιλίας αναφέρεται στο δείγμα αρχείου ήχου:

```
...
 "audio": {
  "uri":"gs://cloud-samples-tests/speech/brooklyn.flac"
 }
...
```

Πρέπει να έχετε τα κατάλληλα δικαιώματα πρόσβασης για να διαβάσετε αρχεία Google Cloud Storage, όπως ένα από τα ακόλουθα:

- Με δυνατότητα ανάγνωσης στο κοινό (όπως τα δείγματα αρχείων ήχου)
- Αναγνώσιμο από τον λογαριασμό υπηρεσίας σας, εάν χρησιμοποιείτε εξουσιοδότηση λογαριασμού υπηρεσίας
- Μπορεί να διαβαστεί από λογαριασμό χρήστη, εάν χρησιμοποιείτε 3-legged OAuth για εξουσιοδότηση λογαριασμού χρήστη.

6.1.2 Speech-to-Text API responses

Όπως αναφέρθηκε προηγουμένως, μια σύγχρονη απόκριση Speech-to-Text API ενδέχεται να χρειαστεί λίγο χρόνο για την επιστροφή των αποτελεσμάτων, ανάλογα με τη διάρκεια του παρεχόμενου ήχου. Μόλις υποβληθεί σε επεξεργασία, το API θα επιστρέψει μια απάντηση όπως φαίνεται παρακάτω:

```
{
 "results": [
  {
   "alternatives": [
   {
    "confidence": 0.98267895,
    "transcript": "how old is the Brooklyn Bridge"
   }
```

```
    ]
    }
  ]
}
```

Αυτά τα πεδία εξηγούνται παρακάτω:

- Το **results** περιέχουν τη λίστα αποτελεσμάτων (του τύπου **SpeechRecognitionResult**) όπου κάθε αποτέλεσμα αντιστοιχεί σε ένα τμήμα ήχου (τα τμήματα του ήχου διαχωρίζονται με παύσεις). Κάθε αποτέλεσμα θα αποτελείται από ένα ή περισσότερα από τα ακόλουθα πεδία:
  - Το **Alternatives** περιέχει μια λίστα πιθανών μεταγραφών, τύπου **SpeechRecognitionAlternatives**. Το εάν εμφανίζονται περισσότερες από μία εναλλακτικές επιλογές εξαρτάται τόσο από το αν ζητήσατε περισσότερες από μία εναλλακτικές (ορίζοντας **maxAlternatives** σε τιμή μεγαλύτερη από **1**) όσο και από το εάν το Speech-to-Text παρήγαγε εναλλακτικές λύσεις αρκετά υψηλής ποιότητας. Κάθε εναλλακτική λύση θα αποτελείται από τα ακόλουθα πεδία:
    * **transcript** περιέχει το μεταγραμμένο κείμενο.
    * **confidence** περιέχει μια τιμή μεταξύ 0 και 1 που δείχνει πόσο σίγουρη είναι η ομιλία σε κείμενο για τη δεδομένη μεταγραφή.

Εάν δεν μπορεί να αναγνωριστεί ομιλία από τον παρεχόμενο ήχο, τότε η λίστα αποτελεσμάτων που επιστρέφεται δεν θα περιέχει στοιχεία. Η μη αναγνωρισμένη ομιλία είναι συνήθως το αποτέλεσμα ήχου πολύ κακής ποιότητας ή από κωδικούς γλώσσας, κωδικοποίηση ή τιμές δείγματος που δεν ταιριάζουν με τον παρεχόμενο ήχο. Τα στοιχεία αυτής της απόκρισης εξηγούνται στις ακόλουθες ενότητες. Κάθε σύγχρονη απόκριση Speech-to-Text API επιστρέφει μια λίστα αποτελεσμάτων και όχι ένα αποτέλεσμα που περιέχει όλο τον αναγνωρισμένο ήχο. Η λίστα των αναγνωρισμένων ήχων (εντός των στοιχείων μεταγραφής) θα εμφανιστεί σε συνεχόμενη σειρά.

6.1.2.1 Επιλογή εναλλακτικών λύσεων

Κάθε αποτέλεσμα σε μια επιτυχημένη απόκριση σύγχρονης αναγνώρισης μπορεί να περιέχει μία ή περισσότερες εναλλακτικές (εάν η τιμή maxAlternatives για το αί-

τημα είναι μεγαλύτερη από 1). Εάν το Speech-to-Text προσδιορίσει ότι μια εναλλακτική έχει επαρκή τιμή εμπιστοσύνης, τότε αυτή η εναλλακτική συμπεριλαμβάνεται στην απόκριση. Η πρώτη εναλλακτική λύση στην απάντηση είναι πάντα η καλύτερη (πιθανότατα) εναλλακτική λύση.

Ο ορισμός maxAlternatives σε υψηλότερη τιμή από 1 δεν συνεπάγεται ούτε εγγυάται την επιστροφή πολλαπλών εναλλακτικών. Γενικά, περισσότερες από μία εναλλακτικές είναι καταλληλότερες για την παροχή επιλογών σε πραγματικό χρόνο στους χρήστες που λαμβάνουν αποτελέσματα μέσω ενός αιτήματος αναγνώρισης ροής.

6.1.2.2 Χειρισμός μεταγραφών

Κάθε εναλλακτική λύση που παρέχεται εντός της απόκρισης θα περιέχει ένα αντίγραφο που περιέχει το αναγνωρισμένο κείμενο. Όταν παρέχονται διαδοχικές εναλλακτικές λύσεις, θα πρέπει να συνδυάσετε αυτές τις μεταγραφές μαζί.

6.1.2.3 Τιμές εμπιστοσύνης

Η τιμή εμπιστοσύνης είναι μια εκτίμηση μεταξύ 0,0 και 1,0. Υπολογίζεται συγκεντρώνοντας τις τιμές ”πιθανότητας” που αντιστοιχούν σε κάθε λέξη στον ήχο. Ένας υψηλότερος αριθμός δείχνει μια εκτιμώμενη μεγαλύτερη πιθανότητα ότι οι μεμονωμένες λέξεις αναγνωρίστηκαν σωστά. Αυτό το πεδίο παρέχεται συνήθως μόνο για την κορυφαία υπόθεση και μόνο για αποτελέσματα όπου is_final = true. Για παράδειγμα, μπορείτε να χρησιμοποιήσετε την τιμή εμπιστοσύνης για να αποφασίσετε εάν θα εμφανίσετε εναλλακτικά αποτελέσματα στον χρήστη ή να ζητήσετε επιβεβαίωση από τον χρήστη.

Λάβετε υπόψη, ωστόσο, ότι το μοντέλο καθορίζει το ”καλύτερο”, κορυφαίο αποτέλεσμα με βάση περισσότερα σήματα από το σκορ εμπιστοσύνης μόνο (όπως το πλαίσιο προτάσεων). Εξαιτίας αυτού υπάρχουν περιστασιακές περιπτώσεις όπου το κορυφαίο αποτέλεσμα δεν έχει το υψηλότερο σκορ εμπιστοσύνης. Εάν δεν έχετε ζητήσει πολλά εναλλακτικά αποτελέσματα, το μοναδικό ”καλύτερο” αποτέλεσμα που επιστρέφεται μπορεί να έχει χαμηλότερη τιμή εμπιστοσύνης από το αναμενόμενο. Αυτό μπορεί να συμβεί, για παράδειγμα, σε περιπτώσεις όπου χρησιμοποιούνται σπάνιες λέξεις. Σε μια λέξη που σπάνια χρησιμοποιείται μπορεί να εκχωρηθεί μια χαμηλή τιμή ”πιθανότητας” ακόμη και αν αναγνωρίζεται σωστά. Εάν το μοντέλο προσδιορίσει τη σπάνια λέξη ως

την πιο πιθανή επιλογή βάσει του περιβάλλοντος, το αποτέλεσμα επιστρέφεται στην κορυφή ακόμα και αν η τιμή εμπιστοσύνης του αποτελέσματος είναι χαμηλότερη από τις εναλλακτικές επιλογές.

6.1.3 Ασύγχρονα αιτήματα και απαντήσεις

Ένα ασύγχρονο Speech-to-Text API αίτημα για τη μέθοδο LongRunningRecognize είναι πανομοιότυπο σε μορφή με ένα σύγχρονο Speech-to-Text API αίτημα. Ωστόσο, αντί να επιστρέψει μια απάντηση, το ασύγχρονο αίτημα θα ξεκινήσει μια λειτουργία μεγάλης διάρκειας (τύπου λειτουργίας) και θα επιστρέψει αυτήν τη λειτουργία στον καλούντα αμέσως.

Μια τυπική απόκριση λειτουργίας φαίνεται παρακάτω:

```
{
  "name": "operation_name",
  "metadata": {
    "@type": "type.googleapis.com/google.cloud.speech.v1.
        ↪ LongRunningRecognizeMetadata"
    "progressPercent": 34,
    "startTime": "2016-08-30T23:26:29.579144Z",
    "lastUpdateTime": "2016-08-30T23:26:29.826903Z"
  }
}
```

Λάβετε υπόψη ότι δεν υπάρχουν ακόμη αποτελέσματα. Το Speech-to-Text θα συνεχίσει να επεξεργάζεται τον παρεχόμενο ήχο και θα χρησιμοποιεί αυτήν τη λειτουργία για την αποθήκευση τελικών αποτελεσμάτων, τα οποία θα εμφανίζονται στο πεδίο απόκρισης της λειτουργίας (του τύπου LongRunningRecognizeResponse) μετά την ολοκλήρωση του αιτήματος.

Μια πλήρης απάντηση μετά την ολοκλήρωση του αιτήματος εμφανίζεται παρακάτω:

```
{
  "name": "1268386125834704889",
  "metadata": {
    "lastUpdateTime": "2016-08-31T00:16:32.169Z",
    "@type": "type.googleapis.com/google.cloud.speech.v1.
        ↪ LongrunningRecognizeMetadata",
    "startTime": "2016-08-31T00:16:29.539820Z",
    "progressPercent": 100
  }
```

```
  "response": {
  "@type": "type.googleapis.com/google.cloud.speech.v1.
      ↪ LongRunningRecognizeResponse",
  "results": [{
    "alternatives": [{
      "confidence": 0.98267895,
      "transcript": "how old is the Brooklyn Bridge"
    }]}]
  },
  "done": True,
}
```

Σημειώστε ότι η ολοκλήρωση έχει οριστεί σε True και ότι η απόκριση της λειτουργίας περιέχει ένα σύνολο αποτελεσμάτων του τύπου SpeechRecognitionResult που είναι ο ίδιος τύπος που επιστρέφεται από ένα σύγχρονο αίτημα αναγνώρισης API ομιλίας σε κείμενο.

Από προεπιλογή, μια ασύγχρονη απόκριση REST θα οριστεί σε False, η προεπιλεγμένη τιμή της. Ωστόσο, επειδή το JSON δεν απαιτεί να υπάρχουν προεπιλεγμένες τιμές μέσα σε ένα πεδίο, όταν ελέγχετε εάν μια λειτουργία έχει ολοκληρωθεί, θα πρέπει να ελέγξετε τόσο το υπάρχον πεδίο όσο και ότι έχει οριστεί σε True.

6.1.4 Ροή αιτήσεων αναγνώρισης API ομιλίας σε κείμενο

Μια κλήση αναγνώρισης Speech-to-Text API ροής έχει σχεδιαστεί για λήψη σε πραγματικό χρόνο και αναγνώριση ήχου, σε μια αμφίδρομη ροή. Η εφαρμογή σας μπορεί να στείλει ήχο στη ροή αιτημάτων και να λάβει προσωρινά και τελικά αποτελέσματα αναγνώρισης στη ροή απόκρισης σε πραγματικό χρόνο. Τα ενδιάμεσα αποτελέσματα αντιπροσωπεύουν το τρέχον αποτέλεσμα αναγνώρισης για μια ενότητα ήχου, ενώ το τελικό αποτέλεσμα αναγνώρισης αντιπροσωπεύει την τελευταία, καλύτερη εκτίμηση για αυτήν την ενότητα ήχου.

6.1.4.1 Αιτήματα ροής

Σε αντίθεση με τις σύγχρονες και ασύγχρονες κλήσεις, στις οποίες στέλνετε τόσο τη διαμόρφωση όσο και τον ήχο σε ένα μόνο αίτημα, καλώντας την ροή Speech API απαιτεί την αποστολή πολλαπλών αιτημάτων. Το πρώτο StreamingRecognizeRequest πρέπει να περιέχει μια διαμόρφωση του τύπου StreamingRecognitionConfig χωρίς συ-

νοδευτικό ήχο. Στη συνέχεια, το StreamingRecognizeRequests που αποστέλλεται μέσω της ίδιας ροής θα αποτελείται στη συνέχεια από συνεχόμενα καρέ ακατέργαστων byte ήχου.

Το StreamingRecognitionConfig αποτελείται από τα ακόλουθα πεδία:

- **config** - (απαιτείται) περιέχει πληροφορίες διαμόρφωσης για τον ήχο, τύπου RecognitionConfig και είναι το ίδιο με αυτό που εμφανίζεται σε σύγχρονα και ασύγχρονα αιτήματα.

- **single_utterance** - (προαιρετικά, από προεπιλογή σε false) υποδεικνύει εάν αυτό το αίτημα θα λήξει αυτόματα μετά την ανίχνευση ομιλίας. Εάν οριστεί, το Speech-to-Text θα εντοπίσει παύσεις, σιωπή ή ήχο χωρίς ομιλία για να καθορίσει πότε θα τερματίσει την αναγνώριση. Εάν δεν έχει οριστεί, η ροή θα συνεχίσει να ακούει και να επεξεργάζεται ήχο έως ότου είτε η ροή κλείσει απευθείας, είτε δεν έχει ξεπεραστεί το όριο της ροής. Η ρύθμιση single_utterance σε true είναι χρήσιμη για την επεξεργασία φωνητικών εντολών.

- **interim_results** - (προαιρετικό, προεπιλογή σε false) υποδεικνύει ότι αυτό το αίτημα ροής θα πρέπει να επιστρέφει προσωρινά αποτελέσματα που ενδέχεται να βελτιωθούν αργότερα (μετά την επεξεργασία περισσότερου ήχου). Τα ενδιάμεσα αποτελέσματα θα σημειωθούν εντός των απαντήσεων μέσω της ρύθμισης του is_final έως false

6.1.4.2 Ροή απαντήσεων

Τα αποτελέσματα αναγνώρισης ομιλίας ροής επιστρέφονται σε μια σειρά απαντήσεων τύπου StreamingRecognitionResponse. Μια τέτοια απάντηση αποτελείται από τα ακόλουθα πεδία:

- **speechEventType** περιέχει συμβάντα τύπου SpeechEventType. Η αξία αυτών των συμβάντων θα υποδεικνύει πότε έχει καθοριστεί ότι έχει ολοκληρωθεί μία μόνο προφορά. Τα συμβάντα ομιλίας χρησιμεύουν ως δείκτες στην απόκριση της ροής σας.

- **results** περιέχουν τη λίστα των αποτελεσμάτων, τα οποία μπορεί να είναι ενδιάμεσα ή τελικά αποτελέσματα, τύπου StreamingRecognitionResult. Η λίστα

αποτελεσμάτων περιέχει τα ακόλουθα υπο-πεδία:

- **alternatives** περιέχει μια λίστα εναλλακτικών μεταγραφών.
- **isFinal** υποδεικνύει εάν τα αποτελέσματα που λαμβάνονται σε αυτήν την καταχώριση λίστας είναι προσωρινά ή είναι οριστικά.
- **stability** δείχνει την μεταβλητότητα των αποτελεσμάτων που έχουν ληφθεί μέχρι στιγμής, με το 0,0 να δείχνει την πλήρη αστάθεια ενώ το 1,0 δείχνει την πλήρη **stability**. Σημειώστε ότι σε αντίθεση με την εμπιστοσύνη, η οποία εκτιμά αν η μεταγραφή είναι σωστή, η σταθερότητα εκτιμά αν το δεδομένο μερικό αποτέλεσμα μπορεί να αλλάξει. Εάν το **isFinal** έχει οριστεί σε αληθές, **stability** δεν θα ρυθμιστεί.

Κεφάλαιο 7

Παράδειγμα χρήσης API

7.1 Πρόλογος

Στο Google Cloud Platform γίνεται η δημιουργία εργασιών που πάνω στις εργασίες ενεργοποιούνται οι υπηρεσίες. Για το παράδειγμα χρήσης η υπηρεσία είναι το Cloud Speech-to-Text API, δημιουργόντας ένα Service Account στην υπηρεσία δίνει την δυνατότητα να χρησιμοποιηθεί μέσω ενός αρχείου json. Το αρχείο json μπορεί να φορτωθεί μέσα στον κώδικα για να μπορεί να χρησιμοποιηθεί η υπηρεσία, παρακάτω το παράδειγμα χρήσης της υπηρεσίας και του Service Account.

7.2 Χρήση βιβλιοθηκών πελατών

Αυτό το κεφάλαιο σας δείχνει πώς μπορείτε να στείλετε ένα αίτημα αναγνώρισης ομιλίας στο Speech-to-Text στην αγαπημένη σας γλώσσα προγραμματισμού χρησιμοποιώντας τις Βιβλιοθήκες πελατών Google Cloud. Σε αυτή την εργασία θα χρησιμοποιήσουμε την γλώσσα προγραμματισμού Java για κατασκευη εφαρμογής σε πλατφόρμα Android.

7.3 Εγκατάσταση της βιβλιοθήκης πελάτη

Στην γλώσσα προγραμματισμού Java η εξάρτηση της βιβλιοθήκης μέσω του Gradle εργαλείου κατασκευής γίνεται ως εξής:

```
1 compile 'com.google.cloud:google-cloud-speech:1.24.2'
```

Στο Android ενσωματώνεται στο **build.gradle** στο αντικείμενο **dependencies**:

```
dependencies {
  ...
  compile 'com.google.cloud:google-cloud-speech:1.24.2'
  ...
}
```

7.4 Χρήση API

7.4.1 Service Account

Παράδειγμα κώδικα για την φόρτωσει του Service Account μέσω του αρχείου json που είναι τοπικά αποθηκευμένο

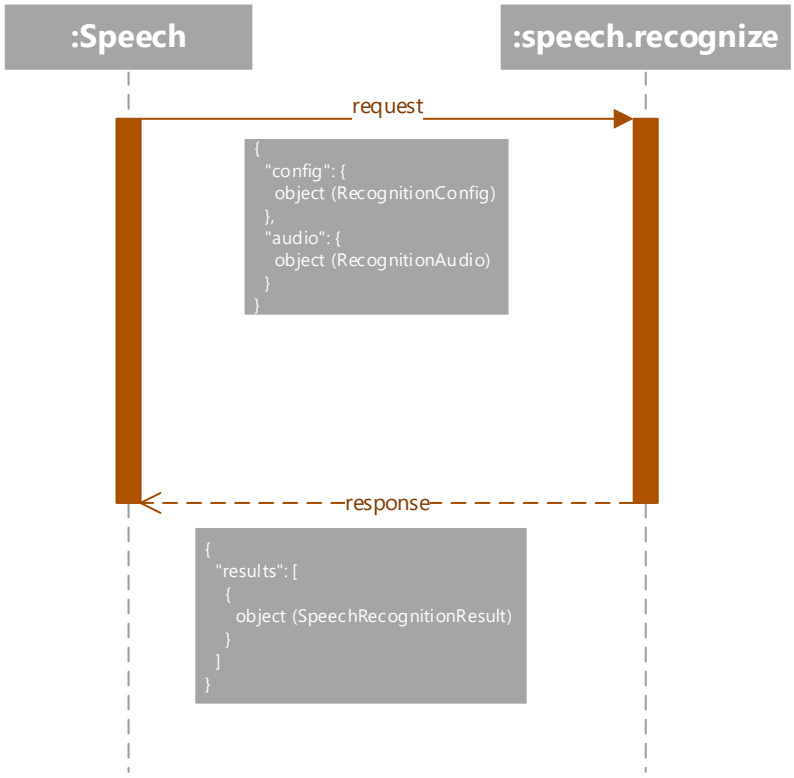
```
InputStream is = getResources().openRawResource(R.raw.credentials);
GoogleCredentials credentials = GoogleCredentials.fromStream(is);
FixedCredentialsProvider credentialsProvider = FixedCredentialsProvider.
    ↪ create(credentials);
SpeechSettings speechSettings = SpeechSettings.newBuilder()
   .setCredentialsProvider(credentialsProvider)
   .build();
```

7.4.2 Υποβολή αιτήματος μεταγραφής ήχου

```
// Imports the Google Cloud client library
import com.google.cloud.speech.v1.RecognitionAudio;
import com.google.cloud.speech.v1.RecognitionConfig;
import com.google.cloud.speech.v1.RecognitionConfig.AudioEncoding;
import com.google.cloud.speech.v1.RecognizeResponse;
import com.google.cloud.speech.v1.SpeechClient;
import com.google.cloud.speech.v1.SpeechRecognitionAlternative;
import com.google.cloud.speech.v1.SpeechRecognitionResult;
import com.google.protobuf.ByteString;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

public class QuickstartSample {

  /** Demonstrates using the Speech API to transcribe an audio file. */
  public static void main(String... args) throws Exception {
```

```
    InputStream is = getResources().openRawResource(R.raw.credentials);
    GoogleCredentials credentials = GoogleCredentials.fromStream(is);
    FixedCredentialsProvider credentialsProvider = FixedCredentialsProvider.
        ↪ create(credentials);
    SpeechSettings speechSettings = SpeechSettings.newBuilder()
    .setCredentialsProvider(credentialsProvider)
    .build();
    // Instantiates a client
    try (SpeechClient speechClient = SpeechClient.create(speechSettings)) {

      // The path to the audio file to transcribe
      String fileName = "./resources/audio.raw";

      // Reads the audio file into memory
      Path path = Paths.get(fileName);
      byte[] data = Files.readAllBytes(path);
      ByteString audioBytes = ByteString.copyFrom(data);

      // Builds the sync recognize request
      RecognitionConfig config =
      RecognitionConfig.newBuilder()
      .setEncoding(AudioEncoding.LINEAR16)
      .setSampleRateHertz(16000)
      .setLanguageCode("en-US")
      .build();
      RecognitionAudio audio = RecognitionAudio.newBuilder().setContent(
          ↪ audioBytes).build();

      // Performs speech recognition on the audio file
      RecognizeResponse response = speechClient.recognize(config, audio);
      List<SpeechRecognitionResult> results = response.getResultsList();

      for (SpeechRecognitionResult result : results) {
        // There can be several alternative transcripts for a given chunk of
            speech. Just use the
        // first (most likely) one here.
        SpeechRecognitionAlternative alternative = result.getAlternativesList
            ↪ ().get(0);
        System.out.printf("Transcription: %s%n", alternative.getTranscript());
      }
    }
  }
}
```

7.4.3 Επικοινωνία με την υπηρεσία

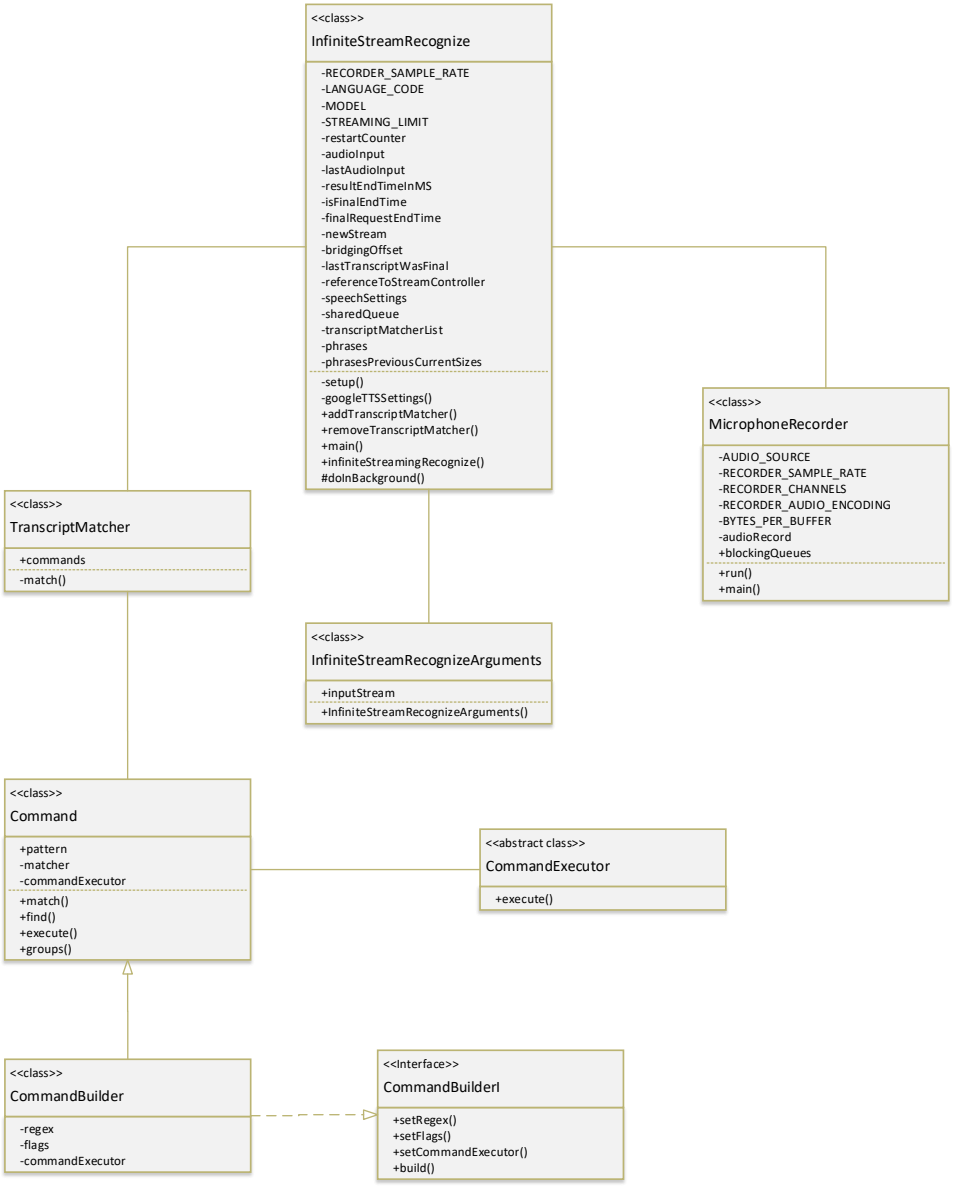


Κεφάλαιο 8

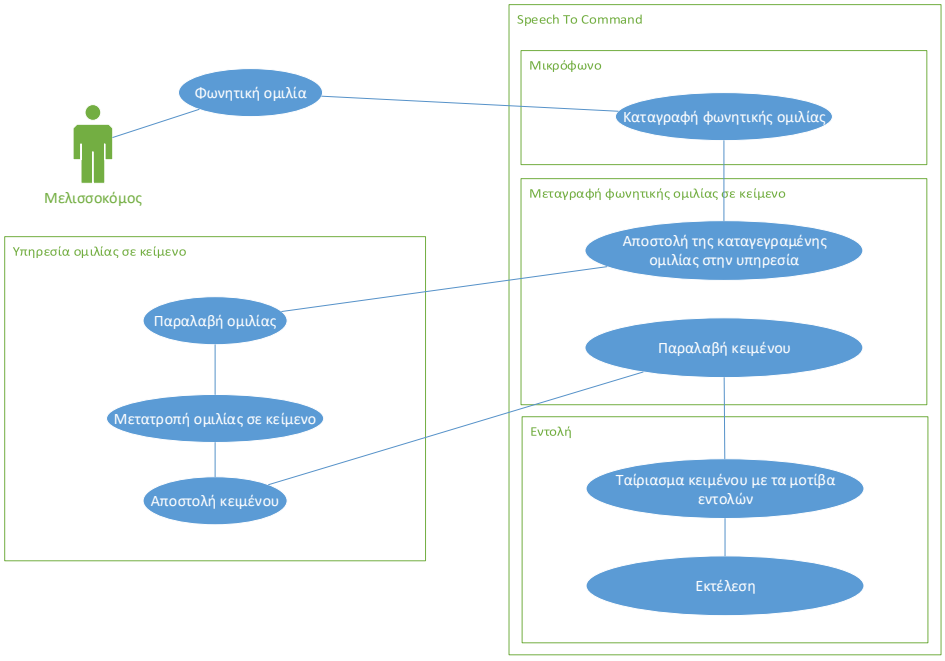
Βιβλιοθήκη Speech-to-Command

8.1 Σχεδίαση Βιβλιοθήκης

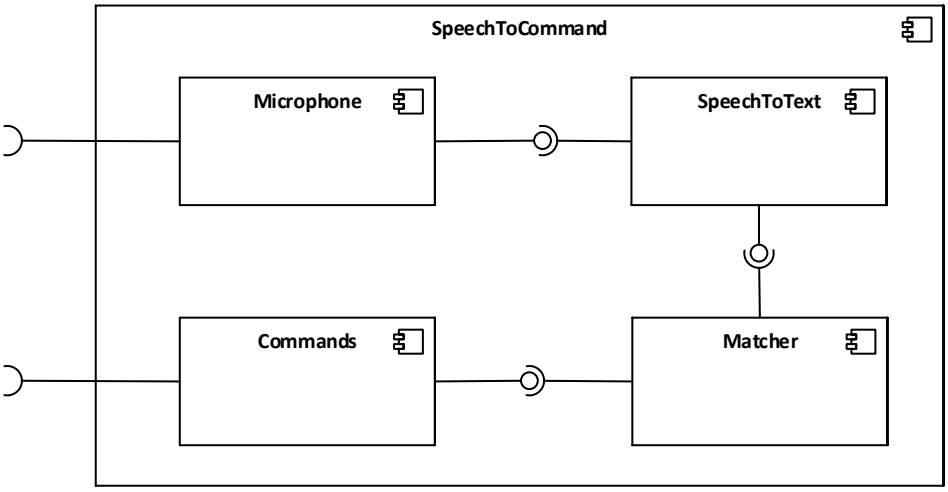
8.1.1 Διάγραμμα κλάσεων



8.1.2 Διάγραμμα σεναρίων χρήσης



8.1.3 Διάγραμμα συστατικών



8.2 Ανάπτυξη βιβλιοθήκης

Η ανάπτυξη της βιβλιοθήκης έγινε στην γλώσσα προγραμματισμού Java για Android πλατφόρμα.Τα μέροι της βιβλιοθήκης είναι τα παρακάτω:

1. Καταγραφή μικρόφωνου
2. Speech-to-Text API αναγνώριση ροής
3. Εντολή
4. Εκτελεστής εντολών
5. Ταιριαστής εντολών

8.2.1 Καταγραφή Μικροφώνου

Η καταγραφή του μικροφώνου τρέχει σε ένα νήμα στην εφαρμογή, αποθηκεύοντας τα καταγεγραμμένα δεδομένα σε μία κοινόχρηστη ουρά.

Δείγμα κώδικα:

```
1  public class MicrophoneRecorder extends Thread {
2     private static final int AUDIO_SOURCE = MediaRecorder.AudioSource.
          ↪ UNPROCESSED;
3     private static final int RECORDER_SAMPLE_RATE = 16000;
4     private static final int RECORDER_CHANNELS = AudioFormat.CHANNEL_IN_MONO;
5     private static final int RECORDER_AUDIO_ENCODING = AudioFormat.
          ↪ ENCODING_PCM_16BIT;
6     private static final int BYTES_PER_BUFFER = 1600; //  100ms
7     private static AudioRecord audioRecord;
8     public static volatile List<BlockingQueue<byte[]>> blockingQueues;
9
10    @Override
11    public void run() {
12       audioRecord.startRecording();
13       byte[] data = new byte[BYTES_PER_BUFFER];
14       while (true) {
15          try {
16             int numBytesRead = audioRecord.read(data, 0, data.length);
17             if ((numBytesRead <= 0)) {
18                continue;
19             }
20             for(BlockingQueue<byte[]> blockingQueue : blockingQueues){
```

```
21                  blockingQueue.put(data.clone());
22              }
23          } catch (InterruptedException e) {
24              System.out.println("Microphone input buffering interrupted : " +
                    ↪ e.getMessage());
25          }
26      }
27  }
28
29  public static void main() {
30      audioRecord = new AudioRecord(AUDIO_SOURCE,
31              RECORDER_SAMPLE_RATE, RECORDER_CHANNELS,
32              RECORDER_AUDIO_ENCODING, BYTES_PER_BUFFER);
33      blockingQueues = new ArrayList<>();
34      new MicrophoneRecorder().start();
35  }
36 }
```

8.2.2 Speech-to-Text API αναγνώριση ροής

Η αναγνώριση ροής τρέχει σε ένα ασύχρονο έργο στο παρασκήνιο, διαβάζοντας τα καταγεγραμμένα δεδομένα από την ουρά και κάνοντας συνέχες αιτήσεις στην υπηρεσία Speech-to-Text. Η υπηρεσία ανταποκρίνεται συνεχώς μέχρις ώτου έχει το τελικό αποτέλεσμα της μεταγραφής σε κείμενο της καταγεγραμμένης φωνητικής ομιλίας. Ο ταιριαστής των εντολών λαμβάνει την μεταγραφή ώστε να ταιριάξει το κείμενο.

Δείγμα κώδικα:

```
1  public class InfiniteStreamRecognize extends AsyncTask<
       ↪ InfiniteStreamRecognizeArguments, Void, Void> {
2
3      private static final int RECORDER_SAMPLE_RATE = 16000;
4      private static final String LANGUAGE_CODE = "el-GR";
5      private static final String MODEL = "command_and_search";
6
7      private static final int STREAMING_LIMIT = 290000; //  5 minutes
8
9      public static final String RED = "\033[0;31m";
10     public static final String GREEN = "\033[0;32m";
11     public static final String YELLOW = "\033[0;33m";
12
13     private static int restartCounter = 0;
14     private static ArrayList<ByteString> audioInput = new ArrayList<>();
```

```
15      private static ArrayList<ByteString> lastAudioInput = new ArrayList<>();
16      private static int resultEndTimeInMS = 0;
17      private static int isFinalEndTime = 0;
18      private static int finalRequestEndTime = 0;
19      private static boolean newStream = true;
20      private static double bridgingOffset = 0;
21      private static boolean lastTranscriptWasFinal = false;
22      private static StreamController referenceToStreamController;
23      private static SpeechSettings speechSettings;
24
25      // Creating shared object
26      private static volatile BlockingQueue<byte[]> sharedQueue = new
            ↪ LinkedBlockingQueue<>();
27      private static volatile List<TranscriptMatcher> transcriptMatcherList =
            ↪ new ArrayList<>();
28      private static List<String> phrases = new ArrayList<>();
29      private static int[] phrasesPreviousCurrentSizes = {0, 0};
30
31
32      private static void setup(InputStream is) throws IOException {
33          MicrophoneRecorder.blockingQueues.add(sharedQueue);
34          GoogleCredentials credentials = GoogleCredentials.fromStream(is);
35          FixedCredentialsProvider credentialsProvider =
                ↪ FixedCredentialsProvider.create(credentials);
36
37          speechSettings = SpeechSettings.newBuilder()
38                  .setCredentialsProvider(credentialsProvider)
39                  .build();
40          googleSTTSettings();
41
42      }
43
44      private static StreamingRecognitionConfig googleSTTSettings(){
45          phrasesPreviousCurrentSizes = new int[]{phrases.size(), phrases.size()
                ↪ };
46          SpeechContext sc = SpeechContext.newBuilder().addAllPhrases(phrases).
                ↪ build();
47
48          RecognitionMetadata rm = RecognitionMetadata.newBuilder()
49                  .setInteractionType(RecognitionMetadata.InteractionType.
                        ↪ VOICE_COMMAND)
50                  .setMicrophoneDistance(RecognitionMetadata.MicrophoneDistance.
                        ↪ NEARFIELD)
51                  .setOriginalMediaType(RecognitionMetadata.OriginalMediaType.
                        ↪ AUDIO)
52                  .setRecordingDeviceType(RecognitionMetadata.RecordingDeviceType.
```

```
                ↪ SMARTPHONE)
53              .build();
54
55          RecognitionConfig recognitionConfig = RecognitionConfig.newBuilder()
56                .setEncoding(RecognitionConfig.AudioEncoding.LINEAR16)
57                .setSampleRateHertz(RECORDER_SAMPLE_RATE)
58                .setLanguageCode(LANGUAGE_CODE)
59                .setModel(MODEL)
60                .addSpeechContexts(sc)
61                .setMetadata(rm)
62                .build();
63
64          return StreamingRecognitionConfig.newBuilder()
65                .setConfig(recognitionConfig)
66                .setInterimResults(true)
67    // .setSingleUtterance(true)
68                .build();
69       }
70
71       public static void addTranscriptMatcher(TranscriptMatcher
             ↪ transcriptMatcher){
72          transcriptMatcherList.add(transcriptMatcher);
73          for(Command command: transcriptMatcher.commands){
74             phrases.addAll(command.getPhrases());
75          }
76          phrasesPreviousCurrentSizes = new int[]{phrasesPreviousCurrentSizes
                ↪ [1], phrases.size()};
77       }
78
79       public static void removeTranscriptMatcher(TranscriptMatcher
             ↪ transcriptMatcher){
80          transcriptMatcherList.remove(transcriptMatcher);
81          for(Command command: transcriptMatcher.commands){
82             phrases.removeAll(command.getPhrases());
83          }
84          phrasesPreviousCurrentSizes = new int[]{phrasesPreviousCurrentSizes
                ↪ [1], phrases.size()};
85       }
86
87       public static void main(InputStream is) {
88          try {
89             setup(is);
90             infiniteStreamingRecognize();
91          } catch (Exception e) {
92             System.out.println("Exception caught: " + e);
93          }
```

```
94    }
95
96    public static String convertMillisToDate(double milliSeconds) {
97       long millis = (long) milliSeconds;
98       DecimalFormat format = new DecimalFormat();
99       format.setMinimumIntegerDigits(2);
100      return String.format(
101            "%s:%s /",
102            format.format(TimeUnit.MILLISECONDS.toMinutes(millis)),
103            format.format(
104                  TimeUnit.MILLISECONDS.toSeconds(millis)
105                        - TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.
                              ↪ toMinutes(millis))));
106   }
107
108   /** Performs infinite streaming speech recognition */
109   public static void infiniteStreamingRecognize() throws Exception {
110      ResponseObserver<StreamingRecognizeResponse> responseObserver;
111
112      try (SpeechClient client = SpeechClient.create(speechSettings)) {
113         ClientStream<StreamingRecognizeRequest> clientStream;
114         responseObserver = new ResponseObserver<StreamingRecognizeResponse
              ↪ >() {
115            ArrayList<StreamingRecognizeResponse> responses = new ArrayList
                 ↪ <>();
116
117            public void onStart(StreamController controller) {
118               referenceToStreamController = controller;
119            }
120
121            public void onResponse(StreamingRecognizeResponse response) {
122               responses.add(response);
123               StreamingRecognitionResult result = response.getResultsList()
                    ↪ .get(0);
124               Duration resultEndTime = result.getResultEndTime();
125               resultEndTimeInMS =
126                     (int)
127                           ((resultEndTime.getSeconds() * 1000) + (
                                 ↪ resultEndTime.getNanos() / 1000000));
128               double correctedTime =
129                     resultEndTimeInMS - bridgingOffset + (STREAMING_LIMIT *
                          ↪  restartCounter);
130
131               SpeechRecognitionAlternative alternative = result.
                    ↪ getAlternativesList().get(0);
132               if (result.getIsFinal()) {
```

```
133                 for(TranscriptMatcher transcriptMatcher :
                        ↪ transcriptMatcherList){
134                     transcriptMatcher.match(alternative.getTranscript().
                            ↪ trim());
135                 }
136                 System.out.print(GREEN);
137                 System.out.print("\033[2K\r");
138                 System.out.printf(
139                         "%s: %s [confidence: %.2f]\n",
140                         convertMillisToDate(correctedTime),
141                         alternative.getTranscript(),
142                         alternative.getConfidence());
143                 isFinalEndTime = resultEndTimeInMS;
144                 lastTranscriptWasFinal = true;
145             } else {
146                 System.out.print(RED);
147                 System.out.print("\033[2K\r");
148                 System.out.printf(
149                         "%s: %s", convertMillisToDate(correctedTime),
                                ↪ alternative.getTranscript());
150                 lastTranscriptWasFinal = false;
151             }
152         }
153
154         public void onComplete() {
155             System.out.println("COMPLETEEEEEE");
156         }
157
158         public void onError(Throwable t) {
159             System.out.println(t.getMessage());
160         }
161     };
162     clientStream = client.streamingRecognizeCallable().splitCall(
            ↪ responseObserver);
163
164     StreamingRecognitionConfig streamingRecognitionConfig =
            ↪ googleSTTSettings();
165     StreamingRecognizeRequest request =
166             StreamingRecognizeRequest.newBuilder()
167                     .setStreamingConfig(streamingRecognitionConfig)
168                     .build(); // The first request in a streaming call has
                            to be a config
169
170     clientStream.send(request);
171
172     try {
173         long startTime = System.currentTimeMillis();
```

```
174
175             while (true) {
176
177                 long estimatedTime = System.currentTimeMillis() - startTime;
178
179                 if (estimatedTime >= STREAMING_LIMIT || (
                        ↪ phrasesPreviousCurrentSizes[0] !=
                        ↪ phrasesPreviousCurrentSizes[1])) {
180
181                     clientStream.closeSend();
182                     referenceToStreamController.cancel(); // remove Observer
183
184                     if (resultEndTimeInMS > 0) {
185                         finalRequestEndTime = isFinalEndTime;
186                     }
187                     resultEndTimeInMS = 0;
188
189                     lastAudioInput = null;
190                     lastAudioInput = audioInput;
191                     audioInput = new ArrayList<>();
192
193                     restartCounter++;
194
195                     if (!lastTranscriptWasFinal) {
196                         System.out.print('\n');
197                     }
198
199                     newStream = phrasesPreviousCurrentSizes[0] ==
                            ↪ phrasesPreviousCurrentSizes[1];
200
201                     clientStream = client.streamingRecognizeCallable().
                            ↪ splitCall(responseObserver);
202                     streamingRecognitionConfig = googleSTTSettings();
203                     request =
204                           StreamingRecognizeRequest.newBuilder()
205                                 .setStreamingConfig(streamingRecognitionConfig
                                        ↪ )
206                                 .build();
207
208                     System.out.println(YELLOW);
209                     System.out.printf("%d: RESTARTING REQUEST\n",
                            ↪ restartCounter * STREAMING_LIMIT);
210
211                     startTime = System.currentTimeMillis();
212
213                 } else {
```

```
214
215                     if ((newStream) && (lastAudioInput.size() > 0)) {
216                         // if this is the first audio from a new request
217                         // calculate amount of unfinalized audio from last
                                request
218                         // resend the audio to the speech client before
                                incoming audio
219                         double chunkTime = STREAMING_LIMIT / lastAudioInput.
                                ↪ size();
220                         // ms length of each chunk in previous request audio
                                arrayList
221                         if (chunkTime != 0) {
222                             if (bridgingOffset < 0) {
223                                 // bridging Offset accounts for time of resent
                                        audio
224                                 // calculated from last request
225                                 bridgingOffset = 0;
226                             }
227                             if (bridgingOffset > finalRequestEndTime) {
228                                 bridgingOffset = finalRequestEndTime;
229                             }
230                             int chunksFromMS =
231                                     (int) Math.floor((finalRequestEndTime -
                                            ↪ bridgingOffset) / chunkTime);
232                             // chunks from MS is number of chunks to resend
233                             bridgingOffset =
234                                     (int) Math.floor((lastAudioInput.size() -
                                            ↪ chunksFromMS) * chunkTime);
235                             // set bridging offset for next request
236                             for (int i = chunksFromMS; i < lastAudioInput.size()
                                    ↪ ; i++) {
237                                 request =
238                                         StreamingRecognizeRequest.newBuilder()
239                                                 .setAudioContent(lastAudioInput.get(i
                                                        ↪ ))
240                                                 .build();
241                                 clientStream.send(request);
242                             }
243                         }
244                         newStream = false;
245                     }
246
247                     ByteString tempByteString = ByteString.copyFrom(
                            ↪ sharedQueue.take());
248
249                     request =
250                             StreamingRecognizeRequest.newBuilder().
                                ↪ setAudioContent(tempByteString).build();
```

```
251
252                     audioInput.add(tempByteString);
253                 }
254
255                 clientStream.send(request);
256             }
257         } catch (Exception e) {
258             System.out.println(e.getMessage());
259         }
260     }
261 }
262
263 @Override
264 protected Void doInBackground(InfiniteStreamRecognizeArguments...
      ↪ infiniteStreamRecognizeArguments) {
265     try {
266         setup(infiniteStreamRecognizeArguments[0].inputStream);
267         infiniteStreamingRecognize();
268     } catch (Exception e) {
269         System.out.println("Exception caught: " + e);
270     }
271     return null;
272 }
273 }
```

8.2.3 Εντολή

Η εντολή αποτελείται από μία κανονική έκφραση η οποία είναι μια ακολουθία χαρακτήρων που ορίζουν ένα μοτίβο αναζήτησης, από τον κώδικα που θα εκτελεστεί αν το μοτίβο ταιριάξει με την μεταγραφή απο το Speech-to-Text API και μια λίστα λέξεων και φράσεων που παρέχουν συμβουλές στην υπηρεσία για την εργασία αναγνώρισης ομιλίας.

Δείγμα κώδικα:

```
1 public class Command {
2     public Pattern pattern;
3     private Matcher matcher;
4     private CommandExecutor commandExecutor;
5     private List<String> phrases;
6
7     public Command(String regex, CommandExecutor commandExecutor, int flags,
      ↪ List<String> phrases){
```

```
8         this.commandExecutor = commandExecutor;
9         this.phrases = phrases;
10        try{
11           pattern = Pattern.compile(regex, flags);
12        }
13        catch(PatternSyntaxException pSE){
14           System.out.println("Incorrect Regular Expression: " + pSE.
                 ↪ getMessage());
15        }
16     }
17
18     public Boolean match(String text){
19        matcher = pattern.matcher(text);
20        return matcher.matches();
21     }
22
23     public Boolean find(String text){
24        matcher = pattern.matcher(text);
25        return matcher.find();
26     }
27
28     public void execute(List<String> list){
29        commandExecutor.execute(list);
30     }
31
32     public List<String> groups(){
33        List<String> groups = new ArrayList<>();
34        if(matcher != null){
35           int noGroups = matcher.groupCount();
36           for(int i=0; i <= noGroups; i++) {
37              groups.add(matcher.group(i));
38           }
39        }
40        else{
41           System.out.println("Couldn't match!");
42        }
43        return groups;
44     }
45
46     public List<String> getPhrases(){
47        if(phrases == null){
48           phrases = new ArrayList<>();
49        }
50        return phrases;
51     }
52  }
```

8.2.4 Εκτελεστής Εντολών

Ο εκτελεστής εντολών εκτελεί τη συνάρτηση που δηλώνεται όταν το μοτίβο της εντολής ταιριάξει με την μεταγραφή απο την υπηρεσία Speech-to-Text.

Δείγμα κώδικα:

```
1 public abstract class CommandExecutor {
2
3     public abstract void execute(List<String> list);
4 }
```

8.2.5 Ταιριαστής Εντολών

Ο ταιριαστής εντολών ταιριάζει την μεταγραφή απο την υπηρεσία Speech-to-Text πάνω στο μοτίβο και καλεί τον εκτελεστή εντολών αν ταιριάξει επιτυχώς.

Δείγμα κώδικα:

```
1  public class TranscriptMatcher {
2     public List<Command> commands;
3
4     public TranscriptMatcher(List<Command> commands){
5        this.commands = commands;
6     }
7
8     public void match(String transcript){
9        for(Command command : commands){
10          if(command.match(transcript)){
11             command.execute(command.groups());
12             break;
13          }
14       }
15    }
16 }
```

Κεφάλαιο 9

Μελισσοκομική Εφαρμογή

Η εφαρμογή είναι το πρακτικό παράδειγμα χρήσης της βιβλιοθήκης Speech-to-Command.

9.1 Υλοποιήση φωνητικών εντολών

Δείγμα κώδικα της υλοποιήσης των φωνητικών εντολών:

- Μελίσσι

```
1  Command melissiCommand = new CommandBuilder().setCommandExecutor(new
       ↪ CommandExecutor() {
2          @Override
3          public void execute(List<String> list) {
4             Beehive beehive = db.beehiveDao().findBeehiveId(Integer.
                  ↪ parseInt(list.get(1)));
5             if(beehive != null) {
6                Intent intent = new Intent(MainActivity.this,
                     ↪ BeehiveChecksActivity.class);
7                intent.putExtra(BeehiveChecksActivity.
                     ↪ EXTRA_BEEHIVE_NUMBER, beehive.number);
8                intent.putExtra(BeehiveChecksActivity.EXTRA_BEEHIVE_ID,
                     ↪  beehive.id);
9                MainActivity.this.startActivity(intent);
10            }
11            else {
12               Toast.makeText(MainActivity.this, "Μελίσσι "+list.get
                     ↪ (1)+" δενυπάρχει!", Toast.LENGTH_LONG).show();
13            }
14         }
```

```
15      }).setRegex("μελιίσσι[]\\s+(\\d+)").setFlags(Pattern.
            ↪ CASE_INSENSITIVE|Pattern.DOTALL)
16          .setPhrases(beehivePhrases).build();
```

- Πλαίσια

```
1       Command frameCommand = new CommandBuilder().setCommandExecutor(
            ↪ new CommandExecutor() {
2           @Override
3           public void execute(final List<String> list) {
4               Log.i("Πλαισια:", list.get(1));
5               NewEditBeehiveCheckActivity.this.runOnUiThread(new
                    ↪ Runnable() {
6                   public void run() {
7                       editFrames.setText(list.get(1));
8                       Toast.makeText(NewEditBeehiveCheckActivity.this, "
                            ↪ Πλαίσιο " + list.get(1) + "!", Toast.
                            ↪ LENGTH_LONG).show();
9                   }
10              });
11          }
12      }).setRegex("πλαιίσιοα[][]\\s+(\\d+)").setFlags(Pattern.
            ↪ CASE_INSENSITIVE | Pattern.DOTALL)
13          .setPhrases(framePhrases).build();
```

- Πλυθησμός

```
1       Command populationCommand = new CommandBuilder().
            ↪ setCommandExecutor(new CommandExecutor() {
2           @Override
3           public void execute(final List<String> list) {
4               Log.i("Πληθυσμός:", list.get(1));
5               NewEditBeehiveCheckActivity.this.runOnUiThread(new
                    ↪ Runnable() {
6                   public void run() {
7                       editPopulation.setText(list.get(1));
8                       Toast.makeText(NewEditBeehiveCheckActivity.this, "
                            ↪ Πληθυσμός " + list.get(1) + "!", Toast.
                            ↪ LENGTH_LONG).show();
9                   }
10              });
11          }
12      }).setRegex("πληθυσμοός[]\\s+(\\d+)").setFlags(Pattern.
            ↪ CASE_INSENSITIVE | Pattern.DOTALL)
13          .setPhrases(populationPhrases).build();
```

- Γόνος

```
Command progenyCommand = new CommandBuilder().setCommandExecutor
    ↪ (new CommandExecutor() {
   @Override
   public void execute(final List<String> list) {
      Log.i("ΤΟΝΟΣ:", list.get(1));
      NewEditBeehiveCheckActivity.this.runOnUiThread(new
          ↪ Runnable() {
         public void run() {
            editProgeny.setText(list.get(1));
            Toast.makeText(NewEditBeehiveCheckActivity.this, "
                ↪ ΤΟΝΟΣ " + list.get(1) + "!", Toast.LENGTH_LONG
                ↪ ).show();
         }
      });
   }
}).setRegex("γόονος[]\\s+(\\d+)").setFlags(Pattern.
    ↪ CASE_INSENSITIVE | Pattern.DOTALL)
      .setPhrases(progenyPhrases).build();
```

- Μέλι

```
Command honeyCommand = new CommandBuilder().setCommandExecutor(
    ↪ new CommandExecutor() {
   @Override
   public void execute(final List<String> list) {
      Log.i("Μέλι:", list.get(1));
      NewEditBeehiveCheckActivity.this.runOnUiThread(new
          ↪ Runnable() {
         public void run() {
            editHoney.setText(list.get(1));
            Toast.makeText(NewEditBeehiveCheckActivity.this, "
                ↪ Μέλι " + list.get(1) + "!", Toast.LENGTH_LONG)
                ↪ .show();
         }
      });
   }
}).setRegex("μέελι[]\\s+(\\d+)").setFlags(Pattern.
    ↪ CASE_INSENSITIVE | Pattern.DOTALL)
      .setPhrases(honeyPhrases).build();
```

- Γύρη

```
Command pollenCommand = new CommandBuilder().setCommandExecutor(
    ↪ new CommandExecutor() {
   @Override
```

```
3          public void execute(final List<String> list) {
4             Log.i("Γύρη:", list.get(1));
5             NewEditBeehiveCheckActivity.this.runOnUiThread(new
                  ↪ Runnable() {
6                public void run() {
7                   editPollen.setText(list.get(1));
8                   Toast.makeText(NewEditBeehiveCheckActivity.this, "
                        ↪ Γύρη " + list.get(1) + "!", Toast.LENGTH_LONG)
                        ↪ .show();
9                }
10            });
11         }
12      }).setRegex("γύυρη[]\\s+(\\d+)").setFlags(Pattern.
            ↪ CASE_INSENSITIVE | Pattern.DOTALL)
13            .setPhrases(pollenPhrases).build();
```

- Παρατήρηση

```
1       Command noteCommand = new CommandBuilder().setCommandExecutor(
            ↪ new CommandExecutor() {
2          @Override
3          public void execute(final List<String> list) {
4             Log.i("Παρατήρηση:", list.get(1));
5             NewEditBeehiveCheckActivity.this.runOnUiThread(new
                  ↪ Runnable() {
6                public void run() {
7                   if(editNotes.getText().length() > 0){
8                      editNotes.append("," + list.get(1));
9                   }
10                  else{
11                     editNotes.setText(list.get(1));
12                  }
13                  Toast.makeText(NewEditBeehiveCheckActivity.this, "
                        ↪ Παρατήρηση " + list.get(1) + "!", Toast.
                        ↪ LENGTH_LONG).show();
14               }
15            });
16         }
17      }).setRegex("παρατήηρηση[]\\s+(.*)").setFlags(Pattern.
            ↪ CASE_INSENSITIVE | Pattern.DOTALL)
18            .setPhrases(notesPhrases).build();
```

9.2 Παράδειγμα χρήσης

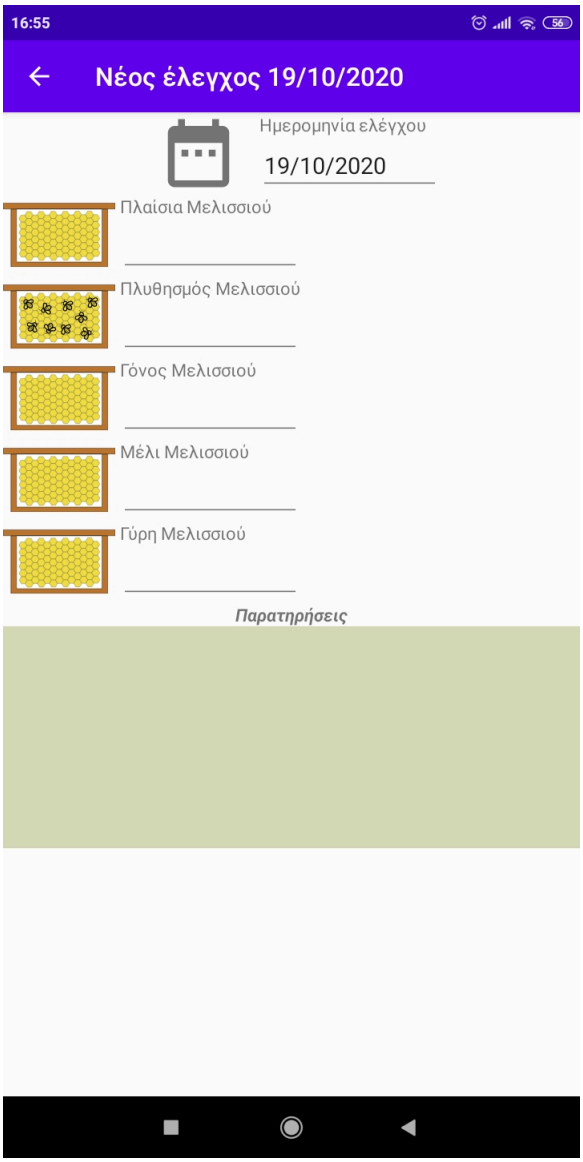
Το παρακάτω παράδειγμα χρήσης είναι για την δοκιμή των υπηρεσιών της βιβλιοθήκης μέσω της εφαρμογής. Σε αυτό το παράδειγμα υλοποιείται μια φυσιολογική ροή με σωστή χρήση των φωνητικών εντολών. Κεντρικό γραφικό

Διάγραμμα 9.1: Κεντρικό γραφικό της εφαρμογής

Έναρξη νέου ελέγχου μελισσιού μέσω φωνητικών εντολών:

Φωνητική εντολή: Μελίσσι 1

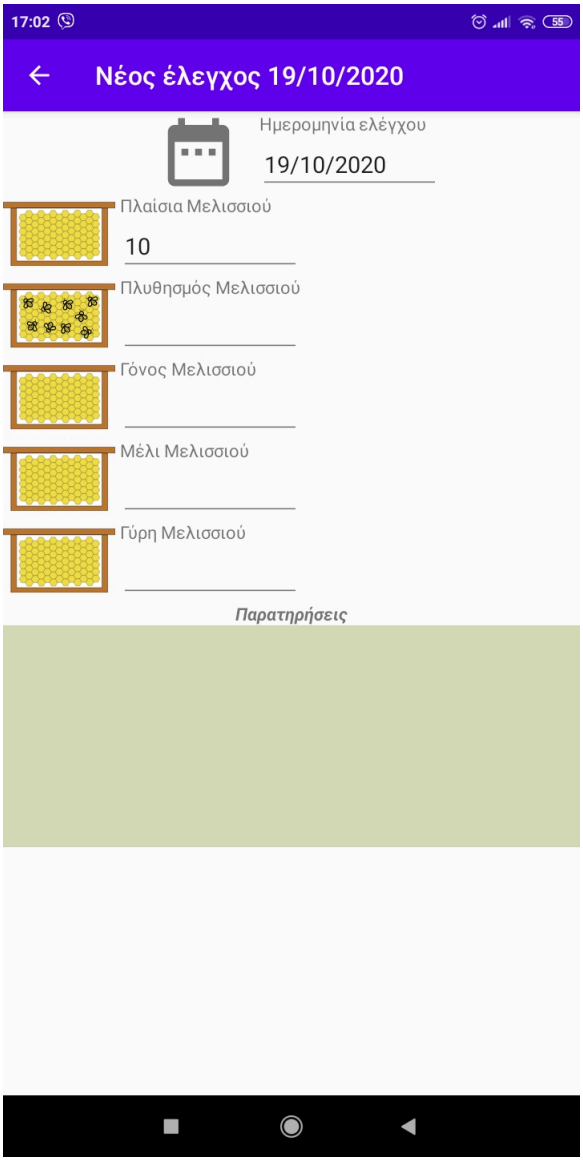
Γραφικό:



Διάγραμμα 9.2: Νέος έλεγχος μελισσιού

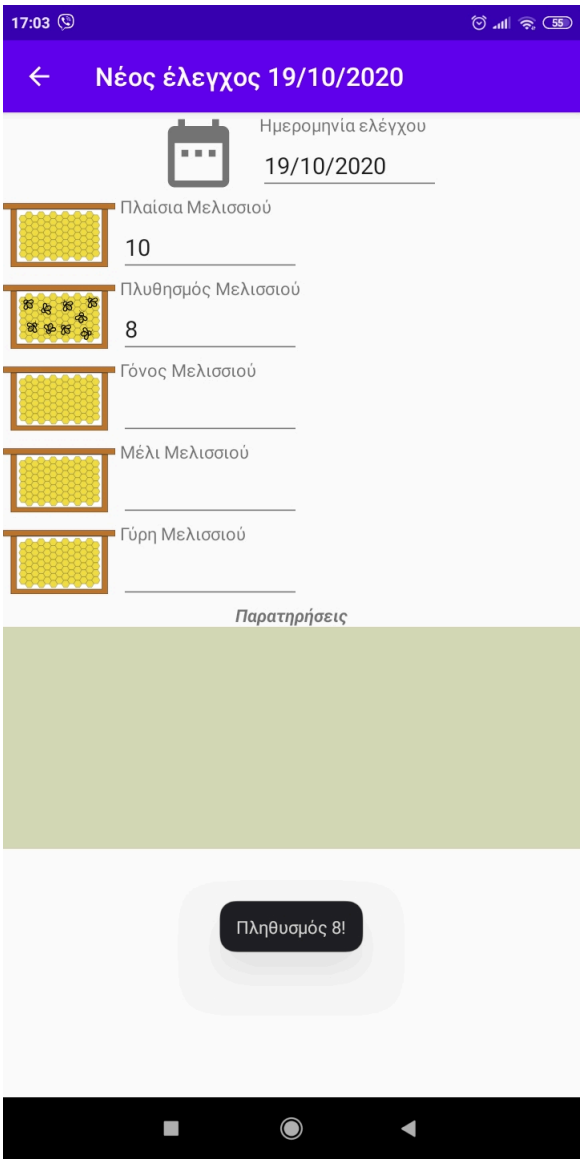
Φωνητικές εντολές για το γέμιστα των πεδιών του γραφικού

1. Φωνητική εντολή: Πλαίσια 10



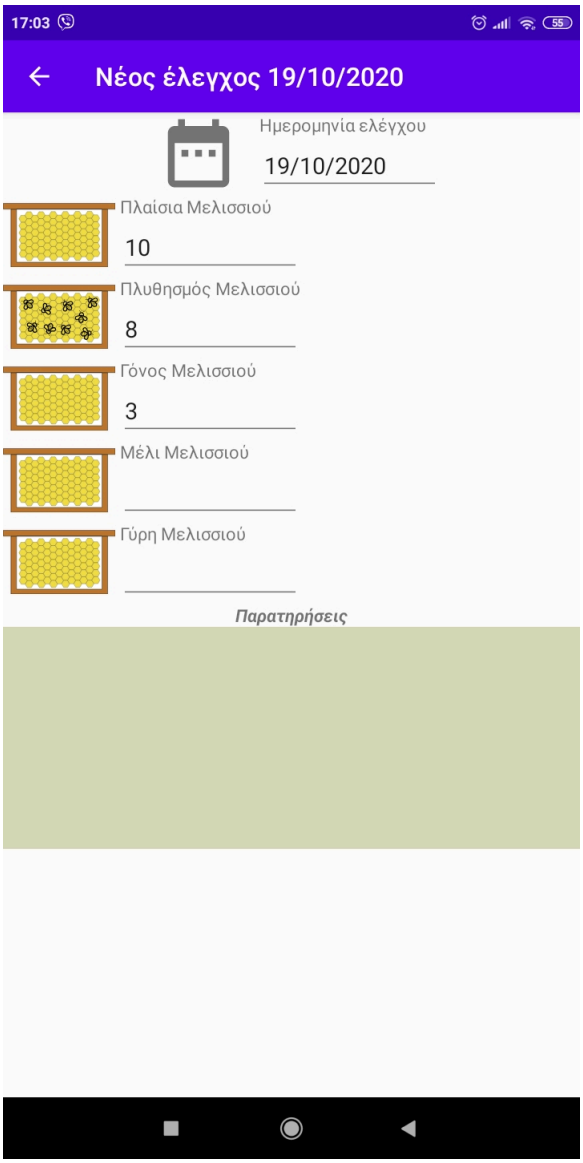
Διάγραμμα 9.3: Προσθήκη αριθμού πλαισίων

2. Φωνητική εντολή: Πλυθησμός 8



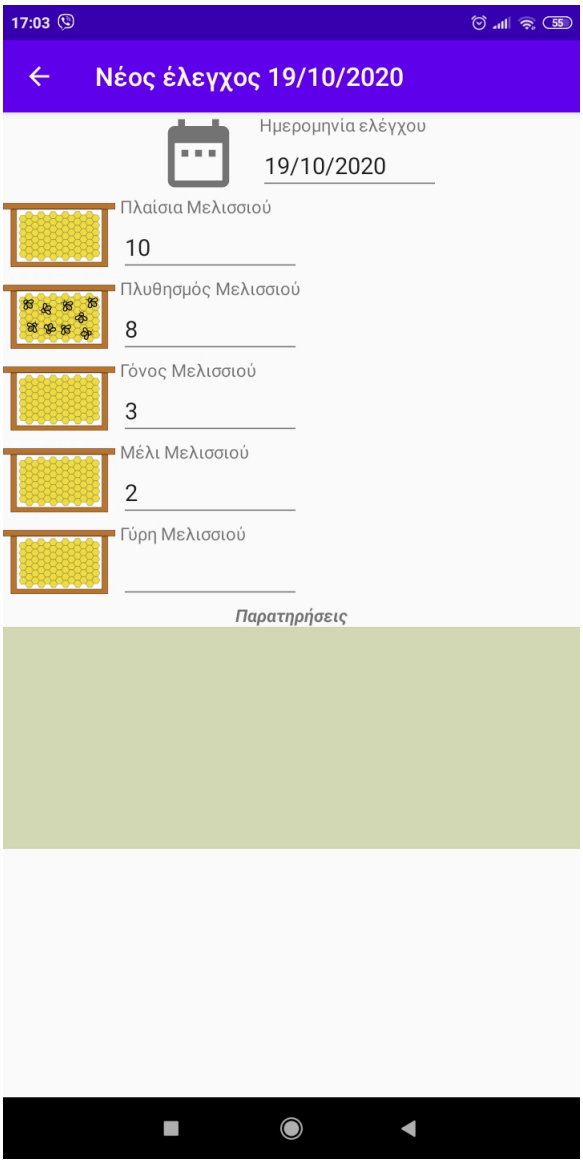
Διάγραμμα 9.4: Προσθήκη αριθμού πλαισίων πλυθησμού

3. Φωνητική εντολή: Γόνος 3



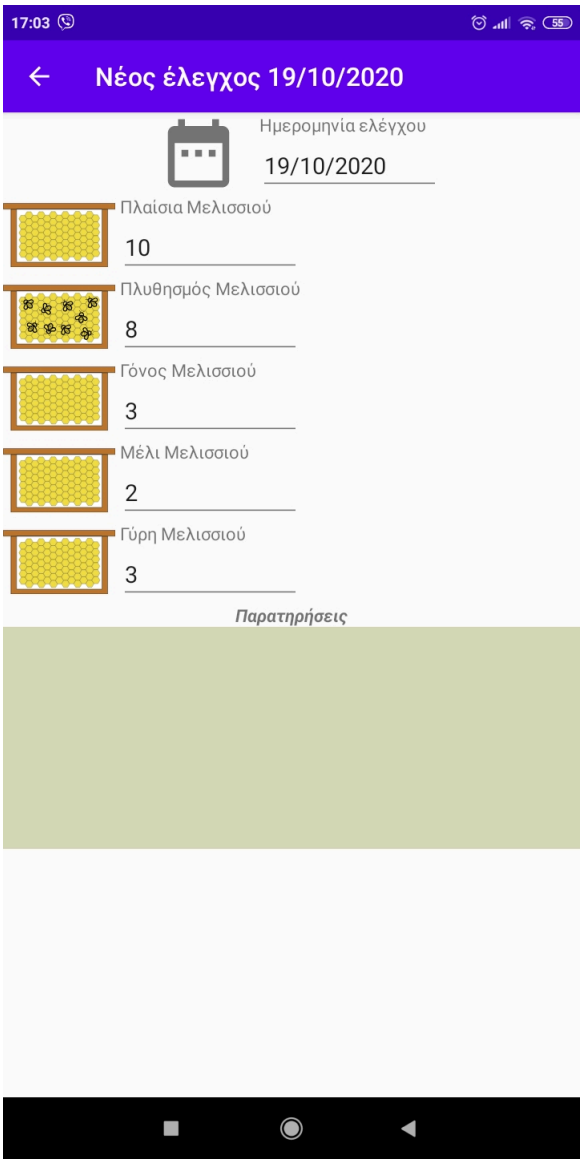
Διάγραμμα 9.5: Προσθήκη αριθμού πλαισίων γόνου

4. Φωνητική εντολή: Μέλι 2



Διάγραμμα 9.6: Προσθήκη αριθμού πλαισίων μελιού

5. Φωνητική εντολή: Γύρη 3



Διάγραμμα 9.7: Προσθήκη αριθμού πλαισίων γύρης

6. Φωνητική εντολή: Παρατήρηση θέλει νέο όροφο

Διάγραμμα 9.8: Προσθήκη παρατήρησης

Κεφάλαιο 10

Συμπεράσματα

10.1 Στόχοι

Κατά την διάρκεια εκπόνησης της πτυχιακής εργασίας, επιτεύχθηκε ο κύριος στόχος που αρχικά είχαμε θέσει, αυτός της δημιουργίας μιας βιβλιοθήκης και μιας εφαρμογή που την χρησιμοποιεί. Η δημιουργία της βιβλιοθήκης και της εφαρμογής δεν ήταν οι μοναδικοί στόχοι που επιτεύχθηκαν. Σε προσωπικό επίπεδο τα κέρδη ήταν πολλαπλά. Η μελέτη και η έρευνα της βιβλιογραφίας σε τέτοιο βαθμό ήταν κάτι πρωτόγνωρο αλλά και παράλληλα εποικοδομητική καθώς βάσει αυτής γράφτηκε το παρόν βιβλίο με τις όποιες τεκμηριώσεις. Επίσης η αναπτύξη αυτού του εργαλειού μπορεί να εφαρμοστεί σε πραγματικά προβλήματα όπως αυτά του μελισσοκόμου που σύμβαλε στην ανάπτυξη της εφαρμαγής για την δοκιμή της βιβλιοθήκης.

10.2 Δυσκολίες

Κατά την διάρκεια ανάπτυξης της εργασίας, η μοναδικη δυσκολία που παρουσιάστηκε ήταν τεχνικής φύσεως. Συγκεκριμένα, στην ανάπτυξη της βιβλιοθήκης έτσι ώστε να είναι φιλική προς τον προγραμματιστή όταν την χρησιμοποιήσει.

10.3 Προτάσεις εξέλιξεις

Όπως είναι φυσιολογικό, η βιβλιοθήκη επιδέχεται βελτίωσης αλλά και νέων προσθηκών που θα το κάνουν πιο δυναμικό αλλά και πιο φιλικό προς τον χρήστη. Ο πη-

γαίος κώδικας όπως και οποιαδήποτε πρόσθετη πληροφορία υπάρχει στην διεύθυνση `https://github.com/EvanPetr/Thesis` Αναφορικά μερικές προτάσεις εξέλιξης είναι οι εξής:

- Δημιουργία κλάσεων για τα exceptions της βιβλιοθήκης για την καλύτερη καθοδήγηση του χρήστη στην επίλυση των προβλημάτων.
- Δυνατότητα χρησιμοποιήσεις διαφορετικού Speech-to-text API από αυτό της Google.
- Υλοποιήση Testing στην βιβλιοθήκη για πρόληψη σφάλματων.

Γλωσσάρι

AI Artificial Intelligence

API Application Programming Interface

GUI Graphical User Interface

REST REpresentational State Transfer

gRPC google Remote Procedure Calls

FLAC Free Lossless Audio Codec

WAV Waveform Audio File Format

URI Uniform Resource Identifier

ASR Automatic Speech Recognition

Βιβλιογραφία

Android (2020). *Android developer*. URL: https://developer.android.com (επίσκεψη 22/03/2020).

Github (2020). *Infinite Stream Recognize*. URL: https://github.com/GoogleCloudPlatfo java-docs-samples/blob/master/speech/cloud-client/src/main/java/com/example/speech/InfiniteStreamRecognize.java (επίσκεψη 30/06/2020).

Google (2020a). *Google cloud console*. URL: https://console.cloud.google.com/home/dashboard (επίσκεψη 30/03/2020).

— (2020b). *Speech-to-Text API*. URL: https://cloud.google.com/speech-to-text (επίσκεψη 22/04/2020).

Guru, Refactoring (2020). *Refactoring Guru-Desing pattern Builder*. URL: https://refactoring.guru/design-patterns/builder (επίσκεψη 30/03/2020).

IBM (2020). *watson Speech-to-Text API*. URL: https://www.ibm.com/cloud/watson-speech-to-text (επίσκεψη 22/04/2020).

Microsoft (2020). *Speech-to-Text API*. URL: https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text/ (επίσκεψη 22/04/2020).

Simpson, J. (2019). *5 best speech to text apis*. URL: https://nordicapis.com/5-best-speech-to-text-apis/ (επίσκεψη 22/04/2020).

Η εργασία αυτή στοιχειοθετήθηκε με το πρόγραμμα XeLaTeX. Για τη στοιχειοθέτηση της βιβλιογραφίας χρησιμοποιήθηκε το πρόγραμμα `biber` και `biblatex`. Οι γραμματοσειρές που χρησιμοποιήθηκαν είναι οι Times New Roman και `Courier New`.