Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης
Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

Τίτλος:
Ανάπτυξη Εφαρμογής σε Android.
Το Ειδικό Λεξικό Εκμάθησης Αγγλικών Quick Speak.

Τσαμπακιούρης Πέτρος
3342

Επιβλέπων εκπαιδευτικός : Κωστής Αϊβαλής

Επιτροπή Αξιολόγησης : Νικόλαος Βιδάκης, Νικόλαος Παπαδάκης

Ημερομηνία Παρουσίασης : 5 / 6 / 2019

Abstract

Language learning can be an intimidating task to accomplish at first. Most people find hard to devote time and effort to learn a new language, because a pretty good knowledge of its words and grammar is required, to be used it for basic communication at least. This difficulty also varies by the nature of the language itself each time (languages with different alphabet for example). Quick Speak, the Android app presented in this thesis, tries to resolve this problem, by getting the user into the most usual words and phrases, used in English for basic communication. It's based on the fact that human brain can remember words and phrasal patterns, pretty easily when repeated many times.

Σύνοψη

Η εκμάθηση γλωσσών, μπορεί να είναι μία εκφοβιστική εργασία να πραγματοποιηθεί στην αρχή. Ο περισσότερος κόσμος, βρίσκει δύσκολο να αφιερώσει χρόνο και προσπάθεια να μάθει μια νέα γλώσσα, διότι απαιτείται μια αρκετά καλή γνώση των λέξεων και γραμματικών της, για να χρησιμοποιηθεί για βασική επικοινωνία τουλάχιστον. Η δυσκολία αυτή, ποικίλει επίσης κι από τη φύση της εκάστοτε γλώσσας (γλώσσες με διαφορετικό αλφάβητο για παράδειγμα). Το Quick Speak, η Android εφαρμογή που παρουσιάζεται στην παρούσα πτυχιακή, προσπαθεί να επιλύσει το πρόβλημα αυτό, με το να μάθει στο χρήστη τις πιο συνήθεις λέξεις και φράσεις που χρησιμοποιούνται στα Αγγλικά για βασική επικοινωνία. Βασίζεται στο γεγονός, ότι ο ανθρώπινος εγκέφαλος, μπορεί να θυμάται λέξεις και φραστικά πρότυπα, αρκετά εύκολα, όταν αυτά επαναλαμβάνονται πολλές φορές.

Πίνακας Περιεχομένων

Abstract i
Σύνοψη ii
Πίνακας Περιεχομένων iii
1 Εισαγωγή 1
1.1 Το Android 1
1.1.1 Η Αρχιτεκτονική του Android 2
Εικόνα 1: Η στοίβα λογισμικού του Android 2
1.1.2 Ανάπτυξη Εφαρμογών Android 4
1.2 Το Ειδικό Λεξικό Quick Speak 4
1.3 Δομή Εργασίας 5
2 Βασικές Αρχές των Android Εφαρμογών 7
2.1 Τα κύρια συστατικά μιας Android εφαρμογής 7
2.1.1 Οι Δραστηριότητες (Activities) 7
2.1.2 Οι Υπηρεσίες (Services) 8
2.1.3 Οι Δέκτες Εκπομπής (Broadcast Receivers) 8
2.1.4 Οι Παροχείς Περιεχομένου (Content Providers) 8
2.2 Επικοινωνία Μεταξύ Συστατικών 9
2.3 Το Αρχείο Μανιφέστου (Manifest File) 9
2.3.1 Δήλωση Συστατικών 9
2.4 Αρχεία Πόρων 10
3 Προετοιμασία Ανάπτυξης του Quick Speak 11
3.1 Αντικειμενοστρεφής Προγραμματισμός 11
3.1.1 Ενθυλάκωση 11
3.1.2 Κληρονομικότητα 12
3.2 Η Γλώσσα Προγραμματισμού Java 12
3.3 Η Γλώσσα XML 13
3.4 Ολοκληρωμένο Περιβάλλον Ανάπτυξης 14
3.5 Το Διάγραμμα Κλάσεων 15
3.6 Το Android SDK 15
3.7 Activities και Views 15
3.7.1 Layouts 16
3.8 Ο Κύκλος Ζωής Ενός Activity 20
3.8.1 Η Κλήση onCreate 21
3.8.2 Η Κλήση onStart 22

3.8.3 Η Κλήση onResume 22
3.8.4 Η Κλήση onPause 22
3.8.5 Η Κλήση onStop 22
3.8.6 Η Κλήση onRestart 23
3.8.7 Η Κλήση OnDestroy 23
3.9 Η Σημασία των Δεδομένων και της Λίστας 24
3.9.1 Ο Προσαρμογέας (Adapter) 24
4.0 Το Θραύσμα (Fragment) 25
4 Υλοποίηση του Quick Speak 28
4.1 Το πρόβλημα των Γλωσσών 28
4.2 Απαιτήσεις Συστήματος 29
4.3 Το Προσχέδιο του Quick Speak 29
4.4 Προετοιμασία του Project στο Android Studio 29
4.5 Το Main Activity 35
4.5.1 Το Αρχείο activity_main 35
4.5.2 Το Αρχείο MainActivity.java 42
4.6 Το WordListActivity 45
4.6.1 Το Αρχείο word_list_activity.xml 45
4.6.2 Το WordListActivity.java 47
4.7 Ανάπτυξη της Βάσης Δεδομένων του Quick Speak 49
4.7.1 Ο Πίνακας της Βάσης Δεδομένων 50
4.7.2 Η Κλάση WordDbHelper 52
5. Δεύτερο Μέρος Υλοποίησης του Quick Speak 57
5.1 Ο CursorAdapter του Quick Speak 57
5.1.1 Η Κλάση MediaPlayer και η Υλοποίηση Ήχου στο Quick Speak 60
5.2 Το WordFragment 63
5.3 Η Υλοποίηση του Navigation Drawer 67
5.3.1 Τοποθέτηση του Menu στο WordListActivity 69
5.3.2 Υλοποίηση της Λογικής του Navigation Drawer 70
5.4 Η Τελική Μορφή του Quick Speak 75
5.4.1 Παρουσίαση Υλοποιημένων Αρχείων 75
5.4.2 Το Διάγραμμα Κλάσεων του Quick Speak 101
5.5 Πιλοτική Επίδειξη του Quick Speak 103
6. Αποτελέσματα και Επίλογος 105
6.1 Συμπεράσματα 105
6.2 Περαιτέρω Εργασία και Πιθανές Επεκτάσεις του Quick Speak 106

6.2.1 Πολυγλωσσικότητα της Εφαρμογής 106
6.2.2 Το Ειδικό Ελληνο-αγγλικό (;) Λεξικό Quick Speak 111
6.3 Επίλογος 117
Βιβλιογραφία 118
ΠΑΡΑΡΤΗΜΑ Α: Το TTSAutomate 126

Λίστα Εικόνων

Εικόνα 1: Η στοίβα λογισμικού του Android 2
Εικόνα 2: Απεικόνιση Κλάσης στη UML.....14
Εικόνα 3: Παράδειγμα δομής Layout.....15
Εικόνα 4: Προβολή Βασικής Διεπαφής Ενός Activity.....16
Εικόνα 5: Αλλαγή Κειμένου Διεπαφής.....18
Εικόνα 6: Απεικόνιση Κύκλου Ζωής Ενός Activity 19
Εικόνα 7: Ο Κύκλος Ζωής Ενός Fragment.....24
Εικόνα 8: Ρύθμιση και Δημιουργία του Quick Speak Project.....28
Εικόνα 9: Το Περιβάλλον του Android Studio.....29
Εικόνα 10: Η Δομή Ενός Android Studio Project 30
Εικόνα 11: Ο Design Editor του Android Studio.....32
Εικόνα 12: Ο Text Editor του XML αρχείου της Διεπαφής ενός Activity.....32
Εικόνες 13 και 14: Τοποθέτηση Νέας Γραμματοσειράς στο Android Studio 36
Εικόνα 15: Τα Constraints του ImageView 38
Εικόνα 16: Τα Constraints του TextView 38
Εικόνα 17: Τα Constraints του ProgressBar.....39
Εικόνα 18: Η Τελική Μορφή της Διεπαφής του MainActivity.....39
Εικόνα 19: Δημιουργία του WordListActivity 41
Εικόνα 20: Το UML του MainActivity 42
Εικόνα 21: Στιγμιότυπο Εκκίνησης Εκτέλεσης Εφαρμογής σε Πραγματική Συσκευή 46
Εικόνα 22: Στιγμιότυπο Εκκίνησης του Quick Speak – MainActivity.....46
Εικόνα 23: Στιγμιότυπο Κυρίως Αρχικού Περιεχομένου του Quick Speak – WordListActivity.....47
Εικόνα 24: Το Διάγραμμα Κλάσεων της Υλοποίησης της Βάσης Δεδομένων του Quick Speak 53
Εικόνα 25: Η Διεπαφή του list_item.xml.....60
Εικόνα 26: Το Διάγραμμα Κλάσεων του Quick Speak.....98
Εικόνα 27: Ανοίγοντας τον Translations Editor του Android Studio.....102
Εικόνα 28 : Ο Translations Editor του Android Studio 103
Εικόνες 29 και 30 : Το Overflow Menu του Quick Speak.....112
Εικόνα 31: Το TSSAutomate.....115
Εικόνες 32, 33 και 34 : Δημιουργία Ηχητικών Αποσπασμάτων στο TTSAutomate.....117

Λίστα Πινάκων

Πίνακας 1: Ο Πίνακας english στο Σχεσιακό Μοντέλο της Βάσης – Ενδεικτικό Παράδειγμα μιας Εγγραφής Του Πίνακα 49

Πίνακας 2: Απεικόνιση Αρχείων του Quick Speak Project.....72

1 Εισαγωγή

Οι έξυπνες συσκευές χειρός (smartphones, tablets, smartwatches κτλ.) είναι ένα από τα σημαντικότερα επιτεύγματα της σύγχρονης τεχνολογίας. Αποτελούν ένα μεγάλο μέρος της καθημερινότητας και είναι πλέον βέβαιο ότι θα συνεχιστεί εντονότερα αυτό και για επόμενα χρόνια. Οι συσκευές αυτές, επιτρέπουν στους χρήστες να κάνουν απλά καθημερινά πράγματα όπως, πλοήγηση στο διαδίκτυο, παρακολούθηση ειδήσεων για ποικίλα θέματα, επικοινωνία, μάθηση, ψυχαγωγία και πολλά άλλα.

Το κύριο χαρακτηριστικό των συσκευών αυτών είναι το λειτουργικό σύστημα τους. Στην ουσία, πρόκειται για υπολογιστές χειρός, οι οποίοι εξελίσσονται ραγδαία σε τέτοιο βαθμό ώστε να έρχεται πολύ κοντά - χωρίς να ξεπερνά βέβαια - στην ποιότητα των σταθερών και φορητών υπολογιστών τόσο στις επιδόσεις όσο και στην εμπειρία χρήστη γενικά. Θα μπορούσε να πει κανείς, ότι τα λειτουργικά συστήματα των συσκευών χειρός (Android, iOS και Windows Phone) δεν έχουν να «ζηλέψουν» και πολλά πράγματα από εκείνα των προσωπικών κι όχι μόνο υπολογιστών. Στην παρούσα πτυχιακή, θα εστιάσουμε τη προσοχή μας στο λειτουργικό σύστημα Android, εξ αυτών.

1.1 Το Android

Το Android[1], είναι ένα λειτουργικό σύστημα της εταιρίας Google[2], ειδικά σχεδιασμένο για φορητές συσκευές, όπως κινητά, tablets και smartwatches (έξυπνα ρολόγια) πλέον. Ιστορικά η κυκλοφορία Android συσκευών ξεκίνησε γύρω στο 2008. Το λειτουργικό, συνοδεύεται συνήθως από προ-εγκατεστημένο λογισμικό (εφαρμογές) της Google, όπως, το Gmail, το Google Search, το Google Play, το Google Music κτλ[3]. Το λογισμικό χρησιμοποιείται από τους χρηστές για διάφορες εργασίες όπως κλήσεις, μηνύματα, παρακολούθηση ειδήσεων, διαχείριση email, ψυχαγωγία κτλ. Το Android, θεωρείται το πιο δημοφιλές λειτουργικό στο κόσμο και τη στιγμή που γράφεται το παρόν έγγραφο, υπάρχουν πάνω από 2 δισεκατομμύρια ενεργοί χρήστες Android συσκευών στο κόσμο και περίπου πάνω από 2.6 εκατομμύρια διαθέσιμες εφαρμογές[1].

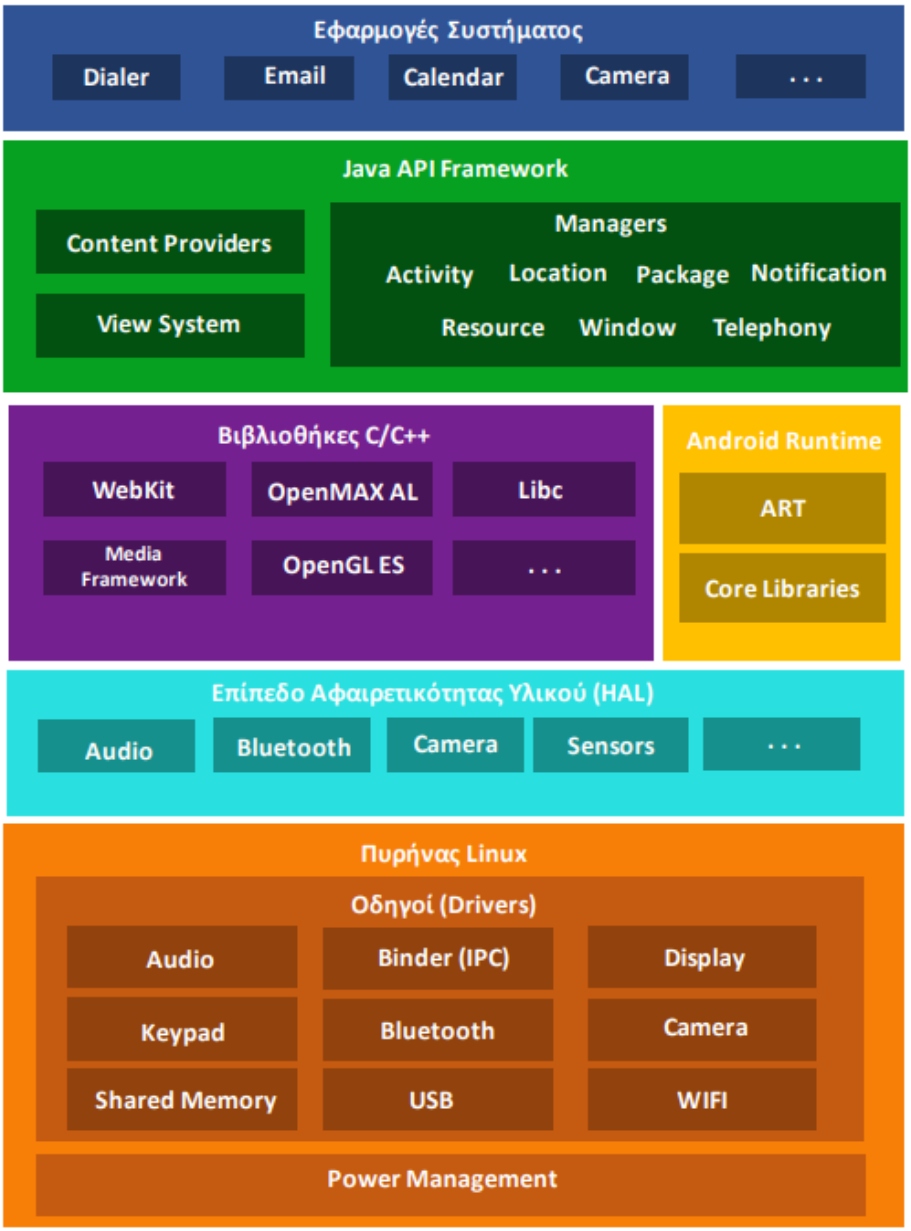
Μερικά από τα χαρακτηριστικά[1] του Android είναι τα εξής:

- Πρόσβαση στο διαδίκτυο με χρήση φιλομετρητών ιστού (web browsers) ή web εφαρμογών.
- Αυτόματη διόρθωση λέξεων μέσω λεξικών εγκατεστημένα στο σύστημα. (Auto correction) Υποστήριξη πολλαπλών γλωσσών.
- Φωνητικές εντολές (συνήθως με χρήση του Google App)
- Πολυεπεξεργασία εφαρμογών, η δυνατότητα να τρέχουν στο σύστημα παραπάνω από μια εφαρμογές ταυτόχρονα καθιστώντας επίσης, εύκολη και γρήγορη την εναλλαγή μεταξύ αυτών.
- Βοηθήματα για άτομα με ειδικές ανάγκες όπως, φωνητική ανάγνωση κειμένου για άτομα με χαμηλή έως μη επαρκή όραση (Google TalkBack)

- Αναπαραγωγή πολυμέσων ήχου και εικόνας.

1.1.1 Η Αρχιτεκτονική του Android

Το Android, αποτελεί μια στοίβα λογισμικού ανοιχτού κώδικα (open source) βασισμένη σε πυρήνα Linux[4]. Κάθε επίπεδο της στοίβας αυτής, αποτελεί κόμματι των λειτουργικών στοιχείων του συστήματος. Χαμηλά στην ιεραρχία βρίσκονται τα κομμάτια που αφορούν την άμεση διαχείριση του υλικού της συσκευής (hardware). Στα υψηλά επίπεδα μιλάμε φυσικά για τα κομμάτια τα οποία βλέπει και χρησιμοποιεί ο χρήστης, όπως είναι το γραφικό περιβάλλον (UI) για παράδειγμα. Μερικά από τα συστατικά της στοίβας λογισμικού του Android, παρουσιάζονται στο παρακάτω σχήμα:



Εικόνα 1: Η στοίβα λογισμικού του Android

1.1.1.1 Ο Πυρήνας Linux

Η «καρδιά» του Android, είναι ο πυρήνας Linux[5] [6]. Όπως φαίνεται και στο σχήμα, ο πυρήνας είναι υπεύθυνος για τις λειτουργίες χαμηλού επιπέδου όπως, τη διαχείριση των προγραμμάτων οδήγησης (drivers) των συστατικών της συσκευής (ήχος, κάμερα, Bluetooth κτλ.) και τη διαχείριση ενέργειας της συσκευής.

1.1.1.2 Το Hardware Abstraction Layer (HAL)

Το HAL αποτελείται από ένα σύνολο βιβλιοθηκών που χρησιμοποιούνται από το Java API Framework. Στην ουσία υπάρχει μια βιβλιοθήκη (library module) για κάθε κομμάτι του hardware της συσκευής όπως για παράδειγμα η κάμερα ή το Bluetooth, η οποία υλοποιεί μια διασύνδεση με το κομμάτι αυτό. Όταν το Java API κάνει μια κλήση ζητώντας πρόσβαση στο hardware της συσκευής, το Android φορτώνει από το επίπεδο αυτό την αντίστοιχη βιβλιοθήκη για το hardware που ζητήθηκε.

1.1.1.3 Το Android Runtime

Κάθε εφαρμογή, τρέχει σε μια δική της διεργασία, πάνω στο Android Runtime (ART) . Το ART[7] τρέχει εικονικές μηχανές (virtual machines) μέσω DEX αρχείων. Ένα DEX αρχείο, είναι ένας κώδικας byte (bytecode) ειδικά σχεδιασμένος για το Android ώστε να τρέχει πάνω σε συσκευές με μικρό μέγεθος μνήμης. Οι εκδόσεις κάτω του Android 5.0, χρησιμοποιούν το Dalvik ως runtime. Μια εφαρμογή που τρέχει καλά σε ένα ART σύστημα, μπορεί να τρέξει και στο Dalvik, αλλά το αντίθετο δεν ισχύει πάντα.

1.1.1.4 Οι Βιβλιοθήκες C / C++

Τα περισσότερα από τα κομμάτια εκείνα που υλοποιούν τα στοιχεία των χαμηλών επιπέδων που έχουμε δει έως τώρα, είναι υλοποιημένα από βιβλιοθήκες της γλώσσας C και C++. Το Android χρησιμοποιεί το Java API Framework για να αξιοποιήσει τις λειτουργίες των βιβλιοθηκών αυτών για τις εφαρμογές χρήστη.

1.1.1.5 Το Java API Framework

Το Java API Framework (Application Programming Interface[8]) περιέχει ένα σύνολο βιβλιοθηκών Java, ειδικά σχεδιασμένες για ανάπτυξη και εκτέλεση εφαρμογών. Όλες οι εφαρμογές που ίσως έχει χρησιμοποιήσει μέχρι στιγμής ο αναγνώστης, περιέχουν στοιχεία του Java API Framework. Μερικά από τα στοιχεία του Java API Framework που χρησιμοποιούν τα Android apps είναι :

- Το View System μέσω του οποίου σχεδιάζονται components (τα λεγόμενα Views) που συνθέτουν τη διεπαφή μιας εφαρμογής.

- Ο Resource Manager, που αναλαμβάνει να διαχειριστεί τους πόρους που χρησιμοποιεί μια εφαρμογή όπως, οι εικόνες, τα βίντεο κτλ.
- Ο Notification Manager που διαχειρίζεται τις ειδοποιήσεις των εφαρμογών.
- Ο Activity Manager που διαχειρίζεται το κύκλο ζωής των Activities των εφαρμογών.

1.1.1.6 Οι Εφαρμογές Συστήματος

Σε αυτό το επίπεδο μιλάμε πλέον για το χρήστη της συσκευής. Εδώ εγκαθίστανται όλες οι εφαρμογές που χρησιμοποιεί, καθώς επίσης υπάρχουν και οι ήδη προ-εγκατεστημένες της Google όπως το Play Store, το Gmail, το YouTube κτλ. Επίσης σε αυτό το επίπεδο, βρίσκονται και οι εφαρμογές διαχείρισης της συσκευής όπως είναι οι ρυθμίσεις της, οι διαχείριση επαφών, η πραγματοποίηση κλήσεων, το πληκτρολόγιο κτλ.

1.1.2 Ανάπτυξη Εφαρμογών Android

Ο προγραμματισμός των Android εφαρμογών[9], έχει τεράστιο αντίκτυπο στην εξέλιξη όχι μόνο των εφαρμογών αλλά και του ίδιου του λειτουργικού συστήματος. Οι ανάγκες των χρηστών, συνεχώς αυξάνονται, με τέτοιο ρυθμό ώστε να αναπτύσσονται διαρκώς, διάφορες ακόμα και πρωτότυπες εφαρμογές, οι οποίες προσπαθούν να βελτιώσουν την εμπειρία χρήστη. Το κύριο χαρακτηριστικό των εφαρμογών είναι η ευκολία χρήσης τους. Ενέργειες που θα απαιτούσαν έναν αριθμό βημάτων σε έναν υπολογιστή γραφείου ακόμα και φορητού (laptop), στις Android συσκευές πραγματοποιούνται απλά και μόνο με πατήματα στην οθόνη. Οι διεπαφές σε ένα Android σύστημα ακολουθούν την αρχή της απλότητας, δεν είναι ιδιαίτερα πολύπλοκες (αναλόγως βέβαια την εφαρμογή και το σκοπό της). Επίσης η ταχύτητα παίζει κι αυτή σημαντικό ρόλο. Σήμερα το Android είναι αρκετά γρήγορο σαν σύστημα, ανεξάρτητα από το γεγονός ότι, παράλληλα και το υλικό (hardware) των συσκευών διαρκώς βελτιώνεται σε σημείο που φτάνει πολύ κοντά στη ποιότητα των laptops.

Τα τελευταία χρόνια, η ανάπτυξη εφαρμογών σε Android, έχει προσελκύσει αρκετούς developers, καθώς αποτελεί μια φιλική και όχι τόσο απαιτητική δουλεία, μιας και τα εργαλεία που χρησιμοποιεί, είναι δωρεάν και διαθέσιμα για όλα τα συστήματα υπολογιστών. Ο τρόπος με τον οποίο σχεδιάζεται και αναπτύσσεται μια Android εφαρμογή, είναι αρκετά προσιτός ακόμα και σε άτομα που δεν έχουν ιδιαίτερη προγραμματιστική εμπειρία. Παρόλα αυτά όπως θα δούμε και στο 2ο κεφάλαιο, συνιστώνται κάποιες βασικές γνώσεις ώστε η εμπειρία αυτή να είναι πιο ακόμα ευχάριστη και εύκολη σε βασικό επίπεδο τουλάχιστον.

1.2 Το Ειδικό Λεξικό Quick Speak

Η εφαρμογή που θα αναπτυχθεί στη παρούσα πτυχιακή, ονομάζεται Quick Speak. Όπως ίσως να υπονοεί το όνομα της, στόχος της εφαρμογής αυτής να παρουσιάσει μια μορφή ειδικού λεξικού γρήγορης εκμάθησης της Αγγλικής γλώσσας. Σε αυτό το λεξικό, θα παρουσιάζονται βασικές λέξεις και φράσεις που χρησιμοποιούνται συχνά στα Αγγλικά, χωρισμένες σε επίπεδα δυσκολίας, με επιπλέον δυνατότητα ακρόασης προφορών τους. Απευθύνεται κυρίως, σε άτομα, τα οποία επιθυμούν να μάθουν τα απολύτως βασικά στοιχεία της γλώσσας ώστε να τη

χρησιμοποιήσουν ως τουρίστες στο εξωτερικό για παράδειγμα ή απλώς και για γενική χρήση της στη καθημερινότητα.

Έχοντας υπόψιν το προαναφερθέντα για το προγραμματισμό Android, θα δούμε τα εργαλεία και τις τεχνικές που θα χρησιμοποιηθούν, ώστε να αναπτυχθεί το Quick Speak σε μια stable (σταθερή) μορφή, και επίσης θα παρουσιαστούν τυχόν βελτιώσεις και επεκτάσεις της εφαρμογής.

1.3 Δομή Εργασίας

Η παρούσα πτυχιακή, χωρίζεται σε 6 κεφάλαια, συν του παρόντος. Κάθε κεφάλαιο, παρουσιάζει τόσο τις τεχνικές και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής, όσο και τα βήματα που ακολουθήθηκαν για την υλοποίηση της σε πρακτικό επίπεδο. Σε αυτή την ενότητα θα δούμε συνοπτικά το σκοπό καθενός από αυτά τα κεφάλαια.

- Στο κεφάλαιο 1, κάναμε μια ανασκόπηση για τις φορητές συσκευές χειρός και την σημαντικότητα τους στη σημερινή εποχή. Επίσης εστιάσαμε στο λειτουργικό Android, αναλύοντας περιληπτικά τουλάχιστον, τα σημαντικότερα στοιχεία του. Τέλος αναφερθήκαμε στο βασικό στόχο της πτυχιακής, όπου είναι η ανάπτυξη της εφαρμογής Quick Speak.

- Στο κεφάλαιο 2, θα δούμε τις βασικές αρχές που αφορούν τις Android εφαρμογές, κάνοντας αναφορά στα σημαντικά κομμάτια που συνθέτουν μια ολοκληρωμένη Android εφαρμογή.

- Στο κεφάλαιο 3, θα δούμε τις τεχνικές και τα εργαλεία που θα χρησιμοποιηθούν για την ανάπτυξη του Quick Speak. Με λίγα λόγια θα ασχοληθούμε με τη προετοιμασία της ανάπτυξης του.

- Στο κεφάλαιο 4, θα παρουσιαστεί το 1ο μέρος, της υλοποίησης του Quick Speak, στο οποίο θα σχεδιάσουμε και θα αναπτύξουμε ένα στιγμιότυπο οθόνης καλωσορίσματος (loading screen) το οποίο θα μεταφέρει το χρήστη μετά από σύντομο χρονικό διάστημα σε ένα νέο στιγμιότυπο οθόνης το οποίο αρχικά θα είναι κενό. Επίσης θα ασχοληθούμε με την ανάπτυξη βάσης δεδομένων μέσω Android, την οποία θα χρησιμοποιήσουμε για να αντλήσουμε τα δεδομένα που θα χρησιμοποιηθούν στο Quick Speak.

- Στο κεφάλαιο 5, θα προχωρήσουμε στο 2ο μέρος της ανάπτυξης του Quick Speak. Εδώ θα ολοκληρώσουμε με την υλοποίηση του. Θα παρουσιάσουμε την ανάπτυξη ενός fragment, το οποίο θα αξιοποιεί τα data της βάσης που δημιουργήθηκε στο 1ο μέρος και θα τα παρουσιάζει κατάλληλα στον χρήστη της εφαρμογής. Τέλος θα παρουσιάσουμε όλα τα σημαντικά αρχεία του project της εφαρμογής και των τελικών μορφών τους, και θα πραγματοποιήσουμε μια πιλοτική επίδειξη της εφαρμογής. Να σημειωθεί προκαταβολικά, ότι η μορφή του τελικού project του συνοδευτικού υλικού της πτυχιακής, ενδέχεται να είναι ελαφρώς διαφορετική από αυτή των αρχείων που θα παρουσιαστούν εδώ, καθώς από το επόμενο κεφάλαιο και μετά, αναπτύχθηκε μια επέκταση της εφαρμογής και διορθώθηκαν πιθανά σφάλματα αυτής.

- Στο κεφάλαιο 6, θα κάνουμε μια γενική ανασκόπηση της πτυχιακής, οδηγούμενοι σε συμπεράσματα και αποτελέσματα της ανάπτυξης του Quick Speak. Επιπλέον θα παρουσιάσουμε 2 πιθανές επεκτάσεις της εφαρμογής (η $2^{η}$ χωρίς υλοποίηση), με τις οποίες μπορούμε να υποστηρίξουμε επιπλέον δυνατότητες στην εφαρμογή. Τέλος θα κλείσουμε με έναν επίλογο της παρούσας πτυχιακής.

2 Βασικές Αρχές των Android Εφαρμογών

Οι Android εφαρμογές, αναπτύσσονται με τις γλώσσες προγραμματισμού C++[10], Java[11][12] και Kotlin[13][14]. Το Android SDK[15], είναι μια συλλογή εργαλείων που χρησιμοποιούνται για την ανάπτυξη των εφαρμογών. Με τη χρήση τους, ο κώδικας μιας εφαρμογής, μαζί με όλα τα δεδομένα (data) και τα αρχεία πόρων που χρησιμοποιεί (resource files – εικόνες, βίντεο κτλ.) , συντάσσονται σε ένα APK αρχείο (Android Application Package[16][17]). Για παράδειγμα στη περίπτωση του Quick Speak το αντίστοιχο αρχείο αυτό, θα μπορούσε να είναι το quickspeak.apk . Αυτό το αρχείο, περιέχει όλα τα στοιχεία (πηγαίος κώδικας, αρχεία πόρων, λοιπές βιβλιοθήκες κτλ.) που χρειάζεται το Android σύστημα ώστε να το εγκαταστήσει σε μια συσκευή.

Κάθε εφαρμογή στο Android, ζει σε ένα δικό της «οικοσύστημα» όπως θα μπορούσε να πει κανείς, με την έννοια, ότι υπάρχει μια διεργασία ανά εφαρμογή, και επειδή μιλάμε για Java κώδικα, στην ουσία κάθε εφαρμογή τρέχει πάνω στο δικό της μοναδικό VM (virtual machine [18]). Η κύρια αρχή που ακολουθείται, είναι αυτή του ελάχιστου προνομίου[17], όπως λέγεται. Σύμφωνα με αυτή, κάθε εφαρμογή έχει πρόσβαση μόνο σε πόρους που πραγματικά χρειάζεται και εφόσον ολοκληρώσει τη δουλεία της, οι πόροι αυτοί απελευθερώνονται από το σύστημα. Έτσι εξασφαλίζεται ότι οι εφαρμογές, δεν αποκτούν πρόσβαση σε οτιδήποτε δε τους έχει δοθεί άδεια (τα λεγόμενα permissions) .

2.1 Τα κύρια συστατικά μιας Android εφαρμογής

Όλες οι εφαρμογές στο Android, αποτελούνται από 4 βασικά συστατικά[17], τα οποία την συνθέτουν. Κάποια από αυτά, είναι κύρια για να εισαχθεί ένας χρήστης σε οποιαδήποτε εφαρμογή, ενώ άλλα είναι είτε προαιρετικά, είτε εξαρτόμενα από άλλα.

2.1.1 Οι Δραστηριότητες (Activities)

Τα Activities, είναι το κύριο στοιχείο από το οποία ξεκινά μια εφαρμογή. Είναι στην ουσία το στοιχείο μέσω του οποίου ο χρήστης αρχίζει να αλληλοεπιδρά με αυτήν. Πρόκειται για ένα στιγμιότυπο οθόνης, το οποίο αποτελείται από μια διεπαφή χρήστη με σκοπό να πραγματοποιήσει μια εργασία. Για παράδειγμα, μια εφαρμογή email, θα μπορούσε να χρησιμοποιεί ένα activity για να παρουσιάζει τις λίστες με τα μηνύματα ηλεκτρονικού ταχυδρομείου, ένα άλλο για την ανάγνωση ενός μεμονωμένου μηνύματος και ένα άλλο για τη σύνταξη του. Όλα αυτά τα activities συνεργάζονται μεταξύ τους για να συνθέσουν την ολική εμπειρία χρήστη της εφαρμογής, παρόλα αυτά κάθε ένα είναι ανεξάρτητο από τα άλλα. Μια εφαρμογή μπορεί να αποτελείται από ένα μέχρι και πολλά activities, παρόλα αυτά το πλήθος τους είναι συνήθως μικρός ώστε να διατηρείται η ομαλή λειτουργία τόσο της εφαρμογής όσο και του συστήματος.

Ένα activity, επικοινωνεί με το σύστημα, με τέτοιο τρόπο ώστε, να μπορεί ο χρήστης να επικεντρωθεί στα σημεία της εφαρμογής που τον ενδιαφέρουν κάθε φορά. Επίσης το σύστημα, «θυμάται» και διατηρεί τα σημεία στα οποία βρισκόταν ο χρήστης σε περίπτωση που επιθυμήσει να επιστρέψει σε αυτά. Αν για παράδειγμα, βρισκόμαστε στο activity A και

μεταφερθούμε στο Β, αν θελήσουμε να επιστρέψουμε στο Α, το σύστημα γνωρίζει πώς να το πραγματοποιήσει. Στη παρούσα πτυχιακή θα ασχοληθούμε κυρίως με αυτό το συστατικό. Τα υπόλοιπα συστατικά που παρουσιάζονται παρακάτω, δε θα χρησιμοποιηθούν στη παρούσα πτυχιακή παρόλα αυτά αποτελούν σημαντικό ρόλο στην ανάπτυξη Android εφαρμογών γενικά, για αυτό και παρουσιάζονται εδώ.

2.1.2 Οι Υπηρεσίες (Services)

Τα Services, αποτελούν εργασίες που εκτελούνται στο παρασκήνιο μιας εφαρμογής. Συνήθως πρόκειται για εργασίες που απαιτούν αρκετό χρονικό διάστημα για να ολοκληρωθούν ή είναι απομακρυσμένες από το σύστημα που τις εκτελεί (παρακολούθηση κατάστασης λήψης αρχείου από server για παράδειγμα) . Δεν παρέχουν κάποια διεπαφή χρήστη, όπως τα activities, παρόλα αυτά, ένα activity μπορεί να ξεκινήσει ένα service. Για παράδειγμα, μια εφαρμογή ακροάσεως μουσικής, μπορεί να εκκινήσει ένα service, το οποίο θα διαχειρίζεται την αναπαραγωγή ενός κομματιού στο παρασκήνιο ακόμα κι όταν δεν είναι εστιασμένος ο χρήστης στην εφαρμογή. Αν το σύστημα κρίνει ότι ένα service, καταναλώνει αρκετούς πόρους μνήμης σε σημείο που θα προκαλέσει πρόβλημα για άλλες διεργασίες, είναι στη θέση να τερματίσει το service πριν ολοκληρωθεί. Βέβαια στις μέρες μας, αυτό το φαινόμενο είναι σπάνιο, καθώς οι συσκευές έχουν αρκετά μεγάλο μέγεθος μνήμης, συνήθως της τάξης 3GB και άνω.

2.1.3 Οι Δέκτες Εκπομπής (Broadcast Receivers)

Το Android σύστημα, είναι ικανό να εκπέμπει συμβάντα όπως ειδοποίηση κατάστασης χαμηλής μπαταρίας, εισερχομένη κλήση, εκκίνηση λειτουργικού συστήματος κτλ. Όταν λάβει χώρα κάποιο από αυτά τα συμβάντα, οι εφαρμογές μπορούν να το «ακούσουν» και να ανταποκριθούν σε αυτό πραγματοποιώντας μια ενέργεια. Το συστατικό το οποίο, πραγματοποιεί το παραπάνω, είναι οι Broadcast Receivers. Για παράδειγμα, μια εφαρμογή μουσικής θα μπορούσε να χρησιμοποιήσει ένα broadcast receiver, για να λάβει το συμβάν στο οποίο ο χρήστης αφαιρεί τα ακουστικά της συσκευής, και να το χρησιμοποιήσει για να σταματήσει την αναπαραγωγή μουσικής.

2.1.4 Οι Παροχείς Περιεχομένου (Content Providers)

Οι Content Providers είναι ένα συστατικό μιας εφαρμογής, που επιτρέπει την ανάκτηση και το μοίρασμα δεδομένων, τα οποία αντλεί από μια πηγή δεδομένων (συνήθως βάσης δεδομένων ή και το διαδικτύου) . Μια εφαρμογή μπορεί να ζητήσει δεδομένα από μια πηγή, τα οποία αποθηκεύονται σε ένα content provider, και να τα χρησιμοποιήσει. Μέσω των content providers, τα αποθηκευμένα δεδομένα μπορούν να διαμοιραστούν μεταξύ εφαρμογών. Για παράδειγμα κάποιες εφαρμογές θα μπορούσαν να αναζητήσουν τα στοιχεία των αποθηκευμένων επαφών σε μια συσκευή, προκειμένου να τα χρησιμοποιήσουν για διάφορες ενέργειες, όπως να χρησιμοποιηθεί η email διεύθυνση μιας επαφής για να σταλθεί ένα μήνυμα.

2.2 Επικοινωνία Μεταξύ Συστατικών

Όλα τα συστατικά που συνθέτουν μια εφαρμογή, αν και ανεξάρτητα, μπορούν να επικοινωνήσουν και να συνεργαστούν μεταξύ τους, έτσι ώστε να συμβάλουν στη λειτουργικότητα της. Το εργαλείο που πραγματοποιεί την επικοινωνία αυτή, είναι τα λεγόμενα Intents. Το Intent[19], είναι ένα ειδικό ασύγχρονο μήνυμα, το οποίο στέλνεται μεταξύ συστατικών, με σκοπό να τα διασυνδέσει μεταξύ τους. Για παράδειγμα, ας υποθέσουμε ότι έχουμε 2 activities σε μια εφαρμογή, τα A και B, και θέλουμε να μεταφερθούμε από το A στο B. Μέσω του A, μπορεί να σταλθεί ένα intent μήνυμα στο B, το οποίο θα του ζητά να ξεκινήσει, ενέργεια που συνήθως υλοποιείται μετά από κάποιο συμβάν, όπως για παράδειγμα το πάτημα ενός κουμπιού από το χρήστη. Ιδιαίτερο χαρακτηριστικό των intents είναι επίσης το γεγονός ότι μπορεί να μεταφερθεί πληροφορία μέσω αυτών. Ενισχύοντας το παραπάνω, θα μπορούσαμε για παράδειγμα να μεταφέρουμε το τίτλο ενός στοιχείου μια λίστας που θα βρισκόταν στο activity A μέσω του intent ώστε να ανακτήσει τη πληροφορία αυτή το activity B και να τη χρησιμοποιήσει για κάποιο σκοπό της.

Όσον αφορά τα activities και τα services, τα intents ορίζουν τη λειτουργία που θα πραγματοποιηθεί και επιπλέον μεταφέρεται τυχόν πληροφορία από το ένα συστατικό στο άλλο. Στη περίπτωση των broadcast receivers, αυτό που ορίζεται, είναι η ανακοίνωση που εκπέμπεται, όπως η ανακοίνωση χαμηλής μπαταρίας της συσκευής για παράδειγμα. Τέλος οι content providers, δεν χρησιμοποιούν intents, αλλά την κλάση ContentResolver[20].

2.3 Το Αρχείο Μανιφέστου (Manifest File)

Προκειμένου να ενεργοποιήσει οποιαδήποτε συστατικά μιας εφαρμογής, το Android διαβάζει ένα ειδικό αρχείο, το λεγόμενο manifest file, το οποίο περιγράφει όλα τα συστατικά που περιέχει μια εφαρμογή, καθώς και τις συνδέσεις μεταξύ αυτών. Οποιοδήποτε συστατικό που δεν δηλώνεται σε αυτό το αρχείο, δε μπορεί να βρεθεί από το σύστημα. Τα manifest files, πέρα από τα components, περιέχει και πληροφορίες όπως :

- Τα permissions της εφαρμογής (όπως επίτρεψη χρήσης της κάμερας, του GPS κτλ.)
- Το ελάχιστο επίπεδο του Android API που θα απαιτεί η εφαρμογή (ή αλλιώς η ελάχιστη υποστηριζόμενη Android έκδοση – πληροφορίες σε επόμενο κεφάλαιο)
- Υπηρεσίες συστήματος που θα χρησιμοποιεί η εφαρμογή (χρήση κάμερας, Bluetooth, GPS κτλ.)

2.3.1 Δήλωση Συστατικών

Το manifest file, έχει την εξής δομή στην οποία περιγράφονται όλες οι πληροφορίες μιας εφαρμογής:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
```

```
        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
        </activity>
        ...
    </application>
</manifest>
```

Το στοιχείο `<application>`, περιέχει όλες τις πληροφορίες που αφορούν την εφαρμογή. Η ιδιότητα `android:icon`, δηλώνει το εικονίδιο που θα χρησιμοποιεί η εφαρμογή. Το στοιχείο `<activity>`, περιέχει τις πληροφορίες που αφορούν ένα συγκεκριμένο activity. Στο παραπάνω παράδειγμα, η ιδιότητα `android:name`, δηλώνει το όνομα της Java κλάσης (σε μορφή πακέτου συνήθως) που χρησιμοποιεί το activity και η ιδιότητα `android:label`, ορίζει τη τιμή του κείμενου που θα βλέπει ο χρήστης ως τίτλο του activity. Υπάρχουν κι άλλα στοιχεία και ιδιότητες, που θα μελετηθούν στα επόμενα κεφάλαια και ενότητες όπου χρειάζονται.

2.4 Αρχεία Πόρων

Μια εφαρμογή, πέρα του κώδικα πηγής (source code) χρησιμοποιεί επιπλέον και αρχεία πόρων. Πρόκειται για αρχεία όπως εικόνες, βίντεο, ήχους, κείμενα κ.α. , τα οποία συμπεριλαμβάνονται στην εφαρμογή βάσει της διαμόρφωσης (configuration) της συσκευής. Για παράδειγμα εικόνες σε μια μικρού μεγέθους συσκευή δεν χρησιμοποιούνται σε μια μεγάλου μεγέθους. Μπορούμε δηλαδή να συμπεριλάβουμε για ένα resource, πολλαπλά αρχεία, τα οποία θα χρησιμοποιούνται κατάλληλα, ανάλογα με τη περίπτωση της συσκευής ή και άλλων όπως η γλώσσα συστήματος που χρησιμοποιείται. Θα δούμε λεπτομερώς το τρόπο με τον οποίο χρησιμοποιούνται χρησιμοποιούνται σε επόμενο κεφάλαιο.

3 Προετοιμασία Ανάπτυξης του Quick Speak

Στο παρόν κεφάλαιο, θα παρουσιαστούν τα εργαλεία και οι τεχνικές της μελέτης, σχεδίασης και ανάπτυξης του Quick Speak. Προ-απαιτούμενα για την ανάπτυξη, είναι ένα Windows, ή Linux σύστημα (υποστηρίζεται και το MacOS απλώς δεν εξετάζεται εδώ) και μια συσκευή Android για να τρέξει και να δοκιμαστεί η εφαρμογή.

3.1 Αντικειμενοστρεφής Προγραμματισμός

Η τεχνική του αντικειμενοστρεφούς προγραμματισμού[21] (OOP), παρουσιάστηκε για πρώτη φορά το 1960 η οποία μέχρι και σήμερα θεωρείται μια από τις σημαντικότερες μεθοδολογίες σχεδίασης κι ανάπτυξης λογισμικού. Η κύρια αρχή της, είναι οι έννοιες της κλάσης και του αντικειμένου. Ως αντικείμενο (οντότητα αλλιώς) , ορίζουμε μια δομή δεδομένων, η οποία έχει κάποια χαρακτηριστικά (ιδιότητες) και συμπεριφορές μέσα σε ένα πρόγραμμα. Ως κλάση, ορίζουμε ένα τύπο δεδομένων που περιγράφει τις ιδιότητες και τις συμπεριφορές αυτές.

Ένα αντικείμενο αποτελεί ένα στιγμιότυπο μιας κλάσης, που περιέχει τις ιδιότητες και συμπεριφορές που ορίζει αυτή. Μπορούμε να μιλάμε για ένα μέχρι και πολλά αντικείμενα μιας κλάσης, τα οποία είναι ανεξάρτητα το ένα από το άλλο. Ουσιαστικά μια κλάση αποτελεί ένα προσχέδιο των αντικειμένων που θα τη χρησιμοποιούν. Τα αντικείμενα με λίγα λόγια είναι αυτά που καταλαμβάνουν χώρο στη μνήμη (RAM) του συστήματος που τα χρησιμοποιεί. Οι ιδιότητες όπως θα δούμε και στη πράξη αργότερα, υλοποιούνται με μεταβλητές υπόστασης ορισμένες από ένα τύπο δεδομένων και μια αρχική τιμή. Οι συμπεριφορές, υλοποιούνται με τη χρήση μεθόδων που εκτελούν μια εργασία.

3.1.1 Ενθυλάκωση

Η Ενθυλάκωση (encapsulation), είναι μια έννοια κατά την οποία, οι κλάσεις μπορούν να υποκρύπτουν τα ιδιωτικά δεδομένα τους από το υπόλοιπο πρόγραμμα, εξασφαλίζοντας πρόσβαση αυτών μόνο μέσω των δημοσίων μεθόδων της. Η ενθυλάκωση, ουσιαστικά εξασφαλίζει ότι τίποτα δε θα διαχειριστεί τα ιδιωτικά δεδομένα με εσφαλμένο τρόπο που θα παρουσίαζε προβλήματα στο λογισμικό που υλοποιείται. Με λίγα λόγια, οι ιδιότητες και συμπεριφορές μια κλάσης υποκρύπτονται σε ένα αντικείμενο της, χωρίς να χρειάζεται κάποιος να γνωρίζει τις λεπτομέρειες της. Όταν λέμε δηλαδή ότι κάποιος χρησιμοποίει ένα αντικείμενο, στην ουσία δε κάνει τίποτα άλλο από το να καλέσει τις δημόσιες μεθόδους της κλάσης του για να βγάλει εις πέρας μια καθορισμένη εργασία, χωρίς να τον ενδιαφέρει το «πώς» την πραγματοποιεί το αντικείμενο (τι κώδικας εκτελείται στη μέθοδο, τι μεταβλητές χρησιμοποιούνται κτλ.) .

3.1.2 Κληρονομικότητα

Ένα αντικειμενοστρεφές πρόγραμμα, αποτελείται από ένα σύνολο κλάσεων, υλοποιημένες είτε από τις βιβλιοθήκες της γλώσσας που χρησιμοποιείται , είτε και από τον ίδιο το χρήστη που το αναπτύσσει. Κάθε κλάση του προγράμματος, χρησιμοποιείται από τα αντικείμενα της, με την έννοια ότι χωρίς το αντικείμενο, μια κλάση θεωρείται «αφηρημένη» (abstract). Οι κλάσεις μπορεί να είναι ανεξάρτητες μεταξύ τους ή να τοποθετηθούν σε μια ιεραρχία, και να παρουσιάζουν μια σημαντική ιδιότητα που ονομάζεται Κληρονομικότητα (Inheritance). Η κληρονομικότητα ορίζει πως μια κλάση, συνήθως πιο γενικού σκοπού και ενδεχομένως αφηρημένης (χωρίς να χρησιμοποιεί αντικείμενα) αποτελεί τη υπερ-κλάση (ή αλλιώς τη κλάση-πατέρα) στην ιεραρχία. Οι κλάσεις που βρίσκονται χαμηλότερα στην ιεραρχία αυτή, αποτελούν τις υπο-κλάσεις της (ή αλλιώς κλάσεις-παιδιά) και κληρονομούν τις ιδιότητες και συμπεριφορές που περιγράφει η κλάση-πατέρας, ορίζοντας ίσως και επιπλέον δικές τους. Η ιεραρχία της κληρονομικότητας, δεν έχει όρια βάθους, με την έννοια ότι μπορούμε να έχουμε παιδία που έχουν άλλα παιδία κ.ο.κ. (κλάσεις-εγγόνια της αρχικής κλάσης πατέρα) .

Για παράδειγμα, αν θέλαμε να περιγράψουμε ένα ζωολογικό κήπο με κλάσεις, τότε η κύρια κλάση πατέρας θα ήταν η κλάση Ζώο και ως παιδία θα μπορούσε να είχε κλάσεις όπως, τη κλάση Λιοντάρι και τη κλάση Αετός. Η κλάση Ζώο, περιγράφει τις ιδιότητες και συμπεριφορές όλων των ζώων γενικά. Οι υπο-κλάσεις της, κληρονομούν τις ιδιότητες και συμπεριφορές της και επιπλέον έχουν κάποιες ιδιότητες και συμπεριφορές που ορίζονται από την εκάστοτε κλάση-παιδί. Έτσι αν η κλάση Ζώο, έχει τις ιδιότητες ύψος, είδος, φύλο και συμπεριφορές να τραφεί και να κινηθεί, η κλάση Αετός θα μπορούσε μαζί με αυτά τα στοιχεία, να είχε επιπλέον την συμπεριφορά να πετάει. Αν εκείνη η κλάση με τη σειρά της , είχε κάποια άλλη κλάση-παιδί, η τελευταία , πάλι θα κληρονομούσε τα στοιχεία της κλάσης Αετός μαζί με όλα τα στοιχεία των υπερ-κλάσεων της (στη περίπτωση μας τη κλάση Ζώο), συν οποιαδήποτε άλλα στοιχεία ορίσει στον εαυτό της. Με λίγα λόγια η κληρονομικότητα, εξασφαλίζει ότι μια κλάση γενικού σκοπού υλοποιείται με τέτοιο τρόπο ώστε κάθε κλάση που κάνει πιο ειδικό το σκοπό αυτό, απαιτείται να υλοποιήσει μόνο τα στοιχεία του εαυτού της (δεν «ανακαλύπτουμε» το τροχό από την αρχή κάθε φόρα) και μάλιστα όσο πιο «βαθιά» πηγαίνουμε στην ιεραρχία των κλάσεων, τόσο πιο ειδικού σκοπού είναι οι κλάσεις των επιπέδων που βρίσκονται εκεί.

3.2 Η Γλώσσα Προγραμματισμού Java

Η γλώσσα που θα χρησιμοποιηθεί στη παρούσα πτυχιακή, είναι η Java. Πρόκειται για μια αντικειμενοστρεφή γενικού σκοπού γλώσσα, η οποία δημιουργήθηκε από τον James Gosling και κυκλοφορεί από το 1995 μέχρι και σήμερα. Η κύρια αρχή της είναι η «write once, run anywhere[12]", με την εξής έννοια. Όλα τα προγράμματα Java, γράφονται μια φόρα, μεταφράζονται σε κώδικα byte (bytecode) ο οποίος τρέχει πάνω σε μια εικονική μηχανή το λεγόμενο JVM (Java Virtual Machine), χωρίς να ενδιαφέρει το λειτουργικό σύστημα πάνω στο οποίο τρέχει η μηχανή αυτή. Οποιοδήποτε σύστημα και να χρησιμοποιείται, τα μόνα εργαλεία που χρειάζονται για να αναπτύξει και να τρέξει κανείς ένα Java πρόγραμμα, είναι το JDK (Java Development Kit) που περιέχει όλα τα εργαλεία ανάπτυξης κώδικα Java και μια JVM του εκάστοτε συστήματος πάνω στην οποία θα τρέχει το πρόγραμμα μας. Έτσι

εξασφαλίζεται ότι ένα πρόγραμμα μπορεί να γραφτεί και να τρέξει σε οποιοδήποτε σύστημα έχει εγκατεστημένη μια JVM, αντίθετα με γλώσσες που τρέχουν κώδικα απευθείας πάνω στο εκάστοτε σύστημα, απαιτώντας έτσι επιπλέον σύνταξη κώδικα (τα λεγόμενα ports[22][23]) , ώστε να υποστηριχθεί κάθε σύστημα χωριστά. Η παρούσα πτυχιακή, θα εστιάσει μόνο στα στοιχεία της γλώσσας που αφορούν την υλοποίηση του Quick Speak, χωρίς να εμβαθύνει σε πιο προχωρημένο επίπεδο πέραν αυτής.

3.3 Η Γλώσσα XML

Η XML[24][25] είναι μια ειδική γλώσσα σήμανσης που χρησιμοποιείται κυρίως στο διαδίκτυο. Πρόκειται για μια γλώσσα, η οποία οργανώνει τα αποθηκευμένα δεδομένα σε μια δομή παρόμοια με αυτήν της HTML[26][27], με τη διαφορά ότι ο χρήστης που συντάσσει ένα XML αρχείο, μπορεί να χρησιμοποιήσει δικά του ονόματα για τα tags της δομής καθώς ακόμα και να χρησιμοποιήσει τη δική του γλώσσα. Ένα XML αρχείο είναι μια μορφής όπως της παρακάτω:

```
<?xml version="1.0" encoding="UTF-8"?>
<breakfast_menu>
<food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>
   Two of our famous Belgian Waffles with plenty of real maple syrup
   </description>
    <calories>650</calories>
</food>
<food>
    <name>Strawberry Belgian Waffles</name>
    <price>$7.95</price>
    <description>
    Light Belgian waffles covered with strawberries and whipped cream
    </description>
    <calories>900</calories>
</food>
<food>
    <name>Berry-Berry Belgian Waffles</name>
    <price>$8.95</price>
    <description>
    Belgian waffles covered with assorted fresh berries and whipped cream
    </description>
    <calories>900</calories>
</food>
<food>
    <name>French Toast</name>
    <price>$4.50</price>
    <description>
    Thick slices made from our homemade sourdough bread
    </description>
    <calories>600</calories>
</food>
<food>
    <name>Homestyle Breakfast</name>
```

```
    <price>$6.95</price>
    <description>
    Two eggs, bacon or sausage, toast, and our ever-popular hash browns
    </description>
    <calories>950</calories>
</food>
</breakfast_menu>
```

Βλέποντας κανείς το παραπάνω μπορεί να καταλάβει ότι περιγράφεται ένα μενού πρωινών γευμάτων. Τα αρχεία XML συνήθως διαβάζονται από λογισμικό (το λεγόμενο parse[28]) ώστε να ανακτάται η πληροφορία που μας ενδιαφέρει. Ο τρόπος με τον οποίο βρίσκουμε μια πληροφορία σε ένα XML αρχείο, είναι ακολουθώντας ένα μονοπάτι μέσα στη δομή, το οποίο καταλήγει στη πληροφορία θέλουμε να ανακτηθεί. Αν για παράδειγμα, μας ενδιέφερε να ανακτήσουμε τη περιγραφή του φαγητού με όνομα `Homestyle Breakfast`, τότε θα πρέπει αρχικά να βρεθεί το tag `breakfast_menu`, και στη συνέχεια να βρεθεί η δομή `food`. Η δομή που μας ενδιαφέρει είναι αυτή που περιέχει το tag `name` με τιμή `Homestyle Breakfast`. Μέσα σε αυτή τη δομή, η πληροφορία που μας ενδιαφέρει, είναι η τιμή που βρίσκεται μέσα στο tag `description`. Το μονοπάτι για να βρεθούμε μέχρι εκεί είναι το εξής:

`breakfast_menu` ⟶ `food` ⟶ `description`.

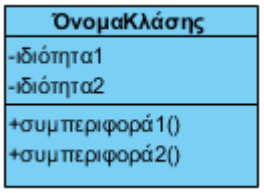
Αυτό το μονοπάτι θα ακολουθούσε κανείς για να ανακτήσει τη περιγραφή οποιουδήποτε φαγητού. Αυτός είναι και ο κύριος σκοπός της XML. Οργανώνονται τα δεδομένα σε συνεπείς δομές ώστε να είναι εύκολη η διαδικασία εντοπισμού και ανάκτησης δεδομένων. Έτσι με το ίδιο αρχείο μπορούν διάφοροι τύποι λογισμικού, ασχέτως της γλώσσας με την οποία τρέχουν, να αξιοποιήσουν τα δεδομένα που αυτό περιέχει. Η ανάγνωση (parse) και των XML αρχείων, γίνεται μέσω χρήσης ειδικών βιβλιοθηκών API (Application Program Interface) που είναι σχεδιασμένες για αυτό το σκοπό. Ένα τέτοιο API, διαβάζει τα XML αρχεία, αναζητά τα δεδομένα που ενδιαφέρουν προς ανάκτηση και το λογισμικό τα χρησιμοποιεί για διάφορους σκοπούς. Ήδη έχουμε δει ένα παράδειγμα χρήσης XML αρχείων στη περίπτωση του Android με το manifest file. Σε επόμενα κεφάλαια, θα δούμε το τρόπο με τον οποίο το Android, χρησιμοποιεί την XML για να περιγράψει και να σχεδιάσει συστατικά στην διεπαφή χρήστη ενός Android app.

3.4 Ολοκληρωμένο Περιβάλλον Ανάπτυξης

Τα Integrated Development Environments (IDE[29]), είναι περιβάλλοντα τα οποία περιέχουν εργαλεία και βιβλιοθήκες και χρησιμοποιούνται για ανάπτυξη λογισμικού. Σκοπός τους, είναι να διευκολύνουν τη διαδικασία συγγραφής και μεταγλώττισης πηγαίου κώδικα, περιέχοντας επεξεργαστές κειμένου με χαρακτηριστικά όπως αυτόματη συμπλήρωση μπλοκ κώδικα, εντοπισμού σφαλμάτων κτλ. Επίσης ένα IDE μπορεί να περιέχει εργαλεία σχεδίασης διεπαφών, καθώς και εργαλεία αποσφαλμάτωσης (debuggers). Το IDE που θα χρησιμοποιηθεί στη παρούσα πτυχιακή, είναι το Android Studio[30][31].

3.5 Το Διάγραμμα Κλάσεων

Ένα σημαντικό εργαλείο στην ανάπτυξη λογισμικού, είναι το διάγραμμα κλάσεων (class diagram). Σκοπός του είναι, να περιγράψει τη δομή του συστήματος, παρουσιάζοντας τις κλάσεις που χρησιμοποιούνται, τις ιδιότητες και τις συμπεριφορές τους καθώς και τις σχέσεις που υπάρχουν μεταξύ αυτών. Η γλώσσα που χρησιμοποιείται για τη σχεδίαση των διαγραμμάτων αυτών, είναι η Unified Modeling Language (UML[32][33]). Οι κλάσεις στη UML σχεδιάζονται σε μορφή ορθογωνίων σχημάτων όπως το παρακάτω:



Εικόνα 2: Απεικόνιση Κλάσης στη UML

Όπως φαίνεται και στο σχήμα, με έντονο στυλ γραμματοσειράς (bold) απεικονίζεται το όνομα της κλάσης που παρουσιάζεται, στο πρώτο πλαίσιο εμφανίζονται οι ιδιότητες της κλάσης και στο δεύτερο οι συμπεριφορές της. Θα δούμε λεπτομέρειες της UML όπως είναι η απεικόνιση σχέσεων μεταξύ κλάσεων, στο κύριο μέρος της πτυχιακής χρησιμοποιώντας το εργαλείο Visual Paradigm[34].

3.6 Το Android SDK

Το Android Software Development Kit (Android SDK[35][36]), είναι μια συλλογή εργαλείων και βιβλιοθηκών, η οποία χρησιμοποιείται για ανάπτυξη Android εφαρμογών. Κυκλοφόρησε το 2009 από τη Google, γραμμένο σε Java. Στα αρχικά του στάδια, το Android SDK αποτελούσε μέρος του Eclipse IDE[37][38] μέσω του ADT plugin[39] (Android Development Tools). Επίσης μπορούσε να χρησιμοποιηθεί στο NetBeans IDE[40][41] καθώς και στο IntelliJ DEA IDE[42][43]. Βασισμένο στο τελευταίο, η Google το 2015, ανάπτυξε το Android Studio, το οποίο, είναι πλέον το επίσημο IDE που περιέχει το Android SDK κι αυτό που θα χρησιμοποιηθεί στη παρούσα πτυχιακή για την ανάπτυξη του Quick Speak.

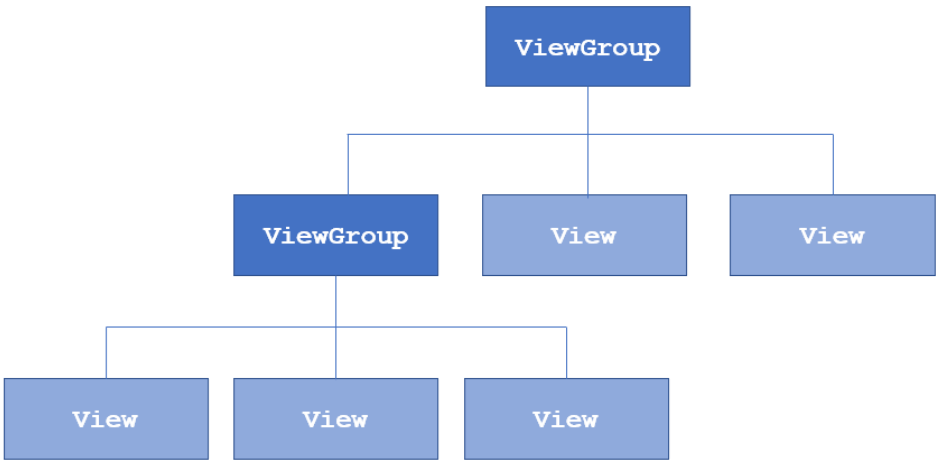
3.7 Activities και Views

Το activity[44], αποτελεί ένα στιγμιότυπο οθόνης που βλέπει και χρησιμοποιεί ο χρήστης, το οποίο παρουσιάζει μια διεπαφή χρήστη. Μια εφαρμογή μπορεί να περιέχει από ένα έως και πολλά activities, τα οποία συνήθως επικοινωνούν μεταξύ τους, μέσω των intents όπως προαναφέρθηκε. Ο χρήστης μπορεί να βλέπει ένα activity τη φορά, προτού μεταφερθεί με κάποιο συμβάν (πάτημα κουμπιού, πέραση κάποιου χρονικού ορίου – loading screen πχ κτλ.) σε ένα νεότερο ή και προηγούμενο του, activity. Τα activities όπως προαναφέρθηκε, αποτελούνται από μια διεπαφή χρήστη, η οποία περιέχει ένα σύνολο συστατικών που

ονομάζονται views. Τα views τοποθετούνται συνήθως βάσει κάποιας διάταξης (layout), η οποία ορίζει τον τρόπο με τον οποίο αυτά θα τοποθετούνται[45]. Ένα activity αποτελείται από 2 στοιχεία όπου το ένα ορίζει την εμφάνιση του και το άλλο τη λειτουργικότητα του. Τα views περιγράφονται με τη χρήση XML αρχείων, όπου περιγράφεται στην ουσία η δομή που θα ακολουθούν καθώς και οι ιδιότητες τους. Όσον αφορά το κομμάτι της λειτουργικότητας, ένα activity, είναι στην ουσία μια Java κλάση η οποία κληρονομεί τις ιδιότητες και συμπεριφορές από τη κλάση του Android API, με όνομα `Activity`[46]. Να διασαφηνιστεί ότι από εδώ και πέρα, όροι όπως `Activity`, `Intent`, `View` κτλ. , θα αφορούν τις κλάσεις του Android API κι όχι τις έννοιες τους, καθώς αυτές θα γράφονται με μικρά όπως activity για παράδειγμα.

3.7.1 Layouts

Μια διάταξη (layout), ορίζει τη δομή μιας διεπαφής, συνήθως αυτής ενός activity. Στην ουσία η δομή αυτή είναι ένα ιεραρχικό δέντρο που αποτελείται από αντικείμενα `ViewGroup` και `View`. Το view group είναι ένα «δοχείο» (container) στο όποιο τοποθετούνται διάφορα views. Για παράδειγμα, μια τέτοια δομή περιγράφεται στο παρακάτω σχήμα:



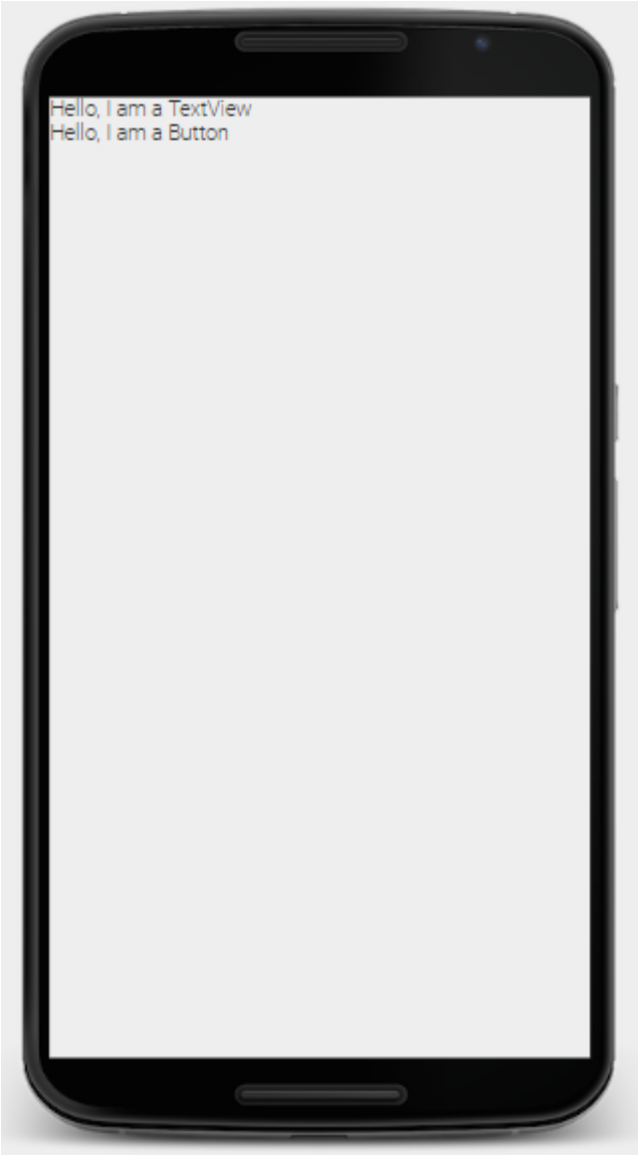
Εικόνα 3: Παράδειγμα δομής Layout

Ένα view group, μπορεί να περιέχει από ένα μέχρι και πολλά views αλλά ακόμα και άλλα εμφωλευμένα view groups. Βασικά παραδείγματα views είναι το `TextView`, το `Button` και το `ImageView` και βασικά παραδείγματα view groups είναι το `RelativeLayout`, το `LinearLayout` και ένα πρόσφατα προστεθειμένο στη βιβλιοθήκη Android, το `ConstraintLayout`. Για κάθε view ή group view, ορίζονται ιδιότητες όπως το μέγεθος τους, το τρόπο με το οποίο θα τοποθετηθεί στο layout κτλ.

Ένα layout, μπορεί να δημιουργηθεί μέσω ενός XML αρχείου που περιγράφει μια δομή όπως αυτή της Εικόνας 3. Για παράδειγμα, παρακάτω περιγράφεται η XML δομή μιας γραμμικής διάταξης (`LinearLayout`) με τη χρήση του Udacity Android Visualizer[47] , η οποία περιέχει ένα κείμενο και ένα κουμπί:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

Η διεπαφή που υλοποιεί η παραπάνω δομή είναι η εξής:



Εικόνα 4: Προβολή Βασικής Διεπαφής Ενός Activity

Ένα αρχείο διάταξης (layout file), αποτελείται από δύο στοιχεία. Το ένα είναι τα αντικείμενα που θα χρησιμοποιηθούν (views) στη διεπαφή και το άλλο είναι οι ιδιότητες (attributes) τους. Το υψηλότερο στην ιεραρχία component, αποτελεί το view group της διεπαφής, στο οποίο πρέπει να δηλωθεί η ιδιότητα:

```
xmlns:android=http://schemas.android.com/apk/res/android
```

Η ιδιότητα αυτή είναι το λεγόμενο Android namespace. Επιτρέπει σε όλα τα στοιχεία της δομής να χρησιμοποιούν τις ιδιότητες του Android μέσω της σύνταξης `android:`. Αυτές συνήθως ορίζουν ιδιότητες όπως, το μέγεθος των components, ένα μοναδικό αναγνωριστικό ID το οποίο συνήθως χρησιμοποιείται από το API για χρήση τους στο κώδικα του activity.

Ένα αρχείο όπως το παραπάνω, διαβάζεται από τη κλάση παιδί του activity, που περιγράψαμε προηγουμένως, με τη χρήση Java κώδικα. Για παράδειγμα αν ονομάζαμε το παραπάνω αρχείο `main_layout.xml`, για να διαβαστεί θα χρειάζονταν το παρακάτω μπλοκ κώδικα:

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_layout);
}
```

Στην ουσία η κλήση της μεθόδου `setContentView` είναι αυτή που διαβάζει το αρχείο και μετατρέπει τα views και view groups της XML δομής στα αντίστοιχα αντικείμενα Java του Android API (`TextView`, `Button` κτλ.). Το παραπάνω μπλοκ κώδικα είναι το κλασσικό κόμματι που υλοποιεί αρχικά κάθε ένα activity που δημιουργείται για τους σκοπούς μιας εφαρμογής.

Έστω ότι θέλαμε να αλλάξουμε τη τιμή του κείμενου της διεπαφής. Αυτό μπορεί να γίνει με 2 τρόπους. Ένας είναι να αλλάξουμε τη τιμή της ιδιότητας text του `TextView` από το XML αρχείο. Έτσι αν έχουμε την ιδιότητα ως εξής:

```
android:text="Hello TextView nice to meet you!"
```

Τότε η διεπαφή γίνεται ως εξής:

Εικόνα 5: Αλλαγή Κειμένου Διεπαφής

Το ίδιο μπορεί να γίνει και μέσω του κώδικα Java. Παρατηρώντας τη δομή, βλέπουμε ότι τα 2 components χρησιμοποιούν την ιδιότητα id με τιμές της μορφής `"@+id/name"`. Το ID είναι ένα μοναδικό αναγνωριστικό που μπορούμε να δώσουμε σε κάθε ένα από τα components ώστε να το χειριστούμε προγραμματιστικά στο κώδικα Java. Το παρακάτω μπλοκ κώδικα, ορίζει στο `TextView` της δομής, το κείμενο `"Hello TextView nice to meet you!"`, χρησιμοποιώντας το αντικείμενο `helloTextView` :

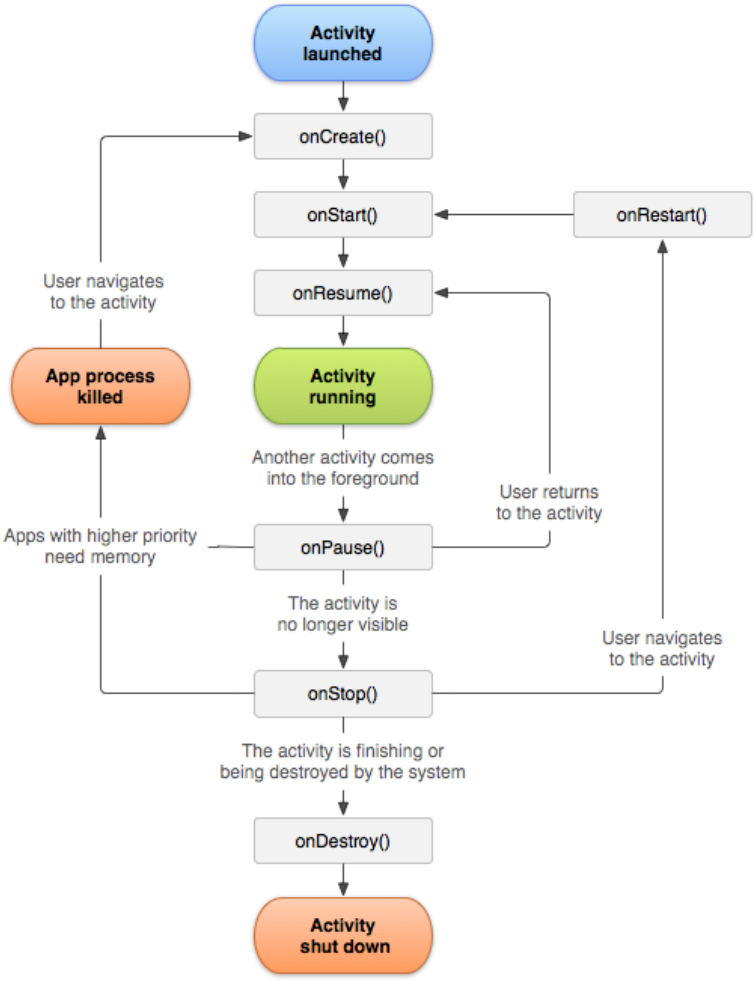
```
TextView helloTextView = (TextView) findViewById(R.id.text);
helloTextView.setText("Hello text nice to meet you!");
```

Η κλήση `findViewById` εντοπίζει το `TextView` με βάση το id με τιμή `text`, και στη συνέχεια αποθηκεύεται στο αντικείμενο `helloTextView`. Έτσι μέσω του αντικειμένου αυτού, μπορούμε να κάνουμε ενέργειες όπως τη χρήση της μεθόδου `setText` για να ορίσουμε το κείμενο του

`TextView`. Όπως βλέπουμε, ένα activity είναι στην ουσία μια διεπαφή-στιγμιότυπο μιας οθόνης, η οποία αποτελείται από ένα σύνολο συστατικών που τη περιβάλλουν και μιας κλάσης παιδί της κλάσης `Activity` του Android API, όπως θα δούμε και κατά την παρουσίαση του Quick Speak.

3.8 Ο Κύκλος Ζωής Ενός Activity

Όπως είναι γνωστό, ένα από τα πολλά χαρακτηριστικά του Android, είναι αυτό της πολυδιεργασίας (multitasking). Ο χρήστης μπορεί εύκολα να χρησιμοποιεί διάφορες εφαρμογές ταυτόχρονα και να μεταφέρεται από τη μια στην άλλη. Αυτό πραγματοποιείται με μια διαδικασία που ονομάζεται κύκλος ζωής ενός activity[48]. Κάθε activity, διατηρεί διάφορες καταστάσεις (states) για τις οποίες, το Android πραγματοποιεί κλήσεις συστήματος (callbacks) ώστε να εκτελέσει κώδικα ανάλογα τη κατάσταση στην οποία βρίσκεται. Με αυτό το τρόπο το σύστημα γνωρίζει τι ενέργειες θα πρέπει να εκτελεστούν καθώς ο χρήστης μεταφέρεται μεταξύ διαφόρων activities μιας εφαρμογής, καθώς επίσης και τι χρειάζεται να γίνει όταν μεταφερόμαστε σε διαφορετικές εφαρμογές. Ο κύκλος αυτός παρουσιάζεται στη παρακάτω εικόνα[48]:



Εικόνα 6: Απεικόνιση Κύκλου Ζωής Ενός Activity

Κάθε κλήση, αντιπροσωπεύει βήματα κατά τα οποία ένα activity μεταφέρεται από μια κατάσταση σε μια άλλη. Οι καταστάσεις ενός activity είναι οι εξής:

- **Εκκίνηση Activity**: Σε αυτή τη κατάσταση το activity ξεκινάει χωρίς να είναι ακόμα ορατό στον χρήστη.
- **Εκτέλεση Activity**: Σε αυτή τη κατάσταση το activity εκτελείται, όντας ορατό στο χρηστή και διαθέσιμο προς χρήση.
- **Κατάσταση «Σκοτώματος» Διεργασίας Activity**: Αν το σύστημα κρίνει ότι δεν υπάρχει αρκετή διαθέσιμη μνήμη και μια διεργασία με μεγαλύτερη προτεραιότητα τη χρειάζεται, τότε αυτό τερματίζει (σκοτώνει) εντελώς την τρέχουσα διεργασία που αφορά το activity. Επίσης το activity έρχεται σε αυτή τη κατάσταση όταν εκτελεστεί στο προσκήνιο ένα άλλο activity, ή αλλιώς όταν έχει την εστίαση (focus) του χρήστη ένα άλλο activity. Ο τερματισμός της τρέχουσας διεργασίας, δε σημαίνει απόλυτα τον τερματισμό του activity ή της εφαρμογής, πράγμα που σημαίνει ότι από αυτή τη κατάσταση, μπορούμε με τις κατάλληλες κλήσεις συστήματος, να βρεθούμε πάλι στη κατάσταση **Εκκίνηση Activity**. Με λίγα λόγια η κατάσταση αυτή, έχει την έννοια της προσωρινής παύσης του activity και της πιθανής επιστροφής του χρήστη σε αυτό.
- **Τερματισμός Activity:** Σε αυτή τη κατάσταση το activity τερματίζεται ολοκληρωτικά, απελευθερώνοντας ότι πόρους χρησιμοποιούσε. Από αυτό το σημείο δηλαδή και μέτα, το σύστημα δε γνωρίζει πλέον τίποτα για το activity, μέχρι ο χρήστης να το εκκινήσει ξανά. Ένα κλασικό παράδειγμα, είναι αυτό στο οποίο, ο χρήστης κάνει ολίσθηση με το δάκτυλο (swipe) σε μια εφαρμογή, από τη λίστα των πρόσφατων χρησιμοποιημένων εφαρμογών για να τη κλείσει ολοκληρωτικά.

Ας δούμε τις κλήσεις συστήματος του κύκλου ζωής και σε ποιες καταστάσεις, μπορούν αυτές να κληθούν.

3.8.1 Η Κλήση onCreate

Η onCreate είναι η αρχική κλήση ώστε να ξεκινήσει ένα activity. Κάθε ένα activity, πρέπει υποχρεωτικά να υλοποιεί αυτή τη μέθοδο. Σε αυτή τη μέθοδο, αρχικοποιούνται όλα τα στοιχεία εκείνα τα οποία θα χρειαστεί το activity, καθ' όλη τη διάρκεια ζωής του. Η πρώτη και κλασσική διαδικασία, είναι η κλήση της μεθόδου `setContentView`, η οποία διαβάζει ένα αρχείο διάταξης XML (layout file) και μετατρέπει τα components που περιγράφει, σε αντικείμενα Java. Επίσης εδώ, υλοποιείται η γενική λειτουργικότητα του activity, καθώς η onCreate αποτελεί ουσιαστικά το «πυρήνα» του. Η onCreate θυμίζει κάπως, την αντίστοιχη κλήση main της γλώσσας C ή Java, η οποία αποτελεί το σημείο από το οποίο ξεκινάει ένα πρόγραμμα την εκτέλεση του. Η βασική δομή onCreate αποτελείται από το εξής μπλοκ κώδικα:

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_activity);
}
```

Η πρώτη γραμμή του μπλοκ, είναι η απαραίτητη κλήση που γίνεται λόγο του γεγονότος ότι οι κλάσεις activity που υλοποιούνται, είναι ουσιαστικά παιδιά της κλάσης `Activity`.

3.8.2 Η Κλήση onStart

Η κλήση onStart είναι το σημείο στο οποίο η εφαρμογή γίνεται ορατή στο χρήστη. Ουσιαστικά με τη κλήση αυτή, γίνεται η προετοιμασία του activity, ώστε να μπει στο προσκήνιο και γίνει λειτουργικό. Πρόκειται για μια σύντομη κλήση στην οποία συνήθως η εφαρμογή αρχικοποιεί κώδικα που διατηρεί το γραφικό περιβάλλον (UI).

3.8.3 Η Κλήση onResume

Η κλήση onResume πραγματοποιείται μέτα την onStart, φέρνοντας πλέον το activity στο προσκήνιο. Σε αυτό το σημείο δηλαδή ο χρήστης μπορεί να το δει και να το χρησιμοποιήσει. Το activity βρίσκεται στη κατάσταση αυτή, μέχρι κάποιο event να συμβεί έτσι ώστε να χαθεί η εστίαση σε αυτό. Για παράδειγμα ένα τέτοιο event θα μπορούσε να είναι η ειδοποίηση εισερχόμενης κλήσης, ή το γεγονός ότι κάποια άλλη εφαρμογή ήρθε στο προσκήνιο. Σε αυτή τη περίπτωση καλείται η μέθοδος onPause κι εφόσον αυτή ολοκληρώσει και το σταματημένο activity αποκτήσει την εστίαση του χρήστη και πάλι, καλείται η onResume ώστε να το φέρει και πάλι μπροστά στο προσκήνιο. Στην ουσία μιλάμε για μια προσωρινή συνήθως, παύση του activity και επιστροφή σε αυτό όταν το χρειαστούμε ξανά.

3.8.4 Η Κλήση onPause

Η κλήση onPause, έχει την έννοια της πρώτης προειδοποίησης ότι ο χρήστης πιθανόν να εγκαταλείψει το activity. Η παύση λειτουργίας του activity που συμβαίνει με αυτή τη κλήση, δε σημαίνει απόλυτα τη «καταστροφή» του από το σύστημα. Μπορεί όπως είδαμε και πριν, να σημαίνει ότι ο χρήστης ίσως επιστρέψει όταν εστιάσει ξανά σε αυτό. Στη περίπτωση που δε συμβεί αυτό, τότε το σύστημα θα καλέσει την onStop όπως θα δούμε παρακάτω. Η onPause, είναι συνήθως μια κλήση, στην οποία χρειάζεται να σταματήσουν προσωρινά ή να ρυθμιστούν κάποιες λειτουργίες του activity για όλη τη διάρκεια που αυτό θα είναι σε κατάσταση παύσεως, και θα επαναφερθούν όταν αυτό θα συνεχίσει να λειτουργεί ξανά.

3.8.5 Η Κλήση onStop

Η εφαρμογή με τη κλήση αυτή δεν είναι πλέον ορατή στο χρήστη. Αυτό μπορεί για παράδειγμα να συμβεί, αν κάποιο άλλο activity ενεργοποιηθεί δεσμεύοντας ολόκληρη την οθόνη (δεν είναι ορατό με κάποιο τρόπο δηλαδή το σταματημένο activity) ή αν το activity στο όποιο βρισκόμασταν, έχει ολοκληρώσει τη δουλεία του και είναι έτοιμο να τερματιστεί. Σε αυτή τη κλήση, συνηθώς υλοποιείται η διακοπή των λειτουργιών του activity, όσο αυτό βρίσκεται σε αυτή τη κατάσταση. Η διαφορά της κλήσης αυτής με την onPause έχει να κάνει με το γεγονός ότι εδώ το activity δεν είναι καθόλου ορατό στο χρήστη, έναντι της onPause στην οποία είναι μερικώς ορατό. Από αυτή τη κατάσταση το activity μπορεί να επιστρέψει σε λειτουργία με 2 τρόπους. Ο ένας είναι να επιστρέψει ο χρήστης σε αυτό, κι έτσι το σύστημα θα

καλέσει την onRestart ώστε ξεκινήσει να το φέρει και πάλι στο προσκήνιο. Ο άλλος τρόπος, έχει να κάνει με την μνήμη του συστήματος που προαναφέρθηκε, δηλαδή με το γεγονός ότι το σύστημα έκρινε ότι χρειάζεται επιπλέον διαθέσιμη μνήμη για μια εφαρμογή και χρειάζεται να τερματίσει το τρέχον activity. Έτσι καλεί την onStop για να διακόψει το activity (μερικές φορές αυτό μπορεί να προκαλέσει το λεγόμενο "crash" της εφαρμογής) και στη συνέχεια όταν το χρησιμοποιήσει ο χρήστης ξανά, το σύστημα καλεί την onCreate για το να δημιουργήσει από την αρχή.

3.8.6 Η Κλήση onRestart

Όπως προαναφέρθηκε, η κλήση onRestart, καλείται μετά τον onStop, εφόσον ο χρήστης επιτρέψει στο σταματημένο και μη ορατό activity, έτσι ώστε να μπορεί να το χρησιμοποιεί και πάλι. Μετά την ολοκλήρωση της onRestart, όπως φαίνεται και στο διάγραμμα του κύκλου ζωής, καλείται η onStart και ύστερα η onResume, και έτσι πλέον το activity είναι πλέον ορατό και πάλι στον χρήστη.

3.8.7 Η Κλήση OnDestroy

Αν και το όνομα της ακούγεται κάπως ανησυχητικό, η onDestroy καλείται όταν το activity είναι έτοιμο να «καταστραφεί» και να εξαφανιστεί από το σύστημα, τερματίζοντας οριστικά την λειτουργία του. Αυτό μπορεί να συμβεί, είτε επειδή έχει τελειώσει τη δουλεία και άρα δεν είναι πλέον αναγκαίο να τρέχει, καθώς δεσμεύει μνήμη στο σύστημα, είτε επειδή το σύστημα σταματάει προσωρινά τη λειτουργία του για να το δημιουργήσει από την αρχή. Το τελευταίο συμβαίνει όταν το σύστημα εντοπίσει μια αλλαγή στη διαμόρφωσή του (configuration change[49]). Η πιο συνηθισμένη αλλαγή, είναι συνήθως αυτή του προσανατολισμού της οθόνης από προσανατολισμό πορτραίτου (portrait mode) σε τοπίου (landscape) και αντίστροφα. Συνήθως στην onDestroy απελευθερώνονται πόροι πριν το activity καταστραφεί οριστικά. Οτιδήποτε χρειαστεί τερματισμό και δε τερματιστεί στη κλήση onStop, είναι απαραίτητο να γίνει εδώ, διαφορετικά μέτα τη καταστροφή του activity, δεν είναι εφικτό αυτό.

Ο κύκλος ζωής με λίγα λόγια, ορίζει και ρυθμίζει όλες τις συμπεριφορές των activities μιας εφαρμογής ανάλογα με τη κατάσταση στην οποία βρίσκονται κάθε φορά. Αυτό μειώνει αποτελεσματικά προβλήματα όπως απρόβλεπτη συμπεριφορά μιας εφαρμογής η απρόβλεπτη διακοπή της (crash) κάτω από μια συγκεκριμένη κατάσταση. Επίσης με τη σωστή υλοποίηση των κλήσεων αυτών αποφεύγονται τέτοιου είδους προβλήματα, αφού ελέγχεται ο τρόπος με τον οποίο θα συμπεριφέρεται αυτή σε κάθε κατάσταση. Η μόνη κλήση που χρειάζεται απαραίτητα υλοποίηση σε μια activity κλάση, είναι αυτή της onCreate. Οι υπόλοιπες είναι προαιρετικές και υλοποιούνται ανάλογα τις ανάγκες των activities μιας εφαρμογής.

3.9 Η Σημασία των Δεδομένων και της Λίστας

Τα δεδομένα είναι το κυριότερο μέρος μιας εφαρμογής. Ο χρήστης δίνει κάποια δεδομένα σε ένα λογισμικό, αυτό τα επεξεργάζεται, δίνοντας του πίσω τα αποτελέσματα αυτής της επεξεργασίας. Πέραν αυτού ο χρήσης πολλές φορές μπορεί και να ζητήσει μια πληροφορία από ένα λογισμικό κι αυτή η πληροφορία δεν είναι τίποτα άλλο παρά μιας αναζήτησης δεδομένων των οποίων η εύρεση τους θα καταλήξει σε αυτό που ενδιαφέρει το χρήστη. Φυσικά και μια εφαρμογή σε φορητές συσκευές δεν αποτελεί εξαίρεση όλων των παραπάνω. Η απεικόνιση κειμένων, εικόνων, ακόμα και πολυμέσων, αποτελεί χρήση δεδομένων. Σε ένα απλό σενάριο μιας εφαρμογής, μας αρκεί να απεικονίσουμε κάποιο η κάποια από αυτά δεδομένα μια φορά. Αυτά είναι τα views στη περίπτωση του Android όπως είχε προαναφερθεί. Ένα view, έχει την έννοια του «δοχείου» (placeholder) μια μορφής δεδομένου. Πολλές φορές όμως η απλή προσθήκη μεμονωμένων views σε ένα layout δεν αρκεί για να παρουσιάσει μια πληθώρα δεδομένων ειδικά όταν αυτά χρειάζεται και να επαναλβάνονται. Η προσθήκη ενός ίδιου view 2, 3 και παραπάνω φορές σε μια δομή XML, με διαφορετικά δεδομένα το καθένα, αποτελεί μια ιδιαίτερα προβληματική τεχνική ως προς τη συντήρησή τους. Αν θέλαμε για παράδειγμα να προσθέσουμε σε μια διάταξη 100 ονόματα ενός καταλόγου πελατών και γνωρίζοντας ότι τα ονόματα αυτά θα παρουσιαστούν μια ένα text view, η προσθήκη 100 τέτοιων views παρουσιάζει προβλήματα όπως εντοπισμών τυχών λαθών στα δεδομένα, συντήρησή του κάθε view χωριστά (αλλαγή ιδιοτήτων συνήθως) κτλ. Τι μπορεί να κάνει κανείς για να επιλύσει ένα τέτοιου είδους πρόβλημα;

Το Android περιέχει ένα ειδικό view, που ονομάζεται `ListView`[50] το οποίο έχει τη δυνατότητα να προβάλει μια λίστα με views. Έτσι, συνεχίζοντας με το προηγούμενο παράδειγμα, τα ονόματα θα παρουσιάζονταν με τη προσθήκη ενός list view και μόνο. Κάθε στοιχείο της λίστας ενός list view, αποτελεί μια διάταξη, η οποία περιέχει μέσα τα views που θα επαναλαμβάνονται. Αν για παράδειγμα θέλαμε επίσης να διατηρούμε και τα τηλέφωνα των πελατών, πάλι γνωρίζοντας ότι αυτά θα παρουσιάζονται μέσω ενός text view, τότε το list view θα αποτελούσε μια λίστα της οποίας κάθε στοιχείο θα ήταν μια διάταξη η οποία με τη σειρά της θα αποτελούνταν από 2 text views, ένα για το όνομα πελάτη και ένα για το τηλέφωνο. Έχοντας ορίζει έτσι το τρόπο με τον οποίο θα παρουσιάζονται πολλά δεδομένα με τη χρήση λίστας, μένει το εξής ερώτημα. Πώς μπορεί κάποιος να συνδέσει τα δεδομένα με τα views της λίστας αυτής;

3.9.1 Ο Προσαρμογέας (Adapter)

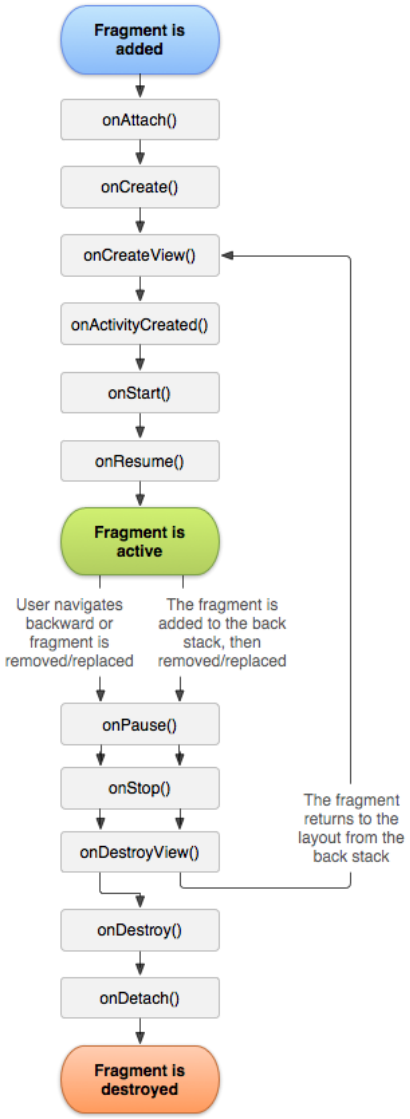
Τα δεδομένα τοποθετούνται στα views, με τη χρήση ενός ειδικού εργαλείου που ονομάζεται `Adapter`[51]. Πρόκειται για ένα είδος «γέφυρας» μεταξύ μια πηγής δεδομένων (Java κλάσης που περιγράφει τη δομή τους ή βάσης δεδομένων) και των views. Στην ουσία ένας adapter, παίρνει τα δεδομένα , και αναλαμβάνει να τα συσχετίσει με τα αντίστοιχα views. Οι adapters, είναι Java κλάσεις που υλοποιούν τη λογική με την οποία γίνεται η συσχέτιση αυτή. Δύο από τους σημαντικότερους είναι ο `ArrayAdapter`[52] κι ο `CursorAdapter`[53]. Στη παρούσα πτυχιακή θα δούμε τη σπουδαιότητα των adapters καθώς και το τρόπο με τον οποίο θα παρουσιάζουμε εύκολα μια λίστα δεδομένων χωρίς την χρήση πολλαπλών views.

4.0 Το Θραύσμα (Fragment)

Το Fragment[54], αντιπροσωπεύει ένα τμήμα μιας διεπαφής ενός activity. Το fragment είναι κάτι σαν ένα υπό-activity, το οποίο έχει το δικό του κύκλο ζωής, τη δική του διεπαφή χρήστη και διαχειρίζεται τα δικά του συμβάντα. Το fragment είναι εξαρτώμενο από το activity που το φιλοξενεί, με την έννοια ότι αν το τελευταίο σταματήσει, σταματάνε και τα fragments που φιλοξενούνται και επίσης αν το activity καταστραφεί εντελώς από το σύστημα, τότε αυτό συνεπάγει και την καταστροφή των fragments του. Κάθε fragment, είναι ανεξάρτητο από τα άλλα. Αυτό σημαίνει ότι μπορεί κάποιος να προσθέτει ή να αφαιρεί ένα fragment από ένα activity, χωρίς να επηρεάζει το ίδιο. Αυτή η ενέργεια ονομάζεται fragment transaction και γίνεται μέσω μια στοίβας (back stack) την οποία διαχειρίζεται το activity. Αυτό επιτρέπει την αναίρεση μιας συναλλαγής fragment που έγινε, επιστρέφοντας στη προηγούμενη κατάσταση του activity (ή ενός προηγούμενου fragment).

Τα fragments υλοποιούνται με μια Java κλάση, η οποία κληρονομεί τα στοιχεία της κλάσης `Fragment`[55]. Ένα fragment, έχει όπως προαναφέρθηκε, τη δική του διεπαφή, η οποία φυσικά περιγράφεται με το γνωστό XML αρχείο, όπως γίνεται και με τα activities. Σε αυτό το σημείο να τονίσουμε ότι αν και έχει ομοιότητες με αυτό, ένα fragment δεν είναι το ίδιο πράγμα με ένα activity και οι αντίστοιχες κλάσεις τους, δεν ανήκουν στην ίδια ιεραρχία κλάσεων. Πρόκειται λοιπόν για δυο ανεξάρτητες κλάσεις μεταξύ τους.

Το fragment όπως και το activity, έχει το δικό του κύκλο ζωής, ο οποίος παρουσιάζει ομοιότητες με αυτόν ενός activity, συν κάποιων δικών του κλήσεων. Ο κύκλος ζωής των fragments παρουσιάζεται στο παρακάτω σχήμα:



Εικόνα 7: Ο Κύκλος Ζωής Ενός Fragment

Η κλήση που υλοποιείται συνήθως είναι η `onCreateView`. Το σύστημα πραγματοποιεί αυτή τη κλήση, όταν σχεδιάζει τη διεπαφή του fragment. Η `onCreateView`, επιστρέφει ένα αντικείμενο `View`, το οποίο αντιπροσωπεύει όπως θα δούμε και στην υλοποίηση της εφαρμογής της πτυχιακής, το view που βρίσκεται στη κορυφή της δομής XML που χρησιμοποιεί το fragment. Παρακάτω παρουσιάζεται ένα ενδεικτικό παράδειγμα σε κώδικα, της υλοποίησης ενός βασικού fragment:

```
public static class ExampleFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.example_fragment, container,
false);
    }
}
```

Στο παράδειγμα αυτό, η δομή που θα χρησιμοποιήσει το fragment, περιγράφεται από το XML αρχείο `example_fragment.xml`. Μέσω του αντικειμένου `inflater`, καλείται η μέθοδος `inflate`, η οποία διαβάζει το αρχείο και επιστρέφει το View αντικείμενο που αφορά το component πατέρα της δομής. Το όρισμα container αφορά το component πατέρα της διάταξης του activity μέσα στο οποίο θα τοποθετηθεί το fragment. Η έννοια του inflate, ουσιαστικά είναι η διαδικασία κατά την οποία ένα fragment, ορίζει τη διάταξη που θα χρησιμοποιήσει και αρχικοποιεί τα αντικείμενα `View` του. Η διαφορά με τα activities, είναι ότι τα αντικείμενα δεν μπορούν να χρησιμοποιηθούν απευθείας (με τη κλήση `findViewById` που είδαμε στα activities) αλλά μέσω ενός ειδικού αντικειμένου όπως θα δούμε και στην υλοποίηση του Quick Speak.

Τα fragments έχουν τη λογική επαναχρησιμοποίησης, με την έννοια ότι δε πρέπει να εξαρτώνται ιδιαίτερα από ένα συγκεκριμένο activity, άλλα να μπορούν να τοποθετηθούν σε διάφορα. Αυτή η ικανότητα τους δίνει ένα σημαντικό προνόμιο στη σχεδίαση και υλοποίηση μιας εφαρμογής ώστε να υποστηρίζει και συσκευές μεγαλύτερης ανάλυσης όπως είναι τα tablets. Για παράδειγμα, αν θέλαμε μια εφαρμογή η οποία περιέχει 2 λίστες την Α και Β και από την Α πάμε στη Β, σε μια συσκευή tablet υπαρχεί χώρος για να εμφανιστουν και οι 2. Η σχεδίαση τους μέσω ενός activity και μόνο θα παρουσίαζε προβλήματα σε μια μικρότερη συσκευή καθώς δε θα υπήρχε χώρος για να εμφανιστούν οι λίστες. Η λύση σε αυτό το πρόβλημα είναι τα fragments. Στο παραπάνω σενάριο, αρκεί να υλοποιηθούν 2 fragments, όπου το καθένα θα υλοποιεί τη κάθε λίστα χωριστά. Έτσι στη περίπτωση του tablet, τα fragments θα τοποθετηθούν σε ένα activity εφόσον υπάρχει χώρος στην οθόνη για να εμφανιστούν οι 2 λίστες. Στη περίπτωση των μικρότερων συσκευών, η λογική είναι ότι υλοποιούμε 2 activities, τα οποία θα φιλοξενούν χωριστά τις 2 λίστες. Έτσι από το activity πχ Α στο οποίο τοποθετούμε το fragment με τη λίστα Α, μεταβαίνουμε στο activity Β, στο οποίο φυσικά έχει προστεθεί το fragment με τη λίστα Β. Και στις 2 περιπτώσεις τα fragments δεν έχουν υποστεί καμία αλλαγή. Αυτός είναι και ο βασικός σκοπός χρήσης τους.

4 Υλοποίηση του Quick Speak

Στο παρόν κεφάλαιο, θα ασχοληθούμε με το κύριο μέρος της πτυχιακής που αφορά, τη μελέτη, τη σχεδίαση και την ανάπτυξη της εφαρμογής του Quick Speak. Μέσω της ανάπτυξης της, θα δούμε το τρόπο με τον οποίο συνδέονται μεταξύ τους τα εργαλεία και οι τεχνικές που παρουσιάστηκαν στα προηγούμενα κεφάλαια.

4.1 Το πρόβλημα των Γλωσσών

Η εκμάθηση ξένων γλωσσών, είναι μια διαδικασία, η οποία απαιτεί χρόνο και αρκετή εξάσκηση, τόσο στη γραφή όσο και στην ομιλία. Η γνώση μιας ξένης γλώσσας πέρα της μητρικής, αποτελεί ένα σημαντικό εργαλείο, όχι μόνο στη καθημερινότητα αλλά και στον επαγγελματικό τομέα. Όπως υπάρχουν άτομα σήμερα με γνώση σε διάφορες γλώσσες, έτσι υπάρχουν και άτομα τα οποία δεν γνωρίζουν κάποια ξένη γλώσσα. Μια γλώσσα για να μαθευτεί σε επαρκές επίπεδο γνώσης, απαιτεί ίσως και μερικών χρόνων εκμάθησης, μια διαδικασία που δεν είναι πάντα εφικτή πόσο μάλλον αν σκεφτεί κανείς και το κόστος των μαθημάτων. Η εξέλιξη της τεχνολογίας έχει βοηθήσει σημαντικά στην εκμάθηση οποιασδήποτε γνώσης, με τη χρήση εφαρμογών σε φορητές συσκευές. Πλέον είναι σχετικά εύκολο κάποιος να παρακολουθήσει online μαθήματα πάνω σε ένα τομέα χωρίς την απαραίτητη παρουσία σε κάποια αίθουσα ή μέσω κάποιου αυστηρού προγράμματος. Ο χρήστης έχει στα χέρια του, την ικανότητα να κανονίσει πότε και με τι τρόπο θα παρακολουθεί τέτοιου είδους μαθήματα. Μαθήματα όπως αυτά, βρίσκουν εφαρμογή και στην εκμάθηση γλωσσών. Παρόλα αυτά, το πρόβλημα με το χρόνο διάθεσης εκμάθησης, παραμένει. Πώς μπορεί να μάθει ένας χρήστης τα βασικά στοιχεία μια γλώσσας, χωρίς να διαθέσει πάρα πολύ χρόνο πόσο μάλλον και χρόνια.

Εδώ έρχεται η ιδέα του Quick Speak, η εκμάθηση δηλαδή ξένων γλωσσών, με γρήγορο τρόπο μέσω μιας Android εφαρμογής (η οποία θα μπορούσε βέβαια να ήταν και μια εφαρμογή για οποιοδήποτε άλλο σύστημα). Η λογική του Quick Speak βασίζεται στο γεγονός ότι ο ανθρώπινος εγκέφαλος, έχει την ικανότητα να συγκρατεί πληροφορία σε σχηματισμούς (patterns), οι οποίοι όταν επαναλαμβάνονται καταλήγουν σε ρουτίνες, τις οποίες θυμάται. Σίγουρα κάνεις δε μπορεί να μάθει μια γλώσσα έχοντας μόνο το στοιχείο αυτό στα χέρια του. Παρόλα αυτά, το Quick Speak αποτελεί ένα εργαλείο με το οποίο ο χρήσης θα μπορέσει να μάθει τα βασικά μιας ξένης γλώσσας, με τέτοιο τρόπο ώστε να μπορέσει να εμβαθύνει τις γνώσεις του περαιτέρω αν το θελήσει.

Το Quick Speak, όπως προαναφέρθηκε, είναι ένα λεξικό, το οποίο αποτελείται από μια συλλογή λημμάτων λέξεων και φράσεων, συχνά καθομιλουμένων, τις οποίες μπορεί να χρησιμοποιήσει ο χρήστης για τη βασική επικοινωνία μιας γλώσσας, χωρίς να χρειάζεται απαραίτητα η γνώση γραμματικών. Η γλώσσα με την οποία θα ξεκινήσουμε είναι τα Αγγλικά, παρόλα αυτά θα δούμε με ποιο τρόπο, η εφαρμογή μπορεί να επεκταθεί, για την εκμάθηση επιπλέον γλωσσών.

4.2 Απαιτήσεις Συστήματος

Το Quick Speak, αφορά χρήστες Android συσκευών, με υποστήριξη και παλαιών εκδόσεων του λειτουργικού. Είναι κυρίως σχεδιασμένη για κινητά κι όχι τόσο για συσκευές tablet παρόλα αυτά είναι λειτουργική και σε αυτά. Να τονιστεί ότι το Quick Speak δε πρόκειται για μια εμπορική εφαρμογή, οπότε δεν θα είναι διαθέσιμη σε κάποιο online store εφαρμογών, όπως το Play Store, ούτε και απαιτεί μια σύνδεση Internet για να λειτουργήσει. Πρόκειται για μια εφαρμογή ακαδημαϊκού σκοπού, στα πλαίσια της παρούσας πτυχιακής και μόνο. Θα είναι διαθέσιμη στο συνοδευτικό υλικό της παρούσας πτυχιακής. Όσον αφορά την ανάπτυξη της, τα εργαλεία που θα χρησιμοποιηθούν είναι, μια εγκατεστημένη έκδοση Java, καθώς και μια εγκατεστημένη έκδοση του Android Studio, μέσω του οποίου θα εγκατασταθεί το Android SDK και όποιες επιπλέον βιβλιοθήκες χρειαστούν.

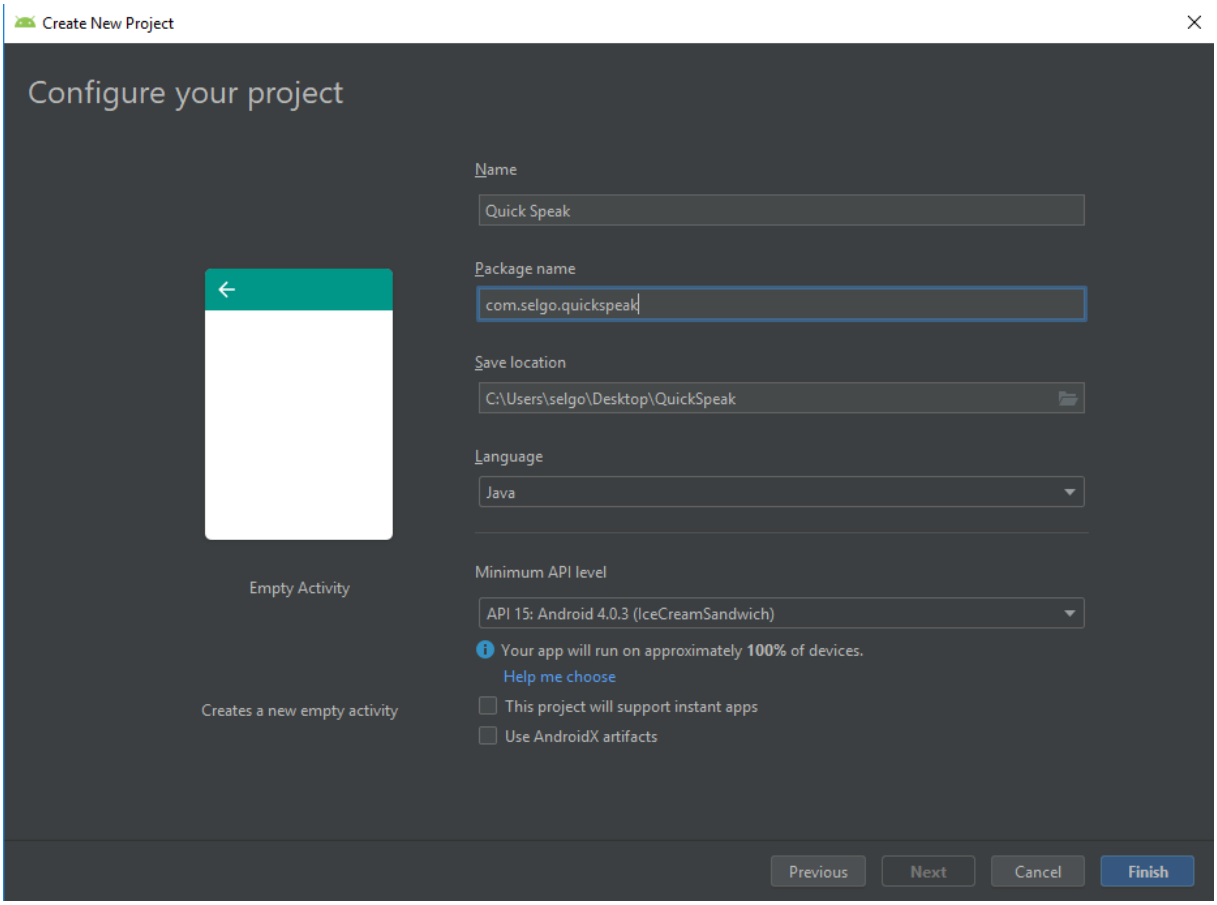
4.3 Το Προσχέδιο του Quick Speak

Το Quick Speak αποτελείται από 2 activities και ένα fragment. Αρχικά θα παρουσιάζεται η σχεδίαση καθενός κομματιού της εφαρμογής και στη συνέχεια θα παρουσιάζεται βήμα βήμα η υλοποίηση της λειτουργικότητας του.

Αρχικά θα πάμε να σχεδιάσουμε το κυρίως activity της εφαρμογής. Το activity αυτό, έχει την έννοια της αρχικής οθόνης καλωσορίσματος που βρίσκονται στις περισσότερες εφαρμογές. Στη συνέχεια αφού υλοποιήσουμε και τη λογική του, θα προχωρήσουμε στο δεύτερο activity σχεδιάζοντας το και υλοποιώντας το. Έπειτα θα αναπτύξουμε μια βάση δεδομένων στην οποία θα αποθηκευτούν τα λήμματα που θα χρησιμοποιηθούν για το λεξικό. Αφού ολοκληρωθεί αυτό, θα περάσουμε στην ανάπτυξη ενός fragment που θα παρουσιάζει τη λίστα με τα λήμματα ανά κατηγορίες και θα αναπτύξουμε ένα side menu, με τη χρήση του navigation drawer όπως θα δούμε, το οποίο θα παρουσιάζει τη λίστα των επιπέδων. Με λίγα λόγια κάθε σύνολο λημμάτων είναι διαχωρισμένο σε επίπεδα γνώσης τα οποία θα εμφανίζονται σε αυτό το side menu. Τέλος με τη χρήση ενός λογισμικού τρίτου, το TTSAutomate[56] θα δημιουργήσουμε τα ηχητικά κομμάτια των λημμάτων και θα υλοποιηθεί η ικανότητα, με την οποία ο χρήστης θα πατάει ένα κουμπί στο κάθε λήμμα ώστε να ακούει τη προφορά του.

4.4 Προετοιμασία του Project στο Android Studio

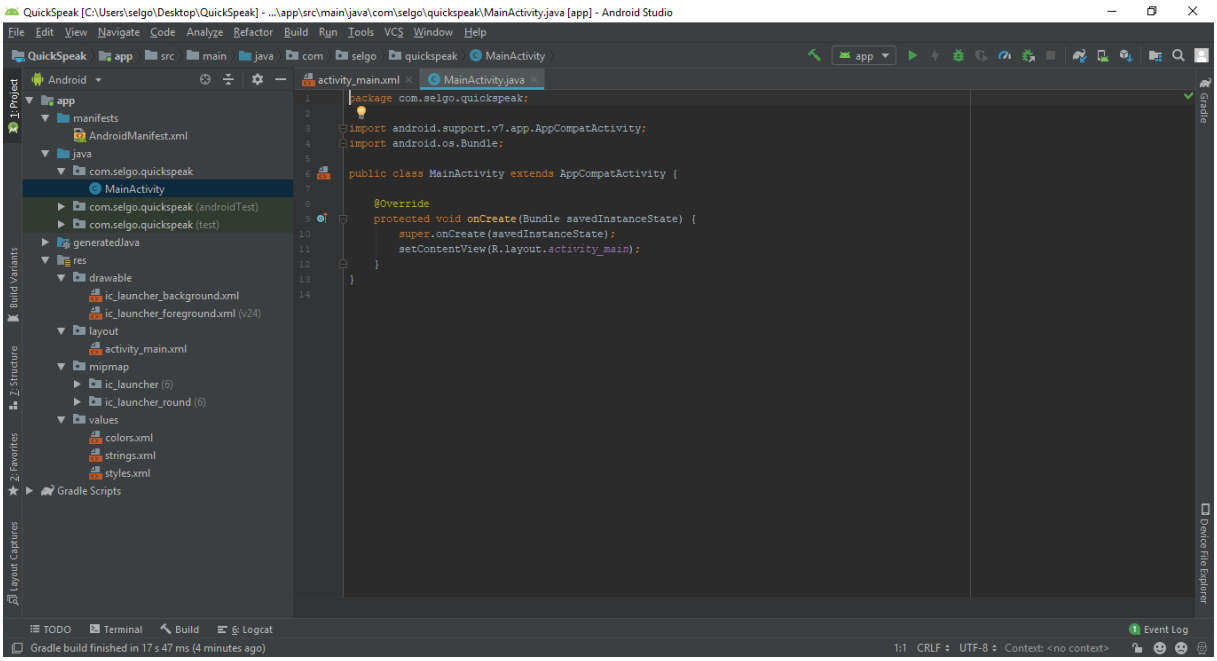
Ανοίγοντας το Android Studio, θα δημιουργήσουμε το project του Quick Speak. Οι ρυθμίσεις του project, παρουσιάζονται στη παρακάτω εικόνα:



Εικόνα 8: Ρύθμιση και Δημιουργία του Quick Speak Project

Σχόλιο σχετικά με το παραπάνω στιγμιότυπο. Το όνομα πακέτου, είναι υποκειμενική επιλογή. Σκοπός του είναι να αποτρέψει ονομαστικές συγκρούσεις με άλλες εφαρμογές αν για παράδειγμα είχαμε συμπωματικά, να κάνουμε με μια άλλη ήδη υπάρχων εφαρμογή με όνομα Quick Speak.. Για κάθε έκδοση του λειτουργικού συστήματος του Android, υπάρχει ένα αντίστοιχο επίπεδο[57] (level) του Android API που την υποστηρίζει για ανάπτυξη εφαρμογών σε αυτήν, αριθμημένο από το 1 και μετά. Το Minimum API Level, αφορά την ελάχιστη έκδοση του Android API από την οποία θα υποστηρίζεται το Quick Speak. Τη στιγμή που γράφεται η παρούσα πτυχιακή, η ελάχιστη συνηθισμένη έκδοση Android που χρησιμοποιείται, είναι η IceCreamSandwich 4.0.3 με API level το 15.

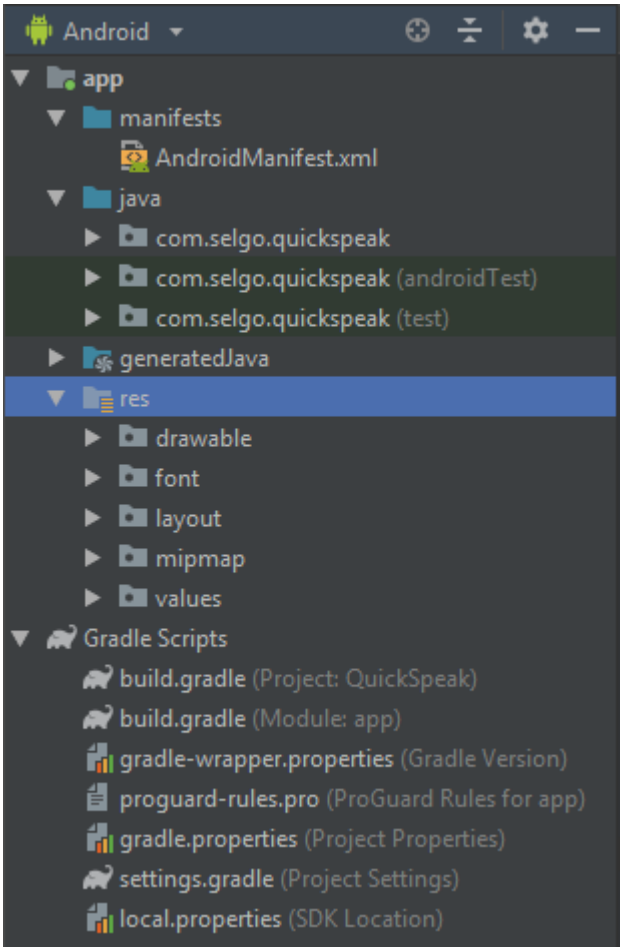
Πατώντας Finish και περιμένοντας να ολοκληρωθεί η διαδικασία δημιουργίας του Quick Speak project θα μεταφερθούμε στο εξής παράθυρο:



Εικόνα 9: Το Περιβάλλον του Android Studio

Το Android Studio αυτομάτως ανοίγει το αρχείο XML της διάταξης που περιγράφει τη διεπαφή του αρχικού activity (MainActivity) της εφαρμογής, καθώς και το αρχείο Java που υλοποιεί τη λογική του. Και στα δύο αυτά αρχεία, το περιβάλλον παρέχει αρχικό κώδικα από τον οποίο μπορούμε να αρχίσουμε να υλοποιούμε κομμάτια της εφαρμογής χωρίς να χρειάζεται να ξεκινήσουμε εξολοκλήρου από το μηδέν. Αν πάμε να τρέξουμε την εφαρμογή συνήθως αυτό που υλοποίει αρχικά το Android Studio, είναι μια εφαρμογή ενός activity που εμφανίζει ένα κείμενο όπως «Hello World». Παρόλα αυτά, θα μεταποιηθούν τα 2 αρχεία με τέτοιο τρόπο, ώστε να υλοποιηθεί ένα στιγμιότυπο οθόνης που θα έχει το ρόλο μιας οθόνης καλωσορίσματος.

Κάθε project στο Android Studio, ακολουθεί μια δομή με την οποία εντοπίζονται τα αρχεία που χρειάζεται να αναπτυχθούν, διαχειρίζονται οι πόροι (κείμενα, εικόνες, βίντεο κτλ.) και ορίζονται γενικά όλες οι ρυθμίσεις του project μιας εφαρμογής. Κάθε φάκελος του project συνήθως έχει ένα συγκεκριμένο ρόλο σε τέτοιο βαθμό, ώστε η ονοματολογία τους και μόνο, μπορεί παίζει σημαντικό ρόλο στο τρόπο με τον οποίο το Android διαχειρίζεται αυτούς και την εφαρμογή. Παρακάτω, παρουσιάζεται ένα ενδεικτικό παράδειγμα της δομής του project που δημιουργήθηκε. Κάποιοι φάκελοι και αρχεία δημιουργούνται από το ίδιο το Android Studio, κι άλλοι δημιουργούνται από το χρήστη που αναπτύσσει την εφαρμογή.



Εικόνα 10: Η Δομή Ενός Android Studio Project

Στη παρούσα πτυχιακή καθώς και σε κάθε Android project γενικά, μας απασχολούν κυρίως 3 από τους φακέλους. Οι φάκελοι αυτοί, είναι:

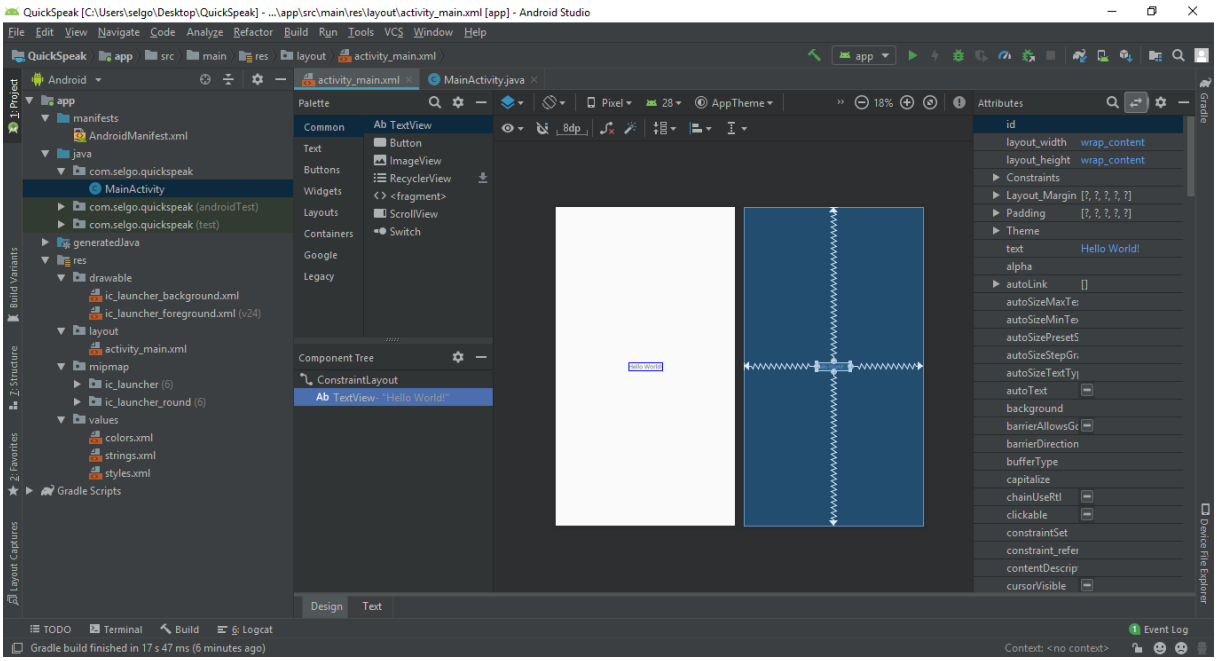
- **manifests :** Εδώ τοποθετείται το αρχείο manifest του project.
- **java :** Εδώ συμπεριλαμβάνεται το πακέτο που δημιουργήθηκε το οποίο περιέχει τα αρχεία Java της εφαρμογής. Συνήθως όπως είδαμε, το Android Studio δημιουργεί το αρχικό activity όπως ρυθμίστηκε παραπάνω. Από εκεί και πέρα ο χρήστης μπορεί να χρησιμοποιήσει επιπλέον activities των οποίων το Java αρχείο τοποθετείται σε αυτόν το φάκελο. Οι υπόλοιποι 2 φάκελοι, αφορούν τα tests της εφαρμογής με τη χρήση του JUnit[58][59]. Στη παρούσα πτυχιακή, δε θα ασχοληθούμε με αυτά τα tests.
- **res :** Εδώ τοποθετούνται τα αρχεία πόρων (resource files) του project που θα χρησιμοποιεί η εφαρμογή. Θα δούμε κατά την υλοποίηση των κομματιών του Quick Speak, το ρόλο κάθε υποφακέλου, έχοντας υπόψιν για την ώρα ότι όλα τα resources χωρίζονται σε φακέλους αναλόγως με το είδος και τον ρόλο που θα έχουν στην εφαρμογή που τα χειρίζεται.
- **Gradle Scripts (φάκελος gradle) :** Σε αυτό το φάκελο, περιέχονται όλα τα αρχεία τα οποία αφορούν το σύστημα «χτισίματος» (building system) Gradle[60][61]. Το Gradle χρησιμοποιείται για να μεταγλωττίσει τους κώδικες του project και να συνδυάσει όλα

τα απαραίτητα αρχεία δημιουργώντας έτσι το τελικό εκτελέσιμο αρχείο (στη περίπτωση μας το αρχείο apk). Τα αρχεία αυτά κρατάνε πληροφορίες όπως εκδόσεις των βιβλιοθηκών που χρησιμοποιούνται και τυχόν ρυθμίσεις του project που ορίζει ο χρήστης. Το μόνο αρχείο το οποίο πιθανώς να μας απασχολήσει είναι το build.gradle (Module: app). Σε αυτό αποθηκεύονται όλες οι ρυθμίσεις που έγιναν κατά την δημιουργία του project, συν τον πιθανώς επιπλέον που χρειάζονται. Για παράδειγμα το αρχείο σε αυτό το σημείο, έχει το εξής περιεχόμενο:

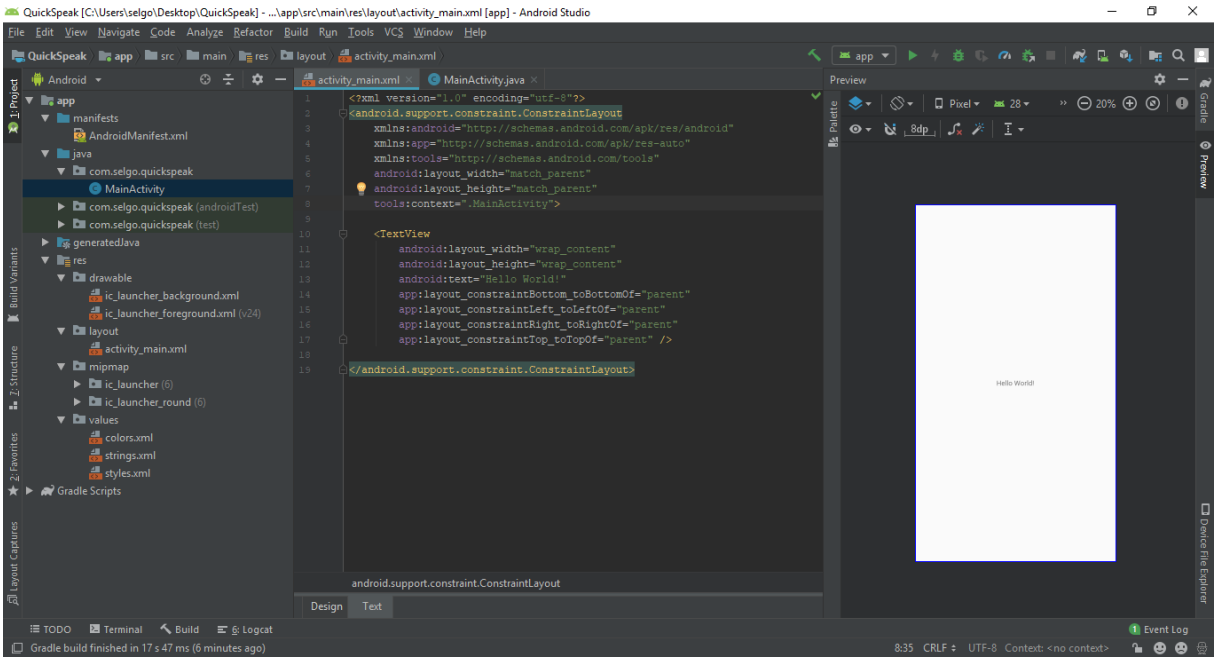
```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "com.selgo.quickspeak"
        minSdkVersion 15
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUni-
tRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-opti-
mize.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-
core:3.0.2'
}
```

Τα αρχεία XML, αφορούν το γραφικό περιβάλλον των activities και υλοποιούνται με δύο τρόπους. Ο ένας είναι μέσω του design editor, ο οποίος παρουσιάζει το activity σαν κάποιας μορφής καμβά, παρέχοντας τα αντικείμενα που μπορούν να τοποθετηθούν με τη μέθοδο drag n drop στο activity και να οριστούν οι ιδιότητες αυτών. Ο άλλος, και ουσιαστικός τρόπος με τον οποίο δουλεύουμε, είναι η απευθείας σύνταξη του XML αρχείου. Στην ουσία ότι κάνουμε στον design editor, αλλάζει και το XML αρχείο αντίστοιχα. Η απευθείας σύνταξη των αρχείων απαιτεί γνώση των ιδιοτήτων και της σύνταξης των αντικειμένων σε αυτό, παρόλα αυτά είναι ο πιο απλός και άμεσος τρόπος να εργαστεί κανείς με το αρχείο αυτό. Φυσικά σε κάθε περίπτωση η online τεκμηρίωση του Android[62], παρουσιάζει όλα τα αντικείμενα και τις ιδιότητες τους. Δεν υπάρχει βέλτιστος ή χείριστος τρόπος μεταξύ αυτών των δύο τρόπων υλοποίησης. Είναι περισσότερο υποκειμενική προτίμηση. Αυτό που ισχύει γενικά είναι το γεγονός ότι αλλαγές που γίνονται είτε στο design editor είτε στο αρχείο XML απευθείας, προσαρμόζονται με τέτοιο τρόπο και στις δύο περιπτώσεις έτσι ώστε να παρουσιάζουν την ίδια

διεπαφή στο τέλος. Στη παρούσα πτυχιακή θα ασχοληθούμε περισσότερο με την απευθείας σύνταξη των XML αρχείων του project, αναφέροντας και την άλλη περίπτωση όπου χρειάζεται παρουσιάζοντας επίσης τις λεπτομερείς αυτών.



Εικόνα 11: Ο Design Editor του Android Studio



Εικόνα 12: Ο Text Editor του XML αρχείου της Διεπαφής ενός Activity

Η υλοποίηση των Java αρχείων καθώς και των υπολοίπων αρχείων προφανώς και γίνεται απλά ανοίγοντας το αρχείο που μας ενδιαφέρει στον editor του Android Studio. Κανένα

επιπλέον σχόλιο δεν είναι απαραίτητο εδώ. Έχοντας υπόψιν τα παραπάνω ας προχωρήσουμε στην ανάπτυξη της εφαρμογής.

4.5 Το Main Activity

Το αρχικό activity έχει όπως είπαμε, το ρόλο ενός welcome screen. Τα περιεχόμενα αυτού του activity, είναι το όνομα της εφαρμογής, ένα εικονίδιο που αντιπροσωπεύει την εφαρμογή και μια κυκλική μπάρα προόδου, που θα κινείται ώσπου να μεταφερθούμε στο κύριο μέρος της εφαρμογής.

4.5.1 Το Αρχείο activity_main

Το αρχείο της διεπαφής του `MainActivity`, παρουσιάζει 3 views,τοποθετώντας τα σε ένα view group. Τυπικά, group views όπως το `RelativeLayout`[63][64] ή το `LinearLayout`[65][66], συνδυάζονται ώστε να δημιουργήσουν μια απλή ή και πολύπλοκη διεπαφή. Συνήθως ξεκινάμε από ένα view group που αποτελεί τη ρίζα της δομής (root view) της διεπαφής στο οποίο τοποθετούμε άλλα views ή και view groups όπως είδαμε και στο κεφάλαιο 2. Το πρόβλημα όμως με αυτή τη τεχνική, είναι το γεγονός ότι τα πολλαπλά και εμφωλευμένα group views, είναι ευκόλως ανοιχτά σε σχεδιαστικά λάθη τα οποία δε μπορούν να εντοπιστούν εύκολα. Χρειάζεται με λίγα λόγια ιδιαίτερη προσοχή στο να σχεδιαστούν με τέτοιο τρόπο ώστε να μπορούν να μεταποιηθούν ή να ανταποκριθούν σε τυχόν αλλαγές της διεπαφής ενός activity.

Το πρόβλημα αυτό έρχεται να επιλύσει ένα νεότερο είδος view group, το `ConstraintLayout`[67][68]. Το view group αυτό, δίνει τη δυνατότητα να τοποθετεί components βάσει κάποιων περιορισμών (constraints) που ορίζονται μεταξύ τους. Οι περιορισμοί αυτοί, καθιστούν μη αναγκαία την ύπαρξη εμφωλευμένων view groups. Για κάθε component, μπορούμε να ορίσουμε ένα constraint σε 4 διαφορετικά σημεία, πάνω, κάτω, δεξιά και αριστερά από αυτό. Τα constraints θέτονται είτε με την οθόνη ως σημείο αναφοράς, είτε με κάποιο άλλο component. Ένα constraint, είναι στην ουσία ένα βέλος το οποίο ξεκινά από μία πλευρά ενός component και καταλήγει στο σημείο αναφοράς που μας ενδιαφέρει. Η απόσταση κάθε τέτοιου βέλους έχει συγκεκριμένη τιμή δημιουργώντας έτσι μια διεπαφή με συμμετρικότητα.

Το Android γενικά για αποστάσεις, χρησιμοποιεί μια μονάδα μέτρησης, το density independent pixel[69] (dp και sp για μεγέθη κειμένων). Αυτή η μονάδα μας επιτρέπει να ορίζουμε μια τιμή σε κάθε απόσταση η μέγεθος, διατηρώντας τη κατάσταση αυτών των αλλαγών σε κάθε μέγεθος οθόνης, χωρίς δηλαδή να μας ενδιαφέρει το πλήθος των pixels κάθε οθόνης. Έτσι με μια τιμή το αποτέλεσμα οπτικά σε κάθε οθόνη είναι το ίδιο.

Ας πάμε να δούμε βήμα βήμα, το τρόπο με τον οποίο, θα χρησιμοποιήσουμε το αρχείο `activity_main.xml` για να σχεδιάσουμε τη διεπαφή του `MainActivity`. Αρχικά θα τοποθετήσουμε το view group της δομής:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

</android.support.constraint.ConstraintLayout>
```

Το ConstraintLayout, αποτελεί μέρος της βιβλιοθήκης υποστήριξης του Android (android support library) και μπορεί να χρησιμοποιηθεί από το API 9 (Gingerbread) και έπειτα. Αν και στις νεότερες εκδόσεις το Android Studio το πραγματοποιεί αυτόματα, για να δουλέψουμε με το ConstraintLayout απαιτείται να τοποθετηθεί η παρακάτω γραμμή στο dependencies μπλοκ του build.gradle (Module: app) αρχείου:

```
implementation 'com.android.support.constraint:constraint-layout:1.1.3'
```

Υπόψιν ότι η έκδοση 1.1.3 είναι αυτή που χρησιμοποιήθηκε στη παρούσα πτυχιακή και ίσως μελλοντικά να αλλάξει σε μεταγενέστερη.

Αν πάμε να θέσουμε constraint σε ένα component με σημείο αναφοράς την οθόνη του activity, τότε αυτά το τοποθετούν με τέτοιο τρόπο ώστε να έρθει ακριβώς στη μέση αυτής. Έτσι αυτή η διάταξη εκμεταλλεύεται σε πολύ καλό βαθμό το design editor του Android Studio. Παρόλα αυτά εδώ θα παρουσιαστεί η XML δομή που δημιουργείται κατά την επεξεργασία της διεπαφής με τον design editor. Τοποθετώντας τα 3 views που αναφέρθηκαν η τελική μορφή της διεπαφής στο XML είναι:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:contentDescription="welcome_icon"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.353"
        app:srcCompat="@mipmap/english_flag" />

    <TextView
        android:id="@+id/textView"
```

```
        style="@style/Theme.AppCompat.Light"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="4dp"
        android:layout_marginEnd="8dp"
        android:fontFamily="@font/open_sans_light"
        android:text="@string/app_name"
        android:textSize="25sp"
        android:textStyle="bold"
        app:fontFamily="@font/open_sans_light"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView" />

    <ProgressBar
        android:id="@+id/progress_bar"
        style="?android:attr/progressBarStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:indeterminate="true"
        android:max="100"
        android:progress="1"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        app:layout_constraintVertical_bias="0.064" />

</android.support.constraint.ConstraintLayout>
```

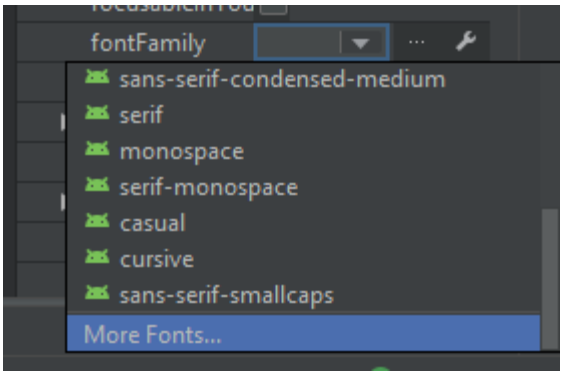
Ας πάμε να δούμε το ρόλο των ιδιοτήτων κάθε component. Το `android:id`, χρησιμοποιείται για να εντοπίσουμε μέσω αυτού τα components ώστε να τα χρησιμοποιήσουμε στο κώδικα Java, όπως θα δούμε αργότερα. Το id αποτελεί γενικά καλή τεχνική να υπάρχει ακόμα κι όταν δεν χρησιμοποιείται το component από τη Java, καθώς αν χρειαστεί να γίνει κάποια μελλοντική αναφορά σε αυτό, μπορεί να γίνει εύκολα με τη χρήση του id του. Η τιμή `@+id`, ορίζει ότι στο σημείο αυτό έχουμε δημιουργήσει για πρώτη φορά το id του αντικειμένου. Σε διαφορετική περίπτωση αναφερόμαστε σε αυτό με τη χρήση του ονόματος του (εξαίρεση οι ιδιότητες constraints) . Για παράδειγμα αν θέλουμε να αναφερθούμε στο `TextView` της παραπάνω δομής θα χρησιμοποιήσουμε το id του, `textView`.

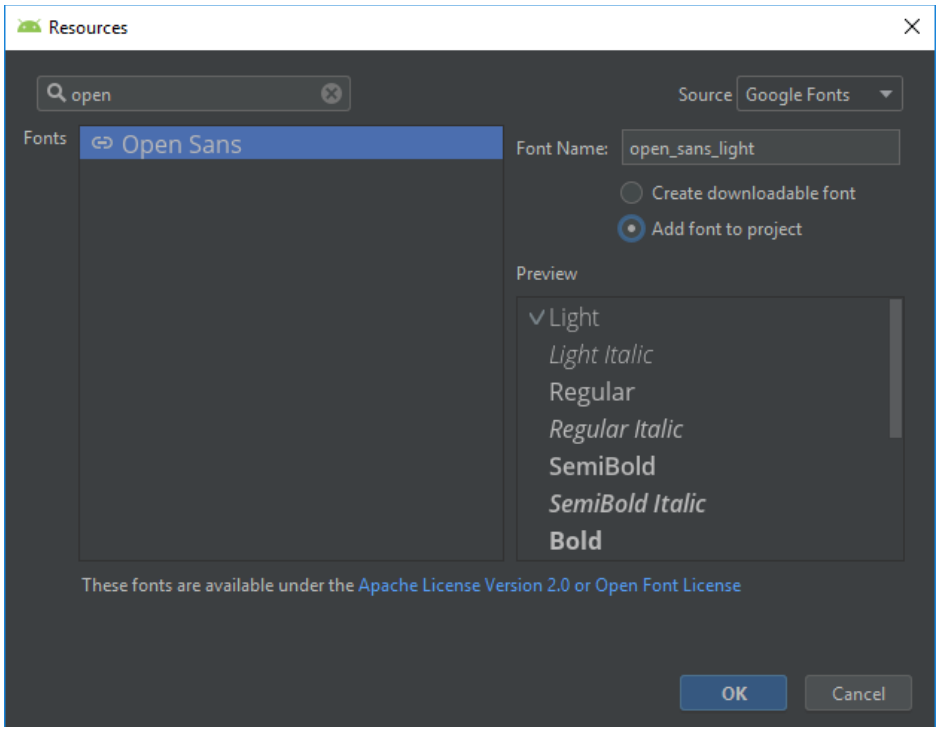
Κάθε component και view group, χρειάζεται να ορίσει ένα μέγεθος πλάτους και ύψους. Αυτά ορίζονται με τη χρήση των ιδιοτήτων `android:layout_width` και `android:layout_height`. Για το root view της δομής, οι τιμές αυτές ορίζονται με βάση το μέγεθος της οθόνης της συσκευής ενώ στη περίπτωση των components, οι τιμές ορίζονται με βάση το μέγεθος του root view στο οποίο βρίσκονται. Έτσι συνήθως στο root view ορίζουμε και στις δύο ιδιότητες, τη τιμή `match_parent` η οποία στην ουσία μεγαλώνει το στοιχείο με τέτοιο τρόπο ώστε να έχει το ίδιο μέγεθος με τον πατέρα στον οποίο βρίσκεται που στην ουσία εδώ είναι το μέγεθος της οθόνης. Στα components, η ίδια τιμή θα τα μεγάλωνε τόσο όσο είναι το μέγεθος του root view, δηλαδή το `ConstraintLayout`. Έτσι αντί αυτού, στα components

χρησιμοποιείται συνήθως η τιμή `wrap_content`, η οποία στην ουσία μεγαλώνει το συστατικό μέχρι να έχει μέγεθος ίσο με το συνολικό μέγεθος των περιεχομένων του.

Οι ιδιότητες margins, ορίζουν τα κενά γύρω από το component. Αυτές οι ιδιότητες καθώς και οι ιδιότητες των constraints ορίστηκαν αυτόματα στο XML κατά τη σχεδίαση με τη χρήση του design editor. Γενικά στο constraint layout η λογική είναι ότι τα κενά αυτά ορίζονται με όμοιες τιμές και συνήθως της τάξης των 8dp.

Για τη συγκεκριμένη διεπαφή θα χρειαστούν κάποια επιπλέον resources. Μερικά από αυτά τα resources βέβαια αφορούν και ολόκληρη την εφαρμογή. Θα δούμε το ρόλο καθενός φακέλου που βρίσκεται στο φάκελο res. Παρακάτω παρουσιάζονται δύο εικόνες όπου επιδεικνύεται ο τρόπος με τον οποίο μπορούμε να προσθέσουμε μια νέα γραμματοσειρά για χρήση της στην εφαρμογή. Συγκεκριμένα εδώ θα χρησιμοποιηθεί η Open Sans Light





Εικόνες 13 και 14: Τοποθέτηση Νέας Γραμματοσειράς στο Android Studio

Αφού ολοκληρωθεί η διαδικασία αυτή το αρχείο `open_sans_light.ttf`, βρίσκεται πλέον στο φάκελο font κάτω από φάκελο res πάντα. Ο φάκελος values, αφορά όλα τα resources, που έχουν να κάνουν με τα χρώματα, τα κείμενα και τα οπτικά στυλ που θα χρησιμοποιηθούν στην εφαρμογή. Τροποποιώντας το styles.xml και παρουσιάζοντας τα αρχεία colors.xml και strings.xml αντίστοιχα, έχουμε:

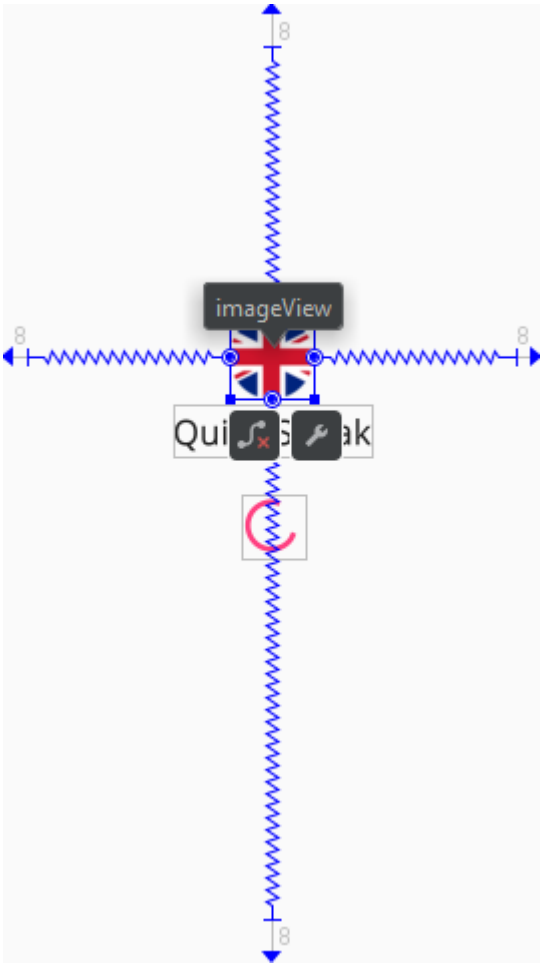
```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <!-- Customize your theme here. -->
        <item name="fontFamily">@font/open_sans_light</item>
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

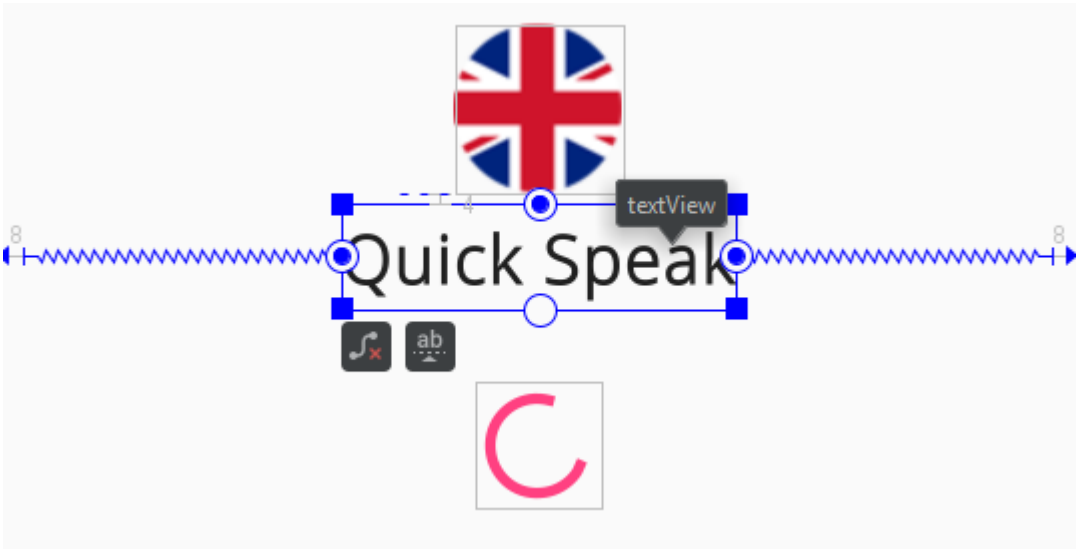
```
<resources>
    <string name="app_name">Quick Speak</string>
</resources>
```

Χρησιμοποιώντας το όνομα κάθε tag του κάθε αρχείου, μπορούμε να αναφερθούμε σε αυτό και να το χρησιμοποιήσουμε σε διάφορα κομμάτια της εφαρμογής που υλοποιούνται. Για παράδειγμα όπως βλέπουμε παραπάνω, το αρχείο styles.xml χρησιμοποιεί χρώματα τα οποία αντλεί από το colors.xml αρχείο. Με την ίδια λογική μπορούμε να χρησιμοποιήσουμε τα resources αυτά, και στα υπόλοιπα κομμάτια της εφαρμογής. Στο φάκελο layout, αποθηκεύονται τα αρχεία XML που περιγράφουν τη δομή μιας διεπαφής. Τέλος στο φάκελο mipmap και drawable, αποθηκεύονται τα αρχεία εικόνων. Το drawable πρόκειται για φάκελο που δημιούργει αρχικά το Android Studio για να τοποθετήσει το προκαθορισμένο εικονίδιο που θα εμφανίζεται στο εικονίδιο της εφαρμογής στη συσκευή. Ο mipmap αφορά εικονίδια που προστίθενται κατά τη διάρκεια της ανάπτυξης του Quick Speak. Εδώ θα προστεθεί το **`english_flag.png`**. Όπως βλέπουμε και στο XML, η χρήση κάθε resource γίνεται με τη χρήση τιμών όπως `@φάκελος/όνομα_resource`.

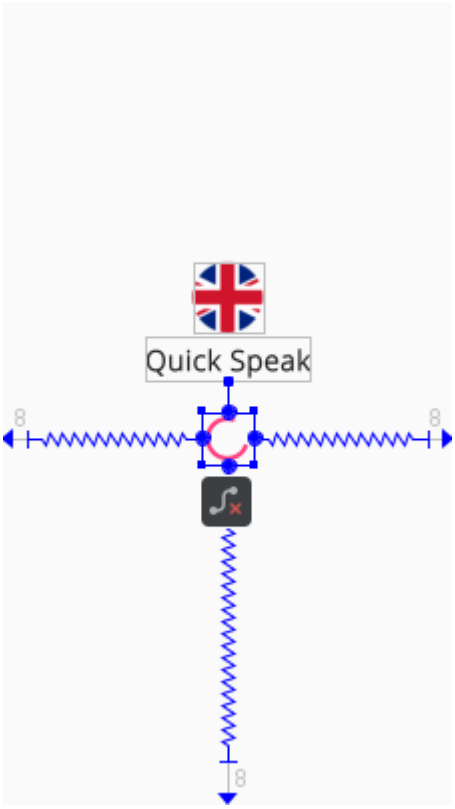
Στις παρακάτω εικόνες παρουσιάζονται τα constraints μεταξύ των components καθώς και η τελική μορφή της διεπαφής του `MainActivity`.



Εικόνα 15: Τα Constraints του ImageView



Εικόνα 16: Τα Constraints του TextView



Εικόνα 17: Τα Constraints του ProgressBar

Εικόνα 18: Η Τελική Μορφή της Διεπαφής του MainActivity

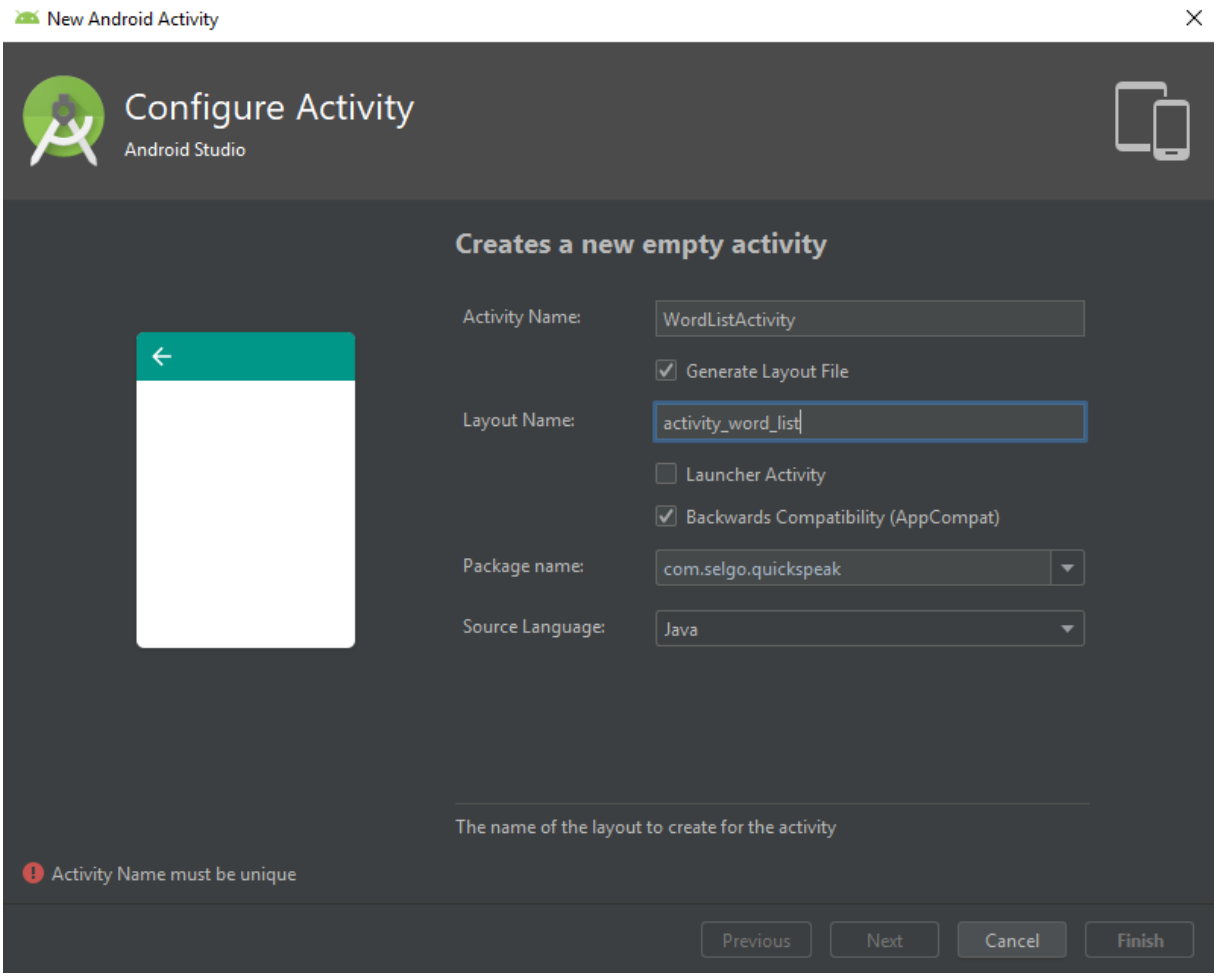
4.5.2 Το Αρχείο MainActivity.java

Το `MainActivity`, υλοποιεί μια απλή διαδικασία. Στην ουσία εμφανίζει στο χρήστη, μια οθόνη με τη διεπαφή που σχεδιάστηκε παραπάνω και μετά από ένα σύντομο χρονικό διάστημα (1.5 δευτερόλεπτο περίπου) μεταφέρει το χρήστη στο επόμενο activity που αφορά το κύριο κομμάτι της εφαρμογής. Στο σημείο αυτό αν και θα ασχοληθούμε μόνο με το συγκεκριμένο activity, θα δημιουργήσουμε το `WordListActivity` για τις ανάγκες του `Intent` όπως θα δούμε.

Ο κώδικας του `MainActivity` υλοποιείται με τη μέθοδο `onCreate`[70]. Οι δύο πρώτες γραμμές στο κώδικα που θα παρουσιαστεί, αφορούν την απαραίτητη κλήση της αντίστοιχης `onCreate` μεθόδου της κλάσης `Activity`, και έπειτα πραγματοποιείται η κλήση της μεθόδου `setContentView`[71]. Τυπικά αυτές οι δύο κλήσεις γίνονται πάντα προτού ξεκινήσουμε να υλοποιούμε οποιοδήποτε κώδικα από εκεί και πέρα. Το όρισμα της `setContentView` στην ουσία είναι το αρχείο `activity_main.xml` που δημιουργήσαμε και αυτό που θέλουμε να χρησιμοποιήσει το `MainActivity` για να σχεδιάσει τη διεπαφή που αναπτύχθηκε. Για να εντοπίσει τα components σε αυτό το αρχείο, το Android API χρησιμοποιεί μια ειδική κλάση που ονομάζεται `R`[72][73]. Η κλάση `R`, δημιουργείται αυτόματα από το Android Studio και εκεί μέσα αποθηκεύονται όλα τα id των resources που υπάρχουν σε ένα project. Ουσιαστικά στο κώδικα ένα id είναι συνδυασμός δύο πραγμάτων. Ένα είναι ο τύπος του resource που χρησιμοποιείται όπως `R.layout`, `R.drawable` κτλ. κι ο άλλος είναι το όνομα του ίδιου του resource. Έτσι στη συγκεκριμένη περίπτωση όπως θα δούμε και στο κώδικα, για να εντοπίσουμε το αρχείο της διάταξης που μας ενδιαφέρει, χρησιμοποιούμε το `R.layout.activity_main`.

Έχοντας κάνει αυτό λοιπόν, μπορούμε πλέον να χρησιμοποιήσουμε τα components της διάταξης και να τα εκμεταλλευτούμε προγραμματιστικά, κάτι που θα δούμε στο κύριο activity του Quick Speak. Η λογική του activity, απαιτεί να εκτελέσει κώδικα μέτα από συγκεκριμένο χρονικό διάστημα. Στη Java αυτό μπορεί να υλοποιηθεί με την έννοια του νήματος (thread). Συνήθως τα νήματα χρησιμοποιούνται, όταν θέλουμε να εκτελέσουμε κάτι παράλληλα με τις κύριες λειτουργίες ενός λογισμικού, παρόλα αυτά αυτό που θα το δημιουργήσουμε εδώ, είναι ένα απλό νήμα το οποίο θα περιμένει για περίπου 1.5 δευτερόλεπτο και στην συνέχεια θα στείλει ένα μήνυμα `Intent` στο `WordListActivity` ώστε να το ξεκινήσει.

Ένα νήμα, υλοποιείται με τη χρήση του αντικειμένου `Handler`[74]. Θα χρησιμοποιήσουμε τη μέθοδο `postDelayed`[75], η οποία εκτελεί κώδικα μετά το πέρας καθορισμένου χρονικού διαστήματος. Η μέθοδος αυτή, δέχεται δύο ορίσματα. Το ένα είναι ένα αντικείμενο `Runnable`[76], μέσω του οποίου υλοποιείται η μέθοδος `run`, που θα εκτελεί το κώδικα του νήματος. Το δεύτερο όρισμα αφορά το χρονικό διάστημα στο οποίο θα περιμένει η `postDelayed` πριν εκτελέσει το κώδικα της `run`. Δημιουργώντας το `WordListActivity` (File -> New -> Activity -> Empty Activity στο Android Studio) ας δούμε το κώδικα του `MainActivity`.



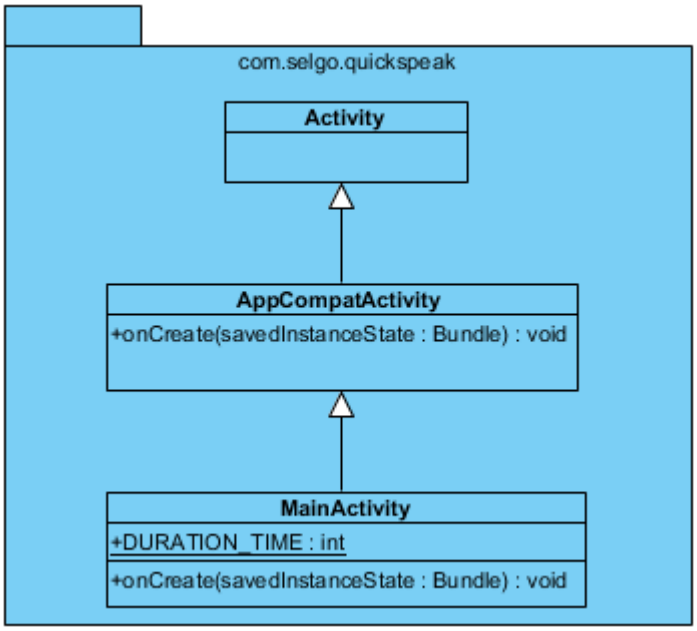
Εικόνα 19: Δημιουργία του WordListActivity

```
package com.selgo.quickspeak;

import android.content.Intent;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    public static int DURATION_TIME = 1500;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        new Handler().postDelayed(new Runnable(){
            @Override
            public void run() {
                Intent intent = new Intent(MainActivity.this, WordListAc-
tivity.class);
                startActivity(intent);
                finish();
            }
        }, DURATION_TIME);
```

```
    }
}
```

Η λογική του κώδικα της μεθόδου run είναι η εξής. Αρχικά δημιουργείται το αντικείμενο `Intent`. Τα ορίσματα του είναι η κλάση από την οποία προέρχεται το μήνυμα (η κλάση αποστολέας αλλιώς) και η κλάση στην οποία θα σταλθεί (η κλάση παραλήπτης). Το `this` χρησιμοποιείται επειδή αναφερόμαστε στο παρόν αντικείμενο της κλάσης - στον εαυτό της αλλιώς - και το `class` αναφέρεται ουσιαστικά στη κλάση `Class`[77] που περιέχει πληροφορίες για τη κλάση που συμπεριλαμβάνει (την `WordListActivity` εδώ). Είναι θα έλεγε κανείς ένας τρόπος να αναφερθούμε προγραμματιστικά σε μια κλάση κι όχι στα αντικείμενα της. Η κλήση `startActivity` είναι αυτή που μεταφέρει τη ροή της εφαρμογής σε επόμενο activity, βάση ενός intent. Στο αυτό το σημείο το MainActivity έχει τελειώσει τη δουλεία του και χρειάζεται να τερματιστεί. Αυτό το τερματισμό πραγματοποιεί η κλήση της μεθόδου finish της κλάσης Activity. Ο λόγος που εργαζόμαστε με τη finish είναι το γεγονός ότι με αυτό το activity, δεν αλληλοεπιδρά ο χρήστης έτσι δε χρειάζεται να περιμένει τις κλήσεις του κύκλου ζωής των activities. Επίσης η διάρκεια «ζωής» ενός τέτοιου activity είναι πολύ σύντομη και η κλήση finish απλώς εξασφαλίζει ότι μόλις τελειώσει το thread και μεταφερθεί ο χρήστης στο επόμενο activity, το παλαιό δεν θα εκτελείται στο παρασκήνιο εφόσον δεν είναι αναγκαίο να επιστρέψουμε ποτέ σε αυτό. Τέλος το χρονικό διάστημα που θα περιμένει το thread προτού εκτελεστεί, δηλώνεται με μια σταθερά μεταβλητή ακεραίου τη `DURATION_TIME` με αρχική τιμή 1500 που αφορά τον χρόνο σε ms, και χρησιμοποιείται ως όρισμα στη `postDelayed`.

Έχοντας υλοποιήσει το αρχικό activity του quick speak, παρουσιάζεται το διάγραμμα UML των κλάσεων που χρησιμοποιήθηκαν.



Εικόνα 20: Το UML του MainActivity

Σχόλιο για το παραπάνω UML. Δεν έχει συμπεριληφθεί η ιεραρχία κλάσεων που υπάρχει μεταξύ των κλάσεων `AppCompatActivity`[78] και `Activity`, καθαρά για λόγους απλότητας και επειδή ουσιαστικά πρόκειται για κλάσεις που χρησιμοποιεί το Android API στο παρασκήνιο.

Παρόλα αυτά, ο λόγος που συμπεριλήφθηκε η κλάση `Activity` στο διάγραμμα, είναι για διασαφηνίσει ότι ο τρόπος με τον οποίο υλοποιούμε μια κλάση-παιδί της `AppCompatActivity` δεν διαφέρει ιδιαίτερα από αυτόν με τον οποίο θα υλοποιούσαμε μια αντίστοιχη κλάση-παιδί της `Activity`. Ο λόγος που χρησιμοποιήθηκε η `AppCompatActivity`, είναι για την υποστήριξη των σύγχρονων βιβλιοθηκών του Android από παλαιότερες εκδόσεις του λειτουργικού.

4.6 Το WordListActivity

Το `WordListActivity` αποτελεί το κύριο μέρος του Quick Speak. Αρχικά θα δημιουργήσουμε τη βασική διάταξη του. Το συστατικό το οποίο θα χρησιμοποιηθεί, είναι το `DrawerLayout`[79]. Ο λόγος που χρησιμοποιήθηκε αυτό είναι το γεγονός ότι επιτρέπει τη προσθήκη ενός side menu με τέτοιο τρόπο ώστε να κρύβεται από την οθόνη όταν είναι ανενεργό και να εμφανίζεται από το πλάι της οθόνης όταν ενεργοποιηθεί. Στο σημείο αυτό, δε θα προσθέσουμε ακόμα το menu μέχρι να υλοποιήσουμε το κυρίως μέρος της εφαρμογής. Έχοντας αναφέρει αυτό, ας δούμε το αρχείο της διάταξης `activity_word_list.xml` με τη προσθήκη μόνο του `DrawerLayout`.

4.6.1 Το Αρχείο word_list_activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">


</android.support.v4.widget.DrawerLayout>
```

4.6.1.1 Το Tool Bar

Είναι αρκετά συνηθισμένο στις εφαρμογές να υπαρχεί μια ειδική μπάρα στη κορυφή των activities, η οποία περιέχει το τίτλο της εφαρμογής ή του μεμονωμένου activity και επίσης επιπλέον λειτουργίες όπως το κουμπί που ενεργοποιεί το side menu όπως θα δούμε. Στην ουσία το app bar[80] (ή αλλιώς και tool bar) φαντάζει κάπως σαν την επικεφαλίδα μιας ιστοσελίδας, όπου ο χρήστης βλέπει ένα λογότυπο και το τίτλο της ιστοσελίδας με ίσως ένα μικρό μενού. Ο κύριος ρόλος του είναι να καθοδηγεί το χρήστη στα διάφορα σημεία της εφαρμογής.

Το tool bar υλοποιείται με τη χρήση του αντικειμένου `ToolBar`[81]. Είναι καλή τεχνική να συμπεριλαμβάνουμε το αντικείμενο αυτό σε ξεχωριστό αρχείο διάταξης. Ο λόγος είναι ότι με αυτόν τον τρόπο αρκεί να δημιουργήσουμε την ετικέτα και τις ιδιότητες του αντικειμένου μια φόρα. Από εκεί και πέρα μπορεί κανείς να συμπεριλάβει το αρχείο αυτό σε οποιοδήποτε αρχείο διάταξης ενός activity. Ας δούμε το τρόπο με τον οποίο πραγματοποιείται αυτό. Θα δημιουργήσουμε το tool bar που θα χρησιμοποιεί το Quick Speak, και θα το προσθέσουμε στο

αρχείο `activity_word_list.xml`. Παρακάτω παρουσιάζεται ο κώδικας του αρχείου `app_toolbar.xml`.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.Toolbar
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="?attr/colorPrimary"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"/>
```

Ορίζουμε το `android:layout_height` με τη τιμή `?attr/actionBarSize`, η οποία θέτει το ύψος του συστατικού σε αυτό ενός τυπικού Android app bar. Πρόκειται για τιμή η οποία είναι προκαθορισμένη από την βιβλιοθήκη του Android. Η ιδιότητα `android:background` έχει τη τιμή `?attr/colorPrimary` η οποία αφορά το primary color που ορίζεται στο colors.xml. Τέλος η ιδιότητα `android:theme` ορίζει το θέμα που θα εφαρμοστεί στο tool bar. Το θέμα που εφαρμόστηκε παραπάνω ανήκει κι αυτό στη βιβλιοθήκη του Android, εγκυκλοπαιδικά βέβαια θα μπορούσαμε να δημιουργήσουμε και κάποιο custom θέμα, το οποίο δεν απασχολεί τη παρούσα πτυχιακή.

Πριν προσθέσουμε τη δομή που περιγράφει το παραπάνω αρχείο, θα πάμε στο αρχείο διάταξης του `WordListActivity` και θα δημιουργήσουμε μια γραμμική διάταξη μέσα στο `DrawerLayout`, με το αντικείμενο `LinearLayout`. Εκεί μέσα θα τοποθετηθεί το tool bar. Έτσι λοιπόν με τις κατάλληλες τροποποιήσεις το αρχείο `activity_word_list.xml` γίνεται:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <include
            layout="@layout/app_toolbar"
            android:id="@+id/app_toolbar"/>

    </LinearLayout>

</android.support.v4.widget.DrawerLayout>
```

Ορίζουμε το πλάτος και ύψος του `LinearLayout` με `match_parent`. Στην ουσία αυτό θα πιάσει όλο το πλάτος και ύψος του `DrawerLayout` το όποιο με τη σειρά του πιάνει όλη την

οθόνη του activity. Το `LinearLayout` τοποθετεί τα συστατικά γραμμικά, είτε οριζόντια είτε κάθετα. Η ιδιότητα που το ορίζει αυτό είναι η `android:orientation` κι όπως βλέπουμε κι από τη τιμή **`vertical`**, τα συστατικά θα τοποθετούνται το ένα μετά από το άλλο καθέτως. Μέσα στο `LinearLayout` αρχικά τοποθετείται το tool bar και του ορίζεται το id **`app_toolbar`**.

Αυτό θα χρησιμοποιηθεί από το `WordListActivity.java` για να τοποθετήσει στο tool bar το τίτλο της εφαρμογής.

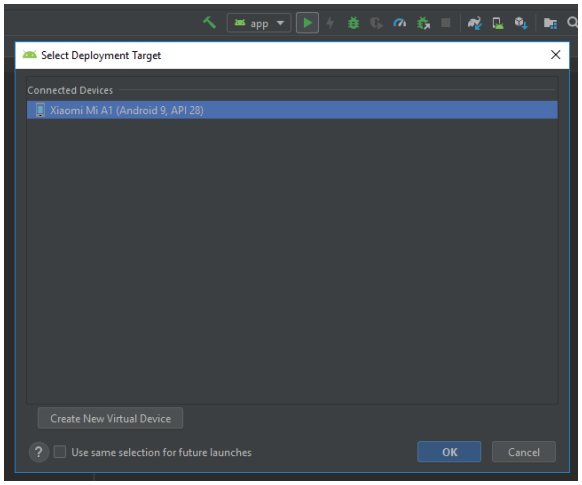
4.6.2 Το WordListActivity.java

Ας δούμε το αρχικό κομμάτι του κώδικα που υλοποιεί το `WordListActivity`. Η μόνη λειτουργικότητα που υλοποιείται εδώ, είναι η τοποθέτηση του tool bar που χρησιμοποιήθηκε παραπάνω και η προσθήκη του τίτλου της εφαρμογής σε αυτό . Έτσι λοιπόν το αρχείο `WordListActivity.java` γίνεται:

```
package com.selgo.quickspeak;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;

public class WordListActivity extends AppCompatActivity {

    private Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_word_list);

        toolbar = (Toolbar) findViewById(R.id.app_toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle(R.string.app_name);
    }
}
```

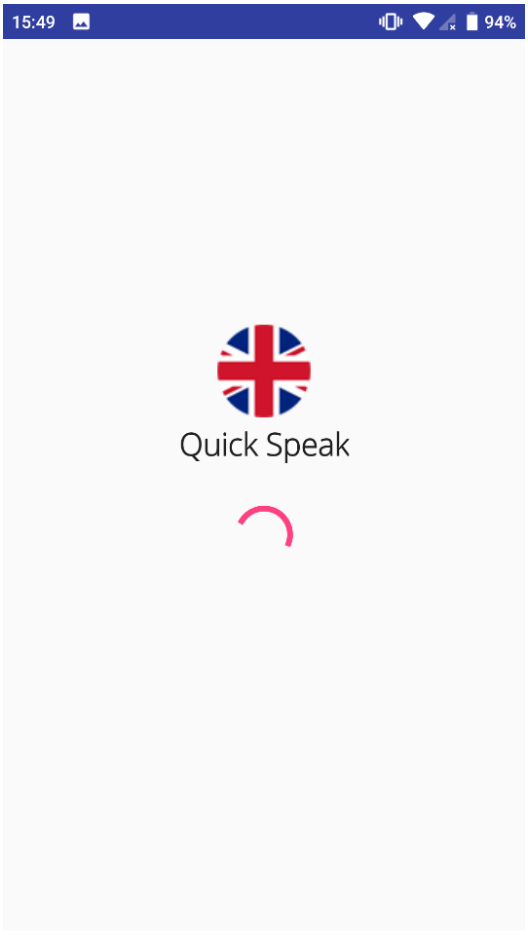
Η κλήση `setSupportActionBar` ορίζει στο κώδικά, το tool bar που θα χρησιμοποιηθεί στο activity το οποίο αποθηκεύεται στο αντικείμενο `toolbar`. Στη συνέχεια μέσω της κλήσης `getSupportActionBar`, ορίζουμε με τη κλήση `setTitle` το τίτλο του tool bar. Ο τίτλος της εφαρμογής βρίσκεται όπως βλέπουμε στο `app_name` που βρίσκεται στο αρχείο `strings.xml`.

Σε αυτό το σημείο, μπορούμε να τρέξουμε την εφαρμογή. Στη παρούσα πτυχιακή, το Quick Speak δοκιμάστηκε σε πραγματική συσκευή εκδόσεως Android 9. Ο λόγος είναι επειδή ο Android Emulator που διαθέτει το Android Studio, δεν πληρούσε τις απαιτήσεις συστήματος, του συστήματος στο οποίο αναπτύχθηκε το Quick Speak ώστε να τρέχει ομαλά. Για να τρέξει μια εφαρμογή, πατάμε στο Android Studio το πράσινο κουμπί στη δεξια κορυφή  και επιλέγουμε την συσκευή (πραγματική ή εικονική) στην οποία θέλουμε να τρέξει η εφαρμογή αυτή και πατάμε το OK. Η διαδικασία αυτή, παρουσιάζεται στη παρακάτω εικόνα:

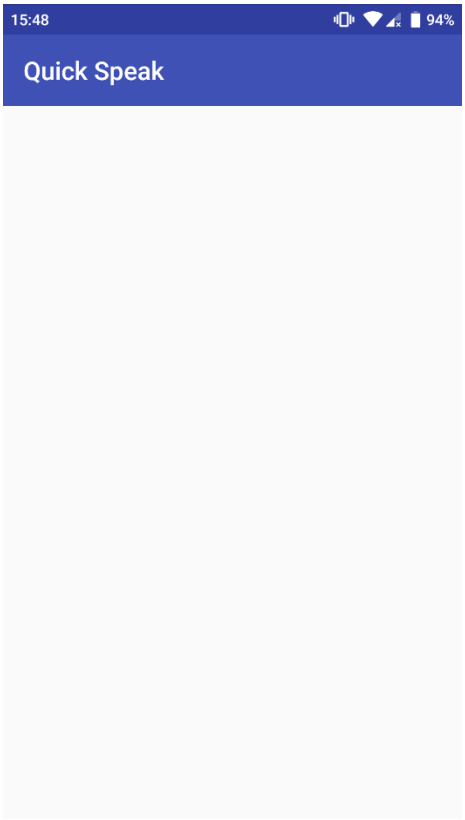


Εικόνα 21: Στιγμιότυπο Εκκίνησης Εκτέλεσης Εφαρμογής σε Πραγματική Συσκευή

Με την παραπάνω υλοποίηση το Quick Speak αρχικά τρέχει το MainActivity για περίπου 1.5 δευτερόλεπτα και στη συνέχεια ενεργοποιεί το WordListActivity, το οποίο παρουσιάζει το tool bar που προσθέσαμε με το τίτλο της εφαρμογής.



Εικόνα 22: Στιγμιότυπο Εκκίνησης του Quick Speak – MainActivity



Εικόνα 23: Στιγμιότυπο Κυρίως Αρχικού Περιεχομένου του Quick Speak – WordListActivity

4.7 Ανάπτυξη της Βάσης Δεδομένων του Quick Speak

Προτού συνεχίσουμε με την υλοποίηση του Quick Speak, χρειαζόμαστε τα λήμματα τα οποία θα προστεθούν ως δεδομένα στην εφαρμογή. Υπάρχουν δύο κυρίως τρόποι με τους οποίους μπορούμε να αντλήσουμε δεδομένα και να τα χρησιμοποιήσουμε σε μια Android εφαρμογή. Ο ένας και πιο βασικός, είναι η χρήση Java κλάσης η οποία θα τα περιγράφει. Κάθε αντικείμενο της κλάσης ουσιαστικά θα είναι μια μορφή δεδομένου. Τα αντικείμενα αυτά συνηθίζεται σε τέτοιες περιπτώσεις να αποθηκεύονται σε μια δυναμική λίστα δεδομένων και όταν χρειάζονται αντλούνται από αυτή για χρήση τους στην εφαρμογή. Για παράδειγμα αν στη περίπτωση του Quick Speak, θέλαμε να επιτύχουμε το παραπάνω, θα αρκούσε μια κλάση - έστω η `EnglishWord` - η οποία για κάθε λήμμα θα είχε την αγγλική λέξη και την αντίστοιχη μετάφραση της στα Ελληνικά. Μια υλοποίηση της κλάσης αυτής θα ήταν:

```
public class EnglishWord {
    private String englishWord;
    private String translatedWord;

    public EnglishWord(String englishWord, String translatedWord) {
        this.englishWord = englishWord;
        this.translatedWord = translatedWord;
    }
    public String getEnglishWord() {
        return englishWord;
```

```
    }

    public String getTranslatedWord() {
        return translatedWord;
    }

    public String toString() {
        return this.englishWord;
    }
}
```

Όπως βλέπουμε, έχουμε να κάνουμε με μια τυπική Java κλάση και τίποτα παραπάνω. Αν τώρα σε ένα activity θέλαμε να χρησιμοποιήσουμε δεδομένα με βάση αυτή τη κλάση, μας αρκεί η λίστα που προαναφέρθηκε η οποία φυσικά θα είναι τύπου `EnglishWord`. Κάθε στοιχείο της, είναι ουσιαστικά ένα αντικείμενο, το οποίο αντιπροσωπεύει το λήμμα στα αγγλικά μαζί με την μετάφραση του.

```
private ArrayList<EnglishWord> words = new ArrayList<>();
words.add(new EnglishWord(" A Book ", " Ένα Βιβλίο "));
```

Μέσω αυτής της λίστας από εκεί και πέρα μπορούμε να εκμεταλλευτούμε τα δεδομένα της, για χρήση τους σε μια εφαρμογή. Αν και είναι ένας αρκετά απλός τρόπος να διατηρήσουμε δεδομένα στην ουσία δεν διατηρούνται για πάντα. Επίσης η ύπαρξη πολλών τέτοιου είδους δεδομένων, μέσω αντικειμένων Java δηλαδή, δημιούργει ένα σημαντικό πρόβλημα της μεγάλης χρήσης περιορισμένης μνήμης. Ειδικά στις φορητές συσκευές η μνήμη παίζει σημαντικό ρόλο στις επιδόσεις τόσο μιας εφαρμογής όσο και του συστήματος στο οποίο τρέχει. Αρά με λίγα λόγια μιλάμε για δεδομένα που δεν διατηρούνται στη συσκευή ακόμα και μετά το πέρας του activity που τα χρησιμοποιεί, και επιπλέον υπάρχει σπατάλη χρήσιμων πόρων. Πώς μπορεί λοιπόν να λυθεί ένα τέτοιου είδους πρόβλημα;

Όταν μιλάμε για διατήρηση δεδομένων, σχεδόν πάντα υπονοείται η ύπαρξη κάποιας βάσης δεδομένων που τα διατηρεί και το Android φυσικά δεν αποτελεί εξαίρεση. Ο 2ος και σημαντικότερος τρόπος με τον οποίο, μπορεί κανείς να διατηρήσει δεδομένα σε μια εφαρμογή, είναι με τη κατασκευή μιας βάσης δεδομένων. Η κατασκευή αυτή σίγουρα σε πολλούς ίσως υπονοήσει τη χρήση κάποιου συστήματος διαχείρισης βάσεων δεδομένων όπως το MySQL για παράδειγμα. Παρόλα αυτά η ανάπτυξη βάσεων στο Android, δεν απαιτεί τίποτα άλλο από μία βιβλιοθήκη. Χρειάζεται μόνο το σύνολο κλάσεων που βρίσκονται στο πακέτο `android.database.sqlite`[82][83] και την ειδική διεπαφή (interface) `BaseColumns`[84].

Στη περίπτωση της πτυχιακής, θα υλοποιηθούν δύο κλάσεις για την ανάπτυξη της βάσης δεδομένων του Quick Speak. Να διασαφηνιστεί ότι δεν είναι απαραίτητη η προχωρημένη εμπειρία στην SQLite για τη μελέτη και την ανάπτυξη της βάσης, καθώς είναι ιδιαίτερα απλή.

4.7.1 Ο Πίνακας της Βάσης Δεδομένων

Η βάση δεδομένων της εφαρμογής, περιέχει ένα και μόνο πίνακα, τουλάχιστον όσον αφορά τα λήμματα των Αγγλικών και τον ονομάζουμε english. Για κάθε στοιχείο (εγγραφή) του πίνακα οι πληροφορίες που χρειαζόμαστε είναι το επίπεδο δυσκολίας των λημμάτων, το λήμμα στα Αγγλικά, η αντίστοιχη μετάφραση του στα Ελληνικά, και ένα id μέσω του οποίου θα εντοπίζουμε το αντίστοιχο ηχητικό απόσπασμα προφοράς του λήμματος. Επίσης σημαντική

πληροφορία είναι η ύπαρξη ενός μοναδικού ID (το οποίο ονομάζεται ως _ID) κάθε εγγραφής του πίνακα και φυσικά το όνομα του. Ο πίνακας λοιπόν θα έχει μια μορφή όπως τη παρακάτω:

english				
_ID	*level*	*english_word*	*translated_word*	*word_sound_id*
1	1	apple	μήλο	R.raw.apple

Πίνακας 1: Ο Πίνακας english στο Σχεσιακό Μοντέλο της Βάσης – Ενδεικτικό Παράδειγμα μιας Εγγραφής Του Πίνακα

Θα περιγράψουμε τα στοιχεία αυτά με μια κλάση που θα την ονομάσουμε `WordContract`. Ο ρόλος της κλάσης αυτής, είναι να κρατήσει τις πληροφορίες του πίνακα μέσω σταθερών μεταβλητών. Οι σταθερές αυτές θα χρησιμοποιηθούν στα SQL ερωτήματα, που θα χρησιμοποιηθούν για τη διαχείριση της βάσης και της αρχικής δημιουργίας του πίνακα. Η τεχνική υλοποίησης μια τέτοιας κλάσης είναι αρχικά να δηλωθεί ο κατασκευαστής της (constructor) ως ιδιωτικός έναντι του δημόσιου που γίνεται συνήθως, έτσι ώστε να μην αρχικοποιηθεί από άλλες κλάσεις. Στη συνέχεια υλοποιείται μια εσωτερική κλάση που ουσιαστικά περιγράφει τα στοιχεία ενός πίνακα η οποία υλοποιεί την `BaseColumns`. Παρακάτω παρουσιάζεται ο κώδικας της κλάσης `WordContract`.

```
package com.selgo.quickspeak.data;

import android.provider.BaseColumns;

public class WordContract {

    private WordContract() {}

    public static final class WordEntry implements BaseColumns {

        public final static String TABLE_NAME = "english";

        public final static String _ID = BaseColumns._ID;
        public final static String COLUMN_LEVEL = "level";
        public final static String COLUMN_ENGLISH_WORD = "english_word";
        public final static String COLUMN_TRANSLATED_WORD = "trans-
lated_word";
        public final static String COLUMN_WORD_SOUND_ID = "word_sound_id";

    }
}
```

4.7.2 Η Κλάση WordDbHelper

Το Android, δημιουργεί και διαχειρίζεται μια βάση δεδομένων, με τη χρήση του SQLite. Στην ουσία υλοποιείται μια κλάση που κληρονομεί από τη κλάση `SQLiteOpenHelper`[85]. Όπως λέει και το όνομα της κλάσης, χρησιμοποιείται ένα αντικείμενο helper το οποίο αρχικά δημιουργεί τη βάση εφόσον δεν υπάρχει και διαχειρίζεται με διάφορες κλήσεις, αυτή και τα δεδομένα της.

Η υλοποίηση της βάσης του Quick Speak, θα γίνει με τη κλάση `WordDbHelper` της οποίας ο κώδικας παρουσιάζεται παρακάτω.

```
package com.selgo.quickspeak.data;

import android.content.ContentValues;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import com.selgo.quickspeak.R;
import com.selgo.quickspeak.data.WordContract.WordEntry;

public class WordDbHelper extends SQLiteOpenHelper {

    private static final String DATABASE_NAME = "quick_speak.db";
    private static final int DATABASE_VERSION = 1;

    public WordDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String SQL_CREATE_ENGLISH_TABLE = "CREATE TABLE " + WordEntry.TA-
BLE_NAME + "("
                + WordEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + WordEntry.COLUMN_LEVEL + " TEXT NOT NULL, "
                + WordEntry.COLUMN_ENGLISH_WORD + " TEXT, "
                + WordEntry.COLUMN_TRANSLATED_WORD + " TEXT, "
                + WordEntry.COLUMN_WORD_SOUND_ID + " TEXT);";

        db.execSQL(SQL_CREATE_ENGLISH_TABLE);

        /* Level 1 Words */
        insertWord(db, "Level 1", "A a", "Το 1ο γράμμα του αγγλικού
αλφαβήτου", R.raw.a);
        insertWord(db, "Level 1", "B b", "Το 2ο γράμμα του αγγλικού αλφαβή-
του", R.raw.b);
        insertWord(db, "Level 1", "C c", "Το 3ο γράμμα του αγγλικού αλφαβή-
του", R.raw.c);
        insertWord(db, "Level 1", "D d", "Το 4ο γράμμα του αγγλικού αλφαβή-
του", R.raw.d);
        insertWord(db, "Level 1", "E e", "Το 5ο γράμμα του αγγλικού αλφαβή-
του", R.raw.e);
        insertWord(db, "Level 1", "F f", "Το 6ο γράμμα του αγγλικού αλφαβή-
του", R.raw.f);
        insertWord(db, "Level 1", "G g", "Το 7ο γράμμα του αγγλικού αλφαβή-
του", R.raw.g);
```

```
        insertWord(db, "Level 1", "H h", "Το 8ο γράμμα του αγγλικού αλφαβή-
του", R.raw.h);
        insertWord(db, "Level 1", "I i", "Το 9ο γράμμα του αγγλικού αλφαβή-
του", R.raw.i);
        insertWord(db, "Level 1", "J j", "Το 10ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.j);
        insertWord(db, "Level 1", "K k", "Το 11ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.k);
        insertWord(db, "Level 1", "L l", "Το 12ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.l);
        insertWord(db, "Level 1", "M m", "Το 13ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.m);
        insertWord(db, "Level 1", "N n", "Το 14ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.n);
        insertWord(db, "Level 1", "O o", "Το 15ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.o);
        insertWord(db, "Level 1", "P p", "Το 16ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.p);
        insertWord(db, "Level 1", "Q q", "Το 17ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.q);
        insertWord(db, "Level 1", "R r", "Το 18ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.r);
        insertWord(db, "Level 1", "S s", "Το 19ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.s);
        insertWord(db, "Level 1", "T t", "Το 20ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.t);
        insertWord(db, "Level 1", "U u", "Το 21ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.u);
        insertWord(db, "Level 1", "V v", "Το 22ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.v);
        insertWord(db, "Level 1", "W w", "Το 23ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.w);
        insertWord(db, "Level 1", "X x", "Το 24ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.x);
        insertWord(db, "Level 1", "Y y", "Το 25ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.y);
        insertWord(db, "Level 1", "Z z", "Το 26ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.z);

        /* Level 2 Words */
        insertWord(db, "Level 2", "Hi", "Γειά σου", R.raw.hi);
        insertWord(db, "Level 2", "Let's go", "Πάμε", R.raw.let_s_go);
        insertWord(db, "Level 2", "please", "παρακαλώ", R.raw.please);
        insertWord(db, "Level 2", "good morning", "καλημέρα",
R.raw.good_morning);
        insertWord(db, "Level 2", "good night", "καληνύχτα",
R.raw.good_night);
        insertWord(db, "Level 2", "yes", "ναι", R.raw.yes);
        insertWord(db, "Level 2", "no", "όχι", R.raw.no);
        insertWord(db, "Level 2", "cheers", "Στην υγειά μας!",
R.raw.cheers);
        insertWord(db, "Level 2", "thank you", "ευχαριστώ",
R.raw.thank_you);
        insertWord(db, "Level 2", "you are welcome", "παρακαλώ (όταν
προσφέρω κάτι)", R.raw.you_re_welcome);
        insertWord(db, "Level 2", "I'm sorry", "Συγνώμη / Λυπάμαι",
R.raw.i_m_sorry);
        insertWord(db, "Level 2", "see you later", "τα λέμε μετά",
R.raw.see_you_later);
        insertWord(db, "Level 2", "goodbye", "αντίο", R.raw.goodbye);
```

```
    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

    }

    private static void insertWord(SQLiteDatabase db, String level, String
englishWord, String translatedWord) {

        ContentValues wordValues = new ContentValues();

        wordValues.put(WordEntry.COLUMN_LEVEL, level);
        wordValues.put(WordEntry.COLUMN_ENGLISH_WORD, englishWord);
        wordValues.put(WordEntry.COLUMN_TRANSLATED_WORD, translatedWord);

        db.insert(WordEntry.TABLE_NAME, null, wordValues);
    }

    private static void insertWord(SQLiteDatabase db, String level, String
englishWord, String translatedWord, int soundId) {

        ContentValues wordValues = new ContentValues();

        wordValues.put(WordEntry.COLUMN_LEVEL, level);
        wordValues.put(WordEntry.COLUMN_ENGLISH_WORD, englishWord);
        wordValues.put(WordEntry.COLUMN_TRANSLATED_WORD, translatedWord);
        wordValues.put(WordEntry.COLUMN_WORD_SOUND_ID, soundId);

        db.insert(WordEntry.TABLE_NAME, null, wordValues);
    }
}
```

Για τη βάση διατηρούμε ένα db αρχείο όπως θα έκανε κάνεις κλασσικά σε μια τυπική SQL βάση και επιπλέον διατηρούμε και έναν αριθμό έκδοσης. Ο αριθμός αυτός αφορά τη μελλοντική αναβάθμιση της βάσης, κάτι που δεν θα απασχολήσει τη παρούσα πτυχιακή παρόλα αυτά είναι καλή τεχνική να υπαρχεί. Ο αριθμός αυτός αλλάζει από 1 σε 2 κτλ. για κάθε αναβάθμιση της βάσης και μέσω του αριθμού αυτού με τις κατάλληλες κλήσεις, η βάση τροποποιείται με τέτοιο τρόπο ώστε όλοι οι χρήστες να είναι ενήμεροι με τη πρόσφατη έκδοση της βάσης και των δεδομένων της. Η μέθοδος `onUpgrade`[86] είναι αυτή που πραγματοποιεί την αναβάθμιση της βάσης.

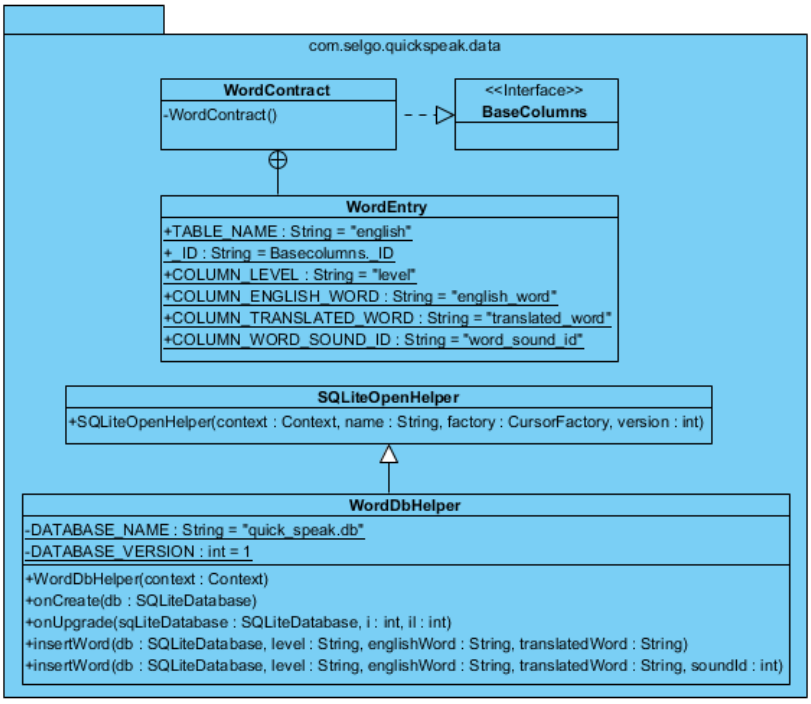
Αρχικά περνάμε στο constructor της κλάσης `SQLiteOpenHelper`, το όνομα και την έκδοση της βάσης που θα χρησιμοποιηθεί. Στην ουσία όπως βλέπουμε, το όνομα ισοδυναμεί με το όνομα του αρχείου db της βάσης. Η μέθοδος `onCreate`[87], δημιουργεί τη βάση εφόσον αυτή δεν υπαρχεί, χρησιμοποιώντας ένα κλασσικό SQL ερώτημα (SQL query) αποθηκευμένο σε μια μεταβλητή τύπου `String`. Το ερώτημα εκμεταλλεύεται τις μεταβλητές της κλάσης `WordContract`, και η χρήση τους εξασφαλίζει ότι δεν γίνονται τυχόν λάθη στις ονοματολογίες του πίνακα και των στηλών (columns) του. Επίσης αν θελήσει κάποιος να αλλάξει κάποιο από αυτά τα ονόματα, μπορεί να το κάνει από τη κλάση `WordContract`, χωρίς να χρειάζεται να αλλάξει οτιδήποτε άλλο. Η μεταβλητή μεταβιβάζεται ως όρισμα στη μέθοδο `execSQL`[88] του αντικειμένου `db` της κλάσης `SQLiteDatabase`. Αυτή η κλήση εκτελεί το ερώτημα δημιουργώντας το πίνακα `english`.

Έχοντας δημιουργήσει λοιπόν τη βάση και τον πίνακα των λημμάτων, μπορούμε να προσθέσουμε δεδομένα σε αυτόν. Αυτό γίνεται με τη κλήση της μεθόδου `insertWord`. Η

μέθοδος αυτή έχει δύο μορφές χρησιμοποιώντας τη τεχνική method overloading της Java[89]. Η μια και κυρίως μορφή της, δέχεται ως όρισμα το αντικείμενο της βάσης, και όλα τα columns (πλην του μοναδικού _ID) του πίνακα `english`, ενώ η δεύτερη μορφή παραλείπει το id των ηχητικών αποσπασμάτων κάθε λήμματος σε περίπτωση που αυτό δεν είναι διαθέσιμο. Όλα τα columns αποθηκεύονται σε ένα ειδικό αντικείμενο της κλάσης `ContentValues`[90]. Η προσθήκη τους στο αντικείμενο αυτό γίνεται ανά ζεύγη, μέσω της μεθόδου `put`, με την έννοια ότι συσχετίζουμε ένα προς ένα τα columns του πίνακα, με τις αντίστοιχες τιμές τους. Το αντικείμενο της κλάσης `ContentValues`, η οποία πλέον περιέχει τα δεδομένα προς προσθήκη, μεταβιβάζεται ως όρισμα στη μέθοδο `insert` που ουσιαστικά κάνει τη προσθήκη των δεδομένων στη βάση. Έτσι κάθε αυτό το τρόπο, κάθε φορά που καλείται η `insertWord`, προστίθεται στον english μια νέα εγγραφή. Να τονιστεί ότι, στο παραπάνω κώδικα, δεν έχουν συμπεριληφθεί όλα τα λήμματα της εφαρμογής, για λόγους απλότητας του κειμένου, παρόλα αυτά εργαζόμαστε ομοίως για όλα, με τον τρόπο που παρουσιάστηκε εδώ. Ο ολοκληρωμένος κώδικας θα βρίσκεται στο τέλος της παρουσίασης της υλοποίησης του Quick Speak καθώς και στο συνοδευτικό υλικό της παρούσας πτυχιακής.

Επιπλέον χρειάζεται να συμπεριληφθούν τα ηχητικά αποσπάσματα των λημμάτων του Quick Speak. Το εργαλείο που χρησιμοποιήθηκε, παρουσιάζεται στο παράρτημα της πτυχιακής. Τα ηχητικά αποσπάσματα, θα συμπεριλαμβάνονται στο συνοδευτικό υλικό, στο φάκελο `raw` ο οποίος φυσικά θα βρίσκεται κάτω από το φάκελο `res` του project του Quick Speak.

Ας ρίξουμε μια ματιά στο διάγραμμα κλάσεων των 2 κλάσεων που χρησιμοποιήθηκαν για την υλοποίηση της βάσης δεδομένων του Quick Speak.



Εικόνα 24: Το Διάγραμμα Κλάσεων της Υλοποίησης της Βάσης Δεδομένων του Quick Speak

Στο επόμενο κεφάλαιο θα γίνει η ανάπτυξη ενός fragment, το οποίο θα περιέχει το κυρίως περιεχόμενο του Quick Speak. Επιπλέον θα δούμε στη πράξη, την έννοια του adapter, υλοποιώντας έναν που θα διασυνδέει τα δεδομένα της βάσης μέσω ερωτήματος σε αυτήν, με views μιας τυπικής XML δομής. Τέλος θα γίνουν τροποποιήσεις στο project, έτσι ώστε να ολοκληρωθεί πλέον η υλοποίηση της εφαρμογής.

5. Δεύτερο Μέρος Υλοποίησης του Quick Speak

Στο παρόν κεφάλαιο, θα υλοποιήσουμε τα τελικά κομμάτια του Quick Speak. Το κυρίως μέρος της εφαρμογής όπως ειπώθηκε, αφορά τη λίστα με την οποία εμφανίζονται τα λήμματα του λεξικού, χωρισμένα σε επίπεδα δυσκολίας. Επίσης για κάθε λήμμα, υπάρχει διαθέσιμο, ηχητικό υλικό που αφορά τη προφορά του.

5.1 Ο CursorAdapter του Quick Speak

Προκειμένου να υλοποιηθεί το fragment που θα περιέχει τις λίστες, είναι απαραίτητο να περιγράψουμε το τρόπο με τον οποίο θα απεικονίζεται κάθε στοιχείο της λίστας. Αυτό γίνεται όπως γνωρίζουμε, δηλαδή με τη δημιουργία ενός αρχείου διάταξης XML. Όπως είδαμε, ένα σημαντικό εργαλείο στη δημιουργία λιστών δεδομένων, είναι ο adapter, ο οποίος ενώνει τα δεδομένα με τα views μιας διάταξης. Κάθε στοιχείο της λίστας, ακολουθεί πιστά τη διάταξη που περιγράφεται και περιέχει τους ίδιους τύπους δεδομένων. Το μόνο που είναι διαφορετικό, είναι φυσικά τα δεδομένα κάθε στοιχείου της λίστας.

Υπάρχουν διάφοροι τύποι adapters στο Android, ανάλογα κυρίως με το μέσο από το οποίο αντλούνται τα δεδομένα, για τη σύνδεση τους με τα αντίστοιχα views. Ο adapter που θα χρησιμοποιηθεί στη περίπτωση του Quick Speak, είναι ο `CursorAdapter`[91][92]. Στο κεφάλαιο 4 είδαμε το τρόπο με τον οποίο υλοποιείται μια βάση δεδομένων. Το επόμενο βήμα είναι να υλοποιηθεί ο τρόπος με τον οποίο η εφαρμογή θα ζητάει τα δεδομένα, μέσω SQLite ερωτημάτων.

Το εργαλείο που χρησιμοποιείται για την αναζήτηση και ανάκτηση δεδομένων από μια βάση, ονομάζεται `Cursor`[93]. Πρόκειται για μια κλάση, η οποία αναλαμβάνει να αναζητήσει δεδομένα, βάσει ενός SQLite ερωτήματος και να τα αποθηκεύσει στο αντικείμενο της. Ένας cursor, φαντάζει κάτι σαν ένα δείκτη , ο οποίος με βάσει το ερώτημα, αναζητά και επιστρέφει από ένα πίνακα, το αποτέλεσμα του ερωτήματος αυτού, σε μορφή πίνακα, όπως γίνεται δηλαδή και με ένα τυπικό SQL ερώτημα. Ο cursor έχει την ικανότητα να διατρέχει εγγραφή προς εγγραφή τα αποτελέσματα του ερωτήματος. Ο `CursorAdapter`, όπως δηλώνει και το όνομα του, εκμεταλλεύεται ουσιαστικά ένα `Cursor` αντικείμενο, για να συνδέσει τα δεδομένα που αποθηκεύονται σε αυτό, με τα εκάστοτε views του αρχείου διάταξης που χρησιμοποιείται για τα στοιχεία μιας λίστας.

Ένας cursor adapter, υλοποιείται με μία κλάση, που κληρονομεί τα στοιχεία του. Η κλάση που θα χρησιμοποιηθεί είναι η `WordCursorAdapter`. Ο λόγος που χρησιμοποιείται η κλάση αύτη, έναντι της κλάσης `CursorAdapter` και μόνο, είναι ότι χρειαζόμαστε να χειριστούμε παραπάνω του ενός τύπου δεδομένων. Κάθε adapter, υλοποιεί από μόνος του, τη λογική ώστε να διασυνδέσει ένα μόνο View με ένα τύπο δεδομένου που τον αντιστοιχεί, χρησιμοποιώντας προ-καθορισμένες διατάξεις του Android, όπως την `android.R.simple_list_item_1` για παράδειγμα. Η υλοποίηση μιας κλάσης-παιδί ενός adapter, μας δίνει τη δυνατότητα να υλοποιήσουμε τη λογική σύνδεσης views-data, χρησιμοποιώντας οποιαδήποτε διάταξη και οσοδήποτε τύπους δεδομένων θέλουμε. Η σύνδεση των δεδομένων με τα views, γίνεται μέσω δύο μεθόδων πέρα της απαραίτητης κλήσης του constructor της κλάσης ώστε να κληθεί η μέθοδος super της `CursorAdapter`. Αρχικά

υλοποιούμε τη μέθοδο `newView`[94], η οποία επιστρέφει σε ένα αντικείμενο `View`, το group view που βρίσκεται στη κορυφή ενός αρχείου διάταξης XML. Το αρχείο που θα χρησιμοποιηθεί εδώ είναι το `list_item.xml`. Μέσω αυτού του αντικείμενου `View`, μπορούμε να αναφερθούμε στα συστατικά που περιέχονται στο group, με τη κλασσική μορφή αντικειμένων Java, μέσω της χρήσης της `findViewById`. Η επόμενη μέθοδος είναι η `bindView`[95] στην οποία γίνεται η σύνδεση των δεδομένων με τα views. Έτσι με τον adapter αυτόν, η λογική έχει ως εξής. Δημιουργείται το view group μιας διάταξης και στη συνέχεια ορίζεται η λογική με την οποία τα δεδομένα αντιστοιχίζονται στα views της διάταξης αυτής τα οποία φυσικά ανήκουν στο view group που δημιουργήθηκε. Η αρχική μορφή της κλάσης WordCursorAdapter, είναι η εξής:

```
package com.selgo.quickspeak;

import android.content.Context;
import android.database.Cursor;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CursorAdapter;
import android.widget.TextView;


public class WordCursorAdapter extends CursorAdapter {

    public WordCursorAdapter(Context context, Cursor cursor) {
        super(context, cursor, 0);
    }

    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        return LayoutInflater.from(context).inflate(R.layout.list_item,
parent, false);
    }

    @Override
    public void bindView(View view, Context context, final Cursor cursor) {

        TextView word = (TextView) view.findViewById(R.id.language_word);
        TextView translatedWord = (TextView) view.findViewById(R.id.trans-
lated_word);

        String wordText = cursor.getString(1);
        String translatedWordText = cursor.getString(2);

        word.setText(wordText);
        translatedWord.setText(translatedWordText);

    }
}
```

Ο adapter βασίζεται στην υπάρξη ενός cursor με τα δεδομένα που θα χρησιμοποιηθούν. Ο cursor δημιουργείται στο fragment που θα υλοποιηθεί στη συνέχεια. Παρόλα αυτά, μπορεί κάποιος να παρατηρήσει στο σημείο αυτό, ότι η παρούσα βασική μορφή του adapter, προϋποθέτει την ύπαρξη 2 text view, ένα για το λήμμα της Αγγλικής γλώσσας και ένα για την αντίστοιχη μετάφραση του στα ελληνικά.

Ο λόγος που αρχικά δημιουργήσαμε τον adapter, πριν δημιουργήσουμε το fragment, είναι για να υπάρχει μια ξεκάθαρη εικόνα των δεδομένων που θα χρειαστούμε να τοποθετηθούν στη λίστα. Έτσι στη περίπτωση μας γνωρίζουμε ήδη ότι μιλάμε για μια λίστα, κάθε στοιχείο της οποίας αποτελείται από δύο κείμενα. Το κομμάτι πού μένει να υλοποιηθεί ακόμα, στον cursor adapter, είναι αυτό των ηχητικών αποσπασμάτων.

5.1.1 Η Κλάση MediaPlayer και η Υλοποίηση Ήχου στο Quick Speak

Το Android, υποστηρίζει τη δυνατότητα αναπαραγωγής ήχου, με τη χρήση της κλάσης `MediaPlayer`[96][97]. Ένας media player, λειτουργεί ως εξής. Αρχικά δημιουργούμε τον player με τη χρήση της μεθόδου `create`[98] η οποία δέχεται ένα ηχητικό αρχείο προς αναπαραγωγή. Στη συνέχεια ενεργοποιούμε τον player με τη χρήση της μεθόδου `start`[99], όπου στην ουσία αναπαράγεται το αρχείο αυτό. Τέλος απενεργοποιούμε τον player για να απελευθερώσουμε πόρους από το σύστημα με τη χρήσης της μεθόδου `release`[100]. Συνηθισμένη τεχνική στο σημείο αυτό, είναι η χρήση ενός ειδικού listener, του `onCompletionListener`[101], ο οποίος εκτελεί κώδικα μετά την ολοκλήρωση της αναπαραγωγής ενός ηχητικού αρχείου, στην ουσία εδώ μπορούμε να κάνουμε τη παραπάνω απελευθέρωση.

Η παραπάνω λογική θα υλοποιηθεί μέσω ενός click listener[102], που θα ενεργοποιείται από το πάτημα ενός image button, ενός κουμπιού με εικόνα δηλαδή. Το κουμπί αυτό έχει το ρόλο της ηχητικής προφοράς των λημμάτων του λεξικού.

Έτσι έχοντας αναφέρει όλα τα παραπάνω, ας πάμε να δούμε τη τελική μορφή της κλάσης `WordCursorAdapter`:

```
package com.selgo.quickspeak;

import android.content.Context;
import android.database.Cursor;
import android.media.MediaPlayer;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CursorAdapter;
import android.widget.ImageButton;
import android.widget.TextView;


public class WordCursorAdapter extends CursorAdapter {

    private ImageButton imageButton;
    private MediaPlayer mediaPlayer;

    public WordCursorAdapter(Context context, Cursor cursor) {
        super(context, cursor, 0);
    }

    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        return LayoutInflater.from(context).inflate(R.layout.list_item,
parent, false);
    }

    @Override
    public void bindView(View view, Context context, final Cursor cursor) {

        TextView word = (TextView) view.findViewById(R.id.language_word);
        TextView translatedWord = (TextView) view.findViewById(R.id.trans-
lated_word);
```

```
        imageButton = (ImageButton) view.findViewById(R.id.image_button);

        String wordText = cursor.getString(1);
        String translatedWordText = cursor.getString(2);
        final int soundId = cursor.getInt(3);

        word.setText(wordText);
        translatedWord.setText(translatedWordText);

        View.OnClickListener onClickListener = new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                mediaPlayer = MediaPlayer.create(view.getContext(),
soundId);
                mediaPlayer.start();
                mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCom-
pletionListener() {
                    @Override
                    public void onCompletion(MediaPlayer mediaPlayer) {
                        mediaPlayer.release();
                    }
                });
            }
        };

        imageButton.setOnClickListener(onClickListener);
    }
}
```

Τελικά σχόλια για τον adapter. Οι μεταβλητές `wordText`, `translatedWordText` και `soundId`, αφορούν τα δεδομένα που λήφθηκαν από το cursor που θα δημιουργήσουμε στο fragment της εφαρμογής. Στην ουσία μέσω αυτού, γίνεται η κλήση της εκάστοτε μεθόδου που αφορά το τύπο δεδομένων που θέλουμε να αποθηκεύσουμε σε αυτές τις μεταβλητές (`getString` για String `getInt` για int και ούτω κάθε εξής). Το όρισμα των μεθόδων αυτών, αντιστοιχεί στον αριθμό στήλης από την οποία θέλουμε ένα δεδομένο. Όπως είδαμε, ένας cursor αποθηκεύει δεδομένα σε μορφή πίνακα. Άρα μπορούμε να πούμε ότι συμφώνα με το παραπάνω adapter, ο cursor που αναμένεται να ληφθεί από το fragment, θα αφορά ένα αποτέλεσμα σε μορφή εγγραφής, η οποία περιέχει τρία στοιχεία, το λήμμα, τη μετάφραση του στα ελληνικά και το id του ηχητικού αποσπάσματος. Μια λίγα λόγια, έχουμε ένα ερώτημα στη βάση, το αποτέλεσμα του επιστρέφει δεδομένα και αποθηκεύονται στον cursor. Έπειτα αντιστοιχίζονται εγγραφή προς εγγραφή με το κάθε στοιχείο μιας λίστας, όπου στη περίπτωση μας αυτή θα βρίσκεται μέσα στο fragment που θα υλοποιηθεί.

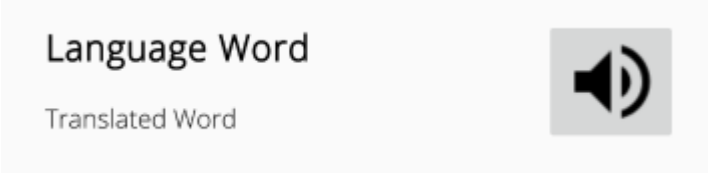
Παρακάτω παρουσιάζεται, η τελική μορφή του αρχείου διάταξης συμπεριλαμβανομένου και του `ImageButton`[103] για την αναπαραγωγή ήχου. Στη περίπτωση αυτή, βλέπουμε το τρόπο με τον οποίο μπορούμε να συνδυάσουμε εμφωλευμένα layouts για να αναπτύξουμε μια διεπαφή.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/card_view"
```

```
android:layout_gravity="center"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:padding="10dp"
android:layout_margin="10dp">

<android.support.constraint.ConstraintLayout
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/language_word"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:fontFamily="@font/open_sans_light"
        android:text="Language Word"
        android:textColor="@android:color/black"
        android:textSize="20sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/guideline3"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />

    <TextView
        android:id="@+id/translated_word"
        style="@style/LanguageTranslatedWordTextStyle"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:fontFamily="@font/open_sans_light"
        android:text="Translated Word"
        android:textColor="@android:color/black"
        android:textSize="15sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/guideline3"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/language_word" />

    <ImageButton
        android:id="@+id/image_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="@+id/guideline3"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/baseline_volume_up_black_36" />
```

```
        <android.support.constraint.Guideline
            android:id="@+id/guideline3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            app:layout_constraintGuide_percent="0.8" />

    </android.support.constraint.ConstraintLayout>
</LinearLayout>
```

Η διεπαφή που δημιουργεί το αρχείο αυτό, είναι η εξής:



Εικόνα 25: Η Διεπαφή του list_item.xml

5.2 Το WordFragment

Η υλοποίηση ενός fragment, είναι παρόμοια με ενός activity, με τη διαφορά ότι όπως είδαμε είναι ανεξάρτητο από αυτό. Ο λόγος που συμβαίνει αυτό, είναι κυρίως επειδή το fragment αποτελεί μια υπό-διεπαφή παρόμοια με αυτή ενός widget και έχει τη δυνατότητα να τοποθετηθεί σε πολλαπλά activities μιας εφαρμογής. Έτσι η σχεδίαση και ανάπτυξη τους, γίνεται με τρόπο τέτοιο ώστε να γνωρίζει όσο είναι δυνατόν, ελάχιστες πληροφορίες για το activity.

Η κλάση με την οποία θα αναπτύξουμε το fragment, είναι η `WordFragment`. Η λογική που υλοποιεί, είναι σχετικά απλή. Ουσιαστικά παρουσιάζεται μια λίστα με τα λήμματα του λεξικού χωρισμένα ανά επίπεδα δυσκολίας. Τα δεδομένα των λημμάτων αναζητούνται όπως θα δούμε με ερώτημα στη βάση δεδομένων το οποίο πραγματοποιείται με τη χρήση του αντικειμένου της κλάσης `Cursor`. Το αρχείο διάταξης που χρησιμοποιείται για τη διεπαφή του fragment, είναι το `fragment_word.xml`. Η διεπαφή του, είναι απλή. Στην ουσία χρησιμοποιείται μια γραμμική διάταξη, η οποία περιέχει ένα `ListView`. Ο κώδικας XML του αρχείου παρουσιάζεται παρακάτω:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ListView
        android:id="@+id/word_listview"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Ας πάμε να δούμε την υλοποίηση του WordFragment και τη λογική με την οποία λειτουργεί στη πράξη:

```
package com.selgo.quickspeak;

import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.Toast;

import com.selgo.quickspeak.data.WordContract;
import com.selgo.quickspeak.data.WordDbHelper;


public class WordFragment extends Fragment {

    private SQLiteDatabase db;
    private Cursor cursor;
    private WordDbHelper mDbHelper;
    private ListView listView;
    private LinearLayout linearLayout;
    private String levelValue;

    public WordFragment() {
        // Required empty public constructor
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        linearLayout = (LinearLayout) inflater.inflate(R.layout.frag-
ment_word, container, false);
        return linearLayout;
    }

    @Override
    public void onStart() {
        super.onStart();

        View view = getView();

        levelValue = (String) getArguments().getCharSequence("itemTitle");
        wordsRetrieveQuery(view, levelValue);
    }

    public void wordsRetrieveQuery(View view, String conditionValue) {
        try {
            mDbHelper = new WordDbHelper(view.getContext());
            db = mDbHelper.getReadableDatabase();

            cursor = db.query(WordContract.WordEntry.TABLE_NAME,
```

```
                    new String[]{WordContract.WordEntry._ID, WordCon-
tract.WordEntry.COLUMN_ENGLISH_WORD, WordContract.WordEntry.COLUMN_TRANS-
LATED_WORD, WordContract.WordEntry.COLUMN_WORD_SOUND_ID},
                    WordContract.WordEntry.COLUMN_LEVEL + "=?",
                    new String[] {conditionValue},
                    null, null, null);


            listView = view.findViewById(R.id.word_listview);
            WordCursorAdapter wordCursorAdapter = new WordCur-
sorAdapter(view.getContext(), cursor);
            listView.setAdapter(wordCursorAdapter);

        } catch (SQLException e) {
            Toast toast = Toast.makeText(view.getContext(), "Database una-
vailable", Toast.LENGTH_SHORT);
            toast.show();
        }
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        cursor.close();
        db.close();
    }
}
```

Όπως ένα activity, διαβάζει ένα αρχείο διάταξης για να μετατρέψει τα components του σε αντικείμενα, έτσι και το fragment λειτουργεί με παρόμοιο τρόπο. Ουσιαστικά λειτουργεί με όμοιο τρόπο με αυτόν που είδαμε στη περίπτωση του `WordCursorAdapter`, δηλαδή διαβάζει - το λεγόμενο inflate - ένα αρχείο XML και τοποθετεί σε ένα αντικείμενο `View`, το group view που βρίσκεται στη κορυφή της δομής που περιγράφεται.

Μέσω αυτού του αντικειμένου μπορούμε να αναφερθούμε στα components της διεπαφής, με τη χρήση Java αντικειμένων, έναντι της περίπτωσης του activity όπου στην ουσία αυτό γίνεται άμεσα εφόσον έχει κληθεί η μέθοδος `setContentView`. Όσον αφορά το fragment, η μέθοδος στην οποία πραγματοποιείται το παραπάνω, είναι η `onCreateView`[104].

Η μέθοδος `onStart`[105], καλείται όταν το fragment γίνεται ορατό στον χρήστη. Σε αυτή τη μέθοδο υλοποιείται η λογική του `WordFragment`. Αρχικά καλείται η μέθοδος `getView`[106], η οποία επιστρέφει το view group της διεπαφής του fragment που δημιουργήθηκε στη `onCreateView`, σε μορφή αντικειμένου Java. Η μεταβλητή `levelValue`, αποθηκεύει το επίπεδο γνώσης των λημμάτων του λεξικού. Η τιμή αυτή προέρχεται από το `WordListActivity` κι όπως θα δούμε θα τη χρησιμοποιήσουμε ιδιαίτερα, στην ανάπτυξη του navigation menu της εφαρμογής για να επιλέγουμε την εκάστοτε λίστα λημμάτων μέσω αυτής. Για την ώρα ας θεωρήσουμε ότι η μεταβλητή αυτή περιέχει τη τιμή `"Level 1"`. Η μεταβλητή δίνεται ως όρισμα στη μέθοδο `wordRetrieveQuery`, η οποία πραγματοποιεί ένα SQL ερώτημα ώστε να λάβει όλα τα λήμματα που αφορούν τη τιμή που αναφέρθηκε. Σε αυτό το σημείο δημιουργείται ένα αντικείμενο `Cursor`, το οποίο εκμεταλλεύεται ο `WordCursorAdapter` για να δημιουργήσει τη λίστα με τα λήμματα και στη συνέχεια αυτός εφαρμόζεται στο list view της διεπαφής του fragment. Έτσι τελικά δημιουργείται μια τελική λίστα με βάση το επίπεδο

γνώσης. Ας πάμε να δούμε το τρόπο με τον οποίο πραγματοποιείται ένα ερώτημα στη βάση με χρήση του cursor.

Ένα ερώτημα σε μια βάση δεδομένων είναι συνήθως μια εντολή όπως η εξής:

```
SELECT * FROM TABLE_NAME;
```

Ένα ερώτημα μέσω του SQLite στο Android έχει ουσιαστικά μια μορφή όπως τη παραπάνω με τη διαφορά ότι η σύνταξη του, γίνεται με έναν διαφορετικό τρόπο από αυτόν που είδαμε για παράδειγμα στη δημιουργία ενός πίνακα. Το πρώτο βήμα στη δημιουργία ερωτήματος σε μια βάση δεδομένων είναι η δήλωση του helper που θα χρησιμοποιηθεί. Ο helper έχει το ρόλο διαχείρισής της βάσης όπως τη δημιουργία της, την αναβάθμιση της κτλ. Με άλλα λόγια στη περίπτωση του Quick Speak θα χρησιμοποιηθεί ο `WordDbHelper` που αναπτύξαμε για τη βάση δεδομένων. Έπειτα καλείται η μέθοδος `getReadableDatabase`[107], η οποία δηλώνει ότι επιχειρούμε να ανακτήσουμε δεδομένα από τη βάση, επιστρέφοντας έτσι ένα αντικείμενο `SQLiteDatabase`, το οποίο αντιπροσωπεύει τη βάση ως αναγνώσιμη (read-only).

Έχοντας πραγματοποιήσει το παραπάνω, μπορούμε να δημιουργήσουμε το ερώτημα με το οποίο θα ανακτήσουμε τα λήμματα. Το ερώτημα γίνεται μέσω της κλήσης `query`[108] του αντικειμένου `SQLiteDatabase`. Κάθε όρισμα αυτής της μέθοδού, ορίζει τη τελική μορφή του ερωτήματος (τα keywords του SQL query δηλαδή). Όπως θα παρατηρήσει κανείς από το παραπάνω κώδικα, εκμεταλλευόμαστε κι εδώ, τις σταθερές που δημιουργήσαμε στην εσωτερική κλάση `WordEntry` της κλάσης `WordContract`. Αρχικά ορίζεται το όνομα του πίνακα από τον οποίο θα αντλήσουμε τα δεδομένα. Στη συνέχεια, ορίζεται από ποια από τα columns του πίνακα θέλουμε δεδομένα, συμπεριλαμβανομένου και του μοναδικού id (πληροφορία _ID) κάθε εγγραφής των αποτελεσμάτων. Έπειτα ορίζεται η συνθήκη με την οποία θα επιστραφούν τα αποτελέσματα των δεδομένων που μας ενδιαφέρουν. Αυτό ισοδυναμεί με την αντίστοιχη χρήση της εντολής `WHERE` *ΣΥΝΘΗΚΗ*. Το WHERE κομμάτι ορίζεται ως εξής. Στο πρώτο όρισμα μέτα το όρισμα των columns, ορίζουμε το column που θα χρησιμοποιηθεί στην συνθήκη, και το ενώνουμε (string concatenation[109]) με τη τιμή «=?». Έπειτα ορίζουμε τη τιμή της συνθήκης, όπου στη περίπτωση μας, ισοδυναμεί με τη τιμή της μεταβλητής `levelValue` που αναφέραμε προηγουμένως. Τα υπόλοιπα ορίσματα της `query` αφορούν επιπλέον εντολές ενός SQL ερωτήματος όπως `GROUP BY`, `ORDER BY` κτλ. Μιας και δε μας απασχολεί η χρήση τους εδώ αφήνουμε τις προκαθορισμένες τιμές των ορισμάτων αυτών ως null. Έτσι με αυτό το τρόπο, το τελικό ερώτημα αν υποθέσουμε ότι η τιμή της levelValue είναι `"Level 1"`, είναι το εξής:

```
SELECT _ID, level, english_word, translated_word,  word_sound_id FROM english
WHERE level='Level 1';
```

Πραγματοποιώντας το παραπάνω και αποθηκεύοντας το στο αντικείμενο cursor, δημιουργούμε στη συνέχεια το αντικείμενο το οποίο περιέχει το list view της διεπαφής του fragment και ένα αντικείμενο `WordListAdapter`, ορίζοντας το ως adapter του list view. Έτσι η λίστα με τα λήμματα είναι πλέον ολοκληρωμένη και ορατή στο χρήστη. Να σημειωθεί ότι κάθε ερώτημα σε μια βάση, υλοποιείται σε ένα μπλοκ try-catch[110] ώστε να αντιμετωπιστεί το σφάλμα `SQLException`[111] σε περίπτωση που δεν είναι διαθέσιμη η βάση. Η αντιμετώπιση

του σφάλματος γίνεται με την εμφάνιση ενός μηνύματος `Toast`[112][113] στο χρήστη με τιμή `"Database unavailable"`.

Τέλος, έτσι ώστε να μην «σπαταλούνται» πόροι από την εφαρμογή, χρειάζεται να απελευθερωθούν τα αντικείμενα `cursor` και `db`. Αυτό πραγματοποιείται με τις κλήση της `close` και στα 2 αντίστοιχα. Το σημείο στο οποίο πραγματοποιείται η υλοποίηση της λογικής αυτής, είναι στη μέθοδο `onDestroy` του fragment, η οποία καλείται όταν αυτό «καταστραφεί» εντελώς από το σύστημα. Και με αυτό ολοκληρώνεται η υλοποίηση του `WordFragment`.

5.3 Η Υλοποίηση του Navigation Drawer

Το τελευταίο κομμάτι της υλοποίησης του Quick Speak, είναι η ανάπτυξη ενός side menu, το οποίο θα υλοποιηθεί με τη χρήση αυτού που ονομάζεται στο Android, ως navigation drawer. Ένα side menu, αποτελεί σημαντικό ρόλο στην ύπαρξη μιας εφαρμογής. Σκοπός τους είναι συνήθως να διευκολύνουν τη πλοήγηση του χρήστη στα κύρια μέρη της εφαρμογής. Τα menus αυτά συνήθως ενεργοποιούνται από το αριστερό μέρος της οθόνης (είτε με swipe left είτε με το ειδικό κουμπί ☰). Ένα τέτοιο menu, αποτελείται από 2 κυρίως μέρη. Το πρώτο μέρος, είναι μια επικεφαλίδα (header) η οποία συνήθως φέρει το όνομα της εφαρμογής και μια εικόνα ή γενικά δείχνει το σκοπό του εκάστοτε μενού στο χρήστη. Για παράδειγμα στη περίπτωση του Quick Speak, η επικεφαλίδα θα περιέχει απλά το όνομα της εφαρμογής. Το δεύτερο μέρος του menu, είναι οι κατηγορίες που καταλήγουν στα κύρια μέρη της εφαρμογής. Για παράδειγμα σε μια εφαρμογή email οι κατηγορίες ενός side menu, θα μπορούσαν να είναι οι κατηγορίες μηνυμάτων όπως κοινωνικά, σημαντικά, προσφορές κτλ. Στο Quick Speak, οι κατηγορίες θα είναι τα επίπεδα γνώσης των λημμάτων.

Ας πάμε αρχικά να δούμε την υλοποίηση της επικεφαλίδας του menu. Το αρχείο διάταξης του είναι το `navigation_menu_header.xml` .

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="40dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="start|bottom"
        android:layout_marginLeft="15dp"
        android:fontFamily="@font/open_sans_light"
        android:text="@string/app_name"
        android:textAppearance="@style/TextAppearance.AppCompat.Body1"
        android:textSize="20sp"
        android:textStyle="bold" />

</FrameLayout>
```

Το group view, που χρησιμοποιείται, είναι το `FrameLayout`[114]. Πρόκειται για ένα group view, το οποίο τοποθετεί συστατικά το ένα πάνω στο άλλο. Συνηθίζεται να χρησιμοποιείται στα menu

headers, έτσι ώστε να μπορεί κάποιος να τοποθετήσει για παράδειγμα, μια εικόνα ως παρασκήνιο το τίτλου της εφαρμογής ή του menu. Παρόλα αυτά, δεν υπάρχουν περιορισμοί στο τρόπο με τον οποίο θα υλοποιηθεί ένα menu header, μιας και το μόνο που γίνεται, είναι η ανάπτυξη ενός τυπικού XML αρχείου διάταξης.

Το menu header του Quick Speak, αποτελείται από ένα `FrameLayout`, το οποίο περιέχει ένα `TextView`, που έχει το ρόλο του τίτλου της εφαρμογής. Οι ιδιότητες που χρησιμοποιούνται, εφαρμόζουν το στυλ με το οποίο θα εμφανιστεί ο τίτλος. Το επόμενο βήμα, είναι η ανάπτυξη του αρχείου διάταξης του menu.

Το κυρίως μέρος του menu, παρουσιάζεται παρακάτω. Το αρχείο που χρησιμοποιείται είναι το `navigation_drawer_menu.xml`, το οποίο αποθηκεύεται σε ένα ειδικό φάκελο, τον `menu` και βρίσκεται κάτω από το φάκελο `res`. Με αυτό το τρόπο το σύστημα του Android γνωρίζει ότι πρόκειται για αρχείο διάταξης ενός menu.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:title="@string/english">
        <menu>
            <group android:checkableBehavior="single">
                <item
                    android:id="@+id/level_1"
                    android:title="@string/level_1" />

                <item
                    android:id="@+id/level_2"
                    android:title="@string/level_2" />

                <item
                    android:id="@+id/level_3"
                    android:title="@string/level_3" />

                <item
                    android:id="@+id/level_4"
                    android:title="@string/level_4" />

                <item
                    android:id="@+id/level_5"
                    android:title="@string/level_5" />

                <item
                    android:id="@+id/level_6"
                    android:title="@string/level_6" />

                <item
                    android:id="@+id/level_7"
                    android:title="@string/level_7" />

                <item
                    android:id="@+id/level_8"
                    android:title="@string/level_8" />

                <item
                    android:id="@+id/level_9"
                    android:title="@string/level_9" />

                <item
                    android:id="@+id/level_10"
```

```
                android:title="@string/level_10" />

            <item
                android:id="@+id/level_11"
                android:title="@string/level_11" />

            <item
                android:id="@+id/level_12"
                android:title="@string/level_12" />
        </group>
    </menu>
</item>
</menu>
```

Παρατηρούμε ότι η δομή που περιγράφεται στο αρχείο αυτό, είναι διαφορετική από αυτή την οποία περιγράφαμε με τα υπόλοιπα αρχεία διάταξης. Ένα menu, περιγράφεται κι αυτό με τη χρήση της XML ως εξής. Αρχικά χρησιμοποιείται το tag `menu`, στο οποίο ότι τοποθετείται μέσα, αφορά τα περιεχόμενα του menu. Ένα menu μπορεί να αποτελείται είτε από μεμονωμένα στοιχεία, που ονομάζονται items, είτε και από ένα εμφωλευμένο menu. Επίσης τα items μπορούν να ομαδοποιηθούν ώστε να τα χειριστούμε ως μια κοινή ομάδα (group). Όπως βλέπουμε και παραπάνω, τα items, περιγράφονται από το `item` tag, και τα groups από το `group` tag.

Το menu της εφαρμογής, περιγράφεται ως εξής. Το πρώτο item του, είναι στην ουσία ένα άλλο menu (δηλαδή εμφωλευμένο) και χρησιμοποιεί την ιδιότητα `android:title`, για να ορίσει το τίτλο της γλώσσας που χρησιμοποιείται για τα λήμματα (Αγγλικά). Μέσα στο menu αυτό, ομαδοποιούμε τα items με τη χρήση του `group` tag. Το tag αυτό, χρησιμοποιεί την ιδιότητα `android:checkableBehavior="single"`, για να δηλώσει ότι μπορεί να επιλεχθεί μόνο ένα item από το menu κάθε φορά.

Τα items του group, είναι φυσικά τα επίπεδα γνώσης των λιστών λημμάτων του λεξικού. Για κάθε ένα από αυτά, ορίζονται 2 ιδιότητες, το id μέσω του οποίου μπορούμε να χρησιμοποιήσουμε το item στο κώδικα του activity στο οποίο θα τοποθετηθεί το menu, και ο τίτλος του item, τον οποίο αντλούμε από το αρχείο `strings.xml` . Έτσι ολοκληρώνεται η ανάπτυξη της διάταξης του κυρίως μέρος του menu.

5.3.1 Τοποθέτηση του Menu στο WordListActivity

Ας πάμε να δούμε το τρόπο με τον οποίο τοποθετείται το menu που υλοποιήθηκε στο `WordListActivity`. Όπως είχαμε δει στη δομή διάταξης του activity, το group view που χρησιμοποιήθηκε ήταν το `DrawerLayout`. Μέσα σε αυτό θα τοποθετήσουμε τον drawer, ο οποίος είναι στην ουσία κάτι σαν ένα «συρτάρι» που ανοίγει ή κλείνει από το αριστερό πλαϊνό μέρος της οθόνης.

Το component που χρησιμοποιείται για να παρουσιάσει το menu, είναι το `NavigationView`[115]. Τοποθετώντας με λίγα λόγια το view αυτό σε ένα `DrawerLayout`, στην ουσία έχουμε τοποθετήσει το menu στο activity. Κάτω από το `LinearLayout` που βρίσκεται στο αρχείο activity_word_list.xml, τοποθετούμε τον εξής κώδικα:

```
<android.support.design.widget.NavigationView
    android:id="@+id/navigation_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:headerLayout="@layout/navigation_menu_header"
    app:menu="@menu/navigation_drawer_menu"/>
```

Όπως βλέπουμε, το view αυτό συμπεριλαμβάνει τα 2 αρχεία που δημιουργήθηκαν προηγουμένως με τις κατάλληλες ιδιότητες `app:headerLayout` και `app:menu`. Η ιδιότητα `android:fitsSystemWindows`, ορίζει ότι το view δε θα καλύψει τη μπάρα του συστήματος Android της συσκευής που θα φιλοξενεί την εφαρμογή.

5.3.2 Υλοποίηση της Λογικής του Navigation Drawer

Στο σημείο αυτό, θα πάμε να δούμε το τρόπο με τον οποίο, υλοποιούμε τη λογική χρήσης του side menu στην εφαρμογή. Στην ουσία θα υλοποιήσουμε τη συμπεριφορά, τόσο του του menu όσο και της εφαρμογής, όταν ο χρήστης επιλέγει ένα αντικείμενο του (item) Όπως είδαμε, το `WordListActivity`, αποτελείται από ένα fragment το οποίο με τη σειρά του παρουσιάζει μια λίστα λημμάτων βάσει ενός επιπέδου γνώσης. Το side menu, είναι αυτό που περιέχει τη λίστα με όλα τα διαθέσιμα επίπεδα γνώσης. Στόχος εδώ είναι να παρουσιάζεται η κατάλληλη λίστα λημμάτων ανάλογα με το ποιο αντικείμενο επιλέχθηκε από τη λίστα επιπέδων.

Παρακάτω παρουσιάζεται ο κώδικας που υλοποιεί τη τοποθέτηση του menu.

```
drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
actionBarDrawerToggle = new ActionBarDrawerToggle(this,
drawerLayout, toolbar, R.string.nav_open_drawer,
R.string.nav_close_drawer);

drawerLayout.addDrawerListener(actionBarDrawerToggle);
actionBarDrawerToggle.syncState();
```

Αρχικά κρατάμε το τίτλο κάθε επιπέδου σε μια μεταβλητή, ορίζοντας ως αρχική τιμή τη `"Level 1"`.

```
private CharSequence itemTitle = "Level 1";
```

Η πληροφορία αυτή, καθορίζει το επίπεδο που επιλέγεται κάθε φορά, όπως επίσης και τη τιμή της συνθήκης του ερωτήματος στη βάση (`WHERE level='Level 1'` για παράδειγμα). Αρχικά τοποθετούμε το menu, μαζί με το εικονίδιο ☰, το οποίο θα ενεργοποιεί και θα απενεργοποιεί το menu (σε στυλ toggle). Η λογική αυτή έχει ως εξής. Αναφερόμαστε στο drawer layout που δημιουργήσαμε. Στη συνέχεια χρησιμοποιούμε ένα αντικείμενο `ActionBarDrawerToggle`[116], το οποίο αφορά την ενέργεια του toggle που προαναφέρθηκε. Αυτό δέχεται ως ορίσματα το αντικείμενο του activity στο οποίο θα ανήκει, το drawer layout στο οποίο θα τοποθετηθεί, το αντικείμενο toolbar του οποίου θα αποτελεί μέρος, και τέλος 2 strings, τα οποία αφορούν τις καταστάσεις open και close του toggle. Οι τιμές τους τοποθετούνται στο αρχείο strings.xml και έχουν ως εξής.

```
<string name="nav_open_drawer">Open navigation drawer</string>
<string name="nav_close_drawer">Close navigation drawer</string>
```

Στη συνέχεια, ορίζουμε στο drawer layout, τον listener που θα εκτελείται για το toggle. Ο listener δέχεται ως όρισμα το αντικείμενο `actionBarDrawerToggle`. Τέλος καλούμε τη μέθοδο `syncState`, ώστε η κατάσταση του εικονιδίου να συγχρονίζεται με τη κατάσταση του menu (ανοιχτό κλειστό δηλαδή).

Έχοντας υλοποιήσει τη λογική του drawer layout, χρειάζεται να υλοποιήσουμε τη λογική που θα ακολουθεί η εφαρμογή, όταν ο χρήστης πατήσει ένα item από το menu. Αυτό πραγματοποιείται, θέτοντας στο navigation view που δημιουργήθηκε προηγουμένως, έναν ειδικό listener, τον `onNavigationItemSelectedListener`[117]. Στη κλάση `WordListActivity`, δηλώνουμε ότι θα χρησιμοποιήσουμε τον listener αυτόν με τον εξής επιπλέον κώδικα στην υπογραφή της (signature).

```
implements NavigationView.OnNavigationItemSelectedListener
```

Η μέθοδος που θα υλοποιούμε είναι η `onNavigationItemSelected`[117]. Πριν δούμε την υλοποίηση της μεθόδου θα δούμε το τρόπο με τον οποίο, τοποθετούμε ένα fragment προγραμματιστικά σε ένα activity, και συγκεκριμένα θα δούμε το τρόπο με τον οποίο εναλλάσσουμε fragments σε ένα activity.

Για την ώρα ας θεωρήσουμε ότι έχουμε υλοποιήσει τη λογική του `OnNavigationItemSelectedListener`. Το activity χρειάζεται σε πρώτο στάδιο, να εκκινήσει και να τοποθετήσει τη πρώτη λίστα λημμάτων του λεξικού (τα λήμματα του Level 1 δηλαδή). Πριν γίνει αυτό χρειάζεται να τοποθετήσουμε στο navigation view τον listener και να δηλώσουμε ότι το αντικείμενο που πατιέται κάθε φορά στο menu, θα είναι μαρκαρισμένο. Ο κώδικας που υλοποιεί το παραπάνω είναι.

```
navigationView = (NavigationView) findViewById(R.id.navigation_view);
navigationView.setNavigationItemSelectedListener(this);
navigationView.setCheckedItem(R.id.level_1);
```

Ο λόγος που δεν έχουμε παρουσιάσει ακόμα την υλοποίηση του listener, έχει να κάνει με τη κατανόηση του κώδικα που υλοποιείται στη συνέχεια του παραπάνω κομματιού. Αυτό που θέλουμε να κάνουμε, είναι να περάσουμε τη πληροφορία του τίτλου του επιπέδου λημμάτων (`itemTitle`) στο fragment, το οποίο θα το χρησιμοποιήσει για να πραγματοποιήσει το κατάλληλο ερώτημα στη βάση και να επιστρέψει τη αντίστοιχη λίστα λημμάτων. Έχοντας κάνει αυτό, θα τοποθετήσουμε με κώδικα Java, το fragment στο `WordListActivity`. Όπως είχαμε τονίσει, ένα fragment δεν είναι το ίδιο με ένα activity. Τα activities μπορούν να μεταβιβάσουν πληροφορία μεταξύ τους μέσω των intents. Πώς όμως ένα activity μπορεί να μεταβιβάσει πληροφορία σε ένα fragment;

Η απάντηση είναι, μέσω των ορισμάτων fragment (fragment arguments) . Ουσιαστικά η λογική που υλοποιείται εδώ, είναι η αποθήκευση της πληροφορίας που θέλουμε να στείλουμε, σε ένα αντικείμενο `Bundle`[118], και το πέρασμα του ως όρισμα στο fragment, με τη κλήση της μεθόδου `setArguments`[119]. Το αντικείμενο `Bundle`, έχει μεθόδους για κάθε τύπο δεδομένου θέλουμε να μεταβιβάσουμε. Έτσι στη περίπτωση του Quick Speak, η πληροφορία που θέλουμε να μεταβιβάσουμε, είναι τύπου `CharSequence`[120], οπότε θα χρησιμοποιήσουμε τη μέθοδο `putCharSequence`[121]. Η υλοποίηση του παραπάνω παρουσιάζεται με τον εξής κώδικα.

```
navigationView = (NavigationView) findViewById(R.id.navigation_view);
navigationView.setNavigationItemSelectedListener(this);
navigationView.setCheckedItem(R.id.level_1);

arguments = new Bundle();
arguments.putCharSequence("itemTitle", itemTitle);
```

Αρχικά χρειαζόμαστε το 1ο επίπεδο, οπότε χρησιμοποιούμε τη μέθοδο `setCheckedItem`[122] με όρισμα το id του level 1 από το item του navigation menu. Αυτό έχει ως αποτέλεσμα στο menu, να είναι μαρκαρισμένο το 1ο item του, δηλαδή το Level 1.

Ας πάμε να δούμε το κώδικα με τον οποίο δημιουργείται και τοποθετείται στο activity ένα fragment, και συγκεκριμένα τη τοποθέτηση του `WordFragment` στο `WordListActivity`.

```
wordFragment = new WordFragment();
wordFragment.setArguments(arguments);

fragmentTransaction = getSupportFragmentManager().beginTransaction();
fragmentTransaction.add(R.id.frag_container, wordFragment);
fragmentTransaction.commit();
```

Η μέθοδος `setArguments` ορίζει το αντικείμενο `Bundle` που δημιουργήσαμε πριν. Η τοποθέτηση ενός fragment σε ένα activity προγραμματιστικά, πραγματοποιείται με τη χρήση ενός αντικειμένου `FragmentTransaction`[123]. Τα fragments, έχουν τη δυνατότητα να προσαφαιρούνται στο activity που τα φιλοξενεί. Κάθε τέτοια προσθαφαίρεση, είναι στην ουσία αυτό που ονομάζεται συναλλαγή (transaction). Στο παραπάνω κώδικα, καλούμε αρχικά τον fragment manager που διαχειρίζεται τα transactions και ξεκινάμε ένα transaction. Αυτό γίνεται με τη αλυσιδωτή χρήση των μεθόδων `getSupportFragmentManager`[124] (έναντι της `getFragmentManager` επειδή χρησιμοποιούμε τη support library) και `beginTransaction`[125]. Στη συνέχεια προσθέτουμε στο container του αρχείου διάταξης του `WordListActivity` (`activity_word_list.xml`) το fragment που δημιουργήσαμε. Έπειτα ολοκληρώνουμε το transaction με τη χρήση της μεθόδου `commit`[126]. Με αυτόν το τρόπο δημιουργούμε το πρώτο fragment που θα περιέχει τη λίστα με τα λήμματα του πρώτου επιπέδου.

Ας πάμε να δούμε την υλοποίηση της μεθόδου `onNavigationItemSelected`, η οποία εκτελεί κώδικα, όταν ο χρήστης πατήσει ένα item του menu.

```
    public boolean onNavigationItemSelected(MenuItem item) {
        itemTitle = item.getTitle();

        getSupportActionBar().setTitle(itemTitle);

        drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);

        item.setChecked(true);
        drawerLayout.closeDrawers();

        arguments = new Bundle();
        arguments.putCharSequence("itemTitle", itemTitle);

        wordFragment = new WordFragment();
        wordFragment.setArguments(arguments);

        fragmentTransaction = getSupportFragmentManager().beginTransac-
tion();
```

```
        fragmentTransaction.replace(R.id.frag_container, wordFragment);
        fragmentTransaction.commit();

        return true;
    }
```

Το όρισμα της μεθόδου, αφορά το item που πατιέται από το χρήστη. Η λογική του παραπάνω κώδικα έχει ως εξής. Ο χρήστης επιλέγει ένα item. Από αυτό αποθηκεύουμε το τίτλο του στη μεταβλητή `itemTitle`, ορίζοντας τον κι ως τίτλο του toolbar. Μαρκάροντας το item που επιλέχθηκε, το menu κλείνει (βλέπε μέθοδο `closeDrawers`[127]). Έπειτα, πραγματοποιούμε πάλι ένα fragment transaction, με τη διαφορά ότι αντί να προσθέσουμε ένα νέο fragment, αντικαθιστούμε το ήδη υπάρχον με αυτό. Γι' αυτό το σκοπό, χρησιμοποιούμε τη μέθοδο `replace`[128]. Η επιστροφή της τιμής true από τη μέθοδο, δηλώνει ότι το item που πατήθηκε, θεωρείται επιλεγμένο.

Σε πολλές εφαρμογές, είναι συνηθισμένο να ορίζουμε κάποια λειτουργία στο activity όταν ο χρήστης πατήσει το κουμπί επιστροφής της συσκευής. Αυτό γίνεται με τη χρήση της μεθόδου `onBackPressed`[129]. Στη περίπτωση του Quick Speak, θα υλοποιήσουμε τη λογική με την οποία αν έχουμε το menu ανοιχτό στην εφαρμογή, η χρήση του κουμπιού θα το κλείνει. Η υλοποίηση αυτή παρουσιάζεται παρακάτω.

```
    public void onBackPressed() {
        drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);

        if(drawerLayout.isDrawerOpen(GravityCompat.START)) {
            drawerLayout.closeDrawers();
        } else {
            super.onBackPressed();
        }
    }
```

Στην ουσία ελέγχουμε με τη μέθοδο `isDrawerOpen`, εάν το menu είναι ήδη ανοιχτό (το όρισμα `GravityCompat.START` συνεπάγεται με το γεγονός ότι το menu ανοίγει από την αρχή της οθόνης αριστερά.). Αν ναι τότε το κλείνουμε διαφορετικά καλούμε τη αντίστοιχη μέθοδο της κλάσης `Activity`, χωρίς να γίνει κάποια ενέργεια στο προσκήνιο.

Στο σημείο αυτό, θα εστιάσουμε σε μια λεπτομέρεια των activities, που αφορά την λογική συμπεριφορά τους. Μέχρι στιγμής η ολική υλοποίηση του Quick Speak, προϋποθέτει ότι ο χρήστης το εκτελεί, σε κατάσταση πορτρέτου (portrait mode). Τι συμβαίνει όμως όταν ο χρήστης αλλάξει το προσανατολισμό της εφαρμογής; Η απάντηση είναι ότι το activity διακόπτει τη λειτουργία του (συγκεκριμένα καλείται η μέθοδος `OnDestroy`) και επαναεκκινείται. Αυτό έχει ως αποτέλεσμα, ότι δεδομένα χρησιμοποιούνται προσωρινά (η μεταβλητή `itemTitle` για παράδειγμα) να καταστρέφονται και να αναδημιουργούνται στην επόμενη εκκίνηση του activity. (κλήση της `onCreate`). Το πρόβλημα στη περίπτωση του Quick Speak, είναι το γεγονός ότι η επιλεγμένη λίστα λημμάτων με βάση της παραπάνω λογικής του menu, δε παραμένει κατά την αλλαγή του προσανατολισμού της εφαρμογής (η την αλλαγή του configuration όπως προαναφέρθηκε σε προηγούμενο κεφάλαιο) και έτσι επιστρέφει πάντα σε κάθε τέτοια αλλαγή, στη κατάσταση κατά την οποία δημιουργεί και προβάλει τη πρώτη λίστα. Σε τέτοιες περιπτώσεις όπως αυτή, μπορούμε να εφαρμόσουμε μια τεχνική κατά την οποία, το activity μπορεί να διατηρεί τη κατάσταση του ακόμα κι αν αλλάξει ο προσανατολισμός της εφαρμογής.

Ας δούμε την υλοποίηση της λογικής αυτής της τεχνικής. Όταν θέλουμε να αποθηκεύσουμε πληροφορία ώστε να διατηρηθεί η κατάσταση ενός activity, σε περίπτωση που για παράδειγμα αυτό χρειαστεί να τερματιστεί για κάποιο λόγο (όπως αυτόν που αναφέρθηκε παραπάνω), χρησιμοποιούμε μια ειδική μέθοδο για αυτό το σκοπό, την `onSaveInstanceState`[130]. Η μέθοδος αυτή, χρησιμοποιεί ένα `Bundle` αντικείμενο, όπως αυτό που είδαμε στη περίπτωση του `WordFragment`, μέσω του οποίου αποθηκεύει και ανακτά τη πληροφορία που χρειάζεται για να διατηρήσει τη κατάσταση του activity. Η κλήση της μεθόδου αυτής, πραγματοποιείται λίγο πριν καταστραφεί το activity. Το αντικείμενο `Bundle`, όντας όρισμα της μεθόδου `onSaveInstanceState`, είναι επίσης και όρισμα της μεθόδου `onCreate`, όπως θα είχε παρατηρήσει ο αναγνώστης κατά την ανάγνωση της παρούσας πτυχιακής. Στην ουσία το όρισμα αυτό της `onCreate`, μπορούμε να το εκμεταλλευτούμε και να ανακτήσουμε τη πληροφορία που αποθηκεύτηκε με την `onSaveInstanceState`, ώστε να διατηρηθεί η κατάσταση του activity που την αφορά. Στη περίπτωση του Quick Speak, μας ενδιαφέρει να διατηρούμε τη τιμή της μεταβλητής `itemTitle`, μέσω της οποίας δίνουμε τίτλο στο toolbar της εφαρμογής, ανάλογα με το επίπεδο λημμάτων που έχει επιλεχθεί, και ιδιαίτερα, διατηρούμε επίσης τη κατάσταση με την οποία έχει δημιουργηθεί η αντίστοιχη λίστα λημμάτων, βάσει αυτής τη τιμής. Ο κώδικας λοιπόν με τον οποίο, υλοποιείται η κατάσταση αυτή του Quick Speak είναι:

```
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    outState.putCharSequence("itemTitle", itemTitle);
}


if (savedInstanceState != null) {
    itemTitle = savedInstanceState.getCharSequence("itemTitle");
}
```

Όπως βλέπουμε κι από την υλοποίηση της `onSaveInstanceState`, χειριζόμαστε το αντικείμενο `Bundle`, με τον ίδιο τρόπο που πραγματοποιήσαμε και στη περίπτωση του `WordFragment`. Για να ανακτήσουμε τη πληροφορία από το `Bundle`, ελέγχουμε αρχικά αν αυτό υπάρχει ήδη (δηλαδή αν όντως κάτι είναι αποθηκευμένο στο αντικείμενο `Bundle` διαφορετικά είναι `null`) και αν ναι, τότε ανακτούμε και αποθηκεύουμε τη πληροφορία και πάλι στη μεταβλητή `itemTitle`. Αυτό γίνεται με την αντίστοιχη `get` μέθοδο ανάλογα με το τύπο της πληροφορίας που μεταφέρεται (`CharSequence` εδώ). Το String `"itemTitle"`, έχει το ρόλο του id κατά κάποιο τρόπο. Εξασφαλίζει απλώς ότι μόνο η μεταβλητή `itemTitle` συσχετίζεται με αυτό. (φυσικά και θα μπορούσε να έχει άλλη ονομασία όπως `savedTitle` για παράδειγμα). Συνήθως ο παραπάνω έλεγχος υλοποιείται αμέσως μετά τη κλήση της μεθόδου `setContentView`.

Τελειώνοντας την υλοποίηση τόσο του WordListActivity όσο και του Quick Speak, εφαρμόζουμε έναν έλεγχο στο σημείο όπου δημιουργείται η πρώτη λίστα του activity.

```
if (wordFragment == null) {
    wordFragment = new WordFragment();
    wordFragment.setArguments(arguments);

    fragmentTransaction =
    getSupportFragmentManager().beginTransaction();
```

```
            fragmentTransaction.add(R.id.frag_container, wordFragment);
            fragmentTransaction.commit();
        }
```

Ο λόγος που υλοποιήθηκε ο έλεγχος αυτός, είναι για να εξασφαλίσει ότι το παραπάνω fragment θα δημιουργηθεί και θα τοποθετηθεί στο activity εάν και μόνο εάν, δεν υπάρχει (δεν έχει διατηρηθεί η κατάσταση κάποιου υπάρχοντος δηλαδή) ήδη κάποιο fragment. Αυτή η ενέργεια επιλύνει ένα σφάλμα (bug) κατά το οποίο, τοποθετείται ένα επιπλέον fragment πάνω σε ένα ήδη υπάρχον προκαλώντας ένα είδους overlay (συγκεκριμένα μια λίστα πάνω σε μια άλλη) που δεν επιτρέπει την ομαλή χρήση της εφαρμογής.

5.4 Η Τελική Μορφή του Quick Speak

Τροποποιώντας τα αρχεία του project του Quick Speak, όπου χρειάζεται, θα παρουσιάσουμε όλα τα αρχεία που χρησιμοποιήθηκαν για την υλοποίηση του, καθώς επίσης και το τελικό UML class diagram του project. Το τελικό project, θα είναι διαθέσιμο στο συνοδευτικό υλικό της παρούσας πτυχιακής.

5.4.1 Παρουσίαση Υλοποιημένων Αρχείων

Στο σημείο αυτό, παρουσιάζονται τα αρχεία κλάσεων, τα XML αρχεία διάταξης καθώς κι όλα τα σημαντικά κυρίως αρχεία που χρησιμοποιήθηκαν για την υλοποίηση του Quick Speak. Ο παρακάτω πίνακας, παρουσιάζει τις συνδέσεις μεταξύ των κλάσεων και των αρχείων διάταξης που χρησιμοποιήθηκαν.

Αρχείο Κλάσης	**Αρχεία Διάταξης που Χρησιμοποιεί**
MainActivity.java	activity_main.xml
WordContract.java	-
WordDbHelper.java	-
WordCursorAdapter.java	list_view.xml
WordFragment.java	fragment_word.xml
WordListActivity.java	app_toolbar.xml
	activity_word_list.xml
	navigation_menu_header.xml
	navigation_drawer_menu.xml
-	colors.xml
-	styles.xml
-	strings.xml
-	AndroidManifest.xml

Πίνακας 2: Απεικόνιση Αρχείων του Quick Speak Project

Παρακάτω παρουσιάζονται οι τελικές μορφές των αρχείων σε κώδικα, που εμφανίζονται στο πίνακα.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://sche-
mas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:contentDescription="welcome_icon"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.353"
        app:srcCompat="@mipmap/english_flag" />

    <TextView
        android:id="@+id/textView"
        style="@style/Theme.AppCompat.Light"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="4dp"
        android:layout_marginEnd="8dp"
        android:text="@string/app_name"
        android:textSize="25sp"
        android:textStyle="bold"
        app:fontFamily="@font/open_sans_light"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView" />

    <ProgressBar
        android:id="@+id/progress_bar"
        style="?android:attr/progressBarStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:indeterminate="true"
```

```
        android:max="100"
        android:progress="1"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        app:layout_constraintVertical_bias="0.064" />
</android.support.constraint.ConstraintLayout>
```

MainActivity.java

```
package com.selgo.quickspeak;

import android.content.Intent;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    public static int DURATION_TIME = 1500;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        new Handler().postDelayed(new Runnable(){
            @Override
            public void run() {
                Intent intent = new Intent(MainActivity.this, WordListAc-
tivity.class);
                startActivity(intent);
                finish();
            }
        }, DURATION_TIME);

    }
}
```

WordContract.java

```
package com.selgo.quickspeak.data;

import android.provider.BaseColumns;

public class WordContract {

    private WordContract() {}

    public static final class WordEntry implements BaseColumns {

        public final static String TABLE_NAME = "english";

        public final static String _ID = BaseColumns._ID;
        public final static String COLUMN_LEVEL = "level";
        public final static String COLUMN_ENGLISH_WORD = "english_word";
        public final static String COLUMN_TRANSLATED_WORD = "trans-
lated_word";
        public final static String COLUMN_WORD_SOUND_ID = "word_sound_id";

    }
}
```

WordDbHelper.java

```
package com.selgo.quickspeak.data;

import android.content.ContentValues;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import com.selgo.quickspeak.R;
import com.selgo.quickspeak.data.WordContract.WordEntry;

public class WordDbHelper extends SQLiteOpenHelper {

    private static final String DATABASE_NAME = "quick_speak.db";
    private static final int DATABASE_VERSION = 1;

    public WordDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String SQL_CREATE_ENGLISH_TABLE = "CREATE TABLE " + WordEntry.TA-
BLE_NAME + "("
                + WordEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + WordEntry.COLUMN_LEVEL + " TEXT NOT NULL, "
                + WordEntry.COLUMN_ENGLISH_WORD + " TEXT, "
                + WordEntry.COLUMN_TRANSLATED_WORD + " TEXT, "
                + WordEntry.COLUMN_WORD_SOUND_ID + " TEXT);";

        db.execSQL(SQL_CREATE_ENGLISH_TABLE);

        /* Level 1 Words */
        insertWord(db, "Level 1", "A a", "Το 1ο γράμμα του αγγλικού
αλφαβήτου", R.raw.a);
        insertWord(db, "Level 1", "B b", "Το 2ο γράμμα του αγγλικού αλφαβή-
του", R.raw.b);
        insertWord(db, "Level 1", "C c", "Το 3ο γράμμα του αγγλικού αλφαβή-
του", R.raw.c);
        insertWord(db, "Level 1", "D d", "Το 4ο γράμμα του αγγλικού αλφαβή-
του", R.raw.d);
        insertWord(db, "Level 1", "E e", "Το 5ο γράμμα του αγγλικού αλφαβή-
του", R.raw.e);
        insertWord(db, "Level 1", "F f", "Το 6ο γράμμα του αγγλικού αλφαβή-
του", R.raw.f);
        insertWord(db, "Level 1", "G g", "Το 7ο γράμμα του αγγλικού αλφαβή-
του", R.raw.g);
        insertWord(db, "Level 1", "H h", "Το 8ο γράμμα του αγγλικού αλφαβή-
του", R.raw.h);
        insertWord(db, "Level 1", "I i", "Το 9ο γράμμα του αγγλικού αλφαβή-
του", R.raw.i);
        insertWord(db, "Level 1", "J j", "Το 10ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.j);
        insertWord(db, "Level 1", "K k", "Το 11ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.k);
        insertWord(db, "Level 1", "L l", "Το 12ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.l);
        insertWord(db, "Level 1", "M m", "Το 13ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.m);
```

```
        insertWord(db, "Level 1", "N n", "Το 14ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.n);
        insertWord(db, "Level 1", "O o", "Το 15ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.o);
        insertWord(db, "Level 1", "P p", "Το 16ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.p);
        insertWord(db, "Level 1", "Q q", "Το 17ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.q);
        insertWord(db, "Level 1", "R r", "Το 18ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.r);
        insertWord(db, "Level 1", "S s", "Το 19ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.s);
        insertWord(db, "Level 1", "T t", "Το 20ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.t);
        insertWord(db, "Level 1", "U u", "Το 21ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.u);
        insertWord(db, "Level 1", "V v", "Το 22ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.v);
        insertWord(db, "Level 1", "W w", "Το 23ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.w);
        insertWord(db, "Level 1", "X x", "Το 24ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.x);
        insertWord(db, "Level 1", "Y y", "Το 25ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.y);
        insertWord(db, "Level 1", "Z z", "Το 26ο γράμμα του αγγλικού αλφα-
βήτου", R.raw.z);


        /* Level 2 Words */
        insertWord(db, "Level 2", "Hi", "Γειά σου", R.raw.hi);
        insertWord(db, "Level 2", "Let's go", "Πάμε", R.raw.let_s_go);
        insertWord(db, "Level 2", "please", "παρακαλώ", R.raw.please);
        insertWord(db, "Level 2", "good morning", "καλημέρα",
R.raw.good_morning);
        insertWord(db, "Level 2", "good night", "καληνύχτα",
R.raw.good_night);
        insertWord(db, "Level 2", "yes", "ναι", R.raw.yes);
        insertWord(db, "Level 2", "no", "όχι", R.raw.no);
        insertWord(db, "Level 2", "cheers", "Στην υγειά μας!",
R.raw.cheers);
        insertWord(db, "Level 2", "thank you", "ευχαριστώ",
R.raw.thank_you);
        insertWord(db, "Level 2", "you are welcome", "παρακαλώ (όταν
προσφέρω κάτι)", R.raw.you_re_welcome);
        insertWord(db, "Level 2", "I'm sorry", "Συγνώμη / Λυπάμαι",
R.raw.i_m_sorry);
        insertWord(db, "Level 2", "see you later", "τα λέμε μετά",
R.raw.see_you_later);
        insertWord(db, "Level 2", "goodbye", "αντίο", R.raw.goodbye);

        /* Level 3 Words */
        insertWord(db, "Level 3", "how", "πώς", R.raw.how);
        insertWord(db, "Level 3", "how are you?", "Πώς είσαι / Τι κάνεις",
R.raw.how_are_you);
        insertWord(db, "Level 3", "very", "πολύ", R.raw.very);
        insertWord(db, "Level 3", "very well", "πολύ καλά",
R.raw.very_well);
        insertWord(db, "Level 3", "I'm very well", "Είμαι πολύ καλά",
R.raw.i_m_very_well);
        insertWord(db, "Level 3", "I", "εγώ", R.raw.i);
        insertWord(db, "Level 3", "you", "εσύ / εσείς", R.raw.you);
```

```
        insertWord(db, "Level 3", "one", "ένας / μία / ένα", R.raw.one);
        insertWord(db, "Level 3", "call", "καλώ", R.raw.call);
        insertWord(db, "Level 3", "you are called", "καλείσαι / λέγεσαι",
R.raw.you_re_called);
        insertWord(db, "Level 3", "what's your name", "ποιό είναι το όνομά
σου; / πώς σε λένε;", R.raw.what_s_your_name);
        insertWord(db, "Level 3", "my name is ...", "Το όνομά μου είναι ...
/ Λέγομαι ...", R.raw.my_name_is);
        insertWord(db, "Level 3", "the genius", "ο ιδιοφυής", R.raw.the_ge-
nius);
        insertWord(db, "Level 3", "you're a genius", "Είσαι μια ιδιοφϋια!",
R.raw.you_re_a_genius);

        /* Level 4 Words */
        insertWord(db, "Level 4", "I am", "εγώ είμαι", R.raw.i_am);
        insertWord(db, "Level 4", "I'm", "εγώ είμαι (συντομογραφία)",
R.raw.i_m);
        insertWord(db, "Level 4", "you are", "εσύ / εσείς είσαι / είστε",
R.raw.you_are);
        insertWord(db, "Level 4", "you're", "εσύ / εσείς είσαι / είστε
(συντομογραφία)", R.raw.you_re);
        insertWord(db, "Level 4", "happy", "χαρούμενος/η", R.raw.happy);
        insertWord(db, "Level 4", "sad", "λυπημένος/η", R.raw.sad);
        insertWord(db, "Level 4", "angry", "θυμωμένος", R.raw.angry);
        insertWord(db, "Level 4", "to be wrong", "να έχει κάποιος άδικο",
R.raw.to_be_wrong);
        insertWord(db, "Level 4", "the reason", "ο λόγος", R.raw.the_rea-
son);
        insertWord(db, "Level 4", "tired", "κουρασμένος/η", R.raw.tired);
        insertWord(db, "Level 4", "sick", "άρρωστος/η", R.raw.sick);
        insertWord(db, "Level 4", "I have", "έχω", R.raw.i_have);
        insertWord(db, "Level 4", "you have", "έχεις", R.raw.you_have);
        insertWord(db, "Level 4", "the hunger", "η πείνα", R.raw.hunger);
        insertWord(db, "Level 4", "the thirst", "η δίψα", R.raw.thirst);
        insertWord(db, "Level 4", "are you hungry?", "πεινάς;",
R.raw.are_you_hungry);
        insertWord(db, "Level 4", "am i wrong?", "έχω άδικο;",
R.raw.am_i_wrong);
        insertWord(db, "Level 4", "you're right", "έχεις δίκιο",
R.raw.you_re_right);

        /* Level 5 Words */
        insertWord(db, "Level 5", "the apple", "το μήλο", R.raw.the_apple);
        insertWord(db, "Level 5", "the banana", "η μπανάνα", R.raw.the_ba-
nana);
        insertWord(db, "Level 5", "the beef", "το βοδινό", R.raw.the_beef);
        insertWord(db, "Level 5", "the beer", "η μπύρα", R.raw.the_beer);
        insertWord(db, "Level 5", "the bread", "το ψωμί", R.raw.the_bread);
        insertWord(db, "Level 5", "the butter", "το βούτυρο",
R.raw.the_butter);
        insertWord(db, "Level 5", "the cheese", "το τυρί",
R.raw.the_cheese);
        insertWord(db, "Level 5", "the chicken", "το κοτόπουλο",
R.raw.the_chicken);
        insertWord(db, "Level 5", "the coffee", "ο καφές", R.raw.the_cof-
fee);
        insertWord(db, "Level 5", "the egg", "το αυγό", R.raw.the_egg);
        insertWord(db, "Level 5", "the fish", "το ψάρι", R.raw.the_fish);
        insertWord(db, "Level 5", "the food", "το φαγητό", R.raw.the_food);
        insertWord(db, "Level 5", "the fruit", "το φρούτο",
R.raw.the_fruit);
```

```
        insertWord(db, "Level 5", "the juice", "ο χυμός", R.raw.the_juice);
        insertWord(db, "Level 5", "the lemon", "το λεμόνι",
R.raw.the_lemon);
        insertWord(db, "Level 5", "the meat", "το κρέας", R.raw.the_meat);
        insertWord(db, "Level 5", "the milk", "το γάλα", R.raw.the_milk);
        insertWord(db, "Level 5", "the orange", "το πορτοκάλι",
R.raw.the_orange);
        insertWord(db, "Level 5", "the pasta", "τα μακαρόνια",
R.raw.the_pasta);
        insertWord(db, "Level 5", "the pork", "το χοιρινό",
R.raw.the_pork);
        insertWord(db, "Level 5", "the potato", "η πατάτα", R.raw.the_po-
tato);
        insertWord(db, "Level 5", "the rice", "το ρύζι", R.raw.the_rice);
        insertWord(db, "Level 5", "the salad", "η σαλάτα",
R.raw.the_salad);
        insertWord(db, "Level 5", "the sauce", "η σάλτσα",
R.raw.the_sauce);
        insertWord(db, "Level 5", "the soup", "η σούπα", R.raw.the_soup);
        insertWord(db, "Level 5", "the tea", "το τσάι", R.raw.the_tea);
        insertWord(db, "Level 5", "the vegetables", "τα λαχανικά",
R.raw.the_vegetables);
        insertWord(db, "Level 5", "the water", "το νερό", R.raw.the_water);
        insertWord(db, "Level 5", "the wine", "το κρασί", R.raw.the_wine);

        /* Level 6 Words */
        insertWord(db, "Level 6", "and", "και", R.raw.and);
        insertWord(db, "Level 6", "awesome", "φοβερός/η", R.raw.awesome);
        insertWord(db, "Level 6", "delicious", "νόστιμος/η", R.raw.deli-
cious);
        insertWord(db, "Level 6", "disgusting", "αηδιαστικός/η", R.raw.dis-
gusting);
        insertWord(db, "Level 6", "he is", "αυτός είναι", R.raw.he_is);
        insertWord(db, "Level 6", "he's", "αυτός έιναι (συντομογραφία)",
R.raw.he_s);
        insertWord(db, "Level 6", "I don't like", "Δε μου αρέσει",
R.raw.i_don_t_like);
        insertWord(db, "Level 6", "I like", "Μου αρέσει", R.raw.i_like);
        insertWord(db, "Level 6", "it is", "αυτό είναι", R.raw.it_is);
        insertWord(db, "Level 6", "it's", "αυτό είναι (συντομογραφία)",
R.raw.it_s);
        insertWord(db, "Level 6", "not", "δεν", R.raw.not);
        insertWord(db, "Level 6", "she is", "αυτή είναι", R.raw.she_is);
        insertWord(db, "Level 6", "she's", "αυτή είναι (συντομογραφία)",
R.raw.she_s);
        insertWord(db, "Level 6", "they are", "αυτοί / αυτές / αυτά είναι",
R.raw.they_are);
        insertWord(db, "Level 6", "they're", "αυτοί / αυτές / αυτά είναι
(συντομογραφία)", R.raw.they_re);
        insertWord(db, "Level 6", "we are", "εμείς είμαστε", R.raw.we_are);
        insertWord(db, "Level 6", "we're", "εμείς είμαστε (συντομογραφία)",
R.raw.we_re);

        /* Level 7 Words */
        insertWord(db, "Level 7", "where", "που", R.raw.where);
        insertWord(db, "Level 7", "Where are you from?", "Απο που είσαι;",
R.raw.where_are_you_from);
        insertWord(db, "Level 7", "from", "από", R.raw.from);
        insertWord(db, "Level 7", "I'm from ...", "Είμαι από ...",
R.raw.i_m_from);
        insertWord(db, "Level 7", "Africa", "Αφρική", R.raw.africa);
```

```
        insertWord(db, "Level 7", "African", "Αφρικανός", R.raw.african);
        insertWord(db, "Level 7", "Albania", "Αλβανία", R.raw.albania);
        insertWord(db, "Level 7", "Albanian", "Αλβανός", R.raw.albanian);
        insertWord(db, "Level 7", "America", "Αμερική", R.raw.america);
        insertWord(db, "Level 7", "American", "Αμερικάνος", R.raw.ameri-
can);
        insertWord(db, "Level 7", "Asia", "Ασία", R.raw.asia);
        insertWord(db, "Level 7", "Asian", "Ασιάτης", R.raw.asian);
        insertWord(db, "Level 7", "Australia", "Αυστραλία", R.raw.aus-
tralia);
        insertWord(db, "Level 7", "Australian", "Αυστραλός", R.raw.austral-
ian);
        insertWord(db, "Level 7", "Austria", "Αυστρία", R.raw.austria);
        insertWord(db, "Level 7", "Austrian", "Αυστριακός", R.raw.aus-
trian);
        insertWord(db, "Level 7", "Belarus", "Λευκορωσία", R.raw.belarus);
        insertWord(db, "Level 7", "Belarusian", "Λευκορώσος", R.raw.bela-
rusian);
        insertWord(db, "Level 7", "Belgium", "Βέλγιο", R.raw.belgium);
        insertWord(db, "Level 7", "Belgian", "Βέλγος", R.raw.belgian);
        insertWord(db, "Level 7", "Bosnia", "Βοσνία", R.raw.bosnia);
        insertWord(db, "Level 7", "Bosnian", "Βόσνιος", R.raw.bosnian);
        insertWord(db, "Level 7", "Brazil", "Βραζιλία", R.raw.brazil);
        insertWord(db, "Level 7", "Brazilian", "Βραζιλιάνος", R.raw.brazil-
ian);
        insertWord(db, "Level 7", "Britain", "Βρετανία", R.raw.britain);
        insertWord(db, "Level 7", "British", "Βρετανός", R.raw.british);
        insertWord(db, "Level 7", "Bulgaria", "Βουλγαρία", R.raw.bulgaria);
        insertWord(db, "Level 7", "Bulgarian", "Βούλγαρος", R.raw.bulgar-
ian);
        insertWord(db, "Level 7", "China", "Κίνα", R.raw.china);
        insertWord(db, "Level 7", "Chinese", "Κινέζος", R.raw.china);
        insertWord(db, "Level 7", "Croatia", "Κροατία", R.raw.croatia);
        insertWord(db, "Level 7", "Croatian", "Κροάτης", R.raw.croatian);
        insertWord(db, "Level 7", "Cyprus", "Κύπρος", R.raw.cyprus);
        insertWord(db, "Level 7", "Cypriot", "Κύπριος", R.raw.cypriot);
        insertWord(db, "Level 7", "Czech Republic", "Τσέχικη Δημοκρατία
(Τσεχία)", R.raw.czech_republic);
        insertWord(db, "Level 7", "Czech", "Τσέχος", R.raw.czech);
        insertWord(db, "Level 7", "Denmark", "Δανία", R.raw.denmark);
        insertWord(db, "Level 7", "Danish", "Δανός", R.raw.danish);
        insertWord(db, "Level 7", "England", "Αγγλία", R.raw.england);
        insertWord(db, "Level 7", "English", "Άγγλος (Αγγλικά)", R.raw.eng-
lish);
        insertWord(db, "Level 7", "Estonia", "Εσθονία", R.raw.estonia);
        insertWord(db, "Level 7", "Estonian", "Εσθονός", R.raw.estonian);
        insertWord(db, "Level 7", "Europe", "Ευρώπη", R.raw.europe);
        insertWord(db, "Level 7", "European", "Ευρωπαίος", R.raw.european);
        insertWord(db, "Level 7", "Finland", "Φινλανδία", R.raw.finland);
        insertWord(db, "Level 7", "Finnish", "Φινλανδός", R.raw.finnish);
        insertWord(db, "Level 7", "France", "Γαλλία", R.raw.france);
        insertWord(db, "Level 7", "French", "Γάλλος", R.raw.french);
        insertWord(db, "Level 7", "Georgia", "Γεωργία", R.raw.georgia);
        insertWord(db, "Level 7", "Georgian", "Γεωργός", R.raw.georgian);
        insertWord(db, "Level 7", "Germany", "Γερμανία", R.raw.germany);
        insertWord(db, "Level 7", "German", "Γερμανός", R.raw.germany);
        insertWord(db, "Level 7", "Greece", "Ελλάδα", R.raw.greece);
        insertWord(db, "Level 7", "Greek", "Έλληνας", R.raw.greek);
        insertWord(db, "Level 7", "Hungary", "Ουγγαρία", R.raw.hungary);
        insertWord(db, "Level 7", "Hungarian", "Ουγγαρός", R.raw.hungar-
ian);
```

```
        insertWord(db, "Level 7", "India", "Ινδία", R.raw.india);
        insertWord(db, "Level 7", "Indian", "Ινδός", R.raw.indian);
        insertWord(db, "Level 7", "Ireland", "Ιρλανδία", R.raw.ireland);
        insertWord(db, "Level 7", "Irish", "Ιρλανδός", R.raw.irish);
        insertWord(db, "Level 7", "Italy", "Ιταλία", R.raw.italy);
        insertWord(db, "Level 7", "Italian", "Ιταλός", R.raw.italian);
        insertWord(db, "Level 7", "Japan", "Ιαπωνία", R.raw.japan);
        insertWord(db, "Level 7", "Japanese", "Ιάπωνας", R.raw.japanese);
        insertWord(db, "Level 7", "Latvia", "Λετονία", R.raw.latvia);
        insertWord(db, "Level 7", "Latvian", "Λετονός", R.raw.latvian);
        insertWord(db, "Level 7", "Lithuania", "Λιθουανία", R.raw.lithua-
nia);
        insertWord(db, "Level 7", "Lithuanian", "Λιθουανός", R.raw.lithua-
nian);
        insertWord(db, "Level 7", "Malta", "Μάλτα", R.raw.malta);
        insertWord(db, "Level 7", "Maltese", "Μαλτέζος", R.raw.maltese);
        insertWord(db, "Level 7", "Mexico", "Μεξικό", R.raw.mexico);
        insertWord(db, "Level 7", "Mexican", "Μεξικάνος", R.raw.mexican);
        insertWord(db, "Level 7", "Moldova", "Μολδαβία", R.raw.moldova);
        insertWord(db, "Level 7", "Moldovan", "Μολδαβός", R.raw.moldovan);
        insertWord(db, "Level 7", "Netherlands", "Ολλανδία", R.raw.nether-
lands);
        insertWord(db, "Level 7", "Dutch", "Ολλανδός", R.raw.dutch);
        insertWord(db, "Level 7", "Norway", "Νορβηγία", R.raw.norway);
        insertWord(db, "Level 7", "Norwegian", "Νορβηγός", R.raw.norwe-
gian);
        insertWord(db, "Level 7", "Poland", "Πολωνία", R.raw.poland);
        insertWord(db, "Level 7", "Polish", "Πολωνός", R.raw.polish);
        insertWord(db, "Level 7", "Portugal", "Πορτογαλία", R.raw.portu-
gal);
        insertWord(db, "Level 7", "Portuguese", "Πορτογάλος", R.raw.portu-
guese);
        insertWord(db, "Level 7", "Romania", "Ρουμανία", R.raw.romania);
        insertWord(db, "Level 7", "Romanian", "Ρουμάνος", R.raw.romanian);
        insertWord(db, "Level 7", "Russia", "Ρωσία", R.raw.russia);
        insertWord(db, "Level 7", "Russian", "Ρώσος", R.raw.russian);
        insertWord(db, "Level 7", "Serbia", "Σερβία", R.raw.serbia);
        insertWord(db, "Level 7", "Serbian", "Σέρβος", R.raw.serbian);
        insertWord(db, "Level 7", "Slovakia", "Σλοβακία", R.raw.slovakia);
        insertWord(db, "Level 7", "Slovakian", "Σλοβάκος", R.raw.slo-
vakian);
        insertWord(db, "Level 7", "Spain", "Ισπανία", R.raw.spain);
        insertWord(db, "Level 7", "Spanish", "Ισπανός", R.raw.spanish);
        insertWord(db, "Level 7", "Sweden", "Σουηδία", R.raw.sweden);
        insertWord(db, "Level 7", "Swedish", "Σουηδός", R.raw.swedish);
        insertWord(db, "Level 7", "Switzerland", "Ελβετία", R.raw.switzer-
land);
        insertWord(db, "Level 7", "Swiss", "Ελβετός", R.raw.swiss);
        insertWord(db, "Level 7", "Turkey", "Τουρκία", R.raw.turkey);
        insertWord(db, "Level 7", "Turkish", "Τούρκος", R.raw.turkish);
        insertWord(db, "Level 7", "Ukraine", "Ουκρανία", R.raw.ukraine);
        insertWord(db, "Level 7", "Ukrainian", "Ουκρανός", R.raw.ukrain-
ian);

        /* Level 8 Words */
        insertWord(db, "Level 8", "What's your telephone number?", "Ποιός
είναι ο αριθμός τηλεφώνου σου;", R.raw.what_s_your_telephone_number);
        insertWord(db, "Level 8", "My telephone number is ...", "Ο αριθμός
τηλεφώνου μου είναι ...", R.raw.my_telephone_number_is);
        insertWord(db, "Level 8", "the telephone", "το τηλέφωνο",
R.raw.the_telephone);
```

```
        insertWord(db, "Level 8", "the number", "ο αριθμός", R.raw.the_num-
ber);
        insertWord(db, "Level 8", "the telephone number", "ο αριθμός
τηλεφώνου", R.raw.the_telephone_number);
        insertWord(db, "Level 8","0 zero", "μηδέν", R.raw.zero);
        insertWord(db, "Level 8","1 one", "ένα", R.raw.one);
        insertWord(db, "Level 8","2 two", "δύο", R.raw.two);
        insertWord(db, "Level 8","3 three", "τρία", R.raw.three);
        insertWord(db, "Level 8","4 four", "τέσσερα", R.raw.four);
        insertWord(db, "Level 8","5 five", "πέντε", R.raw.five);
        insertWord(db, "Level 8","6 six", "έξι", R.raw.six);
        insertWord(db, "Level 8","7 seven", "επτά", R.raw.seven);
        insertWord(db, "Level 8","8 eight", "οκτώ", R.raw.eight);
        insertWord(db, "Level 8","9 nine", "εννέα", R.raw.nine);
        insertWord(db, "Level 8","10 ten", "δέκα", R.raw.ten);
        insertWord(db, "Level 8","11 eleven", "έντεκα", R.raw.eleven);
        insertWord(db, "Level 8","12 twelve", "δώδεκα", R.raw.twelve);
        insertWord(db, "Level 8","13 thirteen", "δέκα-τρια", R.raw.thir-
teen);
        insertWord(db, "Level 8","14 fourteen", "δεκα-τέσσερα", R.raw.four-
teen);
        insertWord(db, "Level 8","15 fifteen", "δεκα-πέντε", R.raw.fif-
teen);
        insertWord(db, "Level 8","16 sixteen", "δεκα-έξι", R.raw.sixteen);
        insertWord(db, "Level 8","17 seventeen", "δεκα-έπτα", R.raw.seven-
teen);
        insertWord(db, "Level 8","18 eighteen", "δεκα-οκτω", R.raw.eight-
een);
        insertWord(db, "Level 8","19 nineteen", "δεκα-εννέα", R.raw.nine-
teen);
        insertWord(db, "Level 8","20 twenty", "είκοσι", R.raw.twenty);
        insertWord(db, "Level 8","21 twenty-one", "είκοσι-ένα",
R.raw.twenty_one);
        insertWord(db, "Level 8","22 twenty-two", "είκοσι-δύο",
R.raw.twenty_two);
        insertWord(db, "Level 8","23 twenty-three", "είκοσι-τρία",
R.raw.twenty_three);
        insertWord(db, "Level 8","24 twenty-four", "είκοσι-τέσσερα",
R.raw.twenty_four);
        insertWord(db, "Level 8","25 twenty-five", "είκοσι-πεντε",
R.raw.twenty_five);
        insertWord(db, "Level 8","26 twenty-six", "είκοσι-έξι",
R.raw.twenty_six);
        insertWord(db, "Level 8","27 twenty-seven", "είκοσι-επτά",
R.raw.twenty_seven);
        insertWord(db, "Level 8","28 twenty-eight", "είκοσι-οκτώ",
R.raw.twenty_eight);
        insertWord(db, "Level 8","29 twenty-nine", "είκοσι-εννέα",
R.raw.twenty_nine);
        insertWord(db, "Level 8","30 thirty", "τριάντα", R.raw.thirty);
        insertWord(db, "Level 8","31 thirty-one", "τριάντα-ένα",
R.raw.thirty_one);
        insertWord(db, "Level 8","32 thirty-two", "τριάντα-δύο",
R.raw.thirty_two);
        insertWord(db, "Level 8","33 thirty-three", "τριάντα-τρία",
R.raw.thirty_three);
        insertWord(db, "Level 8","34 thirty-four", "τριάντα-τέσσερα",
R.raw.thirty_four);
        insertWord(db, "Level 8","35 thirty-five", "τριαντα-πέντε",
R.raw.thirty_five);
```

```
        insertWord(db, "Level 8","36 thirty-six", "τριάντα-έξι",
R.raw.thirty_six);
        insertWord(db, "Level 8","37 thirty-seven", "τριάντα-επτά",
R.raw.thirty_seven);
        insertWord(db, "Level 8","38 thirty-eight", "τριάντα-οκτώ",
R.raw.thirty_eight);
        insertWord(db, "Level 8","39 thirty-nine", "τριάντα-εννέα",
R.raw.thirty_nine);
        insertWord(db, "Level 8","40 forty", "σαράντα", R.raw.forty);
        insertWord(db, "Level 8","41 forty-one", "σαραντα-ένα",
R.raw.forty_one);
        insertWord(db, "Level 8","42 forty-two", "σαράντα-δύο",
R.raw.forty_two);
        insertWord(db, "Level 8","43 forty-three", "σαράντα-τρία",
R.raw.forty_three);
        insertWord(db, "Level 8","44 forty-four", "σαράντα-τέσσερα",
R.raw.forty_four);
        insertWord(db, "Level 8","45 forty-five", "σαράντα-πέντε",
R.raw.forty_five);
        insertWord(db, "Level 8","46 forty-six", "σαράντα-έξι",
R.raw.forty_six);
        insertWord(db, "Level 8","47 forty-seven", "σαράντα-επτά",
R.raw.forty_seven);
        insertWord(db, "Level 8","48 forty-eight", "σαράντα-οκτώ",
R.raw.forty_eight);
        insertWord(db, "Level 8","49 forty-nine", "σαράντα-εννεά",
R.raw.forty_nine);
        insertWord(db, "Level 8","50 fifty", "πενήντα", R.raw.fifty);
        insertWord(db, "Level 8","51 fifty-one", "πενήντα-ένα",
R.raw.fifty_one);
        insertWord(db, "Level 8","52 fifty-two", "πενήντα-δύο",
R.raw.fifty_two);
        insertWord(db, "Level 8","53 fifty-three", "πενήντα-τρία",
R.raw.fifty_three);
        insertWord(db, "Level 8", "54 fifty-four", "πενήντα-τέσσερα",
R.raw.fifty_four);
        insertWord(db, "Level 8", "55 fifty-five", "πενήντα-πέντε",
R.raw.fifty_five);
        insertWord(db, "Level 8", "56 fifty-six", "πενήντα-έξι",
R.raw.fifty_six);
        insertWord(db, "Level 8", "57 fifty-seven", "πενήντα-επτά",
R.raw.fifty_seven);
        insertWord(db, "Level 8", "58 fifty-eight", "πενήντα-οκτώ",
R.raw.fifty_eight);
        insertWord(db, "Level 8", "59 fifty-nine", "πενήντα-εννέα",
R.raw.fifty_nine);
        insertWord(db, "Level 8", "60 sixty", "εξήντα", R.raw.sixty);
        insertWord(db, "Level 8", "61 sixty-one", "εξήντα-ένα",
R.raw.sixty_one);
        insertWord(db, "Level 8", "62 sixty-two", "εξήντα-δύο",
R.raw.sixty_two);
        insertWord(db, "Level 8", "63 sixty-three", "εξηντα-τρία",
R.raw.sixty_three);
        insertWord(db, "Level 8", "64 sixty-four", "εξήντα-τέσσερα",
R.raw.sixty_four);
        insertWord(db, "Level 8", "65 sixty-five", "εξήντα-πέντε",
R.raw.sixty_five);
        insertWord(db, "Level 8", "66 sixty-six", "εξήντα-έξι",
R.raw.sixty_six);
        insertWord(db, "Level 8", "67 sixty-seven", "εξήντα-επτά",
R.raw.sixty_seven);
```

```
        insertWord(db, "Level 8", "68 sixty-eight", "εξήντα-οκτώ",
R.raw.sixty_eight);
        insertWord(db, "Level 8", "69 sixty-nine", "εξήντα-εννεά",
R.raw.sixty_nine);
        insertWord(db, "Level 8", "70 seventy", "εβδομήντα", R.raw.sev-
enty);
        insertWord(db, "Level 8", "71 seventy-one", "εβδομήντα-ένα",
R.raw.seventy_one);
        insertWord(db, "Level 8", "72 seventy-two", "εβδομήντα-δύο",
R.raw.seventy_two);
        insertWord(db, "Level 8", "73 seventy-three", "εβδομήντα-τρία",
R.raw.seventy_three);
        insertWord(db, "Level 8", "74 seventy-four", "εβδομήντα-τέσσερα",
R.raw.seventy_four);
        insertWord(db, "Level 8", "75 seventy-five", "εβδομήντα-πέντε",
R.raw.seventy_five);
        insertWord(db, "Level 8", "76 seventy-six", "εβδομήντα-έξι",
R.raw.seventy_six);
        insertWord(db, "Level 8", "77 seventy-seven", "εβδομήντα-επτά",
R.raw.seventy_seven);
        insertWord(db, "Level 8", "78 seventy-eight", "εβδομήντα-οκτώ",
R.raw.seventy_eight);
        insertWord(db, "Level 8", "79 seventy-nine", "εβδομήντα-εννέα",
R.raw.seventy_nine);
        insertWord(db, "Level 8", "80 eighty", "ογδόντα", R.raw.eighty);
        insertWord(db, "Level 8", "81 eighty-one", "ογδόντα-ένα",
R.raw.eighty_one);
        insertWord(db, "Level 8", "82 eighty-two", "ογδόντα-δύο",
R.raw.eighty_two);
        insertWord(db, "Level 8", "83 eighty-three", "ογδόντα-τρία",
R.raw.eighty_three);
        insertWord(db, "Level 8", "84 eighty-four", "ογδοντά-τέσσερα",
R.raw.eighty_four);
        insertWord(db, "Level 8", "85 eighty-five", "ογδόντα-πέντε",
R.raw.eighty_five);
        insertWord(db, "Level 8", "86 eighty-six", "ογδόντα-έξι",
R.raw.eighty_six);
        insertWord(db, "Level 8", "87 eighty-seven", "ογδόντα-έπτα",
R.raw.eighty_seven);
        insertWord(db, "Level 8", "88 eighty-eight", "ογδόντα-οκτώ",
R.raw.eighty_eight);
        insertWord(db, "Level 8", "89 eighty-nine", "ογδόντα-εννέα",
R.raw.eighty_nine);
        insertWord(db, "Level 8", "90 ninety", "ενενήντα", R.raw.ninety);
        insertWord(db, "Level 8", "91 ninety-one", "ενενήντα-ένα",
R.raw.ninety_one);
        insertWord(db, "Level 8", "92 ninety-two", "ενενήντα-δύο",
R.raw.ninety_two);
        insertWord(db, "Level 8", "93 ninety-three", "ενενήντα-τρία",
R.raw.ninety_three);
        insertWord(db, "Level 8", "94 ninety-four", "ενενήντα-τέσσερα",
R.raw.ninety_four);
        insertWord(db, "Level 8", "95 ninety-five", "ενενήντα-πέντε",
R.raw.ninety_five);
        insertWord(db, "Level 8", "96 ninety-six", "ενενήντα-έξι",
R.raw.ninety_six);
        insertWord(db, "Level 8", "97 ninety-seven", "ενενήντα-έπτα",
R.raw.ninety_seven);
        insertWord(db, "Level 8", "98 ninety-eight", "ενενήντα-όκτω",
R.raw.ninety_eight);
```

```
        insertWord(db, "Level 8", "99 ninety-nine", "ενενήντα-εννέα",
R.raw.ninety_nine);
        insertWord(db, "Level 8","100 one hundred", "εκατό", R.raw.one_hun-
dred);
        insertWord(db, "Level 8","1000 one thousand", "χίλια",
R.raw.one_thousand);

        /* Level 9 Words */
        insertWord(db, "Level 9", "Are you ready to order?", "Είστε έτοιμοι
να παραγγείλετε;", R.raw.are_you_ready_to_order);
        insertWord(db, "Level 9", "can", "μπορώ", R.raw.can);
        insertWord(db, "Level 9", "can we have the menu please?", "Μπορούμε
να έχουμε το μενού παρακαλώ;", R.raw.can_we_have_the_menu_please);
        insertWord(db, "Level 9", "drink", "πίνω", R.raw.drink);
        insertWord(db, "Level 9", "eat", "τρώω", R.raw.eat);
        insertWord(db, "Level 9", "for", "για", R.raw.for_);
        insertWord(db, "Level 9", "I would like", "Θα ήθελα",
R.raw.i_would_like);
        insertWord(db, "Level 9", "order", "παραγγέλνω", R.raw.order);
        insertWord(db, "Level 9", "ready", "έτοιμος", R.raw.ready);
        insertWord(db, "Level 9", "the table", "το τραπέζι", R.raw.the_ta-
ble);
        insertWord(db, "Level 9", "take-away", "πακέτο προς αποστολή",
R.raw.take_away);
        insertWord(db, "Level 9", "the bill", "ο λογαριασμός",
R.raw.the_bill);
        insertWord(db, "Level 9", "the chopsticks", "τα κινέζικα ξυλάκια",
R.raw.the_chopsticks);
        insertWord(db, "Level 9", "the fork", "το πιρούνι",
R.raw.the_fork);
        insertWord(db, "Level 9", "the knife", "το μαχαιρί",
R.raw.the_knife);
        insertWord(db, "Level 9", "the menu", "το μενού", R.raw.the_menu);
        insertWord(db, "Level 9", "the_restaurant", "το εστιατόριο",
R.raw.the_restaurant);
        insertWord(db, "Level 9", "the spoon", "το κουτάλι",
R.raw.the_spoon);
        insertWord(db, "Level 9", "to", "να", R.raw.to);

        /* Level 10 Words */
        insertWord(db, "Level 10", "also", "επίσης", R.raw.also);
        insertWord(db, "Level 10", "beautiful", "πανέμορφος", R.raw.beauti-
ful);
        insertWord(db, "Level 10", "big", "μεγάλος (μέγεθος)", R.raw.big);
        insertWord(db, "Level 10", "cool", "δροσερός", R.raw.cool);
        insertWord(db, "Level 10", "cute", "χαριτωμένος", R.raw.cute);
        insertWord(db, "Level 10", "fat", "χοντρός", R.raw.fat);
        insertWord(db, "Level 10", "handsome", "όμορφος", R.raw.handsome);
        insertWord(db, "Level 10", "I don't think so", "Δεν νομίζω",
R.raw.i_don_t_think_so);
        insertWord(db, "Level 10", "I think so too", "Νομίζω κι γω",
R.raw.i_think_so_too);
        insertWord(db, "Level 10", "large", "μεγάλος (μέγεθος - μεγαλύτερο
του big)", R.raw.large);
        insertWord(db, "Level 10", "little", "μικρός (μέγεθος)", R.raw.lit-
tle);
        insertWord(db, "Level 10", "long", "μακρύς", R.raw.long_);
        insertWord(db, "Level 10", "me neither", "ούτε κι γω",
R.raw.me_neither);
        insertWord(db, "Level 10", "me too", "κι γω", R.raw.me_too);
        insertWord(db, "Level 10", "nice", "ωραίος", R.raw.nice);
```

```
        insertWord(db, "Level 10", "pretty", "όμορφος", R.raw.pretty);
        insertWord(db, "Level 10", "short", "κοντός", R.raw.short_);
        insertWord(db, "Level 10", "slim", "αδύνατος", R.raw.slim);
        insertWord(db, "Level 10", "small", "μικρός", R.raw.small);
        insertWord(db, "Level 10", "strong", "δυνατός", R.raw.strong);
        insertWord(db, "Level 10", "that", "εκείνο", R.raw.that);
        insertWord(db, "Level 10", "thin", "λεπτός", R.raw.thin);
        insertWord(db, "Level 10", "think", "νομίζω", R.raw.think);
        insertWord(db, "Level 10", "too", "επίσης", R.raw.too);
        insertWord(db, "Level 10", "too many", "υπερβολικά πολλά",
R.raw.too_many);
        insertWord(db, "Level 10", "too much", "υπερβολικά πολύ",
R.raw.too_much);
        insertWord(db, "Level 10", "ugly", "άσχημος", R.raw.ugly);
        insertWord(db, "Level 10", "weak", "αδύναμος", R.raw.weak);
        insertWord(db, "Level 10", "what", "τι", R.raw.what);
        insertWord(db, "Level 10", "which", "το οποίο", R.raw.which);
        insertWord(db, "Level 10", "who", "ο οποίος", R.raw.who);
        insertWord(db, "Level 10", "wonderful", "υπέροχος", R.raw.wonder-
ful);

        /* Level 11 Words */
        insertWord(db, "Level 11", "any", "όποιος", R.raw.any);
        insertWord(db, "Level 11", "Do you understand?", "Καταλαβαίνεις;",
R.raw.do_you_understand);
        insertWord(db, "Level 11", "good luck", "καλή τύχη",
R.raw.good_luck);
        insertWord(db, "Level 11", "happy to help you", "χαίρομαι που
βοηθώ", R.raw.happy_to_help_you);
        insertWord(db, "Level 11", "help", "βοήθεια / βοηθώ", R.raw.help);
        insertWord(db, "Level 11", "here", "εδώ", R.raw.here);
        insertWord(db, "Level 11", "here you go", "ορίστε",
R.raw.here_you_go);
        insertWord(db, "Level 11", "how do you say ...?", "Πως λές ... ;",
R.raw.how_do_you_say);
        insertWord(db, "Level 11", "know", "ξέρω / γνωρίζω", R.raw.know);
        insertWord(db, "Level 11", "meet", "συναντώ", R.raw.meet);
        insertWord(db, "Level 11", "pleasure", "ευχαρίστηση", R.raw.pleas-
ure);
        insertWord(db, "Level 11", "Pleasure to meet you", "Χάρηκα για τη
γνωριμία", R.raw.pleasure);
        insertWord(db, "Level 11", "say", "λέω", R.raw.say);
        insertWord(db, "Level 11", "some", "κάμποσο / μερικό", R.raw.some);
        insertWord(db, "Level 11", "tell", "λέω", R.raw.tell);
        insertWord(db, "Level 11", "That's great", "Αυτό είναι ωραίο",
R.raw.that_s_great);
        insertWord(db, "Level 11", "the luck", "η τύχη", R.raw.the_luck);
        insertWord(db, "Level 11", "understand", "καταλαβαίνω", R.raw.un-
derstand);
        insertWord(db, "Level 11", "welcome", "καλώς ορίσατε", R.raw.wel-
come);

        /* Level 12 Words */
        insertWord(db, "Level 12", "all", "όλα", R.raw.all);
        insertWord(db, "Level 12", "anything", "οτιδήποτε", R.raw.any-
thing);
        insertWord(db, "Level 12", "everyone", "όλοι", R.raw.everyone);
        insertWord(db, "Level 12", "everything", "όλα", R.raw.everything);
        insertWord(db, "Level 12", "of", "από (όχι για καταγωγές)",
R.raw.of);
        insertWord(db, "Level 12", "something", "κάτι", R.raw.something);
```

```
        insertWord(db, "Level 12", "the bottle", "το μπουκάλι",
R.raw.the_bottle);
        insertWord(db, "Level 12", "the cup", "το φλυτζάνι",
R.raw.the_cup);
        insertWord(db, "Level 12", "the dessert", "το επιδόρπιο",
R.raw.the_dessert);
        insertWord(db, "Level 12", "the dish", "το πιάτο (συνήθως όχι το α-
ντικείμενο)", R.raw.the_dish);
        insertWord(db, "Level 12", "the glass", "το ποτήρι",
R.raw.the_glass);
        insertWord(db, "Level 12", "the main course", "το κυρίως πιάτο",
R.raw.the_main_course);
        insertWord(db, "Level 12", "the plate", "το πιάτο",
R.raw.the_plate);
        insertWord(db, "Level 12", "the starter", "το ορεκτικό",
R.raw.the_starter);
        insertWord(db, "Level 12", "Would you like ...?", "Θα θέλατε ...
;", R.raw.would_you_like);

    }


    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

    }

    private static void insertWord(SQLiteDatabase db, String level, String
englishWord, String translatedWord) {

        ContentValues wordValues = new ContentValues();

        wordValues.put(WordEntry.COLUMN_LEVEL, level);
        wordValues.put(WordEntry.COLUMN_ENGLISH_WORD, englishWord);
        wordValues.put(WordEntry.COLUMN_TRANSLATED_WORD, translatedWord);

        db.insert(WordEntry.TABLE_NAME, null, wordValues);
    }

    private static void insertWord(SQLiteDatabase db, String level, String
englishWord, String translatedWord, int soundId) {

        ContentValues wordValues = new ContentValues();

        wordValues.put(WordEntry.COLUMN_LEVEL, level);
        wordValues.put(WordEntry.COLUMN_ENGLISH_WORD, englishWord);
        wordValues.put(WordEntry.COLUMN_TRANSLATED_WORD, translatedWord);
        wordValues.put(WordEntry.COLUMN_WORD_SOUND_ID, soundId);

        db.insert(WordEntry.TABLE_NAME, null, wordValues);
    }
}
```

list_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/card_view"
    android:layout_gravity="center"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="10dp"
    android:layout_margin="10dp">

    <android.support.constraint.ConstraintLayout
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:id="@+id/language_word"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginBottom="8dp"
            android:fontFamily="@font/open_sans_light"
            android:text="Language Word"
            android:textColor="@android:color/black"
            android:textSize="20sp"
            android:textStyle="bold"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toStartOf="@+id/guideline3"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.0" />

        <TextView
            android:id="@+id/translated_word"
            style="@style/LanguageTranslatedWordTextStyle"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginBottom="8dp"
            android:fontFamily="@font/open_sans_light"
            android:text="Translated Word"
            android:textColor="@android:color/black"
            android:textSize="15sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toStartOf="@+id/guideline3"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/language_word" />

        <ImageButton
            android:id="@+id/image_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
```

```
            android:layout_marginEnd="8dp"
            android:layout_marginBottom="8dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="@+id/guideline3"
            app:layout_constraintTop_toTopOf="parent"
            app:srcCompat="@drawable/baseline_volume_up_black_36" />

        <android.support.constraint.Guideline
            android:id="@+id/guideline3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            app:layout_constraintGuide_percent="0.8" />

    </android.support.constraint.ConstraintLayout>
</LinearLayout>
```

WordCursorAdapter.java

```
package com.selgo.quickspeak;

import android.content.Context;
import android.database.Cursor;
import android.media.MediaPlayer;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CursorAdapter;
import android.widget.ImageButton;
import android.widget.TextView;

public class WordCursorAdapter extends CursorAdapter {

    private TextView word;
    private TextView translatedWord;
    private ImageButton imageButton;

    private String wordText;
    private String translatedWordText;

    View.OnClickListener onClickListener;

    private MediaPlayer mediaPlayer;


    public WordCursorAdapter(Context context, Cursor cursor) {
        super(context, cursor, 0);
    }

    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        return LayoutInflater.from(context).inflate(R.layout.list_item,
parent, false);
    }

    @Override
    public void bindView(View view, Context context, Cursor cursor) {

        word = (TextView) view.findViewById(R.id.language_word);
```

```
        translatedWord = (TextView) view.findViewById(R.id.trans-
lated_word);
        imageButton = (ImageButton) view.findViewById(R.id.image_button);

        wordText = cursor.getString(1);
        translatedWordText = cursor.getString(2);
        final int soundId = cursor.getInt(3);

        word.setText(wordText);
        translatedWord.setText(translatedWordText);

        onClickListener = new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (mediaPlayer != null) {
                    mediaPlayer.release();
                    mediaPlayer = null;
                }
                mediaPlayer = MediaPlayer.create(view.getContext(),
soundId);
                mediaPlayer.start();
                mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCom-
pletionListener() {
                    @Override
                    public void onCompletion(MediaPlayer mediaPlayer) {
                        mediaPlayer.release();
                        mediaPlayer = null;
                    }
                });
            }
        };

        imageButton.setOnClickListener(onClickListener);
    }

    public MediaPlayer getMediaPlayer() {
        return mediaPlayer;
    }
}
```

fragment_word.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ListView
        android:id="@+id/word_listview"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

WordFragment.java

```
package com.selgo.quickspeak;

import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.Toast;

import com.selgo.quickspeak.data.WordContract;
import com.selgo.quickspeak.data.WordDbHelper;

public class WordFragment extends Fragment {

    private View view;
    private SQLiteDatabase db;
    private Cursor cursor;
    private WordDbHelper mDbHelper;
    private WordCursorAdapter wordCursorAdapter;
    private ListView listView;
    private LinearLayout linearLayout;
    private String levelValue;
    private MediaPlayer mediaPlayer;

    private Toast toast;

    public WordFragment() {
        setRetainInstance(true);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        linearLayout = (LinearLayout) inflater.inflate(R.layout.frag-
ment_word, container, false);
        return linearLayout;
    }

    @Override
    public void onStart() {
        super.onStart();

        view = getView();

        levelValue = (String) getArguments().getCharSequence("itemTitle");
        wordsRetrieveQuery(view, levelValue);
    }

    public void wordsRetrieveQuery(View view, String conditionValue) {
        try {
            mDbHelper = new WordDbHelper(view.getContext());
            db = mDbHelper.getReadableDatabase();
```

```
            cursor = db.query(WordContract.WordEntry.TABLE_NAME,
                    new String[]{WordContract.WordEntry._ID, WordCon-
tract.WordEntry.COLUMN_ENGLISH_WORD, WordContract.WordEntry.COLUMN_TRANS-
LATED_WORD, WordContract.WordEntry.COLUMN_WORD_SOUND_ID},
                    WordContract.WordEntry.COLUMN_LEVEL + "=?",
                    new String[] {conditionValue},
                    null, null, null);


            listView = view.findViewById(R.id.word_listview);
            wordCursorAdapter = new WordCursorAdapter(view.getContext(),
cursor);
            mediaPlayer = wordCursorAdapter.getMediaPlayer();
            listView.setAdapter(wordCursorAdapter);

        } catch (SQLException e) {
            toast = Toast.makeText(view.getContext(), "Database unavaila-
ble", Toast.LENGTH_SHORT);
            toast.show();
        }
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        cursor.close();
        db.close();
    }


}
```

app_toolbar.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.Toolbar
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="?attr/colorPrimary"
    android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"/>
```

activity_word_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <include
            layout="@layout/app_toolbar"
            android:id="@+id/app_toolbar"/>

        <FrameLayout
            android:id="@+id/frag_container"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />

    </LinearLayout>

    <android.support.design.widget.NavigationView
        android:id="@+id/navigation_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/navigation_menu_header"
        app:menu="@menu/navigation_drawer_menu"/>

</android.support.v4.widget.DrawerLayout>
```

navigation_menu_header.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="40dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="start|bottom"
        android:layout_marginLeft="15dp"
        android:fontFamily="@font/open_sans_light"
        android:text="@string/app_name"
        android:textAppearance="@style/TextAppearance.AppCompat.Body1"
        android:textSize="20sp"
        android:textStyle="bold" />

</FrameLayout>
```

navigation_drawer_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:title="@string/english">
        <menu>
            <group android:checkableBehavior="single">
                <item
                    android:id="@+id/level_1"
                    android:title="@string/level_1" />

                <item
                    android:id="@+id/level_2"
                    android:title="@string/level_2" />

                <item
                    android:id="@+id/level_3"
                    android:title="@string/level_3" />

                <item
                    android:id="@+id/level_4"
                    android:title="@string/level_4" />

                <item
                    android:id="@+id/level_5"
                    android:title="@string/level_5" />

                <item
                    android:id="@+id/level_6"
                    android:title="@string/level_6" />

                <item
                    android:id="@+id/level_7"
                    android:title="@string/level_7" />

                <item
                    android:id="@+id/level_8"
                    android:title="@string/level_8" />

                <item
                    android:id="@+id/level_9"
                    android:title="@string/level_9" />

                <item
                    android:id="@+id/level_10"
                    android:title="@string/level_10" />

                <item
                    android:id="@+id/level_11"
                    android:title="@string/level_11" />

                <item
                    android:id="@+id/level_12"
                    android:title="@string/level_12" />
            </group>
        </menu>
    </item>
</menu>
```

WordListActivity.java

```
package com.selgo.quickspeak;

import android.os.Bundle;
import android.support.design.widget.NavigationView;
import android.support.v4.app.FragmentTransaction;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;

public class WordListActivity extends AppCompatActivity implements Naviga-
tionView.OnNavigationItemSelectedListener {

    private CharSequence itemTitle = "Level 1";
    private Toolbar toolbar;
    private DrawerLayout drawerLayout;
    private ActionBarDrawerToggle actionBarDrawerToggle;
    private WordFragment wordFragment;
    private Bundle arguments;
    private NavigationView navigationView;
    private FragmentTransaction fragmentTransaction;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_word_list);

        if (savedInstanceState != null) {
            itemTitle = savedInstanceState.getCharSequence("itemTitle");
        }

        toolbar = (Toolbar) findViewById(R.id.app_toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle(itemTitle);

        drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
        actionBarDrawerToggle = new ActionBarDrawerToggle(this, drawerLay-
out, toolbar, R.string.nav_open_drawer, R.string.nav_close_drawer);
        drawerLayout.addDrawerListener(actionBarDrawerToggle);
        actionBarDrawerToggle.syncState();

        navigationView = (NavigationView) findViewById(R.id.naviga-
tion_view);
        navigationView.setNavigationItemSelectedListener(this);
        navigationView.setCheckedItem(R.id.level_1);

        arguments = new Bundle();
        arguments.putCharSequence("itemTitle", itemTitle);

        if (wordFragment == null) {
            wordFragment = new WordFragment();
            wordFragment.setArguments(arguments);

            fragmentTransaction = getSupportFragmentManager().beginTransac-
tion();
            fragmentTransaction.add(R.id.frag_container, wordFragment);
            fragmentTransaction.commit();
```

```
        }

    }

    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        itemTitle = item.getTitle();

        getSupportActionBar().setTitle(itemTitle);

        drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);

        item.setChecked(true);
        drawerLayout.closeDrawers();

        arguments = new Bundle();
        arguments.putCharSequence("itemTitle", itemTitle);

        wordFragment = new WordFragment();
        wordFragment.setArguments(arguments);

        fragmentTransaction = getSupportFragmentManager().beginTransac-
tion();
        fragmentTransaction.replace(R.id.frag_container, wordFragment);
        fragmentTransaction.commit();

        return true;
    }

    @Override
    public void onBackPressed() {
        drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);

        if(drawerLayout.isDrawerOpen(GravityCompat.START)) {
            drawerLayout.closeDrawers();
        } else {
            super.onBackPressed();
        }
    }

    @Override
    protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putCharSequence("itemTitle", itemTitle);
    }
}
```

colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

styles.xml

```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

    <style name="LanguageWordTextStyle">
        <item name="android:textSize">15dp</item>
        <item name="android:textColor">#000000</item>
        <item name="android:fontFamily">@font/open_sans_light</item>
    </style>

    <style name="LanguageTranslatedWordTextStyle">
        <item name="android:textSize">13sp</item>
        <item name="android:textColor">#424242</item>
        <item name="android:fontFamily">@font/open_sans_light</item>
    </style>
</resources>
```

strings.xml

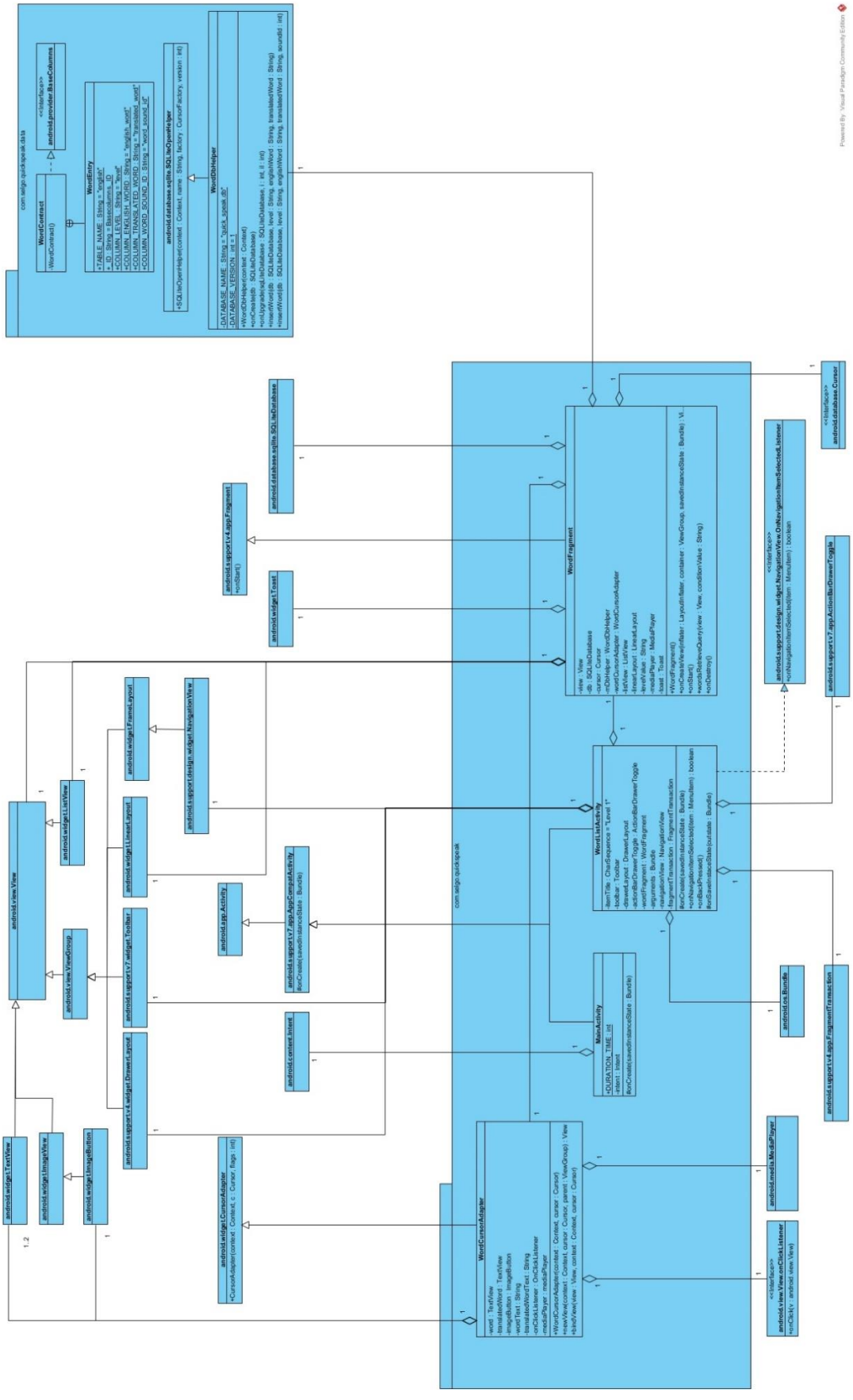
```
<resources>
    <string name="app_name">Quick Speak</string>
    <string name="nav_open_drawer">Open navigation drawer</string>
    <string name="nav_close_drawer">Close navigation drawer</string>


    <string name="english">English</string>

    <string name="level_1">Level 1</string>
    <string name="level_2">Level 2</string>
    <string name="level_3">Level 3</string>
    <string name="level_4">Level 4</string>
    <string name="level_5">Level 5</string>
    <string name="level_6">Level 6</string>
    <string name="level_7">Level 7</string>
    <string name="level_8">Level 8</string>
    <string name="level_9">Level 9</string>
    <string name="level_10">Level 10</string>
    <string name="level_11">Level 11</string>
    <string name="level_12">Level 12</string>
</resources>
```

AndroidManifest.xml

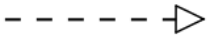
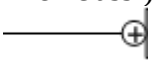
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.selgo.quickspeak">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".WordListActivity"
            android:configChanges="orientation|keyboardHidden"></activity>
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

5.4.2 Το Διάγραμμα Κλάσεων του Quick Speak

Σε αυτό το σημείο, θα παρουσιάσουμε το διάγραμμα κλάσεων της εφαρμογής, το οποίο παρουσιάζει το σύνολο όλων των κλάσεων της βιβλιοθήκης που χρησιμοποιήθηκαν καθώς και τις κλάσεις που δημιουργήθηκαν κατά την ανάπτυξη του Quick Speak, συμπεριλαμβανομένου και των σχέσεων τους. Το διάγραμμα, παρουσιάζεται σε μορφή UML. Για να διαχωρίσουμε, τις κλάσεις που δημιουργήθηκαν για την εφαρμογή, από τις κλάσεις της βιβλιοθήκης Android που χρησιμοποιήθηκαν, δώσαμε ως τίτλους των τελευταίων, τα ονόματα των πακέτων της βιβλιοθήκης μαζί με το αντίστοιχα ονόματα των κλάσεων (τα αντίστοιχα imports που έγιναν για κάθε κλάση του project όπως `android.content.Intent` για παράδειγμα). Το αρχείο εικόνας του διαγράμματος θα συμπεριληφθεί και στο συνοδευτικό υλικό της παρούσας πτυχιακής. Έτσι το ολοκληρωμένο διάγραμμα κλάσεων του Quick Speak, είναι:



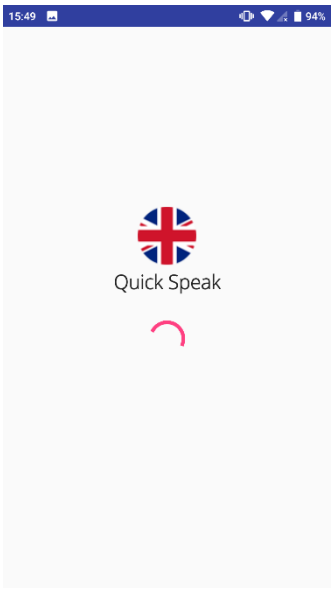
Εικόνα 26: Το Διάγραμμα Κλάσεων του Quick Speak

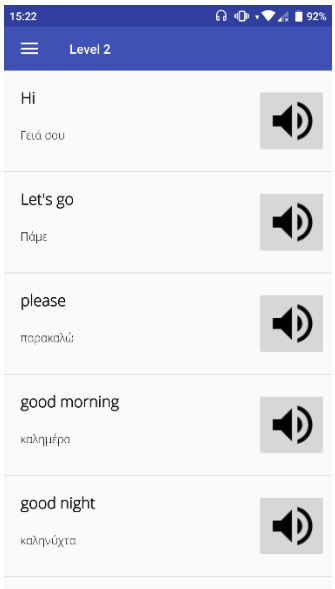
Οι σχέσεις των κλάσεων έχουν ως εξής. Κάθε κλάση-παιδί συνδέεται με τη κλάση πατέρα της, με τη χρήση του βέλους Κληρονομικότητας (Dependency)  . Για να δείξουμε το τρόπο με τον οποίο μια κλάση χρησιμοποιεί μια διεπαφή (java interface) χρησιμοποιήσαμε το βέλος της Υλοποίησης (Implementation)  . Για να δείξουμε τα αντικείμενα που χρησιμοποιεί κάθε κλάση, τη λεγόμενη σχέση **«HAS A»,** χρησιμοποιήσαμε το βέλος της Συσσωμάτωσης (Aggregation)  . Στο άκρα του βέλους αυτού, τοποθετήθηκε η πολλαπλότητα των στοιχείων κάθε κλάσης. Για παράδειγμα 1 στιγμιότυπο της κλάσης `MainActivity`, έχει 1 αντικείμενο της κλάσης `Intent`. Οι αντιστοιχίες αυτές σε όλο το διάγραμμα, εξαιρουμένως της περίπτωσης των κλάσεων `WordCursorAdapter` – `TextView`, είναι 1 προς 1. Τέλος για να δείξουμε το γεγονός ότι, η κλάση WordEntry είναι εσωτερική (inner class) της κλάσης WordContract, χρησιμοποιήσαμε το βέλος της Συνοχής (Containment)  .

5.5 Πιλοτική Επίδειξη του Quick Speak

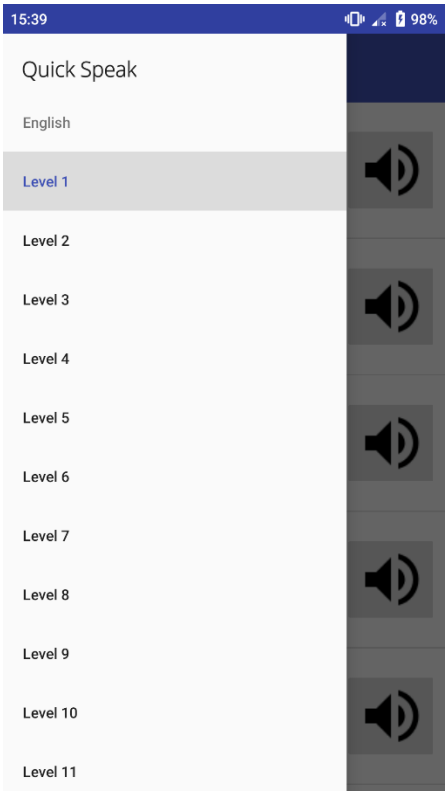
Σε αυτή την ενότητα θα πραγματοποιηθεί η επίδειξη χρήσης του Quick Speak. Όπως έχει ήδη αναφερθεί στη παρούσα πτυχιακή, η εφαρμογή απευθύνεται κυρίως σε χρήστες που έχουν καμία ή έστω και ελάχιστη γνώση των Αγγλικών. Έχοντας εγκαταστήσει την εφαρμογή σε μια συσκευή Android (με τη χρήση του αρχείου apk του συνοδευτικού υλικού), εντοπίζουμε το εικονίδιο του Quick Speak 

και το πατάμε (Στη παρούσα πτυχιακή δε προσθέσαμε κάποιο custom εικονίδιο, έτσι χρησιμοποιήθηκε το προκαθορισμένο από το Android Studio). Παρόλα αυτά η αλλαγή αυτή μπορεί να γίνει εύκολα με τη αλλαγή της ιδιότητας `android:icon`, στο αρχείο `AndroidManifest.xml`, ορίζοντας το εικονίδιο της επιλογής μας.





Αρχικά θα εμφανιστεί το αρχικό activity της εφαρμογής. Έπειτα από περίπου 1.2 δευτερόλεπτα, θα εμφανιστεί η αρχική λίστα με τα λήμματα του λεξικού (Level 1). Όπως είδαμε και στην υλοποίηση της εφαρμογής, κάθε στοιχείο μιας λίστας λημμάτων, περιέχει τη αγγλική λέξη ή φράση του επιπέδου που επιλέχθηκε, την αντίστοιχη μετάφραση στα ελληνικά, και ένα κουμπί το οποίο ο χρήστης μπορεί να πατήσει, και να ακροαστεί το εκάστοτε λήμμα στα αγγλικά. Αριστερά απεικονίζεται ένα στιγμιότυπο της αρχικής κατάστασης της εφαρμογής, καθώς και της λίστας λημμάτων επιπέδου 2 (Level 2). Για να αλλάξουμε το επίπεδο και να οδηγηθούμε σε άλλη λίστα λημμάτων, πατάμε στο εικονίδιο πάνω αριστερά  . Αυτό ενεργοποιεί το πλαϊνό menu με λίστα όλων των επιπέδων λημμάτων του Quick Speak. Παρακάτω εμφανίζεται η λίστα του navigation menu της εφαρμογής.



Από το menu, επιλέγουμε το επίπεδο που μας ενδιαφέρει, κι έτσι η εφαρμογή κλείνει το menu, μαρκάροντας το επιλεγμένο επίπεδο, και οδηγώντας μας στην αντίστοιχη λημμάτων, του επιπέδου αυτού. Το Quick Speak, υποστηρίζεται και σε όψη τοπίου (landscape mode). Η προκαθορισμένη γλώσσα εμφάνισης της διεπαφής του Quick Speak, είναι τα αγγλικά, παρόλα αυτά υποστηρίζονται και τα ελληνικά, για χρήστες που τα έχουν ως προκαθορισμένη γλώσσα στη συσκευή τους. Και έτσι ολοκληρώνεται η επίδειξη χρήσης του Quick Speak.

6. Αποτελέσματα και Επίλογος

Η ολοκλήρωση μιας εφαρμογής, όπως είναι το Quick Speak, μας δείχνει τη σπουδαιότητα της ανάπτυξης εφαρμογών Android. Όπως είδαμε η βιβλιοθήκη του Android, είναι ειδικά σχεδιασμένη, ώστε να αναπτύξει κάνεις μια απλή είτε και μια πολύπλοκη εφαρμογή, χωρίς παράλληλα να γνωρίζει ιδιαίτερες λεπτομέρειες για την ίδια την βιβλιοθήκη στο παρασκήνιο. Αυτή είναι άλλωστε και η έννοια της ενθυλάκωσης. Το Android Development όπως ονομάζεται και επίσημα, θεωρείται ένα πολύ καλό σημείο εκκίνησης, για να εκμεταλλευτεί κανείς τη «δύναμη» της αντικειμενοστρέφειας και συγκεκριμένα της γλώσσας Java, όπως είδαμε και στη παρούσα πτυχιακή. Στην ουσία για την ανάπτυξη του Quick Speak, δεν χρησιμοποιήθηκε τίποτα παραπάνω από ότι θα χρησιμοποιούσε κάνεις για τη κατασκευή ενός desktop λογισμικού, δηλαδή μια τυπική βιβλιοθήκη Java. Η μόνη διαφορά είναι ότι σε κάθε περίπτωση χρησιμοποιούμε τη κατάλληλη βιβλιοθήκη με βάση τη δουλειά και τη πλατφόρμα που θέλουμε να υποστηρίξουμε (όπως πχ τη χρήση του Swing για desktop εφαρμογές, το Android API για Android εφαρμογές κτλ.) . Έννοιες όπως αντικείμενο, κλάση, κληρονομικότητα κτλ, βρίσκουν πάντα εφαρμογή σε κάθε περίπτωση. Έτσι ένας χρήστης της γλώσσας Java, μπορεί εύκολα να προσαρμοστεί στο Android Development. Στη βιβλιογραφία παρατίθεται επίσης, ένα βιβλίο[133], το οποίο αποτελεί μια από τις εμπνεύσεις που κατέληξαν στο έργο της παρούσας πτυχιακής, και προτείνεται σε γνώστες των βασικών στοιχείων της Java.

Το αποτέλεσμα της πτυχιακής αυτής, ήταν στην ουσία να αναπτυχθεί μια εφαρμογή, η οποία θα είχε την ικανότητα να δείξει τα βασικά λήμματα της αγγλικής γλώσσας, με έναν απλό και γρήγορο τρόπο, βασισμένο στην επανάληψη και μάθηση αυτών. Πρόκειται για μία απλή εφαρμογή, η οποία δίνει έναυσμα στον αναγνώστη ώστε να σκεφτεί, την ευκολία να δημιουργήσει κάνεις τέτοιου είδους εφαρμογές, εκμεταλλευόμενος πάντα τις γνώσεις ανάπτυξης της, ώστε να μεταβεί σε ένα πιο προχωρημένο επίπεδο πέραν των γνώσεων αυτών. Η σκοπιμότητα της παρούσας πτυχιακής, είναι κυρίως εκπαιδευτική. Ο συγγραφέας της παρούσας πτυχιακής. επωφελήθηκε και εξοικειώθηκε με τα βασικά και ελαφρώς πιο προχωρημένα στοιχεία των εργαλείων σχεδίασης και ανάπτυξης που οδήγησαν στο τελικό αποτέλεσμα, του Quick Speak, με επιπλέον στόχο το να εμπνεύσει τον αναγνώστη να εισέλθει με τη σειρά του στο κόσμο του Android Development. Ακόμα κι αν δεν εμπνεύσει κάποιον η ανάπτυξη εφαρμογών, τουλάχιστον θα δώσει σίγουρα μια γενική εικόνα του τρόπου με τον οποίο λειτουργεί μια τυπική εφαρμογή Android.

6.1 Συμπεράσματα

Η ανάπτυξη εφαρμογών Android, δεν αποτελεί μια δύσκολη διαδικασία, ιδιαίτερα σε άτομα αρκετά εξοικειωμένα με την έννοια της αντικειμενοστρέφειας και της χρήσης της γλώσσας Java. Μια εφαρμογή όπως είναι το Quick Speak, μπορεί να βοηθήσει τον χρήστη σε μια ανάγκη του, όπως αυτή της εκμάθησης μιας ξένης γλώσσας. Απλές ή ακόμα και πιο πολύπλοκες (τουλάχιστον στα πλαίσια της ανάπτυξης τους) εφαρμογές, αποτελούν σημαντικά εργαλεία στη καθημερινότητα των χρηστών.

6.2 Περαιτέρω Εργασία και Πιθανές Επεκτάσεις του Quick Speak

Το Quick Speak, όντας μια απλή και εκπαιδευτικού κυρίως σκοπού εφαρμογή, μπορεί να βελτιωθεί με διάφορους τρόπους, ακόμα και σε σημείο να αποτελέσει ένα κανονικό λεξικό (αν και βέβαια ξεφεύγουμε κάπως από τον αρχικό σκοπό για τον οποίο δημιουργήθηκε). Στην παρούσα ενότητα, θα δούμε επιπλέον στοιχεία, που μπορούν να προστεθούν ώστε να επεκταθεί το Quick Speak.

6.2.1 Πολυγλωσσικότητα της Εφαρμογής

Ένα τυπικό χαρακτηριστικό των εφαρμογών, είναι η υποστήριξη διαφόρων γλωσσών από αυτές. Έτσι η ίδια εφαρμογή, είναι διαθέσιμη σε διάφορες γλώσσες άρα μπορούν να τη χρησιμοποιήσουν ακόμα περισσότεροι χρήστες. Ας πάμε να δούμε το τρόπο με τον οποίο μπορούμε να υποστηρίξουμε παραπάνω της μιας γλώσσας, σε μια εφαρμογή Android, και συγκεκριμένα στο Quick Speak.

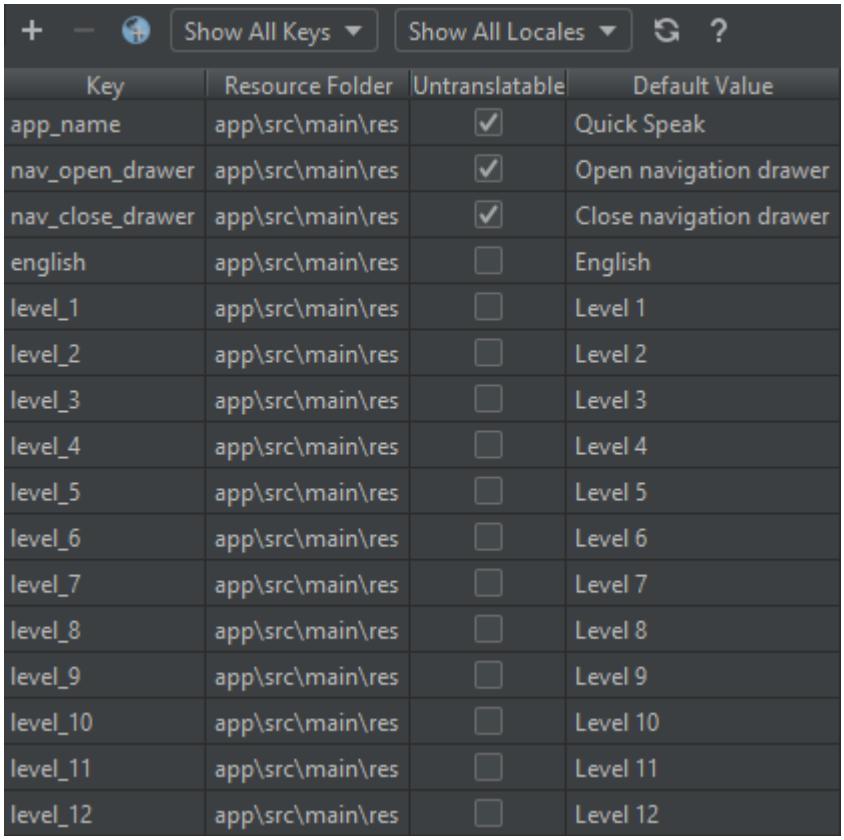
Κατά την ανάπτυξη της εφαρμογής, γνωρίσαμε το αρχείο `strings.xml` και είδαμε το ρόλο του σε αυτήν. Το αρχείο αυτό, όπως είδαμε, περιέχει όλα τα έτοιμα κείμενα που εμφανίζονται στην διεπαφή της (ΠΡΟΣΟΧΗ στη περίπτωση του Quick Speak δεν μιλάμε για τα κείμενα της βάσης, τα λήμματα του λεξικού δηλαδή, καθώς αυτά αποτελούν διαφορετική περίπτωση). Έτσι μπορούμε εύκολα να αλλάξουμε κάποιο κείμενο από το αρχείο αυτό, χωρίς να χρειάζεται επιπλέον τροποποίηση κάποιου άλλου αρχείου της εφαρμογής, όπως κάποιο XML αρχείο διάταξης που το χρησιμοποιεί για παράδειγμα.

Λόγω αυτού, μπορούμε να εκμεταλλευτούμε το αρχείο αυτό, για να υποστηρίξουμε τα κείμενα που περιέχει σε διάφορες γλώσσες. Αυτό πραγματοποιείται με αντίγραφα του strings.xml τα οποία έχουν ακριβώς την ίδια δομή με το αρχικό αρχείο. Συνήθως το αρχικό αρχείο όπως είδαμε, περιέχει τα κείμενα στα Αγγλικά. Η διάφορα που έχουν τα αντίγραφα, είναι η ύπαρξη της τοποθεσίας (locale) της εκάστοτε γλώσσας που θα υποστηριχθεί[131]. Έτσι το Android επιλέγει να συμπεριλάβει στην εφαρμογή, τα κείμενα από το αρχείο που αφορά τη γλώσσα που έχει ενεργό το σύστημα.

Για να φτιάξουμε ένα τέτοιο αντίγραφο, χρησιμοποιούμε τον Translations Editor. Αν ανοίξουμε το αρχείο strings.xml, συνήθως στη κορυφή του εμφανίζεται μια προτροπή, με την οποία μπορούμε να ανοίξουμε τον editor αυτόν.

Εικόνα 27: Ανοίγοντας τον Translations Editor του Android Studio

Ο editor, παρουσιάζεται στο παρακάτω στιγμιότυπο:



Εικόνα 28 : Ο Translations Editor του Android Studio

Όπως φαίνεται και στο παραπάνω στιγμιότυπο, το σύνολο των κειμένων εμφανίζεται σε μορφή λιστών με διάφορα χαρακτηριστικά. Η παραπάνω περίπτωση αφορά τα κείμενα του αρχικού strings.xml αρχείου, δηλαδή αυτού των αγγλικών. Αλλαγές που γίνονται εδώ, γίνονται αυτόματα και στο xml αρχείο και αντιστρόφως. Το key είναι η ονομασία του string που θα χρησιμοποιηθεί. Το resource folder μας δείχνει τη τοποθεσία του strings.xml αρχείου στο project. Αν θέλουμε να δηλώσουμε ένα string με μια μοναδική τιμή που δε θα μεταφράζεται ποτέ (συνήθως ο τίτλος της εφαρμογής), τότε μαρκάρουμε το κουτάκι του Untranslatable. Το Default Value, αφορά φυσικά την προκαθορισμένη τιμή του κάθε string.

Για να προσθέσουμε επιπλέον γλώσσες για τα κείμενα, πατάμε πάνω από τη λίστα το κουμπί  , και επιλέγουμε τη γλώσσα που θέλουμε να υποστηρίξουμε. Στη περίπτωση μας, θέλουμε τα ελληνικά, οπότε επιλέγουμε Greek (el). Στη λίστα θα εμφανιστεί μια στήλη με τις τιμές που θα προσθέσουμε για κάθε αντίστοιχο κείμενο των ελληνικών keys. Αυτό είναι και η μόνη αλλαγή που χρειάζεται να γίνει ώστε να υποστηριχτεί η γλώσσα που επιλέξαμε. Το Android Studio, δημιουργεί αυτόματα το αρχείο με τα ελληνικά κείμενα (ή γενικά οποιασδήποτε άλλης γλώσσας), σε ένα φάκελο, με όνομα `values-όνομα-locale` όπου το όνομα-locale, αφορά τη τοποθεσία της γλώσσας που προστέθηκε. Στη περίπτωση μας ο φάκελος αυτός, έχει όνομα `values-el-rGR`. Κάθε αρχείο κειμένων, έχει πάντα το όνομα strings.xml. Το Android, αναλαμβάνει να επιλέξει το αντίστοιχο αρχείο που θα χρησιμοποιείται κάθε φορά, βάσει της γλώσσας που χρησιμοποιεί το σύστημα. Αν το σύστημα δηλαδή είναι στα ελληνικά, τότε θα χρησιμοποιηθεί το strings.xml που βρίσκεται στο φάκελο `values-el-rGR`, διαφορετικά θα χρησιμοποιηθεί το αρχικό, που βρίσκεται στο φάκελο `values`. Με τον ίδιο τρόπο, μπορούμε να προσθέσουμε οποιαδήποτε άλλη επιπλέον γλώσσα. Παρακάτω

παρουσιάζεται το αρχείο strings.xml που αφορά την ελληνική γλώσσα και είναι αυτό που χρησιμοποιήθηκε στο τελικό project του συνοδευτικού υλικού της πτυχιακής.

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="english">Αγγλικά</string>
    <string name="level_1">Επίπεδο 1</string>
    <string name="level_2">Επίπεδο 2</string>
    <string name="level_3">Επίπεδο 3</string>
    <string name="level_4">Επίπεδο 4</string>
    <string name="level_5">Επίπεδο 5</string>
    <string name="level_6">Επίπεδο 6</string>
    <string name="level_7">Επίπεδο 7</string>
    <string name="level_8">Επίπεδο 8</string>
    <string name="level_9">Επίπεδο 9</string>
    <string name="level_10">Επίπεδο 10</string>
    <string name="level_11">Επίπεδο 11</string>
    <string name="level_12">Επίπεδο 12</string>
</resources>
```

Στο Quick Speak, χρειάστηκε να προστεθεί επιπλέον κώδικας, ο οποίος μετατρέπει τους ελληνικούς τίτλους της λίστας του menu, έτσι ώστε να χρησιμοποιούνται οι αγγλικοί για τα ερωτήματα της βάσης. Αυτό φυσικά δεν είναι ορατό στον χρήστη, καθώς βλέπει κανονικά την εφαρμογή στη γλώσσα των ελληνικών, εφόσον την έχει ενεργή στη συσκευή του, διαφορετικά χρησιμοποιείται η προκαθορισμένη γλώσσα, η οποία είναι τα αγγλικά. Χωρίς να αναλυθεί ιδιαίτερα ο κώδικας, η λογική έχει ως εξής. Αφού εμφανιστούν όλα τα κείμενα της επιπλέον γλώσσας -εδώ των ελληνικών- μετατρέπουμε όλα τα κείμενα των επιπέδων στα αντίστοιχα κείμενα που είναι διαθέσιμα στη βάση (Level 1, Level 2 κτλ.). Ο λόγος που γίνεται αυτό, είναι φυσικά επειδή σε διαφορετική περίπτωση, τα ερωτήματα δεν θα επέστρεφαν κάποιο αποτέλεσμα με τα ελληνικά κείμενα, και άρα δεν θα εμφανιζόταν καμία λίστα λημμάτων για κάθε επίπεδο. Η μετατροπή αυτή, γίνεται κρατώντας μια ακέραια μεταβλητή, και αντιστοιχίζοντας την με το εκάστοτε id της θέσης του στοιχείου του menu που επιλέχθηκε (1 για το Level 1, 2 για το Level 2 κτλ.) . Έπειτα και εφόσον έχει εμφανιστεί ήδη ο τίτλος του επιλεγμένου επιπέδου, αλλάζουμε τιμή στη μεταβλητή `itemTitle`, ως "Level " + τον αριθμό που αντιστοιχίσαμε ανάλογα με το id του στοιχείου που επιλέχθηκε από το menu. Αν για παράδειγμα έχει επιλεχθεί το 5ο επίπεδο, τότε θα εμφανιστεί το κείμενο «Επίπεδο 5» που είναι η τιμή της `itemTitle` (βλέπε `WordListActivity.java`) και έπειτα θα μετατραπεί σε "Level 5" για να χρησιμοποιηθεί στην αναζήτηση των λημμάτων 5ου επιπέδου από τη βάση της εφαρμογής.

Όλο το παραπάνω, αφορά μια περίπτωση του Quick Speak. Γενικά για την υποστήριξη μιας γλώσσας χρειαζόμαστε απλά ένα επιπλέον αντίγραφο του strings.xml αρχείου με τα αντίστοιχα κείμενα της γλώσσας που επιθυμούμε. Παρακάτω παρουσιάζεται η τελική μορφή του αρχείου `WordListActivity.java`, με τις αλλαγές που έγιναν, για να είναι διαθέσιμη και η ελληνική γλώσσα. Να σημειωθεί ότι στο μπλοκ από το οποίο, ανακτούμε τη μεταβλητή `itemTitle` για να διατηρούμε τη κατάσταση της, δημιουργήσαμε ένα προσωρινό fragment. Ο λόγος που έγινε αυτό, είναι για να διορθώσει το bug με τα διπλότυπα fragments που τοποθετούνται το ένα πάνω στο άλλο. Εξασφαλίζουμε με αυτό το τρόπο, ότι το αρχικό fragment

θα δημιουργείται μόνο μια φόρα μέτα την εκκίνηση της εφαρμογής και σε καμία άλλη περίπτωση.

```
package com.selgo.quickspeak;


import android.os.Bundle;
import android.support.design.widget.NavigationView;
import android.support.v4.app.FragmentTransaction;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;

public class WordListActivity extends AppCompatActivity implements Naviga-
tionView.OnNavigationItemSelectedListener {

    private CharSequence itemTitle;
    private Toolbar toolbar;
    private DrawerLayout drawerLayout;
    private ActionBarDrawerToggle actionBarDrawerToggle;
    private WordFragment wordFragment;
    private Bundle arguments;
    private NavigationView navigationView;
    private FragmentTransaction fragmentTransaction;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_word_list);

        itemTitle = getResources().getString(R.string.level_1);

        if (savedInstanceState != null) {
            itemTitle = savedInstanceState.getCharSequence("itemTitle");
            itemTitle = getResources().getString(R.string.level_1);
            wordFragment = new WordFragment();
        }

        toolbar = (Toolbar) findViewById(R.id.app_toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle(itemTitle);

        drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
        actionBarDrawerToggle = new ActionBarDrawerToggle(this, drawerLay-
out, toolbar, R.string.nav_open_drawer, R.string.nav_close_drawer);
        drawerLayout.addDrawerListener(actionBarDrawerToggle);
        actionBarDrawerToggle.syncState();

        navigationView = (NavigationView) findViewById(R.id.naviga-
tion_view);
        navigationView.setNavigationItemSelectedListener(this);
        navigationView.setCheckedItem(R.id.level_1);

        itemTitle = "Level 1";
        arguments = new Bundle();
        arguments.putCharSequence("itemTitle", itemTitle);
```

```
        if (wordFragment == null) {
            wordFragment = new WordFragment();
            wordFragment.setArguments(arguments);

            fragmentTransaction = getSupportFragmentManager().beginTransac-
tion();
            fragmentTransaction.add(R.id.frag_container, wordFragment);
            fragmentTransaction.commit();
        }

    }

    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        int position = 0;

        itemTitle = item.getTitle();
        getSupportActionBar().setTitle(itemTitle);

        drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);

        item.setChecked(true);
        drawerLayout.closeDrawers();

        switch (item.getItemId()) {
            case R.id.level_1:
                position = 1;
                break;
            case R.id.level_2:
                position = 2;
                break;
            case R.id.level_3:
                position = 3;
                break;
            case R.id.level_4:
                position = 4;
                break;
            case R.id.level_5:
                position = 5;
                break;
            case R.id.level_6:
                position = 6;
                break;
            case R.id.level_7:
                position = 7;
                break;
            case R.id.level_8:
                position = 8;
                break;
            case R.id.level_9:
                position = 9;
                break;
            case R.id.level_10:
                position = 10;
                break;
            case R.id.level_11:
                position = 11;
                break;
            case R.id.level_12:
                position = 12;
```

```
                break;
        }

        itemTitle = "Level " + position;

        arguments = new Bundle();
        arguments.putCharSequence("itemTitle", itemTitle);

        wordFragment = new WordFragment();
        wordFragment.setArguments(arguments);

        fragmentTransaction = getSupportFragmentManager().beginTransac-
tion();
        fragmentTransaction.replace(R.id.frag_container, wordFragment);
        fragmentTransaction.commit();

        return true;
    }

    @Override
    public void onBackPressed() {
        drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);

        if(drawerLayout.isDrawerOpen(GravityCompat.START)) {
            drawerLayout.closeDrawers();
        } else {
            super.onBackPressed();
        }
    }

    @Override
    protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putCharSequence("itemTitle", itemTitle);
    }
}
```

6.2.2 Το Ειδικό Ελληνο-αγγλικό (;) Λεξικό Quick Speak

Στα πλαίσια της πτυχιακής, αναπτύξαμε το Quick Speak έτσι ώστε να είναι ένα ειδικό λεξικό μάθησης των Αγγλικών και μόνο. Μια σημαντική επέκταση της εφαρμογής, είναι αυτή της υποστήριξής μάθησης επιπλέον γλωσσών από αυτή. Ο σκοπός της εφαρμογής παραμένει πάντα φυσικά ο ίδιος. Η λογική είναι να έχει την δυνατότητα ο χρήστης, να επιλέξει κι άλλες γλώσσες πέρα των Αγγλικών. Σε κάθε περίπτωση, το Quick Speak απευθύνεται σε χρήστες της Ελληνικής γλώσσας, γι' αυτό άλλωστε κι όλες οι μεταφράσεις θα γίνονται σε αυτή τη γλώσσα. Η ενότητα αυτή δεν έχει σκοπό τόσο να παρουσιάσει μια πλήρη υλοποίηση της επέκτασης, όσο να παρουσιάσει την ιδέα γύρω από τη οποία θα στηριζόμασταν για να την πραγματοποιήσουμε. Ως επιπλέον γλώσσα ας πούμε ότι θέλαμε να προσθέσουμε τα Ιταλικά.

Η επέκταση αυτή, δείχνει πόσο σημαντική είναι η χρήση βάσεων δεδομένων. Μπορεί να γίνει αρκετά εύκολα χωρίς να χρειάζονται πολύπλοκες αλλαγές στον ήδη υπάρχων κώδικα της εφαρμογής. Εκμεταλλευόμενοι τη βάση δεδομένων που δημιουργήσαμε, μπορούμε να προσθέσουμε επιπλέον δεδομένα που θα αφορούν τα λήμματα των Ιταλικών. Η προσθήκη των δεδομένων είναι απλή, αφού έχει ήδη υλοποιηθεί η λογική με την οποία αυτά προστίθενται στη βάση (βλέπε μέθοδο `insertWord` του `WordDbHelper.java`). Το μόνο επιπλέον στοιχείο που

χρειαζόμαστε, πέραν της προσθήκης των data, είναι η δημιουργία ενός νέου πίνακα που θα διατηρεί τα λήμματα των Ιταλικών, τον οποίο, ας ονομάσουμε `italian`. Όπως είδαμε από τη κλάση `WordContract`, χρησιμοποιούμε σταθερές για να δηλώσουμε το όνομα ενός SQL table και των columns που θα περιέχει, ώστε να μπορούμε να τα χρησιμοποιήσουμε για τη διαχείριση του μέσω SQL ερωτημάτων. Με την ίδια λογική θα δημιουργήσουμε μια νέα εσωτερική κλάση που θα αφορά τη περίπτωση των ιταλικών και συγκεκριμένα τον πίνακα `italian`. Παρακάτω παρουσιάζεται ενδεικτικά ο κώδικας με τον οποίο θα υλοποιούσαμε τα στοιχεία του νέου πίνακα.

```
    public static final class ItalianWordEntry implements BaseColumns {
        public final static String TABLE_NAME = "italian";

        public final static String _ID = BaseColumns._ID;
        public final static String COLUMN_LEVEL = "level";
        public final static String COLUMN_ENGLISH_WORD = "italian_word";
        public final static String COLUMN_TRANSLATED_WORD = "trans-
lated_word";
        public final static String COLUMN_WORD_SOUND_ID = "word_sound_id";
    }
```

Όπως βλέπουμε ο νέος πίνακας, έχει την ίδια δομή με τον `english`. Μερικά από τα στοιχεία παραμένουν ακριβώς τα ίδια, και επαναλαμβάνονται μόνο και μόνο για να διασαφηνίσουν, σε ποια κλάση ανήκει το καθένα. Για παράδειγμα και στις 2 κλάσεις έχουμε μια μεταβλητή `COLUMN_LEVEL` με τιμή «`level`». Παρακάτω θα δούμε το τρόπο με τον οποίο διαφοροποιούνται για κάθε πίνακα.

Η βάση σε αυτό το σημείο, χρειάζεται τροποποίηση. Υπάρχουν όμως χρήστες οι οποίοι πιθανώς να έχουν την εφαρμογή ήδη εγκατεστημένη στη συσκευή τους, αρά και τη βάση της με τη παλαιά της έκδοση (πριν τη τροποποίηση της δηλαδή). Και φυσικά δεν αρκεί να προσθέσουμε κώδικα για την τροποποίηση στην `onCreate` μέθοδο της κλάσης της βάσης (βλέπε `WordDbHelper.java`), διότι αυτό προϋποθέτει, ότι κάθε χρήστης που έχει ήδη εγκατεστημένη την εφαρμογή, θα έπρεπε να την απεγκαταστήσει και εγκαταστήσει από την αρχή, έτσι ώστε να διαγραφτεί η παλαιά έκδοση της βάσης και να αντικατασταθεί με τη νέα τροποποιημένη.

Γίνεται λόγος για έκδοση της βάσης, οπότε θα εκμεταλλευτούμε το αρχικό βήμα που είχε πραγματοποιηθεί για την υλοποίηση της, το οποίο ήταν να οριστεί η μεταβλητή `DATABASE_VERSION` με τιμή 1. Χρειάζεται λοιπόν να γίνει η τροποποίηση της βάσης κι αφού ολοκληρωθεί, να αυξήσουμε τη τιμή αυτή κατά 1, και να γίνει 2. Αυτό θα γίνεται σε κάθε τροποποίηση της βάσης για τον εξής λόγο. Το Android, σε κάθε εκκίνηση της εφαρμογής, ελέγχει για κάθε χρήστη, τη τιμή αυτή ώστε να δει την έκδοση της βάσης που έχει, και να εκτελέσει κώδικα τροποποιήσεων της που αφορά την έκδοση αυτή. Με λίγα λόγια, γίνεται προσπάθεια από το σύστημα, ώστε κάθε χρήστης να έχει την πιο πρόσφατη έκδοση.

Αυτό σημαίνει αναβάθμιση της βάσης, το οποίο ορθώς υπονοεί ότι ο κώδικας που χρειάζεται να εκτελεστεί για τροποποιήσεις, τοποθετείται στη μέθοδο `onUpgrade` (πάλι βλέπε `WordDbHelper.java`). Η μέθοδος αυτή, δέχεται ως ορίσματα, το `SQLiteDatabase` αντικείμενο το οποίο χρησιμοποιείται για εκτέλεση SQL queries στη βάση, τον αριθμό της παλαιάς έκδοσης, και τον αριθμό της καινούργιας έκδοσης. Στο κώδικα υλοποιούμε ελέγχους για κάθε έκδοση και τις κινήσεις που πρέπει να γίνουν στη βάση σε κάθε μια από της εκδόσεις

αυτές. Πριν υλοποιηθεί αυτό, χρειάζεται να αλλάξουμε τη τιμή της `DATABASE_VERSION` με βάση τον αριθμό των αλλαγών που θέλουμε να πραγματοποιηθούν στη βάση. Στη περίπτωση μας ο αριθμός αυτός θα είναι το 2, επειδή στη 1η έκδοση προστέθηκε ο πίνακας `english`, και στη 2η, θα προσθέσουμε το πίνακα `italian`. Να σημειωθεί ότι η αύξηση της τιμής κατά 1, γίνεται από την ίδια την `onUpgrade`.

Η αναβάθμιση λοιπόν γίνεται ως εξής. Αρχικά ελέγχουμε αν είμαστε στην 1η έκδοση. Σε αυτήν απλώς δημιουργούμε το πίνακα `english` όπως γνωρίζουμε, οπότε δεν έχουμε πλέον το κώδικα που υλοποιεί αυτό στην `onCreate`. Έπειτα ελέγχουμε την 2η έκδοση, στην οποία προσθέτουμε το πίνακα italian. Προτείνεται η χρήση custom μεθόδου (ας την ονομάσουμε `updateDatabase`), η οποία θα δέχεται τα 3 ορίσματα της `onUpgrade`. Ο λόγος είναι επειδή, η μέθοδος αυτή θα καλείται αρχικά στην `onCreate` για να δημιουργήσει τον πίνακα, και στην `onUpgrade` όταν θα χρειαστεί τροποποίηση. Σε κάθε περίπτωση η έκδοση της βάσης θα αλλάζει κατάλληλα. Στην `onCreate` το όρισμα της παλαιάς βάσης ορίζεται με 0. Ο πρώτος έλεγχος και για διασαφηνιστεί το πρώτο βήμα, ελέγχει τη περίπτωση όπου δεν έχουμε φτάσει ακόμα στη 1η έκδοση, άρα φτιάχνουμε για πρώτη φορά το πίνακα `english`. Αφού πραγματοποιηθεί αυτό, η έκδοση της βάσης γίνεται 1. Ο επόμενος έλεγχος της `updateDatabase` είναι η περίπτωση της βάσης όπου δεν έχουμε φτάσει ακόμα στη 2η έκδοση. Εδώ φυσικά προσθέτουμε το πίνακα italian, χρησιμοποιώντας τις σταθερές της κλάσης `ItalianWordEntry`. Έπειτα προσθέτουμε τα λήμματα των Ιταλικών με τη χρήση της `insertWord`, όπως ήδη γνωρίζουμε δηλαδή. Έτσι η έκδοση της βάσης είναι 2 και έχουμε τελειώσει με την αναβάθμιση. Όλη αυτή η λογική εξασφαλίζει στην ουσία ότι, αν κάποιος είναι για πρώτη φορά χρήστης του Quick Speak, τότε θα δημιουργηθούν οι πίνακες `english` και `italian`, διαφορετικά αν ήδη υπάρχει ο πίνακας `english`, τότε απλώς θα προστεθεί ο επιπλέον πίνακας `italian` μαζί με τα δεδομένα του στη βάση. Και στις 2 περιπτώσεις, το αποτέλεσμα είναι ότι οι χρήστες πλέον έχουν διαθέσιμα τα δεδομένα των Ιταλικών στην εφαρμογή τους. Παρακάτω παρουσιάζεται ένα παράδειγμα πιθανής τροποποίησης του κώδικα της κλάσης `WordDbHelper`. Για τη χρήση των σταθερών της κλάσης `ItalianWordEntry` χρειάζεται η εισαγωγή (import) του πακέτου της κλάσης δηλαδή

```
import com.selgo.quickspeak.data.WordContract.ItalianWordEntry;
```

Ορισμός της updateDatabase

```
private void updateDatabase(SQLiteDatabase db, int oldVersion, int newVer-
sion) {
    if (oldVersion < 1) {
        // Κώδικας για τη δημιουργία του πίνακα english με χρήση των σταθε-
ρών WordEntry ( πχ WordEntry.LEVEL )
            String SQL_CREATE_ENGLISH_TABLE = "CREATE TABLE " +
            WordEntry.TABLE_NAME + "("
                + WordEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + WordEntry.COLUMN_LEVEL + " TEXT NOT NULL, "
                + WordEntry.COLUMN_ENGLISH_WORD + " TEXT, "
                + WordEntry.COLUMN_TRANSLATED_WORD + " TEXT, "
                + WordEntry.COLUMN_WORD_SOUND_ID + " TEXT);";

        db.execSQL(SQL_CREATE_ENGLISH_TABLE);
    }

    if (oldVersion < 2) {
```

```
        // Κώδικας για τη δημιουργία του πίνακα italian με χρήση των σταθε-
ρών ItalianWordEntry ( πχ ItalianWordEntry.LEVEL )
            String SQL_CREATE_ITALIAN_TABLE = "CREATE TABLE " +
            ItalianWordEntry.TABLE_NAME + "("
            + ItalianWordEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
            + ItalianWordEntry.COLUMN_LEVEL + " TEXT NOT NULL, "
            + ItalianWordEntry.COLUMN_ENGLISH_WORD + " TEXT, "
            + ItalianWordEntry.COLUMN_TRANSLATED_WORD + " TEXT, "
            + ItalianWordEntry.COLUMN_WORD_SOUND_ID + " TEXT);";

      db.execSQL(SQL_CREATE_ITALIAN_TABLE);
    }
}
```

Χρήση της μεθόδου

```
public void onCreate(SQLiteDatabase db){
    updateDatabase(db, 0, DB_VERSION); // Θεωρώντας οτι DB_VERSION = 2
}


public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    updateDatabase(db, oldVersion, newVersion);
}
```

6.2.2.1 Υποστήριξη Ιταλικών στην Διεπαφή του Quick Speak

Η ολοκλήρωση της κεντρική ιδέας των επιπλέον υποστηριζόμενων γλωσσών, γίνεται με την ανάπτυξη της ιδέας αυτής και στο κομμάτι της διεπαφής του Quick Speak. Ο χρήστης έχει τη δυνατότητα να επιλέγει τη γλώσσα που επιθυμεί να μάθει. Έτσι θα του παρουσιάζονται τα λήμματα της επιλεγμένης γλώσσας, πάντα χωρισμένα ανά επίπεδα.

Μπορούμε να εκμεταλλευτούμε το toolbar της εφαρμογής για να υποστηρίξουμε την δυνατότητα αυτή. Ο χρήστης θα μπορεί να επιλέγει την γλώσσα που τον ενδιαφέρει, μέσω ενός menu - όχι το navigation menu – το οποίο θα παρουσιάζει τη λίστα με τις διαθέσιμες γλώσσες.

Τα toolbars, έχουν τη δυνατότητα να συμπεριλαμβάνουν αντικείμενα, τα οποία πραγματοποιούν ενέργειες, τα λεγόμενα actions. Τα αντικείμενα αυτά είναι συνήθως κουμπιά με τα οποία μπορούμε να υποστηρίξουμε ενέργειες, όπως σύνταξη νέου μηνύματος, δημοσίευση άρθρου κτλ. Κάθε ένα από αυτά τα αντικείμενα έχει δύο καταστάσεις σε ένα toolbar. Μπορεί να είναι ορατό με ένα εικονίδιο, εφόσον υπάρχει χώρος στο toolbar, διαφορετικά τοποθετούνται σε ένα κρυμμένο αρχικά menu, το λεγόμενο overflow menu[132], το οποίο ενεργοποιείται μέσω ενός ειδικού κουμπιού, το οποίο συνήθως φέρει αυτής της μορφής  .

Τα αντικείμενα, τοποθετούνται ένα προς ένα, δεξιά προς τα αριστερά, με τέτοιο τρόπο ώστε να υπάρχει όσο είναι δυνατόν, περισσότερος χώρος για το τίτλο του activity.

Θα πάμε να δούμε την ιδέα, με την οποία μπορούμε να υλοποιήσουμε ένα τέτοιου είδους menu στο Quick Speak. Αρχικά, μιλάμε για menu, οπότε αυτό, υπονοεί τη χρήση ενός menu resource αρχείου, όπως αυτό που είδαμε και στην υλοποίηση του navigation menu. Στη

περίπτωση των γλωσσών, χρειαζόμαστε ένα μενού 2 στοιχείων, ένα για την επιλογή των Αγγλικών και ένα για αυτή των Ιταλικών. Μια πιθανή μορφή του αρχείου αυτού παρουσιάζεται παρακάτω.

languages_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">
        <item
            android:id="@+id/english"
            android:icon="@mipmap/united_kingdom_flag_round_xs"
            android:title="@string/english"
            app:showAsAction="never" />
        <item
            android:id="@+id/italian"
            android:icon="@mipmap/italy_flag_round_xs"
            android:title="@string/italian"
            app:showAsAction="never" />
</menu>
```

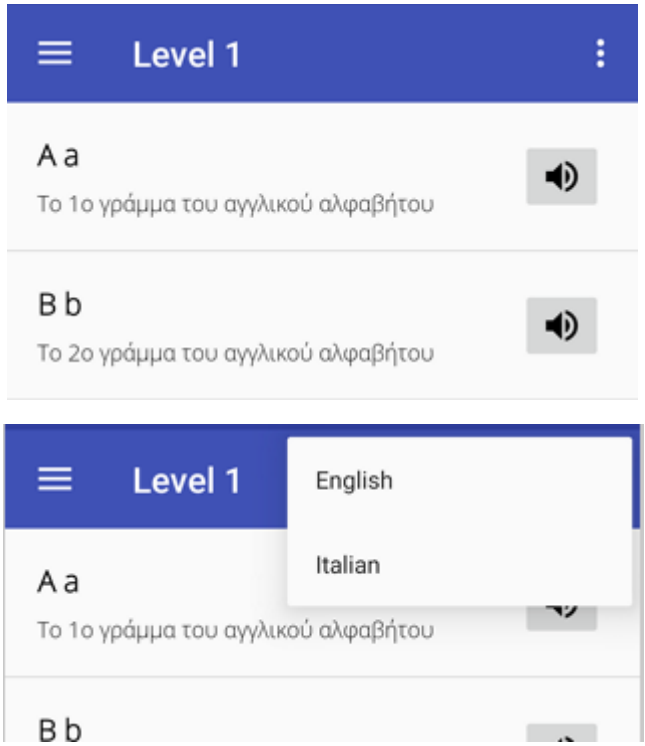
Θεωρούμε φυσικά ότι για την περίπτωση των Ιταλικών, ότι είναι διαθέσιμα και το εικονίδιο της γλώσσας και το αντίστοιχο string για το τίτλο της επιλογής μεταφρασμένο και στα Ελληνικά. Έχοντας αυτό το αρχείο, μένει να προχωρήσουμε στο προγραμματιστικό κομμάτι της υλοποίησης του menu.

Χρειάζεται η προσθήκη δύο επιπλέον μεθόδων, στο `WordListActivity.java`. Η πρώτη είναι η `onCreateOptionsMenu`. Με αυτή τη μέθοδο ορίζεται το menu resource file – το υποτιθέμενο `languages_menu.xml` - που θα χρησιμοποιηθεί για να δημιουργηθεί το menu. Ο κώδικας με τον οποίο υλοποιείται αυτό, θα μπορούσε να έχει ως εξής:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.languages_menu, menu);
    return true;
}
```

Η δεύτερη μέθοδος, έχει να κάνει με τη λειτουργικότητα του menu. Με λίγα λόγια, σε αυτήν, υλοποιούμε τις ενέργειες που θα κάνει η εφαρμογή για κάθε επιλογή που επιλέγει ο χρήστης. Η ιδέα της υποστήριξής των γλωσσών θα έχει την εξής λογική. Πατώντας πάνω δεξιά το κουμπί που αναφέραμε πριν, ο χρήστης επιλέγει μια γλώσσα. Όταν το πραγματοποιήσει αυτό, η εφαρμογή κάνει μετάβαση στα αντίστοιχα λήμματα του πρώτου επιπέδου αρχικά, και έπειτα με την λογική του Quick Speak όπως τη γνωρίζουμε να επιλέγει και να βλέπει τα λήμματα κάθε επιπέδου για τη γλώσσα που επέλεξε. Σε κάθε περίπτωση δηλαδή, αλλάζουν μόνο τα δεδομένα κι όχι η λογική με την οποία λειτουργεί η εφαρμογή. Έτσι για να ολοκληρώσουμε, με την ιδέα, παρουσιάζουμε τα σημεία σε ενδεικτικό κώδικα που μπορούν υλοποιήσουν τη λειτουργικότητα της, και ένα στιγμιότυπο το οποίο δίνει τη τελική εικόνα την οποία έχει ο χρήστης.

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        Toast testToast;

        switch(item.getItemId()) {
            case R.id.english:
                // Ενδεικτική ενέργεια όταν ο χρήστης επιλέξει τα Αγγλικά
                testToast = Toast.makeText(this, "English Selected",
Toast.LENGTH_SHORT);
                testToast.show();
                return true;
            case R.id.italian:
                // Ενδεικτική ενέργεια όταν ο χρήστης επιλέξει τα Ιταλικά
                testToast = Toast.makeText(this, "Italian Selected",
Toast.LENGTH_SHORT);
                testToast.show();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
```



Εικόνες 29 και 30 : Το Overflow Menu του Quick Speak

6.3 Επίλογος

Σε αυτό το σημείο ολοκληρώνουμε με τη παρούσα πτυχιακή. Η εφαρμογή του Quick Speak και οι τεχνικές ανάπτυξης του, αποτελούν ένα σημείο από το οποίο, μπορεί κάποιος να προχωρήσει σε προχωρημένες γνώσεις που αφορούν την ανάπτυξη Android εφαρμογών. Στο παράρτημα Α, παρουσιάζεται το λογισμικό που χρησιμοποιήθηκε, για τη δημιουργία των ηχητικών αποσπασμάτων που χρησιμοποιήθηκαν για την εφαρμογή.

Βιβλιογραφία

[1] "Android", Wikipedia, 2019 https://en.wikipedia.org/wiki/Android_(operating_system)

[2] "Google", Wikipedia, 2019 https://en.wikipedia.org/wiki/Google

[3] "Google Products List", Wikipedia, 2019 https://en.wikipedia.org/wiki/List_of_Google_products

[4] "Platform Architecture", Android Developers, 2019 https://developer.android.com/guide/platform

[5] "Linux Kernel", Wikipedia, 2019 https://en.wikipedia.org/wiki/Linux_kernel

[6] "The Linux Kernel Archives", Ιστότοπος, 2019 https://www.kernel.org/

[7] "ART and DALVIK", Android Open Source Project, 2019 https://source.android.com/devices/tech/dalvik/index.html

[8] "Application Programming Interface", Wikipedia, 2019 https://en.wikipedia.org/wiki/Application_programming_interface

[9] "Android Software Development", Wikipedia, 2019 https://en.wikipedia.org/wiki/Android_software_development

[10] "C Programming Language", Wikipedia, 2019 https://en.wikipedia.org/wiki/C%2B%2B

[11] "Java", Ιστότοπος, 2019 https://www.java.com/en/

[12] "Java Software Platform", Wikipedia, 2019 https://en.wikipedia.org/wiki/Java_(software_platform)

[13] "Kotlin", Ιστότοπος, 2019 https://kotlinlang.org/

[14] "Kotlin Programming Language", Wikipedia, 2019 https://en.wikipedia.org/wiki/Kotlin_(programming_language)

[15] "Android SDK", Wikipedia, 2019 https://en.wikipedia.org/wiki/Android_software_development#Android_SDK

[16] "Android Application Package", Wikipedia, 2019 https://en.wikipedia.org/wiki/Android_application_package#cite_note-1

[17] "Application Fundamentals", Android Developers, 2019 https://developer.android.com/guide/components/fundamentals

[18] "Virtual Machine", Wikipedia, 2019 https://en.wikipedia.org/wiki/Virtual_machine

[19] "Intents and Intent Filters", Android Developers, 2019 https://developer.android.com/guide/components/intents-filters

[20] "ContentResolver", Android Developers, 2019 https://developer.android.com/reference/kotlin/android/content/ContentResolver?hl=en

[21] "Object-Oriented Programming", Wikipedia, 2019 https://en.wikipedia.org/wiki/Object-oriented_programming

[22] "Porting", Wikipedia, 2019 https://en.wikipedia.org/wiki/Porting

[23] "Port Definition", Margaret Rouse, 2005 https://searchnetworking.techtarget.com/definition/port

[24] "XML", Ιστότοπος, 2019 https://www.w3schools.com/xml/xml_whatis.asp

[25] "XML", Wikipedia, 2019 https://en.wikipedia.org/wiki/XML

[26] "HTML", Ιστότοπος, 2019 https://www.w3schools.com/htmL/html_intro.asp

[27] "HTML", Wikipedia, 2019 https://en.wikipedia.org/wiki/HTML

[28] "XML - Parsing", Wikipedia, 2019 https://en.wikipedia.org/wiki/XML#Pull_parsing

[29] "Integrated Development Environment", Wikipedia, 2019 https://en.wikipedia.org/wiki/Integrated_development_environment

[30] "Android Studio", Ιστότοπος, 2019 https://developer.android.com/studio

[31] "Android Studio", Wikipedia, 2019 https://en.wikipedia.org/wiki/Android_Studio

[32] "What is Class Diagram", Ιστότοπος, 2019 https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/

[33] "UML Class Diagram Tutorial", Ιστότοπος, 2019 https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/

[34] "Visual Paradigm", Ιστότοπος, 2019 https://www.visual-paradigm.com/

[35] "Android SDK", Wikipedia, 2019 https://en.wikipedia.org/wiki/Android_software_development#Android_SDK

[36] "Software Development Kit", Wikipedia, 2019 https://en.wikipedia.org/wiki/Software_development_kit

[37] "Eclipse IDE", Ιστότοπος, 2019 https://www.eclipse.org/ide/

[38] "Eclipse Software", Wikipedia, 2019 https://en.wikipedia.org/wiki/Eclipse_(software)

[39] "ADT Plugin for Eclipse", Ιστότοπος, 2019 http://www.dre.vanderbilt.edu/~schmidt/android/android-4.0/out/target/common/docs/doc-comment-check/sdk/eclipse-adt.html

[40] "NetBeans IDE", Ιστότοπος, 2019 https://netbeans.org/

[41] "NetBeans", Wikipedia, 2019 https://en.wikipedia.org/wiki/NetBeans

[42] "IntelliJ IDEA", Ιστότοπος, 2019 https://www.jetbrains.com/idea/

[43] "IntelliJ IDEA", Wikipedia, 2019 https://en.wikipedia.org/wiki/IntelliJ_IDEA

[44] “Introduction to Activities”, Android Developers, 2019 https://developer.android.com/guide/components/activities/intro-activities

[45] “Layouts”, Android Developers, 2019 https://developer.android.com/guide/topics/ui/declaring-layout

[46] “Activity”, Android Developers, 2019 https://developer.android.com/reference/android/app/Activity

[47] “Android Visualizer”, Ιστότοπος, 2019 https://labs.udacity.com/android-visualizer/

[48] “Understand the Activity Lifecycle”, Android Developers, 2019 https://developer.android.com/guide/components/activities/activity-lifecycle

[49] “Handle Configuration Changes”, Android Developers, 2019 https://developer.android.com/guide/topics/resources/runtime-changes

[50] “ListView”, Android Developers, 2019 https://developer.android.com/reference/android/widget/ListView

[51] “Adapter”, Android Developers, 2019 https://developer.android.com/reference/android/widget/Adapter

[52] “ArrayAdapter”, Android Developers, 2019 https://developer.android.com/reference/android/widget/ArrayAdapter

[53] “CursorAdapter”, Android Developers, 2019 https://developer.android.com/reference/android/widget/CursorAdapter

[54] “Fragments”, Android Developers, 2019 https://developer.android.com/guide/components/fragments

[55] “Fragment”, Android Developers, 2019 https://developer.android.com/reference/androidx/fragment/app/Fragment.html

[56] “TTSAutomate”, Ιστότοπος, 2019 https://www.ttsautomate.com/

[57] “Codenames, Tags and Build Numbers”, Android Open Source Project, 2019 https://source.android.com/setup/start/build-numbers

[58] “JUnit 5”, Ιστότοπος, 2019 https://junit.org/junit5/

[59] “JUnit”, Wikipedia, 2019 https://en.wikipedia.org/wiki/JUnit

[60] “Gradle”, Ιστότοπος, 2019 https://gradle.org

[61] “Gradle”, Wikipedia, 2019 https://en.wikipedia.org/wiki/Gradle

[62] “Android Documentation”, Android Developers, 2019 https://developer.android.com/docs

[63] “Relative Layout”, Android Developers, 2019 https://developer.android.com/guide/topics/ui/layout/relative

[64] “RelativeLayout”, Android Developers, 2019 https://developer.android.com/reference/android/widget/RelativeLayout.html

[65] “Linear Layout”, Android Developers, 2019 https://developer.android.com/guide/topics/ui/layout/linear

[66] “LinearLayout”, Android Developers, 2019 https://developer.android.com/reference/android/widget/LinearLayout

[67] “Build a Responsive UI with ConstraintLayout”, Android Developers, 2019 https://developer.android.com/training/constraint-layout

[68] “ConstraintLayout” Android Developers, 2019 https://developer.android.com/reference/android/support/constraint/ConstraintLayout

[69] “Support different pixel densities”, Android Developers, 2019 https://developer.android.com/training/multiscreen/screendensities

[70] “Activity - onCreate”, Android Developers, 2019 https://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)

[71] “Activity - setContentView”, Android Developers, 2019 https://developer.android.com/reference/android/app/Activity.html#setContentView(int)

[72] “R”, Android Developers, 2019 https://developer.android.com/reference/android/R

[73] “Understanding R.java”, Ιστότοπος, 2014 http://knowledgefolders.com/akc/display?url=displaynoteimpurl&ownerUserId=satya&reportId=2883

[74] “Handler”, Android Developers, 2019 https://developer.android.com/reference/android/os/Handler

[75] “Handler - postDelayed”, Android Developers, 2019 https://developer.android.com/reference/android/os/Handler.html#postDelayed(java.lang.Runnable,%20long)

[76] “Runnable”, Android Developers, 2019 https://developer.android.com/reference/java/lang/Runnable.html

[77] “Class”, Oracle Docs, 2019 https://docs.oracle.com/javase/8/docs/api/java/lang/Class.html

[78] “AppCompatActivity”, Android Developers, 2019 https://developer.android.com/reference/android/support/v7/app/AppCompatActivity

[79] “DrawerLayout”, Android Developers, 2019 https://developer.android.com/reference/android/support/v4/widget/DrawerLayout

[80] “App Bars Top”, Ιστότοπος, 2019 https://material.io/design/components/app-bars-top.html

[81] “Toolbar”, Android Developers, 2019 https://developer.android.com/reference/android/widget/Toolbar

[82] “android.database.sqlite”, Android Developers, 2019
https://developer.android.com/reference/android/database/sqlite/package-summary

[83] “Save Data Using SQLite”, Android Developers, 2019
https://developer.android.com/training/data-storage/sqlite

[84] “BaseColumns”, Android Developers, 2019
https://developer.android.com/reference/android/provider/BaseColumns.html

[85] “SQLiteOpenHelper”, Android Developers, 2019
https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper

[86] “SQLiteOpenHelper - onUpgrade”, Android Developers, 2019
https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html#onUpgrade(android.database.sqlite.SQLiteDatabase,%20int,%20int)

[87] “SQLiteOpenHelper - onCreate”, Android Developers, 2019
https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html#onCreate(android.database.sqlite.SQLiteDatabase)

[88] “SQLiteDatabase - execSQL”, Android Developers, 2019
https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html#execSQL(java.lang.String)

[89] “Method Overloading in Java with Examples”,Chaitanya Singh, Ιστότοπος, 2019
https://beginnersbook.com/2013/05/method-overloading/

[90] “ContentValues”, Android Developers, 2019
https://developer.android.com/reference/android/content/ContentValues

[91] “CursorAdapter”, Android Developers, 2019
https://developer.android.com/reference/android/widget/CursorAdapter

[92] “Populating a ListView with a CursorAdapter”, Android Developers, 2019
https://guides.codepath.com/android/Populating-a-ListView-with-a-CursorAdapter

[93] “Cursor”, Android Developers, 2019
https://developer.android.com/reference/android/database/Cursor

[94] “CursorAdapter - newView”, Android Developers, 2019
https://developer.android.com/reference/android/widget/CursorAdapter.html#newView(android.content.Context,%20android.database.Cursor,%20android.view.ViewGroup)

[95] “CursorAdapter - bindView”, Android Developers, 2019
https://developer.android.com/reference/android/widget/CursorAdapter.html#bindView(android.view.View,%20android.content.Context,%20android.database.Cursor)

[96] “MediaPlayer”, Android Developers, 2019
https://developer.android.com/reference/android/media/MediaPlayer

[97] “MediaPlayer Overview”, Android Developers, 2019
https://developer.android.com/guide/topics/media/mediaplayer

[98] “MediaPlayer - create”, Android Developers, 2019 https://developer.android.com/reference/android/media/MediaPlayer.html#create(android.content.Context,%20android.net.Uri)

[99] “MediaPlayer - start”, Android Developers, 2019 https://developer.android.com/reference/android/media/MediaPlayer.html#start()

[100] “MediaPlayer - release”, Android Developers, 2019 https://developer.android.com/reference/android/media/MediaPlayer.html#release()

[101] “MediaPlayer.OnCompletionListener”, Android Developers, 2019 https://developer.android.com/reference/android/media/MediaPlayer.OnCompletionListener.html

[102] “View.OnClickListener”, Android Developers, 2019 https://developer.android.com/reference/android/view/View.OnClickListener.html

[103] “ImageButton”, Android Developers, 2019 https://developer.android.com/reference/android/widget/ImageButton

[104] “Fragment - onCreateView”, 2019 https://developer.android.com/reference/android/app/Fragment.html#onCreateView(android.view.LayoutInflater,%20android.view.ViewGroup,%20android.os.Bundle)

[105] “Fragment – onStart”, Android Developers, 2019 https://developer.android.com/reference/android/app/Fragment.html#onStart()

[106] “Fragment - getView”, Android, 2019 https://developer.android.com/reference/android/app/Fragment.html#getView()

[107] “SQLiteOpenHelper - getReadableDatabase”, Android Developers, 2019 https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html#getReadableDatabase()

[108] “SQLiteDatabase - query”, Android Developers, 2019 https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html#query(java.lang.String,%20java.lang.String[],%20java.lang.String,%20java.lang.String[],%20java.lang.String,%20java.lang.String,%20java.lang.String)

[109] “Concatenation”, Wikipedia, 2019 https://en.wikipedia.org/wiki/Concatenation

[110] “The catch Blocks”, Oracle Docs, Ιστότοπος, 2019 https://docs.oracle.com/javase/tutorial/essential/exceptions/catch.html

[111] “SQLException”, Android Developers, 2019 https://developer.android.com/reference/android/database/SQLException

[112] “Toast”, Android Developers, 2019 https://developer.android.com/reference/android/widget/Toast.html

[113] “Toasts overview”, Android Developers, 2019 https://developer.android.com/guide/topics/ui/notifiers/toasts

[114] “FrameLayout”, Android Developers, 2019
https://developer.android.com/reference/android/widget/FrameLayout

[115] “NavigationView”, Android Developers, 2019
https://developer.android.com/reference/com/google/android/material/navigation/NavigationView.html

[116] “ActionBarDrawerToggle”, Android Developers, 2019
https://developer.android.com/reference/android/support/v4/app/ActionBarDrawerToggle

[117] “NavigationView.OnNavigationItemSelectedListener”, Android Developers, 2019
https://developer.android.com/reference/android/support/design/widget/NavigationView.OnNavigationItemSelectedListener

[118] “Bundle”, Android Developers, 2019
https://developer.android.com/reference/android/os/Bundle

[119] “Fragment - setArguments”, Android Developers, 2019
https://developer.android.com/reference/androidx/fragment/app/Fragment.html#setArguments(android.os.Bundle)

[120] “CharSequence”, Oracle Docs, Ιστότοπος, 2019
https://docs.oracle.com/javase/7/docs/api/java/lang/CharSequence.html

[121] “Bundle - putCharSequence”, Android Developers, 2019
https://developer.android.com/reference/android/os/Bundle.html#putCharSequence(java.lang.String,%20java.lang.CharSequence)

[122] “NavigationView - setCheckedItem”, Android Developers, 2019
https://developer.android.com/reference/android/support/design/widget/NavigationView.html#setCheckedItem(int)

[123] “FragmentTransaction”, Android Developers, 2019
https://developer.android.com/reference/android/app/FragmentTransaction

[124] “FragmentActivity - getSupportFragmentManager”, Android Developers, 2019
https://developer.android.com/reference/android/support/v4/app/FragmentActivity.html#getSupportFragmentManager()

[125] “FragmentManager - beginTransaction”, Android Developers, 2019
https://developer.android.com/reference/android/app/FragmentManager.html#beginTransaction()

[126] “FragmentTransaction - commit”, Android Developers, 2019
https://developer.android.com/reference/android/app/FragmentTransaction.html#commit()

[127] “DrawerLayout - closeDrawers”, Android Developers, 2019
https://developer.android.com/reference/android/support/v4/widget/DrawerLayout.html#closeDrawers()

[128] “FragmentTransaction - replace”, Android Developers, 2019 https://developer.android.com/reference/android/support/v4/app/FragmentTransaction.html#replace(int,%20android.support.v4.app.Fragment,%20java.lang.String)

[129] “Activity - onBackPressed”, Android Developers, 2019 https://developer.android.com/reference/android/app/Activity.html#onBackPressed()

[130] “Activity - onSaveInstanceState”, Android Developers, 2019 https://developer.android.com/reference/android/app/Activity.html#onSaveInstanceState(android.os.Bundle)

[131] “Support different languages and cultures”, Android Developers, 2019 https://developer.android.com/training/basics/supporting-devices/languages

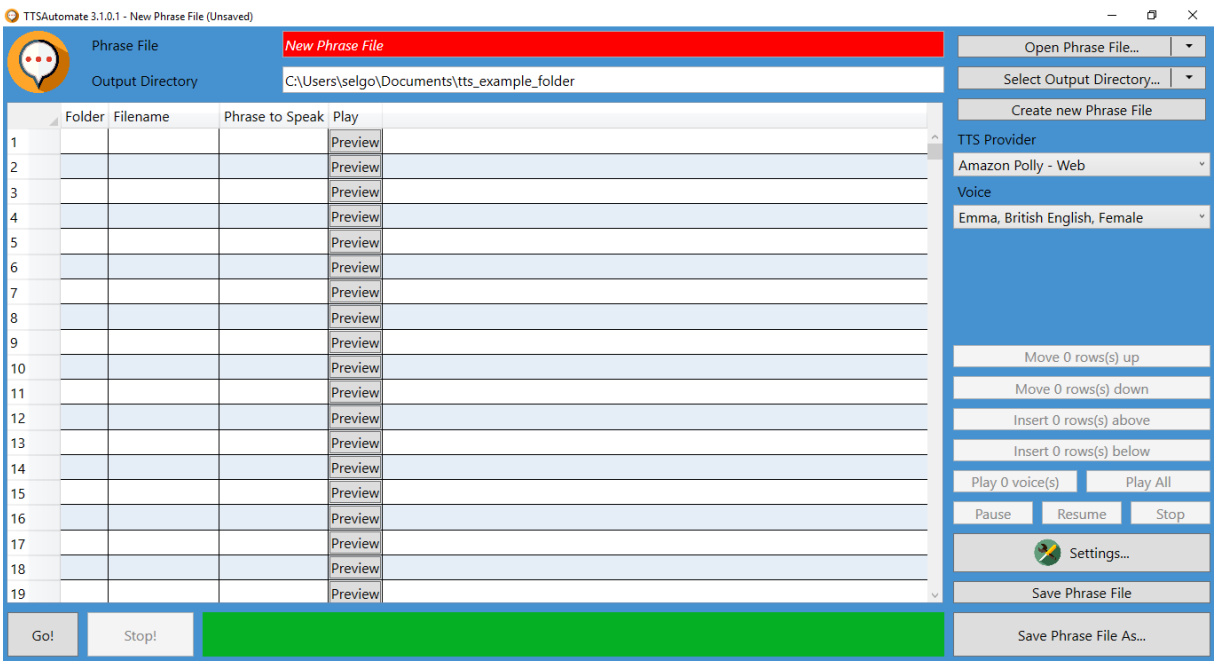
[132] “Creating and Managing Overflow Menus on Android”, Ιστότοπος, 2019 https://www.techotopia.com/index.php/Creating_and_Managing_Overflow_Menus_on_Android

[133] “Head First Android Development, 2nd Edition”, David Griffiths - Dawn Griffiths, 2017 https://www.oreilly.com/library/view/head-first-android/9781491974049/

ΠΑΡΑΡΤΗΜΑ Α: Το TTSAutomate

Η δημιουργία των ηχητικών αποσπασμάτων που χρησιμοποιήθηκαν στο Quick Speak, πραγματοποιήθηκε μέσω του λογισμικού TTSAutomate. Σκοπός του, είναι η δημιουργία ηχητικών αποσπασμάτων μέσω διαβάσματος κειμένου. Τα κείμενα στη περίπτωση του Quick Speak, είναι φυσικά τα λήμματα του λεξικού. Το TSSAutomate, είναι διαθέσιμο στον σύνδεσμο https://www.ttsautomate.com/download.html σε φορητή – portable – ή και μέσω εγκατάστασης, έκδοση και υποστηρίζεται από συστήματα Windows 7 και άνω. Κατά την συγγραφή της πτυχιακής, ο παρόν σύνδεσμος είναι ενεργός. Παρόλα αυτά, στη παρούσα πτυχιακή, χρησιμοποιήθηκε η έκδοση 3.1.0.1, η οποία βρίσκεται σε portable μορφή, στο συνοδευτικό υλικό της πτυχιακής. Ας πάμε να δούμε το τρόπο με τον οποίο εργαστήκαμε για να τη δημιουργία των λημμάτων μέσω του TTSAutomate.

Αφού κάνουμε unzip το αρχείο της portable έκδοσης, εντοπίζουμε στο φάκελο *TTSAutomate.3.1.0.1.Portable*, το αρχείο *TTSAutomate.exe* και το εκτελούμε. Αφού εκτελεστεί το λογισμικό, παρουσιάζεται το βασικό παράθυρο του, όπως παρουσιάζεται και στο παρακάτω στιγμιότυπο:

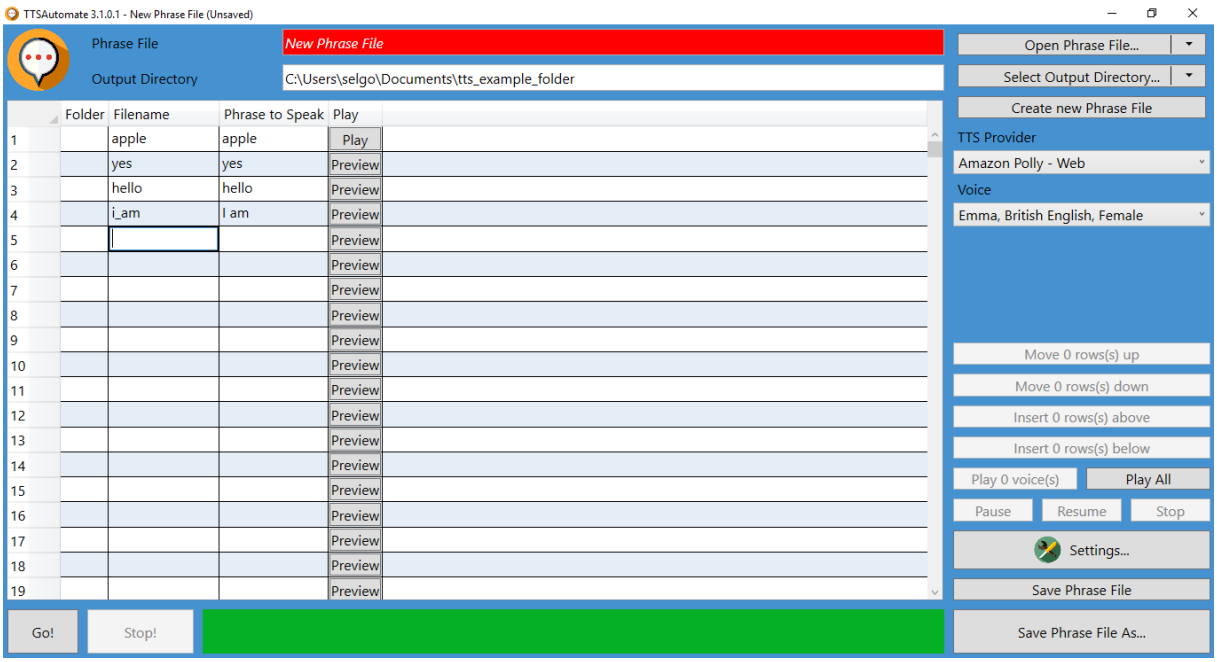


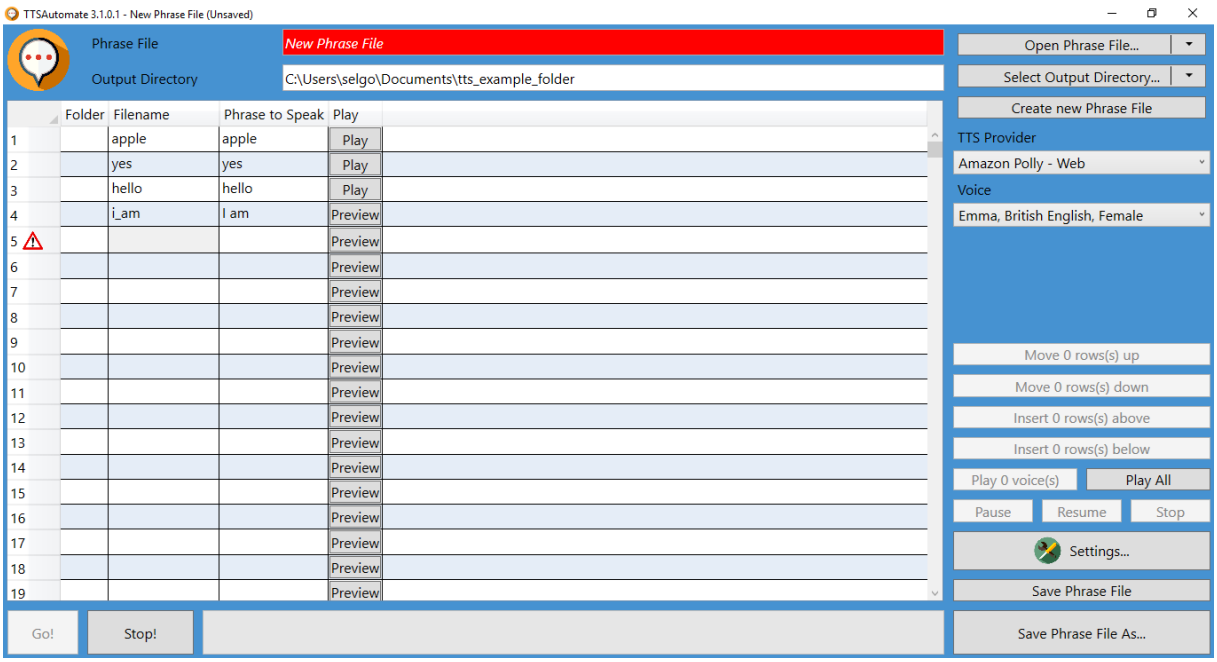
Εικόνα 31: Το TSSAutomate

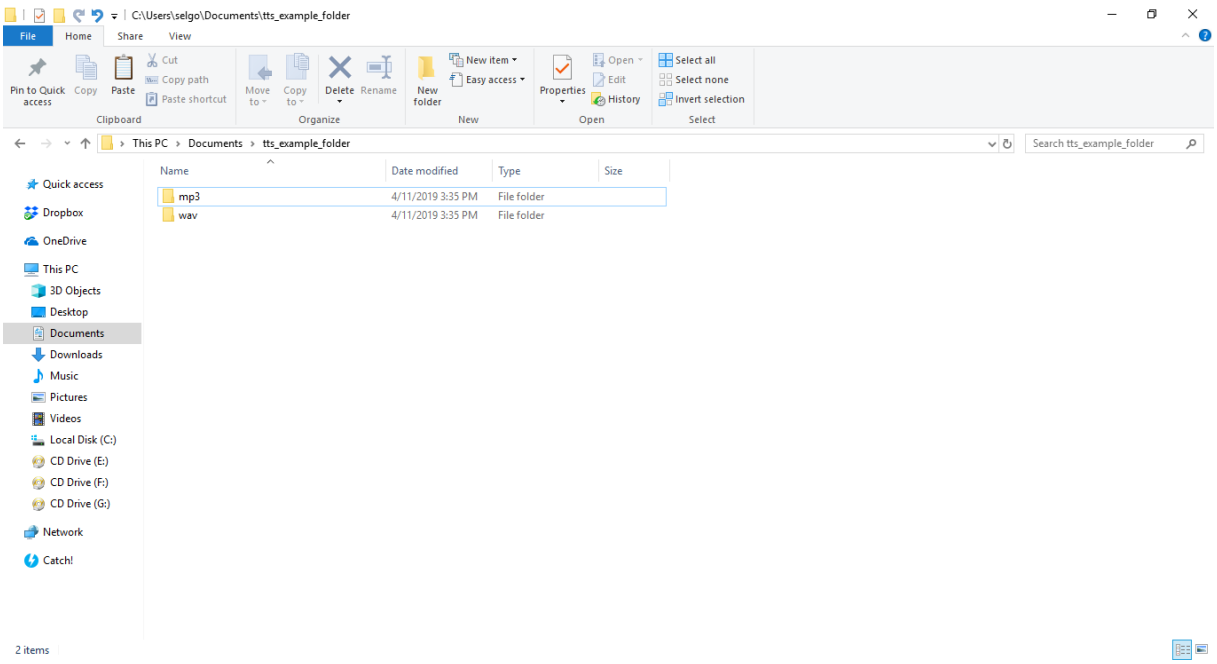
Το λογισμικό λειτουργεί ως εξής. Το κάθε στοιχείο της λίστας που παρουσιάζεται παραπάνω, αφορά ένα ηχητικό απόσπασμα. Η στήλη Folder, αφορά το φάκελο στον οποίο θα αποθηκευτεί το αρχείο ήχου. Η στήλη Filename, αφορά το όνομα του αρχείου, και η Phrase to Speak, το κείμενο το οποίο θα διαβαστεί και θα αποθηκευτεί ως ηχητικό απόσπασμα. Κάθε στοιχείο της λίστας – το σύνολο όλων των αποσπασμάτων δηλαδή – μπορεί να αποθηκευτεί σε ένα αρχείο φράσεων με κατάληξη .psv . Βλέποντας πάνω αριστερά, το κουμπί Open Phrase File, ανοίγει ένα phrase file εφόσον υπάρχει. Το κουμπί Select Output Directory, ανοίγει μια προτροπή στην

οποία δηλώνουμε το μονοπάτι στο οποίο θα αποθηκευτεί το phrase file. To Create New Phrase File, χρησιμοποιείται για να δημιουργήσει ένα νέο phrase file. Τα πεδία TTS Provider και Voice δηλώνουν το σύστημα το οποίο θα χρησιμοποιεί ως τη «φωνή» που θα διαβάζει και θα εκφωνεί τα ηχητικά αποσπάσματα. Στη παρούσα πτυχιακή χρησιμοποιήθηκε όπως φαίνεται και στο παραπάνω στιγμιότυπο, το σύστημα Amazon Polly – Web με φωνή την Emma – British English – Female.

Έχοντας έτσι μια γενική εικόνα του λογισμικού ας πάμε να δούμε το τρόπο με τον οποίο δημιουργείται ένα ηχητικό απόσπασμα. Ας θεωρήσουμε ότι θέλαμε το ηχητικό απόσπασμα της λέξης "apple". Πηγαίνουμε στη κενή λίστα και εισάγουμε τη λέξη apple στο filename και στο phrase to speak. Δεν είναι απαραίτητο το filename να έχει πάντα το ίδιο όνομα με το περιεχόμενο του κειμένου, απλώς στη περίπτωση των λημμάτων είναι βολικό να εργαστεί κάνεις έτσι, ώστε να μπορεί να εντοπίζει εύκολα κάθε λήμμα με το όνομα του και μόνο. Πατώντας το κουμπί preview – μερικές φορές ίσως χρειαστεί παραπάνω της μιας φοράς – μπορούμε να ακούσουμε το απόσπασμα και το κουμπί πλέον να γίνει play. Εργαζόμαστε ομοίως για να προσθέσουμε επιπλέον αποσπάσματα. Αν θέλουμε να αποθηκεύσουμε, όλα τα αποσπάσματα σε ένα αρχείο phrase ή σε ένα κοινό folder, τότε αφήνουμε τη στήλη folder κενή. Για να αποθηκεύσουμε το αρχείο phrase, πατάμε το κουμπί save phrase file ή save phrase file as, για να ορίσουμε όνομα και τοποθεσία του αρχείου. Για να δημιουργήσουμε τα αποσπάσματα, επιλέγουμε το φάκελο στον οποίο θα αποθηκευτούν, με τη χρήση του Select Output Directory όπως είπαμε. Στη συνέχεια πατάμε κάτω αριστερά το κουμπί go και αναμένουμε. Αφού ολοκληρωθεί η διαδικασία δημιουργίας των αποσπασμάτων, πηγαίνοντας στο folder που επιλέξαμε πριν, θα δούμε ότι εμφανίζονται 2 φάκελοι mp3 και wav οι οποίοι, περιέχουν φυσικά τα αρχεία ήχου σε μορφή mp3 και wav αντίστοιχα. Παρακάτω παρουσιάζεται ένα παράδειγμα χρήσης του TTSAutomate:







Εικόνες 32, 33 και 34 : Δημιουργία Ηχητικών Αποσπασμάτων στο TTSAutomate

Τα αποσπάσματα που χρησιμοποιήθηκαν στο Quick Speak, βρίσκονται στο φάκελο tts_audio_files στο συνοδευτικό υλικό της πτυχιακής.