ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΜΣ «Προηγμένα Συστήματα Πληροφορικής – Ανάπτυξη Λογισμικού Και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής:	**Android Εφαρμογή Για Ταινίες και Ηθοποιούς** **Android Application For Movies And Actors**
Ονοματεπώνυμο φοιτητή:	Χρήστος Κανελλόπουλος
Πατρώνυμο:	Μενέλαος
Αριθμός Μητρώου:	ΜΠΣΠ19016
Επιβλέπων:	Ευθύμιος Αλέπης, Αναπληρωτής Καθηγητής

Ημερομηνία Παράδοσης: **Μάιος 2022**

Τριμελής Εξεταστική Επιτροπή

Ευθύμιος Αλέπης
Αναπληρωτής Καθηγητής

Μαρία Βίρβου
Καθηγήτρια

Κωνσταντίνος Πατσάκης
Αναπληρωτής Καθηγητής

Table of Contents

2. Περίληψη

Σε αυτή την ενότητα θα γίνει μία περίληψη της εργασίας γενικά αλλά και θα επισημανθούν σημαντικά στοιχεία της. Το θέμα της εργασίας ήταν να γίνει μία εφαρμογή Android με σκοπό ο κάθε χρήστης να μπορεί να διαβάσει τα βιογραφικά των αγαπημένων του ηθοποιών αλλά και να μπορεί να δει trailer των ταινιών του κάθε ηθοποιού της εφαρμογής. Η εργασία στοχεύει στο να δημιουργηθεί μία ωραία εφαρμογή και φιλική προς τον χρήστη αλλά και να διαπιστωθούν οι δυνατότητες που έχουν οι εφαρμογές Android σήμερα. Επίσης αυτό που έχει σκοπό η εργασία να αναδείξει είναι η τεχνολογίες που μπορούν να χρησιμοποιηθούν σε εφαρμογές Android. Η εργασία κατάφερε να αναδείξει τις τεχνολογίες όλες όσες χρησιμοποιήθηκαν αλλά και να φανεί το πόσο μεγάλες είναι οι δυνατότητες των εφαρμογών Android. Η τεχνολογία που χρησιμοποιήθηκε κατά κύριο λόγο είναι η Firebase η οποία χρησιμοποιήθηκε για τη βάση δεδομένων της εφαρμογής. Οι δυνατότητες που προσφέρει αυτό το εργαλείο στο Android το βοηθούν και δίνει τη δυνατότητα στον χρήστη να κάνει διάφορα πράγματα. Μετά από αυτή την εργασία ανακαλύφθηκαν πολλές από τις δυνατότητες που υπάρχουν πλέον στο Android αλλά και η πρόοδος που έχει σημειωθεί σε αυτό. Επίσης κάτι σημαντικό που πρέπει να σημειωθεί είναι η χρήση βιβλιοθηκών για την προβολή βίντεο και για την χρήση πολλών δυνατοτήτων που προσφέρει η εφαρμογή. Τέλος από την εργασία αυτή αναλύθηκε η διαδικασία του sign up , του sign in και του sign out για ένα χρήστη μέσω της Google αλλά και της Firebase. Όλα αυτά ήταν κάποια από τα πράγματα που αναλύθηκαν και χρησιμοποιήθηκαν στην εργασία καθώς και η γλώσσα προγραμματισμού Java που χρησιμοποιήθηκε για την υλοποίηση της.

In this section, a summary of the work in general will be made, as well as important elements of it. The theme of the work was to make an Android application in order for each user to be able to read the biographies of their favorite actors but also to be able to see the trailer of the films of each actor of the application. The work aims to create a nice application and user friendly but also to identify the capabilities of Android application today. Also what the work aims to highlight is the technologies that can be used in Android applications. The work managed to highlight all the technologies that were used but also to show how great of capabilities of Android applications are. The technology used mainly is Firebase which was used for the application database. The features offered by this tool on Android help it and enable the user to do various things. After this work, many of features that now exist in Android were discovered as well as the progress that has been made in it. Also something important to note is the use of libraries for video viewing and the use of many features offered by the application. Finally, this work analyzed the process of sign up, sign in and sign out for a user through Google and Firebase. All of these were some of the things that were analyzed and used in the work as well as the Java programming language used to implement it.

3.Σύντομη Περιγραφή της Εργασίας

Στα πλαίσια της εργασίας, αναπτύχθηκε μία εφαρμογή android στην οποία ένας χρήστης έχει τις παρακάτω δυνατότητες:

- Να κάνει login με τον λογαριασμό που διαθέτει στη google.
- Να κάνει signup/login μέσω της πλατφόρμα της εφαρμογής.
- Να διαβάσει τα βιογραφικά των αγαπημένων του ηθοποιών.

- Να διαβάσει την περιγραφή, να δει το τρέιλερ των ταινιών.
- Να αναζητήσει ταινίες με το όνομα της.
- Να αναζητήσει ηθοποιό με το όνομα του.

Το login με το λογαριασμό της google γίνετε πατώντας το button της google και ακολουθώντας τα βήματα που έχει. Το signup / login από την εφαρμογή γίνετε δίνοντας ο χρήστης τα στοιχεία που ζητάει η εφαρμογή για την καθεμία διαδικασία. Το να διαβάσει ο χρήστης ένα βιογραφικό ενός ηθοποιού γίνετε αν πατήσει πάνω στο εικονίδιο του ηθοποιού που θέλει Αντίστοιχα για να δει πληροφορίες και τρέιλερ μίας ταινίας ο χρήστης θα πρέπει πάλι να πατήσει το εικονίδιο της ταινίας που θέλει. Η αναζήτηση της ταινίας αλλά και του ηθοποιού που θέλει ο χρήστης γίνετε πατώντας το εικονίδιο της αναζήτησης και πληκτρολογώντας το όνομα της ταινίας ή του ηθοποιού που θέλει να αναζητήσει. Τέλος ο χρήστης μπορεί να δει και τα βιογραφικά των ηθοποιών και μετά την αναζήτηση πατώντας πάνω στο εικονίδιο. Το ίδιο συμβαίνει αντίστοιχα και με τις ταινίες.

4.Τεχνολογίες και Εργαλεία που χρησιμοποιήθηκαν

Σε αυτό το κεφάλαιο θα αναλυθούν τα εργαλεία αλλά και οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη αυτής της εφαρμογής. Τα εργαλεία που χρησιμοποιήθηκαν είναι τα εξής:

- Android Studio
- Firebase
- Java

FIGURE 1: ANDROID STUDIO

.

Το σημαντικότερο εργαλείο που χρησιμοποιήθηκε είναι το android studio. Το android studio είναι το επίσημο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE)για την Google χτισμένο σε Jetbrains IntelliJ IDEA λογισμικό και είναι σχεδιασμένο για Android εφαρμογές. Είναι διαθέσιμο για download σε Windows, MacOs και Linux. Αντικαθιστά το Eclipse Android Development Tools (E-ADT) ως το κύριο IDE για την ανάπτυξη εφαρμογών Android.

Το Android Studio ανακοινώθηκε στις 16 Μαΐου 2013 στο συνέδριο της Google I/O. Ήταν στο στάδιο προεπισκόπησης πρώιμης πρόσβασης ξεκινώντας από την έκδοση 0.1 τον Μάιο του 2013και μετά μπήκε στο beta ξεκινώντας από την έκδοση 0.8 που κυκλοφόρησε τον Ιούνιο του 2014. Η πρώτη σταθερή

έκδοση κυκλοφόρησε τον Δεκέμβριο του 2014 ξεκινώντας από την έκδοση 1.0. Στις 7 Μαΐου 2019,η Kotlin αντικατέστησε την Java ως την προτεινόμενη γλώσσα της Google για την ανάπτυξη εφαρμογών Android. Η Java υποστηρίζεται ακόμα, όπως και η C++.

Το ιδιαίτερο χαρακτηριστικό του Android Studio είναι η απουσία της δυνατότητας απενεργοποίησης της λειτουργίας αυτόματης αποθήκευσης. Οι ακόλουθες δυνατότητες παρέχονται στην τρέχουσα σταθερή έκδοση:

- Υποστήριξη κατασκευής με βάση το Gradle.
- Ανακατασκευή και γρήγορες διορθώσεις για Android.
- Εργαλεία για την απόδοση, τη χρηστικότητα, τη συμβατότητα έκδοσης και άλλα προβλήματα.
- Ενσωμάτωση ProGuard και δυνατότητες και app-signing δυνατότητες
- Οδηγούς βάσει προτύπων για τη δημιουργία κοινών σχεδίων και στοιχείων Android.
- Ένα layout editor ο οποίος επιτρέπει στο χρήστη το drag and drop UI και μία επιλογή προεπισκόπησης διατάξεων σε πολλές οθόνες διαμόρφωσης.
- Υποστήριξη για Android Wear εφαρμογές.
- Ενσωματωμένη Υποστήριξη για το Google Platform,που επιτρέπει την ενοποίηση με το Firebase Cloud Messaging και το Google App Engine.
- Εικονική συσκευή Android (Emulator) για εκτέλεση και εντοπισμό σφαλμάτων εφαρμογών στο Android Studio.

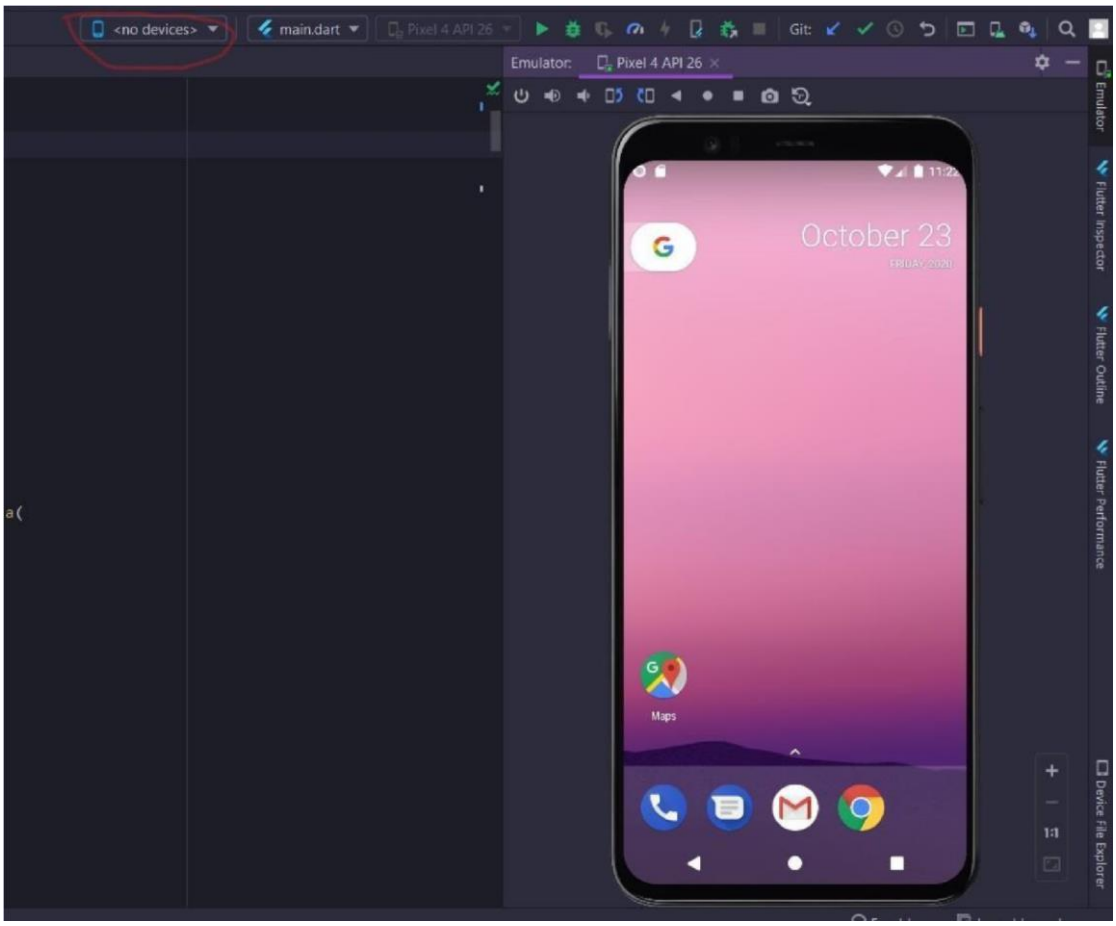

FIGURE 2: ANDROID STUDIO EMULATOR

Κάποια από τα παραπάνω χαρακτηριστικά θα αναλυθούν λίγο παραπάνω. Για το έκτο χαρακτηριστικό δηλαδή αυτό του editor layout υπάρχουν οι εξής παραπάνω πληροφορίες:

Το editor layout επιτρέπει να δημιουργείτε γρήγορα layouts χρησιμοποιώντας layouts αντί να γράφονται χειροκίνητα στα XML αρχεία. Το πρόγραμμα επεξεργασίας σχεδιασμού μπορεί να κάνει προεπισκόπηση του layout σε διαφορετικές εκδόσεις και συσκευές Android και μπορεί ο προγραμματιστής να αλλάξει το μέγεθος του layout για να βεβαιωθεί ότι η λειτουργεί καλά σε διάφορα μεγέθη οθόνης. Το editor layout είναι ιδιαίτερα ισχυρό κατά τη δημιουργία ενός constraint layout, ένα layout που είναι συμβατό με το Android 2.3 (επίπεδο API 9) και νεότερη έκδοση.

Το Layout Editor εμφανίζεται όταν ανοίγετε ένα αρχείο XML.

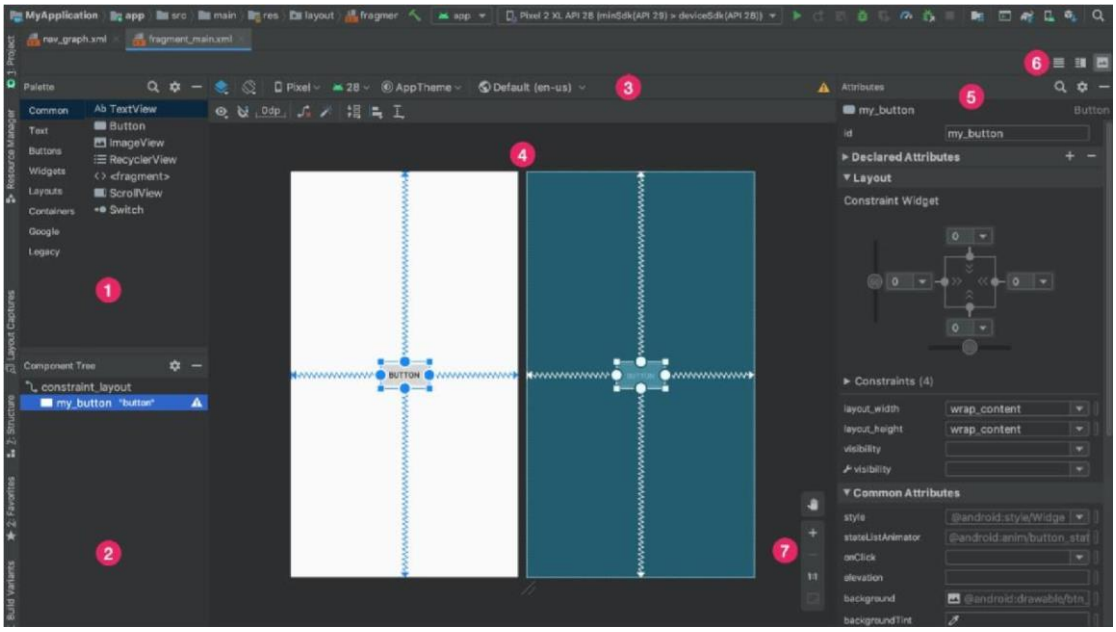

FIGURE 3: LAYOUT EDITOR

Το νούμερο 1 στην εικόνα 3 είναι η παλέτα(Pallet). Η παλέτα περιέχει διάφορες προβολές και ομάδες προβολής που μπορεί να μεταφέρει κάποιος στη διάταξη του. Το νούμερο 2 είναι το component tree. Η χρήση του είναι να δείχνει την ιεραρχία των στοιχείων της διάταξης. Το νούμερο 3 είναι η γραμμή εντολών. Αν κάνει κλικ σε αυτά τα κουμπιά ο προγραμματιστής θα μπορεί να διαμορφώσει την εμφάνιση του layout στο πρόγραμμα επεξεργασίας και να αλλάξει τα χαρακτηριστικά της διάταξης. Το νούμερο 4 είναι το πρόγραμμα επεξεργασίας σχεδίασης. Σε αυτό το σημείο μπορεί να επεξεργαστεί κάποιος το layout με προβολή σχεδίου ή προβολή σχεδίασης ‘η και τα δύο μαζί. Το νούμερο 5 είναι τα χαρακτηριστικά. Αυτό που κάνει ουσιαστικά είναι να δείχνει τα στοιχεία ελέγχου για τα χαρακτηριστικά της επιλεγμένης προβολής. Το νούμερο 6 είναι η λειτουργία προβολής. Η λειτουργία προβολής μπορεί να προβάλει τη διάταξη σε λειτουργίες όπως Code, Design ή Split. Η λειτουργία διαχωρισμού εμφανίζεται ταυτόχρονα με τα παράθυρα σχεδίασης και κώδικα. Το νούμερο 7 είναι τα στοιχεία ελέγχου ζουμ και μετακίνησης. Ουσιαστικά ελέγχει το μέγεθος και τη θέση της προεπισκόπησης στο πρόγραμμα επεξεργασίας.

Όταν ανοίγει κάποιος ένα αρχείο XML, το design editor ανοίγει από προεπιλογή. Για να επεξεργαστεί τη διάταξη XML στο πρόγραμμα επεξεργασίας κειμένου, πρέπει να κάνει κλικ στο κουμπί Code στην επάνω δεξιά γωνία του παραθύρου. Επίσης τα παράθυρα 1, 2 και 5 δεν είναι διαθέσιμα κατά την επεξεργασία του layout σε προβολή code.

Τα κουμπιά στην επάνω σειρά του προγράμματος επεξεργασίας σχεδίασης σας επιτρέπουν να διαμορφώνετε η εμφάνιση της διάταξης στον επεξεργαστή.

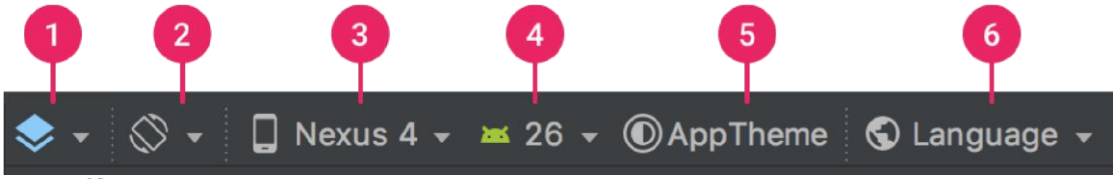

FIGURE 4: ΚΟΥΜΠΙΑ ΣΤΗ ΓΡΑΜΜΗ ΕΡΓΑΛΕΙΩΝ ΤΟΥ LAYOUT EDITOR ΠΟΥ ΡΥΘΜΙΖΟΥΝ ΤΗΝ ΕΜΦΑΝΙΣΗ ΤΗΣ ΔΙΑΤΑΞΗΣ.

Το νούμερο 1 στην παραπάνω εικόνα είναι ο σχεδιασμός και σχεδιάγραμμα. Το κουμπί αυτό ουσιαστικά δείχνει το layout στο πρόγραμμα επεξεργασίας. Αν επιλέξει τη Σχεδίαση κάποιος θα δει τη προεπισκόπηση του layout. Αν επιλέξει κάποιος το Blueprint μπορεί να δει μόνο τα περιγράμματα για κάθε προβολή. Αν κάποιος επιλέξει το Design + Blueprint μπορεί να δει τις δύο προβολές δίπλα δίπλα. Μπορεί επίσης να πατήσει το B για κύλιση σε αυτούς τους τύπους προβολής. Το νούμερο 2 είναι ο προσανατολισμός οθόνης και παραλλαγές του layout. Αν κάποιος πατήσει το νούμερο 2 θα μπορεί να επιλέξει ανάμεσα σε οριζόντιο και κατακόρυφο προσανατολισμό οθόνης ή να επιλέξει κάποιες άλλες λειτουργίες οθόνης για τις οποίες η εφαρμογή περιέχει εναλλακτικά layout όπως νυχτερινή λειτουργία. Αυτό το μενού περιέχει επίσης εντολές για τη δημιουργία μίας νέας παραλλαγής του layout. Μπορεί επίσης κάποιος να πατήσει το 0 για αλλαγή προσανατολισμού. Το νούμερο 3 είναι τύπος και μέγεθος συσκευής. Μπορεί κάποιος να επιλέξει τον τύπο συσκευής (τηλέφωνο / tablet, Android TV ή Wear OS) και διαμόρφωση οθόνης (μέγεθος και πυκνότητα). Μπορεί να επιλέξει επίσης από διάφορους προκαθορισμένους τύπους συσκευών και τους δικούς του ορισμούς AVD ή μπορεί να δημιουργήσει ένα νέο AVD επιλέγοντας Προσθήκη ορισμού συσκευής από τη λίστα. Μπορεί να αλλάξει το μέγεθος της συσκευής σύροντας την κάτω δεξιά γωνία του layout. Μπορεί επίσης να πατήσει D για κύλιση στη λίστα συσκευών. Το νούμερο 4 είναι η έκδοση του API. Με αυτό το κουμπί επιλέγετε η έκδοση του Android για την προεπισκόπηση του layout. Το νούμερο 5 είναι το θέμα εφαρμογής. Αυτό σημαίνει ότι με αυτό το κουμπί επιλέγει ποιο θέμα layout θα εφαρμοστεί στην προεπισκόπηση. Αυτό βέβαια, λειτουργεί μόνο για υποστηριζόμενα στυλ layout, επομένως πολλά θέματα σε αυτήν τη λίστα οδηγούν σε σφάλμα. Το νούμερο 6 είναι η γλώσσα. Επιλέγοντας το νούμερο 6 επιλέγετε τη γλώσσα που θα εμφανίζεται για τις συμβολοσειρές UI. Αυτή η λίστα εμφανίζει μόνο τις διαθέσιμες γλώσσες στους πόρους της συμβολοσειράς. Εάν θέλει κάποιος να επεξεργαστεί τη μετάφραση, κάνει κλικ στην επιλογή Επεξεργασία μεταφράσεων από το αναπτυσσόμενο μενού.

Κατά την προσθήκη ενός νέου layout στην εφαρμογή, πρέπει να δημιουργηθεί πρώτα ένα προεπιλεγμένο αρχείο διάταξης στο προεπιλεγμένο layout, έτσι ώστε να ισχύει για όλες τις διαμορφώσεις συσκευών. Μόλις έχετε το προεπιλεγμένο layout, μπορείτε να δημιουργήσετε παραλλαγές των layouts, για συγκεκριμένες διαμορφώσεις συσκευών, όπως για μεγάλες οθόνες. Μπορείτε να δημιουργήσετε μία νέα διάταξη με έναν από τους παρακάτω τρόπους:

- Χρησιμοποίηση του κύριου Μενού του Android Studio.
- Χρησιμοποίηση της προβολής έργου.
- Χρησιμοποίηση της προβολής Android.
- Χρησιμοποίηση του Resource Manager.

Για τον πρώτο τρόπο πρέπει να ακολουθηθούν τα εξής βήματα:

1. Στο παράθυρο του έργου, κάντε κλικ στην ενότητα στην οποία θέλετε να προστεθεί ένα layout.
2. Στο κύριο μενού, επιλέγω Αρχείο μετά Νέο μετά XML και μετά layout XML.

3. Στο παράθυρο διαλόγου που εμφανίζεται, πρέπει να καταχωρηθεί το όνομα αρχείου, την ετικέτα layout και το σύνολο προέλευσης στο οποίο ανήκει το layout.
4. Κάντε κλικ στο Τέλος για να δημιουργήσετε ένα layout.

Για το δεύτερο τρόπο πρέπει να ακολουθηθούν τα εξής βήματα:

1. Επιλογή της προβολής έργου μέσα από το παράθυρο έργου.
2. Δεξί κλικ στο directory layout στο οποίο θέλει ο προγραμματιστής να προσθέσει το layout.
3. Στο μενού περιβάλλοντος που εμφανίζεται, κλικ στο Νέο και μετά στο Αρχείο πόρων layout

Για το τρίτο τρόπο πρέπει να ακολουθηθούν τα εξής:

1. Επιλογή της προβολής Android από το παράθυρο έργου.
2. Κλικ στο φάκελο layout
3. Στο μενού περιβάλλοντος που εμφανίζεται, επιλογή Νέο και μετά Αρχείο πόρου layout.

Για το τέταρτο τρόπο πρέπει να ακολουθηθούν τα εξής:

1. Στο Resource Manager, επιλογή της καρτέλας layout.
2. Κλικ στο + και, στη συνέχεια, κλικ στο Layout Resource File.

Άλλο ένα επιπλέον χαρακτηριστικό είναι η έκδοση του APK. Το Android Studio περιλαμβάνει ένα αναλυτή APK που παρέχει άμεσες πληροφορίες σχετικά με τη σύνθεση του APK μετά την ολοκλήρωση της διαδικασίας κατασκευής. Η χρήση του APK analyzer μπορεί να μειώσει το χρόνο που αφιερώνετε σε θέματα εντοπισμού σφαλμάτων με αρχεία και πόρους DEX στην εφαρμογή και να μειώσει το μέγεθος του APK.

Με τον APK analyzer μπορείτε να χρησιμοποιήσετε τα εξής:

- Δείτε το απόλυτο και σχετικό μέγεθος των αρχείων στο APK, όπως τα αρχεία DEX και Android. • Κατανόηση της σύνθεσης των αρχείων DEX.
- Προβάλετε γρήγορα τις τελικές εκδόσεις αρχείων στο APK,όπως το AndroidManifest.xml αρχείο.
- Εκτέλεση μία παράλληλης σύγκρισης δύο APK.

Υπάρχουν τρεις τρόποι πρόσβασης στο APK analyzer όταν είναι ανοιχτό ένα έργο:

- Σύρετε ένα APK στο παράθυρο του προγράμματος επεξεργασίας του Android Studio.
- Μετάβαση στο Project και στο παράθυρο του Project και στη συνέχεια ένα διπλό κλικ στο αρχείο APK στον προεπιλεγμένο directory build/output/apk.
- Επιλογή Κατασκευή και μετά APK analyzer στη γραμμή μενού και ,στη συνέχεια, επιλέξτε το APK.

Εάν το έργο περιλαμβάνει πολλά AndroidManifest.xml αρχεία ή περιλαμβάνει βιβλιοθήκες που παρέχουν επίσης ένα manifest αρχείο, αυτά συγχωνεύονται σε ένα μόνο αρχείο APK. Αυτό το αρχείο manifest είναι συνήθως ένα δυαδικό αρχείο εντός του APK, αλλά όταν επιλέγεται στο APK analyzer , η φόρμα XML αυτής της οντότητας ανακατασκευάζεται και παρουσιάζεται. Αυτό το πρόγραμμα προβολής επιτρέπει να κατανοήσει κάποιος τυχόν αλλαγές που ενδέχεται να έχουν γίνει στην εφαρμογή κατά τη διάρκεια της έκδοσης. Επιπλέον, αυτό το πρόγραμμα προβολής παρέχει ορισμένες δυνατότητες, προειδοποιήσεις ή σφάλματα που εμφανίζονται στη πάνω δεξιά γωνία.

Επίσης κάποια επιπλέον πράγματα θα προσθέσουμε και για το Android Emulator. Το Android Emulator προσομοιώνει συσκευές Android στον υπολογιστή, έτσι ώστε να μπορεί ο προγραμματιστής να δοκιμάσει την εφαρμογή σε διάφορες συσκευές και σε επίπεδα API Android χωρίς να χρειάζεται να έχει κάποια φυσική συσκευή.

Ο εξομοιωτής παρέχει σχεδόν όλες τις δυνατότητες μιας πραγματικής συσκευής Android. Μπορεί να προσομοιώσει εισερχόμενες τηλεφωνικές κλήσεις και μηνύματα κειμένου, να καθορίσει τη θέση της συσκευής, να προσομοιώσει διαφορετικές ταχύτητες δικτύου, να προσομοιώσει περιστροφή και άλλους αισθητήρες υλικού, να αποκτήσει πρόσβαση στο Google Play Store και πολλά άλλα.

Ο έλεγχος της εφαρμογής στον εξομοιωτή είναι με κάποιους τρόπους πιο γρήγορος και πιο εύκολος από ό,τι σε μια φυσική συσκευή. Ο εξομοιωτής έρχεται με προκαθορισμένες διαμορφώσεις για διάφορες συσκευές Android,tablet,Wear OS και Android TV.

Το Android Emulator έχει πρόσθετες απαιτήσεις πέρα από τις βασικές απαιτήσεις του συστήματος για το Android Studio, οι οποίες περιγράφονται παρακάτω:

- SDK Tools 26.1.1 ή νεότερη έκδοση
- Επεξεργαστής 64-bit
- Windows: CPU με υποστήριξη UG
- HAXM 6.2.1 ή μεταγενέστερη έκδοση

Η χρήση επιτάχυνσης υλικού (hardware) έχει πρόσθετες απαιτήσεις σε Windows και Linux:

- Επεξεργαστής Intel σε Windows ή Linux: επεξεργαστής Intel με υποστήριξη για λειτουργίες VT-x,Intel MT64T(Intel 64) και Executive Disable(XD)Bit.
- Επεξεργαστής AMD σε Linux: Επεξεργαστής AMD με υποστήριξη για AMDVirtualization(AMD-V) και Supplemental Streaming SIMD Extensions 3 (SSSE3).
- Επεξεργαστή AMD στα Windows: Android Studio 3.2 ή νεότερη έκδοση και Windows10 ή υψηλότερη έκδοση για την λειτουργικότητα Windows Hypervisor Platform (WHPX).

Μετά τη σύνταξη μιας εφαρμογής με το Android Studio ,μπορεί να δημοσιευτεί στο Google Play Store. Η εφαρμογή πρέπει να συμμορφώνεται με την πολιτική περιεχομένου προγραμματιστή (developer content policy) του Google Play Store.

Για να λειτουργεί με Android 8.1 (επίπεδο API 27) και υψηλότερες εικόνες συστήματος, μία συνδεδεμένη κάμερα web πρέπει να έχει την δυνατότητα λήψης πλαισίων 720p.

Το Android Emulator καταργήθηκε για τα Windows 32-bit. Η υποστήριξη για τον emulator των Windows 32-bit συνεχίζεται αλλά δεν θα προστεθούν νέες δυνατότητες. Εάν χρησιμοποιεί κάποιος Windows 32-bit θα πρέπει να εγκαταστήσει τα Windows 64-bit.

Ο Android emulator εγκαθίστανται αν επιλέξει ο προγραμματιστής το στοιχείο Android Emulator στην καρτέλα SDK Tools του SDK Manager.

Το επόμενο πολύ σημαντικό στοιχείο που χρησιμοποιήθηκε για την υλοποίηση της εργασίας είναι η γλώσσα προγραμματισμού. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε

είναι η Java 8.0. Η Java είναι μία αντικειμενοστραφής γλώσσα προγραμματισμού που σχεδιάστηκε από την εταιρεία πληροφορικής Sun Microsystems.

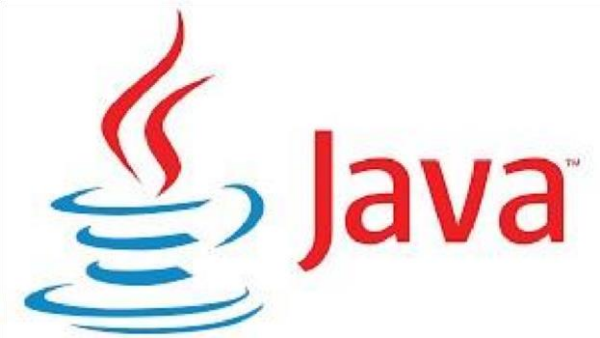

FIGURE 5: JAVA LOGO

Στις αρχές του 1991, η Sun αναζητούσε το κατάλληλο εργαλείο για να αποτελέσει την πλατφόρμα ανάπτυξης λογισμικού σε μικρο-συσκευές (έξυπνες οικιακές συσκευές έως πολύπλοκα συστήματα παραγωγής γραφικών). Τα εργαλεία της εποχής ήταν γλώσσες όπως η C++ και η C. Μετά από διάφορους πειραματισμούς προέκυψε το συμπέρασμα ότι οι υπάρχουσες γλώσσες δεν μπορούσαν να καλύψουν τις ανάγκες τους. Ο James Gosling, που εργαζόταν εκείνη τη στιγμή για την Sun, έκανε ήδη πειραματισμούς πάνω στη C++ και είχε παρουσιάσει κατά καιρούς πειραματικές γλώσσες όπως την C++ ++ η οποία μετά ονομάστηκε C# ως πρότυπα για το νέο εργαλείο που αναζητούσαν στην Sun. Τελικά μετά από λίγο καιρό κατέληξαν με μία πρόταση για το επιτελείο της εταιρείας, η οποία ήταν η γλώσσα Oak. Το όνομα της το πήρε από το ομώνυμο δέντρο το οποίο ο Gosling είχε έξω από το γραφείο του και έβλεπε κάθε μέρα. Η Oak ήταν μία γλώσσα που διατηρούσε μεγάλη συγγένεια με την C++. Παρόλα αυτά είχε πολύ πιο έντονο αντικειμενοστραφή χαρακτήρα (object oriented) σε σχέση με την C++ και χαρακτηριζόταν από την απλότητα της. Σύντομα οι υπεύθυνοι ανάπτυξης της νέας γλώσσας ανακάλυψαν ότι το όνομα Oak ήταν ήδη καταχωρημένο οπότε κατά την διάρκεια μιας εκ των πολλών συναντήσεων σε τοπικό καφέ αποφάσισαν να μετονομάσουν το νέο τους δημιούργημα σε Java που εκτός των άλλων ήταν το όνομα της αγαπημένης ποικιλίας καφέ για τους δημιουργούς της. Η επίσημη εμφάνιση της Java αλλά και του HotJava (πλοηγός με υποστήριξη Java) στη βιομηχανία της πληροφορικής έγινε το Μάρτιο του 1995 όταν η Sun την ανακοίνωσε στο συνέδριο Sun World 1995. Ο πρώτος μεταγλωττιστής (compiler) της ήταν γραμμένος σε γλώσσα C από τον James
Gosling. Το 1994, ο A.Van Hoff ξαναγράφει το μεταγλωττιστή της γλώσσας σε Java, ενώ το Δεκέμβριο του 1995 πρώτες οι IBM, Borland, Mitsubishi Electronics, Sybase και Symantec ανακοινώνουν σχέδια να χρησιμοποιήσουν Java για την δημιουργία λογισμικού. Από εκεί και πέρα η Java ακολουθεί μία ανοδική πορεία και είναι πλέον από τις πιο δημοφιλείς γλώσσες στο χώρο της πληροφορικής. Στις 13 Νοεμβρίου του 2006 η Java έγινε πλέον μία γλώσσα ανοιχτού κώδικα (GPL) όσον αφορά τον μεταγλωττιστή (javac) και το πακέτο ανάπτυξης (JDK,Java Development Kit).

Στις 27 Απριλίου 2010 η εταιρεία λογισμικού Oracle Corporation ανακοίνωσε ότι μετά από πολύμηνες συζητήσεις ήρθε σε συμφωνία για την εξαγορά της Sun Microsystems και των τεχνολογιών της (πνευματικά δικαιώματα) που η δεύτερη είχε στην κατοχή της ή έχει δημιουργήσει. Η συγκεκριμένη συμφωνία θεωρείται σημαντική για το μέλλον της Java και του γενικότερου οικοσυστήματος τεχνολογιών γύρω από αυτή μιας και ο έμμεσος έλεγχος της τεχνολογίας και η εξέλιξη της περνάει σε άλλα χέρια.

Σήμερα οι υποστηριζόμενες εκδόσεις είναι η Java 8 και 9. Οι σημαντικότερες που έχουν κυκλοφορήσει, με τις αντίστοιχες ημερομηνίες κυκλοφορίας, είναι οι εξής:

- JDK 1.0 (23 Δεκεμβρίου 1996)
- JDK 1.1 (19 Φεβρουαρίου 1997)
- J2SE 1.2 (8 Δεκεμβρίου 1998)
- J2SE 1.3 (8 Μαΐου 2000)
- J2SE 1.4 (6 Φεβρουαρίου 2002)
- J2SE 5.0 (30 Σεπτεμβρίου 2004)
- JAVA SE 6 (11 Δεκεμβρίου 2006)
- JAVA SE 7 (28 Ιουλίου 2011)
- JAVA SE 8 (18 Μαρτίου 2014)
- JAVA SE 9 (21 Σεπτεμβρίου 2017)
- JAVA SE 10 (20 Μαρτίου 2018)
- JAVA SE 11 (25 Σεπτεμβρίου 2018)

Ένα από τα βασικά πλεονεκτήματα της Java έναντι των περισσότερων άλλων γλωσσών είναι η ανεξαρτησία του λειτουργικού συστήματος και πλατφόρμας. Τα προγράμματα που είναι γραμμένα σε Java τρέχουνε ακριβώς το ίδιο σε Windows, Linux, Unix και Macintosh (σύντομα θα τρέχουν και σε PlayStation καθώς και σε άλλες κονσόλες παιχνιδιών) χωρίς να χρειαστεί να ξαναγίνει μεταγλώττιση (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα. Για να επιτευχθεί όμως αυτό χρειαζόταν κάποιος τρόπος έτσι ώστε τα προγράμματα γραμμένα σε Java να μπορούν να είναι κατανοητά από κάθε υπολογιστή ανεξάρτητα του είδους επεξεργαστή (Intel x86, IBM, Sun SPARC, Motorola) αλλά και λειτουργικού συστήματος (Windows,Unix,Linux, BSD, MacOS) . Ο λόγος είναι ότι κάθε κεντρική μονάδα επεξεργασίας κατανοεί διαφορετικό κώδικα μηχανής. Ο κώδικας assembly που μεταφράζεται και εκτελείται σε Windows είναι διαφορετικός από αυτόν που μεταφράζεται και εκτελείται σε ένα υπολογιστή Macintosh. Η λύση δόθηκε με την ανάπτυξη της Εικονικής Μηχανής(Virtual Machine ή VM ή ΕΜ).

Αφού γράφει κάποιο πρόγραμμα σε Java , στη συνέχεια μεταγλωττίζεται μέσω του μεταγλωττιστή javac, ο οποίος παράγει έναν αριθμό από αρχεία .class (κώδικας byte ή bytecode). Ο code byte είμαι η μορφή που παίρνει ο πηγαίος κώδικας της Java όταν μεταγλωττιστεί. Όταν πρόκειται να εκτελεστεί η εφαρμογή σε ένα μηχάνημα, το Java Virtual Machine που πρέπει να είναι εγκατεστημένο σε αυτό θα αναλάβει να διαβάσει τα αρχεία .class. Στη συνέχεια μεταφράζει σε γλώσσα μηχανής που να υποστηρίζεται από το λειτουργικό σύστημα και τον επεξεργαστή, έτσι ώστε να εκτελεστεί. Αυτό συμβαίνει με την παραδοσιακή Εικονική Μηχανή (Virtual Machine).

Πιο σύγχρονες εφαρμογές της εικονικής Μηχανής μπορούν και μεταγλωττίζουν εκ των προτέρων τμήματα bytecode απευθείας σε κώδικα μηχανής (εγγενή κώδικα ή native code) με αποτέλεσμα να βελτιώνεται η ταχύτητα. Χωρίς αυτό δεν θα ήτα δυνατή η εκτέλεση λογισμικού γραμμένου σε Java .Η JVM είναι λογισμικό που εξαρτάται από την πλατφόρμα, δηλαδή για κάθε είδους λειτουργικού συστήματος και αρχιτεκτονικής επεξεργαστή υπάρχει και διαφορετική έκδοση του. Έτσι υπάρχουν διαφορετικές JVM για Windows, Linux, Unix, Macintosh, κινητά τηλέφωνα, παιχνιδομηχανές.

Οτιδήποτε θέλει να κάνει ο προγραμματιστής (ή ο χρήστης) γίνεται μέσω της εικονικής μηχανής. Αυτό βοηθάει στο να υπάρχει μεγαλύτερη ασφάλεια στο σύστημα γιατί η εικονική μηχανή είναι υπεύθυνη για την επικοινωνία χρήστη – υπολογιστή. Ο προγραμματιστής δεν μπορεί να γράψει κώδικα ο οποίος θα έχει καταστροφικά αποτελέσματα για τον υπολογιστή γιατί η εικονική μηχανή θα τον ανιχνεύσει και δεν θα επιτρέψει να εκτελεστεί. Από την άλλη μεριά ούτε ο χρήστης μπορεί να κατεβάσει κακό κώδικα από το δίκτυο και να τον εκτελέσει. Αυτό είναι

ιδιαίτερα χρήσιμο για μεγάλα κατανεμημένα συστήματα όπου πολλοί χρήστες χρησιμοποιούν τα ίδιο πρόγραμμα συγχρόνως.

Ακόμα μία ιδέα που βρίσκεται πίσω από την Java είναι η ύπαρξη του Garbage Collector. Garbage Collector είναι μια κοινή ονομασία που χρησιμοποιείται στον τομέα της πληροφορικής για να δηλώσει την ελευθέρωση τμημάτων μνήμης από δεδομένα που δε χρειάζονται και που δε χρησιμοποιούνται άλλο. Αυτή η απελευθέρωση μνήμης στη Java είναι αυτόματη και γίνεται μέσω του Garbage Collector. Υπεύθυνη για αυτό είναι και πάλι η εικονική μνήμη η οποία μόλις καταλάβει ότι η σωρός (heap) της μνήμης (στη Java η συντριπτική πλειοψηφία των αντικειμένων αποθηκεύονται σε σωρό σε αντίθεση με τη C++ όπου αποθηκεύονται κυρίως στη στοίβα) κοντεύει να γεμίσει ενεργοποιεί το Garbage Collector. Έτσι ο προγραμματιστής δεν χρειάζεται να ανησυχεί για το πότε και αν θα ελευθερώσει ένα συγκεκριμένο τμήμα της μνήμης, ούτε και για σφάλματα δεικτών. Αυτό είναι ιδιαίτερα γιατί είναι κοινά τα σφάλματα προγραμμάτων που οφείλονται σε λανθασμένο χειρισμό της μνήμης.

Η προεπιλεγμένη μέθοδος για το Garbage Collector μοιράζει τη σωρό σε δύο κομμάτια. Το πρώτο, χρησιμοποιείται για νέα αντικείμενα και λέγεται maternity ward. Αφού επιβιώσουν ένα προεπιλεγμένο αριθμό εκκαθαρίσεων στο maternity ward,τα αντικείμενα μεταφέρονται στο δεύτερο κομμάτι της σωρού. Το maternity ward είναι μοιρασμένο σε δύο κομμάτια ,το από χώρο και τον προς χώρο. Νέα αντικείμενα τοποθετούνται αρχικά στο πρώτο κομμάτι .Όταν αυτός γεμίσει, αντικείμενα αντιγράφονται στον δεύτερο κομμάτι και οι δύο χώροι ανταλλάσσουν ρόλο.

Παρόλο που η εικονική μνήμη προσφέρει όλα αυτά (και όχι μόνο) τα πλεονεκτήματα, η Java αρχικά ήταν πιο αργή σε σχέση με τις άλλες προγραμματιστικές γλώσσες υψηλού επιπέδου(high-level) όπως η C++. Εμπειρικές μετρήσεις στο παρελθόν είχαν δείξει ότι η C++ μπορούσε να είναι αρκετές φορές γρηγορότερη από την Java.Ωστόσο γίνονται προσπάθειες από την Oracle για την βελτιστοποίηση της εικονικής μηχανής, ενώ υπάρχουν και άλλες υλοποιήσεις εικονικής μηχανής από άλλες εταιρείες (όπως της IBM), οι οποίες μπορεί σε κάποια σημεία να προσφέρουν καλύτερα και σε κάποια άλλα χειρότερα αποτελέσματα. Επιπλέον με την καθιέρωση των μεταγλωττιστών JIT (Just In Time) οι οποίοι μετατρέπουν τον bytecode απευθείας σε γλώσσα μηχανής, η διαφορά ταχύτητας από την C++ έχει μικρύνει πολύ.

Οι τελευταίες εκδόσεις του javac με τη χρήση της τεχνολογίας Hot Spot έχουν καταφέρει αξιόλογες επιδόσεις που πλησιάζουν ή και ξεπερνούν σε μερικές περιπτώσεις τον εγγενή κώδικα.

Όλα τα εργαλεία που χρειάζεται κάποιος για να γράψει Javaπρογράμματα έρχονται δωρεάν, από το περιβάλλον ανάπτυξης μέχρι εργαλεία build όπως το Apache Ant και βιβλιοθήκες, ενώ υπάρχουν πολλές διαφορετικές υλοποιήσεις της Εικονικής Μηχανής και του μεταγλωττιστή(όπως πχ GNU Compiler for Java)

Πολλά εργαλεία και τεχνολογίες σε Java μπορούν να βρεθούν στο Apache Software Foundation αλλά και στο Jakarta Project.

Για να γράψει κάποιος κώδικα Java δε χρειάζεται τίποτα άλλο παρά έναν επεξεργαστή κειμένου, όπως το Σημειωματάριο(Notepad) των Windows ή ο vi(γνωστός στον χώρο των Unix).
Παρ' όλα αυτά, ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) βοηθάει πολύ, ιδιαίτερα στον εντοπισμό σφαλμάτων (debugging). Υπάρχουν αρκετά διαθέσιμα, ενώ πολλά από αυτά έρχονται δωρεάν.

Το τρίτο εργαλείο που χρησιμοποιήθηκε για την εργασία ονομάζεται Firebase.

FIGURE 6: FIREBASE LOGO

Η Firebase είναι μία πλατφόρμα που αναπτύχθηκε από τη Google για τη δημιουργία εφαρμογών για κινητά και web. Αρχικά ήταν μια ανεξάρτητη εταιρεία που ιδρύθηκε το 2011. Το 2014, η Google απέκτησε την πλατφόρμα και είναι πλέον η κορυφαία προσφορά τους για ανάπτυξη εφαρμογών.

Το Firebase εξελίχθηκε από την Envolve, μία προηγούμενη κίνηση που ιδρύθηκε από τους James Tamplin και Andrew Lee το 2011.Η Envolve παρείχε στους προγραμματιστές ένα API που επιτρέπει την ενσωμάτωση της λειτουργικότητας των διαδικτυακών συνομιλιών στους ιστοτόπους τους. Μετά την απελευθέρωση της υπηρεσίας συνομιλίας, οι Tamplin και Lee διαπίστωσαν ότι χρησιμοποιείται για τη διαβίβαση δεδομένων εφαρμογής που δεν ήταν μηνύματα συνομιλίας. Οι προγραμματιστές χρησιμοποιούν το Envolve για να συγχρονίσουν δεδομένα εφαρμογών, όπως η κατάσταση παιχνιδιού σε πραγματικό χρόνο μεταξύ των χρηστών τους. Ο Tamplin και ο Lee αποφάσισαν να διαχωρίσουν το σύστημα συνομιλίας και την αρχιτεκτονική σε πραγματικό χρόνο που το τροφοδότησε. Ίδρυσαν το Firebase ως ξεχωριστή εταιρεία το Σεπτέμβριο του 2011 και κυκλοφόρησε στο κοινό τον Απρίλιο του 2012.

Το πρώτο προϊόν του Firebase ήταν το Firebase Realtime Database, ένα API που συγχρονίζει δεδομένα εφαρμογών σε συσκευές iOS, Android και Web και τα αποθηκεύει στο cloud του Firebase. Το προϊόν βοηθάει τους προγραμματιστές λογισμικού στη δημιουργία συνεργατικών εφαρμογών σε πραγματικό χρόνο.

Τον Μάιο του 2012, ένα μήνα μετά την κυκλοφορία του beta, η Firebase συγκέντρωσε 1.1 εκατομμύρια δολάρια σε χρηματοδότηση πόρων από επιχειρηματίες κεφαλαίων Flybridge Capital Partners, Greylock Partners Founder Collective και New Enterprise Associates. Τον Ιούνιο του 2013, η εταιρεία συγκέντρωσε περαιτέρω 5.6 εκατομμύρια δολάρια σε χρηματοδότηση της σειράς από την Union Square Ventures και την Flybridge Capital Parterns.

Το 2014, η Firebase παρουσίασε δύο προϊόντα το Firebase Hosting και το Firebase Authentication. Αυτό τοποθέτησε την εταιρεία ως κινητό backend ως υπηρεσία.

Τον Οκτώβριο του 2014 η Firebase εξαγοράστηκε από την Google. Ένα χρόνο αργότερο τον Οκτώβριο του 2015, η Google απέκτησε το Divshot, μια πλατφόρμα φιλοξενίας HTML5, για την συγχώνευση με την ομάδα του Firebase.

Τον Μάιο του 2016, στο Google I/O, το ετήσιο συνέδριο προγραμματιστών της εταιρείας, το Firebase παρουσίασε το Firebase Analytics και ανακοίνωσε ότι επεκτείνει τις υπηρεσίες του για να γίνει ενοποιημένη πλατφόρμα backend as a Service (BaaS) για προγραμματιστές κινητών. Το Firebase πλέον ενοποιείται με διάφορες άλλες υπηρεσίες της Google, όπως το Google Cloud Platform, το AdMob και το Google Ads, για να προσφέρουν ευρύτερα προϊόντα και κλίμακα για προγραμματιστές. Το Google Cloud Messaging, η υπηρεσία Google για αποστολή ειδοποιήσεων push σε συσκευές Android, αντικαταστάθηκε από ένα προϊόν Firebase, το Firebase Cloud Messaging, η οποία πρόσθεσε τη λειτουργικότητα για την παροχή ειδοποιήσεων push τόσο σε συσκευές iOS όσο και σε συσκευές web. Τον Ιανουάριο του 2017, η Google απέκτησε το Fabric και το Crashlytics από το Twitter για να προσθέσει αυτές τις υπηρεσίες στο Firebase.

Τον Οκτώβριο του 2017 το Firebase κυκλοφόρησε το Cloud Firestore, μία βάση δεδομένων εγγράφων σε πραγματικό χρόνο ως προϊόν που διαδέχθηκε την αρχική βάση δεδομένων Firebase Realtime.

Η Firebase ισχυρίστηκε ότι χρησιμοποιείται από την Google για την παρακολούθηση των χρηστών χωρίς να το γνωρίζουν. Στις 14 Ιουλίου 2020, κατατέθηκε αγωγή στη Google ότι παραβίασε τον ομοσπονδιακό νόμο περί απορρήτου της Καλιφόρνια. Δήλωσε ότι μέσω του Firebase, η Google συνέλεξε και αποθήκευσε δεδομένα χρηστών, καταγράφοντας αυτό που ο χρήστης βλέπει σε πολλούς τύπους εφαρμογών, παρά ότι ο χρήστης ακολουθεί τις οδηγίες της Google για να απενεργοποιήσει τη δραστηριότητά web και εφαρμογών που συλλέγει η εταιρεία.

5.Παρουσίαση Εργασίας

5.1 Dependencies

Η παρουσίαση της εργασίας θα ξεκινήσει με την παρουσίαση κάποιων αρχικών στοιχείων που είναι πολύ σημαντικά έτσι ώστε να λειτουργεί η εφαρμογή όπως πρέπει. Αυτά είναι κάποια στοιχεία που υπάρχουν στο αρχείο AndroidManifest καθώς και στο αρχείο gradle. Επίσης θα αναλυθούν και τα layout της εφαρμογής. Η αρχή θα γίνει με το πρώτο αρχείο που αναφέρθηκε παραπάνω. Στο AndroidManifest υπάρχουν κάποιες γραμμές οι οποίες χωρίς αυτές δεν θα έτρεχε η εφαρμογή. Αυτές είναι η εντολές που ξεκινούν με uses-permission. Οι εντολές αυτές ουσιαστικά αυτό που κάνουν είναι να δίνουν δικαιώματα στο χρήστη. Αυτά τα δικαιώματα που πρέπει να έχει ο χρήστης σε αυτή την εφαρμογή είναι δύο . Το πρώτο είναι αυτό του Internet. Το δεύτερο είναι το δικαίωμα του να διαβάζει. Αυτό συμβαίνει με την εντολή που έχει το uses-permission το READ_EXTERNAL_STORAGE.

Στο αρχείο gradle αυτό που θα αναλυθεί είναι τα στοιχεία μέσα στο dependency. Σε αυτό το σημείο του αρχείου είναι κάποια στοιχεία τα οποία βοηθάνε την εφαρμογή να μπορεί να χρησιμοποιήσει κάποιες από τις βιβλιοθήκες που αλλιώς δεν θα μπορούσε. Dependency γενικά στην πληροφορική είναι μία κατάσταση στην οποία ένα αντικείμενο χρησιμοποιεί μια συνάρτηση ενός άλλου αντικειμένου. Τα dependency του προγράμματος είναι τα εξής που φαίνονται στην εικόνα που ακολουθεί.

```
implementation 'com.karumi:dexter:6.2.2'
implementation 'com.google.android.exoplayer:exoplayer:2.10.8'
implementation 'com.google.android.exoplayer:exoplayer-core:2.10.8'
implementation 'com.google.android.exoplayer:exoplayer-dash:2.10.8'
implementation 'com.google.android.exoplayer:exoplayer-hls:2.10.8'
implementation 'com.google.android.exoplayer:exoplayer-smoothstreaming:2.10.8'
implementation 'com.google.android.exoplayer:exoplayer-ui:2.10.8'
implementation 'com.rengwuxian.materialedittext:library:2.1.4'
implementation 'com.firebaseui:firebase-ui:6.2.0'
implementation 'de.hdodenhof:circleimageview:3.1.0'
implementation "androidx.cardview:cardview:1.0.0"
implementation 'com.squareup.picasso:picasso:2.5.2'
implementation 'androidx.appcompat:appcompat:1.2.0'
implementation 'com.google.android.material:material:1.3.0'
implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
implementation 'com.google.firebase:firebase-auth:20.0.3'
implementation 'com.google.firebase:firebase-database:19.6.0'
implementation 'com.google.firebase:firebase-storage:19.2.1'
testImplementation 'junit:junit:4.+'
androidTestImplementation 'androidx.test.ext:junit:1.1.2'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
implementation 'com.github.bumptech.glide:glide:4.12.0'
annotationProcessor 'com.github.bumptech.glide:compiler:4.12.0'
```

FIGURE 7: DEPENDENCIES

Σε αυτή την εικόνα υπάρχουν κάποιες παράμετροι που πρέπει να αναλυθούν. Τα πρώτα στοιχεία που θα πρέπει να αναλυθούν είναι τα implementation που έχουν μέσα στην έκφραση τη λέξη firebase. Οι εκφράσεις αυτές βοηθάνε στο να ενεργοποιηθεί η βάση δεδομένων του προγράμματος. Επίσης βοηθάει στο να μπορεί η βάση δεδομένων να αποθηκεύει τις εικόνες και τα video που χρειάζονται για την εφαρμογή. Άλλη μία σημαντική αναφορά που πρέπει να γίνει είναι στο implementation την έκφραση circleimageview. Η έκφραση αυτή δίνει το δικαίωμα στον προγραμματιστή να χρησιμοποιήσει για τις εικόνες που θα χρησιμοποιηθούν στην εφαρμογή να τις κάνει κυκλικές. Επόμενη σημαντική παρατήρηση που μπορεί να γίνει είναι για το implementation το οποίο περιέχει την έκφραση cardview. Σε αυτή την περίπτωση μπορεί ο προγραμματιστής να χρησιμοποιήσει αυτή την έκφραση έτσι ώστε να έχει το design μίας κάρτας. Οι τελευταίες δύο εντολές που πρέπει να αναλυθούν. Αυτά είναι το Picasso και η glide. Αυτές οι δύο εντολές δίνουν το δικαίωμα στον προγραμματιστή να μπορεί να βάλει στην εφαρμογή εικόνες με όποιο τρόπο θέλει.

5.2Menu Folder

5.2.1. Item Log Out

Επόμενη σημαντική πληροφορία είναι ένας φάκελος που έχει δημιουργηθεί με το όνομα menu. Αυτός ο φάκελος περιέχει δύο αρχεία. Το πρώτο ονομάζεται example_menu. Αυτό φαίνεται και στην παρακάτω εικόνα.

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

    <item
        android:id="@+id/logout"
        android:title="Log Out"
        app:showAsAction="never" />

</menu>
```

FIGURE 8: ITEM LOG OUT

Στην παραπάνω εικόνα φαίνεται το αρχείο example_menu. Με τις εντολές xmlns επιτρέπονται εντολές τύπου android: και τύπου app: . Η εντολή item δείχνει τα αντικείμενα που υπάρχουν στο menu αυτού του αρχείου. Σε αυτό το κώδικα υπάρχει ένα item με το όνομα logout σύμφωνα με την εντολή android:id. Το item γράφει Log Out.

5.2.2 Search Menu

Το δεύτερο αρχείο το οποίο υπάρχει στο φάκελο με το όνομα menu φαίνεται στην παρακάτω εικόνα.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
<item android:id="@+id/search"
    android:icon="@drawable/ic_baseline_search_24"
    android:title="Search Data"
    app:showAsAction="always"
    app:actionViewClass="android.widget.SearchView" />
    <item
        android:id="@+id/logout"
        android:title="Log Out"
        app:showAsAction="never" />
</menu>
```

FIGURE 9 : SEARCH MENU

Στην παραπάνω εικόνα φαίνεται το αρχείο searchmenu. Με τις εντολές τύπου xmlns επιτρέπου εντολές τύπου android: και τύπου app:. Σε αυτό το αρχείο υπάρχουν δύο εντολές item. Η πρώτη εντολή έχει μία εικόνα η οποία δείχνει το icon της αναζήτησης. Αυτό γίνεται με την εντολή android:icon. Το όνομα το οποίο έχει το search και φαίνεται με την εντολή android:id. Τέλος το item γράφει Search Data. Το τελευταίο στοιχείο είναι αυτό που ονομάζεται logout. Αυτό που γράφει το item είναι Logout.

5.3. Layouts

5.3.1 Activity Main

Επόμενη σημαντική πληροφορία είναι αυτή των layouts. Η ανάλυση των layouts θα ξεκινήσει από το πρώτο το οποίο ονομάζεται activity_main. Στις επόμενες εικόνες φαίνεται ο κώδικας που χρησιμοποιήθηκε.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:text="All Movies Trailers"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.391"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.175" />
```

FIGURE 10 : ACTIVITY MAIN (ΕΙΚΟΝΑ 1)

```
<Button
    android:id="@+id/btn_login"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="68dp"
    android:text="Login"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.439"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView" />

<Button
    android:id="@+id/btn_signup"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="316dp"
    android:text="SignUp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.437"
    app:layout_constraintStart_toStartOf="parent" />
```

FIGURE 11: ACTIVITY MAIN (ΕΙΚΟΝΑ 2)

```
<com.google.android.gms.common.SignInButton
    android:id="@+id/signInButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="38dp"
    android:layout_marginTop="339dp"
    android:layout_marginEnd="150dp"
    android:layout_marginBottom="100dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.927"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.886" />
```

FIGURE 12: ACTIVITY MAIN (ΕΙΚΟΝΑ 3)

Σε αυτές τις εικόνες φαίνεται το πώς έχει σχεδιαστεί η εφαρμογή στη πρώτη σελίδα της. Αρχικά με τις εντολές xmlns επιτρέπει στο layout να χρησιμοποιήσει εντολές τύπου android: και τύπου tools: . Στη συνέχεια με τις εντολές android:layout_width και android:layout_height ορίζουν το πλάτος και το

ύψος του layout. Για να κλείσει η ανάλυση τουλάχιστον για τα στοιχεία που ορίζουν τις προδιαγραφές του layout υπάρχει η τελευταία εντολή η οποία έχει τη μορφή tools:context και ορίζει σε ποια σελίδα απευθύνεται το συγκεκριμένο layout.

Επόμενα στοιχεία που πρέπει να αναλυθούν για το activity main είναι τα στοιχεία τα οποία βλέπει ο χρήστης στην οθόνη. Σε αυτό το layout υπάρχουν τρία buttons και ένα textview .Πρώτο στοιχείο που φαίνεται στις παραπάνω εικόνες είναι το textview. Το όνομα του textview στο layout αυτό είναι textview.Αυτό φαίνεται στην εντολή android:id. Το μέγεθος των γραμμάτων που θα έχει το κείμενο του είναι 20sp. Αυτό φαίνεται στην εντολή android:textSize. Το κείμενο το οποίο έχει το textview φαίνεται στην εντολή android:text. Οι υπόλοιπες εντολές ορίζουν το σημείο που θα φαίνεται στην οθόνη της εφαρμογής. Το πρώτο button ονομάζεται btn_login και αυτό ορίζεται με την εντολή android:id. Με την εντολή android:text κάποιος μπορεί να ορίσει τι θα λέει το button . οι υπόλοιπες εντολές του button ουσιαστικά ορίζουν που θα εμφανίζεται στην οθόνη. Το δεύτερο button ονομάζεται btn_signup το οποίο ορίζεται όπως έχει αναφερθεί και παραπάνω από την εντολή android:id. Το κείμενο το οποίο φαίνεται στο button αυτό γράφεται μέσα στην εντολή android:text. Οι υπόλοιπες εντολές ορίζουν τις συντεταγμένες που έχει το button πάνω στο layout και το που φαίνεται αυτό στην οθόνη του χρήστη. Το τρίτο και τελευταίο button είναι τύπου SignInButton και κάνει το signIn στην Google. Το όνομα στο button αυτό είναι signInButton. Οι υπόλοιπες εντολές ορίζουν τις συντεταγμένες που έχει το button στο layout πάνω.

5.3.2 Activity Sign Up

Επόμενο layout που θα αναλυθεί είναι αυτό που ονομάζεται activity_signup. Αυτό το layout δείχνει τη σελίδα που βλέπει ο χρήστης στην εφαρμογή όταν πάει να κάνει signup στην εφαρμογή. Στις παρακάτω εικόνες φαίνεται ο κώδικας που έχει χρησιμοποιηθεί για αυτό το layout.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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=".SignUp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:orientation="vertical"
        android:padding="16dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Create a new Account"
            android:textSize="20sp"
            android:textStyle="bold" />
```

FIGURE 13 : ACTIVITY SIGN UP (ΕΙΚΟΝΑ 1)

```
<ImageView

    android:id="@+id/profile_image"
    android:layout_width="73dp"
    android:layout_height="72dp"
    android:src="@drawable/ic_launcher_background"
    />

<com.rengwuxian.materialedittext.MaterialEditText
    android:id="@+id/username"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:hint="Username"
    app:met_floatingLabel="normal" />

<com.rengwuxian.materialedittext.MaterialEditText
    android:id="@+id/email"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:hint="Email"
    android:inputType="textEmailAddress"
    app:met_floatingLabel="normal" />
```

FIGURE 14: ACTIVITY SING UP (ΕΙΚΟΝΑ 2)

```
        <com.rengwuxian.materialedittext.MaterialEditText
            android:id="@+id/password"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:hint="Password"
            android:inputType="textPassword"
            app:met_floatingLabel="normal" />
        <Button
            android:id="@+id/btn_register"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:backgroundTint="@color/material_on_primary_disabled"
            android:text="register"
            android:textColor="@color/black" />

    </LinearLayout>

</RelativeLayout>
```

FIGURE 15 : ACTIVITY SIGN UP (ΕΙΚΟΝΑ 3)

Σε αυτές τις εικόνες φαίνεται το πώς έχει σχεδιαστεί η εφαρμογή στη signup σελίδα της. Αρχικά με τις εντολές xmlns επιτρέπει στο layout να χρησιμοποιήσει εντολές τύπου android: και τύπου tools:. Στη συνέχεια με τις εντολές android:layout_width και android:layout_height ορίζεται το ύψος και το πλάτος του layout. Για να κλείσει η ανάλυση όσον αφορά τουλάχιστον τα στοιχεία που ορίζουν τις προδιαγραφές του layout υπάρχει η τελευταία εντολή η οποία έχει τη μορφή tools:context και ορίζει σε ποια σελίδα αναφέρεται το συγκεκριμένο layout.

Επόμενα στοιχεία που πρέπει να αναλυθούν για το activity αυτό είναι τα στοιχεία τα οποία βλέπει ο χρήστης στην οθόνη. Σε αυτό το layout υπάρχουν ένα imageview, τρία materialedittext, ένα textview και ένα button. Το πρώτο στοιχείο που θα αναλυθεί είναι το textview . Το στοιχείο αυτό έχει χρησιμοποιηθεί με το κείμενο που φαίνεται στην εντολή android:text για να δηλώσει στο χρήστη το σκοπό της σελίδας. Τα γράμματα στο κείμενο έχουν μέγεθος το οποίο αναφέρεται στην εντολή android:textSize και είναι 20sp. Τα γράμματα του κειμένου επίσης σε αυτό το layout έχουν και το style που ορίζεται από την εντολή android:textStyle. Όπως φαίνεται στην εικόνα αυτή η εντολή ισούται με τη λέξη bold. Επόμενο στοιχείο που θα αναλυθεί είναι το imageview. Αυτό το στοιχείο χρησιμοποιείται με σκοπό να μπορεί ο χρήστης να προσθέσει την εικόνα που επιθυμεί. Το όνομα που έχει στο layout είναι το profile_image και αυτό φαίνεται στην εντολή android:id. Επίσης για να μπορεί να καταλάβει ο χρήστης που πρέπει να πατήσει για να προσθέσει την εικόνα του υπάρχει και η εντολή android:src η οποία χρησιμοποιεί μία φωτογραφία του συστήματος έτσι ώστε ο χρήστης να πατήσει πάνω της και να βάλει την δικιά του φωτογραφία.

Οι υπόλοιπες εντολές που υπάρχουν είναι για να χαρακτηρίσουν τις συντεταγμένες του imageview πάνω στο layout. Επόμενο στοιχείο που θα αναλυθεί είναι το πρώτο materialedittext. Το όνομα που έχει στο layout αυτό είναι username και φαίνεται από την εντολή android:id. Επίσης άλλη μία

εντολή που θα πρέπει να αναλυθεί είναι η android:hint. Σε αυτή την εντολή ο προγραμματιστής γράφει το στοιχείο το οποίο θέλει να δώσει ο χρήστης. Το στοιχείο αυτό φαίνεται με γκρι γράμματα και δεν επηρεάζουν απλά είναι εκεί για να βοηθήσουν το χρήστη. Σε αυτό το στοιχείο αυτό που γράφει η εντολή αυτή είναι username. Οι υπόλοιπες εντολές είναι για να στοιχίσουν πάνω στο layout το στοιχείο αυτό. Το δεύτερο materialedittext έχει το όνομα email όπως φαίνεται και στην εικόνα στην εντολή android:id. Επίσης το μήνυμα που υπάρχει μέσα στην εντολή android:hint είναι το email . Αυτό σημαίνει ότι σε αυτό το πεδίο ο χρήστης πρέπει να γράψει το email που θα χρησιμοποιήσει στην εφαρμογή. Υπάρχει ακόμα μία εντολή η οποία χαρακτηρίζει τον τύπο εισόδου του edittext. Σε αυτή την περίπτωση το edittext μπορεί να δεχτεί κείμενο που έχει τα χαρακτηριστικά ενός email. Οι υπόλοιπες εντολές χρησιμοποιούνται για να προσδιοριστεί το σημείο που θα έχει στο layout. Τρίτο και τελευταίο materialedittext είναι αυτό το οποίο έχει όνομα password όπως φαίνεται και στην εντολή android:id. Επίσης η εντολή android:hint γράφει password το οποίο σημαίνει ότι ο χρήστης πρέπει να πληκτρολογήσει τον κωδικό τον οποίο θέλει να χρησιμοποιεί. Η εντολή android:InputType στην οποία δηλώνεται ότι το στοιχείο αυτό δείχνει τα σύμβολα που δέχεται όπως όταν πληκτρολογείτε ένας κωδικός. Οι υπόλοιπες εντολές είναι για τη στοίχιση του στο layout. Τελευταίο στοιχείο που χρησιμοποιείται για το layout αυτό είναι ένα button. Το όνομα που έχει στο layout είναι btn_register. Επίσης χρησιμοποιείται η εντολή android:text η οποία γράφει register. Ακόμα ορίζεται και το χρώμα των γραμμάτων του κουμπιού. Τέλος όλες οι υπόλοιπες εντολές χαρακτηρίζουν τις συντεταγμένες του button πάνω στο layout.

5.3.3 Activity Login Επόμενο layout που θα αναλυθεί είναι αυτό που ονομάζεται activity_login.

Αυτό φαίνεται και στις παρακάτω εικόνες.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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=".Login"
    >


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"

        android:orientation="vertical"
        android:padding="16dp"
        android:gravity="center_horizontal"
        >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Login"
            android:textSize="20sp"
            android:textStyle="bold"/>
```

FIGURE 16 : ACTIVITY LOGIN (ΕΙΚΟΝΑ 1)

```
<com.rengwuxian.materialedittext.MaterialEditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/email_login"
    android:inputType="textEmailAddress"
    android:layout_marginTop="10dp"
    app:met_floatingLabel="normal"
    android:hint="Email"/>
<com.rengwuxian.materialedittext.MaterialEditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/password_login"
    android:inputType="textPassword"
    android:layout_marginTop="10dp"
    app:met_floatingLabel="normal"
    android:hint="Password"/>
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Login"
    android:id="@+id/btn_login"
    android:backgroundTint="@color/material_on_primary_disabled"
    android:textColor="@color/black"
    android:layout_margin="10dp"/>
</LinearLayout>
```

FIGURE 17 : ACTIVITY LOGIN (EIKONA 2)

Σε αυτές τις εικόνες φαίνεται το πώς έχει σχεδιαστεί η εφαρμογή στην σελίδα του login. Αρχικά με τις εντολές xmlns επιτρέπει στο layout να χρησιμοποιήσει εντολές τύπου android : και τύπου tools:. Στη συνέχεια με τις εντολές android:layout_width και android:layout_height ορίζουν το πλάτος και το ύψος του layout. Για να κλείσει η ανάλυση τουλάχιστον για τα στοιχεία που ορίζουν τις προδιαγραφές του layout υπάρχει η τελευταία εντολή η οποία έχει τη μορφή tools:context και ορίζει σε ποια σελίδα απευθύνεται το συγκεκριμένο layout.

Επόμενα στοιχεία τα οποία θα πρέπει να αναλυθούν για το activity_login είναι τα στοιχεία τα οποία βλέπει ο χρήστης στην οθόνη. Στο layout αυτό υπάρχουν ένα textview , δύο materialedittext και ένα button . πρώτο στοιχείο όπως φαίνεται και στην εικόνα είναι το textview. Το textview αυτό έχει δίνει στο χρήστη το σκοπό αυτής της σελίδας με το κείμενο που υπάρχει μέσα στην εντολή android:text. Επίσης υπάρχουν εντολές που ορίζουν το μέγεθος των γραμμάτων του textview αλλά και το style τους . το πρώτο γίνεται με την εντολή android:textSize και το δεύτερο με την εντολή android:textStyle. Στην πρώτη εντολή έχουμε την τιμή 20sp ενώ στην δεύτερη την τιμή bold. Οι υπόλοιπες εντολές είναι για να χαρακτηρίσουν τις συντεταγμένες του πάνω στο layout. Επόμενο στοιχείο είναι το πρώτο materialedittext. Σε αυτό έχει δοθεί η ονομασία email_login. Επίσης στην εντολή android:hint αυτό που γράφεται είναι η λέξη email που σημαίνει ότι εδώ ο χρήστης θα πρέπει να πληκτρολογήσει το email το οποίο είχε δώσει κατά την διαδικασία της εγγραφής του στο σύστημα. Ακόμα μία εντολή η οποία δείχνει ότι αυτό το στοιχείο παίρνει

email είναι η εντολή android:inputType η οποία έχει την τιμή textEmailAddress. Οι υπόλοιπες εντολές είναι για να δείξουν το σημείο του layout που είναι τοποθετημένο το στοιχείο αυτό. Δεύτερο materialedittext είναι αυτό το οποίο ονομάζεται password_login . επίσης στην εντολή android:hint έχει την λέξη password η οποία δείχνει ότι ο χρήστης πρέπει να βάλει το κωδικό που είχε δηλώσει κατά την εγγραφή του. Ακόμα η εντολή android:inputText δείχνει ότι το edittext αυτό παίρνει ως είσοδο τα χαρακτηριστικά του password. Οι υπόλοιπες εντολές χρησιμοποιούνται ως συντεταγμένες αυτού του στοιχείου πάνω στο layout. Τελευταίο στοιχείο του layout είναι το button. Το button αυτό ονομάζεται btn_login σύμφωνα με την εντολή android:id. Επίσης με την εντολή text το κείμενο που φαίνεται στο button είναι login. Οι υπόλοιπες εντολές δείχνουν τα σημεία που είναι τοποθετημένο το button στο layout.

5.3.4 Actor List Item

Επόμενο layout το οποίο θα αναλυθεί είναι το actor_list_item. Το layout αυτό απεικονίζεται παρακάτω.

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

    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:cardElevation="10dp"
        android:layout_margin="10dp"
        app:cardCornerRadius="10dp"
        app:cardBackgroundColor="@color/white">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">
            <de.hdodenhof.circleimageview.CircleImageView
                android:src="@drawable/ic_launcher_background"
                android:layout_margin="10dp"
                android:id="@+id/img1"
                android:layout_width="120dp"
                android:layout_height="150dp"/>
```

FIGURE 18 : ACTOR LIST ITEM (ΕΙΚΟΝΑ 1).

```
<LinearLayout
    android:layout_gravity="center"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >
    <TextView
        android:id="@+id/nametext"
        android:textColor="@color/black"
        android:textStyle="bold"
        android:textSize="25sp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:text="Actor Name"
        />

    <TextView
        android:id="@+id/birthdaytext"
        android:textColor="@color/black"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:text="Birth of Date"
        android:textStyle="bold"
        android:textSize="10sp"
        />
```

FIGURE 19 : ACTOR LIST ITEM

Στις εικόνες παραπάνω φαίνεται ο τρόπος με τον οποίο ένας ηθοποιός απεικονίζεται στη λίστα που έχει η σελίδα όταν ο χρήστης πατήσει το κουμπί Actors. Αρχικά με τις εντολές τύπου xmlns επιτρέπει στο layout να χρησιμοποιήσει εντολές τύπου android: και τύπου app: . Τέλος με τις εντολές android: layout_width και android: layout_height ορίζουν το πλάτος και το ύψος του layout.

Επόμενα στοιχεία που πρέπει να αναλυθούν για αυτό το layout είναι τα στοιχεία τα οποία βλέπει ο χρήστης στην οθόνη. Σε αυτό το layout υπάρχουν ένα imageview και δύο textview. Το imageview έχει το όνομα img1 σύμφωνα με την εντολή android:id. Επίσης έχει μία default εικόνα που έχει χρησιμοποιηθεί για να φαίνεται το σημείο το οποίο είναι στο layout. Αυτό φαίνεται από την εντολή android:src. Οι υπόλοιπες εντολές δείχνουν τις συντεταγμένες που έχει πάνω στο layout. Επόμενο στοιχείο το οποίο θα αναλυθεί είναι το πρώτο textview που υπάρχει στο κώδικα. Το στοιχείο αυτό έχει το όνομα nametext στο layout σύμφωνα με την εντολή android:id. Εκεί φαίνεται το όνομα του κάθε ηθοποιού με γράμματα μεγέθους 25sp και τα οποία είναι bold . Αυτά φαίνονται στις εντολές android:textSize και android:textStyle αντίστοιχα. Οι υπόλοιπες εντολές ορίζουν τις συντεταγμένες τις οποίες έχει το textview στο layout. Τελευταίο σημείο του layout το

οποίο θα αναλυθεί είναι το δεύτερο textview. Αυτό το textview έχει το όνομα birthdaytext . Σε αυτό το σημείο φαίνεται η ημερομηνία γέννησης του κάθε ηθοποιού με μέγεθος γραμμάτων 10sp και bold να είναι οι χαρακτήρες. Αυτά φαίνονται στις εντολές android:textSize android:textStyle . οι υπόλοιπες εντολές δείχνουν τις συντεταγμένες τις οποίες έχει το textview στο layout.

5.3.5 Movie List Item

Επόμενο layout το οποίο θα αναλυθεί είναι αυτό του movie_list_item. Αυτό απεικονίζεται παρακάτω.

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

    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:cardElevation="10dp"
        android:layout_margin="10dp"
        app:cardCornerRadius="10dp"
        app:cardBackgroundColor="@color/white">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">
            <de.hdodenhof.circleimageview.CircleImageView
                android:src="@drawable/ic_launcher_background"
                android:layout_margin="10dp"
                android:id="@+id/imagemovie"
                android:layout_width="80dp"
                android:layout_height="80dp"/>
```

FIGURE 20 : MOVIE LIST ITEM (EIKONA 1).

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="156dp"
    android:layout_gravity="center"
    android:orientation="vertical">

    <TextView
        android:id="@+id/namemovie"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:text="Movie Name"
        android:textColor="@color/black"
        android:textSize="20sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/movieyear"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:text="Birth of Date"
        android:textColor="@color/black"
        android:textSize="15sp"
        android:textStyle="bold" />
```

FIGURE 21 : MOVIE LIST ITEM (EIKONA 2)

```
        <RatingBar
            android:id="@+id/ratingBar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />


    </LinearLayout>
</LinearLayout>
</androidx.cardview.widget.CardView>
```

FIGURE 22 : MOVIE LIST ITEM (EIKONA 3)

Στις παραπάνω εικόνες φαίνεται ο τρόπος με τον οποίο μία ταινία απεικονίζεται στη λίστα που έχει η σελίδα όταν ο χρήστης πατήσει το κουμπί Movies. Αρχικά με τις εντολές xmlns επιτρέπει στο layout να χρησιμοποιήσει τις εντολές τύπου android: και τύπου app: .Τέλος με τις εντολές android:layout_width και android:layout_height ορίζουν το πλάτος και το ύψος του layout.

Επόμενα στοιχεία που πρέπει να αναλυθούν για αυτό το layout είναι τα στοιχεία τα οποία βλέπει ο χρήστης στην οθόνη. Σε αυτό το layout υπάρχουν ένα imageview δύο textview και ένα ratingbar. Το imageview έχει το όνομα imagename σύμφωνα με την εντολή android:id. Επίσης έχει μία default εικόνα που έχει χρησιμοποιηθεί για να φαίνεται το σημείο που είναι στο layout. Αυτό φαίνεται από την εντολή android:src. Οι υπόλοιπες εντολές δείχνουν τις συντεταγμένες που έχει πάνω στο layout. Επόμενο στοιχείο το οποίο θα αναλυθεί είναι το πρώτο textview που υπάρχει στο κώδικα. Το στοιχείο αυτό έχει το όνομα namemovie στο layout σύμφωνα με την εντολή android:id. Εκεί φαίνεται το όνομα της κάθε ταινίας με γράμματα μεγέθους 20sp και τα οποία είναι bold. Αυτά φαίνονται στις εντολές android:textSize και android:textStyle αντίστοιχα. Οι υπόλοιπες εντολές ορίζουν τις συντεταγμένες τις οποίες έχει το textview στο layout. Το δεύτερο textview το οποίο θα αναλυθεί έχει το όνομα movieyear σύμφωνα με την εντολή android:id. Σε αυτό το σημείο φαίνεται η χρονιά που προβλήθηκε η ταινία για πρώτη φορά, με μέγεθος γραμμάτων 15sp και bold οι χαρακτήρες. Αυτά φαίνονται στις εντολές android:textSize και android:textStyle. Οι υπόλοιπες εντολές δείχνουν τις συντεταγμένες τις οποίες έχει το textview στο layout. Τελευταίο στοιχείο είναι το ratingbar. Το όνομα σύμφωνα με την εντολή android:id είναι ratingBar. Οι υπόλοιπες εντολές δείχνουν τις συντεταγμένες που έχει στο layout.

5.3.6 Toolbar

Επόμενο layout το οποίο θα αναλυθεί είναι το toolbar. Η εικόνα παρακάτω απεικονίζει ακριβώς αυτό.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.Toolbar xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:background="@color/design_default_color_primary"
    android:elevation="4dp"
    android:theme="@style/Theme.MaterialComponents.DayNight.NoActionBar"
    app:popupTheme="@style/ThemeOverlay.AppCompat.Light">
    <de.hdodenhof.circleimageview.CircleImageView
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:id="@+id/imageView"
        android:src="@mipmap/ic_launcher"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/username_arxiki"
        android:text="gjhjhg"
        android:textColor="@color/white"/>

</androidx.appcompat.widget.Toolbar>
```

FIGURE 23 : TOOLBAR

Στη παραπάνω εικόνα φαίνονται τα στοιχεία τα οποία διαθέτει το toolbar μετά από το login. Αρχικά με τις εντολές τύπου xmlns επιτρέπει στο layout να χρησιμοποιήσει εντολές τύπου android: και app:. Με τις εντολές android : layout_width android : layout_height ορίζουν το πλάτος και το ύψος του layout.

Επόμενα στοιχεία που θα πρέπει να αναλυθούν για αυτό το layout είναι τα στοιχεία τα οποία βλέπει ο χρήστης στην οθόνη. Σε αυτό το layout είναι ένα circleimageview και ένα textview. Το circleimageview έχει το όνομα imageView σύμφωνα με την εντολή android:id. Επίσης υπάρχει

μία default εικόνα για να δείχνει το σημείο το οποίο είναι τοποθετημένη στο layout. Αυτό φαίνεται με την εντολή android:src. Οι υπόλοιπες εντολές δείχνουν τις συντεταγμένες που έχει πάνω στο layout. Τελευταίο στοιχείο το οποίο θα αναλυθεί είναι το textview το οποίο υπάρχει. Αυτό το textview έχει το όνομα username_arxiki. Οι υπόλοιπες εντολές δείχνουν τις συντεταγμένες του πάνω στο layout.

5.3.7 Activity Arxiki Selida

Επόμενο layout το οποίο θα αναλυθεί είναι το activity_arxiki. Το layout αυτό απεικονίζεται παρακάτω.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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=".Arxiki_Selida">

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

    <Button
        android:id="@+id/actorsbtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="143dp"
        android:text="Actors"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/toolbar" />
```

FIGURE 24 : ACTIVITY ARXIKI_SELIDA (EIKONA 1)

```
    <Button
        android:id="@+id/moviesbtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="300dp"
        android:text="Movies"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.501"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/toolbar" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

FIGURE 25 : ACTIVITY ARXIKI_SELIDA (EIKONA 2)

Στις παραπάνω εικόνες φαίνεται η σελίδα που βλέπει ο χρήστης όταν έχει ήδη κάνει login στο σύστημα. Αρχικά με τις εντολές τύπου xmlns επιτρέπονται στο layout εντολές τύπου android:, τύπου app: και τύπου tools:. Τέλος με τις εντολές android:layout_width και android:layout_height ορίζουν το πλάτος και το ύψος του layout.

Επόμενα στοιχεία τα οποία θα πρέπει να αναλυθούν για αυτό το layout είναι τα στοιχεία που βλέπει ο χρήστης στην οθόνη. Με την εντολή include πάνω στην οθόνη το layout το οποίο αναφέρεται στην εντολή με το ίδιο όνομα. Επόμενο στοιχείο το οποίο θα αναλυθεί είναι το button με το όνομα actorsbtn σύμφωνα με την εντολή android:id. Το button αυτό γράφει Actors και χρησιμοποιείται για να πάει ο χρήστης στους ηθοποιούς. Οι υπόλοιπες εντολές δείχνουν τις συντεταγμένες του στο layout. Τελευταίο στοιχείο το οποίο θα αναλυθεί είναι το δεύτερο button το οποίο ονομάζεται moviesbtn σύμφωνα με την εντολή android:id. Το button αυτό γράφει Movies και χρησιμοποιείται για να πάει ο χρήστης στις ταινίες. Οι υπόλοιπες εντολές δείχνουν τις συντεταγμένες του στο layout.

5.3.8 Activity Actors

Επόμενο layout το οποίο θα αναλυθεί είναι το activity_actors. Το layout αυτό απεικονίζεται παρακάτω.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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=".Actors">
    <include
        android:id="@+id/toolbar"
        layout="@layout/toolbar"/>

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerview"
        android:layout_width="406dp"
        android:layout_height="649dp"
        android:layout_marginStart="1dp"
        android:layout_marginEnd="1dp"
        android:layout_marginBottom="2dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

FIGURE 26 : ACTIVITY ACTORS

Στην παραπάνω εικόνα φαίνεται το activity_actors. Αρχικά με τις εντολές τύπου xmlns επιτρέπει να χρησιμοποιηθούν στο layout εντολές τύπου android:, τύπου app: και τύπου tools:. Με τις εντολές android:layout_width και android:layout_height ορίζουν το πλάτος και το ύψος του layout.

Επόμενα στοιχεία τα οποία θα πρέπει να αναλυθούν για αυτό το layout είναι δύο. Η εντολή include η οποία προσθέτει στο layout το layout το layout το οποίο αναφέρεται στην ομώνυμη εντολή. Το τελευταίο στοιχείο το οποίο πρέπει να αναλυθεί για αυτό το layout είναι το recyclerview. Αυτή η εντολή χρησιμοποιείται για να προσθέσει όλα τα στοιχεία τα οποία παίρνει ο κώδικας από τη βάση και προστίθενται στο layout actor_list_item.Το όνομα το οποίο έχει στο layout αυτό που αναλύεται είναι recyclerview. Οι υπόλοιπες εντολές δείχνουν τις συντεταγμένες του πάνω στο layout .

5.3.9 Activity Movies

Επόμενο layout το οποίο θα αναλυθεί είναι το activity_movies. Το layout αυτό απεικονίζεται παρακάτω.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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=".Movies">
    <include
        android:id="@+id/toolbar"
        layout="@layout/toolbar"/>
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerviewMovies"
        android:layout_width="406dp"
        android:layout_height="649dp"
        android:layout_marginStart="1dp"
        android:layout_marginEnd="1dp"
        android:layout_marginBottom="2dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

FIGURE 27 : ACTIVITY MOVIES

Στη παραπάνω εικόνα φαίνεται το activity_movies. Αρχικά με τις εντολές τύπου xmlns επιτρέπονται στο layout εντολές τύπου android:, τύπου tools: και τύπου app:. Τέλος με τις εντολές που έχουν την μορφή android:layout_width και android:layout_height ορίζουν το πλάτος και το ύψος του layout.

Επόμενα στοιχεία τα οποία θα πρέπει να αναλυθούν για αυτό το layout είναι δύο. Η εντολή include η οποία προσθέτει στο layout το layout το οποίο αναφέρεται στην ομώνυμη εντολή. Το τελευταίο στοιχείο το οποίο πρέπει να αναλυθεί για αυτό το layout είναι το recyclerview. Αυτή η εντολή χρησιμοποιείται για να προσθέσει όλα τα στοιχεία τα οποία παίρνει ο κώδικας από τη βάση και προστίθενται στο layout movies_list_item. Το όνομα το οποίο έχει στο layout το οποίο αναλύεται είναι το recyclerviewMovies. Οι υπόλοιπες εντολές δείχνουν τις συντεταγμένες τις οποίες έχει στο layout.

5.3.10 Activity Full Screen Movies

Επόμενο layout το οποίο θα αναλυθεί είναι το activity_fullscreen_movies. Το layout αυτό απεικονίζεται παρακάτω.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView 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=".FullScreenMovies">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <com.google.android.exoplayer2.ui.SimpleExoPlayerView
            android:id="@+id/exo_player_view"
            android:layout_width="match_parent"
            android:layout_height="230dp"
            android:layout_margin="3dp"
            android:layout_marginStart="3dp"
            android:layout_marginEnd="3dp"
            app:layout_constraintEnd_toStartOf="@+id/tv_fullscreen"
            app:layout_constraintStart_toEndOf="@+id/tv_fullscreen"
            app:layout_constraintTop_toTopOf="parent"
            app:resize_mode="fill"
            app:use_controller="true" />
```

FIGURE 28 : ACTIVITY FULL SCREEN MOVIES (ΕΙΚΟΝΑ 1)

```
            <ScrollView
                android:id="@+id/SCROLLER_ID"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:scrollbars="vertical"
                android:fillViewport="true">
                <TextView
                    android:id="@+id/tv_fullscreen"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="3dp"

                    android:text="name"
                    android:scrollbarAlwaysDrawVerticalTrack="true"
                    android:textSize="15sp"
                    app:layout_constraintEnd_toEndOf="parent"
                    app:layout_constraintStart_toStartOf="parent"
                    app:layout_constraintTop_toBottomOf="@+id/exo_player_view" />
            </ScrollView>
    </LinearLayout>
</androidx.cardview.widget.CardView>
```

FIGURE 29 : ACTIVITY FULL SCREEN MOVIES (ΕΙΚΟΝΑ 2)

Στις παραπάνω εικόνες φαίνεται η σελίδα όταν ο χρήστης πατήσει να δει επιπλέον πληροφορίες για την ταινία που θέλει. Αρχικά με τις εντολές τύπου xmlns επιτρέπονται στο layout να χρησιμοποιούνται εντολές τύπου android: , τύπου app: και τύπου tools:. Τέλος με τις εντολές android:layout_width και την εντολή android:layout_height ορίζουν το πλάτος και το ύψος του layout.

Επόμενα στοιχεία τα οποία θα αναλυθούν για αυτό το layout είναι τα στοιχεία τα οποία βλέπει ο χρήστης στην οθόνη. Σε αυτό το layout υπάρχει ένα simpleexoplayer και ένα textview. Το simpleexoplayer έχει το όνομα exo_player_view σύμφωνα με την εντολή android:id. Οι

υπόλοιπες εντολές δείχνουν τις συντεταγμένες του πάνω στο layout. Τέλος το textview έχει το όνομα tv_fullscreen και είναι τύπου scrollview. Το scrollview έχει το όνομα στο layout scrolled_id όπως φαίνεται στην εντολή android:id. Οι υπόλοιπες εντολές είναι για τις συντεταγμένες layout.

5.3.11 Activity Full Screen

Επόμενο layout το οποίο θα αναλυθεί είναι το activity_fullscreen . Το layout αυτό απεικονίζεται παρακάτω.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView 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=".FullScreen"
    app:cardUseCompatPadding="true"
    app:cardCornerRadius="4dp">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <ImageView
            android:layout_width="match_parent"
            android:layout_height="230dp"
            android:id="@+id/exo_player_view"
            android:layout_margin="3dp"
            />
```

FIGURE 30 : ACTIVITY FULL SCREEN (EIKONA 1)

```
<ScrollView
    android:id="@+id/SCROLLER_ID"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:scrollbars="vertical"
    android:fillViewport="true">

    <TextView

        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1.0"
        android:text="Actor Description"
        android:id="@+id/a_description"
        />
</ScrollView>


</LinearLayout>

</androidx.cardview.widget.CardView>
```

FIGURE 31 : ACTIVITY FULL SCREEN (EIKONA 2)

Σε αυτές τις εικόνες φαίνεται το πώς έχει σχεδιαστεί η εφαρμογή όταν ο χρήστης θέλει να δει επιπλέον στοιχεία για κάποιο από τους ηθοποιούς. Αρχικά με τις εντολές xmlns επιτρέπει στο layout να χρησιμοποιήσει εντολές τύπου android: και τύπου tools :. Στη συνέχεια με τις εντολές android:layout_width και android:layout_height ορίζουν το πλάτος και το ύψος του layout. Για να κλείσει η ανάλυση τουλάχιστον για τα στοιχεία που ορίζουν τις προδιαγραφές του layout υπάρχει η τελευταία εντολή η οποία έχει τη μορφή tools:context και ορίζει σε ποια σελίδα απευθύνεται το συγκεκριμένο layout.

Σημαντικό είναι επίσης να γίνει αναφορά στα στοιχεία που βλέπει ο χρήστης στην οθόνη. Τα στοιχεία που βλέπει είναι μία εικόνα και το βιογραφικό. Αυτά στο layout είναι ένα imageview και ένα textview. Το textview είναι μέσα σε ένα scrollview το οποίο βοηθάει να μπορεί να κάνει scroll αν το βιογραφικό έχει πολλά στοιχεία. Το imageview έχει την ονομασία exo_player_view σύμφωνα με την εντολή android:id. Οι υπόλοιπες εντολές δείχνουν τις συντεταγμένες του στο layout. Η scroll δυνατότητα εμφανίζεται με το χαρακτηριστικό scrollview και έχει το όνομα scrolled_id. Τα υπόλοιπα είναι εντολές για τη θέση που έχει στην οθόνη. Τέλος το textview έχει την ονομασία a_description σύμφωνα με την εντολή android:id. Επίσης υπάρχει η εντολή android:text η οποία έχει την φράση που φαίνεται και εκεί πάντα στην οθόνη του χρήστη φαίνεται το βιογραφικό του ηθοποιού που έχει πατηθεί. Οι υπόλοιπες εντολές δείχνουν το σημείο που βρίσκεται στο layout και την οθόνη.

5.4. Βασικά Αρχεία

5.4.1 Main Activity

Σε αυτό το κεφάλαιο θα γίνει η παρουσίαση της εργασίας αναλυτικά. Αρχικά στην εφαρμογή έχουμε την αρχική σελίδα. Εκεί ο χρήστης διαλέγει αν θέλει να κάνει login : 1. Με το λογαριασμό που διαθέτει στην Google

2. Με τον τρόπο της εφαρμογής.

Επίσης αν δεν θέλει να κάνει login με το λογαριασμό της Google ή αν δεν έχει λογαριασμό στην Google και είναι νέος χρήστης της εφαρμογής θα χρειαστεί να κάνει signup στην εφαρμογή με τα στοιχεία που ζητάει. Σε αυτό το σημείο θα υπάρχει ανάλυση του κώδικα.

```
package com.example.telikiergasia;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import com.google.android.gms.auth.api.signin.GoogleSignIn;
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;
import com.google.android.gms.auth.api.signin.GoogleSignInClient;
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;
import com.google.android.gms.common.SignInButton;
import com.google.android.gms.common.api.ApiException;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthCredential;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.auth.GoogleAuthProvider;
```

FIGURE 32 : PACKAGE ΚΑΙ ΒΙΒΛΙΟΘΗΚΕΣ

Στην παραπάνω φωτογραφία αυτό που απεικονίζεται είναι τα εξής δύο πράγματα. Το πρώτο είναι το package δηλαδή η διαδρομή την οποία ακολουθεί ο προγραμματιστής για να ανοίξει το αρχείο. Το δεύτερο στοιχείο είναι οι βιβλιοθήκες. Οι βιβλιοθήκες στη Java ρέουν από τα packages. Στην Java υπάρχουν TONS που μας επιτρέπουν να κάνουμε πράγματα όπως:

- Να διαβάσει κάποιος τα περιεχόμενα ενός αρχείου / Δημιουργία ενός αρχείου και συμπλήρωση των περιεχομένων
- Σύγκριση ημερομηνιών μεταξύ τους
- Αποστολή μηνυμάτων ηλεκτρονικού ταχυδρομείου

Το import android. widget. Button δηλώνει ότι θα χρησιμοποιήσουμε τη βιβλιοθήκη για τα Buttons.

Στη συνέχεια θα απεικονιστεί ο τρόπος με τον οποίο δηλώνει κάποιος μεταβλητές στο Android Studio. Αυτό απεικονίζεται στην παρακάτω εικόνα.

```
Button buttonlogin,buttonsignup;
private SignInButton signInButton;
private GoogleSignInClient mG00gleSignInClient;
private String TAG="MainActivity";
private FirebaseAuth mAuth;
private Button btnSignOut;
private int RC_SIGN_IN=1;
```

FIGURE 33 : ΔΗΛΩΣΗ ΜΕΤΑΒΛΗΤΩΝ

Αρχικά κάποιος πρέπει να δηλώσει τα στοιχεία που έχει χρησιμοποιήσει στο layout. Σε αυτήν την κλάση τα μόνα που έχουν χρησιμοποιηθεί είναι τρία κουμπιά. Αυτά είναι το buttonlogin, buttonsignup και το signInButton. Οι υπόλοιπες μεταβλητές είναι μεταβλητές που δηλώνονται επειδή χρειάζονται στο πρόγραμμα και όχι επειδή υπάρχουν και κάπου αλλού. Μία private μεταβλητή είναι μεταβλητή η οποία δεν μπορεί να φαίνεται στις υπόλοιπες κλάσεις του προγράμματος δηλαδή όποια αλλαγή γίνει θα είναι σε αυτή την κλάση και μόνο.

Στη συνέχεια αυτό που έχει υλοποιηθεί στο πρόγραμμα είναι η αρχικοποίηση των μεταβλητών . Στην παρακάτω εικόνα φαίνεται η αρχικοποίηση των μεταβλητών.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    buttonlogin=findViewById(R.id.btn_login);
    buttonsignup=findViewById(R.id.btn_signup);
    signInButton=findViewById(R.id.signInButton);

    mAuth=FirebaseAuth.getInstance();
```

FIGURE 34 : ΑΡΧΙΚΟΠΟΙΗΣΗ ΜΕΤΑΒΛΗΤΩΝ

Ο ορισμός της γραμμής buttonlogin=findViewById(R.id.btn_login) είναι ο εξής:

Η μεταβλητή buttonlogin βρίσκεται στο layout με το όνομα activity_main που έχει δηλωθεί στη παραπάνω γραμμή με το όνομα btn_login.

Το ίδιο συμβαίνει και με τις επόμενες δύο γραμμές .Το buttonsignup αλλά και το signInButton ορίζονται στο layout που δηλώνει το setContentView με τα ονόματα που υπάρχουν στο findViewById, δηλαδή είναι το btn_signup και το signInButton αντίστοιχα. Η τελευταία γραμμή δηλώνει ότι αρχικοποιείται το Firebase Auth.

Επίσης σε αυτή την κλάση της εφαρμογής υπάρχει και άλλο ένα μέρος του κώδικα που ουσιαστικά ενεργοποιεί τις λειτουργίες του κάθε κουμπιού. Αυτό παρουσιάζει και η παρακάτω εικόνα.

```
buttonsignup.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent=new Intent(MainActivity.this,SignUp.class);

        String item=String.valueOf(false);
        intent.putExtra("boolean",item);
        startActivity(intent);

    }
});
buttonlogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent=new Intent(MainActivity.this,Login.class);

        String item=String.valueOf(false);
        intent.putExtra("boolean",item);
        startActivity(intent);

    }
});
GoogleSignInOptions gso=new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAUL
        .requestIdToken(getString(R.string.default_web_client_id))
        .requestEmail()
        .build();
mG00gleSignInClient= GoogleSignIn.getClient(this,gso);
signInButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        signIn();
    }
});
```

FIGURE 35 : ΛΕΙΤΟΥΡΓΙΕΣ ΓΙΑ ΚΑΘΕ BUTTON

Σε αυτή την εικόνα έχουν υλοποιηθεί οι λειτουργίες για τα κουμπιά signInButton, buttonsignup και το buttonlogin. Για το πρώτο κουμπί που φαίνεται στην εικόνα το buttonsignup έχουμε αρχικά την εντολή .setOnClickListener(new View.OnClickLister) η οποία ουσιαστικά ενεργοποιεί το πάτημα του κουμπιού και στις παρακάτω γραμμές δηλώνει ο προγραμματιστής τις λειτουργίες που θα κάνει το κουμπί. Στην συνάρτηση onClick που δημιουργείται με την εντολή .setOnClickListener(new View.OnClickListener) υπάρχον οι εντολές :

- Intent intent=new Intent (MainActivity.this, Signup. Class);
- String item=String.valueOf(false);
- Intent.putExtra(“Boolean”, item) και την εντολή
- startActivity(intent).

Η πρώτη εντολή είναι αυτή που δηλώνει ότι όταν ο χρήστης πατήσει το κουμπί θα πάει η εφαρμογή από την κλάση MainActivity στην SignUp κλάση. Η δεύτερη και η Τρίτη εντολή χρησιμοποιείται για την δημιουργία μίας μεταβλητής τύπου String η οποία ονομάζεται item και έχει πάρει την τιμή false. Με την Τρίτη εντολή στέλνουμε την τιμή της μεταβλητής item στην κλάση SignUp. Η τέταρτη και τελευταία εντολή ενεργοποιεί την πρώτη εντολή και από την κλάση που βρίσκεται τώρα η εφαρμογή πάει στην κλάση που έχει δηλωθεί για δεύτερη.

Για το επόμενο button της εικόνας το buttonlogin έχουμε τις ίδιες εντολές μέσα στην συνάρτηση onClick με το προηγούμενο κουμπί μόνο που τώρα όταν πατηθεί το κουμπί θα πάει η εφαρμογή στην κλάση Login.

Στην συνέχεια έχουμε την εντολή η οποία ξεκινάει με το GoogleSignInOptions . Αυτή η εντολή δημιουργεί έναν Builder ο οποίος καθορίζει μία από τις προεπιλεγμένες διαμορφώσεις που παρέχει η Google και κάνει τις πρόσθετες διαμορφώσεις βάση αυτού. Η επόμενη εντολή η οποία είναι η GoogleSignIn.getClient δημιουργεί ένα instance του GoogleSignIn. Υπάρχουν ακόμα οι εντολές για το τρίτο και τελευταίο κουμπί της κλάσης αυτής το SignInButton.

Για αυτό το τρίτο κουμπί έχουμε πάλι τη δημιουργία της συνάρτησης OnClick μέσω της εντολής .setOnClickListener(View.OnClickListener) και στη συνάρτηση αυτή το μόνο που γίνεται είναι να καλείται μία συνάρτηση που ονομάζεται signIn που δημιουργείται και που θα αναλυθεί στη συνέχεια.

Για να τελειώσει η ανάλυση αυτής της κλάσης πρέπει να αναλυθούν και οι συναρτήσεις οι οποίες έχουν δημιουργηθεί. Οι συναρτήσεις αυτές είναι πέντε και ονομάζονται signIn, on ActivityResult, handleSignInResult, firebaseGoogleAuth και updateUI. Οι συναρτήσεις αυτές απεικονίζονται στις παρακάτω εικόνες.

```
private void signIn() {
    Intent signInIntent=mG00gleSignInClient.getSignInIntent();
    startActivityForResult(signInIntent,RC_SIGN_IN);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode==RC_SIGN_IN)
    {
        Task<GoogleSignInAccount> task=GoogleSignIn.getSignedInAccountFromIntent(data);
        handleSignInResult(task);
    }
}

private void handleSignInResult(Task<GoogleSignInAccount> task) {
    try {
        GoogleSignInAccount acc=task.getResult(ApiException.class);
        Toast.makeText(this, "Login Successfully", Toast.LENGTH_SHORT).show();
        FirebaseGoogleAuth(acc);
    } catch (ApiException e) {
        Toast.makeText(this, "Login Failed", Toast.LENGTH_SHORT).show();
        FirebaseGoogleAuth(null);
    }
}
```

FIGURE 36: ΣΥΝΑΡΤΗΣΗ 1

```
private void updateUI(FirebaseUser user) {

    GoogleSignInAccount account=GoogleSignIn.getLastSignedInAccount(getApplicationContext());
    if (account!=null)
    {
        boolean boolean_i=true;
        String personName=account.getDisplayName();
        //String personGivenName=account.getGivenName();
        // String personFamilyName=account.getFamilyName();
        String personEmail=account.getEmail();
        // String personId=account.getId();
        String imgurl = account.getPhotoUrl().toString();
        Intent intent=new Intent(getApplicationContext(),Arxiki_Selida.class);
        intent.putExtra("image",imgurl);
        String item=String.valueOf(boolean_i);
        intent.putExtra("boolean",item);
        intent.putExtra("username",personName);
        startActivity(intent);

        Toast.makeText(this, personName+ personEmail, Toast.LENGTH_SHORT).show();

    }
}
```

FIGURE 37 : ΣΥΝΑΡΤΗΣΗ 2

```
private void FirebaseGoogleAuth(GoogleSignInAccount acc) {
    AuthCredential authCredential= GoogleAuthProvider.getCredential(acc.getIdToken(),null);
    mAuth.signInWithCredential(authCredential).addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful())
            {
                Toast.makeText(MainActivity.this, "Successful", Toast.LENGTH_SHORT).show();
                FirebaseUser user=mAuth.getCurrentUser();
                updateUI(user);
            }else {
                Toast.makeText(MainActivity.this, "Failed", Toast.LENGTH_SHORT).show();

                updateUI(null);
            }
        }
    });
}
```

FIGURE 38 : ΣΥΝΑΡΤΗΣΗ 3

Οι συναρτήσεις αυτές βοηθάνε να γίνουν κάποιες λειτουργίες στην εφαρμογή. Η συνάρτηση signIn χρησιμοποιείται για να γίνει το signIn στην εφαρμογή μέσω του λογαριασμού της Google.

5.4.2 User

Σε αυτό το σημείο καλό είναι πριν να συνεχίσει η ανάλυση με τις δύο επόμενες κλάσεις που έχουν δημιουργηθεί από αυτή την σελίδα να αναλυθεί μία πολύ σημαντική ενότητα. Αυτή είναι μία τάξη η οποία έχει φτιαχτεί στο πρόγραμμα και ονομάζεται User. Η τάξη αυτή έχει δημιουργηθεί με σκοπό να βοηθήσει τη βάση δεδομένων να αποθηκεύσει τα στοιχεία των χρηστών της εφαρμογής. Στην εικόνα που ακολουθεί απεικονίζονται οι μεταβλητές καθώς και οι συναρτήσεις constructors που χρησιμοποιούνται στην τάξη αυτή.

```
package com.example.telikiergasia;

public class User {
    String imageurl,username,email,password;

    public User(String imageurl, String username, String email, String password) {
        this.imageurl = imageurl;
        this.username = username;
        this.email = email;
        this.password = password;
    }

    public User() {
    }
```

FIGURE 39 : ΜΕΤΑΒΛΗΤΕΣ

Σε αυτή την φωτογραφία φαίνονται οι μεταβλητές που θα έχει στη βάση δεδομένων μία αποθήκευση τύπου User. Οι μεταβλητές αυτές είναι όλες τύπου String. Τα ονόματα αυτών των μεταβλητών είναι imageurl, username,email,password. Επίσης υπάρχουν και δύο συναρτήσεις οι οποίες είναι τύπου constructor. Οι συναρτήσεις αυτές βοηθάνε το πρόγραμμα να πάρει όλα τα στοιχεία που έχει πληκτρολογήσει ο χρήστης και να αποθηκευτούν με τη σειρά που υπάρχουν σαν ορίσματα στη βάση. Σε αυτή την περίπτωση η σειρά αποθήκευσης είναι όπως φαίνεται στην εικόνα δηλαδή imageurl,username,email και password. Στη συνέχεια και την επόμενη εικόνα απεικονίζονται όλες οι μέθοδοι getters and setters που έχουν χρησιμοποιηθεί στην τάξη αυτή.

```
public String getImageurl() {
    return imageurl;
}

public void setImageurl(String imageurl) {
    this.imageurl = imageurl;
}

public String getUsername() {
    return username;
}

public void setUsername(String username) {
    this.username = username;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}
```

FIGURE 40 : GETTERS AND SETTERS

Στην παραπάνω εικόνα φαίνονται όλες οι συναρτήσεις getters and setters για όλες τις μεταβλητές. Οι συναρτήσεις αυτές είναι ίδιου τύπου με τον τύπο της μεταβλητής δηλαδή σε αυτή την κλάση όλες οι συναρτήσεις είναι τύπου String. Οι συναρτήσεις αυτές βοηθούν το πρόγραμμα να παίρνει ή να δίνει μεμονωμένα στοιχεία του χρήστη ή και όλα μαζί. Κάθε φορά που ένας χρήστης πληκτρολογεί τα στοιχεία του χρησιμοποιείται είτε η συνάρτηση getter είτε η συνάρτηση setter αναλόγως αν είναι εγγραφή ή κάτι άλλο. Οι συναρτήσεις getters χρησιμοποιούνται για να γραφτεί κάτι στη βάση δεδομένων. Αντίθετα οι συναρτήσεις setters χρησιμοποιούνται για να τραβήξουμε κάτι από τη βάση δεδομένων. Αυτά ήταν και τα σημαντικά στοιχεία που έπρεπε να γνωρίζει κάποιος πριν να συνεχίσει στην ανάλυση από την κλάση MainActivity.

5.4.3 Sign Up

Η κλάση MainActivity έχει δημιουργήσει άλλες δύο κλάσεις. Αυτές είναι η SignUp και η Login. Αρχικά θα αναλυθεί η SignUp. Σε αυτή την κλάση υπάρχει όλη η λειτουργία της εγγραφής ενός χρήστη στο σύστημα για να μπορεί μετέπειτα να κάνει login στην εφαρμογή. Στην παρακάτω εικόνα μπορεί κάποιος να δει τις βιβλιοθήκες τις οποίες χρησιμοποιεί η κλάση αυτή.

```
package com.example.telikiergasia;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import android.Manifest;
import android.app.ProgressDialog;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.OnProgressListener;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;
import com.karumi.dexter.Dexter;
import com.karumi.dexter.PermissionToken;
import com.karumi.dexter.listener.PermissionDeniedResponse;
import com.karumi.dexter.listener.PermissionGrantedResponse;
import com.karumi.dexter.listener.PermissionRequest;
import com.karumi.dexter.listener.single.PermissionListener;
import com.rengwuxian.materialedittext.MaterialEditText;
import java.util.Random;
```

FIGURE 41 : ΒΙΒΛΙΟΘΗΚΕΣ

Αυτές είναι οι βιβλιοθήκες που έχουν χρησιμοποιηθεί για να μπορεί κάποιος να χρησιμοποιήσει όλο τον παρακάτω κώδικα για το signup της εφαρμογής. Χωρίς αυτές ο κώδικας που θα αναλυθεί παρακάτω δεν θα μπορούσε να τρέξει. Στη συνέχεια πρέπει κάποιος να δηλώσει τις μεταβλητές που χρησιμοποιήθηκαν για το συγκεκριμένο activity. Η παρακάτω εικόνα δείχνει όλες τις μεταβλητές που χρησιμοποιήθηκαν .

```
MaterialEditText username,email,password;
Button btn_register;
FirebaseAuth auth;

android.widget.ImageView ImageView;
Uri filepath;
```

FIGURE 42 : ΜΕΤΑΒΛΗΤΕΣ

Οι μεταβλητές αυτές θα χρησιμοποιηθούν για το signup που θα κάνει ο χρήστης. Όπως εύκολα μπορεί να διακρίνει κάποιος αυτά που θα ζητηθούν από τον χρήστη είναι τα εξής:

- username
- email
- password και
- image

Επίσης χρησιμοποιείται και ένα button για να μπορεί να γίνει το signup.Επόμενο βήμα είναι και πάλι η αρχικοποίηση των μεταβλητών. Στην εικόνα 17 υπάρχει η αρχικοποίηση των μεταβλητών. Στη συνέχεια θα υπάρχει και ανάλυση της αρχικοποίησης.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_sign_up);
    email=findViewById(R.id.email);
    password=findViewById(R.id.password);
    btn_register=findViewById(R.id.btn_register);
    username=findViewById(R.id.username);
    ImageView=findViewById(R.id.profile_image);
```

FIGURE 43 : ΑΡΧΙΚΟΠΟΙΗΣΗ ΜΕΤΑΒΛΗΤΩΝ

Όπως φαίνεται και στην εικόνα οι μεταβλητές που αρχικοποιήθηκαν είναι αυτές που αναφέρθηκαν και πιο πάνω. Το email,το password, το btn_register και το username έχουν το ίδιο όνομα και στο layout. Αντίθετα η εικόνα(imageview) στο layout ονομάζεται profile_image. Στην κλάση SignUp θα πρέπει να αναλυθεί επίσης η λειτουργία της εγγραφής που υπάρχει αλλά και κάποιες συναρτήσεις .

Η λειτουργία της εγγραφής απεικονίζεται στις επόμενες εικόνες. Επίσης φαίνεται και ο κώδικας που έχει χρησιμοποιηθεί για να μπορεί ο χρήστης να διαλέξει εικόνα για το profile του. Η λειτουργία της εγγραφής γίνεται μόνο όταν ο χρήστης συμπληρώσει όλα τα πεδία και πατήσει το κουμπί.

```
ImageView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Dexter.withActivity(SignUp.this)
                .withPermission(Manifest.permission.READ_EXTERNAL_STORAGE)
                .withListener(new PermissionListener() {
                    @Override
                    public void onPermissionGranted(PermissionGrantedResponse response) {
                        Intent galleryIntent=new Intent();
                        galleryIntent.setAction(Intent.ACTION_GET_CONTENT);
                        galleryIntent.setType("image/*");
                        startActivityForResult(galleryIntent,2);
                    }

                    @Override
                    public void onPermissionDenied(PermissionDeniedResponse response) {

                    }

                    @Override
                    public void onPermissionRationaleShouldBeShown(PermissionRequest permission, PermissionToken token) {

                    }
                }).check();
    }
});
```

FIGURE 44 : ΥΛΟΠΟΙΗΣΗ ΕΠΙΛΟΓΗΣ ΕΙΚΟΝΑΣ ΧΡΗΣΤΗ

```
auth=FirebaseAuth.getInstance();
btn_register.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        uploadtofirebase();
        String txt_email=email.getText().toString();
        String txt_password=password.getText().toString();
        String txt_username=username.getText().toString();

        if (TextUtils.isEmpty(txt_username)||TextUtils.isEmpty(txt_email)||TextUtils.isEmpty(txt_password)){
            Toast.makeText(SignUp.this, "All fields are required", Toast.LENGTH_SHORT).show();
        }else if (txt_password.length()<6)
        {
            Toast.makeText(SignUp.this, "password must be at least 6 characters", Toast.LENGTH_SHORT).show();
        }else {
            registerUser(txt_username,txt_email,txt_password);
        }


    }
});
```

FIGURE 45 : ΥΛΟΠΟΙΗΣΗ SIGN UP

Η υλοποίηση της επιλογής εικόνας χρήστη φαίνεται στην πάνω εικόνα. Στην πρώτη γραμμή της εικόνας υπάρχει η εντολή ImageView.setOnClickListener(new View.OnClickListener) και με αυτή την εντολή δημιουργείται η συνάρτηση OnClick. Σε αυτή τη συνάρτηση έχουμε τον κώδικα της εικόνας και η λειτουργία του είναι η εξής: Με την εντολή

Dexter

. withActivity (SignUp.this)

. withPermission

(Manifest.permission. READ_EXTERNAL_SORAGE). withListener

(new PermissionListener ()

ουσιαστικά αυτό που θέλει να πετύχει ο προγραμματιστής είναι να ζητήσει μία άδεια με την εντολή .withPermission μεταβιβάζοντας την απαιτούμενη άδεια. Με την εντολή

.withListener(PermissionListener()) ο προγραμματιστής παίρνει ΄την κατάσταση του permission.

Στη συνέχεια υπάρχουν τρεις συναρτήσεις:

- PermissionGranted (),
- onPermissionDenied και η
- onPermissionRationaleShouldBeShown

Η πρώτη συνάρτηση δηλαδή η PermissionGranted() θα κληθεί μόλις χορηγηθεί η άδεια (permission). Η δεύτερη συνάρτηση η onPermissionDenied θα κληθεί όταν απορριφθεί η άδεια. Μπορεί κάποιος να ελέγξει αν η άδεια απορρίφθηκε οριστικά χρησιμοποιώντας το response.isPermanentlyDenied. η Τρίτη και τελευταία συνάρτηση χρησιμοποιείται όταν ο χρήστης παραχωρήσει και αρνηθεί κάποια από τα permissions.

Στην εφαρμογή κώδικας υπάρχει μόνο στη πρώτη συνάρτηση. Ο κώδικας έχει σκοπό να πάρει την άδεια από τον χρήστη για να μπει στην gallery του κινητού του έτσι ώστε να διαλέξει μία φωτογραφία και θα αναλυθεί γραμμή γραμμή παρακάτω. Στην πρώτη γραμμή της συνάρτησης έχουμε την εντολή Intent galleryIntent=new Intent(); Με αυτή την εντολή ο προγραμματιστής δημιουργεί μία μεταβλητή τύπου Intent η οποία ονομάζεται galleyIntent. Οι επόμενες γραμμές έχουν τις εντολές :

- galleryIntent.setAction(Intent.ACTION_GET_INTENT)
- galleryIntent.setType(“image/*) και η
- startActivityForResult(galleryIntent,2)

Αυτές οι εντολές υπάρχουν στο πρόγραμμα έτσι ώστε ο χρήστης αφού δώσει την πρώτη φορά μόνο την άδεια του για να μπορεί η εφαρμογή να πάει στην gallery του κινητού του να μπορεί να διαλέξει την φωτογραφία που θέλει να χρησιμοποιήσει για φωτογραφία profile.

Όσο αφορά την επιλογή εικόνας για το profile του χρήστη η ανάλυση τελείωσε. Επόμενο στην ανάλυση για την κλάση SignUp είναι η λειτουργία του btn_register που απεικονίζεται στην εικόνα 19. Η πρώτη εντολή που χρησιμοποιήθηκε για την υλοποίηση του signup είναι η auth=FirebaseAuth.getInstance η οποία αρχικοποιεί το FirebaseAuth. Στη συνέχεια ενεργοποιούμε με την εντολή .setOnClickListener το κουμπί btn_register. Επίσης μέσα στην εντολή .setOnClickListener υπάρχει η συνάρτηση onClick στην οποία υπάρχουν και όλοι οι έλεγχοι καθώς και η λειτουργία του signup. Η πρώτη εντολή της συνάρτησης είναι η κλήση μίας άλλης συνάρτησης της uploadtofirebase(). Στις επόμενες τρεις εντολές ο προγραμματιστής δημιουργεί τρεις μεταβλητές τύπου String τις txt_email,txt_password και την txt_username και δίνει σε καθεμία από τις μεταβλητές αυτό που έχει πληκτρολογήσει ο χρήστης για κάθε πεδίο. Αυτό πετυχαίνετε με τη εντολή .getText.toString.

Τέλος υπάρχουν κάποιοι έλεγχοι οι οποίοι γίνονται πριν να γίνει η εγγραφή. Οι έλεγχοι αυτοί είναι τρεις. Οι έλεγχοι που γίνονται είναι σε εμφωλευμένο έλεγχο και ο έλεγχος που το αρχίζει αυτό είναι ο κώδικας που υπάρχει από το if έως το else if. Το if ελέγχει αν ο χρήστης έχει δώσει τα στοιχεία username, email και password. Αν ο χρήστης έχει δώσει κανένα ένα ή δύο και όχι όλα τα πεδία τότε εμφανίζεται με την εντολή Toast το μήνυμα που υπάρχει μέσα στην εντολή αυτή δηλαδή το «all fields are required». Ο δεύτερος έλεγχος είναι από το else if έως το else. Αυτός ο δεύτερος έλεγχος έχει να κάνει με το μέγεθος του κωδικού που έχει δώσει ο χρήστης και αν είναι μικρότερος από έξι ψηφία εμφανίζεται το μήνυμα που εμφανίζεται στην εντολή Toast και το οποίο είναι το «Password must be at least six characters». Τρίτο και τελευταίο μέρος είναι το else που υπάρχει. Στο else αυτό γίνεται το signup αν ο χρήστης έχει δώσει όλα τα πεδία και έναν κωδικό που θα είναι μεγαλύτερος από έξι ή έξι ψηφία. Το signup γίνεται στην συνάρτηση την οποία καλεί ο προγραμματιστής μέσα στο else, ονομάζεται registerUser και παίρνει τρία ορίσματα το txt_username, το txt_email και το txt_password.

Τέλος για να κλείσει η κλάση του SignUp θα αναλυθούν και κάποιες συναρτήσεις που χρησιμοποιήθηκαν στην κλάση αυτή. Οι συναρτήσεις αυτές είναι οι registerUser, η uploadtofirebase και η onActivityResult. Οι συναρτήσεις αυτές απεικονίζονται στις παρακάτω εικόνες.

```
private void registerUser(String username, String email,String password ) {
    auth.createUserWithEmailAndPassword(email,password).addOnCompleteListener(SignUp.this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful())
            {
                Toast.makeText(SignUp.this,"Registration User successfully",Toast.LENGTH_SHORT).show();

            }else {
                Toast.makeText(SignUp.this,"Registration User unsuccessfull",Toast.LENGTH_SHORT).show();
            }

        }
    });
}
```

FIGURE 46 : ΣΥΝΑΡΤΗΣΗ 4

```
private void uploadtofirebase() {
    ProgressDialog dialog=new ProgressDialog(this);
    dialog.setTitle("File Uploader");
    dialog.show();
    FirebaseStorage storage=FirebaseStorage.getInstance();
    StorageReference uploader=storage.getReference("image1" +new Random().nextInt(50));
    uploader.putFile(filepath)
            .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
                @Override
                public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                    uploader.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
                        @Override
                        public void onSuccess(Uri uri) {
                            FirebaseUser firebaseUser=auth.getCurrentUser();
                            String userid=firebaseUser.getUid();
                            FirebaseDatabase firebaseDatabase= FirebaseDatabase.getInstance();


                            User obj=new User(uri.toString(),username.getText().toString(),email.getText().toString(),password.getText().toString());
                            DatabaseReference root=firebaseDatabase.getReference("users").child(userid);
                            root.setValue(obj);
                            Toast.makeText(getApplicationContext(),"Uploaded",Toast.LENGTH_LONG).show();
                        }
                    });
                }
            }).addOnProgressListener(new OnProgressListener<UploadTask.TaskSnapshot>() {
        @Override
        public void onProgress(@NonNull UploadTask.TaskSnapshot snapshot) {
            float percent=(100*snapshot.getBytesTransferred())/snapshot.getTotalByteCount();
            dialog.setMessage("Uploaded: " + (int)percent + " %");
        }
    });
```

FIGURE 47 : ΣΥΝΑΡΤΗΣΗ 5

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode==2 && resultCode==RESULT_OK )
    {
        filepath=data.getData();
        ImageView.setImageURI(filepath);
    }
}
```

FIGURE 48 : ΣΥΝΑΡΤΗΣΗ 6

Η πρώτη συνάρτηση δηλαδή η συνάρτηση registerUser υλοποιεί το SignUp. Σε αυτό το σημείο θα αναλυθεί ώστε να γίνει όσο πιο κατανοητή γίνεται. Στην αρχή υπάρχει η εντολή που φαίνεται στην πρώτη γραμμή και είναι η auth.createUserWithEmailAndPassword . Σε αυτή την εντολή δημιουργείται ουσιαστικά ο χρήστης για την βάση δεδομένων, η οποία αποθηκεύει σε κατάλληλο χώρο το email αλλά και το password του χρήστη. Στη συνέχεια και στις υπόλοιπες γραμμές βλέπουμε ότι υπάρχει μία συνάρτηση η οποία ονομάζεται onComplete . Αυτή η συνάρτηση δημιουργήθηκε από την πρώτη γραμμή και το .addOnClickListener(SignUp.this,new OnCompleteListener<AuthResult>()). Η χρήση της συνάρτησης αυτής είναι για να μπορεί ο χρήστης να καταλάβει αν έχει τελειώσει η εγγραφή επιτυχώς ή αν κάτι πήγε λάθος. Αυτό μπορεί να το διαπιστώσει με τον έλεγχο που υπάρχει μέσα στην συνάρτηση. Ο έλεγχος αυτός στο if ελέγχει αν όλα πήγαν καλά και αν ναι εμφανίζει το μήνυμα της εντολής Toast που υπάρχει μέσα στο if. Το μήνυμα αυτό λέει «Registration User Successfully». Σε περίπτωση που κάτι δεν πήγε καλά η εφαρμογή εμφανίζει στον χρήστη το μήνυμα που υπάρχει στην εντολή Toast του else. Το μήνυμα αυτό λέει «Registration User Unsuccessfully» . Αυτή είναι και η συνάρτηση registerUser.

Στη συνέχεια θα αναλυθεί η δεύτερη συνάρτηση η οποία ονομάζεται uploadtofirebase(). Η συνάρτηση αυτή έχει ως στόχο να αποθηκεύσει όλα τα στοιχεία του χρήστη στη βάση δεδομένων. Αρχικά στη συνάρτηση δημιουργούμε ένα ProgressDialog το οποίο ονομάζεται dialog και έχει ως τίτλο «File Uploader». Αφού δημιουργηθεί το ProgressDialog αρχικοποιείτε μία μεταβλητή τύπου FirebaseStorage η οποία ονομάζεται storage. Επίσης δημιουργείται μία μεταβλητή τύπου StorageReference η οποία ονομάζεται uploader. Η μεταβλητή αυτή δημιουργεί ένα reference για να ανεβάσει την εικόνα που έχει χρησιμοποιήσει ο χρήστης με το όνομα image1 και κάποιους τυχαίους αριθμούς πίσω από το image1. Έπειτα και με την βοήθεια της μεταβλητής uploader δημιουργούνται κάποιες συναρτήσεις οι οποίες ονομάζονται onSuccess και onProgress. Στην πρώτη συνάρτηση έχει υλοποιηθεί ο κώδικας για το όταν έχει επιτύχει η εγγραφή του χρήστη στη βάση δεδομένων. Σε αυτή την περίπτωση το σύστημα δημιουργεί μία μεταβλητή τύπου FirebaseDatabase η οποία ονομάζεται firebaseDatabase. Επίσης αποθηκεύει σε μία String

μεταβλητή η οποία ονομάζεται userid το id του χρήστη που δημιουργείται. Άλλο ένα που πρέπει να γίνει για να αποθηκευτούν όλα τα στοιχεία είναι να δημιουργηθεί στο πρόγραμμα ένας νέος χρήστης. Αυτό συμβαίνει με την εντολή User obj=new User() και μέσα στη παρένθεση υπάρχουν τα ορίσματα δηλαδή τα στοιχεία που έχουν ζητηθεί από τον χρήστη. Με την εντολή DatabaseReference δημιουργείται μία αναφορά στην ομάδα users με αριθμό το userid η οποία ονομάζεται root. Στη μεταβλητή αυτή στη συνέχεια μπαίνουν όλα τα στοιχεία που έδωσε ο χρήστης και αποθηκεύονται στην βάση δεδομένων όπως αναφέρθηκε πιο πριν. Τέλος με την εντολή Toast ο χρήστης βλέπει ένα μήνυμα στην οθόνη του που λέει «Uploaded».

Στη δεύτερη συνάρτηση η οποία είναι η onProgress υπάρχουν δύο εντολές. Αυτές είναι η εξής:

- float percent=
  (100 * snapshot. getBytesTransferred ())/snapshot. getTotalByCount; και
- dialog. setMessage (“Uploaded “+(int) percent +” %”);

Αυτές οι εντολές δείχνουν στον χρήστη πόσο επί τις εκατό έχει ολοκληρωθεί η εγγραφή του. Όταν τελειώσει και φτάσει 100% η εγγραφή έχει ολοκληρωθεί.

5.4.4 Login

Σε αυτό το σημείο τελείωσε η ανάλυση για την κλάση Sign Up. Επόμενη κλάση που δεν έχει αναλυθεί ακόμα αλλά έχει δημιουργηθεί προς το παρόν στο πρόγραμμα είναι αυτή του Login. Σε αυτή ο χρήστης δίνει τα στοιχεία του για να συνδεθεί και να προηγηθεί στην εφαρμογή. Αρχικά όπως πάντα οι βιβλιοθήκες που χρησιμοποιήθηκαν προηγούνται του κώδικα. Αυτές φαίνονται και στην παρακάτω εικόνα.

```
package com.example.telikiergasia;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.rengwuxian.materialedittext.MaterialEditText;
```

FIGURE 49 : ΒΙΒΛΙΟΘΗΚΕΣ LOGIN

Αυτές είναι οι βιβλιοθήκες που χρειάζεται το πρόγραμμα να έχει για να υλοποιηθεί ο κώδικας για το Login. Στη συνέχεια η εικόνα που ακολουθεί δείχνει τις μεταβλητές αυτής της κλάσης.

```
MaterialEditText login_email,login_password;
Button btn_login;
FirebaseAuth auth;
DatabaseReference reference;
FirebaseAuth mAuth;
```

FIGURE 50 : ΔΗΛΩΣΗ ΜΕΤΑΒΛΗΤΩΝ

Από την παραπάνω εικόνα μπορεί εύκολα κάποιος να διαπιστώσει ότι οι μεταβλητές που θα χρησιμοποιηθούν σε αυτή τη κλάση είναι η εξής:

- login_email
- login_password
- btn_login
- reference
- mAuth

Οι πρώτες τρεις μεταβλητές είναι αυτές που χρησιμοποιούνται και στο layout. Οι υπόλοιπες μεταβλητές είναι για να βοήσουν την εφαρμογή να υλοποιήσει το login. Στη συνέχεια και στην εικόνα που ακολουθεί φαίνονται οι αρχικοποιήσεις των μεταβλητών.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);
    login_email=findViewById(R.id.email_login);
    login_password=findViewById(R.id.password_login);
    btn_login=findViewById(R.id.btn_login);
    auth=FirebaseAuth.getInstance();
```

FIGURE 51 : ΑΡΧΙΚΟΠΟΙΗΣΗ ΜΕΤΑΒΛΗΤΩΝ

Στην παραπάνω εικόνα υπάρχει η αρχικοποίηση των μεταβλητών της εφαρμογής. Οι μεταβλητές login_email, login_password και το btn_login υπάρχουν και στο layout όπως αναφέρονται μέσα στην εντολή findViewById. Επίσης έχει αρχικοποιηθεί η FirebaseAuth. Στη συνέχεια έχουμε την ενεργοποιήσει του btn_login με την εντολή .setOnClickListener και τη δημιουργία της συνάρτησης onClick. Στην παρακάτω εικόνα φαίνονται και ο κώδικας που υπάρχει μέσα στην onClick και αυτό θα αναλυθεί.

```
btn_login.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String txt_email=login_email.getText().toString();
        String txt_password=login_password.getText().toString();
        loginUser(txt_email,txt_password);
    }
});
```

FIGURE 52 : BUTTON LOGIN

Σε αυτή την εικόνα υπάρχουν τρεις εντολές μέσα σε στην συνάρτηση onClick. Αυτές είναι οι εξής:

- String txt_email=login_email. getText (). toString ();
- String txt_password=login_password. getText(). toString ();
- loginUser(txt_email,txt_password) Οι πρώτες δύο εντολές δημιουργούν μεταβλητές τύπου String οι οποίες ονομάζονται txt_email και txt_password. Αυτές οι δύο μεταβλητές παίρνουν αυτά που έχει πληκτρολογήσει ο χρήστης στο αντίστοιχο πεδίο. Η Τρίτη και η τελευταία εντολή καλεί την συνάρτηση loginUser με ορίσματα τις δύο παραπάνω μεταβλητές. Η συνάρτηση loginUser θα αναλυθεί στην συνέχεια αλλά και απεικονίζεται στην παρακάτω εικόνα.

```
private void loginUser(String email, String password) {
    auth.signInWithEmailAndPassword(email, password).addOnSuccessListener(new OnSuccessListener<AuthResult>() {
        @Override
        public void onSuccess(AuthResult authResult) {
            Toast.makeText(Login.this, "Login Succesfull", Toast.LENGTH_SHORT).show();

            Intent intent=new Intent(Login.this,Arxiki_Selida.class);
            String item=String.valueOf(false);
            intent.putExtra("boolean",item);
            startActivity(intent);
            startActivity(intent);

        }
    });
}
```

FIGURE 53: ΣΥΝΑΡΤΗΣΗ LOGIN USER

Αυτή η συνάρτηση ουσιαστικά υλοποιεί το login του χρήστη. Στην πρώτη γραμμή υπάρχει μία εντολή που ξεκινάει με auth.signInWithEmailAndPassword . Με αυτήν την εντολή το σύστημα παίρνει τα στοιχεία που έχει πληκτρολογήσει ο χρήστης και ελέγχει αν υπάρχουν στην βάση. Στην συνάρτηση onSuccesss υπάρχουν οι εντολές που θα υλοποιηθούν αν είναι σωστά τα στοιχεία που έχει δώσει ο χρήστης. Αν ο χρήστης έχει δώσει τα σωστά στοιχεία τότε εμφανίζεται το μήνυμα που έχει γραφτεί στην εντολή Toast και το μήνυμα είναι «Login Successfull». Στη συνέχεια με την επόμενη εντολή δημιουργείται μία μεταβλητή τύπου Intent η οποία ονομάζεται intent και έχει σκοπό αφού όλα πήγαν καλά να πάει το χρήστη από την σελίδα που είναι στην επόμενη. Έπειτα δημιουργείται μία μεταβλητή τύπου String η οποία ονομάζεται itemκαι παίρνει την τιμή false όπως επίσης και η μεταβλητή αυτή περνάει και στην επόμενη σελίδα με την εντολή intent.putExtra. Τέλος υπάρχει και η εντολή startActivity η οποία πάει το χρήστη από τη σελίδα που είναι στην επόμενη που έχει δηλωθεί.

5.4.5 Arxiki Selida

Επόμενη κλάση που θα αναλυθεί είναι αυτή που ονομάζεται ArxikiSelida. Σε αυτή την κλάση ο χρήστης έχει μπει στην αρχική σελίδα της εφαρμογής μετά το login που έχει κάνει. Σε αυτή τη σελίδα ο χρήστης θα μπορέσει να διαλέξει για το αν θέλει πληροφορίες για τους ηθοποιούς ή για τις ταινίες. Αρχικά στην επόμενη εικόνα φαίνονται οι βιβλιοθήκες που έχουν χρησιμοποιηθεί για αυτή την κλάση.

```
package com.example.telikiergasia;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.bumptech.glide.Glide;
import com.google.android.gms.auth.api.signin.GoogleSignIn;
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import de.hdodenhof.circleimageview.CircleImageView;
```

FIGURE 54 : ΒΙΒΛΙΟΘΗΚΕΣ

Αυτές είναι και οι βιβλιοθήκες που θα χρησιμοποιηθούν για την κλάση αυτή με σκοπό να μπορεί να είναι εκτελέσιμος ο κώδικας που έχει γραφτεί. Στην παρακάτω εικόνα απεικονίζονται οι μεταβλητές οι οποίες χρειάζονται να υπάρχουν σε αυτή την κλάση.

```
CircleImageView profile_image;
TextView username;
FirebaseUser firebaseUser;
DatabaseReference databaseReference;
```

FIGURE 55: ΜΕΤΑΒΛΗΤΕΣ ΤΗΣ ΚΛΑΣΗΣ ARXIKI_SELIDA

Όπως κάποιος μπορεί να διαπιστώσει πολύ εύκολα οι μεταβλητές οι οποίες θα χρησιμοποιηθούν είναι τέσσερις. Αυτές είναι οι profile_image τύπου CircleImageView, η username τύπου TextView, η firebaseUser τύπου FirebaseUser και η databaseReference τύπου DatabaseReference. Οι πρώτες δύο μεταβλητές έχουν χρησιμοποιηθεί και στο layout της κλάσης αυτής. Οι επόμενες δύο είναι βοηθητικές για την βάση δεδομένων που έχει χρησιμοποιηθεί. Στην επόμενη φωτογραφία απεικονίζεται η δημιουργία κάποιον νέων μεταβλητών και η αρχικοποίηση κάποιων από αυτές που αναφέρθηκαν πιο πάνω.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_arxiki__selida);
    profile_image = findViewById(R.id.imageView);
    username = findViewById(R.id.username_arxiki);
    firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
    databaseReference = FirebaseDatabase.getInstance().getReference("users").child(firebaseUser.getUid());
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    Intent intent = getIntent();
    String item;
    item = intent.getExtras().getString("boolean");
```

FIGURE 56 : ΑΡΧΙΚΟΠΟΙΗΣΗ ΜΕΤΑΒΛΗΤΩΝ

Η μεταβλητή profile_image όπως και η username βρίσκονται στο layout με τα ονόματα imageview και username_arxiki. Η μεταβλητή firebaseUser παίρνει το id που έχει ο κάθε χρήστης στη βάση δεδομένων. Η μεταβλητή databaseRefernece παίρνει αναφορά για την βάση δεδομένων δηλαδή σε ποιο μονοπάτι υπάρχουν τα στοιχεία των χρηστών. Οι επόμενες δύο γραμμές είναι αυτές που ουσιαστικά ενεργοποιούν το toolbar που έχει δημιουργηθεί και χρησιμοποιείται σε αυτή την κλάση. Το toolbar έχει δημιουργηθεί σε ξεχωριστό σημείο και μπορεί να χρησιμοποιηθεί σε όλες τις κλάσεις. Οι τελευταίες τρεις σειρές δίνουν στην μεταβλητή item την τιμή που έχει στείλει το πρόγραμμα από τις προηγούμενες επιλογές του. Οι τιμές που μπορεί να έχει η μεταβλητή item είναι true ή false. Στην συνέχεια η εικόνα που ακολουθεί δείχνει το πώς το toolbar παίρνει το username και την εικόνα του χρήστη αναλόγως με ποιο λογαριασμό έχει κάνει login.

```
databaseReference.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {

        if (item.equals("false")) {
            User user = snapshot.getValue(User.class);
            username.setText(user.getUsername());
            Glide.with(Arxiki_Selida.this).load(user.getImageurl()).into(profile_image);
            Toast.makeText(Arxiki_Selida.this, item, Toast.LENGTH_SHORT).show();
        }
        if (item.equals("true")) {
            Intent intent = getIntent();
            String img = intent.getExtras().getString("image");
            String username_google = intent.getExtras().getString("username");
            username.setText(username_google);
            Glide.with(Arxiki_Selida.this).load(img).into(profile_image);
        }

    }


    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }
});
```

FIGURE 57 : USERNAME ΚΑΙ ΕΙΚΟΝΑ ΣΕ TOOLBAR

Αν στην μεταβλητή item υπάρχει η τιμή false τότε ο χρήστης έχει κάνει login με την εφαρμογή. Ο κώδικας που έχει χρησιμοποιηθεί για να πάρει το toolbar το username και την εικόνα του χρήστη υπάρχουν τέσσερις εντολές. Αυτές κάνουν τα εξής πράγματα.

Αρχικά δημιουργείται μία μεταβλητή η οποία ονομάζεται user και είναι τύπου User και παίρνει όλα τα στοιχεία του χρήστη. Στη συνέχεια η μεταβλητή username παίρνει το username του χρήστη και το εμφανίζει στην οθόνη. Τέλος με την εντολή Glide παίρνει η εφαρμογή την εικόνα που έχει ορίσει ο χρήστης στο signup και εμφανίζεται στην οθόνη. Αντίθετα αν η τιμή του item είναι true ο χρήστης έχει κάνει login με το λογαριασμό της Google . Για αυτή την περίπτωση έχουν χρησιμοποιηθεί πέντε γραμμές κώδικα για να μπορεί το toolbar να

πάρει τα στοιχεία που θέλει. Αρχικά δημιουργείται μία μεταβλητή τύπου Intent που ονομάζεται intent έτσι ώστε να μπορεί μετά το πρόγραμμα να πάρει τις τιμές και τα στοιχεία που θέλει για το toolbar. Στις επόμενες δύο γραμμές ο κώδικας παίρνει τις τιμές που του έχει «στείλει» το προηγούμενο activity. Στις τελευταίες δύο εντολές το username γράφεται στην οθόνη αλλά και η εικόνα του χρήστη εμφανίζεται και αυτή στην οθόνη. Επόμενο βήμα στον κώδικα και το activity αυτό είναι η δημιουργία και η αρχικοποίηση δύο buttons που ο χρήστης θα μπορεί να διαλέξει αν θέλει να πάει να δει όλους τους actors ή όλες τις movies. Αυτό δείχνει και η παρακάτω εικόνα.

```
Button actorsbtn=findViewById(R.id.actorsbtn);
Button moviesbtn=findViewById(R.id.moviesbtn);
actorsbtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
       Intent intent=new Intent(getApplicationContext(),Actors.class);
       if (item.equals("false")){
           String item=String.valueOf(false);
           intent.putExtra("boolean",item);
       }
       if (item.equals("true"))
       {
           updateUI(firebaseUser);
       }


        startActivity(intent);
    }
});
moviesbtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent=new Intent(getApplicationContext(),Movies.class);
        if (item.equals("false")){
            String item=String.valueOf(false);
            intent.putExtra("boolean",item);
        }
        if (item.equals("true"))
        {
            updateUIMovies(firebaseUser);
        }

        startActivity(intent);
    }
});
```

FIGURE 58 : ΔΗΜΙΟΥΡΓΙΑ, ΑΡΧΙΚΟΠΟΙΗΣΗ BUTTONS, ΕΚΤΕΛΕΣΗ ΕΝΤΟΛΩΝ ΓΙΑ ΚΑΘΕ BUTTON .

Σε αυτή την εικόνα έχουν δημιουργηθεί δύο νέα buttons : το actorsbtn αλλά και το moviesbtn . Αρχικά ενεργοποιούμε το actorsbtn με την εντολή .setOnClickListener και μέσα στη συνάρτηση onClick δημιουργείται ο κώδικας που θα στείλει τα δεδομένα που χρειάζεται το επόμενο activity αλλά και που θα σε πάει σε αυτό. Έτσι ξεκινάει η συνάρτηση με την εντολή Intent η οποία ουσιαστικά δείχνει σε ποιο activity θα πάει η εφαρμογή μετά από το πάτημα του κουμπιού. Στη συνέχεια υπάρχει έλεγχος για την τιμή της μεταβλητής item. Αν η μεταβλητή item είναι false τότε περνάει η τιμή της στο επόμενο activity αλλιώς καλείται η συνάρτηση updateUI με όρισμα την μεταβλητή firebaseUser. Στο κουμπί moviesbtn χρησιμοποιείται ο ίδιος κώδικας. Η μόνη διαφορά είναι ότι αν είναι true η μεταβλητή item καλείται μία άλλη συνάρτηση η οποία ονομάζεται updateUIMovie και ότι με το πάτημα αυτού του κουμπιού θα πάει η εφαρμογή σε άλλο activity από αυτό που πήγε πριν. Τέλος για να κλείσει αυτή η κλάση έχουν μείνει για

ανάλυση οι συναρτήσεις που έχουν χρησιμοποιηθεί. Αυτές απεικονίζονται στις παρακάτω εικόνες.

```
private void updateUI(FirebaseUser user) {

    GoogleSignInAccount account=GoogleSignIn.getLastSignedInAccount(getApplicationContext());
    if (account!=null)
    {
        boolean boolean_i=true;
        String personName=account.getDisplayName();
        //String personGivenName=account.getGivenName();
        // String personFamilyName=account.getFamilyName();
        String personEmail=account.getEmail();
        // String personId=account.getId();
        String imgurl = account.getPhotoUrl().toString();
        Intent intent=new Intent(getApplicationContext(),Actors.class);
        intent.putExtra("image",imgurl);
        String item=String.valueOf(boolean_i);
        intent.putExtra("boolean",item);
        intent.putExtra("username",personName);
        startActivity(intent);

        Toast.makeText(this, personName+ personEmail, Toast.LENGTH_SHORT).show();

    }
}
```

FIGURE 59 : ΣΥΝΑΡΤΗΣΗ 7

```
private void updateUIMovies(FirebaseUser user) {

    GoogleSignInAccount account=GoogleSignIn.getLastSignedInAccount(getApplicationContext());
    if (account!=null)
    {
        boolean boolean_i=true;
        String personName=account.getDisplayName();
        //String personGivenName=account.getGivenName();
        // String personFamilyName=account.getFamilyName();
        String personEmail=account.getEmail();
        // String personId=account.getId();
        String imgurl = account.getPhotoUrl().toString();
        Intent intent=new Intent(getApplicationContext(),Movies.class);
        intent.putExtra("image",imgurl);
        String item=String.valueOf(boolean_i);
        intent.putExtra("boolean",item);
        intent.putExtra("username",personName);
        startActivity(intent);

        Toast.makeText(this, personName+ personEmail, Toast.LENGTH_SHORT).show();

    }
}
```

FIGURE 60 : ΣΥΝΑΡΤΗΣΗ 8

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.example_menu, menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    switch (item.getItemId()){
        case R.id.logout:
            FirebaseAuth.getInstance().signOut();
            startActivity(new Intent(Arxiki_Selida.this,MainActivity.class));
            return true;
    }
    return false;
}
```

FIGURE 61 : ΣΥΝΑΡΤΗΣΗ 9

Αρχικά οι συναρτήσεις που απεικονίζονται είναι οι UpdateUI και η UpdateUIMovies. Σε αυτές τις δύο συναρτήσεις ουσιαστικά ο χρήστης έχει συνδεθεί με το λογαριασμό της Google που διαθέτει και πρέπει να βρεθεί τρόπος για να υπάρχουν στο επόμενο activity κάποιες πληροφορίες που χρειάζονται. Αυτός είναι και ο λόγος της δημιουργίας αυτών των δύο συναρτήσεων. Οι μεταβλητές οι οποίες πρέπει να έχει το επόμενο activity για να λειτουργεί η εφαρμογή σωστά είναι τρεις. Αυτές είναι οι εξής: • username

- εικόνα του χρήστη και
- τιμή της μεταβλητής item

Οι τιμές αυτών των τριών στοιχείων που πρέπει να υπάρχουν στο επόμενο activity περνάνε με την εντολή intent.putExtra με την ετικέτα ουσιαστικά που υπάρχει ανάμεσα στα εισαγωγικά της κάθε εντολής. Οι επόμενες δύο συναρτήσεις έχουν τα ονόματα onCreateOptionMenu με όρισμα μία μεταβλητή τύπου Menu και η onOptionsItemSelected με όρισμα μία μεταβλητή τύπου MenuItem. Η πρώτη συνάρτηση έχει τρεις εντολές και η χρήση της είναι ουσιαστικά να βάλει στο activity αυτό το style του menu που έχει δημιουργηθεί στον κατάλληλο φάκελο έτσι ώστε να βλέπει ο χρήστης της εφαρμογής την εικόνα του, το username του αλλά και να μπορεί να βλέπει και την επιλογή του logout. Η πρώτη εντολή της συνάρτησης αυτής δημιουργεί μία μεταβλητή τύπου MenuInflater. Στη συνέχεια η μεταβλητή αυτή παίρνει το αρχείο στο οποίο υπάρχουν τα στοιχεία που θέλει να φαίνονται στο πάνω μέρος της οθόνης του χρήστη. Τέλος επιστρέφει true .

Στην δεύτερη συνάρτηση ενεργοποιούνται οι ιδιότητες που θέλει ο προγραμματιστής να έχει καθένα από τα στοιχεία του menu που έχει δημιουργήσει στο menu. Αυτό γίνεται ουσιαστικά με τις εντολές που υπάρχουν στην συνάρτηση. Αρχικά ο κώδικας παίρνει το αριθμό id που πατήθηκε από το χρήστη. Στην συνέχεια με την εντολή case βρίσκει ποιο από τα menuitem είναι. Σε αυτή την εφαρμογή υπάρχει μόνο ένα το οποίο ονομάζεται logout. Αν ο χρήστης πατήσει το logout τότε με την πρώτη εντολή που υπάρχει στην case ο χρήστης κάνει logout. Με την δεύτερη εντολή η εφαρμογή πάει στην αρχική σελίδα που μπορεί να ξανακάνει login ή signup.

5.4.6 Model Actor

Πριν αναλυθούν οι σελίδες που έχουν δημιουργηθεί από αυτό το activity θα αναλυθούν κάποιες ακόμα τάξεις . Πρώτη τάξη που θα αναλυθεί είναι η ModelActor. Σε αυτή την τάξη φαίνονται τα στοιχεία που αποθηκεύονται στην βάση δεδομένων για τις ταινίες που υπάρχουν στην εφαρμογή. Στη συνέχεια η εικόνα δείχνει τις μεταβλητές και τις συναρτήσεις constructors που χρησιμοποιούνται εδώ.

```
package com.example.telikiergasia;

public class ModelActor {
    String aImageUrl,date_of_birth,name,description;

    public ModelActor() {
    }

    public ModelActor(String aImageUrl, String date_of_birth, String name, String description) {
        this.aImageUrl = aImageUrl;
        this.date_of_birth = date_of_birth;
        this.name = name;
        this.description = description;
    }
```

FIGURE 62 : ΜΕΤΑΒΛΗΤΕΣ ΚΑΙ CONSTRUCTORS

Όπως φαίνεται και στην εικόνα οι μεταβλητές οι οποίες υπάρχουν είναι όλες τύπου String. Οι μεταβλητές αυτές είναι η aImageUrl, date_of_birth,name,description. Με τις συναρτήσεις constructor η εφαρμογή στέλνει με την σειρά που φαίνεται στα ορίσματα τα δεδομένα στη βάση που χρησιμοποιεί. Η σειρά αυτή είναι :

- aImageUrl

- date_of_birth
- name και
- description

Τα στοιχεία που υπάρχουν και που δέχεται η βάση για τον κάθε ηθοποιό είναι στο πρώτο όρισμα μία φωτογραφία, στο δεύτερο όρισμα η ημερομηνία γενεθλίων του και το τελευταίο είναι το βιογραφικό του κάθε ηθοποιού. Στην φωτογραφία που ακολουθεί φαίνονται όλες οι getters and setters συναρτήσεις που υπάρχουν για τους ηθοποιούς της εφαρμογής.

```
public String getaImageUrl() {
    return aImageUrl;
}

public void setaImageUrl(String aImageUrl) {
    this.aImageUrl = aImageUrl;
}

public String getDate_of_birth() {
    return date_of_birth;
}

public void setDate_of_birth(String date_of_birth) {
    this.date_of_birth = date_of_birth;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}
```

FIGURE 63 : GETTERS AND SETTERS

Σε αυτή τη φωτογραφία φαίνονται οι συναρτήσεις που χρησιμοποιούνται για να τραβήξουν ή να στείλουν δεδομένα στη βάση . Εννοείται ότι αν χρειάζεται κάποιος ή το πρόγραμμα να στείλει μόνο ένα ή να πάρει μόνο ένα στοιχείο γίνεται. Οι συναρτήσεις getters χρησιμοποιούνται ώστε να γράψει κάποιος στην βάση ενώ οι συναρτήσεις setters για να πάρει από την βάση. Κάτι πολύ σημαντικό είναι ότι πρέπει για όλες τις μεταβλητές να υπάρχουν getters and setters.

5.4.7 Model Movies

Επόμενη τάξη που θα αναλυθεί είναι αυτή που ονομάζεται ModelMovies. Σε αυτή την τάξη υπάρχουν οι μεταβλητές που χρησιμοποιήθηκαν για να αποθηκευτούν όλες οι ταινίες της εφαρμογής. Επίσης υπάρχουν και οι συναρτήσεις constructors αλλά και οι getters and setters συναρτήσεις. Στην επόμενη εικόνα υπάρχουν οι μεταβλητές που χρειάζονται για να αποθηκευτεί μια ταινία στη βάση δεδομένων αλλά και οι συναρτήσεις constructors.

```
package com.example.telikiergasia;

public class ModelMovies {
     String description,mImageUrl,movieVideo,movieYear,name;
     float rating;

    public ModelMovies() {
    }

    public ModelMovies(String description, String mImageUrl, String movieVideo, String movieYear, String name, float rating) {
        this.description = description;
        this.mImageUrl = mImageUrl;
        this.movieVideo = movieVideo;
        this.movieYear = movieYear;
        this.name = name;
        this.rating = rating;
    }
```

FIGURE 64 : ΜΕΤΑΒΛΗΤΕΣ

Οι μεταβλητές σε σύγκριση με άλλες φορές είναι δύο τύπων. Οι περισσότερες είναι τύπου String ενώ υπάρχει και μία τύπου float. Οι μεταβλητές οι οποίες χρησιμοποιούνται είναι οι εξής:

- description
- mImageUrl
- movieVideo
- movieYear
- name και
- rating

Οι μεταβλητές οι οποίες υπάρχουν είναι η περιγραφή της ταινίας, μία εικόνα της ταινίας, το video της ταινίας, τη χρονιά της ταινίας, το όνομα και τη βαθμολογία της. Επίσης υπάρχουν οι constructors συναρτήσεις οι οποίες βοηθάνε τις ταινίες να αποθηκευτούν στη βάση δεδομένων με την σειρά που φαίνεται. Στις επόμενες φωτογραφίες φαίνονται οι συναρτήσεις getters and setters.

```
public String getmImageUrl() {
    return mImageUrl;
}

public void setmImageUrl(String mImageUrl) {
    this.mImageUrl = mImageUrl;
}

public String getMovieVideo() {
    return movieVideo;
}

public void setMovieVideo(String movieVideo) {
    this.movieVideo = movieVideo;
}

public String getMovieYear() {
    return movieYear;
}

public void setMovieYear(String movieYear) {
    this.movieYear = movieYear;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public float getRating() {
    return rating;
}
```

FIGURE 65 : GETTERS AND SETTERS (EIKONA 1)

```
public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}
```

FIGURE 66 : GETTERS AND SETTERS (EIKONA 2)

Οι συναρτήσεις getters χρησιμοποιούνται για να δώσουν στοιχεία και πληροφορίες στη βάση δεδομένων. Οι συναρτήσεις setters βοηθάνε έτσι ώστε να δώσει η βάση δεδομένων στοιχεία στην εφαρμογή.

5.4.8 Adapter Actors

Επόμενη κλάση η οποία θα αναλυθεί είναι η AdapterActors. Η κλάση αυτή βοηθάει την εφαρμογή να υλοποιήσει κάποιες από τις λειτουργίες της. Αυτές θα αναλυθούν στις επόμενες σελίδες. Η εικόνα που ακολουθεί δείχνει τις βιβλιοθήκες που χρησιμοποιήθηκαν σε αυτή την κλάση.

```
package com.example.telikiergasia;

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;
import com.firebase.ui.database.FirebaseRecyclerAdapter;
import com.firebase.ui.database.FirebaseRecyclerOptions;

import de.hdodenhof.circleimageview.CircleImageView;
```

FIGURE 67 : ΒΙΒΛΙΟΘΗΚΕΣ

Η παραπάνω εικόνα δείχνει τις βιβλιοθήκες που έχουν χρησιμοποιηθεί σε αυτή την κλάση. Στις επόμενες εικόνες φαίνονται οι συναρτήσεις που έχουν χρησιμοποιηθεί.

```
public class AdapterActor extends FirebaseRecyclerAdapter<ModelActor, AdapterActor.myviewholder> {
Context context;

    public AdapterActor(@NonNull FirebaseRecyclerOptions<ModelActor> options,Context context) {

        super(options);
        this.context=context;
    }


    @Override
    protected void onBindViewHolder(@NonNull myviewholder holder, int position, @NonNull ModelActor ModelActor) {

        holder.username.setText(ModelActor.getName());
        holder.date_of_birth.setText(ModelActor.getDate_of_birth());
        Glide.with(holder.img.getContext()).load(ModelActor.getaImageUrl()).into(holder.img);
        holder.img.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent=new Intent(context,FullScreen.class);
                intent.putExtra("image", ModelActor.getaImageUrl());
                intent.putExtra("description",ModelActor.getDescription());
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                context.startActivity(intent);
            }
        });
    }

    @NonNull
    @Override
    public myviewholder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view= LayoutInflater.from(parent.getContext()).inflate(R.layout.actor_item_list,parent,false);
        return new myviewholder(view);
    }
```

FIGURE 68 : ΣΥΝΑΡΤΗΣΗ 10

```
class myviewholder extends RecyclerView.ViewHolder implements View.OnClickListener
{
    CircleImageView img;
    TextView date_of_birth,username;
    public myviewholder(@NonNull View itemView) {
        super(itemView);
        img=(CircleImageView)itemView.findViewById(R.id.img1);
        username=(TextView)itemView.findViewById(R.id.nametext);
        date_of_birth=(TextView)itemView.findViewById(R.id.birthdaytext);
        itemView.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        int position=getAdapterPosition();


    }
}
```

FIGURE 69 : ΣΥΝΑΡΤΗΣΗ 11

Οι συναρτήσεις οι οποίες έχουν χρησιμοποιηθεί είναι οι εξής:

- μία συνάρτηση constructor
- onBindViewHolder
- onCreateViewHolder

η πρώτη συνάρτηση χρησιμοποιείται για να δηλώσει τα ορίσματα που θα χρειαστεί να έχει κάποια άλλη κλάση έτσι ώστε να την καλέσει. Η δεύτερη συνάρτηση χρησιμοποιείται για να ορίσει τι γίνεται σε περίπτωση που ο χρήστης πατήσει την εικόνα ενός ηθοποιού. Αυτό που θα συμβεί υλοποιείται μέσα στη συνάρτηση onClick. Η συνάρτηση onClick έχει πέντε εντολές. Η πρώτη εντολή αυτό που κάνει είναι να δημιουργεί μία μεταβλητή τύπου Intent η οποία ονομάζεται intent και δείχνει σε ποια σελίδα θα πάει η εφαρμογή αν πατηθεί η εικόνα του ηθοποιού. Στη συνέχεια τραβάει από τη βάση δεδομένων την εικόνα του ηθοποιού αλλά και το βιογραφικό του και τα περνάει στην επόμενη σελίδα με τις ετικέτες image και description αντίστοιχα. Τέλος ενεργοποιεί το να μπορεί η εφαρμογή να πάει στην επόμενη σελίδα.

Η Τρίτη και τελευταία συνάρτηση έχει δύο εντολές. Αρχικά δημιουργεί μία μεταβλητή τύπου View η οποία ονομάζεται view. Με αυτή τη μεταβλητή ορίζεται σε ποιο layout «βλέπει» η κλάση αυτή. Η δεύτερη εντολή γυρίζει τη μεταβλητή που δημιουργήθηκε πιο πάνω στην κλάση την οποία θα αναλυθεί στη συνέχεια. Η κλάση αυτή ονομάζεται myviewholder και η λειτουργία της είναι να αρχικοποιήσει τα στοιχεία που ήδη υπάρχουν στο layout. Τα στοιχεία που υπάρχουν εκεί για τους ηθοποιούς είναι τρία . Το πρώτο στοιχείο είναι η φωτογραφία του ηθοποιού, δεύτερο είναι το όνομα του ηθοποιού και τελευταίο είναι η ημερομηνία γέννησης του. Τέλος ενεργοποιεί και το να μπορεί να πατηθεί η εικόνα του ηθοποιού με την τελευταία της εντολή.

5.4.9 Adapter Movies

Η επόμενη κλάση που θα αναλυθεί είναι η AdapterMovies. Σε αυτή τη κλάση υπάρχουν κάποιες από τις ιδιότητες που χρειάζονται για να λειτουργήσει όπως πρέπει η εφαρμογή για τις ταινίες. Αρχικά στην επόμενη εικόνα φαίνονται οι βιβλιοθήκες που χρησιμοποιήθηκαν για αυτή την κλάση.

```
package com.example.telikiergasia;

import android.content.Context;
import android.content.Intent;
import android.net.Uri;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.RatingBar;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;
import com.firebase.ui.database.FirebaseRecyclerAdapter;
import com.firebase.ui.database.FirebaseRecyclerOptions;
import com.google.android.exoplayer2.ExoPlayerFactory;
import com.google.android.exoplayer2.SimpleExoPlayer;
import com.google.android.exoplayer2.extractor.DefaultExtractorsFactory;
import com.google.android.exoplayer2.extractor.ExtractorsFactory;
import com.google.android.exoplayer2.source.ExtractorMediaSource;
import com.google.android.exoplayer2.source.MediaSource;
import com.google.android.exoplayer2.trackselection.AdaptiveTrackSelection;
import com.google.android.exoplayer2.trackselection.DefaultTrackSelector;
import com.google.android.exoplayer2.trackselection.TrackSelector;
import com.google.android.exoplayer2.ui.SimpleExoPlayerView;
import com.google.android.exoplayer2.upstream.BandwidthMeter;
import com.google.android.exoplayer2.upstream.DefaultBandwidthMeter;
import com.google.android.exoplayer2.upstream.DefaultHttpDataSourceFactory;

import de.hdodenhof.circleimageview.CircleImageView;
```

FIGURE 70: ΒΙΒΛΙΟΘΗΚΕΣ

Στην παραπάνω εικόνα φαίνονται οι βιβλιοθήκες που έχουν χρησιμοποιηθεί για αυτή την κλάση. Μία βιβλιοθήκη την οποία δεν υπάρχει προς το παρόν κάπου αλλού είναι αυτή που έχει την φράση .exoplayer2.όλες αυτές οι βιβλιοθήκες βοηθάνε στο να τρέξει το video της ταινίας που θέλει να δει ο χρήστης. Στις επόμενες εικόνες φαίνονται οι συναρτήσεις που έχουν χρησιμοποιηθεί για αυτή την κλάση.

```
public class AdapterMovies extends FirebaseRecyclerAdapter<ModelMovies, AdapterMovies.myviewholder> {
    SimpleExoPlayerView simpleExoPlayerView;
    SimpleExoPlayer simpleExoPlayer;
    TextView vtitleview;
    Context context;
    public AdapterMovies(@NonNull FirebaseRecyclerOptions<ModelMovies> options,Context context) {
        super(options);
        this.context=context;

    }

    public AdapterMovies(FirebaseRecyclerOptions<ModelMovies> options) {
        super(options);
    }
```

FIGURE 71 : ΣΥΝΑΡΤΗΣΗ 12

```
@Override
protected void onBindViewHolder(@NonNull myviewholder holder, int position, @NonNull ModelMovies ModelMovies) {
    holder.name.setText(ModelMovies.getName());
    holder.movieYear.setText(ModelMovies.getMovieYear());
    Glide.with(holder.circleImageView.getContext()).load(ModelMovies.getmImageUrl()).into(holder.circleImageView);
    holder.ratingBar.setRating(ModelMovies.getRating()/2);
    holder.circleImageView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent=new Intent(context,FullScreenMovies.class);
            intent.putExtra("image", ModelMovies.getMovieVideo());
            intent.putExtra("description",ModelMovies.getDescription());
            prepareexoplayer(context,ModelMovies.getMovieVideo(),ModelMovies.getMovieVideo());
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            context.startActivity(intent);
        }
    });
}
```

FIGURE 72 : ΣΥΝΑΡΤΗΣΗ 13

```
void prepareexoplayer(Context context, String name, String movieVideo)
{
    try
    {
        vtitleview.setText(name);
        BandwidthMeter bandwidthMeter = new DefaultBandwidthMeter();
        TrackSelector trackSelector = new DefaultTrackSelector(new AdaptiveTrackSelection.Factory(bandwidthMeter));
        simpleExoPlayer =(SimpleExoPlayer) ExoPlayerFactory.newSimpleInstance(context,trackSelector);

        Uri videoURI = Uri.parse(movieVideo);

        DefaultHttpDataSourceFactory dataSourceFactory = new DefaultHttpDataSourceFactory("exoplayer_video");
        ExtractorsFactory extractorsFactory = new DefaultExtractorsFactory();
        MediaSource mediaSource = new ExtractorMediaSource(videoURI, dataSourceFactory, extractorsFactory, null, null);

        simpleExoPlayerView.setPlayer(simpleExoPlayer);
        simpleExoPlayer.prepare(mediaSource);
        simpleExoPlayer.setPlayWhenReady(false);


    }catch (Exception ex)
    {
        Log.d("Explayer Creshed", ex.getMessage().toString());
    }
}
```

FIGURE 73 : ΣΥΝΑΡΤΗΣΗ 14

```
@NonNull
@Override
public myviewholder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view= LayoutInflater.from(parent.getContext()).inflate(R.layout.movie_item_list,parent,false);
    return new myviewholder(view);
}

class myviewholder extends RecyclerView.ViewHolder
{
    CircleImageView circleImageView;
    TextView name,movieYear;
    RatingBar ratingBar;
    public myviewholder(@NonNull View itemView) {
        super(itemView);
        circleImageView=(CircleImageView)itemView.findViewById(R.id.imagemovie);
        name=(TextView)itemView.findViewById(R.id.namemovie);
        movieYear=(TextView)itemView.findViewById(R.id.movieyear);
        ratingBar=itemView.findViewById(R.id.ratingBar);


    }
}
```

FIGURE 74 : ΣΥΝΑΡΤΗΣΗ 15

Οι συναρτήσεις οι οποίες έχουν χρησιμοποιηθεί είναι οι εξής:

- συναρτήσεις constructor
- onBindViewHolder
- onCreateViewHolder

- prepareexoplayer Οι συναρτήσεις constructors χρησιμοποιούνται για να δηλώσουν τα ορίσματα που θα χρειαστεί να έχει κάποια άλλη κλάση έτσι ώστε να την καλέσει. Η δεύτερη συνάρτηση χρησιμοποιείται για να δηλώσει τι γίνεται σε περίπτωση που ο χρήστης πατήσει την εικόνα της ταινίας. Αυτό που θα συμβεί υλοποιείται μέσα στη συνάρτηση onClick. Η συνάρτηση onClick έχει έξι εντολές. Η πρώτη εντολή αυτό που κάνει είναι να δημιουργεί μία μεταβλητή τύπου Intent η οποία ονομάζεται intent και δείχνει σε ποια σελίδα θα πάει η εφαρμογή αν πατηθεί η εικόνα της ταινίας. Στη συνέχεια τραβάει από τη βάση δεδομένων την εικόνα της ταινίας καθώς και την περιγραφή της και τα περνάει στην επόμενη σελίδα με τις ετικέτες image και description. Επόμενη εντολή είναι να καλέσει την συνάρτηση prepareexoplayer η οποία θα αναλυθεί στη συνέχεια. Τέλος ενεργοποιεί το να μπορεί η εφαρμογή να πάει στην επόμενη σελίδα. Η Τρίτη συνάρτηση έχει δύο εντολές. Αρχικά δημιουργεί μια μεταβλητή τύπου View η οποία ονομάζεται view. Με αυτή τη μεταβλητή ορίζεται σε ποιο layout «βλέπει» η κλάση αυτή. Η δεύτερη εντολή γυρίζει τη μεταβλητή που δημιουργήθηκε πιο πάνω, στη κλάση την οποία θα αναλυθεί αμέσως τώρα. Η κλάση αυτή ονομάζεται myviewholder και η λειτουργία της είναι να αρχικοποιήσει τα στοιχεία που υπάρχουν στο layout. Τα στοιχεία που υπάρχουν εκεί για τις ταινίες είναι τέσσερα. Το πρώτο στοιχείο είναι η φωτογραφία της ταινίας, το δεύτερο είναι το όνομα της, το τρίτο είναι η χρόνιά της και το τελευταίο είναι η βαθμολογία της. Τέλος ενεργοποιεί και το να μπορεί να πατηθεί η εικόνα της ταινίας με την τελευταία εντολή.

Η τελευταία συνάρτηση είναι η prepareexoplayer. Αυτή η συνάρτηση ουσιαστικά προετοιμάζει το να μπορεί να τρέξει το video που υπάρχει στην κάθε ταινία. Έχει επίσης τρία ορίσματα που πρέπει να δέχεται κάθε φορά που καλείται. Η πρώτη είναι η σελίδα στην οποία βρίσκεται, η δεύτερη είναι το όνομα της ταινίας και η τελευταία είναι το video της ταινίας.

5.4.10 myviewholder

Επόμενη κλάση που θα αναλυθεί είναι η myviewholder. Αυτή η κλάση βοηθάει να τρέξει σωστά η αναπαραγωγή του κάθε video. Πρώτο πράγμα που θα αναλυθεί είναι οι βιβλιοθήκες που έχουν χρησιμοποιηθεί. Αυτές φαίνονται παρακάτω.

```
package com.example.telikiergasia;

import android.app.Application;
import android.net.Uri;
import android.util.Log;
import android.view.View;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.google.android.exoplayer2.ExoPlayerFactory;
import com.google.android.exoplayer2.SimpleExoPlayer;
import com.google.android.exoplayer2.extractor.DefaultExtractorsFactory;
import com.google.android.exoplayer2.extractor.ExtractorsFactory;
import com.google.android.exoplayer2.source.ExtractorMediaSource;
import com.google.android.exoplayer2.source.MediaSource;
import com.google.android.exoplayer2.trackselection.AdaptiveTrackSelection;
import com.google.android.exoplayer2.trackselection.DefaultTrackSelector;
import com.google.android.exoplayer2.trackselection.TrackSelector;
import com.google.android.exoplayer2.ui.SimpleExoPlayerView;
import com.google.android.exoplayer2.upstream.BandwidthMeter;
import com.google.android.exoplayer2.upstream.DefaultBandwidthMeter;
import com.google.android.exoplayer2.upstream.DefaultHttpDataSourceFactory;
```

FIGURE 75 : ΒΙΒΛΙΟΘΗΚΕΣ

Οι βιβλιοθήκες που χρησιμοποιήθηκαν φαίνονται στην παρακάτω εικόνα. Όλες αυτές οι βιβλιοθήκες βοηθάνε να τρέξει όπως πρέπει η λειτουργία της κλάσης αυτής. Στη συνέχεια υπάρχουν οι μεταβλητές οι οποίες χρησιμοποιούνται . Αυτές απεικονίζονται στην εικόνα που ακολουθεί.

```
SimpleExoPlayerView simpleExoPlayerView;
SimpleExoPlayer simpleExoPlayer;
TextView vtitleview;
```

FIGURE 76 : ΜΕΤΑΒΛΗΤΕΣ

Οι μεταβλητές αυτής της κλάσης είναι τρεις στον αριθμό. Η πρώτη μεταβλητή είναι τύπου SimpleExoPlayerView η οποία ονομάζεται simpleExoPlayerView. Η επόμενη μεταβλητή είναι τύπου SimpleExoPlayer και ονομάζεται simpleExoPlayer. Η Τρίτη και η τελευταία μεταβλητή είναι τύπου TextView και ονομάζεται textview. Για να τελειώσει η ανάλυση αυτής της κλάσης θα πρέπει να αναλυθούν και οι συναρτήσεις οι οποίες χρησιμοποιήθηκαν. Αυτές φαίνονται παρακάτω.

```
public myviewholder(@NonNull View itemView)
{
    super(itemView);
    itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            mClickListener.onItemClick(v,getAdapterPosition());
        }
    });
    itemView.setOnLongClickListener(new View.OnLongClickListener() {
        @Override
        public boolean onLongClick(View v) {
            mClickListener.onItemLongClick(v,getAdapterPosition());
            return false;
        }
    });

}
```

FIGURE 77 : ΣΥΝΑΡΤΗΣΗ 16

```
void prepareexoplayer(Application application, String videotitle, String videourl)
{
    try
    {
        vtitleview.setText(videotitle);
        BandwidthMeter bandwidthMeter = new DefaultBandwidthMeter();
        TrackSelector trackSelector = new DefaultTrackSelector(new AdaptiveTrackSelection.Factory(bandwidthMeter));
        simpleExoPlayer =(SimpleExoPlayer) ExoPlayerFactory.newSimpleInstance(application,trackSelector);

        Uri videoURI = Uri.parse(videourl);

        DefaultHttpDataSourceFactory dataSourceFactory = new DefaultHttpDataSourceFactory("exoplayer_video");
        ExtractorsFactory extractorsFactory = new DefaultExtractorsFactory();
        MediaSource mediaSource = new ExtractorMediaSource(videoURI, dataSourceFactory, extractorsFactory, null, null);

        simpleExoPlayerView.setPlayer(simpleExoPlayer);
        simpleExoPlayer.prepare(mediaSource);
        simpleExoPlayer.setPlayWhenReady(false);


    }catch (Exception ex)
    {
        Log.d("Explayer Creshed", ex.getMessage().toString());
    }
}
private myviewholder.ClickListener mClickListener;
public  interface  ClickListener{
    void onItemClick(View view,int position);
    void onItemLongClick(View view,int position);
}
public void setOnClickListener(myviewholder.ClickListener clickListener)
{
   mClickListener=clickListener;
}
```

FIGURE 78 : ΣΥΝΑΡΤΗΣΗ 17

Οι συναρτήσεις που έχουν χρησιμοποιηθεί σε αυτή την κλάση είναι τέσσερις. Αυτές είναι οι εξής:

- myviewholder
- prepareexoplayer
- ClickListener
- setOnClickListener Η πρώτη συνάρτηση είναι η constructor της κλάσης αυτής. Σε αυτή τη συνάρτηση ενεργοποιείται η δυνατότητα κάποιος να μπορεί να πάει να πατήσει την εικόνα του ηθοποιού ή της ταινίας που επιθυμεί να δει παραπάνω στοιχεία. Η επόμενη συνάρτηση είναι η prepareexoplayer. Αυτή η συνάρτηση βοηθάει να ξεκινήσει η αναπαραγωγή του video της κάθε ταινίας σωστά. Η Τρίτη συνάρτηση η οποία είναι τύπου interface αυτό που υλοποιεί είναι να δηλώνει τα ορίσματα τα οποία θα πάρουν οι δύο συναρτήσεις onItemClick και onItemLongClick. Αυτά είναι η σελίδα την οποία βρίσκεται η εφαρμογή και η θέση του αντικειμένου που πατήθηκε.

Η τέταρτη και τελευταία συνάρτηση αυτό που κάνει είναι να ορίζει μία μεταβλητή η οποία ονομάζεται η clickListener.

5.4.11 Actors

Επόμενη σελίδα που θα αναλυθεί είναι αυτή με το όνομα Actors. Ο χρήστης σε αυτό τη σελίδα μπορεί να δει όλους τους ηθοποιούς της εφαρμογής να κάνει αναζήτηση κάποιου με βάση το όνομα του ηθοποιού και τέλος να πατήσει στην εικόνα κάποιου ηθοποιού για να δει επιπλέον στοιχεία για αυτόν. Στην επόμενη εικόνα και για να ξεκινήσει η ανάλυση της σελίδας θα παρουσιαστούν οι βιβλιοθήκες οι οποίες χρησιμοποιήθηκαν.

```
package com.example.telikiergasia;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.widget.SearchView;
import android.widget.TextView;
import android.widget.Toast;

import com.bumptech.glide.Glide;
import com.firebase.ui.database.FirebaseRecyclerOptions;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import de.hdodenhof.circleimageview.CircleImageView;
```

FIGURE 79 : ΒΙΒΛΙΟΘΗΚΕΣ

Αυτές είναι οι βιβλιοθήκες που χρησιμοποιούνται σε αυτή την σελίδα έτσι ώστε να τρέξει σωστά ο κώδικας. Στην συνέχεια θα πρέπει να δηλωθούν οι μεταβλητές που θα χρησιμοποιηθούν. Αυτές φαίνονται παρακάτω.

```
RecyclerView recyclerView;
AdapterActor adapter;
CircleImageView profile_image;
TextView username;
FirebaseUser firebaseUser;
DatabaseReference databaseReference;
```

FIGURE 80 : ΔΗΛΩΣΗ ΜΕΤΑΒΛΗΤΩΝ

Όπως φαίνεται και παραπάνω οι μεταβλητές που χρησιμοποιούνται στην σελίδα αυτή είναι έξι στον αριθμό. Η πρώτη ονομάζεται recyclerview και είναι τύπου RecyclerView η οποία χρησιμοποιείται και στο layout. Η επόμενη μεταβλητή ονομάζεται adapter και είναι τύπου AdapterActor μίας κλάσης που έχει δημιουργηθεί και η οποία θα αναλυθεί στη συνέχεια με σκοπό να βοηθήσει αυτή τη σελίδα να λειτουργήσει σωστά. Επόμενη μεταβλητή είναι η profile_image η οποία είναι τύπου CircleImageView. Η επόμενη μεταβλητή ονομάζεται username και είναι τύπου TextView. Προτελευταία μεταβλητή είναι η firebaseUser η οποία είναι τύπου FirebaseUser. Η τελευταία μεταβλητή ονομάζεται databaserefernce και είναι τύπου DatabaseReference. Στη συνέχεια απεικονίζεται η αρχικοποίηση των μεταβλητών.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_actors);
    recyclerView=findViewById(R.id.recyclerview);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));
    FirebaseRecyclerOptions<ModelActor> options =
            new FirebaseRecyclerOptions.Builder<ModelActor>()
                    .setQuery(FirebaseDatabase.getInstance().getReference().child("Actors"), ModelActor.class
                    .build();
    profile_image=findViewById(R.id.imageView);
    username=findViewById(R.id.username_arxiki);
    firebaseUser= FirebaseAuth.getInstance().getCurrentUser();
    databaseReference= FirebaseDatabase.getInstance().getReference("users").child(firebaseUser.getUid());
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    Intent intent=getIntent();
    String item;
    item=intent.getExtras().getString("boolean");
```

FIGURE 81 : ΑΡΧΙΚΟΠΟΙΗΣΗ ΜΕΤΑΒΛΗΤΩΝ

Σε αυτή την φωτογραφία υπάρχουν η αρχικοποίηση της μεταβλητής recyclerview και όπως καταγράφεται στο layout. Επίσης το ίδιο ισχύει και για τις μεταβλητές profile_image όπως και το username. Η εντολή που ξεκινάει με την εντολή FirebaseRecyclerOptions ουσιαστικά παίρνει αναφορά από τη βάση δεδομένων στην ομάδα της βάσης που έχουν ως αρχικό παιδί τη λέξη Actors. Στη συνέχεια η μεταβλητή databasereference παίρνει αναφορά στη βάση δεδομένων στην ομάδα που έχει ως παιδί το users.Επίσης αρχικοποιείται και η toolbar. Τέλος δημιουργείται μία μεταβλητή item και παίρνει την τιμή που έχει σταλθεί με ετικέτα Boolean. Στη συνέχεια η εικόνα που ακολουθεί δείχνει το πώς φορτώνετε η εικόνα και το username στο κατάλληλο εικονίδιο και textview.

```
databaseReference= FirebaseDatabase.getInstance().getReference("users").child(firebaseUser.getUid());
databaseReference.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        if (item.equals("false"))
        {
            User user=snapshot.getValue(User.class);
            username.setText(user.getUsername());
            Glide.with(Actors.this).load(user.getImageurl()).into(profile_image);
            Toast.makeText(Actors.this, item, Toast.LENGTH_SHORT).show();
        }
        if (item.equals("true"))
        {
            Intent intent=  getIntent();
            String img= intent.getExtras().getString("image");
            String username_google=intent.getExtras().getString("username");
            username.setText(username_google);
            Glide.with(Actors.this).load(img).into(profile_image);
        }
    }
```

FIGURE 82 : ΕΠΙΛΟΓΗ ΕΙΚΟΝΑΣ ΚΑΙ USERNAME

Σε αυτή την εικόνα φαίνεται ότι μετά την αρχικοποίηση της μεταβλητής databasereference μέσα στη συνάρτηση onDataChange ουσιαστικά χρησιμοποιείται η μεταβλητή item για να επιλέξει η εφαρμογή ποια εικόνα πρέπει να εμφανίσει. Αν η τιμή της μεταβλητής item είναι true τότε η εφαρμογή παίρνει τις τιμές με τις ετικέτες image και username. Στη συνέχεια γράφει το username στο textview και βάζει και την εικόνα με την εντολή Glide στο κατάλληλο σημείο. Αν η τιμή false είναι αυτή που έχει το item τότε δημιουργείται μία μεταβλητή τύπου User η οποία ονομάζεται user και τραβάει όλα τα στοιχεία του χρήστη από την βάση δεδομένων. Στη συνέχεια γράφει στο textview το username και βάζει στην μεταβλητή που πρέπει την εικόνα. Τέλος για την κλάση αυτή το τελευταίο πράγμα που θα πρέπει να αναλυθεί είναι οι συναρτήσεις που χρησιμοποιήθηκαν. Στις επόμενες εικόνες φαίνονται όλες οι συναρτήσεις.

```
        adapter=new AdapterActor(options,getApplicationContext());
        recyclerView.setAdapter(adapter);


    }
    private void processsearch(String s)
    {

        FirebaseRecyclerOptions<ModelActor> options =
                new FirebaseRecyclerOptions.Builder<ModelActor>()
                        .setQuery(FirebaseDatabase.getInstance().getReference().child("Actors").orderByChild("name").startAt(s).endAt(s+"\uf8ff"), ModelActor.class)
                        .build();
        adapter = new AdapterActor(options,getApplicationContext());
        adapter.startListening();
        recyclerView.setAdapter(adapter);
    }
```

FIGURE 83 : ΣΥΝΑΡΤΗΣΗ 18

```
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.searchmenu, menu);
    MenuItem item = menu.findItem(R.id.search);
    SearchView searchView = (SearchView) item.getActionView();
    searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String s) {
            processsearch(s);


            return false;
        }

        @Override
        public boolean onQueryTextChange(String s) {
            processsearch(s);

            return false;
        }
    });
    return super.onCreateOptionsMenu(menu);

}


@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    switch (item.getItemId()){
        case R.id.logout:
            FirebaseAuth.getInstance().signOut();
            startActivity(new Intent(Actors.this,MainActivity.class));
            return true;
    }
    return false;
}
```

FIGURE 84 : ΣΥΝΑΡΤΗΣΗ 19

```
    @Override
    protected void onStart() {
        super.onStart();
        adapter.startListening();




    }



    @Override
    protected void onStop() {
        super.onStop();
        adapter.stopListening();
    }
}
```

FIGURE 85 : ΣΥΝΑΡΤΗΣΗ 20

Σε αυτές τις φωτογραφίες πρώτη συνάρτηση που υπάρχει είναι αυτή που ονομάζεται processsearch. Σε αυτή τη συνάρτηση υλοποιείται η αναζήτηση. Η αναζήτηση γίνεται όταν ο χρήστης πατήσει το κουμπί με το οποίο συνήθως γίνεται η αναζήτηση. Σε αυτή τη συνάρτηση η αναζήτηση ξεκινάει με την εντολή FirebaseRecyclerOprions η οποία ουσιαστικά κάνει την αναζήτηση. Στη συνέχεια με τις επόμενες εντολές η σελίδα παίρνει τα αποτελέσματα της αναζήτησης και δείχνει μόνο αυτά στην οθόνη. Η επόμενη συνάρτηση ονομάζεται onCreateOptionsMenu. Η συνάρτηση αυτή ουσιαστικά παίρνει το κατάλληλο menu που θέλει ο προγραμματιστής να έχει η εφαρμογή του. Σε αυτή την σελίδα αυτό που έχει χρησιμοποιηθεί ονομάζεται searchmenu. Αυτό που κάνει στη συνέχεια η συνάρτηση αυτή είναι όταν έχει πατηθεί το κουμπί αναζήτησης από το χρήστη και πληκτρολογεί το όνομα της αναζήτησης να στέλνει στην συνάρτηση της αναζήτησης την processsearch το κείμενο του χρήστη και να τρέχει ουσιαστικά αυτή. Η επόμενη συνάρτηση που έχει χρησιμοποιηθεί σε αυτή την σελίδα είναι η onOptionsItemSelected. Αυτή η συνάρτηση ουσιαστικά ενεργοποιεί τα στοιχεία που είναι πάνω στο toolbar. Σε αυτή την σελίδα έχουμε μόνο το logout. Η συνάρτηση παίρνει το id του κουμπιού που πατήθηκε και κάνει κάποια ενέργεια. Επειδή όπως αναφέρθηκε και πριν υπάρχει μόνο το logout σε αυτή την περίπτωση και με τις εντολές που υπάρχουν μέσα σε αυτό το στοιχείο γίνεται logout και ο χρήστης επιστρέφει στην αρχική σελίδα της εφαρμογής.

5.4.12 Movies

Επόμενη σελίδα που θα αναλυθεί είναι αυτή με το όνομα Movies. Αυτή η σελίδα ανοίγει όταν ο χρήστης πατήσει το κουμπί που λέει Movies όταν συνδεθεί στην εφαρμογή. Αρχικά και στην επόμενη εικόνα υπάρχουν οι βιβλιοθήκες οι οποίες χρησιμοποιήθηκαν για αυτή την σελίδα.

```
package com.example.telikiergasia;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.widget.SearchView;
import android.widget.TextView;
import android.widget.Toast;

import com.bumptech.glide.Glide;
import com.firebase.ui.database.FirebaseRecyclerOptions;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.List;

import de.hdodenhof.circleimageview.CircleImageView;
```

FIGURE 86 : ΒΙΒΛΙΟΘΗΚΕΣ

Οι βιβλιοθήκες που χρησιμοποιήθηκαν σε αυτήν την σελίδα είναι οι παραπάνω. Επόμενο βήμα είναι η δήλωση των μεταβλητών των οποίων χρησιμοποιήθηκαν στη σελίδα. Οι μεταβλητές αυτές φαίνονται στην παρακάτω εικόνα.

```
RecyclerView recyclerView;
AdapterMovies adapterMovies;
List<ModelMovies> movieList=new ArrayList<>();
String name,url;
CircleImageView profile_image;
TextView username;
FirebaseUser firebaseUser;
DatabaseReference databaseReference;
```

FIGURE 87 : ΜΕΤΑΒΛΗΤΕΣ

Σε αυτή την εικόνα φαίνεται ποιες μεταβλητές χρησιμοποιούνται στη σελίδα. Η πρώτη μεταβλητή που χρησιμοποιείται στη σελίδα είναι τύπου RecyclerView και ονομάζεται recyclerview. Η δεύτερη μεταβλητή που χρησιμοποιείται είναι τύπου AdapterMovies και ονομάζεται adapterMovies. Η Τρίτη μεταβλητή είναι μία λίστα από Movies και ονομάζεται movieList. Επόμενη μεταβλητή είναι τύπου String και ονομάζεται name. Επίσης υπάρχει και άλλη μία μεταβλητή τύπου String η οποία ονομάζεται url. Άλλη μία μεταβλητή είναι τύπου CircleImageView και ονομάζεται profile_image.

Ακόμα υπάρχει και μία μεταβλητή τύπου Textview η οποία ονομάζεται username. Προτελευταία μεταβλητή είναι τύπου FirebaseUser και ονομάζεται firebaseuser. Τελευταία μεταβλητή είναι τύπου DatabaseReference και ονομάζεται databasereference.

Σχεδόν πάντα επόμενο βήμα είναι η αρχικοποίηση των μεταβλητών. Αυτό φαίνεται και στην επόμενη εικόνα.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_movies);
    recyclerView = findViewById(R.id.recyclerviewMovies);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));
    FirebaseRecyclerOptions<ModelMovies> options =
            new FirebaseRecyclerOptions.Builder<ModelMovies>()
                    .setQuery(FirebaseDatabase.getInstance().getReference().child("Movies"), ModelMovies.class)
                    .build();
    profile_image=findViewById(R.id.imageView);
    username=findViewById(R.id.username_arxiki);
    Intent intent=getIntent();
    String item;
    item=intent.getExtras().getString("boolean");


    firebaseUser= FirebaseAuth.getInstance().getCurrentUser();
    databaseReference= FirebaseDatabase.getInstance().getReference("users").child(firebaseUser.getUid());
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
```

FIGURE 88 : ΑΡΧΙΚΟΠΟΙΗΣΗ ΜΕΤΑΒΛΗΤΩΝ

Όπως φαίνεται στην παραπάνω εικόνα οι μεταβλητές που έχουν χρησιμοποιηθεί και στο layout της σελίδας είναι τρεις. Αυτές είναι οι recyclerview με το όνομα recyclerviewMovies ,η profile_image με το όνομα imageview και η username με το όνομα username_arxiki. Επίσης δημιουργείται και μία μεταβλητή τύπου String η οποία ονομάζεται item και παίρνει την τιμή που έχει σταλθεί με την ετικέτα Boolean.

Δύο πράγματα τα οποία συμβαίνουν ακόμα σε αυτό το σημείο του κώδικα είναι η αρχικοποίηση του toolbar καθώς επίσης και η φόρτωση των δεδομένων από τη βάση με την εντολή FirebaseRecyclerOptions. Στη συνέχεια και την επόμενη εικόνα φαίνεται πως φορτώνεται το username και η εικόνα του χρήστη στη σελίδα αλλά και το πώς γεμίζει η σελίδα με τα ονόματα των ταινιών.

```
databaseReference.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        if (item.equals("false"))
        {
            User user=snapshot.getValue(User.class);
            username.setText(user.getUsername());
            Glide.with(Movies.this).load(user.getImageurl()).into(profile_image);
            Toast.makeText(Movies.this, item, Toast.LENGTH_SHORT).show();
        }
        if (item.equals("true"))
        {
            Intent intent=  getIntent();
            String img= intent.getExtras().getString("image");
            String username_google=intent.getExtras().getString("username");
            username.setText(username_google);
            Glide.with(Movies.this).load(img).into(profile_image);
        }
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }
});
// adapterMovies=new AdapterMovies(options);
//recyclerView.setAdapter(adapterMovies);

adapterMovies=new AdapterMovies(options,getApplicationContext());
recyclerView.setAdapter(adapterMovies);
```

FIGURE 89 : USERNAME ΚΑΙ ΕΙΚΟΝΑ

Στην παραπάνω εικόνα εύκολα διαπιστώνετε ότι μέσα στη συνάρτηση onDataChanged ο κώδικας βασίζεται σε μεγάλο βαθμό στην τιμή που έχει η μεταβλητή item. Αν η μεταβλητή είναι false τότε δημιουργείται μία μεταβλητή τύπου User η οποία ονομάζεται user και παίρνει όλα τα δεδομένα που έχει δώσει ο χρήστης. Στη συνέχεια γράφει το username του χρήστη στο textview και τοποθετεί την εικόνα του στο σημείο που δείχνει η εντολή Glide.

Στην περίπτωση που η τιμή της μεταβλητής item είναι true τότε δημιουργούνται δύο νέες μεταβλητές τύπου String οι οποίες ονομάζονται img και username_google οι οποίες αντίστοιχα παίρνουν τις τιμές που έχουν οι ετικέτες image και username. Στη συνέχεια γράφει το username του χρήστη στο textview και τοποθετεί την εικόνα του στο σημείο που δείχνει η εντολή Glide. Οι δύο τελευταίες εντολές που εμφανίζονται στην εικόνα γεμίζει την σελίδα με όλες τις ταινίες που υπάρχουν στην εφαρμογή.

Τελευταίο μέρος της ανάλυση αυτής της σελίδας είναι οι συναρτήσεις που έχουν χρησιμοποιηθεί. Αυτές φαίνονται στις παρακάτω εικόνες.

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.searchmenu, menu);
    MenuItem item = menu.findItem(R.id.search);
    SearchView searchView = (SearchView) item.getActionView();
    searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String s) {
            processsearch(s);
            return false;
        }

        @Override
        public boolean onQueryTextChange(String s) {
            processsearch(s);
            return false;
        }
    });
    return super.onCreateOptionsMenu(menu);

}
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    switch (item.getItemId()){
        case R.id.logout:
            FirebaseAuth.getInstance().signOut();
            startActivity(new Intent(Movies.this,MainActivity.class));

            return true;
    }
    return false;
}
```

FIGURE 90 : ΣΥΝΑΡΤΗΣΗ 21

```
private void processsearch(String s)
{
    FirebaseRecyclerOptions<ModelMovies> options =
            new FirebaseRecyclerOptions.Builder<ModelMovies>()
                    .setQuery(FirebaseDatabase.getInstance().getReference().child("Movies").orderByChild("name").startAt(s).endAt(s+"\uf8ff"), ModelMovies.class)
                    .build();
        adapterMovies = new AdapterMovies(options,getApplicationContext());
    adapterMovies.startListening();
    recyclerView.setAdapter(adapterMovies);
}

@Override
protected void onStart() {
    super.onStart();
    adapterMovies.startListening();
}
@Override
protected void onStop() {
    super.onStop();
    adapterMovies.stopListening();
}
```

FIGURE 91 : ΣΥΝΑΡΤΗΣΗ 22

Η πρώτη συνάρτηση που εμφανίζεται στις εικόνες είναι η onCreateOptionsMenu . Σε αυτή τη συνάρτηση ο προγραμματιστής επιλέγει ένα menu από αυτά που έχει δημιουργήσει. Αυτό που θα επιλέξει το βλέπει ο χρήστης της εφαρμογής ψηλά επάνω. Επίσης η συνάρτηση αυτή κάθε φορά που ο χρήστης πληκτρολογεί κάτι για αναζήτηση η συνάρτηση αυτή στέλνει την λέξη στη συνάρτηση της αναζήτησης processsearch η οποία θα αναλυθεί παρακάτω. Δεύτερη συνάρτηση είναι η onOptionsItemSelected η οποία υλοποιεί κώδικα για κάθε ένα item που έχει το menu. Σε αυτή την εφαρμογή υπάρχει μόνο το logout. Έτσι αυτό που κάνει ο κώδικας είναι τα εξής:

- Βρίσκει το id του item
- Διαλέγει στη συνέχεια το item και

- Υλοποιεί τον κώδικα

Σε αυτή την περίπτωση ο κώδικας που πρέπει να υλοποιηθεί κάνει δύο πράγματα. Το πρώτο είναι ότι κάνει logout το χρήστη από την εφαρμογή και το δεύτερο ότι γυρίζει τον χρήστη στη πρώτη σελίδα της εφαρμογής.

Επόμενη συνάρτηση είναι η processsearch. Σε αυτή τη συνάρτηση υλοποιείται η αναζήτηση. Όλη η υλοποίηση γίνεται με την εντολή που υπάρχει στην πρώτη γραμμή και ξεκινάει με FirebaseRecyclerOptions. Πιο συγκεκριμένα η αναζήτηση υλοποιείται στην εντολή .setQuery στην οποία ο προγραμματιστής επιλέγει βάση με τι θα γίνεται η αναζήτηση. Σε αυτή την εφαρμογή η αναζήτηση γίνεται βάση του ονόματος.

5.4.13 Full Screen Movies

Επόμενη σελίδα που θα αναλυθεί είναι η FullScrennMovies. Σε αυτή την σελίδα τρέχει το video της ταινίας και φαίνεται και η περιγραφή της εικόνας. Αρχικά η επόμενη εικόνα δείχνει τις βιβλιοθήκες οι οποίες χρησιμοποιήθηκαν σε αυτή την σελίδα.

```
package com.example.telikiergasia;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.widget.TextView;

import com.google.android.exoplayer2.ExoPlayerFactory;
import com.google.android.exoplayer2.SimpleExoPlayer;
import com.google.android.exoplayer2.source.MediaSource;
import com.google.android.exoplayer2.source.ProgressiveMediaSource;
import com.google.android.exoplayer2.ui.SimpleExoPlayerView;
import com.google.android.exoplayer2.upstream.DataSource;
import com.google.android.exoplayer2.upstream.DefaultHttpDataSourceFactory;
import com.google.android.exoplayer2.util.Util;
```

FIGURE 92 : ΒΙΒΛΙΟΘΗΚΕΣ

Αυτές είναι οι βιβλιοθήκες που χρησιμοποιήθηκαν σε αυτή την σελίδα. Με κάποιες από αυτές τις βιβλιοθήκες γίνεται και η αναπαραγωγή του video της ταινίας. Αυτές οι βιβλιοθήκες είναι αυτές οι οποίες έχουν την έκφραση .exoplayer2. Στη συνέχεια υπάρχουν οι μεταβλητές οι οποίες χρησιμοποιήθηκαν σε αυτή τη σελίδα. Αυτές παρουσιάζονται στην παρακάτω εικόνα.

```
private SimpleExoPlayer player;
private SimpleExoPlayerView playerView;
TextView textView;
private String url;
private boolean playwhenready=false;
private int currentWindow=0;
private long playbackposition=0;
```

FIGURE 93 : ΜΕΤΑΒΛΗΤΕΣ

Οι μεταβλητές οι οποίες έχουν χρησιμοποιηθεί είναι εφτά στον αριθμό. Οι πρώτες δύο είναι για τη αναπαραγωγή του video της ταινίας. Η πρώτη είναι τύπου SimpleExoPlayer και η δεύτερη είναι τύπου SimpleExoPLayerView και ονομάζονται player και playerView. Η επόμενη μεταβλητή είναι τύπου TextView και ονομάζεται textView. Επόμενη μεταβλητή είναι η url και είναι τύπου String. Τέλος υπάρχουν τρεις μεταβλητές οι οποίες είναι και αρχικοποιημένες. Η πρώτη είναι μία τύπου Boolean και ονομάζεται playerwhenready και έχει την τιμή false. Οι επόμενες δύο μεταβλητές έχουν την τιμή μηδέν. Η πρώτη μεταβλητή ονομάζεται currentWindow και είναι τύπου int, ενώ η δεύτερη μεταβλητή είναι η playbackposition και είναι τύπου long. Στην επόμενη εικόνα φαίνονται οι αρχικοποιήσεις των μεταβλητών.

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_full_screen_movies);
playerView=findViewById(R.id.exo_player_view);
textView=findViewById(R.id.tv_fullscreen);
Intent intent=getIntent();
url=intent.getExtras().getString("image");
String title=intent.getExtras().getString("description");
textView.setText(title);
```

FIGURE 94 : ΑΡΧΙΚΟΠΟΙΗΣΗ ΜΕΤΑΒΛΗΤΩΝ

Στην παραπάνω εικόνα αυτό που απεικονίζεται είναι η αρχικοποίηση των μεταβλητών που υπάρχουν και στο layout. Επίσης υπάρχουν και τα στοιχεία που έχουν σταλθεί από την προηγούμενη σελίδα τα οποία και τα «τραβάει» η σελίδα αυτή. Οι μεταβλητές οι οποίες υπάρχουν και στο layout της σελίδας αυτής είναι οι πρώτες δύο που φαίνονται στην εικόνα δηλαδή η playerView και η textView. Τα ονόματα τα οποία έχουν στο layout είναι exo_player_view και tv_fullscreen αντίστοιχα. Οι επόμενες εντολές δημιουργούν μία μεταβλητή τύπου Intent η οποία ονομάζεται intent και με αυτή τη μεταβλητή παίρνουν τις τιμές τις οποίες έχουν οι μεταβλητές με ετικέτες image και description αντίστοιχα. Τέλος με την τελευταία εντολή γράφει την τιμή της μεταβλητής με ετικέτα description στην οθόνη. Αυτό που έχει μείνει για να τελειώσει η ανάλυση αυτής της σελίδας είναι οι συναρτήσεις. Αυτές απεικονίζονται παρακάτω.

```
private MediaSource buildMediaSource(Uri uri){
    DataSource.Factory datasourcefactory=new DefaultHttpDataSourceFactory("video");
    return new ProgressiveMediaSource.Factory(datasourcefactory).createMediaSource(uri);

}
private void initializeplayer()
{
    player= ExoPlayerFactory.newSimpleInstance(this);
    playerView.setPlayer(player);
    Uri uri=Uri.parse(url);
    MediaSource mediaSource=buildMediaSource(uri);
    player.setPlayWhenReady(playwhenready);
    player.seekTo(currentWindow,playbackposition);
    player.prepare(mediaSource,false,false);
}
@Override
protected void onStart() {
    super.onStart();
    if(Util.SDK_INT>=26)
    {
        initializeplayer();
    }
}
```

FIGURE 95 : ΣΥΝΑΡΤΗΣΗ 23

```
protected void onRestart() {
    super.onRestart();
    if(Util.SDK_INT>=26 || player==null)
    {
        //initializeplayer();
    }
}

@Override
protected void onPause() {
    super.onPause();
    if(Util.SDK_INT>26)
    {
        releasePlayer();
    }
}

@Override
protected void onStop() {
    super.onStop();
    if(Util.SDK_INT>=26)
    {
        releasePlayer();
    }
}

private void releasePlayer() {
    if (player!=null)
    {
        playwhenready=player.getPlayWhenReady();
        playbackposition=player.getCurrentPosition();
        currentWindow=player.getCurrentWindowIndex();
        player=null;
    }
}
```

FIGURE 96 : ΠΡΟΣΘΕΤΕΣ ΣΥΝΑΡΤΗΣΕΙΣ

Οι παραπάνω συναρτήσεις έχουν χρησιμοποιηθεί για να λειτουργήσει αυτή η σελίδα. Οι συναρτήσεις αυτές είναι εφτά στον αριθμό. Πιο συγκεκριμένα είναι οι εξής :

- buildMediaSource
- initializeplayer
- onStart
- onRestart
- onPause
- onStop
- releasePlayer

Η πρώτη συνάρτηση είναι αυτή που τραβάει το video και δημιουργεί μία μεταβλητή τύπου DataSource. Η δεύτερη συνάρτηση είναι αυτή που ουσιαστικά τοποθετεί το video στην θέση που ορίζεται

μέσα στη συνάρτηση . Αυτό συμβαίνει με τη τέταρτη εντολή της συνάρτησης. Οι επόμενες τέσσερις συναρτήσεις αυτό που ουσιαστικά κάνουν είναι να εκτελούν τις λειτουργίες start, restart, pause και stop.

Η τελευταία συνάρτηση υλοποιεί την δυνατότητα κάποιος να μπορεί να σταματήσει το video που παίζει. Το video θα ξεκινήσει από τη αρχή και όχι από εκεί που έμεινε όταν πατήθηκε το κουμπί αυτό.

5.4.14 Full Screen

Η τελευταία κλάση που θα αναλυθεί είναι αυτή που ονομάζεται FullScreen. Στην επόμενη εικόνα φαίνονται οι βιβλιοθήκες που χρησιμοποιήθηκαν.

```
package com.example.telikiergasia;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.widget.ImageView;
import android.widget.TextView;

import com.bumptech.glide.Glide;
```

FIGURE 97 : ΒΙΒΛΙΟΘΗΚΕΣ

Αυτές είναι οι βιβλιοθήκες που χρησιμοποιήθηκαν για τη σελίδα αυτή. Επόμενο στοιχείο που θα πρέπει να αναλυθεί είναι οι μεταβλητές που χρησιμοποιήθηκαν σε αυτή τη σελίδα. Αυτές φαίνονται παρακάτω.

```
TextView a_description;
ImageView imageView;
```

FIGURE 98 : ΜΕΤΑΒΛΗΤΕΣ

Η πρώτη μεταβλητή είναι η a_description και είναι τύπου TextView. Η επόμενη μεταβλητή ονομάζεται imageView και είναι τύπου ImageView. Τέλος για αυτή την σελίδα θα αναλυθούν οι αρχικοποιήσεις των μεταβλητών και θα αναλυθούν επίσης οι τιμές που παίρνει κάθε στοιχείο στο layout. Αυτά απεικονίζονται παρακάτω.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_full_screen);
    a_description=findViewById(R.id.a_description);
    imageView=findViewById(R.id.exo_player_view);
    Intent intent = getIntent();
    String img= intent.getExtras().getString("image");



    Glide.with(getApplicationContext()).load(img).into(imageView);
    String description=intent.getExtras().getString("description");
    a_description.setText(description);
}
```

FIGURE 99 : ΑΡΧΙΚΟΠΟΙΗΣΗ ΜΕΤΑΒΛΗΤΩΝ

Σε αυτή την εικόνα υπάρχουν οι αρχικοποιήσεις των μεταβλητών. Αυτό συμβαίνει με την εντολή findViewById. Η μεταβλητή a_description που έχει δηλωθεί παραπάνω αρχικοποιείται στο layout με το ίδιο όνομα δηλαδή a_description. Η επόμενη μεταβλητή η οποία είναι η imageView έχει το όνομα exo_player_view στο layout όπως φαίνεται και στην εντολή findViewById. Στη συνέχεια γίνεται η χρήση της εντολής Intent για να μπορεί να πάρει η μεταβλητή img την τιμή της ετικέτας image και η μεταβλητή description να πάρει την τιμή με την ετικέτα description. Τέλος με την εντολή Glide τοποθετείται η εικόνα στο σημείο της οθόνης που ορίζει το layout και με την τελευταία εντολή φαίνεται στην οθόνη το βιογραφικό του κάθε ηθοποιού.

6.Πίνακας Εικόνων

7. Συμπέρασμα

Σε αυτή την ενότητα θα γίνει η ανάλυση των συμπερασμάτων για την εργασία αυτή. Το θέμα της εργασίας ήταν να δημιουργηθεί μία εφαρμογή Android όπου ο χρήστης θα μπορεί να διαβάσει βιογραφικά κάποιων δημοφιλών ηθοποιών αλλά και θα μπορεί να δει τα trailer κάποιων από τις ταινίες που έχουν παίξει οι ηθοποιοί της εφαρμογής. Για την δημιουργία της εφαρμογής χρησιμοποιήθηκαν τα εξής τρία εργαλεία:

- Android Studio
- Java
- Firebase

Σε αυτό το σημείο θα διατυπωθούν τα συμπεράσματα ξεχωριστά για το καθένα από τα τρία εργαλεία που χρησιμοποιήθηκαν για την εργασία.

- Android Studio: Ένα από τα πιο χρήσιμα εργαλεία που μπορεί να έχει ένας προγραμματιστής για την δημιουργία εφαρμογών Android. Στην εργασία αυτή ανακαλύφθηκε ότι το συγκεκριμένο εργαλείο έχει πολλές δυνατότητες και μπορούν να γίνουν πολλά και ενδιαφέροντα πράγματα σε όλα τα επίπεδα της εφαρμογής. Στην εργασία αυτή χρησιμοποιήθηκαν τα layouts, τα activities όπως και άλλα πολλά αρχεία.
  Τα αρχεία αυτά που προαναφέρθηκαν εξετάστηκαν και ανακαλύφθηκαν σε πολύ μεγάλο βαθμό. Τέλος το συμπέρασμα για τα στοιχεία του Android Studio αλλά και για το Android Studio είναι ότι υπάρχουν πολύ μεγάλες δυνατότητες με το συγκεκριμένο πρόγραμμα.
- Java: Για την εργασία αυτή ως γλώσσα προγραμματισμού χρησιμοποιήθηκε η Java. Η χρησιμοποίηση της ήταν πολύ σημαντική καθώς για την δημιουργία της εφαρμογής χρησιμοποιήθηκαν πολλές νέες τεχνολογίες και μέθοδοι. Γενικά το συμπέρασμα για την γλώσσα προγραμματισμού Java σε σχέση με το Android είναι ότι μπορεί να είναι χρήσιμη και πολύ σημαντική για τις εφαρμογές αλλά ταυτόχρονα δίνει

και μεγάλες δυνατότητες στη δημιουργία των εφαρμογών • Firebase: Η Firebase είναι μια τεχνολογία που χρησιμοποιήθηκε για τη βάση δεδομένων.

Αυτά τα στοιχεία της Firebase που χρησιμοποιήθηκαν για την εργασία είναι το Authentication, το Storage και η Realtime Database. Η χρήση αυτών των στοιχείων έγινε σε μεγάλο βαθμό καθώς είναι απαραίτητα για την χρήση της βάσης σε ότι έχει να κάνει με τη σύνδεση των χρηστών (sign in, sign-out), τις φωτογραφίες της εφαρμογής

(χρήστες, ταινίες και ηθοποιοί) αλλά και τα βίντεο της εφαρμογής. Το συμπέρασμα για την Firebase ως εργαλείο σε εφαρμογές είναι ότι έχει μεγάλες δυνατότητες για την εκπλήρωση όλων όσων χρειάζονται και είναι εύκολη στη χρήση της.

8. Βιβλιογραφία

1. https://www.google.com/search?q=firebase+png&tbm=isch&chips=q:firebase+png,online_chips:transparent+png:mnL7h_zqFfI%3D&rlz=1C1PNBB_enGR990GR990&hl=en&sa=X&ved=2ahUKEwiz8oXl2J32AhVkgc4BHRurCFQQ4lYoDXoECAEQMw&biw=2543&bih=937

2. https://www.google.com/search?q=android+studio+logo+png&tbm=isch&ved=2ahUKEwif36uf2Z32AhXpvKQKHdt-AcMQ2-cCegQIABAA&oq=android+studio+logo+png&gs_lcp=CgNpbWcQAzIFCAAQgAQyBQgAEIAEMgYIABAHEB4yBggAEAUQHjIGCAAQCBAeOgQIABBDOggIABAHEAUQHjoICAAQCBAHEB5QxRYrjhgtT1oAXAAeACAAXOIAeALkgEEMTMuM5gBAKABAaoBC2d3cy13aXotaW1nwAEB&sclient=img&ei=-EgaYp-DA-n5kgXb_YWYDA&bih=937&biw=2560&rlz=1C1PNBB_enGR990GR990#imgrc=uZD3B96uI1WybM

3. https://www.google.com/search?q=java+logo+png&rlz=1C1PNBB_enGR990GR990&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjA8_6d2Z32AhW8SvEDHYcPC84Q_AUoAXoECAEQAw&biw=2560&bih=937&dpr=1#imgrc=D9-YK7Lwieyt6M

4. https://developer.android.com/studio/intro

5. https://firebase.google.com/

6. https://en.wikipedia.org/wiki/Java_(programming_language)

7. https://en.wikipedia.org/wiki/Firebase

8. https://en.wikipedia.org/wiki/Firebase

9. https://en.wikipedia.org/wiki/Android_Studio